A Learning-Theoretic Framework for Certified Auditing with Explanations

Chhavi Yadav  
UC San Diego

Michal Moshkovitz  
Tel Aviv University

Kamalika Chaudhuri  
UC San Diego

Abstract

Responsible use of machine learning requires models be audited for undesirable properties. While a number of auditing algorithms have been proposed in prior work, how to principled auditing in a general setting has remained ill-understood. This work proposes a formal learning-theoretic framework for auditing, and uses it to investigate if and how model explanations can help audits. Specifically, we propose algorithms for auditing linear classifiers for feature sensitivity using label queries as well as two kinds of explanations, and provide performance guarantees. Our results illustrate that while counterfactual explanations can be extremely helpful for auditing, anchor explanations may not be as beneficial in the worst case.

1 Introduction

The recent success of machine learning (ML) has opened up new avenues for its potential use in high-stakes societal applications. For example, Bojarski et al. (2016) uses ML based detectors for self-driving cars; Aini et al. (2020) summarizes the utility of ML models in judiciary, and Castiglioni et al. (2021) discusses ML image systems to help doctors make decisions. However, this kind of wide-spread usage requires us to be able to audit these models extensively and verify that they possess certain desirable properties related to safety, robustness and fairness.

So far, due to complexities in the pipeline and the black box nature of most machine learning models, auditing properties of machine learning models has been primarily performed in a somewhat ad-hoc manner, usually by creating a separate audit testing set. Lately, a body of work has also posited that model explanations might be helpful for the purpose of auditing. Bhatt et al. (2020) Oala et al. 2020 [Adebayo and Gorelick 2017] Watson and Floridi 2021 Zhang et al. 2022 Poland 2022 Hamelers 2021 Carvalho et al. 2019), however, a precise understanding of how or why that is the case is also lacking.

This work seeks to address these limitations by proposing a formal learning-theoretic framework for auditing properties of machine learning models. Our framework formalizes auditing as a conversation between two entities – a data scientist, who holds a model, and an auditor who seeks to audit a specific property. Due to confidentiality reasons, the auditor does not have direct access to the model, and can only access it through querying specific data points from the data scientist. The auditor’s goal is to determine if the model has some property using as few queries as possible, where the response to a query is comprised of both labels and explanations.

To measure success in this framework, we propose a formal definition for a good auditor that measures both soundness and completeness. Soundness ensures that if the auditing property holds to a large extent, then the auditor agrees, and completeness ensures that the auditor says no when the auditing property does not hold. We then provide a fallback auditing strategy that applies to any general auditing property that is a property of the model, and is inspired by a connection between auditing in our framework and membership query active learning. This fallback strategy allows us to give a baseline on the number of queries required for auditing.

We then exemplify our framework by considering a use-case: auditing if a model is sensitive to a specific feature – a property that is relevant for detecting spurious correlations as well as fairness. We consider two hypothesis classes – linear classifiers and extended thresholds – and two kinds of explanations – counterfactual and anchors – that could be returned by the data scientist. To audit anchor explanations, we propose a novel algorithm called Anchor Augmentation that incorporates these explanations into an active querying strategy for efficient auditing. We provide guarantees on its query complexity. For counterfactual explanations, we provide algorithms for auditing that use a single query (for linear classifiers) and two queries (for extended thresholds). Our results illustrate that the two explanation methods differ greatly in their efficacy in auditing –

Preliminary work. Under review.
while worst-case anchors do not provide any more information than labels, counterfactuals greatly bring down the number of queries to one or two.

Finally, we evaluate our algorithms on two standard datasets. Our investigations with linear classifiers show that while worst-case anchors do not provide any more information than labels, typical anchors significantly reduce the number of queries needed to audit. Additionally, our proposed Anchor Augmentation technique helps reduce the query complexity of auditing over using a passive approach. Thus, our formal framework allows us to determine if and when explanations are helpful for the purpose of auditing ML models, as well as design good algorithms using these explanations effectively in the auditing process. We see our work as an early step towards paving the way for principled and theoretically sound ML governance.

1.1 Related Work

Auditing has been used to uncover undesirable behavior in ML models in the past. For instance, different kinds of biases were identified by Buolamwini and Gebru (2018) in commercial facial recognition models, by Koekeke et al. (2020) in automated speech recognition tools and by Tatman and Kasten (2017); Tatman (2017) in automatic captions on Youtube. However, these works were based on creating a audit dataset in an ad-hoc manner. Recently, there have been efforts towards streamlining and developing a structured process for auditing. For instance, Gebru et al. (2018); Mitchell et al. (2019) introduce datasheets for datasets and model cards and Raji et al. (2020) introduce an end-to-end internal auditing process. In contrast with these works, we look at creating a formal theoretical framework which allows us to determine what kind of properties can be audited and under what conditions.

Auditing with Formal Guarantees. The work most related to ours is concurrent work due to Yan and Zhang (2022), who present a formal framework as well as algorithms for auditing fairness in ML models by checking if a model has a certain demographic parity on a data distribution. Their algorithms are motivated by connections to active learning as well as machine teaching Goldman and Kearns (1995). In contrast, we look at auditing any general property that can be described by the score function, and we incorporate explanations into our auditing process. Our instantiation – auditing feature sensitivity – is different from demographic parity, which, together with the use of explanations, leads to very different algorithms.

Another work on auditing with guarantees is Goldwasser et al. (2021), who audit the accuracy of ML models. They use an interactive protocol between the verifier and the prover like our work, and their algorithm is related to a notion of property testing due to Balcan et al. (2011). However, our paper is more general from these in the following sense – it formalizes auditing for general auditing tasks (not just accuracy) and also demonstrates the utility of explanations in auditing.

Auditing and Explanations. Explanations are supposed to help in auditing as suggested in (Bhatt et al., 2020; Oala et al., 2020; Adebayo and Gorelick, 2017; Watson and Floridi, 2021; Zhang et al., 2022; Poland, 2022; Hamelers, 2021; Carvalho et al., 2019), but exactly how and under what conditions was not understood. Our results throw some light on these questions by showing how explanations can be incorporated in the auditing process theoretically and giving algorithms that use these explanations. We show how explanations can affect the query complexity of auditing algorithms.

Other Auditing Methods. Specific statistical methods have been proposed by Jagielski et al. (2020); Ye et al. (2021) to audit privacy and by Liu and Tsattaras (2020); Huang et al. (2021) to audit data deletion. These are properties of the learning algorithm rather than the model. In our paper we focus on the latter and hence our algorithms are not applicable to auditing privacy and data deletion. Extending our framework to broader settings is an important direction of future work.

2 Framework

Informally, an audit is an examination of an algorithm or model to check for consistency with specified norms or policies. To ground auditing for ML, we introduce the main components involved in the process, specifically, 1) the model/hypothesis to be audited \( h \), 2) the auditing property (AP) aka the norm/policy, 3) the Data Scientist (DS), aka the creator of the model \( h \) and has complete access to it and 4) the Auditor, aka the one who audits the model \( h \) for AP and has query access to \( h \) through the DS. Next we introduce notation and define these components mathematically.

2.1 Setup

Let \( \mathcal{X} \subseteq \mathbb{R}^d \) be the instance space and \( \mathcal{Y} \) be the label space. Let \( \mathcal{H} \) be the hypothesis class, consisting of hypotheses \( h : \mathcal{X} \rightarrow \mathcal{Y} \). \( \mathcal{H} \) is known to both the DS and auditor. We assume that the model to be audited belongs to the hypothesis class – that is, \( \hat{h} \in \mathcal{H} \). The DS has white box access to \( \hat{h} \), while auditor only has query access.

When a query \( x \in \mathcal{X} \) is asked to the DS, it returns the corresponding label and a local model explanation. We consider two types of local explanations – Counterfactuals (Wachter et al., 2017) and Anchors (Ribeiro et al., 2018). We define them formally below.
Definition 1 (Counterfactual Explanations). Given an input \( x \) and a model \( h \), a counterfactual explanation returns the closest instance in \( L_2 \) distance \( x' \) that \( h \) labels differently from \( x \). More precisely, 
\[
x' = \arg\min_{x': h(x') \neq h(x)} \| x - x' \|_2.
\]

Definition 2 (Anchor Explanations). Given a model \( h \), an instance \( x \), a distribution \( D \), and a precision parameter \( \tau \), an anchor explanation returns a hyperrectangle \( A_x \) such that (a) \( A_x \) contains \( x \) (b) at least \( \tau \) fraction of the points in \( A_x \) under \( D \) have the same label \( h(x) \) \cite{Ribeiro2018, Dasgupta2022}. Specifically, 
\[
\tau = \Pr_{x' \in D | A_x} (h(x) = h(x')).
\]
Here, the precision parameter \( \tau \) measures the quality of the anchor explanation. To describe the quality of an anchor explanation, we also use a coverage parameter, \( c \) – which is the probability that a point sampled according to \( D \) lies in the \( A_x \), 
\[
c = \Pr_{x' \sim D} (x' \in A_x).
\]

Both DS and the auditor agree upon a specific explanation method before auditing begins. Let \( e_h : \mathcal{X} \to \mathcal{E} \) denote this explanation method where \( \mathcal{E} \) represents the codomain of \( e_h \) and \( h \in \mathcal{H} \). For example, for counterfactuals, \( \mathcal{E} = \mathcal{X} \).

