U.C. Berkeley | Handout N18
CS294: Pseudorandomness and Combinatorial Constructions | November 1, 2005
Professor Luca Trevisan | Scribe: Constantinos Daskalakis

Notes for Lecture 18

1 Basic Definitions

In the previous lecture we defined the notion of a *randomness extractor* as follows:

Definition 1 *A function $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-*extractor *if, for every random variable X of range $\{0,1\}^n$ -which we will call a* source- *of* min-entropy *$H_\infty(X) \geq k$, the following holds for the* statistical difference *of $Ext(X, U_d)$ and U_m:*

$$||Ext(X, U_d) - U_m||_{SD} \leq \epsilon,$$

where:

- *we denote by U_l the uniform distribution on $\{0,1\}^l$*
- *the* min-entropy *of a random variable X of a finite range $\mathcal{A}$ is defined as $H_\infty(X) = \min_{a \in \mathcal{A}} \frac{1}{\mathbf{Pr}[X=a]}$*
- *the* statistical difference *$||\cdot||_{SD}$ of two random variables Y and Z of a finite range $\mathcal{A}$ is defined as:*
$$||Y - Z||_{SD} = \max_{T:\mathcal{A} \to \{0,1\}} |\mathbf{Pr}[T(Y) = 1] - \mathbf{Pr}[T(Z) = 1]|$$

The general goal is to keep the “high-quality” input randomness -parameter d of the model- as small as possible, while maintaining the output randomness -parameter m- as close to the sum $k + d$ as possible. Last time, we showed some negative results asserting that the best tradeoff we can hope for is something of the flavor:

$$m = k + d - 2\log\frac{1}{\epsilon} + \Theta(1)$$

$$\text{and } d = \log(n-k) + 2\log\frac{1}{\epsilon} - \Theta(1)$$

Today we will give a construction of randomness extractors. But first we will motivate our pursuit by giving an interesting application.

2 An Example Application of Extractors

Suppose that A_L is a randomized algorithm for a language $L \subseteq \{0,1\}^l$, which uses m bits of randomness and has error probability $\leq \frac{1}{4}$. A common way to boost the probability of success of A_L is to execute it t times on independent randomness and output the majority answer. In this case, the random bits that are needed are $t \cdot m$ and the probability of error is bounded by $e^{-\Omega(t)}$.

A different way to boost the probability of success is via the use of randomness extractors. Indeed, suppose that $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a $(k, \frac{1}{8})$-extractor and let us define algorithm A'_L as follows:

```
Algorithm $A'_L$
Input: $z \in \{0,1\}^l$
    pick $n$ truly random bits $\mathbf{r}$
     compute $\mathbf{m_1} = Ext(\mathbf{r}, 00...0)$ and then $A_{L,\mathbf{m_1}}(z)$
     compute $\mathbf{m_2} = Ext(\mathbf{r}, 10...0)$ and then $A_{L,\mathbf{m_2}}(z)$
     ...
     compute $\mathbf{m_{2^d}} = Ext(\mathbf{r}, 11...1)$ and then $A_{L,\mathbf{m_{2^d}}}(z)$
    output the majority of $A_{L,\mathbf{m_1}}(z), \ldots, A_{L,\mathbf{m_{2^d}}}(z)$
```

The performance of A'_L can be analyzed as follows. Let us fix an input $z \in \{0,1\}^l$ and let us define

$$B_z := \{r \in \{0,1\}^n | \text{the output of } A'_L \text{ is wrong if } r \text{ is picked at the first step}\}.$$

If X is uniform over B_z, then, by definition, $H_\infty(X) = \log |B_z|$. Moreover, since for every $r \in B_z$ the answer of A'_L is wrong, it follows that for this specific r the majority of the computations of A_L within A'_L output the wrong answer. Therefore, $\mathbf{Pr}[A_{L,Ext(X,U_d)}(z) = \text{incorrect answer}] > \frac{1}{2}$. However, the algorithm A_L outputs the correct answer with probability of error at most 1/4. Therefore, $\mathbf{Pr}[A_{L,U_m}(z) = \text{incorrect answer}] \leq \frac{1}{4}$ and, thus,

$$||Ext(X, U_d) - U_m||_{SD} > \frac{1}{4}.$$

Now, if we assume that $B_z \geq 2^k$, i.e. $H_\infty(X) \geq k$, the above inequality contradicts the fact that Ext is a $(k, \frac{1}{8})$ randomness extractor.

