Online Selection with Cumulative Fairness Constraints

We propose and study the problem of online selection with cumulative fairness constraints. In this problem, candidates arrive online, i.e., one at a time, and the decision maker must choose to `accept` or `reject` each candidate subject to a constraint on the history of decisions made thus far. We introduce deterministic, randomized, and learned policies for selection in this setting. Empirically, we demonstrate that our learned policies achieve the highest utility. However, we also show—using 700 synthetically generated datasets—that the simple, greedy algorithm is often competitive with the optimal sequence of decisions, obviating the need for complex (and often inscrutable) learned policies, in many cases. Theoretically, we analyze the limiting behavior of our randomized approach and prove that it satisfies the fairness constraint with high probability.

Introduction

Due to their proven effectiveness across a plethora of tasks, machine learning (ML) algorithms have been incorporated into many technologies that are widely used across the globe. In many cases, these technologies are utilitized in high-stakes situations, such as: hiring, risk assessment in the judicial system, and law enforcement (Raghavan et al. 2019; Elyounes 2019; Meijer and Wessels 2019). However, ML algorithms have also been shown to discriminate—especially against candidates from minority groups (Bogen 2019; Larson, Roswell, and Atlidakis 2017; Lum and Isaac 2016). Not only is this often unjust, but it can also detract from both short-term and long-term utility of the services these algorithms support (Kleinberg and Raghavan 2018; Wick, Panda, and Tristan 2019).

When designing ML algorithms intended to minimize such discriminatory behavior, it is critical to consider a handful of desiderata; two of which we focus on here. First, many technologies that encapsulate ML algorithms are used over time. Instead of all users interacting with these technologies simultaneously, the ML algorithms are often invoked repeatedly for a small batch of users—or even a single user. This motivates us to focus on the *online* problem setting. Second, algorithmic decision-making must be considered *cumulatively* and *continueously*, rather than instantaneously. In other words, when testing for discriminatory behavior, the entire sequence of decisions made by an ML algorithm should be considered throughout the algorithm's lifetime, rather than only at the end, or any instantaneous point in time. For example, it is insufficient for a credit scoring algorithm to assign lower scores to candidates of one demographic group for 11 months, then, assign similar candidates high scores for one month, and be considered "fair" over the course of a year.

Literature in algorithmic fairness proposes a number of strategies for mitigating discrimination. Broadly speaking, many strategies take one of the following 3 approaches: 1. transform the data on which an ML algorithm is trained (pre-processing), 2. alter the training procedure (in-processing), or 3. modify the output of the trained ML algorithm (post-processing) (Friedler et al. 2019; Romei and Ruggieri 2011). We study post-processing since it can be readily utilized in concert with black-box ML algorithms.

In this work, we focus on the family of *online selection problems*, which are broadly applicable in domains such as lending, hiring, and promotion. In an online selection problem, candidates arrive one at a time. Upon arrival—and using whatever information is available—the decision-making system must choose whether to select the candidate or not. As is often the case, we treat these decisions as irrevocable (Elmachtoub and Levi 2015; Correa et al. 2020). Intending to mitigate discriminatory behavior at all points in the decision-making process, we study selection problems subject to cumulative constraints. In particular, these constraints consider the entire history of decisions at all points throughout the process. Our online selection problem can be applied in conjuctation with a variety of fairness constraints.

We introduce and examine deterministic, randomized, and learned selection policies in the context of 4 datasets. While the learned policies perform best in terms of utility maximization, we find that a simple, greedy approach that ensures the constraints are always met is often competitive. In order to understand when the greedy policy performs well, we compare it to the optimal sequence of selections in 700 distinct, synthetically generated datasets. The results reveal that the greedy policy is consistently competitive with the optimal sequence of decisions when candidates from different groups exhibit similar mean utility; this is independent of the variance in utility, the composition of the sequence (in terms of fraction of candidates from each group), or the value

of the mean utility. Moreover, when mean utilities diverge, the greedy selection policy is still often competitive with the optimal sequence, however its competitiveness exhibits higher variance. While our learned policies perform best, our work suggests that when group-dependent mean utility is nearly equal (as is expected in many settings), greedy but fair selection is sufficient and complicated learned policies are unnecessary.

For completeness, we also experiment with policies constructed offline using pre-processed data as well as policies trained explicitly to be fair (in-processing). We find that these policies are often unable to satisfy our cumulative fairness constraints. Furthermore, we find that our online selection policies tend to achieve higher utility and exhibit more even selections across groups. Our work underscores the importance of thoughtfully analyzing and considering the stakeholders of a deployed system *before* building any ML components, as well as the importance of designing algorithms for real-world, online usage. In summary a) we pose the online selection problem with cumulative constraints and study it in the non-batched setting, b) we introduce deterministic, randomized and learned policies for the online selection problem, c) we prove strong asymptotic properties of our randomized policy and d) we conduct a large empirical study that reveals that the greedy strategy is often competitive with the optimal selection algorithm. We emphasize that our goal is to study various policies for fair online selection, and analyze the strengths and weaknesses of each. We do not advocate for applying these policies in practice precisely as written, nor are we suggesting that they would necessarily withstand legal scrutiny. Likewise, we do not advocate for a specific fairness criterion, but rather consider well-known legal criteria for concreteness. Throughout this document, we use the term "fair" to mean adherence to (fairness) constraints.

