CS225: Pseudorandomness Prof. Salil Vadhan

Lecture 10: Randomness Extractors

March 13, 2007

Based on scribe notes by Vitaly Feldman, Andrei Jorza, and Pavlo Pylyavskyy.

Having spent several lectures on expander graphs, the first major pseudorandom object in this course, we now move on to the second: randomness extractors. We begin by discussing the original motivation for extractors, which was to simulate randomized algorithms with sources of biased and correlated bits. This motivation is still compelling, but extractors have taken on a much wider significance in the years since they were introduced. They have found numerous applications in theoretical computer science beyond this initial motivating one, in areas random from cryptography to distributed algorithms to metric embeddings. More importantly from the perspective of this course, they have played a major unifying role in the theory of pseudorandomness. Indeed, the the links between the various pseudorandom objects we will study in this course (expander graphs, randomness extractors, list-decodable codes, pseudorandom generators, samplers) were all discovered through extractors and are still best understood through extractors.

1 Weak Random Sources and Deterministic Extractors

Typically, when we design randomized algorithms or protocols, we assume that all algorithms/parties have access to sources of *perfect* randomness, i.e. bits that are unbiased and completely independent. However, when we implement these algorithms, the physical sources of randomness to which we have access may contain biases and correlations. For example, we may use low-order bits of the system clock, the user's mouse movements, or a noisy diode based on quantum effects. While these sources may have some randomness in them, the assumption that the source is perfect is a strong one, and thus it is of interest to try and relax it.

Ideally, what we would like is a compiler that takes any algorithm A that works correctly when fed perfectly random bits U_m, and produces a new algorithm A' that will work even if it is fed random bits $X \in \{0,1\}^n$ that come from a 'weak' random source. For example, if A is a **BPP** algorithm, then we would like A' to also run in probabilistic polynomial time. One way to design such compilers is to design a *randomness extractor* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ such that $A(X) \equiv U_m$.

Von Neumann Sources. A simple version of this question was already considered by von Neumann. He looked at sources that consist of identical random boolean variables $X_1, X_2, \ldots, X_n \in \{0,1\}$ which are independent but biased. That is, for every i, $\Pr[X_i = 1] = \delta$ for some unknown δ. How can such a source be converted into a source of independent, unbiased bits?

Sources of Independent Bits. Lets now look at a bit more interesting source in which all the variables are still independent but the bias is no longer the same. Specifically, for every i, $\Pr[X_i = 1] = \delta_i$ and $0 < \delta \le \delta_i \le 1 - \delta$. How can we deal with such a source?

Let's be more precise about the problems we are studying. A *source* on $\{0,1\}^n$ is simply a random variable X taking values in $\{0,1\}^n$. In each of the above examples, there is an implicit *class* of sources being studied. For example, $\mathrm{IndBits}_{n,\delta}$ is the class of sources X on $\{0,1\}^n$ where the bits X_i are independent and satisfy $\delta \le \Pr[X_i = 1] \le 1-\delta$. We could define $\mathrm{VN}_{n,\delta}$ to be the same with the further restriction that all of the X_i's are identically distributed, i.e. $\Pr[X_i = 1] = \Pr[X_j = 1]$ for all i, j.

Definition 1 (deterministic extractors) [1] *Let $\mathcal{C}$ be a class of sources on $\{0,1\}^n$. An* ε-extractor *for $\mathcal{C}$ is a function* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ *such that for every $X \in \mathcal{C}$,* $\mathrm{Ext}(X)$ *is 'ε-close' to U_m.*

Note that we want a *single* function Ext that works for all sources in the class. This captures the idea that we do not want to assume we know the exact distribution of the physical source we are using, but only that it comes from some class. For example, for $\mathrm{IndBits}_{n,\delta}$, we know that the bits are independent and none are too biased, but not the specific bias of each bit. Note also that we only allow the extractor *one* sample from the source X. If we want to allow multiple independent samples, then this should be modelled explicitly in our class of sources; ideally we would like to minimize the independence assumptions used.

We still need to define what we mean for the output to be ε-close to U_m.

