Fair for All: Best-effort fairness guarantees for classification. Supplementary Materials

Anilesh K. Krishnaswamy[*], **Zhihao Jiang**[†], **Kangning Wang**[*], **Yu Cheng**[‡], **Kamesh Munagala**[*]
[*]Duke University, [†]Tsinghua University, [‡]University of Illinois at Chicago

1 Omitted proofs

We first recall Theorem 1:

Theorem 1. *On any data set $\mathcal{N}$, there is no randomized classifier h (for some $\mathcal{H}$) such that for all $g \subseteq \mathcal{N}$ admitting a perfect classifier $h_g^* \in \mathcal{H}$ (i.e., $u_g(h_g^*) = 1$), we have $u_g(h) > \frac{|g|}{|\mathcal{N}|}$.*

Proof. Suppose there are only two classifiers h_1, h_2 in $\mathcal{H}$: h_1 classifies set g_1 correctly, and h_2 (equivalently $\overline{h_1}$) classifies $g_2 = \mathcal{N} \setminus g_1$ correctly. If an algorithm chooses h_1 with probability $p \geq 0$ and h_2 with probability $1-p$, then the average utility on g_1 is p, and that on g_2 is $1-p$. Clearly, we cannot simultaneously have $p > \frac{|g_1|}{n}$ and $1 - p > 1 - \frac{|g_1|}{n}$. □

Next we look at Theorem 2:

Theorem 2. *For any subset $g \subseteq \mathcal{N}$ that admits a perfect classifier $h_g^* \in \mathcal{H}$ (i.e., $u_g(h_g^*) = 1$) we have $u_g(h_{\text{PF}}) \geq \frac{|g|}{|\mathcal{N}|}$.*

Proof. Let h_{PF} be the PF classifier. Thus $h = h_{\text{PF}}$ maximizes $f(h) := \sum_{i \in \mathcal{N}} \ln u_i(h)$. Therefore, for any $h \in \mathcal{H}$ and any $\varepsilon \geq 0$,

$$f(\varepsilon \cdot h + (1-\varepsilon) \cdot h_{\text{PF}}) - f(h_{\text{PF}}) = \sum_{i \in \mathcal{N}} \ln u_i(\varepsilon \cdot h + (1-\varepsilon) \cdot h_{\text{PF}}) - \ln u_i(h_{\text{PF}}) \leq 0.$$

Since the above expression attains its maxima at $\varepsilon = 0$, we take the derivative with respect to ε, and evaluate it at $\varepsilon = 0$, to get:

$$\sum_{i \in \mathcal{N}} (u_i(h) - u_i(h_{\text{PF}})) \cdot \frac{1}{u_i(h_{\text{PF}})} \leq 0.$$

Define $n = |\mathcal{N}|$. Rearranging the inequality above, we get $\sum_{i \in \mathcal{N}} \frac{u_i(h)}{u_i(h_{\text{PF}})} \leq n$, and consequently,

$$\sum_{i \in \mathcal{N} : u_i(h) = 1} \frac{1}{u_i(h_{\text{PF}})} \leq \sum_{i \in \mathcal{N}} \frac{u_i(h)}{u_i(h_{\text{PF}})} \leq n.$$

If $g \subseteq \mathcal{N}$ of size αn is perfectly classified by h, then the above inequality gives

$$\sum_{i \in g} \frac{1}{u_i(h_{\text{PF}})} \leq n,$$

which in turn implies that

$$\frac{|g|}{\sum_{i \in g} \frac{1}{u_i(h_{\text{PF}})}} \geq \frac{|g|}{n} = \alpha.$$

Since the arithmetic mean is at least the harmonic mean, we get

$$u_g(h_{\text{PF}}) = \frac{1}{|g|} \sum_{i \in g} u_i(h_{\text{PF}}) \geq \alpha.$$

□

Recall Corollary 2.1:

Corollary 2.1. *For any subset $g \subseteq \mathcal{N}$, with its best classifier $h_g^* = \arg\max_{h \in \mathcal{H}} u_g(h)$, we have $u_g(h_{\text{PF}}) \geq \alpha \left[u_g(h_g^*)\right]^2$, where $\alpha = \frac{|g|}{n}$.*

