Notes on Computer Theory Last updated: November, 2015

Randomized computation, Chernoff bounds.

Notes by Jonathan Katz, lightly edited by Dov Gordon.

1 Randomized Time Complexity

Is deterministic polynomial-time computation the only way to define "feasible" computation? Allowing *probabilistic* algorithms, that may fail with tiny probability, seems reasonable. (In particular, consider an algorithm whose error probability is lower than the probability that there will be a hardware error during the computation, or the probability that the computer will be hit by a meteor during the computation.) This motivates our exploration of probabilistic complexity classes.

There are two different ways to define a randomized model of computation. The first is via Turing machines with a *probabilistic transition function*: as in the case of non-deterministic machines, we have a Turing machine with two transitions functions, and a *random* one is applied at each step. The second way to model randomized computation is by augmenting Turing machines with an additional (read-only) *random tape*. For the latter approach, one can consider either one-way or two-way random tapes; the difference between these models is unimportant for randomized time complexity classes, but (as we will see) becomes important for randomized space classes. Whichever approach we take, we denote by $M(x)$ a random computation of M on input x, and by $M(x;r)$ the (deterministic) computation of M on input x using random choices r (where, in the first case, the ith bit of r determines which transition function is used at the ith step, and in the second case r is the value written on M's random tape).

We now define some randomized time-complexity classes; in the following, PPT stands for "probabilistic, polynomial time" (where this is measured as *worst-case* time complexity over all inputs, and as always the running time is measured as a function of the length of the input x).

Definition 1 *$L \in \mathcal{RP}$ if there exists a* PPT *machine M such that:*

$$\begin{aligned} x \in L &\Rightarrow \Pr[M(x) = 1] \geq 1/2 \\ x \notin L &\Rightarrow \Pr[M(x) = 0] = 1. \end{aligned}$$

Note that if $M(x)$ outputs "1" we are sure that $x \in L$; if $M(x)$ outputs "0" we cannot make any definitive claim.

Viewing M as a non-deterministic machine for L, the above means that when $x \in L$ at least half of the computation paths of $M(x)$ accept, and when $x \notin L$ then none of the computation paths of $M(x)$ accept. Put differently, a random tape r for which $M(x;r) = 1$ serves as a witness that $x \in L$. We thus have $\mathcal{RP} \subseteq \mathcal{NP}$.

Symmetrically:

Definition 2 *$L \in \mathsf{co}\mathcal{RP}$ if there exists a* PPT *machine M such that:*

$$\begin{aligned} x \in L &\Rightarrow \Pr[M(x) = 1] = 1 \\ x \notin L &\Rightarrow \Pr[M(x) = 0] \geq 1/2. \end{aligned}$$

Here, if $M(x)$ outputs "0' we are sure that $x \notin L$, but if $M(x)$ outputs "1" we cannot make any definitive claim.

The above classes allow *one-sided* error. A more general notion of randomized computation allows for *two-sided* error. For a language L, let $\chi_L(x) = 1$ iff $x \in L$.

Definition 3 *$L \in \mathcal{BPP}$ if there exists a* PPT *machine M such that:*

$$\Pr[M(x) = \chi_L(x)] \geq 2/3.$$

In other words,

$$\begin{aligned} x \in L &\Rightarrow \Pr[M(x) = 1] \geq 2/3 \\ x \notin L &\Rightarrow \Pr[M(x) = 1] \leq 1/3. \end{aligned}$$

Finally, we may also consider randomized algorithms that make no errors (but may not give a result at all):

Definition 4 *$L \in \mathcal{ZPP}$ if there exists a* PPT *machine M such that:*

$$\Pr[M(x) = \chi_L(x)] \geq 1/2$$

$$\Pr\left[M(x) \in \{\chi_L(x), \perp\}\right] = 1.$$

We now explore these definitions further. A first observation is that, for all the above definitions, the constants are essentially arbitrary. We focus on the case of $\mathcal{BPP}$ and leave consideration of the rest as an exercise.

Theorem 1 *The following are both equivalent definitions of $\mathcal{BPP}$:*

1. *$L \in \mathcal{BPP}$ if there exists a* PPT *machine M and a polynomial p such that:*

$$\Pr[M(x) = \chi_L(x)] \geq \frac{1}{2} + \frac{1}{p(|x|)}.$$

2. *$L \in \mathcal{BPP}$ if there exists a* PPT *machine M and a polynomial q such that:*

$$\Pr[M(x) = \chi_L(x)] \geq 1 - 2^{-q(|x|)}.$$

Proof We show how to transform an algorithm M satisfying the first definition into an algorithm M' satisfying the second definition. $M'(x)$ is defined as follows: run $M(x)$ a total of $t(|x|)$ times (for some polynomial t to be fixed later) using independent random coins in each execution. Then M' outputs the bit that was output by a majority of these executions.

