Interactive Proofs for Verifying Machine Learning

Shafi Goldwasser
University of California, Berkeley, CA, USA
https://simons.berkeley.edu/people/shafi-goldwasser
shafi.goldwasser@berkeley.edu

Guy N. Rothblum
Weizmann Institute of Science, Rehovot, Israel
https://guyrothblum.wordpress.com
rothblum@alum.mit.edu

Jonathan Shafer
University of California, Berkeley, CA, USA
https://shaferjo.com
shaferjo@berkeley.edu

Amir Yehudayoff
Technion – Israel Institute of Technology, Haifa, Israel
https://yehudayoff.net.technion.ac.il
amir.yehudayoff@gmail.com

Abstract

We consider the following question: using a source of labeled data and interaction with an untrusted prover, what is the complexity of verifying that a given hypothesis is "approximately correct"? We study interactive proof systems for *PAC verification*, where a verifier that interacts with a prover is required to accept good hypotheses, and reject bad hypotheses. Both the verifier and the prover are efficient and have access to labeled data samples from an unknown distribution. We are interested in cases where the verifier can use significantly less data than is required for (agnostic) PAC learning, or use a substantially cheaper data source (e.g., using only random samples for verification, even though learning requires membership queries). We believe that today, when data and data-driven algorithms are quickly gaining prominence, the question of verifying purported outcomes of data analyses is very well-motivated.

We show three main results. First, we prove that for a specific hypothesis class, verification is significantly cheaper than learning in terms of sample complexity, even if the verifier engages with the prover only in a single-round (NP-like) protocol. Moreover, for this class we prove that single-round verification is also significantly cheaper than testing closeness to the class. Second, for the broad class of Fourier-sparse boolean functions, we show a multi-round (IP-like) verification protocol, where the prover uses membership queries, and the verifier is able to assess the result while only using random samples. Third, we show that verification is not always more efficient. Namely, we show a class of functions where verification requires as many samples as learning does, up to a logarithmic factor.

2012 ACM Subject Classification Theory of computation → Interactive proof systems; Theory of computation → Machine learning theory

Keywords and phrases PAC learning, Fourier analysis of boolean functions, Complexity gaps, Complexity lower bounds, Goldreich-Levin algorithm, Kushilevitz-Mansour algorithm, Distribution testing

Digital Object Identifier 10.4230/LIPIcs.ITCS.2021.41

Related Version A full version of the paper is available at https://eccc.weizmann.ac.il/report/2020/058.

Funding *Shafi Goldwasser*: Research was supported in part by DARPA under Contract No. HR001120C0015.
Guy N. Rothblum: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702).
Amir Yehudayoff: Research was supported by ISF grant No. 1162/15.

Acknowledgements The authors would like to thank Oded Ben-David, Eyal Ben-David, Alessandro Chiesa, Constantinos Daskalakis, Oded Goldreich, Ankur Moitra, Ido Nachum, Orr Paradise and Avishay Tal for insightful discussions and helpful references.

A simple idea underpins science: "trust, but verify". Results should always be subject to challenge from experiment. That simple but powerful idea has generated a vast body of knowledge. Since its birth in the 17th century, modern science has changed the world beyond recognition, and overwhelmingly for the better. But success can breed complacency. Modern scientists are doing too much trusting and not enough verifying – to the detriment of the whole of science, and of humanity.

The Economist, "How Science Goes Wrong" (2013)

1 Introduction

Data and data-driven algorithms are transforming science and society. State-of-the-art machine learning and statistical analysis algorithms use access to data at scales and granularities that would have been unimaginable even a few years ago. From medical records and genomic information to financial transactions and transportation networks, this revolution spans scientific studies, commercial applications and the operation of governments. It holds transformational promise, but also raises new concerns. If data analysis requires huge amounts of data and computational power, how can one verify the correctness and accuracy of the results? Might there be asymmetric cases, where performing the analysis is expensive, but verification is cheap?

There are many types of statistical analyses, and many ways to formalize the notion of verifying the outcome. In this work we focus on interactive proof systems [12] for verifying supervised learning, as defined by the PAC model of learning [22]. Our emphasis throughout is on access to the underlying data distribution as the critical resource: both quantitatively (how many samples are used for learning versus for verification), and qualitatively (what types of samples are used). We embark on tackling a series of new questions:

Suppose a learner (which we also call *prover*) claims to have arrived at a good hypothesis with regard to an unknown data distribution by analyzing random samples from the distribution. Can one verify the quality of the hypothesis with respect to the unknown distribution by using significantly fewer samples than the number needed to independently repeat the analysis? The crucial difference between this question and questions that appear in the property testing and distribution testing literature is that we allow the prover and verifier to engage in an *interactive* communication protocol (see Section 1.1.1 for a comparison). We are interested in the case where both the verifier and an honest prover are efficient (i.e., use polynomial runtime and sample complexity), and furthermore, a dishonest prover with unbounded computational resources cannot fool the verifier:

► **Question 1 (Runtime and sample complexity of learning vs. verifying).** *Are there machine learning tasks for which the runtime and sample complexity of learning a good hypothesis is significantly larger than the complexity of verifying a hypothesis provided by someone else?*

In the learning theory literature, various types of access to training data have been considered, such as random samples, membership queries, and statistical queries. In the real world, some types of access are more costly than others. Therefore, it is interesting to consider whether it is possible to verify a hypothesis using a cheaper type of access than is necessary for learning:

▶ **Question 2 (Sample type of learning vs. verifying).** *Are there machine learning problems where membership queries are necessary for finding a good hypothesis, but verification is possible using random samples alone?*

The answers to these fundamental questions are motivated by real-world applications. If data analysis requires huge amounts of data and computational resources while verification is a simpler task, then a natural approach for individuals and weaker entities would be to delegate the data collection and analysis to more powerful entities. Going beyond machine learning, this applies also to verifying the results of scientific studies without replicating the entire experiment. We elaborate on these motivating applications in Section 1.2 below.

1.1 PAC Verification: A Proposed Model

Our primary focus in this work is verifying the results of agnostic supervised machine learning algorithms that receive a labeled dataset, and aim to learn a classifier that predicts the labels of unseen examples. We introduce a notion of interactive proof systems for verification of PAC learning, which we call *PAC Verification* (see Definition 4). Here, the entity running the learning algorithms (which we refer to as the *prover* or the *learner*) proves the correctness of the results by engaging in an interactive communication protocol with a verifier. One special case is where the prover only sends a single message constituting an (NP-like) certificate of correctness. The honest prover should be able to convince the verifier to accept its proposed hypothesis with high probability. A dishonest prover (even an unbounded one) should not be able to convince the verifier to accept a hypothesis that is not sufficiently good (as defined below), except with small probability over the verifier's random coins and samples. The proof system is interesting if the amount of resources used for verification is significantly smaller than what is needed for performing the learning task. We are especially interested in *doubly-efficient* proof systems [11], where the honest prover also runs in polynomial time.

More formally, let $\mathcal{X}$ be a set, and consider a distribution $\mathcal{D}$ over samples of the form (x, y) where $x \in \mathcal{X}$ and $y \in \{0, 1\}$. Assume there is some *hypothesis class* $\mathcal{H}$, which is a set of functions $\mathcal{X} \to \{0, 1\}$, and we are interested in finding a function $h \in \mathcal{H}$ that predicts the label y given a previously unseen x with high accuracy with respect to $\mathcal{D}$. To capture this we use the *loss function* $L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\in\mathcal{D}}[h(x) \neq y]$. Our goal is to design protocols consisting of a prover and verifier that satisfy: (i) When the verifier interacts with an honest prover, with high probability the verifier outputs a hypothesis h that is *ε-good*, meaning that

$$L_{\mathcal{D}}(h) \leq L_{\mathcal{D}}(\mathcal{H}) + \varepsilon, \tag{1}$$

where $L_{\mathcal{D}}(\mathcal{H}) = \inf_{f\in\mathcal{H}} L_{\mathcal{D}}(f)$; (ii) For any (possibly dishonest and unbounded) prover, the verifier can choose to reject the interaction, and with high probability the verifier will not output a hypothesis that is not *ε-good*.

