THEORY OF COMPUTING, Volume 22 (5), 2026, pp. 1–40
www.theoryofcomputing.org

PAC Verification of Statistical Algorithms

Saachi Mutreja   Jonathan Shafer

Received September 2, 2023; Revised March 23, 2026; Published July 10, 2026

Abstract. Goldwasser et al. (ITCS'21) proposed the setting of PAC verification, where a hypothesis (machine learning model) that purportedly satisfies the agnostic PAC learning objective is verified using an interactive proof. In this paper we develop this notion further in a number of ways. First, we prove a lower bound of $\Omega\left(\sqrt{d}/\varepsilon^2\right)$ i.i.d. samples for PAC verification of hypothesis classes with VC dimension d. Second, we present a protocol for PAC verification of unions of intervals over $\mathbb{R}$ that improves upon their proposed protocol for that task, and matches our lower bound's dependence on d. Third, we introduce a natural generalization of their definition to verification of general statistical algorithms, which is applicable to a wider variety of settings beyond agnostic PAC learning. Showcasing our proposed definition, our final result is a protocol for the verification of statistical query algorithms that satisfy a combinatorial constraint on their queries. In particular, this protocol can be used for verification of adaptive data analysis.

1 Introduction

Comparing what can be computed in a given model of computation versus what can be verified in that model is a recurring theme throughout the fields of computability and computational

A conference version of this paper appeared in the Proceedings of the Thirty-Sixth Conference on Learning Theory (COLT'23) [25].

ACM Classification: F.1.3, F.2.3

AMS Classification: 68Q17, 68Q15, 03F20, 68Q25, 68Q32

Key words and phrases: interactive proofs, PAC learning, PAC verification, VC dimension, statistical query algorithms

 DOI: 10.4086/toc.2026.v022a005

complexity. The most notorious example is of course the P vs. NP problem, which asks whether the set of decision problems that can be solved in polynomial time equals the set of decision problems whose solution can be verified in polynomial time given a suitable proof string. But the same question has been studied for many other settings and models of computation as well, with prominent examples including L vs. NL (for logspace computation), P vs. IP = PSPACE (polytime computation, with an interactive proof) and MIP* = RE (ditto, with multiple quantum provers). The existence of a gap between computing and verifying is sometimes interpreted as capturing the notion of *creativity*, in the sense that finding a solution to a problem might require discovery or inventiveness, while verifying a formal proof for the same is merely rote work.

While this theme has deep roots in the literature and an appealing interpretation, its parallels for *learning* have only recently been explored for the first time. In the context of PAC[1] learning, Goldwasser et al. [15] introduced the setting of *PAC verification*, in which an untrusted prover attempts to convince a verifier that a certain classifier has nearly optimal loss with respect to a fixed unknown distribution from which the verifier can take random samples. Specifically, they work in the agnostic PAC learning setting, where the objective is to find a hypothesis h that has nearly optimal loss compared to a hypothesis class $\mathcal{H}$ in the sense

$$L_{\mathcal{D}}^{0\text{-}1}(h) \leq \inf_{h' \in \mathcal{H}} L_{\mathcal{D}}^{0\text{-}1}(h') + \varepsilon, \tag{1.1}$$

where $L_{\mathcal{D}}^{0\text{-}1}$ denotes 0-1 population loss[2] and $\mathcal{H}$ is some fixed and known hypothesis class (formal definitions appear in Sections 1.3 and 2.2 below).

Seeing as complexity gaps between computation and verification for general computational tasks are already well-studied[3], the main novelty in this setting concerns sample complexity gaps. They show that for some hypothesis classes (but not for others) the number of i.i.d. samples necessary to find a hypothesis with nearly optimal loss is strictly greater than the number of i.i.d. samples necessary for verifying, with the help of an untrusted prover, that a proposed hypothesis has nearly optimal loss.

Beyond the (substantial) theoretical motivation, this setting could have meaningful (and timely) real-world applications. First, if a sample complexity gap exists then a verifiable "data collection with ML training" service becomes a viable business model. The provider would collect suitable training data from the desired population distribution, execute a chosen ML algorithm, and subsequently prove to the client that the end result is good with respect to the population distribution. The client would only need a small amount of independent data from the population distribution to determine the veracity of the claim. Beyond this, Goldwasser et al. [15] envision a variety of other applications, such as more efficient schemes for replicating scientific results in the empirical sciences.

[1]Probably Approximately Correct (PAC) is the standard theoretical model for supervised learning, introduced by Vapnik and Chervonenkis [36] and Valiant [35]. Agnostic PAC learning is a generalization to the non-realizable case, introduced by Haussler [17]. See also [32].

[2]The 0-1 loss is $L_{\mathcal{D}}^{0\text{-}1}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$. See Definition 1.5.

[3]Namely, the relations between P, NP, MA, AM and IP have been studied extensively. In particular, doubly efficient delegation of general computational tasks has also been studied, see, e. g., [14].

1.1 Our contributions

PAC verification is novel territory, and very little is currently known. The current paper aims to make some modest steps towards charting this landscape. We focus on studying sample complexity gaps between learning and verifying specifically in terms of the dependence on the VC (Vapnik–Chervonenkis) dimension. We start with showing a lower bound for the sample complexity gap. Prior to our work, one could imagine that some classes would give rise to very large gaps, e. g., $O\left(\log(d)\right)$ i.i.d. samples for verifying vs. the $\Theta(d)$ samples that are known to be necessary and sufficient for learning, where $d = \mathsf{VC}(\mathcal{H})$. Our first result shows that the gap can be at most quadratic. Namely, for every hypothesis class, PAC verification requires that the verifier use at least $\Omega\left(\sqrt{d}\right)$ i.i.d. random samples.

Second, we show that our lower bound's dependence on the VC dimension is tight in some cases, by improving upon a result of Goldwasser et al. [15] to obtain a PAC verifier for the class of unions of intervals on $\mathbb{R}$ that uses $O\left(\sqrt{d}\right)$ i.i.d. random samples. The previous result was an upper bound for a weaker notion of verification, that guarantees only that $L_{\mathcal{D}}^{0\text{-}1}(h) \leq 2 \cdot \mathsf{Opt} + \varepsilon$, where $\mathsf{Opt} = \inf_{h' \in \mathcal{H}} L_{\mathcal{D}}^{0\text{-}1}(h')$ (instead of $\mathsf{Opt} + \varepsilon$ as in Eq. (1.1)). Their result applied only to a specific restriction of the class of unions of intervals, while our technique works for the restricted and for the unrestricted versions of the class.[4]

Third, we take a step towards making the notion of PAC verification more applicable in practical settings. Many ML and data science algorithms that people use in practice, and might like to delegate to an untrusted service, do not obtain (or at least do not provably obtain) the objective of agnostic PAC learning as in Eq. (1.1). Instead, they obtain some quantity of loss which is typically good enough in practice. With this reality in mind, we introduce a generalization of PAC verification that guarantees that the outcome is competitive with a specific algorithm. Namely, the verifier guarantees that with high probability, the hypothesis h satisfies $L_{\mathcal{D}}^{0\text{-}1}(h) \leq \mathbb{E}\left[L_{\mathcal{D}}^{0\text{-}1}(h_A)\right] + \varepsilon$, where h_A is the (possibly randomized) output of the algorithm (see Definition 2.4).

Fourth, we study PAC verification of statistical query algorithms. For a batch $\mathbf{q}$ of statistical queries, we define a notion of *partition size*, denoted $\mathsf{PS}(\mathbf{q})$, which is the number of atoms in the σ-algebra generated by $\mathbf{q}$. We show that whenever this quantity is sufficiently small, there is a sample complexity gap between execution and verification of the statistical query algorithm.

Lastly, we show natural examples that exhibit sample complexity gaps. Both our lower bound and our upper bound apply to a simple example of optimizing a portfolio with advice. We also present examples of verification in graphs, and verification of adaptive data analysis.

[4] Specifically, their result applied to functions where the domain is partitioned into d fixed and known subdomains of equal length. Within each subdomain, the function is a threshold function of the form $f(x) = \mathbb{1}(x \geq t)$.

1.2 Related work

Chiesa and Gur [10] initiated the study of interactive proofs for properties of distributions. That setting is a closely related precursor to PAC verification, since PAC verification can be viewed as the special case of verifying that an unknown population distribution $\mathcal{D}$ has the property that a proposed hypothesis is nearly optimal with respect to $\mathcal{D}$. They showed general bounds in terms of the support size. However, they did not consider tighter bounds that depend on combinatorial characterizations of the distribution testing property of interest (e. g., bounds that depend on the VC dimension).[5,6]

The study of PAC verification of a hypothesis class was introduced by Goldwasser et al. [15], who considered interactive proofs for properties of distributions in the specific context of machine learning. In particular, they also considered the relationship between the VC dimension of the class and the sample complexity of verification. They showed a lower bound that is incomparable with our lower bound[7], and they showed an upper bound for unions of intervals which is weaker than our upper bound. Our definition of PAC verification of an algorithm is closely modeled on their definition.

Recently, a line of work has emerged on the general theme of distribution testing and interactive proofs for properties of distributions in the context of machine learning. These include [5], [2], [30] and [18], among others. Caro et al. [9] studied PAC verification with a quantum prover. Seshia et al. [31] survey the use of formal methods for verification of AI systems.

Shortly before this article went to print, the authors were informed of concurrent ongoing work by Ngo and Kim [26], who investigate proof systems for delegating adaptive statistical query algorithms in detail. They independently discovered the results of Section 3.3, and further, show how to provide non-adaptive, publicly verifiable certificates of statistical validity.

1.3 Preliminaries

Notation 1.1. $\mathbb{N} = \{1, 2, 3, \dots\}$, i. e., $0 \notin \mathbb{N}$. For every $n \in \mathbb{N}$, we denote $[n] = \{1, 2, 3, \dots, n\}$.

Notation 1.2. For a set Ω, we write $\Delta(\Omega)$ to denote the set of all probability measures defined on the measurable space $(\Omega, \mathcal{F})$, where $\mathcal{F}$ is some fixed σ-algebra that is implicitly understood.

[5]For example, Theorem 1.1 in Chiesa and Gur [10] describes a proof system where the verifier has sample complexity $O(\sqrt{n})$, where n is the size of the domain. In contrast, our results depend on combinatorial quantities like the VC dimension and the partition size, which may be much smaller (independent of n).

[6]Additionally, Chiesa and Gur [10] consider interactive proofs where the prover is unbounded, while we follow Goldwasser et al. [15] and insist on doubly efficient proof systems, where both the prover and verifier must be efficient in terms of runtime and sample complexity.

[7]Their lower bound is stronger than ours, since it is a lower bound of $\widetilde{\Omega}(d)$ for a class with VC dimension d, while our lower bound is $\Omega\left(\sqrt{d}\right)$. However, our lower bound is stronger than theirs in the sense that it applies to all classes, while their lower bound applies only to specific classes. Additionally, their lower bound only applies to proper PAC verification, while our lower bound applies to both proper and improper PAC verification.

Definition 1.3. Let $\mathcal{P}, \mathcal{Q}$ be probability measures defined on a measurable space $(\Omega, \mathcal{F})$. The total variation distance between $\mathcal{P}$ and $\mathcal{Q}$ is $\mathsf{TV}(\mathcal{P}, \mathcal{Q}) = \sup_{A \in \mathcal{F}} |\mathcal{P}(A) - \mathcal{Q}(A)|$.

PAC Learning

Definition 1.4. Let $\mathcal{X}$ be a set, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a set of functions. Let $k \in \mathbb{N}$, $X = \{x_1, x_2, \ldots, x_k\} \subseteq \mathcal{X}$. We say that $\mathcal{H}$ shatters X if for every $y_1, y_2, \ldots, y_k \in \{0,1\}$ there exists $h \in \mathcal{H}$ such that $h(x_i) = y_i$ for all $i \in [k]$. The Vapnik–Chervonenkis (VC) dimension of $\mathcal{H}$, denoted $\mathsf{VC}(\mathcal{H})$, is the largest $d \in \mathbb{N}$ for which there exists a set $X \subseteq \mathcal{X}$ of cardinality d that is shattered by $\mathcal{H}$. If $\mathcal{H}$ shatters sets of arbitrarily large cardinality, we say that $\mathsf{VC}(\mathcal{H}) = \infty$.

Throughout most of this paper we use loss functions of the type common in PAC learning, where the loss of a hypothesis with respect to a distribution is defined as the expected loss of that hypothesis on a randomly drawn sample from the distribution, as follows.

Definition 1.5. Let Ω and $\mathcal{H}$ be sets. A loss function is a function $L: \Omega \times \mathcal{H} \to [0,1]$. Let $h \in \mathcal{H}$, and let $S = (z_1, \ldots, z_m) \in \Omega^m$ be a vector. The empirical loss of h with respect to S is $L_S(h) = \frac{1}{m} \sum_{i \in [m]} L(z_i, h)$. For every distribution $\mathcal{D} \in \Delta(\Omega)$, the loss of h with respect to $\mathcal{D}$ is $L_{\mathcal{D}}(h) = \mathbb{E}_{Z \sim \mathcal{D}}[L(Z, h)]$. The loss of $\mathcal{H}$ with respect to $\mathcal{D}$ is $L_{\mathcal{D}}(\mathcal{H}) = \inf_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$.

The 0-1 loss, denoted $L^{0\text{-}1}$, is the special case in which $\mathcal{X}$ is a set, $\Omega = \mathcal{X} \times \{0,1\}$, $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$, and $L((x,y), h) = \mathbb{1}(h(x) \neq y)$.

However, in Definition 2.4 below we also consider more general types of loss.

Definition 1.6. Let $\mathcal{X}$ be a set, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a class of hypotheses. We say that $\mathcal{H}$ is agnostically PAC learnable if there exist an algorithm A and a function $m_A: [0,1]^2 \to \mathbb{N}$ such that for every $\varepsilon, \delta \in (0,1)$ and every distribution $\mathcal{D} \in \Delta(\mathcal{X} \times \{0,1\})$, if A receives as input a tuple of $m_A(\varepsilon, \delta)$ i.i.d. samples from $\mathcal{D}$, then A outputs a function $h \in \mathcal{H}$ satisfying

$$\mathbb{P}\left[L_{\mathcal{D}}^{0\text{-}1}(h) \leq L_{\mathcal{D}}^{0\text{-}1}(\mathcal{H}) + \varepsilon\right] \geq 1 - \delta.$$

In words, this means that h is probably (with confidence $1 - \delta$) approximately correct (has loss at most ε worse than the optimal loss in $\mathcal{H}$). The point-wise minimal such function m_A is called the sample complexity of $\mathcal{H}$.

PAC Verification of a Hypothesis Class

Definition 1.7 (PAC Verification of a Hypothesis Class; a special case of Goldwasser et al. [15], Definition 4)**.** Let $\mathcal{X}$ be a set, let $\mathbb{D} \subseteq \Delta(\mathcal{X} \times \{0,1\})$ be a set of distributions, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a class of hypotheses. We say that $\mathcal{H}$ is PAC verifiable with respect to $\mathbb{D}$ using random samples if there exist an interactive proof system consisting of a verifier V and an honest prover P such that for every $\varepsilon, \delta \in (0,1)$ there exist $m_V, m_P \in \mathbb{N}$ such that for every $\mathcal{D} \in \mathbb{D}$, the following conditions are satisfied:

- **Completeness.** Let the random variable

$$h_V = [V(S_V, \varepsilon, \delta), P(S_P, \varepsilon, \delta)] \in \{0,1\}^{\mathcal{X}} \cup \{\textsf{reject}\}$$

denote the output of V after interacting with P, where they receive as input parameters ε and δ, as well as sets S_V and S_P (respectively) of i.i.d. samples from $\mathcal{D}$. Then

$$\mathbb{P}_{S_V \sim \mathcal{D}^{m_V}, S_P \sim \mathcal{D}^{m_P}}\left[h_V \neq \textsf{reject} \ \wedge \ \left(L_{\mathcal{D}}^{0\text{-}1}(h_V) \leq L_{\mathcal{D}}^{0\text{-}1}(\mathcal{H}) + \varepsilon\right)\right] \geq 1 - \delta.$$

- **Soundness.** For every (possibly malicious and computationally unbounded) prover P' (which may depend on $\mathcal{D}$, ε, and δ), the verifier's output $h_V = [V(S_V, \varepsilon, \delta), P']$ satisfies

$$\mathbb{P}_{S_V \sim \mathcal{D}^{m_V}, S_P \sim \mathcal{D}^{m_P}}\left[h_V = \textsf{reject} \ \vee \ \left(L_{\mathcal{D}}^{0\text{-}1}(h_V) \leq L_{\mathcal{D}}^{0\text{-}1}(\mathcal{H}) + \varepsilon\right)\right] \geq 1 - \delta.$$

In both conditions, the probability is over the randomness of the samples S_V and S_P, as well as the randomness of V, P and P'.

