CMPT 710 - Complexity Theory: Lecture 16

Valentine Kabanets

October 28, 2003

1 Reducing the error in BPP

Recall that a language L is in BPP if there is a deterministic polytime TM $M(x,r)$, such that for $x \in L$, $M(x,r)$ accepts at least $3/4$ of r's, and for $x \notin L$, $M(x,r)$ accepts at most $1/4$ of all r's. In each case, the probability that M makes a mistake (e.g., accepts $x \notin L$) is at most $1/4$. The choice of this error probability to be $1/4$ is rather arbitrary. As we show next, it is always possible to make the error probability exponentially small, by increasing the running time only slightly.

The idea is to choose l independent copies of a random string r, i.e., chose independently uniformly at random $r_1, r_2, \dots, r_l$. Then simulate $M(x, r_i)$, for each $1 \leq i \leq l$, noting whether M accepts or rejects for each r_i. Finally, our new algorithm will accept if and only if the *majority* (i.e., $> l/2$) of random strings r_i's were accepted.

Intuitively, if $x \in L$, then each r_i has the probability at least $3/4$ of being accepted by $M(x, r_i)$. So, on average, M will accept at least $3/4$ of the strings in the list $r_1, r_2, \dots, r_l$. It seems very unlikely that the actual number of accepted r_i's will deviate significantly from the expected number. (This is basically the Law of Large Numbers from Probability Theory.)

To make this intuition precise, we'll need the following theorem.

Theorem 1 (Chernoff bounds). *Let $X_1, \dots, X_n$ be independent identically distributed random variables such that $\mathbf{Pr}[X_i = 1] = p$ and $\mathbf{Pr}[X_i = 0] = 1 - p$, for some $0 \leq p \leq 1$. Consider the new random variable $X = \sum_{i=1}^n$ with expectation $\mu = pn$. Then*

$$\mathbf{Pr}[|X - \mu| > \delta\mu] < 2e^{-\delta^2\mu/3}.$$

The proof of this theorem can be found in many textbooks (e.g., Papadimitriou's book), and will not be proved in class.

Back to the analysis of our new algorithm that simulates $M(x,r)$ on l independent copies of random string r_i. Let's define a random variable X_i, for $1 \leq i \leq l$, such that $X_i = 1$ if $M(x, r_i)$ produces a correct answer (i.e., accepts if $x \in L$, and rejects if $x \notin L$), and $X_i = 0$ otherwise. Observe that, by definition of BPP, we have $\mathbf{Pr}[X_i = 1] = p \geq 3/4$. Let's analyze the probability that our new algorithm makes a mistake, i.e., that the majority of the variables $X_1, \dots, X_l$ are 0. Let $X = \sum_{i=1}^l X_i$, and let $\mu = pl \geq (3/4)l$ be the expectation of X. We have

$$\mathbf{Pr}[X < l/2] \leq \mathbf{Pr}[X < (2/3)\mu] = \mathbf{Pr}[|X - \mu| > \mu/3].$$

By the Chernoff theorem, the last probability is at most $2e^{-\mu/27} \leq 2e^{-l/36}$, which is exponentially small in l, the number of times we run the original algorithm M.

Thus, by taking $l = \mathsf{poly}(n)$, where $n = |x|$, we can reduce the probability error of a new algorithm below an inverse exponential in n, while still running in polytime.

2 BPP and Small Circuits

The ability to reduce the error probability in BPP has a curious consequence that every language in BPP is computable by a family of polysize Boolean circuits.

Theorem 2. $\mathsf{BPP} \subset \mathsf{P/poly}$

Remark: Note that $\mathsf{P} \subset \mathsf{P/poly}$ trivially, since a polytime deterministic TM can be simulated by a polysize Boolean circuit. However, it is not at all trivial to argue that every BPP algorithm can also be simulated by a small Boolean circuit. The problem seems to be: what do we do with random strings used by the BPP algorithm? Now, we'll prove the theorem, and thus explain what to do with all those random strings.

Proof. Consider an arbitrary $L \in \mathsf{BPP}$. Since we know how to reduce the error probability below any inverse exponential in the input size, we may assume that there is a deterministic polytime TM $M(x, r)$ such that, for every $x \in L$, $\mathbf{Pr}_r[M(x, r) = 1] > 1 - 2^{-2n}$, and for every $x \notin L$, $\mathbf{Pr}_r[M(x, r) = 1] < 2^{-2n}$, where $|x| = n$, and $|r|$ is some polynomial in n.

Now, $\mathbf{Pr}_r[\exists x \text{ s.t. } M(x, r) \text{ is wrong}] \leq 2^n \mathbf{Pr}_r[M(x, r) \text{ is wrong for a fixed } x] \leq 2^n 2^{-2n} = 2^{-n} \ll 1$. The first inequality is the so-called "union bound" saying that for any events $E_1, \dots, E_n$, $\mathbf{Pr}[E_1 \vee E_2 \dots E_n] \leq \sum_{i=1}^n \mathbf{Pr}[E_i]$. The second inequality uses the fact that M has error probability at most 2^{-2n} for every input of size n.

Hence, there must *exists* at least one string $\hat{r}$ such that $M(x, \hat{r})$ is correct *for every input* x of length n. We can use such a string $\hat{r}$ as an advice string, and then just simulate $M(x, \hat{r})$ on any given input x of length n. Thus, $L \in \mathsf{P/poly}$. □