An Algorithmic Framework for Bias Bounties

Ira Globus-Harris
University of Pennsylvania and Amazon AWS AI
USA
igh@seas.upenn.edu

Michael Kearns
University of Pennsylvania and Amazon AWS AI
USA
mkearns@cis.upenn.edu

Aaron Roth
University of Pennsylvania and Amazon AWS AI
USA
aaroth@cis.upenn.edu

ABSTRACT

We propose and analyze an algorithmic framework for "bias bounties" — events in which external participants are invited to propose improvements to a trained model, akin to bug bounty events in software and security. Our framework allows participants to submit arbitrary subgroup improvements, which are then algorithmically incorporated into an updated model. Our algorithm has the property that there is no tension between overall and subgroup accuracies, nor between different subgroup accuracies, and it enjoys provable convergence to either the Bayes optimal model or a state in which no further improvements can be found by the participants. We provide formal analyses of our framework, experimental evaluation, and findings from a preliminary bias bounty event.[1]

CCS CONCEPTS

• **Theory of computation → Machine learning theory**.

KEYWORDS

bias bounty, subgroup fairness, multigroup fairness

ACM Reference Format:
Ira Globus-Harris, Michael Kearns, and Aaron Roth. 2022. An Algorithmic Framework for Bias Bounties. In *FAccT '22: ACM Conference on Fairness, Accountability, and Transparency*. ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3531146.3533172

1 INTRODUCTION

Modern machine learning (ML) is well known for its powerful applications, and more recently, for its potential to train discriminatory models. As organizations become more aware of these negative consequences, many are instituting partial remedies such as responsible AI teams, auditing practices, and model cards [28] and data sheets [16] that document ML workflows.

Such practices are likely to remain insufficient for at least two reasons. First, even an organization that diligently audits its models on carefully curated datasets cannot anticipate all possible downstream use cases. A face recognition system designed and tested for use in well-lit, front-facing conditions may be applied by a customer in less ideal conditions, leading to degradation in performance overall and on particular subgroups. Second, we live in an era of AI activism, in which teams of researchers, journalists, and other external parties can independently audit models with commercial or open source APIs, and publish their findings in both the research community and mainstream media. In the absence of more stringent regulation of AI and ML, this activism is one of the strongest forces pushing for change on organizations developing ML models and tools. It has the ability to leverage distributed teams of researchers searching for a wide variety of kinds of problems in deployed models and systems.

Unfortunately, the dynamic between ML developers and their critics currently has a somewhat adversarial tone. An external audit will often be conducted privately, and its results aired publicly, without prior consultation with the organization that built the model and system. That organization may endure criticism, and attempt internally to fix the identified biases or problems, but typically there is little or no direct interaction with external auditors.

Over time, the software and security communities have developed "bug bounties" in an attempt to turn similar dynamics between system developers and their critics (or hackers) towards more interactive and productive ends. The hope is that by deliberately inviting external parties to find software or hardware bugs in their systems, and often providing monetary incentives for doing so, a healthier and more rapidly responding ecosystem will evolve.

It is natural for the ML community to consider a similar "bias bounty" approach to the timely discovery and repair of models and systems with bias or other undesirable behaviors. Rather than finding bugs in software, external parties are invited to find biases — for instance, (demographic or other) subgroups of inputs on which a trained model underperforms — and are rewarded for doing so. Indeed, we are already starting to see early field experiments in such events [6].

In this work, we propose and analyze a principled algorithmic framework for conducting bias bounties against an existing trained model $f(x)$. Our framework has the following properties and features:

- "Bias hunters" audit f by submitting pairs of models (g, h)— a model g identifying a subset of inputs to f that f performs poorly on, and a proposed model h which improves on this subset. Requiring participants to identify not just a group but an improved model ensures that improvement on the group is in fact possible (as opposed to identifying a fundamentally hard subgroup, such as images of occluded faces in a face recognition task).
- The proposed groups do not need to be identified in advance, and the improving models do not need to be chosen

[1] The most up-to-date version of this paper may be found at https://arxiv.org/abs/2201.10408

FAccT '22, 2022,

ACM ISBN 978-1-4503-9352-2/22/06…$15.00
https://doi.org/10.1145/3531146.3533172

from predetermined parametric classes; indeed, both groups and improving models can be arbitrarily complex. This is in contrast to most fair ML frameworks, in which training is formulated as a constrained optimization problem over a fixed parametric family of models, with fairness constraints over fixed demographic subgroups.
- Given a proposed group and model pair, our algorithm validates the proposed improvement on a holdout set, using techniques from adaptive data analysis to circumvent the potential for overfitting due to the potentially unlimited complexity of the submitted pair.
- If the improvement is validated, our algorithm has a simple mechanism for automatically incorporating the improvement into an update of f in a way that reduces *both the overall error and the error on the proposed subgroup*. Indeed, our algorithm has the further property that once a subgroup improvement has been accepted, the error on that subgroup will never increase (much) due to subsequent subgroup introductions. Thus there is *no tradeoff between overall and subgroup errors, or between different subgroups*. This is again in contrast to constrained optimization approaches, where there is necessarily tension between fairness and accuracy. Our algorithm achieves this through the use of a new model class called *pointer decision lists*.
- Our algorithm provably and monotonically converges quickly to one of two possible outcomes: either we reach the Bayes optimal model, or we reach a model that cannot be distinguished from the Bayes optimal, in the sense that the bias hunters can find no further improvements. If payments are made to the bias hunters in proportion to the scale of the improvement, we guarantee that the total payments made to the bias hunters can be bounded in advance.
- We can alternatively view our framework as an entirely algorithmic approach to training (minimax) fair and accurate models, by replacing the bias hunters with automated mechanisms for finding improving pairs. We propose and analyze two such mechanisms.

Our contributions include the introduction of the bias bounty framework; proofs of the convergence, generalization and monotonicity properties of our algorithm; experimental results on census-derived Folktables datasets [9]; and preliminary findings from an initial bias bounty event held in an undergraduate class at the University of Pennsylvania.

1.1 Limitations and Open Questions

The primary limitation of our proposed framework is that it can only identify and correct sub-optimal performance on groups *as measured on the data distribution for which we have training data*. It does not solve either of the following related problems:

(1) Our model appears to perform well on every group *only because in gathering data, we have failed to sample representative members of that group*.
(2) The model that we have cannot be improved on some group *only because we have failed to gather the features that are most predictive on that group*, and the performance would be improvable if only we had gathered better features.

That is, our framework can be used to find and correct biases as measured on the data distribution from which we have data, but cannot be used to find and correct biases that come from having gathered the wrong dataset. In both cases, one primary obstacle to extending our framework is the need to be able to efficiently validate proposed fixes. For example, because we restrict attention to a single data distribution, given a proposed pair (g, h), we can check on a holdout set whether h in fact has improved performance compared to our model, on examples from group g. This is important to disambiguate distributional improvements compared to subsets of examples that amount to cherrypicking. How can we approach this problem when proposed improvements include evaluations on new datasets, for which we by definition do not have held out data? Compelling solutions to this problem seem to us to be of high interest. We remark that a bias bounty program held under our proposed framework would at least serve to highlight *where* new data collection efforts are needed, by disambiguating failures of training from failures for the data to properly represent a population: if a group continues to have persistently high error even in the presence of a large population of auditors in our framework, this is evidence that in order to obtain improved error on that group, we need to focus on better representing them within our data.

1.2 Related Work

There are several strands of the algorithmic fairness literature that are closely related to our work. Most popular notions of algorithmic fairness (e.g. those that propose to equalize notions of error across protected groups as in e.g. [1, 18, 24, 34], or those that aim to "treat similar individuals similarly" as in e.g. [14, 20, 21, 33]) involve tradeoffs, in that asking for "fairness" involves settling for reduced accuracy. Several papers [5, 10] show that fairness constraints of these types need not involve tradeoffs (or can even be accuracy improving) *on test data* if the training data has been corrupted by some bias model and is not representative of the test data. In cases like this, fairness constraints can act as corrections to undo the errors that have been introduced in the data. These kinds of results leverage differences between the training and evaluation data, and unlike our work, do not avoid tradeoffs between fairness and accuracy in settings in which the training data is representative of the true distribution.

A notable exception to the rule that fairness and accuracy must involve tradeoffs is the literature on multicalibration initiated by Hébert-Johnson et al. [15, 17, 19, 22, 26] that asks that a model's predictions be calibrated not just overall, but also when restricted to a large number of protected subgroups g. Hébert-Johnson et al. [19] and Kim, Ghorbani, and Zou [26] show that an arbitrary model f can be postprocessed to satisfy multicalibration (and the related notion of "multi-accuracy") without sacrificing (much) in terms of model accuracy. Our aim is to achieve something similar, but for predictive error, rather than model calibration. There are two things driving the "no tradeoff" result for multicalibration, from which we take inspiration: 1) the fairness notion that they ask for is satisfied by the Bayes-optimal model, and 2) they do not optimize over a fixed model class, but rather a model class defined in terms of the groups $\mathcal{G}$ that define their fairness notion. These two things will be true for us as well.

The notion of fairness that we ask for in this paper was studied in an online setting (in which the data, rather than the protected groups arrive online) by Blum and Lykouris [4] and generalized by Rothblum and Yona [31] as "multigroup agnostic learnability." Noarov, Pai, and Roth [29] show how to obtain it in an online setting as part of the same unified framework of algorithms that can obtain multicalibrated predictions. The algorithmic results in these papers lead to complex models — in contrast, our algorithm produces "simple" predictors in the form of a decision list. In contrast to these prior works, we do not view the set of groups $\mathcal{G}$ that we wish to offer guarantees to as fixed up front, but instead as something that can be discovered online, after models are deployed. Our focus is on fast algorithms to update existing models when new groups g on which our model is performing poorly are discovered.

Concurrently and independently of our paper, Tosh and Hsu [32] study algorithms and sample complexity for multi-group agnostic learnability and give an algorithm ("Prepend") that is equivalent to our Algorithm 1 ("ListUpdate"). Their focus is on sample complexity of batch optimization, however — in contrast to our focus on the discovery of groups on which our model is performing poorly online (e.g. as part of a "bias bounty program"). They also are not concerned with the details of the optimization that needs to be solved to produce an update — we give practical algorithms based on reductions to cost sensitive classification and empirical evaluation. Tosh and Hsu [32] also contains additional results, including algorithms producing more complex hypotheses but with improved sample complexity (again in the setting in which the groups are fixed up front).

The multigroup notion of fairness we employ [4, 31] aims to perform *optimally* on each subgroup g, rather than *equalizing* the performance across subgroups. This is similar in motivation to *minimax* fairness, studied in [7, 8, 27], which aim to minimize the error on the maximum error sub-group. Our approach avoids tradeoffs by optimizing over a class that is dynamically expanded as the set of groups to be protected expands.

The idea of a "bias bug bounty" program dates back at least to a 2018 editorial of Amit Elazari Bar On [30], and Twitter ran a version of a bounty program in 2021 to find bias issues in its image cropping algorithm [6]. These programs are generally envisioned to be quite different than what we propose here. On the one hand, we are proposing to automatically audit models and award bounties for the discovery of a narrow form of technical bias — sub-optimal error on well defined subgroups — whereas the bounty program run by Twitter was open ended, with human judges and written submissions. On the other hand, the method we propose could underly a long-running program that could automatically correct the bias issues discovered in production systems at scale, whereas Twitter's bounty program was a labor intensive event that ran over the course of a week.

2 PRELIMINARIES

We consider a supervised learning problem defined over *labelled examples* $(x, y) \in \mathcal{X} \times \mathcal{Y}$. This can represent (for example) a binary classification problem if $\mathcal{Y} = \{0, 1\}$, a discrete multiclass classification problem if $\mathcal{Y} = \{1, \ldots, k\}$, or a regression problem if $\mathcal{Y} = \mathbb{R}$. We write $\mathcal{D}$ to denote a joint probability distribution over labelled examples: $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$. We will write $D \sim \mathcal{D}^n$ to denote a dataset consisting of n labelled examples sampled i.i.d. from $\mathcal{D}$. Our goal is to learn some *model* represented as a function $f : \mathcal{X} \to \mathcal{Y}$ which aims to predict the label of an example from its features, and we will evaluate the performance of our model with a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$, where $\ell(\hat{y}, y)$ represents the "cost" of mistakenly labelling an example that has true label y with the prediction $f(x) = \hat{y}$. We will be interested in the performance of models f not just *overall* on the underlying distribution, but also on particular subgroups of interest. A subgroup corresponds to an arbitrary subset of the feature space $\mathcal{X}$, which we will model using an indicator function:

Definition 2.1 (Subgroups). A *subgroup* of the feature space $\mathcal{X}$ will be represented as an indicator function $g : \mathcal{X} \to \{0, 1\}$. We say that $x \in \mathcal{X}$ is in group g if $g(x) = 1$ and x is *not* in group g otherwise. Given a group g, we write $\mu_g(\mathcal{D})$ to denote its measure under the probability distribution $\mathcal{D}$: $\mu_g(\mathcal{D}) = \Pr_{\mathcal{D}}[g(x) = 1]$. We write $\mu_g(D)$ to denote the corresponding empirical measure under D, which results from viewing D as the uniform distribution over its elements.

We can now define the loss of a model both overall and on different subgroups:

Definition 2.2 (Model Loss). Given a model $f : \mathcal{X} \to \mathcal{Y}$ We write $L(\mathcal{D}, f)$ to denote the average loss of f on distribution $\mathcal{D}$: $L(\mathcal{D}, f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$. We write $L(\mathcal{D}, f, g)$ to denote the loss on f conditional on membership in g:

$$L(\mathcal{D}, f, g) = \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)|g(x) = 1].$$

Given a dataset D, we write $L(D, f)$ and $L(D, f, g)$ to denote the corresponding empirical losses on D, which result from viewing D as the uniform distribution over its elements.

The best we can ever hope to do in any prediction problem (fixing the loss function and the distribution) is to make predictions that are as accurate as those of a Bayes optimal model:

Definition 2.3. A *Bayes Optimal* model $f^* : \mathcal{X} \to \mathcal{Y}$ with respect to a loss function ℓ and a distribution $\mathcal{D}$ satisfies: $f^*(x) \in \arg\min_{y\in\mathcal{Y}} \mathbb{E}_{(x',y')\sim\mathcal{D}}[\ell(y, y')|x' = x]$, where $f^*(x)$ can be defined arbitrarily for any x that is not in the support of $\mathcal{D}$.

