Training Well-Generalizing Classifiers for Fairness Metrics and Other Data-Dependent Constraints

Andrew Cotter [1] **Maya Gupta** [1] **Heinrich Jiang** [1] **Nathan Srebro** [2] **Karthik Sridharan** [3] **Serena Wang** [1]
Blake Woodworth [2] **Seungil You** [4]

Abstract

Classifiers can be trained with data-dependent constraints to satisfy fairness goals, reduce churn, achieve a targeted false positive rate, or other policy goals. We study the generalization performance for such constrained optimization problems, in terms of how well the constraints are satisfied at evaluation time, given that they are satisfied at training time. To improve generalization, we frame the problem as a two-player game where one player optimizes the model parameters on a training dataset, and the other player enforces the constraints on an independent validation dataset. We build on recent work in two-player constrained optimization to show that if one uses this two-dataset approach, then constraint generalization can be significantly improved. As we illustrate experimentally, this approach works not only in theory, but also in practice.

1. Introduction

It is useful to train classifiers with data-dependent constraints in order to achieve certain guarantees on the training set, such as statistical parity or other fairness guarantees, specified recall, or a desired positive classification rate (e.g. Scott & Nowak, 2005; Zafar et al., 2015; Goh et al., 2016; Woodworth et al., 2017; Narasimhan, 2018)). However, a key question is whether the achieved constraints will *generalize*. For example: will a classifier trained to produce 80% statistical parity on training examples still achieve 80% statistical parity at evaluation time?

Unfortunately, the answer is "not quite." Because such constraints are data-dependent, overfitting can occur, and constraints that were satisfied on the training set should be expected to be slightly violated on an *i.i.d.* test set. This is particularly problematic in the context of fairness constraints, which will typically be chosen based on real-world requirements (e.g. the 80% rule of some US laws (Biddle, 2005; Vuolo & Levy, 2013; Zafar et al., 2015; Hardt et al., 2016)). In this paper, we investigate how well constraints generalize, and propose algorithms to improve the generalization of constraints to new examples.

Specifically, we consider problems that minimize a loss function subject to data-dependent constraints, expressed in terms of *expectations* over a data distribution $\mathcal{D}$:

$$\min_{\theta\in\Theta} \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0\left(x;\theta\right)\right] \quad \underset{\forall i\in[m]}{\text{s.t.}} \ \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_i\left(x;\theta\right)\right] \leq 0 \quad (1)$$

where $x \in \mathcal{X}$ is a feature vector, $\mathcal{D}$ is the data distribution over $\mathcal{X}$, Θ is a space of model parameters for the function class of interest, and $\ell_0, \ell_1, \ldots, \ell_m : \mathcal{X} \times \Theta \to \mathbb{R}$ are losses associated with the objective and the m constraints*. We do *not* require these loss functions to be convex. Appendix B† contains two examples of how Equation 1 can be used to express certain data-dependent constraints (see Goh et al. (2016); Narasimhan (2018) for more).

One typically trains a classifier on a finite training set drawn from $\mathcal{D}$, but the true goal is to satisfy constraints *in expectation over* $\mathcal{D}$, as in Equation 1. To this end, we build on a long line of prior work that treats constrained optimization as a two-player game (e.g. Christiano et al., 2011; Arora et al., 2012; Rakhlin & Sridharan, 2013; Kearns et al., 2017; Narasimhan, 2018; Agarwal et al., 2018). The first player optimizes the model parameters θ, and the second player enforces the constraints, e.g. using the Lagrangian:

$$\mathcal{L}\left(\theta,\lambda\right) := \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0\left(x;\theta\right) + \sum_{i=1}^{m} \lambda_i \ell_i\left(x;\theta\right)\right] \quad (2)$$

In practice, one would approximate the Lagrangian with an *i.i.d.* training sample from $\mathcal{D}$, and the first player would

[1]Google AI, Mountain View, CA, USA [2]Toyota Technological Institute at Chicago, Chicago, IL, USA [3]Cornell University, Computer Science Department, Ithaca, NY, USA [4]Kakao Mobility, Seongnam-si, Geyonggi-do, South Korea. Correspondence to: Andrew Cotter <acotter@google.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

*Table 5, in the supplementary material, summarizes notation.

†Appendices can be found in the supplementary material.

minimize over the model parameters $\theta \in \Theta$ while the second player maximizes over the Lagrange multipliers $\lambda \in \mathbb{R}^m_+$.

Our key idea is to treat constrained optimization similarly to hyperparameter optimization: just as one typically chooses hyperparameters based on a validation set, instead of the training set, to improve classifier generalization, we would like to choose the Lagrange multipliers on a validation set to improve *constraint* generalization. In "inner" optimizations we would, given a fixed λ, minimize the empirical Lagrangian on the training set. Then, in an "outer" optimization, we would choose a λ that results in the constraints being satisfied on the validation set. Such an approach, could it be made to work, would not eliminate the constraint generalization problem completely—hyperparameter overfitting (e.g. Ng, 1997) is a real problem—but would mitigate it, since constraint generalization would no longer depend on size of the training sample and the complexity of Θ (which could be extremely large, e.g. for a deep neural network), but rather on the size of the validation sample and the effective complexity of $\mathbb{R}^m_+ \ni \lambda$, which, being m-dimensional, is presumably much simpler than Θ.

The above approach is intuitive, but challenges arise when attempting to analyze it. The most serious is that since θ is chosen based on the training set, and λ on the validation set, the θ-player is minimizing a *different function* than the λ-player is maximizing, so the two-player game is *non-zero-sum* (the players have different cost functions). To handle this, we must depart from the Lagrangian formulation, but the key idea remains: improving generalization by using a separate validation set to enforce the constraints.

Fortunately, the recent work of Cotter et al. (2019) gives a strategy for dealing with a non-zero-sum game in the context of constrained supervised learning. We adapt their approach to our setting to give bounds on constraint generalization that are agnostic to model complexity. After some preliminary definitions in Section 3, in Section 4 we present algorithms for which we can provide theoretical bounds.

