Regression under a Modern Optimization Lens

by

Angela King

B.Sc. Mathematics, McGill University (2010)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

Signature redacted

Author .....
Sloan School of Management
May 11, 2015

Signature redacted

Certified by .....
Dimitris Bertsimas
Boeing Leader for Global Operations Professor
Co-Director, Operations Research Center
Thesis Supervisor

Signature redacted

Accepted by .....
Patrick Jaillet
Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science
Co-Director, Operations Research Center

Regression under a Modern Optimization Lens

by

Angela King

Submitted to the Sloan School of Management
on May 11, 2015, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

In the last twenty-five years (1990-2014), algorithmic advances in integer optimization combined with hardware improvements have resulted in an astonishing 200 billion factor speedup in solving mixed integer optimization (MIO) problems ([16], [85], [104]). The common mindset of MIO as theoretically elegant but practically irrelevant is no longer justified. In this thesis, we propose a methodology for regression modeling that is based on optimization techniques and centered around MIO.

In Part I we propose a method to select a subset of variables to include in a linear regression model using continuous and integer optimization. Despite the natural formulation of subset selection as an optimization problem with an ℓ_0-norm constraint, current methods for subset selection do not attempt to use integer optimization to select the best subset. We show that, although this problem is non-convex and NP-hard, it can be practically solved for large scale problems. We numerically demonstrate that our approach outperforms other sparse learning procedures.

In Part II of the thesis, we build off of Part I to modify the objective function and include constraints that will produce linear regression models with other desirable properties, in addition to sparsity. We develop a unified framework based on MIO which aims to algorithmize the process of building a high-quality linear regression model. This is the only methodology we are aware of to construct models that imposes statistical properties simultaneously rather than sequentially.

Finally, we turn our attention to logistic regression modeling. It is the goal of Part III of the thesis to efficiently solve the mixed integer convex optimization problem of logistic regression with cardinality constraints to provable optimality. We develop a tailored algorithm to solve this challenging problem and demonstrate its speed and performance. We then show how this method can be used within the framework of Part II, thereby also creating an algorithmic approach to fitting high-quality logistic regression models.

In each part of the thesis, we illustrate the effectiveness of our proposed approach on both real and synthetic datasets.

Thesis Supervisor: Dimitris Bertsimas
Title: Boeing Leader for Global Operations Professor
Co-Director, Operations Research Center

Acknowledgments

I would like to acknowledge and thank my supervisor, Dimitris Bertimas, for initially piquing my curiosity in problems at the intersection of discrete optimization and statistics, and for guiding the research projects that ultimately formed this thesis. His faith both in me and in the work we have done together have been invaluable to my growth as a researcher. Chapter 2 of this thesis is joint work with Rahul Mazumder in addition to Dimitris. Rahul's tireless work ethic and optimistic attitude have made him an irreplaceable coauthor. Many thanks to him for generously taking the time to teach me and share his insights into the field of statistics.

I would also like to thank the other ORC faculty who have played a part in my graduate education. I would especially like to acknowledge David Gamarnik and Devavrat Shah for supporting me by serving as part of my thesis committee and Itai Ashlagi and Retsef Levi for guiding my initial foray into the murky world of academic research.

The community of students at the ORC has been far and away the most rewarding part of being a graduate student at MIT. The intellectual energy buzzing at the ORC has encouraged and inspired me, and my day-to-day for the past five years has been shaped by the ORC students. In particular, I've pored over problem sets and braved quals with the students in my cohort – Ben, David Z, Fernanda, Kris, Maxime, Nathan, and Will – and I couldn't have made it through the first two years without them. Over the years, I've been lucky to have Andre, Alex W, David F, and Ross as officemates. They have been there every day to say hello, listen to my thoughts, and take a break. And the students in the year above me took me under their wing from the very beginning: Adam, Allison, Andre, Florin, Gonzalo, Ross, and Vishal. In addition to daily research, I've had the pleasure of working on two exciting and impactful teaching projects with many of my fellow ORC students. Working on the IAP software class with my fellow instructors Chiwei, Clark, Evan, He, Iain, Jerry, Joey, John, Miles, Ross, and Vishal has pushed me to become a better teacher and presenter, and has been a rewarding way to give back to the ORC community. I've also gotten to think deeply about pedagogy with Allison, Iain, John, Nataly, and Velibor as we built a full-fledged online

analytics course along with Dimitris and watched it blossom into a class that thousands of students around the world have already taken.

On top of being brilliant colleagues and thoughtful collaborators, so many of the ORCers have become fantastic friends and I feel privileged to have gotten to know nearly ten years' worth of amazing ORC students during my time at MIT. There have been countless meals shared, trips taken, successes celebrated, drinks poured, challenges commiserated, and congratulations exchanged. Thank you all for five years of friendship and laughter. A special thank you to Fernanda and Kris, who have literally and figuratively been to the bottom of the world with me and back. I would not have gotten to this point today without all of their help and encouragement, and can't thank them enough for the memories.

Some of my strongest support has come from outside the ORC. Many of my friends are scattered across the country and the world, but both those near and far have always been close at heart, and have been there to remind me that there is a world beyond the grad school bubble. I am grateful to my parents for ending almost every phone call with a reminder to not work too hard, and to my sister for indulging my silly side. Finally, I'd like to thank Doug for being a constant source of love, support, and happiness. Being with him has made me want to be the best version of myself, and makes me smile every day.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of this thesis is to illustrate the applicability of discrete optimization methods to statistical problems. Mixed integer optimization (MIO) has typically not been used in statistical contexts. However, when fitting statistical models, modelers often have discrete goals in mind, and often use convex approximations to their true objectives, or heuristics, rather than approach the problem via MIO. It is our aim to show that MIO is a practical tool for statistical modeling. We study this in the context of regression modeling. In particular, we address best subset selection in linear and logistic regression using MIO, and develop an MIO-based framework for algorithmizing the process of building linear and logistic regression models. We first give a brief overview of MIO, including the simply astonishing advances it has enjoyed in the last twenty-five years, and then describe our contributions in each of the three main chapters of the thesis.

The general form of a Mixed Integer Optimization (MIO) problem is as follows:

$$\begin{aligned} \min \quad & h(\boldsymbol{\alpha}) \\ s.t. \quad & g_j(\boldsymbol{\alpha}) \leq \mathbf{0} \quad \forall j \in J \\ & \alpha_i \in \{0,1\}, \quad \forall i \in \mathcal{I} \\ & \alpha_j \in \mathbb{R}, \quad \forall j \notin \mathcal{I}, \end{aligned}$$

where $\mathbb{R}$ denotes the real numbers, the symbol $\leq$ denotes element-wise inequalities and we optimize over $\boldsymbol{\alpha} \in \mathbb{R}^m$ containing both discrete $(\alpha_i, i \in \mathcal{I})$ and continuous $(\alpha_i, i \notin \mathcal{I})$ variables, with $\mathcal{I} \subset \{1, \ldots, m\}$.

Types of MIO problems include mixed integer linear optimization (MILO) problems (h, g_j are linear functions), mixed integer quadratic optimization (MIQO) problems (h is quadratic, g_j are linear functions), and mixed integer nonlinear optimization (MINLO) problems (h and g_j are continuously differentiable nonlinear functions). When $\mathcal{I} = \emptyset$, MILO problems reduce to linear optimization (LO) problems, MIQO problems reduce to quadratic optimization (QO) problems, and MINLO problems reduce to nonlinear optimization (NLO) problems.

In the last twenty-five years (1991-2014) the computational power of MIO solvers has increased at an astonishing rate. In [16], to measure the speedup of MIO solvers, the same set of MILO problems were tested on the same computers using twelve consecutive versions of CPLEX and version-on-version speedups were reported. The versions tested ranged from CPLEX 1.2, released in 1991 to CPLEX 11, released in 2007. Each version released in these years produced a speed improvement on the previous version, leading to a total speedup factor of more than 29,000 between the first and last version tested (see [16], [85] for details). Gurobi 1.0, a MIO solver which was first released in 2009, was measured to have similar performance to CPLEX 11. Version-on-version speed comparisons of successive Gurobi releases have shown a speedup factor of more than 20 between Gurobi 5.5, released in 2013, and Gurobi 1.0 ([16], [85]). The combined machine-independent speedup factor in MIO solvers between 1991 and 2013 is 580,000. This impressive speedup factor is due to incorporating both theoretical and practical advances into MIO solvers. Cutting plane theory, disjunctive programming for branching rules, improved heuristic methods, techniques for preprocessing MIOs, using linear optimization as a black box to be called by MIO solvers, and improved linear optimization methods have all contributed greatly to the speed improvements in MIO solvers [16].

In addition, the past twenty years have also brought dramatic improvements in hardware. Figure 1-1 shows the exponentially increasing speed of supercomputers over the past twenty

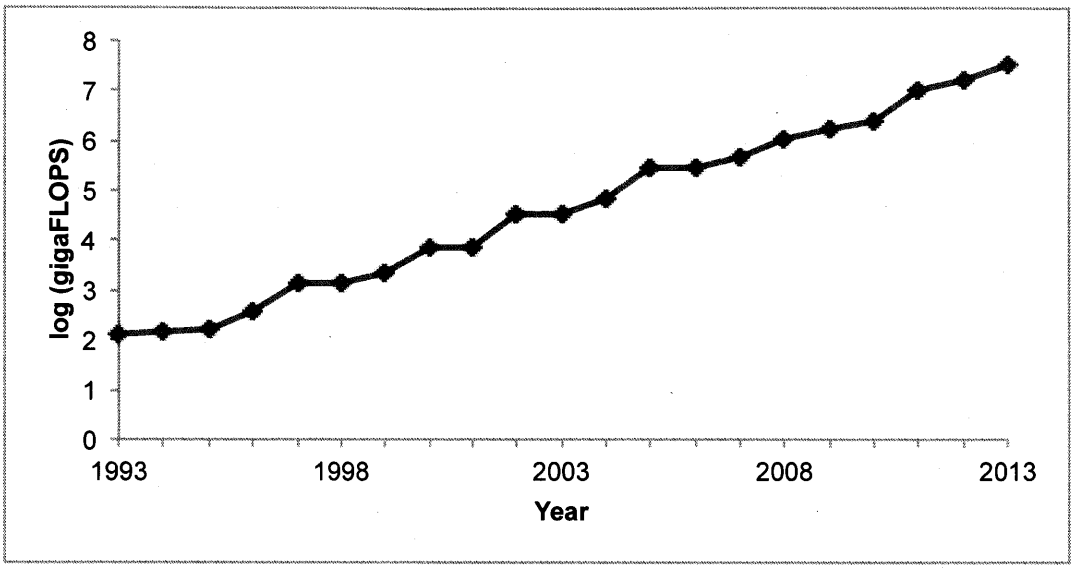

Figure 1-1: Log of Peak Supercomputer Speed from 1993–2013.

years, measured in billion floating point operations per second [104]. The hardware speedup from 1993 to 2013 is approximately $10^{5.5} \sim 320,000$. When both hardware and software improvements are considered, the overall speedup is approximately 200 billion! Note that the speedup factors cited here refer to MILO problems. The speedup factors for MIQO problems are similar. MIO solvers provide both feasible solutions as well as lower bounds to the optimal value. As the MIO solver progresses towards the optimal solution, the lower bounds improve and provide an increasingly better guarantee of suboptimality, which is especially useful if the MIO solver is stopped before reaching the global optimum. In contrast, heuristic methods do not provide such a certificate of suboptimality.

The belief that MIO approaches to problems in statistics are not practically relevant was formed in the 1970s and 1980s and it was at the time justified. Given the astonishing speedup of MIO solvers and computer hardware in the last twenty-five years, the mindset of MIO as theoretically elegant but practically irrelevant is no longer justified. In this thesis, we provide empirical evidence of this fact in the context of the best subset selection problem and in the wider context of building a high-quality regression model.

In Section 2, we present a MIO approach for solving the classical best subset selection

problem of choosing k out of p features in linear regression given n observations. We develop a discrete extension of modern first order continuous optimization methods to find high quality feasible solutions that we use as warm starts to a MIO solver that finds provably optimal solutions. The resulting algorithm (a) provides a solution with a guarantee on its suboptimality even if we terminate the algorithm early, (b) can accommodate side constraints on the coefficients of the linear regression and (c) extends to finding best subset solutions for the least absolute deviation loss function. Using a wide variety of synthetic and real datasets, we demonstrate that our approach solves problems with n in the 1000s and p in the 100s in minutes to provable optimality, and finds near optimal solutions for n in the 100s and p in the 1000s in minutes. We also establish via numerical experiments that the MIO approach performs better than Lasso and other popularly used sparse learning procedures, in terms of achieving sparse solutions with good predictive power.

Linear regression models are traditionally built through trial and error in order to balance many competing goals such as predictive power, interpretability, significance, robustness to error in data, and sparsity, among others. Balancing these goals is a problem which lends itself naturally to a mixed integer quadratic optimization (MIQO) approach, but has not been modeled this way due to the belief in the statistics community that MIQO is intractable for large scale problems. However, in the light of the hardware and software improvements in MIO, we tackle this problem explicitly via MIQO in Section 3. We present an MIQO-based algorithm for designing high-quality linear regression models that explicitly addresses various competing objectives, and demonstrate our algorithm's effectiveness on both real and synthetic datasets.

In Section 4, we turn our attention to logistic regression. We consider the same questions: how to choose the best subset of variables in a model, and more generally, how to best determine a high-quality logistic regression model. The logistic regression objective function is convex, and we can model the cardinality constraint, as in Section 2, via MIO. However, MIO solvers for general convex programs are not nearly as developed as MILO and MIQO solvers; while software exists, there is high variation in solver performance for

different problem instance families [19]. Additionally, the unconstrained logistic regression NLO cannot generally be solved analytically in closed form.

These differences between linear and logistic regression lead to the main challenge of this part of the thesis: developing a method to efficiently solve the mixed integer convex optimization problem of logistic regression with cardinality constraints to provable optimality. We develop a tailored algorithm to do so where we combine (a) outer approximation techniques in mixed integer nonlinear optimization with (b) our discrete first order heuristic from Section 2 and (c) lazy constraint callbacks, a feature of modern optimization solvers. We demonstrate that our method outperforms existing MINLO software and apply this method within the algorithmic framework developed in Section 3. By doing so, we extend the algorithmic approach to linear regression to the logistic regression case. Again, we illustrate the success of this methodology using real and synthetic data.

Finally, we give concluding remarks in Section 5.

Chapter 2

Best Subset Selection in Linear Regression

2.1 Introduction

We consider the linear regression model with response vector $\mathbf{y}_{n\times 1}$, model matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathbb{R}^{n\times p}$, regression coefficients $\boldsymbol{\beta} \in \mathbb{R}^{p\times 1}$ and errors $\boldsymbol{\epsilon} \in \mathbb{R}^{n\times 1}$:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

We will assume that the columns of $\mathbf{X}$ have been standardized to have zero means and unit ℓ_2-norm. In many important classical and modern statistical applications, it is desirable to obtain a parsimonious fit to the data by finding the best k-feature fit to the response $\mathbf{y}$. Especially in the high-dimensional regime with $p \gg n$, in order to conduct statistically meaningful inference, it is desirable to assume that the true regression coefficient $\boldsymbol{\beta}$ is sparse or may be well approximated by a sparse vector. Quite naturally, the last few decades have seen a flurry of activity in estimating sparse linear models with good explanatory power. Central to this statistical task lies the best subset problem [82] with subset size k, which is

given by the following optimization problem:

$$\min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \ \text{ subject to } \ \|\boldsymbol{\beta}\|_0 \leq k, \tag{2.1}$$

where the ℓ_0 (pseudo)norm of a vector $\boldsymbol{\beta}$ counts the number of nonzeros in $\boldsymbol{\beta}$ and is given by $\|\boldsymbol{\beta}\|_0 = \sum_{i=1}^{p} 1(\beta_i \neq 0)$, where $1(\cdot)$ denotes the indicator function. The cardinality constraint makes Problem (2.1) NP-hard [84]. Indeed, state-of-the-art algorithms to solve Problem (2.1), as implemented in popular statistical packages, like `leaps` in `R`, do not scale to problem sizes larger than $p = 30$. Due to this reason, it is not surprising that the best subset problem has been widely dismissed as being *intractable* by the greater statistical community.

In this section we address Problem (2.1) using modern optimization methods, specifically mixed integer optimization (MIO) and a discrete extension of first order continuous optimization methods. Using a wide variety of synthetic and real datasets, we demonstrate that our approach solves problems with n in the 1000s and p in the 100s in minutes to provable optimality, and finds near optimal solutions for n in the 100s and p in the 1000s in minutes. To the best of our knowledge, this is the first time that MIO has been demonstrated to be a tractable solution method for Problem (2.1). We note that we use the term tractability not to mean the usual polynomial solvability for problems, but rather the ability to solve problems of realistic size in times that are appropriate for the applications we consider.

As there is a vast literature on the best subset problem, we next give a brief and selective overview of related approaches for the problem.

Brief Context and Background

To overcome the computational difficulties of the best subset problem, computationally tractable convex optimization based methods like Lasso [101, 29] have been proposed as a convex surrogate for Problem (2.1). For the linear regression problem, the Lagrangian form of Lasso solves

$$\min_{\boldsymbol{\beta}} \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{2.2}$$

where the ℓ_1 penalty on $\boldsymbol{\beta}$, i.e., $\|\boldsymbol{\beta}\|_1 = \sum_i |\beta_i|$ shrinks the coefficients towards zero and naturally produces a sparse solution by setting many coefficients to be exactly zero. There has been a substantial amount of impressive work on Lasso [44, 25, 14, 117, 55, 121, 36, 67, 80, 108, 102] in terms of algorithms and understanding of its theoretical properties—see for example the excellent books or surveys [21, 60, 102] and the references therein.

Indeed, Lasso enjoys several attractive statistical properties and has drawn a significant amount of attention from the statistics community as well as other closely related fields. Under various conditions on the model matrix $\mathbf{X}$ and $n, p, \boldsymbol{\beta}$ it can be shown that Lasso delivers a sparse model with good predictive performance [21, 60]. In order to perform exact variable selection, much stronger assumptions are required [21]. Sufficient conditions under which Lasso gives a sparse model with good predictive performance are the restricted eigenvalue conditions and compatibility conditions [21]. These involve statements about the range of the spectrum of sub-matrices of $\mathbf{X}$ and are difficult to verify, for a given data-matrix $\mathbf{X}$.

An important reason behind the popularity of Lasso is its computational feasibility and scalability to practical sized problems. Problem (3.2) is a convex quadratic optimization problem and there are several efficient solvers for it, see for example [86, 44, 50].

In spite of its favorable statistical properties, Lasso has several shortcomings. In the presence of noise and correlated variables, in order to deliver a model with good predictive accuracy, Lasso brings in a large number of nonzero coefficients (all of which are shrunk towards zero) including noise variables. Lasso leads to biased regression coefficient estimates, since the ℓ_1-norm penalizes both large and small coefficients uniformly. In contrast, if the best subset selection procedure decides to include a variable in the model, it brings it in without any shrinkage thereby draining the effect of its correlated surrogates. Upon increasing the degree of regularization, Lasso sets more coefficients to zero, but in the process ends up leaving out true predictors from the active set. Thus, as soon as certain sufficient regularity conditions on the data are violated, Lasso becomes suboptimal as (a) a variable selector and (b) in terms of delivering a model with good predictive performance.

The shortcomings of Lasso are also known in the statistical literature. In fact, there is a significant gap between what can be achieved via best subset selection and Lasso: this is supported by empirical (for small problem sizes, i.e., $p \leq 30$) and theoretical evidence, see for example, [90, 120, 78, 54, 118, 97] and the references therein.

To address the shortcomings, non-convex penalized regression is often used to "bridge" the gap between the convex ℓ_1 penalty and the combinatorial ℓ_0 penalty [78, 48, 46, 116, 117, 49, 123, 124, 119, 23]. Written in Lagrangian form, this gives rise to continuous non-convex optimization problems of the form:

$$\tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \sum_i p(|\beta_i|; \gamma; \lambda), \tag{2.3}$$

where $p(|\beta|; \gamma; \lambda)$ is a non-convex function in β with λ and γ denoting the degree of regularization and non-convexity, respectively. Typical examples of non-convex penalties include the minimax concave penalty (MCP), the smoothly clipped absolute deviation (SCAD), and ℓ_γ penalties (see for example, [48, 78, 124, 46]). There is strong statistical evidence indicating the usefulness of estimators obtained as minimizers of non-convex penalized problems (2.3) over Lasso see for example [118, 73, 116].

Problem (2.3) mainly leads to a family of continuous and non-convex optimization problems. Various effective nonlinear optimization based methods (see for example [124, 46, 23, 73, 116, 78] and the references therein) have been proposed in the literature to obtain good local minimizers to Problem (2.3). In particular [78] proposes Sparsenet, a coordinate-descent procedure to trace out a surface of local minimizers for Problem (2.3) for the MCP penalty using effective warm start procedures. None of the existing approaches for solving Problem (2.3), however, come with guarantees of how close the solutions are to the global minimum of Problem (2.3).

The Lagrangian version of (2.1) given by

$$\tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{i=1}^{p} 1(\beta_i \neq 0), \tag{2.4}$$

may be seen as a special case of (2.3). Note that, due to non-convexity, problems (2.4) and (2.1) are *not* equivalent. Problem (2.1) allows one to control the exact level of sparsity via the choice of k, unlike (2.4) where there is no clear correspondence between λ and k. Problem (2.4) is a discrete optimization problem unlike continuous optimization problems (2.3) arising from continuous non-convex penalties.

Insightful statistical properties of Problem (2.4) have been explored from a theoretical viewpoint in [118, 54, 55, 97]. [97] points out that (2.1) is preferable over (2.4) in terms of superior statistical properties of the resulting estimator. None of the aforementioned papers, however, discuss methods to obtain provably optimal solutions to problems (2.4) or (2.1), and to the best of our knowledge, computing optimal solutions to problems (2.4) and (2.1) is deemed as intractable.

Our Approach We propose a novel framework via which the best subset selection problem can be solved to optimality or near optimality in problems of practical interest within a reasonable time frame. At the core of our proposal is a computationally tractable framework that brings to bear the power of modern discrete optimization methods: discrete first order methods motivated by first order methods in convex optimization [87] and mixed integer optimization (MIO), see [11]. We do not guarantee polynomial time solution times as these do not exist for the best subset problem unless P=NP. Rather, our view of computational tractability is the ability of a method to solve problems of practical interest in times that are appropriate for the application addressed. An advantage of our approach is that it adapts to variants of the best subset regression problem of the form:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_q^q \\ s.t. \quad & \|\boldsymbol{\beta}\|_0 \leq k \\ & \mathbf{A}\boldsymbol{\beta} \leq \mathbf{b}, \end{aligned}$$

where $\mathbf{A}\boldsymbol{\beta} \leq \mathbf{b}$ represents polyhedral constraints and $q \in \{1, 2\}$ refers to a least absolute deviation or the least squares loss function on the residuals $\mathbf{r} := \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$. We will explore

such polyhedral constraints in Chapter 3 in the context of building a high quality linear regression model. For the moment, we consider only the basic version - that is, Problem 2.1.

Existing approaches in the Mathematical Optimization Literature In a seminal paper [52], the authors describe a leaps and bounds procedure for computing global solutions to Problem (2.1) (for the classical $n > p$ case) which can be achieved with computational effort significantly less than complete enumeration. `leaps`, a state-of-the-art `R` package uses this principle to perform best subset selection for problems with $n > p$ and $p \leq 30$. [10] proposed a tailored branch-and-bound scheme that can be applied to Problem (2.1) using ideas from [52] and techniques in quadratic optimization, extending and enhancing the proposal of [15]. The proposal of [10] concentrates on obtaining high quality upper bounds for Problem (2.1) and is less scalable than the methods presented here.

Contributions We summarize the contributions we make in this chapter below:

1. We use MIO to find a provably optimal solution for the best subset problem. Our approach has the appealing characteristic that if we terminate the algorithm early, we obtain a solution with a guarantee on its suboptimality. Furthermore, our framework can accommodate side constraints on $\boldsymbol{\beta}$ and also extends to finding best subset solutions for the least absolute deviation loss function.

2. We introduce a general algorithmic framework based on a discrete extension of modern first order continuous optimization methods that provide near-optimal solutions for the best subset problem. The MIO algorithm significantly benefits from solutions obtained by the first order methods and problem specific information that can be computed in a data-driven fashion.

3. We report computational results with both synthetic and real-world datasets that show that our proposed framework can deliver provably optimal solutions for problems of size n in the 1000s and p in the 100s in minutes. For high-dimensional problems with

$n \in \{50, 100\}$ and $p \in \{1000, 2000\}$, with the aid of warm starts and further problem-specific information, our approach finds near optimal solutions in minutes but takes hours to prove optimality.

4. We investigate the statistical properties of best subset selection procedures for practical problem sizes, which to the best of our knowledge, have remained largely unexplored to date. We demonstrate the favorable predictive performance and sparsity-inducing properties of the best subset selection procedure over its competitors in a wide variety of real and synthetic examples for the least squares and absolute deviation loss functions.

The structure of this chapter is as follows. In Section 2.2, we present the proposed MIO formulations for the best subset problem as well as some connections with the compressed sensing literature for estimating parameters and providing lower bounds for the MIO formulations that improve their computational performance. In Section 2.3, we develop a discrete extension of first order methods in convex optimization to obtain near optimal solutions for the best subset problem and establish its convergence properties, a method that may be of independent interest. In Section 2.4, we perform a variety of computational tests on synthetic and real datasets to assess the algorithmic and statistical performances of our approach for the least squares loss function for both the classical overdetermined case $n > p$, and the high-dimensional case $p \gg n$. In Section 2.5, we include our concluding remarks.

2.2 Mixed Integer Optimization Formulations

We present the proposed MIO formulations for the best subset problem as well as some connections with the compressed sensing literature for estimating parameters and providing lower bounds for the MIO formulations that improve their computational performance.

2.2.1 MIO Formulations for the Best Subset Selection Problem

We first present a simple reformulation to Problem (2.1) as a MIO (in fact a MIQO) problem:

$$\begin{aligned} Z_1 = \min_{\boldsymbol{\beta},\mathbf{z}} \quad & \tfrac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 \\ s.t. \quad & -\mathcal{M}_U z_i \le \beta_i \le \mathcal{M}_U z_i, i=1,\ldots,p \\ & z_i \in \{0,1\}, i=1,\ldots,p \\ & \sum_{i=1}^{p} z_i \le k, \end{aligned} \tag{2.5}$$

where $\mathbf{z} \in \{0,1\}^p$ is a binary variable and $\mathcal{M}_U$ is a constant such that if $\widehat{\boldsymbol{\beta}}$ is a minimizer of Problem (2.5), then $\mathcal{M}_U \ge \|\widehat{\boldsymbol{\beta}}\|_\infty$. If $z_i = 1$, then $|\beta_i| \le \mathcal{M}_U$ and if $z_i = 0$, then $\beta_i = 0$. Thus, $\sum_{i=1}^{p} z_i$ is an indicator of the number of zeros in $\boldsymbol{\beta}$.

Provided that $\mathcal{M}_U$ is chosen to be sufficently large with $\mathcal{M}_U \ge \|\widehat{\boldsymbol{\beta}}\|_\infty$, a solution to Problem (2.5) will be a solution to Problem (2.1). Of course, $\mathcal{M}_U$ is not known a priori, and a small value of $\mathcal{M}_U$ may lead to a solution different from (2.1). The choice of $\mathcal{M}_U$ affects the strength of the formulation and is critical for obtaining solutions quickly in practice. In Section 2.2.2 we describe how to find appropriate values for $\mathcal{M}_U$.

Formulation (2.5) leads to interesting insights, especially via the structure of the convex hull of its constraints, as illustrated next :

$$\begin{aligned} & \text{Conv}\left(\left\{\boldsymbol{\beta} : |\beta_i| \le \mathcal{M}_U z_i, z_i \in \{0,1\}, i=1,\ldots,p, \sum_{i=1}^{p} z_i \le k\right\}\right) \\ = \; & \{\boldsymbol{\beta} : \|\boldsymbol{\beta}\|_\infty \le \mathcal{M}_U, \|\boldsymbol{\beta}\|_1 \le \mathcal{M}_U k\} \\ \subseteq \; & \{\boldsymbol{\beta} : \|\boldsymbol{\beta}\|_1 \le \mathcal{M}_U k\}. \end{aligned}$$

Thus, the minimum of Problem (2.5) is lower-bounded by the optimum objective value of

both the following convex optimization problems:

$$Z_2 := \min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \qquad \text{subject to } \|\boldsymbol{\beta}\|_\infty \leq \mathcal{M}_U, \|\boldsymbol{\beta}\|_1 \leq \mathcal{M}_U k \tag{2.6}$$

$$Z_3 := \min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \qquad \text{subject to } \|\boldsymbol{\beta}\|_1 \leq \mathcal{M}_U k, \tag{2.7}$$

where (2.7) is the familiar Lasso in constrained form. This is a weaker relaxation than formulation (2.6), which in addition to the ℓ_1 constraint on $\boldsymbol{\beta}$, has box-constraints controlling the values of the β_i's. It is easy to see that the following ordering exists:

$$Z_3 \leq Z_2 \leq Z_1,$$

with the inequalities being strict in most instances.

In terms of approximating the optimal solution to Problem (2.5), the MIO solver begins by first solving a continuous relaxation of Problem (2.5). The Lasso formulation (2.7) is weaker than this root node relaxation. Additionally, MIO is typically able to significantly improve the quality of the root node solution as the MIO solver progresses toward the optimal solution.

To motivate the reader we provide an example of the evolution (see Figure 2-1) of the MIO formulation (2.8) for the Diabetes dataset [44], with $n = 350, p = 64$ (for further details on the dataset see Section 2.4).

Since formulation (2.5) is sensitive to the choice of $\mathcal{M}_U$, we consider an alternative MIO formulation based on Specially Ordered Sets [11] as described next.

