Training Fairness-Constrained Classifiers to Generalize

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 performance, 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 improved.

1. Introduction

It is useful to train classifiers with data-dependent constraints in order to achieve certain guarantees, such as statistical parity or other fairness metrics, specified recall, or a desired positive classification rate [e.g. 16, 19, 7, 18, 13]). 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 a 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 [3, 17, 19, 9]). 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 $\mathcal{D}$:

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

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 loss functions associated with the objective and the m constraints (which are *not* required to be convex).

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 prior work that treats constrained optimization as a two-player game [e.g. 2, 5, 15, 1, 10, 13]. In this setting, the first player optimizes the model parameters θ, and the second player enforces the constraints, e.g. using the Lagrangian:

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

In practice, one would approximate the Lagrangian with a finite *i.i.d.* training sample from $\mathcal{D}$, and the first player would minimize over the model parameters $\theta \in \Theta$ while the second 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. 14] 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

[1]Google AI [2]Toyota Technological Institute at Chicago [3]Cornell University [4]Kakao Mobility. Correspondence to: Andrew Cotter <acotter@google.com>.

5^{th} *Workshop on Fairness, Accountability, and Transparency in Machine Learning*, Stockholm, Sweden, 2018.

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).

While the above approach is intuitive, 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, θ is minimizing a *different function* than λ is maximizing, so the corresponding two-player game is *non-zero-sum* (the players have different cost functions). To handle this, we must depart from the typical Lagrangian formulation, but the key idea remains: improving generalization by using a separate validation set to enforce the constraints.

The recent work of Cotter et al. [6] gives a strategy for dealing with such a non-zero-sum game when optimizing a constrained machine learning problem. We adapt their theory to this new setting to give a bound on the generalization of the constraint violations that is agnostic to model complexity (Section 3), showing that it is possible significantly improve the generalization performance of constraints, at the cost of some of the optimization and generalization performance of the objective function. In Section 4, we present a case study showing that we can achieve better constraint generalization. The full version of our paper [1] contains additional algorithms and experiments.

This paper also extends the work of Woodworth et al. [18] on improving generalization of fairness metrics. They also used a separate validation set, but in a more restrictive setting: given a fixed classifier learned on a training set, they proposed adding model parameters (such as an additive bias per sensitive class, as in Hardt et al. [9]) suitable for satisfying the fairness metric(s) of interest, and then learning these new parameters on the validation set.

2. Preliminaries

Our algorithm is based on the non-zero sum two-player game proposed by Cotter et al. [6], which they call the "proxy-Lagrangian" formulation. Like them, to cope with the practical difficulties of satisfying non-differentiable constraints (e.g. a constraint on the positive classification rate, which is a sum of indicators), we permit the approximation of each of the constraint losses ℓ_i with a (presumably differentiable) upper-bound $\tilde{\ell}_i$, called a *proxy constraint loss*. These are used *only* by the θ-player—the λ-player uses the original constraint losses. In words, the λ-player will attempt to satisfy the *original* constraints by choosing appropriate penalties on the *proxy* constraints.

Definition 1. *Let $S^{(\text{train})}$ and $S^{(\text{val})}$ 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:*

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

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

where $\Lambda := \Delta^{m+1}$ is the $(m+1)$-dimensional simplex.

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. The difference between the above, and Definition 2 of Cotter et al. [6], is that $\hat{\mathcal{L}}_\theta$ is an empirical average over the *training* set, while $\hat{\mathcal{L}}_\lambda$ is over the *validation* set.

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 $\tilde{G}^{(\text{train})}(\Theta)$ such that:*

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

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 original constraints).

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

Instead of finding a single $\theta \in \hat{\Theta}$, we will find a *randomized model* $\bar{\theta}$ supported on $\hat{\Theta}$ (i.e. a distribution over $\hat{\Theta}$), but the above definition still applies: by the triangle inequality, if every $\theta \in \hat{\Theta}$ generalizes well, then any such $\bar{\theta}$ generalizes equally well, in expectation.

3. Algorithm

We seek a solution that (i) is nearly-optimal, (ii) nearly-feasible, and (iii) generalizes well on the validation set. The optimality and feasibility goals were already tackled by Cotter et al. [6] in the context of the proxy-Lagrangian formulation of Definition 1. They proposed having the θ-player minimize ordinary external regret, while the λ-player minimized swap regret using an algorithm based on Gordon et al. [8]. 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 challenge (iii): generalization.

The simplest way to attack the generalization problem is to *discretize* the space of allowed λs, and associate each λ

[1] https://arxiv.org/abs/1807.00028.