In Section 5, we perform experiments demonstrating that our two-dataset approach successfully improves constraint generalization *even when our theorems do not hold*. In other words, providing independent datasets to each player works well as a heuristic for improving constraint generalization.

2. Related Work

While several recent papers have proved generalization bounds for constrained problems (e.g. Goh et al., 2016; Agarwal et al., 2018; Donini et al., 2018), the problem of *improving* constraint generalization is a fairly new one, having, so far as we know, only been previously considered in the work of Woodworth et al. (2017), who handled generalization subject to "equalized odds" constraints in the setting of Hardt et al. (2016). Specifically, their approach is to first learn a predictor on $S^{(\mathrm{trn})}$, and then to learn a "correction" on $S^{(\mathrm{val})}$ to more tightly satisfy the fairness constraints. The second stage requires estimating only a constant number of parameters, and the final predictor consequently enjoys a generalization guarantee for the fairness constraints which is independent of the predictor's complexity, with only a modest penalty to the loss. However, their approach relies heavily upon the structure of equalized odds constraints: it requires that any classifier can be modified to satisfy the fairness constraints *and* have low loss on a validation set by tuning only a small number of parameters.

Woodworth et al. (2017)'s approach can be summarized as "train a complicated model on a training set, and then a simple correction on a validation set". If, as they show to be the case for equalized odds constraints, the "simple correction" is capable of satisfying the constraints without significantly compromising on quality, then this technique results in a well-performing model for which validation constraint generalization depends not on the complexity of the "complicated model", but rather of the "simple correction". In this paper, we extend this two-dataset idea to work on data-dependent constraints *in general*.

Our main baseline is Agarwal et al. (2018)'s recently-proposed algorithm for fair classification using the Lagrangian. Their proposal, like Algorithm 1, uses an oracle to optimize w.r.t. θ (they use the terminology "best response"), and, like all of our algorithms, results in a stochastic classifier. However, our setting differs slightly from theirs—they focus on fair classification, while we work in the more general inequality-constrained setting (Equation 1). For this reason, in Appendix F we provide an analysis of the Lagrangian formulation for inequality constrained optimization.

3. Background & Definitions

Our algorithms are based on the non-zero-sum two-player game proposed by Cotter et al. (2019), which they call the "proxy-Lagrangian" formulation. The key novelty of their approach is the use of "proxy" constraint losses, which are essentially surrogate losses that are used by only *one* of the two players (the θ-player). It is because the two players use different losses that their proposed game is non-zero-sum. The motivation behind their work is that a surrogate might be necessary when the constraint functions are non-differentiable or discontinuous (e.g. for fairness metrics, which typically constrain proportions, i.e. linear combinations of indicators), but the overall goal is still to satisfy the original (non-surrogate) constraints. Our work differs in that we use a non-zero-sum game to provide different *datasets* to the two players, rather than different *losses*.

Despite this difference, the use of proxy-constraints is per-

fectly compatible with our proposal, so we permit the approximation of each of our constraint losses ℓ_i with a (presumably differentiable) upper-bound $\tilde{\ell}_i$. These are used *only* by the θ-player; the λ-player uses the original constraint losses. The use of proxy constraint losses is entirely optional: one is free to choose $\tilde{\ell}_i := \ell_i$ for all i.

Definition 1. *Let $S^{(\mathrm{trn})}$ and $S^{(\mathrm{val})}$ be two random datasets each drawn* i.i.d. *from a data distribution $\mathcal{D}$. Given proxy constraint losses $\tilde{\ell}_i(x;\theta) \geq \ell_i(x;\theta)$ for all $x \in \mathcal{X}$, $\theta \in \Theta$ and $i \in [m]$, the empirical proxy-Lagrangians $\hat{\mathcal{L}}_\theta, \hat{\mathcal{L}}_\lambda : \Theta \times \Lambda \to \mathbb{R}$ of Equation 1 are:*

$$\hat{\mathcal{L}}_\theta(\theta,\lambda) := \frac{1}{|S^{(\mathrm{trn})}|} \sum_{x \in S^{(\mathrm{trn})}} \left(\lambda_1 \ell_0(x;\theta) + \sum_{i=1}^{m} \lambda_{i+1} \tilde{\ell}_i(x;\theta) \right)$$

$$\hat{\mathcal{L}}_\lambda(\theta,\lambda) := \frac{1}{|S^{(\mathrm{val})}|} \sum_{x \in S^{(\mathrm{val})}} \sum_{i=1}^{m} \lambda_{i+1} \ell_i(x;\theta)$$

where $\Lambda := \Delta^m \subseteq \mathbb{R}_+^{m+1}$ is the m-probability-simplex.

The difference between the above, and Definition 2 of Cotter et al. (2019), is that $\hat{\mathcal{L}}_\theta$ is an empirical average over the training set, while $\hat{\mathcal{L}}_\lambda$ is over the validation set. The θ-player seeks to minimize $\hat{\mathcal{L}}_\theta$ over θ, while the λ-player seeks to maximize $\hat{\mathcal{L}}_\lambda$ over λ. In words, the λ-player will attempt to satisfy the *original* constraints on the *validation* set by choosing how much to penalize the *proxy* constraints on the *training* set.

3.1. Generalization

Our ultimate interest is in generalization, and our bounds will be expressed in terms of both the training and validation generalization errors, defined as follows:

Definition 2. *Define the training generalization error bound $\tilde{G}^{(\mathrm{trn})}(\Theta)$ such that:*

$$\left| \mathbb{E}_{x \sim \mathcal{D}}[\ell(x,\theta)] - \frac{1}{|S^{(\mathrm{trn})}|} \sum_{x \in S^{(\mathrm{trn})}} \ell(x,\theta) \right| \leq \tilde{G}^{(\mathrm{trn})}(\Theta)$$

for all $\theta \in \Theta$ and all $\ell \in \left\{\ell_0, \tilde{\ell}_1, \ldots, \tilde{\ell}_m\right\}$ (the objective and proxy constraints, but not the original constraints).

Likewise, define the validation generalization error bound $G^{(\mathrm{val})}(\hat{\Theta})$ to satisfy the analogous inequality in terms of $S^{(\mathrm{val})}$, for all $\theta \in \hat{\Theta} \subseteq \Theta$ and all $\ell \in \{\ell_1, \ldots, \ell_m\}$ (the original constraints, but not the objective or proxy constraints).

