Sparse learning: Statistical and Optimization perspectives

by

Antoine Dedieu


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

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

Signature redacted

Author ............... Sloan School of Management

May 22, 2018

Signature redacted

Certified by ............

Rahul Mazumder
Assistant Professor of Operations Research and Statistics
Thesis Supervisor

Signature redacted

Accepted by .............

Patrick Jaillet
Dugald C. Jackson Professor, Department of Electrical Engineering and
Computer Science
Co-director, Operations Research Center
Sparse learning: Statistical and Optimization perspectives

by

Antoine Dedieu

Submitted to the Sloan School of Management
on May 22, 2018, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

In this thesis, we study the computational and statistical aspects of several sparse models when the number of samples and/or features is large. We propose new statistical estimators and build new computational algorithms - borrowing tools and techniques from areas of convex and discrete optimization.

First, we explore an Lq-regularized version of the Best Subset selection procedure which mitigates the poor statistical performance of the best-subsets estimator in the low SNR regimes. The statistical and empirical properties of the estimator are explored, especially when compared to best-subsets selection, Lasso and Ridge.

Second, we propose new computational algorithms for a family of penalized linear Support Vector Machine (SVM) problem with a hinge loss function and sparsity-inducing regularizations. Our methods bring together techniques from Column (and Constraint) Generation and modern First Order methods for non-smooth convex optimization. These two components complement each others’ strengths, leading to improvements of 2 orders of magnitude when compared to commercial LP solvers.

Third, we present a novel framework inspired by Hierarchical Bayesian modeling to predict user session-length on on-line streaming services. The time spent by a user on a platform depends upon user-specific latent variables which are learned via hierarchical shrinkage. Our framework incorporates flexible parametric/nonparametric models on the covariates and outperforms state-of-th- art estimators in terms of efficiency and predictive performance on real world datasets from the internet radio company Pandora Media Inc.

Thesis Supervisor: Rahul Mazumder
Title: Assistant Professor of Operations Research and Statistics
Acknowledgments

I would like to thank my advisor, Rahul Mazumder for his strong dedication and support during the past two years. He introduced me to the fascinating universe of research, and to the beauty of applying optimization methods to build statistical estimators through our exciting projects. He gave me the freedom to explore my own ideas, and steered me in the right direction whenever he thought I needed it. This thesis work would not have been successfully conducted without his passionate participation and constant enthusiasm.

I am greatly appreciative of the support of Pandora Media Inc. In particular, I feel fortunate to have worked closely with Zhen Zhu and Hossein Vahabi. They provided me guidance and valuable comments to build and apply models to their real-world datasets. I would also like to thank Peter Radchenko for his collaboration, and for stimulating my interest in high-dimensional statistics through our discussions.

This experience would not have been the same without the people I met - at the ORC and in other departments - who have made my years at MIT unforgettable. I particularly thank Jonathan Amar, Jean Pauphilet, Pierre-Luc Vautrey, and my roomates Antoine Levy and Jad Sassine for their research advices and their friendship: you made Cambridge a stimulating environment and a home-like atmosphere. I also want to thank Andrew Vanden Berg, Nicolas Guenon des Mesnards, Maximilien Burq, Sebastien Martin, Maxime Inizan and my dear long-term friends Karl Hajjar, Ferdinand Legros, and Antoine Michon.

I express my very profound gratitude to my parents and my brother, Jules, for their unfailing support throughout my years of study. They made sacrifices to provide me with the best opportunities: your love is a strength in all the steps I take. Finally, I want to sincerely thank my love, Christelle, for enlightening my life and continuously encouraging me through this journey. I am happy to write my next chapter with you.
Contents

1 Introduction

2 Subset Selection with Shrinkage: Sparse Linear Modeling when the SNR is low 15
   2.1 Introduction ............................................. 15
   2.2 Methodological Framework ............................. 21
      2.2.1 Mixed Integer Optimization formulations ....... 22
      2.2.2 Discrete First Order Algorithms ................. 24
      2.2.3 Neighborhood continuation and local search heuristics ....... 27
   2.3 Statistical Theory ....................................... 30
      2.3.1 Comparing the Upper Bounds on the Prediction Error .... 31
      2.3.2 Comparing the Actual Prediction Errors .......... 32
      2.3.3 Outperforming the Null Model ................. 33
      2.3.4 Simultaneous Comparison with best-subsets and the Lasso .... 34
   2.4 Related work and connections to existing estimators .... 35
   2.5 Experiments ............................................. 38

3 Solving large-scale L1-regularized SVM and cousins: A cutting plane approach 47
   3.1 Introduction ............................................. 47
   3.2 Cutting planes for L1-SVM and its group extensions .... 51
      3.2.1 Primal and dual formulations of L1-SVM .......... 51
      3.2.2 Methodology for column and constraint generation .... 52
3.2.3 Column generation for L1-SVM ................................... 53
3.2.4 Combining column and constraint generation for L1-SVM .... 56
3.2.5 Application to the Group-SVM problem .......................... 59
3.3 First order methods ................................................. 61
  3.3.1 Smoothing the hinge-loss ....................................... 61
  3.3.2 Thresholding operators ......................................... 62
  3.3.3 Algorithms ...................................................... 64
  3.3.4 Scalability technique and Heuristics .......................... 66
3.4 Computation ......................................................... 68
  3.4.1 Computation results for L1-SVM .............................. 68
  3.4.2 Computation results for Group-SVM ........................... 74

4 Hierarchical Modeling and Shrinkage for User Session Length Prediction in Media Streaming ........................... 77
  4.1 Introduction ...................................................... 77
  4.2 Key idea ......................................................... 79
  4.3 Mathematical Framework ......................................... 80
    4.3.1 Review of Bayes, Empirical Bayes and MAP ............... 81
    4.3.2 Model 1: Modeling user effects ............................ 83
    4.3.3 Model 2: Modeling with covariates ....................... 84
    4.3.4 Computation via Convex Optimization .................... 88
    4.3.5 Computational Considerations ............................. 89
  4.4 Experiments ...................................................... 92
    4.4.1 Datasets .................................................... 92
    4.4.2 Evaluation .................................................. 93
    4.4.3 Comparisons ............................................... 94
    4.4.4 Effectiveness .............................................. 95
    4.4.5 Efficiency ................................................ 97
  4.5 Related work ................................................... 98
  4.6 Conclusions and future work .................................... 100
Appendices

A Appendices for Chapter 2

A.1 Computational details ................................................. 111
    A.1.1 Proof of Proposition 2 .......................................... 111
    A.1.2 Stronger formulations: adding implied inequalities ............ 112
    A.1.3 Computing problem specific parameters .......................... 113

A.2 Proofs of the results in Section 2.3 .............................. 115
    A.2.1 Proof of Theorem 1 ............................................... 115
    A.2.2 Proof of Theorem 2 ............................................... 116
    A.2.3 Proof of Lemma 1 ............................................... 117

B Appendices for Chapter 3

B.1 Proof of Theorem 3 ................................................... 119
Chapter 1

Introduction

A data revolution has emerged in the past twenty years. This shift has been driven by two major changes: (a) the astonishing advances of computer science and (b) the availability of high scale datasets with hundreds of thousands of variables. As a result, new challenges have came up in learning patterns and taking decisions from the data. Two areas of research have evolved to address these new issues. On the one hand, statistical learning has been building models and studying their predictive performance in vast amount of data. On the other hand, operations research has focused on deriving optimal solutions for complex decision-making problems of constantly increasing size.

This thesis lies at the intersection between these two booming areas of research. We consider statistical problems and build predictive models when the data is in high-dimension: the number of features / samples is very large. When the number of features is large, many statistical applications seek to obtain a parsimonious model, which delivers actionable insight for the decision-maker (e.g. gene selection problem). In Chapters 2 and 3, we deal with the problems of computing and studying sparse models for regression and classification tasks. All predictive models studied can all be formulated as optimization problems, which can be scaled up to very large dimensions with modern large-scale optimization methods. In particular, in Chapter 4 we consider a predictive task on a large real-world dataset with millons of entries. The following chapters share a similar stucture. We propose new statistical estimators for the problems studied, building computational algorithms -
borrowing tools and techniques in optimization - to achieve computational scalability and efficiency; and exploring the statistical performance of the estimators in high dimensions.

**Sparse regression:** In Chapter 2, we study the behavior of a fundamental tool in sparse statistical modeling – the best-subset selection procedure. Assuming that the underlying linear model is sparse, it is well known, both in theory and in practice, that best-subset selection works extremely well in terms of several statistical metrics (prediction, estimation and variable selection) when the signal to noise ratio (SNR) is high. However, its performance degrades substantially when the SNR is low – it is outperformed in predictive accuracy by continuous shrinkage methods, such as ridge regression and the Lasso. We explain why this behavior should not come as a surprise, and contend that the original version of the classical best-subset selection procedure was, perhaps, not designed to be used in the low SNR regimes. We propose a close cousin of best-subset selection, namely, its $\ell_q$-regularized version, for $q \in \{1, 2\}$, which (a) mitigates, to a large extent, the poor predictive performance of best-subset selection in the low SNR regimes; (b) performs favorably and generally delivers a substantially sparser model when compared to the best predictive models available via ridge regression and the Lasso. Our estimator can be expressed as a solution to a mixed integer second order conic optimization problem and, hence, is amenable to modern computational tools from mathematical optimization. We explore the theoretical properties of the predictive capabilities of the proposed estimator and complement our findings via several numerical experiments. This work has been done with the collaboration of Peter Radchneko (University of Sydney).

**Sparse classification:** In Chapter 3, we propose new computational algorithms for a family of penalized linear Support Vector Machine (SVM) problems with a hinge loss function and convex sparsity-inducing regularizations; that can be written as linear optimization problems. Our penalty functions include the usual $\ell_1$-norm on the coefficients and its grouped generalization. All problems studied here can be expressed as Linear Programs (LP) and become computationally challenging when the number of features or samples is large – the current state of specialized algorithmics for these problems is rather nascent
when compared to variants like the L2-regularized linear SVM. Our proposal brings together techniques from two apparently disparate areas of convex optimization (a) classical column (and constraint) generation methods traditionally used in the context of problems arising in operations research and (b) modern first order methods for non-smooth convex optimization. These components have their respective strengths; and while they are found to be useful as separate entities, they have not been used together in the context of solving large scale linear optimization problems arising in machine learning and in particular those studied herein. Our proposed approach complements the strengths of (a) and (b) — leading to a scheme that seem to outperform off-the-shelf commercial solvers in terms of better optimization in considerably shorter runtimes. Our methods can handle cases where both the number of samples and features are large.

**Bayesian modeling** : Finally, Chapter 4 focuses on an important metric of users’ satisfaction and engagement within on-line streaming services is the user session length, i.e. the amount of time they spend on a service continuously without interruption. Being able to predict this value directly benefits the recommendation and ad pacing contexts in music and video streaming services. Recent research has shown that predicting the exact amount of time spent is highly nontrivial due to many external factors for which a user can end a session, and the lack of predictive covariates. Most of the other related literature on duration based user engagement has focused on dwell time for websites, for search and display ads, mainly for post-click satisfaction prediction or ad ranking.

In this work we present a novel framework inspired by hierarchical Bayesian modeling to predict, at the moment of login, the amount of time a user will spend in the streaming service. The time spent by a user on a platform depends upon user-specific latent variables which are learned via hierarchical shrinkage. Our framework enjoys theoretical guarantees, naturally incorporates flexible parametric/nonparametric models on the covariates and is found to outperform state-of- the-art estimators in terms of efficiency and predictive performance on real world datasets. This project has been done in collaboration with Zhen Zhu, Hossein Vahabi (Pandora).
Chapter 2

Subset Selection with Shrinkage: Sparse Linear Modeling when the SNR is low

2.1 Introduction

We consider the usual linear regression framework, with response $y \in \mathbb{R}^{n \times 1}$, model matrix $X = [x_1, \ldots, x_p] \in \mathbb{R}^{n \times p}$ and regression coefficients $\beta \in \mathbb{R}^{p \times 1}$. We assume that columns of $X$ have been standardized to have zero means and unit $\ell_2$-norms. In many classical and modern statistical applications it is desirable to obtain a parsimonious model with good data-fidelity. Towards this end, the well-known “best-subset” regression [52] estimator (or “best-subsets” in short), given by the following combinatorial optimization problem:

$$
\hat{\beta}_{\ell_0} \in \text{argmin} \| y - X \beta \|_2^2 \quad \text{s.t.} \quad \| \beta \|_0 \leq k,
$$

is a natural candidate. Criterion (2.1) is simple to interpret: it seeks to obtain the best least squares fit with at most $k$ nonzero regression coefficients. There is a rich body of theoretical work studying the statistical properties of Problem (2.1) – see, for example, [27, 26, 62, 80] and references therein. The caveat, however, is that Problem (2.1) is usually regarded as computationally infeasible [54] – the popular R-package “leaps” can obtain solutions to Problem (2.1) for $n \geq p \approx 30$. Rigorous mathematical optimization
based approaches to compute solutions to Problem (2.1) for an arbitrary dataset have been rather scarce in the wider statistics literature – perhaps creating an aura of mystery around its operational characteristics on data-instances that arise in practice. In a recent paper, [6] demonstrate that Problem (2.1) can be solved to certifiable global optimality via mathematical optimization techniques [55, 8]; in particular, leveraging the tremendous advances in mixed integer optimization (MIO) over the past ten or so years. For additional motivation and background pertaining to MIO, we refer the reader to the recent works of [6, 49]. These works show that, despite the worst-case intractability results, subset selection can be solved for instances much larger than what was considered possible. From a practical viewpoint, this line of research has made it possible to use subset selection procedures on real and synthetic datasets, and to explore their statistical properties – the research herein is motivated by such an exploration.

**Does best-subsets overfit?** Suppose that data is generated from a linear model \( y = X\beta^* + \epsilon \) with \( \epsilon \overset{\text{iid}}{\sim} N(0, \sigma^2) \), where \( \beta^* \) is sparse, i.e., has very few nonzero elements. It is well known that if the noise level, measured by \( \sigma \), is small relative to the signal level (the \( \ell_2 \) norm of \( X\beta^* \), for example), the best-subsets estimator leads to models with excellent statistical properties [62, 80, 13] in terms of prediction, estimation and variable selection (minor additional assumptions are required for the latter two metrics). However, the situation can be quite different if the noise level is large. As we explain subsequently, this behavior is not, by any means, surprising.

To gather intuition, we first consider the Gaussian sequence model with \( n = p, X = I \) (the identity matrix), and \( y_i = \beta_i + \epsilon_i \) for \( i \in [n] \). It is well known that if \( |\beta_i| \) is comparable to \( \sigma \), then it is problematic to identify the nonzero \( \beta_i \)'s. Moreover, \( \hat{\beta}_{\ell_0} \) will be outperformed by shrinkage estimators available via ridge [32] or \( \ell_1 \) penalization, also known as the Lasso [66], in terms of the estimation and, thus, the prediction error [40]. When the features are correlated this problem is exacerbated. As a second example, we consider a general (nonorthogonal design) setting with \( \beta^* = 0 \) and use a nonzero \( k \) in Problem (2.1). Then, in terms of the prediction error, the corresponding best-subsets estimator selects the worst

\[ i \in [n] \text{ means that } i = 1, \ldots, n. \]
possible subset of all those that satisfy the cardinality constraint. More specifically, \( \hat{\beta}_{t_0} \) displays the worst predictive performance\(^2\) among all least-squares estimators with at most \( k \) nonzero coefficients. This simple example illustrates a general phenomenon in low signal regimes: best-subsets overfits, and the prediction error suffers. We examine this issue thoroughly in our theoretical and empirical analysis in Sections 2.3 and 2.5.

The best-subsets estimator, i.e., Problem (2.1), focuses on two goals: it (a) searches for the best subset of features \( \mathcal{I} \subseteq [p] \) of size \( k \) and (b) estimates \( \hat{\beta}_{t_0} \) via the (unconstrained) least squares on the selected features, i.e., \( \hat{\beta}_{t_0}(\mathcal{I}) \in \arg\min_{\theta} \| y - X_{\mathcal{I}} \theta \|_2^2 \), where \( \hat{\beta}_{t_0}(\mathcal{I}) \) denotes the entries of \( \hat{\beta}_{t_0} \) restricted to \( \mathcal{I} \); and \( X_{\mathcal{I}} \) denotes the sub-matrix of \( X \) restricted to columns \( \mathcal{I} \). Even if best-subsets selects \( \mathcal{I} \) to be the support of \( \beta^* \), the un-regularized fit on features \( \mathcal{I} \) would benefit from additional shrinkage when \( \sigma \) is large. For a simple illustration of this, consider the setting where \( n > p \) and \( k = p \). Here, \( \hat{\beta}_{t_0} \) is the usual least-squares solution, which may benefit from additional shrinkage [37] to achieve a better bias-variance trade-off in the presence of noise. Further problems may arise when the SNR is low: \( \mathcal{I} \) may be different from the support of \( \beta^* \); and there is variability associated with the choice of \( \mathcal{I} \). Many of these observations are well-known in the statistics literature. See for example, the works of [74, 17, 18] discussing the impossibility of variable selection when the signal is weak.

The explanation above suggests that best-subsets is not a good choice when the noise level is large. We contend that the vanilla best-subsets estimator was not designed to be used for low SNR regimes. Figure 2-1 presents a concrete example illustrating this point. Data is generated from an underlying linear model with \( n = 40, p = 60 \) and \( \beta_j^* = 1 \) for \( j \leq 5 \), and \( \beta_j^* = 0 \) for \( j > 5 \). The model matrix \( X \) is drawn from a multivariate Gaussian distribution with mean zero and the population pairwise correlations equal to \( \rho \). The features are standardized to have unit \( \ell_2 \)-norm, and \( \sigma^2 \) is set to match the value of SNR \( = \| X\beta^* \|_2^2 / \| \epsilon \|_2^2 \). Figure 2-1 shows the prediction error for the best-subsets estimator [computed using the framework of 6] for different values of \( k \) — the results have been averaged over ten different replications of \( (X, \epsilon) \). As expected, Figure 2-1 suggests that the predictive accuracy of

\(^2\)It selects the subset with the largest value of \( \| X\hat{\beta} \|^2 \), which in this case is proportional to the prediction error, defined as \( \frac{1}{n} \| X(\hat{\beta} - \beta^*) \|_2^2 \) for an estimator \( \hat{\beta} \).
best-subsets deteriorates as the SNR decreases – it is outperformed by continuous shrinkage methods such as ridge regression and the Lasso. In light of the preceding discussion, the overfitting behavior of best-subsets can be attributed to its aggressive search for \( \mathcal{I} \) and the fact that it does not perform any shrinkage on the selected coefficients. It is clear that the best-subsets estimator, at least in the form stated in Problem (2.1), is not a good choice in the presence of large noise.

We provide herein a theoretical explanation for the shortcomings of best-subsets when contrasted with shrinkage methods, such as the Lasso. Another natural question to ask at this point is: how do we fix this problem? Addressing this question with an associated methodological development is the main focus of this paper. We rule out the ambitious goal of correct variable selection, as this may be not be statistically possible when the noise level is high [74, 17, 18]. Instead, we focus on improving the predictive performance of the best-subsets approach, with an explicit control on the model-size – we also desire to devise an estimator that is based on a simple, transparent and easy-to-interpret optimization criterion.

Continuous shrinkage methods, such as the ridge \((q = 2)\) and Lasso \((q = 1)\):

\[
\text{minimize } \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_q^q,
\]

are well-known to produce estimators with excellent predictive performance, however, the corresponding estimated models are denser than the ones produced by best-subsets (see Figure 2-1). The Lasso searches for a set of (active) variables (indexed by \( \mathcal{J} \), say), but, unlike best-subsets (which searches for \( \mathcal{I} \) but does not shrink the coefficients), the Lasso performs an \( \ell_1 \)-penalized regression on the selected variables. The superior predictive behavior of the Lasso can be attributed in part to the shrinkage effect of the \( \ell_1 \)-penalty. Perhaps even more compelling is the example of ridge regression – there is no “searching” here per-se (as all the estimated coefficients are generally nonzero) – only shrinkage. The excellent predictive performance of ridge regression can be attributed to the pure shrinkage induced by the \( \ell_2^2 \)-penalty. The notion of “model search” is closely related to the degrees of freedom of an estimator and has been explored by [68].
Figure 2-1: Prediction errors (on the synthetic datasets described in the text) for Lasso (L1), Ridge (L2), best-subsets (LO), and the estimators proposed in Problem (2.3): “LO+L1” (for $q = 1$) and “LO+L2” (for $q = 2$). The horizontal axis represents the value of parameter $k$ for the last three methods. For LO+L1 and LO+L2, the best predictive model across $\lambda$ is plotted for every $k$. For L1 and L2, the horizontal line corresponds to the smallest prediction error – the best L1 models have (average) sizes 10.8, 18.1, 17.7 [top panel] and 10.6, 17.5, 19.9 [bottom panel], while the L2 models are completely dense. Our experimental results confirm the intuition presented in the text: in low SNR regimes L0 works poorly in terms of the prediction error – both L1 and L2 lead to smaller errors. L2 seems to be the best in terms of the prediction accuracy when SNR=0.5. L0 performs the best in terms of sparsity and the predictive accuracy when the SNR is high. The L0+L1/L0+L2 models seem to be hard to beat in terms of obtaining good predictive models that are also sparse (they are sparser than L1 but denser than L0.)
The proposed estimator: The above discussion suggests the possibility of obtaining a sparse linear model with predictive performance better than best-subsets and comparable to, or even better than, ridge regression and the Lasso. In terms of sparsity, we desire a model with fewer nonzero coefficients than the Lasso, for example. Towards this end, we propose the following regularized best-subsets estimator\(^3\):

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_q \quad \text{s.t.} \quad \|\beta\|_0 \leq k.$$ \hspace{1cm} (2.3)

Above, the cardinality constraint on $\beta$ directly controls the model size, and the $\ell_q$ penalty with $q \in \{1, 2\}$ shrinks the regression coefficients towards zero with $\lambda > 0$ as the shrinkage parameter. Furthermore, Problem (2.3) separates out the effect of shrinkage (via $\lambda\|\beta\|_q$) and sparsity (via $\|\beta\|_0 \leq k$) — this may be contrasted with the Lasso, where the tuning parameter simultaneously controls both shrinkage and sparsity, and best-subsets, which only selects but does not shrink. The family of estimators (2.3) in the special case of $\lambda = 0$ leads to the best-subsets estimator (Problem 2.1); and for $k = p$ leads to the Lasso family ($q = 1$) and the ridge family ($q = 2$) of estimators. For other values of $\lambda$ and $k$, Problem (2.3) combines the best of both worlds: best-subsets (Problem 2.1) and continuous shrinkage methods such as Lasso and ridge (Problem 2.2).

Figure 2-1 shows that when the SNR is high, the best predictive models from (2.3) coincide with best-subsets (i.e., the best choice of $\lambda$ for the true value of $k$ is close to zero). Furthermore, when $k > \|\beta^*\|_0$, continuous shrinkage regulates the overfitting behavior of best-subsets — Problem 2.3 overfits more slowly (as $k$ increases) when compared to best-subsets. This observation is also supported by our theory in Section 2.3.1. When the SNR is low, shrinkage imparted via $\ell_q$-regularization becomes critical — estimator (2.3) prefers to choose a strictly positive value of $\lambda$ to get a good predictive model. The $\ell_1$-penalty in estimator (2.3) (with $q = 1$) can also act as an additional sparsification tool when $k$ becomes large — this partially explains its (marginally) superior predictive accuracy over $q = 2$ for

\(^3\)As explained in Section 2.4, our proposed estimator (2.3) is inspired by "regularized SVD" estimators commonly used in collaborative filtering [42] and matrix completion [28].

\(^4\)Note that Problem (2.3) uses the $\ell_q$, rather than the $\ell_q^*$ penalization to be consistent with the theoretical results in Section 2.3. However, our computational framework can handle both versions of the problem.
larger SNR values. Overall, Figure 2-1 suggests that estimator (2.3) leads to models with fewer nonzeros than the Lasso; and for low SNR-values its predictive performance is similar to the best among ridge regression and/or the Lasso.

Problem (2.3) is a combinatorial optimization problem. However, it can be expressed as a mixed integer second order conic optimization (MISOCO) problem, and can be solved to certifiable optimality by leveraging advances in modern integer optimization techniques, using standard (commercial and non-commercial) solvers like Cplex, Gurobi, Knitro, Mosek, Glpk, Scip [46, 73]. To obtain good solutions to Problem (2.3) with low computational cost, we propose specialized discrete first order methods by extending the framework proposed in [6, 49]. When these algorithms are used with continuation schemes across (λ, k) and randomized local search heuristics (Section 2.2), a family of (near optimal) solutions to Problem (2.3) can be computed within a few minutes\(^5\). These algorithms however, do not certify the quality of the solutions in terms of lower-bounds on the objective function. For the latter, we need to use the power of MIO techniques. When these heuristic algorithms are used in conjunction with MISOCO solvers for Problem (2.3), they usually lead to improved computational performance – see for example, [6, 49] for similar observations on related problems.

The remainder of the paper is organized as follows. In Section 2.2 we discuss how to compute solutions to Problem (2.3); in Section 2.3 we study the theoretical properties of our proposed estimators; in Section 2.4 we discuss the connections between our proposal and existing work; and in Section 2.5 we evaluate the performance of the proposed estimators empirically. Theoretical proofs and some computational details are provided in the Supplementary Material.

## 2.2 Methodological Framework

In this section we discuss mathematical optimization methods to compute near-optimal solutions for Problem (2.3). In particular, (a) we show that Problem (2.3) can be expressed

\(^5\)A simple non-optimized Python implementation leads to a path of solutions with 100 values of λ and 15 values of k in 5 and 15 minutes for \(p = 100\) and \(p = 1000\), respectively – see Section 2.5 for more details.
as a MISOCO problem, which can be computed to certifiable (near)optimality using modern integer optimization solvers (Section 2.2.1); (b) we propose, in Section 2.2.2, first order stylized methods [56] with continuation strategies and stochastic local search heuristics [1, 53], which can be used as stand-alone methods to obtain high quality solutions to Problem (2.3) with low computational cost.