Definition 2 *For random variables X and Y taking values in U, their* statistical difference *(also known as* variation distance*) is $\Delta(X,Y) = \max_{T\subseteq U} |\Pr[X \in T] - \Pr[Y \in T]|$. We say that X and Y are* ϵ-close *if $\Delta(X,Y) \le \epsilon$.*

The intuitive understanding would be that any event in X happens in Y with same probability $\pm\epsilon$. This is really the most natural measure of distance for probability distributions (much moreso than the ℓ_2 distance we used in the study of random walks). In particular, it satisfies the following natural properties.

1. $0 \le \Delta(X,Y) \le 1$, equality holds for identical distributions and for distributions with disjoint support correspondingly.
2. $\Delta(X,Y)$ is symmetric.
3. $\Delta(X,Z) \le \Delta(X,Y) + \Delta(X,Z)$.
4. for any function f we have $\Delta(f(X), f(Y)) \le \Delta(X,Y)$.
5. $\Delta((X_1,X_2),(Y_1,Y_2)) \le \Delta(X_1,Y_1) + \Delta(X_2,Y_2)$ if X_1 and X_2, as well as Y_1 and Y_2, are independent.
6. $\Delta(X,Y) = \frac{1}{2} \cdot |X - Y|_1$, where $|\cdot|_1$ is the ℓ_1 distance. Thus, X is ε-close to Y iff we can transform X into Y by 'shifting' at most an ε fraction of probability mass.

We now observe that extractors according to this definition give us the 'compilers' we want.

[1] Such extractors are called *deterministic* or *seedless* to contrast with the probabilistic or *seeded* randomness extractors we will see later.

Proposition 3 *Let $A(w;r)$ be a randomized algorithm such that $A(w;U_m)$ has error probability at most γ, and let* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ *be an ε-extractor for a class $\mathcal{C}$ of sources on $\{0,1\}^n$. Define $A'(w;x) = A'(w;\mathrm{Ext}(x))$. Then for every source $X \in \mathcal{C}$, $A'(w;X)$ has error probability at most $\gamma + \varepsilon$.*

From this, we see some additional properties we'd like from our extractors. We'd like the extractor itself to be efficiently computable (e.g. polynomial time). In particular, to get m almost-uniform bits out, we should need at most $n = \mathrm{poly}(m)$ bits bits from the weak random source.

Using our earlier observations (i.e. the parity extractor), we achieve these properties for extracting from sources of independent bits:

Proposition 4 *For every constant $\delta > 0$, every $n, m \in \mathbb{N}$, there is a polynomial-time computable function* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ *that is an ε-extractor for* $\mathrm{IndBits}_{n,\delta}$*, with $\varepsilon = m \cdot 2^{-\Omega(n/m)}$.*

In particular, taking $n = m^2$, we get exponentially small error with a source of polynomial length.

Proof: Ext breaks the source into m blocks of length $\lfloor n/m \rfloor$ and outputs the parity of each block. ■

Santha-Vazirani Sources. Another interesting class of sources, which looks similar to the previous example is the class $\mathrm{SV}_{n,\delta}$ of *Santha-Vazirani sources*. These are the sources that for every i, every $x_1, \ldots, x_n \in \{0,1\}$ and some constant $\delta > 0$, satisfy

$$\delta \leq \Pr\left[X_i = 1 \mid X_1 = x_1, X_2 = x_2, \ldots, X_{i-1} = x_{i-1}\right] \leq 1 - \delta$$

The parity extractor used above will be of no help with this source since the next bit could be chosen in a way that the parity will be equal to 1 with probability δ. It turns out that this can be said about any fixed randomness extraction function.

Proposition 5 (PS 4) *For every $n \in \mathbb{N}$, $\delta > 0$, and fixed extraction function* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}$ *there exists a source $X \in \mathrm{SV}_{n,\delta}$ such that either* $\Pr[\mathrm{Ext}(X) = 1] \leq \delta$ *or* $\Pr[\mathrm{Ext}(X) = 1] \geq 1 - \delta$*. That is, there is no ε-extractor for $\mathrm{SV}_{n,\delta}$ for $\varepsilon < 1/2 - \delta$.*

Nevertheless, as we will see, the answer to the question whether we can simulate **BPP** algorithms with Santha-Vazirani sources will be "yes"! Indeed, we will even be able to handle a much more general class of sources, introduced in the next section.

2 General Weak Sources

2.1 Measures of Entropy

Intuitively, to extract m almost-uniform bits from a source, the source must have at least 'm bits of randomness' in it (e.g. its support cannot be much smaller than 2^m). Ideally, this is all we would like to assume about a source. Thus, we need some measure of how much randomness is in a random variable; this can be done using various notions of *entropy* described below.