The Bayes optimal model is pointwise optimal, and hence has the lowest loss of any possible model, not just overall, but simultaneously on *every* subgroup. In fact, its easy to see that this is a characterization of Bayes optimality.

Observation 2.4. *Fixing a loss function ℓ and a distribution $\mathcal{D}$, f^* is a Bayes optimal model if and only if for every group g and every alternative model h: $L(\mathcal{D}, f^*, g) \leq L(\mathcal{D}, h, g)$.*

The above characterization states that a model is Bayes optimal if and only if it induces loss that is as exactly as low as that of *any possible* model h, when restricted to *any possible* group g. It will also be useful to refer to approximate notions of Bayes optimality, in which the exactness is relaxed, as well as possibly the class of comparison models $\mathcal{H}$, and the class of groups $\mathcal{G}$. We call this $(\epsilon, \mathcal{G}, \mathcal{H})$-Bayes optimality to highlight the connection to (exact) Bayes Optimality, but it is identical to what Rothblum and Yona

[31] call a "multigroup agnostic PAC solution" with respect to $\mathcal{G}$ and $\mathcal{H}$. Related notions were also studied in [4, 29].

Definition 2.5. A model $f : \mathcal{X} \to \mathcal{Y}$ is $(\epsilon, \mathcal{C})$-Bayes optimal with respect to a collection of (group, model) pairs $\mathcal{C}$ if for each $(g, h) \in \mathcal{C}$, the performance of f on g is within ϵ of the performance of h on g. In other words, for every $(g, h) \in \mathcal{C}$: $L(\mathcal{D}, f, g) \leq L(\mathcal{D}, h, g) + \frac{\epsilon}{\mu_g(\mathcal{D})}$. When $\mathcal{C}$ is a product set $\mathcal{C} = \mathcal{G} \times \mathcal{H}$, then we call f "$(\epsilon, \mathcal{G}, \mathcal{H})$-Bayes Optimal" and the condition is equivalent to requiring that for every $g \in G$, f has performance on g that is within ϵ of the *best* model $h \in \mathcal{H}$ on g. When $\mathcal{G}$ and $\mathcal{H}$ represent the set of all groups and models respectively, we call f ϵ-Bayes optimal.

Remark 2.6. *We have chosen to define approximate Bayes optimality by letting the approximation term ϵ scale proportionately to the inverse probability of the group g, similar to how notions of multigroup fairness are defined in [17, 22, 24]. An alternative (slightly weaker) option would be to require error that is uniformly bounded by ϵ for all groups, but to only make promises for groups g that have probability μ_g larger than some threshold, as is done in [19]. Some relaxation of this sort is necessary to provide guarantees on an unknown distribution based only on a finite sample from it, since we will necessarily have less statistical certainty about smaller subsets of our data.*

Note that $(\epsilon, \mathcal{G}, \mathcal{H})$-Bayes optimality is identical to Bayes optimality when $\epsilon = 0$ and when $\mathcal{G}$ and $\mathcal{H}$ represent the classes of all possible groups and models respectively, and that it becomes an increasingly stronger condition as $\mathcal{G}$ and $\mathcal{H}$ grow in expressivity.

3 CERTIFICATES OF SUB-OPTIMALITY AND UPDATE ALGORITHMS

Recall that "bias hunters" submit a group that our existing model performs poorly on and an improvement on that group. We need to formulate what our requirements for accepting their proposed improvement would be, and develop a method to incorporate these fixes into our model without retraining it. Formally, suppose we have an existing model f, and we find that it is performing suboptimally on some group g. By Observation 2.4, it must be that f is not Bayes optimal, and this will be witnessed by some model h such that: $L(\mathcal{D}, f, g) > L(\mathcal{D}, h, g)$. We call such a pair (g, h) a certificate of sub-optimality. We can define a quantitative version of these certificates:

Definition 3.1. A group indicator function $g : \mathcal{X} \to \{0, 1\}$ together with a model $h : \mathcal{X} \to \mathcal{Y}$ form a (μ, Δ)-certificate of sub-optimality for a model f under distribution $\mathcal{D}$ if:

(1) Group g has probability mass at least μ under $\mathcal{D}$: $\mu_g(\mathcal{D}) \geq \mu$, and
(2) h improves on the performance of f on group g by at least Δ: $L(\mathcal{D}, f, g) \geq L(\mathcal{D}, h, g) + \Delta$

The core of our algorithmic updating framework will rely on a close quantitative connection between certificates of sub-optimality and approximate Bayes optimality. The following theorem can be viewed as a quantitative version of Observation 2.4. Its proof can be found in the appendix.

Theorem 3.2. *Fix any $\epsilon > 0$, and any collection of (group,model) pairs $\mathcal{C}$. There exists a (μ, Δ)-certificate of sub-optimality $(g, h) \in \mathcal{C}$ for f if and only if f is not $(\epsilon, \mathcal{C})$-Bayes optimal for $\epsilon < \mu\Delta$.*

Theorem 3.2 tells us that if we are looking for evidence that a model f fails to be Bayes Optimal (or more generally, fails to be $(\epsilon, \mathcal{C})$-Bayes optimal), then without loss of generality, we can restrict our attention to certificates of sub-optimality with large parameters — these exist if and only if f is significantly far from Bayes optimal. But it does not tell us what to do if we find such a certificate. Can we use a certificate of sub-optimality (g, h) for f to easily update f to a new model that both corrects the suboptimality witnessed by (g, h) and makes measurable progress towards Bayes Optimality? It turns out that the answer is *yes*, and we can do this with an exceedingly simple update algorithm, which we analyze next. The update algorithm (Algorithm 1) takes as input a model f together with a certificate of sub-optimality for f, (g, h), and outputs an improved model based on the following intuitive update rule: If an example x is in group g (i.e. if $g(x) = 1$), then we will classify x using h; otherwise we will classify x using f.

Input: A model f_t, and a certificate of suboptimality (g_{t+1}, h_{t+1})
Define f_{t+1} as follows:

$$f_{t+1}(x) = \begin{cases} f_t(x) & \text{if } g_{t+1}(x) = 0 \\ h_{t+1}(x) & \text{if } g_{t+1}(x) = 1 \end{cases}$$

Output: f_{t+1}

Algorithm 1: ListUpdate$(f_t, (g_{t+1}, h_{t+1}))$: An Update Algorithm Constructing a Decision List

Theorem 3.3. *Algorithm 1 (ListUpdate) has the following properties. If (g_{t+1}, h_{t+1}) form a (μ, Δ)-certificate of sub-optimality for f_t, and $f_{t+1} = ListUpdate(f_t, (g_{t+1}, h_{t+1}))$ then:*

(1) *The new model matches the performance of h_{t+1} on group g_{t+1}: $L(\mathcal{D}, f_{t+1}, g_{t+1}) = L(\mathcal{D}, h_{t+1}, g_{t+1})$, and*
(2) *The overall performance of the model is improved by at least $\mu \cdot \Delta$: $L(\mathcal{D}, f_{t+1}) \leq L(\mathcal{D}, f_t) - \mu \cdot \Delta$.*

The proof can be found in the Appendix. We can use Algorithm 1 as an iterative update algorithm: If we have a model f_0, and then discover a certificate of sub-optimality (g_1, h_1) for f_0, we can update our model to a new model f_1. If we then find a new certificate of sub-optimality (g_2, h_2), we can once again use Algorithm 1 to update our model to a new model, f_2, and so on. The result is that at time t, we have a model f_t in the form of a decision list in which the internal nodes branch on the group indicator functions g_i and the leaves invoke the models h_i or the initial model f_0. The fact that each update makes progress towards Bayes Optimality (in fact, optimal progress, given Theorem 3.2) means that this updating process cannot go on for too long:

Theorem 3.4. *Fix any $\epsilon > 0$. For any initial model f_0 with loss $L(\mathcal{D}, f_0) = \ell_0$ and any sequence of models $f_1, \ldots, f_T$, such that $f_i = ListUpdate(f_{i-1}, (g_i, h_i))$ and each pair (g_i, h_i) forms a (μ, Δ)-certificate of suboptimality for f_{i-1} for some μ, Δ such that $\mu \cdot \Delta \geq \epsilon$, the length of the update sequence must be at most $T \leq \frac{\ell_0}{\epsilon} \leq \frac{1}{\epsilon}$.*

What can we do with such an update algorithm? Given a model f_t, we can search for certificates of sub-optimality, and if we find them, we can make quantitative progress towards improving our model. We can then repeat the process. The guarantee of Theorem 3.4 is that this process of searching and updating cannot go on for very many rounds T before we arrive at a model f_T that *our search process is unable to demonstrate is not Bayes Optimal*. How interesting this is depends on what our search process is.

Suppose, for example, that we have an optimization algorithm that for some class of (group,model) pairs C can find a certificate of sub-optimality $(g, h) \in C$ whenever one exists. Paired with our update algorithm, we obtain an algorithm which quickly converges to an (ϵ, C)-Bayes Optimal model. We give such an algorithm in Appendix C.1.

Suppose alternately that we open the search for certificates of sub-optimality to a large and motivated population: for example, to machine learning engineers, regulators and the general public, incentivized by possibly large monetary rewards. In this case, the guarantee of Theorem 3.4 is that the process of iteratively opening our models up to scrutiny and updating whenever certificates of suboptimality are found cannot go on for too many rounds: at convergence, it must be *either* that our deployed model is ϵ-Bayes optimal, *or* that if not, at least nobody can find any evidence to contradict this hypothesis. Since in general it is not possible to falsify Bayes optimality given only a polynomial amount of data and computation, this is in a strong sense the best we can hope for. We give a procedure for checking large numbers of arbitrarily complex submitted proposals for certificates of sub-optimality (e.g. that arrive as part of a bias bounty program) in Section 4.1. There are two remaining obstacles, which we address in the next sections:

(1) Our analysis so far is predicated on our update algorithm being given (Δ, μ) certificates of sub-optimality (g, h). But μ and Δ are defined with respect to the distribution $\mathcal{D}$, and we will not have direct access to $\mathcal{D}$ — we will only have samples drawn from $\mathcal{D}$. So how can we find certificates of sub-optimality and check their parameters?

(2) Theorem 3.3 gives us a guarantee that whenever we are given a certificate of sub-optimality (g_{t+1}, h_{t+1}), our new model f_{t+1} makes improvements both with respect to its error on g_{t+1}, and with respect to overall error. But it does not promise that the update does not increase error for some other previously identified group g_i for $i \leq t$. This would be particularly undesirable in a "bias bounty" application, and would represent the kind of tradeoff that our framework aims to circumvent. However, we show in Section 4.1.1 that (up to small additive terms that come from statistical estimation error), our updates can be made to be groupwise monotonically error improving.

4 OBTAINING CERTIFICATES OF SUBOPTIMALITY

In this section we show how to find and verify proposed certificates of sub-optimality (g, h) given only a finite sample of data $D \sim \mathcal{D}^n$. We consider two important cases: In Section 4.1, we consider the "bias bounty" application in which the discovery of certificates of sub-optimality is crowd-sourced (aided perhaps with API access to the model f_t and a training dataset). In this case, we face two main difficulties: (1) The submitted certificates (g, h) might be arbitrary (and in particular, not guaranteed to come from a class of bounded complexity or expressivity and (2) We expect to receive a very large number of submitted certificates, all of which need to be checked. The first of these difficulties means that we cannot appeal to uniform convergence arguments to obtain rigorous bounds on the sub-optimality parameters μ and Δ. The second of these difficulties means that we cannot naively rely on estimates from a (single, re-used) holdout set to obtain rigorous bounds on μ and Δ. In Section 4.2 we consider the algorithmic application in which the discovery of certificates is treated as an optimization problem over C, for particular classes C. In this case we give two algorithms for finding (ϵ, C)-Bayes optimal models via efficient reductions to cost sensitive classification problems over an appropriately defined class, solved over a polynomially sized dataset sampled from the underlying distribution, which we discuss in detail in Appendix C.1.

4.1 Unrestricted Certificates and Bias Bounties

In this section we develop a procedure to re-use a holdout set to check the validity of a very large number of proposed certificates of sub-optimality (g_i, h_i) with rigorous guarantees. Here we make no assumptions at all about the structure or complexity of either the groups g_i or models h_i, or the process by which they are generated. This allows us the flexibility to model e.g. a public bias bounty, in which a large collection of human competitors use arbitrary methods to find and propose certificates of sub-optimality, potentially adaptively as a function of all of the information that they have available. We use simple description length techniques developed in the adaptive data analysis literature [3, 11]. Somewhat more sophisticated techniques which employ noise addition [2, 12, 13, 23] could also be directly used here to improve the sample complexity bound in Theorems 4.1 and 4.2 by a $\sqrt{1/\epsilon}$ factor, but we elide this for clarity of exposition. First we give a simple algorithm (Algorithm 2) that takes as input a stream of arbitrary adaptively chosen triples (f_i, g_i, h_i), and checks if each (g_i, h_i) form a certificate of sub-optimality for f_i. We then use this as a sub-routine in Algorithm 3 which maintains a sequence of models f_t produced by ListUpdate (Algorithm 1) and takes as input a sequence of proposed certificates (g_i, h_i) which claim to be certificates of sub-optimality for the current model f_t: it updates the current model whenever such a proposed certificate is verified.

Input: Holdout dataset D, Target ϵ, and a stream of submissions $(f_1, g_1, h_1), (f_2, g_2, h_2), \ldots$ of length at most U
NumberAccepted $\leftarrow 0$
while NumberAccepted $\leq 2/\epsilon$ *and* $i \leq U$ **do**
  Consider the next submission (f_i, g_i, h_i)
  Compute $\mu_i \leftarrow \mu_D(g_i), \Delta_i \leftarrow L(D, f_i, g_i) - L(D, h_i, g_i)$
  if $\mu_i \cdot \Delta_i < \frac{3\epsilon}{4}$ **then**
    Output: $\pi_i = \perp$ (Submission Rejected)
  else
    Output: $\pi_i = \top$ (Submission Accepted)
    NumberAccepted $\leftarrow$ **NumberAccepted**+1.