To analyze the behavior of M', we rely on the additive version of the *Chernoff bound* [2]:

Claim 2 *Let $\rho \leq \frac{1}{2}$ and let $X_1, \ldots, X_t$ be independent, identically-distributed 0-1 random variables with $\Pr[X_i = 1] = \rho$ for each i. Then for all $\varepsilon > 0$, we have:*

$$\Pr\left[\left|\sum_{i=1}^{t} X_i - \rho t\right| > \varepsilon t\right] < 2e^{-2t\varepsilon^2}$$

Although we don't use it here, there is also a very useful Chernoff bound on the multiplicative error:

Claim 3 *Let $\rho \leq \frac{1}{2}$ and let $X_1, \ldots, X_t$ be independent, identically-distributed 0-1 random variables with $\Pr[X_i = 1] = \rho$ for each i. Then for all $\varepsilon > 0$, we have:*

$$\Pr\left[\sum_{i=1}^{t} X_i > (1+\varepsilon)\rho t\right] \leq e^{-\frac{\rho t \varepsilon^2}{3}}$$

and

$$\Pr\left[\sum_{i=1}^{t} X_i < (1-\varepsilon)\rho t\right] \leq e^{-\frac{\rho t \varepsilon^2}{2}}$$

Let X_i denote the output of the i^{th} execution of $M(x)$. When $x \notin L$

$$\Pr[X_i = 1] < \frac{1}{2} - \frac{1}{p(|x|)} \stackrel{\text{def}}{=} \rho.$$

Furthermore, by definition of M' (letting $t \stackrel{\text{def}}{=} t(|x|)$):

$$\begin{aligned}
\Pr[M'(x) = 1] &= \Pr\left[\frac{\sum_{i=1}^{t} X_i}{t} > \frac{1}{2}\right] \\
&= \Pr\left[\frac{\sum_{i=1}^{t} X_i}{t} > \rho + \frac{1}{p(|x|)}\right] \\
&\leq \Pr\left[\left|\sum_{i=1}^{t} X_i - \rho t\right| > \frac{t}{p(|x|)}\right] \\
&< e^{-\frac{2t}{p(|x|)^2}}
\end{aligned}$$

Setting $t = O\left(q(|x|) \cdot p(|x|)^2\right)$ gives the desired result. (An exactly analogous argument works for the case $x \in L$.) ■

How do the above classes relate to each other? It is immediate that

$$\mathcal{RP} \cup \mathsf{co}\mathcal{RP} \subseteq \mathcal{BPP},$$

and so $\mathcal{BPP}$ is a (potentially) more powerful class. Indeed, $\mathcal{BPP}$ appears to capture feasible probabilistic computation. We also claim that

$$\mathcal{ZPP} = \mathcal{RP} \cap \mathsf{co}\mathcal{RP};$$

this is left as an exercise.

1.1 An Example of a Randomized Algorithm

There are several examples of where randomized algorithms are more efficient, or simpler, than known deterministic algorithms. However, there are not as many examples of problems that are known to be solvable by polynomial-time randomized algorithms, but not known to be solved by polynomial-time deterministic algorithms. One famous former example was testing primality: this problem was known to be in $\mathsf{co}\mathcal{RP}$ since the late 1970s, but was only shown to be in $\mathcal{P}$ in 2005. (Nevertheless, in practice the randomized algorithms are still used since they are faster.)

Before going through a few examples, we introduce the concept of a *finite field*. In our case, we will focus only on fields of prime order (size): for some prime p, the integers $\{0, 1, \ldots, p-1\}$ constitute the field $\mathcal{GF}(p)$, where addition and multiplication are done modulo p. For example, in $\mathcal{GF}(7)$, we have $5 \times 3 = 1$, and $6 + 4 = 3$.

