Approximate Heavily-Constrained Learning with Lagrange Multiplier Models

Harikrishna Narasimhan, Andrew Cotter, Yichen Zhou
Google Research, USA
{hnarasimhan,acotter,yichenzhou}@google.com

Serena Wang, Wenshuo Guo
University of California, Berkeley
{serenalwang,wsguo}@berkeley.edu

Abstract

In machine learning applications such as ranking fairness or fairness over intersectional groups, one often encounters optimization problems with extremely large numbers of constraints. In particular, with ranking fairness tasks, there may even be a variable number of constraints, e.g. one for each query in the training set. In these cases, the standard approach of optimizing a Lagrangian while maintaining one Lagrange multiplier per constraint may no longer be practical. Our proposal is to associate a feature vector with each constraint, and to learn a "multiplier model" that maps each such vector to the corresponding Lagrange multiplier. We prove optimality, approximate feasibility and generalization guarantees under assumptions on the flexibility of the multiplier model, and empirically demonstrate that our method is effective on real-world case studies.

1 Introduction

Constrained optimization has proven to be useful for a variety of machine learning applications, including churn reduction, Neyman-Pearson classification, and the imposition of statistical fairness constraints [e.g. 1, 2]. In such problems, there are generally only a handful of constraints: for example, in a fairness problem, there will typically be only one constraint per protected group. As a result, while optimizing such a constrained ML problem *is* more difficult than optimizing an unconstrained problem, the difference is usually relatively small.

In some cases, however, it may be desirable to include an extremely large number of constraints. For example, in a fairness problem in which the data are partitioned into protected groups in multiple distinct ways (e.g. race, gender, age buckets, etc.), enforcing a constraint independently for each of the individual groups may not satisfy the constraint for each *intersection* of those groups [3]. Therefore, it may be necessary to impose a separate constraint for each such intersection. As one tries to impose statistical fairness requirements along an increasing number of such dimensions, the total number of constraints can quickly get out of hand.

Despite the difficulties with such a setting, the situation in such an intersectional statistical fairness problem isn't as bad as it *could* be, since there are still a known, finite number of constraints. In other settings, even this might not be the case. In many ranking problems, for example, the workflow is that a "query" is provided, with each query being associated with some set of "documents", which are then handed off to a model to be ranked. Here, we're mainly interested in the problem of imposing fairness constraints on a *per-query* basis. When statistical fairness constraints (such as ranking analogues to demographic parity or equal opportunity [4], adapted to the ranking setting as in Narasimhan et al.

[5]) are imposed on a ranking problem, they are generally enforced *on average* across all queries [e.g. 6, 5]. As a result, any *particular* query might be "unfair" (w.r.t. whatever fairness metric is being used), as long as this unfairness averages out. Existing work on per-query fairness metrics is, so far as we're aware, restricted to post-processing approaches [7–9]. Our goal is to formulate such a problem as an explicit constrained optimization problem, with one (or potentially more) fairness constraints per query. One of the main difficulties of this setting is that, while one could think of there being one constraint per query in the *training set*, it would be more accurate to imagine that there are potentially an infinite number of constraints, one for each *possible* query, and those corresponding to the training set are merely those that we happen to be able to easily observe. This raises the interesting question of how well such constraints *generalize*.

A third example we consider is a variant of *robust optimization*, specifically in a statistical fairness setting in which protected group information isn't available, but we *do* have access to correlated "noisy" features. In one canonical example, "zip codes" serve as a noisy proxy for "race". Wang et al. [10] proposed using a robust-optimization-like approach to this problem, in which the correlations between the "true" and "noisy" groups were assumed to be known (or estimated from a side-dataset), and the fairness constraints were required to hold for the *worst* true-group labeling that was consistent with both the known proxy-group labeling and these known correlations. Applied to their problem, our approach differs only formally, in that instead of imposing one constraint for the *worst* consistent labeling, we have a separate constraint for *each* such labeling, and want *all* of them to hold.

The unifying property of the above tasks is that each includes either a very large constraint set, or worse, an infinite one. In the former case, the model being learned could easily have insufficient capacity to satisfy all of the constraints simultaneously (i.e. the problem could be *infeasible*), while in the latter case, unless the constraints are highly-structured, expecting all of them to hold is unrealistic. For this reason, our approach *does not attempt* to satisfy all of the constraints simultaneously. Instead, it is based on the Lagrangian formulation, and parameterizes the Lagrange multipliers using a *model*. In general, as the complexity of this model increases, so too does its ability to satisfy the constraints. If it is under-parameterized (as will typically be the case), then some constraints are likely to be violated (Section 4 provides some intuition on how our approach copes with this situation). However, if we use e.g. a neural network, then because the same set of weights will be used for every constraint, the model will be capable of learning relationships and redundancies between constraints, and therefore could perform better than its apparent complexity might indicate.

We make five main contributions: (i) in Section 3, we introduce the idea of using the Lagrangian formulation when the Lagrange multipliers are not taken to be a simple vector, but instead are the output of a model, given some set of features; (ii) in Definition 1 of Section 5, we introduce a new notion of how one can *measure* constraint violations in our Lagrangian-model setting, in a way that permits theoretical results to be proved; (iii) later in Section 5, we prove a suboptimality and infeasibility guarantee; (iv) in Section 5.2 we prove a generalization result that applies to the per-query ranking fairness setting described above; (v) in Section 6, we provide an extensive experimental evaluation of several of the highly-constrained settings we have discussed.

2 Related Work

There is a significant body of work that handles heavily-constrained problems by projecting onto each constraint (see e.g. Wang and Bertsekas [11, 12] and references therein). The use of a Lagrangian-like formulation is also popular, but we are aware of relatively little work that tries to reduce the complexity of the Lagrange multiplier space itself. A near-exception is Cotter et al. [13], which points out that one can partition the constraint set, and then associate each partition element with the *maximum* over the constraint functions of its components, thereby creating an equivalent problem with only as many constraints as partition elements.

In recent years, with the growing popularity of various statistical fairness metrics, Lagrangian-like approaches have started to be applied to fairness problems, with the goal being to satisfy the constraints as a part of the main training pipeline, instead of being applied as a post processing step [14–17]. In such approaches, intersectional groups have always been a potential problem, as they require constraining an exponential number of group intersections, many of which may contain very few training examples. As a remedy, Kearns et al. [3] propose imposing constraints instead on subgroups defined by a class of membership functions over protected attributes. They then show how one can provably learn a classifier that satisfies false positive rate and coverage constraints over this class as long as it has bounded VC dimension. They additionally assume access to an oracle to find the maximally violated constraint at each step of their optimization. Hebert-Johnson et al. [18] also propose a similar approach for calibration-based fairness constraints.

Like these prior methods, our approach also seeks to exploit the redundancies in the dual solution, but does so by limiting the flexibility of the Lagrange multiplier model used to enforce constraints. As a result, our optimization strategy is simpler, and does not require keeping track of the most violated constraint. We are thus able to handle a more general set of problems, including the ranking fairness example in which we have one constraint per query, and finding the maximum violation over *all* queries seen so far would be prohibitively expensive. The downside to this generality is that our approach is only able to satisfy the constraints approximately, but as we show in our experiments, the quality of the approximation improves with the capacity of the Lagrange multiplier class used.

Another closely related area of research is fairness in ranking [e.g. 19, 6, 8, 20, 5, 21]. Many existing proposals for ranking fairness impose constraints *on average*, instead of *per query*, including more recent policy gradient based methods [20]. The exceptions that we are aware of that do handle per-query constraints do not impose them during training, but instead later while post-processing a pre-trained scoring model [7–9]. Our approach differs from both of these lines of research in that it attempts to impose per-query fairness constraints *during training*.

3 Modeling Lagrange Multipliers

Our goal is to find a parameter vector $\theta \in \Theta$ that minimizes an objective function $g : \Theta \to \mathbb{R}$, subject to a total of m inequality constraints defined by the functions $h_i : \Theta \to \mathbb{R}$ for $i \in [m]$:

$$\underset{\theta \in \Theta}{\operatorname{minimize}}\, g(\theta) \quad \text{s.t.} \quad \forall i \in [m]\,.h_i(\theta) \leq 0 \tag{1}$$

We do not assume that any of these functions are convex. Typically, the objective function g will be an average loss over some dataset, but our main focus is on the constraints. Our interest is in problems for which m is extremely large (or even unknown). For example, there might be one constraint per training example, in which case we could define $h_j(\theta) = h^\dagger(x_j; \theta)$ to be a function of the associated feature vector x_j. Alternatively, in a ranking problem, we might have one constraint per query, and could take $h_j(\theta) = h^\dagger(z_j; \theta)$, where z_j would contain features summarizing the jth query.

The usual approach to tackling such problems is to formulate the Lagrangian:

$$\mathcal{L}(\theta, \lambda) := g(\theta) + \sum_{i=1}^{m} \lambda_i h_i(\theta) \tag{2}$$

and to then seek a mixed (due to the non-convexity of g and the h_is, see e.g. Cotter et al. [15]) Nash equilibrium of the game that results from minimizing the Lagrangian in θ and maximizing it in λ. If we can find such an equilibrium, then the random variable corresponding to the θ portion of the mixed equilibrium will be feasible and optimal, in expectation. One way to accomplish this would be to iteratively choose the best-response in θ, and perform a stochastic gradient ascent update in λ, resulting in a long sequence of θ and λ, over which the uniform distribution will be an approximate mixed Nash equilibrium [e.g. 22, 23].

In our setting, m might be extremely large, so this approach can easily fail in practice, since there will simply be far too many Lagrange multipliers. For example, if the training data are provided as a continuous stream of *i.i.d.* samples, and there is one constraint per training example (or per-query, in the ranking setting), then the approach described above would perform *only one* update to each Lagrange multiplier (occurring the first and only time that it is encountered), and the constraints will be all-but ignored.