Algorithm 2: CertificateChecker$(\epsilon, D, U, (f_1, g_1, h_1), \ldots)$: An algorithm that takes as input a stream of submissions (f_i, g_i, h_i) and checks if (g_i, h_i) is a certificate of sub-optimality for f_i with sufficiently high parameter values.

Theorem 4.1. *Let $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ be any distribution over labelled examples, and let $D \sim \mathcal{D}^n$ be a holdout dataset consisting of n i.i.d. samples from $\mathcal{D}$. Suppose: $n \geq \frac{65\ln(2U/\delta')}{\epsilon^3}$. Let π be the output stream generated by CertificateChecker$(\epsilon, D, (f_1, g_1, h_1), \ldots)$ (Algorithm 2). Then for any possibly adaptive process generating a stream of up to U submissions (f_i, g_i, h_i) as a function of the output stream $\pi \in \{\bot, \top\}^*$, with probability $1 - \delta$ over the randomness of D:*

1. *For every round i such that $\pi_i = \bot$ (the submission is rejected), we have that (g_i, h_i) is not a (μ, Δ) certificate of sub-optimality for f_i for any (μ, Δ) with $\mu \cdot \Delta \geq \epsilon$. And:*
2. *For every round i such that $\pi_i = \top$ (the submission is accepted), we have that (g_i, h_i) is a (μ, Δ)-certificate of sub-optimality for f_i for $\mu \cdot \Delta \geq \epsilon/2$.*

The proof can be found in the Appendix.

We conclude this section by showing that we can use CertificateChecker (Algorithm 2) to run an algorithm FalsifyAndUpdate (Algorithm 3) which persistently maintains a current model, and is able to correctly accept and reject proposed certificates of sub-optimality.

Input: An initial model f_0, a holdout dataset D, Target ϵ, and a stream of submissions (g_1, h_1), (g_2, h_2), ... of length at most U
Let $t \leftarrow 0$
Initialize an instance of CertificateChecker(ϵ, D, U, ...)
while *CertificateChecker has not halted* **do**
  Consider the next submission (g_i, h_i)
  Feed the triple (f_t, g_i, h_i) to CertificateChecker and receive $\pi_i \in \{\bot, \top\}$
  if $\pi_i = \bot$ **then**
    Output: Submission (g_i, h_i) is rejected.
  else
    Let $t \leftarrow t + 1$ and let f_t = ListUpdate$(f_{t-1}, (g_i, h_i))$
    Output: Submission (g_i, h_i) is accepted. The new model is f_t.

Algorithm 3: FalsifyAndUpdate$(\epsilon, D, (f_1, g_1, h_1), \ldots)$: An algorithm that maintains a sequence of models $f_1, \ldots, f_T$ and accepts submissions of proposed certificates of sub-optimality (g_i, h_i) that attempt to falsify the assertion that the current model f_t is not ϵ-Bayes optimal. It either accepts or rejects each submission: if it accepts a submission then it also updates its current model f_t and outputs a new model f_{t+1}.

Theorem 4.2. *Fix any $\epsilon, \delta > 0$. Let $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ be any distribution over labelled examples, and let $D \sim \mathcal{D}^n$ be a holdout dataset consisting of n i.i.d. samples from $\mathcal{D}$. Suppose: $n \geq \frac{65\ln(2U/\delta')}{\epsilon^3}$. Then for any (possibly adaptive) process generating a sequence of at most U submissions $\{(g_i, h_i)\}_{i=1}^U$, with probability at least $1 - \delta$, we have that FalsifyAndUpdate$(\epsilon, D, \cdots)$ satisfies:*

1. *If (g_i, h_i) is rejected, then (g_i, h_i) is not a (μ, Δ)-certificate of sub-optimality for f_t, where f_t is the current model at the time of submission i, for any μ, Δ such that $\mu \cdot \Delta \geq \epsilon$.*
2. *If (g_i, h_i) is accepted, then (g_i, h_i) is a (μ, Δ)-certificate of sub-optimality for f_t, where f_t is the current model at the time of submission i, for some μ, Δ such that $\mu \cdot \Delta \geq \frac{\epsilon}{2}$. Moreover, the new model f_{t+1} output satisfies $L(\mathcal{D}, f_{t+1}, g_i) = L(\mathcal{D}, h_i, g_i)$ and $L(\mathcal{D}, f_{t+1}) \leq L(\mathcal{D}, f_t) - \frac{\epsilon}{2}$.*
3. *FalsifyAndUpdate does not halt before receiving all U submissions.*

The proof can be found in the Appendix.

Remark 4.3. *Note that FalsifyAndUpdate has sample complexity scaling only logarithmically with the total number of submissions U that we can accept, and no dependence on the complexity of the submissions. This means that a relatively small holdout dataset D is sufficient to run an extremely long-running bias bounty program (i.e. handling a number of submissions that is exponentially large in the size of the holdout set) that automatically updates the current model whenever submissions are accepted and bounties are awarded.*

4.1.1 Guaranteeing Groupwise Monotone Improvements. Because the groups g_i can have arbitrary intersections, Algorithm 3 does *not* guarantee that f_{t+1} has error that is lower than that of f_t on all groups that were previously identified. Specifically, let $\mathcal{G}(f_t)$ denote the set of at most $2/\epsilon$ groups g_i that make up the internal nodes of decision list f_t — i.e. the set of groups corresponding to submissions (g_i, h_i) that were previously accepted and incorporated into model f_t. It might be that for some $g_j \in \mathcal{G}(f_t)$, $L(\mathcal{D}, f_{t+1}, g_j) \gg L(\mathcal{D}, f_t, g_j)$. This kind of non-monotonic behavior is extremely undesirable in the context of a bias bug bounty program, because it means that previous instances of sub-optimal behavior on a group g_i that were explicitly identified and corrected for can be undone by future updates.

There is a simple fix, however: whenever a new proposed certificate of sub-optimality (g_i, h_i) for a model f_t is accepted and a new model f_{t+1} is generated, add the proposed certificates (g_j, f_ℓ) to the front of the stream of submissions, for each pair of $g_j \in \mathcal{G}(f_t)$ and $\ell \leq t$. Updates resulting from these submissions (which we call *repairs*) might themselves generate new non-monotonicities, but repeating this process recursively is sufficient to guarantee approximate groupwise monotonicity — and because we know from Theorem 3.4 that the total number of updates T cannot exceed $2/\epsilon$, this process never adds more than $\frac{8}{\epsilon^3}$ submissions to the existing stream, and thus affects the sample complexity bound only by low order terms. The procedure, which we call MonotoneFalsifyAndUpdate, is described as Algorithm 4. Here we state its guarantee:

Theorem 4.4. *Fix any $\epsilon, \delta > 0$. Let $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ be any distribution over labelled examples, and let $D \sim \mathcal{D}^n$ be a holdout dataset consisting of n i.i.d. samples from $\mathcal{D}$. Suppose: $n \geq \frac{65\ln\left(\frac{2(U+\frac{8}{\epsilon^3})}{\delta'}\right)}{\epsilon^3}$. Then for any (possibly adaptive) process generating a sequence of at most U submissions $\{(g_i, h_i)\}_{i=1}^U$, with probability at least $1 - \delta$, we have that MonotoneFalsifyAndUpdate$(\epsilon, D, \cdots)$ satisfies all of the properties proven in Theorem 4.2 for FalsifyAndUpdate, and additionally satisfies the following error monotonicity property. Consider any model f_t that is output, and any group $g_j \in \mathcal{G}(f_t)$. Then: $L(\mathcal{D}, f_t, g_j) \leq \min_{\ell < t} L(\mathcal{D}, f_\ell, g_j) + \frac{\epsilon}{\mu_{\mathcal{D}}(g_j)}$.*

The proof can be found in the appendix.

Algorithm 4 introduces updates of the form (g_j, f_ℓ), where f_ℓ is a decision list previously generated by ListUpdate. We might worry that these updates could produce a very large model, since on such

Input: A model f_0, a holdout dataset D, Target ϵ, and a stream of submissions (g_1, h_1), (g_2, h_2), ... of length at most U
Let $t \leftarrow 0$ and Initialize an instance of CertificateChecker(ϵ, D, $U + \frac{8}{\epsilon^3}$, ...)
while *CertificateChecker has not halted* **do**
  Consider the next submission (g_i, h_i)
  Feed the triple (f_t, g_i, h_i) to CertificateChecker and receive $\pi_i \in \{\perp, \top\}$
  if $\pi_i = \perp$ **then**
    Output: Submission (g_i, h_i) is rejected.
  else
    MonotoneProgress $\leftarrow$ FALSE
    $(g_U, h_U) \leftarrow (g_i, h_i)$
    $t' \leftarrow t$ and $f'_{t'} \leftarrow f_t$
    while *MonotoneProgress = FALSE and CertificateChecker has not halted* **do**
      Let $t' \leftarrow t' + 1$ and let $f'_{t'} = \text{ListUpdate}(f'_{t'-1}, (g_U, h_U))$
      MonotoneProgress $\leftarrow$ TRUE
      for *each pair* $\ell < t, g_j \in \mathcal{G}(f_t)$ **do**
        Feed the triple (f_t, g_j, f_ℓ) to CertificateChecker and receive $\pi_U \in \{\perp, \top\}$
        if $\pi_U = \top$ *(Submission accepted)* **then**
          MonotoneProgress $\leftarrow$ FALSE
          $(g_U, h_U) \leftarrow (g_j, f_\ell)$
    Let $t \leftarrow t + 1$ and let $f_t = f'_{t'}$
    Output: Submission (g_i, h_i) is accepted. The new model is f_t.

Algorithm 4: MonotoneFalsifyAndUpdate($\epsilon, D, (f_1, g_1, h_1), \ldots$): A version of FalsifyAndUpdate that guarantees approximate group-wise error monotonicity.

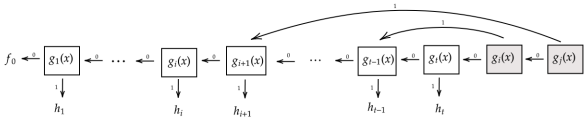

Figure 1: A pointer decision list. The shaded nodes are monotonicity repairs caused by update g_t. In this example, there were two monotonicity repairs, corresponding to certificates of sub-optimality (g_i, f_{t-1}) and (g_j, f_{i+1}). Rather than replicating the models f_{t-1} and f_{i+1}, the updates are implemented using pointers to the prefix of the decision list representing models f_{t-1} and f_{i-1}. Theorem 4.4 guarantees that the entire length of the list (including nodes that implement back-pointers) cannot grow beyond $2/\epsilon$.

an update, the new model f_ℓ at the first leaf of the new decision list entirely replicates some previous decision list f_ℓ. However, note that for any such update, f_ℓ is a *prefix* of f_t. Therefore, rather than replicating f_ℓ, we can introduce a backpointer to level ℓ of our existing decision list, without increasing our model's size. We call such a model a *pointer decision list*, as shown in Figure 1. We call the pointer nodes that are introduced to repair non-monotonicities introduced by previous updates *repair nodes* of our pointer decision list. These are the nodes that are shaded in our illustration in Figure 1.

4.2 Certificates of Bounded Complexity and Algorithmic Optimization

We can also use our ListUpdate method as part of an algorithm for explicitly computing (ϵ, C)-Bayes optimal models from data sampled i.i.d. from the underlying distribution $\mathcal{D}$. First, we must show that if we find certificates of sub-optimality $(g, h) \in C$ on our dataset, that we can be assured that they are certificates of sub-optimality on the underlying distribution. Next, we must describe an algorithmic method for finding (μ, Δ) certificates of sub-optimality that maximize $\mu \cdot \Delta$. We provide such guarantees and algorithmic methods in Appendix C.1.

5 EXPERIMENTS

We next provide an experimental illustration and evaluation of our main algorithm MonotoneFalsifyAndUpdate (Algorithm 4), as well as our optimization approach (Algorithm 5) paired with our reduction to ternary cost sensitive classification. We report findings on a number of different datasets and classification tasks from the recently published Folktables package [9], which provides extensive U.S. census-derived Public Use Microdata Samples (PUMS). These granular and large datasets are well-suited to experimental evaluations of algorithmic fairness, as they include a number of demographic or protected features including race categories, age, disability status, and binary sex categories. We refer to [9] for details of the datasets examined below.

We implemented Algorithm 4 and divided each dataset examined into 80% for training (g, h) pairs as proposed certificates of sub-optimality for the algorithm's current model, and 20% for use by the certificate checker to validate improvements. In order to clearly illustrate the gradual subgroup and overall error improvements, we train a deliberately simple initial model (a decision stump). In the first set of experiments, the sequence of subgroups is 11 different demographic groups given by features common to all of the Folktables datasets. We consider four different datasets (three pictured in Figure 2, a fourth in the appendix) corresponding to four different states and prediction tasks, all for the year 2018 (the most recent available). Table 2 in the Appendix and its caption detail the states and tasks, the total dataset sizes, and the definitions and counts of each demographic subgroup considered. Figure 5 in the Appendix shows a sample fragment and the full feature set for one of the prediction tasks.

In our first experiments, these 11 subgroups g were introduced to Algorithm 4 in some order, and for each a depth 10 decision tree h was trained on just the training data falling into that subgroup. These (g, h) pairs are then given to the certificate checker, and if h improves the holdout set error for g, the improvement is accepted and incorporated into the model, and any needed monotonicity repairs of previously introduced groups are made. Otherwise the pair is rejected and we proceed to the next subgroup.

Each of the plots in Figure 2 shows the results of one experiment of this type on test data. Since the subgroups are introduced in some sequential order (given in the legends), we plot the subgroup errors using dashed lines up until the round at which the group is introduced, and solid lines thereafter. Subgroups whose trained decision tree was rejected by the certificate checker are not shown

at all. In Figure 3, a depiction of the decision list generated by the employment task shown in the leftmost plot in Figure 2 is given.

As promised by Theorem 4.4, the overall and subgroup test errors are all monotonically non-increasing after the introduction of the subgroup, and generally enjoy significant improvement as the algorithm proceeds. Note that even prior to the introduction of a subgroup (dashed lines), the test errors are generally non-increasing. Although this is not guaranteed by the theory, it is not necessarily surprising, since before groups are introduced they often start from a high baseline error. However, this is not always the case — for example, on the Oregon income prediction task (middle panel) the test error for the Native subgroup (light green) increases significantly with the introduction of the subgroups for young and middle-aged at rounds 3 and 4 before it has been introduced.

