Learning to create is as hard as learning to appreciate

David Xiao [*]

February 26, 2010

Abstract

We explore the relationship between a natural notion of "learning to create" (LTC) studied by Kearns et al. (STOC '94) and the standard PAC model of Valiant (CACM '84), which can be thought of as a formalization of "learning to appreciate". Our main theorem states that "if learning to appreciate is hard, then so is learning to create". More formally, we prove that if there exists a concept class for which PAC learning with respect to efficiently samplable input distributions is hard, then there exists another (possibly richer) concept class for which the LTC problem is hard. We also show that our theorem is tight in two senses, by proving that there exist concrete concept classes for which PAC learning is hard but LTC is easy, and by showing that it is unlikely our main theorem can be improved to the case of PAC learning with respect to unsamplable input distributions.

1 Introduction

The **P** vs. **NP** question is often cast in the intuitively appealing language of "creativity" and whether "creativity can be automated" (see *e.g.* the survey of Wigderson [27]). To explain this view, one often uses as an analogy a great artist, say Beethoven, who produced widely appreciated works of music. We model the process of deciding whether a piece of music is pleasing by an efficient circuit f_{music}. Then the process of creating pleasing music amounts to finding a satisfying assignment to f_{music}, while appreciating music only requires evaluating f_{music} on a given input.[1] Therefore, if $\mathbf{P} = \mathbf{NP}$, one can automate the task of composing pleasing music because there would be an efficient algorithm that found pleasing pieces of music (*i.e.* the satisfying assignments of f_{music}).

The above analogy is not the only way one can view creativity through a computational lens. In this paper we explore the question of automating creativity from a learning-theoretic point of view. We will explore two models of learning that correspond to "learning to appreciate" and "learning to create". Both the studied models are standard: the first is the PAC model of Valiant [26], while the second is a form of unsupervised learning, which we call "learning to create" (LTC), whose complexity-theoretic study was initiated by Kearns et al. [15]. We prove that with respect to these models, again it holds that learning to create is as hard as (if not harder than) learning to appreciate.

[*]LRI, Université Paris-Sud, `dxiao@lri.fr`

[1]Of course the analogy is not entirely accurate since we believe that $\mathbf{P} \neq \mathbf{NP}$ while, presumably, Beethoven was bound by the Extended Church-Turing Hypothesis and could not solve **NP**-hard problems. But let us ignore this detail and suppose that Beethoven's creativity was indeed the result of solving an **NP**-hard problem.

In the PAC model, the learning algorithm is given examples labelled according to a hidden function f and is supposed to learn how to label new examples as f would. For example, one can think of f as taking input a piece of music, and outputting whether that piece of music is pleasing or not. In the LTC model, the learning algorithm is given many unlabelled examples drawn from a hidden distribution D, and is supposed to construct a circuit that generates new examples that are distributed close to D. One can think of D as say generating a piece of music as Beethoven would.

In the **P** vs. **NP** analogy, the task of appreciating (evaluating SAT formulas) is trivially efficient, while the task of creating (guessing satisfying solutions) is believed inefficient. In our interpretation of appreciating and creating, both appreciating and creating are efficient (*i.e.* computable by polynomial-size circuits), but it is up to the learning algorithm to discover these circuits. For example, each person has a different music appreciation function, and $f^{\mathsf{Alice}}_{\mathsf{music}}$ may be very different from $f^{\mathsf{Bob}}_{\mathsf{music}}$. The goal of the learning algoithm is to deduce from a set of examples labelled by Alice how to appreciate music in the same way as Alice. Similarly, the distribution of music produced by Beethoven is very different from the distribution produced by John Lennon, and the learning algorithm should learn using examples how to produce music according to one style or the other.

To be more precise in our discussion, we briefly describe these two models, though we defer formal definitions to Section 2. In the PAC model, the learning algorithm A is asked to learn a concept class $\mathcal{F}$ of Boolean functions. A is given a polynomial-size set of labelled examples $S = \{(x_1, y_1), \ldots, (x_t, y_t)\}$ such that each of the x_i is drawn from a fixed distribution X, and there exists $f \in \mathcal{F}$ (unknown to A) such that $\forall i \in [t],\ f(x_i) = y_i$. A should output a hypothesis h such that $\Pr[f(X) \neq h(X)]$ is small.

In the LTC model of Kearns et al. [15], A is asked to learn a concept class $\mathcal{F}$ of functions $f : \{0,1\}^m \to \{0,1\}^n$. Let $f(U_m)$ be the distribution of outputs of f given uniform input. A is given a polynomial-size set of examples $S = \{f(x_1), \ldots, f(x_t)\}$ where each of the x_i are uniform (notice that A only gets $f(x_i)$ and not the x_i themselves). A should output a hypothesis $h : \{0,1\}^{m'} \to \{0,1\}^n$ such that the distribution $h(U_{m'})$ is close to $f(U_m)$ (say in total variation distance).

In contrast to the **P** vs. **NP** question where the notion of "appreciating" is defined to be **P** and therefore efficient, neither PAC learning nor LTC are known to be efficiently solvable for complete concept classes (*e.g.* if $\mathcal{F} = \mathsf{SIZE}(n^2)$ is the set of all functions computable by size n^2 Boolean circuits). Our first result is the following:

Theorem 1.1 (LTC is as hard as PAC learning)**.** *If there is an algorithm A that learns to create,* i.e. *that efficiently solves LTC for the concept class $\mathsf{SIZE}(n^2)$, then there is an algorithm A' that learns to appreciate,* i.e. *A' solves PAC for the concept class $\mathsf{SIZE}(n^2)$ with respect to all efficiently samplable input distributions.*

Contrapositively, if PAC learning w.r.t. efficiently samplable input distributions is hard, then LTC is also hard. Our results hold even for the more stringent requirement of agnostic learning.

In contrast to Theorem 1.1, which is concerned with the complete concept class $\mathsf{SIZE}(n^2)$, when considering concrete concept classes that are not complete, the situation may be reversed.

Theorem 1.2 (PAC vs. LTC for incomplete concept classes, informal)**.** *Under standard cryptographic assumptions, there exist concept classes for which PAC learning (even with respect to the uniform input distribution) is hard while LTC is easy.*

Therefore it is not possible to prove a version of Theorem 1.1 that says given an algorithm solving LTC for a class $\mathcal{F}$, there is an algorithm that PAC learns for the same class $\mathcal{F}$.[2]

Finally we study whether Theorem 1.1 can be generalized to encompass PAC learning with respect to *unsamplable* input distributions. We show this is unlikely: we exhibit an oracle relative to which LTC is easy (and therefore PAC learning is easy for all efficiently samplable input distributions), but there exist functions that are hard to learn with respect to an unsamplable distribution.

Theorem 1.3 (Separating LTC and PAC learning unsamplable input distributions)**.** *There exists an oracle $\mathcal{O}$ relative to which solving LTC for $\mathsf{SIZE}^{\mathcal{O}}(n^2)$ is easy, while there is a function $f \in \mathsf{SIZE}^{\mathcal{O}}(n^2)$ that is an efficiently computable function that is hard to PAC learn on an unsamplable input distribution.*

1.1 Relation to previous work

Previous work on complexity of LTC The computational complexity of PAC learning has been studied extensively since its first appearance [26, 14, 21, 16, 4]. The complexity of LTC was first studied in [15] but overall is less well-understood. One question about LTC that has received some attention is its relation to a related notion of "learning to evaluate probabilities": given samples as in the LTC problem, construct a hypothesis $h : \{0,1\}^n \to [0,1]$ such that $h(x) = \Pr[D = x]$, *i.e.* the hypothesis evaluates the probability that x is drawn from D. It is known there is a concrete concept class for which "learning to evaluate probabilities" is hard and LTC is easy [15], while there is also a concept class for which "learning to evaluate probabilities" is easy yet LTC is hard [19].

Complexity of learning via complete problems Much of the literature comparing different learning models has focused on *concrete* problems. In this kind of comparison, one exhibits a single concept class that is learnable in one model but not in another (under some reasonable complexity assumption). This has led to valuable insights into the complexity of various models, including the power of membership oracles [5], faulty vs. perfect membership oracles [24, 6], and the aforementioned difference between learning to evaluate probabilities and LTC [15, 19].

On the other hand, this approach has the drawback that it can lead to conflicting evidence, as in the case of learning to evaluate probabilities and LTC, where looking at one concept class suggests that the learning in the first model is harder than in the second model, but looking at a different concept class suggests the opposite.

A different approach to understanding the complexity of the learning model is to examine a *complete problem* for the model. Namely, if we consider the problem of learning a universal concept class (*e.g.* polynomial-size circuits $\mathsf{SIZE}(n^c)$) in learning model M, then the existence of any hard-to-learn concept class for M would imply that learning $\mathsf{SIZE}(n^c)$ is hard. This approach was explored in Applebaum et al. [1], Xiao [28, 29] to understand the complexity of PAC learning relative to the complexity of **NP**, auxiliary-input one-way functions [20], and zero knowledge. In this paper we apply this approach to the complete problems for PAC learning and for LTC.

[2]Since LTC deals with classes of functions with multi-bit outputs while PAC deals with classes of functions with single-bit outputs, it must be clarified how we obtain both single- and multi-bit functions from a single class $\mathcal{F}$. We defer this discussion to Section 4

PAC learning with respect to efficient distributions We already noted that Theorem 1.1 only relates LTC to PAC learning or agnostic learning with respect to efficiently samplable input distributions. We believe that this is a reasonable restriction: according to the strong Church-Turing thesis, all physical phenomena can be explained by efficient (polynomial-time) computation. Therefore, one can suppose that from whatever source one obtains the examples to be learned, if one can suppose that they are i.i.d. samples from a distribution then one may as well suppose that this distribution is polynomial-time samplable.

