# A uniform dichotomy between pseudorandom functions and learnability

Eric Binnendyk

New Mexico Institute of Mining and Technology

*eric.binnendyk@student.nmt.edu*

April 15, 2021

# Overview

# Existing result

In the *non-uniform setting*, there is a dichotomy between learnability and the existence of pseudorandom functions.
i.e.,
either a class of functions is too weak, and so can be learned
or the class is strong enough to contain pseudorandom generators.

# OUR result

In the*uniform setting*, there is a dichotomy between learnability and the existence of pseudorandom functions.
i.e.,
either a class of functions is too weak, and so can be learned
or the class is strong enough to contain pseudorandom generators.

# Existing proof

- Assumption: Class A of circuits is to weak to have pseudorandom generators.
  For every pseudorandom generator, there is a detector that can find out that it is not truly random.

- Conclusion: For every input size, the circuits in class A has a learner.

# Existing proof

- Assumption: Class A of circuits is to weak to have pseudorandom generators.
  For every pseudorandom generator, there is a detector that can find out that it is not truly random. (we don't know how to find the detector

- Conclusion: For every input size, the circuits in class A has a learner.
  we don't have a universal learner

# Existing proof

- Assumption: Class A of circuits is to weak to have pseudorandom generators.
  For every pseudorandom generator, there is a detector that can find out that it is not truly random. (we don't know how to find the detector

  Non-uniform min-max theorem

- Conclusion: For every input size, the circuits in class A has a learner.
  we don't have a universal learner

# OUR proof

- Assumption: Class A of circuits is to weak to have pseudorandom generators.
  There is a universal detector - (for any pseudorandom generator), that can find out that it is not truly random.

  uniform min-max theorem

- Conclusion: we have a universal learner

# Learnability

A foundational machine learning theory question is which concepts can be learnt and by which hypotheses and which learners.

- Concept class: This limits the allowed complexity of concept. E.g., convex shapes, linearly separable classes
- hypothesis class: This limits the allowed complexity of hypothesis learned. E.g.,: Perceptron, quadratic decision boundary etc.
- Learner
    - Queries randomly chosen examples of a concept, knowing what class the concept belongs to
    - Outputs a machine that can predict whether new things are examples of that concept

  e.g., perceptron learning algorithm

Many applications: speech recognition, predictive data analytics, etc.

# PAC Learnability

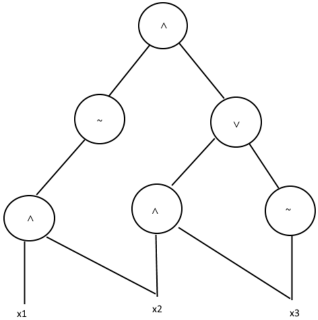
- Probability: the learning algorithm can fail on *a few* concepts in the concept class
- Approximate: successful learning may have some error
- resource bounds: computational complexity of the learner

# Learnability of a class $A$ by a class $B$

The question of learnability is:
can the concept class $A$ be learned to a hypothesis class $B$ by some PAC learner?
A **learner** for a family $C_n[s]$ of Boolean functions is an algorithm that takes a function $f \in C_n[s]$ and outputs a function $h$ that approximates $f$.
If $C_n[s]$ has a learner, we say that it is learnable.

# Circuit

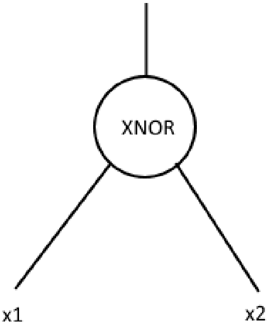
- **Circuit**: A model of computation which is a DAG (directed acyclic graph) with nodes for each operation (and, or) as well as nodes for inputs and outputs.
- size: number of nodes.
- has a description with size polynomial in the size
- It simulates a Boolean function of a **fixed** input size.