While not visible in Figure 2, we note that the order the demographic groups are introduced can have qualitative effects on the behavior of the algorithm, including which subgroup improvements are accepted, as shown in the first two rows of Figure 6 in the Appendix. However, regardless of the ordering, the final errors across all groups end up comparable, with the differences generally being in the second or third decimal place, as shown by Table 3 in the Appendix.

Examining the actual pointer decision list produced in one of these experiments is also enlightening — see Figure 3. We note a couple of things. First, observing the repairs for the group "Young", we see that via the sequence of repairs, any "young" individual will always be classified by the model h_2 introduced at round 2 together with the group "young". This is in contrast to the other groups, whose members are split across a number of different models by the pointer decision list. This indicates that being in the group "young" is more salient than being in any of the racial or gender groups in this task from the point of view of accurate classification. Second, we note that unlike for the group "young", repairs for other groups do not necessarily point back to the round at which they were introduced. For example, the right-most repairs for "Native American", "Black or AA", and "Some Other Race" point back to round 6, at which the group "Old" was introduced. This indicates that the pointer decision list at round 6 is more accurate for each of these groups then the decision trees h_4 h_3 and h_1 that were introduced along with these groups, despite the fact that these decision trees were trained *only* on examples from their corresponding groups.

The experiments described so far apply our framework and algorithm to a setting in which the groups considered are simple demographic groups, introduced in a fixed ordering. We finish with a brief experimental investigation of the approach described in Section 4.2and Appendix C.1, in which the discovery of updates is posed as an optimization problem. Specifically, we implemented the ternary CSC approach of Section C.1.1. In our implementation, following the approach taken in [1, 25], we learn a ternary classifier by first learning two separate depth 7 decision tree regression functions for the costs of predicting 0 and 1, while the cost of predicting "?" is always 0 as per Section C.1.1. Our final ternary classifier chooses the prediction that minimizes the costs predicted by the regression functions.

We applied the CSC approach to the ACS income task using Folktables datasets for thirteen different states in the year 2016. One interesting empirical phenomenon is the speed with which this approach converges, generally stopping after two to five rounds and not finding any further subgroup improvements to the current model. In Figure 4, we compare these subgroup test errors to those obtained by the simple sequential introduction of those subgroups discussed earlier. In the left panel, the x axis represents each of the 11 subgroups, and the color coding of the points represents which of the 13 state datasets is considered. The y values of the points measure the signed difference of the test errors of the CSC approach and the simple sequential approach (using a fixed ordering), with positive values being a win for sequential and negative values a win for CSC. The right panel visualizes the same data, but now grouped by states on the x axis and with color coding for the groups.

The overarching message of the figure is that though CSC does not directly consider these subgroups, and instead optimizes for the complex subgroup giving the best weighted improvement to the overall error, it is nevertheless quite competitive on the simple subgroups, with the mass of points above and below $y = 0$ being approximately the same. Combined with the very rapid convergence of CSC, we can thus view it as an approach that seeks rich subgroup optimization while providing strong basic demographic group performance "for free". Further, combining the two approaches (e.g. running the CSC approach then sequentially introducing the basic groups) should only yield further improvement.

6 A PRELIMINARY DEPLOYMENT

We conclude by informally reporting findings from a preliminary deployment of our bias bounty framework to a group of 83 students divided into 36 teams in an undergraduate class on ethical algorithm design at the University of Pennsylvania in the Spring of 2022. In order to expedite this deployment in a limited time window, we made a few departures from the formal framework. First, we did not implement the adaptive data analysis mechanisms to prevent overfitting to the holdout set, but instead reported errors on the holdout set regardless of whether a proposed update is accepted or not. Second, in order to avoid the systems overhead of implementing a full client-server architecture and its potential security mechanisms, each team was instead given a self-contained notebook implementing both tools for finding (g, h) pairs and the functionality for accepting updates and incorporating them into a revised decision list. Each notebook contained access to one of two Folktables datasets and classification tasks, which were divided roughly evenly between the teams. Students were told there were absolutely no constraints on the approaches they could take to finding improving g, h pairs — they could do so by human intuition, manual exploratory data analysis, or they could entirely automate the process using ML or other techniques. They were generally encouraged to find as many accepted updates as they could during the duration of the deployment.

Despite the very preliminary nature of this exercise, a number of informal and high-level observations can nevertheless be made. First, despite wide variations in the quantitative backgrounds of the students (the only prerequisite to the course being exposure to introductory programming), virtually all of the most successful teams adopted hybrid approaches that involved some combination of manual or data-driven discovery of groups along with more automated means of finding g, followed by training of h on g.

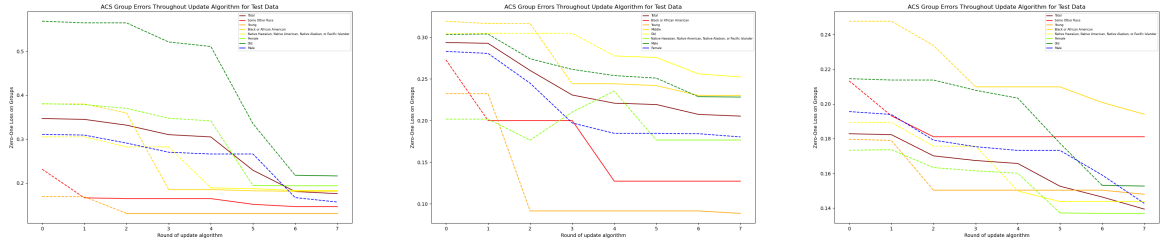

Figure 2: Results of an implementation of algorithm Algorithm 4 on a number of different tasks and U.S. state datasets from the Folktables package. Left: employment task for NY, middle: income task for OR, and right: coverage task for TX. See text for discussion.

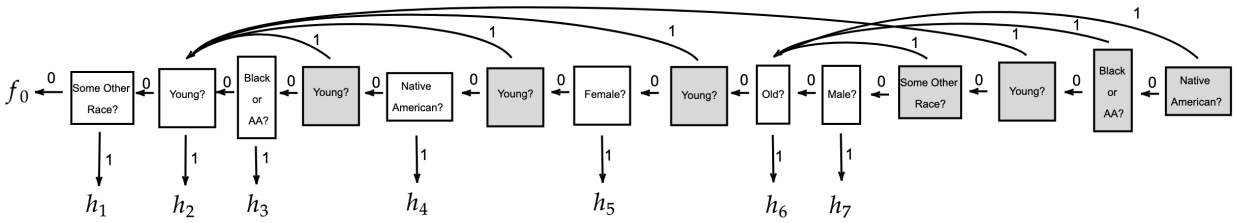

Figure 3: The pointer decision list generated by an implementation of algorithm MonotoneFalsifyAndUpdate (Algorithm 4) on an ACS employment task for New York State. Repair nodes are in gray.

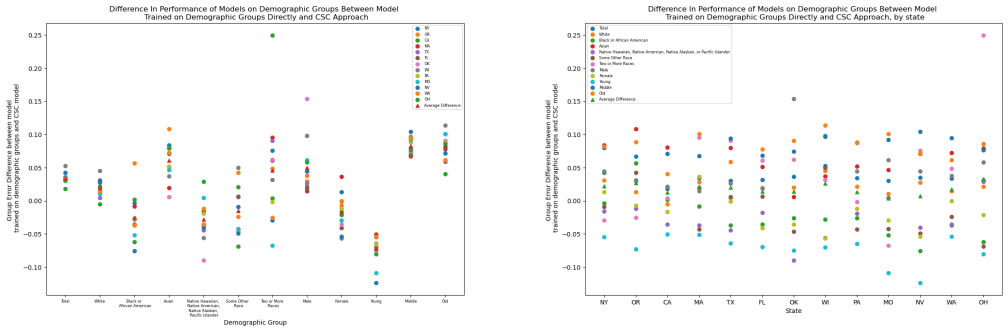

Figure 4: Comparison of basic demographic subgroup test errors for sequential and CSC approaches. See text for details.

For instance, many groups automatically scanned through the feature space by column, or combinations of a small number of columns, to identify which values for each feature the current model performed poorly on, and then trained models on these subsets of the data. Note this has an advantage of interpretability over the groups generated by the algorithmic approaches developed in Section 4.2, as the groups are always explicit functions of categorical values over a small number of features (e.g. "Native American Women" or "Young Men"). However, this method is also restricted to tabular data that has a nicely structured feature space: if instead students were tasked with bias hunting over a collection of images, given only the input pixel matrix, such a strategy would not be sufficient.

Another notable feature of the exercise is that the purely algorithmic approaches appeared insufficient: groups who employed a combination of manual data analytics and algorithmic methods were able to bring errors down better than groups who employed purely algorithmic approaches similar to those in Section 4.2. This emphasizes the fact that there is really an advantage to doing a true deployment of such a system within a community, as opposed to attempting to automatically identify all issues.

Round	Group Description	Group Error Pre Update	Group Error Post Update	Group Error Improvement	Group Weight	Overall Model Error
0					1.0000	0.2796
1	male sex (M)	0.2866	0.1734	0.1133	0.5238	0.2203
2	female sex (M)	0.2719	0.1658	0.1061	0.4762	0.1698
3	ages 17 to 24 and non-white (M)	0.0447	0.0414	0.0032	0.0526	0.1696
4	over the age of 30 working without pay in a family business (A)	0.2258	0.1452	0.0806	0.0021	0.1694
5	self-employed in own not incorporated business, professional practice, or farm (A)	0.2255	0.2205	0.0050	0.0884	0.1690
6	over the age of 62 (retired) (M)	0.2322	0.2291	0.0031	0.1193	0.1686
7	First-line supervisors of office and administrative support workers (A)	0.3073	0.2661	0.0413	0.0074	0.1683
8	real estate brokers and sales agents (A)	0.3154	0.2905	0.0249	0.0082	0.1681
9	first-line supervisors of retail sales workers (A)	0.2583	0.2494	0.0088	0.0154	0.1680
10	office clerks, general (A)	0.2135	0.1842	0.0292	0.0117	0.1676
11	paint, coating, and adhesive manufacturing (A)	0.1713	0.1657	0.0055	0.0185	0.1675
12	those born in California, Mexico, or Southeast Asia generally working as medical technicians (M)	0.2126	0.2032	0.0094	0.0255	0.1671
13	accountants/Auditors who work 40 hour work weeks (A)	0.2883	0.2793	0.0090	0.0076	0.1671
14	female sex secretaries and administrative assistants, except legal, medical, and executive (A)	0.2586	0.2533	0.0053	0.0129	0.1670
15	machine learning classifier over errors on current model (A)	0.5600	0.3200	0.2400	0.0009	0.1668

Table 1: Details of the efforts of one particular team, see text for discussion.

Some detail on the efforts of one of the most successful teams is provided in Table 1, where for each successive (g, h) pair accepted, we show an English description of the group, and indication of whether it was discovered by (A)utomated or (M)anual techniques, as well as the pre- and post-update group errors, the group weight, and the overall model error after each update. A number of observations are in order. First, in general the earlier groups discovered are simpler and often manually discovered, whereas the later ones tend to be more complex and automated. Indeed, the final group accepted is not even defined by features of the dataset at all, but rather is trained on the errors of the overall model. Second, echoing the increasing complexity and specificity of successive groups, we see that there are generally diminishing marginal returns in both the group weights and the reduction of overall model error. We might expect both increasing group complexity and diminishing returns to be general features of bias bounty events conducted under our particular algorithmic framework.

ACKNOWLEDGMENTS

We give warm thanks to Peter Hallinan, Pietro Perona and Alex Tolbert for many enlightening conversations at an early stage of this work, and to Bri Cervantes and Declan Harrison for providing details of their class bias bounty efforts, and permission to include them here.

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] Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. 2021. Algorithmic stability for adaptive data analysis. *SIAM J. Comput.* 0 (2021), STOC16–377.

[3] Avrim Blum and Moritz Hardt. 2015. The ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning*. PMLR, 1006–1014.

[4] Avrim Blum and Thodoris Lykouris. 2020. Advancing Subgroup Fairness via Sleeping Experts. In *Innovations in Theoretical Computer Science Conference (ITCS)*, Vol. 11.

[5] Avrim Blum and Kevin Stangl. 2019. Recovering from biased data: Can fairness constraints improve accuracy? *arXiv preprint arXiv:1912.01094* (2019).

[6] Rumman Chowdhury and Jutta Williams. 2021. Introducing Twitter's First Algorithmic Bias Bounty Challenge. https://blog.twitter.com/engineering/en_us/topics/insights/2021/algorithmic-bias-bounty-challenge.

[7] Emily Diana, Wesley Gill, Ira Globus-Harris, Michael Kearns, Aaron Roth, and Saeed Sharifi-Malvajerdi. 2021. Lexicographically Fair Learning: Algorithms and Generalization. In *2nd Symposium on Foundations of Responsible Computing (FORC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[8] Emily Diana, Wesley Gill, Michael Kearns, Krishnaram Kenthapadi, and Aaron Roth. 2021. Minimax group fairness: Algorithms and experiments. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 66–76.

[9] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. 2021. Retiring Adult: New Datasets For Fair Machine Learning. *arXiv preprint arXiv:2108.04884* (2021).

[10] Sanghamitra Dutta, Dennis Wei, Hazar Yueksel, Pin-Yu Chen, Sijia Liu, and Kush Varshney. 2020. Is there a trade-off between fairness and accuracy? a perspective using mismatched hypothesis testing. In *International Conference on Machine Learning*. PMLR, 2803–2813.

[11] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. 2015. Generalization in Adaptive Data Analysis and Holdout Reuse. In *NIPS*.

[12] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. 2015. The reusable holdout: Preserving validity in adaptive data analysis. *Science* 349, 6248 (2015), 636–638.

[13] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. 2015. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 117–126.

[14] 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.

[15] Cynthia Dwork, Michael P Kim, Omer Reingold, Guy N Rothblum, and Gal Yona. 2021. Outcome indistinguishability. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 1095–1108.

[16] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM* 64, 12 (2021), 86–92.

