Quantum machine learning advantages beyond hardness of evaluation

Riccardo Molteni *1, Simon C. Marshall †1, and Vedran Dunjko ‡1

¹applied Quantum algorithms (aQa), Leiden University, The Netherlands

Abstract

In recent years we have seen numerous rigorous proofs of quantum advantages in machine learning across various scenarios. The most general and least contrived advantages, involving data labeled by cryptographic or intrinsically quantum functions, rely on the infeasibility of classical polynomial-sized circuits to evaluate the true labeling functions. As a result, classical learning algorithms, with or without training data, cannot label new points accurately. While broad in scope, these results however reveal little about advantages stemming from the actual learning process itself.

In cryptographic settings, additional insights are possible. By leveraging the property of random-generatability—the ability to generate data classically—proofs of classical hardness for identification tasks, where the goal is to "just" identify the labeling function behind a dataset, even if that function is not classically tractable, are possible. Identification tasks are especially relevant in quantum contexts, as Hamiltonian learning and identifying physically meaningful order parameters fit in this paradigm. However, for quantum functions, random-generatability has been conjectured not to hold, leaving no known identification advantages for genuinely quantum settings.

In this work, we provide the first proofs of identification learning advantages for quantum functions under complexity-theoretic assumptions. First, we confirm the conjecture that quantum-hard functions are not random-generatable unless BQP is in the second level of the polynomial hierarchy, ruling out strategies analogous to those used for cryptographic functions. We then establish a new approach, first showing that *verifiable identification*— a task where the learner must rejects invalid datasets but solve the identification problem for valid ones —is intractable for classical learners, for quantum labeling functions, unless BQP is in the polynomial hierarchy. Then, we demonstrate that in specific task classes, the ability to solve just the identification problem implies verifiable identification within the polynomial hierarchy. This leads to our main result: there exists a broad class of identification tasks involving quantum functions that are efficiently solvable by quantum learners but intractable for classical learners unless BQP is contained in a low level of the polynomial hierarchy. These findings suggest that for many quantum-associated learning tasks, the entire learning process—not just final evaluation—gains significant advantages from quantum computation.

Contents

1	Introduction	3
_	Preliminaries 2.1 Definition of Random generatability	4
	2.2 PAC framework	5

^{*}r.molteni@liacs.leidenuniv.nl

[†]s.c.marshall@liacs.leidenuniv.nl

[‡]v.dunjko@liacs.leidenuniv.nl

	2.3 Definitions of the identification tasks	6
	2.4 Tools from complexity theory	9
3	Related works	10
3	Related Works	10
4	Hardness results for random generatability of quantum functions 4.1 Hardness of random generatability based on the assumption BQP $\not\subset$ PH	11 11
5	Hardness results for the verifiable identification problem 5.1 Hardness result for the verifiable identification problem with BQP functions	13 13
6	Hardness result for the identification problem - Non-verifiable case 6.1 Intermediate result - Hardness result for approximate-verifiable identification algorithm 6.2 Construction of an approximate-verifiable algorithm in the PH 6.2.1 Construction of an approximate-verifiable algorithm with c-distinct concepts 6.2.2 Construction of an approximate-verifiable algorithm for classes with similarity-preserving parametrizations	15 18 19 20 20
7	Discussion: some implications of our results 7.1 Hardness of identification for a physically motivated BQP-complete concept class	21 21 22 23
8	Acknowledgments	25
A	Further definitions from complexity theory	26
В	Proofs about random generatability B.1 Hardness of random generatability based on the assumption BPP/samp ^{BQP} ⊈ BPP B.2 Proofs of the theorems from the main text concerning the hardness of random generatability B.2.1 Hardness of exact random generatability	26 27 27 28
C	Proofs of the theorems from the main text concerning the identification task C.1 Proof of Proposition 2	30 30 30 32
Ъ	Dragf of Theorem 11	96

1 Introduction

A central question in quantum machine learning (QML) is understanding which learning problems inherently require a quantum computer for efficient solutions. As shown in [HBM⁺21], access to data enables classical learners to efficiently evaluate some hard-to-compute functions that are otherwise intractable for classical algorithms operating within BPP. Most of the known examples of learning problems where data does not aid in computing the target function involve cryptographic functions, for which random labeled samples can be generated efficiently using classical algorithms. While the proofs may be more involved, the intuition remains simple: if a classical machine could generate the data, then the data itself cannot be what makes a hard problem easy. In the context of QML, where the greatest advantages are intuitively expected for "fully quantum" functions (i.e., functions that are BQP-complete¹), the focus on cryptographic tasks was somewhat unsatisfactory. This limitation was addressed in [GD23] by employing stronger computational complexity assumptions and, crucially, by leveraging the fact that standard learning definitions require the learned model to evaluate new data points.

However, a different learning condition could be considered where the learning algorithm is solely required to identify the target labeling function from a set of labeled data. In this scenario, the classical learner would only need to output a description of the unknown target function, without needing to also evaluate it on new input points. From a fundamental standpoint, analyzing the hardness of the identification task in learning problems involving quantum functions helps clarify the source of learning separations. From a practical viewpoint, particularly in expected applications of quantum machine learning, identifying the data-generating function can in fact be the primary goal, such as in Hamiltonian learning or in tasks related to finding order parameters [GD23], as we will comment in the paper. Furthermore, as discussed below, it is known that in this regard better results can be achieved in the case of cryptographic problems [GD23, JGM+24]. However, as was proved in [GD23], separations for the identification problem cannot exist without assumptions on the hypothesis class (intuitively, the classical learner can always outputs the circuit for the whole quantum learner with hardwired data as the output). The main goal of this paper is then to determine the conditions under which a learning separation for fully quantum functions can already emerge from the identification step.

As mentioned above, the hardness of mere identification was proven for certain cryptographic functions. A key element in the corresponding proofs of hardness is the ability of a classical algorithm to generate random samples of the target function. In this paper, we use the term "random generatability" to describe the property of a function f that allows efficient generation of samples (x, f(x)) for random input x. We also use the term "quantum functions" to denote the characteristic functions of BQP or PromiseBQP complete languages (see Definitions 8 and 9). The main contributions of the paper are the following.

- 1. We show that quantum functions are not random generatable unless BQP is in the second level of the polynomial hierarchy. This result has consequences on quantum generative modelling as well, see Corollary 1.
- 2. We introduce the task of verifiable-identification, where the algorithm must reject invalid datasets, and solve identification correctly otherwise, and prove classical verifiable-identification for quantum target function is impossible, unless BQP is in HeurBPP^{NP}.
- 3. We identify sufficient conditions on learning algorithm and concept class such that "mere" classical identification implies approximate-verifiable identification (the algorithm must reject datasets unless more than two-thirds of the samples are labeled according to the same concept) within the polynomial hierarchy. This implies identification is classically intractable unless BQP is in HeurBPP^{NPNPNP} (two levels "up" in the hierarchy).
- 4. We provide examples of classes of learning tasks which satisfy the above condition, for which then the quantum advantage for identification tasks holds.

¹More precisely, the class BQP does not contain complete problems. Throughout this paper, whenever we refer to complete problems, we will instead consider the class PromiseBQP.

We note that it is currently believed that BQP does not lie within the PH, and we will discuss this further later in the text. In particular, the cryptographic functions used in the previously mentioned results are explicitly contained within the PH, whereas our separation results rely on the full computational power of BQP. The key insight in our proofs is that with a NP oracle, it is possible to effectively "invert" functions. In particular, if an efficient algorithm could output, given a training set as input, the description of the correct function which labeled the samples, then we could invert that algorithm to generate a consistent dataset with the target function. This would also allow us to obtain a training set containing any x, enabling us to evaluate f(x).

The rest of the paper is organized as follows. In Section 2, we introduce the two tasks of random generatability and identification, along with essential definitions from complexity theory. In Section 4 we prove hardness for random generatability for quantum functions. In Section 5, we show hardness results for the identification task under the assumption that the learning algorithm can reject invalid datasets. In Section 6, we significantly relax this assumption and show hardness results for the identification task without requiring the learning algorithm to reject invalid datasets. Finally, in Section 7, we discuss the implications of our findings for various practically relevant tasks.

2 Preliminaries

We now provide the precise definitions of the two tasks we are addressing in this work. We will prove our corresponding hardness results in the next sections.

2.1 Definition of Random generatability

Given a uniform family of functions $f = \{f_n\}_{n \in \mathbb{N}}$, with $f_n : \{0,1\}^n \to \{0,1\}$, we say that f is "random-generable" if there exists a uniform (randomized) algorithm capable of producing samples $(x, f_n(x))$ with x uniformly sampled from the set $\{0,1\}^n$ efficiently for any n. The concept of functions that permit an efficient procedure to generate random samples (x, f(x)) was introduced in [AS06] under the term "random verifiability". In this paper, we refer to the same property as "random generatability" to avoid potential confusion with the "verifiable case" of the identification task described in Def. 5. Specifically, we consider two cases: exact random generatability in Def. 1, where the algorithm is not allowed to make any errors, and approximate random generatability in Def. 2, where the algorithm outputs samples from a distribution only close to the true target distribution.

Definition 1 (Exact random generatability). Let $f = \{f_n\}_{n \in \mathbb{N}}$ be a uniform family of functions, with $f_n : \{0,1\}^n \to \{0,1\}^2$. f is exact random generatable under the input distribution \mathcal{D} if there exists an efficient uniform (randomized) algorithm $\mathcal{A}_{\mathcal{D}}$ such that given as input a description of f_n for any n and a random string r is such that with probability 1:

$$\mathcal{A}_{\mathcal{D}}(f_n, r) = (x_r, f_n(x_r)). \tag{1}$$

When r is chosen uniformly at random from the set of all random strings with length polynomial in n, then $(x_r, f_n(x_r)) \sim \pi^f(\mathcal{D})$, where $\pi^f(\mathcal{D})$ samples x from the target distribution \mathcal{D} over $x \in \{0, 1\}^n$ and assigns the corresponding label $f_n(x_r) \in \{0, 1\}$.

If \mathcal{D} is not efficiently exactly samplable then no exact random algorithm $\mathcal{A}_{\mathcal{D}}$ can exist for generating samples $(x, f_n(x))$ for any given family of functions f. However, we note that we can set \mathcal{D} to be the uniform distribution over all the $x \in \{0, 1\}^n$ and in this case there exists such an algorithm $\mathcal{A}_{\mathcal{D}}$ whenever there exists an efficient method to evaluate f_n on every x. We also consider the case where the random algorithm $\mathcal{A}_{\mathcal{D}}$ can make mistakes, both in the distribution of the x's and in the labelling. We address both of the errors in the following definition.

²Committing slight abuse of notation, we will use f_n to denote both the function itself and its description.

Definition 2 (Approximate random generatability). Let $f = \{f_n\}_{n \in \mathbb{N}}$ be a uniform family of functions, with $f_n : \{0,1\}^n \to \{0,1\}$. f is approximately random generatable under the distribution \mathcal{D} if there exists an efficient uniform (randomized) algorithm $\mathcal{A}_{\mathcal{D}}$ such that given as input a description of f_n for any n, a random string r and an error value ϵ outputs:

$$\mathcal{A}_{\mathcal{D}}(f_n, 0^{1/\epsilon}, r) = (x_r, f_n(x_r)) \tag{2}$$

with $(x_r, f_n(x_r)) \sim \pi_{\epsilon}^f(\mathcal{D})$ when r is sampled uniformly at random from the distribution of all the random strings (with polynomial size). In particular, $\pi_{\epsilon}^f(\mathcal{D})$ is a probability distribution over $\{0, 1\}^n \times \{0, 1\}$ which satisfies: $\|\pi_{\epsilon}^f(\mathcal{D}) - \pi^f(\mathcal{D})\|_{TV} \leq \epsilon$, where $\pi^f(\mathcal{D})$ is the same distribution defined in Def. 1.

As the total variation distance between two distributions p and q over $x \in \{0,1\}^n$ is defined as $||p-q||_{TV} = \frac{1}{2} \sum_x |p(x)-q(x)|$, the algorithm \mathcal{A} in Def. 2 is allowed to make mistakes both with respect to the probability distribution of the outputted xs (e.g. they may not be generated from the uniform distribution) and with respect to the assigned labels.

2.2 PAC framework

The results in this paper are expressed in the standard terms of the efficient probably approximately correct (PAC) learning framework [KV94b, Moh18]. In the case of supervised learning, a learning problem in the PAC framework is defined by a concept class $C = \{C_n\}_{n \in \mathbb{N}}$, where each C_n is a set of concepts, which are functions from some input space \mathcal{X}_n (in this paper we assume \mathcal{X}_n is a subset of $\{0,1\}^n$) to some label set \mathcal{Y}_n (in this paper we assume \mathcal{Y}_n is $\{0,1\}$, unless stated otherwise). The learning algorithm receives on input samples of the target concepts $T = \{(x_\ell, c(x_\ell))\}_\ell$, where $x_\ell \in \mathcal{X}_n$ are drawn according to target distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$. Finally, the learning algorithm has a hypothesis class $\mathcal{H} = \{\mathcal{H}_n\}_{n \in \mathbb{N}}$ associated to it, and the learning algorithm should output a hypothesis $h \in \mathcal{H}_n$ – which is another function from \mathcal{X}_n to \mathcal{Y}_n – that is in some sense "close" (see Eq. (3) below) to the concept $c \in \mathcal{C}_n$ generating the samples in T.

Definition 3 (Efficient probably approximately correct learnability). A concept class $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$ is efficiently PAC learnable under target distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ if there exists a hypothesis class $\mathcal{H} = \{\mathcal{H}_n\}_{n \in \mathbb{N}}$ and a (randomized) learning algorithm \mathcal{L} with the following property: for every $c \in \mathcal{C}_n$, and for all $0 < \epsilon < 1/2$ and $0 < \delta < 1/2$, if \mathcal{L} receives in input a training set T of size greater than $M \in \mathcal{O}(\text{poly}(n))$, then with probability at least $1 - \delta$ over the random samples in T and over the internal randomization of \mathcal{L} , the learning algorithm \mathcal{L} outputs a specification³ of some $h \in \mathcal{H}_n$ that satisfies:

$$\Pr_{x \sim \mathcal{D}_n} \left[h(x) \neq c(x) \right] \le \epsilon.$$
 (3)

Moreover, the learning algorithm \mathcal{L} must run in time $\mathcal{O}(\text{poly}(n, 1/\epsilon, 1/\delta))$. If the learning algorithm runs in classical (or quantum) polynomial time, and the hypothesis functions can be evaluated efficiently on a classical (or quantum) computer, we refer to the concept class as *classically learnable* (or *quantumly learnable*, respectively).

In classical machine learning literature, hypothesis functions are generally assumed to be efficiently evaluable by classical algorithms, as the primary goal is to accurately label new data points. In this paper, however, we focus on learning separations that arise from the inability of a classical algorithm to identify the target concept that labels the data, rather than from the hardness of evaluating the concept itself. Therefore, for our purposes we consider hypothesis functions that may be computationally hard to evaluate classically. Specifically, we will consider target concept functions, as defined in Def. 9, that are related to (Promise—) BQP complete languages. As we will discuss in the next subsection, obtaining a learning separation for the identification problem in this case requires restricting the hypothesis class available to the classical learning algorithm. A common approach is to define the hypothesis class as exactly the same set of

³The hypotheses (and concepts) are specified according to some enumeration $R: \cup_{n\in\mathbb{N}} \{0,1\}^n \to \cup_n \mathcal{H}_n$ (or, $\cup_n \mathcal{C}_n$) and by a "specification of $h\in\mathcal{H}_n$ " we mean a string $\sigma\in\{0,1\}^*$ such that $R(\sigma)=h$ (see [KV94b] for more details).

functions contained in the concept class that label the data. The classical learning algorithm will then have to correctly identify which is the concept, among the ones in the concept class, that labeled the data. As we will make clear later, the task is close to a variant of the PAC learning framework in Def. 3, called proper PAC learning.

Definition 4 (Proper PAC learning [KV94b]). A concept class $C = \{C_n\}_{n \in \mathbb{N}}$ is efficiently proper PAC learnable under the distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ if it satisfies the definition of PAC learnability given in Def. 3, with the additional requirement that the learning algorithm \mathcal{L} uses a hypothesis class \mathcal{H} identical to the concept class, i.e., $\mathcal{H} = \mathcal{C}$.

2.3 Definitions of the identification tasks

Imagine we are given a machine learning task defined by a concept class \mathcal{F} composed of 2^m target functions, each one specified by a vector alpha $\alpha \in \{0,1\}^m$ with m scaling polynomially with input size n. Formally, that means that there exists a function $e: S \subseteq \{0,1\}^m \to \mathcal{F}$ which is bijective and such that $e(\alpha) = f^\alpha \in \mathcal{F}$. By abuse of notation, in this paper we often use α to refer to $e(\alpha)$ when it is clear from the context that we mean the function f^α , rather than the vector $\alpha \in \{0,1\}^m$. Given a training set of points labeled by one of these concepts, the goal of the *identification learning algorithm* is to "recognize" the target concept which labeled the data and output the corresponding α^4 .

In this paper we want to show that this identification task, which we will precisely define in this section, is classically hard when the concept classes contain (Promise—) BQP-complete functions as defined in Def. 9. Without additional restrictions on the hypothesis class, however, it is important to note that no learning separation for the identification task can occur. If the hypothesis class includes all quantumly evaluable functions, it is always possible to consider a hypothesis function that effectively "offloads" the quantum learning process onto the hypothesis itself. In this scenario, the hypotheses are indexed by the possible datasets, allowing a classical algorithm to readily identify the correct hypothesis solely from the training points it receives (see Lemma 1 in [GD23] for further details). To remedy this, it is necessary to restrict the hypothesis class of the classical learning algorithm. In this work we consider the natural case of proper PAC learning in Def. 4 and we will limit the hypothesis class to be exactly the concept class.

In this paper, we address two closely related but subtly different versions of identification tasks, for which we provide precise definitions below. In the first one, we assume the learning algorithm can decide if a dataset is *invalid*, i.e. if it is not consistent with any of the concepts in the concept class. In particular, we adopt the notion of consistency as defined in Definition 3 of [KV94b] and regard a dataset as valid if every point in it is labeled in accordance with a concept. In Section 5.1 we explain that this version of the task is closely related to the learning framework of the so-called consistency model found in the literature [KV94b, Moh18, Sch90] and we prove hardness for classical algorithms in solving it unless BQP languages are on average efficiently evaluable within the second level of the PH.

From this point onward, we will slightly abuse notation and use f or f^{α} to refer to a uniform family of functions, with one function defined for each input size n.

Definition 5 (Identification task - verifiable case). Let $\mathcal{F} = \{f^{\alpha}(x) \in \{0,1\} \mid \alpha \in \{0,1\}^m\}^5$ be a concept class. A *verifiable* identification algorithm is a (randomized) algorithm \mathcal{A}_B such that when given as input a set $T = \{(x_{\ell}, y_{\ell})\}_{\ell=1}^B$ of at least B pairs $(x, y) \in \{0,1\}^n \times \{0,1\}$, an error parameter $\epsilon > 0$ and a random string $r \in R$, it satisfies:

- If $\nexists \alpha \in \{0,1\}^m$ such that $y_\ell = f^\alpha(x_\ell)$ holds for all $(x_\ell,y_\ell) \in T$, then \mathcal{A}_B outputs "invalid dataset".
- If the samples in T come from the distribution \mathcal{D} , i.e. $x_{\ell} \sim \mathcal{D}$, and there exists an $\alpha \in \{0,1\}^m$ such that $y_{\ell} = f^{\alpha}(x_{\ell}) \in \{0,1\}$ holds for all $(x_{\ell}, y_{\ell}) \in T = \{(x_{\ell}, y_{\ell})\}_{\ell=1}^B$, then with probability 1δ it

⁴More precisely, the algorithm is allowed to output any $\tilde{\alpha}$ which is close in PAC condition with α . See Def. 5 and 6.

