Runtime Monitoring of Dynamic Fairness Properties

Thomas A. Henzinger
Institute of Science and Technology Austria (ISTA)
Klosterneuburg, Austria
tah@ist.ac.at

Mahyar Karimi
Mahyar.Karimi@ist.ac.at
Institute of Science and Technology Austria (ISTA)
Klosterneuburg, Austria

Konstantin Kueffner
konstantin.kueffner@ist.ac.at
Institute of Science and Technology Austria (ISTA)
Klosterneuburg, Austria

Kaushik Mallik
kaushik.mallik@ist.ac.at
Institute of Science and Technology Austria (ISTA)
Klosterneuburg, Austria

ABSTRACT

A machine-learned system that is fair in static decision-making tasks may have biased societal impacts in the long-run. This may happen when the system interacts with humans and feedback patterns emerge, reinforcing old biases in the system and creating new biases. While existing works try to identify and mitigate long-run biases through smart system design, we introduce techniques for monitoring fairness in real time. Our goal is to build and deploy a monitor that will continuously observe a long sequence of events generated by the system in the wild, and will output, with each event, a verdict on how fair the system is at the current point in time. The advantages of monitoring are two-fold. Firstly, fairness is evaluated at run-time, which is important because unfair behaviors may not be eliminated a priori, at design-time, due to partial knowledge about the system and the environment, as well as uncertainties and dynamic changes in the system and the environment, such as the unpredictability of human behavior. Secondly, monitors are by design oblivious to how the monitored system is constructed, which makes them suitable to be used as trusted third-party fairness watchdogs. They function as computationally lightweight statistical estimators, and their correctness proofs rely on the rigorous analysis of the stochastic process that models the assumptions about the underlying dynamics of the system. We show, both in theory and experiments, how monitors can warn us (1) if a bank's credit policy over time has created an unfair distribution of credit scores among the population, and (2) if a resource allocator's allocation policy over time has made unfair allocations. Our experiments demonstrate that the monitors introduce very low overhead. We believe that runtime monitoring is an important and mathematically rigorous new addition to the fairness toolbox.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; • **Social and professional topics** → **Computing / technology policy**.

KEYWORDS

algorithmic fairness, dynamic fairness, runtime monitor, online statistical estimator

ACM Reference Format:
Thomas A. Henzinger, Mahyar Karimi, Konstantin Kueffner, and Kaushik Mallik. 2023. Runtime Monitoring of Dynamic Fairness Properties. In *2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23), June 12–15, 2023, Chicago, IL, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3593013.3594028

1 INTRODUCTION

A majority of works in the fairness literature have considered fairness in static decision making problems, such as classification, regression, etc [10, 13, 17]. Recent results suggest that fairness itself is not static, but rather dynamic: a system that is fair in its static decision-making tasks may become biased in its overall societal impacts over time [9, 20, 25, 26, 30, 39, 40]. This happens when the system makes sequential decisions about humans, and every decision of the system is met with some human reaction in return, possibly changing the parameters and the future decisions of the system. Such feedback patterns often reinforce historical biases in the dataset and introduce new biases in the society in the long-run as well. While there are many works that have proposed analysis and mitigation techniques for long-run biases, to our best knowledge, there does not exist any technique that could detect such biases in real-time. We propose *runtime monitoring*, as a new addition to the fairness toolbox, for the real-time detection of dynamic social biases in deployed machine-learned decision makers, whose models are unknown and may change over time (e.g., due to retraining, changes in parameters, etc.).

The goal of runtime monitoring is to design a *monitor* which will observe the sequential interactions between the decision-maker and its environment, and, after each observation, will output a quantitative, statistically rigorous estimate of how fair or biased the system is at that point in time. Unlike most existing approaches [26, 30, 39], our monitors do not require any assumption or explicit knowledge of the system model.

Monitoring can help us in two ways. Firstly, by detecting biases in real-time, it can trigger corrective measures or retraining, whenever necessary. Statically designed fairness interventions are based on an assumed dynamic model of the system. In practice, models are rarely perfect due to imperfect knowledge of the systems and the involved uncertainties, making it often impossible to predict if a long-run fairness intervention is going to work in practice. Moreover, the underlying environment conditions may change over time, making static interventions even harder. Monitoring offers an additional, complementary tool that enables us to close this gap by warning us of the presence of biases in real-time, so that we can adapt our intervention techniques whenever necessary. There is an analogy to control theory, where it is well-known that closed-loop (feedback) controllers fare much better against modeling uncertainties than open-loop (feed-forward) controllers [31, Sec. 1.3].

The other area where monitoring can help us is in the creation of trusted third-party watchdogs for overseeing the fairness of decision-makers. They can work neutrally in public interest, since they are by-design independent of the implementation of the system.

Consider the following situation where fairness is dynamic, and we show that monitoring will be useful. Consider a bank that gives loans to individuals based on their credit scores. The population is divided into two groups, with one group having higher average credit score than the other. A policy of the bank that gives loans to the eligible individuals from each group with equal probabilities (equalized opportunity [17]) may seem fair and noble. However, in doing so, the bank may end up giving more loans to less eligible individuals from the disadvantaged group. If the credit score distribution of the disadvantaged group is heavily skewed towards a higher default rate, then there will be many loan defaults, causing a further drop in average credit score of the disadvantaged group [40]. For this example, we present a monitor which observes a single long sequence of lending events, consisting of sampling of an applicant, the decision made by the bank on this applicant, and if the loan was granted then whether it was repaid or not. After each observation, the monitor computes a quantitative statistical estimate of the difference between the average credit scores of the two groups. It does so by being completely oblivious to the bank's policy and by not assuming any prior knowledge about the humans' behaviors (whether they repay or not).

Now consider the following situation. It has been shown that voice assistants, such as Amazon Alexa and Google Home, are biased towards the English accents of native speakers, where the native speakers experience significantly higher quality service than the non-native speakers [18]. This happens when there is an imbalance between group representations in the dataset, with more data available for one demographic group than the other. If over time, more and more non-native speakers stop using the service out of dissatisfaction, then the dataset gets more skewed towards the native speakers, intensifying the biases further [19]. Similar representation-driven biases were reported in other areas as well, such as recommendation systems [7], credit market [15], and crime prediction [12]. While, in theory, there are remedies that work if the reactions of the humans can be perfectly predicted, in practice, they may worsen the situation whenever the modeling assumptions do not align well with the true intentions of the humans [39]. This demonstrates that it is difficult to design a static fairness intervention that will always work in the long run. Monitoring can help us to, firstly, detect such dynamic biases and warn us in time, and, secondly, to change the interventions whenever necessary.

We consider *time-varying social fairness properties*, as a class of dynamic fairness properties. They can be written as the difference in expected values of a given function over unknown time-varying feature distributions across two demographic groups. Such properties can capture many existing aspects of long-run fairness properties in the society, such as the time-varying difference in expected credit scores across two groups [40], the time-varying difference in group representations [19], etc.

Our monitors perform statistical estimations to obtain a PAC-style estimate of the value of the social fairness properties in real-time. We do not make any assumptions about the policies of the already deployed machine-learned agent and the human users (i.e., the environment). The only assumption we make is that the monitor can observe the features of the selected individual, the actions of the agent, and the reactions of the individual. Moreover, we assume the availability of a *change function*, such that from each observation the monitor can infer the resulting change in the expected value of the unknown distribution. For instance, in the lending example, we assume the observability of the credit scores and the group memberships of the sampled individuals, the bank's decisions, and the reactions of repaying or defaulting of loans by the individuals. At any time, if the individual is selected from a group with size N, then the change function tells us that a repayment of the loan will increase the credit score of the individual by, say, 1 point, thereby increasing the average credit score of their group by $1/N$. Similarly, a loan default will decrease the credit score of the individual by, say, 1 point, thereby decreasing the average credit score of their group by $1/N$. Our monitor observes one long sequence of lending events, and, after each new observation and based on the given history of past lending events and the past valuations of the change function, computes an updated PAC-style estimate of the disparity in average credit scores across the two groups.

