CS 226r Final Project: Achieving Fairness in the Multi-Round Online Gaming Context

Gary Wu & Jessica Chen

Harvard John A. Paulsen School of Engineering and Applied Sciences

1 Abstract

Inspired by work from Bechavod et al [1], we are interested in seeing how fairness falls apart under Online Learning settings with strategic gaming involved, and also how fairness algorithms can be used to make them fair. The paper discusses how individuals, when aware of the current regression model, may strategically modify the reporting of their own features to achieve a more favorable outcome. Each agent may then modify the reporting of their own utility for the next round to increase their chance of selection, though this modification incurs a certain cost. Advantaged agents may have less severe cost functions, while disadvantaged agents may incur heavier costs for modification. We find that the standard online gaming setting is not fair, and introduce changes to ensure fairness across multiple metrics.

We have made our fairness simulation code publicly available here: https://github.com/JessSanChen/226-fair-gaming.

2 Introduction

The loan-application process is a widely studied topic within the Algorithmic Fairness literature. Fairness metrics and objectives are measured based on protected attributes, and much work in the field involves how to achieve these metrics especially with real-world data. In essentially all of this work though, we assume that applicant information is stagnant, and that there is only one chance for the bank or other loan-providing entity to provide a sufficiently fair classifier or model. We look at a paper within the EconCS literature titled "Gaming Helps! Learning from Strategic Interactions in Natural Dynamics", that provides a framework for allowing the learner (bank or loan-provider) to interact with agents (loan-applicants) in a back-and-forth manner with the condition that agents are able to strategically alter their "features" to improve their chances of positive classification.

We believe that this is an important addition to the Fairness literature given that any classifier or model put out to the real world will likely be strategically gamed in some way by agents who have utility-maximizing incentives. The intersection of EconCS and Fairness presents novel opportunities and challenges, and we wish to tackle just one of them by analyzing the Online Gaming setting and how we can make it more fair.

Ultimately the paper shows that this sequential process allows the learner to uncover which features are truly meaningful and can predict true ability to pay back a loan, and distinguish them

from non-meaningful features, which can be gamed (agents can alter them for positive classification even though it doesn't increase their chances of paying back the loan). Thus, the paper introduces a way to learn useful classifiers even in settings in which agents can strategically game their chances of classification. We believe this resembles how things work in the real world, and want to analyze if fairness can be ensured within this Online Gaming framework.

To do this, we simulate the Gaming Helps! environment with agents instantiated based on a real dataset, and have them strategically game their features in order to improve their chances of a positive classification by the classifier. Our simulation first shows that without extra fairness modifications, the classifier under the Gaming Helps! framework does not achieve fairness as measured by the criteria we outline later.

Through pre and post-processing algorithms, however, we can achieve much better fairness measures across multiple criteria. We hope that our work can inspire further research into the intersection of algorithmic fairness and strategic agents.

3 Related Work & Literature Review

3.1 Gaming Helps! Learning from Strategic Interactions in Natural Dynamics

This is the primary paper we wish to analyze and extend. It proposes a framework to allow a learner (classifier) to recover what the meaningful and non-meaningful features are in a classification problem, specifically given that agents can strategically alter their features to improve their chances of positive classification [1]. Agents are given a cost function and utility budget, are initialized with a feature vector $x \in [-1, 1]^d$ and have some true label y that indicates their true ability of paying back a loan. Meaningful features accurately predict an agent's true label, for example a higher income definitively improves someone's ability of paying back a loan and cannot be "gamed". However, other factors such as FICO credit scores may not as accurately predict this true ability as they can be gamed by those who have more knowledge of the scoring system, and these people usually fall into advantaged populations. Non-meaningful features can be gamed. The sequential recovery framework is as follows:

1. **Learner posts their estimate of the true regression parameter** $\beta* \in [-1, 1]^d$

   The true regression parameter is the weighing of how important each feature is to actually paying back loans.

