Electronic Colloquium on Computational Complexity, Report No. 132 (2021)

ECCC

Pseudo-random functions and uniform learnability

Eric R. Binnendyk

May 2021

Abstract

Boolean circuits are a model of computation. A class of Boolean circuits is called a polynomial class if the number of nodes is bounded by a polynomial function of the number of input variables. A class $C_n[s(n)]$ of Boolean functions is called learnable if there are algorithms that can approximate functions in $C_n[s(n)]$ when given oracle access to the function. A distribution D of functions is called pseudorandom against a circuit class $C[t(n)]$ if any oracle circuit C^f from $C[t(n)]$ outputs 1 with the same probability if the oracle f is chosen from D as it would if the oracle were random. It is known that a polynomial class of circuits is learnable if and only if it contains no pseudorandom distributions of functions. However, there is no known efficient algorithm to produce the learner given the number of inputs the circuits have (the learner is non-uniform). In this paper we use a uniform version of the minmax theorem to prove the existence of uniform learners under certain conditions.

1 Introduction

We aim to understand the constructive relationships between pseudorandomness and learnability for concept classes. It is well-known that pseudorandomness and learnability are closely related to each other. In the non-uniform setting, there is a dichotomy between learnability and pseudorandomness, i.e., duality between "inability to simulate randomness" by a class and its "feasible learnability". We want to explore this dichotomy in the uniform setting.

In the non-uniform setting, Λ can be learned non-trivially if and only if Λ cannot compute secure pseudorandom functions, i.e., nontrivial circuits can learn the class of Λ-concepts if and only if Λ-functions are too "weak" to convincingly simulate randomness. This is a striking and interesting result, because it characterizes the difficulty of learning by the expressive power of the target concept class.

One direction is well-established ([1], Lemma 5). If Λ-concepts can be learned, then Λ-functions cannot simulate randomness. A randomness distinguisher can be built using the learner as follows: The learner can be run on the

ISSN 1433-8092

candidate random object. Those output by Λ-functions can be learned successfully. However, most randomly chosen functions will have almost no correlation with the learner's output. (This follows a counting argument that the Boolean functions vastly outnumber Λ-functions, and so possible outputs of the learner).

It has recently been shown that if Λ functions cannot simulate randomness, then Λ concepts can be learned. The proof proceeds as follows: assume the existence of detectors that distinguish functions in Λ from random responses to their queries, thus preventing individual Λ-functions from successfully simulating randomness; then, we apply the minmax theorem, which guarantees a universal detector that can identify any Λ function's outputs; this universal detector is used in a blackbox generator to approximate Λ functions, thus learning them. However, the learning "algorithm" produced by this result is non-constructive, in the sense that it requires "advice" to operate. The natural question to ask is: can we get a normal algorithm? And under what assumptions? In this thesis, we investigate this question and provide an answer.

The main tool used in proving the dichotomy in the non-uniform case is two-player zero-sum game formulations. A version of the minmax theorem applied to such a game is central to the proof. The non-uniformity of the minmax theorem contributes to the non-uniformity of the dichotomy. This makes the entire proof non-uniform, i.e., under the assumption that there are no pseudorandom generators in Λ against non-uniform adversaries/detectors, the proof gives a separate algorithm to learn Λ concepts for each input size n, but not an algorithm that works for all n. Though the assumption appears harder than the uniform version, it is also non-constructive, i.e., the assumption only requires the existence of adversaries for any potential pseudorandom generator.

We will explore the use of a uniform version of the minmax theorem and show that this leads to a uniform version of this result. To constructively show the existence of a uniform learning algorithm for the class Λ, we need to strengthen our assumption, requiring a constructive way to obtain a detector/adversary against any pseudorandom generator that works for any input size.

1.1 Orgaization of the paper

Section 2 gives some definitions and discusses the setting in which these results are useful.
Section 3 gives a summary of the result stating that non-existence of PRFs implies learnability. This is simply an extraction of the relevant concepts and proofs from the paper [1]. We give the set up and proofs which will be modified. This is followed by an analysis of the use of non-uniformity.
Section 4 gives the results pertaining to the uniform minmax theorem given by Vadhan and Zheng. Here, we only state the main results, leaving the proofs to appendix, because we will only be using the results and not modifying the proofs involved.
Section 5 gives our contribution which shows how to use the uniform minmax theorem to get a result stating that non-existence of PRFs implies uniform learnability, under some assumptions.

2 Basics - Definitions, Notations and established results

This section covers definitions about circuit complexity, probabilistic and randomized algorithms, learning, psuedorandom function families, black box generators and minmax theorem. Here we only give definitions and theorems that are well-established and can be considered basic. Other definitions and theorems that are more advanced are presented in the following sections.

2.1 Circuit Complexity

Boolean circuit: A Boolean circuit is a node-labeled DAG (directed, acyclic graph) in which

- Source nodes are labeled by inputs
- Sink nodes are labeled by outputs
- Internal nodes are labeled by Boolean gates (we restrict this to be from $\{\neg, \wedge, \vee\}$).

An edge is also called a wire. Each of the input nodes can be negated as soon as it is processed. Due to De Morgan's Law, this is equivalent to not gates being allowed as nodes anywhere in the circuit.

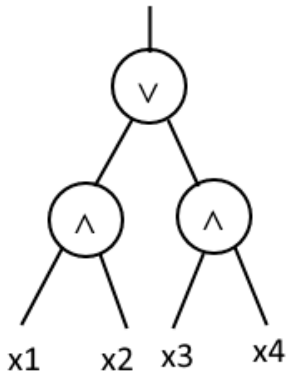

Figure 1: An example of a Boolean circuit

Fan-in / Fan-out: Fan-in of a node is the number of wires going into it; Fan-out of a node is the number of wires going out of it.
Note that the fan-in of a node should match the arity of the gate which is the label of the node. If the gates can be made general (e.g. $\wedge$, $\vee$), there is no restriction on the fan-in.

Size of the circuit: Size is the number of nodes in a circuit. Given that the number of edges is polynomial in the number of nodes, we may also specify the size of a circuit as the number of edges in it.

Depth of a circuit: Depth of a circuit is the longest path in the DAG.

Descriptions of circuits: Since a circuit is just a labeled DAG, it is possible to describe a circuit using a bit-string of length polynomial in its size. We will assume such a standard encoding.

Evaluation of a circuit:

- Input nodes' evaluation is given by the input corresponding to their label.
- Each edge's evaluation is given by its source node's evaluation.
- Internal nodes' evaluation is the output produced by the gate corresponding to their label, with inputs given by the edges coming into them.
- Output node's evaluation is taken as the output corresponding to the node's label.

