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

Lecture 19: October 27, 2022

Lecturer: Eshan Chattopadhyay *Scribe: Tomas Alvarez, Noam Ringach*

1 Introduction

In our last lecture, we saw an explicit construction of a (k, ε)-extractor $\text{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ such that for all $k < n$ we have $d = O(n)$ and $m = k - 2\log\left(\frac{1}{\varepsilon}\right) - O(1)$. So we are extracting almost all of the randomness from our (n, k)-source, but the size of our seed is linear with respect to n, which is sub-optimal. In fact, we know from the existential construction of an extractor via the probabilistic method that we can get d as low as $d = \log(n-k) + 2\log\left(\frac{1}{\varepsilon}\right) + O(1)$. Here, we will improve upon this by using the Nisan-Wigderson PRG to create an extractor that only requires $d = O(\log(n))$ random bits and extracts $m = \sqrt{k - \log(1/\varepsilon) + O(1)}$ bits.

2 Trevisan's extractor and connection to the Nisan-Wigderson PRG

At first glance, it might not seem obvious how we can use the Nisan-Wigderson PRG to obtain a good randomness extractor. Recall, the Nisan-Wigderson PRG takes in a (s, ε)-hard function and uses a set design to extend $O(\mathsf{polylog}(n))$ random bits to n bits that looks $n\varepsilon$-random to circuits of size $s' = s - O\left(n2^k\right)$. On the other hand, for a seeded extractor we are given a seed y, and a sample from X, an (n, k)-source, from which we would like to approximnately produce U_m, the uniform distribution on m bits. The first insight from Trevisan is to consider samples $x \in \{0,1\}^n$ from X as functions $x : \{0,1\}^{\log(n)} \to \{0,1\}$. The function for a fixed sample takes in an index i and outputs 0 or 1 corresponding to the i'th bit in x. For example, for $x = 011\cdots$ we would say $x(1) = 0, x(2) = 1, x(3) = 1$ and so on.

3 Extractor construction

This construction is simple and relies heavily on the Nisan-Wigderson PRG. For the extractor we are given (n, k)-source X and $x \in \{0,1\}^n$ a sample from X and a seed $y \in \{0,1\}^r$. We also have access to a set system $S_1, \cdots, S_m \subseteq [r]$ which is an $(\bar{n}, \lambda)$-design. As before, we think of x as a function $x : \{0,1\}^{\bar{n}} \to \{0,1\}$ where $\bar{n} = \log(n)$. Now we claim $\text{Ext}(x, y) = NW^x(y) = x(\alpha_1) \circ x(\alpha_2) \circ \cdots \circ x(\alpha_m)$ is a (k, ϵ) seeded extractor. In this case $NW^x(y)$ denotes the output of the Nisan-Wigderson PRG with x as the hard function and y as the input, $a \circ b$ denotes concatenation, and $\alpha_i = y|_{S_i} \in \{0,1\}^{\bar{n}}$ is the projection of y onto the indices in the set S_i. Next, we prove that this construction indeed gives us an extractor.

Proof. Suppose for the sake of contradiction that Ext is not a (k, ϵ) extractor where k is yet to be determined. Then WLOG there exists a distinguisher $D : \{0,1\}^m \to \{0,1\}$ such that

$$\Pr_{x \sim X, y \sim Y}[D(NW^x(y)) = 1] - \Pr[D(U_m) = 1] > \epsilon$$

Now a Markov argument shows that

$$\Pr_{x\sim X}\left[\left|\Pr_{y\sim Y}[D(NW^x(y))=1]-\Pr[D(U_m)=1]\right|>\frac{\epsilon}{2}\right]>\frac{\epsilon}{2}$$

For any x where the distinguisher is good at distinguishing we can construct hybrids

$$\begin{aligned}
H_0 &= x(y|_{S_1})\circ x(y|_{S_2})\circ\cdots\circ x(y|_{S_m})\\
H_1 &= b_1\circ x(y|_{S_2})\circ\cdots\circ x(y|_{S_m})\\
H_2 &= b_1\circ b_2\circ\cdots\circ x(y|_{S_m})\\
H_i &= b_1\circ b_2\circ\cdots\circ b_i\circ x(y|_{S_{i+1}})\circ\cdots\circ x(y|_{S_m})\\
&\vdots\\
H_m &= b_1\circ b_2\circ\cdots\circ b_m
\end{aligned}$$

where $b_1, b_2, \cdots b_m \sim U_1$. As in the Nisan-Wigderson PRG analysis we can rewrite $\Pr\limits_{y\sim Y}[D(NW^x(y))=1]-\Pr[D(U_m)=1]$ as a telescoping sum between the m consecutive hybrids and since their sum is at least $\epsilon/2$ we conclude that there exists some index i for which $|\Pr[D(H_i)=1]-\Pr[H_{i+1}=1]|\geq\frac{\epsilon}{2m}$ and thus this is an approximator of x.

Sampling at most m random bits $b_1, b_2, \cdots, b_i$, calculating $x(y|_{S_{i+1}}), x(y|_{S_{i+2}}), x(y|_{S_m})$ with a circuit of size $m2^\lambda$ and a description of i in $\log(m)$ bits, gives the distinguisher access to H_i and H_{i+1}. If we let $2^\lambda = m$ then this is $O(m^2)$ bits of information and knowledge of D to differentiate between the two hybrids. We can rewrite all of this to more closely resemble an approximator of x as a function which is given D, the extra $O(m^2)$ information, referred to as β, as follows: $\Pr\limits_{\alpha}[g^D(\alpha,\beta)=x(\alpha)]\geq\frac{1}{2}+\frac{\epsilon}{2m}$.