Auditing Property and Score Functions. We specify the auditing property (AP) using a continuous score function, \( s : \mathcal{H} \to [0, 1] \). \( s(\cdot) \) captures the extent to which the model “follows” the AP, with \( s(\cdot) = 0 \) when the AP is not followed by the model. While the exact form of function \( s \) changes according to the AP, \( s \) encapsulates any auditing property that depends on the hypothesis like accuracy, precision, demographic parity etc. Auditor knows the score function but does not know its value for \( h \) (since \( h \) is hidden from it). For example, consider “unfairness” to be the AP and its corresponding score function. If a model is unfair on a large part of the data, then its score should be higher than a model which is unfair on only a small part. For a completely fair model, i.e., a model with no unfairness, the score should be zero. A lot of score functions will be probabilistic in nature and will have to be estimated using samples from a distribution. We assume that \( s \) can be computed to an arbitrary precision for any hypothesis. Also for most auditing algorithms considered in Section 2.3 the score function becomes deterministic.

We assume that the DS is truthful, meaning the labels and explanations returned for \( x \) are \( y = h(x) \) and \( e_h(x) \) respectively. In Section 2.3 we discuss what happens when this assumption does not hold.

2.2 Auditing Framework

In this section, we conceptualize the auditing process as an interaction between the DS and auditor. Our protocol is as follows.

At each time step \( t = 1, 2, \ldots \)
- Auditor picks a new query \( x_t \in \mathcal{X} \) and supplies it to the DS.
- DS returns a label \( y_t \in \mathcal{Y} \) and an explanation \( e_t \in \mathcal{E} \) to the auditor.
- Auditor decides whether or not to stop. If auditor decides to stop, it returns a decision \( Y_a \in \{\text{Yes, No}\} \), otherwise it continues to the next time step.

At the end of the auditing process, the auditor responds with an answer \( Y_a \) which takes values \( \{\text{Yes, No}\} \). This is a random variable since both the DS and auditor can be randomized algorithms. Next we formally define what makes a successful auditor.

Definition 3 ((\( \epsilon, \delta \))-auditor). An auditor is an \( (\epsilon, \delta) \)-auditor for \( \epsilon, \delta \in [0, 1] \), hypothesis class \( \mathcal{H} \) and a score function \( s(\cdot) \) if \( \forall h \in \mathcal{H} \) the following conditions hold: 1) if \( s(h) > \epsilon \), \( \Pr(Y_a = \text{Yes}) \geq 1 - \delta \) and 2) if \( s(h) = 0 \), \( \Pr(Y_a = \text{No}) = 1 \).

The first condition pertains to soundness, and says that a successful auditor should return a \text{Yes} with high probability when AP is followed by the model to a large extent. The second condition measures completeness, and requires the auditor to say \text{No} when AP is not followed at all. Observe that when \( 0 < s(h) \leq \epsilon \), we cannot place a guarantee on the auditor’s decision with a finite query budget.

The query complexity \( T \) of an \( (\epsilon, \delta) \)-auditor is the total number of queries it asks DS before stopping. Efficient auditing requires that \( T \) be small. Ideally, the \( (\epsilon, \delta) \) auditor should also be efficient computationally in addition to efficiency in the number of queries.

2.3 A General Auditor

We now propose a fallback strategy for the auditor that applies to any auditing property encapsulated by the score function \( s(\cdot) \), but may be computationally inefficient. The auditor’s strategy consists of two parts – strategy to pick the next query and the stopping condition.

Picking next query. In our fallback strategy, the auditor maintains a search space \( S_t \) over hypotheses with time \( t \), \( S_t \subseteq \mathcal{H} \). The search space consists of all hypotheses that could presumably be \( \hat{h} \) based on the evidence that the auditor has seen so far. In the beginning, \( S_0 = \mathcal{H} \), and then it is narrowed down through queries made to the DS at each subsequent time step until the stopping condition is reached.

Let \( Z_{t-1} \subseteq \mathcal{Z} \) and \( E_{t-1} \subseteq \mathcal{X} \times \mathcal{E} \) be the set of labeled examples and explanations acquired from the DS till \( t-1 \). We define the search space at \( t \) to be the subset of hypotheses \( S_t \subseteq \mathcal{H} \) that are consistent with all the labels.
and explanations in $Z_{t-1}$ and $E_{t-1}$ respectively. As an example for counterfactual explanations, the hypothesis $h$ is consistent if $\forall (x, h(x)) \in Z_{t-1}$ and their counterfactuals $(x, x' = e_h(x)) \in E_{t-1}$, $h(x) = \bar{h}(x)$ and $h(x') = \bar{h}(x')$.

For efficient pruning of the search space, the auditor chooses a query point that reduces the search space size the most, no matter what explanation and label are returned by the DS. Formally,

$$x_t = \arg\max_{x \in \mathcal{X}} (\min_{T \in \mathcal{Y}} |S_t| / |S_{t+1}|) = \arg\max_{x \in \mathcal{X}} \text{value}_x$$

where $|S_t|$ and $|S_{t+1}|$ denote the volume of the search spaces $S_t$ and $S_{t+1}$ and value$_x$ corresponds to the search space reduction by worst case label and explanation for a point $x$. Note that this is a computationally inefficient algorithm, since we need to search over all points in $\mathcal{X}$.

**Stopping.** An important aspect of the auditor is when does it stop and make a decision. This is determined by its stopping condition. We propose one such stopping condition next.

Since $\bar{h}$ is always guaranteed to be in the search space $S_t$, if $s(h) > \epsilon$ for all $h$ in $S_t$ then we can safely conclude that $\bar{h}$ has the desired property and stop and say Yes. Similarly, if $s(h) \leq \epsilon$ for all $h$ in $S_t$ the auditor will stop and say No. Observe that the auditor will ultimately arrive at one of these two cases – if none of the two stopping conditions hold, then the next query will cause the search space to shrink – and ultimately our search space will consist of a single $h$. Note that the stopping condition decides the query complexity of the auditor.

A formal outline for our fallback strategy is in Algorithm 1 in the Appendix. We prove that this strategy is an $(\epsilon, \delta)$-auditor, which follows directly from $\bar{h}$ belonging to the search space at all times and the stopping conditions.

**Theorem 2.1.** Algorithm1 is an $(\epsilon, \delta)$-auditor.

### 2.4 Connections to Active Learning

For readers who are knowledgeable on active learning, our fallback auditing strategy in Section 2.3 may appear familiar. In active learning (Dasgupta 2005), the goal is to learn a classifier in an interactive manner by querying highly informative unlabeled data points for labels. A specific variant of active learning is Membership Query Active Learning (MQAL) (Angluin 1988, Feldman 2009) where the learner synthesizes queries rather than sampling from the data distribution or selecting from a pool.

Our fallback strategy is essentially an active partial learning algorithm in the MQAL setting, with a modified oracle that returns explanations in addition to labels. Here, the DS is the oracle and the auditor is the learner. The algorithm does partial learning – since it stops when the auditing goal is complete and before the classifier $\bar{h}$ is fully learnt.

On this note, we observe that learning is a harder task than auditing – since the goal there is to find the classifier generating the labels while in auditing, we hope to decide if this classifier has a certain property. This implies that if there exists an algorithm that can learn a classifier, it can also audit it, and a membership query active learning algorithm can be used as a fallback auditing algorithm. This leads to the following fallback guarantee for any auditing property encapsulated by the score function.

**Theorem 2.2.** If there exists a membership query active learner that can learn $\bar{h}$ exactly in $T$ queries, then the active learner can also audit in $T$ queries.

Proof can be found in the appendix Section A.2.

Thus in summary, while our fallback strategy is connected to active learning, the key differences are – 1) the goals as mentioned above, 2) usage of explanations and 3) unlike active learning, partial learning of the hypothesis can be sufficient for auditing. For example, if all hypotheses $h$ in the search space have $s(h) = 0$, the auditor can return a decision without learning $\bar{h}$ exactly; or learning only some parameters of $\bar{h}$ may be enough to audit as shown in Section 3.2.1 Since partial learning can be sufficient for auditing, it also means that an auditing algorithm is not necessarily an active learning algorithm.

### 3 Auditing Feature Sensitivity

We next look at a specific instantiation of the proposed framework for auditing ‘feature sensitivity’. If flipping a specific feature in the input leads to a different prediction, we call it a sensitive feature for the model. Identifying sensitive features can lead to insights into a model’s working and uncover spurious correlations and harmful biases that it may have learnt. For instance, while auditing a model that predicts the presence of lung cancer based on a radiology image, a human auditor may suspect a feature in the input, such as the presence of pen markings on X-rays, to be spuriously correlated to the output; in this case, we might want to audit if the model is sensitive to this feature or not. Another example is when an auditor might suspect that an input feature such as home zipcode, which might act as a proxy for race (not included in the data), may be correlated to the output; in this case, we might want to audit if the model is sensitive to this feature or not.

