Optimized Risk Scores

Berk Ustun
Massachusetts Institute of Technology
Cambridge, MA, USA
ustunb@mit.edu

Cynthia Rudin
Duke University
Durham, NC, USA
cynthia@cs.duke.edu

ABSTRACT

Risk scores are simple classification models that let users quickly assess risk by adding, subtracting, and multiplying a few small numbers. Such models are widely used in healthcare and criminal justice, but are often built ad hoc. In this paper, we present a principled 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. Our approach can fit optimized risk scores in a way that scales linearly with the sample size of a dataset, provides a proof of optimality, and obeys complex constraints without parameter tuning. We illustrate these benefits through an extensive set of numerical experiments, and an application where we build a customized risk score for ICU seizure prediction.

KEYWORDS

classification; cutting plane methods; mixed integer non-linear programming; risk scores; interpretable models; seizure prediction

1 INTRODUCTION

Risk scores are simple linear classification models to assess risk by adding, subtracting, and multiplying a few small numbers. These models let users make quick predictions, without extensive training, and without use of a computer.

Despite widespread use in medicine and criminal justice, there has been no principled approach to learn risk scores from data. This is partly due to the challenging nature of the learning problem: risk scores need to be rank-accurate (i.e., high AUC), risk-calibrated, sparse, and use small integer coefficients. In practice, domain experts may also require risk scores to satisfy operational constraints before they can be deployed, such as limits on model size ("use at most 5 features"), feature composition ("if the model uses *Hypertension*, then it should also use *Age* $\geq$ *75*"), and prediction ("predicted risk should be lower for males than females").

The extensive set of requirements is best illustrated by the fact that many existing models used in practice are built ad hoc [see e.g., 1]. Existing models are built by a panel of experts (e.g., the $CHADS_2$ score in Figure 1) or by combining multiple heuristics (e.g., by rounding logistic regression coefficients after manual feature selection, as recommended by the U.S. Department of Justice [30]). These approaches may produce risk scores with poor rank-accuracy or risk calibration (see e.g., the validated performance of $CHADS_2$ in [22], and heuristic risk scores in [8]).

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

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

Figure 1: $CHADS_2$ score to assess stroke risk [12]. Such models are widely used for risk assessment in medicine (see www.mdcalc.com) and criminal justice [8, 13, 25, 30].

In this paper, we present a principled approach to learn risk scores by solving a mixed-integer nonlinear program (MINLP) that we call the *risk score problem*. We consider an exact formulation 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 obtained by solving this problem as a *Risk-calibrated Supersparse Linear Integer Model* (RiskSLIM).

Our proposed approach is unique in that it can fit models that are fully optimized for feature selection and small integer coefficients. In addition, it allows users to easily address operational constraints without parameter tuning or post-processing, by directly including these constraints in the MINLP formulation. In light of these benefits, a major goal is to recover the optimal solution to the risk score problem *and* pair this solution with a certificate of optimality. By design, the optimal solution to the risk score problem attains the best performance among all models that satisfy our constraints. By solving this problem to optimality, we therefore end up with a risk score with acceptable performance, or a risk score with unacceptable performance along with a certificate proving that the constraints were overly restrictive.

As we show, solving the risk score problem with an off-the-shelf MINLP solver is time-consuming even on small datasets, as algorithms for generic MINLPs are slowed down by excessive data-related computation in this case. 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 linear approximation of the loss that is much cheaper to evaluate. Cutting plane algorithms have an impressive track record on large-scale learning problems [see 10, 16, 29], as they scale linearly with the number of samples and provide precise control over data-related computation. Unfortunately, previous algorithms were designed under the assumption that the surrogate can be solved to

KDD'17, August 13–17, 2017, Halifax, NS, Canada.

978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098161

optimality at each iteration. While this is perfectly reasonable in a convex setting, it leads cutting plane algorithms to *stall* on non-convex problems, as the time to optimize the surrogate increases exponentially with each iteration. We demonstrate this stalling phenomenon, and present a cutting plane algorithm to overcome this issue, called the *lattice cutting plane algorithm*. The resulting approach allows us to learn optimized risk scores in a way that scales linearly in the number of samples, provides a certificate of optimality, and accommodates non-trivial operational constraints that are often crucial for deployment.

Related Work. Our work is broadly related to new methods for interpretable machine learning [see e.g. 5, 15, 21, 22, 32, 36]. Interpretability has become crucial for models in high-stakes applications [19], as evidenced by new EU regulations that require a "right to an explanation" from algorithmic decision-making tools [14].

We focus on the simple risk assessment models such as those considered by Ertekin and Rudin [9, which uses a Bayesian approach] and Jung et al. [17, which combines stepwise regression, scaling, and rounding]. This class of models generalizes boolean risk models [e.g. 35] when we use binary features and restrict coefficients to $\lambda_j \in \{0, 1\}$. In contrast to these approaches, we use an optimization-based approach that can consistently recover a globally optimal solution, provide a certificate of optimality, address operational constraints, and scale seamlessly in the number of samples.

RiskSLIM models are similar to SLIM models [28, 32, 34, 36] in that the score functions are fully optimized for feature selection, small integer coefficients and operational constraints. However, RiskSLIM models are designed for *risk assessment* and optimize the *logistic loss*. In contrast, SLIM models are designed for *decision-making* and optimize the *0–1 loss*. Optimizing the 0–1 loss results in models that are optimized for accuracy, meaning that SLIM models will not necessarily have high AUC if used for ranking. Optimizing the 0–1 loss is also *NP*-hard, so training SLIM may be challenging for datasets with large sample sizes. In practice, RiskSLIM is better-suited for problems where: (i) users need calibrated probability estimates; (ii) the sample size is large; (iii) users need a model that performs well at several operating points across the ROC curve (e.g. when users want to adjust their decision point on-the-fly).

We solve the risk score problem with a cutting plane algorithm. Such algorithms have been extensively studied by the optimization community [see 3, 18]. Our algorithm uses callbacks in modern MIP solvers to build a cutting plane approximation while performing a branch-and-bound search. It differs from existing algorithms [10, 16, 29] in that it does not *stall* in settings with non-convex regularizers and constraints. Our algorithm may be used to solve other problems with non-convex penalties and constraints, such as ℓ_0-regularized risk minimization [27], or discrete linear classification problems that minimize a convex loss over a small integers [4, 7, 24, 35].

Software and Additional Resources. We provide software to learn optimized risk scores at http://github.com/ustunb/risk-slim. We provide additional details in the full version of this paper [33], which includes new empirical results as well as techniques to improve LCPA by generating feasible solutions, narrowing the optimality gap, and reducing data-related computation. In addition to the seizure prediction application in Section 5, RiskSLIM has also been used to create a screening tool to diagnose adult ADHD [31].

2 PROBLEM STATEMENT

We define the risk score problem as follows. We start with a set of training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where $\boldsymbol{x}_i = [1, x_{i1} \dots x_{id}]^\top \subseteq \mathbb{R}^{d+1}$ is a feature vector and $y_i \in \{-1, +1\}$ is a class label. We consider a score function $\langle \boldsymbol{\lambda}, \boldsymbol{x} \rangle$ where $\boldsymbol{\lambda} \subseteq \mathbb{R}^{d+1}$ is a coefficient vector $[\lambda_0, \lambda_1, \dots, \lambda_d]^\top$ and λ_0 is the intercept. We model the *predicted risk* that example i belongs to the positive class as:

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

In this setup, λ_j represents the *points* for feature j. Given features $\boldsymbol{x}_i$, users tally the points to obtain a *score* $s_i = \langle \boldsymbol{\lambda}, \boldsymbol{x}_i \rangle$, and use the score s_i to estimate predicted risk.