Observe that in the *realizable case* (or promise case), where we assume that $L_{\mathcal{D}}(\mathcal{H}) = 0$, one immediately obtains a strong result: given a hypothesis $\tilde{h}$ proposed by the prover, a natural strategy for the verifier is to take a few samples from $\mathcal{D}$, and accept if and only if $\tilde{h}$ classifies at most, say, an $\frac{9}{10}\varepsilon$-fraction of them incorrectly. From Hoeffding's inequality, taking $O\left(\frac{1}{\varepsilon^2}\right)$ samples is sufficient to ensure that with probability at least $\frac{9}{10}$ the *empirical loss*[1] of $\tilde{h}$ is $\frac{\varepsilon}{10}$-close to the true loss. Therefore, if $L_{\mathcal{D}}(\tilde{h}) \leq \frac{8}{10}\varepsilon$ then $\tilde{h}$ is accepted with probability at least $\frac{9}{10}$, and if $L_{\mathcal{D}}(\tilde{h}) > \varepsilon$ then $\tilde{h}$ is rejected with probability at least $\frac{9}{10}$. In contrast, PAC learning a hypothesis that with probability at least $\frac{9}{10}$ has loss at most ε requires $\Omega\left(\frac{d}{\varepsilon}\right)$ samples, where the parameter d, which is the VC dimension of the class, can be arbitrarily large.[2] That is, in the realizable case there is a sample complexity and time complexity separation of unbounded magnitude between learning and verifying. Furthermore, this result holds also under the weaker assumption that $L_{\mathcal{D}}(\mathcal{H}) \leq \frac{\varepsilon}{2}$.

Encouraged by this strong result, the present paper focuses on the *agnostic case*, where no assumptions are made regarding $L_{\mathcal{D}}(\mathcal{H})$. Here, things become more interesting, and deciding whether a proposed hypothesis $\tilde{h}$ is ε-good is non-trivial. Indeed, the verifier can efficiently estimate $L_{\mathcal{D}}(\tilde{h})$ using Hoeffding's inequality as before, but estimating the term $L_{\mathcal{D}}(\mathcal{H})$ on the right hand side of (1) is considerably more challenging. If $\tilde{h}$ has a loss of say 15%, it could be an amazingly-good hypothesis compared to the other members of $\mathcal{H}$, or it could be very poor. Distinguishing between these two cases may be difficult when $\mathcal{H}$ is a large and complicated class.

1.1.1 Related Models

We discuss two related models studied in prior work, and their relationship to the PAC verification model proposed in this work.

Property Testing. Goldreich, Goldwasser and Ron [9] initiated the study of a property testing problem that naturally accompanies proper PAC learning: Given access to samples from an unknown distribution $\mathcal{D}$, decide whether $L_{\mathcal{D}}(\mathcal{H}) = 0$ or $L_{\mathcal{D}}(\mathcal{H}) \geq \varepsilon$ for some fixed hypothesis class $\mathcal{H}$. Further developments and variations appeared in Kearns and Ron [15] and Balcan et. al [2]. Blum and Hu [4] consider *tolerant* closeness testing and a related task of distance approximation (see [19]), where the algorithm is required to approximate $L_{\mathcal{D}}(\mathcal{H})$ up to a small additive error. As discussed above, the main challenge faced by the verifier in PAC verification is approximating $L_{\mathcal{D}}(\mathcal{H})$. However, there is a crucial difference between testing and PAC verification: In addition to taking samples from $\mathcal{D}$, the verifier in PAC verification can also interact with a prover, and thus PAC verification can (potentially) be easier than testing. Indeed, this difference is exemplified by the *proper* testing question, where we only need to distinguish the zero-loss case from large loss. As discussed above, proper PAC verification is trivial. Proper testing, one the other hand, can be a challenging goal (and, indeed, has been the focus of a rich body of work). For the *tolerant* setting, we prove a separation between testing and PAC verification: we show a hypothesis class for which the help of the prover allows the verifier to save a (roughly) quadratic factor over the number of samples that are required for closeness testing or distance approximation. See Section 3 for further details.

Proofs of Proximity for Distributions. Chiesa and Gur [6] study interactive proof systems for distribution testing. For some fixed property Π, the verifier receives samples from an unknown distribution $\mathcal{D}$, and interacts with a prover to decide whether $\mathcal{D} \in \Pi$ or whether $\mathcal{D}$ is ε-far in total variation distance from any distribution in Π. While that work does not consider

[1] I.e., the fraction of the samples that is misclassified.
[2] See preliminaries in [13, Section 1.6.2] for more about VC dimension.

machine learning, the question of verifying a lower bound ℓ on the loss of a hypothesis class can be viewed as a special case of distribution testing, where $\Pi = \{\mathcal{D} : L_{\mathcal{D}}(\mathcal{H}) \geq \ell\}$. Beyond our focus on PAC verification, an important distinction between the works is that in Chiesa and Gur's model and results, the honest prover's access to the distribution is unlimited – the honest prover can have complete information about the distribution. In this paper, we focus on doubly-efficient proofs, where the verifier and the honest prover must both be efficient in the number of data samples they require. With real-world applications in mind, this focus seems quite natural.[3]

We survey further related works in Section 1.5 in the full version of the paper [13].

1.2 Applications

The P vs. NP problem asks whether finding a solution ourselves is harder than verifying a solution supplied by someone else. It is natural to ask a similar question in learning theory: Are there machine learning problems for which learning a good hypothesis is harder than verifying one proposed by someone else? We find this theoretical motivation compelling in and of itself. Nevertheless, we now proceed to elaborate on a few more practical aspects of this question.

1.2.1 Delegation of Learning

In a commercial context, consider a scenario in which a client is interested in developing a machine learning (ML) model, and decides to outsource that task to a company P that provides ML services. For example, P promises to train a deep neural net using a big server farm. Furthermore, P claims to possess a large amount of high quality data that is not available to the client, and promises to use that data for training.

How could the client ascertain that a model provided by P is actually a good model? The client could use a general-purpose cryptographic delegation-of-computation protocol, but that would be insufficient. Indeed, a general-purpose delegation protocol can only ensure that P executed the computation as promised, but it cannot provide any guarantees about the quality of the outcome, and in particular cannot ensure that the outcome is ε-good: If P used skewed or otherwise low-quality training data (whether maliciously or inadvertently), a general-purpose delegation protocol has no way of detecting that. Moreover, even if the the data and the execution of the computation were both flawless, this still provides no guarantees on the quality of the output, because an ML model might have poor performance despite being trained as prescribed.[4,5]

A different solution could be to have P provide a proof to establish that its output is indeed ε-good. In cases where the resource gap between learning and verifying is significant enough, the client could cost-effectively verify the proof, obtaining sound guarantees on the quality of the ML model it is purchasing from P.

[3] In Chiesa and Gur's setting, it would also be sufficient for the prover to only know the distribution up to $O(\varepsilon)$ total variation distance, and this can be achieved using random samples from the distribution. However, the number of samples necessary for the prover would be linear in the domain size, which is typically exponential, and so this approach would not work for constructing doubly-efficient PAC verification protocols.

[4] E.g., a neural network might get stuck at a local minimum.

[5] Additionally, note that state-of-the-art delegation protocols are not efficient enough at present to make it practicable to delegate intensive ML computations. See the survey by Walfish and Blumberg [23] for progress and challenges in developing such systems.

1.2.2 Verification of Scientific Studies

It has been claimed that many or most published research findings are false [14]. Others refer to an ongoing *replication crisis* [20, 7], where many scientific studies are hard or impossible to replicate or reproduce [21, 3]. Addressing these issues is a scientific and societal priority.

There are many factors contributing to this problem, including: structural incentives faced by researchers, scientific journals, referees, and funding bodies; the level of statistical expertise among researchers and referees; differences in the sources of data used for studies and their replication attempts; choice of standards of statistical significance; and norms pertaining to the publication of detailed replicable experimental procedures and complete datasets of experimental results.

We stress that the current paper does not touch on the majority of these issues, and our discussion of the replication crisis (as well as our choice of quotation at the beginning of the paper) does *not* by any means suggest that adoption of PAC verification protocols will single-handedly solve all issues pertaining to replication. Rather, the contribution of the current paper with respect to scientific replication is very specific: we suggest that for some specific types of experiments, PAC verification can be used to design protocols that allow to verify the results of an experiment in a manner that uses a quantitatively smaller (or otherwise cheaper) set of independent experimental data than would be necessary for a traditional replication that fully repeats the original experiment. In [13, Appendix A] we list four such types of experiments. We argue that devising PAC verification protocols that make scientific replication procedures even modestly cheaper for specific types of experiments is a worthwhile endeavor that could help increase the amount of scientific replication or verification that occurs, and decrease the prevalence of errors that remain undiscovered in the scientific literature.