Therefore, it must be that $B_z < 2^k$ for every $z \in \{0,1\}^l$ and, therefore,

$$\mathbf{Pr}[A'_L \text{ outputs incorrect answer}] \leq \frac{2^k}{2^n} \leq 2^{k-n}.$$

The above inequality asserts that every extra bit of randomness that is invested above the value k results in a decrease of the probability of error by a factor of 2. This is actually the optimal rate with which probability of error can decrease.

In fact -to compare the boosting achieved by extractors with other methods of boosting- let us note that, if we want to decrease the probability of error of A_L to ϵ, then:

- boosting with **independent repetitions** needs $O((\log \frac{1}{\epsilon}) \cdot m)$ random bits
- boosting with a $(k, \frac{1}{8})$ **extractor** with $m \simeq k^{0.99}$ ($k \simeq m^{1.01}$) needs $m^{1.01} + \log \frac{1}{\epsilon}$ random bits
- boosting with a **random walk on an expander** needs $O(m + \log \frac{1}{\epsilon})$ random bits (in fact at least $m + 2 \log \frac{1}{\epsilon}$)

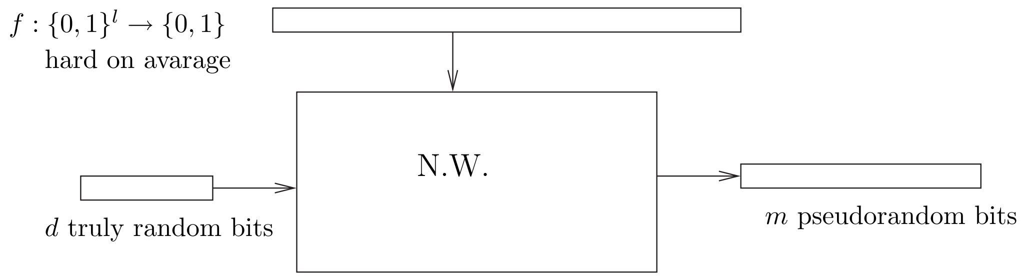
Therefore, for non-trivial boosting applications, the use of extractors is more economical in terms of required randomness.

3 Construction of Randomness Extractors

In this section, we investigate how a construction similar to the Nisan-Wigderson generator could be used to obtain randomness extractors. Below we briefly describe the structure of the Nisan-Wigderson generator and its analysis, but for more details one should consult the notes of lectures 11 and 12.

Brief Description of the Nisan-Wigderson Pseudorandom Generator

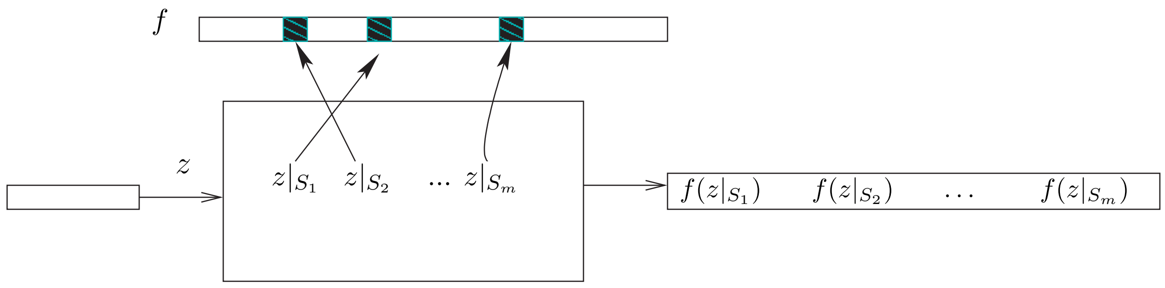
Recall that the Nisan-Wigderson pseudorandom generator has the following high level structure



