Making Fair Classifiers Robust

Abstract

We initiate the study of fair classifiers that are robust to perturbations in the training distribution. Despite recent progress, the literature on fairness has largely ignored the design of fair and robust classifiers. In this work, we develop classifiers that are fair not only with respect to the training distribution but a class of distributions that are weighted perturbations of the training samples. We set up a min-max objective function whose goal is to minimize a distributionally robust training loss, and at the same time, find a classifier that is fair with respect to a class of distributions. We first reduce this problem to finding a fair classifier that is robust with respect to the class of distributions. Based on a classical online learning algorithm, we develop an iterative algorithm that provably converges to such a fair and robust solution. Along the way, we also show that the number of iterations of this algorithm can be made sub-linear in number of training points, improving upon several prior work in the design of fair classifiers. Experiments on standard machine learning fairness datasets suggest that our classifier retains fairness guarantees and test accuracy competitive to existing fair classifiers with respect to a large class of perturbations on the test set.

1 Introduction

Machine learning systems are increasingly used in various high-stakes decision making scenarios, including bail decision, credit approval, and housing allocation, to name a few. Often these applications use learning algorithms trained on past biased data, and such bias is often reflected in the eventual decision. For example, Bolukbasi et al. [2016] show that popular word embeddings implicitly encode societal biases, such as gender norms. Similarly, Buolamwini and Gebru [2018] evaluate existing facial recognition systems and find that they perform better on lighter-skinned subjects than on darker-skinned subjects. To mitigate these biases, there have been several approaches in the ML fairness community to design fair classifiers Zemel et al. [2013], Hardt et al. [2016], Agarwal et al. [2018].

However, the literature has largely ignored the robustness of such fair classifiers. As an example, we consider the performance of the optimized pre-processing algorithm Calmon et al. [2017] on the popular COMPAS dataset COM. As a metric of fairness we consider the notion of *demographic parity* (DP), which measures the difference in accuracy between two protected groups. Figure 1 shows two situations – (1) unweighted training distribution (in blue), and (2) weighted training distributions. The optimized pre-processing algorithm Calmon et al. [2017] is almost fair on the unweighted training dataset (DP ≤ 0.02). However, it shows demographic parity of at least 0.1 on the weighted dataset, despite the fact that the marginal distributions of the features look almost the same for the two scenarios. This example motivates our work and we aim to design a fair classifier that is robust to such perturbations. We also show how to construct such reweighted examples.

Nonetheless, since different algorithms adopt different definitions of fairness and provide different trade-offs with respect to accuracy and utility, it is neither legal nor ethical to enforce businesses to use such algorithms. In this paper, we approach this problem with a perspective from the literature of automated verification, and aim to build tools that can verify whether an algorithm satisfies a given fairness criteria irrespective of the particular algorithm or dataset used. We show using these tools

Preprint. Under review.

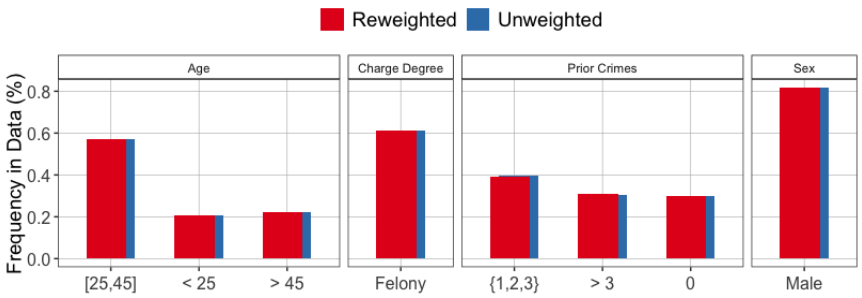

Figure 1: Unweighted vs Reweighted COMPAS dataset. Although the marginals of the two distributions are almost the same, standard fair classifiers show demographic parity of at least 0.1 on the reweighted dataset.

that, although current group fairness algorithms may mitigate fairness for a specific distribution of data, slight perturbations to that data's distribution result in violations of the fairness criteria.

Contributions

Technical Overview

Experimental Findings

Related Work

2 Problem and Definitions

We will write $((x, a), y)$ to denote a training instance where $x \in \mathcal{X}$ denotes unprotected attributes, $a \in \mathcal{A}$ denotes the protected attributes, and $y \in \{0, 1\}$ denotes the outcome label. For a hypothesis h, $h(x, a) \in \{0, 1\}$ denotes the outcome predicted by it. We assume that the set of hypothesis is given by a class $\mathcal{H}$. Given a loss function $\ell : \{0, 1\} \times \{0, 1\} \to \mathbb{R}$, the goal of a standard fair classifier is to find a hypothesis $h^* \in \mathcal{H}$ that minimizes the training loss $\sum_{i=1}^{n} \ell(h(x_i, a_i), y_i)$ and is also fair according to some standard fairness criteria.

In this paper, our goal is different and we wish to find a hypothesis that is fair and robust. We will be designing classifiers that are robust with respect to distributions that are weighted perturbations of the original training distribution. For this, let $\mathcal{W}$ be the set of all possible weights i.e. $\mathcal{W} = \{w \in \mathbb{R}_n^+ : \sum_i w_i = 1\}$. For a hypothesis h and weight w, we define the following loss function $\ell(h, w) = \sum_{i=1}^{n} w_i \ell(h(x_i, a_i), y_i)$. We will write $\delta_F^w(f)$ to define the "unfairness gap" with respect to the weighted empirical distribution defined by the weight w and fairness constraint F e.g. demographic parity(DP), equalized odds (EO). For example, $\delta_{DP}^w(f)$ is defined as

$$\delta_{DP}^w(f) = \left| \frac{\sum_{i:a_i=a} w_i f(x_i, a)}{\sum_{i:a_i=a} w_i} - \frac{\sum_{i:a_i=a'} w_i f(x_i, a')}{\sum_{i:a_i=a'} w_i} \right|. \tag{1}$$

Therefore, $\delta_{DP}^w(f)$ measures the weighted difference in accuracy between the two groups with respect to the training distribution that assigns weight w to the training examples. On the other hand, $\delta_{EO}^w(f) = \max_{y \in \{0,1\}} \delta_{EO}^w(f|y)$, where $\delta_{EO}^w(f|y)$ is defined as

$$\delta_{EO}^w(f|y) = \left| \frac{\sum_{i:a_i=a, y_i=y} w_i f(x_i, a)}{\sum_{i:a_i=a, y_i=y} w_i} - \frac{\sum_{i:a_i=a', y_i=y} w_i f(x_i, a')}{\sum_{i:a_i=a', y_i=y} w_i} \right|.$$

Therefore, $\delta_{EO}^w(f|0)$ (resp. $\delta_{EO}^w(f|1)$) measures the weighted difference in false (resp. true) positive rates between the two groups with respect to the weight w. We will be mainly working with the notion of weighted demographic parity for developing the theory behind our fair and robust classifiers. However, we will provide experimental results concerning both weighted demographic parity, and equalized odds.