We learn the values of the coefficients from data, by solving a MINLP that we call the *risk score problem* or RiskSlimMINLP:

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

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

Constraint Type	Example
Feature Selection	Choose up to 10 features
Group Sparsity	Include either *Male* or *Female*, not both
Optimal Thresholding	Use at most 3 thresholds for *Age*: $\sum_{k=1}^{100} 1[Age \leq k] \leq 3$
Logical Structure	If *Male* is in model, then also include *Hypertension*
Side Information	Predict $\Pr(y = +1 \mid \boldsymbol{x}) \geq 0.90$ when *Male* = TRUE

Table 1: Operational constraints that can be added to the feasible region of the risk score problem (1).

RiskSlimMINLP aims to capture the *exact* objectives and constraints of risk scores, so that its optimizer attains the minimum logistic loss among feasible models on the training data (provided that C_0 is small enough). In Section 4, we show that models that minimize the logistic loss achieve high AUC and risk calibration on training data, and that this generalizes to test data due to the simplicity of our hypothesis space. There is some theory to explain these results. Specifically, the logistic loss is *strictly proper*, meaning it yields calibrated estimates of predicted risk under the parametric assumption that the true risk can be modeled with the logistic function [26]. In addition, Kotlowski et al. [20] show that a "balanced" logistic loss is a lower bound on 1-AUC, which means that minimizing the logistic loss maximizes a surrogate of AUC.

Using an exact formulation provides an alternative way to set the trade-off parameter C_0:

- If we are given a limit on the model size (e.g. $\|\lambda\|_0 \leq k$), we can add this as a constraint in the formulation and set C_0 to a small

value (e.g. $C_0 = 10^{-8}$). In this case, the optimal solution corresponds to the best model that obeys the model size constraint, provided C_0 is small enough [see 33, for a proof].

- Alternatively, we can choose the model size based on cross-validated (CV) performance. In this case, we would repeat the previous process for $\|\lambda\|_0 \leq k$ for $k = 1 \dots d$. This lets us fit the full range of risk scores (i.e. the full ℓ_0-regularization path) by solving at most d instances of RiskSlimMINLP. In comparison, a standard CV-based approach (i.e. where we treat C_0 as the hyperparameter) is likely to require solving more than d instances as one cannot determine d values of C_0 to return the full range of risk scores *a priori*.

Optimizing RiskSlimMINLP is a difficult computational task given that ℓ_0-regularization, minimizing over integers, and MINLP problems are all *NP*-hard [2]. These worst-case complexity results mean that finding an optimal solution to RiskSlimMINLP may be intractable for high dimensional datasets. As we show, however, RiskSlimMINLP can be solved to optimality for many real-world datasets in minutes, and in a way that scales linearly in N.

Notation and Terminology. We denote the set of feasible values for λ_j as $\mathcal{L}_j = \{\Lambda_j^{\min}, \dots, \Lambda_j^{\max}\} \subset \mathbb{Z}$. We denote the objective of RiskSlimMINLP as $V(\boldsymbol{\lambda}) = l(\boldsymbol{\lambda}) + C_0 \|\boldsymbol{\lambda}\|_0$ and an optimal solution as $\boldsymbol{\lambda}^* \in \operatorname{argmin}_{\boldsymbol{\lambda} \in \mathcal{L}} V(\boldsymbol{\lambda})$. We bound the optimal value as $V(\boldsymbol{\lambda}^*) \in [V^{\min}, V^{\max}]$, and define the *optimality gap* as $1 - (V^{\min}/V^{\max})$.

Solving RiskSlimMINLP *to optimality* means that we have found a solution with an optimality gap of 0.0%. This implies that we have: (i) found the best integer feasible solution to RiskSlimMINLP; and (ii) paired the solution with a lower bound $V^{\min} = V(\boldsymbol{\lambda}^*)$.

We make two following assumptions for clarity of exposition: (i) $\mathbf{0} \in \mathcal{L}$, which ensures that RiskSlimMINLP is always feasible; (ii) the intercept is not regularized, which means the precise version of the RiskSlimMINLP objective is $V(\boldsymbol{\lambda}) = l(\boldsymbol{\lambda}) + C_0 \left\|\boldsymbol{\lambda}_{[1,d]}\right\|_0$ where $\boldsymbol{\lambda} = [\lambda_0, \boldsymbol{\lambda}_{[1,d]}]$.

3 METHODOLOGY

In Algorithm 1, we present a simple cutting plane algorithm that we refer to as CPA. In what follows, we use CPA to discuss the benefits of cutting plane algorithms, and to explain why existing algorithms stall in non-convex settings. We then present a new cutting plane algorithm that does not stall in non-convex settings, and compare the performance of the cutting-plane algorithms with an off-the-shelf MINLP solver in Figure 5.

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

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

Algorithm 1 Cutting Plane Algorithm (CPA)

Input

$(\boldsymbol{x}_i, y_i)_{i=1}^N$	*training data*
$\mathcal{L}$	*constraint set*
C_0	*ℓ_0 penalty parameter*
$\varepsilon^{\text{stop}} \in [0, 1]$	*optimality gap of acceptable solution*

Initialize

$k \leftarrow 0$	*number of cuts*
$\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$	*cutting-plane approximation of loss function*
$(V^{\min}, V^{\max}) \leftarrow (0, \infty)$	*bounds on the optimal value*
$\varepsilon \leftarrow \infty$	*optimality gap*

while $\varepsilon > \varepsilon^{\text{stop}}$ **do**
  $(\theta^k, \boldsymbol{\lambda}^k) \leftarrow$ provably optimal solution to RiskSlimMIP($\hat{l}^k(\cdot)$)
  compute cut parameters $l(\boldsymbol{\lambda}^k)$ and $\nabla l(\boldsymbol{\lambda}^k)$
  $\hat{l}^{k+1}(\boldsymbol{\lambda}) \leftarrow \max\{\hat{l}^k(\boldsymbol{\lambda}), l(\boldsymbol{\lambda}^k) + \langle \nabla l(\boldsymbol{\lambda}^k), \boldsymbol{\lambda} - \boldsymbol{\lambda}^k \rangle\}$ ▷ *update approximation $\forall \boldsymbol{\lambda}$*
  $V^{\min} \leftarrow \theta^k + C_0 \|\boldsymbol{\lambda}^k\|_0$ ▷ *optimal value of RiskSlimMIP is lower bound*
  if $V(\boldsymbol{\lambda}^k) < V^{\max}$ **then**
    $V^{\max} \leftarrow V(\boldsymbol{\lambda}^k)$ ▷ *update upper bound*
    $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^k$ ▷ *update best solution*
  end if
  $\varepsilon \leftarrow 1 - V^{\min}/V^{\max}$
  $k \leftarrow k + 1$
end while

Output: $\boldsymbol{\lambda}^{\text{best}}$ *ε-optimal solution to RiskSlimMINLP*

RiskSlimMIP($\hat{l}(\cdot)$) is a MIP surrogate of RiskSlimMINLP where the loss function $l(\cdot)$ is replaced by the cutting-plane approximation $\hat{l}(\cdot)$:

$$\begin{aligned} \min_{\theta, \boldsymbol{\lambda}} \quad & \theta + C_0 \|\boldsymbol{\lambda}\|_0 \\ \text{s.t.} \quad & \theta \geq \hat{l}(\boldsymbol{\lambda}) \\ & \boldsymbol{\lambda} \in \mathcal{L}. \end{aligned} \tag{2}$$