Proof. For any $T \subseteq g$, we have:

$$u_g(h_{\mathrm{PF}}) \geq u_T(h_{\mathrm{PF}})\frac{|T|}{|g|}.$$

Let T be the largest subset of g that is perfectly classifiable by h_j. By Theorem 2, $u_T(h_{\mathrm{PF}}) \geq \alpha\frac{|T|}{|g|}$. Then, we get

$$u_g(h_{\mathrm{PF}}) \geq u_T(h_{\mathrm{PF}})\frac{|T|}{|g|} \geq \alpha\left(\frac{|T|}{|g|}\right)^2.$$

Since $\frac{|T|}{|g|} = u_g(h_g^*)$, the above inequality turns into

$$u_g(h_{\mathrm{PF}}) \geq \alpha\left[u_g(h_g^*)\right]^2.$$

□

2 Omitted examples

In the following example, we see that there are instances where, for some specific $\mathcal{H}$, the claim of Theorem 1 does not hold.

Example 2. Suppose there are three data points $\{1, 2, 3\}$, and there are four classifers $h_1, \bar{h_1}, h_2$ and $\bar{h_2}$ in $\mathcal{H}$. Classifier h_1 classifies $\{1, 2\}$ correctly, and h_2 classifies $\{2, 3\}$ correctly. If a randomized classifier h picks h_1 and h_2 with the same probability $1/2$, then for all subsets S which are perfectly classifiable, i.e., for each of $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$ and $\{2, 3\}$, the utility $u_S(h)$ is 0.5, 1, 0.5, 0.75 and 0.75, respectively. Each of these utilities is greater than the fractional size of the subsets, which does not agree with the claim of Theorem 1.

3 Computing the Proportional Fairness classifier

In this section, we describe how to compute our Proportional Fairness (PF) classifier. We present computational results for two different settings. Lemma 1 states that when the set of deterministic classifiers are given explicitly, we can compute the PF classifiers in polynomial time. Lemma 2 focuses on the case where there are exponentially or infinitely many deterministic classifiers, and shows that the PF classifier can still be computed in polynomial time, assuming that we have black-box access to an agnostic learning oracle.

Recall that $\mathcal{N}$ is the set of data-points with $|\mathcal{N}| = n$, $\mathcal{H}$ is the set of (deterministic) classifiers with $|\mathcal{H}| = m$, and $u_i(h) = 1$ if the classifier h labels the i-th data-point correctly, and $u_i(h) = 0$ otherwise.

The PF classifier is a distribution $(p_j)_{j:h_j\in\mathcal{H}}$ over deterministic classifiers. PF corresponds to the optimal solution to the following mathematical program:

$$\begin{array}{ll} \text{maximize} & \sum_{i\in\mathcal{N}} \ln v_i \\ \text{subject to} & v_i \leq \sum_{h_j\in\mathcal{H}} p_j u_i(h_j), \quad \forall i \in \mathcal{N} \\ & \sum_{h_j\in\mathcal{H}} p_j \leq 1 \\ & p_j \geq 0, \quad \forall h_j \in \mathcal{H} \end{array} \tag{1}$$

where the variables p_j describes the randomized classifier (which chooses h_j with probability p_j); and $0 \leq v_i \leq 1$ is the utility of the i-th data-point under the distribution p (i.e., the probability that the i-th data-point is classified correctly). Observe that the objective function is monotone in every v_i, so at optimality, we always have $\sum_{h_j\in\mathcal{H}} p_j = 1$, and $v_i = \sum_{h_j\in\mathcal{H}} p_j u_i(h_j)$ for every $i \in \mathcal{N}$.

Lemma 1. *Given a classification instance $(\mathcal{N}, \mathcal{H})$ with $n = |\mathcal{N}|$ data-points and $m = |\mathcal{H}|$ classifiers, the mathematical program (1) can be solved to precision $\varepsilon > 0$ in time* $\mathrm{poly}(n, m, \log(1/\varepsilon))$.