Fair Online Selection

We study an online selection process in which candidates arrive one at a time and are immediately either `accept`ed or `reject`ed. Each candidate belongs to a known group and exhibits a score. The decision maker would like to `accept` candidates with high scores, and `reject` candidates with low scores. However, throughout this process, the decision maker's actions are subject to a *fairness audit*, which determines whether its decisions so far are fair with respect to the candidates' groups.

Formally, let $\mathcal{X} = \{s_t, c_t\}_{t=1}^{N}$ be a sequence of candidates where the candidate arriving at time t has score $s_t \in \mathbb{R}$ and group $c_t \in \{1, \cdots, K\}$. Let $A = \{\texttt{accept}, \texttt{reject}\}$, be the outcomes for a candidate and let $U : \mathcal{X} \times A \to \mathbb{R}$ be the utility for either `accept`ing or `reject`ing a candidate. Define a decision making *policy*, $\pi : \mathcal{X} \times \psi \to A$, where ψ represents a finite history of decision, and let π_t be the decision for the candidate x_t. Finally, let $V : \psi \to \{\texttt{pass}, \texttt{fail}\}$ be a function that determines whether the history of decisions, ψ is fair. Requiring a policy to always pass the fairness audit, V, is an example of a *cumulative* constraint since it applies to an aggregate of all decisions made thus far. As such, we refer to such a requirement as either a fairness constraint or a fairness audit. Then the decision maker's goal is to select the policy,

$$\pi^\star = \arg\max_{\pi \in \Pi} \mathrm{E}_{\mathcal{X} \sim \mathcal{D}} \left[\sum_{t=1}^{N} U(x_t, \pi_t) \right]$$
$$\text{s.t. } V(\psi_t) = \texttt{pass}, \forall t.$$

In words, the optimal policy, $\pi^\star$, is one that maximizes expected utility over possible candidate sequences while `pass`ing the fairness audit at each timestep. While this problem definition is general, we focus on the case in which we make no distributional assumptions on candidates and where the candidate sequence length, N, is unknown.

Illustrative Example

To ground the fair online selection problem, we present the example in Figure 1. In this example, the sequence of incoming candidates is of length 10. Each candidate belongs to one of two groups: $\diamondsuit$ or $\heartsuit$. Moreover, the utility of `accept`ing (✓) a candidate is equal to the candidate's score, and the utility of `reject`ing a candidate (⊗) is the negated candidate score, i.e., $U(x_t, \texttt{accept}) = -U(x_i t, \texttt{reject}) = s_t$. Here, the cumulative fairness constraint checks whether the absolute value of the difference between the number of `accept`ed candidates from the two groups is greater than 2. Formally, if $\mathcal{A}_\pi[\diamondsuit]_t$ and $\mathcal{A}_\pi[\heartsuit]_t$ are the number of candidates `accept`ed from the two groups by a policy, π, after the first t candidates arrive, then π fails the audit if $|\mathcal{A}_\pi[\diamondsuit]_t - \mathcal{A}_\pi[\heartsuit]_t| > 2$. Concretely, if at time t, a policy has `accept`ed 5 candidates from group $\diamondsuit$ and only 2 candidates from group $\heartsuit$, the policy `fail`s the audit.

The Figure visualizes 3 selection policies.

UNFAIR (rows 4-7) This policy greedily `accept`s and `reject`s candidates to maximize utility without regard for the fairness constraint. In this example, UNFAIR accepts candidates with positive scores and rejects candidates with negative scores. In doing so, it violates the fairness constraint at 5 timesteps.

Greedy But Fair (GBF; rows 8-11) This policy makes the same decisions as UNFAIR unless doing so would violate the fairness constraints. When GBF deviates from UNFAIR, we say that GBF has triggered the *failsafe*. In Figure 1, GBF triggers the failsafe twice (visualized by yellow-filled cells), at timesteps 4 and 8, missing out on two high-scoring candidates.

Optimal (OPT; rows 12-15) This reflects the optimal sequence of decisions. Notice that OPT deviates from GBF at critical points (timesteps 1 and 7). In doing so, it provides itself with the flexibility to `accept` the high-scoring candidates at timesteps 4 and 8, culminating in the maximal cumulative utility for any fair policy. Computing OPT requires knowledge of the full sequence a priori and is therefore unrealizable in online settings.

Algorithmic Policies

In addition to introducing the problem, in the previous section, we discussed 3 deterministic policies for online selection: UNFAIR, GBF, and OPT. Of these, only GBF can be used in practice since UNFAIR does not adhere to the fairness constraints, and computing OPT requires knowledge

1	Candidate	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
2	score	1	2	-1	3	5	-4	1	9	9	3
3	group	♢	♢	♡	♢	♡	♢	♢	♢	♡	♢
4	**UNFAIR**										
5	`accept/reject`	✓	✓	⊗	✓	✓	⊗	✓	✓	✓	✓
6	$\lvert\mathcal{A}[♢]_t - \mathcal{A}[♡]_t\rvert$	1	2	2	3	2	2	3	4	3	4
7	Cum. Utility	1	3	4	7	12	16	17	26	35	38
8	**Greedy But Fair**										
9	`accept/reject`	✓	✓	⊗	⊗	✓	⊗	✓	⊗	✓	✓
10	$\lvert\mathcal{A}[♢]_t - \mathcal{A}[♡]_t\rvert$	1	2	2	2	1	1	2	2	1	2
11	Cum. Utility	1	3	4	1	6	10	11	2	11	14
12	**Optimal**										
13	`accept/reject`	⊗	✓	⊗	✓	✓	⊗	⊗	✓	✓	✓
14	$\lvert\mathcal{A}[♢]_t - \mathcal{A}[♡]_t\rvert$	1	2	2	2	1	1	1	2	1	2
15	Cum. Utility	-1	1	2	5	8	12	11	20	29	32