Formulations via Specially Ordered Sets Any feasible solution to formulation (2.5) will have $(1 - z_i)\beta_i = 0$ for every $i \in \{1, \ldots, p\}$. This constraint can be modeled via integer optimization using Specially Ordered Sets of Type 1 [11] (SOS-1). In an SOS-1 constraint, at most one variable in the set can take a nonzero value, that is

$$(1 - z_i)\beta_i = 0 \iff (\beta_i, 1 - z_i) : \text{SOS-1},$$

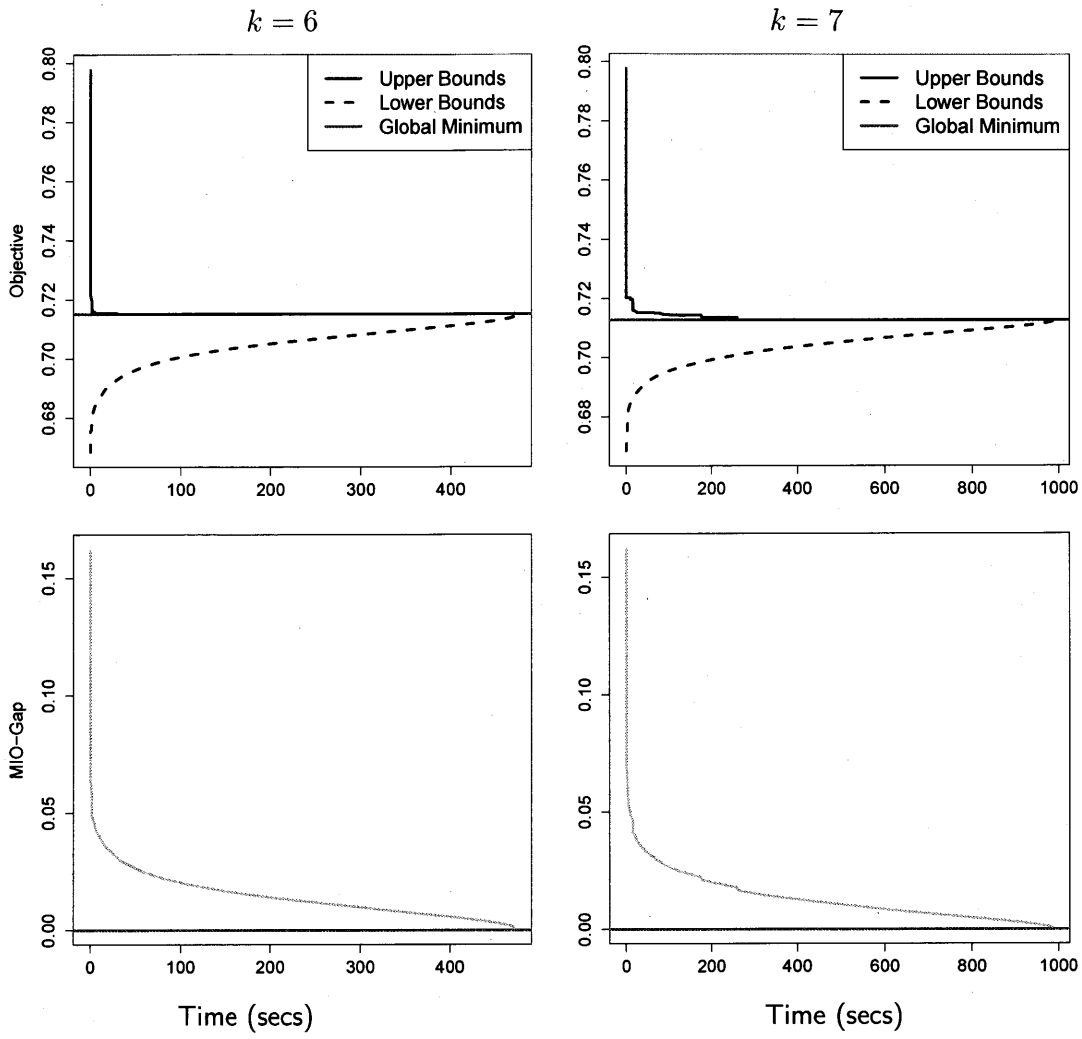

Figure 2-1: Figure showing the typical evolution of the MIO formulation (2.8) for the diabetes' dataset with $n = 350, p = 64$ with $k = 6$ (left panel) and $k = 7$ (right panel). The top panel shows the evolution of upper bounds, lower bounds with time. The lower panel shows the evolution of the corresponding MIO-Gap, with time. Global solutions for both the problems are found quite quickly in both examples, but it takes longer to certify global optimality via the lower bounds. As expected, the time taken for the MIO to certify convergence to the global optimum increases with increasing k.

for every $i = 1, \ldots, p$. This leads to the following formulation of (2.1):

$$\begin{aligned} \min_{\boldsymbol{\beta}, \mathbf{z}} \quad & \tfrac{1}{2} \left\| \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \right\|_2^2 \\ s.t. \quad & (\beta_i, 1 - z_i) : \text{SOS-1}, \ i = 1, \ldots, p \\ & z_i \in \{0, 1\}, i = 1, \ldots, p \\ & \sum_{i=1}^{p} z_i \leq k, \end{aligned} \tag{2.8}$$

The objective function in Problem (2.8) is a convex quadratic function in the continuous variable $\boldsymbol{\beta}$, which can be formulated explicitly as:

$$\begin{aligned}
\underset{\boldsymbol{\beta},\mathbf{z}}{\text{minimize}} \quad & \boldsymbol{\beta}^T\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} - 2\langle \mathbf{X}'\mathbf{y}, \boldsymbol{\beta}\rangle + \|\mathbf{y}\|_2^2 \\
\text{subject to} \quad & (\beta_i, 1 - z_i) : \text{SOS type-1}, \ i = 1, \ldots, p \\
& z_i \in \{0, 1\}, i = 1, \ldots, p \\
& \sum_{i=1}^{p} z_i \leq k \\
& -\mathcal{M}_U \leq \beta_i \leq \mathcal{M}_U, i = 1, \ldots, p \\
& \|\boldsymbol{\beta}\|_1 \leq \mathcal{M}_\ell,
\end{aligned} \tag{2.9}$$

We also provide problem-dependent constants $\mathcal{M}_U$ and $\mathcal{M}_\ell \in [0, \infty]$. $\mathcal{M}_U$ provides an upper bound on the absolute value of the regression coefficients and $\mathcal{M}_\ell$ provides an upper bound on the ℓ_1-norm of $\boldsymbol{\beta}$. Adding these bounds typically leads to improved performance of the MIO. In Section 2.2.2 we describe methods to compute these parameters from the data.

We also consider another formulation for (2.9):

$$
\begin{aligned}
\min_{\boldsymbol{\beta},\mathbf{z},\boldsymbol{\zeta}} \quad & \tfrac{1}{2}\,\boldsymbol{\zeta}^T\boldsymbol{\zeta} - \langle \mathbf{X}'\mathbf{y}, \boldsymbol{\beta}\rangle + \tfrac{1}{2}\,\|\mathbf{y}\|_2^2 \\
s.t. \quad & \boldsymbol{\zeta} = \mathbf{X}\boldsymbol{\beta} \\
& (\beta_i, 1 - z_i) : \text{SOS-1}, \;\; i = 1, \ldots, p \\
& z_i \in \{0,1\}, i = 1, \ldots, p \\
& \sum_{i=1}^{p} z_i \leq k \\
& -\mathcal{M}_U \leq \beta_i \leq \mathcal{M}_U, i = 1, \ldots, p \\
& \|\boldsymbol{\beta}\|_1 \leq \mathcal{M}_\ell \\
& -\mathcal{M}_U^{\zeta} \leq \zeta_i \leq \mathcal{M}_U^{\zeta}, i = 1, \ldots, n \\
& \|\boldsymbol{\zeta}\|_1 \leq \mathcal{M}_\ell^{\zeta},
\end{aligned}
\tag{2.10}
$$

where the optimization variables are $\boldsymbol{\beta} \in \mathbb{R}^p, \boldsymbol{\zeta} \in \mathbb{R}^n, \mathbf{z} \in \{0,1\}^p$ and $\mathcal{M}_U, \mathcal{M}_\ell, \mathcal{M}_U^{\zeta}, \mathcal{M}_\ell^{\zeta} \in [0, \infty]$ are problem specific parameters. Note that the objective function in formulation (2.10) involves a quadratic form in n variables and a linear function in p variables. Problem (2.10) is equivalent to the following variant of the best subset problem:

$$
\begin{aligned}
\min_{\boldsymbol{\beta}} \quad & \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \\
s.t. \quad & \|\boldsymbol{\beta}\|_\infty \leq \mathcal{M}_U, \|\boldsymbol{\beta}\|_1 \leq \mathcal{M}_\ell \\
& \|\mathbf{X}\boldsymbol{\beta}\|_\infty \leq \mathcal{M}_U^{\zeta}, \|\mathbf{X}\boldsymbol{\beta}\|_1 \leq \mathcal{M}_\ell^{\zeta}.
\end{aligned}
\tag{2.11}
$$

Formulations (2.9) and (2.10) differ in the size of the quadratic forms that are involved. The current state-of-the-art MIO solvers are better-equipped to handle mixed integer linear optimization problems than MIQO problems. Formulation (2.9) has fewer variables but a

quadratic form in p variables—we find this formulation more useful in the $n > p$ regime, with p in the 100s. Formulation (2.10) on the other hand has more variables, but involves a quadratic form in n variables—this formulation is more useful for high-dimensional problems $p \gg n$, with n in the 100s and p in the 1000s.

The bounds on $\boldsymbol{\beta}$ and $\boldsymbol{\zeta}$ are not required, but if these constraints are provided, they improve the strength of the MIO formulation. We next show how these bounds can be computed from given data.

2.2.2 Specification of Parameters

In this section, we obtain estimates for the quantities $\mathcal{M}_U, \mathcal{M}_\ell, \mathcal{M}_U^\zeta, \mathcal{M}_\ell^\zeta$ such that an optimal solution to problem (2.11) is also an optimal solution to Problem (2.1), and vice-versa.

Coherence and Restricted Eigenvalues of a Model Matrix

Given a model matrix $\mathbf{X}$, [106] introduced the cumulative coherence function

$$\mu[k] := \max_{|I|=k} \max_{j \notin I} \sum_{i \in I} |\langle \mathbf{X}_j, \mathbf{X}_i \rangle|,$$

where, $\mathbf{X}_j$, $j = 1, \ldots, p$ represent the columns of $\mathbf{X}$, i.e., features.

For $k = 1$, we obtain the notion of coherence introduced in [40, 38] as a measure of the maximal pairwise correlation in absolute value of the columns of $\mathbf{X}$:

$$\mu := \mu[1] = \max_{i \neq j} |\langle \mathbf{X}_i, \mathbf{X}_j \rangle|.$$

[26, 24] (see also [21] and references therein) introduced the notion that a matrix $\mathbf{X}$ satisfies a restricted eigenvalue condition if

$$\lambda_{\min}(\mathbf{X}_I' \mathbf{X}_I) \geq \eta_k \text{ for every } I \subset \{1, \ldots, p\} : |I| \leq k, \tag{2.12}$$

where $\lambda_{\min}(\mathbf{X}_I' \mathbf{X}_I)$ denotes the smallest eigenvalue of the matrix $\mathbf{X}_I' \mathbf{X}_I$. An inequality linking

$\mu[k]$ and η_k is as follows.

Proposition 1. *The following bounds hold :*

(a) *[106]:* $\mu[k] \leq \mu \cdot k$.

(b) *[38] :* $\eta_k \geq 1 - \mu[k-1] \geq 1 - \mu \cdot (k-1)$.

The computations of $\mu[k]$ and η_k for general k are difficult, while μ is simple to compute. Proposition 1 provides bounds for $\mu[k]$ and η_k in terms of the coherence μ.

Operator Norms of Submatrices

The (p, q) operator norm of matrix $\mathbf{A}$ is

$$\|\mathbf{A}\|_{p,q} := \max_{\|\mathbf{u}\|_q=1} \|\mathbf{A}\mathbf{u}\|_p.$$

We will use extensively here the $(1, 1)$ operator norm. We assume that each column vector of $\mathbf{X}$ has unit ℓ_2-norm. The results derived in the next proposition borrow and enhance techniques developed by [106] in the context of analyzing the ℓ_1—ℓ_0 equivalence in compressed sensing.

Proposition 2. *For any* $I \subset \{1, \ldots, p\}$ *with* $|I| = k$ *we have :*

(a) $\|\mathbf{X}_I'\mathbf{X}_I - \mathbf{I}\|_{1,1} \leq \mu[k-1]$.

(b) *If the matrix* $\mathbf{X}_I'\mathbf{X}_I$ *is invertible and* $\|\mathbf{X}_I'\mathbf{X}_I - \mathbf{I}\|_{1,1} < 1$, *then*

$$\|(\mathbf{X}_I'\mathbf{X}_I)^{-1}\|_{1,1} \leq \frac{1}{1 - \mu[k-1]}. \tag{2.13}$$

Proof

(a) Given a set I, we define $\mathbf{G} := \mathbf{X}_I'\mathbf{X}_I - \mathbf{I}$, and let g_{ij} denote the (i,j)th entry of $\mathbf{G}$. For any $\mathbf{u} \in \mathbb{R}^k$ we have

$$\begin{aligned}
\max_{\|\mathbf{u}\|_1=1} \|\mathbf{G}\mathbf{u}\|_1 &= \max_{\|\mathbf{u}\|_1=1} \left(\sum_{i=1}^{k} \left| \sum_{j=1}^{k} g_{ij} u_j \right| \right) \\
&\leq \max_{\|\mathbf{u}\|_1=1} \left(\sum_{i=1}^{k} \sum_{j=1}^{k} |u_j||g_{ij}| \right) \\
&= \max_{\|\mathbf{u}\|_1=1} \left(\sum_{j=1}^{k} |u_j| \sum_{i \neq j} |g_{ij}| \right) && (g_{jj} = 0) \\
&\leq \max_{\|\mathbf{u}\|_1=1} (\mu[k-1]\|\mathbf{u}\|_1) && \left(\sum_{i \neq j} |g_{ij}| \leq \mu[k-1] \right) \\
&= \mu[k-1].
\end{aligned}$$

(b) Using $\mathbf{X}_I'\mathbf{X}_I = \mathbf{I} + \mathbf{G}$ and standard power-series convergence (which is valid since $\|\mathbf{G}\|_{1,1} < 1$) we obtain

$$\|(\mathbf{X}_I'\mathbf{X}_I)^{-1}\|_{1,1} = \| (\mathbf{I}+\mathbf{G})^{-1} \|_{1,1} = \sum_{i=0}^{\infty} \|\mathbf{G}\|_{1,1}^i \leq \frac{1}{1 - \|\mathbf{G}\|_{1,1}} \leq \frac{1}{1 - \mu[k-1]}. \qquad \square$$

We note that Part (b) also appears in [106] for the operator norm $\|(\mathbf{X}_I'\mathbf{X}_I)^{-1}\|_{\infty,\infty}$.

Given a set $I \subset \{1, \ldots, p\}$ with $|I| = k$ we let $\widehat{\boldsymbol{\beta}}_I$ denote the least squares regression coefficients obtained by regressing $\mathbf{y}$ on $\mathbf{X}_I$, i.e., $\widehat{\boldsymbol{\beta}}_I = (\mathbf{X}_I'\mathbf{X}_I)^{-1}\mathbf{X}_I'\mathbf{y}$. If we append $\widehat{\boldsymbol{\beta}}_I$ with zeros in the remaining coordinates we obtain $\widehat{\boldsymbol{\beta}}$:

$$\widehat{\boldsymbol{\beta}} \in \operatorname*{arg\,min}_{\boldsymbol{\beta}:\beta_i=0, i \notin I} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2.$$

Note that $\widehat{\boldsymbol{\beta}}$ depends on I but we will suppress the dependence on I for notational convenience.

Recall that $\mathbf{X}_j$, $j = 1, \ldots, p$ represent the columns of $\mathbf{X}$; and we will use $\mathbf{x}_i$, $i = 1, \ldots, n$ to denote the rows of $\mathbf{X}$. As discussed above $\|\mathbf{X}_j\| = 1$. We order the correlations $|\langle \mathbf{X}_j, \mathbf{y} \rangle|$:

$$|\langle \mathbf{X}_{(1)}, \mathbf{y} \rangle| \geq |\langle \mathbf{X}_{(2)}, \mathbf{y} \rangle| \ldots \geq |\langle \mathbf{X}_{(p)}, \mathbf{y} \rangle|.$$

We finally denote by $\|\mathbf{x}_i\|_{1:k}$ the sum of the top k absolute values of the entries of $x_{ij}, j \in \{1, 2, \ldots, p\}$.

Theorem 1. *For any $k \geq 1$ such that $\mu[k-1] < 1$ any optimal solution $\widehat{\boldsymbol{\beta}}$ to (2.1) satisfies:*

$$\textbf{(a)} \qquad \|\widehat{\boldsymbol{\beta}}\|_1 \leq \frac{1}{1 - \mu[k-1]} \sum_{j=1}^{k} |\langle \mathbf{X}_{(j)}, \mathbf{y} \rangle|. \tag{2.14}$$

$$\textbf{(b)} \qquad \|\widehat{\boldsymbol{\beta}}\|_\infty \leq \min \left\{ \frac{1}{\eta_k} \sqrt{\sum_{j=1}^{k} |\langle \mathbf{X}_{(j)}, \mathbf{y} \rangle|^2}, \frac{1}{\sqrt{\eta_k}} \|\mathbf{y}\|_2 \right\}. \tag{2.15}$$

$$\textbf{(c)} \qquad \|\mathbf{X}\widehat{\boldsymbol{\beta}}\|_1 \leq \min \left\{ \sum_{i=1}^{n} \|\mathbf{x}_i\|_\infty \|\widehat{\boldsymbol{\beta}}\|_1, \sqrt{k} \|\mathbf{y}\|_2 \right\}. \tag{2.16}$$

$$\textbf{(d)} \qquad \|\mathbf{X}\widehat{\boldsymbol{\beta}}\|_\infty \leq \left(\max_{i=1,\ldots,n} \|\mathbf{x}_i\|_{1:k} \right) \|\widehat{\boldsymbol{\beta}}\|_\infty. \tag{2.17}$$

Proof

(a) Since $\widehat{\boldsymbol{\beta}}_I = (\mathbf{X}_I' \mathbf{X}_I)^{-1} \mathbf{X}_I' \mathbf{y}$ we have

$$\|\widehat{\boldsymbol{\beta}}\|_1 = \|\widehat{\boldsymbol{\beta}}_I\|_1 \leq \|(\mathbf{X}_I' \mathbf{X}_I)^{-1}\|_{1,1} \|\mathbf{X}_I' \mathbf{y}\|_1. \tag{2.18}$$

Note that

$$\|\mathbf{X}_I' \mathbf{y}\|_1 = \sum_{j \in I} |\langle \mathbf{X}_j, \mathbf{y} \rangle| \leq \max_{I, |I| = k} \sum_{j \in I} |\langle \mathbf{X}_j, \mathbf{y} \rangle| \leq \sum_{j=1}^{k} |\langle \mathbf{X}_{(j)}, \mathbf{y} \rangle|. \tag{2.19}$$

Applying (2.13) and (2.19) to (2.18), we obtain (2.14) .

(b) We write $\widehat{\boldsymbol{\beta}}_I = \mathbf{A}\mathbf{y}$ for $\mathbf{A} = (\mathbf{X}_I' \mathbf{X}_I)^{-1} \mathbf{X}_I'$. If $\mathbf{a}_i, i = 1, \ldots, k$ denote the rows of $\mathbf{A}$ we

have:

$$\|\widehat{\boldsymbol{\beta}}_I\|_\infty = \max_{i=1,\ldots,k} |\langle \mathbf{a}_i, \mathbf{y} \rangle| \leq \left(\max_{i=1,\ldots,k} \|\mathbf{a}_i\|_2 \right) \|\mathbf{y}\|_2. \tag{2.20}$$

For every $i = 1, \ldots, k$ we have

$$\begin{aligned} \|\mathbf{a}_i\|_2 \leq & \max_{\|\mathbf{u}\|_2=1} \|\mathbf{A}\mathbf{u}\|_2 \\ = & \max_{\|\mathbf{u}\|_2=1} \|(\mathbf{X}_I'\mathbf{X}_I)^{-1}\mathbf{X}_I'\mathbf{u}\|_2 \\ \leq & \lambda_{\max}\left((\mathbf{X}_I'\mathbf{X}_I)^{-1}\mathbf{X}_I'\right) \\ = & \max\left\{\frac{1}{d_1}, \ldots, \frac{1}{d_k}\right\}, \end{aligned} \tag{2.21}$$

where $d_1, \ldots, d_k$ are the (nonzero) singular values of the matrix $\mathbf{X}_I$. To see how one arrives at (2.21) let us denote the singular value decomposition of $\mathbf{X}_I = \mathbf{U}\mathbf{D}\mathbf{V}'$ with $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_k)$. We then have

$$(\mathbf{X}_I'\mathbf{X}_I)^{-1}\mathbf{X}_I' = (\mathbf{V}\mathbf{D}^{-2}\mathbf{V}')(\mathbf{U}\mathbf{D}\mathbf{V}')' = \mathbf{V}\mathbf{D}^{-1}\mathbf{U}'$$

and the singular values of $(\mathbf{X}_I'\mathbf{X}_I)^{-1}\mathbf{X}_I'$ are thus $1/d_i$, $i = 1, \ldots, k$.

The eigenvalues of $\mathbf{X}_I'\mathbf{X}_I$ are d_i^2 and from (2.12) we obtain that $d_i^2 \geq \eta_k$. Using (2.21) we thus obtain

$$\max_{i=1,\ldots,k} \|\mathbf{a}_i\|_2 \leq \frac{1}{\sqrt{\eta_k}}. \tag{2.22}$$

Substituting the bound (2.22) to (2.20) we obtain

$$\|\widehat{\boldsymbol{\beta}}_I\|_\infty \leq \frac{1}{\sqrt{\eta_k}} \|\mathbf{y}\|_2. \tag{2.23}$$

Using the notation $\tilde{\mathbf{A}} = (\mathbf{X}_I'\mathbf{X}_I)^{-1}$, we have

$$\begin{aligned}
\|\widehat{\boldsymbol{\beta}}_I\|_\infty &= \max_{i=1,\dots,k} |\langle \tilde{\mathbf{a}}_i, \mathbf{X}_I'\mathbf{y}\rangle| \\
&\le \left(\max_{i=1,\dots,k|} \|\tilde{\mathbf{a}}_i\|_2\right) \|\mathbf{X}_I'\mathbf{y}\|_2 \\
&\le \lambda_{\max}\left((\mathbf{X}_I'\mathbf{X}_I)^{-1}\right) \|\mathbf{X}_I'\mathbf{y}\|_2 \\
&= \left(\max_{i=1,\dots,k} \frac{1}{d_i^2}\right) \cdot \sqrt{\sum_{j\in I} |\langle \mathbf{X}_j, \mathbf{y}\rangle|^2} \\
&\le \frac{1}{\eta_k} \sqrt{\sum_{j=1}^{k} |\langle \mathbf{X}_{(j)}, \mathbf{y}\rangle|^2}. \qquad (2.24)
\end{aligned}$$

Combining (2.23) and (2.24) we obtain (2.15).

(c) We have

$$\|\mathbf{X}_I\widehat{\boldsymbol{\beta}}_I\|_1 \le \sum_{i=1}^{n} |\langle \mathbf{x}_i, \widehat{\boldsymbol{\beta}}_I\rangle| \le \sum_{i=1}^{n} \|\mathbf{x}_i\|_\infty \|\widehat{\boldsymbol{\beta}}_I\|_1 = \sum_{i=1}^{n} \|\mathbf{x}_i\|_\infty \|\widehat{\boldsymbol{\beta}}_I\|_1. \qquad (2.25)$$

Let $\mathbf{P}_I := \mathbf{X}_I(\mathbf{X}_I'\mathbf{X}_I)^{-1}\mathbf{X}_I'$ denote the projection onto the columns of $\mathbf{X}_I$. We have $\|\mathbf{P}_I\mathbf{y}\|_2 \le \|\mathbf{y}\|_2$, leading to:

$$\|\mathbf{X}_I\widehat{\boldsymbol{\beta}}_I\|_1 = \|\mathbf{P}_I\mathbf{y}\|_1 \le \sqrt{k}\|\mathbf{P}_I\mathbf{y}\|_2 \le \sqrt{k}\|\mathbf{y}\|_2, \qquad (2.26)$$

where we used that for any $\mathbf{a} \in \mathbb{R}^m$, we have $\sqrt{m}\|\mathbf{a}\|_2 \ge \|\mathbf{a}\|_1$. Combining (2.25) and (2.26) we obtain (2.16).

(d) For any vector $\boldsymbol{\beta}_I$ which has zero entries in the coordinates outside I, we have:

$$\|\mathbf{X}\boldsymbol{\beta}_I\|_\infty \le \max_{i=1,\dots,n} |\langle \mathbf{x}_i, \boldsymbol{\beta}_I\rangle| \le \max_{i=1,\dots,n} \|\mathbf{x}_i\|_{1:k} \|\boldsymbol{\beta}_I\|_\infty,$$

leading to (2.17). □

2.3 Discrete First Order Algorithms

In this section, we develop a discrete extension of first order methods in convex optimization [87, 86] to obtain near optimal solutions for Problem (2.1). Our approach applies to the problem of minimizing any sufficiently smooth convex function subject to cardinality constraints.

We will use these discrete first order methods to obtain solutions to warm start the MIO formulation. In Section 2.4, we will demonstrate how these methods greatly enhance the performance of the MIO.

2.3.1 Algorithms for minimizing smooth functions subject to cardinality constraints

Related work and contributions In the signal processing literature [17, 18] proposed iterative hard-thresholding algorithms in the context of ℓ_0-regularized least squares problems, i.e., Problem (2.4). The authors establish convergence properties of the algorithm under the assumption that $\mathbf{X}$ satisfies coherence [17] or Restricted Isometry Property [18]. The method we propose here applies to a larger class of cardinality constrained optimization problems of the form (2.27), in particular, in the context of Problem (2.1) our algorithm and its convergence analysis do not require any form of restricted isometry property on the model matrix $\mathbf{X}$.

Our proposed algorithm borrows ideas from projected gradient descent methods in first order convex optimization problems [87] and generalizes it to the discrete optimization Problem (2.27). We also derive new global convergence results for our proposed algorithms as presented in Theorem 2.

Consider the following optimization problem:

$$\min_{\boldsymbol{\beta}} \quad g(\boldsymbol{\beta}) \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_0 \leq k, \tag{2.27}$$

where $g(\boldsymbol{\beta}) \geq 0$ is convex and has Lipschitz continuous gradient:

$$\|\nabla g(\boldsymbol{\beta}) - \nabla g(\widetilde{\boldsymbol{\beta}})\| \leq \ell \|\boldsymbol{\beta} - \widetilde{\boldsymbol{\beta}}\|. \tag{2.28}$$

The first ingredient of our approach is the observation that when $g(\boldsymbol{\beta}) = \|\boldsymbol{\beta} - \mathbf{c}\|_2^2$ for a given $\mathbf{c}$, problem (2.27) admits a closed form solution.

Proposition 3. *An optimal solution, denoted as* $\mathbf{H}_k(\mathbf{c})$*, to the problem*

$$\min_{\|\boldsymbol{\beta}\|_0 \leq k} \|\boldsymbol{\beta} - \mathbf{c}\|_2^2, \tag{2.29}$$

can be computed as follows: $\mathbf{H}_k(\mathbf{c})$ *retains the* k *largest (in absolute value) elements of* $\mathbf{c} \in \mathbb{R}^p$ *and sets the rest to zero, i.e., if* $|c_{(1)}| \geq |c_{(2)}| \geq \ldots \geq |c_{(p)}|$*, denote the ordered values of the absolute values of the vector* $\mathbf{c}$*, then:*

$$(\mathbf{H}_k(\mathbf{c}))_i = \begin{cases} c_i, & \text{if } i \in \{(1), \ldots, (k)\}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.30}$$

Proof

It suffices to consider $|c_i| > 0$ for all i. Let $\boldsymbol{\beta}$ be an optimal solution to Problem (2.29) and let $S := \{i : \beta_i \neq 0\}$. The objective function is given by $\sum_{i \notin S} |c_i|^2 + \sum_{i \in S} (\beta_i - c_i)^2$. Note that by selecting $\beta_i = c_i$ for $i \in S$, we can make the objective function $\sum_{i \notin S} |c_i|^2$. Thus, to minimize the objective function, S must correspond to the indices of the largest k values of $|c_i|, i \geq 1$. □

The operator (2.30) is also known as the hard-thresholding operator [37]—a notion that arises in the context of the following related optimization problem:

$$\hat{\boldsymbol{\beta}} \in \arg\min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\boldsymbol{\beta} - \mathbf{c}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_0, \tag{2.31}$$

where $\hat{\boldsymbol{\beta}}$ admits a simple closed form expression given by $\hat{\beta}_i = c_i$ if $|c_i| > \lambda$ and $\hat{\beta}_i = 0$ otherwise, for $i = 1, \ldots, p$.

Given a current solution $\boldsymbol{\beta}$, the second ingredient of our approach is to upper bound the function $g(\boldsymbol{\eta})$ around $g(\boldsymbol{\beta})$. To do so, we use ideas from projected gradient descent methods in first order convex optimization problems [87, 86].

Proposition 4. *([87, 86]) For a convex function $g(\boldsymbol{\beta})$ satisfying condition* (2.28) *and for any $L \geq \ell$ we have :*

$$g(\boldsymbol{\eta}) \leq Q_L(\boldsymbol{\eta}, \boldsymbol{\beta}) := g(\boldsymbol{\beta}) + \frac{L}{2}\|\boldsymbol{\eta} - \boldsymbol{\beta}\|_2^2 + \langle \nabla g(\boldsymbol{\beta}), \boldsymbol{\eta} - \boldsymbol{\beta} \rangle \tag{2.32}$$

for all $\boldsymbol{\beta}, \boldsymbol{\eta}$ with equality holding at $\boldsymbol{\beta} = \boldsymbol{\eta}$.

Applying Proposition 3 to the upper bound $Q_L(\boldsymbol{\eta}, \boldsymbol{\beta})$ in Proposition 4 we obtain

$$\begin{aligned}
\underset{\|\boldsymbol{\eta}\|_0 \leq k}{\arg\min}\ Q_L(\boldsymbol{\eta}, \boldsymbol{\beta}) &= \underset{\|\boldsymbol{\eta}\|_0 \leq k}{\arg\min}\ \left(\frac{L}{2} \left\| \boldsymbol{\eta} - \left(\boldsymbol{\beta} - \frac{1}{L}\nabla g(\boldsymbol{\beta}) \right) \right\|_2^2 - \frac{1}{2L}\|\nabla g(\boldsymbol{\beta})\|_2^2 + g(\boldsymbol{\beta}) \right) \\
&= \underset{\|\boldsymbol{\eta}\|_0 \leq k}{\arg\min}\ \left\| \boldsymbol{\eta} - \left(\boldsymbol{\beta} - \frac{1}{L}\nabla g(\boldsymbol{\beta}) \right) \right\|_2^2 \\
&= \mathbf{H}_k \left(\boldsymbol{\beta} - \frac{1}{L}\nabla g(\boldsymbol{\beta}) \right),
\end{aligned} \tag{2.33}$$

where $\mathbf{H}_k(\cdot)$ is defined in (2.30). In light of (2.33) we are now ready to present Algorithm 1 to find a local optimal solution to problem (2.27).

Algorithm 1

Input: $g(\boldsymbol{\beta})$, L, ϵ.

Output: A local optimal solution $\boldsymbol{\beta}^*$.

Algorithm:

1. Initialize with $\boldsymbol{\beta}_1 \in \mathbb{R}^p$ such that $\|\boldsymbol{\beta}_1\|_0 \leq k$.

2. For $m \geq 1$, apply (2.33) with $\boldsymbol{\beta} = \boldsymbol{\beta}_m$ to obtain $\boldsymbol{\beta}_{m+1}$ as:

$$\boldsymbol{\beta}_{m+1} = \mathbf{H}_k\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right) \tag{2.34}$$

3. Repeat Step 2, until $\|\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m\|_2 \leq \epsilon$.

4. Let $\boldsymbol{\beta}_m := (\beta_{m1}, \ldots, \beta_{mp})$ denote the current estimate and let $I = \text{Supp}(\boldsymbol{\beta}_m) := \{i : \beta_{mi} \neq 0\}$. Solve the continuous optimization problem:

$$\min_{\boldsymbol{\beta}, \beta_i = 0,\ i \notin I} g(\boldsymbol{\beta}), \tag{2.35}$$

and let $\boldsymbol{\beta}^*$ be a minimizer.

The convergence properties of Algorithm 1 are presented in Section 2.3.2. A variant of Algorithm 1 that has better empirical performance and uses line searches is presented next.

Algorithm 2 (with Line Search)

1. Initialize with $\boldsymbol{\beta}_1 \in \mathbb{R}^p$ such that $\|\boldsymbol{\beta}_1\|_0 \leq k$.

2. For $m \geq 1$,

$$\begin{aligned} \boldsymbol{\eta}_m &= \mathbf{H}_k\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right), \\ \boldsymbol{\beta}_{m+1} &= \lambda_m \boldsymbol{\eta}_m + (1 - \lambda_m)\boldsymbol{\beta}_m, \end{aligned} \tag{2.36}$$

where λ_m is chosen to minimize the one-dimensional optimization problem:

$$\lambda_m \in \arg\min_{\lambda} \; g\left(\lambda \boldsymbol{\eta}_m + (1 - \lambda)\boldsymbol{\beta}_m\right). \tag{2.37}$$

3. Repeat Step 2, until $\|\boldsymbol{\eta}_{m+1} - \boldsymbol{\eta}_m\|_2 \leq \epsilon$.

4. Let $\boldsymbol{\eta}_m$ denote the current estimate and let $I = \text{Supp}(\boldsymbol{\eta}_m)$. Solve problem (2.35) and let $\boldsymbol{\beta}^*$ be a mininizer.

Note that the iterate $\boldsymbol{\beta}_m$ in Algorithm 2 need not be k-sparse (i.e., need not satisfy: $\|\boldsymbol{\beta}_m\|_0 \leq k$), however, $\boldsymbol{\eta}_m$ is k-sparse ($\|\boldsymbol{\eta}_m\|_0 \leq k$). Moreover, the sequence may not lead to a decreasing set of objective values, but it satisfies:

$$g(\boldsymbol{\beta}_{m+1}) \leq g(\boldsymbol{\eta}_m) \not\leq g(\boldsymbol{\beta}_m).$$

2.3.2 Convergence Analysis of Algorithm 1

In this section, we study convergence properties for Algorithm 1. Before we embark on the analysis, we need to define the notion of first order optimality for Problem (2.27).

