Lecture 16: Randomized Computation

Arijit Bishnu

22.04.2010
Outline

1. Introduction
2. Probabilistic Turing Machine and the class BPP
3. One-Sided and Zero-Sided Error
4. Error Reduction for BPP
5. Relation of BPP with other classes
Introduction

Probabilistic Turing Machine and the class BPP

One-Sided and Zero-Sided Error

Error Reduction for BPP

Relation of BPP with other classes
Introduction to Randomized Algorithms and Probabilistic Turing Machines

- A randomized algorithm is an algorithm that is allowed access to a source of independent, unbiased, random bits. The algorithm is then permitted to use these random bits to influence its computation.
A randomized algorithm is an algorithm that is allowed access to a source of independent, unbiased, random bits. The algorithm is then permitted to use these random bits to influence its computation.

We want to study TMs that has the power to toss random coins.
Outline

1. Introduction
2. Probabilistic Turing Machine and the class BPP
3. One-Sided and Zero-Sided Error
4. Error Reduction for BPP
5. Relation of BPP with other classes
**Probabilistic Turing Machine**

**Definition: Probabilistic Turing Machine**

A Probabilistic Turing Machine (PTM) is a Turing machine with two transition functions $\delta_0, \delta_1$. To execute a PTM $M$ on an input $x$, we choose in each step with probability $1/2$ to apply $\delta_0$ and with probability $1/2$ to apply $\delta_1$. This choice is made independently of all previous choices.

The machine outputs ACCEPT (1) or REJECT (0). $M(x)$ denotes the output of $M$ on $x$ and surely this is a random variable.

For a function $T : \mathbb{N} \rightarrow \mathbb{N}$, we say that $M$ runs in $T(n)$-time if for any input $x$, $M$ halts on $x$ within $T(|x|)$ steps regardless of the random choices $M$ makes.
Interpretation of the Definition

- In a PTM, each transition is taken with probability 1/2.
Interpretation of the Definition

- In a PTM, each transition is taken with probability $1/2$.
- If the PTM $M$ has chosen the transition function $t$ times, then $M$ would have chosen any one of the $2^t$ branches with a probability of $\frac{1}{2^t}$.
Interpretation of the Definition

- In a PTM, each transition is taken with probability $1/2$.
- If the PTM $M$ has chosen the transition function $t$ times, then $M$ would have chosen any one of the $2^t$ branches with a probability of $\frac{1}{2^t}$.
- So how do we interpret $Pr[M(x) = 1]$?
Interpretation of the Definition

- In a PTM, each transition is taken with probability 1/2.
- If the PTM $M$ has chosen the transition function $t$ times, then $M$ would have chosen any one of the $2^t$ branches with a probability of $\frac{1}{2^t}$.
- So how do we interpret $Pr[M(x) = 1]$?
- It is simply the fraction of branches that end with $M$’s output of 1.
Interpretation of the Definition

- In a PTM, each transition is taken with probability 1/2.
- If the PTM $M$ has chosen the transition function $t$ times, then $M$ would have chosen any one of the $2^t$ branches with a probability of $\frac{1}{2^t}$.
- So how do we interpret $Pr[M(x) = 1]$?
- It is simply the fraction of branches that end with $M$’s output of 1.
- An NDTM accepts if $\exists$ one accepting branch; for a PTM, we consider the fraction of branches that leads to a 1.
A New Class: BPP

For a language $L \subseteq \{0, 1\}^*$ and an input $x \in \{0, 1\}^*$, we define $L(x) = 1$, if $x \in L$ and $L(x) = 0$, otherwise.
A New Class: BPP

- For a language $L \subseteq \{0, 1\}^*$ and an input $x \in \{0, 1\}^*$, we define $L(x) = 1$, if $x \in L$ and $L(x) = 0$, otherwise.