1.3 Our Setting

In this paper we consider the following form of interaction between a verifier and a prover.

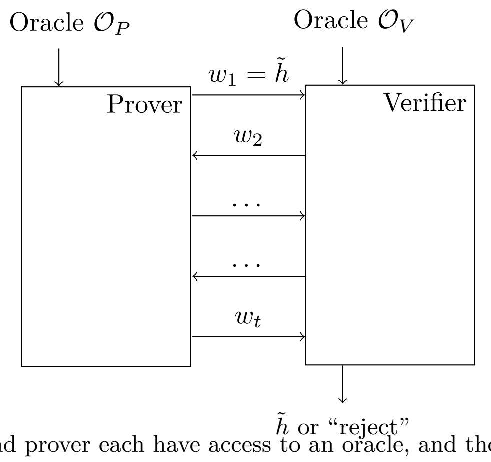


Figure 1 The verifier and prover each have access to an oracle, and they exchange messages with each other. Eventually, the verifier outputs a hypothesis, or rejects the interaction. One natural case is where the prover suggests a hypothesis $\tilde{h}$, and the verifier either accepts or rejects this suggestion.

Let $\mathcal{H} \subseteq \{0,1\}^{\mathcal{X}}$ be a class of hypotheses, and let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \{0,1\}$. The verifier and the prover each have access to an oracle, denoted $\mathcal{O}_V$ and $\mathcal{O}_P$ respectively. In the simplest case, both oracles provide i.i.d. samples from $\mathcal{D}$. That is, each time an oracle is accessed, it returns a sample from $\mathcal{D}$ taken independently of all previous samples and events. In addition, the verifier and prover each have access to a (private) random coin value, denoted ρ_V and ρ_P respectively, which are sampled from some known distributions over $\{0,1\}^*$ independently of each other and of all other events. During the interaction, the prover and verifier take turns sending each other messages $w_1, w_2, \ldots$, where $w_i \in \{0,1\}^*$ for all i. Finally, at some point during the exchange of messages, V halts and outputs either "reject" or a hypothesis $h: \mathcal{X} \to \{0,1\}$. The goal of the verifier is to output an *ε-good* hypothesis, meaning that

$$L_{\mathcal{D}}(h) \leq L_{\mathcal{D}}(\mathcal{H}) + \varepsilon.$$

A natural special case of interest is when the prover's and verifier's oracles provide sample access to $\mathcal{D}$. The prover can learn a "good" hypothesis $\tilde{h}: \mathcal{X} \to \{0,1\}$ and send it to the verifier as its first message, as in Figure 1 above. The prover and verifier then exchange further messages, wherein the prover tries to convince the verifier that $\tilde{h}$ is ε-good, and the verifier tries to asses the veracity of that claim. If the verifier is convinced, it outputs $\tilde{h}$, otherwise it rejects.

We proceed with an informal definition of PAC verification (see full definitions in Section 1.5). Before doing so, we first recall a relaxed variant of PAC learning, called *semi-agnostic* PAC learning, where we allow a multiplicative slack of $\alpha \geq 1$ in the error guarantee.

▶ **Definition** (α-PAC Learnability – informal version of [13, Definition 1.22])**.** *A class of hypothesis $\mathcal{H}$ is α-PAC learnable (or semi-agnostic PAC learnable with parameter α) if there exists an algorithm A such that for every distribution $\mathcal{D}$ and every $\varepsilon, \delta > 0$, with probability at least $1-\delta$, A outputs h that satisfies*

$$L_{\mathcal{D}}(h) \leq \alpha \cdot L_{\mathcal{D}}(\mathcal{H}) + \varepsilon. \tag{2}$$

PAC verification is the corresponding notion for interactive proof systems:

▶ **Definition** (α-PAC Verifiability – informal version of Definition 4)**.** *A class of hypothesis $\mathcal{H}$ is α-PAC verifiable if there exists a pair of algorithms (P, V) that satisfy the following conditions for every distribution $\mathcal{D}$ and every $\varepsilon, \delta > 0$:*

- ***Completeness.*** *After interacting with P, V outputs h such that with probability at least $1-\delta$, $h \neq$ reject and h satisfies (2).*
- ***Soundness.*** *After interacting with any (possibly unbounded) prover P', V outputs h such that with probability at least $1-\delta$, either $h =$ reject or h satisfies (2).*

▶ Remark 1. We insist on double efficiency; that is, that the sample complexity and running times of both V and P must be polynomial in $\frac{1}{\varepsilon}$, $\log\left(\frac{1}{\delta}\right)$, and perhaps also in some parameters that depend on $\mathcal{H}$, such as the VC dimension or Fourier sparsity of $\mathcal{H}$. ⌟

1.4 Overview of Results

In this paper, we start charting the landscape of machine learning problems with respect to Questions 1 and 2 mentioned above. First, in Section 2 we provide evidence for an affirmative answer to Questions 2. We show an interactive proof system that efficiently verifies the class of Fourier-sparse boolean functions, where the prover uses an oracle that provides query access, and the verifier uses an oracle that only provides random samples. In this proof system, both the verifier and prover send and receive messages.

The class of Fourier-sparse functions is very broad, and includes decision trees, bounded-depth boolean circuits and many other important classes of functions. Moreover, the result is interesting because it supplements the widely-held learning parity with noise (LPN) assumption, which entails that PAC learning this class from random samples alone without the help of a prover is hard [5, 24].

▶ **Lemma** (Informal version of Lemma 14). *Let $\mathcal{H}$ be the class of boolean functions $\{0,1\}^n \to \mathbb{R}$ that are t-sparse.*[6] *Then $\mathcal{H}$ is 1-PAC verifiable with respect to the uniform distribution using a verifier that has access only to random samples of the form $(x, f(x))$, and a prover that has query access to f. The verifier in this protocol is not proper; the output is not necessarily t-sparse, but it is $\text{poly}(n,t)$-sparse. The number of samples used by the verifier, the number of queries made by the prover, and their running times are all bounded by $\text{poly}\left(n, t, \log\left(\frac{1}{\delta}\right), \frac{1}{\varepsilon}\right)$.*

Proof Idea. The proof uses two standard tools, albeit in a less-standard way. The first standard tool is the Kushilevitz-Mansour algorithm [16], which can PAC learn any t-sparse function using random samples, but only if the set of non-zero Fourier coefficients is *known*. The second standard tool is the Goldreich-Levin algorithm [10] and [8, Section 2.5.2.3], which can identify the set of non-zero Fourier coefficients, but requires *query access* in order to do so. The protocol combines the two tools in a manner that overcomes the limitations of each of them. First, the verifier executes the Goldreich-Levin algorithm, but whenever it needs to query the target function, it requests that the prover perform the query and send back the result. However, the verifier cannot trust the prover, and so the verifier engineers the queries in such a way that the answers to a certain random subset of the queries are known to the verifier based on its random sample access. This allows the verifier to detect dishonest provers. When the Goldreich-Levin algorithm terminates and outputs the set of non-zero coefficients, the verifier then feeds them as input to the Kushilevitz-Mansour algorithm to find an ε-good hypothesis using its random sample access. ◀

In [13, Section 3] we formally answer Question 1 affirmatively by showing that a certain simple class of functions (generalized thresholds) exhibits a quadratic gap in sample complexity between learning and verifying:

▶ **Lemma** (Informal version of Lemma 15). *There exists a sequence of classes of functions $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, ... \subseteq \{0,1\}^{\mathbb{R}}$ such that for any fixed $\varepsilon, \delta \in (0, \frac{1}{2})$:*

(i) *The class $\mathcal{T}_d$ is proper 2-PAC verifiable, where both the verifier and prover have access to random samples, and the verifier requires only $\tilde{O}\left(\sqrt{d}\right)$ samples. Moreover, both the prover and verifier are efficient.*

(ii) *PAC learning the class $\mathcal{T}_d$ requires $\Omega(d)$ samples.*

At this point, a perceptive reader would be justified in raising the following challenges. Perhaps 2-PAC verification requires less samples than 1-PAC learning simply because of the multiplicative slack factor of 2? Alternatively, perhaps the separation follows trivially from property testing results: maybe it is possible to achieve 2-PAC verification simply by having the verifier perform closeness testing using random samples, without needing the help of the prover except for finding the candidate hypothesis? The second part of the lemma dismisses both of these concerns.

