Editors' Suggestion

Superpolynomial quantum-classical separation for density modeling

Niklas Pirnay,[1] Ryan Sweke,[2,*] Jens Eisert,[2,3] and Jean-Pierre Seifert[1,4]
[1]*Electrical Engineering and Computer Science, Technische Universität Berlin, 10587 Berlin, Germany*
[2]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*
[3]*Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany*
[4]*Fraunhofer SIT, D-64295 Darmstadt, Germany*

(Received 7 December 2022; accepted 20 March 2023; published 13 April 2023)

Density modeling is the task of learning an unknown probability density function from samples, and is one of the central problems of unsupervised machine learning. In this work, we show that there exists a density modeling problem for which fault-tolerant quantum computers can offer a superpolynomial advantage over classical learning algorithms, given standard cryptographic assumptions. Along the way, we provide a variety of additional results and insights of potential interest for proving future distribution learning separations between quantum and classical learning algorithms. Specifically, we (a) provide an overview of the relationships between hardness results in supervised learning and distribution learning, and (b) show that any *weak* pseudorandom function can be used to construct a classically hard density modeling problem. The latter result opens up the possibility of proving quantum-classical separations for density modeling based on weaker assumptions than those necessary for pseudorandom functions.

DOI: 10.1103/PhysRevA.107.042416

I. INTRODUCTION

The task of learning a representation of a probability distribution from samples is of importance in a wide variety of contexts, from the natural sciences to industry. As such, a central focus of modern machine learning is to develop algorithms and models for this task. Of particular importance is the distinction between *density modeling* and *generative modeling*. In density modeling the task is to learn an *evaluator* for a distribution, i.e., a function which on input of a sample returns the probability weight assigned to that sample by the underlying distribution. As such, density modeling is sometimes referred to, as we do here, as *evaluator learning*. In generative modeling, the task is to learn a *generator* for a distribution, i.e., a function which, given a (uniformly) random input seed, outputs a sample with probabilities according to the target distribution. As a result, generative modeling is sometimes referred to as *generator learning*. As an example, in a generative modeling problem the goal might be to generate images of cats or dogs, while the associated density modeling problem would be to evaluate the probability that an image depicts a cat or a dog. It is important to stress that these two learning tasks are indeed fundamentally different. That is to say, efficiently learning a generator for a distribution does not imply efficiently learning an evaluator and vice versa [1].

Given the incredible success of modern machine learning, and the rapidly increasing availability of *quantum* computational devices, a natural question is whether or not quantum devices can provide any advantage in this domain [2–5]. Most current research in this direction is of a heuristic nature [6,7]. However, there also exists an emerging body of results which provide examples of machine learning tasks for which one can prove rigorously a meaningful separation between the power of classical and quantum computational devices [8–11], even though results on such rigorous separations are still rather scarce [12]. One of these, the work of Ref. [11], has shown rigorously that one can obtain a quantum advantage in *generative modeling* by constructing a distribution class which is (a) provably hard to generator learn classically but (b) efficiently generator learnable using a fault-tolerant quantum computer. Importantly, however, Ref. [11] has not addressed the related task of *density modeling*.

In this work, we close this gap by showing that the class of probability distributions constructed in Ref. [11] in fact also allows one to demonstrate a superpolynomial quantum-classical separation for density modeling. Additionally, along the way we provide a variety of insights and additional results, which may be of independent interest for constructing future quantum-classical separations in distribution learning. More specifically, in this work we do the following:

(1) Any quantum-classical separation requires a proof of classical hardness. The generative modeling separation of Ref. [11] relies crucially on a result from Ref. [1] which shows that from any *pseudorandom function* (PRF) one can construct a distribution class which is provably hard to generator learn classically. We strengthen this fundamental tool by showing that for the case of density modeling, any *weak PRF* can be used to construct a distribution class which is provably hard to evaluator learn classically. This opens up the door for proving classical hardness results for density modeling, based on weaker assumptions than those necessary for candidate PRF constructions. In particular, the hope is that one may be able to prove classical hardness using assumptions which

[*]Currently at IBM Quantum, Almaden Research Center, San Jose, CA 95120, USA.

do not immediately also rule out the possibility of efficient learning algorithms running on *near-term quantum devices*.

(2) We prove a superpolynomial quantum-classical separation for density modeling using the distribution class from Ref. [11]. As the distribution class from Ref. [11] is constructed from a PRF, the classical hardness follows immediately given the above-mentioned insight that even weak PRFs are sufficient for density modeling hardness. As such, what remains is to provide an efficient quantum evaluator learner for this distribution class, and we show that a simple modification of the quantum generator learner from Ref. [11] is sufficient to achieve this.

(3) The majority of work in computational learning theory has been focused on the task of *supervised learning* Boolean functions. As such, it is natural to ask to what extent hardness results and quantum-classical separations in supervised learning can be leveraged to obtain separations for distribution learning. We provide an overview of the extent to which this is or is not possible, for both generative and density modeling. Once again, the hope is that this provides a toolbox for proving future quantum-classical separations in distribution learning.

This work is structured as follows: We start in Sec. II by providing some essential definitions and background from both computational learning theory and cryptography. We note that as this work to a large extent generalizes and extends Ref. [11], we do not provide all necessary background here, and we refer often to Ref. [11] for a variety of definitions and constructions. Given the necessary background, we proceed in Sec. III to present a variety of techniques for proving hardness results in distribution learning from hardness results in supervised learning. Of particular interest is Sec. III B, in which we show that one can use any *weak* PRF to construct a distribution class which is classically hard to evaluator learn. Using these tools, we then show in Sec. IV a superpolynomial quantum-classical separation for density modeling, using the distribution class from Ref. [11]. Finally, we conclude in Sec. V with a discussion and outlook.

II. BACKGROUND

To show a quantum-classical learning separation, on the highest level, one needs to prove two things: classical learning hardness and efficiency of quantum learning. To introduce the necessary formalism, we will start in this section by providing an overview of the probably approximately correct (PAC) framework for learning both functions and distributions. Given this, we will then present a construction from Ref. [1] which allows one to define distribution classes from function classes in a way which facilitates the conversion of function learning hardness to distribution learning hardness. Finally, we introduce weak-secure pseudorandom functions, which will later be used to construct distribution classes, via the aforementioned construction from Ref. [1], for which the density modeling problem is provably hard for classical learning algorithms. In what follows, we denote:

(a) $\mathbb{1}(x, y)$: the index function evaluating to 1 if and only if $x = y$,

(b) $x \sim U(\mathcal{X})$: sample x from the uniform distribution over a set $\mathcal{X}$ (sometimes $\mathcal{X}$ is omitted if $\mathcal{X}$ is clear from the context),

(c) $\mathrm{poly}(a, b)$: any polynomial in a and b, or sometimes also meaning the set of all polynomials in a, b,

(d) $\{0, 1\}^n$: the set of bit strings of length n,

(e) $a\|b$: the concatenation of bit strings a, b,

(f) 1^n: the bit string consisting of n 1's,

(g) $D(x)$: the probability mass assigned to x, if D is a probability distribution,

(h) $d_{TV}(P, Q)$: the total variation distance between distributions P and Q, and

(i) $\mathbb{Z}_q$: the residue class ring $\mathbb{Z}/q\mathbb{Z}$.

Before introducing the formalism for analyzing distribution learning, we introduce Valiant's PAC learning framework for function learning [13], since its proposal is the standard framework for rigorously analyzing the complexity of supervised learning problems [14]. In it we are concerned with learning some class of functions that map n bits to m bits. At a high level, for any such target function in the class, when given some sort of oracle access to the unknown target function, a learning algorithm should with high probability output a hypothesis function that is close to the target function.

For this function learning task, we distinguish between two different types of oracle access to the function f that is to be learned.[1] Firstly, the membership query oracle to f, $\mathrm{MQ}(f)$, which when queried with x yields the tuple $(x, f(x))$. We denote this via

$$\mathrm{query}[\mathrm{MQ}(f)](x) = (x, f(x)). \tag{1}$$

The membership query access corresponds to the ability to evaluate f on chosen points. We sometimes refer to the membership query oracle in general without any fixed function simply as MQ. Secondly, the η-noisy random example oracle $\mathrm{REX}(f, P, \eta)$ to f is defined via

$$\mathrm{query}[\mathrm{REX}(f, P, \eta)] = \begin{cases} (x, f(x)) \text{ with probability } P(x)(1-\eta) \\ (x, \neg f(x)) \text{ with probability } P(x)(\eta) \end{cases}, \tag{2}$$

where P is a probability distribution over inputs, $\eta \in [0, 1]$ is a noise rate, and $\neg f(x)$ is any element in the image space of f except $f(x)$. If $\eta = 0$, we also write $\mathrm{REX}(f, P)$. At a high level, this oracle generates random (possibly noisy) input-output tuples from f. If we refer to the random example oracle in general, without any fixed function, we write $\mathrm{REX}(P, \eta)$. Algorithmically, we consider a query to MQ or REX to take unit time.