We are now ready to formally define our main objective. For a class of hypothesis $\mathcal{H}$, let $\mathcal{H}_{\mathcal{W}} = \{h \in \mathcal{H} : \delta_F^w(h) \leq \epsilon \; \forall w \in \mathcal{W}\}$ be the set of hypothesis that are ϵ-fair with respect to all the weights

in the set $\mathcal{W}$. Our goal is to solve the following minmax problem:

$$\min_{h \in \mathcal{H}_{\mathcal{W}}} \max_{w \in \mathcal{W}} \ell(h, w) \tag{2}$$

Therefore, we aim to minimize a robust loss with respect to a class of distributions that are perturbations of the training distribution. Additionally, we also aim to find a classifier that is fair with respect to such perturbations. For our experiments, we will also consider a simplified version of the problem:

$$\min_{h \in \mathcal{H}_{\mathcal{W}}} \ell(h, \vec{u}) \tag{3}$$

where $\vec{u}$ is a distribution that assigns uniform weights to all the training points. Contrary to equation 2, the goal here is to just find a classifier that works well with the original training distribution but is fair with respect to all the weighted perturbations.

We will allow our algorithm to output a classifier which is randomized i.e. it is a distribution over the hypothesis $\mathcal{H}$. This will also be necessary if the space $\mathcal{H}$ is non-convex or if the fairness constraints are such that the set of feasible hypothesis $\mathcal{H}_{\mathcal{W}}$ is non-convex. For a randomized classifier μ define the expected loss of μ as $\ell(\mu, w) = \sum_h \mu(h)\ell(h, w)$.

3 Design

We design a robust and fair classifier in a top-down fashion. First we design a meta algorithm that reduces the minmax problem described in equation 2 to a loss minimization problem with respect to a sequence of weight vectors. Then we show how we can design a fair classifier that performs well with respect a fixed weight vector $\vec{w}$ in terms of accuracy, but is fair with respect to the entire class of weights $\mathcal{W}$.

3.1 Meta Algorithm

We now provide a meta-algorithm that helps us design fair classifiers that are robust with respect to any distribution that are some weighted perturbations of the empirical distribution of the training data. The meta-algorithm repeatedly calls an oracle that solves the fair classification problem with respect to a given weighted empirical distribution.

ALGORITHM 1: Meta-Algorithm

Input: Training Set: $\{x_i, a_i, y_i\}_{i=1}^n$, set of weights: $\mathcal{W}$, hypothesis class $\mathcal{H}$, parameters T and η.
$w_0(i) = 1/n$ for all $i \in [n]$
$h_0 = \text{ApxFair}(w_0)$ /* Approximate solution of $\arg\min_{h \in \mathcal{H}_{\mathcal{W}}} \sum_{i=1}^n \ell(h(x_i, a_i), y_i)$. */
for *each* $t \in [T]$ **do**
  $w_t = w_{t-1} + \eta \nabla_w \ell(h_{t-1}, w_{t-1})$
  $w_t = \Pi_{\mathcal{W}}(w_t)$ /* Project w_t onto the set of weights $\mathcal{W}$. */
  $h_t = \text{ApxFair}(w_t)$ /* Approximate solution of $\min_{h \in \mathcal{H}_{\mathcal{W}}} \sum_{i=1}^n w_t(i)\ell(h(x_i, a_i), y_i)]$. */
end
Output: h_f: Uniform distribution over $\{h_0, h_1, \ldots, h_T\}$.

Algorithm 1 provides a meta algorithm to solve the min-max optimization problem defined in equation 2. The algorithm is based on ideas presented in Chen et al. [2017], which, given an α-approximate Bayesian oracle for distributions over loss functions, provides an α-approximate robust solution. The algorithm can be viewed as a two-player zero-sum game between the learner who picks the hypothesis h_t, and an adversary who picks the weight vector w_t. The adversary performs a projected gradient descent every step to compute the best response. On the other hand, the learner solves a fair classification problem to pick the best hypothesis which is fair with respect to the class of weights $\mathcal{W}$ and minimizes training loss with respect to the weight vector w_t. However, it is infeasible to compute an exact optima of the problem $\min_{h \in \mathcal{H}_{\mathcal{W}}} \sum_{i=1}^n w_t(i)\ell(h(x_i, a_i), y_i)]$. So the learner uses an approximate fair classifier $\text{ApxFair}(\cdot)$, which we define next.

Definition 1. *ApxFair*$(\cdot)$ *is an* α*-approximate fair classifier, if for any weight* $w \in \mathbb{R}_+^n$, *ApxFair*(w) *returns a hypothesis* $\widehat{h}$ *such that*

$$\sum_{i=1}^n w_i \ell(\widehat{h}(x_i, a_i), y_i) \leq \min_{h \in \mathcal{H}_{\mathcal{W}}} \sum_{i=1}^n w_i \ell(h(x_i, a_i), y_i) + \alpha.$$

Using the approximate Bayesian oracle, we have the following gurantee on the output of algorithm 1.

Theorem 1. *Suppose the loss function $\ell(\cdot,\cdot)$ is convex in its first argument and ApxFair$(\cdot)$ is an α-approximate fair classifier. Then the ensemble hypothesis h_f output by the meta-algorithm 1, satisfies the following:*

$$\max_{w\in\mathcal{W}} \mathbb{E}_{h\sim h_f}\left[\sum_{i=1}^{n} w_i \ell(h(x_i,a_i),y_i)\right] \leq \min_{h\in\mathcal{H}_\mathcal{W}} \max_{w\in\mathcal{W}} \ell(h,w) + \max_{w\in\mathcal{W}} \|w\|_2 \sqrt{\frac{2}{T}} + \alpha$$

The proof of this theorem uses ideas from Chen et al. [2017], except that we use an additive approximate oracle.

3.2 Approximate Fair Classifier

In this section, we develop an α-approximate fair and robust classifier. We follow the following three steps to develop such an algorithm, and we belive that this approach might be helpful for developing other robust classifiers.