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

$$\begin{aligned} l(\boldsymbol{\lambda}^t) &= \frac{1}{N} \sum_{i=1}^N \log(1 + \exp(-\langle \boldsymbol{\lambda}^t, y_i \boldsymbol{x}_i \rangle)), \\ \nabla l(\boldsymbol{\lambda}^t) &= \frac{1}{N} \sum_{i=1}^N \frac{-y_i \boldsymbol{x}_i}{1 + \exp(-\langle \boldsymbol{\lambda}^t, y_i \boldsymbol{x}_i \rangle)}. \end{aligned} \tag{3}$$

Cuts can be combined to produce a piecewise linear approximation of the loss function as shown in Figure 2. We denote a *cutting plane approximation* of the loss function built from k cuts as:

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

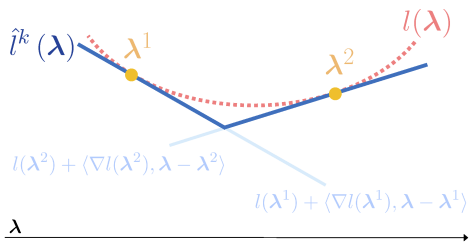

Figure 2: A convex loss function $l(\boldsymbol{\lambda})$ and its cutting plane approximation $\hat{l}^2(\boldsymbol{\lambda})$ built using cuts at the points $\boldsymbol{\lambda}^1$ and $\boldsymbol{\lambda}^2$.

On iteration k, CPA solves the surrogate RiskSlimMIP($\hat{l}^k(\boldsymbol{\lambda})$) whose objective contains the approximate loss $\hat{l}^k(\boldsymbol{\lambda})$. CPA uses

the optimizer of the surrogate $(\theta^k, \boldsymbol{\lambda}^k)$ to: (i) improve $\hat{l}^k(\boldsymbol{\lambda})$ with a new cut at $\boldsymbol{\lambda}^k$; (ii) compute bounds on the optimal value of RiskSlimMINLP to check convergence. The upper bound is set as the objective value of the best solution from all iterations $V^{\text{max}} = \min_{t=1\ldots k} l(\boldsymbol{\lambda}^t) + C_0\|\boldsymbol{\lambda}^t\|_0$. The lower bound is set as the optimal value of the surrogate at the last iteration $V^{\text{min}} = \hat{l}^k(\boldsymbol{\lambda}^k) + C_0\|\boldsymbol{\lambda}^k\|_0$.

CPA converges to an ε-optimal solution of RiskSlimMINLP in a finite number of iterations [see 18, for a proof]. The cutting plane approximation of a convex loss function improves with each cut:

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

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

Key Benefits of Cutting-Plane Algorithms. CPA highlights two major benefits of cutting plane algorithms for empirical risk minimization: (i) scalability in the sample size; (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, running time scales linearly in N for fixed d (see Figure 3). Moreover, since cut parameters are computed in an isolated step (e.g. Step 3 in Algorithm 1), users can easily reduce data-related computation by customizing their implementation to compute cut parameters more efficiently (e.g. via parallelization).

CPA also highlights a unique benefit of cutting plane algorithms in our setting. Specifically, it recovers the optimal solution to the non-linear problem RiskSlimMINLP by iteratively solving a linearized surrogate RiskSlimMIP. In practice, this allows us to fit risk scores with a MIP solver instead of a MINLP solver. As shown in Figure 5, this can substantially improve our ability to solve RiskSlimMINLP since MIP solvers typically exhibit better off-the-shelf performance than MINLP solvers (as MIP solvers have better implementations of branch-and-bound, and MINLP solvers are designed to handle a far more diverse set of optimization problems).

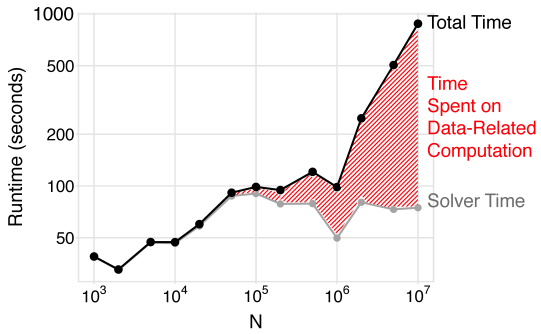

Figure 3: CPA runtime (log-scale) for simulated datasets with $d = 10$ and $N \in [10^3, 10^8]$. As N increases, total running time scales at $O(N)$, which reflects the time to compute cut parameters. Solver time remains roughly constant.

Stalling in Non-Convex Settings. Cutting plane algorithms for empirical risk minimization [10, 16, 29] are similar to CPA in that they solve a surrogate problem at each iteration (i.e., on Step 5 of Algorithm 1). When these algorithms are applied to problems with non-convex regularizers or constraints, the surrogate problems are non-convex and may require an unreasonable amount of time to solve to optimality (especially on on higher-dimensional problems). In practice, this prevents the algorithm from improving the cutting-plane approximation and computing a valid lower bound. We refer to this behavior as *stalling*.

There is no easy fix to prevent cutting plane algorithms such as CPA from stalling in non-convex settings. This is because they need a *provably* optimal solution at each iteration to compute a valid lower bound (i.e., a solution 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, the resulting lower bound could exceed the true optimal value, leading the algorithm to terminate early and return a suboptimal solution.

In Figure 4, we illustrate the stalling of CPA on a RiskSlimMINLP instance where $d = 20$ (in black). As shown, the time to solve RiskSlimMIP increases exponentially with each iteration and CPA stalls on iteration $k = 87$ as it attempts to optimize the surrogate MIP. In this case, the best feasible solution that we recover after 6 hours has a large optimality gap as well as a highly suboptimal loss (which makes sense as the solution optimizes a cutting-plane approximation that uses at most 86 cuts). Given that the value of the loss is closely related to the performance of the model, this means that the risk score we obtain after 6 hours performs poorly.

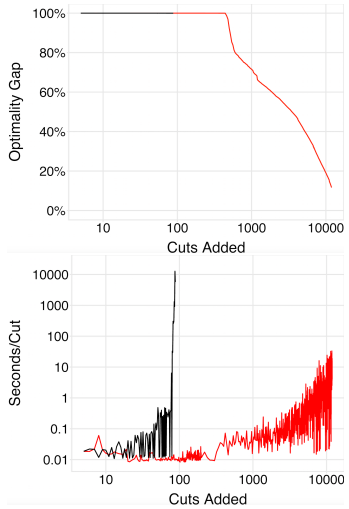

Figure 4: CPA (black) and LCPA (red) on a simulated dataset with $d = 20$ and $N = 50{,}000$. We show the optimality gap (top) and the time to add a new cut (bottom; log-scale) over 6 hours. CPA stalls after adding 86 cuts as the time to optimize RiskSlimMIP increases exponentially. The resulting solution corresponds to a risk score with poor performance. In contrast, LCPA does not stall, finding a near-optimal solution in 9 minutes, and the optimal solution in 234 minutes. LCPA uses the remaining time to reduce the optimality gap.

3.1 Lattice Cutting Plane Algorithm

In order to avoid the stalling behavior of existing cutting-plane algorithms in non-convex settings, we solve the risk score problem using the *lattice cutting plane algorithm* (LCPA; Algorithm 2).