We can now give the definition of a PAC learning algorithm for function classes. Note that here we use the notation $O(f, D)$ to denote some oracle, which might be either $\mathrm{MQ}(f)$ or $\mathrm{REX}(f, D)$.

Definition 1 ((ϵ, δ, O)-PAC function learner for $\mathcal{F}$). Let $\mathcal{F}$ be a class of functions, with $f : \{0, 1\}^n \to \{0, 1\}^m$ for all $f \in \mathcal{F}$. Given some fixed $\epsilon, \delta \in (0, 1)$, an algorithm $\mathcal{A}$ is an (ϵ, δ, O, D)-PAC function learner $\mathcal{F}$, if for all $f \in \mathcal{F}$, when given oracle access $O(f, D)$, with probability at least $1 - \delta$, $\mathcal{A}$ outputs a hypothesis h satisfying

$$\Pr_{x\sim D}[f(x) \neq h(x)] \leqslant \epsilon. \tag{3}$$

[1] We note that one can consider many other types of oracle access as well, such as, for example, statistical query access [15].

The algorithm $\mathcal{A}$ is an (ϵ, δ, O)-PAC function learner for $\mathcal{F}$ if it is a (ϵ, δ, O, D)-PAC function learner for $\mathcal{F}$ for all distributions D. We call $\mathcal{A}$ an efficient (ϵ, δ, O)-PAC function learner for $\mathcal{F}$ if the time complexity of $\mathcal{A}$ is $O((poly(n))$. We call $\mathcal{F}$ (ϵ, δ, O)-PAC-hard, if there exists no efficient (ϵ, δ, O)-PAC function learner for it.

The above definition refers to fixed accuracy and probability parameters (ϵ, δ), but we note that if these parameters are considered as variables, then an efficient learner is taken as one with time complexity $O(\text{poly}(n, 1/\epsilon, 1/\delta))$. In this case, the algorithm will be efficient with respect to the definition above for any $\epsilon, \delta = \Omega(1/\text{poly}(n))$. Additionally, we stress that the learning algorithm could be either classical or quantum. Indeed, as we will see in this work, it is possible that there exists an efficient quantum learning algorithm for a given class, but no efficient classical learning algorithm.

We would now like to generalize the PAC framework for learning functions to the natural and important problem of learning distributions. To formulate this problem rigorously, it is necessary to first introduce the different possible representations of a distribution that one might want to learn, namely, generators and evaluators:

Definition 2 (Generator and evaluator for D). Let D be a discrete probability distribution over $\{0, 1\}^n$. A generator for D is any function $\text{GEN}_D : \{0, 1\}^m \to \{0, 1\}^n$ that on uniformly random inputs outputs samples according to D, i.e.,

$$\Pr_{x\sim U(\{0,1\}^m)}[\text{GEN}_D(x) = y] = D(y). \tag{4}$$

An evaluator for D is any function $\text{EVAL}_D : \{0, 1\}^n \to [0, 1]$ that evaluates the probability mass assigned to an event with respect to D, i.e.,

$$\text{EVAL}_D(x) = D(x). \tag{5}$$

We note that evaluating and generating are indeed two distinct tasks and, in general, the ability to do the one does not imply the ability to do the other. To avoid any complexity-theoretic loopholes, during the course of this work, we assume that any evaluators or generators of interest are computable in time $poly(n)$. With the definition of a generator and an evaluator of a distribution at hand, we can now define PAC learners for distributions.

Definition 3 ((ϵ, δ)-PAC generator and evaluator learner for $\mathcal{D}$). Let $\mathcal{D}$ be a class of discrete probability distributions over $\{0, 1\}^n$. Given some fixed $\epsilon, \delta \in (0, 1)$, an algorithm $\mathcal{A}$ is an (ϵ, δ)-PAC (a) generator (GEN) or (b) evaluator (EVAL) learner of $\mathcal{D}$, if for all $D \in \mathcal{D}$, when given access to samples from D, with probability at least $1 - \delta$, $\mathcal{A}$ outputs a (a) generator or (b) evaluator for some distribution D', satisfying

$$d_{TV}(D, D') \leqslant \epsilon. \tag{6}$$

We call $\mathcal{A}$ an efficient (ϵ, δ)-PAC (generator or evaluator) learner for $\mathcal{D}$ if its time complexity is $O(\text{poly}(n))$. We call $\mathcal{D}$ (ϵ, δ)-PAC (generator or evaluator) hard if there is no efficient (ϵ, δ)-PAC (generator or evaluator) learner for $\mathcal{D}$.

In this work, we aim at proving a quantum-classical separation for distribution learning, and we want to do this by leveraging known classical hardness results for learning functions. In order to carry PAC function learning hardness results to the distribution learning regime, we use a generalization of a construction from Ref. [1], which allows one to define from any function a corresponding distribution. More specifically, we consider *induced distributions*, defined as follows:

Definition 4 (Induced distribution of f). For any function $f : \{0, 1\}^n \to \{0, 1\}^m$, we define the induced distribution $D_{f,P,\eta}$ as the discrete probability distribution over $\{0, 1\}^{n+m}$ via

$$D_{f,P,\eta}(x\|y) = \begin{cases} P(x)(1-\eta), \text{ if } f(x) = y \\ P(x)[\eta/(2^m - 1)], \text{ else} \end{cases} \tag{7}$$

for $x \in \{0, 1\}^n$ and $y \in \{0, 1\}^m$ and $\eta \in [0, 1]$ and P any probability distribution over $\{0, 1\}^n$. If $\eta = 0$, we simply write $D_{f,P}$. Similarly, we define the induced distribution class of the function class $\mathcal{F}$ by $\mathcal{D}_{\mathcal{F},P,\eta} = \{D_{f,P,\eta} | f \in \mathcal{F}\}$ and write $\mathcal{D}_{\mathcal{F},P}$ if $\eta = 0$.

We note that the induced distributions defined above are constructed precisely to allow a direct correspondence between oracle access to the function and sample access to the induced distribution. In particular, we note that a query to $\text{REX}(f, P, \eta)$ is precisely the same as drawing a sample from $D_{f,P,\eta}$.

The final background ingredient we require is that of *pseudorandom functions*, which, as we will soon see, allows us to prove distribution learning hardness results for the associated induced distributions. Intuitively, a pseudorandom function f is one that cannot be distinguished from a completely random function, by any polynomial-time algorithm that has oracle access to f, with nonnegligible probability. Before giving the definition of pseudorandom functions, let Θ be a parameter set, for which there exists an instance generation algorithm $\mathcal{IG}$ which on input 1^n outputs some "size n" parameter $\theta \in \Theta$. We call $\mathcal{IG}$ efficient and Θ efficiently sampleable if $\mathcal{IG}$ runs in time $O(\text{poly}(n))$. For more details on why we require this efficiently sampleable parameter set, we refer the reader to Ref. [11].

Definition 5 (Pseudorandom function collection). A set of efficiently computable functions

$$\{F_\theta : \mathcal{K}_\theta \times \mathcal{X}_\theta \to \mathcal{Y}_\theta | \theta \in \Theta\} \tag{8}$$

is called a (a) classic-secure or (b) weak-secure pseudorandom function collection if for all classical probabilistic polynomial time algorithms $\mathcal{A}$, all polynomials p, and all sufficiently large n, it holds that

$$\left| \Pr_{\substack{k\sim U(\mathcal{K}_\theta) \\ \theta\leftarrow\mathcal{IG}(1^n)}}[\mathcal{A}^{O(F_\theta(k,\cdot))}(\theta) = 1] - \Pr_{\substack{R\sim U(F:\mathcal{X}_\theta\to\mathcal{Y}_\theta) \\ \theta\leftarrow\mathcal{IG}(1^n)}}[\mathcal{A}^{O(R)}(\theta) = 1] \right| < \frac{1}{p(n)} \tag{9}$$

where $U(F : \mathcal{X}_\theta \to \mathcal{Y}_\theta)$ denotes the uniform distribution over all functions from $\mathcal{X}_\theta$ to $\mathcal{Y}_\theta$, $\mathcal{K}_\theta$ denotes the key space, $\mathcal{IG}$ is the efficient instance generation algorithm for the parameters θ, and $\mathcal{A}$ is given oracle access to (a) $O(f) = \text{MQ}(f)$ or (b) $O(f) = \text{REX}(f, U)$.

Note the core statement of the definition above: Any polynomial time algorithm $\mathcal{A}$ with access to the oracle $O(f)$ cannot determine with nonnegligible probability whether f was drawn from the function collection, or is a truly random function. While a *classic-secure* PRF cannot be distinguished from a random function using membership query access to the function, a *weak-secure* PRF cannot be distinguished from a

TABLE I. Given some hard-to-learn function class $\mathcal{F}$ that contains functions mapping from n bits to m bits, can we obtain EVAL or GEN learning hardness for $\mathcal{D}_{\mathcal{F},P}$? The table places our contributions within the context of prior work on this question. Firstly, in Theorem 1 we generalize the implicit techniques from Ref. [1] to show in Corollary 1 that PAC-hard functions translate to EVAL learning hardness for functions with $m = O(\log(n))$ output bits. Secondly, in Theorem 2 we show that EVAL learning hardness can generally be obtained from weak PRFs.