Proof. When $|\mathcal{N}| = n$ and $|\mathcal{H}| = m$, the mathematical program (1) has $n + m$ variables. The feasible region is given explicitly by a set of $n + 1$ linear constraints. Because the objective function is concave in the variables (v, p), we can minimize it using the ellipsoid method (Khachiyan, 1979; Grötschel et al., 1988) in time $\mathrm{poly}(n, m, \log(1/\varepsilon))$. □

In many applications, we often have infinitely many hypotheses in $\mathcal{H}$ (e.g., all hyperplanes in $\mathbb{R}^d$). If this is the case, the mathematical program (1) has infinitely many variables, and to solve it, we need to make some assumptions on the structure of $\mathcal{H}$.

A commonly used assumption is that there exists an agnostic learning oracle: given a set of weights on the data-points, the oracle returns an optimal classifier $h \in \mathcal{H}$ subject to these weights. Formally, we assume black-box access to an oracle for the following problem:

Definition 1 (Agnostic Learning)**.** Fix a set of data-points $\mathcal{N}$ and a family of classifiers $\mathcal{H}$. Given any weights $(w_i)_{i\in\mathcal{N}}$, find a classifier $h \in \mathcal{H}$ that maximizes the (weighted) average accuracy on $\mathcal{N}$. That is, h maximizes $\sum_{i\in\mathcal{N}} w_i u_i(h)$.

Given such an oracle, there are a few ways in which Problem (1) can be solved theoretically in polynomial time. We first sketch the outline of a *multiplicative weights* approach: Consider the feasibility version of the problem, where we want to check if there is a feasible solution that gives us an objective value of at least U^*. If we define a region P as the set of v, p, where $v = \{v_i\}_{i\in\mathcal{N}}$ and $p = \{p_j\}_{j:h_j\in\mathcal{H}}$, that satisfy:

$$\sum_{i\in\mathcal{N}} \ln v_i \geq U^*, \text{ and } v_i \geq 0 \quad \forall i \in \mathcal{N}, \tag{2}$$

$$\sum_{j:h_j\in\mathcal{H}} p_j = 1, \text{ and } p_j \geq 0 \quad \forall j : h_j \in \mathcal{H}, \tag{3}$$

then the problem can be restated as:

$$\exists?(v,p) \in P \text{ such that } \sum_{j:h_j\in\mathcal{H}} p_j u_i(h_j) \geq v_i, \quad \forall i \in \mathcal{N}.$$

This can be solved via the multiplicative weights if we have an efficient oracle for solving the following optimization problem for a given $y \geq 0$ (Bhalgat et al., 2013; Arora et al., 2012):

$$\begin{aligned} \text{maximize} \quad & \sum_{i\in\mathcal{N}} y_i \left(\sum_{j:h_j\in\mathcal{H}} p_j u_i(h_j) - v_i \right) \\ \text{subject to} \quad & (v,p) \in P. \end{aligned}$$

It can be seen that the above devolves into two decoupled problems: The first one is for v, which can be solved analytically,

$$\begin{aligned} \text{minimize} & \sum_{i\in\mathcal{N}} y_i v_i \\ \text{subject to} & \sum_{i\in\mathcal{N}} \ln(v_i) \geq U^* \\ & v \geq 0, \end{aligned}$$

and the second for p, which can be solved with access to an agnostic learning oracle:

$$\begin{aligned} \text{minimize} & \sum_{i\in\mathcal{N}} \sum_{j:h_j\in\mathcal{H}} p_j y_i u_i(h_j) \\ \text{subject to} & \sum_{j:h_j\in\mathcal{H}} p_j = 1 \\ & p \geq 0. \end{aligned}$$

As the next lemma shows, Problem (1) can also be theoretically solved in a more straightforward way using the ellipsoid method.

Lemma 2. *Given a classification instance $(\mathcal{N}, \mathcal{H})$ with $n = |\mathcal{N}|$ data-points and an agnostic learning oracle for the set of classifiers $\mathcal{H}$, the mathematical program* (1) *can be solved to precision $\varepsilon > 0$ in time* $\text{poly}(n, \log(1/\varepsilon))$.

Proof. We use the "Ellipsoid Against Hope" algorithm proposed in Papadimitriou and Roughgarden (2008).

Consider the dual of (1):

$$\begin{array}{ll} \text{minimize} & -\left(\sum_{i\in\mathcal{N}} \ln w_i\right) - n + z \\ \text{subject to} & \sum_i w_i u_i(h_j) \le z, \quad \forall h_j \in \mathcal{H} \\ & w_i \ge 0, \quad \forall i \in \mathcal{N} \end{array} \tag{4}$$

The dual program (4) has $n+1$ variables (w, z) and infinitely many constraints. Strong duality holds despite the infinite-dimensionality of (1).