⁵Here and throughout the paper, we assume that the concepts are labeled by a vector α in $\{0,1\}^m$. However, it is not required that α spans the entire set of bitstrings in $\{0,1\}^m$. The key requirement is that m is sufficiently large to ensure that every concept in the concept class can be assigned a unique vector in $\{0,1\}^m$

outputs:

$$\mathcal{A}_B(T,\epsilon,r) = \tilde{\alpha}, \quad \tilde{\alpha} \in \{0,1\}^m, \tag{4}$$

satisfying $\mathbb{E}_{x \sim \mathcal{D}} |f^{\alpha}(x) - f^{\tilde{\alpha}}(x)| \leq \epsilon$.

We say that \mathcal{A}_B solves the identification task for a concept class \mathcal{F} under the input distribution \mathcal{D} if the algorithm works for any values of $\epsilon, \delta \geq 0$. The success probability $1 - \delta$ is over the training sets where the input points are sampled from the distribution \mathcal{D} and the internal randomization of the algorithm. The required minimum size B of the input set T scales as $poly(n,1/\epsilon,1/\delta)$, while the running time of the algorithm scales as poly(B, n).

It will be instructive to think about the algorithm \mathcal{A}_B as a mapping that, once ϵ and r are fixed, takes datasets $T \subseteq \{(x,y) \mid x \in \{0,1\}^n, y \in \{0,1\}\}$ with $|T| \geq B$, and outputs labels $\alpha \in \{0,1\}^m$. In Section 5.1 we will show that the verifiability condition plays a crucial role in our hardness result for the verifiable case of the identification task. In our main hardness result we show how to relax the verifiability condition on the learning algorithm and redefine the identification task as in Def. 7, considering a dataset valid if at least $\frac{2}{3}$ of the points agree with a single concept.

Definition 6 (Identification task - non-verifiable case). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class. A *non-verifiable* identification algorithm is a (randomized) algorithm \mathcal{A}_B such that when given as input a set $T = \{x_{\ell}, y_{\ell}\}_{\ell=1}^{B}$ of at least B pairs $(x,y) \in \{0,1\}^n \times \{0,1\}$, an error parameter $\epsilon > 0$ and a random string $r \in R$, it satisfies the following property:

• Completeness If $T = \{(x_{\ell}, y_{\ell})\}_{\ell}$ with $x_{\ell} \sim \mathcal{D}$, and there exists a subset $T' \subset T$, with $|T'| \geq \frac{2}{3}B$, such that $y_{\ell} = f^{\alpha}(x_{\ell})$ for all $(x_{\ell}, y_{\ell}) \in T'$, then, for any $\epsilon \geq \frac{1}{3}$ and $\delta' \geq 0$, the following condition holds with probability $1 - \delta'$:

$$\exists r \text{ s.t. } \mathcal{A}_B(T, \epsilon, r) = \alpha,$$

where the probability is taken over all possible datasets $T = \{(x_{\ell}, y_{\ell})\}_{\ell}$ with $x_{\ell} \sim \mathcal{D}$, in which at least $\frac{2}{3}$ of the points are labeled by f^{α} .

In addition to being a proper PAC learner: if the samples in T come from the distribution \mathcal{D} , i.e. $x_{\ell} \sim \mathcal{D}$, and there exists an $\alpha \in \{0,1\}^m$ such that $T = \{(x_{\ell},y_{\ell})\}_{\ell=1}^B$, with $x_{\ell} \sim \mathcal{D}$ and $y_{\ell} = f^{\alpha}(x_{\ell}) \in \{0,1\}$ then with probability $1 - \delta$ outputs:

$$\mathcal{A}_B(T^\alpha, \epsilon, r) = \tilde{\alpha}, \quad \tilde{\alpha} \in \{0, 1\}^m, \tag{5}$$

with the condition $\mathbb{E}_{x \sim \mathcal{D}} |f^{\alpha}(x) - f^{\tilde{\alpha}}(x)| \leq \epsilon$.

We say that \mathcal{A}_B solves the identification task for a concept class \mathcal{F} under the input distribution \mathcal{D} if the algorithm works for any value of $\epsilon, \delta \geq 0$. The success probability $1 - \delta$ is over the training sets T where the input points are sampled from the distribution \mathcal{D} and the internal randomization of the algorithm. The required minimum size B of the input set T is assumed to scale as $\operatorname{poly}(n,1/\epsilon,1/\delta)$, while the running time of the algorithm scales as $\operatorname{poly}(B, n, \delta')$.

As a preliminary step toward proving hardness in the non-verifiable case, we will first establish the hardness of the approximate-verifiable identification task, as defined in the following.

Definition 7 (Identification task - approximate-verifiable case). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class. An approximate-verifiable identification algorithm is a (randomized) algorithm \mathcal{A}_B such that when given as input a set $T = \{x_{\ell}, y_{\ell}\}_{\ell=1}^B$ of at least B pairs $(x,y) \in \{0,1\}^n \times \{0,1\}$, an error parameter $\epsilon > 0$ and a random string $r \in R$, it satisfies the two following properties:

• Soundness If $A_B(T, \epsilon, r) = \alpha$ then there exists a subset $T' \subset T$, with $|T'| \geq \frac{2}{3}B$ such that $y_\ell = f^\alpha(x_\ell)$ for all the $(x_\ell, y_\ell) \in T'$. In case there does not exist a subset $T' \subset T$ of $\frac{2}{3}B$ points consistent with any one of the concepts, then A_B outputs "invalid dataset".

• Completeness If $T = \{(x_{\ell}, y_{\ell})\}_{\ell}$ with $x_{\ell} \sim \mathcal{D}$, and there exists a subset $T' \subset T$, with $|T'| \geq \frac{2}{3}B$, such that $y_{\ell} = f^{\alpha}(x_{\ell})$ for all $(x_{\ell}, y_{\ell}) \in T'$, then, for any $\epsilon \geq \frac{1}{3}$ and $\delta' \geq 0$, the following condition holds with probability $1 - \delta'$:

$$\exists r \text{ s.t. } \mathcal{A}_B(T, \epsilon, r) = \alpha,$$

where the probability is taken over all possible datasets $T = \{(x_{\ell}, y_{\ell})\}_{\ell}$ with $x_{\ell} \sim \mathcal{D}$, in which at least $\frac{2}{3}$ of the points are labeled by f^{α} .

In addition to being a proper PAC learner: if the samples in T come from the distribution \mathcal{D} , i.e. $x_{\ell} \sim \mathcal{D}$, and there exists an $\alpha \in \{0,1\}^m$ such that $T = \{(x_{\ell},y_{\ell})\}_{\ell=1}^B$, with $x_{\ell} \sim \mathcal{D}$ and $y_{\ell} = f^{\alpha}(x_{\ell}) \in \{0,1\}$ then with probability $1 - \delta$ outputs:

$$\mathcal{A}_B(T^\alpha, \epsilon, r) = \tilde{\alpha}, \quad \tilde{\alpha} \in \{0, 1\}^m, \tag{6}$$

with the condition $\mathbb{E}_{x \sim \mathcal{D}} |f^{\alpha}(x) - f^{\tilde{\alpha}}(x)| \leq \epsilon$.

We say that \mathcal{A}_B solves the identification task for a concept class \mathcal{F} under the input distribution \mathcal{D} if the algorithm works for any value of $\epsilon, \delta \geq 0$. The success probability $1 - \delta$ is over the training sets T where the input points are sampled from the distribution \mathcal{D} and the internal randomization of the algorithm. The required minimum size B of the input set T is assumed to scale as $\operatorname{poly}(n,1/\epsilon,1/\delta)$, while the running time of the algorithm scales as $\operatorname{poly}(B, n, \delta')$.

2.4 Tools from complexity theory

In this section, we introduce the complexity theory tools that we use to prove our results in this paper. Both of our hardness results in Section 5 and 6 are based on the assumption that the class BQP is not contained in (a low level of) the polynomial hierarchy (PH). Specifically, given a BQP language \mathcal{L} , we define a function f which "correctly decides" (see Def. 9) \mathcal{L} .

Definition 8 (BQP function f). We refer to a function $f : \{0,1\}^n \to \{0,1\}$ as a BQP function if there exists a language $\mathcal{L} \in \mathsf{BQP}$ such that f is its characteristic function. In particular:

$$f(x) = \begin{cases} f(x) = 0 \text{ if } x \notin \mathcal{L} \\ f(x) = 1 \text{ if } x \in \mathcal{L} \end{cases}$$
 (7)

Where, as previously said, we use f to refer to a uniform family of functions, one for each input size. As BQP it is not know to have complete problems, in our hardness results it will also be convenient to consider the class PromiseBQP which instead allows for complete problems.

Definition 9 (PromiseBQP function f). We refer to a function $f: \{0,1\}^n \to \{0,1\}$ as a PromiseBQP function if there exists a language with a promise $\mathcal{L}_P = (\mathcal{L}_{yes}, \mathcal{L}_{no}) \subset \{0,1\}^n$ in PromiseBQP such that f is the characteristic function of \mathcal{L}_P . In particular:

$$f(x) = \begin{cases} f(x) = 0 \text{ if } x \notin \mathcal{L}_{yes} \\ f(x) = 1 \text{ if } x \in \mathcal{L}_{yes} \end{cases}$$
 (8)

Additionally, we say that f is a BQP-complete⁶ function if the associated language \mathcal{L} is complete for PromiseBQP.

Notice that for a PromiseBQP function f, f(x) = 0 both if x belongs to the NO subset of the promise, i.e. $x \in \mathcal{L}_{no}$, and if x does not belong to the promise subset at all, i.e. $x \notin \mathcal{L}_{yes} \cup \mathcal{L}_{no}$. In general the promise subset $\mathcal{L}_{yes} \cup \mathcal{L}_{no}$ contains an exponentially small fraction of all the possible input $x \in \{0,1\}^n$ and consequently f = 0 for the majority of $x \in \{0,1\}^n$. This means that correctly evaluating f on average over the input, for instance under the uniform distribution over all $x \in \{0,1\}^n$, becomes trivial unless the error is

 $^{^6\}mathrm{It}$ is known that BQP do not have complete problems. Here, we will abuse notation and actually refer to PromiseBQP problems.

exponentially small. We refer to [GD23] for a more detailed discussion on this. We will prove our hardness results by showing that a classical machine with access to an NP oracle can achieve achieve average-case correctness for evaluating a BQP or PromiseBQP function f with respect a given input distribution. In case of BQP functions, we will then assume that there exists a $\mathcal{L} \in \mathsf{BQP}$ which is not in heuristically evaluable by classical machines in the PH with respect to the uniform distribution over all the input $x \in \{0,1\}^n$. For PromiseBQP functions, to ensure the task remains non-trivial, we will state our hardness results with respect to the input distribution $\mathsf{Unif}_{\mathcal{L}_P}$ which is defined as the uniform distribution over a subset of the promise set of input $x \in \mathcal{L}_{\mathsf{yes}} \cup \mathcal{L}_{\mathsf{no}}$. In our hardness results, we assume the existence of a PromiseBQP complete language \mathcal{L}_P such that there exists a distribution $\mathsf{Unif}_{\mathcal{L}_P}$ of hard-to-evaluate but classically samplable instances.

Definition 10 (Distribution $\mathsf{Unif}_{\mathcal{L}_P}$). Let $\mathcal{L}_P = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ be a PromiseBQP complete language. We call $\mathsf{Unif}_{\mathcal{L}_P}$, if it exists, the uniform distribution over a subset $\Pi \subset \mathcal{L}_{yes} \cup \mathcal{L}_{no}$ such that the following holds:

• Unif_{\mathcal{L}_P} is efficiently classically samplable.

For our results in Theorem 2 and Theorem 8 and 3 we will then assume the existence of a PromiseBQP language \mathcal{L}_P for which there exists a $\mathsf{Unif}_{\mathcal{L}_P}$ such that $(\mathcal{L}_P, \mathsf{Unif}_{\mathcal{L}_P})$ is not contained in HeurBPP with access to NP oracles in the PH.

We next provide a precise definition for the complexity class HeurBPP (for more details see [BT⁺06]). To define heuristic complexity classes, we first need to consider the so-called distributional problems (which should not be confused with learning distributions in the context of unsupervised learning).

Definition 11 (Distributional problem). A distributional problem $(\mathcal{L}, \mathcal{D})$ consists of a language $L \subseteq \{0, 1\}^*$ and a family of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n \in \mathbb{N}}$ such that $\operatorname{supp}(\mathcal{D}_n) \subseteq \{0, 1\}^n$.

We are now ready to provide a precise definition of the class HeurBPP.

Definition 12 (Class HeurBPP). A distributional problem $(\mathcal{L}, \mathcal{D})$ is in HeurBPP if there exists a polynomial-time classical algorithm such that for all n and $\epsilon \geq 0$:

$$\Pr_{x \sim \mathcal{D}} \left[\Pr[\mathcal{A}(x, 0^{1/\epsilon}) = L(x) \ge \frac{2}{3} \right] \ge 1 - \epsilon, \tag{9}$$

in the above the inner probability is taken over the internal randomization of A_B .

2.5 Evidence that $BQP \not\subset PH$

In this section, we provide a brief discussion on the primary assumptions underlying our work, which are derivatives and versions of BQP $\not\subset$ PH. The first main result in this direction is given in [Aar10], where an oracle separation between the relational version of the two classes was proven. Specifically, Aaronson proved that the relation problem Fourier Fishing, a variant of the Forrelation problem, exhibits an oracular separation FBQP^A $\not\subset$ FBPP^{PH^A}. In the same paper Aaronson motivated the study of oracle separation for BQP and PH as lower bounds in a concrete computational model, claiming them as a natural starting point for showing evidence of BQP $\not\subset$ PH. Almost ten years later, in the remarkable work [RT22], the authors managed to prove an oracle separation for the decision versions of the classes. Namely they proved the existence of an oracle A such that BQP^A $\not\subset$ PH^A. Although an unconditional proof of separation between BQP and PH is not likely to appear anytime soon⁷, the assumption BQP $\not\subset$ PH is generally considered reasonable.

Most of the results in this paper furthermore rely on the assumption that BQP languages cannot be decided correctly on average by algorithms in HeurBPP, even when given access to oracles for the polynomial hierarchy. This is a stronger assumption than simply BQP $\not\subset$ PH, but we believe not unreasonable. Some evidence in this direction comes from sampling problems, where the proof of the best-known separation of SampP and SampBQP [AA11, MGD24a] is identical for the heuristic case and produces the same collapse.

⁷Note that any proof of BQP $\not\subset$ PH would immediately imply the hard-to-prove claim that BQP \neq BPP.

To be more exact, suppose that SampBQP can be decided in some heuristic level of the polynomial hierarchy, follow the proof of [AA11] for a heuristic-equivalent theorem until we get to the definition of (Heur)GPE $_{\pm}$, at which point we note that the preexisting definition of GPE $_{\pm}$ is already heuristic, so our HeurGPE $_{\pm}$ is equal to GPE $_{\pm}$. Therefore, assuming that SampBQP is in some heuristic level of the sampling analogue of the polynomial hierarchy would still imply the collapse of the standard polynomial hierarchy to some level (standard assumptions from quantum supremacy arguments notwithstanding).

From these arguments, however, no analogous claims regarding the decision (and distributional) variants BQP and Heur PH can be proven using known techniques. For this reason, we stand by the following careful claim: there is no reason to believe that BQP is in Heur PH (relative to relevant distributions).

As a final remark, we do highlight that our results do not require considering the whole PH. Specifically it is important to note that for the hardness of random generatability of quantum functions in Section 4 we only need to assume that BQP is not in the second level of the PH, while the assumption extends up to the fourth level for the hardness of the identification problem in Section 6.

3 Related works

The study of different types of learning separations in quantum machine learning was initiated in [GD23]. distinguishing between advantages in the identification step and those arising solely from the evaluation of the target function. For BQP-complete functions, the authors showed that learning separations straightforwardly follow under the assumption that there exists an input distribution \mathcal{D} and a BQP language L such that $(L,\mathcal{D}) \not\subseteq \mathsf{HeurP/poly}$. This separation relied on requiring the learning algorithm to evaluate the learned function on new inputs, thus considering the evaluation step only. The paper left open the question of whether a stronger result could be established by proving that even identifying the correct target function is classically hard for BQP-complete concepts, which we address in this paper. In a follow-up work by some of the authors [MGD24b], stronger learning separations were established for BQP functions under the widely studied assumption that $BQP \not\subseteq P/poly$ (exactly, not with respect to heuristic conditions as in [GD23]), by reverting to standard PAC learning where the learning process must be successful for all input distributions (as opposed to settings where the distribution is fixed). Additionally, the authors proposed a quantum algorithm capable of correctly identifying and evaluating the target concept in the nontrivial case of a superpolynomially large concept class, as opposed to [GD23] where only polynomially large classes were studied. However, even in this case, the learning separation was demonstrated only for the evaluation step. Learning separations for superpolynomially large concept classes were also presented in [YIM23], though they are based on assumptions about heuristic classes similar to [GD23] and were not focused on finding physically motivated problems, which was instead the goal in [MGD24b]. In Section 7.1, we show that our result directly applies to the physically relevant learning problem considered in [MGD24b].

Results regarding the hardness of the identification step are already known for certain cryptographic functions, such as modular exponentiation [GD23] and the discrete cube root [JGM+24]. In these cases, the hardness of identification was established by leveraging two key properties of these cryptographic functions: random generatability and the existence of a concise, efficiently evaluable description of the functions. The latter property implies a mapping between two representations of the same set of functions: one that is computationally intractable and another that is classically efficient to compute. If identifying the target concept in the second representation were classically efficient, it would also allow the evaluation of the target function in its hard-to-compute representation. Specifically, thanks to the random generatability property, a classical algorithm could easily generate labeled data and solve the identification problem using the hypothesis class corresponding to the easy-to-compute representation of the function. Once the correct concept is identified, the algorithm could then evaluate it, thereby violating the cryptographic assumptions of the considered set of functions. For the BQP-complete functions analyzed in this paper, our findings on the hardness of random generatability already rule out the possibility of employing similar proof techniques to establish the hardness of identification, which, in fact, we prove using a different strategy and different complexity assumptions.

4 Hardness results for random generatability of quantum functions

In this section, we address the task of random generatability defined in the section before and prove its classical hardness for BQP functions as defined in Def. 9. As explained in Section 2.1, random generatability for a family of functions f implies the existence of a uniform (randomized) algorithm such that on input a random string of length n outputs a sample of $(x, f_n(x))$, with x sampled from a given target distribution⁸. Notice that this is at least not harder than requiring the ability to evaluate f_n on inputs x given a polynomial time Turing Machine. Moreover, there are many examples of likely hard-to-compute⁹ functions which are however random generatable (under the uniform distribution over the inputs). Examples include any hard function with an efficiently computable inverse, such as the discrete logarithm (examples can be constructed by generating the label first, see [KV94a, GD23] for more details), as well as the graph isomorphism problem, which remains random generatable even though its efficient evaluation is not known¹⁰, even for quantum computers [AS06]. In general, establishing whether a family of functions f is random generatable is itself a non-trivial problem.