If V has the additional property that for every prover, with probability 1, $h_V \in \mathcal{H} \cup \{\textsf{reject}\}$, then we say that the interactive proof system is proper and that $\mathcal{H}$ is proper PAC verifiable. Otherwise, they are improper.

Remark 1.8. [15, Definition 4] is more general than Definition 1.7. In particular, they allow for other types of oracle access to the unknown distribution beyond random samples, and they also consider verifying multiplicative approximations to the optimal loss. We focus on the special case of verifying additive approximations where both parties use random samples. ◻

2 Technical overview

2.1 Bounds for verification of VC classes

Our first result is a lower bound for the number of i.i.d. random samples the verifier requires to successfully PAC verify a class. Full proofs for all results are deferred to Sections 4 to 6.

Theorem 2.1. *There exist constants $C, c > 0$ as follows. Let $\varepsilon \in (0,1)$, $\delta = 1/3$, let $\mathcal{X}$ be a set, and let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a hypothesis class with $\mathsf{VC}(\mathcal{H}) = d \in \mathbb{N}$. Assume that (V, P) is a (proper or improper) interactive proof system that PAC verifies $\mathcal{H}$ with parameters (ε, δ) with respect to the set of all distributions $\mathbb{D} = \Delta(\mathcal{X} \times \{0,1\})$, and the verifier V uses $m_V = m_V(d, \varepsilon)$ i.i.d. labeled samples. Then $m_V(d, \varepsilon) \geq (C \cdot \sqrt{d} - c)/\varepsilon^2$.*

Remark 2.2. The lower bound in Theorem 2.1 holds:

1. Regardless of whether the prover is efficient and has sample access to the population distribution, or is computationally unbounded and has full knowledge of the population distribution; and
2. Regardless of whether the verification protocol is proper or improper. This is in contrast to the $\widetilde{\Omega}(d)$ lower bound in [15, Lemma 4.1], which applies only to proper verification.

Proof Idea. This is an application of Le Cam's method (Claim 8.6), together with a reduction from distribution testing to PAC verification. Consider distributions where the marginal over the domain is uniform on a fixed $\mathcal{H}$-shattered set of size d. PAC verification requires distinguishing the case of truly random labels (where the loss of the class is $1/2$), from the case where the labels are ε-biased (and the loss of the class is $1/2 - \varepsilon$). An $\Omega\left(\sqrt{d}/\varepsilon^2\right)$ lower bound for distinguishing these two cases is due to Paninski [27]. □

Our second result shows that the lower bound's dependence on d is tight for a specific class.

Theorem 2.3. *Let $d \in \mathbb{N}$, and let*

$$\mathcal{H}_d = \left\{ \mathbb{1}_X : \ X = \bigcup_{i \in [d]} [a_i, b_i] \ \wedge \ (\forall i \in [d] : \ 0 \leq a_i \leq b_i \leq 1) \right\} \subseteq \{0,1\}^{[0,1]}$$

be the class of boolean-valued functions over the domain $[0,1]$ that are indicator functions for a union of d intervals. There exists a proper interactive proof system that PAC verifies the class $\mathcal{H}_d$ with respect to the set of all distributions over $[0,1] \times \{0,1\}$, such that the verifier uses

$$m_V = O\left(\frac{\sqrt{d}\log(1/\delta)}{\varepsilon^{2.5}}\right)$$

random samples, the honest prover uses

$$m_P = O\left(\frac{d^2}{\varepsilon^4} + \frac{d\log(1/\delta)}{\varepsilon^3}\right)$$

random samples, and both the verifier and the honest prover run in time polynomial in their numbers of samples.

Proof Idea. A discretization of the population distribution is induced by partitioning the domain $[0,1]$ into d/ε intervals, each of which has weight ε/d according to the population distribution. In the discretized distribution, the probability mass from each interval is lumped together into a single arbitrary point in that interval. We show that to find an ε-suboptimal union of intervals, it suffices to know this discretized distribution. The prover sends the (purported) discretized distribution to the verifier. The verifier uses a distribution identity tester to verify that the provided distribution is a correct discretization of the population distribution. This is possible using $O\left(\sqrt{d}\right)$ samples, because the support of the discretized distribution is of size $O(d)$. □

2.2 Verification of statistical algorithms

Many popular algorithms do not come with provable PAC-like guarantees, but tend to work well in practice. Such heuristics are common in machine learning, data science, optimization, operations research, finance, etc. People might like to delegate the task of collecting data and executing an algorithm on that data to an untrusted party. To capture this notion, our next contribution is a new definition of PAC verification of an *algorithm*.[8] This generalizes the definition of PAC verification of a *hypothesis class* (Definition 1.7, introduced by [15]), which corresponds to the special case of PAC verifying an algorithm that is an agnostic PAC learner for the class.

Definition 2.4 (PAC Verification of an Algorithm)**.** Let Ω be a set, let $\mathbb{D} \subseteq \Delta(\Omega)$ be a set of distributions, let $\mathcal{H}$ be a set (called the set of *possible outputs*), and for each $\mathcal{D} \in \mathbb{D}$ let $\mathcal{O}_{\mathcal{D}}$ be an oracle. Let A be a (possibly randomized) algorithm that takes no inputs, has query access to $\mathcal{O}_{\mathcal{D}}$, and outputs a value $h_A = A^{\mathcal{O}_{\mathcal{D}}} \in \mathcal{H}$. Let $L : \mathbb{D} \times (\mathcal{H} \cup \{\mathsf{reject}\}) \to [0,1]$ be an arbitrary function[9], let $L_{\mathcal{D}}(\cdot)$ denote $L(\mathcal{D}, \cdot)$, and define the loss of an algorithm as $L_{\mathcal{D}}\left(A^{\mathcal{O}_{\mathcal{D}}}\right) = \mathbb{E}[L_{\mathcal{D}}(h_A)]$, where the expectation is over the randomness of A and of the oracle $\mathcal{O}_{\mathcal{D}}$. We say that the algorithm A with access to oracles $\{\mathcal{O}_{\mathcal{D}}\}_{\mathcal{D} \in \mathbb{D}}$ is PAC verifiable with respect to $\mathbb{D}$ by a verification protocol that uses random samples if there exists an interactive proof system consisting of a verifier V and an honest prover P such that for every $\varepsilon, \delta \in (0,1)$ there exist $m_V, m_P \in \mathbb{N}$ such that for every $\mathcal{D} \in \mathbb{D}$, the following conditions are satisfied:

- **Completeness.** Let the random variable

$$h_V = [V(S_V, \varepsilon, \delta), P(S_P, \varepsilon, \delta)] \in \mathcal{H} \cup \{\mathsf{reject}\}$$

  denote the output of V after interacting with P, where they receive as input parameters ε and δ, as well as sets S_V and S_P (respectively) of i.i.d. samples from $\mathcal{D}$. Then

$$\mathbb{P}_{S_V \sim \mathcal{D}^{m_V}, S_P \sim \mathcal{D}^{m_P}}\left[h_V \neq \mathsf{reject} \ \wedge \ L_{\mathcal{D}}(h_V) \leq L_{\mathcal{D}}\left(A^{\mathcal{O}_{\mathcal{D}}}\right) + \varepsilon\right] \geq 1 - \delta.$$

- **Soundness.** For every deterministic or randomized (possibly malicious and computationally unbounded) prover P' (which may depend on $\mathcal{D}$, ε, δ and $\{\mathcal{O}_{\mathcal{D}}\}_{\mathcal{D} \in \mathbb{D}}$), the verifier's output $h = [V(S_V, \varepsilon, \delta), P']$ satisfies

$$\mathbb{P}_{S_V \sim \mathcal{D}^{m_V}}\left[h_V = \mathsf{reject} \ \vee \ L_{\mathcal{D}}(h_V) \leq L_{\mathcal{D}}\left(A^{\mathcal{O}_{\mathcal{D}}}\right) + \varepsilon\right] \geq 1 - \delta.$$

In both conditions, the probability is over the randomness of the samples S_V and S_P, as well as the randomness of V, P and P'.

[8]This notion differs from delegation of computation, in that the data (the input to the algorithm) is collected by the untrusted prover.

[9]Note that this is more general than in Definition 1.5.

In other words, whereas the definition of Goldwasser et al. [15] required that the interactive proof system guarantee that a hypothesis is competitive with respect to every hypothesis in $\mathcal{H}$, our definition requires that it be competitive with respect to a specific algorithm.

Remark 2.5. PAC verification of an algorithm A requires that $L_{\mathcal{D}}(h_V) \le \mathsf{Opt}_A + \varepsilon$ with high probability. Two natural candidate definitions for Opt_A include (1) $\mathsf{Opt}_A = L_{\mathcal{D}}(h_A)$, and (2) $\mathsf{Opt}_A = \mathbb{E}[L_{\mathcal{D}}(h_A)]$. Candidate (1) requires that with high probability the verifier's output be at most ε worse than the output of executing algorithm A, while (2) requires that it be at most ε worse than the expected loss of A.

The loss $L_{\mathcal{D}}(h_A)$ is a random variable that depends, inter alia, on the random samples used by A (more generally: on the randomness of the oracle used by A). A crucial aspect of PAC verification is that the verifier use less random samples than are necessary for executing A, and in particular it cannot access the random samples used by A. So the verifier cannot know what loss was obtained in any particular execution of A. Therefore, we reject candidate (1) and adopt candidate (2). □

As an application of this new definition, we show that some statistical query algorithms (see Definitions 6.1 and 6.3) can be PAC verified via a protocol in which the verifier uses less i.i.d. samples than would be required for simulating the statistical query oracle used by the algorithm. Specifically, for a batch $\mathbf{q}$ of statistical queries, the *partition size* $\mathsf{PS}(\mathbf{q})$ is the number of atoms in the σ-algebra generated by $\mathbf{q}$. If the algorithm uses only batches with small partition size then verification is cheap, as in the following theorem.[10]

Theorem 6.12 (Informal version)**.** Let A be a statistical query algorithm that adaptively generates at most b batches of queries with precision τ such that each batch $\mathbf{q}$ satisfies $\mathsf{PS}(\mathbf{q}) \le s$. Then A is PAC verifiable by an interactive proof system where the verifier uses

$$m_V = \Theta\left(\frac{\sqrt{s}\log(b/\varepsilon\delta)}{\tau^2} + \frac{\log(1/\varepsilon\delta)}{\varepsilon^2}\right)$$

i.i.d. samples. Both the verifier and the honest prover run in polynomial time.

Proof Idea. The verifier simulates algorithm A. Each time A sends a batch of queries to be evaluated by the statistical query oracle, the verifier sends the queries to the prover, and the prover sends back a vector of purported evaluations. The verifier uses $O(\sqrt{s}/\tau^2)$ i.i.d. random samples to execute a distribution identity tester (Theorem 5.1) to verify that the prover's evaluations are correct up to the desired accuracy τ. □

In particular, Theorem 6.12 implies the following separation:

[10]In other words, if the set of queries $\mathbf{q}$ used by the algorithm is generated by a small set of queries $\mathbf{a}$ ("atoms"), in the sense that every query in $\mathbf{q}$ indicates a disjoint union of queries in $\mathbf{a}$, then estimating $\mathbf{a}$ suffices for estimating $\mathbf{q}$.

Corollary 6.15 (Informal version)**.** Let $d \in \mathbb{N}$ and let A be a statistical query algorithm such that each batch of queries generated by A corresponds precisely to a σ-algebra with d atoms. Then simulating A using random samples requires $\Omega(d/\tau^2)$ random samples, but there exists a PAC verification protocol for A where the verifier uses $O\left(\sqrt{d}/\tau^2\right)$ random samples.

The $O\left(\sqrt{d}/\tau^2\right)$ upper bound in Corollary 6.15 is tight due to the lower bound of Theorem 2.1.

3 Examples

This section shows how our setting and results can be applied. In particular, it illustrates the utility of our new definition of PAC verification of an algorithm (Definition 2.4) for modeling verification in cases that do not fit naturally within the definition of Goldwasser et al. [15] of PAC verification of a hypothesis class (Definition 1.7).

There are perhaps two main differences between our setting in Definition 2.4 and that of Definition 1.7.

1. **Task.** PAC verification of a hypothesis class was intended to capture verification of PAC learning. In particular, it focuses on a setting of classification or regression, where there exists a population distribution $\mathcal{D}$ of (x, y) pairs, and the objective is to find a function f that minimizes the probability that $f(x)$ is far from y for a random pair $(x, y) \sim \mathcal{D}$ (formally, to minimize $\mathbb{P}[y \neq f(x)]$ or $\mathbb{E}\left[(y - f(x))^2\right]$, etc.). In contrast, PAC verification of an algorithm is interested in verification of any statistical algorithm, not just classification or regression. In particular:
   (a) The population distribution might not be structured as (x, y) pairs; and
   (b) The objective is not necessarily to minimize the expected value $\mathbb{E}_{z\sim\mathcal{D}}[L(f, z)]$ of a local loss function L that is defined for individual z.
2. **Benchmark.** Perhaps most importantly, in PAC verification of a hypothesis class, the objective is to ensure that a hypothesis is competitive with the best hypothesis in a hypothesis class. In contrast, in PAC verification of an algorithm, the objective is to ensure that the selected hypothesis is competitive with the output of a chosen algorithm, which may not itself possess any formal guarantees.

3.1 Basic tasks beyond PAC learning

Our first two examples demonstrate Item 1 above, by showing that PAC verification of an algorithm can be used to model tasks that are not instances of PAC learning.

Example 3.1 (Optimizing a portfolio with advice)**.** Consider a task in which an agent selects a subset S consisting of n items from the set $\Omega = [2n]$. Subsequently, an item $i \in \Omega$ is chosen at random according to a distribution $\mathcal{D} \in \Delta(\Omega)$ that is unknown to the agent, and the agent experiences loss $L(i, S) = \mathbb{1}(i \notin S)$.

To help make its decision, the agent has access to an i.i.d. sample $Z = (z_1, \dots, z_m) \sim \mathcal{D}^m$. Let $\mathcal{H} = \binom{\Omega}{n}$ denote the collection of subsets of size n that the agent could select. $\mathsf{VC}(\mathcal{H}) = n$, and therefore estimating the expected loss $L_{\mathcal{D}}(S)$ of each possible choice $S \in \mathcal{H}$ up to precision $\varepsilon > 0$ requires $m_A = \Omega\big((n + \log(1/\delta))/\varepsilon^2\big)$ samples.

By Corollary 6.15, if the agent can receive advice from an untrusted prover, it can make an ε-optimal choice using $m_V = O\big(\sqrt{n}\log(1/\delta\varepsilon)/\varepsilon^2\big)$ i.i.d. samples.[11] Note that $m_V \ll m_A$ for large n. Furthermore, our expression for m_V is tight in the sense that, by Theorem 2.1, $\Omega\big(\sqrt{n}\big)$ samples are necessary for verifying the advice of an untrusted prover. □

Strictly speaking, the foregoing example is not an instance of PAC verification of a hypothesis class, because the population distribution is not structured as (x, y) pairs (Item 1a above), and the objective is not to predict the correct label y for a given input x.