[17] Varun Gupta, Christopher Jung, Georgy Noarov, Mallesh M Pai, and Aaron Roth. 2021. Online multivalid learning: Means, moments, and prediction intervals. *arXiv preprint arXiv:2101.01739* (2021).

[18] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. *Advances in neural information processing systems* 29 (2016), 3315–3323.

[19] Ursula Hébert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. 2018. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*. PMLR, 1939–1948.

[20] Christina Ilvento. 2020. Metric Learning for Individual Fairness. In *1st Symposium on Foundations of Responsible Computing (FORC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[21] Christopher Jung, Michael Kearns, Seth Neel, Aaron Roth, Logan Stapleton, and Zhiwei Steven Wu. 2019. An Algorithmic Framework for Fairness Elicitation. *arXiv e-prints* (2019), arXiv–1905.

[22] Christopher Jung, Changhwa Lee, Mallesh M Pai, Aaron Roth, and Rakesh Vohra. 2021. Moment Multicalibration for Uncertainty Estimation. In *Conference on Learning Theory*. PMLR.

[23] Christopher Jung, Katrina Ligett, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Moshe Shenfeld. 2020. A New Analysis of Differential Privacy's Generalization Guarantees. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[24] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2018. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International Conference on Machine Learning*. PMLR, 2564–2572.

[25] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2019. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 100–109.

[26] Michael P Kim, Amirata Ghorbani, and James Zou. 2019. Multiaccuracy: Black-box post-processing for fairness in classification. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 247–254.

[27] Natalia Martinez, Martin Bertran, and Guillermo Sapiro. 2020. Minimax pareto fairness: A multi objective perspective. In *International Conference on Machine Learning*. PMLR, 6755–6764.

[28] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*. 220–229.

[29] Georgy Noarov, Mallesh Pai, and Aaron Roth. 2021. Online Multiobjective Minimax Optimization and Applications. *arXiv preprint arXiv:2108.03837* (2021).

[30] Amit Elazari Bar On. 2018. We Need Bug Bounties for Bad Algorithms. https://www.vice.com/en/article/8xkyj3/we-need-bug-bounties-for-bad-algorithms.

[31] Guy N Rothblum and Gal Yona. 2021. Multi-group Agnostic PAC Learnability. *arXiv preprint arXiv:2105.09989* (2021).

[32] Christopher Tosh and Daniel Hsu. 2021. Simple and near-optimal algorithms for hidden stratification and multi-group learning. *arXiv preprint arXiv:2112.12181* (2021).

[33] Gal Yona and Guy Rothblum. 2018. Probably approximately metric-fair learning. In *International Conference on Machine Learning*. PMLR, 5680–5688.

[34] 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.

A DESIDERATA FOR A BIAS BOUNTY PROGRAM

Here we lay out a number of desiderata for a bias bounty program that would be run continuously and at scale, and how our method addresses these needs.

(1) The program should be able to accept a very large number of submissions on an ongoing basis. As a result, there should be a category of bounties for which the submissions should be able to be evaluated automatically. Our method suggests the following category: submissions that identify a group of non-negligible size such on which our existing model has performance that is worse than optimal by some non-negligible amount.
(2) Since the program allows open participation and judges submissions automatically, it should be prepared for adversarial submissions, and be able to distinguish between genuine problems discovered on the underlying population or distribution, from cherrypicked examples designed to overfit the competition dataset. Our framework suggests a natural solution to this problem: bounty hunters submit a model describing a group g on which performance is sub-optimal, and a model h which demonstrates the sub-optimality of our existing model on g. Since g and h are themselves models, they can be evaluated on a holdout set to prevent overfitting, and we can use techniques from adaptive data analysis to safely re-use the same holdout set across many submissions [11–13].
(3) For bounties it awards automatically, it should be able to correct the problem identified automatically. Our framework naturally does this, given that bounty hunters submit pairs (g, h), which are the objects we need to correct our model.
(4) The process should be convergent in the sense that there should be no way to produce a sequence of submissions that guarantees the submitter an unlimited amount of money. Said another way, if we award bounties for identifying legitimate problems that we then go on to "fix", our fixes should actually make progress in some quantifiable way. The convergence analysis of our framework establishes this.
(5) The model that results from the automatic patches resulting from the submitted bounties should be "simple" in the sense that the patching operation should itself be computationally easy (since it may need to be repeated many times), and evaluating the final model should also be computationally easy. The "updates" in our framework are elementary, and produces a simple object (a decision list composed of the models submitted) that is easy to evaluate.
(6) Awarding a bounty (and correcting the corresponding problem) should not decrease the performance for any other group: fixing one problem should not introduce others. The monotone improvement property of our method satisfies this.
(7) We should impose as few burdens on the "bounty hunters" as possible. Here, it might seem that our framework falls short because it places two kinds of burdens on "bounty hunters": first, it requires that they submit models g that identify the group on which the current model performs poorly, rather than just identifying subsets of examples on which the model performs poorly. Second, it requires that they submit models h that demonstrate improvements on these groups — and perhaps the burden of improving on a group g should be left to the organization deploying the model in the first place! But we note that these requirements are solving real problems that would otherwise arise:
   (a) As noted above, if bounty hunters simply submitted examples on which the current model performed poorly, rather than a model g identifying such examples, then we would be unable to distinguish cherrypicked examples from real distributional problems.
   (b) If we did not require that bounty hunters submit a model h that demonstrates improved performance on the group g they submitted, then we would be subject to the following difficulty: it might be, for example, that the group g that the bounty hunter identifies (correctly) as having high error corresponds to a group like "blurry images" on which improved performance is impossible. Problems of this sort cannot be fixed, and do not correspond to deviations from Bayes optimality. Requiring bounty hunters to submit a model h that demonstrates improved performance disambiguates "blurry images" problems from real deviations from Bayes optimality.

   Of course, in a real system, we should strive to reduce the burden on the bounty hunter as much as possible. So, for example, we might provide an interface that allows the bounty hunter to identify a collection of examples on which the current model performs poorly, and then attempts to automatically train a model g to capture examples like those identified by the bounty hunter. Similarly, given a group g, we could provide an interface that automatically attempts to use standard methods to train a good model h on examples from g. Interfaces like this could make the bounty program accessible to a wider audience, without requiring any machine learning expertise.

B PROOFS FROM SECTION 3

Theorem B.1. *Fix any $\epsilon > 0$, and any collection of (group,model) pairs C. There exists a (μ, Δ)-certificate of sub-optimality $(g, h) \in C$ for f if and only if f is not (ϵ, C)-Bayes optimal for $\epsilon < \mu\Delta$.*

Proof. We need to prove two directions. First, we will assume that f is (ϵ, C)-Bayes optimal, and show that in this case there do not exist any pairs $(g, h) \in C$ such that (g, h) form a (μ, Δ)-certificate of sub-optimality with $\mu \cdot \Delta > \epsilon$. Fix a pair $(g, h) \in C$. Without loss of generality, we can take $\Delta = L(\mathcal{D}, f, g) - L(\mathcal{D}, h, g)$ (and if $\Delta \leq 0$ we are done, so we can also assume that $\Delta > 0$). Since f is (ϵ, C)-Bayes optimal, by definition we have that:

$$\Delta = L(\mathcal{D}, f, g) - L(\mathcal{D}, h, g) \leq \frac{\epsilon}{\mu_g(\mathcal{D})}$$

Solving, we get that $\Delta \cdot \mu_g \leq \epsilon$ as desired.

Next, we prove the other direction: We assume that there exists a pair $(g, h) \in C$ that form an (μ, Δ)-certificate of sub-optimality, and show that f is not (ϵ, C)-Bayes optimal for any $\epsilon < \mu \cdot \Delta$. Without loss of generality we can take $\mu = \mu_g(\mathcal{D})$ and conclude:

$$L(\mathcal{D}, f, g) - L(\mathcal{D}, h, g) \geq \Delta = \frac{\mu \cdot \Delta}{\mu_g(\mathcal{D})} \geq \frac{\epsilon}{\mu_g(\mathcal{D})}$$

which falsifies (ϵ, C)-Bayes optimality for any $\epsilon < \mu \cdot \Delta$ as desired. □

Theorem B.2. *Algorithm 1 (ListUpdate) has the following properties. If (g_{t+1}, h_{t+1}) form a (μ, Δ)-certificate of sub-optimality for f_t, and $f_{t+1} = ListUpdate(f_t, (g_{t+1}, h_{t+1}))$ then:*

1. *The new model matches the performance of h_{t+1} on group g_{t+1}: $L(\mathcal{D}, f_{t+1}, g_{t+1}) = L(\mathcal{D}, h_{t+1}, g_{t+1})$, and*
2. *The overall performance of the model is improved by at least $\mu \cdot \Delta$: $L(\mathcal{D}, f_{t+1}) \leq L(\mathcal{D}, f_t) - \mu \cdot \Delta$.*

Proof. It is immediate from the definition of f_{t+1} that $L(\mathcal{D}, f_{t+1}, g_{t+1}) = L(\mathcal{D}, h_{t+1}, g_{t+1})$, since for any x such that $g_{t+1}(x) = 1$, $f_{t+1}(x) = h_{t+1}(x)$. It remains to verify the 2nd condition. Because we also have that for every x such that $g_{t+1}(x) = 0$, $f_{t+1}(x) = f_t(x)$, we can calculate:

$$\begin{aligned}
L(\mathcal{D}, f_{t+1}) &= \Pr_{\mathcal{D}}[g_{t+1}(x) = 0] \cdot \mathop{\mathbb{E}}_{\mathcal{D}}[\ell(f_{t+1}(x), y) | g_{t+1}(x) = 0] + \Pr_{\mathcal{D}}[g_{t+1}(x) = 1] \cdot \mathop{\mathbb{E}}_{\mathcal{D}}[\ell(f_{t+1}(x), y) | g_{t+1}(x) = 1] \\
&= \Pr_{\mathcal{D}}[g_{t+1}(x) = 0] \cdot \mathop{\mathbb{E}}_{\mathcal{D}}[\ell(f_t(x), y) | g_{t+1}(x) = 0] + \Pr_{\mathcal{D}}[g_{t+1}(x) = 1] \cdot \mathop{\mathbb{E}}_{\mathcal{D}}[\ell(h_{t+1}(x), y) | g_{t+1}(x) = 1] \\
&\leq \Pr_{\mathcal{D}}[g_{t+1}(x) = 0] \cdot \mathop{\mathbb{E}}_{\mathcal{D}}[\ell(f_t(x), y) | g_{t+1}(x) = 0] + \Pr_{\mathcal{D}}[g_{t+1}(x) = 1] \left(\mathop{\mathbb{E}}_{\mathcal{D}}[\ell(f_t(x), y) | g_{t+1}(x) = 1] - \Delta \right) \\
&\leq L(\mathcal{D}, f_t) - \mu\Delta
\end{aligned}$$

□

Theorem B.3. *Fix any $\epsilon > 0$. For any initial model f_0 with loss $L(\mathcal{D}, f_0) = \ell_0$ and any sequence of models $f_1, \ldots, f_T$, such that $f_i = ListUpdate(f_{i-1}, (g_i, h_i))$ and each pair (g_i, h_i) forms a (μ, Δ)-certificate of suboptimality for f_{i-1} for some μ, Δ such that $\mu \cdot \Delta \geq \epsilon$, the length of the update sequence must be at most $T \leq \frac{\ell_0}{\epsilon} \leq \frac{1}{\epsilon}$.*

Proof. By assumption $L(\mathcal{D}, f_0) = \ell_0$. Because each (g_i, h_i) is a (μ, Δ)-certificate of suboptimality of f_{i-1} with $\mu \cdot \Delta \geq \epsilon$, we know from Theorem 3.3 that for each i, $L(\mathcal{D}, f_i) \leq L(\mathcal{D}, f_{i-1}) - \epsilon$. Hence $L(\mathcal{D}, f_T) \leq \ell_0 - T\epsilon$. But loss is non-negative: $L(\mathcal{D}, f_T) \geq 0$. Thus it must be that $T \leq \frac{\ell_0}{\epsilon}$ as desired. □

C MATERIAL FROM SECTION 4

Theorem C.1. *Let $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ be any distribution over labelled examples, and let $D \sim \mathcal{D}^n$ be a holdout dataset consisting of n i.i.d. samples from $\mathcal{D}$. Suppose: $n \geq \frac{65 \ln(2U/\delta')}{\epsilon^3}$. Let π be the output stream generated by CertificateChecker$(\epsilon, D, (f_1, g_1, h_1), \ldots)$ (Algorithm 2). Then for any possibly adaptive process generating a stream of up to U submissions (f_i, g_i, h_i) as a function of the output stream $\pi \in \{\perp, \top\}^*$, with probability $1 - \delta$ over the randomness of D:*

1. *For every round i such that $\pi_i = \perp$ (the submission is rejected), we have that (g_i, h_i) is not a (μ, Δ) certificate of sub-optimality for f_i for any (μ, Δ) with $\mu \cdot \Delta \geq \epsilon$. And:*
2. *For every round i such that $\pi_i = \top$ (the submission is accepted), we have that (g_i, h_i) is a (μ, Δ)-certificate of sub-optimality for f_i for $\mu \cdot \Delta \geq \epsilon/2$.*

Proof. We first consider any *fixed* triple of functions $f_i : \mathcal{X} \to \mathcal{Y}, h_i : \mathcal{X} \to, \mathcal{Y}, g_i : \mathcal{X} \to \{0, 1\}$. Observe that we can write:

$$\begin{aligned}
\mu_i \cdot \Delta_i &= \mu_D(g_i) \cdot (L(D, f_i, g_i) - L(D, h_i, g_i)) \\
&= \frac{\sum_{(x,y) \in D} \mathbb{1}[g_i(x) = 1]}{n} \cdot \frac{\sum_{(x,y) \in D} \mathbb{1}[g_i(x) = 1] \cdot (\ell(f_i(x), y) - \ell(h_i(x), y))}{\sum_{(x,y) \in D} \mathbb{1}[g_i(x) = 1]} \\
&= \frac{1}{n} \sum_{(x,y) \in D} \mathbb{1}[g_i(x) = 1] \cdot (\ell(f_i(x), y) - \ell(h_i(x), y))
\end{aligned}$$

