CS 710: Complexity Theory 11/8/2011

Lecture 18: Pseudorandomness

Instructor: Dieter van Melkebeek Scribe: Sachin Ravi

In the last lecture, we introduced the notion of a pseudorandom generator (PRG), discussed how a PRG can be used to reduce the amount of randomness required to run a randomized algorithm, and outlined the construction of a PRG that derandomized space-bounded randomized computation. In this lecture, we complete our discussion of Space-Bounded Derandomization and start considering Time-Bounded Derandomization by exploring the connection between the two notions of pseudorandomness.

1 Space-Bounded Derandomization (Continued)

Let us first recap the defintion of a pseudorandom generator, specifically for a class of decision problems.

Definition 1. *An ϵ-PRG for a class $\mathcal{A}$ of decision problems is a family $(G_r)_r$ of functions where $G_r : \{0,1\}^{l(r)} \longrightarrow \{0,1\}^r$ such that for all $A \in \mathcal{A}$:*

$$(\forall^\infty)\ |\Pr_{\rho \leftarrow U_r}[A(x,\rho)=1] - \Pr_{\sigma \leftarrow U_{l(r)}}[A(x, G_r(\sigma))=1]| < \epsilon$$

The important parameters in the above definition are:

- Error ϵ: the deviation of the original randomized algorithm from the pseudorandomized algorithm.
- Seed-Length $l(r)$: the number of random bits required as input to the pseudorandom generator.
- Complexity: measured in terms of output string r.

Essentially, the defining property of PRGs is that no algorithm in the class can distingish the pseudorandom distribution from the uniform distribution. This was what we called the indistinguishability requirement for the psedUorandom distribution.

Now, we prove the key lemma that we used in the last lecture to prove the properties of the PRG constructed for space-bounded derandomization. This construction implied that BPSPACE(s) $\subseteq$ DSPACE(s^2). We were already aware of this result but now our proof involved the use of PRGs. In fact, using a slight variation of this constructed PRG, it can be shown that BSPACE(s) $\subseteq$ DSPACE($s^{1.5}$), highlighting the importance of PRGs.

The key lemma in this previous construction was:

Lemma 1. *For any distribution S_{in} on s bits where λ is the second largest eigenvalue of the expander,*

$$|S_{out}(S_{in}, U_{2r'}) - S_{out}(S_{in}, G_{2r'}(U_{r'}, U_{\log d}))|_1 \leq 2^s \cdot \lambda$$

Proof. First notice that we can prove the lemma by only considering point distributions for S_{in}. Point distributions place all probability in a singles point. Any distribution S_{in} is a convex combination of point distributions $X_1, X_2, ..., X_3$ such that $S_{in} = \sum_k p_k X_k$. For a randomized process Z, if we consider the quantity

$$\Delta(S_{in}) = \|Z(S_{in}, U_{2r'}) - Z(S_{in}, G_r(U_{r'}, U_{\log d}))\|_1 ,$$

we know by the triangle inequality for 1-norm that $\Delta(S_{in}) \le \sum_k p_k \Delta(X_k)$. So, we only need to consider a point distribution s_{in}. We defined S_{in} and S_{out} as the distribution inputs to and outputs from, respectively, a pair of blocks. We now define S_{mid} as distribution of states when passing from the first block of a pair to the second block.

$$\Pr[s_{in} \to s_{out}] = \sum_{s_{mid}} \Pr[s_{in} \to s_{mid} \wedge s_{mid} \to s_{out}] \tag{1}$$

$$= \sum_{s_{mid}} Pr[(\rho_{\text{left}}, \rho_{\text{right}}) \in S \times T] \tag{2}$$

$$\tag{3}$$

where S and T are defined as:

$$S = S_{s_{in}, s_{mid}} = \{\rho_{\text{left}} | s_{in} \to s_{mid}\} \tag{4}$$

$$T = T_{s_{mid}, s_{out}} = \{\rho_{\text{right}} | s_{mid} \to s_{out}\}, \tag{5}$$

meaning that S is the set of all random strings that will cause us to proceed from fixed s_{in} to fixed s_{mid} and T is the set of all random strings that will cause use to proceed from fixed s_{mid} to fixed s_{out}.

For the random distribution, (2) can be written as:

$$\Pr_{(\rho_{\text{left}}, \rho_{\text{right}}) \leftarrow U_{2r'}} [s_{in} \to s_{out}] = \sum_{s_{mid}} \mu(S)\mu(T),$$

since ρ_{left} and ρ_{right} are chosen independently at random. Similarly, for the pseudorandom distribution (2) is written as:

$$\Pr_{(\rho_{\text{left}}, \rho_{\text{right}}) \leftarrow G_{2r'}(U_{r'}, U_{log\ d})} [s_{in} \to s_{out}] = \sum_{s_{mid}} \frac{|E(S,T)|}{dN}$$

as we are calculating the probability of picking an edge between S and T.