However, there is a very simple reduction from the previous example to proper PAC verification of a hypothesis class, because the objective is still expressed in terms of the expectation of a local loss function $L(i, S)$ as in PAC learning.[12] In contrast, our next example is further removed from PAC learning. Here too, the distribution is not structured as (x, y) pairs. But furthermore, the objective is not to minimize the expected value of a local loss function that is known to the verifier (Item 1b).[13]

Example 3.2 (Verification of distribution learning[14])**.** Let $\Omega = [n]$. Consider a task in which an agent has access to an i.i.d. sample $Z = (z_1, \dots, z_m) \sim \mathcal{D}^m$ from some distribution $\mathcal{D} \in \Delta(\Omega)$ that is unknown to the agent. The agent selects a distribution $\hat{\mathcal{D}} \in \Delta(\Omega)$, and experiences loss $L_{\mathcal{D}}\Big(\hat{\mathcal{D}}\Big) = \mathsf{TV}\Big(\hat{\mathcal{D}}, \mathcal{D}\Big)$.

It is well known that to achieve loss at most ε with probability at least $1 - \delta$, it is necessary and sufficient to take $m_A = \Theta\big((n + \log(1/\delta))/\varepsilon^2\big)$ samples ([6], Theorem 1). In contrast, if the agent has access to advice from an untrusted prover then $m_V = O\big(\sqrt{n}\log(1/\delta)\varepsilon^{-2}\big)$ i.i.d. samples are sufficient. The honest prover simply sends the verifier a description of a distribution $\tilde{\mathcal{D}} \in \Delta(\Omega)$

[11] We invoke Corollary 6.15 with a single set of queries, $\mathbf{q}$, which is the set of all functions $[2n] \to \{0, 1\}$. This set has $\mathsf{VC}(\mathbf{q}) = 2n$ and $|\mathbf{q}| = 2^{2n}$.

[12] Namely, let $\mathcal{D}_1$ be the distribution over pairs $(x, 1)$ where $x \sim \mathcal{D}$, and let $\mathcal{H}$ be the collection of indicator functions for subsets of size n of $[2n]$. Then proper PAC verification of $\mathcal{H}$ with respect to $\mathcal{D}_1$ is equivalent to the task in the example.

[13] To be precise, in Example 3.2 the objective is to minimize the total variation distance between the true distribution $\mathcal{D}$ and a distribution $\hat{\mathcal{D}}$ selected by the agent. Because the total variation distance is an f-divergence [28], it can be expressed as $\mathsf{TV}\Big(\hat{\mathcal{D}}, \mathcal{D}\Big) = \mathbb{E}_{z \sim \mathcal{D}}[f(z)]$ for $f(z) = \frac{1}{2}|\hat{\mathcal{D}}(z)/\mathcal{D}(z) - 1|$. However, the function f depends on the distribution $\mathcal{D}$, which is unknown to the agent. This is different from the PAC verification setting, where the loss function (e. g., the 0-1 loss or square loss) is known to the verifier.

[14] Example 3.2 is similar to [10, Theorem 1] (see Corollary 3.5 in the full version of that paper).

that has loss at most $\varepsilon/\sqrt{n}$. The verifier uses distribution testing (Theorem 5.1) to decide whether $L_{\mathcal{D}}\left(\tilde{\mathcal{D}}\right) \leq \varepsilon/\sqrt{n}$ or $L_{\mathcal{D}}\left(\tilde{\mathcal{D}}\right) \geq \varepsilon$, and accepts if and only if the former case holds. □

Distribution learning is a very basic and useful task, so we will invoke Example 3.2 as a subroutine in some of the following examples.

3.2 Verification in graphs

A large collection of concrete tasks that might be of interest and that fall within the setting of Definition 2.4 involve solving various problems on graphs given random samples that convey information about the graph.

We will in particular use an example on graphs to demonstrate the difference in *benchmark* between PAC verification of a hypothesis class and PAC verification of an algorithm (Item 2 above).

Example 3.3 (Approximate Vertex Cover)**.** Fix $n \in \mathbb{N}$. For every undirected graph $G = (V, E)$ with $V = [n]$, let $\mathcal{D}_G$ be the uniform distribution on E.

Consider the task of finding a vertex cover in the graph G. Recall that a vertex cover is a subset of vertices $S \subseteq V$ such that for each $e \in E$, $e \cap S \neq \varnothing$. The agent does not know G, but it knows n, and it has access to an i.i.d. sample $Z = (z_1, \ldots, z_m) \sim \mathcal{D}_G^m$.

Finding a minimum vertex cover is NP-hard, but it is easy to find an approximation. Specifically, we are interested in PAC verification of GreedyVertexCover (Algorithm 1), which is a simple algorithm that constructs a 2-approximation of the minimum vertex cover. The approximation algorithm simply constructs a maximal matching in the graph in a greedy manner, and then outputs all the vertices that are incident to an edge in the matching. The resulting cover has cardinality at most $2 \cdot \mathsf{Opt}$, i. e., it is at most twice as large as the minimum vertex cover (Claim 8.1).[15]

We now construct a PAC verification protocol that is competitive with this algorithm. We use the following loss function for a vertex cover S:

$$L_{\mathcal{D}_G}(S) = \max\left\{\frac{|S|}{n},\, \mathbb{1}\left(\frac{|\{e \in E:\ e \cap S = \varnothing\}|}{|E|} \leq \frac{1}{100}\right)\right\}.$$

Namely, the loss is proportional to the size of the set S as long as it is close to a valid vertex cover (i. e., the fraction of edges not covered is at most 1%), and otherwise the loss is 1.

Consider the number of i.i.d. samples from $\mathcal{D}_G$ that an agent needs in order to construct a set S that is 1/100-close to a vertex cover and has cardinality at most $2 \cdot \mathsf{Opt}$. There is an easy lower bound of $\Omega(n)$ samples for this task. To see this, consider a graph that is a disjoint union of t

[15]Obtaining an approximation factor better than 2 is NP-hard, unless the Unique Games Conjecture is false [22].

"tetrads" (sets of 4 vertices), such that each tetrad contains precisely one edge chosen uniformly at random (so the agent doesn't know in advance which of the 6 possible edges will be present in each tetrad). Here, $\mathsf{Opt} = t = n/4$. Let m be the number of samples the agent saw, let S be the set of vertices selected by the agent, and let q be the number of tetrads in which S contains at least 3 vertices. Seeing as $3 \cdot q \leq |S| \leq 2 \cdot \mathsf{Opt} = 2t$, we have $q \leq \frac{2}{3}t$. For every tetrad $T \subseteq V$ such that none of the m samples belongs to T and $|S \cap T| < 3$, with probability at least $1/6$ the edge in T is not covered by S. So the total expected number of uncovered edges is at least

$$\frac{t - q - m}{6} \geq \frac{t/3 - m}{6}.$$

If with probability 99%, S is $1/100$-close to a vertex cover, then the expected number of edges not covered by S is at most $2t/100$. So $2t/100 \geq (t/3 - m)/6$, which implies $m \geq \Omega(t) = \Omega(n)$. Hence, constructing a 2-approximation of the minimum vertex cover without the help of a prover requires $\Omega(n)$ samples.

However, if we assume that G has maximum degree bounded by a constant (as in the lower bound), then $\mathcal{D}_G$ is a uniform distribution with support size $O(n)$. Hence, given advice from an untrusted prover, the agent can solve this task with quadratically fewer samples by using the verification procedure of Example 3.2.

Concretely, the prover sends a (purported) description $\widetilde{G} = (V, \widetilde{E})$ of the graph, and then uses the verification procedure of Example 3.2 to prove that $\mathsf{TV}(\mathcal{D}_G, \mathcal{D}_{\widetilde{G}}) \leq O(\varepsilon)$, with the verifier using $m_V = O\left(\sqrt{n}\log(1/\delta)\varepsilon^{-2}\right)$ i.i.d. samples. If the verifier did not reject, then

$$\mathbb{P}\left[|E\Delta\widetilde{E}|/n \leq O(\varepsilon)\right] \geq 1 - \delta. \tag{3.1}$$

In that case, the verifier constructs a set $\widetilde{S}$ by running GreedyVertexCover (Algorithm 1) on $\widetilde{G}$.[16] Let S be the benchmark, namely, S is the output obtained by running GreedyVertexCover on G. The output $\widetilde{S}$ of the verifier satisfies the following two properties:

- *$\widetilde{S}$ is nearly a vertex cover.* $\widetilde{S}$ is a vertex cover for $\widetilde{E}$, so by Eq. (3.1), $\widetilde{S}$ covers all but at most an $O(\varepsilon)$-fraction of E.
- *$\widetilde{S}$ is competitive.* By Claim 8.3 and Eq. (3.1), $|\widetilde{S}| \leq |S| + O(\varepsilon n)$.

Thus, with probability at least $1 - \delta$, the output $\widetilde{S}$ of the verifier satisfies

$$L_{\mathcal{D}_G}\left(\widetilde{S}\right) \leq L_{\mathcal{D}_G}(S) + O(\varepsilon).$$

To see that $\Omega\left(\sqrt{n}\right)$ samples are necessary for verification with the help of a prover, consider a family of graphs consisting of a disjoint union of tetrads as above, but where only a quarter

[16] For simplicity, we present the verification protocol as if the verifier executes the 2-approximation algorithm itself. As presented, the protocol delegates the sample complexity (the burden of collecting data) to the prover, but does not delegate the computational complexity. This is purely for simplicity of presentation. In general, the bulk of the computational cost of executing the verifier can always be delegated to the prover using standard techniques for delegation of computation (e. g., [14]).

of the tetrads contain an edge. Distinguishing between this family and the previous family requires observing a collision (receiving a sample that contains the same edge twice), which requires $\Omega(\sqrt{n})$ samples by the 'birthday paradox'. ◻

The main takeaway of Example 3.3 is that the verification protocol is competitive with the *specific performance* of the algorithm. Indeed, in the worst case GreedyVertexCover provides a 2-approximation, and the verification protocol is guaranteed to be competitive with that. But the key point is that in some cases, GreedyVertexCover yields a cover that is much better than a 2-approximation—and in those cases the verification protocol is guaranteed to be competitive with that![17] This is a clear example of the difference in benchmark between Definition 1.7 (where verification is competitive with the best hypothesis in a class) and Definition 2.4 (where verification is competitive with the specific performance of the benchmark algorithm).

So far, all our examples involved a quadratic gap between learning and verifying. However, larger gaps are possible if we make strong assumptions on the unknown distribution. One example of this, pointed out by Goldwasser et al. [15], is that the gap between learning and verifying for *realizable* PAC learning is unbounded. Unbounded gaps can exist also for other tasks as well, as in the following example.

Example 3.4 (Unbounded gap in a graph task)**.** Let n, $G = (V, E)$, and $\mathcal{D}_G$ be as in Example 3.3, and consider the minimum vertex cover task from that example, with the additional assumption that the minimum vertex cover is of a known size $k \in \mathbb{N}$. Recall that there is an easy lower bound of $\Omega(n)$ on the number of random samples for finding a vertex cover of size k (or even $2k$) without the help of an untrusted prover.

In contrast, given advice from an untrusted prover, $m_V = O\big(\log(1/\delta)/\varepsilon\big)$ samples are sufficient to obtain a set $\widetilde{S} \subseteq V$ of size k that covers all but at most an ε-fraction of the edges in E. The protocol is simple. The prover proposes a set $\widetilde{S} \subseteq V$. If $|\widetilde{S}| \neq k$ then the verifier rejects. Otherwise, the verifier takes m_V samples from $\mathcal{D}_G$, and accepts if and only if all the edges in the sample are covered by $\widetilde{S}$. For completeness, if $\widetilde{S}$ is a vertex cover of size k then the verifier always accepts. For soundness, if the fraction of uncovered edges is greater than ε, then $\mathcal{D}_G$ has weight at least ε on edges that are not covered by $\widetilde{S}$, and so taking m_V samples is sufficient to ensure that the verifier rejects with probability at least $1 - \delta$. ◻

For the minimum vertex cover task, we have seen that the sample complexity gap for verifying an algorithm can be quadratic (Example 3.3), but that the gap can be unbounded in other cases that make stronger assumptions (Example 3.4). We view this as a testament to the richness of this setting. We feel that these examples only scratch the surface, seeing as one can apply this

[17] For example, if G is a path of length 3 consisting of edges $(i, j), (j, k), (k, \ell)$ such that (j, k) is the first edge in lexicographic order, then GreedyVertexCover yields a vertex cover of size 2, which is optimal. Similarly, if G consists of two copies of the star graph $K_{1,n/2-1}$ with an edge between the centers, and that edge is first in lexicographic order, then GreedyVertexCover yields a vertex cover of size 2, which is both optimal and, moreover, is of constant size compared to the size n of the graph.

type of thinking to the vast array of standard tasks and heuristic algorithms in graphs (e. g., computing matchings, colorings, cuts, flows, Hamiltonian cycles, traveling salesman paths, and many more).

3.3 PAC verification for adaptive data analysis

The following example relates to the setting of *adaptive data analysis*, studied in Hardt and Ullman [16] and Dwork et al. [13]. Intuitively, this is a setting involving two parties: an *analyst* and a *statistical query (SQ) oracle* (as in Definition 6.1). There is some unknown arbitrary distribution $\mathcal{D} \in \Delta(\mathcal{X})$ over a domain $\mathcal{X}$. The process of adaptive data analysis proceeds in $T \in \mathbb{N}$ steps, as follows. For each step $i \in [T]$:

1. The analyst selects an arbitrary statistical query, which is a function $q_i : \mathcal{X} \to \{0, 1\}$.
2. The oracle provides an evaluation $v_i \in [0, 1]$.

Importantly, in each step i, the analyst's query may depend on the previous queries and evaluations $q_1, v_1, \dots, q_{i-1}, v_{i-1}$.

In this section, we think of the SQ oracle as an *efficient algorithm*. Specifically, we are interested in designing an efficient oracle that uses i.i.d. samples from $\mathcal{D}$, and answers queries with the following guarantee.

Definition 3.5 (SQ Oracle High-Probability Accuracy)**.** Let $\mathcal{X}$ be a set, and let $\tau, \delta \in [0, 1]$ and $T \in \mathbb{N}$. An SQ oracle $\mathcal{O}$ is $\underline{(\tau, T, \delta)\text{-accurate for } \mathcal{X}}$ if for every distribution $\mathcal{D} \in \Delta(\mathcal{X})$ and for every sequence of queries $q_1, \dots, q_T : \mathcal{X} \to \{0, 1\}$,

$$\mathbb{P}_{v_1 \leftarrow \mathcal{O}(q_1), \dots, v_T \leftarrow \mathcal{O}(q_T)}\left[\exists i \in [T] : \left|v_i - \mathbb{E}_{X \sim \mathcal{D}}[q_i(X)]\right| > \tau\right] \leq \delta.$$

Dwork et al. [13] constructed an efficient SQ oracle using techniques from differential privacy.

Theorem 3.6 (Theorem 2 in Dwork et al. [13]; Corollary 16 in Dwork et al. [12])**.** *In the notation of Definition 3.5, there exists an efficient (τ, T, δ)-accurate SQ oracle that uses*

$$m = O\left(\frac{\sqrt{T} \log^{1.5}(T/\delta)}{\tau^{2.5}}\right)$$

i.i.d. samples from $\mathcal{D}$.

Steinke and Ullman [33] showed a roughly matching lower bound, stating that when the domain $\mathcal{X}$ is large, an order of $\sqrt{T}$ i.i.d. samples are necessary, even if the oracle is not efficient, as follows.

Theorem 3.7 (Theorem 35 in Steinke and Ullman [33])**.** *There exists a constant $\tau_0 \geq 0$ and a function $f(m) = O(m^2)$ as follows. In the notation of Definition 3.5, if $|\mathcal{X}| \geq 2^{f(m)}$ then there exists* *no (efficient or computationally unbounded) algorithm that uses m i.i.d. samples from $\mathcal{D}$ and is a $(\tau_0, f(m), 1/2)$-accurate SQ oracle for $\mathcal{X}$.*[18]

Hence, implementing an accurate SQ oracle can be expensive, and one might want to delegate that task to an untrusted party. In such a case, it becomes natural to ask whether an analyst might have a way to cheaply verify that the evaluations provided by the oracle are accurate.

Hoeffding's inequality (Theorem 8.5) implies that indeed, after the T steps of adaptive data analysis have been completed, the analyst can take $O\left(\log(T)/\tau^2\right)$ i.i.d. samples from $\mathcal{D}$ and verify that with probability 2/3, all the evaluations it received from the oracle were τ-accurate. Intuitively, verification is exponentially cheaper than implementing the SQ oracle, because the SQ oracle must deal with *adaptive* queries, while the verification process is executed at the end, at which point the sequence of queries has been fixed.