Since each $(x, y) \in D$ is drawn independently from $\mathcal{D}$, each term in the sum $\mathbb{1}[g_i(x) = 1] \cdot (\ell(f_i(x), y) - \ell(h_i(x), y))$ is an independent random variable taking value in the range $[-1, 1]$. Thus $\mu_i \cdot \Delta_i$ is the average of n independent bounded random variables and we can apply a Chernoff bound to conclude that for any value of $\delta' > 0$:

$$\Pr_{D \sim \mathcal{D}^n} \left[|\mu_i \cdot \Delta_i - \mu_{\mathcal{D}}(g_i) \cdot (L(\mathcal{D}, f_i, g_i) - L(\mathcal{D}, h_i, g_i))| \geq \sqrt{\frac{2 \ln(2/\delta')}{n}} \right] \leq \delta'$$

Solving for n we have that with probability $1-\delta'$, we have $|\mu_i \cdot \Delta_i - \mu_{\mathcal{D}}(g_i) \cdot (L(\mathcal{D}, f_i, g_i) - L(\mathcal{D}, h_i, g_i))| \leq \frac{\epsilon}{4}$ if:

$$n \geq \frac{32 \ln(2/\delta')}{\epsilon^2}$$

This analysis was for a fixed triple of functions (f_i, h_i, g_i), but these triples can be chosen arbitrarily as a function of the transcript π. We therefore need to count how many transcripts π might arise. By construction, π has length at most U and has at most $2/\epsilon$ indices such that $\pi_i = \top$. Thus the number of transcripts π that can arise is at most: $\binom{U}{\frac{2}{\epsilon}} \cdot 2^{2/\epsilon} \leq U^{2/\epsilon}$, and each transcript results in some sequence of U triples (f_i, h_i, g_i). Thus for any mechanism for generating triples from transcript prefixes, there are at most $U^{2/\epsilon+1}$ triples that can ever arise. We can complete the proof by union bounding over this set. Taking $\delta' = \frac{\delta}{U^{2/\epsilon+1}}$ and plugging into our Chernoff bound above, we obtain that with probability $1-\delta$ over the choice of D, for any method of generating a sequence of U triples $\{(f_i, h_i, g_i)\}_{i=1}^{U}$ from transcripts π, we have that: $\max_i |\mu_i \cdot \Delta_i - \mu_{\mathcal{D}}(g_i) \cdot (L(\mathcal{D}, f_i, g_i) - L(\mathcal{D}, h_i, g_i))| \leq \frac{\epsilon}{4}$ so long as:

$$n \geq \frac{32(\frac{2}{\epsilon}+1)\ln(2U/\delta')}{\epsilon^2} \geq \frac{65\ln(2U/\delta')}{\epsilon^3}$$

Finally, note that whenever this event obtains, the conclusions of the theorem hold, because we have that $\pi_i = \top$ exactly when $\mu_i \cdot \Delta_i \geq \frac{3\epsilon}{4}$. In this case, $\mu_{\mathcal{D}}(g_i) \cdot (L(\mathcal{D}, f_i, g_i) - L(\mathcal{D}, h_i, g_i)) \geq \frac{3\epsilon}{4} - \frac{\epsilon}{4} = \frac{\epsilon}{2}$ as desired. Similarly, whenever $\pi_i = \perp$, we have that $\mu_{\mathcal{D}}(g_i) \cdot (L(\mathcal{D}, f_i, g_i) - L(\mathcal{D}, h_i, g_i)) \leq \frac{3\epsilon}{4} + \frac{\epsilon}{4} = \epsilon$ as desired. □

Theorem C.2. *Fix any $\epsilon, \delta > 0$. Let $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ be any distribution over labelled examples, and let $D \sim \mathcal{D}^n$ be a holdout dataset consisting of n i.i.d. samples from $\mathcal{D}$. Suppose: $n \geq \frac{65\ln(2U/\delta')}{\epsilon^3}$. Then for any (possibly adaptive) process generating a sequence of at most U submissions $\{(g_i, h_i)\}_{i=1}^{U}$, with probability at least $1-\delta$, we have that FalsifyAndUpdate$(\epsilon, D, \cdots)$ satisfies:*

1. *If (g_i, h_i) is rejected, then (g_i, h_i) is not a (μ, Δ)-certificate of sub-optimality for f_t, where f_t is the current model at the time of submission i, for any μ, Δ such that $\mu \cdot \Delta \geq \epsilon$.*
2. *If (g_i, h_i) is accepted, then (g_i, h_i) is a (μ, Δ)-certificate of sub-optimality for f_t, where f_t is the current model at the time of submission i, for some μ, Δ such that $\mu \cdot \Delta \geq \frac{\epsilon}{2}$. Moreover, the new model f_{t+1} output satisfies $L(\mathcal{D}, f_{t+1}, g_i) = L(\mathcal{D}, h_i, g_i)$ and $L(\mathcal{D}, f_{t+1}) \leq L(\mathcal{D}, f_t) - \frac{\epsilon}{2}$.*
3. *FalsifyAndUpdate does not halt before receiving all U submissions.*

Proof. This theorem follows straightforwardly from the properties of Algorithm 1 and Algorithm 2. From Theorem 4.1, we have that with probability $1-\delta$, every submission accepted by CertificateChecker (and hence by FalsifyandUpdate) is a (μ, Δ)-certificate of sub-optimality for f_t with $\mu \cdot \Delta \geq \epsilon/2$ and every submission rejected is not a (μ, Δ)-certificate of sub-optimality for any μ, Δ with $\mu \cdot \Delta \geq \epsilon$.

Whenever this event obtains, then for every call that FalsifyAndUpdate makes to ListUpdate$(f_{t-1}, (g_i, h_i))$ is such that (g_i, h_i) is a (μ, Δ)-certificate of sub-optimality for f_{t-1} for $\mu \cdot \Delta \geq \epsilon/2$. Therefore by Theorem 3.3, we have that $L(\mathcal{D}, f_{t+1}, g_i) = L(\mathcal{D}, h_i, g_i)$ and $L(\mathcal{D}, f_{t+1}) \leq L(\mathcal{D}, f_t) - \frac{\epsilon}{2}$. Finally, by Theorem 3.4, if each invocation of the iteration $f_t = \text{ListUpdate}(f_{t-1}, (g_i, h_i))$ is such that (g_i, h_i) is a (μ, Δ)-certificate of sub-optimality for f_{t-1} with $\mu \cdot \Delta \geq \epsilon/2$, then there can be at most $2/\epsilon$ such invocations. Since FalsifyAndUpdate makes one such invocation for every submission that is accepted, this means there can be at most $2/\epsilon$ submissions accepted in total. But CertificateChecker has only two halting conditions: it halts when either more than $2/\epsilon$ submissions are accepted, or when U submissions have been made in total. Because with probability at least $1-\delta$ no more than $2/\epsilon$ submissions are accepted, it must be that with probability $1-\delta$, FalsifyAndUpdate does not halt until all U submissions have been received. □

Theorem C.3. *Fix any $\epsilon, \delta > 0$. Let $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ be any distribution over labelled examples, and let $D \sim \mathcal{D}^n$ be a holdout dataset consisting of n i.i.d. samples from $\mathcal{D}$. Suppose: $n \geq \frac{65\ln\left(\frac{2(U+\frac{8}{\epsilon^3})}{\delta'}\right)}{\epsilon^3}$. Then for any (possibly adaptive) process generating a sequence of at most U submissions $\{(g_i, h_i)\}_{i=1}^{U}$, with probability at least $1-\delta$, we have that MonotoneFalsifyAndUpdate$(\epsilon, D, \cdots)$ satisfies all of the properties proven in Theorem 4.2 for FalsifyAndUpdate, and additionally satisfies the following error monotonicity property. Consider any model f_t that is output, and any group $g_j \in \mathcal{G}(f_t)$. Then: $L(\mathcal{D}, f_t, g_j) \leq \min_{\ell < t} L(\mathcal{D}, f_\ell, g_j) + \frac{\epsilon}{\mu_{\mathcal{D}}(g_j)}$.*

Proof. The proof that MonotoneFalsifyAndUpdate satisfies the first two conclusions of Theorem 4.2:

1. If (g_i, h_i) is rejected, then (g_i, h_i) is not a (μ, Δ)-certificate of sub-optimality for f_t, where f_t is the current model at the time of submission i, for any μ, Δ such that $\mu \cdot \Delta \geq \epsilon$.
2. If (g_i, h_i) is accepted, then (g_i, h_i) is a (μ, Δ)-certificate of sub-optimality for f_t, where f_t is the current model at the time of submission i, for some μ, Δ such that $\mu \cdot \Delta \geq \frac{\epsilon}{2}$. Moreover, the new model f_{t+1} output satisfies $L(\mathcal{D}, f_{t+1}) \leq L(\mathcal{D}, f_t) - \frac{\epsilon}{2}$.

are identical and we do not repeat them here. We must show that with probability $1-\delta$, CertificateChecker (and hence MonotoneFalsifyAndUpdate) does not halt before processing all U submissions. Note that MonotoneFalsifyAndUpdate initializes an instance of CertificateChecker that will not halt before receiving $U + \frac{8}{\epsilon^3}$ many submissions. Thus it remains to verify that our algorithm does not produce more than $8/\epsilon^3$ many submissions to CertificateChecker in its monotonicity update process. But this will be the case, because by Theorem 3.4, $t \leq \frac{2}{\epsilon}$, and so

after each call to ListUpdate, we generate at most $4/\epsilon^2$ many submissions to CertificateChecker. Since there can be at most $2/\epsilon$ such calls to ListUpdate, the claim follows.

To see that the monotonicity property holds, assume for sake of contradiction that it does not — i.e. that there is a model f_t, a group $g_j \in \mathcal{G}(f_t)$, and a model f_ℓ with $\ell < t$ such that:

$$L(\mathcal{D}, f_t, g_j) > L(\mathcal{D}, f_\ell, g_j) + \frac{\epsilon}{\mu_{\mathcal{D}}(g_j)}$$

In this case, the pair (g_j, f_ℓ) would form a (μ, Δ)-certificate of sub-optimality for f_t with $\mu \cdot \Delta \geq \epsilon$. But if $L(\mathcal{D}, f_t, g_j) > L(\mathcal{D}, f_\ell, g_j) + \frac{\epsilon}{\mu_{\mathcal{D}}(g_j)}$, then this certificate must have been rejected, which we have already established is an event that occurs with probability at most δ. □

C.1 Certificates of Bounded Complexity and Algorithmic Optimization

In this section we show how to use our ListUpdate method as part of an algorithm for explicitly computing $(\epsilon, \mathcal{C})$-Bayes optimal models from data sampled i.i.d. from the underlying distribution $\mathcal{D}$. First, we must show that if we find certificates of sub-optimality $(g, h) \in \mathcal{C}$ on our dataset, that we can be assured that they are certificates of sub-optimality on the underlying distribution. Here, we invoke uniform convergence bounds that depend on $\mathcal{C}$ being a class of bounded complexity. Next, we must describe an algorithmic method for finding (μ, Δ) certificates of sub-optimality that maximize $\mu \cdot \Delta$. Here we give two approaches. The first approach is a reduction to cost sensitive (ternary) classification: the result of the reduction is that the ability to solve weighted multi-class classification problems over some class of models gives us the ability to find certificates of sub-optimality over a related class whenever they exist. The second approach takes an "EM" style alternating maximization approach over $g_i \in \mathcal{G}$ and $h_i \in \mathcal{H}$ in turn, where each alternating maximization step can be reduced to a binary classification problem. It is only guaranteed to converge to a local optimum (or saddlepoint) of its objective — i.e. to find a certificate of sub-optimality (g_i, h_i) that cannot be improved by changing either g_i or h_i unilaterally — but has the merit that it requires only standard binary classification algorithms for a class $\mathcal{G}$ and $\mathcal{H}$ to search for certificates of sub-optimality in $\mathcal{G} \times \mathcal{H}$. For simplicity of exposition, in this section we restrict attention to the binary classification problem, where the labels are binary ($\mathcal{Y} = \{0, 1\}$) and our loss function corresponds to classification error ($\ell(\hat{y}, y) = \mathbb{1}[\hat{y} \neq y]$) — but the approach readily extends to more general label sets (replacing VC-dimension with the appropriate notion of combinatorial dimension as necessary).

The algorithmic problem we need to solve at each round t is, given an existing model f_{t-1}, find a (μ, Δ)-certificate of optimality $(g, h) \in \mathcal{C}$ for f_{t-1} that maximizes $\mu \cdot \Delta$ as computed on the empirical data D — i.e. to solve:

$$(g_t, h_t) = \arg \max_{(g,h) \in \mathcal{C}} \mu_D(g) \cdot (L(D, f_{t-1}, g) - L(D, h, g)) \tag{1}$$

We defer for now the algorithmic problem of finding these certificates, and describe a generic algorithm that can be invoked with any method for finding such certificates. First, we state a useful sample complexity bound proven in [24] in a related context of multi-group fairness — we here state the adaptation to our setting.

Lemma C.4 (Adapted from [24]). *Let $\mathcal{G}$ denote a class of group indicator functions with VC-dimension d_G and $\mathcal{H}$ denote a class of binary models with VC-dimension d_H. Let f be an arbitrary binary model. Then if:*

$$n \geq \tilde{O}\left(\frac{(d_H + d_G) + \log(1/\delta)}{\eta^2}\right)$$

we have that with probability $1 - \delta$ over the draw of a dataset $D \sim \mathcal{D}^n$, for every $g \in \mathcal{G}$ and $h \in \mathcal{H}$:

$$\left|\mu_{\mathcal{D}}(g) \cdot (L(\mathcal{D}, f, g) - L(\mathcal{D}, h, g)) - \mu_D(g_p) \cdot (L(D, f, g) - L(D, h, g))\right| \leq \eta$$

With our sample complexity lemma in hand we are ready to present our generic reduction from training an $(\epsilon, \mathcal{C})$-Bayes optimal model to the optimization problem over $\mathcal{C}$ given in (1). It is a reduction from the problem of training an $(\epsilon, \mathcal{C})$-Bayes optimal model to the optimization problem over $\mathcal{C}$ given in (1).

