Submitted 9/18; Revised 10/19; Published 11/19

Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals

Andrew Cotter ACOTTER@GOOGLE.COM
Heinrich Jiang HEINRICHJ@GOOGLE.COM
Maya Gupta MAYAGUPTA@GOOGLE.COM
Serena Wang SERENAWANG@GOOGLE.COM
Taman Narayan TAMANN@GOOGLE.COM
Google Research
1600 Amphitheatre Pkwy
Mountain View, CA, USA
Seungil You SEUNGIL.YOU@GMAIL.COM
Kakao Mobility
Seongnam-si, Gyeonggi-do, South Korea
Karthik Sridharan SRIDHARAN@CS.CORNELL.EDU
Cornell University
Ithaca, NY, USA

Editor: Shivani Agarwal

Abstract

We show that many machine learning goals can be expressed as "rate constraints" on a model's predictions. We study the problem of training non-convex models subject to these rate constraints (or other non-convex or non-differentiable constraints). In the non-convex setting, the standard approach of Lagrange multipliers may fail. Furthermore, if the constraints are non-differentiable, then one cannot optimize the Lagrangian with gradient-based methods. To solve these issues, we introduce a new "proxy-Lagrangian" formulation. This leads to an algorithm that, assuming access to an optimization oracle, produces a stochastic classifier by playing a two-player non-zero-sum game solving for what we call a semi-coarse correlated equilibrium, which in turn corresponds to an approximately optimal and feasible solution to the constrained optimization problem. We then give a procedure that shrinks the randomized solution down to a mixture of at most $m+1$ deterministic solutions, given m constraints. This culminates in a procedure that can solve non-convex constrained optimization problems with possibly non-differentiable and non-convex constraints, and enjoys theoretical guarantees. We provide extensive experimental results covering a broad range of policy goals, including various fairness metrics, accuracy, coverage, recall, and churn.

Keywords: constrained optimization, non-convex, fairness, churn, swap regret, non-zero-sum game

1. Introduction

We seek to provide better ways to control machine learning to meet societal, legal, and practical goals, and to take advantage of different kinds of side information and intuition that practitioners may have about their machine learning problem. In this paper, we show that many real-world goals and side information can be expressed as constraints on the model's prediction rates on different datasets, which we refer to as *rate constraints*, turning training into a constrained optimization problem. A simple example of a rate constraint is requiring a binary classifier to make positive predictions on at least 80% of examples. One can incorporate that rate constraint into training. That is, if h is a classifier parameterized by $\theta \in \Theta$, $\{(x_j, y_j)\}$ is a classifier training set with $j = 1, \ldots, N$, ℓ is the loss, and I is the usual indicator, the constrained optimization to minimize the empirical risk subject to this 80% positive rate constraint is:

$$\begin{aligned} &\min_{\theta} \frac{1}{N} \sum_{j=1}^{N} \ell(h(x_j;\theta), y_j) \\ &\text{s.t. } \frac{1}{N} \sum_{j=1}^{N} I_{h(x_j;\theta) \geq 0} \ \geq \ 0.8. \end{aligned} \tag{1}$$

1.1. The Broad Applicability Of Rate Constraints

One can express a surprisingly large set of real-world goals using rate constraints. Here we preview some categories of goals, with more details in Section 3.

Fairness: Many fairness goals can be expressed as rate constraints, including the popular fairness goal of *statistical parity*. For example, one can constrain a classifier so that its positive prediction rate for men and women differs by no more than ten percent. Other fairness goals that can be expressed as rate constraints are *equal opportunity* and *equal odds* (Hardt et al., 2016). In Section 3 we introduce some other fairness goals that we have encountered in real-world problems but have not previously seen in the machine learning literature, such as *no worse off*.

Performance Measures: Some standard performance metrics can be expressed as rate constraints, for example, one can lower-bound the recall, or constrain the model to have some minimal accuracy on specific slices of the data. Precision and win-loss ratio (WLR) compared to a baseline classifier can be expressed with rate constraints; however, there are some caveats about how satisfying constraints on these metrics will generalize to test samples (details below). AUC can be approximated as a set of rate constraints, using the approximation proposed in Eban et al. (2017).

Churn: Given a current classifier, the churn of a new classifier on a specific distribution of examples is the probability that the new classifier decision differs from the current classifier's decision (Cormier et al., 2016). Reducing classifier *churn* is important in many practical machine learning systems to improve overall system stability and to make changes easier to measure and test (Cormier et al., 2016). Churn can be expressed with rate constraints (Goh et al., 2016), thus one can constrain the churn of a new classifier to some desired level.

Multiple Training Datasets: Sometimes one has multiple labeled sets of varying quality and size. For example, one might have only a small set of data labeled by experts, but a large set of noisy training data. One can train the classifier to minimize errors on large noisy data, with a rate constraint that it must achieve at least a certain accuracy on the small expert-labeled dataset.

Unlabeled Datasets: Many of the rate constraints we discuss do not require labels, such as constraints on the positive rate of the classifier, or churn constraints. Training with these rate constraints enables one to take advantage of large unlabeled datasets, which are cheaper to obtain than labeled data.

1.2. Why Constrain? Why Not Penalize?

Rather than expressing each goal as a constraint during training, one could instead add an additive penalty to the loss. If there are multiple goals, one could use a linear combination of such additive penalties. However, the penalty approach requires the practitioner to determine the right weight for each penalty. In practice we find this gets difficult fast: there may be multiple constraints, possibly defined over multiple datasets, and the weights on the multiple penalties may interact with each other. In conclusion, it may be difficult to determine how much to weight each penalty.

We have found that specifying goals using constraints is in practice a cleaner and easier interface for practitioners. The key reason is that a constraint has an absolute meaning, making it possible for a practitioner to specify their goal as a constraint without regard for the presence of other constraints. For example, the meaning of a constraint that the classifier have 80% recall in India does not change if someone else adds other locale-specific constraints on the classifier. We also found that using hard constraints leads to a more understandable machine learning model because it is clearer what the model was trained to do, and it is clearer to measure and verify whether the training sufficiently achieved the practitioner's intent for each individual goal.

1.3. Training With Constraints:

Training with rate constraints poses some difficult challenges:

1. **Non-convex:** For nonlinear function classes, such as neural networks, the objective and constraint functions will be non-convex, *even with convex loss functions*.
2. **Non-differentiable:** Rate constraints are linear combinations of positive and negative classification rates. That is, they are made up of *indicator* functions (0-1 losses), and therefore have zero gradients almost everywhere.
3. **Data-dependent:** The constraints are data-dependent, so for large datasets it may be impractical to fully evaluate the constraints at every iteration—we'd prefer to work with *minibatches*.

While our motivating optimization problem is training with rate constraints, the analysis and algorithms we present will apply generally to constrained optimization problems of the form:

$$\begin{aligned} &\min_{\theta\in\Theta} g_0(\theta) \\ &\text{s.t. } \; g_i(\theta) \le 0 \text{ for } i = 1, \dots, m, \end{aligned} \tag{2}$$

where the real-valued functions g_0 and the g_is may be non-convex. Furthermore, each of the m constraint functions g_i may be data-dependent, non-convex and even non-differentiable.

1.4. The Lagrangian May Have No Pure Equilibrium For Non-Convex Problems

A popular approach to constrained optimization problems of the form in Equation 2 is the method of Lagrange multipliers. Define the Lagrangian:

$$\mathcal{L}(\theta, \lambda) \triangleq g_0(\theta) + \sum_{i=1}^{m} \lambda_i g_i(\theta), \tag{3}$$

where λ is an m-dimensional non-negative vector of Lagrange multipliers. The method of Lagrange multipliers can be viewed as a two-player zero-sum game where one player minimizes Equation 3 with respect to the model parameters $\theta \in \Theta$, and the other player maximizes it with respect to the

Lagrange multipliers $\lambda \in \Lambda$. If the objective and constraints are all convex in θ, and the action spaces Λ and Θ are compact and convex, then this is a convex game, and it has a pure Nash equilibrium (von Neumann, 1928), i.e. there exists a θ for the first player and a λ for the second player such that neither player has the incentive to change their choice given the other player's choice. Furthermore, a pure Nash equilibrium (equivalently, a saddle point of the Lagrangian) gives us an optimal and feasible solution to the original constrained optimization problem specified in Equation 2.

On a constrained non-convex problem the Lagrangian might not even *have* a pure Nash equilibrium (see Figure 1 for an example). Hence, instead of converging, an iterative first-order algorithm may oscillate between different solutions, or it may converge to a locally optimal point—but not a Nash equilibrium—for which it is difficult to establish optimality and feasibility properties. However, if we allow each player to choose a *distribution* over their respective spaces Θ and Λ, and take the value of the Lagrangian to be the expectation over these distributions, then, under general conditions, the resulting *mixed* Nash equilibrium will exist.

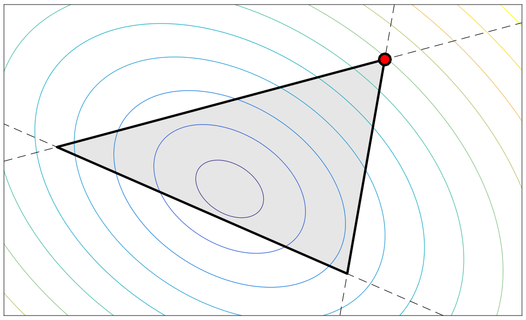

Figure 1: **Example of when no pure Nash equilibrium exists for the Lagrangian**: The plotted rectangular region is the domain Θ, the contours are those of the *strictly concave* minimization objective function g_0, and the shaded triangle is the feasible region determined by the three linear inequality constraints g_1, g_2, g_3. The red dot is the optimal feasible point. The Lagrangian $\mathcal{L}(\theta, \lambda)$ is strictly concave in θ for any choice of λ, so the optimal choice(s) for the θ-player will always lie on the four corners of the plotted rectangle. However, these points are infeasible, and therefore suboptimal for the λ-player, assuming that $\lambda \in \Lambda = \mathbb{R}_+^3$.

In this paper, we provide algorithms that approximately find such mixed equilibria, and we show that these correspond to nearly-feasible and nearly-optimal *stochastic* solutions to the original constrained optimization problem given in Equation 2. Such a stochastic solution is a random model: every time we classify an example x, we will independently sample a θ from the equilibrium distribution over Θ. Our guarantees will be expressed in terms of expectations with respect to this random θ.

1.5. The Lagrangian Is Impractical For Non-differentiable Constraints

Next, consider the issue of non-differentiable constraints (such as rate constraints). A major shortcoming of the Lagrangian is that one cannot use gradient-based methods to optimize non-differentiable constraints. One approach is to use the Lagrangian but replace non-differentiable constraints with differentiable surrogates (e.g. Davenport et al., 2010; Gasso et al., 2011; Eban et al., 2017). However changing the constraint functions may lead to solutions which either over-constrain or fail to satisfy the original constraints, as shown in Figure 2.

To address this, we introduce what we call the *proxy-Lagrangian* formulation, where the key idea is to relax the non-differentiable constraints *only when necessary*. Solving the proxy-Lagrangian poses technical challenges but leads to a number of interesting insights, and we provide algorithms

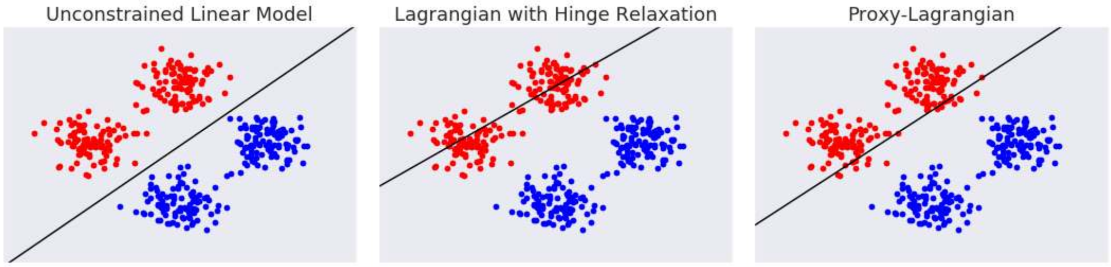

Figure 2: Mixture of Gaussians simulation: we generate 400 datapoints from a mixture of four Gaussians in two dimensions centered at $(0,1), (-1,0), (0,-1), (1,0)$ in equal proportion each with covariance matrix $0.05\mathbf{I}$ where $\mathbf{I}$ denotes the identity matrix. The two top-left Gaussians are class red while the two bottom-right Gaussians are class blue. **Left**: The black line is the decision boundary for a linear model trained without any constraints. **Middle**: We trained this linear model subject to the rate constraint that the classifier must predict at least 55% of examples as blue. Here, we use the classical Lagrangian formulation and a hinge relaxation of the indicators in the constraints. Since the hinge relaxation is overly conservative, this classifier ends up overconstraining and actually predicts blue for 80.5% of the 400 examples in order to satisfy the *relaxed* constraint, at the cost of unnecessary loss of accuracy. **Right**: We trained this linear model subject to the same rate constraint that it must predict 55% of examples as blue, but this time we trained the model using the proposed proxy-Lagrangian formulation, and took the last iterate as the model. This model exactly enforces the requested 55% prediction rate for blue examples.

which attain solutions with optimality and feasibility guarantees on the original non-differentiable constraints.

Overall, we give an end-to-end recipe to provably (given access to an optimization oracle) and efficiently solve non-convex optimization problems with non-differentiable constraints, for which the solution will be a mixture of at most $m+1$ deterministic solutions. In practice, we use SGD in place of the oracle. To our knowledge, this is the first time such a procedure has been found to provably solve such non-convex problems with such irregular constraints and return a sparse solution.

In addition, for those practical situations where a stochastic model is unappealing, we also experimentally consider algorithms that do produce deterministic models, though they do not come with guarantees.

1.6. Main Contributions And Organization

The main contributions of this paper are:

- We show that training with rate constraints can be used to address many real-world goals and capture realistic prior knowledge in the training.
- We give a new *proxy-Lagrangian* formulation for optimizing non-convex objectives with non-differentiable constraints.

- We provide an algorithm that outputs a $m+1$ *sparse* stochastic classifier with theoretical guarantees, where m is the number of constraints.
- We show that our *proxy-Lagrangian* formulation can also be used to produce a deterministic classifier that may be more practical for some applications, but without guarantees.
- We provide an open-source Tensor Flow library that implements the presented algorithms.
- We experimentally demonstrate that the proposed optimization can be used to train classifiers with rate constraints, on both benchmark datasets and for real-world case studies.

Although our motivation and experimental focus is on the problem of training classifiers with rate constraints, our proposed proxy-Lagrangian formulation and theoretical results have broader application to other constrained optimization problems.

We next review related work. Then in Section 3 we detail many different goals that can be expressed with rate constraints. We then turn to the question of how to actually optimize with constraints, proposing new algorithms and theoretical results in Section 4. Section 5 presents a diverse set of experiments on benchmark and real datasets to illustrate the applicability of rate constraints and the proposed optimization. We close with a discussion of conclusions in Section 6 and open questions in Section 7.

2. Related Work

We begin by reviewing our own prior work which this paper builds upon, then other work that considers specific rate constraints, and then related work in constrained optimization.

2.1. Related Work On Specifying And Optimizing With Rate Constraints

Goh et al. (2016) showed that many different types of policy goals and side information can be expressed as constraints on the classifier's decisions on targeted datasets, and that one can then train the classifier to respect these constraints as part of the empirical risk minimization. Goh et al. (2016) referred to this class of constraints as *dataset constraints*, but we use the more precise term *rate constraints* to reflect that these constraints are functions of the classifier's positive and negative decision rates. In this paper, we will present many more goals and types of prior information that can be expressed as rate constraints that are useful in practice but have not previously appeared in the literature, such as *no lost benefit*, *not worse off*, and *loss-only churn*. Further, we provide more insight and analysis on how to use rate constraints in practice.

To optimize models with rate constraints, Goh et al. (2016) proposed a constrained optimization algorithm that was limited to linear classifiers, and used a new cutting-plane algorithm to iteratively upper-bound the ramp loss with a convex loss, then solved the resulting inner-loop minimizations using an SVM solver. While amenable to theoretical analysis, this strategy is a bit slow and difficult to scale to more than a handful of constraints. In contrast, in this paper we show we can effectively and efficiently train *nonlinear* classifiers with rate constraints using the more popular and scalable approach of stochastic gradients.

An important classic special case of rate constraints is Neyman-Pearson classification, which constrains the false positive rate (Scott and Nowak, 2005). Davenport et al. (2010) optimized Neyman-Pearson support vector machines with hinge loss relaxations using coordinate descent. Gasso et al. (2011) relaxed the indicators to the ramp loss (both in the objective and constraints). Eban et al. (2017) optimized the model parameters and Lagrangian multiplier using stochastic gradients with a hinge approximation for the indicators in the empirical loss and constraints, and took the last training iterate as their solution. We compare to that optimization strategy in our experiments (listed as *Hinge Last* in the result tables).

Mann and McCallum (2007) and follow-on work (Bellare et al., 2009; Mann and McCallum, 2010) optimized probabilistic models with constraints in order to incorporate side information about the prior priors on class labels, which in the context of a binary classifier is a special case of rate constraints that we call a *coverage constraint*. They note their strategy could also be applied to any constraints that can be written as an expectation over a score on the random (X, Y) samples. They incorporated this side information as an additive regularizer and penalized the relative entropy between the given priors and estimated multi-class logistic regression models. They noted their approximation for the indicator could lead to degenerate solutions, which they indirectly addressed by additional regularization.

Agarwal et al. (2018) recently addressed training classifiers with fairness constraints that can be expressed as rate constraints. Like this work, their proposed algorithm is based on the two-player game perspective. Unlike this paper, they assume a *zero-sum* game, which works because they also assume oracle solvers for the two players, side-stepping the practical issues of dealing with the non-differentible non-convex indicators in the constraints, which is the focus of our algorithmic and theoretical contributions. Similar to this work, they output a stochastic classifier, but do not provide the sparse $m + 1$ solution that we present in this work. They also consider a deterministic solution, which they produce by searching over a grid of values for λ for the best λ. They noted in their experimental section that the resulting deterministic solution was generally as good as their stochastic solutions on test data for those experiments they tried it on. As they note, a grid-search over λ is less ideal as the number of constraints grows.

Some other work in training with fairness constraints has used weaker constraints or relaxed them immediately to weaker constraints such as correlation, e.g. Zafar et al. (2015, 2017). Another set of work in fair classification only corrects a model post-training by optimizing additive group-specific bias parameters, e.g. Hardt et al. (2016) and Woodworth et al. (2017). Donini et al. (2018) studies optimization of fairness constraints for kernel methods by formulating the fairness constraints as orthogonality constraints. The goal of *equal accuracy* has also been explored recently in Buolamwini and Gebru (2018) in the context of matching the accuracies of male/female classifiers across race.

2.2. Other Types Of Constraints On Machine Learned Models

We focus on rate constraints in this paper, which have tend to have the following properties. First, because rate constraints depend on $f(x)$, they generally depend on all the model parameters θ. Second, rate constraints are usually relatively expensive to compute. Third, we do not generally expect to have a very large number of rate constraints.

These qualities are different from the popular constrained machine learning problem of *shape constraints*, which requires that the model is restricted to functions with a certain shape such as monotonic functions (e.g. Barlow et al. (1972); Groeneboom and Jongbloed (2014); Gupta et al. (2016); Canini et al. (2016); Luss and Rosset (2017); You et al. (2017); Bonakdarpour et al. (2018)), or other shapes (e.g. Chetverikov et al. (2018); Pya and Wood (2015); Chen and Samworth (2016); Gupta et al. (2018); Cotter et al. (2019b)).

In contrast to rate constraints, shape constraints generally require adding many sparse, cheap-to-evaluate constraints. For example, for isotonic regression on N training examples, there are $O(N)$ constraints, and each is a function of only two model parameters (Barlow et al., 1972). Similarly, in some of the experiments of Canini et al. (2016), the models are trained with around 100,000 constraints, but each constraint only touches two model parameters. Problems like that with many cheap sparse constraints can be well-handled by stochastic sampling of the constraints, as in Cotter et al. (2016), but that strategy is less well-suited to rate constraints because there tend to be fewer constraints, and each constraint is expensive to evaluate.

Another type of constrained machine learning aims to constrain the model parameters to obey known physical limits on the learned system (e.g. Long et al. (2018); Stewart and Ermon (2017)). These constraints generally do not take the form of rate constraints, but such constrained machine learning models may also benefit from the presented algorithms and theory.

Some fairness constraints are more complicated than can be handled as rate constraints. For example, Heidari et al. (2018) give a new individual fairness notion which ensures the expected utility an individual receives as a result of the model.

2.3. Related Work In Constrained Optimization As A Two Player Game

Our strategy for treating non-differentiable problems as a non-zero sum two-player game using a proxy Lagrangian formulation was first presented in our conference paper, Cotter et al. (2019c). This journal paper extends that work with more discussion of how a broad set of goals can be expressed as rate constraints, much more comprehensive experiments, some additional theoretical perspectives, and more advice for practitioners.

Our constrained optimization algorithms and analyses build on the long history of treating constrained optimization as a two-player game: Arora et al. (2012) surveys some such work, and there are several more recent examples (e.g. Agarwal et al. (2018); Kearns et al. (2018); Narasimhan (2018)). We extend that prior work in three key ways. First, to handle non-differentiable constraints, we propose a new proxy-Lagrangian *non-zero-sum* formulation, whereas prior work formulates the optimization as a zero-sum game. Second, we introduce a shrinking procedure that significantly simplifies a "T-stochastic" solution (i.e. a stochastic classifier supported on all T iterates) to a sparse "m-stochastic" solution (a stochastic classifier supported on only $m+1$ iterates, where m is the number of constraints). Third, we consider a broader set of problems than prior work.

Our contributions also apply to robust optimization problems of the form:

$$\min_{\theta \in \Theta} \max_{i \in [m]} g_i(\theta),$$

where each $g_i : \Theta \to \mathbb{R}$. The most related work addressing non-convex robust optimization is Chen et al. (2017). Like both Agarwal et al. (2018) and this paper, Chen et al. (2017) (i) model the problem as a two-player game where one player chooses a mixture of objective functions, and the other player minimizes the loss of the mixture, and (ii) they find a *distribution* over solutions rather than a pure equilibrium. These similarities are unsurprising in light of the fact that robust optimization can be reformulated as constrained optimization via the introduction of a slack variable:

$$\begin{aligned} &\min_{\theta \in \Theta, \xi \in \mathbb{R}} \xi \\ &\text{s.t. } \xi \geq g_i(\theta) \quad \text{for all } i \in 1, \ldots, m. \end{aligned} \tag{4}$$

Correspondingly, one can transform a robust problem to a constrained one at the cost of an extra bisection search (e.g. Christiano et al., 2011; Rakhlin and Sridharan, 2013). As this relationship suggests, our main contributions can be adapted to the robust optimization setting. In particular: (i) our proposed shrinking procedure can be applied to Equation 4 to yield a distribution over only $m+1$ solutions, and (ii) one could perform robust optimization over non-differentiable (even discontinuous) losses using "proxy objectives," just as we use proxy constraints.

2.4. Other Strategies For Constrained Optimization

There are other strategies for constrained optimization, each of which we argue is not well-suited to the problem of training classifiers with rate constraints.

The computational complexity of rate constraints makes them generally unattractive to try to optimize with approaches that require projections, such as projected SGD, or optimization of constrained subproblems, such as Frank-Wolfe (Hazan and Kale, 2012; Jaggi, 2013; Garber and Hazan, 2013)).

Another strategy for constrained optimization is to penalize violations of the constraints (e.g. Arora et al., 2012; Rakhlin and Sridharan, 2013; Mahdavi et al., 2012; Cotter et al., 2016; Yang et al., 2017), for example by adding $\gamma \max_{i\in[m]} \max\left\{0, g_i\left(\theta\right)\right\}$ to the objective, where $\gamma \in \mathbb{R}_+$ is a hyperparameter, and optimizing the resulting problem using a first-order method. This strategy is not ideal for rate constraints for two reasons. First, rate constraints are non-(semi)differentiable. Second, each rate constraint is data-dependent, so evaluating g_i, or even determining whether it is positive (as is necessary for such methods, due to the max with 0), requires enumerating over the entire constraint dataset, making this incompatible with the use with a computationally-cheap stochastic gradient optimizer.

