On distribution-specific learning with membership queries versus pseudorandom generation

Johannes Köbler and Wolfgang Lindner*

Humboldt-Universität zu Berlin
Institut für Informatik
D-10099 Berlin, Germany

Abstract. We consider a weak version of pseudorandom function generators and show that their existence is equivalent to the non-learnability of Boolean circuits in Valiant's pac-learning model with membership queries on the uniform distribution. Furthermore, we show that this equivalence holds still for the case of non-adaptive membership queries and for any (non-trivial) p-samplable distribution.

1 Introduction

In computational learning theory, many non-learnability results for (the representation independent version of) Valiant's pac-learning model are based on cryptographic tools and assumptions. Already in [19], Valiant pointed out how the results of [8] can be used to show that (arbitrary) Boolean circuits are not efficiently pac-learnable if cryptographically secure one-way functions exist. Related results can be found in, e.g., [11, 3, 1, 12, 13].

On the other hand, it is known [4] that the non-learnability (in polynomial-time) of Boolean circuits provides a sufficient condition under which $\mathcal{RP}$ and, hence, $\mathcal{P}$ is different from $\mathcal{NP}$. Recently, Impagliazzo and Wigderson [9] showed that every problem in $\mathcal{BPP}$ can be solved deterministically in sub-exponential time on almost every input (for infinitely many input lengths), provided that $\mathcal{EXP} \neq \mathcal{BPP}$. Since $\mathcal{EXP} = \mathcal{BPP}$ implies $\mathcal{NP} = \mathcal{RP}$, this means that the non-learnability of Boolean circuits can further serve as a hypothesis to achieve derandomization results for probabilistic polynomial-time computations.

Interestingly, both implications, namely that Boolean circuits are not learnable (if one-way functions exist) as well as the derandomization of $\mathcal{BPP}$ (if Boolean circuits are not learnable) are based on the concept of *pseudorandom generation*, i.e., the possibility to expand a small number of random bits (also known as the *seed*) into a large amount of *pseudorandom* bits which cannot be significantly distinguished from truely random bits by any polynomial-time computation. An important difference between these two implications is, however, that in the first case the pseudorandom generator has to run in polynomial

* Supported by DFG grant Ko1053/1-1.

time, while for the derandomization of $\mathcal{BPP}$ it suffices that the pseudorandom generator runs in exponential time.

A close connection between pseudorandom generation and non-learnability has been demonstrated already in [8], where the existence of a pseudorandom *function* generator (see Section 3) is shown to be in fact equivalent to the non-learnability of Boolean circuits. This, however, only holds if we consider learning agents that are successful with high probability for a randomly chosen target (instead of being successful for *all* targets as in the standard model of pac-learning). Furthermore, in the model of [8], the learning task has to be accomplished with the help of membership queries and with respect to the uniform distribution. Recall that the original model of [19] is *distribution-free* and *passive*, i.e., the learning algorithm has to be successful for any (unknown) distribution $\mathcal{D}$, and access to the target is given via random *examples* chosen according to $\mathcal{D}$ together with their classification by the target. Further connections between the average-case model of pac-learning and cryptographic primitives have been found in [2].

Recently [14], also the worst-case model of (passive) pac-learning under the uniform distribution has been closely related to pseudorandom function generation. In contrast to [8] the pseudorandomness condition in [14] is expressed in terms of the *worst-case* advantage of the distinguishing algorithms (see section 3). Moreover, the distinguishing algorithms for the function generator in [14] are required to be passive, i.e., the distinguishing algorithm may access its oracle only via random classified examples. This leads to an apparently weaker notion of pseudorandom function generation as compared to the standard definition [8].

Here we take a similar approach as in [14] for pac-learning *with* membership queries under the uniform distribution [1, 12, 10]. As our main result we show that the non-learnability of Boolean circuits is equivalent to the existence of a weak pseudorandom function generator, where as in [14] the pseudorandomness condition is expressed in terms of worst-case advantage of distinguishing algorithms. Further we show that this kind of weak pseudorandom generation is still strong enough to yield a derandomization result for $\mathcal{RP}$. In contrast to the above mentioned derandomization for $\mathcal{BPP}$, the function generator that we use is polynomial-time computable (instead of exponential time). Hence, we even get that every problem in $\mathcal{RP}$ can be solved nondeterministically in polynomial-time on almost every input (for infinitely many input lengths), where for every given $\epsilon > 0$, only n^ϵ many nondeterministic bits are used. Of course, this implies that every problem in $\mathcal{RP}$ can also be solved deterministically in sub-exponential time on almost every input.

