Probabilistic Complexity

S. Soroush H. Zargarbashi
s.zargarbashi@ut.ac.ir

University of Tehran

February 4, 2019

Preliminaries and Motivation

Recall From Complexity

Definition

- **Turing Machine**
- **Class P**
- **Class NP**
- **Class co-NP**
- **Class** Σ_2^p

Example of P and NP

- SAT
- k-SAT
- 2-SAT

Probabilistic Algorithm

- An algorithm that uses the outcome of a random process during its computation is called **Randomized Algorithm**.

Probabilistic Algorithm

- An algorithm that uses the outcome of a random process during its computation is called **Randomized Algorithm**.
- Randomized Algorithms can be formalized via **Probabilistic Turing Machine**

Probabilistic Algorithm

- An algorithm that uses the outcome of a random process during its computation is called **Randomized Algorithm**.
- Randomized Algorithms can be formalized via **Probabilistic Turing Machine**

Definition

Probabilistic Turing Machine is a Turing machine with two transition functions δ_0 δ_1.
To execute a PTM M on an input x, we choose in each step with probability $1/2$ to apply the transition function δ_0 and with probability $1/2$ to apply δ_1. This choice is made independently of all previous choices.

Example

Determinant of Polynomial Matrix

Given an $m \times m$ matrix $\mathcal{Q} = [Q_{ij}]$ of n-variable polynomials written in normal form, and a Q_0 as n-variable polynomial, decide whether or not $\det(\mathcal{Q}) = Q_0$
[DK11]

Helpful Fact!

Determinant of an $m \times m$ matrix over integers in $\{-k, ..., k\}$ can be computed in $\mathcal{O}(m + \log k)$.

Example

Algorithm

Input: $\mathcal{Q} = [Q_{i,j}]$, ϵ
$d' \leftarrow \max\{$degree of $Q_{i,j}, 1 \leq i, j \leq m\}$;
$d_0 \leftarrow$ degree of Q_0;
$d \leftarrow \max\{md', d_0\}$;

Example

Algorithm

Input: $\mathcal{Q} = [Q_{i,j}]$, ϵ
$d' \leftarrow \max\{$degree of $Q_{i,j}, 1 \leq i, j \leq m\}$;
$d_0 \leftarrow$ degree of Q_0;
$d \leftarrow \max\{md', d_0\}$;
$k \leftarrow \lceil -\log \epsilon \rceil$ **for** $i : 1 \rightarrow k$ **do**
    assign $(u_1, ..., u_n)$ randomly each in range $\{-d, ..., d\}$;
    if $\det(\mathcal{Q}(u)) \neq Q_0(u)$ **then**
        return **No**;
    end
end
return **Yes**

Example

Theorem

Let Q be an n-variable integer polynomial of degree d that is not identical to zero. Let $m \geq 0$ and $A_{n,m} = \{(u_1, ..., u_n) \mid \forall 1 \leq i \leq n : |u_i| \leq m\}$. Then, Q has at most $d.(2m+1)^{n-1}$ zeros $(u_1, ..., u_n)$ in $A_{n,m}$.

- If $\det(\mathcal{Q}) = Q_0$ the algorithm always returns **Yes**.
- Otherwise, algorithm returns **No** with probability $\geq 1 - \epsilon$

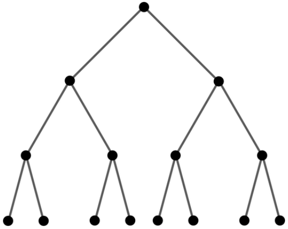
Probabilistic Turing Machine

- $M(x)$: Random variable of what the Turing Machine writes at the end of its computation.
- Time complexity of the Turing Machine is the number of steps it takes at most regardless of the random choice.

Probabilistic Turing Machine

- $M(x)$: Random variable of what the Turing Machine writes at the end of its computation.
- Time complexity of the Turing Machine is the number of steps it takes at most regardless of the random choice.

- Computation can be seen as a tree. Like NDTM
- The probability of each computation path is 2^{-m} where m is the number of random bits used in the computation.



Polynomial Probability

Probabilistic Polynomial Class

Definition

The probabilistic polynomial (PP) class is the set of languages L for which there is a PTM M running in poly-time such that for all words x,

- If $x \in L$ then $\Pr[M(x) = 1] > \frac{1}{2}$
- If $x \notin L$ then $\Pr[M(x) = 0] > \frac{1}{2}$

[LO]

- Other equivalent definition includes $\Pr[M(x) = L(x)] > \frac{1}{2}$

PP Class: Example

$\frac{1}{2}$-SAT

Given a boolean formula $F : \{0,1\}^n \to \{0,1\}$ in conjunctive normal form, decide if more that half of assignments to $x_1, ..., x_n$ are satisfying F. [PS12]

PP Class: Example

$\frac{1}{2}$-SAT

Given a boolean formula $F : \{0, 1\}^n \to \{0, 1\}$ in conjunctive normal form, decide if more that half of assignments to $x_1, ..., x_n$ are satisfying F. [PS12]

- Obviously problem is NP-Hard.

Proposition

There is a polynomial time algorithm which returns the correct answer with probability at least $\frac{1}{2}$.

PP Class: Example

Proof.