Note that the evaluation of the node is propagated to all the wires coming out of it.

It is direct to see that, given the description of a circuit, it can be evaluated in time polynomial in it the size of the description, and so in time polynomial in the size of the circuit.

Model of computation: Given that a circuit is a DAG and its nodes can be topographically sorted, a circuit computes a Boolean function $f : \mathbb{B}^M \to \mathbb{B}^N$ where M is the number of inputs, and N is the number of outputs.

Unlike a program which can take inputs of any lengths, a single circuit computing a Boolean function $f : \mathbb{B}^M \to \mathbb{B}^N$ can only take M-bit string as input. There are 2^M such bit-strings which this circuit can handle, whereas a Turing machine can input any (finite) bit-string. Though this seems like a property of circuits, it is a limitation: there are Boolean functions such as conjunction, parity, majority etc. that can be naturally defined on any input size; extending this idea, we may want to consider any function $f : \mathbb{B} * \mathbb{B}$.

Circuit family: A family of circuit indexed by a set. Typically, a circuit family is indexed by natural numbers, and the i^{th} in the family takes i Boolean inputs and one output. So, a circuit family C is a function F from $\mathbb{N}$ to circuits. Thus, this circuit family simulates a Boolean function of any input size, and so is a model of computation similar to Turing machines.

Size/Depth: Size/depth of a circuit family is the function mapping n to the size/depth of the circuit taking n inputs. We can apply asymptotic resource measures here.

Non-uniform model of computation: As defined, a circuit family need not be computable, i,e, there need not exist a Turing machine that takes a natural number i and outputs the circuit with i inputs. Because of this, such a circuit family can compute undecidable functions.

Non-uniformity and advice: Even if no algorithm is known for a circuit family, it can still be proven to exist non-constructively. This can be turned

into a partial algorithm for the circuit class, which requires consulting an oracle for advice for the non-constructive part of the proof.

Uniform model of computation: A uniform circuit family is a circuit family that is constructible, i,e, there is a Turing machine M that inputs i in unary, and outputs a description of a circuit with i inputs.

Circuit classes: Circuit classes are sets of circuit families that satisfy some restrictions. There are many ways to restrict a circuit family to get complexity considerations in.

- We can restrict the size of a circuit family. $C_n[s(n)]$: the class of Boolean functions of n input variables computable by circuits of at most $s(n)$ wires. A typical restriction is that if polynomial or subexponential. $C[poly(n)]$: the class of Boolean functions computable by circuit families $C_1, C_2, ...$ where the size of all C_n is bounded by a polynomial of n.
  Note that given there are only 2^n inputs of size n, and only 2^{2^n} functions, if a circuit is allowed to be exponential, any circuit is possible.
- We can restrict the depth of a circuit family. A typical restriction is that of logarithmic depth.
- We can restrict the complexity of the Turing machine that generates the function family. Typical restrictions include polynomial time or logarithmic space.

2.1.1 Variants and extensions to circuits

Randomized circuits: These are circuits that take m random bits as inputs in addition to their regular inputs. For such circuits we talk of the probability of outputting "True" or 1 for a given input, when the random bits are drawn from some distribution, typically the uniform distribution.

In case of circuit families, we may require that the probability of acceptance should be a function of the input size, or that it should be bound away from 1/2 by a value that is a function of the input size. We may also require that the number of random bits used by a circuit is a function of the number of inputs of the circuit.

In the following, we use U_n to denote the uniform distribution over n-bit strings, and F_n to denote the uniform distribution over all Boolean function over n inputs. We write $x \sim D$ to denote that x is drawn from the distribution D. We write $x \sim A$ to denote that x is drawn uniformly at random from the *finite* set A.

Oracle circuit: circuit with "oracle gates" that can be substituted with any Boolean function h (with the same arity) as its oracle. We denote an oracle circuit by C^O, and the output of the circuit with input x and oracle h is $C^h(x)$. Note that this corresponds to the idea that an oracle can be queried in one time step.

If a circuit C has oracle access to a function h of m inputs, then for C to determine the function h, it has to query 2^m different inputs, thus it has to be of size $> 2^m$.

2.2 Computational Indistinguishability

A central concept of cryptography is "effective similarity". Under this notion, the inequality or dissimilarity between objects matter only if the difference is detectable. This naturally gives rise to the notion of a detector (distinguisher) and "cheater". Generalizing this, we have a class of distingushers and try to detect the 'identity' of another class of objects. Typically, the cheaters we are interested in are probability distributions (that try to pass themselves off as other probability distributions, e.g.,: as a uniformly random distribution); typically, the detectors we are interested in are statistical tests with limited resources.

Computational Indistinguishability: Let X and Y be probability distributions over strings of length n. We say that X and Y are computationally indistinguishable if for every feasible algorithm A, we have $|Pr[A(X) = 1] - Pr[A(Y) = 1]|$ is a negligible function of n (e.g. exponentially decreasing with n).

Distinguishers and distinguishing circuits: Given a probability distribution W_n over $\{0,1\}^n$, and a Boolean function $b : \{0,1\}^n \to \{0,1\}$, we say that h_n is a *distinguisher* for W_n if $|Pr_{w \sim W_n} h_n(w) - Pr_{x \sim U_n} h_n(x)| \geq 1/4$.
A circuit D_n is a distinguishing circuit for W_n if D_n computes h_n described above.

The definitions of distinguishers and distinguishing circuits can be generalized directly to families indexed by n. Here, if the distinguishing circuit family is computable, then we have uniform distinguishing circuits. Further, the resource requirements of generating the probability distribution W_n and the distinguishers can be limited separately, so it is possible to talk about one class of functions being distinguishable (or not) by another class.

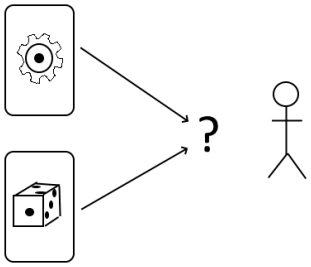

Figure 2: What a function distinguisher does

2.3 Pseudorandom generator

A true random string is a sequence generated by an unpredictable physical process, such as a quantum event, or tossing of a coin. Pseudorandom string is

a sequence generated by a program, and so, is predicable; if a program takes a true random string as seed to generate a much longer sequence, it would not be completely predictable; but it would not be completely unpredictable, as a true random string would be. Given these definitions are based on predictability, we could consider the strength of the predictors as a parameter, i.e., predictability can be defined with respect to a resource limited class of predictors. Then, we can use a pseudorandom string instead of a true random string, even in cryptographic settings.