Computationally, our monitors are extremely lightweight, and their implementations required only a few lines of code. Yet, the mathematical analysis of their correctness is nontrivial. The difficulty stems from the fact that the samples observed on any given sequence are all statistically dependent on each other. For instance, the probability of sampling an individual with a certain credit score will depend on whether the previous individual who was from the same group and had the same credit score repaid the loan or not. As our monitor, we present an unbiased statistical estimator as well as PAC-style bounds for its estimates. The bounds are obtained by constructing a martingale from the estimates, analyzing the corresponding martingale difference sequence, and applying suitable concentration inequalities for martingale difference sequences.

We implemented our monitors in a prototype tool. Using this implementation, we designed monitors for two practical examples from the literature. The first example concerns the *lending problem* that we discussed earlier, where we monitored, in real-time, to what degree the lending policy of the bank has widened the disparity of average credit scores across the two demographic groups. The second example is an *attention allocation problem* [9], where incidents keep occurring at every step in multiple locations, and we have a machine-learned allocator for allocating its limited units of attention to the locations to discover the incidents. The rate at which incidents occur at each location is inversely proportional to the amount of attention allocated to that location in the previous step. Real-world applications of this example include child services, pest control, etc. We monitored, in real-time, to what degree the allocator's allocation policy has widened the disparity of discovery probability of incidents among two of the given locations. Implementations of these systems were already available in the tool `ml-fairness-gym` [9]. We executed our monitors on the simulation traces of the systems as extracted from `ml-fairness-gym`. We demonstrate that our monitors are able to produce tight statistical estimates of the considered fairness properties in real-time.

We believe that runtime monitors will be an important new addition to the fairness toolbox. On one hand, they will complement the existing model-based analysis and design tools by checking dynamic fairness in real-time, and helping us to trigger on-demand corrective measures. On the other hand, they will be useful in building trusted third-party fairness watchdogs.

All the technical proofs have been omitted for the lack of space; they can be found in the longer version of our paper [22].

Related Work

Fairness in automated decision making has become an active field of research in recent years. Early works only considered fairness in the static decision making settings, where the decision maker needs to be fair with respect to a time-invariant distribution. Several group fairness [13, 17] and individual fairness [10] criteria were proposed, and measures for implementing them were invented. The proposed measures in this setting can be grouped into three categories: (a) ones which *pre-process* the training dataset to eliminate historical biases [6, 16, 24, 38], (b) ones which design training algorithms that are more robust to biases (called *in-processing*) [1, 4, 36, 37], and (c) ones which *post-process* the decision-maker's output to eliminate biases [17].

Later, it was observed by many authors that, surprisingly, decision policies that are statically fair may lead to unfair behavior in the sequential setting. In this regard, the simplest sequential setting studied in the literature is the two-stage one: in the first stage, the agent makes decisions on humans from two groups, which may cause the humans to take certain actions, and the resulting impact on the groups are then examined in the second stage [20, 25]. In the more general long-term setting, the agent is allowed to retrain its decision policy over time, which may be affected by a change or bias in the dataset, caused by the reactions of humans to decisions made by the agent in the past. This closed feedback was shown to self-reinforce biases that were present in the dataset as well as introduce new biases. Relevant works on the sequential setting can be found in a recent survey [40]. While most of the existing works attempt to eliminate biases at design-time and assume information about the model [26, 30, 39], we detect them at runtime with little knowledge about the model. There are also simulation-based studies which study long-term impacts of static fairness measures [9]. They are also incomparable to our monitoring setup: in simulations, it is shown how bias changes over time for an assumed model of interactions, whereas we make almost no assumptions on the model and use the concrete measurements to estimate the bias in the system.

Our monitors are designed to operate in a dynamic setting. Hence, static systems or systems where the decisions of the agent do not affect the parameters of the underlying population, which have been studied extensively in the literature (see Mehrabi et.al. [29]), are a special case of our setting. Therefore, monitoring could be applied. A natural setting would be the deployment of monitors to check whether an agent in a bandit setting is fair [8, 23].

Runtime monitoring is a well-studied subject in the area of formal methods in computer science [3]. The goal is to check, at runtime, if an unknown system satisfies or violates a given safety property. For instance, a monitor may be used to detect traffic congestion in the roads of a smart city [27], or safety violation of autonomous vehicles [28]. The outputs of monitors are usually passed to a safety-supervisory control layer, which takes necessary actions to prevent damages, for example through a default fail-safe action [5].

Unfortunately, a majority of the existing works in runtime monitoring cannot handle statistical properties, such as fairness. Notable exceptions include the work by Ferrère et al. [14], which develops efficient techniques for monitoring statistical properties of systems. However, they do not consider fairness properties. Moreover, their monitors' outputs are correct only asymptotically, whereas our monitors output PAC-style error bounds for every observed sequence of finite length.

The closest to our work are the papers by Albarghouthi et al. [2] and a recent paper by us [21]. Albarghouthi et al. [2] presented an approach for monitoring fairness in sequential decision-making tasks, which we generalized to monitoring fairness over Markov chains using techniques from both frequentist and Bayesian statistics [21]. These works can be used to monitor only group fairness and individual fairness properties in static decision-making problems, whereas we monitor time-varying social fairness properties in dynamic decision-making problems.

There is a body of research that is ideologically similar to ours, and developed sequential statistical tests to evaluate the performance of already deployed machine-learned systems at runtime. Podkopaev et al. [32] proposed an algorithm for monitoring the expected loss of a given classifier due to shift in the dataset distribution. The expected loss is based on the misclassification rate of a classifier, and is uncomparable to fairness properties that we consider in this work. Waudby-Smith et al. [34] proposed a sequential estimation algorithm that was used to estimate the time-varying average treatment effect (ATE) in a randomized experiment, which gives a measure of the expected difference in outcome between an individual chosen from the population receiving a treatment (like a medical drug that is being tested) and not receiving the treatment. Although there are some structural similarities between ATEs and fairness properties, they estimate how much the ATE was *on an average until* the present time, whereas we estimate how much fair the system is *at* the present time. Moreover, their estimates are asymptotically correct, whereas we provide finite-sample correctness guarantees.

2 PROBLEM SETUP

2.1 The Sequential Agent-Environment-Interaction Model

We call a machine-learned decision maker an *agent*, and the population of the subjects of its decisions the *environment*. For example, in a lending scenario, a bank's machine-learned lending policy is the agent, and the population of the loan applicants is the environment.

We use a setup similar to the work of D'Amour et al. [9], where an agent engages in a sequential interaction with its environment, and as a result the parameters of the environment change. The environment contains a distribution over the individuals, where each individual is represented by a real-valued (scalar) feature of interest, such as their credit score, and a sensitive attribute, such as their ethnicity. In this work we only consider fairness properties that depend on the single available feature of the individuals; extension to fairness with respect to feature vectors is left open for future work. In general, we allow the individuals to have additional features, though they do not influence the fairness. For simplicity of notation, we suppress such additional unimportant features when considering the individuals.

At each step t, the environment samples a *single* individual with feature X_t and the group membership G_t, where X_t is a real-valued random variable and G_t is a random variable which is assumed to have a binary support $\{A, B\}$ for simplicity. We use the shorthand notations $\mathbb{P}_A(X_t = x_t)$ and $\mathbb{P}_B(X_t = x_t)$ to denote, respectively, the conditional probabilities $\mathbb{P}(X_t = x_t \mid G_t = A)$ and $\mathbb{P}(X_t = x_t \mid G_t = B)$. Moreover, for any two random variables W and V with their respective outcomes w and v, we use the shorthand notation $\mathbb{P}(w \mid v)$ and $\mathbb{E}(W \mid v)$ instead of $\mathbb{P}(W = w \mid V = v)$ and $\mathbb{E}(W \mid V = v)$, respectively.

At time t, the agent performs an *action* Y_t, which is also treated as a random variable. Given the agent's action, the environment may react by using its own *reactions*, which we denote using the random variable Z_t. The randomness in Y_t and Z_t capture the modeling uncertainties, such as unknown factors that influence the agent's actions and unpredictability in the environment's reactions. In the lending example, the agent's (i.e., the bank's) actions are granting or rejecting the loan to the selected individual, whereas the environment's reactions are repaying or defaulting of the loan by the same individual. Some problems, such as the attention allocation example, do not require environment's reactions. (Although, in practice, Z_t may lag from Y_t, for simplicity, we assume that they happen at the same time step.) This completes one round of interaction between the agent and the environment, and a sequence contains many such interaction rounds.