While Hoeffding's inequality gives an exponential gap between implementing and verifying an SQ oracle (in terms of the dependence on the number of SQ queries), Theorem 6.12 implies that in some cases, verification can be cheaper than a naïve application of Hoeffding's inequality, as in the following example.

Example 3.8. Fix $d, T \in \mathbb{N}$. Consider a setting of adaptive data analysis where at each time step $t \in [T]$:

1. The analyst selects a batch of 2^d statistical queries $\mathbf{q}_t = \left(q_t^1, \dots, q_t^{2^d}\right)$ with $\mathsf{VC}(\mathbf{q}_t) = d$; and then
2. The oracle responds with a corresponding batch of 2^d evaluations $\mathbf{v}_t = \left(v_t^1, \dots, v_t^{2^d}\right) \in [0,1]^{2^d}$.

The analyst's choice of $\mathbf{q}_t$ at each time step t may depend arbitrarily on the sequence $\mathbf{q}_1, \mathbf{v}_1, \dots, \mathbf{q}_{t-1}, \mathbf{v}_{t-1}$.

By Hoeffding's inequality, at the end of the interaction the analyst can verify that, with probability 2/3, all the evaluations provided by the oracle were correct up to precision τ, using

$$m_{\mathsf{Hoeffding}} = O\left(\frac{d + \log(T)}{\tau^2}\right)$$

i.i.d. samples. In contrast, Corollary 6.15 implies that verification is also possible using

$$m_V = O\left(\frac{\sqrt{d} \cdot \log(T)}{\tau^2}\right)$$

[18] Technically, [33, Theorem 35] is stated for statistical queries that are functions $q : \{0,1\}^d \to [-1,1]$, whereas here we use statistical queries $\mathcal{X} \to \{0,1\}$ for a general domain $\mathcal{X}$. However, to the best of our understanding, this technical difference is not mathematically consequential. Their result is stated for $\tau_0 = 0.99$, so one might expect this to translate into a constant of $\tau_0 \approx 0.49$ in our setting.

i.i.d. samples. For $d \geq \log^{\alpha}(T)$, $\alpha > 2$, we have $m_V = o(m_{\mathsf{Hoeffding}})$. In particular, for $d = \sqrt{T}$, we have

$$m_V = \widetilde{O}\left(\frac{T^{1/4}}{\tau^2}\right) \quad \text{vs.} \quad m_{\mathsf{Hoeffding}} = O\left(\frac{T^{1/2}}{\tau^2}\right).$$

So in this parameter regime, the sample complexity m_V of Corollary 6.15 represents a quadratic improvement compared to Hoeffding's inequality and to the SQ oracle of Theorem 3.6.[19] □

4 A lower bound for PAC verification of VC classes

Theorem 2.1 is proved via a reduction from the following distribution testing lower bound.

Theorem 4.1 (Reformulation of Theorem 4 in Paninski [27])**.** *Let $d, t \in \mathbb{N}$ and let $\varepsilon \in (0,1)$. For every $\sigma \in \Sigma = \{\pm 1\}^d$, let $\mathcal{D}_{\sigma,\varepsilon} \in \Delta([2d])$ be a distribution such that for all $i \in [d]$,*

$$\mathcal{D}_{\sigma,\varepsilon}(2i-1) = \frac{1+\sigma_i \cdot \varepsilon}{2d}, \quad \textit{and} \quad \mathcal{D}_{\sigma,\varepsilon}(2i) = \frac{1-\sigma_i \cdot \varepsilon}{2d}.$$

Let $\mathcal{D}_{\Sigma,\varepsilon,t}$ be the distribution over $[2d]^t$ generated by selecting a vector $\sigma \in \Sigma$ uniformly at random, and then taking t i.i.d. samples from $\mathcal{D}_{\sigma,\varepsilon}$. Let $\mathcal{D}_{U,t} = \mathrm{U}([2d])^t$ be the distribution over $[2d]^t$ generated by selecting t i.i.d. uniform samples from $[2d]$. Then $\mathsf{TV}(\mathcal{D}_{U,t}, \mathcal{D}_{\Sigma,\varepsilon,t}) \leq f_{\mathsf{Paninski}}(t,\varepsilon,d)$ for

$$f_{\mathsf{Paninski}}(t,\varepsilon,d) = \frac{1}{2} \cdot \left(\exp\left(\frac{t^2\varepsilon^4}{d}\right) - 1\right)^{1/2}.$$

The proof also uses the following well-known fact about maximal couplings (see, e. g., [29, Lemma 4.1.13]).

Theorem 4.2. *Let Ω be a set, and let $p_X, p_Y \in \Delta(\Omega)$ be distributions. Then*

$$\mathsf{TV}(p_X, p_Y) = \inf\left\{\mathbb{P}[X \neq Y] : (X,Y) \textit{ is a joint distribution with marginals } X \sim p_X \textit{ and } Y \sim p_Y\right\}.$$

4.1 Proof of Theorem 2.1

Proof of Theorem 2.1. Let $X = \{x_1, \ldots, x_d\} \subseteq \mathcal{X}$ be a set of size d that is shattered by $\mathcal{H}$ (such a set exists because $\mathsf{VC}(\mathcal{H}) = d$). Let $\mathcal{D}_U = \mathrm{U}(X \times \{0,1\})$.

For every $h \in \mathcal{H}_X = \{0,1\}^X$, let $\mathcal{D}_{h,8\varepsilon} \in \Delta(X \times \{0,1\})$ be a distribution such that

$$\forall\, (x,y) \in X \times \{0,1\} : \mathcal{D}_{h,8\varepsilon}\big((x,y)\big) = \begin{cases} (1+8\varepsilon)/2d & h(x) = y \\ (1-8\varepsilon)/2d & h(x) \neq y \end{cases}.$$

[19] The sample complexity m_V also compares favorably in this parameter regime to the SQ oracle of Theorem 3 in [13].

Consider a (possibly randomized) testing algorithm T that takes t i.i.d. samples from an unknown distribution $\mathcal{D}$ and decides correctly with probability at least $1-\beta$ whether $\mathcal{D} = \mathcal{D}_U$ or whether $\mathcal{D} \in \{\mathcal{D}_{h,8\varepsilon} : h \in \mathcal{H}_X\}$ (if $\mathcal{D}$ is not one of these $|\mathcal{H}_X| + 1$ options then we make no assumptions regarding the behavior of T).

Let $\mathcal{D}_{U,t} = (\mathcal{D}_U)^t$ and let $\mathcal{D}_{\mathcal{H}_X,8\varepsilon,t}$ be the distribution generated by selecting $h \in \mathcal{H}_X$ uniformly at random and then taking t i.i.d. samples from $\mathcal{D}_{h,8\varepsilon}$. By Theorem 4.1, $\mathsf{TV}(\mathcal{D}_{U,t}, \mathcal{D}_{\mathcal{H}_X,8\varepsilon,t}) \leq f_{\mathsf{Paninski}}(t, 8\varepsilon, d)$. By Theorem 4.2, for every $\alpha > 0$ there exists a joint distribution $(S_U, S_{\mathcal{H}})$ such that $S_U \sim \mathcal{D}_{U,t}$, $S_{\mathcal{H}} \sim \mathcal{D}_{\mathcal{H}_X,8\varepsilon,t}$, and $\mathbb{P}[S_U \neq S_{\mathcal{H}}] \leq f_{\mathsf{Paninski}}(t, 8\varepsilon, d) + \alpha$.

For every such α and $(S_U, S_{\mathcal{H}})$, no tester can distinguish with probability strictly greater than $1/2$ between S_U and $S_{\mathcal{H}}$ in the event where $S_U = S_{\mathcal{H}}$. Hence,

$$\beta \geq 1/2 \cdot \mathbb{P}[S_U = S_{\mathcal{H}}] = 1/2 \cdot (1 - \mathbb{P}[S_U \neq S_{\mathcal{H}}]) \geq 1/2 \cdot (1 - f_{\mathsf{Paninski}}(t, 8\varepsilon, d) - \alpha).$$ [20]

Taking $\alpha \longrightarrow 0$ and rearranging yields

$$t \geq \frac{\sqrt{d \cdot \ln(1 + (4\beta - 2)^2)}}{64\varepsilon^2}. \tag{4.1}$$

This establishes a lower bound on the sample complexity for the $\mathcal{D}_U$ vs. $\{\mathcal{D}_{h,8\varepsilon} : h \in \mathcal{H}_X\}$ distribution testing problem.

Next, we show a reduction from the distribution testing problem to PAC verification of $\mathcal{H}$. Let (V, P) be a (proper or improper) interactive proof system that PAC verifies $\mathcal{H}$ such that the verifier V and honest prover P use m_V and m_P i.i.d. samples from the unknown distribution respectively, and satisfy Definition 1.7 with parameters ε and δ, as in the statement of Theorem 2.1. Using (V, P), we construct a tester T for the $\mathcal{D}_U$ vs. $\{\mathcal{D}_{h,8\varepsilon} : h \in \mathcal{H}_X\}$ testing problem. Given sample access to an unknown distribution $\mathcal{D}$ for the testing problem, T operates as follows:

1. Compute $h_V = [V(\mathcal{D}), P(\mathcal{D}_U)]$. Namely, simulate an execution of the PAC verification protocol as follows. Take a sample $S_V \sim \mathcal{D}^{m_V}$ of m_V i.i.d. samples from $\mathcal{D}$, and take a sample $S_P \sim (\mathcal{D}_U)^{m_P}$ of m_P i.i.d. samples from $\mathcal{D}_U$ (seeing as the specification of $\mathcal{D}_U$ is completely known to T, T can generate as many samples from $\mathcal{D}_U$ as necessary using uniform random coins). Execute the PAC verification protocol such that V receives input S_V, P receives input S_P, and the output of the verifier at the end of the protocol is $h_V \in \{0,1\}^{\mathcal{X}} \cup \{\mathsf{reject}\}$.
2. Take a sample $S_{\mathsf{test}} \sim \mathcal{D}^\ell$ of $\ell = \lceil \ln(24)/2\varepsilon^2 \rceil < 3/\varepsilon^2$ i.i.d. samples from $\mathcal{D}$.
3. If $(h_V = \mathsf{reject}) \vee (h_V \neq \mathsf{reject} \wedge L^{0\text{-}1}_{S_{\mathsf{test}}}(h_V) \leq {}^1\!/\!_2 - 2\varepsilon)$ then output "$\mathcal{D} \in \{\mathcal{D}_{h,8\varepsilon} : h \in \mathcal{H}_X\}$". Otherwise, output "$\mathcal{D} = \mathcal{D}_U$".

We argue that the tester T defined in this manner solves the testing problem correctly with probability at least $7/12$. If $\mathcal{D} = \mathcal{D}_U$, then $L^{0\text{-}1}_{\mathcal{D}}(h) = 1/2$ for every $h \in \{0,1\}^{\mathcal{X}}$. In particular, if

[20] More formally, this follows from Claim 8.6.

$h_V \neq \mathsf{reject}$ then $L^{\text{0-1}}_{S_{\text{test}}}(h_V) \geq 1/2 - \varepsilon$ with probability at least $11/12$ (by Hoeffding's inequality and the choice of ℓ). Thus, if $\mathcal{D} = \mathcal{D}_U$ then T outputs "$\mathcal{D} = \mathcal{D}_U$" with probability at least $11/12$.

Conversely, if $\mathcal{D} = \mathcal{D}_{h',8\varepsilon}$ for some $h' \in \mathcal{H}_X$, then $L^{\text{0-1}}_{\mathcal{D}}(h) = 1/2 - 4\varepsilon$ for $h \in \mathcal{H}$ such that $h|_X = h'$. From the correctness of the PAC verification protocol, with probability at least $2/3$, either $h_V = \mathsf{reject}$, or $L^{\text{0-1}}_{\mathcal{D}}(h_V) \leq 1/2 - 3\varepsilon$, and in that case with probability at least $11/12$, $L^{\text{0-1}}_{S_{\text{test}}}(h) \leq 1/2 - 2\varepsilon$ (again by Hoeffding's inequality and choice of ℓ). A union bound implies that if $\mathcal{D} = \mathcal{D}_{h',8\varepsilon}$ for some $h' \in \mathcal{H}_X$ then T outputs "$\mathcal{D} \in \{\mathcal{D}_{h,8\varepsilon} : h \in \mathcal{H}_X\}$" with probability at least $1 - 1/3 - 1/12 = 7/12$.

We conclude that T correctly solves the $\mathcal{D}_U$ vs. $\{\mathcal{D}_{h,8\varepsilon} : h \in \mathcal{H}_X\}$ testing problem with probability at least $7/12$ using $t = m_V + \ell$ i.i.d. samples from the unknown distribution $\mathcal{D}$. Plugging $\beta = 5/12$ in Eq. (4.1), this implies that $m_V \geq (0.005 \cdot \sqrt{d} - 3)/\varepsilon^2$, as desired. □

Remark 4.3. A previous version of this paper [24] presented a proof of an $\Omega\left(\sqrt{d}\right)$ lower bound, without the dependence on ε. That proof uses a reduction to a simpler distribution testing lower bound based on the 'birthday paradox' (instead of the Paninski bound), and it may be better suited for pedagogical expositions. □

5 Verification of unions of intervals

Theorem 5.1 (Canonne et al. [8], Theorem 1[21])**.** *Let $\varepsilon, \delta \in (0,1)$, let $n \in \mathbb{N}$, and let $\mathcal{P}, \tilde{\mathcal{P}} \in \Delta([n])$ be distributions. There exists a tolerant distribution identity tester that, given a complete description of $\tilde{\mathcal{P}}$ and $m = O\left(\sqrt{n}\log(1/\delta)\varepsilon^{-2}\right)$ i.i.d. samples from $\mathcal{P}$, satisfies the following:*

- ***Completeness.*** *If $\mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) \leq \varepsilon/\sqrt{n}$ then the tester accepts with probability at least $1 - \delta$.*
- ***Soundness.*** *If $\mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) > \varepsilon$ then the tester rejects with probability at least $1 - \delta$.*

Furthermore, the tester runs in time $\mathsf{poly}\left(n, \log(1/\delta), 1/\varepsilon\right)$.

Definition 5.2. Let $\varepsilon \in [0,1]$, let $\mathcal{X}$ be a set and let $\mathcal{F} \subseteq \{0,1\}^{\mathcal{X}}$ be a set of functions. Let $\mathcal{D} \in \Delta(\mathcal{X})$, and let $S \in \mathcal{X}^m$ for some $m \in \mathbb{N}$. We say that <u>S is an ε-sample for $\mathcal{D}$ with respect to $\mathcal{F}$</u> if

$$\forall f \in \mathcal{F} : \left|\frac{|\{x \in S : f(x) = 1\}|}{m} - \mathbb{P}_{x\sim\mathcal{D}}[f(x) = 1]\right| \leq \varepsilon.$$

Theorem 5.3 (Vapnik and Chervonenkis [36][22]; Talagrand [34][23])**.** *Let $d \in \mathbb{N}$ and $\varepsilon, \delta \in (0,1)$. Let*

[21] See also the discussion following [7, Theorem 5.4].
[22] Cf. [1, Theorem 13.4.4].
[23] Cf. [3, Theorem 4.9].

$\mathcal{X}$ be a set and let $\mathcal{F} \subseteq \{0,1\}^{\mathcal{X}}$ be a set of functions with $\mathsf{VC}(\mathcal{F}) = d$. Let $\mathcal{D} \in \Delta(\mathcal{X})$, and let $S \sim \mathcal{D}^m$, where

$$m = \Omega\left(\frac{d + \log(1/\delta)}{\varepsilon^2}\right).$$

Then with probability at least $1 - \delta$, S is an ε-sample for $\mathcal{D}$ with respect to $\mathcal{F}$.