3. What Are Rate Constraints Good For?

In this section, we first present the mathematical formulation of rate constraints and the resulting constrained empirical risk minimization training. Table 1 provides a handy reference for key notation. Then, we provide a list of metrics that can be expressed as rate constraints in Table 2, and detail in the following subsections how these rate constraints can be used to impose a broad set of policy goals and take advantage of side information.

Given a classifier $h : \mathcal{X} \times \Theta \rightarrow \mathbb{R}$ (where $\mathcal{X}$ is the feature space and Θ is the parameter space) a dataset D, and using I to denote the usual indicator, define the classifier's positive classification rate on D as $p^+\left(D;\theta\right)$, and the classifier's negative classification rate on D as $p^-\left(D;\theta\right)$, where

$$p^+(D;\theta) \triangleq \frac{1}{|D|}\sum_{x\in D} I_{h(x;\theta)\geq 0} \quad \text{and} \quad p^-(D;\theta) \triangleq \frac{1}{|D|}\sum_{x\in D} I_{h(x;\theta)<0}. \tag{5}$$

We call a constraint a *rate constraint* if it can be expressed in terms of a non-negative linear combination of positive classification rates $p^+\left(D_k;\theta\right)$ and negative classification rates $p^-\left(D_k;\theta\right)$ over different datasets $\{D_k\}$. That is, a *rate constraint* is a constraint expressible as:

$$\sum_{k=1}^{K} \alpha_k p^+\left(D_k;\theta\right) + \beta_k p^-\left(D_k;\theta\right) \leq \kappa, \tag{6}$$

for some $\alpha \in \mathbb{R}^+$, $\beta \in \mathbb{R}^+$, and $\kappa \in \mathbb{R}^+$.

Table 2 shows how different choices of scalars $\alpha_k, \beta_k, \kappa \in \mathbb{R}$ and datasets $\{D_k\}$ correspond to different standard performance metrics like accuracy and recall. One can add m rate constraints to

Table 1: Basic Notation

D	Set of examples
$D[*]$	Subset of D that satisfies expression *, e.g. $D[x \in \text{male}]$ is the subset of D of male examples, $D[y = 1]$ is the subset of D whose label is 1, etc.
$\tilde{h} : \mathcal{X} \times \Theta \to \mathbb{R}$	A given classifier to which a candidate classifier h can be compared
$p^+(D; \theta) \in [0, 1]$	Proportion of D classified positive
$p^-(D; \theta) \in [0, 1]$	Proportion of D classified negative
$c^+(D; \theta) \in \mathbb{N}$	Count of D classified positive: $c^+(D; \theta) = \lvert D\rvert \, p^+(D; \theta)$
$c^-(D; \theta) \in \mathbb{N}$	Count of D classified negative: $c^-(D; \theta) = \lvert D\rvert \, p^-(D; \theta)$

the standard structural risk minimization to train a classifier with parameters $\theta \in \Theta$ on train dataset D_0, producing the constrained empirical risk minimization:

$$\begin{aligned} &\min_{\theta \in \Theta} \frac{1}{|D_0|} \sum_{(x,y) \in D_0} \ell(h(x; \theta), y) + R(\theta) \\ &\text{s.t.} \sum_{k=1}^{K_i} \alpha_{ik} p^+ (D_{ik}; \theta) + \beta_{ik} p^- (D_{ik}; \theta) \leq \kappa_i \;\text{ for } i = 1, \ldots, m, \end{aligned} \tag{7}$$

where $\alpha_{ik}, \beta_{ik} \in \mathbb{R}$, D_{ik} is the kth dataset for the ith constraint, K_i is the number of datasets used to specify the ith constraint, and $\kappa_i \in \mathbb{R}$.

For some applications it is notationally more convenient to drop the normalization on the rate constraints and express the constraint in terms of counts, let $c^+ (D; \theta)$ and $c^- (D; \theta)$ denote the count of the positive and negative classifications:

$$c^+(D; \theta) \triangleq \sum_{x \in D} I_{h(x;\theta) \geq 0} \quad \text{and} \quad c^-(D; \theta) \triangleq \sum_{x \in D} I_{h(x;\theta) < 0}. \tag{8}$$

Throughout this work, we focus on inequality constraints, for lower-bounding or upper-bounding some rate. Equality constraints can be imposed by using both a lower-bound and upper-bound inequality constraint, though we suggest doing so with some margin between the lower and upper bound to make the optimization problem easier.

In the rest of this section we show how different rate constraints can be used to impose various policy goals or capture side information. A key insight is that one can add constraints just on specific groups or subsets of the dataset by the choice of the datasets used for a constraint, which makes this approach particularly useful for fairness goals or other slice-specific metrics that are measured in terms of statistics on different datasets (see Table 3 and further details below).

3.1. Coverage Constraints

Coverage is the proportion of classifications that are positive: $p^+(D; \theta)$ (a variant is *negative coverage* $p^-(D; \theta)$). For example, if a company wants to train a classifier to identify the best 10% of all customers to receive a printed catalog, then one could train the classifier with a 10% coverage constraint.

Table 2: Examples of Metrics Expressed With Rates And Notation From Table 1

Recall	$p^+(D[y=1];\theta)$
Precision	$c^+(D[y=1];\theta)/c^+(D;\theta)$
Accuracy	$(c^+(D[y=1];\theta)+c^-(D[y=-1];\theta)/\lvert D\rvert$
AUCROC	$\lim_{L,J\to\infty}\frac{1}{L}\sum_{\ell=1}^{L}\quad \max_{j\in[J]:p^+_{\alpha_j}(D[y=-1];\theta)\le\frac{\ell}{L}}\quad p^+_{\alpha_j}(D[y=1];\theta)$
Wins Compared to $\tilde{h}$	$c^+(D[\tilde{h}=-1,y=1];\theta)+c^-(D[\tilde{h}=1,y=-1];\theta)$
Losses Compared to $\tilde{h}$	$c^+(D[\tilde{h}=-1,y=-1];\theta)+c^-(D[\tilde{h}=1,y=1];\theta)$
Win Loss Ratio (WLR)	Wins Compared to $\tilde{h}$ / Losses Compared to $\tilde{h}$
Churn	$(c^+(D[\tilde{h}=-1];\theta)+c^-(D[\tilde{h}=1];\theta))/\lvert D\rvert$
Loss-only Churn	$(c^+(D[\tilde{h}=-1,y=-1];\theta)+c^-(D[\tilde{h}=1,y=1];\theta)/\lvert D[\tilde{h}=y]\rvert$

Coverage constraints can also be used to capture prior knowledge in the training. For example, if training a model to classify Americans as male or female, one can regularize the classifier by incorporating the prior knowledge that 51% of examples should be predicted to be female, by using a 51% coverage constraint.

Using slice-specific coverage constraints can capture more side information. For example, for the American male/female classifier, in addition to the overall coverage constraint of 51%, one could also add constraints capturing prior information about state sex distributions, such as constraining 51.5% of examples from New York to be classified as women, but constraining only 47.6% of examples from Alaska to be classified as women.

A key advantage of coverage constraints is that they do not require labeled examples. This enables one to train on labeled training examples from a convenient distribution (such as actively-sampled examples), but then add a coverage constraint to ensure the classifier is optimized to positively classify the desired proportion of positive classifications on a larger unlabeled dataset drawn *i.i.d.* from the true underlying distribution. This usage of a coverage constraint forms a semi-supervised regularization of the classifier.

Another good use case for coverage constraints is to help make a controlled comparison of two model structures. For example, suppose one has a model type A (say, a kernel SVM), and wonders if an alternative B (say, a DNN) is better, where A makes positive predictions on 40% of test examples, while B appears to be more accurate, but only predicts the positive class for 35% of test examples. If precision errors are worse than recall errors, we cannot be sure that B is better than A. We can try to quantify the misclassification costs of a false negative vs. a false positive, but that may be difficult to agree upon. It would be simpler to compare B to A at the same coverage as A, or at some other relevant coverage. Coverage-matching B to A can be done by tuning the decision threshold of B post-training, but including the coverage constraint in the training can help B learn to be a better classifier when tested at the desired coverage.

3.2. Constraints On Accuracy, Recall, Precision, AUC

As shown in Table 2, classifier accuracy can be expressed in terms of rates, and thus accuracy on auxiliary datasets or slices of the training data can be constrained with rate constraints.

Recall, defined as TP / (TP + FN), can be written as $p^+(D[y=1];\theta)$, and thus one can put a lower-bound constraint on recall $p^+(D[y=1];\theta)>\kappa$ for the user's choice of $\kappa\in[0,1]$. For example, one may wish to train a classifier that awards free meals to poor students, but constrain it to obtain at least 95% recall.

Precision can be expressed in rates as $c^{+}(D[y=1];\theta)/c^{+}(D;\theta)$, and thus to get precision of at least κ, one can add a rate constraint:

$$c^{+}(D[y=1];\theta)-\kappa c^{+}(D;\theta)\geq 0. \tag{9}$$

If (9) holds, then mathematically the precision is lower-bounded by κ on the dataset D. However, since the expectation of a ratio does not equal the ratio of the expected numerator and denominator, analyzing how well the empirical constraint holding generalizes to new *i.i.d.* samples is not straightforward, and violating the constraint (9) by some $\epsilon > 0$ does not translate directly into a precision error of ϵ.

The ROC AUC (Area under the ROC curve) can be approximated using a rate constraint, as in Eban et al. (2017). The ROC curve is obtained by plotting the true positive rate (TPR) vs. the false positive rate (FPR). First, slice up the FPR-axis into L slices (to approximate the required Riemann integral). Then for the ℓth slice, consider J different decision thresholds and choose the threshold that maximizes TPR and satisfies the ℓth slice FPR bound ℓ/L, and then the averaged maximum precision across the L FPR slices is bounded:

$$\frac{1}{L}\sum_{\ell=1}^{L}\max_{j\in[J]:p^{+}_{\alpha_j}(D[y=-1];\theta)\leq\frac{\ell}{L}} p^{+}_{\alpha_j}(D[y=1];\theta)\geq\kappa. \tag{10}$$

where $p^{+}_{\alpha}(D;\theta)\triangleq\frac{1}{|D|}\sum_{x\in D}I_{h(x;\theta)\geq\alpha}$, $c^{+}_{\alpha}(D;\theta)\triangleq\sum_{x\in D}I_{h(x;\theta)\geq\alpha}$, and $\alpha_j := \frac{2j-1}{2J}$ for $j\in[J]$. In particular, $p^{+}_{0}\equiv p^{+}$. Taking $L\to\infty, J\to\infty$ will have the expression on the LHS of (10) converge to the exact ROC AUC.

3.3. Churn And Win Loss Ratio Constraints

In practice, a new classifier is often being trained to replace an existing classifier $\tilde{h}$, in which case the new classifier may be evaluated using metrics that compare the new classifier to the old classifier $\tilde{h}$.

One common metric to compare two classifiers is the win-loss ratio (WLR), which is the number of times the new classifier is right and the old classifier is wrong, divided by the number of times the new classifier is wrong and the old classifier is right.

A WLR constraint can be expressed in terms of rates as in Table 2, where we use $D[\tilde{h}=-1]$ to denote the subset of D that is labeled negatively by the classifier $\tilde{h}$, and $D[\tilde{h}=-1,y=1]$ to denote the subset of D of whose training label y is 1, so that $c^{+}(D[\tilde{h}=-1,y=1];\theta)$ is the number of wins of the new classifier over h, and so on. Re-arranging terms, one can constrain for WLR using a rate constraint:

$$\begin{aligned}&c^{+}(D[\tilde{h}=-1,y=1];\theta)+c^{-}(D[\tilde{h}=1,y=-1];\theta)\\&-\kappa(c^{+}(D[\tilde{h}=-1,y=-1];\theta)+c^{-}(D[\tilde{h}=1,y=1];\theta))\geq 0,\end{aligned} \tag{11}$$

where $\kappa\in\mathbb{R}^{+}$ is the lower-bound on the WLR. However, enforcing this constraint on a training dataset D does not necessarily guarantee that the desired WLR threshold will be achieved on fresh *i.i.d.* samples, not only due to the potential for overfitting, but also because the expectation of a ratio does not equal the ratio of expectations.

WLR constraints on different slices of the data can ensure that a new classifier's gains are not coming at the expense of an important subset of examples. (See also our discussion of *no worse off* and *no lost benefits* for related fairness constraints).

In practice, labeling data can be expensive, so it is common to test a new classifier by drawing a fresh test set including only examples on which the new classifier and previous classifier $\tilde{h}$ disagree. We refer to this as a *fresh test*. Fresh tests reduce the chance of overfitting to a fixed test set that is used over many model iterations. Fresh tests only incur labeling costs for those examples whose decisions have changed. Thus with a fresh test, higher WLR means fewer fresh test examples have to be rated to statistically significantly confirm that the new classifier is better than the old classifier $\tilde{h}$ (Cormier et al., 2016).

Note that two new classifiers can have the same accuracy but different WLRs compared to the previous classifier h. The proportion of a dataset D on which the classifications change when one changes classifiers is called *churn* (Cormier et al., 2016; Goh et al., 2016). When using a fresh test, the labeling costs scale linearly with the churn (and the size of the test set D). High churn also causes more instability for follow-on systems, and can confuse users. Goh et al. (2016) proposed explicitly constraining the churn, which can be directly expressed as the rate constraint:

$$c^{+}(D[\tilde{h} = -1]; \theta) + c^{-}(D[\tilde{h} = 1]; \theta) \leq \kappa |D|, \tag{12}$$

where $\kappa \in [0, 1]$ is the proportion of D on which the classification decision is allowed to change. Constraining churn on different slices of the data, with tighter and looser constraints, can be useful. For example, if the classifier is to be used worldwide, but labeling is more expensive in Norway than in Vietnam, or if there is known to be less headroom to improve on examples from Norway, then it could be beneficial to constrain the churn more tightly on examples from Norway, but more loosely on examples from Vietnam.

Of course, constraining churn too tightly limits the potential accuracy gains. Thus we also propose considering loss-only churn constraints, which only penalizes new losses:

$$c^{+}(D[\tilde{h} = -1, y = -1]; \theta) + c^{-}(D[\tilde{h} = 1, y = 1]; \theta) \leq \kappa |D|, \tag{13}$$

where $\kappa \in [0, 1]$ is the proportion of D whose classification decision is allowed to flip.

One disadvantage of constraining loss-only churn is it requires labeled examples, whereas churn constraints can be more conveniently used on a dataset of unlabeled examples.

3.4. Fairness Goals And Other Group-Specific Goals

An important use case for rate constraints is enforcing metrics for different groups or categories of examples. For example, ensuring that a classifier for identifying family-friendly videos works roughly equally well at filtering different types of objectionable adult content. Rate constraints can be used to enforce a broad set of such group-specific goals, as detailed in Table 3, where k indexes the K different groups of interest.

A special case of group-specific goals are those that are designed to improve some fairness metric. In these cases the groups are usually defined as different groups of people, e.g. different genders or age brackets. Table 3 shows that many of the fairness goals already studied in the machine learning literature can be expressed with rate constraints. With that said, fairness is a complex moral and policy problem, and depending on the context and application, different formulations may be appropriate, with some such formulations not being group-based at all.

Many fairness goals are designed for applications where positive classification endows a benefit, such as being awarded a loan, a job, or a free meal. For example, the goal of *statistical parity* reflects that a bank might be legally required to give loans at equal rates to different groups to alleviate disparity (Bocian et al., 2008), that is, the classifier is required to provide equal positive rates of classification across groups (see e.g. Zafar et al. (2015); Fish et al. (2016); Hardt et al. (2016); Goh et al. (2016)). Statistical parity is also known as *demographic parity* (Hardt et al., 2016), and *equal coverage* (Goh et al., 2016). Notice that a statistical parity constraint ignores the labels of the training data. We introduce the related goal of *minimum coverage*, which enforces some minimal benefit rate for each group.

Next we detail some fairness goals we find useful in practice, but have not seen previously formalized in the literature. The goal of *accurate coverage* requires the classifier to give free meals to each group to match that group's positive training label rate. This goal ignores whether the individual predictions are accurate, but tries to ensure that each group overall receives a rate of benefits that it is labeled as deserving.

No lost benefits: which requires a model to classify examples positively from each group at least as often as the classifier $\tilde{h}$ that it is replacing. *No lost benefits* is a type of churn goal (see Sec. 3.3) that is measured for the whole group (rather than for individual decisions).

The other fairness goals in Table 3 depend on the training labels. Our *not worse off* fairness goal requires that accuracy with the new classifier for each group is not worse than it was under the classifier $\tilde{h}$ that it would replace. For example, suppose someone invents a new driving test, and shows that it is more accurate than the current written driving test at diagnosing whether illiterate people are safe drivers, then *not worse off* requires that the new driving test not reduce accuracy compared to the old test for other groups, e.g. senior citizens and teenagers. *Not worse off* is a label-dependent group-specific churn goal.

Minimum accuracy, which requires that every group experience some pre-set level of accuracy. *Minimum accuracy* ensures that no group is left behind, but respects that for some problems some groups may be much easier to classify than other groups. For such problems, constraining accuracy to be similar across groups can lead to degenerate solutions, as the only way to make all groups have equal metrics may be to produce a degenerate classifier.

Equal opportunity and *equal odds* (Hardt et al., 2016) also rely on the training labels. For example, *equal opportunity* requires that *if* a classifier awards free meals (positive classification) to half of the east-side children who are labeled as deserving free meals, then it should also award free meals to half of the west-side children who are labeled as deserving free meals. Notice that *equal opportunity* imposes no conditions whatsoever on the negatively-labeled examples (in this case, those students who are not labeled as deserving of free meals). In contrast, the fairness goal of *equal odds* requires both the true positive rate and the false positive rate to be the same for all groups. Variations are *equal accuracy*, *equal recall*, *equal precision*, and so on, all of which aim to make the classifier equally good at some metric for different groups.

Fairness goals that depend on the training labels are most compelling when the training examples and labels are believed to have been fairly sampled and labeled. These goals are less compelling when the training dataset is not entirely trusted, or thought to be misaligned with the policy goals, a situation referred to as *negative legacy* (Kamishima et al., 2012). There are many reasons why training data and labels may not be fully trustworthy. Selection biases on the training examples and raters labeling them can affect the training distribution in ways that negatively impact certain groups. Further, training labels can have unbiased noise due to all sorts of cognitive biases. One key

Table 3: Group-Specific and Fairness Goals Expressed As Rate Constraints for Groups $k = 1, \ldots, K$

Statistical Parity	$p^+(D_k;\theta) = p^+(D;\theta)\ \forall k$
Minimum Coverage	$p^+(D_k;\theta) \geq \kappa\ \forall k$ and user-specified $\kappa \in [0,1]$
No Lost Benefits	$p^+(D_k;\theta) \geq \lvert(D_k[\tilde{h}=1])\rvert / \lvert D_k\rvert\ \forall k$
Accurate Coverage	$p^+(D_k;\theta) = \lvert D_k[y=1]\rvert / \lvert D_k\rvert\ \forall k$
Equal Opportunity	$p^+(D_k[y=1];\theta) = p^+(D[y=1];\theta)\ \forall k$
Equal Odds	$p^+(D_k[y=1];\theta) = p^+(D[y=1];\theta)\ \forall k$
	and $p^+(D_k[y=-1];\theta) = p^+(D[y=-1];\theta)\ \forall k$
Equal Accuracy	$(c^+(D_k[y=1]) + c^-(D_k[y=-1]))/\lvert D_k\rvert$
	$= (c^+(D[y=1]) + c^-(D[y=-1]))/\lvert D\rvert\ \forall k$
Minimum Accuracy	$(c^+(D_k[y=1]) + c^-(D_k[y=-1]))/\lvert D_k\rvert \geq \kappa\ \forall k$
Not Worse Off	$(c^+(D_k[y=1]) + c^-(D_k[y=-1]))/\lvert D_k\rvert \geq \lvert(D_k[y=\tilde{h}])\rvert / \lvert D_k\rvert\ \forall k$

problem is raters are generally more accurate at labeling examples they are more familiar to them, for example, consider a situation where adult raters are asked to label whether children will find a video interesting.

3.5. Egregious Examples And Steering Examples

Another use of rate constraints is to constrain the performance on auxiliary labeled datasets to control the classifier. For example, Goh et al. (2016) proposed constraining the classifier for high accuracy on a small set of particularly *egregious examples* that should definitely not be mislabeled. Egregious examples act as an *integrated unit test*: as the classifier trains, it actively tests itself to make sure it satisfies the constraint on the egregious examples and is able to correct the training accordingly.

Another practical example of using an auxiliary labeled dataset is what we term *steering examples*, which we define as a set of labeled examples that are more accurately labeled than the training set. For example, one may have access to a large but noisy training set of clicks on news articles. However, an article might be clicked either because it was relevant news, or because it had a catchy headline. We can try to steer the classifier to focus on the relevant news articles by providing a smaller but expertly-labeled curated set of examples that mark catchy headlines as negative, and then constrain the classifier to achieve some reasonable minimal accuracy on these steering examples. Such a rate constraint will steer the classifier to be consistent with the steering examples, and helping it disregard any badly labeled training examples. A second example is a classifier that attempts to determine whether an online store should advertise to a given customer. Suppose that there is a large dataset of training examples with the positive label, “customer clicked advertisement and *visited* website”, but a relatively small set of examples where the positive label is, “customer clicked advertisement and made a *purchase*.” It may be better to train on the large set of “visited” examples due to its much larger size and coverage, but also to constrain at least some specified accuracy on the smaller “purchase” examples in order to steer the classifier towards prioritizing clicks that lead to purchases.

3.6. Decision Rule Priors

Machine learning practitioners often have prior knowledge about a classification problem that they can communicate as a decision rule on a tiny set of features. For example, "Don't recommend a book to a user if it is in a language they haven't purchased before." We propose a simple way of incorporating such decision rule priors into the structural risk minimization problem by creating an auxiliary dataset consisting of many unlabeled samples, labeling it with the desired decision rule, and adding an accuracy rate constraint on that auxiliary data set.

Such decision rule priors can act as regularizers against noisy and poorly-sampled training examples, and can produce a classifier that is more interpretable because it is known to (probably) obey the given decision rules (like all rate constraints, this depends on whether one constrains with slack or not, and also on exactly how well the satisfied constraint generalizes, which depends on whether the examples observed at evaluation time will truly be drawn *i.i.d.* from the same distribution as the training data, as well as the function class, and how hard the constraint is to satisfy).

This proposal is similar to Bayesian Rule Lists (BRL) (Letham et al., 2015) in that a decision rule (or set of decision rules) is given a priori to training the model. However, BRL training takes as input a large set of decision rules and outputs a posterior over the rules, rather than incorporating a decision rule into a structural risk minimization problem.

3.7. How To Best Specify Rate Constraints

For any rate constraint, we recommend allowing some slack in order to find a feasible solution. For example, statistical parity could be written as a constraint with an *additive slack* of κ like this:

$$p^{+}(D;\theta) - p^{+}(D_k;\theta) \leq \kappa,$$

or instead with *multiplicative slack* of κ like this:

$$p^{+}(D;\theta) - \kappa p^{+}(D_k;\theta) \geq 0,$$

where k is the index of the kth subset of interest.

Our experience is that additive slack tends to be more likely to produce reasonable solutions than multiplicative slack for many constraints. The danger to watch out for is whether the constraint is specified in a way that encourages the training to satisfy the constraint in a suboptimal way. For example, if one constrains the false positive rate of each group to be no worse than 125% of the overall false positive rate (multiplicative slack), then the training is incentivized to increase the overall false positive rate because that loosens the constraint further (the same effect occurs for additive slack, but the effect tends to be larger for multiplicative slack).

Constraints can also be expressed *pairwise* between groups, instead of against the global rate:

$$p^{+}(D_j;\theta) - p^{+}(D_k;\theta) \leq \kappa,$$

for all j, k pairs. Our experience is that constraints that involve a larger dataset are generally preferable, and that the smaller the dataset generally the greater the risk of overfitting the constraint or ending up with a degenerate solution to achieve feasibility.