#### 3.1 Setup and Preliminaries

Suppose we would like to audit a model $\bar{h}$ that maps an input $x \in \mathbb{R}^d$ to a label $y \in \{-1, 1\}$. $x_i$ denotes the $i^{th}$ feature of input $x$. Together with the model, we are given an input feature that we call the Feature of Interest; our goal is to determine if the model $\bar{h}$ is sensitive to this feature.

Two inputs $x^i, x^j \in \mathcal{X}$ form a pair $p_{ij}$ if for every feature
We call a pair \( (h, x) \) a **responsive pair** for \( h \in \mathcal{H} \), if \( h(x_i) \neq h(x_j) \). Unless mentioned, responsive pairs are with respect to \( h \). Lastly, the feature of interest is a **sensitive feature** for hypothesis \( h \) if a responsive pair exists for this feature.

The Auditing Property (AP) here is the sensitivity of small \( \epsilon \)-audits either. However, a lot of pairs have to be queried for a better option than auditing through learning might be hard. In such cases auditing using baseline might be a responsive pair according to \( \tilde{h} \), that is \( s(\tilde{h}) = \Pr_{P_{ij\in P}}(p_{ij} \text{ is a responsive pair}) \). This score function can be interpreted as the fraction of responsive pairs. To be an \((\epsilon, \delta)\)-auditor, an auditor should detect the existence of more than \( \epsilon \) fraction of responsive pairs with high probability and when the fraction is zero with probability one.

Before we go into our algorithm, we first propose a simple baseline: random testing.

**A Simple Baseline: Random Testing.** A simple way to detect feature sensitivity is to query a large number of pairs \( (h, x) \) at random from the DS. If any of them is a responsive pair, auditor knows that the feature is sensitive. The following theorem states a bound on the query complexity for this algorithm.

**Theorem 3.1.** For \( \epsilon, \delta \in (0, 1) \), auditor is an \((\epsilon, \delta)\)-auditor if it queries \( m \geq \frac{1}{\epsilon} \log \frac{1}{\delta} \) pairs.

Proof can be found in the appendix Section A.1. Note that this baseline is independent of the hypothesis class. It does not learn \( \tilde{h} \) (even partially) in order to audit or use explanations either. However, a lot of pairs have to be queried for a small \( \epsilon \) since the structure of the class is not exploited. Next we discuss cases where using explanations and exploiting the structure of hypothesis class leads to more efficient auditing.

### 3.2 Linear Classifiers

In this section, we audit **Linear Classifiers** for feature sensitivity using different explanation methods. **Linear Classifiers** are defined as follows.

\[
\mathcal{H}_{LC} = \{w, b \in \mathbb{R}^d, b \in \mathbb{R} \mid h_{w,b}(x) = \text{sign}(\langle w, x \rangle + b)\}
\]

Note that responsive pairs exist for \( h \in \mathcal{H}_{LC} \) for a feature of interest \( x_i \), if and only if weight \( w_i \) is non-zero.

#### 3.2.1 Counterfactual Explanations

Recall that the counterfactual explanation \( x' \) for an input \( x \) is the closest point to \( x \) that has a different label. For linear classifiers, we observe that the difference \( x - x' \) is parallel to \( w \). The auditor tests if the feature of interest in \( x - x' \) is zero. This observation can be used to design an algorithm that uses a single query denoted by \( \text{Alg}_{LC, c}^5 \).

**Theorem 3.2.** For any \( \epsilon \in [0, 1] \), auditor \( \text{Alg}_{LC, c}^5 \) is an \((\epsilon, 0)\)-auditor for \( \mathcal{H}_{LC} \) and score function \( s(\cdot) \) with \( T = 1 \) query.

Proof for the theorem can be found in the appendix section A.5.

#### 3.2.2 Anchor Explanations

We consider homogeneous linear classifiers \( (b = 0) \) in this section; however our techniques can be easily extended to non-homogeneous linear classifiers by considering \( d + 1 \) dimensions and concatenating \( 1 \) to \( x \) and \( b \) to \( w \). For simplicity, we assume our anchor explanations have perfect precision, \( \tau = 1 \) and coverage \( c = c_0 \) where \( c_0 \) is a scalar.

We propose an auditing algorithm for linear classifiers that is inspired by the active learning algorithm of [Alabdulmohsin et al. (2015)] that uses label queries only. The main idea of their algorithm is to construct an ellipsoidal approximation of the search space by solving a convex optimization problem and then extracting a principal component which serves as the synthesized query to the oracle. We will use this skeleton to narrow down our search space for \( \tilde{h} \).

Our additional contribution is a new method for incorporating anchors for higher auditing efficiency – a procedure that we call **Anchor Augmentation**. The main idea is that anchor explanations give us a region of space around a point \( x \) where the labels are the same as the label of \( x \); we use this fact to generate more synthetic labeled examples (without actually querying the DS) that can be fed into the algorithm. Through these observations we can come up with an auditing algorithm for linear classifiers using anchors, denoted by \( \text{Alg}_{LC, a} \).

**Worst Case Query Complexity.** In the worst-case, anchor explanations for \( x \) are degenerate rectangles of the form \( \{x \mid \lambda \in R^+\} \) and do not provide any extra information than the label itself. The query complexity of this is presented below.

**Theorem 3.3.** For every dimension \( d \), there exists \( c > 0 \) such that for any \( \epsilon \in (0, 1) \), auditor \( \text{Alg}_{LC, a}^c \) is an \((\epsilon, 0)\)-auditor for \( \mathcal{H}_{LC} \) and score function \( s(\cdot) \) with \( T = O \left( d \log \frac{2c}{\epsilon} \right) \) queries.

The proof is different from [Alabdulmohsin et al. (2015)] since in auditing the weight \( w_i \) has to be connected to the
score function. The main idea is that if \( h \) is well approximated, then as a result, the score function can be estimated well too. More details can be found in Appendix Section A.6.

However, this is a very pessimistic bound, as typical anchors are not worst-case and can lead to faster auditing. We demonstrate this empirically in Section 4.1.

3.3 Extended Thresholds

In this section, we consider another class of models for auditing – Extended Thresholds. While normal threshold models have a single threshold value, extended thresholds have two of them. Here, the threshold value varies with the value of a specific input feature, called the threshold-deciding feature. For example, consider a toy loan-granting model with two features - gender (threshold-deciding feature) and salary (thresholded feature). Then based on the value of gender, the model can have different thresholds on salary to grant or not grant loans.

Extended Thresholds are defined mathematically as follows. Let input \( x = (x', g) \) where \( x' \in R^d, g \in \{0, 1\} \) the threshold-deciding feature. Let \( f : R^d \rightarrow [l, u] \) where \( l, u \in R \) be a function that takes \( x' \in R^d \) and returns a single value to be thresholded.

\[
\mathcal{H}_{ET} = \{h_{f, \theta_1, \theta_2} \theta_1, \theta_2, f : R^d \rightarrow [l, u] \}
\]

\[
h_{\theta_1, \theta_2, f}((x, g)) = \begin{cases} +1 & \text{if } (1-g)\theta_1 + g\theta_2 \leq f(x') \\ -1 & \text{otherwise.} \end{cases}
\]

For auditing feature sensitivity, we assume that the auditor’s feature of interest is the threshold-deciding feature \( g \). Then the auditing task boils down to finding 1) if \( \theta_1 = \theta_2 \), in which case \( g \) is not sensitive, or 2) if \( \theta_1 \neq \theta_2 \), in which case \( g \) is a sensitive feature. Note that responsive pairs exist only when \( \theta_1 \neq \theta_2 \), in between \( \theta_1 \) and \( \theta_2 \). The score function is given by \( |\theta_2 - \theta_1| \).

In Appendix Section A.7 we show that auditing extended thresholds can be reduced to a special case of auditing if the slope of a two-dimensional linear classifier is zero. However, the challenge is that the explanations considered by linear classifier auditors do not consider \( g \) to be limited to \{0, 1\} and hence can be unrealistic for extended thresholds case. Therefore we propose simple auditors \( \text{AlgET}_c \) and \( \text{AlgET}_a \) for counterfactuals and anchors respectively where the explanations are realistic. \( \text{AlgET}_c \) exploits the fact that counterfactual explanation returned by the DS inadvertsely includes the thresholds and has a query complexity of two. \( \text{AlgET}_a \) is basically binary search and has a query complexity of \( [2 \log(\frac{1}{\epsilon})] \). As with linear classifiers, we find that worst-case anchors do not help. A detailed treatment of this section can be found in the Appendix Sections A.8 and A.9

| Algorithm | Query Complexity |
|-----------|------------------|
| Baseline  | \( O(\frac{1}{\epsilon} \log \frac{1}{\delta}) \) |
| \( \text{AlgLC}_c \) | 1 |
| \( \text{AlgLC}_a \) | \( O(d \log(\frac{2d}{\epsilon})) \) |

4 Experiments

In this section we conduct experiments on standard datasets to test some aspects of feature sensitivity auditing. Specifically, we answer the following questions:

1. Does anchor augmentation of typical anchors reduce query complexity for linear classifiers?
2. What is the average query complexity for auditing extended thresholds?