Pseudorandom generator: Let $l : \mathbb{N} \to \mathbb{N}$ with $l(n) > n$. A pseudorandom generator with stretch function l is an efficient, deterministic algorithm that inputs a (true) random n bit seed and outputs an $l(n)$ bit string which is computationaly indistinguishable from a (true) random $l(n)$ bit string.

A psuedorandom function allows efficient, direct access to very long pseudorandom sequence (too large to read in full). Pseudorandom functions can replace truly random functions (a function chosen at random from a class of functions) in applications where the function is only called (and not subjected to a source-code analysis).

Pseudorandom function: A pseudorandom function is an efficient, deterministic algorithm which inputs s, an n bit seed, and an n bit argument x, and returns an n bit string, $f_s(x)$, so that it is infeasible to distinguish the response of f_S for a uniformly chosen s from a response of a truly random function.

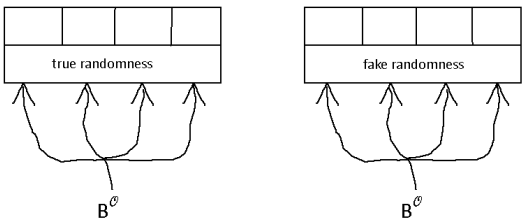

Figure 3: How a distinguisher samples from a pseudorandom function

Pseudorandom function family.
Function family: a pair (G_n, D_n)

- G_n: a set of Boolean functions of n variables
- D_n: a probability distribution over G_n

A $(t(n), \varepsilon(n))$-**pseudorandom function family** or **PRF** in $C_n[s]$ is a function family (G_n, D_n) in $C[s(n)]$ such that

$$\left|Pr_{h\sim D_n, w_1}[B^h(w_1) = 1] - Pr_{f\sim F_n, w_2}[B^f(w_2) = 1]\right| \leq \varepsilon$$

for every randomized oracle circuit in $B^O \in Circuit^O[t(n)]$.

As the circuits are parametrized by n, the definitions directly generalize to families, and if the circuit family is computable, then we have pseudorandom

function families; similarly we may also have pseudorandom function families secure against a uniform class. Further, the resource limits on pseudorandom generators and distinguishers can be set separately.

2.4 Learning

PAC learners: Probably approximately correct (PAC) learning is a type of function learning. It involves an instance class I and a hypothesis class H. An oracle algorithm A^O samples queries from a function $f \in I$ and returns a function $h \in H$ (as a circuit). The learner has two criteria based on parameters δ and ε:

- The learner algorithm has a $1 - \delta$ chance of running successfully, given a particular distribution over the instance class.
- If successful, the output h matches f on most inputs: $Pr_{x \sim D}[h(x) = f(x)] \geq 1 - \varepsilon$ where D is some distribution over inputs.

We are interested in studying the learnability of some circuit classes by oracles in other circuit classes.

Learning a circuit family: We say that a class $C[s(n)]$ of Boolean functions has (ε, δ)-learners running in time t if for large enough n: For every function $f \in C_n[s(n)]$, with probability at least $1 - \delta$ over the algorithm's randomness, the call $A^f(1^n)$ outputs a circuit h such that $Pr_{x \sim U_n}[h(x) \neq f(x)] < \varepsilon(n)$.

Learner circuit: We are interested in learning functions from a class $C[s(n)]$ using a randomized oracle algorithm with membership queries. If a function f is represented by a circuit C, the learner has oracle access to f and outputs a circuit description C' that is a circuit for f. Note that C need not be the same as C'.

Circuit learning a circuit: Here, the learner L is a circuit - so L is an oracle circuit that has oracle access to the function f and outputs a circuit description of C where C computes f. (The outputs of L can encode the description of circuits. We will assume a standard encoding of this sort).

2.5 Black-box generator

A black-box generator (for this project) is a pseudorandom generator specifically designed for the purposes of derandomization, as opposed to cryptography. It assumes the hardness of a function f (even to approximate), and guarantees that if the pseudorandomness generated is distinguishable from true randomness, then f can be approximated by simple algorithms, violating the hardness assumption.

Definition:
A (γ, ℓ)-black-box generator for a circuit class $C_n[s(n)]$ is a mapping that assigns to each function $f \in F_n$ a set of functions $\{g_z | z \in U_m\}$ where $g_z : \{0,1\}^\ell \to \{0,1\}$, for which the following conditions hold:

- size: the parameter m is polynomial in n and $1/\gamma$

- complexity: For every $z \in \{0,1\}^\ell$, we have $g_z \in C^f[poly(m)]$
- reconstruction: Let $L = 2^\ell$ and W_L be the distribution supported over $\{0,1\}^L$ that is generated by the truth table of g_z, where z is uniformly sampled. There is a randomized algorithm A^f, taking as input a circuit D and having oracle access to f, which when D is a distinguishing circuit for W_L, outputs a circuit that is γ-close to f and is of size polynomial in n, $1/\gamma$, and $size(D)$ with probability $\geq 1 - 1/n$.

2.6 Games and Minmax theorem

In a two-player game, player 1 and player 2 both have a *finite* number of **pure strategies**. Player 1 and player 2 each play pure strategies and the outcome or value of the game is a numeric value. Player 1 tries to minimize this value while player 2 tries to maximize it.

2.6.1 Game matrix

This idea can be described by a matrix M, where each row is a pure strategy by player 1 and each column is a pure strategy by player 2. Following the notation used in [1], $M(i,j)$ is the value of the game when player 1 plays pure strategy i and player 2 plays pure strategy j.

2.6.2 von Neumann's minmax theorem

An important result in game theory is von Neumann's **minmax theorem**:

Theorem 1 (minmax theorem): Let P denote a probability distribution over matrix rows and Q denote a probability distribution over matrix columns. Then

$$\min_P \max_Q \mathbb{E}_{i \in P, j \in Q} M(i,j) = \max_Q \min_P \mathbb{E}_{i \in P, j \in Q} M(i,j)$$

The distributions P and Q are called *mixed strategies*, and we use the notation $M(P,Q) = \mathbb{E}_{i \in P, j \in Q} M(i,j)$.

Because the optimal response to any given mixed strategy is a pure strategy (due to the convexity of the space of mixed strategies), this can be rewritten:

$$\min_P \max_j \mathbb{E}_{i \in P} M(i,j) = \max_Q \min_i \mathbb{E}_{j \in Q} M(i,j)$$

Definition: Value of a game: The value $v(M)$ of a game with matrix M is the value on both sides of the minmax equation.