1. Discretize the set of all possible weights $\mathcal{W}$, so that it is sufficient to develop an apporoximate fair classifier with respect to the set of discretized weights. In particular, if we discretize each weight upto an error ϵ, then developoing an α-approximate fair classifier with respect to the discretized weights gives $O(\alpha+\epsilon)$-fair classifier wrt the set $\mathcal{W}$.
2. Set up the problem of designing an approximate fair classifier wrt the set of discretized weights as a two-player zero-sum game. Here, the learner player chooses a hypothesis, whereas an adversary picks the most "unfair" weight in the set of discretized weights.
3. Design a learning algorithm for the learner's learning algorithm, and design an approximate solution to adversary's best response to the learner's chosen hypothesis.

We would like to point out that Agarwal et al. [2018] was the first to show that the design of a fair classifier can be formulated as a two-player zero-sum game (step 2). However, they only considered group-fairness constraints with respect to the training distribution. On ther other hand, we consider the design of robust and fair classifier and had to include an additional discretization step (1). Finally, the design of our learning algorithm and the best response oracle is significantly different than Agarwal et al. [2018], Kearns et al. [2017]. For the remainder of this subsection, assume that the meta algorithm (1) called the ApxFair$(\cdot)$ with a weight vector w^0 and our goal is to design a classifier that minimizes weighted loss with respect to the weight w^0, but is fair wrt the set of all weights $\mathcal{W}$ i.e. find $f \in \arg\min_{h\in\mathcal{H}_\mathcal{W}} \ell(h,w^0)$.

3.2.1 Discretization of the Weights

We first discretize the set of weights $\mathcal{W}$. Divide the interval $[0,1]$ into buckets $B_0=[0,\delta)$, $B_{j+1}=[(1+\gamma_1)^j\delta,(1+\gamma_1)^{j+1}\delta)$ for $j=0,1,\ldots,M-1$ for $M=\lceil\log_{1+\gamma_1}(1/\delta)\rceil$. For any weight $w\in\mathcal{W}$, we construct a new weight $w'=(w'_1,\ldots,w'_n)$ as follows. For each $i\in[n]$, w'_i is the upper-end point of the bucket containing w_i. Note that this guarantees that either $w_i\leq\delta$ or $\frac{w'_i}{1+\gamma_1}\leq w_i\leq w'_i$. We now substitute $\delta=\frac{\gamma_1}{2n}$ and write $\mathcal{N}(\gamma_1,\mathcal{W})$ to denote the set containing all the discretized weights of the set $\mathcal{W}$. Now we show that designing fair classifier wrt to the set of weights $\mathcal{N}(\gamma_1,\mathcal{W})$ is sufficient to designing fair classifier wrt to the set of weights $\mathcal{W}$.

Lemma 1. *If $\delta^w_{DP}(f)\leq\epsilon$ for any weight $w\in\mathcal{N}(\gamma_1,\mathcal{W})$, then we have $\delta^w_{DP}(f)\leq\epsilon+4\gamma_1$ for any weight $w\in\mathcal{W}$.*

Therefore, in order to ensure that $\delta^w_{DP}(f)\leq\varepsilon$ we discretize the set of weights $\mathcal{W}$ and enforce $\varepsilon-4\gamma_1$ fairness for all the weights in the set $\mathcal{N}(\gamma_1,\mathcal{W})$. This result makes our work easier as we need to guarantee fairness with respect to a finite set of weights $\mathcal{N}(\gamma_1,\mathcal{W})$, instead of a large and continuous set of weights $\mathcal{W}$. However, note that, the number of weights in $\mathcal{N}(\gamma_1,\mathcal{W})$ can be $O\left(\log^n_{1+\gamma_1}(2n/\gamma_1)\right)$, which can be exponentially large in n. We next see how to circumvent this problem.

3.3 Setting up a Two-Player Zero-Sum Game

In this section, we set up the problem of designing a fair and robust classifier with respect to the set of weights in $\mathcal{N}(\gamma_1, \mathcal{W})$ as a two-player zero-sum game. Let us introduce the notation $T(w, a, f) = \frac{\sum_{i:a_i=a} w_i f(x_i,a)}{\sum_{i:a_i=a} w_i}$. Then $\delta^w_{DP}(f) = \sup_{a,a'} |T(w, a, f) - T(w, a', f)|$. Now our aim is to solve the following problem.

$$\min_{h\in\mathcal{H}} \sum_{i=1}^{n} w_i^0 \ell(h(x_i, a_i), y_i) \tag{4}$$
$$\text{s.t.} T(w, a, h) - T(w, a', h) \leq \varepsilon - 4\gamma_1 \; \forall w \in N(\gamma_1, \mathcal{W}) \; \forall a, a' \in \mathcal{A}$$

We form the following Lagrangian.

$$\min_{h\in\mathcal{H}} \max_{\lambda:\|\lambda\|_1\leq B} \sum_{i=1}^{n} w_i^0 \ell(h(x_i, a_i), y_i) + \sum_{w\in N(\gamma_1,\mathcal{W})} \sum_{a,a'\in\mathcal{A}} \lambda_w^{a,a'} (T(w, a, h) - T(w, a', h) - \varepsilon + 4\gamma_1) \tag{5}$$

Notice that we restrict the lagrangian multipliers so that its L_1-norm is bounded by the parameter B. We will later see how to choose this parameter B. In order to solve the minmax problem defined in equation 5, we first convert equation 5 as a two-player zero-sum game. Here the learner's pure strategy is to play a hypothesis h in $\mathcal{H}$. Given the learner's hypothesis $h \in \mathcal{H}$, the adversary can pick the constraint that violates fairness the most and set the corresponding coordinate of λ to B. Therefore, for a fixed hypothesis h, it is sufficient for the adversary to play a vector λ such that either all the coordinates of λ are zero or exactly one is set to B. We denote these set of "pure" strategies by Λ_p. Then for any pair of actions $(h, \lambda) \in \mathcal{H} \times \Lambda_p$, the payoff of the h-player of the two-player zero-sum game is defined as

$$U(h, \lambda) = \sum_{i=1}^{n} w_i^0 \ell(h(x_i, a_i), y_i) + \sum_{w\in N(\gamma_1,\mathcal{W})} \sum_{a,a'\in\mathcal{A}} \lambda_w^{a,a'} (T(w, a, h) - T(w, a', h) - \varepsilon + 4\gamma_1)$$

Now our goal is to compute a ν-approximate minmax equilibrium of this game. In the next subsection, we will design an algorithm for this problem based on online learning. But we first see how both the h-player and the λ-player compute their best responses. These will be the main components of the learning algorithm discussed later.