2.2.1 Mixed Integer Optimization formulations

We note that every solution to Problem (2.3) is bounded as soon as $\lambda > 0$ – this is because the level sets of the objective function are bounded\(^6\). Assuming without loss of generality that $\beta \in [-M, M]^p$, we have the following MIO representation for Problem (2.3):

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\theta\|_q \\
\text{s.t.} & \quad -Mz_j \leq \beta_j \leq Mz_j, \quad j \in [p] \\
& \quad z_j \in \{0, 1\}, \quad j \in [p] \\
& \quad \sum_j z_j = k,
\end{align*}$$

(2.4)

where $\beta$ and $z$ are the optimization variables; $M < \infty$ is a BigM parameter [8, 6], which is sufficiently large, so that a solution to Problem (2.4) is also a solution to Problem (2.3). The binary variable $z_i$ controls whether $\beta_i$ is zero or not: $z_i = 1 \implies \beta_i \in [-M, M]$ is free to vary and $z_i = 0$ implies $\beta_i = 0$. The constraint $\sum_i z_i = k$ allows at most $k$ regression coefficients to be nonzero. The nonconvexity in Problem (2.4) stems from the binary variable $z$. Problem (2.4), as written, is a general nonlinear MIO problem – we show that for both choices of $\ell_q$, the problem can be expressed as a MISOCO – a class of nonlinear MIO problems that has received a great deal of attention in the mathematical optimization

\(^6\)Boundedness can also be assumed if $\lambda$ is zero – this has been addressed in [6]
literature, see for example the recent work of [73]. Problem (2.4) can be expressed as:

\[
\begin{align*}
\text{minimize} & \quad \frac{u}{2} + \lambda v \\
\text{s.t.} & \quad \|y - X\beta\|_2^2 \leq u \\
& \quad \|\beta\|_q \leq v \\
& \quad -Mz_j \leq \beta_j \leq Mz_j, j \in [p] \\
& \quad z_j \in \{0, 1\}, j \in [p] \\
& \quad \sum_j z_j = k,
\end{align*}
\]

with the variables \((u, v, \beta, z) \subset \mathbb{R} \times \mathbb{R} \times \mathbb{R}^p \times \{0, 1\}^p\). Note that constraint (2.5a) can be expressed as a second order cone [11]

\[
\{(\beta, u) : \|y - X\beta\|_2^2 \leq u, u \geq 0\} \equiv \{(\beta, u) : \|(y - X\beta, (u - 1)/2)\|_2 \leq (u + 1)/2, u \geq 0\}.
\]

Consider the case \(\|\beta\|_q = \|\beta\|_1\). Here, the constraint (2.5b) can be expressed by introducing auxiliary continuous variables \(\tilde{\beta}_i\)'s via the following polyhedral set:

\[
\{(\beta, v) : \|\beta\|_1 \leq v, v \geq 0\} \equiv \{(\beta, v) : \exists \bar{\beta} \geq 0 \text{ s.t. } -\bar{\beta}_i \leq \beta_i \leq \bar{\beta}_i, \sum_i \bar{\beta}_i \leq v, v \geq 0\}.
\]

This shows that problem (2.5) admits a MISOCO formulation. We now consider the choice \(q = 2\), which leads to \(\|\beta\|_q = \|\beta\|_2\). In this case, the epigraph version of \(\|\beta\|_2 \leq v\) is already a second order cone, and, hence, the resultant Problem (2.5) is a MISOCO problem.

**Structured Formulations:** The computational performance of MISOCO solvers (of Gurobi, for e.g.) is found to improve by adding structural implied inequalities, or cuts, to the basic formulation (2.5). The construction of such inequalities extends the framework proposed in [6] to the penalized form of Problem (2.3); it is discussed in Section A.1.2 of the Supplementary Material. Computation of problem-specific BigM parameters and other bounds are discussed in Section A.1.3. We do not go into further detail of the advanced computational aspects of the problem, as it is not central to the main focus of the paper—we
refer the reader to [6, 49] and [73] for related discussions.

**Mixed Integer Quadratic Optimization (MIQO):** We note that Problem (2.4) with \( q = 1 \) can also be expressed as a MIQO problem. To this end, note that the least squares loss is a quadratic function in \( \beta \), and the epigraph version of \( \|\beta\|_1 \leq v \) can be expressed via linear inequalities using the extended formulation (2.6).

Also note that if we replaced \( \lambda \|\beta\|_2 \) with \( \lambda' \|\beta\|_2^2 \) in Problem (2.4), then we could readily express this problem as a MIQO. If we denote the solution to the modified problem by \( \hat{\beta}_{d_3}(\lambda', k) \), then, for every fixed \( k \), the solution path \( \{\hat{\beta}_{d_3}(\lambda', k)\}_{\lambda \geq 0} \) recovers the corresponding path for the original Problem (2.4) with \( q = 2 \). However, we will stick to the MISOCO formulation presented above in order to be consistent with our theoretical results in Section 2.3.

### 2.2.2 Discrete First Order Algorithms

In this section, we propose discrete first order (DFO) methods to obtain good upper bounds for Problem (2.3). The DFO algorithms may be perceived as adaptations of proximal gradient methods [56, 58], popularly used in convex optimization, to the composite form of Problem (2.3). The DFO methods have low iteration complexity and can gracefully exploit warm-start information across the \((\lambda, k)\)-space – with a clever combination of elaborate neighborhood continuation schemes and local combinatorial search methods they lead to near-optimal solutions to Problem (2.3). We note that these DFO methods are heuristic algorithms and are not capable of certifying the quality of the solutions via dual-bounds. For the latter, we rely on the capabilities of MIO solvers (such as Gurobi or Cplex), which work towards obtaining globally optimal solutions via a combination of upper bounds and lower bounds (i.e., dual bounds). MIO solvers accept warm-starts available from the DFO algorithm, then subsequently improve the solution and certify optimality, at the cost of additional computational time.

We describe our proposed DFO method generalizing the framework in [6] to the composite

\footnote{In our experiments, we observed that the solutions obtained by our elaborate heuristics are often close to the optimal solutions returned by the MIO solvers in the neighborhood of the optimal \((\lambda, k)\) choice (based on minimizing the prediction error on a separate validation set).}
minimize $F(\beta) := f(\beta) + \lambda \|\beta\|_q \quad \text{s.t.} \quad \|\beta\|_0 \leq k,$ \hfill (2.7)

where $f(\beta) = \frac{1}{2} \|y - \mathbf{X}\beta\|_2^2$. Our framework applies to any convex $f(\beta)$ whose gradient is Lipschitz continuous with parameter $L_0$:

$$\|\nabla f(\beta) - \nabla f(\alpha)\|_2 \leq L_0 \|\beta - \alpha\|_2 \quad \forall \beta, \alpha \in \mathbb{R}^p.$$ \hfill (2.8)

For $f(\beta) = \frac{1}{2} \|y - \mathbf{X}\beta\|_2^2$ we can use $L_0 = \sigma_{\text{max}}(\mathbf{X})^2$, where $\sigma_{\text{max}}(\cdot)$ is the maximum singular value of $\mathbf{X}$. As a consequence of (2.8), for any $L \geq L_0$, we have the following bound [56] in place:

$$f(\beta) \leq f(\alpha) + \langle \nabla f(\alpha), \beta - \alpha \rangle + \frac{L}{2} \|\beta - \alpha\|_2^2 := Q_L(\beta; \alpha), \quad \forall \alpha, \beta \in \mathbb{R}^p.$$ \hfill (2.9)

Our algorithm minimizes an upper bound to $F(\beta)$:

$$\minimize_{\|\beta\|_0 \leq k} Q_L(\beta; \alpha) + \lambda \|\beta\|_q \iff \minimize_{\|\beta\|_0 \leq k} \frac{L}{2} \left\| \beta - \left( \alpha - \frac{1}{L} \nabla f(\alpha) \right) \right\|_2^2 + \lambda \|\beta\|_q.$$ \hfill (2.10)

A key ingredient in solving the above is the thresholding operator:

$$S(\mathbf{u}; k; \lambda \ell_q) := \arg\min_{\beta, \|\beta\|_0 \leq k} \frac{1}{2} \|\beta - \mathbf{u}\|_2^2 + \lambda \|\beta\|_q,$$ \hfill (2.11)

where $S(\mathbf{u}; k; \lambda \ell_q)$ denotes the set of optimal solutions to Problem (2.11). We note that $S(\mathbf{u}; k; \lambda \ell_q)$ may be set-valued – the non-uniqueness of an optimal solution to Problem (2.11) arises from the fact that $|u_i|$’s may have ties.

**Proposition 1** Let $(1), \ldots, (p)$ be a permutation of the indices $1, \ldots, p$, such that the entries in $\mathbf{u}$ are sorted as: $|u_{(1)}| \geq |u_{(2)}| \geq \ldots \geq |u_{(p)}|$. Then, the thresholding operator (2.11) has the following form.
For the $\ell_1$-regularizer (with $q = 1$) any $\hat{\beta} \in S(u; k; \lambda \ell_q)$ is given by:

$$
\hat{\beta}_i = \begin{cases} 
\text{sgn}(u_i) \max\{|u_i| - \lambda, 0\} & i \in \{(1), (2), \ldots, (k)\} \\
0 & \text{otherwise}
\end{cases}
$$

(2.12)

For the $\ell_2$-regularizer (with $q = 2$) any $\hat{\beta} \in S(u; k; \lambda \ell_q)$ is given by:

$$
\hat{\beta}_i = \begin{cases} 
\frac{u_i}{\tau_u} \max\{\tau_u - \lambda, 0\} & i \in \{(1), (2), \ldots, (k)\} \\
0 & \text{otherwise},
\end{cases}
$$

(2.13)

where $\tau_u = \sqrt{\sum_{i=1}^k u^2(i)}$ is the $\ell_2$-norm of the $k$ largest (in absolute value) entries of $u$.

The DFO algorithm relies on the following simple update sequence (see (2.10)):

$$
\beta^{(m+1)} \in \arg\min_{\|\beta\|_0 \leq k} \left\{ \frac{L}{2} \left\| \beta - \left( \beta^{(m)} - \frac{1}{L} \nabla f(\beta^{(m)}) \right) \right\|_2^2 + \lambda \|\beta\|_q \right\}
$$

(2.14)

and the iterations are repeated until some convergence criterion is met. We summarize the DFO algorithm below for convenience.

**Discrete First Order Algorithm (DFO)**

1. Fix $L \geq L_0$ and a convergence threshold $\tau > 0$. Initialize with $\beta^{(1)}$ that is $k$-sparse. Repeat update (2.14) until $\|\beta^{(m+1)} - \beta^{(m)}\|_2^2 \leq \tau$.

2. Let $I(\tilde{\beta})$ denote the support of the $\tilde{\beta}$ obtained from Step 1, i.e., $I(\tilde{\beta}) = \{i : \tilde{\beta}_i \neq 0, i \in [p]\}$. Solve the convex problem (2.7) restricted to the support $I(\tilde{\beta})$: $\min F(\beta)$ s.t. $\beta_i = 0, i \notin I(\tilde{\beta})$.

For the sake of completeness, we establish convergence properties of the sequence $\{\beta^{(m)}\}_{m \geq 1}$ in terms of reaching a first order stationary point. Our work adapts the framework proposed in [6] to the composite form. Towards this end, we need the following definition.
Definition 1 We say that $\eta$ is a first order stationary point of Problem (2.7) if $\eta \in S(\eta - \frac{1}{L} \nabla f(\eta); k; \frac{1}{L} \ell_q)$. We say that $\eta$ is an $\epsilon$-accurate first order stationary point if $\| \eta - S(\eta - \frac{1}{L} \nabla g(\eta); k; \frac{1}{L} \ell_q) \|_2^2 \leq \epsilon$ and $\| \eta \|_0 \leq k$.

We discuss convergence properties of the sequence $\{ \beta^{(m)} \}_{m \geq 1}$ in terms of reaching a first order stationary point.

Proposition 2 Let $\{ \beta^{(m)} \}$ denote a sequence generated by Algorithm DFO. Then,

(a) for $L \geq L_0$, the sequence $F(\beta^{(m)})$ is decreasing, and it converges to some $F^* \geq 0$;

(b) for $L > L_0$, we have the following finite-time convergence rate:

$$\min_{1 \leq j \leq M} \| \beta^{(m+1)} - \beta^{(m)} \|_2^2 \leq \frac{2(F(\beta^{(1)}) - F^*)}{M(L - L_0)}.$$  

The proof of Proposition (2) appears in Section A.1.1 of the Supplementary Material.

Proposition 2 suggests that the DFO algorithm applied to Problem (2.7) leads to a decreasing sequence of objective values, which eventually converges. Under minor assumptions on the choice of $L$, the algorithm reaches an $\epsilon$-optimal first order stationary point (Definition 1) in $O(\frac{1}{\epsilon})$ many iterations. We note that the proposition makes no assumption on the data at hand – improved convergence rates may be achievable by making further assumptions on the problem data – see, for example, [6] and the discussion therein.

2.2.3 Neighborhood continuation and local search heuristics

Due to the nonconvexity of Problem (2.3), Algorithm DFO is sensitive to the initialization $\beta^{(1)}$. When $n$ is relatively small compared to $p$ and the pairwise (sample) correlations among the features are high, a base initialization, such as $\beta^{(1)} = 0$, for example, does not lead to good empirical performance (measured in terms of the objective value). These solutions can be improved, often substantially (in terms of obtaining a good objective value for Problem (2.3)), using continuation schemes and randomized local search-heuristics, as we discuss below. Furthermore, these continuation schemes lead to improved run-times.
due to the warm-starting capabilities of the DFO algorithms. We emphasize here that these algorithms can be used as stand-alone methods to obtain good feasible solutions for Problem (2.3) for a family of tuning parameters \((\lambda, k)\). They are particularly useful if a practitioner simply desires to obtain good solutions to Problem (2.3) (for a family of \((\lambda, k)\) values) in a relatively short time, instead of worrying about dual-bound certificates that state how close these solutions are to the global minimum of Problem (2.3). With low computational cost, these algorithms can be used to obtain a good estimate for the optimal tuning parameter \((\lambda, k)\), based on a separate validation set.

**Neighborhood Continuation:** Let \(\hat{\beta}(\lambda, k)\) denote an estimate obtained via the DFO algorithm for Problem (2.3) with the regularization parameter \(\lambda\) – we drop the dependence on \(q\) for notational convenience. We let \(F(\lambda, k)\) denote the corresponding objective value. We consider a 2D grid of \(\Lambda \times K = \{\lambda_1, \ldots, \lambda_N\} \times \{k_1, \ldots, k_r\}\) with \(\lambda_i > \lambda_{i+1}\) and \(k_i > k_{i+1}\) for all \(i\). We set \(k_1 = p, k_r = 1\). We set \(\lambda_1 = \|X'y\|_q\) with \(\bar{q} = \infty\) if \(q = 1\) and \(\bar{q} = 2\) if \(q = 2\) – the rationale being that if \(\lambda = \lambda_1\), then an optimal solution to Problem (2.3) is zero.

**Algorithm 1: Neighborhood Continuation**

(i) Initialize \(\hat{\beta}(\lambda_i; k_j) \leftarrow 0\) for every \(i, j \in [N] \times [r]\). Repeat Step (ii) until the array of objective values \(\{F(\lambda_i; k_j)\}_{i,j}\) stops changing between successive sweeps across the 2D grid \(\Lambda \times K\):

(ii) For \(i \in [N], j \in [r]\) do the following:

(a) Set \((\lambda, k) = (\lambda_i, k_j)\) and use the DFO algorithm with (at most) four different neighborhood initializations \(\hat{\beta}(\lambda_a; k_b)\) for \((a, b) \in \mathcal{N}(i, j)\), where \(\mathcal{N}(i, j)\) denotes the neighbors\(^8\) of \((i, j)\):

\[
\mathcal{N}(i, j) = \{(a, b) : |a - i| \leq 1, |b - j| \leq 1, |a - i| + |b - j| \leq 2, a \in [N], b \in [r]\}.
\]

For every \((a, b)\) in the neighborhood \(\mathcal{N}(i, j)\), let \(\hat{\beta}_{a,b}\) and \(F_{a,b}\) denote the corresponding estimate and objective value, respectively.

\(^8\)We note that \(\mathcal{N}(i, j)\) includes the current pair \((i, j)\) by definition.
We set $\hat{\beta}(\lambda_i; k_j)$ equal to the estimate $\hat{\beta}_{a,b}$ with the smallest objective value:

$$F(\lambda_i; k_j) = \min\{F_{a,b} : (a, b) \in \mathcal{N}(i, j)\}.$$  

**Remarks:** We make a series of remarks pertaining to Algorithm 1.

- If we denote one execution of Step-(ii) (formed by looping across all $i, j \in [N] \times [r]$) as a sweep; then successive sweeps usually lead to a strict improvement in the objective values $\{F(\lambda_i, k_j)\}_{i,j}$ for several indices $i, j$. This is because the initializations $\hat{\beta}(\lambda_a; k_b)$ for $(a, b) \in \mathcal{N}(i, j)$ potentially change across successive sweeps; resulting in new estimates $\{\hat{\beta}(\lambda_i, k_j)\}_{i,j}$.

- In the first sweep of Algorithm 1 many neighbors $\hat{\beta}(\lambda_a; r_b)$ of $(i, j)$ are zero. After the first sweep, however, all entries $(i, j)$ get populated.

- The neighborhood initializations $\hat{\beta}(\lambda_a; k_b)$ for $(a, b) \in \mathcal{N}(i, j)$ provide excellent warm-starts for the Problem (2.3) at $(\lambda_i, r_j)$. This improves the overall runtime of the algorithm (as compared to independently computing the solutions on the 2D grid); and also helps in obtaining estimates with good objective values.

**A randomized local search heuristic:** We present a local-search heuristic method, which, loosely speaking, is capable of navigating different parts of the model space via a form of stochastic search. We draw inspiration from local search schemes that are popularly used in combinatorial optimization problems [1, 53]. In our context, we use these methods as initialization schemes to the DFO algorithm – they are found to work quite well in our numerical experiments in obtaining high quality solutions to Problem (2.3). This approach is motivated by the following empirical observation: For every $(i, j)$, the estimates $\hat{\beta}(\lambda_a, k_b)$ for $(a, b)$ in the neighborhood $\mathcal{N}(i, j)$ often have similar supports – this may lead to the DFO algorithm getting trapped in a local minimizer of Problem (2.7) at $(\lambda_i, k_j)$. For a better exploration of the nonconvex landscape of Problem (2.3), we use the following stochastic search scheme: for every initialization $\hat{\beta}(\lambda_a, k_b)$, we randomly swap roughly 50% of the nonzero coefficients with 50% of the zero coefficients before passing the resulting estimate as an initialization to the DFO algorithm. This stochastic search scheme is performed as a

---

We note that, by construction, for every $(i, j) \in [N] \times [r]$, the objective value $F(\lambda_i, k_j)$ cannot increase between successive sweeps.
part of the 2D continuation scheme (described above)—we register the estimate if it leads to an improvement in the objective value.

2.3 Statistical Theory

We focus on comparing the predictive performance of the best subset selection estimator, $\hat{\beta}_{t_0}$, and the Lasso estimator, $\hat{\beta}_L$, to that of the estimators available from Problem (2.3):

$$
\begin{align*}
\hat{\beta}_1 &= \arg\min_{\beta} \|y - X\beta\|^2 + \lambda_1\|\beta\|_1 \quad \text{s.t.} \quad \|\beta\|_0 \leq k \\
\hat{\beta}_2 &= \arg\min_{\beta} \|y - X\beta\|^2 + \lambda_2\|\beta\| \quad \text{s.t.} \quad \|\beta\|_0 \leq k,
\end{align*}
$$

where, unless otherwise mentioned, $\|\cdot\|$ denotes the $\ell_2$-norm. The dependence on the tuning parameters is suppressed for notational simplicity. For the purposes of our theoretical discussion, we use the following definition of the prediction error (PE):

$$PE(\hat{\beta}) = \|X\hat{\beta} - X\beta^*\|.$$

In this section we: (A) compare the upper bounds on the prediction error for the regularized best subset estimators, $\hat{\beta}_1$ and $\hat{\beta}_2$, to those that exist for the unregularized estimator, $\hat{\beta}_{t_0}$, and demonstrate that the former are superior in the case of low signal (Section 2.3.1); (B) compare the actual prediction errors and demonstrate that the ones for the regularized estimators are smaller, with high probability, than the one for the unregularized estimator, when the signal is sufficiently low (Section 2.3.2); (C) provide a scenario where the above point is true, but the signal is strong enough for the regularized estimators to also outperform the zero vector (corresponding to the null model) in terms of the prediction error (Section 2.3.3); (D) discuss why the prediction error bounds for $\hat{\beta}_1$ compare favorably to those for $\hat{\beta}_L$ and describe the scenario where the bounds for $\hat{\beta}_1$ are simultaneously better than those for $\hat{\beta}_L$ and $\hat{\beta}_{t_0}$ (Section 2.3.4).

From here on, estimators $\hat{\beta}_1$ and $\hat{\beta}_2$ correspond to $\lambda_1 = k^{-1/2}\lambda_2 = 2\sigma\sqrt{2(1 + a)\log p}$, where $a$ can be chosen as any positive universal constant. The choice of $k$ is specified in
the presented results. We denote $\| \beta^* \|_0$ by $k^*$. Throughout this section the features are assumed to be centered with unit $\ell_2$ norms. We will use the term high probability to mean that the probability is bounded below by $1 - a_1 p^{-a_2}$, for some universal positive constants $a_1$ and $a_2$, which depend on the choice of $a$, but not on $n$, $p$, $k$ or $k^*$.

2.3.1 Comparing the Upper Bounds on the Prediction Error

For $\lambda_1 = \lambda_2 = 0$, the regularized best subset estimators coincide with the unregularized one. However, when the signal is sufficiently low, the prediction error bounds available for $\hat{\beta}_1$ and $\hat{\beta}_2$ are superior to the one for $\hat{\beta}_{k_0}$. Consider the following result, which is proved in Section A.2.1 of the Supplementary Material.

**Theorem 1** Let $k \geq k^*$. Then, there exists a universal constant $c$, such that

\[
PE^2(\hat{\beta}_1) \leq c\sigma \sqrt{\log p} \| \beta^* \|_1, \quad \text{and}
\]

\[
PE^2(\hat{\beta}_2) \leq c\sigma \sqrt{k \log p} \| \beta^* \|,
\]

with high probability.

The prediction error upper bounds for best-subsets [e.g., 62] are

\[
PE^2(\hat{\beta}_{k_0}) \leq C\sigma^2 k \log(p/k),
\]

(2.15)

where $C$ is a universal constant. The above rate is also the optimal minimax rate when $k = k^*$ [see Section D in 62, for example].

To simplify the exposition of the comparison of the bounds, we exclude the extreme settings where $k$ grows nearly as fast as $p$. More specifically, suppose that there exists a positive universal constant $b$, such that $k \leq p^{1-b}$. In this case, provided $\| \beta^* \|_1 < (bC/c)k^* \sigma \sqrt{\log p}$, the prediction error bounds for $\hat{\beta}_1$ are superior. Similarly, the bounds for $\hat{\beta}_2$ are better than the ones for $\hat{\beta}_{k_0}$ when $\| \beta^* \| < (bC/c)\sigma \sqrt{k^* \log p}$. Also note that the bounds for $\hat{\beta}_{k_0}$ grow linearly in $k$, while the ones for $\hat{\beta}_1$ do not depend on $k$. This agrees with the empirical evidence in Figure 2-1, where the predictive accuracy of $\hat{\beta}_1$ remains
stable for $k \geq k^*$, while that of $\hat{\beta}_{10}$ deteriorates as soon as $k$ is larger than $k^*$. The bounds for $\hat{\beta}_2$ grow as $\sqrt{k}$, so they are also more robust to the wrong choice of $k$ than those for $\hat{\beta}_{10}$. However, they are less robust than the bounds for $\hat{\beta}_1$. More generally, the presented bounds for $\hat{\beta}_1$ are better than those for $\hat{\beta}_2$ when $\|\beta^*\|_1/\|\beta^*\| < \sqrt{k}$.

2.3.2 Comparing the Actual Prediction Errors

Let $\sigma_{\text{max}}(\cdot)$ and $\sigma_{\text{min}}(\cdot)$ denote the largest and the smallest eigenvalue of a matrix, respectively. Given a set $\mathcal{N} \subset \{1, \ldots, p\}$, we define:

$$
\kappa_k(\mathcal{N}) = \frac{\max_{S \subseteq \mathcal{N}, |S| = k} \sigma_{\text{max}}(X_S^T X_S)}{\min_{S \subseteq \mathcal{N}, |S| = 2k} \sigma_{\text{min}}(X_S^T X_S)}.
$$

Theorem 2 below imposes an assumption that if we let $\mathcal{N}$ index the set of the noise predictors, then $\kappa_k(\mathcal{N})$ is bounded above by a universal constant. By the concentration properties of the singular values of Gaussian matrices, this condition holds with high probability in the case where the noise columns of $X$ are standardized versions of random vectors containing independent realizations of $N(0,1)$, and $(k \log p)/n$ is bounded above by an appropriate universal constant.

The next result specifies a low signal regime in which the actual prediction error for the regularized best-subsets estimators is guaranteed, with high probability, to be smaller than that for the unregularized estimator. The proof is given in Section A.2.2 of the supplement.

**Theorem 2** Let $k \geq k^*$, take $\mathcal{N}$ to be the index set of the noise predictors, and suppose that $\kappa_k(\mathcal{N})$ is bounded above by a universal constant. Suppose that $k < 2|\mathcal{N}|/3$, and there exists a positive universal constant $b$ such that $k \leq p^{1-b}$. Then, there also exist universal positive constants $c_1$ and $c_2$, for which, with high probability, inequality $\|\beta^*\|_1 < c_1\sigma\sqrt{k \log p}$ implies

$$
PE^2(\hat{\beta}_{10}) > PE^2(\hat{\beta}_1) + c_2\sigma^2k \log p, \quad \text{and}
$$

$$
PE^2(\hat{\beta}_{10}) > PE^2(\hat{\beta}_2) + c_2\sigma^2k \log p.
$$

**Remark 1** It follows from the proof that
(i) instead of being the index set of the noise predictors, $\mathcal{N}$ can be taken as any subset of 
$\{1, \ldots, p\}$, such that $|\mathcal{N}| \geq b_2 p$ for some universal positive constant $b_2$;

(ii) the assumption on the boundedness of $\kappa_k(\mathcal{N})$ can be removed at the cost of multi-
plying $c_1$, $c_2$ and $c_3$ by $\kappa_k^{-1}(\mathcal{N})$ in the three inequalities above, and replacing “high
probability” with “probability bounded below by $1 - 2p^{-c_3 k \kappa_k^{-1}(\mathcal{N})}$, for some positive
universal constant $c_3$”.

It is natural to ask whether in the low signal regime considered above our estimators can
still achieve smaller prediction error than the zero estimator, corresponding to the null
model. We answer this question affirmatively in the next section.

### 2.3.3 Outperforming the Null Model

Now we provide a scenario, where, under the setting of Theorem 2, estimators $\hat{\beta}_1$ and $\hat{\beta}_2$
outperform both $\hat{\beta}_{k_0}$ and $\hat{\beta} = 0$ in terms of prediction. Thus, while the signal is low, it is
still strong enough that taking $k = 0$ is suboptimal to $k = k^*$. Suppose that all pairwise
correlations among the signal predictor variables are bounded below by a positive universal
constant $\rho$. In other words, if $X_{S^*}$ denotes the submatrix containing the signal predictors,
$I$ is an $k^*$ by $k^*$ identity matrix, and $1$ is an $k^*$-dimensional vector of ones, then there
exists a matrix $E$ with only nonnegative entries, such that

$$X_{S^*}^T X_{S^*} = (1 - \rho)I + \rho 11^T + E.$$ 

As in the statement of Theorem 2, we suppose that $\kappa_k(\mathcal{N})$ is bounded above by a
universal constant. Assume that $\sigma = 1$, and the nonzero elements of $\beta^*$ have the same sign.
Note that

$$\|X\beta^*\|^2 = (\beta^*)^T [(1 - \rho)I + \rho 11^T + E]\beta^* \geq \rho \|\beta^*\|^2.$$ 

Also note that the prediction error for the null model is simply $\|X\beta^*\|$. Consequently,
taking advantage of Theorems 1 and 2, we can conclude that there exist universal positive
constants $c_4, c_5, c_6$, and $c_7$, such that inequalities $c_4 \sqrt{\log p} < \|\beta^*\|_1 < c_5 \sqrt{k^* \log p}$ imply,
with high probability, that

\[ PE^2(0) > PE^2(\hat{\beta}_1) + c_0 \| \beta^* \|_1^2 \]

\[ PE^2(\hat{\beta}_{\ell_0}) > PE^2(\hat{\beta}_1) + c_7 k* \log p. \]

The above result also holds for the estimator \( \hat{\beta}_2 \) if we assume that all the nonzero elements of \( \beta^* \) are equal.