Equality constraints can be expressed by using both a lower-bound and an upper-bound inequality constraint. In practice, we suggest allowing some slack between the lower and upper bounds in order to increase the number of feasible solutions, thereby making stochastic gradient optimization more stable.

4. Optimizing With Constraints

For nonlinear function classes, training a classifier with rate constraints as per Equation 7 is a non-convex optimization over a non-convex constraint set. In this section we provide new theoretical insights and algorithms to optimize general non-convex problems with non-convex constraints, then demonstrate our algorithmic proposals work well in practice with multiple real-world constraints in Section 5. We first outline our two main contributions for this section below.

A Minimal Stochastic Solution: Algorithms that solve non-convex constrained optimization problems based on regret minimization, which includes our approach as well as previous work (e.g. Chen et al., 2017; Agarwal et al., 2018) will output a distribution over θs which has discrete support over T different θ (resulting from the T different epochs of the training algorithm), requiring us to store and sample from T different models. In practice, large T may be problematic to store and serve. Surprisingly, we prove that there always exists an equilibrium that has *sparse support* on at most $m + 1$ choices of model parameters, where m is the number of constraints. We use this result to provide a new practical algorithm to shrink the approximated equilibrium down to a nearly-optimal and nearly-feasible solution supported on at most $m + 1$ models, which is guaranteed to be at least as good as the original stochastic classifier supported on T models.

Handling Non-Differentiable Constraints: A key issue for Equation 7 is the non-differentiability of the constraints due to the indicators in the rate constraints. To handle this, in Section 4.3, we introduce a new formulation we call the *proxy-Lagrangian* that changes the standard two-player *zero-sum* game to a two-player *non-zero-sum* game, which presents new challenges to analysis. In fact, solving a Nash equilibrium is PPAD-complete in the non-zero-sum setting (Chen and Deng, 2006). We prove that a particular game theory solution concept, which we call *semi-coarse correlated equilibrium*, results in a stochastic classifier that is feasible and optimal. This is surprising because the semi-coarse correlated equilibrium is a weaker notion of equilibrium than Nash equilibrium. We go on to provide a novel algorithm that converges to such an equilibrium. To our knowledge, we give the first reduction to this particular solution concept and the first practical use for it, which may be of independent interest. Interestingly, the θ-player needs to only minimize the usual external regret, but the λ-player must minimize the *swap regret* (Blum and Mansour, 2007), a stronger notion of regret. While the resulting distribution is supported on (a possibly large number of) (θ, λ) pairs, applying the same "shrinking" procedure as before yields a distribution over only $m + 1$ of the θs that is at least as good as the original.

In Section 4.1, we handle the optimization of the zero-sum Lagrangian game with an oracle-based algorithm and introduce our proposed "shrinking" procedure. Then, in Section 4.2 we introduce the concept of proxy constraints, describe how it is useful to handle non-differentiable constraints, and formulate the non-zero-sum modification of the Lagrangian, which we call the proxy-Lagrangian. Section 4.3 describes the equilibrium required out of this non-zero-sum game so that it will correspond to an approximately feasible and optimal solution to the constrained optimization problem. Section 4.4 gives an oracle-based procedure for solving for such an equilibrium. Section 4.5 gives a more practical stochastic gradient-based optimizer along with improved guarantees in the convex setting. Finally, Section 4.6 shows that the "shrinking" procedure holds for the non-zero-sum solution as well.

Algorithm 1 Optimizes the Lagrangian formulation (Equation 3) in the non-convex setting via the use of an approximate Bayesian optimization oracle $\mathcal{O}_\rho$ (Definition 1) for the θ-player. The parameter R is the radius of the Lagrange multiplier space $\Lambda := \left\{\lambda \in \mathbb{R}_+^m : \|\lambda\|_1 \leq R\right\}$, and the function Π_Λ projects its argument onto Λ w.r.t. the Euclidean norm.

```
OracleLagrangian ($R \in \mathbb{R}_+, \mathcal{L} : \Theta \times \Lambda \to \mathbb{R}, \mathcal{O}_\rho : (\Theta \to \mathbb{R}) \to \Theta, T \in \mathbb{N}, \eta_\lambda \in \mathbb{R}_+$):
    Initialize $\lambda^{(1)} = 0$
    For $t \in [T]$:
        Let $\theta^{(t)} = \mathcal{O}_\rho\left(\mathcal{L}\left(\cdot, \lambda^{(t)}\right)\right)$                // Oracle optimization
        Let $\Delta_\lambda^{(t)}$ be a gradient of $\mathcal{L}\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. $\lambda$
        Update $\lambda^{(t+1)} = \Pi_\Lambda\left(\lambda^{(t)} + \eta_\lambda \Delta_\lambda^{(t)}\right)$          // Projected gradient update
    Return $\theta^{(1)}, \dots, \theta^{(T)}$ and $\lambda^{(1)}, \dots, \lambda^{(T)}$
```

4.1. Lagrangian Optimization In The Non-convex Setting

We start by assuming an approximate Bayesian optimization oracle (defined in Section 4.1.1), which enables us to use the Lagrangian formulation and not relax the non-convex and/or non-differentiable constraints. This setting is a slight generalization of that presented in Agarwal et al. (2018). Algorithm 1 solves for a stochastic solution to the non-convex constrained optimization problem. It proceeds by playing the following for T rounds: the model parameter player plays best-response (that is, the θ which minimizes the Lagrangian given the last choice of Lagrange multipliers), and the Lagrange multiplier player plays a regret minimizing strategy (here we use projected SGD).

Our first contribution of this section (in Section 4.1.2) is showing that the resulting stochastic classifier is provably approximately feasible and optimal in expectation. This extends the fair classification work of Agarwal et al. (2018) to our slightly more general setting. Our second contribution comes in Section 4.1.4: we will show how the support of the stochastic solution can be efficiently "shrunk" to one that is *at least as good*, but is supported on only $m+1$ solutions and is shown to also have a considerable gain empirically.

4.1.1. Oracle For Unconstrained Non-convex Minimization (Additive Approximation)

Algorithm 1, like the robust optimization algorithm of Chen et al. (2017), requires an *oracle* for performing approximate non-convex minimization. The oracle is a simply a function that takes in a function f and returns an approximate minimizer of f.

Definition 1 *A ρ-approximate Bayesian optimization oracle is a function $\mathcal{O}_\rho : (\Theta \to \mathbb{R}) \to \Theta$ such that:*

$$f\left(\mathcal{O}_\rho\left(f\right)\right) \leq \inf_{\theta^* \in \Theta} f\left(\theta^*\right) + \rho$$

for any $f : \Theta \to \mathbb{R}$ that can be written as a nonnegative linear combination of the objective and constraint functions $g_0, g_1, \dots, g_m$.

The oracle will be used by the θ-player, and the λ-player will use projected gradient ascent. We note that this is a standard assumption in order to obtain theoretical guarantees. (e.g. see Chen et al. (2017), which uses a multiplicative instead of additive approximation).

4.1.2. APPROXIMATE MIXED NASH EQUILIBRIUM

We characterize the relationship between an approximate Nash equilibrium of the Lagrangian game, and a nearly-optimal nearly-feasible solution to the non-convex constrained problem (Equation 2) in our theorem below. This theorem has a few differences from the more typical equivalence between Nash equilibria and optimal feasible solutions in the convex setting. First, it characterizes *mixed* equilibria, in that uniformly sampling from the sequences $\theta^{(t)}$ and $\lambda^{(t)}$ can be interpreted as defining distributions over Θ and Λ. Second, we require compact domains in order to prove convergence rates (below) so Λ is taken to consist only of sets of Lagrange multipliers with bounded 1-norm. In Appendix A, this is generalized to p-norms..

As a reminder, a mixed Nash equilibrium to a two-player game is a pair of distributions over the strategy spaces, one distribution assigned to each player such that neither player can improve their expected payoff (over these distributions) by changing their distribution given that the other player uses their assigned distribution. An *ϵ-approximate* mixed Nash equilibrium is where neither player can improve by more than ϵ by changing their assigned distribution given that the other player uses their assigned distribution.

Finally, as a consequence of the compact domains, the feasibility guarantee of Theorem 2 only holds if the Lagrange multipliers are, on average, smaller than the maximum 1-norm radius R. Thankfully, as is shown by the final result of Theorem 2, if there exists a point satisfying the constraints with some margin $\gamma > 0$, then there will exist an R that is large enough to guarantee feasibility to within $O(\epsilon)$.

Theorem 2 *Define:*

$$\Lambda \triangleq \{\lambda \in \mathbb{R}_+^m : \|\lambda\|_1 \leq R\} \tag{14}$$

and let $\theta^{(1)}, \dots, \theta^{(T)} \in \Theta$ and $\lambda^{(1)}, \dots, \lambda^{(T)} \in \Lambda$ be sequences of parameter vectors and Lagrange multipliers that comprise an approximate mixed Nash equilibrium, i.e.:

$$\max_{\lambda^* \in \Lambda} \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}\left(\theta^{(t)}, \lambda^*\right) - \inf_{\theta^* \in \Theta} \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}\left(\theta^*, \lambda^{(t)}\right) \leq \epsilon.$$

Define $\bar{\theta}$ as a random variable for which $\bar{\theta} = \theta^{(t)}$ with probability $1/T$, and let $\bar{\lambda} \triangleq \left(\sum_{t=1}^{T} \lambda^{(t)}\right)/T$. Then $\bar{\theta}$ is nearly-optimal and nearly-feasible in expectation:

$$\mathbb{E}_{\bar{\theta}}\left[g_0\left(\bar{\theta}\right)\right] \leq \inf_{\theta^* \in \Theta : \forall i. g_i(\theta^*) \leq 0} g_0\left(\theta^*\right) + \epsilon \quad \text{and} \quad \max_{i \in [m]} \mathbb{E}_{\bar{\theta}}\left[g_i\left(\bar{\theta}\right)\right] \leq \frac{\epsilon}{R - \|\bar{\lambda}\|_1}.$$

Additionally, if there exists a $\theta' \in \Theta$ that satisfies all of the constraints with margin γ (i.e. $g_i\left(\theta'\right) \leq -\gamma$ for all $i \in [m]$), then:

$$\|\bar{\lambda}\|_1 \leq \frac{\epsilon + B_{g_0}}{\gamma},$$

where $B_{g_0} \geq \sup_{\theta \in \Theta} g_0\left(\theta\right) - \inf_{\theta \in \Theta} g_0\left(\theta\right)$ is a bound on the range of the objective function g_0.

Proof This is a special case of Theorem 9 and Lemma 10 in Appendix A. ■

4.1.3. CONVERGENCE OF ALGORITHM 1

Algorithm 1's convergence rate is given by the following lemma:

Lemma 3 ***(Algorithm 1)*** *Suppose that Λ and R are as in Theorem 2, and define $B_\Delta \geq \max_{t\in[T]} \left\| \Delta_\lambda^{(t)} \right\|_2$. If we run Algorithm 1 with the step size $\eta_\lambda := R/B_\Delta\sqrt{2T}$, then the result satisfies Theorem 2 for:*

$$\epsilon = \rho + RB_\Delta\sqrt{\frac{2}{T}},$$

where ρ is the error associated with the oracle $\mathcal{O}_\rho$.

Combined with Theorem 2, we therefore have that if R is sufficiently large, then Algorithm 1 will converge to a distribution over Θ that is, in expectation, $O(\rho)$-far from being optimal and feasible at a $O(1/\sqrt{T})$ rate, where ρ is defined in Section 4.1.1.

4.1.4. SHRINKING THE STOCHASTIC SOLUTION

A disadvantage of Algorithm 1 is that it results in a mixture of T solutions, which may be large and thus undesirable in practice. However, we can show that much smaller mixed Nash equilibria exist:

Lemma 4 *If Θ is a compact Hausdorff space, Λ is compact, and the objective and constraint functions $g_0, g_1, \dots, g_m$ are continuous, then the Lagrangian game (Equation 3) has a mixed Nash equilibrium pair (θ, λ) where θ is a random variable supported on at most $m+1$ elements of Θ, and λ is non-random.*

Proof Follows from Theorem 13 in Appendix B. ■

We do not content ourselves with merely having shown the existence of such an equilibrium. Fortunately, we can re-formulate the problem of finding the optimal ϵ-feasible mixture of the $\theta^{(t)}$s as a linear program (LP) that can be solved to *shrink* the support set to $m+1$ solutions. We must first evaluate the objective and constraint functions for every $\theta^{(t)}$, yielding a T-dimensional vector of objective function values, and m such vectors of constraint function evaluations, which are then used to specify the LP.

Lemma 5 *Let $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(T)} \in \Theta$ be a sequence of T "candidate solutions" of Equation 2. Define $\vec{g_0}, \vec{g_i} \in \mathbb{R}^T$ such that $(\vec{g_0})_t = g_0\left(\theta^{(t)}\right)$ and $(\vec{g_i})_t = g_i\left(\theta^{(t)}\right)$ for $i \in [m]$, and consider the linear program:*

$$\begin{aligned} \min_{p\in\Delta^T} \; & \langle p, \vec{g_0} \rangle \\ \text{s.t.} \; & \langle p, \vec{g_i} \rangle \leq \epsilon \textit{for all } i \in 1, \dots, m, \end{aligned}$$

where Δ^T is the T-dimensional simplex. Then every vertex p^* of the feasible region—in particular an optimal one—has at most $m^* + 1 \leq m + 1$ nonzero elements, where m^* is the number of active $\langle p^*, \vec{g_i} \rangle \leq \epsilon$ constraints.

This lemma suggests a two-phase approach to actually finding the $m + 1$ stochastic solution. In the first phase, apply Algorithm 1, yielding a sequence of iterates for which the uniform distribution over the $\theta^{(t)}$s is approximately feasible and optimal. Then apply the procedure of Lemma 5 to find the *best* distribution over these iterates, which in particular can be no worse than the uniform distribution, and is supported on at most $m + 1$ iterates.

4.2. Proxy Constraints And A Non-Zero Sum Game

Most real-world machine learning implementations use first-order methods (even on non-convex problems, e.g. DNNs); however, to use these methods, one must have gradients which are unavailable for rate constraints due to the indicators. Since the constraint functions are piecewise-constant, their gradients are zero almost everywhere, and a gradient-based method cannot be expected to succeed. In general, for constrained optimization problems in the form of Equation 2, non-differentiable constraints arise naturally when one wishes to constrain counts or proportions.

The obvious solution is to use a surrogate for each indicator. For example, we might consider replacing the indicators defining a rate with sigmoids, and then optimizing the Lagrangian. This solves the differentiability problem, but introduces a new one: a (mixed) Nash equilibrium would correspond to a solution satisfying the sigmoid-relaxed constraints, instead of the *actual* constraints. Interestingly, it turns out that we can seek to satisfy the original un-relaxed constraints, even while using a surrogate. Our proposal is motivated by the observation that, while differentiating the Lagrangian (Equation 3) w.r.t. θ requires differentiating the constraint functions $g_i(\theta)$, to differentiate it w.r.t. λ we only need to *evaluate* them. Hence, a surrogate is only necessary for the θ-player; the λ-player can continue to use the original constraint functions.

We refer to a surrogate that is used by only one of the two players as a "proxy", and introduce the notion of "proxy constraints" by taking $\tilde{g}_i(\theta)$ to be a sufficiently-smooth upper bound on $g_i(\theta)$ for $i \in [m]$, and formulating two functions that we call "proxy-Lagrangians":

$$\begin{aligned}\mathcal{L}_\theta(\theta, \lambda) &\triangleq \lambda_1 g_0(\theta) + \sum_{i=1}^{m} \lambda_{i+1} \tilde{g}_i(\theta) \qquad (15)\\ \mathcal{L}_\lambda(\theta, \lambda) &\triangleq \sum_{i=1}^{m} \lambda_{i+1} g_i(\theta),\end{aligned}$$

where we restrict Λ to be the $(m+1)$-dimensional simplex Δ^{m+1}. The θ-player seeks to minimize $\mathcal{L}_\theta(\theta, \lambda)$, while the λ-player seeks to maximize $\mathcal{L}_\lambda(\theta, \lambda)$. Notice that the $\tilde{g}_i$s are *only* used by the θ-player. Intuitively, the λ-player chooses how much to weigh the proxy constraint functions, but—and this is the key to our proposal—does so in such a way as to satisfy the *original* constraints.

Viewed as a two-player game, what we have changed is that now the θ and λ players each have their own payoff functions $\mathcal{L}_\theta(\theta, \lambda)$ and $\mathcal{L}_\lambda(\theta, \lambda)$ respectively, making the game *non-zero sum*. Finding a Nash equilibrium of a non-zero-sum game is much more difficult than for a zero-sum game—in fact, it's PPAD-complete even in the finite setting (Chen and Deng, 2006). We will present a procedure which approximates a *weaker* type of equilibrium: instead of converging to a Nash equilibrium, it converges to a new solution concept, which we call a *semi-coarse correlated equilibrium*. Despite being weaker than a Nash equilibrium, we show that it still corresponds to a nearly-optimal and nearly-feasible solution to constrained optimization in expectation.

The proxy-Lagrangian formulation leads to a tighter approximation than the popular approach of using a surrogate for *both* players, as has been previously proposed, e.g. for Neyman-Pearson classification (Davenport et al., 2010; Gasso et al., 2011), and AUC optimization (Eban et al., 2017). Those proposals optimize a simpler zero-sum game, but one that is a worse reflection of the true goal. In the experimental section, we will provide evidence that the proposed proxy-Lagrangian formulation can provide higher accuracy while still satisfying the constraints. This is especially important when the rate constraints express real-world restrictions on how the learned model is permitted to behave.

4.3. Proxy-Lagrangian Equilibrium

For the proxy-Lagrangian game (Equation 15), we cannot expect to find a Nash equilibrium, at least not efficiently, since it is non-zero-sum. However, the analogous result to Theorem 2 requires a *weaker* type of equilibrium: a joint distribution over Θ and Λ w.r.t. which the θ-player can only make a negligible improvement compared to the best constant strategy, and the λ-player compared to the best action-swapping strategy; this is a type of Φ-correlated equilibrium (Rakhlin et al., 2011). We call this semi-coarse-correlated equilibrium because it exhibits properties of a coarse-correlated equilibrium for one player (θ-player) and that of correlated equilibrium for the other player (λ-player). In a coarse-correlated equilibrium, each player is assigned a distribution over their respective strategy spaces where these distributions can be mutually dependent and no player can improve their payoff by switching to any *fixed* strategy given that the other players use their assigned distributions. In a correlated equilibrium, again each player is assigned a distribution over their respective strategy spaces where these distributions can be mutually dependent, but no player can improve their payoff by changing their assigned distribution given that the other players use their assigned distributions.

We present our theorem showing the achievability of this type of equilibrium, then we present Algorithm 2 to satisfy the theorem.

Theorem 6 *Define $\mathcal{M}$ as the set of all left-stochastic $(m+1)\times(m+1)$ matrices, $\Lambda \triangleq \Delta^{m+1}$ as the $(m+1)$-dimensional simplex, and assume that each $\tilde{g}_i$ upper bounds the corresponding g_i. Let $\theta^{(1)},\dots,\theta^{(T)} \in \Theta$ and $\lambda^{(1)},\dots,\lambda^{(T)} \in \Lambda$ be sequences satisfying:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_\theta\left(\theta^{(t)},\lambda^{(t)}\right) - \inf_{\theta^*\in\Theta}\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_\theta\left(\theta^*,\lambda^{(t)}\right) \leq \epsilon_\theta$$

$$\max_{M^*\in\mathcal{M}}\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_\lambda\left(\theta^{(t)},M^*\lambda^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_\lambda\left(\theta^{(t)},\lambda^{(t)}\right) \leq \epsilon_\lambda .$$

Define $\bar{\theta}$ as a random variable for which $\bar{\theta} = \theta^{(t)}$ with probability $\lambda_1^{(t)} / \sum_{s=1}^{T}\lambda_1^{(s)}$, and let $\bar{\lambda} \triangleq \left(\sum_{t=1}^{T}\lambda^{(t)}\right)/T$. Then $\bar{\theta}$ is nearly-optimal and nearly-feasible in expectation:

$$\mathbb{E}_{\bar{\theta}}\left[g_0\left(\bar{\theta}\right)\right] \leq \inf_{\theta^*\in\Theta:\forall i.\tilde{g}_i(\theta^*)\leq 0} g_0\left(\theta^*\right) + \frac{\epsilon_\theta+\epsilon_\lambda}{\bar{\lambda}_1} \tag{16}$$

and,

$$\max_{i\in[m]}\mathbb{E}_{\bar{\theta}}\left[g_i\left(\bar{\theta}\right)\right] \leq \frac{\epsilon_\lambda}{\bar{\lambda}_1}. \tag{17}$$

Additionally, if there exists a $\theta' \in \Theta$ that satisfies all of the proxy constraints with margin γ (i.e. $\tilde{g}_i\left(\theta'\right) \leq -\gamma$ for all $i \in [m]$), then:

$$\bar{\lambda}_1 \geq \frac{\gamma-\epsilon_\theta-\epsilon_\lambda}{\gamma+B_{g_0}},$$

Algorithm 2 Optimizes the proxy-Lagrangian formulation (Equation 15) in the non-convex setting via the use of an approximate Bayesian optimization oracle $\mathcal{O}_\rho$ (Definition 1, but with $\tilde{g}_i$s instead of g_is in the linear combination defining f) for the θ-player, with the λ-player minimizing swap regret. The $\pi(M)$ operation on line 3 results in a stationary distribution of M (i.e. a $\lambda \in \Lambda$ such that $M\lambda = \lambda$, which can be derived from the top eigenvector).

```
   OracleProxyLagrangian $\left(\mathcal{L}_\theta, \mathcal{L}_\lambda : \Theta \times \Delta^{m+1} \rightarrow \mathbb{R}, \mathcal{O}_\rho : (\Theta \rightarrow \mathbb{R}) \rightarrow \Theta, T \in \mathbb{N}, \eta_\lambda \in \mathbb{R}_+\right)$:
1      Initialize $M^{(1)} \in \mathbb{R}^{(m+1)\times(m+1)}$ with $M_{i,j} = 1/(m+1)$
2      For $t \in [T]$:
3          Let $\lambda^{(t)} = \pi\left(M^{(t)}\right)$                                    // Stationary distribution of $M^{(t)}$
4          Let $\theta^{(t)} = \mathcal{O}_\rho\left(\mathcal{L}_\theta\left(\cdot, \lambda^{(t)}\right)\right)$                                  // Oracle optimization
5          Let $\Delta_\lambda^{(t)}$ be a gradient of $\mathcal{L}_\lambda\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. $\lambda$
6          Update $\tilde{M}^{(t+1)} = M^{(t)} \odot .\exp\left(\eta_\lambda \Delta_\lambda^{(t)} \left(\lambda^{(t)}\right)^T\right)$          // $\odot$ and $.\exp$ are element-wise
7          Project $M_{:,i}^{(t+1)} = \tilde{M}_{:,i}^{(t+1)} / \left\|\tilde{M}_{:,i}^{(t+1)}\right\|_1$ for $i \in [m+1]$          // Column-wise projection
8      Return $\theta^{(1)}, \ldots, \theta^{(T)}$ and $\lambda^{(1)}, \ldots, \lambda^{(T)}$
```

where $B_{g_0} \geq \sup_{\theta \in \Theta} g_0(\theta) - \inf_{\theta \in \Theta} g_0(\theta)$ is a bound on the range of the objective function g_0.

Proof This is a special case of Theorem 11 and Lemma 12 in Appendix A. ■

Notice that Equation 17 guarantees feasibility w.r.t. the original constraints, while Equation 16 shows that the solution minimizes the objective approximately as well as the best solution that's feasible w.r.t. the *proxy* constraints. Hence, the guarantee for minimizing the objective is no better than what we would have obtained if we took $g_i \overset{\triangle}{=} \tilde{g}_i$ for all $i \in [m]$, and optimized the Lagrangian as in Section 4.1. However, because the feasible region w.r.t. the original constraints is larger (perhaps significantly so) than that w.r.t. the proxy constraints, the proxy-Lagrangian approach has more "room" to find a better solution in practice (this is demonstrated in the experiments).