Definition: Value of a strategy: The value of a mixed strategy P for player 1 is $v(P) = \max_Q[M(P,Q)]$. The value of a mixed strategy Q for player 2 is $v(Q) = \min_P[M(P,Q)]$.

Interpretation: The minmax theorem gives us information about how well a player's mixed strategies perform.

The minmax theorem can be written as:
For any ε,

$$LHS = \min_P \max_Q \mathbb{E}_{i \in P, j \in Q} M(i,j) = \min_P v(P) > \varepsilon$$

if and only if

$$RHS = \max_Q \min_P \mathbb{E}_{i \in P, j \in Q} M(i,j) = \max_Q v(Q) > \varepsilon$$

$LHS > \varepsilon$ can be interpreted as: for all mixed strategies P of player 1 there exists a response j from player 2 such that $M(P,j) > \varepsilon$.
$RHS > \varepsilon$ can be interpreted as: there exists a mixed strategy Q for player 2 such that for all i, $M(i,Q) > \varepsilon$
In other words, if player 2 has a response to each of player 1's mixed strategies, then player 2 has a single mixed strategy that can responds to any of player 1's strategies.

The minmax theorem is non-uniform. This means that there is no known efficient algorithm for finding the maxmin strategy given the responses to player 1's strategies. In fact, there may be no efficient way to describe the distribution at all.

3 Non existence of PRF implies learnability: non-uniform case

This section examines the proof of learnability in the absence of pseudorandom functions in the non-uniform setting, with the aim of isolating the sources of non-uniformity.

The definitions and proof given here are extracted from [1]. We first give all the relevant definitions and theorem statements. These theorems may or may not be used in our proof, but we will not modify them or their proofs. We split the proof into two subsections - first showing the existence of a universal distinguisher, then using it to show the existence of a learner. We give full proofs in these subsections because we intend to modify these.

3.1 Game matrix of PRF

The result in [1] is that nonexistence of sampleable PRFs implies learnability for $C_n[n^k]$ for all k (and a particular value of n). This result is non-uniform because the proof involves the minmax theorem.

3.1.1 Set up

We will create a game, the **PRF distinguisher game**, that involves distinguishing function families from random functions.

- Player 1's pure strategies: functions $h \in C_n[s]$
- Player 2's pure strategy: circuits $C \in Circuit^O[t]$
- Game matrix values: $M(h, C^O) = C^h - \mathbb{E}_{f \sim F_n}(C^f)$.

We also require that $Circuit^O[t]$ must be closed under complementation, for reasons that will become clear soon.

In this setup, the minmax theorem states: $\min_P \max_{C^O \in Circuit^O(t)} \mathbb{E}_{h \sim P} M(h, C) \geq \varepsilon$ if and only if $\max_{Q \sim Circuit^O(t)} \min_h \mathbb{E}_{C \sim Q} M(h, C) \geq \varepsilon$.

The value of the game with mixed strategies P and Q is given by:

$$\begin{aligned} &\mathbb{E}_{h \sim P, C \sim Q} M(h, C) \\ &= \mathbb{E}_{h \sim P, C \sim Q}[C^h - \mathbb{E}_{f \sim F_n}(C^f)] \\ &= \mathbb{E}_{h \sim P, C \sim Q}[C^h] - \mathbb{E}_{f \sim F_n, C \sim Q}[C^f] \\ &= Pr_{h \sim P, C \sim Q}[C^h = 1] - Pr_{f \sim F_n, C \sim Q}[C^f = 1] \end{aligned}$$

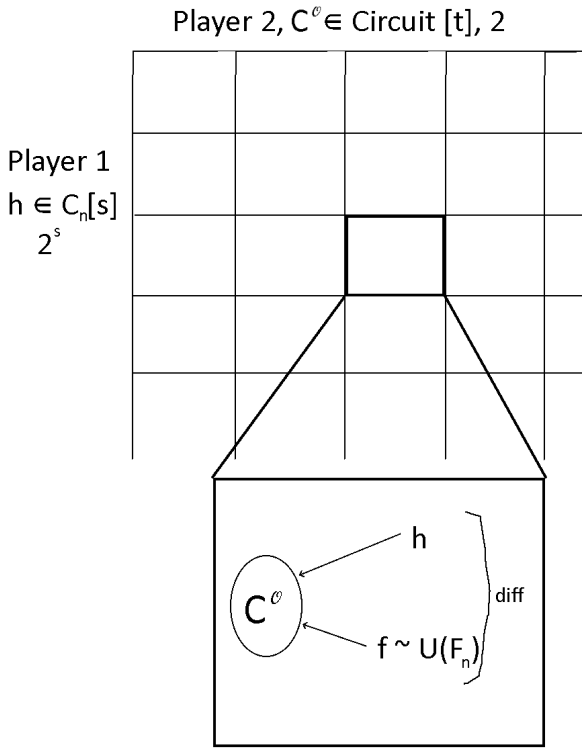

Figure 4: Setup of the PRF distinguisher game.

3.1.2 Non-existence of PRF - interpretation

The LHS of the minmax theorem says:
For all distributions P over $C_n[s]$, there exists a circuit $C \in Circuit^O[t]$ such that $Pr_{h\in P}[C^h = 1] - Pr_{f\in F_n}[C^f = 1] \geq \varepsilon$.
This is *almost* the same as the statement "There are no (ε, t)-PRFs in $C_n[s]$." The only difference is that it is missing the absolute value sign. But since we assumed that the class $Circuit^O[t]$ is closed under complementation, this statement follows from the nonexistence of PRFs. This is because for any circuit C that distinguishes $C_n[s]$, either C or its complement C' satisfies $Pr_{h\in P}[C^h = 1] - Pr_{f\in F_n}[C^f = 1] \geq \varepsilon$.

3.1.3 Universal ensemble distinguisher - interpretation

The RHS of the minmax theorem says:
There exists a distribution Q over $Circuit^O(t)$ such that for every function $h \in C_n[s]$, we have $Pr_{C\in Q}[C^h = 1] - Pr_{C\in Q, f\in F_n}[C^f = 1] \geq \varepsilon$.

This distribution Q acts like a distinguisher for every circuit in $C_n[s]$, in

terms of its expected behavior. It is possible to make this into a single distinguishing circuit for the class, by combining all the circuits in Q into a single randomized circuit. However, the number of circuits in $Circuit^O[t]$ is exponential in t, and thus this distinguisher may be larger than exponential sized in n. The proof of efficient learnability requires an reasonable sized universal distinguisher, which Oliveira and Santhanam achieve using the small-support minmax theorem.