Claim 4 *Given elements $a, b, c \in \mathcal{GF}(P)$ such that $a \cdot b \equiv a \cdot c$, either $a \equiv 0$, or $b \equiv c$.*

Proof If we instead perform operations over the integers, our assumptions equivalently state that $a \cdot b = a \cdot c + k \cdot p$ for some integer k. It follows that $a(b-c) = k \cdot p$, which implies that either $a \equiv 0 \mod p$, or $(b-c) \equiv 0 \mod p$. ■

Claim 5 *For any $a, b \in \mathcal{GF}(p)$ where a is non-zero, and for r sampled uniformly from $\mathcal{GF}(p)$, $\Pr[a \cdot r = b] = 1/p$.*

Proof By the Claim 4, and because we know that a is nonzero, there is only one value $r \in \mathcal{GF}$ for which $a \cdot r \equiv b$. The probability of choosing this value is $1/p$. ■

A search problem for which probabilistic polynomial-time algorithms are known, but deterministic polynomial-time algorithms are not, is computing square roots modulo a prime. We will instead give a randomized algorithm for a problem that has a simple, but less efficient deterministic algorithm.

Matrix multiplication testing. The language we're interested in is

$$\{(A, B, C) | AB = C, \text{ and each element is a matrix in } \mathcal{GF}(p)^{n \times n}\}$$

A simple deterministic algorithm follows by matrix multiplication: simply compute AB and compare each of the n^2 elements against those of C. This takes time $O(n^3)$ if the multiplication is done naively. The best known construction for matrix multiplication is $O(n^{2.3729})$, though the algorithm would be quite complex. We give a randomized algorithm that takes time $O(n^2)$, with error $1/p$. Repeating will lower the error rate, and, of course the exact run-time will depend on the desired error. The algorithm is simple: instead of computing AB, which is expensive, we sample a random vector $\vec{r} \leftarrow \mathcal{GF}(p)^n$, and compare $A \cdot (B \cdot r)$ with $C \cdot r$, which takes time $O(n^2)$. We now analyze the error probability. Clearly the error is one-sided: if $AB = C$, then $A \cdot (B \cdot r) \equiv C \cdot r$. So we assume that AB differs from C in at least one place, which we'll let be (i, j). For the sake of analysis, it is easier to consider $AB - C = D$, and look at the probability that $D \cdot r \equiv 0$. Consider the ith element of $D \cdot r$, which is $d_{i,1}r_1 + d_{i,2}r_2 + \cdots + d_{i,j}r_j + \cdots + d_{i,n}r_n = d_{i,j}r_j + y$ for some element $y \in \mathcal{GF}(p)$. The probability that this value is 0 can be written as

$$\Pr[(d_{i,j}r_j + y) = 0 \mid y = 0] \cdot \Pr[y = 0] + \Pr[(d_{i,j}r_j + y) = 0 \mid y \neq 0] \cdot \Pr[y \neq 0].$$

Because we know that $d_{i,j}$ is nonzero, $\Pr[(d_{i,j}r_j + y) = 0 \mid y = 0] = 1/p$ (i.e. the condition is satisfied only when $r_j = 0$). From Claim 5, we know that $\Pr[(d_{i,j}r_j + y) = 0 \mid y \neq 0] = 1/p$: there is exactly one additive inverse of y (namely, $p - y$), and the probability that $d_{i,j} \cdot r_j = p - y$ is $1/p$. Putting these facts together, we have that the probability that the ith element of $D \cdot r \equiv 0$ is

$$\frac{1}{p} \cdot \Pr[y = 0] + \frac{1}{p} \cdot \Pr[y \neq 0] = \frac{1}{p}$$

1.2 How Large is $\mathcal{BPP}$?

We know that

$$\mathcal{P} \subseteq \mathcal{ZPP} = \mathcal{RP} \cap \mathsf{co}\mathcal{RP} \subseteq \mathcal{RP} \cup \mathsf{co}\mathcal{RP} \subseteq \mathcal{BPP} \subseteq \mathsf{PSPACE}$$

We currently do not have a very good unconditional bound on the power of $\mathcal{BPP}$ — in particular, it could be that $\mathcal{BPP} = \mathsf{NEXP}$. Perhaps surprisingly, especially in light of the many randomized algorithms known, the current conjecture is that $\mathcal{BPP}$ is *not* more powerful than $\mathcal{P}$. We will return to this point later in the semester when we talk about derandomization.

What (unconditional) upper bounds *can* we place on $\mathcal{BPP}$? Interestingly, we know that it is not more powerful than polynomial-size circuits; actually, the following theorem is also a good illustration of the power of non-uniformity.

Theorem 6 $\mathcal{BPP} \subset \mathcal{P}/\mathsf{poly}$.