LCPA is a cutting-plane algorithm that recovers the optimal solution to RiskSlimMINLP via *branch-and-bound* (B&B) search. The search recursively splits the feasible region of RiskSlimMINLP into disjoint partitions, discarding partitions that are infeasible or provably suboptimal. LCPA solves a *surrogate linear program* (LP) over each partition. In this approach, the cutting-plane approximation is updated whenever the surrogate LP yields an integer feasible solution. The lower bound is set as the smallest possible value of the surrogate LP over the remaining search region.

As shown in Figure 4, LCPA (in red) does not stall. This is because – unlike CPA – LCPA does not need to optimize a non-convex surrogate to add cuts and compute a valid lower bound. Even so, LCPA retains the key benefits of CPA such as: scalability in the sample size, control over data-related computation, and the ability to use a MIP solver.

In what follows, we describe the main elements of LCPA:

B&B Search. In Algorithm 2, we represent the state of the B&B search using a B&B tree. This tree is composed of nodes (i.e. leaves) in the *node set* $\mathcal{N}$. Each *node* $(\mathcal{P}_n, v_n) \in \mathcal{N}$ consists of a *partition* of the convex hull of constraint set $\mathcal{P}_n \subseteq \text{conv}(\mathcal{L})$, and a lower bound for the optimal value of the surrogate over this partition, v_n.

Each iteration of LCPA starts by removing a node $(\mathcal{P}_n, v_n)$ from the node set $\mathcal{N}$ and solving the surrogate over $\mathcal{P}_n$. The next steps depend on the feasibility of RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{P}_n)$:

- If RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{P}_n)$ yields an integer solution $\boldsymbol{\lambda}^{\text{LP}} \in \mathcal{L}$, LCPA updates the cutting plane approximation $\hat{l}^k(\cdot)$ with a cut at $\boldsymbol{\lambda}^{\text{LP}}$ in Step 8.
- If RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{P}_n)$ yields a continuous solution $\boldsymbol{\lambda}^{\text{LP}} \notin \mathcal{L}$, then LCPA splits the partition $\mathcal{P}_n$ into disjoint subsets $\mathcal{P}'$ and $\mathcal{P}''$. Each subset is paired with the optimal value of the surrogate LP to yield the child nodes $(\mathcal{P}', v^{\text{LP}})$ and $(\mathcal{P}'', v^{\text{LP}})$. The child nodes are added back into $\mathcal{N}$ in Step 18.
- If RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{P}_n)$ is infeasible, the node is discarded.

The search uses rules that are provided by a MIP solver:

- RemoveNode, which takes as input the node set $\mathcal{N}$ and outputs a node $(\mathcal{P}_n, v_n)$ (e.g., the node with the smallest v_n).
- SplitPartition, which takes as input a partition $\mathcal{P}_n$ and the current solution $\boldsymbol{\lambda}^{\text{LP}}$ and outputs disjoint partitions that do not cover $\mathcal{P}_n$ (e.g. split on a fractional component of the solution λ_j^{LP}, which returns $\mathcal{P}' = \{\boldsymbol{\lambda} \in \mathcal{P}_{\text{LP}} \mid \lambda_j^{\text{LP}} \geq \lceil \lambda_j^{\text{LP}} \rceil\}$ and $\mathcal{P}'' = \{\boldsymbol{\lambda} \in \mathcal{P}_{\text{LP}} \mid \lambda_j^{\text{LP}} \leq \lceil \lambda_j^{\text{LP}} \rceil\}$). The output conditions ensure that: (i) the partitions of all nodes in the node set remain disjoint; (ii) the search region shrinks even if the solution to the surrogate is not integer feasible; (iii) the number of nodes is finite.

Convergence. LCPA checks convergence using bounds on the optimal value of RiskSlimMINLP. The upper bound $V^{\max}$ is set as the objective value of the best integer feasible solution in Step 11. The lower bound $V^{\min}$ is set as the smallest lower bound among all nodes in Step 20. This quantity is a lower bound on the optimal value of the surrogate over the *remaining search region* $\bigcup_n \mathcal{P}_n$; that is, the optimal value of RiskSlimLP$(\hat{l}^k(\cdot), \bigcup_n \mathcal{P}_n)$. Thus, $V^{\min}$ improves when we add cuts or reduce the remaining search region.

Each iteration of LCPA reduces the remaining search region as it either finds an integer feasible solution, identifies an infeasible partition, or splits a partition into disjoint subsets. Thus, $V^{\min}$ increases monotonically as the search region becomes smaller, and cuts are added at integer feasible solutions. Likewise, $V^{\max}$ decreases monotonically as the search is guaranteed to find the optimal solution. Since there are a finite number of nodes, LCPA terminates after a finite number of iterations.

Implementation. We implement LCPA using a MIP solver that provides *control callbacks*, such as CPLEX. The solver handles all B&B related steps in Algorithm 2 and control callbacks let update the cutting-plane approximation by intervening in the search. In a basic implementation, we use a control callback to intervene when Algorirthm 2 reaches Step 6. Our code retrieves the integer feasible solution, computes the cut parameters, adds a cut, and returns control back to solver by Step 9.

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	ℓ_0 *penalty parameter*
$\varepsilon^{\text{stop}} \in [0, 1]$	*optimality gap of acceptable solution*
RemoveNode	*rule to pick a node from a node set (provided by MIP solver)*
SplitPartition	*rule to split a partition into disjoint subsets (provided by MIP solver)*

Initialize

$k \leftarrow 0$	*number of cuts*
$\hat{l}^0(\boldsymbol{\lambda}) \leftarrow \{0\}$	*cutting-plane approximation of loss function*
$(V^{\min}, V^{\max}) \leftarrow (0, \infty)$	*bounds on the optimal value*
$\varepsilon \leftarrow \infty$	*optimality gap*
$\mathcal{P}_0 \leftarrow \text{conv}(\mathcal{L})$	*partition for initial node*
$v_0 \leftarrow V^{\min}$	*lower bound for initial node*
$\mathcal{N} \leftarrow \{(\mathcal{P}_0, v_0)\}$	*initial node set*

1: **while** $\varepsilon > \varepsilon^{\text{stop}}$ **do**
2: $(\mathcal{P}_n, v_n) \leftarrow$ RemoveNode $(\mathcal{N})$ ▷ n *is index of removed node*
3: solve RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{P}_n)$
4: $\boldsymbol{\lambda}^{\text{LP}} \leftarrow$ coefficients from optimal solution to RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{P}_n)$
5: $v^{\text{LP}} \leftarrow$ optimal value of RiskSlimLP$(\hat{l}^k(\cdot), \mathcal{P}_n)$
6: **if** optimal solution is integer feasible **then**
7: compute cut parameters $l(\boldsymbol{\lambda}^{\text{LP}})$ and $\nabla l(\boldsymbol{\lambda}^{\text{LP}})$
8: $\hat{l}^{k+1}(\boldsymbol{\lambda}) \leftarrow \max\{\hat{l}^k(\boldsymbol{\lambda}), l(\boldsymbol{\lambda}^{\text{LP}}) + \langle \nabla l(\boldsymbol{\lambda}^k), \boldsymbol{\lambda} - \boldsymbol{\lambda}^{\text{LP}} \rangle\}$ ▷ *update approximation* $\forall \boldsymbol{\lambda}$
9: **if** $v^{\text{LP}} < V^{\max}$ **then**
10: $V^{\max} \leftarrow v^{\text{LP}}$ ▷ *update lower bound*
11: $\boldsymbol{\lambda}^{\text{best}} \leftarrow \boldsymbol{\lambda}^{\text{LP}}$ ▷ *update best solution*
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: $k \leftarrow k + 1$
15: **else if** optimal solution is not integer feasible **then**
16: $(\mathcal{P}', \mathcal{P}'') \leftarrow$ SplitPartition$(\mathcal{P}_n, \boldsymbol{\lambda}^{\text{LP}})$ ▷ $\mathcal{P}', \mathcal{P}''$ *are disjoint subsets of* $\mathcal{P}_n$
17: $(v', v'') \leftarrow (v^{\text{LP}}, v^{\text{LP}})$ ▷ v^{LP} *is lower bound for* $\mathcal{P}', \mathcal{P}''$
18: $\mathcal{N} \leftarrow \mathcal{N} \cup \{(\mathcal{P}', v'), (\mathcal{P}'', v'')\}$ ▷ *add child nodes to* $\mathcal{N}$
19: **end if**
20: $V^{\min} \leftarrow \min_{\mathcal{N}} v_s$ ▷ *lower bound is smallest lower bound among nodes in* $\mathcal{N}$
21: $\varepsilon \leftarrow 1 - V^{\min}/V^{\max}$ ▷ *update optimality gap*
22: **end while**