5.1 Proof of Theorem 2.3

Proof of Theorem 2.3. We show that Protocol 1 satisfies the requirements of the theorem. From the construction of Protocol 1, it is clear that the verifier is proper. For completeness, note that if the prover follows the protocol then $\tilde{P}_{j,0} + \tilde{P}_{j,1} = 1/k$ for all j, so the verifier will never reject at the first 'if' statement. Let $\mathcal{B} = \{I_j \times \{y\} : \ j \in [k] \ \wedge \ y \in \{0,1\}\}$, and let $\mathcal{F} = \{\mathbb{1}_E : \ E \in \sigma(\mathcal{B})\} \subseteq \{0,1\}^{[0,1]\times\{0,1\}}$. In words, $\mathcal{F}$ is the set of indicator functions for events in the σ-algebra generated by $\mathcal{B}$. $\mathsf{VC}(\mathcal{F}) = 2k = O(d/\varepsilon)$, so Theorem 5.3 and the choice of m_P imply that with probability at least $1 - \delta/2$, S_P is an $\varepsilon/(6\sqrt{2k})$-sample for $\mathcal{D}$ with respect to $\mathcal{F}$. By the definitions of total variation distance and of an ε-sample, this implies that $\mathbb{P}\left[\mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) \leq \varepsilon/(6\sqrt{2k})\right] \geq 1 - \delta/2$. From the completeness of the tester of Theorem 5.1 and a union bound we conclude that with probability at least $1 - \delta$, the verifier does not reject. This establishes completeness.

For soundness, consider two cases.

- The prover is too dishonest, such that $\mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) > \varepsilon/6$. Then by the soundness of the tester of Theorem 5.1, the verifier rejects with probability at least $1 - \delta/2$.

- The prover is sufficiently honest, such that $\mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) \leq \varepsilon/6$. Then for every $h' \in \mathcal{H}_d$,

$$\begin{aligned}\left|L_{\mathcal{D}}^{0\text{-}1}(h') - L_{\tilde{\mathcal{P}}}^{0\text{-}1}(h')\right| &\leq \left|L_{\mathcal{D}}^{0\text{-}1}(h') - L_{\mathcal{P}}^{0\text{-}1}(h')\right| + \left|L_{\mathcal{P}}^{0\text{-}1}(h') - L_{\tilde{\mathcal{P}}}^{0\text{-}1}(h')\right| \\ &\leq \left|L_{\mathcal{D}}^{0\text{-}1}(h') - L_{\mathcal{P}}^{0\text{-}1}(h')\right| + \varepsilon/6, \end{aligned} \tag{5.1}$$

  where the last inequality follows from $\mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) \leq \varepsilon/6$.

  Fix $h' \in \mathcal{H}_d$. We argue that $\left|L_{\mathcal{D}}^{0\text{-}1}(h') - L_{\mathcal{P}}^{0\text{-}1}(h')\right| \leq \varepsilon/3$. Let $Q = \{x \in [0,1] : \ h'(x) \neq h'(x^*)\}$, where for each $x \in [0,1]$, we define $x^* = x_j^*$ such that $x \in I_j$. Namely, Q is the set of points for which applying the discretization procedure alters the output of h'. Then

$$\begin{aligned}\left|L_{\mathcal{D}}^{0\text{-}1}(h') - L_{\mathcal{P}}^{0\text{-}1}(h')\right| &= \left|\mathbb{P}_{(x,y)\sim\mathcal{D}}[h'(x) \neq y] - \mathbb{P}_{(x,y)\sim\mathcal{D}}[h'(x^*) \neq y]\right| \\ &= \Big|\mathbb{P}_{(x,y)\sim\mathcal{D}}[h'(x) \neq y \ \wedge \ x \in Q] \\ &\qquad - \mathbb{P}_{(x,y)\sim\mathcal{D}}[h'(x^*) \neq y \ \wedge \ x \in Q]\Big| \end{aligned} \tag{5.2}$$

$$
\begin{aligned}
&\leq \mathcal{D}(Q') && (Q' = Q \times \{0,1\}) \\
&\leq \sum_{j \in [k]:\ I_j \cap Q \neq \varnothing} \mathcal{D}(I'_j) && (I'_j = I_j \times \{0,1\}) \\
&= \sum_{j \in [k]:\ I_j \cap Q \neq \varnothing} \mathcal{P}(I'_j) && (\mathcal{D}(I'_j) = \mathcal{P}(I'_j)) \\
&= \mathcal{P}\left(\bigcup \left\{I'_j :\ I_j \cap Q \neq \varnothing\right\}\right) \\
&\leq \tilde{\mathcal{P}}\left(\bigcup \left\{I'_j :\ I_j \cap Q \neq \varnothing\right\}\right) + \mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) \\
&\leq 2d/k + \mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) && (5.3) \\
&\leq 2d/k + \varepsilon/6 = \varepsilon/3, && (5.4)
\end{aligned}
$$

where Eq. (5.2) holds since the loss of h' can differ between $\mathcal{D}$ and $\mathcal{P}$ only for points in Q; Eq. (5.3) holds because h' consists of d intervals, which together have $2d$ endpoints, $I_j \cap Q \neq \varnothing$ only if I_j contains one of these endpoints, and if the verifier did not reject then $\tilde{\mathcal{P}}(I'_j) = 1/k$ for all j; finally Eq. (5.4) holds by the assumption (in the current case) that the prover is sufficiently honest.

Combining Eq. (5.4) with Eq. (5.1) yields $\forall h' \in \mathcal{H}_d :\ \left|L_{\mathcal{D}}^{0\text{-}1}(h') - L_{\tilde{\mathcal{P}}}^{0\text{-}1}(h')\right| \leq \varepsilon/2$. This implies that a hypothesis h that has minimum loss with respect to $\tilde{\mathcal{P}}$ satisfies $L_{\mathcal{D}}^{0\text{-}1}(h) \leq L_{\mathcal{D}}^{0\text{-}1}(\mathcal{H}) + \varepsilon$.

We conclude that regardless of the prover's behavior, with probability at least $1 - \delta/2$ the verifier either rejects or outputs a hypothesis with excess loss at most ε. This establishes correctness.

We now consider the running time. Clearly, the prover runs in time polynomial in m_P. For the verifier, given that the tester of Theorem 5.1 is efficient, it remains to consider the runtime of the final step, computing a hypothesis $h \in \operatorname{argmin}_{h' \in \mathcal{H}_d} L_{\tilde{\mathcal{P}}}^{0\text{-}1}(h')$. This can be done in time $\mathsf{poly}(k) = \mathsf{poly}(d, 1/\varepsilon)$ using dynamic programming ([21, Theorem 4][24]). □

Remark 5.4. The dependence of the tolerance parameter in Theorem 5.1 on the domain size is quadratic, namely the verifier accepts if $\mathsf{TV}\left(\mathcal{P}, \tilde{\mathcal{P}}\right) \leq \varepsilon/\sqrt{n}$. Notice that this affects the sample complexity of the honest prover but not of the verifier. For instance, if the tolerance was ε/e^n instead of $\varepsilon/\sqrt{n}$, the verifier's sample complexity would remain unchanged. □

[24][21, Theorem 4] says more than what is needed in our case. It states that if for function classes $\mathcal{T}$ and $\mathcal{H}$ there is a polynomial time algorithm such that for any sample S, the algorithm finds $h \in \mathcal{H}$ such that $L_S^{0\text{-}1}(h) \leq \min_{t \in \mathcal{T}} L_S^{0\text{-}1}(t)$, then there is a polynomial time algorithm that for any sample S finds $h \in \mathsf{PW}_s(\mathcal{H})$ such that $L_S^{0\text{-}1}(h) \leq \min_{t \in \mathsf{PW}_s(\mathcal{T})} L_S^{0\text{-}1}(t)$. Here, $\mathsf{PW}_s(\mathcal{F})$ denotes the set of s-piecewise functions, namely functions $g :\ \mathbb{R} \to \mathbb{R}$ for which there exists $f_1, \ldots, f_s \in \mathcal{F}$ and intervals $I_1, \ldots, I_s \subseteq \mathbb{R}$ that form a partition of $\mathbb{R}$, such that $\forall j \in [s]\ \forall x \in I_j :\ g(x) = f_j(x)$. For our purposes, we only need the simple case where $\mathcal{H} = \mathcal{T} = \{h_0, h_1\}$, with h_b denoting the constant function $h_b(x) \equiv b$.

5.2 Protocol for unions of intervals

Assumptions:

- $d, 1/\varepsilon \in \mathbb{N}$ (this can always be achieved by making ε smaller if necessary), $k = 12d/\varepsilon$.
- $m_P = O\left(d^2\varepsilon^{-4} + d\log(1/\delta)\varepsilon^{-3}\right)$ is a multiple of k.
- $m_V = O\left(\sqrt{d}\log(1/\delta)\varepsilon^{-2.5}\right)$.
- $S_V = ((x_1^V, y_1^V), \ldots, (x_{m_V}^V, y_{m_V}^V)) \sim \mathcal{D}^{m_V}$, $S_P = ((x_1^P, y_1^P), \ldots, (x_{m_P}^P, y_{m_P}^P)) \sim \mathcal{D}^{m_P}$.
- $\mathcal{D} \in \Delta([0,1] \times \{0,1\})$ is an unknown target distribution.
- $\mathcal{H}_d$ is the class of unions of d intervals, as in Theorem 2.3.

PROVER(S_P, δ, ε):

$I_1, I_2, \ldots, I_k \leftarrow$ a partition of $[0,1]$ into disjoint intervals such that $\cup_{j\in[k]} I_j = [0,1]$ and $\forall j \in [k]: \ |\{x_1^P, \ldots, x_{m_P}^P\} \cap I_j| = m_P/k$.

for $j \in [k]$:
  for $b \in \{0,1\}$:
    $\tilde{P}_{j,b} \leftarrow |\{(x,y) \in S_P : \ x \in I_j \ \wedge \ y = b\}|/m_P$ ▷ Counted as a multiset

send $(I_1, \ldots, I_k)$ and $\left(\tilde{P}_{j,y}\right)_{j\in[k], y\in\{0,1\}}$ to the verifier

VERIFIER(S_V, δ, ε):

receive $(I_1, \ldots, I_k)$ and $\left(\tilde{P}_{j,y}\right)_{j\in[k], y\in\{0,1\}}$ from the prover

if $\exists j \in [k]$ s.t. $\tilde{P}_{j,0} + \tilde{P}_{j,1} \neq 1/k$:
  output reject and **terminate**

$x_1^*, \ldots, x_k^* \leftarrow$ arbitrary points such that $\forall j \in [k]: \ x_j^* \in I_j$

execute the tester of Theorem 5.1 and parameters $\varepsilon/6$, $\delta/2$ where the samples and distributions $\mathcal{P}, \tilde{\mathcal{P}} \in \Delta([0,1] \times \{0,1\})$ are as follows:

- $\mathcal{P}$ is the distribution generated by sampling $(x,y) \sim \mathcal{D}$ and then outputting (x^*, y) where $x^* = x_j^*$ such that $x \in I_j$
- $\tilde{\mathcal{P}}$ is the distribution such that $\mathbb{P}\left[(x_j^*, y)\right] = \tilde{P}_{j,y}$ for all $j \in [k], y \in \{0,1\}$
- The tester uses samples $S_V^* = \left((x_{j_1}^*, y_1^V), \ldots, (x_{j_{m_V}}^*, y_{m_V}^V)\right)$ where for each $i \in [m_V]$, j_i is such that $x_i^V \in I_{j_i}$

if distribution identity tester rejects:
  output reject and **terminate**

$h \leftarrow \operatorname{argmin}_{h'\in\mathcal{H}_d} L_{\tilde{\mathcal{P}}}^{0\text{-}1}(h')$

output h

Protocol 1: Verification protocol for unions of d-intervals.

Remark 5.5. Protocol 1 uses a discretization scheme where a compressed and discrete distribution $\tilde{P}$ is constructed as a representation of the true population distribution $\mathcal{D}$. The discretization scheme presented in Protocol 1 assume that the marginal distribution of $\mathcal{D}$ on the domain $[0, 1]$ has a PDF (and in particular, the prover's samples $x_1^P, \ldots, x_{m_P}^P$ are distinct). This assumption is made purely to simplify presentation; a similar discretization can be generated efficiently for any distribution $\mathcal{D} \in \Delta([0, 1] \times \{0, 1\})$.

In particular, note that there can be at most $O(d/\varepsilon)$ point masses with probability mass more than $\Omega(\varepsilon/d)$. The discretization can include an explicit description of these points (without compressing them), and give an approximate discretization (compressed description) of the remaining probability mass. The resulting discretization has support size $O(d/\varepsilon)$, and it has the required property that every union of d intervals that is optimal with respect to the discretization is $O(\varepsilon)$-close to optimal with respect to $\mathcal{D}$. □

6 Verification of statistical query algorithms

6.1 Definitions

6.1.1 Statistical query algorithms

Definition 6.1 ([20]). Let Ω be a set, let $\mathcal{D} \in \Delta(\Omega)$ be a distribution, and let $\tau \geq 0$. A statistical query is an indicator function $q : \Omega \to \{0, 1\}$.[25] An oracle $\mathcal{O}$ is a statistical query oracle for $\mathcal{D}$ with precision τ, denoted $\mathcal{O} \in \mathsf{SQ}(\mathcal{D}, \tau)$, if at each invocation, $\mathcal{O}$ takes a statistical query q as input and produces an arbitrary evaluation $\mathcal{O}(q) \in [0, 1]$ as output such that

$$\left|\mathcal{O}(q) - \mathbb{E}_{X \sim \mathcal{D}}[q(X)]\right| \leq \tau. \tag{6.1}$$

In particular, the oracle's evaluations may be adversarial and adaptive, as long as each of them satisfies Eq. (6.1).

Remark 6.2. The notion of PAC verification of an algorithm (Definition 2.4) requires that the verifier's output be competitive with $L_{\mathcal{D}}\left(A^{\mathcal{O}}\right) = \mathbb{E}[L_{\mathcal{D}}(h_A)]$, the expected loss of the output h_A of algorithm A when executed with access to oracle $\mathcal{O}$. For this expectation to be defined, throughout this paper we only consider oracles whose behavior can be described by a probability measure. In particular, oracles may be adaptive and adversarial in a deterministic or randomized manner, but they cannot be arbitrary. □

Definition 6.3. A statistical query algorithm is a (possibly randomized) algorithm A that takes no inputs and has access to a statistical query oracle $\mathcal{O}$. At each time step $t = 1, 2, 3, \ldots$:

- A chooses a finite batch $\mathbf{q}_t = \left(q_t^1, \ldots, q_t^{n_t}\right)$ of statistical queries and sends it to the oracle $\mathcal{O}$.

[25]Statistical queries are often defined as functions $q : \mathcal{X} \times \{0, 1\} \to [-1, 1]$ for some domain $\mathcal{X}$, which is a special case $\Omega = \mathcal{X} \times \{0, 1\}$ of our definition.

- $\mathcal{O}$ sends a batch of evaluations $\mathbf{v}_t = \left(v_t^1, \ldots, v_t^{n_t}\right) \in [0,1]^{n_t}$ to A, such that $v_t^i = \mathcal{O}(q_t^i)$ for all $i \in [n_t]$.
- A either produces an output and terminates, or continues to time step $t+1$.

The resulting sequence $\mathbf{r} = (\mathbf{q}_1, \mathbf{v}_1, \mathbf{q}_2, \mathbf{v}_2, \ldots)$ is called a transcript of the execution.

Note that for each t, the choice of $\mathbf{q}_t$ is a deterministic function of $(\mathbf{r}_{<t}, \rho)$, where

$$\mathbf{r}_{<t} = (\mathbf{q}_1, \mathbf{v}_1, \mathbf{q}_2, \mathbf{v}_2, \ldots, \mathbf{q}_{t-1}, \mathbf{v}_{t-1}),$$

and ρ denotes the randomness of A. If A terminates, its final output is a deterministic function of $(\mathbf{r}, \rho)$.

6.1.2 The partition size

Definition 6.4. Let Ω be a set, and let $\mathcal{S} \subseteq 2^{\Omega}$ be a collection of subsets. We say that $\mathcal{S}$ is a σ-algebra for Ω if it satisfies the following properties:

- $\Omega \in \mathcal{S}$.
- $\forall S \in \mathcal{S} : \ (\Omega \setminus S) \in \mathcal{S}$.
- For every countable sequence $S_1, S_2, \ldots \in \mathcal{S} : \ \left(\cup_{i=1}^{\infty} S_i\right) \in \mathcal{S}$.