Proof Let $L \in \mathcal{BPP}$. Using amplification, we know that there exists a polynomial-time Turing machine M such that $\Pr[M(x) \neq \chi_L(x)] < 2^{-|x|^2}$. Say M uses (at most) $p(|x|)$ random coins for some polynomial p. (Note that p is upper-bounded by the running time of M.) An equivalent way of stating this is that for each n, and each $x \in \{0,1\}^n$, the set of "bad" coins for x (i.e., coins for which $M(x)$ outputs the wrong answer) has size at most $2^{p(n)} \cdot 2^{-n^2}$. Taking the union of these "bad" sets over all $x \in \{0,1\}^n$, we find that the total number of random coins which are "bad" for *some* x is at most $2^{p(n)} \cdot 2^{-n} < 2^{p(n)}$. In particular, there exists at least one set of random coins $r_n^* \in \{0,1\}^{p(n)}$ that is "good" for *every* $x \in \{0,1\}^n$ (in fact, there are many such random coins). If we let the sequence of "advice strings" be exactly $\{r_n^*\}$ (using the alternate definition of $\mathcal{P}_{/\mathsf{poly}}$), we obtain the result of the theorem. ■

We can also place $\mathcal{BPP}$ in the polynomial hierarchy:

Theorem 7 $\mathcal{BPP} \subseteq \Sigma_2 \cap \Pi_2$.

Proof We show that $\mathcal{BPP} \subseteq \Sigma_2$; since $\mathcal{BPP}$ is closed under complement, this proves the theorem.

We begin by proving some probabilistic lemmas. Say $S \subseteq \{0,1\}^m$ is *large* if $|S| \geq (1 - \frac{1}{m})2^m$, and is *small* if $|S| < \frac{2^m}{m}$. For a string $z \in \{0,1\}^m$ define $S \oplus z \stackrel{\text{def}}{=} \{s \oplus z \mid s \in S\}$.

Claim 8 *If S is small, then for all $z_1, \ldots, z_m \in \{0,1\}^m$ we have $\bigcup_i (S \oplus z_i) \neq \{0,1\}^m$.*

This follows easily since

$$\left|\bigcup_i (S \oplus z_i)\right| \leq \sum_i |S \oplus z_i| = m \cdot |S| < 2^m.$$

Claim 9 *If S is large, then there exist $z_1, \ldots, z_m \in \{0,1\}^m$ such that $\bigcup_i (S \oplus z_i) = \{0,1\}^m$.*

In fact, we show that choosing at random works with high probability; i.e.,

$$\Pr_{z_1,\ldots,z_m \in \{0,1\}^m} \left[\bigcup_i (S \oplus z_i) = \{0,1\}^m\right] \geq 1 - \left(\frac{2}{m}\right)^m.$$

To see this, consider the probability that some fixed y is not in $\bigcup_i (S \oplus z_i)$. This is given by:

$$\begin{aligned} \Pr_{z_1,\ldots,z_m \in \{0,1\}^m}[y \notin \bigcup_i (S \oplus z_i)] &= \prod_i \Pr_{z \in \{0,1\}^m}[y \notin (S \oplus z)] \\ &\leq \left(\frac{1}{m}\right)^m. \end{aligned}$$

Applying a union bound by summing over all $y \in \{0,1\}^m$, we see that the probability that there *exists* a $y \in \{0,1\}^m$ which is not in $\bigcup_i (S \oplus z_i)$ is at most $\frac{2^m}{m^m}$.

We now prove the theorem. Given $L \in \mathcal{BPP}$, there exist a polynomial m and an algorithm M such that M uses $m(|x|)$ random coins and errs with probability less than $1/m$. For any input x, let $S_x \subseteq \{0,1\}^{m(|x|)}$ denote the set of random coins for which $M(x;r)$ outputs 1. Thus, if $x \in L$ (letting $m = m(|x|)$) we have $|S_x| > (1 - \frac{1}{m}) \cdot 2^m$ while if $x \notin L$ then $|S_x| < \frac{2^m}{m}$. This leads to the following Σ_2 characterization of L:

$$x \in L \Leftrightarrow \exists z_1, \ldots, z_m \in \{0,1\}^m \; \forall y \in \{0,1\}^m : y \in \bigcup_i (S_x \oplus z_i).$$

(Note the desired condition can be efficiently verified by checking if $M(x; y \oplus z_i) \stackrel{?}{=} 1$ for some i.) ■

References

[1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] Wikipedia: https://en.wikipedia.org/wiki/Chernoff_bound