More precisely, the functionality of the generator relies on the construction of a set system $S_1, S_2, \ldots, S_m \subseteq [d]$ with the following properties:

- $|S_i| = l$, for every i
- $|S_i \cap S_j| \leq \alpha$, for every $i \neq j$

Such a set system can be constructed with $\alpha = \log m$, $l = c \log m$ and $d = e^2c^2 \log m$, and the generator has the following functionality:



We, now, briefly describe the *analysis* of the Nisan-Wigderson generator; elements of this analysis will be useful later in constructing the randomness extractor. Let us denote by $NW^f : \{0,1\}^d \to \{0,1\}^m$ the function defined by the Nisan-Wigderson generator, that is, $NW^f(z) = f(z|_{S_1}) \ldots f(z|_{S_m})$, and suppose, by contradiction, that there exists a distinguishing circuit D :

$\{0,1\}^m \to \{0,1\}$ such that

$$\left| \Pr_{z \sim U_d}[D(NW^f(z)) = 1] - \Pr_{z \sim U_m}[D(z) = 1] \right| \geq \epsilon. \tag{1}$$

Without loss of generality, we can remove the absolute value from the above inequality and apply a hybrid argument using a set of hybrid distributions $H_0, H_1, \dots, H_m$ defined as follows

$$H_i = r_1 r_2 \dots r_{i-1} r_i f(z|_{S_{i+1}}) \dots f(z|_{S_m}), \text{ where } (r_1 \dots r_i) \sim U_i \text{ and } z \sim U_d,$$

to conclude that

$$\epsilon \leq \Pr_{z \sim U_d}[D(NW^f(z)) = 1] - \Pr_{z \sim U_m}[D(z) = 1] = \sum_{i=1}^{m} (\Pr[D(H_{i-1}) = 1] - \Pr[D(H_i) = 1]).$$

Thus, if we define algorithm A as follows

```
Algorithm A
Input: $x, b$ /*can be either $x, f(x)$ or $x, r$*/
    pick $i \in \{1, \dots, m\}$ /*pick one of the hybrids at random*/
    pick $r_1, \dots, r_{i-1} \in \{0,1\}$ at random
    pick $z \sim U_m$ conditioned on $z|_{S_i} = x$
    output $D(r_1 \dots r_{i-1} b f(z|_{S_{i+1}}) \dots f(z|_{S_m}))$
```

we have that

$$\Pr[A(x, f(x)) = 1] - \Pr[A(x, r) = 1] = \mathbb{E}_i (\Pr[D(H_{i-1}) = 1] - \Pr[D(H_i) = 1]) \geq \frac{\epsilon}{m}$$

where the probabilities of the left hand side are taken both over the inputs to the algorithm and over the randomness used by the algorithm itself. In fact, we can fix the randomness used inside the algorithm, i.e. index i, bits $r_1, r_2, \dots, r_{i-1}$ and the bits of z that are not indexed by S_i, so that, with these values fixed, it still holds that

$$\Pr[A(x, f(x)) = 1] - \Pr[A(x, r) = 1] \geq \frac{\epsilon}{m},$$

which implies that either $\overline{A(x,0)}$ or $A(x,1)$ computes f correctly on at least a $\frac{1}{2} + \frac{\epsilon}{m}$ fraction of the inputs.

In general, that is, independent of the way the absolute value from (1) is removed, we can fix index i, bits $r_1, \dots, r_{i-1}$ and the bits of z not indexed by S_i, so that one of $A(x,0), A(x,1), \overline{A(x,0)}, \overline{A(x,1)}$ computes f correctly on at least a $\frac{1}{2} + \frac{\epsilon}{m}$ fraction of the inputs. The later assertion contradicts the hardness on average of function f (after playing a bit with the parameters specifying the hardness of f and the pseudorandomness of NW^f).