In this section we prove hardness results for the task of random generability of PromiseBQP complete functions, defined as in Def. 9. In particular we show that the task of exact random generatability cannot be efficiently done classically unless PromiseBQP is contained in the second level of the PH. For the case of approximate random generability, we show classical hardness unless PromiseBQP can be heuristically decided by an efficient classical machine with a NP oracle. Classical hardness for exact random generatability of quantum functions can be proved also based on a different complexity-theoretic assumption, i.e. $BPP/samp^{BQP} \not\subseteq BPP$ (we refer to Appendix A and the works in [MGD24a, HBM^+21] for the precise definition of the class $BPP/samp^{BQP}$, but roughly speaking, this class requires the sample generator to be a polynomial-time quantum computer). For completeness, we also prove this latter result in the Appendix B.

4.1 Hardness of random generatability based on the assumption BQP $\not\subset$ PH

We first show that exact random generatability implies that BQP is contained within the second level of the PH.

Theorem 1 (Exact Random generatability implies PromiseBQP \subseteq P^{NP}). Let $f = \{f_n\}_{n \in \mathbb{N}}$ be a family of PromiseBQP complete functions as in Def. 9. If there exists a classical randomized poly-time uniform algorithm that generates samples $(x, f_n(x))$ correctly with probability 1 as in Def.1, with $x \sim \text{Unif}(\{0, 1\}^n)$, then P^{NP} contains PromiseBQP.

Proof sketch. The whole proof can be found in the Appendix B.2.1, here we give the main idea. Suppose there exists an algorithm \mathcal{A} which satisfies Def. 1 for a PromiseBQP complete family of function f and for the uniform distribution over the inputs $x \in \{0,1\}^n$. For a fixed function $f_n : \{0,1\}^n \to \{0,1\}$, the algorithm \mathcal{A} maps a random string $r \in \{0,1\}^{\text{poly}(n)}$ to a tuple of $(x_r, f_n(x_r))$, i.e. $\mathcal{A}(f_n, .) : \{0,1\}^{\text{poly}(n)} \to \{0,1\}^n \times \{0,1\}$. Then, in order to prove Theorem 1 we construct an algorithm \mathcal{A}' which can decide the BQP-complete language associated to f_n by using \mathcal{A} and an NP oracle. Such algorithm \mathcal{A}' will essentially invert \mathcal{A} on a given input $x_{\tilde{r}}$ to find a corresponding valid random string \tilde{r} and then computes $f_n(x_{\tilde{r}})$ using $\mathcal{A}(f_n, \tilde{r})$. Specifically, by using the NP oracle, \mathcal{A}' can find the random string \tilde{r} associated to $x_{\tilde{r}}$ for which $\mathcal{A}(f_n, \tilde{r}) = (x_{\tilde{r}}, f_n(x_{\tilde{r}}))$ in polynomial time. Importantly, finding the string \tilde{r} can be achieved using an NP oracle since \mathcal{A} operates in classical polynomial time, and thus it can efficiently verify the correct string \tilde{r} . This concludes the proof as,

 $^{^8}$ If f was a sample of two bits from a quantum circuit (as opposed to a single output bit), then multiplicatively close random generability would collapse the polynomial hierarchy via post selection: If two output bits are accurately supplied by a classical poly-time machine, then $PostBQP \subseteq MA$, as PostBQP = PP and $PH \subseteq P^{PP}$ this would imply PH = AM, a collapse to the second leve of the polynomial hierarchy.

⁹In the paper we say that a function is "hard-to-compute" if there is not polynomial-sized classical circuit which evaluates it correctly on every input.

¹⁰Although a very efficient quasi-polynomial algorithm for it has been recently proposed in [Bab16].

by Def. 9, f correctly decides a PromiseBQP complete language and \mathcal{A}'_B can evaluate any f_n on every $x_{\tilde{r}}$ by just running $\mathcal{A}(x_{\tilde{r}}, \tilde{r})$.

In Theorem 1 the algorithm \mathcal{A} must output samples $(x, f_n(x))$ with x which exactly comes from the uniform distribution and the corresponding label $f_n(x)$ is always correct. A natural question to consider is whether our hardness result still holds if these assumptions on \mathcal{A} are relaxed. Specifically we allow the algorithm \mathcal{A} to make mistakes both on the distribution followed by the outputted x and we also allow errors on some of the outputted $(x, f_n(x))$. In Theorem 2 we specifically address such cases, proving that approximate random generatability in Def. 2 is still classically hard unless PromiseBQP is contained in HeurBPP^{NP} for the previously introduced uniform distribution Unif $_{\mathcal{L}_P}$ over a subset of the inputs in the promise.

Theorem 2 (Approximate Random generatability implies (PromiseBQP, Unif_P) \in HeurBPP^{NP}). Let $f = \{f_n\}_{n\in\mathbb{N}}$ be a family of PromiseBQP functions which is the characteristic function of a language $\mathcal{L}_P \in$ PromiseBQP complete as in Def. 9, and let Unif_{\mathcal{L}_P} be a distribution defined as in Def. 10. If there exists a polynomial time algorithm \mathcal{A} which satisfies Def.2 for the input distribution Unif_{\mathcal{L}_P}, then $(\mathcal{L}_P, \text{Unif}_{\mathcal{L}_P}) \in$ HeurBPP^{NP}.

Proof sketch. The full proof can be found in Appendix B.2.2, here we outline the main idea. We present an algorithm $\mathcal{A}' \in \mathsf{HeurBPP^{NP}}$ which can heuristically decide the language \mathcal{L}_P with respect to the uniform distribution over the input in the promise, i.e. it satisfies Eq. (9) for the input distribution $\mathcal{D} = \mathsf{Unif}_{\mathcal{L}_P}$. Let $f = \{f_n\}_n$ be the family of functions associated to the PromiseBQP complete language as in Def. 9. The algorithm \mathcal{A}' follows directly from the one described in the proof of Theorem 1. For a fixed function f_n and error parameter ϵ , the algorithm $\mathcal{A}(f_n, 0^{1/\epsilon}, .)$ in Def. 2 still maps random strings $r \in \{0, 1\}^{\operatorname{poly}(n)}$ to tuples $(x_r, f_n(x_r)) \in \{0, 1\}^n \times \{0, 1\}$. Then, on an input $x_{\tilde{r}}$, \mathcal{A}' still inverts the algorithm \mathcal{A} in order to obtain a random string \tilde{r} such that $\mathcal{A}(f_n, 0^{1/\epsilon}, \tilde{r}) = (x_{\tilde{r}}, f_n(x_{\tilde{r}}))$. This time however, it samples multiple such random strings uniformly at random (this can be done in polynomial time using the result from [BGP00]). By doing so, we can guarantee that taking the average of the corresponding $f_n(x_{\tilde{r}_i})$, obtained from $\mathcal{A}(f_n, 0^{1/\epsilon}, \tilde{r}_i) = (x_{\tilde{r}_i}, f_n(x_{\tilde{r}_i}))$ for different \tilde{r}_i , will correctly classify the input point $x_{\tilde{r}}$ with high probability. More precisely, as stated in Def. 2, the algorithm \mathcal{A} outputs samples $(x, f_n(x))$ which follow a distribution \mathcal{L}_{ϵ} ϵ -close in total variation with the exactly labeled, uniformly sampled over $x \in \{0,1\}^n$ one. It follows (see full proof in the Appendix B.2.2) that the maximum fraction of point x which \mathcal{A}' may misclassify is upper bounded by a linear function of ϵ .

We notice here that an analogous proof would have also shown that if a BQP function associated to $\mathcal{L} \in \mathsf{BQP}$ is approximately random generatable for the uniform distribution over all the $x \in \{0,1\}^n$, then $(\mathcal{L},\mathsf{Unif}(\{0,1\}^n)) \in \mathsf{HeurBPP}^\mathsf{NP}$. This will then also imply hardness results for random generatability of f assuming the existence of a BQP language not heuristically decidable in the second level of the PH with respect to the uniform distribution over the inputs.

Although this is tangential to our present discussion we note the following Corollary of the last theorem, proving that a certain class of quantum generative models called expectation value samplers (introduced in [RAG21], and proven universal in [BGV⁺24] and generalized in [SKS⁺24]) is classically intractable.

Corollary 1. The expectation value sampling (EVS) generative model, as defined in $[BGV^+24]$ is classically intractable.

Proof. To not detract from the main topic we just provide a quick proof sketch. By intractable we mean that there cannot exist a classical algorithm which takes on input an arbitrary polynomial sized EVS M specified by its parametrized circuit (see $[BGV^+24]$) and outputs an ϵ approximation of the distribution outputted by M. To prove this we note that for any BQP function $f:\{0,1\}^n \to \{0,1\}$ we can construct a poly-sized EVS which produces samples of the form (x, f(x)), where x is an n-bit bitsting sampled (approximately) uniformly (or according to any desired classically samplable distribution). There are many ways to realize this: the easiest is to coherently read out the bits of input random continuous-valued variable using phase estimation (for this, this variable x is uploaded using prefactors 2^k for each bit position k), forwards this to part of the output, and in a parallel register computes f(x) and outputs that as well. This is a valid

5 Hardness results for the verifiable identification problem

We now address the second problem studied in this paper, specifically the hardness of the identification task defined in Def. 5 and Def. 6. We recall that by Proposition 2 the identification task in Def. 6 is closely related to the problem of proper PAC learning, namely the case where the hypothesis class of a learning algorithm coincides with the concept class of the target functions.

In this section and the next, we outline sufficient conditions on the concept class and the learning algorithm to derive classical hardness results. In particular, we show hardness results for the identification task outlined in Def. 5, where the learning algorithm can also detect invalid dataset, i.e., cases where the dataset is not consistent with any single concept. This case relates to the consistency learning model found in the literature and will serve as a toy example for presenting our broader result concerning the identification task defined in Def. 6, presented in Section 6. There, we prove a stronger hardness result showing that the identification task remains classically intractable even when the learning algorithm cannot directly distinguish between valid and invalid datasets.

5.1 Hardness result for the verifiable identification problem with BQP functions

We now state the first hardness result for the identification task of Def. 5 in case of verifiable learning algorithms.

Theorem 3 (Identification hardness for the verifiable case - promise). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m, m = \text{poly}(n)\}$ be a concept class such that there exists a $f^{\alpha} \in \mathcal{F}$ which is the characteristic function of a language $\mathcal{L}_P \in \mathsf{Promise} - \mathsf{BQP}$ as in Def. 9, and let $\mathsf{Unif}_{\mathcal{L}_P}$ be a distribution over a subset of the promise as in Def. 10. If there exists a verifiable identification algorithm \mathcal{A}_B for the input distribution $\mathsf{Unif}_{\mathcal{L}_P}$ as given in Def. 5, then $(\mathcal{L}_P, \mathsf{Unif}_{\mathcal{L}_P}) \in \mathsf{HeurBPP}^\mathsf{NP}$.

Proof sketch. The full proof can be found in the Appendix C.2, here we outline the proof sketch. The core idea is to show that if there exists an algorithm A_B that satisfies Def. 5 for the concept class $\mathcal F$ in the theorem, then there exists a polynomial time algorithm \mathcal{A}' which, using \mathcal{A}_B and an NP oracle, takes as input any $\alpha \in \{0,1\}^m$ (which uniquely specifies a concept¹¹) and outputs a dataset of B-many points labeled by a concept $f^{\tilde{\alpha}}$ in agreement with f^{α} under the PAC condition of Eq. 3. We recall here that, given an error parameter ϵ and a random string r, the algorithm A_B takes as input any set T of B pairs $(x,y) \in \{0,1\}^n \times \{0,1\}$ and, if and only if the set T is consistent with one of the target concepts α , outputs with high probability a $\tilde{\alpha} \in \{0,1\}^m$ close in PAC condition to α . Note that the crucial "if and only if" condition stems directly from the ability of A_B to detect invalid datasets, as described in Def. 5. We can then leverage this to construct the algorithm \mathcal{A}' which correctly classifies the \mathcal{L} language associated to f^{α} . Specifically, on any input $x \in \{0,1\}^n$, the algorithm \mathcal{A}' first inverts \mathcal{A}_B on the target α in order to obtain a dataset $T = \{(x_{\ell}, y_{\ell})\}_{\ell=1}^{B}$ of B points such that $\mathcal{A}_{B}(T, \epsilon, .) = \alpha$. It then labels the input x based on consistency with the training set T generated in the previous step. By selecting an appropriate number of points B, it is possible to bound the probability that the dataset T is consistent with a concept $f^{\tilde{\alpha}}$ heuristically close to the target f^{α} , thereby bounding the probability that the label assigned to x corresponds to $f^{\alpha}(x)$.

In Theorem 3 we based our hardness result on the assumption that PromiseBQP complete problem are not decidable in HeurBPP^{NP} with respect to the input distribution $\mathsf{Unif}_{\mathcal{L}_P}$ of Def. 10. In Appendix C.2 we notice that a similar proof strategy can be used to prove that if a verifiable learning algorithm exists for a concept

¹¹We assume that each α provides an unambiguous specification of the concept f^{α} (possibly as a quantum circuit, i.e. that a quantum circuit can, given α , evaluate f^{α} in polynomial time).

class containing a BQP function associated to a BQP language \mathcal{L} as in Def. 8, then $(\mathcal{L}, \mathsf{Unif}(\{0,1\}^n)) \in \mathsf{HeurBPP^{NP}}$. Therefore, hardness of the verifiable identification task can also be proven for BQP functions under the assumption that there exists a BQP language which is not in $\mathsf{HeurBPP^{NP}}$ with respect to the uniform distribution over all the input $x \in \{0,1\}^n$. As the relation between these two different assumptions is not clear, we state here the result also based on the assumption on BQP languages.

Theorem 4 (Identification hardness for the verifiable case - uniform). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m, m = \text{poly}(n)\}$ be a concept class such that there exists a $f^{\alpha} \in \mathcal{F}$ which is the characteristic function of a language $\mathcal{L} \in \mathsf{BQP}$ as in Def. 8, and let $\mathsf{Unif}(\{0,1\}^n)$ be the uniform distribution over all the input $x \in \{0,1\}^n$. If there exists a verifiable identification algorithm \mathcal{A}_B for the input distribution Unif as given in Def. 5, then $(\mathcal{L}, \mathsf{Unif}) \in \mathsf{HeurBPP}^\mathsf{NP}$.

The proof of Theorem 4 closely follows the argument used for Theorem 3, with a minor modification to account for the different input distribution considered, as discussed in Appendix C.2.

As argued before, the ability of \mathcal{A}_B to detect invalid dataset is crucial for the construction of \mathcal{A}_B' in the proof of Theorem 3. Although it is a strong assumption (which we will remove in Section 6 for certain families of concept classes) we can still argue that the verifiable case of Def. 5 is of interest. Firstly, Def. 5 perfectly captures the framework of learning in the consistency model present in the literature [Bal15, Sch08, Moh18, KV94b]. In [KV94b], a concept f^{α} is defined to be consistent with a dataset $T^{\alpha} = \{(x_{\ell}, y_{\ell})\}_{\ell=1}^{B}$ if $y_{\ell} = f^{\alpha}(x_{\ell})$ for every $\ell = 1, ..., B$. Based on that, we give the following definition of learning in the consistency model framework, which includes the ability of the learning model to distinguish invalid datasets. As we explain below, this assumption aligns with the general case found in the literature [KV94b, Sch90, Moh18], where only efficient hypothesis classes are considered.

Definition 13 (Consistency model learning [Bal15, Sch08]). We say that an algorithm \mathcal{A} learns the class $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m, m = \text{poly}(n)\}$ in the consistency model if, given any set of labeled examples T, the algorithm produces a concept $f \in \mathcal{F}$ that is consistent with T, if such a concept exists, and outputs "there is no consistent concept" otherwise. The algorithm runs in polynomial time (in the size of T and the size n of the examples).

Given the definition of learning in the consistency model in Def. 13, it is clear that an algorithm \mathcal{A}_B solves the identification task in Def. 5 for a given concept class \mathcal{F} if and only if it learns \mathcal{F} in the consistency model.

We remark here another crucial point. The reason it is meaningful in our work to differentiate between the verifiable case in Def. 5 (associated to the consistency model learning 12) and the non-verifiable case in Def. 6 (associated to proper PAC learning) is that we are considering concepts (and hypothesis) that cannot be efficiently evaluated classically. In general, for efficiently evaluable concepts, it can be shown that if a concept class is (proper) PAC learnable, then there always exists a learning algorithm which outputs a hypothesis consistent with the training data, see Theorem 5 in [Sch90]. Furthermore, since the hypotheses can be efficiently evaluated, it is straightforward to verify whether the outputted concept is consistent with the given dataset or if the input dataset is invalid, as defined in Def. 5. Crucially, this is exactly the case when considering a quantum learner. In fact, another compelling reason to consider learning algorithms within the consistency model framework is that quantum learners can verify whether a dataset is valid for a given α . Specifically, given an input dataset T, the quantum learning algorithm outputs the guessed α and then can check whether every point in the dataset are correctly labeled by f^{α} , similar to the process for efficiently evaluable hypotheses. Assuming f^{α} is quantum evaluable, and the dataset is polynomial in size, the verification procedure runs in polynomial time. A following question is then whether in order to verify a dataset we necessarily need a quantum computer for the BQP functions defined in Def. 9. We address this question in the following proposition, which asserts that for a singleton concept class¹³, determining whether

¹²Although the definition of learning in the consistency model matches our definition of a verifiable learner, we prefer to use distinct terminology. This distinction emphasizes the difference from the non-verifiable case, which is our ultimate focus and cannot easily reduce to the consistency model for classically intractable concepts.

¹³A singleton concept class is a concept class that consists of only one concept.

a dataset is valid is possible if there exists an efficient non-uniform algorithm which correctly evaluates the concept. In other words, the concept can be evaluated in the class P/poly (see Def. 19 in Appendix A for a definition of P/poly).

Proposition 1 (Hardness of verification - singleton case). Let $\mathcal{F} = \{f : \{0,1\}^n \to \{0,1\}\}$ be a singleton concept class. If there exists an efficient algorithm \mathcal{A}_B such that for every set $T = \{(x_\ell, y_\ell)\}_{\ell=1}^B$, with B polynomial in $n, x_\ell \in \{0,1\}^n$ such that $x_i \neq x_j \ \forall i,j \in \{1,...,B\}$ and $y_\ell \in \{0,1\}$, satisfies the following:

• If $\exists (x_{\ell}, y_{\ell}) \in T$, $y_{\ell} \neq f^{\alpha}(x_{\ell})$, $\mathcal{A}_{B}(T)$ outputs "invalid dataset".

Then there exists a polynomial time non-uniform algorithm which computes f(x) for every x. In particular, there exists an algorithm in P/poly which computes f(x).

Proof. For every input size n, let us take as polynomial advice the B-1 samples $T^{B-1}=\{(x_\ell,f(x_\ell))\}_{\ell=1}^{B-1}$, for any sequence of different $x_\ell\in\{0,1\}^n$. Then, on any input $x\in\{0,1\}^n$, the algorithm in P/poly would run \mathcal{A}_B on the dataset $T_x=\{(x,0)\}\cup T^{B-1}$ and decide x based on the corresponding output of \mathcal{A}_B .