- Shanon entropy:

$$\mathrm{H}_{Sh}(X) = \mathop{\mathrm{E}}_{x \stackrel{\mathrm{R}}{\leftarrow} X}\left[\log \frac{1}{\Pr\left[X = x\right]}\right].$$

- Renyi entropy:

$$\mathrm{H}_2(X) = \log\left(\frac{1}{\mathrm{E}_{x \stackrel{\mathrm{R}}{\leftarrow} X}[\Pr\left[X = x\right]]}\right) = \log \frac{1}{\mathrm{CP}(X)}.$$

- Min-entropy:

$$\mathrm{H}_\infty(X) = \min_x \left\{\log \frac{1}{\Pr\left[X = x\right]}\right\}.$$

(All log-s are in base 2.)

2.2 Basic Properties

All the three measures satisfy the following properties:

- $0 \leq \mathrm{H}(X) \leq \log |\mathrm{Supp}(X)|$. Thus, $\mathrm{H}(X) = 0$ if X is constant, and $\mathrm{H}(X) = |\mathrm{Supp}(X)|$ if X is uniform on $\mathrm{Supp}(X)$.
- If X, Y are independent, then $\mathrm{H}((X, Y)) = \mathrm{H}(X) + \mathrm{H}(Y)$.
- For every deterministic function f, we have $\mathrm{H}(f(X)) \leq \mathrm{H}(X)$.
- For every X, we have $\mathrm{H}_\infty(X) \leq \mathrm{H}_2(X) \leq \mathrm{H}_{Sh}(X)$.

To illustrate the differences between the three notions, consider a source X such that $X = 0^n$ with probability 0.99 and $X = U_n$ with probability 0.01. Then $\mathrm{H}_{Sh}(X) \geq 0.01n$ (contribution from the uniform distribution), $\mathrm{H}_2(X) \leq \log\left(\frac{1}{0.99^2}\right) < 1$ and $\mathrm{H}_\infty(X) \leq \log\left(\frac{1}{0.99}\right) < 1$ (contributions from the constant distribution with probability 0.99). Note that even though X has relatively high Shannon entropy, we cannot expect to extract bits that are close to uniform or carry out any useful randomized computations with one sample from X, because it gives us nothing useful 99% of the time. Thus, we should use the stronger measures of entropy given by H or H_∞.

Then why is Shannon entropy so widely used in information theory results? The reason is that such results typically study what happens when you have many independent samples from the source. In such a case, it turns out that the the source is "close" to one where the min-entropy is roughly equal to the Shannon entropy. Thus the distinction between these entropy measures becomes less significant. (Recall that we only allow one sample from the source.) Moreover, Shannon entropy satisfies many nice identities that make it quite easy to work with. Min-entropy and Renyi entropy are much more delicate.

2.3 k-sources

Definition 6 *X is a k-source is* $\mathrm{H}_\infty(X) \geq k$*, i.e., if* $\Pr[X = x] \leq 2^{-k}$.

A typical setting of parameters is $k = \delta n$ for some fixed δ, e.g., 0.01. We call δ the *min-entropy rate*. Some different ranges that are commonly studied (and are useful for different applications): $k = \mathrm{polylog}(n)$, $k = n^\gamma$ for a constant $\gamma \in (0,1)$, $k = \delta n$ for a constant $\delta \in (0,1)$, and $k = n - O(1)$. The middle two ($k = n^\gamma$ and $k = \delta n$) are the most natural for simulating randomized algorithms with weak random sources.

Examples of k-sources:

- k random and independent bits, together with $n-k$ fixed bits. These are called *oblivious bit-fixing* sources.
- k random and independent bits, and $n-k$ bits that depend arbitrarily on the first k bits. These are called *adaptive bit-fixing sources.*
- Santha-Vazirani δ sources. Take $k = \log \frac{1}{(1-\delta)^n} = \Theta(\delta n)$.
- Uniform distribution on $S \subset \{0,1\}^n$ with $|S| = 2^k$. These are called *flat k-sources.*

It turns out that flat k-sources are really representative of general k-sources.