Instead, we propose modifying the Lagrangian in such a way that λ is no longer an element of $\mathbb{R}_+^m$, but is instead represented as a *function* $\vec{\lambda} : \Gamma \to \mathbb{R}_+^m$, where Γ is a set of learned parameters. The Lagrangian can then be approximated as:

$$\tilde{\mathcal{L}}(\theta, \gamma) = g(\theta) + \left\langle \vec{\lambda}(\gamma), \vec{h}(\theta) \right\rangle \tag{3}$$

where, to simplify the notation, we've written the sum of Equation 2 as an inner product, with $\vec{\lambda}(\gamma) = [\lambda_1(\gamma), \lambda_2(\gamma), \dots, \lambda_m(\gamma)]^T$ and $\vec{h}(\theta) = [h_1(\theta), h_2(\theta), \dots, h_m(\theta)]^T$. Notice that we recover the usual Lagrangian by defining $\Gamma = \mathbb{R}_+^m$ and taking $\vec{\lambda}$ to be the identity function.

4 Simple Example: Linear Multiplier Model

As a thought experiment, imagine that the ith constraint is associated with a non-negative feature vector $M_{i,:} \in \mathbb{R}_+^{\tilde{m}}$, where $\tilde{m} \ll m$, and that we've stacked these feature vectors in the matrix $M \in \mathbb{R}_+^{m \times \tilde{m}}$ with one row for each constraint. We'll take our Lagrange multiplier model to have a particularly simple form, with $\gamma \in \Gamma := \mathbb{R}_+^{\tilde{m}}$ being the parameter vector that we will learn, and $\vec{\lambda}$ being a *linear* function of these parameters, i.e. $\vec{\lambda}(\gamma) := M\gamma$. With this Lagrange multiplier model, the Lagrangian becomes:

$$\tilde{\mathcal{L}}(\theta, \gamma) = g(\theta) + \left\langle M\gamma, \vec{h}(\theta) \right\rangle = g(\theta) + \left\langle \gamma, M^T \vec{h}(\theta) \right\rangle.$$

Interestingly, the latter expression is the "usual" Lagrangian of the constrained problem:

$$\underset{\theta \in \Theta}{\text{minimize}}\; g(\theta) \quad \text{s.t.} \quad \forall i \in [\tilde{m}] \,.\, \left\langle M_{:,i}, \vec{h}(\theta) \right\rangle \leq 0. \tag{4}$$

Hence, there is an alternative interpretation of a linear Lagrange multiplier model: we can instead consider it to be the *usual* (non-approximated) Lagrangian, with a linear transformation being applied to the *constraint functions*, instead of the γs.

Shared multipliers: As an even simpler special-case, imagine that γ consists of a set of Lagrange multipliers that are *shared* among the constraints, i.e. for each $i \in [\tilde{m}]$, there is a set $S_i \subseteq [m]$ containing the indices of all constraints that will use γ_i as their Lagrange multiplier, with the S_is forming a partition of $[m]$. Then M will have exactly one element in each row equal to one and the others equal to zero, and the resulting constrained problem (Equation 4) will be:

$$\underset{\theta \in \Theta}{\text{minimize}}\; g(\theta) \quad \text{s.t.} \quad \forall i \in [\tilde{m}] \,.\, \frac{1}{|S_i|} \sum_{j \in S_i} h_j(\theta) \leq 0.$$

Here, we've written the constraints as an average, rather than a sum, since an arbitrary positive scaling factor can be applied to each constraint without changing the constrained problem.

As this example illustrates, this highly-simplified approach of "sharing" Lagrange multipliers has an important property: it *fails gracefully*. Specifically, when the Γ space doesn't have enough capacity to satisfy all of the constraints, it falls back on the behavior of satisfying particular *average* constraints.

Discussion: Our actual proposal (Section 3) is to associate a feature vector with each constraint, and to learn a not-necessarily-linear model (a neural network, for example) that learns the Lagrange multipliers based on this data. Hence, information could be shared between constraints in a more nuanced manner than a simple average (or a weighted average, as will be the case for a general non-negative M). However, we believe, based on the above discussion, that it will still be possible to interpret such a model as a transformation of the *constraints*, and that our proposal will therefore still tend to "fail gracefully". In Section 6.1, we provide experimental evidence that this intuition holds in practice. Making this notion precise is, we believe, an exciting area for future research.

5 Algorithm & Analysis

In the previous section, we observed that an under-parameterized Lagrange multiplier model cannot be expected to satisfy all of the constraints simultaneously (and expressed our hope that it would still tend to "fail gracefully"). For this reason, we need to *define* how we will measure the magnitudes of the constraint violations.

Definition 1. (**Violation function**) *We measure the violation in the m constraints by a function $\Phi : \mathbb{R}^m \to \mathbb{R}_+$, which for any random variable θ^* taking values in parameters Θ satisfies:*

$$\Phi\left(\mathbb{E}_{\theta \sim \theta^*}\left[\vec{h}(\theta)\right]\right) \leq \max_{\gamma \in \Gamma} \left\langle \vec{\lambda}(\gamma), \mathbb{E}_{\theta \sim \theta^*}\left[\vec{h}(\theta)\right]\right\rangle.$$

Table 1: Examples of violation functions that satisfy Definition 1. Here, θ^* is a random variable taking values on Θ, $\mathbf{1}\{\cdot\}$ is an indicator function, and $(z)_+$ retains the positive entries of vector z and replaces the others with 0. In the fourth row, $p, q \geq 1$ and $1/p + 1/q = 1$. The four examples of Φ include the positive part of the average constraint violation, the L_1-norm of the positive violations, the squared L_2-norm of the positive violations, and the p-norm violation measured on a linear transformation of the constraints (see linear model example in Section 4).

Assumption on multiplier model $\lambda(\cdot)$	$\Phi(z)$
$\Gamma = [0, 1/m]$ and $\forall x,\ \lambda(\gamma; x) = \gamma$	$\left(\frac{1}{m}\sum_{j=1}^{m} z_j\right)_+$
$\forall\theta^*, \exists\gamma$ s.t. $\vec{\lambda}(\gamma) = \mathbf{1}\left\{\mathbb{E}_{\theta\sim\theta^*}\left[\vec{h}(\theta)\right] > 0\right\}$	$\lVert(z)_+\rVert_1$
$\forall\theta^*, \exists\gamma$ s.t. $\vec{\lambda}(\gamma) = \left(\mathbb{E}_{\theta\sim\theta^*}\left[\vec{h}(\theta)\right]\right)_+$	$\lVert(z)_+\rVert_2^2$
$\Gamma = \left\{\gamma \in \mathbb{R}_+^{\tilde{m}} : \lVert\gamma\rVert_q \leq 1\right\}$ and $\vec{\lambda}(\gamma) = M\gamma$ for $M \in \mathbb{R}_+^{m\times\tilde{m}}$	$\lVert(M^\top z)_+\rVert_p$

One should observe that Φ is needed purely for our theory: it has no influence on the algorithm, which depends only on the Lagrange multiplier model *itself*. Given the form of this model (i.e. its function class), there are *many* compatible choices of Φ, each of which determines the sort of infeasibility and generalization guarantees that we can achieve. For example, if we choose the Lagrange multiplier models to be constant functions of the form $\lambda(\gamma; x) = \gamma$, for $\gamma \in [0, 1/m]$, we would be constraining the average constraint function $\frac{1}{m}\sum_{j=1}^{m} h_j(\theta)$ to be non-negative, and a choice of Φ that satisfies Definition 1 in this case is $\Phi(z) = \left(\frac{1}{m}\sum_{j=1}^{m} z_j\right)_+$. In Table 1, we give examples of Φ satisfying Definition 1 for some other simple function classes. For more complicated function classes, the form of Φ may not be known, but can still be reasoned about formally.

Ideally, we would like Φ to measure the maximum constraint violation, but this can be a very strong requirement, since it essentially states that the Lagrange multiplier model has sufficient capacity to penalize violations in each of the m constraints, even when m is extremely large. With that said, Table 1 shows that we can come up with reasonable candidates for Φ for some settings. Notice that, in this table, none of the provided choices of λ can grow without bound, which should be a cause for concern, since we might expect it to be necessary for λ to be very large for nearly-infeasible problems. For this reason, we actually scale the constraint portion of the Lagrangian by an additional positive scaling factor (R) in the upcoming theorem.

We are now ready to prove our main result, which builds on Agarwal et al. [14]. We will make a mild assumption that the Lagrange multiplier *model* is capable of representing an all-zero λ:

Assumption 1. *There exists a $\gamma \in \Gamma$ such that $\vec{\lambda}(\gamma) = 0$.*

Theorem 1. *For any given radius $R > 0$, suppose that θ^* and γ^* are random variables taking values from Θ and Γ (respectively), and that they're an $R\epsilon$-approximate mixed Nash equilibrium, i.e.:*

$$\sup_{\gamma\in\Gamma} \mathbb{E}\left[g(\theta^*) + R\left\langle \vec{\lambda}(\gamma), \vec{h}(\theta^*)\right\rangle\right] - \inf_{\theta\in\Theta} \mathbb{E}\left[g(\theta) + R\left\langle \vec{\lambda}(\gamma^*), \vec{h}(\theta)\right\rangle\right] \leq R\epsilon. \tag{5}$$

Notice that an additional weight of R has been applied to the constraint portion of the Lagrangian. Assuming that the problem is feasible and that Assumption 1 holds, and choosing $R = 1/\sqrt{\epsilon}$, we have that θ^* will be approximately optimal, in expectation:

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

Further for any violation function $\Phi : \mathbb{R}^m \to \mathbb{R}_+$ that satisfies Definition 1, we have that θ^* will likewise be approximately feasible (measured in terms of Φ):

$$\Phi\left(\mathbb{E}\left[\vec{h}(\theta^*)\right]\right) \leq G\sqrt{\epsilon} + \epsilon.$$

Where $G \geq \sup_{\theta\in\Theta} g(\theta) - \inf_{\theta\in\Theta} g(\theta)$ bounds the range of the objective function g.

Proof. In Appendix A. □

This theorem shows that, assuming that Assumption 1 holds and a suitable Φ exists, an approximate Nash equilibrium will be approximately feasible and optimal. Other refinements from the literature, such as the use of a non-zero-sum formulation with proxy-constraints [15, 17], or to improve constraint generalization [16], could be integrated with some extra effort.