[6] A function $f: \{0,1\}^n \to \mathbb{R}$ is *t-sparse* if it has at most t non-zero Fourier coefficients, namely $|\{S \subseteq [n] : \hat{f}(S) \neq 0\}| \leq t$. See preliminaries in [13, Section 1.6.3] for further details.

▶ **Lemma** (Informal version of Lemma 15 – Continued). *Furthermore, for any fixed $\varepsilon, \delta \in (0, \frac{1}{2})$:*

(iii) *2-PAC learning the class $\mathcal{T}_d$ requires $\tilde{\Omega}(d)$ samples. This is true even if we assume that $L_{\mathcal{D}}(\mathcal{T}_d) > 0$, where $\mathcal{D}$ is the underlying distribution.*[7]

(iv) *Testing whether $L_{\mathcal{D}}(\mathcal{T}_d) \leq \alpha$ or $L_{\mathcal{D}}(\mathcal{T}_d) \geq \beta$ for any $0 < \alpha < \beta < \frac{1}{2}$ with success probability at least $1 - \delta$ when $\mathcal{D}$ is an unknown distribution (without the help of a prover) requires $\tilde{\Omega}\left(d\right)$ random samples from $\mathcal{D}$.*

Proof Idea. (ii) follows from a standard application of [13, Theorem 1.13], because $\mathsf{VC}\left(\mathcal{T}_d\right) = d$. (iii) follows by a reduction from (iv). We prove (iv) by showing a further reduction from the problem of approximating the support size of a distribution, and applying a lower bound for that problem (see [13, Theorem 3.20]).

For (i), recall from the introduction that the difficulty in designing a PAC verification proof system revolves around convincing the verifier that the term $L_{\mathcal{D}}(\mathcal{H})$ in Equation (1) is large. Therefore, we design our class $\mathcal{T}_d$ such that it admits a simple *certificate of loss*, which is a string that helps the verifier ascertain that $L_{\mathcal{D}}(\mathcal{H}) \geq \ell$ for some value ℓ.

To see how that works, first consider the simple class $\mathcal{T}_1$ of monotone increasing threshold functions $\mathbb{R} \to \{0, 1\}$, as in Figure 5a on page 17 below. Observe that if there are two events $A = [0, a) \times \{1\}$ and $B = [b, 1] \times \{0\}$ such that $a \leq b$ and $\mathcal{D}(A) = \mathcal{D}(B) = \ell$, then it must be the case that $L_{\mathcal{D}}(\mathcal{T}_1) \geq \ell$. This is true because $a \leq b$, and so if a monotone increasing threshold classifies any point in A correctly it must classify all point in B incorrectly. Furthermore, if the prover sends a description of A and B to the verifier, then the verifier can check, using a constant number of samples, that each of these events has weight approximately ℓ with high probability.

This type of certificate of loss can be generalized to the class $\mathcal{T}_d$, in which each function is a concatenation of d monotone increasing thresholds. A certificate of loss for $\mathcal{T}_d$ is simply a set of d certificates of loss $\{A_i, B_i\}_{i=1}^d$, one for each of the d thresholds. The question that arises at this point is how can the verifier verify d separate certificates while using only $\tilde{O}\left(\sqrt{d}\right)$ samples. This is performed using tools from distribution testing: the verifier checks whether the distribution of "errors" in the sets specified by the certificates is close to the prover's claims. I.e., whether the "weight" of 1-labels in each A_i and 0-labels in each B_i in the actual distribution, are close to the weights claimed by the prover. Using an identity tester for distributions this can be done using $O(\sqrt{d})$ samples (note that the identity tester need not be tolerant!). See [13, Theorem E.1] for further details. ◀

In contrast, in [13, Section 4] we show that verification is not always easier than learning:

▶ **Lemma** (Informal version of [13, Lemma 4.1]). *There exists a sequence of classes $\mathcal{H}_1, \mathcal{H}_2, \ldots$ such that:*

- *It is possible to PAC learn the class $\mathcal{H}_d$ using $\tilde{O}(d)$ samples.*
- *For any interactive proof system that proper 1-PAC verifies $\mathcal{H}_d$, in which the verifier uses an oracle providing random samples, the verifier must use at least $\Omega(d)$ samples.*

▶ Remark 2. The lower bound on the sample complexity of the verifier holds regardless of what oracle is used by the prover. ⌟

Proof Idea. We specify a set $\mathcal{X}$ of cardinality $\Omega(d^2)$, and take $\mathcal{H}_d$ to be a randomly-chosen subset of all the balanced functions $\mathcal{X} \to \{0, 1\}$ (i.e., functions f such that $|f^{-1}(0)| = |f^{-1}(1)|$). The sample complexity of PAC learning $\mathcal{H}_d$ follows from its VC dimension being

[7] In the case where $L_{\mathcal{D}}(\mathcal{T}_d) = 0$, 2-PAC learning is the same as PAC learning, so the stronger lower bound in (ii) applies.

$\tilde{O}(d)$. For the lower bound, consider proper PAC verifying $\mathcal{H}_d$ in the special case where the distribution $\mathcal{D}$ satisfies $\mathbb{P}_{(x,y)\in\mathcal{D}}[y=1]=1$, but the marginal of $\mathcal{D}$ on $\mathcal{X}$ is unknown to the verifier. Because every hypothesis in the class assigns the incorrect label 0 to precisely half of the domain, a hypothesis achieves minimal loss if it assigns the 0 labels to a subset of size $\frac{|\mathcal{X}|}{2}$ that has minimal weight. Hence, the verifier must learn enough about the distribution to identify a specific subset of size $\frac{|\mathcal{X}|}{2}$ with weight close to minimal. We show that doing so requires $\Omega\left(\sqrt{|\mathcal{X}|}\right)=\Omega(d)$ samples. ◀

Finally, in [13, Section 5] we show that in the setting of semi-supervised learning, where unlabeled samples are cheap, it is possible to perform PAC verification such that the verifier requires significantly less labeled samples than are required for learning. This verification uses a technique we call *query delegation*, and is efficient in terms of time complexity whenever there exists an efficient ERM algorithm that PAC learns the class using random samples.

1.5 Definition of PAC Verification

In Section 1.3 we informally described the setting of this paper. Here, we complete that discussion by providing a formal definition of PAC verification, which is the main object of study in this paper.

▶ **Notation 3.** *We write $[V^{\mathcal{O}_V}(x_V), P^{\mathcal{O}_P}(x_P)]$ for the random variable denoting the output of the verifier V after interacting with a prover P, when V and P receive inputs x_V and x_P respectively, and have access to oracles $\mathcal{O}_V$ and $\mathcal{O}_P$ respectively. The inputs x_V and x_P can specify parameters of the interaction, such as the accuracy and confidence parameters ε and δ. This random variable takes values in $\{0,1\}^{\mathcal{X}}\cup\{\text{reject}\}$, namely, it is either a function $\mathcal{X}\to\{0,1\}$ or it is the value "reject". The random variable depends on the (possibly randomized) responses of the oracles, and on the random coins of V and P.*

For a distribution $\mathcal{D}$, we write $V^{\mathcal{D}}$ (or $P^{\mathcal{D}}$) to denote use of an oracle that provides i.i.d. samples from the distributions $\mathcal{D}$. Likewise, for a function f, we write V^f (or P^f) to denote use of an oracle that provides query access to f. That is, in each access to the oracle, V (or P) sends some $x\in\mathcal{X}$ to the oracle, and receives the answer $f(x)$.

We also write $[V(S_V,\rho_V), P(S_P,\rho_P)]\in\{0,1\}^{\mathcal{X}}\cup\{\text{reject}\}$ to denote the deterministic output of the verifier V after interacting with P in the case where V and P receive fixed random coin values ρ_V and ρ_P respectively, and receive fixed samples S_V and S_P from their oracles $\mathcal{O}_V$ and $\mathcal{O}_P$ respectively.

We are interested in classes $\mathcal{H}$ for which an ε-good hypothesis can always be verified with high probability via this form of interaction between an efficient prover and verifier, as formalized in the following definition. Note that the following definitions include an additional multiplicative slack parameter $\alpha\geq 1$ in the error guarantee. This parameter does not exist in the standard definition of PAC learning; the standard definition corresponds to the case $\alpha=1$.