**Definition: Class BPP (Bounded Error Probabilistic Polynomial Time)**

For $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0, 1\}^*$ we say that a PTM $M$ decides $L$ in time $T(n)$ if for every $x \in \{0, 1\}^*$, $M$ halts in $T(|x|)$ steps irrespective of its random choices, and $\Pr[M(x) = L(x)] \geq \frac{2}{3}$, i.e.

$$\forall x \in L, \Pr[M \text{ accepts } x] \geq \frac{2}{3} \text{ and }$$

$$\forall x \notin L, \Pr[M \text{ rejects } x] \geq \frac{2}{3}.$$  

We let $\text{BPTIME}(T(n))$ be the class of languages decided by PTMs in $O(T(n))$ time and define $\text{BPP} = \bigcup_{c} \text{BPTIME}(n^c)$. 
Some Characteristics of the Definition

- The above PTM satisfies the **excluded middle property**. That is, the PTM either accepts or rejects every input with a prob. at least 2/3.
Some Characteristics of the Definition

- The above PTM satisfies the excluded middle property. That is, the PTM either accepts or rejects every input with a prob. at least $2/3$.

- For every input $x$, $M(x)$ will output the right value $L(x)$ with prob. at least $2/3$. The input $x$ can be the worst case input also.
Some Characteristics of the Definition

- The above PTM satisfies the *excluded middle property*. That is, the PTM either accepts or rejects every input with a prob. at least $\frac{2}{3}$.
- For every input $x$, $M(x)$ will output the right value $L(x)$ with prob. at least $\frac{2}{3}$. The input $x$ can be the worst case input also.
- The class BPP has *two-sided error* (what it is?).
Some Characteristics of the Definition

- The above PTM satisfies the excluded middle property. That is, the PTM either accepts or rejects every input with a prob. at least $\frac{2}{3}$.
- For every input $x$, $M(x)$ will output the right value $L(x)$ with prob. at least $\frac{2}{3}$. The input $x$ can be the worst case input also.
- The class BPP has two-sided error (what it is?).

<table>
<thead>
<tr>
<th>$x \notin L$</th>
<th>$x \in L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M(x) = 0$</td>
<td>$M(x) = 0$</td>
</tr>
<tr>
<td>$M(x) = 1$</td>
<td>$M(x) = 1$</td>
</tr>
</tbody>
</table>
An Alternate Definition

Definition: Class BPP

A language \( L \in \text{BPP} \) if there exists a poly-time TM \( M \) and a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) such that for every \( x \in \{0, 1\}^* \),
\[
\Pr_{r \in \{0, 1\}^{p(|x|)}}[M(x, r) = L(x)] \geq \frac{2}{3}
\]
where \( r \in R \) \( X \) denotes that \( r \) was chosen from the sample space \( X \).
An Alternate Definition

**Definition: Class BPP**

A language \( L \in \text{BPP} \) if there exists a poly-time TM \( M \) and a polynomial \( p : \mathbb{N} \rightarrow \mathbb{N} \) such that for every \( x \in \{0, 1\}^* \),
\[
\Pr_{r \in_R \{0, 1\}^{p(|x|)}}[M(x, r) = L(x)] \geq \frac{2}{3}
\]
where \( r \in_R X \) denotes that \( r \) was chosen from the sample space \( X \).

- We can interpret the above definition as giving to the deterministic TM a sequence of coin tosses for every step of its computation, apart from the input.
An Alternate Definition

**Definition: Class BPP**

A language $L \in \text{BPP}$ if there exists a poly-time TM $M$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \{0, 1\}^*$,

$$\Pr_{r \in_R \{0, 1\}^{|x|}}[M(x, r) = L(x)] \geq \frac{2}{3}$$

where $r \in_R X$ denotes that $r$ was chosen from the sample space $X$.

- We can interpret the above definition as giving to the deterministic TM a sequence of coin tosses for every step of its computation, apart from the input.

**Relations between Classes P, EXP and BPP**

$P \subseteq \text{BPP} \subseteq \text{EXP}$. 
An Alternate Definition

**Definition: Class BPP**

A language $L \in \text{BPP}$ if there exists a poly-time TM $M$ and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \{0, 1\}^*$, $\Pr_{r \in_R \{0,1\}^{|x|}}[M(x, r) = L(x)] \geq \frac{2}{3}$ where $r \in_R X$ denotes that $r$ was chosen from the sample space $X$.

- We can interpret the above definition as giving to the deterministic TM a sequence of coin tosses for every step of its computation, apart from the input.

**Relations between Classes P, EXP and BPP**

$P \subseteq \text{BPP} \subseteq \text{EXP}$.