The interactions between the agent and the environment form a sequence of (tuples of) random variables, i.e., a stochastic process $\vec{O} = ((G_t, X_t, Y_t, Z_t))_{t>0}$. For every t, the tuple of concrete values that the random variables take is called an *observation*, denoted as $o_t := (g_t, x_t, y_t, z_t)$. The sequence $\vec{o}_t = (o_s)_{s\in[1..t]} = ((g_s, x_s, y_s, z_s))_{s\in[1..t]}$ is called an *observation sequence.*

In the process $\vec{O}$, the feature distribution X_t is subject to changes over time; Y_t and Z_t may also change, but that is irrelevant for us. We assume that the monitor can infer, from the observations, the resulting change in the current expected value of X_t (given the history of observations). For instance, in the lending scenario, if at any time the selected individual fails to repay the loan, then the credit score of that individual goes down, and so the distribution of credit scores in the population shifts. We assume that we can infer the shift in expected credit scores from the lending decision of the bank and the event of repayment/default. We formalize this in the following.

Assumption 1. *Runtime monitors have access to a function Δ, called the* change function, *which maps every concrete observation o_t to a change in the expected value of X_t, such that for each group $g \in \{A, B\}$, for every time t, and for every past sequence of observations $\vec{o}_t$, we have: (i) $\mathbb{E}_g(X_{t+1} | \vec{o}_t) = \mathbb{E}_g(X_t | \vec{o}_{t-1}) + \Delta(o_t)$; (ii) $\mathbb{E}_g(|X_t| \mid \vec{o}_{t-1}) < \infty$; (iii) X_t, when centered, is a sub-exponential random variable with parameters (σ^2, ν).*

Assump. 1 imposes mild technical restrictions that are fulfilled by many real-world problems, including the lending example and the attention allocation example that we consider here. Whenever clear from the context, for simplicity, we write Δ_t instead of $\Delta(g_t, x_t, y_t, z_t)$.

2.2 Time-Varying Social Fairness Properties

Let $f : o_t = (g_t, x_t, y_t, z_t) \mapsto \mathbb{R}$ be a function, called the *well-being function*, which is a measure of the well-being of the individual (g_t, x_t) who was subjected to the agent's action y_t to which they reacted with z_t. In the lending example, f maps an observation to the credit score of the selected individual. In the attention allocation example, f maps an observation to the ratio of the attention (action) to the number of incidents (reaction).

For each group g, and for every observation sequence $\vec{o}_t$, we define the (group-specific) *expected well-being* as:

$$\omega^g(\vec{o}_t) := \mathbb{E}_g(f(O_t) \mid \vec{o}_{t-1}, y_t, z_t). \tag{1}$$

Observe that the expectation is with respect to the randomness in the feature distribution X_t, which makes O_t also random. We do not condition on the currently observed feature x_t, as it would make the expectation trivially equal to $f(o_t)$. In other words, the expectation in the well-being is only with respect to the past observations of credit scores in the lending example, and is only with respect to the past observations of incidents in the attention allocation example.

We consider a class of fairness properties, which we call the *time-varying social fairness properties*, defined as the difference in expected well-beings of the two groups for a given observation sequence $\vec{o}_t$:

$$\varphi(\vec{o}_t) := \omega^A(\vec{o}_t) - \omega^B(\vec{o}_t). \tag{2}$$

Time-varying social fairness properties capture many interesting properties that were already studied in the context of sequential decision-making, such as the time-varying disparity in average credit score [25], time-varying disparity in the discovery probability of incidents [11, 12], etc.

In (2), we present the general class of time-varying fairness properties that we consider, and the exact property will depend on the application and the definition of the function f. For instance, in the lending example, $f(o_t)$ will be independent of g_t, y_t, z_t and will give us the credit score of the individual sampled at time t.

We point out that we do not impose any assumption on the agent's and the environment's policies for choosing their respective actions and reactions. However, following Assumption 1, (re-)actions at each time influence the expected observation at the next step. Hence, it is impossible to statically predict the conditional expectation in advance. Intuitively, this means that without observing the loan decisions of the bank and the subsequent repayment or default events, we cannot predict what the expected credit score will be at a particular point in the future.

As a result, we cannot statically predict the social fairness in the system in the long-run, even if we knew its initial value. Thereby, it is only possible to measure social fairness retrospectively, which is what we do using runtime monitoring. To our best knowledge, no prior work in the fairness literature considered this problem.

2.3 The Monitoring Problem

A *monitor* is a function that maps every observation sequence to a real interval, where the output interval computed by the monitor is a PAC-style statistical estimate of the given social fairness property. We summarize the monitoring problem in the following.

Problem 1. *Let $\vec{O}$ be a stochastic process, φ be a social fairness property, and $\delta \in [0,1]$ be a parameter. Design a monitor M such that for every time t, the following holds:*

$$\mathbb{P}\left(\varphi(\vec{O}_t) \in M(\vec{O}_t)\right) \geq 1 - \delta.$$

The probabilistic uncertainty in the monitor's output is due to the non-availability of the parameters of the initial feature distribution: were the initial parameters known to the monitor, at every time, a precise value of the fairness property could be calculated from the net change in the parameters as deduced from the change function. On the other hand, a naïve PAC estimate of $\varphi(\vec{O}_t)$ at each time step is also not feasible, since the feature distribution is constantly changing.

For the fixed observation sequence $\vec{o}_t$, the estimate $[l,u] = M(\vec{o}_t)$ is called the $(1-\delta)\cdot 100\%$ *confidence interval* for $\varphi(o_t)$. The radius, given by $\varepsilon = 0.5 \cdot (u-l)$, is called the *estimation error*, and the quantity $1-\delta$ is called the *confidence*. The estimate gets more precise as the error gets smaller and the confidence gets higher. For the lending example, Prob. 1 asks us to design a monitor which will observe a sequence of lending events, and, after each observation, will output a $(1-\delta)\cdot 100\%$ confidence interval for the estimated disparity in average credit scores.

While our monitors output *interval estimates* of fairness properties in the form of confidence intervals, internally, they first compute *point estimates* of the *expected feature* of each group g for a given observation sequence $\vec{o}_{t-1}$, defined as:

$$\psi^g(\vec{o}_{t-1}) \coloneqq \mathbb{E}_g(X_t \mid \vec{o}_{t-1}). \tag{3}$$

Note that the quantity, $\psi^g(\vec{o}_{t-1})$ gives us only the expected feature, which will be an intermediate step for estimating the well-being. Notice that the expected feature at time t only depends on the past observations until time $t-1$, whereas the well-being at time t requires the action and the reactions, such as units of attentions allocated by the attention allocator at the current time t. A *point estimator* $\hat{E}$ of X_t for a given $\vec{o}_t$ and a given group g is a function $\hat{E}\colon \vec{o}_t \mapsto \mathbb{R}$. Additionally, $\hat{E}$ will be called *unbiased*, if for every observation sequence $\vec{o}_{t-1}$ that may occur with positive probability, we have:

$$\mathbb{E}_g\left(\hat{E}(\vec{O}_t) \,\middle|\, \vec{o}_{t-1}\right) = \psi^g(\vec{o}_{t-1}).$$

Intuitively, unbiasedness guarantees that, for any given history $\vec{o}_{t-1}$ of loan events that may occur with positive probability, the expected credit score of a group at time t will be equal to the expected output of the estimator at time t.

While unbiasedness guarantees that the estimator $\hat{E}$'s output coincides with $\psi^g(\vec{o}_{t-1})$ *in expectation*, we also require that the output error remains statistically bounded at all time. To this end, we bound the estimation error by computing confidence intervals for $\psi^g(\vec{o}_{t-1})$, obtained through application of concentration inequalities around the point estimate. These confidence intervals of group-specific expected features are then used to obtain confidence intervals for group-specific expected well-beings (i.e., $\omega^g(\vec{o}_t)$), which are then subtracted from each other to finally obtain the output confidence interval of the monitor for the time-varying social fairness property $\varphi(\vec{o}_t)$. We illustrate this sequence of steps in Fig. 1, along with references to the sections where they are described.