5.1 Algorithm

Theorem 1 does not describe how one can *find* an approximate mixed Nash equilibrium, particularly given that the objective and constraint functions are not assumed to be convex. This however, is well-studied: as we mentioned in Section 3, it's straightforward to see that if we treat the Lagrangian as a two-player game, and have the players iteratively play against each other, with the θ player using best-response (i.e. choosing the global minimum in θ for a fixed γ—this is necessary due to the non-convexity of the objective g and constraints h_i), and the γ player playing e.g. stochastic gradient ascent (assuming that the Lagrange multiplier model λ is a concave function of γ), then the uniform distribution over the resulting sequence of iterates will define a pair of random variables θ^* and γ^* that form an approximate mixed Nash equilibrium, and, by Theorem 1, θ^* will be approximately feasible and optimal. In practice, having both players use e.g. SGD, and ignoring the non-convexity, often works well [e.g. 24], and this is the approach that we take in the experiments of Section 6.

We seek a mixed equilibrium because our Lagrangian is non-convex in θ, so a pure Nash equilibrium might not even *exist* [15]. A mixed equilibrium is therefore difficult to avoid, but it does come with the unfortunate consequence that it results in a *stochastic* classifier: instead of finding a single parameter vector θ, we find a *distribution* θ^* over multiple parameter vectors. At evaluation time, each time that we receive a new example, we will sample a $\theta \sim \theta^*$, and then classify the example using the sampled θ. Deterministic classifiers are of course much more convenient, and we've found that ignoring this issue by deterministically taking the last iterate (instead of the uniform distribution over all iterates), often (but not always) works well. Alternatively, one could first create a stochastic classifier by finding a mixed equilibrium, and then convert it into a deterministic classifier using e.g. the procedure of Cotter et al. [25].

We include further details in Appendix C.

5.2 Generalization

In heavily constrained problems, one obvious area of concern is *generalization*. The naïve approach would be to determine a separate generalization bound for each constraint, and to then apply the union bound to find a common bound on *all* constraints. However, this might not scale well to heavily-constrained problems like those that we discussed in Section 1, and in particular, cannot be applied *at all* in the setting in which we have (implicitly) an infinite number of constraints, e.g. when there is one constraint per example (or per query).

In Appendix B, we consider the setting in which the number of constraints is constant (but large), and each constraint function is represented as an *expectation* over the data distribution (or a subset of it, as in e.g. fairness over intersectional groups). Many previous papers describe how such constraints can be constructed [e.g. 1, 2, 24]. Interestingly, it turns out that it is possible to bound the generalization of constraints in such a way that the bound depends not on the number of constraints, but rather on a complexity measure of the function class associated with the Lagrange multiplier model, although the performance is measured indirectly, in terms of a particular choice of Φ.

Below, we will study the constraint generalization of our approach assuming that there is one constraint per example (or per query), and that the constraint function itself only depends on the particular example (or query) with which it is associated (e.g. ranking fairness). In the experiments of Section 6.3, we construct constraints in the ranking setting using an approach that is basically identical to that of Narasimhan et al. [5], except that we will include one constraint per query, instead of averaging over the entire dataset. One interesting thing about this setting is that, while during training we will only have access to those constraints that correspond to queries that occur in the training set, implicitly there is a constraint for every *possible* query. Since, at evaluation time, we will observe queries, and therefore constraints, that were not present in the training dataset, it's very important to ensure that the constraints *generalize well*.

We will use $\mathcal{D}$ to denote the underlying data distribution over instances $x \in \mathcal{X}$. We would like to find parameters θ that satisfy the following constraints: for each instance $x \in \mathcal{X}$, $h(\theta; x) \leq 0$. For the constraint associated with each x, we will use the Lagrange multiplier $\lambda(\gamma; x)$. Similar to Definition 1, we will measure the overall constraint violations with a particular choice of Φ:

$$\Phi(\theta) = \max_{\gamma \in \Gamma} \mathbb{E}_{x \sim \mathcal{D}} \left[\lambda(\gamma; x) h(\theta; x) \right].$$

Notice that unlike Definition 1, we've written Φ in terms of θ instead of the constraint functions $h(\theta; x)$, but the hs are themselves functions of θ, so this is nothing but a change of notation. In practice, we are given a sample S of n examples drawn from $\mathcal{D}$, and we instead work with the empirical constraint violation, given by:

$$\hat{\Phi}(\theta) = \max_{\gamma \in \Gamma} \frac{1}{n} \sum_{j=1}^{n} \lambda(\gamma; x_j) h(\theta; x_j).$$

We now bound the gap between the expected and empirical constraint violations.

Theorem 2. *Suppose* $\sup_{\gamma,x} \lambda(\gamma; x) \leq B_\lambda$ *and* $\sup_{\theta,x} |h(\theta; x)| \leq B_\theta$*. With probability at least* $1 - \delta$ *over the* i.i.d. *draw of* $S \sim \mathcal{D}^n$*, for all* $\theta \in \Theta$*:*

$$\Phi(\theta) \leq \widehat{\Phi}(\theta) + \widehat{\mathcal{R}}_n(\mathcal{H}_\Theta) + \widehat{\mathcal{R}}_n(\Lambda_\Gamma) + B_\lambda B_\theta \sqrt{\frac{\log(1/\delta)}{n}},$$

where $\widehat{\mathcal{R}}_n(\mathcal{H}_\Theta)$ *is the empirical Rademacher complexity of the class of constraint functions* $\{x \mapsto h(\theta; x) | \theta \in \Theta\}$ *for any sample of size* n *and* $\widehat{\mathcal{R}}_n(\Lambda_\Gamma)$ *is the empirical Rademacher complexity of the class of multipliers* $\{x \mapsto \lambda(\gamma; x) | \gamma \in \Gamma\}$.

Proof. In Appendix A. $\square$

6 Experiments

We present experiments on (i) fairness task with intersectional protected groups, (ii) a fairness task with noisy protected groups, and (iii) a ranking fairness task with per-query constraints. We report running times and additional experimental details in Appendices D–F.[1]

6.1 Fairness Constraints on Intersectional Groups

We first demonstrate how our approach *fails gracefully* as we decrease the complexity of the Lagrangian model. For this, we consider the task of training a linear classifier to predict crime rate for a community, subject to intersectional group fairness constraints. We use the Communities and Crime dataset [26], which contains 1,994 communities in the US described by 140 features, and seek to predict the per capita crime rate for each community. As with prior work [e.g. 15], we label the communities with a crime rate above the 70th percentile as 'high crime' and the others as 'low crime'. We form intersectional groups based on the percentages of the Black, Hispanic and Asian populations (z_1, z_2, z_3) in a community. We define a protected group using three thresholds (t_1, t_2, t_3), and include in it all communities where the race percentages $z_k \geq t_k, \forall k \in [3]$. We consider a total of 1000 threshold combinations (t_1, t_2, t_3) from $[0, 1]^3$, refer to each such group by G_{t_1,t_2,t_3}, and impose a constraint on its error rate: error(G_{t_1,t_2,t_3}). Specifically, we seek to minimize the overall error rate, subject to the constraint error$(G_{t_1,t_2,t_3}) \leq$ error(ALL) $+ 0.01$, for all (t_1, t_2, t_3), except those groups that contain less than 1% of the data. We end up with a total of 535 groups.

We solve this constrained optimization problem by using a multiplier model that takes the three thresholds (t_1, t_2, t_3) as input and assigns a Lagrange multiplier to the corresponding error rate constraint. During training, we sample a threshold combination at random and perform gradient updates on the Lagrangian for the sampled constraint. We experiment with five multiplier architectures, ranging from under-parameterized to over-parameterized models. This includes a *common* multiplier for all constraints, a linear model, and neural networks with one, two and three hidden layers (with 50 nodes each). We compare with an *unconstrained* baseline that optimizes error rate without constraints.

The violation in the constraint for group G_{t_1,t_2,t_3} is measured as error(G_{t_1,t_2,t_3}) − error(ALL) $- 0.01$, where a negative value indicates that the constraint is satisfied. The plots in Figure 1 shows different percentiles of the constraint violations, with the left-most point showing the median violation and the right-most point showing the 95th percentile violation. The unconstrained classifier incurs positive violations for all percentiles, while the 3-hidden layer multiplier model is able to satisfy most constraints on the training set. Interestingly, as we move from a 3-hidden layer model to a common multiplier, the higher percentile violations go up, but the lower percentile violations are still

[1]Code available at: https://github.com/google-research/google-research/tree/master/many_constraints.

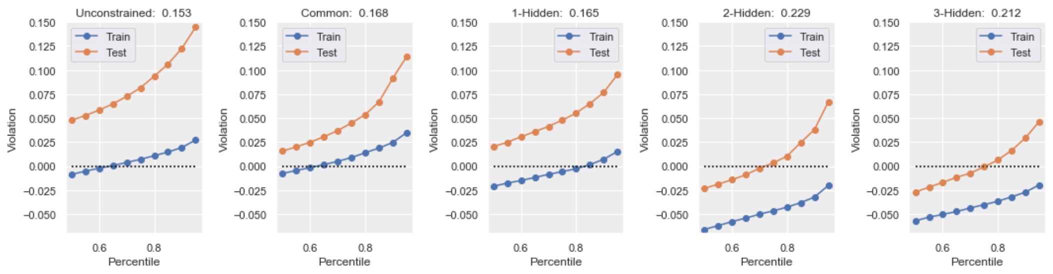

Figure 1: Plots of pth percentile violations as a function of p (averaged over 5 trials) for the intersectional group constraints on the Communities & Crime dataset. The overall test error rates (*lower* better) are reported in the titles. Negative violations indicate the constraints are satisfied.

Table 2: Comparison with the FairFictitiousPlay approach of Kearns et al. [3] with a brute-force oracle to find the most-violated constraint on the Communities & Crime intersectional fairness task. We report the error rates and 95th percentile violations (means and standard errors over 5 trials).

