6.895 Essential Coding Theory November 17, 2004

Lecture 20

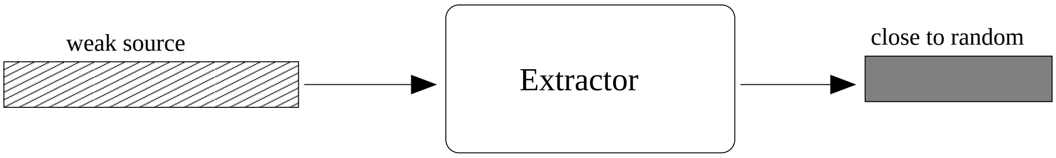
Lecturer: Madhu Sudan *Scribe: Anup Rao*

1 Overview

- Extractors
- Extractors vs Codes

2 Extractors

Suppose we have access to a source of randomness that isn't completely random, but does contain a lot of randomness. Is there a way to convert this source into a source of truly random bits? Ideally we'd like an algorithm (called an Extractor) that does something like this:



To formalize this, we need to decide what we mean by a *weak* source and what it means to say that a distribution is *close* to being random.

Definition 1 *Let D_1 and D_2 be two distributions on a set S. Their* statistical distance *is*

$$\| D_1 - D_2 \| \stackrel{\text{def}}{=} \max_{T \subseteq S}(|D_1(T) - D_2(T)|) = \frac{1}{2}\sum_{s \in S} |D_1(s) - D_2(s)|$$

If $\|D_1 - D_2\| \leq \epsilon$ we shall say that D_1 is ϵ-close to D_2.

Note that this definition of *closeness* is as strong as we might hope for. Specifically, if D_1 and D_2 are ϵ-close, then for any algorithm A, $\| \Pr_{x \leftarrow_{\mathrm{R}} D_1}[A(x) = 1] - \Pr_{x \leftarrow_{\mathrm{R}} D_2}[A(x) = 1]\| \leq \epsilon$.

Different models have been considered to formalize the notion of *weak* sources. von-Neumann considered sources where each bit from the source is independent, but biased, i.e. the probability of the bit being 1 is p, where p may not be $\frac{1}{2}$. Santha and Vazirani [SV84] considered sources where each bit is biased and the bias of the bit depends on the bits that have come before it. The most general notion of a *weak* source to date was first introduced by Nisan and Zuckerman [NZ93]. Here, we bound the *min-entropy* of the source:

Definition 2 *The* min-entropy *of a source D is defined to be:*

$$H_\infty(D) \stackrel{\text{def}}{=} -\log(\max_{x \in D}(D(x))$$

We will call a distribution over $\{0,1\}^n$ that has min-entropy k an (n,k) source.

So if a source has min-entropy k, the probability that the source gives a particular string is at most 2^{-k}. This definition is fairly general. If a source has min-entropy k, then we expect that we will have to take more than $2^{k/2}$ samples from the source before we see a repetition. Such a source must also have a support that contains more than 2^k elements.

Fact 3 *Every (n,k) source is a convex combination of sources which are uniform distributions on sets of size 2^k.*

Our extractor problem is now well defined. We'd like to construct an efficiently computable deterministic function EXT that takes n bits of input from any source of min-entropy k and produces m bits that are statistically close to being uniformly random. Unfortunately such a function does not exist! To see this, suppose EXT was an extractor that could extract just one bit from a source of min-entropy $n-1$. At least half the points (2^{n-1} points) in $\{0,1\}^n$ must get mapped to either 0 or 1 by the extractor, so if we choose our input weak source to be the flat distribution on these points, we see that our source has min-entropy $n-1$, but the output of the extractor is statistically far from being uniformly random.

Notice that in the argument above there was a strong relation between the source that we picked as a counterexample to the extractor and the extractor itself. This suggests a relaxation to the problem: pick a function randomly from a family and use that as an extractor.

Definition 4 *We say a function $EXT : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ) extractor if for any (n,k) source X and for an Y chosen independently uniformly at random from $\{0,1\}^d$, we have*

$$\| EXT(X,Y) - U_m \| \leq \epsilon$$

where U_m is the uniform distribution on m bits.

