Learning Optimized Risk Scores from Large-Scale Datasets

Berk Ustun ustunb@mit.edu
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

Cynthia Rudin cynthia@cs.duke.edu
Department of Computer Science
Duke University

Abstract

Risk scores are simple classification models that let users quickly assess risk by adding, subtracting, and multiplying a few small numbers. These models are widely used for high-stakes applications in healthcare and criminology, but are difficult to create because they need to be risk-calibrated, sparse, use small integer coefficients, and obey operational constraints. In this paper, we present a new approach to learn risk scores that are fully optimized for feature selection, integer coefficients, and operational constraints. We formulate the risk score problem as a mixed integer nonlinear program, and present a new cutting plane algorithm to efficiently recover its optimal solution while avoiding the stalling behavior of existing cutting plane algorithms in non-convex settings. We pair our algorithm with specialized techniques to generate feasible solutions, narrow the optimality gap, and reduce data-related computation. The resulting approach can learn optimized risk scores in a way that scales linearly in the number of samples, provides a proof of optimality, and accommodates complex operational constraints. We illustrate the benefits of this approach through extensive numerical experiments.

Keywords: classification; cutting plane methods; discrete optimization; mixed integer non-linear programming; risk assessment; scoring systems;

1. Introduction

Risk scores are simple linear classification models that let users assess risk by adding, subtracting, and multiplying a few small numbers. These models are widely used for high-stakes applications in medicine and criminology because they let users make quick predictions, without extensive training, and without the use of a computer or calculator. The nature of such applications means that risk scores have to be rank-accurate (i.e., high AUC), risk-calibrated, sparse, and use small integer coefficients. In addition, domain experts may require risk scores to satisfy operational constraints to be deployed, such as limits on model size ("the model should use at most 4 features"), feature composition ("if the model uses *hypertension*, then it should also use *bmi* ≥ 25"), and prediction ("the model should *not* predict that males have higher risk than females").

In spite of extensive use in high-stakes applications, there is currently no principled approach to learn risk scores from data. Existing models are created ad hoc, either by a panel of experts (see e.g., the CHADS$_2$ score of Gage et al., 2001) or by combining heuristics (e.g. rounding logistic regression coefficients after manual feature selection, as recommended by the U.S. Department of Justice, 2005). These approaches may produce risk scores with poor rank-accuracy or risk calibration compared to the state-of-the-art (see e.g., the performance of CHADS$_2$ in Letham et al. 2013). As we show, modern methods for penalized logistic regression are also ill-suited to create risk scores as they must be paired with a rounding heuristic to produce models with small integer coefficients; in that case, the resulting approach combines multiple approximations, depends on extensive parameter tuning, and provides no guarantee as to whether it can produce a feasible risk score, let alone one that is optimal. In practice, the lack of a formal guarantee makes it difficult to navigate trade-offs between model form and model performance, as practitioners cannot tell if a model performs poorly due to restrictive constraints on model form, or lax approximations in the approach to determine the coefficients.

Our goal in this paper is to present a principled approach to learn risk scores by solving a discrete optimization problem, namely the *risk score problem*. We consider a mixed integer nonlinear program (MINLP) that minimizes the logistic loss (for rank accuracy and risk calibration), penalizes the ℓ_0-norm (for sparsity), and uses discrete variables to restrict coefficients to small integers and enforce operational constraints. We refer to the risk score built by solving this problem as a *Risk-calibrated Supersparse Linear Integer Model* (RiskSLIM). Unlike existing approaches, our approach is designed to fit models that are fully optimized for feature selection, small integer coefficients, and operational constraints. In addition, it provides a simple mechanism to address operational constraints without parameter tuning or post processing, by explicitly including these constraints in the MINLP formulation. In light of these benefits, a key part of our approach hinges on solving the risk score problem to optimality (i.e., to recover the optimal solution and pair it with a certificate of optimality). By design, the optimal solution to the risk score problem attains the best performance among a class of models that satisfy a fixed set of constraints. In solving this problem to optimality, we end up with a risk score with suitable performance, or a certificate proving that the constraints were overly restrictive.

As we show, solving the risk score problem to optimality with off-the-shelf MINLP solvers is time-consuming even on small datasets, as algorithms for generic MINLPs are slowed down by excessive data-related computation for our problem. Accordingly, we solve the risk score problem with a *cutting plane algorithm*, which reduces data-related computation by iteratively solving a surrogate problem with a piecewise-linear approximation of the loss function that is much cheaper to evaluate. Cutting plane algorithms have an impressive track record on large-scale supervised learning problems (see Teo et al., 2009; Franc and Sonnenburg, 2009; Joachims et al., 2009), as they scale linearly with the number of samples in a dataset and provide precise control over data-related computation. Unfortunately, existing cutting plane algorithms are designed under the assumption that the surrogate optimization problem can be solved to optimality at each iteration. Although this is perfectly reasonable in a convex setting, it leads cutting plane algorithms to *stall* on non-convex problems, as they eventually reach an iteration where the time to solve the surrogate problem to optimality increases exponentially. We demonstrate the stalling phenomenon, and present a cutting plane algorithm to overcome this issue, called the *lattice cutting plane algorithm* (LCPA). The resulting approach extends the benefits of cutting plane algorithms to discrete optimization problems, allowing us to learn optimized risk scores in a way that scales linearly in the number of samples, provides a certificate of optimality, and complex operational constraints needed for these models to be deployed.

Contributions

The main contributions of this paper are as follows.

- We propose a new approach to build risk scores that are fully optimized for feature selection, small integer coefficients, and operational constraints. Our approach has the benefit that it fits simple models without parameter tuning or post processing, and pair all models with a certificate of optimality.
- We present a new cutting plane algorithm for discrete optimization problems, called the lattice cutting plane algorithm. Our algorithm retains the scalability and flexibility of cutting plane algorithms designed for convex optimization problems, but does not stall in non-convex settings.
- We improve the performance of LCPA for the risk score problem by designing specialized techniques to: (i) generate and polish integer feasible solutions; (ii) narrow the optimality gap; (iii) reduce data-related computation. We use the techniques in (i) to propose new methods to build risk scores heuristically.
- We benchmark risk scores built using several methods on publicly available datasets. Our results show that we can use our approach to fit optimized risk scores in minutes, and illustrate the pitfalls of heuristics that are often used in practice.
- We provide software to create optimized risk scores using Python and the CPLEX API.

Structure

Our paper is structured as follows. In Section 1.1, we discuss related work. In Section 2, we formalize the risk score problem. In Section 3, we briefly review existing cutting plane algorithms, and present our proposed cutting plane algorithm, LCPA. In Sections 4 and 5, we present specialized techniques to improve the performance of LCPA for the risk score problem. In Section 6, we present experimental results to benchmark risk scores built using various methods in terms of rank-accuracy and risk calibration. In the appendices, we include: proofs of all theorems (Appendix A); details on our implementation (Appendix B); details on computational experiments in Sections 3–5 (Appendix C).

1.1 Related Work

In what follows, we discuss related work in risk scores, machine learning, and optimization.

Risk Scores

Starting possibly with the work of Burgess (1928), simple risk scores have been used in many high-stakes applications. Examples in criminology include: the Ohio Risk Assessment System (Latessa et al., 2009); the Kentucky Pretrial Risk Assessment Instrument (Austin et al., 2010); and the Offense Gravity Score (Pennsylvania Commission on Sentencing, 2012). Examples in medicine include: SAPS I, II, III (Le Gall et al., 1984, 1993; Moreno et al., 2005) and APACHE I, II, III (Knaus et al., 1981, 1985, 1991) to assess ICU mortality risk.

Risk scores with few terms and small integer coefficients are generally preferred for these applications due to their high level of interpretability and their ability to make predictions without a computer or a calculator. Duwe and Kim (2016), for example, write: "*It is commonplace... for fine-tuned regression coefficients to be replaced with a simple-point system... to promote the easy implementation, transparency, and interpretability of risk-assessment instruments.*" Ridgeway (2013) provides similar reasons to justify why the LAPD uses a simple risk score to identify recruits who may become officers: "*A decent transparent model that is actually used will outperform a sophisticated system that predicts better but sits on a shelf. If the researchers had created a model that predicted well but was more complicated, the LAPD likely would have ignored it, thus defeating the whole purpose.*" Interpretability has become a crucial requirement for models in high-stakes applications (Kodratoff, 1994), as evidenced by upcoming EU regulations that require a "right to an explanation" from algorithmic decision-making tools (Goodman and Flaxman, 2016) as well as recent stream of work to improve transparency and interpretability in machine learning (see e.g., **?????**).

The lack of a principled methodology to create risk scores is best evidenced by the number of ad hoc approaches that are used in practice (see e.g., the review of Bobko et al., 2007). Existing tools are built by combining heuristics for fitting, rounding and feature selection. Examples include: (i) having a panel of experts build a model by hand and only using data to validate the model as it is built (see e.g., Gage et al., 2001); (ii) creating a risk score by adding together each variable that are strongly correlated with the outcome variable (i.e., the *Burgess* method as it was first used by Burgess, 1928); (iii) rounding the coefficients of a logistic regression model (Goel et al., 2015) or a linear probability model (U.S. Department of Justice, 2005). Risk scores built using ad hoc approaches can perform poorly compared to models built using modern data-driven methods (see e.g., the poor performance of CHADS$_2$ in Letham et al. 2013, as well as risk scores built with the Burgess method in Duwe and Kim 2016).

Machine Learning

The RiskSLIM models in this paper have the same form as the SLIM models in Ustun and Rudin (2015), but differ in the way they are used and built. RiskSLIM models are designed for risk assessment and built by solving a mixed-integer nonlinear program (MINLP) that optimizes the logistic loss. In contrast, SLIM models are designed for decision-making and built by solving a mixed-integer linear program (MIP) that optimizes the 0–1 loss. Optimizing the 0–1 loss does not necessarily produce models that are risk-calibrated or rank accurate. In addition, SLIM does not generally scale to problems with large sample sizes as is the case here.

Our risk score problem resembles the one in Ertekin and Rudin (2015) in that it minimizes the logistic loss, penalizes the ℓ_0-norm, and restricts coefficients to a discrete set. Ertekin and Rudin (2015) solve this problem approximately using a Bayesian approach, which has a benefit in that it produces a posterior distribution on the coefficients. In contrast, we use an optimization approach, which can consistently recover a globally optimal solution even under operational constraints. We pair our approach with new techniques to generate feasible solutions and reduce data-related computation, resulting in a substantially more efficient and flexible approach than that of Ertekin and Rudin (2015).

Our cutting plane algorithm can be extended to solve a wide class of risk minimization problems that optimize a convex loss function and include non-convex penalties and/or constraints – in a way that returns a global optimum and scales linearly in the sample size. Examples include ℓ_0-regularized risk minimization problems (**??**), or discrete linear classification problems that minimize the hinge loss over a small set of integers (Chevaleyre et al., 2013; Malioutov and Varshney, 2013; Carrizosa et al., 2016).

Optimization

We fit risk scores by solving a MINLP with three main components: (i) a convex loss function; (ii) a non-convex regularization penalty (i.e., the ℓ_0-penalty); (iii) a discrete and bounded feasible region. In Section 3.3, we argue that this MINLP requires a specialized algorithm, because off-the-shelf MINLP solvers fail to solve instances for small datasets.

Recent work in optimization has proposed methods to solve problems that contain some, but not all, of these components. This work cannot be readily adapted to handle our problem. The quadratic integer programming method of Park and Boyd (2015a), for instance, can be extended to handle problems with a convex loss but cannot handle a non-convex regularizer. Similarly, Hübner and Schöbel (2014) derive sufficient conditions to recover an integer minimizer to a nonlinear optimization problem by rounding its fractional minimizer. However, these conditions do not hold in our setting due to additional non-convexities imposed by the ℓ_0-penalty and operational constraints.

We solve the risk score problem with a cutting plane algorithm, which have been extensively studied by the optimization community (see e.g., Kelley, 1960), and successfully used to solve large-scale supervised learning problems (Teo et al., 2007, 2009; Franc and Sonnenburg, 2008, 2009; Joachims, 2006; Joachims et al., 2009). Our algorithm makes use of callbacks in modern MIP solvers to dynamically add cutting planes while performing a branch-and-bound search (see e.g., Bai and Rubin, 2009; Naoum-Sawaya and Elhedhli, 2010, for similar uses of callbacks in the optimization literature). LCPA differs from existing algorithms in that it does not *stall* on non-convex problems (i.e., it does not spend too much time solving a surrogate problem to optimality before converging). As we explain in Section 3.1, many cutting plane algorithms will stall on non-convex problems including those that are not used in machine learning (see Boyd and Vandenberghe, 2004, for a list).

2. Problem Statement

Our goal is to build risk assessment tools (i.e., *risk scores*) such as the one in Figure 1.

1.	*Congestive Heart Failure*	1 point		$\cdots\cdots$
2.	*Hypertension*	1 point	$+$	$\cdots\cdots$
3.	*Age $\geq$ 75*	1 point	$+$	$\cdots\cdots$
4.	*Diabetes Mellitus*	1 point	$+$	$\cdots\cdots$
5.	*Prior Stroke or Transient Ischemic Attack*	2 points	$+$	$\cdots\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$	$\cdots\cdots$

SCORE	0	1	2	3	4	5	6
RISK	1.9%	2.8%	4.0%	5.9%	8.5%	12.5%	18.2

Figure 1: CHADS$_2$ risk score of Gage et al. (2001) for assessing stroke risk.

We formalize this problem as follows. We start with a dataset of N i.i.d. training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where $\boldsymbol{x}_i \subseteq \mathbb{R}^{d+1}$ denotes a vector of features $[1, x_{i,1}, \ldots, x_{i,d}]^T$ and $y_i \in \{\pm 1\}$ denotes a class label. We consider a linear score function $\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle$ where $\boldsymbol{\lambda} \subseteq \mathbb{R}^{d+1}$ is a vector of coefficients $[\lambda_0, \lambda_1, \ldots, \lambda_d]^\top$, and λ_0 is an intercept term. We estimate the *predicted risk* that example i belongs to the positive class using the logistic link function as

$$\Pr(y_i = +1 \mid \boldsymbol{x}_i) = \frac{1}{1 + \exp(-\langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle)}.$$

In this setup, each coefficient λ_j represents the number of points for a given feature. Given an example with features $\boldsymbol{x}_i$, users tally the points to obtain a total score $s_i := \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle$, and use the total score s_i to obtain an estimate of predicted risk. Alternatively, users can obtain a predicted label $\hat{y}_i \in \{\pm 1\}$ by comparing the predicted risk to a threshold risk (e.g., predict $\hat{y}_i = +1$ if and only if $\Pr(y_i = +1) > 50\%$).

In practice, the desirable characteristics of a risk score include all of the following:

- *Rank Accuracy*: A rank-accurate model is a model with high AUC. Such a model outputs scores that can be used to order examples in terms of their true risk.

- *Risk Calibration*: A risk-calibrated model yields risk predictions that match observed risk. A risk-calibrated model has high AUC, but the converse is not necessarily true.

- *Sparsity and Small Integer Coefficients*: sparse linear models with small integer coefficients allow for quick predictions without a computer or calculator. Such models help users gauge the influence of each input variable, and understand how multiple input variables interact to produce an estimate of predicted risk – especially as humans have severe difficulties when handling multiple cognitive entities (at most 4 according to Cowan 2010, which updates the estimate of 7 ± 2 from Miller 1984) and understanding associations between multiple variables (3 or more according to Jennings et al., 1982).

Building risk scores involves trade-offs between model performance (i.e., AUC and risk calibration) and model form (i.e., sparsity and small integer coefficients). We expect to sacrifice some performance by constraining models to use a few small integer coefficients, and expect to recover this as we relax constraints on model form (see e.g., the trade-offs between model size, AUC and risk calibration in Figures 14 and 15 from Section 6). In practice, fitting a risk score that strikes an acceptable balance between performance and form is difficult because : (i) AUC and risk calibration are potentially competing objectives (see Caruana and Niculescu-Mizil, 2006, for a discussion); (ii) restricting coefficients to small integers may not impact performance for highly sparse models (i.e., $\|\boldsymbol{\lambda}\|_0 \leq 5$); (iii) constraints on model form may be desirable because they mitigate overfitting; (iv) the trade-off is difficult to understand when models have to obey additional operational constraints. Current approaches address sparsity and integer coefficients through approximations, and cannot guarantee a pareto-optimal model with respect to these trade-offs.

Risk Score Problem

We learn the values of the coefficients from data by solving the following mixed integer non-linear program (MINLP), which we refer to as the *risk score problem* or RiskSlimMINLP:

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \quad & l(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0 \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in \mathcal{L}. \end{aligned} \tag{1}$$

Here, the objective minimizes the *logistic loss* $l(\boldsymbol{\lambda}) := \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-\langle \boldsymbol{\lambda}, y_i \boldsymbol{x}_i\rangle))$ for AUC and risk calibration, and penalizes the ℓ_0-norm $\|\boldsymbol{\lambda}\|_0 := \sum_{j=0}^{d} \mathbb{1}[\lambda_j \neq 0]$ for sparsity. The trade-off parameter C_0 controls the balance between these competing objectives, and represents the maximum log-likelihood that will be sacrifice to remove a feature from the optimal model[1] . The feasible region restricts coefficients to a small set of bounded integers such as $\mathcal{L} = \{-5, \ldots, 5\}^{d+1}$, and may be further customized to include operational constraints, such as those in Table 1.

Constraint Type	**Example**
Feature Selection	Choose between 5 to 10 total features
Group Sparsity	Include either *male* or *female* in the model but not both
Optimal Thresholding	Use at most 3 thresholds for indicator variables: $\sum_{k=1}^{100} \mathbb{1}[age \leq k] \leq 3$
Logical Structure	If *male* is in model, then include *hypertension* or *bmi* ≥ 30 as a control
Probability	Predict $\Pr(y=+1\|\boldsymbol{x}) \geq 0.90$ when *male* = TRUE and *hypertension* = TRUE

Table 1: Operational constraints that can be enforced on risk scores by including additional constraints in the feasible region of RiskSlimMINLP.

RiskSlimMINLP is formulated to capture exact objectives and constraints for risk scores so that an optimal solution to RiskSlimMINLP corresponds to a model that attains the best possible performance among the class of models that satisfy constraints on model form. This statement deserves some qualification. In particular, an optimal solution to RiskSlimMINLP attains the lowest value of the logistic loss among feasible models on the training data – provided that C_0 is small enough (see Appendix A.1 for a proof). Our

1. When $\mathbf{0} \in \mathcal{L}$, we can bound the trade-off parameter to values such that $C_0 \in (0, \log(2))$. If $\mathbf{0} \in \mathcal{L}$, then setting $C_0 \geq \log(2)$ will provably return $\boldsymbol{\lambda}^* = \mathbf{0}$ as the optimal solution.

empirical results in Section 6 show that models that minimize the logistic loss achieve good performance in terms of AUC and risk calibration on the training data, and that this performance generalizes to the test data given the simplicity of the model class. There are some theoretical results to motivate why minimizers of the logistic loss have good AUC and risk calibration. In particular, the logistic loss is a strictly proper loss (Reid and Williamson, 2010) which yields calibrated estimates of predicted risk under the parametric assumption that the true risk can be modeled using a logistic link function (see Menon et al., 2012). In addition, the work of Kotlowski et al. (2011) shows that a "balanced" version of the logistic loss forms a lower bound on $1-$AUC, which means that minimizing the logistic loss indirectly maximizes a surrogate of AUC (i.e., a lower bound on a "scaled" AUC).

Optimizing RiskSlimMINLP is a challenging computational task since ℓ_0-regularization leads to problems that are $\mathcal{NP}$-hard (Amaldi and Kann, 1998), minimizing over integers is $\mathcal{NP}$-hard (Gary and Johnson, 1979), and MINLP problems are $\mathcal{NP}$-hard (Bonami et al., 2012). These worst-case complexity results mean that finding a provably optimal solution to RiskSlimMINLP may be intractable for large dimensional datasets. As we will show, however, RiskSlimMINLP can be solved to optimality for a range of real-world datasets in minutes, and in a way that scales linearly in N.

Assumptions, Notation and Terminology

We use the following notation. We denote the objective function of RiskSlimMINLP as

$$V(\boldsymbol{\lambda}) := l(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0, \tag{2}$$

and denote an optimal solution as $\boldsymbol{\lambda}^* \in \operatorname{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} V(\boldsymbol{\lambda})$. We bound the values of objective value function and its components at the optimal solution $\boldsymbol{\lambda}^*$ as follows: $V(\boldsymbol{\lambda}^*) \in [V^{\min}, V^{\max}]$, $l(\boldsymbol{\lambda}^*) \in [L^{\min}, L^{\max}]$, $\|\boldsymbol{\lambda}^*\|_0 \in [R^{\min}, R^{\max}]$. In addition, we define the set of feasible values for each coefficient as $\lambda_j \in \mathcal{L}_j$ for $j = 0, \ldots, d$, and denote bounds as $\Lambda_j^{\min} = \min_{\lambda_j \in \mathcal{L}_j} \lambda_j$ and $\Lambda_j^{\max} = \max_{\lambda_j \in \mathcal{L}_j} \lambda_j$.

Unless otherwise specified, we make the following assumptions for clarity of exposition: (i) the coefficient set contains the null vector, $\mathbf{0} \in \mathcal{L}$; (ii) the intercept is never penalized, which means that the more precise version of the objective function in (1) is $V(\boldsymbol{\lambda}) := l(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}_{[1,d]}\right\|_0$ where $\boldsymbol{\lambda} = [\lambda_0, \boldsymbol{\lambda}_{[1,d]}]$. Here, the first assumption is to ensure that a generic formulation of RiskSlimMINLP is always feasible, and the second assumption ensures that the intercept term is not regularized.

Since RiskSlimMINLP is a non-convex optimization problem, we need to distinguish between finding the optimal solution to RiskSlimMINLP and proving that this solution is optimal. We say that we have *solved* RiskSlimMINLP *to optimality* if and only if we achieve both tasks. We can assess the optimality of any solution $\boldsymbol{\lambda}$ through its *optimality gap* $\frac{V(\boldsymbol{\lambda})-V^{\min}}{V(\boldsymbol{\lambda})}$. The optimality gap associated with RiskSlimMINLP is defined by the best integer feasible solution, $\boldsymbol{\lambda} \in \mathcal{L}$ such that $V(\boldsymbol{\lambda}) = V^{\max}$, and computed as $\frac{V^{\max}-V^{\min}}{V^{\max}}$. A *certificate of optimality* for RiskSlimMINLP corresponds to an optimality gap of 0.0%.

3. Methodology

In this section, we introduce the cutting plane algorithm that we use to solve the risk score problem, RiskSlimMINLP. In Section 3.1, we discuss a simple cutting plane algorithm to explain the benefits of using cutting plane algorithms to solve RiskSlimMINLP and explain why such algorithms stall on non-convex problems. In Section 3.2, we present a new cutting plane algorithm that does not stall for non-convex problems. In Section 3.3, we provide insights into the real-world performance of both cutting plane algorithms and off-the-shelf MINLP solvers by using them to solve difficult instances of RiskSlimMINLP.

3.1 Cutting Plane Algorithms

In Algorithm 1, we present a simple cutting plane algorithm to solve RiskSlimMINLP that we refer to as CPA. In what follows, we use CPA to briefly introduce cutting plane algorithms, explain why they are well-suited to solve RiskSlimMINLP, and why they stall on risk minimization problems with non-convex constraints and regularizers.