The datasets used in our experiments are - Adult Income\(^1\) and Covertype\(^2\). The features of interest are gender and wilderness area type for Adult Income and Covertype respectively. Details about the datasets can be found in the Appendix Section A.10.1.

4.1 Anchor Augmentation of Typical Anchors

As discussed in section 3.2.2 augmentation of worst-case anchors does not help in reducing the query complexity of auditing. In other words, augmenting worst-case anchors is equivalent to not using anchor explanations. A natural question then is - do typical anchors help? Since auditing with anchors using \( \text{AlgLC}_a \) means learning the DS’s model under the hood, reduction in query complexity of learning implies reduction in that of auditing. Hence, we check experimentally if faster learning is achieved through anchor augmentation of typical anchor points.

Methodology We learn a linear classifier with weights \( w \) for each dataset; these correspond to the DS’s model. Then the weights are estimated during auditing using Algorithm 6 We consider two different augmentations 1) worst-case anchor points which is equivalent to not using anchors 2) typical anchor points. We set the augmentation size (number of anchor points augmented) to a maximum of 30. Our anchors are hyperrectangles of a fixed volume surrounding the query point. We sample points with the same label as the query point from this hyperrectangle and augment to the set of queries in the typical case. Note that we have relaxed the \( \tau = 1 \) assumption here.

Results The results are shown in Figure 1. We see that the anchor augmentation of typical anchors drastically reduces the query complexity to achieve the same estimation error between weights as compared to not using anchors.

\(^1\)https://archive.ics.uci.edu/ml/datasets/adult
\(^2\)https://archive.ics.uci.edu/ml/datasets/covertype
(or equivalently using worst-case anchors). This saving directly translates to efficiency in auditing. For example, in the adults dataset, less than 50% of the queries are needed to achieve a lower error with typical anchors than without them. This illustrates that anchor explanations can be helpful for auditing, suggesting an application for these explanations.

Given input \( x = (x', g) \), we learn a feedforward NN to predict one of the features in \( x' \) using the remainder features of \( x' \). This NN corresponds to the function \( f \) in eq. 2. The idea of predicting one feature from the rest is inspired from Continuous-bag-of-words [Mikolov et al. (2013)] in NLP, wherein the target word is predicted through its context. Next we learn different thresholds corresponding to \( g \) on the aforementioned predicted feature. \( g \) is set to the feature of interest. Figure 5 in the Appendix illustrates our methodology for learning extended thresholds with NNs.

For any NN \( f \) learnt above, we randomly pick thresholds \( \theta_1, \theta_2 \) in the range of \( f \), to be treated as the DS’s model. Next we use Algorithm 8 and 7 to audit and estimate \( \theta_1, \theta_2 \). We keep track of the number of steps needed to audit and average them over multiple runs. Note that here we are considering two dimensional features, one being \( f(\cdot) \) and the other being the feature of interest.

**Results** Appendix tables 2 and 3 display the learned thresholds and test accuracy for various predicted features. Figure 2 displays average query complexity (AQC) results for the NN predicting education. For counterfactual explanations, AQC of auditing is fixed and equals two. For anchors, AQC decreases logarithmically with increasing difference in thresholds. This experiment confirms that the auditing problem is easier when the difference between the two thresholds is larger, i.e. when a lot of responsive pairs exist.

![Figure 1: Number of queries required to learn a linear classifier with anchor augmentation of worst-case anchor points (in blue) and typical anchor points (in orange). For the latter, a clear reduction in query complexity is observed.](image)

**4.2 Average Query Complexity of Extended Thresholds**

The main question we investigate in this section is - how many queries are needed on average for auditing extended thresholds? The number of queries changes based on the type of explanation method, difference between thresholds, and numeric value of the thresholds. Value of thresholds affects query complexity for anchor explanations since Algorithm 8 is a binary search algorithm.

**Methodology** We first propose a way to use Neural Networks (NNs) in the extended thresholds formalism. Next we utilize one of the learned NN to demonstrate how the average query complexity changes with difference in thresholds.

![Figure 2: Average query complexity (AQC) vs. difference in thresholds for auditing feature sensitivity in extended thresholds. For Counterfactuals, AQC is constant. For anchors, AQC decreases logarithmically with increasing difference in thresholds.](image)

**5 Discussion**

**Untruthful Data Scientist.** A natural question to ask is what happens when a DS is not entirely truthful in the auditing process, as we assumed in Section 2.1. What kind of auditing is possible in this case?
Suppose the DS returns all labels and explanations from an entirely different model than $\hat{h}$. In this case, there is no way for an auditor to detect it; but perhaps this can be dealt with in a procedural manner – like the DS hands its model to a trusted third party who answers the auditor’s queries.

What if the DS returns the correct labels but incorrect explanations? DS might be forced to return correct labels when the auditor has some labeled samples already and hence can catch the DS if it lies with labels. This scenario motivates verification of explanations. Next, we discuss schemes to verify anchors and counterfactuals for this scenario.

**Anchors.** For anchors, verification is possible if we have samples from the underlying distribution $D$. For a query $x$, DS returns an unverified anchor $A_x$ with precision $\tau_x$ and coverage $\epsilon_x$. The correctness of $\tau_x$ can be detected as follows. First, get an estimate for the true value of the precision parameter by sampling $n$ points from anchor $A_x$ according to $D$. Let the true and estimated values of precision parameter be $\tau$ and $\hat{\tau}$ respectively. $\hat{\tau}$ approaches $\tau$ with sufficiently large number of points as mentioned in lemma 1. Second, compare $\tau_x$ with $\hat{\tau}$. A large difference between $\tau_x$ and $\hat{\tau}$ implies false anchors.

**Lemma 1.** For any $\Delta > 0$ and integer $n$, $\Pr(|\tau - \hat{\tau}| \geq \Delta) \leq 2 \exp^{-2\Delta^2 n}$.

Lemma 1 is immediate from Hoeffding’s Inequality. Notice that the number of samples $n$ required to verify precision changes with $1/\Delta^2$. If the fraction of responsive pairs $\epsilon$ equals $\Delta$, our baseline in section 3.1 can audit with $O(1/\Delta)$ samples without using explanations. Hence, in adversarial conditions where the probability of a lying DS is high, it is better to audit with our explanations-free baseline than auditing with anchors and verifying them.

Since coverage is the probability that a point sampled from $D$ belongs to $A_x$, it can be easily checked 1) by calculating the volume of $A_x$ when all dimensions of $x$ are bounded or 2) by sampling points from $D$ when the features are unbounded.

**Counterfactuals.** Given $x$, let $x'$ be the unverified counterfactual explanation returned by the DS. There are two aspects to a counterfactual explanation – its label which should be different from $x$ and it should be the closest such point to $x$. The first aspect can be easily verified by querying $x'$ from the DS. For the second aspect, we observe that finding the counterfactual is equivalent to finding the closest adversarial point. Deriving from adversarial robustness literature, verifying the closeness access can be a computationally hard problem for some hypothesis classes like discussed in Weng et al. (2018). However, we propose a sampling based algorithm to estimate the true counterfactual, assuming that the DS is lying. Firstly, sample points from the ball $B(x, d(x, x'))$. For $x'$ to be the true counterfactual, all points within the ball $B(x, d(x, x'))$ should have the same label as $x$. If a point $x''$ with a different label is sampled, select this point as an estimate of the true counterfactual and repeat the scheme with the new ball $B(x, d(x, x''))$. By following this procedure iteratively, we get closer to the correct counterfactual explanation as the radius of the ball reduces at each iteration. There are cases where our algorithm may not work well. We leave designing better algorithms for verifying closeness to future work.

Upon verification, if auditor finds that the DS is untruthful, it can choose to 1) stop auditing and declare that the DS is lying, 2) audit with estimated explanations or 3) audit with explanations-free baseline algorithm (section 3.1) since option 2 is computationally intensive.

**Privacy.** Auditing also raises a legitimate privacy concern. On one hand, model is hidden from the auditor for confidentiality reasons while on the other hand, to be able to give a correct auditing decision efficiently, the auditor has to extract/learn the hidden model, albeit partially. This concern can be resolved by using cryptographic tools like Zero-Knowledge proofs (Singh et al., 2021). A future direction of our work is modifying our framework to work with cryptographic tools.

## 6 Conclusions and Future Work

To summarize, we propose a general learning-theoretic framework for certified auditing, which is a major requirement for responsible ML. We show how explanations can be incorporated into our auditing framework and therefore present auditing as a use-case for existing explanation methods. We instantiate our framework through auditing feature sensitivity for linear classifiers and provide two auditing algorithms based on different explanations. We show that while counterfactual explanations greatly bring down the query complexity of auditing, anchor explanations help auditing in the typical case, but not in the worst case.