**Proof**

Obvious.
Outline

1. Introduction
2. Probabilistic Turing Machine and the class BPP
3. One-Sided and Zero-Sided Error
4. Error Reduction for BPP
5. Relation of BPP with other classes
One-Sided Error

A PTM is said to have the one-sided error if for $x \not\in L$, $M(x) \neq 1$ but it may happen that for $x \in L$, $M(x) = 0$. 
One-Sided Error

- A PTM is said to have the one-sided error if for \( x \not\in L \), \( M(x) \neq 1 \) but it may happen that for \( x \in L \), \( M(x) = 0 \).

\[
\begin{array}{c|c|c|c}
 x \not\in L & x \in L \\
 M(x) = 0 & M(x) = 0 \\
\hline
 x \not\in L & x \in L \\
 M(x) = 1 & M(x) = 1 \\
\end{array}
\]

This error is not allowed.

So, \( \forall x \not\in L, \Pr[M(x) = 0] = 1 \)
One-Sided Error

Definition: Class RP

A language $L \subseteq \{0, 1\}^*$ is said to be in $\text{RTIME}(T(n))$ if there exists a PTM running in time $T(n)$ s.t.

$$\forall x \in L, \Pr[M(x) = 1] \geq \frac{2}{3}$$

$$\forall x \notin L, \Pr[M(x) = 0] = 1$$

The class $\text{RP} = \bigcup_{c > 0} \text{RTIME}(T(n))$. 
One-Sided Error

Definition: Class RP

A language $L \subseteq \{0, 1\}^*$ is said to be in \text{RTIME}(T(n)) if there exists a PTM running in time $T(n)$ s.t.

$$\forall x \in L, \Pr[M(x) = 1] \geq \frac{2}{3}$$

$$\forall x \not\in L, \Pr[M(x) = 0] = 1$$

The class $\text{RP} = \bigcup_{c>0} \text{RTIME}(T(n))$.

Observation

$\text{RP} \subseteq \text{NP}$ but we do not know whether $\text{BPP} \subseteq \text{NP}$. 
One-Sided Error: The Complement Class

Definition: Class coRP

\[ \text{coRP} = \{ L \mid \overline{L} \in \text{RP} \}. \]
One-Sided Error: The Complement Class

Definition: Class \( \text{coRP} \)

\[ \text{coRP} = \{ L \mid \overline{L} \in \text{RP} \} . \]

Definition: Class \( \text{coRP} \)

A language \( L \subseteq \{0, 1\}^* \) is said to be in \( \text{coRP} \) if there exists a PTM running in polynomial time s.t.

\[ \forall x \in L, \Pr[M(x) = 1] = 1 \]

\[ \forall x \notin L, \Pr[M(x) = 0] \geq \frac{2}{3} \]
Outline

1. Introduction
2. Probabilistic Turing Machine and the class BPP
3. One-Sided and Zero-Sided Error
4. Error Reduction for BPP
5. Relation of BPP with other classes
Error Reduction

**Theorem**

Let $L \subseteq \{0, 1\}^*$ be a language and suppose there exists a poly-time PTM $M$ such that for every $x \in \{0, 1\}^*$,

$$\Pr[M(x) = L(x)] \geq \frac{1}{2} + |x|^{-c}.$$  
Then, for every constant $d > 0$, \exists a poly-time PTM $M'$ such that for every $x \in \{0, 1\}^*$,

$$\Pr[M'(x) = L(x)] \geq 1 - 2^{-|x|^d}.$$
Error Reduction

Theorem

Let $L \subseteq \{0, 1\}^*$ be a language and suppose there exists a poly-time PTM $M$ such that for every $x \in \{0, 1\}^*$,

$$\Pr[M(x) = L(x)] \geq \frac{1}{2} + \left| |x|^{-c} \right.$$. Then, for every constant $d > 0$, $\exists$ a poly-time PTM $M'$ such that for every $x \in \{0, 1\}^*$,

$$\Pr[M'(x) = L(x)] \geq 1 - 2^{-|x|^d}$$.

