TU Dresden, Fakultät Informatik
Daniel Borchmann Summer Term 2016

Advanced Topics in Complexity Theory

Exercise 2: Revision Randomized Computation

2016-04-12

The purpose of this exercise is to recall the basic notions of *randomized computation*. For this we first need to formalize what we mean by this.

As the computation model for randomized computations we stick with our usual notion of deterministic Turing machines. However, such machine get in addition to the actual input x some "source of randomness" that they can use while performing their computation.

Definition 2.1 A *probabilistic Turing maching* (PTM) is a deterministic Turing machine over an input alphabet of the form $\Sigma^* \times \{ 0, 1 \}^*$. A PTM M is called $p(n)$*-time bounded* if M stops after at most $p(|x|)$ steps on inputs (x, y). $\diamondsuit$

Note that for $p(n)$-time bounded PTMs we can assume without loss of generality that $y \in \{ 0, 1 \}^{p(|x|)}$.

In contrast to the notion of classical computation, a probabilistic Turing machine does not have a definite answer on some input x. Instead, the result is a *random variable* that depends on the choice of the randomness y.

To make this more precise, we first introduce probability spaces Ω_m for each $m \in \mathbb{N}$. These spaces are meant to model the probability of choosing a particular input of length m. They are defined as $\Omega_m := (\{ 0, 1 \}^m, \mathfrak{P}(\{ 0, 1 \}^m), \mathrm{Pr}_m)$, where for each $u \in \{ 0, 1 \}^m$ we have $\mathrm{Pr}_m(\{ u \}) = 2^{-m}$. In other words, for inputs of length m each choice of u is equally likely.

Definition 2.2 Let M be a $p(n)$-time bounded PTM and let $x \in \Sigma^*$. Let $n = |x|$. Then the *probability* that M accepts x is defined as

$$\mathrm{Pr}(M \text{ accepts } x) = \mathrm{Pr}_{p(n)}(\{ y \in \{ 0, 1 \}^{p(n)} \mid (x, y) \in \mathcal{L}(M) \}). \qquad \diamondsuit$$

In other words, the probability that M accepts input x is the probability to choose a random string $y \in \{ 0, 1 \}^{p(|x|)}$ that makes M accept on input (x, y). Indeed, it is not hard to see that

$$\mathrm{Pr}(M \text{ accepts } x) = \frac{1}{2^{p(n)}} |\{ y \in \{ 0, 1 \}^{p(n)} \mid (x, y) \in \mathcal{L}(M) \}|,$$

where again $n = |x|$.

Probabilistic Turing machines can be used to define languages. The following is an easy characterization of NP in terms of PTMs.

Exercise 2.3 Show that $L \in \mathsf{NP}$ if and only if there exists a polynomial-time bounded PTM M such that

$$L = \{ x \in \Sigma^* \mid \mathrm{Pr}(M \text{ accepts } x) > 0 \}.$$

Solution Let $L \in \mathsf{NP}$. Then there exists a deterministic polynomial-time Turing machine M and some polynomial $p(n)$ such that

$$L = \{\, x \in \Sigma^* \mid \exists y\colon |y| \leq |p(|x|)| \wedge M \text{ accepts } (x, y) \,\}.$$

Wlog assume that M runs in time $p(n)$ where n is the length of the first component. Then M is a PTM over Σ^* such that

$$\begin{aligned} L &= \{\, x \in \Sigma^* \mid \Pr_{p(|x|)}((x, y) \in \mathcal{L}(M)) > 0 \,\} \\ &= \{\, x \in \Sigma^* \mid \Pr(M \text{ accepts } x) > 0 \,\}. \end{aligned}$$

Conversely, let $L = \{\, x \in \Sigma^* \mid \Pr(M \text{ accepts } x) > 0 \,\}$ for some polynomial-time bounded PTM M. Then for some polynomial p we have

$$L = \{\, x \in \Sigma^* \mid \Pr_{p(|x|)}((x, y) \in \mathcal{L}(M)) > 0 \,\}.$$

This means that

$$L = \{\, x \in \Sigma^* \mid \exists y. |y| \leq |p(|x|)| \wedge (x, y) \in \mathcal{L}(M) \,\}$$

and thus $L \in \mathsf{NP}$. □

Thus, we can see NP as the class of all languages for which a randomized algorithm can guess the right answer. However, for a "good" randomized algorithm, this is not sufficient. Instead, one wants that the *majority* of guess are good. This leads to the definition of several complexity classes based on randomized computation.

Definition 2.4 We define the class PP (for *probabilistic polynomial time*) as the class of all languages L for which a polynomial-time PTM M exists such that

$$L = \{\, x \in \Sigma^* \mid \Pr(M \text{ accepts } x) > 1/2 \,\}.$$

Furthermore, the class BPP (for *bounded-error probabilistic polynomial time*) is the class of all languages L such that there exists a polynomial-time PTM M such that

$$\begin{aligned} x \in L &\implies \Pr(M \text{ accepts } x) \geq 2/3 \\ x \notin L &\implies \Pr(M \text{ accepts } x) \leq 1/3. \end{aligned}$$

Finally, the class RP (for *randomized polynomial time*) is the class of all languages L such that there exists a polynomial-time PTM M such that

$$\begin{aligned} x \in L &\implies \Pr(M \text{ accepts } x) \geq 2/3 \\ x \notin L &\implies \Pr(M \text{ accepts } x) = 0. \end{aligned}$$ ♢

In other words, the class PP consists of all languages for which a probabilistic algorithm exists that guesses more often right than wrong. A language is in BPP if a probabilistic algorithm exists that always guesses right with a distinguished majority. A language is in RP if a probabilistic algorithm can guess right in the majority of cases for a valid input, and rejects all invalid inputs.