Best response of the h-player: For each $i \in [n]$, we introduce the following notation

$$\Delta_i = \sum_{w\in N(\gamma_1,\mathcal{W})} \sum_{a'\neq a_i} \left(\lambda_w^{a_i,a'} - \lambda_w^{a',a_i}\right) \frac{w_i}{\sum_{j:a_j=a_i} w_j}$$

With this notation, the payoff becomes

$$U(h, \lambda) = \sum_{i=1}^{n} w_i^0 \ell(h(x_i, a_i), y_i) + \Delta_i h(x_i, a_i) - (\varepsilon - 4\gamma_1) \sum_{w\in N(\varepsilon/5,\mathcal{W})} \sum_{a,a'\in\mathcal{A}} \lambda_w^{a,a'}$$

Let us introduce the following costs.

$$c_i^0 = \begin{cases} \ell(0,1)w_i^0 & \text{if } y_i = 1 \\ \ell(0,0)w_i^0 & \text{if } y_i = 0 \end{cases} \qquad c_i^1 = \begin{cases} \ell(1,1)w_i^0 + \Delta_i & \text{if } y_i = 1 \\ \ell(1,0)w_i^0 + \Delta_i & \text{if } y_i = 0 \end{cases} \tag{6}$$

Then the h-player's best response becomes the following cost-sensitive classification problem.

$$\widehat{h} \in \arg\min_{h\in\mathcal{H}} \sum_{i=1}^{n} \left\{c_i^1 h(x_i, a_i) + c_i^0 (1 - h(x_i, a_i))\right\} \tag{7}$$

Therefore, as long as we have access to an oracle that solves the cost-sensitive classification problem, the h-player can solve it's best response problem. Note that, the ntion of a cost-sensitive classification as an oracle was also used by Agarwal et al. [2018], Kearns et al. [2017]. In general, solving this problem is NP-hard, but several efficient implementations are available. We provide further details about how we implement this oracle in the scetion devoted to the experiments.

Best response of the λ-player: Given a hypothesis $h \in \mathcal{H}$, the best response of the λ-player is simple – find the weight in $\mathcal{N}(\gamma_1, \mathcal{W})$ that violates the fairness constraint most and set that coordinate to B. However, finding such a constraint is a non-linear optimization problem because the fairness constraints depend on the weights non-linearly (e.g. see definition of demographic parity in eq. 1). However, we show that an approximate best response can be computed by solving a system of linear programs. The idea is that we can guess the marginal probabilities over the protected groups. If we are correct, then the most violating weight vector for demographic parity can be found by a linear program. Since we cannot exactly guess this particular value, we instead discretize the set of marignal probabilities, iterate over them and choose the option with largest violation in fairness.

The above intuition can be formalized as follows. We first discretize the set of all marginals over the groups i.e. the simplex over $|\mathcal{A}|$. For example, we discretize $[0, 1]$ as $0, \delta, (1 + \gamma_2)^j \delta$ for $j = 1, 2, \ldots, M$ for $M = O(\log_{1+\gamma_2}(1/\delta))$. This discretizes $[0, 1]^{\mathcal{A}}$ into $M^{|\mathcal{A}|}$ points, and then we retain the discretized marginals whose total sum is at most $1 + \gamma_2$, and discard all other points. Let us denote the set of such marginals as $\Pi(\gamma_2, \mathcal{A})$. Algorithm 2 describes the best response of the λ-player for a given choice of hypothesis h. It goes through all the marginals π in $\Pi(\gamma_2, \mathcal{A})$ and for each such tuple and a pair of groups a, a' finds the weight w which maximizes $T(w, a, h) - T(w, a', h)$. Note that this can be solved using a Linear Program as the weights assigned to a group is fixed by the marginal tuple π. Out of all the solutions, the algorithm finds the one with the maximum value. Then it checks whether this maximum violates the constraint i.e. greater than $\varepsilon - 4\gamma_1$. If so, it sets the corresponding λ value to B and everything else to 0. Otherwise, it returns the zero vector. Note that, the weight returned by the linear program need not correspond to a weight in $\mathcal{N}(\gamma_1, \mathcal{W})$. In that case, the algorithm rounds the weight to the nearest weight in $\mathcal{N}(\gamma_1, \mathcal{W})$ and sets the corresponding λ variable to B. For discretizing the marginals we will set $\delta = (1 + \gamma_2)\frac{\gamma_1}{n}$, which implies that the number of LP-s run by algorithm 2 is at most $O\left(\log^{|\mathcal{A}|}_{1+\gamma_2}\left(\frac{n}{(1+\gamma_2)\gamma_1}\right)\right) = O(\text{poly}(\log n))$, as the number of groups is fixed.

ALGORITHM 2: Best Response of the λ-player

Input: Training Set: $\{x_i, a_i, y_i\}_{i=1}^n$, and hypothesis $h \in \mathcal{H}$.
for *each* $\pi \in \Pi(\gamma_2, \mathcal{A})$ **do**
  for *each* $a, a' \in \mathcal{A}$ **do**
    Solve the following LP:

$$w(a, a', \pi) = \arg\max_{w} \quad \frac{1}{\pi_a} \sum_{i:a_i=a} w_i h(x_i, a) - \frac{1}{\pi_{a'}} \sum_{i:a_i=a'} w_i h(x_i, a')$$

$$\text{s.t.} \quad \sum_{i:a_i=a} w_i = \pi_a \quad \sum_{i:a_i=a'} w_i = \pi_{a'} \quad w_i \geq 0 \quad \forall i \in [n] \quad \sum_{i=1}^{n} w_i = 1$$

    Set $\text{val}(a, a', \pi) = \frac{1}{\pi_a} \sum_{i:a_i=a} w(a, a', \pi)_i h(x_i, a) - \frac{1}{\pi_{a'}} \sum_{i:a_i=a'} w(a, a', \pi)_i h(x_i, a')$
  end
end
Set $(a_1^*, a_2^*, \pi^*) = \arg\max_{a,a',\pi} \text{val}(a, a', \pi)$
if *val*$(a_1^*, a_2^*, \pi^*) > \varepsilon$ **then**
  Let $w = w(a_1^*, a_2^*, \pi^*)$.
  For each $i \in [n]$, let w_i' be the upper-end point of the bucket containing w_i.
  return $\lambda_w^{a,a'} = \begin{cases} B & \text{if } (a, a', w) = (a_1^*, a_2^*, w') \\ 0 & \text{o.w.} \end{cases}$
else
  return $\vec{0}$

The next lemma shows that algorithm 2 computes an approximate best-response for the λ-player.