Let OPT denote the optimal value of the primal and dual programs. The dual objective is convex in (w, z), so we can add a constraint

$$-\left(\sum_{i\in\mathcal{N}} \ln w_i\right) - n + z \le \text{OPT} - \varepsilon \tag{5}$$

and use ellipsoid method to check the feasibility of the dual with this additional constraint. We show a separation oracle exists so we can run the ellipsoid method. For a fixed point (w, z), we can verify Constraint (5) directly, and we can use the agnostic learning oracle to check the infinitely many constraints

$$\sum_i w_i u_i(h_j) \le z, \quad \forall h_j \in \mathcal{H} \tag{6}$$

because it is sufficient to first use the oracle to find some h_j that maximizes $\sum_i w_i u_i(h_j)$, and then check only the j-th constraint.

We know the ellipsoid method must conclude infeasibility, because the minimum possible value of the dual is OPT but we are asking for $\text{OPT} - \varepsilon$. Let $\mathcal{H}'$ denote the set of classifiers whose corresponding constraints (6) are checked in the execution of the ellipsoid method. Because the ellipsoid only examines the constraints in $\mathcal{H}'$ and concludes infeasibility, we know that if we replace $\mathcal{H}$ with $\mathcal{H}'$ in the dual program (4), the objective value is still larger than $\text{OPT} - \varepsilon$. Moreover, the cardinality of $\mathcal{H}'$ is at most $\text{poly}(n, \log(1/\varepsilon))$ because the ellipsoid method terminates in $\text{poly}(n, \log(1/\varepsilon))$ steps.

Consequently, we can replace $\mathcal{H}$ with $\mathcal{H}'$ in the primal convex program (1) to make it finite-dimensional, and invoke Lemma 1 to solve it where $m = |\mathcal{H}'| = \text{poly}(n, \log(1/\varepsilon))$. Therefore, the overall running time is $\text{poly}(n, \log(1/\varepsilon))$. □

It is worth noting that the proof of Lemma 2 continues to hold even if we only have an ε-approximately optimal agnostic learning oracle.

4 A heuristic for PF

Here we describe a heuristic for PF drawing inspiration from some literature on social choice (voting). Imagine a social choice setting: we have a set N of voters and a set C of candidates. Each voter $i \in N$ has a subset $A_i \subseteq C$ of candidates which she prefers. The goal here is to select a committee of a fixed size k, i.e., a subset $W \subseteq C$ ($|W| = k$), which "satisfies" the voters as much as possible

One method to do so is Proportional Approval Voting (Aziz et al., 2017), or PAV for short. Here, a voter is assumed (for the sake of computation) to derive a utility of $1 + \frac{1}{2} + \ldots + \frac{1}{j}$ from a committee W that contains exactly j of her approved candidates, i.e., $|A_i \cap W| = j$. For PAV, as a method of computing a committee W, the overall goal is to maximize the sum of the voters' utilities – in other words, PAV outputs a set $W^* = \arg\max_{W\subseteq C:|W|=k} \sum_{i\in N} |A_i \cap W|$.

Reweighted Approval Voting (RAV for short) converts PAV into a multi-round rule as follows: Start by setting $W = \varnothing$. Then in round j ($j = 1, \ldots, k$), select (without replacement) a candidate c which maximizes $\sum_{i\in N: c\in A_i} \frac{1}{1+|W\cap A_i|}$, and adds it to W. Finally, it outputs the set W, after k rounds. i.e., having chosen k candidates. RAV is also sometimes referred to as "Sequential PAV" (Brams and Kilgour, 2014).

In our case, for PF, we want to maximize $f(h) := \sum_{i\in\mathcal{N}} \ln u_i(h)$. Since $\ln(t) \approx \sum_{i=1}^{t} \frac{1}{i}$, we can think of PAV as a close enough proxy (where the voters are the data points, and the candidates are all the available classifiers). In the same token, we can potentially apply RAV to our problem (assuming black box access to an agnostic

learning oracle). In fact, in practice, we find that a slight modification to RAV works better. We describe this in terms of our problem setting and notation in Algorithm 1.