Throughout this paper, $\hat{\Theta} := \{\theta^{(1)}, \ldots, \theta^{(T)}\} \subseteq \Theta$ is the set of T iterates found by one of our proposed algorithms. Each of our guarantees will be stated for a particular stochastic model $\bar{\theta}$ supported on $\hat{\Theta}$ (i.e. $\bar{\theta}$ is a distribution over $\hat{\Theta}$), instead of for a single deterministic $\theta \in \hat{\Theta}$. Notice that the above definitions of $\tilde{G}^{(\mathrm{trn})}(\Theta)$ and $G^{(\mathrm{val})}(\hat{\Theta})$ also apply to such stochastic models: by the triangle inequality, if every $\theta \in \hat{\Theta}$ generalizes well, then any $\bar{\theta}$ supported on $\hat{\Theta}$ generalizes equally well, in expectation.

4. Algorithms

Our overall goal is to demonstrate that providing the θ- and λ-players with independent datasets improves constraint generalization, and in this section we prove that it does so in the context of the proxy-Lagrangian game of Cotter et al. (2019). They proposed having the θ-player minimize ordinary external regret, and the λ-player minimize swap regret using an algorithm based on Gordon et al. (2008). Rather than finding a *single* solution (a pure equilibrium of Definition 1), they found a *distribution* over solutions (a mixed equilibrium). Our proposed approach follows this same pattern, but we build on top of it to address *generalization*.

To this end, we draw inspiration from Woodworth et al. (2017) (see Section 2), and isolate the constraints from the complexity of Θ by using two independent datasets: $S^{(\mathrm{trn})}$ and $S^{(\mathrm{val})}$. The "training" dataset will be used to choose the model parameters θ, and the "validation" dataset to choose λ, and thereby impose the constraints. Like Woodworth et al. (2017), the resulting constraint generalization bounds will be *independent* of the complexity of the function class.

We'll begin, in Section 4.1, by proposing and analyzing an oracle-based algorithm that improves generalization by discretizing the candidate set, but makes few assumptions (not even convexity). Next, in Section 4.2, we give an algorithm that is more "realistic"—there is no oracle, and no discretization—but requires stronger assumptions.

In Section 5, we will present and perform experiments on simplified "practical" algorithms with no guarantees, but that incorporate our key idea: having the λ-player use an independent validation set.

4.1. Covering-based Algorithm

The simplest way to attack the generalization problem, and the first that we propose, is to *discretize* the space of allowed λs, and associate each λ with a unique $\theta \in \Theta$, where this association is based *only* on the training set. If the set of discretized λs is sufficiently small, then the set of discretized θs will likewise be small, and since it was chosen independently of the validation set, its validation performance will generalize well.

Specifically, we take C_r to be a radius-r (external) covering of $\Lambda := \Delta^m \subseteq \mathbb{R}_+^{m+1}$ w.r.t. the 1-norm. The set of allowed λs is the covering centers, while, following Chen et al. (2017), Agarwal et al. (2018) and Cotter et al. (2019), the θs are found using an oracle:

Definition 3. *A ρ-approximate Bayesian optimization oracle is a function $\mathcal{O}_\rho : (\Theta \to \mathbb{R}) \to \Theta$ mapping functions to*

Algorithm 1 Finds an approximate equilibrium of the empirical proxy-Lagrangian game (Definition 1), with Theorem 1 being its convergence and generalization guarantee. This is essentially a discretized version of Algorithm 4 of Cotter et al. (2019)—like that algorithm, because of its dependence on an oracle, this algorithm does *not* require convexity. Here, $\mathcal{O}_\rho$ is a deterministic Bayesian optimization oracle (Definition 3), and C_r is a radius-r (external) covering of $\Lambda := \Delta^m \subseteq \mathbb{R}_+^{m+1}$ w.r.t. the 1-norm. The θ-player uses oracle calls to approximately minimize $\hat{\mathcal{L}}_\theta(\cdot, \lambda)$, while the λ-player uses a swap-regret minimizing algorithm in the style of Gordon et al. (2008), using the left-stochastic state matrices $M^{(t)} \in \mathbb{R}^{(m+1)\times(m+1)}$.

DiscreteTwoDataset $\left(\hat{\mathcal{L}}_\theta, \hat{\mathcal{L}}_\lambda : \Theta \times \Delta^m \to \mathbb{R}, \mathcal{O}_\rho : (\Theta \to \mathbb{R}) \to \Theta, C_r \subseteq \mathbb{R}^{m+1}, T \in \mathbb{N}, \eta_\lambda \in \mathbb{R}_+\right)$:

Initialize $M^{(1)} \in \mathbb{R}^{(m+1)\times(m+1)}$ with $M_{i,j} = 1/(m+1)$

For $t \in [T]$:

  Let $\lambda^{(t)} = \text{fix}\, M^{(t)}$ *// Fixed point of $M^{(t)}$, i.e. a stationary distribution*

  Let $\tilde{\lambda}^{(t)} = \text{argmin}_{\tilde{\lambda} \in C_r} \left\| \lambda^{(t)} - \tilde{\lambda} \right\|_1$ *// Discretization to closest point in C_r*

  Let $\theta^{(t)} = \mathcal{O}_\rho\left(\hat{\mathcal{L}}_\theta\left(\cdot, \tilde{\lambda}^{(t)}\right)\right)$

  Let $\hat{\Delta}_\lambda^{(t)}$ be a supergradient of $\hat{\mathcal{L}}_\lambda\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. λ

  Update $\tilde{M}^{(t+1)} = M^{(t)} \odot . \exp\left(\eta_\lambda \hat{\Delta}_\lambda^{(t)} \left(\lambda^{(t)}\right)^T\right)$ *// $\Delta\lambda^T$ is an outer product; $\odot$ and $.\exp$ are element-wise*

  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 w.r.t. KL divergence*

Return $\theta^{(1)}, \ldots, \theta^{(T)}$ and $\lambda^{(1)}, \ldots, \lambda^{(T)}$

θs such that, if we take $\hat{\theta} = \mathcal{O}_\rho(\hat{\mathcal{L}}_\theta(\cdot, \lambda))$ for some λ, and θ^* is a minimizer of $\hat{\mathcal{L}}_\theta(\cdot, \lambda)$, then $f(\hat{\theta}) \leq f(\theta^*) + \rho$. Furthermore, every time it is given the same f, $\mathcal{O}_\rho$ will return the same θ (i.e. it is deterministic).