CPA recovers the optimal solution to RiskSlimMINLP by repeatedly solving a mixed-integer programming (MIP) *surrogate problem* in which the loss function $l(\boldsymbol{\lambda})$ is approximated with cutting planes. A *cutting plane* or *cut* is a supporting hyperplane to the loss function at a point $\boldsymbol{\lambda}^t \in \mathcal{L}$ with the form

$$l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle.$$

where $l(\boldsymbol{\lambda}^t) \in \mathbb{R}_+$ and $\nabla l(\boldsymbol{\lambda}^t) \in \mathbb{R}^d$ are *cut parameters* that represent the value and gradient of the loss function at $\boldsymbol{\lambda}^t$

$$l(\boldsymbol{\lambda}^t) = \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-\langle \boldsymbol{\lambda}^t, y_i\boldsymbol{x}_i\rangle)), \quad \nabla l(\boldsymbol{\lambda}^t) = \frac{1}{N}\sum_{i=1}^{N} \frac{-y_i\boldsymbol{x}_i}{1+\exp(-\langle \boldsymbol{\lambda}^t, y_i\boldsymbol{x}_i\rangle)}. \tag{4}$$

Multiple cuts can be combined to produce a piecewise linear approximation of the loss function as shown in Figure 2. We denote the *cutting plane approximation* of the loss function built using k cuts at the points $\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^k$ as

$$\hat{l}^k(\boldsymbol{\lambda}) = \max_{t=1\ldots k} l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle.$$

On iteration k, CPA solves the surrogate problem RiskSlimMIP($\hat{l}^k(\boldsymbol{\lambda})$) which minimizes the approximate loss $\hat{l}^k(\boldsymbol{\lambda})$. CPA uses the optimizer of the surrogate problem $(\theta^k, \boldsymbol{\lambda}^k)$ in two ways: (i) it adds a cut at $\boldsymbol{\lambda}^k$ to improve the cutting plane approximation $\hat{l}^k(\boldsymbol{\lambda})$; (ii) it computes bounds on optimal value of RiskSlimMINLP to check convergence. The upper bound is set as the objective value of the best solution across all iterations

$$V^{\max} = \min_{t=1\ldots k} l(\boldsymbol{\lambda}^t) + C_0\|\boldsymbol{\lambda}^t\|_0$$

The lower bound is set as the optimal value to the surrogate problem at the latest iteration

$$V^{\min} = \hat{l}^k(\boldsymbol{\lambda}^k) + C_0\|\boldsymbol{\lambda}^k\|_0. \tag{5}$$

Algorithm 1 Cutting Plane Algorithm (CPA)

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^N$                                   training data
    $\mathcal{L}$                                         constraint set for RiskSlimMINLP
    $C_0$                                             $\ell_0$ penalty parameter for RiskSlimMINLP
    $\varepsilon^{\text{stop}} \in [0,1]$                       maximum optimality gap of acceptable solution

Initialize
    $k \leftarrow 0$                                                  iteration counter
    $\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$                            initial approximation of loss function
    $(V^{\min}, V^{\max}) \leftarrow (0, \infty)$              bounds on the optimal value of RiskSlimMINLP
    $\varepsilon \leftarrow \infty$                        optimality gap of current best solution to RiskSlimMINLP

 1: while $\varepsilon > \varepsilon^{\text{stop}}$ do
 2:     $(\theta^k, \boldsymbol{\lambda}^k) \leftarrow$ provably optimal solution to RiskSlimMIP$(\hat{l}^k(\boldsymbol{\lambda}))$
 3:     compute cut parameters $l(\boldsymbol{\lambda}^k)$ and $\nabla l(\boldsymbol{\lambda}^k)$
 4:     $\hat{l}^{k+1}(\boldsymbol{\lambda}) \leftarrow \max\{\hat{l}^k(\boldsymbol{\lambda}), l(\boldsymbol{\lambda}^k) + \langle \nabla l(\boldsymbol{\lambda}^k), \boldsymbol{\lambda} - \boldsymbol{\lambda}^k \rangle\}$ for all $\boldsymbol{\lambda}$      ▷add cut
 5:     $V^{\min} \leftarrow \theta^k + C_0 \left\|\boldsymbol{\lambda}^k\right\|_0$             ▷lower bound is optimal value of RiskSlimMIP
 6:     $v \leftarrow l(\boldsymbol{\lambda}^k) + C_0 \left\|\boldsymbol{\lambda}^k\right\|_0$
 7:     if $v < V^{\max}$ then
 8:         $V^{\max} \leftarrow v$                                           ▷update upper bound
 9:         $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^k$                                     ▷update incumbent solution
10:     end if
11:     $\varepsilon \leftarrow 1 - V^{\min}/V^{\max}$                                 ▷update optimality gap
12:     $k \leftarrow k + 1$
13: end while
Output: $\boldsymbol{\lambda}^{\text{best}}$, which is an $\varepsilon$-optimal solution to RiskSlimMINLP
```

RiskSlimMIP$(\hat{l}(\boldsymbol{\lambda}))$ is a MIP surrogate of RiskSlimMINLP that minimizes the cutting plane approximation of the loss function $\hat{l}(\boldsymbol{\lambda})$:

$$\begin{aligned} \min_{\theta, \boldsymbol{\lambda}} \quad & \theta + C_0 \|\boldsymbol{\lambda}\|_0 \\ \text{s.t.} \quad & \theta \geq \hat{l}(\boldsymbol{\lambda}) \\ & \boldsymbol{\lambda} \in \mathcal{L}. \end{aligned} \tag{3}$$

We provide a MIP formulation for RiskSlimMIP in Appendix B.

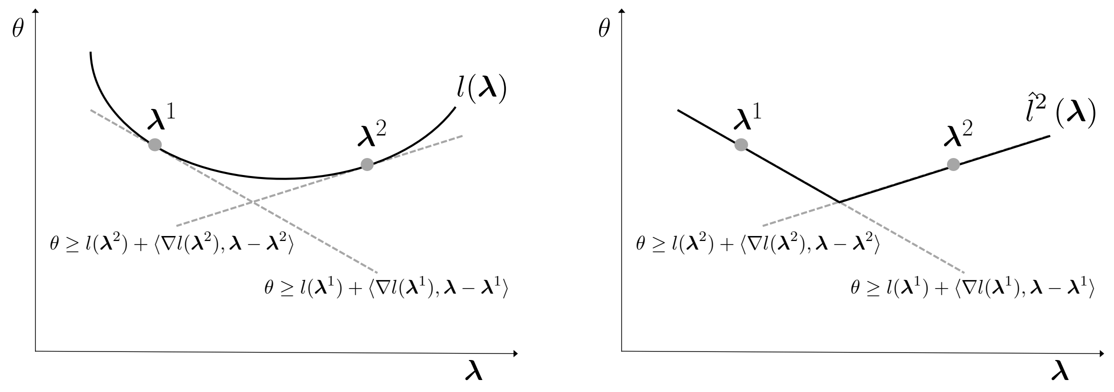
θ
$l(\boldsymbol{\lambda})$
$\boldsymbol{\lambda}^1$
$\boldsymbol{\lambda}^2$
$\theta \geq l(\boldsymbol{\lambda}^2) + \langle \nabla l(\boldsymbol{\lambda}^2), \boldsymbol{\lambda} - \boldsymbol{\lambda}^2 \rangle$
$\theta \geq l(\boldsymbol{\lambda}^1) + \langle \nabla l(\boldsymbol{\lambda}^1), \boldsymbol{\lambda} - \boldsymbol{\lambda}^1 \rangle$
$\boldsymbol{\lambda}$
θ
$\hat{l}^2(\boldsymbol{\lambda})$
$\boldsymbol{\lambda}^1$
$\boldsymbol{\lambda}^2$
$\theta \geq l(\boldsymbol{\lambda}^2) + \langle \nabla l(\boldsymbol{\lambda}^2), \boldsymbol{\lambda} - \boldsymbol{\lambda}^2 \rangle$
$\theta \geq l(\boldsymbol{\lambda}^1) + \langle \nabla l(\boldsymbol{\lambda}^1), \boldsymbol{\lambda} - \boldsymbol{\lambda}^1 \rangle$
$\boldsymbol{\lambda}$

Figure 2: Cutting plane algorithms such as CPA build a piecewise linear approximation of the loss function using cutting planes (i.e., cuts). The plot on the left shows the loss function $l(\boldsymbol{\lambda})$ with cuts at the points $\boldsymbol{\lambda}^1$ and $\boldsymbol{\lambda}^2$. The plot on the right shows the approximate loss function $\hat{l}^2(\boldsymbol{\lambda}) = \max_{t=1,2} l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle$.

CPA converges to an ε-optimal solution of RiskSlimMINLP in a finite number of iterations (see Kelley, 1960, for a proof). To see this, note that the cutting plane approximation of a convex loss function improves monotonically with each added cut:

$$\hat{l}^k(\boldsymbol{\lambda}) \leq \hat{l}^{k+m}(\boldsymbol{\lambda}) \leq l(\boldsymbol{\lambda}) \text{ for all } k, m \in \mathbb{N} \text{ and } \boldsymbol{\lambda} \in \mathcal{L}. \tag{6}$$

Since the cuts added at each iteration are not redundant, the lower bound improves monotonically as the algorithm progress. Once the optimality gap ε is less than a stopping threshold $\varepsilon^{\text{stop}}$, CPA terminates and returns a solution $\boldsymbol{\lambda}^{\text{best}}$ from the ε-level set of optimizers to RiskSlimMINLP.

Benefits of Solving RiskSlimMINLP with Cutting Plane Algorithms

CPA highlights two well-known benefits of cutting plane algorithms on empirical risk minimization problems: (i) scalability in the sample size; and (ii) control over data-related computation. Since cutting plane algorithms only use the training data to compute cut parameters, which can be achieved using elementary matrix-vector operations in O(Nd) time at each iteration, the running time of the algorithm scales linearly in N for fixed d (see Figure 3.1). Moreover, since cut parameters are computed in an isolated step (i.e., Step 3 in Algorithm 1), users can easily reduce data-related computation by, for instance, customizing their implementation to use parallel computing or techniques that exploit structural properties of their model class (see Section 5, where we do this for the risk score problem).

In addition, CPA also highlights a unique benefit of using cutting plane algorithms in our setting. Specifically, CPA recovers the optimal solution to the risk score problem by iteratively solving a linearized surrogate problem RiskSlimMIP. In practice, this means that we can fit risk scores with a MIP solver instead of a MINLP solver. As we show in Section 3.3, this can substantially improve our ability to solve RiskSlimMINLP since MIP solvers typically exhibit better off-the-shelf performance than MINLP solvers (due to the fact that MINLP solvers are designed to handle a diverse set of optimization problems, and that MIP solvers have better implementations of branch-and-bound).

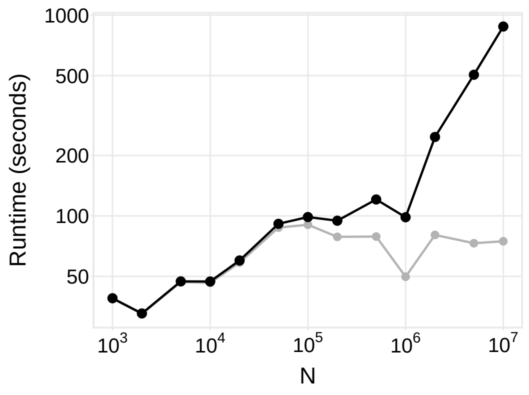

Figure 3: Runtime of CPA on simulated datasets with $d = 10$ and $N \in [10^3, 10^8]$ (see Appendix C for details). As N increases, solver time (grey) remains roughly constant. The total runtime (black) scales at $O(N)$, which reflects the additional time needed to compute cut parameters for datasets with large sample sizes.

Stalling of Cutting Plane Algorithms in Non-Convex Settings

Cutting plane algorithms used in empirical risk minimization (Joachims, 2006; Franc and Sonnenburg, 2008; Teo et al., 2009) are similar to CPA in that they solve a surrogate optimization problem at each iteration (e.g., on Step 5 of Algorithm 1). When these algorithms are used to solve convex problems, the surrogate problem is convex and therefore tractable. On problems with non-convex regularizers or constraints, however, the surrogate problem is non-convex and may require an unreasonable amount of time to solve to optimality, thereby preventing the algorithm from improving the approximate loss function and computing a valid lower bound. We refer to this behavior as *stalling*.

In Figure 4, we provide insight into the stalling behavior of CPA. Here, the first few iterations are quick as the surrogate problem is easy to solve to optimality when it contains a trivial approximation of the loss function. However, the surrogate becomes increasingly difficult to optimize with each iteration. On the $d = 10$ instance, CPA does not stall as the MIP solver is powerful enough to solve the surrogate problem RiskSlimMIP at all iterations. On the $d = 20$ instance, however, the time to solve RiskSlimMIP increases exponentially, leading CPA to stall at iteration $k = 86$. In this case, the best feasible solution has a large optimality gap *and* a highly suboptimal loss (as we would expect given that it was obtained by optimizing the approximation of a 20-dimensional function that uses at most 85 cuts). As a result, the solution obtained from CPA after 6 hours corresponds to a risk score model that has poor performance.

There is no easy fix to prevent cutting plane algorithms such as CPA from stalling on non-convex problems. This is because such algorithms need a provably optimal solution at each iteration to compute a valid lower bound and check convergence. In a non-convex setting, this requires pairing the optimal solution from each iteration with an optimality gap of 0.0%. If, for example, CPA only solved RiskSlimMIP until it found a feasible solution with a non-zero optimality gap, then the updated lower bound could exceed the true optimal value, leading the algorithm to terminate early and return a suboptimal solution with invalid bounds. Seeing how stalling is related to the mechanism to check convergence, a tempting (but flawed) solution is to use a cutting plane algorithm that adds cuts at central points of RiskSlimMIP (e.g., the center of gravity as in Levin 1965, or the analytic center as in Atkinson and Vaidya 1995), as these algorithms are guaranteed to converge in a fixed number of iterations and do not require computing a lower bound. Even in this case, however, stalling would still occur on high-dimensional problems as we would have to solve a non-convex optimization problem at each iteration to compute central points.

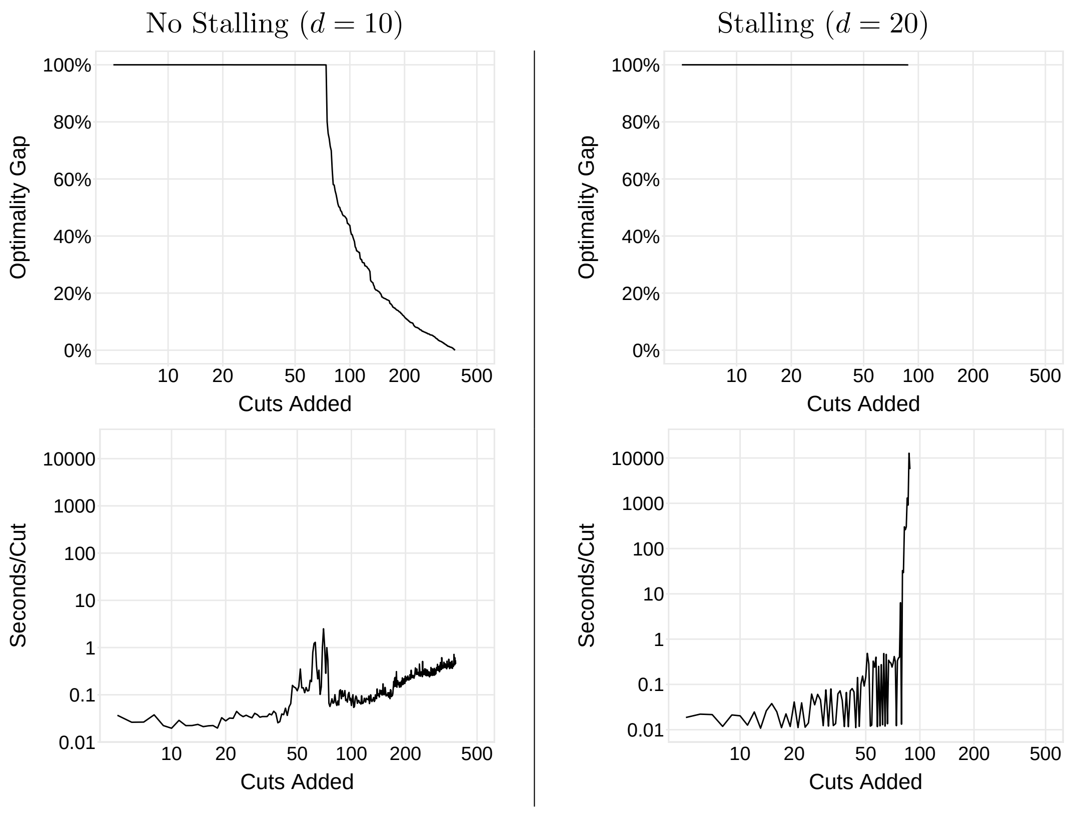

Figure 4: Progress of CPA on RiskSlimMINLP for simulated datasets with $d = 10$ (left) and $d = 20$ (right) and $N = 50{,}000$ (see Appendix C for details). We show the optimality gap (top) and time per iteration (bottom, in log-scale) for each iteration over 6 hours. CPA solves the $d = 10$ instance, but stalls on the $d = 20$ instance as the time to solve RiskSlimMIP to optimality increases exponentially starting on iteration 86, and the best solution obtained after 6 hours corresponds to a risk score with poor performance.

3.2 Lattice Cutting Plane Algorithm

In Algorithm 2, we present the cutting plane algorithm that we use to solve the risk score problem, which we refer to as the *lattice cutting plane algorithm* (LCPA).

LCPA builds a cutting plane approximation of the loss function while performing a branch-and-bound search. In particular, LCPA recovers the optimal solution to the risk score problem by solving a convex surrogate problem shown in (7) at various nodes of a branch-and-bound tree. The algorithm adds cuts at integer feasible solutions found during the search process, and uses a weaker lower bound that represents the smallest possible value of the surrogate over the remaining search region.

As shown in Figure 5, LCPA does not stall because, unlike CPA, it adds cuts and computes a lower bound without solving a non-convex problem to optimality. Even so, LCPA retains the key benefits of CPA for the risk score problem, namely: (i) the ability to use a MIP solver, since the surrogate problem minimizes a linear approximation of the loss function; (ii) scalability in the sample size, since the training data are only used to compute cut parameters; and (iii) control over data-related computation, since cut parameters can be computed outside of the MIP solver.

Although LCPA may appear to be more complicated than CPA, this is only because we show the steps of the branch-and-bound search in Algorithm 2. In practice, it easy to implement LCPA using modern MIP solvers (e.g., CPLEX) since they handle branch-and-bound and provide *control callbacks* to intervene in the search. In a basic implementation, for example, we use a control callback to intervene in Step 5 of Algorithm 2. We then only need to write the code to retrieve the integer feasible solution, compute cut parameters, and add a new cut in Step 7, returning control over to the MIP solver right before Step 9. As discussed in Section 4, our implementation uses control callbacks to run additional techniques that improve the performance of LCPA on the risk score problem.

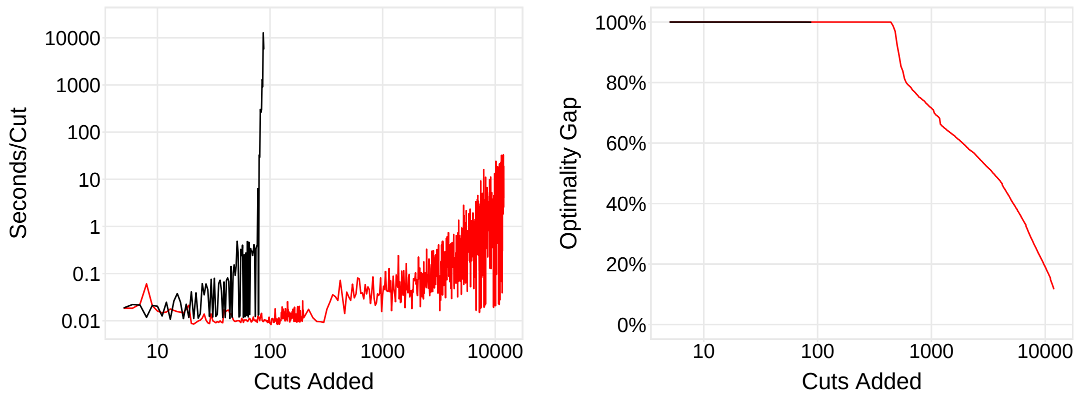

Figure 5: Progress of LCPA (red) and CPA (black) on the RiskSlimMINLP instance with $d = 20$ from Figure 4. Unlike CPA, LCPA does not stall. The algorithm finds a solution corresponding to a high-quality risk score in 9 minutes after adding 4655 cuts, and the optimal solution in 234 minutes after adding 11,665 cuts. The remaining time is used to reduce the optimality gap.

In what follows, we describe the major elements of Algorithm 2 in greater detail.

Surrogate Linear Program: LCPA recovers the optimal solution by repeatedly solving the surrogate linear program (LP). As shown in (7), the surrogate $\text{RiskSlimLP}(\hat{l}(\boldsymbol{\lambda}), \mathcal{P})$ is a LP relaxation of $\textsc{RiskSlimMIP}(\hat{l}(\boldsymbol{\lambda}))$ where the coefficients are restricted to a convex *partition* $\mathcal{P} \subseteq \operatorname{conv}(\mathcal{L})$.

Branch-and-Bound Search: LCPA repeatedly solves the surrogate LP at different nodes of a branch-and-bound tree. In Algorithm 2, we represent this tree as a *node set* $\mathcal{N} = (\mathcal{P}^t, V^t)_{t=1}^{t=|\mathcal{N}|}$ where each *node* $(\mathcal{P}^t, V^t)$ consists of a partition $\mathcal{P}^t \subseteq \operatorname{conv}(\mathcal{L})$, and a lower bound on the optimal value of the surrogate over this partition, V^t.

At each iteration, LCPA removes a node $(\mathcal{P}^t, V^t)$ from the node set $\mathcal{N}$ in Step 2, and solves the surrogate over the specified partition $\textsc{RiskSlimLP}(\hat{l}^k(\boldsymbol{\lambda}), \mathcal{P}^t)$ in Step 3. What happens next depends on the outcome of this step:

1. If $\textsc{RiskSlimLP}(\hat{l}^k(\boldsymbol{\lambda}), \mathcal{P}^t)$ yields a solution that is integer feasible $\boldsymbol{\lambda}^t \in \mathcal{L}$ (Step 5), then LCPA computes cut parameters at $\boldsymbol{\lambda}^t$, and adds a cut to improve the cutting plane approximation in the surrogate LP (Step 7).
2. If $\textsc{RiskSlimLP}(\hat{l}^k(\boldsymbol{\lambda}), \mathcal{P}^t)$ yields a solution that is not integer feasible $\boldsymbol{\lambda}^t \notin \mathcal{L}$ (Step 14), then LCPA splits $\mathcal{P}^t$ into two disjoint subsets $\mathcal{P}'$ and $\mathcal{P}''$. Each subset is paired with the optimal value of the surrogate problem over $\mathcal{P}^t$ to create two child nodes $(V^t, \mathcal{P}')$ and $(V^t, \mathcal{P}'')$. These child nodes are then added to the node set $\mathcal{N}$ (Step 17).
3. If $\textsc{RiskSlimLP}(\hat{l}^k(\boldsymbol{\lambda}), \mathcal{P}^t)$ is infeasible, then the node is discarded.

The search uses NodeSelectionRule to pick a node in Step 2, and BranchingRule to split a partition in Step 15. Both rules are typically provided by the MIP solver. NodeSelectionRule takes as input the node set $\mathcal{N}$ and returns one node $\mathcal{P}^t$ (e.g., the node with the smallest value of V^t). BranchingRule takes as input a partition $\mathcal{P}^t$ and returns as output two partitions $\mathcal{P}'$ and $\mathcal{P}''$ that are disjoint (i.e., $\mathcal{P}' \cap \mathcal{P}'' = \emptyset$), and that do not fully cover $\mathcal{P}^t$ (i.e., $\mathcal{P}' \cup \mathcal{P}'' \subset \mathcal{P}^t$)[2]. Thus, BranchingRule ensures that: (i) the partitions of all nodes in the node set are disjoint; (ii) the remaining search region shrinks even when the solution to the surrogate is not integer feasible; (iii) there are a finite number of nodes.

Bounds: LCPA checks convergence using bounds on the optimal value of the risk score problem. The upper bound $V^{\max}$ is set as the objective value of the best feasible solution in Step 11. The lower bound $V^{\min}$ is set as the smallest lower bound among nodes in the node set in Step 18. This quantity[3] is roughly equivalent to the objective value of $\textsc{RiskSlimLP}(\hat{l}^k(\boldsymbol{\lambda}), \bigcup_s \mathcal{P}^s)$ over the remaining search region $\bigcup_s \mathcal{P}^s$. Thus, $V^{\min}$ improves when we add more cuts to $\hat{l}^k(\boldsymbol{\lambda})$, or reduce the size of the remaining search region $\bigcup_s \mathcal{P}^s$.

Convergence: As LCPA progresses, each iteration reduces the remaining search region $\bigcup_s \mathcal{P}^s$ as it either finds an integer feasible solution, identifies an infeasible partition, or splits a given partition into disjoint subsets. As a result, the lower bound $V^{\min}$ increases

2. A common BranchingRule is to branch on a fractional component of the solution to $\textsc{RiskSlimLP}$ so that $\mathcal{P}^t$ is split into $\mathcal{P}' = \{\boldsymbol{\lambda} \in \mathcal{P}^t \mid \boldsymbol{\lambda}_j^t \geq \lceil \boldsymbol{\lambda}_j^t \rceil\}$ and $\mathcal{P}'' = \{\boldsymbol{\lambda} \in \mathcal{P}^t \mid \boldsymbol{\lambda}_j^t \leq \lceil \boldsymbol{\lambda}_j^t \rceil\}$.
3. Observe that LCPA lower bound is typically weaker than the CPA lower bound. Specifically, the CPA lower bound is built by solving a surrogate problem that requires an integer feasible solution while the LCPA lower bound is built by solving a surrogate problem that does not not require an integer feasible solution. As such, on instances where it does not stall, CPA may converge faster than the LCPA since it produces a stronger lower bound at each iteration.

monotonically as the search region becomes smaller, and cuts are added at integer feasible solutions. Likewise, the upper bound $V^{\max}$ decreases monotonically as it is set as the objective value of the best integer feasible solution, and the search is guaranteed to find the optimal solution. Since that there are a finite number of nodes in the worst-case, LCPA terminates after a finite number of iterations (see Remark 1) and returns an ε-optimal solution to the risk score problem.

Remark 1 (Convergence of Lattice Cutting Plane Algorithm)
Given any training dataset $(\boldsymbol{x}_i, y_i)_{i=1}^N$, any trade-off parameter $C_0 > 0$, and any finite coefficient set $\mathcal{L} \subset \mathbb{Z}^{d+1}$, Algorithm 2 returns an optimal solution to the risk score problem after computing at most $|\mathcal{L}|$ cutting planes and processing at most $2^{D(\mathcal{L})} - 1$ nodes, where $D(\mathcal{L}) = \prod_{j=0}^{d} (\Lambda_j^{max} - \Lambda_j^{min} + 1)$.

Proof See Appendix A. ■

Optional Improvements: LCPA can be substantially improved using techniques that we present in Section 4. In what follows, we describe these techniques at a high level.

Polishing Heuristic: We polish all integer feasible solutions that are found by the MIP solver in Step 5 using a technique that we call discrete coordinate descent (Algorithm 3; Section 4.1.1). Polished solutions may update the best solution found by LCPA, which results in stronger upper bounds over the course of LCPA, and reduces the time for LCPA to return a high-quality solution.

Rounding Heuristic: We produce new integer feasible solutions using a new rounding technique that we call sequential rounding (Algorithm 4; Section 4.1.2). Specifically, we round the continuous solution to the surrogate LP in Step 14, and polish the resulting integer feasible solution using discrete coordinate descent. Rounded solutions may improve the best solution found by LCPA, which produces stronger upper bounds over the course of LCPA, and reduces the time to find a high-quality solution.

Bounds on Objective Terms: We design a procedure to strengthen bounds on the optimal values of the objective function, loss function, and number of non-zero coefficients (Algorithm 5; Section 4.2.1). We call this procedure whenever the solver updates the upper bound in Step 11 or the lower bound in Step 18. Using this procedure improves the lower bound and the optimality gap over the course of LCPA.

Initialization Procedure: Since solution quality is affected by the fidelity of the approximate loss function, and LCPA only adds cuts at integer feasible solutions, early solutions from LCPA may correspond to low-quality models. In Section 4.2.2, we present an initialization procedure to mitigate this issue by quickly generating a set of cutting planes to warm-start LCPA. This reduces the time required for LCPA to return a high-quality solution, and improves both the upper and the lower bound over the course of LCPA.

Algorithm 2 Lattice Cutting Plane Algorithm (LCPA)

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^N$                    training data
    $\mathcal{L}$                                        constraint set for RiskSlimMINLP
    $C_0$                                                $\ell_0$ penalty parameter for RiskSlimMINLP
    $\varepsilon^{\text{stop}} \in [0,1]$                maximum optimality gap of acceptable solution
    NodeSelectionRule                                    rule to choose a node from a node set (provided by MIP solver)
    BranchingRule                                        rule to split a partition into two disjoint subsets (provided by MIP solver)

Initialize
    $k \leftarrow 0$                                     number of cuts added
    $\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$   initial approximation of loss function
    $(V^{\min}, V^{\max}) \leftarrow (0, \infty)$        bounds on the optimal value of RiskSlimMINLP
    $\varepsilon \leftarrow \infty$                      optimality gap of current best solution to RiskSlimMINLP
    $\mathcal{P}^0 \leftarrow \text{conv}(\mathcal{L})$  partition for initial node
    $\mathcal{N} \leftarrow \{(\mathcal{P}^0, V^{\min})\}$   node set

 1: while $\varepsilon > \varepsilon^{\text{stop}}$ do
 2:     apply NodeSelectionRule to remove a node $(\mathcal{P}^t, V^t)$ from $\mathcal{N}$
 3:     solve RiskSlimLP$(\hat{l}^k(\boldsymbol{\lambda}), \mathcal{P}^t)$
 4:     if optimal solution exists and is integer feasible then
 5:         $\boldsymbol{\lambda}^t \leftarrow$ coefficients from optimal solution to RiskSlimLP$(\hat{l}^k(\boldsymbol{\lambda}), \mathcal{P}^t)$
 6:         compute cut parameters $l(\boldsymbol{\lambda}^t)$ and $\nabla l(\boldsymbol{\lambda}^t)$
 7:         $\hat{l}^{k+1}(\boldsymbol{\lambda}) \leftarrow \max\{\hat{l}^k(\boldsymbol{\lambda}), l(\boldsymbol{\lambda}^k) + \langle \nabla l(\boldsymbol{\lambda}^k), \boldsymbol{\lambda} - \boldsymbol{\lambda}^k \rangle\}$ for all $\boldsymbol{\lambda}$; $k \leftarrow k+1$    ▷add cut
 8:         $v \leftarrow l(\boldsymbol{\lambda}^t) + C_0 \|\boldsymbol{\lambda}^t\|_0$
 9:         if $v < V^{\max}$ then
10:             $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^t$    ▷update incumbent solution
11:             $V^{\max} \leftarrow v$    ▷tighten upper bound
12:             $\mathcal{N} \leftarrow \mathcal{N} \setminus \{(\mathcal{P}^s, V^s) \mid V^s \geq V^{\max}\}$    ▷prune suboptimal nodes
13:         end if
14:     else if optimal solution exists but is not integer feasible then
15:         $(\mathcal{P}', \mathcal{P}'') \leftarrow$ BranchingRule$(\mathcal{P}^t)$    ▷$\mathcal{P}'$, $\mathcal{P}''$ are disjoint subsets of $\mathcal{P}^t$
16:         $V^t \leftarrow$ optimal value of RiskSlimLP$(\hat{l}^k(\boldsymbol{\lambda}), \mathcal{P}^t)$    ▷$V^t$ is lower bound for optimal value over $\mathcal{P}', \mathcal{P}''$
17:         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(\mathcal{P}', V^t), (\mathcal{P}'', V^t)\}$    ▷add child nodes to $\mathcal{N}$
18:         $V^{\min} \leftarrow \min_{(\mathcal{P}, V) \in \mathcal{N}} V$    ▷global lower bound is smallest lower bound from all nodes in $\mathcal{N}$
19:     end if
20:     $\varepsilon \leftarrow 1 - V^{\min}/V^{\max}$    ▷update optimality gap
21: end while
Output: $\boldsymbol{\lambda}^{\text{best}}$, which is an $\varepsilon$-optimal solution to RiskSlimMINLP
```