As an application, we get that every learning algorithm for Boolean circuits which uses membership queries and which is successful for some arbitrary p-samplable distribution can be transformed into a learning algorithm which uses only *non-adaptive* membership queries and is successful for the uniform distribution. Thus, if Boolean circuits are not learnable with non-adaptive membership queries under the uniform distribution, then Boolean circuits are not learnable with adaptive membership queries under any non-trivial [12] p-samplable distribution.

2 Preliminaries

We use $\mathcal{B}_n$ to denote the set of all Boolean functions $f : \{0,1\}^n \to \{0,1\}$.

A *probability ensemble* $\mathcal{D} = \{\mathcal{D}_n : n \geq 0\}$ is a sequence of probability distributions $\mathcal{D}_n$ on $\{0,1\}^n$. The ensemble $\mathcal{D}$ is *p-samplable* if there is a polynomial time computable function f and a polynomial p such that for all n, $f(X)$ is distributed according to $\mathcal{D}_n$ when X is uniformly distributed over $\{0,1\}^{p(n)}$.

For a polynomially bounded function $f : \mathbb{N} \to \mathbb{R}$, let $\mathcal{NP}[f]$ [7] denote the class of all sets $A \in \mathcal{NP}$ for which there exists a set $B \in \mathcal{P}$ such that for all strings x, $x \in A \Leftrightarrow \exists y \in \{0,1\}^{\lceil f(|x|)\rceil} : (x,y) \in B$. For a function $\epsilon : \mathbb{N} \to [0,1]$ and any class $\mathcal{C}$ of sets, $\mathcal{H}eur_{\epsilon(n)}\mathcal{C}$ denotes the class of all sets A such that for any p-samplable distribution ensemble $\mathcal{D}$ there is a set $A' \in \mathcal{C}$ such that for all n and for $X \in_{\mathcal{D}_n} \{0,1\}^n$, $\mathsf{Pr}\,(A(X) \neq A'(X)) < \epsilon(n)$. Here, $A(x)$ is used to denote the characteristic function of a set A.

Let X and Y be independent and identically distributed random variables on $\{0,1\}^n$. The *Renyi entropy* of X is $\text{ent}_{Ren}(X) = -\log(\mathsf{Pr}\,(X = Y))$, and the *minimum entropy* of X is $\text{ent}_{min}(X) = \min\{-\log(\mathsf{Pr}\,(X = x) : x \in \{0,1\}^n\}$. Note that for any random variable X defined on $\{0,1\}^n$, $\text{ent}_{Ren}(X)/2 \leq \text{ent}_{min}(X) \leq \text{ent}_{Ren}(X)$. The Renyi entropy of a distribution $\mathcal{D}$ is the Renyi entropy of a random variable X chosen according to $\mathcal{D}$.

2.1 Predictability

We recall the learning model of efficient prediction with membership queries (cf. [1, 12]). A *representation of concepts* C is any subset of $\{0,1\}^* \times \{0,1\}^*$. A pair (u,x) of $\{0,1\}^* \times \{0,1\}^*$ is interpreted as consisting of a *concept name* u and an *example* x. The *concept* represented by u is $\kappa_C(u) = \{x : (u,x) \in C\}$.

A *prediction with membership queries algorithm*, or *pwm-algorithm*, is a possibly randomized algorithm A that takes as input two positive integers s and n, where s is the length of the target concept name and n is the length of examples n as well as a rational *accuracy bound* ϵ. It may make three different kinds of oracle calls, the responses to which are determined by the unknown target concept $c = \kappa_C(u)$ with $|u| = s$, and an unknown distribution $\mathcal{D}_n$ on $\{0,1\}^n$ as follows.

1. A *membership query* takes a string $x \in \{0,1\}^*$ as input and returns 1 if $x \in c$ and 0 otherwise.
2. A *request for a random classified example* takes no input and returns a pair (x,b), where x is a string chosen according to $\mathcal{D}_n$, and $b = 1$ if $x \in c$ and $b = 0$ otherwise.
3. A *request for an element to predict* takes no input and returns a string x chosen according to $\mathcal{D}_n$.

The algorithm A may make any number of membership queries or requests for random classified examples. However, A must eventually make one and only one request for an element to predict and then eventually halt with an output of 1 or 0 without making any further oracle calls.