We specify that we require the inputs x_{ℓ} to be distinct in the training set T to strengthen the result in Proposition 1. Without this condition, the result would be trivial, as one could provide \mathcal{A}_B with B identical copies of (x,0) as input and correctly decide each x using a polynomial-time uniform algorithm by simply examining the corresponding output. Furthermore, in this paper, we are taking into consideration the case where training points are drawn from the uniform distribution over a set of different points, as outlined in the definition of the identification task in Def. 5 and 7. In case of exponential-sized concept classes containing a PromiseBQP function which uniquely labels a set of polynomial number of points, the verifiability property of the identification algorithm can be used to prove that PromiseBQP is contained in the class P/poly.

Theorem 5. Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m, m = \operatorname{poly}(n)\}\$ be a concept class such that there exists at least one function $f^{\alpha} \in \mathcal{F}$ that decides a language $\mathcal{L} \in \operatorname{PromiseBQP}$, and for which there exists a polynomial-sized subset $S \subset \{0,1\}^n$ such that f^{α} is uniquely determined by its labels on S. If there exists a verifiable identification algorithm \mathcal{A}_B as given in Def. 5, then $\operatorname{PromiseBQP} \subseteq \operatorname{P/poly}$.

Proof. For every input size n, let us take as polynomial advice the samples from the subset S which uniquely determine f^{α} and the corresponding labels, i.e. $T^{S} = \{(x_{\ell}, f(x_{\ell}))\}_{\ell=1}^{|S|}$, such that $y_{\ell} = f^{\alpha}(x_{\ell})$ for every $x_{\ell} \in S$. We then consider exactly that dataset as advice and we label any new input $x \in \{0,1\}^n$ selecting the $y \in \{0,1\}$ which keeps the dataset valid. More precisely, on any input $x \in \{0,1\}^n$, the algorithm in P/poly would run A_B on the dataset $T_x = \{(x,0)\} \cup T^S$ and label x so that the algorithm A_B accepts the dataset.

From Proposition 1 and Theorem 5, it becomes evident that imposing the exact verification property, as defined in Def. 5, might be an overly strong assumption for a classical learning algorithm. In the following section, we will derive hardness results for the identification task with a weaker assumption on the learning algorithm, moving us closer to the proper PAC framework.

6 Hardness result for the identification problem - Non-verifiable case

In the previous section, we derived hardness results for the identification task in Def. 5 assuming that the learning algorithm \mathcal{A} can distinguish if a dataset is fully valid or not. In this section we remove the verifiability condition and show that hardness for the non-verifiable identification task in Def. 6 still persists if we put additional assumptions on the learning algorithm and the structure of the concept class. In particular, we call an identification algorithm approximately-correct if it satisfies the additional assumptions outlined in

Def. 14. The key difficulty in establishing classical hardness for the non-verifiable identification task solved by an approximate-correct identification algorithm in Def. 14, as opposed to the verifiable case in Def. 5, lies in the lack of guarantees when inverting the algorithm $A(.,\epsilon,r):\{0,1\}^{B(n+1)}\to\{0,1\}^m$ with fixed ϵ and r. Specifically, determining the preimage of a given $\alpha \in \{0,1\}^m$ produces a dataset $T = \{(x_\ell, y_\ell)\}_{\ell=1}^B$, where the number of points x_{ℓ} that are correctly labeled by f^{α} is not known. This is because we make no assumptions about the behavior of A_B when given an invalid dataset as input, i.e. a dataset that is not entirely consistent with one concept. Nonetheless, we will demonstrate a hardness result for the non-verifiable identification task under the assumption that it is solvable by an approximate-correct algorithm, for concept classes that either include c-distinct concepts with $c \geq \frac{1}{3}$ or satisfy the average-case-smoothness property, which we will precisely define after stating our main hardness result.

First, we provide the formal definition of approximate-correct algorithm:

Definition 14 (Approximate-correct identification algorithm). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \mathbb{C}\}$ $\{0,1\}^m\}$ be a concept class. An approximate-correct identification algorithm is a (randomized) algorithm \mathcal{A}_B such that when given as input a set $T = \{(x_\ell, y_\ell)\}_{\ell=1}^B$ of at least B pairs $(x, y) \in \{0, 1\}^n \times \{0, 1\}$, an error parameter $\epsilon > 0$ and a random string $r \in R$ and it satisfies:

1. If for any α all the $(x_{\ell}, y_{\ell}) \in T$ are such that $y_{\ell} \neq f^{\alpha}(x_{\ell})$ then there exists an ϵ_1 such that for all $\epsilon \leq \epsilon_1$ and all $r \in R$:

$$\mathcal{A}_B(T, \epsilon_1, r) \neq \alpha. \tag{10}$$

Therefore, for no dataset the algorithm can output a totally wrong α , i.e. an α inconsistent with all the points in the dataset.

2. If $T = \{(x_{\ell}, y_{\ell})\}_{\ell=1}^{B}$ is composed of different inputs x_{ℓ} and if there exists an α such that the corresponding labels follow $y_{\ell} = f^{\alpha}(x_{\ell})$ for all $(x_{\ell}, y_{\ell}) \in T$, then there exists a threshold ϵ_2 such that for any $\epsilon \leq \epsilon_2$ there exists at least one $r \in R$ for which:

$$\mathcal{A}_B(T, \epsilon_2, r) = \alpha_2 \tag{11}$$

With the condition: $\mathbb{E}_{x \sim \text{Unif}(\{0,1\}^n)} |f^{\alpha}(x) - f^{\alpha_2}(x)| \leq \frac{1}{3}$. Therefore, if the dataset is fully consistent with one of the concept α , then there is at least one random string for which the identification algorithm will output a $\tilde{\alpha}$ closer than $\frac{1}{3}$ in PAC condition to the true labelling α .

3. It satisfies the completeness property of Def. 7. If $T = \{(x_{\ell}, y_{\ell})\}_{\ell}$ with $x_{\ell} \sim \mathcal{D}$, and there exists a subset $T' \subset T$, with $|T'| \geq \frac{2}{3}B$, such that $y_{\ell} = f^{\alpha}(x_{\ell})$ for all $(x_{\ell}, y_{\ell}) \in T'$, then, for any $\epsilon \geq \frac{1}{3}$ and $\delta' \geq 0$, the following condition holds with probability $1 - \delta'$:

$$\exists r \text{ s.t. } \mathcal{A}_B(T, \epsilon, r) = \alpha,$$

where the probability is taken over all possible datasets $T = \{(x_{\ell}, y_{\ell})\}_{\ell}$ with $x_{\ell} \sim \mathcal{D}$, in which at least $\frac{2}{3}$ of the points are labeled by f^{α} .

4. If $T = \{(x_\ell, y_\ell)\}_{\ell=1}^B$ is composed of inputs x_ℓ sampled from the distribution \mathcal{D} and if there exists an $\alpha \in \{0, 1\}^m$ such that $\forall (x_\ell, y_\ell) \in T$ it holds $y_\ell = f^\alpha(x_\ell)$, then with probability $1 - \delta$ the algorithm \mathcal{A}_B outputs:

$$\mathcal{A}_B(T^{\alpha}, \epsilon, r) = \tilde{\alpha}, \quad \tilde{\alpha} \in \{0, 1\}^m, \tag{12}$$

with the condition $\mathbb{E}_{x \sim \mathcal{D}}|f^{\alpha}(x) - f^{\tilde{\alpha}}(x)| \leq \epsilon$. The success probability $1 - \delta$ is over the training sets T and the internal randomization of the algorithm.

We say that A_B solves the identification task for a concept class $\mathcal F$ under the input distribution $\mathcal D$ if the algorithm works for any value of $\epsilon, \delta \geq 0$. The required minimum size B of the input set T is assumed to scale as $poly(n,1/\epsilon,1/\delta)$, while the running time of the algorithm scales as poly(B,n). Moreover, the ϵ_1 and ϵ_2 required values scale at most inverse polynomially with n.

We notice that while the first condition in Def. 14 simply requires that the learning algorithm never outputs an α that is entirely inconsistent with the dataset, which is a natural assumption, the second condition is somewhat stronger. Specifically, it demands that when the dataset is perfectly consistent with a particular concept, the algorithm must output that concept for at least one choice of random string. However, we rely heavily on this condition in our proofs of Theorem 9 and 10, and it is an open question if one could relax this further.

We are now ready to state our main result, which will be again based on the average-case hardness of evaluating BQP or PromiseBQP languages:

Theorem 6 (Hardness of the identification task - non verifiable case). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class containing at least a BQP (PromiseBQP) function f as in Def. 8 (as in Def. 9) associated to a language $\mathcal{L} \in \mathsf{BQP}$ ($\mathcal{L}_P \in \mathsf{PromiseBQP}$). Assume further that \mathcal{F} is either a c-distinct concept class with $c \geq 1/3$ or an average-case-smooth concept class. If the non-verifiable identification task for \mathcal{F} , as defined in Def. 6, is solvable by an approximate-correct identification algorithm \mathcal{A}_B (Def. 7), then it follows that $(\mathcal{L}, \mathsf{Unif}) \in \mathsf{HeurBPP}^{\mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}}$ (respectively, $(\mathcal{L}_P, \mathsf{Unif}) \in \mathsf{HeurBPP}^{\mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}}$).

Proof. The proof combines the intermediate result in Theorem 7 (or Theorem 8) with the result in Theorem 9 for the c-distinct concept classes or in Theorem 10 for average-case-smooth concept classes. Specifically, Theorems 9 and 10 show that if an approximate-correct algorithm exists for a concept class \mathcal{F} satisfying Def. 15 or Def. 18, then there exists an approximate-verifiable algorithm for \mathcal{F} running in BPP^{NP}. Therefore, as a consequence of the intermediate result in Theorem 7 (Theorem 8), it follows that if such approximate-verifiable algorithm would exist, then $(\mathcal{L}, \mathsf{Unif}) \in \mathsf{HeurBPP}^{\mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}}$ ($(\mathcal{L}_P, \mathsf{Unif}) \in \mathsf{HeurBPP}^{\mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}}$).

A full scheme of the proof overview can also be found in Figure 1.

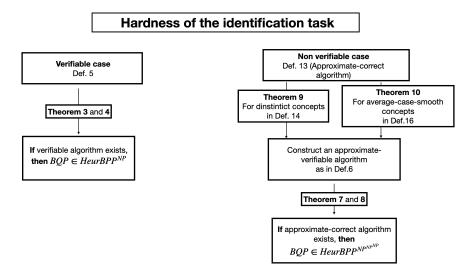


Figure 1: An overview of our proof strategy for the identification task.

We provide now the definition of the two types of concept classes considered in Theorem 6.

c-distinct concept class In the first scenario, we require the concept class to consist of BQP-complete functions, as defined in Def. 8, that are distinct on a sufficiently large fraction of points. More specifically, we define a c-distinct concept class as follows.

Definition 15 (c-distinct concept class). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class. We say \mathcal{F} is a c-distinct concept class if

for all
$$\alpha_1, \alpha_2 \in \{0, 1\}^m, \alpha_1 \neq \alpha_2 \quad \exists S \subset \{0, 1\}^n, |S|/2^n \geq c \quad \text{s.t. } f^{\alpha_1}(x) \neq f^{\alpha_2}(x) \quad \forall x \in S.$$
 (13)

We note that in the case of concept classes containing PromiseBQP functions as defined in Def. 9, for the definition of c-distinct concepts to be meaningful, we require that the set S is a subset of the inputs specified in the promise. We provide here an example of a 0.5-distinct concept class, which therefore satisfies the condition of Theorem 9, where the concepts are all (Promise—)BQP complete functions.

Definition 16 (0.5-distinct concept class with BQP concepts). Let f be a PromiseBQP complete function as in Def. 9 on input $x \in \{0,1\}^n$. Then for any f there exists a concept class \mathcal{F} containing 2^n PromiseBQP-complete concepts defined as:

$$\mathcal{F} = \{ f^{\alpha} : \{0, 1\}^n \to \{0, 1\} \mid \alpha \in \{0, 1\}^n \}$$
 (14)

$$f^{\alpha}: x \to f(x) \oplus (\alpha \cdot x)$$
 (15)

where $\alpha \cdot x$ is the bitwise inner product modulo 2 of the vectors α and x.

Clearly for any $\alpha_1, \alpha_2 \in \{0,1\}^n$, $\alpha_1 \neq \alpha_2$, it holds that $f^{\alpha_1}(x) \neq f^{\alpha_2}(x)$ on exactly half of the input x, also they are all PromiseBQP complete functions as by evaluating one of them we can easily recover the value of f(x).

Average-case-smooth concept class We now consider concept classes where the label space is equipped with a metric such that if two concepts are close under the PAC conditions, then their corresponding labels α_1 and α_2 are also close with respect to the metric on the label space. We formalize this notion of closeness in the definition below.

Definition 17 (Average-case-smooth concept class). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class. We say that \mathcal{F} is average-case-smooth if there exists a distance function $d : \{0,1\}^m \times \{0,1\}^m \to \mathbb{R}^+$ defined over the labels $\alpha \in \{0,1\}^m$ and a $C \geq 0$ such that $\forall \alpha_1, \alpha_2 \in \{0,1\}^m$:

$$\mathbb{E}_{r \sim \text{Unif}(0,1^n)} |f^{\alpha_1}(x) - f^{\alpha_2}(x)| \ge C d(\alpha_1, \alpha_2). \tag{16}$$

We note that in machine learning, it is often the case that closeness of functions in parameter space implies closeness in the PAC sense. However, in Def. 18, we require the reverse: that closeness at the function level implies closeness in parameter space.

6.1 Intermediate result - Hardness result for approximate-verifiable identification algorithm

In order to prove the result in Theorem 6 we first prove the following intermediate result about the hardness for the task of approximate-verifiable identification introduced in Def. 7. To conclude the proof of Theorem 6 we will then show in Section 6.2 how to construct an approximate-verifiable algorithm from an approximate correct-algorithm using higher levels of the PH.

We require that an approximate-verifiable algorithm satisfies the two properties of "soundness" and "completeness" described in Def. 7. We can now state our hardness result for the approximate-verifiable case of the identification task in Def. 7. As in previous cases, we can state our results either on the assumption that there exists a language $\mathcal{L} \in \mathsf{BQP}$ for which $(\mathcal{L},\mathsf{Unif}) \not\in \mathsf{HeurBPP}^\mathsf{NP}^\mathsf{NP}$ or that for a given PromiseBQP language \mathcal{L}_P there exists a distribution $\mathsf{Unif}_{\mathcal{L}_P}$ such that $(\mathcal{L}_P,\mathsf{Unif}_{\mathcal{L}_P}) \not\in \mathsf{HeurBPP}^\mathsf{NP}^\mathsf{NP}$.

Theorem 7 (Approximate-verifiable identification implies (BQP, Unif) \subset HeurBPP^{NPP}). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class containing at least a BQP function f as in Def. 8 associated to a language $\mathcal{L} \in \mathsf{BQP}$. If there exists an approximate-verifiable identification algorithm \mathcal{A}_B for the input distribution Unif as in Def. 7 then $(\mathcal{L}, \mathsf{Unif}) \in \mathsf{HeurBPP}^\mathsf{NPP}$.

Assuming the existence of a distribution $\mathsf{Unif}_{\mathcal{L}_P}$ as in Def. 10 associated to a PromiseBQP complete language, we can also prove the following result.

Theorem 8 (Approximate-verifiable identification implies (BQP, Unif_{\mathcal{L}_P}) \subset HeurBPP^{NPNP}). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class containing at least a PromiseBQP function f as in Def. 9 associated to a language $\mathcal{L}_P \in$ PromiseBQP and let Unif_{\mathcal{L}_P} be an associated input distribution as in Def. 10. If there exists an approximate-verifiable identification algorithm \mathcal{A}_B for the input distribution Unif_{\mathcal{L}_P} as in Def. 7 then $(\mathcal{L}_P, \mathsf{Unif}_{\mathcal{L}_P}) \in \mathsf{HeurBPP}^{\mathsf{NP}^{\mathsf{NP}}}$.

The full proofs of both theorems can be found in Appendix C.3. Here, we outline the main idea behind the proofs, which is shared by both versions of the theorems.

Proof. The idea of the proof is to construct a probabilistic algorithm $M^{\alpha}(x,s)$ which takes as input an x and a random string s and outputs a label y such that for any $\delta \geq 0$:

$$\Pr_{x \sim \mathcal{D}} \left[\Pr_s[M^{\alpha}(x, s) \neq f^{\alpha}(x)] \ge \frac{2}{5} \right] \le \delta$$
 (17)

where the distribution \mathcal{D} is either the uniform distribution for Theorem 7 or $\mathsf{Unif}_{\mathcal{L}_P}$ in case of Theorem 8. In case f^α is a BQP function (or PromiseBQP in case of Theorem 8) then Eq. (17) would imply the corresponding language is contained in HeurBPP, defined in Def. 12. The key idea to construct such an algorithm $M^\alpha(x,s)$ is that with a NP oracle it is possible to sample among the datasets T that satisfy the relation $\mathcal{A}_B(T,\epsilon,r)=\alpha$. Because of the "soundness" and "completeness" properties of the approximate-verifiable algorithm in Def. 7, the algorithm M^α will actually sample from a set of almost all possible datasets with 2/3 of the points labeled by f^α . Therefore, the algorithm M^α is able to output the correct label for almost all the inputs x.

We notice here that, as anticipated in Section 2, that the approximate-verifiable identification task in Def. 7 is closely connected to a variant of the PAC learning framework [KV94b, Moh18], so called proper PAC in Def. 4. The key distinction between the definition of proper PAC learning (Def. 4) and approximate-verifiable identification lies in the requirement that an approximate-verifiable identification algorithm must additionally satisfies the completeness and soundness properties of Def. 7. For the soundness property the identification algorithm must be able to detect invalid datasets in which less than a fraction $\frac{2}{3}$ of the points are labeled consistently with a single concept. As we establish in Proposition 2, in case of classically efficiently evaluable concept classes a proper PAC learner can already satisfies the property of soundness.

Proposition 2. Let $\mathcal{F} = \{f^{\alpha}(x) \in \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a set of classically efficiently evaluable functions parametrized by α . Let $\mathcal{C} = \mathcal{F}$ be a concept class and $\mathcal{H} = \mathcal{F}$ the hypothesis class of the learning algorithm \mathcal{L} . Then, a proper PAC learner \mathcal{L} which satisfies the "completeness" property in Def. 7 exists if and only if there exists an approximate-verifiable identification algorithm as given in Def. 7 for \mathcal{C} .

We refer to Appendix C.1 for the proof of the above proposition. The key idea is that the ability to evaluate the target concepts allows to check the validity of datasets. It is important to note that this will be the main obstacle in proving the hardness of identification for quantum functions, as a classical learner cannot verify quantum-generated datasets.

6.2 Construction of an approximate-verifiable algorithm in the PH

Having established the classical hardness of an approximate-verifiable algorithm in Theorem 7 and 8, in order to prove our main result in Theorem 6 about hardness of the approximate-correct algorithm for the non-verifiable identification task, we show that if there exists an approximate-correct algorithm as in Def. 14, then there also exists an approximate-verifiable algorithm in Def. 7 which operates in the second level of the PH. From the results in Theorem 7 (or 8), this would imply that BQP (or PromiseBQP) concepts would be heuristically computable in the fourth level of the PH.

In particular, since an approximate-correct algorithm already satisfies the "completeness" property of Def. 7, in the following two theorems we show how relax the assumption on the soundness property of Def. 7 and prove hardness results even in the case of an approximate-correct identification algorithm which does not reject any input dataset.

6.2.1 Construction of an approximate-verifiable algorithm with c-distinct concepts

Theorem 9. If there exists an approximate-correct identification algorithm A_B as in Def. 14 for a c-distinct concept class \mathcal{F} , with $c \geq 1/3$, which works correctly for a dataset of size B, then there also exists an approximate-verifiable algorithm A'_{3B} as in Def. 7 in the second level of the PH which works correctly for a dataset of size 3B.