While we provide a general procedure for estimating $\psi^g(\vec{o}_{t-1})$ in the first step, a general overall estimation procedure for any arbitrary $\varphi(\vec{o}_t)$ is difficult to derive and is left open. This is because the final confidence interval for $\varphi(\vec{o}_t)$ depends on the structure of the well-being function, which is problem-specific.

As a convention, by "monitor," we will exclusively refer to the final interval estimator of $\varphi(\vec{o}_t)$, though it is not the only interval estimator that we will use.

3 AN INTERVAL ESTIMATOR FOR THE TIME-VARYING EXPECTED FEATURE

For any group g and an arbitrary observation $\vec{o}_t$, in this section, we construct an interval estimator for the group-specific expected feature $\psi^g(\vec{o}_{t-1}) = \mathbb{E}_g(X_t \mid \vec{o}_{t-1})$ (defined in (3)). As $\psi^g(\vec{o}_{t-1})$ does not depend on the fairness metric, hence the estimator is not tied to the fairness monitoring problem and can have other use. The estimator for $\psi^g(\vec{o}_{t-1})$ will be the essential component of the monitors (i.e., the interval estimator for the social fairness property $\varphi(\vec{o}_t)$), which will be presented later in the respective example sections. The interval estimate of $\psi^g(\vec{o}_{t-1})$ is obtained by first computing a point estimate of it, and then using concentration inequalities to bound the estimation error. The first part, i.e., the point estimation part, is explained using a coin-toss analogy.

3.1 Warm-Up: A Coin-Toss Puzzle

Suppose we have a coin with unknown initial bias. After each toss, its bias changes in a predefined manner as a function of the outcome of the previous toss. How to compute a point estimate of the bias of the coin at any given point in time, based on the given sequence of the observed past outcomes?

Let, us formalize the problem first. Let, at time t, the probability that the coin shows 1 (heads) be p_t, and the probability that the coin shows 0 (tails) be $1-p_t$. The toss outcome at time t is denoted using the random variable X_t. Let p_1 be the initial bias which is fixed but unknown. After every toss, the coin's bias shifts by a constant $\epsilon \in [0,1]$, and the direction of the shift depends on the outcome of the previous toss: if we see $X_t = 1$ at time t, the bias shifts to $p_{t+1} = p_t + \epsilon$ in the next step, whereas if we see $X_t = 0$ at time t, the bias shifts to $p_{t+1} = p_t - \epsilon$. Let's assume for simplicity that the true initial bias is not too close to the boundaries 0 and 1, and moreover ϵ is small enough and the observed sequence is short enough that the true bias never reaches the boundaries. We can succinctly write $p_{t+1} = p_t + \Delta_t$, where $\Delta_t = x_t\epsilon + (1-x_t)(-\epsilon)$ is the change of bias recorded at time t. Then the estimation problem

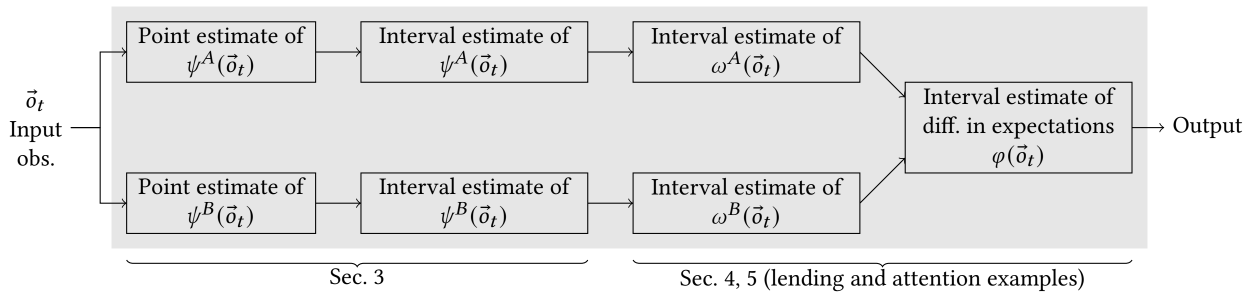
$\vec{o}_t$
Input
obs.
Point estimate of $\psi^A(\vec{o}_t)$
Interval estimate of $\psi^A(\vec{o}_t)$
Interval estimate of $\omega^A(\vec{o}_t)$
Point estimate of $\psi^B(\vec{o}_t)$
Interval estimate of $\psi^B(\vec{o}_t)$
Interval estimate of $\omega^B(\vec{o}_t)$
Interval estimate of diff. in expectations $\varphi(\vec{o}_t)$
Output
Sec. 3
Sec. 4, 5 (lending and attention examples)

Figure 1: The operational diagram of the monitor and the sections of this paper where the components are presented.

asks: for any given sequence $x_1, \ldots, x_t$ of toss outcomes, how to compute a point estimate of the unknown parameter p_t, i.e., the value of $\mathbb{E}(X_t \mid x_1, \ldots, x_{t-1})$?

Observe that even if we knew p_1, without seeing the observations $x_1, \ldots, x_t$, the value of p_t would only be probabilistically known. Thus a static analysis would not be possible even in that case.

For the trivial case of $\epsilon = 0$, i.e., when we know that the bias remains fixed at p_1 throughout, we can compute an unbiased point estimate of $p_t = p_1$ by simply computing the empirical average of the observed sequence as $\hat{p}_t = \frac{1}{t}\sum_{s=1}^{t} x_s$.

When $\epsilon > 0$, we show (proof in the longer version [22]) that, for the given observation sequence $x_1, \ldots, x_t$, the following is an unbiased point estimator of p_1:

$$\hat{p}_1 = \frac{1}{t}\sum_{s=1}^{t}\left(x_s - \sum_{r=1}^{s-1}\Delta_r\right). \quad (4)$$

Once an estimate of p_1 is known, we can obtain an unbiased point estimate for $\mathbb{E}(X_t \mid x_1, \ldots, x_{t-1})$ by accounting for the observed changes up to time t:

$$\hat{p}_t = \frac{1}{t}\sum_{s=1}^{t}\left(x_s - \sum_{r=1}^{s-1}\Delta_r\right) + \sum_{s=1}^{t-1}\Delta_s. \quad (5)$$

While there are techniques to estimate the non-time-varying mean of a statistical process from sequential observations [35], to our best knowledge, the problem we consider and the solution we propose are completely novel.

3.2 The Interval Estimator for Expected Features

Now we extend the coin-toss analogy to the point estimation, followed by the interval estimation, of the expected features $\psi^g(\cdot)$. For simpler notation, in the rest of this section, we assume that there is only one group, so that all the past observations correspond to that group only. We drop the superscript g from the property $\psi^g(\cdot)$.

Drawing comparison with the coin-toss setting, the bias p_t of the coin at every t is now replaced by $\mathbb{E}(X_t \mid \vec{o}_{t-1})$, and the bias shift Δ_t of the coin at time t is now replaced by the value of the change function $\Delta(o_t)$. We make these adjustments in the point estimator of bias of the coin in (5), and obtain the following point estimator for $\psi^g(\vec{o}_{t-1})$:

$$\hat{E}(\vec{o}_t) \coloneqq \frac{1}{t}\sum_{s=1}^{t}\left(x_s - \sum_{r=1}^{s-1}\Delta(o_r)\right) + \sum_{s=1}^{t-1}\Delta(o_s).$$

From this point estimator, we can obtain an interval estimator $\psi(\vec{o}_{t-1})$ by applying a suitable version of Azuma-style inequality (see Theorem 3.3) to compute a $(1-\delta)\cdot 100\%$ confidence interval around $\hat{E}(\vec{o}_t)$ for any given δ. We call the interval estimator of $\psi(\cdot)$ `ExpEstimator`, and present its pseudocode in Alg. 1. In the function `Init`, the monitor first initializes various internal registers. After each new observation (g, x, y, z), the monitor invokes the function `Compute` to compute a new $(1-\delta)\cdot 100\%$ confidence interval for the expected features $\psi^g(\vec{o}_{t-1})$.