A pwm-algorithm A is said to *ϵ-predict* $\kappa_C(u)$ on $\mathcal{D}_n$ if on input $s = |u|$, n and ϵ, when A is run with respect to the target concept $\kappa_C(u)$ and distribution $\mathcal{D}_n$, the probability is at most ϵ that the output of A is not equal to the correct classification of X by $\kappa_C(u)$, where $X \in_{\mathcal{D}_n} \{0,1\}^n$ is the request for an element to predict. Furthermore, a pwm-algorithm A is said to run in *polynomial time* if its running time is bounded by a polynomial in s, n, and $1/\epsilon$.

Definition 1. *Let C be a representation of concepts, and let $\mathcal{D} = \{\mathcal{D}_n : n \geq 0\}$ be a probability ensemble. Then C is* predictable on $\mathcal{D}$ with membership queries in polynomial time *if there exists a polynomial-time pwm-algorithm A such that for all positive integers s and n, for all concept names u of length s, and for all positive rationals ϵ, A ϵ-predicts $\kappa_C(u)$ on $\mathcal{D}_n$.*

Recall that in the *weak* model of learning [11] the learning algorithm has to predict the target only slightly better than a completely random guess.

Definition 2. *Let C be a representation of concepts, and let $\mathcal{D}$ be a probability ensemble. Then C is* weakly *predictable on $\mathcal{D}$ with membership queries in polynomial time if there exists a constant $c > 0$ and a polynomial-time pwm-algorithm A such that for all positive integers s and n, and for all concept names u of length s, A $(\frac{1}{2} - \frac{1}{(sn)^c})$-predicts $\kappa_C(u)$ on $\mathcal{D}_n$.*

In the distribution-free model of learning, weak and strong learning are shown to be equivalent in [18]. Based on Yao's XOR lemma [20], a similar result is shown in [5] also for learning on the uniform distribution.

Theorem 1. *[5] For every representation of concepts $C \in \mathcal{P}$ there exists a representation of concepts $C' \leq_{tt}^{p} C$ such that if C' is weakly predictable on the uniform distribution (with membership queries) in polynomial time, then also C is predictable on the uniform distribution (with membership queries) in polynomial time.*

We also consider predictability with *non-adaptive* membership queries, where by a non-adaptive membership query we mean a query that does not depend on the responses to previous queries. It may, however, still depend on the random coin tosses used by the prediction algorithm. Note that Theorem 1 holds also for non-adaptive membership queries.

Recall that in the pac-learning model the learning algorithm is required to output a concept name which approximates the target well rather than to guess the correct classification of the target by itself. For boolean circuits, however, polynomial-time pac-learnability and predictability coincide. Furthermore, if there exists some representation of concepts $C \in \mathcal{P}$ that is not predictable in polynomial time then boolean circuits are not predictable in polynomial time.

3 Weak pseudorandom generators

In this section we introduce our weak version of a pseudorandom function generator. But first let us recall the standard definition. Let $f : \{0,1\}^{l(n)} \times \{0,1\}^n \rightarrow$

$\{0,1\}$ be a polynomial-time computable function ensemble. Note that the polynomial-time computability of f implies that l is polynomially bounded. For a fixed string x of length $l(n)$ we can view $f(x,\cdot)$ as a function $f_x \in \mathcal{B}_n$ that is generated by f on the *seed* x, and therefore we refer to f also as a *function generator*. Now f is a *pseudorandom function generator* if the function f_X produced by f for a random seed $X \in_{\mathcal{U}} \{0,1\}^{l(n)}$ cannot be significantly distinguished from a truely random function $F \in_{\mathcal{U}} \mathcal{B}_n$ by any polynomial-time computation, i.e., for all probabilistic polynomial-time oracle algorithms T, for all positive integers c, and for infinitely many integers n, the *success*

$$\delta(n) = |\operatorname{Pr}\left(T^{f_X}(0^n) = 1\right) - \operatorname{Pr}\left(T^F(0^n) = 1\right)|$$

of T for f is less than $\frac{1}{n^c}$. The algorithm T is also called a *distinguishing algorithm* or *test* for f. Our definition of a weak pseudorandom function generator is based on the *advantage* of a test T for f (cf. [14]) which is defined with respect to a *fixed* seed $x \in \{0,1\}^{l(n)}$ as

$$\epsilon(x) = \operatorname{Pr}\left(T^{f_x}(0^n) = 1\right) - \operatorname{Pr}\left(T^F(0^n) = 1\right).$$

Note that the success of T for f can be expressed by the average advantage of T for f as $\delta(n) = |\operatorname{E}(\epsilon(X))|$ for a random seed $X \in_{\mathcal{U}} \{0,1\}^{l(n)}$. So we refer to $\epsilon(n) = \min\{\epsilon(x) : x \in \{0,1\}^{l(n)}\}$ as the *worst-case advantage* of T for f.