We believe that this work is a first step towards understanding certified auditing and much needs to be done before we can design useful certified auditors for complex machine learning models and properties. Some fruitful directions include designing certified auditors for other auditing properties and hypothesis classes, as well as incorporating other kinds of explanations or different kinds of information into the auditing process.

**Acknowledgements** This work was supported by NSF under CNS 1804829 and ARO MURI W911NF2110317. MM has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation (grant number 993/17), Tel Aviv University Center for AI and Data Sci-
ence (TAD), and the Yandex Initiative for Machine Learning at Tel Aviv University. CY thanks Geelon So for providing valuable feedback on an early draft of the paper.

References

J. Adebayo and M. Gorelick. Fairml: Auditing black-box predictive models. *Blog. Fastforwardlabs. Com.*, 2017.

G. Aini et al. A summary of the research on the judicial application of artificial intelligence. *Chinese Studies, 9* (01):14, 2020.

I. Alabdulmohsin, X. Gao, and X. Zhang. Efficient active learning of halfspaces via query synthesis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015*.

D. Angluin. Queries and concept learning. *Machine learning, 2*(4):319–342, 1988.

M.-F. Balcan, E. Blais, A. Blum, and L. Yang. Active property testing. *arXiv preprint arXiv:1111.0897*, 2011.

U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Pur, J. M. Moura, and P. Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 648–657, 2020.

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

D. V. Carvalho, E. M. Pereira, and J. S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics, 8*(8):832, 2019.

I. Castiglioni, L. Rundo, M. Codari, G. Di Leo, C. Salvatore, M. Interlenghi, F. Gallivanone, A. Cozzi, N. C. D’Amico, and F. Sardanelli. Ai applications to medical images: From machine learning to deep learning. *Physica Medica, 83*:9–24, 2021.

S. Dasgupta. Coarse sample complexity bounds for active learning. *Advances in neural information processing systems, 18*, 2005.

S. Dasgupta, N. Frost, and M. Moshkovitz. Framework for evaluating faithfulness of local explanations. *arXiv preprint arXiv:2202.00734*, 2022.

V. Feldman. On the power of membership queries in agnostic learning. *The Journal of Machine Learning Research, 10*:163–182, 2009.

T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford. Datasheets for datasets. *arXiv preprint arXiv:1802.09010*, 2018.

S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences, 50*(1):20–31, 1995.

S. Goldwasser, G. N. Rothblum, J. Shafer, and A. Yehudayoff. Interactive proofs for verifying machine learning. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

L. Hamelers. Detecting and explaining potential financial fraud cases in invoice data with machine learning. Master’s thesis, University of Twente, 2021.

Y. Huang, X. Li, and K. Li. Ema: Auditing data removal from trained models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 793–803. Springer, 2021.

M. Jagielski, J. Ullman, and A. Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems, 33*:22205–22216, 2020.

A. Koenecke, A. Nam, E. Lake, J. Nudell, M. Quartey, Z. Mengesha, C. Toups, J. R. Rickford, D. Jurafsky, and S. Goel. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences, 117*(14):7684–7689, 2020.

T. Liu, X. Xie, and Y. Zhang. Zkcn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985, 2021.

X. Liu and S. A. Tsaftaris. Have you forgotten? a method to assess if machine learning models have forgotten data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 95–105. Springer, 2020.

T. Mikolov, K. Chen, G. Corrada, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

L. Oala, J. Fehr, L. Gilli, P. Balachandran, A. W. Leite, S. Calderon-Ramirez, D. X. Li, G. Nobis, E. A. M. Alvarado, G. Jaramillo-Gutierrez, et al. Ml4h auditing: From paper to practice. In *Machine learning for health*, pages 280–317. PMLR, 2020.

C. M. Poland. The right tool for the job: Open-source auditing tools in machine learning. *arXiv preprint arXiv:2206.10613*, 2022.

I. D. Raji, A. Smart, R. N. White, M. Mitchell, T. Gebru, B. Hutchinson, J. Smith-Loud, D. Theron, and P. Barnes.
Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In Proceedings of the 2020 conference on fairness, accountability, and transparency, pages 33–44, 2020.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

N. Singh, P. Dayama, and V. Pandit. Zero knowledge proofs towards verifiable decentralized ai pipelines. Cryptology ePrint Archive, 2021.

R. Tatman. Gender and dialect bias in youtube’s automatic captions. In Proceedings of the First ACL Workshop on Ethics in Natural Language Processing, pages 53–59, 2017.

R. Tatman and C. Kasten. Effects of talker dialect, gender & race on accuracy of bing speech and youtube automatic captions. In Interspeech, pages 934–938, 2017.

S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. Harv. JL & Tech., 31:841, 2017.

D. S. Watson and L. Floridi. The explanation game: a formal framework for interpretable machine learning. In Ethics, Governance, and Policies in Artificial Intelligence, pages 185–219. Springer, 2021.

L. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon. Towards fast computation of certified robustness for relu networks. In International Conference on Machine Learning, pages 5276–5285. PMLR, 2018.

T. Yan and C. Zhang. Active fairness auditing. In International Conference on Machine Learning, pages 24929–24962. PMLR, 2022.

J. Ye, A. Maddi, S. K. Murakonda, and R. Shokri. Privacy auditing of machine learning using membership inference attacks. 2021.

C. A. Zhang, S. Cho, and M. Vasarhelyi. Explainable artificial intelligence (xai) in auditing. International Journal of Accounting Information Systems, page 100572, 2022.
A Appendix

A.1 General Auditor

In section 2.3, we discussed how to construct a general auditor. Next, we outline the same in Algorithm 1.

Algorithm 1 A General Auditor

1: \( S_0 := \emptyset; \) stop_flag := False; \( t := 1; \epsilon := \text{eps} \)
2: while \(!\text{stop\_flag}\) do
3: \( x_t := \text{picking\_next\_query}(S_{t-1}) \)
4: Auditor receives label \( y_t \) and explanation \( e_t \) from the DS
5: \( S_{t+1} := \text{update\_search\_space}(S_t, x_t, y_t, e_t) \)
6: \( Y_{\emptyset}, \text{stop\_flag} := \text{check\_stopping\_condition}(S_t) \)
7: end while
8: return \( Y_{\emptyset} \)

Algorithm 2 picking\_next\_query()

1: \( S_t \)
2: for \( x \in X \) do
3: for \((e, y) \in E \times Y\) do
4: \( S_{t+1} := \text{update\_search\_space}(S_t, x, y, e) \)
5: end for
6: value \( x := \min_{e \times y} |S_t|/|S_{t+1}| \)
7: end for
8: return \( \arg\max_{x \in X} \text{value} \)

Algorithm 3 update\_search\_space()

1: \( S_{\text{new}}, x, y, e \)
2: \( S_{\text{new}} := \{h \in S | h(x) = y, \text{check\_consistent\_explanation}(x, e_h(x), e)\} \)
3: return \( S_{\text{new}} \)

Algorithm 4 check\_stopping\_condition()

1: \( S_t \)
2: if \( \forall h \in S_t \) \( > \epsilon \text{eps} \) then
3: decision := \text{Yes}, stop\_flag := True
4: else if \( \forall h \in S_t \leq \epsilon \text{eps} \) then
5: decision := \text{No}, stop\_flag := True
6: else
7: decision := \text{None}, stop\_flag := False
8: end if
9: return decision, \!\text{stop\_flag}

Next we prove that Algorithm 1 is an \((\epsilon, \delta)\)-auditor next.

Theorem 2.1. Algorithm 1 is an \((\epsilon, \delta)\)-auditor.

Proof. Firstly, note that \( \bar{h} \) is in the search space \( S_t \) at all times \( t \). This is because the search space is reduced based on labels and explanations w.r.t. \( \bar{h} \) (provided by the data scientist).

Next, we assume that value of function \( s \) can be computed to arbitrary precision for any hypothesis \( h \). Then the proposed stopping conditions satisfy the soundness and completeness properties of the \((\epsilon, \delta)\)-auditor from definition.

A.2 Similarities and Differences from Active Learning

To formalize the connection between active learning and auditing, we observe that learning is a harder task than auditing and if there exists an algorithm that can learn a hypothesis, it can also audit it. The following theorem states this connection.

Theorem 2.2. If there exists a membership query active learner that can learn \( \bar{h} \) exactly in \( T \) queries, then the active learner can also audit in \( T \) queries.

Proof. Once \( \bar{h} \) is known, \( s(\bar{h}) \) returns the auditing decision. If \( s(\bar{h}) = 0 \), the decision is \text{No} and if \( s(\bar{h}) > \epsilon \), the decision is \text{Yes}.

A.3 A Simple Baseline: Random Testing

A simple algorithm to detect feature sensitivity is to query a large number of pairs at random from the DS. If any of them is a responsive pair, auditor knows that the feature is sensitive. The following theorem states a bound on the query complexity for this algorithm.

Theorem 3.1. For \( \epsilon, \delta \in (0, 1) \), auditor is an \((\epsilon, \delta)\)-auditor if it queries \( m \geq \frac{1}{\epsilon} \log \frac{1}{\delta} \) pairs.