Definition 1. *Given an $L \geq \ell$, the vector $\boldsymbol{\eta} \in \mathbb{R}^p$ is said to be a first order stationary point of Problem (2.27) if $\|\boldsymbol{\eta}\|_0 \leq k$ and it satisfies the following fixed point equation:*

$$\boldsymbol{\eta} = \mathbf{H}_k\left(\boldsymbol{\eta} - \frac{1}{L}\nabla g(\boldsymbol{\eta})\right). \tag{2.38}$$

We next define the notion of an ϵ-approximate first order stationary point of Problem (2.27):

Definition 2. *Given an $\epsilon > 0$, and $L \geq \ell$ we say that $\boldsymbol{\eta}$ satisfies an ϵ-approximate first order optimality condition of Problem (2.27) if $\|\boldsymbol{\eta}\|_0 \leq k$ and*

$$\left\|\boldsymbol{\eta} - \mathbf{H}_k\left(\boldsymbol{\eta} - \frac{1}{L}\nabla g(\boldsymbol{\eta})\right)\right\|_2 \leq \epsilon.$$

Let $\boldsymbol{\beta}_m = (\beta_{m1}, \ldots, \beta_{mp})$ and $\mathbf{1}_m = (e_1, \ldots, e_p)$ with $e_j = 1$, if $\beta_{mj} \neq 0$, and $e_j = 0$, if $\beta_{mj} = 0$, $j = 1, \ldots, p$, i.e., $\mathbf{1}_m$ represents the sparsity pattern of the support of $\boldsymbol{\beta}_m$.

Proposition 5. *Consider $g(\boldsymbol{\beta})$ and ℓ as defined in (2.27) and (2.28). Let $\boldsymbol{\beta}_m, m \geq 1$ be the sequence generated by Algorithm 1. Then we have :*

(a) *For any $L \geq \ell$, the sequence $g(\boldsymbol{\beta}_m)$ satisfies*

$$g(\boldsymbol{\beta}_m) - g(\boldsymbol{\beta}_{m+1}) \geq \frac{L-\ell}{2} \left\|\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m\right\|_2^2, \tag{2.39}$$

is decreasing and converges.

(b) *If $L > \ell$, then $\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m \to \mathbf{0}$ as $m \to \infty$.*

(c) *If $L > \ell$ and $\| \liminf_{m\to\infty} \boldsymbol{\beta}_m \|_0 = k$ then the sequence $\mathbf{1}_m$ converges after finitely many iterations, i.e., there exists an iteration index M^* such that $\mathbf{1}_m = \mathbf{1}_{m+1}$ for all $m \geq M^*$. Furthermore, the sequence $\boldsymbol{\beta}_m$ is bounded and converges to a first order stationary point.*

(d) *If $L > \ell$ and $\| \liminf_{m\to\infty} \boldsymbol{\beta}_m \|_0 < k$, then $g(\boldsymbol{\beta}_m) \to g(\boldsymbol{\beta}^*)$ where $\boldsymbol{\beta}^* \in \arg\min g(\boldsymbol{\beta})$ is an unconstrained minimizer.*

Proof

(a) Let $\boldsymbol{\beta}$ be a vector satisfying $\|\boldsymbol{\beta}\|_0 \leq k$. Using the notation $\widehat{\boldsymbol{\eta}} = \mathbf{H}_k\left(\boldsymbol{\beta} - \frac{1}{L}\nabla g(\boldsymbol{\beta})\right)$ we have the following chain of inequalities:

$$\begin{aligned}
g(\boldsymbol{\beta}) &= Q_L(\boldsymbol{\beta}, \boldsymbol{\beta}) \\
&\geq \inf_{\|\boldsymbol{\eta}\|_0 \leq k} \; Q_L(\boldsymbol{\eta}, \boldsymbol{\beta}) \\
&= \inf_{\|\boldsymbol{\eta}\|_0 \leq k} \; \left(\frac{L}{2}\|\boldsymbol{\eta} - \boldsymbol{\beta}\|_2^2 + \langle \nabla g(\boldsymbol{\beta}), \boldsymbol{\eta} - \boldsymbol{\beta}\rangle + g(\boldsymbol{\beta})\right) \\
&= \inf_{\|\boldsymbol{\eta}\|_0 \leq k} \; \left(\frac{L}{2}\left\|\boldsymbol{\eta} - \left(\boldsymbol{\beta} - \frac{1}{L}\nabla g(\boldsymbol{\beta})\right)\right\|_2^2 - \frac{1}{2L}\|\nabla g(\boldsymbol{\beta})\|_2^2 + g(\boldsymbol{\beta})\right) \\
&= \left(\frac{L}{2}\|\widehat{\boldsymbol{\eta}} - \left(\boldsymbol{\beta} - \frac{1}{L}\nabla g(\boldsymbol{\beta})\right)\|_2^2 - \frac{1}{2L}\|\nabla g(\boldsymbol{\beta})\|_2^2 + g(\boldsymbol{\beta})\right) \qquad \text{(From (2.33))} \\
&= \left(\frac{L}{2}\|\widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\|_2^2 + \langle \nabla g(\boldsymbol{\beta}), \widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\rangle + g(\boldsymbol{\beta})\right)
\end{aligned}$$

$$
\begin{aligned}
&= \left(\frac{L-\ell}{2} \|\widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\|_2^2 + \frac{\ell}{2} \|\widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\|_2^2 + \langle \nabla g(\boldsymbol{\beta}), \widehat{\boldsymbol{\eta}} - \boldsymbol{\beta} \rangle + g(\boldsymbol{\beta}) \right) \\
&\geq \frac{L-\ell}{2} \|\widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\|_2^2 + \underbrace{\left(\frac{\ell}{2} \|\widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\|_2^2 + \langle \nabla g(\boldsymbol{\beta}), \widehat{\boldsymbol{\eta}} - \boldsymbol{\beta} \rangle + g(\boldsymbol{\beta}) \right)}_{Q_\ell(\widehat{\boldsymbol{\eta}}, \boldsymbol{\beta})} \\
&\geq \frac{L-\ell}{2} \|\widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\|_2^2 + g(\widehat{\boldsymbol{\eta}}). \qquad \text{(From (2.32))}
\end{aligned}
$$

This chain of inequalities leads to:

$$
g(\boldsymbol{\beta}) - g(\widehat{\boldsymbol{\eta}}) \geq \frac{L-\ell}{2} \|\widehat{\boldsymbol{\eta}} - \boldsymbol{\beta}\|_2^2 . \tag{2.40}
$$

Applying (2.40) for $\boldsymbol{\beta} = \boldsymbol{\beta}_m$ and $\widehat{\boldsymbol{\eta}} = \boldsymbol{\beta}_{m+1}$, the vectors generated by Algorithm 1, we obtain (2.39). This implies that the objective values $g(\boldsymbol{\beta}_m)$ are decreasing and since the sequence is bounded below ($g(\boldsymbol{\beta}) \geq 0$), we obtain that $g(\boldsymbol{\beta}_m)$ converges as $m \to \infty$.

(b) If $L > \ell$ and from part (a), the result follows.

(c) We begin by observing that the condition $\|\liminf_{m\to\infty} \boldsymbol{\beta}_m\|_0 = k$ is equivalent to $\liminf_{m\to\infty} \min_{i:\beta_{mi}\neq 0} |\beta_{mi}| > 0$. We next prove that the support of $\boldsymbol{\beta}_m$ converges. For the purpose of establishing of contradiction suppose that the support does not converge. Then, there are infinitely many values of m' such that $\mathbf{1}_{m'} \neq \mathbf{1}_{m'+1}$. Using the fact that $\|\boldsymbol{\beta}_m\|_0 = k$ for all large m we have

$$
\|\boldsymbol{\beta}_{m'} - \boldsymbol{\beta}_{m'+1}\|_2 \geq \sqrt{\beta_{m',i}^2 + \beta_{m'+1,j}^2} \geq \frac{|\beta_{m',i}| + |\beta_{m'+1,j}|}{\sqrt{2}}, \tag{2.41}
$$

where i, j are such that $\beta_{m'+1,i} = \beta_{m',j} = 0$. As $m' \to \infty$, the quantity in the rhs of (2.41) remains bounded away from zero since $\liminf_{m\to\infty} \min_{i:\beta_{mi}\neq 0} |\beta_{mi}| > 0$. This contradicts the fact that $\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m \to \mathbf{0}$, as established in part (b). Thus, $\mathbf{1}_m$ converges, and since $\mathbf{1}_m$ is a discrete sequence, it converges after finitely many iterations, that is $\mathbf{1}_m = \mathbf{1}_{m+1}$ for all $m \geq M^*$. Algorithm 1 becomes a vanilla gradient descent algorithm, restricted to the space $\mathbf{1}_m$ for $m \geq M^*$. Since a gradient descent algorithm for minimizing a convex function over a closed convex set leads to a sequence of iter-

ates that converge [92, 87], we conclude that Algorithm 1 converges. Therefore, the sequence $\boldsymbol{\beta}_m$ converges to $\widehat{\boldsymbol{\beta}}$, a first order stationarity point :

$$\mathbf{H}_k\left(\widehat{\boldsymbol{\beta}} - \frac{1}{L}\nabla g(\widehat{\boldsymbol{\beta}})\right) = \widehat{\boldsymbol{\beta}}.$$

(d) Let $\mathcal{I}_m \subset \{1, \ldots, p\}$ denote the set of k largest values of the vector $\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right)$ in absolute value. By the definition of $\mathbf{H}_k\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right)$, we have

$$\left|\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right)_i\right| \geq \left|\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right)_j\right|,$$

for all i, j with $i \in \mathcal{I}_m$ and $j \notin \mathcal{I}_m$. Thus,

$$\liminf_{m\to\infty} \min_{i\in\mathcal{I}_m} \left|\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right)_i\right| \geq \liminf_{m\to\infty} \max_{j\notin\mathcal{I}_m} \left|\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right)_j\right|. \quad (2.42)$$

Moreover,

$$\left(\boldsymbol{\beta}_m - \mathbf{H}_k\left(\boldsymbol{\beta}_m - \frac{1}{L}\nabla g(\boldsymbol{\beta}_m)\right)\right)_i = \begin{cases} \frac{1}{L}(\nabla g(\boldsymbol{\beta}_m))_i, & i \in \mathcal{I}_m, \\ \beta_{m,i}, & \text{otherwise.} \end{cases}$$

Using the fact that $\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m \to \mathbf{0}$ we have

$$(\nabla g(\boldsymbol{\beta}_m))_i \to 0, i \in \mathcal{I}_m \text{ and } \beta_{m,j} \to 0, j \notin \mathcal{I}_m$$

as $m \to \infty$. Combining with (2.42) we have that:

$$\liminf_{m\to\infty} \min_{i\in\mathcal{I}_m} |\boldsymbol{\beta}_{mi}| \geq \liminf_{m\to\infty} \max_{j\notin\mathcal{I}_m} \frac{1}{L}\left|(\nabla g(\boldsymbol{\beta}_m))_j\right| = \frac{1}{L}\liminf_{m\to\infty} \|\nabla g(\boldsymbol{\beta}_m)\|_\infty.$$

Since, $\liminf_{m\to\infty} \min_{i\in\mathcal{I}_m} |\boldsymbol{\beta}_{mi}| = 0$ by hypothesis, the lhs of the above inequality equals zero, which leads to $\liminf_{m\to\infty} \|\nabla g(\boldsymbol{\beta}_m)\|_\infty = 0$. Thus, there is a subsequence $m' \subset \{1, 2, \ldots, \}$ such that $\nabla g(\boldsymbol{\beta}_{m'}) \to \mathbf{0}$, i.e., $\liminf_{m\to\infty} \nabla g(\boldsymbol{\beta}_m) \to \mathbf{0}$. Since,

$g(\boldsymbol{\beta}_m)$ is a decreasing sequence, this implies that $g(\boldsymbol{\beta}_m) \to g(\boldsymbol{\beta}^*)$, where, $\boldsymbol{\beta}^*$ is an unconstrained (without cardinality constraints) solution to min $g(\boldsymbol{\beta})$. □

Proposition 5 establishes that Algorithm 1 either converges to a first order stationarity point (part (c)) or it converges to a global optimal solution (part (d)), but does not quantify the rate of convergence. We next characterize the rate of convergence of the algorithm to an ϵ-approximate first order stationary point.

Theorem 2. *Let $L > \ell$ and $\widehat{\boldsymbol{\beta}}$ denote a first order stationary point of Algorithm 1. After M iterations Algorithm 1 satisfies*

$$\min_{m=1,\ldots,M} \|\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m\|_2^2 \leq \frac{2(g(\boldsymbol{\beta}_1) - g(\widehat{\boldsymbol{\beta}}))}{M(L-\ell)}, \tag{2.43}$$

where $g(\boldsymbol{\beta}_m) \downarrow g(\widehat{\boldsymbol{\beta}})$ as $m \to \infty$.

Proof

Summing inequalities (2.39) for $1 \leq m \leq M$. we obtain

$$\sum_{m=1}^{M} \left(g(\boldsymbol{\beta}_m) - g(\boldsymbol{\beta}_{m+1})\right) \geq \frac{L-\ell}{2} \sum_{m=1}^{M} \|\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m\|_2^2, \tag{2.44}$$

leading to

$$g(\boldsymbol{\beta}_1) - g(\boldsymbol{\beta}_{M+1}) \geq \frac{M(L-\ell)}{2} \min_{m=1,\ldots,M} \|\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m\|_2^2.$$

Since the decreasing sequence $g(\boldsymbol{\beta}_{m+1})$ converges to $g(\widehat{\boldsymbol{\beta}})$ by Proposition 5 we have:

$$\frac{g(\boldsymbol{\beta}_1) - g(\widehat{\boldsymbol{\beta}})}{M} \geq \frac{g(\boldsymbol{\beta}_1) - g(\boldsymbol{\beta}_{M+1})}{M} \geq \frac{(L-\ell)}{2} \min_{m=1,\ldots,M} \|\boldsymbol{\beta}_{m+1} - \boldsymbol{\beta}_m\|_2^2. \quad \square$$

Theorem 2 implies that for any $\epsilon > 0$ there exists $M = O(\frac{1}{\epsilon})$ such that for some $1 \leq m^* \leq M$

$$\|\boldsymbol{\beta}_{m^*+1} - \boldsymbol{\beta}_{m^*}\|_2^2 \leq \epsilon.$$

Polishing coefficients on the active set

Algorithm 1 *detects* the active set after a few iterations. Once the active set stabilizes, the algorithm may take a number of iterations to estimate the values of the regression coefficients on the active set to a high accuracy level.

In this context, we found the following simple polishing of coefficients to be useful. When the algorithm has converged to a tolerance of ϵ ($\approx 10^{-4}$), we fix the current active set, $\mathcal{I}$, and solve the following lower-dimensional convex optimization problem:

$$\min_{\boldsymbol{\beta}, \beta_i = 0, i \notin \mathcal{I}} \quad g(\boldsymbol{\beta}). \tag{2.45}$$

In the context of the least squares the optimization Problem (2.45) reduces to to a smaller dimensional least squares which can be solved very efficiently up to a very high level of accuracy.

2.3.3 Application to Least Squares

For the support constrained problem with squared error loss, we have

$$g(\boldsymbol{\beta}) = \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2, \;\; \nabla g(\boldsymbol{\beta}) = -\mathbf{X}'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

The general algorithmic framework developed above applies in a straightforward fashion for this special case. Note that for this case $\ell = \lambda_{\max}(\mathbf{X}'\mathbf{X})$.

The polishing of the regression coefficients in the active set can be performed via a least squares problem on $\mathbf{y}, \mathbf{X}_I$, where I denotes the support of the regression coefficients.

2.4 Computational Experiments for Subset Selection with Least Squares Loss

In this section, we present a variety of computational tests to assess the algorithmic and statistical performances of our approach. We consider both the classical overdetermined case with $n > p$ (Section 2.4.2) and the high dimensional $p \gg n$ case (Section 2.4.3) for the least squares loss function with support constraints.

2.4.1 Description of Experimental Data

We demonstrate the performance of our proposal via a series of experiments on both synthetic and real data.

Synthetic Datasets. We consider a collection of problems where $\mathbf{x}_i \sim \mathrm{N}(\mathbf{0}, \mathbf{\Sigma}), i = 1, \ldots, n$ are independent realizations from a p-dimensional multivariate normal distribution with mean zero and covariance matrix $\mathbf{\Sigma} := (\sigma_{ij})$. The columns of the $\mathbf{X}$ matrix were subsequently standardized to have unit ℓ_2 norm. For a fixed $\mathbf{X}_{n\times p}$, we generated the response $\mathbf{y}$ as follows: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^0 + \boldsymbol{\epsilon}$, where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. We denote the number of nonzeros in $\boldsymbol{\beta}^0$ by k_0. The choice of $\mathbf{X}, \boldsymbol{\beta}^0, \sigma$ determines the Signal-to-Noise Ratio (SNR) of the problem, which is defined as:

$$\mathrm{SNR} = \frac{\mathrm{var}(\mathbf{x}'\boldsymbol{\beta}^0)}{\sigma^2}.$$

We considered the following four different examples:

Example 1: We took $\sigma_{ij} = \rho^{|i-j|}$ for $i, j \in \{1, \ldots, p\} \times \{1, \ldots, p\}$. We consider different values of $k_0 \in \{5, 10\}$ and $\beta_i^0 = 1$ for k_0 equi-spaced values. In the case where exactly equi-spaced values are not possible we rounded the indices to the nearest large integer value. of i in the range $\{1, 2, \ldots, p\}$.

Example 2: We took $\mathbf{\Sigma} = \mathbf{I}_{p\times p}$, $k_0 = 5$ and $\boldsymbol{\beta}^0 = (\mathbf{1}'_{5\times 1}, \mathbf{0}'_{p-5\times 1})' \in \mathbb{R}^p$.

Example 3: We took $\mathbf{\Sigma} = \mathbf{I}_{p\times p}$, $k_0 = 10$ and $\beta_i^0 = \frac{1}{2} + (10 - \frac{1}{2})\frac{(i-1)}{k_0}, i = 1, \ldots, 10$ and $\beta_i^0 = 0, \forall i > 10$ — i.e., a vector with ten nonzero entries, with the nonzero values being

equally spaced in the interval $[\frac{1}{2}, 10]$.

Example 4: We took $\mathbf{\Sigma} = \mathbf{I}_{p\times p}$, $k_0 = 6$ and $\boldsymbol{\beta}^0 = (-10, -6, -2, 2, 6, 10, \mathbf{0}_{p-6})$, i.e., a vector with six nonzero entries, equally spaced in the interval $[-10, 10]$.

Real Datasets. We considered the Diabetes dataset analyzed in [44]. We used the dataset with all the second order interactions included in the model, which resulted in 64 predictors. We reduced the sample size to $n = 350$ by taking a random sample and standardized the response and the columns of the model matrix to have zero mean and unit ℓ_2-norm.

In addition to the above, we also considered a real microarray dataset, the Leukemia data [33]. We downloaded the processed dataset from `http://stat.ethz.ch/~dettling/bagboost.html`, which had $n = 72$ binary responses and more than 3000 predictors. We standardized the response and columns of features to have zero means and unit ℓ_2-norm. We reduced the set features to 1000 by retaining the features maximally correlated (in absolute value) to the response. We call the resulting feature matrix $\mathbf{X}_{n\times p}$ with $n = 72, p = 1000$. We then generated a semi-synthetic dataset with continuous response as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^0 + \epsilon$, where the first five coefficients of $\boldsymbol{\beta}^0$ were taken as one and the rest as zero. The noise was distributed as $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$, with σ^2 chosen to get a SNR=7.

Computer Specifications and Software Computations were carried out in a linux 64 bit server—Intel(R) Xeon(R) eight-core processor @ 1.80GHz, 16 GB of RAM for the overdetermined $n > p$ case and in a Dell Precision T7600 computer with an Intel Xeon E52687 sixteen-core processor @ 3.1GHz, 128GB of Ram for the high-dimensional $p \gg n$ case. The discrete first order methods were implemented in MATLAB 2012b. We used Gurobi [63] version 5.5 and the MATLAB interface to Gurobi for all of our experiments, apart from the computations for synthetic data for $n > p$, which were done in Gurobi via its Python 2.7 interface.

2.4.2 The Overdetermined Regime: $n > p$

Using the Diabetes dataset and synthetic datasets, we demonstrate the combined effect of using the discrete first order methods with the MIO approach. Together, these methods show improvements in obtaining good upper bounds and in closing the MIO gap to certify global optimality. Using synthetic datasets where we know the true linear regression model, we perform side-by-side comparisons of this method with several other state-of-the-art algorithms designed to estimate sparse linear models.

Obtaining Good Upper Bounds

We conducted experiments to evaluate the performance of our methods in terms of obtaining high quality solutions for Problem (2.1).

We considered the following three algorithms:

(a) Algorithm 2 with fifty random initializations[1]. We took the solution corresponding to the best objective value.

(b) MIO with cold start, i.e., formulation (2.9) with a time limit of 500 seconds.

(c) MIO with warm start. This was the MIO formulation initialized with the discrete first order optimization solution obtained from **(a)**. This was run for a total of 500 seconds.

To compare the different algorithms in terms of the quality of upper bounds, we run for every instance all the algorithms and obtain the best solution among them, say, f_*. If f_{alg} denotes the value of the best subset objective function for method "alg", then we define the relative accuracy of the solution obtained by "alg" as:

$$\text{Relative Accuracy} = (f_{\text{alg}} - f_*)/f_*, \tag{2.46}$$

where alg $\in \{(\text{a}), (\text{b}), (\text{c})\}$ as described above.

[1] we took fifty random starting values around $\mathbf{0}$ of the form $\min(i-1, 1)\epsilon, i = 1, \ldots, 50$, where $\epsilon \sim N(\mathbf{0}_{p\times 1}, 4\mathbf{I})$. We found empirically that Algorithm 2 provided better upper bounds than Algorithm 1.

We did experiments for the Diabetes dataset for different values of k (see Table 2.1). For each of the algorithms we report the amount of time taken by the algorithm to reach the best objective value during the time of 500 seconds.

k	Discrete First Order		MIO Cold Start		MIO Warm Start	
	Accuracy	Time	Accuracy	Time	Accuracy	Time
9	0.1306	1	0.0036	500	0	346
20	0.1541	1	0.0042	500	0	77
49	0.1915	1	0.0015	500	0	87
57	0.1933	1	0	500	0	2

Table 2.1: Quality of upper bounds for Problem (2.1) for the Diabetes dataset, for different values of k. We see that the MIO equipped with warm starts deliver the best upper bounds in the shortest overall times. The run time for the MIO with warm start includes the time taken by the discrete first order method (which were all less than a second).

Using the discrete first order methods in combination with the MIO algorithm resulted in finding the best possible relative accuracy in a matter of a few minutes.

Improving MIO Performance via Warm Starts

We performed a series of experiments on the Diabetes dataset to obtain a globally optimal solution to Problem (2.1) via our approach and to understand the implications of using advanced warm starts to the MIO formulation in terms of certifying optimality. For each choice of k we ran Algorithm 2 with fifty random initializations. They took less than a few seconds to run. We used the best solution as an advanced warm start to the MIO formulation (2.9). For each of these examples, we also ran the MIO formulation without any warm start (we refer to this as "cold start"). Figure 2-2 summarizes the results. The figure shows that in the presence of warm starts, the MIO closes the optimality gap significantly faster than those without advanced warm starts.

Statistical Performance

We considered datasets as described in Example 1, Section 2.4.1—we took different values of n, p with $n > p$, ρ with $k_0 = 10$.

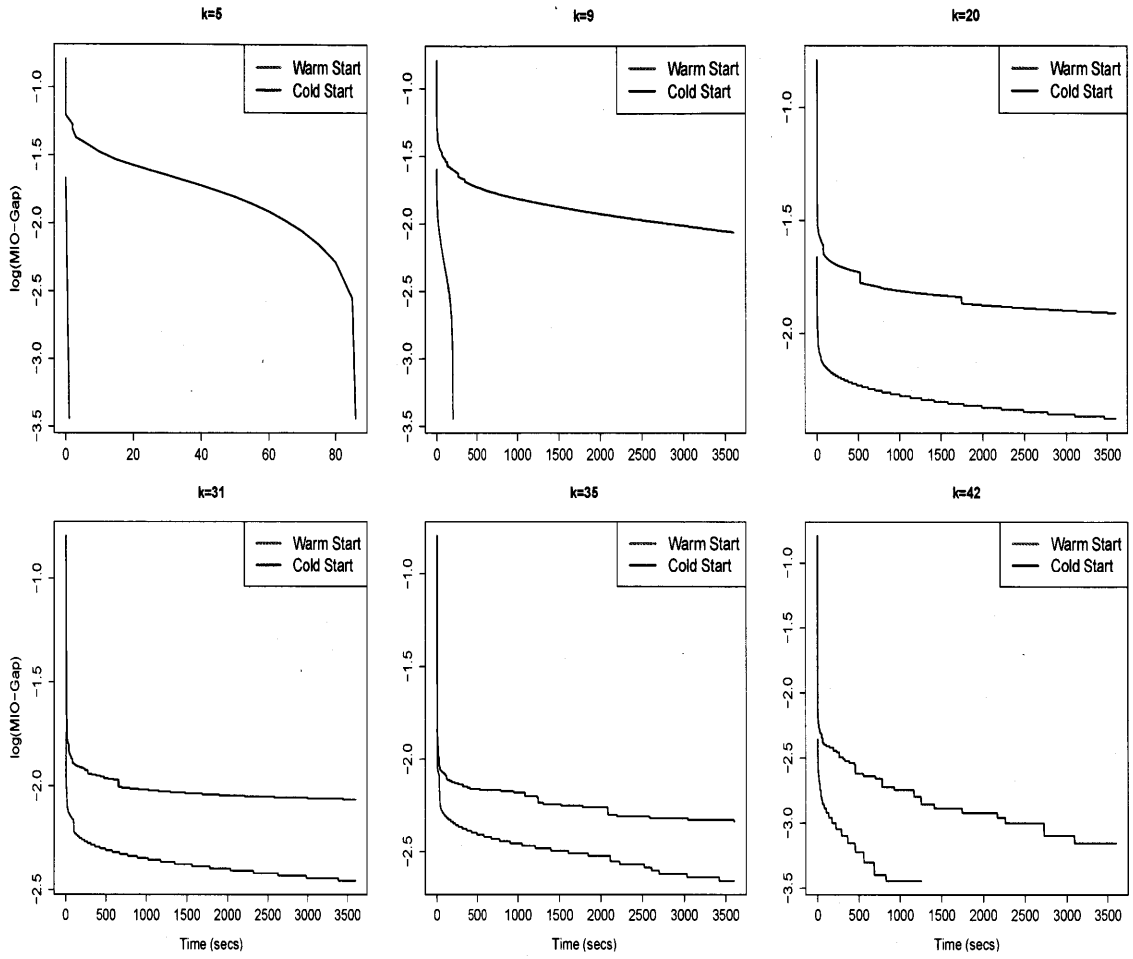

Figure 2-2: The evolution of the MIO optimality gap (in $\log_{10}(\cdot)$ scale) for Problem (2.1), for the Diabetes dataset with $n = 350, p = 64$ with and without warm starts for different values of k. The MIO significantly benefits by advanced warm starts delivered by Algorithm 2. In all of these examples, the global optimum was found within a very small fraction of the total time, but the proof of global optimality came later. As the number of possible solutions grows as $\binom{p}{k}$, it takes longer to prove optimality for $k = 31, 35$ compared to $k = 42$.

Competing Methods and Performance Measures For every example, we considered the following learning procedures for comparison purposes: (a) the MIO approach equipped warm starts from Algorithm 2 (annotated as "MIO" in the figure), (b) the Lasso, (c) Sparsenet and (d) stepwise regression (annotated as "Step" in the figure).

We used R to compute Lasso, Sparsenet and stepwise regression using the glmnet 1.7.3,

Sparsenet and Stats 3.0.2 packages respectively, which were all downloaded from CRAN at `http://cran.us.r-project.org/`.

For each procedure, we obtained the "optimal" tuning parameter by selecting the model that achieved the best predictive performance on a held out validation set. Once the model $\widehat{\boldsymbol{\beta}}$ was selected, we obtained the prediction error as:

$$\text{Prediction Error} = \|\mathbf{X}\widehat{\boldsymbol{\beta}} - \mathbf{X}\boldsymbol{\beta}^0\|_2^2 / \|\mathbf{X}\boldsymbol{\beta}^0\|_2^2. \tag{2.47}$$

We report "prediction error" and number of non-zeros in the optimal model in our results. The results were averaged over ten random instances, for different realizations of $\mathbf{X}, \epsilon$. For every run: the training and validation data had a fixed $\mathbf{X}$ but random noise ϵ.

Figure 2-3 presents results for data generated as per Example 1 with $n = 500$ and $p = 100$. We see that the MIO procedure performs very well across all the examples. Among the methods, MIO performs the best, followed by Sparsenet, Lasso with Step(wise) exhibiting the worst performance. In terms of prediction error, the MIO performs the best, only to be marginally outperformed by Sparsenet in a few instances. This further illustrates the importance of using non-convex methods in sparse learning. Note that the MIO approach, unlike Sparsenet certifies global optimality in terms of solving Problem 2.1. However, based on the plots in the upper panel, Sparsenet selects a few redundant variables unlike MIO. Lasso delivers quite dense models and pays the price in predictive performance too, by selecting wrong variables. As the value of SNR increases, the predictive power of the methods improve, as expected. The differences in predictive errors between the methods diminish with increasing SNR values. With increasing values of ρ (from left panel to right panel in the figure), the number of non-zeros selected by the Lasso in the optimal model increases.

We also performed experiments with $n = 1000, p = 50$ for data generated as per Example 1. We solved the problems to provable optimality and found that the MIO performed very well when compared to other competing methods. We do not report the experiments for brevity.