Definition 3. *A* weak pseudorandom function generator *is a polynomial-time computable function ensemble $f : \{0,1\}^{l(n)} \times \{0,1\}^n \to \{0,1\}$, such that for all probabilistic polynomial-time oracle algorithms T, for all positive integers c, and for infinitely many integers n, the worst-case advantage of T for f is less than $\frac{1}{n^c}$.*

As a first property let us mention that a weak pseudorandom function generator is still useful for derandomizing probabilistic polynomial-time computations. We omit the proof.

Proposition 1. *If there exists a weak pseudorandom function generator, then for any constant $c \geq 1$, $\mathcal{RP} \subseteq$ io-$\mathcal{H}eur_{n^{-c}}\mathcal{NP}[n^{1/c}]$.*

As opposed to a pseudorandom generator, which has to expand its seed only by at least one bit, we can consider a pseudorandom function generator as a function which expands a seed of polynomial length into a pseudorandom bit-sequence of length 2^n. In the standard setting it is well known that a pseudorandom generator can be used to construct a pseudorandom function generator [8]. It is thus an immediate question whether a similar fact can also be shown with respect to worst-case advantage. Unfortunately, we are not able to answer this question completely. We can show, however, that there exists a weak pseudorandom function generator if there exist weak pseudorandom generators with an arbitrary polynomial expansion.

The proof is based on universal hashing, an ubiquitous tool in cryptography. A *linear hash function* h from $\{0,1\}^n$ to $\{0,1\}^m$ is given by a Boolean $n \times m$

matrix (a_{ij}), (or, equivalently, by a mn-bit string $a_{1,1} \dots a_{1,m} \dots a_{n,1} \dots a_{n,m}$) and maps any string $x = x_1 \dots x_n$ to a string $y = y_1 \dots y_m$, where y_i is the inner product $a_i \cdot x = \sum_{j=1}^{n} a_{ij} x_j \pmod 2$ of the i-th row a_i and x. In [6] it is shown that the set of all linear hash functions from $\{0,1\}^n$ to $\{0,1\}^m$ is *universal*, i.e., for all n-bit strings x and y with $x \neq y$, $\Pr(H(x) = H(y)) = \frac{1}{2^m}$, when H is uniformly at random chosen from the set of all linear hash functions from $\{0,1\}^n$ to $\{0,1\}^m$. Now let us say that a hash function h from $\{0,1\}^n$ to $\{0,1\}^m$ *causes a collision* on a set $Q \subseteq \{0,1\}^n$ if there exist two different strings x and y in $\{0,1\}^m$ such that $h(x) = h(y)$. Otherwise, h is said to be *collision-free on* Q. Then a random linear hash function causes a collision on Q with probability at most $\frac{|Q|(|Q|-1)}{m}$.

Lemma 1. *Suppose that there exists a family $\{g_k : k \geq 1\}$ of function ensembles $g_k : \{0,1\}^n \to \{0,1\}^{n^k}$ with the following properties:*

1. *The i-th bit of $g_k(x)$ is computable in time polynomial in $|x|$ and k.*
2. *For all positive integers k and c, for all probabilistic polynomial-time algorithms T, and for infinitely many integers n, there exists some $x \in \{0,1\}^n$ such that for $Y \in_{\mathcal{U}} \{0,1\}^{n^k}$,*

$$\Pr(T(g(x)) = 1) - \Pr(T(Y) = 1) < \frac{1}{n^c}.$$

Then there exists a weak pseudorandom function generator.

Proof. Based on the family $\{g_k : k \geq 1\}$ we define a function generator f as follows. Given binary strings x and z of length n, and further two binary strings k and h of length $\log n$ and $n^2 \log n$, respectively, we think of k as encoding an integer $k \in \{1, \dots, n\}$, and of h as a linear hash function mapping strings of length n to strings of length $m = \lfloor \log n^k \rfloor$ (by using only the first nm bits of the string h). Then we can interpret the m-bit string $h(z)$ as a positive integer $h(z) \in \{1, \dots, n^k\}$, and define

$$f(k \circ h \circ x, z) = g_k(x)_{\{h(z)\}},$$

where $g_k(x)_{\{h(z)\}}$ denotes the $h(z)$-th bit of the n^k-bit string $g_k(x)$.