Definition 6.5. Let Ω be a set.

- Let $\mathcal{A} \subseteq 2^{\Omega}$ be a collection of subsets. The σ-algebra generated by $\mathcal{A}$ for Ω, denoted $\sigma(\mathcal{A})$, is the intersection of all σ-algebras for Ω that are supersets of $\mathcal{A}$.
- Let $\mathcal{F} \subseteq \{0,1\}^{\Omega}$ be a set of indicator functions. The σ-algebra generated by $\mathcal{F}$ for Ω is $\sigma(\mathcal{F}) = \sigma\left(\{A \subseteq \Omega : \ \mathbb{1}_A \in \mathcal{F}\}\right)$.

Definition 6.6. Let $\mathcal{S}$ be a σ-algebra. The set of atoms of $\mathcal{S}$ is

$$\mathsf{Atoms}(\mathcal{S}) = \{S \in \mathcal{S} : \ (\forall S' \in \mathcal{S} \setminus \varnothing : \ S' \not\subset S)\}.$$ [26]

Definition 6.7. Let Ω be a set and let $\mathcal{F} = \{f_1, f_2, \ldots, f_k\} \subseteq \{0,1\}^{\Omega}$ be a finite set of indicator functions. The partition size of $\mathcal{F}$ is $\mathsf{PS}(\mathcal{F}) = |\mathsf{Atoms}(\sigma(\mathcal{F}))| \in \mathbb{N}$, i. e., the number of atoms in the σ-algebra generated by $\mathcal{F}$ for Ω.

Remark 6.8 (Connection between VC dimension and partition size)**.** Let Ω be a finite set. For any collection $\mathcal{F} \subseteq \{0,1\}^{\Omega}$, we have $\mathsf{VC}(\mathcal{F}) \leq \mathsf{PS}(\mathcal{F})$. Furthermore, $\mathsf{VC}(\mathcal{F}) = \mathsf{PS}(\mathcal{F})$ if and only if $\mathcal{F}$ is a σ-algebra.

[26] $S' \not\subset S$ denotes that S' is not a strict subset of S.

Indeed, if $x_1, x_2 \in \Omega$ belong to the same atom of $\sigma(\mathcal{F})$, then $f(x_1) = f(x_2)$ for every $f \in \mathcal{F}$. So a shattered set can contain at most one element from each atom of $\sigma(\mathcal{F})$, namely, $\mathsf{VC}(\mathcal{F}) \leq \mathsf{PS}(\mathcal{F})$.

Assume $\mathcal{F}$ is a σ-algebra. Let $A_1, \dots, A_k$, $k = \mathsf{PS}(\mathcal{F})$, be the atoms of $\mathcal{F}$, and let $x_1, \dots, x_k$ be points such that $x_i \in A_i$ for all $i \in [k]$. Then $\mathcal{F}$ shatters $x_1, \dots, x_k$. So $\mathsf{VC}(\mathcal{F}) \geq \mathsf{PS}(\mathcal{F})$.

Finally, if $\mathsf{VC}(\mathcal{F}) \geq \mathsf{PS}(\mathcal{F})$ then there exists a shattered set $\{x_1, \dots, x_k\}$ with $k = \mathsf{PS}(\mathcal{F})$. As above, each x_i belongs to a distinct atom. So without loss of generality, let $A_1, \dots, A_k$ be the atoms of $\sigma(\mathcal{F})$ such that $x_i \in A_i$ for all $i \in [k]$. Let $S \in \sigma(\mathcal{F})$. Then $S = \cup_{i \in I} A_i$ for some $I \subseteq [k]$. By choice of $\{x_1, \dots, x_k\}$, there exists $f \in \mathcal{F}$ such that $f(x_i) = \mathbb{1}(i \in I)$. Seeing as functions in $\mathcal{F}$ are constant on atoms, $f(x) = \mathbb{1}_{\cup_{i \in I} A_i} = \mathbb{1}_S$. So $\mathbb{1}_S \in \mathcal{F}$. Hence $\{\mathbb{1}_S\}_{S \in \sigma(\mathcal{F})} \subseteq \mathcal{F}$, so $\mathcal{F} = \{\mathbb{1}_S\}_{S \in \sigma(\mathcal{F})}$ is a σ-algebra. □

6.2 Efficiently manipulable statistical queries

Definition 6.9. Let Ω be a set and let $\mathcal{F}$ be a collection of functions $\Omega \to \mathbb{R}$. We say that $\mathcal{F}$ is efficiently evaluable if there exists an algorithm that, given $f \in \mathcal{F}$ and $z \in \Omega$, outputs $q(z)$ in constant time. [27]

Definition 6.10. Let Ω be a set and let $\mathcal{Q}$ be a collection of functions $\Omega \to \{0, 1\}$. We say that $\mathcal{Q}$ is efficiently manipulable if the following two conditions hold:

- *Efficient evaluation.* $\mathcal{Q}$ satisfies the property in Definition 6.9.
- *Efficient atomization.* There exists an algorithm ComputeAtoms that, for any $n \in \mathbb{N}$, takes as input a sequence $q_1, \dots, q_n \in \mathcal{Q}$, and outputs a sequence $a_1, \dots, a_t : \Omega \to \{0, 1\}$ such that $\mathsf{Atoms}(\sigma(q_1, \dots, q_n)) = (a_1, \dots, a_t)$. Furthermore, the algorithm runs in time $\mathsf{poly}(n)$.

Claim 6.11. *Let Ω be a set and let $\mathcal{Q}$ be a collection of functions $\Omega \to \{0, 1\}$. Suppose that there exists an algorithm* IsEmpty *that, for any $m \in \mathbb{N}$, takes as input a sequence $f_1, \dots, f_m$, where for each $i \in [m]$, $f_i \in \mathcal{Q}$ or $(1 - f_i) \in \mathcal{Q}$.* IsEmpty *runs in time $\mathsf{poly}(m)$ and outputs*

$$\text{IsEmpty}(f_1, \dots, f_m) = \mathbb{1}\left(\cap_{i=1}^m \{z \in \Omega : \ f_i(z) = 1\} = \varnothing \right).$$

Then $\mathcal{Q}$ satisfies the efficient atomization *property in Definition 6.10. Specifically, there exists an algorithm* ComputeAtoms *(Algorithm 2) that runs in time $O(ns)$, where n is the number of statistical queries and s is an upper bound on the number of atoms, and makes $O(ns)$ invocations of* IsEmpty.

6.3 Formal statements

Theorem 6.12 (PAC Verification of an SQ Algorithm)**.** *Let $b, s \in \mathbb{N}$, let Ω be a set, let $\mathcal{Q}$ be a collection of functions $\Omega \to \{0, 1\}$ that are efficiently manipulable (Definition 6.10), and let $\mathcal{H}$ be a discrete set.*

[27] More generally, one could consider a parameterized sequence of domains $\Omega_1, \Omega_2, \dots$, and require that the evaluation $q(z)$ for $z \in \Omega_k$ run in time $\mathsf{poly}(k)$. Or similarly, that $q(z)$ run in time $\mathsf{poly}(|z|)$, where $|z|$ is the length of the encoding of z. For simplicity, we consider a single fixed domain Ω.

Let A be a statistical query algorithm (Definition 6.3) that adaptively and randomly generates some random number T of batches $\mathbf{q}_1, \ldots, \mathbf{q}_T \in \mathcal{Q}$ of statistical queries such that with probability 1, $T \leq b$ and $\mathsf{PS}(\mathbf{q}_t) \leq s$ for each $t \in [T]$, and the algorithm outputs a random value $h \in \mathcal{H}$. Let $\mathbb{D} \subseteq \Delta(\Omega)$ be a set of distributions, let $\tau > 0$, and let $L : \Omega \times \mathcal{H} \to [0,1]$ be loss function that is efficiently evaluable (Definition 6.9).

Then there exists a collection of oracles $\mathbb{O} = \{\mathcal{O}_\mathcal{D}\}_{\mathcal{D} \in \mathbb{D}}$ where $\mathcal{O}_\mathcal{D} \in \mathsf{SQ}(\mathcal{D}, \tau)$ for all $\mathcal{D} \in \mathbb{D}$, such that algorithm A with access to oracles $\mathbb{O}$ is PAC verifiable with respect to $\mathbb{D}$ by a verification protocol (Protocol 2) that uses random samples, where the verifier and honest prover respectively use

$$m_V = \Theta\left(\frac{\sqrt{s}\log(bk/\delta)}{\tau^2} + \frac{\log(k/\delta)}{\varepsilon^2}\right),$$

and

$$m_P = \Theta\left(\frac{s^3\log(bk/\delta)}{\tau^2}\right)$$

i.i.d. samples, with $k = \lceil 8\log(4/\delta)/\varepsilon \rceil$. Furthermore, the verifier and honest prover run in time $\mathsf{poly}\big(s, t_A, 1/\varepsilon, 1/\tau, \log(1/\delta)\big)$, where t_A is the running time of algorithm A.

Remark 6.13 (On the choice of SQ oracles in Theorem 6.12)**.** In Definition 2.4, an algorithm is said to be PAC verifiable with respect to a specific collection of oracles. Namely, for each $\mathcal{D} \in \mathbb{D}$, we fix an oracle $\mathcal{O}_\mathcal{D}$, and PAC verification guarantees a loss that is ε-close to the expected loss of the algorithm when executed with that choice of oracle $\mathcal{O}_\mathcal{D}$. However, the standard definition of statistical query oracles (Definition 6.1) does not specify exactly how an SQ oracles $\mathcal{O}_\mathcal{D}$ must behave; it requires that the oracle provide evaluations that are τ-close to the correct expected value of a query, but it does not specify exactly which value to provide. This means that PAC verification of an algorithm that uses an SQ oracle (as in Definition 6.3) is not well-defined in general—rather, it is necessarily to precisely define the functionality of the SQ oracle.

In particular, if the SQ oracle is not fully specified, then some strange corner cases are possible. For instance, consider PAC verification for an SQ algorithm tasked with selecting a certain output $h \in \mathcal{H}$ from a discrete set $\mathcal{H}$ such that the "correct" output $h = h^*_\mathcal{D}$ depends on the unknown distribution $\mathcal{D}$. One technically valid pair of an SQ oracle and an SQ algorithm operate as follows. The SQ algorithm issues a single statistical query, for the constant function $q(x) \equiv 0$. The correct expectation of q is 0, yet the SQ oracle $\mathcal{O}_\mathcal{D}$ outputs some number $v \in (0, \tau)$ such that the lower order bits in the binary representation of v encode the correct output $h^*_\mathcal{D}$. The SQ algorithm simply decodes $h^*_\mathcal{D}$ from v and outputs $h^*_\mathcal{D}$. Thus, if we allow arbitrary SQ oracles then essentially any task can be solved using a trivial algorithm that uses a single SQ query, even if the SQ oracle is valid according to Definition 6.1.

To avoid these types of issues, Theorem 6.12 states that there "exists a collection of oracles $\mathbb{O} = \{\mathcal{O}_\mathcal{D}\}_{\mathcal{D} \in \mathbb{D}}$ where $\mathcal{O}_\mathcal{D} \in \mathsf{SQ}(\mathcal{D}, \tau)$" for which the algorithm is PAC verifiable. Effectively, this is a guarantee with respect to a worst-case oracle, stating that the verifier will obtain an output with loss comparable to the expected loss of executing the algorithm with the worst-possible choice of valid SQ oracles. This is captured by the notion of an ε-maximizing oracle (Definition 6.16).[28] □

Remark 6.14. In Theorem 6.12 it is assumed that $\mathsf{PS}(\mathbf{q}_t) \leq s$, namely, that the σ-algebra $\sigma(\mathbf{q}_t)$ has at most s atoms. Not every σ-algebra is atomic, but every σ-algebra generated by a finite collection of sets (including $\sigma(\mathbf{q}_t)$) is atomic. □

As a corollary of Theorem 6.12, we obtain that for statistical query algorithms of a particular type, the sample complexity of PAC verification has a quadratically lower dependence on the VC dimension of the batches of statistical queries compared to simulating the algorithm using random samples.

Corollary 6.15. *Let A be a statistical query algorithm as in Theorem 6.12, and let $d \in \mathbb{N}$. Assume that in each time step $t \in [T]$, $\mathsf{VC}(\mathbf{q}_t) = d$ and $|\mathbf{q}_t| = 2^d$. Namely, $\mathbf{q}_t$ is the set of indicator functions of a σ-algebra with d atoms. Consider an implementation of A that uses random samples to simulate the SQ oracle accessed by A, such that the implementation uses random samples only and does not use any oracles. Simulating an oracle $\mathcal{O} \in \mathsf{SQ}(\mathcal{D}, \tau)$ requires*

$$m = \Omega\left(\frac{d + \log(1/\delta)}{\tau^2}\right)$$

i.i.d. samples from $\mathcal{D}$. In contrast, there exists a protocol that PAC verifies A such that the verifier uses only

$$m_V = \Theta\left(\frac{\sqrt{d}\log(bk/\delta)}{\tau^2} + \frac{\log(k/\delta)}{\varepsilon^2}\right)$$

i.i.d. samples from $\mathcal{D}$, with $k = \lceil 8\log(4/\delta)/\varepsilon \rceil$.

The lower bound in the corollary is the standard VC lower bound.

6.4 Proofs

Definition 6.16. Let A be a statistical query algorithm, let $\mathbb{D}$ be a collection of distributions, and let $\varepsilon, \tau > 0$. We say that a collection of oracles $\mathbb{O} = \{\mathcal{O}_{\mathcal{D}}\}_{\mathcal{D}\in\mathbb{D}}$ is <u>ε-maximizing with respect to A and $\mathbb{D}$</u> if for each $\mathcal{D} \in \mathbb{D}$, $\mathcal{O}_{\mathcal{D}} \in \mathsf{SQ}(\mathcal{D}, \tau)$ and $\mathbb{E}\left[L_{\mathcal{D}}\left(A^{\mathcal{O}_{\mathcal{D}}}\right)\right] \geq \sup_{\mathcal{O}\in\mathsf{SQ}(\mathcal{D},\tau)} \mathbb{E}\left[L_{\mathcal{D}}\left(A^{\mathcal{O}}\right)\right] - \varepsilon$.

Proof of Theorem 6.12. Fix a collection of oracles $\mathbb{O} = \{\mathcal{O}_{\mathcal{D}}\}_{\mathcal{D}\in\mathbb{D}}$ that is $\varepsilon/4$-maximizing with respect to A and $\mathbb{D}$. We show that algorithm A with access to the oracles $\mathbb{O}$ is PAC verified by Protocol 2.

[28] There are of course other natural ways to define the set of oracles in PAC verification of SQ algorithms. For instance, one could consider "benign" average-case oracles, where the distribution of the output is equal to the distribution one would obtain if each query q was answered with an evaluation $v = (1/k)\sum_{i\in[k]} q(z_i)$ where $(z_1, \dots, z_k) \sim \mathcal{D}^k$ is a fresh sample of size k from $\mathcal{D}$.

To establish completeness, notice that each batch $\mathbf{a}_t$ of queries sent to the prover by VerifierIteration satisfies $\mathsf{VC}(\mathbf{a}_t) = 1$, and there are at most $b \cdot k$ such batches. Hence, by Theorem 5.3 and a union bound, taking m_P as in the statement is sufficient to guarantee that with probability at least $1 - \delta/4$,

$$\forall \text{ iteration } i \in [k]\ \forall t \in [T] :\ \|\tilde{\mathbf{p}}_t - \mathbf{p}_t\|_\infty \leq \frac{\tau}{s\sqrt{s}},$$

where $\mathbf{p}_t$ is the vector of correct evaluations, with components $p_t^j = \mathbb{E}_{Z\sim\mathcal{D}}\left[a_t^j(Z)\right]$. Hence, with probability at least $1 - \delta/4$,

$$\forall \text{ iteration } i \in [k]\ \forall t \in [T] :\ \|\tilde{\mathbf{p}}_t - \mathbf{p}_t\|_1 \leq s \cdot \|\tilde{\mathbf{p}}_t - \mathbf{p}_t\|_\infty \leq \frac{\tau}{\sqrt{s}}, \tag{6.2}$$

where we have used the fact that $\mathbf{p}_t$ and $\tilde{\mathbf{p}}_t$ are vectors of length at most s. By Eq. (6.2), Theorem 5.1, and the choice of m_V, with probability at least $1 - \delta/4$, none of the executions of IdentityTest cause the verifier to reject.