We will take the discretized set of θs to be the oracle solutions corresponding to the covering centers, i.e. $\Theta_{C_r} := \{\mathcal{O}_\rho(\hat{\mathcal{L}}_\theta(\cdot, \tilde{\lambda})) : \tilde{\lambda} \in C_r\}$. The proof of the upcoming theorem shows that if the radius parameter r is sufficiently small, then for any achievable objective function value and corresponding constraint violations, there will be a $\theta \in \Theta_{C_r}$ that is almost as good. Hence, despite the use of discretization, we will still be able to find a nearly-optimal and nearly-feasible solution. Additionally, since the set of discretized classifiers is finite, we can apply the standard generalization bound for a finite function class, which will be tightest when we take r to be as large as possible while still satisfying our optimality and feasibility requirements.

Algorithm 1 combines our proposed discretization with the oracle-based optimization procedure proposed by Cotter et al. (2019). As desired, it finds a sequence of solutions $\hat{\Theta} := \{\theta^{(1)}, \ldots, \theta^{(T)}\}$ for which it is possible to bound $G^{(\text{val})}(\hat{\Theta})$ *independently* of the complexity of the function class parameterized by Θ, *and* finds a random parameter vector $\bar{\theta}$ supported on $\hat{\Theta}$ that is nearly optimal and feasible.

Theorem 1. *Given any $\epsilon > 0$, there exists a covering C_r such that, if we take $T \geq 4B_\Delta^2 (m+1)\ln(m+1)/\epsilon^2$ and $\eta_\lambda = \sqrt{(m+1)\ln(m+1)/TB_\Delta^2}$, where $B_\Delta \geq \max_{t\in[T]} \left\|\Delta_\lambda^{(t)}\right\|_\infty$ is a bound on the gradients, then the following hold, where $\hat{\Theta} := \left\{\theta^{(1)}, \ldots, \theta^{(T)}\right\}$ is the set of results of Algorithm 1.*

Optimality and Feasibility: *Let $\bar{\theta}$ be a random variable taking values from $\hat{\Theta}$, defined such that $\bar{\theta} = \theta^{(t)}$ with probability $\lambda_1^{(t)} / \sum_{s=1}^T \lambda_1^{(s)}$, and let $\bar{\lambda} := \left(\sum_{t=1}^T \lambda^{(t)}\right)/T$. Then $\bar{\theta}$ is nearly-optimal in expectation:*

$$\mathbb{E}_{\bar{\theta}, x\sim\mathcal{D}}\left[\ell_0\left(x;\bar{\theta}\right)\right] \leq \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0\left(x;\theta^*\right)\right] + \frac{1}{\bar{\lambda}_1}\left(\rho + 2\epsilon + 2\tilde{G}^{(\text{trn})}(\Theta) + G^{(\text{val})}(\hat{\Theta})\right) \tag{3}$$

where θ^* minimizes $\mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0(x;\cdot)\right]$ subject to the proxy-constraints $\mathbb{E}_{x\sim\mathcal{D}}\left[\tilde{\ell}_i(x;\theta^*)\right] \leq 0$. It is also nearly-feasible:

$$\max_{i\in[m]} \mathbb{E}_{\bar{\theta}, x\sim\mathcal{D}}\left[\ell_i\left(x;\bar{\theta}\right)\right] \leq \frac{\epsilon}{\bar{\lambda}_1} + G^{(\text{val})}(\hat{\Theta}) \tag{4}$$

Additionally, if there exists a $\theta' \in \Theta$ that satisfies all of the constraints with margin γ (i.e. $\mathbb{E}_{x\sim\mathcal{D}}\left[\ell_i(x;\theta')\right] \leq -\gamma$ for all $i \in [m]$), then:

$$\bar{\lambda}_1 \geq \frac{1}{\gamma + B_{\ell_0}}\left(\gamma - \rho - 2\epsilon - 2\tilde{G}^{(\text{trn})}(\Theta) - G^{(\text{val})}(\hat{\Theta})\right) \tag{5}$$

where $B_{\ell_0} \geq \sup_{\theta\in\Theta} \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0(x;\theta)\right] - \inf_{\theta\in\Theta} \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0(x;\theta)\right]$ is a bound on the range of the objective loss.

Generalization: *With probability $1-\delta$ over the sampling of $S^{(\text{val})}$:*

$$G^{(\text{val})}(\hat{\Theta}) < B_\ell \sqrt{\frac{m\ln(10B_{\tilde{\ell}}/\epsilon) + \ln(2m/\delta)}{2\left|S^{(\text{val})}\right|}} \tag{6}$$

where $B_{\tilde{\ell}} \geq |\ell(x,\theta)|$ for all $\ell \in \left\{\ell_0, \tilde{\ell}_1, \ldots, \tilde{\ell}_m\right\}$, and $B_\ell \geq \max_{i\in[m]}(b_i - a_i)$ assuming that the range of each ℓ_i is the interval $[a_i, b_i]$.

Proof. In Appendix E.1. □

When reading the above result, it's natural to wonder about the role played by $\bar{\lambda}_1$. Recall that, unlike the Lagrangian formulation, the proxy-Lagrangian formulation (Definition 1) has a weight λ_1 associated with the *objective*, in addition to the m weights $\lambda_2, \ldots, \lambda_{m+1}$ associated with the constraints. When the ith constraint is violated, the corresponding λ_{i+1} will grow, pushing λ_1 towards zero. Conversely, when the constraints are satisfied, λ_1 will be pushed towards one. In other words, the magnitude of λ_1 encodes the λ-player's "belief" about the feasibility of the solution. Just as, when using the Lagrangian formulation, Lagrange multipliers will tend to be small on a feasible problem, the proxy-Lagrangian objective weight $\bar{\lambda}_1$ will tend to be large on a feasible problem, as shown by Equation 5, which guarantees that $\bar{\lambda}_1$ will be bounded away from zero provided that there exists a margin-feasible solution with a sufficiently large margin γ. In practice, of course, one need not rely on this lower bound: one can instead simply inspect the behavior of the sequence of $\lambda_1^{(t)}$'s during optimization.