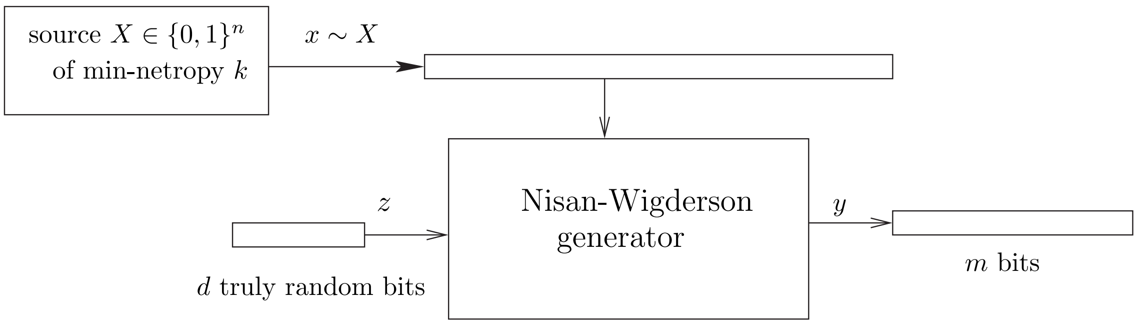
Let us keep from the above analysis that, if the distinguisher exists, then given the distinguisher circuit D and additional

$$\log m + (i^* - 1) + (m - i^*)2^{\alpha} + 2 = O(m^2)$$

bits of information, where i^* is the optimal choice for index i, we can construct a circuit that agrees with f on at least a $\frac{1}{2}+\frac{\epsilon}{m}$ fraction of the inputs. Indeed, $\log m$ bits are needed to specify i^* and another i^*-1 bits to specify the optimal choice for $r_1,\ldots,r_{i^*-1}$. Moreover, for every $j\in\{i^*+1,\ldots,m\}$, we need to give the value of f on at most 2^α points of $\{0,1\}^l$, so that we know $f(z^*|_{S_j})$ for every $x\in\{0,1\}^l$; note that we do not need to know the optimal choice z^* for z. Finally, another 2 bits of information are needed to specify which of $A(x,0), A(x,1), \overline{A(x,0)}, \overline{A(x,1)}$ to output.

Construction of Randomness Extractors

First Attempt: Let us investigate whether a construction of the following form could possess good randomness extraction properties.


source $X\in\{0,1\}^n$ of min-netropy k
$x\sim X$
z
Nisan-Wigderson generator
y
d truly random bits
m bits

In the above construction we can view each sample from $x\sim X$ as a funtion $x:\{0,1\}^l\to\{0,1\}$, where $l=\log n$, and, so, the suggested structure actually implements function NW^x, if we consider a set system $S_1,S_2,\ldots,S_m\subseteq[d]$ with the following properties (δ is some free parameter):

- $m=n^\delta$
- $d=\frac{e^2}{\delta}\log n$
- $|S_i|=l=\log n$, for every i
- $|S_i\cap S_j|\le\alpha=\delta\log n=\log m$, for every $i\neq j$

In order to establish randomness extraction properties for the above construction, one could hope to go by contradiction and, via an argument similar to that in the analysis of the Nisan-Wigderson generator, conclude that source X has sufficiently small description to contradict the fact that the min-entropy of X is at least k. Indeed, let us pursue this proof idea to get some feel of what the real construction should look like. Let us suppose that the above construction is not a (k,ϵ) randomness extractor and, therefore, there exists a statistical test $T:\{0,1\}^m\to\{0,1\}$ such that

$$|\Pr_{x\sim X,z\sim U_d}[T(NW^x(z))=1]-\mathbf{Pr}[T(U_m)=1]|>\epsilon$$

Without loss of generality we can remove the absolute value from the above inequality and, then, conclude that:

$$\mathop{\mathbf{Pr}}_{x\sim X}\left[\mathop{\mathbf{Pr}}_{z\sim U_d}[T(NW^x(z))=1]-\mathbf{Pr}[T(U_m)=1]>\frac{\epsilon}{2}\right]\geq\frac{\epsilon}{2} \tag{2}$$