Algorithm 1 PF Heuristic Classifier

Input: data set $\mathcal{N}$, family of classifiers $\mathcal{H}$, number of iterations R
Initialize: $r \leftarrow 1$, $(c_i)_{i\in\mathcal{N}} \leftarrow \mathbf{0}$, $(w_i)_{i\in\{1,\dots,R\}} \leftarrow \mathbf{0}$
while $r \leq R$ **do**
  $h_r^* \leftarrow \operatorname{argmax}_{h_j\in\mathcal{H}} \sum_{i\in\mathcal{N}} \frac{1}{1+c_i} \cdot u_i(h_j)$
  $\mathcal{T}_r \leftarrow \{i \in \mathcal{N} \mid u_i(h_r^*) = 1\}$
  $w_r \leftarrow \sum_{i\in\mathcal{T}_r} \frac{1}{1+c_i}$
  for all $i \in \mathcal{T}_r$ **do**
    $c_i \leftarrow c_i + 1$
  end for
  $r \leftarrow r + 1$
end while
Return: classifier h_{PF} that chooses $h_t^* \in \mathcal{H}$ with probability $p_t \propto w_t$ for $t = 1, 2, \dots, R$.

Note that a direct analog of RAV would be choosing h_t^* with probability just $\frac{1}{R}$, i.e., uniform over all $r \in \{1, \dots, R\}$. Also note that any implementation of RAV would not have black-box access to an agnostic learning oracle. And when data points are reweighted each time, standard out-of-the-box training methods apply the weights to the respective gradient updates. Since scaling the gradient updates affects the convergence of the training procedure, some kind of rescaling is necessary. In practice, we find that reweighting the probabilities to be proportional to w_t (see Algorithm 1) works well. For example, doing so helps us to always beat the lower bound (which does not happen otherwise). For details about the exact implementation, please refer to the attached code.

5 A Greedy approximation of PF

The next question to consider is whether there is a simpler classifier that achieves similar guarantees to those given by the PF classifier. We answer this question in the affirmative by presenting the iterative Greedy algorithm (Algorithm 2): select the classifier that classifies the most number of data points correctly, allocate it a weight proportional to this number, discard the data points that it classifies correctly, and simply repeat the above procedure until there are no data points left. The randomized classifier Greedy is defined by giving to each of these classifiers a probability proportional to its weight, i.e., the number of data points classified correctly in the corresponding iteration.

We now show that the Greedy algorithm provides a constant-factor approximation to the guarantee provided by the PF classifier. Note that the first step in the *while* loop of Greedy (Algorithm 2) involves using the ERM agnostic learning black box.

Theorem 3. *If h_G is the* Greedy *classifier, then for any subset $\mathcal{S} \subseteq \mathcal{N}$ that admits a perfect classifier $h \in \mathcal{H}$, we have $u_S(h_G) \geq \frac{\alpha}{2} + \frac{1}{2\alpha}\left(\max\left(0, 2\alpha - 1\right)\right)^2 \geq \frac{\alpha}{2}$, where $\frac{|S|}{n} = \alpha$.*

Proof. Consider any set $\mathcal{S}$ of size αn (where $n = |\mathcal{N}|$) that admits a perfect classifier $h_{\mathcal{S}}$. Let h_1^* be the classifier found at the first step of Greedy (Algorithm 2). Suppose it classifies an points in $\mathcal{S}$ correctly, bn points in S incorrectly, and cn points not in $\mathcal{S}$ correctly. Clearly, $a + b = \alpha$, and $c \leq 1 - \alpha$. Further, the greedy choice implies $b \leq c$, so that $b \leq 1 - \alpha$. Therefore, $a = \alpha - b \geq \max(2\alpha - 1, 0)$.

Since Greedy assigns $p_1 = a + d$ and classifies an points in S correctly, the total utility generated on $\mathcal{S}$ is $a(a + c)n \geq a(a + b)n$.