Definition 5 *We say a function $EXT : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is a* strong *(k,ϵ) extractor if for any (n,k) source X and for Y chosen uniformly at random from $\{0,1\}^t$, we have*

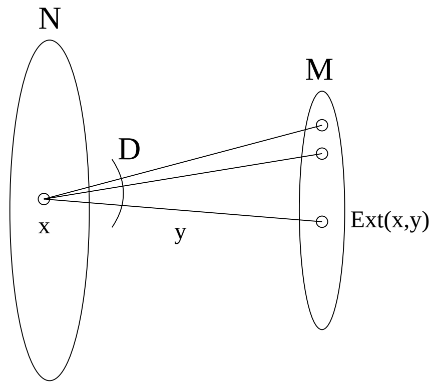
$$\| Y \circ EXT(X,Y) - Y \circ U_m \| \leq \epsilon$$

A strong extractor ensures that even when given the seed used, the output of the extractor is statistically close to being uniformly random.

Fact 3 implies that if our extractor can extract from any flat distribution with min-entropy k, then it can also extract from a general source that has min-entropy k.

2.1 Graph theoretic view of Extractors

Let $N = 2^n$, $M = 2^m$, $D = 2^d$, $K = 2^k$. Then we can view an extractor as a bipartite graph with N D-regular vertices on the left and M vertices on the right.



In terms of the extractor graph the extractor property translates to the following. If S is a subset of the vertices on the left with $|S| \geq K$ and T is any subset of the vertices on the right, the number of edges going from S to T is close to what we'd expect if the graph was chosen randomly, i.e.,

$$\left| \#\text{ of edges going from } S \text{ to } T - D|S||T|/M \right| \leq \epsilon D|S|$$

This property looks very similar to something we've seen before. Recall the definition of (D,ϵ)-uniform graphs that we saw in lecture 18:

Definition 6 *Let G be a D-regular, bipartite graph with a set L of left vertices and a set R of right vertices satisfying $|L| = |R| = N$. G is a (D, ϵ)-$uniform$ graph if $\forall X \subseteq L, Y \subseteq R, \beta = \frac{|X|}{N}, \gamma = \frac{|Y|}{N}$, the number of edges between X and Y is $\geq (\beta \cdot \gamma - \epsilon) \cdot D \cdot N$.*

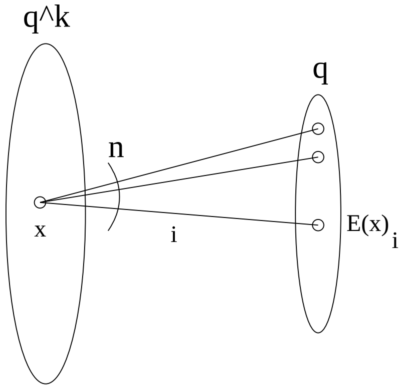
The key differences to note between the two properties are that:

- An extractor graph may be imbalanced, while a uniform graph must be balanced.
- An extractor graph is required to satisfy the *uniformness* property only with regards to large (cardinality $\geq K$) sets on the left, whereas a uniform graph allows a more general choice of sets.
- The uniformness condition for an extractor graph is somewhat stronger in the sense that the number of edges going from a particular set S on the left to a set T on the right is allowed to deviate by only $\epsilon D|S|$ from the expected number for a random graph. On the other hand in a uniform graph the number of edges is allowed to deviate by ϵDN from the expected number.

3 Extractors vs Codes

We have already seen some ways to use bipartite graphs to construct codes. Now we will see a new way to interpret a code as a bipartite graph that will allow us to get a connection between extractors and codes. Given any $[n, k, ?]_q$ code with an encoding function E, we construct the bipartite graph (L, R, G), where L is the set of left vertices, R is the set of right vertices and G is the set of edges, so that:

- $L \stackrel{\text{def}}{=} [\mathbb{F}_q]^k$
- $R \stackrel{\text{def}}{=} [\mathbb{F}_q]$
- $G \stackrel{\text{def}}{=} \{(x, y) | \exists i \text{ s.t. } E(x)_i = y\}$, i.e., for any element $x \in L$, x is connected to the n elements $E(x)_1, E(x)_2, ..., E(x)_n$ in R.



Ta-Shma and Zuckerman [TSZ01] observed that this graph is a good extractor exactly when the corresponding code is a good list decodable code! In fact, when the graph is a good extractor, the corresponding code has a property that is slightly stronger than just being list decodable, it is *soft decision decodable*. To motivate this concept, let us review what it means to decode words in the classical sense.