Figure 1: **Fair Online Selection Example.** The first 3 rows visualize a sequence of 10 candidates, their scores, and their groups (♢ or ♡). The following rows visualize the 3 selection policies: **Unfair**, **Greedy But Fair**, and **Opt**. For each policy, 3 statistics are show: 1) its decision—either `accept` (✓) or `reject` (⊗), 2) the absolute difference between the number of `accept`ed candidates from both groups, and 3) the cumulative utility achieved. Red-filled cells indicate timesteps at which a policy is unfair; yellow-filled cells indicate instances in which the failsafe is triggered, i.e., the is decision made exclusivley for the sake of satisfying the fairness constraint. The myopic decisions of **Greedy But Fair** force unfortunate `reject` decisions at timesteps 4 and 8, ultimately leading to its low utility in comparison to **Optimal**.

of the entire sequence of candidates. In this section we introduce a randomized policy as well as two policies trained on historical data. We design these policies with the goal of maximizing cumulative utility subject to the cumulative constraints.

Randomized Positive Override

We open the discussion with a statistical policy called Randomized Positive Override (RPO). At a high-level, this policy aims to increase the probability of `accept`ing candidates that belong to groups with low acceptance rates. This policy is appropriate when the fairness constraint tests whether the selection rates between the two groups exceeds some ratio, such as in the $^4/_5$-Rule used by the U.S. Equal Employment Opportunity Commision (Commission 1978).

Formally, RPO assumes the following model. Let there be two groups of candidates, `C1` and `C2`. Define the *acceptance rate* of a group, `C`, to be the number of candidates `accept`ed from `C` as the number of candidates arriving from `C` goes to infinty. Let θ and ϕ be the True acceptance rates for `C1` and `C2`, respectively, and assume π_t follows a group-dependent Bernoulli distribution with the True parameters, i.e.,

$$Y_t \sim \begin{cases} Bern(\theta) & \text{if } c_t = \texttt{C1} \\ Bern(\phi) & \text{if } c_t = \texttt{C2}, \end{cases}$$

where π_t `accept`s a candidate if $Y_t = 1$. We would like the *impact ratio*, $\rho = \frac{\phi}{\theta}$, of RPO to be close to some target value, η (e.g., $\frac{4}{5}$). Finally, let $\hat{\theta}$ and $\hat{\phi}$ be the empirical acceptance rates of the two groups *under the* UNFAIR *policy*. Without loss of generalized, assume $\hat{\theta} > \hat{\phi}$.

RPO operates as follows. When a candidate arrives from `C1`, RPO makes the same decision as UNFAIR. Similarly, when a candidate arrives from `C2` that the UNFAIR policy would `accept`, RPO also `accept`s the candidate. Otherwise, draw a random variable, Z_t as follows:

$$Z_t \sim Bern\left(\frac{(\eta + \epsilon(t) - \hat{\rho}_t)\hat{\theta}_t}{1 - \hat{\phi}_t}\right),$$

where the function ϵ controls how aggressively the impact ratio should be forced to η. The only requirement on this function is that it be a decreasing function that converges to 0. If $Z_t = 1$, RPO `accept`s the candidate. In practice, we choose $\epsilon(t) = e^{-\kappa t}$ where κ controls the convergence rate. A pseudocode specification of RPO appears in Algorithm 1. In the algorithm, we constrain the parameter to the Bernoulli distribution used for sampling so that it lies in the interval $[0, 1]$.

We prove that, in the limit, this procedure drives the acceptance rate for `C2` to $\eta\theta$ almost surely, so that the impact ratio of the RPO policy converges to η. In the following, define $Y'_t = 1$ if RPO `accept`s the t^{th} candidate of `C2`; otherwise $Y'_t = 0$.

Theorem 1. *If* $\epsilon(t) \to 0$ *then* $\frac{1}{t}\left(\sum_{i=1}^{t} Y'_t\right) \overset{a.s.}{\to} \eta\theta$

Proof. Set $Q_t = \mathbb{E}\left[Y'_t \mid \hat{\theta}_t, \hat{\phi}_t\right]$. Then:

$$Q_t = \phi + (1 - \phi)\mathbb{E}\left[Z_t \mid \hat{\theta}_t, \hat{\phi}_t\right] \tag{1}$$

$$= \phi + \hat{\theta}_t \cdot \left(\frac{1 - \phi}{1 - \hat{\phi}_t}\right)(\eta + \epsilon(t) - \hat{\rho}_t) \tag{2}$$

By the strong law of large numbers, $\hat{\theta}_t \overset{a.s.}{\to} \theta$ and $\hat{\phi}_t \overset{a.s.}{\to} \phi$. Hence $Q_t \overset{a.s.}{\to} \eta\theta$.