By a union bound, with probability at least $1 - \delta/2$, Eq. (6.2) holds and the verifier does not reject. Then, by Lemma 6.17,

$$\forall i \in [k] :\ \mathbb{P}\left[L_{\mathcal{D}}(h_i) \leq L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right) + \frac{\varepsilon}{2}\right] \geq \frac{\varepsilon}{8}. \tag{6.3}$$

By the choice of k,

$$\mathbb{P}\left[\forall i \in [k] :\ L_{\mathcal{D}}(h_i) > L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right) + \frac{\varepsilon}{2}\right] \leq \left(1 - \frac{\varepsilon}{8}\right)^k \leq e^{-\varepsilon k/8} \leq \frac{\delta}{4}. \tag{6.4}$$

By Hoeffding's inequality, a union bound, and the choice of m_V,

$$\mathbb{P}\left[\forall i \in [k] :\ \left|L_{S'_V}(h_i) - L_{\mathcal{D}}(h_i)\right| \leq \frac{\varepsilon}{2}\right] \geq 1 - \frac{\delta}{4}. \tag{6.5}$$

Combining Eqs. (6.2), (6.4) and (6.5) via a union bound, we conclude that with probability at least $1 - \delta$, the verifier does not reject and it outputs $h \in \mathcal{H}$ such that $L_{\mathcal{D}}(h) \leq L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right) + \varepsilon$. This establishes completeness.

To establish soundness, consider an interaction between the verifier of Protocol 2 and any deterministic or randomized (possibly malicious and computationally unbounded) prover P', and examine the following two events.

- Event I: the evaluations provided by P' satisfy

$$\forall \text{ iteration } i \in [k]\ \forall t \in [T] :\ \|\tilde{\mathbf{p}}_t - \mathbf{p}_t\|_1 \leq \tau. \tag{6.6}$$

  If the verifier does not reject then Lemma 6.17 implies that Eq. (6.3) holds. As we saw in the proof for the completeness requirement, this implies that with probability at least $1 - \delta$, the verifier outputs $h \in \mathcal{H}$ such that $L_{\mathcal{D}}(h) \leq L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right) + \varepsilon$.

- Event II: there exists an iteration $i \in [k]$ containing a time step $t^* \in [T]$ such that $\|\tilde{\mathbf{p}}_{t^*} - \mathbf{p}_{t^*}\|_1 > \tau$. By Theorem 5.1 and the choice of m_V, with probability at least $1 - \delta/4$ the verifier rejects in time step t^*.

We conclude that in both cases,

$$\mathbb{P}_{S_V \sim \mathcal{D}^{m_V}}\left[h = \mathsf{reject} \ \vee \ L_{\mathcal{D}}(h) \leq L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right) + \varepsilon\right] \geq 1 - \delta,$$

and this establishes soundness.

The runtime complexities follow from the assumptions that Q is efficiently manipulable and L is efficiently evaluable, as well as the runtime of the distribution identity tester in Theorem 5.1. ◻

Lemma 6.17. *In the context of Theorem 6.12, fix a distribution $\mathcal{D} \in \mathbb{D}$ and let $O_{\mathcal{D}} \in \mathsf{SQ}(\mathcal{D}, \tau)$ be an oracle such that*

$$\mathbb{E}\left[L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right)\right] \geq \sup_{O \in \mathsf{SQ}(\mathcal{D},\tau)} \mathbb{E}\left[L_{\mathcal{D}}\left(A^{O}\right)\right] - \varepsilon/4.$$

Consider an execution of VERIFIERITERATION *(Protocol 3). Let G denote the event in which the verifier does not reject, and the query evaluations $\tilde{\mathbf{p}}_t$ provided by the prover satisfy*

$$\forall t \in [T]: \ \ \|\tilde{\mathbf{p}}_t - \mathbf{p}_t\|_1 \leq \tau, \tag{6.7}$$

where $\mathbf{p}_t$ is the vector of correct evaluations $p_t^i = \mathbb{E}_{Z\sim\mathcal{D}}\left[a_t^i(Z)\right]$. Then the output $h_i \in \mathcal{H}$ returned by VERIFIERITERATION *satisfies*

$$\mathbb{P}\left[L_{\mathcal{D}}(h_i) \leq \mathbb{E}\left[L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right)\right] + \frac{\varepsilon}{2} \,\middle|\, G\right] \geq \frac{\varepsilon}{8}. \tag{6.8}$$

Proof. Let O_G denote the oracle with evaluations that are equal in distribution to the evaluations provided by the prover conditioned on event G occurring. By the choice of $O_{\mathcal{D}}$,

$$\mathbb{E}[L_{\mathcal{D}}(h_i) \mid G] = \mathbb{E}\left[L_{\mathcal{D}}\left(A^{O_G}\right)\right] \leq \mathbb{E}\left[L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right)\right] + \varepsilon/4.$$

By Markov's inequality,

$$\begin{aligned}\mathbb{P}\left[L_{\mathcal{D}}(h_i) > \mathbb{E}\left[L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right)\right] + \frac{\varepsilon}{2} \mid G\right] &\leq \mathbb{P}\left[L_{\mathcal{D}}(h_i) > \mathbb{E}[L_{\mathcal{D}}(h_i) \mid G] + \varepsilon/4 \,\middle|\, G\right] \\ &\leq \frac{\mathbb{E}[L_{\mathcal{D}}(h_i) \mid G]}{\mathbb{E}[L_{\mathcal{D}}(h_i) \mid G] + \varepsilon/4} \\ &\leq \frac{1}{1+\varepsilon/4},\end{aligned}$$

since $L_{\mathcal{D}}$ is at most 1. Hence, the complement satisfies

$$\mathbb{P}\left[L_{\mathcal{D}}(h_i) \leq \mathbb{E}\left[L_{\mathcal{D}}\left(A^{O_{\mathcal{D}}}\right)\right] + \frac{\varepsilon}{2} \,\middle|\, G\right] \geq \frac{\varepsilon/4}{1+\varepsilon/4} \geq \frac{\varepsilon}{8},$$

as desired. ◻

Assumptions:

- Ω is a set, $\mathcal{D} \in \Delta(\Omega)$ is the population distribution.
- A is a statistical query algorithm to be verified.
- $\tau \in (0,1)$ is the accuracy parameter for statistical queries used by A.
- $b \in \mathbb{N}$ is an upper bound on the number of statistical query batches generated by A.
- $\varepsilon, \delta \in (0,1)$ are the desired accuracy and confidence parameters for the verification.
- $k = \lceil 8 \log(4/\delta)/\varepsilon \rceil$.
- $m_V = \Theta\left(\sqrt{s}\log(bk/\delta)\tau^{-2} + \log(k/\delta)\varepsilon^{-2}\right)$.
- $m_P = \Theta\left(s^3 \log(bk/\delta)\tau^{-2}\right)$.
- $S_V, S'_V \sim \mathcal{D}^{m_V}$, $S_P \sim \mathcal{D}^{m_P}$ are independent sets of i.i.d. samples.
- $S_V = (z_1^V, \ldots, z_{m_V}^V)$, $S'_V = (z_1^{V'}, \ldots, z_{m_V}^{V'})$, $S_P = (z_1^P, \ldots, z_{m_P}^P)$.
- $L : \Omega \times \mathcal{H} \to [0,1]$ is a loss function that is efficiently evaluable.

VERIFIER(S_V, S'_V):
  for $i \in [k]$:
    $h_i \leftarrow$ VERIFIERITERATION(S_V) ▷ Protocol 3
  $i^* \leftarrow \mathrm{argmin}_{i \in [k]} L_{S'_V}(h_i)$
  output h_{i^*}

PROVER(S_P):
  loop forever:
    $q \leftarrow$ **receive** query from verifier
    $v \leftarrow \frac{1}{m_P} \sum_{i \in [m_P]} q\left(z_i^P\right)$
    send v to verifier

Protocol 2: A PAC verification protocol for statistical query algorithms.

Assumptions: As in Protocol 2.

VERIFIERITERATION(S_V):
  for $t \leftarrow 1, 2, \ldots$:
    simulate A until it sends a batch of queries or produces an output
    if A sends a batch of queries $\mathbf{q}_t$:
      if $t \geq b$:
        output reject and **terminate**
      $\mathbf{a}_t \leftarrow$ COMPUTEATOMS($\mathbf{q}_t, S_V$) ▷ Algorithm 2 and Claim 6.11
      send $\mathbf{a}_t$ to prover
      receive $\tilde{\mathbf{p}}_t$ from prover
      IDENTITYTEST($S_V, \mathbf{a}_t, \tilde{\mathbf{p}}_t, \tau$)
      $\tilde{\mathbf{v}}_t \leftarrow$ RECONSTRUCT($\mathbf{q}_t, \mathbf{a}_t, \tilde{\mathbf{p}}_t$) ▷ Protocol 4
      send $\tilde{\mathbf{v}}_t$ to A
    else if A produces output h:
      return h

IDENTITYTEST($S_V, \mathbf{a}_t, \tilde{\mathbf{p}}_t, \tau$):
  for $j \in [m_V]$:
    $i_j \leftarrow i \in [|\mathbf{a}_t|]$ such that $a_t^i(z_j^V) = 1$
  execute the distribution identity tester of Theorem 5.1
    with sample $I = (i_1, \ldots, i_{m_V})$ to distinguish with
    probability at least $1 - \varepsilon\delta/4b$ between

$$\mathsf{TV}(\tilde{\mathbf{p}}_t, \mathbf{p}_t) \leq \frac{\tau}{2\sqrt{|\mathbf{a}_t|}}, \quad \text{and} \quad \tau \leq \mathsf{TV}(\tilde{\mathbf{p}}_t, \mathbf{p}_t)$$

    where $\mathbf{p}_t$ is the distribution that generated I
  if identity tester rejects:
    output reject and **terminate**

Protocol 3: A subroutine of Protocol 2.

Assumptions:

- Ω is a set; $n, n' \in \mathbb{N}$.
- $\mathbf{q} = \left(q^1, \ldots, q^n\right)$ such that for each $i \in [n]$, $q^i : \Omega \to \{0,1\}$ is a function.
- $\mathbf{a} = \left(a^1, \ldots, a^{n'}\right)$ such that for each $j \in [n']$, $a^j : \Omega \to \{0,1\}$ is a function. Furthermore, $\mathbf{a} = \mathsf{Atoms}(\sigma(\mathbf{q}))$, so $\forall\, i \in [n]\ \exists\, \mathcal{J}(q^i) \subseteq [n'] :\ q^i = \sum_{j \in \mathcal{J}(q_i)} a^j$.
- $\mathbf{p} = \left(p^1, \ldots, p^{n'}\right) \in [0,1]^{n'}$

RECONSTRUCT($\mathbf{q}, \mathbf{a}, \mathbf{p}$):
  for $i \in [n]$:
    $v^i \leftarrow \sum_{j \in \mathcal{J}(q^i)} p^j$
  return $(v^1, \ldots, v^n)$

Protocol 4: A subroutine of Protocol 3. Compute the evaluations for a collection of statistical queries $\mathbf{q}$ with atoms $\mathbf{a}$ given evaluations $\mathbf{p}$ for the atoms.

7 Discussion and future work

In this paper, we have shown that $\Omega\left(\sqrt{d}\right)$ samples are necessary for PAC verifying a class of VC dimension d, and furthermore, for some classes $O\left(\sqrt{d}\right)$ samples are sufficient. In contrast, [15, Lemma 4.1] states that there also exist VC classes where the sample complexity for verification is $\widetilde{\Omega}(d)$ under the assumption that the verifier is proper (outputs a hypothesis from the class), and we believe it is likely that there exist VC classes for which an $\widetilde{\Omega}(d)$ lower bound holds for every verifier.

Hence, it appears likely that the VC dimension does not characterize the sample complexity of PAC verification. In that case, finding an alternative combinatorial quantity that does characterize that sample complexity is an exciting open problem.

A potentially easier problem is to devise upper bounds (PAC verification protocols) for specific classes of interest. For example, the main property of the thresholds class utilized in the proof of Theorem 2.3 is that it has low 'surface area' or noise sensitivity (cf. [4]). Perhaps a similar proof technique could apply to other classes as well.

Additionally, we introduced a notion of PAC verification of an algorithm. We believe this is very natural definition, because many of the algorithms that people might like to delegate in practice are not PAC learners, including unsupervised learning algorithms (e. g., clustering and dimensionality reduction algorithms), and supervised algorithms that are not provably PAC learners (e. g., neural networks trained via SGD). Devising PAC verification protocols for specific algorithms of interest could be a rewarding endeavor.

8 Appendix

8.1 Approximation algorithm for Vertex Cover

Assumptions:

- $n \in \mathbb{N}$, $V = [n]$, $G = (V, E)$ is an undirected graph.

GREEDYVERTEXCOVER(G):

$M \leftarrow \varnothing$

for $e \in E$ in lexicographic order: ▷ Greedily construct a maximal matching
  if $\forall e' \in M : e \cap e' = \varnothing$:
    $M \leftarrow M \cup \{e\}$

$S \leftarrow \bigcup_{e \in M} e$

output S

Algorithm 1: A greedy 2-approximation algorithm for the vertex cover problem.

Claim 8.1. *Let $n \in \mathbb{N}$, $V = [n]$, and let $G = (V, E)$ be an undirected graph. Let S be the set that is generated when executing* GREEDYVERTEXCOVER (Algorithm 1) on G. Then, S is a valid vertex cover for G, and $|S| \leq 2 \cdot \mathsf{Opt}$, where Opt is the size of a minimum vertex cover of G.

Proof of Claim 8.1. Notice that every edge $e \in E$ is either in M or adjacent to at least one edge in M (otherwise, we would have added e to M). Hence, every edge in E is incident to at least one vertex in $(\cup_{e \in M} e) = S$. So S is a vertex cover.

Let S^* be a minimum vertex cover. Because S^* is a vertex cover, every edge in M is incident to at least one vertex in S^*. Because M is a matching, every edge in M is incident to a unique vertex in S^*, namely, there exists an injective function $M \to S^*$. Hence, $\frac{1}{2}|S| = |M| \leq |S^*| = \mathsf{Opt}$. □

Claim 8.2. *Let $n, k \in \mathbb{N}$, $k \geq 1$, $V = [n]$, and let $E, \widetilde{E} \subseteq \binom{V}{2}$ be two sets of edges that differ by at most k edges ($|E \Delta \widetilde{E}| \leq k$). Let M and $\widetilde{M}$ be the maximal matchings that are generated when executing* GREEDYVERTEXCOVER (Algorithm 1) on $G = (V, E)$ and $\widetilde{G} = (V, \widetilde{E})$, respectively. Then, $|\widetilde{M}| \leq |M| + k$.

Proof of Claim 8.2. We prove the claim by induction. For the base case, assume $k = 1$. Let e^* denote the unique edge that has changed, i. e., $\{e^*\} = E \Delta \widetilde{E}$. The decision in GREEDYVERTEXCOVER of whether to add an edge e to the matching M is made *locally*, based on whether any of the edges adjacent to e are already included in M. Hence, the collection $M^* = \widetilde{M} \Delta M$ of edges whose inclusion in the matching changed due to adding or removing the edge e^* has the property that $G^* = (V, E^*)$ is a connected graph, where $E^* = M^* \cup \{e^*\}$. Furthermore, if two edges $e, e' \in E^*$ share a vertex $v \in e \cap e'$, then at most one of e and e' can be in each matching, namely, $|\{e, e'\} \cap M| \leq 1$ and $|\{e, e'\} \cap \widetilde{M}| \leq 1$. Hence, the maximum degree in G^* is at most 2, that is, G^* is a path or a cycle. Furthermore, G^* is alternating. Namely, if $E^* = \{e_1, e_2, ..., e_t\}$ such that $|e_i \cap e_{i+1}| = 1$ for all $i \in [t-1]$, then $\widetilde{M}$ is the set of odd-indexed edges in E^* and M is the set of even-indexed edges, or vice versa. Hence, $|\widetilde{M}| \leq |M| + 1$. This concludes the base case $k = 1$.