RiskSlimLP$(\hat{l}(\boldsymbol{\lambda}), \mathcal{P})$ is the LP relaxation of RiskSlimMIP$(\hat{l}(\boldsymbol{\lambda}))$ over the partition $\mathcal{P} \subseteq \text{conv}(\mathcal{L})$:

$$\begin{aligned} \min_{\theta, \boldsymbol{\lambda}, \boldsymbol{\alpha}} \quad & \theta + C_0 \sum_{j=1}^{d} \alpha_j \\ \text{s.t.} \quad & \theta \geq \hat{l}(\boldsymbol{\lambda}) \\ & \boldsymbol{\lambda} \in \mathcal{P} \\ & \alpha_j = \max(\lambda_j, 0)/\Lambda_j^{\max} + \min(\lambda_j, 0)/\Lambda_j^{\min} \quad \text{for } j = 1, \ldots, d. \end{aligned} \tag{7}$$

We provide an LP formulation for RiskSlimLP in Appendix B.

3.3 Empirical Performance of Algorithms to Solve the Risk Score Problem

In order to illustrate the performance of CPA and LCPA, we used each algorithm to solve difficult instances of RiskSlimMINLP on simulated datasets with varying dimensions d and sample sizes N (see Appendix C for details). As a baseline, we solved the same instances using 3 MINLP algorithms[4] as implemented commercial MINLP solver[5]. In Figure 6, we compare all methods in terms of the following metrics:

- *time to find a near-optimal solution*: that is, a solution whose loss is within 10% of the optimal loss. This measures the time needed to fit a risk score with good AUC and risk calibration, but without a proof of optimality.
- *optimality gap of best solution at termination*, which is 0.0% if and only if the method finds the optimal solution *and* provides the proof of optimality within a 6 hour time limit.
- *% of time spent on data-related computation*, meaning the time spent evaluating the value/gradient/Hessian of the loss function.

As shown, a basic implementation of LCPA finds the optimal solution to RiskSlimMINLP for almost all instances, and pairs this solution with a small optimality gap. Note that this performance reflects a basic implementation of LCPA on difficult instances of RiskSlimMINLP. We improve the performance of LCPA with regards to these metrics using specialized techniques in Sections 4 and 5, and use LCPA to recover the optimal solution to larger real-world problems in Sections **??** and 6.

CPA performs similarly to LCPA on low-dimensional instances. On instances with $d \geq 15$, however, CPA stalls after a few iterations and ultimately returns a highly suboptimal solution that corresponds to a risk score with poor performance.

In comparison to the cutting plane algorithms, the MINLP solver could only handle instances of RiskSlimMINLP for datasets with limited sample sizes and/or dimensions – regardless of the algorithm that we used to solve the problem. On large instances, the solver spends the majority of its time dealing with operations that involve data-related computation, fails to converge in the 6-hour time limit, and fails to yield high quality feasible solution. Seeing how a MINLP solver is designed to solve a diverse set of optimization problems, it is unlikely that it can identify and exploit the structure of the risk score problem in the same way as the cutting plane algorithms.

4. Since all 3 MINLP algorithms behave similarly, we only show the best performing algorithm in Figure 6 (i.e., ActiveSetMINLP). We include results for the remaining MINLP algorithms in Appendix C.
5. There exist several off-the-shelf MINLP solvers (see Bussieck and Vigerske, 2010, for a list). We used Artelsys Knitro 9.0 (i.e., an updated version of the MINLP solver from Byrd et al., 2006) because it let us: (i) monitor and minimize data-related computation, by letting us write our own functions to evaluate the objective, its gradient and Hessian; and (ii) solve LP subproblems using CPLEX, which is the same solver used in CPA and LCPA.

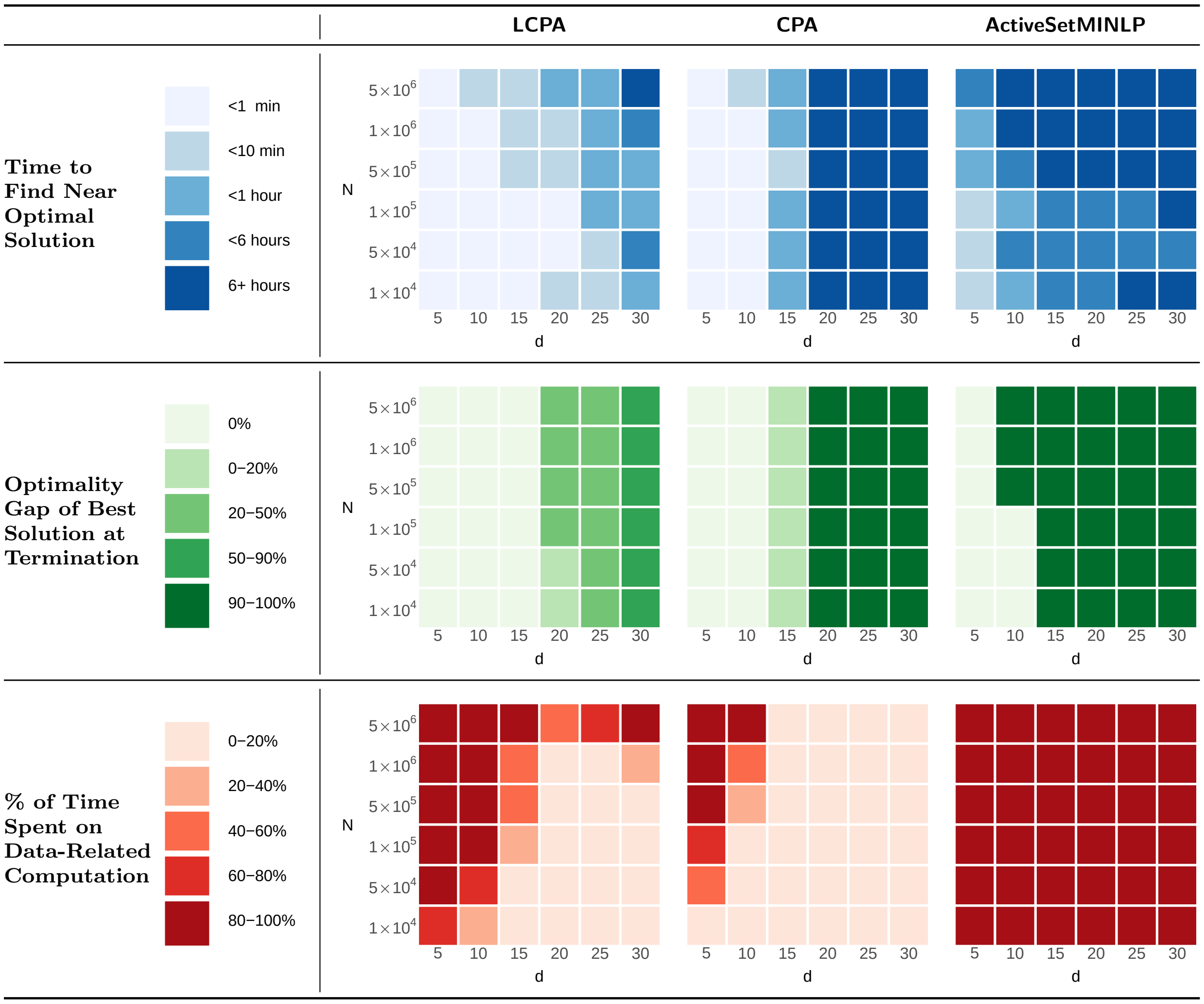

Figure 6: Performance of algorithms on difficult instances of RiskSlimMINLP for simulated datasets with varying dimensions d and sample sizes N (see Appendix C for details). ActiveSetMINLP fails on instances with large d or N as it struggles with data-related computation. In comparison, CPA and LCPA scale linearly in N – that is, if they can solve an instance for fixed d, then they can solve instances for larger N in $O(N)$ time. CPA stalls on all instances when $d \geq 15$ and returns highly suboptimal solutions when $d \geq 20$. In contrast, LCPA does not stall on any instances, and recovers a near optimal solution in all cases, pairing them with optimality gaps between 0.0 - 62.2% depending on d. Results for LCPA reflect the performance for basic implementation without the improvements in Sections 4 and 5. We include results for additional MINLP algorithms in Appendix C as they perform similarly to ActiveSetMINLP.

4. Algorithmic Improvements

In this section, we describe specialized techniques to improve the performance of the lattice cutting plane algorithm (LCPA) on the risk score problem (RiskSlimMINLP).

4.1 Generating High Quality Feasible Solutions

In what follows, we present two techniques to generate and improve integer feasible solutions for RiskSlimMINLP. We pair these techniques with LCPA to produce new integer solutions by rounding continuous solutions and to polish integer feasible solutions obtained by rounding, or found by the MIP solver. In addition, we make use these techniques in new heuristic methods to learn risk scores in Sections **??** and 6.

4.1.1 Discrete Coordinate Descent

Discrete coordinate descent (DCD) is a technique for polishing integer feasible solutions (Algorithm 3). It takes as input an integer feasible solution $\boldsymbol{\lambda} = (\lambda_0, \ldots, \lambda_d) \in \mathcal{L}$ and iteratively moves along a single dimension j to attain an integer feasible solution with a lower objective value. The descent direction at each iteration is chosen greedily as the dimension that minimizes the objective value $j \in \operatorname{argmin} V(\boldsymbol{\lambda} + \delta_j e_j)$.

DCD terminates once it can no longer strictly improve the the objective value along any dimension. This eliminates the possibility of cycling, and ensures that it terminates after a finite number of iterations. The polished solution satisfies a type of local optimality guarantee in the discrete setting: formally, the solution is *1-opt* with respect to the objective, meaning that the objective cannot improve in any single dimension (see e.g., Park and Boyd, 2015b, for a technique to find a 1-opt point for a different optimization problem).

In practice, the most expensive part of DCD involves determining a step-size $\delta_j \in \Delta_j$ that minimizes the objective in dimension j (Step 4 of Algorithm 3). The computation for this step can be significantly reduced by using a bisection search algorithm that exploits the convexity of the loss function, to reduce the number of loss function evaluations. This approach requires a total of $\log_2(|\mathcal{L}_j|)Nd$ flops per iteration, which is an improvement over the $(|\mathcal{L}_j| - 1)Nd$ flops per iteration required from the naïve exhaustive search strategy (i.e., where we evaluate the loss for all $|\mathcal{L}_j| - 1$ feasible values of λ_j other than current value).

In Figure 7, we show how DCD can improve the performance of LCPA when we use it to polish feasible solutions found by the MIP solver (i.e., in Step 5 of Algorithm 2).

Algorithm 3 Discrete Coordinate Descent (DCD)

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^N$                      training data
    $\mathcal{L}$                                          constraint set for RiskSlimMINLP
    $C_0$                                                  $\ell_0$ penalty parameter for RiskSlimMINLP
    $\boldsymbol{\lambda} \in \mathcal{L}$                 integer feasible solution to RiskSlimMINLP

Initialize
    $V \leftarrow V(\boldsymbol{\lambda})$                 objective value at current solution
    $\mathcal{J} \leftarrow \{0, \ldots, d\}$              valid search dimensions
1:  repeat
2:      for $j \in \mathcal{J}$ do
3:          $\Delta_j \leftarrow \{\delta \in \mathbb{Z} \mid \boldsymbol{\lambda} + \delta e_j \in \mathcal{L}\}$      ▷list feasible moves along dim $j$
4:          $\delta_j \leftarrow \mathrm{argmin}_{\delta \in \Delta_j} V(\boldsymbol{\lambda} + \delta)$      ▷find best move along dim $j$
5:          $v_j \leftarrow V(\boldsymbol{\lambda} + \delta_j e_j)$      ▷store objective value for best move along dim $j$
6:      end for
7:      $m \leftarrow \mathrm{argmin}_{j \in \mathcal{J}} v_j$      ▷descend along dim that minimizes objective
8:      if $v_m < V$ then
9:          $V \leftarrow v_m$
10:         $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \delta_m e_m$
11:         $\mathcal{J} \leftarrow \{0, \ldots, d\} \setminus \{m\}$      ▷ignore dim $m$ on next iteration
12:     end if
13: until $v_m \geq V$
Output: $\boldsymbol{\lambda}$, solution that is 1-opt with respect to the objective of RiskSlimMINLP
```

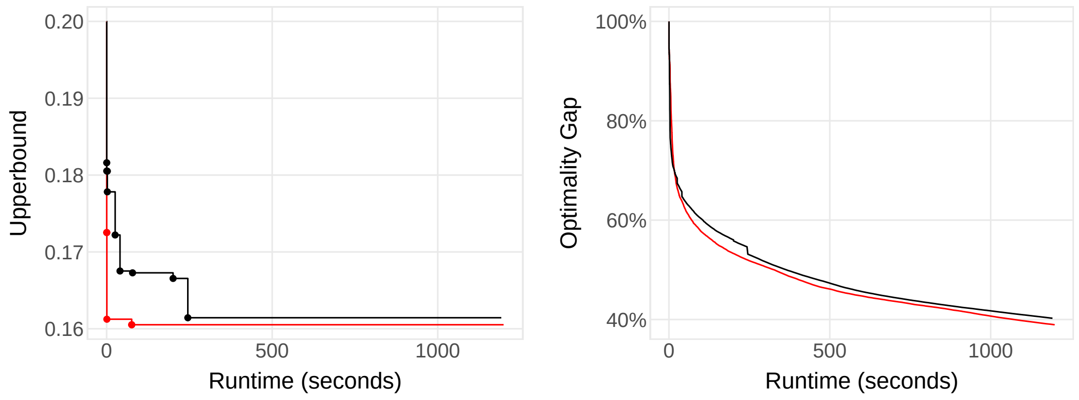

Figure 7: Performance of LCPA in a basic implementation (black) and with DCD (red). We use DCD to polish every integer feasible solution found by the MIP solver whose objective value is within 10% of the current upper bound. We plot large points to show when LCPA updates the incumbent solution. Results reflect performance on RiskSlimMINLP for a simulated dataset with $d = 30$ and $N = 50{,}000$ (see Appendix C for details).

4.1.2 Sequential Rounding

Sequential rounding (Algorithm 4) is a technique to round continuous solutions in a way that accounts for objective of RiskSlimMINLP. Given a continuous solution $\boldsymbol{\lambda}^{\text{cts}} \in \operatorname{conv}(\mathcal{L})$, SequentialRounding rounds one component at a time, either up or down, in a way that minimizes the objective value of RiskSlimMINLP. In comparison to naïve rounding, which returns the closest rounding from a set of 2^{d+1} possible roundings, SequentialRounding returns a rounding that optimizes the value of the objective function.

On Step k, the technique has already rounded k components, and needs to round one of the remaining $d-k+1$ components to either $\lceil \lambda_j^{\text{cts}} \rceil$ or $\lfloor \lambda_j^{\text{cts}} \rfloor$. To this end, it computes the objective value of each feasible component-direction pair, and chooses the best one. The minimization on Step k requires $\sum_{i=1}^{i=d-k+1} 2i = (d-k+1)(d-k+2)$ evaluations of the loss function. Thus, given that there are $d+1$ steps, the technique terminates after $\frac{1}{3}d(d^2+3d+2)$ evaluations of the loss function.

In Figure 8, we show the impact of using SequentialRounding in LCPA to round the continuous solution to RiskSlimLP when the lower bound changes (i.e., Step 3 of Algorithm 2). We then polish the rounded solution via DCD to increase the likelihood that it will update the incumbent solution. As shown, this strategy reduces the time needed for LCPA to produce a higher quality risk score, and attain a lower optimality gap.

Algorithm 4 Sequential Rounding (SequentialRounding)

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^N$                         training data
    $\mathcal{L}$                                   constraint set for RiskSlimMINLP
    $C_0$                                  $\ell_0$ penalty parameter for RiskSlimMINLP
    $\boldsymbol{\lambda} \in \operatorname{conv}(\mathcal{L})$                        feasible solution to RiskSlimLP

Initialize
    $\mathcal{J}^{\text{cts}} \leftarrow \{0, \dots, d\}$          index set of features that need to be rounded
 1: repeat
 2:     $\boldsymbol{\lambda}^{\text{floor}(j)} \leftarrow (\lambda_1, \dots, \lfloor \lambda_j \rfloor, \dots, \lambda_d)$ for all $j \in \mathcal{J}^{\text{cts}}$
 3:     $\boldsymbol{\lambda}^{\text{ceil}(j)} \leftarrow (\lambda_1, \dots, \lceil \lambda_j \rceil, \dots, \lambda_d)$ for all $j \in \mathcal{J}^{\text{cts}}$
 4:     $v^{\text{floor}} \leftarrow \min_{j \in \mathcal{J}^{\text{cts}}} V(\boldsymbol{\lambda}^{\text{floor}(j)})$
 5:     $v^{\text{ceil}} \leftarrow \min_{j \in \mathcal{J}^{\text{cts}}} V(\boldsymbol{\lambda}^{\text{ceil}(j)})$
 6:     if $v^{\text{floor}} \leq v^{\text{ceil}}$ then
 7:         $k \leftarrow \operatorname{argmin}_{j \in \mathcal{J}} V(\boldsymbol{\lambda}^{\text{floor}(j)})$
 8:         $\lambda_k \leftarrow \lfloor \lambda_k \rfloor$
 9:     else
10:         $k \leftarrow \operatorname{argmin}_{j \in \mathcal{J}} V(\boldsymbol{\lambda}^{\text{ceil}(j)})$
11:         $\lambda_k \leftarrow \lceil \lambda_k \rceil$
12:     end if
13:     $\mathcal{J}^{\text{cts}} \leftarrow \mathcal{J}^{\text{cts}} \setminus \{k\}$
14: until $\mathcal{J}^{\text{cts}} = \varnothing$
Output: $\boldsymbol{\lambda} \in \mathcal{L}$, integer feasible solution
```