▶ **Definition 4 (α-PAC Verifiability).** *Let $\mathcal{H}\subseteq\{0,1\}^{\mathcal{X}}$ be a class of hypotheses, let $\mathfrak{D}\subseteq\Delta(\mathcal{X}\times\{0,1\})$ be some family of distributions, and let $\alpha\geq 1$. We say that $\mathcal{H}$ is $\underline{\alpha\text{-PAC verifiable with respect to } \mathfrak{D} \text{ using oracles } \mathcal{O}_V \text{ and } \mathcal{O}_P}$ if there exists a pair of algorithms (V,P) that satisfy the following conditions for every input $\varepsilon,\delta>0$:*

- ***Completeness.** For any distribution $\mathcal{D}\in\mathfrak{D}$, the random variable $h := [V^{\mathcal{O}_V}(\varepsilon,\delta), P^{\mathcal{O}_P}(\varepsilon,\delta)]$ satisfies*

$$\mathbb{P}\left[h\neq\text{reject}\ \wedge\ \left(L_{\mathcal{D}}(h)\leq\alpha\cdot L_{\mathcal{D}}(\mathcal{H})+\varepsilon\right)\right]\geq 1-\delta.$$

- ***Soundness.*** *For any distribution $\mathcal{D} \in \mathfrak{D}$ and any (possibly unbounded) prover P', the random variable $h := [V^{\mathcal{O}_V}(\varepsilon, \delta), P'^{\mathcal{O}_P}(\varepsilon, \delta)]$ satisfies*

$$\mathbb{P}\left[h \neq \text{reject} \;\wedge\; \Big(L_{\mathcal{D}}(h) > \alpha \cdot L_{\mathcal{D}}(\mathcal{H}) + \varepsilon\Big)\right] \leq \delta.$$

▶ Remark 5. Some comments about this definition:

- The behavior of the oracles $\mathcal{O}_V$ and $\mathcal{O}_P$ may depend on the specific underlying distribution $\mathcal{D} \in \mathfrak{D}$, which is unknown to the prover and verifier. For example, they may provide samples from $\mathcal{D}$.
- We insist on double efficiency; that is, that the sample complexity and running times of both V and P must be polynomial in $\frac{1}{\varepsilon}$, $\log\left(\frac{1}{\delta}\right)$, and perhaps also in some parameters that depend on $\mathcal{H}$, such as the VC dimension or Fourier sparsity of $\mathcal{H}$.
- If for every $\varepsilon, \delta > 0$, and for any (possibly unbounded) prover P', the value $h := [V^{\mathcal{O}_V}(\varepsilon, \delta), P'^{\mathcal{O}_P}(\varepsilon, \delta)]$ satisfies $h \in \mathcal{H} \cup \{\text{reject}\}$ with probability 1 (i.e., V never outputs a function that is not in $\mathcal{H}$), then we say that $\mathcal{H}$ is proper α-PAC verifiable, and that the proof system proper α-PAC verifies $\mathcal{H}$. ⌟

▶ Remark 6. An important type of learning (studied e.g. by Angluin [1] and Kushilevitz and Mansour [16]) is *learning with membership queries with respect to the uniform distribution*. In this setting, the family $\mathfrak{D}$ consists of distributions $\mathcal{D}$ such that: (1) the marginal distribution of $\mathcal{D}$ over $\mathcal{X}$ is uniform; (2) $\mathcal{D}$ has a target function $f : \mathcal{X} \to \{1, -1\}$ satisfying $\mathbb{P}_{(x,y)\sim\mathcal{D}}[y = f(x)] = 1$.[8] In Section 2, we will consider protocols for this type of learning that have the form $[V^{\mathcal{D}}, P^f]$, such that the verifier has access to an oracle providing random samples from a distribution $\mathcal{D} \in \mathfrak{D}$, and the prover has access to an oracle providing query access to f, the target function of $\mathcal{D}$. This type of protocol models a real-world scenario where P has qualitatively more powerful access to training data than V. ⌟

2 Efficient Verification for the Class of Fourier-Sparse Functions

In Section 3 below we show that in some cases verification is strictly easier than learning and closeness testing. The verification protocol presented there has a single round, where the prover simply sends a hypothesis and a proof that it is (approximately) optimal. In this section, we describe a multi-round protocol that demonstrates that interaction is helpful for verification.

The interactive protocol we present PAC verifies the class of *Fourier-sparse functions*. This is a broad class of functions, which includes decision trees, DNF formulas with small clauses, and AC^0 circuits.[9] Every function $f : \{0,1\}^n \to \mathbb{R}$ can be written as a linear combination $f = \sum_{T\subseteq[n]} \hat{f}(T)\chi_T$.[10] In Fourier-sparse functions, only a small number of coefficients are non-zero. Note that according to the learning parity with noise (LPN) assumption [5, 24] it is not possible to learn the Fourier-sparse functions efficiently using random samples only.

An important technicality is that throughout this section we focus solely on PAC verification with respect to families of distributions that have a uniform marginal over $\{0,1\}^n$, and have a target function $f : \{0,1\}^n \to \{1,-1\}$ such that $\mathbb{P}_{(x,y)\sim\mathcal{D}}[y = f(x)] = 1$. See further discussion in Remark 6 on page 11. In this setting, in order to learn f it is sufficient to approximate its heavy Fourier coefficients.

[8] Note that f is not necessarily a member of $\mathcal{H}$, so this is still an *agnostic* (rather than *realizable*) case.
[9] See Mansour [18, Section 5.2.2, Theorems 5.15 and 5.16]. (AC^0 is the set of functions computable by constant-depth boolean circuits with a polynomial number of AND, OR and NOT gates.)
[10] The real numbers $\hat{f}(T)$ are called *Fourier coefficients*, and the functions χ_T are called *characters*.

▶ **Notation 7.** *Let $f : \{0,1\}^n \to \mathbb{R}$, and let $\tau \geq 0$. The set of τ-heavy coefficients of f is $\hat{f}^{\geq\tau} = \{T \subseteq [n] : \ |\hat{f}(T)| \geq \tau\}$.*

Furthermore, approximating a single coefficient is easy given random samples from the uniform distribution [13, Claim 2.11]. There are, however, an exponential number of coefficients, so approximating all of them is not feasible. This is where verification comes in. If the set of heavy coefficients is known, and if the function is Fourier-sparse, then one can efficiently learn the function by approximating that particular set of coefficients. The prover can provide the list of heavy coefficients, and then the verifier can learn the function by approximating these coefficients.

The challenge that remains in designing such a verification protocol is to verify that the provided list of heavy coefficients is correct. If the list contains some characters that are not actually heavy, no harm is done.[11] However, if a dishonest prover omits some of the heavy coefficients from the list, how can the verifier detect this omission? The following result provides an answer to this question.

▶ **Lemma 8 (Interactive Goldreich-Levin).** *There exists an interactive proof system (V, P^*) as follows. For every $n \in \mathbb{N}$, $\delta > 0$, every $\tau \geq 2^{-\frac{n}{10}}$, every function $f : \{0,1\}^n \to \{0,1\}$, and every prover P, let*

$$L_P = [V(S, n, \tau, \delta, \rho_V), P^f(n, \tau, \delta, \rho_P)]$$

be a random variable denoting the output of V after interacting with the prover P, which has query access to f, where $S = \big((x_1, f(x_1)), \ldots, (x_m, f(x_m))\big)$ is a random sample with $x_1, \ldots, x_m$ taken independently and uniformly from $\{0,1\}^n$, and ρ_V, ρ_P are strings of private random coins. L_P takes values that are either a collection of subsets of $[n]$, or "reject".

The following properties hold:

- ***Completeness.*** $\mathbb{P}\left[L_{P^*} \neq \text{reject} \ \wedge \ \hat{f}^{\geq\tau} \subseteq L_{P^*}\right] \geq 1 - \delta$.
- ***Soundness.*** *For any (possibly unbounded) prover P,*

$$\mathbb{P}\left[L_P \neq \text{reject} \wedge \hat{f}^{\geq\tau} \nsubseteq L_P\right] \leq \delta.$$

- ***Double efficiency.*** *The verifier V uses at most $O\left(\frac{n}{\tau}\log\left(\frac{n}{\tau}\right)\log\left(\frac{1}{\delta}\right)\right)$ random samples from f and runs in time $\text{poly}\left(n, \frac{1}{\tau}, \log\left(\frac{1}{\delta}\right)\right)$. The runtime of the prover P^*, and the number of queries it makes to f, are at most $O\left(\frac{n^3}{\tau^5}\log\left(\frac{1}{\delta}\right)\right)$. Whenever $L_P \neq \text{reject}$, the cardinality of L_P is at most $O\left(\frac{n^2}{\tau^5}\log\left(\frac{1}{\delta}\right)\right)$.*

All proofs for this section appear in Section 2 in the full version of the paper [13].