### 2.3.4 Simultaneous Comparison with best-subsets and the Lasso

When the signal level is low, our regularized estimator \( \hat{\beta}_1 \) satisfies the same favorable prediction error bound as the Lasso (see Theorem 1), with the added benefit of controlling the sparsity level through the tuning parameter \( k \). On the other hand, it is well known in the statistical literature that, when the signal level is sufficiently high, the prediction error bounds for \( \hat{\beta}_{\ell_0} \) are superior to those for the Lasso estimator, \( \hat{\beta}_L \) [see, for example, 62, 61]. Our regularized estimators can take advantage of this by setting the regularization parameter to zero.

Now we show that \( \hat{\beta}_1 \) can simultaneously outperform \( \hat{\beta}_{\ell_0} \) and \( \hat{\beta}_L \) in terms of prediction. For each nonempty index set \( S \subseteq \{1, \ldots, p\} \) we define

\[ \hat{\gamma}_{S,k} = \min_{\theta \neq 0, \| \theta_S \|_0 \leq k, \| \theta_S \|_1 \leq 3 \| \theta_S \|_1} \frac{\| \mathbf{X}\theta \| \sqrt{|S|}}{\| \theta_S \|_1}, \]

and we set \( \hat{\gamma}_{\emptyset,k} = 1 \). Let \( S_* \) denote the index set of the nonzero coefficients in \( \beta^* \). Consider the following result, where we use the same (nonzero) choice of \( \lambda_1 \) as in the previous subsections. We note that this result follows via a slight modification to the proof of Theorem 6.3 in [14], which consists of incorporating the newly available bound \( \| \hat{\beta}_1 \|_0 \leq k \).

**Theorem 3** Let \( k \geq k^* \). Then, there exist universal constants \( \tilde{C} \) and \( \tilde{c} \), such that, with high probability,

\[ PE^2(\hat{\beta}_1) \leq \tilde{C} \hat{\gamma}_{S^*_*,k}^{-2} \sigma^2 |S^*_*| \log p + \tilde{c} \sigma \sqrt{\log p} \| \beta^*_{S_* \setminus S^*_*} \|_1, \]
for every choice of the subset $S'_* \subseteq S_*$. Under the setting considered at the end of Section 2.3.1, the above error bound is superior to the best-subsets bound (2.15) when there exists $S'_* \subseteq S_*$, such that

$$\|\beta'_{S_* \setminus S'_*}\|_1 < (\sigma/\delta) \sqrt{\log p} \left[ bCk^* - \tilde{C}\tilde{\gamma}_{S'_* \setminus S_*}^2, |S'_*| \right].$$

On the other hand, provided that the subset $S'_*$ is nonempty, the bound in Theorem 3 is better than the one available for the Lasso [Theorem 6.3 in 14]. More specifically, the latter bound replaces $\gamma_{S'_*,k}$ with $\gamma_{S'_*}$, where

$$\gamma_S = \min_{\theta \neq 0, \|\theta_S\|_1 \leq \|\theta_S\|_1} \frac{\|X\theta\| \sqrt{|S|}}{\|\theta_S\|_1}.$$

Note that $\tilde{\gamma}_{S,k}/\gamma_S \geq 1$ for all $k$ and $S$. Moreover, for highly correlated designs this ratio can be quite large and possibly infinite.

### 2.4 Related work and connections to existing estimators

Curiously enough, our estimator (2.3) was motivated by the “regularized SVD” estimator due to Simon Funk, popularly used in the context of collaborative filtering/matrix completion [42, also c.f. the Netflix prize]. Even though the contexts are very different, there are uncanny similarities between the estimators, as we outline below. In matrix completion, the task is to complete a partially observed matrix $\{\theta_{ij}\}$ with $(i,j) \in \Omega \subset [m] \times [n]$ by a low rank matrix $\Gamma = (\gamma_{ij})$. By using a result in [28], the regularized SVD estimator [see (2) in 42] can be shown to be equivalent to the following optimization\(^10\) problem:

$$\minimize \sum_{ij \in \Omega} (\gamma_{ij} - \theta_{ij})^2 + \lambda \sum_j \sigma_j \quad \text{s.t.} \quad \text{rank}(\Gamma) := \|\sigma\|_0 \leq k, \quad (2.16)$$

where, $\{\sigma_j\}$ denote the singular values of $\Gamma$, with $\sum_j \sigma_j$ being the nuclear norm of $\Gamma$.

The first term appearing in the objective of (2.16) is the data-fidelity term, and $\lambda \sum_j \sigma_j$

\(^10\)Problem (2.16), unlike Problem (2.3) cannot be expressed as a MISOCO problem. Heuristic solutions to Problem (2.16) can be obtained by methods described in [42, 28].
“regularizes” \( \Gamma \), the low-rank approximation to \( \{\theta_{ij}\} \). If \( \Gamma \) is a diagonal matrix with diagonal entries given by \( (\gamma_1, \ldots, \gamma_n) \), then the penalty term and cardinality constraint reduce to the penalty and constraint appearing in Problem (2.3) (with \( q = 1 \)).

Basic principles in combinatorial optimization immediately suggest natural convex relaxations for Problem (2.4) obtained by relaxing the binary variables to \( z_j \in [0, 1] \), leading to the following optimization problem:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|y - X\beta\|^2 + \lambda\|eta\|_q \\
\text{s.t.} & \quad \|eta\|_1 \leq M k
\end{align*}
\]

(2.17)

\[\iff\]

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2}\|y - X\beta\|^2 + \lambda\|eta\|_q + \delta\|eta\|_1, \\
\end{align*}
\]

(2.18)

where Problem (2.18) is the equivalent Lagrangian form of Problem (2.17) for some choice of \( \delta > 0 \). For \( q = 1 \), Problem (2.18) is a Lasso problem with regularization parameter \( \lambda + \delta \)—the sparsity inducing constraint \( \|eta\|_0 \leq k \) and \( \ell_1 \)-shrinkage (Problem (2.3)) get combined together. For \( q = 2 \), Problem (2.18) is closely related to the elastic net estimator [82] – the difference is that we penalize the \( \ell_2 \)-norm of \( \beta \) and not the squared \( \ell_2 \) norm, as is done in the elastic net. We note that the elastic net estimator was designed to encourage correlated features to have similar regression coefficients, with the \( \ell_1 \)-penalty encouraging sparsity in \( \beta \). Our experiments in Section 2.5 suggest that the elastic net leads to models that are much more dense than what is available via Lasso and hence Problem (2.3).

Estimator (2.3) bears similarities with some other proposals in the statistics literature – see for example, [24, 34, 83, 48, 21]. However, there are differences in motivation and approaches – the exact form of the estimator (2.3) does not seem to have appeared before. Our motivation is to design estimators that may “regularize” the overfitting behavior (see Figure 2-1) of subset selection in high noise regimes – we focus on getting sparse models with good predictive power. Our estimator is based on a combinatorial optimization problem, and we address the computational issues by using modern mathematical optimization methods such as MIO. Computational experiments and comparisons of our proposed estimators with some of the above methods, are presented in Section 2.5. [83] propose replacing the \( \ell_1 \) penalty in the elastic net with the adaptive lasso [81] to overcome the biasing effect of Lasso that may interfere with variable selection. In similar spirit, [34] use the nonconvex
MCP penalty [79] instead of the adaptive lasso penalty of [83]. In both these works, the authors study the penalized version of the least-squares problem and not the constrained problem with a direct control on sparsity as in (2.3). Their focus seems to be on getting superior variable selection performance/estimation error by using a (sparsity inducing) nonconvex penalty along with ridge regularization. Due to the curious structural similarities among the estimators, we contend that the estimator proposed herein with $q = 2$ will have similar operational characteristics in the context of the situations studied in [34, 83]. In addition, we believe that our proposed computational framework provides a new perspective for these prior approaches. [48] propose using a convex combination of the $\ell_0$ and the $\ell_1$ penalties on $\beta$, and study statistical properties of the estimator — their theoretical results, however, do not apply to the high-dimensional setting. [21] impose both a concave penalty and the $\ell_1$ penalty on $\beta$. Their theoretical results demonstrate that the corresponding estimator combines the predictive strength of the $\ell_1$ regularization together with the variable selection strength of the nonconvex regularization. However, the principal focus of the papers mentioned above is not on understanding and mitigating the overfitting behavior of best-subsets selection when the SNR is low, which is precisely the focus of our work.

During the final stages of this paper, we became aware of the interesting work of [31], where the authors performed a suite of experiments comparing best-subsets, the Lasso and stepwise regression procedures, expanding upon the experiments performed in [6]. Similarly to our present paper, [31] also note that in low SNR regimes the Lasso leads to better predictive models than best-subsets, while the latter dominates the Lasso for large values of the SNR. [31] demonstrate empirically that a two-stage estimation procedure: a variant of the relaxed Lasso$^{11}$ [51] leads to models with good predictive performance for both large and small SNR values. Our proposed estimator (2.3) may be contrasted with the relaxed Lasso in two important ways:

- Estimator (2.3) is characterized by a transparent optimization criterion that is easy to interpret. Furthermore, it places an explicit control on the model size via the cardinality constraint — the relaxed Lasso estimator, in contrast, exercises an implicit

---

$^{11}$The authors present a minor variant of the relaxed Lasso, which is a convex combination of the Lasso estimator and its polished version (obtained by performing a least squares fit on the Lasso support).
control on the model size via its shrinkage parameter. It is not clear to us whether the relaxed Lasso corresponds to a simple \textit{joint} optimization criterion.

- We derive comprehensive statistical properties of estimator (2.3) to understand its superior predictive performance when compared to the best-subsets and the Lasso. It may be interesting to study if the relaxed Lasso enjoys similar statistical properties.

In addition, when the SNR is large, estimator (2.3) will behave similarly to the best-subsets procedure, which is well known to have excellent variable selection properties (assuming an underlying sparse linear model). As the support of the relaxed Lasso is determined by the Lasso, this two-stage estimator will generally fall short in terms of variable selection – essentially inheriting the well-known suboptimal variable selection properties of the Lasso [80, 13, 65].

While the focus of this work is on best subset selection in the case of the least squares loss, we expect a similar story to hold for the recently proposed Discrete Dantzig Selector [49], which has superior computational performance. Our preliminary experiments (not reported here) show that, in the low SNR regimes, an $\ell_1$-regularized version of the Discrete Dantzig Selector improves upon the unregularized one.

2.5 Experiments

We explore the statistical performance of our estimator on several synthetic datasets for varying $n,p$ values with $p \gg n$, different values of SNR and correlations among the predictors; and also on a real dataset. The code implementing our proposed approach is available on the authors’ website.

\textbf{Synthetic Datasets:} The model matrix $X$ is formed by drawing $n$ independent realizations of a $p$ dimensional multivariate normal distribution with mean zero and covariance matrix $\Sigma = (\sigma_{ij})$. The columns of $X$ are standardized to have mean zero and unit $\ell_2$ norm. We generate $y = X\beta^* + \epsilon$ with $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ and $\beta^* \in \mathbb{R}^p$; and let $k^* = \|\beta^*\|_0$ denote the true number of nonzeros. Recall that $\text{SNR} = \|X\beta^*\|_2^2 / \|\epsilon\|_2^2$. We consider the following examples:
**Example 1:** We set $\sigma_{ij} = \rho^{|i-j|}, \forall i, j$ (with the convention $0^0 = 1$); $\beta^*_i = 1$ for $k^* = 7$ near equispaced values in $[p]$ and $\beta^*_i = 0$ otherwise.

**Example 2:** We set $\sigma_{ij} = \rho, \forall i \neq j$ and $\sigma_{ii} = 1$ for $i \in [p]$; $\beta^*_i = 1$ for $i \leq k^* = 7$ and $\beta^*_i = 0$ otherwise.

Note that in the above examples we take all the nonzero coefficients in $\beta^*$ to have the same magnitude – we do this to clearly understand how our proposed estimator regulates the overfitting behavior of best-subsets and compares with estimators such as ridge regression and the Lasso, as the SNR is varied. In our simulations, we also take different values of $\rho$ and $n, p$.

We conduct a comparison across the following methods:

**(L0+L1)** This is estimator (2.3) with $q = 1$. We took a 2D grid of tuning parameters as:

$\Lambda \times K$, where $\{\lambda_i\}_{1}^{N}$ is a geometrically spaced sequence of 100 values with $\lambda_1 = \|X'y\|_\infty$ and $\lambda_N \sim 10^{-4}\lambda_1$; we chose the $k$ values in $\{0, \ldots, 15\}$.

**(L0+L2)** Estimator (2.3) with $q = 2$. The 2D grid was similar to the above, with $\lambda_1$ set to $\|X'y\|_2$ (which ensures a zero solution).

**L0** Best-subsets estimator (2.1) with $k \in \{0, \ldots, 15\}$.

**L1** Lasso estimator Problem (2.2) with $q = 1$ on a grid of 100 values of $\lambda$.

**L1P** Polished version of the Lasso estimator, i.e., for every L1 solution (above) we obtain a least squares fit on its support.

**L2** Ridge regression estimator with 100 tuning parameters.

**L1+L2** Elastic net estimator [82]. For each parameter $\lambda$, we consider a sequence of 20 values $\alpha \in [0.05, 0.95]$, weighing the $\ell_1$ and $\ell_2^2$ penalties.

The estimators in (2.3) are computed with 3 rounds of the Neighborhood Continuation algorithm (Algorithm 1) with stochastic local search presented in Section 2.2.3. Let $\{\hat{\beta}(\lambda, k)\}$ denote the 2D family of solutions obtained. For $n = 50, p = 100$ computing the family $\{\hat{\beta}(\lambda, k)\}$ takes less than a minute for 5 rounds of stochastic search. For $n = 100, p = 1000$
this takes approximately 2-3 minutes on a standard Mac desktop. The DFO algorithm is run until the convergence threshold of \( \tau = 10^{-3} \).

Once the family of estimates \( \{ \hat{\beta}(\lambda, k) \} \) is obtained; the best choice \( (\hat{\lambda}, \hat{k}) \) is obtained based on a held-out validation set (discussed below). At this chosen tuning parameter \( (\hat{\lambda}, \hat{k}) \), we solve a MIO formulation (2.4) with a time-limit of 30 minutes\(^{12}\) – the resultant solutions are referred to as the L0+L1 (L0+L2) estimates. We emphasize that the heuristic methods are only used to search for an optimal tuning parameter \( (\hat{\lambda}, \hat{k}) \), and the final solution is produced by using a MIO solver warm-started with the solution obtained via the DFO algorithms. The “L0” solution is obtained similarly to above – we use \( \hat{\beta}(\lambda_N, k) \) from Problem (2.3) with \( q = 1 \) to warm-start the discrete first order algorithm (DFO). This solution is subsequently used to warm-start a MIO solver with a time-limit of 30 minutes (at the best choice of \( \hat{k} \) based on a validation set). All the common methods, L1, L1P, L2 and L1+L2, are computed using Python’s scikit-learn suite of algorithms.

The simulations with different values of \( n, p, \rho \) and SNR, and the different replications, were performed in parallel on a server, in a distributed computing platform.

**Selecting the tuning parameters:** For each of the above methods, we pick the estimator that minimizes the least squares loss on a validation set simulated as \( y = X\beta^* + \epsilon \), with the fixed \( X \) and an independent realization of \( \epsilon \) from \( N(0, \sigma^2) \) (with the same SNR). We then compute the obtained estimator’s prediction error computed as: \( \frac{1}{n} \| X\hat{\beta} - X\beta^* \|_2^2 \); and the associated sparsity level, i.e., the number of nonzero regression coefficients. The results are averaged over 10 independent simulations. Figures 2-2, 2-3 and 2-4 summarize the results via box plots – extending from the lower to the upper quartile of the data, with a line at the median – to aggregate the results over 10 simulations. We do not present the sparsity levels of L1+L2 and L2, as these are considerably denser than L1 (which in turn, leads to the most dense solutions among all competing methods).

**Summary of observations:** We summarize our general observations below:

- For a low SNR (SNR=1), L0 performs poorly in terms of prediction accuracy, due to the high level of noise in the problem. To mitigate its overfitting effect (in the

\(^{12}\)We used a Python interface to the Gurobi solver for our experiments.
Example 1: Small settings: \( n = 50, p = 100 \)

\( \rho = 0.5, \text{SNR} = 1 \)

\( \rho = 0.2, \text{SNR} = 2 \)

\( \rho = 0.5, \text{SNR} = 3 \)

Example 1: Large settings: \( n = 100, p = 1000 \)

\( \rho = 0.2, \text{SNR} = 1 \)

\( \rho = 0.2, \text{SNR} = 2 \)

\( \rho = 0.8, \text{SNR} = 3 \)

Figure 2-2: Example 1 simulations for different values of \( n, p, \rho \) and SNR. Prediction error refers to the best predictive models obtained after tuning on a separate validation set. Sparsity refers to the corresponding number of nonzero coefficients. For low SNR values \( L_0 \) leads to poor predictive models; and is dominated by \( L_1 \) and \( L_2 \). The best predictive models are available from \( L_0+L_1/L_0+L_2 \) – on occasions they are comparable to the best \( L_1/L_2 \) models, but are much more sparse. As the SNR becomes larger, \( L_0 \) starts improving. For low SNR and large \( \rho \) values ridge regression seems to work very well in terms of prediction error (though the models are dense).
Example 2: Small settings: $n = 50, p = 100$

- $\rho = 0.2, \text{SNR} = 1$
- $\rho = 0.2, \text{SNR} = 2$
- $\rho = 0.2, \text{SNR} = 3$

Figure 2-3: Experimental results for Example 2. The results are qualitatively similar to Figure 2-2 however, this example is “harder” than Example 1 due to the uniform correlations across all pairs of features (in the population) - a larger nominal value of SNR is required before L0 matches the performance of L0+L1/L0+L2. The L0+L1/L0+L2 models seem to work the best across all the regimes. The regularized subset selection methods seem to work the best in terms of obtaining a good prediction model with few nonzeros - the model sizes are larger than $k^*$ but smaller than the best L1 models.
presence of large noise) L0 realizes that it needs to regularize more – it does so by selecting a very sparse model – the best L0 predictive model has fewer nonzeros than $\beta^*$. In all these cases, methods L1 and L2 work better than L0 in terms of the prediction accuracy. However, the estimated models are rather dense. The polished version of the Lasso: L1P, selects a model that is sparser than the Lasso – but it suffers in prediction accuracy.

The two new methods, L0+L1 and L0+L2, are comparable – they can be hardly beaten in terms of the prediction accuracy. They fix the overfitting behavior of L0 via the additional shrinkage. The best predictive models available from L0+L1/L0+L2 are similar to that of the best predictive models available via L1 and L2, however, they lead to models that are significantly sparser (i.e., fewer nonzeros). These observations are consistent with the discussion and the theoretical results in Sections 2.3.1 and 2.3.2.

The L0 models are sparser than those for L0+L1 and L0+L2 – but, as we have mentioned before, L0 suffers in terms of the prediction accuracy. In summary, the methods L0+L1/L0+L2 significantly improve upon the predictive performance of L0 at the cost of marginally decreasing the model sparsity (when compared to L0 and also $k^*$).

- When the SNR becomes larger, L0+L1 and L0+L2 start behaving similarly to L0 in terms of both sparsity and the prediction accuracy. Additional shrinkage marginally helps the prediction accuracy, but the model sparsity becomes comparable to L0 – and both of them are concentrated around $||\beta^*||_0$. This is also consistent with our theoretical findings, more specifically those in Section 2.3.4. L1 is better than both L1+L2 and L2. L1 benefits from polishing – L1P gets closer to L0 in terms of the prediction accuracy but selects a denser model.

**Comparisons with Adaptive Elastic net, Mnet and Relaxed Lasso:** We present some computational results that compare our proposal with the methods Mnet [34], Adaptive elastic net [83] and the relaxed lasso estimator [31]. As mentioned before, it seems that the Mnet and Adaptive elastic net were designed to *mimic* the behavior of the elastic net, with the intention of encouraging greater sparsity and improved estimation error by using a
Figure 2-4: Aggregated results over 10 simulations for our methods, L0+L1 and L0+L2, and the adaptive elastic net (AEN), Mnet (MCP) and Relaxed Lasso (RL1) methods (as described in the text). Here, $n = 100$, $p = 1000$, $\rho = 0.2$; SNR = 2 for Example 1 and $\rho = 0.1$, SNR = 3 for Example 2. We observe that our proposed methods seems to work better in terms of both the prediction accuracy and model sparsity.

nonconvex penalty on $\beta$ instead of the usual $\ell_1$-norm. Our proposed estimator with $q = 2$ can be certainly useful in the regimes where [34, 83] are found to be useful — however, our motivation to study the proposed estimator (2.3) is different. We (empirically) observe important differences in the statistical performances of the estimators proposed in [34, 83] and (2.3). These differences are likely a consequence of (a) the optimization algorithms and (b) the exact forms of the estimators, including the choice of the penalty function.

Figure 2-4 shows comparisons for $n = 100$, $p = 1000$ (for data generated as per Examples 1 and 2). For the adaptive elastic net, we used the function gcdnet from the R package gcdnet with weights chosen based on Example 1 in [83]. For the Mnet method we use the ncvreg function from the R package ncvreg with the MCP penalty and ridge regularization. The tuning parameters are chosen based on a held-out validation set (similarly to the description above). For the relaxed lasso estimator, we used the code available in [31]. For all these methods including ours, we used the same number of tuning parameters. In summary, it seems that estimator (2.3) leads to models with (significantly) higher sparsity.

---

13[34] use a coordinate descent method directly on the $\ell_2^2$+MCP penalized problem; and [83] solve an $\ell_2^2$ + adaptive lasso regularized least squares, which is a convex problem.

14For Mnet we took 15 values of the tuning parameter to combine the ridge and MCP penalty, and 100 tuning parameters for the MCP penalty. We used a similar choice for adaptive elastic net. For relaxed Lasso, we took 15 values of the tuning parameter to create a convex combination of the lasso and the polished lasso estimator.
and often better predictive performance.

Figure 2-5: Figure showing performance of different methods (prediction error and sparsity) for different real datasets. We observe that pure L0 underfits (as it selects a model with very few nonzeros); L0+L1/L2 seem to work well both in terms of prediction error and in obtaining a model with fewer nonzeros, when compared to the best models available from L1. L1 leads to models with good predictive accuracy, but at the cost of larger number of nonzeros (higher sparsity).

**Real Datasets:** We now consider the performance of the different methods on some real-datasets, as described below.

**Triazine dataset:** We obtained this dataset from the **libsvm** website. The dataset contains 186 observations and 60 features, to which we added 500 noisy features (i.e., standard Gaussian noise).

**Riboflavin dataset:** This dataset pertains to riboflavin production for \( n = 71 \) observations of Bacillus subtilis. For each observation, there are \( p = 4088 \) gene expression features.

**Leukemia dataset** We consider the publicly available leukemia classification dataset.\(^{17}\)

\(^{15}\)The url is [https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/triazines](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/triazines)

\(^{16}\)The dataset is taken from the R package **hdi**

\(^{17}\)The dataset is taken from [http://cilab.ujn.edu.cn/datasets.htm](http://cilab.ujn.edu.cn/datasets.htm)
with 72 samples and 7129 features. We keep the top 2000 features based on correlation screening; and create a semi-synthetic dataset as \( y = X\beta^0 + \epsilon \) with \( \epsilon_i \overset{iid}{\sim} N(0, \sigma^2) \); where, \( \beta^0_i \in \{0, 1\} \) with \( k_0 = 10 \) randomly chosen coefficients equal to 1, and the rest are 0. We set SNR=4.

**Rat dataset:** This is the rat microarray dataset studied in [75]. We use the same processing steps as the authors, that is, we analyze the RNA from the eyes of 120 twelve-week old male rats by considering 18,975 probes which are expressed in the eye tissue.

We centered and standardized each dataset and the associated response. Then, for each example, we split randomly the original dataset into a new train and test set, compute all the estimators and keep the estimator with lowest test accuracy. Figure 2-5 displays the results averaged over 10 random splits.

\(^{18}\)We thank Dr. Haolei Weng for giving us the dataset and the preprocessing code.
Chapter 3

Solving large-scale L1-regularized SVM and cousins: A cutting plane approach

3.1 Introduction

SVMs [71, 30] are a powerful method that are used as a staple for classification problems. Given training data \( \{(x_i, y_i)\}_{i=1}^n : (x_i, y_i) \in \mathbb{R}^p \times \{-1, 1\} \) the task is to learn a linear classifier of the form \( \text{sign}(x^T \beta + \beta_0) \), where, \( \beta_0 \in \mathbb{R} \) is the offset. The popular L2-regularized linear SVM (L2-SVM) performs this by considering the minimization problem

\[
\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \sum_{i=1}^n (1 - y_i(x_i^T \beta + \beta_0))_+ + \frac{\lambda}{2} \| \beta \|_2^2 ,
\]

where, \( (a)_+ := \max\{a, 0\} \) denotes the maximum of \( a \) and 0; and is often noted as the hinge-loss function. Several significant advances have been made to speed up L2-SVM training, and we provide a rather incomplete list here. [10] proposes stochastic gradient descent; [64] uses primal subgradient descent; [33] proposes a dual coordinate descent approach. [39, 23] develop Cutting-Plane methods that scale linearly with the sample size. The L2-SVM estimator leads to dense regression coefficients – towards this end, the L1 penalty [12, 30] might be used as a convex surrogate to encourage sparsity in the coefficients.
This leads to the L1-SVM problem:

\[
\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \sum_{i=1}^{n} \left( 1 - y_i (x_i^T \beta + \beta_0) \right)_+ + \lambda \| \beta \|_1, \tag{3.2}
\]

which can be written as an LP. The L1-SVM estimator is known to encourage sparsity in \( \beta \), i.e., many coefficients will be exactly equal to 0. \( \lambda \geq 0 \) controls the amount of regularization. Off-the-shelf solvers, including commercial LP solvers have difficulties to scale and solve Problem (3.2) when \( p \) is large (at least tens of thousands). To this end, [29] proposes a homotopy based method to compute the entire (piecewise linear) regularization path in \( \beta \), [3] uses the Alternating Direction Method of Multipliers; and [77] propose a coordinate descent algorithm on the dual of a regularized version of Problem (3.2). A recent method [59] proposes a parametric simplex approach to solve L1-SVM. We later refer to this method in our experiments.

In some applications, the sparsity is structured — i.e., the coefficient indices are naturally found to occur in groups that are known a-priori (depending upon the application); and it is desirable to select (or set to zero) a whole group together as a “unit”. In this context, a group version of the usual L1 regularization is often used to improve the performance and interpretability of the model [78, 35]. In this paper, we consider the popular L1/L\( \infty \) penalty [2] leading to the Group-SVM problem:

\[
\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \sum_{i=1}^{n} \left( 1 - y_i (x_i^T \beta + \beta_0) \right)_+ + \lambda \sum_{g=1}^{G} \| \beta_g \|_{\infty}, \tag{3.3}
\]

where, \( g \in [G] \) denotes a group (the groups are disjoint), \( \beta_g \) denotes the subvector of coefficients belonging to group \( g \); and \( \beta = (\beta_1, \ldots, \beta_G) \). Problem (3.3) can also be expressed as an LP.