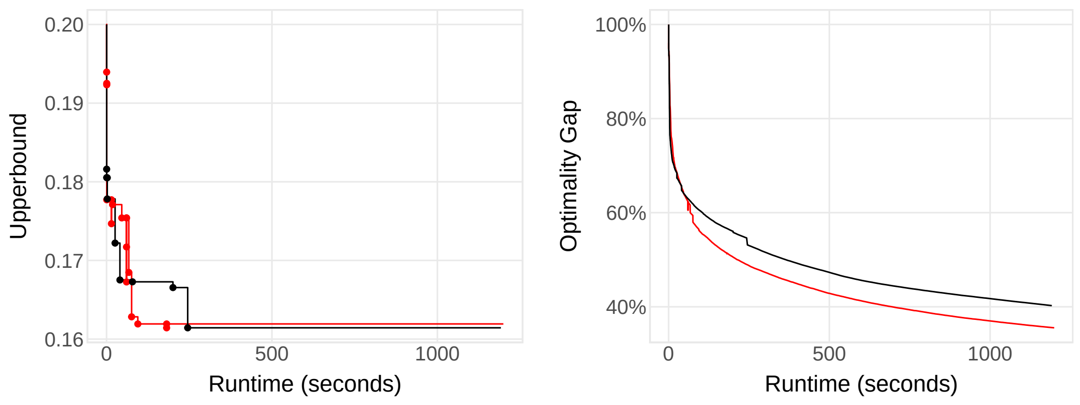

Figure 8: Performance of LCPA in a basic implementation (black) and with sequential rounding then polishing (red). We call SequentialRounding in Step 14 for continuous solutions to RiskSlimLP, and the integer feasible solution using DCD. We plot large points to show when LCPA updates the incumbent solution. Results reflect performance on RiskSlim-MINLP for a simulated dataset with $d = 30$ and $N = 50{,}000$ (see Appendix C for details).

4.2 Reducing the Optimality Gap

We now present two techniques to reduce the optimality gap produced by LCPA. These techniques require that we run LCPA with an *augmented* formulation of RiskSlimLP. We show augmented formulation of RiskSlimMIP in (8), which can be relaxed to obtain the augmented formulation of RiskSlimLP in (9) (see Appendix B for IP formulations). Both formulations include auxiliary variables and constraints to bound the values of the objective, loss, and ℓ_0-norm at the optimal solution $\boldsymbol{\lambda}^*$. Our techniques are designed to run LCPA using an augmented formulation of RiskSlimLP where these bounds as strong as possible, but still valid. In doing so, they effectively restrict the search region without discarding the optimal solution, thereby improving the lower bound and reducing the optimality gap.

AugmentedRiskSlimMIP $\left(\hat{l}(\boldsymbol{\lambda}), \mathcal{L}, V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}\right)$

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & V \\
\text{s.t.} \quad & \theta \geq \hat{l}(\boldsymbol{\lambda}) \\
& \boldsymbol{\lambda} \in \mathcal{L} \\
& V = \theta + C_0 R \\
& R = \|\boldsymbol{\lambda}\|_0 \\
& V \in [V^{\min}, V^{\max}] \\
& \theta \in [L^{\min}, L^{\max}] \\
& R \in \{R^{\min}, \ldots, R^{\max}\}.
\end{aligned} \tag{8}$$

AugmentedRiskSlimLP $\left(\hat{l}(\boldsymbol{\lambda}), \mathcal{P}, V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}\right)$

$$\begin{aligned}
\min_{\boldsymbol{\lambda}} \quad & V \\
\text{s.t.} \quad & \theta \geq \hat{l}(\boldsymbol{\lambda}) \\
& \boldsymbol{\lambda} \in \mathcal{P} \\
& V = \theta + C_0 R \\
& R = \sum_{j=1}^{d} \alpha_j \\
& \alpha_j = \max(\lambda_j, 0)/\Lambda_j^{\max} + \min(\lambda_j, 0)/\Lambda_j^{\min} \text{ for } j = 1, \ldots, d. \\
& V \in [V^{\min}, V^{\max}] \\
& \theta \in [L^{\min}, L^{\max}] \\
& R \in [R^{\min}, R^{\max}].
\end{aligned} \tag{9}$$

Note that we implicitly assume $\Lambda_j^{\min}$ is negative and $\Lambda_j^{\max}$ is positive. In what follows, we will discuss how to set the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$.

4.2.1 Chained Updates

Chained updates (Algorithm 5) is a simple procedure to strengthen the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, and $R^{\max}$ over the course of LCPA. It requires no assumptions, is easy to implement, and involves minimal computation.

Initial Bounds on Objective Terms

To initialize the procedure, we need bounds that can be computed using only the training data $(\boldsymbol{x}_i, y_i)_{i=1}^N$ and the coefficient set $\mathcal{L}$. We start with Proposition 1, which bounds on the logistic loss by exploiting the fact that $\mathcal{L}$ is bounded (see Appendix A for a proof).

Proposition 1 (Bounds on Logistic Loss over a Bounded Coefficient Set) *Let $(\boldsymbol{x}_i, y_i)_{i=1}^N$ denote a dataset where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ for $i = 1, \ldots, N$. Consider the value of the normalized logistic loss for a linear classifier with coefficients $\boldsymbol{\lambda} \in \mathcal{L} \subset \mathbb{R}^d$*

$$l(\boldsymbol{\lambda}) = \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-\langle \boldsymbol{\lambda}, y_i \boldsymbol{x}_i \rangle)).$$

If the coefficient set $\mathcal{L}$ is bounded, then $l(\boldsymbol{\lambda}) \in [L^{\min}, L^{\max}]$ for all $\boldsymbol{\lambda} \in \mathcal{L}$ where

$$\begin{aligned}
L^{\min} &= \frac{1}{N} \sum_{i:y_i=+1} \log\left(1 + \exp(-s_i^{\max})\right) + \frac{1}{N} \sum_{i:y_i=-1} \log\left(1 + \exp(s_i^{\min})\right), \\
L^{\max} &= \frac{1}{N} \sum_{i:y_i=+1} \log\left(1 + \exp(-s_i^{\min})\right) + \frac{1}{N} \sum_{i:y_i=-1} \log\left(1 + \exp(s_i^{\max})\right), \\
s_i^{\min} &= \min_{\boldsymbol{\lambda} \in \mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle \text{ for } i = 1, \ldots, N, \\
s_i^{\max} &= \max_{\boldsymbol{\lambda} \in \mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle \text{ for } i = 1, \ldots, N.
\end{aligned}$$

The value of $L^{\min}$ in Proposition 1 represents the best case loss in a perfectly separable setting when we assign each positive example the largest possible score $s_i^{\max}$, and each negative example the smallest possible score $s_i^{\min}$. Conversely, $L^{\max}$ represents the worst case loss when we assign each positive example the smallest score $s_i^{\min}$, and each negative example the largest score $s_i^{\max}$. Both $L^{\min}$ and $L^{\max}$ can be computed in $\mathrm{O}(N)$ flops using only the training data and the coefficient set by evaluating $s_i^{\min}$ and $s_i^{\max}$ as follows:

$$s_i^{\min} = \min_{\boldsymbol{\lambda}} \sum_{j=0}^{d} x_{ij}\lambda_j = \sum_{j=0}^{d} \min_{\lambda_j} x_{ij}\boldsymbol{\lambda}_j = \sum_{j=0}^{d} \mathbb{1}\left[x_{ij} > 0\right] x_{ij}\Lambda_j^{\min} + \mathbb{1}\left[x_{ij} < 0\right] x_{ij}\Lambda_j^{\max}, \quad (10)$$

$$s_i^{\max} = \max_{\boldsymbol{\lambda}} \sum_{j=0}^{d} x_{ij}\lambda_j = \sum_{j=0}^{d} \max_{\boldsymbol{\lambda}_j} x_{ij} = \sum_{j=0}^{d} \mathbb{1}\left[x_{ij} > 0\right] x_{ij}\Lambda_j^{\max} + \mathbb{1}\left[x_{ij} < 0\right] x_{ij}\Lambda_j^{\min}. \quad (11)$$

The values of $L^{\min}$ and $L^{\max}$ may be strengthened when we have a non-trivial limit on the number of features (i.e., $R^{\max} < d$).

We set the initial bounds for the number of non-zero coefficients R to $[0, d]$, trivially. In some cases, however, these bounds are stronger since users limit the number of non-zero coefficients (e.g., if we wish to fit models with at most 5 features, then $R \in [0, 5]$). In this case, the values of $L^{\min}$ and $L^{\max}$ are further restricted since $s_i^{\min}$ and $s_i^{\max}$ are also bounded by the number of non-zero coefficients. Here, $s_i^{\min}$ or $s_i^{\max}$ can still be computed efficiently in $\mathrm{O}(N)$ flops by choosing the $R^{\max}$ smallest or largest terms in the left hand side of equation (10) or (11). Having initialized $L^{\min}$, $L^{\max}$, $R^{\min}$ and $R^{\max}$, we can set the bounds on the optimal objective value as $V^{\min} = L^{\min} + C_0 R^{\min}$ and $V^{\max} = L^{\max} + C_0 R^{\max}$, respectively.

Dynamic Bounds on Objective Terms

Propositions 2–4 can strengthen the initial values of $L^{\min}$, $L^{\max}$, $R^{\max}$, $V^{\min}$ and $V^{\max}$ using information provided by the MIP solver in LCPA (see Appendix A for proofs).

Proposition 2 (Upper Bound on Optimal Number of Non-Zero Coefficients)
Given an upper bound on the optimal objective value $V^{\max} \geq V(\boldsymbol{\lambda}^*)$, and a lower bound on the optimal loss $L^{\min} \leq l(\boldsymbol{\lambda}^*)$, we can derive an upper bound on the optimal number of non-zero coefficients $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$ as

$$R^{\max} = \left\lfloor \frac{V^{\max} - L^{\min}}{C_0} \right\rfloor.$$

Proposition 3 (Upper Bound on Optimal Loss)
Given an upper bound on the optimal objective value $V^{\max} \geq V(\boldsymbol{\lambda}^*)$, and a lower bound on the optimal number of non-zero coefficients $R^{\min} \leq \|\boldsymbol{\lambda}^*\|_0$, we can derive an upper bound on the optimal loss $L^{\max} \geq l(\boldsymbol{\lambda}^*)$ as

$$L^{\max} = V^{\max} - C_0 R^{\min}.$$

Proposition 4 (Lower Bound on Optimal Loss)
Given a lower bound on the optimal objective value $V^{\min} \leq V(\boldsymbol{\lambda}^*)$, and an upper bound on the optimal number of non-zero coefficients $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$, we can derive a lower bound on value of the loss function $L^{\min} \leq l(\boldsymbol{\lambda}^*)$ as

$$L^{\min} = V^{\min} - C_0 R^{\max}.$$

Chained Updates Procedure

In Algorithm 5, we present a simple procedure that uses Propositions 2–4 to strengthen the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, and $R^{\max}$ in the augmented formulation of RiskSlimLP.

Algorithm 5 Chained Updates for LCPA (ChainedUpdates)

```
Input
    $C_0$                                                   $\ell_0$ penalty parameter for RiskSlimMINLP
    $V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}$          initial bounds on $V(\boldsymbol{\lambda}^*)$, $l(\boldsymbol{\lambda}^*)$ and $\|\boldsymbol{\lambda}^*\|_0$

1: repeat
2:     $V^{\min} \leftarrow \max\left(V^{\min},\ L^{\min} + C_0 R^{\min}\right)$          ▷update lower bound on $V(\boldsymbol{\lambda}^*)$
3:     $V^{\max} \leftarrow \min\left(V^{\max},\ L^{\max} + C_0 R^{\max}\right)$          ▷update upper bound on $V(\boldsymbol{\lambda}^*)$
4:     $L^{\min} \leftarrow \max\left(L^{\min},\ V^{\min} - C_0 R^{\max}\right)$          ▷update lower bound on $l(\boldsymbol{\lambda}^*)$
5:     $L^{\max} \leftarrow \min\left(L^{\max},\ V^{\max} - C_0 R^{\min}\right)$          ▷update upper bound on $l(\boldsymbol{\lambda}^*)$
6:     $R^{\max} \leftarrow \min\left(R^{\max},\ \left\lfloor \frac{V^{\max} - L^{\min}}{C_0} \right\rfloor\right)$          ▷update upper bound on $\|\boldsymbol{\lambda}^*\|_0$
7: until there are no more bound updates due to Steps 2 to 6.
Output: $V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}$
```

Propositions 2–4 impose dependencies between the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, $R^{\min}$ and $R^{\max}$ that may lead to a complex "chain" of updates. As shown in Figure 9, it may be possible to update more than one value, and update some values multiple times. Consider a case where we call the procedure once our MIP solver improves the lower bound on the objective value $V^{\min}$. If it updates $L^{\min}$ on Step 4, but does not update $R^{\max}$ in Step 6, then it will not update $V^{\max}$, $L^{\min}$, $L^{\max}$, $V^{\min}$ at the next iteration. However, if $R^{\max}$ is updated after rounding, then $V^{\max}$, $L^{\min}$, $L^{\max}$, $V^{\min}$ will be updated.

In light of these dependencies, Algorithm 5 cycles through Propositions 2–4 until it cannot update any of the values of $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, and $R^{\max}$. This ensures that Algorithm 5 returns the strongest possible bounds, regardless of the term that was first updated, In addition, it allows us to call Algorithm 5 in other settings, such as the initialization procedure in Section 4.2.2.

In Figure 10, we show how the chained updates procedure improves the lower bound and optimality gap produced over the course of LCPA. Here, we call Algorithm 5 whenever LCPA updates $V^{\max}$ in Step 11, or $V^{\min}$ in Step 18. If the bounds on the objective value terms improve as a result of this call, we pass this information to the MIP solver by updating the bounds in the augmented formulation shown in Appendix B.

4.2.2 Initialization Procedure

In Algorithm 6, we present an initialization procedure to kick-start LCPA with a high quality feasible solution, a set of initial cutting planes, and strong bounds on the values of the objective, loss, and number of non-zero coefficients. The procedure uses all techniques presented so far, as follows:

1. *Run CPA on* RiskSlimLP: We apply CPA to solve the surrogate LP, RiskSlimLP until a time limit is reached (or any other user-specified stopping condition). We store the

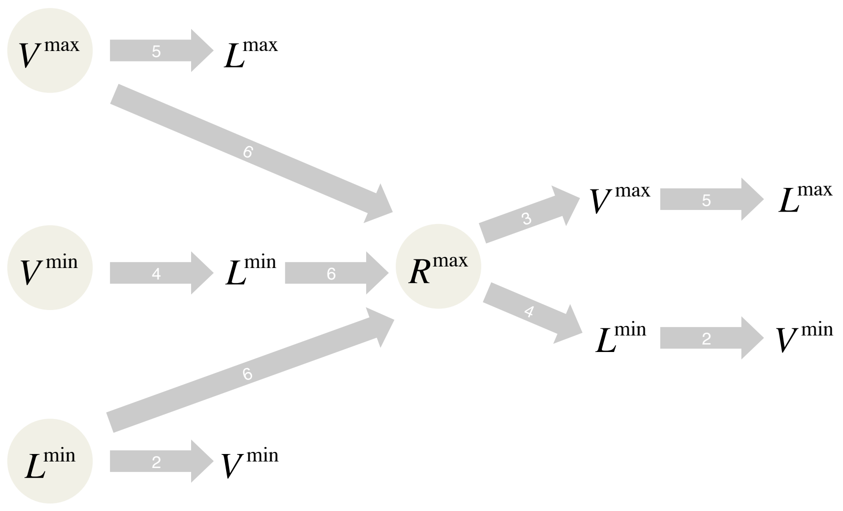

Figure 9: All possible update "chains" in Algorithm 5. We circle terms that can be updated externally by the MIP solver in LCPA. When any of these terms is updated, we run Algorithm 5 and potentially strengthen all terms in the outward path marked by the arrows. The numbers inside each arrow refer to the relevant step in Algorithm 5.

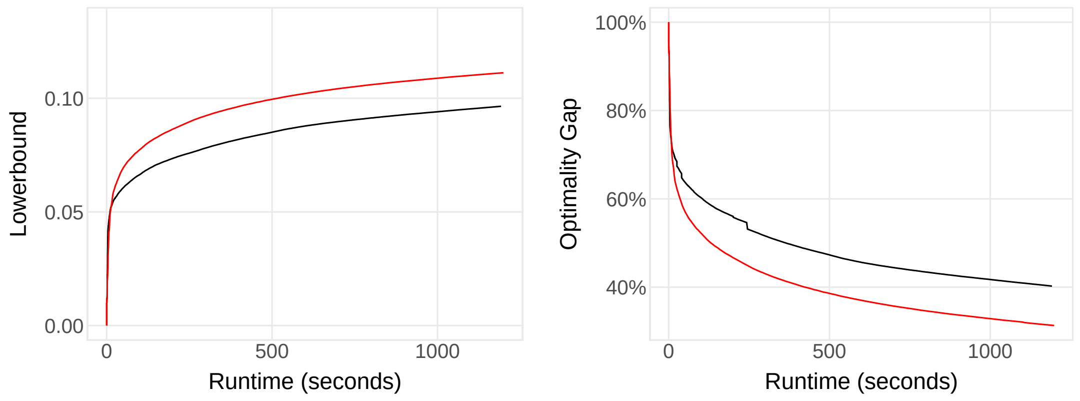

Figure 10: Performance of LCPA in a basic implementation (black) and with ChainedUpdates (red). Results reflect performance on an RiskSlimMINLP instance for a simulated dataset with $d = 30$ and $N = 50{,}000$ (see Appendix C for details).

cuts from RiskSlimLP to initialize LCPA, and record the lower bound from CPA on the objective value of RiskSlimLP as it is also a lower bound on the optimal value of RiskSlimMINLP.

2. *Sequential Rounding and Polishing*: We collect the solutions produced at each iteration of CPA. For each solution, we run SequentialRounding (Algorithm 4) to obtain an integer feasible solution for RiskSlimMINLP. We then polish this solution using DCD (Algorithm 3). We use the best solution to update the upper bound on the optimal value to RiskSlimMINLP.

3. *Chained Updates*: Having obtained strong bounds on $V^{\min}$ and $V^{\max}$, we update all bounds using ChainedUpdates (Algorithm 5).

Algorithm 6 Initialization Procedure for LCPA

```
Input
    $(\boldsymbol{x}_i, y_i)_{i=1}^N$                                   training data
    $\mathcal{L}$                                          constraint set for RiskSlimMINLP
    $C_0$                                          $\ell_0$ penalty parameter for RiskSlimMINLP
    $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, $R^{\min}$, $R^{\max}$    initial bounds on $V(\boldsymbol{\lambda}^*)$, $l(\boldsymbol{\lambda}^*)$ and $\|\boldsymbol{\lambda}^*\|_0$
    $T^{\max}$                                     time limit for CPA on RiskSlimLP

Initialize
    $\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$                       initial approximation of loss function

Step I: Solve RiskSlimLP with CPA
 1: Solve RiskSlimLP$(\hat{l}^0(\boldsymbol{\lambda}), \mathrm{conv}(\mathcal{L}))$ using CPA      ▷Algorithm 1
 2: $k \leftarrow$ number of CPA iterations completed in $T^{\max}$
 3: $\hat{l}^{\mathrm{initial}}(\boldsymbol{\lambda}) \leftarrow \hat{l}^k(\boldsymbol{\lambda})$               ▷store cuts from each iteration
 4: $\mathcal{Q}^{\mathrm{cts}} \leftarrow \{\boldsymbol{\lambda}^t\}_{t=1}^k$               ▷store solutions from each iteration
 5: $V^{\min} \leftarrow$ lower bound from CPA   ▷CPA lower bound for RiskSlimLP is lower bound for $V(\boldsymbol{\lambda}^*)$

Step II: Round and Polish Continuous Solutions from CPA
 6: for each $\boldsymbol{\lambda}^{\mathrm{cts}} \in \mathcal{Q}^{\mathrm{cts}}$ do
 7:     $\boldsymbol{\lambda}^{\mathrm{sr}} \leftarrow$ SequentialRounding$(\boldsymbol{\lambda}^{\mathrm{cts}}, \mathcal{L}, C_0)$     ▷Algorithm 4
 8:     $\boldsymbol{\lambda}^{\mathrm{dcd}} \leftarrow$ DCD$(\boldsymbol{\lambda}^{\mathrm{sr}}, \mathcal{L}, C_0)$     ▷Algorithm 3
 9:     $\mathcal{Q}^{\mathrm{int}} \leftarrow \mathcal{Q}^{\mathrm{int}} \cup \{\boldsymbol{\lambda}^{\mathrm{dcd}}\}$     ▷store polished integer feasible solutions
10: end for
11: $\boldsymbol{\lambda}^{\mathrm{best}} \leftarrow \mathrm{argmin}_{\boldsymbol{\lambda} \in \mathcal{Q}^{\mathrm{int}}} V(\boldsymbol{\lambda})$
12: $V^{\max} \leftarrow V(\boldsymbol{\lambda}^{\mathrm{best}})$     ▷best integer feasible solution is upper bound for $V(\boldsymbol{\lambda}^*)$

Step III: Update Bounds on Objective Terms
13: $(V^{\min}, V^{\max}, L^{\min}, L^{\max}, R^{\min}, R^{\max}) \leftarrow$ ChainedUpdates$(V^{\min}, \ldots, R^{\max}, C_0)$   ▷Algorithm 5

Output: $\boldsymbol{\lambda}^{\mathrm{best}}$, $\hat{l}^{\mathrm{initial}}(\boldsymbol{\lambda})$, $V^{\min}$, $V^{\max}$, $L^{\min}$, $L^{\max}$, $R^{\min}$, $R^{\max}$
```

In Figure 11, we show how the initialization procedure in Algorithm 6 improves the lower bound and the optimality gap over the course of LCPA.