Equation 5 causes our results to be gated by the feasibility margin. Specifically, it requires the training and validation datasets to generalize well enough for $\rho + 2\epsilon + 2\tilde{G}^{(\mathrm{trn})}(\Theta) + G^{(\mathrm{val})}(\hat{\Theta})$ to stay within the feasibility margin γ. Past this critical threshold, $\bar{\lambda}_1$ can be lower-bounded by a constant, and can therefore be essentially ignored. To get an intuitive grasp of this condition, notice that it is similar to requiring γ-margin-feasible solutions on the training dataset to generalize well enough to also be margin-feasible (with a smaller margin) on the validation dataset, and vice-versa.

Table 1 contains a comparison of our bounds, obtained with the proxy-Lagrangian formulation and two datasets, versus bounds for the standard Lagrangian on one dataset. The "Assuming" column contains a condition resulting from the above discussion. There are two key ways in which our results improve on those for the one-dataset Lagrangian: (i) in the "Infeasibility" column, our approach depends on $G^{(\mathrm{val})}(\hat{\Theta})$ instead of $\tilde{G}^{(\mathrm{trn})}(\Theta)$, and (ii): as shown in Table 2, for our algorithms the generalization performance $G^{(\mathrm{val})}(\hat{\Theta})$ of the constraints is bounded *independently* of the complexity of Θ.

It's worth emphasizing that this generalization bound (Table 2) is distinct from the feasibility bound (the "Infeasibility" column of Table 1). When using our algorithms, testing constraint violations will *always* be close to the validation violations, regardless of the value of $\bar{\lambda}_1$. The "Assuming" column is only needed when asking whether the validation violations are close to zero.

‡ This condition could be removed by defining the feasibility margin γ in terms of $S^{(\mathrm{trn})}$ instead of $\mathcal{D}$, but in the context of generalization, we should assume that $S^{(\mathrm{trn})}$ is drawn *i.i.d.* from $\mathcal{D}$; imposing a stricter requirement (a margin condition) on how it is sampled would defeat the purpose.

4.2. Gradient-based Algorithm

Aside from the unrealistic requirement for a Bayesian optimization oracle, the main disadvantage of Algorithm 1 is that it relies on discretization. Our next algorithm instead makes much stronger assumptions—strong convexity of the objective and proxy constraint losses, and Lipschitz continuity of the original constraint losses—enabling us to dispense with discretization entirely in both the algorithm and the corresponding theorem statement.

The *proof* of the upcoming theorem, however, still uses a covering. The central idea is the same as before, with one extra step: thanks to strong convexity, every (approximate) minimizer of $\hat{\mathcal{L}}_\theta(\cdot, \lambda)$ is close to one of the discretized parameter vectors $\theta \in \Theta_{C_r}$. Hence, the set of such minimizers generalizes as well as Θ_{C_r}, plus an additional term measuring the cost that we pay for approximating the minimizers with elements of Θ_{C_r}.

The strong convexity assumption also enables us to replace the oracle call with an explicit minimization procedure: gradient descent. The result is Algorithm 2, which, like Algorithm 1, both finds a nearly-optimal and nearly-feasible solution, *and* enables us to bound $G^{(\mathrm{val})}(\hat{\Theta})$ independently of the complexity of Θ. Unlike Algorithm 1, however, it is realistic enough to permit a straightforward implementation.

Theorem 2. *Suppose that Θ is compact and convex, and that $\ell(x;\theta)$ is μ-strongly convex in θ for all $\ell \in \left\{\ell_0, \tilde{\ell}_1, \ldots, \tilde{\ell}_m\right\}$. Given any $\epsilon > 0$, if we take $T_\theta \geq \left(B_{\check{\Delta}}^2/\mu\epsilon\right)\ln\left(B_{\check{\Delta}}^2/\mu\epsilon\right)$, $T_\lambda \geq 4B_\Delta^2\,(m+1)\ln(m+1)/\epsilon^2$ and $\eta_\lambda = \sqrt{(m+1)\ln(m+1)/T_\lambda B_\Delta^2}$, where B_Δ is as in Theorem 1 and $B_{\check{\Delta}} \geq \max_{s,t\in[T_\theta]\times[T_\lambda]}\left\|\check{\Delta}_\theta^{(t,s)}\right\|_2$ is a bound on the subgradients, then the following hold, where $\hat{\Theta} := \left\{\theta^{(1)}, \ldots, \theta^{(T_\lambda)}\right\}$ is the set of results of Algorithm 1.*

Optimality and Feasibility: *Let $\bar{\theta}$ be a random variable taking values from $\hat{\Theta}$, defined such that $\bar{\theta} = \theta^{(t)}$ with probability $\lambda_1^{(t)}/\sum_{s=1}^T \lambda_1^{(s)}$, and let $\bar{\lambda} := \left(\sum_{t=1}^T \lambda^{(t)}\right)/T_\lambda$. Then $\bar{\theta}$ is nearly-optimal in expectation:*

$$\mathbb{E}_{\bar{\theta},x\sim\mathcal{D}}\left[\ell_0\left(x;\bar{\theta}\right)\right] \leq \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0\left(x;\theta^*\right)\right] + \frac{1}{\bar{\lambda}_1}\left(2\epsilon + 2\tilde{G}^{(\mathrm{trn})}(\Theta) + G^{(\mathrm{val})}(\hat{\Theta})\right)$$

where θ^* minimizes $\mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0(x;\cdot)\right]$ subject to the proxy-constraints $\mathbb{E}_{x\sim\mathcal{D}}\left[\tilde{\ell}_i(x;\theta^*)\right] \leq 0$. It is also nearly-feasible:

$$\max_{i\in[m]} \mathbb{E}_{\bar{\theta},x\sim\mathcal{D}}\left[\ell_i\left(x;\bar{\theta}\right)\right] \leq \frac{\epsilon}{\bar{\lambda}_1} + G^{(\mathrm{val})}(\hat{\Theta})$$