3.3 Soundness of the Interval Estimator

For soundness, we need to show that (a) the point estimator of $\psi(\vec{o}_{t-1})$ is unbiased, and that (b) the interval estimate computed using the Azuma-style inequality is statistically sound. Claim (a) follows from the definition of unbiasedness. For claim (b), we show that the sequence of the expected point estimates, conditioned on the increasingly longer sequence of prefixes of a given observation, is a Doob martingale. Furthermore, we show that the difference between any two consecutive elements of the Doob martingale is a sub-exponential random variable, which enables us to use an Azuma-style concentration inequality to compute the desired confidence interval from the point estimate. Below, we present the highlights of the soundness proof (details in the longer version [22]), which can be skipped over without any loss of continuity.

Unbiasedness: To demonstrate that the estimator $\hat{E}(\vec{O}_t)$ is unbiased, we first show:

Lemma 3.1. *The estimator $\hat{E}_1(\vec{o}_t) \coloneqq \frac{1}{t}\sum_{s=1}^{t}\left(x_s - \sum_{r=1}^{s-1}\Delta(o_r)\right)$ is an unbiased estimator of $\mathbb{E}(X_1)$.*

Then we utilize the change function Δ and the definition of conditional expectation to transfer the result to $\hat{E}(\vec{O}_t)$.

$$\mathbb{E}\left(X_t \,\middle|\, \vec{O}_{t-1}\right) = \mathbb{E}(X_1) + \sum_{s=1}^{t-1}\Delta(O_s) \quad (6)$$

$$= \mathbb{E}\left(\hat{E}_1(\vec{O}_t)\right) + \sum_{s=1}^{t-1}\Delta(O_s) = \mathbb{E}\left(\hat{E}(\vec{O}_t) \,\middle|\, \vec{O}_{t-1}\right) \quad (7)$$

Algorithm 1 `ExpEstimator`

```
1: function Init($\Delta, \delta, \sigma, \nu$)
2:     $\Delta \leftarrow \Delta$                              ▷change function
3:     $\delta \leftarrow \delta$                              ▷target confidence level
4:     $(\sigma, \nu) \leftarrow (\sigma, \nu)$                ▷sub-exp. parameters
5:     $t \leftarrow 0$                                        ▷clock counter
6:     $\hat{E}_1 \leftarrow 0$        ▷curr. estimate of the initial expected value
7:     $d \leftarrow 0$                                        ▷net distribution shift

1: function Compute($x, y$)
2:     $t \leftarrow t + 1$
3:     $\hat{E}_1 \leftarrow \frac{1}{t}\left(\hat{E} \cdot (t-1) + (x - d)\right)$      ▷update
4:     $\hat{E} \leftarrow \hat{E}_1 + d$                      ▷curr. estimate
5:     $d \leftarrow d + \Delta(x, y)$                         ▷update net shift
6:     $\varepsilon \leftarrow Azuma(t, \delta, \sigma, \nu)$  ▷see Theorem 3.3
7:     return $[\hat{E} - \varepsilon, \hat{E} + \varepsilon]$
```

Proving concentration around the mean using martingales: To demonstrate that the estimator $\hat{E}(\vec{O}_t)$ concentrates around its mean, we construct the *Doob martingale*: $(\mathbb{E}(\hat{E}_1(\vec{O}_t) \mid \vec{O}_s))_{s\in[0..t]}$. Intuitively, this martingale is a step-by-step approximation process. That is, it starts from the quantity we want to estimate, i.e. $\mathbb{E}(\hat{E}_1(\vec{O}_t))$ which due to Lemma 3.1 is equal to $\mathbb{E}(X_1)$, and ends at $\mathbb{E}(\hat{E}_1(\vec{O}_t) \mid \vec{O}_t)$ which given the definition of the conditional expectation is the complete random variable, i.e. our estimator $\hat{E}_1(\vec{O}_t)$.

Applying an Azuma-style concentration inequality: To bound the distance between the first and last step of the martingale we want to apply some form of Azuma-style concentration inequality. However, this requires knowledge about the behavior of the difference between two consecutive martingale steps.

Lemma 3.2. *The martingale difference for any $s \in [1..t]$*

$$\mathbb{E}\left(\hat{E}_1(\vec{O}_t) \,\middle|\, \vec{O}_{s+1}\right) - \mathbb{E}\left(\hat{E}_1(\vec{O}_t) \,\middle|\, \vec{O}_s\right) = \frac{1}{t}\left(X_{s+1} - \mathbb{E}\left(X_{s+1} \,\middle|\, \vec{O}_s\right)\right)$$

and therefore it is a sub-exponential random variable with parameter $(\frac{\sigma^2}{t^2}, \frac{\nu}{t})$.

Hence, we can bound the probability that $\mathbb{E}(X_1)$ is within an $\varepsilon > 0$ interval around the estimator $\hat{E}_1(\vec{O}_t)$, i.e. $\mathbb{P}(|\mathbb{E}(\hat{E}_1(\vec{O}_t)) - \mathbb{E}(X_1)| \geq \varepsilon)$. This directly translates into a bound of the difference between $\mathbb{E}(X_t \mid \vec{O}_{t-1})$ and $\hat{E}(\vec{O}_t)$,

$$\mathbb{P}\left(\left|\mathbb{E}\left(\hat{E}_1(\vec{O}_t)\right) - \mathbb{E}(X_1) + \sum_{s=1}^{t-1} \Delta(O_s) - \Delta(O_s)\right| \geq \varepsilon\right)$$
$$= \mathbb{P}\left(\left|\mathbb{E}\left(\hat{E}(\vec{O}_t)\right) - \mathbb{E}\left(X_t \,\middle|\, \vec{O}_{t-1}\right)\right| \geq \varepsilon\right)$$

Thus we finally obtain the soundness theorem of our expected value estimator.

Theorem 3.3. *Let $\delta \in [0,1]$. Let $\vec{O}$ be a stochastic process as specified in Assumption 1. For every time step t,*

$$\left|\psi(\vec{O}_{t-1}) - \hat{E}(\vec{O}_t)\right| \leq \max\left\{\sqrt{\frac{2\sigma^2}{t} \cdot \log\left(\frac{2}{\delta}\right)}, \frac{2\nu}{t}\log\left(\frac{2}{\delta}\right)\right\}$$

holds with probability $1-\delta$. We define $Azuma(t, \delta, \sigma^2, \nu)$ to refer to this bound.

4 A DYNAMIC LENDING PROBLEM

Now we present a monitoring algorithm for the lending example that we took from the literature [25]. Suppose we have a bank that lends money to individuals by taking into account their credit score and group membership information. Every individual has a credit score that may change over time, and let $[0..c_{\max}]$ be the set of all possible credit scores of every individual. Also, the population is divided into groups A and B, as usual, and let the number of individuals in the groups be N_A and N_B, respectively. At every step, the bank gets the credit score and the group information of a randomly chosen individual, and decides whether to accept or reject the loan application. If the loan is granted and is subsequently repaid, then the respective individual's credit score increases by 1, provided her initial credit score was smaller than $c_{\max}$. On the other hand, if the loan is granted but defaulted, then the credit score decreases by 1, provided her initial credit score was greater than 0. If the loan is rejected, then the individual's credit score remains unchanged. We want to monitor, after each observation, if the bank's policy leads to unfair distribution of the expected credit score among the individuals of the two groups A and B.