Proof. Let the set of all pairs be denoted by \( P \). Assume the fraction of responsive pairs is at least \( \epsilon \). Then the probability that a randomly drawn pair is not a responsive pair is given by \( \Pr_{p_i \sim P}(p_{ij} \text{ is not a responsive pair}) \leq 1 - \epsilon \). If \( m \) pairs are drawn randomly, then the probability that all of the pairs are not responsive is bounded by \( (1 - \epsilon)^m \leq \exp(-em) \leq \delta \), by the choice of \( m \).

A.4 Connection between Model Parameters and Score Function

Notation. For vector \( v \), the \( i^{\text{th}} \) feature is denoted \( v_i \). Let hypothesis \( h_{w,b} \in \mathcal{H}_{LC} \). When \( w, b \) are clear from the context, we simply write \( h \). Let \( w' \) be the \((d-1)\)-dimensional vector \([w_1 \ldots w_{d-1}]^T\). Hence \( w \) is a concatenation of \( w' \) and \( w_d \), denoted by the shorthand \( w = [w', w_d] \). We assume that \( \|w'\|_2 = 1 \).

Let \( x \) be a \( d \)-dimensional input to this hypothesis. Let \( x' \) be the \((d-1)\)-dimensional vector \([x_1 \ldots x_{d-1}]^T \). Let \( \bar{x} = [x_1 \ldots x_{d-1}, 1]^T \). Without loss of generality, let the
The feature of interest and let $x_d \in \{0, 1\}$. The score function for a hypothesis $h$ is given as, $s(h) = \Pr \left( (x^i, x^j) \text{ forms a responsive pair} \right)$ where $(x^i, x^j)$ is sampled uniformly from the set of all pairs and labeled by $h$. Henceforth we use this score function.

In the following theorem, we bound the fraction of responsive pairs, also our score function, using the weight of our feature of interest, $w_d$. The score function is bounded by $c \cdot |w_d|$ where $c$ depends on the dimension of the input. This implies that if $|w_d|$ is small, there are not a lot of responsive pairs (low score function value).

**Theorem 1.** Assume $\forall x \in \mathcal{X}, \|x\|_2 \leq 1$. Let $h_{[w', w_d], h} \in \mathcal{H}_{LC}$. Then 

$$s(h) \leq C \cdot |w_d|$$

(3)

where $C = \frac{2^{d-2}}{\pi^{(\frac{d}{2}-1)}}\Gamma(\frac{d}{2})$ is a constant for finite dimension $d$ and $\Gamma$ is Euler’s Gamma function.

**Proof.** Let $P$ be the set of all pairs of points. Let $x^i$ and $x^j$ denote two inputs forming a pair. Let the pair $(x^i, x^j)$ drawn uniformly from $P$ form a responsive pair.

From the definition of a pair, $x^i$ and $x^j$ only differ in the $d$th feature. Hence, 

$$w^T x^j + b = w'^T x^j + b + w_d x_d^j$$

$$= w'^T x^i + b + w_d (1 - x_d^i)$$

$$= w'^T x^i + b + w_d - w_d x_d^j.$$ 

(4)

Without loss of generality, let $x_d^j = 0$; Hence $x_d^i = 1$. Therefore, 

$$w^T x^j + b = w'^T x^i + b + w_d$$ 

(5)

Next, writing the definition of a responsive pair for $\mathcal{H}_{LC}$ we get, 

$$\text{sign } (w^T x^i + b) \neq \text{sign } (w'^T x^j + b)$$ 

(6)

Substituting eq. (5) into the RHS of eq. (6) we get, 

$$\text{sign } (w^T x^i + b) \neq \text{sign } (w'^T x^i + b + w_d)$$ 

(7)

Expanding the LHS of eq. (7) and substituting $x_d^i = 0$, we get, 

$$\text{sign } (w'^T x^i + b) \neq \text{sign } (w'^T x^i + b + w_d)$$ 

(8)

Eq. (8) implies the following, 

$$0 \leq w^T x^i + b \Rightarrow w'^T x^i + b + w_d < 0$$

$$0 > w^T x^i + b \Rightarrow w'^T x^i + b + w_d \geq 0$$ 

(9)

Combining the two equations in eq. (9) we get, 

$$0 \leq w'^T x^i + b < -w_d$$

$$0 < - (w'^T x^i + b) \leq w_d$$ 

(10)

Note that the model (defined by $w, b$) is fixed. Hence the variables in the above conditions are the inputs $x$.

Note that only one of the conditions in eq. (10) can be satisfied at any time, based on whether $w_d \geq 0$ or $w_d < 0$. The fraction of the inputs which satisfy one of the above conditions correspond to the fraction of responsive pairs and hence is the value of the score function.

Conditions in eq. (10) correspond to intersecting halfspaces formed by parallel hyperplanes. If $\|\vec{x}\|_2 \leq r$, the region of intersection can be upper bounded by a hypercuboid of length $2r$ in $d - 2$ dimensions and perpendicular length between the two hyperplanes $l = \frac{|w_d|}{\|w'\|_2}$ in the $(d - 1)$-th dimension.

Hence, we can upper bound score function $s(h)$ as, 

$$s(h) \leq \frac{(2r)^{d-2} \cdot l}{V_{d-1}(r)}$$ 

(11)

where $l = \frac{|w_d|}{\|w'\|_2}$ and $V_{d-1}(r)$ is the volume of the $(d-1)$-dimensional ball given by $\frac{(2r)^{(d-1)/2}}{\Gamma \left( \frac{d-1}{2} \right)} r^{d-1}$ and $\Gamma$ is Euler’s Gamma function.

Upon simplification we get, 

$$s(h) \leq \frac{2^{d-2}}{\pi^{(\frac{d}{2}-1)}} \frac{l}{r \Gamma \left( \frac{d-1}{2} \right)}$$ 

(12)

Assuming $r = 1$ and $\|w'\|_2 = 1$, we can write eq. (12) as, 

$$s(h) \leq C \cdot |w_d|$$ 

(13)

where $C = \frac{2^{d-2}}{\pi^{(\frac{d}{2}-1)}}\Gamma(\frac{d}{2})$ is a constant for small dimensions.

**A.5 Auditing Linear Classifiers with Counterfactual Explanations**

In this section, we will prove that auditing linear classifiers using counterfactual explanations requires only one query. We denote our auditor by AlgLC, as outlined in alg. 5.

The proof goes by noting that the counterfactual explanation $x'$ returned by the DS is very close to the projection of input $x$ and that $x - x'$ is parallel to $w$. We consider the $d$-th feature to be our feature of interest without loss of generality.

**Lemma 2.** Given hyperplane $w^T x + b = 0$, point $x$ and its projection on the hyperplane $x''$, $x - x'' = \lambda w$ where $\lambda = \frac{w^T x + b}{\|w\|^2}$. 


Taking derivative of the lagrangian with respect to $x$ equating with zero we get,

$$\min_{x''} \|x'' - x\|^2 \quad \text{s.t.} \quad w^T x'' + b = 0$$  \hspace{1cm} (14)

Let $L(x'',\lambda)$ be the lagrangian for the above optimization problem.

$$L(x'',\lambda) = \|x'' - x\|^2 + 2\lambda(w^T x'' + b)$$

$$= \|x''\|^2 + \|x\|^2 - 2x^T x'' + 2\lambda w^T x'' + 2\lambda b$$

$$= \|x''\|^2 + \|x\|^2 - 2(x - \lambda w)^T x'' + 2\lambda b$$  \hspace{1cm} (15)

Taking derivative of the lagrangian with respect to $x''$ and equating with zero we get,

$$\frac{\partial L}{\partial x''} = 2x'' - 2(x - \lambda w) = 0$$

$$x - x'' = \lambda w$$  \hspace{1cm} (16)

By substituting above equation in the constraint for the optimization problem $w^T x'' + b = 0$, we get $\lambda = \frac{w^T x + b}{\|w\|^2}$.

**Theorem 3.2.** For any $\epsilon \in [0,1]$, auditor $A_{LC_c}$ is an $(\epsilon,0)$-auditor for $H_{LC}$ and score function $s(\cdot)$ with $T = 1$ query.

**Proof.** Recall that the counterfactual explanation returned by the DS for input $x$ is given as $x' = \arg\min_{x':h(x') \neq h(x)} d(x,x')$ where $d(x,x') = \|x - x'\|_2$.

The projection of $x$ on the hyperplane, $x''$ is the closest point to $x$ on the hyperplane. Therefore $x' = x'' + \Delta$ where $\Delta$ is a vector in the direction of $w$, $\Delta = \gamma w$, $\gamma$ is a very small non-zero constant.

Therefore, $x - x' = x - (x'' + \Delta) = (x - x'') + \Delta$.

Using lemma 2

$$\hat{w} := x - x' = \lambda w + \gamma w = c_0 w,$$  \hspace{1cm} (17)

where $c_0$ is a non-zero constant. ($x - x'$ is non-zero due to the definition of counterfactuals, specifically that they have different labels.)