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. [6]—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 $\Lambda := \Delta^{m+1}$ w.r.t. the 1-norm. The θ-player uses oracle calls (Definition 3) 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. [8], using the left-stochastic state matrices $M^{(t)} \in \mathbb{R}^{(m+1)\times(m+1)}$.

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

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

For $t \in [T]$:

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

  Let $\tilde{\lambda}^{(t)} = \operatorname{argmin}_{\tilde{\lambda} \in C_r} \left\| \lambda^{(t)} - \tilde{\lambda} \right\|_1$ *// 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 $\tilde{M}^{(t+1)} = M^{(t)} \odot . \exp\left(\eta_\lambda \hat{\Delta}_\lambda^{(t)} \left(\lambda^{(t)}\right)^T\right)$ *// $\odot$ and $.\exp$ are element-wise*

  Project $M_{:,i}^{(t+1)} = \tilde{M}_{:,i}^{(t+1)} / \left\| \tilde{M}_{:,i}^{(t+1)} \right\|_1$ for $i \in [m+1]$

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

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 $\Lambda := \Delta^{m+1}$ w.r.t. the 1-norm. The set of allowed λs is exactly the covering centers, while, following Chen et al. [4] and Cotter et al. [6], the associated θs are found using an approximate Bayesian optimization oracle:

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

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

for any $f : \Theta \to \mathbb{R}$ that can be written as $f(\theta) = \hat{\mathcal{L}}_\theta(\theta, \lambda)$ for some λ. Furthermore, every time it is given the same f, $\mathcal{O}_\rho$ will return the same θ (i.e. it is deterministic).

Algorithm 1 combines our proposed discretization with the oracle-based proxy-Lagrangian optimization procedure proposed by Cotter et al. [6]. As desired, it finds a sequence of solutions $\hat{\Theta} := \left\{\theta^{(1)}, \ldots, \theta^{(T)}\right\}$ for which it is possible to bound $G^{(\text{val})}(\hat{\Theta})$ *independently* of the complexity of the function class parameterized by Θ and the size of $S^{(\text{train})}$, *and* finds a random parameter vector $\bar{\theta}$ supported on $\hat{\Theta}$ that is nearly-optimal and nearly-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:*

$$\mathbb{E}_{\bar{\theta}, x\sim\mathcal{D}}\left[\ell_0\left(x;\bar{\theta}\right)\right] \leq \inf_{\theta^* \in \Theta : \forall i. \mathbb{E}_{x\sim\mathcal{D}}\left[\tilde{\ell}_i(x;\theta^*)\right] \leq 0} \mathbb{E}_{x\sim\mathcal{D}}\left[\ell_0\left(x;\theta^*\right)\right] + \frac{1}{\bar{\lambda}_1}\left(\rho + 2\epsilon + 2\tilde{G}^{(\text{train})}(\Theta) + 2G^{(\text{val})}(\hat{\Theta})\right) \tag{3}$$

and nearly-feasible:

$$\max_{i\in[m]} \mathbb{E}_{\bar{\theta}, x\sim\mathcal{D}}\left[\ell_i\left(x;\bar{\theta}\right)\right] \leq \frac{1}{\bar{\lambda}_1}\left(\epsilon + 2G^{(\text{val})}(\hat{\Theta})\right) \tag{4}$$

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

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

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

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

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

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* $[a_i, b_i]$.

The first two results of this theorem (Equations 3 and 4) are better for larger $\bar{\lambda}_1$, where $\bar{\lambda}_1$ can be interpreted as the "weight" on the objective loss, averaged over iterates. Just as 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.

Equation 5, unfortunately, depends on $\tilde{G}^{(\text{train})}(\Theta)$. Thankfully, this dependence is gated by whether the feasibility margin is satisfied. In particular, the feasibility guarantee (Equation 4, or the "Infeasibility" column) will depend on $\tilde{G}^{(\text{train})}(\Theta)$ only if the training and validation datasets do not generalize well enough for $\rho + 2\epsilon + 2\tilde{G}^{(\text{train})}(\Theta) + 2G^{(\text{val})}(\hat{\Theta})$ to stay within the feasibility margin γ. Past this critical threshold, $\bar{\lambda}_1$ can be lower-bounded by a constant, and *only* the validation generalization error $G^{(\text{val})}(\hat{\Theta})$—which Equation 6 bounds *independently* of the complexity of Θ—will matter. In practice, of course, one need not rely on this lower bound on $\bar{\lambda}_1$: one can instead simply inspect the behavior of the sequence of $\lambda_1^{(t)}$'s during optimization.