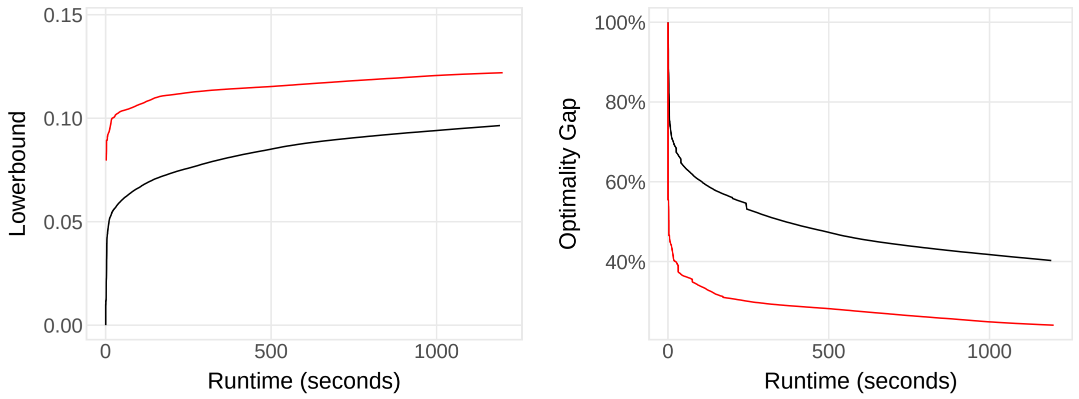

Figure 11: Performance of LCPA in a basic implementation (black) and with the initialization procedure in Algorithm 6 (red). Results reflect performance on an RiskSlimMINLP instance for a simulated dataset with $d = 30$ and $N = 50{,}000$ (see Appendix C for details).

5. Reducing Data-Related Computation

In this section, we present techniques to reduce data-related computation for the risk score problem. The reduction is achieved by exploiting the fact that coefficients belong to a bounded discrete set $\mathcal{L}$. The resulting speed-up lets us fit high quality models in less time and for a larger range of datasets.

5.1 Fast Loss Evaluation via Lookup Tables

The first technique aims to reduce computation when evaluating the loss function and its gradient, which affects runtime when we compute cut parameters (4) and run the rounding and polishing procedures in Section 4.1. The technique requires that the features $\boldsymbol{x}_i$ and coefficients $\boldsymbol{\lambda}$ belong to sets that are bounded, discrete, and regularly spaced, such as $\boldsymbol{x}_i \in \mathcal{X} \subseteq \{0,1\}^d$ and $\boldsymbol{\lambda} \in \mathcal{L} \subseteq \{-10,\ldots,10\}^{d+1}$.

Evaluating the logistic loss, $\log(1+\exp(-\langle \boldsymbol{\lambda}, y_i\boldsymbol{x}_i\rangle))$, is a relatively expensive computation because it involves exponentiation and must be carried out in multiple steps to avoid numerical overflow/underflow when the scores $s_i := \langle \boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle$ are too small/large[6]. When the training data and coefficients belong to discrete bounded sets, the scores $s_i = \langle \boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle$ belong to a discrete and bounded set

$$\mathcal{S} = \left\{\langle \boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle \;\middle|\; i = 1,\ldots,N \text{ and } \boldsymbol{\lambda} \in \mathcal{L}\right\}.$$

If the elements of the feature set $\mathcal{X}$ and the coefficient set $\mathcal{L}$ are regularly spaced, then the scores belong to the set of integers $\mathcal{S} \subseteq \mathbb{Z} \cap [s^{\min}, s^{\max}]$ where:

$$s^{\min} = \min_{i,\boldsymbol{\lambda}} \left\{\langle \boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle \text{ for all } (\boldsymbol{x}_i, y_i) \in \mathcal{D} \text{ and } \boldsymbol{\lambda} \in \mathcal{L}\right\},$$
$$s^{\max} = \max_{i,\boldsymbol{\lambda}} \left\{\langle \boldsymbol{\lambda}, \boldsymbol{x}_i y_i\rangle \text{ for all } (\boldsymbol{x}_i, y_i) \in \mathcal{D} \text{ and } \boldsymbol{\lambda} \in \mathcal{L}\right\}.$$

Thus, we can precompute and store all possible values of the loss function in a lookup table with $s^{\max} - s^{\min} + 1$ rows, where row m contains the value of $[\log(1+\exp(-(m+s^{\min}-1)))]$.

This strategy can reduce the time to evaluate the loss as we replace a computationally expensive operation with a simple lookup operation. In practice, the lookup table is usually small enough to be cached in memory, which yields a substantial runtime speedup. Further, since the values of $s^{\min}$ and $s^{\max}$ can be computed exactly in $\mathrm{O}(N)$ time, the lookup table can be narrowed down as $R^{\max}$ is updated over the course of LCPA.

As shown in Figure 12, using a lookup table reduces the total amount of data-related computation compared to a standard high performance numerical computation library. The reduction in data-related computation may translate into a significant difference in the ability of the algorithm to return a high quality risk score, with a stronger proof of optimality under a time constraint.

5.2 Cheaper Heuristics via Subsampling

The next technique aims to reduce data-related computation for heuristics such as SequentialRounding by using a subsample of the full training dataset.

6. The value of $\exp(s)$ can be computed reliably using IEEE 754 double precision floating point numbers for $s \in [-700, 700]$. The term will overflow to ∞ when $s < -700$, and underflow to 0 when when $s > 700$.

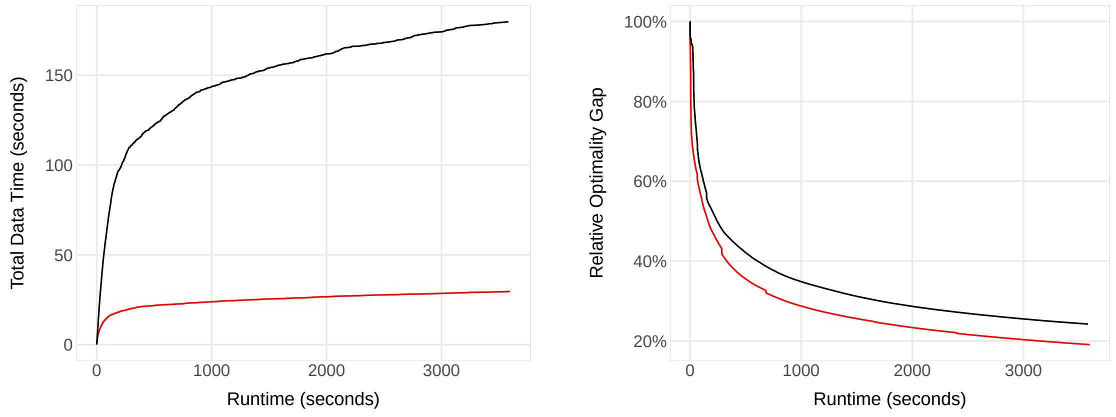

Figure 12: Time spent on data-related computation and optimality gap for LCPA when we evaluate the loss function using a standard high performance numerical computation library (black) and a lookup table (red). Results reflect performance on a simulated dataset with $d = 30$ and $N = 10^6$ (see Appendix C for details).

In theory, we wish to run heuristic procedures frequently because they can yield feasible solutions to RiskSlimMINLP that may update the incumbent solution. In practice, however, each procedure requires multiple evaluations of the loss function, which means that runs that fail to update the incumbent solution effectively slow down the progress of LCPA. If, for example, we ran SequentialRounding each time we found a new set of continuous coefficients in LCPA (i.e, in Step 14 in Algorithm 2), then we would spend too much time rounding, without necessarily finding a better solution.

Our proposed technique works as follows. Before running LCPA, we set aside a *heuristics dataset* $\mathcal{D}_M$ by sampling M points without replacement from the full training dataset $\mathcal{D}_N$. Once we run LCPA, we then reduce data-related computation by running heuristics with $\mathcal{D}_M$. To clarify when the loss and objective are computed using $\mathcal{D}_M$ or $\mathcal{D}_N$, we let $l_i(\boldsymbol{\lambda}) = \log(1 + \exp(\langle \boldsymbol{\lambda}, y_i\boldsymbol{x}_i, \rangle))$ and define:

$$l_M(\boldsymbol{\lambda}) = \frac{1}{M}\sum_{i=1}^{M} l_i(\boldsymbol{\lambda}), \qquad V_M(\boldsymbol{\lambda}) = l_M(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}\right\|_0,$$

$$l_N(\boldsymbol{\lambda}) = \frac{1}{N}\sum_{i=1}^{N} l_i(\boldsymbol{\lambda}), \qquad V_N(\boldsymbol{\lambda}) = l_N(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}\right\|_0.$$

Consider a case where a heuristic procedure returns a promising solution $\boldsymbol{\lambda}^{\text{hr}}$ such that:

$$V_M(\boldsymbol{\lambda}^{\text{hr}}) < V^{\max}. \tag{12}$$

In this case, we compute the objective value on the full training dataset $\mathcal{D}_N$ by evaluating the loss at each of the $N - M$ points that were not included in $\mathcal{D}_M$. As usual, we then update the incumbent solution if $\boldsymbol{\lambda}^{\text{hr}}$ attains an objective value that is less than the current upper bound on RiskSlimMINLP:

$$V_N(\boldsymbol{\lambda}^{\text{hr}}) < V^{\max}. \tag{13}$$

Note that, even though we need to evaluate the loss for the full training dataset $\mathcal{D}_N$ to confirm an incumbent update, this strategy still reduces data-related computation because heuristic procedures require multiple evaluations of the loss functions (e.g., sequential rounding requires $\frac{1}{3}d(d^2-1)$ evaluations of the loss). In the interest of reducing computation, this technique also ignores "false negative" solutions $\boldsymbol{\lambda}^{\text{hr}}$ that do poorly on the heuristics dataset $V_M(\boldsymbol{\lambda}^{\text{hr}}) \geq V^{\max}$ but would update the incumbent on the true dataset $V_N(\boldsymbol{\lambda}^{\text{hr}}) < V^{\max}$.

The main draw of using the subsampling technique is the following generalization bound that guarantees that any solution that updates the incumbent when the objective is evaluated with $\mathcal{D}_M$ will also update incumbent when the objective is evaluated with $\mathcal{D}_N$ (i.e., that any solution that satisfies (12) will also satisfy (13)).

Theorem 1 (Generalization of Sampled Loss on Finite Coefficient Set) *Let $\mathcal{D}_N = (\boldsymbol{x}_i, y_i)_{i=1}^N$ denote a training dataset with $N > 1$ points, and let $\mathcal{D}_M = (\boldsymbol{x}_i, y_i)_{i=1}^M$ denote a sample of M points drawn without replacement from $\mathcal{D}_N$. Let $\boldsymbol{\lambda}$ denote the coefficients of a linear classifier from a finite set $\mathcal{L}$. For all $\varepsilon > 0$, it holds that*

$$\Pr\left(\max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda})\right) \geq \varepsilon\right) \leq |\mathcal{L}| \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{M})(1-\frac{M}{N})(1+\frac{M}{N})\Delta^{\max}(\mathcal{L},\mathcal{D}_N)^2}\right),$$

where

$$\Delta^{\max}(\mathcal{L},\mathcal{D}_N) = \max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(\max_{i=1,\ldots,N} l_i(\boldsymbol{\lambda}) - \min_{i=1,\ldots,N} l_i(\boldsymbol{\lambda})\right).$$

Proof See Appendix A. ■

Theorem 1 is a generalization bound that is derived from a concentration inequality for problems where we are sampling without replacement, known as the Hoeffding-Serfling inequality (see Bardenet et al., 2015). The Hoeffding-Serfling inequality can be significantly tighter than the classical Hoeffding inequality as it ensures that $\Pr\left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda}) \geq \epsilon\right) \to 0$ as $M \to N$ for all $\epsilon > 0$. Here, $\Delta^{\max}(\mathcal{L},\mathcal{D}_N)$ is a normalization term that represents the maximum range of loss values on the full training dataset $\mathcal{D}_N$ for the coefficient set $\mathcal{L}$. This term can be computed cheaply using the smallest and largest values of the coefficients and features as shown in Proposition 1 from Section 4.2.1.

Most machine learning settings are unlike the one we consider here. In such settings, the $|\mathcal{L}|$ term in Theorem 1 produces a bound that is infinite, and thus vacuous. In this case, however, rounding ensures that the $|\mathcal{L}|$ term has at most 2^d elements, which effectively constrains the difference between $l_N(\boldsymbol{\lambda})$ and $l_M(\boldsymbol{\lambda})$. Thus, Theorem 1 can be used to assess the probability that a proposed incumbent update will lead to an actual incumbent update, as shown in Corollary 1. Alternatively, it can be used to choose the size of the subsampled dataset M so that an incumbent update on $\mathcal{D}_M$ is yield an incumbent update on $\mathcal{D}_N$. In either case, the bound can be strengthened by recomputing the normalization term $\Delta^{\max}(\mathcal{L}(\boldsymbol{\rho}),\mathcal{D}_N)$ separately for each continuous solution ρ, or periodically over the course of LCPA (as the MIP solver reduces the set of feasible coefficients via branch-and-bound).

Corollary 1 (Update Probabilities of Rounding Heuristics on Sampled Data) *Consider a rounding heuristic that takes as input a vector of continuous coefficients* $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d) \in \operatorname{conv}(\mathcal{L})$ *and produces as output a vector of integer coefficients* $\boldsymbol{\lambda} \in \mathcal{L}(\boldsymbol{\rho})$ *where*

$$\mathcal{L}(\boldsymbol{\rho}) = \big(\boldsymbol{\lambda} \in \mathcal{L} \;\big|\; \lambda_j \in \{\lceil \rho_j \rceil, \lfloor \rho_j \rfloor\} \textit{ for } j = 1, \ldots, d\big)\,.$$

Consider evaluating the rounding heuristic using a sample of M *points* $\mathcal{D}_M = (\boldsymbol{x}_i, y_i)_{i=1}^M$ *drawn without replacement from the full training dataset* $\mathcal{D}_N = (\boldsymbol{x}_i, y_i)_{i=1}^N$. *Pick a tolerance* $\delta > 0$. *Given rounded coefficients* $\boldsymbol{\lambda} \in \mathcal{L}(\boldsymbol{\rho})$, *compute* $V_M(\boldsymbol{\lambda})$. *If*

$$V_M(\boldsymbol{\lambda}) < V^{\max} - \varepsilon_\delta,$$

then w.p. at least $1 - \delta$, *we have*

$$V_N(\boldsymbol{\lambda}) \leq V^{\max},$$

where

$$\varepsilon_\delta = \Delta^{\max}(\mathcal{L}(\boldsymbol{\rho}), \mathcal{D}_N)\sqrt{\frac{\log(1/\delta) + d\log(2)}{2}\left(1 - \frac{M}{N}\right)\left(1 + \frac{M}{N}\right)}.$$

Proof See Appendix A. ■

6. Benchmarking

In this section, we benchmark methods to learn risk scores with few terms and small integer coefficients. In addition to our proposed method, we consider baseline methods that process the coefficients of penalized logistic regression models using the rounding and polishing techniques from Section 4.1, as well as techniques used in practice.

6.1 Setup

Datasets: We ran experiments on 6 publicly available datasets shown in Table 2. We picked these datasets to explore the performance of each method as we varied the size and nature of the training data. Other than `arrest`, all datasets can be found at the UCI ML repository (Lichman, 2013). The `arrest` dataset must be requested from ICPSR as described in Zeng et al. (2015). We processed each dataset by binarizing all categorical features and some real-valued features. For the purposes of reproducibility, we include all processed datasets other than `arrest` with our final submission.

Dataset	**Source**	**Classification Task**
`adult`	Kohavi (1996)	predict if a U.S. resident earns more than $50 000
`arrest`	Zeng et al. (2015)	predict if a prisoner will be arrested within 3 years of release
`bank`	Moro et al. (2014)	predict if person signs up for bank account after marketing call
`mammo`	Elter et al. (2007)	detect breast cancer using a mammogram
`mushroom`	Schlimmer (1987)	predict if a mushroom is poisonous
`spambase`	Cranor and LaMacchia (1998)	predict if an e-mail is spam

Table 2: Datasets used for benchmarking experiments.

Methods: For each dataset, we fit a risk score with small integer coefficients $\lambda_j \in [-5, 5]$ and limited model size $\|\boldsymbol{\lambda}\|_0 \leq 5$ to match models that are currently used in practice (e.g., Gage et al., 2001). We used a total of 8 methods, described below.

- RiskSLIM (Optimized Risk Score): We formulate an instance of RiskSlimMINLP with the following constraints: $\lambda_0 \in \{-100, \ldots, 100\}$, $\lambda_j \in \{-5, \ldots, 5\}$, and $\|\boldsymbol{\lambda}\|_0 \leq 5$. We set C_0 to a small value (10^{-8}) so that the optimizer of RiskSlimMINLP belongs to the C_0–level set of feasible models (if C_0 is small enough, and the optimality gap is 0.0%, then this ensures we obtain the best-fitting model that satisfies all constraints as discussed in Appendix A.1). We solve each instance using LCPA along with the improvements in Sections 4 and 5. We cap runtime to 20 minutes, and use the CPLEX 12.6.3 Python API on a 3.33GHz CPU with 16GB RAM.

- PLR (Penalized Logistic Regression): We fit logistic regression models with a combined $\ell_1 + \ell_2$ penalty using the glmnet package in R 3.2.3 (Friedman et al., 2010). We consider models for 1100 free parameter instances: 11 values of the elastic-net mixing parameter *alpha* $\in \{0.0, 0.1, \ldots, 1.0\} \times 100$ values of the regularization penalty *lambda* picked by glmnet. We include explicit constraints to bound $\lambda_j \in [-5, 5]$ for $j = 1, \ldots, d$.

- Rd (PLR + Naïve Rounding): We fit a pool of models using PLR. For each model in the pool, we round each coefficient to the nearest integer in $\{-5, \ldots, 5\}$ by setting

$\lambda_j \leftarrow \lceil \min(\max(\lambda_j, -5), 5) \rfloor$ for $j = 1, \ldots, d$, and round the intercept to the nearest integer by setting $\lambda_0 \leftarrow \lceil \lambda_0 \rfloor$.

- RsRd (PLR + Rescaled Rounding): We fit a pool of models using PLR. For each model in the pool, we rescale all coefficients so that the largest coefficient is 5 or -5, and then round them to the nearest integer (i.e., $\lambda_j \to \lceil \gamma \lambda_j \rfloor$ where $\gamma = 5/\max_{j=1,\ldots,d}(|\lambda_j|)$). Here, rescaling is meant to prevent rounding λ_j to zero when $\lambda_j \in [-0.5, 0.5]$.

- SeqRd (PLR + Sequential Rounding): We fit a pool of models using PLR. For each model in the pool, we round the coefficients using sequential rounding (Algorithm 4).

- Rd*/RsRd*/SeqRd* (Polished Versions of Rd/RsRd/SeqRd): We fit a pool of models using Rd/RsRd/SeqRd and polish the coefficients using DCD (Algorithm 3). To ensure that the number of non-zero coefficients does not increase (which would violate the model size constraint), we only run DCD on the set of non-zero coefficients $\{j \mid \lambda_j \neq 0\}$.

Performance Metrics: We evaluated all models in terms of the following metrics.

- Rank Accuracy: We used the standard area under the ROC curve (AUC).

- Sparsity: We used the *model size*, defined as the number of non-zero coefficients, not counting the intercept.

- Risk Calibration: We constructed *reliability diagrams* that show how the predicted risk (x-axis) matches the observed risk (y-axis) for each distinct score[7] (see DeGroot and Fienberg, 1983). We summarize calibration over the full reliability diagram through the *calibration error* (CAL), which represents the root mean squared error between the predicted risk and the observed risk[8] (see Caruana and Niculescu-Mizil, 2004, 2006). A model with perfect calibration yields predicted risk estimates that are perfectly aligned with observed values, meaning that the points on the reliability diagram should fall on the $x = y$ line, and CAL should equal 0.0%.

Model Selection and Assessment: We used a standard nested 5-fold cross-validation (5-CV) to select a final model and assess its predictive accuracy. We fit a final model using all of training data for the instance of the free parameters that: (i) satisfied the model size constraint; and (ii) maximized the 5-CV mean test AUC. Since 5-CV statistics are used to choose free parameters for the final model, they provide an optimistic measure of expected performance for this model. To avoid this bias, we used a nested CV setup where we ran the previous model selection procedure for each fold. Explicitly, for each of the 5 "outer" folds, we ran an "inner" 5-CV and fit a final model using all the data for that fold. We picked a final model for each outer fold for the free parameter instance that: (i) satisfied the model size constraint; and (ii) maximized the inner 5-CV mean test AUC.

7. We estimate the observed risk for each distinct score s as $p(s) = \frac{1}{N(s)} \sum_{i:s=s_i} \mathbb{1}[y_i = +1]$ where s_i is the score for example i. For clarity, when a risk score has over 30 distinct scores (i.e., when the data has real-valued features or the model uses real-valued coefficients), we bin scores into 10 equally-spaced intervals from 0 to 1.
8. CAL is defined as $\frac{1}{N}\sqrt{\sum_s \sum_{i:s_i=s}(p_i - p(s))^2}$ where s_i is the predicted score for example i, and p_i is the predicted risk for example i, and $N(s)$ is the number of examples with a score of s.

6.2 Results

In Table 3, we summarize the performance of risk scores from all methods on all datasets, and provide reliability diagrams to show calibration performance of these models in greater detail in Figure 13. In Figures 14–15, we plot ℓ_0-regularization paths to show how the AUC and CAL of risk scores change with the model size constraint. In Figures 16–18, we show RiskSLIM models for `arrest`, `adult` and `bank`. In what follows, we discuss these results.

On Performance

As shown in Table 2, RiskSLIM models have superior risk calibration and rank accuracy compared to models built with other methods. Specifically, RiskSLIM models have the best 5-CV mean test AUC (i.e., *test AUC*) on 5/6 datasets, and the best 5-CV mean test CAL (i.e.,*test CAL*) on 6/6 datasets. In comparison, models built using baseline methods (i.e., all methods other than PLR) perform slightly worse in terms of test AUC and significantly worse in terms of test CAL. As shown in Figures 14 and 15, the relative performance advantages of RiskSLIM models are more notable for smaller model sizes – which is beneficial since risk scores typically need to have few terms.

Most of these results can be explained by noting that: (i) models that attain low values of the logistic loss have good risk calibration (as shown in Table 3, and observed by Caruana and Niculescu-Mizil, 2004); and, (ii) since we are fitting from a simple class of models, almost all risk scores in Table 2 generalize well (i.e., the test AUC/CAL is very close to their training AUC/CAL). RiskSLIM models attain the smallest value of the logistic loss as they minimize the loss over exact constraints on model form. As such, they perform well in terms of training CAL as per (i), and subsequently in terms of test CAL as per (ii). Likewise, baseline methods that use loss minimizing heuristics such as DCD and SequentialRounding (i.e., Rd*, RsRd* and SeqRd*) produce models with lower loss relative to baseline methods that do not use such techniques (i.e.,Rd, RsRd, and SeqRd). As a result, models built using Rd*, RsRd* and SeqRd* have improved risk calibration.