3.2 Non-uniform small support minmax theorem

The small-support minmax theorem [2] [3] was introduced by Richard Lipton and Neal Young. It guarantees the existence of small distributions that are almost as good as the optimal row and column strategies.

Definition: k-uniform strategies: We say that a mixed strategy is **k-uniform** if it is a uniform distribution over a multiset of at most k columns or rows. We use P_k and Q_k to denote the sets of all k-uniform row and column strategies, respectively.

Theorem 2 - Small support minmax theorem: Let M be a $r \times c$ real-valued matrix with entries in the interval $[-1, 1]$. For every $\delta > 0$, if $k_r \geq 10\ln(c)/\delta^2$ and $k_c \geq 10\ln(r)/\delta^2$ then

$$\min_{P \in P_{k_r}} v(P) \leq v(M) + \delta$$

and

$$\max_{Q \in Q_{k_r}} v(Q) \geq v(M) - \delta$$

This theorem allows for an ensemble of circuits that distinguish $Circuit^O[t]$ and can be converted into a single reasonable sized random circuit.

3.3 Non-existence of PRF implies universal distinguisher

Theorem 3 (Oliveira and Santhanam): Let $s(n) \geq n$, $t(n) \geq n$, $\varepsilon(n) > 0$,and $\gamma(n) > 0$ be arbitrary functions. If the circuit class $C_n[s(n)]$ contains no $(t(n), \varepsilon(n))$-PRF, then there is a randomized oracle circuit $B^O \in Circuit^O[O(ts/\gamma)^c]$ (for some universal constant $c \in \mathbb{N}$) that distinguishes every distribution over $C_n[s(n)]$ from randomness with advantage at least $\epsilon(n) - \gamma(n)$.

Because the minmax theorem is non-uniform (there is no efficient algorithm to compute the max-min strategy), the universal distinguisher is also non-uniform.

3.4 Universal distinguishers imply learners

Main lemma for this subsection: **Lemma**: Assume that for every $k > 1$ and large enough n there is a randomized oracle circuit B_n^O in $Circuit^O[2^{O(n)}]$ which

distinguishes every distribution D_n over $C_n[n^k]$ from randomness with advantage 1/40. Then for every $\ell > 1$ and $\varepsilon > 0$, there is a non-uniform sequence of randomized oracle circuits in $Circuit^O[2^{n^\varepsilon}]$ that learn every function $f \in C_n[n^\ell]$ to error at most $n^{-\ell}$.

The proof of the above theorem is basically from [1]. In this subsection, we give the background needed and the proof itself.

This definition, in some sense, allows us to connect distinguishers for functions to learners for similar functions.

Lemma: The class AC^0 can be learned in quasi-polynomial time.

Learners for the circuit classes $C_n[n^k]$ are constructed from distinguishers for $C_n[n^k]$ via a black-box generator, as described above.

This next theorem guarantees that black-box generators exist for a large class of circuit families.

Theorem 4 - existence of black-box generators. Let p be a fixed prime, and C be a typical circuit class containing $AC^0[p]$. For every $\gamma : \mathbb{N} \to [0, 1]$ and $\ell : \mathbb{N} \to \mathbb{N}$, there exists a (γ, ℓ)-black-box generator within C. Furthermore (although unstated in Oliveira and Santhanam), the reconstruction algorithm A^f can be considered a function of n and will still run in polynomial time when taking input 1^n. The function mapping z, x to $g_z(x)$ (with oracle access to f) can also be computed in polynomial time.

This allows us to prove learners for a circuit class whenever we can find a distinguisher D for the black-box distribution W_L.

Lemma - faster learners from distinguishers. Let C be a typical circuit class. If $C[poly(n)]$ has distinguishers running in time $2^{O(n)}$, then for every $\varepsilon > 0$, $C[poly(n)]$ has strong learners running in time $O(2^{n^\varepsilon})$.

Requirements: There are *separate distinguishing algorithms* for $C_n[f(n)]$ for each polynomial $f(n)$ and each n.
Guarantees: There are *separate learning algorithms* for $C_n[f(n)]$ for each polynomial $f(n)$ and each n.

Proof: Let A_0 be a complexity distinguisher for $C_n[s]$ (or alternatively a distinguisher that works for any distribution whose support is a subset of $C_n[s]$), taking input 1^ℓ. If C is the class AC^0, then it can be learned due to the above lemma. Otherwise, C contains $AC^O[p]$ for some prime p.
We show the existence of $(1/n^k, 1/n)$ learners.
Let $0 < \varepsilon' < \varepsilon$. Theorem 4 guarantees a $(1/n^k, n^{\varepsilon'})$ black-box generator A_1 of functions $g_z \in C^f_\ell[poly(m)]$, where $m = O(poly(n, 1/\gamma)) = O(poly(n))$.
The learner A^f is constructed as follows:
Interpret the oracle algorithm A^g_0 as a probabilistic polynomial time algorithm explicitly given the truth table of g as input.
This produces an algorithm $D(\cdot, \overrightarrow{r})$, where r is a vector of random bits. It is converted into a circuit with these random bits fixed. Because we started with an algorithm in time $O(2^n)$, and it took an input of size ℓ, it converts into a circuit D_L of size $O(2^\ell)$.
We then run A_1 on input D_L, and A^f returns with the same output.
Because of the complexity restriction of f, we have $g_z \in C_\ell[poly(\ell)]$. Thus, D_L

works as a distinguisher of W_L and running $A_1(D_L)$ outputs a learner.
We can also prove the same thing assuming A_0 is a distinguisher for the uniform distribution over truth tables of circuits in any subset (or multiset) of $C_n[s]$, rather than a complexity distinguisher. This proof is more direct; we simply fix the subset of $C_n[s]$ to be $\{g_z | z \in \{0,1\}^m\}$ and the distribution to be W_L, and the result follows directly because we have a distinguisher for W_L.

To prove that no PRFs in $C_n[n^\ell]$ implies learnability, we use Theorem 3 to construct a universal distinguisher for $C_n[n^\ell]$ and Lemma 33 to show that $C_n[n^\ell]$ (for each values of n, ℓ) has a learner. Because the distinguisher is non-uniform, this will result in a non-uniform learner.

4 Uniform Minmax theorem with application to hardcore lemma

Vadhan and Zheng (2014) present a uniform version of the minmax theorem. This theorem constructs an algorithm to compute an approximate minmax strategy given oracle access to the optimal response to a pure strategy of player 1. [4]

Definition. Let N be a natural number. We denote the set $\{1, 2, \ldots, N\}$ as $[N]$.

