6.842 Randomness and Computation April 23, 2014

Lecture 20

Lecturer: Ronitt Rubinfeld *Scribe: Aldo Pacchiano and Luis Voloch*

1 Pseudorandomness

The goal of this lecture is to explore the possibility of constructing a scheme whereby a string of m random bits could be converted into a string of n bits, with $n >> m$, such that the new string looks random. The goal is to convert a small source of randomness into a larger stream of "random looking" bits. We call this process pseudorandom number generation (PSG):

$$PSG\,(n \text{ random bits}) = m \text{ random looking bits.}$$

In this lecture we will try to elucidate what we mean by "random looking bits". There are several candidate notions for the defining the idea of approximately random. We will first go through some of them and understand why they are not proper definitions, and then we will settle for the definition that we call *computational indistinguishability.*

1.1 Candidate Notions:

(a) **Kolmogorov Complexity:** We could try to quantify the randomness of a string via its Kolmogorov complexity. Where a string x is seen as a pair y, M, y being a string and M being a turing machine such that $M(y) = x$. The Kolmogorov complexity of x is the minimum length of its description y, M.

(b) **Statistical Distance:** We could also try to quantify the randomness of the resulting string via its statistical distance to uniform. This is the L_1 distance between the resulting distribution over the strings of the output and the uniform distribution. Unfortunately, because a Pseudo Random Generator is a deterministic machine, the distribution over the output is supported in at most 2^n strings, therefore having a statistical distance to the uniform distribution at least as large as $1/2^m * (2^m - 2^n) = 1 - 1/2^{m-n}$.

(c) k**-wise Independence:** We could also try to enforce that the output of the PSG be useful for randomized algorithms, for example, producing strings that are pairwise independence, or k wise independence. The drawback of the last approach is that these types of distributions do not work for all randomized algorithms.

2 Computational indistinguishability

In order to fix the problems described by the above approaches, we will require that the PSG produces a distribution over m bit strings that "looks random" for all probabilistic polynomial time algorithms.

In synthesis, the notion of PSG is with respect to a class of algorithms. For instance, it could be that the PSG produces random looking strings with respect to probabilistic polynomial time algorithms or with respect to constant size circuits.

When we say the resulting strings look random we would like to say they cannot be distinguished from the uniform distribution by any reasonable computational scheme.

Definition 1 (Computational indistinguishability (C.I.)). *Let X_n, Y_n be sequences of random variables on $\{0,1\}^n$ (the exponent n may be substituted by $p(n)$ for any fixed polynomial p). We say $\{X_n\}, \{Y_n\}$ are $\epsilon(n)$ indistinguishable for time $t(n)$ machines (similarly for circuit size $t(n)$) if for all probabilistic polynomial time T Turing machines running in time $t(n)$ it follows that:*

$$|\mathbb{P}(X_n = 1) - \mathbb{P}(Y_n = 1)]| \leq \epsilon(n)$$

for all n large enough.

Note: If we were to hardware the inputs to our circuit T, then this becomes a statistical test for $\{X_n\}$ and $\{Y_n\}$. As expected, a distribution that is computationally indistinguishable to the uniform distribution can be used as a randomness souurce for BPP and RP algorithms:

Lemma 2. *(C.I., BPP, RP) If X_n and Y_n are C.I. (where Y_n is the uniform distribution) for size $t(n)$ circuits then we can use X_n and Y_n interchangeably for all RP and BPP algorithms A.*

Proof. The BPP condition on A implies that: $\forall x \in \{0,1\}^n$ we have that $\mathbb{P}_{r\in Y_{r(n)}}[A(x,r)=1] \geq \frac{2}{3}$ or $\leq \frac{1}{3}$. By amplification we can assume A is as follows: $\forall x \in \{0,1\}^n$ we have that $\mathbb{P}_{r\in Y_{r(n)}}[A(x,r)=1] \geq \frac{3}{4}$ or $\leq \frac{1}{4}$.