Method	Train		Test	
	Error	95th Perc. Violation	Error	95th Perc. Violation
Unconstrained	**0.11** $\pm$ **0.002**	0.05 $\pm$ 0.006	**0.15** $\pm$ **0.003**	0.09 $\pm$ 0.012
Common multiplier	0.14 $\pm$ 0.002	0.03 $\pm$ 0.003	0.17 $\pm$ 0.006	0.11 $\pm$ 0.015
Linear	0.14 $\pm$ 0.003	0.03 $\pm$ 0.004	0.15 $\pm$ 0.005	0.14 $\pm$ 0.004
1-hidden-layer	0.14 $\pm$ 0.013	0.02 $\pm$ 0.005	0.16 $\pm$ 0.010	0.10 $\pm$ 0.015
2-hidden-layer	0.21 $\pm$ 0.013	-0.02 $\pm$ 0.006	0.23 $\pm$ 0.012	0.07 $\pm$ 0.021
3-hidden-layer	0.20 $\pm$ 0.008	**-0.02** $\pm$ **0.004**	0.21 $\pm$ 0.009	0.05 $\pm$ 0.017
FairFictitiousPlay	0.20 $\pm$ 0.010	**-0.02** $\pm$ **0.004**	0.22 $\pm$ 0.011	**0.04** $\pm$ **0.008**

better than the unconstrained approach. Thus with a simpler architecture, the Lagrange multiplier model tries to satisfy a simpler transformation of the constraints, and is better-able to fail gracefully (Section 4). Despite the large number of constraints, the overall error doesn't reach the trivial error rate (0.3 for an all-negatives classifier), likely because of redundancies in the constraints.

We also compare our results with the FairFictitiousPlay algorithm of Kearns et al. [3], who also impose constraints on intersectional groups, but assume access to an oracle for finding the maximally violated constraint at each step. We implement this oracle using a brute-force search. As seen in Table 2, despite not having access to a brute-force oracle, the 3-hidden layer multiplier model is able to satisfy the constraints almost as well as FairFictitiousPlay, while yielding a similar error rate.

6.2 Fairness Constraints with Noisy Group Memberships

We next apply our proposed approach to impose group-based fairness constraints when the protected groups are noisy. The goal is to equalize true positive rates over the unknown, "true" group membership assignments, when we only have access to known noisy group membership assignments. Wang et al. [10] propose robust-optimization-like approaches for this problem. We replicate their setup, but propose a different optimization strategy. Denoting the true group for an instance by $G \in [k]$ and the noisy group by $\hat{G} \in [k]$, we assume access to the marginal probabilities $P(G = i|\hat{G} = j), \forall i, j \in [k]$. Then given a dataset $\{(x_i, y_i, \hat{G}_i)\}_{i=1}^n$ with noisy group memberships, we consider the set of all candidate true group memberships $\mathcal{Z} = \left\{\{G_i^{(1)}\}_{i=1}^n, \{G_i^{(2)}\}_{i=1}^n, \ldots\right\}$ that satisfy the marginal probabilities, and seek to impose the fairness constraints on each of these candidate memberships. Specifically, we wish to solve the following optimization problem:

$$\underset{\theta \in \Theta}{\text{minimize}}\; g(\theta) \;\text{ s.t. }\; \forall Z \in \mathcal{Z},\, j \in [k].\; h_j(\theta, Z) \leq 0, \tag{6}$$

where $h_j(\theta, Z)$ denotes an equal opportunity fairness constraint evaluated on group j with the candidate group memberships Z. Note that the number of constraints can be exponential in the number of instances n, and hence its impractical to maintain one Lagrange multiplier per constraint. Instead, we use a model that maps a group membership assignment Z to a Lagrange multiplier, and we sample group memberships from $\mathcal{Z}$ at random to perform gradient updates.

We use the UCI Adult dataset [26]. We optimize for the error rate subject to *equality of opportunity* [4] on three race groups 'white', 'black' and 'other', and train a linear classification model. We use

Table 3: Equal opportunity constraints on Adult with different group noise levels (mean and standard error over 10 trials). We report the test error rates (*lower* better) and test equal opportunity violations evaluated with the *true* groups. A negative fairness violation indicates the constraints are satisfied.

	DRO [10]		SoftAssign [10]		Proposed	
Noise	Error rate	Violation	Error rate	Violation	Error rate	Violation
0.1	0.152 ± 0.001	0.002 ± 0.019	$\mathbf{0.148 \pm 0.001}$	-0.048 ± 0.002	$\mathbf{0.147 \pm 0.002}$	0.015 ± 0.035
0.2	0.200 ± 0.002	-0.045 ± 0.003	$\mathbf{0.157 \pm 0.003}$	-0.048 ± 0.002	0.168 ± 0.006	-0.004 ± 0.013
0.3	0.216 ± 0.010	-0.044 ± 0.004	$\mathbf{0.158 \pm 0.005}$	0.002 ± 0.030	0.165 ± 0.003	-0.016 ± 0.002
0.4	0.209 ± 0.006	-0.019 ± 0.031	0.188 ± 0.003	-0.016 ± 0.016	$\mathbf{0.157 \pm 0.002}$	-0.005 ± 0.028
0.5	0.219 ± 0.012	-0.030 ± 0.032	0.218 ± 0.002	0.004 ± 0.006	$\mathbf{0.189 \pm 0.006}$	-0.020 ± 0.009

Table 4: Pairwise ranking fairness with per-query constraints on MSLR-WEB10K (averaged over 5 trials). Constraint violation is measured as $|\text{err}_{0,1} - \text{err}_{1,0}| - 0.25$ (*lower* is better).

	Pairwise Error		90th Percentile Violation	
Method	Train	Test	Train	Test
Unconstrained	$\mathbf{0.386 \pm 0.010}$	$\mathbf{0.394 \pm 0.008}$	0.068 ± 0.003	0.282 ± 0.029
Narasimhan et al. [5]	0.396 ± 0.014	0.400 ± 0.014	0.060 ± 0.004	0.246 ± 0.025
Proposed	0.406 ± 0.012	0.412 ± 0.011	$\mathbf{0.040 \pm 0.005}$	$\mathbf{0.195 \pm 0.023}$

a linear multiplier model which takes as input an n-dimensional boolean encoding of the sampled group membership assignment Z. We compare our approach with the two approaches proposed by Wang et al. [10], based on distributionally robust optimization (DRO) and "soft" group assignments.

Table 3 shows the error rates, and fairness violations on the true groups G for the trained classifiers, evaluated on the test set. As done by Wang et al. [10], we run five different experiments with five different noise levels added to the true groups in the training set: the "noisy" groups that we observe are a perturbation of the true groups using the given noise level. Higher noise yields a larger candidate set $\mathcal{Z}$. The baseline unconstrained model achieves an error rate of 0.145 ± 0.004 and a maximum constraint violation of 0.02 ± 0.05 [10]. Most importantly, the proposed approach is able to satisfy the fairness constraints on the true groups not seen during training, albeit with a higher variance for lower noise levels. Table 3 shows that when the noise level is low, the soft group assignments approach from Wang et al. [10] has better performance; when the noise level is high, the proposed approach achieves lower error rates than the more conservative soft assignments approach.

6.3 Ranking Fairness with Per-query Constraints

We finally consider the problem of imposing per-query constraints for ranking fairness. We adopt the pairwise ranking fairness setup from Narasimhan et al. [5], but unlike them, we wish to satisfy the fairness constraints per-query, and not on average across all queries. We use the Microsoft Learning to Rank Dataset (MSLR-WEB10K) [27] which contains 136 document features for each query-document pair. Similar to Yadav et al. [20], we divide all documents into two protected groups $G \in \{0, 1\}$ based on the 40th percentile of the QualityScore feature, and binarize the relevance labels. We also consider a second W3C experts ranking dataset [21], which we include in Appendix F.

Denoting the set of document features by $\mathcal{X}$ and query features by $\mathcal{Q}$, the goal is to learn a ranking function $f : \mathcal{X} \times \mathcal{Q} \to \mathbb{R}$ that assigns a score $f(X, Q)$ to a given query-document pair (X, Q). We adopt the pairwise equal opportunity fairness criteria from Narasimhan et al. [5] and measure the group-specific pairwise errors for groups $i, j \in \{0, 1\}$ and a given query $q \in \mathcal{Q}$ as:

$$\text{err}_{i,j}(q) = \mathbb{E}[\mathbf{1}\{f(X, q) < f(X', q)\} | Y > Y', G = i, G' = j, Q = q]$$

where (X, Y, G) and (X', Y', G') are tuples of document features, labels and groups sampled for query $Q = q$. We seek to then optimize the overall pairwise error subject to the constraint that the cross-group errors are similar for each query: $|\text{err}_{0,1}(q) - \text{err}_{1,0}(q)| \leq 0.25$, for all $q \in \mathcal{Q}$.

We use a one-hidden layer multiplier neural network model containing 128 nodes to assign a Lagrange multiplier for each query $q \in \mathcal{Q}$, with the average of the feature vectors within a query used as input. The ranking model is also a one-hidden layer neural network. Table 4 shows the pairwise error rates and 90th percentile query level constraint violations for our proposed method on a training set of 1000 queries and a test set of 100 queries, and compares against unconstrained training and the approach of Narasimhan et al. [5], which enforces the same constraints but *on average*. The proposed method achieves the least 90th percentile violation at the cost of a larger error rate.

Broader Impact

From an ethical standpoint, we believe that the main contribution of our work is that it will make it easier to approximately impose fine-grained statistical fairness constraints, particularly in the intersectional and ranking settings. The greatest weakness is that we cannot guarantee that every constraint will hold: instead, constraint violations are measured in terms of a function Φ, and while this function could technically be e.g. the magnitude of the most violated constraint, it is unlikely that any practical Lagrange multiplier model will work with such a Φ when the number of constraints is extremely large. However, as we show in our experiments, for many real-world problems our approach is effective in optimizing for reasonable choices of Φ (such as the 90-th percentile violation).

A particular advantage of our approach is that we hope that it will *fail gracefully*, i.e. even when all of the constraints cannot be satisfied simultaneously, the observed behavior will still fall-back to something that most would consider "sensible". However, except for the simple types of Lagrange multiplier models considered in Section 4, this remains largely an *intuition*, and while we provide experimental evidence that this intuition holds in practice, making this notion more precise is an exciting area for future research.

