CS 6815: Pseudorandomness and Combinatorial Constructions **Fall 2022**

Lecture 12: Sep 29, 2022

Lecturer: Eshan Chattopadhyay *Scribe: Surendra Ghentiyala*

1 Hardness vs. Randomness Definitions

In this lecture, we explore the connection between constructing pseudorandom generators for a class of functions and constructing hard to compute functions against this class.

Definition 1.1. *$g : \{0,1\}^n \to \{0,1\}$ is (S, ϵ) hard if for all circuits C of size less or equal to than S, $corr(g, C) \leq \epsilon$ (ie. $\mathbb{P}_{x \sim U_n}[g(x) = C(x)] \leq \frac{1}{2} + \epsilon/2$).*

Intuitively, a function g is hard to compute if no "small" circuit can do much better at computing the function than just guessing. This captures the notion of average-case hardness. When we just have the weaker guarantee that $corr(g, C) < 1/2$, we just say g is S-hard for C. This weaker guarantee corresponds to worst-case hardness.

Definition 1.2. *A generator $G : \{0,1\}^{s(n,\epsilon)} \to \{0,1\}^n$ is (S, ϵ) pseudorandom if for all circuits C of size less than or equal to S, $|\mathbb{E}_{x \sim U_n}[C(G(x))] - \mathbb{E}[C(U_n)]| \leq \epsilon$.*

Intuitively, a generator G is pseudorandom if no "small" circuit can distinguish the outputs of G from truly random bits with significant advantage.

Remark 1.3. *We note that S, ϵ are functions of n, and what we really mean by g is actually a series of functions parameterized by n: $\{g_i\}_{i \geq 0}$.*

Definition 1.4. *Let $L_n = \{x : g_n(x) = 1\} \subseteq \{0,1\}^n$. The language associated with g is $L = \cup_{n \geq 0} L_n \subseteq \{0,1\}^*$. L is (S, ϵ) hard if g is (S, ϵ) hard.*

2 Pseudorandomness Implies Hardness

Claim 2.1. *Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be an $(S, \epsilon = 1/2 - \delta)$ pseudorandom generator, for any $\delta > 0$. Let $T = G(\{0,1\}^n)$ (the image of G). Define $f : \{0,1\}^{n+1} \to \{0,1\}$ as follows: $f(x) = 1$ if $x \in T$ and $f(x) = 0$ if $x \notin T$. f is S hard.*

We will show the above by contradiction. We will assume that f is not hard (that there is a series of small circuits that compute it) and then show that this implies that G is not (S, ϵ) pseudorandom.

Proof: Assume that f is not S hard. Let C be the circuit of size $\leq S$ such that $C(x) = f(x)$ for all x. We will now show that C breaks G.

Notice first that $\mathbb{E}[C(U_{n+1})] \leq \frac{1}{2}$ since $\mathbb{E}[C(U_{n+1})]$ is the fraction of strings in $\{0,1\}^{n+1}$ on which the circuit outputs 1. The circuit only outputs 1 when f outputs one, and f only outputs one when the input string is in the image of G. There are at most 2^n strings in the image of G. Thus, the fraction of strings on which the circuit accepts is $\leq \frac{2^n}{2^{n+1}} = \frac{1}{2}$. Therefore, $\mathbb{E}[C(U_{n+1})] \leq \frac{1}{2}$.

Secondly, notice that $\mathbb{E}_{x \sim U_n}[C(G(x))] = 1$.$C$ outputs 1 when its input is in the image of G. Clearly, $G(x)$ is in the image of G, thus $C(G(x))$ is always 1 and $\mathbb{E}_{x \sim U_n}[C(G(x))] = 1$.

Therefore

$$
\begin{aligned}
&|\mathbb{E}_{x \sim U_n}[C(G(x))] - \mathbb{E}[C(U_n)]| \\
&= \mathbb{E}_{x \sim U_n}[C(G(x))] - \mathbb{E}[C(U_n)] \\
&\geq 1 - \frac{1}{2} = \frac{1}{2}
\end{aligned}
$$

This provides the necessary contradiction to our assumption that G is a $(S, 1/2 - \delta)$ pseudorandom generator.

3 Hardness Implies Pseudorandomness

We now prove the more interesting direction.

Claim 3.1. *Suppose $f : \{0,1\}^n \to \{0,1\}$ is (S, ϵ) hard. Then $G : \{0,1\}^n \to \{0,1\}^{n+1}$ defined as $G(x) = (x, f(x))$ (where $(x, f(x))$ is x concatenated with $f(x)$) is (S', ϵ') pseudorandom, where $\epsilon' = \epsilon$ and $S' = S - 1$.*

Like the previous proof, this one will be by contradiction. We will assume that there is a small distinguisher for G and create a small circuit that can compute f.