Furthermore, Theorem 1.3 says that no black-box (nor relativizing) reduction can strengthen Theorem 1.1 to include PAC learning (without the restriction to efficiently samplable distributions). Theorems such as Theorem 1.3 are called *oracle separations*, and have long been used in theoretical computer science to "separate" various classes. Baker et al. [2] showed an oracle separation between **P** and **NP** and Impagliazzo and Rudich [13] showed an oracle separation between one-way permutations and the intuitively harder task of key exchange. More recently, such oracle separations were also used in learning theory to separate PAC learning from the hardness of zero knowledge [28].

The motivation behind such oracle separations is as follows. There are very few unconditional separations in theoretical computer science (and almost non-existent when going beyond weak complexity classes such as $\mathsf{AC0}$). On the other hand, many if not most complexity results are proved using relativizing techniques (for example black-box reductions and diagonalization). Therefore by proving an oracle separation between classes A and B, one shows that in order to prove A reduces to B, one would need to come up with a non-relativizing technique. This arguably attests to the difficulty of the task.[3] In this spirit, we interpret Theorem 1.3 to mean that strengthening Theorem 1.1 to encompass PAC learning even with respect to unsamplable distributions would require new non-relativizing techniques.

2 Preliminaries

2.1 Notation and basic lemmas

If X is a probability distribution, then let $\mathrm{supp}(X)$ denote the support of X, *i.e.* the values that X takes on with positive probability. For two distributions X_1, X_2, the total variation distance (or statistical distance) is defined as $\Delta(X_1, X_2) = \frac{1}{2}\sum_x |\Pr[X_1 = x] - \Pr[X_2 = x]|$, where the sum is taken over all x in the supports of X_1, X_2. For finite sets S, we will sometimes abuse notation and let S also stand for the uniform distribution over S. For a circuit $C : \{0,1\}^m \to \{0,1\}^n$, we say that C samples a distribution X if $C(U_m) = X$. We let $\mathsf{SIZE}(q(n))$ denote the set of (functions computable by) circuits of size $q(n)$, and we let $\mathsf{SIZE}^{\mathcal{O}}(q(n))$ denote the same where the circuits are allowed oracle gates $\mathcal{O}$ at unit cost.

We will use the following lemma of Borel-Cantelli, which says that if a countable sequence of events each have small probability of occuring, then the probability that an infinite number of them occurs is 0.

Theorem 2.1 (Borel-Cantelli lemma). *Let $\{E_n\}_{n\in\mathbb{N}}$ be a sequence of events. Suppose $\sum_{n=1}^{\infty}\Pr[E_n]$ exists and is finite. Then $\Pr[\exists I \subseteq \mathbb{N}, |I| = \infty,\ \forall n \in I,\ E_n \text{ occurs}] = 0$.*

[3] Of course, such a result does *not* imply that the task is impossible. Indeed, many of the most surprising results bypass oracle separations or other notions of separation (for example [18, 23, 3]).

2.2 PAC learning (or learning to appreciate)

Define the learning error of a function $f : \{0,1\}^n \to \{0,1\}$ with respect to a distribution (X, Y) over $\{0,1\}^{n+1}$ to be $\mathsf{err}((X,Y), f) = \Pr_{X,Y}[f(X) \neq Y]$. For a class of functions $\mathcal{F}$, define $\mathsf{err}((X,Y), \mathcal{F}) = \min_{f \in \mathcal{F}} \mathsf{err}((X,Y), f)$.

Definition 2.2 (PAC Learning)**.** An algorithm A PAC learns the concept class $\mathcal{F}$ if the following holds for every $n \in \mathbb{N}, \varepsilon > 0$, $f \in \mathcal{F}$, and every distribution X over $\{0,1\}^n$. Given access to an example oracle that generates labelled examples according to $(X, f(X))$, A produces with success probability $\geq 1 - 2^{-n}$ an ε-good hypothesis h (represented as a circuit), namely $\mathsf{err}((X, f(X)), h) \leq \varepsilon$. Furthermore, A runs in time $\mathrm{poly}(n, 1/\varepsilon)$.

Definition 2.3. We say that A learns $\mathcal{F}$ *w.r.t. efficient distributions* if the PAC learning guarantee is only required to hold for all $X = C(U_m)$, where $m = \mathrm{poly}(n)$ and C is a polynomial-size circuit.

Worst possible error is $1/2$*:* we will assume that A always outputs a hypothesis h such that $\mathsf{err}((X, f(X)), h) \leq 1/2$. One can assure that this occurs with probability $\geq 1 - 2^{-n}$ by checking whether the majority of labels output by h agrees with the majority on the examples drawn from the oracle, and if they disagree outputting $1 - h$ instead of h.

A word on padding: In this paper we use $\mathsf{SIZE}(n^2)$, the class of functions computable by size n^2 circuits, as a complete concept class. This class is complete for functions computable by polynomial-size circuits because of a padding argument. For example, in order to learn circuits of size n^c, it suffices to pad an example x of length n to $x0^{n^{c/2}-n}$ which has length $n' = n^{c/2}$, and then running the learner for $\mathsf{SIZE}(n^2)$. This same kind of padding works for the LTC setting defined below.

Agnostic learning: we will also work with an even more demanding notion of learning, called *agnostic learning*, defined in Kearns et al. [16], where the examples may not be labelled according to any fixed function. Here, the goal is to obtain a hypothesis that performs (almost) as well as the best hypothesis in a concept class.

Definition 2.4 (Agnostic Learning)**.** A procedure A agnostically learns the concept class $\mathcal{F}$ if the following holds for every $n \in \mathbb{N}, \varepsilon > 0$, $f \in \mathcal{F}$ and every distribution (X, Y) over $\{0,1\}^{n+1}$. Given access to an example oracle that generates labelled examples according to (X, Y), A produces with success probability $\geq 1 - 2^{-n}$ an ε-good hypothesis h (represented as a circuit), namely $\mathsf{err}((X,Y), h) \leq \mathsf{err}((X,Y), \mathcal{F}) + \varepsilon$. Furthermore, A runs in time $\mathrm{poly}(n, 1/\varepsilon)$.

Definition 2.5. We say that A agnostically learns $\mathcal{F}$ *w.r.t. efficient distributions* if the agnostic learning guarantee holds for all $(X, Y) = C(U_m)$, where $m = \mathrm{poly}(n)$ and C is a polynomial-size circuit.

2.3 Learning to create

Definition 2.6 (LTC)**.** A procedure A *solves LTC* for the concept class $\mathcal{F}$ if the following holds for every $n \in \mathbb{N}, \varepsilon > 0$, and $f \in \mathcal{F}$, where $f : \{0,1\}^m \to \{0,1\}^n$. Given access to an oracle that generates samples according to $f(U_m)$, A outputs with probability $1 - \varepsilon$ an ε-close hypothesis

$h : \{0,1\}^{m'} \to \{0,1\}^n$ (represented as a circuit), namely $\Delta(f(U_m), h(U_{m'})) \le \varepsilon$. [4] Furthermore, A runs in time $\mathrm{poly}(n, 1/\varepsilon)$.

The accuracy of the hypotheses: our definition of PAC learning requires a strong notion of accuracy: for an example oracle (X, Y), we require the hypothesis h to satisfy $\mathsf{err}((X,Y),h) \le 1/\mathrm{poly}(n)$. In the PAC model we know one may apply Boosting [22, 7, 8] to show that "strong PAC learning" is equivalent to "weak PAC learning", where the hypothesis is only required to satisfy $\mathsf{err}((X,Y),h) \le \frac{1}{2} - 1/\mathrm{poly}(n)$. In contrast, there is no known equivalent boosting technique for the LTC problem, so we must acknowledge that our definition that $\Delta(f(U_m), h(U_{m'})) \le 1/\mathrm{poly}(n)$ is indeed a strong requirement, and this is necessary for our results.

2.4 Cryptography using circuits

Standard cryptography is based on *uniform* cryptographic primitives: for example, one-way functions or pseudo-random functions that are computable using uniform Turing Machines. Here we define analogues of these two cryptographic primitives that are computable using non-uniform families of circuits.[5] We assume the reader is familiar with the standard notions of one-way functions and pseudorandom functions/permutations, and refer to [9] for further details.

Definition 2.7. Pseudorandom circuits (PRC) exist if for every efficient distinguisher D, there exists an infinite collection W of functions where for every $f \in W$, $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, f is computable by a circuit of size $s(n) = \mathrm{poly}(n)$ and it holds that

$$\left| \Pr_{D, k \overset{R}{\leftarrow} U_n} [D^{f_k}(f, 1^s) = 1] - \Pr_{D,\phi} [D^{\phi}(f, 1^s) = 1] \right| \le s^{-\omega(1)} \tag{2.1}$$

where $f_k = f(k, \cdot)$ and ϕ is a truly random function from $\{0,1\}^n \to \{0,1\}$.

f is a pseudorandom permutation (PRP) if $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, f_k is a permutation for all k, f is computable by a uniform Turing Machine, and Equation 2.1 holds for all efficient D.

Definition 2.8. Distributional one-way circuits (DOWC) exist if for every efficient algorithm I, there exists a polynomial $p(s)$ and an infinite collection W of functions where for every $f \in W$, $f : \{0,1\}^n \to \{0,1\}^m$, f is computable by a circuit of size $s(n) = \mathrm{poly}(n)$ and it holds that

$$\Delta((x, f(x)), (I(f, y), y \mid y = f(x))) > 1/p(s)$$

over random choice of$x \leftarrow_{\mathrm{R}} U_n$ and the random coins of I, and f is given to I as a circuit.

Theorem 2.9 ([11, 10, 12])**.** *DOWC exist if and only if PRC exist.*

[4] Kearns et al. [15] defined closeness using KL divergence. We use statistical distance as this is sufficient in most applications and simplifies our presentation. All of our results also hold for KL divergence with appropriate (but qualitatively equivalent) scaling of parameters.