References

[1] Gabriel Goh, Andrew Cotter, Maya Gupta, and Michael P Friedlander. Satisfying real-world goals with dataset constraints. In *NIPS*, pages 2415–2423, 2016.

[2] Harikrishna Narasimhan. Learning with complex loss functions and constraints. In *AISTATS*, 2018.

[3] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *ICML*, pages 2564–2572, 2018.

[4] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *NIPS*, 2016.

[5] Harikrishna Narasimhan, Andrew Cotter, Maya Gupta, and Serena Wang. Pairwise fairness for ranking and regression. In *AAAI*, 2020.

[6] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. Fairness in recommendation through pairwise experiments. *KDD Applied Data Science Track*, 2019.

[7] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. Ranking with fairness constraints. *arXiv:1704.06840*, 2018.

[8] Ashudeep Singh and Thorsten Joachims. Policy learning for fairness in ranking. In *NeurIPS*, 2019.

[9] Krishnamurthy Dvijotham, Ivan Lobov, Anton Zhernov, Dan Calain, Michelle Gong, Timothy Mann, and Natrajan Chandrasekhar. The NodeHopper: Enabling low latency ranking with constraints via a fast dual solver. In *KDD*, 2020.

[10] Serena Wang, Wenshuo Guo, Harikrishna Narasimhan, Andrew Cotter, Maya Gupta, and Michael I Jordan. Robust optimization for fairness with noisy protected groups. *arXiv:2002.09343*, 2020.

[11] Mengdi Wang and Dimitri Bertsekas. Incremental constraint projection methods for variational inequalities. *Mathematical Programming*, 150, 05 2014.

[12] Mengdi Wang and Dimitri Bertsekas. Stochastic first-order methods with random constraint projection. *SIAM Journal on Optimization*, 26:681–717, 01 2016.

[13] Andrew Cotter, Maya Gupta, and Jan Pfeifer. A Light Touch for heavily constrained SGD. In *COLT*, pages 729–771, 2016.

[14] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna M. Wallach. A reductions approach to fair classification. In *ICML*, pages 60–69, 2018.

[15] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*, pages 300–332, 2019.

[16] Andrew Cotter, Maya Gupta, Heinrich Jiang, Nathan Srebro, Karthik Sridharan, Serena Wang, Blake Woodworth, and Seungil You. Training well-generalizing classifiers for fairness metrics and other data-dependent constraints. In *ICML*, 2019.

[17] Harikrishna Narasimhan, Andrew Cotter, and Maya Gupta. Optimizing generalized rate metrics with three players. In *NeurIPS*, 2019.

[18] Ursula Hebert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (Computationally-identifiable) masses. In *ICML*, pages 1939–1948, 2018.

[19] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *KDD*, pages 2219–2228, 2018.

[20] Himank Yadav, Zhengxiao Du, and Thorsten Joachims. Fair learning-to-rank from implicit feedback. *arXiv:1911.08054*, 2019.

[21] Meike Zehlike and Carlos Castillo. Reducing disparate exposure in ranking: A learning to rank approach. In *The Web Conference*, pages 2849–2855, 2020.

[22] S. A. Plotkin, D. B. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *FOCS*, pages 495–504, 1991.

[23] Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In Avrim Blum and Michael J. Kearns, editors, *COLT*, pages 325–332. ACM, 1996.

[24] Andrew Cotter, Heinrich Jiang, Serena Wang, Taman Narayan, Maya Gupta, Seungil You, and Karthik Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *JMLR*, 2019.

[25] Andrew Cotter, Harikrishna Narasimhan, and Maya Gupta. On making stochastic classifiers deterministic. In *NeurIPS*, 2019.

[26] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[27] Tao Qin and Tie-Yan Liu. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597, 2013. URL `http://arxiv.org/abs/1306.2597`.

[28] Giulia DeSalvo, Mehryar Mohri, and Umar Syed. Learning with deep cascades. In *Algorithmic Learning Theory*, pages 254–269. Springer, 2015.

[29] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. Cambridge University Press, 2009.

A Proofs

Theorem 1. *For any given radius $R > 0$, suppose that θ^* and γ^* are random variables taking values from Θ and Γ (respectively), and that they're an $R\epsilon$-approximate mixed Nash equilibrium, i.e.:*

$$\sup_{\gamma \in \Gamma} \mathbb{E}\left[g\left(\theta^*\right) + R\left\langle \vec{\lambda}\left(\gamma\right), \vec{h}\left(\theta^*\right)\right\rangle\right] - \inf_{\theta \in \Theta} \mathbb{E}\left[g\left(\theta\right) + R\left\langle \vec{\lambda}\left(\gamma^*\right), \vec{h}\left(\theta\right)\right\rangle\right] \leq R\epsilon. \tag{5}$$

Notice that an additional weight of R has been applied to the constraint portion of the Lagrangian. Assuming that the problem is feasible and that Assumption 1 holds, and choosing $R = 1/\sqrt{\epsilon}$, we have that θ^* will be approximately optimal, in expectation:

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

Further for any violation function $\Phi : \mathbb{R}^m \to \mathbb{R}_+$ that satisfies Definition 1, we have that θ^* will likewise be approximately feasible (measured in terms of Φ):

$$\Phi\left(\mathbb{E}\left[\vec{h}\left(\theta^*\right)\right]\right) \;\leq\; G\sqrt{\epsilon} + \epsilon.$$

Where $G \geq \sup_{\theta \in \Theta} g\left(\theta\right) - \inf_{\theta \in \Theta} g\left(\theta\right)$ bounds the range of the objective function g.

Proof. We'll mostly follow the analysis of Agarwal et al. [14]. We'll begin by proving the optimality portion. Let $\hat{\theta}$ minimize $g\left(\theta\right)$ over all *feasible* $\theta \in \Theta$ (i.e. all θs for which $\vec{h}\left(\theta\right) \preceq 0$). Plugging this into the right-hand side of Equation 5, and using Assumption 1:

$$\mathbb{E}\left[g\left(\theta^*\right)\right] \leq g\left(\hat{\theta}\right) + R\epsilon$$

which completes the optimality portion of the proof.

We'll now move on to feasibility. Again plugging in $\hat{\theta}$ on the right-hand side of Equation 5:

$$\begin{aligned}
\sup_{\gamma \in \Gamma} \mathbb{E}\left[g\left(\theta^*\right) + R\left\langle \vec{\lambda}\left(\gamma\right), \vec{h}\left(\theta^*\right)\right\rangle\right] &\leq g(\hat{\theta}) + R\epsilon \\
\mathbb{E}\left[g\left(\theta^*\right)\right] \;+\; R\sup_{\gamma \in \Gamma} \mathbb{E}\left[\left\langle \vec{\lambda}\left(\gamma\right), \vec{h}\left(\theta^*\right)\right\rangle\right] &\leq g(\hat{\theta}) + R\epsilon \\
R\sup_{\gamma \in \Gamma} \mathbb{E}\left[\left\langle \vec{\lambda}\left(\gamma\right), \vec{h}\left(\theta^*\right)\right\rangle\right] &\leq g(\hat{\theta}) - \mathbb{E}\left[g\left(\theta^*\right)\right] + R\epsilon \\
\sup_{\gamma \in \Gamma} \mathbb{E}\left[\left\langle \vec{\lambda}\left(\gamma\right), \vec{h}\left(\theta^*\right)\right\rangle\right] &\leq G/R + \epsilon.
\end{aligned}$$

where we used the upper bound on g. Further, using Definition 1, we get:

$$\Phi\left(\mathbb{E}\left[\vec{h}\left(\theta^*\right)\right]\right) \leq G/R + \epsilon.$$

Setting $R = 1/\sqrt{\epsilon}$ completes the feasibility portion of the proof. □

Theorem 2. *Suppose $\sup_{\gamma,x} \lambda(\gamma; x) \leq B_\lambda$ and $\sup_{\theta,x} |h(\theta; x)| \leq B_\theta$. With probability at least $1 - \delta$ over the* i.i.d. *draw of $S \sim \mathcal{D}^n$, for all $\theta \in \Theta$:*

$$\Phi(\theta) \leq \widehat{\Phi}(\theta) + \widehat{\mathcal{R}}_n(\mathcal{H}_\Theta) + \widehat{\mathcal{R}}_n(\Lambda_\Gamma) + B_\lambda B_\theta \sqrt{\frac{\log(1/\delta)}{n}},$$

where $\widehat{\mathcal{R}}_n(\mathcal{H}_\Theta)$ is the empirical Rademacher complexity of the class of constraint functions $\{x \mapsto h(\theta; x) | \theta \in \Theta\}$ for any sample of size n and $\widehat{\mathcal{R}}_n(\Lambda_\Gamma)$ is the empirical Rademacher complexity of the class of multipliers $\{x \mapsto \lambda(\gamma; x) | \gamma \in \Gamma\}$.

Proof. We would like to bound:

$$\max_{\theta \in \Theta}\left(\Phi(\theta) - \widehat{\Phi}(\theta)\right) \leq \max_{\theta \in \Theta, \gamma \in \Gamma}\left(\mathbb{E}_x\left[\lambda(\gamma; x) h(\theta; x)\right] - \frac{1}{n}\sum_{j=1}^{n} \lambda(\gamma; x_j) h(\theta; x_j)\right). \tag{7}$$

All we need to do is to bound the generalization error for the product $\lambda(\gamma;\cdot)h(\theta;\cdot)$ over $\Theta \times \Gamma$. We first directly apply a result from DeSalvo et al. [28] (Lemma 2) to bound the Rademacher complexity of this product class by sum of the Rademacher complexities of the individual classes. This result states that, if $\mathcal{C} = \{x \mapsto \lambda(\gamma;x)h(\theta;x) | \gamma \in \Gamma, \theta \in \Theta\}$, then the empirical Rademacher complexity of $\mathcal{C}$ for any sample of size n is given by $\widehat{\mathcal{R}}_n(\mathcal{C}) \le \widehat{\mathcal{R}}_n(\Lambda_\Gamma) + \widehat{\mathcal{R}}_n(\mathcal{H}_\Theta)$.