Function class $\mathcal{F}$	EVAL learning $\mathcal{D}_{\mathcal{F},P}$	GEN learning $\mathcal{D}_{\mathcal{F},P}$
Weak PRFs	Hard (Theorem 2)	Open question
PRFs	Hard (Corollary of Theorem 2)	Hard (Refs. [1] and [16])
PAC-hard $m = 1$	Hard (follows from Corollary 1)	Not necessarily hard (Ref. [16])
PAC-hard $m = O(\log(n))$	Hard (implicit in Ref. [1], explicit in Corollary 1)	Open question
PAC-hard $m = \Omega(\log(n))$	Open question	Open question

random function using random example access. Since there exists an algorithm that can simulate random example queries using membership queries, membership query access is more powerful than random example access and any classic-secure PRF is also weak-secure.

III. FROM SUPERVISED LEARNING TO DISTRIBUTION LEARNING

As we have mentioned, showing a quantum-classical distribution learning separation requires us to show two things: classical hardness and efficiency of quantum learning (for the same distribution learning task). For showing the former, a variety of techniques have been used previously, most of which exploit either PAC-hard functions or PRFs, primarily through the "function to distribution construction" in Definition 4 of the previous section. In order to consolidate and make explicit these techniques, we provide in this section an overview of known results and methods, as well as two extensions and generalizations. In particular, we first provide a theorem which abstracts and generalizes the technique of translating PAC function learning hardness to PAC evaluator learning hardness (used implicitly in Ref. [1]) for functions that map to $m = O(\log(n))$ bits, even in the case of noisy random examples. Additionally, we then provide a theorem which shows that one can prove evaluator learning hardness for distributions induced by *weak* PRFs. This strengthens, and makes applicable to density modeling, the technique used in Ref. [1] to prove hardness of *generative* modeling from PRFs. Table I puts our unique contributions in the context of prior work. We note that the primary focus of our work is on *density* modeling (i.e., evaluator learning) and we refer to Ref. [16] for a similar study focused on generative modeling, which considers additional "function to distribution" constructions to the one presented here.

A. Learning distributions induced by $m = O(\log(n))$-functions

We begin by establishing (in Theorem 1) a direct relationship between PAC learning a function class and PAC evaluator learning the induced distribution class for functions $f : \{0,1\}^n \to \{0,1\}^m$, with $m = O(\log(n))$. This result essentially generalizes and makes explicit a technique used implicitly in Ref. [1]. In particular, the formulation we provide in Theorem 1 makes clear (a) the applicability of the technique even in the case of *noisy* random example access in the function case (i.e., when $\eta \neq 0$) and (b) that one can consider functions with up to logarithmically many output bits. Additionally, this result serves as a warm-up to familiarize the reader with the definitions from Sec. II.

We start with a lemma that shows the equivalence between the error (or "loss") in function learning and the error in distribution learning.

Lemma 1 (Equivalence of distribution and function loss). Let $f, h : \{0,1\}^n \to \{0,1\}^m$ be two functions and $D_{f,P}$, $D_{h,P}$ be their two induced distributions. It holds for all distributions P that

$$\Pr_{x\sim P}[f(x) \neq h(x)] = d_{TV}(D_{f,P}, D_{h,P}). \tag{10}$$

Proof. See the Appendix. ■

To prove that hardness of learning a function class implies hardness of evaluator learning the induced distribution class, we will show that if we had an evaluator learner for some induced distribution class, then we can get a function learner for the underlying function class. To do this, we need a way to construct a function hypothesis from a given evaluator. A natural way to do this is to take a hypothesis that on any given input x outputs a y, such that $x\|y$ is assigned the highest probability under the evaluator. More formally, if D is some discrete probability distribution over $\{0,1\}^{n+m}$, then we let h be defined via $h(x) = \arg\max_y D(x\|y)$. This argmax construction is a natural way to obtain a function hypothesis from an evaluator, and below we show that this is optimal.

Lemma 2 (Optimal function hypothesis from an evaluator). Let D be some discrete probability distribution over $\{0,1\}^{n+m}$ and let h be defined via $h(x) = \arg\max_y D(x\|y)$, for $x \in \{0,1\}^n$, $y \in \{0,1\}^m$, then it holds for all functions $f : \{0,1\}^n \to \{0,1\}^m$ and all probability distributions P that

$$d_{TV}(D, D_{h,P}) \leqslant d_{TV}(D, D_{f,P}). \tag{11}$$

■

Proof. See the Appendix.

With these two lemmata in hand, we can prove the following result, which at a high level says that any PAC function learner for a given class of functions (mapping to at most logarithmically many output bits) can be turned into a evaluator learner for the induced distribution class, and vice versa.

Theorem 1. Let $\mathcal{F}$ be some function class consisting only of functions $f : \{0,1\}^n \to \{0,1\}^m$, for some $m = O(\log(n))$. Let $\mathcal{D}_{\mathcal{F},P,\eta}$ be the induced distribution class for some fixed probability distribution P over $\{0,1\}^n$ and $0 \leqslant \eta < \frac{1}{2}$.

(a) If $\mathcal{D}_{\mathcal{F},P,\eta}$ is efficiently (ϵ,δ)-PAC EVAL learnable, then $\mathcal{F}$ is efficiently $[2(\eta+\epsilon),\delta,\mathrm{REX}(P,\eta)]$-PAC learnable.

(b) If $\mathcal{F}$ is efficiently $[\epsilon,\delta,\mathrm{REX}(P,\eta)]$-PAC learnable, then $\mathcal{D}_{\mathcal{F},P,\eta}$ is efficiently $(\eta+\epsilon,\delta)$-PAC EVAL learnable.

Proof. First statement: Assume $\mathcal{A}_{\mathcal{D}}$ is an efficient (ϵ,δ)-PAC EVAL learner for $\mathcal{D}_{\mathcal{F},P,\eta}$, then $\mathcal{A}_{\mathcal{D}}$ outputs an evaluator $\mathrm{EVAL}_{D'}$ to $D_{f,P,\eta}$ with $d_{TV}(D',D_{f,P,\eta})\leqslant\epsilon$ for all $D_{f,P,\eta}\in\mathcal{D}_{\mathcal{F},P,\eta}$ with probability at least $1-\delta$. Note that D' has no subscripts, as it is not necessarily an induced distribution and can have any structure. All we know is that D' is ϵ-close to $D_{f,P,\eta}$. We construct now the algorithm $\mathcal{A}_{\mathcal{F}}$, which simulates $\mathcal{A}_{\mathcal{D}}$ by answering any sample accesses to $D_{f,P,\eta}$ with query[$\mathrm{REX}(f,P,\eta)$] and obtains $\mathrm{EVAL}_{D'}$. $\mathcal{A}_{\mathcal{F}}$ then outputs the function h as an algorithm, which on input x calculates $p_i=\mathrm{EVAL}_{D'}(x\|y_i)$ for all possible $y_i\in\{0,1\}^m$ and returns the y_i with the largest p_i. Since $m=O(\log(n))$, this is done in time $O(n)$. Due to Lemma 2, Lemma 1, and the triangle inequality, we get

$$\begin{aligned}
&\Pr_{x\sim P}[f(x)\neq h(x)]\\
&\quad=d_{TV}(D_{f,P},D_{h,P})\\
&\quad\leqslant d_{TV}(D_{f,P},D_{h,P,\eta})+d_{TV}(D_{f,P,\eta},D')+d_{TV}(D',D_{h,P})\\
&\quad\leqslant\eta+\epsilon+d_{TV}(D',D_{f,P})\\
&\quad\leqslant\eta+\epsilon+d_{TV}(D',D_{f,P,\eta})+d_{TV}(D_{f,P,\eta},D_{f,P})\\
&\quad\leqslant 2(\eta+\epsilon).
\end{aligned}\tag{12}$$

Thus, $\mathcal{A}_{\mathcal{F}}$ is an $[2(\epsilon+\eta),\delta,\mathrm{REX}(P,\eta)]$-PAC learner for $\mathcal{F}$.