In this game setup, $[N]$ is the set of options for player 1 and $\mathcal{W}$ is the set of options for player 2. Unlike in the PRF distinguisher game, we allow player 1's *pure strategies* to be any particular distribution P over rows, rather than individual rows. We require that the are a finite number of pure strategies, and denote the set of all pure strategies as $\mathcal{V}$. A mixed strategy is then a linear combination of these pure strategies; the space of mixed strategies may not cover all distributions over rows. Player 2's pure strategies are still individual columns.

Unlike in the PRF distinguisher game, the payoff of the game $M(x, y)$ is a real number in the range $[0, 1]$, not $[-1, 1]$. Player 1 tries to minimize the payoff while player 2 maximizes it.

The uniform minmax theorem goes as follows:

Theorem 5 - Uniform Minmax Theorem. Let there be a 2-player game set up as above. For every $0 < \varepsilon \leq 1$ and every S, there is an algorithm to output a mixed strategy Q^* for Player 2 such that for all Player 1 pure strategies P:

$$M(P, Q^*) \geq \mathbb{E}_{1 \leq i \leq S} M(P^{(i)}, Q^{(i)}) - O(\varepsilon)$$

for a particular sequence of mixed strategies $P^{(1)}, P^{(2)}, ..., P^{(S)}$ and their response strategies $Q^{(1)}, Q^{(2)}, ..., Q^{(S)}$.
In particular, if we have an algorithm to compute a strategy $Q^{(i)}$ from a representation of $P^{(i)}$ such that $M(P^{(i)}, Q^{(i)}) \geq v(P) - \delta$, the resulting strategy Q^* is such that for all Player 1 pure strategies P:

$$M(P, Q^*) \geq v(M) - \delta - O(\varepsilon)$$

We need some criteria regarding efficient computability in order for this theorem to hold:

- A *compact* representation of the mixed strategies $P^{(i)}$ ($O(Poly(\log(n)))$ space).
- A fast algorithm to obtain a response $Q^{(i)}$ for each mixed strategy $P^{(i)}$.
- A fast algorithm to perform weight update, and to project onto $Conv(\mathcal{V})$.
- A choice of pure strategies so that $U_{[N]} \in Conv(\mathcal{V})$

An application of this theorem is the uniform hardcore theorem. This is a uniform version of Russell Impagliazzo's Hardcore Theorem. which goes as follows:

Theorem 6 - Uniform hardcore theorem: Let L be a parameter indexing a family of distributions, $m = m(L)$ be a polynomial function of L, $\delta = \delta(L)$, $\varepsilon' = \varepsilon'(L)$ computable in $poly(L)$ time, and $(X, B) = G(U_m)$ be a joint distribution where $G : \{0,1\}^m \to \{0,1\}^L \times \{0,1\}$ is computable in polynomial time.

- U_m is the uniform distribution over $\{0,1\}^m$.

Assume that for every $C \in \mathcal{C}_{m,2\delta}$ and infinitely many L, we have $Pr_{(x,b)\sim G(C)}[A^C(x) = b] > 1/2 + \varepsilon'$, where A is an oracle algorithm that runs in time t.

- $\mathcal{C}_{m,2\delta}$ is the set of all 2δ-dense probability distributions over $\{0,1\}^m$.
- This set contains the uniform distribution U_m.

Then there is a $poly(t, L, 1/\delta, 1/\varepsilon')$ algorithm P such that for sufficiently large L, $Pr_{(x,b)\sim G(U_m)}[P(x) = b] > 1 - \delta$.

Note that this theorem is uniform because it assumes a single oracle algorithm and guarantees a single predictor working for every value of L.

5 Non existence of PRF implies learnability: uniform case

In this section, we show that there exist uniform learners, assuming there is a constructive way to find distinguishers for every function family in $C_n[poly(n)]$. We use the uniform hardcore lemma to produce uniform predictors which work as distinguishers, and use the black-box generator to prove the existence of a uniform learner.

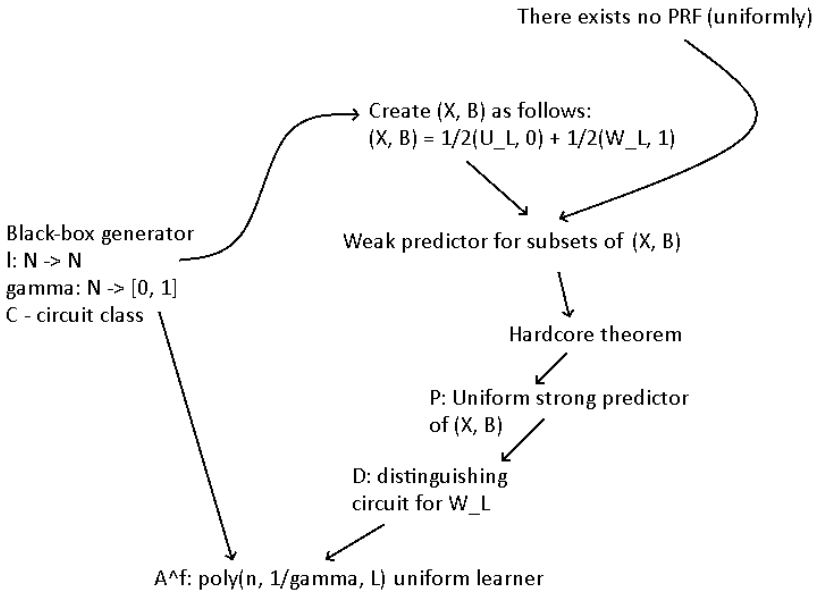

Figure 5: Outline of the proof of uniform learners

5.1 Definitions

tt_f - The truth table of the Boolean function f. If f has n inputs, tt_f is a binary string of length 2^n where the i-th bit is the result of f evaluated on the i-th configuration of inputs.

5.2 Nonexistence of PRF implies uniform learners

5.2.1 General set up and black-box distribution

We set up this theorem so that the predictor guaranteed by the hardcore theorem is a distinguisher for the black-box distribution W_L. This means that we have to set up the distribution (X, B) to be related to W_L.

Then, the existence of a black-box generator and a distinguisher directly implies the existence of a learner. Because the constructions of both the black-box generator and distinguisher are uniform, the learner is also uniform.