At the second step, $|T_2| \geq nb$, since h_S classifies bn points correctly. Suppose $|T_2 \cap \mathcal{S}| = y_1 n$. Similarly, $|T_3| \geq (b - y_1)n$; let $|T_3 \cap \mathcal{S}| = y_2 n$, and so on. Therefore, the total utility generated by Greedy is at least

$$n \cdot \left(a(a + b) + \sum_{q\geq 1} (b - z_{q-1}) y_q \right),$$

Algorithm 2 GREEDY Classifier

Input: data set $\mathcal{N}$, family of classifiers $\mathcal{H}$
Initialize: $r \leftarrow 1$, $\mathcal{S}_r \leftarrow \mathcal{N}$
while $|\mathcal{S}_r| > 0$ **do**
  $h_r^* \leftarrow \text{argmax}_{h_j} u_{\mathcal{S}_r}(h_j)$
  $\mathcal{T}_r \leftarrow \{i \in \mathcal{S}_r \mid u_i(h_r^*) = 1\}$
  $p_r \leftarrow \frac{|\mathcal{T}_r|}{n}$
  $\mathcal{S}_{r+1} \leftarrow \mathcal{S}_r \setminus \mathcal{T}_r$
  $r \leftarrow r + 1$
end while
Return: classifier $h_{\mathtt{G}}$ that chooses $h_s^* \in \mathcal{H}$ with probability p_s for $s = 1, 2, \ldots, r-1$

where $z_q = \sum_{t=1}^{q} y_t$ and $\sum_{q \geq 1} y_q = b$.

Now focus on the term $\sum_{q \geq 1}(b - z_{q-1})y_q$.

$$\begin{aligned}\sum_{q \geq 1}(b - z_{q-1})y_q &= \sum_{q \geq 1}(b - y_{q-1})(z_q - z_{q-1}) \\ &\geq \sum_{q \geq 1}(b - z_{q-1})(z_q - z_{q-1}) \\ &\geq \int_0^b (b - x)\mathrm{d}x.\end{aligned}$$

Therefore,

$$\begin{aligned}& n \cdot \left(a(a+b) + \sum_{q \geq 1}(b - z_{q-1})y_q \right) \\ & \geq n \cdot \left(a(a+b) + \int_0^b x\mathrm{d}x \right), \end{aligned} \tag{7}$$

which on further simplification yields

$$n \cdot \left(a(a+b) + \frac{b^2}{2} \right) = n \cdot \left(\frac{a^2}{2} + \frac{(a+b)^2}{2} \right).$$

Now, since $|S| = \alpha n$, $a + b = \alpha$, and $a \geq \max(2\alpha - 1, 0)$,

$$\begin{aligned}u_S(h_{\mathrm{G}}) &\geq \frac{1}{\alpha}\left(\frac{(a+b)^2}{2} + \frac{a^2}{2} \right) \\ &\geq \frac{\alpha}{2} + \frac{1}{2\alpha}\left(\max\left(0, 2\alpha - 1\right)\right)^2.\end{aligned}$$

□

Therefore, for any set $\mathcal{S}$ that has a perfect classifier, the average utility of GREEDY is at least half the average utility of Proportional Fairness; for large sets, the approximation factor is better, and approaches 1 as $\alpha \to 1$. We complement our $\frac{\alpha}{2}$ analysis with the example below which shows that the above bound is tight when $\alpha \to 0$.

Example 3. Let $\mathcal{N}$ consist of the following $n = \frac{k(k+1)}{2}$ data-points:

$$\begin{array}{llll} (1,1), & (1,2), & \ldots, & (1,k-1), \quad (1,k), \\ (2,1), & (2,2), & \ldots, & (2,k-1), \\ \ldots, & \ldots, & \ldots, & \\ (k-1,1), & (k-1,2), & & \\ (k,1). & & & \end{array}$$

There are $k+1$ classifiers:

1. h_1 correctly classifies every $(1, i)$ for $i \in [k]$.
2. The classifier h_j $(j = 2, 3, \ldots, k)$ correctly classifies every (j, i) for $i \in [k - j + 1]$ as well as $(1, j)$.
3. The classifier h_{k+1} only correctly classifies $(1, 1)$.

Let $\mathcal{S} = \{(1, i) \mid i \in [k]\}$, i.e., the set which h_1 correctly classifies. In the j-th round, GREEDY can pick h_{j+1} since it covers $k - j + 1$ new data-points (tied with h_1). However, each of $h_2, h_3, \ldots, h_{k+1}$ only covers one data-point in S, meaning that the accuracy of GREEDY on S is only $\frac{1}{k} = \frac{\alpha}{2} \cdot \frac{k+1}{k}$, where $\alpha = \frac{|\mathcal{S}|}{|\mathcal{N}|}$.