Proof
Error Reduction

Theorem

Let $L \subseteq \{0, 1\}^*$ be a language and suppose there exists a poly-time PTM $M$ such that for every $x \in \{0, 1\}^*$,

\[ \Pr[M(x) = L(x)] \geq \frac{1}{2} + |x|^{-c}. \]

Then, for every constant $d > 0$, there exists a poly-time PTM $M'$ such that for every $x \in \{0, 1\}^*$,

\[ \Pr[M'(x) = L(x)] \geq 1 - 2^{-|x|^d}. \]

Proof

$M'$ does the following. For every input $x \in \{0, 1\}^*$, run $M(x)$ for $k = \text{poly}(|x|)$ times obtaining $k$ outputs $y_1, \ldots, y_k \in \{0, 1\}$. If the majority of these outputs is 1, then $M'$ outputs 1, else $M'$ outputs 0.
Error Reduction

**Theorem**

Let $L \subseteq \{0, 1\}^*$ be a language and suppose there exists a poly-time PTM $M$ such that for every $x \in \{0, 1\}^*$,

$$\Pr[M(x) = L(x)] \geq \frac{1}{2} + |x|^{-c}.$$  

Then, for every constant $d > 0$, there exists a poly-time PTM $M'$ such that for every $x \in \{0, 1\}^*$,

$$\Pr[M'(x) = L(x)] \geq 1 - 2^{-|x|^d}.$$

**Proof**

- $M'$ does the following. For every input $x \in \{0, 1\}^*$, run $M(x)$ for $k = \text{poly}(|x|)$ times obtaining $k$ outputs $y_1, \ldots, y_k \in \{0, 1\}$. If the majority of these outputs is 1, then $M'$ outputs 1, else $M'$ outputs 0.

- Now, use Chernoff bounds to fix the error probability as mentioned in the theorem.
Outline

1. Introduction
2. Probabilistic Turing Machine and the class BPP
3. One-Sided and Zero-Sided Error
4. Error Reduction for BPP
5. Relation of BPP with other classes
BPP and $\mathbb{P}/\text{poly}$

**Theorem**

$\text{BPP} \subseteq \mathbb{P}/\text{poly}$. 

Proof

Suppose $L \in \text{BPP}$. There exists a TM $M$ that on inputs of size $n$ uses $m$ random bits such that for every $x \in \{0,1\}^*$, $\Pr_r [M(x,r) \neq L(x)] \leq 2^{-n-1}$.

A random bit string $r \in \{0,1\}^m$ is good for an input $x$ if it leads to $M(x,r) = L(x)$; else $r$ is bad. $m = \text{poly}(n)$.

For every $x$, at most $2^{m - 1}$ strings are bad for $x$.

Add over all $x \in \{0,1\}^n$ to have at most $2^{m - 1}$ strings $r$ that are bad for some $x$. 
Theorem

\[ \text{BPP} \subseteq \text{P/poly}. \]

Proof
BPP and $\text{P}/\text{poly}$

**Theorem**

$\text{BPP} \subseteq \text{P}/\text{poly}$. 

**Proof**

- Suppose $L \in \text{BPP}$. 

BPP and \( \mathbb{P} / \text{poly} \)

**Theorem**

\[ \text{BPP} \subseteq \mathbb{P} / \text{poly}. \]

**Proof**

- Suppose \( L \in \text{BPP} \).
- \( \exists \) a TM \( M \) that on inputs of size \( n \) uses \( m \) random bits such that for every \( x \in \{0, 1\}^* \), \( \Pr_r [M(x, r) \neq L(x)] \leq 2^{-n-1} \).
BPP and \( P_{/poly} \)

**Theorem**

\[ \text{BPP} \subseteq P_{/poly}. \]

**Proof**

- Suppose \( L \in \text{BPP} \).
- \( \exists \) a TM \( M \) that on inputs of size \( n \) uses \( m \) random bits such that for every \( x \in \{0, 1\}^* \), \( \Pr_r[M(x, r) \neq L(x)] \leq 2^{-n-1} \).
- A random bit string \( r \in \{0, 1\}^m \) is **good** for an input \( x \) if it leads to \( M(x, r) = L(x) \); else \( r \) is **bad**. \( m = \text{poly}(n) \).
BPP and $P_{/\text{poly}}$

**Theorem**

$\text{BPP} \subseteq P_{/\text{poly}}$.