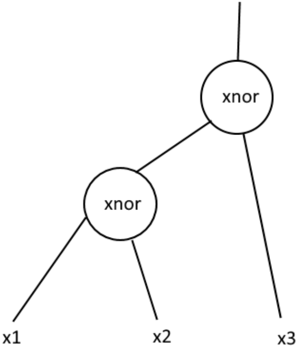


# The circuit model of computation

- A model of computation should be able to take inputs of any size, similar to a program.
- A family of circuits indexed by input size.
- function $f : \mathbb{N} \to ckt$ so that $f(n)$ is a circuit with $n$-bit input.

Complexity: function from input size to size of circuit -“simple function”



# Non-uniformity

function $f$ taking $n$ and producing a ckt with $n$ inputs.

- Non-uniform: $f$ is not computable.
- Circuits with different input sizes are doing different things.
- Eg: And(a,b); OR(a,b,c), xor(a,b,c,d), majority(a,b,c,d,e)
- can solve non-computable problems.

- Can still have complexity resource measures in terms of size of circuits, e.g., $n^{th}$ circuit should have size $\leq poly(n)$.

# Uniformity

function $f$ taking $n$ and producing a ckt with $n$ inputs.

- Uniform: $f$ is computable.
  There is a computable relationship between circuits of different sized.

- resource limitations on $f$: e.g., $f$ should be computable by a polytime algorithm.
- Can still have complexity resource measures in terms of size of circuits, e.g., $n^{th}$ circuit should have size $\leq poly(n)$.

# Oracle circuits

**Oracle circuit**: circuit with "oracle gates" that can be substituted with any Boolean function $h$ (with the correct number of input wires)
Oracle gate is like a black box.

# Pseudorandomness

Pseudorandomness is the property that a Boolean function cannot be "recognized" as distinct from a random function. In this case, that means that a set of "simple" functions cannot tell the difference.
This is a fundamental concept in cryptography, where we are interested in *cryptographically secure* encryption functions - meaning the outputs of the function cannot be distinguished from randomness by any statistical tests.

# Pseudorandom generator

Informal idea:

- Pseudorandom generator: a function $G$ is capable of producing a probability distribution $X$ that 'looks' random.
- detectors: $D$ is the function to which $X$ should appear random, i.e., $G$ should fool $D$ into believing that $X$ is random.
- $D$ samples from an unknown distribution, and has to determine the distribution is truly random or not.
- $G$ may take a truly random seed of length $m$ and $X$ may be distribution over $n > m$ bits.

- $D$ can only do statistical tests; i.e., sample from the distributions many times, compute something about each sample individually, then sum up the answers,

# Pseudorandom circuits

- A circuit with $l$ inputs is a representation of a $2^l$ bitstring - its truth table.
- A *size-limited* circuit with $l$ inputs is a *compact* representation of a $2^l$ bit-string.
  Not all $2^l$ bit strings have compact representation in this manner.
- Pseudorandom circuits are distributions over size-limited circuits. (distributions over a small subset of $2^l$ bit strings)

- Access: these distributions are not sampled as bitstrings.
  they are sampled at specific locations,
  i.e., we cannot ask for a (random) $2^l$ bit string, or a circuit;
  we can only ask for the $w^{th}$ bit of a random string, or a random circuit's output on input $w$.

# What is a PRF?

A **PRF** (pseudorandom function family) formalizes the notion of "pseudorandomness".
Informal idea:
The circuits in $Circuit^{O}[t]$ allow us to observe properties of the functions to test for non-random behavior. Functions in a PRF cheat these filters.

# Non-uniform dichotomy

# The PRF distinguisher game

This is the basis of Oliveira and Santhanam's proof that a non-pseudorandom class of functions has learners.

- Informally, you can view the definition of PRF as a 2-player game:
- One person tries to choose filters that detect non-randomness, the other person tries to choose functions from $C_n[s]$ that cheat the filters.

This allows us to use results from **game theory** to analyze PRFs.