Second statement: Assume $\mathcal{F}$ is efficiently $[\epsilon,\delta,\mathrm{REX}(P,\eta)]$-PAC learnable, then there exists an algorithm $\mathcal{A}_{\mathcal{F}}$ that for all $f\in\mathcal{F}$, with probability $1-\delta$ and query access to $\mathrm{REX}(f,P,\eta)$ outputs a hypothesis $h\in\mathcal{F}$ where $\Pr_{x\sim P}[f(x)\neq h(x)]\leqslant\epsilon$. We construct the learning algorithm $\mathcal{A}_{\mathcal{D}}$ to simulate $\mathcal{A}_{\mathcal{F}}$ (by answering any queries to the REX oracle with a sample from $D_{f,P,\eta}$) and output

$$\mathrm{EVAL}_{D_{h,P}}(x\|y)=\begin{cases}P(x),\text{ if }h(x)=y\\0,\text{ else}\end{cases}.\tag{13}$$

Since

$$d_{TV}(D_{h,P},D_{f,P})=\Pr_{x\sim P}[f(x)\neq h(x)]\leqslant\epsilon,\tag{14}$$

we have

$$\begin{aligned}
d_{TV}(D_{h,P},D_{f,P,\eta})&\leqslant d_{TV}(D_{h,P},D_{f,P})+d_{TV}(D_{f,P},D_{f,P,\eta})\\
&\leqslant\eta+\epsilon.
\end{aligned}\tag{15}$$

Thus, $\mathrm{EVAL}_{D_{h,P}}$ is an $(\eta+\epsilon)$-close evaluator of $D_{f,P,\eta}$ and $\mathcal{A}_{\mathcal{D}}$ is an $(\eta+\epsilon,\delta)$-PAC EVAL learner for $\mathcal{D}_{\mathcal{F},P,\eta}$. ■

Theorem 1 has been phrased in terms of efficient learnability, i.e., it shows how efficient learners for one problem imply efficient learners for another. However, we can straightforwardly rephrase Theorem 1 in terms of hardness implications. More specifically, we obtain from Theorem 1 the following simple corollary:

Corollary 1. Let $\mathcal{F}$ be some function class consisting only of functions $f:\{0,1\}^n\to\{0,1\}^m$, for some suitable $m=O(\log(n))$. Let $\mathcal{D}_{\mathcal{F},P,\eta}$ be the induced distribution class for some fixed probability distribution P over $\{0,1\}^n$ and $0\leqslant\eta<\frac{1}{2}$.

(a) If $\mathcal{D}_{\mathcal{F},P,\eta}$ is $(\eta+\epsilon,\delta)$-PAC EVAL hard, then $\mathcal{F}$ is $[\epsilon,\delta,\mathrm{REX}(P,\eta)]$-PAC hard.

(b) If $\mathcal{F}$ is $[2(\eta+\epsilon),\delta,\mathrm{REX}(P,\eta)]$-PAC hard, then $\mathcal{D}_{\mathcal{F},P,\eta}$ is (ϵ,δ)-PAC EVAL hard.

As claimed in Table I, the above corollary shows clearly that—at least for the case of functions with at most logarithmically many output bits—one can use function learning hardness results to prove distribution learning hardness results, and vice versa. We note that this correspondence holds even for function learning with *noisy* random examples, provided one considers the appropriate associated induced distribution class. Additionally, we note that one can also straightforwardly extend Theorem 1 to the setting in which both the function and distribution learner have *statistical query* access, as opposed to (noisy) random example and sample access as considered here.

B. Evaluator learning hardness from weak PRFs

In the previous section, we laid out how one can obtain EVAL learning hardness from PAC-hard functions by using the construction of induced distributions, for functions that map to $m=O(\log(n))$ bits. We are now interested in whether one can obtain EVAL learning hardness in a more general setting, or from other primitives. We will see in this section that we can indeed obtain EVAL learning hardness by using weak-secure PRFs as the distribution inducing functions. We note that this is very closely related to prior work in Ref. [1], where it was implicitly shown that one can use classic-secure PRFs to obtain generator learning hardness. This was made explicit and generalized in Ref. [11], which used this technique to prove a quantum-classical separation for generative modeling. However, in Ref. [11] it was posed as an open question whether or not one can obtain distribution learning hardness results from weak PRFs, and it is this question which we answer in the affirmative here, for the case of evaluator learning. Apart from allowing us to prove the quantum-classical distribution learning separation in Sec. IV, this result also opens the possibility of proving classical hardness results based on weaker assumptions than those necessary for candidate classic-secure PRFs, or hard-to-learn function classes.

To this end, we note that there is a very useful characterization of classic-secure PRFs, namely, the ability to withstand a so-called *inference exam* [17]. In particular, this characterization has been crucial for proving generator learning hardness for distributions induced by classic-secure PRFs [1,11]. As shown in Fig. 1, the inference exam is a procedure where a learner is tested to determine whether it has really learned anything about the function by asking whether it can distinguish between a function input-output pair and a random input-output pair. It has been shown that this distinguishing task is not possible efficiently, with convincing probability, if the function is a classic-secure PRF [17]. For our proof of EVAL learning hardness from weak-secure PRFs we will make use of a similar characterization of weak PRFs in terms of a slightly modified inference exam. More specifically, as shown in Fig. 1, in a conventional inference exam (defined in Ref. [17]), the distinguishing algorithm has (a) *membership query* access to the unknown function and (b)

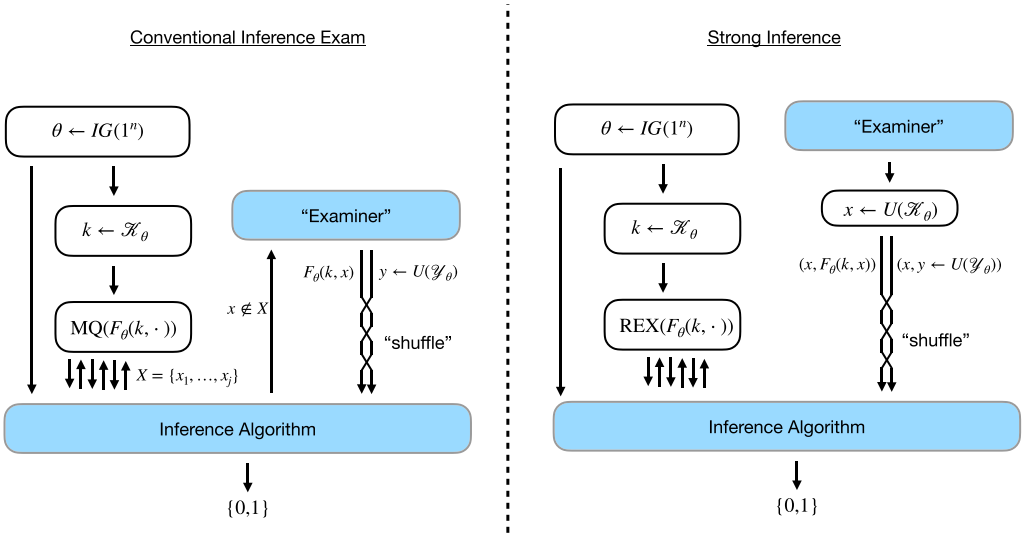

FIG. 1. Illustration of the difference between a conventional inference exam and a strong inference exam. Conventional inference exams have been introduced in Ref. [17] as an alternative characterization of PRFs, and we note here that strong inference exams can be used for the characterization of weak PRFs.

the ability to choose its own exam string. We will define a *strong* inference exam, in which the distinguishing algorithm has (a) only random example access to the unknown function and (b) gets given an exam pair drawn at random. Intuitively, we call this a strong inference exam as it is harder to pass than the conventional inference exam, due to the weaker oracle access and lack of ability to choose the exam. As we will see, we can characterize weak PRFs in terms of the existence or nonexistence of algorithms which succeed in the task of strong inference, analogously to how classic-secure PRFs can be characterized in terms of inference exams.

Let us now describe the strong inference exam that is used to characterize weak-secure PRFs, and we refer the interested reader to Refs. [11,17] for the details of standard inference exams. In the following we consider function collections $\{F_\theta : \mathcal{K}_\theta \times \mathcal{X}_\theta \to \mathcal{Y}_\theta | \theta \in \Theta\}$ which have the structure of PRF collections, where $\mathcal{K}_\theta$ is the so-called secret key space. In particular, let $k \in \mathcal{K}_\theta$ be some fixed but unknown secret key. We then define a strong inference exam as follows:

Definition 6 (Strong inference exam). Let $\mathcal{A}$ be some probabilistic polynomial time classical algorithm that "takes the exam." On input $\theta \in \Theta$, $\mathcal{A}$ can carry out any computation while having access to REX$[F_\theta(k,\cdot), U]$. After some time, when $\mathcal{A}$ signals that it is ready for the exam, $\mathcal{A}$ is presented two pairs (x', f_1) and (x', f_2) in random order, where $x' \sim U(\mathcal{X}_\theta)$, $f_1 = F_\theta(k, x')$, and $f_2 \sim U(\mathcal{Y}_\theta \setminus \{F_\theta(k, x')\})$. We say that $\mathcal{A}$ "passes the exam" if it correctly guesses which of the two pairs stems from the function F_θ and which was the random value from $U(\mathcal{Y}_\theta)$.

Analogously to the definition of Q inference in Ref. [17], we now define the notion of strong Q inference, which defines a lower bound on the probability of $\mathcal{A}$ passing the random example inference exam.

Definition 7 (Strong Q inference). Let Q be some function. We say that $\mathcal{A}$ strongly Q-infers the collection $\{F_\theta : \mathcal{K}_\theta \times \mathcal{X}_\theta \to \mathcal{Y}_\theta | \theta \in \Theta\}$ if for infinitely many n, given input $\theta \in \Theta$, it holds that

$$\Pr[\text{"}\mathcal{A} \text{ passes the exam"}] \geqslant 1/2 + 1/Q(n), \tag{16}$$

where the probability is taken uniformly over all possible choices of $\theta \leftarrow \mathcal{IG}(1^n)$, $k \in \mathcal{K}_\theta$, $x' \in \mathcal{X}_\theta$, $f_2 \in \mathcal{Y}_\theta$, and all possible orderings of the exam pairs. We say that a function collection can be polynomially strongly inferred if there exists a polynomial Q and a probabilistic polynomial time algorithm $\mathcal{A}$ which strongly Q-infers the collection.