Thus using the expander mixing lemma:

$$\left| \Pr_{(\rho_{\text{left}}, \rho_{\text{right}}) \leftarrow U_{2r'}} [s_{in} \to s_{out}] - \Pr_{(\rho_{\text{left}}, \rho_{\text{right}}) \leftarrow G_{2r'}(U_{r'}, U_{log\ d})} [s_{in} \to s_{out}] \right| \le \sum_{s_{mid}} \lambda \sqrt{\mu(S)\mu(T)} \tag{6}$$

This inequality represents the probability of generating a fixed s_{out}. To find the difference in probability over S_{out}, we take the sum:

$$||S_{out}(s_{in}, random) - S_{out}(s_{in}, pseudo)||_1 \le \sum_{s_{out}} \sum_{s_{mid}} \lambda \sqrt{\mu(S)\mu(T)}$$

Using Cauchy-Scwartz, we can bound this summation by:

$$\sum_{s_{out}}\sum_{s_{mid}} \lambda\sqrt{\mu(S)\mu(T)} \leq \lambda \cdot \sqrt{\sum_{s_{out}}\sum_{s_{mid}} \mu(S)}\sqrt{\sum_{s_{out}}\sum_{s_{mid}} \mu(T)}$$

Given the definition of S given above, it is clear that $\sum_{s_{mid}} \mu(S) = 1$ and for every value of s_{mid}, $\sum_{s_{out}} \mu(T) = 1$. Since there are 2^s choices for s_{out} and s_{mid}, we get:

$$||S_{out}(s_{in}, random) - S_{out}(s_{in}, pseudo)||_1 \leq \lambda \cdot \sqrt{2^s} \cdot \sqrt{2^s} = 2^s \cdot \lambda$$

$\square$

2 Pseudorandom Generators for Time-Bounded Computations

Pseudorandom Generators are only useful if they can be efficiently computed by a deterministic machine. A PRG is *quick* if it can can be computed in time $2^{O(l(r))}$ where $l(r)$ denotes a PRG's seed length. At first look, this notion of "quickness" might seem to be too slow (and in the crytographic setting it would be, as the desirable time there is something polynomial in the size of the seed length). Also, this may not be efficient enough if our goal is merely to reduce the amount of randomness needed by a computation. However, our present focus is full derandomization, achieved by trying all possible seeds and explicitly computing the probability that our algorithm accepts under the pseudorandom distribution. In this setting, we need $2^{l(r)}$ time just to look at all possible seeds, and so the factor $2^{O(l(r))}$ overhead in computing the PRGs output is affordable even though the running time would be more than polynomial if $l(r)$ is more than log.

The results that we obtain for the time-bounded setting differ from the space-bounded setting in that we do not know of any unconditional, non-trivial results. So, only under some reasonable hypothesis, can we reduce the amount of randomness for time-bounded comptuation, again by constructing pseudorandom generators. With this one in mind, the theorem we prove in the next two lectures is:

Theorem 1. *If there is $L \in E$ such that $C_L(n) \geq 2^{c \cdot n}$, for some constant $c > 0$ where E contains all decision problems which can be solved in linear exponential time ($2^{O(n)}$), then BPP = P.*

The theorem essentially states that if non-uniformity does not allow you a speed-up of a generic linear exponential algorithm, then we can derandomize BPP. Our proof of this fact will involve the construction of a PRG with logarithmic seed length that is computable in polynomial time. In fact, we will construct a PRG for not only BPP, but an even larger class - Boolean circuits. In particular the class of circuits we want to fool with our PRGs are circuits of linear size.

Definition 2. *An ϵ-PRG is a family of functions $(G_r)_r$ where $G_r : \{0,1\}^{l(r)} \to \{0,1\}^r$ such that for all circuits C of size at most r,*

$$\left| \Pr_{\rho \in \{0,1\}^r}[C(\rho) = 1] - \Pr_{\sigma \in \{0,1\}^{l(r)}}[C(G_r(\sigma) = 1] \right| \leq \epsilon$$

where ρ and σ are both chosen uniformly at random.

Essentially, the definition above says that any circuit of size at most r will have trouble determining whether its input was sampled from the uniform distribution or the pseudorandom distribution. We consider two questions that might naturally arise from the definition.

1. Why do want our constructed PRG to fool Boolean circuits when we are interested in fooling uniform computation? Since BPTIME(t) computation can be mimicked by circuits with size polynomial in t, we can use our constructed PRG to fool the uniform computaiton also. We consider Boolean circuits because we want our PRG to succeed in fooling the computations on all but finitely many inputs, and this is easily captured in the nonuniform setting by constructing a different circuit for each input where the input is hard-wired and the random bits are left as inputs to the circuit.