We now consider the set $Bad_X=\{x\in X: |\Pr\limits_{y\sim Y}[D(NW^x(y))=1]-\Pr[D(U_m)=1]|>\frac{\epsilon}{2}\}$ over which our analysis was done. If we fix β based on some set design then we know for $x\in Bad_X$ that $\Pr\limits_{\alpha}[g^D(\alpha,\beta)=x(\alpha)]\geq\frac{1}{2}+\frac{\epsilon}{2m}$. We can actually strengthen the inequality to $\Pr\limits_{\alpha}[g^D(\alpha,\beta)=x(\alpha)]=1$ if we are given some extra information. We continue with this assumption and show how to achieve equality in the next section. With this strengthened form, the function is no longer an approximator, but rather fully determines x given some β. If we assume WLOG that X is flat then $|Bad_X|>\frac{\epsilon}{2}2^k$ since it is an (n,k)-source and $\Pr\limits_{x\sim X}[|\Pr\limits_{y\sim Y}[D(NW^x(y))=1]-\Pr[D(U_m)=1]|>\frac{\epsilon}{2}]>\frac{\epsilon}{2}$ means at least $\epsilon/2$ of all $x\in X$ are in Bad_X. Since there are only $2^{O(m^2)}$ different β's (set designs), and each one fully determines a value of x, it cannot be the case that we have more x's in Bad_X than g's. Thus, for contradiction let $\frac{\epsilon}{2}2^k>2^{O(m^2)}\implies k=m^2+\log(1/\epsilon)+O(1)$. Since this is a contradiction we have that Ext is a (k,ϵ) extractor.

□

4 Aid of list decodable codes

Recall from above that we assumed that we could achieve $\Pr_\alpha\left[g^D(\alpha,\beta)=x(\alpha)\right]=1$ by giving g^D some "extra information" contained in β. But how is this extra information determined and how can we guarantee that we don't require too much extra information (i.e., at most $O(m^2)$ extra information so that we don't hurt our complexity)? The strategy for efficiently constructing such extra information becomes clear if we consider $x\in\{0,1\}^n$ to be a codeword in some code which we will determine later[1].

Let $z_{\beta_i} = g^D(\alpha_1, \beta_i) \circ \cdots \circ g^D(\alpha_m, \beta_i)$ be the concatenation of g^D given β_i over all α_i's. Recall that, without any extra tricks, we already have $\Pr_\alpha\left[g^D(\alpha,\beta) = x(\alpha)\right] \geq \frac{1}{2} + \frac{\varepsilon}{2m}$, meaning that for any $x \in Bad_X$ we have that x is in a $n\left(\frac{1}{2} - \frac{\varepsilon}{2m}\right)$-Hamming ball around some z_{β_i}. In other words, all the elements of Bad_X are in $n\left(\frac{1}{2} - \frac{\varepsilon}{2m}\right)$-Hamming balls around the z_{β_i}'s. However, we need to be able to identify a specific x in one of these balls around a z_{β_i}, which is where our extra information comes in. If there aren't too many x's around each z_{β_i}, then we can use very few bits to identify a specific x (by simply indexing them, for example). This general idea of not having too many points within Hamming balls of a certain radius is made concrete by List-decodable codes.

Definition 4.1 (List-decodable codes)**.** *We say that a code $\mathcal{C} \subset \{0,1\}^t$ is a $(\gamma, L)-$binary List-decodable code if for all $z \in \{0,1\}^n$ we have*

$$\left|B\left(z, \left(\frac{1}{2} - \gamma\right)t\right) \cap \mathcal{C}\right| \leq L,$$

where $B(z,\rho)$ is the Hamming ball of raidus ρ about z. In other words, there are at most L codewords in the Hamming ball of radius $\left(\frac{1}{2} - \gamma\right)t$ about any z.

Thus, if we structure Bad_X as a (γ, L)-List-decodable code, then we only need $\log(L)$ more bits to identify an individual x around any z_{β_i}. Now the question of course becomes whether there exist such List-decodable codes with proper γ and L that will satisfy our requirements. Thankfully, the following theorem (which we will not prove) guarantees the existence of such codes.

Theorem 4.2. *There exist (γ, L)-list-decodable codes with an encoder* $\mathrm{Enc} : \{0,1\}^{\bar{t}} \to \{0,1\}^t$ *with parameters $L = \mathsf{poly}\left(\frac{1}{\gamma}\right)$ and $t = \mathsf{poly}\left(\bar{t}, \frac{1}{\gamma}\right)$.*

Therefore, if we choose $\gamma = \frac{\varepsilon}{2m}$ then we only need

$$\begin{aligned}
\log(L) &= \log\left(\mathsf{poly}\left(\frac{2m}{\varepsilon}\right)\right) \\
&= \log\left(\left(\frac{2m}{\varepsilon}\right)^{O(1)}\right) \\
&= O\left(\log\left(\frac{m}{\varepsilon}\right)\right) \\
&= O\left(m^2\right).
\end{aligned}$$

Consequently, the extra information required to choose a codeword from a list-decodable code isn't asymptotically more than what we're already using. In this way, we are able to encode each $x \in \{0,1\}^n$ into a space $\{0,1\}^{n'}$ where $n' = \mathsf{poly}\left(\gamma n, \frac{2m}{\varepsilon}\right)$ before using them in the Nisan-Wigderson PRG to get our desired guarantee of $\Pr_\alpha\left[g^D(\alpha,\beta) = x(\alpha)\right] = 1$.

[1] Note that we in fact have $n = m$ since by construction, but we will keep using n and m separately to keep the context clear.