# The PRF distinguisher game

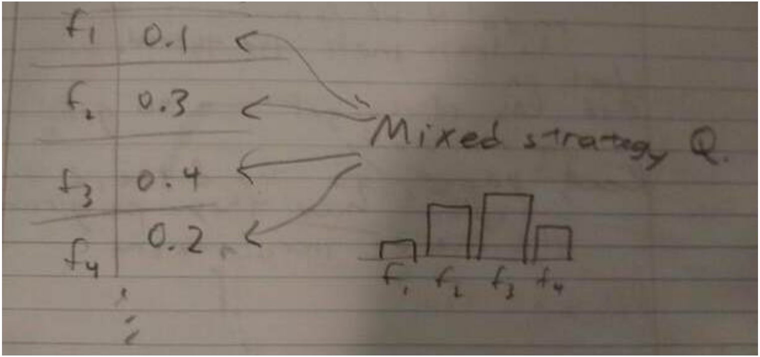
Game theory deals with a simple model of a 2-player game:

- In a two-player game, player 1 and player 2 both have a number of **pure strategies**
- The outcome of the game is a numeric value, the payoff for player 2.

These values can be placed in a matrix $M$.

# The PRF distinguisher game

Mixed strategy: A distribution over pure strategies.



# The PRF distinguisher game

Min-max theorem:

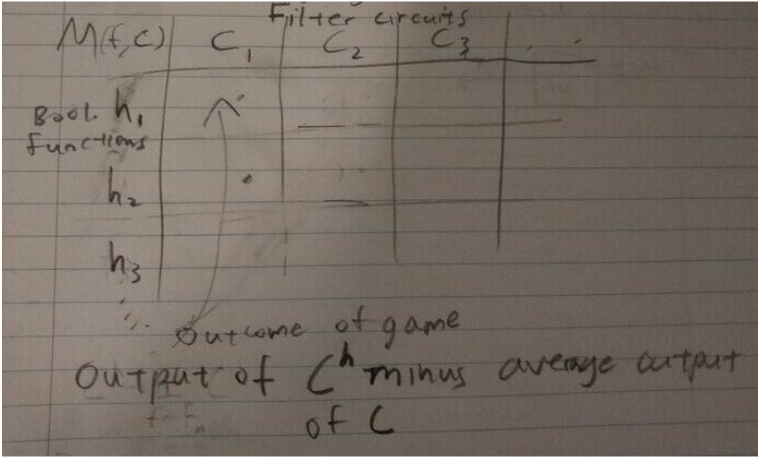
$$\min_P \max_j \mathbb{E}_{i \in P} M(i,j) = \max_Q \min_i \mathbb{E}_{j \in Q} M(i,j)$$

In English:

If player 2 has a response strategy for any probability distribution over player 1's mixed strategies that gives payoff at least $\varepsilon$, there is a single distribution over player 2's strategies that has payoff at least $\varepsilon$ for all of player 1's strategies.

# The PRF distinguisher game

We can set up a game that involves distinguishing PRFs.
Player 1 plays a function from a function family, player 2 plays a filter circuit.



# The PRF distinguisher game

In this context, the min-max theorem becomes:
If there are no $(t(n), \varepsilon)$-PRFs in $C_n[s]$, then there exists a distribution $Q$ of filter circuits such that for every function $h \in C_n[s]$, we have
$Pr_{C \in Q}[C^h = 1] - Pr_{C \in Q, f \in F_n}[C^f = 1] \geq \varepsilon$:
$Q$ is a **universal distinguisher**.

# Small support min-max theorem

Small support min-max theorem:
Like the min-max theorem, but it creates a small distribution $Q$.
We can make a randomized circuit $D$ that chooses one of the circuits from $Q$ at random.
$D$ is a universal distinguisher.

# Black-box generators

A black-box generator allows us to compute a family of functions $g_z : \{0,1\}^\ell \to \{0,1\}$ for $z \in \{0,1\}^m$, from a function $f$.
If $f \in C_n[poly(n)]$, then $g_z \in C_n[poly(n)]$.
There is an algorithm $A$ such that if $D$ is a distinguisher for the uniform distribution $W_L$ over all $g_z$, then $A^f(D)$ learns $f$.
All we need to do is find a distinguisher for $W_L$ and then we can use $A$ as our learner.
Oliveira and Santhanam plug in $D$ as their distinguisher for $W_L$ and output a learner.