We are now ready to state a core lemma, that is analogous to the result in Ref. [17], which characterizes PRFs via the strong polynomial Q-inference exam.

Lemma 3 (Weak-secure PRFs cannot be polynomially strongly inferred). Let $\mathcal{F} = \{F_\theta : \mathcal{K}_\theta \times \mathcal{X}_\theta \to \mathcal{Y}_\theta | \theta \in \Theta\}$ be a collection of efficiently computable functions. $\mathcal{F}$ is weak-secure pseudorandom if and only if $\mathcal{F}$ cannot be polynomially strongly inferred.

Proof. The proof is a direct generalization of the proof for Theorem 4 in Ref. [17]. ■

With the necessary definitions at hand, we can now present Theorem 2, which shows that distributions induced by weak-secure pseudorandom functions are hard to EVAL learn.

Theorem 2 (Classical evaluator learning hardness from weak pseudorandom functions). Let

$$\mathcal{F} = \{F_\theta : \mathcal{K}_\theta \times \mathcal{X}_\theta \to \mathcal{Y}_\theta | \theta \in \Theta\} \tag{17}$$

be a weak-secure pseudorandom function collection, where for all θ one has that $\mathcal{X}_\theta = \mathcal{Y}_\theta = \{0,1\}^n$ for some n. For all $\theta \in \Theta$ and all $k \in \mathcal{K}_\theta$, we define the induced probability distribution $D_{(\theta,k)} := D_{F_\theta(k,\cdot),U}$, and the associated distribution class $\mathcal{D} := \{D_{(\theta,k)} \mid \theta \in \Theta, k \in \mathcal{K}_\theta\}$. For all sufficiently large n, all $\epsilon < 1/9$, and $\delta \leqslant 1/5 - \Omega[1/\text{poly}(n)]$, there exists no efficient classical (ϵ, δ)-PAC EVAL learner of $\mathcal{D}$.

In order to prove Theorem 2, we require the following two technical lemmas.

Lemma 4. Let $f : \{0,1\}^n \to \{0,1\}^n$ and let D' be some distribution satisfying $d_{TV}(D_{f,U}, D') \leqslant \epsilon$, for some $\epsilon < 1/9$.

Then, for at least $3/4 \times 2^n$ strings x, it holds that

$$D'(x\|f(x)) \geqslant \frac{\epsilon}{2^n}. \tag{18}$$

Proof. See the Appendix. ■

Lemma 5. Let $f : \{0,1\}^n \to \{0,1\}^n$ and let D' be some distribution satisfying $d_{TV}(D_{f,U}, D') \leqslant \epsilon$. For at least $\frac{1}{2} \times (2^{2n} - 2^n)$ of the strings $x\|y$ with $y \neq f(x)$, it holds that

$$D'(x\|y) \leqslant \frac{4\epsilon}{2^{2n} - 2^n}. \tag{19}$$

Proof. See the Appendix. ■

Given the above lemmas, we can now prove Theorem 2.

Proof for Theorem 2. The proof will be by contradiction. To do this, we assume that, for some $\epsilon < 1/9$ and $\delta \leqslant 1/5 - \Omega[1/\text{poly}(n)]$, there exists a classical efficient (ϵ, δ)-PAC evaluator learner of $\mathcal{D}$, i.e., a polynomial time probabilistic classical algorithm $\tilde{\mathcal{A}}$, which for all $D_{(\theta,k)} \in \mathcal{D}$, when given sample access to $D_{(\theta,k)}$, outputs with probability at least $1 - \delta$, an evaluator $\text{EVAL}_{D'}$ for some distribution D' satisfying $d_{TV}(D_{(\theta,k)}, D') \leqslant \epsilon$. We now use this assumption to construct an efficient classical algorithm $\mathcal{A}$ that uses $\tilde{\mathcal{A}}$ to polynomially strongly infer $\mathcal{F}$. This strong polynomial inference is per definition of $\mathcal{F}$ not possible, resulting in the sought contradiction.

So, let us describe algorithm $\mathcal{A}$: When given access to $\text{REX}[F_\theta(k,\cdot), U]$, algorithm $\mathcal{A}$ starts by simulating algorithm $\tilde{\mathcal{A}}$. In particular, for every query made by $\tilde{\mathcal{A}}$, algorithm $\mathcal{A}$ queries $\text{REX}[F_\theta(k,\cdot), U]$, obtains some tuple $[x, F_\theta(k,x)]$, and then passes the string $x||F_\theta(k,x)$ to $\tilde{\mathcal{A}}$. As this is indistinguishable from a sample query to the distribution $D_{(\theta,k)}$, algorithm $\tilde{\mathcal{A}}$ will, after a polynomial number of queries, output with probability at least $1 - \delta$, an evaluator $\text{EVAL}_{D'}$ for some distribution D' satisfying $d_{TV}(D_{(\theta,k)}, D') \leqslant \epsilon$.

At this stage, $\mathcal{A}$ is ready to take the random example inference exam, and when presented the two exam pairs $s_1 = (x', f_1)$ and $s_2 = (x', f_2)$, $\mathcal{A}$ will run the strategy presented in Algorithm 1 to determine which of the two values f_1, f_2 is $F_\theta(k,x')$.

We will now analyze the probability that $\mathcal{A}$ passes the random example inference exam. Firstly, note that for $n \geqslant 3$, we have that $\epsilon/2^n > 4\epsilon/(2^{2n} - 2^n)$ and thus the conditions in lines 3 and 5 of Algorithm 1 cannot be true at the same time. Additionally, if algorithm $\tilde{\mathcal{A}}$ was successful, which happens

Algorithm 1 Exam strategy to infer $\mathcal{F}$

Input : Exam strings: $s_1, s_2 \in \{0,1\}^{2n}$,
Evaluator: $\text{EVAL}_{D'}$, parameters: n, ϵ
Output: Index of the exam string that has been produced by $F_\theta(k,x')$

1 $p_1 \leftarrow \text{EVAL}_{D'}(s_1)$;
2 $p_2 \leftarrow \text{EVAL}_{D'}(s_2)$;
3 **if** $p_1 \geq \frac{\epsilon}{2^n}$ *and* $p_2 \leq \frac{4\epsilon}{2^{2n}-2^n}$ **then**
4 | return 1;
5 **else if** $p_1 \leq \frac{4\epsilon}{2^{2n}-2^n}$ *and* $p_2 \geq \frac{\epsilon}{2^n}$ **then**
6 | return 2;
7 **else**
8 | return *random* $\sim U(\{1,2\})$;
end

with probability at least $1 - \delta$, then it follows from Lemmas 4 and 5, as well as the promise that $\epsilon < 1/9$, that

$$\Pr_{x' \sim U(\mathcal{X}_\theta)}\left[\text{EVAL}_{D'}(x'\|F_\theta(k,x')) \geqslant \frac{\epsilon}{2^n}\right] \geqslant \frac{3}{4}, \tag{20}$$

$$\Pr_{x' \sim U(\mathcal{X}_\theta)}\left[\text{EVAL}_{D'}(x'\|F_\theta(k,x')) < \frac{\epsilon}{2^n}\right] < \frac{1}{4}, \tag{21}$$

$$\Pr_{\substack{x' \sim U(\mathcal{X}_\theta) \\ u \sim U(\mathcal{Y}_\theta \setminus \{F_\theta(k,x')\})}}\left[\text{EVAL}_{D'}(x'\|u) \leqslant \frac{4\epsilon}{2^{2n} - 2^n}\right] \geqslant \frac{1}{2}, \tag{22}$$

$$\Pr_{\substack{x' \sim U(\mathcal{X}_\theta) \\ u \sim U(\mathcal{Y}_\theta \setminus \{F_\theta(k,x')\})}}\left[\text{EVAL}_{D'}(x'\|u) > \frac{4\epsilon}{2^{2n} - 2^n}\right] < \frac{1}{2}. \tag{23}$$

From the above, we can now bound the probability that algorithm $\mathcal{A}$ is successful, conditioned on $\tilde{\mathcal{A}}$ being successful. To do this, we denote the event that "Algorithm 1 returns on line ① given that $\tilde{\mathcal{A}}$ was successful" by "①." Using this, we have the following.