Lemma 2. *Algorithm 2 is an $B(4\gamma_1 + \gamma_2)$-approximate best response for the λ-player i.e. for any $h \in \mathcal{H}$, it returns λ^* such that $U(h, \lambda^*) \geq \max_\lambda U(h, \lambda) - B(4\gamma_1 + \gamma_2)$.*

Learning Algorithm: We are now ready to introduce our algorithm for the problem defined in equation 5. In this algorithm, the h-player will use a learning algorithm, but the λ-player will use algorithm 2 to compute its approximate best response. The learning algorithm will use the Regularized Follow the Leader (RFTL) algorithm and we now briefly review this algorithm and its guarantees. RFTL is a classical algorithm for online convex optimization (OCO). In OCO, the

decision maker takes a decision $x_t \in \mathcal{K}$ at round t, an adversary reveals a convex loss function $f_t : \mathcal{K} \to \mathbb{R}$, and the decision maker suffers a loss of $f_t(x_t)$. The goal is to minimize regret, which is defined as $\max_{u \in \mathcal{K}} \left\{ \sum_{t=1}^{T} f_t(x_t) - f_t(u) \right\}$ i.e. the difference between the loss suffered by the learner and the best fixed decision that could have been made in hindsight. RFTL algorithm requires a strongly convex regularization function $R : \mathcal{K} \to \mathbb{R}_{\geq 0}$ chooses x_t according to the following rule:

$$x_t = \arg\min_{x \in \mathcal{K}} \left\{ \eta \sum_{s=1}^{t-1} \nabla f_s(x_s)^T x + R(x) \right\}.$$

The interested reader is referred to the textbook by Hazan [2016] for details about RFTL and guarantees on its performance.

We now use RFTL algorithm to design an algorithm (3) for solving a minmax approximate equilibrium of the game $U(h, \lambda)$. The h-player uses RFTL as a learning algorithm to choose h_t, whereas the λ-player uses algorithm 2 to generate an approximate best response λ_t to h_t. Recall that, in order to use the RFTL algorithm we need to specify the regularization function R and cost function f_t in each round. We choose $R(x) = 1/2\|x\|_2^2$. At round t, for an action h_t, the cost function is $f_t(h_t) = U(h_t, \lambda_t)$ where λ_t is the approximate best-response to h_t. Therefore, in order to implement RFTL the learner needs to solve the following optimization problem.

$$h_t \in \arg\min_{h \in \mathcal{H}} \left\{ \eta \sum_{s=1}^{t-1} \nabla_{h_s} U(h_s, \lambda_s)^T h + \frac{1}{2}\|h\|_2^2 \right\} \tag{8}$$

Here with slight abuse of notation we write $\mathcal{H}$ to include the set of randomized classifiers. So that $h(x_i, a_i)$ is interpreted as the probability that hypothesis h outputs 1 on an input (x_i, a_i). Now we show that the optimization problem (eq. 8) faced by the learner becomes a cost-sensitive classification problem. Recall that given λ_s, the best response of the learner is given by the following cost-sensitive classification problem.

$$\widehat{h} \in \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} c_i^1(\lambda_s) h(x_i, a_i) + c_i^0(\lambda_s)(1 - h(x_i, a_i))$$

We now use a linearization trick used by Kearns et al. [2017]. Writing $L_i(\lambda_s) = c_i^1(\lambda_s) - c_i^0(\lambda_s)$, the objective in the problem above becomes $\sum_{i=1}^{n} L_i(\lambda_s) h(x_i, a_i)$. Hence, $\nabla_{h_s} U(h_s, \lambda_s)$ is linear in h_s and equals the vector $\{L_i(\lambda_s)\}_{i=1}^n$. With this observation, the objective in equation 8 becomes

$$\begin{aligned}
&= \eta \sum_{s=1}^{t} \sum_{i=1}^{n} L(\lambda_s) h(x_i, a_i) + \frac{1}{2} \sum_{i=1}^{n} (h(x_i, a_i))^2 \\
&\leq \eta \sum_{i=1}^{n} L\left(\sum_{s=1}^{t} \lambda_s \right) h(x_i, a_i) + \frac{1}{2} \sum_{i=1}^{n} h(x_i, a_i) \\
&= \sum_{i=1}^{n} \left(\eta L\left(\sum_{s=1}^{t} \lambda_s \right) + \frac{1}{2} \right) h(x_i, a_i)
\end{aligned}$$

The second inequality follows from two observations. First, $L(\lambda)$ is linear in λ. Second, since the predictions $h(x_i, a_i) \in [0, 1]$ we replace the quadratic term by a linear term, an upper bound[1]. We would like to contrast our approach with Kearns et al. [2017], who used Follow the Perturbed Leader (FTPL) algorithm for the learner. Our approach has several benefits.

1. Since RFTL doesn't use randomized regularization, the predictions generated by our method are more robust. [Deb: Rephrase.]
2. We will use RFTL based learning algorithm to design a fast solver for the minmax objective.

Finally, we observe that even though the number of λ-variables is exponential in n, the algorithm can be efficiently implemented. In fact, the best response of the λ-player always returns a solution where all the variables are zero or exactly one is set to B. Therefore, instead of recording the entire λ vector the learning algorithm can just record the non-zero variables and there will be at most T of them. The next lemma provides performance guarantees of algorithm 3.

[1] Without this relaxation we will be solving a regularized version of cost-sensitive classification

ALGORITHM 3: Approximate Fair Classifier (ApxFair)

Input: $\eta > 0$, weight $w^0 \in \mathbb{R}^n_+$, number of rounds T
Set $h_1 = 0$
for $t \in [T]$ **do**
  $\lambda_t = \text{Best}_\lambda(h_t)$
  Set $\widetilde{\lambda}_t = \sum_{t'=1}^{t} \lambda_{t'}$
  $h_{t+1} = \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{n} (\eta L_i(\widetilde{\lambda}_t) + 1/2) h(x_i, a_i)$
end
return *Uniform distribution over* $\{h_1, \ldots, h_T\}$.

Theorem 2. *Given a desired fairness level ε, if algorithm 3 is run for $T = O\left(\frac{n}{\varepsilon^2}\right)$ rounds, then the ensemble hypothesis $\widehat{h}$ provides the following guarantees:*

$$\sum_{i=1}^{n} w_i^0 \ell(\widehat{h}(x_i, a_i), y_i) \leq \min_{h \in \mathcal{H}} \sum_{i=1}^{n} w_i^0 \ell(h(x_i, a_i), y_i) + O(\varepsilon) \text{ and } \delta^w_{DP}(\widehat{h}) \leq 2\varepsilon \quad \forall w \in \mathcal{W}.$$

4 Faster Approximate Fair Classifier

5 Experiment

6 Conclusion

References

Compas dataset. https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis. Accessed: 2019-10-26.

