

CMPT 710 - Complexity Theory: Lecture 16

Valentine Kabanets

November 1, 2007

1 Randomized Computation

Why is randomness useful? Imagine you have a stack of bank notes, with very few counterfeit ones. You want to choose a genuine bank note to pay at a store. However, suppose that you don't know how to distinguish between a "good" bank note and a "bad" one. What can you do? Well, if you pick a bank note at random, you will be lucky with high probability (here the probability of picking a good bank note is equal to the fraction of good bank notes in your stack).

Let's consider a more realistic example. Alice and Bob communicate over some channel. The communication is very expensive. Both Alice and Bob have an n -bit number. Alice has $a = a_1 \dots a_n$ and Bob has $b = b_1 \dots b_n$. They want to know if $a = b$.

Clearly, this check can be achieved with at most $n + 1$ communicated: Alice just sends her number a to Bob. Bob then compares a and b , and sends back to Alice one bit (1 if $a = b$, and 0 otherwise). If we allow only deterministic protocols, then this bound is the best one can get.

However, if we allow randomized protocols, we can do much better. Here, a randomized protocol may have Alice and Bob make a mistake (e.g., they may think that $a = b$ when in fact $a \neq b$), but this error should happen with very small probability over the random choices of Alice and Bob. Next, we'll describe a $O(\log n)$ -communication protocol to check if $a = b$.

Alice and Bob will first find the smallest prime number p such that $n^2 \leq p \leq n^3$. (The high density of primes guarantees that there will be a prime between n^2 and n^3 ; in the interval $1, 2, \dots, m$, there are about $\Theta(m/\ln m)$ primes.) Then Alice picks a random element $r \in \mathbb{Z}_p$, computes $A(r) = a_1 r^{n-1} + a_2 r^{n-2} + \dots + a_n \pmod p$, and sends to Bob the pair $(r, A(r))$. Upon receiving this pair, Bob will compute $B(r) = b_1 r^{n-1} + b_2 r^{n-2} + \dots + b_n \pmod p$ and test if $A(r) = B(r)$. If they are equal, Bob will send 1 to Alice (saying that he thinks that $a = b$); otherwise, Bob will send 0 to Alice (saying that he thinks $a \neq b$).

Observe that the amount of communication for the described protocol is at most $2|p|+1 \in O(\log n)$, by our choice of $p \leq n^3$. Now let's analyze the correctness. First, if $a = b$, then $A(x) = B(x)$ as polynomials, and hence, $A(r) = B(r)$ for any r . So in this case, Alice and Bob correctly decide that $a = b$ with probability 1.

In the case where $a \neq b$, we have that $A(x)$ and $B(x)$ are different polynomials of degree at most $n - 1$. So, their difference $C(x) = A(x) - B(x)$ is a *non-zero* polynomial of degree

at most $n - 1$. The probability that Alice and Bob erroneously decide that $a = b$ is exactly $\Pr_{r \in Z_p}[C(r) = 0]$. Now, since $C(x)$ is of degree at most $n - 1$, it may have at most $n - 1$ roots, i.e., values r at which C is zero. So, we have

$$\Pr_{r \in Z_p}[C(r) = 0] \leq (n - 1)/|Z_p| \leq n/n^2 = 1/n.$$

So, in case $a \neq b$, Alice and Bob will decide that $a \neq b$ with probability at least $1 - 1/n$.

Note that by picking p to be a larger number, e.g., at least n^{100} , we can make the error probability of this protocol smaller than n^{-99} . The communication complexity remains $O(\log n)$.

2 Randomized Complexity Classes

A language $L \in \text{RP}$ if there is a deterministic polytime TM $M(x, r)$ such that

1. for all $x \in L$, $\Pr_r[M(x, r) \text{ accepts}] \geq 1/2$, and
2. for all $x \notin L$, $\Pr_r[M(x, r) \text{ accepts}] = 0$.

Now, to decide L , on input x , a randomized algorithm may first flip $|r|$ random coins to compute a random string r , and then simulate $M(x, r)$. This randomized algorithm will run in polytime, and will be correct for all $x \notin L$ with probability 1, and will be correct for all $x \in L$ with probability at least $1/2$.

Note that if $L \in \text{RP}$, then $L \in \text{NP}$. So, we get that $\text{RP} \subseteq \text{NP}$. The class RP contains those languages that can be decided probabilistically with *one-sided* error: an algorithm may err on positive instances, but never on negative instances. Next, we define the class of languages decidable with two-sided error.

A language $L \in \text{BPP}$ if there is a polytime DTM $M(x, r)$ such that

1. for all $x \in L$, $\Pr_r[M(x, r) \text{ accepts}] \geq 3/4$, and
2. for all $x \notin L$, $\Pr_r[M(x, r) \text{ accepts}] \leq 1/4$.

Note that now we allow a “small” probability of error even on inputs not in the language.

3 Reducing the error probability

Consider any $L \in \text{RP}$. Let $M(x, r)$ be the corresponding polytime DTM. We design a new DTM $M'(x; r_1, \dots, r_l)$ such that $M'(x; r_1, \dots, r_l)$ accepts iff there is some $1 \leq i \leq l$ such that $M(x, r_i)$ accepts; in other words, M' simulates M for l independent random strings. We claim that the error probability of the new M' is at most 2^{-l} .

Indeed, if $x \notin L$, then $M(x, r_i)$ rejects for every i , and so M' also rejects for all sequences r_1, \dots, r_l . On the other hand, if $x \in L$, then $\Pr_{r_1, \dots, r_l}[M'(x; r_1, \dots, r_l) \text{ rejects}]$ is equal to $\prod_{i=1}^l \Pr_{r_i}[M(x, r_i) \text{ rejects}] \leq (1/2)^l$. So, in the case where $x \in L$, the TM M' accept for a fraction $1 - 2^{-l}$ of all random sequences $r_1 \dots r_l$.

This ability to reduce error probability allows us to prove the following.

Theorem 1. $\text{RP} \subseteq \text{BPP}$

Proof. By the argument above, for any $L \in \text{RP}$, there is a DTM $M(x, r)$ such that, for every $x \in L$, $\Pr_r[M(x, r) \text{ accepts}] \geq 1 - 1/4 = 3/4$ — simply take $l = 2$ in the above-mentioned error reduction procedure. On the other hand, for every $x \notin L$, $\Pr_r[M(x, r) \text{ accepts}] = 0 < 1/4$. So, $L \in \text{BPP}$. \square

What about NP and BPP ? Is one a subclass of the other? It is unknown! The general belief is that $\text{BPP} = \text{P}$, and therefore, BPP is a subset of NP . But, no unconditional result of that kind is known.