Case 1. $s_1 = x'\|F_\theta(k,x')$ and $s_2 = x'\|u$:

$$\Pr[④] \geqslant \tfrac{1}{2} \times \tfrac{3}{4} = \tfrac{3}{8}, \tag{24}$$

$$\Pr[⑥] < \tfrac{1}{2} \times \tfrac{1}{4} = \tfrac{1}{8}, \tag{25}$$

and therefore

$$\begin{aligned} &\Pr[\text{Algorithm 1 returns 1} \mid \tilde{\mathcal{A}} \text{ successful}] \\ &= \Pr[④] + \tfrac{1}{2}\Pr[⑧] \\ &= \Pr[④] + \tfrac{1}{2}[1 - (\Pr[④] + \Pr[⑥])] \\ &= \tfrac{1}{2} + \tfrac{1}{2}\Pr[④] - \tfrac{1}{2}\Pr[⑥], \\ &\geqslant \tfrac{5}{8}. \end{aligned} \tag{26}$$

Case 2. $s_1 = x'\|u$ and $s_2 = x'\|F_\theta(k,x')$:

$$\Pr[④] < \tfrac{1}{2} \times \tfrac{1}{4} = \tfrac{1}{8}, \tag{27}$$

$$\Pr[⑥] \geqslant \tfrac{1}{2} \times \tfrac{3}{4} = \tfrac{3}{8}, \tag{28}$$

and therefore

$$\begin{aligned} &\Pr[\text{Algorithm 1 returns 2} \mid \tilde{\mathcal{A}} \text{ successful}] \\ &= \Pr[⑥] + \tfrac{1}{2}\Pr[⑧] \\ &= \Pr[⑥] + \tfrac{1}{2}[1 - (\Pr[⑥] + \Pr[④])] \\ &= \tfrac{1}{2} + \tfrac{1}{2}\Pr[⑥] - \tfrac{1}{2}\Pr[④] \\ &\geqslant \tfrac{5}{8}. \end{aligned} \tag{29}$$

Thus, taking both cases together, we have that

$$\Pr[\text{Algorithm 1 returns correct index} \mid \tilde{\mathcal{A}} \text{ successful}] \geqslant \tfrac{5}{8}. \tag{30}$$

Using the above, we can now lower bound the probability that $\mathcal{A}$ passes the exam, to get

$$\begin{aligned} &\Pr[\mathcal{A} \text{ passes exam}] \\ &= \Pr[\text{Algorithm 1 returns correct index} \mid \tilde{\mathcal{A}} \text{ successful}] \\ &\quad \times \Pr[\tilde{\mathcal{A}} \text{ successful}] \\ &\quad + \Pr[\text{Algorithm 1 returns correct index} \mid \tilde{\mathcal{A}} \text{ unsuccessful}] \end{aligned}$$

$$
\begin{aligned}
&\times \Pr[\tilde{\mathcal{A}} \text{ unsuccessful}] \\
&\geqslant \Pr[\text{Algorithm 1 returns correct index} \mid \tilde{\mathcal{A}} \text{ successful}] \\
&\quad \times \Pr[\tilde{\mathcal{A}} \text{ successful}] \\
&\geqslant \frac{5}{8}(1-\delta) \\
&= \frac{5}{8} - \frac{5}{8}\delta \\
&\geqslant \frac{5}{8} - \frac{5}{8}\left[\frac{1}{5} - \Omega\left(\frac{1}{\text{poly}(n)}\right)\right] \\
&= \frac{1}{2} + \Omega\left(\frac{1}{\text{poly}(n)}\right). \qquad (31)
\end{aligned}
$$

Therefore, $\mathcal{A}$ polynomially strongly infers $\mathcal{F}$, which contradicts the assumption that $\mathcal{F}$ is weak-secure pseudorandom. ■

IV. A QUANTUM-CLASSICAL SEPARATION FOR DENSITY MODELING

In this work, we are interested in obtaining a quantum-classical separation for density modeling (evaluator learning). So far, we have seen in Sec. III B that when we instantiate the "function to distribution" construction with a weak-secure PRF, we can achieve a classical hardness result. The question, therefore, is whether there is a weak-secure PRF which allows us to also prove an efficient quantum learning result. In this section we show that by using the PRF previously utilized in Ref. [11] to show a separation for GEN learning, we can also achieve a separation for EVAL learning.

To begin, we restate the definition of the PRF used in Ref. [11], and we refer there for additional details and discussion. Let $p \in \mathbb{N}$, we say that an element $y \in \mathbb{Z}_q$ is a quadratic residue modulo p if there exists an $x \in \mathbb{Z}_q$ such that $x^2 \equiv y \mod p$. Additionally, we say that p is a safe prime if $p = 2q+1$ with q prime. Let QR_p be the set of quadratic residues modulo p and $\{\text{QR}_p\}$ be the set of such sets where p is a safe prime. Define the parameter set $\mathcal{P}_{(p,g,g^a)}$ as the infinite set of all tuples of the form (p, g, g^a), where p is some safe prime, g is a generator for QR_p, and $a \in \mathbb{Z}_q$. We denote the subset of all such tuples in which p is an n-bit prime as $\mathcal{P}_{n,(p,g,g^a)}$, and we note that $\mathcal{P}_{(p,g,g^a)} = \bigcup_{n\in\mathbb{N}} \mathcal{P}_{n,(p,g,g^a)}$ is an efficiently sampleable parameter set (see Ref. [11] for a description of the efficient instance generation algorithm). Now, given some safe prime $p = 2q+1$, define the function $f_p : \text{QR}_p \to \mathbb{Z}_q$ via

$$
f_p(x) = \begin{cases} x & \text{if } x \leqslant q \\ p - x & \text{if } x > q \end{cases}. \qquad (32)
$$

This allows us to define the functions $\tilde{G}^0_{(p,g,g^a)}$ and $\tilde{G}^1_{(p,g,g^a)} : \mathbb{Z}_q \to \mathbb{Z}_q$ via

$$
\tilde{G}^0_{(p,g,g^a)}(b) := f_p(g^b \mod p), \qquad (33)
$$

$$
\tilde{G}^1_{(p,g,g^a)}(b) := f_p(g^{ab} \mod p). \qquad (34)
$$

With this in hand, we can finally define the function collection $\{F_{(p,g,g^a)} \mid (p, g, g^a) \in \mathcal{P}_{p,g,g^a}\}$, where

$$
F_{(p,g,g^a)} : \mathbb{Z}_q \times \{0,1\}^n \to \mathbb{Z}_q \qquad (35)
$$

is defined algorithmically in Algorithm 2.

Algorithm 2 Algorithmic implementation of $F_{(p,g,g^a)}$

Input : Function input: $x \in \{0,1\}^n$,
Secret key: $k \in \mathbb{Z}_q$, Parameters: p, g, g^a
Output: The function value of $F_{(p,g,g^a)}(k, x)$

$b_0 \leftarrow k$;
for $1 \leq j \leq n$ **do**
  if $x_j = 0$ **then**
    $b_j \leftarrow \tilde{G}^0_{(p,g,g^a)}(b_{j-1})$;
  else if $x_j = 1$ **then**
    $b_j \leftarrow \tilde{G}^1_{(p,g,g^a)}(b_{j-1})$;
end
return b_n;

As shown in Ref. [11], the function collection $\{F_{(p,g,g^a)} \mid (p, g, g^a) \in \mathcal{P}_{p,g,g^a}\}$ is a classic-secure PRF collection, under the decisional Diffie-Hellman (DDH) assumption.[2] Given that any classic-secure PRF is also a weak-PRF, we know (from Theorem 2) that we can instantiate the "function to distribution" construction with $\{F_{(p,g,g^a)}(k,\cdot) \mid (p, g, g^a) \in \mathcal{P}_{p,g,g^a}, k \in \mathbb{Z}_q\}$ to obtain a distribution class $\mathcal{D} = \{D_{(p,g,g^a),k} \mid (p, g, g^a) \in \mathcal{P}_{p,g,g^a}, k \in \mathbb{Z}_q\}$ which is hard to evaluator learn. However, as done in Ref. [11], we will in fact consider a slight modification of the induced distribution class—for reasons which will soon become clear—in which an encoding of the parameters (p, g, g^a) is appended onto the samples. More specifically, recall that one samples from $D_{(p,g,g^a),k}$ by first drawing $x \leftarrow \{0,1\}^n$ and then outputting $x \| F_{(p,g,g^a)}(k, x)$. We define the distribution $\tilde{D}_{(p,g,g^a),k}$ as the distribution which is sampled from by first drawing $x \leftarrow \{0,1\}^n$ and then outputting $x \| F_{(p,g,g^a)}(k, x) \| (p, g, g^a)$, i.e., the exact same distribution as $D_{(p,g,g^a),k}$, but with the parameters appended to each sample. Naturally, we then define the distribution class as

$$
\tilde{\mathcal{D}} = \{\tilde{D}_{(p,g,g^a),k} \mid (p, g, g^a) \in \mathcal{P}_{p,g,g^a}, k \in \mathbb{Z}_q\}. \qquad (36)
$$

As discussed in Ref. [11], given the fact that any candidate inference algorithm for a PRF (or weak PRF) is also given the parameters of the unknown PRF (see Fig. 1), the proof of Theorem 2 is unaffected if one uses the distribution class $\tilde{\mathcal{D}}$ in place of $\mathcal{D}$. As such, we obtain the following corollary from Theorem 2 and the fact that $\{F_{(p,g,g^a)} \mid (p, g, g^a) \in \mathcal{P}_{p,g,g^a}\}$ is a classic-secure PRF collection under the DDH assumption:

Corollary 2 (Evaluator learning hardness of $\tilde{\mathcal{D}}$). Under the decisional Diffie-Hellman assumption, for all sufficiently large n, all $\epsilon < 1/9$, and all $\delta \leqslant 1/5 - \Omega[1/\text{poly}(n)]$, there is no efficient classical (ϵ, δ)-PAC EVAL learner for $\tilde{\mathcal{D}}$.