Equipped with this above result, we can apply standard uniform convergence based techniques to derive a generalization bound for the product $\lambda(\gamma;x)h(\theta;x)$. Using the fact that $\lambda(\gamma;x)h(\theta;x) \le B_\lambda B_\theta$ for all θ and γ, we have with probability at least $1-\delta$ that, for all $\theta \in \Theta$ and $\gamma \in \Gamma$:

$$\mathbb{E}_x\left[\lambda(\gamma;x)h(\theta;x)\right] \le \frac{1}{n}\sum_{j=1}^{n} \lambda(\gamma;x_j)h(\theta;x_j) + \widehat{\mathcal{R}}_n(\Lambda_\Gamma) + \widehat{\mathcal{R}}_n(\mathcal{H}_\Theta) + B_\lambda B_\theta \sqrt{\frac{\log(1/\delta)}{n}}.$$

Plugging this back into Equation 7 completes the proof. □

B Generalization for a Fixed Number of in-Expectation Constraints

In Section 5.2, we considered the one-constraint-per-example (or per-query) setting. Here, we consider the other major setting of interest, in which there are a known, fixed number of queries, and each constraint function is represented as an *expectation* over the data distribution (or a subset of it, as in e.g. fairness over intersectional groups). Examples of how such constraints might be constructed can be found in e.g. Goh et al. [1], Narasimhan [2], Cotter et al. [24].

We would like to find parameters θ that satisfy a constant number of m constraints, where the ith constraint is defined by:

$$h_i(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_i(x;\theta)\right],$$

where $\ell_i : \mathcal{X} \times \theta \to \mathbb{R}$. In practice, we are given a sample $S = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ drawn *i.i.d.* from $\mathcal{D}$, and we find a θ that satisfies the following empirical constraints instead:

$$h_i(\theta;S) = \frac{1}{n}\sum_{j=1}^{n} \ell_i(x_j;\theta).$$

We would like to bound the violations in the expected constraints in terms of the violations in the empirical constraints. To this end, we measure the expected constraint violation by:

$$\Phi(\theta) = \max_{\gamma\in\Gamma} \frac{1}{m}\left\langle \vec{\lambda}(\gamma), \vec{h}(\theta)\right\rangle$$

and the empirical constraint violation by:

$$\widehat{\Phi}(\theta) = \max_{\gamma\in\Gamma} \frac{1}{m}\left\langle \vec{\lambda}(\gamma), \vec{h}(\theta;S)\right\rangle.$$

We measure the complexity of the Lagrange multiplier model class using its pseudo-dimension.

Definition 2 (Pseudo-dimension of multiplier class). *We define the pseudo-dimension of the multiplier class Γ as the VC-dimension of the function class $\{(i,r) \mapsto \operatorname{sign}(\lambda_i(\gamma) - r) : \gamma \in \Gamma\}$.*

We now present our generalization bound for the constraint violation that is *independent of the number of constraints* and instead depends on the pseudo-dimension of the multiplier class:

Theorem 3. *Suppose $\left\|\vec{h}(\theta) - \vec{h}(\theta;S)\right\|_\infty \le G$ for all $\theta \in \Theta$ and $\left\|\vec{\lambda}(\gamma)\right\|_\infty \le B$ for all $\gamma \in \Gamma$. Suppose that, for any $\gamma \in \Gamma$, with probability at least $1-\delta$ over draw of $S \sim \mathcal{D}^n$, for all $\theta \in \Theta$:*

$$\frac{1}{m}\left\langle \vec{\lambda}(\gamma), \vec{h}(\theta) - \vec{h}(\theta;S)\right\rangle \le \operatorname{GenBnd}(\delta) \tag{8}$$

where $\operatorname{GenBnd}(\delta)$ *is an upper bound on the generalization error of the inner product on the LHS and a decreasing function of δ, and holds for any* particular γ. *We show that this generalization bound can be "lifted" to one that holds uniformly, with an additional term depending on the complexity of the multiplier model. To this end, fix $\rho > 0$. Then with probability at least $1-\delta$ over draw of $S \sim \mathcal{D}^n$, for all $\theta \in \Theta$:*

$$\Phi(\theta) \le \widehat{\Phi}(\theta) + \operatorname{GenBnd}\left(\delta\left(\frac{\rho}{B}\right)^D\right) + G\rho,$$

where D is the pseudo-dimension of the multiplier class, i.e. the class of functions $\{i \mapsto \lambda_i(\gamma) : \gamma \in \Gamma\}$. Also observe that for most generalization bounds, $\mathrm{GenBnd}(\delta)$ *will have a δ-dependence of the form* $\log(1/\delta)$.

Proof. We would like to bound:

$$\Phi(\theta) \leq \widehat{\Phi}(\theta) + \max_{\gamma \in \Gamma} \frac{1}{m} \left\langle \vec{\lambda}\,(\gamma), \vec{h}\,(\theta) - \vec{h}\,(\theta; S) \right\rangle . \tag{9}$$

Let $\tilde{\Gamma} \subseteq \Gamma$ be a finite set of "covering centers" such that, for all $\gamma \in \Gamma$ there exists a $\tilde{\gamma} \in \tilde{\Gamma}$ for which $\frac{1}{m} \left\| \vec{\lambda}\,(\gamma) - \vec{\lambda}\,(\tilde{\gamma}) \right\|_1 \leq \rho$. Then:

$$\begin{aligned}\frac{1}{m} \left\langle \vec{\lambda}\,(\gamma), \vec{h}\,(\theta) - \vec{h}\,(\theta; S) \right\rangle \leq & \frac{1}{m} \left| \left\langle \vec{\lambda}\,(\tilde{\gamma}), \vec{h}\,(\theta) - \vec{h}\,(\theta; S) \right\rangle \right| \\ & + \frac{1}{m} \left\| \vec{\lambda}\,(\gamma) - \vec{\lambda}\,(\tilde{\gamma}) \right\|_1 \left\| \vec{h}\,(\theta) - \vec{h}\,(\theta; S) \right\|_\infty \\ \leq & \frac{1}{m} \left\langle \vec{\lambda}\,(\tilde{\gamma}), \vec{h}\,(\theta) - \vec{h}\,(\theta; S) \right\rangle + G\rho .\end{aligned}$$

Using the above relationship we can approximate the the 'max' over Γ in Equation 9 with a 'max' over the covering centers $\tilde{\Gamma}$:

$$\Phi(\theta) \leq \widehat{\Phi}(\theta) + \max_{\gamma \in \tilde{\Gamma}} \frac{1}{m} \left\langle \vec{\lambda}\,(\gamma), \vec{h}\,(\theta) - \vec{h}\,(\theta; S) \right\rangle + G\rho .$$

We then bound the 'max' over the covering centers using the generalization bound in Equation 8 and a union bound over $\tilde{\Gamma}$, and have with probability at least $1 - \delta$:

$$\Phi(\theta) \leq \widehat{\Phi}(\theta) + \mathrm{GenBnd}\left(\frac{\delta}{|\tilde{\gamma}|} \right) + G\rho . \tag{10}$$

A standard bound [e.g. 29] on the size of the covering centers for a function class in terms of its pseudo-dimension D is:

$$\left| \tilde{\Gamma} \right| \leq \mathcal{O}\left(\left(\frac{B}{\rho} \right)^D \right) .$$

Substituting this into Equation 10, combined with the observation that $\mathrm{GenBnd}(\cdot)$ will be a decreasing function of its argument, completes the proof. ☐

C Outline of Algorithm

"Theoretical" algorithm: We'll begin by showing that the best-response based algorithm that we mentioned in Section 5.1 converges to a mixed Nash equilibrium. Algorithm 1 has the θ-player using best response at each iteration (i.e. finding a global minimizer of the modeled Lagrangian of Equation 3), and the γ-player using (stochastic) gradient ascent, under the assumption that $\vec{\lambda}$ is a concave function of γ. The particular choice of algorithm for the γ-player doesn't matter—we only need it to satisfy a low-regret guarantee:

$$\sup_{\gamma} \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathcal{L}}(\theta^{(t)}, \gamma) - \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathcal{L}}(\theta^{(t)}, \gamma^{(t)}) \leq \epsilon \tag{11}$$

where ϵ depends the regret bound of whatever algorithm is being used (for stochastic gradient ascent on a concave objective, it will be $\mathcal{O}(1/\sqrt{T})$). For the θ-player, best-response trivially gives us a no-regret guarantee:

$$\begin{aligned}\frac{1}{T} \sum_{t=1}^{T} \tilde{\mathcal{L}}(\theta^{(t)}, \gamma^{(t)}) - \frac{1}{T} \sum_{t=1}^{T} \inf_{\theta \in \Theta} \tilde{\mathcal{L}}(\theta, \gamma^{(t)}) =& 0 \\ \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathcal{L}}(\theta^{(t)}, \gamma^{(t)}) - \inf_{\theta \in \Theta} \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathcal{L}}(\theta, \gamma^{(t)}) \leq& 0 \end{aligned} \tag{12}$$

Algorithm 1 finds an approximate mixed Nash equilibrium of the modeled Lagrangian of Equation 3: $\tilde{\mathcal{L}}(\theta,\gamma) = g(\theta) + \left\langle \vec{\lambda}(\gamma), \vec{h}(\theta) \right\rangle$. As we discussed in Section 5.1, the θ-player plays best response, while the γ-player plays (stochastic) gradient ascent. The use of best response is necessary because we do not assume that the objective or constraint functions are convex. There is nothing special about gradient ascent, however. In Appendix C, we discuss this further, and include an outline of a convergence proof.

```
$\theta^{(0)} \leftarrow 0$    // or some other initial value ....
$\gamma^{(0)} \leftarrow 0$    // ....
for $t = 1$ to $T$ do
    $J_\gamma \leftarrow$ the Jacobian of $\vec{\lambda}(\gamma^{(t-1)})$    // or a stochastic Jacobian
    $\Delta_\gamma \leftarrow J_\gamma^T \vec{h}(\theta^{(t-1)})$
    $\gamma^{(t)} \leftarrow \gamma^{(t-1)} + \eta \Delta_\gamma$
    $\theta^{(t)} \leftarrow \mathrm{argmin}_{\theta \in \Theta} \left( g(\theta) + \left\langle \vec{\lambda}(\gamma^{(t)}), \vec{h}(\theta) \right\rangle \right)$
end for
$\theta^* \leftarrow$ a random variable that equals $\theta^{(t)}$ with probability $1/T$ for every $t \in [T]$
$\gamma^* \leftarrow$ a random variable that equals $\gamma^{(t)}$ with probability $1/T$ for every $t \in [T]$
```