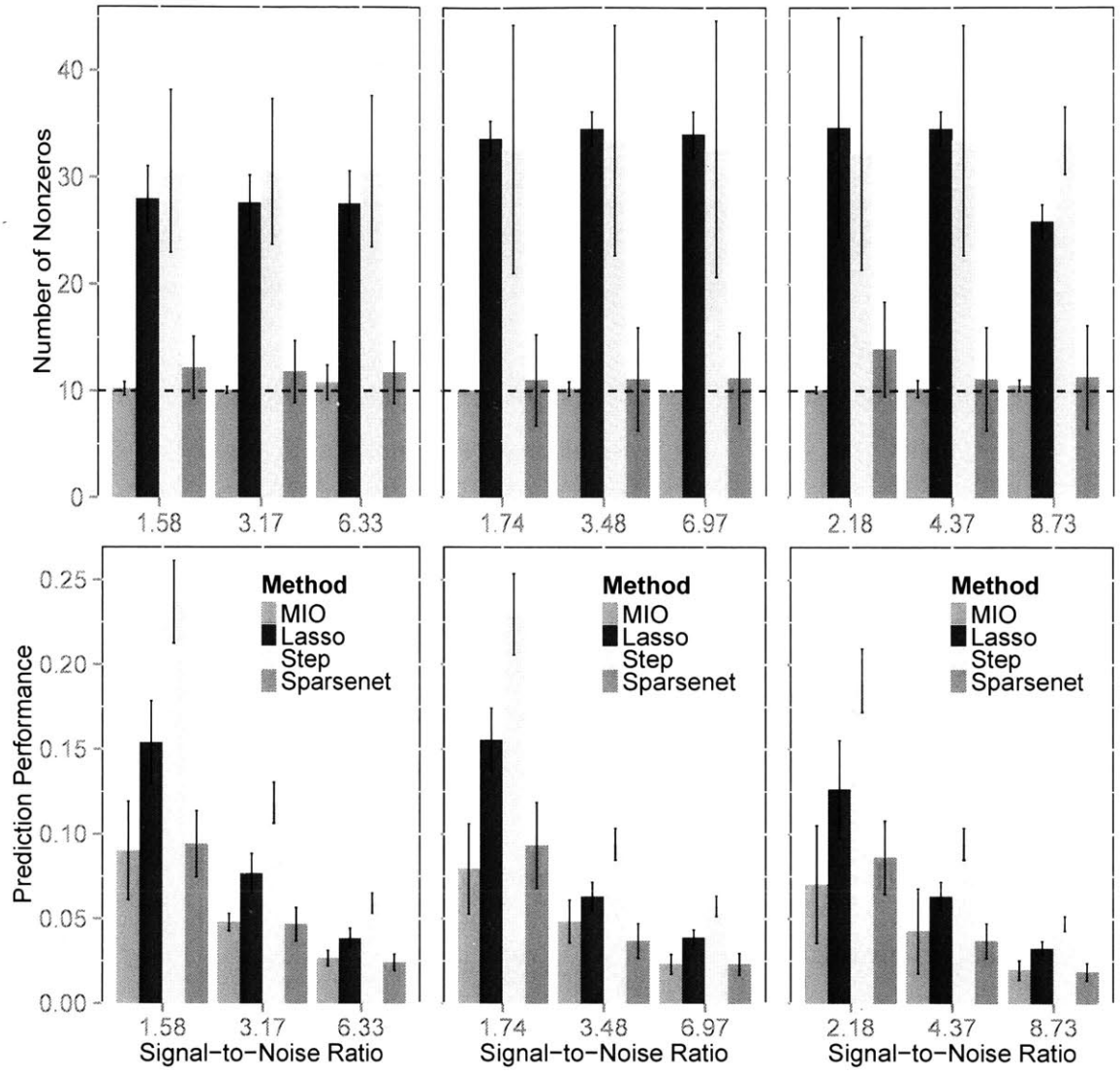

Figure 2-3: Figure showing the sparsity (upper panel) and predictive performances (bottom panel) for different subset selection procedures for the least squares loss. Here, we consider data generated as per Example 1, with $n = 500, p = 100$, $k_0 = 10$, for three different SNR values with [Left Panel] $\rho = 0.5$, [Middle Panel] $\rho = 0.8$, and [Right Panel] $\rho = 0.9$. The dashed line in the top panel represents the true number of nonzero values. For each of the procedures, the optimal model was selected as the one which produced the best prediction accuracy on a separate validation set, as described in Section 2.4.2.

MIO model training

We trained a sequence of best subset models (indexed by k) by applying the MIO approach with warm starts. Instead of running the MIO solvers from scratch for different values of k,

we used *callbacks*, a feature of integer optimization solvers. Callbacks allow the user to solve an initial model, and then add additional constraints to the model one at a time. These "cuts" reduce the size of the feasible region without having to rebuild the entire optimization model. Thus, in our case, we can save time by building the initial optimization model for $k = p$. Once the solution for $k = p$ is obtained, a cut can be added to the model: $\sum_{i=1}^{p} z_i \leq k$ for $k = p - 1$ and the model can be re-solved from this point. We apply this procedure until we arrive at a model with $k = 1$.

For each value of k tested, the MIO best subset algorithm was set to stop the first time either an optimality gap of 1% was reached or a time limit of 15 minutes was reached. Additionally, we only tested values of k from 5 through 25, and used Algorithm 2 to warm start the MIO algorithm. We observed that it was possible to obtain speedups of a factor of 2-4 by carefully tuning the optimization solver for a particular problem, but chose to maintain generality by solving with default parameters. Thus, we do not report times with the intention of accurately benchmarking the best possible time but rather to show that it is computationally tractable to solve problems to optimality using modern MIO solvers.

2.4.3 The High-Dimensional Regime: $p \gg n$

In this section, we investigate

(a) the evolution of upper bounds in the high-dimensional regime,

(b) the effect of a bounding box formulation on the speed of closing the optimality gap,

(c) the statistical performance of the MIO approach in comparison to other state-of-the art methods

Obtaining Good Upper Bounds

We performed tests similar to those in Section 2.4.2 for the $p \gg n$ regime. We tested a synthetic dataset corresponding to Example 2 with $n = 30, p = 2000$ for varying SNR values (see Table 2.2) over a time of 500s. As before, using the discrete first order methods in

combination with the MIO algorithm resulted in finding the best possible upper bounds in the shortest possible times.

	k	Discrete First Order		MIO Cold Start		MIO Warm Start	
		Accuracy	Time	Accuracy	Time	Accuracy	Time
SNR = 3	5	0.1647	37.2	1.0510	500	0	72.2
	6	0.6152	41.1	0.2769	500	0	77.1
	7	0.7843	40.7	0.8715	500	0	160.7
	8	0.5515	38.8	2.1797	500	0	295.8
	9	0.7131	45.0	0.4204	500	0	96.0
SNR = 7	5	0.5072	45.6	0.7737	500	0	65.6
	6	1.3221	40.3	0.5121	500	0	82.3
	7	0.9745	40.9	0.7578	500	0	210.9
	8	0.8293	40.5	1.8972	500	0	262.5
	9	1.1879	44.2	0.4515	500	0	254.2

Table 2.2: The quality of upper bounds for Problem (2.1) obtained by Algorithm 2, MIO with cold start and MIO warm-started with Algorithm 2. We consider the synthetic dataset of Example 2 with $n = 30, p = 2000$ and different values of SNR. The MIO method, when warm-started with the first order solution performs the best in terms of getting a good upper bound in the shortest time. The metric "Accuracy" is defined in (2.46). The first order methods are fast but need not lead to highest quality solutions on their own. MIO improves the quality of upper bounds delivered by the first order methods and their combined effect leads to the best performance.

We also did experiments on the Leukemia dataset. In Figure 2-4 we demonstrate the evolution of the objective value of the best subset problem for different values of k. For each value of k, we warm-started the MIO with the solution obtained by Algorithm 2 and allowed the MIO solver to run for 4000 seconds. The best objective value obtained at the end of 4000 seconds is denoted by f_*. We plot the Relative Accuracy, i.e., $(f_t - f_*)/f_*$, where f_t is the objective value obtained after t seconds. The figure shows that the solution obtained by Algorithm 2 is improved by the MIO on various instances and the time taken to improve the upper bounds depends upon k. In general, for smaller values of k the upper bounds obtained by the MIO algorithm stabilize earlier, i.e., the MIO finds improved solutions faster than for larger values of k.

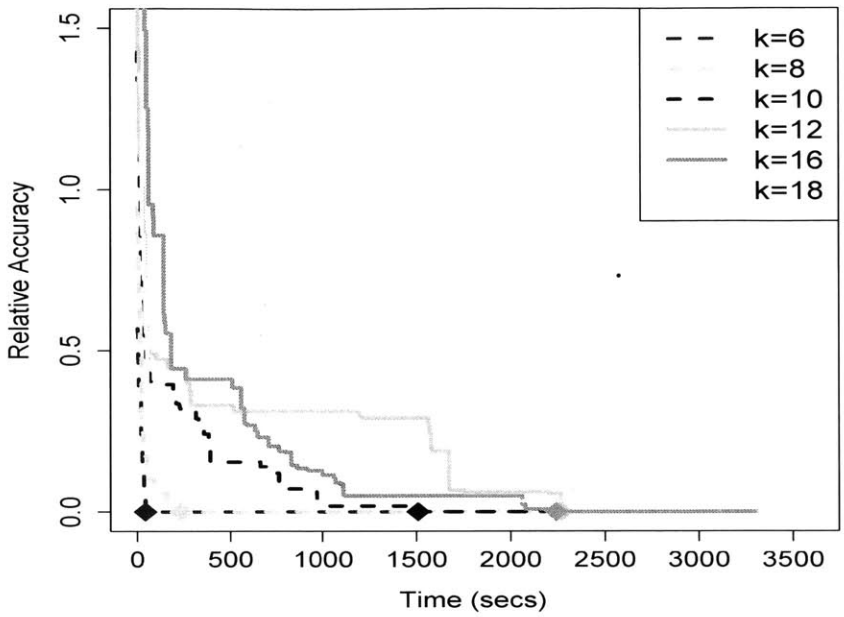

Figure 2-4: Behavior of MIO aided with warm start in obtaining good upper bounds over time for the Leukemia dataset ($n = 72, p = 1000$). The vertical axis shows relative accuracy, i.e., $(f_t - f_*)/f_*$, where f_t is the objective value obtained after t seconds and f_* denotes the best objective value obtained by the method after 4000 seconds. The colored diamonds correspond to the locations where the MIO (with warm start) attains the best solution. The figure shows that MIO improves the solution obtained by the first order method in all the instances. The time at which the best possible upper bound is obtained depends upon the choice of k. Typically larger k values make the problem harder—hence the best solutions are obtained after a longer wait.

Bounding Box Formulation

With the aid of advanced warm starts as provided by Algorithm 2, the MIO obtains a very high quality solution very quickly—in most of the examples the solution thus obtained turns out to be the global minimum. However, in the typical "high-dimensional" regime, with $p \gg n$, we observe that the certificate of global optimality comes later as the lower bounds of the problem "evolve" slowly. This is observed even in the presence of warm starts and using the implied bounds as developed in Section 2.2.1 and is aggravated for the cold-started MIO formulation (2.10).

To address this, we consider the MIO formulation (2.48) obtained by adding bounding

boxes around a local solution. These restrictions *guide* the MIO in restricting its *search* space and enable the MIO to certify global optimality inside that bounding box. We consider the following additional bounding box constraints to the MIO formulation (2.10):

$$\left\{\boldsymbol{\beta} : \|\mathbf{X}\boldsymbol{\beta} - \mathbf{X}\boldsymbol{\beta}_0\|_1 \leq \mathcal{L}^{\zeta}_{\ell,\text{loc}}\right\} \cap \left\{\boldsymbol{\beta} : \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_1 \leq \mathcal{L}^{\beta}_{\ell,\text{loc}}\right\},$$

where $\boldsymbol{\beta}_0$ is a candidate sparse solution. The radii of the two ℓ_1-balls above, namely, $\mathcal{L}^{\zeta}_{\ell,\text{loc}}$ and $\mathcal{L}^{\beta}_{\ell,\text{loc}}$ are user-defined parameters and control the size of the feasible set.

Using the notation $\boldsymbol{\zeta} = \mathbf{X}\boldsymbol{\beta}$ we have the following MIO formulation (equipped with the additional bounding boxes):

$$\begin{aligned}
\min_{\boldsymbol{\beta},\mathbf{z},\boldsymbol{\zeta}} \quad & \tfrac{1}{2}\,\boldsymbol{\zeta}^T\boldsymbol{\zeta} - \langle \mathbf{X}'\mathbf{y}, \boldsymbol{\beta}\rangle + \tfrac{1}{2}\,\|\mathbf{y}\|_2^2 \\
s.t. \quad & \boldsymbol{\zeta} = \mathbf{X}\boldsymbol{\beta} \\
& (\beta_i, 1 - z_i) : \text{SOS type-1}, \;\; i = 1, \ldots, p \\
& z_i \in \{0,1\}, i = 1, \ldots, p \\
& \sum_{i=1}^{p} z_i \leq k \\
& -\mathcal{M}_U \leq \beta_i \leq \mathcal{M}_U, i = 1, \ldots, p \\
& \|\boldsymbol{\beta}\|_1 \leq \mathcal{M}_\ell \\
& -\mathcal{M}^{\zeta}_U \leq \zeta_i \leq \mathcal{M}^{\zeta}_U, i = 1, \ldots, n \\
& \|\boldsymbol{\zeta}\|_1 \leq \mathcal{M}^{\zeta}_\ell \\
& \|\boldsymbol{\zeta} - \boldsymbol{\zeta}_0\|_1 \leq \mathcal{L}^{\zeta}_{\ell,\text{loc}} \\
& \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_1 \leq \mathcal{L}^{\beta}_{\ell,\text{loc}}.
\end{aligned} \tag{2.48}$$

For large values of $\mathcal{L}^{\zeta}_{\ell,\text{loc}}$ (respectively, $\mathcal{L}^{\beta}_{\ell,\text{loc}}$) the constraints on $\mathbf{X}\boldsymbol{\beta}$ (respectively, $\boldsymbol{\beta}$) become

ineffective and one gets back formulation (2.10). To see the impact of these additional cutting planes in the MIO formulation, we consider a few examples as illustrated in Figures 2-5,2-6,2-7.

Interpretation of the bounding boxes. A local bounding box in the variable $\zeta = \mathbf{X}\beta$ directs the MIO solver to seek for candidate solutions that deliver models with predictive accuracy "similar" (controlled by the radius of the ball) to a reference predictive model, given by ζ_0. In our experiments, we typically chose ζ_0 as the solution delivered by running MIO (warm-started with a first order solution) for a few hundred to a few thousand seconds. More generally, ζ_0 may be selected by any other sparse learning method. In our experiments, we found that the run-time behavior of the MIO depends upon how correlated the columns of $\mathbf{X}$ are.

Similarly, a bounding box around β directs the MIO to look for solutions in the neighborhood of a reference point β_0. In our experiments, we chose the reference β_0 as the solution obtained by MIO (warm-started with a first order solution) and allowing it to run for a few hundred to a few thousand seconds. We observed in our experiments that the MIO solver in presence of bounding boxes in the β-space certified optimality and in the process finding better solutions; much faster than the ζ-bounding box method.

Note that the β-bounding box constraint leads to $O(p)$ and the ζ-box leads to $O(n)$ constraints. Thus, when $p \gg n$ the additional ζ constraints add a fewer number of extra variables when compared to the β constraints.

Experiments In the first set of experiments, we consider the Leukemia dataset with $n = 72, p = 1000$. We took two different values of $k \in \{5, 10\}$ and for each case we ran Algorithm 2 with several random restarts. The best solution thus obtained was used to warm start the MIO formulation (2.10), which we ran for an additional 3600 seconds. The solution thus obtained is denoted by β_0. We then consider formulation (2.48) with $\mathcal{L}^{\zeta}_{\ell,\text{loc}} = \infty$ and different values of $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \text{Frac}$ (as annotated in Figure 2-5) — the results are displayed in Figure 2-5.

We consider another set of experiments to demonstrate the performance of the MIO

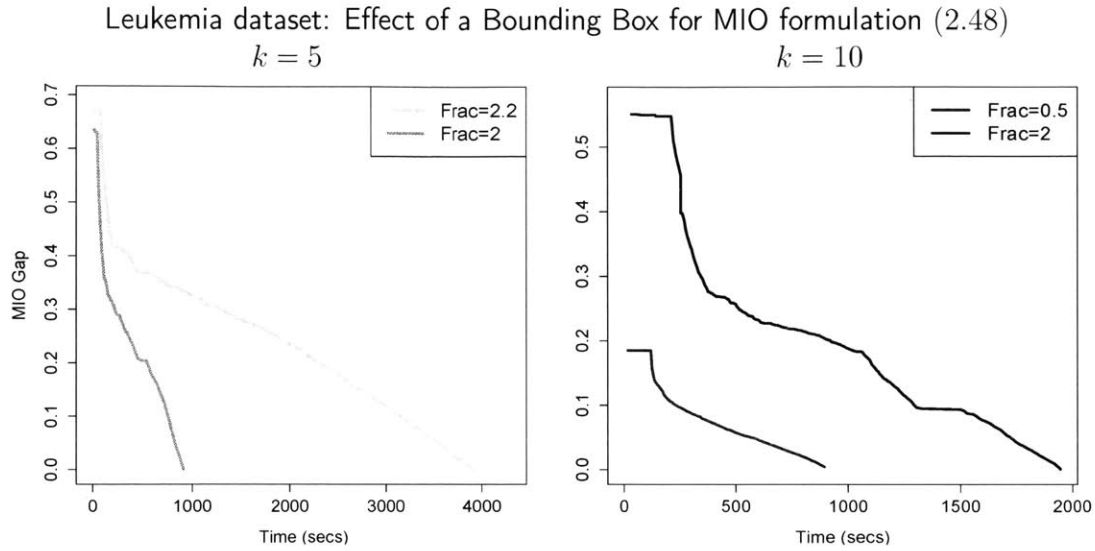

Figure 2-5: The effect of the MIO formulation (2.48) for the Leukemia dataset, for different values of k. Here $\mathcal{L}^{\zeta}_{\ell,\text{loc}} = \infty$ and $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \text{Frac}$. For each value of k, the global minimum obtained was the same for the different choices of $\mathcal{L}^{\beta}_{\ell,\text{loc}}$.

in certifying global optimality for different synthetic datasets with varying n, p, k as well as with different structures on the bounding box. In the first case, we generated data as per Example 1 with $\rho = 0.9$, $k_0 = 5$. We consider the case with $\boldsymbol{\zeta}_0 = \mathbf{X}\boldsymbol{\beta}_0$, $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \infty$ and $\mathcal{L}^{\zeta}_{\ell,\text{loc}} = 0.5\|\mathbf{X}\boldsymbol{\beta}_0\|_1$, where $\boldsymbol{\beta}_0$ is a k-sparse solution obtained from the MIO formulation (2.10) run with a time limit of 1000 seconds, after being warm-started with Algorithm 2. The results are displayed in Figure 2-6[Left Panel]. In the second case (with data same as before) we obtained $\boldsymbol{\beta}_0$ in the same fashion as described before—we took a bounding box around $\boldsymbol{\beta}_0$, and left the box constraint around $\mathbf{X}\boldsymbol{\beta}_0$ inactive, i.e., we set $\mathcal{L}^{\zeta}_{\ell,\text{loc}} = \infty$ and $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \|\boldsymbol{\beta}_0\|_1/k$. We performed two sets of experiments, where the data were generated based on different SNR values—the results are displayed in Figure 2-6 with SNR=1 [Middle Panel] and SNR = 3[Right Panel].

In the same vein, we have Figure 2-7 studying the effect of formulations (2.48) for synthetic datasets generated as per Example 1 with $n = 50, p = 1000, \rho = 0.9$ and $k_0 = 5$.

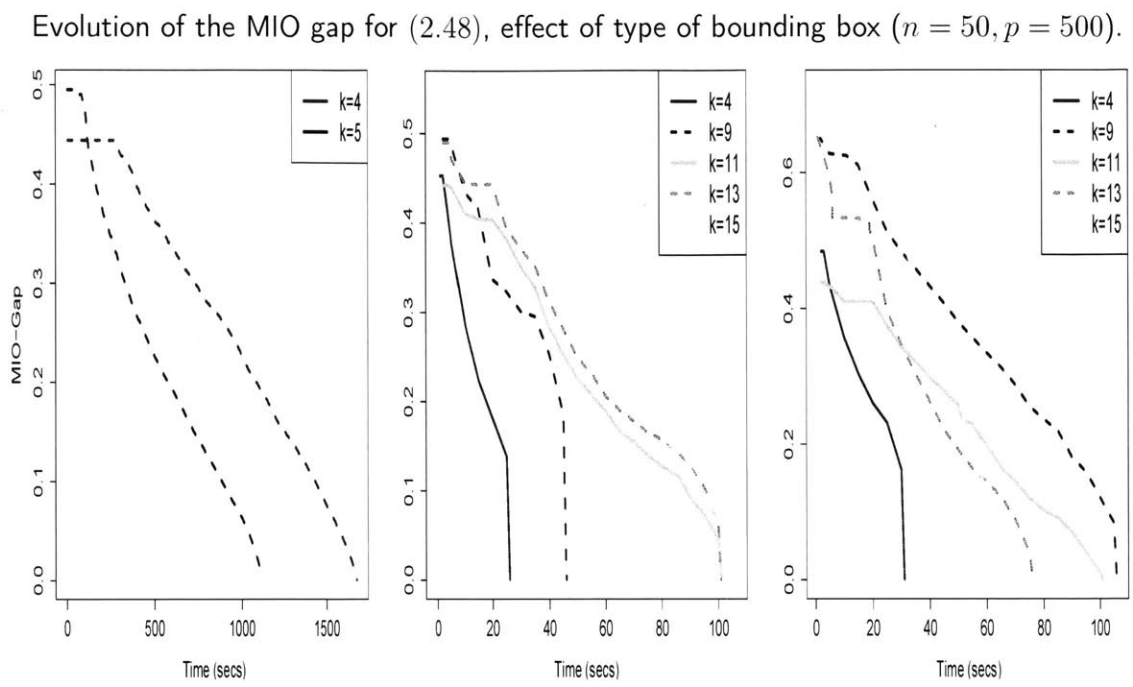

Figure 2-6: The effect of the MIO formulation (2.48) for a synthetic dataset as in Example 1 with $\rho = 0.9$, $k_0 = 5$, $n = 50, p = 500$, for different values of k. [Left Panel] $\mathcal{L}^{\zeta}_{\ell,\text{loc}} = 0.5\|\mathbf{X}\boldsymbol{\beta}_0\|_1$ and $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \infty$ for a data-set with SNR $= 3$. [Middle Panel] $\mathcal{L}^{\zeta}_{\ell,\text{loc}} = \infty$, $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \|\boldsymbol{\beta}_0\|_1/k$ and SNR $= 1$. [Right Panel] $\mathcal{L}^{\zeta}_{\ell,\text{loc}} = \infty$, $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \|\boldsymbol{\beta}_0\|_1/k$ and SNR $= 3$. The figure shows that the bounding boxes in terms of $\mathbf{X}\boldsymbol{\beta}$ (left-panel) make the problem harder to solve, when compared to bounding boxes around $\boldsymbol{\beta}$ (middle and right panels). A possible reason is due to the strong correlations among the columns of $\mathbf{X}$. The SNR values do not seem to have a big impact on the run-times of the algorithms (middle and right panels).

Statistical Performance

To understand the statistical behavior of MIO when compared to other approaches for learning sparse models, we considered synthetic datasets for values of n ranging from $30 - 50$ and values of p ranging from $1000 - 2000$. The following methods were used for comparison purposes

(a) Algorithm 2. Here we used fifty different random initializations around $\mathbf{0}$, of the form $\min(i-1, 1)N(\mathbf{0}_{p\times 1}, 4\mathbf{I}), i = 1, \ldots, 50$ and took the solution corresponding to the best objective value.

Evolution of the MIO gap for (2.48), effect of bounding box radii ($n = 50, p = 1000$).

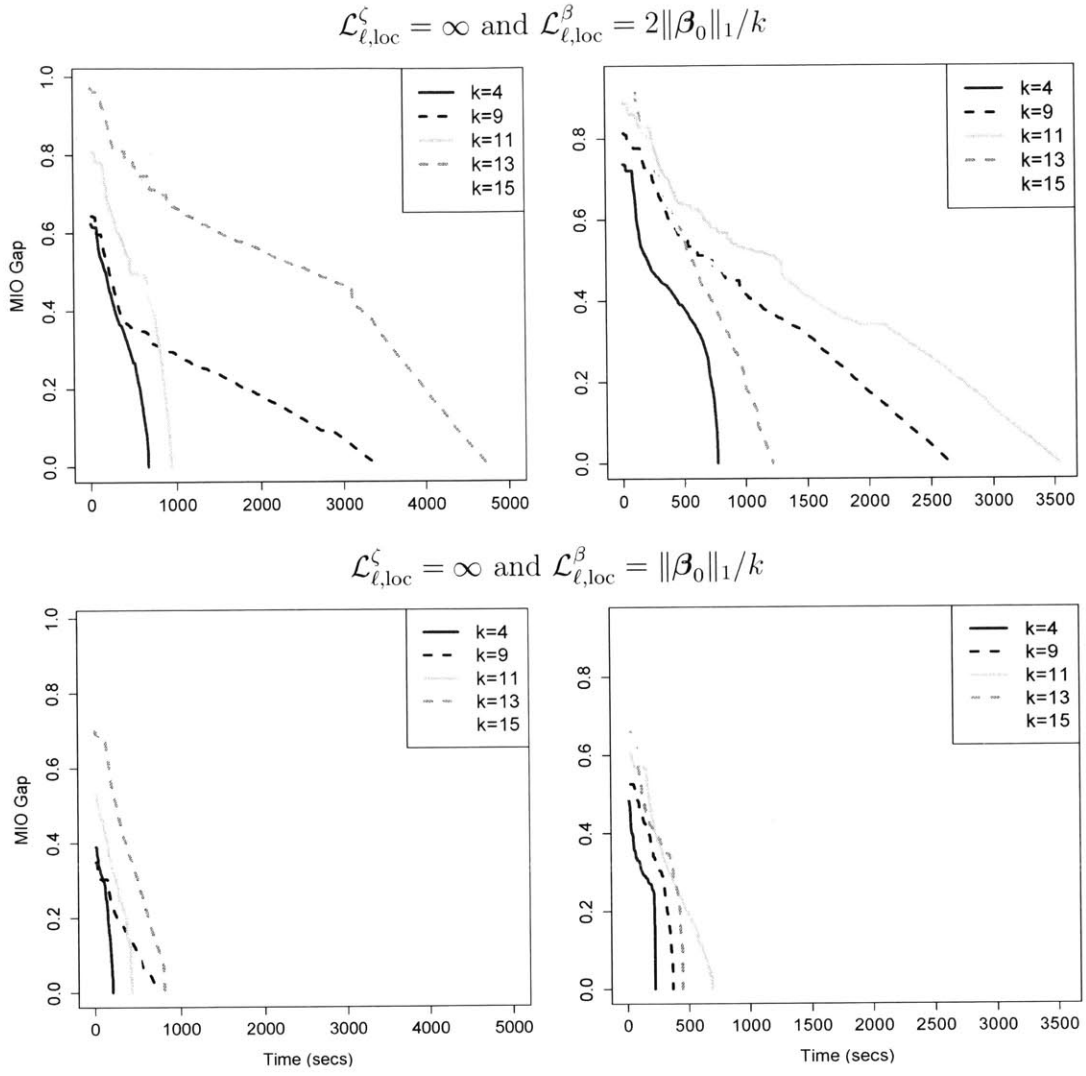

Figure 2-7: The evolution of the MIO gap with varying radii of bounding boxes for MIO formulation (2.48). The top panel has radii twice the size of the bottom panel. The dataset considered is generated as per Example 1 with $n = 50, p = 1000, \rho = 0.9$ and $k_0 = 5$ for different values of SNR: [Left Panel] SNR $= 1$, [Right Panel] SNR $= 3$. For each case, different values of k have been considered. The top panel has a bounding box radii which is twice the corresponding case in the lower panel. As expected, the times for the MIO gaps to close depends upon the radii of the boxes. The optimal solutions obtained were found to be insensitive to the choice of the bounding box radius.

(b) The MIO approach with warm starts from part (a).

(c) The Lasso solution.

(d) The Sparsenet solution.

For methods (a), (b) we considered ten equi-spaced values of k in the range $[3, 2k_0]$ (including the optimal value of k_0). For each of the methods, the best model was selected in the same fashion as described in Section 2.4.2 using separate validation sets.

In Figure 2-8 and Figure 2-9 we present selected representative results from four different examples described in Section 2.4.1.

In Figure 2-8 the left panel shows the performance of different methods for Example 1 with $n = 50, p = 1000, \rho = 0.8, k_0 = 5$. In this example, there are five non-zero coefficients: the features corresponding to the non-zero coefficients are weakly correlated and a feature having a non-zero coefficient is highly correlated with a feature having a zero coefficient. In this situation, the Lasso selects a very dense model since it fails to distinguish between a zero and a non-zero coefficient when the variables are correlated—it brings both the coefficients in the model (with shrinkage). MIO (with warm-start) performs the best—both in terms of predictive accuracy and in selecting a sparse set of coefficients. MIO obtains the sparsest model among the four methods and seems to find better solutions in terms of statistical properties than the models obtained by the first order methods alone. Interestingly, the "optimal model" selected by the first order methods is more dense than that selected by the MIO. The number of non-zero coefficients selected by MIO remains fairly stable across different SNR values, unlike the other three methods.

In Figure 2-8 the right panel shows Example 2, with $n = 30, p = 1000, k_0 = 5$ and all non-zero coefficients equal one. In this example, all the methods perform similarly in terms of predictive accuracy. This is because all non-zero coefficients in $\boldsymbol{\beta}^0$ have the same value. In fact for the smallest value of SNR, the Lasso achieves the best predictive model. In all the cases however, the MIO achieves the sparsest model with favorable predictive accuracy.

In Figure 2-9, for both the examples, the model matrix is an iid Gaussian ensemble. The underlying regression coefficient $\boldsymbol{\beta}^0$ however, is structurally different than Example 2 (as in

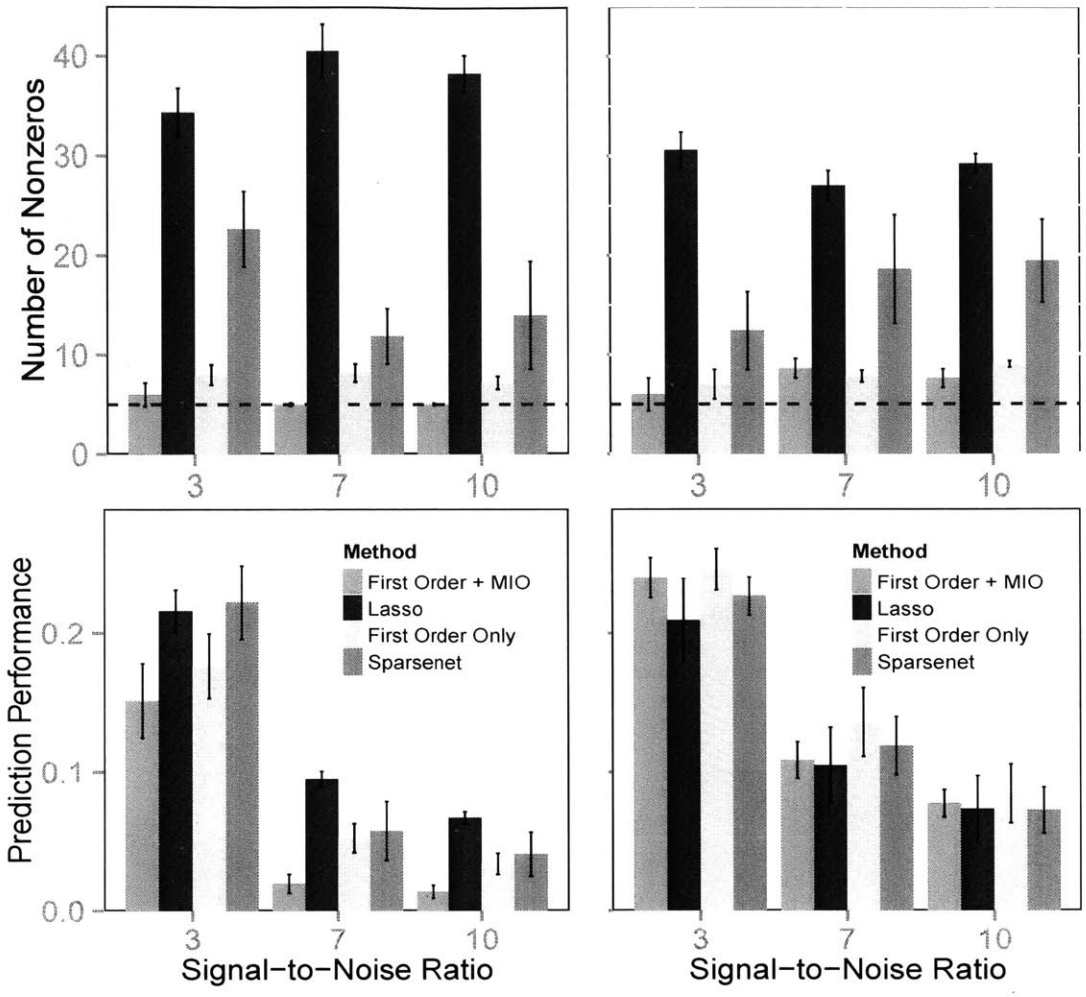

Figure 2-8: The sparsity and predictive performance for different procedures: [Left Panel] shows Example 1 with $n = 50, p = 1000, \rho = 0.8, k_0 = 5$ and [Right Panel] shows Example 2 with $n = 30, p = 1000$—for each instance several SNR values have been shown.

Figure 2-8, right-panel). The structure in $\boldsymbol{\beta}^0$ is responsible for different statistical behaviors of the four methods across Figures 2-8 (right-panel) and Figure 2-9 (both panels). The alternating signs and varying amplitudes of $\boldsymbol{\beta}^0$ are responsible for the poor behavior of Lasso. The MIO (with warm-starts) seems to be the best among all the methods. For Example 3 (Figure 2-9, left panel) the predictive performances of Lasso and MIO are comparable—the MIO however delivers much sparser models than the Lasso.