By the computability condition on the family $\{g_k : k \geq 1\}$ it follows that f is polynomial-time computable. To see that f is a weak pseudorandom function generator, assume to the contrary that there exist some $c \geq 1$ and a probabilistic oracle test T whose running-time on input 0^n and with any oracle in $\mathcal{B}_n$ is bounded by n^c, and such that for all (sufficiently large) integers n, the worst-case advantage of T for f is at least $\frac{1}{n^c}$. Note that this implies that for all $k \in \{0,1\}^{\log n}$ and $x \in \{0,1\}^n$, and for random $H \in_{\mathcal{U}} \{0,1\}^{n^2 \log n}$ and $F \in_{\mathcal{U}} \mathcal{B}_n$,

$$\Pr\left(T^{f_{k \circ H \circ x}}(0^n) = 1\right) - \Pr\left(T^F(0^n) = 1\right) \geq \frac{1}{n^c}.$$

Now fix $k = 4c$ and consider the test T' for g_k which on input $y \in \{0,1\}^{n^k}$ works as follows. First choose $h \in_{\mathcal{U}} \{0,1\}^{n^2 \log n}$. Then simulate T on input 0^n, where each query $z \in \{0,1\}^n$ is answered by the $h(z)$-th bit of y. Finally accept if and only if T accepts.

Under the condition that the hash function h chosen by T' is collision-free on the set of queries Q asked by T, the test T' accepts a random $Y \in_{\mathcal{U}} \{0,1\}^{n^k}$ with the same probability as T accepts with a random oracle $F \in_{\mathcal{U}} \mathcal{B}_n$. Even though the queries in Q might depend on h for some specific $y \in \{0,1\}^{n^k}$, it is not hard to see that for the random Y and H, the probability that H is collision-free on Q coincides with the probability that H is collision-free on the set of queries asked by T when T is run with the random oracle F. It follows that H causes a collision on Q with probability at most

$$\frac{|Q|^2}{2^m} \leq \frac{n^{2c}}{2^m} < \frac{1}{2n^c},$$

where the last inequality follows by the choice of $k = 4c$ which implies that $2^m \geq n^k/2 = n^{4c}/2 > 2n^{3c}$, and hence we have

$$|\Pr(T'(Y)=1) - \Pr\left(T^F(0^n)=1\right)| \leq \frac{1}{2n^c}.$$

On the other hand, for all $x \in \{0,1\}^n$, the probability that T' accepts $g_k(x)$ is just the probability that T accepts with oracle $f_{k \circ H \circ x}$ for a random hash function $H \in \{0,1\}^{n^2 \log n}$. It follows that for all $x \in \{0,1\}^n$,

$$\Pr(T'(g_k(x))=1) - \Pr(T'(Y)=1) \geq \frac{1}{2n^c}.$$

But this contradicts the pseudorandomness condition of the lemma. □

4 Weak pseudorandomness versus predictability

In this section we show that there exists some non-predictable representation of concepts $C \in \mathcal{P}$ if and only if there exists a weak pseudorandom function generator. The implication from non-predictability to the weak function generator is based on the (well-known) construction of a pseudorandom generator due to Nisan and Wigderson [17].

Definition 4. *A* (r,l,n,k)-design *is a collection* $\mathcal{S} = (S_1, \ldots, S_r)$ *of sets* $S_i \subseteq \{1, \ldots, l\}$*, each of which has cardinality* n*, such that for all* $i \neq j$*,* $|S_i \cap S_j| \leq k$*. Given a function* $f : \{0,1\}^n \to \{0,1\}$*, the* Nisan-Wigderson generator (based on f and $\mathcal{S}$)*,* $f^{\mathcal{S}} : \{0,1\}^l \to \{0,1\}^r$*, is for every seed* $x = x_1 \ldots x_l$ *of length* l *defined as*

$$f^{\mathcal{S}}(x) = f(x_{S_1}) \ldots f(x_{S_r}),$$

where x_{S_i}*, for* $1 \leq i \leq r$*, denotes the restriction of* x *to* $S_i = \{i_1 < \ldots < i_n\}$ *defined as* $x_{S_i} = x_{i_1} \ldots x_{i_n}$*.*

It is shown in [17] that for all positive integers r and n we can use polynomials of degree at most $\log r$ over a suitably chosen field to construct a $(r, 4n^2, n, \log r)$-design $\mathcal{S} = (S_1, \ldots, S_r)$ such that each S_i can be computed in time polynomial in n and $\log r$ [17]. In the following, we refer to this design as the *low-degree polynomial design*.