4.1 Problem Formulation

We assume a uniform distribution over the set of individuals at every time step. Given a (random) individual with features X_t and group G_t sampled uniformly at time t, the bank's (random) action Y_t of rejecting or accepting the individual is given by $Y_t = 0$ or $Y_t = 1$, respectively. Once the bank has chosen an action, the individual reacts as follows: if $Y_t = 0$ then the individual's action is immaterial, and if $Y_t = 1$ then the individual performs (random) action Z_t, i.e., they can pick either $Z_t = 1$ or $Z_t = 0$ denoting, respectively, whether they repay the loan or not. The resulting change in the distribution is specified using the change function Δ defined as below:

$$\Delta(x_t, g_t, y_t, z_t) = \begin{cases} +\frac{1}{N_{g_t}} & \text{if } c < c_{\max},\ y_t = 1,\ z_t = 1, \\ -\frac{1}{N_{g_t}} & \text{if } c > 0,\ y_t = 1,\ z_t = 0, \\ 0 & \text{otherwise.} \end{cases}$$

That is, if the loan is granted and the individual repays the loan, the expected credit score increases by $\frac{1}{N_{g_t}}$; if the loan is granted and the individual fails to repay the loan, the expected credit score decreases by $\frac{1}{N_{g_t}}$; otherwise the expected credit score remains the same.

The well-being function f maps to the credit score (i.e., the feature itself), and the time-varying social fairness criteria is given by the disparity in expected credit scores of the two groups, i.e.,

$$\varphi_t = \mathbb{E}_A(X_t \mid \vec{o}_{t-1}) - \mathbb{E}_B(X_t \mid \vec{o}_{t-1}).$$

4.2 The Runtime Monitor

The monitor for the lending problem, called LendingMonitor, is shown in Alg. 2. Internally, LendingMonitor runs two

ExpEstimator (see Alg. 1) sub-monitors M_A and M_B in parallel, which compute the interval estimates for the expected credit scores of the two groups respectively. After each new observation, depending on the group information of the selected individual, the respective monitor's Compute function is invoked, and a new estimate for that group is computed. This is possible because X_t is a bounded random variable, making it sub-gaussian with parameter $c_{\max}^2$ (which implies it is sub-exponential with $\nu = 0$). The output of LendingMonitor is the difference between the two interval estimates computed by M_A and M_B. Observe that each of M_A and M_B uses higher confidence level $1 - \frac{\delta}{2}$, so that the final confidence of the output estimate becomes $1 - \delta$, after applying the union bound. For simplicity, we chose union bound to compute the overall confidence, however, tighter interval estimates can be computed by observing that the group-specific stochastic processes are statistically independent, thereby allowing us to use the sharper bounds from the Hoeffding's inequality (see [33, p. 24]).

Algorithm 2 LendingMonitor

```
1: function Init($\Delta, \delta, c_{\max}$)
2:     $M_A \leftarrow$ ExpEstimator.Init($\Delta, \frac{\delta}{2}, c_{\max}, 0$)
3:     $M_B \leftarrow$ ExpEstimator.Init($\Delta, \frac{\delta}{2}, c_{\max}, 0$)
4:     $\overline{E}_A \leftarrow 0$
5:     $\overline{E}_B \leftarrow 0$
   1: function Compute($x, g, y, z$)
   2:     $\overline{E}_g \leftarrow M_g$.Compute($x, g, y, z$)
   3:     $\overline{\varphi} \leftarrow \overline{E}_A - \overline{E}_B$          ▷interval difference
   4:     return $\overline{\varphi}$
```

4.3 Experimental Outcome

We summarize the outputs of our monitor in Fig. 2. We observe that our monitor's outputs match with the correct value of bias in the system at every point in time. Moreover, the monitor Lending-Monitor took on an average 3 µs to compute a new estimate after receiving every new observation.

Interestingly, we note that the MaxRwrd agent's policy becomes more fair in the long run compared to the EqOpp agent's policy. This phenomenon has already been demonstrated in the existing literature [9]. With our monitors, it is actually possible to detect such biases in real-time, without any information or assumption about what policy the bank is using or how the individuals will react (i.e., whether they repay or not).

5 A DYNAMIC ATTENTION ALLOCATION PROBLEM

Now we consider the attention allocation problem. Suppose there are $L \geq 2$ locations, and at each location and at each time step, a number of incidents take place. There is a machine-learned allocator who needs to allocate its limited amount of resources to the locations in order to discover the incidents, where every event needs one unit of attention to be discovered. We design a runtime monitor to check, in real-time, if the allocator is fair in allocating its limited amount of attention among two particular locations A and B. Suppose in each location and at every time step, some number of events appear according to the Poisson distribution with unknown parameters. At any location and at any given time, the rate with which events appear is inversely proportional to the number of attention units allocated to that location at the last time step; the exact relationship will be formalized in Sec. 5.1. We assume that the allocator has knowledge about this relationship. The fairness criteria that we wish to monitor requires that the probability with which any event will be discovered across the two locations should be close to each other.

We streamline our exposition on a simpler instance of the original problem that was studied by D'Amour et al. [9], where they considered the fairness measure as the maximum pairwise disparity in discovery probabilities. We point out that this general property can also be handled using our monitors, by simply having a separate monitor for each pair of locations, and then aggregating the outputs of all the monitors using interval arithmetic and union bound.

5.1 Problem Formulation

The agent and the environment: Here, the two locations A and B are the two groups, as well as the only members in the respective groups. The feature of each location is the number of events (in $\mathbb{N}$). At every time step t, the environment samples a pair of (random) features X_t^A and X_t^B for the two locations, such that $X_t^g \sim \mathit{Poisson}(\lambda_t^g)$, where $\mathit{Poisson}(\lambda_t^g)$ represents the Poisson distribution with parameter λ_t^g. The numbers of incidents in the two locations are given by $X_t^A + 1$ and $X_t^B + 1$, ensuring that the minimum number of events is 1, which is necessary for technical reasons (will be explained later). Observe that, we slightly deviate from the setting that we introduced in Sec. 2, in that we obtain features of two individuals (i.e., the locations) from both groups simultaneously at each time step. Technique-wise, this is not a problem since we are going to use two separate ExpEstimator monitors for the two locations. Notation-wise, this is simpler, as otherwise we would need *vectors*-valued features.

The agent's action is a (random) vector $(Y_t^A, Y_t^B) \in \mathbb{N}^2$ to allocate its K units of attention to the respective locations, i.e. $Y_t^A + Y_t^B \leq K$. The entry Y_t^g represents the number of attention units allocated to location g at time t. In this example, the reaction of the environment to the agent's action has no role, i.e., we set $Z_t = \bot$ (a dummy symbol).

As usual, we assume a change function Δ_t is given (defined below) that causes a shift in the expected value of X_t. Since the expected value of $\mathit{Poisson}(\lambda_t^g)$ is λ_t^g, this corresponds to a change in the Poisson parameter, causing changes in the rate at which events appear in the respective locations in the next step. Given a fixed parameter $\gamma > 0$, which controls how dynamic the system is, the change function is defined as:

$$\forall g \in \{A, B\} \ . \ \Delta(g, x_t^g, y_t^g) = \begin{cases} \gamma & \text{if } y_t^g = 0, \\ -\gamma \cdot y_t^g & \text{otherwise,} \end{cases}$$

where we drop the fourth argument z_t from Δ_t as it has no role.

The fairness property: The well-being function f in (1) in this example is called the discovery probability, and we want to monitor its disparity between the two locations. The discovery probability at time t and in location g can be formalized as the expected value

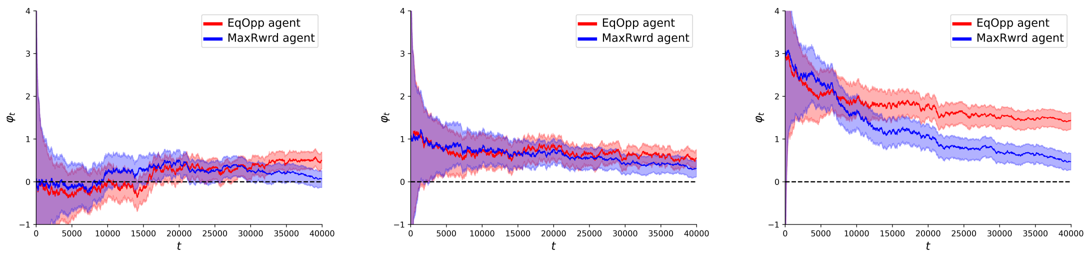

Figure 2: Output estimates of the monitors at each time step on simulated trajectories for the lending example obtained from `ml-fairness-gym`. The three plots are ordered from left to right in the increasing order of initial bias. For each case, we considered two different policies of the agent: the `MaxRwrd` agent (blue) maximizes its own reward without trying to optimize any fairness criterion, whereas the `EqOpp` agent (red) also tries to ensure equalized opportunity statically (i.e., in its one-shot decisions). The shaded regions are the intervals computed by the monitor LendingMonitor, whereas the solid lines are the (unknown to the monitor) true values of the properties. The horizontal dashed line corresponds to the perfectly fair scenario (i.e., $\varphi_t = 0$).

of the ratio of the discovered events $\overline{X}_t^g = \min\{X_t^g + 1, Y_t^g\}$ to the the actual number of events $X_t^g + 1$. Notice that had we defined the number of events as X_t^g, discovery probability would be undefined (since X_t^g can be zero). The discovery probability at time t for a given observation sequence $\vec{o}_t$ can also be represented as the following conditional expectation:

$$\omega_t^g := \mathbb{E}_g\left(\frac{\overline{X}_t^g}{X_t^g + 1}\,\middle|\,\vec{o}_{t-1}, y_t^g\right).$$

The time-varying social fairness criteria at every time t is given by $\varphi_t := \omega_t^A - \omega_t^B$.