One key difference between this result and Theorem 2 is that the R bound on λ is now gone. Instead, its role, and that of $\left\|\bar{\lambda}\right\|_1$, is played by the first coordinate of $\bar{\lambda}$. Inspection of Equation 15 reveals that, if one or more of the constraints are violated, then the λ-player would prefer the corresponding entries in λ to be higher, which in turn causes λ_1 to become closer to 0 from our procedures. Likewise, if they are satisfied (with some margin), then it would prefer the entries after the first in λ to be 0 which causes λ_1 to be one in our procedures. In other words, the first coordinate of $\lambda^{(t)}$ encodes the λ-player's belief about the feasibility of $\theta^{(t)}$, for which reason $\theta^{(t)}$ is weighted by $\lambda_1^{(t)}$ in the density defining $\bar{\theta}$.

4.4. Proxy-Lagrangian Optimization Algorithm

To optimize the proxy-Lagrangian formulation, we present Algorithm 2, which is motivated by the observation that, while Theorem 6 only requires that the $\theta^{(t)}$ sequence suffer low external regret w.r.t. $\mathcal{L}_\theta\left(\cdot, \lambda^{(t)}\right)$, the condition on the $\lambda^{(t)}$ sequence is stronger, requiring it to suffer low *swap regret* (Blum and Mansour, 2007) w.r.t. $\mathcal{L}_\lambda\left(\theta^{(t)}, \cdot\right)$.

Algorithm 3 Optimizes the proxy-Lagrangian formulation (Equation 15) in the convex setting, with the θ-player minimizing external regret, and the λ-player minimizing swap regret. The $\pi(M)$ operation on line 4 outputs the stationary distribution of M (that is, a $\lambda \in \Lambda$ such that $M\lambda = \lambda$) which can be derived from the top eigenvector. The function Π_Θ projects its argument onto Θ w.r.t. the Euclidean norm.

```
    StochasticProxyLagrangian $\left(\mathcal{L}_\theta, \mathcal{L}_\lambda : \Theta \times \Delta^{m+1} \to \mathbb{R}, T \in \mathbb{N}, \eta_\theta, \eta_\lambda \in \mathbb{R}_+\right)$:
1       Initialize $\theta^{(1)} = 0$                                          // Assumes $0 \in \Theta$
2       Initialize $M^{(1)} \in \mathbb{R}^{(m+1)\times(m+1)}$ with $M_{i,j} = 1/(m+1)$
3       For $t \in [T]$:
4           Let $\lambda^{(t)} = \pi\left(M^{(t)}\right)$                       // Stationary distribution of $M^{(t)}$
5           Let $\check{\Delta}_\theta^{(t)}$ be a stochastic subgradient of $\mathcal{L}_\theta\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. $\theta$
6           Let $\Delta_\lambda^{(t)}$ be a stochastic gradient of $\mathcal{L}_\lambda\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. $\lambda$
7           Update $\theta^{(t+1)} = \Pi_\Theta\left(\theta^{(t)} - \eta_\theta \check{\Delta}_\theta^{(t)}\right)$          // Projected SGD update
8           Update $\tilde{M}^{(t+1)} = M^{(t)} \odot .\exp\left(\eta_\lambda \Delta_\lambda^{(t)} \left(\lambda^{(t)}\right)^T\right)$    // $\odot$ and $.\exp$ are element-wise
9           Project $M_{:,i}^{(t+1)} = \tilde{M}_{:,i}^{(t+1)} / \left\|\tilde{M}_{:,i}^{(t+1)}\right\|_1$ for $i \in [m+1]$      // Column-wise projection
10      Return $\theta^{(1)}, \ldots, \theta^{(T)}$ and $\lambda^{(1)}, \ldots, \lambda^{(T)}$
```

Hence, the θ-player uses the oracle to minimize external regret, while the λ-player uses a swap-regret minimization algorithm of the type proposed by Gordon et al. (2008), yielding the convergence guarantee:

Lemma 7 ***(Algorithm 2)*** *Suppose that $\mathcal{M}$ and Λ are as in Theorem 6, and define the upper bound $B_\Delta \geq \max_{t\in[T]} \left\|\Delta_\lambda^{(t)}\right\|_\infty$.*

If we run Algorithm 2 with the step size $\eta_\lambda := \sqrt{(m+1)\ln(m+1)/TB_\Delta^2}$, then the result satisfies satisfies the conditions of Theorem 6 for:

$$
\begin{aligned}
\epsilon_\theta =& \rho \\
\epsilon_\lambda =& 2B_\Delta \sqrt{\frac{(m+1)\ln(m+1)}{T}},
\end{aligned}
$$

where ρ is the error associated with the oracle $\mathcal{O}_\rho$.

4.5. Practical Stochastic Proxy-Lagrangian Algorithm

Algorithm 3 is designed for the setting of a convex objective, thus we can safely use SGD for the θ-updates instead of the oracle and enjoy a more practical procedure. We stress that this is a considerable improvement over previous Lagrangian methods in the convex setting, as they require both the loss and constraints to be convex in order to attain optimality and feasibility guarantees. Here, while we assume convexity of the objective and proxy-constraints, the *original* constraints do not need to be convex, but we are still able to prove similar guarantees.

Lemma 8 ***(Algorithm 3)*** *Suppose that Θ is a compact convex set, $\mathcal{M}$ and Λ are as in Theorem 6, and that the objective and proxy constraint functions $g_0, \tilde{g}_1, \dots, \tilde{g}_m$ are convex (but not $g_1, \dots, g_m$). Define the three upper bounds $B_\Theta \geq \max_{\theta \in \Theta} \left\Vert \theta \right\Vert_2$, $B_{\check{\Delta}} \geq \max_{t \in [T]} \left\Vert \check{\Delta}_\theta^{(t)} \right\Vert_2$, and $B_\Delta \geq \max_{t \in [T]} \left\Vert \Delta_\lambda^{(t)} \right\Vert_\infty$.*

If we run Algorithm 3 with the step sizes $\eta_\theta := B_\Theta / B_{\check{\Delta}} \sqrt{2T}$ and $\eta_\lambda := \sqrt{(m+1)\ln(m+1)/TB_{\check{\Delta}}^2}$, then the result satisfies the conditions of Theorem 6 for:

$$\epsilon_\theta = 2B_\Theta B_{\check{\Delta}} \sqrt{\frac{1 + 16 \ln \frac{2}{\delta}}{T}}$$

$$\epsilon_\lambda = 2B_\Delta \sqrt{\frac{2(m+1)\ln(m+1)\left(1 + 16\ln\frac{2}{\delta}\right)}{T}},$$

with probability $1 - \delta$ over the draws of the stochastic (sub)gradients.

4.6. Shrinking The Stochastic Proxy Lagrangian Solution

Like Algorithm 1, Algorithms 2 and 3 return a stochastic model with support on T solutions. Again, we show that we can find just as good a stochastic model with minimal support on $m+1$ solutions.

It turns out that the same existence result that we provided for the Lagrangian game (Lemma 4)—of a *Nash* equilibrium—holds for the proxy-Lagrangian (this is Lemma 14 in Appendix B). Furthermore, the exact same linear programming procedure of Lemma 5 can be applied (with the $\vec{g_i}$s being defined in terms of the *original*—not proxy—constraints) to yield a solution with support size $m+1$, and works equally well. This is easy to verify: since $\bar{\theta}$, as defined in Theorem 6, is a distribution over the $\theta^{(t)}$s, and is therefore feasible for the LP, the *best* distribution over the iterates will be at least as good.

5. Experiments

We illustrate the broad applicability of rate constraints and investigate how well different optimization strategies perform. We use the experiments to investigate the following questions:

Do rate constraints help in practice?

- Can we effectively solve the rate-constrained optimization problem?
- Can we get good results *at test time* by training with rate constraints?
- Do rate constraints interact well with other types of constraints (e.g. data-independent monotonicity shape constraints)?

Does the proxy-Lagrangian better solve the constrained optimization problem?

- Does simply using a hinge surrogate for both players as done in prior work over-constrain in practice?
- Does the proposed proxy-Lagrangian formulation result in better solutions?
- With the proxy-Lagrangian, is it necessary in practice for the λ-player to minimize the swap regret or does simply minimizing the external regret work just as well?

Do we really need stochastic classifiers?

- Do the iterates oscillate due to non-existence of an equilibrium in the non-convex setting, causing the last iterate to sometimes be very bad?

- Does the proposed sparsely supported m-stochastic classifier work at least as well in practice as the T-stochastic classifier?
- Does the *best* iterate perform as well as the stochastic classifiers?

To investigate these questions, we compared twelve optimization algorithms for each of seven datasets. Table 4 lists the three benchmark and four real-world datasets we used, each randomly split into train, validation and test sets. We experimented with seven different rate constraints and monotonicity constraints (Groeneboom and Jongbloed, 2014) as described in Table 5 and the following subsections. The last column of Table 5 states whether the classifier has access to information about the different datasets used in the constraints, for example, if there are ten constraints defined on ten different countries, is country also in the feature vector x?

As listed in Table 4, we performed the experiments on linear models and two types of nonlinear models: standard two-layer ReLU neural nets (NN), and a two-layer calibrated ensemble of lattices (Lattices) (Canini et al., 2016).

The rest of this section delves deeper into experimental details and result tables. Then, Section 6 discusses the results and how they provide positive and negative evidence for the above research questions – the reader may prefer to skip to Section 6 and only consult the following experimental details as needed.

All of our experiments are on binary classification datasets and the objective used was hinge loss and the constraint type used (e.g. unconstrained, hinge relaxation or original 0-1) depends on the algorithm and is made clear in the results.

5.1. TensorFlow Implementation

Our experiments were all run using TensorFlow. We have open-sourced our implementation of Lagrangian and proxy-Lagrangian optimization in a library called TensorFlow Constrained Optimization: `https://github.com/google-research/tensorflow_constrained_optimization`.

Experiments with linear models and DNN models used standard TensorFlow functions. Experiments on lattice models used the open-source TensorFlow Lattice package, and consist of learned one-dimensional piecewise linear feature transformations followed by an ensemble of lattices; all model parameters were jointly trained. For more details on lattice models see Gupta et al. (2016); Canini et al. (2016); You et al. (2017). Lattice models can be efficiently constrained for *partial monotonicity* shape constraints, where the term *partial* refers to the practitioner specifying which features can only have a positive (or negative) impact on $f(x)$. To produce the desired partial monotonicity, a large number of data-independent linear inequality constraints are needed, each constraining a pair of model parameters. In the TensorFlow Lattice package, these monotonicity shape constraints are handled by a projection after each minibatch of stochastic gradients.

5.2. Hyperparameter Optimization

For each of the different datasets, we fix the number of loops and model architecture ahead of time to perform well for the unconstrained problem. For the unconstrained problem, we validated the ADAM learning rate. Then for each of the twelve constrained optimization algorithms, we validated two ADAM learning rates, one for optimizing the model parameters θ, and the other for optimizing the constraints parameters λ. All ADAM learning rates were varied by powers of 10 around the usual default of ADAM learning rate of 0.001.

Table 4: Datasets and Model Types Used in Experiments

Dataset	Features	Train	Valid	Test	Model Type	Model Size or # Parameters
Bank Marketing	60	31,647	4,521	9,042	Linear	61
Adult	122	34,189	4,884	9,768	Linear	123
COMPAS	31	4,320	612	1,225	2 Layer NN	10 hidden units
Business Entity	37	11,560	3,856	3,856	2 Layer NN	16 hidden units
Thresholding	7	70,874	10,125	20,250	2 Layer NN	32 hidden units
Map Intent	32	420,000	60,000	120,000	Lattice Ens.	93,600
Filtering	16	1,282,532	183,219	366,440	Lattice Ens.	3,305

The usual strategy of choosing hyperparameters that score best on the validation set is not satisfying in the constrained optimization setting, because now there are two metrics of interest: accuracy and constraint violation, and the appropriate trade-off between them may be problem dependent. One solution researchers turn to is to side-step the issue of choosing one set of hyperparameters, and instead present the Pareto frontier of results over many hyperparameters on the test set. While certainly valuable in a research setting, we must be mindful that in practice one cannot see the Pareto frontier on the test set, and must make a choice for hyperparameters based only on the training and validation sets (as is standard).

For our experiments, we investigate the practical setting in which one must choose one set of hyperparameters on which to evaluate the test set. For that, we need a heuristic to choose the best hyperparameters based only on the training and validation data. We analyzed a number of such heuristics that differently balance the validation accuracy and constraint violation, and were unable to find any heuristic that was perfect, but settled on the following strategy that has some nice properties. Rank each hyperparameter vector β by its validation loss LossRank(β), and create a second ranking of each hyperparamter choice by its maximum constraint violation on the validation set WorstConstraintRank(β). Then choose the hyperparameter vector β that satisfies:

$$\operatorname*{argmin}_{\beta} \max \left\{ \text{LossRank}(\beta), \text{WorstConstraintRank}(\beta) \right\}, \tag{18}$$

with ties broken by the minimizing the validation loss.

This strategy chooses the hyperparameter set that has both low loss and small constraint violations, and guarantees that no other hyperparameter set choice would have both better validation accuracy and smaller constraint violations.

5.3. Algorithms Tested

We experimented with four groups of algorithms:

1. **Unconstrained:** the model is trained without any constraints.
2. **Hinge:** We use a hinge relaxation of the constraints in place of the actual constraints in the Lagrangian as per Algorithm 5.
3. **0-1 swap:** This refers to Algorithm 3, which directly uses the 0-1 constraint in the proxy-Lagrangian, the λ-player minimizes swap-regret and the θ-player minimizes external regret.

Table 5: Constraints Used in Experiments

Dataset	Constraints (# of constraints)	Constraint Group in x?
Bank Marketing	Demographic Parity (5)	Y
Adult	Equal Opportunity (4)	Y
COMPAS	Equal Opportunity (4)	Y
Business Entity Res.	Minimum Recall (18) and Equal Accuracy (1)	Y
Thresholding	Steering Examples Minimum Acc. (1)	N
Map Intent	Not Worse Off (10), Monotonicity (148,800)	Y
Filtering	Loss-only Churn (11), Monotonicity (9,740)	Y

Algorithm 4 Optimizes the Lagrangian formulation *with* proxy constraints. Like the proxy-Lagrangian, this is a non-zero-sum game, but unlike the proxy-Lagrangian, we have no theoretical justification for it. That said, it makes intuitive sense, and works well in practice. The λ-player optimizes based on the proxy-constraints and the θ-player optimizes based on the original constraints. The parameter R is the radius of the Lagrange multiplier space $\Lambda := \left\{\lambda \in \mathbb{R}_{+}^{m} : \|\lambda\|_1 \leq R\right\}$, and the functions Π_Θ and Π_Λ project their arguments onto Θ and Λ (respectively) w.r.t. the Euclidean norm. $\{g_i\}_{i=1}^m, \{\widetilde{g}_i\}_{i=1}^m$ are respectively the original constraints and proxy-constraints.

```
ProxyAdditiveExternalLagrangian $(R \in \mathbb{R}_+, g_0 : \Theta \to \mathbb{R}, \{g_i\}_{i=1}^m, \{\widetilde{g}_i\}_{i=1}^m, T \in \mathbb{N}, \eta_\theta, \eta_\lambda \in \mathbb{R}_+)$:
    Initialize $\theta^{(1)} = 0$, $\lambda^{(1)} = 0$                                  // Assumes $0 \in \Theta$
    For $t \in [T]$:
        Let $\check{\Delta}_\theta^{(t)}$ be a stochastic subgradient of $g_0(\theta^{(t)}) + \sum_{i=1}^m \lambda_i^{(t)} \widetilde{g}_i(\theta)$ w.r.t. $\theta$
        Let $\Delta_\lambda^{(t)}$ be a stochastic gradient of $g_0(\theta^{(t)}) + \sum_{i=1}^m \lambda_i^{(t)} g_i(\theta)$ w.r.t. $\lambda$
        Update $\theta^{(t+1)} = \Pi_\Theta\left(\theta^{(t)} - \eta_\theta \check{\Delta}_\theta^{(t)}\right)$              // Projected SGD updates ...
        Update $\lambda^{(t+1)} = \Pi_\Lambda\left(\lambda^{(t)} + \eta_\lambda \Delta_\lambda^{(t)}\right)$                    //    ...
    Return $\theta^{(1)}, \ldots, \theta^{(T)}$ and $\lambda^{(1)}, \ldots, \lambda^{(T)}$
```

4. **0-1 ext:** This refers to Algorithm 4 training the non-zero-sum game where θ player minimizes the original Lagrangian but the λ-player minimizes external-regret on the Lagrangian with the original constraints replaced by the proxy constraints. This is the "obvious" non-zero-sum analogue of the Lagrangian, but does not enjoy the theoretical guarantees of the proxy-Lagrangian. This is used as a comparison to 0-1 swap to see whether minimizing external regret (instead of the more complex swap regret) suffices in practice.

Then, for each constrained optimization technique, we show the results for the following four solution types:

1. **T-stoch**: the stochastic solution that is the uniform distribution over the T iterates $\theta^{(1)}, ..., \theta^{(T)}$.
2. **m-stoch**: the stochastic solution obtained by applying the "shrinking" technique to the T-stoch solution on the training set, which will have support on at most $m + 1$ deterministic solutions.
3. **Last**: the deterministic solution defined by the last iterate $\theta^{(T)}$.
4. **Best**: the deterministic solution defined by the "best" iterate out of all T iterates $\theta^{(1)}, ..., \theta^{(T)}$, where "best" is chosen by the heuristic given in Equation 18 applied on the training set.

Table 6: Bank Marketing Experiment Results

Algorithm	Train Err.	Valid Err.	Test Err.	Train Vio.	Valid Vio.	Test Vio.
Unconstrained	0.0948	0.0935	0.0937	0.0202	0.0220	0.0152
Hinge m-stoch.	0.0955	0.0954	0.0949	0	-0.0008	-0.0030
Hinge T-stoch.	0.1109	0.1114	0.1121	-0.0177	-0.0181	-0.0179
Hinge Best	0.0964	0.0969	0.0955	-0.0032	-0.0045	-0.0047
Hinge Last	0.1122	0.1129	0.1140	-0.02	-0.02	-0.02
0-1 swap. m-stoch.	0.0939	0.0943	0.0951	0	-0.0005	0.0019
0-1 swap. T-stoch.	0.0963	0.0955	0.0947	0.0004	-0.0003	-0.0031
0-1 swap. Best	0.0936	0.0935	0.0932	-0.0004	-0.0009	-0.0041
0-1 swap. Last	0.0963	0.0957	0.0954	-0.0007	-0.001	-0.0035
0-1 ext. m-stoch.	0.0946	0.0952	0.0946	0	-0.001	-0.0024
0-1 ext. T-stoch.	0.1083	0.1087	0.1085	-0.0135	-0.0146	-0.0139
0-1 ext. Best	0.0963	0.0964	0.0953	-0.0021	-0.0016	-0.0056
0-1 ext. Last	0.1029	0.1032	0.1010	-0.0046	-0.0072	-0.0056

We note that in the non-convex proxy-Lagrangian setting, the 0-1 swap algorithm's T-stoch or m-stoch solutions come with theoretical guarantees if we replace the SGD with the approximate optimization oracle. In contrast, the 0-1 ext algorithm has no such guarantees, but is simpler. Similarly, in the non-convex setting, the deterministic solutions will not have any guarantees, but are even simpler.

5.4. Bank Marketing

The Bank Marketing UCI benchmark dataset (Lichman, 2013) classifier predicts whether someone will sign up for the bank product being marketed. This dataset was used to test improving statistical parity for a linear model in Zafar et al. (2015) but with only one protected group based on age. We similarly use a linear model and age as a protected feature, but create 5 protected groups based on the five training set quantiles of age. We add a statistical parity rate constraint for each of the five age quantiles with an additive slack of 2%:

$$p^+(D_k;\theta) \leq p^+(D;\theta)) - .02,$$

where D_k are the training examples from the kth protected group for $k = 1, 2, \dots, 5$, and D are all the training examples.

The results can be found in Table 6. We note that the Hinge Last solution is a *degenerate* solution in that it always predicts the a priori more probable class.

5.5. Adult

We used the benchmark Adult income UCI dataset (Lichman, 2013). The goal is to predict whether someone makes more than 50k per year, and also do well at the *equal opportunity* fairness metric. We used four protected groups: two race-based (Black or White) and two sex-based (Male or Female). We preprocessed the dataset consistent with Zafar et al. (2015) and Goh et al. (2016). Goh et al. (2016)

Table 7: Adult Experiment Results

Algorithm	Train Err.	Valid Err.	Test Err.	Train Vio.	Valid Vio.	Test Vio.
Unconstrained	0.1421	0.1348	0.1428	0.0803	0.0604	0.0555
Hinge m-stoch.	0.1431	0.1348	0.1442	0	-0.0088	0.0025
Hinge T-stoch.	0.1462	0.1394	0.1481	-0.0409	-0.0372	-0.0436
Hinge Best	0.1424	0.1333	0.1447	-0.0280	-0.0154	-0.0317
Hinge Last	0.1532	0.1490	0.1551	-0.0174	-0.0217	-0.0254
0-1 swap. m-stoch.	0.1431	0.1349	0.1432	0.0176	0.0023	0.0559
0-1 swap. T-stoch.	0.1428	0.1365	0.1436	0.0054	0.0354	0.0285
0-1 swap. Best	0.1426	0.1354	0.1440	-0.0016	0.0140	0.0154
0-1 swap. Last	0.1436	0.1358	0.1443	0.0069	0.0248	0.0221
0-1 ext. m-stoch.	0.1418	0.1348	0.1432	0	-0.0019	0.0059
0-1 ext. T-stoch.	0.1441	0.1369	0.1447	0.0034	0.0220	0.0174
0-1 ext. Best	0.1420	0.1348	0.1432	-0.0374	-0.0333	-0.0015
0-1 ext. Last	0.1436	0.1358	0.1448	-0.0116	0.0078	0.0028

showed that by explicitly constraining the difference in coverage and using a *linear* model, they could achieve higher p fairness and better accuracy than earlier work using correlation constraints of Zafar et al. (2015) by up to 0.5% on this dataset.

For these experiments, we added four rate constraints to the training to impose *equal opportunity* at 95%, that is for each of the protected groups (Black, White, Female and Male) the constraints force the classifier's coverage (the proportion classified positive) on the positively labeled examples for each protected group to be at least 95% of the overall coverage on the positively labeled examples:

$$p^+(D_k[y=1];\theta) \geq 0.95p^+(D[y=1];\theta), \tag{19}$$

where D_k are the training examples from the kth protected group for $k = 1, 2, \ldots, 4$, and D are all the training examples.

We use a linear model. The results can be found in Table 7.

5.6. COMPAS

The positive label in the ProPublicas COMPAS recidivism data is a prediction the person will re-offend. The goal is to predict recidivism with fairness constraints and we preprocess this dataset in a similar manner as in the Adult dataset and the protected groups are also similar: two race-based (Black and White) and two sex-based (Male and Female). The classifier we use is a 2 layer neural network with 10 hidden units.

In this experiment, the goals are quite similar to that of the Adult experiment. Our protected groups are again two races (Black and White) and two sexes (Male and Female) and the goal is to constrain equal opportunity such that no group is unfairly getting targeted. However, instead of expressing the constraint with multiplicative slack as in the Adult experiments, we expressed it as an additive slack of 5%:

$$p^+(D_k[y=1];\theta) \leq p^+(D[y=1];\theta) + .05,$$

Table 8: COMPAS Experiment Results

Algorithm	Train Err.	Valid Err.	Test Err.	Train Vio.	Valid Vio.	Test Vio.
Unconstrained	0.3056	0.3160	0.3109	0.1151	0.2143	0.1082
Hinge m-stoch.	0.3711	0.3744	0.3676	0	0.0395	0.0284
Hinge T-stoch.	0.2880	0.3387	0.3198	0.1093	0.1779	0.0917
Hinge Best	0.2840	0.3322	0.3223	0.0803	0.1262	0.0800
Hinge Last	0.2882	0.3322	0.3231	0.1275	0.1968	0.0996
0-1 swap. m-stoch.	0.3132	0.3015	0.3174	0.0004	0.0851	0.0111
0-1 swap. T-stoch.	0.2968	0.3208	0.3219	0.0257	0.1286	0.0547
0-1 swap. Best	0.3009	0.3096	0.3125	0.0281	0.1084	0.0356
0-1 swap. Last	0.3023	0.3096	0.3158	0.0412	0.1153	0.0480
0-1 ext. m-stoch.	0.3145	0.3080	0.3146	0	0.0813	0.0147
0-1 ext. T-stoch.	0.2990	0.3128	0.3086	0.0323	0.1154	0.0321
0-1 ext. Best	0.3106	0.3160	0.3101	-0.0069	0.0797	-0.0085
0-1 ext. Last	0.2935	0.3160	0.3125	0.0330	0.1231	0.0325

where D_k are the training examples from the kth protected group for $k = 1, 2, \ldots, 4$, and D are all the training examples. That is, the positive prediction rate of the positively labeled examples for each protected class can exceed that of the overall dataset by at most 5%.