Proposition 7 *Every k-source is a convex combination of flat k-sources (provided that $2^k \in \mathbb{N}$), i.e., $X = \sum p_i X_i$ with $0 \leq p_i \leq 1$, $\sum p_i = 1$ and all the X_i are flat k-sources.*

Proof Sketch: Consider each source on $[N]$ (recall that $N = 2^n$) as a vector $X \in \mathbb{R}^N$. Then X is a k-source if and only if $\forall i$ one has $X(i) \in [0,1]$ so that $\sum X(i) = 1$ (condition for probabilities) and $\forall i$ one has $X(i) \leq 2^{-k}$ (condition for k-source).

The set of all k-sources is a polytope determined by all these vectors, since all these conditions are linear. More precisely, the set of k-sources is the intersection of the hypercube $[0, 2^{-k}]^N$ and the hyperplane $\sum X(i) = 1$. This is a convex polytope and so any k-source is a convex combination of the vertices of the polytope. The vertices of the polytope are the points that make a maximal subset of the inequalities tight. Since $\sum X(i) = 1$, these sources are precisely those where $X(i) = 2^{-k}$ for 2^k values of i and $X(i) = 0$ for the remaining values of i. Therefore the vertices are represented by the flat k-sources. □

Thus, we can think of any k-source as being obtained by first selecting a flat k-source X_i according to some distribution (given by the p_i's) and then selecting a random sample from X_i. This means that if we can compile probabilistic algorithms to work with flat k-sources, then we can compile them to work with any k-source.

3 Seeded Extractors

Proposition 5 tell us that it impossible to have deterministic extractors for Santha-Vazirani sources. Here we consider k-sources, which are more general than Santha-Vazirani sources, and hence also impossible to have deterministic extractors. The impossibility result for k-sources is stronger and simpler to prove.

Proposition 8 *For any* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}$ *there exists an* $(n-1)$*-source* X *so that* $\mathrm{Ext}(X)$ *is constant.*

Proof: There exists $b \in \{0,1\}$ so that $|\mathrm{Ext}^{-1}(b)| \geq 2^n/2 = 2^{n-1}$. Then let X be the uniform distribution on $\mathrm{Ext}^{-1}(b)$. ■

On the other hand, if we reverse the order of quantifiers, allowing the extractor to depend on the source, it is easy to see that good extractors exist and in fact a randomly chosen function will be a good extractor with high probability.

Proposition 9 *For every* $n, k, m \in \mathbb{N}$*, every* $\varepsilon > 0$*, and every flat* k*-source* X*, if we choose a random function* $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ *with* $m = k - 2\log(1/\varepsilon) - O(1)$*, then* $\mathrm{Ext}(X)$ *will be* ε*-close to* U_m *with probability* $1 - 2^{-\Omega(K\varepsilon^2)}$*, where* $K = 2^k$*.*

(We will commonly use the convention that capital variables are 2 raised to the power of the corresponding lowercase variable, such as $K = 2^k$ above.)

Proof: Choose our extractor randomly. We want it to have following property: for all $T \subseteq [M]$, $|\Pr[\mathrm{Ext}(X) \subseteq T] - \Pr[U_m \subseteq T| \leq \epsilon$. This can be reformulated as $\frac{|\{x \in X | \mathrm{Ext}(x) \in T\}|}{K}$ differs from density $\mu(T)$ on not more than ϵ. For each point $x \in \mathrm{Supp}(X)$ the chance to be in T is $\mu(T)$. Chernoff bound says in this case that for each fixed T this condition holds with high probability, that is at least $1 - 2^{-\Omega(K\epsilon^2)}$. Then the probability that condition is violated for at least one T is at most $2^M 2^{-\Omega(K\epsilon^2)}$, which is less than 1 for $m = k - 2\log(\frac{1}{\epsilon}) - O(1)$. ■

Note that the failure probability is doubly-exponentially small in k. Naively, one might hope that we could get an extractor that's good for all flat k-sources by a union bound. But the number of flat k-sources is $\binom{N}{K} \approx N^K$ (where $n = 2^n$), which is unfortunately a larger double-exponential in k. We can overcome this gap by allowing the extractor to be 'slightly' probabilistic, i.e. allowing the extractor a *seed* consisting of a small number of truly random bits in addition to the weak random source. We can think of this seed of truly random bits as random choice of an extractor from family of extractors. This leads to the following crucial definition:

Definition 10 (seeded extractors) *Extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is a* (k,ϵ)-extractor *if for any* k*-source* X *on* $\{0,1\}^n$*,* $\mathrm{Ext}(X, U_d)$ *is* ϵ*-close to* U_m*.*

We want to give a construction which minimizes d and maximizes m. We prove the following theorem.