[5] Note that even though we allow the primitives to be computable by circuits, we only require security to hold against uniform adversaries. This is the reverse of what is normally studied in cryptography, where the primitive is computable by a uniform TM but should be secure against non-uniform adversaries. This difference stems from the different definitions and intuitions underlying learning theory and cryptography.

Historical notes: one-way circuits were studied in the context of zero knowledge, first in Ostrovsky and Wigderson [20] and later in Vadhan [25], who used a slightly different definition. They were called *auxiliary-input one-way functions* in these contexts. Our definition is based on [20] (although we require the inverter to *distributionally* invert the circuit, *i.e.* it must output a nearly-uniform preimage rather than an arbitrary preimage). One-wayness and pseudorandom functions are known to be equivalent for the uniform case [11, 10, 12], and the reductions establishing this extend immediately to the non-uniform case. The connection between one-way circuits and learning theory (also linking learning theory to zero knowledge) was explored in Applebaum et al. [1], Xiao [28, 29].

2.5 Solving LTC implies inverting circuits

It was shown in Kearns et al. [15] that one can use an algorithm solving LTC in order to distinguish PRP from truly random functions. By looking at their proof, we observe that it also applies to PRC.

Theorem 2.10 (Kearns et al. [15])**.** *If LTC is solvable for the class* $\mathsf{SIZE}(n^2)$*, then PRC do not exist,* i.e. *there is an algorithm that distinguishes any efficient circuit from a truly random function.*

Corollary 2.11 (Follows from Theorem 2.10 and Theorem 2.9)**.** *If LTC is solvable for the class* $\mathsf{SIZE}(n^2)$*, then no family of circuits is distributionally one-way,* i.e. *there is an algorithm that distributionally inverts any polynomial-size circuit.*

3 Solving LTC implies solving agnostic learning w.r.t. efficient distributions

The idea of our proof of Theorem 1.1 is that we can use an LTC solver to learn the circuit that samples the distribution of labelled examples. This makes the PAC learning problem easier because we now have a circuit generating labelled exampled (rather than just a set of labelled examples), and we show that the LTC solver can also be used to solve this relaxed PAC learning problem.

3.1 Circuit agnostic learning

To prove Theorem 1.1, we introduce as a tool a relaxed notion of learning called "circuit agnostic learning". In standard notions of learning, the learning algorithm is given access only to examples drawn from the distribution (X, Y). One can also ask what happens when the learning algorithm gets access to a circuit that samples from the distribution (X, Y). For the setting of agnostic learning, we call this relaxed (and potentially easier) problem *circuit agnostic learning*:

Definition 3.1. A procedure A circuit-agnostic-learns a concept class $\mathcal{F}$ if on input circuit C of size s sampling a distribution (X, Y) over $\{0,1\}^{n+1}$, A outputs a hypothesis h such that

$$\mathsf{err}((X,Y),h) \leq \mathsf{err}((X,Y),\mathcal{F}) + \varepsilon$$

and A runs in time $\mathrm{poly}(s, 1/\varepsilon)$.

The following lemma implicitly was proved in [1] and explicitly appears in Xiao [29]

Lemma 3.2 ([1] (see also [29], Lemma 3.5.1))**.** *If there is an efficient algorithm that distributionally inverts all polynomial-size circuits, then there is an algorithm running in time* $\text{poly}(n, 1/\varepsilon)$ *that circuit-agnostically-learns* $\mathsf{SIZE}(n^2)$.

3.2 Proof of Theorem 1.1

Proof of Theorem 1.1. By hypothesis there exists a polynomial-time algorithm A_{LTC} that solves LTC for the class $\mathsf{SIZE}(n^2)$. By Corollary 2.11 it follows that there is a polynomial-time algorithm I that distributionally inverts all circuits. By Lemma 3.2 there is a polynomial-time algorithm $A_{\mathsf{CircLearn}}$ that circuit-agnostically-learns $\mathsf{SIZE}(n^2)$.

Defining the agnostic learning algorithm: the algorithm A_{Agn} that learns agnostically w.r.t. efficiently samplable distributions does the following. By padding, we may assume that the input distribution X, Y is sampled by a circuit of size n^2. (One can remove the assumption that we know the complexity of sampling (X, Y) by repeatedly running the algorithm A_{Agn} for successively larger guesses of the complexity until it produces a good hypothesis.) First, A_{Agn} obtains enough samples $(x_1, y_1), \ldots, (x_{t(n)}, y_{t(n)})$ from the example oracle to run A_{LTC} with error parameter $\varepsilon/2$. Let C be its output. Run $A_{\mathsf{CircLearn}}$ on C with error $\varepsilon/2$, and let h be the output of $A_{\mathsf{CircLearn}}$. Output h.

Analyzing A_{Agn}: since A_{LTC} solves LTC for the concept class $\mathsf{SIZE}(n^2)$, we get with probability $1 - 2^{-n}$ a hypothesis $C : \{0,1\}^m \to \{0,1\}^n$ such that $\Delta(C(U_m), (X, Y)) \leq \varepsilon/2$. Letting (X', Y') be the distribution samples by C, this implies that $\Delta((X', Y'), (X, Y)) \leq \varepsilon/2$. Since $A_{\mathsf{CircLearn}}$ solves $\mathsf{CircLearn}^{\mathsf{SIZE}(n^2)}$, it follows that with probability $1 - 2^{-n}$, the output hypothesis h satisfies $\mathsf{err}((X', Y'), h) \leq \varepsilon/2$. Together, it follows that $\mathsf{err}((X, Y), h) \leq \varepsilon$. ■

Remark 3.3. All of the ingredients used in the proof of Theorem 1.1 relativize, and therefore the statement of Theorem 1.1 also relativizes. Namely, relative to any oracle $\mathcal{O}$, if solving LTC for the class $\mathsf{SIZE}^{\mathcal{O}}(n^2)$ is easy, then PAC learning $\mathsf{SIZE}^{\mathcal{O}}(n^2)$ with respect to input distributions samplable by $\mathsf{SIZE}^{\mathcal{O}}(\text{poly}(n))$ circuits is easy.

4 LTC and PAC learning for concrete classes

In this section we show that, in contrast to Theorem 1.1, if one studies concrete concept classes that are not complete, then it is possible that PAC learning is harder than LTC.

Because PAC learning deals with single-bit output function while LTC deals with multi-bit output functions, in order to compare the two models for concrete concept classes we use two different ways to obtain both single- and multi-bit output functions from a single concept class:

1. Direct products: let $\mathcal{F}$ be a class of single-bit output functions. Then we compare PAC learning the class $\mathcal{F}$ to solving LTC for the class $\mathcal{F}^\ell$ where each function $f \in \mathcal{F}^\ell$ maps $\{0,1\}^n \to \{0,1\}^\ell$ and can be decomposed as $f(x) = (f_1(x), \ldots, f_\ell(x))$ where each $f_i \in \mathcal{F}$. Here, $\omega(\log n) \leq \ell(n) \leq \text{poly}(n)$.

2. Generating labelled examples: let $\mathcal{F}$ be a class of single-bit output functions and let $\mathcal{D}$ be a class of distributions that are efficiently samplable. Then we compare PAC learning $\mathcal{F}$ with respect to input distributions in $\mathcal{D}$ to solving LTC for the class of distributions of the form $(X, f(X))$ where $X \in \mathcal{D}$ and $f \in \mathcal{F}$.

To motivate the above notions, the direct product notion is natural when thinking of $\mathcal{F}$ as being a syntactic complexity class, such as DNF formulas or $\mathsf{AC0}$ circuits. Thus, in the PAC model the function to be learned has complexity $\mathcal{F}$, and similarly in the LTC problem each bit of output in the output distribution has complexity $\mathcal{F}$.

The "generating labelled examples" notion is motivated by the proof of Theorem 1.1. In the proof of Theorem 1.1, the first step is to apply the LTC algorithm to produce a circuit that generates (approximately) the distribution of labelled examples. By considering this notion, we will see that the proof of Theorem 1.1 does not immediately generalize to hold for concrete concept classes.

Since we have no unconditional lower bounds for polynomial-time computation (which would be necessary to show that polynomial-time algorithms cannot solve PAC or LTC), all of the following results are conditional, *i.e.* they assume that some (standard) computational problem is hard.

4.1 Direct product

Proposition 4.1. *Assuming one-way functions exist, then there exists a concept class $\mathcal{F}$ that is hard to learn in the PAC model but such that solving LTC for the class $\mathcal{F}^\ell$ is easy.*

Proof. Since we assume one-way functions exist, therefore [11, 10, 17] implies that there exists a pseudorandom permutation (PRP) of the form $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ (we use the short-hand $f_k(x) = f(k, x)$) such that f_k is a permutation, and for all efficient distinguishers D,

$$|\Pr_{\phi,D}[D^{\phi}(1^n) = 1] - \Pr_{k,D}[D^{f_k}(1^n) = 1]| \leq \varepsilon(n)$$

where the distinguishing advantage $\varepsilon(n) = n^{-\omega(1)}$ is negligible. We make the following claim, which says that there exists an infinite collection of keys K such that f_k is hard-to-compute for all $k \in K$:

Lemma 4.2. *Let $\{f_k\}_{k\in\{0,1\}^*}$ be a collection of pseudorandom permutations with distinguishing advantage $\varepsilon(n)$. Then there exists an infinite set $K = \{k_n\}_{n\in\mathbb{N}}$ such that $\forall k \in K$, $n = |k|$, it holds for all efficient algorithms A that for large enough n,*

$$\Pr_A[A^{f_{k_n}} = h \textit{ and } \mathsf{err}((U_n, f_{k_n}(U_n)), h) \leq 1/2 - 2\sqrt{\varepsilon(n)}] \leq n^2\sqrt{\varepsilon(n)}$$

We will prove this lemma shortly, first we use it to define $\mathcal{F}$ and prove Proposition 4.1.