The results are shown in Table 8.

5.7. Business Entity Resolution

In this entity resolution problem from Google, the task is to classify whether a pair of business descriptions describe the same real-world business. For example, is *Siam Thai Restaurant at Main and 5th* the same business as *Old Siam Thai at 5070 Main St*? Features include measures of similarity of the two business titles, phone numbers, and so on. We add two types of constraints to the training. First, the dataset is world-wide, and for each of the 16 most frequent countries, we imposed a minimum recall rate constraint of 95 percent:

$$p^+(D_k[y = 1]; \theta) \geq .95,$$

where D_k are the training examples from the kth country for $k = 1, 2, \ldots, 16$. It is also known whether each example is a chain business or not. We impose the same minimum recall rate constraint on chain business examples and non-chain business examples. Additionally, we add an equal accuracy constraint that the accuracy on not-chain businesses should not be worse than the accuracy on chain businesses by more than ten percent, as a *proxy fairness constraint* (Gupta et al., 2019) to making sure large and small businesses receive similar performance from the model:

$$\frac{c^+(D_{\text{notCh}}[y = 1]; \theta) + c^-(D_{\text{notCh}}[y = -1]; \theta)}{|D_{\text{notCh}}|} \geq \frac{c^+(D_{\text{ch}}[y = 1]; \theta) + c^-(D_{\text{ch}}[y = -1]; \theta)}{|D_{\text{ch}}|} - 0.1,$$

where ch is an abbreviation for chain.

Table 9: Business Entity Resolution Experiment Results: 2 Layer NN

Algorithm	Train Err.	Valid Err.	Test Err.	Train Vio.	Valid Vio.	Test Vio.
Unconstrained	0.1223	0.1505	0.1520	0.1727	0.2172	0.2357
Hinge m-stoch.	0.2405	0.2509	0.2535	0	0.0341	0.0282
Hinge T-stoch.	0.3308	0.3351	0.3446	-0.0258	0.0196	-0.0082
Hinge Best	0.2657	0.2720	0.2786	-0.0083	0.0437	0.0026
Hinge Last	0.2483	0.2624	0.2617	-0.0175	0.0125	0.0421
0-1 swap. m-stoch.	0.1751	0.1953	0.1983	0	0.0745	0.0898
0-1 swap. T-stoch.	0.1506	0.1749	0.1760	0.0950	0.1427	0.1933
0-1 swap. Best	0.1407	0.1687	0.1696	0.0681	0.1224	0.1706
0-1 swap. Last	0.1699	0.1910	0.1927	0.0252	0.0864	0.0846
0-1 ext. m-stoch.	0.1891	0.2060	0.2063	0	0.0741	0.0752
0-1 ext. T-stoch.	0.1934	0.2082	0.2092	0.0011	0.0652	0.0770
0-1 ext. Best	0.1889	0.2053	0.2049	0.0026	0.0750	0.0750
0-1 ext. Last	0.1968	0.2118	0.2130	0.0008	0.0594	0.0750

We ran this experiment with a two-layer neural network, the results are shown in Table 9. In the top row, one sees that the unconstrained model has a very high maximum constraint violation, because it is very difficult to achieve 95% recall for all regions.

5.8. Thresholding

For this Google problem, a ranked list of hundreds of business results is given for a query, and the task is to threshold the list to return only the results worth showing a user. We use a 2 layer neural network with 32 hidden units as the classifier.

A medium-size labeled set is available with labels that are known to be noisy, and the label noise is not zero-mean and not homogeneous across the feature space. That set is broken uniformly and randomly into train/validation/test sets.

We also have an auxiliary independent set of $1,814$ *steering examples* (see Section 3.5) which were more carefully labeled by expert labelers, and were actively sampled to pinpoint key types of problems. If one only uses the steering examples (ignoring the noisy labeled data), previous experiments have shown that one can stably achieve a 33% cross-validation error rate on the steering examples. The goal is to have a model that gets that 33% error on the steering examples, but also works as well as possible on the larger noisy data.

The top row of Table 10 shows that only training on the noisy train data produces a error rate of 35% on the noisy test data, which violates our goal of 33% error on the steering examples by 3% (that is, it has an error rate 36% on the steering examples).

In the other extreme, only training on the steering examples is also unsatisfying: as reported in the second row of Table 10 that performs poorly on the large noisy test set with an error rate of 39%, because the steering example set does not cover the entire feature space.

Table 10: Thresholding Experiment Results

Algorithm	Train Err.	Valid Err.	Test Err.	Steering Violation
Unconstrained	0.3595	0.3491	0.3538	0.0316
Unconstrained Trained on Steering	0.3909	0.3924	0.3930	-0.0456
Hinge m-stoch.	0.3601	0.3512	0.3582	0
Hinge T-stoch.	0.3635	0.3558	0.3594	-0.0037
Hinge Best	0.3606	0.3509	0.3560	-0.0031
Hinge Last	0.3621	0.3542	0.3594	-0.0003
0-1 swap. m-stoch.	0.3574	0.3500	0.3557	-0.0025
0-1 swap. T-stoch.	0.3593	0.3513	0.3551	0.0010
0-1 swap. Best	0.3561	0.3484	0.3532	-0.0020
0-1 swap. Last	0.3584	0.3497	0.3543	-0.0020
0-1 ext. m-stoch.	0.3605	0.3504	0.3568	-0.0009
0-1 ext. T-stoch.	0.3602	0.352	0.3553	0.0010
0-1 ext. Best	0.3569	0.3486	0.3515	-0.0014
0-1 ext. Last	0.3579	0.3500	0.3539	-0.0009

For the rest of the rows in Table 10, we train on the noisy data with a minimum accuracy rate constraint for 67% accuracy on the steering examples:

$$\frac{c^+(D_{\text{steering}}[y=1];\theta) + c^-(D_{\text{steering}}[y=-1];\theta)}{|D_{\text{steering}}|} \geq 0.67.$$

All of the different optimization methods find essentially feasible solutions, with many able to achieve the same or better test set performance as the unconstrained training (top row).

5.9. Map Intent

For this Google problem, the task is to classify whether a query is seeking a result on a map. We add ten *not worse off* rate constraints for ten regions that constrain the new model training to be at least as accurate as the production classifier is for each of those ten regions.

$$\frac{c^+(D_{\text{region}}[y=1];\theta) + c^-(D_{\text{region}}[y=-1];\theta)}{|D_{\text{region}}|} \geq \kappa_{\text{region}},$$

where κ_{region} is the accuracy of the production classifier for that region. The feature vector x includes ten Bool features that indicate if x belongs to each of these ten regions (each example belongs to at most one region).

Thirty-two dense and categorical features are available. We train a model that is an ensemble of 300 calibrated lattices, where each lattice acts on 8 of the 32 features, with shared calibrators, and the lattices are interpolated using multi-linear interpolation, all implemented using the TensorFlow Lattice package. We enforce monotonicity constraints on 28 of the 32 features, resulting in an additional 148,800 constraints (each one is a linear inequality constraint on a pair of model parameters) applied during training; see Canini et al. (2016) for more technical details.

Table 11: Map Intent Experiment Results

Algorithm	Train Err.	Valid Err.	Test Err.	Train Vio.	Valid Vio.	Test Vio.
Unconstrained	0.3093	0.3122	0.3104	0.0187	0.0162	0.0319
Hinge m-stoch.	0.3130	0.3129	0.3124	0.0182	0.0176	0.0313
Hinge T-stoch.	0.3096	0.3136	0.3106	0.0194	0.0197	0.0210
Hinge Best	0.3056	0.3131	0.3104	0.0172	0.0194	0.0247
Hinge Last	0.3058	0.3130	0.3099	0.0177	0.0189	0.0220
0-1 swap. m-stoch.	0.2949	0.3002	0.2997	-0.0003	0.0025	0.0176
0-1 swap. T-stoch.	0.3004	0.3022	0.3024	0.0022	0.0061	0.0204
0-1 swap. Best	0.2949	0.3002	0.2997	-0.0003	0.0025	0.0176
0-1 swap. Last	0.2953	0.3004	0.3002	0.0013	0.0034	0.0192
0-1 ext. m-stoch.	0.3069	0.3115	0.3101	0.0094	0.0144	0.0231
0-1 ext. T-stoch.	0.3101	0.3121	0.3107	0.0132	0.0157	0.0243
0-1 ext. Best	0.3069	0.3115	0.3101	0.0094	0.0144	0.0231
0-1 ext. Last	0.3071	0.3111	0.3103	0.0096	0.0140	0.0242

5.10. Filtering

For this Google problem, the task is to classify whether a candidate result for a query should be immediately discarded as too irrelevant to be worthy of further processing. For this problem we take as given a base classifier $\tilde{h}$ and the goal is to maximize accuracy with minimal loss-only churn (see Section 3.3 for details). The base classifier h was trained as a regression model to minimize mean squared error with respect to a real-valued label on $[-1, 1]$, but then used as a classifier with decision threshold 0.0 to filter the results. The new classifier is trained on the same training data, but we pre-threshold the real-valued training labels to form binary classification labels, then train the new classifier to minimize the classification error rate. We add ten loss-only churn rate constraints to individually restrict the loss-only churn with respect to the production model for each of ten mutually-exclusive geographic regions to less than 5%:

$$\frac{c^{+}(D_{\text{region}}[y=-1, h=-1]; \theta) + c^{-}(D_{\text{region}}[y=1, h=1]; \theta)}{|D_{\text{region}}[h=y]|} \leq 0.05.$$

That is, we ask that no more than five percent of the base classifier's wins are lost for each of the ten regions. The feature vector x includes ten binary features that indicate if x belongs to one of these ten regions (some examples do not belong to any of the ten regions).

Both the given regression model $h(x)$ and the new classifier $f(x)$ use the same model architecture: both are lattice models that are an ensemble of 50 lattices, where each lattice acts on 6 of 16 continuous-valued features, each feature is calibrated by a monotonic piecewise linear transform that is shared across the lattices, the lattices are interpolated using multilinear interpolation, all model parameters trained jointly using the TensorFlow Lattice package. We enforce monotonicity constraints on 14 of the 16 features, resulting in an additional 9,740 constraints applied during training (each of these is simply a linear inequality constraint on a pair of model parameters); see Canini et al. (2016) for more technical details.

Table 12: Filtering Experiment Results

Algorithm	Train Err.	Valid Err.	Test Err.	Train Vio.	Valid Vio.	Test Vio.
Unconstrained	0.2747	0.2723	0.2761	0.3164	0.3107	0.3227
Hinge m-stoch.	0.3363	0.3362	0.3369	0	-0.0023	-0.0012
Hinge T-stoch.	0.3658	0.3656	0.3665	-0.0297	-0.0262	-0.0243
Hinge Best	0.3404	0.3403	0.3409	-0.0075	-0.0080	-0.0068
Hinge Last	0.3622	0.3618	0.3630	-0.0239	-0.0239	-0.0242
0-1 swap. m-stoch.	0.3230	0.3231	0.3239	0	0.0071	0.0130
0-1 swap. T-stoch.	0.3205	0.3208	0.3217	0.0096	0.0192	0.0227
0-1 swap. Best	0.3175	0.3178	0.3186	0.0081	0.0116	0.0156
0-1 swap. Last	0.3185	0.3189	0.3195	0.0112	0.0146	0.0118
0-1 ext. m-stoch.	0.3231	0.3234	0.3243	0	0.0048	0.0065
0-1 ext. T-stoch.	0.3300	0.3302	0.3309	0.0004	0.0008	0.0014
0-1 ext. Best	0.3180	0.3179	0.3190	0.0079	0.0116	0.0138
0-1 ext. Last	0.3268	0.3272	0.3278	0.0021	0.0055	0.0087

The production classifier $\tilde{h}$ had a test error rate of 39.72%. As hoped, by training specifically for this classification task, the new classifier $f(x)$ achieves lower test error rates: as low as 27.61% for the unconstrained training. However, the high test constraint violation of 32.27% (measured as the maximum violation over the ten regions) shows that the new unconstrained classifier loses a large number of the wins the base classifier had for at least one of the ten countries considered.

6. Discussion Of Experimental Results

Now that we have presented the experimental results, we return to discuss the experimental and theoretical evidence for and against the hypotheses and questions posed at the beginning of Section 5.

6.1. Do Rate Constraints Help In Practice?

Yes, overall the experiments show rate constraints are are a useful machine learning tool. Let us consider some more specific questions.

6.1.1. Can we effectively solve the rate-constrained optimization problem?

Yes, but the optimization algorithm does matter. Note here we are asking whether the optimization problem is well-solved, and thus we focus on the *training* error and the *training* violation.

The good news is that compared to unconstrained (top row in result tables) the 0-1 swap regret m-stochastic optimization (row 6 in result tables) consistently across all experiments did produce lower training constraint violations while still achieving reasonable training error compared with the unconstrained. Recall that each m-stochastic solves a linear program that sparsifies the corresponding T-stochastic such that the constraints are exactly satisfied if the T-stochastic solution is feasible, so it is by design that the m-stochastic solution train constraint violation is exactly 0.0 for many of the experiments. For Adult (see Table 7), the 0-1 swap m-stochastic train error is only $.001$ worse, but the train violation drops from .0803 to .0176. For Bank Marketing (see Table 6), the train error is slightly better for , and the train violation drops from .0202 to 0.0. Similarly for COMPAS (see Table 8), the 0-1 swap m-stochastic has slightly higher training error but drops the train constraint violation from 0.1151 to almost zero. For Business Entity Resolution (Table 9), the training error does increase with 0-1 swap m-stochastic, but it is a reasonable price to pay in training accuracy for the huge reduction of the worst case equal-accuracy or min-recall constraint violation from 0.1727 to 0.0. For the Thresholding problem (Table 10), the 0-1 swap m-stochastic is again slightly better on training error and effectively reduces the constraint violation to 0.0, and similarly for the Map Intent experiment (Table 11), the training error is lower and the training constraint violation is lower. For Filtering (Table 12), the training error for 0-1 swap regret m-stochastic did go up significantly from 0.2747 to 0.3230, but the unconstrained training violation was horrendous at 0.3164 whereas the m-stochastic found a feasible solution. In conclusion for all experiments run, we found the 0-1 swap regret m-stochastic did a good or reasonable job at the optimization problem of minimizing training error and satisfying the constraints on the training set.

In contrast, one can see that using the baseline strategy of approximating all indicators with the hinge throughout the optimization can provide poor or even worse results than the unconstrained. For example, on the Map Intent experiment (see Table 11), the hinge T-stochastic solution manages to have slightly both worse training error and worse training constraint violation than the unconstrained. The other hinge optimizations are also un-compelling in this experiment. In contrast, the swap regret optimizations consistently find good solutions with lower training error and roughly zero training constraint violations. This is a challenging optimization problem because there are ten rate constraints on ten regions of differing sizes.

The baseline strategy of simply taking the last iterate often does a good job at solving the constrained problem, but sometimes is worse at optimizing the constrained problem than even the unconstrained solver. For example, on COMPAS (see Table 8) the Hinge Last training violation is actually bigger than the unconstrained training violation. While Hinge Last does achieve slightly better training error, it hasn't achieve better validation error (or test error), so we don't believe this was simply an unlucky validation of hyperparameter choice. For more details on why last iterate can perform badly, see Section 6.3.1.

While theory dictates a stochastic solution is necessary for guarantees, in practice the T-stochastic solutions can be quite poor, for example on Map Intent (Table 11) the Hinge T-stochastic solution is worse than unconstrained on both training error and training constraint violation. This may be due to bad early iterates, which would be diluted with a longer run time. Compared to the T-stochastic solutions, the m-stochastic solutions are always better on training error and never more violating, as designed.

The best iterate is by definition always at least as good as the last iterate on the training error and/or training violation. For all three optimization strategies (hinge, 0-1 swap regret, 0-1 external regret), the best iterate manages to consistently produce solutions that are better than the unconstrained in terms of training violations and have reasonable or good training errors.

6.1.2. CAN WE GET GOOD TEST RESULTS BY TRAINING WITH RATE CONSTRAINTS?

Yes, mostly. The m-stochastic and best iterate solutions do result in lower test violations and reasonable test errors for six of the seven experiments. However, for Adult (Table 7), the 0-1 swap m-stochastic failed to produce lower test violation nor lower test error than the unconstrained, despite having much lower training and validation violations. Sadly, the good training and validation performance simply did not generalize to the test set. This case is hard in part because the Black constraint in the Adult dataset is based on a relatively small sample: only 345 positive training examples, 42 positive validation examples, and 179 positive test examples.

Overall, small constraint datasets can lead to poor generalization that can significantly hurt the overall metrics. The worst generalization happened with the Business Entity Resolution, where training violations for the proxy-Lagrangian methods ranged from $[0 - .095]$, but the test violations ranged from $[0.075 - 0.19]$. For that experiment, the hinge solutions generalized better, but at the cost of much higher test errors. Business Entity is a particularly hard problem because there are 16 constraints on different regions, some of which have very small datasets, and just like training a model, there is a greater risk of poor generalization if the datasets used in the constraints are small.

For the larger datasets (Map Intent in Table 11 and Filtering in Table 12), the classifier performance was much more similar on training and test sets.

For a further discussion of generalization for rate-constraints, with some theoretical results and practical strategies, see Cotter et al. (2019a).

6.1.3. DO RATE CONSTRAINTS INTERACT WELL WITH OTHER TYPES OF CONSTRAINTS?

We did not see any problems from combining rate constraints with monotonicity constraints. For Map Intent (Table 11), both 0-1 optimization strategies worked reasonably, with the 0-1 swap regret producing attractive solutions that both notably lowered training error and satisfied the constraints on the training set. This shows that the addition of the 148,800 sparse linear inequality constraints for monotonicity did not cause a problem in optimizing the rate constrained problem. Similarly, for the Filtering (Table 12), the addition of the 9,740 sparse linear inequality constraints for monotonicity did not keep the optimizers from satisfying the rate constraints.

6.2. Does the Proxy-Lagrangian Better Solve the Constrained Optimization Problem?

We break this question into a few specific questions.

6.2.1. DOES SIMPLY USING HINGE SURROGATE FOR BOTH PLAYERS OVERCONSTRAIN?

We hypothesized that using the hinge loss as a convex relaxation to the 0-1 indicators in the rate constraints would cause the constrained optimization to find overly-constrained solutions at the cost of more training accuracy than needed to satisfy the constraints. This was not as large an effect as we expected. However, it can be seen in the Business Entity Resolution (Table 9) experiment where the hinge training violations are negative and the training errors are relatively high, whereas the 0-1 m-stochastic solutions crisply achieve the constraint with much lower training errors.

6.2.2. DOES THE PROXY-LAGRANGIAN FORMULATION RESULT IN BETTER SOLUTIONS?

In most experiments, there were trade-offs between test constraint violation and test accuracy which make it difficult to compare the hinge solutions to the proxy-Lagrangian solutions (denoted 0-1 in the tables) on the test metrics.

On the training metrics, there is stronger evidence the $0 - 1$ m-stochastic optimization is in fact doing a better job solving the optimization problem than the hinge m-stochastic. For seven of the seven experiments, the $0 - 1$ ext. m-stochastic produced both lower train error and lower train violation than the Hinge m-stochastic solution. This was also true for five out of the seven experiments for the $0-1$ swap m-stochastic, and the solutions were close for the remaining two experiments.

In the case of Map Intent (Table 11), we clearly see that the Hinge solutions perform worse in both accuracy and fairness constraints than the 0-1 proxy-Lagrangian procedures on both training and testing. In the case of Thresholding, we see that the Hinge procedures seem to do slightly worse in final accuracy at the cost of over-constraining. We see that in Business Entity Resolution (Table 9), the Hinge procedures attain significantly higher errors than the other methods but do attain better constraint satisfaction on testing. Thus, even though proxy-Lagrangian formulation may seem better on a few of the datasets, this effect was not seen consistently across the remaining datasets and thus, the question of whether the proxy-Lagrangian attains better solutions in practice remains inconclusive.

6.2.3. Is minimizing swap regret necessary, or does external regret suffice?

Our theoretical results show that in the proxy-Lagrangian setting, the appropriate type of equilibrium (i.e. semi-coarse correlated equilibrium) has optimality and feasibility guarantees for the original constrained optimization problem. In order to attain such an equilibrium, we needed the λ-player to minimize swap-regret (while the θ-player minimizes the classic external regret). However, minimizing swap-regret involves a more complicated procedure. We used the strategy of Gordon et al. (2008), who showed that any external regret minimizing procedure can be turned into one that minimizes swap regret by a meta-algorithm which runs m copies of the procedure. We questioned whether it would be just as good in practice to use the simpler external-regret minimizing procedure, which still leads to a coarse-correlated equilibrium (which is a *weaker* notion than semi-coarse correlated equilibrium).

Comparing the swap regret to the external regret for the same solution type (m-stochastic/T-stochastic/best/last), the external regret usually ends up with a solution with slightly lower test violations but slightly higher test error. The only exception was the Map Intent experiment in which the swap-regret solutions were both considerably more accurate and better at satisfying the constraints. In conclusion, we have not seen experimental evidence that the extra complexity of swap regret is warranted in practice.

6.3. Do We Really Need Stochastic Classifiers?

Next, we investigate some specific questions regarding the necessity of stochastic solutions over a deterministic classifier.

6.3.1. Do the iterates oscillate in the non-convex setting?

As noted in Section 6.1.1, simply taking the last iterate can produce worse constraint violations to the optimization problem then solving the unconstrained problem. Figure 3 plots the error and constraints for each of the iterates on the COMPAS dataset which shows such oscillation. This suggests that, as we showed in Section 1.4, the phenomenon of the non-convex Lagrangian having no pure Nash equilibrium to which it can converge, may occur in practice.

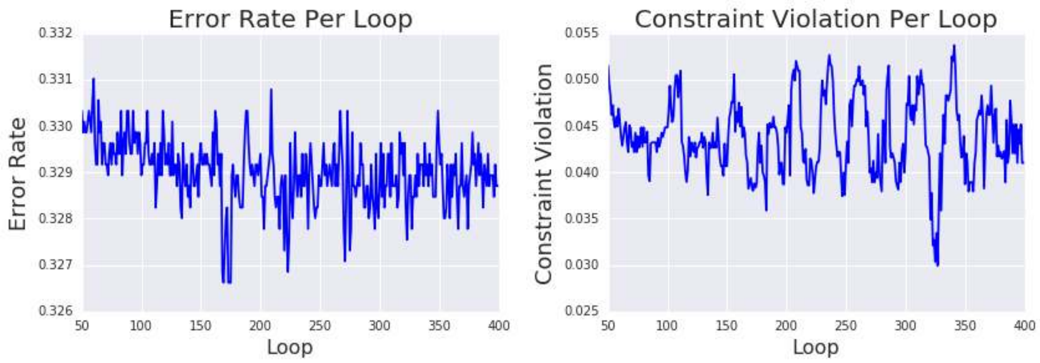

Figure 3: The plots for the errors and constraint violations for each iteration during training on the COMPAS dataset with equal opportunity constraints with an additive slack of 5%. The oscillation due to the conflicting goals of accuracy and constraints suggest that there may be no pure equilibrium to converge to in the non-convex setting.