Now we aim to show that for all input x we have that $\mathbb{P}_{r\in X_{r(n)}}[A(x,r)=1] \geq \frac{2}{3}$ or $\leq \frac{1}{3}$. We will do so by contradiction. If the last was not true, then, there would be a family of inputs $a_n \in \{0,1\}^n$ such that for infinitely many n:

$$|\mathbb{P}_{r\in X_{r(n)}}[A(a_n,r)=1] - \mathbb{P}_{r\in Y_{r(n)}}[A(a_n,r)=1]| \geq \frac{1}{12}$$

By hardcoding the inputs a_n into the existing circuit we can construct a circuit that serves as a statistical test capable of distinguishing the two distributions X_n and Y_n. Hence, this violates the computational indistinguishability condition of X_n and Y_n. Notice that the same proof would not necessarily work for the Turing machine model. In this model, we cannot hard code the family of inputs $\{a_n\}$. $\square$

3 Pseudorandomness

In what follows we will assume the computational indistinguishability conventions:

- When $\epsilon(n)$ is not specified, we will assume that $\epsilon(n) = \frac{1}{t(n)}$ (likewise for $t(n)$),
- $X_n \overset{c}{\equiv} Y_n$ is the notation for C.I. if $\epsilon(n) = \frac{1}{n^c}$, for $t(n) = n^c$ and for all c. Equivalently, for every polynomial time $T, \exists \epsilon(n) = n^{-\omega(1)}$ such that $|\mathbb{P}[T(X_n)=1] - \mathbb{P}[T(Y_n)=1]| \leq \epsilon(n)$, and
- X_n, Y_n C.I. in non-uniform model time $t(n)$ if it also holds when given $\leq t(n)$ advice bits.

Definition 3 (Negligible Distinguishability)**.** *We call the distinguishability $\epsilon(n)$ negligible if for all c we have that $\epsilon(n) \leq O(\frac{1}{n^c})$.*

Definition 4 (Pseudorandom)**.** *We say that X_n is pseudorandom if $X_n \overset{c}{\equiv} U_n$, where U_n is the uniform distribution.*

Theorem 5. *Let X_n and Y_n be NCI sequence of random variables, and let X_n^k (Y_n^k) be k independent copies of X_n (Y_n). If k is poly(n), then X_n^k, Y_n^k are NCI.*

Proof. Define $H_i = X_n^{k-i} Y_n^i$. This means $H_0 = X_n^k$ and $H_k = Y_n^k$. We now use the following hybrid construction:

$$\begin{aligned}
H_0 &= & X_n X_n \cdots X_n X_n \\
H_1 &= & X_n X_n \cdots X_n Y_n \\
&. \\
&. \\
&. \\
H_k &= & Y_n Y_n \cdots Y_n Y_n
\end{aligned}$$

Assume for the sake of contradiction that $|\mathbb{P}[T(X_n^k) = 1] - \mathbb{P}[T(Y_n^k) = 1]| > \epsilon$. Now notice, by telescoping, that

$$|\mathbb{P}[T(X_n^k) = 1] - \mathbb{P}[T(Y_n^k) = 1]| = |\sum_{i=1}^{k} \mathbb{P}[T(H_{i-1}) = 1] - \mathbb{P}[T(H_i) = 1]|.$$

Therefore, there exists an i_0 such that $|\mathbb{P}[T(H_{i_0-1}) = 1] - \mathbb{P}[T(H_{i_0}) = 1]| > \frac{\epsilon}{k}$ Notice that $T(H_{i_0-1}) = T(X_n^{k-i_0} X_n Y_n^{i_0-1})$ and $T(H_{i_0}) = T(X_n^{k-i_0} Y_n Y_n^{i_0-1})$. After averaging, this implies there is an assignment of the first $k-i_0$ variables, $x_n^1 \cdots x_n^{k-i_0}$ and the last i_0-1 variables, $y_n^1, \cdots, y_n^{i_0-1}$, such that the difference $|\mathbb{P}(x_n^1 \cdots x_n^{k-i_0} X_n y_n^1, \cdots, y_n^{i_0-1}) = 1] - \mathbb{P}[T(x_n^1 \cdots x_n^{k-i_0} Y_n y_n^1, \cdots, y_n^{i_0-1}) = 1]| > \frac{\epsilon}{k}$.

Define now a new circuit $T'(Z)$, which has hardwired into T the assignments $x_n^1 \cdots x_n^{k-i_0}$ and $y_n^1, \cdots, y_n^{i_0-1}$ and such that $T'(Z) = T(x_n^1 \cdots x_n^{k-i_0} Z y_n^1, \cdots, y_n^{i_0-1})$. Notice that the size of the circuit for T' will be comparable to the size of the circuit for T. Furthermore, notice that $|\mathbb{P}[T'(X_n) = 1] - \mathbb{P}[T'(Y_n) = 1]| \geq \frac{\epsilon}{k}$. Since k is poly(n), this contradicts the assumption that X_n and Y_n are N.C.I. $\square$

Note: This proof requires the circuit model of computation because we really need the sequences x_n^i and y_n^i to be hardwired into the circuit. In case of using the Turing machine model, one can get the same result if we can sample from X_n and Y_n in polynomial time.