Setting $R_t = Y'_t - Q_t$, we have that $\mathbb{E}\left[R_t \mid \hat{\theta}_t, \hat{\phi}_t\right] = 0$. Moreover, $\mathbb{E}\left[R_t^2\right]$ is bounded by a constant, so by the strong

law of large numbers for martingale differences (Feller 1971, VII.9, Theorem 3), we have that

$$\frac{1}{t}\sum_{i=1}^{t}(Y_i' - Q_i) \overset{\text{a.s.}}{\to} 0$$

Because $Q_t \overset{\text{a.s.}}{\to} \eta\theta$, we have that $t^{-1}\sum_{i=1}^{t} Q_i \overset{\text{a.s.}}{\to} \eta\theta$ as well, and hence $t^{-1}\sum_{i=1}^{t} Y_i' \overset{\text{a.s.}}{\to} \eta\theta$. □

When deploying RPO, we still ensure that its decisions satisfy the cumulative fairness constraints by triggering the failsafe as necessary. To better understand the complex trade-off between reduction of utility coming from such tiggers, we further analyze a simplified version of the RPO policy, which provides explicit formulae for the expected costs of the policy.

The analysis appears in the appendix.

Algorithm 1 Randomized Positive Override

Input: UNFAIR policy's decision, G_t, fairness constraint, V, decision history, ψ_{t-1},candidate of class `C`
Output: Decision π_t
if $V(\psi_{t-1} \cup \texttt{accept}) == \texttt{fail}$ **then**
  $\pi_t = \texttt{reject}$
else if $V(\psi_{t-1} \cup \texttt{reject}) == \texttt{fail}$ **then**
  $\pi_t = \texttt{accept}$
else if $\texttt{C} == \texttt{C2} \wedge G_t = 0$ **then**
  $\hat{\rho}_t = \hat{\phi}_t/\hat{\theta}_t, D_t \sim Bern\left(\frac{(\eta+\epsilon(t)-\hat{\rho}_t)\hat{\theta}_t}{1-\hat{\phi}_t}\right)$
  if $D_t == 1$ **then**
    $\pi_t = \texttt{accept}$
  else
    $\pi_t = \texttt{reject}$
  end if
else
  $\pi_t = G_t$
end if

Imitation Learning

Given a full candidate sequence, rather than one candidate at a time, it is possible to compute the sequence of decisions that globally maximizes utility subject to the fairness constraint. The dynamic programming algorithm for computing the optimal sequence, OPT, runs in $\mathcal{O}(Cn^2)$ time and space for a sequence of length n and where C is the cost of evaluating the fairness audit.

Therefore, it is possible train a policy to *mimic* OPT on simulated or historical data. We call this policy Imitation Learning (IL). To train IL, we simulate sequences of candidates and use OPT to compute the optimal sequence of decisions. Then we train a binary classifier to predict this sequence. Features used during training include statistics summarizing the current candidate, the sequence of candidates observed so far and the impending `accept`/`reject` decision.

Learning to Search

The fair online selection problem can be viewed as an instance of structured prediction. In this light, we also investigate training via learning to search. In this work, we experiment with LOLS, a member of the family of learning to search algorithms (Chang et al. 2015). Like in IL, we train a binary classifier, h_L. However, the classifier is trained on a sequence of cost sensitive examples generated as follows. Let S be a sequence of candidates. To generate training example t, we use h_L as a policy to decide whether to `accept` or `reject` the first $t-1$ candidates. Then, at time t, we hallucinate two futures: in one the candidate is `reject`ed and in the other, the candidate is `accept`ed. Then, for each future, we run a *reference* policy (we select either OPT or GBF as the reference) to make decisions for the remainder of the candidates. A cost-sensitive training instance is then generated that includes the state at time t (including candidate x_t) and a label that is determined by which hallucinated future achieved higher utility. The weight of this training instance is equal to the difference in utility between the two futures. We call this policy L2S and note that it was shown to be successful in recent work that studied online selection in the batched setting (Gupta et al. 2021).

Experiments

In this section, we study fair online selection empirically. First, we compare our proposed policies on real and synthetic data. We analyze the learned policies in an attempt to better understand their decision making strategies. Noticing the effectiveness of GBF, we analyze how group score distributions and group proportions effect utility maximization. Finally, we compare fairness of all policies, and also demonstrate that in-processing methods for learning fair classifiers are not sufficient for guaranteeing fairness in the online setting. Our synthetic experiments are inspired, in part, by recent work that highlights computation as a means for problem diagnosis (Abebe et al. 2019).

Setup and Audit In each experiment, there are a sequence of candidates that arrive online (i.e., one at a time). The candidate arriving at timestep t has score s_t. We use the score and a score threshold, δ, to compute the utility of `accept`ing and `reject`ing the candidate in one of two ways: $U_t(\texttt{accept}) = |s_t - \delta|$ and $U_t^{(\text{exp})}(\texttt{accept}) = (1 + (|s_t - \delta|))^{\gamma}$ where the second represents the case where the utility of `accept`ing a candidate with a high score earns exponentially more utility than `accept`ing a candidate with a score close to the threshold. In either case, we define $U(\texttt{accept}) = -U(\texttt{reject})$. Each candidate belongs to one of two groups: `C1` or `C2`. Throughout the decision making process, each policy must satisfy demographic parity—in particular, the $^4/_5$-Rule. Formally, let $\hat{\phi}_n, \hat{\theta}_n$ be the acceptance rate among candidates of `C1` and `C2`, respectively. Then, any policy must satisfy $\min(\hat{\phi}_n, \hat{\theta}_n) \geq \frac{4}{5}\max(\hat{\phi}_n, \hat{\theta}_n)$, or in words, the lower acceptance rate must be at least $^4/_5$ the higher rate. We choose this fairness constraint because it is conceptually simple, it is related to other fairness audits based on selection rates (Raghavan et al. 2019), and is used in legal standards (Commission 1978).