2. **Agents observe posted regression parameter and alter their features**

   Agents are instantiated with a set of features and are given a cost function and budget each round to alter their features based on expected utility.

3. **Learner classifies agents and uses LSE technique to update regression parameter**

   The idea is that by comparing reported feature changes and an agent's true label, we can uncover which features are truly meaningful to the classification problem.

4. **Repeat with new set of agents until true regression parameter is recovered**

   Refer to paper for more specific mathematical details.

3.2 Equality of Opportunity in Supervised Learning

This paper lays the foundation for analyzing Equality of Opportunity in supervised learning problems. We will use the definition of Equal opportunity as defined in the paper [2]:

Definition 2.2 (Equal opportunity). We say that a binary predictor $\hat{Y}$ satisfies equal opportunity with respect to A and Y if $\Pr\{\hat{Y} = 1|A = 0, Y = 1\} = Pr\{\hat{Y} = 1|A = 1, Y = 1\}$.

The paper outlines a framework and checklist to consider when working on fair supervised learning problems.

3.3 Fairness in Credit Scoring: Assessment, Implementation and Profit Implications

This paper explores fairness specifically in the Credit-Scoring context, providing an analysis on statistical fairness criteria given that in the credit-scoring problem, many of the factors and features used to decide loan-worthiness are not independent of each other or conflict with each other [3]. Then it provides an overview of implementing fairness processors as well as a comparison of each.

4 Model & Methodology

We use the AIF360 German Dataset with real-world classification data, and implement the sequential framework introduced in the Gaming Helps! paper. Specifically, we can represent the different individuals expressed in the dataset as the agents trying to strategically "game" the classifier. At each round, we can take a row from the dataset and generate a cost function and utility budget for them drawn from different distributions depending on whether they are a member of the protected attribute. For our purposes, we will use age as the protected attribute. The classifier we train in this context will be the equivalent to the learner.

Each agent is instantiated with several features and a true label, along with group status. Group status indicates whether the agent belongs to a disadvantaged racial group. The true label is whether or not the agent will default on the loan. Meaningful features are ones that, when changed, affect the true label; these may include features such as payment history, debt-to-income ratio, financial stability, and more. Non-meaningful features are ones that do not affect the label, and are thus susceptible to gaming; these may include credit score gaming, short-term credit management, or superficial financial moves.

To simplify our simulation, we select representative features from either category and make assumptions on credit assessment. For the meaningful feature, we use the credit history attribute. For the non-meaningful feature, we use savings (for the purpose of gaming). This selection also satisfies the assumption for non-heavily-correlated features.

Agents are also instantiated with a linear cost function and a total budget, to be used when strategically modifying their features. Agents of disadvantaged groups sample their cost function weights from a higher-mean distribution to simulate higher difficulty of gaming.

In each training round, the learner does not immediately know which features are meaningful vs. non-meaningful and weighs them similarly; it updates weights to reflect the best estimate parameter

by solving for the estimated regression based on agent observations over time. The learner then publicly announces its LSE regression vector. Agents may then observe this regression vector, then choose to strategically invest in one of their features. Based on the classification result, the agent receives a certain utility and the learner receives feedback.

We retrain a classifier at each round. To evaluate fairness, we first run the simulation without any additional fairness measures, and then we have separate runs with the following fairness implementations:

4.1 Re-Weighing

Re-Weighing is a pre-processing fairness technique that modifies the dataset before training to remove potential bias. The technique will assign weights to each group in the 2-by-2 combination of disadvantaged/advantaged groups with the positively/negatively classified groups.

4.2 Reject-Option Classification

ROC is a post-processing fairness technique that rejects making a decision in situations where a classification is too uncertain. The core assumption is that much of the discrimination happens at classification decisions extremely close to the threshold, i.e. they could go one way or the other. ROC creates a range for which the classifier should reject making a decision - in practice, this would then lead to a closer examination of the specific individual, outside of the classifier's jurisdiction.