6 Generalization bound for PF

In this section, we will outline how to derive generalization bounds for PF. To begin with, assume that $\mathcal{H}$ is a hypothesis space of finite VC dimension, say d. Let $\Delta(\mathcal{H})$ be the space of randomized classifiers. Let D represent the distribution of data.

For some $h \in \mathcal{H}$, as before, let $u_i(h)$ be the utility of a point i from classifier h. Let $S_i(h) = \log(u_i(h) + \varepsilon)$, where $\varepsilon > 0$. Let $A_{\Delta(\mathcal{H})}$ be the set of (expected) outcomes for i in a finite sample S of size m based on the function $S_i()$ and the space $\Delta(\mathcal{H})$. From standard generalization bounds (Theorem 26.3 in Shalev-Shwartz and Ben-David (2014)) based on Rademacher complexity, we have a bound of $\frac{2R}{\delta}$ with probability $1 - \delta$ over the choice of S (drawn randomly from D), where R is the Rademacher complexity of $A_{\Delta(\mathcal{H})}$.

We now show how to bound R. First note that in the domain $[\varepsilon, +\infty)$, log is $\frac{1}{\varepsilon}$-Lipschitz. By Lemma 26.9 in Shalev-Shwartz and Ben-David (2014), we need only bound the Rademacher complexity corresponding to $u_i(h)$ for $h \in \Delta(\mathcal{H})$, and R will be at most factor $\frac{1}{\varepsilon}$ of this bound. Moreover, by Lemma 26.7 of Shalev-Shwartz and Ben-David (2014), the Rademacher complexity is unaffected by taking the convex hull of a set, and therefore, we need only consider $h \in \mathcal{H}$. Since $\mathcal{H}$ is of finite VC dimension d, we can bound the resulting Rademacher complexity by $O(\sqrt{d/m})$ (Bartlett and Mendelson, 2002).

For example, if $\mathcal{H}$ consisted of linear classifiers whose coefficients have a bounded ℓ_2 norm of 1, and the features x are similarly bounded, then the R is $O(\frac{1}{\varepsilon\sqrt{m}})$. Therefore, the generalization error is bounded by $O(\frac{1}{\delta\varepsilon\sqrt{m}})$. And if this is at most ε, then we get a 2ε-approximation of the optimal PF solution, i.e., for $m = O(\frac{1}{\varepsilon^4\delta^2})$.

7 Omitted details from the experiments

In all our experiments, we drop sensitive features such as race and gender. For all the methods in the paper, whenever possible, we use standard implementations from the `scikit-learn` (0.21.3) package (Pedregosa et al., 2011). Wherever we need an agnostic learning black-box, we just use Logistic Regression, since it performs reasonably well on the data-sets explored in this paper. For the `adult` data set, we use the train data and test data as available at `https://archive.ics.uci.edu/ml/datasets/Adult`. For the `compas` data set, we do a train-test split of 80-20. We use the same features as discussed in `https://github.com/propublica/compas-analysis`. All code to compute the various classifiers is attached herewith.

Below, we provide some additional plots.

7.1 Additional plots for the `compas` data-set

We look at subsets that are mis-classified to varying degrees by LR and ADA, and check how well PF and GREEDY perform on them. More precisely, for each $x \in \{0, 0.1, \ldots, 1\}$, we sample 25% of the test set such that a fraction x of it comes from the points in the test set that are classified correctly by LR and ADA respectively. For example, if $x = 0.2$, then one-fifth of the sampled subset comes from the correctly classified points, and four-fifths from the incorrectly classified. Figure 1 shows the accuracy obtained by the different methods averaged across 100 samples of subsets obtained (as described above) based on LR and ADA. We see that the performance of *both* PF *and* GREEDY *is similar irrespective of whether the subsets are defined based on* LR *or* ADA. Similar trends are seen when other methods are used instead of LR or ADA.