Algorithm 2 uses stochastic gradient descent and ascent to optimize the modeled Lagrangian of Equation 3: $\tilde{\mathcal{L}}(\theta,\gamma) = g(\theta) + \left\langle \vec{\lambda}(\gamma), \vec{h}(\theta) \right\rangle$. This is basically the algorithm that we use in our experiments (although often with a different first-order method than SGD). Because we do not assume convexity, this algorithm is *not* guaranteed to converge to a mixed Nash equilibrium, but it seems to work well in practice.

```
$\theta^{(0)} \leftarrow 0$    // or some other initial value ....
$\gamma^{(0)} \leftarrow 0$    // ....
for $t = 1$ to $T$ do
    $G_\theta \leftarrow$ a stochastic gradient of $g(\theta^{(t-1)})$
    $J_\theta \leftarrow$ a stochastic Jacobian of $\vec{h}(\theta^{(t-1)})$
    $\Delta_\theta \leftarrow G_\theta + J_\theta^T \vec{\lambda}(\gamma^{(t-1)})$
    $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta_\theta \Delta_\theta$
    $J_\gamma \leftarrow$ a stochastic Jacobian of $\vec{\lambda}(\gamma^{(t-1)})$
    $\Delta_\gamma \leftarrow J_\gamma^T \vec{h}(\theta^{(t-1)})$
    $\gamma^{(t)} \leftarrow \gamma^{(t-1)} + \eta_\gamma \Delta_\gamma$
end for
```

Adding Equations 11 and 12 together yields:

$$\sup_{\gamma} \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathcal{L}}(\theta^{(t)}, \gamma) - \inf_{\theta \in \Theta} \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathcal{L}}(\theta, \gamma^{(t)}) \leq \epsilon$$
$$\sup_{\gamma} \mathbb{E}\left[g(\theta^*) + \left\langle \vec{\lambda}(\gamma), \vec{h}(\theta^*) \right\rangle \right] - \inf_{\theta \in \Theta} \mathbb{E}\left[g(\theta) + \left\langle \vec{\lambda}(\gamma^*), \vec{h}(\theta) \right\rangle \right] \leq \epsilon$$

which is, aside from the R scaling, exactly the requirement of Theorem 1 (Equation 5).

"Practical" algorithm: In our experiments, we use something closer to Algorithm 2, which is not guaranteed to converge to a mixed Nash equilibrium, but is easy to implement and works well in practice. This algorithm, however, leaves one major question unanswered: "how do we find stochastic gradients?". Typically, this will be done by sampling. The objective function will generally be an empirical average over a dataset, so we can find a stochastic gradient of g by sampling a minibatch from the training set. When the constraints have the same form, the same approach can be used, and when the constraints are functions of individual examples (or queries), each row of the Jacobian can be calculated exactly without sampling.

Since our underlying assumption is that m is extremely large, we would *also* like to sample the constraints. This, again, is straightforward: we sample a constraint $i \in [m]$ uniformly, find a (stochastic) gradient of h_i, and construct a Jacobian matrix that is all-zero, except that the ith row is equal to this gradient, multiplied by m (to account for the sampling probability). This approach could of course be improved upon (minibatching being the most obvious change, but different importance sampling strategies could also be beneficial), but this discussion shows that, in principle, each iteration can be made extremely computationally inexpensive.

Notice that, unlike Algorithm 1, in Algorithm 2 we do not construct and return uniform distributions over the iterates. While one could do this, there are numerous other options, as well: one could take the last iterate, or heuristically search for the "best" iterate, or even search for the best pair of random variables supported on the iterates (i.e. the "shrinking" procedure of Cotter et al. [15]).

D Experimental Details for Fairness over Intersectional Groups

As noted in Section 6.1, we seek to minimize the overall error rate, subject to the constraint error$(G_{t_1,t_2,t_3}) \leq$ error(ALL) $+ 0.01$, for all 1000 thresholds (t_1, t_2, t_3), except those groups that contain less than 1% of the data. This results in a total of 822 constraints. Since some of the threshold combinations represent the same partitioning of the data, only 535 of the constraints are unique (i.e. are computed on distinct groups).

We use Algorithm 2 to solve the resulting constrained optimization problem, where at each step we sample a single threshold combination (t_1, t_2, t_3) uniformly at random, and compute a stochastic gradient using the sampled constraint. We use Adagrad for the individual gradient updates, run the algorithm for a total $T = 10000$ gradient steps, with step-sizes $\eta_\theta = 0.1$ and $\eta_\gamma = 0.5$. After training, we use the "best iterate" heuristic of Cotter et al. [24] to pick a model that best trades-off between the objective and constraints. We train a linear classifier in all experiments and use hinge surrogates to approximate the objective and constraints. We implemented our method in TensorFlow using the open-source TensorFlow Constrained Optimization (TFCO) library [24][2].

We split the dataset into 2/3-rd for training and 1/3-rd for testing, and average the results over 5 such random train-test splits. We measure the violation in the constraint for group G_{t_1,t_2,t_3} as: error$(G_{t_1,t_2,t_3}) -$ error(ALL) $- 0.01$, where a negative violation value indicates that the constraint is satisfied. In Table 2, we report the error rates and the 95-th percentile constraint violation for imposing the constraints with five Lagrangian model architectures: (i) a *common* multiplier for all constraints, (ii) a linear model, (iii) a neural network with a single hidden layer of 50 nodes, (iv) a neural network with two hidden layers of 50 nodes each, and (v) a neural network with three hidden layers of 50 nodes each. Note that (iv) and (v) are over-parameterized models, i.e. have more parameters than the number of constraints. We compare our approach with an *unconstrained* baseline that optimizes error rate without constraints.

We ran experiments on a single machine with a 36-core Intel(R) Xeon(R) Gold 6154 Processor (3.00 GHz) and 191GB RAM. For this dataset, our approach incurs similar running times with the different Lagrange multiplier models. For example, with the linear multiplier model it takes 31.7 minutes, whereas with the 3-layer multiplier model it takes 33 minutes.

D.1 Comparison with Kearns et al. [3]

As noted in Section 2, Kearns et al. [3] also consider the setting of intersectional fairness constraints, but impose constraints on subgroups defined by a class of membership functions over protected attributes. They then show how one can provably impose constraints on these subgroups when the class of membership functions has bounded VC dimension. On the other hand, we make no assumption on the given groups, but instead limit the flexibility of the Lagrange multiplier model used to impose constraints on them.

Our optimization strategy is also very different. Kearns et al. [3] propose having the λ- and θ-players play Fictitious Play using an oracle to find the most-violated constraint at each step. Such an oracle, however, many not be efficient to implement for general constraint sets. So in our algorithm (Algorithm 2), we do not assume access to an oracle for picking the most-violated constraint, but instead sample a constraint uniformly at random, compute gradients for the sampled constraint, and perform stochastic gradient updates on γ and θ.

[2]https://github.com/google-research/tensorflow_constrained_optimization

For completeness, we compare our results with the FairFictitiousPlay algorithm of Kearns et al. [3] (see Algorithm 3 in their paper). We impose equal error rate constraints on the same threshold-based groups considered in our original experiment, and use a brute-force search strategy to find the most-violated constraint. Note that such a brute-force oracle may not be efficient to implement when the number of protected attributes is very large. We run their algorithm for a total of 100 rounds of Fictitious Play, with 250 gradient steps to implement the cost-sensitive learner needed for the θ-player's update step, and with the multiplier radius parameter C chosen from the range $\{1.0, 5.0, 10.0\}$ to minimize the train error rate while satisfying the constraints.

As seen in Table 2, the proposed approach with a 3-hidden layer Lagrangian model is able to satisfy the constraints at least as well as the brute-force FairFictitiousPlay variant, while yielding a similar error rate. Moreover, the proposed approach is able to achieve this performance using a simple random sampling strategy to pick the constraint at each step, and without relying on an inefficient brute-force search to pick the maximally violated constraint.

E Experimental Details for Fairness with Noisy Group Memberships

The generation of noisy protected groups is the same as in Wang et al. [10]. In the experiments, we replace all the expectations in the objective and constraints with finite-sample empirical versions.

Let $f(x;\theta)$ be a binary classifier, where $f(x;\theta) > 0$ indicates a positive classification. Let $Y \in \{0,1\}$ be a binary classification label. Recall that M is a candidate true group assignment over a dataset $\mathcal{D}$: in finite samples, $M \in \mathcal{M} = \left\{ \{G_j^{(1)}\}_{i=1}^n, \{G_i^{(2)}\}_{i=1}^n, \ldots \right\}$, and $M_i = j$ implies that the ith example is assigned to group j.

To equalize true positive rates between groups with slack α, we set

$$h_j(\theta, M) = \frac{\mathbb{E}_{x,y\sim\mathcal{D}}[\mathbf{1}\{f(x;\theta) > 0, y = 1\}]}{\mathbb{E}_{x,y\sim\mathcal{D}}[\mathbf{1}\{y = 1\}]} - \frac{\mathbb{E}_{x,y,M\sim\mathcal{D}}[\mathbf{1}\{f(x;\theta) > 0, y = 1, M = j\}]}{\mathbb{E}_{x,y,M\sim\mathcal{D}}[\mathbf{1}\{y = 1, M = j\}]} - \alpha.$$

We evaluate fairness violations $h_j(\theta, G)$ for the true group membership assignment G on the test dataset. We do not directly use G to train the binary classifier f. The only place we use G in the training process is to estimate the marginal probabilities $P(G = i|\hat{G} = j)$ ahead of time on the training set, as done by Wang et al. [10]. These marginal probabilities are used to generate the candidate set $\mathcal{M}$.