5.2.2 Hardcore Lemma set up

In the Hardcore Lemma, we need:

- A constant m: we take the uniform distribution U_m over binary strings of length m
- A set S_X containing the values of X in the pair (X, B)
- A generator function G: It maps each of the strings in U_m to a distribution (X, B).
- A value δ that is the accuracy of the predictor. We need a distinguisher for 2δ-dense distributions.

We satisfy these requirements as follows:

- m is the parameter from the black box generator.
- S_X: we take $S_X = \{0,1\}^{2^\ell}$ – the set of binary strings of length 2^ℓ.
- G: We set up G as follows: $G(x) = 1/2(< U_{2^\ell}, 0 > + < tt_{g_x}, 1 >)$, where g_x is the function from the black-box generator. In other words, a value of X has a $1/2$ chance of being chosen from U_{2^m} (in which case the value from B is 0) and a $1/2$ chance of being tt_{g_x} (in which case the value from B is 1).
- We let $\delta = 3/16$.

5.2.3 No PRF implies weak predictability of dense distributions in BB distribution

Theorem 7 - No PRF implies weak predictability: Let $k_W(n)$ be computable in $poly(n)$ time. Assume that for every distribution D over F_n with $Support(D) \subseteq C_n[s]$, there is a circuit $B^O \in Circuit^O[t(n)]$ such that $|Pr_{f\sim F_n}[B^f = 1] - Pr_{h\sim D}[B^h = 1]| \geq \varepsilon$. Furthermore, assume there is a $t_W(n)$-time algorithm $W(1^n)$ that with probability $1 - \varepsilon_W$ constructs B^O given $k_W(n)$ random samples from D. It follows that there is an oracle algorithm $A^O(1^\ell, x)$ that runs in time t, sampling from a distribution C, such that:

For every integer n and every distribution C over $U_{m(n)}$, $Pr_{(x,b)\sim G(C)}[A^C(1^{\ell(n)}, x) = b] > 1/2 + \varepsilon'$ for $\varepsilon' = \varepsilon/2$.

Proof:
Construct an algorithm $or^C(1^\ell)$ that samples up to $2k_W(\ell)$ samples (x, b) from a distribution $G(C)$, until $b = 1$. In this case, x is the truth table of a function sampled from the distribution $W_L \subset C_\ell[\ell^k]$. This algorithm has a $1 - 1/2^{2k_W(\ell)}$ chance of success on each run. This algorithm runs in $2k_W(\ell) + poly(\ell)$ time.
Construct the algorithm A as follows:
Input the strings 1^ℓ and x with oracle access to $C \subseteq U_m$.
Use or^C to answer the $k_W(\ell)$ queries used by $W(1^\ell)$ to return an oracle circuit B^O. The chance of all the queries being answered successfully is $(1 - 1/2^{2k_W(\ell)})^{k_W(\ell)} > 1 - 1/2^{k_W(\ell)}$.
Because *or* queries from a distribution over $C_\ell[\ell^k]$, the oracle circuit B^O acts as a distinguisher for this distribution.
Convert B^O into a circuit B' whose input is of size 2^ℓ, such that B^f has the same output as $B'(tt_f)$.
Finally, run $B'(x)$ and output its return value.

Now we argue that A is a weak predictor. Let $\alpha = Pr_{f\sim F_n}[B^f = 1]$. We have $Pr_{f\sim F_n}(A^C(tt_f) = 0) = 1 - \alpha$ and $Pr_{h\sim D}(A^C(tt_h) = 1) > \alpha + \varepsilon$. Since $G(C)$ is of the form $1/2(< F_n, 0 > + < D, 1 >)$, $Pr_{<x,b>\sim G(C)}[A^C(x) = b] > (1 - \alpha)/2 + (\alpha + \varepsilon)/2 > 1/2 + \varepsilon'$ with $\varepsilon' = \varepsilon/2$.

The total run time is $O(t_W(\ell) + k_W(\ell)^2 + t(\ell) + 2^\ell)$.

5.2.4 Strong predictability of BB distribuion implies distinguisher for BB distribution

Theorem 8 - Conclusion of HC implies dist for all W_L: If there is an algorithm P such that $Pr_{(x,b)\sim G(U_m)}[P(x) = b] > 13/16$, then P is also a distinguisher for the distribution $W_L = \{tt_{g_z} | z \sim \{0,1\}^m\}$.

Proof: Let $\delta = 3/16$. We have $Pr_{(x,b)\sim G(U_m)}[P(x) = b] > 1 - \delta$. Because $G(U_m) = 1/2 < U_L, 0 > +1/2 < W_L, 1 >$, we have $Pr_{x\sim W_L}[P(x) = 1] > 1 - 2\delta$.
Now we calculate $Pr_{x\sim U_L}[P(x) = 1]$.
For a similar reason as above, $Pr_{x\sim U_L}[P(x) = 0] > 1 - 2\delta$, so $Pr_{x\sim U_L}[P(x) = 1] < 2\delta$.
Thus, $|Pr_{x\sim W_L}[P(x) = 1] - Pr_{x\sim U_L}[P(x) = 1]| > 1 - 4\delta = 1/4$.
From the definition of distinguisher, P is a distinguisher for W_L (when input 1^ℓ is fixed).

5.2.5 Proof of uniform learners

Theorem 9 - No PRF implies uniform learners: Let $\gamma : \mathbb{N} \to [0, 1]$ and $\ell : \mathbb{N} \to \mathbb{N}$ where $\ell(n)$ is computable in polynomial time and $n = poly(\ell)$. If there are no (ε, t)-PRFs in a circuit class $C_n[n^k]$ and there is an algorithm W that constructs a distinguishing circuit B^O for a distribution over $C_n[n^k]$ by sampling from the distribution, then the class $C_n[n^k]$ has $poly(n, 1/\gamma, 2^\ell)$-time learners A^O that output circuits of size $t \cdot poly(n)$.

Proof:
Let f be the input to the learner algorithm A.
A takes input 1^n and a function f from F_n as oracle.
Theorem 4 says that there is a $poly(n, 1/\gamma, L(n))$-time algorithm A_0^O that takes in f and returns a (γ, ℓ)-black box generator in $C_n^f[n^k]$ (the oracle version of the class $C_n[n^k]$).
Theorem 7, combined with the fact that there is an oracle algorithm W^O to generate distinguishers for all function distributions, proves that there exists a t-time algorithm $A_2^O(1^n)$, that samples from a distribution C over $G(U_m)$ and is a weak predictor for all 3/8-dense distributions.
Because $A_2^O(1^n)$ is a weak predictor for all 3/8-dense distributions, the Uniform Hardcore Theorem (using $\delta = 3/16$) provides an $poly(t, 2^\ell, 1/\varepsilon)$-time algorithm P such that for sufficiently large n:

$$Pr_{(x,b)\sim G(U_m)}[P(x) = b] > 13/16$$

We convert P, on ℓ-bit inputs, into a circuit D_P which is a strong predictor for the distribution $G(U_m)$.
We run $A_0^f(1^n)$ to output a circuit Q and run $Q^f(D_P)$ to output a circuit h. Finally, A returns the circuit h.