In Figure 2, we repeat the procedure used in Figure 1, but this time the sampled subset is 75% of the test set. More precisely, for each x, we sample 75% of the test set such that a fraction x of it comes from the points in

the test set that are classified correctly by LR and ADA respectively. Since 75% is a large subset, we see both PF and GREEDY doing a bit worse than the ERM methods, and also getting close to the lower bound when the subset becomes perfectly classifiable.

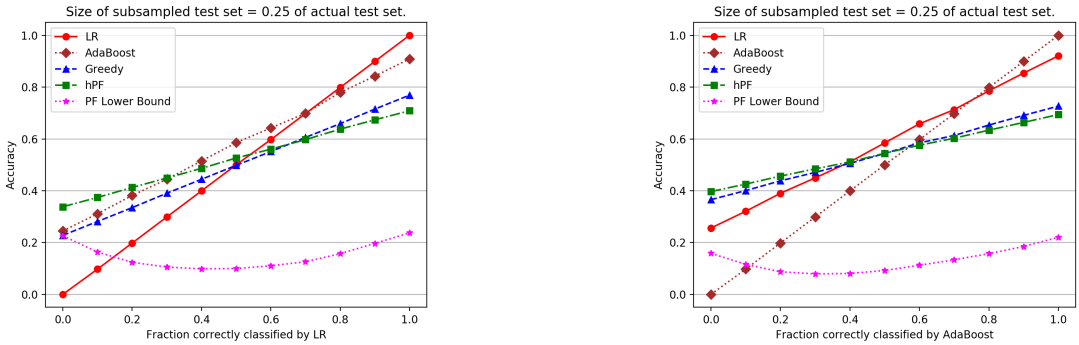

Figure 1: Accuracy on sampled subsets which are 25% of the test set (`compas`): x axis denotes the fraction of points that are classified correctly by LR (left) and ADA (right).

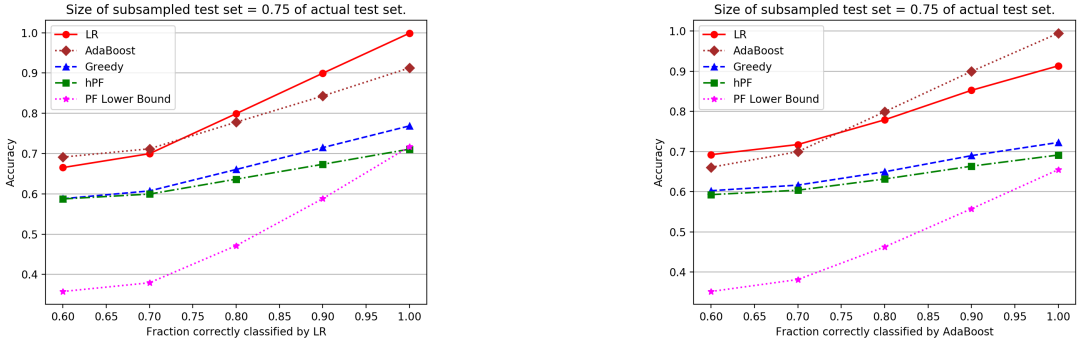

Figure 2: Accuracy on sampled subsets which are 75% of the test set (`compas`): x axis denotes the fraction of points that are classified correctly by LR (left) and ADA (right).

Similar observations can be made on the `adult` data set.

References

Arora, S., Hazan, E., and Kale, S. (2012). The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164.

Aziz, H., Brill, M., Conitzer, V., Elkind, E., Freeman, R., and Walsh, T. (2017). Justified representation in approval-based committee voting. *Social Choice and Welfare*, 48(2):461–485.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Bhalgat, A., Gollapudi, S., and Munagala, K. (2013). Optimal auctions via the multiplicative weight method. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 73–90.

Brams, S. J. and Kilgour, D. M. (2014). Satisfaction approval voting. In *Voting Power and Procedures*, pages 323–346. Springer.

Grötschel, M., Lovász, L., and Schrijver, A. (1988). *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer. Second Edition, 1993.

Khachiyan, L. G. (1979). A polynomial algorithm in linear programming. In *Doklady Akademii Nauk*, volume 244, pages 1093–1096. Russian Academy of Sciences.

Papadimitriou, C. H. and Roughgarden, T. (2008). Computing correlated equilibria in multi-player games. *Journal of the ACM (JACM)*, 55(3):1–29.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.