Output: $\boldsymbol{\lambda}^{\text{best}}$ — ε*-optimal solution to RiskSlimMINLP*

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

$$\begin{aligned} \min_{\theta, \boldsymbol{\lambda}, \boldsymbol{\alpha}} \quad & \theta + C_0 \sum_{j=1}^{d} \alpha_j \\ \text{s.t.} \quad & \boldsymbol{\lambda} \in \mathcal{P} \\ & \theta \geq \hat{l}(\boldsymbol{\lambda}) \\ & \alpha_j = \max(\lambda_j, 0)/\Lambda_j^{\max} + \min(\lambda_j, 0)/\Lambda_j^{\min} \text{ for } j = 1 \ldots d. \end{aligned} \tag{4}$$

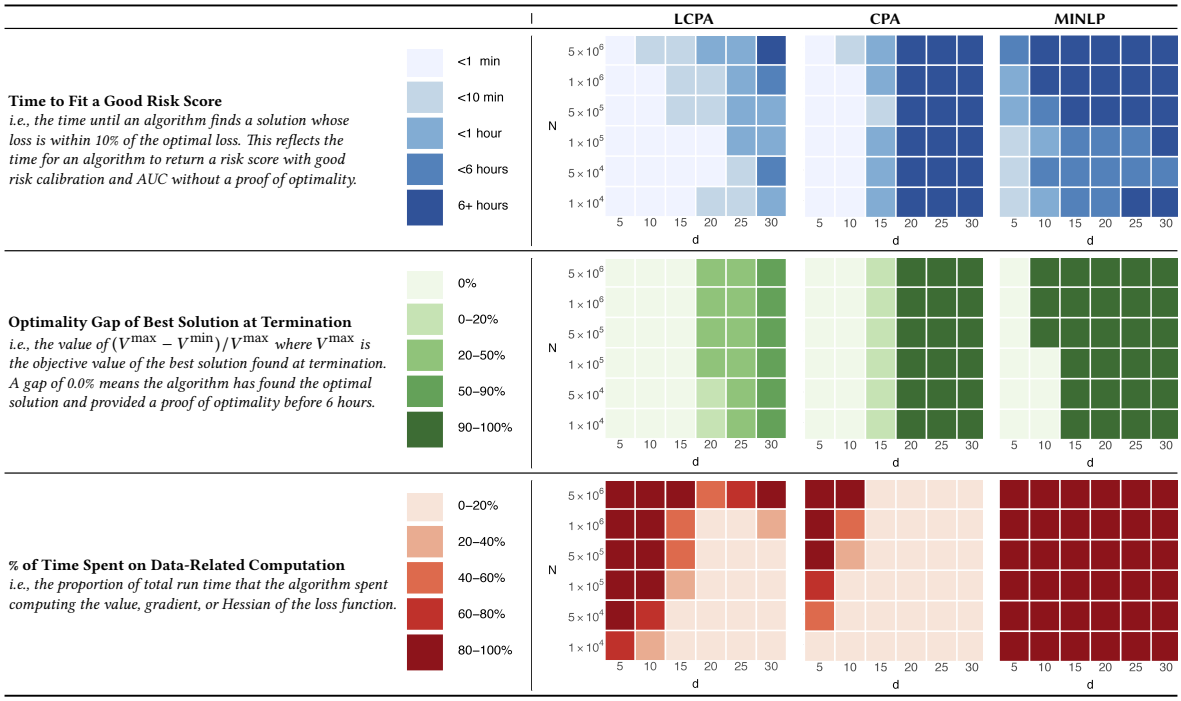

Figure 5: Performance of LCPA, CPA, and a commercial MINLP solver (Artelsys Knitro) on hard instances of RiskSlimMINLP for simulated datasets [details in 33]. MINLP fails to produce good risk scores on instances with large d or N as it struggles with data-related computation. CPA and LCPA scale linearly in N when d is fixed: if they solve an instance for a given d, then they also solve instances for larger N in $O(N)$ additional time. CPA stalls when $d \geq 15$ and returns low-quality risk scores when $d \geq 20$. In contrast, LCPA does not stall, and recovers a good model in all cases.

4 BENCHMARKING

Datasets. We used 6 datasets shown in Table 2. We selected these datasets so as to vary the size and types of variables (`adult`, `mammo`, `mushroom`, `spambase`), and to illustrate potential applications of risk scores (`arrest`, `bank`). All datasets are available at the UCI Repository [23], other than `arrest`, which can be requested from ICPSR. We processed each dataset by binarizing all categorical features and some real-valued features. For reproducibility, we include all processed UCI datasets at github.com/ustunb/risk-slim [see 36, for code to process `arrest`].

Methods. For each dataset, we fit a risk score with integer coefficients $\lambda_j \in [-5, 5]$ and model size $\|\boldsymbol{\lambda}\|_0 \leq 5$ to match the form of models used in practice [e.g., 12]. We use the following methods:

- RiskSLIM: We formulate an instance of RiskSlimMINLP where: $\lambda_0 \in \{-100, \ldots, 100\}$; $\lambda_j \in \{-5, \ldots, 5\}$; and $\|\boldsymbol{\lambda}\|_0 \leq 5$. We set C_0 to a small value (10^{-8}) to recover the best model under these constraints [33]. We solve each instance using LCPA along with the improvements in [33]. We use CPLEX 12.6.3 on a 3.33 GHz CPU with 16 GB RAM and cap runtime to 20 minutes.
- PLR (*Penalized Logistic Regression*): We use the glmnet package [11] to fit logistic regression models with a combined $\ell_1 + \ell_2$ penalty. We add constraints to bound $\lambda_j \in [-5, 5]$, and fit models for 1,100 free parameter instances: 11 values of the mixing parameter $\{0.0, 0.1, \ldots, 1.0\} \times 100$ values of the regularization penalty (chosen by glmnet). These free parameters mean that PLR also covers the following variants of logistic regression as special cases: standard logistic regression (no penalty); Lasso (pure ℓ_1-penalty); and Ridge (pure ℓ_2-penalty).
- Rd (*Naïve Rounding*): We fit a pool of models with PLR. For each model in the pool, we round each coefficient to the nearest integer in $\{-5, \ldots, 5\}$ by setting $\lambda_j \leftarrow \lceil \min(\max(\lambda_j, -5), 5) \rfloor$. We round the intercept to the nearest integer by setting $\lambda_0 \leftarrow \lceil \lambda_0 \rfloor$.
- RsRd (*Rescaled Rounding*): We fit a pool of models with PLR. For each model in the pool, we rescale coefficients so that the largest coefficient is ±5, then round to the nearest integer (i.e. $\lambda_j \rightarrow \lceil \gamma \lambda_j \rfloor$ where $\gamma = 5/\max_j |\lambda_j|$). Rescaling aims to prevent rounding coefficients to zero when $|\lambda_j| < 0.5$ for many j.