**What this paper is about:** First order methods have enjoyed a great deal of success in solving large scale structured convex optimization problems arising in machine learning applications. Many of these methods (such as proximal gradient and its accelerated variants) are particularly appealing both in theory and practice for problems [58] involving
the minimization of a smooth function — they also extend to many problems of the composite form [58]. These algorithms however, do not directly apply to the nonsmooth problems discussed above. Accelerated proximal gradient methods can be applied on a smoothed version (aka approximate version) of the hinge loss function [57] — a method that we explore in Section 3.3. Loosely speaking, the smoothing operation hurts the overall runtime of the algorithm (when compared to a smooth minimization problem) — in practice, first-order methods may take several iterations towards obtaining a near-optimal solution to the original nonsmooth problem.

We pursue a different approach — we attempt to directly solve the LP exploiting the special structure that solutions to these problems have. For example, in Problem (3.2), large values of $\lambda$ will encourage an optimal solution to Problem (3.2), $\hat{\beta}$ (say), to be sparse (i.e., few nonzero entries). If $p \gg n$ this sparsity might be exploited in developing efficient algorithms for Problem (3.2) — in principle, it is possible to obtain a solution to Problem (3.2) without even creating an LP model with all $p$ variables. One classical method commonly used to solve large scale LPs where many of the variables are anticipated to be zero are column generation methods, a method that originated as early as 1958 [22].

On a different note, let us consider another special structural aspect of a solution to Problem (3.2) when $n$ is large (and $p$ is not too large). In this case, at an optimal solution, many of the points $y_i$'s (in the training dataset) will be correctly classified — that is, $\alpha_i := 1 - y_i(x_i^T\beta + \beta_0) < 0$ and hence $\tilde{\alpha}_i := (\alpha_i)_+ \geq 0$ will be zero for many indices $i \in [n]$. The contribution to the loss term from these correctly classified points will be zero — this might be leveraged to develop efficient algorithms for Problem (3.2). As we will see in Section 3.2, we will use a special form of the cutting plane method [], namely, constraint generation to exploit the sparsity in $\tilde{\alpha}_i$'s. This will be critical for problems where $n$ is very large, and hence, Problem (3.2) can be solved without creating an LP model with as many samples.

To summarize, there are two characteristics special to an optimal solution of Problem (3.2): (a) sparsity in the SVM coefficients, i.e., $\beta$ and/or (b) sparsity in $\tilde{\alpha}_i$'s aka sparsity in the support vectors. column generation can be used to handle (a); constraint generation can be used to address (b) — in problems where, both $n, p$ are large, we propose
to combine both column and constraint generation. To our knowledge, while column generation and constraint generation are used separately in the context of solving large scale LPs, using them together, especially in the context of the current application, is to the best of our knowledge novel. In addition, these cutting plane methods, are found to benefit from good initializations — towards this end, we resort to techniques in first order optimization, alluded to above. Roughly speaking, these first order methods enable us to get approximate solutions with low computational cost — and we subsequently refine these solutions via resorting to the LP cutting plane methods. While the idea above was motivated via Problem (3.2), this applies to Problem (3.3). To our knowledge, this is the first paper that discusses the notion of bringing together techniques from first order methods in convex optimization and cutting plane algorithms for solving large scale LPs, in the context of solving a problem of key importance in machine learning.

Organization of paper The rest of this paper is organized as follows. Section 3.2 reviews cutting plane techniques (with a particular emphasis on column and constraint generation methods) for LP and discusses its application to L1-SVM and Group-SVM problems. Section 3.3 discusses first order methods for non-smooth convex optimization. Section 3.4 shows the performances of our methods compared to commercial LP solvers and other state of the art algorithms on synthetic and real datasets.

Notation For an integer \( a \) we use \( i \in [a] \) to denote \( i \in \{1, 2, \ldots, a\} \). The \( i \)th entry of a vector \( \mathbf{u} \) is denoted by \( u_i \). For a set \( \mathcal{A} \), we use the notation \( |\mathcal{A}| \) to denote its size.
3.2 Cutting planes for L1-SVM and its group extensions

3.2.1 Primal and dual formulations of L1-SVM

We present an LP formulation for Problem (3.2). We denote the positive and negative parts of every coefficient \( \beta_i \) as \( \beta_i^+ = \max\{\beta_i, 0\} \), \( \beta_i^- = \max\{-\beta_i, 0\} \).

\[
\begin{align*}
\text{(L1-SVM)} & \quad \min_{\xi \in \mathbb{R}^n, \beta^+, \beta^- \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \quad \sum_{i=1}^{n} \xi_i + \lambda \sum_{j=1}^{p} \beta_j^+ + \lambda \sum_{j=1}^{p} \beta_j^- \\
& \quad \text{s.t.} \quad \xi_i + y_i x_i^T \beta^+ - y_i x_i^T \beta^- + y_i \beta_0 \geq 1 \quad i \in [n] \\
& \quad \quad \quad \xi \geq 0, \beta^+ \geq 0, \beta^- \geq 0.
\end{align*}
\]

(3.4)

The feasible set of Problem (3.4) is nonempty. A dual is the following LP:

\[
\begin{align*}
\text{(Dual-L1-SVM)} : \quad & \max_{\pi \in \mathbb{R}^n} \sum_{i=1}^{n} \pi_i \\
& \text{s.t.} \quad -\lambda \leq \sum_{i=1}^{n} y_i x_{ij} \pi_i \leq \lambda \quad j \in [p] \\
& \quad \quad \quad y^T \pi = 0 \\
& \quad \quad \quad 0 \leq \pi_i \leq 1 \quad i \in [n].
\end{align*}
\]

(3.5)

From (3.4) and (3.5) we obtain the standard complementary slackness conditions of an optimal solution to the L1-SVM problem:

\[
(1 - \pi_i) \xi_i = 0 \quad \pi_i (\xi_i + y_i x_i^T \beta + y_i \beta_0 - 1) = 0 \quad i \in [n].
\]

(3.6)

Let \( (\beta^*(\lambda), \beta_0^*(\lambda)) \) and \( \pi^*(\lambda) \) denote optimal solutions for Problems (3.4) and (3.5). In what follows, for notational convenience, we will drop the dependence on \( \lambda \) when there is no confusion. In standard SVM terminology [30], the vectors on the correct side of the margin satisfy \( \xi_i = 0, \pi_i = 0 \) and those on the wrong side satisfy \( \xi_i > 0, \pi_i = 1 \). Samples that satisfy the latter condition, are often called the "support vectors". For a large value of \( \lambda \) the \( \ell_1 \) penalty selects a small number of features; and a majority of the the samples
are on the wrong side of the margin. When \( \lambda \) decreases, the size of the support of \( \beta^* \) generally increases whereas the size of the support (i.e., the number of nonzero entries) of \( \pi^* \) generally decreases.

### 3.2.2 Methodology for column and constraint generation

Column generation [22, 7] is a well-known method often used to solve large scale LPs with a large number of columns, wherein, we anticipate that an optimal solution will be comprised of only a very small subset of columns. The basic idea is to start with a candidate set of columns and incrementally add additional columns into the model, based on some criteria, until some optimality conditions are met. Consider the primal LP problem \((P)\) where, \( \bar{n}, \bar{p} \) are integers, \( A \in \mathbb{R}^{\bar{n} \times \bar{p}}, \ b \in \mathbb{R}^{\bar{n}}, \ c \in \mathbb{R}^{\bar{p}} \) are problem data. We assume that the optimal cost of \((P)\) is finite. The strong duality theorem [see 7, Theorem 4.4] guarantees that this cost is equal to the optimal cost of the dual problem \((D)\):

\[
(P) : \min_{\theta \in \mathbb{R}^{\bar{p}}} \ c^T \theta \\
\text{s.t. } A\theta \geq b, \ \theta \geq 0
\]

\[
(D) : \max_{q \in \mathbb{R}^{\bar{n}}} \ q^T b \\
\text{s.t. } q^T A \leq c, \ q \geq 0.
\]

We assume the rows of the matrix \( A \) are linearly independent\(^1\). We use \( A_i \) to denote the \( i \)th column of \( A \); and \( a_i \) to denote the \( i \)th row of \( A \). A subset \( B = \{B(1), \ldots, B(\bar{n})\} \subset \{1, \ldots, \bar{p}\} \) is said to define a basis if the columns of the matrix \( B = \{A_{B(1)}, \ldots, A_{B(\bar{n})}\} \) are linearly independent. The associated solution to the primal problem is \( \theta = (\theta_B, 0_{\bar{p}-\bar{n}}) \) with \( \theta_B = B^{-1}b \) and is feasible if \( B^{-1}b \geq 0 \). To lower the reduced cost \( c^T \theta \), we select a non basic variable \( \theta_j, j \notin B \) and increase it to a nonnegative value \( \mu \) in a direction \( d \in \mathbb{R}^{\bar{p}} \) such that \( \theta + \mu d \) is feasible for \((P)\). The change in the objective value of Problem \((P)\) is equal to \( \mu \tilde{c}_j \) where \( \tilde{c}_j \) is the reduced cost of the variable \( \theta_j \), defined as \( \tilde{c}_j = c_j - c_B^T B^{-1} A_j \).

The simplex algorithm solves problem \((P)\) by looking at every iteration for a non basic variable with negative reduced cost. If such a variable cannot be found, the current solution is optimal and the algorithm terminates. The basis \( B \) also determines a basic solution to

---

\(^1\)Such conditions are satisfied for the L1-SVM LP \((3.4)\)
the dual Problem (D) given by \( q^T = c_g B^{-1} \): it is feasible if \( q \geq 0 \) and if all the reduced costs are nonnegative, since \( \hat{c}_j = c_j - q^T A_j, \forall j \).

**Column generation:** We now consider the case \( \tilde{p} \gg \tilde{n} \). A solution to Problem (P) has at most \( \tilde{n} \) nonzeros coefficients. Most of the columns will never enter the basis, hence we can reduce the number of variables to consider. Starting from a subset of variables \( J \subset \{1, \ldots, \tilde{p} \} \), we define the *restricted columns* problem

\[
\begin{align*}
\min_{\theta \in \mathbb{R}^{|J|}} & \quad \sum_{j \in J} c_j \theta_j \\
\text{s.t.} & \quad \sum_{j \in J} A_j \theta_j \geq b, \quad \theta \geq 0.
\end{align*}
\]

(3.8)

The common version of the column generation algorithm adds to the set \( J \) a non basic variable with lowest negative reduced cost and solves the updated problem. The algorithm terminates after a finite number of iterations when no such variable can be found [cf 7, Section 6.2].

**Constraint generation:** When \( \tilde{n} \gg \tilde{p} \), at an optimal solution to (P) we expect only a small fraction of the \( \tilde{n} \) constraints \( a_i^T \theta \geq b_i \) to be active or binding — hence an optimal solution can be obtained by considering only a small subset of the \( \tilde{n} \) constraints. This inspires the use of a constraint generation algorithm: we start from a subset of constraints \( I \subset \{1, \ldots, \tilde{n} \} \) solve the restricted problem with these constraints; and add the most violated constraint. We will not provide further details, as this method is equivalent to applying column generation [7] to the dual Problem (D).

### 3.2.3 Column generation for L1-SVM

We consider the L1-SVM Problem (3.2) for a given \( \lambda \) — our objective is to able to solve this problem with \( n \) a few hundred and \( p \) up to a million\(^2\). Suppose we have a set of candidate variables \( J \subset \{1, \ldots, p \} \) (the choice of this initial set of variables is important from a

\(^2\)When \( p \) is a few thousand, LP solvers will work quite well and hence the cutting plane techniques discussed herein are not required. For larger \( n \) values we will need constraint generation as in Section 3.2.4
computational viewpoint, and we address this aspect in Section 3.2.3). As presented in Section 3.2.2, we form the \textit{restricted columns} L1-SVM problem

\[
\min_{\xi \in \mathbb{R}^n, \beta^+ \in \mathbb{R}^{|\mathcal{J}|}} \sum_{i=1}^{n} \xi_i + \lambda \sum_{j \in \mathcal{J}} \beta^+_j + \lambda \sum_{j \in \mathcal{J}} \beta^-_j \\
\text{s.t.} \quad \xi_i + \sum_{j \in \mathcal{J}} y_i x_{ij} \beta^+_j - \sum_{j \in \mathcal{J}} y_i x_{ij} \beta^-_j + y_i \beta_0 \geq 1 \quad i \in [n] \\
\xi \geq 0, \quad \beta^+ \geq 0, \quad \beta^- \geq 0.
\] (3.9)

Similar to Problem (3.5), we can get a dual of the restricted problem \( \mathcal{M}_{\ell_1} ([n], \mathcal{J}) \) — let \( \pi^* \in \mathbb{R}^n \) be an optimal solution to this restricted dual. For every variable pair \( \beta^+_j, \beta^-_j, j \notin \mathcal{J} \), the minimum of their reduced costs\(^3\) \( \overline{\beta}^+_j, \overline{\beta}^-_j \) (see Section 3.2.2) is,

\[
\min \{ \overline{\beta}^+_j, \overline{\beta}^-_j \} = \lambda - \left| \sum_{i=1}^{n} y_i x_{ij} \pi^*_i \right|.
\] (3.10)

For a tolerance threshold \( \epsilon > 0 \) (We use \( \epsilon = 10^{-2} \) in our experiments), we update the set \( \mathcal{J} \) by adding all the two columns \( \beta^+_j \) and \( \beta^-_j \) such that the minimum of their reduced costs is lower than \(-\epsilon\). We process iteratively until no column is added, as presented in the following algorithm.

\textbf{Algorithm 1:} \textit{Column generation for L1-SVM}

\textbf{Input:} \( X, y \), regularization parameter \( \lambda \), a convergence threshold \( \epsilon > 0 \), an initial set of columns \( \mathcal{J} \), an initial vector \( \beta^* \in \mathbb{R}^{|\mathcal{J}|} \).

\textbf{Output:} A near-optimal solution \( \beta^* \) for the L1-SVM Problem (3.2).

1. Repeat Steps 2 to 3 until \( \mathcal{J} \) stabilizes.

2. Solve the model \( \mathcal{M}_{\ell_1} ([n], \mathcal{J}) \) in Problem (3.9).

3. Form the set \( \mathcal{J}^\epsilon \) of columns in \( \{1, \ldots, p\} \setminus \mathcal{J} \) with reduced cost lower than \(-\epsilon\). Update \( \mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}^\epsilon \) and solve the LP in Step 2, with warm-starting.

\(^3\)For variable \( \beta^+_j \), we denote its corresponding reduced cost by \( \overline{\beta}^+_j \). A similar notation is used for \( \beta^-_j \).
We solve the first model for the initial set of columns \( J \), and warm-start the next LPs solved.

**Initializing column generation with a candidate set of columns**

In practice, the overall column generation algorithm is found to benefit from a good initial choice for \( J \). Ideally, one would like to get a reasonable estimate for \( J \), with low computational cost. To this end, we observed the following schemes to be quite useful:

(a) (Correlation screening) A simple approach, based on correlation screening, selects \( J \) as a subset (of size close to \( n \)) of variables with highest absolute correlations with \( y \).

(b) (Regularization path) In this method, we compute a path of solutions to L1-SVM with column generation for a decreasing sequence of \( \lambda \in \Lambda \) values, with the smallest one set to the current value of interest. This is discussed in Section 3.2.3. This can also be used to compute a regularization path for the L1-SVM problem.

(c) (first order methods) Section 3.3 discusses the use of techniques from first order optimization to estimate a subset \( J \).

**Computing a regularization path with column generation**

Note that the subgradient conditions of optimality for the L1-SVM Problem (3.2) is given by:

\[
\lambda \text{sign}(\beta_j^*) = \sum_{i=1}^{n} y_i x_{ij} \pi_i^* ,
\]

where, \( \text{sign}(u) \) denotes a subgradient of \( u \mapsto |u| \). For all values of \( \lambda \) larger than \( \lambda_{\text{max}} = \max_j \sum_{i=1}^{n} |x_{ij}| \), an optimal solution to Problem (3.2) is zero: \( \beta^*(\lambda) = 0 \).

Let \( \mathcal{I}_+ \), \( \mathcal{I}_- \) denote the sample indices corresponding to the classes with labels +1 and -1; and let \( N_+ \), \( N_- \) denote their respective sizes. Suppose that (without loss of generality), \( N_+ \geq N_- \). Then for \( \lambda \geq \lambda_{\text{max}} \) a solution to Problem (3.5) is \( \pi_i(\lambda) = N_- / N_+ \), \( \forall i \in \mathcal{I}_+ \) and \( \pi_i(\lambda) = 1 \), \( \forall i \in \mathcal{I}_- \). For \( \lambda = \lambda_{\text{max}} \) using (3.10), the minimum of the reduced costs of the variables \( \beta_j^+ \) and \( \beta_j^- \) is

\[
\min \{ \beta_j^+(\lambda_{\text{max}}), \beta_j^-(\lambda_{\text{max}}) \} = \lambda_{\text{max}} - \left| \frac{N_-}{N_+} \sum_{i \in \mathcal{I}_+} y_i x_{ij} + \sum_{i \in \mathcal{I}_-} y_i x_{ij} \right| . \quad (3.11)
\]
When $\lambda = \lambda_1$ is slightly smaller than $\lambda_{\max}$, (3.11) suggests to run a column generation algorithm by selecting $J$ as a small subset of variables that minimize the right-hand side of (3.11). When the dataset is balanced and the columns of the design matrix $(X_1, \ldots, X_p)$ are mean-centered and standardized, $J$ corresponds to the columns with highest absolute correlation with the output $y$. Once the solution to Problem (3.2) is obtained for $\lambda_1$, we can compute a solution for a smaller value of $\lambda$ by using the warm-start capabilities of a simplex-based LP solver, along with column generation. This suggests that one can compute an entire regularization path via the column generation algorithm — starting from $\lambda_0 = \lambda_{\max}$, we solve Problem (3.2) for a decreasing sequence of $\lambda$ values till we reach the desired value of $\lambda$. The algorithm is summarized below, for convenience.

**ALGORITHM 2: Regularization Path algorithm for L1-SVM**

**Input:** $X$, $y$, convergence tolerance $\varepsilon$, a grid of decreasing $\lambda$ values: $\{\lambda_0 = \lambda_{\max}, \ldots, \lambda_M = \lambda\}$, a small integer $j_0$.

**Output:** A near-optimal regularization path $\{\beta^*(\lambda_0), \ldots, \beta^*(\lambda_M)\}$ for the L1-SVM Problem (3.2).

1. Let $\beta^*(\lambda_0) = 0$ and estimate $J(\lambda_0)$ by obtaining the $j_0$ variables minimizing the rhs of (3.11).

2. For $l \in \{1, \ldots, M\}$, initialize $J(\lambda_l) \leftarrow J(\lambda_{l-1})$, $\beta^*(\lambda_l) \leftarrow \beta^*(\lambda_{l-1})$. Run the column generation algorithm to obtain the new estimate $\beta^*(\lambda_l)$ with $J(\lambda_l)$ denoting the corresponding set of columns.

**3.2.4 Combining column and constraint generation for L1-SVM**

We now address another important case where, the number of observations $n$ is large (up to hundreds of thousands). Recall that for the L1-SVM Problem (3.2), the samples that characterize the hyperplane (corresponding to a solution of the problem) are the support vectors on the wrong side of the margin. An optimal solution can be obtained, if we can identify these support vectors — significant computational gains are possible if one might obtain these support vectors from a suitable initialization, without having to ever form the
whole LP model with all the $n$ samples. To this end, we use constraint generation ideas (Section 3.2.2). We first present the case where $n$ is large but $p$ is small (Section 3.2.4), and then discuss the case where, both $n, p$ are large (Section 3.2.4).

**Constraint generation for large $n$ and small $p$**

**Restricted dual problem:** We consider the case where, $n$ is large and $p$ is small. Suppose that for a given $\lambda > 0$, we have an initial estimate $I \subset \{1, \ldots, n\}$ of critical constraints.

We define the restricted constraints L1-SVM problem as

$$
\text{min}_{I \subset \{1, \ldots, n\}} \sum_{i \in I} \xi_i + \lambda \sum_{j=1}^{p} \beta_j^+ + \lambda \sum_{j=1}^{p} \beta_j^-
$$

subject to

$$
\xi_i + \sum_{j=1}^{p} y_i x_{ij} \beta_j^+ - \sum_{j=1}^{p} y_i x_{ij} \beta_j^- + y_i \beta_0 \geq 1 \quad i \in I
$$

$$
\xi \geq 0, \beta^+ \geq 0, \beta^- \geq 0.
$$

(3.12)

A dual of Problem (3.12) is given by:

$$
\text{min}_{\pi \in \mathbb{R}^{|I|}} \sum_{i \in I} \pi_i
$$

subject to

$$
-\lambda \leq \sum_{i \in I} y_i x_{ij} \pi_i \leq \lambda \quad j \in [p]
$$

$$
\sum_{i \in I} y_i \pi_i = 0
$$

$$
0 \leq \pi_i \leq 1 \quad i \in I.
$$

(3.13)

Let $(\beta^*, \beta_0^*) \in \mathbb{R}^{p+1}$ and $\pi^* \in \mathbb{R}^{|I|}$ denote optimal solutions of Problem (3.12) and (3.13) (respectively). We add to $I$ all the dual variables with reduced cost higher than a threshold $c$. They correspond to violations of constraints in the primal Problem (3.12) — this follows by observing that the reduced cost $\pi_i$ of a dual variable $\pi_i$, $i \notin I$ (cf. Section 3.2.2) is

$$
\pi_i = 1 - y_i (x_i^T \beta^* + \beta_0^*).
$$

Consequently, we summarize the constraint generation algorithm for L1-SVM below:
Algorithm 3: Constraint generation for L1-SVM

Input: $X, y$, a regularization coefficient $\lambda$, a threshold $\epsilon > 0$, an initial set of constraints $\mathcal{I}$, an initial vector $\beta^* \in \mathbb{R}^p$.

Output: A near-optimal solution $\beta^*$ for the L1-SVM Problem (3.2).

1. Repeat Steps 2 to 3 until $\mathcal{I}$ stabilizes.

2. Solve Problem $\mathcal{M}_{\ell_1}(\mathcal{I}, [p])$ in Problem (3.12).

3. Let $\mathcal{I}^c \subset \{1, \ldots, n\} \setminus \mathcal{I}$ denote constraints with reduced cost higher than $\epsilon$. Update $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}^c$.

Initialization: As in the case for column generation, the success of the constraint generation procedure relies on a good initialization scheme — to this end, we found the first-order methods described in Section 3.3.3 to be particularly useful.

These first order methods suffer from increased computational cost when $n$ becomes large, due to the large cost associated with gradient computations — thusly motivated, Section 3.3.4 proposes a sampling procedure that obtains estimators on different subsamples of the data; and averages the estimators to obtain an estimate of the violated constraints $\mathcal{I}$.

A column-and-constraint generation algorithm when both $n$ and $p$ are large

When both $n$ and $p$ are large, we propose combining column and constraint generation to solve L1-SVM. For a given $\lambda$ let $\mathcal{I}$ and $\mathcal{J}$ denote subsets of relevant columns and constraints (respectively). We define the restricted constraints-columns L1-SVM problem as follows:

$$
\begin{aligned}
\mathcal{M}_{\ell_1}(\mathcal{I}, \mathcal{J}) & \min_{\xi \in \mathbb{R}^{||\mathcal{I}||}, \beta_0 \in \mathbb{R}, \beta^+, \beta^- \in \mathbb{R}^{||\mathcal{J}||}} \quad \sum_{i \in \mathcal{I}} \xi_i + \lambda \sum_{j \in \mathcal{J}} \beta_j^+ + \lambda \sum_{j \in \mathcal{J}} \beta_j^- \\
& \text{s.t.} \quad \xi_i + \sum_{j \in \mathcal{J}} y_i x_{ij} \beta_j^+ - \sum_{j \in \mathcal{J}} y_i x_{ij} \beta_j^- + y_i \beta_0 \geq 1 \quad i \in \mathcal{I} \\
& \quad \xi \geq 0, \beta^+ \geq 0, \beta^- \geq 0.
\end{aligned}
$$

Let $(\beta^*, \beta_0^*) \in \mathbb{R}^{||\mathcal{J}||+1}$, $\pi^* \in \mathbb{R}^{||\mathcal{I}||}$ be a pair of optimal primal and dual solutions. We add variables to the sets $\mathcal{J}$ and $\mathcal{I}$ by noting that the minimum of the reduced costs of a couple
of primal pair variables $\beta_j^+, \beta_j^-$, $j \notin J$ and the reduced cost of a dual variable $\pi_i$, $i \notin I$ respectively are

$$\min \{\beta_j^+, \beta_j^-\} = \lambda - \left| \sum_{i \in I} y_i x_{ij} \pi_i \right| ; \quad \pi_i = 1 - y_i \left( \sum_{j \in J} x_{ij} \beta_j^* + \beta_0 \right).$$  (3.15)

We discuss in Section 3.3.4 how to use first order optimization methods for to initialize $I$ and $J$. With the formulations of Problem (3.14) and Equation (3.15), we propose the following algorithm which combines column generation and constraint generation to solve the L1-SVM Problem (3.2).

**Algorithm 4: Combined column-and-constraint generation for L1-SVM**

**Input:** $X$, $y$, a regularization coefficient $\lambda$, a threshold $\epsilon > 0$.

**Output:** A near-optimal solution $\beta^*$ for the L1-SVM Problem (3.2).

1. Initialize $\beta^*$, $I$ and $J$ with the method described in Section 3.3.4. Repeat Steps 2 to 4 until $I$ and $J$ stabilize.

2. Solve the model $\mathcal{M}_{\ell_1}(I, J)$ in Problem (3.14).

3. Let $I^c \subset \{1, \ldots, n\} \setminus I$ denote constraints with reduced cost higher than $\epsilon$. Update $I \leftarrow I \cup I^c$.

4. Let $J^c \subset \{1, \ldots, p\} \setminus J$ denote columns with reduced cost lower than $-\epsilon$. Update $J \leftarrow J \cup J^c$.

### 3.2.5 Application to the Group-SVM problem

We now discuss how the framework presented above can be adapted to address the Group-SVM i.e., Problem (3.3). We denote by $I_g$, $g \in [G]$ the set of coefficient indices belonging to group $g$.

