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

Lecture 7: Sep 13, 2022

Lecturer: Eshan Chattopadhyay *Scribe: Noam Ringach*

1 Introduction

In this lecture, we will see the connection between PRGs and ε-biased spaces, and we will begin building up our knowledge of algebraic error-correcting codes to eventually see their connection to PRGs as well.

2 ε-biased spaces and PARITY

Here, we will connect PRGs to a class of distributions called ε-biased spaces/distributions and see their dual view as exactly the PRGs that fool languages in PARITY. Before introducing ε-biased distributions, we need to understand the Boolean function class PARITY.

Definition 2.1 (The class PARITY)**.** *The Boolean funciton class* PARITY *is the class of decision problems solvable by a nondeterministic polynomial-time Turing machine where the acceptance condition is that the number of accepting paths is odd (i.e., their parity is equal to 1). Another way of viewing* PARITY *is as the set of functions $\mathcal{F} = \bigcup_{n\in\mathbb{N}} \mathcal{F}_n$ where each $\mathcal{F}_n$ is the set of all parity functions on n elements. Explicitly, we can write*

$$\mathcal{F}_n := \{\chi_S : \{0,1\}^n \to \{0,1\} \mid \forall S \subseteq [n]\}$$

where χ_S is the parity function on a subset of indices defined as

$$\chi_S(x) := \bigoplus_{i\in S} x_i.$$

Alternatively, instead of taking the parity directly we can simply do addition in $\mathbb{F}_2^n$ and define

$$\begin{aligned}\chi_S(x) : \mathbb{F}_2^n &\to \mathbb{F}_2\\ x &\mapsto \sum_{i\in S} x_i.\end{aligned}$$

Now, recall from previous lectures that we can define a PRG by the class of functions that it fools. In the context of PARITY, we will see that the PRGs that fool PARITYare exactly those that are also ε-biased distributions! We now proceed to define these distributions and give an explicity construction of such PRGs.

Definition 2.2 (ε-biased space)**.** *Let U_n be the uniform distribution on n bits. An ε-biased space/distribution (also referred to as a small-bias sample space) is a distribution D on $\{0,1\}^n$ such that for all $S \subseteq [n]$ we have*

$$\left| \mathop{\mathbb{E}}_{x\sim U_n}[\chi_S(x)] - \mathop{\mathbb{E}}_{x\sim D}[\chi_S(x)] \right| \leq \varepsilon.$$

In other words, D is a distribution that fools all parity functions.

We can obtain a dual view of an ε-biased distribution by considering D to be the output of a PRG $G_n : \{0,1\}^{s(n,\varepsilon)} \to \{0,1\}^n$ such that

$$\left| \mathop{\mathbb{E}}_{x \sim U_n} [\chi_S(x)] - \mathop{\mathbb{E}}_{y \sim U_{s(n,\varepsilon)}} [\chi_S(G_n(y))] \right| \leq \varepsilon.$$

We would say that such a PRG fools PARITY. Surprisingly, while the class of problems in PARITYis very broad, we can explicitly construct an efficient algorithm for computing G_n.

2.1 Constructing a PRG to fool PARITY

We begin by considering the case where $S \subseteq [n]$ is empty. If we define $\chi_S(x) = 0$ when $S = \varnothing$, then we simply see that $\mathbb{E}_{x \sim U_n}[\chi_S(x)] = 0$, so fooling this constant function is trivial since we can simply let $G_0 = 0$. On the other hand, if $S \neq \varnothing$, then we have that $\mathbb{E}_{x \sim U_n}[\chi_S(x)] = \frac{1}{2}$. Thus, our goal is that for all nonempty $S \subseteq [n]$ we have

$$\mathop{\mathbb{E}}_{y \sim U_{s(n,\varepsilon)}} [\chi_S(G_n(y))] \in \left(\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon \right). \tag{1}$$

While we won't go into error-correcting codes yet, it turns out that the connection of ε-biased spaces to error-correcting codes has shown that the optimal lower bound on the seed length for G_n is $s(n,\varepsilon) \geq \log(n) + 2\log\left(\frac{1}{\varepsilon}\right)$. In fact, the current state of the art has only shown an explicit construction for $s(n,\varepsilon) = \log(n) + (2 + o(1))\log\left(\frac{1}{\varepsilon}\right)$.[1] We now present our specific construction.