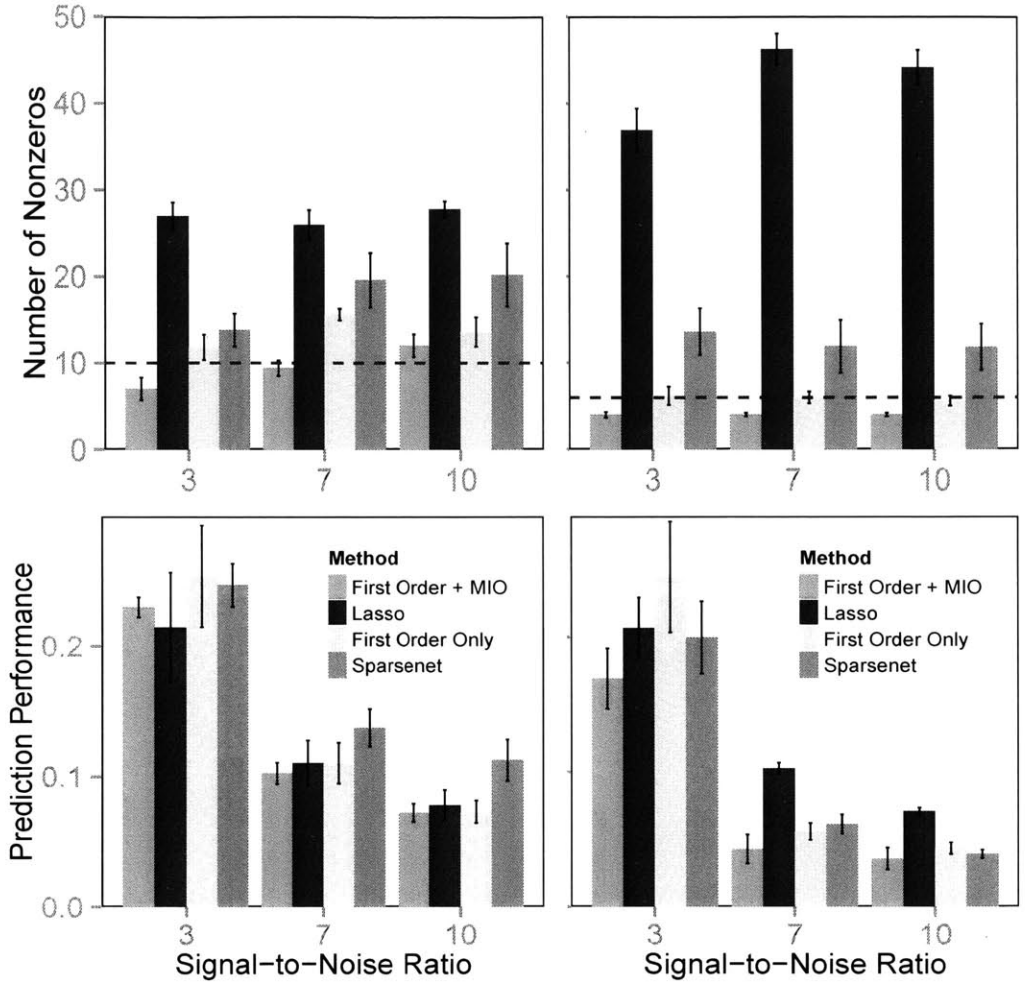

Figure 2-9: [Left Panel] Shows performance for data generated according to Example 3 with $n = 30, p = 1000$ and [Right Panel] shows Example 4 with $n = 50, p = 2000$.

The key conclusions are as follows:

1. The MIO best subset algorithm has a significant edge in detecting the correct sparsity structure for all examples compared to Lasso, Sparsenet and the stand-alone discrete first order method.

2. For data generated as per Example 1 with large values of ρ, the MIO best subset algorithm gives better predictive performance compared to its competitors.

3. For data generated as per Examples 2 and 3, MIO delivers similar predictive models like the Lasso, but produces much sparser models. In fact, Lasso seems to perform marginally better than MIO, as a predictive model for small values of SNR.

4. For Example 4, MIO performs the best both in terms of predictive accuracy and delivering sparse models.

2.5 Conclusions

We have revisited the classical best subset selection problem of choosing k out of p features in linear regression given n observations using a modern optimization lens, i.e., MIO and a discrete extension of first order methods from continuous optimization. Exploiting the astonishing progress of MIO solvers in the last twenty-five years, we have shown that this approach solves problems with n in the 1000s and p in the 100s in minutes to provable optimality, and finds near optimal solutions for n in the 100s and p in the 1000s in minutes. Importantly, the solutions provided by the MIO approach significantly outperform other state of the art methods like Lasso in achieving sparse models with good predictive power. Unlike all other methods, the MIO approach always provides a guarantee on its sub-optimality even if the algorithm is terminated early.

While continuous optimization methods have played and continue to play an important role in statistics over the years, discrete optimization methods have not. The evidence here as well as in [9] suggests that MIO methods are tractable and lead to desirable properties (improved accuracy and sparsity among others) at the expense of higher, but still reasonable, computational times.

Chapter 3

An Algorithmic Approach to Linear Regression

3.1 Introduction

We consider the linear regression model with response vector $\mathbf{y}_{n\times 1}$, model matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \Re^{n\times p}$, regression coefficients $\boldsymbol{\beta} \in \Re^{p\times 1}$ and errors $\boldsymbol{\epsilon} \in \Re^{n\times 1}$:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

The linear regression model is a powerful tool for modeling the relationship between a dependent variable and explanatory variables, and is well studied in theory as well as widely applied in practice. However, going from raw data to a high quality linear regression model is a non-trivial task; the modeler must ensure that all modeling assumptions are met, while building a parsimonious model that is able to separate signal from noise. The modeler rarely builds a single model. Rather, she undertakes an iterative process of refinement to produce the best model she can. This task manifests itself as a series of checks during the model building process: is there evidence of multicollinearity? of outliers? Are there too many variables present, or not enough? How well does the model generalize? What about

measurement error in the data, or missing data? Are the variables significant? Does the resulting model make sense for the application at hand? And so on.

The modeler must balance these competing objectives as she creates a regression model as best as she can. In this paper, we propose an algorithmic, optimization-based method for jointly balancing such objectives.

3.1.1 Aspirations

Currently, regression modeling is done in a fairly ad hoc manner. The various properties of a high quality linear regression model are typically built into the model one at a time and through repeated trial and error by the modeler. Hence, there is no guarantee that the final model produced satisfactorily addresses all of them, let alone optimally addresses them. The goal of this work is to design an optimization-based algorithm which simultaneously takes into account these desirable properties and, whenever it is not possible to satisfy all these properties simultaneously, the algorithm provides a guarantee that it is indeed infeasible to do so. The output of such an algorithm is a set of high quality regression models containing as many of the desired properties as possible.

We feel that humans and machines have different strengths and our proposed approach aims to utilize both these strengths. The modeler typically has subject matter expertise; for example, she may know of a particular structure present in the data, or can require that certain variables be present in the final model. While humans have intuition and contextual knowledge and understanding, computers have significantly more computational power. Our aspiration in this work is to empower modelers with a methodology that builds models with properties that a human modeler can require based on intuition and expertise.

3.1.2 Current Practice

Fitting regression models has long been viewed as an art, left to the savvy modeler who manages often-competing goals. The result is that two modelers may begin with the same set of data and end with quite different models.

We consulted several widely-used regression textbooks (*Regression Analysis by Example* by [28], *Applied Regression Analysis* by [41], *Linear Regression Analysis* by [96], and *Applied Linear Regression* by [110]) to see how modelers are instructed to approach the difficult task of fitting a linear regression model, and our findings show that while many textbooks discuss these competing objectives individually, most textbooks do not provide guidance to modelers on how to balance these objectives in organizing their search for the best model. For instance, [41], [96], and [110] each contain a chapter on model selection and discuss topics such as selection criteria, the best subset problem, stepwise methods, shrinkage methods, and computational approaches. Many techniques are offered, but little guidance is provided as to which method a modeler should use, if any, under particular circumstances. [28] also contains a chapter on model selection with a similar set of topics, but differs from the other texts in that it also provides a potential strategy for fitting regression models. In our experience, many modelers follow a process similar to what is outlined in [28]. We summarize their suggestions here, and henceforth refer to this as "the standard approach":

1. Examine the variables one by one, looking for outliers and making transformations.

2. Construct pairwise scatterplots for each variable, if possible. Examine the correlation matrix and delete redundant variables. Calculate the condition number of the correlation matrix to understand the extent of the effect of multicollinearity.

3. Fit the full ordinary least squares model and delete variables with insignificant t-tests. For the reduced model, examine the residuals for linearity, heteroscedasticity, autocorrelation, and outliers.

4. See if additional variables can be dropped and/or if new variables need to be brought in. Repeat step 3.

5. Check variance inflation factors (VIFs) and residual diagnostics.

6. Validate the fitted model on a test set, or using other methods such as cross validation, bootstrapping, etc.

In [28], the authors are quick to note that the procedure they outline is frequently implemented synchronously rather than entirely sequentially, and that it may be necessary to repeat the steps several times. They qualify their recommended steps by noting that "One important component that we have not included in our outlined steps is the subject matter knowledge of the analyst in the area in which the model is constructed. ... After all is said and done, statistical model building is an art. The techniques that we have described are the tools by which this task can be attempted methodically."

In contrast our goal is to design an algorithm that eliminates the modeler's tedious task of repeating the model-building steps several times and produce a high quality set of models.

3.1.3 Contribution and Structure

In this section of the thesis, we propose an MIQO approach to model a variety of desired properties in statistical models. In Table 3.1 we summarize the properties we model and how they are built into the MIQO model in Section 3.3. Our approach provides the only methodology we are aware of to construct models that impose statistical properties simultaneously. Using both real and synthetic data, we demonstrate that the approach is generally applicable, tractable in the sense of providing solutions in realistic timelines and provides a guarantee of suboptimality as it is based on a MIQO model. Specifically, when the MIQO is infeasible we obtain a guarantee that imposing distinct statistical properties is simply not feasible.

This chapter of the thesis is structured as follows. We begin in Section 3.2, by introducing and discussing the beneficial statistical properties we want the regression model to have. In Section 3.3, we develop the MIQO-based algorithm to impose these properties. In Section 3.4, we provide empirical evidence of our algorithm's abilities using a wide variety of real and synthetic datasets. We conclude this chapter in Section 3.5.

Table 3.1: Desirable properties of a linear regression model and how they are incorporated into the model.

Property	Chapter Section	MIQO Model
General Sparsity	3.2.1	Constraint (4.8d)
Group Sparsity	3.2.2	Constraint (4.8e)
Limited Pairwise Multicollinearity	3.2.2	Constraint (4.8f)
Nonlinear Transformations	3.2.2	Constraint (4.8g)
Robustness	3.2.3	Objective (4.8a)
Stable to Outliers	3.2.4	Objective (4.8a)
Modeler Expertise	3.2.5	Constraint (4.8h)
Statistical Significance	3.2.6	Constraint (4.8i)
Low Global Multicollinearity	3.2.7	Constraint (4.8i)

3.2 Desirable Properties of a Linear Regression Model

In this section, we review desirable characteristics of a linear regression model. We discuss our MIQO approach to build these properties into a model, and contrast it to other approaches to achieving each property.

3.2.1 General Sparsity

When the number of potential features is large, we often wish to identify a critical subset which are primarily responsible for producing the response. This leads to more interpretable models, and aids prediction accuracy by eliminating noise variables to increase the model's ability to generalize.

To achieve sparsity, we follow the approach we developed in Chapter 2 and use a combination of continuous and discrete optimization methods to efficiently solve the best subset regression problem [82]. That is, we will solve the following problem for all values of $k \in \{1, \ldots, p\}$ and return the solution and value of k with the smallest residual sum of squares:

$$\min_{\boldsymbol{\beta}} \ \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 \ \text{ subject to } \ \|\boldsymbol{\beta}\|_0 \leq k. \tag{3.1}$$

Due to the difficulty of scaling algorithms like leaps and bounds [52], research on the best subset regression problem in the past few decades has mainly focused on methods that

solve a convex approximation of Problem (3.1). For example, Lasso ([101], [29]) is a popular model that solves the following problem:

$$\min_{\boldsymbol{\beta}} \tfrac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1. \tag{3.2}$$

The ℓ_1 penalty on $\boldsymbol{\beta}$ in Problem (3.2) shrinks the coefficients towards zero and sets many coefficients to be exactly zero, which induces a sparse estimate of $\boldsymbol{\beta}$. Under sufficient regularity conditions, it has been shown that the sparsity pattern of this solution perfectly coincides with the true underlying sparsity pattern ([21]). However, the regularity conditions required to guarantee this are difficult to verify in practice, and are not typically satisfied by highly correlated data – which is a common occurrence in practice.

3.2.2 Selective Sparsity

We use the term "selective sparsity" to refer to settings where we would like to constrain the joint inclusion of subsets of independent variables. Modeling selective sparsity via MIQO can cover a broad range of settings and we will consider several here: group sparsity, pairwise multicollinearity, and nonlinear transformations.

Group Sparsity

Some applications exhibit a block- or group-sparse structure, with groups of independent variables whose coefficients are either all zero or all nonzero. Categorical variables, when expressed as a collection of dummy variables, form a natural group structure. Clear group formations also appear in compressed sensing ([45]), microarray analysis ([75]), and other applications.

By encoding this structure directly into the MIQO model, we ensure that the resulting solution preserves the group sparsity property. Moreover, MIQO can easily handle overlapping groups – a common phenomenon in microarray data, where some genes may play a role in several functional groups ([64]). Group sparsity has been highly studied in recent years (for example, see [115], [1], [122]). The most common approach is group lasso, proposed by

[115], and therefore much of the literature focuses on how well the group sparsity property is recovered. With an MIQO approach, a feasible solution guarantees the group sparsity property. To the best of our knowledge, there has not been previous work on group sparsity via MIQO.

Limited Pairwise Multicollinearity

A near-linear relationship between independent variables obfuscates the true contribution of each feature to the response and leads to unstable parameter estimates. In the worst case, multicollinearity can even cause parameter estimates to take the opposite sign from their true contribution to the response. To avoid these issues and produce interpretable models, a high quality regression model will contain features that are as orthogonal as possible. Of course, real data do not come with guarantees of orthogonality and are often far from this ideal. Thus, we suggest using pairwise correlation between independent variables as a measurement of multicollinearity, and building in selective sparsity by limiting the independent variables in the regression model to those which have relatively low pairwise correlation with the other included variables . This is a standard technique in practice – for example, in their textbook, [100] recommend that "independent variables with a pairwise correlation more than 0.70 should not be included in multiple regression analysis."

Principal Components Regression (PCR), proposed in [77], orthogonalizes the data matrix $\mathbf{X}$ via principal components analysis, and the response $\mathbf{y}$ is regressed upon the new, uncorrelated feature variables. While this effectively solves the issue of multicollinearity, it does not lend itself to interpretable models, as it is difficult to interpret the new features and therefore unclear the extent to which the original variables affect the response. Similarly, Partial Least Squares (PLS), proposed by [112] derives new, orthogonalized features, but differs from PCR in that the response is also used to create the new features. PLS models are similarly difficult to interpret. Penalized regression, which gives biased estimates but reduces variance, is another common method of attacking the inflated variances that result from multicollinearity. While this may induce lower variances, the shrinkage induced by

these methods does not actually make the data any less correlated, and hence we do not view it as an appropriate tool for encouraging interpretable models.

Detecting Appropriate Nonlinear Transformations

The data may not be collected in the units that are most explanatory of the dependent variable. It may turn out that a nonlinear transformation of an independent variable results in a new variable which can explain the variance in the dependent variable much better than the original measured variable could.

Typically, modelers detect the need for nonlinear transformations through graphical examination and trial and error. Alternatively, an automated method for detecting nonlinear transformations is the Box-Tidwell procedure ([20]), which is an iterative process that suggests statistically significantly power transformations of independent variables. The suggested transformations need to be interpreted by a human analyst before being incorporated into a model, since they are rarely interpretable whole-number powers. They also do not take into account other constraints we would like to include.

For a fixed set of nonlinear transformations, MIQO can optimally determine whether to use the original variable or a transformed version of the variable. For any variable j for which nonlinear transformations may be desired, we simply include all the potential transformed versions of the variable in the dataset passed to the algorithm. Let the set T_j contain the original variable j and its nonlinear transformations. Then we incorporate selective sparsity by including a constraint in the MIQO model that at most one of the variables from the set T_j can appear in the final model.

3.2.3 Robustness

Data quality varies widely based on the nature of the data being collected and the collection process. In [58], the authors define gross errors as errors due to a source of deviations which acts only occasionally but are quite powerful. They estimate that while high quality data may contain next to no gross errors, routine data usually contain $1-10\%$ gross errors. In addition

to the inaccuracies of gross errors, systematic errors may lead to imprecise measurements.

Robust optimization directly addresses errors in the data by considering uncertainty sets for the data and calculates solutions that are immune to worst-case uncertainty under these sets (see [3] and [4]). For the linear regression problem with data $(\mathbf{y}, \mathbf{X})$, the data associated with the independent variables have error $\mathbf{\Delta X}$ that belong to a given uncertainty set U. For example,

$$U = \left\{ \mathbf{\Delta X} \mid \|\mathbf{\Delta X}\|_{p-F} = \left(\sum_{i=1}^{n} \sum_{j=1}^{p} |\mathbf{\Delta X}_{i,j}|^p \right)^{1/p} \leq \Gamma. \right\}.$$

The robust least squares problem is then

$$\min_{\boldsymbol{\beta}} \max_{\mathbf{\Delta X} \in U} \ \tfrac{1}{2} \|\mathbf{y} - (\mathbf{X} + \mathbf{\Delta X})\boldsymbol{\beta}\|_p \quad . \tag{3.3}$$

The key result is as follows.

Theorem 3. *([6], [113]) Problem (3.3) is equivalent to*

$$\min_{\boldsymbol{\beta}} \ \tfrac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_p + \Gamma \|\boldsymbol{\beta}\|_{\frac{p}{p-1}}. \tag{3.4}$$

This result demonstrates that penalized regression models like Lasso are actually *robust* models against uncertainty in data. Although Lasso is revered for its ability to induce sparse solutions and much work has been done on the ability of Lasso to recover the true model (see [21] for an overview of the conditions under which Lasso identifies the true sparsity pattern), its predictive power is a result of being robust to uncertainty in data. In our algorithm, we regularize our MIQO model as a way to immunize the model from data uncertainty. As there are many cases in which regularization alone is not able to ensure sparsity ([90], [120], [78], [54], [118], [97]), we use the regularization approach in addition to the general sparsity approach outlined in Section 3.2.1. The robust optimization approach focuses on the worst-case error in the data. The approach is flexible in that it can handle different regularization parameters for different corresponding coefficients. For a characterization of the relation between robustification and regularization see [5].

3.2.4 Stability against Outliers

In the ideal modeling scenario, all data is representative of the population from which it is gathered. The presence of outliers can render a model's coefficient estimates useless, and can seriously impede the model's generalization ability. For the purpose of avoiding the effect of outliers we can use a median regression objective function rather than a least squares objective. The least squares objective is known to produce coefficients which are highly sensitive to outliers. Coefficient sensitivity to outliers is typically quantified using the metric of finite sample breakdown point [39]. The least squares objective leads to estimates with a limiting breakdown point of zero ([57]). The Least Absolute Deviations objective, which minimizes the ℓ_1-norm of the residuals rather than the ℓ_2-norm also has a breakdown point of zero. However, the Least Median of Squares (LMS) objective, introduced in [93], minimizes the median of the ℓ_2-norm of the residuals. The limiting breakdown point of LMS estimators is 50% – the maximum achievable. In [9], the authors formulate the LMS regression problem using MIQO. Their approach is easily adapted to our setting. To address outliers, we can adopt the LMS objective in place of the least squares objective in Problem (3.5), while retaining the other constraints and the regularization parameter in the objective.

3.2.5 Modeler Expertise

In some cases, the modeler has particular expertise with the application at hand. In that case, she might wish to specify that certain independent variables must be included in the final regression model, due to a known correlation with the response. This can be incorporated directly into the model building process by adding a constraint to the MIQO model.

3.2.6 Statistical Significance

Statistical inference relies not just on parameter estimates, but on specifications of uncertainty and confidence regarding those estimates. It is critical when interpreting parameters to have a sense of whether the model is truly detecting an underlying correlation between the

variable and the response. The standard way of quantifying this in the scientific literature is through the concept of statistical significance. An independent variable in a regression model is labeled as "statistically significant" if, in the presence of the other variables in the model, the probability α that the observed effect occurred by chance is low, conventionally 5% or less. Modelers typically exclude insignificant variables from regression models, because they can only give murky interpretations of their effects on the response. Of course, this occasionally results in Type II error – excluding a variable which does have a true effect – but the chances of this are minimized in a model that is not severely affected by multicollinearity, a property our MIQO model already builds in.

We would like our algorithm to provide confidence intervals and judge whether a given variable is statistically significant. However, we would also like our methodology to be free of distributional assumptions, to handle high dimensional settings, and to incorporate regularization, as described Section 3.2.3. All these properties invalidate the standard least squares assumptions. Our approach, then, is not to compromise these goals but rather to use bootstrapping techniques to generate confidence intervals and test for statistical significance. The bootstrap method was introduced in the seminal paper [43], and bootstrapped confidence intervals have been shown to be asymptotically more accurate than standard confidence intervals obtained using sample variance and normality assumptions ([34]).

Coming up with analytical formulae for significance measures and confidence intervals in regularized, potentially high-dimensional settings is challenging, and an area of current research (see [65] and [72], for example). We prefer our methodology to be flexible and able to handle a variety of objective functions and constraints, and so we opt for a bootstrapping approach instead, which harnesses the power of modern computing.

3.2.7 Low Global Multicollinearity

We note that it is possible to have multicollinearity without having any high pairwise correlations; see [94] for an example where four variables all have pairwise correlation ≤ 0.57 but have a perfect linear relationship. Thus, using a pairwise correlation threshold as a surrogate

for eliminating multicollinearity may not catch all cases of multicollinearity.

Global multicollinearity can be measured by checking the condition number of the correlation matrix resulting from the submatrix of included variables. A high condition number indicates a multicollinearity problem. A condition number greater than 15 is usually taken as evidence of multicollinearity and a condition number greater than 30 is usually an instance of severe multicollinearity ([28]).

3.3 Algorithm

In this section, we describe our algorithm for producing high quality regression models. The algorithm applies to any dataset that an analyst wishes to model using linear regression. The algorithm is composed of three stages: (1) preprocessing, (2) building and solving the MIQO model, and (3) generating any additional constraints and repeating step (2).

3.3.1 Stage 1: Preprocessing

The first stage begins with dataset preprocessing and parameter-setting. The dataset is split randomly 50%/25%/25% into a training, validation, and test set. Each set is standardized so that the training set has columns with zero mean and unit ℓ_2-norm. The modeler may also choose to set the number of robustification parameters Γ to be tested in the model (the default is 10), and ρ, the maximum pairwise correlation that will be allowed between included variables (the default is 0.8). The algorithm then generates the correlation matrix for the training data and identifies variables which are correlated in absolute value beyond ρ, and calls this set of pairs of variables $\mathcal{HC}$, for highly correlated variables. The algorithm identifies categorical variables and expresses them as groups of dummy variables. At this point, the modeler can specify any additional group-sparsity structure. We denote the m^{th} set of group-sparse variables as $\mathcal{GS}_m$. The modeler can specify a set of variables to be considered for a nonlinear transformation, and generates transformed versions of those variables. The default transformations for variable x are $x^2, x^{1/2}$, and $\log x$. We denote the m^{th} set of transformed

variables by $\mathcal{T}_m$. If the modeler believes the dataset to contain a significant number of outliers, he can specify at this point to use the median objective function, rather than the least squares objective. Finally, if the modeler has subject expertise, she can specify a set $\mathcal{I}$ of variables that must be included in the model. Then the algorithm calculates k_{max}, the maximum possible subset size such that the selective sparsity and modeler expertise constraints are still feasible. This is determined by solving a maximum independent set problem. We construct a graph containing vertices corresponding to each of the p potential variables and an edge between nodes i, j such that $(i, j) \in HC$. Then a maximum independent set, or stable set, for this graph is a set such that no two vertices are adjacent. The cardinality of this set is exactly equal to the maximum value of k that will result in a feasible MIO model, and is the objective value of the following MIO problem:

$$
\begin{aligned}
k_{max} = \max_{\mathbf{z}} \quad & \sum_{i=1}^{p} z_i \\
s.t. \quad & z_i + z_j \leq 1 \quad \forall (i,j) \in HC \\
& z_i \in \{0,1\}, i = 1, \ldots, p.
\end{aligned}
$$

If the maximum independent set problem produces an objective value of 0, the algorithm stops and reports to the user that there is no subset of variables satisfying the maximum correlation value specified. Otherwise, the algorithm sets the parameter k_{max} to the objective value, and then proceeds to determine a set of Γ values to test. By default, the set is logarithmically spaced between 0 and the value of Γ that would force $\boldsymbol{\beta} = 0$ if the problem were completely unconstrained. This allows a wide variety of robustification parameters to be tested. At this point, all the parameters of the algorithm have been set and the algorithm proceeds to Stage 2.

3.3.2 Stage 2: The MIQO model

The algorithm solves the following MIQO model for each value of k from 1 to k_{max} and each value of Γ using the training data $\mathbf{y}$ and $\mathbf{X}$.

$$\min_{\boldsymbol{\beta},\boldsymbol{z}} \quad \frac{1}{2}\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2+\Gamma\|\boldsymbol{\beta}\|_1, \tag{3.5a}$$

$$\text{s.t.} \quad z_\ell \in \{0,1\}, \quad \ell=1,\ldots,p, \tag{3.5b}$$

$$-\mathcal{M}z_\ell \leq \beta_\ell \leq \mathcal{M}z_\ell, \quad \ell=1,\ldots,p, \tag{3.5c}$$

$$\sum_{\ell=1}^{p} z_\ell \leq k, \tag{3.5d}$$

$$z_1=\ldots=z_\ell \quad (1,\ldots,\ell)\in\mathcal{GS}_m, \quad \forall m, \tag{3.5e}$$

$$z_i+z_j\leq 1 \quad \forall (i,j)\in\mathcal{HC}, \tag{3.5f}$$

$$\sum_{i\in\mathcal{T}_m} z_i \leq 1 \quad \forall m, \tag{3.5g}$$

$$z_\ell = 1 \quad \forall \ell\in\mathcal{I}, \tag{3.5h}$$

$$\sum_{\ell\in\mathcal{S}_i} z_\ell \leq |\mathcal{S}_i|-1 \quad \forall \mathcal{S}_1,\ldots,S_j. \tag{3.5i}$$

In the objective function (4.8a), the robustification parameter Γ immunizes the resulting model against uncertainty in the data. In Constraint (4.8b), a binary indicator variable z_ℓ is introduced for every β_ℓ in the model. For a large enough constant $\mathcal{M}$, the constraint (4.8c) ensures that β_ℓ will only be included in the model if $z_\ell = 1$[1]. The constraint (4.8d) limits the number of total variables that will be included in the model. This ensures general sparsity of the resulting model. The constraints in (4.8e), (4.8f), and (4.8g) are selective sparsity constraints. For the m^{th} set of variables with a group sparsity structure, the set of constraints defined in (4.8e) ensures that the variables in $\mathcal{GS}_m$ are either all zero, or all nonzero. The set of constraints in (4.8f) ensure that the resulting model is free from extreme pairwise multicollinearity. The set $\mathcal{T}_m$ refers to the m^{th} variable which was flagged as a

[1] $\mathcal{M}$ can be estimated from data (see Chapter 2 for details).

candidate for transformation and all of its possible nonlinear transformations. The set of constraints (4.8g) ensure that at most one of the variables from the set $\mathcal{T}_m$ will be included in the final model for each of the candidate variables m. If $\mathcal{I} \neq \emptyset$, Constraint (4.8h) will be included in the model and will ensure that each of the specified independent variables appears in the final model. (4.8i) is a set of constraints to exclude particular solutions $\mathcal{S}_i$, such as those with high global multicollinearity or containing variables which are statistically insignificant. $\mathcal{S}_i$ is the set of indices corresponding to nonzero $\boldsymbol{\beta}$ value in the i^{th} solution. The initial MIQO model will not contain line (4.8i); these constraints will be generated in Stage 3, if necessary.

Once the MIQO model is run for all potential values of k and Γ, the algorithm chooses the three sets of $\boldsymbol{\beta}$ with the highest R^2 on the validation set as the top three regression models, and proceeds to Stage 3.

3.3.3 Stage 3: Generating Additional Constraints

We denote the top three sets of $\boldsymbol{\beta}$ by $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$. For each of the sets $\mathcal{S}_i$, the algorithm computes the significance levels for each of the variables via bootstrap methods, and calculates the condition number of the model. If a set $\mathcal{S}_i$ produces undesirable results – a condition number higher than desired, or a model with insignificant variables – the algorithm generates the Constraint (4.8i) to exclude that set from the candidates of sets of best regression models.

Excluding set $\mathcal{S}_i$ can be achieved by "cutting off" the corner from the binary hypercube formed by the z variables using the constraint $\sum_{\ell \in \mathcal{S}_i} z_\ell \leq |S_i| - 1$. For example, to exclude set $\mathcal{S}_1 = \{1, 4, 7\}$, we can insert the constraint $z_1 + z_4 + z_7 \leq 2$ into Problem (3.5) and resolve.

The algorithm generates these additional constraints to exclude sets $\mathcal{S}_1, \ldots, \mathcal{S}_j$ as needed, and returns to Stage 2. The modeler may set the maximum condition number she will accept in the model, as well as the number of iterations she will permit between Stage 2 and Stage 3. The defaults are 30 and 3, respectively. In our experience, if a linear regression model is a good fit for the data, few iterations are necessary.

When the algorithm ends, it presents the top three models, along with their condition numbers and confidence intervals of the bootstrapped coefficients. Confidence histograms and diagnostic plots can also be generated.

3.3.4 Contrast with Current Practice

In many ways, our algorithm simply automates several of the steps outlined in the standard approach. However, we highlight a few key differences.

1. Our algorithm validates models out-of-sample, rather than relying on in-sample criteria. This ensures that the model selected does not overfit the training data.

2. Our algorithm does not have to choose which model properties to favor by performing the steps in a certain order; since it is based on optimization these properties can be addressed jointly rather than sequentially. For example, rather than noticing pairwise multicollinearity and preemptively deleting one variable, our MIQO model simply chooses which variable is best to delete in the course of the optimization.

3. Our algorithm is capable of handling datasets with more variables than a modeler can address manually. The steps suggested in [28] become difficult when p is large, and the modeler must often resort to a computational method for variable selection prior to performing the rest of the steps.

4. Our algorithm is capable of returning a set of high-quality models, rather than focusing on refining a single good model.

As an example we illustrate our algorithm's performance on two datasets and compare it to a model that a modeler might develop using these data.

3.3.5 Example 1

We compare and contrast our algorithm with the standard approach using the Croq'Pain dataset from [8].

The dataset originally comes from Croq'Pain, a French "restaurant rapide", and contains data on sixty Croq'Pain stores. For each store, the dataset provides information on the store and the surrounding area. There are a total of sixteen variables provided per store (see Table 3.2 for details).

Table 3.2: Variables in the Croq'Pain dataset.

Variable	Description
EARN	Operating earnings in $1000s
SIZE	Total area inside store
EMPL	Number of employees as of Dec. 31, 1994
P15	Number of 15-24 year olds in a 3 km radius
P25	Number of 25-34 year olds in a 3 km radius
P35	Number of 35-44 year olds in a 3 km radius
P45	Number of 45-54 year olds in a 3 km radius
P55	Number of people age 55+ in a 3 km radius
TOTAL	Total population in a 3 km radius
INC	Average income in town/neighborhood surrounding site
COMP	Number of competitors in 1 km radius
NCOMP	Number of restaurants that do not compete with Croq'Pain in a 1 km radius
NREST	Number of non-restaurant businesses in a 1 km radius
PRICE	Monthly rent per square meter of retail properties in the same locale
CLI	Cost of Living Index
K	Invested capital

The case described in [8] asks the student to use these data to build a regression model to help Croq'Pain decide whether to open a new store. The decision will be based on the store's performance ratio, which is measured as the ratio of operating earnings to invested capital. The goal is to build a high quality regression model with performance ratio as the dependent variable, and the first step of this – the fitted model with all independent variables included – is given in [8].

