CS 6815: Pseudorandomness **Fall 2022**

Probabilistic Complexity Classes

Lecturer: Eshan Chattopadhyay *Aug 25, 2022*

1 Introduction

In computational complexity theory, there exists a family of randomized complexity classes for which decision problems are taken to be solved by a deterministic Turing machine given a polynomial number of random coin flips in the size of its input. The first notion of a probabilistic complexity class was **PP**, defined below:

Definition 1.1 (**PP**). *A language L is in* **PP** *if and only if there exists a probabilistic Turing machine M that runs in polynomial time in the length of its input x, for any input, such that:*

- $x \in L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] > 1/2$
- $x \notin L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] \leq 1/2.$

Since the probability that the Turing machine accepts an element of the language is so close the the probability it rejects an element not in the language, one cannot use an algorithm to amplify the certainty that they are working with a language in the complexity class **PP**. As a result, we consider more functional probabilistic complexity classes **BPP**, **RP**, and **coRP**, defined below:

Definition 1.2 (**BPP**). *A language L is in* **BPP** *if and only if there exists a probabilistic Turing machine M that runs in polynomial time in the length of its input x, for any input, such that:*

- $x \in L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] \geq 2/3$
- $x \notin L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] < 1/3.$

Definition 1.3 (**RP**). *A language L is in* **RP** *if and only if there exists a probabilistic Turing machine M that runs in polynomial time in the length of its input x, for any input, such that:*

- $x \in L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] \geq 1/2$
- $x \notin L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] = 0$

Definition 1.4 (**coRP**). *A language L is in* **coRP** *if and only if there exists a probabilistic Turing machine M that runs in polynomial time in the length of its input x, for any input, such that:*

- $x \in L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] = 1$
- $x \notin L \iff \Pr_{r \sim \{0,1\}^{poly(|x|)}}[M(x,r) = 1] \leq 1/2.$

These definitions are particularly useful because, for any language L in one of these complexity classes, we can now utilize the minimum distance between the probability of accepting an element in L and accepting an element not in L as a means to develop an algorithm that determines, with

high probability, determines whether an element is in L. We will see this more formally in the next section.

2 Error Reduction by Repetition

Interestingly, with the same computational resources (asymptotically), for some language L in a probabilistic complexity class where there is a non-negligible distance between the probability of accepting an element in L and accepting an element not in L, we can construct an algorithm that decides L with high certainty. Intuitively, if we run an algorithm that decides L with some non-negligible certainty given some random coins, if we run many trials of the algorithm independently and look at the collection of its outputs, we can amplify the certainty of our decision. Here we will show such a reduction for languages in **RP** and **BPP**

2.1 Error Reduction for RP

Say we have a language $L \in \mathbf{RP}$. Then there exists a probabilistic Turing machine M that decides L with error probability $1/2$ in polynomial time with respect to the length of its input given $|r| = \mathit{poly}(|x|)$ random bits. Well, what if we defined a new Turing machine M' that does the following computation:

$$M'(x, (r_1, r_2, \ldots, r_t)) = M(x, r_1) \vee M(x, r_2) \vee \ldots \vee M(x, r_t)$$

where each $r_i \in \{0,1\}^{\mathit{poly}(|x|)}$. Well, if t is polynomial with respect to $|x|$, then M' also runs in polynomial time with respect to length of its input and yet we will see its error probability has been reduced significantly. By the definition of **RP**, we have:

$$\begin{aligned} x \in L &\implies \Pr[M(x, r_1) = 1] = \ldots = \Pr[M(x, r_t) = 1] \geq 1/2 \\ &\implies \Pr[M'(x, (r_1, r_2, \ldots, r_t)) = 1] \geq 1 - 2^t \end{aligned}$$

and,

$$\begin{aligned} x \notin L &\implies \Pr[M(x, r_1) = 1] = \ldots = \Pr[M(x, r_t) = 1] = 0 \\ &\implies \Pr[M'(x, (r_1, r_2, \ldots, r_t)) = 1] = 0 \end{aligned}$$

implying we can guarantee an exponentially small error by having M' make $t = O(n)$ calls to M.