Defining $\mathcal{F}$: let $K = \{k_n\}_{n\in\mathbb{N}}$ be the set of hard keys defined by Lemma 4.2. Let $f_{k_n}(x)_i$ denote the i'th bit of $f_{k_n}(x)$. We define

$$\mathcal{F} = \bigcup_{n\in\mathbb{N}} \{g_i : \{0,1\}^n \to \{0,1\}, g_i(x) = f_{k_n}(x)_i \mid i \in [n]\}$$

Claim 4.3. *PAC learning $\mathcal{F}$ is hard.*

This follows from Lemma 4.2: if there were an algorithm that PAC learns $\mathcal{F}$, then it could in particular be used to compute f_{k_n} for all n: given oracle access to f_{k_n}, one can simulate example oracles $(U_n, g_i(U_n))$ for all $i \in [n]$, and using the PAC learning algorithm for $\mathcal{F}$ with error $1/n^2$ one could obtain with high probability hypotheses h_i such that $\Pr[h_i(U_n) = g_i(U_n)] \geq 1 - 1/n^2$. Letting $h = (h_1, \ldots, h_n)$, we see that $\mathsf{err}((U_n, f_{k_n}(U_n)), h) \leq 1/n$, which contradicts Lemma 4.2 since $1/n \ll 1/2 - 2\sqrt{\varepsilon(n)}$.

Claim 4.4. *Solving LTC for the class $\mathcal{F}^\ell$ is easy.*

Fix any function $(g_{i_1}, \ldots, g_{i_\ell}) \in \mathcal{F}^\ell$. Since f_{k_n} is a permutation, $f_{k_n}(U_n)$ is uniform. Therefore, $g_{i_p}(U_n)$ and $g_{i_q}(U_n)$ are independent uniform bits if $i_p \neq i_q$, and they are always equal if $i_p = i_q$. We now describe an algorithm that solves LTC for the class $\mathcal{F}^\ell$. Let D be the distribution to be learned, $D = (g_{i_1}(r), \ldots, g_{i_q}(r) \mid r \leftarrow_{\mathrm{R}} U_n)$.

1. Initialize a graph G on n vertices to be the complete graph, where the vertices are labelled $1, \ldots, n$.
2. Repeat the following $t = n \log \binom{n}{2}$ times. Sample $x \leftarrow_{\mathrm{R}} D$, and for every pair $u, v \in [n]$ such that $x_u \neq x_v$, remove the edge (u, v) from G.
3. The output hypothesis h does the following: for each connected component of G, sample a random bit. Output x where x_u equals the bit of the connected component containing u.

We claim that with all but probably 2^{-n}, the distribution sampled by h will be *exactly* the distribution generated by $(g_{i_1}, \ldots, g_{i_\ell})$. The only time that this hypothesis will be different is if there is some pair u, v such that $i_u \neq i_v$ and yet, for all examples x that the algorithm draws, it holds that $x_{i_u} = x_{i_v}$. Since for each sample this happens independently with probably $1/2$, and since there are $t = n \log \binom{n}{2}$ samples, by a union bound over all edges this happens with probably at most $\binom{n}{2} 2^{-n \log \binom{n}{2}} \leq 2^{-n}$. ■

Proof of Lemma 4.2. For an algorithm A and a key k of length n, let

$$p_{A,k} = \Pr_A[A^{f_k} = h \text{ and } \mathsf{err}((U_n, f_k(U_n)), h) \leq 1/2 - 2\sqrt{\varepsilon(n)}]$$

where the probability is only over the random coins of A. It must hold that:

Claim 4.5. $\Pr_k[p_{A,k} > n^2 \sqrt{\varepsilon(n)}] \leq 1/n^2$

This claim holds because otherwise one could use A to break the pseudorandomness of f by the following distinguisher D. First D runs A to obtain h. Then, D queries its oracle on a new uniform $x \leftarrow_{\mathrm{R}} U_n$; let b be the oracle's response. D accepts if $b = h(x)$ and rejects otherwise. It is easy to compute $\Pr_{\phi,D}[D^\phi(1^n) = 1] \leq \frac{1}{2} + p(n)2^{-n}$ where $p(n) = \mathrm{poly}(n)$ is the maximum number of queries made by A. On the other hand, if Claim 4.5 does not hold, then $\Pr_{k,D}[D^{f_k}(1^n) = 1] \geq \frac{1}{2} + 2\varepsilon(n)$. We can assume *w.l.o.g.* that $\varepsilon \geq 2^{-n/2}$ (Lemma 4.2 only gets weaker if we increase ε), therefore this gives a distinguishing probability $2\varepsilon - p(n)2^{-n} \gg \varepsilon(n)$, contradicting the pseudorandomness of f_k.

Define $p_A = \Pr_{k_1,k_2,\ldots}[\text{For inifinitely many } n,\ p_{A,k_n} > n^2\sqrt{\varepsilon(n)}]$ where $k_n \leftarrow_{\mathrm{R}} \{0,1\}^n$. Since the series $\sum_{n=1}^{\infty} 1/n^2 < \infty$, applying Theorem 2.1 and Claim 4.5 implies that $p_A = 0$. Since there is a countable number of algorithms A, this implies

$$\Pr_{k_1,k_2,\ldots}[\exists A,\ \text{For infinitely many } n,\ p_{A,k_n} > n^2\sqrt{\varepsilon(n)}] \leq \sum_A p_A = 0$$

Therefore, a random choice of K will satisfy the conclusion of Lemma 4.2 with probability 1. ■

4.2 Generating labelled examples

Our result for this model is based on the hardness of quadratic residuosity over Blum integers. We say that $N = pq$ is a Blum integer of length n if p, q are prime, $\lceil \log N \rceil = n$, $n - \lceil \log p \rceil \leq 2$, $n - \lceil \log q \rceil \leq 2$ and $p \equiv q \equiv 3 \pmod 4$. We say that x is a quadratic residue mod a if $\exists y \in \mathbb{Z}_N$ such that $x = y^2 \bmod a$ for $a \in \mathbb{N}$. The *Legendre symbol* $(\frac{x}{p})$ is equal to 0 if $x = 0 \bmod p$, it is equal to 1 if x is a quadratic residue mod p and -1 if x is a quadratic non-residue mod p. The *Jacobi symbol* is defined $(\frac{x}{N}) = (\frac{x}{p})(\frac{x}{q})$. It is possible to efficiently compute the Jacobi symbol using Euclid's algorithm. It is known that for a Blum integer N, $(\frac{-1}{N}) = 1$ but -1 is *not* a quadratic residue mod N.

Let $\mathrm{QR}(N, x) = 1$ if $x = y^2 \mod N$ and 0 otherwise, and write $\mathrm{QR}_N(x) = \mathrm{QR}(N, x)$. The hardness of quadratic residuosity over Blum integers says that there is no polynomial time algorithm that evaluates $\mathrm{QR}_N(x)$ given a Blum integer N and $x \in \mathbb{Z}_N$.

Proposition 4.6. *Assuming that Quadratic Residuosity is hard over Blum integers, there is a concept class $\mathcal{F}$ for which PAC learning with respect to the uniform distribution is hard while solving LTC for distributions $(U_n, f(U_n))$ where $f \in \mathcal{F}$ is easy.*

Proof. Define the functions $f_N : \mathbb{Z}_N \times [\lceil \log N \rceil] \times \{0,1\} \to \{0,1\}$ where

$$f_N(x, i, b) = \begin{cases} \mathrm{QR}_N(x) & b = 0 \\ N_i & b = 1 \end{cases}$$

where N_i denotes the i'th bit of N. Let n the input length of f_N and let $\mathcal{F} = \{f_N \mid N \text{ is a Blum integer}\}$.

Claim 4.7. *PAC learning $\mathcal{F}$ is hard for the uniform distribution.*

Suppose we have a PAC learning algorithm A for $\mathcal{F}$ (we can even require it to work only for uniformly distributed inputs). Note that, given N, one can simulate an example oracle for $(U_n, f_N(U_n))$ as follows:

Algorithm 4.8 (Sampling from $(U_n, f_N(U_n))$:)**.**

1. Pick $x' \leftarrow_{\mathrm{R}} \mathbb{Z}_N, i \leftarrow_{\mathrm{R}} [\lceil \log N \rceil], b \leftarrow_{\mathrm{R}} \{0,1\}$.
2. If $b = 1$ then output $((x', i, b), N_i)$, otherwise compute the Jacobi symbol $(\frac{x'}{N})$.
3. If $(\frac{x'}{N}) \neq 1$, then output $((x', i, b), 0)$.
4. If $(\frac{x'}{N}) = 1$, then sample $r \leftarrow_{\mathrm{R}} \mathbb{Z}_N, a \leftarrow_{\mathrm{R}} \{-1, 1\}$, and output $((ar^2, i, b), \frac{1+a}{2})$.

This simulated example oracle is identical to a true example oracle for f_N.

Using a PAC learner to define an algorithm A' solving quadratic residuosity: on input (x, N), A' first checks if $(\frac{x}{N}) \neq 1$, and if so outputs 0. Otherwise, use Algorithm 4.8 to simulate an example oracle for $(U_n, f_N(U_n))$, then run A on this example oracle to obtain a hypothesis h. Finally, A' picks a random $r \leftarrow_{\mathrm{R}} \mathbb{Z}_N, i \leftarrow_{\mathrm{R}} [n]$ and outputs $h(xr^2, i, 0)$.

We claim that A' solves quadratic residuosity. Let S_1 be the set of quadratic residues in $\mathbb{Z}_N$, and let S_{-1} be the set of quadratic non-residues $y \in Z_N$ with Jacobi symbol $(\frac{y}{N}) = 1$. Observe that $\Pr_{y \leftarrow_{\mathrm{R}} \mathbb{Z}_N}[y \in S_1] = \Pr_{y \leftarrow_{\mathrm{R}} Z_N}[y \in S_{-1}] = 1/4 + o(1)$. Therefore, for all $a \in \{-1, 1\}$, it holds over uniform i that

$$\mathsf{err}(((S_a, i, 0), f_N(S_a, i, 0)), h) \leq (8 + o(1))\mathsf{err}((U_n, f_N(U_n)), h) \leq 9\varepsilon$$

for large enough n. Since for $a \in \{-1, 1\}$ and every $x \in S_a$, the variable $xr^2 \bmod N$ for random r is distributed uniformly in S_a, this implies that A' outputs $\mathrm{QR}_N(x)$ correctly with probability $1 - 9\varepsilon - 2^{-n}$.