Since our work centers on online decision making, the fairness audit must be robust when the number of candidates seen so far from any group is small. Therefore, we

apply an audit that uses Bayes Factors (Kass and Raftery 1995), as they allow for the incorporation of a prior, and are capable of representing uncertainty. To do so, we calculate the likelihood ratio between two binomial distributions: the first has success parameter drawn uniformly at random between $[0, \frac{4}{5}\max\{\hat{\phi}_n, \hat{\theta}_n\}]$ and the second between $[\frac{4}{5}\max\{\hat{\phi}_n, \hat{\theta}_n\}, 1]$. If the ratio is greater than 10[1] then the decision maker violates the fairness constraint. We use Monte Carlo integration to estimate the Bayes Factors. Because of variance in these estimates, a policy may appear to temporarily violate the fairness threshold by a small amount.

Utility Maximization

First, we compare all proposed polices with respect to utility maximization on 3 real datasets (below) and 1 synthetic dataset.

1. **Compas** (Larson, Roswell, and Atlidakis 2017): The COMPAS recidivism prediction dataset compiled by ProPublica lists data on 5278 criminal offenders screened using this risk assessment tool in Broward County, Florida, during 2013-14. We set gender and race as the protected attribute (in separate set of experiments).
2. **German** (Dua and Graff 2017): This dataset from the UCI repository consists of a 1000 instances, each with 20 attributes and assigned to good or bad credit risk. We set gender and age (≥ 25 and < 25) as the protected attribute in independent experiments.
3. **Income** (Dua and Graff 2017): This dataset extracted from the 1994 Census database contains 14 attributes on demographic information from 45221 adults and whether each person earns $\geq$ 50K USD. We set gender and race as the protected attribute.

We divide each real dataset into train, development and test splits and train a logistic regression classifier on the training set. The output of the classifier is the *score* of the candidate. The synthetic dataset contains candidates belonging to `C1` and `C2`. The scores of the `C1` candidates are drawn from a Beta distribution with $\mu_1 = 0.83$ and the scores of the `C2` are drawn from a Beta with $\mu_2 = 0.16$; for both groups,$\tau = 6$. There are an equal number of candidates from both groups. For all 4 datasets, we use the second (exponentiated) utility function defined above.

IL and L2S We experiment with two trained policies: 1. imitation learning (IL), and 2. learning to search (L2S). We train both policies offline on designated training data. For each dataset, we train 10 models. We evaluate each model on the corresponding development set and select the best performing, which is, in turn, used in experiments on the test set.

In training L2S on the real data, we use a mixture reference policy that chooses GBF 90% of the time and the learned policy otherwise. For the simulated data, we set the reference to be OPT, the dynamic program that computes an optimal fair selection of candidates. We find that these choices of reference policy are most effective. While we hypothesize that this could be related to local optimality of the reference (as mentioned in previous work (Chang et al. 2015)), we leave investigation of how to choose appropriate reference policies for future work.

Our implementation deviates from the originally proposed LOLS algorithm in two ways. First, during training, we only roll out the reference policy for 20 steps, rather than rolling out the reference policy for the entire sequence length. We do this because the training sequences are long (100s to 1000s of candidates), and thus any reference policy is likely to be locally optimal (i.e., at time t, either decision may lead to the same cumulative utility after a full roll out). Since the training set is of a fixed size, this helps our model become more robust to arbitrarily sized testing sequences. Finally, rolling out 20 steps, rather than to the end of the candidate sequence, drastically improves the speed of training.

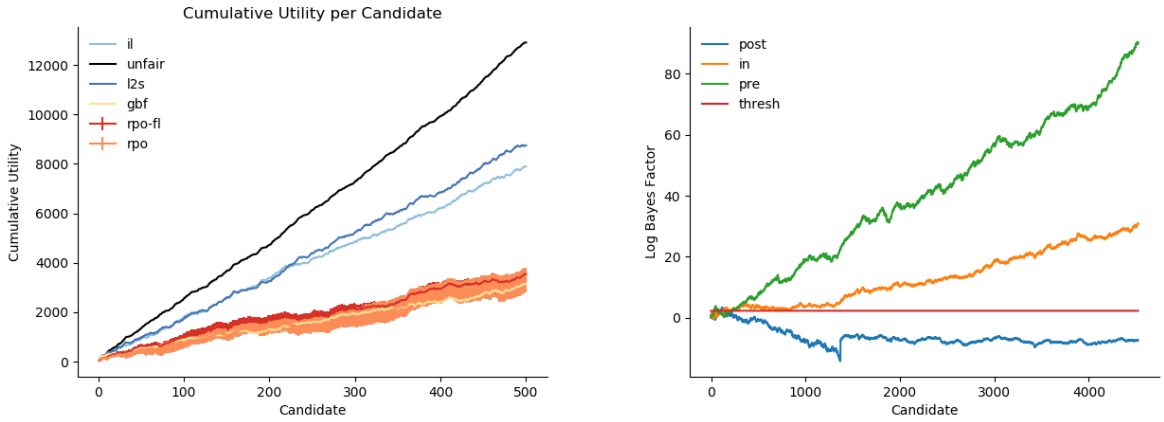
For IL implementation, we use a 3-layer feed-forward neural network with 24 hidden units per layer and Relu activations. We train with a cross-entropy loss using Adagrad with a learning rate of 0.1. For L2S implementation, the architecture is composed of 2 parallel sequences of layers, one representing each potential action. Each sequence has two layers of 24 hidden units each and Relu activations. The output of each sequence is a 24 dimentional vector. The two output vectors are substracted (as in ranking) and passed through a final layer with sigmoid activation. We train with a cross-entropy loss using Adagrad with a learning rate of 0.1.