If $w_d = 0$, then it implies that $\hat{w}_d = 0$ and the feature has no effect on the prediction. Thus the score function is zero. Since $A_{LC_c}$ returns a No when $\hat{w}_d = 0$, it is always correct in this case. For all the other cases when $w_d \neq 0$, it implies that $\hat{w}_d \neq 0$ and therefore, the feature has an effect on the prediction. Since $A_{LC_c}$ returns a Yes when $\hat{w}_d \neq 0$, it is always correct.

Also $\delta = 0$ since our auditor and DS are deterministic. $\square$

Note that this is partial learning since 1) we do not need to learn $w$ exactly and 2) we do not need to learn the bias term $b$.

### A.6 Auditing Linear Classifiers with Anchor Explanations

[Alabdulmohsin et al. (2015)] proposed a query synthesis spectral algorithm to learn homogeneous linear classifiers in $O(d \log \frac{1}{\delta})$ steps where $\Delta$ corresponds to a bound on the error between estimated and true classifier. They maintain a version space of consistent hypotheses approximated using the largest ellipsoid $\Sigma^* = (\mu^*, \Sigma^*)$ where $\mu^*$ is the center and $\Sigma^*$ is the covariance matrix of the ellipsoid. They prove that the optimal query which halves the version space is orthogonal to $\mu^*$ and maximize the projection in the direction of the eigenvectors of $\Sigma^*$.

We propose an auditor $A_{LC_a}$ as depicted in alg. using their algorithm. The anchor explanations are incorporated through anchor augmentation. But, in the worst-case anchors are not helpful and hence the algorithm reduces essentially to that of [Alabdulmohsin et al. (2015)] (without anchors). In this section we find the query complexity of this auditor.

**Notation** In $A_{LC_a}$, $\mathcal{E}^* = (\mu^*, \Sigma^*)$ denotes the largest ellipsoid that approximates the version space (corresponds to search space in our case) at time $t$ where $\mu^*$ is the center and $\Sigma^*$ is the covariance matrix of the ellipsoid at time $t$. $N^t$ is the orthonormal basis of the orthogonal complement of $\mu^*$. $\alpha^*$ is the top eigenvector of the matrix $N^T \Sigma^* N^t$. In the implementation by [Alabdulmohsin et al. (2015)] some warm-up labeled points are supplied by the user, we denote this set as $W$. Let $W(t)$ denote the $t$-th element of this set. Let the $d^{th}$ feature be the feature of interest without loss of generality.

**Worst-case anchors** have the same set of consistent hypotheses as the input query and therefore, do not cut down the search space of hypotheses by any more amount. This is formally written below.

**Lemma 3.** Given input $x$, a worst-case anchor for $A_{LC_a}$ is of the form $A_x = \{x|\lambda \in \mathcal{R}^+\}$ with preci-
A Learning-Theoretic Framework for Certified Auditing with Explanations

Algorithm 6 AlgLC_a: Auditing Linear Classifiers using Anchors
1: Input: T, augmentation size s, set of warm-up labeled points W
2: set of queried points Q := ∅, l := size(W)
3: for t = 1, 2, 3, ..., T + l do
4: if t ≤ l then
5: (x, y) := W(t)
6: Q ← Q ∪ (x, y)
7: goto step[15]
8: else
9: Query point x^t := N^t a^t from the DS
10: Auditor receives label y and explanation A_x from the DS
11: Q ← Q ∪ (x^t, y)
12: Sample randomly s points x_1^t ... x_s^t from A_x
13: Q ← Q ∪ {(x_1^t, y) ... (x_s^t, y)}
14: end if
15: \( \varepsilon^{(t+1)*} = (\mu^{(t+1)*} , \Sigma^{(t+1)*}) := \text{estimate}\_\text{ellipsoid}(Q) \)
16: \( N^{t+1} := \text{update}_N(\mu^{(t+1)*}) \)
17: \( a^{(t+1)*} := \text{update}_alpha(N^{t+1}, \Sigma^{(t+1)*}) \)
18: \{\text{Exact formulae for steps 15, 16 and 17 can be found in Alabdulmohsin et al. (2015)}\}
19: end for
20: \( \hat{w} = \mu^{T+1} \)
21: if \( |\hat{w}| \leq \Delta \) then
22: return No
23: else
24: return Yes
25: end if

Theorem 3.3. For every dimension d, there exists \( c > 0 \) such that for any \( \epsilon \in (0,1) \), auditor AlgLC_a is an \( (\epsilon, 0) \)-auditor for \( H_{LC} \) and score function \( s(\cdot) \) with \( T = O(\log \frac{2c}{\epsilon}) \) queries.

Proof. Let \( w \) be the true classifier and \( \hat{w} \) be the estimated classifier learnt by AlgLC_a.

Let the difference between \( w \) and \( \hat{w} \) be bounded by \( \Delta \) as follows,

\[
\|w - \hat{w}\|_2 \leq \Delta. \quad (19)
\]

The value of \( \Delta \) will be set later on. Since AlgLC_a uses \( |\hat{w}| \) to make its decision, the worst case is when the entire error in estimation is on the \( d^{th} \) dimension. Hence, we consider \( |w_d - \hat{w}_d| \leq \Delta \).

To guarantee that the auditor is an \( (\epsilon, 0) \)-auditor we need to verify for every hypothesis in the class that if \( s(\cdot) = 0 \), then the answer is No and if \( s(\cdot) > \epsilon \), then the answer is Yes, see def. 5.

Importantly, \( s(w) \) is zero only if \( w_d = 0 \) from theorem 1. If \( w_d = 0 \Rightarrow |\hat{w}_d| \leq \Delta \), by eq. 19. Since AlgLC_a returns a No for \( |\hat{w}_d| \leq \Delta \), AlgLC_a satisfies def. 5 when \( s(w) = 0 \) with \( \delta = 0 \).

Next, we have the case \( s(w) > \epsilon \) when auditor should return a Yes with high probability. AlgLC_a returns a Yes
when $|\hat{w}_d| > \Delta$. Hence for AlgLC$_a$ to be correct, we need that $s(w) > \epsilon$ imply that $|\hat{w}_d| > \Delta$.

We can upper bound $s(w)$ using eq. [12] as,

$$s(w) \leq C \cdot |w_d|$$

(20)

Since $\epsilon < s(w)$,

$$\epsilon < C \cdot |w_d|$$

(21)

Since $|w_d - \hat{w}_d| \leq \Delta$,

$$\epsilon \leq C (|\hat{w}_d| + \Delta)$$

(22)

On rearranging,

$$\frac{\epsilon}{C} - \Delta \leq |\hat{w}_d|$$

(23)

Eq. [23] connects $\epsilon$ with $\Delta$ and $\hat{w}_d$. For $s(w) > \epsilon$, $|\hat{w}_d|$ should be greater than $\Delta$ for AlgLC$_a$ to be correct. Hence, lower bounding the LHS of eq. [23] we get,

$$\Delta \leq \frac{\epsilon}{C} - \Delta$$

(24)

We set $\Delta = \frac{\epsilon}{2C}$.

From Lemma 2, AlgLC$_a$ reduces to Abdulmohsin et al. (2015)'s spectral algorithm in the worst-case. This algorithm has a bound of $O(d \log(\frac{1}{\epsilon}))$. Hence, the query complexity of AlgLC$_a$ is $O(d \log(\frac{2C}{\epsilon}))$.

**A.7 Extended Thresholds as Linear Classifiers**

According to the definition of linear classifiers, output is 1 if $w^T x + b \geq 0$ and -1 otherwise. Writing the if condition of extended thresholds in terms of linear classifiers, we get,

$$(g - 1)\theta_1 - g\theta_2 + f(x') \geq 0$$

$$\iff (\theta_1 - \theta_2)g + f(x') + -\theta_1 \geq 0$$

$$\iff w = [1, \theta_1 - \theta_2]^T, b = -\theta_1$$

Thus, extended thresholds are 2D linear classifiers with weights and bias as given above.

**A.8 Auditing Extended Thresholds with Counterfactual Explanations**

We propose AlgET$_c$ to audit extended thresholds with counterfactual explanations. Auditor queries $(l,0)$ and $(u,1)$ from the DS where the first feature corresponds to the range of $f(\cdot)$ and the second feature corresponds to the threshold-deciding feature or the feature of interest (they're

```
Algorithm 7 AlgET$_c$ : Auditing Extended Thresholds using Counterfactuals

1: Query points $(l,0)$ and $(u,1)$ from the DS
2: Auditor receives labels $y_0, y_1$ and explanations $(\theta_1,0), (\theta_2,1)$ from the DS
3: if $\theta_1 = \theta_2$ then
4: return No
5: else
6: return Yes
7: end if
```

the same). DS responds with threshold parameters of $\bar{h}$ inadvertently as $(\bar{h},0)$ and $(\bar{h},1)$ are the counterfactual explanations for the asked queries.

**Theorem 2.** For any $\epsilon \in [0,1]$, auditor AlgET$_c$ is an $(\epsilon,0)$-auditor for $H_{ET}$ and score function $s(\cdot)$ with $T = 2$ query.