Claim 4.9. *Solving LTC for $(U_n, f_N(U_n))$ for $f_N \in \mathcal{F}$ is easy.*

After seeing $O((\log N)^2) = \mathrm{poly}(n)$ samples, with all but negligible probability, N is revealed. Given N, one can sample from $(U_n, f_N(U_n))$ using Algorithm 4.8. ■

5 PAC learning unsamplable distributions

Our construction of $\mathcal{O}$ will be randomized: we will select $\mathcal{R}$ from a distribution of oracles and show that with high probability that the oracle $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ satisfies Theorem 1.3. The distribution will be as follows:

Definition 5.1. On input length n, let $\mathcal{R} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be chosen as follows: select $z \leftarrow_{\mathrm{R}} \{0,1\}^n$. Pick the set S_z from the following distribution: for each $x \in \{0,1\}^n$, put x into S_n with probability $2^{-n/2}$. Then, for each $x \in S_z$, let $\mathcal{R}_z(x) = \mathcal{R}(z, x) = 1$ with probability $1/2$ and 0 otherwise. For all $z' \neq z$, let $\mathcal{R}_{z'}(x) = 0$ for all $x \in \{0,1\}^n$.

Intuitively, for each input length we first pick a "hard instance" z, then we pick a "hard set" S_z that is a sparse random subset of $\{0,1\}^n$ of size roughly $2^{n/2}$. We then define $\mathcal{R}_z$ to be a random function on S_z and 0 elsewhere, and also $\mathcal{R}_{z'}$ is identically zero for all $z' \neq z$.

Proof of Theorem 1.3. We will show that with overwhelming probability over choice of such $\mathcal{R}$, solving LTC relative to $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ is easy, but PAC learning relative to $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ with respect to unsamplable distributions is hard.

Lemma 5.2 (PAC learning unsamplable distribution is hard)**.** *With probability* 1 *over $\mathcal{R}$, for all efficient algorithms A with oracle access to $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$, and for all but finitely many n, let $z \in \{0,1\}^n$ be the hard instance on length n, then*

$$\Pr_A[A^{\mathcal{O}} \textit{ given access to } (S_z, \mathcal{R}_z(S_z)) \textit{ outputs } h \textit{ s.t. } \mathsf{err}((S_z, \mathcal{R}_z(S_z)), h) \leq \tfrac{1}{2} - n^{-\log n}] \leq n^{-\log n}$$

The intuition is that $\mathcal{R}_z(x)$ over $x \leftarrow_{\mathrm{R}} S_z$ is a random function, so no algorithm should be able to predict $\mathcal{R}_z(x)$ for some $x \in S_z$ that it has not yet seen.

On the other hand, we will prove the following:

Lemma 5.3 (Solving LTC easy)**.** *There is an efficient $A^{\mathcal{O}}$ such that with probability* 1 *over the choice of $\mathcal{R}$ where $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$, $A^{\mathcal{O}}$ solves LTC for the concept class $\mathsf{SIZE}^{\mathcal{O}}(n^2)$.*

Together, these two lemmas imply Theorem 1.3. Notice that we did not need to prove separately that S_z is unsamplable; this follows immediately since learning w.r.t. S_z is hard while learning w.r.t efficiently samplable distributions is easy: Remark 3.3 and Lemma 5.3 imply that there is an efficient algorithm that solves PAC learning for the concept class $\mathsf{SIZE}^{\mathcal{O}}(n^2)$ with respect to efficiently samplable distributions. ■

Proof of Lemma 5.2. Fix a polynomial-time algorithm A, and let $t(n) = \mathrm{poly}(n)$ be an upper bound on the number of samples that A observes. Let $T = \{(x_i, \mathcal{R}_z(x_i))\}_{i=1,\ldots,t}$ be the samples that A observes. Define

$$p_{A,\mathcal{R},n} = \Pr_{A,T}\left[\text{On samples } T,\ A^{\mathcal{O}} \text{ outputs } h \text{ such that } \mathsf{err}((S_z, \mathcal{R}_z(R_z)), h) \le \tfrac{1}{2} - n^{-\log n}\right]$$

and also let $p_{A,n} = \mathbb{E}_{\mathcal{R}}[p_{A,\mathcal{R},n}]$. Then it holds that

$$\begin{aligned} p_{A,n} &= \Pr_{A,T,\mathcal{R}}\left[\text{On samples } T,\ A^{\mathcal{O}} \text{ outputs } h \text{ such that } \mathsf{err}((S_z, \mathcal{R}_z(R_z)), h) \le \tfrac{1}{2} - n^{-\log n}\right] \\ &\le n^{\log n}\left(\Pr_{\mathcal{R},A,T,x' \leftarrow_{\mathrm{R}} S_z}\left[A^{\mathcal{O}} \text{ outputs } h \quad \wedge \quad h(x') = \mathcal{R}_z(x')\right] - 1/2\right) \end{aligned} \tag{5.1}$$

Let B_n be the event that $A^{\mathcal{O}}$ outputs h (which may contain oracle gates) and that in the computation of $h(x')$, h queries $\mathcal{R}_z$ where z is the hard instance of length n. We will break the analysis into two cases: first we prove that B_n occurs with very low probability, and then we prove that if B_n does not occur and x' does not appear in T then $\mathcal{R}_z(x')$ is random and so $h(x')$ is correct with probability 1/2.

To show B_n occurs with low probability, let Y denote the labels to examples in T, *i.e.* T fixes $x_1, \ldots, x_t$ and Y fixes $y_1, \ldots, y_t$ such that the set of examples A sees is $(x_1, y_1), \ldots, (x_t, y_t)$. Observe that one can fix A, T, Y, x' and still z will still be uniform and independent in $\{0,1\}^n$. This is because the choice of z is independent of the choice of S_z and $\mathcal{R}_z(x)$ for $x \in S_z$. Therefore:

$$\Pr_{\mathcal{R},A,T,x'}[B_n] = \mathbb{E}_{A,T,Y,x'} \Pr_{\mathcal{R}}[B_n \mid A, T, Y, x']$$

However, if A, T, Y, x' are all fixed, then the queries of h are all fixed. h can make at most $\mathrm{poly}(n)$ queries to $\mathcal{R}$, and for each query since z has not yet been fixed the probability that the query hits the hard $\mathcal{R}_z$ is 2^{-n}. Therefore

$$\Pr[B_n] = \mathbb{E}_{A,T,Y,x'} \Pr_{\mathcal{R}}[B_n \mid A, T, Y, x'] \le \mathrm{poly}(n)/2^n \le 2^{-n/3} \tag{5.2}$$

On the other hand, suppose B_n does not occur. Let B'_n be the event that either $|S_z| \le 2^{-n/2-1}$ or x' already appears in T. A Chernoff bound implies that $\Pr[|S_z| \le 2^{n/2-1}] \le 2^{-\Omega(2^n)}$, and conditioned on $|S_z| > 2^{n/2-1}$ it holds that the probability x' already appears in T is at most $2^{-n/2+1}$, therefore for sufficiently large n, it holds that $\Pr[B'_n] \le 2^{-n/3}$. We may write:

$$\begin{aligned} \Pr[A \text{ outputs } h \wedge h(x') = \mathcal{R}_z(x') \wedge \overline{B_n}] &\le \Pr[A \text{ outputs } h \wedge h(x') = \mathcal{R}_z(x') \mid \overline{B_n}] && (5.3) \\ &\le 2^{-n/3} + \Pr[A \text{ outputs } h \wedge h(x') = \mathcal{R}_z(x') \mid \overline{B'_n} \wedge \overline{B_n}] && (5.4) \\ &\le 2^{-n/3} + 1/2 && (5.5) \end{aligned}$$

where the last inequality is because if x' does not appear in T and h does not query z, then the value of $\mathcal{R}_z(x')$ is uniform and independent and therefore $h(x')$ is correct with probability 1/2.

Combining Equation 5.2 and Equation 5.5 along, we see that

$$\Pr[A \text{ outputs } h \wedge h(x') = \mathcal{R}_z(x')] \leq \Pr[B_n] + \Pr[A \text{ outputs } h \wedge h(x') = \mathcal{R}_z(x') \wedge \overline{B_n}] \leq 1/2 + 2^{-n/3+1}$$

Substituting into Equation 5.1, we get that

$$p_{A,n} \leq (2^{-n/3+1}) \cdot n^{\log n} \ll n^{-2\log n}$$

Let E_n be the event over $\mathcal{R}$ that $p_{A,\mathcal{R},n} > n^{\log n}$. Then by a standard averaging argument, $\Pr_{\mathcal{R}}[E_n] \leq n^{-\log n}$. By Theorem 2.1, only a finite number of the E_n can occur, therefore

$$\Pr_{\mathcal{R}}\left[\text{for infinitely many } n,\, p_{A,\mathcal{R},n} > n^{-\log n}\right] = 0$$

The above is still for a fixed algorithm A, but since there are only countably many algorithms A, we can take a union bound over all algorithms to conclude that with probability 1 over $\mathcal{R}$, for all but finitely many n, it holds that $\Pr_{\mathcal{R}}[E_n] \leq n^{-\log n}$, proving the claim. ■

Proof of Lemma 5.3. **The maximum likelihood approach:** We will use a maximum likelihood approach to solve LTC. The maximum likelihood algorithm says that, given a sample $T = (x_1, \dots, x_t)$ that was obtained from one distribution out of a class of distributions $\mathcal{D}$, it suffices to pick the $D \in \mathcal{D}$ such that $\Pr[D^t = T]$ is maximized.