For the induction step, assume the claim holds for $k-1$. Let $E' \subseteq \binom{V}{2}$ be an "intermediate" collection of edges such that $|E \Delta E'| \leq k-1$ and $|E' \Delta \widetilde{E}| \leq 1$. Let M' be the matching generated when running GreedyVertexCover on $G' = (V, E')$. Then $|\widetilde{M}| \leq |M'| + 1 \leq |M| + k$, where the first inequality follows from the base case and the second inequality follows from the induction hypothesis. Hence, the claim holds for k. □

Claim 8.3. *Let $n, k \in \mathbb{N}$, $V = [n]$, and let $E, \widetilde{E} \subseteq \binom{V}{2}$ be two sets of edges that differ by at most k edges ($|E \Delta \widetilde{E}| \leq k$). Let S and $\widetilde{S}$ be the vertex covers that are generated when executing* GreedyVertexCover (Algorithm 1) on $G = (V, E)$ and $\widetilde{G} = (V, \widetilde{E})$, respectively. Then, $|\widetilde{S}| \leq |S| + 2k$.

Proof of Claim 8.3. Let M and $\widetilde{M}$ be the maximal matchings that are generated when executing GreedyVertexCover on G and $\widetilde{G}$, respectively. Then

$$|\widetilde{S}| = 2|\widetilde{M}| \leq 2(|M| + k) = |S| + 2k,$$

where the inequality follows from Claim 8.2. □

8.2 Computing the atoms of a set of statistical queries

Algorithm 2 below computes $\mathsf{Atoms}(\sigma(\mathbf{q}))$, i. e., the atoms of a set of statistical queries $\mathbf{q}$. It assumes that there is an efficient procedure IsEmpty for checking whether the intersection of a set of queries is empty, as in Claim 6.11.

Algorithm 2 runs in time $O(ns)$, where n is the number of statistical queries and s is an upper bound on the number of atoms, and makes $O(ns)$ invocations of IsEmpty.

Proof hints for Claim 6.11. Correctness of the algorithm is not hard to see by induction on the number n of queries. For the running time, note that the set A will never contain more than s elements, and so the runtime and number of invocations of IsEmpty are at most $O(ns)$. □

Remark 8.4. Note that there are simple classes of functions that do not have an efficient procedure IsEmpty (unless $\mathsf{P} = \mathsf{NP}$). For instance, consider the class of all functions $\{0,1\}^k \to \{0,1\}$ that are an OR clause with three literals, i. e., $f(x) = \ell_1 \vee \ell_2 \vee \ell_3$ where $\ell_i = x_j$ or $\ell_i = (1 - x_j)$ for some $j \in [k]$. Deciding whether the intersection of such functions f is empty is equivalent to solving the UnSAT problem for 3-CNFs, which is CoNP-complete. □

Assumptions:

- $n \in \mathbb{N}$, Ω is a set.
- $\mathcal{Q}$ is a collection of function $\Omega \to \{0,1\}$.
- IsEmpty is a procedure for testing whether the intersection of a set of functions from $\mathcal{Q}$ is empty, as in Claim 6.11.
- $\mathbf{q} = \left(q^1, \ldots, q^n\right) \in \mathcal{Q}^n$.

ComputeAtoms($\mathbf{q}$):

$A \leftarrow \{\mathbb{1}_\Omega\}$

for $i \in [n]$:
  $A' \leftarrow \varnothing$
  for $a \in A$:
    if IsEmpty(a, q^i) $\vee$ IsEmpty$(a, 1 - q^i)$
      $A' \leftarrow A' \cup \{a\}$
    else
      $A' \leftarrow A' \cup \{aq^i, a(1-q^i)\}$
  $A \leftarrow A'$

output A

Algorithm 2: An algorithm for computing the atoms of a set of statistical queries.

8.3 Concentration of measure

Theorem 8.5 (Hoeffding [19]). *Let $a, b, \mu \in \mathbb{R}$ and $m \in \mathbb{N}$. Let $Z_1, \ldots, Z_m$ be a sequence of i.i.d. real-valued random variables and let $Z = \frac{1}{m}\sum_{i=1}^m Z_i$. Assume that $\mathbb{E}[Z] = \mu$, and for every $i \in [m]$, $\mathbb{P}[a \le Z_i \le b] = 1$. Then, for every $\varepsilon > 0$,*

$$\mathbb{P}[|Z - \mu| > \varepsilon] \le 2\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right).$$

8.4 Le Cam's method

The following is a simple instantiation of Le Cam's method for proving a lower bound for hypothesis testing. This version is similar to [11, Proposition 2.3.1]. More advanced versions can be found in [37, Lemma 1], and in [23, Chapter 16].

Claim 8.6. *Let Ω be a set, and let $\mathcal{P}_0, \mathcal{P}_1 \in \Delta(\Omega)$ be distributions. Let T be a (possibly randomized) mapping $\Omega \to \{0,1\}$. Let*

$$\mathrm{error}(T) = \max\left\{\mathbb{P}_{Z\sim\mathcal{P}_0}[T(Z) = 1], \mathbb{P}_{Z\sim\mathcal{P}_1}[T(Z) = 0]\right\}.$$

Then

$$\mathrm{error}(T) \geq \frac{1}{2}\left(1 - \mathsf{TV}(\mathcal{P}_0, \mathcal{P}_1)\right).$$

Acknowledgments

An initial version of the lower bound in Theorem 2.1 resulted from a conversation with Lijie Chen and Guy Rothblum.

We thank the anonymous *Theory of Computing* reviewers for many insightful comments and suggestions. In particular, they referred us to the lower bound of Theorem 3.7, and pointed out that our Theorem 6.12 can circumvent this lower bound. We added Section 3.3 thanks to their suggestion. Overall, the paper has benefitted significantly from their contributions.

JS would like to thank Shafi Goldwasser, Steve Hanneke, Bobby Kleinberg, Shay Moran, Ido Nachum, Guy Rothblum and Abhishek Shetty for helpful comments and suggestions. Part of this work was done while JS was visiting the Weizmann Institute of Science (hosted by Guy Rothblum), Cornell University (hosted by Bobby Kleinberg) and the Technion (hosted by Shay Moran). JS is grateful for their hospitality and support.

This work was supported in part by DARPA (Defense Advanced Research Projects Agency) contract #HR001120C0015, and the Simons Collaboration on the Theory of Algorithmic Fairness. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Simons Foundation or DARPA.

References

[1] NOGA ALON AND JOEL H. SPENCER: *The Probabilistic Method*. Wiley, 2nd edition, 2000. [doi:10.1002/0471722154] 19

[2] CEM ANIL, GUODONG ZHANG, YUHUAI WU, AND ROGER B. GROSSE: Learning to give checkable answers with prover-verifier games, 2021. [arXiv:2108.12099] 4

[3] MARTIN ANTHONY AND PETER L. BARTLETT: *Neural Network Learning: Theoretical Foundations*. Cambridge Univ. Press, 2002. [doi:10.1017/CBO9780511624216] 19

[4] MARIA-FLORINA BALCAN, ERIC BLAIS, AVRIM BLUM, AND LIU YANG: Active property testing. In *Proc. 53rd FOCS*, pp. 21–30. IEEE Comp. Soc., 2012. [doi:10.1109/FOCS.2012.64] 32

[5] RAN CANETTI AND ARI KARCHMER: Covert learning: How to learn with an untrusted intermediary. In KOBBI NISSIM AND BRENT WATERS, editors, *Proc. Theory of Cryptography Conf. (TCC'21)*, pp. 1–31. Springer, 2021. [doi:10.1007/978-3-030-90456-2_1] 4

[6] CLÉMENT L. CANONNE: A short note on learning discrete distributions, 2020. [arXiv:2002.11457] 11

[7] Clément L. Canonne: A survey on distribution testing: Your data is big. But is it blue? *Theory of Computing Library Graduate Surveys*, 9:1–100, 2020. [doi:10.4086/toc.gs.2020.009] 19

[8] Clément L. Canonne, Ayush Jain, Gautam Kamath, and Jerry Li: The price of tolerance in distribution testing. In *Proc. 35th Ann. Conf. on Learning Theory (COLT'22)*, pp. 573–624. MLR Press, 2022. PMLR vol. 178. 19

[9] Matthias C. Caro, Marcel Hinsche, Marios Ioannou, Alexander Nietner, and Ryan Sweke: Classical verification of quantum learning. In *Proc. 14th Innovations in Theoret. Comp. Sci. Conf. (ITCS'23)*, pp. 24:1–23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2023. [doi:10.4230/LIPIcs.ITCS.2024.24, arXiv:2306.04843] 4

[10] Alessandro Chiesa and Tom Gur: Proofs of proximity for distribution testing. In *Proc. 9th Innovations in Theoret. Comp. Sci. Conf. (ITCS'18)*, pp. 53:1–14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. [doi:10.4230/LIPIcs.ITCS.2018.53] 4, 11

[11] John Duchi: Lecture Notes on Statistics and Information Theory, 2024. Available at Stanford. Accessed: 2024-11-26. 35

[12] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth: Preserving statistical validity in adaptive data analysis, 2014. [arXiv:1411.2664] 15

[13] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth: Preserving statistical validity in adaptive data analysis. In *Proc. 47th STOC*, pp. 117–126. ACM Press, 2015. [doi:10.1145/2746539.2746580] 15, 17

[14] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum: Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–64, 2015. [doi:10.1145/2699436] 2, 13

[15] Shafi Goldwasser, Guy N. Rothblum, Jonathan Shafer, and Amir Yehudayoff: Interactive proofs for verifying machine learning. In *Proc. 12th Innovations in Theoret. Comp. Sci. Conf. (ITCS'21)*, pp. 41:1–19. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021 (virtual conference). [doi:10.4230/LIPIcs.ITCS.2021.41] 2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 32

[16] Moritz Hardt and Jonathan R. Ullman: Preventing false discovery in interactive data analysis is hard. In *Proc. 55th FOCS*, pp. 454–463. IEEE Comp. Soc., 2014. [doi:10.1109/FOCS.2014.55] 15

[17] David Haussler: Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inform. Comput.*, 100(1):78–150, 1992. [doi:10.1016/0890-5401(92)90010-D] 2

[18] Tal Herman and Guy N. Rothblum: Verifying the unseen: Interactive proofs for label-invariant distribution properties. In *Proc. 54th STOC*, pp. 1208–1219. ACM Press, 2022. [doi:10.1145/3519935.3519987] 4

[19] WASSILY HOEFFDING: Probability inequalities for sums of bounded random variables. *J. Amer. Statistical Assoc.*, 58(301):13–30, 1963. [doi:10.2307/2282952] 35

[20] MICHAEL J. KEARNS: Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. [doi:10.1145/293347.293351] 23

[21] MICHAEL J. KEARNS, ROBERT E. SCHAPIRE, AND LINDA SELLIE: Toward efficient agnostic learning. *Machine Learning*, 17(2–3):115–141, 1994. [doi:10.1007/BF00993468] 21

[22] SUBHASH KHOT AND ODED REGEV: Vertex cover might be hard to approximate to within $2 - \varepsilon$. *J. Comput. System Sci.*, 74(3):335–349, 2008. Preliminary version in CCC'03. [doi:10.1016/J.JCSS.2007.06.019] 12

[23] LUCIEN LE CAM: *Asymptotic Methods in Statistical Decision Theory*. Springer, 1986. [doi:10.1007/978-1-4612-4946-7] 35

[24] SAACHI MUTREJA AND JONATHAN SHAFER: PAC verification of statistical algorithms, 2022. [arXiv:2211.17096v1] 19

[25] SAACHI MUTREJA AND JONATHAN SHAFER: PAC verification of statistical algorithms. In GERGELY NEU AND LORENZO ROSASCO, editors, *Proc. 36th Ann. Conf. on Learning Theory (COLT'23)*, volume 195, pp. 5021–5043. MLR Press, 2023. PMLR vol. 195. 1

[26] MICHAEL NGO AND MICHAEL P. KIM: Publicly-verifiable certificates for statistical algorithms, 2026. Unpublished manuscript, see blog post. 4

[27] LIAM PANINSKI: A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Trans. Inform. Theory*, 54(10):4750–4755, 2008. [doi:10.1109/TIT.2008.928987] 7, 17

[28] ALFRÉD RÉNYI: On measures of entropy and information. In JERZY NEYMAN, editor, *Proc. 4th Berkeley Symp. on Math. Statistics and Probability, Vol. 1: Contributions to the Theory of Statistics*, volume 4, pp. 547–561. Univ. California Press, 1961. Available at Project Euclid. 11

[29] SEBASTIEN ROCH: *Modern Discrete Probability: An Essential Toolkit*. Cambridge Univ. Press, 2023. [doi:10.1017/9781009305129] 17

[30] RONITT RUBINFELD AND ARSEN VASILYAN: Testing distributional assumptions of learning algorithms. In *Proc. 55th STOC*, pp. 1643–1656. ACM Press, 2023. [doi:10.1145/3564246.3585117, arXiv:2204.07196] 4

[31] SANJIT A. SESHIA, DORSA SADIGH, AND S. SHANKAR SASTRY: Toward verified artificial intelligence. *Comm. ACM*, 65(7):46–55, 2022. [doi:10.1145/3503914] 4

[32] SHAI SHALEV-SHWARTZ AND SHAI BEN-DAVID: *Understanding Machine Learning: From Theory to Algorithms*. Cambridge Univ. Press, 2014. [doi:doi.org/10.1017/CBO9781107298019] 2

[33] THOMAS STEINKE AND JONATHAN R. ULLMAN: Interactive fingerprinting codes and the hardness of preventing false discovery. In *Proc. 28th Ann. Conf. on Learning Theory (COLT'15)*, pp.

1588–1628. Springer, 2015. PMLR vol. 40. 15, 16

[34] MICHEL TALAGRAND: Sharper bounds for Gaussian and empirical processes. *Ann. Probab.*, 22(1):28–76, 1994. [doi:10.1214/aop/1176988847] 19

[35] LESLIE G. VALIANT: A theory of the learnable. *Comm. ACM*, 27(11):1134–1142, 1984. [doi:10.1145/1968.1972] 2

[36] VLADIMIR N. VAPNIK AND ALEXEI YA. CHERVONENKIS: On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Appl.*, 16(2):264–280, 1971. Russian original at Teor. Veroyatnostei i ee Primeneniya. Preliminary version in Dokl. Akad. Nauk SSSR 181 (1968), 781–783. [doi:10.1137/1116025] 2, 19

[37] BIN YU: Assouad, Fano, and Le Cam. In *Festschrift for Lucien Le Cam: Research papers in probability and statistics*, pp. 423–435. Springer, 1997. [doi:10.1007/978-1-4612-1880-7_29] 35

AUTHORS

Saachi Mutreja
Department of Computer Science
Columbia University
New York, NY, USA
sm5540@columbia.edu
https://sites.google.com/view/saachimutreja

Jonathan Shafer
Computer Science and Artificial Intelligence Laboratory
Schwarzman College of Computing
Massachusetts Institute of Technology
Cambridge, MA, USA
shaferjo@mit.edu
https://shaferjo.com

ABOUT THE AUTHORS

SAACHI MUTREJA is currently a Ph. D. student at Columbia, advised by Henry Yuen. She received her bachelor's in EECS from UC Berkeley in 2023. This paper was written while she was an undergraduate at Berkeley. Her research interests are in quantum cryptography, post-quantum cryptography and quantum complexity.

JONATHAN SHAFER is a Postdoctoral Associate at MIT, working with Vinod Vaikuntanathan. He earned a Ph. D. from UC Berkeley, advised by Shafi Goldwasser. This paper was written while he was at Berkeley and then a visiting Ph. D. student at Cornell. Previously he received an M. Sc. from Tel Aviv University, where he was advised by Amir Yehudayoff and Amir Shpilka. He completed his undergraduate studies at the Lautman Interdisciplinary Program. Jonathan has taught high school students in Jamaica and graduate students in Rwanda (JamCoders, A-PIC Summer School). Recently, he cycled along the EuroVelo 7 from Prague to Berlin.