The results in Table 2 suggest a similar relationship between the logistic loss and rank accuracy, but this has some caveats. In particular, a method such as RsRd, can provably improve AUC by rescaling coefficients before rounding. However, given that the logistic loss is not scale invariant, the rescaled models have high loss and poor risk calibration (see e.g., the reliability diagrams for RsRd in Figure 13). In addition, it is possible for a model that minimizes the logistic loss to have the best training AUC (see e.g., `mammo` where the RiskSLIM model has an optimality gap of 0.0%).

On Computation

RiskSLIM is the only method to pair models with a measure of optimality. Although the risk score problem is $\mathcal{NP}$-hard, we fit models with small optimality gaps in an hour or less by pairing LCPA with the techniques in Sections 4 and 5.

There are also some practical benefits that are difficult to measure. In particular, RiskSLIM could build and evaluate risk scores without the need for parameter tuning and nested CV, meaning that we had to train only 6 models. In comparison, the baseline methods do require parameter tuning and nested CV, meaning that we had to run rounding and polishing techniques and compute AUC for over 33,000 models. Thus, even though the baseline methods are much faster to run for a single choice of parameters, when we consider the computation time needed for parameter tuning, the training process could take far longer than the time used to fit RiskSLIM, especially on large datasets.

On Pitfalls and Best-Practices for Heuristic Methods

Our results show that the performance of risk scores built with heuristics methods depends on a range of factors, including: the rounding technique; the constraints on model form; and the range of feature values. In some cases, risk scores built by simply rounding coefficients to the nearest integer perform well (see e.g., Rd on `bank`). In others, however, performance can falter (see e.g., Rd for `spambase`).

In practice, it is difficult to spot performance issues without carefully evaluating rank accuracy and risk calibration. Common techniques may produce models with good AUC but poor CAL (e.g., RsRd, which is used by Pennsylvania Commission on Sentencing, 2012). In addition, summary statistics such as AUC and CAL may conceal performance issues over the full ROC curve and reliability diagram. The fact that that risk scores built with heuristic methods may have inconsistent performance highlights the importance of the optimality gap, which can help determine if performance issues are due to overly restrictive constraints on the model class. The optimality gap is especially valuable in this setting since PLR models, which provide a natural baseline for risk scores since they do not obey the integrality constraints, may perform poorly due to suboptimal feature selection and overfitting (see e.g., `adult`, `mushroom` and `spambase` in Table 3).

Our results broadly suggest the following best practices for heuristic approaches.

1. Use a loss-minimizing rounding or polishing procedure, ideally SeqRd*
2. Run model selection after rounding since this preserves a large pool of models from which to choose. If we were to run any model selection procedure on the pool of PLR models, and then round the coefficients from that model, then we could dramatically alter the value of the loss, and thereby performance.
3. Use a model selection produce that optimizes the K-CV AUC instead of the K-CV CAL. In fitting models, we initially compared both techniques. As expected, we found that using a model selection procedure that optimizes the mean 5-CV test CAL produces models that perform slightly better in terms of CAL (though not as good as RiskSLIM) but that had far worse AUC. This is because trivial models can have low CAL on problems with class imbalance.
4. Binarize real-valued features. On datasets that contain real-valued features at different scales (e.g., `spambase`), PLR may assign small coefficients to important features with large values. In such cases, rounding techniques set any coefficient $\lambda_j \in [-0.5, 0.5]$ to 0, thereby removing them the model entirely and impacting performance. This issue is difficult to address without binarizing features. Normalizing features, for example, reduces usability as it would also require users to also normalize features when making predictions with the resulting risk score. An alternative option is to rescale the coefficients before rounding. However, this can affect risk calibration given that the logistic loss is not scale invariant (see the reliability diagrams for RsRd in Figure 13).

Dataset	Details	Metric	PLR	Rd	RsRd	SeqRd	Rd*	RsRd*	SeqRd*	RiskSLIM
adult	$N = 32561$ $d = 36$	**test auc**	0.817	0.830	0.830	0.830	0.832	0.832	0.832	0.854
		test cal	5.5%	4.3%	9.1%	4.3%	4.2%	6.3%	4.2%	2.6%
		model size	4	4	4	4	4	4	4	5
		loss value	0.451	0.417	0.484	0.417	0.417	0.458	0.417	0.385
		optimality gap	-	-	-	-	-	-	-	9.7%
arrest	$N = 22530$ $d = 48$	**test auc**	0.700	0.691	0.691	0.691	0.677	0.690	0.677	0.697
		test cal	7.5%	5.7%	20.8%	5.7%	3.8%	15.6%	3.8%	1.7%
		model size	5	5	5	5	4	5	4	5
		loss value	0.638	0.626	1.282	0.626	0.624	0.895	0.624	0.609
		optimality gap	-	-	-	-	-	-	-	4.0%
bank	$N = 41188$ $d = 57$	**test auc**	0.725	0.759	0.759	0.759	0.760	0.749	0.760	0.760
		test cal	2.2%	1.4%	9.5%	1.4%	1.3%	7.2%	1.3%	1.3%
		model size	2	5	5	5	5	2	5	5
		loss value	0.339	0.289	0.953	0.289	0.289	0.333	0.289	0.289
		optimality gap	-	-	-	-	-	-	-	3.5%
mammo	$N = 961$ $d = 14$	**test auc**	0.845	0.845	0.845	0.845	0.845	0.836	0.845	0.843
		test cal	7.3%	8.1%	15.3%	8.1%	7.4%	7.2%	7.4%	5.0%
		model size	3	3	3	3	3	3	3	5
		loss value	0.482	0.480	0.624	0.480	0.480	0.496	0.480	0.469
		optimality gap	-	-	-	-	-	-	-	0.0%
mushroom	$N = 8124$ $d = 113$	**test auc**	0.976	0.973	0.977	0.973	0.978	0.980	0.978	0.989
		test cal	20.9%	12.3%	6.5%	12.3%	5.4%	3.1%	5.4%	1.8%
		model size	5	5	5	5	5	5	5	5
		loss value	0.362	0.200	0.162	0.200	0.144	0.139	0.144	0.069
		optimality gap	-	-	-	-	-	-	-	0.0%
spambase	$N = 4601$ $d = 57$	**test auc**	0.823	0.908	0.862	0.908	0.908	0.913	0.908	0.928
		test cal	10.5%	24.2%	23.6%	24.2%	17.9%	10.3%	17.9%	11.7%
		model size	4	5	5	5	5	5	5	5
		loss value	0.553	0.472	5.670	0.472	0.402	0.381	0.402	0.349
		optimality gap	-	-	-	-	-	-	-	27.8%

Table 3: Performance of risk scores with model size $\|\boldsymbol{\lambda}\|_0 \leq 5$ and integer coefficients $\lambda_j \in \{-5, \ldots, 5\}$. Note PLR models have real-valued coefficients $\lambda_j \in [-5, 5]$. Here: *test auc* is the 5-CV mean test AUC; *test cal* is the 5-CV mean test calibration error; *model size* is the model size of the final model; *loss value* is the value of the logistic loss for the final model; *optimality gap* is the relative optimality gap of the final model.

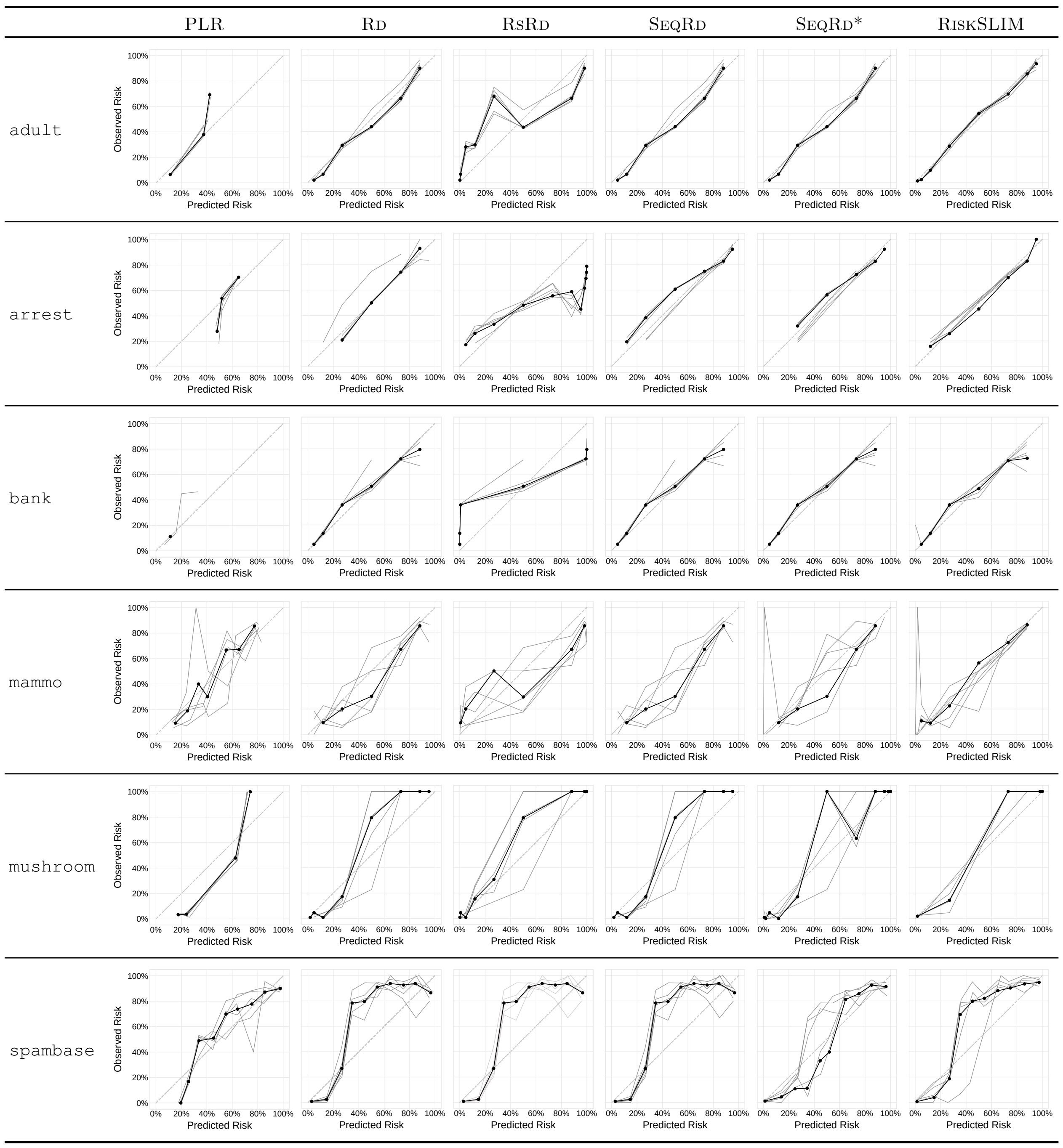

Figure 13: Reliability diagrams for risk scores with model size $\|\boldsymbol{\lambda}\|_0 \leq 5$ and integer coefficients $\lambda_j \in \{-5, \ldots, 5\}$. We include results for PLR as a baseline since its models have real-valued coefficients $\lambda_j \in [-5, 5]$. We plot results for models from each fold on the test data in grey, and for the final model on training data in black.

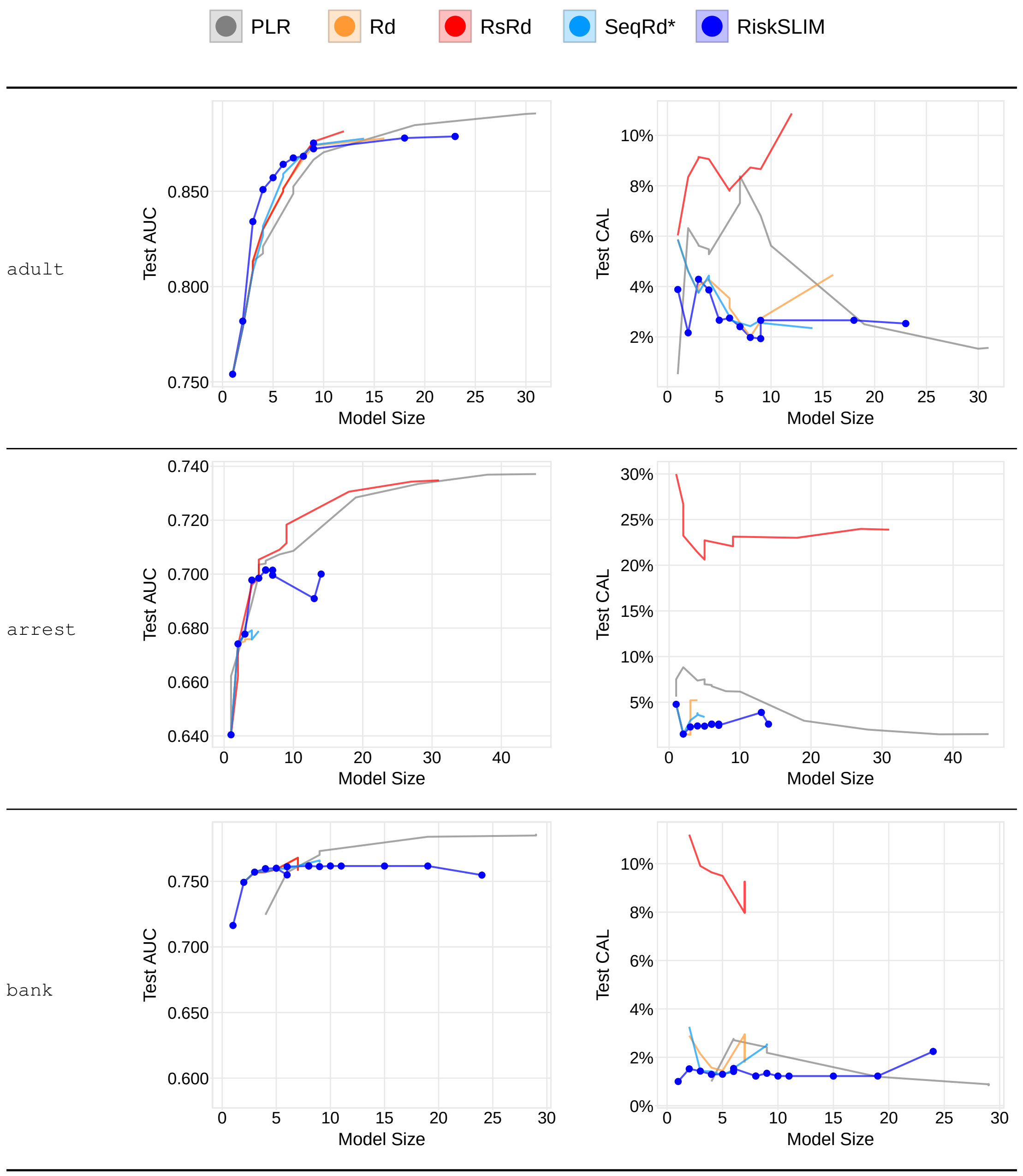

Figure 14: Test AUC (left) and Test CAL (right) of risk scores with integer coefficients $\lambda_j \in \{-5, \ldots, 5\}$ and model sizes $\|\boldsymbol{\lambda}\|_0 \leq R^{\max}$ for $R^{\max} \in \{1, 2, \ldots, 10, 20, 30, 40, 50, \infty\}$. Note PLR risk scores have real-valued coefficients $\lambda_j \in [-5, 5]$.

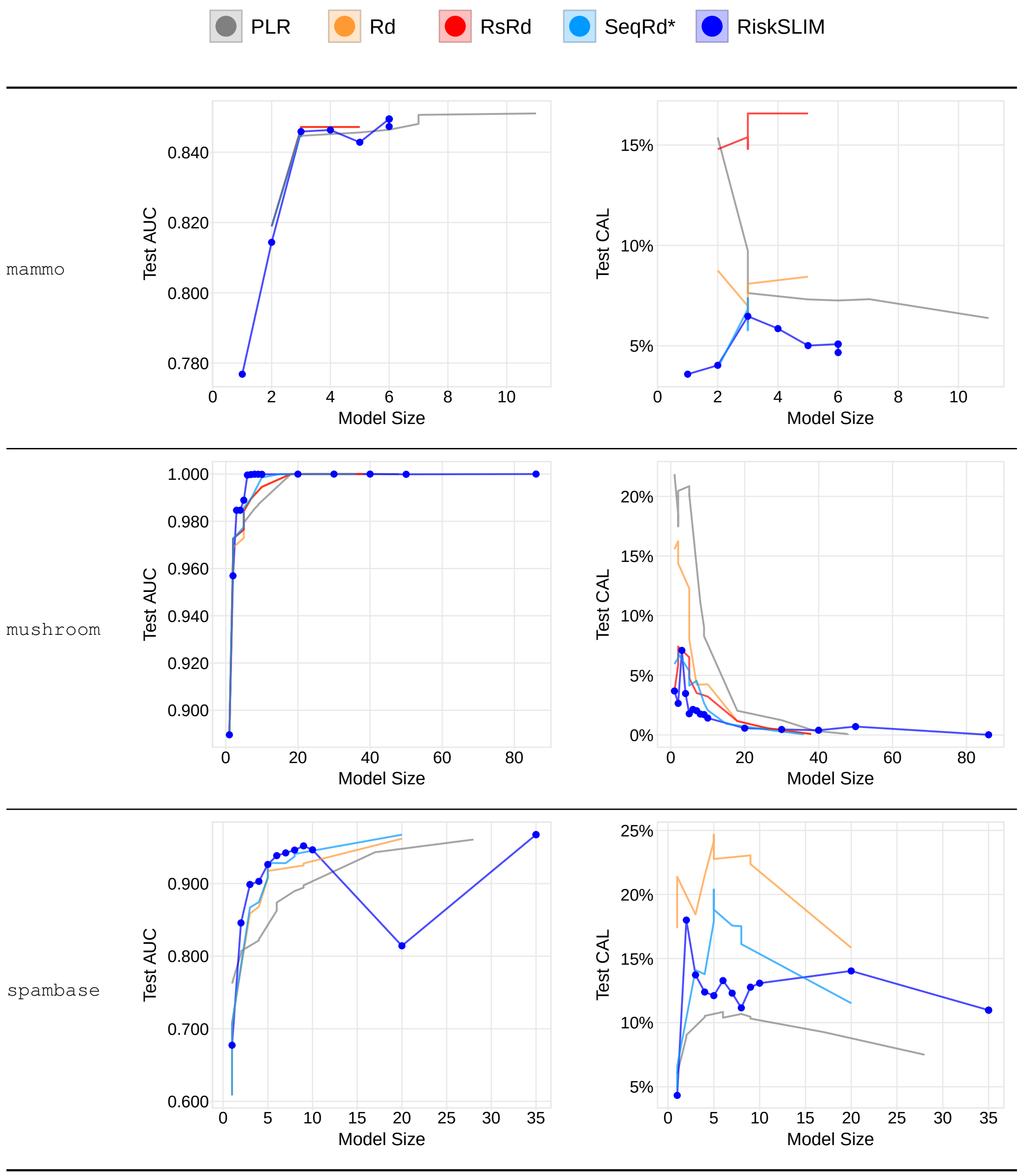

Figure 15: Test AUC (left) and Test CAL (right) of risk scores with integer coefficients $\lambda_j \in \{-5, \ldots, 5\}$ and model sizes $\|\boldsymbol{\lambda}\|_0 \leq R^{\max}$ for $R^{\max} \in \{1, 2, \ldots, 10, 20, 30, 40, 50, \infty\}$. Note PLR risk scores have real-valued coefficients $\lambda_j \in [-5, 5]$.

1.	*Prior Arrests* $\geq$ *2*	1 point	$\cdots\cdots$
2.	*Prior Arrests* $\geq$ *5*	1 point	$+$ $\cdots\cdots$
3.	*Prior Arrests for Local Ordinance*	1 point	$+$ $\cdots\cdots$
4.	*Age at Release between 18 to 24*	1 point	$+$ $\cdots\cdots$
5.	*Age at Release* $\geq$ *40*	-1 points	$+$ $\cdots\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$ $\cdots\cdots$

SCORE	-1	0	1	2	3	4
RISK	11.9%	26.9%	50.0%	73.1%	88.1%	95.3%

Figure 16: RiskSLIM model for `arrest`. RISK represents the predicted probability that a prisoner is arrested for any offense within 3 years of release from prison. This model has a 5-CV mean test AUC/CAL of 0.697/1.7% and training AUC/CAL of 0.701/2.6%.

1.	*Married*	3 points	$\cdots\cdots$
2.	*Reported Capital Gains*	2 points	$+$ $\cdots\cdots$
3.	*Age between 22 to 29*	-1 point	$+$ $\cdots\cdots$
4.	*Highest Level of Education is High School Diploma*	-2 points	$+$ $\cdots\cdots$
5.	*No High School Diploma*	-3 points	$+$ $\cdots\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$ $\cdots\cdots$

SCORE	≤ -1	0	1	2	3	4	5
RISK	5.0%	11.9	26.9%	50.0%	73.1%	88.1%	95.3

Figure 17: RiskSLIM model for `adult`. RISK represents the predicted probability that a US resident earns over \$50 000. This model has a 5-CV mean test AUC/CAL of 0.854/2.4% and training AUC/CAL of 0.860/4.1%.

1.	*Call between January and March*	1 point	$\cdots\cdots$
2.	*Called Previously*	1 point	$+$ $\cdots\cdots$
3.	*Previous Call was Successful*	1 point	$+$ $\cdots\cdots$
4.	*Employment Indicator* $<$ *5100*	1 point	$+$ $\cdots\cdots$
5.	*3 Month Euribor Rate* $\geq$ *100*	-1 point	$+$ $\cdots\cdots$
	ADD POINTS FROM ROWS 1–5	**SCORE**	$=$ $\cdots\cdots$

SCORE	-1	0	1	2	3	4
RISK	4.7%	11.9	26.9%	50.0%	73.1%	88.1%

Figure 18: RiskSLIM model for `bank`. RISK represents the predicted probability that a client opens a new bank account after a marketing call. This model has a 5-CV mean test AUC/CAL of 0.760/1.3% and a training AUC/CAL of 0.760/1.1%.

7. Discussion

Our goal in this paper was to develop a principled approach to create simple risk scores. These models that have been readily used and accepted in high-stakes applications since 1928 (Burgess, 1928) but are still built using ad hoc approaches.

Our approach consists in formulating an exact optimization problem to produce risk scores that were optimized for feature selection, small integer coefficients, and operational constraints. We then solved problem using a cutting plane method that did not stall on non-convex problems, which we paired with techniques to generate feasible solutions, narrow the optimality gap, and reduce data-related computation. An important characteristic of these techniques is that they are more powerful when used together: for instance, using SequentialRounding and DCD to generate a feasible solution can lead to a tighter bound on the number of features through ChainedUpdates, which can reduce computation by reducing the size of the lookup table.

As shown in Sections **??** and 6, the optimal solution to our problem produces risk scores with improved AUC and risk-calibration – especially when fitting from heavily constrained model classes that are typical for risk scores. In addition to these performance benefits, our approach has practical benefits, namely: (i) it can fit models that obey complex operational constraints without parameter tuning, which greatly reduces the number of models that need to be fit to build and evaluate a risk score; (ii) it can assess the optimality of each model, which can help users tell if poor performance is due to the model class or the fitting process.

One interesting direction for future work includes using our proposed approach to build risk scores with constraints to enforce fairness (see Kamishima et al., 2011; Dwork et al., 2012; Zafar et al., 2015). A different direction for future research is to apply our cutting plane algorithm on supervised learning problems with other convex loss functions, but non-convex regularizers or feasible regions.

Acknowledgments

We gratefully acknowledge support from Siemens, Phillips, and Wistron. We would like to thank Paul Rubin for countless helpful discussions.

References

Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237–260, 1998.

David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical Programming*, 69(1-3):1–43, 1995.