More formally, let $\mathrm{ML}_{\mathcal{D}}(x_1, \dots, x_t) = \mathrm{argmax}_{D \in \mathcal{D}}\{\Pr[D^t = (x_1, \dots, x_t)]\}$. The following holds:

Claim 5.4 (Folklore)**.** *Fix $\mathcal{D}$ of size $|\mathcal{D}| \leq 2^{\mathrm{poly}(n)}$ and such that for every $D \in \mathcal{D}$ and every $x \in \mathrm{supp}(D)$, $\Pr[D = x] \geq 2^{-\mathrm{poly}(n)}$. Then for $t = \mathrm{poly}(n, 1/\varepsilon)$ and for any $D \in \mathcal{D}$, it holds that:*

$$\Pr_{x_1,\dots,x_t \leftarrow_R D}[\mathrm{ML}_{\mathcal{D}}(x_1, \dots, x_t) = D' \wedge \Delta(D', D) > \varepsilon] \leq 2^{-n}$$

We include a proof in the appendix for the sake of completeness.

Let $\mathcal{D}^{\mathcal{O}}$ be the class of distributions sampled by circuits in $\mathsf{SIZE}^{\mathcal{O}}(n^2)$. To solve LTC for $\mathsf{SIZE}^{\mathcal{O}}(n^2)$, we would like to run the maximum likelihood approach over $\mathcal{D}^{\mathcal{O}}$. However, in order to calculate or even roughly approximate $\Pr[(D^{\mathcal{O}})^t = (x_1, \dots, x_t)]$ for distributions $D^{\mathcal{O}}$ that might be sampled by circuits including $\mathcal{O}$ gates, one would need to know how $\mathcal{O} = (\mathcal{R}, \mathbf{PSPACE})$ behaves everywhere, and this requires querying $\mathcal{R}$ exponentially many times.

Our approach is to still use the maximum likelihood approach, but rather than applying the approach using $\mathcal{D}^{\mathcal{O}}$ as the class of distributions, we apply it to a related class $\mathcal{D}'_q = \{\mathcal{D}'_{q,n}\}_{n \in \mathbb{N}}$ for which having a **PSPACE** oracle is sufficient to calculate the maximum likelihood hypothesis.

Defining $\mathcal{D}'$ using truncated oracles: $\mathcal{D}'_{q,n}$ will be the following class of distributions. For $z \in \{0,1\}^n, S \subseteq \{0,1\}^n$, define the function $R_n^{z,S} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ to be $R_n^{z,S}(z, x) = 1$ if $x \in S$ and zero elsewhere. Note that $R_n^{z,S}$ can be concisely represented if $|S| = \mathrm{poly}(n)$. The class of $q(n)$-truncated oracles on length n is the following:

$$\mathcal{R}_{q,n} = \left\{ R = (R_1^{z_1,S_1}, \dots, R_n^{z_n,S_n}) \mid \forall i \in [n],\; z_i \in \{0,1\}^i, S_i \subseteq \{0,1\}^i,\; |S_i| \leq q(n) \right\}$$

A distribution D is in $\mathcal{D}'_{q,n}$ if there exists an $R \in \mathcal{R}_{q,n^2}$ and oracle circuit of size n^2 that has (in addition to AND, OR, and NOT gates) **PSPACE** gates and R gates. Note that the circuit is allowed oracle gates only for a single $R \in \mathcal{R}_{q,n^2}$, or in other words it cannot have two different kind of oracle gates evaluating two different $R \neq R' \in \mathcal{R}_{\varepsilon,n^2}$. Also note that because we can explicitly represent S_i, each of these circuits is contained in $\mathsf{SIZE}^{\mathbf{PSPACE}}(n^3(1+q))$.

The following straightforward claim says that, with a **PSPACE** oracle, it is possible to efficiently evaluate the probability that $D \in \mathcal{D}'_{q,n}$ generates a particular sample:

Claim 5.5. *There exists an algorithm using a* **PSPACE** *oracle that runs in time* $\mathrm{poly}(n, 1/\varepsilon)$ *and computes* $\mathrm{ML}_{\mathcal{D}'_{q,n}}$.

This follows from the simple fact that calculating $\Pr[D = x]$ for a distribution D that is samplable by a $\mathsf{SIZE}^{\mathbf{PSPACE}}(\mathrm{poly}(n))$ circuit can be efficiently done with a **PSPACE** oracle. Since, as remarked above, $\mathcal{D}'_{q,n}$ can be sampled by $\mathsf{SIZE}^{\mathbf{PSPACE}}(n^3(1+q))$ circuits, this suffices to build the algorithm in Claim 5.5 (we include a proof in the appendix for the sake of completeness).

It therefore remains to prove that, with high probability over the choice of funtion $\mathcal{R}$, the class $\mathcal{D}'_{q,n}$ is a good approximation for the class $\mathcal{D}^{\mathcal{O}} = \{\mathcal{D}^{\mathcal{O}}_n\}_{n\in\mathbb{N}}$ sampled by circuits in $\mathsf{SIZE}^{\mathcal{O}}(n^2)$. We say that $\mathcal{D}^{\mathcal{O}}_n$ is ε-approximable by $\mathcal{D}'$ if for every $D \in \mathcal{D}^{\mathcal{O}}_n$, there exists $D' \in \mathcal{D}'$ such that $\Delta(D, D') \leq \varepsilon$.

Lemma 5.6. *For all* n, ε, *let* $q = 16n^9/\varepsilon^3$, *then* $\Pr_{\mathcal{R}}[\mathcal{D}^{\mathcal{O}}_n$ *is* ε*-approximable by* $\mathcal{D}'_{q,n}] > 1 - 2^{-n}$.

We first use this to prove the theorem.

The learning algorithm A_{LTC}. We now combine our claims to obtain the following algorithm:

Algorithm 5.7.
Algorithm A_{LTC}: input size n, error parameter ε.

1. Let $t = \mathrm{poly}(n, 2/\varepsilon)$ be the appropriate polynomial to apply Claim 5.4 with error $\varepsilon/2$. A_{LTC} draws t examples $x_1, \ldots, x_t$ from D.

2. Using the algorithm of Claim 5.5, set $q = 16n^9(2t/\varepsilon)^3$ and compute $D' = \mathrm{ML}_{\mathcal{D}'_{q,n}}(x_1, \ldots, x_t)$. Output D'.

Proof of correctness: We prove that A_{LTC} indeed solves the LTC problem for $\mathsf{SIZE}^{\mathcal{O}}(n^2)$. By Theorem 2.1, it suffices to show that for all n, with probability $1 - 2^{-n}$ over the choice of $\mathcal{R}$, it holds for all $D \in \mathcal{D}^{\mathcal{O}}_n$ that

$$\Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}[A_{\mathsf{LTC}} = D' \wedge \Delta(D', D) > \varepsilon] \leq \varepsilon \tag{5.6}$$

(This error can be reduced to 2^{-n} by repeating A_{LTC} and taking the best hypothesis it output.) For $q = 16n^9(2t/\varepsilon)^3$, Lemma 5.6 implies that with probability $1 - 2^{-n}$ over the choice of $\mathcal{R}$, $\mathcal{D}^{\mathcal{O}}_n$ is $(\varepsilon/2t)$-approximable by $\mathcal{D}'_{q,n}$. In this case, for every $D \in \mathcal{D}^{\mathcal{O}}_n$, there exists $D' \in \mathcal{D}'_{q,n}$ such that by the triangle inequality it holds that $\Delta(D^t, (D')^t) \leq \varepsilon/2$. Therefore

$$\begin{aligned}\Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}[A_{\mathsf{LTC}} = D' \wedge \Delta(D', D) > \varepsilon] &\leq \Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D'}[A_{\mathsf{LTC}} = D'' \wedge \Delta(D'', D) > \varepsilon] + \varepsilon/2 \\ &\leq \Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D'}[A_{\mathsf{LTC}} = D'' \wedge \Delta(D'', D') > \varepsilon - \varepsilon/(2t)] + \varepsilon/2 \\ &\leq 2^{-n} + \varepsilon/2 \leq \varepsilon\end{aligned}$$

where penultimate inequality follows from Claim 5.4, since $D' \in \mathcal{D}'_{q,n}$ and A_{LTC} evaluates $\mathrm{ML}_{\mathcal{D}'_{q,n}}$ (the conditions of the hypothesis are satisfied because D is samplable by a polynomial-size oracle circuit). This proves Equation 5.6. ∎

Proof of Lemma 5.6. Recalling the definition of ε-approximable, we want to show that for every $D \in \mathcal{D}_n^{\mathcal{O}}$, there exists $D' \in \mathcal{D}'_{q,n}$ such that $\Delta(D, D') \leq \varepsilon$.

Let $C \in \mathsf{SIZE}^{\mathcal{O}}(n^2)$ be the circuit sampling D. Our proof idea follows that of [28]: show that it is only necessary to know the queries that C makes to $\mathcal{R}$ that are "heavy", *i.e.* that occur with large probability. Then we can simply replace $\mathcal{R}$ gates by a truth table that includes values for all the heavy queries.

Fix $C \in \mathsf{SIZE}^{\mathcal{O}}(n^2)$ and the oracle $\mathcal{R}$. Let z_i be $\mathcal{R}$'s hard instance on input length i. For $i \geq 2\log(\varepsilon/(2n^2))$, define the set of δ-heavy queries for C as

$$S_{\delta,i} = \{x \mid \mathcal{R}(z_i, x) = 1 \ \wedge \ \Pr[C(U_n) \text{ queries } \mathcal{R}(z_i, x)] \geq \delta\}$$

For $i < 2\log(\varepsilon/(2n^2))$, define $S_{\delta,i} = \{x \mid \mathcal{R}(z_i, x) = 1\}$. It holds that $\sum_{i=2\log(\varepsilon/(2n^2))}^{n^2} |S_{\delta,i}| \leq 1/\delta$, and $S_{\delta,i} = \varnothing$ for $i > n^2$ since C cannot make queries longer than n^2. Let $S_\delta = \bigcup_i S_{\delta,i}$.