**Column generation:** Below we present an LP formulation for Problem (3.3). To this end, we introduce the variables $v \in \mathbb{R}^G$ such that $v_g$ refers to the $L_\infty$-norm of the coefficients
in group $g$.

\[
\text{(Group-SVM)} \quad \min_{\xi \in \mathbb{R}^n, \beta_0 \in \mathbb{R}, \beta^+, \beta^- \in \mathbb{R}^p, \nu \in \mathbb{R}^G} \sum_{i=1}^{n} \xi_i + \lambda \sum_{g=1}^{G} v_g \\
\text{s.t.} \quad \xi_i + y_i x_i^T \beta^+ - y_i x_i^T \beta^- + y_i \beta_0 \geq 1 \quad i \in [n] \\
\quad \quad \quad \nu_g - \beta^+_j - \beta^-_j \geq 0 \quad j \in \mathcal{I}_g, \ g \in [G] \\
\quad \quad \quad \xi \geq 0, \ \beta^+ \geq 0, \ \beta^- \geq 0, \ \nu \geq 0. \tag{3.16}
\]

A dual of Problem (3.16) is given by:

\[
\text{(Dual-Group-SVM)} \quad \max_{\pi \in \mathbb{R}^n} \sum_{i=1}^{n} \pi_i \\
\text{s.t.} \quad \sum_{j \in \mathcal{I}_g} \left| \sum_{i=1}^{n} y_i x_{ij} \pi_i \right| \leq \lambda \quad g \in [G] \\
\quad \quad \quad y^T \pi = 0 \\
\quad \quad \quad 0 \leq \pi_i \leq 1 \quad i \in [n]. \tag{3.17}
\]

The dual uses the $\ell_\infty - \ell_1$ norm, the dual of the regularization. We extend the idea in Section 3.2.2 and apply column generation on the groups, by bringing groups with lowest reduced cost into the model. The reduced cost $\bar{\beta}_g$ of group $g$ is expressed as:

\[
\bar{\beta}_g = \lambda - \sum_{j \in \mathcal{I}_g} \left| \sum_{i=1}^{n} y_i x_{ij} \pi_i \right|. \tag{3.18}
\]

**Computing a regularization path:** The regularization path algorithm presented in Section 3.2.3 can be adapted to the Group-SVM problem. first, note that:

\[
\beta^*(\lambda) = 0, \quad \forall \lambda \geq \lambda_{\max} = \max_{g \in [G]} \sum_{j \in \mathcal{I}_g} \sum_{i=1}^{n} |x_{ij}|
\]

For $\lambda = \lambda_{\max}$, the reduced cost for group $g$ is given by the "group" analogue of (3.11):

\[
\bar{\beta}_g = \lambda_{\max} - \sum_{j \in \mathcal{I}_g} \left| \frac{N_-}{N_+} \sum_{i \in \mathcal{I}_+} y_i x_{ij} + \sum_{i \in \mathcal{I}_-} y_i x_{ij} \right|. \tag{3.19}
\]
As in Section 3.2.3, we can obtain a small set of groups maximizing (3.19) — we use these groups to initialize the LP solver to solve Problem (3.16) for the next smaller value of \( \lambda \). The problem can then be solved via column generation (for example) — this will lead to important computational savings, for a large number of groups. We repeat this process for smaller values of \( \lambda \).

**First order methods:** first order methods can also be used to obtain approximate solutions to the Group-SVM problem — they are discussed in Section 3.3.2.

**Constraint generation and column generation:** When \( n \) is large, constraint generation can be used for the Group-SVM problem in a manner similar to that used for the L1-SVM problem. Similarly, the methods of column and constraint generation can be applied together to obtain computational savings when both \( n \) and the number of groups are large.

### 3.3 First order methods

This section presents first order methods for non-smooth convex optimization [57] which deliver a low-accuracy solution quickly and initialize the algorithms introduced in Section 3.2 for L1-SVM and Group-SVM. These are all non-smooth convex problems with a special structure – amenable to the smoothing scheme pioneered by [57]. We propose to use this scheme for a few iterations, to obtain an approximate solution – nevertheless this provides an estimate for the sets of columns (resp. constraints) necessary for the column (resp. constraint) generation methods.

#### 3.3.1 Smoothing the hinge-loss

We start by building a sequence of differentiable convex functions with continuous Lipschitz gradient \( F^\tau \), converging to the hinge-loss for \( \tau \to 0 \). Let us first note that \( \max\{0, x\} = \frac{1}{2}(x + |x|) = \max_{|w| \leq 1} \frac{1}{2}(x + wx) \) as this maximum is achieved for \( \text{sign}(x) \). Consequently
the hinge-loss in the SVM problem can be expressed as a maximum over the $\ell_\infty$ unit ball:

$$\sum_{i=1}^{n} (1 - y_i(x_i^T \beta + \beta_0))_+ = \max_{\|w\|_\infty \leq 1} \sum_{i=1}^{n} \frac{1}{2} [1 - y_i(x_i^T \beta + \beta_0) + w_i(1 - y_i(x_i^T \beta + \beta_0))] .$$

We apply the technique suggested by Nesterov in [57] and define for a $\tau > 0$ the smooth hinge-loss

$$F^\tau(\beta, \beta_0) = \max_{\|w\|_\infty \leq 1} \left\{ \sum_{i=1}^{n} \frac{1}{2} [1 - y_i(x_i^T \beta + \beta_0) + w_i(1 - y_i(x_i^T \beta + \beta_0))] - \frac{\tau}{2} \|w\|_2^2 \right\} .$$

We will use (Accelerated) Gradient Descent [5] to solve the following smooth SVM Problem

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} F^\tau(\beta, \beta_0) + \Omega(\beta), \quad (3.20)$$

where in the case of our study, $\Omega(\beta) = \lambda \|\beta\|_1$ for L1-SVM and $\Omega(\beta) = \lambda \sum_{g=1}^{G} \|\beta_g\|_\infty$ for Group-SVM.

We introduce $z, w^T \in \mathbb{R}^n : z_i = 1 - y_i(x_i^T \beta + \beta_0), \forall i$ and $w_i^T = \min \left(1, \frac{1}{2\tau} |z_i|\right) \text{sign}(z_i), \forall i$. We compute the gradient of $F^\tau$ and its Lipschitz constant with Proposition (3), using [57]:

$$\nabla F^\tau(\beta, \beta_0) = -\frac{1}{2} \sum_{i=1}^{n} (1 + w_i^T) y_i(x_i^T, 1)^T \in \mathbb{R}^{p+1} . \quad (3.21)$$

**Theorem 1** Let $X = (X_1, \ldots, X_p, e) \in \mathbb{R}^{n \times (p+1)}$ be the design matrix to which we have added the column $e = (1, \ldots, 1)^T \in \mathbb{R}^n$ and let $\mu_{\text{max}}(X^TX)$ be the highest eigenvalue of the matrix $X^TX$. Then $\nabla F^\tau$ is Lipschitz continuous with constant $C^\tau = \mu_{\text{max}}(X^TX)/4\tau$.

### 3.3.2 Thresholding operators

For compact notation, we set $\gamma = (\beta, \beta_0) \in \mathbb{R}^{p+1}$. Following [5] for $L \geq C^\tau$ we can upper bound the function $F^\tau$ around any $\alpha$ by the following quadratic form $Q_L$:

$$F^\tau(\gamma) \leq Q_L(\alpha, \gamma) := F^\tau(\alpha) + \nabla F^\tau(\alpha)^T(\gamma - \alpha) + \frac{L}{2} \|\gamma - \alpha\|_2^2, \forall \alpha, \gamma \in \mathbb{R}^{p+1}. \quad (3.22)$$
The smooth SVM Problem (3.20) minimizes the sum of the convex loss function $F^\tau$ with continuous Lipschitz gradient and a penalization on $\beta$. We approximate its solution by considering the quadratic loss $Q_L$ and solving:

$$\hat{\gamma} \in \arg\min_{\gamma=(\beta,\gamma^0)} Q_L(\alpha, \gamma) + \Omega(\beta) = \arg\min_{\gamma} \frac{1}{2} \left\| \gamma - \left( \alpha - \frac{1}{L} \nabla F^\tau(\alpha) \right) \right\|^2 + \frac{1}{L} \Omega(\beta). \quad (3.23)$$

We note $\hat{\gamma} = (\hat{\beta}, \hat{\gamma}^0)$. $\hat{\gamma}^0$ is simple to compute. To compute $\hat{\beta}$, we need to compute the following thresholding operator, for any $\mu > 0$:

$$S_{\mu\Omega(\cdot)}(\eta) \in \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2} \left\| \beta - \eta \right\|^2 + \mu \Omega(\beta).$$

Consequently, the general solution of Problem (4.14) depends upon $S_{\Omega(\cdot)}$. Below are given explicit examples of $S_{\mu\Omega(\cdot)}$, for $\Omega(\cdot)$ being the $\ell_1$ or $\ell_1 - \ell_\infty$ regularizations.

**Thresholding operator for the $\ell_1$ regularization:** In the case of the $\ell_1$ regularization, the thresholding operator is given by:

$$S_{\lambda \ell_1}(\eta) = \left( S_\lambda^0(\eta_1), \ldots, S_\lambda^0(\eta_p) \right),$$

where for $c \in \mathbb{R}$ and $\lambda > 0$, $S^0$ is the soft-thresholding operator defined as:

$$S_\lambda^0(c) = \max (0, |c| - \lambda) \text{sign}(c) = \arg\min \frac{1}{2} (x - c)^2 + \frac{\lambda}{L} |x|.$$

**Thresholding operator for $\ell_1 - \ell_\infty$ regularization:** Before we derive the thresholding operator for Group-SVM, let us introduce the following operator which computes the projection onto an $\ell_1$ ball with radius $\lambda/L$:

$$\hat{S}_{\frac{\lambda}{L}, \ell_1}(\eta) \in \arg\min_{\beta} \frac{1}{2} \left\| \beta - \eta \right\|^2 \text{ s.t. } \frac{L}{\lambda} \left\| \beta \right\|_1 \leq 1. \quad (3.24)$$
From standard results, Moreau decomposition [2] gives the relation:

\[ S_{\frac{1}{2} \| \cdot \|_\infty} + \tilde{S}_{\frac{1}{2} \| \cdot \|_1} = Id. \]  

(3.25)

and the projection on the \( \ell_1 \) ball is done effectively using [69] [70]. Now, for Group-SVM, by using separability of the thresholding operators across the different groups, Problem (4.14) can be expressed as:

\[ \arg\min_{\beta} \sum_{g=1}^{G} \left\{ \frac{1}{2} \| \beta_g - \eta_g \|_2^2 + \frac{\lambda}{L} \| \beta_g \|_\infty \right\}. \]  

(3.26)

As a consequence of Equation (3.25), for each group, we solve Problem (3.26) by aggregating the estimators \( S_{\frac{1}{2} \| \cdot \|_\infty}(\eta_g) \) among the different groups.

### 3.3.3 Algorithms

We propose two types of algorithms relying on Gradient Descent and Block Coordinate Descent to solve the smooth SVM Problem (3.20) using the thresholding operators defined in Section 3.3.2.

**Gradient Descent**

The following algorithm applies the (accelerated) gradient descent method [5, 57] on the smooth SVM Problem (3.20). We iterate the descent until a stopping condition is satisfied or a maximum number of iterations \( T_{\text{max}} \) is reached.

**Accelerated Gradient Descent algorithm**

**Input:** \( X, y \), a regularization coefficient \( \lambda \), a smoothing coefficient \( \tau \), a stopping criterion \( \eta \), a maximum number of iterations \( T_{\text{max}} \).

**Output:** An approximate solution \( \beta \) of the smooth SVM Problem (3.20).

1. Initialize \( T = 1, q_1 = 1, \beta_1 = \delta_0 \) to zero. \( \nabla F^r, C^r \) are as defined in (B.1) and Theorem (3).
2. While $\|\beta_T - \beta_{T-1}\|_2 > \eta$ and $T < T_{\text{max}}$ do:

(a) Compute $\delta_T = S_{\lambda/C_T}\left(\beta_T - \frac{1}{C_T} \nabla F^T(\beta_T)\right)$.

(b) Define $q_{T+1} = \frac{1 + \sqrt{1 + 4q_T^2}}{2}$ and compute $\beta_{T+1} = \delta_T + \frac{q_T - 1}{q_{T+1}} (\delta_T - \delta_{T-1})$.

Note that without Step 2(b), the previous algorithm would be a usual Gradient Descent.

**Block Coordinate Descent**

In the linear regression case, efficient algorithms for group regularizations use Block Coordinate Descent [61]. We adapt this idea herein in the case of Group-SVM: at every iteration $T$, the Gradient Descent algorithm computes one gradient step and uses the thresholding operator for the $\ell_1 - \ell_{\infty}$ regularization. Alternatively, Block Coordinate Descent loops iteratively over the $G$ groups: it updates the block of coefficients $\beta_g$ on group $g$ by considering the $\ell_{\infty}$ proximal operator and solving:

$$\beta_{g}^{t+1} \in \arg\min_{\beta_g} \frac{1}{2} \left\| \beta_g - \beta_g^t - \frac{1}{C_g^T} \left\{ \nabla F^T(\beta_g^{t+1}, \ldots, \beta_g^{t-1}, \beta_g^t, \ldots, \beta_G^t, \beta_0^t) \right\} \right\|_2^2 + \frac{\lambda}{C_g} \|\beta_g\|_{\infty},$$

(3.27)

where $\{\nabla F^T\}_{I_g}$ notes the gradient restricted to its components in $I_g$; and $C_g^T$ is its associated Lipschitz coefficient on group $g$ – defined as $C_g^T = \mu_{\text{max}}(X_g^T X_g)/4\tau$.

**Residual updates:** When updating one block, we do not need to update the whole gradient as a residual update method reduces the number of computations. Indeed, for block $g$, Equation (B.1) gives:

$$\{\nabla F^T(\beta, \beta_0)\}_{I_g} = -\frac{1}{2}X_g^T \{y * (1 + w^T)\},$$

where $*$ is the element-wise multiplication. If $w^T$ is known, this step requires $n|I_g|$ operations. In addition, we have to update $w^T$ – which is recalled to be defined as: $w_i^T = \min\left(1, \frac{1}{2\tau}|z_i|\right) \text{sign}(z_i)$ where $z_i = 1 - y_i(x_i^T \beta + \beta_0)$, $\forall i$. This is equivalent to updating
\( \mathbf{X} \mathbf{\beta} \), which can be done efficiently by noting that:

\[
\mathbf{X} \mathbf{\beta}^{new} = \sum_{g=1}^{G} \mathbf{X}_{\mathcal{I}_g} \mathbf{\beta}_g = \mathbf{X} \mathbf{\beta}^{old} + \mathbf{X}_{\mathcal{I}_g} \Delta \mathbf{\beta}_g,
\]

where \( \Delta \mathbf{\beta}_g = \mathbf{\beta}_g^{new} - \mathbf{\beta}_g^{old} \). Hence updating \( w^r \) also requires \( n|\mathcal{I}_g| \) operations. To conclude, Block Coordinate Descent for group L1-SVM is as expensive as Gradient Descent. The next points were found to be quite useful in practice:

1. Active set strategy: at every iteration, we record the active set of coefficients that are non zero. At the next descent iteration, we will only cycle through this set. We stop when the number of active set stabilizes from one iteration to the other. This method also holds for the Gradient Descent algorithm.

2. Decreasing sequence of parameters: We run Block Coordinate Descent for a decreasing sequence of coefficients \( (\lambda_{max}, \ldots, \lambda_k, \lambda_{k+1}, \ldots, \lambda) \), where the estimate for \( \lambda_{k+1} \) is initialized with the estimate obtained for \( \lambda_k \).

### 3.3.4 Scalability technique and Heuristics

**Case \( p \) large, \( n \) small**

When \( p \gg n \), we first use correlation screening to restrict the number of features – or groups – on which we run the first order methods. More precisely, for L1-SVM, we select the top \( 10n \) columns with highest absolute correlations with the output. For Group-SVM with Accelerated Gradient Descent or Block Coordinate Descent, we keep the \( n \) groups with the highest sums of absolute correlations. Then, in all cases, we run the above algorithm with a tolerance \( \eta = 10^{-2} \) and a maximum of \( T_{max} = 200 \) iterations. Column generation is initialized with the support of the estimate.

**Case \( n \) large \( p \) small: a subsampling heuristic**

We now consider L1-SVM with a large number of samples. First order methods are usually slow to converge for very large values of \( n \), but they are much faster for smaller number
of samples. Using an idea from distributed statistical learning [45] [50], we average local estimators $\beta_q, q = 1, \ldots, Q$, each defined as first order estimators on small randomly drawn samples of the design matrix $X$. The aggregated estimator $\bar{\beta} = \frac{1}{Q} \sum_{q=1}^{Q} \beta_q$ has a lower variance thus can get a good sense of the set of violated constraints $I$. We consequently propose to use the following heuristic:

**Subsampling heuristic for a large number of samples**

**Input:** $X$, $y$, a regularization coefficient $\lambda$, a smoothing coefficient $\tau$, a stopping criterion $\mu$, a maximum number of iterations $Q_{\text{max}}$, a subsample size $n_0$, the two stopping criterions $\eta$ and $T_{\text{max}}$ defined for the first order algorithm.

**Output:** An approximate set of violated constraints $I$.

1. While $\|\bar{\beta}_Q - \bar{\beta}_{Q-1}\|_2 > \mu$ and $Q < Q_{\text{max}}$ do:

   (a) Select a random subsample $I_{Q+1} \subset \{1, \ldots, n\}$ of size $n_0$ and define $\beta_{Q+1}$ as the first order estimator for the restricted design matrix $X_{I_{Q+1}}$ with restricted prediction vector $y_{I_{Q+1}}$, and parameters $\lambda, \tau, \eta, T_{\text{max}}$. For large values of $p$, we use the correlation screening described below.

   (b) Restrict the parameters $X_{I_{Q+1}}$ and $y_{I_{Q+1}}$ to the constraints violated by $\beta_{Q+1}$ and run the first order algorithm for a decreasing sequence of values of $\tau$. For large values of $p$ we also restrict the columns to the support of $\beta_{Q+1}$.

   (c) Update $\beta_{Q+1}$ as the final estimator obtained and define $\bar{\beta}_{Q+1} = \sum_{q=1}^{Q+1} \beta_q$.

2. Define $I \subset \{1, \ldots, n\}$ as the set of violated constraints for the estimator $\bar{\beta}_{Q+1}$.

When $n \gg p$, we run the algorithm with a block size $n_0 = 10p$ a tolerance $\mu = 10^{-1}$ and a maximum of $Q_{\text{max}} = n/n_0$ iterations. Further details are given in Section 3.4.1.

---

4In practice, we found continuation in $\tau$ on each sample to be quite useful.
5That is, we restrict the rows of $X$ to the index in $I_{Q+1}$.  

67
Case $n$ and $p$ large: Adapting the subsampling heuristic

When both $n$ and $p$ are large, the sets $I$ and $J$ of respectively violated constraints and columns can still be estimated using the subsampling heuristic. We define $n_0 = \max\{\sqrt{n}, \sqrt{p}\}$ and define at every iteration sub-matrices $X_{\mathcal{I}_Q}$ by selecting randomly $5n_0$ rows of the design matrix. The correlation screening then restricts the set of columns of each sub-matrix to the top $5n_0$ columns the more correlated with the restricted output $y_{\mathcal{I}_Q}$. Then we run Step 1.(b) of the algorithm to get our estimator $\beta_Q$.

The set $I$ is still defined as the set of violated constraints for the averaged estimator $\bar{\beta}_Q$. However, when the number of features is large, the averaged estimator $\bar{\beta}_Q$ usually has a denser support than the correct one: soft or hard-thresholding can be applied to remedy this problem [45]. If we have a rough idea of the size of the support we use hard thresholding to keep only the top features to form the set of columns $J$ of comparable size — for instance, a few hundreds of features.

3.4 Computation

We demonstrate the performance of our different methods on synthetic and real datasets for varying $n, p$ values. We use the LP solver of Gurobi version 6.5.2 with Python interface. All computations were carried out on a MacBook with processor 2.7 GHz 12-Core Intel Xeon E5, 64GB of RAM.

3.4.1 Computation results for L1-SVM

Synthetic datasets results for $p$ large, $n$ small

Data generation: We consider $n$ independent realizations of a $p$ dimensional multivariate normal centered distribution, with only $k_0$ dimensions being relevant for classification. Half of the samples are from the +1 class and have mean $\mu_+ = (1_{k_0}, 0_{p-k_0})$. The other half are from the -1 class and have mean $\mu_- = -\mu_+$. We consider a covariance matrix $\Sigma_{ij} = \rho$ if $i \neq j$ and 1 otherwise. The data of both $\pm 1$ classes respectively have the distribution: $\forall i, \ x_i^\pm \sim N(\mu_\pm, \Sigma)$. 

68
Results for computing the path: We first address the problem of computing the regularization path. We fix $n = 100$ and consider different values of $p$. For each training set, we use a geometric sequence of 20 decreasing values of $\lambda$ with common ratio $0.7$. We compare our Regularization Path algorithm presented in Section 3.2.3 for $j_0 = 10$, $\epsilon = 0.5$, $\epsilon = 0.1$, $\epsilon = 0.01$, and the LP solver used with two approaches. The first naive approach – referred as LP naive in Table 3.1 – solves the L1-SVM Problem (3.2) for every $\lambda$ with no initialization whereas the second one – LP warm start – uses the solution associated to a higher $\lambda$ to start solving the problem for a lower $\lambda$. This procedure speeds up drastically the training time.

For a method $m$, a coefficient $\lambda$, and a simulation $s_i \leq R$, let $f^m_\lambda(s_i)$ be the objective value of the method and $f^*_\lambda(s_i)$ the lowest objective values among all methods. We define the averaged relative accuracy (ARA) of the method $m$ over all the $R$ simulations run for this value of $\lambda$:

$$ARA(m, \lambda) = \frac{1}{R} \sum_{s_i=1}^{R} \frac{f^m_\lambda(s_i) - f^*_\lambda(s_i)}{f^*_\lambda(s_i)}.$$  

Table 3.1 shows the mean and standard deviation (in parenthesis) – over 10 repetitions – of the time for computing the path and of the averaged ARA for all values of $\lambda$.

Table 3.1: Times for computing the whole regularization path on synthetic datasets with the LP solver versus using column generation (CLG) with three tolerance levels. Here $n = 100$, $k_0 = 10$, $\rho = 0.1$. The naive LP solver training greatly benefits of using warm start. Both methods are outperformed by several orders by column generation with each of the 3 thresholds. A higher $\epsilon$ computes the path faster but gets a worse objective value for every $\lambda$, thus a worse ARA.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (s)</th>
<th>ARA (%)</th>
<th>Time (s)</th>
<th>ARA (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP naive</td>
<td>777.4(2.9)</td>
<td>0.0</td>
<td>&gt;2h</td>
<td>0.0</td>
</tr>
<tr>
<td>LP warm start</td>
<td>44.4 (0.6)</td>
<td>0.0</td>
<td>471.4 (10)</td>
<td>0.0</td>
</tr>
<tr>
<td>CLG, $\epsilon$=0.5</td>
<td>0.36(0.03)</td>
<td>2.8 (0.8)</td>
<td>2.89 (0.24)</td>
<td>3.2 (1.1)</td>
</tr>
<tr>
<td>CLG, $\epsilon$=0.1</td>
<td>0.53 (0.05)</td>
<td>0.1 (0.1)</td>
<td>3.98 (0.84)</td>
<td>0.1 (0.0)</td>
</tr>
<tr>
<td>CLG, $\epsilon$=0.01</td>
<td>0.61 (0.04)</td>
<td>0.0 (0.0)</td>
<td>4.74 (0.92)</td>
<td>0.0 (0.0)</td>
</tr>
</tbody>
</table>
Results for a fixed $\lambda$: We now fix $\lambda = 0.01\lambda_{\text{max}}$ and compare 5 approaches to solve the L1-SVM Problem (3.2). Method (a) is referred as RP CLG in the legend of the plots of Figure 3-1. It runs the Regularization Path with column generation algorithm from Section 3.2.3 with threshold $\epsilon = 0.01$, starting from the 10 features with highest reduced cost. Method (b) is referred as FO + CLG. It forms the set $J$ by running the first order algorithm presented in Section 3.3.3, with $\tau = 0.2$, on the $10n$ columns with highest absolute correlations with the output. column generation is then used for the same threshold $\epsilon = 0.01$. In addition, we report the time of column generation without the initialization step - referred as CLG wo FO. We run the default parameter for L1-SVM. Method (c) – Cor. screening – keeps the 50 best features with correlation screening before applying column generation once, whereas Method (d) – Random init. - starts with a random subset of 50 features. Finally Method (e) runs the LP solver: we run a first time the solver to optimality and get the exact objective value. We then run it a second time with a duality gap of $f^*_\epsilon - f^*$ where $f^*_\epsilon$ is the objective value of Method (b) and $f^*$ the optimal objective value obtained.

In the next Figure 3-1, we fix the parameters $n = 100$, $k_0 = 10$, $\rho = 0.1$ and the results are averaged over 10 simulations.
Training times for all methods

Figure 3-1: [Top: ] compares the 5 training times for methods (a)-(e) for solving L1-SVM for a fixed $\lambda$ and $n = 100$. [Bottom: Left panel] zooms on the best methods (a) and (b). [Bottom: Right panel] presents the associated Average Relative Accuracies (in %). The LP solver (e) is outperformed by the 4 opponents. Method (b) proves to be extremely fast: for $p = 10^6$ it solves the L1-SVM Problem in 10 seconds which is 400 times faster than the LP solver. The training time is equally split between first order and column generation algorithms.

Real datasets

Results for a fixed $\lambda$: We use 4 real gene expression microarray datasets with small number of samples and large number of features. We compare our best Method (b) with the LP solver (e). All methods are run as explained in Section 3.4.1. For each dataset we merge the default train and test sets, standardize the columns and split them into two new train and test sets of same size and same ratio between the two classes. Table 3.2 presents the averaged training times over 10 repetitions for $\lambda = 0.01\lambda_{\text{max}}$. We use hard-thresholding on the first order Method estimators to keep only the top 200 highest coefficients.

---

Table 3.2: Training times of Methods (b) and (e) for $\lambda = 0.01\lambda_{\text{max}}$ on 5 real datasets. Our method outperforms the LP solver at every run.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$n$</th>
<th>$p$</th>
<th>Time (s)</th>
<th>ARA (%)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leukemia</td>
<td>35</td>
<td>7,129</td>
<td>0.59(0.02)</td>
<td>8.8 $10^{-2}$</td>
<td>20.94(1.08)</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>90</td>
<td>12,533</td>
<td>1.45(0.02)</td>
<td>6.3 $10^{-5}$</td>
<td>87.92(2.75)</td>
</tr>
<tr>
<td>Ovarian</td>
<td>253</td>
<td>15,155</td>
<td>2.59(0.01)</td>
<td>1.2 $10^{-2}$</td>
<td>146.52(1.49)</td>
</tr>
<tr>
<td>Radsens</td>
<td>58</td>
<td>12,625</td>
<td>0.43(0.01)</td>
<td>1.6 $10^{-1}$</td>
<td>29.20(0.13)</td>
</tr>
</tbody>
</table>

**Computational results for $n$ large, $p$ small**

**Results for a fixed $\lambda$:** The next Figure 3-2 assesses the good performance of our algorithm when $n \gg p$. We fix $p = 100$ and consider an increasing sequence of values of $n$. We still fix $\lambda = 0.01\lambda_{\text{max}}$ and compare the LP solver (e) presented in Section 3.4.1 with our new Method (f) referred as SFO + CNG in the next plots. It runs constraint generation initialized with the subsampling first order heuristic with a subsample size $n_0 = 10p$, a tolerance $\mu = 10^{-1}$, and a maximum of $Q_{\text{max}} = n/n_0$ iterations. On each subsample we consider a decreasing sequence of 20 values of $\tau$ with common ratio 0.7. We then run constraint generation for a tolerance $\epsilon = 10^{-2}$. In addition, we report the time of the constraint generation algorithm without the initialization – referred as CNG wo SFO.

In the next Figure 3-2, we fix the parameters $p = 100$, $k_0 = 10$, $\rho = 0.1$ and average the results over 10 simulations.