Furthermore, as shown in [17], $f^{\mathcal{S}}$ is a pseudorandom generator with respect to non-uniformly computable distinguishing algorithms, provided that the function f is hard to approximate by polynomial-size circuits. This means that if there is a polynomial-size circuit T with sufficiently large distinguishing success for $f^{\mathcal{S}}$, then there is a polynomial-size circuit T' that approximates f. It is known that in fact T' can be uniformly obtained from T, though at the expense of polynomially many membership queries to f (cf. [2, 9]). By inspecting the proof in, e.g., [9] it is not hard to see that the required membership queries do not depend on the answers to previously asked queries, i.e., T' can be obtained from T by using only non-adaptive queries to f.

Lemma 2. *[17, 2, 9] There exists a probabilistic oracle algorithm A which given as input an integer n, a circuit T with input length r, a rational $\epsilon > 0$, and further access to a function $f : \{0,1\}^n \to \{0,1\}$ computes with probability at least $1 - \epsilon$ a circuit T' such that for $Z \in_{\mathcal{U}} \{0,1\}^n$,*

$$\mathsf{Pr}\left(T'(Z) = f(Z)\right) \geq \frac{1}{2} + \frac{\delta}{r} - \epsilon,$$

provided that for $X \in_{\mathcal{U}} \{0,1\}^{4n^2}$, $Y \in_{\mathcal{U}} \{0,1\}^r$, and the low-degree polynomial $(r, 4n^2, n, \log r)$-design $\mathcal{S}$,

$$|\,\mathsf{Pr}\left(T(f^{\mathcal{S}}(X)) = 1\right) - \mathsf{Pr}\left(T(Y) = 1\right)| \geq \delta.$$

Moreover, A runs in time polynomial in n, $|T|$ and $1/\epsilon$, and A asks only non-adaptive oracle queries to f.

Now, based on Lemmas 1 and 2 we can construct a weak pseudorandom function generator under non-predictability assumption.

Theorem 2. *If there exists a representation of concepts $C \in \mathcal{P}$ that is not weakly predictable with non-adaptive membership queries on the uniform distribution in polynomial time, then there exists a polynomial-time computable weak pseudorandom function generator.*

Proof. Let $l(n) = 4n^2$. W.l.o.g. we assume that also the $l(n)$-size bounded restriction of C defined as $C_{l(n)} = \{(u, x) \in C : |u| = l(|x|)\}$ is not weakly predictable with non-adaptive membership queries on the uniform distribution in polynomial time. (Otherwise this can be achieved by a simple padding argument.) Based on $C_{l(n)}$ we define a family $\{g_k : k \geq 1\}$ of function ensembles $g_k : \{0,1\}^{2l(n)} \to \{0,1\}^{n^k}$ satisfying the conditions of Lemma 1. The theorem will follow.

For all positive integers n and k with $k \leq n$, and for all binary strings u and x of length $l(n)$ define

$$g_k(u \circ x) = u_n^{\mathcal{S}}(x),$$

where u_n denotes the characteristic function of the set $\kappa_C(u) \cap \{0,1\}^n$, and $\mathcal{S} = (S_1, \ldots, S_{n^k})$ is the low-degree polynomial $(n^k, 4n^2, n, \log n^k)$-design. For $k > n$, put $g_k(u \circ x) = 0^{n^k}$.

Since $C \in \mathcal{P}$, and since each S_i for $i = 1, \ldots, n^k$ can be computed in time polynomial in n and $\log n^k$, it follows that the family $\{g_k : k \geq 1\}$ satisfies the computability condition of Lemma 1.

To see that the family $\{g_k : k \geq 1\}$ also satisfies the pseudorandomness condition of Lemma 1, assume to the contrary that there exist positive integers k and c and a polynomial-time computable test T, such that for all sufficiently large n, for all binary strings u and x of length $l(n)$, and for $Y \in_{\mathcal{U}} \{0,1\}^{n^k}$ it holds that

$$\mathsf{Pr}\left(T(g_k(u \circ x)) = 1\right) - \mathsf{Pr}\left(T(Y) = 1\right) \geq \frac{1}{n^c}.$$

Note that this implies that the test T distinguishes the random variable $g_k(U \circ X)$ with $U \in_{\mathcal{U}} \{0,1\}^{l(n)}$ and $X \in_{\mathcal{U}} \{0,1\}^{l(n)}$ from a uniformly distributed $Y \in_{\mathcal{U}} \{0,1\}^{n^k}$ by at least $\frac{1}{n^c}$.

Now consider the pwm-algorithm A for $C_{l(n)}$, which on input $s = l(n)$ and n and with respect to some concept $\kappa_C(u)$ with $|u| = l(n)$ works as follows. First obtain a probabilistic circuit T_{n^k} that computes T on input length n^k by using a (finite) description of the Turing machine computing T. Then run the algorithm of Lemma 2 with the circuit T_{n^k}, oracle u_n and parameter $\epsilon = \frac{1}{4n^{k+c}}$ to obtain a circuit T'. Finally request an element z to predict, and answer with $T'(z)$.