Let R^δ be the function $R^\delta = (R_1^{z_1, S_{\delta,1}}, \ldots, R^{z_1, S_{\delta,n^2}})$. Let C^δ be the same as C except that $\mathcal{R}$ gates are replaced by $\mathcal{R}^\delta$ gates. Observe that the distribution sampled by C^δ is contained in $\mathcal{D}'_{1/\delta,n}$.

Claim 5.8. *For any ε, δ, $\Pr_{\mathcal{R}}[\Delta(C(U_n), C^\delta(U_n)) \leq \varepsilon] \geq 1 - n^2 2^{-\varepsilon^3/(8\delta n^6)}$.*

Setting $\delta = \varepsilon^3/(16n^9)$ we see that for large enough n, $1 - n^2 2^{-\varepsilon^3/(8\delta n^6)} \geq 1 - 2^{-n^3}$. Finally, there are $2^{O(n^2 \log n^2)}$ possible oracle circuits C of size n^2, so taking a union bound over all these implies that with probability $1 - 2^{-n}$ over the choice of $\mathcal{R}$, for every C the corresponding C^δ satisfies $\Delta(C(U_n), C^\delta(U_n)) \leq \varepsilon$.

We now prove Claim 5.8: let $B_\delta = \{x \mid x \notin S_\delta \wedge \mathcal{R}(z_{|x|}, x) = 1\}$. From the definitions, the only inputs r on which C and C^δ can possibly differ are when $C(r)$ queries $\mathcal{R}(z_{|x|}, x)$ for $x \in B_\delta$. Since the circuit C can make at most n^2 queries, define for $j \in [n^2]$ the "bad event" $\mathsf{Bad}_{\delta,j}(C, r)$ that the j'th query that $C(r)$ makes is to $\mathcal{R}(z_{|x|}, x)$ where $x \in B_\delta$. (If C makes fewer than j queries, simply set $\mathsf{Bad}_{\delta,j}(C, r)$ to be the empty event.) By the above, we have that

$$\Delta(C(U_n), C^\delta(U_n)) \leq \Pr_{r \leftarrow_{\mathrm{R}} U_n}[\exists j,\ \mathsf{Bad}_{\delta,j}(C, r)] \leq \sum_{j=1}^{n^2} \Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Bad}_{\delta,j}(C, r)] \tag{5.7}$$

We prove the following claim:

Lemma 5.9. *For all $j \in [n^2]$,*

$$\Pr_{\mathcal{R}}\left[\Pr_{r \leftarrow_R U_n}[\mathsf{Bad}_{\delta,j}(C, r)] > \varepsilon/n^2\right] \leq 2^{-\varepsilon^3/(8\delta n^6)}$$

Notice that by definition x is bad iff it is light (namely, $x \notin S_\delta$) and also $\mathcal{R}(z_{|x|}, x) = 1$. Therefore, Lemma 5.9 holds intuitively because either the probability of making a light query is small, in which case the probability of making a light query that is also bad is small, or else the probability of making a light query is large, in which case there are many light queries and since $\mathcal{R}(z_{|x|}, x) = 1$ with probability $2^{-n/2}$, with overwhelming probability the fraction of light queries that satisfy $\mathcal{R}(z_{|x|}, x) = 1$ must be small.

We first use Lemma 5.9 to conclude the proof: taking a union bound over all $j = 1, \ldots, n^2$ queries as in Equation 5.7, we obtain that with probability $1 - n^2 2^{-\varepsilon^3/(8\delta n^6)}$ over the choice of oracle $\mathcal{R}$, it holds that $\Delta(C(U_n), C^\delta(U_n)) \leq \varepsilon/n^2 \cdot n^2 = \varepsilon$. ■

Analyzing bad queries

Proof of Lemma 5.9. For $j \in [n^2]$, let $\mathsf{Light}_{\delta,j}(C, r)$ be the event that the j'th query that $C(r)$ queries is $\mathcal{R}(z_i, x)$ for some $x \notin S_\delta$ of length greater than $2\log(\varepsilon/(2n^2))$. For every j, it holds that:

$$\begin{aligned}
&\Pr_{\mathcal{R}}\left[\Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Bad}_{\delta,j}(C, r)] > \varepsilon/n^2\right] \\
&\leq \Pr_{\mathcal{R}}\left[\Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Bad}_{\delta,j}(C, r)] > \varepsilon/n^2 \mid \Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Light}_{\delta,j}(C, r)] \leq \varepsilon/n^2\right] \\
&+ \Pr_{\mathcal{R}}\left[\Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Bad}_{\delta,j}(C, r)] > \varepsilon/n^2 \mid \Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Light}_{\delta,j}(C, r)] > \varepsilon/n^2\right]
\end{aligned}$$

Using the fact that the event $\mathsf{Bad}_{\delta,j}(C, r)$ is contained in the event $\mathsf{Light}_{\delta,j}(C, r)$, we may bound the first term by 0 while the second may be bounded by:

$$\leq \Pr_{\mathcal{R}}\left[\Pr_{r \leftarrow_{\mathrm{R}} U_n}\left[\mathsf{Bad}_{\delta,j}(C, r) \mid \mathsf{Light}_{\delta,j}(C, r)\right] > \varepsilon/n^2 \mid \Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Light}_{\delta,j}(C, r)] > \varepsilon/n^2\right]$$

Let X denote the distribution over $C(r)$'s j'th queries conditioned on $\mathsf{Light}_{\delta,j}(C, r)$. Then

$$\begin{aligned}
&\Pr_{\mathcal{R}}\left[\Pr_{r \leftarrow_{\mathrm{R}} U_n}\left[\mathsf{Bad}_{\delta,j}(C, r) \mid \mathsf{Light}_{\delta,j}(C, r)\right] > \varepsilon/n^2 \mid \Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Light}_{\delta,j}(C, r)] > \varepsilon/n^2\right] \\
&= \Pr_{\mathcal{R}}\left[\mathbb{E}_{x \leftarrow_{\mathrm{R}} X}[\mathcal{R}(z_{|x|}, x)] > \varepsilon/n^2 \mid \Pr_{r \leftarrow_{\mathrm{R}} U_n}[\mathsf{Light}_{\delta,j}(C, r)] > \varepsilon/n^2\right]
\end{aligned}$$

By the definition of $\mathcal{R}$, for each x of length greater than $2\log(\varepsilon/(2n^2))$ it holds that $\mathbb{E}_{\mathcal{R}}[\mathcal{R}(z_{|x|}, x) = 1] \leq 2^{-\log(\varepsilon/(2n^2))} = \varepsilon/(2n^2)$. Every x in the support of X is not in S_δ and has length greater than $2\log(\varepsilon/(2n^2))$ has $\Pr[X = x] \leq n^2\delta/\varepsilon$. Therefore, we can apply a Chernoff bound (in a slightly generalized form, found in Lemma B.1) to obtain:

$$\Pr_{\mathcal{R}}[\mathbb{E}_{x \leftarrow_{\mathrm{R}} X}[\mathcal{R}(z_{|x|}, x)] > \varepsilon/n^2] \leq 2^{-(\varepsilon/(2n^2))^2/2 \ \cdot \ \varepsilon/(n^2\delta)}$$

■

References

[1] B. Applebaum, B. Barak, and D. Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Proc. FOCS '08*, pages 211–220, 2008.

[2] T. Baker, J. Gill, and R. Solovay. Relativizations of the $\mathcal{P} =?\mathcal{NP}$ question. *SIAM Journal on Computing*, 4(4):431–442, 1975. doi: 10.1137/0204037. URL http://link.aip.org/link/?SMJ/4/431/1.

[3] B. Barak. How to go beyond the black-box simulation barrier. In *Proc. 42nd FOCS*, pages 106–115. IEEE, 2001.

[4] A. Blum, M. L. Furst, M. J. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO '93*, pages 278–291, 1993. ISBN 3-540-57766-1.

[5] V. Feldman. On the power of membership queries in agnostic learning. *J. Mach. Learn. Res.*, 10:163–182, 2009. ISSN 1532-4435.

[6] V. Feldman and S. Shah. Separating models of learning with faulty teachers. *Theor. Comput. Sci.*, 410(19):1903–1912, 2009. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/j.tcs.2009.01.017.

[7] Y. Freund. Boosting a weak learning algorithm by majority. In *Proc. COLT '90*, pages 202–216, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. ISBN 1-55860-146-5.

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting,. *Journal of Comp. and Sys. Sci.*, 55(1):119–139, 1997.

[9] O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001. Earlier version available on http://www.wisdom.weizmann.ac.il/~oded/frag.html .

[10] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. ISSN 0004-5411. Preliminary version in FOCS' 84.

[11] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. of Com.*, 28(4):1364–1396, 1999. Preliminary versions appeared in STOC' 89 and STOC' 90.

[12] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *Proc. 30th FOCS*, pages 230–235, 1989.

[13] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC '89*, pages 44–61. ACM, 1989. ISBN 0-89791-307-8. doi: http://doi.acm.org/10.1145/73007.73012.

[14] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *Proc. STOC '89*, pages 433–444, New York, NY, USA, 1989. ACM. ISBN 0-89791-307-8. doi: http://doi.acm.org/10.1145/73007.73049.

[15] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282, New York, NY, USA, 1994. ACM. ISBN 0-89791-663-8. doi: http://doi.acm.org/10.1145/195058.195155.

[16] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *COLT '92*, pages 341–352, 1992. ISBN 0-89791-497-X. doi: http://doi.acm.org/10.1145/130385.130424.

[17] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. of Com.*, 17(2):373–386, 1988. Preliminary version in STOC' 86.