Proof sketch. The proof idea is to show that we can construct an algorithm \mathcal{A}'_{3B} which using the approximate-correct algorithm \mathcal{A}_B and a NP oracle can detect invalid dataset with fewer than 2/3 of the points labeled by a concept α . By doing so we guarantee that \mathcal{A}'_{3B} satisfies the soundness property of Def. 7. Given a set T of 3B training points as input, we first run the approximate-correct algorithm \mathcal{A}_B on T and consider the outputted α . Then we use the NP oracle from the PH to look for a subset of B points which are not labeled by the outputted concept f^{α} , by running the assumed-to-exist approximate-correct identification algorithm on all the subset of B points in T. Since we assume the concepts are c-distinct with $c \geq \frac{1}{3}$, the second property of the identification algorithm in Def. 14 ensures that $\mathcal{A}_B(T,\epsilon,r) = \alpha$ if $\epsilon \leq \epsilon_2$. We can then search for such a subset of points in the second level of the polynomial hierarchy as shown in the full proof in Appendix C.3. If we find such a subset, we then reject the input training set as for sure B of the B points are incorrectly labeled. Otherwise, we accept the dataset as a valid dataset. The algorithm A'_{3B} then performs this check on each input dataset and is then able to identify any dataset as invalid if fewer than 2/3 of its points are labeled by f^{α} . This guarantees that A'_{3B} satisfies the soundness property required by Def. 7.

6.2.2 Construction of an approximate-verifiable algorithm for classes with similarity-preserving parametrizations

We now consider concept classes where the label space is equipped with a metric such that if two concepts are close under the PAC conditions, their corresponding labels α_1 and α_2 are also close with respect to the metric on the label space. We formalize this notion of closeness in the definition below.

Definition 18 (Average-case-smooth concept class). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class. We say that \mathcal{F} is average-case-smooth if there exists a distance function $d : \{0,1\}^m \times \{0,1\}^m \to \mathbb{R}^+$ defined over the labels $\alpha \in \{0,1\}^m$ and a $C \geq 0$ such that $\forall \alpha_1, \alpha_2 \in \{0,1\}^m$:

$$\mathbb{E}_{x \sim \text{Unif}(0,1^n)} |f^{\alpha_1}(x) - f^{\alpha_2}(x)| \ge C \ d(\alpha_1, \alpha_2). \tag{18}$$

We note that in machine learning, it is often the case that closeness of functions in parameter space implies closeness in the PAC sense. However, in Def. 18, we require the reverse: that closeness at the function level implies closeness in parameter space. In the case of a concept class which satisfies Def. 18, we can show the following result.

Theorem 10. If there exists an approximate-correct identification algorithm A_B as in Def. 14 for an average-case-smooth concept class \mathcal{F} , which works correctly for a dataset of size B, then there also exists an approximate-verifiable algorithm A'_{3B} as in Def. 7 in the second level of the PH which works correctly for a dataset of size 3B.

Proof sketch. The full proof can be found in Appendix C.3. The key idea is analogous to the one in Theorem 9. Given a set T of 3B training points as input, we first run the approximate-correct algorithm \mathcal{A}_B on T and consider the outputted α . We then look for a subset of B points which are not labeled by the outputted

concept f^{α} using the NP oracle. In particular, we run the algorithm \mathcal{A}_B on every subset of B points in T and leverage the smoothness condition of the concept class in Def. 18 to check that the distance between the outputted concept by \mathcal{A}_B and the "opposite" concept $f^{\bar{\alpha}}$ (i.e., a function which disagrees with f^{α} on every input $x \in \{0,1\}^n$) is less than $\frac{1}{3}$. Because of the second property in Def. 14, we can search for such a subset of points in the second level of the polynomial hierarchy as shown in the full proof in Appendix C.3. If we find such a subset, we then reject the input training set as for sure B of the B points are incorrectly labeled. Otherwise, we accept the dataset as a valid dataset.

7 Discussion: some implications of our results

In this section, we discuss the connections between our theoretical findings with other well-known physically relevant tasks: Hamiltonian learning, the learning of parametrized observables and the learning of order parameters.

For Hamiltonian learning, we clarify why an efficient classical algorithm for the identification task remains feasible. We then show how our results demonstrate the hardness of the identification task in learning parametrized observables. Finally, we discuss the connection to the task of learning of order parameters and the potential for quantum advantages therein.

7.1 Hardness of identification for a physically motivated BQP-complete concept class

In the recent work [MGD24b] some of the authors provided a physically motivated learning problem for which a learning separation was proved. Importantly, the classical hardness for the task was achieved by considering concepts being BQP-complete functions as in Def. 9 and then arguing that a classical learner would not be able to evaluate such concepts unless BQP \subset P/poly. An interesting question is whether the classical hardness could arise already from identifying the target concept that labels the data, rather than only from evaluating it. Using the results in Section 6, we can answer this question in an affirmative way for a subclass of problems for which the hardness of evaluation holds. Let us restate the learning problem presented in [MGD24b] and show how our results in the previous section apply to it. The learning problem considered is described by the following concept class:

$$\mathcal{F}^{U,O} = \{ f^{\alpha}(\boldsymbol{x}) \in \mathbb{R} \mid \boldsymbol{\alpha} \in [-1,1]^{m} \}$$
 with:
$$f^{\alpha}(\boldsymbol{x}) : \quad \boldsymbol{x} \in \{0,1\}^{n} \to f^{\alpha}(\boldsymbol{x}) = \text{Tr}[\rho_{H}(\boldsymbol{x})O(\boldsymbol{\alpha})]$$
$$O(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_{i} P_{i}.$$
 (19)

Where $\rho_U(\mathbf{x})$ is a "classically hard to compute" quantum state (i.e., certain properties of this state are hard to compute) depending on $\mathbf{x} \in \{0,1\}^n$. An example could be $\rho_U(\mathbf{x}) = U |\mathbf{x}\rangle \langle \mathbf{x}| U^{\dagger}$ with $U = e^{iH\tau}$ where H is a local Hamiltonian whose time evolution is universal for BQP [Fey85]. As each P_i represents a k-local Pauli string, the function f^{α} is a BQP-complete function for some values of α . For example, for the α_Z which corresponds to $O(\alpha_Z) = Z \otimes I \otimes ... \otimes I$ the function f^{α_Z} is exactly the function which corresponds to the time evolution problem in the literature [KSV02], which is known to be universal for BQP. This already motivates the classical hardness of the learning task under the assumption BQP $\not\subseteq P/\text{poly}$ as the classical learner is required to evaluate a correct hypothesis on new input data. We now argue that concept classes of the kind $\mathcal{F}^{U,O}$ are also not classically identifiable as in Def. 14. In order to show it, we will consider a particular construction for the quantum states $\rho_U(\mathbf{x})$ and observables $O(\alpha)$ such that the corresponding concept class $\mathcal{F}^{U,O}$ satisfies the definition of c-distinct concept class in Def. 15 with c = 1/3.

Theorem 11. There exists a family of unitaries $\{U(x)\}_x$ and observables $\{O(\alpha)\}_{\alpha}$ such that the corresponding concept class $\mathcal{F}^{U,O}$ is not classically identifiable by an approximate-correct algorithm as in Def. 14, but is identifiable by a quantum learning algorithm.

The proof of Theorem 11 can be found in Appendix D. The key idea is to construct a particular family of quantum states ρ_U and observables O such that the corresponding concept class in Eq. (19) contains c-distinct concepts with $c \geq 1/3$. We achieve this by leveraging the Reed-Solomon code to construct functions that differ from each other on at least 1/3 of all the input x. As this is still an instance of the same category of concept classes as studied in [MGD24b], the quantum learning algorithm from this work will solve the identification task, and also the corresponding evaluation task.

7.2 Hamiltonian Learning

Hamiltonian learning is a well-known problem which, arguably surprisingly, admits an efficient classical solution. Moreover, the Hamiltonian learning problem can be easily framed as a standard learning problem within the PAC framework, revealing clear connections to the identification task discussed in this paper. Although the underlying concepts may initially seem inherently quantum, the existence of an efficient classical algorithm for solving the problem raises the question of why our hardness results do not extend to this setting. In its main variant, the task involves reconstructing an unknown Hamiltonian from measurements of its Gibbs state at a temperature T. Specifically, given an unknown local Hamiltonian of the form $H(\lambda) = \sum_i \lambda_i P_i$, the goal of the Hamiltonian learning procedure is to recover λ from measurements of its Gibbs state $\rho(\beta, \lambda)$ at temperature T:

$$\rho(\beta, \lambda) = \frac{e^{\beta H(\lambda)}}{\text{Tr}[e^{\beta H(\lambda)}]}$$

where $\beta = \frac{1}{T}$. In [AAKS20], the authors proposed a classical algorithm capable of recovering the unknown Hamiltonian parameterized by λ using a polynomial number of measurements of the local Pauli terms $\{P_i\}_i$ that constitute the target Hamiltonian $H(\lambda) = \sum_i \lambda_i P_i$. In [HKT24], an also time-efficient algorithm was proposed. The above task can be reformulated as an identification problem by considering the following concept class:

$$\mathcal{F}_{\beta} = \{ f^{\lambda}(\boldsymbol{x}) \in \mathbb{R} \mid \lambda \in [-1, 1]^{m} \}$$
 (20)

with:
$$f^{\lambda}(\boldsymbol{x}): \boldsymbol{x} \in \mathcal{X} \subseteq \{0,1\}^n \to \text{Tr}[\rho(\beta, \boldsymbol{\lambda})O(\boldsymbol{x})]$$

$$O(\boldsymbol{x}) = \sum_{i=1}^{\text{poly}(n)} x_i P_i.$$

Then given polynomially many training samples of the kind $T = \{(\boldsymbol{x}_{\ell}, f^{\boldsymbol{\lambda}}(\boldsymbol{x}_{\ell}))\}_{\ell}$ it is possible to recover the expectation values of the local Pauli terms $\{P_i\}_i$ and therefore the identification problem can be solved using the learning algorithm in [HKT24] for learning the vector $\boldsymbol{\lambda}$.

We now provide a list of incompatibilities between the task of Hamiltonian learning and the identification task considered in our results, and examine them to try to find the crux of the reason why Hamiltonian learning remains classically feasible.

• Hamiltonian learning is a regression problem. It is important to highlight that the concepts in Eq. (20) are not binary functions, as they are in our theorems, but rather form a regression problem. We note that this categorical difference alone is also not the end of the explanation. It is in fact possible to "binarize" the concepts in the following way:

$$f^{\lambda}(\boldsymbol{x}, i) : (\boldsymbol{x}, i) \in \subseteq \{0, 1\}^n \times \{0, 1\}^{\log n} \to \min \left[\text{Tr}[\rho(\beta, \boldsymbol{\lambda}) O(\boldsymbol{x})], i \right]$$
 (21)

where $\operatorname{bin} \left[\boldsymbol{y}, i \right]$ returns the *i*-th bit of \boldsymbol{y} . A Hamiltonian learning algorithm can still be used to determine the unknown $\boldsymbol{\lambda}$. This can be done, for example, by considering an input distribution that focuses on exploring the most significant bits of the extreme points where the observable $O(\boldsymbol{x})$ reduces to a single Pauli string, specifically inputs $\boldsymbol{x} \in \{0,1\}^n$ with Hamming weight 1.

- The concepts are not BQP-hard. The concepts in \mathcal{F}_{β} , binarized as in Eq. (21), involve measurements performed on Gibbs states. For large values of β , these states closely approximate ground states, which might suggest a connection to tasks where quantum computers are expected to outperform classical ones. However, regardless of how hard it is to estimate expectation values from cold Gibbs states, it is important to clarify that each concept is associated with a fixed Gibbs state, and the input $x \in \{0,1\}^n$ simply determines the observable being measured. This is similar to the "flipped concepts" case studied in [MGD24b]. It is easy to see that a P/poly machine can compute these concepts given the expectation values of the (polynomially many) Pauli strings $\{P_i\}_i$ as advice, due to the linearity of the trace. Thus, the concepts we have here are somewhere in the intersection of P/poly and BQP-like problems (depending on what temperature Gibbs states we are dealing with, let us call the corresponding class A, where A could be QXC [BCG⁺24]). However, even this is not sufficient to full explain the apparent gap between our no-go results and the efficiency of Hamiltonian learning. In our proofs we worked towards the (unlikely) implication that BQP was not in a heuristic version of a level of the PH. Here, we can construct fully analogous arguments and obtain the implication that, e.g., $P/poly \cap A$ is in PH, and we also have no reason to believe this to be true. Indeed, proving that this inclusion holds would, to our understanding, constitute a major result in quantum complexity theory. The reasons why the no-go's do not apply are more subtle still.
- Hamiltonian learning algorithm doesn't satisfy the assumptions of the right type of identification algorithm. It is clear that the Hamiltonian learning algorithm does not satisfy the conditions of an approximate-verifiable algorithm in Def. 7, as it fails to detect datasets that do not contain enough points labeled by a single concept it always outputs some guess for the Hamiltonian. However we could again try to circumvent this issue by employing the constructions from Theorem 9 or Theorem 10, to construct approximate-verifiable identification algorithm somewhere in the PH, which would again suffice for a likely contradiction. However, it is important to note that in normal Hamiltonian learning settings neither of the two Theorems apply. The theorems require that concept class must either consist of sufficiently distinct concepts (Def.15) or exhibit average-case smoothness (Def.18). On the face of it, neither of these conditions seem to hold for the concept class in Eq. (20), and it is not clear how one would go about attempting to enforce them. This final point is in our view at the crux of the difference between the settings where our no-go's apply and Hamiltonian learning.

While we leave the investigation of the hardness of the Hamiltonian learning task as a potential direction for future work, already at this point an interesting observation emerges. It is a natural question if the conditions of the two Theorems 9 and 10 could be relaxed and generalized, which would lead to the hardness of identification for broader classes of problems. Due to the analysis of the Hamiltonian learning case we see that if one could generalize the settings to the point they apply to Hamiltonian learning, since Hamiltonian learning is classically tractable, this would imply very surprising results in complexity theory (see second bullet point above). We see it more likely that this is a reason to believe the range of generalization of the settings where identification is intractable is more limited and will not include the standard settings of Hamiltonian learning.

7.3 The case of learning of order parameters

Another physically meaningful problem that can be framed as an identification task is learning the order parameter that distinguishes between different phases of matter. Consider, for example, a family of Hamiltonians $\{H(\boldsymbol{x})\}_{\boldsymbol{x}}$ parametrized by vectors $\boldsymbol{x} \in [-1,1]^m$, where the corresponding ground states exhibit distinct phases depending on the value of \boldsymbol{x} . In many cases, such as symmetry-breaking phases, there exists a local order parameter of the form $O = \sum_i \alpha_i P_i$, whose expectation value on a given ground state reveals the phase to which it belongs to. The task is to learn the order parameter from a collection of samples $\{(\rho(\boldsymbol{x}_\ell), y_\ell)\}_\ell$, where each $\rho(\boldsymbol{x}_\ell)$ represents the ground state of the Hamiltonian $H(\boldsymbol{x}_\ell)$, and y_ℓ denotes the label identifying its associated phase. We can formalize the problem by considering the following concept class:

$$\mathcal{F}_{\alpha} = \{ f^{\alpha}(\mathbf{x}) \in \mathbb{R} \mid \alpha \in [-1, 1]^m \}$$
 (22)

with:
$$f^{\alpha}(x)$$
: $x \in \mathcal{X} \subseteq \{0,1\}^n \to \text{Tr}[\rho(x)O(\alpha)]$
 $O(\alpha) = \sum_i \alpha_i P_i$.

Learning the correct order parameter from ground states labeled by their phases can thus be viewed as identifying the true α^* , using training data of the form $T = \{(\rho(x_\ell), \text{Tr}[\rho(x_\ell)O(\alpha^*)])\}_\ell$. In this setting, the value $\text{Tr}[\rho(x_\ell)O(\alpha^*)]$ acts as a phase indicator. For instance, in systems exhibiting two distinct phases, the sign of $\text{Tr}[\rho(x_\ell)O(\alpha^*)]$ serves to distinguish between them. The concepts in Eq. (22) compute expectation values on ground states, a task that for a general local Hamiltonian is in QMA-hard, but the situation is actually more involved.

First, we note that the closely related task learning of phases of matter given the shadows of the ground states - so where the data consists of pairs $(\sigma(\rho(x)), \text{phase}(x))$, where $\sigma(\cdot)$ denotes the classical shadow of a state, and "phase" assigns a binary label specifying the phase - is classically easy, even for many topological phases of matter [HKT⁺22]. In this case, the task is to assign the correct phase to a new datum which is a shadow of a new state given as $\sigma(\rho(x))$. This is different from the setting we consider here, where we explicitly deal with the specification x of the Hamiltonian; that is, pairs $(x, \text{phase}(x))^{-14}$. While [HKT⁺22] also shows how mappings $x \to \sigma(\rho(x))$ can be classically learned as long as the Hamiltonians specified by all x are within the same phase, the identifying of phases requires the crossing of the phase boundaries, and it is not clear how the approaches could be combined.

To analyze the perspectives of classical intractability and then quantum tractability of this task from the lens of the work of this paper, we analyze the key criteria. First, for our hardness results to apply at all, the concepts in the concept class should be sufficiently hard, in a complexity-theoretic sense. That is, the functions that we consider $x \to \text{Tr}[\rho(x)O(\alpha)]$ should be classically intractable, yet quantum easy. In general, this is easy to achieve: by using the standard Kitaev circuit-to-Hamiltonian constructions we can construct Hamiltonians whose ground states encode the output of arbitrary quantum circuits (here encoded in x), as was used by some of the authors in [MGD24b]. However, from the perspective of phases of matter identification, it is important to notice that such BQP—hard Hamiltonians are critical: they have an algebraically vanishing gap, and it is an open question whether in this sense BQP—hard Hamiltonians could be gapped at all [GGC18]. At least in the cases of conventional phases of matter (symmetry breaking, even topological), the phases are typically characterized by a constant gap. This suggests that the "hard computations" could only be happening at (or increasingly close to) the phase boundary.

However, we are also reminded that the observable we need to ultimately measure has a meaning: it is the order parameter. This has an interesting implication. We are interested in the setting where the function $x \to \text{Tr}[\rho(x)O(\alpha)]$ corresponds to, intuitively, the characteristic function of some BQP(-hard) language \mathcal{L} . But since this function is also the order parameter, the *yes* instances of the language $(x \in \mathcal{L}_{yes})$ must correspond to Hamiltonians in one phase, whereas the *no* instances must correspond to the other. This observation allows for a simple cheat, as it highlights the importance of the mapping $x \to H(x)$, which we have the freedom to specify. One can conceal all the hardness in this map and simply choose it such that H(x) is some Hamiltonian in one phase for $x \in \mathcal{L}_{yes}$, and in another for the rest. This is of course unsatisfactory as it involves a highly contrived parametrization.

For natural, smooth parametrizations of the Hamiltonian space, we do need to worry about the constant gap property and so it is not clear whether hard functions emerge in standard settings. If, however, we move to more exotic systems, e.g. general critical systems, systems with complex non-local order parameters¹⁵ and dynamic phases of matter, it becomes more likely that the right theoretical conditions arise even with natural parametrizations.