Construction 2.3. *For a desired output length n and approximation parameter $\varepsilon > 0$, let our seed length be $r = s(n,\varepsilon)$, let $q = 2^r$, and define our base field as $\mathbb{F} = \mathbb{F}_q$. Our PRG G_n now proceeds as follows:*

1. *Sample $x, y \sim \mathbb{F}$ uniformly*
   (a) *This requires $2\log(2^r) = 2r$ bits.*
2. *Compute $z_i = \langle x, y^i \rangle$*
   (a) *Here, the power y^i is taken in $\mathbb{F} = \mathbb{F}_q$, but we take the inner product in $\mathbb{F}_2^r \approx \mathbb{F}_q$ by considering x and y_i as n-length vectors with elements in $\mathbb{F}_2$.*
3. *Output $z = (z_1, z_2, \ldots, z_n)$.*

We claim that G_n satisfies (1).

Proof. Because the output of χ_S is an element of $\mathbb{F}_2$, we can reduce the condition in (1) to

$$\Pr_{z \sim G_n} [\chi_S(z) = 0] \in \left(\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon \right) \tag{2}$$

[1] See *Explicit, Almost Optimal, Epsilon-Balanced Codes* (Ta-Shma, 2017) for more details.

for all nonempty $S \subseteq [n]$. By expanding the definition of χ_S and using the bilinearity of the inner product, we get that

$$\begin{aligned}
\Pr_{z \sim G_n}[\chi_S(z) = 0] &= \Pr_{z \sim G_n}\left[\sum_{i \in S} z_i = 0\right] \\
&= \Pr_{z \sim G_n}\left[\sum_{i \in S} \langle x, y^i \rangle = 0\right] \\
&= \Pr_{z \sim G_n}\left[\left\langle x, \sum_{i \in S} y^i \right\rangle = 0\right] \\
&= \Pr_{z \sim G_n}\left[\langle x, P_S(y) \rangle = 0\right],
\end{aligned}$$

where we define $P_S(y) = \sum_{i \in S} y^i$. By the same logic as just taking the parity over two bit strings (i.e., that if we fix the rest of the bits and only flip the last one, then taking parity will give us 0 half the time and 1 the rest of the time), we easily see that for a fixed $P_S(y) \neq 0$, we have that $\langle x, P_S(y) \rangle$ is uniform over $\mathbb{F}_2$ for $x \sim \mathbb{F}_q$, meaning that

$$\Pr_{x \sim \mathbb{F}_q}[\langle x, P_S(y) \rangle = 0] = \frac{1}{2}.$$

Observe, however, that if $P_S(y) = 0 \in \mathbb{F}_q$, then $\langle x, P_S(y) \rangle = 0$ for all $x \in \mathbb{F}_q$, which ruins our overall randomness of $\langle x, P_S(y) \rangle$ by biasing it towards 0.

Nevertheless, because we can consider $P_S(y)$ as a univariate polynomial in y over $\mathbb{F}_q$ with at most degree n, we see that

$$\Pr_{y \sim \mathbb{F}_q}[P_S(y) = 0] \leq \frac{n}{q}$$

because $P_S(y)$ can have at most n zeroes and $|\mathbb{F}_q| = q$. Therefore, combining this with the fact that for a fixed $P_S(y) \neq 0$, we have that $\langle x, P_S(y) \rangle$ is uniform over $\mathbb{F}_2$ for $x \sim \mathbb{F}_q$, we see

$$\Pr_{x,y \sim \mathbb{F}_q}\left(\left\langle x, \sum_{i \in S} y^i \right\rangle = 0\right) \in \left(\frac{1}{2} - \frac{n}{q}, \frac{1}{2} + \frac{n}{q}\right).$$

In other words, to satisfy (2), we solve $\varepsilon = \frac{n}{q}$ to get $r = \log(q) = \log \frac{n}{\varepsilon} = \log(n) + \log \frac{1}{\varepsilon}$. Note, though, that we do not beat the SOTA of $\log(n) + (2 + o(1)) \log\left(\frac{1}{\varepsilon}\right)$ since we need to sample $2r = 2\log(n) + 2\log\left(\frac{1}{\varepsilon}\right)$ random bits. □

3 (Brief introduction to) Error-Correcting Codes

It turns out that that there are important connections between the pseudorandom primitives we have seen so far and error-correcting codes (that we now introduce).