4. Fairness Case Study

While Algorithm 1 performs well theoretically, it contains some elements that were needed for the analysis, but could be needlessly expensive in practice. Hence, our case study is performed on a "bare-bones" version of Algorithm 1, in which the discretization on lines 3-4 is removed entirely, and the oracle optimization on line 5 and the multiplicative update of lines 7-8 are each replaced with a single iteration of ADAM [11]. This algorithm implements our central idea—imposing constraints using an independent validation dataset—without compromising on simplicity or speed. It does not enjoy the theoretical guarantee of Theorem 1, but, as we will see, still improves constraint generalization.

Our case study uses the UCI Communities and Crime dataset [12], which includes $d = 122$ continuous 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. [10].

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-white, high-white, low-asian, high-asian, low-black, high-black, low-hispanic, and high-hispanic. There is one fairness constraint for each of the eight protected groups, which constrains the group's false positive rate to be at most the overall false positive rate. To convert this to the form of Equation 1, we took the objective and proxy constraint functions $\ell_0, \tilde{\ell}_1, \ldots, \tilde{\ell}_m$ to be hinge upper bounds on the quantities of interest, with the *original* constraint functions $\ell_1, \ldots, \ell_m$ being just what we claim to constrain (false positive rates, represented as linear combinations of indicators).

Table 1. Results for case study of Section 4. The "Violation" columns show the maximum constraint violation over all protected groups, where each group's violation is the difference (group FPR − overall FPR). In the first two rows, the constraints are exactly satisfied on the training set thanks to our use of "Shrinking" [6].

	Training		Testing	
	Error	Violation	Error	Violation
Ours	0.149	0	0.212	0.008
Baseline	0.147	0	0.188	0.016
Unconstrained	0.102	0.083	0.151	0.125

The data was split equally into thee parts: training, validation and testing. We compare our proposed approach, in which the model parameters θ are learned on the training set, and λ on the validation set, to the natural baseline approach of using the *union* of the training and validation sets for learning both θ and λ. Hence, both approaches "see" the same amount of data during training.

For both our algorithm and the baseline, the result of training is a sequence of iterates $\theta^{(1)}, \ldots, \theta^{(T)}$. Rather than using the weighted predictor of Theorem 1, we instead use the "shrinking" procedure of Cotter et al. [6] to find the *best* stochastic classifier supported on the sequence of iterates.

All reported numbers are averaged over ten runs, with different random splits of the data. The difference between the data provided to the two algorithms leads to a slight complication when reporting "training" error rates and constraint violations. For our algorithm, the former are reported on the training set (used to learn θ), and the latter on the validation set (used to learn λ). For the baseline, both are reported on the union of the training and validation sets. "Testing" numbers are always reported on the testing dataset.

The results of this experiment can be found in Table 1, and closely match with our expectations: on testing data, our approach halves the maximum constraint violation compared to the baseline, but suffers from about 13% higher testing error. Unsurprisingly, an unconstrained classifier performs much better than both in terms of testing accuracy, but much worse in terms of constraint violations.

References

[1] Agarwal, Alekh, Beygelzimer, Alina, Dudík, Miroslav, Langford, John, and Wallach, Hanna. A reductions approach to fair classification, 2018. URL https://arxiv.org/abs/1803.02453.

[2] Arora, Sanjeev, Hazan, Elad, and Kale, Satyen. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8 (6):121–164, 2012.

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

[4] Chen, Robert S, Lucier, Brendan, Singer, Yaron, and Syrgkanis, Vasilis. Robust optimization for non-convex objectives. In *Nips'17*, 2017.

[5] Christiano, Paul, Kelner, Jonathan A., Madry, Aleksander, Spielman, Daniel A., and Teng, Shang-Hua. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In *STOC '11*, pp. 273–282, 2011.

[6] Cotter, Andrew, Jiang, Heinrich, and Sridharan, Karthik. Two-player games for efficient non-convex constrained optimization, 2018. URL https://arxiv.org/abs/1804.06500.

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

[8] Gordon, Geoffrey J., Greenwald, Amy, and Marks, Casey. No-regret learning in convex games. In *ICML'08*, pp. 360–367, 2008.

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

[10] Kearns, Michael, Neel, Seth, Roth, Aaron, and Wu, Zhiwei Steven. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness, 2017. URL https://arxiv.org/abs/1711.05144.

[11] Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *ICLR'14*, 2014.

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

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

[14] Ng, Andrew Y. Preventing "overfitting" of cross-validation data. In *ICML'97*, pp. 245–253, 1997.

[15] Rakhlin, Alexander and Sridharan, Karthik. Optimization, learning, and games with predictable sequences. In *NIPS'13*, pp. 3066–3074, 2013.

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

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

[18] Woodworth, Blake E., Gunasekar, Suriya, Ohannessian, Mesrob I., and Srebro, Nathan. Learning non-discriminatory predictors. In *COLT'17*, pp. 1920–1953, 2017.

[19] 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.