We would now like to show that one can indeed obtain an efficient quantum evaluator learner for $\tilde{\mathcal{D}}$. To this end, we start with the following observation.

Observation 1 (Exact evaluator from knowing the secret key). For all $\tilde{D}_{(p,g,g^a),k} \in \tilde{\mathcal{D}}$, given the secret key k, along with parameters (p, g, g^a), one can output an efficient exact evaluator of $\tilde{D}_{(p,g,g^a),k}$.

[2] A reader familiar with pseudorandom functions may recognize Algorithm 2 as the Goldreich-Goldwasser-Micali construction of a PRF, from the pseudorandom generator implicit in the Diffie-Hellman assumption.

The above observation can be easily understood by the considering the following evaluator:

$$\mathrm{EVAL}_{\tilde{D}_{(p,g,g^a),k}}(x\|y\|(p,g,g^a)) = \begin{cases} \frac{1}{2^n}, & \text{if } F_{(p,g,g^a)}(k,x) = y \\ 0, & \text{else} \end{cases}. \tag{37}$$

It follows from the construction of $F_{(p,g,g^a)}$ that the evaluator $\mathrm{EVAL}_{\tilde{D}_{(p,g,g^a),k}}$ is computable in poly-time if both k as well as (p, g, g^a) are known. In light of this, we see that learning an evaluator for $\tilde{D}_{(p,g,g^a),k}$ reduces to learning, from samples, the parameters (p, g, g^a) as well as the secret key k. However, by design, the parameters (p, g, g^a) come "for free" with each sample, and therefore one only needs to learn the secret key k. To this end, we note that precisely such an algorithm has already been constructed in Ref. [11]. More specifically, Ref. [11] has constructed an efficient quantum algorithm which, by using the exact quantum algorithm for discrete logarithms as a subroutine, can deterministically recover the secret key k from samples from $\tilde{D}_{(p,g,g^a),k}$. Putting it all together, we see that by using the efficient (deterministic) quantum key-learning algorithm from Ref. [11], coupled with the fact that the parameters (p, g, g^a) are given for free, and that together the key k and parameters (p, g, g^a) fully specify an (exact) evaluator for $\tilde{D}_{(p,g,g^a),k}$, we obtain the following corollary.

Corollary 3. $\tilde{\mathcal{D}}$ is quantumly efficiently $(\epsilon = 0, \delta = 0)$-PAC evaluator learnable.

Juxtaposing Corollary 2 with Corollary 3 yields a superpolynomial quantum-classical separation for density modeling, up to the *decisional Diffie-Hellman* assumption.

V. CONCLUSIONS AND OUTLOOK

In this work, we have provided a variety of rigorous insights into the relative power of classical and quantum computers for the task of density modeling. Specifically, we first provided an overview of techniques for proving distribution learning hardness from various classes of functions. Apart from providing a comprehensive picture of existing techniques, we have (a) provided a generalization of methods for proving distribution learning hardness from PAC hard-to-learn functions and (b) shown that weak-secure PRFs are sufficient to prove hardness of evaluator learning. Given this, we have then shown that there exists a density modeling task which is provably hard for classical computers, but can be solved by an efficient quantum learning algorithm. This separation contributes to the relatively scarce collection of machine learning type problems for which one can rigorously prove a quantum advantage [12]. In our outlook, we like to formulate the following open research questions:

(1) Can one get a computational separation (possibly with a fault-tolerant quantum computer) for a realistic learning task? Indeed, the learning task considered in this work involved a synthetic and highly fine-tuned distribution that almost certainly does not appear naturally and is not of any practical relevance. As such, it is still an open question whether one can find a practical—or "real world"—learning task for which quantum computers offer a superpolynomial speedup.

(2) Regarding the question phrased above: In this work we have considered highly fine-structured distributions constructed from cryptographic primitives, and it is certainly true that, while very helpful for proving separations, this approach is limited in its ability to make statements about "natural" distributions of interest. As discussed at length in this manuscript, any quantum-classical separation requires (a) a classical hardness result and (b) a quantum learnability result. As a refinement of the above question one could therefore ask whether there exist more natural distributions for which there are known hardness results, and for which quantum learnability is plausible. To this end, we note that there are known classical hardness results for a variety of natural distributions, including restricted Boltzmann machines [18], hidden Markov models [19], graphical models [20], and mixture models [21]. Unfortunately, however, the majority of these hardness results are *statistical* in nature (as opposed to *computational*) and therefore also immediately hold for quantum learning algorithms with classical data access (e.g., samples or statistical queries). As such, a natural first-step open question is whether there exist natural distribution classes, outside of those constructed directly from cryptographic primitives, admitting computational lower bounds for classical learning.

(3) Furthermore, a major question is whether (even for learning problems that are synthetic and not of any practical relevance) one can prove a quantum advantage using a quantum algorithm that works on *noisy, near-term quantum devices* instead of large-scale error-corrected quantum computers. Indeed, it is the hope that by formalizing and abstracting methods for proving classical hardness results in distribution learning, this work stimulates and facilitates such research efforts. Below we detail two specific approaches to answering this question:

(a) Firstly, can one find a quantum-classical learning separation based on weak- but not classic-secure PRFs—more specifically, a separation which requires a weaker assumption than that necessary for the existence of classic-secure PRFs? The hope is that whichever assumption is used for classical hardness can be broken by *near-term* quantum devices. What first comes to mind when pursuing this idea is to use weak-secure PRFs based on the hardness of *learning parity with noise* (LPN) [22]. In particular, while learning such a PRF classically is believed to be hard, there are efficient quantum learning algorithms [23]. However, these quantum algorithms require access to a quantum random example oracle, and it is not clear how to overcome this limitation. Indeed, it is an interesting question of independent interest whether there exist candidate weak PRFs which are not secure against near-term quantum adversaries.

(b) Secondly, we note that *the output distributions of quantum circuits themselves* are a very natural candidate for a distribution class which is classically hard to learn, but efficiently quantum learnable using near-term quantum learning algorithms based on variational updates of parametrized quantum circuits [24]. In particular, such distributions are perfectly suited to the inductive bias of "quantum Born machines" [25] (which suggests the possibility of efficient quantum learning) while simultaneously

being classically hard to simulate (which suggests the possibility of classical hardness for learning). Unfortunately, recent theoretical results have shown that, contrary to expectations, the output distributions of superlogarithmic depth local quantum circuits are statistically hard to learn for both quantum and classical learning algorithms. The question of shorter depths remains open.

(4) Finally, the answers to the open questions in Table I are certainly interesting and important.

ACKNOWLEDGMENTS

We would like to thank Thomas Vidick for discussions. R.S. is very grateful to Alex Nietner, Marcel Hinsche, and Marios Ioannou for many discussions and insights into both quantum and classical distribution learning. The authors acknowledge partial funding by the Einstein Research Unit "Perspectives of a quantum digital transformation: Near-term quantum computational devices and quantum processors" of the Berlin University Alliance. J.E. and R.S. have also received funding from the DFG under Germany's Excellence Strategy–The Berlin Mathematics Research Center MATH+ (EXC-2046/1, Project ID 390685689) and CRC 183, the BMWK (PlanQK, EniQmA), the BMBF (Hybrid), and the QuantERA (HQCC). This research is also part of the Munich Quantum Valley (K8), which is supported by the Bavarian State Government with funds from the Hightech Agenda Bayern Plus. J.-P.S. received funding from the Berlin Institute for the Foundations of Learning and Data (BIFOLD, Grant No. 01IS18037A).