The hardness of the concepts will ensure the hardness of evaluation-type tasks which is the first step. To achieve the hardness of identification, we need more, i.e., either c-distinct concepts or a smooth family. Learning of order parameters is a closely related task to the learning of observables, and as we discussed in Section 7.1, for this more general case it is possible to construct cases that satisfy all the desired assumptions.

¹⁴Also, the setting in [HKT⁺22] do not explicitly find the observable, which is the order parameter, although we suspect this can be computed from the hyperplane found by the linear classifier.

¹⁵Note that we could encode universal quantum computations in complex enough global measurements.

Whether similar conditions will also be met for some natural settings involving exotic (or standard) phases of matter remains a target of ongoing research.

8 Acknowledgments

The authors are thankful to Sofiene Jerbi for numerous discussions and suggestions to look into Reed-Solomon codes, and to Matthias Caro for discussions involving the packing numbers of certain quantum concept classes. This publication is part of the project Divide & Quantum (with project number 1389.20.241) of the research programme NWA-ORC which is (partly) financed by the Dutch Research Council (NWO). This work was also supported by the Dutch National Growth Fund (NGF), as part of the Quantum Delta NL programme. This work was also supported by the European Union's Horizon Europe program through the ERC CoG BeMAIQuantum (Grant No. 101124342).

A Further definitions from complexity theory

In this section we provide further useful definitions. We first define the complexity class P/poly which appears in Proposition 1 and Theorem 5.

Definition 19 (Polynomial advice [AB09]). A problem $L: \{0,1\}^* \to \{0,1\}$ is in P/poly if there exists a polynomial-time classical algorithm \mathcal{A}_B with the following property: for every n there exists an advice bitstring $\alpha_n \in \{0,1\}^{\text{poly}(n)}$ such that for all $x \in \{0,1\}^n$:

$$A_B(x, \alpha_n) = L(x). \tag{23}$$

We also provide the definition of the class BPP/samp^{BQP}, which will appear in Theorem 12.

Definition 20 (BPP/samp^{BQP}). A problem $L: \{0,1\}^* \to \{0,1\}$ is in BPP/samp^{BQP} if there exists a polynomial-time quantum algorithm \mathcal{S} and a polynomial-time classical randomized algorithm \mathcal{A} such that for every n:

- \mathcal{S} generates random instances $x \in \{0,1\}^n$ sampled from the distribution \mathcal{D}_n .
- \mathcal{A} receives as input $\mathcal{T} = \{(x_i, L(x_i)) \mid x_i \sim \mathcal{D}_n\}_{i=1}^{\operatorname{poly}(n)}$ and satisfies for all $x \in \{0, 1\}^n$:

$$\Pr(\mathcal{A}(x,\mathcal{T}) = L(x)) \ge \frac{2}{3},$$
 (24)

where the probability is taken over the internal randomization of A and T.

B Proofs about random generatability

B.1 Hardness of random generatability based on the assumption $BPP/samp^{BQP} \not\subseteq BPP$

In this section, we demonstrate that achieving exact random generatability for quantum functions would lead to $\mathsf{BPP}/\mathsf{samp}^{\mathsf{BQP}} \subseteq \mathsf{BPP}$. Similar to Theorem 1, we show here classical hardness of the task defined in Def. 1 for quantum functions, though based on a different complexity-theoretic assumption. The proof is straightforward and relies on the idea that if a classical machine could efficiently generate samples for any quantum function, passing those samples as advice would offer no additional advantage.

Theorem 12 (Exact RG implies BPP/samp^{BQP} \subseteq BPP). If BPP/samp^{BQP} $\not\subseteq$ BPP, then there exists a quantum function $f:\{0,1\}^n \to \{0,1\}$ as given Def. 9 which is not exact random verifiable as in Def. 1 with a classical algorithm \mathcal{A} .

Proof. Suppose $\forall f$ in Def. 9 there exists a randomized algorithm \mathcal{A}_f such that $\mathcal{A}_f(r) \to (x, f(x))$ with $x \sim \text{Unif}(\{0,1\}^n)$, for r sampled uniformly at random. Then every function $g \in \mathsf{BPP/samp}^{\mathsf{BQP}}$ can be computed in BPP by first generating the samples (x, g(x)) using \mathcal{A}_g .

We now argue that the hypothesis of the Theorem 12 are indeed reasonable. Indeed, a separation between the class BPP and BPP/samp^{BQP} can be proven if we assume the existence of a sequence of quantum circuits $\{U_n\}_n$, one for each size n, such that the Z measurement on the first qubit is hard to decide classically. A proof idea for the following theorem is stated in [HBM+21], and we include here the whole proof for completeness.

Theorem 13 (BPP/samp^{BQP} $\not\subset$ BPP, unless BPP = BQP for unary languages). If there exists a uniform sequence of quantum circuits $\{U_n\}_n$, one for each size n=1,2,... such that the Z measurement on the first qubit is hard to decide classically, then BPP/samp^{BQP} $\not\subset$ BPP.

Proof. As shown in [HBM⁺21], such a sequence of circuits would define a unary language

$$L_{BOP} = \{1^n \mid \langle 0^n | (U_n)^{\dagger} Z U_n | 0^n \rangle \ge 0\}$$
 (25)

that is outside BPP but inside BQP. The authors then consider a classically easy language $L_{\text{easy}} \in \text{BPP}$ and assume that for each input size n, there exists an input $a_n \in L_{\text{easy}}$ and an input $b_n \notin L_{\text{easy}}$. Then it is possible to define a new language:

$$L = \bigcup_{n=1}^{\infty} \left\{ x \mid \forall x \in L_{\text{easy}}, \ 1^n \in L_{\text{hard}}, \ |x| = n \right\} \cup \left\{ x \mid \forall x \notin L_{\text{easy}}, \ 1^n \notin L_{\text{hard}}, \ |x| = n \right\}.$$
 (26)

For each size n, if $1^n \in L_{\text{BQP}}$, L would include all $x \in L_{\text{easy}}$ with |x| = n. If $1^n \notin L_{\text{hard}}$, L would include all $x \notin L_{\text{easy}}$ with |x| = n. By definition, if we can determine whether $x \in L$ for an input x using a classical algorithm (BPP), we could also determine whether $1^n \in L_{\text{BQP}}$ by checking whether $x \in L_{\text{easy}}$. However, this must be impossible as we are assuming that L_{BQP} cannot be decided classically. Hence, the language L is not in BPP. On the other hand, for every size n, a classical machine learning algorithm can use a single training data point (x_0, y_0) to decide whether $x \in L$ by what said above.

We note here that the existence of unary languages in BQP but not in BPP is a well believed assumption.

B.2 Proofs of the theorems from the main text concerning the hardness of random generatability

B.2.1 Hardness of exact random generatability

Theorem 1 (Exact Random generatability implies PromiseBQP $\subseteq P^{NP}$). Let $f = \{f_n\}_{n \in \mathbb{N}}$ be a family of PromiseBQP complete functions as in Def. 9. If there exists a classical randomized poly-time uniform algorithm that generates samples $(x, f_n(x))$ correctly with probability 1 as in Def.1, with $x \sim \text{Unif}(\{0,1\}^n)$, then P^{NP} contains PromiseBQP.

Proof. The existence of a randomized poly-time algorithm which satisfies Theorem 1 is equivalent to the existence of a uniform family of poly-sized algorithms C(r) such that a random choice of $r \in \{0,1\}^{\text{poly}(n)}$ outputs a tuple (x, f(x)) uniformly at random, i.e. :

$$C = \{C(r) \mid r \in \{0, 1\}^{\text{poly(n)}}\}$$
(27)

Then, for every x there exists at least one r_x such that $C(r_x) = (x, f(x))$. In particular, for a given x a P^{NP} machine can in polynomial time find a corresponding r_x . Consider the set

$$P(\tilde{C}) = \{(u, x) \mid \exists v \in \{0, 1\}^* \text{ s.t. } \tilde{C}(uv) = x\}$$
(28)

where \tilde{C} is the same family of algorithms C except that the last bit of the output (i.e., the value f(x)) is omitted and $\tilde{C}(r) \in \tilde{C}$. Then (u, x) is in $P(\tilde{C})$ if and only if u is a prefix of some r_x inverse of x with respect to \tilde{C} . Clearly $P(\tilde{C})$ is in NP as a correct $u \in \{0, 1\}^*$ can be verified in polynomial time as also the family \tilde{C} consists of polynomial sized algorithms. Then we can run the following algorithm in P^{NP} :

As for every x there exists at least one corresponding r_x , the above algorithm always succeeds in finding a correct r_x . Once the string r_x is found, the P machine can run C on that string and evaluate f(x). Since f(x) can decide a PromiseBQP complete language by Def. 9, it follows that P^{NP} can also correctly decide every $x \in \{0,1\}^n$.

Algorithm 1 Prefix Search Algorithm

```
1: function PrefixSearch(x)
2: u \leftarrow \varepsilon; \Rightarrow where \varepsilon denotes the empty string
3: for |r| times do
4: if (u1, x) \in P(\tilde{C}) then u \leftarrow u1 else u \leftarrow u0
5: end if
6: end for
7: return u
8: end function
```

B.2.2 Hardness of approximate random generatability

In this and the following sections, our proofs concerning the hardness of the approximate random generatability and of the identification task will rely on a well-known result about sampling witnesses of an NP relation using an NP oracle. Specifically, the authors of [BGP00] showed that for a given NP language L and relation R, where $L = \{x \mid \exists w \text{ such that } R[x,w] = 1\}$, it is possible to uniformly sample witnesses w from the set $R_x = \{w \mid R[x,w] = 1\}$ for any $x \in L$. Since this result will be central to all our subsequent proofs, we include the main theorem from [BGP00] here for reference.

Theorem 14 (Theorem 3.1 in [BGP00]). Let R be an NP-relation. Then there is a uniform generator for R which is implementable in probabilistic, polynomial time with an NP-oracle.

For convenience, we also report here the definition of approximate random generatability of Def. 2 in the main text.

Definition 21 (Approximate random generatability (Def. 2 in the main text).). Let $f = \{f_n\}_{n \in \mathbb{N}}$ be a uniform family of functions, with $f_n : \{0,1\}^n \to \{0,1\}$. f is approximately random generatable under the distribution \mathcal{D} if there exists an efficient non-uniform (randomized) algorithm \mathcal{A} such that given as input a description of f_n for any n, a random string r and an error value ϵ , outputs:

$$\mathcal{A}(f_n, \epsilon, r) = (x_r, f_n(x_r)) \tag{29}$$

with $(x_r, f_n(x_r)) \sim \pi_{\epsilon}^f(\mathcal{D})$ when r is sampled uniformly at random from the distribution of all the random strings. In particular, $\pi_{\epsilon}^f(\mathcal{D})$ is a probability distribution over $\{0,1\}^n \times \{0,1\}$ which satisfies: $\|\pi_{\epsilon}^f(\mathcal{D}) - \pi^f(\mathcal{D})\|_{TV} \leq \epsilon$, where $\pi^f(\mathcal{D})$ is the same distribution defined in Def. 1.

We can now state our hardness result for the approximate case of random generatability.

Theorem 2 (Approximate Random generatability implies (PromiseBQP, Unif_P) \in HeurBPP^{NP}). Let $f = \{f_n\}_{n \in \mathbb{N}}$ be a family of PromiseBQP functions which is the characteristic function of a language $\mathcal{L}_P \in$ PromiseBQP complete as in Def. 9, and let Unif_{\mathcal{L}_P} be a distribution defined as in Def. 10. If there exists a polynomial time algorithm \mathcal{A} which satisfies Def.2 for the input distribution Unif_{\mathcal{L}_P}, then $(\mathcal{L}_P, \mathsf{Unif}_{\mathcal{L}_P}) \in$ HeurBPP^{NP}.

Proof. The proof follows from the proof of Theorem 1. Consider the same families of algorithms \mathcal{C} and $\tilde{\mathcal{C}}$ introduced in the proof of Theorem 1. The existence of an algorithm \mathcal{A} as in Def. 2 guarantees the existence of such families \mathcal{C} and $\tilde{\mathcal{C}}$. Let $W = \{r_i\}_{i=1}^M$ be a set of M random strings $r_i \in \{0,1\}^{|r|}$, with |r| = poly(n) and consider the set:

$$P(\tilde{C}, W) = \{ x \in \{0, 1\}^n \mid \exists r \in \{0, 1\}^{|r|} \text{ s.t. } \tilde{C}(r) = x \& r \notin W \}$$
(30)

If the set W has a size M = poly(n) then $P(\tilde{C}, W)$ can be decided in NP since both the conditions $\tilde{C}(r) = x$ and $r \notin W$ can be verified in polynomial time. Theorem 14 from [BGP00] guarantees the existence of an algorithm A_R such that, given the NP relation $R: \tilde{C}(r) = x$ and an input x, outputs, with high

probability, a sample r such that $\tilde{C}(r) = x$ uniformly at random among all the witnesses of x. Then the following algorithm runs in P^{NP} .

```
1: function MultiPrefixSearch(x)
 2:
        r \leftarrow \varepsilon
                                                                                         \triangleright where \varepsilon denotes the empty string
        W = \{r\}
 3:
 4:
         for poly(|x|) times do
            r \leftarrow A_R[x]
 5:
               W = W + \{r\}
 6:
 7:
         end for
         for i: 1 < i < |W| times do
 8:
             return r_i \in W
 9:
         end for
10:
11: end function
```

Where we denote as $A_R[x]$ the algorithm in [BGP00] which uniformly samples witnesses for the relation $R: \tilde{C}(r) = x$ corresponding to the input x. On an input x, the algorithm MultiPrefixSearch outputs a list of polynomial different random strings $\{r_x^i\}_i$ (if they exists) for which $\tilde{C}(r_x^i) = x \ \forall i$, sampled uniformly at random among all the random strings associated to x. We now construct the following algorithm \mathcal{A}' acting on input x. Specifically, on any $x \in \{0,1\}^n$, \mathcal{A}' first applies the algorithm MultiPrefixSearch(x), then either:

- If MultiPrefixSearch(x) outputs an empty string, then the algorithm assigns a random value to f(x)
- Otherwise, the algorithm computes the lables $\mathcal{A}(r_x^i) = (x, f(x))_{r_x^i}$ for any of the random strings outputted by MultiPrefixSearch(x). It then assigns to x the label determined by the majority vote among all the $f(x)_{r_x^i}$ values obtained.

Now we show that the above algorithm \mathcal{A}' is able to classify the language \mathcal{L} correctly on average with respect to the uniform input distribution. More precisely, for each x consider the sets $R_x^T = \{r_x \mid \mathcal{A}(r_x) = (x,y) \text{ with } y = f(x) \oplus 1\}$. Also, let R_{tot} be the set of all the random strings of the algorithm \mathcal{A} , clearly $|R_{tot}| = \sum_{x \in \{0,1\}^n} |R_x^T| + |R_x^T|$. Notice that $|R_x^T|/|R_{tot}|$ precisely represents the probability that the classical algorithm \mathcal{A} outputs the pair (x, f(x)). Similarly, the probability that \mathcal{A} outputs a tuple containing x is $|R_{tot}(x)|/|R_{tot}|$ where $|R_{tot}(x)| = |R_x^T| + |R_x^F|$. By definition of difference in total variation between two distributions, i.e. $||p-q||_{TV} = \frac{1}{2} \sum_x |p(x)-q(x)|$, we have the following relation between the distribution π_{ϵ} generated by \mathcal{A} and the target distribution Unif $P(\{0,1\}^n) \times f(\text{Unif } P(\{0,1\}^n))$ uniform over the input of the promise of the language \mathcal{L} :

$$\|\pi_{\epsilon} - \operatorname{Unif}_{P}(\{0,1\}^{n}) \times f(\operatorname{Unif}_{P}(\{0,1\}^{n}))\|_{TV} = \frac{1}{2} \sum_{x \in P} \left| \frac{|R_{x}^{T}|}{|R_{\text{tot}}|} - \frac{1}{|P|} \right| + \frac{1}{2} \sum_{x \in P} \frac{|R_{x}^{F}|}{|R_{tot}|} + \frac{1}{2} \sum_{x \notin P} \frac{|R_{x}^{T}| + |R_{x}^{F}|}{|R_{tot}|},$$
(31)

where $P = \mathcal{L}_{yes} \cup \mathcal{L}_{no} \subset \{0,1\}^n$ denotes the set of x belonging to the promise of for the language \mathcal{L} . We now bound the number of $x \in P$ for which the probability of \mathcal{A} computing incorrectly them exceeds 1/3, specifically we are interested in the size of the set X_{wrong} such that:

$$X_{\text{wrong}} = \left\{ x \in P \mid \frac{|R_x^T|}{|R_{\text{tot}}(x)|} \le \frac{2}{3} \right\}. \tag{32}$$

From Eq. (31), it follows:

$$\|\mathcal{L}_{\epsilon} - \text{Unif}(\{0,1\}^n) \times f(\text{Unif}(\{0,1\}^n))\|_{TV} \ge \frac{1}{2} \sum_{x \in P} \left| \frac{|R_x^T|}{|R_{\text{tot}}|} - \frac{1}{|P|} \right|$$
 (33)

$$\geq \frac{1}{2} \sum_{x \in X_{\text{wrong}}} \left| \frac{\frac{2}{3} |R_{\text{tot}}(x)|}{|R_{\text{tot}}|} - \frac{1}{|P|} \right|$$
 (34)

$$\sim \frac{1}{2} \sum_{x \in X_{\text{wrong}}} \left| \frac{2}{3} \frac{1}{|P|} - \frac{1}{|P|} \right|$$
 (35)

$$=\frac{1}{2}\frac{|X_{\text{wrong}}|}{3\cdot|P|},\tag{36}$$

where we used the fact that as the output distribution \mathcal{L}_{ϵ} is close in TV with the uniform distribution over the $x \in \{0,1\}^n$ (and the corresponding labels y = f(x)), then $\frac{|R_{\text{tot}}(x)|}{|R_{\text{tot}}|} = \frac{1}{|P|} + \delta_x$ with the constraint $\sum_{x} |\delta_x| \leq \epsilon$. We can then assume $\delta_x << \epsilon$, and therefore neglect it in the derivation above. From Eq. (36), it directly follows that $|X_{\text{wrong}}| \leq 6\epsilon \cdot |P|$ and therefore the fraction of $x \in |P|$ for which \mathcal{A} outputs the correct label with a probability less than 2/3 is approximately 6ϵ .

Then, the algorithm \mathcal{A}' , with a NP oracle, correctly decides in BPP at least a fraction $1-6\epsilon$ of all the possible $x \in \{0,1\}^n$. When x are sampled uniformly, algorithm \mathcal{A}' is therefore able to decide the language \mathcal{L} in HeurBPP^{NP} with respect to the distribution Unif $_{\mathcal{L}_P}$ uniform over the set of x belonging to the promise.

C Proofs of the theorems from the main text concerning the identification task

C.1 Proof of Proposition 2

Proposition 2. Let $\mathcal{F} = \{f^{\alpha}(x) \in \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a set of classically efficiently evaluable functions parametrized by α . Let $\mathcal{C} = \mathcal{F}$ be a concept class and $\mathcal{H} = \mathcal{F}$ the hypothesis class of the learning algorithm \mathcal{L} . Then, a proper PAC learner \mathcal{L} which satisfies the "completeness" property in Def. 7 exists if and only if there exists an approximate-verifiable identification algorithm as given in Def. 7 for \mathcal{C} .