Baseline Selection Policies We test 4 selection policies that require no training:1. UNFAIR, 2. greedy but fair (GBF), 3. randomized positive override (RPO), and 4. the randomized positive override with feedback loop (RPO-FL). RPO-FL is the same as RPO except that instead setting $\hat{\theta}_t$ and $\hat{\phi}_t$ to be the empirical estimates of UNFAIR's selection rates for candidates of the two groups, they are the policy's own empirical selection rates. We run RPO and RPO-FL 10 times on each test set and report means and standard deviations.

Results We run each policy on each dataset and measure cumulative and average utility per candidate. A visual representation of the cumulative utility on the simulated dataset appears in Figure 2(a). For the simulated data, the Figure shows that the learned policies dominate the baseline strategies, and that L2S outperforms IL. This echos recent results showing that L2S is effective in the batched online setting (Gupta et al. 2021). UNFAIR (the top-most black line), represents the maximum achievable utility per candidate. Of the baseline approaches, RPO and RPO-FL achieve higher cumulative utility than GBF, however RPO exhibits relatively high variance across the 10 runs.

Results for the remaining datasets appear in Table 1. Interestingly, on the remaining datasets, GBF is a strong contender while RPO and RPO-FL are relatively weaker. This is related to the fact that the GBF is competitive with OPT: since it is possible to largely follow a greedy policy, randomly accepting candidates from the group with lower acceptance rate (as in RPO and RPO-FL), is not required. The table also shows that the learned methods are top-performers across datasets, highlighting their robustness and the viability of learning

[1] A Bayes Factor above 10 constitutes "strong" evidence according to Kass and Raftery (1995)



(a) (b)

Figure 2: (a) **Cumulative Utility on Simulated Data.** The learned policies L2S and IL outperform the non-learned policies. RPO and RPO-FL outperform GBF however RPO exhibits high variance across 10 runs. Error bars indicate 1 standard dev. (b) **Evaluation of offline fairness intervention algorithms in the online setting, on income dataset.** Pre and in-processing algorithms aggressively violate the audit threshold demonstrating their insufficiency. Post-processing also violates the threshold initially.

Policy	Income-G	Income-R	Compas-G	Compas-R	German-A	German-G	Simulated
L2S	0.89 ± 0.00	0.99 ± 0.01	0.92 ± 0.00	0.94 ± 0.00	0.97 ± 0.00	1.00 ± 0.00	0.71 ± 0.01
IL	0.87 ± 0.02	0.99 ± 0.01	0.92 ± 0.00	0.94 ± 0.00	0.97 ± 0.00	1.00 ± 0.00	0.63 ± 0.02
RPO	0.89 ± 0.00	0.97 ± 0.00	0.80 ± 0.02	0.93 ± 0.01	0.98 ± 0.01	1.00 ± 0.00	0.27 ± 0.04
RPO-FL	0.89 ± 0.00	0.98 ± 0.00	0.89 ± 0.01	0.94 ± 0.00	0.98 ± 0.01	1.00 ± 0.01	0.31 ± 0.02
GBF	0.90	0.97	0.92	0.93	0.97	1.00	0.24
UNFAIR	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 1: **Competitive ratios of cumulative utility.** These ratios represent the average and standard deviation of cumulative utility achieved be each policy reported as a fraction of the maximum possible cumulative utility (i.e., achieved by UNFAIR).

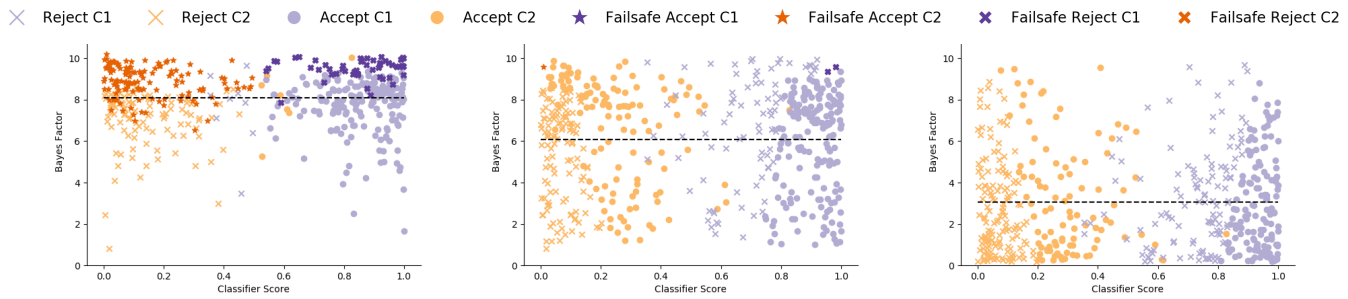
selection policies. Since GBF is frequently competitive with OPT, we conduct experiments to understand the conditions under which GBF maximizes utility (details in appendix). We find that GBF (and other methods) are close to optimal when the means of the score distributions for `C1` and `C2` are close. For cases where a marked difference in average ability between two groups is unexpected (e.g., in loan disbursement between men and women), this result suggests that learned policies are unnecessary. Instead, greedily selecting candidates while abiding by fairness mandates yields close to optimal utility. Moreover if the model assigning scores to candidates *does* show significant difference in average scores between candidates from `C1` and `C2`, it may be more appropriate to improve the scoring model than compensate with a complex selection policy.