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. *arXiv preprint arXiv:1803.02453*, 2018.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. In *Advances in neural information processing systems*, pages 4349–4357, 2016.

Joy Buolamwini and Timnit Gebru. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems 30*, pages 3992–4001. 2017.

Robert S Chen, Brendan Lucier, Yaron Singer, and Vasilis Syrgkanis. Robust Optimization for Non-Convex Objectives. In *Advances in Neural Information Processing Systems*, pages 4705–4714, 2017.

Yoav Freund and Robert E Schapire. Game theory, on-line prediction and boosting. In *COLT*, volume 96, pages 325–332. Citeseer, 1996.

Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of Opportunity in Supervised Learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. *arXiv preprint arXiv:1711.05144*, 2017.

Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning Fair Representations. In *International Conference on Machine Learning*, pages 325–333, 2013.

A Appendix

Lemma 3. *If $\delta^w_{DP}(f) \le \epsilon$ for any weight $w \in \mathcal{N}(\gamma_1, \mathcal{W})$, then we have $\delta^w_{DP}(f) \le \epsilon + 4\gamma_1$ for any weight $w \in \mathcal{W}$.*

Proof. Recall the definition of demographic parity with respect to a weight vector w.

$$\delta^w_{DP}(f) = \left| \frac{\sum_{i:a_i=a} w_i f(x_i, a)}{\sum_{i:a_i=a} w_i} - \frac{\sum_{i:a_i=a'} w_i f(x_i, a')}{\sum_{i:a_i=a'} w_i} \right|$$

For a given weight w, we construct a new weight $w' = (w'_1, \ldots, w'_n)$ as follows. For each $i \in [n]$, w'_i is the upper-end point of the bucket containing w_i. Note that this guarantees that either $w_i \le \delta$ or $\frac{w'_i}{1+\gamma_1} \le w_i \le w'_i$. We now establish the following lower bound.

$$\frac{\sum_{i:a_i=a} w_i f(x_i, a)}{\sum_{i:a_i=a} w_i} \ge \frac{1}{1+\gamma_1} \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} \ge (1-\gamma_1) \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} \tag{9}$$

Also note that,

$$\sum_{i:a_i=a} w'_i \le \sum_{i:a_i=a, w_i>\delta} w_i + \sum_{i:a_i=a, w_i\le\delta} \delta \le (1+\gamma_1) \sum_{i:a_i=a, w_i>\delta} w'_i + n\delta$$

This gives us the following.

$$\frac{\sum_{i:a_i=a} w_i f(x_i, a)}{\sum_{i:a_i=a} w_i} \le \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\frac{1}{1+\gamma_1} \sum_{i:a_i=a} w'_i - \frac{n\delta}{1+\gamma_1}} \le (1+\gamma_1) \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i - n\delta}$$

Now we substitute, $\delta = \gamma_1/(2n)$ and get the following upper bound.

$$\begin{aligned} \frac{\sum_{i:a_i=a} w_i f(x_i, a)}{\sum_{i:a_i=a} w_i} &\le (1+\gamma_1) \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i - \gamma_1/2} \\ &\le \frac{1+\gamma_1}{1-\gamma_1} \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} \le (1+3\gamma_1) \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} \end{aligned} \tag{10}$$

Now we bound $\delta^w_{DP}(f)$ using the results above. Suppose $\frac{\sum_{i:a_i=a} w_i f(x_i,a)}{\sum_{i:a_i=a} w_i} > \frac{\sum_{i:a_i=a'} w_i f(x_i,a')}{\sum_{i:a_i=a'} w_i}$. Then we have,

$$\begin{aligned} \delta^w_{DP}(f) &\le (1+3\gamma_1) \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} - (1-\gamma_1) \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} \\ &\le \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} - \frac{\sum_{i:a_i=a} w'_i f(x_i, a)}{\sum_{i:a_i=a} w'_i} + 4\gamma_1 \\ &\le \delta^{w'}_{DP}(f) + 4\gamma_1 \end{aligned}$$

The first inequality uses the upper bound for the first term (eq. 10) and the lower bound for the second term (eq. 9). The proof when the first term is less than the second term in the definition of $\delta^w_{DP}(f)$ is similar. Therefore, if we guarantee that $\delta^{w'}_{DP}(f) \le \varepsilon$, we have $\delta^w_{DP}(f) \le \varepsilon + 4\gamma_1$. □

Lemma 4. *Algorithm 2 is an $B(4\gamma_1 + \gamma_2)$-approximate best response for the λ-player i.e. for any $h \in \mathcal{H}$, it returns λ^* such that*

$$U(h, \lambda^*) \ge \max_{\lambda} U(h, \lambda) - B(4\gamma_1 + \gamma_2).$$

Proof. We need to consider two cases. First, suppose that $T(w, a, h) - T(w, a', h) \le \varepsilon - 4\gamma_1$ for all $w \in \mathcal{N}(\gamma_1, \mathcal{W})$ and $a, a' \in \mathcal{A}$. Then, $\delta^w_{DP}(h) = \sup_{a,a' \in \mathcal{A}} |T(w, a, h) - T(w, a', h)| \le \varepsilon - 4\gamma_1$ for any weight $w \in \mathcal{N}(\gamma_1, \mathcal{W})$. Therefore, by lemma 1, for any weight $w \in \mathcal{W}$, we have $\delta^w_{DP}(h) \le \varepsilon$. Now, for any marginal $\pi \in \Pi(\gamma_2, \mathcal{A})$, and a, a' consider the corresponding linear program. We

show that the optimal value of the LP is bounded by ε. Indeed, consider any weight w satisfying the marginal conditions i.e. $\sum_{i:a_i=a} w_i = \pi_a$ and $\sum_{i:a_i=a'} w_i = \pi_{a'}$. Then, the objective of the LP is

$$\frac{1}{\pi_a}\sum_{i:a_i=a} w_i h(x_i,a) - \frac{1}{\pi_{a'}}\sum_{i:a_i=a'} w_i h(x_i,a') \le \sup_{w\in\mathcal{W}} \delta^w_{DP}(h) \le \varepsilon.$$

This implies that the optimal value of the LP is always less than ε. So algorithm 2 returns the zero vector, which is also the optimal solution in this case.