**Proof.** Without loss of generality, let $l = -1$ and $u = 1$. Then $(-1,0)$ and $(1,1)$ are the two queries. We also consider computers with finite precision. Hence for example, if the point 0.999999 . . . is saved as 1 by the computer.

Let $\theta_1, \theta_2$ correspond to thresholds for $\bar{h}$. Next we prove that the counterfactual explanations returned by the DS for the queried points have to be $(\theta_1,0), (\theta_2,1)$.

If $\theta_1 \leq \theta_2$, $(\theta_1, 0)$ is the closest point to $(-1,0)$ with a different label and is therefore the counterfactual explanation. Note that $(\theta_1, 1)$ cannot be the counterfactual since changing the second feature would add a cost of 1 to that of $(\theta_1, 0)$. Similarly, $(\theta_2, 1)$ is the closest point to $(1,1)$. Hence $(\theta_1,0)$ and $(\theta_2,1)$ are returned as explanations by the DS.

If $\theta_1 > \theta_2$ and $|\theta_1 - \theta_2| < 1$, $(\theta_1,0)$ is the counterfactual for $(-1,0)$ and $(\theta_2,1)$ is the counterfactual for $(1,1)$.

If $\theta_1 > \theta_2$ and $|\theta_1 - \theta_2| \geq 1$, $(\theta_2,1)$ is the counterfactual for $(-1,0)$ and $(\theta_1,0)$ is the counterfactual for $(1,1)$.

In all the cases, DS returns $(\theta_1,0)$ and $(\theta_2,1)$ as the explanations. This allows the auditor to exactly know $\bar{h}$ after querying 2 points. Responsive pairs exist between the thresholds. Hence if the thresholds are equal, it is implied that no responsive pairs exist and therefore feature of interest is not a sensitive feature for the model. The opposite case holds when the thresholds are not equal. Then, $\epsilon = \frac{|\theta_2 - \theta_1|}{u - l}$ and $\delta = 0$ since the auditor and DS are deterministic in this case.

**A.9 Auditing Extended Thresholds with Anchor Explanations**

In this section, we propose AlgET$_a$ with guarantees to audit extended thresholds with anchor explanations.

In our case, anchor hyperrectangles correspond to intervals in $[l, u]$. Note that anchor explanations can only be provided in terms of $f(x')$, otherwise the auditing decision is
clear. This restriction does not apply to counterfactual explanations from definition - they are points with the same of features as the input.

Let us assume $l = -1, u = 1$ without loss of generality. Let $\tau = 1$ and fix coverage length $c$ for simplicity.

Auditor $\text{AlgET}_a$ is as follows. Let search space at time $t$ correspond to the interval $[\theta_{\text{min}}, \theta_{\text{max}}]$. Auditor queries the mid points $((\theta_{\text{min}} + \theta_{\text{max}})/2)$ from the DS, where the first feature corresponds to $f(\cdot)$ and second feature corresponds to the threshold-deciding feature or the feature of interest (they’re the same). These queries halve the size of the search space, which is the maximum amount in the worst case. The worst case expansion is the interval between $(\theta_{\text{min}} + \theta_{\text{max}})/2$ and $(\theta_{\text{min}} + \theta_{\text{max}})/2$ where $-c$ is for label -1 and $+c$ for label 1. Note that these explanations do not provide extra information as they do not reduce the search space by any more amount than labels. Based on the output of the DS, auditor updates $\theta_{\text{min}}$ and $\theta_{\text{max}}$. Auditing stops when a responsive pair is detected or if the auditing budget is over. If a responsive pair is detected, the auditor returns $Y_a = \text{Yes}$ otherwise it returns $Y_a = \text{No}$. This algorithm is demonstrated in algorithm$^{[8]}$.

Next we will prove that the query complexity of $\text{AlgET}_a$ is $[2\log(\frac{c}{\tau})]$ as stated in theorem$^3$ using lemmas$^4$ and$^5$.

Algorithm 8 $\text{AlgET}_a$: Auditing Extended Thresholds using Anchors

1: Input: $T$
2: $bp_{\text{detected}} := \text{No}, \theta_{\text{min}} := -1, \theta_{\text{max}} := 1$
3: for $t = 1, 2, 3, \ldots, T$ do
4: Query points $(\theta_{\text{min}} + \theta_{\text{max}})/2, 0)$ and $(\theta_{\text{min}} + \theta_{\text{max}})/2, 1)$
5: Auditor receives labels $y_0, y_1$ and explanations $e_0, e_1$ from the DS
6: if $y_0 \neq y_1$ then
7: $bp_{\text{detected}} = \text{Yes}$
8: break
9: end if
10: if $y_0 = -1$ then
11: $\theta_{\text{min}} := \theta_{\text{min}} + \theta_{\text{max}}$
12: $\theta_{\text{max}} := \theta_{\text{min}} + \theta_{\text{max}}$
13: else
14: $\theta_{\text{min}} := \theta_{\text{min}} + \theta_{\text{max}}$
15: $\theta_{\text{max}} := \theta_{\text{max}}$
16: end if
17: end for
18: return $bp_{\text{detected}}$

Lemma 4. At time $t$, $h \in \mathcal{S}_t$.  

Proof: This is true from the definition of the search space and $h \in \mathcal{H}$ assumption. □

$\theta_{1h}, \theta_{2h}$ are the thresholds $\theta_1, \theta_2$ for hypothesis $h$. At each time step, the search space is reduced by half. $\forall h \in \mathcal{H}_t$, $\theta_1$ and $\theta_2$ lie within an interval which narrows down at each time step. Responsive pairs exist within this interval. Hence, the probability of responsive pairs also reduces subsequently. This intuition is formalized below.

Lemma 5. At time $t$, $\exists \theta_{\text{min}}, \theta_{\text{max}} \in [-1, 1]$ such that, $\forall h \in \mathcal{S}_t$

1. $\theta_{1h}, \theta_{2h} \in [\theta_{\text{min}}, \theta_{\text{max}}]$
2. $|\theta_{1h} - \theta_{2h}| \leq 2^{t-1}$

Proof. At time $t = 0, \mathcal{S}_0 = \mathcal{H}$.

$\forall h \in \mathcal{S}_0, \theta_{1h}, \theta_{2h} \in [-1, 1]$ and $\theta_{1h} - \theta_{2h} \leq 2$. Hence, $\theta_{\text{min}} = -1$ and $\theta_{\text{max}} = 1$.

Inductive Step: Consider at time $t$, the lemma holds.

At time $t$, the auditor picks the queries $((\theta_{\text{min}} + \theta_{\text{max}})/2, \cdot)$. Without loss of generality, using worst-case label = -1, $\theta_{\text{min}} = \theta_{\text{min}} + \theta_{\text{max}}/2$.

$\forall h \in \mathcal{H}$, $\theta_{1h}, \theta_{2h} \in [\theta_{\text{min}}, (\theta_{\text{min}} + \theta_{\text{max}})/4]$ or $\theta_{1h}, \theta_{2h} \in [(\theta_{\text{min}} + \theta_{\text{max}})/4, \theta_{\text{max}}].$

Hence $\theta_{1h}, \theta_{2h} \in [\theta_{\text{min}}, \theta_{\text{max}}].$

$|\theta_{1h} - \theta_{2h}| \leq \max(|\theta_{1h} - \theta_{2h}|) = |\theta_{\text{min}} - \theta_{\text{max}}| = 2^{-t}$

Theorem 3. For any $\epsilon \in [0, 1]$, auditor $\text{AlgET}_a$ is an $(\epsilon, 0)$ auditor for $\mathcal{H}_{ET}$ and score function $s(\cdot)$ with $T = (2\log(\frac{c}{\tau}))$ queries.

Proof. At time $t$, responsive pairs exist between $\theta_{\text{min}}$ and $\theta_{\text{max}}$. Since each pair corresponds to 2 queries and using lemma$^4$ and$^5$ we get theorem$^3$.

A.10 Experiments

A.10.1 Datasets

Both Adult and Covertype have a mix of categorical and continuous features. Categorical features are processed such that each category corresponds to a binary feature in itself. For Adult dataset, the output variable is whether Income exceeds $500K/yr$. For Covertype, the output variable is whether forest covertype is category 1 or not.

A.10.2 Auditing Neural Networks using Extended Thresholds

| Feature predicted by NN | $(\theta_1, \theta_2)$ | Test Accuracy % |
|-------------------------|----------------------|----------------|
| Education               | (0.5, 1.5)           | 78.8           |
| Age                     | (2.6, 2.7)           | 75.79          |
| Hours/week              | (0.4, 2.2)           | 78.6           |
| Capital-gains           | (0, 0.2)             | 79.86          |
| Capital-losses          | (0.2, 0.4)           | 78.15          |
Figure 3: Neural Networks as Extended Thresholds

Table 3: Thresholds learnt on Covertype dataset

| Feature predicted by NN | Test Accuracy % |
|-------------------------|-----------------|
| Elevation | (1, −1, 3, 0) | 66.65 |
| Aspect | (2, −1, 2, 2) | 64.79 |
| Scope | (2, −1, 5, 3.5) | 64.6 |