Proof: Assume there is a curcuit that C, $|C| \leq S'$ such that

$$\mathbb{E}_{x \sim U_{n+1}}[C(G(x))] - \mathbb{E}[C(U_{n+1})] > \epsilon'$$

$$\mathbb{E}_{x \sim U_n}[C((x, f(x)))] - \mathbb{E}[C(U_{n+1})] > \epsilon'$$

Notice that sampling $n+1$ random bits is the same as sampling n random bits and then sampling 1 random bit and then concatenating them. Therefore the above statement is equivalent to

$$\mathbb{E}_{x \sim U_n}[C((x, f(x)))] - \mathbb{E}_{x \sim U_n, b \sim \{0,1\}}[C((x, b))] > \epsilon'$$

$$\mathbb{P}_{x \sim U_n}[C((x, f(x))) = 1] - \frac{1}{2}\mathbb{P}_{x \sim U_n}[C((x, f(x))) = 1] - \frac{1}{2}\mathbb{P}_{x \sim U_n}[C((x, \overline{f(x)})) = 1] > \epsilon'$$

$$\frac{1}{2}\mathbb{P}_{x \sim U_n}[C((x, f(x))) = 1] - \frac{1}{2}\mathbb{P}_{x \sim U_n}[C((x, \overline{f(x)})) = 1] > \epsilon'$$

$$\frac{1}{2}(\mathbb{P}_{x \sim U_n}[C((x, f(x))) = 1] - \mathbb{P}_{x \sim U_n}[C((x, \overline{f(x)})) = 1]) > \epsilon'$$

$$\mathbb{P}_{x \sim U_n}[C((x, f(x))) = 1] - \mathbb{P}_{x \sim U_n}[C((x, \overline{f(x)})) = 1] > 2\epsilon'$$

Therefore, the circuit is more likely to output 1 when $(x, f(x))$ is given as the input than when $(x, \overline{f(x)})$ is given as the input. We will use this observation to design a randomized algorithm A that takes an input x and uses C to compute $f(x)$. Then we will use A to design a circuit C' that computes f.

Let us now consider the probability that A successfully computes $f(x)$ on a random x.

$$\mathbb{P}_{x \sim U_n, b \sim \{0,1\}}[A(x) = f(x)]$$

Algorithm 1 A

$b \sim \{0,1\}$
if $C((x,b)) = 1$ **then**
  return b
end if
return $\bar{b}$

$$= \frac{1}{2}\mathbb{P}_{x\sim U_n,}[C((x,f(x))) = 1] + \frac{1}{2}\mathbb{P}_{x\sim U_n,}[C((x,\overline{f(x)})) = 0]$$

$$= \frac{1}{2}\mathbb{P}_{x\sim U_n,}[C((x,f(x))) = 1] + \frac{1}{2}\mathbb{P}_{x\sim U_n,}[C((x,\overline{f(x)})) = 0]$$

$$= \frac{1}{2}\mathbb{P}_{x\sim U_n,}[C((x,f(x))) = 1] + \frac{1}{2}(1 - \mathbb{P}_{x\sim U_n,}[C((x,\overline{f(x)})) = 1])$$

$$= \frac{1}{2} + \frac{1}{2}(\mathbb{P}_{x\sim U_n,}[C((x,f(x))) = 1] - \mathbb{P}_{x\sim U_n,}[C((x,\overline{f(x)})) = 1])$$

We already show showed above that $(\mathbb{P}_{x\sim U_n,}[C((x,f(x))) = 1] - \mathbb{P}_{x\sim U_n,}[C((x,\overline{f(x)})) = 1]) > 2\epsilon'$. Consequently, the above expression evaluates to

$$= \frac{1}{2} + \epsilon' = \frac{1}{2} + \epsilon$$

Now we need to turn A into a series of circuits. Let A_b be the algorithm A so that rather than sampling the variable b, it has b fixed as b. Observe that since $\mathbb{P}_{x\sim U_n, b\sim\{0,1\}}[A(x) = f(x)] \geq \frac{1}{2} + \epsilon$, there must be a bit $b \in \{0,1\}$ such that the $\mathbb{P}_{x\sim U_n}[A_b(x) = f(x)] \geq \frac{1}{2} + \epsilon$. So for each circuit in the circuit family that will compute f, we will hard code b so that it is the bit with the aforementioned property.

So, the circuit C' will compute and output $C((x,1))$ if $b = 1$ and $\overline{C((x,0))}$ if $b = 0$. This makes C' equivalent to A_b for the best choice of b.

It is easy to see that the extra computation means that the size of C' is $|C| + 1$, and thus we contradict our hardness assumption.

4 Nisan-Wigderson PRG

Nisan-Wigderson showed a way to construct a much better PRG from hardness asssumptions. We will discuss this in next class, and provide some intuition here.

We have shown that the assumption of a hard function f allows us to extend n bits to $n+1$ bits. The Nisan-Wigderson PRG goes further and gives us exponential stretch.

On a high level, the PRG G samples $r = poly(n)$ bits and generates bits $z_1, z_2, \ldots, z_m$ where $m = 2^{\Omega(n)}$.

To do so, we fix a set system $S_1, S_2, \ldots, S_m \subseteq [r], |S_i| = n$, and set z_i to be $f(x_{S_i})$

For the construction, we will see that an additional 'design property' is needed on the set system that bounds the pairwise intersection of any two sets:

$$\forall i \neq j, |S_i \cap S_j| \leq \frac{n}{c}$$

where c is some constant.

We will discuss this in much more detail in the next class.