Now we argue that the algorithm works. By Theorem 8, the strong predictor P is also a distinguisher for W_L:

$$Pr_{x\sim W_L}[P(1^\ell, x) = 1] - Pr_{x\sim U_L}[P(1^\ell, x) = 1] > 1 - 4\delta = 1/4$$

Thus, D_P is a distinguisher for W_L for the particular value $\ell = \ell(n)$.
By the definition of black-box generator (which is constructive) inputting the distinguisher P^O into the reconstruction circuit Q, with oracle access to f, returns a function h such that:

$$Pr_{x\sim U_n}[h(x) = f(x)] \geq 1 - 1/\gamma$$

Thus, our algorithm A is a learner for $C_n[n^k]$.

6 Appendix

6.1 Proof of uniform minmax theorem

Consider any $P \in \mathcal{V}$ (set of pure strategies for the row player) such that $KL(P||P^{(1)}) \leq S \cdot \varepsilon^2$.
It is shown in Lemma A.1 of [4] that:

$$KL(P||P^{(i)}) - KL(P||P^{(i)\prime}) \geq (\log e)\varepsilon(\mathbb{E}[M(P^{(i)}, Q^{(i)})] - \mathbb{E}[M(P, Q^{(i)})] - \varepsilon)$$

Since $P^{(i+1)}$ is an ε^2-approximate KL projection of $P^{(i)\prime}$, it follows that:

$$KL(P||P^{(i)}) - KL(P||P^{(i+1)}) \geq (\log e)\varepsilon(\mathbb{E}[M(P^{(i)}, Q^{(i)}] - \mathbb{E}[M(P, Q^{(i)})] - \varepsilon) - \varepsilon^2$$

Summing for i from 1 to S, we get:

$$KL(P||P^{(1)}) - KL(P||P^{(S+1)}) \geq (\log e)S\varepsilon(\mathbb{E}_{1 \leq i \leq S}[M(P^{(i)}, Q^{(i)}] - \mathbb{E}[M(P, Q^*)] - \varepsilon) - S\varepsilon^2$$

Using our bound on $KL(P, P^{(1)})$, we get:

$$\mathbb{E}_{1 \leq i \leq S}[M(P^{(i)}, Q^{(i)})] - \mathbb{E}[M(P, Q^*)] \leq \frac{KL(P||P^{(1)}) + S\varepsilon^2}{(\log e)S\varepsilon} + \varepsilon = O(\varepsilon)$$

6.2 Proof of uniform hardcore theorem

The setup for the minmax theorem is as follows:

- $\mathcal{V} = C_{m,2\delta}$ - mixed strategies for player 1
- $\mathcal{W} = \{\text{ckts of size } tm + poly(t)\}$ - responses for player 2

$M(z, W) = 1$ if $W(x) = b$, else 0, where $G(z) = (x, b)$.

Vadhan and Zheng show that the uniform minmax algorithm can be implemented efficiently in this setting, using $\varepsilon = \varepsilon'/c$ for a sufficiently large constant c, $\gamma = \varepsilon/2S$, and $S = (\log(1/\delta) - 1)/\varepsilon^2$.

From the weak predictor A, we use a randomized algorithm to produce a circuit W of size $tm + poly(t)$ that approximates A with probability $1 - \gamma$ such that $Pr[W(x) = b] > 1/2 + \varepsilon' - 4\varepsilon$. We represent the strategy $P^{(i)}$ as a circuit $M^{(i)}$ of size t_i that computes a measure for $P^{(i)}$, where the value $M^{(i)}(z)$ has bit length $O(i \cdot \log(1/\varepsilon))$.

For each weight update, we compute a circuit $M^{(i)\prime}$. The formula is $M^{(i)\prime}(z) = \exp(-\varepsilon \cdot I(W^{(i)}(x) = b)) \cdot M^{(i)}(z)$, where $(x, b) = G(z)$ and I is the indicator function. The exponential value has bit length $\log(1/\varepsilon)$ and multiplication can be done in time $poly(i \cdot \log(1/\varepsilon))$. Using our sizes for W and G, we get that $M^{(i)\prime}$ has size $t'_i = t_i + tm + poly(t) + i \cdot polylog(1/\varepsilon)$.

We also need a fast algorithm to do KL projection onto $Conv(\mathcal{V})$.

- As described in Lemma A.3 of [4], KL projection algorithm can be done in time $poly(n, \log(1/\delta), 1/\varepsilon, \log(1/\gamma))$.

- Oracle access to the measure $M^{(i)\prime}$.
- Distribution corresponding to $M^{(i)\prime}$ must be in the neighborhood $\mathcal{C}^{\varepsilon}$ of $\mathcal{C}$.
- The result is $M^{(i+1)}$, an ε^2-approximate projection of $M^{(i)\prime}$ on $\mathcal{C}$. The algorithm works with probability $1-\gamma$ and the circuit $M^{(i+1)}$ has size $t_{i+1} = t'_i + polylog(1/\varepsilon)$. Bit length is $O((i+1)\cdot\log(1/\varepsilon))$.

Because the criteria for the uniform minmax theorem are satisfied, we output a universal predicting distribution A^*. Lemma 3.4 in [4] allows us to do this by approximating A^* with a uniform distribution and gives a constructive way to find a circuit P such that $Pr[P(X) = B] > 1 - (1-\varepsilon)\delta$.
Our predictor relies on an unknown value ϕ. We guess the value ϕ by setting $\lambda = 1/S, 2/S, ..., 1/2$ and computing $E_\lambda = Pr[P_{\phi=\lambda}(X) = B]$ for each λ.
By a Chernoff bound, it follows that $Pr[P(X) = B] = 1 - \delta$.

References

[1] Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds and pseudorandomness, 2016.

[2] Ingo Althöfer. On sparse approximations to randomized strategies and convex combinations, 1994.

[3] Richard J. Lipton and Neal E. Young. Simple strategies for large zero-sum games with applications to complexity theory, 1994.

[4] Jia Zheng. A uniform min-max theorem and characterizations of computational randomness., 2014.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il