The Standard Approach

The model with all fourteen independent variables has an R^2 value of 0.867. Five of the fourteen variables are significant at the 0.05 level, and it seems that some of the coefficient

estimates may take the opposite signs from what is expected. For example, the coefficients for number of employees and for total surrounding population are both negative.

A quick look at the correlation matrix shows that there are a number of independent variables which are highly correlated: for example, P35 and TOTAL have a correlation coefficient of 0.96. However, the 14×14 matrix is unwieldy to work with manually. Instead of trying to eliminate correlated variables first, we begin to refine this model by removing variables that are insignificant at the 0.05 level, starting with those with the lowest t-value. Removing variables one at a time according to this method until all variables left are significant results in a new model with only five independent variables (SIZE, P15, INC, NREST, and PRICE) and an training set R^2 value of 0.856. At this point, the number of independent variables is low enough to investigate the correlation matrix manually. None of the remaining five independent variables have correlation over 0.18 in magnitude, so we feel assured that multicollinearity is not a problem in this reduced model. We "sign-check" each of the remaining five independent variables and validate that the signs agree with our intuition. We move on to residual diagnostics, and check for normality of the residuals by plotting a histogram and for heteroscedasticity by plotting each of the independent variables against the residuals. There is no evidence of non-normality or of heteroscedasticity. Therefore, we use this model as the final model.

MIQO-Based Approach

In the original case in [8], the students are first instructed to train their model using the entire dataset. The second part of the case asks them to rebuild using the first fifty data points to train the model and the last ten to validate. As our MIQO-based approach requires a training set and a validation set, we go with the second option.

We run our MIQO-based algorithm on the dataset using the default settings: 0.8 as the maximum pairwise correlation and 10 potential values of Γ. It takes less than 1 minute to run, and returns a model with five independent variables: SIZE, P15, INC, NREST, and PRICE; exactly the same five we chose via the standard approach. The first four variables

are significant at the 0.001 level, the last at the 0.01 level. The model has an out-of-sample validation set R^2 value of 0.80.

With the Croq'Pain data, the MIQO-based approach and the standard approach produced essentially the same model. In cases like these, we feel the main advantage of the MIQO-based approach is the amount of time saved from iterating through potential models. Although the computational time executing the MIQO-based approach is longer, the total time spent model-building is far shorter. In other cases, the standard approach may not lead to as clear of a path to a high-quality solution, or the dataset may contain enough variables to render it intractable for a human modeler. It is these cases for which the MIQO-based approach is not simply a time-saver, but a strong improvement over existing tools. The next example illustrates this case.

3.3.6 Example 2

We consider the Ames Housing Dataset ([32]). The dataset originally comes from the Ames City Assessor's Office and contains data on property sales in Ames, Iowa between 2006 and 2010. The variables include discrete, continuous, nominal, and ordinal variables which describe the quality and quantity of physical attributes of each property sold. The physical attributes measured include building type and style, square footage and lot details, quality and materials of the property's interior and exterior, and many more. The dataset was curated for use as a final group project in a semester-long regression class and is available along with full details in [32]. The prepared dataset contains 2930 observations and 80 variables. After expanding categorical variables into dummy variables and removing outliers and missing values, the final number of observations and variables is 2271 and 315, respectively. The potential project described in [32] asks the student to use these data to build a regression model to predict housing sale prices.

The Standard Approach

This dataset is large and complex enough that there is no single clear best model. Indeed, this is the motivation in [32] behind assigning this dataset as a final course project; the richness of the data leads to fruitful discussion of students' different approaches. [32] mentions that by using only the categorical variable for neighborhood and the two continuous variables that together comprise the property's total square footage leads to a model that explains 80% of the variability. At the other end of the spectrum, the author also admits to spending a fair amount of time constructing a 36-variable model (using some variables he created through recoding and interactions) that explains 92% of the variation in sales. [32] does not give further details of the model. While this may be overly complicated, it illustrates the challenge of building a high-quality regression model using the standard approach.

MIQO-Based Approach

After removing missing values and splitting the dataset into training, validation, and test sets, we ran our MIQO-based algorithm on the dataset using the default settings: 0.8 as the maximum pairwise correlation and 10 potential values of Γ.

The best model generated contained 20 independent variables, all of which were significant at the 0.05 level, and had a test set R^2 value of 0.920. This is competitive with the predictive power of the more complicated 36-variable model constructed by [32], while being more interpretable and still retaining statistically significant variables. The best MIQO model contained variables such as the overall quality and condition of the property, whether the property is identified as being in a particular neighborhood, the number of half bathrooms, whether the foundation of the home constructed from stone or not, the year the garage was built, various measurements of square footage, and the type of electrical system.

In cases like this, where the standard approach does not lead to an obvious single best model, the MIQO algorithm automatically balances desirable qualities such as complexity, predictive power, interpretability, and significance.

3.4 Computational Experiments

In this section, we illustrate our algorithm's capabilities by demonstrating its performance on a wide variety of datasets. We include datasets that are real as well as synthetic; that are from the classical overdetermined regime with $n > p$ as well as from the undetermined high-dimensional regime with $n < p$; and that contain various different structures and built-in properties. Our goal is to demonstrate that all of the desirable characteristics outlined in Section 3.2 can be achieved with MIQO in practical settings.

We begin by examining basic datasets, where all variables are continuous and there is no special structure. In such datasets, the main properties we would like to ensure are interpretability (via general sparsity and limited pairwise multicollinearity constraints) and robustness (via a regularization parameter in the objective function). We consider synthetic examples to highlight the algorithm's performance on these properties individually and real datasets in which we look at all three together. In each case, we compare our algorithm's performance to Lasso, as Lasso is designed to give interpretability and robustness.

We then go on to give results for datasets with additional features: datasets with the group sparsity property, with variables that need a nonlinear transformation, with outliers, and so on. We again compare our results to Lasso, and compare it also to the published approach taken by the modeler, if available, or to specific algorithms designed for the setting at hand (e.g. group lasso for the group sparsity case).

Synthetic Data

We generated data such that $\mathbf{x}_i \sim \mathrm{N}(\mathbf{0}, \mathbf{\Sigma}), i = 1, \ldots, 2n$ are independent realizations from a p-dimensional multivariate normal distribution with mean zero and covariance matrix $\mathbf{\Sigma} := (\sigma_{ij})$. The data was randomly split 50%/25%/25% into training, validation, and test set, respectively. The columns of the $\mathbf{X}$ matrix were standardized such that the training set had columns with zero mean and unit ℓ_2-norm. For a fixed $\mathbf{X}_{n \times p}$, we generated the response $\mathbf{y}$ as follows: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. We denote the number of nonzeros in $\boldsymbol{\beta}$ by k. The choice of $\mathbf{X}, \boldsymbol{\beta}, \sigma$ determines the Signal-to-Noise Ratio (SNR) of the problem, which

is defined as:

$$\text{SNR} = \frac{\text{var}(\mathbf{x}'\boldsymbol{\beta})}{\sigma^2}.$$

In particular, we took $\sigma_{ij} = \rho^{|i-j|}$ for $i, j \in \{1, \ldots, p\} \times \{1, \ldots, p\}$. In our experiments, we consider $k = 10$ and $\beta_i = 1$ for $i \in \{1, \ldots, p\}$ such that $i \bmod p/k = 0$ to generate k equally spaced values.

Real Data

We tested our algorithm on a number of publicly-available datasets. We obtained the datasets White Wine Quality, Red Wine Quality, Yacht Hydrodynamics, and CPU from the University of California Irvine Machine Learning Repository ([2]). We obtained the datasets Elevator, Pyrimidines, and Compact from a data repository at the University of Porto ([105]). We obtained the datasets LPGA 2008, LPGA 2009 and Airline Costs from a data repository at the University of Florida ([111]). We obtained the Diabetes dataset from the `lars` package within `R`. The HIV dataset comes from the study [91] and is available at [59].

Computational Specifications

All computational tests were performed on a Dell Precision T7600 computer with an Intel Xeon E5-2687W (3.1 GHz) processor, 16 cores, and 128 GB of RAM. We used `Gurobi 6.0.0` ([63]) as the optimization solver, and implemented the algorithm in `Julia 0.3.3` ([12]), a technical computing language. We used `JuMP 0.7.0` ([74]), an algebraic modeling language package for Julia, to interface with Gurobi. We used the `GLMNet 0.0.2` package in Julia to compute Lasso solutions. We used the `grplasso` package 0.4-4 in `R` ([89]) to compute group lasso solutions.

3.4.1 Basic Structure

Our main goals are to achieve interpretability and robustness, while retaining predictive power. In order to judge how well our algorithm achieves interpretability we will report on the size k of the subset chosen, the maximum pairwise correlation, and the condition

number of the final model. Although our algorithm returns the top three models, we only present results for the top model for brevity. To judge robustness and predictive power, we will report on the Γ chosen by the algorithm on the validation set and the test set R^2 value. We will compare these results to the size k of the subset chosen by Lasso, the maximum pairwise correlation in the Lasso model, and the test set R^2 value in the Lasso model. For the synthetic datasets, we also report the number of true positives achieved by each algorithm.

We aim to return solutions in practical amounts of time, so we imposed a time limit on each optimization problem solved: 20 seconds in the $n > p$ case and 40 seconds in the $n < p$ case. Often optimality is reached before the time limit. Note that for each dataset, $K_{max} \times$ (# of values of Γ tested) $\times$ (# of iterations of Stage 3) MIQO problems are solved.

We present results in Tables 3.3 - 3.8 for synthetic datasets for the default parameters of the algorithm: ten values of Γ tested and 0.8 as the maximum pairwise correlation allowed. Each experiment corresponds to two rows in a table. The top row presents average results over five trials of the same experiment and the bottom row presents the standard error. We use the following notation: SNR = signal-to-noise ratio, K^* = value of k chosen by the algorithm, TP = number of true nonzero variables identified by the algorithm, MC = the maximum pairwise correlation present in the final model, and Cond = condition number. Time for the MIQO algorithm is presented in hours, and is not meant to accurately benchmark the best possible time but to show that it is computationally tractable to solve these problems in a practical amount of time on standard computers.

Tables 3.3 and 3.4 show results for datasets designed to illustrate general sparsity, for the $n > p$ and $n < p$ case, respectively. Here we observe that the MIQO algorithm consistently identifies the true nonzero variables, and does not bring more than 1-2 additional noise variables into the model. In contrast, Lasso does correctly identify the true nonzero variables, but brings $\approx$ 24 noise variables into the model in the $n > p$ case and $\approx$ 45 noise variables into the model in the $n < p$ case. The MIQO models and Lasso models perform similarly in terms of predictive power.

Tables 3.5 and 3.6 show results for datasets designed to illustrate pairwise multicollinear-

ity, for the $n > p$ and $n < p$ case, respectively. Again in these cases, the MIQO models and Lasso models perform similarly in terms of predictive power. However, the final Lasso models contain very high pairwise collinearity and condition numbers that indicate severe multicollinearity issues. On the other hand, the MIQO algorithm returns models that generally have half or less of the maximum pairwise collinearity as the corresponding Lasso model, and the condition numbers do not show evidence of severe multicollinearity.

Tables 3.7 and 3.8 show results for datasets designed to illustrate robustness, for the $n > p$ and $n < p$ case, respectively. As described in Section 3.2.3, Lasso is designed to be robust to error in data. Indeed, in both the $n > p$ and $n < p$ case, Lasso and the MIQO algorithm achieve similar predictive power. The maximum pairwise collinearity and condition numbers of the MIQO-based models are lower.

Table 3.3: Sparsity; n = 500, p = 100, $\rho = 0$, $\mathbf{\Delta X = 0}$.

	MIQO							Lasso				
SNR	**Γ***	**K***	**TP**	**R^2**	**MC**	**Con**	**Time**	**K***	**TP**	**R^2**	**MC**	**Con**
6.32	0.014	10.6	10	0.716	0.119	1.61	0.448	34.8	10	0.701	0.148	2.782
	0.010	0.358	0	0.007	0.007	0.02	0.011	3.51	0	0.007	0.010	0.127
3.16	0.011	10.6	10	0.909	0.119	1.27	0.439	34.4	10	0.904	0.148	2.805
	0.010	0.358	0	0.003	0.007	0.29	0.011	3.72	0	0.002	0.010	0.146
1.58	0.011	10	10	0.975	0.117	1.58	0.304	34.6	10	0.974	0.160	2.797
	0.009	0	0	0.001	0.007	0.04	0.011	4.40	0	0.001	0.016	0.194

Table 3.4: Sparsity; n = 100, p = 500, $\rho = 0$, $\mathbf{\Delta X = 0}$.

	MIQO							Lasso				
SNR	**Γ***	**K***	**TP**	**R^2**	**MC**	**Con**	**Time**	**K***	**TP**	**R^2**	**MC**	**Con**
10.54	0.107	10.2	10	0.991	0.249	2.664	1.00	55	10	0.982	0.323	139
0	0.028	0.179	0	0.000	0.026	0.195	0.08	7.33	0	0.001	0.006	97
6.32	0.041	10	10	0.976	0.231	2.793	1.18	56	10	0.952	0.323	1472
0	0.023	0	0	0.001	0.018	0.217	0.00	8.78	0	0.003	0.006	1290
3.16	0.076	11	10	0.896	0.216	3.348	1.64	58.2	10	0.813	0.343	5215
0	0.027	0.283	0	0.006	0.012	0.192	0.42	9.10	0	0.012	0.014	4634

We present results on real data in Table 3.9. All optimization problems were solved to optimality except for the diabetes dataset and HIV dataset where a time limit of 20 seconds per optimization problem solved was enforced. Again, for each dataset, $K_{max} \times$ (# of values of Γ tested)$\times$(# of iterations of Stage 3) MIQO problems are solved. Note that

Table 3.5: Pairwise Multicollinearity; n = 500, p = 100, True K = 10, $\rho = 0.9$, $\mathbf{\Delta X = 0}$.

SNR	MIQO Γ^*	K^*	TP	R^2	MC	Con	Time	Lasso K^*	TP	R^2	MC	Con
8.73	0.02	10.00	10.00	0.99	0.40	4.15	0.30	34.40	10.00	0.99	0.91	126.28
	0.01	0.00	0.00	0.00	0.01	0.17	0.02	2.65	0.00	0.00	0.00	13.15
4.37	0.02	10.40	10.00	0.95	0.47	5.65	0.34	37.20	10.00	0.94	0.91	146.36
	0.02	0.36	0.00	0.00	0.07	1.25	0.04	3.66	0.00	0.00	0.00	20.83
2.18	0.03	11.40	9.60	0.81	0.63	7.92	0.63	36.60	10.00	0.81	0.91	142.17
	0.02	0.54	0.22	0.01	0.08	2.25	0.15	3.37	0.00	0.01	0.00	18.89

Table 3.6: Pairwise Multicollinearity; n = 100, p = 500, True K = 10, $\rho = 0.8$, $\mathbf{\Delta X = 0}$.

SNR	MIQO Γ^*	K^*	TP	R^2	MC	Con	Time	Lasso K^*	TP	R^2	MC	Con
10.5	0.090	10.4	10	0.990	0.331	5.58	1.99	56.2	10	0.979	0.850	119.8
0	0.029	0.358	0	0.001	0.089	1.48	0.517	2.92	0	0.004	0.005	8.2
6.32	0.049	10.4	10	0.976	0.436	6.11	2.12	57	10	0.941	0.846	122.2
0	0.020	0.219	0	0.003	0.118	1.24	0.395	2.65	0	0.010	0.005	7.47
3.16	0.037	12.4	8.8	0.835	0.433	4.38	2.11	61.2	9.8	0.768	0.846	245.4
0	0.011	0.219	0.72	0.041	0.099	0.51	0.514	3.70	0.18	0.029	0.005	117.9

Table 3.7: Robustness: n = 500, p = 100, True K = 10, $\rho = 0$, $\mathbf{\Delta X} \sim$ Uniform(0,2).

SNR	MIQO Γ^*	K^*	TP	R^2	MC	Con	Time	Lasso K^*	TP	R^2	MC	Con
6.32	0.011	10	10	0.975	0.117	1.58	0.448	34.6	10	0.974	0.160	2.797
	0.009	0.000	0	0.001	0.007	0.04	0.011	4.40	0	0.001	0.016	0.194
3.16	0.011	10.6	10	0.909	0.119	1.27	0.439	34.4	10	0.904	0.148	2.805
	0.010	0.358	0	0.003	0.007	0.29	0.011	3.72	0	0.002	0.010	0.146
1.58	0.014	10.6	10	0.716	0.119	1.61	0.304	34.8	10	0.701	0.148	2.782
	0.010	0.358	0	0.007	0.007	0.02	0.011	3.51	0	0.007	0.010	0.127

Table 3.8: Robustness: n = 100, p = 500, True K = 10, $\rho = 0$, $\mathbf{\Delta X} \sim$ Uniform(0,1).

SNR	MIQO Γ^*	K^*	TP	R^2	MC	Con	Time	Lasso K^*	TP	R^2	MC	Con
10.5	0.065	10.6	9.6	0.880	0.282	2.777	1.173	53.8	10	0.856	0.376	53.8
0	0.036	0.607	0.358	0.034	0.021	0.131	0.000	4.66	0	0.013	0.018	22.5
6.32	0.044	10	9.4	0.828	0.246	2.716	1.643	53.4	10	0.829	0.357	107.3
0	0.025	0.632	0.358	0.033	0.017	0.268	0.419	7.69	0	0.029	0.014	75.3
3.16	0.038	11	9.4	0.769	0.262	2.475	1.876	61.8	10	0.705	0.338	763.0
0	0.025	0.85	0.358	0.030	0.027	0.585	0.420	10.27	0	0.035	0.010	546.8

n here indicates the size of the training dataset – the original dataset has $2n$ observations.

Our algorithm achieves similar predictive performance to Lasso, but is significantly more interpretable, choosing fewer variables in general and successfully limiting the degree of

Table 3.9: Results for Basic Structure Real Datasets.

Dataset	n	p	MIQO K*	R^2	MaxCor	Lasso K*	R^2	MaxCor
CPU	105	6	5	0.869	0.716	6	0.861	0.716
Yacht	154	6	1	0.600	NA*	1	0.602	NA*
White Quality	2499	11	10	0.270	0.619	9	0.280	0.828
Red Quality	800	11	6	0.384	0.40	7	0.386	0.69
Compact	4096	21	15	0.717	0.733	21	0.725	0.942
Elevator	8280	18	10	0.808	0.678	15	0.809	0.999
Pyrimidines	37	26	15	0.175	0.781	20	0.367	0.928
LPGA 2008	78	6	2	0.877	0.02	3	0.873	0.234
LPGA 2009	73	11	7	0.814	0.784	10	0.807	0.943
Airline Costs	15	9	2	0.672	0.501	9	0.390	0.973
Diabetes	221	64	4	0.334	0.423	14	0.381	0.672
HIV	528	98	11	0.945	0.662	39	0.944	0.760

*Note that both the MIQO and Lasso algorithms choose only one independent variable for the Yacht Hydrodynamics dataset, hence there is no maximum pairwise correlation in this case.

Table 3.10: The Price of Limiting Multicollinearity

Dataset	n	p	MIQO K*	R^2	MaxCor	Lasso K*	R^2	MaxCor
Pyrimidines	37	26	18	0.375	0.870	20	0.367	0.928

multicollinearity present in the final model.

We notice that the Pyrimidines dataset has significantly lower predictive power than Lasso. This is the price of insisting on interpretability, despite a relatively low ratio of observations to variables. We demonstrate the algorithm's performance on this dataset when the maximum correlation threshold is set to 1 (i.e., no limit) and record the performance in Table 3.10.

In these cases it is up to the analyst to judge which model is preferable; one with better predictive performance or one with coefficients that are more interpretable. The benefit of using our algorithm is that it is simple for an analyst to tweak the parameters and quickly understand the tradeoffs.

3.4.2 Special Structure

Nonlinear Transformations

We investigate our algorithm's capability to identify when a nonlinear transformation of an independent variable may be useful. For this task, we used the Concrete Compressive Strength dataset from [114] available in the UCI Machine Learning Repository [2]. The dataset contains 8 independent variables and 1030 observations. As before, we randomly split the dataset into a training set (50%), validation set (25%), and test set (25%).

Table 3.11: Independent Variables in the Concrete Compressive Strength Dataset.

Variable	**Units**
Concrete Compressive Strength	MPa
Cement	kg/m^3
Blast Furnace Slag	kg/m^3
Fly Ash	kg/m^3
Water	kg/m^3
Superplasticizer	kg/m^3
Coarse Aggregate	kg/m^3
Fine Aggregate	kg/m^3
Age	day

The independent variable is the compressive strength of concrete, and the dependent variables are the ingredients as well as the age of the concrete (see Table 3.11 for details). The practical goal in civil engineering is to design a concrete mixture which will have high compressive strength. However, concrete compressive strength is known to be a highly nonlinear function of its age and ingredients.

On the original data, both our algorithm and Lasso chose to use all covariates and produced a test set R^2 of 0.609. We then reran our algorithm with an extended dataset, which contained each of the original columns x as well as three transformed versions of each column: $x^2, \sqrt{x}$, and $\log(x)$. For the variables which take zero values (blast furnace slag, fly ash, and superplasticizer), we adjusted the log transformation to be $\log(x + 0.00001)$. We included Constraint (4.8g) in the optimization model to ensure that for each column x, at most one of x, x^2, $\sqrt{x}$, and $\log(x)$ appeared in the final model.

As expected, the inclusion of transformed covariates significantly improved upon the models created with just the original variables. The MIQO algorithm selected six covariates to appear in the top model. The six covariates chosen were blast furnace slag, water, log(fly ash), log(super plasticizer), log(day), and cement$^{1/2}$. Each covariate was significant at the $\alpha = 0.001$ level and test set R^2 was 0.823, a significant improvement over the original test set R^2 of 0.609.

We also tested Lasso on the dataset that included the nonlinear transformations. Lasso selected a model with twelve covariates which resulted in a test set R^2 of 0.834. In addition to the first five covariates chosen by our algorithm, Lasso also selected cement^2, super plasticizer^2, day^2, log(cement), log(coarse aggregate), log(fine aggregate), and $\sqrt{\text{cement}}$. In our opinion, the minor increase in test set R^2 does not warrant using a significantly less interpretable model.

Although the number of variables went from 8 to 32 when we included nonlinear transformations, the MIQO algorithm took the same amount of time (roughly 1-1.5 minutes) to execute in both cases. By imposing limiting constraints on transformations and pairwise correlation, the feasible space is not significantly enlarged by including nonlinear transformations.

Group Sparsity

We demonstrate our results in a group sparsity setting using the Energy Efficiency dataset from [107] available on the UCI Machine Learning Repository ([2]). The dataset has 768 observations of six continuous independent variables and two categorical independent variables. The independent variables describe building properties (see Table 3.12 for details). There are two dependent variables available: heating load and cooling load. We test our method on both dependent variables.

The binary expansion of the categorical variable orientation into three new binary variables and of glazing area distribution into five new binary variables meant that the dataset passed to the algorithm contained fourteen independent variables. When we ran the algo-

Table 3.12: Independent Variables in the Energy Efficiency Dataset.

Variable	**Type**
Relative Compactness	Continuous
Surface Area	Continuous
Wall Area	Continuous
Roof Area	Continuous
Overall Height	Continuous
Orientation	Categorical; 4 levels
Glazing Area	Continuous
Glazing Area Distribution	Categorical; 6 levels

rithm, the best model contained three variables: wall area, overall height, and glazing area. We found identical results when predicting cooling load. Our algorithm chose not to use either of the categorical variables provided. The top heating load model had test set R^2 of ≈ 0.88 and the top cooling load model had test set R^2 values of ≈ 0.85.

In [107], the original study of this dataset, the authors found that wall area, roof area, and relative compactness were the variables that appear mostly associated with heating load and cooling load, although all variables appear in their model. Using the Random Forest method, they also found importance scores for each variable and found that glazing area was the most important variable. They note that "Interestingly, the most important variable (glazing area) is not the most correlated with either output variable. From an engineering perspective, it can be intuitively understood that the glazing area is of paramount significance... "

We find it notable that our algorithm did not identify the same three variables as most critical for predicting the responses, and could not have: due to correlation of -0.86 between roof area and relative compactness, these two variables could not have both been in our algorithm's final model. However, glazing area, which the authors point out as having paramount significance, is in all three of our top models for both response variables.

We also tested group lasso. The group lasso models for predicting the two response variables each chose to use thirteen of the fourteen variables, including both categorical variables. The only variable excluded was surface area. The heating load model had a test set R^2 of ≈ 0.91 and the cooling load model had a test set R^2 value of ≈ 0.86.

3.4.3 Combined Example

In the previous sections we have demonstrated how our algorithm can handle a wide variety of individual situations: detecting sparsity, limiting pairwise correlation, identifying nonlinear transformations, and others. In this section, we will show the full force of our algorithm: to identify all these properties when presented together. Specifically, we consider an example whose structure incorporates general sparsity, selective sparsity in terms of both high pairwise multicollinearity and group sparsity, and modeler expertise in a single dataset. We test this example on the high-dimensional case where $n = 100$ and $p = 1000$.

We generated a synthetic data matrix $\mathbf{X}$ for $n = 100, p = 500$ according to the process outlined in Section 3.4.1, using a value of $\rho = 0.8$ to ensure that there is high pairwise multicollinearity present between some columns of $\mathbf{X}$. To generate nonlinear transformations, for each column j of $\mathbf{X}$ we included an additional column consisting of the squared entries of j, bringing the total number of potential covariates up to 1000. As before, we consider $k = 10$. However, we generated $\beta_i = 1$ so that 7 positive values occurred in the original 500 columns and 3 were located in the 500 transformed columns. The response $\mathbf{y}$ was generated as before as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. To test our robustness to error in data, we generated a matrix $\Delta\mathbf{X} \sim \text{Unif}(0,f)$ and considered $\mathbf{X} + \Delta\mathbf{X}$ for various values of f. We assume the modeler has some expertise with this sort of data, and knows one of the values of i such that β_i is truly nonzero. Finally, the modeler is also aware of a group sparsity structure and knows that $\beta_a, \beta_b, \beta_c$, and β_d are all either all zero or all nonzero and that $\beta_e, \beta_f, \beta_g$, and β_h are either all zero or all nonzero, where $\{a, b, c, d\} \in \{i|\beta_i = 1\}$ and $\{e, f, g, h\} \in \{i|\beta_i = 0\}$.

Table 4.7 presents results for this combined example. As before, the top row presents average results over five trials of the same experiment and the bottom row presents the standard error.

Table 3.13: Results for Combined Example

ϵ	ΔX	MIO Γ^*	K^*	TP	R^2	MC	Con	Time	Lasso K^*	TP	R^2	MC	Con
0.5	0	0.026	10.4	10	0.981	0.437	4.654	1.14	46.6	10	0.969	0.836	118.1
		0.020	0.219	0	0.001	0.020	0.382	0.17	4.15	0	0.004	0.007	17.4
0.5	1	0.000	11.2	10	0.913	0.556	6.995	1.34	65.8	10	0.854	0.798	424.0
		0.000	0.522	0	0.013	0.073	1.422	0.22	6.97	0	0.017	0.006	177.5
0.5	2	0.030	11.0	9	0.742	0.501	5.291	1.88	69	9.2	0.598	0.708	8147
		0.027	0.490	0.28	0.030	0.061	0.508	0.42	8.54	0.18	0.045	0.006	6994
1	0	0.026	11.2	10	0.931	0.468	5.322	1.08	45.6	10	0.878	0.836	113.9
		0.022	0.522	0	0.007	0.018	0.531	0.04	3.99	0	0.016	0.007	18.2
1	1	0.041	10.4	10	0.878	0.478	4.998	1.61	69.2	10	0.759	0.796	362.8
		0.036	0.219	0	0.016	0.059	0.696	0.43	4.92	0	0.033	0.006	96.4
1	2	0.099	9.8	7.6	0.573	0.436	4.224	1.64	72.4	8.6	0.503	0.702	573.8
		0.041	0.867	0.22	0.042	0.061	0.576	0.42	5.89	0.36	0.064	0.006	228.0
2	0	0.090	10	8.8	0.720	0.451	4.687	1.35	39.4	8.6	0.599	0.836	74.0
		0.045	0.283	0.34	0.046	0.025	0.289	0.23	4.01	0.46	0.055	0.007	9.76
2	1	0.116	9.4	8.2	0.614	0.426	4.262	2.05	53.4	7.8	0.509	0.782	113.5
		0.037	0.358	0.59	0.078	0.025	0.137	0.39	4.41	0.96	0.067	0.010	20.7
2	2	0.032	8.2	4	0.245	0.403	3.506	1.46	55.8	5.8	0.368	0.680	8141
		0.017	0.657	0.98	0.129	0.074	0.720	0.22	10.1	0.52	0.061	0.011	7236

3.5 Conclusions

We have leveraged the power of MIQO and proposed an approach for incorporating a variety of desired properties into a linear regression model. Our approach provides the only methodology we are aware of to construct models that impose statistical properties simultaneously. This results in a generally applicable, unified framework for addressing all aspects of the model-building process. Using both real and synthetic data, we demonstrate that the approach produces high-quality linear regression models in realistic timelines.

Chapter 4

Logistic Regression: Subset Selection and An Algorithmic Approach

4.1 Introduction

Logistic regression is a common classification method for fitting models where the response variable is binary. In a logistic regression model, the log-odds of an observation are assumed to be a linear function of the independent variables. That is, for the response vector $\mathbf{y}_{n\times 1}$, model matrix $\mathbf{X} = [\mathbf{x}_1', \ldots, \mathbf{x}_n'] \in \mathbb{R}^{n\times p}$, and regression coefficients $\boldsymbol{\beta} \in \mathbb{R}^{p\times 1}$, we assume the following model:

$$\log\left(\frac{P(y_i = 1|\mathbf{x}_i)}{P(y_i = 0|\mathbf{x}_i)}\right) = \boldsymbol{\beta}'\mathbf{x}_i. \tag{4.1}$$

Logistic regression minimizes the negative log-likelihood of the data given the model. The negative log-likelihood in logistic regression is given by

$$f(\boldsymbol{\beta}) = \sum_{i=1}^{n} -y_i(\boldsymbol{\beta}'\mathbf{x}_i) + \log\left(1 + \exp(\boldsymbol{\beta}'\mathbf{x}_i)\right),$$

and so the regression coefficients $\boldsymbol{\beta}$ can be found by solving the following convex optimization problem:

$$\min_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) \tag{4.2}$$

As in linear regression, it is often preferable to obtain a parsimonious fit of the data. Thus, we would like to solve the analogous best subset problem for logistic regression:

$$\min_{\boldsymbol{\beta}} \quad f(\boldsymbol{\beta}) \text{ subject to } \|\boldsymbol{\beta}\|_0 \leq k \tag{4.3}$$

where, as before, the ℓ_0 (pseudo)norm of a vector $\boldsymbol{\beta}$ counts the number of nonzeros in $\boldsymbol{\beta}$ and is given by $\|\boldsymbol{\beta}\|_0 = \sum_{i=1}^{p} 1(\beta_i \neq 0)$.

The objective function in Problem 4.3 is convex, and the cardinality constraint can be modeled, as in Chapter 2, via MIO. However, MIO solvers for general convex programs are not nearly as developed as MIO for linear or quadratic problems; while software exists, there is high variation in solver performance for different problem instance families [19]. Additionally, the unconstrained logistic regression problem 4.2 cannot generally be solved analytically in closed form. Typically logistic regression is solved via iterative methods, with gradient descent and iteratively reweighed least squares being two popular options.

These differences between linear and logistic regression lead to the main challenge of this part of the thesis: developing a method to efficiently solve the mixed integer convex optimization problem of logistic regression with cardinality constraints to provable optimality. We develop a tailored algorithm to do so where we combine (a) well-known techniques in mixed integer nonlinear optimization with (b) our discrete first order heuristic from Chapter 2 and (c) lazy constraint callbacks, a feature of modern optimization solvers. We demonstrate that our method outperforms existing MINLO software and also extend the algorithmic approach to linear regression to the logistic regression case.