Figure 3-2: [Left] Training times for methods (e) and (f) for $p = 100$. [Right] Associated ARA(%). LP solver (e) is outperformed by our Method (f). Both methods reach similar objective values. The training time of our method is essentially explained by the constraint generation algorithm.
Computational results for $n$ large, $p$ large

Results for a fixed $\lambda$ on synthetic datasets: Sections 3.4.1 and 3.4.1 have respectively showed the good performance of our column generation and constraint generation algorithms initialized with our first order heuristics. The next Figure 3-3 assesses the gain in performance when combining these methods for large values of $n$ and $p$. We run the column-and-constraint generation algorithm for $\epsilon = 0.01$ presented in Section 3.2.4: the sets $I$ and $J$ are initialized by the subsampling first order heuristic as presented in Section 3.3.4 – using the same values of parameters than in Section 3.4.1. For each simulation, we use $\lambda = 0.001\lambda_{\text{max}}$ and propose to compare Methods (a) and (b) – the best methods in Section 3.4.1 – with our new Method (g), referred as SFO + CL-CNG in the next Figure 3-3. In addition, CL-CNG wo SFO presents the time of the column-and-constraint generation algorithm without the initialization.

Figure 3-3 compares the averaged training times and ARA over 10 repetitions for $n = 5000$, $k_0 = 10$, $\rho = 0.1$ and an increasing sequence of columns.

Figure 3-3: [Left] Training times for Methods (a), (b) and (g) for $n = 5000$. [Right] Associated ARA (%). The Regularization Path algorithm cannot handle large values of $n$ and $p$. Our hybrid method (g) deals with a small subset of constraints: it significantly improves over column generation, while returning an estimator with similar objective value.

Results for a fixed $\lambda$ on real datasets: Finally, we assess the quality of our hybrid column-and-constraint generation method (g) on two large real datasets \footnote{The datasets are from the UCI repository webpage https://archive.ics.uci.edu/ml/datasets.html?format=&task=cla&att=&area=&numAtt=&numIns=&type=&sort=nameUp&view=table.}, when compared
to the LP solver (e). All methods are run as explained in Section 3.4.1, except that we fix $\mu = 0.5$. Because the real datasets are very sparse, we adapt our methods to deal with sparsity. We consider $\lambda = 0.05\lambda_{\text{max}}$ and initialize the set $\mathcal{I}$ by only keeping the 200 highest coefficients of the first order estimator. We average our results across 5 repetitions.

Table 3.3: Training times of our best Method (g) and the LP solver (e) on large sparse real datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>n</th>
<th>p</th>
<th>SFO+CL-CNG Time (s)</th>
<th>CL-CNG wo SFO Time (s)</th>
<th>LP solver (e) Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rcvl</td>
<td>20,242</td>
<td>47,237</td>
<td>188.67(2.88)</td>
<td>75.41(2.91)</td>
<td>&gt;3h</td>
</tr>
<tr>
<td>real-sim</td>
<td>72,309</td>
<td>20,959</td>
<td>712.76(14.03)</td>
<td>444.64(9.63)</td>
<td>&gt;3h</td>
</tr>
</tbody>
</table>

### 3.4.2 Computation results for Group-SVM

In addition, we show the performance of the column-generation algorithm presented in Section 3.2.5 for the Group-SVM Problem (3.3).

**Data generation:** We consider $G$ groups of same size $p_G$ so that $G \times p_G = p$ – with only the first $k_0$ groups being relevant for classification. The groups are independent and two features within a same group have a correlation $\rho$. We consider $n$ independent realizations where half of the samples are from the +1 class and then have a mean $\mu_+ = (1_{p_G}, \ldots, 1_{p_G}, 0_{p_G}, \ldots, 0_{p_G})$ and a block-diagonal covariance matrix $\Sigma$.

**Competitors:** We fix $\lambda = 0.1\lambda_{\text{max}}$ – $\lambda_{\text{max}}$ is defined in Section 3.2.5 – and compare the 4 following approaches for Group-SVM. Method (a) – referred as RP CLG in the legend of the plots of Figure 3-4 – runs the Regularization Path with column generation algorithm presented in Section 3.2.5 – starting from the top 10 groups the more correlated with the output. We use a threshold $\epsilon = 0.01$. Method (b) – referred as FO + CLG – runs the first order algorithm presented in Section 3.3.3, with $\tau = 0.2$, on the $n$ groups with highest absolute correlations before running column generation for the same threshold $\epsilon = 0.01$. Method (c) is referred as FO BCD + CLG: it initializes the column generation algorithm
with the Block-Coordinate Descent method presented in Section 3.3.3 on the same $n$ groups. We also report the times of running of the column generation algorithms without the first order algorithm for Accelerated Gradient Descent – **CLG wo FO** – and Block Coordinate Descent – **CLG wo FO BCD**. Finally Method (d) runs the **LP solver**.

In the next Figure 3-4, each group of features is of size $p_G = 10$. We use the parameters $n = 100$, $k_0 = 10$, $\rho = 0.1$ and average the results over 10 simulations. We do not report the ARA as all methods give the same objective value.

Figure 3-4: [Left panel] compares the 4 training times for methods (a)-(d). [Right panel] zooms on the 3 best methods (a)-(c). The LP solver cannot compete with the 3 other methods. Accelerated Gradient Descent (b) and Block Coordinate Descent (c) both outperform the Regularization Path algorithm (a). Block Coordinate Descent appears as the overall winner: the gain in time compared to Gradient Descent comes from a faster initialization.
Chapter 4

Hierarchical Modeling and Shrinkage for User Session Length Prediction in Media Streaming

4.1 Introduction

On-line streaming services such as Pandora, Netflix, and Youtube constantly seek to increase their share of on-line attention by keeping their users as engaged as possible with their own services [43]. A well known challenge is how to measure the users’ engagement, and what are the key components that create an engaging streaming service. One important engagement metric is the amount of time spent by users within the service. When users access streaming services, they usually watch videos, movies, on-line TV or listen to music, and after a while they leave the service. We refer to the user interaction from the moment they start the service to the moment they leave as a user session, and the time spent during one session as user session length [72].

In this paper, we aim to predict the user session length using real world datasets from two music streaming platforms, i.e. predicting, at the beginning of the session, the amount of time they will spend listening music. Understanding and modeling the factors that can affect the session length is of great use for various downstream tasks. In fact it allows
recommender systems to tune the explore vs exploit parameters for each user. In addition, having an accurate estimate of the users' session lengths allows the streaming service to adjust ad pacing per user. Ads can be rescheduled in a way to keep the revenue target (i.e. total number of ads presented) as well as improve user experience.

Predicting the length of user sessions is very challenging as the authors reported in [72]. First, user sessions can end for many different external reasons that have nothing to do with quality of the streaming services, such as moving to subway, reaching home, and most of these contextual co-variates are not easily accessible because of technological or privacy reasons [72]. Second, a certain amount of users are casual users in the sense that they only use the streaming services a few times per month, which makes the problem of estimating session lengths for those users hard. Furthermore, a lack of predictive covariates makes it harder to correctly predict what is going to be the next session length for a user.

A first approach toward session length analysis and prediction [72] is based on a Boosting algorithm. Most of the other related research were focused on modeling the time spent after clicking a search result [9, 41] or advertisement [4]. The best models are based on survival analysis, mainly because data are censored in these applications. In fact, after clicking of a search result or an ad, users can either turn back to the search page or can abandon the final page, instead of turning back to the main search page. This is not the case of session length data that is not censored\(^1\) – this opens the door to using a suite of methods based on regression modeling, shrinkage and relevant generalizations. Furthermore, in the case of web search or ad click, the user enters a query or click and checks the results, therefore the intersection between search or ad title and the landing page give highly predictive features [41]. In the case of a streaming service, the user interaction can be very low, but still they can have very long sessions ("lean-back" behavior) [72].

In this paper, we propose a novel framework inspired by hierarchical Bayesian modeling and shrinkage principles that allows us to express session lengths in terms of user-specific latent variables. These variables are estimated via a joint learning framework which is rather broad in scope – we use Bayes, Empirical Bayes and MAP estimation techniques–particular

\(^1\)Observations are called censored when the information about their survival time is incomplete.
choices are based on computational tractability considerations. Informally speaking, our framework learns models by borrowing strengths across all users, and also makes use of rich covariate information. A salient aspect of our framework is its modularity – it includes state-of-the-art models as special cases, and naturally allows for a hierarchy of flexible generalizations. Hence, it allows the practitioner to glean insights about the problem, by assessing incremental gains in predictive accuracy associated with different generalizations and the incorporation of covariate-information. We present tailored algorithmic approaches based on modern large convex optimization techniques to address the computational challenges. We summarize some of our key findings in this paper:

- We outperform a baseline estimator by a margin of 12% to 18% and the state-of-the-art in session length prediction [72] up to 3.5% in terms of Mean Absolute Error measured in seconds on 2 different real world datasets.

- We show that some of the proposed prediction model can be as accurate as state of the art, but between 7 to 14 times faster in training time, and at least 30 times faster in prediction time, reaching around 1ms.

- We provide a modular framework specifically for this problem that allow more flexible generalizations, and that includes state of art proposed model [72] as a special case.

### 4.2 Key idea

The main focus of this paper is the prediction of user session length at the moment of login into a streaming service. For this purpose, we exploit the users past interaction with streaming services, previous sessions lengths, build a set of features (as we will describe in Section 4.4.1), and we set a learning framework for prediction purpose. We proceed by discovering that if you take the log of users session length then the distribution looks almost normal (Section 4.4.1). As we will see, this will help us in having a well-grounded and tractable inferential framework. Since we want a prediction framework that is performing well for very active users (with many sessions) and less-frequent users (with only a few
sessions per month), we propose a formal framework inspired by hierarchical Bayesian shrinkage ideas that allows us to model the session-length of a user in terms of user-specific latent variables. Fundamental principles of Bayesian shrinkage and estimation encourage users to borrow strengths across each other, including (but not limited to) covariate information. This (i) ameliorates the high variance (and hence low predictive accuracy) of user-specific maximum likelihood estimates for less-frequent users; and (ii) leads to an overall boost in prediction accuracy for more frequent users as well. Bayesian decision theory and empirical Bayes methodology \cite{19} provides a formal justification of our framework. The notion of shrinkage that we undertake here is quite broad—it applies to cases with or without covariate information; and the previous model \cite{72} on this particular problem is shown to be a special case of our framework. For flexible models and/or priors when Bayes estimators become computationally demanding, we recommend MAP estimation for computational efficiency. All the estimators proposed in this paper can be cast as solutions to convex optimization problems. Towards this end, we resort to techniques in modern large scale convex optimization to achieve computational scalability and efficiency.

4.3 Mathematical Framework

We formally develop the inferential framework to address the session length prediction problem. We denote the total number of users by $N$. For every $i \in [N] := \{1, \ldots, N\}$, let $n_i$ be the number of past sessions of user $i$, and $y_{ij}$ be the time spent by user $i$ in the $j$th session. We will work with log of session length $y_{ij} = \log(y_{ij})$ as our response as this gives a better approximation to the Gaussian distribution (See Table 4.1)\footnote{We also tried a family of transformations of the form $\log(\cdot)^r$ for $r > 0$; and observed that $r \approx 1$ led to the closest approximation to the Gaussian in terms of the Kolmogorov-Smirnov goodness of fit measure.}. We note that similar (variance stabilizing) transformations are often used in the empirical Bayes literature \cite{19} so that one can take recourse to the rich literature in Gaussian estimation theory. We denote by $y_i = (y_{ij})_{j \in [n_i]} \in \mathbb{R}^{n_i}$ a vector of log-session-lengths of user $i$; $N_0 = \sum_{i=1}^{N} n_i$ the total number of sessions across all users; and $y = (y_i)_{i \in [N]} \in \mathbb{R}^{N_0}$ is a vector of log-session-lengths of all users across all sessions. In addition, covariate information per-session is available...
(See Section 4.4.1). For a given user, some of these features are fixed across sessions (age, gender...) and others depend upon the session (network, device...). Let $x_{ij} \in \mathbb{R}^d$ denote covariates corresponding to the $j$th session of user $i$. $X \in \mathbb{R}^{N_0 \times d}$ denotes a matrix with rows $x_{ij}$ that are stacked on top of each other. The $k$th row of $X$ corresponds to $k$th entry of the response vector $y$. The columns of $X$ were all mean-centered and standardized to have unit $\ell_2$ norm.

A natural point estimate of the time spent by the $i$th user may be taken to be the average time spent by the user in past sessions – however, we see that this does not lead to good predictions due to the high variance associated with users with few sessions. This behavior is not surprising and occurs even in the simple Gaussian sequence model as explained in Section 4.3.1.

4.3.1 Review of Bayes, Empirical Bayes and MAP

This section provides a brief review of Bayes, Empirical Bayes (EB) and Maximum A posteriori (MAP) estimation in the context of the Gaussian sequence model [19]. The exposition in this section lays the foundation for generalizations to more flexible structural models that we present subsequently.

The Bayes Estimator: We consider a latent Gaussian vector $\mu_{n \times 1} = (\mu_1, \ldots, \mu_n)$ where, $\mu_i \overset{iid}{\sim} \mathcal{N}(0, A^2)$; that gives rise to an observable Gaussian vector $z = (z_1, \ldots, z_n)$ such that $z_i | \mu_i \sim \mathcal{N}(\mu_i, 1)$ for all $i$. Note that the posterior distribution of $\mu | z$ is given by a $n$-dimensional multivariate Gaussian with mean $B^2 z$ and covariance $B^2 I$, i.e., $\mu | z \sim \mathcal{N}_n(B^2 z, B^2 I)$ where, $B^2 = \frac{A^2}{1 + A^2}$. Recall that the Bayes estimator is the mean of the posterior $\mu | z$ and given by:

$$\hat{\mu}^{\text{Bayes}} = \mathbb{E}(\mu | z) = \left(1 - 1/(1 + A^2)\right) z. \quad (4.1)$$

We remind the reader that the Bayes estimator shrinks each observation towards the mean 0 of the prior distribution – this is to be contrasted with the usual maximum likelihood (ML) estimator, $\hat{\mu}^{\text{ML}} = z$ that does not shrink $\mu_i$’s. The Bayes estimator has smaller risk
than the ML estimator (Theorem 1).

**Empirical Bayes (EB) estimator:** The Bayes estimator depends upon the unknown hyper-parameter $A^2$; which needs to be estimated from data. The “Empirical Bayes” (EB) framework [19] achieves this goal by using a data-driven plug-in estimator for $A^2$ in (4.1) – this leads to an EB estimator for $\mu$.

The basic EB framework obtains an unbiased estimator for $A^2$ based on the marginal distribution of $z \sim \mathcal{N}_n(0, (1 + A^2) I)$. Using standard properties of Gamma and inverse-Gamma distributions, it follows that $(n - 2)/S$ (where, $S = \sum_{i=1}^{n} z_i^2$) is an unbiased estimator of $\frac{1}{A^2 + 1}$.

This leads to an EB estimate $\hat{\mu}^{EB} = (1 - \frac{n-2}{S}) z$, which has smaller risk (Theorem 1) than the ML estimator.

**Theorem 2** [19] For $n \geq 3$, the EB estimator $\hat{\mu}^{EB}$ has smaller risk (defined as $R(\hat{\mu}) := \mathbb{E}(\|\mu - \hat{\mu}\|_2^2)$) than the ML estimator $\hat{\mu}^{ML}$, i.e., $R(\hat{\mu}^{EB}) < R(\hat{\mu}^{ML})$ for any $\mu$. The risks of the EB and Bayes estimators are comparable, with a relative ratio: $\frac{R(\hat{\mu}^{EB}) - R(\hat{\mu}^{Bayes})}{R(\hat{\mu}^{Bayes})} = \frac{2}{nA^2}$.

We make the following remarks: (i) Theorem 1 states that the price to pay for not knowing $A$ is rather small, and as $n$ becomes large the Bayes and EB estimators are similar. (ii) Instead of taking an unbiased estimator for $1/(A^2 + 1)$ as above, one can also take a consistent estimator which might be easier to obtain for more general models (see Section 4.3.2). For more general models, a good plug-in estimate for $A$ may be obtained based on validation tuning. The framework above provides important guidance regarding a range of good choices of $A$ thereby reducing the computational cost associated with the search for tuning parameters.

**MAP estimation:** As an alternative to the Bayes/EB estimator, we can also consider the MAP estimator, a mode of the posterior likelihood $\mu|z$. Here the MAP estimate ($\hat{\mu}^{MAP}$) coincides with the Bayes estimator. The MAP and Bayes estimators are not the same in general. For flexible priors/models, computing Bayes estimators can become computationally challenging and one may need to resort to (intractable) high dimensional MCMC computations. In these situations (see Section 4.3.3), the MAP estimator may be easier to compute from a practical viewpoint. For all models used in this paper, we observe that MAP computation can be tractably performed via convex optimization.
4.3.2 Model 1: Modeling user effects

We present a hierarchical shrinkage framework for predicting user-specific session lengths, generalizing the framework in Section 4.3.1.

Suppose that the log-session lengths of the $i$th user are normally distributed with mean $\mu_i$; and these latent variables $\{\mu_i\}_{i=1}^N$ are generated from a centered Gaussian distribution. The (random) $\mu_i$ is the $i$th user effect. This leads to the following hierarchical model:

$$y_{ij} | \mu_i \overset{\text{iid}}{\sim} N(\mu_i, \sigma_1^2), \ i \in [N], \ j \in [n_i]; \ \mu_i \overset{\text{iid}}{\sim} N(0, \sigma_0^2)$$

(4.2)

generalizing the model in Section 4.3.1 to the case with multiple replications per user. The posterior distribution is given by:

$$\mu_i | y_i \sim N\left(\frac{\bar{y}_i}{1 + \lambda/n_i}, \frac{\sigma_1^2}{\lambda + n_i}\right) \ \forall i,$$

where, $\lambda = \sigma_1^2/\sigma_0^2$ and $\bar{y}_i = \sum_{j=1}^{n_i} y_{ij}/n_i$ is the mean of the vector $y_i$. The Bayes estimator of $\mu$ is given by the posterior mean $\hat{\mu}_i^{\text{Bayes}} = \frac{\bar{y}_i}{1 + \lambda/n_i}$. Here, the MAP estimator of $\mu$ coincides with the Bayes estimator as well. We note that the Bayes/MAP estimators in this example bear similarities with the model in Section 4.3.1 – we shrink the observed mean of each user towards the global mean of the prior distribution: this lowers the variance of the estimator at the cost of (marginally) increasing the bias. The amount of shrinkage depends upon the number of sessions of the $i$th user via the factor $1 + \lambda/n_i$. In particular, the shrinkage effect will be larger for users with a small number of sessions.

**Estimating the hyper-parameters:** The estimators above depend upon hyper-parameters $\sigma_0, \sigma_1$ via $\lambda = \sigma_1^2/\sigma_0^2$, which is unknown and needs to be estimated from data. In the spirit of an EB estimator we obtain a plug-in estimator for $\lambda$. To this end we use the marginal distribution of $y_i$, which follows $N_{n_i}(0, \Sigma_{n_i})$ where, $\Sigma_{n_i}$ has diagonal entries equal to $\sigma_0^2 + \sigma_1^2$ and off-diagonal entries equal to $\sigma_0^2$. Consequently, $y_i y_i^T$ is an unbiased estimator for the covariance matrix $\Sigma_{n_i}$. In particular, if $T_i = \|y_i\|_2^2$ then the estimators

$$\hat{\sigma}_0^2(i) = \frac{(n_i \bar{y}_i)^2 - T_i}{n_i(n_i - 1)} \ \text{and} \ \hat{\sigma}_0^2(i) + \hat{\sigma}_1^2(i) = \frac{T_i}{n_i}$$

(4.3)
are unbiased estimators of $\sigma^2_0$ and $\sigma^2_0 + \sigma^2_2$ (respectively). To see this, note that $\hat{\sigma}^2_0(i)$ is obtained by taking the average of all the $n_i(n_i - 1)$ off-diagonal entries of the matrix $y_iy_i^T$. Similarly, $\hat{\sigma}^2_0(i) + \hat{\sigma}^2_1(i)$ corresponds to the average of the diagonal entries. Estimators in (4.3) are based solely on observations from the $i$th user; and can have high variance if $n_i$ is small (which is the case for less heavy users). Hence, we aggregate the estimators across all $N$ users to obtain improved estimators of $\sigma^2_0, \sigma^2_2$ given by:

\[
\hat{\sigma}^2_0 = \frac{1}{N} \sum_{i=1}^{N} \hat{\sigma}^2_0(i) = \frac{1}{N} \sum_{i=1}^{N} \frac{(1_n^T y_i)^2 - T_i}{n_i(n_i - 1)} \\
\hat{\sigma}^2_1 = \frac{1}{N} \sum_{i=1}^{N} \hat{\sigma}^2_1(i) = \frac{1}{N} \sum_{i=1}^{N} \frac{T_i}{n_i} - \hat{\sigma}^2_0.
\] (4.4)

Using laws of large numbers, one can verify that $\hat{\sigma}^2_0$ (and $\hat{\sigma}^2_1$) are consistent estimators for $\sigma^2_0$ (and $\sigma^2_2$). Interestingly, this holds under an asymptotic regime where, $N \to \infty$ but $\min_i n_i$ remains bounded – this regime is relevant for our problem since there are many users with few/moderate number of sessions. We emphasize that even if $N$ is large but the $n_i$’s are small, shrinkage plays an important role and leads to estimators with smaller risk than the usual maximum likelihood estimator $\mu_i^{ML} = \bar{y}_i$ for $i \in [N]$. The plug-in estimators suggested above lead to consistent estimators for the Bayes and EB estimators. This framework provides guidance regarding the choice of the tuning parameters in practice (and reduces the search-space of tuning parameters).

### 4.3.3 Model 2: Modeling with covariates

We describe a generalization of Model 1 that incorporates user and device-specific covariates (See Section 4.4.1 for details). Our hierarchical model is now given by:

\[
y_{ij} | \beta, \mu_i \overset{\text{iid}}{\sim} \mathcal{N} (x_{ij}^T \beta + \mu_i, \sigma^2_2) , i \in [N], j \in [n_i] \\
\text{where, } \beta \sim \mathcal{N}_d(0, \sigma^2_2 I), \quad \mu_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2_0), i \in [N].
\] (4.5)

The above represents a generative model with latent variables $(\mu, \beta)$ – where, $\beta \in \mathbb{R}^d$ denotes a vector of regression coefficients corresponding to the covariates $X$; and $\mu_i$ explains
the residual user-specific effect of user \( i \). Both the latent variables are normally distributed with mean zero; and given these parameters, \( y_{ij} \)'s are normally distributed with mean \( x_{ij}^T \beta + \mu_i \). This model is more structured and also more flexible than Model 1 in that the \( i \)th user effect has two components: a global regression-based response \( x_{ij}^T \beta \) (this depends upon both the user and the session); and a residual component \( \mu_i \). We now derive the EB and MAP estimators.

Let us define \( \hat{\mu} = \frac{\sigma^2}{\sigma_0} \mu \) and the latent vector \( \gamma = (\beta, \hat{\mu}) \). Model (4.5) can be reformulated as:

\[
y_{ij} \mid \gamma \sim \mathcal{N}(x_{ij}^T \gamma, \sigma_i^2), \forall i, j; \quad \gamma \sim \mathcal{N}_{N+d}(0, \sigma_2^2 I),
\]

where, \( x_{ij} \in \mathbb{R}^{d+N} \) is such that its first \( d \) entries correspond to \( x_{ij} \), its \( (d+i) \)th entry is \( \sigma_0 / \sigma_2 \); and all remaining entries are 0. If \( \hat{X}_{N_0 \times d+N} \) be the matrix obtained by row concatenation of the \( \hat{x}_{ij} \)'s; then the posterior distribution of \( \gamma \mid y \) is given by

\[
\gamma \mid y \sim \mathcal{N}_{N+d}\left(H^{-1}\hat{X}^T y, \sigma_2^2 H^{-1}\right),
\]

where, the matrix \( H = \hat{X}^T \hat{X} + \alpha I \) and the regularization parameter \( \alpha = \sigma_1^2 / \sigma_2^2 \). The Bayes estimate of \( \gamma \) is given as:

\[
\hat{\gamma}^{\text{Bayes}} = \mathbb{E}(\gamma \mid y) = \left(\hat{X}^T \hat{X} + \alpha I_{d+N}\right)^{-1} \hat{X}^T y \in \mathbb{R}^{d+N}.
\]

\( \hat{\beta}^{\text{Bayes}} \) and \( \hat{\mu}^{\text{Bayes}} \) can be derived from the components of \( \hat{\gamma}^{\text{Bayes}} = \left(\hat{\beta}^{\text{Bayes}}, \sigma_2^2 \hat{\mu}^{\text{Bayes}}\right) \). In this model, the MAP estimator coincides with the Bayes estimator, and can be computed as \( (\hat{\beta}^{\text{MAP}}, \hat{\mu}^{\text{MAP}}) \in \text{argmin } L_2(\beta, \mu) \), where, \( L_2(\beta, \mu) \) is the convex function:

\[
L_2(\beta, \mu) := \sum_{i=1}^{N} \left\{ \sum_{j=1}^{n_i} (y_{ij} - x_{ij}^T \beta - \mu_i)^2 + \lambda \mu_i^2 \right\} + \alpha ||\beta||_2^2,
\]

and \( \lambda = \sigma_1^2 / \sigma_0^2, \alpha = \sigma_1^2 / \sigma_2^2 \) are hyper-parameters. In Section 4.3.4, we propose Algorithm 1 to minimize Problem (4.6).

An empirical Bayes estimator of \( (\beta, \mu) \) can be computed by using data-driven estimators for the hyper-parameters. We can obtain consistent estimators of the hyper-parameters
following the derivation in Section 4.3.2. Since this derivation\(^3\) is quite tedious, we do not report it here. In practice, we recommend tuning \((\lambda, \alpha)\) on a validation set (where, \(\lambda\) is taken to be in the neighborhood of the values suggested by Section 4.3.2 pertaining to Model 1). As we discuss in Section 4.3.5, this does not add significantly to the overall computational cost, as our algorithm effectively uses warm-start continuation \([25]\) across different tuning parameter choices.

We now move beyond the Gaussian prior setup considered so far and consider a Laplace prior on \(\beta\). In this case, and the models we consider subsequently, Bayes estimators are difficult to compute due to high-dimensional integration that require MCMC computations. With computational tractability in mind, we will resort to MAP estimation for these models.

**Laplace prior on \(\beta\):** Motivated by \(\ell_1\) regularization techniques \([67]\) popularly used in sparse modeling, we propose a Laplace prior on \(\beta\) – the corresponding MAP estimators lead to sparse, interpretable models \([25, 67]\). Here, computing the Bayes estimator becomes challenging and requires MCMC computation. However, the MAP estimator is particularly appealing from a statistical and computational viewpoint; and given by \((\hat{\beta}^{MAP}, \hat{\mu}^{MAP}) \in \text{argmin} \mathcal{L}_1(\beta, \mu)\), where,

\[
\mathcal{L}_1(\beta, \mu) := \sum_{i=1}^{N} \left\{ \sum_{j=1}^{n_i} (y_{ij} - x_{ij}^T \beta - \mu_i)^2 + \lambda \mu_i^2 \right\} + \alpha \|\beta\|_1 \tag{4.7}
\]

is a convex function with hyper-parameters \(\lambda, \alpha\). The tuning parameters are chosen based on a validation set. Section 4.3.4 presents an algorithmic framework based on first order convex optimization methods \([58, 76]\) for optimizing Problem (4.7) – our proposed algorithm leads to significant computational gains compared to off-the-shelf implementations.