Theorem C.5. *Fix an arbitrary distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, a class of group indicator functions $\mathcal{G}$ of VC-dimension d_G and a class of binary models $\mathcal{H}$ of VC-dimension d_H. Let $\mathcal{C} \subseteq \mathcal{G} \times \mathcal{H}$ and $\epsilon > 0$ be arbitrary. If:*

$$n \geq \tilde{O}\left(\frac{(d_H + d_G) + \log(1/\delta)}{\epsilon^3}\right)$$

and $D \sim \mathcal{D}^n$, then with probability $1 - \delta$, TrainByOpt$(D, \mathcal{C}, \epsilon)$ (Algorithm 5) returns a model f that is $(\epsilon, \mathcal{C})$-Bayes optimal, using at most $2/\epsilon$ calls to a sub-routine for solving the optimization problem over $(g, h) \in \mathcal{C}$ given in (1).

Proof. Each of the partitioned datasets D_t has size at least $\tilde{O}\left(\frac{(d_H+d_G)+\log(1/\delta)}{\epsilon^2}\right)$ and is selected independently of f_{t-1} and so we can invoke Lemma C.4 with $\eta = \epsilon/8$ and $\delta = \delta/T$ to conclude that with probability $1 - \delta$, for every round t:

$$|\mu_D(g_t) \cdot (L(D, f_{t-1}, g_t) - L(D, h_t, g_t)) - \mu_{\mathcal{D}}(g_t) \cdot (L(\mathcal{D}, f_{t-1}, g_t) - L(\mathcal{D}, h_t, g_t))| \leq \frac{\epsilon}{8}$$

Input: A dataset D, a set of group/model pairs C, and accuracy parameter ϵ.
Let $f_0 : \mathcal{X} \to \{0,1\}$ be an arbitrary initial model and $T = 2/\epsilon$
Randomly divide D into T equally sized datasets: $D_1, \ldots, D_T$.
for $t = 1$ *to* T **do**
  Let
$$(g_t, h_t) = \arg\max_{(g,h)\in C} \mu_D(g) \cdot (L(D, f_{t-1}, g) - L(D, h, g))$$
  if $\mu_D(g_t) \cdot (L(D, f_{t-1}, g_t) - L(D, h_t, g_t)) \leq \frac{3\epsilon}{4}$ **then**
    Output: Model f_{t-1}
  else
    Let $f_t = \text{ListUpdate}(f_{t-1}, (g_t, h_t))$
Output: Model f_T

Algorithm 5: TrainByOpt(D, C, ϵ): An algorithm for training an (ϵ, C)-Bayes optimal model given the ability to optimize over certificates $(g, h) \in C$.

For the rest of the argument we will assume this condition obtains. By assumption, at every round t the models (g_t, h_t) exactly maximize $\mu_D(g_t) \cdot (L(D, f_{t-1}, g_t) - L(D, h_t, g_t))$ amongst all models $(g, h) \in C$, and so in combination with the above uniform convergence bound, they are $\epsilon/4$-approximate maximizers of $\mu_{\mathcal{D}}(g_t) \cdot (L(\mathcal{D}, f_{t-1}, g_t) - L(\mathcal{D}, h_t, g_t))$. Therefore, if at any round $t \leq T$ the algorithm outputs a model f_{t-1}, it must be because:

$$\max_{(g,h)\in C} \mu_{\mathcal{D}}(g) \cdot (L(\mathcal{D}, f_{t-1}, g) - L(\mathcal{D}, h, g)) \leq \frac{3\epsilon}{4} + \frac{\epsilon}{4} = \epsilon$$

Equivalently, for every $(g, h) \in C$, it must be that:

$$L(\mathcal{D}, f_{t-1}, g) \leq L(\mathcal{D}, h, g) + \frac{\epsilon}{\mu_{\mathcal{D}}(g)}$$

By definition, such a model is (ϵ, C)-Bayes optimal.

Similarly, if at any round $t < T$ we do not output f_{t-1}, then it must be that (g_t, h_t) forms a (μ, Δ)-certificate of sub-optimality for $\mu = \mu_{\mathcal{D}}(g_t)$ and $\Delta = L(\mathcal{D}, f_{t-1}, g) - L(\mathcal{D}, h, g)$ such that $\mu \cdot \Delta \geq \frac{3\epsilon}{4} - \frac{\epsilon}{4} = \frac{\epsilon}{2}$. By Theorem 3.4 we have that for any sequence of models $f_0, f_1, \ldots, f_T$ such that $f_t = \text{ListUpdate}(f_{t-1}, (g_t, h_t))$ and (g_t, h_t) forms a (μ, Δ)-certificate of sub-optimalty for f_{t-1} with $\mu \cdot \Delta \geq \frac{\epsilon}{2}$, it must be that $T \leq \frac{2}{\epsilon}$. Therefore, it must be that if our algorithm outputs model f_T, then f_T must be $\frac{\epsilon}{2}$-Bayes optimal. If this were not the case, then by Theorem 3.2, there would exist a (μ, Δ)-certificate of sub-optimality (g^*, h^*) for f_T with $\mu \cdot \Delta \geq \frac{\epsilon}{2}$, which we could use to extend the sequence by setting $f_{T+1} = \text{ListUpdate}(f_T, (g^*, h^*))$ — but that would contradict Theorem 3.4. Since $\epsilon/2$-Bayes optimality is strictly stronger than (ϵ, C)-Bayes optimality, this completes the proof. □

Algorithm 5 is an efficient algorithm for training an (ϵ, C)-Bayes optimal model whenever we can solve optimization problem (1) efficiently. We now turn to this optimization problem. In Section C.1.1 we show that the ability to solve cost sensitive classification problems over a *ternary* class K gives us the ability to solve optimization problem 1 over a related class C_K. In Section C.1.2 we show that the ability to solve standard empirical risk minimization problems over classes $\mathcal{G}$ and $\mathcal{H}$ respectively give us the ability to run iterative updates of an alternating-maximization ("EM style") approach to finding certificates $(g, h) \in \mathcal{G} \times \mathcal{H}$.

C.1.1 Finding Certificates via a Reduction to Cost-Sensitive Ternary Classification. For this approach, we start with an arbitrary class K of *ternary* valued functions $p : \mathcal{X} \to \{0, 1, ?\}$. It will be instructive to think of the label "?" as representing the decision to "defer" on an example, leaving the classification outcome to another model. We will identify such a ternary-valued function p with a pair of binary valued functions $g_p : \mathcal{X} \to \{0,1\}, h_p : \mathcal{X} \to \{0,1\}$ representing a group indicator function and a binary model h that might form a certificate of sub-optimality (g_p, h_p). They are defined as follows:

Definition C.6. Given a ternary valued function $p : \mathcal{X} \to \{0, 1, ?\}$, the p-derived group g_p and model h_p are defined as:

$$g_p(x) = \begin{cases} 1 & \text{if } p(x) \in \{0,1\} \\ 0 & \text{if } p(x) = ? \end{cases} \qquad h_p(x) = \begin{cases} p(x) & \text{if } p(x) \in \{0,1\} \\ 0 & \text{if } p(x) = ? \end{cases}$$

In other words, interpreting $p(x) =?$ as the decision for p to "defer" on x, g_p defines exactly the group of examples that p does *not* defer on, and h_p is the model that makes the same prediction as p on every example that p does not defer on. Given a class of ternary functions K, let the K-derived certificates C_K denote the set of pairs (g_p, h_p) that can be so derived from some $p \in K$: $C_K = \{(g_p, h_p) : p \in K\}$. Similarly let $\mathcal{G}_K = \{g_p : p \in K\}$ and $\mathcal{H}_K = \{h_p : p \in K\}$ denote the class of group indicator functions and models derived from K respectively.

Given a model $f : \mathcal{X} \to \{0,1\}$, our goal is to reduce the problem of solving optimization problem (1) over C_K to the problem of solving a ternary *cost-sensitive classification problem* over K:

Definition C.7. A cost-sensitive classification problem is defined by a model class K consisting of functions $p : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{Y}$ is some finite label set, a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, and a set of real valued costs $c_{(x,y)}(\hat{y})$ for each pair (x, y) in the support of $\mathcal{D}$ and each label $\hat{y} \in \mathcal{Y}$. A solution $p^* \in K$ to the cost-sensitive classification problem $(\mathcal{D}, K, \{c_{(x,y)}\})$ is given by:

$$p^* \in \arg\min_{p \in K} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[c_{(x,y)}(p(x))\right]$$

i.e. the model that minimizes the expected costs for the labels it assigns to points drawn from $\mathcal{D}$.

The reduction will make use of the following induced costs:

Definition C.8. Given a binary model $f : \mathcal{X} \to \mathcal{Y}$, the induced costs of f are defined as follows:

$$c^f_{(x,y)}(\hat{y}) = \begin{cases} 0 & \text{if } \hat{y} =? \\ 1 & \text{if } f(x) = y \neq \hat{y} \\ -1 & \text{if } \hat{y} = y \neq f(x) \\ 0 & \text{otherwise.} \end{cases}$$

Intuitively, it costs nothing to defer a decision to the existing model f — or equivalently, to make the same decision as f. On the other hand, making the *wrong* decision on an example x costs 1 when f would have made the right decision, and making the right decision "earns" 1 when f would not have.

Lemma C.9. *Fix an arbitrary distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, let K be a class of ternary valued functions, and let $f : \mathcal{X} \to \{0, 1\}$ be any binary valued model. Let p^* be a solution to the cost-sensitive classification problem $(\mathcal{D}, K, \{c^f_{(x,y)}\})$, where $c^f_{(x,y)}$ are the induced costs of f. We have that:*

$$(g_{p^*}, h_{p^*}) \in \arg\max_{(g,h)\in C_K} \mu_{\mathcal{D}}(g) \cdot (L(\mathcal{D}, f, g) - L(\mathcal{D}, h, g))$$

In other words, when $\mathcal{D}$ is the empirical distribution over D, (g_{p^*}, h_{p^*}) form a solution to optimization problem (1).

Proof. For any model $p \in K$, we can calculate its expected cost under the induced costs of f:

$$\begin{aligned} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[c_{(x,y)}(p(x))\right] &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} [\mathbb{1}[p(x) \neq ?] \cdot \mathbb{1}[p(x) \neq f(x)] \cdot (\mathbb{1}[p(x) \neq y] - \mathbb{1}[p(x) = y])] \\ &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\mathbb{1}[g_p(x) = 1] \cdot \mathbb{1}[h_p(x) \neq f(x)](\mathbb{1}[h_p(x) \neq y] - \mathbb{1}[h_p(x) = y])\right] \\ &= \mu_{\mathcal{D}}(g_p) \cdot (L(\mathcal{D}, h_p, g_p) - L(\mathcal{D}, f, g_p)) \end{aligned}$$

Thus *minimizing* $\mathbb{E}_{(x,y)\sim\mathcal{D}} \left[c_{(x,y)}(p(x))\right]$ is equivalent to *maximizing* $\mu_{\mathcal{D}}(g_p) \cdot (L(\mathcal{D}, f, g_p) - L(\mathcal{D}, h_p, g_p))$ over $p \in K$. □

In other words, in order to be able to efficiently implement algorithm 5 for the class C_K, it suffices to be able to solve a ternary cost sensitive classification problem over K. The up-shot is that if we can efficiently solve weighted multi-class classification problems (for which we have many algorithms which form good heuristics) over K, then we can find approximately C_K-Bayes optimal models as well.

C.1.2 Finding Certificates Using Alternating Maximization. The reduction from optimization problem (1) that we gave in Section C.1.1 to ternary cost sensitive classification starts with a ternary class K and then finds certificates of sub-optimality over a derived class C_K. What if we want to *start* with a pre-defined class of group indicator functions $\mathcal{G}$ and models $\mathcal{H}$, and find certificates of sub-optimality $(g, h) \in \mathcal{G} \times \mathcal{H}$? In this section we give an alternating maximization method that attempts to solve optimization problem 1 by alternating between maximizing over g_t (holding h_t fixed), and maximizing over h_t (holding g_t fixed):

$$g_t = \arg\max_{g\in\mathcal{G}} \mu_D(g) \cdot (L(D, f_{t-1}, g) - L(D, h_t, g)) \tag{2}$$

$$h_t = \arg\max_{h\in\mathcal{H}} \mu_D(g_t) \cdot (L(D, f_{t-1}, g_t) - L(D, h, g_t)) \tag{3}$$

We show that each of these alternating maximization steps can be reduced to solving a standard (unweighted) empirical risk minimization problem over $\mathcal{G}$ and $\mathcal{H}$ respectively. Thus, each can be solved for any heuristic for standard machine learning problems — we do not even require support for weighted examples, as we do in the cost sensitive classification approach from Section C.1.1. This alternating maximization approach quickly converges to a local optimum or saddle point of the optimization objective from from (1) — i.e. a solution that cannot be improved by either a unilateral change of either $g_t \in \mathcal{G}$ or $h_t \in \mathcal{H}$.

We begin with the minimization problem (3) over h, holding g_t fixed, and show that it reduces to an empirical risk minimization problem over $\mathcal{H}$.

Lemma C.10. *Fix any $g_t \in \mathcal{G}$ and dataset $D \in \mathcal{X}^n$. Let $D_{g_t} = \{(x, y) \in \mathcal{D} : g_t(x) = 1\}$ be the subset of D consisting of members of group g. Let $h^* = \arg\min_{h\in\mathcal{H}} L(D_{g_t}, h)$. Then h^* is a solution to optimization problem 3.*

Proof. We observe that only the final term of the optimization objective in (3) has any dependence on h when g_t is held fixed. Therefore:

$$\begin{aligned}\arg\max_{h\in\mathcal{H}} \mu_D(g_t)\cdot(L(D,f_{t-1},g_t)-L(D,h,g_t)) &= \arg\max_{h\in\mathcal{H}} -\mu_D(g_t)\cdot L(D,h,g_t))\\ &= \arg\min_{h\in\mathcal{H}} L(D,h,g_t)\\ &= \arg\min_{h\in\mathcal{H}} L(D_{g_t},h)\end{aligned}$$

□

Next we consider the minimization problem (2) over g, holding h_t fixed and show that it reduces to an empirical risk minimization problem over $\mathcal{G}$. The intuition behind the below construction is that the only points x that matter in optimizing our objective are those on which the models f_{t-1} and h_t disagree. Amongst these points x on which the two models disagree, we want to have $g(x) = 1$ if h correctly predicts the label, and not otherwise.