[18] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proc. 31st FOCS*, pages 2–10. IEEE, 1990.

[19] M. Naor. Evaluation may be easier than generation (extended abstract). In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 74–83, New York, NY, USA, 1996. ACM. ISBN 0-89791-785-5. doi: http://doi.acm.org/10.1145/237814.237833.

[20] R. Ostrovsky and A. Wigderson. One-way functions are essential for non-trivial zero-knowledge. Technical Report TR-93-073, International Computer Science Institute, Berkeley, CA, Nov. 1993. Preliminary version in Proc. 2nd Israeli Symp. on Theory of Computing and Systems, 1993, pp. 3–17.

[21] L. Pitt and M. K. Warmuth. Prediction-preserving reducibility. *J. Comput. Syst. Sci.*, 41 (3):430–467, 1990. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/0022-0000(90)90028-J.

[22] R. Schapire. The strength of weak learnability. *Proc. FOCS '89*, pages 28–33, 1989. doi: http://doi.ieeecomputersociety.org/10.1109/SFCS.1989.63451.

[23] A. Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, 1992. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/146585.146609.

[24] H.-U. Simon. How many missing answers can be tolerated by query learners? In *STACS '02: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 384–395, London, UK, 2002. Springer-Verlag. ISBN 3-540-43283-3.

[25] S. P. Vadhan. An unconditional study of computational zero knowledge. *FOCS '04*, pages 176–185, 2004. ISSN 0272-5428. doi: http://doi.ieeecomputersociety.org/10.1109/FOCS.2004.13.

[26] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1968.1972.

[27] A. Wigderson. **P**, **NP**, and mathematics - a computational complexity perspective. In *Proceedings of the ICM '06*, volume 1, pages 665–712, Zurich, Switzerland, 2006. EMS Publishing House.

[28] D. Xiao. On basing $\mathbf{ZK} \neq \mathbf{BPP}$ on the hardness of pac learning. In *In Proc. CCC '09*, pages 304–315, 2009.

[29] D. Xiao. *New Perspectives on the Complexity of Computational Learning, and Other Problems in Theoretical Computer Science*. PhD thesis, Princeton University, 2009.

A Omitted Proofs

A.1 Maximum Likelihood

Proof of Claim 5.4. Let $m = \mathrm{poly}(n)$ be an integer such that $|\mathcal{D}| \le 2^m$ and for all $D \in \mathcal{D}$ and $x \in \mathrm{supp}(D)$, it holds that $\Pr[D = x] \ge 2^{-m}$. Define $\mathsf{wt}_D(x) = -\log \Pr[D = x]$. Given a set of examples $x_1, \ldots, x_t$, finding the D that maximizes $\Pr[D^t = (x_1, \ldots, x_t)]$ is equivalent to finding the D that minimizes $\frac{1}{n} \sum_{i=1}^{t} \mathsf{wt}_D(x)$.

Define the KL divergence $\mathrm{KL}(D_1 \| D_2) = \mathbb{E}_{x \leftarrow_{\mathrm{R}} D_1}[\mathsf{wt}_{D_2}(x) - \mathsf{wt}_{D_1}(x)]$. It holds that $\mathrm{KL}(D_1 \| D_1) = 0$ and Pinsker's inequality says that $\Delta(D_1, D_2) \le \sqrt{\mathrm{KL}(D_1 \| D_2)/2}$, therefore $\Delta(D_1, D_2) > \varepsilon$ implies $\mathrm{KL}(D_1, D_2) > 2\varepsilon^2$.

We wish to compute:

$$\Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}[\mathrm{ML}_{\mathcal{D}}(x_1, \ldots, x_t) = D' \ \wedge\ \Delta(D', D) > \varepsilon] \tag{1.1}$$

$$\le \Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}[\mathrm{ML}_{\mathcal{D}}(x_1, \ldots, x_t) = D' \ \wedge\ \mathrm{KL}(D' \| D) > 2\varepsilon^2] \tag{1.2}$$

$$\le \Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}\left[\exists D',\ \mathrm{KL}(D' \| D) > 2\varepsilon^2 \ \wedge\ \frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_2}(x_i) \le \frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_1}(x_i)\right] \tag{1.3}$$

Let $\mu = \mathbb{E}_{x \leftarrow_{\mathrm{R}} D}[\mathsf{wt}(x)]$, then it follows that the previous is bounded by:

$$\le \Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}\left[\exists D',\ \mathrm{KL}(D' \| D) > 2\varepsilon^2 \ \wedge\ \left(\frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_2}(x_i) < \mu + \varepsilon^2 \quad \vee \quad \frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_1}(x_i) > \mu + \varepsilon^2\right)\right] \tag{1.4}$$

$$\le 2^m \Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}\left[\frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_2}(x_i) < \mu + \varepsilon^2 \quad \vee \quad \frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_1}(x_i) > \mu + \varepsilon^2 \quad \middle| \quad \mathrm{KL}(D' \| D) > 2\varepsilon^2\right] \tag{1.5}$$

The condition $\mathrm{KL}(D' \| D) > 2\varepsilon^2$ is equivalent to saying

$$\mathbb{E}_{x \leftarrow_{\mathrm{R}} D_1}[\mathsf{wt}_{D_2}(x)] - \mathbb{E}_{x \leftarrow_{\mathrm{R}} D_1}[\mathsf{wt}_{D_1}(x)] \ge 2\varepsilon^2 \quad \Leftrightarrow \quad \mathbb{E}_{x \leftarrow_{\mathrm{R}} D_1}[\mathsf{wt}_{D_2}(x)] \ge \mu + 2\varepsilon^2$$

Since by hypothesis $\mathsf{wt}_D(x) \in [0, m]$ for all D, x, therefore we may apply Chernoff to conclude that

$$\Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}\left[\frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_2}(x_i) < \mu + \varepsilon^2\right] \le 2^{-(\varepsilon/m)^4 t/2}$$
$$\Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}\left[\frac{1}{t}\sum_{i=1}^{t} \mathsf{wt}_{D_1}(x_i) > \mu + \varepsilon^2\right] \le 2^{-(\varepsilon/m)^4 t/2}$$

Applying this to Equation 1.5 implies that

$$\Pr_{x_1,\ldots,x_t \leftarrow_{\mathrm{R}} D}[\mathrm{ML}_{\mathcal{D}}(x_1, \ldots, x_t) = D' \ \wedge\ \Delta(D', D) > \varepsilon] \le 2^{m+1-(\varepsilon/m)^4 t/2}$$

Setting $t = 2(m/\varepsilon)^4(n + 1 + m)$ implies the lemma. ■

A.2 Computing Maximum Likelihood

Proof of Claim 5.5. Fix D that is samplable by a circuit $C \in \mathsf{SIZE}^{\mathbf{PSPACE}}(\mathrm{poly}(n))$, say $C : \{0,1\}^m \to \{0,1\}^n$. Observe that C can be evaluated in polynomial space.

Suppose that we are given $x_1, \ldots, x_t$ and we want to find $\mathrm{ML}_{\mathcal{D}_{q,n}}(x_1, \ldots, x_t)$. One can compute in polynomial space $s = \{r_1, \ldots, r_t \mid \forall i \in [t],\ C(r_i) = x_i\}$, which immediately give $\Pr[D^t = (x_1, \ldots, x_t)] = s/2^{mt}$. Furthermore, $|\mathcal{D}_{q,n}| \leq 2^{\mathrm{poly}(n)}$, therefore in polynomial space one can compute the $D \in \mathcal{D}_{q,n}$ that maximizes $\Pr[D^t = (x_1, \ldots, x_t)]$. Since this can all be done in polynomial space, this can be done in polynomial time using a **PSPACE** oracle. ■

B Chernoff bounds for smooth distributions

The standard Chernoff shows that the empirical average of many samples drawn from a distribution deviates from the mean of the distribution with exponentially small probability. We use the fact that this also holds for weighted empirical averages, as long as the weights are relatively smooth.

Lemma B.1 (Generalized Chernoff bound)**.** *Let D be a distribution over a finite universe U such that $\max_{u \in U} \Pr[D = u] \leq 1/k$ (equivalently, it has min-entropy $H_\infty(D) \geq \log k$). Let F be a distribution on functions $f : U \to \{0,1\}$. Let $\mu = \mathbb{E}_{D,F}[F(D)]$ and let $\mu_u = \mathbb{E}_F[F(u)]$. Then*

$$\Pr_F\left[\mathbb{E}_D[F(D)] > \mu + \gamma\right] < e^{-\gamma^2 k/2}$$

Proof. We derive that for any positive constant t:

$$\begin{aligned}
\Pr_F\left[\mathbb{E}_D[F(D)] > \mu + \gamma\right] &= \Pr_F\left[e^{t(k\mathbb{E}_D[F(D)] - k\mu)} > e^{tk\gamma}\right] \\
&\leq e^{-tk\gamma}\mathbb{E}_F\left[e^{t(k\mathbb{E}_D[F(D)] - k\mu)}\right] \\
&\leq e^{-tk\gamma}\mathbb{E}_F\left[e^{t(k\mathbb{E}_D[F(D) - \mu_D])}\right] \\
&\leq e^{-tk\gamma}\mathbb{E}_F\left[e^{t(\sum_{u \in \mathrm{supp}(D)} F(u) - \mu_u)}\right] \qquad \textit{(using } \Pr[D = u] \leq 1/k\textit{)} \\
&= e^{-tk\gamma} \prod_{u \in \mathrm{supp}(D)} \mathbb{E}_F\left[e^{t(F(u) - \mu_u)}\right] \\
&\leq e^{-tk\gamma + t^2 k} \qquad \textit{(using } |\mathrm{supp}(D)| \geq k \textit{ plus Taylor expansion)} \\
&= e^{-\gamma^2 k/2}
\end{aligned}$$

where the last line follows from setting $t = \gamma/2$. ■