5 Simulation Design

5.1 Overview

Our code is available at https://github.com/JessSanChen/226-fair-gaming.

We take inspiration from the AIF360 python package and examples, and use one of their datasets, the German Dataset. The package provides a simple way to check fairness constraints given privileged attributes and classification results. We extend the framework to include multi-round gaming through agents as well as a learner that not only classifies but also recovers meaningful & non-meaningful features. For the purposes of this project, we alter the German Dataset and drop the features that are not relevant to the goal of determining non-meaningful vs meaningful features.

5.2 Agent Implementation

We represent Agents by rows in the AIF360 dataset, and they are instantiated with the function *strategizing_agents*(). For each row in the dataset, an agent is created with a utility budget to spend toward modifying their features, as well as a cost function to use to decide how much to invest. Based on whether the individual is a member of the privileged class, it will have a cost function and budget from different distributions.

Importantly, there is a distinction between classification and true label. The classification is ultimately a binary outcome of whether or not an agent is classified as loan-worthy, whereas the true label as mentioned in the Gaming Helps paper is a measure of how likely someone is truly able to pay off a loan.

5.3 Learner Implementation

The learner implementation follows closely to the Gaming Helps paper. Each epoch, it will post its estimate of the true latent regression parameters - how much important it perceives each feature to matter to the true label. After agents modify their features, the Learner takes the agent data and updates their β estimate parameters. In the gaming helps paper, the learner does not actually classify the agents since the primary goal of the paper is to recover meaningful/non-meaningful features. We extend the paper such that the learner actually trains a classifier in each round! With this classifier, we can better understand the fairness implications of the paper. For the purposes of this analysis, we train a simple Logistic Regression classifier.

Specifically, the learner takes the first epoch of data and trains on it and updates its parameters. For each subsequent epoch, it appends the history data to the current epoch such that the classifier maintains a holistic representation of all epochs.

5.4 Fairness Implementations

Both Reweighing and Reject-Option-Classification are implemented with the help of the AIF360 package. Reweighing is done to the features of all agents at each Epoch, directly after they have modified their features. Reject-Option-Classification is applied after the classifier is trained each round.

5.5 Design Decisions

1. **Selecting features to keep within simulation:** While the dataset had 11 features, we elected to use 1 meaningful and 1 non-meaningful feature to best measure impact of strategic modification on classification. We chose these by selecting for greater and approximately equal regression coefficients.

2. **Agent budget and strategic modification:** The original gaming mechanisms called for a budget and cost function per agent. Additionally, it assumes that agents will fully invest its budget in modifying just one feature. Since the budgets are per round and agents are not re-used between epochs, we simplified this mechanism by sampling from a Gaussian distribution and comparing to a parametric threshold to determine whether modification occurs. Privileged agents sample from a distribution with a higher mean.

3. **Iterative training updates on new data:** Instead of electing to use other models with warm starts or partial data training capabilities, we used the same base Logistic Regression classifier and trained it on all epochs up until the current one. As such, we keep track of gaming history.

4. **Non-random learner initialization**: Instead of selecting regression parameters at random as in the original gaming mechanism, we initialize the learner by first training it on the first epoch of agents.

5. **Requiring active strategic effort to maintain label:** When we first ran experiments, we found that, contrary to our hypothesis, the majority of strategic modification was from unprivileged agents. This was because, in the given dataset and application, almost all privileged agents (over 25) had good loan credit and are classified for favorable loan rates.

As such, a very small subset would modify for a better classification. To better incentivize modification, we regularly penalize agents' features at the beginning of each round to emulate that, in real life, investing effort into active upkeep of credit history and savings is required to maintain good credit.

6. **Label updates due to strategic investment of effort:** Agents that have the budget for modification may randomly iterate through its features to see which feature, if any, may result in achieving a better classification. If they happen to modify their "meaningful" feature above a parametric threshold, the agent's labels will also update accordingly. This simulates how, in real life, investing effort into the right practices for personal finance may lead to changes in your true probability of default.