6.3.2. DOES m-STOCHASTIC BEAT T-STOCHASTIC?

Our theoretical results guarantee that the m-stochastic solution (which is obtained through solving a simple LP on the T-stochastic iterates) will be no worse than the T-stochastic solution by forcing the m-stochastic solution to be at least as feasible as the T stochastic solution, while having no worse error (at least on the training set). Our hope is therefore that our "shrinking" procedure will find better solutions on *test* data.

We see consistently across datasets as well as optimization techniques that the m-stochastic is indeed better than the T-stochastic in terms of both error and constraint violation on training. Part of this effect may be due to the fact that many of the iterates of the T-stochastic perform poorly, for example the early iterates before our procedures are able to get to reasonable solutions. Or during phase-transitions if there is oscillation between satisfying constraints and satisfying error. Fortunately, the shrinking procedure seems to be able to choose a good re-weighting of the T-stochastic solution in order to attain well-performing final results.

We also see that in the vast majority of situations, the test performance for the m-stochastic either surpasses that of the T-stochastic, or there is an accuracy-fairness trade-off between the two (and hence, not straightforward to compare the two).

6.3.3. DOES THE BEST ITERATE PERFORM AS WELL AS THE STOCHASTIC CLASSIFIERS?

We have already established that a stochastic solution may be difficult to avoid, in theory (Section 1.4). However, stochastic solutions are unappealing in practice: they take more memory, are harder to test and debug due to their inherent randomness, and a randomized decision may feel less *fair* in certain contexts (even if the outcomes statistically improve the desired fairness metric). Here, we ask if a stochastic solution is needed *in practice*, based on test metrics.

First, we compare the 0-1 swap regret m-stochastic solution, which is our theoretically preferred stochastic solution, to the 0-1 swap regret best iterate. The 0-1 swap best iterate is never a strictly worse choice than the 0-1 swap m-stochastic. In some cases the m-stochastic solution puts all or most of its weight on the best iterate—for example, for the Map Intent problem (Table 11) the two solutions are identical. In other experiments the solutions differ but both achieve reasonable different trade-offs of test error and test violation, for example on the Thresholding problem (Table 10) and COMPAS (see Table 8), the best iterate has a lower test error, but a higher test constraint violation.

Comparing the m-stochastic solution and best iterate solution for the 0-1 external regret optimization similarly suggests that much of the time the best iterate works just as well in practice.

6.3.4. DOES BEST ITERATE PERFORM BETTER IN PRACTICE THAN LAST ITERATE?

We have established that using the best iterate works well in practice. Now we discuss how much better best is than simply taking the last iterate. In fact, the last iterate is strictly worse at test metrics than the best iterate for 4 of the 7 experiments: Bank, Thresholding, Adult, and Compass; and the two solutions are similar for the other three experiments.

If there are oscillations on the loss and constraint violation (as shown in Figure 6.3 for COMPAS), then the last iterate could be highly unstable and could produce undesirable solutions. In practice, the strongest evidence for last being a risky choice is Hinge Last on COMPAS, where test error went up from 0.3109 to 0.3231, and training violation only went down from 0.1082 to 0.0996.

Overall, the experimental results suggest that the best iterate is preferable to the last iterate.

7. Conclusions, Advice To Practitioners, And Open Questions

In this paper, we provide the most comprehensive study to-date of training classifiers with a broad array of rate constraints, with new theoretical, algorithmic, and experimental results as well as practical insights and guidance for using rate constraints to solve real-world problems. Next, we provide some conclusions, specifically draw out our best advice to practitioners, and note some open questions.

7.1. Advice To Practitioners: How To Train Classifiers With Rate Constraints

Based on our experiments, our advice to practitioners is to optimize the rate-constrained training using either our proposed non-zero-sum variant of the normal Lagrangian formulation (0-1 external regret) and taking the last iterate.

The 0-1 external regret optimization procedure is simple: when optimizing the model parameters θ use stochastic gradient descent as usual with a hinge relaxation of the indicators in the constraints, and when optimizing the Lagrange multipliers λ use stochastic gradient descent, but do not relax the indicators in the rate constraints. If one needs a deterministic solution, ideally one would take the *best iterate*, but this requires storing all the candidate iterates on the Pareto frontier during training, in order to rank them by the training objective and training error at the end, and in the worst case that could be all candidate iterates. However one can control the number of candidate iterates, for example by sub-sampling them, or waiting until late in training to sample them. Simply taking the last iterate usually yielded reasonable results, but we do see in practice that the last iterate may perform strictly worse under all metrics than the best iterate.

We caution against relaxing the indicators for both the θ-player and λ-player (hinge last). It is hardly simpler than the 0-1 external regret optimization, and experimentally generally (but not always) produced worse test results, sometimes notably worse.

7.2. Advice To Practitioners: Plan To Overfit The Constraints

A key issue with using rate constraints is *generalization*: satisfying the constraints on the training examples does not necessarily mean that they will be satisfied on new test sets, and the generalization may be worse if the test examples are drawn from a different distribution. In expressing the rate constraints, one should add in some slack to account for generalization issues, especially if the constraints are optimized on small datasets.

In Cotter et al. (2019a), we extend the ideas of this paper with a focus on generalization. We show that providing different *datasets* to the two players, instead of (or in addition to) different constraint functions, can theoretically and practically improve generalization.

7.3. Advice To Practitioners: How The Constraints Are Specified Matters

We have learned that in practice that *how* one specifies the datasets and slack in a rate constraint is very important - see Section 3.7 for more discussion.

7.4. More Experimental Conclusions

The clearest experimental finding is that treating the optimization as a non-zero-sum two-player game where the λ-player does not relax the indicators in the rate constraints (notated as 0-1 in the experimental tables) does generally help, both in finding a better solution to the optimization problem (i.e. train metrics), and in practice (i.e. test metrics). Another fairly clear experimental finding is that the T-stochastic solution can effectively be sparsified to an m-stochastic solution, generally with improved metrics.

While the T-stochastic solution has better theoretical guarantees than any of our deterministic solutions, especially for large T, in practice we found the deterministic best iterate generally worked better than the T-stochastic solution. Other comparisons were more cloudy, see Section 6 for details.

7.5. On Making Stochastic Classifiers Deterministic

While it is clear that theoretically one needs a stochastic classifier, practitioners may prefer a deterministic classifier. Given that, what is the best way to convert a stochastic classifier into a deterministic one? Recently, Narasimhan et al. (2019b) investigate this question theoretically and experimentally.

7.6. Nonlinear Rate Constraints

We limited our focus to rate constraints that can be written as in Equation 6 as a linear non-negative combination of the positive and negative classification rates on datasets. We touched on the issues posed by nonlinear rate constraints in Section 3 in our discussion of win-loss ratio and precision. As we go to press, newer work shows promise extending these ideas to nonlinear rate constraints (Narasimhan et al., 2019a). However, many open questions remain in handling generalizations of rate constraints, both theoretically and experimentally.

7.7. Rate Constraints For Ranking or Regression Models

Recent work has shown that the presented rate constraint methodology can be intuitively extended to regression and ranking models by defining rate constraints on pairs of examples, forming, for example, analogous *pairwise fairness* definitions (Narasimhan et al., 2020).

7.8. Some Open Theoretical Questions

One open question is how tight our optimality and feasibility guarantees are for our procedures in the following aspects:

- The dependence on the number of iterations T for our guarantees is $O\left(\sqrt{\frac{1}{T}}\right)$. This rate is an artifact of our usage of regret-minimization procedures, but it could be improved through a number of possible techniques, such as variance reduction (e.g. Johnson and Zhang, 2013), or by making stronger assumptions (e.g. strong convexity and/or smoothness).
- The dependence on m, the number of constraints, is $O(\sqrt{m \log m})$, which also comes from the regret-minimization procedures. This is because the λ-player essentially chooses a distribution over $m+1$ actions and this dependence on the number of arms is tight in the context of regret-minimization, but the question remains of whether there are situations where this could be improved upon for constrained optimization for either feasibility or optimality.
- Our results also have a dependence on the model complexity in both feasibility and optimality guarantees. This may be undesirable in models with a large number of parameters, such as modern neural networks. We explored the question of whether we can improve upon this dependence further in follow-up work of Cotter et al. (2019a), which improves the feasibility guarantee. However, further investigation is required to either establish matching lower bounds and/or obtaining tighter results.

Appendix A. Proofs Of Sub$\{$optimality, feasibility$\}$ Guarantees

Theorem 9 ***(Lagrangian Sub$\{$optimality,feasibility$\}$)*** *Define $\Lambda = \left\{\lambda \in \mathbb{R}_+^m : \left\|\lambda\right\|_p \leq R\right\}$, and consider the Lagrangian of Equation 2 given in Equation 3. Suppose that $\theta \in \Theta$ and $\lambda \in \Lambda$ are random variables such that:*

$$\max_{\lambda^* \in \Lambda} \mathbb{E}_\theta \left[\mathcal{L}\left(\theta, \lambda^*\right)\right] - \inf_{\theta^* \in \Theta} \mathbb{E}_\lambda \left[\mathcal{L}\left(\theta^*, \lambda\right)\right] \leq \epsilon, \tag{20}$$

i.e. θ, λ is an ϵ-approximate Nash equilibrium. Then θ is ϵ-suboptimal:

$$\mathbb{E}_\theta \left[g_0\left(\theta\right)\right] \leq \inf_{\theta^* \in \Theta : \forall i \in [m] . g_i(\theta^*) \leq 0} g_0\left(\theta^*\right) + \epsilon.$$

Furthermore, if λ is in the interior of Λ, in the sense that $\left\|\bar{\lambda}\right\|_p < R$ where $\bar{\lambda} := \mathbb{E}_\lambda\left[\lambda\right]$, then θ is $\epsilon / \left(R - \left\|\bar{\lambda}\right\|_p\right)$-feasible:

$$\left\|\left(\mathbb{E}_\theta\left[g_:\left(\theta\right)\right]\right)_+\right\|_q \leq \frac{\epsilon}{R - \left\|\bar{\lambda}\right\|_p},$$

where $g_:\left(\theta\right)$ is the m-dimensional vector of constraint evaluations, and $\left(\cdot\right)_+$ takes the positive part of its argument, so that $\left\|\left(\mathbb{E}_\theta\left[g_:\left(\theta\right)\right]\right)_+\right\|_q$ is the q-norm of the vector of expected constraint violations.

Proof First notice that $\mathcal{L}$ is linear in λ, so:

$$\max_{\lambda^* \in \Lambda} \mathbb{E}_\theta \left[\mathcal{L}\left(\theta, \lambda^*\right) \right] - \inf_{\theta^* \in \Theta} \mathcal{L}\left(\theta^*, \bar{\lambda}\right) \leq \epsilon. \tag{21}$$

Optimality: Choose θ^* to be the optimal *feasible* solution in Equation 21, so that $g_i\left(\theta^*\right) \leq 0$ for all $i \in [m]$, and also choose $\lambda^* = 0$, which combined with the definition of $\mathcal{L}$ (Equation 3) gives that:

$$\mathbb{E}_\theta \left[g_0\left(\theta\right) \right] - g_0\left(\theta^*\right) \leq \epsilon,$$

which is the optimality claim.

Feasibility: Choose $\theta^* = \theta$ in Equation 21. By the definition of $\mathcal{L}$ (Equation 3):

$$\max_{\lambda^* \in \Lambda} \sum_{i=1}^{m} \lambda_i^* \mathbb{E}_\theta \left[g_i\left(\theta\right) \right] - \sum_{i=1}^{m} \bar{\lambda}_i \mathbb{E}_\theta \left[g_i\left(\theta\right) \right] \leq \epsilon.$$

Then by the definition of a dual norm, Hölder's inequality, and the assumption that $\left\|\bar{\lambda}\right\|_p < R$:

$$R \left\| \left(\mathbb{E}_\theta \left[g_:\left(\theta\right) \right] \right)_+ \right\|_q - \left\|\bar{\lambda}\right\|_p \left\| \left(\mathbb{E}_\theta \left[g_:\left(\theta\right) \right] \right)_+ \right\|_q \leq \epsilon.$$

Rearranging terms gives the feasibility claim. ■

Lemma 10 *In the context of Theorem 9, suppose that there exists a $\theta' \in \Theta$ that satisfies all of the constraints, and does so with q-norm margin γ, i.e. $g_i\left(\theta'\right) \leq 0$ for all $i \in [m]$ and $\left\|g_:\left(\theta'\right)\right\|_q \geq \gamma$. Then:*

$$\left\|\bar{\lambda}\right\|_p \leq \frac{\epsilon + B_{g_0}}{\gamma},$$

where $B_{g_0} \geq \sup_{\theta \in \Theta} g_0\left(\theta\right) - \inf_{\theta \in \Theta} g_0\left(\theta\right)$ is a bound on the range of the objective function g_0.

Proof Starting from Equation 20 (in Theorem 9), and choosing $\theta^* = \theta'$ and $\lambda^* = 0$:

$$\begin{aligned}
\epsilon \geq & \mathbb{E}_\theta \left[g_0\left(\theta\right) \right] - \mathbb{E}_\lambda \left[g_0\left(\theta'\right) + \sum_{i=1}^{m} \lambda_i g_i\left(\theta'\right) \right] \\
\epsilon \geq & \mathbb{E}_\theta \left[g_0\left(\theta\right) - \inf_{\theta' \in \Theta} g_0\left(\theta'\right) \right] - \left(g_0\left(\theta'\right) - \inf_{\theta' \in \Theta} g_0\left(\theta'\right) \right) + \gamma \left\|\bar{\lambda}\right\|_p \\
\epsilon \geq & - B_{g_0} + \gamma \left\|\bar{\lambda}\right\|_p .
\end{aligned}$$

Solving for $\left\|\bar{\lambda}\right\|_p$ yields the claim. ■

We next give the optimality and feasibility guarantees for the proxy-Lagrangian formulation. The result shows that the approximate semi-coarse correlated equilibrium to the two-player non-zero sum game based on the proxy-Lagrangian will correspond to an approximately feasible solution to the constrained optimization problem w.r.t. the original constraints which is also approximately optimal compared to the solution which is optimal and feasible w.r.t. the proxy constraints. The conditions for semi-coarse correlated equilibrium are shown in Equation 22. The first line requires that the solution is approximately as good for the θ-player compared to any fixed choice of θ (i.e. which comes from the θ-player using best-response or minimizing external-regret). The second line requires that the solution is approximately as good for the λ-player when compared to any left-stochastic linear transformation of that solution for the λ-player (which is a result from the λ-player optimizing for swap-regret).

The swap-regret guarantee is required to show feasibility. The key idea is to use the swap regret guarantee to show that the difference in the proxy-Lagrangian when shifting the weight λ_1 on the objective g_0 to any of the constraints $g_1, ..., g_m$ will result in only a small change, and thus the constraint violations themselves are small.

Theorem 11 ***(Proxy-Lagrangian Sub*$\{$*optimality,feasibility*$\}$*)*** *Let*

$$\mathcal{M} := \left\{ M \in \mathbb{R}^{(m+1)\times(m+1)} : \forall i \in [m+1] \,. M_{:,i} \in \Delta^{m+1} \right\}$$

be the set of all left-stochastic $(m+1)\times(m+1)$ matrices, and consider the "proxy-Lagrangians" of Equation 2 given in Equation 15. Suppose that $\theta \in \Theta$ and $\lambda \in \Lambda$ are jointly distributed random variables such that:

$$\begin{aligned} \mathbb{E}_{\theta,\lambda}\left[\mathcal{L}_\theta\left(\theta,\lambda\right)\right] - \inf_{\theta^* \in \Theta} \mathbb{E}_{\lambda}\left[\mathcal{L}_\theta\left(\theta^*,\lambda\right)\right] \leq & \epsilon_\theta \\ \max_{M^* \in \mathcal{M}} \mathbb{E}_{\theta,\lambda}\left[\mathcal{L}_\lambda\left(\theta, M^*\lambda\right)\right] - \mathbb{E}_{\theta,\lambda}\left[\mathcal{L}_\lambda\left(\theta,\lambda\right)\right] \leq & \epsilon_\lambda . \end{aligned} \tag{22}$$

Define $\bar{\lambda} := \mathbb{E}_\lambda[\lambda]$, let $(\Omega, \mathcal{F}, P)$ be the probability space, and define a random variable $\bar{\theta}$ such that:

$$\Pr\left\{\bar{\theta} \in S\right\} = \frac{\int_{\theta^{-1}(S)} \lambda_1(x)\, dP(x)}{\int_\Omega \lambda_1(x)\, dP(x)}.$$

In words, $\bar{\theta}$ is a version of θ that has been resampled with λ_1 being treated as an importance weight. In particular $\mathbb{E}_{\bar{\theta}}\left[f\left(\bar{\theta}\right)\right] = \mathbb{E}_{\theta,\lambda}\left[\lambda_1 f(\theta)\right]/\bar{\lambda}_1$ for any $f : \Theta \to \mathbb{R}$. Then $\bar{\theta}$ is nearly-optimal:

$$\mathbb{E}_{\bar{\theta}}\left[g_0\left(\bar{\theta}\right)\right] \leq \inf_{\theta^* \in \Theta : \forall i \in [m] . \tilde{g}_i(\theta^*) \leq 0} g_0\left(\theta^*\right) + \frac{\epsilon_\theta + \epsilon_\lambda}{\bar{\lambda}_1},$$

and nearly-feasible:

$$\left\| \left(\mathbb{E}_{\bar{\theta}}\left[g_:\left(\bar{\theta}\right)\right]\right)_+ \right\|_\infty \leq \frac{\epsilon_\lambda}{\bar{\lambda}_1}.$$

Notice the optimality inequality is weaker than it may appear, since the comparator in this equation is not *the optimal solution w.r.t. the constraints g_i, but rather w.r.t. the* proxy *constraints $\tilde{g}_i$.*

Proof **Optimality:** If we choose M^* to be the matrix with its first row being all-one, and all other rows being all-zero, then $\mathcal{L}_\lambda\left(\theta, M^*\lambda\right) = 0$, which shows that the first term in the LHS of the second line of Equation 22 is nonnegative. Hence, $-\mathbb{E}_{\theta,\lambda}\left[\mathcal{L}_\lambda\left(\theta,\lambda\right)\right] \leq \epsilon_\lambda$, so by the definition of $\mathcal{L}_\lambda$ (Equation 15), and the fact that $\tilde{g}_i \geq g_i$:

$$\mathbb{E}_{\theta,\lambda}\left[\sum_{i=1}^m \lambda_{i+1}\tilde{g}_i\left(\theta\right)\right] \geq -\epsilon_\lambda .$$

Notice that $\mathcal{L}_\theta$ is linear in λ, so the first line of Equation 22, combined with the above result and the definition of $\mathcal{L}_\theta$ (Equation 15) becomes:

$$\mathbb{E}_{\theta,\lambda}\left[\lambda_1 g_0\left(\theta\right)\right] - \inf_{\theta^* \in \Theta}\left(\bar{\lambda}_1 g_0\left(\theta^*\right) + \sum_{i=1}^{m} \bar{\lambda}_{i+1} \tilde{g}_i\left(\theta^*\right)\right) \leq \epsilon_\theta + \epsilon_\lambda. \tag{23}$$

Choose θ^* to be the optimal solution that satisfies the *proxy* constraints $\tilde{g}$, so that $\tilde{g}_i\left(\theta^*\right) \leq 0$ for all $i \in [m]$. Hence:

$$\mathbb{E}_{\theta,\lambda}\left[\lambda_1 g_0\left(\theta\right)\right] - \bar{\lambda}_1 g_0\left(\theta^*\right) \leq \epsilon_\theta + \epsilon_\lambda,$$

which is the optimality claim.

Feasibility: We'll simplify our notation by defining $\ell_1\left(\theta\right) := 0$ and $\ell_{i+1}\left(\theta\right) := g_i\left(\theta\right)$ for $i \in [m]$, so that $\mathcal{L}_\lambda\left(\theta, \lambda\right) = \langle \lambda, \ell_:\left(\theta\right)\rangle$. Consider the first term in the LHS of the second line of Equation 22:

$$\begin{aligned}
\max_{M^* \in \mathcal{M}} \mathbb{E}_{\theta,\lambda}\left[\mathcal{L}_\lambda\left(\theta, M^*\lambda\right)\right] =& \max_{M^* \in \mathcal{M}} \mathbb{E}_{\theta,\lambda}\left[\langle M^*\lambda, \ell_:\left(\theta\right)\rangle\right] \\
=& \max_{M^* \in \mathcal{M}} \mathbb{E}_{\theta,\lambda}\left[\sum_{i=1}^{m+1}\sum_{j=1}^{m+1} M^*_{j,i}\lambda_i \ell_j\left(\theta\right)\right] \\
=& \sum_{i=1}^{m+1} \max_{M^*_{:,i} \in \Delta^{m+1}} \sum_{j=1}^{m+1} \mathbb{E}_{\theta,\lambda}\left[M^*_{j,i}\lambda_i \ell_j\left(\theta\right)\right] \\
=& \sum_{i=1}^{m+1} \max_{j \in [m+1]} \mathbb{E}_{\theta,\lambda}\left[\lambda_i \ell_j\left(\theta\right)\right],
\end{aligned}$$

where we used the fact that, since M^* is left-stochastic, each of its columns is a $(m+1)$-dimensional multinoulli distribution. For the second term in the LHS of the second line of Equation 22, we can use the fact that $\ell_1\left(\theta\right) = 0$:

$$\mathbb{E}_{\theta,\lambda}\left[\sum_{i=2}^{m+1} \lambda_i \ell_i\left(\theta\right)\right] \leq \sum_{i=2}^{m+1} \max_{j \in [m+1]} \mathbb{E}_{\theta,\lambda}\left[\lambda_i \ell_j\left(\theta\right)\right].$$

Plugging these two results into the second line of Equation 22, the two sums collapse, leaving:

$$\max_{i \in [m+1]} \mathbb{E}_{\theta,\lambda}\left[\lambda_1 \ell_i\left(\theta\right)\right] \leq \epsilon_\lambda.$$

By the definition of ℓ_i, and the fact that $\ell_1 = 0$:

$$\left\|\left(\mathbb{E}_{\theta,\lambda}\left[\lambda_1 g_:\left(\theta\right)\right]\right)_+\right\|_\infty \leq \epsilon_\lambda,$$

which is the feasibility claim. ■

Lemma 12 *In the context of Theorem 11, suppose that there exists a $\theta' \in \Theta$ that satisfies all of the* proxy *constraints with margin γ, i.e. $\tilde{g}_i\left(\theta'\right) \leq -\gamma$ for all $i \in [m]$. Then:*

$$\bar{\lambda}_1 \geq \frac{\gamma - \epsilon_\theta - \epsilon_\lambda}{\gamma + B_{g_0}},$$

where $B_{g_0} \geq \sup_{\theta \in \Theta} g_0\left(\theta\right) - \inf_{\theta \in \Theta} g_0\left(\theta\right)$ is a bound on the range of the objective function g_0.