The concept of error-correcting codes comes up naturally when we consider noisy communication between two entities.

Example 3.1. *Consider two people, Alice and Bob. Alice has a message m that she would like to send to Bob, but in order to do that m must pass through a noisy channel. Consequently, Bob receives a noisy m' and must figure out what was the original message that Alice intended to send.*

In the simplest case of Alice sending Bob a single bit b and the noisy channel being allowed to change at most one bit, we can come up with the simple error-correcting code of Alice sending (b, b, b) (i.e., just repeating b three times) and Bob taking the majority vote to get Alice's intended message. This majority strategy works every time since at most one bit can be corrupted.

In general, we will desire for an error-correcting code over an alphabet Σ (we will often consider $\Sigma = \mathbb{F}_2$) to have an encoder $\text{Enc} : \Sigma^k \to \Sigma^n$, where k is the dimension of our message space and n is the dimension of our code, and a decoder $\text{Dec} : \Sigma^n \to \Sigma^k$ such that $\text{Dec}(\text{Enc}(m)) = m$ for all $m \in \Sigma^k$ as long as too many bits of the message have not been corrupted. Because we measure corruption in messages as switching out one symbol in Σ for another one in Σ, we measure the distance between two encoded messages using the Hamming distance.

Definition 3.2 (Hamming distance)**.** *For any two $c, c' \in \Sigma^n$, their Hamming distance $\Delta(c, c')$ is defined as*

$$\Delta(c, c') = \left|\{i : c_i \neq c'_i\}\right|.$$

Note that the Hamming distance is indeed a distance metric, and thus defines a topology on its given space.

Using this definition, we can formally define error-correcting codes as follows:

Definition 3.3. *An $(n, k, d)_q$ code $\mathcal{C}$,[2] where n is the block length, k is the dimension of the message space, d is the distance, and q is the alphabet size (often referring to $\mathbb{F}_q$) is a subset $\mathcal{C} \subseteq \Sigma^n$ such that $\log_2 |\mathcal{C}| = k$ and $\min_{c,c' \in \mathcal{C}} \Delta(c, c') \geq d$. In other words, $\mathcal{C}$ can correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors and detect up to $d - 1$ errors.*

In light of this definition, we see that our code in Example 3.1 has distance $d = 3$, so it can correct $\left\lfloor \frac{3-1}{2} \right\rfloor = 1$ errors, as we designed it to. The quality of an error-correcting code is measured in terms of its relative rate and relative distance defined below.

Definition 3.4 (Relative rate/distance)**.** *For an $(n, k, d)_q$ code $\mathcal{C}$, its relative rate is $r = \frac{k}{n}$ and its relative distance is $\delta = \frac{d}{n}$.*

Notice that since $k < n$ and $d < n$, we have that both $r, \delta \in [0, 1]$. In particular, a higher relative rate means that the sender, Alice, doesn't have to augment her message too much before sending it to Bob. This is good, since we don't want to have to add many bits when encoding a message, meaning that we want to get r as close to 1 as possible. Similarly, a higher relative distance means that we can correct a larger fraction of the errors in the message, so we want to get δ as close to 1 as possible as well.

As with any optimization problem with two orthogonal objectives, we usually have to balance optimizing for relative rate and relative distance when constructing error-correcting codes. And while we have good existential results, constructing an explicit code that matches these bounds is a challenging open problem. We will explore the connection between these codes and PRGs in the next lecture.

[2]The encoder and decoder for $\mathcal{C}$ are implicitly defined along with $\mathcal{C}$.