Let M be a polynomial-time PTM. Let us call an element $x \in \Sigma^*$ a *false positive* (of M with respect to L) if $x \notin L$, but $\Pr(M \text{ accepts } x) > 0$, and let us call x a *false negative* if $x \in L$ but $\Pr(M \text{ accepts } x) < 1$. The probabilities in those two cases are called the *error probabilities* of x.

We can characterize languages in NP, PP, BPP, and RP in terms of error probabilities as follows:

- $L \in \mathsf{BPP}$ if and only if there exists a polynomial-time PTM where both error probabilities are at most $1/3$.
- $L \in \mathsf{PP}$ if and only if there exists a polynomial-time PTM where both error probabilities are at most $1/2$.
- $L \in \mathsf{RP}$ if and only if there exists a polynomial-time PTM without false positives, and the error probability for false negatives is bounded by $1/3$.
- $L \in \mathsf{NP}$ if and only if there exists a polynomial-time PTM without false positives, but the error probability for false negatives is arbitrarily large.

Exercise 2.5 Show $\mathsf{BPP} = \mathsf{coBPP}$.

Solution Just exchange accepting and rejecting states. □

One could ask whether the choice of $\frac{1}{3}$ is special in the definition of BPP. Indeed, it turns out that there is nothing special about it, apart from $0 < \frac{1}{3} < \frac{1}{2}$.

Theorem 2.6 Let $0 < \varepsilon < \frac{1}{2}$ and let $L \subseteq \Sigma^*$ be such that there exists a polynomial-time PTM such that

$$\begin{aligned} x \in L &\implies \Pr(M \text{ accepts } x) \geq 1 - \varepsilon \\ x \notin L &\implies \Pr(M \text{ accepts } x) \leq \varepsilon. \end{aligned}$$

Then $L \in \mathsf{BPP}$.

The idea behind the proof is to repeat the computation of M for k times and accept if at least half of these computation accept. Indeed, choosing k such that

$$k > \frac{\ln 3}{2(\frac{1}{2} - \varepsilon)^2}$$

is sufficient for this to work. Moreover, instead of choose k as a constant it is also possible to choose it depending on the length of the actual input x: as long as k is chosen such that it depends polynomially on $|x|$, the resulting PTM will have polynomial running time. One can also show that the error probability drops exponentially with the number of repetitions. Therefore, PTMs can have error probabilities of $2^{-p(n)}$ for each polynomial $p(n)$. However, it is not possible to use this technique to obtain super-exponential error probability, as this would require a super-polynomial number of repetitions of the computation of M.

Exercise 2.7 Show $\mathsf{NP} \subseteq \mathsf{PP} \subseteq \mathsf{PSpace}$.

Solution The inclusion $\mathsf{PP} \subseteq \mathsf{PSpace}$ is clear: in polynomial space we can simulate all possible computations of a PTM and count whether there are more accepting than rejecting computations.

To see $\mathsf{NP} \subseteq \mathsf{PP}$, let $L \in \mathsf{NP}$. By Exercise 3 there exists a PTM N such that

$$L = \{\, x \in \Sigma^* \mid \Pr(N \text{ accepts } x) > 0 \,\}.$$

Define another PTM N' as follows: on input $(x, y_0y_1\ldots)$ we set

$$N(x, y_0y_1\ldots) := \begin{cases} \textbf{accept} & \text{if } y_0 = 1, \\ N(x, y_1\ldots) & \text{otherwise.} \end{cases}$$

Then

$$L = \{\, x \in \Sigma^* \mid \Pr(N' \text{ accepts } x) > 1/2 \,\}$$

and thus $L \in \mathsf{PP}$. □

A curiosity of BPP is that no complete problems are known for this class. This is rather surprising, because a common conjecture is $\mathsf{BPP} = \mathsf{P}$. One reason for the fact that no complete problems are known for BPP is that the word problem for probabilistic Turing machines with an error probability of $\frac{1}{3}$ is undecidable, because recognizing the corresponding machine model is already undecidable. It is also not clear how to remedy this fact by adding extra information (as in the case of polynomial-time Turing machines, where one can just add a bounding polynomial).

Exercise 2.8 (Optional) Let M be a polynomial-time probabilistic Turing machine. We say that M has *error probability* $< 1/3$ if and only if

$$\Pr(M \text{ accepts } w) \leq \frac{1}{3} \quad \text{or} \quad \Pr(M \text{ accepts } w) \geq \frac{2}{3}$$

for all inputs w. Show that deciding whether a polynomial-time probabilistic Turing machine (with known running time) has error probability $< 1/3$ is undecidable.

Solution We provide a many-one reduction from $\mathsf{A_{TM}}$. Let M be a Turing machine and let w be a valid input word. We shall construct a probabilistic PTM N such that N has error probability $< 1/3$ if and only if M does not accept w.

We define the machine N as follows. On input $(x, y_0y_1\ldots y_n)$ the machine rejects if $y_0 = 1$. Otherwise, it simulates the computation of M on input w for $|x|$ steps. If this simulation accepts, the machine accepts. Otherwise, it rejects.

Note that M runs in polynomial time (even linear time) in the size of x.

Suppose M accepts w, in say k steps. Then for inputs on length k and more the machine will accept with probability $1/2$ and reject with the same probability. Thus the error probability is not $< 1/3$. On the other hand, if M does not accept w, then N will never accept. Thus the error probability is below $1/3$.

Since N can be computed from M and w, we obtain a reduction from $\mathsf{A_{TM}}$ to the problem of recognizing polynomial-time probabilistic Turing machines with error probability $< 1/3$. This shows the claim. □