**Nonparametric modeling with covariates**

The framework presented above is quite modular—it allows for flexible generalizations, allowing a practitioner to experiment with several modeling ramifications, and understand

---

\(^3\)Note that for Model (4.5) the marginal distribution of \(y\) is a multivariate Gaussian with mean zero and covariance matrix \(\Sigma\), which is a function of \(\{\sigma_i^2\}\) and \(XX^T\). Following Section 4.3.2, we have \(E(yy^T) = \Sigma\). We can then derive consistent estimators of \(\{\sigma_i^2\}\) based on functionals of \(yy^T\) and entries of \(XX^T\).
their incremental value (prediction accuracy vis-a-vis computation time) in the context of the particular application/dataset.

Recall that the basic model put forth by Model 2 is \( y_{ij} | \theta_{ij} \overset{iid}{\sim} \mathcal{N}(\theta_{ij}, \sigma^2_i) \) where, \( \theta_{ij} = \mathbf{x}_{ij}^T \beta + \mu_i \). We propose to generalize this linear ‘link’ by incorporating flexible nonparametric models for the covariates, as follows:

\[
y_{ij} | \theta_{ij} \overset{iid}{\sim} \mathcal{N}(\theta_{ij}, \sigma^2_i), \quad \text{and} \quad \theta_{ij} = f(\mathbf{x}_{ij}) + \mu_i, \quad (4.8)
\]

where, \( f(\cdot) \) is a flexible nonparametric function of the covariates. For example, we can train \( f \) via Gradient Boosting Trees (GBT) \([25]\) as our non-parametric model\(^4\). Trees introduce nonlinearity and higher order interactions among features and can fit complex models. By adjusting tuning parameters like learning rate, maximal tree-depth, number of boosting iterations, etc, they control the bias-variance trade-off and hence the generalization ability of a model. Given a continuous response \( \mathbf{z}_{n \times 1} \) and covariates \( \mathbf{U}_{n \times d} \), GBT creates an additive function of the form \( f(\mathbf{U}) = \sum_k \eta h_k(\mathbf{U}) \) where, \( h_k(\cdot) \)'s are trees of a certain depth and \( \eta \) is the learning rate – the components \( \{h_k\} \) are learned incrementally via steepest descent on the least squares loss \( \| \mathbf{z} - f(\mathbf{U}) \|_2^2 \) with possible early stopping. This imparts regularization and improves prediction accuracy.

**Summarizing the general framework:** In summary, our framework assumes that we have access to an oracle that solves the following optimization problem

\[
\hat{f} \in \arg\min_f \left\{ \| \mathbf{z} - f(\mathbf{U}) \|_2^2 + \Omega(f) \right\}, \quad (4.9)
\]

with a regularizer \( \Omega(\cdot) \) that restricts the family \( f \). Problem (4.9) encompasses the different models that we have discussed thus far: e.g., model (4.7) (here, \( f(\mathbf{U}) = \mathbf{U} \beta \) and \( \Omega(\beta) = \alpha \| \beta \|_1 \)), model (4.6) (here, \( f(\mathbf{U}) = \mathbf{U} \beta \) and \( \Omega(\beta) = \alpha \| \beta \|_2^2 \)); and GBT.

For flexible nonparametric models, a MAP estimator can be obtained by minimizing the negative log-likelihood of the posterior distribution jointly w.r.t \( \mu \) and \( f \). This entails minimizing the negative log-likelihood of the posterior distribution – this is given by the

\(^4\)We also experimented with classical Classification and Regression trees as well as random forests, but the best predictive models were obtained via GBT.
function $\mathcal{L}(f, \mu)$ (up to constants) as follows:

$$
\mathcal{L}(f, \mu) := \sum_{i=1}^{N} \left\{ \sum_{j=1}^{n_i} (y_{ij} - f(x_{ij}) - \mu_i)^2 + \lambda \mu_i^2 \right\} + \Omega(f),
$$

where, $\lambda = \sigma^2 / \sigma_0^2$. Section 4.3.4 presents an algorithmic framework for minimizing (4.10) to obtain estimates $\hat{f}, \hat{\mu}$.

We note that for the class of the models in Section 4.3.3, it is not clear how to tractably construct and compute Bayes/EB estimators—we thus focus on MAP estimation; and note that the associated tasks can be cast as tractable convex optimization problems.

**Some Special Cases**

As we have noted before, an important contribution of this paper is to propose a general modeling framework — so that a practitioner can glean insights from data analyzing the incremental gains available from different modeling components. To this end, we note that if all the residual user effects are set to zero (i.e, $\mu_i = 0$ for all $i$), then we can use covariates alone to model the user effects. In the case of model (4.6) with $\mu = 0$ this is referred to as Ridge in Section 4.4.3. Furthermore, if we learn $f(\cdot)$ via boosting (with $\mu = 0$) then we recover the model proposed in [72] (denoted as SIGIR2017 in Section 4.4) as a special case. Predictive performances of these models are presented in Section 4.4.

**4.3.4 Computation via Convex Optimization**

All the estimation problems alluded to above can be cast as convex optimization problems; for which we resort to modern computational methods [58, 76]. To compute the estimators mentioned in Section 4.3.3, we need to minimize Problem (4.10). To this end, we use a block-coordinate descent scheme [76]: at iteration $t$, it minimizes (4.10) w.r.t. $f$, followed by a minimization w.r.t the latent vector $\mu$. The algorithm is summarized below.

**Algorithm 1: Block-Coordinate-Descent for MAP estimation**

**Input:** $X$, $y$, tuning parameters, tolerance $\epsilon$; initialization $\mu^0, f^0$.

**Output:** An estimate $(\hat{f}, \hat{\mu})$, minimizing Problem (4.10)
(1) Repeat Steps 2 to 4 until \(|L_t - L_{t-1}|/L_{t-1} > \epsilon\) for \(t \geq 1\).

(2) Let \(\hat{f}^{(t)} \in \text{arg min}_f \mathcal{L} \left( f, \hat{\mu}^{(t-1)} \right)\) be a solution of the optimization problem (4.10) with \(\mu\) held fixed at \(\hat{\mu}^{(t-1)}\) — this is equivalent to solving (4.9) with \(z = z^{(t)}\) where,
\[
z^{(t)}_{ij} = y_{ij} - \hat{\mu}^{(t-1)}_{i} \forall i, j.
\]

(3) Update the residuals \(r^{(t)}_{ij} = y_{ij} - \hat{f}^{(t)}(x_{ij}), \forall i, j\). Estimate user-specific effects via: \(\hat{\mu}^{(t)} \in \text{arg min}_\mu \mathcal{L} \left( \hat{f}^{(t)}, \mu \right)\) (with \(f\) set to \(\hat{f}^{(t)}\)). This is a closed-form update: \(\hat{\mu}^{(t)}_i = \frac{1}{n_i + \lambda} \sum_{t=1}^{n_i} r^{(t)}_{ij}\) for all \(i\).

(4) Set the value of \(L_{t+1} = \mathcal{L} \left( \hat{f}^{(t)}, \hat{\mu}^{(t)} \right)\).

Algorithm 1 applies to a fixed choice of the hyper-parameters. We need to consider a sequence of hyper-parameters to obtain the best model based on the minimization of prediction error (see Section 4.4) on a validation set. In the case of models (4.6) and (4.7) estimates of \(\beta\) can be computed over a grid of parameters by using warm-starts across different tuning parameters — to this end, the EB estimators provide a good ballpark estimate of relevant tuning parameters. Section 4.3.5 describes specialized algorithms that are found to speed up the computations pertaining to models (4.6) and (4.7) when compared to off-the-shelf implementations of these algorithms. We note that GBT does not benefit from warm-start continuation across hyper-parameters. For Algorithm 1, we use a stopping criterion of \(\epsilon = 0.01\) (Step 1) in the experiments.

### 4.3.5 Computational Considerations

We consider certain algorithmic enhancements for Algorithm 1 that lead to important savings when the number of sessions become large (of the order of millions). We focus on the critical Step 2 of Algorithm 1 when \(f(U) = U \beta\) and \(\Omega(\beta)\) corresponds to the ridge or \(\ell_1\) regularization — this leads to a problem of the form:

\[
\min_{\beta} \left\{ \|z^{(t)} - X\beta\|^2_2 + \Omega(\beta) \right\}.
\]

where \(\Omega(\beta) \in \{\alpha \|\beta\|_2^2, \alpha \|\beta\|_1\}\). Indeed, in these instances, we found out that the default implementation of Python’s scikit-learn package [60] was prohibitively slow for our
purpose, and hence careful attention to algorithmic details seemed necessary (details below).
We derived new algorithms for (4.11) with an eye towards caching numerical linear algebraic
factorizations, exploiting warm-starts, etc; as we describe below.

\(\ell_2\) regression subproblem

When \(\Omega(\beta) = \alpha \|\beta\|_2^2\), the ridge estimator has an analytical expression:

\[
\hat{\beta}^R = (X^T X + \alpha I_d)^{-1} X^T z(t), \tag{4.12}
\]

which needs to be computed for several tuning parameters, and iterations. To reduce the
computational cost, we obtain an equivalent expression for \(\hat{\beta}^R\) via the eigendecomposition of
\(X^T X \in \mathbb{R}^{d \times d}\) given by, \(X^T X = V \Gamma V^T\), with \(\Gamma\) being a diagonal matrix with eigenvalues
\(\{\gamma_i\}_1^d\) – this has a cost of \(O(d^3)\) in addition to the \(O(N_0 d^2)\) cost of computing \(X^T X\) (and
they can both be done once, off-line). This leads to \(\hat{\beta}^R = V \Gamma X^T z(t)\) which can be computed
with cost \(O(N_0 d + d^2)\). In our experiments (Section 4.4) \(d\) is small, which leads to a cost
that is linear in \(N_0\). The predicted values \(X \hat{\beta}^R\) can be computed with an additional cost of
\(O(N_0 d)\). Note that computing estimator (4.12) for different values of the tuning parameter
\(\alpha\) does not require additional eigendecompositions – this is critical in making the overall
algorithm efficient, especially when training across multiple values of the hyper-parameter.

\(\ell_1\) regression subproblem

When \(\Omega(\beta) = \alpha \|\beta\|_1\), Problem (4.11) becomes equivalent to a Lasso estimator – we
emphasize that scikit-learn’s implementation of Lasso became rather expensive for our
purposes since it could not effectively exploit warm-starts and cached matrix computations.
This motivated us to consider our own implementation, based on proximal gradient descent [5,
58]. To this end, since \(N_0 \gg d\), we precomputed\(^5\) \(Q := X^T X\) and considered a \(d\)-dimensional
quadratic optimization problem of the form:

\[
\min_{\beta} \ F(\beta) := \beta^T Q \beta - 2 (X^T z^t, \beta) + \alpha \|\beta\|_1, \tag{4.13}
\]

\(^5\)Note that this computation is also required for the ridge regression model.
where, the smooth part of $F(\beta)$ has $C$-Lipschitz-continuous gradient — that is, it satisfies 
\[ \| \nabla F(\gamma) - \nabla F(\beta) \|_2 \leq C \| \gamma - \beta \|_2, \forall \gamma, \beta \] for $C = 2 \max_i \gamma_i$ (recall, $\gamma_i$’s are eigenvalues of $Q$). A proximal gradient algorithm for (4.13) performs the following updates:

\[ \beta_{k+1} = \arg \min_{\beta} \left\{ \frac{1}{2} \| \beta - \left( \beta_k - \frac{1}{L} \nabla F(\beta_k) \right) \|_2^2 + \frac{\alpha}{L} \| \beta \|_1 \right\} \]  
(4.14)

till convergence. Note that $\beta_{k+1}$ can be computed via soft-thresholding, i.e., 
\[ \beta_{k+1} = S_{\alpha/L}(\beta_k - \frac{1}{L} \nabla F(\beta_k)) \] where, for a vector $a \in \mathbb{R}^d$ the $i$th coordinate of the soft-thresholding operator $S_\tau(a)$ is given by $\text{sgn}(a_i) \max\{|a_i| - \tau, 0\}$. Note that the objective function $F(\beta)$ is strongly convex\footnote{Note that $F(\beta) - \rho/2\| \beta \|_2^2$ is convex for $\rho = 2 \min_i \gamma_i$, i.e., the minimum eigenvalue of $Q$ — this means that $F(\beta)$ is strongly convex with strong convexity parameter $\rho$.}; and hence sequence $\beta_k$ converges to an $\delta$-suboptimal solution to Problem (4.13) in $O(\log(1/\delta))$ iterations [58] — i.e., it enjoys a linear convergence rate. Every iteration of (4.14) has a cost of $O(d^2)$ (arising from the computation of $\nabla F(\beta)$ and the soft-thresholding operation). In addition, computing $X^T z^{(t)}$ costs $O(N_0d)$ (this is computed once at Step 2 of Algorithm 1). Problem (4.13) needs to be computed for several tuning parameters and iterations (of Algorithm 1) — this does not add much to the overall run-time as the proximal gradient algorithm can be effectively warm-started — this is found to speed-up convergence in practice.

**GBT subproblem**

When the optimization in Step 2 involves performing GBT, the runtimes increase substantially (See Section 4.4.5). Unlike the models in Sections 4.3.5, 4.3.5; GBT is computationally intensive and needs to be done for every iteration of Algorithm 1. Unlike the optimization based algorithms for $\ell_1/\ell_2$ regression as described above, GBT does not naturally accommodate warm-starts across iterations, and/or tuning parameters.
4.4 Experiments

Here, we evaluate the effectiveness of our prediction model with respect to several baselines and state of the art session length prediction solutions. We proceed by describing our datasets, our evaluation framework, the comparisons, and then present the results.

4.4.1 Datasets

We used two different real world datasets of users listening to music, namely, Pandora and lastfm. Pandora is a sample of user interaction data from Pandora Internet Radio, the same dataset used in [72] and lastfm is a publicly available dataset from last.fm [15]. We defined the user sessions as periods of continuous listening, interrupted if the user stop or pause the music for more than 30 minutes [72]. For Pandora we gathered data from a small subset of Pandora users for a period of 3 months (February-May 2016) resulting in 3,976,561 sessions. lastfm public dataset was gathered between 2004 to 2009 and it contains 911,770 sessions for 1,000 different users. Table 4.1 reports some statistics about the user session length in the two datasets. For the log values, we first take the log transform of the raw data, as mentioned in the modeling part, and then normalize. An interesting finding is that mean and median are quite different for the raw data in both datasets. In fact, as reported in [72], Weibull distributions give a better fit to user session lengths, while after a log-transformation, the data are very close to normal distributions, which is what our modeling framework requires for tractable inference.

Feature Engineering: For all the sessions in Pandora we create two kinds of features, namely, user-based and contextual as in [72]. As user-based features we consider "gender (the gender of the user), age (the age of the user), subscription_status (whether the user is ad-supported)", these features are fixed for a given user. As contextual features we consider "device (the device used for the session), network (the type of network used for the session), absence_time (time elapsed since the user's previous session), previous_duration (the duration of the user's previous session)". We refine this set of features to include additional contextual features to [72], this is mainly to lower the variance of

\footnote{Due to confidentiality we can not report the number of users for this dataset.}
Table 4.1: Summary statistics of normalized user session lengths in the two datasets. The upper half are on the normalized raw session lengths. The bottom half are on the normalized log session lengths.

<table>
<thead>
<tr>
<th>Stats</th>
<th>Pandora</th>
<th>lastfm</th>
</tr>
</thead>
<tbody>
<tr>
<td>25th quantile(raw)</td>
<td>0.008</td>
<td>0.009</td>
</tr>
<tr>
<td>median(raw)</td>
<td>0.021</td>
<td>0.029</td>
</tr>
<tr>
<td>mean(raw)</td>
<td>0.044</td>
<td>0.060</td>
</tr>
<tr>
<td>75th quantile(raw)</td>
<td>0.049</td>
<td>0.069</td>
</tr>
<tr>
<td>25th quantile(log)</td>
<td>0.57</td>
<td>0.59</td>
</tr>
<tr>
<td>median(log)</td>
<td>0.66</td>
<td>0.69</td>
</tr>
<tr>
<td>mean(log)</td>
<td>0.65</td>
<td>0.62</td>
</tr>
<tr>
<td>75th quantile(log)</td>
<td>0.74</td>
<td>0.76</td>
</tr>
</tbody>
</table>

the past sessions, and introduce non-linearity. We consider as additional features "average_user_duration (average user session length in training set), log_average_user_duration (logarithm of average_user_duration), log_absence_time (logarithm of absence_time), log_previous_duration (logarithm of previous_duration), session_time (whether the user session started in morning or afternoon)". For lastfm dataset the "age, subscription_status, device, network are missed.

4.4.2 Evaluation

We sort our dataset by chronological order, use the first 80% for the training set, 10% for the validation set, and the rest 10% for the test set. Additionally we require each user in the validation or test set to appear at least once in the training set. The final datasets for Pandora and lastfm have respectively in total 3,949,137, and 713,089 sessions. For the models that need parameter tuning, we first train the models on the training set for each set of the parameters. Then we use the validation set to pick the best set of parameters. Finally, we use that set of parameters for training on the combined set of training and validation, and predict on the test set. For the evaluation metric of our session length prediction model, we use Normalized Mean Absolute Error measured in seconds, averaged over all the test sessions and normalized by the Baseline model which by our definition has MAE = 1. MAE is a good metric due the possibility of important errors resulting from very
large session length. More formally, let $|S_{test}|$ be the number of sessions in the test set and $\bar{y}_{ij}$ be the time spent by user $i$ on his $j$th session, where $j$ is a test session of user $i$, and $\tilde{y}_{ij}^p$ be the predicted value then:

$$\text{MAE} = \frac{1}{|S_{test}|} \sum_{(i,j)\in S_{test}} |\tilde{y}_{ij}^p - \bar{y}_{ij}|$$

### 4.4.3 Comparisons

We compare our model with several baselines and state of the art methods. In particular we have considered the following:

**Baseline.** The baseline model is the per-user mean session length, i.e., we compute for each user the mean session length in the training set and use the value as a prediction value for all the test sessions of the same user.

**SIGIR2017.** This is method in [72] that is using a modified version of boosting algorithm. Our tuned models have a number of trees in $\{10, 15, 50, 100\}$, with depth $\{6, 10\}$ and use a learning rate in $\{0.1, 0.05\}$.

**Baseline** can be interpreted as a natural baseline and **SIGIR2017** is the state of the art in this particular application. In the context of our proposed framework presented in Section 4.3, we consider the following models:

**Model1.** This is the model described in Section 4.3.2, where we don't use any covariates. All the parameters of this model were derived using parameter estimation described in 4.3.2.

**Ridge.** This is the Ridge estimator defined in Section 4.3.3, i.e., we perform a ridge regression only on covariates. We take 50 values of the tuning parameter (as per Section 4.3.3).

**Model2-L2.** This is the Bayes (which is also the MAP) estimator for the model presented in Section 4.3.3 with an $\ell^2$ regularization on $\beta$. We run Algorithm 1 (Section 4.3.5) on a 2D grid of tuning parameters $(\alpha, \lambda)$ with 100 different values (Section 4.3.3).
Model2-L1. This is MAP estimator model presented in Section 4.3.3 with the use of an $\ell_1$ regularization on $\beta$. We use Algorithm 1 (Section 4.3.5) for computation, and take 100 values of the 2D grid of tuning parameters (Section 4.3.3).

Model2-GBT. This model uses Gradient Boosting Trees (GBT) to compute the MAP estimator (4.10) via Algorithm 1 (Section 4.3.5). We use the same sequence of tuning parameters as in SIGIR2017 and a sequence of 10 $\lambda$ values in $[1, 10]$.

Similar to the Baseline model, Model1 does not consider covariates. Model1 however, performs shrinkage on the user-specific effects, and thus any gain in predictive accuracy (as evidenced in Table 4.2) is due to shrinkage. Ridge considers only covariates and does not include the residual user-specific effects. The rest of the models use features regarding the context and user, and additional user-specific effects. As described in Section 4.5 we do not have censored data, and we are interested in making point predictions on user session-lengths, therefore survival analysis models are not suitable for our scenario.

### 4.4.4 Effectiveness

We report the results regarding the effectiveness of our model. Table 4.2 reports the results of the Normalized MAE on all the models in Section 4.4.3. By borrowing strength across users, Model1 improves over the Baseline even without using any covariate-information. SIGIR2017, the model presented in [72] and a special case of the framework presented here (Section 4.3.3) is benefiting from the usage of the covariates, and it is clearly better than Model1. Model2-GBT reaches a significantly lower MAE than SIGIR2017. This observation shows the importance of the user effect in our hierarchical modeling framework. Model2-L2 is performing quite well in all the datasets. This model only considers 2 hyper-parameters, the choice of which are guided by the EB framework in Section 4.3.3. We did not observe any gain in MAE by using an $\ell_1$ penalization, though the models were sparse (in $\beta$) when compared to $\ell_2$ regularization. Model2-GBT has the lowest MAE for all the datasets. Our hierarchical modeling framework greatly benefits from nonparametric function estimation on the covariates. Overall, our Model2-GBT is the best in terms of prediction in both datasets.
Table 4.2: Normalized MAE on test set for our model compared to the baselines and state of the art.

<table>
<thead>
<tr>
<th>Models</th>
<th>MAE Pandora</th>
<th>MAE lastfm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline- no covariates</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Model1- no covariates</td>
<td>0.936</td>
<td>0.830</td>
</tr>
<tr>
<td>SIGIR2017</td>
<td>0.910</td>
<td>0.826</td>
</tr>
<tr>
<td>Ridge</td>
<td>0.921</td>
<td>0.828</td>
</tr>
<tr>
<td>Model2-L1</td>
<td>0.911</td>
<td>0.819</td>
</tr>
<tr>
<td>Model2-L2</td>
<td>0.911</td>
<td>0.819</td>
</tr>
<tr>
<td>Model2-GBT</td>
<td>0.878</td>
<td>0.812</td>
</tr>
</tbody>
</table>

Table 4.3: Feature importance for Pandora dataset considering highest absolute value for Ridge and Model2-L2 (we centered and normalized all the features first).

<table>
<thead>
<tr>
<th></th>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ridge</td>
<td>log_average_user_duration</td>
<td>0.516</td>
</tr>
<tr>
<td></td>
<td>absence_time</td>
<td>0.430</td>
</tr>
<tr>
<td></td>
<td>average_user_duration</td>
<td>0.064</td>
</tr>
<tr>
<td></td>
<td>device=smartphone</td>
<td>0.045</td>
</tr>
<tr>
<td></td>
<td>device=Web</td>
<td>0.042</td>
</tr>
<tr>
<td>Model2-L2</td>
<td>device=smartphone</td>
<td>0.092</td>
</tr>
<tr>
<td></td>
<td>device=Web</td>
<td>0.091</td>
</tr>
<tr>
<td></td>
<td>log_average_user_duration</td>
<td>0.088</td>
</tr>
<tr>
<td></td>
<td>absence_time</td>
<td>0.067</td>
</tr>
<tr>
<td></td>
<td>average_user_duration</td>
<td>0.064</td>
</tr>
</tbody>
</table>

Model2-L1 or Model2-L2 are close to SIGIR2017 for Pandora and better for lastfm and as we see in Table 4.5 they are actually much faster in training time.

**Feature Importance.** By centering and normalizing the columns of the matrix of covariates $X$, the absolute values of the coefficients of $\hat{\beta}$ for Ridge or Model2-L1 suggest the relative importances of the features. Table 4.3 reports the highest absolute values of the coefficients for the Ridge estimator and for Model2-L2 estimator for Pandora dataset. Device and time-related features appear as the most relevant features. The two most important features for Ridge correspond to logarithm of the average user session length in training set and the absence time since last session. In addition, considering user effect on Model2-L2 lowers the magnitude of time-related features, even though they still appear in the top ones.
Table 4.4: Normalized MAE, restricted to people in the first decile, the first two deciles or the last 8th deciles of the training set. (Hierarchical) shrinkage has a more prominent effect over the Baseline model (with MAE=1) for users with fewer sessions.

<table>
<thead>
<tr>
<th>Model</th>
<th>$&lt; q_{10}$</th>
<th>$&lt; q_{20}$</th>
<th>$&gt; q_{20}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pandora</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model1</td>
<td>0.860</td>
<td>0.876</td>
<td>0.938</td>
</tr>
<tr>
<td>Ridge</td>
<td>0.790</td>
<td>0.809</td>
<td>0.925</td>
</tr>
<tr>
<td>SIGIR2017</td>
<td>0.780</td>
<td>0.806</td>
<td>0.904</td>
</tr>
<tr>
<td>Model2-GBT</td>
<td>0.778</td>
<td>0.804</td>
<td>0.881</td>
</tr>
<tr>
<td><strong>lastfm</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model1</td>
<td>0.610</td>
<td>0.798</td>
<td>0.834</td>
</tr>
<tr>
<td>Ridge</td>
<td>0.606</td>
<td>0.789</td>
<td>0.833</td>
</tr>
<tr>
<td>SIGIR2017</td>
<td>0.622</td>
<td>0.779</td>
<td>0.829</td>
</tr>
<tr>
<td>Model2-GBT</td>
<td>0.606</td>
<td>0.775</td>
<td>0.817</td>
</tr>
</tbody>
</table>

**Performance Breakdown by Sessions per User.** We perform a break down of users into three different types by quantiles of number of sessions per user in the training set. Table 4.4 reports the normalized MAE for these three groups. The results show a monotonic decrease in terms of gain in performance (with respect to baseline) for all the models with number of sessions per user. This serves as a validation of an important message presented through our modeling framework – shrinkage is more critical for less active users when compared to the Baseline. In fact, even for the more frequent users, we observe that shrinkage helps when compared to the Baseline (but the gains are less pronounced). Our best model Model2-GBT is outperforming the state of art SIGIR2017 for all three cutoffs chosen. Model2-GBT performs better than SIGIR2017 by modeling the user-specific effects – this gain seems to be most prominent for heavier users for both Pandora and lastfm datasets–this is not surprising since Model2-GBT is a more flexible variation of SIGIR2017 and benefits from more data from heavier users.

### 4.4.5 Efficiency

We further investigate the efficiency of our model by looking at the total training time and average time of one prediction for our best models and state of art solution. We
implemented everything using PYTHON. We run the experiments in a MacBook Pro with 2.7GHz Intel Core i5 with 8 GB of RAM. For each model we fixed a set of parameters to tune. We then run each model sequentially, each time with different fixed parameters. Table 4.5 reports the average running time, number of runs done for tuning purposes and the average time of one prediction for our best models and SIGIR2017. The most important thing to note here is that Model2-L2 or Model2-L1 can be trained at least 7 times faster than any tree based method such as SIGIR2017, and they have an effectiveness that is very similar to SIGIR2017. This is mainly because the Model2-L2 and Model2-L1 computations benefit from warm-starts (see Section 4.3.4), so the increase in number of tuning parameters do not increase the run time as in the case of SIGIR2017 and Model2-GBT that are based on Gradient Boosting Trees. They are also at least 30 to 40 times faster in prediction time (for the Pandora dataset), and therefore they can be used for real-time prediction with time constraints. While Model2-GBT is slow in training time, it shows better performance in terms of effectiveness than other methods. However, we note that these models are meant for off-line training in the context of the application herein–while it is important to have algorithms that are fast, they are not of critical importance as in tasks where real-time learning is of foremost importance.

4.5 Related work

Session length is an important metric serving as a proxy for user engagement. Therefore the solutions and evaluation are tailored to similar duration based engagement metric such as dwell-time prediction.