Additionally, if there exists a $\theta' \in \Theta$ that satisfies all of the

Table 1. Simplified comparison of our suboptimality and infeasibility bounds (Theorems 1 and 2) to those for the Lagrangian formulation trained only on $S^{(\mathrm{trn})}$. The "one-dataset" row is the result of an analysis of this one-dataset Lagrangian approach (essentially the same algorithm as Agarwal et al. (2018)—see Appendix F for details). The "two-dataset" row contains the results for our algorithms, which use the proxy-Lagrangian formulation on two independent datasets. In both cases, ε measures how far the sequence of iterates is from being the appropriate type of equilibrium ($\varepsilon = \rho + 2\epsilon$ for Theorem 1, and $\varepsilon = 2\epsilon$ for Theorem 2). The big-Os absorb only constants which are properties of the constrained problem (Equation 1) and choice of proxy-constraint losses: γ and B_{ℓ_0}. The key difference is in the "Infeasibility" column: our proposal depends on $G^{(\mathrm{val})}(\hat{\Theta})$—which we bound independently of the model complexity (see Table 2)—rather than $\tilde{G}^{(\mathrm{trn})}(\Theta)$.

# Datasets	Suboptimality	Infeasibility	Assuming
One	$O\left(\varepsilon + \tilde{G}^{(\mathrm{trn})}(\Theta)\right)$	$O\left(\varepsilon + \tilde{G}^{(\mathrm{trn})}(\Theta)\right)$	$\tilde{G}^{(\mathrm{trn})}(\Theta) \leq \gamma/2^{\ddagger}$
Two	$O\left(\varepsilon + \tilde{G}^{(\mathrm{trn})}(\Theta) + G^{(\mathrm{val})}(\hat{\Theta})\right)$	$O\left(\varepsilon + G^{(\mathrm{val})}(\hat{\Theta})\right)$	$\varepsilon + 2\tilde{G}^{(\mathrm{trn})}(\Theta) + G^{(\mathrm{val})}(\hat{\Theta}) \leq \gamma/2$

Table 2. Comparison of the standard Rademacher complexity-based generalization bound (of the function class $\mathcal{F}$ parameterized by Θ) to our bounds on $G^{(\mathrm{val})}(\hat{\Theta})$. All bounds hold with probability $1-\delta$, and we assume that $\left|S^{(\mathrm{trn})}\right| \propto n$ and $\left|S^{(\mathrm{val})}\right| \propto n$ (e.g. if the data is split 50/50). The big-Os absorb only constants which are properties of the constrained problem (Equation 1) and choice of proxy-constraint losses: B_ℓ, $B_{\tilde{\ell}}$, L and μ. For our algorithms, the validation generalization performance of the constraints is *independent* of the model complexity.

$\tilde{G}^{(\mathrm{trn})}(\Theta)$	$G^{(\mathrm{val})}(\hat{\Theta})$ **(Theorem 1)**	$G^{(\mathrm{val})}(\hat{\Theta})$ **(Theorem 2)**
$O\left(R_n\left(\mathcal{F}\right) + \sqrt{\frac{\ln(1/\delta)}{n}}\right)$	$O\left(\sqrt{\frac{m\ln(1/\epsilon)+\ln(m/\delta)}{n}}\right)$	$O\left(\sqrt{\frac{m\ln n+\ln(m/\delta)}{n}} + \epsilon\right)$

constraints with margin γ (i.e. $\mathbb{E}_{x\sim\mathcal{D}}\left[\ell_i\left(x;\theta'\right)\right] \leq -\gamma$ for all $i \in [m]$), then:

$$\bar{\lambda}_1 \geq \frac{1}{\gamma + B_{\ell_0}}\left(\gamma - 2\epsilon - 2\tilde{G}^{(\mathrm{trn})}(\Theta) - G^{(\mathrm{val})}(\hat{\Theta})\right)$$

where B_{ℓ_0} is as in Theorem 1.

Generalization: *If, in addition to the above requirements, $\ell\left(x;\theta\right)$ is L-Lipschitz continuous in θ for all $\ell \in \{\ell_1, \ldots, \ell_m\}$, then with probability $1-\delta$ over the sampling of $S^{(\mathrm{val})}$:*

$$G^{(\mathrm{val})}(\hat{\Theta}) < B_\ell\sqrt{\frac{2m}{|S^{(\mathrm{val})}|}\max\left\{1, \ln\left(\frac{160L^2B_{\tilde{\ell}}|S^{(\mathrm{val})}|}{m\mu B_\ell^2}\right)\right\}} + B_\ell\sqrt{\frac{\ln\left(2m/\delta\right)}{2\left|S^{(\mathrm{val})}\right|}} + 2L\epsilon\sqrt{\frac{2}{\mu}} \tag{7}$$

where $B_{\tilde{\ell}}$ and B_ℓ are as in Theorem 1.

Proof. In Appendix E.2. □

The above optimality and feasibility guarantees are very similar to those of Theorem 1, as is shown in Table 1 (in which the only difference is the definition of ε). Algorithm 2's generalization bound (Equation 7) is more complicated than that of Algorithm 1 (Equation 6), but Table 2 shows that the two are roughly comparable. Hence, the overall theoretical performance of Algorithm 2 is very similar to that of Algorithm 1, and, while it does rely on stronger assumptions, it neither uses discretization, nor does it require an oracle.

5. Experiments

While Section 4 has demonstrated the theoretical performance of Algorithms 1 and 2, we believe that our proposed two-dataset approach is useful *as a heuristic* for improving constraint generalization performance, even when one is not using a theoretically-justified algorithm. For this reason, we experiment with two "practical" algorithms. The first, Algorithm 3[§], is a bare-bones version of Algorithm 2, in which θ and λ are updated simultaneously using stochastic gradients, instead of in an inner and outer loop. This algorithm implements our central idea—imposing constraints using an independent validation dataset—without compromising on simplicity or speed. The purpose of the second, Algorithm 4[§], is to explore how well our two-dataset idea can be applied to the usual Lagrangian formulation (similarly to Agarwal et al. (2018)). This algorithm simply "tacks on" proxy-constraints and the use of two independent datasets to the Lagrangian game. Neither algorithm enjoys the theoretical guarantees of Section 4, but, as we will see, both succeed at improving constraint generalization.