Second, there exists $w \in \mathcal{N}(\gamma_1,\mathcal{W})$ and groups a, a' such that $T(w,a,h) - T(w,a',h) > \varepsilon - 4\gamma_1$ and in particular let $(w^*, a_1^*, a_2^*) \in \arg\max_{w,a,a'} T(w,a,h) - T(w,a',h)$. Then the optimal solution sets $\lambda^{a_1^*,a_2^*}_{w^*}$ to B and everything else to zero. Let $\pi_{a_1^*}$ and $\pi_{a_2^*}$ be the corresponding marginals for groups a_1^* and a_2^*, and let $\pi'_{a_1^*}$ and $\pi'_{a_2^*}$ be the upper-end points of the buckets containing $\pi_{a_1^*}$ and π_{a^*2} respectively. As $\pi_{a_1^*}$ is marginal for a weight belonging to the set $\mathcal{N}(\gamma_1,\mathcal{W})$ and any weight in $\mathcal{N}(\gamma_1,\mathcal{W})$ puts at least $2\gamma_1/n$ on any training instance, we are always guaranteed that

$$\pi_{a_1^*} \ge \frac{2\gamma_1}{n} \ge \frac{\delta}{1+\gamma_2}.$$

This guarantees the following inequalities

$$\frac{\pi'_{a_1^*}}{1+\gamma_2} \le \pi_{a_1^*} \le \pi'_{a_1^*}.$$

Similarly, we can show that

$$\frac{\pi'_{a_2^*}}{1+\gamma_2} \le \pi_{a_2^*} \le \pi'_{a_2^*}.$$

Now, consider the LP corresponding to the marginal π' and subgroups a_1^* and a_2^*.

$$\begin{aligned}
&\frac{1}{\pi'_{a_1^*}}\sum_{i:a_i=a_1^*} w_i h(x_i,a_1^*) - \frac{1}{\pi'_{a_2^*}}\sum_{i:a_i=a_2^*} w_i h(x_i,a_2^*)\\
&\ge \frac{1}{(1+\gamma_2)\pi_{a_1^*}}\sum_{i:a_i=a_1^*} w_i h(x_i,a_1^*) - \frac{1}{\pi_{a_2^*}}\sum_{i:a_i=a_2^*} w_i h(x_i,a_2^*)\\
&\ge (1-\gamma_2)T(w,a_1^*,h) - T(w,a_2^*,h)\\
&\ge T(w,a_1^*,h) - T(w,a_2^*,h) - \gamma_2
\end{aligned}$$

Therefore, if the maximum value of $T(w,a,h) - T(w,a',h)$ over all weights w and subgroups a, a' is larger than $\varepsilon+\gamma_2$, the value of the corresponding LP will be larger than ε and the algorithm will set the correct coordinate of λ to B. On the other hand, if the maximum value of $T(w,a,h) - T(w,a',h)$ is between $\varepsilon - 4\gamma_1$ and $\varepsilon+\gamma_2$. In that case, the algorithm might return the zero vector with value zero. However, the optimal value in that case can be as large as $B\times(4\gamma_1+\gamma_2)$. □

Lemma 5 (Restated theorem 5.6 from Hazan [2016])**.** *The RFTL algorithm achieves the following regret bound for any $u \in \mathcal{K}$*

$$\sum_{t=1}^T f_t(x_t) - f_t(u) \le \frac{\eta}{4}\sum_{t=1}^T \|\nabla f_t(x_t)\|_\infty^2 + \frac{R(u)-R(x_1)}{2\eta}$$

Moreover, if $\|\nabla f_t(x_t)\|_\infty \le G_R$ for all t and $R(u) - R(x_1) \le D_R$ for all $u \in \mathcal{K}$, then we can optimize η to get the following bound: $\sum_{t=1}^T f_t(x_t) - f_t(u) \le D_R G_R \sqrt{T}$.

Theorem 3. *Given a desired fairness level ε, if algorithm 3 is run for $T = O\left(\frac{n}{\varepsilon^2}\right)$ rounds, then the ensemble hypothesis $\widehat{h}$ provides the following guarantees:*

$$\sum_{i=1}^n w_i^0 \ell(\widehat{h}(x_i,a_i),y_i) \le \min_{h\in\mathcal{H}} \sum_{i=1}^n w_i^0 \ell(h(x_i,a_i),y_i) + O(\varepsilon)$$

and

$$\forall w \in \mathcal{W} \quad \delta^w_{DP}(\widehat{h}) \le 2\varepsilon.$$

Proof. The statement of this theorem follows from two lemmas. Lemma 6 proves that if algorithm 3 is run for T rounds, it computes a $(2M+B)\sqrt{n/T} + B(4\gamma_1+\gamma_2)$-minmax equilibrium of the game $U(h,\lambda)$. On the other hand, lemma 7 proves that any ν-approximate solution $(\widehat{h},\widehat{\lambda})$ of the game $U(h,\lambda)$ has two properties

1. $\widehat{h}$ minimizes training loss with respect to the weight w^0 up to an additive error of 2ν.
2. $\widehat{h}$ provides ε-fairness guarantee with respect to the set of all weights in $\mathcal{W}$ upto an additive error fo $\frac{M+2\nu}{B}$.

Now substituting $\nu = (2M+B)\sqrt{n/T} + B(4\gamma_1+\gamma_2)$ we get the following two guarantees:

$$\sum_{i=1}^{n} w_i^0 \ell(\widehat{h}(x_i,a_i),y_i) \le \min_{h\in\mathcal{H}} \sum_{i=1}^{n} w_i^0 \ell(h(x_i,a_i),y_i) + 2(2M+B)\sqrt{\frac{n}{T}} + 2B(4\gamma_1+\gamma_2)$$

and

$$\forall w \in \mathcal{W} \quad \delta_{DP}^{w}(\widehat{h}) \le \varepsilon + \frac{M}{B} + 2(4\gamma_1+\gamma_2) + \left(1+\frac{2M}{B}\right)\sqrt{\frac{n}{T}}.$$

Now we can set the following values for the parameters $B = 3M/\varepsilon$, $T = 36n/\varepsilon^2$, $4\gamma_1+\gamma_2 = \varepsilon/6$, and get the desired result. [Deb: Note to self: check the derivation.] □

Lemma 6. *Suppose $|\ell(y,\widehat{y})| \le M$ for all $y,\widehat{y}$. Then algorithm 3 computes a $(2M+B)\sqrt{n/T} + B(4\gamma_1+\gamma_2)$-approximate minmax equilibrium of the game $U(h,\lambda)$ for $h \in \mathcal{H}$ and $\lambda \in \mathbb{R}_+^{|N(\gamma_1,\mathcal{W})|\times|\mathcal{A}|^2}$, $\|\lambda\|_1 \le B$.*