We use Algorithm 2 to solve the resulting constrained optimization problem, where at each step we sample a group membership assignment M uniformly at random, and compute a stochastic gradient using the sampled constraint. We train a linear classifier in all experiments and use hinge surrogates to approximate the objective and constraints. For the Lagrange multiplier, we use a linear model with a one-hot encoding M as input. While this amounts to maintaining a total of n parameters, one for each entry M_i, this is far fewer than the total number of constraints, which is exponential in n.

After training, we report results from the best iterate, where we define "best" as the iterate that achieves the lowest objective value [24], while also satisfying all constraints on a sample of 20 candidates from the candidate set $\mathcal{M}$.

For this experiment, we trained on a single machine with a 6-core Intel Xeon E5-1650 V3 Processor (3.50GHz) and 32GB RAM. The average training time over 5 trials for the proposed method was 13.36 minutes. The results from Wang et al. [10] were taken directly from the results reported in that paper, and were not retrained or reproduced in this paper. Their results were trained using a 4-core Intel Core i7-7700HQ CPU (2.80GHz) and 16GB RAM. The average training times over 5 trials are: DRO [10]: 5.56 minutes, and SoftAssign[10]: 5.52 minutes.

E.1 Comparison to baseline with a single Lagrange multiplier per group

As a baseline to illustrate the effectiveness of the Lagrange multiplier model, we also trained using the same setup as Section 6.2, but instead of the linear Lagrange multiplier model described in Section 6.2, we simply update a single Lagrange multiplier λ_j per group $j \in [k]$. We still resample the group assignments $M \in \mathcal{M}$ each epoch, but we associate a single Lagrange multiplier λ_j with constraint $h_j(\theta, M)$. Results in Table 5 on the five different noise levels show that this single

Table 5: Equal opportunity constraints on Adult with different group noise levels in the training data (averaged over 10 trials). We report the test error rates (*lower* better) and test fairness violations evaluated with the *true* groups. A negative fairness violation indicates the constraints are satisfied.

	Per-group multiplier	
Noise	Error rate	Violation
0.1	0.146 $\pm$ 0.001	0.008 $\pm$ 0.030
0.2	0.146 $\pm$ 0.002	0.040 $\pm$ 0.015
0.3	0.146 $\pm$ 0.001	0.057 $\pm$ 0.018
0.4	0.146 $\pm$ 0.002	0.068 $\pm$ 0.021
0.5	0.145 $\pm$ 0.002	0.068 $\pm$ 0.008

Lagrange multiplier per group achieved approximately the same error rate and true group violations as the unconstrained model. Thus, even though the group assignments $M \in \mathcal{M}$ were resampled each epoch, the shared Lagrange multiplier per group did not have the complexity to control the violations over all group assignments $M \in \mathcal{M}$.

E.2 Relation to soft group assignments from Wang et al. [10]

We discuss the relation of the generated candidate groups with the soft group assignments proposed in Wang et al. [10]. We show that the generated samples are actually "hard" group assignments, which are in a subset of the soft group assignments. Therefore, the proposed algorithm in Section 6.2 solves an optimization problem in which the constraints are sampled from the feasible set of the soft group assignments.

The soft group assignments approach in Wang et al. [10] uses a function $w : [k] \times \{0,1\} \times \{0,1\} \times [k] \to [0,1]$ to estimate $P(G = j|\hat{y}, y, \hat{G} = k)$ by $w(j \mid \hat{y}, y, k)$, where $\hat{y} = \mathbf{1}(f(x;\theta) > 0)$. The soft group assignments approach in Wang et al. [10] then constrains w in a feasible set $\mathcal{W}$ by enforcing the marginal probabilities estimated from auxiliary datasets:

$$\mathcal{W}(\theta) = \left\{ w : \substack{\sum_{\hat{y},y\in\{0,1\}} w(j|\hat{y},y,k)P(\hat{y},y|\hat{G}=k)=P(G=j|\hat{G}=k), \\ \sum_{j=1}^{m} w(j|\hat{y},y,k)=1, w(j|\hat{y},y,k)\geq 0 \quad \forall \hat{y},y\in\{0,1\},k\in\hat{\mathcal{G}}} \right\}. \tag{13}$$

The robust optimization problem takes the form:

$$\begin{aligned} &\underset{\theta\in\Theta}{\text{minimize}}\ g(\theta) \\ &\quad \text{s.t. } \forall j \in [k]. \quad \max_{w\in\mathcal{W}} h_j(\theta, w) \leq 0 \end{aligned}$$

where $g(\theta)$ represents the training loss, i.e. average error rates; $h_j(\theta) = \mathbb{E}[l_1(\theta, x, y)|G = j]$ where $l_1(\theta, x, y)$ represents some function to constrain over the true groups to ensure certain fairness constraints (see Wang et al. [10]).

Instead of solving the maximization as Wang et al. [10], the proposed algorithm in Section 6.2 solves this problem empirically by sampling from the candidate set of possible true group assignments. The candidate set $\mathcal{M}$ that we consider in Section 6.2 is a subset of $\mathcal{W}(\theta)$, where the function w simply assigns a "hard" group assignment to each data point, i.e. $\{w : w \in \mathcal{W}, w(j \mid \hat{y}, y, k) = 1 \text{ for some } j \in [k]\}$. Any feasible w in this subset corresponds to applying a candidate true group assignment to the data points.

While any feasible solution to the robust optimization problem would be guaranteed to satisfy the fairness constraints on the true groups, this is not necessarily true of the empirical procedure in Section 6.2 of sampling a different candidate from $\mathcal{M}$ every epoch to perform gradient updates.

F Experimental Details for Ranking Fairness with Per-query Constraints

By labeling documents with relevance score 3 and 4 as positive, we end up with approximately 12.5% positive documents overall. As the pairwise constraints are better captured when the positive and negative labels are balanced, we take a two-step approach to slightly adjust the ratio of positive and negative documents in the training set. We first generate two [-1,1] uniformly distributed random

Table 6: Pairwise ranking fairness with per-query constraints on W3C Expert (averaged over 5 trials). Constraint violation is measured as $|\text{err}_0 - \text{err}_1| - 0.3$ (*lower* is better).

	Pairwise Error		90th Percentile Violation	
Method	Train	Test	Train	Test
Unconstrained	**0.4446** $\pm$ 0.0009	**0.4704** $\pm$ 0.0758	0.2170 $\pm$ 0.0000	0.3252 $\pm$ 0.0247
Narasimhan et al. [5]	0.4528 $\pm$ 0.0018	0.4992 $\pm$ 0.0937	0.1932 $\pm$ 0.0205	0.2336 $\pm$ 0.0888
Proposed	0.4706 $\pm$ 0.0333	0.5384 $\pm$ 0.0844	**0.1104** $\pm$ 0.0852	**0.2082** $\pm$ 0.0499

numbers (v_1, v_2) per query and discard a negative document from the query with the probability $(0.7 + 0.3 \cdot v_1) \cdot \frac{1}{2}\left(1 - v_2 \cdot \frac{m+1-2i}{m-1}\right)$ if it has the ith smallest QualityScore2 (feature 133) in the query. As a result, v_1 influences the number of negative documents, and v_2 controls the quality of the selected negative documents. After applying this step, we also filter out all queries that have either less than 10 positive documents or less than 10 negative documents. By doing so we prevent the calculation of cross-group error rates with overly small denominators. Finally, reporting the error rates, we assign a penalty of 0.5 for ties in the ranking.

We use Algorithm 2 to solve the constrained optimization problem, where at each step we receive a single query, and compute a stochastic gradient using the sampled constraint. We use Adagrad for the individual gradient updates, run the algorithm for a total $T = 100000$ gradient steps, with step-sizes $\eta_\theta = 0.05$ and $\eta_\gamma = 0.1$. After training, we use the "best iterate" heuristic of Cotter et al. [24] to pick a model that best trades-off between the objective and constraints. We train a ranking model with a single hidden layer of 128 nodes, and use hinge surrogates to approximate the objective and constraints. The Lagrange multiplier model is also a single-hidden-layer neural network with 128 nodes which takes the mean feature vector from a query as input. We implemented our method in TensorFlow using the open-source TensorFlow Constrained Optimization (TFCO) library [24].

We ran experiments on a single machine with a 36-core Intel(R) Xeon(R) Gold 6154 Processor (3.00 GHz) and 188GB RAM. The average training times over 5 trials for the unconstrained method is 3.13 hours, for the approach of Narasimhan et al. [5] is 4.43 hours, and for the proposed approach is 5.05 hours.

F.1 Additional Experiments

We consider a similar ranking problem with a marginal equal opportunity fairness criterion defined as

$$\text{err}_i(q) = \mathbb{E}[\mathbf{1}\{f(X, q) < f(X', q)\}|Y > Y', G = i, Q = q]$$

and constraining $|\text{err}_0(q) - \text{err}_1(q)| \leq 0.3$ for all queries $q \in \mathcal{Q}$. We use the W3C Expert dataset studied in Zehlike and Castillo [21], Narasimhan et al. [5], which is a subset of the TREC 2005 enterprise track data. The dataset contains 60 topics, with 200 candidates per topic, where each candidate labeled as an expert or non-expert for the topic. The goal is to rank the candidates based on their expertise on a topic. We use the same features as Zehlike and Castillo [21] to represent how well each topic matches each candidate (a set of five aggregate features derived from word counts and tf-idf scores), and treat gender as the protected attribute. We learn a ranking model with one hidden layer of 8 nodes, and a multiplier model with one hidden layer of 8 nodes. We use 30 queries to learn both models using 20k gradient steps and a learning rate of 0.01, 15 queries to validate, then test the model performance on the left-out 15 queries. Table 6 summaries the pairwise error rates and the 90th percentile query level constraint violations. We see similar patterns as in Table 4 that the proposed method achieves the least constraint violation. However because of the small size of this dataset, we find that the standard deviations are generally higher, and the overall pairwise error on the test set is closer to 0.5. Because the proposed method more closely satisfies the per-query constraints it incurs a larger training error, and as a result a higher test error than the other methods.