Metrics. We evaluate all models in terms of *risk-calibration* (measured by CAL) and *rank accuracy* (measured by AUC). We use *reliability diagrams* to see how the predicted risk matches the observed risk at each score. We estimate the observed risk at each score s as $p(s) = \frac{1}{|\{i:s_i=s\}|} \sum_{i:s_i=s} 1[y_i = +1]$ and summarize calibration over the full reliability diagram using the *calibration error* $\text{CAL} = \frac{1}{N} \sqrt{\sum_s \sum_{i:s_i=s} (p_i - p(s))^2}$.

Model Selection. We use nested 5-CV to set the parameters of the final model and evaluate its predictive performance. The free parameters of the final model reflect an instance that: (i) satisfies the model size constraint; (ii) maximizes the 5-CV mean test AUC. We

do not tune parameters for RiskSLIM since we include the model size constraint in the coefficient set.

4.1 Results

Performance. As shown in Table 2, RiskSLIM models have the best test CAL on 6/6 datasets and the best test AUC on 5/6 datasets. In comparison, models built heuristically perform far worse in terms of test CAL, and slightly worse in terms of test AUC. We can explain these results by observing that: (i) models that attain low values of the logistic loss have good risk calibration [see also 6]; (ii) as we are fitting from a simple class of models, risk scores generalize well (i.e. test CAL/AUC is close to training CAL/AUC). Since RiskSLIM models optimize the loss over exact constraints on model form, they attain minimal or near-minimal values of the loss. Thus, they perform well in terms of training CAL as per (i), and in terms of test CAL as per (ii).

Computation. Although the risk score problem is *NP*-hard, we fit RiskSLIM models with small optimality gaps in ≤20 minutes. Our approach also has some practical benefits that are difficult to measure. In particular, it can build and evaluate predictive performance without the need for parameter tuning and nested CV, meaning that we had to fit a total of 6 models. In comparison, the baseline methods do require parameter tuning and nested CV, which required training and processing over 33,000 models. Thus, even as baseline methods are much faster to run for a single set of parameters, it may take far longer to train these models depending on the post-processing techniques that are used.

Optimality Gaps. RiskSLIM is the only method to pair models with a measure of optimality. In practice, small optimality gaps are valuable because they suggest we have fit the best model in our model class. Thus, if a risk score with a small optimality gap performs poorly, we can attribute the poor performance to a restrictive model class and improve performance by considering models with more terms or larger coefficients. In contrast, heuristic methods do not provide such a guarantee, so when a risk score performs poorly, we cannot tell if this is because the constraints are overly restrictive or because we used an approach that cannot find the best possible model.

Pitfalls and Best Practices for Heuristics. Our results show that the performance of risk scores built heuristically can vary significantly based on post-processing techniques, constraints on model form, and the range of features. In some cases, risk scores built by rounding coefficients perform well (e.g. RD on `bank`). In others, performance can falter (e.g. RD on `spambase`). In practice, performance issues are often overlooked as common heuristics result in good AUC but poor CAL [e.g. the rescaling in RsRD, used by 25, 30] and summary statistics such as CAL and AUC conceal local issues over the entire reliability diagram and ROC curve (see e.g. Figure 7). To mitigate these issues, we recommend the following practices when using or designing heuristics:

- *Select models after rounding.* If we selected a final model from the pool of PLR models before rounding the coefficients, we could greatly alter the loss and thus reduce performance. To mitigate this risk, we first round the coefficients of all models in the pool, and then select among the rounded models.
- *Avoid scaling.* Rescaling coefficients may improve AUC but drastically reduces CAL. This is because the logistic loss is not scale invariant (see e.g. the reliability diagram for RsRD in Figure 7). The decrease in CAL due to scaling is reflected by the much higher values of the loss for RsRD in Table 2.
- *Select models that optimize K-CV AUC instead of K-CV CAL.* We compared both procedures. Choosing a model that optimizes the *K*-CV CAL leads to models with slightly better CAL but far worse AUC. This is because trivial and near-trivial models have low CAL on problems with class imbalance.
- *Binarize real-valued features.* When datasets contain real-valued features (e.g. `spambase`), PLR may assign small coefficients to features with large values. In such cases, rounding can greatly impact performance by removing features such that $|\lambda_j| < 0.5$. This issue is best addressed by binarizing: rescaling coefficients before rounding affects calibration; normalizing reduces usability as it requires users must also normalize when using the model.

These recommendations are for heuristics only. RiskSLIM does not need them.

Dataset	Metric	PLR	RD	RsRD	RiskSLIM
	test cal	5.5%	4.3%	9.1%	2.6%
`adult`	**test auc**	0.817	0.830	0.830	0.854
N = 32561	**model size**	4	4	4	5
d = 36	**loss value**	0.451	0.417	0.484	0.385
	optimality gap	-	-	-	9.7%
	test cal	7.5%	5.7%	20.8%	1.7%
`arrest`	**test auc**	0.700	0.691	0.691	0.697
N = 22530	**model size**	5	5	5	5
d = 48	**loss value**	0.638	0.626	1.282	0.609
	optimality gap	-	-	-	4.0%
	test cal	2.2%	1.4%	9.5%	1.3%
`bank`	**test auc**	0.725	0.759	0.759	0.760
N = 41188	**model size**	2	5	5	5
d = 57	**loss value**	0.339	0.289	0.953	0.289
	optimality gap	-	-	-	3.5%
	test cal	7.3%	8.1%	15.3%	5.0%
`mammo`	**test auc**	0.845	0.845	0.845	0.843
N = 961	**model size**	3	3	3	5
d = 14	**loss value**	0.482	0.480	0.624	0.469
	optimality gap	-	-	-	0.0%
	test cal	20.9%	12.3%	6.5%	1.8%
`mushroom`	**test auc**	0.976	0.973	0.977	0.989
N = 8124	**model size**	5	5	5	5
d = 113	**loss value**	0.362	0.200	0.162	0.069
	optimality gap	-	-	-	0.0%
	test cal	10.5%	24.2%	23.6%	11.7%
`spambase`	**test auc**	0.823	0.908	0.862	0.928
N = 4601	**model size**	4	5	5	5
d = 57	**loss value**	0.553	0.472	5.670	0.349
	optimality gap	-	-	-	27.8%

Table 2: Performance of risk scores with $\|\lambda\|_0 \leq 5$ and $\lambda_j \in \{-5 \dots 5\}$. Here: *test cal* is the 5-CV mean test CAL; *test auc* is the 5-CV mean test AUC; *model size, loss value* and *optimality gap* pertain to the final model fit using the entire dataset.

`arrest`
test cal 1.7%
train cal 2.6%
test auc 0.697
train auc 0.701

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

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

`bank`
test cal 1.3%
train cal 1.1%
test auc 0.760
train auc 0.760

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

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

Figure 6: RiskSLIM models for `arrest` and `bank`. The `arrest` model predicts the risk that a prisoner is arrested within 3 years of release. The `bank` model predicts the risk that a client opens a bank account after a marketing call.