# Non-uniformity

The problem with this result is that it does not tell us how to *find* these polynomial learners (the proof is non-constructive).
Because the proof involves the minmax theorem, and there is no known efficient algorithm to compute the max-min strategy, we do not know of an algorithm that inputs integer $n$ and outputs a learner for polynomial circuits with $n$ inputs. We say the learner is **non-uniform**.

# Uniform min-max and hardcore theorems

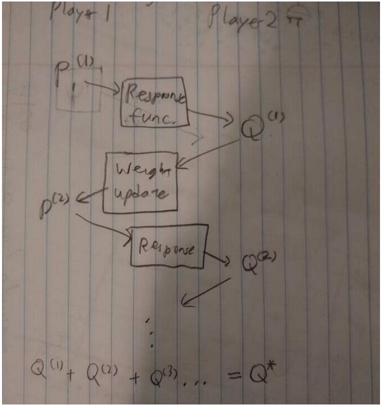
# Uniform min-max theorem

The uniform small support approximate min-max theorem was introduced by Vadhan and Zheng in 2014. This theorem gives an algorithm to find a small-support distribution that approximates the min-max/max-min strategy.

# Uniform min-max theorem

The uniform min-max theorem requires a function to get efficient responses $Q$ to each strategy $P$. The algorithm collects several of these responses into a single strategy $Q^*$, which is returned.
For all Player 1 pure strategies $P$:

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



# Uniform min-max theorem

This algorithm works by updating the weights of Player 1's strategy. At each round, Player 1 tries to minimize the next outcome by emphasizing the pure strategies on which the previous response would perform badly. It is a *boosting* algorithm.

# Uniform min-max theorem

In order for this theorem to give us a fast learner, we need some criteria:

- 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 $U_{[N]} \in Conv(\mathcal{V})$

# Uniform hardcore theorem

An application of the uniform min-max theorem described by Vadhan and Zheng is the **uniform hardcore theorem**. This is an extension of Impagliazzo's Hardcore Theorem.
This theorem involves:

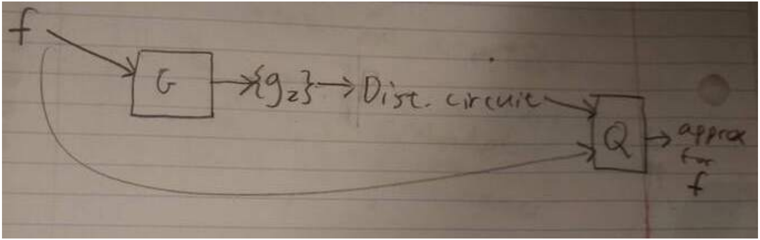
- A function $G$ mapping bitstrings of length $m$ to distributions over $\{0,1\}^L \times \{0,1\}$. The same function can generate distributions for different values of $L$ with inputs of different lengths.
- An algorithm that can predict $b$ given $x$ slightly better than average, when $(x, b)$ is taken from a "dense" subset of $(X, B) = G(U_m)$

The theorem says there is an algorithm $P$ such that for each $m$, $Pr_{(x,b)\in G(U_m)}[P(x) = b] > 1 - \delta$.
In other words, if there are weak predictors for dense distributions over $(X, B)$, there is a strong predictor for all of $(X, b)$.

# Our result

If there are no $(\varepsilon, t)$-PRFs in a circuit class $C_n[poly(n)]$, then the class $C_n[poly(n)]$ has efficient learners $A^O$ which output circuits of size $poly(n, 1/\gamma, size(D))$.

In our proof we assume there is an algorithm $W(1^n)$ that samples from a distribution over $C_n[poly(n)]$ and returns a distinguisher for that distribution.



# The End

Does anyone have any questions?