7. **Learner recovery of meaningful features:** As a result of the label updates, the learner can then fit to the modified features and labels on the next epoch. The learner will see that modifications of the meaningful feature may lead to label updates, whereas modifications of the non-meaningful features will not; this simulates meaningful feature recovery.

5.6 Modifications to the AIF360 Source Library

1. **Dropping features:** Since there is no built-in method to drop features, we created a novel method to load a dataset and perform alternative pre-processing. We take advantage of the class's ability to convert into a Pandas DataFrame. The result outputs a StandardDataset object with the same attributes.

2. **Appending historical data to update learner on partial data:** Since the learner iteratively updates its regression parameter on new data, we require a method that can append datasets together. Since there is no built-in method to do so, we built a novel method to append StandardDatasets together.

6 Results

We will first show in our simulation that the baseline learner algorithm does not achieve multiple fairness criteria. Below are the definitions of the multiple fairness criteria we analyze (not including Equalized Opportunity which was defined earlier):

1. **Balanced Accuracy**

   Balanced Accuracy is the arithmetic mean of sensitivity and specificity. Primarily used with imbalanced data, for example, when one of the target classes appears at a much higher frequency than the other.

2. **Statistical Parity Difference (SPD)**

   SPD measures the difference between the proportion of positive outcomes for two groups.

3. **Consistency (individual fairness)**

   Individual fairness metric that measures how similar the labels are for similar instances.

4. **Disparate Impact**

   Disparate Impact measures the ratio of positive outcomes between two groups.

6.1 Impact of Fairness Metrics in No Gaming Simulation:

Table 1: Final fairness metrics for 2 epochs.

	Balanced Accuracy	SPD	Consistency	Disparate Impact	Equalized Opportunity
Vanilla	0.5292	-0.3108	0.6892	-0.2614	0.6219
RW	0.5134	-0.3522	0.6478	-0.3107	0.6150
ROC	1.0000	-0.1340	0.8151	0.0000	0.6268
RW and ROC	1.0000	0.0191	1.0296	0.0000	0.6187

Table 2: Final fairness metrics for 4 epochs.

	Balanced Accuracy	SPD	Consistency	Disparate Impact	Equalized Opportunity
Vanilla	0.0157	-0.0079	0.0221	-0.0066	0.0184
RW	0.0154	-0.0126	0.0174	-0.0114	0.0185
ROC	0.0300	-0.0044	0.0241	0.0000	0.0187
RW and ROC	0.0300	0.0001	0.0302	0.0000	0.0183

From Table 1 and 2, we see that as we increase the number of epochs trained, the fairness metrics are substantially improved. This makes sense as the more data the model sees and the more agents are introduced, the pre and post-processing algorithms have a better impact on fairness. We can think of this as an adjustment period. Further epochs are not necessarily needed on a dataset of this size when agents are not strategically gaming. For the case with no gaming, we find that implementing both Reweighing and ROC is the most effective strategy with the multi-round setup.

6.2 Impact of Fairness Metrics in Gaming Agents Simulation:

Table 3: Final fairness metrics for 2 epochs of gaming agents.

	Balanced Accuracy	SPD	Consistency	Disparate Impact	Equalized Opportunity
Vanilla	0.5386	-0.4514	0.5486	-0.3899	0.6257
RW	0.5073	-0.2583	0.7417	-0.2349	0.6265
ROC	1.0000	-0.1594	0.7828	0.0000	0.6122
RW and ROC	1.0000	-0.0162	0.9774	0.0000	0.6108

Table 4: Final fairness metrics for 4 epochs of gaming agents.