Analysis of Learned Policies

We study the learned policies to better understand their decision making strategies. In Figure 3, we plot each candidate as an (x, y) pair of their score and the Bayes Factor at the point in time in which that candidate arrives in the candidate sequence. Here, the Bayes Factor for a candidate arriving at time t is a function of the policy's decision for all candidates arriving before t. The colors of the points indicate group membership and the mark type (either `o` or `x`) indicates whether that candidate was either `accept`ed or `reject`ed, respectively. The dotted black lines represents the average Bayes Factor achieved. We compare the IL and L2S to GBF.

The Figure reveals a number of interesting properties about the three policies. First, we observe that GBF has a higher average Bayes Factor than either of the learned methods, and that IL has a higher average Bayes Factor than L2S. In fact, this suggests that among the most competitive methods, operating close to the audit threshold is *inversely proportional* to cumulative utility. This makes intuitive sense: methods that operate more fairly (i.e., further from the audit threshold) are more likely to be able to `accept` or `reject` strings of candidates with extreme scores without violating the fairness constraints. This is in concert with the observation that L2S triggers the failsafe the *least* among the three methods.

IL and L2S seem to learn soft per-group thresholds for `accept`ing candidates, which can be seen by the visual columns of `accept`ed and `reject`ed candidates. Consider Figures 3(b) and 3(c) and the `accept` decisions for `C2` (purple marks). In Figure 3(c) we observe a fuzzy decision boundary with positive slope, i.e, as the Bayes Factor in-



(a) GBF (b) IL (c) L2S

Figure 3: **Classifier scores vs. Bayes Factors on simulated data.** Orange markers represent candidates from C1 and purple markers represent candidates from C2. Circles and 'x's represent accepted and rejected candidates, respectively. Bolded 'x's and '⋆' markers represent decision made by the failsafe. The dotted black line represents the average Bayes Factor achieved.

creases, L2S is less likely to accept a high scoring candidate. In Figure 3(b), a similar fuzzy decision boundary exists however it appears to have much steeper slope. This is likely to account for the higher average Bayes Factor for IL. These structures are in sharp contrast to GBF which exhibits no such structure.

Policy Fairness

While our work focuses on maximizing utility subject to passing the fairness audit at all times, we also study the average Bayes Factor achieved by all methods. Table 2 contains these statistics for the simulated dataset. Unsurprisingly, RPO maintains the lowest Bayes Factor (i.e., is the most fair). However, RPO-FL achieves the second *highest* average Bayes Factor, potentially due to the instability associated the post-processor's average rates of hiring candidates from both classes. Interestingly, L2S achieves the second lowest Bayes Factor while achieve the highest cumulative utility (Table 1) suggesting that it is arguably the best overall policy. GBF achieves the highest average Bayes Factor, as expected.

Average Bayes Factor				
RPO	L2S	IL	RPO-FL	GBF
3.1 ± 2.7	4.6 ± 2.6	6.8 ± 2.1	7.2 ± 2.0	8.1 ± 1.4

Table 2: **Average Bayes Factor.** The table contains the Bayes Factor per policy averaged over the candidate sequence for single run of each algorithm on the Simulated data.

Related Work

The field of Algorithmic Fairness has enjoyed significant research attention, but online selection problems with fairness constraints have only recently become the subject of increased study. The most similar to ours is on fair online ranking where candidates arrive in batches and the ranking algorithm must satisfy a cumulative fairness constraint (Gupta et al. 2021). However, in our work, a single candidate—rather than a batch of candidates—is available at each timestep. Other closely related work is inspired by the *secretary problem*, in which the goal is to select the maximum value from a sequence of known and bounded length (Dynkin 1963; Stoyanovich, Yang, and Jagadish 2018; Salem and Gupta 2019; Kleinberg and Raghavan 2018). However, in our setting, the sequence is of unknown size, and the number of candidates to select is unbounded. Additionally, in our work a policy must satisfy fairness constraints at all timesteps, rather than only at the end of the sequence. Less closely related are studies focused on fairness in online settings, but for problems other than selection (Joseph et al. 2016; Kannan et al. 2017; Bechavod et al. 2019; Lakkaraju et al. 2017; Kilbertus et al. 2020; Cayci, Gupta, and Eryilmaz 2020).

The selection policies we study can be classified as fair post-processing methods (Hardt, Price, and Srebro 2016; Kamiran, Karim, and Zhang 2012; Canetti et al. 2018). In seminal work on post-processing, a randomized classifier is derived from an existing (possibly) unfair classifier, so as to achieve required fairness criteria in expectation (Hardt, Price, and Srebro 2016). However, methods that guarantee fairness in expectation are inappropriate in our setting, in which fairness constraints must be satisfied at all timesteps. Moreover, our work centers on the online setting.

Conclusions

We study the online selection problem with cumulative fairness constraints. We propose deterministic, randomized and learned policies and demonstrate that the learned selection policies achieve the highest utility. However, the greedy algorithm—GBF—is competitive with the optimal sequence of selections whenever the distributions of scores of candidates from C1 and C2 have similar means. Our experiments underscore the importance of deep consideration of the problem at hand, the tendencies of the scoring model, and the characteristics of the expected input sequence, before selecting a policy. In our setup, the goal of the selection policy is to maximize utility. However, using utility maximization as the primary objective should be carefully considered in real-world scenarios. For this reason, RPO and RPO-FL—which have provable convergence properties when it comes to balancing acceptance rates—merit further investigation, despite performing worse than methods like L2S.