Recall from the analysis of the Nisan-Wigderson generator that, if, for some $x\in\{0,1\}^n$, it holds that

$$\mathop{\mathbf{Pr}}_{z\sim U_d}[T(NW^x(z))=1]-\mathbf{Pr}[T(U_m)=1]>\frac{\epsilon}{2},$$

then, knowing the distinguishing circuit T and given additional $O(m^2)$ bits of information, we can construct a circuit $D_x:\{0,1\}^{\log n}\to\{0,1\}$, which -as an n-bit string- agrees with x on at least a fraction of $\frac{1}{2}+\frac{\epsilon}{2m}$ positions. By combining this observation with (2), it follows that:

"If an x is drawn from X, then, with probability at least $\frac{\epsilon}{2}$, $O(m^2)$ bits are enough to specify a string which agrees with x on at least a fraction of $\frac{1}{2}+\frac{\epsilon}{2m}$ positions".

Unfortunately, this assertion is not sufficient to contradict the fact that source X has min entropy $\geq k$. Note, however, that if, in the above assertion, instead of "$\frac{1}{2}+\frac{\epsilon}{2m}$" we had "1", i.e. with probability at least $\frac{\epsilon}{2}$ over the choice of x, x could be described completely with $O(m^2)$ bits, then a contradiction would be easy to get as follows:

- there are at most at most $2^{O(m^2)}$ strings that are described exactly by $O(m^2)$ bits;
- now, if $S\subseteq\{0,1\}^n$ is the support of X, then, from the modified assertion, it follows that at least $\frac{\epsilon}{2}2^k$ elements of S can be described by $O(m^2)$ bits (note that since $H_\infty(X)\geq k$ every atom in S has probability at most $\frac{1}{2^k}$)
- therefore, it must hold that:
  $$\frac{\epsilon}{2}2^k<2^{O(m^2)}$$
  which can be violated by an appropriate choice of δ so that $k>\Omega(m^2)+\log\frac{1}{\epsilon}+1$.

This observation suggests that what we need to establish for our construction to work is, essentially, a worst case to average case reduction. In other words, before throwing the samples drawn by the source into the Nisan-Wigderson generator, we should somehow "disguise" them into longer strings so that corrupted versions of these strings with -potentially- some small additional information are enough to recover the samples we started with. To achieve this, we will use *list decodable codes.*

2nd Attempt: Following the above suggestion, let us suppose that $C:\{0,1\}^n\to\{0,1\}^{\overline{n}}$ is a $(L,\frac{\epsilon}{2m})$-list decodable code, i.e. for every $y\in\{0,1\}^{\overline{n}}$:

$$\left|\left\{x\in\{0,1\}^n \middle|\ C(x)\text{ agrees on at least a }\frac{1}{2}+\frac{\epsilon}{2m}\text{ fraction of positions with }y\right\}\right|\leq L.$$

The following choice of parameters is possible, as we have seen in previous lectures:

$$\overline{n}=poly\left(n,\frac{2m}{\epsilon}\right)\text{ and }L=poly\left(\frac{m}{\epsilon}\right).$$

Now, let's investigate what happens if, instead of applying the construction described above to samples $x \sim X$, we apply it to their encodings $C(x)$. Let us consider the function:

$$NWE : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

defined by

$$NWE(x,z) := NW^{C(x)}(z)$$

where the parameters of the construction are chosen as above but n is replaced by $\overline{n}$, i.e.

- $m = \overline{n}^{\delta}$
- $d = \frac{e^2}{\delta} \log \overline{n}$
- $|S_i| = l = \log \overline{n}$, for every i
- $|S_i \cap S_j| \leq \alpha = \delta \log \overline{n} = \log m$, for every $i \neq j$

and $\delta > 0$ is a free parameter to be decided later in the proof.