▶ Remark 9. In Section 1.5 we defined interactive proof systems specifically for PAC verification. The proof system in Lemma 8 is technically different. The verifier outputs a collection of sets instead of a function, and it satisfies different completeness and soundness conditions. ⌟

The verifier V operates by simulating the Goldreich-Levin (GL) algorithm for finding $\hat{f}^{\geq\tau}$. However, the GL algorithm requires query access to f, while V has access only to random samples. To overcome this limitation, V delegates the task of querying f to the

[11] The verifier can approximate each coefficient in the list and discard of those that are not heavy. Alternatively, the verifier can include the additional coefficients in its approximation of the target function, because the approximation improves as the number of estimated coefficients grows (so long as the list is polynomial in n).

prover P, who does have the necessary query access. Because P is not trusted, V engineers the set of queries it delegates to P in such a way that some random subset of them already appear in the sample S which V has received as input. This allows V to independently verify a random subset of the results sent by P, ensuring that a sufficiently dishonest prover is discovered with high probability.

The Interactive Goldreich-Levin Protocol. The verifier for Lemma 8 uses Protocol 2 (IGL), which repeatedly applies Protocol 3 (IGL-ITERATION).

Protocol 2 Interactive Goldreich-Levin: IGL(n, τ, δ).

```
$V$ performs the following:
  $r \leftarrow \left\lceil \left(\frac{4n}{\tau}+1\right)\log\left(\frac{1}{\delta}\right)\right\rceil$
  **for** $i \in [r]$ **do**
      $L_i \leftarrow$ IGL-ITERATION$(n,\tau)$
      **if** $L_i =$ reject **then**
          **output** reject
  $L \leftarrow \bigcup_{i\in[r]} L_i$
  **output** $L$
```

Lemma 8 follows from two claims: Claim 10 says that if the prover is mostly honest, then the output is correct. Claim 11 says that if the prover is too dishonest, it will be rejected.

▷ Claim 10 (Completeness of IGL). Consider an execution of IGL-ITERATION(n, τ) for $\tau \geq 2^{-\frac{n}{10}}$. For any prover P and any randomness ρ_P, if V did not reject, and the evaluations provided by P were mostly honest, in the sense that $\forall i \in [n] : \mathbb{P}_{x\in H}\left[\tilde{f}(x \oplus e_i) \neq f(x \oplus e_i)\right] \leq \frac{\tau}{4}$, then $\mathbb{P}\left[\hat{f}^{\geq\tau} \subseteq L\right] \geq \frac{1}{2}$, where the probability is over the sample S and the randomness ρ_V. ⌟

Proof Idea. We show that for every heavy coefficient $T \in \hat{f}^{\geq\tau}$, $\mathbb{P}[T \notin L] \leq \frac{\tau^2}{4}$. From a union bound, this is sufficient to prove the claim, because Parseval's identity implies that $|\hat{f}^{\geq\tau}| \leq \frac{1}{\tau^2}$. The main observation is that the Goldreich-Levin algorithm is resilient to some *adversarial* noise. To see this, note that if $T \in \hat{f}^{\geq\tau}$, then $\mathbb{P}_{x\in\{0,1\}^n}[f(x) = \ell(x)] \geq \frac{1}{2} + \frac{\tau}{2}$ for the linear function $\ell(x) = \oplus_{i\in T} x_i$ (or its negation $\neg\ell$). Therefore, f agrees with ℓ with probability at least $\frac{1}{2} + \frac{\tau}{4}$, even if a $\frac{\tau}{4}$-fraction of f's values are adversarially corrupted. Hence, in the iteration of the outer loop in Step 4 in which $y_j = \ell(b_j)$ for all $j \in [k]$, the majority function will output the correct value of $f\left(x^K \oplus e_i\right) \oplus y^K = \ell(x^K \oplus e_i) \oplus \ell(x^K) = \ell(e_i)$, and this results in T being added to the output set L. Some finer details are discussed in the full version of the paper. ◀

▷ Claim 11 (Soundness of IGL). Consider an execution of IGL-ITERATION(n, τ). For any prover P and any randomness value ρ_P, if there exists $i \in [n]$ for which P was too dishonest in the sense that $\mathbb{P}_{x\in H}\left[\tilde{f}(x \oplus e_i) \neq f(x \oplus e_i)\right] > \frac{\tau}{4}$, then $\mathbb{P}[L = \text{reject}] \geq \frac{\tau}{4n}$, where the probability is over the sample S and the randomness ρ_V. ⌟

[12] For any j, e_j is a vector in which the j-th entry is 1 and all other entries are 0.

Protocol 3 Interactive Goldreich-Levin Iteration: IGL-Iteration(n, τ).

Assumption: V receives a sample $S = \Big((x_1, f(x_1)), \ldots, (x_m, f(x_m))\Big)$ such that $m = \left\lceil \log\left(\frac{40n}{\tau^4} + 1\right)\right\rceil$, for all $i \in [m]$, $x_i \in \{0,1\}^n$ is chosen independently and uniformly, and $f(x_i) \in \{0,1\}$.

---

1. V selects $i^* \in [n]$ uniformly at random, and then sends B to P, where $B = \{b_1, \ldots, b_k\} \subseteq \{0,1\}^n$ is a basis chosen uniformly at random from the set of bases of the subspace $H = \operatorname{span}(\{x_1 \oplus e_{i^*}, \ldots, x_m \oplus e_{i^*}\})$.[12]
2. P sends V the set $\{(x \oplus e_i, \tilde{f}(x \oplus e_i)) : \ i \in [n] \ \wedge \ x \in H\}$, where for any z, $\tilde{f}(z)$ is purportedly the value of $f(z)$ obtained using P's query access to f.
3. V checks that for all $i \in [m]$, the evaluation $f(x_i)$ provided by P equals that which appeared in the sample S. If there are any discrepancies, V rejects the interaction and terminates. Otherwise:
4. Let $\mathcal{K} = \{K : \ \varnothing \subsetneq K \subseteq [k]\}$. V Performs the following computation and outputs L:

```
$L \leftarrow \varnothing$
**for** $(y_1, \ldots, y_k) \in \{0,1\}^k$ **do**
    **for** $K \in \mathcal{K}$ **do**
        $x^K \leftarrow \bigoplus_{i \in K} b_i$
        $y^K \leftarrow \bigoplus_{i \in K} y_i$
    **for** $i \in [n]$ **do**
        $a_i \leftarrow \text{majority}_{K \in \mathcal{K}} \left(\tilde{f}\left(x^K \oplus e_i\right) \oplus y^K\right)$
    add $\{i : \ a_i = 1\}$ and $\{i : \ a_i = 0\}$ to $L$
**output** $L$
```

Proof Idea. In Protocol 3, the verifier "hides" the sample it knows in the random subspace $H \oplus e_{i^*}$ for an index $i^* \in [n]$ chosen uniformly at random. With probability at least $\frac{1}{n}$, i^* is an index for which the prover was too dishonest. If that is the case, then in expectation a $\frac{\tau}{4}$-fraction of the prover's answers on the known sample were dishonest, because the known sample is a random subset of $H \oplus e_{i^*}$. The claim now follows from Markov's inequality. ◀

▶ Remark 12. It is possible to run all repetitions of the IGL protocol in parallel such that only 2 messages are exchanged. ⌟

Efficient Verification of Fourier-Sparse Functions. As a corollary of Lemma 8, we obtain the following lemma, which is an interactive version of the Kushilevitz-Mansour algorithm [16, 17, 18]. It says that the class of t-sparse boolean functions is efficiently PAC verifiable with respect to the uniform distribution using an interactive proof system (Protocol 4) of the form $[V^{\mathcal{D}}, P^f]$, where the prover has query access and the verifier has random samples .

Protocol 4 uses a procedure EstimateCoefficient [13, Algorithm 4] that estimates a single fourier coefficient $\hat{f}(T)$ up to precision λ with confidence δ using $\left\lceil \frac{2\ln(2/\delta)}{\lambda^2} \right\rceil$ random samples. Note that the output of Protocol 4 is a function $h : \{0,1\}^n \to \mathbb{R}$, not necessarily a boolean function.