In our experiments, each dataset is split into thee parts:

[§] In the supplementary material.

Algorithm 2 Finds an approximate equilibrium of the empirical proxy-Lagrangian game (Definition 1) assuming that $\ell(x;\theta)$ is μ-strongly convex in θ for all $\ell \in \left\{\ell_0, \tilde{\ell}_1, \ldots, \tilde{\ell}_m\right\}$ (the objective and proxy constraint losses, but *not* the original constraint losses). Theorem 2 is its convergence and generalization guarantee. The θ-player uses gradient descent, while the λ-player uses the same swap-regret minimizing procedure as Algorithm 1.

ContinuousTwoDataset $\left(\hat{\mathcal{L}}_\theta, \hat{\mathcal{L}}_\lambda : \Theta \times \Delta^m \to \mathbb{R}, T_\theta, T_\lambda \in \mathbb{N}, \mu, \eta_\lambda \in \mathbb{R}_+\right)$:

Initialize $M^{(1)} \in \mathbb{R}^{(m+1)\times(m+1)}$ with $M_{i,j} = 1/(m+1)$

For $t \in [T_\lambda]$:

  Let $\lambda^{(t)} = \text{fix}\, M^{(t)}$ *// fixed point of $M^{(t)}$, i.e. a stationary distribution*

  For $s \in [T_\theta]$:

    Initialize $\tilde{\theta}^{(t,1)} = 0$ *// Assumes $0 \in \Theta$*

    Let $\check{\Delta}_\theta^{(t,s)}$ be a subgradient of $\hat{\mathcal{L}}_\theta\left(\tilde{\theta}^{(t,s)}, \lambda^{(t)}\right)$ w.r.t. θ

    Update $\tilde{\theta}^{(t,s+1)} = \Pi_\Theta\left(\tilde{\theta}^{(t,s)} - \check{\Delta}_\theta^{(t,s)}/\mu s\right)$

  Define $\theta^{(t)} := \left(\sum_{s=1}^{T_\theta} \tilde{\theta}^{(t,s)}\right)/T_\theta$

  Let $\Delta_\lambda^{(t)}$ be a gradient of $\hat{\mathcal{L}}_\lambda\left(\theta^{(t)}, \lambda^{(t)}\right)$ w.r.t. λ

  Update $\tilde{M}^{(t+1)} = M^{(t)} \odot .\exp\left(\eta_\lambda \Delta_\lambda^{(t)}\left(\lambda^{(t)}\right)^T\right)$ *// $\Delta\lambda^T$ is an outer product; $\odot$ and $.\exp$ are element-wise*

  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 w.r.t. KL divergence*

Return $\theta^{(1)}, \ldots, \theta^{(T)}$ and $\lambda^{(1)}, \ldots, \lambda^{(T)}$

training, validation and testing. We compare our proposed two-dataset approach, in which $S^{(\text{trn})}$ is the training dataset and $S^{(\text{val})}$ is the validation dataset, to the the natural baseline one-dataset approach of using the *union* of the training and validation sets to define *both* $S^{(\text{trn})}$ and $S^{(\text{val})}$. Hence, all approaches "see" the same total amount of data.

This difference between the data provided to the two algorithms leads to a slight complication when reporting "training" error rates and constraint violations. For the two-dataset approach, the former are reported on $S^{(\text{trn})}$ (used to learn θ), and the latter on $S^{(\text{val})}$ (used to learn λ). For the baseline one-dataset algorithm, both are reported on the full dataset (i.e. the union of the training and validation sets). "Testing" numbers are always reported on the testing dataset.

Our implementation uses TensorFlow, and is based on Cotter et al. (2019)'s open-source constrained optimization library. To avoid a hyperparameter search, we replace the stochastic gradient updates of Algorithms 3 and 4 with ADAM (Kingma & Ba, 2014), using the default parameters. For both our two-dataset algorithm and the one-dataset baseline, the result of training is a sequence of iterates $\theta^{(1)}, \ldots, \theta^{(T)}$, but instead of keeping track of the full sequence, we only store a total of 100 evenly-spaced iterates for each run. Rather than using the weighted predictor of Theorems 1 and 2, we use the "shrinking" procedure of Cotter et al. (2019) (see Appendix C) to find the *best* stochastic classifier supported on the sequence of 100 iterates.

In all experiments, the objective and proxy constraint functions $\ell_0, \tilde{\ell}_1, \ldots, \tilde{\ell}_m$ are hinge upper bounds on the quantities of interest, while the *original* functions $\ell_1, \ldots, \ell_m$ are precisely what we claim to constrain (in these experiments, proportions, i.e. linear combinations of indicators).

5.1. Datasets and Results

Our datasets are summarized in in Table 3, and described below. On each dataset, we trained one of three different types of models: linear, RTL (Canini et al., 2016), or a neural network with one hidden layer containing 50 ReLU neurons. The neural network models are more complex than necessary for these datasets, and are used here *because* they overfit, and therefore illustrate the improved generalization performance of the two-dataset approach.

Communities and Crime: This UCI dataset (Dheeru & Karra Taniskidou, 2017) includes features aggregated from census and law enforcement data, on which we train a linear model. The binary classification task is to predict whether a community has a high (above the 70th percentile) or low crime rate, as in Kearns et al. (2017). To form protected groups, we use four racial percentage features as real-valued protected attributes. Each is thresholded at the 50th percentile to form eight protected groups: low-Asian, high-Asian, low-Black, high-Black, low-Hispanic, high-Hispanic, low-White, and high-White. There are eight fairness constraints, which constrain each protected group's false positive rate to be no larger than that of the full dataset.

Business Entity Resolution: This is a proprietary dataset from Google for which the task is to predict whether a pair of business descriptions describe the same real business.