2. Why do we require that our PRG fool linear sized circuits? ? Using the same parameter r for the size of the circuit and its number of inputs will keep the arguments cleaner, and there is no harm in allowing the circuit to take more random bits than it needs. Mimicking a uniform computation with a circuit may yield a circuit that are larger than the number of random bits it needs, but the computation will not be affected by allowing the PRG to provide more random bits since the extra bits can be ignored.

We claim that if we have PRG that satisfies this property, then we can derandomize BPP. Since we are trying all possible seeds and computing the acceptance of the algorithm under the pseudorandom distribution, this takes time $2^{l(r)}$. Thus, if we construct a PRG with $O(\log r)$ seed length, we get a polynomial derandomized algorithm, implying BPP = P.

3 Notions of Pseudorandomness - Indistinguishability and Unpredictability

One way to think about the circuits we are attempting to fool is viewing them as a statistical test. Considering a PRG G_r with seed length $l(r)$ where $l(r) < r$, where r is the outupt length, then this distribution is very far from the uniform distribution since the PRG can only produce $2^{l(r)}$ strings out of all 2^r possible random strings or length r. Thus, given sufficient complexity, there does exist a statistical test that can distinguish between the uniform and pseudorandom distribution. But, this statistical test will have high circuit complexity and we only want the pseudorandom distribution to be computationally indistinguishable from the uniform distribution and so we only need to show that circuits with low circuit complexity, or simple statistical tests, cannot distinguish between the pseudorandom and uniform distribution.

In fact, we can restrict the type of statistical tests even further, and the only type of tests we need to consider are *Predictors.* A predictor is a small circuit (with size at most r) that looks at a prefix of its inputs, say $i-1$ bits, and tries to predict the i^{th} bit. No predictor exists for the uniform distribution since any circuit succeeds in predicting the i^{th} bit with probability $1/2$. What we want to show is that this unpredictability property implies the desirable indistinguishability property we discussed earlier for a pseudorandom distribution. To show this, we will show the contrapositive, that if we have a circuit that can distinguish between the uniform and pseudorandom distribution, then we can build a predictor that can predict a bit of the pseudorandom distribution, given the previous bits.

Lemma 2. *If there exists a circuit C of size at most r such that*

$$\left| \Pr_{\sigma \in \{0,1\}^{\ell(r)}} \left[C(G_r(\sigma)) = 1\right] - \Pr_{\rho \in \{0,1\}^r} \left[C(\rho) = 1\right] \right| \geq \epsilon$$

then there exists an $i \in \{1, \ldots, r\}$ and a circuit P of size at most r such that

$$\Pr_{\sigma \in \{0,1\}^{\ell(r)}} \left[P\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\right] \geq \frac{1}{2} + \frac{\epsilon}{r}.$$

Proof. Using the distinguishing circuit C, we will show that there exists a predictor P. We must decide which bit position i, our predictor P will predict. Consider the hybrid distribution $D_i (i = 0, ..., r)$ where for D_i, the first i bits are chosen from the output distribution of G_r and the remaining $r - i$ bits are chosen from the uniform distribution. So, D_0 is the uniform distribution for strings of length r and D_r is the output distribution of G_r. Intuitively, capturing the difference in how the circuit C behaves between distributions D_{i-1} and D_i, should give us a good idea whether C is able to predict the i_{th} bit from the first $i-1$ bits of a pseudorandom sample. Let us argue this formally. Let $\Pr_{D_i}[C = 1]$ represent the probability that C outputs 1 on a sample from distribution D_i, then

$$\begin{aligned}
\epsilon &\leq \left| \Pr_{D_r}[C = 1] - \Pr_{D_0}[C = 1] \right| \\
&= \left| \sum_{i=1}^{r} \big(\Pr_{D_i}[C = 1] - \Pr_{D_{i-1}}[C = 1] \big) \right| \\
&\leq \sum_{i=1}^{r} \left| \Pr_{D_i}[C = 1] - \Pr_{D_{i-1}}[C = 1] \right|
\end{aligned}$$

This means that $\exists i$ such that $|\Pr_{D_i}[C = 1] - \Pr_{D_{i-1}}[C = 1]| \geq \epsilon/r$. Let us choose this i as the index of the bit our predictor will predict and assume without loss of generality that $\Pr_{D_i}[C = 1] - \Pr_{D_{i-1}}[C = 1] \geq \epsilon/r$. Now we have that

$$\Pr\left[C\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, (G_r(\sigma))_i, \rho_{i+1}, \ldots, \rho_r\big) = 1\right]$$

is at least ϵ/r more than

$$\Pr\left[C\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, \rho_i, \rho_{i+1}, \ldots, \rho_r\big) = 1\right]$$

where the probabilities are taken over σ and $\rho_i, \rho_{i+1}, \ldots, \rho_r$ are chosen uniformly at random.