5 ICU SEIZURE PREDICTION

Seizure prediction in the ICU is a difficult problem. Current practice is based on *continuous electroencephalography* (cEEG), which is a technique to monitor electrical activity in the brain by means of electrodes. Clinicians are trained to recognize a large set of cEEG patterns, only some of which may be predictive. The characteristics of cEEG patterns are then used to assess seizure risk, and to decide if patients require a medical intervention, which may be dangerous, or further monitoring, which is expensive. In what follows, we discuss a collaboration with the Massachusetts General Hospital (MGH) where we built a customized risk score to inform such decisions.

Dataset. We used a dataset of cEEG recordings from the Critical Care EEG Monitoring Research Consortium. It contains $N = 5427$ patient records and $d = 87$ variables related to medical history, secondary symptoms, and characteristics of 5 well-known cEEG patterns: *lateralized periodic discharges* (LPD); *lateralized rhythmic delta* (LRDA); *generalized periodic discharges* (GPD); *generalized rhythmic delta* (GRDA); and *bilateral periodic discharges* (BiPD). Here, $y_i = +1$ if a patient in the ICU has a seizure in the next 24 hours. The problem is imbalanced with $\Pr(y_i = +1) = 12.5\%$.

Model Requirements. Our collaborators at MGH wanted a model that was risk-calibrated, sparse, aligned with domain knowledge, and let clinicians make predictions without checking too many cEEG patterns. To address these requirements, they specified several operational constraints:

- *Limited Model Size*: The model had to use at most 4 variables so that it would be easy to validate, and use in an ICU.
- *Monotonicity*: The model had to obey monotonicity constraints for well-known risk factors (e.g. it could not suggest that prior seizures reduce seizure risk).
- *No Linear Dependencies*: The model could not include linearly dependent variables (e.g. it could not include *Male* and *Female*).
- *Specific cEEG Patterns or Any cEEG Pattern*: The dataset included variables for specific cEEG patterns (e.g. *MaxFrequencyLPD*) and any cEEG pattern (e.g. *MaxFrequencyAnyPattern*). The model had to use variables for specific patterns or any pattern, not both.
- *Frequency in Continuous Encoding or Binary Encoding*: The dataset included two kinds of variables to measure the frequency of a cEEG pattern: (i) a real-valued variable (e.g. *MaxFrequencyLPD* $\in [0, 3.0]$); (ii) 7 binary variables (e.g. *MaxFrequencyLPD* ≤ 0.5 *Hz*). Models could use the real-valued variable or the binary variables. To prevent clinicians from having to check multiple thresholds, models had to use ≤ 2 binary variables for each cEEG pattern.

Methods. We used the methods and metrics described in Section 4, which we adapted to address operational constraints as follows. We fit a RiskSLIM model by solving RiskSlimMINLP with the operational constraints. This MINLP had 20 additional constraints, 2 additional variables, and was solved to optimality in ≤ 20 minutes. The baseline methods had built-in mechanisms for sign constraints but needed tuning to handle the remaining constraints. We used nested 5-CV and selected a final model that: (i) obeyed all operational constraints and (ii) maximized the mean 5-CV test AUC.

Results. Table 3 illustrates the performance benefits of an optimization based approach in a constrained setting: the RiskSLIM model has a test CAL/AUC of 2.5%/0.801 while the best model from the baseline methods has a test CAL/AUC of 3.7%/0.738.

As shown in Figure 7, models may have important differences in risk calibration over the full reliability diagram. RiskSLIM risk estimates are roughly monotonic and stable. Rd risk estimates are unstable and non-monotonic. RsRd are skewed towards extreme values as a result of scaling. As noted by our collaborators, the non-monotonicity of Rd and RsRd is problematic as it suggests patients with a score of 3.5 may have more seizures compared to patients with a score of 4.0.

Figure 7 also highlights some of the usability benefits of linear models with small integer coefficients. When input variables belong to a small discrete set, scores also belong to a small discrete set. This reduces the number of operating points on the ROC curve and reliability diagram and makes it easy to pick an operating point. When input variables are binary, risk scores have yet another benefit in that the decision rule at each operating point is a boolean function. For the RiskSLIM model, for example, the decision rule predict $\hat{y}_i = +1$ if score ≥ 2 is equivalent to the boolean function:

$$\begin{aligned}\text{predict } \textit{Seizure} \text{ if } \; & \textit{AnyBriefRhythmicDischarge} \\ & \vee \textit{PatternsIncludeLPD} \\ & \vee (\textit{AnyPriorSeizure} \wedge \textit{EpiletiformDischarge}).\end{aligned}$$

Small integer coefficients make it easy to extract such rules by listing the conditions when the score exceeds the threshold. In order to illustrate this, we show the score function of the PLR model in Table 3 below.

$$\begin{aligned}\text{score} = & -2.35 \\ & + 0.91 \; \textit{PatternsIncludeBiPD or LRDA or LPD} \\ & + 0.03 \; \textit{AnyPriorSeizure}. \\ & + 0.61 \times \; \textit{MaxFrequencyLPD}\end{aligned}$$

In this case, it is much harder for users to extract a boolean function since the score function uses real-valued coefficients, and computing the score requires multiplication due to a non-binary feature, *MaxFrequencyLPD*.

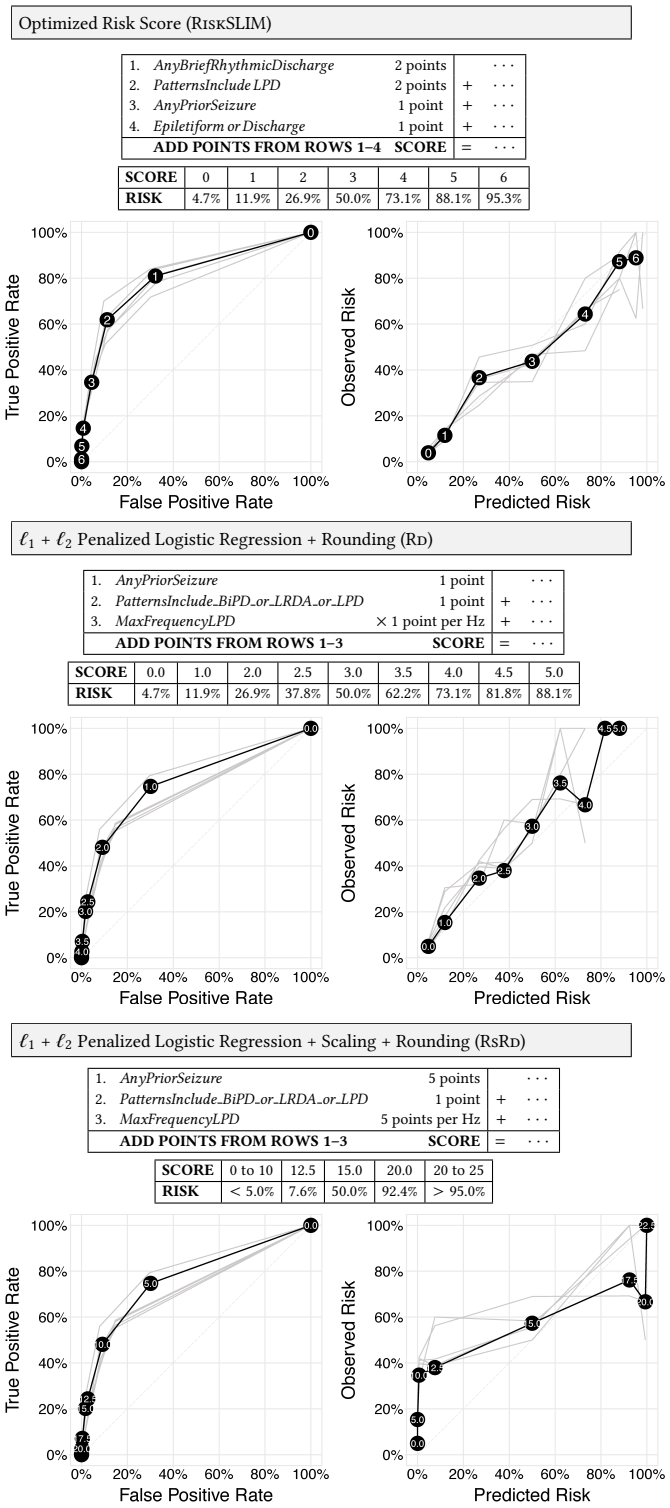

Figure 7: Risk scores, ROC curves, and reliability diagrams for RiskSLIM, RD and RSRD. We show the final model on training data in black, and fold-based models on test data in grey.