James Austin, Roger Ocker, and Avi Bhati. Kentucky pretrial risk assessment instrument validation. *Bureau of Justice Statistics. Grant*, (2009-DB), 2010.

Lihui Bai and Paul A Rubin. Combinatorial Benders Cuts for the Minimum Tollbooth Problem. *Operations Research*, 57(6):1510–1522, 2009. doi: 10.1287/opre.1090.0694. URL http://or.journal.informs.org/cgi/content/abstract/opre.1090.0694v1.

Rémi Bardenet, Odalric-Ambrym Maillard, et al. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

Philip Bobko, Philip L Roth, and Maury A Buster. The usefulness of unit weights in creating composite scores. A literature review, application to content validity, and meta-analysis. *Organizational Research Methods*, 10(4):689–709, 2007.

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

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

Ernest W Burgess. Factors determining success or failure on parole. *The workings of the indeterminate sentence law and the parole system in Illinois*, pages 221–234, 1928.

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

Richard H Byrd, Jorge Nocedal, and Richard A Waltz. Knitro: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

Emilio Carrizosa, Amaya Nogales-Gómez, and Dolores Romero Morales. Strongly agree or strongly disagree?: Rating features in support vector machines. *Information Sciences*, 329:256–273, 2016.

Rich Caruana and Alexandru Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78. ACM, 2004.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

Yann Chevaleyre, Frédéerick Koriche, and Jean-Daniel Zucker. Rounding methods for discrete linear classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 651–659, 2013.

Nelson Cowan. The magical mystery four how is working memory capacity limited, and why? *Current directions in psychological science*, 19(1):51–57, 2010.

Lorrie Faith Cranor and Brian A LaMacchia. Spam! *Communications of the ACM*, 41(8):74–83, 1998.

Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *The statistician*, pages 12–22, 1983.

Grant Duwe and KiDeuk Kim. Sacrificing accuracy for transparency in recidivism risk assessment: The impact of classification method on predictive performance. *Corrections*, pages 1–22, 2016.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226. ACM, 2012.

M Elter, R Schulz-Wendtland, and T Wittenberg. The prediction of breast cancer biopsy outcomes using two cad approaches that both emphasize an intelligible decision process. *Medical Physics*, 34:4164, 2007.

Şeyda Ertekin and Cynthia Rudin. A bayesian approach to learning scoring systems. *Big Data*, 3(4):267–276, 2015.

Vojtěch Franc and Soeren Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th international conference on Machine learning*, pages 320–327. ACM, 2008.

Vojtěch Franc and Sören Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *The Journal of Machine Learning Research*, 10:2157–2192, 2009.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

Brian F Gage, Amy D Waterman, William Shannon, Michael Boechler, Michael W Rich, and Martha J Radford. Validation of clinical classification schemes for predicting stroke. *The Journal of the American Medical Association*, 285(22):2864–2870, 2001.

Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

Sharad Goel, Justin M Rao, and Ravi Shroff. Precinct or prejudice? understanding racial disparities in new york city's stop-and-frisk policy. *Understanding Racial Disparities in New York City's Stop-and-Frisk Policy (March 2, 2015)*, 2015.

Bryce Goodman and Seth Flaxman. Eu regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.

Ruth Hübner and Anita Schöbel. When is rounding allowed in integer nonlinear optimization? *European Journal of Operational Research*, 237(2):404–410, 2014.

D Jennings, TM Amabile, and L Ross. Informal covariation assessment: Data-based vs. theory-based judgments. *Judgment under uncertainty: Heuristics and biases*, pages 211–230, 1982.

Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 643–650. IEEE, 2011.

James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

William A Knaus, Jack E Zimmerman, Douglas P Wagner, Elizabeth A Draper, and Diane E Lawrence. APACHE-acute physiology and chronic health evaluation: a physiologically based classification system. *Critical Care Medicine*, 9(8):591–597, 1981.

William A Knaus, Elizabeth A Draper, Douglas P Wagner, and Jack E Zimmerman. APACHE II: a severity of disease classification system. *Critical Care Medicine*, 13(10):818–829, 1985.

William A Knaus, DP Wagner, EA Draper, JE Zimmerman, Marilyn Bergner, PG Bastos, CA Sirio, DJ Murphy, T Lotring, and A Damiano. The APACHE III prognostic system. risk prediction of hospital mortality for critically ill hospitalized adults. *Chest Journal*, 100(6):1619–1636, 1991.

Y Kodratoff. The comprehensibility manifesto. *KDD Nugget Newsletter*, 94(9), 1994.

Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *KDD*, pages 202–207, 1996.

Wojciech Kotlowski, Krzysztof J Dembczynski, and Eyke Huellermeier. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1113–1120, 2011.

Edward Latessa, Paula Smith, Richard Lemke, Matthew Makarios, and Christopher Lowenkamp. Creation and validation of the ohio risk assessment system: Final report. *Center for Criminal Justice Research, School of Criminal Justice, University of Cincinnati, Cincinnati, OH. Retrieved from http://www. ocjs.*

ohio. gov/ORAS_FinalReport. pd f, 2009.

Jean-Roger Le Gall, Philippe Loirat, Annick Alperovitch, Paul Glaser, Claude Granthil, Daniel Mathieu, Philippe Mercier, Remi Thomas, and Daniel Villers. A simplified acute physiology score for icu patients. *Critical Care Medicine*, 12(11):975–977, 1984.

Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (SAPS II) based on a european/north american multicenter study. *The Journal of the American Medical Association*, 270(24):2957–2963, 1993.

Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. An interpretable stroke prediction model using rules and bayesian analysis. 2013.

A Yu Levin. On an algorithm for the minimization of convex functions. In *Soviet Mathematics Doklady*, volume 160, pages 1244–1247, 1965.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Dmitry M Malioutov and Kush R Varshney. Exact rule learning via boolean compressed sensing. In *ICML (3)*, pages 765–773, 2013.

Olvi L Mangasarian, W Nick Street, and William H Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

Aditya Krishna Menon, Xiaoqian J Jiang, Shankar Vembu, Charles Elkan, and Lucila Ohno-Machado. Predicting accurate probabilities with a ranking loss. In *Machine learning: proceedings of the International Conference. International Conference on Machine Learning*, volume 2012, page 703. NIH Public Access, 2012.

Alan J Miller. Selection of subsets of regression variables. *Journal of the Royal Statistical Society. Series A (General)*, pages 389–425, 1984.

Rui P Moreno, Philipp GH Metnitz, Eduardo Almeida, Barbara Jordan, Peter Bauer, Ricardo Abizanda Campos, Gaetano Iapichino, David Edbrooke, Maurizia Capuzzo, and Jean-Roger Le Gall. SAPS 3 - from evaluation of the patient to evaluation of the intensive care unit. part 2: Development of a prognostic model for hospital mortality at icu admission. *Intensive Care Medicine*, 31(10):1345–1355, 2005.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Joe Naoum-Sawaya and Samir Elhedhli. An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research*, 210(1):33–55, November 2010.

Jaehyun Park and Stephen Boyd. Concave quadratic cuts for mixed-integer quadratic problems. *arXiv preprint arXiv:1510.06421*, 2015a.

Jaehyun Park and Stephen Boyd. A semidefinite programming method for integer convex quadratic minimization. *arXiv preprint arXiv:1504.07672*, 2015b.

Pennsylvania Commission on Sentencing. Interim Report 4: Development of Risk Assessment Scale. Technical report, June 2012.

Mark D Reid and Robert C Williamson. Composite binary losses. *The Journal of Machine Learning Research*, 11:2387–2422, 2010.

Greg Ridgeway. The pitfalls of prediction. *NIJ Journal,* National Institute of Justice, 271:34–40, 2013.

Jeffrey Curtis Schlimmer. Concept acquisition through representational adjustment. 1987.

Choon Hui Teo, Alex Smola, SVN Vishwanathan, and Quoc Viet Le. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2007.

Choon Hui Teo, S Vishwanathan, Alex Smola, and Quoc V Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 1:55, 2009.

U.S. Department of Justice. The Mathematics of Risk Classification: Changing Data into Valid Instruments for Juvenile Courts. July 2005.

Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, pages 1–43, 2015. ISSN 0885-6125. doi: 10.1007/s10994-015-5528-6. URL http://dx.doi.org/10.1007/s10994-015-5528-6.

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: A mechanism for fair classification. *arXiv preprint arXiv:1507.05259*, 2015.

Jiaming Zeng, Berk Ustun, and Cynthia Rudin. Interpretable classification models for recidivism prediction. *arXiv preprint arXiv:1503.07810*, 2015.

Appendix A. Proofs

A.1 Small Regularization Parameters do not Influence Accuracy

In Section 2, we state that if the trade-off parameter C_0 in the objective of the risk score problem is set to a sufficiently small value, then the optimal solution will not sacrifice accuracy (nor logistic loss) for sparsity. Here, we formalize this statement. In what follows, we explicitly include the regularization parameter C_0 in the RiskSLIM objective function so that $V(\boldsymbol{\lambda}; C_0) := l(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$., and omit the intercept term for clarity of exposition.

We start with some definitions. We denote the smallest value of the logistic loss attainable by feasible models with at most t non-zero coefficients as

$$L(t) = \min_{\substack{\boldsymbol{\lambda}\in\mathcal{L} \\ \|\boldsymbol{\lambda}\|_0 \leq t}} l(\boldsymbol{\lambda}).$$

We denote the corresponding sets of minimizers as:

$$\mathcal{M}(t) = \operatorname*{argmin}_{\substack{\boldsymbol{\lambda}\in\mathcal{L} \\ \|\boldsymbol{\lambda}\|_0 \leq t}} l(\boldsymbol{\lambda}).$$

Likewise, we denote the set of minimizers of the logistic loss as

$$\mathcal{M}^{\text{opt}} = \operatorname*{argmin}_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda}).$$

We consider models in $\mathcal{M}^{\text{opt}}$ that attain the smallest ℓ_0 norm, $\boldsymbol{\lambda}^{\text{opt}} \in \operatorname{argmin}_{\boldsymbol{\lambda}\in\mathcal{M}^{\text{opt}}} \|\boldsymbol{\lambda}\|_0$, and denote this value as $t^{\text{opt}} = \left\|\boldsymbol{\lambda}^{\text{opt}}\right\|_0$.

Theorem 2 (Small Regularization Parameters Do Not Influence Accuracy)

1. RiskSLIM *will always choose a solution that achieves a logistic loss of $L(t)$ for some t. That is,*

$$\min_{\boldsymbol{\lambda}\in\mathcal{L}} V(\boldsymbol{\lambda}; C_0) = \min_{t\in\{0,1,2,\ldots,t^{\text{opt}}\}} L(t) + C_0 t.$$

2. *There exists an integer $z \geq 1$ such that if*

$$C_0 \leq \frac{1}{z}\left[L(t^{\text{opt}} - z) - L(t^{\text{opt}})\right],$$

then

$$\boldsymbol{\lambda}^{\text{opt}} \in \operatorname*{argmin}_{\lambda\in\mathcal{L}} V(\boldsymbol{\lambda}; C_0).$$

For this z, the right hand side $\frac{1}{z}\left[L(t^{\text{opt}} - z) - L(t^{\text{opt}})\right]$ is strictly greater than 0. Thus, as long as C_0 is sufficiently small, the regularized objective is minimized by any model $\boldsymbol{\lambda}^{\text{opt}}$ which has the smallest size among the most accurate feasible models.

Proof To prove the first statement, consider the set of feasible models with at most t non-zero coefficients: $\mathcal{F}(t) = \{\boldsymbol{\lambda} \in \mathcal{L} \mid \|\boldsymbol{\lambda}\|_0 \leq t\}$. Since $L(t)$ is the minimal value of the logistic loss for all models with at most t non-zero coefficients, it follows that:

$$L(t) + C_0 t \leq l(\boldsymbol{\lambda}) + C_0 t \text{ for any } \boldsymbol{\lambda} \in \mathcal{F}(t) \tag{14}$$

Define a feasible minimizer of $V(\boldsymbol{\lambda}; C_0)$ as $\boldsymbol{\lambda}^*$, and let $t^* = \|\boldsymbol{\lambda}^*\|_0$. Since $\boldsymbol{\lambda}^* \in \mathcal{F}(t^*)$, the inequality in (14) ensures:

$$L(t^*) + C_0 t^* \leq V(\boldsymbol{\lambda}^*; C_0).$$

By definition of the minimizer:

$$\min_{t \in \{0,1,2,\ldots,t_{\text{opt}}\}} L(t) + C_0 t \leq L(t^*) + C_0 t^*$$

If these inequalities are not all equalities, we have a contradiction with the definition of $\boldsymbol{\lambda}^*$ and t^* as the minimizer of $V(\boldsymbol{\lambda}; C_0)$ and its size. So all must be equality. This proves the first statement.

To prove the second statement, first note that if $C_0 = 0$, then any minimizer of $V(\boldsymbol{\lambda})$ has t^{opt} non-zero coefficients. Say we increase the value of C_0 in the objective starting from zero until the we attain a threshold value such that the optimal solution sacrifices some loss to remove at least one non-zero coefficient. Let $z \geq 1$ be the number of non-zero coefficients removed to obtain the smaller model. At this threshold value of C_0 where we choose the smaller model rather than the one with size t_{opt}, we have

$$L(t^{\text{opt}}) + C_0 t^{\text{opt}} > L(t^{\text{opt}} - z) + C_0(t^{\text{opt}} - z).$$

Simplifying, we obtain:

$$\frac{1}{z}\left[L(t^{\text{opt}} - z) - L(t^{\text{opt}})\right] < C_0.$$

Thus RiskSLIM does not sacrifice sparsity for logistic loss when

$$\frac{1}{z}\left[L(t^{\text{opt}} - z) - L(t^{\text{opt}})\right] \geq C_0.$$

Here, we know that the value on the left side is strictly greater than 0 because otherwise it would contradict the definition of t^{opt} as the smallest size at which the optimal value of the loss would be achieved.

Since we do not know z in advance and since it is just as difficult to compute L as it is to solve the risk score problem, we will not know in advance how small C_0 must be to avoid sacrificing sparsity. However, it does not matter which value of C_0 we pick as long as it is sufficiently small. Thus, setting C_0 to a small value works well in practice. ■

A.2 Proof of Remark 1

Remark 1 (Convergence of Lattice Cutting Plane Algorithm)
Given any training dataset $(\boldsymbol{x}_i, y_i)_{i=1}^N$, any trade-off parameter $C_0 > 0$, and any finite coefficient set $\mathcal{L} \subset \mathbb{Z}^{d+1}$, Algorithm 2 returns an optimal solution to the risk score problem after computing at most $|\mathcal{L}|$ cutting planes and processing at most $2^{D(\mathcal{L})} - 1$ nodes, where $D(\mathcal{L}) = \prod_{j=0}^{d} (\Lambda_j^{max} - \Lambda_j^{min} + 1)$.

Proof We first explain why LCPA attains the optimal objective value, and then justify the upper bounds on the cutting planes and nodes.

Observe that LCPA finds the optimal solution to RiskSlimMINLP through an exhaustive search over the feasible region $\mathcal{L}$. Thus, LCPA is bound to encounter the optimal solution, since it only discards a node $(V^s, \mathcal{P}^s)$ if: (i) the surrogate problem is infeasible over $\mathcal{P}^s$ (in which case RiskSlimMINLP is also infeasible over $\mathcal{P}^s$); or (ii) the surrogate problem has an objective value that exceeds than $V^{\max}$ (in which case, there any integer feasible solution $\mathcal{P}^s$ is also suboptimal).

The bound on the number of cuts follows from the fact that Algorithm 2 only adds cuts at integer feasible solutions, of which there are at most $|\mathcal{L}|$. The bound on the number of processed nodes represents a worst-case limit produced by bounding the depth of the branch-and-bound tree. To do this, we exploit the fact that the splitting rule BranchingRule splits a partition into two mutually exclusive subsets by adding integer-valued bounds such $\lambda_j \geq \lceil \lambda_j \rceil$ and $\lambda_j \leq \lfloor \lambda_j \rfloor - 1$ on a single coefficient to the feasible solution. Consider applying BranchingRule a total of $\Lambda_j^{\max} - \Lambda_j^{\min} + 1$ times in succession on a fixed dimension j. This results in a total of $\Lambda_j^{\max} - \Lambda_j^{\min} + 1$ nodes where each node fixes the coefficient in dimension j to an integer value $\lambda_j \in \{\Lambda_j^{\min}, \ldots, \Lambda_j^{\max}\}$. Pick any node and repeat this process for coefficients in the remaining dimensions. The resulting branch and bound tree will have at most $D(\mathcal{L})$ leaf nodes where $\boldsymbol{\lambda}$ is restricted to integer feasible solutions. ■

A.3 Proof of Proposition 1

Proposition 1 (Bounds on Logistic Loss over a Bounded Coefficient Set) *Let $(\boldsymbol{x}_i, y_i)_{i=1}^N$ denote a dataset where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ for $i = 1, \ldots, N$. Consider the value of the normalized logistic loss for a linear classifier with coefficients $\boldsymbol{\lambda} \in \mathcal{L} \subset \mathbb{R}^d$*

$$l(\boldsymbol{\lambda}) = \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp(-\langle \boldsymbol{\lambda}, y_i \boldsymbol{x}_i \rangle))$$

If the coefficient set $\mathcal{L}$ is bounded, then $l(\boldsymbol{\lambda}) \in [L^{\min}, L^{\max}]$ for all $\boldsymbol{\lambda} \in \mathcal{L}$ where:

$$\begin{aligned}
L^{\min} &= \frac{1}{N} \sum_{i:y_i=+1} \log\left(1 + \exp(-s_i^{\max})\right) + \frac{1}{N} \sum_{i:y_i=-1} \log\left(1 + \exp(s_i^{\min})\right), \\
L^{\max} &= \frac{1}{N} \sum_{i:y_i=+1} \log\left(1 + \exp(-s_i^{\min})\right) + \frac{1}{N} \sum_{i:y_i=-1} \log\left(1 + \exp(s_i^{\max})\right), \\
s_i^{\min} &= \min_{\boldsymbol{\lambda} \in \mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle \quad \textit{for } i = 1, \ldots, N, \\
s_i^{\max} &= \max_{\boldsymbol{\lambda} \in \mathcal{L}} \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle \quad \textit{for } i = 1, \ldots, N.
\end{aligned}$$

Proof Since the coefficient set $\mathcal{L}$ is bounded, the data are $(\boldsymbol{x}_i, y_i)_{i=1}^N$ bounded, and the normalized logistic loss function $l(\boldsymbol{\lambda})$ is continuous, it follows that the value of $l(\boldsymbol{\lambda})$ is also bounded:

$$l(\boldsymbol{\lambda}) \in [\min_{\boldsymbol{\lambda} \in \mathcal{L}} l(\boldsymbol{\lambda}), \max_{\boldsymbol{\lambda} \in \mathcal{L}}, l(\boldsymbol{\lambda})] \text{ for all } \boldsymbol{\lambda} \in \mathcal{L}.$$

Thus we only need to show that $L^{\min} \leq \min_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda})$, and $L^{\max} \geq \max_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda})$. For the lower bound, we observe that:

$$\begin{aligned}
\min_{\boldsymbol{\lambda}\in\mathcal{L}} l(\boldsymbol{\lambda}) &= \min_{\boldsymbol{\lambda}\in\mathcal{L}} \frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-\langle \boldsymbol{\lambda}, y_i\boldsymbol{x}_i\rangle)) \\
&= \min_{\boldsymbol{\lambda}\in\mathcal{L}} \frac{1}{N}\sum_{i:y_i=+1} \log(1+\exp(-\langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) + \frac{1}{N}\sum_{i:y_i=-1} \log(1+\exp(\langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) \\
&\geq \frac{1}{N}\sum_{i:y_i=+1} \min_{\boldsymbol{\lambda}\in\mathcal{L}} \log(1+\exp(-\langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) + \frac{1}{N}\sum_{i:y_i=-1} \min_{\boldsymbol{\lambda}\in\mathcal{L}} \log(1+\exp(\langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) \\
&= \frac{1}{N}\sum_{i:y_i=+1} \log(1+\exp(-\max_{\boldsymbol{\lambda}\in\mathcal{L}}\langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) + \frac{1}{N}\sum_{i:y_i=-1} \log(1+\exp(\min_{\boldsymbol{\lambda}\in\mathcal{L}}\langle \boldsymbol{\lambda}, \boldsymbol{x}_i\rangle)) \\
&= \frac{1}{N}\sum_{i:y_i=+1} \log(1+\exp(-s_i^{\max}) + \frac{1}{N}\sum_{i:y_i=-1} \log(1+\exp(s_i^{\min})) \\
&= L^{\min}.
\end{aligned}$$

The upper bound can be derived in a similar manner.