Theorem 11 *For any n and k ($k \leq n$) and any $\epsilon > 0$ there exists a (k, ϵ) - extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with $m = k + d - 2\log(\frac{1}{\epsilon}) - O(1)$ and $d = \log(n-k) + 2\log(\frac{1}{\epsilon}) + O(1)$.*

One setting of parameters to keep in mind (for our application of simulating randomized algorithms with a weak source) is $k = \delta n$, with δ a fixed constant (e.g. $\delta = 0.01$), and ϵ a fixed constant (e.g. $\epsilon = 0.01$).

Proof: We use probabilistic method to prove the theorem. It suffices for Ext to work for flat k-sources. Choose extractor Ext at random. Then the probability that extractor fails is not more than number of flat k-sources times times the probability Ext fails for fixed flat k-source. By the above proposition, the probability of failure for a fixed flat k-source is at most $2^{-\Omega(KD\epsilon^2)}$, since (X, U_d) is a flat $(k+d)$-source) and $m = k + d - 2\log(\frac{1}{\epsilon}) - O(1)$. Thus the total failure probability is at most

$$\binom{N}{K} \cdot 2^{-\Omega(KD\epsilon^2)} \leq \left(\frac{Ne}{K}\right)^K 2^{-\Omega(KD\epsilon^2)}.$$

The letter expression is less than 1 if $D\epsilon^2 \geq 2\log\frac{Ne}{K} = c(n-k) + c'$ for constants c, c'.This is equivalent to $d = \log(n-k) + 2\log(\frac{1}{\epsilon}) + O(1)$. ■

It turns out that both bounds (on m and d) are individually tight upto the $O(1)$ terms.

4 Simulating Randomized Algorithms

Now we study simulating randomized algorithm having weak random source. Usual randomized algorithm takes input string w and m random bits, and outputs the correct answer with probability at least $1 - \gamma$. Assume now we do not have a source of perfectly random bits. Instead we have a k-source and an extractor, which takes an input from our weak source. We also allow it to take small seed of purely random bits, which as mentioned above, can be viewed as choosing a random extractor from some family. The output of the extractor we feed into our randomized algorithm A instead of purely random bits it took before. Since above we had seed having logarithmic size, we can actually eliminate it just by running through all possible values it can take and ruling my majority vote.

Proposition 12 *Let $A(w; r)$ be a randomized algorithm such that $A(w; U_m)$ has error probability at most δ, and let* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *be a (k, ε)-extractor. Define*

$$A'(w; x) = A'(w, x) = \operatorname*{maj}_{y \in \{0,1\}^d} \{A(w, \mathrm{Ext}(x, y))\}.$$

Then for every k-source X on $\{0,1\}^n$, $A'(w; X)$ has error probability at most $2(\gamma + \varepsilon)$.

Proof: The probability that $A(w, \mathrm{Ext}(X, U_d))$ is incorrect is not more than probability $A(w, U_m)$ is incorrect plus ϵ, $\gamma + \epsilon$ in particular, according to the defining property of statistical difference. Then the probability that $\mathrm{maj}_y\, A(w, \mathrm{Ext}(X, y))$ is incorrect is at most $2(\gamma + \epsilon)$. ■

Note that the running time slowdown is 2^d times the running time of Ext. Thus, we want to construct extractors achieving the following three properties: $d = O(\log n)$; Ext computable in polynomial time; $m = k^{\Omega(1)}$.

The bound on the error probability in Proposition 12 can actually be made exponentially small (say 2^{-t}) by using an extractor that is designed for min-entropy roughly $k - t$ instead of k.

We note that even though seeded extractors suffice for simulating randomized algorithms with only a weak source, they do not suffice for all applications of randomness in theoretical computer science. The trick of eliminating the random seed by enumeration does not work, for example, in cryptographic applications of randomness. Thus the study of deterministic extractors for restricted classes of sources remains a very interesting an active research direction. We, however, will focus on seeded extractors, due to their many applications and their connections to the other pseudorandom objects we are studying.