▶ **Notation 13.** *Let $\mathcal{X}$ be a finite set. We write $\mathfrak{D}_{\mathcal{U}}^{\mathrm{func}}(\mathcal{X})$ to denote the set of all distributions $\mathcal{D}$ over $\mathcal{X} \times \{1,-1\}$ that have the following two properties:*

- *The marginal distribution of $\mathcal{D}$ over $\mathcal{X}$ is uniform. Namely, $\sum_{y\in\{1,-1\}} \mathcal{D}\Big((x,y)\Big) = \frac{1}{|\mathcal{X}|}$ for all $x \in \mathcal{X}$.*
- *$\mathcal{D}$ has a target function $f : \mathcal{X} \to \{1,-1\}$ satisfying $\mathbb{P}_{(x,y)\sim\mathcal{D}}[y = f(x)] = 1$.*

▶ **Lemma 14.** *Let $\mathcal{X} = \{0,1\}^n$, and let $\mathcal{H}$ be the class of functions $\mathcal{X} \to \mathbb{R}$ that are t-sparse.*[6] *The class $\mathcal{H}$ is 1-PAC verifiable for any $\varepsilon \geq 4t \cdot 2^{-\frac{n}{10}}$ with respect to $\mathfrak{D}_{\mathcal{U}}^{\mathrm{func}}(\mathcal{X})$ by a proof system in which the verifier has access to random samples from a distribution $\mathcal{D} \in \mathfrak{D}_{\mathcal{U}}^{\mathrm{func}}(\mathcal{X})$, and the honest prover has oracle access to the target function $f : \mathcal{X} \to \{1,-1\}$ of $\mathcal{D}$. The running time of both parties is at most $\mathrm{poly}\left(n, t, \frac{1}{\varepsilon}, \log\left(\frac{1}{\delta}\right)\right)$. The verifier in this protocol is not proper; the output is not necessarily t-sparse, but it is $\mathrm{poly}\left(n, t, \frac{1}{\varepsilon}, \log\left(\frac{1}{\delta}\right)\right)$-sparse.*

Protocol 4 PAC Verification of t-Sparse Functions: VerifyFourierSparse(n,t,ε,δ).

```
V performs the following:
  $\tau \leftarrow \frac{\varepsilon}{4t}$
  $L \leftarrow \mathrm{IGL}(n, \tau, \frac{\delta}{2})$
  if $L =$ reject then
      output reject
  else
      $\lambda \leftarrow \sqrt{\frac{\varepsilon}{8|L|}}$
      for $T \in L$ do
          $\alpha_T \leftarrow \textsc{EstimateCoefficient}(T, \lambda, \frac{\delta}{2|L|})$
      $h \leftarrow \sum_{T\in L} \alpha_T \chi_T$
      output $h$
```

Complete proofs appear in Section 2 in the full version of the paper [13].

3 Separation Between Learning, Testing, and PAC Verification

In this section we present the following gap in sample complexity between *learning* and *verification*. Conceptually, the result tells us that at least in some scenarios, delegating a learning task to an untrusted party is worthwhile, because verifying that their final result is correct is significantly cheaper than finding that result ourselves.

▶ **Lemma 15** (Separation Between Learning, Testing, and Verification). *There exists a sequence of classes of functions $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots \subseteq \{0,1\}^{\mathbb{R}}$ such that for any fixed $\varepsilon, \delta \in (0, \frac{1}{2})$ all of the following hold:*

(i) *$\mathcal{T}_d$ is proper 2-PAC verifiable, where the verifier uses*[13]

$$m_V = O\left(\frac{\sqrt{d}\log(d)\log\left(\frac{1}{\delta}\right)}{\varepsilon^6}\right)$$

random samples, the honest prover uses

[13] We believe that the dependence of m_V on ε can be improved, see [13, Remark 3.15].

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

random samples, and each of them runs in time polynomial in its number of samples.[14]

(ii) *Agnostic PAC learning $\mathcal{T}_d$ requires $\Omega\left(\frac{d+\log(\frac{1}{\delta})}{\varepsilon^2}\right)$ samples.*

(iii) *If $\varepsilon \leq \frac{1}{32}$ then 2-PAC learning the class $\mathcal{T}_d$ requires $\Omega\left(\frac{d}{\log(d)}\right)$ samples. This is true even if we assume that $L_{\mathcal{D}}(\mathcal{T}_d) > 0$, where $\mathcal{D}$ is the underlying distribution.*

(iv) *Testing whether $L_{\mathcal{D}}(\mathcal{T}_d) \leq \alpha$ or $L_{\mathcal{D}}(\mathcal{T}_d) \geq \beta$ for any $0 < \alpha < \beta < \frac{1}{2}$ with success probability at least $1 - \delta$ when $\mathcal{D}$ is an unknown distribution (without the help of a prover) requires $\Omega\left(\frac{d}{\log(d)}\right)$ random samples from $\mathcal{D}$.*

Our exposition in this section is partial, and aims only to convey the main ideas of part (i). Complete formal proofs appear in [13, Section 3]. We show an MA-like proof system wherein the prover sends a single message $(\tilde{h}, \tilde{C}, \tilde{\ell})$ such that allegedly $\tilde{h}$ is an ε-good hypothesis with loss at most $\tilde{\ell} > 0$. $\tilde{C} \in \{0,1\}^*$ is a string called a *certificate of loss*, which helps the verifier ascertain that $\mathcal{H}$ has a large loss with respect to the unknown distribution $\mathcal{D}$. The verifier operates as follows:[15]

- Verifies that $L_{\mathcal{D}}(\tilde{h}) \leq \tilde{\ell}$ with high probability. That is, it estimates the loss of $\tilde{h}$ with respect to $\mathcal{D}$, and checks that with high probability it is at most $\tilde{\ell}$.
- Uses the certificate of loss $\tilde{C}$ to verify that with high probability, $L_{\mathcal{D}}(\mathcal{H}) \geq \tilde{\ell} - \varepsilon$. This step is called *verifying the certificate.*

The class $\mathcal{T}_d$ of *multi-thresholds* that satisfies Lemma 15 is illustrated in Figure 5, and is defined as follows.

▶ **Definition 16.** *For any $d \in \mathbb{N}$, denote by $\mathcal{T}_d$ the class of functions $\mathcal{T}_d = \{f_{t_1,\ldots,t_d} : t_1, \ldots, t_d \in \mathbb{R}\}$ where for all $t_1, \ldots, t_d \in \mathbb{R}$ and $x \in [0, d]$, the function $f_{t_1,\ldots,t_d} : \mathbb{R} \to \{0,1\}$ is given by*

$$f_{t_1,\ldots,t_d}(x) = \begin{cases} 0 & x < t_{\lceil x \rceil} \\ 1 & x \geq t_{\lceil x \rceil}, \end{cases}$$

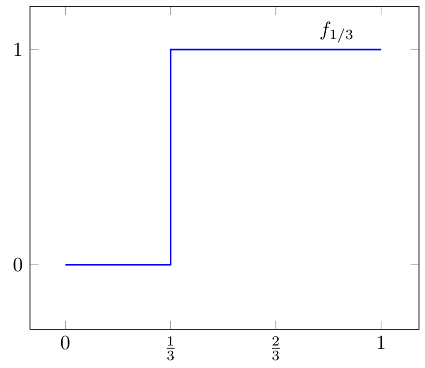
and $f_{t_1,\ldots,t_d}$ vanishes on the complement of $[0, d]$.

▶ Remark 17. For convenience, we present the separation result with respect to functions defined over $\mathbb{R}$, we assume that the marginal distribution of the samples on $\mathbb{R}$ is absolutely continuous with respect to the Lebesgue measure, and we ignore issues relating to the representation of real numbers in computations and protocol messages. This provides for a smoother exposition of the ideas. In [13, Appendix C], we show how the results can be discretized. ⌟

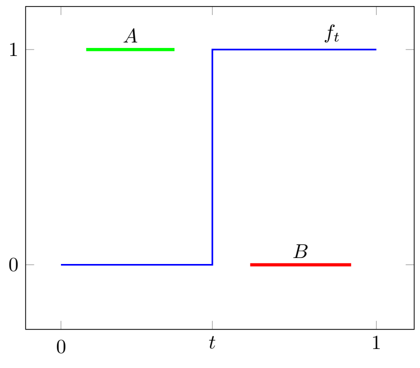
We first present the certificate structure for the class of threshold functions, namely $\mathcal{T}_d$ with $d = 1$. Certificates of loss for $\mathcal{T}_1$ are easy to visualize, and they induce a proof system for PAC verifying $\mathcal{T}_1$ that is complete, sounds, and doubly efficient. However, verifying certificates for $\mathcal{T}_1$ requires as much resources as PAC learning $\mathcal{T}_1$ without the help of a prover. The next step is to show that these certificates generalize to the class $\mathcal{T}_d$ of multi-thresholds, and that for $\mathcal{T}_d$ there indeed is a gap in sample complexity between verifying and learning.