**Proof**

- Suppose $L \in \text{BPP}$.
- $\exists$ a TM $M$ that on inputs of size $n$ uses $m$ random bits such that for every $x \in \{0,1\}^*$, $\Pr_r[M(x,r) \neq L(x)] \leq 2^{-n-1}$.
- A random bit string $r \in \{0,1\}^m$ is good for an input $x$ if it leads to $M(x,r) = L(x)$; else $r$ is bad. $m = \text{poly}(n)$.
- For every $x$, at most $\frac{2^m}{2^{n+1}}$ strings are bad for $x$. 
Theorem

BPP ⊆ P/poly.

Proof

- Suppose \( L \in \text{BPP} \).
- \( \exists \) a TM \( M \) that on inputs of size \( n \) uses \( m \) random bits such that for every \( x \in \{0, 1\}^* \), \( \Pr_r[M(x, r) \neq L(x)] \leq 2^{-n-1} \).
- A random bit string \( r \in \{0, 1\}^m \) is good for an input \( x \) if it leads to \( M(x, r) = L(x) \); else \( r \) is bad. \( m = \text{poly}(n) \).
- For every \( x \), at most \( \frac{2^m}{2^{n+1}} \) strings are bad for \( x \).
- Add over all \( x \in \{0, 1\}^n \) to have at most \( 2^{m-1} \) strings \( r \) that are bad for some \( x \).
BPP and $P_{/\text{poly}}$

Proof

The bound on bad strings imply, $\exists$ a string $r_0 \in \{0, 1\}^m$ that is good for every $x \in \{0, 1\}^n$. Now, $M(x, r)$ is computable in poly-time. So, $\exists$ a poly-sized circuit family $\{C_n\}_{n \in \mathbb{N}}$ where $C_n$ simulates $M(x, r)$ correctly on inputs $<x, r>$ of length $i$. To obtain the circuits, hard-code $r_0$ obtained as above. So, we obtain circuits $C'_1, C'_2, ...$ where $C'_n = C_n + \text{poly}(n)(x, r_0)$ recognizing the language $L$. 
**BPP and $\mathbb{P}^{/\text{poly}}$**

**Proof**

- The bound on bad strings imply, $\exists$ a string $r_0 \in \{0, 1\}^m$ that is good for every $x \in \{0, 1\}^n$. 
BPP and $P_{/\text{poly}}$

**Proof**

- The bound on bad strings imply, $\exists$ a string $r_0 \in \{0,1\}^m$ that is good for every $x \in \{0,1\}^n$.

- Now, $M(x, r)$ is computable in poly-time. So, $\exists$ a poly-sized circuit family $\{C_n\}_{n \in \mathbb{N}}$ where $C_n$ simulates $M(x, r)$ correctly on inputs $<x, r>$ of length $i$. 
BPP and $P_{/\text{poly}}$

**Proof**

- The bound on bad strings imply, $\exists$ a string $r_0 \in \{0, 1\}^m$ that is good for every $x \in \{0, 1\}^n$.

- Now, $M(x, r)$ is computable in poly-time. So, $\exists$ a poly-sized circuit family $\{C_n\}_{n \in \mathbb{N}}$ where $C_n$ simulates $M(x, r)$ correctly on inputs $\langle x, r \rangle$ of length $i$.

- To obtain the circuits, hard-code $r_0$ obtained as above.
BPP and $P_{/ \text{poly}}$

**Proof**

- The bound on bad strings imply, $\exists$ a string $r_0 \in \{0, 1\}^m$ that is good for every $x \in \{0, 1\}^n$.

- Now, $M(x, r)$ is computable in poly-time. So, $\exists$ a poly-sized circuit family $\{C_n\}_{n \in \mathbb{N}}$ where $C_n$ simulates $M(x, r)$ correctly on inputs $< x, r >$ of length $i$.

- To obtain the circuits, hard-code $r_0$ obtained as above.

- So, we obtain circuits $C'_1, C'_2, \ldots$ where $C'_n = C_{n+\text{poly}(n)}(x, r_0)$ recognizing the language $L$. 
Another Relation: Is BPP in PH?

Theorem

\[ \text{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p \]