	Balanced Accuracy	SPD	Consistency	Disparate Impact	Equalized Opportunity
Vanilla	0.0155	-0.0044	-0.0041	-0.3899	0.0190
RW	0.0152	-0.0039	0.7417	-0.0036	0.0187
ROC	0.0300	-0.0043	0.0241	0.0000	0.0185
RW and ROC	0.0300	-0.0023	0.0270	0.0000	0.0184

We observe that gaming does not have a significant effect on the fairness metrics. We hypothesize multiple reasons for why the data results may have subverted our hypothesis:

1. **Limited data and repetition of reshuffling:** The German dataset we used has limited data, and overall very few individuals in each group that received a bad credit score rating. As such, in order to submit multiple epochs of agents, we reused the sample dataset multiple times, but reshuffled. As such, we may see repetitive results.

2. **Insufficient incentive to game:** Privileged agents make up the majority of the dataset, and they are not incentivized to game. As such, we would need more unprivileged agents to see more significant differences in fairness metrics.

3. **Suboptimal parameters:** We did not try to find the maxiamlly optimal parameters for our models.

6.3 Combining Reweighing and ROC

We find that combining these two fairness modifications in the gaming context does not really make the model more "fair" than having any one modification on its own. We suspect that this is due to a mix of the below reasons:

1. **Overlapping Adjustment & Interaction Effects**

   It's possible that the bias accounted for by reweighing makes it such that the ROC step no longer deals with the same bias problem.

2. **Diminishing Returns**

   Over time and across multiple adjustments, it's just not possible to maintain the same levels of improvement with respect to fairness.

3. **Combining Methods Might be Unproductive**

   Each type of pre/in/post-processing algorithm is typically meant to target a specific type of bias or fairness problem, and combining them may not be the most useful strategy. We refer to the literature for more on combining fairness modifications.

7 Discussion and Next Steps

Through this project, we have successfully extended the Online Gaming framework. From what we know about the literature, we are one of the **first** to simulate this sequential agent-learner dynamic. Additionally, we have altered the model such that the learner actually trains a classifier at each epoch instead of just learning a new regression parameter (which we also implement). Additionally, we venture into this branch of fairness where gaming is allowed. In typical settings, fairness is calculated on stagnant and non-changing data where individuals don't strategize. However, given that fairness is a real world problem, we must account for what happens in the real world, and a common phenomena is strategic gaming in order for agents to improve their chances of classification. We find that fairness is not initially achieved through this Online Gaming framework and show key results in achieving fairness criteria through both pre and post-processing algorithms.

Given the limited scope of our work, we propose a number of future next steps that we think would be interesting to explore:

1. **Examining other Fairness Metrics**

   The metrics we examine are a good start, but we think that there are additional metrics that are important specifically within the loan-fairness space. Specifically, we reference more in-processing methods that could be used in place of our simple Logistic Regression classifier, as well as other fairness algorithms in the AIF360 library.

2. **Mix of Gaming and Non-Gaming Agents**

   We believe that the underlying assumption that all agents strategically game is unrealistic. Particularly, it would make sense that those from the advantaged group not only have a more successful time gaming their features, but also do it more often than disadvantaged groups.

3. **Larger and Richer Dataset**

   While the German credit scoring dataset is a literature standard, it is severely inhibited by the size of the dataset, with only 1000 rows. As such, it was particularly difficult to run more epochs of agents to better simulate the gaming mechanisms. Furthermore, many relevant features were given in binary ranges, as opposed to more specific values. That being said, we acknowledge the difficulty in sourcing rich datasets in this credit assessment field, and we hope to invest more time on data collection in the future.

References

[1] **Bechavod, Yahav, et al.**, "Gaming helps! learning from strategic interactions in natural dynamics." International Conference on Artificial Intelligence and Statistics. PMLR, 2021.

[2] **Hardt, Moritz, Eric Price, and Nati Srebro**, "Equality of opportunity in supervised learning." Advances in neural information processing systems 29 (2016).

[3] **Kozodoi, Nikita, Johannes Jacob, and Stefan Lessmann.**, "Fairness in credit scoring: Assessment, implementation and profit implications." European Journal of Operational Research 297.3 (2022): 1083-1094.