5.2 The Runtime Monitor

We show that ω_t^g has a closed-form expression $\eta(y_t^g, \lambda_t^g)$ for a given concrete λ_t^g and y_t^g, where

$$\eta(y, \lambda) := e^{-\lambda} \sum_{k=0}^{y-1} \frac{\lambda^k}{k!}\left(1 - \frac{y}{k+1}\right) + \frac{y}{\lambda}\left(1 - e^{-\lambda}\right). \tag{8}$$

Furthermore, we show that, for a fixed y, the function $\eta(y, \cdot)$ is strictly decreasing everywhere in the positive reals (proof is in the longer version [22]). This property of $\eta(y, \cdot)$ enables us to efficiently compute an interval estimate of $\eta(y, \cdot)$ and, in turn ω_t^g, from an interval estimate of λ.

Using these auxiliary results, we construct the monitor as follows. We first use the general monitor `ExpEstimator` from Alg. 1 to estimate, for each location g, the expected value $\mathbb{E}_g(X_t^g)$, which is the same as λ_t^g (follows from the property of the Poisson distributions). We make two mild assumptions: First, we assume that the Poisson parameters in both locations are bounded between two positive reals $\underline{\lambda} < \overline{\lambda}$, allowing us to establish that X_t^g is a sub-exponential random variable with parameters $(2\lambda, 2)$ (proof is in the longer version [22]). Second, we assume that the sequence of observations are such that the parameter would not reach zero no matter what its true initial value was. Otherwise the function Δ would no longer reflect the differences in the expected values. This can be checked by the monitor at runtime, by checking if the parameter would reach zero had it started from $\underline{\lambda}$ in the worst case (the value closest to zero). We omit the check for simplicity.

Suppose $[\lambda_{\min}^g, \lambda_{\max}^g]$ is the interval output by `ExpEstimator` as the estimate of λ_t^g. Then from $\omega_t^g = \eta(y_t^g, \lambda_t^g)$ and the strictly decreasing property of η, we obtain the corresponding interval estimate for ω_t^g as $[\eta(y_t^g, \lambda_{\max}), \eta(y_t^g, \lambda_{\min})]$. From the interval estimates for the discovery probability of each group, we obtain the overall fairness estimate by computing the interval difference, as we did for the lending monitor. The pseudocode of the monitor is in Alg. 3.

Algorithm 3 AttentionMonitor

```
1: function Init($\Delta, \delta, \overline{\lambda}$)
2:     $M_A \leftarrow$ ExpEstimator.Init($\Delta, \frac{\delta}{2}, 2\overline{\lambda}, 2$)
3:     $M_B \leftarrow$ ExpEstimator.Init($\Delta, \frac{\delta}{2}, 2\overline{\lambda}, 2$)
1: function Compute($(x^A, x^B), (y^A, y^B)$)
2:     $[\lambda_{\min}^A, \lambda_{\max}^A] \leftarrow M_A$.Compute($x^A, A, y^A, \perp$)
3:     $[\lambda_{\min}^B, \lambda_{\max}^B] \leftarrow M_B$.Compute($x^B, B, y^B, \perp$)
4:     $\overline{E}_A \leftarrow [\eta(y^A, \lambda_{\max}^A), \eta(y^A, \lambda_{\min}^A)]$
5:     $\overline{E}_B \leftarrow [\eta(y^B, \lambda_{\max}^B), \eta(y^B, \lambda_{\min}^B)]$
6:     $\overline{\varphi} \leftarrow \overline{E}_A - \overline{E}_B$        ▷interval difference
7:     return $\overline{\varphi}$
```

5.3 Experimental Outcome

We summarize the outputs of our monitor in Fig. 3. We consider three types of agent, uniform, greedy, and fair-constrained greed (with $\alpha = 0.75$). While the uniform agent behaves randomly, without taking into account the actual incidence rate, the greedy agent tries to minimize the chances of missed discoveries by keeping an estimate of the incidence rates. The constrained greedy agent needs to additionally ensure a fairness criteria. Our experiments empirically show that no matter what the experimental conditions are, our monitor is able to provide real-time information about the time-varying biases in the system. Moreover, the monitor AttentionMonitor took on an average 28 µs to compute a new estimate after receiving every new observation. The experiments demonstrate the practical usefulness of our monitors.

6 CONCLUSION AND FUTURE WORK

We present an approach for real-time monitoring of the long-run fairness of machine-learned agents deployed in dynamic environments. Our monitors observe a long sequence of events generated from the interactions between the agent and its environment, and output, after each new observation, a quantitative PAC-style estimate of how fair or biased the agent's policy was until that point in time. The strength of our monitors is their ability to compute interval estimates of fairness values in the face of frequent changes in the underlying distribution, a setting that prevents a static estimation of fairness at each time step. The presented method allows for the computation of confidence intervals when the monitored random variable is sub-exponential. By extension, they can handle sub-Gaussian random variables as well. Using a prototype implementation, we demonstrated the practical usefulness of the monitoring approach on examples taken from the literature.

We took great effort to ensure that the interval estimates computed by the monitor hold non-asymptotically. Consequently, we avoided a direct comparison with methods that rely on the central limit theorem. However, we acknowledge that loosening this restriction will allow for a wider range of applications. Furthermore, we showed computations of PAC estimates on fairness properties with some specific well-being functions, such as the expected credit score in the lending problem and the discovery probability in the dynamic attention allocation problem, etc. Although an extension to general well-being functions is difficult, a generalization to restricted classes of well-being functions is conceivable: for instance, when the well-being function is in the form of arithmetic expressions, we can use interval arithmetic to deduce the overall PAC bound, or when the well-being function is convex, we can use convex optimization to deduce tight PAC estimates, etc.

We see several immediate future directions. Firstly, we considered only one particular class of fairness properties, namely the ones which can be written as a difference between expected values of a given function of population parameters across two groups. Investigating other classes of properties will be an important goal. We will be able to extend our monitors to handle individuals with multiple features, i.e. feature vectors, without adding any additional technical machinery. Secondly, we assumed perfect observability about the actions of the agent and the environment, whereas in reality actions are often either partially observable or the observations are noisy (e.g., the college admission example in D'Amour et al. [9]). Thus extensions to partially observed and noisy models will be helpful. Thirdly, our monitors require, at least partially, information about the changes in the system. A natural extension, would be to relax this condition. Finally, we only considered the monitoring, i.e., the problem of checking fairness in real-time. The next step will be to combine monitoring and intervention, so that we obtain an automated procedure for controlling dynamic fairness properties of a machine-learned agent.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported by the European Research Council under Grant No.: ERC-2020-AdG 101020093.

REFERENCES

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2018. A reductions approach to fair classification. In *International Conference on Machine Learning*. PMLR, 60–69.

[2] Aws Albarghouthi and Samuel Vinitsky. 2019. Fairness-aware programming. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 211–219.

[3] Ezio Bartocci and Yliès Falcone. 2018. *Lectures on Runtime Verification*. Springer.