The circuit C appears to be able to recognize when i^{th} bit comes from the pseudorandom distribution, but it is not a predictor yet because for one, it accepts r bits whereas the predictor would only accept the first $i-1$ bits. However, by an averaging argument, there must be some setting $\widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r$ to the inputs $\rho_{i+1}, \ldots, \rho_r$ such that

$$\begin{aligned}
&\Pr\left[C\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, (G_r(\sigma))_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r\big) = 1\right] - \\
&\quad \Pr\left[C\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, \rho_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r\big) = 1\right] \geq \epsilon/r
\end{aligned} \tag{7}$$

where the probabilities are taken over σ and ρ_i chosen uniformly at random. We can hard-wire $\widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r$ into the circuit without affecting the circuit size. This is where nonuniformity is required as we need to handle each value of r separately.

So from the above discussion we can see that our circuit is ϵ/r more likely to output 1 when provided with i pseudorandom bits rather than when it is given $i-1$ pseudorandom bits plus one truly random bit. This observation suggests how we will build our predictor P'. Given the first $i-1$ bits $(\pi_1, \ldots, \pi_{i-1})$ sampled from the pseudorandom distribution, we will flip a coin to determine the i^{th} bit, calling this value ρ_i. We then evaluate $C(\pi_1, \ldots, \pi_{i-1}, \rho_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r)$ and if this evaluates to 1, we assume our guess was correct for $(G_r(\sigma)_i)$ and output ρ_i. Otherwise, we assume our guess was incorrect and output $\overline{\rho_i}$. To be more exact,

$$P'(\pi_1, \ldots, \pi_{i-1}) = \rho_i \oplus C(\pi_1, \ldots, \pi_{i-1}\rho_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r) \oplus 1.$$

Now, we prove a last claim whose result will prove the lemma.

Claim 1. $\Pr\left[P'\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\right] \geq \frac{1}{2} + \frac{\epsilon}{r}$ *where the probability is over* σ *and* ρ_i.

Proof. Consider two cases:

1. The output of circuit C does not depend on ρ_i, meaning C will output 1 or 0 regardless of ρ_i's value.

2. The output of C *does* depend on ρ_i. If the actual value x_i will produce an output of 1 on C then P' will always guess correctly (outputs 1 $\iff$ $x_i = 1$). If x_i's true value will make C output 0, then our predictor is always wrong.

We need to show that for both cases, $|\Pr\left[P'\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\right]| - 1/2$ is equal to (7). For the first case, (7) is equal to 0 (since both probabilities will be the same), and
$|\Pr\left[P'\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\right]| - 1/2 = 0$ also since P' has 1/2 chance of outputting the right value since C's output does not help P' in any way.

For the second case, if actual value x_i produces an output of 1 on C, then (7) is equal to 1/2 (since the pseudoranom bit $(G_r(\sigma))_i$ causes C's output to be 1 whereas the random bit ρ_i has 1/2 chance of causing output to be 1) and here, P' is always correct in its prediction so
$|\Pr\left[P'\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\right]| - 1/2 = 1/2$ also. When the actual value x_i produces an output of 0 on C, then (7) is equal to $-1/2$ (since the pseudoranom bit $(G_r(\sigma))_i$ causes C's output to be 0 whereas the random bit ρ_i has 1/2 chance of causing output to be 1) and here, P' is always wrong so $|\Pr\left[P'\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\right]| - 1/2 = -1/2$ also.

Thus, since (7) is always greater than $\frac{\epsilon}{r}$, we are done.

□

This is exactly the behavior we want from our predictor, but P' still draws on one random bit. In order to create a deterministic P we can again take advantage of the fact that we're in the nonuniform setting and hard-wire ρ_i to some value $\widetilde{\rho}_i$, either 0 or 1, such that the circuit retains its advantage of ϵ/r in predicting the ith bit. This yields a predictor P where $P(\pi_1, \ldots, \pi_{i-1})$ is just

$$\widetilde{\rho}_i \oplus C(\pi_1, \ldots, \pi_{i-1}, \widetilde{\rho}_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r) \oplus 1,$$

which can be expressed as a circuit of the same size as C (possibly with an additional NOT gate, which we assume doesn't increase the size of the circuit). This predictor satisfies

$$\Pr\left[P\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\right] \geq \frac{1}{2} + \frac{\epsilon}{r},$$

as desired.

□

4 Next Time

We continue with our discussion of Time-Bounded Derandomization.

Acknowledgements

In writing the notes for this lecture, I perused the notes by Michael Correll for lecture 16 from the Spring 2010 offering of CS 710, the notes by Andrew Bolanowski for lecture 17 from the Spring 2010 offering of CS 710, and the notes by Jake Rosin for lecture 14 from the Spring 2007 offering of CS 710