By the assumption on the test T, the algorithm of Lemma 2 produces, for all n and concept names u of length $l(n)$, and with probability at least $1 - \epsilon$, a circuit T' satisfying

$$\mathsf{Pr}\left(T'(Z) = u_n(Z)\right) \geq \frac{1}{2} + \frac{1}{n^{c+k}} - \epsilon$$

for $Z \in_{\mathcal{U}} \{0,1\}^n$. This implies that A's guess for the classification of Z by $\kappa_C(u)$ is correct with probability at least

$$\frac{1}{2} + \frac{1}{n^{c+k}} - 2\epsilon = \frac{1}{2} + \frac{1}{2n^{c+k}}.$$

Further note that A makes no additional membership queries except the non-adaptive queries to $\kappa_C(u)$ required to produce the circuit T'. Thus, $C_{l(n)}$ is weakly predictable with non-adaptive membership queries on the uniform distribution in polynomial time. This, however, contradicts our assumption on C, and hence, g_k is a weak pseudorandom generator for all $k \geq 1$. The theorem now follows from Lemma 1. $\square$

We now proceed to show the converse of Theorem 2 for any p-samplable distribution, provided that the distribution in question does not immediately imply a trivial prediction algorithm. Kharitonov [12] showed that any representation of concepts C is polynomially weakly predictable on any distribution (ensemble) $\mathcal{D} = \{\mathcal{D}_n\}$ with Renyi entropy $O(\log n)$. This motivates the following definition.

Definition 5. *[12] A distribution ensemble $\mathcal{D} = \{\mathcal{D}_n\}$ is* trivial *if for all $n \geq 1$, $\mathcal{D}_n$ has Renyi entropy $O(\log n)$.*

A function generator $f : \{0,1\}^{l(n)} \times \{0,1\}^n \to \{0,1\}$ is associated in a natural way with the representation of concepts $C^f = \{(x,z) : |x| = l(|z|), f(x,z) = 1\}$. Note that for a string x of length $l(n)$, f_x is just the characteristic function of the set $\kappa_C(x) \cap \{0,1\}^n$.

Theorem 3. *If $f : \{0,1\}^{l(n)} \times \{0,1\}^n \to \{0,1\}$ is a weak pseudorandom function generator, then C^f is not weakly predictable with membership queries on any non-trivial p-samplable distribution in polynomial time.*

Proof. Assume to the contrary that there exists a non-trivial p-samplable distribution $\mathcal{D}$ such that C^f is weakly predictable with membership queries on $\mathcal{D}$ in polynomial time, and let A be a pwm-algorithm and c be a positive integer such that for all $n \geq 1$, the running-time of A on inputs $s = l(n)$ and n is bounded by n^c, and such that for all strings x of length $l(n)$, A ϵ-predicts $\kappa_{C^f}(x)$ on $\mathcal{D}_n$ with $\epsilon = \frac{1}{2} - \frac{1}{n^c}$.

We now use A to construct a test T for f. Given input 0^n and an oracle $h : \{0,1\}^n \to \{0,1\}$, the test T simulates A with inputs n and $s = l(n)$. When A makes a membership query z, then T answers this query with $h(z)$. When A requests a random classified example or an element to predict, then T chooses a string $z \in_{\mathcal{D}_n} \{0,1\}^n$ and returns either the example $(z, h(z))$ or the prediction challenge z to A. Finally, T accepts if and only if the guess of A on the prediction challenge z coincides with $h(z)$.

Obviously, for all positive integers n and for all strings x of length $l(n)$, the test T accepts the oracle f_x with probability at least $\frac{1}{2} + \frac{1}{n^c}$. If on the other hand, the oracle for T is a random function $F \in_{\mathcal{U}} \mathcal{B}_n$, then there are two possibilities. If the prediction challenge z coincides with some previously seen labeled sample, then we have to assume that A's guess for z is correct. Since the running time of A, and hence the number of labeled samples it obtains (via example or membership queries) is bounded by n^c, and since $\mathcal{D}$ is a non-trivial distribution, this can happen for a random prediction challenge $Z \in_{\mathcal{D}_n} \{0,1\}^n$ only with probability at most $n^c \cdot 2^{-\operatorname{ent}_{min}(\mathcal{D}_n)} \leq n^c \cdot 2^{-\operatorname{ent}_{Ren}(\mathcal{D}_n)/2} < \frac{1}{2n^c}$. Otherwise, i.e. in the case where all previously seen examples are different from the prediction challenge z, the probability that A classifies z correctly is exactly $\frac{1}{2}$. It follows that T accepts the random oracle F with probability at most $\frac{1}{2} + \frac{1}{2n^c}$. Thus, for all strings x of length $l(n)$,