References

Abebe, R.; Barocas, S.; Kleinberg, J.; Levy, K.; Raghavan, M.; and Robinson, D. G. 2019. Roles for Computing in Social Change.

Bechavod, Y.; Ligett, K.; Roth, A.; Waggoner, B.; and Wu, S. Z. 2019. Equal Opportunity in Online Classification with Partial Feedback. 8974–8984.

Bogen, M. 2019. All the Ways Hiring Algorithms Can Introduce Bias.

Canetti, R.; Cohen, A.; Dikkala, N.; Ramnarayan, G.; Scheffler, S.; and Smith, A. 2018. From Soft Classifiers to Hard Decisions: How fair can we be?

Cayci, S.; Gupta, S.; and Eryilmaz, A. 2020. Group-Fair Online Allocation in Continuous Time. *Adv. Neural Inf. Process. Syst.*, 33.

Chang, K.-W.; Krishnamurthy, A.; Agarwal, A.; Daumé, H.; and Langford, J. 2015. Learning to Search Better than Your Teacher. In *ICML*.

Commission, E. E. O. 1978. Uniform guidelines on employee selection procedures (1978). *Federal Register*, 43(166): 38290–38315.

Correa, J.; Dütting, P.; Fischer, F.; Schewior, K.; and Ziliotto, B. 2020. Streaming Algorithms for Online Selection Problems. *arXiv preprint arXiv:2007.06110*.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Dynkin, E. B. 1963. Optimal choice of the stopping moment of a Markov process. In *Doklady Akademii Nauk*, volume 150, 238–240. Russian Academy of Sciences.

Elmachtoub, A. N.; and Levi, R. 2015. From cost sharing mechanisms to online selection problems. *Mathematics of Operations Research*, 40(3): 542–557.

Elyounes, D. A. 2019. Bail or Jail? Judicial Versus Algorithmic Decision-Making in the Pretrial System. *Colum. Sci. & Tech. L. Rev.*, 21: 376.

Feller, W. 1971. *An introduction to probability theory and its applications. Vol. II.* Second edition. New York: John Wiley & Sons Inc.

Friedler, S. A.; Scheidegger, C.; Venkatasubramanian, S.; Choudhary, S.; Hamilton, E. P.; and Roth, D. 2019. A comparative study of fairness-enhancing interventions in machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, 329–338.

Gupta, A.; Johnson, E.; Roy, A. K.; Payan, J.; Kobren, A.; Panda, S.; Tristan, J.-B.; and Wick, M. 2021. Online Post-Processing in Rankings for Fair Utility Maximization. In *Web Search and Data Mining (WSDM)*.

Hardt, M.; Price, E.; and Srebro, N. 2016. Equality of Opportunity in Supervised Learning. *CoRR*, abs/1610.02413.

Joseph, M.; Kearns, M.; Morgenstern, J. H.; and Roth, A. 2016. Fairness in Learning: Classic and Contextual Bandits. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*, 325–333. Curran Associates, Inc.

Kamiran, F.; Karim, A.; and Zhang, X. 2012. Decision Theory for Discrimination-Aware Classification. In *2012 IEEE 12th International Conference on Data Mining*.

Kannan, S.; Kearns, M.; Morgenstern, J.; Pai, M.; Roth, A.; Vohra, R.; and Wu, Z. S. 2017. Fairness Incentives for Myopic Agents. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, EC '17, 369–386. New York, NY, USA: Association for Computing Machinery. ISBN 9781450345279.

Kass, R. E.; and Raftery, A. E. 1995. Bayes Factors. *Journal of the American Statistical Association*, 90(430): 773–795.

Kilbertus, N.; Rodriguez, M. G.; Schölkopf, B.; Muandet, K.; and Valera, I. 2020. Fair Decisions Despite Imperfect Predictions. volume 108 of *Proceedings of Machine Learning Research*, 277–287. Online: PMLR.

Kleinberg, J.; and Raghavan, M. 2018. Selection Problems in the Presence of Implicit Bias.

Lakkaraju, H.; Kleinberg, J.; Leskovec, J.; Ludwig, J.; and Mullainathan, S. 2017. The Selective Labels Problem: Evaluating Algorithmic Predictions in the Presence of Unobservables. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, 275–284. New York, NY, USA: Association for Computing Machinery. ISBN 9781450348874.

Larson, J.; Roswell, M.; and Atlidakis, V. 2017. Data and Analysis for 'Machine Bias'.

Lum, K.; and Isaac, W. 2016. To predict and serve? *Significance*, 13(5): 14–19.

Meijer, A.; and Wessels, M. 2019. Predictive policing: Review of benefits and drawbacks. *International Journal of Public Administration*, 42(12): 1031–1039.

Raghavan, M.; Barocas, S.; Kleinberg, J.; and Levy, K. 2019. Mitigating Bias in Algorithmic Hiring: Evaluating Claims and Practices.

Romei, A.; and Ruggieri, S. 2011. A multidisciplinary survey on discrimination analysis.

Salem, J.; and Gupta, S. 2019. Closing the Gap: Group-Aware Parallelization for the Secretary Problem with Biased Evaluations.

Stoyanovich, J.; Yang, K.; and Jagadish, H. 2018. Online Set Selection with Fairness and Diversity Constraints.

Wick, M.; Panda, S.; and Tristan, J.-B. 2019. Unlocking fairness: a trade-off revisited. *Advances in Neural Information Processing Systems*.