4.1.1 Literature Review

Even the unconstrained logistic regression problem (4.2) cannot generally be solved analytically in closed form, and is typically solved via iterative methods such as gradient descent

and iteratively reweighed least squares. Problem (4.3) adds a combinatorially challenging cardinality constraint. In [52], the authors propose solving Problem (4.3) via an implicit enumeration algorithm when $f(\boldsymbol{\beta})$ is the linear regression objective function. [61] showed that software implementing the algorithm of [52] can be used directly in the case of logistic regression as well. However, the algorithm of [52] does not scale past $p = 30$, leading much of the statistics community to view solving Problem (4.3) as generally intractable, and to turn to convex relaxations of Problem (4.3).

The familiar ℓ_1 penalty approach is not nearly as straightforward to implement in the case of logistic regression as it is for its linear regression counterpart, since this makes the negative log-likelihood function non-differentiable. Thus, much of the literature on subset selection in logistic regression focuses on how to solve an ℓ_1-regularized likelihood problem.

Many algorithms and computational techniques have been suggested in recent years to solve the ℓ_1-regularized likelihood problem. In [51] the authors propose a cyclical coordinate descent algorithm along a regularization path which estimates generalized linear models with convex penalties. In [71], the authors proposed a modified version of iteratively reweighed least squares which cleverly builds the regularization into the iterative process. Bound optimization, which has the flavor of an expectation-maximization algorithm, is suggested in [69]. [68] reformulates the ℓ_1-penalized logistic regression model as an equivalent convex optimization problem, and suggests an approach based on interior point methods.

An alternative to using a regularizer is to use a prior which encourages sparsity. There is a family of sparse classification algorithms which form classifiers as weighted linear combinations of basis functions. To induce sparsity, the likelihood of the weights is frequently regularized by a prior which promotes sparsity. The Laplacian prior is a common choice, as it gives an ℓ_1 penalty. Relevance vector machines [103], sparse probit regression [47], and the joint classifier and feature optimization algorithm [70] all fall within this family of algorithms.

The only work we are aware of which suggests addressing subset selection in logistic regression using modern MIO solvers is [95]. The authors formulate an optimization problem

with an objective that includes a weight on a piecewise linear version of the log likelihood and an information criterion which penalizes the number of parameters included. They show that MIO outperforms traditional stepwise methods with respect to the information criterion, and suggest solving the full MINLO rather than a linear approximation as a direction of future research. In this chapter, we will directly consider the MINLO.

4.2 Mixed Integer Nonlinear Optimization

MINLOs form a challenging class of optimization problems, due to their inclusion of both integer variables and nonlinear functions. For additional background on MINLO see [19]. A subclass of MINLOs which is much more tractable is convex MINLO. This refers to the case where, when integer constraints are relaxed, the resulting problem forms a convex nonlinear program. This convexity leads to particular algorithms designed for convex MINLO. As Problem 4.3 is in the class of convex MINLO, we will restrict our attention henceforth to convex MINLO.

Developing algorithms for solving convex MINLO to provable optimality has been an active area of research since the 1970s, and a wide variety of MINLO solvers have been built based on these algorithms. Such algorithms integrate techniques from nonlinear optimization, integer optimization, and linear optimization. Typically these algorithms rely on two major solution techniques:

1. Branch and bound with nonlinear relaxations.

2. Linear relaxations of h and g_j.

The first technique typically obtains the nonlinear relaxation by relaxing the integrality requirements. The second technique, on the other hand, maintains the integrality requirements but replaces the nonlinear functions h and g_j by linear relaxations. Two major classes of methods that fall into this category are outer approximations (OA) and extended cutting planes (ECP). OA methods obtain linearizations of h and g_j from gradients at NLO subproblem solutions and add these cutting planes to a reduced master MILO problem. ECP

methods do not solve an NLO subproblem but instead generate cutting planes around linearizations of the most violated constraint in an iteratively updated MILO reduced master problem. There are a few other techniques that MINLO solvers use: for example, integrating linearizations into the branch and cut process or alternating between LO and NLO relaxations during branch and bound. However, the majority of solvers fall into the first two categories. See [22] for a complete categorization of twenty-four existing MINLO solvers.

4.2.1 Computational Tests on Existing MINLO solvers

MINLO has benefited substantially from improvements both in MIO and in NLO. We documented advances in MIO in the introduction; in [109], Waltz suggests that problem instances that can be solved by NLO are growing by almost an order of magnitude each decade. Nevertheless, there is still substantial variation among solver performance on different problem classes and instances. [19] documents this phenomenon by comparing many MINLO solvers on the same set of problems.

The NEOS server, operated by the University of Wisconsin, makes many optimization solvers freely available for use on their servers ([31], [35], [56]). By using the NEOS server, we were able to test six different solvers side by side using an AMPL interface and confirm that solver variation is indeed the norm for best subset logistic regression.

We built three test problems in order to compare MINLO solver performance.

In each case, we generated data such that $\mathbf{x}_i \sim \mathrm{N}(\mathbf{0}, \mathbf{\Sigma}), i = 1, \ldots, n$ were independent realizations from a p-dimensional multivariate normal distribution with mean zero and covariance matrix $\mathbf{\Sigma} := (\sigma_{ij})$. The columns of the $\mathbf{X}$ matrix were standardized such that the training set had columns with zero mean and unit ℓ_2-norm. For a fixed $\mathbf{X}_{n \times p}$, we generated the response $\mathbf{y}$ as follows: $\mathbf{y_i} = \mathrm{Round}\left((1/\left(1 + \exp(-\boldsymbol{\beta}'\mathbf{x}_i + \epsilon_i\right))\right)$, where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. We denote the number of nonzeros in $\boldsymbol{\beta}$ by k. In particular, we took $\sigma_{ij} = \rho^{|i-j|}$ for $i, j \in \{1, \ldots, p\} \times \{1, \ldots, p\}$. In our experiments, we consider $k = 5$ and $\beta_i = 1$ for $i \in \{1, \ldots, p\}$ such that $i \bmod p/k = 0$ to generate k equally spaced values.

The exact parameters of each of the three test problems were as follows:

Problem 1: $n = 100, p = 10, k = 5, \rho = 0.4, \sigma = 2$.

Problem 2: $n = 1000, p = 100, k = 5, \rho = 0.4, \sigma = 2$.

Problem 3: $n = 2000, p = 200, k = 5, \rho = 0.4, \sigma = 2$.

We used the AMPL interface to the NEOS server, and tested all six solvers for which an AMPL interface was available: Bonmin, KNITRO, FilMINT, MINLP, SCIP, and Couenne. We did not change the default options on any of the solvers.

The default algorithm in Bonmin 1.6.0 implements an NLO-based branch and bound algorithm using the MIO solver Cbc 2.7.6 and the NLO solver Ipopt 3.10.2. KNITRO does automatic algorithm selection between interior-point and active-set methods. FilMINT combines the MINTO branch-and-cut framework for MIO with filterSQP, an active set solver for solving the NLO subproblems. MINLP implements an NLO-based branch-and-bound algorithm. The NLO problems at each node of the branch and bound tree are solved using filterSQP. SCIP is a constraint integer optimization solver. Couenne implements a reformulation-based branch and bound algorithm. Table 4.1 presents a comparison of times (in seconds) for each solver to reach optimality on each test problem, up to a maximum cut off time of 7200 seconds (2 hours). Note that we did not solve each test problem for every value of k, but only for the value of k corresponding to the true value of k. Thus the times presented are the solve times for a single instance of the problem, averaged over five runs. To truly solve the best subset problem, we would need to solve each test problem for every value of k. We do not present the time results with the goal of accurately benchmarking the best time possible, but rather to give a sense of generally expected solve times under the standard conditions of operating on a shared server.

The solve times presented in Table 4.1 represent the time to build and solve the problem. Thus, if we can embed the solver within a optimization language such that the problem does not have to rebuilt for successive values of k, we can expect that subsequent re-solves for additional values of k would be much faster. With this in mind, the solve time of the first four solvers on the NEOS server for Problem 1 may be efficient enough for practical purposes, especially since $p = 10$ in Problem 1. However, the increased complexity of Problems 2 and

Table 4.1: MINLO Solver Comparison Times (in seconds).

Solver	Problem 1	Problem 2	Problem 3
Bonmin	11	168	2370
KNITRO	16	29	145
FilMINT	10	1283	633
MINLP	11	$\approx 300^*$	$\approx 6000^*$
SCIP	cut off	cut off	cut off
Couenne	cut off	cut off	cut off

*Note that the MINLP solver did not provide timestamps for solve times beyond 1 minute so these are rounded times based on computer clock time.

3 indicated the variability between solvers – and the inability to scale effectively to problems of a typical size. This provided the motivation to create our own tailored algorithm to solve the best subset logistic regression problem to optimality efficiently.

4.3 Tailored Algorithm

Our tailored algorithm for efficiently solving the mixed integer convex optimization problem of logistic regression with cardinality constraints to provable optimality consists of three main ingredients:

1. Our discrete first order heuristic from Chapter 2.

2. Outer approximation methods, a well-known technique in MINLO.

3. Lazy constraint callbacks, a feature of modern optimization solvers.

4.3.1 Discrete First Order Heuristic

The discrete first order heuristic we developed in Chapter 2 significantly aided the speed in finding MIO solutions when used to warm-start the best subset problem in linear regression. This heuristic applies to any problem of minimizing a convex function with Lipschitz-continuous gradient subject to cardinality constraints.

Proposition 6. *The logistic regression objective function $f(\boldsymbol{\beta})$ of Problem 4.2 has Lipschitz constant ℓ given by $\ell = \frac{1}{4}\lambda_{\max}(X'X)$.*

Proof. Since f is twice differentiable, it will have Lipschitz-continuous gradient if there exists a value of ℓ such that $\ell I \succeq \nabla^2 f$, that is, if there is a value of ℓ which uniformly upper bounds the largest eigenvalue of the Hessian. In this case, $\nabla^2 f = X'WX$ where W is a diagonal matrix with $w_{ii} = P(y_i = 1|\mathbf{x}_i) \cdot P(y_i = 0|\mathbf{x}_i)$. Then since $P(y_i = 1|\mathbf{x}_i) \cdot P(y_i = 0|\mathbf{x}_i) \leq \frac{1}{4}$, $\lambda_{\max}(X'WX) \leq \frac{1}{4}\lambda_{\max}(X'X)$. □

By Proposition 6, the heuristic is applicable in the logistic regression setting, and we can use this heuristic as part of solving the MINLO logistic regression problem.

4.3.2 Outer approximation methods

The outer approximation algorithm for convex MINLO was introduced by Duran and Grossmann in 1986 ([42]). The algorithm alternates between solving a mixed integer linear optimization problem and a pure nonlinear optimization problem, where linearizations of the objective function around solutions to the NLO are added to the MILO. These linearizations are obtained by the convexity and differentiability of f: for any value of $\hat{\boldsymbol{\beta}} \in \mathbb{R}^p$, the following linear inequality is valid: $f(\boldsymbol{\beta}) \geq f(\hat{\boldsymbol{\beta}}) + \nabla f(\hat{\boldsymbol{\beta}})'(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})$.

In the case of best subset logistic regression, the algorithm proceeds as follows. First, Problem 4.2 is solved and has optimal solution $\boldsymbol{\beta}^{NLO}$. The following mixed integer optimization problem, which we call the reduced master problem (RMP), is formed:

$$
\begin{aligned}
\min_{\boldsymbol{\beta}} \quad & \eta \\
s.t. \quad & \eta \geq f(\boldsymbol{\beta}^{NLO}) + \nabla f(\boldsymbol{\beta}^{NLO})'(\boldsymbol{\beta} - \boldsymbol{\beta}^{NLP}) \\
& -M\boldsymbol{z} \leq \boldsymbol{\beta} \leq M\boldsymbol{z} \\
& \sum_{i=1}^{p} z_i \leq k \\
& z_i \in \{0,1\}, \quad i = 1, \ldots, p.
\end{aligned}
\tag{4.4}
$$

The reduced master problem 4.4 is solved to optimality. The support of the resulting solution, $\boldsymbol{\beta}^{RMP}$, is then fixed, and the following nonlinear optimization problem is solved:

$$
\begin{aligned}
\min_{\boldsymbol{\beta}} \quad & f(\boldsymbol{\beta}) \\
s.t. \quad & \text{support}(\boldsymbol{\beta}) = \text{support}(\boldsymbol{\beta}^{RMP})
\end{aligned}
\tag{4.5}
$$

The solution to Problem 4.5 is a new $\boldsymbol{\beta}^{NLO}$. Linearizations around this new $\boldsymbol{\beta}^{NLO}$ are added to the reduced master problem 4.4, and the algorithm continues to alternate between problems 4.4 and 4.5. At each stage, these cutting plane linearizations cut off the current integer solution to Problem 4.4 unless the integer solution is optimal for Problem 4.3. As the algorithm progresses, the reduced master problem 4.4 becomes an increasingly closer approximation to Problem 4.3. The global minimum of Problem 4.3 is reached when the objective function of the reduced master problem 4.4 is within some pre-specified tolerance ϵ of the objective function of the NLO problem 4.5.

We incorporate our discrete first order heuristic at the beginning of this procedure by adding a linearization around the solution to Problem 4.5 with $\boldsymbol{\beta}^{RMP}$ taken as the first order heuristic solution. By doing this in the very first step, we ensure that a high-quality cutting plane is added immediately to Problem 4.4, causing the outer approximation algorithm to converge much more quickly. Additionally, we implement an efficient way to solve the reduced master problem 4.4 that only requires building one branch and bound tree using

lazy constraint callbacks.

4.3.3 Lazy Constraint Callbacks

Callbacks are a general programming concept and not specific to optimization: in general, a callback is a piece of executable code which is passed as a parameter to other code, to be invoked at some pre-specified time. Optimization solvers which allow callbacks give the user the option to monitor and modify the behavior of the solver. Callbacks can be used to access information or provide a status update during the course of the optimization; they can be used to terminate optimization if a particular condition is reached; they can also be used to update bounds on constraints and variables and add user-generated solutions and cutting planes. In our case, we will use *lazy constraint callbacks* which dynamically (or lazily) add cutting planes to the model whenever an integer feasible solution is found. Unless the current integer solution is optimal, this will refine the feasible region of the problem by cutting off the current integer solution.

Lazy constraint callbacks are a relatively new type of callback. CPLEX 12.3 introduced lazy constraint callbacks in 2010 and Gurobi 5.0 introduced lazy constraints in 2012. To date, the only MIO solvers which provide lazy constraint callback functionality are CPLEX([99]), Gurobi ([63]), and GLPK [53]. Lazy constraints are particularly helpful in a few situations: (1) when the pool of constraints is known, but is so large that including all of them explicitly would significantly slow down solver progress; (2) when the pool of constraints is too large to be generated a priori, as in the traveling salesman problem, or (3) when the required constraints are not known at the outset. Cutting plane linearizations generated by the outer approximation method fall into the third category.

It is important to note that the outer approximation method for solving convex MINLO does not require lazy constraint callbacks, but if we do exploit the functionality of lazy constraint callbacks, only one branch and bound tree needs to be built. This saves the rework of rebuilding a new branch and bound tree every time a new integer feasible solution is found in the MIO problem.

Lazy constraints are a fairly new feature within optimization solvers. Although many problems within statistics are naturally formulated as MIO or MINLO problems, to the best of our knowledge, we are the first to integrate the optimization-based concept of lazy constraints into the process of building a statistical model.

4.4 Computational Results – Best Subset

The computational tests in Table 4.2 were performed on a computer with an Intel Xeon E5440 (2.8 GHz) processor with 8 cores and 32 GB of RAM in order to fairly compare times with the NEOS server. All other computational tests were performed on a computer with an Intel Xeon E5687W (3.1 GHz) processor, 16 cores, and 128 GB of RAM. We used `Gurobi 6.0.0` ([63]) as the optimization solver, and implemented the algorithm in `Julia 0.3.3` ([12]), a technical computing language. We used `JuMP 0.7.0` ([74]), an algebraic modeling language package for Julia, to interface with Gurobi.

We begin by comparing our algorithm's performance to the same set of three test problems from Section 4.2.1, again averaging times over five trials:

Table 4.2: MINLO Solver Comparison Times (in seconds).

Solver	Problem 1	Problem 2	Problem 3
Bonmin	11	168	2370
KNITRO	16	29	145
FilMINT	10	1283	633
MINLP	11	≈ 300	≈ 6000
SCIP	cut off	cut off	cut off
Couenne	cut off	cut off	cut off
OUR ALGORITHM	**<1**	**15**	**16**

In these trials, our tailored algorithm was uniformly faster than the six optimization solvers we tested on the NEOS server. Moreover, this speed comparison indicates that our algorithm can scale to higher dimensional problems more easily than other existing MINLO software. Next, we compare the performance of solving the best subset logistic regression using MINLO compared to heuristic methods.

4.4.1 Methodology Comparison

As indicated in Section 4.1.1, logistic regression with an ℓ_1-penalty is the primary method for inducing sparsity in logistic regression models. In this section we compare our methodology with Lasso for logistic regression and report sparsity and predictive performance. We measure predictive performance using area under the ROC curve (AUC) as our metric.

Overdetermined Regime We begin by considering the traditional overdetermined regime with $n > p$. Figure 4-1 shows a representative case within the overdetermined regime with $n = 2000$ and $p = 200$. We note that the MINLO approach and the Lasso approach perform almost identically with respect to AUC across many different noise (σ) and correlation (ρ) levels. Where we notice a large difference between the two methods is in the number of nonzero coefficients chosen by the two methods. MINLO significantly outperforms Lasso in this respect. In this example, there are five true nonzero coefficients. MINLO never selects more than seven. Lasso selects far more variables to enter the model, and is less consistent than MINLO: we observe far greater standard error over the ten trials.

High Dimensional Regime Our method is applicable both in the traditional overdetermined $n > p$ regime and in the increasingly common high dimensional underdetermined $n < p$ regime. The tailored approach of using mixed integer optimization in conjunction with warm starts and lazy constraint cutting planes generated by pure nonlinear optimization rapidly finds the optimal solution.

However, in the $n < p$ regime, we observe that the lower bounds of the mixed integer optimization problem progress slowly, so while the optimal solution may have been found, certification of optimality happens slowly, if at all.

To address this, we consider adding bounding box constraints to the MINLO formulation, as in Chapter 2. These constraints limit the search space, and allow the solver to certify optimality within the bounding box. In particular, using the notation from Chapter 2, we consider the following additional bounding box constraints to the reduced master problem

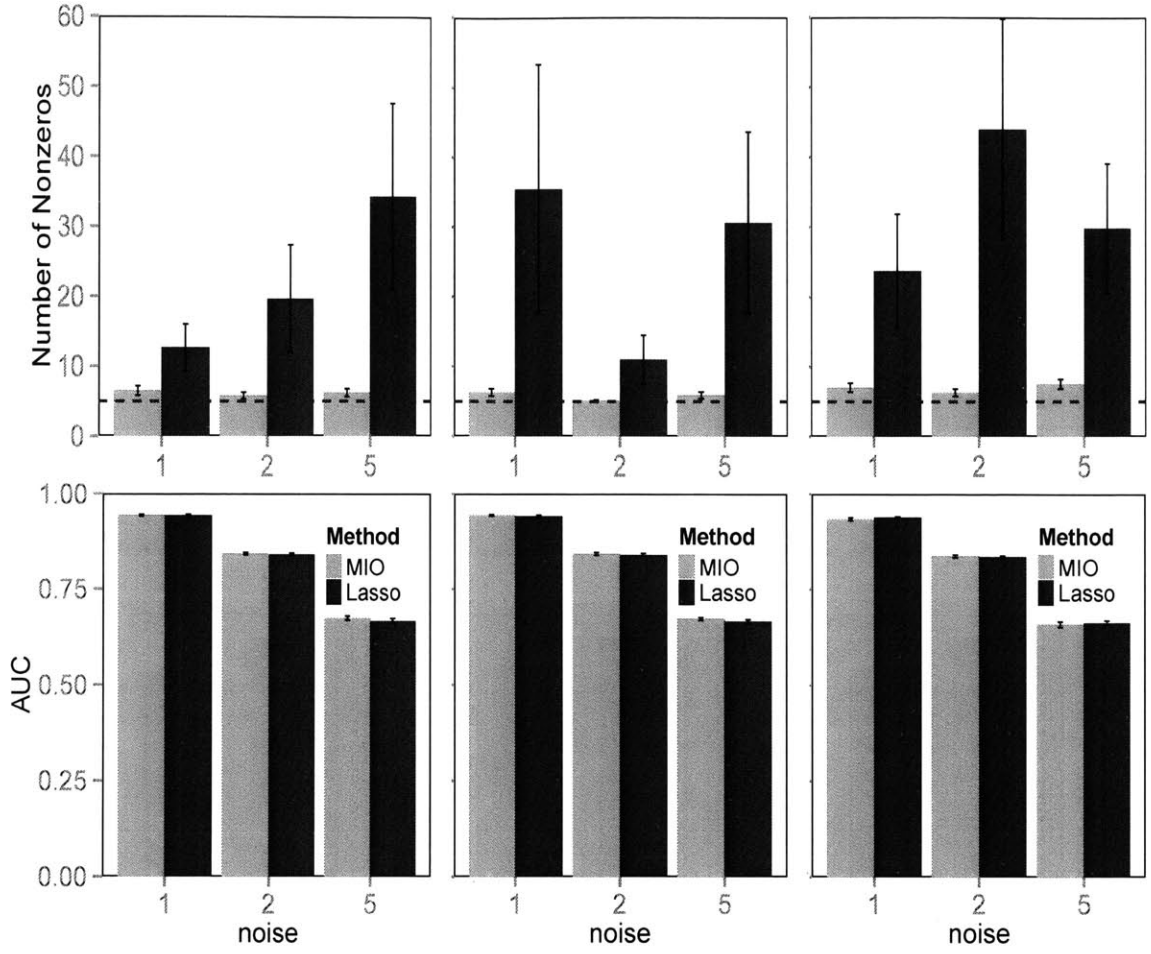

Figure 4-1: Series of computational tests for Problem 2 with $n = 2000, p = 200$. Figure shows number of nonzero values and predictive performance for different values of ρ. The left panel is $\rho = 0$, the middle panel is $\rho = 0.4$, and the right panel is $\rho = 0.8$. The dashed line in the top panel represents the true number of nonzero values.

(4.4):

$$\boldsymbol{\beta} : \|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_1 \leq \mathcal{L}^{\beta}_{\ell,\text{loc}},$$

where $\boldsymbol{\beta}_0$ is a candidate sparse solution. The radius of the ℓ_1-ball above, that is, $\mathcal{L}^{\beta}_{\ell,\text{loc}}$, is a user-defined parameter which controls the size of the feasible set.

In our experiments, we ran our tailored algorithm for 180 seconds, and used the resulting solution as $\boldsymbol{\beta}_0$. We then generated the box constraint using $\mathcal{L}^{\beta}_{\ell,\text{loc}} = \|\boldsymbol{\beta}_0\|_1/k$.

Figure 4-2 gives sparsity and predictive performance results for an example in the high dimensional regime with $n = 400$ and $p = 1000$.

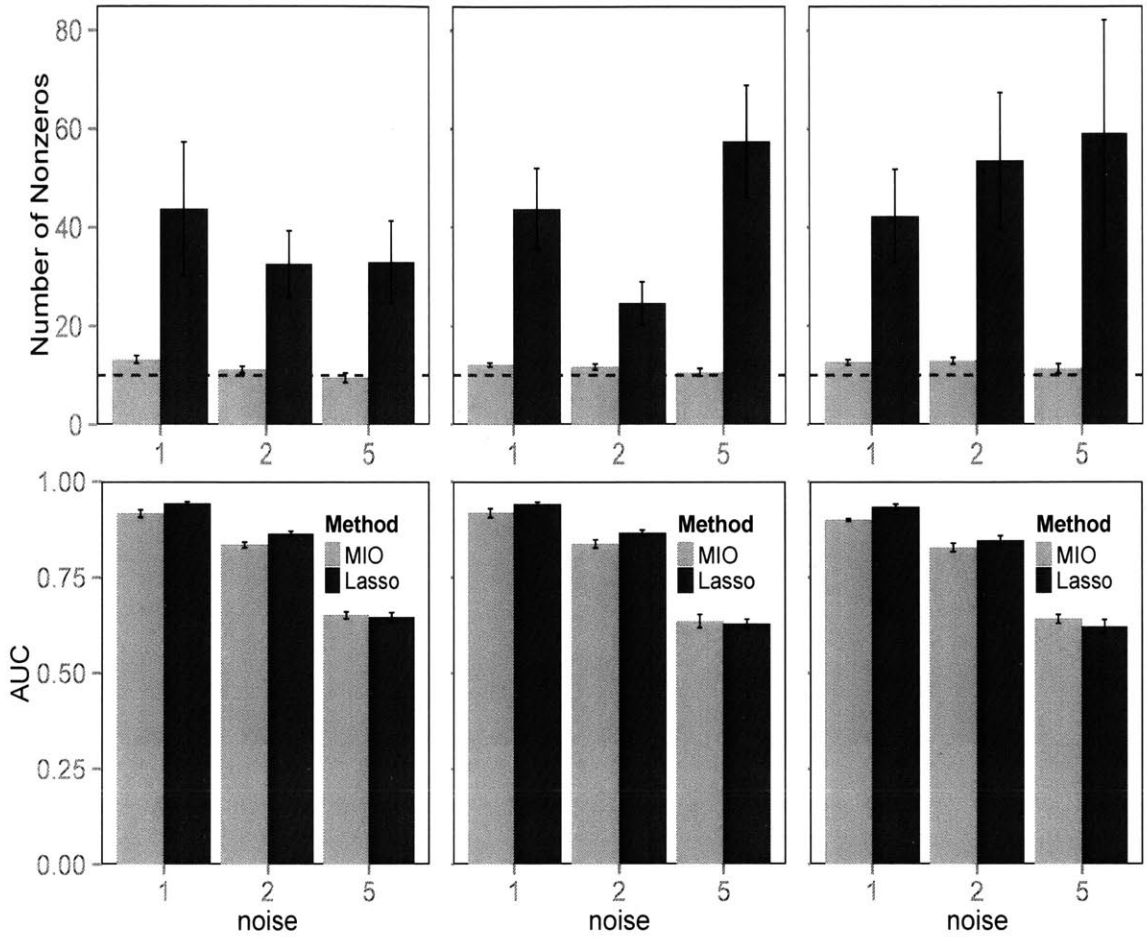

Figure 4-2: Series of computational tests for Problem 4 with $n = 400, p = 1000$. Figure shows number of nonzero values and predictive performance for different values of ρ. The left panel is $\rho = 0$, the middle panel is $\rho = 0.4$, and the right panel is $\rho = 0.8$. The dashed line in the top panel represents the true number of nonzero values.

We notice that in this example, Lasso frequently, but not always, has slightly better predictive performance than MINLO. Nevertheless, the number of nonzero coefficients chosen by Lasso are far higher than the number selected by MINLO. MINLO consistently chooses a number of nonzeros in the neighborhood of the true number; the Lasso solution exhibits much higher standard error of the mean, and is usually 2-4 times the true number of non

zeros.

These observations about predictive performance and sparsity in the high dimensional regime are commensurate with our observations in Chapter 3 that Lasso is a *robust* method first and foremost, and a sparsity-inducing method second. Even so, we doubt that the minor increase in predictive performance that Lasso's robustness may induce is worth the tradeoff of introducing so many variables into the model.

4.5 Algorithmic Approach to Logistic Regression

As with linear regression, logistic regression models are customarily constructed through an iterative process of trial and error in order to balance several competing objectives. In addition to sparsity, we may want to build other properties into a high quality classification model. The framework that we have built for the best subset selection problem in logistic regression is directly adaptable to the case where we have many other goals for our model that can be modeled by mixed integer optimization.

We briefly review desirable characteristics of a logistic regression model. As these closely mirror desirable properties of a linear regression model, we do not go into great detail about each property. Rather, we compare our MINLO approach to achieving these properties in logistic regression models with existing approaches in the literature.

4.5.1 Selective Sparsity

As in Section 3.2.2, we use the term "selective sparsity" to refer to settings where we would like to constrain the joint inclusion of subsets of independent variables. The settings where selective sparsity may be desirable that we will consider here are group sparsity, pairwise multicollinearity, and nonlinear transformations.

Group Sparsity

Some applications exhibit a block- or group-sparse structure, with groups of independent variables whose coefficients are either all zero or all nonzero. Group Lasso, first proposed for linear regression in [115], has analogously been proposed for logistic regression ([66], [79]). The group Lasso behaves like Lasso but on the group level; for large enough regularization parameters, entire groups of variables may drop out of the model. Computational methods for group Lasso in logistic regression continue to be a present area of research [98].

Limited Pairwise Multicollinearity

Multicollinearity in logistic regression models leads to the same problematic instability in parameter estimates that we observed with linear regression. Indeed, to quote [81], "Because the concern is with the relationship among the independent variables, the functional form of the model for the dependent variable is irrelevant to the estimation of collinearity." Similarly, we recommend limiting the pairwise multicollinearity present in the final logistic regression model to a tolerable threshold.

Detecting Appropriate Nonlinear Transformations

As in linear regression, the need for a nonlinear transformation of an independent variable in a logistic regression model is often detected by trial and error or by graphical examination. The Box-Tidwell procedure to automate detection of nonlinear transformations [20] can be extended to logistic regression ([62], [81]). This iterative process suggests statistically significantly power transformations of independent variables. As before, the suggested transformations need to be interpreted by a human analyst before being incorporated into a model, since they are rarely interpretable whole-number powers. They also do not take into account other constraints we would like to include.

4.5.2 Robustness

As in the linear regression case, we will approach errors in data with a robust optimization approach. Robustness in logistic regression has mainly focused on developing alternative objectives to maximum likelihood which are robust against outliers. For example, [27] suggested using a weighted maximum likelihood estimator which downweights high leverage points. [88] introduced robust M-estimates for the logistic regression model, and [13] proposed modifying this M-estimator with a correction term. [30] offered guidance for parameter choice in the estimator given in [13] to obtain estimates with bounded influence. See [76] for a detailed overview of the literature on robust logistic regression.

The MINLO approach is flexible, and can accommodate any of the proposed modifications to the logistic regression maximum likelihood function suggested in the robust statistics literature. Instead, we consider structural uncertainty in data, and address it via robust optimization. As with the other methods, this ultimately involves a modification of the traditional maximum likelihood objective.

Robust optimization directly addresses errors in the data by considering uncertainty sets for the data and calculates solutions that are immune to worst-case uncertainty under these sets (see [3] and [4]). For the logistic regression problem with data $(\mathbf{y}, \mathbf{X})$, the data associated with the independent variables have error $\mathbf{\Delta X}$ that belong to a given uncertainty set U. For example,

$$U = \left\{ \mathbf{\Delta X} \in \mathbb{R}^{n \times p} \mid \|\mathbf{\Delta x}_i\|_\alpha \leq \Gamma \right\},$$

where $\|\mathbf{x}\|_\alpha = (\sum_{l=1}^{n} x_l^\alpha)^{1/\alpha}$. The robust logistic regression problem is then

$$\min_{\boldsymbol{\beta}} \max_{\mathbf{\Delta X} \in U} \sum_{i=1}^{n} -y_i(\boldsymbol{\beta}'(\mathbf{x}_i + \mathbf{\Delta x}_i)) + \log\left(1 + \exp(\boldsymbol{\beta}'(\mathbf{x}_i + \mathbf{\Delta x}_i))\right). \tag{4.6}$$

The key result is as follows.

Theorem 4. *([7]) Problem (4.6) is equivalent to*

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} -y_i \left(\boldsymbol{\beta}'\mathbf{x}_i + (-1)^{y_i}\Gamma\|\boldsymbol{\beta}\|_{\frac{\alpha}{\alpha-1}}\right) + \log\left(1 + \exp(\boldsymbol{\beta}'\mathbf{x}_i + (-1)^{y_i}\Gamma\|\boldsymbol{\beta}\|_{\frac{\alpha}{\alpha-1}})\right). \quad (4.7)$$

Note that this differs from the recent work of [83] which considers robustness in logistic regression in a distributional sense. [83] aims to minimize the worst case expected log loss with respect to an uncertainty set consisting of a family of possible data generating distributions.