Lemma C.11. *Fix any $h_t \in \mathcal{H}$ and dataset $D \in \mathcal{X}^n$. Let:*

$$D^1_{h_t} = \{(x,y)\in D : h_t(x) = y \neq f_{t-1}(x)\} \quad D^0_{h_t} = \{(x,y)\in D : h_t(x) \neq y = f_{t-1}(x)\}$$

$$D_{h_t} = \left(\bigcup_{(x,y)\in D^1_{h_t}} (x,1)\right) \cup \left(\bigcup_{(x,y)\in D^0_{h_t}} (x,0)\right)$$

Let $g^* = \arg\min_{g\in\mathcal{G}} L(D_{h_t}, g)$. Then g^* is a solution to optimization problem (2).

Proof. For each $g \in \mathcal{G}$, we partition the set of points $(x,y) \in D_{h_t}$ such that $g(x) = 1$ according to how they are labelled by f_{t_1} and h_t:

$$S_1(g) = \{(x,y)\in D_{h_t} : g(x)=1, f_{t-1}(x) = h_t(x) = y\} \quad S_2(g) = \{(x,y)\in D_{h_t} : g(x)=1, f_{t-1}(x) = h_t(x) \neq y\}$$

$$S_3(g) = \{(x,y)\in D_{h_t} : g(x)=1, f_{t-1}(x) \neq h_t(x) = y\} \quad S_4(g) = \{(x,y)\in D_{h_t} : g(x)=1, f_{t-1}(x) = y \neq h_t(x)\}$$

i.e. amongst the points in group g, $S_1(g)$ consists of the points that both f_{t-1} and h_t classify correctly, $S_2(g)$ consists of the points that both classify incorrectly, and $S_3(g)$ and $S_4(g)$ consist of points that f_{t-1} and h_t disagree on: $S_3(g)$ are those points that h_t classifies correctly, and $S_4(g)$ are those points that f_{t-1} classifies correctly. Write:

$$w_1(g) = \frac{|S_1(g)|}{|D_{h_t}|} \quad w_2(g) = \frac{|S_2(g)|}{|D_{h_t}|} \quad w_3(g) = \frac{|S_3(g)|}{|D_{h_t}|} \quad w_4(g) = \frac{|S_4(g)|}{|D_{h_t}|}$$

to denote the corresponding proportions of each of the sets S_i within D_{h_t}.

Now observe that $\mu_D(g) = w_1(g) + w_2(g) + w_3(g) + w_4(g)$, $L(D, f_{t-1}, g) = \frac{w_2(g)+w_3(g)}{\mu_D(g)}$, and $L(D, h_t, g) = \frac{w_2(g)+w_4(g)}{\mu_D(g)}$. Therefore we can rewrite the objective of optimization problem (2) as:

$$\begin{aligned}\arg\max_{g\in\mathcal{G}} \mu_D(g)\cdot(L(D,f_{t-1},g)-L(D,h_t,g)) &= \arg\max_{g\in\mathcal{G}}(w_2(g)+w_3(g)) - (w_2(g)+w_4(g))\\ &= \arg\max_{g\in\mathcal{G}} w_3(g) - w_4(g)\\ &= \arg\min_{g\in\mathcal{G}} w_4(g) - w_3(g)\\ &= \arg\min_{g\in\mathcal{G}} L(D_{h_t},g)\end{aligned}$$

□

With these two components in hand, we can describe our alternating maximization algorithm for finding $\mathcal{G}\times\mathcal{H}$ certificates of sub-optimality for a model f_{t-1} (Algorithm 6)

We have the following theorem:

Theorem C.12. *Let $D \in (\mathcal{X}\times\mathcal{Y})^n$ be an arbitrary dataset, $f_{t-1} : \mathcal{X}\to\mathcal{Y}$ be an arbitrary model, $\mathcal{G}$ and $\mathcal{H}$ be arbitrary group and model classes, and $\epsilon > 0$. Then after solving at most $2/\epsilon$ empirical risk minimization problems over each of $\mathcal{G}$ and $\mathcal{H}$, AltMinCertificateFinder (Algorithm 6) returns a (μ,Δ)-certificate of sub-optimality (g_t, h_t) for f_{t-1} that is an ϵ-approximate local optimum (or saddle point) in the sense that:*

(1) *For every $h \in \mathcal{H}$, (g_t, h) is not a (μ',Δ') certificate of sub-optimality for f_{t-1} for any $\mu'\cdot\Delta' \geq \mu\cdot\Delta + \epsilon$, and*
(2) *For every $g \in \mathcal{G}$, (g, h_t) is not a (μ',Δ') certificate of sub-optimality for f_{t-1} for any $\mu'\cdot\Delta' \geq \mu\cdot\Delta + \epsilon$*

Input: A dataset D, a model f_{t-1}, group and model classes $\mathcal{G}$ and $\mathcal{H}$, and an error parameter ϵ.
Let $(g^*, h^*) \in \mathcal{G} \times \mathcal{H}$ be an arbitrary initial certificate.
CurrentValue $\leftarrow \mu_D(g^*) \cdot (L(D, f_{t-1}, g^*) - L(D, h^*, g^*))$
Let:

$$h^* = \arg\min_{h \in \mathcal{H}} L(D_{g^*}, h) \quad g^* = \arg\min_{g \in \mathcal{G}} L(D_{h^*}, g)$$

while $\mu_D(g^*) \cdot (L(D, f_{t-1}, g^*) - L(D, h^*, g^*) \geq \mathit{CurrentValue} + \epsilon$ **do**
  CurrentValue $\leftarrow \mu_D(g^*) \cdot (L(D, f_{t-1}, g^*) - L(D, h^*, g^*))$
  Let:

$$h^* = \arg\min_{h \in \mathcal{H}} L(D_{g^*}, h) \quad g^* = \arg\min_{g \in \mathcal{G}} L(D_{h^*}, g)$$

Output: Certificate $(g_t, h_t) = (g^*, h^*)$

Algorithm 6: AltMinCertificateFinder($D, f_{t-1}, \mathcal{G}, \mathcal{H}, \epsilon$): An algorithm for finding ϵ-locally optimal certificates of sub-optimality $(g_t, h_t) \in \mathcal{G} \times \mathcal{H}$ for model f_{t-1}.

Proof. By the halting condition of the While loop, every iteration of the While loop increases $\mu_D(g^*) \cdot (L(D, f_{t-1}, g^*) - L(D, h^*, g^*))$ by at least ϵ. Since this quantity is bounded in $[-1, 1]$, there cannot be more than $2/\epsilon$ iterations. Each iteration solves a single empirical risk minimization problem over each of $\mathcal{H}$ and $\mathcal{G}$.

The certificate (g_t, h_t) finally output is a (μ, Δ) certificate of sub-optimality for $\mu = \mu_D(g_t)$ and $\Delta = (L(D, f_{t-1}, g_t) - L(D, h_t, g_t))$. It must be that the objective $\mu_D(g_t) \cdot (L(D, f_{t-1}, g_t) - L(D, h_t, g_t)) = \mu \cdot \Delta$ cannot be improved by more than ϵ by re-optimizing either h_t or g_t, by definition of the halting condition and by Lemmas C.10 and C.11. The theorem follows. □

D ADDITIONAL MATERIAL FROM SECTION 5

Dataset	Total	White	Black	Asian	Native	Other	Two+	Male	Female	Young	Middle	Old
NY Employment	196966	138473	24024	17030	10964	5646	829	95162	101804	68163	47469	81334
Oregon Income	21918	18937	311	923	552	823	372	11454	10464	5041	9124	7753
Texas Coverage	98927	72881	11192	4844	6660	2555	795	42128	56799	42048	31300	25579
Florida Travel	88070	70627	10118	2629	2499	1907	290	45324	42746	16710	35151	36209

Table 2: Summaries of the four Folktables state/task datasets we use in the experiments discussed below, indicating total population size and the sizes of the different demographic subgroups considered. The full descriptors of the demographic subgroups are: (race subgroups) White; Black or African American; Asian; Native Hawaiian, Native American, Native Alaskan, or Pacific Islander; Some Other Race; Two or More Races; (binarized sex subgroups) Male; Female; (age subgroups) Young; Middle; Old.

	AGEP	SCHL	MAR	RELP	DIS	ESP	CIT	MIG	MIL	ANC	NATIVITY	DEAR	DEYE	DREM	SEX	RAC1P
0	42.0	19.0	3.0	13.0	2.0	0.0	1.0	1.0	4.0	2.0	1.0	2.0	2.0	2.0	1.0	1.0
1	40.0	21.0	5.0	0.0	2.0	0.0	1.0	3.0	4.0	1.0	1.0	2.0	2.0	2.0	1.0	2.0
2	21.0	15.0	5.0	17.0	1.0	0.0	1.0	1.0	4.0	1.0	1.0	1.0	2.0	2.0	2.0	2.0
3	48.0	16.0	5.0	2.0	2.0	0.0	1.0	1.0	4.0	1.0	1.0	2.0	2.0	2.0	1.0	1.0
4	40.0	9.0	5.0	0.0	2.0	0.0	5.0	1.0	4.0	1.0	2.0	2.0	2.0	2.0	2.0	8.0

Figure 5: A fragment of the Folktables ACS PUMS dataset, with the 16 features considered for an employment prediction task: age (AGEP), education (SCHL), marital status (MAR), relationship (RELP), disability status (DIS), parent employee status (ESP), citizen status (CIT), mobility status (MIG), military service (MIL), ancestry record (ANC), nation of origin (NATIVITY), hearing difficulty (DEAR), visual dificulty (DEYE), learning disability (DREM), sex (SEX), and race (RAC1P).

Total	White	Black	Asian	Native	Some Other Race	Two+	Male	Female	Young	Middle	Old
0.0004	0.016	0.065	0.023	0.037	0.020	0.033	0.016	0.008	0.040	0.045	0.030

Table 3: Absolute differences in final overall and subgroup test errors between two different sequential orderings. See text for details.

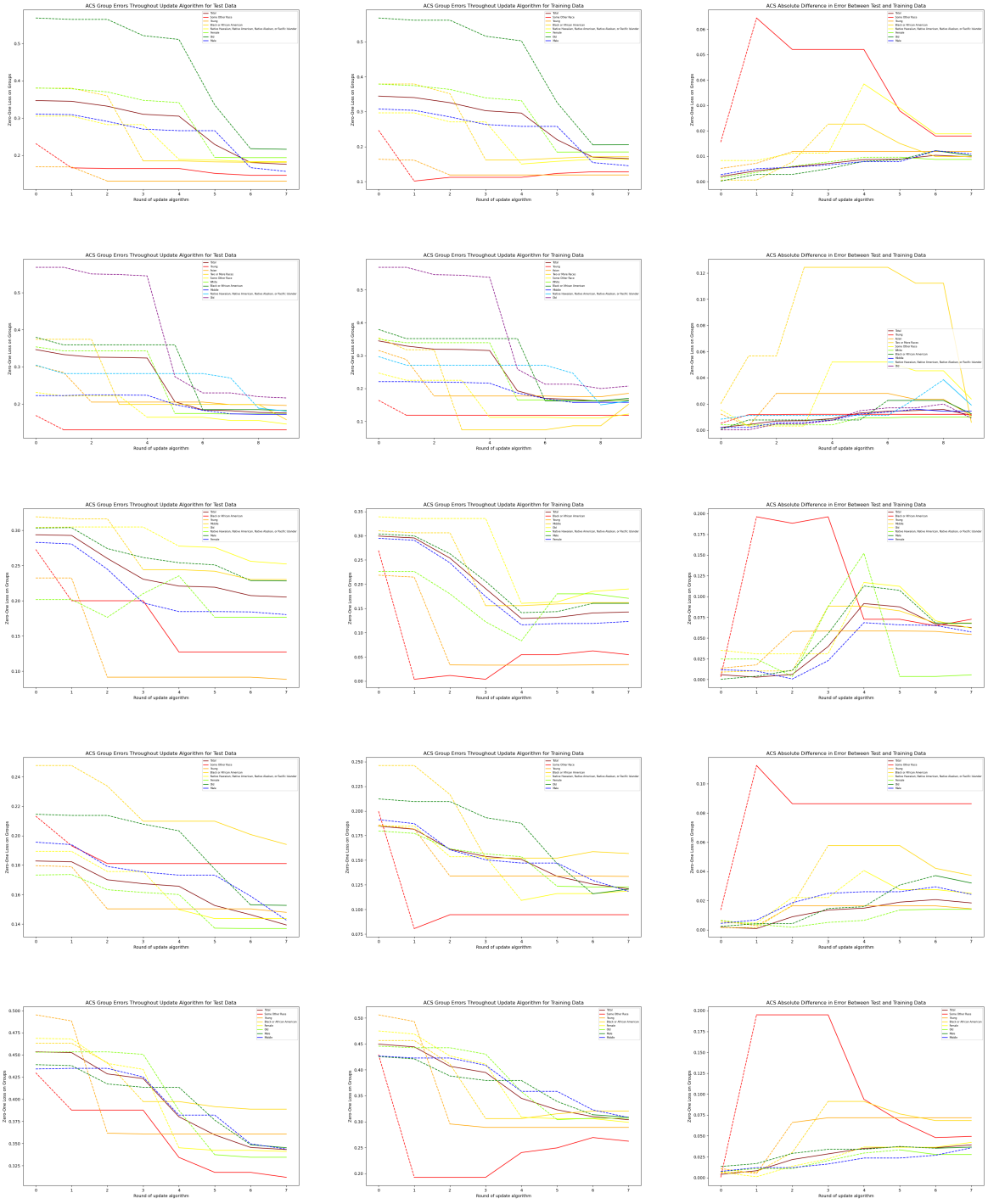

Figure 6: Sample results of an implementation of algorithm MonotoneFalsifyAndUpdate (Algorithm 4) on a number of different task and U.S. state datasets from the Folktables package. In each row, the left panel shows group error rates on the test data; the middle panel shows group error rates on the training data; and the right panel shows the absolute difference of test and training error rates. First row: ACS employment task for New York state. Second row: ACS employment task for New York but with a different order of introduction of the groups. Third row: ACS income task for Oregon. Fourth row: ACS coverage task for Texas. Fifth row: ACS travel time task for Florida. See text for discussion.