Method	OBJECTIVE		CONSTRAINTS			OTHER INFORMATION			
	Test CAL	Test AUC	Model Size	Instances Trained	% Feasible Instances	Loss Value	Opt. Gap	Train CAL	Train AUC
RiskSLIM	2.5% 1.9 - 3.4%	0.801 0.758 - 0.841	4	1	100%	0.293	0.0%	2.0%	0.806
PLR	4.5% 3.4 - 6.6%	0.731 0.712 - 0.772	2	1100	0%	0.326	-	3.9%	0.731
RD	3.7% 2.9 - 5.0%	0.738 0.712 - 0.805	3	1100	12%	0.313	-	1.9%	0.767
RSRD	11.5% 10.7 - 12.7%	0.738 0.712 - 0.805	3	1100	11%	1.003	-	10.3%	0.767

Table 3: Performance of models built with the methods in Section 4. We show the 5-CV mean test CAL/AUC (top) and the 5-CV min-max (bottom). An *instance* is a unique combination of free parameters for a given method.

6 DISCUSSION

Our goal in this paper was to develop a principled approach to learn simple risk scores from data. To this end, we formulated an exact optimization problem to fit risk scores that were optimized for feature selection, small integer coefficients, and operational constraints. We then solved this problem with a new cutting plane algorithm that does not stall in non-convex settings and whose running time scales linearly with the number of samples in a dataset.

As we showed, our approach resulted in risk scores with improved risk-calibration and AUC – especially in constrained settings. In addition, it had practical benefits in that: (i) it can address operational constraints without parameter tuning, which greatly reduces the number of models needed to build and evaluate a risk score; (ii) it pairs models with an optimality gap to determine if poor performance is due to the model class or the fitting process.

Interesting directions for future work include extending our approach to build risk scores with various kinds of operational constraints (e.g. fairness constraints), and using LCPA to fit other kinds of supervised learning models with non-convex regularizers or constraints.

ACKNOWLEDGMENTS

We gratefully acknowledge support from Siemens, Phillips, and Wistron, and the NSF. We would like to thank Paul Rubin for helpful discussions, and our collaborators Aaron Struck and Brandon Westover for their guidance on the seizure prediction problem.

REFERENCES

[1] Philip Bobko, Philip L Roth, and Maury A Buster. 2007. The usefulness of unit weights in creating composite scores. *Organizational Research Methods* 10, 4 (2007), 689–709.

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

[3] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[4] Emilio Carrizosa, Amaya Nogales-Gómez, and Dolores Romero Morales. 2016. Strongly agree or strongly disagree?: Rating features in Support Vector Machines. *Information Sciences* 329 (2016), 256–273.

[5] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1721–1730.

[6] Rich Caruana and Alexandru Niculescu-Mizil. 2004. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining*. ACM, 69–78.

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

[8] Grant Duwe and KiDeuk Kim. 2016. Sacrificing accuracy for transparency in recidivism risk assessment: the impact of classification method on predictive performance. *Corrections* (2016), 1–22.

[9] Şeyda Ertekin and Cynthia Rudin. 2015. A Bayesian Approach to Learning Scoring Systems. *Big Data* 3, 4 (2015), 267–276.

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

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

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

[13] Sharad Goel, Justin M Rao, and Ravi Shroff. 2015. 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).

[14] Bryce Goodman and Seth Flaxman. 2016. EU regulations on algorithmic decision-making and a " right to explanation". *arXiv preprint arXiv:1606.08813* (2016).

[15] Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander Van Esbroeck. 2016. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research* 17, 109 (2016), 1–47.

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

[17] Jongbin Jung, Connor Concannon, Ravi Shroff, Sharad Goel, and Daniel G Goldstein. 2017. Simple rules for complex decisions. (2017).

[18] James E Kelley, Jr. 1960. The cutting-plane method for solving convex programs. *J. Soc. Indust. Appl. Math.* 8, 4 (1960), 703–712.

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

[20] Wojciech Kotlowski, Krzysztof J Dembczynski, and Eyke Huellermeier. 2011. Bipartite ranking through minimization of univariate loss. In *Proceedings of the 28th International Conference on Machine Learning*. 1113–1120.

[21] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1675–1684.

[22] Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. 2015. Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model. *Annals of Applied Statistics* 9, 3 (2015), 1350–1371.

[23] M. Lichman. 2013. UCI Machine Learning Repository. (2013).

[24] Dmitry Malioutov and Kush Varshney. 2013. Exact Rule Learning via Boolean Compressed Sensing.. In *Proceedings of International Conference on Machine Learning*. 765–773.

[25] Pennsylvania Commission on Sentencing. 2012. *Interim Report 4: Development of Risk Assessment Scale*. Technical Report.

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

[27] Toshiki Sato, Yuichi Takano, Ryuhei Miyashiro, and Akiko Yoshise. 2016. Feature subset selection for logistic regression via mixed integer optimization. *Computational Optimization and Applications* (2016), 1–16.

[28] William Souillard-Mandar, Randall Davis, Cynthia Rudin, Rhoda Au, David J Libon, Rodney Swenson, Catherine C Price, Melissa Lamar, and Dana L Penney. 2016. Learning classification models of cognitive conditions from subtle behaviors in the digital clock drawing test. *Machine learning* 102, 3 (2016), 393–441.

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

[30] U.S. Department of Justice. 2005. The mathematics of risk classification: changing data into valid instruments for juvenile courts. (2005).

[31] Berk Ustun, Lenard A Adler, Cynthia Rudin, Stephen V Faraone, Thomas J Spencer, Patricia Berglund, Michael J Gruber, and Ronald C Kessler. 2017. The World Health Organization Adult Attention-Deficit/Hyperactivity Disorder Self-Report Screening Scale for DSM-5. *Jama psychiatry* 74, 5 (2017), 520–526.

[32] Berk Ustun and Cynthia Rudin. 2015. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning* (2015), 1–43.

[33] Berk Ustun and Cynthia Rudin. 2016. Learning optimized risk scores on large-scale datasets. *arXiv preprint arXiv:1610.00168* (2016).

[34] Berk Ustun, M Brandon Westover, Cynthia Rudin, and Matt T Bianchi. 2016. Clinical prediction models for sleep apnea: the importance of medical history over symptoms. *Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine* 12, 2 (2016), 161.

[35] Tong Wang, Cynthia Rudin, F Doshi, Yimin Liu, Erica Klampfl, and Perry MacNeille. 2015. Bayesian Or's of And's for interpretable classification with application to context aware recommender systems. (2015).

[36] Jiaming Zeng, Berk Ustun, and Cynthia Rudin. 2016. Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society: Series A* (2016).