Table 3. Properties of datasets used in Section 5.1. For the one-dataset experiments, the entire training set was used for training. For the two-dataset experiments, the training dataset was split in half between $S^{(\mathrm{trn})}$ and $S^{(\mathrm{val})}$.

Dataset	Model	Training examples	Testing examples	Features
Communities and Crime	Linear	1 459	499	140
Business Entity Resolution	RTL	11 560	3 856	37
Adult	Neural Network	32 561	16 281	122
COMPAS	Neural Network	4 110	2 026	32

Table 4. Error rates and maximum constraint violations for all compared algorithms, on the datasets of Table 3, as described in Section 5.1. The "Unconstrained" columns contain the results for entirely-unconstrained models. All quantities are averaged over 100 runs. The training constraint violations are occasionally exactly zero thanks to our use of Cotter et al. (2019)'s "shrinking" procedure (Appendix C).

				Algorithm 3				**Algorithm 4**			
		Unconstrained		One-dataset		Two-dataset		One-dataset		Two-dataset	
Dataset		Error	Viol.	Error	Viol.	Error	Viol.	Error	Viol.	Error	Viol.
Communities and Crime	Train	.121	.231	.153	0	.161	0	.163	−.001	.165	0
	Test	.142	.300	.173	.022	.199	−.008	.181	.001	.195	−.012
Entity Resolution	Train	.148	.309	.216	.026	.215	.040	.225	0	.261	.003
	Test	.156	.278	.222	.073	.221	.072	.232	.042	.267	.041
Adult	Train	.102	.077	.132	0	.110	0	.131	0	.113	0
	Test	.156	.075	.156	.011	.169	.005	.156	.013	.165	.008
COMPAS	Train	.216	.004	.216	−.005	.154	−.003	.216	−.005	.151	−.003
	Test	.353	.046	.353	.038	.378	.004	.349	.029	.378	.006

Features include measures of similarity between the two business titles, phone numbers, and so on. We impose several constraints: (i) for each of the 16 most common countries, the recall must be at least 95%; (ii) for the set of all chain businesses, and likewise for the set of all non-chain businesses, the recall must be at least 95%; (iii) the accuracy on non-chain businesses must be no more than 10% higher then that on chain businesses. The purpose of this final constraint is to cause small and large business to be treated roughly comparably.

Adult: This is a version of the UCI Adult dataset, preprocessed to include only binary features (using one-hot encodings for categorical features, and bucketing for continuous features). The classification task is to predict whether a person's yearly income is greater than \$50 000, subject to the 80% rule for demographic parity: for each of four overlapping protected classes (Black, White, Female and Male), the positive prediction rate must be at least 80% of the overall positive prediction rate.

COMPAS: This is the ProPublica COMPAS dataset analyzed by Angwin et al. (2016), preprocessed similarly to the Adult dataset. The classification task is to predict recidivism, subject to equal opportunity (Hardt et al., 2016) fairness constraints: for each of four overlapping protected classes (Black, White, Female and Male), the positive prediction rate on the positively-labeled examples must be at most 5% higher than the overall positive prediction rate on positively-labeled examples.

These datasets all have designated training/testing splits. For the one-dataset (baseline) experiments, both $S^{(\mathrm{trn})}$ and $S^{(\mathrm{val})}$ were taken to be the entire training set. For the two-dataset experiments, the training set was randomly permuted and split in half, into $S^{(\mathrm{trn})}$ and $S^{(\mathrm{val})}$. All reported numbers are averaged over 100 such random splits.

Table 4 summarizes the results of these experiments. Except on the Business Entity Resolution dataset, the two-dataset experiments have a clear and significant advantage in terms of constraint generalization performance, although it comes at a cost: the error rates are, as expected, somewhat higher. Additional experiments, performed on a simulated dataset, can be found in Appendix D, and tell the same story.

On the Business Entity Resolution dataset, the testing constraint violations of Algorithms 3 and 4 are comparable to those of the baselines, but the error rates were slightly worse. On this dataset, it appears that the trade-off between (i) improved constraint generalization and (ii) each player seeing only half of the training data, wasn't beneficial overall.

While one must be mindful of this trade-off, these results show that providing independent datasets to the θ- and λ-players tends to improve constraint generalization in practice, *even when our proofs do not apply*.

Acknowledgments

We gratefully acknowledge Phil Long, who was a valuable sounding-board in the early stages of this project, and Harikrishna Narasimhan, who provided many helpful comments and suggestions on the penultimate manuscript.

References

Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J., and Wallach, H. M. A reductions approach to fair classification. In *ICML*, pp. 60–69, 2018.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks, 2016.

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

Bartlett, P. Slides for lecture 12 of stat 210b: Theoretical statistics, February 2013. URL https://www.stat.berkeley.edu/~bartlett/courses/2013spring-stat210b/notes/12notes.pdf.

Biddle, D. *Adverse Impact and Test Validation: A Practitioner's Guide to Valid and Defensible Employment Testing*. Gower, 2005.

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

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

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

Cotter, A., Jiang, H., and Sridharan, K. Two-player games for efficient non-convex constrained optimization. In *ALT*, 2019.

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

Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J., and Pontil, M. Empirical risk minimization under fairness constraints, 2018. URL https://arxiv.org/abs/1802.08626.

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

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

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

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

Kearns, M., Neel, S., Roth, A., and Wu, Z. S. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness, 2017. URL https://arxiv.org/abs/1711.05144.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2014.

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

Ng, A. Y. Preventing "overfitting" of cross-validation data. In *ICML*, pp. 245–253, 1997.

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

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

Shalev-Shwartz, S., Singer, Y., Srebro, N., and Cotter, A. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *Mathematical Programming*, 127(1):3–30, March 2011.

Srebro, N. Slides for lecture 9 of TTIC 31120: Computational and statistical learning theory, October 2016. URL http://ttic.uchicago.edu/~nati/Teaching/TTIC31120/2016/Lecture9.pdf.

Vuolo, M. S. and Levy, N. B. Disparate impact doctrine in fair housing. *New York Law Journal*, 2013.

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

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