[14] Subsequent unpublished work by JS and Saachi Mutreja suggests that it is possible to strengthen this to obtain 1-PAC verification with better sample complexity bounds.

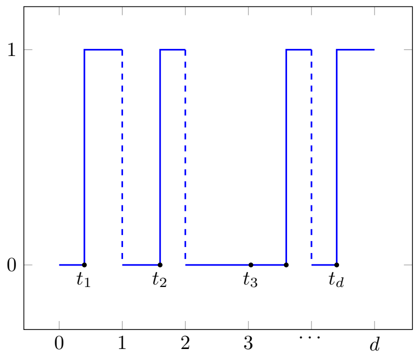
[15] We provide a more detailed description of the verification procedure below, and in [13, Claim 3.14].



(a) The function $f_{1/3} \in \mathcal{T}_1$. $\mathcal{T}_1$ consists of monotone increasing threshold functions $[0,1] \to \{0,1\}$.



(b) Structure of a simple certificate of loss for $\mathcal{T}_1$. The set A is labeled with 1, and B is labeled 0. The depicted threshold f_t happens to misclassify both A and B, but it is just one possible threshold.



(c) Example of a function in $\mathcal{T}_d$.

Figure 5 The class $\mathcal{T}_d$ of multi-thresholds, with the special case $\mathcal{T}_1$ and its certificate structure.

Certificates of loss for $\mathcal{T}_1$. Consider two sets $A \subseteq [0,1] \times \{1\}$ and $B \subseteq [0,1] \times \{0\}$, such that all the points in A are located to the left of all the points in B, as in Figure 5b. Because we only allow thresholds that are monotone increasing, a threshold that labels any point in A correctly must label all points of B incorrectly, and vice versa. Hence, any threshold must have loss at least $\min\{\mathcal{D}(A), \mathcal{D}(B)\}$. Estimating $\mathcal{D}(A)$ and $\mathcal{D}(B)$ is easy (by Hoeffding's inequality). Formally:

▶ **Definition 18.** *Let $\mathcal{D} \in \Delta([0,1] \times \{0,1\})$ be a distribution and $\ell, \eta \geq 0$. A certificate of loss at least ℓ for class $\mathcal{T}_1$ is a pair (a,b) where $0 < a \leq b < 1$.*

We say that the certificate is η-valid with respect to distribution $\mathcal{D}$ if the events $A = [0,a) \times \{1\}$ and $B = [b,1] \times \{0\}$ satisfy $|\mathcal{D}(A) - \ell| + |\mathcal{D}(B) - \ell| \leq \eta$.

▷ Claim 19 ([13, Claims 3.5 and 3.4]). Let $\mathcal{D} \in \Delta([0,1] \times \{0,1\})$ be a distribution and $\ell, \eta \geq 0$.

- Soundness. If $\mathcal{D}$ has a certificate of loss at least ℓ which is η-valid with respect to $\mathcal{D}$, then $L_{\mathcal{D}}(\mathcal{T}_1) \geq \ell - \eta$.
- Completeness. If $L_{\mathcal{D}}(\mathcal{T}_1) = \ell$ then there exists a 0-valid certificate of loss at least $\frac{\ell}{2}$ with respect to $\mathcal{D}$. ⌟

Certificates of loss for $\mathcal{T}_d$ with $d > 1$. A certificate of loss is simply a collection of d certificates of loss for $\mathcal{T}_1$, one for each unit interval in $[0, d]$. Standard techniques from VC theory show that it is possible to efficiently generate certificates for $\mathcal{T}_d$ for a particular distribution using $\tilde{O}(d^2)$ samples [13, Claim 3.13]. This is more expensive than learning the class $\mathcal{T}_d$, but it may be worthwhile seeing as the verifier can apply techniques from distribution testing to verify purported certificates using only $\tilde{O}(\sqrt{d})$ samples – which is cheaper than learning [13, Claim 3.14].

Further discussion and complete proofs of Lemma 15(ii)-(iv), including the lower bound for closeness testing, appear in the full version of the paper [13].

4 Directions for Future Work

This work initializes the study of verification in the context of machine learning. We have seen separations between the sample complexity of verification versus learning and testing, a protocol that uses interaction to efficiently learn sparse boolean functions, and have seen that in some cases the sample complexities of verification and learning are the same.

Building a theory that can help guide verification procedures is a main objective for future research. A specific approach is to identify dimension-like quantities that describe the sample complexity of verification, similarly to role VC dimension plays in characterizing learnability. A different approach is to understand the trade-offs between the various resources in the system – the amount of time, space and samples used by the prover and the verifier, as well as the amount of interaction between the parties.

From a practical perspective, we described potential applications for delegation of machine learning, and for verification of experimental data. It seems beneficial to build efficient verification protocols for machine learning problems that are commonly used in practice, and for the types of scientific experiments mentioned in [13, Appendix A]. This would have commercial and scientific applications.

There are also some technical improvements that we find interesting. For example, is there a simple way to improve the MA-like protocol for the multi-thresholds class $\mathcal{T}_d$ to achieve 1-PAC verification (instead of 2-PAC verification)?

Finally, seeing as learning verification is still a new concept, it would be good to consider alternative formal definitions, investigate how robust our definition is, and discuss what the "right" definition should be.

Additional directions are discussed in Section 6 in the full version of the paper [13]. ◀

References

1 Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987. doi:10.1016/0890-5401(87)90052-6.

2 Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 21–30. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.64.

3 C. Glenn Begley and Lee M. Ellis. Raise standards for preclinical cancer research. *Nature*, 483(7391):531–533, 2012.

4 Avrim Blum and Lunjia Hu. Active tolerant testing. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, volume 75 of *Proceedings of Machine Learning Research*, pages 474–497. PMLR, 2018. URL: http://proceedings.mlr.press/v75/blum18a.html.

5 Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

6 Alessandro Chiesa and Tom Gur. Proofs of proximity for distribution testing. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 53:1–53:14, 2018. doi:10.4230/LIPIcs.ITCS.2018.53.

7 Fiona Fidler and John Wilcox. Reproducibility of scientific results. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2018 edition, 2018.

8 Oded Goldreich. *Foundations of cryptography: volume 1, basic tools*. Cambridge university press, 2007.

9 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

10 Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

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

12 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

13 Shafi Goldwasser, Guy N. Rothblum, Jonathan Shafer, and Amir Yehudayoff. Interactive proofs for verifying machine learning. *Electron. Colloquium Comput. Complex.*, 27:58, 2020. URL: https://eccc.weizmann.ac.il/report/2020/058.

14 John PA Ioannidis. Why most published research findings are false. *PLoS medicine*, 2(8):e124, 2005.

15 Michael J. Kearns and Dana Ron. Testing problems with sublearning sample complexity. *J. Comput. Syst. Sci.*, 61(3):428–456, 2000. doi:10.1006/jcss.1999.1656.

16 Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

17 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993.

18 Yishay Mansour. Learning boolean functions via the Fourier transform. In *Theoretical advances in neural computation and learning*, pages 391–424. Springer, 1994.

19 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006. doi:10.1016/j.jcss.2006.03.002.

20 Harold Pashler and Eric-Jan Wagenmakers. Editors' introduction to the special section on replicability in psychological science: A crisis of confidence? *Perspectives on Psychological Science*, 7(6):528–530, 2012.

21 Florian Prinz, Thomas Schlange, and Khusru Asadullah. Believe it or not: how much can we rely on published data on potential drug targets? *Nature reviews Drug discovery*, 10(9):712–712, 2011.

22 Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

23 Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Commun. ACM*, 58(2):74–84, 2015. doi:10.1145/2641562.

24 Yu Yu and John Steinberger. Pseudorandom functions in almost constant depth from low-noise LPN. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 154–183. Springer, 2016.