4.5.3 Modeler Expertise

As with linear regression in Section 3.2.5, there may cases where the modeler has domain knowledge about the features in the model. In that case, she might wish to specify that certain independent variables must be included in the final logistic regression model, due to a known correlation with the response. This can be incorporated directly into the model building process by adding a constraint to Problem 4.4.

4.5.4 Statistical significance

Statistical significance of logistic regression models is typically estimated via a likelihood ratio test, Wald's test, or a Lagrange multiplier test (also known as score test) ([62]). These three tests are asymptotically equivalent, and all measure some aspect of the likelihood function. However, since we consider a regularized and constrained version of maximum likelihood, none of these tests is directly applicable to our case. As in Section 3.2.6, we will maintain an assumption-free approach by using bootstrapping methods in order to estimate confidence intervals for each feature in the model selected by our algorithm.

4.5.5 Low global multicollinearity

Multicollinearity is a property of the data of the independent variables, and not of the functional form of the regression equation. So, logistic regression may be subject to the same

difficulties in catching all cases of multicollinearity using a pairwise correlation threshold as linear regression. Again, global multicollinearity can be measured by checking the condition number of the correlation matrix resulting from the submatrix of included variables. A high condition number indicates a multicollinearity problem. A condition number greater than 15 is usually taken as evidence of multicollinearity and a condition number greater than 30 is usually an instance of severe multicollinearity ([28]).

4.5.6 Formulation

To scientifically determine a logistic regression model, we propose using the framework we developed for linear regression. Therefore, we will have a preprocessing stage, followed by solving a MINLO problem for different input parameters using the method described in this chapter, and finally solving the MINLO again with any additional constraints. The heart of the method is the following MINLO problem which incorporates all of the above desirable properties. We describe the MINLO here.

$$\min_{\boldsymbol{\beta},\mathbf{z}} \quad \sum_{i=1}^{n} -y_i \left(\boldsymbol{\beta}'\mathbf{x}_i + (-1)^{y_i}\Gamma\|\boldsymbol{\beta}\|_{\frac{\alpha}{\alpha-1}}\right) + \log\left(1+\exp(\boldsymbol{\beta}'\mathbf{x}_i + (-1)^{y_i}\Gamma\|\boldsymbol{\beta}\|_{\frac{\alpha}{\alpha-1}})\right), \tag{4.8a}$$

$$\text{s.t.} \quad z_\ell \in \{0,1\}, \quad \ell = 1,\ldots,p, \tag{4.8b}$$

$$-\mathcal{M}z_\ell \leq \beta_\ell \leq \mathcal{M}z_\ell, \quad \ell = 1,\ldots,p, \tag{4.8c}$$

$$\sum_{\ell=1}^{p} z_\ell \leq k, \tag{4.8d}$$

$$z_1 = \ldots = z_\ell \quad (1,\ldots,\ell) \in \mathcal{GS}_m, \quad \forall m, \tag{4.8e}$$

$$z_i + z_j \leq 1 \quad \forall (i,j) \in \mathcal{HC}, \tag{4.8f}$$

$$\sum_{i\in\mathcal{T}_m} z_i \leq 1 \quad \forall m, \tag{4.8g}$$

$$z_\ell = 1 \quad \forall \ell \in \mathcal{I}, \tag{4.8h}$$

$$\sum_{\ell\in\mathcal{S}_i} z_\ell \leq |\mathcal{S}_i| - 1 \quad \forall \mathcal{S}_1,\ldots,S_j. \tag{4.8i}$$

In the objective function (4.8a), the robustification parameter Γ immunizes the resulting model against structural uncertainty in the data. In Constraint (4.8b), a binary indicator variable z_ℓ is introduced for every β_ℓ in the model. For a large enough constant $\mathcal{M}$, the constraint (4.8c) ensures that β_ℓ will only be included in the model if $z_\ell = 1$. The constraint (4.8d) limits the number of total variables that will be included in the model. This ensures general sparsity of the resulting model. The constraints in (4.8e), (4.8f), and (4.8g) are selective sparsity constraints. For the m^{th} set of variables with a group sparsity structure, the set of constraints defined in (4.8e) ensures that the variables in $\mathcal{GS}_m$ are either all zero, or all nonzero.

We do not describe the preprocessing initial stage or the third stage where we generate additional constraints, as these are exactly the same as the linear case with the minor difference of evaluating models based on AUC rather than R^2.

4.6 Computational Results – Algorithmic Approach

As in Chapter 3, our main goals of the algorithmic approach to logistic regression are to achieve interpretability and robustness, while retaining predictive power. We present analogous results for the logistic regression case in the tables below.

First, we present results in Tables 4.3 and 4.4 for synthetic datasets for the default parameters of the algorithm: five values of Γ tested and 0.7 as the maximum pairwise correlation allowed. These are designed to illustrate the algorithmic's approach ability to handle datasets with high multicollinearity and to be robust against added noise. Then, we tested our algorithm on five publicly-available real datasets and present these results in Table 4.5. Finally, as in Chapter 3, we consider a combined synthetic example designed to demonstrate the capacity of the algorithmic approach to identify various properties when presented in concert. Note that in all cases, all variables selected by the algorithmic approach are significant at the 0.05 level.

Preliminaries We use the same notation as in Chapter 3. Each experiment corresponds to two rows in a table. The top row presents average results over ten trials of the same experiment and the bottom row presents the standard error. We use the following notation: K^* = value of k chosen by the algorithm, TP = number of true nonzero variables identified by the algorithm, for the synthetic datasets, MC = the maximum pairwise correlation present in the final model, and Con = condition number. Time for the MINLO algorithm is presented in seconds, and is not meant to accurately benchmark the best possible time but to show that it is computationally tractable to solve these problems in a practical amount of time on standard computers. The real datasets were obtained from the University of California Irvine Machine Learning Repository ([2]). We abbreviate each real dataset's name as follows: "Bank" stands for the Banknote Authentication dataset; "Telescope" corresponds to the Magic Gamma Telescope dataset; "Mass" stands for the Mammographic Mass dataset; "Ozone 8" corresponds to the Ozone Detection Level Eight dataset; and "Ozone 1" stands for the Ozone Detection Level One dataset. We aim to return solutions in practical amounts of time, so we imposed a 60-second time limit on each optimization problem solved. Often optimality is reached before the time limit. Note that for each dataset, $K_{max} \times$ (# of values of Γ tested) $\times$ (# of iterations of Stage 3) MINLO problems are solved.

Results Table 4.3 shows results for synthetic logistic regression datasets with high pairwise multicollinearity. We observe that the MINLO model achieves the same, or slightly higher, AUC than Lasso. The MINLO model performs better in terms of sparsity, however, as noise increases, this is at the expense of recovering the true set of nonzero coefficients. However, the final Lasso models contain very high pairwise collinearity and condition numbers that indicate severe multicollinearity issues.

Table 4.4 shows results for datasets designed to illustrate robustness. The MINLO model achieves very slightly better predictive power than the Lasso model. In the highest noise setting ($\sigma = 5$), MINLO does not always fully recover the true set of nonzero coefficients. Nevertheless, the proportion of coefficients selected that are truly nonzero remains quite high on average ($4.8/5.3 = 90.6\%$) compared to Lasso ($5.0/17.4 = 28.7\%$).

Table 4.3: Pairwise Multicollinearity; n = 1000, p = 100, True K = 5, $\rho = 0.9$, $\mathbf{\Delta X = 0}$.

ρ	σ	MINLO Γ^*	K*	TP	AUC	MC	Con	Time	Lasso K*	TP	AUC	MC	Con
0.9	1	0.000	4.6	4.6	0.941	0.163	4.8	5606	23.9	5.0	0.938	0.904	84.6
		0.000	0.2	0.2	0.003	0.007	1.3	470	5.7	0.0	0.002	0.002	21.5
0.9	2	0.000	4.8	4.4	0.839	0.223	2.4	4859	20.5	4.9	0.834	0.870	69.7
		0.000	0.2	0.2	0.004	0.053	0.6	583	5.3	0.1	0.006	0.019	21.7
0.9	5	0.001	4.8	2.6	0.658	0.296	1.7	4778	27.2	3.8	0.658	0.824	109.1
		0.000	0.3	0.3	0.004	0.064	0.4	462	8.8	0.3	0.005	0.077	44.7

Table 4.4: Robustness; n = 1000, p = 100, True K = 5, $\rho = 0$, $\mathbf{\Delta X} \sim$ Uniform(0,2).

ρ	σ	MINLO Γ^*	K*	TP	AUC	MC	Con	Time	Lasso K*	TP	AUC	MC	Con
0	1	0.0000	5.1	5.0	0.873	0.057	1.2	1196	27.5	5.0	0.867	0.088	1.8
		0.0000	0.1	0.0	0.004	0.005	0.0	26	7.3	0.0	0.005	0.007	0.2
0	2	0.0002	5.2	5.0	0.793	0.059	1.2	989	19.9	5.0	0.788	0.088	1.6
		0.0001	0.2	0.0	0.007	0.004	0.0	29	5.4	0.0	0.008	0.009	0.1
0	5	0.0000	5.3	4.8	0.655	0.064	1.2	1027	17.4	5.0	0.641	0.091	1.6
		0.0000	0.2	0.1	0.007	0.005	0.0	14	4.6	0.0	0.008	0.005	0.1

We tested our algorithm on five publicly-available real datasets and present these results in Table 4.5. Note that n here indicates the size of the training dataset – the original dataset has $2n$ observations.

Table 4.5: Results for Real Datasets.

Dataset	n	p	MINLO K*	AUC	MC	Con	Time	Lasso K*	AUC	MC	Con
Bank	686	4	2.9	0.956	0.360	3.8	4.1	3.9	0.994	0.783	12.1
			0.1	0.002	0.011	0.2	1.0	0.1	0.000	0.003	0.4
Telescope	9510	10	4.8	0.832	0.668	7.2	1145.3	3.2	0.822	0.272	2.7
			0.2	0.002	0.027	0.8	166.0	0.2	0.002	0.068	0.8
Mass	415	10	4.8	0.875	0.406	6.3	24.0	6.2	0.873	0.434	9.5
			0.6	0.007	0.016	1.3	3.8	0.8	0.006	0.019	2.3
Ozone 8	924	72	3.1	0.869	0.283	2.6	1583.1	38.1	0.895	0.982	9293.8
			0.3	0.005	0.047	0.4	271.2	3.6	0.007	0.010	2827.0
Ozone 1	924	72	6.5	0.885	0.644	20.0	725.5	38.9	0.888	0.984	12283.9
			0.8	0.013	0.026	4.9	122.9	4.8	0.016	0.010	4869.6

In the Banknote Authentication dataset, MINLO achieves slightly better sparsity, but slightly worse AUC; this is the price of interpretability, since with a threshold of 0.7 as the maximum pairwise correlation, the MINLO model cannot include as many variables as the Lasso model. In the Magic Gamma Telescope dataset, however, we see the opposite result:

Lasso outperforms MINLO with respect to sparsity, at the expense of AUC. MINLO achieves a slightly higher AUC within the bounds of a 0.7 maximum pairwise correlation limit. It is not surprising that the algorithmic approach trades off the desirable properties of low multicollinearity, sparsity, and predictive performance in different ways for different datasets. In fact, what we can be assured of is that the MINLO model trades these properties off in an optimal way given the constraints the modeler specifies. It is likely that if a lower maximum correlation threshold were given, the Magic Gamma Telescope results would show a lower K^* selected – but possibly a lower test set AUC as well. We verified this intuition by running the Magic Gamma Telescope dataset again with a maximum pairwise correlation threshold of 0.5 – results are in Table 4.6. Likewise, were a higher maximum correlation threshold specified, it is likely that the Banknote Authentication test set AUC would match Lasso's – but with a higher pairwise correlation, and higher condition number. The Mammographic Mass dataset is an example where the MINLO approach outperforms Lasso on all levels: a lower K^* selected, higher test set AUC, and lower maximum correlation and condition number. The Ozone Detection datasets both have a much greater number of potential variables than the other three datasets. As we have come to expect in such cases, the MINLO model significantly outperforms Lasso with respect to sparsity here. Predictive performance is similar, although slightly lower in the MINLO case. However, the resulting maximum collinearity is drastically improved in the MINLO model.

Table 4.6: Magic Gamma Telescope Results with Maximum Pairwise Correlation Threshold of 0.5.

MINLO K*	**AUC**	**MC**	**Con**	**Time**	**Lasso K***	**AUC**	**MC**	**Con**
3.3	0.815	0.232	1.9	128.8	3.0	0.819	0.184	1.6
0.2	0.002	0.029	0.2	17.0	0.0	0.003	0.002	0.0

Finally, we consider a combined synthetic example which we created in order to test the algorithm's ability to identify many properties when presented together. Specifically, we consider an example whose structure incorporates general sparsity, selective sparsity in terms of both high pairwise multicollinearity and group sparsity, and modeler expertise in a

single dataset. We test this example in the case where $n = 2000$ and $p = 200$, and noise is high.

We generated a synthetic data matrix $\mathbf{X}$ for $n = 2000, p = 100$ according to the process outlined previously in this chapter. We used a value of $\rho = 0.9$ to ensure that there is high pairwise multicollinearity present between some columns of $\mathbf{X}$, and $\sigma = 5$ to ensure high noise. To generate nonlinear transformations, for each column j of $\mathbf{X}$ we included an additional column consisting of the squared entries of j, bringing the total number of potential covariates up to 1000. As before, we consider $k = 10$. However, we generated $\beta_i = 1$ so that 7 positive values occurred in the original 100 columns and 3 were located in the 100 transformed columns. The response $\mathbf{y}$ was generated as before as $\mathbf{y_i} = \text{Round}\left((1/\left(1 + \exp(-\boldsymbol{\beta}'\mathbf{x}_i + \epsilon_i\right))\right)$, where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ To test our robustness to error in data, we generated a matrix $\Delta\mathbf{X} \sim$ Unif(0,2) and considered $\mathbf{X} + \Delta\mathbf{X}$. We assume the modeler has some expertise with this sort of data, and knows one of the values of i such that β_i is truly nonzero. Finally, the modeler is also aware of a group sparsity structure and knows that $\beta_a, \beta_b, \beta_c$, and β_d are all either all zero or all nonzero and that $\beta_e, \beta_f, \beta_g$, and β_h are either all zero or all nonzero, where $\{a, b, c, d\} \in \{i|\beta_i = 1\}$ and $\{e, f, g, h\} \in \{i|\beta_i = 0\}$.

Table 4.7 presents results for this combined example. As before, the top row presents average results over five trials of the same experiment and the bottom row presents the standard error. In this combined example, we see that MINLO produces a much lower

Table 4.7: Results for Combined Example

MINLO Γ^*	**K***	**TP**	**AUC**	**MC**	**Con**	**Time**	**Lasso K***	**TP**	**AUC**	**MC**	**Con**
0.0004	11.2	6.2	0.76	0.56	5.9	1761.3	64.2	9.2	0.77	0.71	32.7
0.0002	0.6	0.6	0.00	0.03	0.5	40.9	6.9	0.3	0.00	0.00	1.1

total number of variables and lower pairwise multicollinearity and condition number while maintaining a similar test set AUC to the Lasso model. Although the true positive rate is lower for MINLO than Lasso in this challenging case, the precision (ratio of number of true positives chosen to total number of variables chosen) is much higher for MINLO.

The general pattern that these computational experiments of our algorithmic approach

to logistic regression is that MINLO and Lasso typically exhibit very similar predictive performance. However, MINLO is frequently able to reduce the number of variables selected and/or reduce the maximum pairwise correlation and overall multicollinearity in the model. The balance between these properties depends on the modeler's own input to the MINLO model.

4.7 Conclusions

In this chapter, we have developed a tailored algorithm for solving the best subset problem in logistic regression to provable optimality. This is the first algorithm that we are aware of to make use of callbacks within optimization software to solve a statistical problem. We have demonstrated that our algorithm converges to the optimal solution in faster times than existing off-the-shelf MINLO software. Our approach is competitive with existing sparsity-inducing heuristics for logistic regression, namely, Lasso, with respect to predictive performance. Moreover, it outperforms Lasso with respect to sparsity detection.

We have extended this technique to the framework introduced in Chapter 3, thereby developing an algorithmic approach to logistic regression. We have demonstrated the effectiveness of this approach on real and synthetic datasets in producing high-quality logistic regression models within reasonable timeframes.

Chapter 5

Conclusion

As the world is growing ever more data-rich, it is of paramount importance that we are able to make sense of this data. Statistical and machine learning models aim to learn patterns from data by maximizing signal and minimizing noise. Indeed, many statistics and machine learning problems are fundamentally optimization problems: for example, linear regression is the problem of minimizing squared error, support vector machines are margin-maximizing classifiers, and clustering algorithms try to maximize some notion of distance between groups. Often, the true underlying optimization problem that a statistical or machine learning model is trying to solve contains discrete elements. However, to date, discrete optimization methods have not played a large part in statistical modeling.

During the past twenty-five years, MIO solvers have been constantly improving in efficiency, and the effect has been compounded by the great improvements in computer hardware. In this thesis, we have aimed to dispel the popular notion that mixed integer optimization is not a viable tool in the context of statistical modeling.

In particular, we have modeled the best subset problem in linear regression as an MIO, and demonstrated that, when combined with novel continuous optimization methods, this approach is highly practical. Indeed, MIO can solve problems with thousands of observations and hundreds of variables in minutes to provable optimality, and can find near-optimal solutions in minutes in the high-dimensional regime where there are thousands of variables

and only hundreds of data points. The solutions produced by MIO methods match or improve upon the predictive performance achieved by other state-of-the-art methods, and typically outperform these methods with respect to sparsity.

In addition to modeling challenging cardinality constraints, we are able to use MIO to model a wide variety of desirable statistical properties in linear regression. Typically modelers approach model-building iteratively in order to ultimately produce a high-quality model. Our approach is the first to provide a unified framework for incorporating such properties simultaneously rather than sequentially. We have demonstrated that this methodology constructs high-quality linear regression models in practical timeframes.

Logistic regression is one of the canonical classification methods, and modelers are often faced with the same questions when building a logistic regression model that they encounter in linear regression: which variables should be included? And overall, what is the best high-quality model that can be produced from this data? We have extended our MIO approach for answering these questions to the setting of logistic regression. The tailored method we developed for solving the resulting MINLO outperforms existing MINLO solvers. As in the linear regression case, the solutions produced by our method match or improve upon the predictive performance achieved by other state-of-the-art methods, and typically outperform these methods with respect to sparsity.

The work presented in this thesis only scratches the surface of potential statistical problems where MIO methods can improve upon the status quo. We hope that our work helps pave the way for the development of more MIO-based methods in statistics.

Bibliography

[1] Francis R Bach. Consistency of the group lasso and multiple kernel learning. *The Journal of Machine Learning Research*, 9:1179–1225, 2008.

[2] Kevin Bache and Moshe Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2014. Accessed: 2014-08-20.

[3] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[4] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[5] Dimitris Bertsimas and Martin Copenhaver. Characterization of the equivalence of robustification and regularization in linear, median, and matrix regression. *Submitted to Annals of Statistics*, 2014.

[6] Dimitris Bertsimas and Apostolos Fertis. On the equivalence of robust optimization and regularization in statistics. *Technical Report*, 2009.

[7] Dimitris Bertsimas and Apostolos Fertis. Robust logistic regression. Technical report, Massachusetts Institute of Technology, 2011.

[8] Dimitris Bertsimas and Robert Freund. *Data, Models, And Decisions: The Fundamentals Of Management Science*. Dynamic Ideas Press, Belmont, Massachusetts, 2004.

[9] Dimitris Bertsimas and Rahul Mazumder. Least quantile regression via modern optimization. *Annals of Statistics*, 42(6):2494–2525, 2014.

[10] Dimitris Bertsimas and Romy Shioda. Algorithm for cardinality-constrained quadratic optimization. *Computational Optimization and Applications*, 43(1):1–22, 2009.

[11] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*. Dynamic Ideas Belmont, 2005.

[12] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. *CoRR*, abs/1209.5145, 2012.

[13] Ana M Bianco and Víctor J Yohai. *Robust estimation in the logistic regression model.* Springer, 1996.

[14] Peter Bickel, Ya'acov Ritov, and Alexandre Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, pages 1705–1732, 2009.

[15] Daniel Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical programming*, 74(2):121–140, 1996.

[16] Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica, Extra Volume: Optimization Stories*, pages 107–121, 2012.

[17] Thomas Blumensath and Mike Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5-6):629–654, 2008.

[18] Thomas Blumensath and Mike Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[19] Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In *Mixed integer nonlinear programming*, pages 1–39. Springer, 2012.

[20] George EP Box and Paul W Tidwell. Transformation of the independent variables. *Technometrics*, 4(4):531–550, 1962.

[21] Peter Bühlmann and Sara van-de-Geer. *Statistics for high-dimensional data.* Springer, 2011.

[22] Michael R Bussieck and Stefan Vigerske. Minlp solver software. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[23] E. Candes, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[24] Emmanuel Candes. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9):589–592, 2008.

[25] Emmanuel Candès and Yaniv Plan. Near-ideal model selection by ℓ_1 minimization. *The Annals of Statistics*, 37(5A):2145–2177, 2009.

[26] Emmanuel Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on*, 52(12):5406–5425, 2006.

[27] Raymond J Carroll and Shane Pederson. On robustness in the logistic regression model. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 693–706, 1993.

[28] Samprit Chatterjee, Ali S Hadi, and Bertram Price. *Regression analysis by example.* John Wiley & Sons, New York, 5th edition, 2012.

[29] Scott Chen, David Donoho, and Michael Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[30] Christophe Croux and Gentiane Haesbroeck. Implementing the bianco and yohai estimator for logistic regression. *Computational statistics & data analysis*, 44(1):273–295, 2003.

[31] Joseph Czyzyk, Michael P Mesnier, and Jorge J Moré. The neos server. *Computing in Science and Engineering*, 5(3):68–75, 1998.

[32] Dean DeCock. Ames, iowa: Alternative to the boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3), 2011.

[33] Marcel Dettling. Bagboosting for tumor classification with gene expression data. *Bioinformatics*, 20(18):3583–3593, 2004.

[34] Thomas J DiCiccio and Bradley Efron. Bootstrap confidence intervals. *Statistical Science*, pages 189–212, 1996.

[35] Elizabeth D Dolan. Neos server 4.0 administrative guide. *arXiv preprint cs/0107034*, 2001.

[36] D. Donoho. For most large underdetermined systems of equations, the minimal ℓ^1-norm solution is the sparsest solution. *Communications on Pure and Applied Mathematics*, 59:797–829, 2006.

[37] D. Donoho and I. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1993.

[38] David Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[39] David Donoho and Peter Huber. The notion of breakdown point. *A Festschrift for Erich L. Lehmann*, pages 157–184, 1983.

[40] David Donoho and Xiaoming Huo. Uncertainty principles and ideal atomic decomposition. *Information Theory, IEEE Transactions on*, 47(7):2845–2862, 2001.

[41] Norman Richard Draper and Harry Smith. *Applied Regression Analysis.* John Wiley & Sons, New York, 3rd edition, 1998.

[42] Marco A Duran and Ignacio E Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, 36(3):307–339, 1986.

[43] Bradley Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, pages 1–26, 1979.

[44] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression (with discussion). *Annals of Statistics*, 32(2):407–499, 2004.

[45] Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge University Press, London, 2012.

[46] Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360(13), 2001.

[47] Mário AT Figueiredo. Adaptive sparseness for supervised learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1150–1159, 2003.

[48] I. Frank and J. Friedman. A statistical view of some chemometrics regression tools (with discussion). *Technometrics*, 35(2):109–148, 1993.

[49] Jerome Friedman. Fast sparse regression and classification. Technical report, Department of Statistics, Stanford University, 2008.

[50] Jerome Friedman, Trevor Hastie, Holger Hoefling, and Robert Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 2(1):302–332, 2007.

[51] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[52] George M Furnival and Robert W Wilson. Regressions by leaps and bounds. *Technometrics*, 16(4):499–511, 1974.

[53] GLPK. Gnu linear programming kit. `http://www.gnu.org/software/glpk/glpk.html`, 2015. Accessed: 2015-03-06.

[54] Eitan Greenshtein. Best subset selection, persistence in high-dimensional statistical learning and optimization under ℓ_1 constraint. *The Annals of Statistics*, 34(5):2367–2386, 2006.

[55] Eitan Greenshtein and Ya'Acov Ritov. Persistence in high-dimensional linear predictor selection and the virtue of overparametrization. *Bernoulli*, 10(6):971–988, 2004.

[56] William Gropp and Jorge Moré. Optimization environments and the neos server. *Approximation theory and optimization*, pages 167–182, 1997.

[57] Frank R Hampel. A general qualitative definition of robustness. *The Annals of Mathematical Statistics*, 42(6):1887–1896, 1971.

[58] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 114. John Wiley & Sons, New York, 2011.

[59] Trevor Hastie. Trevor hastie lectures and talks. `http://www-stat.stanford.edu/~hastie/TALKS/glmnet_webinar_Rsession.tgz`, 2015. Accessed: 2015-02-11.

[60] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction (Springer Series in Statistics)*. Springer New York, 2 edition, 2009.

[61] David W Hosmer, Borko Jovanovic, and Stanley Lemeshow. Best subsets logistic regression. *Biometrics*, pages 1265–1270, 1989.

[62] David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

[63] Gurobi Inc. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2014. Accessed: 2014-08-20.

[64] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440. ACM, 2009.

[65] Adel Javanmard and Andrea Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *arXiv preprint arXiv:1306.3171*, 2013.

[66] Yuwon Kim, Jinseog Kim, and Yongdai Kim. Blockwise sparse regression. *Statistica Sinica*, 16(2):375, 2006.

[67] K. Knight and W. Fu. Asymptotics for lasso-type estimators. *Annals of Statistics*, 28(5):1356–1378, 2000.

[68] Kwangmoo Koh, Seung-Jean Kim, and Stephen P Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine learning research*, 8(8):1519–1555, 2007.

[69] Balaji Krishnapuram, Lawrence Carin, Mario AT Figueiredo, and Alexander J Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(6):957–968, 2005.

[70] Balaji Krishnapuram, AJ Harternink, Lawrence Carin, and Mario AT Figueiredo. A bayesian approach to joint feature selection and classifier design. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1105–1111, 2004.

[71] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y Ng. Efficient l˜ 1 regularized logistic regression. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 401. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[72] Richard Lockhart, Jonathan Taylor, Ryan J Tibshirani, Robert Tibshirani, et al. A significance test for the lasso. *The Annals of Statistics*, 42(2):413–468, 2014.

[73] Po-Ling Loh and Martin Wainwright. Regularized m-estimators with nonconvexity: Statistical and algorithmic theory for local optima. In *Advances in Neural Information Processing Systems*, pages 476–484, 2013.

[74] Miles Lubin and Iain Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

[75] Shuangge Ma, Xiao Song, and Jian Huang. Supervised group lasso with applications to microarray data analysis. *BMC bioinformatics*, 8(1):60, 2007.

[76] Ricardo Maronna, R. Douglas Martin, and Victor Yohai. *Robust statistics*. John Wiley & Sons, Chichester. ISBN, 2006.

[77] William F Massy. Principal components regression in exploratory statistical research. *Journal of the American Statistical Association*, 60(309):234–256, 1965.

[78] Rahul Mazumder, Jerome Friedman, and Trevor Hastie. Sparsenet: Coordinate descent with non-convex penalties. *Journal of the American Statistical Association*, 117(495):1125–1138, 2011.

[79] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, 2008.

[80] N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34:1436–1462, 2006.

[81] Scott Menard. *Applied logistic regression analysis*, volume 106. Sage, 2002.

[82] Alan Miller. *Subset selection in regression*. CRC Press Washington, 2002.

[83] Peyman Mohajerin. Distributionally robust logistic regression. *Submitted to International Conference on Machine Learning*, 2015.

[84] Balas Natarajan. Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234, 1995.

[85] George Nemhauser. Integer programming: the global impact. Presented at EURO, INFORMS, Rome, Italy, 2013. `http://euro2013.org/wp-content/uploads/Nemhauser_EuroXXVI.pdf`. Accessed: 2013-12-04.

[86] Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, 2007. Technical Report number 76.

[87] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course.* Kluwer, Norwell, 2004.

[88] Daryl Pregibon. Logistic regression diagnostics. *The Annals of Statistics*, pages 705–724, 1981.

[89] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2014.

[90] Garvesh Raskutti, Martin Wainwright, and Bin Yu. Minimax rates of estimation for high-dimensional linear regression over-balls. *Information Theory, IEEE Transactions on*, 57(10):6976–6994, 2011.

[91] Soo-Yon Rhee, Jonathan Taylor, Gauhar Wadhera, Asa Ben-Hur, Douglas L Brutlag, and Robert W Shafer. Genotypic predictors of human immunodeficiency virus type 1 drug resistance. *Proceedings of the National Academy of Sciences*, 103(46):17355–17360, 2006.

[92] Ralph Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, 1996.

[93] Peter Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984.

[94] Thomas P Ryan. *Modern regression methods*, volume 655. John Wiley & Sons, New York, 2008.

[95] Toshiki Satoa, Yuichi Takanob, Ryuhei Miyashiroc, and Akiko Yoshised. Feature subset selection for logistic regression via mixed integer optimization. *Submitted to Computational Statistics and Data Analysis*, 2015.

[96] George AF Seber and Alan J Lee. *Linear regression analysis.* John Wiley & Sons, New York, 2nd edition, 2003.

[97] Xiaotong Shen, Wei Pan, Yunzhang Zhu, and Hui Zhou. On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5):807–832, 2013.

[98] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 22(2):231–245, 2013.

[99] IBM ILOG CPLEX Optimization Studio. Cplex optimizer. `http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html`, 2015. Accessed: 2015-03-06.

[100] Barbara G Tabachnick, Linda S Fidell, et al. *Using multivariate statistics.* Allyn and Bacon, Boston, 2001.

[101] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[102] Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011.

[103] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *The journal of machine learning research*, 1:211–244, 2001.

[104] Top500.org. Top500 Supercomputer Sites, Directory page for Top500 lists. Result for each list since June 1993. `http://www.top500.org/statistics/sublist/`, 2013. Accessed: 2013-12-04.

[105] Luís Torgo. Regression datasets. `http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`, 2014. Accessed: 2014-08-20.

[106] Joel Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *Information Theory, IEEE Transactions on*, 52(3):1030–1051, 2006.

[107] Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012.

[108] Martin Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using-constrained quadratic programming (lasso). *Information Theory, IEEE Transactions on*, 55(5):2183–2202, 2009.

[109] R. Waltz. Current challenges in nonlinear optimization. `http://www.sdsc.edu/us/training/workshops/2007sac_studentworkshop/docs/SDSC07.ppt`, 2007. Accessed: 2015-03-08.

[110] Sanford Weisberg. *Applied linear regression.* John Wiley & Sons, New York, 4th edition, 2014.

[111] Larry Winner. Miscellaneous datasets. `http://www.stat.ufl.edu/~winner/datasets.html`, 2014. Accessed: 2014-08-20.

[112] Svante Wold, Arnold Ruhe, Herman Wold, and WJ Dunn, III. The collinearity problem in linear regression. the partial least squares (pls) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing*, 5(3):735–743, 1984.

[113] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *The Journal of Machine Learning Research*, 10:1485–1510, 2009.

[114] I-Cheng Yeh. Modeling of strength of high performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998.

[115] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[116] Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

[117] Cun-Hui Zhang and Jian Huang. The sparsity and bias of the lasso selection in high-dimensional linear regression. *Annals of Statistics*, 36(4):1567–1594, 2008.

[118] Cun-Hui Zhang and Tong Zhang. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, 27(4):576–593, 2012.

[119] Tong Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *The Journal of Machine Learning Research*, 11:1081–1107, 2010.

[120] Yuchen Zhang, Martin Wainwright, and Michael I Jordan. Lower bounds on the performance of polynomial-time algorithms for sparse linear regression. *arXiv preprint arXiv:1402.1918*, 2014.

[121] P. Zhao and B. Yu. On model selection consistency of lasso. *Journal of Machine Learning Research*, 7:2541–2563, 2006.

[122] Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497, 2009.

[123] H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429, 2006.

[124] Hui Zou and Runze Li. One - step sparse estimates in nonconcave penalized likelihood problems. *The Annals of Statistics*, 36(4):1509–1533, 2008.