■

A.4 Proof of Proposition 2

Proposition 2 (Upper Bound on Optimal Number of Non-Zero Coefficients) *Given an upper bound on the objective value $V^{\max} \geq V(\boldsymbol{\lambda}^*)$, and a lower bound on the loss function $L^{\min} \leq l(\boldsymbol{\lambda}^*)$, we can derive an upper bound on the value of the number of optimal non-zero coefficients $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$ as*

$$R^{\max} = \left\lfloor \frac{V^{\max} - L^{\min}}{C_0} \right\rfloor.$$

Proof We are given that $V^{\max} \geq V(\boldsymbol{\lambda}^*)$ where $V(\boldsymbol{\lambda}^*) := l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0$ by definition. Thus, we can recover the upper bound from Proposition 2 as follows:

$$\begin{aligned}
l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0 &\leq V^{\max}, \\
\|\boldsymbol{\lambda}^*\|_0 &\leq \frac{V^{\max} - l(\boldsymbol{\lambda}^*)}{C_0}, \\
\|\boldsymbol{\lambda}^*\|_0 &\leq \frac{V^{\max} - L^{\min}}{C_0}, && (15) \\
\|\boldsymbol{\lambda}^*\|_0 &\leq \left\lfloor \frac{V^{\max} - L^{\min}}{C_0} \right\rfloor. && (16)
\end{aligned}$$

Here, (15) follows from the fact that $L^{\min} \leq l(\boldsymbol{\lambda}^*)$ by definition, and (16) follows from the fact that the number of non-zero coefficients is a natural number. ■

A.5 Proof of Proposition 3

Proposition 3 (Upper Bound on Optimal Loss)
Given an upper bound on the objective value $V^{\max} \geq V(\boldsymbol{\lambda}^*)$, and a lower bound on the number of non-zero coefficients $R^{\min} \leq \|\boldsymbol{\lambda}^*\|_0$, we can derive an upper bound on value of the loss function $L^{\max} \geq l(\boldsymbol{\lambda}^*)$ where

$$L^{\max} = V^{\max} - C_0 R^{\min}.$$

Proof We are given that $V^{\max} \geq V(\boldsymbol{\lambda}^*)$ where $V(\boldsymbol{\lambda}^*) := l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0$ by definition. Thus, we can recover the upper bound from Proposition 3 as follows:

$$\begin{aligned} l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0 &\leq V^{\max}, \\ l(\boldsymbol{\lambda}^*) &\leq V^{\max} - C_0 \|\boldsymbol{\lambda}^*\|_0, \\ l(\boldsymbol{\lambda}^*) &\leq V^{\max} - C_0 R^{\min}. \end{aligned}$$

Here, the last line follows from the fact that $R^{\min} \leq \|\boldsymbol{\lambda}^*\|_0$ by definition. ■

A.6 Proof of Proposition 4

Proposition 4 (Lower Bound on Optimal Loss)
Given a lower bound on the objective value $V^{\min} \leq V(\boldsymbol{\lambda}^*)$, and an upper bound on the number of non-zero coefficients $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$, we can derive a lower bound on value of the loss function $L^{\min} \leq l(\boldsymbol{\lambda}^*)$ where

$$L^{\min} = V^{\min} - C_0 R^{\max}.$$

Proof We are given that $V^{\min} \leq V(\boldsymbol{\lambda}^*)$ where $V(\boldsymbol{\lambda}^*) := l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0$ by definition. Thus, we can recover the lower bound from Proposition 4 as follows:

$$\begin{aligned} l(\boldsymbol{\lambda}^*) + C_0 \|\boldsymbol{\lambda}^*\|_0 &\geq V^{\min}, \\ l(\boldsymbol{\lambda}^*) &\geq V^{\min} - C_0 \|\boldsymbol{\lambda}^*\|_0, \\ l(\boldsymbol{\lambda}^*) &\geq V^{\min} - C_0 R^{\max}. \end{aligned}$$

Here, the last line follows from the fact that $R^{\max} \geq \|\boldsymbol{\lambda}^*\|_0$ by definition. ■

A.7 Proof of Theorem 1

Theorem 1 (Generalization of Sampled Loss on Finite Coefficient Set) *Let $\mathcal{D}_N = (\boldsymbol{x}_i, y_i)_{i=1}^N$ denote a finite training set composed of $N > 1$ points and $\mathcal{D}_M = (\boldsymbol{x}_i, y_i)_{i=1}^M$ denote a sample of M points drawn without replacement from $\mathcal{D}_N$. Let $\boldsymbol{\lambda}$ denote the coefficients of a linear classifier from a finite set $\mathcal{L}$. Then, for all $\varepsilon > 0$, we have that*

$$\Pr\left(\max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda})\right) \geq \varepsilon\right) \leq |\mathcal{L}| \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{M})(1-\frac{M}{N})(1+\frac{M}{N})\Delta^{\max}(\mathcal{L},\mathcal{D}_N)^2}\right),$$

where

$$\Delta^{\max}(\mathcal{L},\mathcal{D}_N) = \max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(\max_{i=1,\ldots,N} l_i(\boldsymbol{\lambda}) - \min_{i=1,\ldots,N} l_i(\boldsymbol{\lambda})\right).$$

Proof For a fixed set coefficients $\boldsymbol{\lambda} \in \mathcal{L}$, consider a finite sample of N points composed of the values for the loss function $l_i(\boldsymbol{\lambda})$ for each example in the full training dataset $\mathcal{D}_N = (\boldsymbol{x}_i, y_i)_{i=1}^N$. Let $l_N(\boldsymbol{\lambda}) = \frac{1}{N}\sum_{i=1}^N l_i(\boldsymbol{\lambda})$ and $l_M(\boldsymbol{\lambda}) = \frac{1}{M}\sum_{i=1}^M l_i(\boldsymbol{\lambda})$. Then, the Hoeffding-Serfling inequality (see e.g., Theorem 2.4 in Bardenet et al., 2015) guarantees the following for all $\varepsilon > 0$:

$$\Pr\left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda}) \geq \varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{M})(1-\frac{M}{N})(1+\frac{M}{N})\Delta(\boldsymbol{\lambda},\mathcal{D}_N)^2}\right),$$

where

$$\Delta(\boldsymbol{\lambda},\mathcal{D}_N) = \max_{i=1,\ldots,N} l_i(\boldsymbol{\lambda}) - \min_{i=1,\ldots,N} l_i(\boldsymbol{\lambda}).$$

We recover the desired inequality by generalizing this bound to hold for all $\boldsymbol{\lambda} \in \mathcal{L}$ as follows.

$$\begin{aligned}
\Pr\left(\max_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda})\right) \geq \varepsilon\right) &= \Pr\left(\bigcup_{\boldsymbol{\lambda}\in\mathcal{L}}\left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda}) \geq \varepsilon\right)\right), \\
&\leq \sum_{\boldsymbol{\lambda}\in\mathcal{L}} \Pr\left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda}) \geq \varepsilon\right), \qquad (17) \\
&\leq \sum_{\boldsymbol{\lambda}\in\mathcal{L}} \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{M})(1-\frac{M}{N})(1+\frac{M}{N})\Delta(\boldsymbol{\lambda},\mathcal{D}_N)^2}\right), \qquad (18) \\
&\leq |\mathcal{L}| \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{M})(1-\frac{M}{N})(1+\frac{M}{N})\Delta^{\max}(\mathcal{L},\mathcal{D}_N)^2}\right). \qquad (19)
\end{aligned}$$

Here, (17) follows from the union bound, (18) follows from the Hoeffding Serling inequality, (19) follows from the fact that $\Delta(\boldsymbol{\lambda},\mathcal{D}_N) \leq \Delta^{\max}(\mathcal{L},\mathcal{D}_N)$ given that $\boldsymbol{\lambda} \in \mathcal{L}$. ■

A.8 Proof of Corollary 1

Corollary 1 (Update Probabilities of Rounding Heuristics on Sampled Data) *Consider a rounding heuristic that takes as input a vector of continuous coefficients* $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d) \in \operatorname{conv}(\mathcal{L})$ *and produces as output a vector of integer coefficients* $\boldsymbol{\lambda} \in \mathcal{L}(\boldsymbol{\rho})$ *where*

$$\mathcal{L}(\boldsymbol{\rho}) = \left(\boldsymbol{\lambda} \in \mathcal{L} \;\middle|\; \lambda_j \in \{\lceil \rho_j \rceil, \lfloor \rho_j \rfloor\} \text{ for } j = 1, \ldots, d\right).$$

Consider evaluating the rounding heuristic using a sample of M *points* $\mathcal{D}_M = (\boldsymbol{x}_i, y_i)_{i=1}^{M}$ *drawn without replacement from the full training dataset* $\mathcal{D}_N = (\boldsymbol{x}_i, y_i)_{i=1}^{N}$*. Pick a tolerance* $\delta > 0$*. Given rounded coefficients* $\boldsymbol{\lambda} \in \mathcal{L}(\boldsymbol{\rho})$*, compute* $V_M(\boldsymbol{\lambda})$*. If*

$$V_M(\boldsymbol{\lambda}) < V^{\max} - \varepsilon_\delta,$$

then w.p. at least $1 - \delta$*, we have*

$$V_N(\boldsymbol{\lambda}) \leq V^{\max}.$$

Proof We will first show that for any tolerance that we pick $\delta > 0$, the prescribed choice of ε_δ will ensure that $V_N(\boldsymbol{\lambda}) - V_M(\boldsymbol{\lambda}) \leq \varepsilon_\delta$ w.p. at least $1 - \delta$. Restating the result of Theorem 1, we have that for any $\varepsilon > 0$:

$$\Pr\left(\max_{\boldsymbol{\lambda} \in \mathcal{L}} \left(l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda})\right) \geq \varepsilon\right) \leq |\mathcal{L}| \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{M})(1 - \frac{M}{N})(1 + \frac{M}{N})\Delta^{\max}(\mathcal{L}, \mathcal{D}_N)^2}\right). \tag{20}$$

Note that $l_N(\boldsymbol{\lambda}) - l_M(\boldsymbol{\lambda}) = V_N(\boldsymbol{\lambda}) - V_M(\boldsymbol{\lambda})$ for any fixed $\boldsymbol{\lambda}$. In addition, note that the set of rounded coefficients $\mathcal{L}(\boldsymbol{\rho})$ contains at most $|\mathcal{L}(\boldsymbol{\rho})| \leq 2^d$ coefficients vectors. Therefore, in this setting, (20) implies that for any $\varepsilon > 0$,

$$\Pr\left(V_N(\boldsymbol{\lambda}) - V_M(\boldsymbol{\lambda}) \geq \varepsilon\right) \leq 2^d \exp\left(-\frac{2\varepsilon^2}{(\frac{1}{M})(1 - \frac{M}{N})(1 + \frac{M}{N})\Delta(\mathcal{L}(\boldsymbol{\rho}), \mathcal{D}_N)^2}\right). \tag{21}$$

By setting $\varepsilon = \varepsilon_\delta$ and simplifying the terms on the right hand side in (21), we can see that

$$\Pr\left(V_N(\boldsymbol{\lambda}) - V_M(\boldsymbol{\lambda}) \geq \varepsilon_\delta\right) \leq \delta.$$

Thus, the prescribed value of ε_δ ensures that $V_N(\boldsymbol{\lambda}) - V_M(\boldsymbol{\lambda}) \leq \varepsilon_\delta$ w.p. at least $1 - \delta$.

Since we have set ε_δ so that $V_N(\boldsymbol{\lambda}) - V_M(\boldsymbol{\lambda}) \leq \varepsilon_\delta$ w.p. at least $1 - \delta$, we now only need to show any $\boldsymbol{\lambda}$ that satisfies $V_M(\boldsymbol{\lambda}) < V^{\max} - \varepsilon_\delta$ will also satisfy $V_N(\boldsymbol{\lambda}) \leq V^{\max}$ to complete the proof. To see this, observe that:

$$\begin{aligned} V_N(\boldsymbol{\lambda}) - V_M(\boldsymbol{\lambda}) &\leq \varepsilon_\delta, \\ V_N(\boldsymbol{\lambda}) &\leq V_M(\boldsymbol{\lambda}) + \varepsilon_\delta, \\ V_N(\boldsymbol{\lambda}) &< V^{\max}. \end{aligned} \tag{22}$$

Here, (22) follows from the fact that $V_M(\boldsymbol{\lambda}) < V^{\max} - \varepsilon_\delta \implies V_M(\boldsymbol{\lambda}) + \varepsilon_\delta < V^{\max}$ ■

Appendix B. IP and LP Formulations

Basic Formulation for RiskSlimMIP and RiskSlimLP

$$\begin{array}{llllr}
\min\limits_{\theta,\boldsymbol{\lambda},\alpha} & \theta & + & C_0 \sum\limits_{j=1}^{d} \alpha_j & \\
\text{s.t.} & \theta & \geq & l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle \quad t = 1,\ldots,k & \textit{loss cuts} \text{ (23a)} \\
& \lambda_j & \leq & \Lambda_j^{\max} \alpha_j \quad j = 1,\ldots,d & \textit{set } \ell_0 \textit{ indicator for +ve coefs} \text{ (23b)} \\
& \lambda_j & \geq & -\Lambda_j^{\min} \alpha_j \quad j = 1,\ldots,d & \textit{set } \ell_0 \textit{ indicator for }-\textit{ve coefs} \text{ (23c)} \\
& \theta & \geq & 0 & \textit{approximate loss value} \text{ (23d)} \\
& \lambda_j & \in & \{\Lambda_j^{\min}, \ldots, \Lambda_j^{\max}\} \quad j = 1,\ldots,d & \textit{coef. bounds} \text{ (23e)} \\
& \alpha_j & \in & \{0, 1\} \quad j = 1,\ldots,d & \ell_0 \textit{ indicators} \text{ (23f)}
\end{array}$$

We present an IP formulation for RiskSlimMIP in (23). The LP formulation for RiskSlimLP can be obtained by relaxing the integrality constraints in (23e) and (23f). This formulation contains $2d + 2$ constraints and $k + 2d$ variables. The approximate loss function is built through cuts constraints in (23a) as described in Section 3.1. Here, $\theta \in \mathbb{R}_+$ is an auxiliary variable that represents the value of the approximate loss function, and λ_j are coefficients that are bound to integer values in (23e). The ℓ_0-penalty is computed through binary indicator variables $\alpha_j = 1[\lambda_j \neq 0]$ in constraints (23b) and (23c).

Augmented Formulation for RiskSlimMIP and RiskSlimLP

$$\begin{array}{llllr}
\min\limits_{\theta,\boldsymbol{\lambda},\alpha} & & & V & \\
\text{s.t.} & V & = & \theta + C_0 R & \textit{objective value} \text{ (24a)} \\
& R & = & \sum\limits_{j=1}^{d} \alpha_j & \ell_0 \textit{ norm} \text{ (24b)} \\
& \theta & \geq & l(\boldsymbol{\lambda}^t) + \langle \nabla l(\boldsymbol{\lambda}^t), \boldsymbol{\lambda} - \boldsymbol{\lambda}^t \rangle \quad t = 1,\ldots,k & \textit{loss cuts} \\
& \lambda_j & \leq & \Lambda_j^{\max} \alpha_j \quad j = 1,\ldots,d & \textit{set } \ell_0 \textit{ indicator for +ve coefs} \\
& \lambda_j & \geq & -\Lambda_j^{\min} \alpha_j \quad j = 1,\ldots,d & \textit{set } \ell_0 \textit{ indicator for }-\textit{ve coefs} \\
& \lambda_j & \in & \{\Lambda_j^{\min}, \ldots, \Lambda_j^{\max}\} \quad j = 1,\ldots,d & \textit{coefficient bounds} \\
& \alpha_j & \in & \{0, 1\} \quad j = 1,\ldots,d & \ell_0 \textit{ indicators} \\
& V & \in & [V^{\min}, V^{\max}] & \textit{objective bounds} \text{ (24c)} \\
& \theta & \in & [L^{\min}, L^{\max}] & \textit{loss bounds} \text{ (24d)} \\
& R & \in & \{R^{\min}, \ldots, R^{\max}\} & \ell_0 \textit{ bounds} \text{ (24e)}
\end{array}$$

We present an IP formulation for the augmented version of RiskSlimMIP in (24). The corresponding LP formulation is obtained by relaxing the integrality constraints in (24c) and (24c). The formulation in (24) includes two auxiliary variables: V which captures the objective value as per (24a); and R, which captures the ℓ_0-norm as per (24b). All other variables and constraints are identical to those in formulation (23). Note that the variables V, θ and R may appear redundant, but are needed to effectively use the Chained Updates (Algorithm 5) using the *HeuristicCallback* in CPLEX 12.6 (in particular, this callback can only change upper and lower bounds on individual variables in the MIP, but cannot change the LHS / RHS value of a constraint).

Appendix C. Details on Experimental Comparison with Simulated Data

C.1 Data Simulation Procedure

We simulated data from the `breastcancer` dataset (Mangasarian et al., 1995). The original dataset can be obtained from the UCI ML repository (Lichman, 2013), and has a total of $N = 683$ samples and $d = 9$ features $x_{ij} \in \{0, \ldots, 10\}$. Using the original dataset, we generated a collection of simulated datasets using the procedure in Algorithm 7. Specifically, we first simulated the largest dataset we needed (with $N^{\max} = 5 \times 10^6$ samples and $d^{\max} = 30$ features) by replicating features and samples from the original dataset and adding a small amount of normally distributed noise. Next, we created smaller datasets by taking *nested* subsets of the samples and the features. This ensured that any simulated dataset with d features and N samples contains all of the features and examples for a simulated dataset with $d' < d$ features and $N' < N$ samples. We designed this procedure to have two useful properties:

- It would produce difficult instances of the risk score problem. Here, RiskSlimMINLP instances for simulated datasets with $d > 9$ are challenging in terms of feature selection because they contain replicates of the original 9 features, which are strongly correlated with each another. Feature selection becomes exponentially harder when collections of highly correlated features are used, since this means that there are an exponentially larger number of slightly suboptimal solutions.

- We could make inferences about the optimal objective value of RiskSlimMINLP instances we may not have been able to solve. Say, for example, that we could not solve an instance of the risk score problem for the simulated dataset with $(d, N) = (20, 10^6)$, but could solve an instance for the simulated dataset with $(d, N) = (10, 10^6)$. In this case, we knew that the optimal objective value of the $(d, N) = (20, 10^6)$ instance had to be less than or equal to the optimal objective value of the $(d, N) = (10, 10^6)$ instance because the $(d, N) = (20, 10^6)$ dataset contained all of the features as the $(d, N) = (10, 10^6)$ dataset.

C.2 Implementation Details

We fit risk scores for all datasets by formulating an RiskSlimMINLP instance where $C_0 = 10^{-8}$, $\lambda_0 \in \{-100, 100\}$, and $\lambda_j = \{-10, \ldots, 10\}$ for $j = 1, \ldots, d$. We solved this instance using the following techniques: (i) CPA (Algorithm 1); (ii) LCPA (Algorithm 2); (iii) an active set MINLP algorithm (ActiveSetMINLP); (iv) an interior point MINLP algorithm (InteriorMINLP); (v) an interior CG MINLP algorithm (InteriorCGMINLP). We solved all instances on a 3.33 GHz Intel Xeon CPU with 16GB of RAM and capped total runtime to 6 hours. If an algorithm could not solve the problem to optimality before the 6 hour time limit, we reported results based on the best feasible solution it found. We implemented both CPA and LCPA using the CPLEX 12.6.3 Python API. We solved instances using MINLP algorithms using the Artelsys Knitro 9.0 MINLP solver (Byrd et al., 2006), which we accessed using MATLAB 2015b.

Algorithm 7 Simulation Procedure

```
Input
    $X^{\text{original}} = [x_{ij}]_{i=1\ldots N^{\text{original}}, j=1\ldots d^{\text{original}}}$        feature matrix of original dataset
    $Y^{\text{original}} = [y_i]_{i=1\ldots N^{\text{original}}}$        label matrix of original dataset
    $d^1 \ldots d^{\max}$ s.t. $0 < d^1 < \ldots < d^{\max}$        desired dimensions for simulated datasets
    $N^1 \ldots N^{\max}$ s.t. $0 < N^1 < \ldots < N^{\max}$        desired sample sizes for simulated datasets

Initialize
    $\mathcal{J}^{\text{original}} = [1, \ldots, d^{\text{original}}]$        index array for original features
    $\mathcal{J}^{\max} \leftarrow []$        index array of features for largest simulated dataset
    $m^{\text{full}} \leftarrow \lfloor d^{\max}/d^{\text{original}} \rfloor$
    $m^{\text{remainder}} \leftarrow d^{\max} - d^{\text{original}}$

Step I: Generate Largest Dataset

 1: for $m = 1, \ldots, m^{\text{full}}$ do
 2:     $\mathcal{J}^{\max} \leftarrow [\mathcal{J}^{max}, \mathsf{RandomPermute}(\mathcal{J}^{\text{original}})]$
 3: end for
        $\mathcal{J}^{\max} \leftarrow [\mathcal{J}^{max}, \mathsf{RandomSampleWithoutReplacement}(\mathcal{J}^{\text{original}}, m^{\text{remainder}})]$

 4: for $i = 1, \ldots, N^{\max}$ do
 5:     sample $l$ with replacement from $1, \ldots N^{\text{original}}$
 6:     $y_i^{\max} \leftarrow y_i$
 7:     for $j = 1, \ldots, d^{\max}$ do
 8:         $k \leftarrow \mathcal{J}^{\max}[j]$
 9:         sample $\varepsilon$ from Normal$(0, 0.5)$
10:         $x_{ij}^{\max} \leftarrow \lceil x_{l,k} + \varepsilon \rfloor$        ▷new features are noisy versions of original features
11:         $x_{ij}^{\max} \leftarrow \min(10, \max(0, x_{ij}^{\max}))$        ▷new features have same bounds as old features
12:     end for
13: end for
14: $X^{\max} \leftarrow [x_{ij}]_{i=1\ldots N^{\max}, j=1\ldots d^{\max}}$
15: $Y^{\max} \leftarrow [y_i^{max}]_{i=1\ldots N^{\max}}$

Step II: Generate Smaller Datasets

16: for $d = [d^1, \ldots, d^{\max}]$ do
17:     for $N = [N^1, \ldots, N^{\max}]$ do
18:         $X^{(N,d)} = X^{\max}[1:N, 1:d]$
19:         $Y^N = Y^{\max}[1:N]$
20:     end for
21: end for

Output: simulated datasets $(X^{(N,d)}, Y^N)$ for all $N^1 \ldots N^{\max}$ and $d^1 \ldots d^{\max}$.
```

C.3 Results for All MINLP Algorithms

We examined the performance of 3 MINLP algorithms in Artelsys Knitro 9.0: an active set algorithm (ActiveSetMINLP); an interior-point algorithm (InteriorMINLP); and, an interior-point algorithm where the primal-dual KKT system is solved with a conjugate gradient method (InteriorCGMINLP). We only show results from ActiveSetMINLP in the main text because all three algorithms behave similarly. For completeness, we show the results for all three MINLP algorithms in Figure 19.

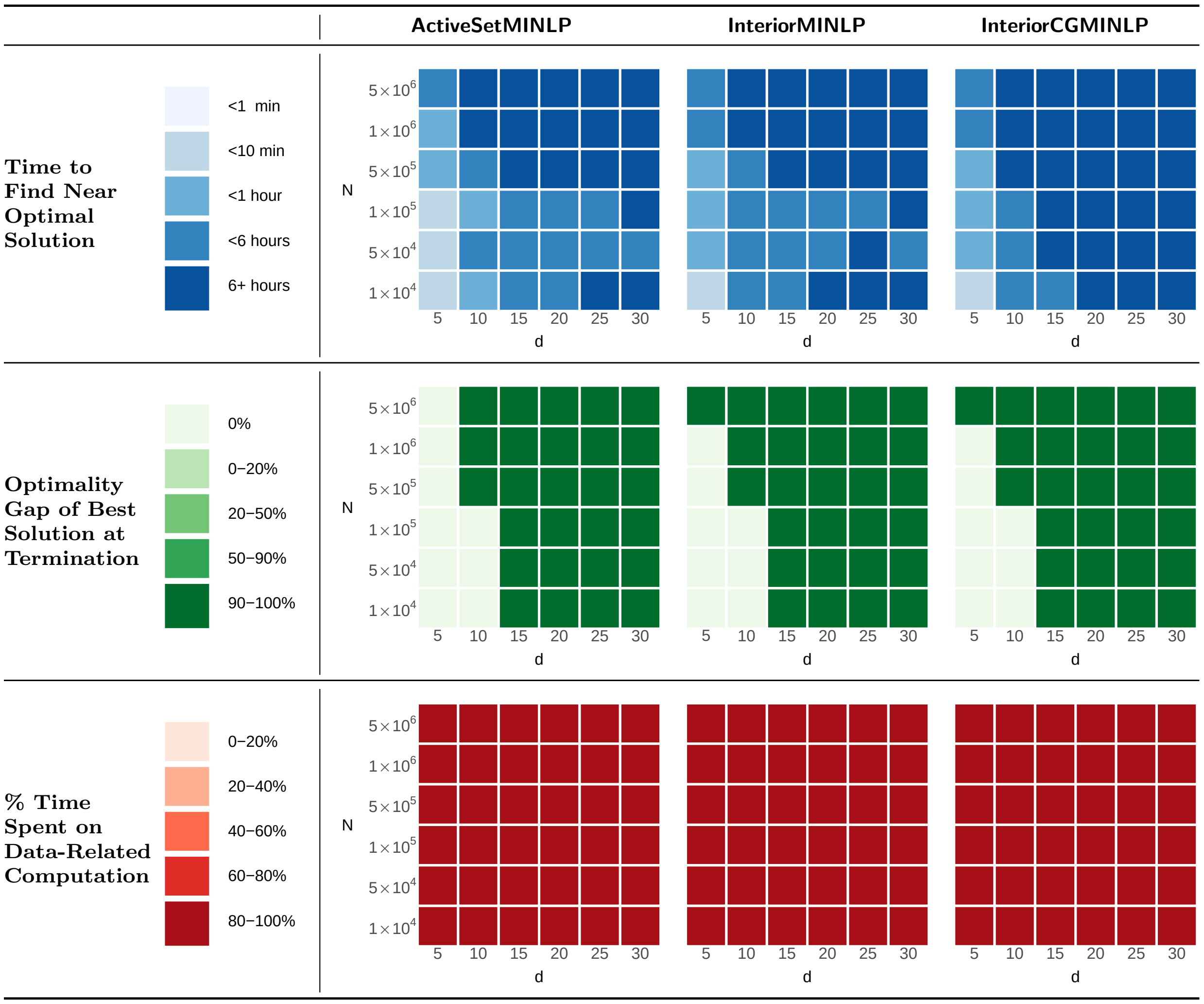

Figure 19: Performance of MINLP algorithms on instances RiskSlimMINLP for simulated datasets with various dimensions d and sample sizes N. All algorithms perform similarly. We report results for ActiveSetMINLP in the main text because it solves the most instances to optimality.