APPENDIX: DEFERRED PROOFS

In this Appendix, we provide proofs that were omitted in the main text. For convenience, we restate the result before providing the proof. We start with Lemma 1:

Lemma 1 (Equivalence of distribution and function loss). Let $f, h : \{0,1\}^n \to \{0,1\}^m$ be two functions and $D_{f,P}$, $D_{h,P}$ be their two induced distributions. It holds for all distributions P that

$$\Pr_{x\sim P}[f(x) \neq h(x)] = d_{TV}(D_{f,P}, D_{h,P}). \tag{10}$$

Proof. We have

$$\begin{aligned} d_{TV}(D_{f,P}, D_{h,P}) &= \frac{1}{2}\sum_{x\in\{0,1\}^n} \underbrace{\sum_{y\in\{0,1\}^m} \left|D_{f,P}(x\|y) - D_{h,P}(x\|y)\right|}_{\substack{=0, \text{ if } f(x)=h(x) \\ =2\times P(x), \text{ if } f(x)\neq h(x)}} \\ &= \sum_{x\in\{0,1\}^n} P(x)\times\{1-\mathbb{1}[f(x),h(x)]\} \\ &= \Pr_{x\sim P}[f(x)\neq h(x)]. \end{aligned} \tag{A1}$$

■

Next, we have Lemma 2:

Lemma 2 (Optimal function hypothesis from an evaluator). Let D be some discrete probability distribution over $\{0,1\}^{n+m}$ and let h be defined via $h(x) = \arg\max_y D(x\|y)$, for $x \in \{0,1\}^n$, $y \in \{0,1\}^m$, then it holds for all functions $f : \{0,1\}^n \to \{0,1\}^m$ and all probability distributions P that

$$d_{TV}(D, D_{h,P}) \leqslant d_{TV}(D, D_{f,P}). \tag{11}$$

Proof. By definition, we have that

$$\begin{aligned} d_{TV}(D, D_{h,P}) &= \frac{1}{2}\sum_x\sum_y |D(x\|y) - D_{h,P}(x\|y)| \\ &= \frac{1}{2}\sum_x\left(\sum_{y\neq h(x)} D(x\|y) + |D[x\|h(x)] - P(x)|\right) \\ &:= \frac{1}{2}\sum_x L(x,h). \end{aligned} \tag{A2}$$

Now, we would like to show that $d_{TV}(D, D_{f,P}) - d_{TV}(D, D_{h,P}) \geqslant 0$. Note, we have that

$$d_{TV}(D, D_{f,P}) - d_{TV}(D, D_{h,P}) = \frac{1}{2}\sum_x [L(x,f) - L(x,h)], \tag{A3}$$

and therefore it is sufficient to show that $L(x,f) - L(x,h) \geqslant 0$ for all x. To this end,

$$\begin{aligned} L(x,f) - L(x,h) = &\left[\left(\sum_{y\neq f(x),h(x)} D(x\|y)\right)\right. \\ &\left. + D(x\|h(x)) + |D(x\|f(x)) - P(x)|\right] \\ &- \left[\left(\sum_{y\neq f(x),h(x)} D(x\|y)\right)\right. \\ &\left. + D(x\|f(x)) + |D(x\|h(x)) - P(x)|\right] \\ = &\,|D(x\|f(x)) - P(x)| + D(x\|h(x)) \\ &- |D(x\|h(x)) - P(x)| - D(x\|f(x)). \end{aligned} \tag{A4}$$

Thus, we need to show that

$$\begin{aligned} &|D(x\|f(x)) - P(x)| + D(x\|h(x)) \\ &\geqslant |D(x\|h(x)) - P(x)| - D(x\|f(x)). \end{aligned} \tag{A5}$$

Due to how h is constructed, we have

$$D(x\|h(x)) \geqslant D(x\|f(x)) \tag{A6}$$

for all x and can express $D(x\|h(x)) = P(x) + \alpha$ and $D(x\|f(x)) = P(x) + \beta$, where $1 - P(x) \geqslant \alpha \geqslant \beta \geqslant -P(x)$. We plug this into the inequality (A5) and obtain

$$|\beta| + \alpha \geqslant \alpha + \beta. \tag{A7}$$

■

Finally, we present the two technical Lemmata from Sec. III B.

Lemma 3. Let $f : \{0,1\}^n \to \{0,1\}^n$ and let D' be some distribution satisfying $d_{TV}(D_{f,U}, D') \leqslant \epsilon$, for some $\epsilon < 1/9$. Then, for at least $3/4 \times 2^n$ strings x, it holds that

$$D'(x\|f(x)) \geqslant \frac{\epsilon}{2^n}. \tag{18}$$

Proof. Per contradiction, assume that the claim is false: thus, it holds that for at least $1/4 \times 2^n$ strings x,

$$D'(x\|f(x)) < \frac{\epsilon}{2^n}. \tag{A8}$$

It follows that

$$\begin{aligned} d_{TV}(D_{f,U}, D') &= \frac{1}{2}\sum_{x,y\in\{0,1\}^n} \left|D_{f,U}(x\|y) - D'(x\|y)\right| \\ &= \frac{1}{2}\sum_{x\in\{0,1\}^n} \left|\frac{1}{2^n} - D'(x\|f(x))\right| \\ &\quad + \sum_{\text{other } x,y} \left|D'(x\|y)\right| \\ &\geqslant \frac{1}{2}\sum_{x\in\{0,1\}^n} \left|\frac{1}{2^n} - D'(x\|f(x))\right| \\ &> \frac{1}{2}\left(\frac{2^n}{4}\right)\left[\frac{1}{2^n} - \frac{\epsilon}{2^n}\right] \\ &= \frac{1}{8}(1-\epsilon) \\ &> \epsilon \qquad \left(\text{when } \epsilon < \frac{1}{9}\right), \end{aligned} \tag{A9}$$

which contradicts the assumption. ■

Lemma 4. Let $f : \{0,1\}^n \to \{0,1\}^n$ and let D' be some distribution satisfying $d_{TV}(D_{f,U}, D') \leqslant \epsilon$. For at least $\frac{1}{2} \times (2^{2n} - 2^n)$ of the strings $x\|y$ with $y \neq f(x)$, it holds that

$$D'(x\|y) \leqslant \frac{4\epsilon}{2^{2n} - 2^n}. \tag{19}$$

Proof. Per contradiction, assume that the claim is false, and therefore, for at least $\frac{1}{2} \times (2^{2n} - 2^n)$ of the strings $x\|y$ with $y \neq f(x)$, it holds that

$$D'(x\|y) > \frac{4\epsilon}{2^{2n} - 2^n}. \tag{A10}$$

From this, it follows that

$$\begin{aligned} d_{TV}(D_{f,U}, D') &= \frac{1}{2}\sum_{x,y\in\{0,1\}^n} \left|D_{f,U}(x\|y) - D'(x\|y)\right| \\ &= \frac{1}{2}\sum_{x\in\{0,1\}^n} \left|\frac{1}{2^n} - D'(x\|f(x))\right| \\ &\quad + \sum_{\text{other } x,y} \left|D'(x\|y)\right| \\ &\geqslant \frac{1}{2}\underbrace{\sum_{\text{other } x,y} \left|D'(x\|y)\right|}_{(2^{2n}-2^n)\text{ many}} \\ &> \frac{1}{2} \times \frac{1}{2}\left(2^{2n} - 2^n\right)\left[\frac{4\epsilon}{2^{2n} - 2^n}\right] \\ &= \epsilon, \end{aligned} \tag{A11}$$

$$\tag{A12}$$

which contradicts the assumption. ■

[1] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie, in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '94 (Association for Computing Machinery, New York, 1994), pp. 273–282

[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature (London) **549**, 195 (2017).

[3] S. Arunachalam and R. de Wolf, A survey of quantum learning theory, arXiv:1701.06806.

[4] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arXiv:1307.0411.

[5] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91**, 045002 (2019).

[6] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Quantum Sci. Technol. **4**, 043001 (2019).

[7] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, Nat. Rev. Phys. **3**, 625 (2021).

[8] R. A. Servedio and S. J. Gortler, SIAM J. Comput. **33**, 1067 (2004).

[9] V. Dunjko, Y.-K. Liu, X. Wu, and J. M. Taylor, arXiv:1710.11160.

[10] Y. Liu, S. Arunachalam, and K. Temme, Nat. Phys. **17**, 1013 (2021).

[11] R. Sweke, J.-P. Seifert, D. Hangleiter, and J. Eisert, Quantum **5**, 417 (2021).

[12] C. Gyurik and V. Dunjko, On establishing learning separations between classical and quantum machine learning with classical data, arXiv:2208.06339.

[13] L. G. Valiant, Commun. ACM **27**, 1134 (1984).

[14] M. J. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory* (MIT Press, Cambridge, 1994).

[15] M. Kearns, J. ACM **45**, 983 (1998).

[16] D. Xiao, in *COLT 2010–The 23rd Conference on Learning Theory*, Haifa, Israel, June 27–29, 2010, edited by A. T. Kalai and M. Mohri (Omnipress, Madison, 2010), pp. 516–528.

[17] O. Goldreich, S. Goldwasser, and S. Micali, J. ACM **33**, 792 (1986).

[18] G. Bresler, F. Koehler, and A. Moitra, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (ACM, 2019), pp. 828–839.

[19] S. A. Terwijn, in *Grammatical Inference: Algorithms and Applications: 6th International Colloquium, ICGI 2002, Amsterdam, The Netherlands, September 23–25, 2002* (Springer, New York, 2002), pp. 261–268.

[20] G. Bresler, D. Gamarnik, and D. Shah, Adv. Neural Inf. Proc. Sys. **27**, 1062 (2014).

[21] I. Diakonikolas, D. M. Kane, and A. Stewart, in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, New York, 2017), pp. 73–84.

[22] A. Bogdanov and A. Rosen, *Tutorials on the Foundations of Cryptography* (Springer, Berlin, 2017), pp. 79–158.

[23] A. W. Cross, G. Smith, and J. A. Smolin, Phys. Rev. A **92**, 012327 (2015).

[24] B. Coyle, D. Mills, V. Danos, and E. Kashefi, npj Quantum Inf. **6**, 60 (2020).

[25] J.-G. Liu and L. Wang, Phys. Rev. A **98**, 062324 (2018).