Dwell-time. Liu et al. in [47] presented one of the first studies on dwell-time for web search. Kim et al. in [41] proposed a dwell-time based user satisfaction prediction model in the web search context. Lalmas et al. in [44] proposed new way to improve ad ranking. Barbieri et al. in [4] propose to use survival forest [36] using landing page and user feature in the ad context to estimate the dwell-time and incorporate it in the ad ranking system as a quality score. Vasiloudis et al. in [72] presented recently a first session length prediction.

---

8Our code is available at: https://github.com/pvahabi/MIT_session_length_prediction
Table 4.5: Training time, and average time of prediction for our best models and the state of the art. We report for each model the effectiveness (in terms of MAE), total training time (in seconds) to train across “#run”-many tuning parameters, and average prediction time (in seconds).

<table>
<thead>
<tr>
<th>Models</th>
<th>MAE</th>
<th>Train. Time</th>
<th>#run</th>
<th>Avg. Pred. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pandora</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIGIR2017</td>
<td>0.910</td>
<td>11,703</td>
<td>16</td>
<td>7.70</td>
</tr>
<tr>
<td>Mode2-L1</td>
<td>0.911</td>
<td>835</td>
<td>50</td>
<td>0.24</td>
</tr>
<tr>
<td>Mode2-L2</td>
<td>0.911</td>
<td>840</td>
<td>50</td>
<td>0.24</td>
</tr>
<tr>
<td>Mode2-GBT</td>
<td>0.878</td>
<td>52,360</td>
<td>16</td>
<td>9.71</td>
</tr>
<tr>
<td>lastfm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIGIR2017</td>
<td>0.826</td>
<td>354</td>
<td>16</td>
<td>0.53</td>
</tr>
<tr>
<td>Mode2-L1</td>
<td>0.819</td>
<td>50</td>
<td>50</td>
<td>0.001</td>
</tr>
<tr>
<td>Mode2-L2</td>
<td>0.819</td>
<td>50</td>
<td>50</td>
<td>0.001</td>
</tr>
<tr>
<td>Mode2-GBT</td>
<td>0.812</td>
<td>820</td>
<td>16</td>
<td>0.64</td>
</tr>
</tbody>
</table>

model in the music streaming context using survival analysis and gradient boosting [16]. In that work, the authors show that in the case of media streaming services the probability of a session to end evolves differently for different users. In particular 44% of the users exhibit negative-aging length distributions, i.e. sessions that become less likely to end as they grow longer. Although not directly comparable, we report that this percentage is going to 98.5% for dwell-time on a web page after a search, i.e., after clicking on a search result the more you stay the more you are likely to stay on the clicked page. Finally, Jing et al. in [38] presented a neural network based model combined with survival analysis for recommendation purpose and absence time prediction at the same time. In our work we compared against the most recent related work done by Vasiloudis et al. [72]. It is worth mentioning that while dwell-time can benefit from survival analysis, because a user can click on a search engine result and never turn back, in our case we don’t have censored data, therefore our is a regression problem.

**Empirical Bayes and MAP estimation:** The statistical models proposed in this paper are inspired by Empirical Bayes methods that are well-known in statistics community, dating back to [63, 20]. In theory, when the model is true, these estimators are known to lead to estimators with better prediction accuracy when compared to (unregularized)
maximum likelihood estimators. Empirical Bayes estimators offer an appealing trade-off between frequentist and Bayesian modeling [19]. We also consider more flexible models wherein MAP (maximum a posteriori) estimation becomes a pragmatic choice from a computational viewpoint. To our knowledge, this is the first time such a methodology is used in the context of user session length prediction.

4.6 Conclusions and future work

In this paper, we presented a new hierarchical modeling framework, inspired by core Bayesian modeling principles to predict the amount of time people will spent in a streaming service, and in particular listening to streaming music. We also propose modern convex optimization algorithms for enhanced computational efficiency and tractable inference. Our family of flexible models is meant to provide a practitioner insights regarding the incremental gains in predictive accuracy with enhanced modeling components. We focused on predicting the amount of time a user might spend on a platform, at the beginning of the user session.

In our experimental section, we have shown that our method is performing better than the state of the art in this context. Furthermore, we have shown that our model is better for heavy users as well as for users with few sessions. Due to the flexibility of our models, we can achieve lower prediction error for users with many sessions. Our results show that some of our models can be as good as Boosted-tree-based state of the art models, but they can be trained up to 14 times faster, and they are at least 30 times faster in prediction time.

So far we have always investigated a scenario where the model is learned off-line, and it tries to predict the user session length with only few static and off-line extractable features. In the future we aim to extend our model to an on-line version. Furthermore we want to investigate the session length prediction utility within advertising or recommender systems context.
Bibliography


Appendices
Appendix A

Appendices for Chapter 2

A.1 Computational details

A.1.1 Proof of Proposition 2

(a) Note that, from (2.9), for any $\beta$ satisfying $\|\beta\|_0 \leq k$:

$$F(\beta) = Q_L(\beta, \beta) + \lambda \|\beta\|_q$$

$$\geq \inf_{\|\eta\|_0 \leq k} (Q_L(\eta, \beta) + \lambda \|\eta\|_q)$$

$$= \inf_{\|\eta\|_0 \leq k} \left( \frac{L}{2} \|\eta - \beta\|_2^2 + \langle \nabla f(\beta), \eta - \beta \rangle + f(\beta) + \lambda \|\eta\|_q \right)$$

$$= \inf_{\|\eta\|_0 \leq k} \left( \frac{L}{2} \|\eta - \left( \beta - \frac{1}{L} \nabla f(\beta) \right) \|_2^2 - \frac{1}{2L} \|\nabla f(\beta)\|_2^2 + f(\beta) + \lambda \|\eta\|_q \right)$$

(A.1)

$$= \left( \frac{L}{2} \|\hat{\eta} - \left( \beta - \frac{1}{L} \nabla f(\beta) \right) \|_2^2 - \frac{1}{2L} \|\nabla f(\beta)\|_2^2 + f(\beta) \right) + \lambda \|\hat{\eta}\|_q.$$  

(A.2)

Note that in (A.2) above we use the notation $\hat{\eta}$ to denote a minimizer of (A.1). We now follow the proof in Proposition 6 [6] to arrive at:

$$F(\beta) \geq \frac{L - L_0}{2} \|\hat{\eta} - \beta\|_2^2 + F(\hat{\eta}),$$  

(A.3)
and, in particular, using $\eta = \beta^{(m+1)}$ and $\beta = \beta^{(m)}$ and $L \geq L_0$, we see that the sequence $F(\beta^{(m)})$ is decreasing. Because $F(\beta) \geq 0$, we observe that the sequence $F(\beta^{(m)})$ converges to some $F^* \geq 0$.

(b) Summing inequalities (A.3) for $1 \leq m \leq M$, we obtain

$$
\sum_{m=1}^{M} \left( F(\beta^{(m)}) - F(\beta^{(m+1)}) \right) \geq \frac{L - L_0}{2} \sum_{m=1}^{M} \|\beta^{(m+1)} - \beta^{(m)}\|_2^2,
$$

leading to

$$
F(\beta^{(1)}) - F(\beta^{(M+1)}) \geq \frac{M(L - L_0)}{2} \min_{m=1,...,M} \|\beta^{(m+1)} - \beta^{(m)}\|_2^2.
$$

Because the decreasing sequence $F(\beta^{(m)})$ converges to $F(\beta^*) = F^*$, say, we arrive at the conclusion in Part (b).

A.1.2 Stronger formulations: adding implied inequalities

We consider a structured version of Problem (2.5) with additional implied inequalities (cuts) for improved lower bounds:

$$
\text{minimize } \frac{u}{2} + \lambda v \\
\text{s.t. } \|y - X\beta\|_2^2 \leq u \tag{A.5a} \\
\|\beta\|_q \leq v \tag{A.5b} \\
- M_j z_j \leq \beta_j \leq M_j z_j, j \in [p] \\
z_j \in \{0, 1\}, j \in [p] \\
\sum_j z_j = k \\
- M_i \leq \beta_i \leq M_i, i \in [p] \tag{A.5c} \\
- \tilde{M}_i \leq \langle x_i, \beta \rangle \leq \tilde{M}_i, i \in [n] \tag{A.5d} \\
\|\beta\|_1 \leq M_{t_1}, \tag{A.5e}
$$

112
where (a) $M_i, i \in [p]$ denote bounds on $\beta_i$'s via constraint (A.5c); (b) $-\tilde{M}_i, \tilde{M}_i^+$ denote bounds on the predicted values $\langle x_i, \beta \rangle$ for $i \in [n]$ via constraint (A.5d); (c) $M_{\ell_1}$, in constraint (A.5e), denotes an upper bound on the $\ell_1$-norm of the regression coefficients $\|\beta\|_1$.

The additional cuts in Problem (A.5) help in the progress of the MIO solver – the implied inequalities rule out several fractional solutions, thereby helping in obtaining superior lower bounds within a fixed computational budget. The caveat, however, is that the resulting formulation has additional variables – hence more work needs to be done within every node of the branch-and-bound tree. Section A.1.3 presents ways to compute these bounds – Section A.1.3 describes ways to compute them via convex optimization – these are bounds implied by an optimal solution to Problem (2.3). Section A.1.3 describes ways to compute these bounds based on good heuristic solutions.

### A.1.3 Computing problem specific parameters

#### Computing parameters via convex optimization

Formulation (2.4) involves a BigM value $M$ – tighter formulations can be obtained by using variable dependent BigM values for the $\beta_i$'s:

$$-M_i z_i \leq \beta_i \leq M_i z_i, \quad i \in [p].$$

In addition, implied constraints (or bounds) on $\langle x_i, \beta \rangle$'s can also be added:

$$-\tilde{M}_i \leq \langle x_i, \beta \rangle \leq \tilde{M}_i, \quad i \in [n].$$

We discuss how to compute these from data using convex optimization. Note that, because $\beta$ is $k$-sparse, we have $|\langle x_i, \beta \rangle| \leq M \|x_i\|_{k,1}$, where for a vector $a \in \mathbb{R}^p$ the quantity $\|a\|_{k,1}$ denotes the $\ell_1$-norm of the $k$-largest (in absolute value) entries of $a$. We can set $\tilde{M}_i \leq M \|x_i\|_{k,1}$. Note also that $\|\beta\|_1 \leq M k := M_{\ell_1}$. We now upper bound each coefficient
\(\beta_i\) by solving the quadratic optimization problems:

\[
\mathcal{M}_i^+ = \max \quad \beta_i
\]

s.t. \(\frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_q \leq \text{UB}\)

\[
\|\beta\|_\infty \leq \mathcal{M}
\]

\[
\|\beta\|_1 \leq \mathcal{M}_{\ell_1}
\]

\[-\tilde{\mathcal{M}}_i^- \leq \langle x_i, \beta \rangle \leq \tilde{\mathcal{M}}_i^+, i \in [n]\]

\[
\mathcal{M}_i^- = \max \quad -\beta_i
\]

s.t. \(\frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_q \leq \text{UB}\)

\[
\|\beta\|_\infty \leq \mathcal{M}
\]

\[
\|\beta\|_1 \leq \mathcal{M}_{\ell_1}
\]

\[-\tilde{\mathcal{M}}_i^- \leq \langle x_i, \beta \rangle \leq \tilde{\mathcal{M}}_i^+, i \in [n]\]

where UB is an upper bound to Problem (2.3) obtained via Algorithm 1, for example. Upon solving Problem (A.6), we set \(\mathcal{M}_i = \max\{\mathcal{M}_i^+, \mathcal{M}_i^-\}\) for all \(i \in [p]\). Consequently, we can update the bounds \(\mathcal{M} = \|\mathcal{M}_i\|_\infty, \tilde{\mathcal{M}}_i\) and \(\mathcal{M}_{\ell_1}\) – such bound tightening methods have been proposed in [49] in the context of the Discrete Dantzig Selector problem.

Similarly, we can also obtain bounds on \(\langle x_j, \beta \rangle\) by solving the following pair of optimization problems for all \(j \in [n]\).

\[
\tilde{\mathcal{M}}_j^+ = \max \quad \langle x_j, \beta \rangle
\]

s.t. \(\frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_q \leq \text{UB}\)

\[-\mathcal{M}_i^- \leq \beta_i \leq \mathcal{M}_i^+, i \in [p]\]

\[
\|\beta\|_1 \leq \mathcal{M}_{\ell_1}
\]

\[-\tilde{\mathcal{M}}_i^- \leq \langle x_i, \beta \rangle \leq \tilde{\mathcal{M}}_i^+, i \in [n]\]
\[
\tilde{M}_j^* = \max_{\mathcal{M}_j} -\langle x_j, \beta \rangle \\
\text{s.t.} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_q \leq UB \\
-\mathcal{M}_i^- \leq \beta_i \leq \mathcal{M}_i^+, i \in [p] \\
\|\beta\|_1 \leq \mathcal{M}_\ell, \\
-\tilde{M}_i^- \leq \langle x_i, \beta \rangle \leq \tilde{M}_i^+, i \in [n].
\]

Upon solving Problem (A.8), we can set \(\tilde{M}_i = \max\{|\tilde{M}_j^+|, |\tilde{M}_j^-|\}\). The bounds thus obtained can be used to tighten the bounds used in Problems (A.6) and (A.8). New bounds on \(\mathcal{M}_i\) and \(\tilde{M}_i\) can be obtained by solving the new problems with the updated bounds.

**Remark 2** Problems (A.6), (A.8) drop the cardinality constraint on \(\beta\) — hence the derived bounds need not be tight, i.e., \(\mathcal{M}_i > |\tilde{\beta}_i(\lambda; k)|\), where \(\tilde{\beta}(\lambda; k)\) denotes an optimal solution to Problem (2.3).

**Computing parameters via Algorithm 1**

We note that the BigM values \(\mathcal{M}_i, i \in [p]\) can also be based on the solutions obtained from the heuristic algorithms. For example, we can set \(\mathcal{M}_i = \tau\|\tilde{\beta}(\lambda; k)\|_\infty\) for all \(i \in [p]\) for some multiplier \(\tau \in \{1.5, 2\}\), for example. Similarly, the bounds \(\tilde{M}_i\)’s can be set to \(\tau|\langle x_i, \tilde{\beta}(\lambda; k)\rangle|\) for all \(i \in [n]\).

Such bounds are usually tighter and are obtained as a simple by-product of Algorithm 1.

**A.2 Proofs of the results in Section 2.3**

**A.2.1 Proof of Theorem 1**

The proof follows the one for Corollary 6.1 in [14], with only minor modifications. We start with the first prediction error bound. Note that \(\beta^*\) is feasible for the \(\ell_1\)-regularized best
subset selection optimization problem for $k \geq k^*$. Consequently,

$$\|y - X\widehat{\beta}_1\|^2 + \lambda_1\|\widehat{\beta}_1\|_1 \leq \|y - X\beta^*\|^2 + \lambda\|\beta^*\|_1,$$

which implies

$$\|X\widehat{\beta}_1 - X\beta^*\|^2 + \lambda_1\|\widehat{\beta}_1\|_1 \leq 2\|X^T\epsilon\|_\infty\|\widehat{\beta}_1 - \beta^*\|_1 + \lambda_1\|\beta^*\|_1$$

$$\leq 2\|X^T\epsilon\|_\infty\|\widehat{\beta}_1\|_1 + (\lambda_1 + 2\|X^T\epsilon\|_\infty)\|\beta^*\|_1.$$

Note that $X^T\epsilon$ is a mean zero Gaussian vector, such that the variance of each component is $\sigma^2$. Consequently, it follows from well-known maximal inequalities for Gaussian variables that the bound $2\|X^T\epsilon\|_\infty \leq \lambda_1$ holds with probability at least $1 - 2p^{-\alpha}$ [see Lemma 6.2 in 14, for example]. Thus, on this high-probability set we have

$$\|X\widehat{\beta}_1 - X\beta^*\|^2 \leq 2\lambda_1\|\beta^*\|_1,$$

which completes the proof.

The second error bound follows from an analogous argument, with the help of inequalities $\|\widehat{\beta}_1\|_1 \leq k^{1/2}\|\beta_1\|$ and $\|\beta^*\|_1 \leq (k^*)^{1/2}\|\beta^*\|$.

**A.2.2 Proof of Theorem 2**

We present the general version of the proof that corresponds to the remark after the statement of the theorem.

Define $\sigma_M = \max_{S \subset \mathcal{N}, |S| = k} \sigma_{\max}(X_S^TX_S)$ and $\sigma_m = \min_{S \subset \mathcal{N}, |S| = 2k} \sigma_{\min}(X_S^TX_S)$. We will use $\gtrsim$ and $\lesssim$ to mean that inequalities $\geq$ and $\leq$, respectively, hold when the right hand side is multiplied by some positive universal constant. We will need the following result, which is proved in Section A.2.3.

**Lemma 1** There exist positive universal constants $c_8$ and $c_9$, such that

$$P\left(\max_{S \subset \mathcal{N}, |S| = k} \|X_S^T\epsilon\| \leq c_8\sigma_m^{1/2}\sqrt{k\log p}\right) \leq 2\exp(-c_9[\sigma_m/\sigma_M]k\log p).$$

116
We write $P_{X_S}v$ for the projection of the vector $v$ onto the space spanned by the columns of $X$ that are indexed by the set $S$. Define

$$\tilde{S} = \arg\max_{S \subseteq \mathcal{N}, |S|=k} \|P_{X_S} \epsilon\| \quad \text{and} \quad S' = \arg\max_{S \subseteq \mathcal{N}, |S|=k} \|X_S^T \epsilon\|.$$ 

Note that

$$PE(\hat{\beta}_{t_0}) \geq \|X_\hat{\beta}_{t_0}\| - \|X\beta^*\| = \max_{S:|S|=k} \|P_{X_S}y\| - \|X\beta^*\| \geq \|P_{X_S}y\| - \|X\beta^*\| \geq \|P_{X_S} \epsilon\| - 2\|\beta^*\|.$$ 

(A.10)

Observe that

$$\|P_{X_S} \epsilon\| \geq \|P_{X_{S'}} \epsilon\| \geq \sigma_M^{-1/2} \max_{S \subseteq \mathcal{N}, |S|=k} \|X_S^T \epsilon\|.$$ 

Lemma 1 then gives

$$\|P_{X_S} \epsilon\| \geq \sigma_{[\sigma_m/\sigma_M]}^{1/2} \sqrt{k \log p},$$

with probability at least $1 - 2 \exp(-c_0(\sigma_m/\sigma_M)k \log p)$. Thus, by inequality (A.10), if $\|\beta^*\|_1 \lesssim \sigma_{[\sigma_m/\sigma_M]}\sqrt{k \log p}$ with a sufficiently small universal constant, then

$$PE(\hat{\beta}_{t_0}) \lesssim \sigma_{[\sigma_m/\sigma_M]}^{1/2} \sqrt{k \log p},$$

(A.11)

with probability at least $1 - 2 \exp(-c_0(\sigma_m/\sigma_M)k \log p)$. To complete the proof, note that the aforementioned universal constant can be chosen sufficiently small to ensure that the upper bounds on $PE^2(\hat{\beta}_1)$ and $PE^2(\hat{\beta}_2)$ in Theorem 1 are at most half the square of the right-hand side in (A.11).

A.2.3 Proof of Lemma 1

Let $\mathcal{H} = \{v \in \{-1,0,1\}^p, v_{\mathcal{N}^c} = 0, \|v\|_0 = k\}$ and note that

$$\max_{S \subseteq \mathcal{N}, |S|=k} \|X_S^T \epsilon\| \geq k^{-1/2} \max_{S \subseteq \mathcal{N}, |S|=k} \|e^T X_S\| \geq k^{-1/2} \max_{v \in \mathcal{H}} |e^T X v|. \quad (A.12)$$
By Lemma 4 in [62], there exists a subset \( \hat{\mathcal{H}} \subseteq \mathcal{H} \), with cardinality bounded below by \( \left\lceil \frac{|\mathcal{H}|}{k^2} \right\rceil k/2 \), such that \( \|v_1 - v_2\|^2 \geq k/2 \), for \( v_1, v_2 \in \hat{\mathcal{H}} \). The last inequality implies

\[
\|Xv_1 - Xv_2\|^2 \geq \sigma_m k/2.
\]

Consequently, by Sudakov's minoration:

\[
\mathbb{E} \max_{v \in \hat{\mathcal{H}}} \epsilon^\top Xv \gtrsim \sigma^1/2_m k \sqrt{k \log(|\hat{\mathcal{H}}|)} \gtrsim \sigma^1/2_m k \sqrt{\log p}.
\]

Denote \( \max_{v \in \hat{\mathcal{H}}} \epsilon^\top Xv \) by \( W \) and \( \max_{v \in \hat{\mathcal{H}}} SD(\epsilon^\top Xv) \) by \( v \). By the concentration inequality for the supremum of a Gaussian process, we have, for all \( t \geq 0 \),

\[
P(W \leq EW - vt) \leq 2 \exp(-t^2/2).\]

Note that

\[
v \leq \sigma \max_{v \in \hat{\mathcal{H}}} \|Xv\| \leq \sigma^1/2_M \sqrt{k}.
\]

Thus, for \( t = \sigma^1/2_m \sigma^1/2_M \sqrt{k \log p} \) we have \( EW - vt \gtrsim \sigma^1/2_m k \sqrt{\log p} \), and

\[
P\left(W \leq c_8 \sigma^1/2_M \sqrt{\log p}\right) \leq 2 \exp(-c_9 [\sigma_m/\sigma_M] k \log p),
\]

for some positive universal constants \( c_8 \) and \( c_9 \). In view of the inequalities in (A.12), this completes the proof of the lemma.
Appendix B

Appendices for Chapter 3

B.1 Proof of Theorem 3

We fix $\tau > 0$ and denote $X = (X_1, \ldots, X_p, e) \in \mathbb{R}^{n \times (p+1)}$ the design matrix to which we have added the column $e = (1, \ldots, 1)^T \in \mathbb{R}^n$.

For $(\beta, \beta_0) \in \mathbb{R}^{p+1}$, we define $w^T(\beta, \beta_0) \in \mathbb{R}^n$ by:

$$w_i^T(\beta, \beta_0) = \min \left(1, \frac{1}{2\tau} |1 - y_i(x_i^T \beta + \beta_0)| \right) \text{sign}(1 - y_i(x_i^T \beta + \beta_0)), \forall i$$

We verify that

$$w^T(\beta, \beta_0) = \arg \max_{||w||_\infty \leq 1} \sum_{i=1}^n \frac{1}{2} \left[1 - y_i(x_i^T \beta + \beta_0) + w_i(1 - y_i(x_i^T \beta + \beta_0))\right] - \frac{T}{2} ||w||^2_2.$$

Then the gradient of the smooth hinge-loss is

$$\nabla L^r(\beta, \beta_0) = -\frac{1}{2} \sum_{i=1}^n (1 + w_i^T(\beta, \beta_0))y_i(x_i^T, 1)^T \in \mathbb{R}^{p+1}. \quad (B.1)$$

We fix $(\beta, \beta_0), (\gamma, \gamma_0) \in \mathbb{R}^{p+1}$ and we have:

$$\nabla L^r(\beta, \beta_0) - \nabla L^r(\gamma, \gamma_0) = \frac{1}{2} \sum_{i=1}^n (w_i^r(\gamma, \gamma_0) - w_i^r(\beta, \beta_0))y_i(x_i^T, 1)^T. \quad (B.2)$$
For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ we define the vector $\mathbf{a} \ast \mathbf{b} = (a_i b_i)_{i=1}^n$. We can rewrite Equation (B.2) as

$$\nabla L^T(\mathbf{\beta}, \beta_0) - \nabla L^T(\gamma, \gamma_0) = \frac{1}{2} \mathbf{x}^T [\mathbf{y} \ast (\mathbf{w}^T(\gamma, \gamma_0) - \mathbf{w}^T(\beta, \beta_0))].$$  \tag{B.3}

The operator norm associated to the Euclidean norm of the matrix $\mathbf{X}$ is $\|\mathbf{X}\| = \max_{\|\mathbf{z}\|_2 = 1} \|\mathbf{Xz}\|_2$.

Let us recall that $\|\mathbf{X}\|^2 = \|\mathbf{X}^T\|^2 = \|\mathbf{X}^T\mathbf{X}\| = \mu_{\max}(\mathbf{X}^T\mathbf{X})$, which corresponds to the highest eigenvalue of the matrix $\mathbf{X}^T \mathbf{X}$.

Consequently, Equation (B.3) leads to:

$$\|\nabla L^T(\mathbf{\beta}, \beta_0) - \nabla L^T(\gamma, \gamma_0)\|_2 \leq \frac{1}{2} \|\mathbf{X}\| \|\mathbf{w}^T(\gamma, \gamma_0) - \mathbf{w}^T(\beta, \beta_0)\|_2.$$

(B.4)

Besides the first order necessary conditions for optimality applied to $\mathbf{w}^T(\mathbf{\beta}, \beta_0)$ and $\mathbf{w}^T(\gamma, \gamma_0)$ give

$$\sum_{i=1}^n \left\{ \frac{1}{2} (1 - y_i (\mathbf{x}_i^T \mathbf{\beta} + \beta_0)) - \tau \mathbf{w}_i^T(\mathbf{\beta}, \beta_0) \right\} \{w_i^T(\gamma, \gamma_0) - w_i^T(\beta, \beta_0)\} \leq 0,$$  \tag{B.5}

and

$$\sum_{i=1}^n \left\{ \frac{1}{2} (1 - y_i (\mathbf{x}_i^T \gamma + \gamma_0)) - \tau \mathbf{w}_i^T(\gamma, \gamma_0) \right\} \{w_i^T(\beta, \beta_0) - w_i^T(\gamma, \gamma_0)\} \leq 0.$$  \tag{B.6}

Then by adding Equations (B.5) and (B.6) and rearranging the term we have:

$$\tau \|\mathbf{w}^T(\gamma, \gamma_0) - \mathbf{w}^T(\beta, \beta_0)\|_2^2$$

$$\leq \frac{1}{2} \sum_{i=1}^n y_i \{\mathbf{x}_i^T (\mathbf{\beta} - \gamma) + \beta_0 - \gamma_0\} \{w_i^T(\gamma, \gamma_0) - w_i^T(\beta, \beta_0)\}$$

$$\leq \frac{1}{2} \|\mathbf{X} ((\mathbf{\beta}, \beta_0) - (\gamma, \gamma_0)) \|_2 \|\mathbf{w}^T(\gamma, \gamma_0) - \mathbf{w}^T(\beta, \beta_0)\|_2$$

$$\leq \frac{1}{2} \|\mathbf{X}\| \|(\mathbf{\beta}, \beta_0) - (\gamma, \gamma_0)\|_2 \|\mathbf{w}^T(\gamma, \gamma_0) - \mathbf{w}^T(\beta, \beta_0)\|_2,$$

where we have used Cauchy-Schwartz inequality. We easily derive:

$$\|\mathbf{w}^T(\gamma, \gamma_0) - \mathbf{w}^T(\beta, \beta_0)\|_2 \leq \frac{1}{2\tau} \|\mathbf{X}\| \|(\mathbf{\beta}, \beta_0) - (\gamma, \gamma_0)\|_2.$$  \tag{B.8}
From Equations (B.4) and (B.8) we conclude the proof by noting that:

\[
\|\nabla L^T(\beta, \beta_0) - \nabla L^T(\gamma, \gamma_0)\|_2 \leq \frac{1}{4\tau} \|X\|^2 \|\beta - \beta_0\|_2 \quad (B.9)
\]

\[
= \frac{\mu_{\text{max}}(X^TX)}{4\tau} \|\beta - \beta_0\|_2. \quad (B.10)
\]