Proof Starting from Equation 23 (in the proof of Theorem 11), and choosing $\theta^* = \theta'$:

$$\mathbb{E}_{\theta,\lambda}\left[\lambda_1 g_0\left(\theta\right)\right] - \left(\bar{\lambda}_1 g_0\left(\theta'\right) + \sum_{i=1}^{m} \bar{\lambda}_{i+1} \tilde{g}_i\left(\theta'\right)\right) \leq \epsilon_\theta + \epsilon_\lambda.$$

Since $\tilde{g}_i\left(\theta'\right) \leq -\gamma$ for all $i \in [m]$:

$$\begin{aligned}
\epsilon_\theta + \epsilon_\lambda \geq & \mathbb{E}_{\theta,\lambda}\left[\lambda_1 g_0\left(\theta\right)\right] - \bar{\lambda}_1 g_0\left(\theta'\right) + \left(1 - \bar{\lambda}_1\right)\gamma \\
\geq & \mathbb{E}_{\theta,\lambda}\left[\lambda_1 \left(g_0\left(\theta\right) - \inf_{\theta' \in \Theta} g_0\left(\theta'\right)\right)\right] - \bar{\lambda}_1 \left(g_0\left(\theta'\right) - \inf_{\theta' \in \Theta} g_0\left(\theta'\right)\right) + \left(1 - \bar{\lambda}_1\right)\gamma \\
\geq & -\bar{\lambda}_1 B_{g_0} + \left(1 - \bar{\lambda}_1\right)\gamma.
\end{aligned}$$

Solving for $\bar{\lambda}_1$ yields the claim. ■

Appendix B. Proofs Of Existence Of Sparse Equilibria

Theorem 13 *Consider a two player game, played on the compact Hausdorff spaces Θ and $\Lambda \subseteq \mathbb{R}^m$. Imagine that the θ-player wishes to minimize $\mathcal{L}_\theta : \Theta \times \Lambda \to \mathbb{R}$, and the λ-player wishes to maximize $\mathcal{L}_\lambda : \Theta \times \Lambda \to \mathbb{R}$, with both of these functions being continuous in θ and linear in λ. Then there exists a Nash equilibrium θ, λ:*

$$\begin{aligned}
\mathbb{E}_\theta\left[\mathcal{L}_\theta\left(\theta, \lambda\right)\right] &= \min_{\theta^* \in \Theta} \mathcal{L}_\theta\left(\theta^*, \lambda\right) \\
\mathbb{E}_\theta\left[\mathcal{L}_\lambda\left(\theta, \lambda\right)\right] &= \max_{\lambda^* \in \Lambda} \mathbb{E}_\theta\left[\mathcal{L}_\lambda\left(\theta, \lambda^*\right)\right].
\end{aligned}$$

where θ is a random variable placing nonzero probability mass on at most $m+1$ *elements of Θ, and $\lambda \in \Lambda$ is non-random.*

Proof There are some extremely similar (and in some ways more general) results than this in the game theory literature (e.g. Bohnenblust et al., 1950; Parthasarathy, 1975), but for our particular (Lagrangian and proxy-Lagrangian) setting it's possible to provide a fairly straightforward proof.

To begin with, Glicksberg (1952) gives that there exists a mixed strategy in the form of two random variables $\tilde{\theta}$ and $\tilde{\lambda}$:

$$\begin{aligned}
\mathbb{E}_{\tilde{\theta},\tilde{\lambda}}\left[\mathcal{L}_\theta\left(\tilde{\theta}, \tilde{\lambda}\right)\right] &= \min_{\theta^* \in \Theta} \mathbb{E}_{\tilde{\lambda}}\left[\mathcal{L}_\theta\left(\theta^*, \tilde{\lambda}\right)\right] \\
\mathbb{E}_{\tilde{\theta},\tilde{\lambda}}\left[\mathcal{L}_\lambda\left(\tilde{\theta}, \tilde{\lambda}\right)\right] &= \max_{\lambda^* \in \Lambda} \mathbb{E}_{\tilde{\theta}}\left[\mathcal{L}_\lambda\left(\tilde{\theta}, \lambda^*\right)\right].
\end{aligned}$$

Since both functions are linear in $\tilde{\lambda}$, we can define $\lambda := \mathbb{E}_{\tilde{\lambda}}\left[\tilde{\lambda}\right]$, and these conditions become:

$$\begin{aligned}\mathbb{E}_{\tilde{\theta}}\left[\mathcal{L}_\theta\left(\tilde{\theta},\lambda\right)\right] =& \min_{\theta^*\in\Theta}\mathcal{L}_\theta\left(\theta^*,\lambda\right) := \ell_{\min}\\ \mathbb{E}_{\tilde{\theta}}\left[\mathcal{L}_\lambda\left(\tilde{\theta},\lambda\right)\right] =& \max_{\lambda^*\in\Lambda}\mathbb{E}_{\tilde{\theta}}\left[\mathcal{L}_\lambda\left(\tilde{\theta},\lambda^*\right)\right].\end{aligned}$$

Let's focus on the first condition. Let $p_\epsilon := \Pr\left\{\mathcal{L}_\theta\left(\tilde{\theta},\lambda\right) \geq \ell_{\min}+\epsilon\right\}$, and notice that $p_{1/n}$ must equal zero for any $n \in \{1,2,\dots\}$ (otherwise we would contradict the above), implying by the countable additivity of measures that $\Pr\left\{\mathcal{L}_\theta\left(\tilde{\theta},\lambda\right) = \ell_{\min}\right\} = 1$. We therefore assume henceforth, without loss of generality, that the support of $\tilde{\theta}$ consists entirely of minimizers of $\mathcal{L}_\theta\left(\cdot,\lambda\right)$. Let $S \subseteq \Theta$ be this support set.

Define $G := \left\{\nabla_{\tilde{\lambda}}\mathcal{L}_\lambda\left(\theta',\lambda\right) : \theta' \in S\right\}$, and take $\bar{G}$ to be the closure of the convex hull of G. Since $\mathbb{E}_{\tilde{\theta}}\left[\nabla_{\tilde{\lambda}}\mathcal{L}_\lambda\left(\tilde{\theta},\lambda\right)\right] \in \bar{G} \subseteq \mathbb{R}^m$, we can write it as a convex combination of at most $m+1$ extreme points of $\bar{G}$, or equivalently of $m+1$ elements of G. Hence, we can take θ to be a discrete random variable that places nonzero mass on at most $m+1$ elements of S, and:

$$\mathbb{E}_\theta\left[\nabla_{\tilde{\lambda}}\mathcal{L}_\lambda\left(\theta,\lambda\right)\right] = \mathbb{E}_{\tilde{\theta}}\left[\nabla_{\tilde{\lambda}}\mathcal{L}_\lambda\left(\tilde{\theta},\lambda\right)\right].$$

Linearity in λ then implies that $\mathbb{E}_\theta\left[\mathcal{L}_\lambda\left(\theta,\cdot\right)\right]$ and $\mathbb{E}_{\tilde{\theta}}\left[\mathcal{L}_\lambda\left(\tilde{\theta},\cdot\right)\right]$ are the *same function* up to a constant, and therefore have the same maximizer(s). Correspondingly, θ is supported on S, which contains only minimizers of $\mathcal{L}_\theta\left(\cdot,\lambda\right)$ by construction. ■

Lemma 14 *If Θ is a compact Hausdorff space and the objective, constraint and proxy constraint functions $g_0, g_1, \dots, g_m, \tilde{g}_1, \dots, \tilde{g}_m$ are continuous, then the proxy-Lagrangian game (Equation 15) has a mixed Nash equilibrium pair (θ,λ) where θ is a random variable supported on at most $m+1$ elements of Θ, and λ is non-random.*

Proof Applying Theorem 13 directly would result in a support size of $m+2$, rather than the desired $m+1$, since Λ is $(m+1)$-dimensional. Instead, we define $\tilde{\Lambda} = \left\{\tilde{\lambda} \in \mathbb{R}_+^m : \left\|\tilde{\lambda}\right\|_1 \leq 1\right\}$ as the space containing the last m coordinates of Λ. Then we can rewrite the proxy-Lagrangian functions $\tilde{\mathcal{L}}_\theta, \tilde{\mathcal{L}}_\lambda : \Theta \times \tilde{\Lambda} \to \mathbb{R}$ as:

$$\begin{aligned}\tilde{\mathcal{L}}_\theta\left(\theta,\tilde{\lambda}\right) =& \left(1-\left\|\tilde{\lambda}\right\|_1\right)g_0\left(\theta\right) + \sum_{i=1}^m \tilde{\lambda}_i\tilde{g}_i\left(\theta\right)\\ \tilde{\mathcal{L}}_\lambda\left(\theta,\tilde{\lambda}\right) =& \sum_{i=1}^m \tilde{\lambda}_i g_i\left(\theta\right).\end{aligned}$$

These functions are linear in $\tilde{\lambda}$, which is a m-dimensional space, so the conditions of Theorem 13 apply, yielding the claimed result. ■

Proof [Proof of Lemma 5] The linear program contains not only the m explicit linearized functional constraints, but also, since $p \in \Delta^T$, the T nonnegativity constraints $p_t \geq 0$, and the sum-to-one constraint $\sum_{t=1}^{T} p_t = 1$.

Since p is T-dimensional, every vertex p^* of the feasible region must include T active constraints. Letting $m^* \leq m$ be the number of active linearized functional constraints, and accounting for the sum-to-one constraint, it follows that at least $T - m^* - 1$ nonnegativity constraints are active, implying that p^* contains at most $m^* + 1$ nonzero elements. ■

Appendix C. Proofs Of Convergence Rates

C.1. Non-Stochastic One-Player Convergence Rates

Theorem 15 ***(Mirror Descent)*** *Let $f_1, f_2, \ldots : \Theta \to \mathbb{R}$ be a sequence of convex functions that we wish to minimize on a compact convex set Θ. Suppose that the "distance generating function" $\Psi : \Theta \to \mathbb{R}_+$ is nonnegative and 1-strongly convex w.r.t. a norm $\|\cdot\|$ with dual norm $\|\cdot\|_*$.*

Define the step size $\eta = \sqrt{B_\Psi / T B_{\check{\nabla}}^2}$, where $B_\Psi \geq \max_{\theta\in\Theta} \Psi(\theta)$ is a uniform upper bound on Ψ, and $B_{\check{\nabla}} \geq \left\| \check{\nabla} f_t\left(\theta^{(t)}\right) \right\|_*$ is a uniform upper bound on the norms of the subgradients. Suppose that we perform T iterations of the following update, starting from $\theta^{(1)} = \operatorname{argmin}_{\theta\in\Theta} \Psi(\theta)$:

$$\begin{aligned} \tilde{\theta}^{(t+1)} =& \nabla\Psi^*\left(\nabla\Psi\left(\theta^{(t)}\right) - \eta\check{\nabla} f_t\left(\theta^{(t)}\right)\right) \\ \theta^{(t+1)} =& \operatorname*{argmin}_{\theta\in\Theta} D_\Psi\left(\theta \mid \tilde{\theta}^{(t+1)}\right), \end{aligned}$$

where $\check{\nabla} f_t(\theta) \in \partial f_t(\theta^{(t)})$ is a subgradient of f_t at θ, and $D_\Psi(\theta \mid \theta') := \Psi(\theta) - \Psi(\theta') - \langle \nabla\Psi(\theta'), \theta - \theta' \rangle$ is the Bregman divergence associated with Ψ. Then:

$$\frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} f_t(\theta^*) \leq 2B_{\check{\nabla}}\sqrt{\frac{B_\Psi}{T}},$$

where $\theta^* \in \Theta$ is an arbitrary reference vector.

Proof Mirror descent (Nemirovski and Yudin, 1983; Beck and Teboulle, 2003) dates back to 1983, but this particular statement is taken from Lemma 2 of Srebro et al. (2011). ■

Corollary 16 ***(Gradient Descent)*** *Let $f_1, f_2, \ldots : \Theta \to \mathbb{R}$ be a sequence of convex functions that we wish to minimize on a compact convex set Θ.*

Define the step size $\eta = B_\Theta / B_{\check{\nabla}}\sqrt{2T}$, where $B_\Theta \geq \max_{\theta\in\Theta} \|\theta\|_2$, and $B_{\check{\nabla}} \geq \left\| \check{\nabla} f_t\left(\theta^{(t)}\right) \right\|_2$ is a uniform upper bound on the norms of the subgradients. Suppose that we perform T iterations of the following update, starting from $\theta^{(1)} = \operatorname{argmin}_{\theta\in\Theta} \|\theta\|_2$:

$$\theta^{(t+1)} = \Pi_\Theta\left(\theta^{(t)} - \eta\check{\nabla} f_t\left(\theta^{(t)}\right)\right),$$

where $\check{\nabla} f_t(\theta) \in \partial f_t(\theta^{(t)})$ is a subgradient of f_t at θ, and Π_Θ projects its argument onto Θ w.r.t. the Euclidean norm. Then:

$$\frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^*\right) \le B_\Theta B_{\check{\nabla}} \sqrt{\frac{2}{T}},$$

where $\theta^* \in \Theta$ is an arbitrary reference vector.

Proof Follows from taking $\Psi(\theta) = \left\|\theta\right\|_2^2 / 2$ in Theorem 15. ■

Corollary 17 *Let $\mathcal{M} := \left\{ M \in \mathbb{R}^{\tilde{m}\times\tilde{m}} : \forall i \in [\tilde{m}] \,. M_{:,i} \in \Delta^{\tilde{m}} \right\}$ be the set of all left-stochastic $\tilde{m} \times \tilde{m}$ matrices, and let $f_1, f_2, \ldots : \mathcal{M} \to \mathbb{R}$ be a sequence of concave functions that we wish to maximize.*

Define the step size $\eta = \sqrt{\tilde{m}\ln\tilde{m} / T B_{\hat{\nabla}}^2}$, where $B_{\hat{\nabla}} \ge \left\| \hat{\nabla} f_t\left(M^{(t)}\right) \right\|_{\infty,2}$ is a uniform upper bound on the norms of the supergradients, and $\left\|\cdot\right\|_{\infty,2} := \sqrt{\sum_{i=1}^{\tilde{m}} \left\|M_{:,i}\right\|_\infty^2}$ is the $L_{\infty,2}$ matrix norm. Suppose that we perform T iterations of the following update starting from the matrix $M^{(1)}$ with all elements equal to $1/\tilde{m}$:

$$\begin{aligned}\tilde{M}^{(t+1)} =& M^{(t)} \odot . \exp\left(\eta \hat{\nabla} f_t\left(M^{(t)}\right)\right)\\ M_{:,i}^{(t+1)} =& \tilde{M}_{:,i}^{(t+1)} / \left\| \tilde{M}_{:,i}^{(t+1)} \right\|_1,\end{aligned}$$

where $-\hat{\nabla} f_t\left(M^{(t)}\right) \in \partial\left(-f_t(M^{(t)})\right)$, i.e. $\hat{\nabla} f_t\left(M^{(t)}\right)$ is a supergradient of f_t at $M^{(t)}$, and the multiplication and exponentiation in the first step are performed element-wise. Then:

$$\frac{1}{T}\sum_{t=1}^{T} f_t\left(M^*\right) - \frac{1}{T}\sum_{t=1}^{T} f_t\left(M^{(t)}\right) \le 2 B_{\hat{\nabla}} \sqrt{\frac{\tilde{m}\ln\tilde{m}}{T}},$$

where $M^* \in \mathcal{M}$ is an arbitrary reference matrix.

Proof Define $\Psi : \mathcal{M} \to \mathbb{R} := \tilde{m}\ln\tilde{m} + \sum_{i,j\in[\tilde{m}]} M_{i,j} \ln M_{i,j}$ as $\tilde{m}\ln\tilde{m}$ plus the negative Shannon entropy, applied to its (matrix) argument element-wise ($\tilde{m}\ln\tilde{m}$ is added to make Ψ nonnegative on $\mathcal{M}$). As in the vector setting, the resulting mirror descent update will be (element-wise) multiplicative.

The Bregman divergence satisfies:

$$\begin{aligned}D_\Psi\left(M | M'\right) =& \Psi\left(M\right) - \Psi\left(M'\right) - \left\langle \nabla\Psi\left(M'\right), M - M' \right\rangle\\ =& \left\|M'\right\|_{1,1} - \left\|M\right\|_{1,1} + \sum_{i=1}^{\tilde{m}} D_{KL}\left(M_{:,i} \| M'_{:,i}\right),\end{aligned} \tag{24}$$

where $\left\|M\right\|_{1,1} = \sum_{i=1}^{\tilde{m}} \left\|M_{:,i}\right\|_1$ is the $L_{1,1}$ matrix norm. This incidentally shows that one projects onto $\mathcal{M}$ w.r.t. D_Ψ by projecting each column w.r.t. the KL divergence, i.e. by normalizing the columns.

By Pinsker's inequality (applied to each column of an $M \in \mathcal{M}$):

$$\left\|M - M'\right\|_{1,2}^2 \leq 2 \sum_{i=1}^{\tilde{m}} D_{KL}\left(M_{:,i} \| M'_{:,i}\right),$$

where $\left\|M\right\|_{1,2} = \sqrt{\sum_{i=1}^{\tilde{m}} \left\|M_{:,i}\right\|_1^2}$ is the $L_{1,2}$ matrix norm. Substituting this into Equation 24, and using the fact that $\left\|M\right\|_{1,1} = \tilde{m}$ for all $M \in \mathcal{M}$, we have that for all $M, M' \in \mathcal{M}$:

$$D_{\Psi}\left(M | M'\right) \geq \frac{1}{2} \left\|M - M'\right\|_{1,2}^2,$$

which shows that Ψ is 1-strongly convex w.r.t. the $L_{1,2}$ matrix norm. The dual norm of the $L_{1,2}$ matrix norm is the $L_{\infty,2}$ norm, which is the last piece needed to apply Theorem 15, yielding the claimed result. ■

Lemma 18 *Let $\Lambda := \Delta^{\tilde{m}}$ be the $\tilde{m}$-dimensional simplex, define*

$$\mathcal{M} := \left\{M \in \mathbb{R}^{\tilde{m} \times \tilde{m}} : \forall i \in [\tilde{m}] . M_{:,i} \in \Delta^{\tilde{m}}\right\}$$

as the set of all left-stochastic $\tilde{m} \times \tilde{m}$ matrices, and take $f_1, f_2, \ldots : \Lambda \rightarrow \mathbb{R}$ to be a sequence of concave functions that we wish to maximize.

Define the step size $\eta = \sqrt{\tilde{m} \ln \tilde{m} / T B_{\hat{\nabla}}^2}$, where $B_{\hat{\nabla}} \geq \left\|\hat{\nabla} f_t\left(\lambda^{(t)}\right)\right\|_{\infty}$ is an uniform upper bound on the ∞-norms of the supergradients. Suppose that we perform T iterations of the following update, starting from the matrix $M^{(1)}$ with all elements equal to $1/\tilde{m}$:

$$\begin{aligned}
&\lambda^{(t)} \textit{ is any stationary distribution of } M^{(t)} \\
A^{(t)} =& \left(\hat{\nabla} f_t\left(\lambda^{(t)}\right)\right)\left(\lambda^{(t)}\right)^T \\
\tilde{M}^{(t+1)} =& M^{(t)} \odot . \exp\left(\eta A^{(t)}\right) \\
M_{:,i}^{(t+1)} =& \tilde{M}_{:,i}^{(t+1)} / \left\|\tilde{M}_{:,i}^{(t+1)}\right\|_1,
\end{aligned}$$

where a stationary distribution of M (i.e. a $\lambda \in \Lambda$ such that $M\lambda = \lambda$) always exists because M is left-stochastic, $-\hat{\nabla} f_t\left(\lambda^{(t)}\right) \in \partial\left(-f_t(\lambda^{(t)})\right)$, i.e. $\hat{\nabla} f_t\left(\lambda^{(t)}\right)$ is a supergradient of f_t at $\lambda^{(t)}$, and the multiplication and exponentiation of the third step are performed element-wise. Then:

$$\frac{1}{T} \sum_{t=1}^{T} f_t\left(M^* \lambda^{(t)}\right) - \frac{1}{T} \sum_{t=1}^{T} f_t\left(\lambda^{(t)}\right) \leq 2 B_{\hat{\nabla}} \sqrt{\frac{\tilde{m} \ln \tilde{m}}{T}},$$

where $M^* \in \mathcal{M}$ is an arbitrary left-stochastic reference matrix.

Proof This algorithm is an instance of that contained in Figure 1 of Gordon et al. (2008).

Define $\tilde{f}_t\left(M\right) := f_t\left(M^{(t)} \lambda^{(t)}\right)$. Observe that since $\hat{\nabla} f_t\left(\lambda^{(t)}\right)$ is a supergradient of f_t at $\lambda^{(t)}$, and $M^{(t)} \lambda^{(t)} = \lambda^{(t)}$:

$$\begin{aligned}
f_t\left(\tilde{M} \lambda^{(t)}\right) \leq& f_t\left(M^{(t)} \lambda^{(t)}\right) + \left\langle \hat{\nabla} f_t\left(\lambda^{(t)}\right), \tilde{M} \lambda^{(t)} - M^{(t)} \lambda^{(t)} \right\rangle \\
\leq& f_t\left(M^{(t)} \lambda^{(t)}\right) + A^{(t)} \cdot \left(\tilde{M} - M^{(t)}\right),
\end{aligned}$$

where the matrix product on the last line is performed element-wise. This shows that $A^{(t)}$ is a supergradient of $\tilde{f}_t$ at $M^{(t)}$, from which we conclude that the final two steps of the update are performing the algorithm of Corollary 17, so:

$$\frac{1}{T}\sum_{t=1}^{T}\tilde{f}_t\left(M^*\right)-\frac{1}{T}\sum_{t=1}^{T}\tilde{f}_t\left(M^{(t)}\right)\leq 2B_{\hat{\nabla}}\sqrt{\frac{\tilde{m}\ln\tilde{m}}{T}},$$

where the $B_{\hat{\nabla}}$ of Corollary 17 is a uniform upper bound on the $L_{\infty,2}$ matrix norms of the $A^{(t)}$s. However, by the definition of $A^{(t)}$ and the fact that $\lambda^{(t)} \in \Delta^{\tilde{m}}$, we can instead take $B_{\hat{\nabla}}$ to be a uniform upper bound on $\left\|\hat{\nabla}^{(t)}\right\|_{\infty}$. Substituting the definition of $\tilde{f}_t$ and again using the fact that $M^{(t)}\lambda^{(t)} = \lambda^{(t)}$ then yields the claimed result. ■

C.2. Stochastic One-Player Convergence Rates

Theorem 19 ***(Stochastic Mirror Descent)*** *Let Ψ, $\|\cdot\|$, D_Ψ and B_Ψ be as in Theorem 15, and let $f_1, f_2, \ldots : \Theta \to \mathbb{R}$ be a sequence of convex functions that we wish to minimize on a compact convex set Θ.*

Define the step size $\eta = \sqrt{B_\Psi / T B_{\check{\Delta}}^2}$, where $B_{\check{\Delta}} \geq \left\|\check{\Delta}^{(t)}\right\|_*$ is a uniform upper bound on the norms of the stochastic subgradients. Suppose that we perform T iterations of the following stochastic *update, starting from $\theta^{(1)} = \operatorname{argmin}_{\theta\in\Theta}\Psi\left(\theta\right)$:*

$$\begin{aligned}\tilde{\theta}^{(t+1)} &= \nabla\Psi^*\left(\nabla\Psi\left(\theta^{(t)}\right)-\eta\check{\Delta}^{(t)}\right)\\ \theta^{(t+1)} &= \operatorname*{argmin}_{\theta\in\Theta} D_\Psi\left(\theta \mid \tilde{\theta}^{(t+1)}\right),\end{aligned}$$

where $\mathbb{E}\left[\check{\Delta}^{(t)} \mid \theta^{(t)}\right] \in \partial f_t(\theta^{(t)})$, i.e. $\check{\Delta}^{(t)}$ is a stochastic subgradient of f_t at $\theta^{(t)}$. Then, with probability $1-\delta$ over the draws of the stochastic subgradients:

$$\frac{1}{T}\sum_{t=1}^{T}f_t\left(\theta^{(t)}\right)-\frac{1}{T}\sum_{t=1}^{T}f_t\left(\theta^*\right)\leq 2B_{\check{\nabla}}\sqrt{\frac{2B_\Psi\left(1+16\ln\frac{1}{\delta}\right)}{T}},$$

where $\theta^* \in \Theta$ is an arbitrary reference vector.

Proof This is nothing more than the usual transformation of a uniform regret guarantee into a stochastic one via the Hoeffding-Azuma inequality—we include a proof for completeness.