In the classical decoding problem, we start with a received vector $r \in \mathbb{F}_q^n$ and we try to find the closest (in terms of Hamming distance) codeword to this received word. However, in the real world we usually have a little more information than just a received word. For instance we can imagine using some electrical device to transmit information over a noisy channel, where a +1 volt signal means a 1 and a -1 volt signal means a 0. If we receive a signal of $+.99$ volts, it was probably transmitted as a 1. On the other hand, if we receive 0.10 volts, then it is less likely that the transmitted signal was a 1. Soft decision decoding is one way to take advantage of this extra information.

In soft decision decoding, instead of a received word, we are given a weight function $w : [n] \times \mathbb{F}_q \to [0, 1]$ and the goal is to find a codeword C that maximizes $\sum_{i=1}^{n} w(i, C_i)$. So for instance if we are very sure that the first symbol of the codeword is a 0, we can factor that in by making $w(1, 0)$ very high. It is easy to see that classical decoding is just a special case of soft decision decoding. For any weight function w, we define the relative weight of the function as:

$$\rho(w) \stackrel{\text{def}}{=} \frac{\sum_{i \in [n], y \in \mathbb{F}_q} w(i, y)}{nq}$$

Note that the expected weight of a random word is exactly $\rho(w)n$.

As usual, we can now look at the corresponding list decoding problem:

Definition 7 *A code C is said to be* soft list decodable *from $(1-\epsilon)$ fraction errors if: $\forall$ weight functions w, $\exists$ at most* $\mathrm{poly}(n)$ *codewords in C that have weight $\geq n(\rho(w) + \epsilon)$.*

Ta-Shma and Zuckerman showed that good extractors correspond to codes which are soft list decodable. Now we will sketch the arguments for the connection.

Theorem 8 *$C = (L, R, G)$ is the graph of a code that is soft list decodable from $(1 - \epsilon)$ fraction errors with list size less than $K \Rightarrow C$ is the graph of a strong $(k - \log(\epsilon), 2\epsilon)$ extractor.*

Sketch of Proof

For simplicity, we will just prove that the graph obtained is the graph of a $(k - \log(\epsilon), 2\epsilon)$ extractor. To show that it's a strong extractor we need to make some minor modifications to the same proof.

Suppose the graph is not that of a good extractor. Then $\exists S \subset L, T \subset R$, with $|S| \geq K/\epsilon$, so that the number of edges between S and T is $2\epsilon n|S|$ more (or less) than the expected number of edges. wlog we will assume that the number of edges is more than the expected number (if it's less than the expected number we can take the complement of T instead). Now consider the following weight function

$$w(i, y) = \begin{cases} 0 & y \notin T \\ 1/|T| & y \in T \end{cases}$$

Note that $\rho(w) = 1/q$. The average weight of a codeword in S is at least $2\epsilon n$ more than the expected weight of a random word. On the other hand each word in S has a weight of at most $n/|T|$. If there are x words in S which have a weight greater than $(1/q + \epsilon)n$, we have that

$$\begin{aligned} xn/|T| + (|S| - x)(1/q + \epsilon)n &\geq (1/q + 2\epsilon)n|S| \\ \Rightarrow x(1/|T| - 1/q - \epsilon) &\geq \epsilon|S| \\ \Rightarrow x &\geq \epsilon|S| \\ \Rightarrow x &\geq K \end{aligned}$$

This contradicts the soft list decodability of the code.

■

Using very similar arguments, we can argue the other direction, i.e., that an extractor graph gives a soft list decodable code.

Theorem 9 *C is the graph of a strong (k, ϵ) extractor $\Rightarrow$ $C = (L, R, G)$ is the graph of a code that is soft list decodable from $(1 - \epsilon)$ fraction errors with list size less than K.*

Sketch of Proof Suppose not. Then there is a weight function w for which the list size is larger than K. Consider the behavior of the extractor when it is given the uniform distribution over this list as the input distribution. The weight function w then gives a way to distinguish the output of the extractor from the uniform distribution with probability ϵ, contradicting the extractor property. ■

References

[NZ93] Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 235–244, San Diego, California, 16–18 May 1993.

[SV84] Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from slightly-random sources (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 434–440, Singer Island, Florida, 24–26 October 1984. IEEE.

[TSZ01] Amnon Ta-Shma and David Zuckerman. Extractor codes. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: Hersonissos, Crete, Greece, July 6–8, 2001*, pages 193–199, New York, NY, USA, 2001. ACM Press.