[4] Richard Berk, Hoda Heidari, Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. 2017. A convex framework for fair regression. *arXiv preprint arXiv:1706.02409* (2017).

[5] Roderick Bloem, Bettina Könighofer, Robert Könighofer, and Chao Wang. 2015. Shield synthesis: Runtime enforcement for reactive systems. In *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*. Springer, 533–548.

[6] Toon Calders and Indrė Žliobaitė. 2013. Why unbiased computational processes can lead to discriminative decision procedures. *Discrimination and Privacy in the Information Society: Data mining and profiling in large databases* (2013), 43–57.

[7] Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. 2018. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM conference on recommender systems*. 224–232.

[8] Yifang Chen, Alex Cuellar, Haipeng Luo, Jignesh Modi, Heramb Nemlekar, and Stefanos Nikolaidis. 2020. The fair contextual multi-armed bandit. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*.

[9] Alexander D'Amour, Hansa Srinivasan, James Atwood, Pallavi Baljekar, David Sculley, and Yoni Halpern. 2020. Fairness is not static: deeper understanding of long term fairness via simulation studies. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 525–534.

[10] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*. 214–226.

[11] Hadi Elzayn, Shahin Jabbari, Christopher Jung, Michael Kearns, Seth Neel, Aaron Roth, and Zachary Schutzman. 2019. Fair algorithms for learning in allocation problems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 170–179.

[12] Danielle Ensign, Sorelle A Friedler, Scott Neville, Carlos Scheidegger, and Suresh Venkatasubramanian. 2018. Runaway feedback loops in predictive policing. In *Conference on Fairness, Accountability and Transparency*. PMLR, 160–171.

[13] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 259–268.

[14] Thomas Ferrère, Thomas A. Henzinger, and Bernhard Kragl. 2020. Monitoring Event Frequencies. In *28th EACSL Annual Conference on Computer Science Logic (CSL 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 152)*, Maribel Fernández and Anca Muscholl (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 20:1–20:16. https://doi.org/10.4230/LIPIcs.CSL.2020.20

[15] Andreas Fuster, Paul Goldsmith-Pinkham, Tarun Ramadorai, and Ansgar Walther. 2022. Predictably unequal? The effects of machine learning on credit markets. *The Journal of Finance* 77, 1 (2022), 5–47.

[16] Paula Gordaliza, Eustasio Del Barrio, Gamboa Fabrice, and Jean-Michel Loubes. 2019. Obtaining fairness using optimal transport theory. In *International Conference on Machine Learning*. PMLR, 2357–2365.

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

[18] Drew Harwell. 2018. Amazon's Alexa and Google Home show accent bias, with Chinese and Spanish hardest to understand. http://bit.ly/2QFA1MR. Accessed: 05.02.2023.

[19] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018. Fairness without demographics in repeated loss minimization. In *International Conference on Machine Learning*. PMLR, 1929–1938.

[20] Hoda Heidari, Vedant Nanda, and Krishna P Gummadi. 2019. On the long-term impact of algorithmic decision policies: Effort unfairness and feature segregation through social learning. *arXiv preprint arXiv:1903.01209* (2019).

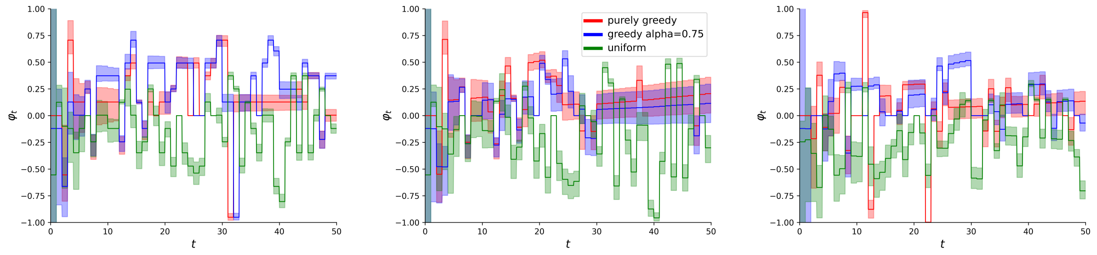

Figure 3: Output estimates of the monitors at each time step on simulation traces for the attention allocation example obtained from `ml-fairness-gym`. Left: $L = 5$ (no. of locations), $K = 6$ (total units of attention), $\gamma = 0$ (change in the Poisson parameters). Middle: $L = 5$, $K = 6$, $\gamma = 0.0025$. Right: $L = 10$, $K = 10$, $\gamma = 0.0025$. For each case, we considered three different policies of the agent (description in the text). The shaded regions are the intervals computed by the monitor AttentionMonitor, whereas the solid lines are the (unknown) true values of the properties.

[21] Thomas A. Henzinger, Mahyar Karimi, Konstantin Kueffner, and Kaushik Mallik. 2023. Monitoring Algorithmic Fairness. In *Computer Aided Verification*. (to appear).

[22] Thomas A. Henzinger, Mahyar Karimi, Konstantin Kueffner, and Kaushik Mallik. 2023. Runtime Monitoring of Dynamic Fairness Properties. arXiv:2305.04699 [cs.CY]

[23] Safwan Hossain, Evi Micha, and Nisarg Shah. 2021. Fair algorithms for multi-agent multi-armed bandits. *Advances in Neural Information Processing Systems* 34 (2021), 24005–24017.

[24] Faisal Kamiran and Toon Calders. 2012. Data preprocessing techniques for classification without discrimination. *Knowledge and information systems* 33, 1 (2012), 1–33.

[25] Lydia T Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. 2018. Delayed impact of fair machine learning. In *International Conference on Machine Learning*. PMLR, 3150–3158.

[26] Lydia T Liu, Ashia Wilson, Nika Haghtalab, Adam Tauman Kalai, Christian Borgs, and Jennifer Chayes. 2020. The disparate equilibria of algorithmic decision making when individuals invest rationally. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 381–391.

[27] Meiyi Ma, John A Stankovic, and Lu Feng. 2017. Runtime monitoring of safety and performance requirements in smart cities. In *Proceedings of the 1st ACM Workshop on the Internet of Safe Things*. 44–50.

[28] Jianhui Mao and Liqian Chen. 2012. Runtime monitoring for cyber-physical systems: a case study of cooperative adaptive cruise control. In *2012 Second International Conference on Intelligent System Design and Engineering Application*. IEEE, 509–515.

[29] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[30] Hussein Mouzannar, Mesrob I Ohannessian, and Nathan Srebro. 2019. From fair decision making to social equality. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 359–368.

[31] Katsuhiko Ogata et al. 2010. *Modern control engineering*. Vol. 5. Prentice hall Upper Saddle River, NJ.

[32] Aleksandr Podkopaev and Aaditya Ramdas. 2021. Tracking the risk of a deployed model and detecting harmful distribution shifts. *arXiv preprint arXiv:2110.06177* (2021).

[33] Martin J Wainwright. 2019. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge university press.

[34] Ian Waudby-Smith, David Arbour, Ritwik Sinha, Edward H Kennedy, and Aaditya Ramdas. 2021. Time-uniform central limit theory, asymptotic confidence sequences, and anytime-valid causal inference. *arXiv preprint arXiv:2103.06476* (2021).

[35] Ian Waudby-Smith and Aaditya Ramdas. 2020. Estimating means of bounded random variables by betting. *arXiv preprint arXiv:2010.09686* (2020).

[36] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*. 1171–1180.

[37] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P Gummadi. 2019. Fairness constraints: A flexible approach for fair classification. *The Journal of Machine Learning Research* 20, 1 (2019), 2737–2778.

[38] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International conference on machine learning*. PMLR, 325–333.

[39] Xueru Zhang, Mohammadmahdi Khaliligarekani, Cem Tekin, et al. 2019. Group retention when using machine learning in sequential decision making: the interplay between user dynamics and fairness. *Advances in neural information processing systems* 32 (2019).

[40] Xueru Zhang and Mingyan Liu. 2021. Fairness in learning-based sequential decision algorithms: A survey. In *Handbook of Reinforcement Learning and Control*. Springer, 525–555.