Proof. At round t, the cost is linear in h_t i.e. $f_t(h_t) = \sum_{i=1}^{n} L(\lambda_t)_i h_t(x_i,a_i)$. Let us write $\bar{\lambda} = \frac{1}{T}\lambda_t$ and D to be the uniform distribution over $h_1,\ldots,h_T$. Since we chose $R(x) = 1/2\|x\|_2^2$ as the regularization function and the actions are $[0,1]$ vectors in n-dimensional space, the diameter D_R is bounded by $\sqrt{n}$. On the other hand, $\|\nabla f_t(h_t)\|_\infty = \max_i |L(\lambda_t)_i|$. We now bound $|L(\lambda_t)_i|$ for an arbitrary i. Suppose $y_i = 1$. The proof when $y = 0$ is identical.

$$\begin{aligned} |L(\lambda_t)_i| = \left|c_i^1 - c_i^0\right| &= \left|w_i^0\right| |\ell(0,1) - \ell(1,1)| + |\Delta_i| \\ &\le 2M + B \end{aligned}$$

The last line follows as $w_i^0 \le 1$ and since λ_t is an approximate best reponse computed by algorithm 2, exactly one λ variable is set to B. Therefore, by theorem 5, for any hypothesis $h \in \mathcal{H}$,

$$\begin{aligned} &\sum_{t=1}^{T}\sum_{i=1}^{n} L(\lambda_t)_i h_t(x_i,a_i) - \sum_{i=1}^{n} L(\lambda_t)_i h(x_i,a_i) \le (2M+B)\sqrt{nT} \\ \Leftrightarrow &\sum_{t=1}^{T} U(h_t,\lambda_t) - U(h,\lambda_t) \le (2M+B)\sqrt{nT} \\ \Leftrightarrow &\frac{1}{T}\sum_{t=1}^{T} U(h_t,\lambda_t) \le U(h,\bar{\lambda}) + \frac{(2M+B)\sqrt{n}}{\sqrt{T}} \end{aligned} \tag{11}$$

On the other hand, λ_t is an approximate $B(4\gamma_1+\gamma_2)$-approximate best response to h_t for each round t. Therefore, for any λ we have,

$$\begin{aligned} &\sum_{t=1}^{T} U(h_t,\lambda_t) \ge \sum_{t=1}^{T} U(h_t,\lambda) - BT(4\gamma_1+\gamma_2) \\ \Leftrightarrow &\frac{1}{T}\sum_{t=1}^{T} U(h_t,\lambda_t) \ge \mathbb{E}_{h\sim D} U(h,\lambda) - B(4\gamma_1+\gamma_2) \end{aligned} \tag{12}$$

Equations 11 and 12 immediately imply that the distribution D and $\bar{\lambda}$ is a $(2M+B)\sqrt{n/T} + B(4\gamma_1+\gamma_2)$-approximate equilibrium of the game $U(h,\lambda)$ (Freund and Schapire [1996]). □

Lemma 7. *Let $(\widehat{h}, \widehat{\lambda})$ be a ν-approximate minmax equilibrium of the game $U(h, \lambda)$. Then,*

$$\sum_{i=1}^{n} w_i^0 \ell(\widehat{h}(x_i, a_i), y_i) \leq \min_{h \in \mathcal{H}} \sum_{i=1}^{n} w_i^0 \ell(h(x_i, a_i), y_i) + 2\nu$$

and

$$\forall w \in \mathcal{W} \quad \delta_{DP}^{w}(\widehat{h}) \leq \varepsilon + \frac{M + 2\nu}{B}$$

Proof. Let $(\widehat{h}, \widehat{\lambda})$ be a ν-approximate minmax equilibrium of the game $U(h, \lambda)$ i.e.

$$\forall h \quad U(\widehat{h}, \widehat{\lambda}) \leq U(h, \widehat{\lambda}) + \nu \quad \text{ and } \quad \forall \lambda \quad U(\widehat{h}, \widehat{\lambda}) \geq U(\widehat{h}, \lambda) - \nu$$

Let h^* be the optimal feasible hypothesis. First suppose that $\widehat{h}$ is feasible i.e. $T(w, a, \widehat{h}) - T(w, a', \widehat{h}) \leq \varepsilon - 4\gamma_1$ for all $w \in N(\gamma_1, \mathcal{W})$ and $a, a' \in \mathcal{A}$. In that case, the optimal λ is the zero vector and $\max_\lambda U(\widehat{h}, \lambda) = \sum_{i=1}^{n} w_i^0 \ell(h(x_i, a_i), y_i)$. Therefore,

$$\sum_{i=1}^{n} w_i^0 \ell(\widehat{h}(x_i, a_i), y_i) = \max_{\lambda} U(\widehat{h}, \lambda) \leq U(\widehat{h}, \widehat{\lambda}) + \nu \leq U(h^*, \widehat{\lambda}) + 2\nu \leq \sum_{i=1}^{n} w_i^0 \ell(h^*(x_i, a_i), y_i) + 2\nu$$

The last inequality follows because h^* is feasible and λ is non-negative. Now consider the case when $\widehat{h}$ is not feasible i.e. there exists w, a, a' such that $T(w, a, \widehat{h}) - T(w, a', \widehat{h}) > \varepsilon - 4\gamma_1$. In that case, let $(\widehat{w}, \widehat{a}, \widehat{a}')$ be the tuple with maximum violation and the optimal λ, say λ^*, sets this coordinate to B and everything else to zero. Then

$$\begin{aligned}\sum_{i=1}^{n} w_i^0 \ell(\widehat{h}(x_i, a_i), y_i) &= U(\widehat{h}, \lambda^*) - B(T(\widehat{w}, \widehat{a}, \widehat{h}) - T(\widehat{w}, \widehat{a}', \widehat{h}) - \varepsilon + 4\gamma_1) \\ &\leq U(\widehat{h}, \lambda^*) \leq U(\widehat{h}, \widehat{\lambda}) + \nu \leq U(h^*, \widehat{\lambda}) + 2\nu \leq \sum_{i=1}^{n} w_i^0 \ell(h^*(x_i, a_i), y_i) + 2\nu.\end{aligned}$$

The previous chain of inequalities also give

$$B\left(\max_{(w,a,a')} T(w, a, \widehat{h}) - T(w, a', \widehat{h}) - \varepsilon + 4\gamma_1\right) \leq \sum_{i=1}^{n} w_i^0 \ell(h^*(x_i, a_i), y_i) + 2\nu \leq M + 2\nu.$$

This implies that for all weights $w \in N(\gamma_1, \mathcal{W})$ the maximum violation of the fairness constraint is $(M + 2\nu)/B$, which in turn implies a bound of at most $(M + 2\nu)/B + \varepsilon$ on the fairness constraint with respect to any weight $w \in \mathcal{W}$. □