2.2 Error Reduction for BPP

The same can be done for **BPP**. Take an arbitrary language $L \in \mathbf{BPP}$. Then there exists a probabilistic Turing machine M that accepts elements of L with probability at least $2/3$ and accepts elements not in L with probability at most $1/3$ in polynomial time with respect to the length of its input given $|r| = \mathit{poly}(|x|)$ random bits. Again, we can define a new Turing machine M' that does the following computation:

$$M'(x, (r_1, r_2, \ldots, r_t)) = \text{Majority}(M(x, r_1), M(x, r_2), \ldots, M(x, r_t))$$

where each $r_i \in \{0,1\}^{p(|x|)}$. Well, again if t is polynomial with respect to $|x|$, then M' also runs in polynomial time with respect to length of its input and yet we will see its error probability has been

reduced significantly. Observe that the $M(x, r_i)$ are a bunch of identically distributed indicator random variables. Therefore, when $x \in L$, by linearity of expectation, we have

$$E\left[\sum_{i=1}^{t} M(x, r_i)\right] = \sum_{i=1}^{t} E[M(x, r_i)] \geq \frac{2t}{3}.$$

Now, using Chernoff's Bound, we can write:

$$\begin{aligned}
\Pr[M'(x, (r_1, r_2, \ldots, r_t)) = 1] &= \Pr[\text{Majority}(M(x, r_1), M(x, r_2), \ldots, M(x, r_t)) = 1] \\
&= 1 - \Pr\left[\left|\sum_{i=1}^{t} M(x, r_i) - E\left[\sum_{i=1}^{t} M(x, r_i)\right]\right| > \frac{t}{6}\right] \\
&\geq 1 - 2e^{-t/18}.
\end{aligned}$$

A very identical argument can be made for $x \notin L$ implying that again we can guarantee an exponentially small error by having M' make $t = O(n)$ calls to M.

3 Relating Probabilistic Complexity Classes to the Rest

It is easy to see a couple interesting relationships between probabilistic complexity classes and other complexity classes.

3.1 Relationships Between RP and Other Complexity Classes

The first trivial relationship is **P** is a subset of **RP** since you can construct a probabilistic Turing machine for any language $L \in \mathbf{P}$ that ignores the random bits it receives and acts deterministically on its input. So we have:

Theorem 3.1. $\mathbf{P} \subseteq \mathbf{RP}$

Furthermore, we can see that **RP** is a subset of **NP**. This is because for any language $L \in \mathbf{RP}$, there exists a probabilistic Turing machine such that there is at least one random string $r \in \{0,1\}^{poly(|x|)}$ such that $M(x, r) = 1$ if and only if $x \in L$. We can then use such an r as the certificate for any $x \in L$ and since there is no such r if $x \notin L$ then we know that we can construct a new deterministic Turing Machine that only accepts if it is given an $x \in L$ and its corresponding certificate r. So we have:

Theorem 3.2. $\mathbf{RP} \subseteq \mathbf{NP}$.

Therefore **RP** sits nicely in the first level of the Polynomial Hierarchy.

3.2 Relationships Between BPP and Other Complexity Classes

Interestingly, although we will not show this as it requires a long proof, there is also a relationship between **BPP** and the polynomial hierarchy, specifically:

Theorem 3.3. $\mathbf{BPP} \subseteq \mathbf{\Sigma_2^P} \cap \mathbf{\Pi_2^P}$.

The proof involves showing that a string $x \in \{0,1\}^n$ is in L if and only if there exists a polynomial number of strings $u_1, \ldots, u_t$ such that for all random strings $r \in \{0,1\}^m$ one can feed into the Turing Machine M which decides L, if we take $S_x = \{r' : M(x, r') = 1\}$, then we have $r \in \bigcup_{i=1}^{t}(u_i \oplus S_x)$.

Finally, **BPP** is a subset of **P/poly** which we formally show below:

Theorem 3.4. BPP $\subseteq$ P/poly.

Proof. Take $L \in \mathbf{BPP}$. By error reduction, we then have a Turing machine M with the property that

$$\Pr_{\substack{x\sim\{0,1\}^n \\ r\sim\{0,1\}^{poly(n)}}} [M(x,r) = L(x)] \geq 1 - 2^{-|x|^2}.$$

It then follows that there exists some r^* for which:

$$\Pr_{x\sim\{0,1\}^n}[M(x,r^*) \neq L(x)] \leq 2^{-|x|^2}$$

Now, this implies that

$$\Pr_{x\sim\{0,1\}^n}[M(x,r^*) \neq L(x)] = 0$$

since if $M(x,r^*)$ failed for even a single $x \in \{0,1\}^n$, then we'd have the probability that M failed over all x was at least 2^{-n}, a contradiction. So, we have shown a single r^* works for all $x \in L$. Therefore there exists a deterministic program which has r^* hard-coded into it that is always correct on any input x. If we convert the program into a polynomial sized circuit, then we have that $L \in \mathbf{P/poly}$ (note here r^* depends on $|x|$ and therefore we cannot use the same logic to say $\mathbf{BPP} \subseteq \mathbf{P}$).