Proof. We will refer to the definition of PAC learning as given in Def. 4. One direction is easy. If there exists an approximate-verifiable algorithm \mathcal{A}_B then there also exists a proper PAC learner \mathcal{L} , by just taking $\mathcal{L} = \mathcal{A}_B$. The opposite direction is also true. Suppose there exists a proper PAC learner \mathcal{L} as in Def. 4. In order to construct an approximate-verifiable identification algorithm we need to ensure that the soundness property in Def. 7 is satisfied, i.e. the learning algorithm is able to reject dataset where the maximum fraction of points consistent with only one concept is less than 2/3. This can be done in the following way. Given an input dataset T, we first run the learner \mathcal{L} which outputs a concept labeled by a $\tilde{\alpha}$. Importantly, we require the error ϵ to be less than 1/3. Since we are assuming that \mathcal{F} is a set of classically evaluable functions, we can now check whether the points in T are labeled by $f^{\tilde{\alpha}}$. Since we required an error 1/3, with probability δ there will be a subset of points labled by $\tilde{\alpha}$. If there is not, we reject the dataset.

C.2 Proof of hardness for the identification task in the verifiable case

Theorem 3 (Identification hardness for the verifiable case - promise). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^{n} \to \{0,1\} \mid \alpha \in \{0,1\}^{m}, m = \text{poly}(n)\}$ be a concept class such that there exists a $f^{\alpha} \in \mathcal{F}$ which is the characteristic function of a language $\mathcal{L}_{P} \in \mathsf{Promise} - \mathsf{BQP}$ as in Def. 9, and let $\mathsf{Unif}_{\mathcal{L}_{P}}$ be a distribution over a subset of the promise as in Def. 10. If there exists a verifiable identification algorithm \mathcal{A}_{B} for the input distribution $\mathsf{Unif}_{\mathcal{L}_{P}}$ as given in Def. 5, then $(\mathcal{L}_{P}, \mathsf{Unif}_{\mathcal{L}_{P}}) \in \mathsf{HeurBPP}^{\mathsf{NP}}$.

Proof. In this proof, we consider any training set $T = \{(x_{\ell}, y_{\ell})\}_{\ell=1}^{B}$ as a sequence of concatenated bitstrings, i.e. $T = \underbrace{x_1 y_1 x_2 y_2 ... x_B y_B}_{B(n+1) \text{ bits}}$. Let $X = \{x_{\ell}\}_{\ell=1}^{B}$ be a set of B points $x_{\ell} \in \{0,1\}^{n}$. We define the following set:

$$P(A_B, \epsilon, B) = \{(X, \alpha) \mid \exists Y \in \{0, 1\}^B, r \in \{0, 1\}^{\text{poly}(n)} \text{ s.t. } A_B(T_{X,Y}, \epsilon, r) = \alpha \},$$
 (37)

where $Y = \{y_\ell\}_{\ell=1}^B$ is a collection of labels $y_\ell \in \{0,1\}$ such that $T_{X,Y} = \{(x_\ell,y_\ell)\}_{\ell=1}^B$. We are also going to assume that for a given α and ϵ , the number of training sets for which there exists an r such that $\mathcal{A}_B(T,\epsilon,r) = \alpha$ is greater than 0 and the majority of them are dataset labeled accordingly to one concept $\tilde{\alpha}$ which is ϵ -close to α in PAC condition. The set $P(\mathcal{A}_B,\epsilon,B)$ contains then all the tuples (X,α) which satisfy the relation $R:\mathcal{A}_B(T_{X,Y},\epsilon,r) = \alpha$. Since \mathcal{A}_B runs in polynomial time (for $\epsilon \sim \frac{1}{\text{poly}(n)}$ and $B \sim \text{poly}(n)$), the relation R belongs to NP. Theorem 14 from [BGP00] guarantees the existence of an algorithm A_R such that given the NP relation $R:\mathcal{A}_B(T_{X,Y},\epsilon,r) = \alpha$ outputs on input (X,α) a tuple $(Y,r) = A_R[(X,\alpha)]$ which satisfies $\mathcal{A}_B(T_{X,Y},\epsilon,r) = \alpha$, uniformly at random among all the possible tuples (Y,r) which satisfies R.

We can now construct an algorithm \mathcal{A}' which, using a NP oracle and the algorithm A_R in [BGP00], evaluates any concept f^{α} with an average-case error of at most ϵ' under the uniform distribution over inputs $x \in \{0,1\}^n$. In other words, the algorithm \mathcal{A}' satisfies satisfies the heuristic condition in Eq. (9) with respect the target function f^{α} . Firstly, since we are considering the verifiable case, we can implement the following function which, given a dataset consistent with one of the concept, tells if the label of an input $x \in \{0,1\}^n$ is consistent with the dataset or not.

```
1: function CHECK(x, \epsilon, T)

2: if \mathcal{A}_B(\{(x, 0)\} \cup T, \epsilon, r_0) ="invalid dataset"

3: return 1

4: else return 0

5: end function
```

We can now construct the algorithm \mathcal{A}' . Fix a desired average-case error ϵ' . First consider an identification algorithm \mathcal{A}_B in Def. 5 which runs with ϵ and δ such that $\epsilon = \frac{\epsilon'}{2}(1-\delta)$. Using the algorithm \mathcal{A}_B and the algorithm R given in [BGP00], we can construct the following algorithm \mathcal{A}' which runs in polynomial time using an NP oracle.

```
1: function \mathcal{A}'(\alpha, x, \epsilon, B)

2: step 1. Sample B points X = x_1, x_2, ..., x_B from the distribution \mathsf{Unif}_{\mathcal{L}_P}.

3: step 2. (Y, r) \leftarrow A_R[(X, \alpha)]

4: step 3. Construct T_{X,Y} by assigning each label in Y to the corresponding point in X.

5: step 4. return y \leftarrow \mathsf{CHECK}(x, \epsilon, T_{X,Y}).

6: end function
```

The labels produced by \mathcal{A}' ensure that all points in $\{(x,y)\} \cup T_{X,Y}$ are consistent with at least one concept labeled by $\tilde{\alpha}$. We will now demonstrate that the concept $f^{\tilde{\alpha}}$ is, with probability greater than 2/3, heuristically close to the target α , meaning that:

$$\mathbb{E}_{x \sim \text{Unif}(\{0,1\}^n)} ||f^{\tilde{\alpha}}(x) - f^{\alpha}(x)|| \le \epsilon'.$$
(38)

In Step 2 of the algorithm \mathcal{A}' , the labeling $Y = \{y_\ell\}_{\ell=1}^B$ is chosen such that \mathcal{A}_B , given the dataset $T_{X,Y}$ and parameter ϵ , outputs α with a probability greater than 0. Specifically, the dataset is sampled uniformly at random from all possible datasets satisfying this condition. The probability that an outputted dataset $T_{X,Y}$ is consistent with a concept $f^{\tilde{\alpha}}$ that is not ϵ' -close to f^{α} (as defined by Eq. (38)) depends on both the points in X and the failure probability δ of the algorithm \mathcal{A}_B . For simplicity, we initially neglect the failure probability δ of \mathcal{A}_B . Even in this case, the dataset $T_{X,Y}$ could still be consistent with a concept $f^{\tilde{\alpha}}$

far from f^{α} if both f^{α} and $f^{\tilde{\alpha}}$ agree on all the B points in X. By setting $\epsilon = \epsilon'/2$, the fraction of points on which a concept $f^{\tilde{\alpha}}$ (not satisfying Eq. (38)) disagrees with a concept that is ϵ -close to f^{α} is at least ϵ . If at least one of those points is in X, then such $f^{\tilde{\alpha}}$ cannot be outputted by $R[P(A_B, \epsilon, B), (X, \alpha)]$ (we are still assuming the algorithm A_B has a zero failure probability, i.e. $\delta = 0$). The probability that a random x lies in a fraction ϵ of all the $x \in \{0,1\}^n$ is clearly ϵ . It follows that the probability of selecting B random points where at least one shows a disagreement between a concept that is ϵ -close to f^{α} and a concept that is more than ϵ' away from f^{α} is:

$$1 - (1 - \epsilon)^B \tag{39}$$

We want to bound this probability such that is above $\frac{2}{3}$. In order to obtain that, we need to extract a number B of points greater than:

$$(1 - \epsilon)^B \le \frac{1}{3} \tag{40}$$

$$B\log(1-\epsilon) \le -\log 3\tag{41}$$

$$-2\epsilon B \le -\log 3\tag{42}$$

$$B \ge \frac{\log 3}{2\epsilon} \tag{43}$$

Where we used the fact that for $\epsilon \leq 0.7$, $-2\epsilon \leq \log(1-\epsilon)$. Therefore, by selecting $B \sim \frac{1}{\epsilon}$ (noting that the ϵ has to scale as $\epsilon \sim \frac{1}{n}$ for the algorithm A to remain efficient), we can ensure with a probability greater than 2/3 that the dataset $T_{X,Y}$ produced in Step 2 of \mathcal{A}' is consistent only with concepts that are ϵ' -close to the target α . We now take into account the failure probability of the algorithm \mathcal{A}_B . Because of this, it is still possible that there exists a random string r_0 such that, for a dataset $T_{X,Y}$ consistent with a concept $f^{\tilde{\alpha}}$ that is more than ϵ' away from f^{α} , the algorithm $A(T_{X,Y},\epsilon,r_0)$ outputs α , regardless of the B points in the dataset. This happens with a probability δ over all the possible datasets. We can then take into account this probability of failure by asking that the probability in Eq. (39) is above $\frac{2}{3(1-\delta)}$. With this modification, the bound on the number of points B becomes:

$$B \ge \frac{\log 3 + 2\delta}{\epsilon} \tag{44}$$

Notice that for $\delta, \epsilon \approx \frac{1}{n}$ the above quantity is still polynomial in n.

For every input x, the algorithm \mathcal{A}' outputs, with probability greater than $\frac{2}{3}$, a label $y \in \{0,1\}$ that aligns with one of the concepts $f^{\tilde{\alpha}}$ satisfying Eq. (38), i.e., concepts that are ϵ' -close to f^{α} . The maximum number of points $x \in \{0,1\}^n$ that the algorithm can consistently misclassify is bounded above by ϵ' . This corresponds to the scenario where all ϵ' -close concepts disagree with α on the same subset of points.

Theorem 4 (Identification hardness for the verifiable case - uniform). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m, m = \text{poly}(n)\}$ be a concept class such that there exists a $f^{\alpha} \in \mathcal{F}$ which is the characteristic function of a language $\mathcal{L} \in \mathsf{BQP}$ as in Def. 8, and let $\mathsf{Unif}(\{0,1\}^n)$ be the uniform distribution over all the input $x \in \{0,1\}^n$. If there exists a verifiable identification algorithm \mathcal{A}_B for the input distribution Unif as given in Def. 5, then $(\mathcal{L}, \mathsf{Unif}) \in \mathsf{HeurBPP}^\mathsf{NP}$.

Proof. The proof proceeds in direct analogy to the proof of Theorem 3. The only change regards the first step in Algorithm \mathcal{A}' . Specifically, instead of sampling the points $X = x_1, x_2, ..., x_B$ from the distribution $\mathsf{Unif}_{\mathcal{L}_P}$ we now sample them from the uniform distribution over $\{0,1\}^n$.

C.3 Proofs of hardness for the identification task in the non-verifiable case

Theorem 7 (Approximate-verifiable identification implies (BQP, Unif) \subset HeurBPP^{NP}). Let $\mathcal{F} = \{f^{\alpha} : \{0,1\}^{n} \to \{0,1\}^{n} \mid \alpha \in \{0,1\}^{m}\}$ be a concept class containing at least a BQP function f as in Def. 8 associated to a language $\mathcal{L} \in \mathsf{BQP}$. If there exists an approximate-verifiable identification algorithm \mathcal{A}_{B} for the input distribution Unif as in Def. 7 then $(\mathcal{L}, \mathsf{Unif}) \in \mathsf{HeurBPP}^\mathsf{NP}$.

Proof. Let the concept $f^{\alpha} \in \mathcal{F}$ the BQP-function associated to a given BQP language. In this proof we are considering training sets $T = \{(x_{\ell}, y_{\ell})\}_{\ell=1}^{B}$ as sequences of concatenated bitstrings, i.e. $T = \underbrace{x_1y_1x_2y_2...x_By_B}_{B(n+1) \text{ bits}}$.

Firstly, let T^{α} be the set of all datasets in which 2/3 of the points are labeled accordingly to f^{α} :

$$T^{\alpha} = \{ T = \{ (x_{\ell}, y_{\ell}) \}_{\ell=1}^{B} \mid \exists T' \subset T, \ |T'| = \frac{2}{3} |T| \text{ s.t. } y_{i} = f^{\alpha}(x_{i}) \quad \forall (x_{i}, y_{i}) \in T' \}$$
 (45)

We then can define for each input $x \in \{0,1\}^n$ a set $T^{\alpha}(x)$ containing all the datasets in T^{α} where the point x appears in the first position:

$$T^{\alpha}(x) = \{ T \in T^{\alpha} \mid T = \{ (x, y), (x_2, y_2), ..., (x_B, y_B) \} \}$$

$$(46)$$

We finally define the set $T^{\alpha}_{\delta}(x)$ as the following:

$$T_{\delta}^{\alpha}(x) = \{ T = \{ (x_{\ell}, y_{\ell}) \}_{\ell=1}^{B} \mid \exists r \text{ s.t. } \mathcal{A}_{B}^{\delta}(T, r, 1/3) = \alpha \land (x_{1} = x, y_{1}) \in T \}$$

$$(47)$$

Notice that $T_0^{\alpha}(x) = T^{\alpha}(x)$.

Given the existence of an approximate-verifiable algorithm \mathcal{A}_B as in Def. 7 we construct an algorithm M^{α} which on input x and a value δ , sample one random dataset from $T^{\alpha}_{\delta}(x)$ and label x accordingly to the label of the first point in $T^{\alpha}_{\delta}(x)$.

- 1: function $M^{\alpha}(\mathcal{A}_B, x, \delta, s)$
- 2: **step 1.** Sample T from the set $T^{\alpha}_{\delta}(x)$ uniformly at random.
- 3: **step 2.** Output y_1 from $(x, y_1) \in T$
- 4: end function

We first show that the algorithm M^{α} belongs to the third level of the PH if \mathcal{A}_B runs in polynomial time. In the algorithm M^{α} the random string s determines the set $T \in T^{\alpha}_{\delta}(x)$ sampled. We then consider the following set:

$$P(\mathcal{A}_B, x) = \{ \alpha \in \{0, 1\}^m \mid \exists T \text{ s.t. } T \in T_\delta^\alpha(x) \}, \tag{48}$$

The set $P(\mathcal{A}_B,x)$ contains all α for which there exists a dataset T and a random string r such that the identification algorithm \mathcal{A}_B in Def. 7 outputs α , i.e. $\mathcal{A}_B(T,1/3,r)=\alpha$, and x appears as the first element of T. Deciding if an α belongs to $P(\mathcal{A}_B,x)$ can be done in NP^{NP}. The reason for that is that within NP it is possible to decide if a given T belongs to $T^{\alpha}_{\delta}(x)$ since the algorithm $\mathcal{A}_B(T,r,1/3)$ runs in polynomial time. Therefore, the condition $T\in T^{\alpha}_{\delta}(x)$, which ensures that $\alpha\in P(\mathcal{A}_B,x)$, can be verified in polynomial time with the help of an NP oracle. We can then use Theorem 14 from [BGP00] that guarantees the existence of an algorithm in BPP^{NP} such that given any NP relation outputs a witness for a given x uniformly at random. In particular, we apply Theorem 14 to sample a training set T from the set $T^{\alpha}_{\delta}(x)$ which can be regarded as witnesses for the relation $\alpha\in P(\mathcal{A}_B,x)$. This allows to perform the first step of the algorithm M^{α} in BPP^{NPNP}. We have then proved that the algorithm M^{α} can be run in the third level of the PH.

We now show that by choosing an adequate value of δ , the algorithm $M(A_B, x, \delta, s)$ can correctly evaluate the BQP function f^{α} on a large fraction of input points. In particular, we obtain that for any $\delta' \geq 0$, linearly proportional to δ , the algorithm M^{α} is such that:

$$\Pr_{x \sim \mathsf{Unif}\{0,1\}^n} \left[\Pr_s[M^{\alpha}(\mathcal{A}_B, x, \delta, s) \neq f^{\alpha}(x)] \geq \frac{2}{5} \right] \leq \delta'.$$

In other words, the fraction of points $x \in \{0,1\}^n$ for which algorithm M^{α} misclassifies with probability greater than 2/5 is less than δ' . Let us consider the set $T^{\alpha}_{\delta}(x)$ for a given x. If $\delta = 0$ and for any dataset in T^{α} there exists a random string r such that $\mathcal{A}_B(T, r, 1/3) = \alpha$, then it would follow that:

$$\forall x \in \{0,1\}^n \ \Pr_s[M^{\alpha}(A_B, x, 0, s) \neq f^{\alpha}(x)] > 1/3$$

as for every x the set $T_0^{\alpha}(x)$ contains all the possible datasets where at least $\frac{2}{3}$ of the points are labeled by f^{α} . The failure probability δ in the completeness property of Def. 7 implies that the identification algorithm \mathcal{A}_B will output α only on a fraction $1-\delta$ of all the possible dataset in T^{α} . We then bound the number of inputs x for which more than a fraction $\frac{2}{5}$ of the datasets in $T^{\alpha}_{\delta}(x)$ assign the wrong label to x. For any given x, the number of datasets in $T^{\alpha}(x)$ which classify correctly x is at least $\frac{2}{3}|T^{\alpha}(x)|$. In order to lower the fraction of datasets correctly labeling x to less than $\frac{3}{5}$, it is necessary that at least $\frac{1}{6}$ of them is not present in $|T^{\alpha}_{\delta}(x)|$. This means that on a fraction $\frac{1}{6}$ of all the dataset in $T^{\alpha}(x)$ there does not exist a random string for which the identification algorithm \mathcal{A}_B outputs α even though there is a fraction $\frac{2}{3}$ of the points labeled by f^{α} . Notice that $\sum_{x} |T^{\alpha}(x)| = |T^{\alpha}|$ and $|T^{\alpha}| \propto N^B$ for a dataset containing B points out of a total of N input points $(N=2^n)$ if we consider all the $x \in \{0,1\}^n$ as input). Then for each x we have that $|T^{\alpha}(x)| \propto N^{B-1}$ and it has to hold that $\sum_{x} |T^{\alpha}_{\delta}(x)| \geq (1-\delta)|T^{\alpha}|$. We can then bound the number x_{wrong} of x's for which $|T_{\delta}^{\alpha}(x)| \leq \frac{5}{6}|T^{\alpha}(x)|$ and thus potentially allow for a probability to be misclassified higher than $\frac{2}{5}$:

$$\sum_{x} \frac{1}{6} |T^{\alpha}(x)| \le \delta |T^{\alpha}| \tag{49}$$

$$\sum_{x} \frac{1}{6} |T^{\alpha}(x)| \le \delta |T^{\alpha}|$$

$$\frac{1}{6} \sum_{x} N^{B-1} \le \delta N^{B}$$

$$(50)$$

$$\frac{1}{6}N^{B-1} \cdot x_{wrong} \le \delta N^B \tag{51}$$

$$x_{wrong} \le 6\delta \cdot N. \tag{52}$$

Therefore, the maximum fraction of x for which $\Pr_s[M^{\alpha}(A_B, x, \delta, s) \neq f^{\alpha}(x)] \geq 2/5$ is upper bounded by $\delta' = 6\delta$. When x comes from the uniform distribution over the set of allowed inputs, this means that:

$$\Pr\nolimits_{x \sim \mathsf{Unif}\{0,1\}^n} \left[\Pr\nolimits_s[M^\alpha(\mathcal{A}_B, x, \delta, s) \neq f^\alpha(x)] \geq \frac{2}{5} \right] \leq \delta'$$

and thus $(\mathcal{L}, \mathsf{Unif}) = \mathsf{HeurBPP}^{\mathsf{NP}^{\mathsf{NP}}}$.