$$\mathsf{Pr}\left(T^{f_x}(0^n) = 1\right) - \mathsf{Pr}\left(T^F(0^n) = 1\right) \geq \frac{1}{2n^c},$$

contradicting the assumption that f is a weak pseudorandom function generator. □

Since Boolean circuits are not pac-learnable if and only if there exists a non-predictable representation of concepts $C \in \mathcal{P}$, we can combine Theorems 1, 2 and 3 to get the following corollary.

Corollary 1. *The following are equivalent.*

1. *There exists a weak pseudorandom function generator.*
2. *Boolean circuits are not pac-learnable with non-adaptive membership queries on the uniform distribution in polynomial time.*
3. *Boolean circuits are not pac-learnable with membership queries on any non-trivial p-samplable distribution in polynomial time.*

As an application of Corollary 1 to the theory of resource-bounded measure [16], let us mention that it is shown in [15] that if $\mathcal{P}/poly$ does not have p_2-measure zero , and furthermore $\mathcal{EXP} \neq \mathcal{MA}$, then Boolean circuits are not pac-learnable with non-adaptive queries on the uniform distribution in polynomial-time. By Corollary 1, this can be extended to pac-learnability with (adaptive) membership queries on arbitrary non-trivial p-samplable distributions.

Furthermore, by Proposition 1, the non-learnability of Boolean circuits with membership queries on any non-trivial p-samplable distribution implies that for any constant $c \geq 1$, $\mathcal{RP} \subseteq io\text{-}\mathcal{H}eur_{n^{-c}}\mathcal{NP}[n^{1/c}]$.

Finally let us remark that the existence of a weak pseudorandom function generator can be expressed also in terms of (standard) success instead of worst-case advantage. But then we need to consider non-uniformly computable function generators and, moreover, the function generator might depend on the (still) uniformly computable distinguishing algorithm. More formally, we say that a test T *breaks* a function generator f if for some positive integer c and for all sufficiently large n, the success of T for f is at least $1/n^c$. Now a slight modification of the proof of Theorem 2 together with Theorem 3 yields the following equivalence.

Corollary 2. *The following are equivalent.*

1. *There exists a weak pseudorandom function generator.*
2. *There is some polynomial size-bound $s(n)$ such that no probabilistic polynomial-time computable test can break every function generator computable by circuits of size $s(n)$.*

References

[1] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proc. 23rd ACM Symposium on Theory of Computing*, pages 444–454. ACM Press, 1991.

[2] A. Blum, M. Furst, M. Kearns, and R. J. Lipton. Cryptographic primitives based on hard learning problems. In *Proc. CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer-Verlag, 1994.

[3] A. L. Blum. Separating distribution-free and mistake-bound learning models over the boolean domain. In *Proc. 31st IEEE Symposium on the Foundations of Computer Science*, pages 211–218. IEEE Computer Society Press, 1990.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

[5] D. Boneh and R. J. Lipton. Amplification of weak learning under the uniform distribution. In *Proc. 6th ACM Conference on Computational Learning Theory*, pages 347–351. ACM Press, 1993.

[6] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[7] J. Díaz and J. Torán. Classes of bounded nondeterminism. *Mathematical Systems Theory*, 23(1):21–32, 1990.

[8] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[9] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proc. 39th IEEE Symposium on the Foundations of Computer Science*, pages 734–743. IEEE Computer Society Press, 1998.

[10] J. C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.

[11] M. J. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41:67–95, 1994.

[12] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 372–381, 1993.

[13] M. Kharitonov. Cryptographic lower bounds for learnability of boolean functions on the uniform distribution. *Journal of Computer and System Sciences*, 50:600–610, 1995.

[14] M. Krause and S. Lucks. On learning versus distinguishing and the minimal hardware complexity of pseudorandom function generators. Technical Report TR00-014, Electronic Colloquium on Computational Complexity, 2000.

[15] W. Lindner, R. Schuler, and O. Watanabe. Resource-bounded measure and learnability. In *Proc. 13th Annual IEEE Conference on Computational Complexity*, pages 261–270, 1998.

[16] J. H. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences*, 44:220–258, 1992.

[17] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[18] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–226, 1990.

[19] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[20] A. C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd IEEE Symposium on the Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, 1982.