Define the sequence:

$$\tilde{f}_t\left(\theta\right) = f_t\left(\theta^{(t)}\right) + \left\langle \check{\Delta}^{(t)}, \theta - \theta^{(t)} \right\rangle.$$

Then applying non-stochastic mirror descent to the sequence $\tilde{f}_t$ will result in exactly the same sequence of iterates $\theta^{(t)}$ as applying stochastic mirror descent (above) to f_t. Hence, by Theorem 15 and the definition of $\tilde{f}_t$ (notice that we can take $B_{\check{\nabla}} = B_{\check{\Delta}}$):

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T} \tilde{f}_t\left(\theta^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} \tilde{f}_t\left(\theta^{*}\right) \leq & 2B_{\check{\nabla}}\sqrt{\frac{B_\Psi}{T}} \\ \frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^{*}\right) \leq & 2B_{\check{\nabla}}\sqrt{\frac{B_\Psi}{T}} + \frac{1}{T}\sum_{t=1}^{T}\left(\tilde{f}_t\left(\theta^{*}\right) - f_t\left(\theta^{*}\right)\right) \\ \leq & 2B_{\check{\nabla}}\sqrt{\frac{B_\Psi}{T}} + \frac{1}{T}\sum_{t=1}^{T}\left\langle \check{\Delta}^{(t)} - \check{\nabla} f_t\left(\theta^{(t)}\right), \theta^{*} - \theta^{(t)}\right\rangle, \end{aligned} \tag{25}$$

where the last step follows from the convexity of the f_ts. Consider the second term on the RHS. Observe that, since the $\check{\Delta}^{(t)}$s are stochastic subgradients, each of the terms in the sum is zero in expectation (conditioned on the past), and the partial sums therefore form a martingale. Furthermore, by Hölder's inequality:

$$\left\langle \check{\Delta}^{(t)} - \check{\nabla} f_t\left(\theta^{(t)}\right), \theta^{*} - \theta^{(t)}\right\rangle \leq \left\| \check{\Delta}^{(t)} - \check{\nabla} f_t\left(\theta^{(t)}\right)\right\|_{*} \left\| \theta^{*} - \theta^{(t)} \right\| \leq 4B_{\check{\Delta}}\sqrt{2B_\Psi},$$

where the last line holds because $\left\|\theta^{*} - \theta^{(t)}\right\| \leq \left\|\theta^{*} - \theta^{(1)}\right\| + \left\|\theta^{(t)} - \theta^{(1)}\right\| \leq 2\sup_{\theta\in\Theta}\sqrt{2D_\Psi\left(\theta \mid \theta^{(1)}\right)} \leq 2\sqrt{2B_\Psi}$, using the fact that D_Ψ is 1-strongly convex w.r.t. $\|\cdot\|$, and the definition of $\theta^{(1)}$. Hence, by the Hoeffding-Azuma inequality:

$$\Pr\left\{\frac{1}{T}\sum_{t=1}^{T}\left\langle \check{\Delta}^{(t)} - \check{\nabla} f_t\left(\theta^{(t)}\right), \theta^{*} - \theta^{(t)}\right\rangle \geq \epsilon\right\} \leq \exp\left(-\frac{T\epsilon^2}{64 B_\Psi B_{\check{\Delta}}^2}\right).$$

Equivalently:

$$\Pr\left\{\frac{1}{T}\sum_{t=1}^{T}\left\langle \check{\Delta}^{(t)} - \check{\nabla} f_t\left(\theta^{(t)}\right), \theta^{*} - \theta^{(t)}\right\rangle \geq 8B_{\check{\Delta}}\sqrt{\frac{B_\Psi \ln\frac{1}{\delta}}{T}}\right\} \leq \delta.$$

Substituting this into Equation 25, and applying the inequality $\sqrt{a} + \sqrt{b} \leq \sqrt{2a + 2b}$, yields the claimed result. ■

Corollary 20 ***(Stochastic Gradient Descent)*** *Let $f_1, f_2, \ldots : \Theta \to \mathbb{R}$ be a sequence of convex functions that we wish to minimize on a compact convex set Θ.*

Define the step size $\eta = B_\Theta / B_{\check{\Delta}}\sqrt{2T}$, where $B_\Theta \geq \max_{\theta\in\Theta}\|\theta\|_2$, and $B_{\check{\Delta}} \geq \left\|\check{\Delta}^{(t)}\right\|_2$ is a uniform upper bound on the norms of the stochastic subgradients. Suppose that we perform T iterations of the following stochastic *update, starting from $\theta^{(1)} = \operatorname{argmin}_{\theta\in\Theta}\|\theta\|_2$:*

$$\theta^{(t+1)} = \Pi_\Theta\left(\theta^{(t)} - \eta\check{\Delta}^{(t)}\right),$$

where $\mathbb{E}\left[\check{\Delta}^{(t)} \mid \theta^{(t)}\right] \in \partial f_t(\theta^{(t)})$, i.e. $\check{\Delta}^{(t)}$ is a stochastic subgradient of f_t at $\theta^{(t)}$, and Π_Θ projects its argument onto Θ w.r.t. the Euclidean norm. Then, with probability $1 - \delta$ over the draws of the *stochastic subgradients:*

$$\frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} f_t\left(\theta^{*}\right) \leq 2B_{\Theta}B_{\check{\nabla}}\sqrt{\frac{1+16\ln\frac{1}{\delta}}{T}},$$

where $\theta^{*} \in \Theta$ is an arbitrary reference vector.

Proof Follows from taking $\Psi\left(\theta\right) = \left\|\theta\right\|_2^2/2$ in Theorem 19. ■

Corollary 21 *Let $\mathcal{M} := \left\{M \in \mathbb{R}^{\tilde{m}\times\tilde{m}} : \forall i \in [\tilde{m}]\,.M_{:,i} \in \Delta^{\tilde{m}}\right\}$ be the set of all left-stochastic $\tilde{m}\times\tilde{m}$ matrices, and let $f_1, f_2, \ldots : \mathcal{M} \rightarrow \mathbb{R}$ be a sequence of concave functions that we wish to maximize.*

Define the step size $\eta = \sqrt{\tilde{m}\ln\tilde{m}/TB_{\hat{\Delta}}^2}$, where $B_{\hat{\Delta}} \geq \left\|\hat{\Delta}^{(t)}\right\|_{\infty,2}$ is a uniform upper bound on the norms of the stochastic supergradients, and $\left\|\cdot\right\|_{\infty,2} := \sqrt{\sum_{i=1}^{\tilde{m}}\left\|M_{:,i}\right\|_{\infty}^2}$ is the $L_{\infty,2}$ matrix norm. Suppose that we perform T iterations of the following stochastic update starting from the matrix $M^{(1)}$ with all elements equal to $1/\tilde{m}$:

$$\begin{aligned}\tilde{M}^{(t+1)} =& M^{(t)} \odot .\exp\left(\eta\hat{\Delta}^{(t)}\right)\\ M_{:,i}^{(t+1)} =& \tilde{M}_{:,i}^{(t+1)}/\left\|\tilde{M}_{:,i}^{(t+1)}\right\|_1,\end{aligned}$$

where $\mathbb{E}\left[-\hat{\Delta}^{(t)} \mid M^{(t)}\right] \in \partial\left(-f_t(M^{(t)})\right)$, i.e. $\hat{\Delta}^{(t)}$ is a stochastic supergradient of f_t at $M^{(t)}$, and the multiplication and exponentiation in the first step are performed element-wise. Then with probability $1-\delta$ over the draws of the stochastic supergradients:

$$\frac{1}{T}\sum_{t=1}^{T} f_t\left(M^{*}\right) - \frac{1}{T}\sum_{t=1}^{T} f_t\left(M^{(t)}\right) \leq 2B_{\hat{\Delta}}\sqrt{\frac{2\left(\tilde{m}\ln\tilde{m}\right)\left(1+16\ln\frac{1}{\delta}\right)}{T}},$$

where $M^{*} \in \mathcal{M}$ is an arbitrary reference matrix.

Proof The same reasoning as was used to prove Corollary 17 from Theorem 15 applies here (but starting from Theorem 19). ■

Lemma 22 *Let $\Lambda := \Delta^{\tilde{m}}$ be the $\tilde{m}$-dimensional simplex, define*

$$\mathcal{M} := \left\{M \in \mathbb{R}^{\tilde{m}\times\tilde{m}} : \forall i \in [\tilde{m}]\,.M_{:,i} \in \Delta^{\tilde{m}}\right\}$$

as the set of all left-stochastic $\tilde{m}\times\tilde{m}$ matrices, and take $f_1, f_2, \ldots : \Lambda \rightarrow \mathbb{R}$ to be a sequence of concave functions that we wish to maximize.

Define the step size $\eta = \sqrt{\tilde{m}\ln\tilde{m}/TB_{\hat{\Delta}}^2}$, where $B_{\hat{\Delta}} \geq \left\|\hat{\Delta}^{(t)}\right\|_{\infty}$ is a uniform upper bound on the ∞-norms of the stochastic supergradients. Suppose that we perform T iterations of the following *update, starting from the matrix $M^{(1)}$ with all elements equal to $1/\tilde{m}$:*

$$\begin{aligned}
&\lambda^{(t)} \text{ is any stationary distribution of } M^{(t)} \\
A^{(t)} =& \hat{\Delta}^{(t)} \left(\lambda^{(t)}\right)^T \\
\tilde{M}^{(t+1)} =& M^{(t)} \odot . \exp\left(\eta A^{(t)}\right) \\
M^{(t+1)}_{:,i} =& \tilde{M}^{(t+1)}_{:,i} / \left\| \tilde{M}^{(t+1)}_{:,i} \right\|_1 ,
\end{aligned}$$

where a stationary distribution of M (i.e. a $\lambda \in \Lambda$ such that $M\lambda = \lambda$) always exists because M is left-stochastic, $\mathbb{E}\left[-\hat{\Delta}^{(t)} \mid \lambda^{(t)}\right] \in \partial\left(-f_t(\lambda^{(t)})\right)$, i.e. $\hat{\Delta}^{(t)}$ is a stochastic supergradient of f_t at $\lambda^{(t)}$, and the multiplication and exponentiation of the third step are performed element-wise. Then with probability $1-\delta$ over the draws of the stochastic supergradients:

$$\frac{1}{T}\sum_{t=1}^{T} f_t\left(M^* \lambda^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} f_t\left(\lambda^{(t)}\right) \le 2B_{\hat{\Delta}} \sqrt{\frac{2\left(\tilde{m}\ln\tilde{m}\right)\left(1+16\ln\frac{1}{\delta}\right)}{T}},$$

where $M^* \in \mathcal{M}$ is an arbitrary left-stochastic reference matrix.

Proof The same reasoning as was used to prove Lemma 18 from Corollary 17 applies here (but starting from Corollary 21). ■

C.3. Two-Player Convergence Rates

Proof [Proof of Lemma 3] Applying Corollary 16 to the optimization over λ gives:

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^{(t)}, \lambda^*\right) - \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^{(t)}, \lambda^{(t)}\right) \le B_\Lambda B_\Delta \sqrt{\frac{2}{T}}.$$

By the definition of $\mathcal{O}_\rho$ (Definition 1):

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^{(t)}, \lambda^*\right) - \inf_{\theta^* \in \Theta} \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^*, \lambda^{(t)}\right) \le \rho + B_\Lambda B_\Delta \sqrt{\frac{2}{T}}.$$

Using the linearity of $\mathcal{L}$ in λ, the fact that $B_\Lambda = R$, and the definitions of $\bar{\theta}$ and $\bar{\lambda}$, yields the claimed result. ■

Lemma 23 ***(Algorithm 5)*** *Suppose that Θ is a compact convex set, Λ and R are as in Theorem 2, and that the objective and constraint functions $g_0, g_1, \dots, g_m$ are convex. Define the three upper bounds $B_\Theta \ge \max_{\theta \in \Theta} \|\theta\|_2$, $B_{\check{\Delta}} \ge \max_{t \in [T]} \left\| \check{\Delta}^{(t)}_\theta \right\|_2$, and $B_\Delta \ge \max_{t \in [T]} \left\| \Delta^{(t)}_\lambda \right\|_2$.*

If we run Algorithm 5 with the step sizes $\eta_\theta := B_\Theta / B_{\check{\Delta}} \sqrt{2T}$ and $\eta_\lambda := R / B_\Delta \sqrt{2T}$, then the result satisfies the conditions of Theorem 2 for:

$$\epsilon = 2\left(B_\Theta B_{\check{\Delta}} + R B_\Delta\right) \sqrt{\frac{1+16\ln\frac{2}{\delta}}{T}},$$

with probability $1-\delta$ over the draws of the stochastic (sub)gradients.

Algorithm 5 Optimizes the Lagrangian formulation (Equation 3) in the convex setting. The parameter R is the radius of the Lagrange multiplier space $\Lambda := \left\{ \lambda \in \mathbb{R}_+^m : \left\| \lambda \right\|_1 \leq R \right\}$, and the functions Π_Θ and Π_Λ project their arguments onto Θ and Λ (respectively) w.r.t. the Euclidean norm.

```
StochasticLagrangian ($R \in \mathbb{R}_+, \mathcal{L} : \Theta \times \Lambda \rightarrow \mathbb{R}, T \in \mathbb{N}, \eta_\theta, \eta_\lambda \in \mathbb{R}_+$):
    Initialize $\theta^{(1)} = 0$, $\lambda^{(1)} = 0$                                   // Assumes $0 \in \Theta$
    For $t \in [T]$:
        Let $\check{\Delta}_\theta^{(t)}$ be a stochastic subgradient of $\mathcal{L}\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. $\theta$
        Let $\Delta_\lambda^{(t)}$ be a stochastic gradient of $\mathcal{L}\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. $\lambda$
        Update $\theta^{(t+1)} = \Pi_\Theta\left(\theta^{(t)} - \eta_\theta \check{\Delta}_\theta^{(t)}\right)$          // Projected SGD updates ...
        Update $\lambda^{(t+1)} = \Pi_\Lambda\left(\lambda^{(t)} + \eta_\lambda \Delta_\lambda^{(t)}\right)$          //     ...
    Return $\theta^{(1)}, \dots, \theta^{(T)}$ and $\lambda^{(1)}, \dots, \lambda^{(T)}$
```

Proof Applying Corollary 20 to the two optimizations (over θ and λ) gives that with probability $1 - 2\delta'$ over the draws of the stochastic (sub)gradients:

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^{(t)}, \lambda^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^{*}, \lambda^{(t)}\right) \leq & 2 B_\Theta B_{\check{\Delta}} \sqrt{\frac{1 + 16 \ln \frac{1}{\delta'}}{T}} \\
\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^{(t)}, \lambda^{*}\right) - \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}\left(\theta^{(t)}, \lambda^{(t)}\right) \leq & 2 B_\Lambda B_{\Delta} \sqrt{\frac{1 + 16 \ln \frac{1}{\delta'}}{T}}.
\end{aligned}$$

Adding these inequalities, taking $\delta = 2\delta'$, using the linearity of $\mathcal{L}$ in λ, the fact that $B_\Lambda = R$, and the definitions of $\bar{\theta}$ and $\bar{\lambda}$, yields the claimed result. ■

Proof [Proof of Lemma 7] Applying Lemma 18 to the optimization over λ (with $\tilde{m} := m + 1$) gives:

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_\lambda\left(\theta^{(t)}, M^* \lambda^{(t)}\right) - \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_\lambda\left(\theta^{(t)}, \lambda^{(t)}\right) \leq 2 B_\Delta \sqrt{\frac{(m+1)\ln(m+1)}{T}}.$$

By the definition of $\mathcal{O}_\rho$ (Definition 1):

$$\frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_\theta\left(\theta^{(t)}, \lambda^{(t)}\right) - \inf_{\theta^* \in \Theta} \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}_\theta\left(\theta^{*}, \lambda^{(t)}\right) \leq \rho.$$

Using the definitions of $\bar{\theta}$ and $\bar{\lambda}$ yields the claimed result. ■

Proof [Proof of Lemma 8] Applying Corollary 20 to the optimization over θ, and Lemma 22 to that over λ (with $\tilde{m} := m + 1$), gives that with probability $1 - 2\delta'$ over the draws of the stochastic (sub)gradients:

$$\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{\theta}\left(\theta^{(t)},\lambda^{(t)}\right)-\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{\theta}\left(\theta^{*},\lambda^{(t)}\right)\leq 2B_{\Theta}B_{\check{\Delta}}\sqrt{\frac{1+16\ln\frac{1}{\delta'}}{T}}$$

$$\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{\lambda}\left(\theta^{(t)},M^{*}\lambda^{(t)}\right)-\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{\lambda}\left(\theta^{(t)},\lambda^{(t)}\right)\leq 2B_{\Delta}\sqrt{\frac{2\left(m+1\right)\ln\left(m+1\right)\left(1+16\ln\frac{1}{\delta'}\right)}{T}}.$$

Taking $\delta=2\delta'$, and using the definitions of $\bar{\theta}$ and $\bar{\lambda}$, yields the claimed result. ■

References

A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach. A reductions approach to fair classification. In *ICML*, 2018.

S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk. *Statistical Inference Under Order Restrictions; The Theory And Application Of Isotonic Regression*. Wiley, New York, USA, 1972.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, May 2003.

K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. *UAI*, 2009.

A. Blum and Y. Mansour. From external to internal regret. *JMLR*, 8:1307–1324, 2007.

D. G. Bocian, K. S. Ernst, and W. Li. Race, ethnicity and subprime home loan pricing. *Journal of Economics and Business*, 60(1-2):110–124, 2008.

H. F. Bohnenblust, S. Karlin, and L. S. Shapley. Games with continuous, convex pay-off. *Contributions to the Theory of Games*, 1(24):181–192, 1950.

M. Bonakdarpour, S. Chatterjee, R. F. Barber, and J. D. Lafferty. Prediction rule reshaping. In *ICML*, 2018.

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pages 77–91, 2018.

K. Canini, A. Cotter, M. R. Gupta, M. Milani Fard, and J. Pfeifer. Fast and flexible monotonic functions with ensembles of lattices. In *NIPS*, pages 2919–2927, 2016.

R. S. Chen, B. Lucier, Y. Singer, and V. Syrgkanis. Robust optimization for non-convex objectives. In *NIPS*, 2017.

X. Chen and X. Deng. Settling the complexity of two-player Nash equilibrium. In *FOCS'06*, pages 261–272. IEEE, 2006.

Y. Chen and R. J. Samworth. Generalized additive and index models with shape constraints. *Journal Royal Statistical Society B*, 2016.

D. Chetverikov, A. Santos, and A. M. Shaikh. The econometrics of shape restrictions. *Annual Review of Economics*, 2018.

P. Christiano, J. A. Kelner, A. Madry, C. A. Spielman, and S. Teng. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In *STOC*, pages 273–282, 2011.

Q. Cormier, M. Milani Fard, K. Canini, and M. R. Gupta. Launch and iterate: Reducing prediction churn. *NIPS*, 2016.

A. Cotter, M. R. Gupta, and J. Pfeifer. A Light Touch for heavily constrained SGD. In *COLT*, pages 729–771, 2016.

A. Cotter, M. Gupta, H. Jiang, N. Srebro, K. Sridharan, S. Wang, B. Woodworth, and S. You. Training well-generalizing classifiers for fairness metrics and other data-dependent constraints. In *ICML*, 2019a.

A. Cotter, M. R. Gupta, H. Jiang, E. Louidor, J. Muller, T. Narayan, S. Wang, and T. Zhu. Shape constraints for set functions. In *ICML*, 2019b.

A. Cotter, H. Jiang, and K. Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*, pages 300–332, 2019c.

M. Davenport, R. G. Baraniuk, and C. D. Scott. Tuning support vector machines for minimax and Neyman-Pearson classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

M. Donini, L. Oneto, S. Ben-David, J. Shawe-Taylor, and M. Pontil. Empirical risk minimization under fairness constraints. *NeurIPS*, 2018.

E. Eban, M. Schain, A. Mackey, A. Gordon, R. A. Saurous, and G. Elidan. Scalable learning of non-decomposable objectives. *AIStats*, 2017.

B. Fish, J. Kun, and A. D. Lelkes. A confidence-based approach for balancing fairness and accuracy. *SIAM ICDM*, 2016.

D. Garber and E. Hazan. Playing non-linear games with linear oracles. In *FOCS*, pages 420–428. IEEE Computer Society, 2013.

G. Gasso, A. Pappaionannou, M. Spivak, and L. Bottou. Batch and online learning algorithms for nonconvex Neyman-Pearson classification. *ACM Transactions on Intelligent Systems and Technology*, 2011.

I. L. Glicksberg. A further generalization of the Kakutani fixed point theorem with application to Nash equilibrium points. *Proceedings American Mathematical Society*, 3:170–174, 1952.

G. Goh, A. Cotter, M. R. Gupta, and M. P. Friedlander. Satisfying real-world goals with dataset constraints. In *NIPS*, pages 2415–2423, 2016.

G. J. Gordon, A. Greenwald, and C. Marks. No-regret learning in convex games. In *ICML*, pages 360–367, 2008.

P. Groeneboom and G. Jongbloed. *Nonparametric Estimation Under Shape Constraints*. Cambridge Press, New York, USA, 2014.

M. R. Gupta, A. Cotter, J. Pfeifer, K. Voevodski, K. Canini, A. Mangylov, W. Moczydlowski, and A. van Esbroeck. Monotonic calibrated interpolated look-up tables. *JMLR*, 17(109):1–47, 2016.

M. R. Gupta, D. Bahri, A. Cotter, and K. Canini. Diminishing returns shape constraints for interpretability and regularization. *NeurIPS*, 2018.

M. R. Gupta, A. Cotter, M. Milani Fard, and S. Wang. Proxy fairness. In *arXiv:1806.11212*, 2019.

M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. *NIPS*, 2016.

E. Hazan and S. Kale. Projection-free online learning. In *ICML*, 2012.

H. Heidari, C. Ferrari, K. Gummadi, and A. Krause. Fairness behind a veil of ignorance: A welfare analysis for automated decision making. In *NeurIPS*, pages 1265–1276, 2018.

M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pages 315–323, 2013.

T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Fairness-aware classifier with prejudice remover regularizer. *Machine Learning and Knowledge Discovery in Databases*, pages 35–50, 2012.

M. Kearns, S. Neel, A. Roth, and Z. S. Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *ICML*, 2018.

B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model. *Annals of Applied Statistics*, 2015.

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

Z. Long, Y. Lu, X. Ma, and B. Dong. PDE-Net: Learning PDEs from Data. In *ICML*, 2018.

R. Luss and S. Rosset. Bounded isotonic regression. *Electronic Journal of Statistics*, 11(2):4488–4514, 2017.

M. Mahdavi, T. Yang, R. Jin, S. Zhu, and J. Yi. Stochastic gradient descent with only one projection. In *NIPS*, pages 494–502, 2012.

G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning with expectation regularization. In *ICML*, 2007.

G. S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *JMLR*, 11, 2010.

H. Narasimhan. Learning with complex loss functions and constraints. In *AIStats*, 2018.

H. Narasimhan, A. Cotter, and M. R. Gupta. Optimizing generalized rate metrics through game equilibrium. In *NeurIPS*, 2019a.

H. Narasimhan, A. Cotter, and M. R. Gupta. On making stochastic classifiers deterministic. In *NeurIPS*, 2019b.

H. Narasimhan, A. Cotter, M. R. Gupta, and S. Wang. Pairwise fairness for ranking and regression. In *AAAI*, 2020.

A. Nemirovski and D. Yudin. *Problem Complexity And Method Efficiency In Optimization*. John Wiley & Sons Ltd, 1983.

T Parthasarathy. Equilibria of continuous two-person games. *Pacific Journal of Mathematics*, 57(1): 265–270, 1975.

N. Pya and S. N. Wood. Shape constrained additive models. *Statistics and Computing*, 2015.

A. Rakhlin and K. Sridharan. Optimization, learning, and games with predictable sequences. In *NIPS*, pages 3066–3074, 2013.

A. Rakhlin, K. Sridharan, and A. Tewari. Online learning: beyond regret. In *COLT*, pages 559–594, 2011.

C. D. Scott and R. D. Nowak. A Neyman-Pearson approach to statistical learning. *IEEE Transactions on Information Theory*, 2005.

N. Srebro, K. Sridharan, and A. Tewari. On the universality of online mirror descent. In *NIPS*, 2011.

R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. *AAAI*, 2017.

J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

B. E. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro. Learning non-discriminatory predictors. In *COLT*, pages 1920–1953, 2017.

T. Yang, Q. Lin, and L. Zhang. A richer theory of convex constrained optimization with reduced projections and improved rates. In *ICML*, pages 3901–3910, 2017.

S. You, K. Canini, D. Ding, J. Pfeifer, and M. R. Gupta. Deep lattice networks for learning partial monotonic functions. *NIPS*, 2017.

M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi. Fairness constraints: A mechanism for fair classification. In *ICML Workshop on Fairness, Accountability, and Transparency in Machine Learning*, 2015.

M. B. Zafar, I. Valera, M. G. Rogriguez, and K. P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *AIStats*, pages 962–970, 2017.