Claim 2 *The function $NWE(x,z) = NW^{C(x)}(z)$ is a (k,ϵ)-extractor.*

Proof: Suppose, by contradiction, that NWE is not a (k,ϵ) extractor. Then for some source X of min-entropy at least k there exists a statistical test $T : \{0,1\}^m \to \{0,1\}$ such that

$$|\mathop{\mathbf{Pr}}_{x\sim X, z\sim U_d}[T(NWE(x,z)) = 1] - \mathbf{Pr}[T(U_m) = 1]| > \epsilon.$$

As in the analysis of our first attempt, we can remove the absolute value from the above inequality without loss of generality and, then, conclude that

$$\mathop{\mathbf{Pr}}_{x\sim X}\left[\mathop{\mathbf{Pr}}_{z\sim U_d}[T(NWE(x,z)) = 1] - \mathbf{Pr}[T(U_m) = 1] > \frac{\epsilon}{2}\right] \geq \frac{\epsilon}{2} \tag{3}$$

So, if we define a $x \in \{0,1\}^n$ *to be bad* iff

$$\mathop{\mathbf{Pr}}_{z\sim U_d}[T(NWE(x,z)) = 1] - \mathbf{Pr}[T(U_m) = 1] > \frac{\epsilon}{2},$$

it follows that

$$\mathop{\mathbf{Pr}}_{x\sim X}[x \text{ is bad}] \geq \frac{\epsilon}{2}.$$

Also, since X is of min-entropy $\geq k$ it follows that

$$\frac{\#\text{ of bad } x\text{'s}}{2^k} \geq \mathop{\mathbf{Pr}}_{x\sim X}[x \text{ is bad}]$$

We will show that

$$\#\text{ of bad } x\text{'s} \leq 2^{O(m^2 + \log 1/\epsilon)},$$

which will give a contradiction for an appropriate choice of δ that makes $k > \Omega(m^2 + \log 1/\epsilon)$.

Claim 3 *# of bad x's* $\leq 2^{O(m^2+\log 1/\epsilon)}$

PROOF: Let us fix a bad x. Call $\overline{x} = C(x)$. It follows that

$$\frac{\epsilon}{2} < \Pr_{z\sim U_d}[T(NWE(x,z)) = 1] - \mathbf{Pr}[T(U_m) = 1] \equiv \Pr_{z\sim U_d}[T(NW^{\overline{x}}(z)) = 1] - \mathbf{Pr}[T(U_m) = 1]$$

Thus, from exactly the same analysis as the one we did for our first attempt above, it follows that $O(m^2)$ bits are enough to specify a string y which agrees with $\overline{x} = C(x)$ on at least a fraction of $\frac{1}{2} + \frac{\epsilon}{2m}$ positions. Now, we resort to the fact that C is a $(L, \frac{\epsilon}{2m})$-list decodable code. This implies that, given a string y, there are at most L candidate elements of $\{0,1\}^n$ that are mapped to codewords which agree with y in at least a fraction of $\frac{1}{2} + \frac{\epsilon}{2m}$ positions. So with additional $\log L$ bits of information we can specify which element of that list is x. Therefore, every bad x is specified with $O(m^2) + \log L = O(m^2) + O(\log m + \log 1/\epsilon) = O(m^2 + \log 1/\epsilon)$ bits of information. Therefore,

$$\#\text{ of bad } x\text{'s} \leq 2^{O(m^2+\log 1/\epsilon)}$$

□ □

To recapitulate, we constructed (k, ϵ) extractors $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, with parameters d, m that satisfy the following ($\delta > 0$ is some free parameter which we have to choose to fit the following requirements):

- $m = \overline{n}^\delta < n$
- $\overline{n} = poly(n, \frac{2m}{\epsilon})$ is the stretch of the list decodable code that we use
- $d = O(\frac{1}{\delta}\log\overline{n}) = \frac{1}{\delta}O(\log n + \log\frac{1}{\epsilon})$
- $k > \Omega(m^2 + \log 1/\epsilon)$

For example, if we use the concatenation of the Reed-Solomon with the Hadamard code for our encoding, then $\overline{n} = O\left(\left(\frac{n\cdot m}{\epsilon}\right)^2\right) = O\left(\frac{n^4}{\epsilon^2}\right)$ and, so, our parameters satisfy the following requirements (again $\delta > 0$ is a free parameter to play with to make sure that the first inequality holds):

- $m = O\left(\left(\frac{n^2}{\epsilon}\right)^\delta\right) < \min\left\{n, O(\sqrt{k} + \sqrt{\log 1/\epsilon})\right\}$
- $d = \frac{1}{\delta}O\left(\log n + \log\frac{1}{\epsilon}\right)$