Theorem 8 (Approximate-verifiable identification implies (BQP, Unif \mathcal{L}_P) \subset HeurBPP^{NPNP}). Let $\mathcal{F} = \{f^{\alpha} : f^{\alpha} : f^{\alpha$ $\{0,1\}^n \to \{0,1\} \mid \alpha \in \{0,1\}^m\}$ be a concept class containing at least a PromiseBQP function f as in Def. 9 associated to a language $\mathcal{L}_P \in \mathsf{PromiseBQP}$ and let $\mathsf{Unif}_{\mathcal{L}_P}$ be an associated input distribution as in Def. 10. If there exists an approximate-verifiable identification algorithm \mathcal{A}_B for the input distribution $\mathsf{Unif}_{\mathcal{L}_P}$ as in $\textit{Def. 7 then } (\mathcal{L}_{P}, \mathsf{Unif}_{\mathcal{L}_{P}}) \in \mathsf{HeurBPP}^{\mathsf{NP}^{\mathsf{NP}}}$

Proof. The argument proceeds in direct analogy to the proof of Theorem 7, the only change regards the sets of trainingsets considered. In particular, we impose that the points x_{ℓ} in the training sets T^{α} , $T^{\alpha}(x)$ and $T_{\delta}^{\alpha}(x)$ of Eqs. (45-46-47) comes from the input distribution Unif_{\mathcal{L}_P}. Since we assume efficient classical samplability of $\mathsf{Unif}_{\mathcal{L}_P}$ in Def. 10, this condition can be enforced within polynomial time. The rest of the proof follow the same steps as the proof of Theorem 7.

Theorem 9. If there exists an approximate-correct identification algorithm A_B as in Def. 14 for a cdistinct concept class \mathcal{F} , with $c \geq 1/3$, which works correctly for a dataset of size B, then there also exists an approximate-verifiable algorithm \mathcal{A}'_{3B} as in Def. 7 in the second level of the PH which works correctly for a dataset of size 3B.

Proof. The proof works by explicitly constructing an algorithm \mathcal{A}'_{3B} that is within the second level of the PH and that, given access to the approximate-correct algorithm A_B in Def. 14, satisfies the condition in Def. 7 of approximate-verifiable identification algorithm. In particular, the main objective of the proof is to show how an approximate-correct identification algorithm in Def. 14 can be used to construct an algorithm in the second level of the PH which satisfies the completeness condition of Def. 7. We first present the main steps of the algorithm, then show that they can be performed within the PH and finally check the correctness of the algorithm.

First we notice the following. If there exists a learning algorithm \mathcal{A}_B which solves the identification task in Def. 14 for the concept class \mathcal{F} , then we can easily construct an algorithm $\bar{\mathcal{A}}_B$ that can solve the identification task even for the concept class $\bar{\mathcal{F}}$ defined as follows:

$$\bar{\mathcal{F}} = \{ f^{\bar{\alpha}} : \{0, 1\}^n \to \{0, 1\} \mid \forall x \in \{0, 1\}^n \ f^{\bar{\alpha}}(x) \oplus 1 = f^{\alpha}(x) \in \mathcal{F} \}$$
 (53)

Specifically, the algorithm $\bar{\mathcal{A}}_B$ solves the identification task for the concept class $\bar{\mathcal{F}}$ by flipping the labels of all points in the input dataset before applying \mathcal{A}_B .

We are now ready to describe the algorithm \mathcal{A}'_{3B} . We notice that having access to the algorithm \mathcal{A}_B in Def. 14 we can implement the following function which, given a dataset $T = \{(x_\ell, y_\ell)\}_{\ell=1}^{3B}$ of 3B points and a label α corresponding to one of the concept, detects if there is a subset T_B of B of points not consistent with α . Consider an algorithm \mathcal{A}_B as in Def. 14 and let ϵ_c be $\epsilon_c = \min(\epsilon_1, \epsilon_2)$, with ϵ_1 and ϵ_2 the error values in the conditions of Eq. (10) and (11) of Def. 14. Then we define the check function as follows.

```
1: function CHECK(\alpha, T)

2: for T_B in T

3: for r in R

4: if \bar{\mathcal{A}}_B(T_B, \epsilon_c, r) = \bar{\alpha} return "invalid dataset"

5: end function
```

The function check can be implemented by a polynomial time algorithm running in the second level of the PH. In order to prove it, consider the following set:

$$S_1(\alpha) = \{ T = \{ (x_\ell, y_\ell) \}_{\ell=1}^{3B} \mid \exists r \in R, T_B = \{ (x_i, y_i) \}_{i=1}^B, T_B \subset T \text{ s.t. } \bar{\mathcal{A}}_B(T_B, \epsilon_c, r) = \bar{\alpha} \}$$
 (54)

The set $S_1(\alpha)$ contains all the datasets T for which there exists at least one subset of B points T_B and a random string $r \in R$ such that $\bar{\mathcal{A}}_B(T_B, \epsilon_c, r) = \bar{\alpha}$. Note that for the property of Eq. (10) in Def. 14 of approximate-correct identification algorithm, a dataset of 3B points entirely consistent with α is not contained in $S_1(\alpha)$. On a given input dataset $T = \{(x_\ell, y_\ell)\}_{\ell=1}^{3B}$, the algorithm check decides if the dataset T belongs to $S_1(\alpha)$ or not. Deciding if a set T belongs to $S_1(\alpha)$ can be done in NP as a YES instance can be verified in polynomial time if \mathcal{A}_B runs in polynomial time.

We construct the algorithm \mathcal{A}'_{3B} as follows.

```
1: function \mathcal{A}'(T, \epsilon, r)

2: \alpha \leftarrow \mathcal{A}_B(T, \epsilon, r) > with \mathcal{A}_B the approximate-correct identification algorithm in Def. 14

3: if \operatorname{check}(\alpha, T) = \text{``invalid dataset''}

4: return ``invalid dataset''

5: else return \alpha

6: end function
```

We now show that, for c-distinct concept classes with $c \geq 1/3$, the algorithm \mathcal{A}'_{3B} described above rejects any dataset $T = \{(x_i, y_i)\}_{i=1}^{3B}$ with B or more points not labeled by f^{α} . Thus, the constructed \mathcal{A}'_{3B} satisfies the soundness condition of the approximate-verifiable algorithm as in Def. 7. First, if there exists a subset $T_B \subset T$ such that every point in it is labeled by $f^{\bar{\alpha}}$, then there exists a random string r and a value $\epsilon \leq \epsilon_2$ such that $\bar{\mathcal{A}}_B(T_B, \epsilon, r) = \bar{\alpha}'$ with the condition $\mathbb{E}_{x \sim \mathsf{Unif}(\{0,1\}^n)} |f^{\bar{\alpha}}(x) - f^{\bar{\alpha}'}(x)| < \frac{1}{3}$. However, since by assumption all the concepts differ one from each other on more than 1/3 of the points, then $\bar{\mathcal{A}}_B(T_B, \epsilon, r) = \bar{\alpha}$. It follows that the function check will successfully detect and discard any input dataset that contains B or

more points not labeled by α . Therefore, for any input dataset of 3B or more points the algorithm \mathcal{A}'_{3B} satisfies the conditions in Def. 7 of approximate-verifiable algorithm.

Theorem 10. If there exists an approximate-correct identification algorithm A_B as in Def. 14 for an average-case-smooth concept class \mathcal{F} , which works correctly for a dataset of size B, then there also exists an approximate-verifiable algorithm A'_{3B} as in Def. 7 in the second level of the PH which works correctly for a dataset of size 3B.

Proof. The proof is analogous to the one of Theorem 9. In particular we again leverage the existence of the algorithm $\bar{\mathcal{A}}_B$ to check if a dataset $T=\{(x_i,y_i)\}_{i=1}^{3B}$ of 3B points contains a fraction of B points not consistent with a previously outputted α . Since now the concept class satisfies the condition in Def. 18, in order to check if a set T_B of B points from T are labeled by $f^{\bar{\alpha}}$ is sufficient to check if $d(\bar{\mathcal{A}}_B(T_B, \epsilon, r), \bar{\alpha}) \leq 1/3$, where d is the distance function in Def 18. By the second property in Def. 14, if all the points in T_B are consistent with $f^{\bar{\alpha}}$, then there exists a r such that $d(\bar{\mathcal{A}}_B(T_B, \epsilon, r), \bar{\alpha}) \leq 1/3$ for any $\epsilon \leq \epsilon_2$ in Def. 14. The proof follows the same structure as that of Theorem 9, with the only change being in the check function, which is now defined as follows.

```
1: function CHECK(\alpha, T)

2: for T_B in T

3: for r in R

4: if d(\bar{\mathcal{A}}_B(T_B, \epsilon_c, r), \bar{\alpha}) \leq 1/3 return "invalid dataset"

5: end function
```

D Proof of Theorem 11

Theorem 11. There exists a family of unitaries $\{U(x)\}_x$ and observables $\{O(\alpha)\}_{\alpha}$ such that the corresponding concept class $\mathcal{F}^{U,O}$ is not classically identifiable by an approximate-correct algorithm as in Def. 14, but is identifiable by a quantum learning algorithm.

Proof. Let us first introduce the set of 3n classical functions $\{f_j\}_{j=1}^{3n}$ defined as follows:

$$f_j(x) = \begin{cases} f_j(x) = 0 \text{ if } x \notin \mathcal{L}_j \\ f_j(x) = 1 \text{ if } x \in \mathcal{L}_j \end{cases}$$
 (55)

Where $\mathcal{L}_j = \{x_j, x_{j+1}, \dots, x_{j+2^n/3n}\}$ denotes the j-th block, a contiguous subset of $\frac{2^n}{3n}$ elements selected from the set of all possible $x \in \{0, 1\}^n$, according to a predefined order. We then consider the Reed-Solomon code [WB99] RS(3|k|,|k|) defined over a field with $q=2^m$ elements which maps a message k of length |k| into a codeword of length 3|k|. Let us consider m and |k| such that $n=|k|\cdot m$. In this way the minimum distance between two different codewords is lower bounded by $d \geq 3|k|-k+1 \geq 2|k|+1 \geq |k|$. This corresponds to a Hamming distance greater than $d_H \geq |k| \cdot m$ between the bitstring representation of codewords corresponding to two different messages k and k'. Next, we define the function $g: \{0,1\}^n \rightarrow \{0,1\}^{3n}$ as the function which maps the bitstring representation of the message k into the bitstring representation of its codeword of the previously introduced RS(3|k|,|k|) code. In this way, $d_H(g(k),g(k')) \geq n$ for every $k \neq k'$, where $d_H(g(k),g(k'))$ stands for the Hamming distance between the 3n-bits given by the function g. Consider now the family of 2^n functions $\{h_k\}_{k=1}^{2^n}$ defined as:

$$h_k(x) = \sum_{j=1}^{3n} [g(k)]_j \cdot f_j(x)$$
 (56)

It is clear that for different k each function h_k will differ from the others on at least $\frac{2^n}{3}$ distinct values of $x \in \{0,1\}^n$. We now want to implement the functions h_k in the form of the concepts in $\mathcal{F}^{H,O}$. We can do so by considering a register of $3n \times n$ qubits. On each of the 3n registers the function f_j in Eq. (55) is implemented by measuring the Z observable on the first qubit after having prepared a corresponding state $\rho_j(x)$. In particular the unitary U is the tensor product of each U_j acting on the j-th register of n qubits, with j = 1, 2, ..., 3n. Each U_j is such that:

$$f_j(x) = \langle x | U_j^{\dagger} Z_1 U_j | x \rangle \tag{57}$$

where Z_1 represent the Z Pauli operator acting on the first qubit of the j-th register. We note that since classical computations are in BQP such a unitary always exists. Let us consider then the following circuit composed of 3n + 1 register of n qubit each. On the first register the input dependent state $|x\rangle$ undergoes an arbitrary BQP computation and the observable Z is measured on the first qubit. The other registers implement the functions f_j as described above. As they are classical functions, they can also be implemented quantumly. Next we define the required set of observables $\{O(\alpha)\}_{\alpha}$ defined as:

$$O(\boldsymbol{\alpha}) = Z_1 \otimes \sum_{j=1}^{3n} \alpha_j Z_{nj+1}, \tag{58}$$

where Z_i represents the Pauli string consisting of identity operators on all qubits except for the Z matrix acting on the i-th qubit. Each concept in $f^{\alpha} \in \mathcal{F}$ is now expressed as $f^{\alpha} = \text{Tr}[\rho(x)O(\alpha)]$, where $\rho(x)$ represents the quantum state formed by the tensor product of the previously described 3n+1 registers, and $O(\alpha)$ is defined as in Eq. (58). Specifically, $\rho(x)$ is defined as follows:

$$\rho(\boldsymbol{x}) = \bigotimes_{j=1}^{3n+1} \rho_j(x), \quad \rho_j = \begin{cases} \text{if } j = 1 \colon \rho_1(\boldsymbol{x}) = |\psi\rangle \langle \psi|_1 \text{ with } |\psi\rangle = U_{BQP} |\boldsymbol{x}\rangle \\ \text{if } j \neq 1 \colon \rho_j(\boldsymbol{x}) = |\psi\rangle \langle \psi|_j \text{ with } |\psi\rangle = U_j |\boldsymbol{x}\rangle \end{cases}$$
(59)

where U_{BQP} represents any arbitrary BQP computation while $\{U_j\}_j$ are the unitaries associated to the functions f_j as defined in Eq. (57).

If we now consider the concept class

$$\mathcal{F} = \{ f^{\alpha} \mid f^{\alpha} = \text{Tr}[\rho(\mathbf{x})O(\alpha)], \ \alpha = q(k) \ \forall k \in \{0,1\}^n \},$$
(60)

where the functions f^{α} and g are defined as above, we obtain a concept class \mathcal{F} such that for any pair of different α_1, α_2 the corresponding concepts differ on at least $2^n/3$ different points $x \in \{0,1\}^n$. Therefore the constructed concept class is a c-distinct concept class as in Def. 15, with c = 1/3. We can then apply our Theorem 9 and construct an approximate-verifiable algorithm from an approximate-correct identification algorithm as in Def. 14 which correctly solves the identification task for \mathcal{F} by using an NP oracle. Moreover, as there is at least a PromiseBQP complete concept for $\alpha = \mathbf{0}^{16}$, for Theorem 7 if there exists an approximate-verifiable algorithm which runs in the second level of the polynomial hierarchy then BQP would be contained in the fourth level of the PH. We can therefore conclude that the constructed concept class \mathcal{F} is not learnable even in the identification sense with an approximate algorithm which satisfies Def. 14.

References

- [AA11] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 333–342, New York, NY, USA, 2011. Association for Computing Machinery.
- [AAKS20] Anurag Anshu, Srinivasan Arunachalam, Tomotaka Kuwahara, and Mehdi Soleimanifar. Sample-efficient learning of quantum many-body systems. In 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pages 685–691. IEEE, 2020.

¹⁶We do note that, even for most values of $\alpha \neq 0$, we still expect the corresponding concepts to be PromiseBQP-complete.

- [Aar10] Scott Aaronson. Bqp and the polynomial hierarchy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 141–150, 2010.
- [AB09] Sanjeev Arora and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009.
- [AS06] Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 4(05):883–898, 2006.
- [Bab16] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697, 2016.
- [Bal15] Maria-Florina Balcan. Lecture notes in foundations of machine learning and data science, September 2015.
- [BCG⁺24] Sergey Bravyi, Anirban Chowdhury, David Gosset, Vojt ěch Havlíček, and Guanyu Zhu. Quantum complexity of the kronecker coefficients. *PRX Quantum*, 5:010329, Feb 2024.
- [BGP00] Mihir Bellare, Oded Goldreich, and Erez Petrank. Uniform generation of np-witnesses using an np-oracle. *Information and Computation*, 163(2):510–526, 2000.
- [BGV⁺24] Alice Barthe, Michele Grossi, Sofia Vallecorsa, Jordi Tura, and Vedran Dunjko. Parameterized quantum circuits as universal generative models for continuous multivariate distributions, 2024.
- [BT⁺06] Andrej Bogdanov, Luca Trevisan, et al. Average-case complexity. Foundations and Trends® in Theoretical Computer Science, 2(1):1–106, 2006.
- [Fey85] Richard Feynman. Quantum mechanical computers. Optics news, 11(2):11–20, 1985.
- [GD23] Casper Gyurik and Vedran Dunjko. Exponential separations between classical and quantum learners. arXiv preprint arXiv:2306.16028, 2023.
- [GGC18] Carlos E. González-Guillén and Toby S. Cubitt. History-state hamiltonians are critical, 2018.
- [HBM+21] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *Nature communications*, 12(1):2631, 2021.
- [HKT⁺22] Hsin-Yuan Huang, Richard Kueng, Giacomo Torlai, Victor V Albert, and John Preskill. Provably efficient machine learning for quantum many-body problems. *Science*, 377(6613):eabk3333, 2022.
- [HKT24] Jeongwan Haah, Robin Kothari, and Ewin Tang. Learning quantum hamiltonians from high-temperature gibbs states and real-time evolutions. *Nature Physics*, pages 1–5, 2024.
- [JGM⁺24] Sofiene Jerbi, Casper Gyurik, Simon C Marshall, Riccardo Molteni, and Vedran Dunjko. Shadows of quantum machine learning. *Nature Communications*, 15(1):5676, 2024.
- [KSV02] Alexei Yu Kitaev, Alexander Shen, and Mikhail N Vyalyi. Classical and quantum computation. Number 47. American Mathematical Soc., 2002.
- [KV94a] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.
- [KV94b] Michael J Kearns and Umesh Vazirani. An introduction to computational learning theory. MIT press, 1994.
- [MGD24a] Simon Marshall, Casper Gyurik, and Vedran Dunjko. On bounded advice classes. arXiv preprint arXiv:2405.18155, 2024.

- [MGD24b] Riccardo Molteni, Casper Gyurik, and Vedran Dunjko. Exponential quantum advantages in learning quantum observables from classical data. arXiv preprint arXiv:2405.02027, 2024.
- [Moh18] Mehryar Mohri. Foundations of machine learning, 2018.
- [RAG21] Jonathan Romero and Alán Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *Advanced Quantum Technologies*, 4(1):2000003, 2021.
- [RT22] Ran Raz and Avishay Tal. Oracle separation of bqp and ph. ACM Journal of the ACM (JACM), 69(4):1-21, 2022.
- [Sch90] Robert E Schapire. The strength of weak learnability. Machine learning, 5:197–227, 1990.
- [Sch08] Rob Schapire. Lecture notes in theoretical machine learning, February 2008.
- [SKS⁺24] Kevin Shen, Andrii Kurkin, Adrián Pérez Salinas, Elvira Shishenina, Vedran Dunjko, and Hao Wang. Shadow-frugal expectation-value-sampling variational quantum generative model, 2024.
- [WB99] Stephen B Wicker and Vijay K Bhargava. Reed-Solomon codes and their applications. John Wiley & Sons, 1999.
- [YIM23] Hayata Yamasaki, Natsuto Isogai, and Mio Murao. Advantage of quantum machine learning from general computational advantages. arXiv preprint arXiv:2312.03057, 2023.