Choose an assignment x independently at random. If x satisfies the formula return "Yes" Otherwise "No" Assume that the correct answer is "Yes". This implies

$$\Pr[M(x) = 1] = \frac{\text{\# of satisfying assignment}}{2^n} > \frac{1}{2}$$

and

$$\Pr[M(x) = 1] = \frac{\text{\# of satisfying assignment}}{2^n} > \frac{1}{2}$$

□

Where is PP?

Theorem

- $P \subseteq PP$
- $NP \subseteq PP$

[LO]

Where is PP?

Theorem

- $P \subseteq PP$
- $NP \subseteq PP$

[LO]

Proof.

First item: is trivial as any polynomial deterministic Turing machine is a probabilistic Turing machine with $\Pr[M(x) = L(x)] = 1$

Where is PP?

Theorem

- $P \subseteq PP$
- $NP \subseteq PP$

[LO]

Proof.

First item: is trivial as any polynomial deterministic Turing machine is a probabilistic Turing machine with $\Pr[M(x) = L(x)] = 1$
Second item: What we know is that a NP-Machine is equivalent to a PTM, computing correct acceptance with probability $\frac{1}{2^{p(n)}}$ and correct rejection with probability 1.

$\square$

Where is PP?

Continuation of Proof.

Claim: The probability of correct acceptance in an NP-machine M (deciding $L(M)$ in $p(|x|)$ as p is polynomial) can be increased to $1/2$ in polynomial time.

With assumption that M is balanced over alphabet $\{0,1\}$ we construct PTM M' as performing following algorithm:

- Accept with probability

$$\frac{1}{2} - \frac{1}{4 \times 2^{p(|x|)}}$$

- If not accepted at first step, then create a random string $w \in \{0,1\}^{p(|x|)}$ and if route w in M accepts x then accept.
- Otherwise, Reject

$\square$

Where is PP?

Continuation of Proof.

- If $x \in L$ then at least one computation branch has accept state and the probability of acceptance will be

$$\Pr[M(x) = 1] = \frac{1}{2} - \frac{1}{4 \times 2^{p(|x|)}} + \frac{1}{2 \times 2^{p(|x|)}} > \frac{1}{2}$$

- If $x \notin L$ then, the probability of rejection is

$$\Pr[M(x) = 0] = \frac{1}{2} - \frac{1}{4 \times 2^{p(|x|)}} < \frac{1}{2}$$

□

Where is PP?

Theorem

$PP \subseteq PSPACE$ *[PS12]*

Proof.

It's by enumeration on all random choices and aggregating the result. $\square$

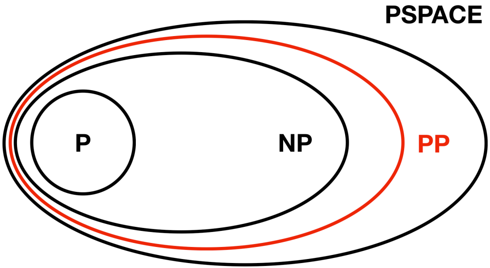
Where is PP?

Theorem

$PP \subseteq PSPACE$ [PS12]

Proof.

It's by enumeration on all random choices and aggregating the result. $\square$



Bounded Error Probabilistic Polynomial

Bounded Probabilistic Polynomial Time Class

Definition

The Bounded probabilistic polynomial (BPP) class is the set of languages L for which there is a PTM M running in poly-time and a constant $0 \leq \epsilon \leq \frac{1}{2}$ such that for all words x,

- If $x \in L$ then $\Pr[M(x) = 1] \geq \frac{1}{2} + \epsilon$
- If $x \notin L$ then $\Pr[M(x) = 1] \leq \frac{1}{2} - \epsilon$

[PS12]

Bounded Probabilistic Polynomial Time Class

Definition

The Bounded probabilistic polynomial (BPP) class is the set of languages L for which there is a PTM M running in poly-time and a constant $0 \leq \epsilon \leq \frac{1}{2}$ such that for all words x,

- If $x \in L$ then $\Pr[M(x) = 1] \geq \frac{1}{2} + \epsilon$
- If $x \notin L$ then $\Pr[M(x) = 1] \leq \frac{1}{2} - \epsilon$

[PS12]

Other Definition

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

Equivalence of BPP Definitions

Theorem

Error reduction for BPP

*Let L be a language for which there is a PTM M, deciding L with probability $\Pr[M(x) = L(x)] \geq \frac{1}{2} + \epsilon$ for some constant $0 \leq \epsilon \leq \frac{1}{2}$, Then for every $0 \leq \delta \leq \frac{1}{2}$ there exists a PTM M' such that $\Pr[M'(x) = L(x)] \geq \frac{1}{2} + \delta$
Also the time complexity of the M' is $T(M(x)).\mathcal{O}(p(n))$ for p as a polynomial function. [AB09]*

Equivalence of BPP Definitions

Sketch of Proof.

Set $c := -\log(\epsilon)$ and $d := -\log(\delta)/\log|x|$
Also set $d := -\log(delta)$
What M does is: for every input $x \in \{0,1\}^*$, run $M(x)$ for $k = 8|x|^{2c+d}$ times obtaining k outputs $y_1, ..., y-k \in 0, 1$. If the majority of these outputs is 1, then output 1; otherwise, output 0. □

One Sided Error Polynomial Time

Definition

One Sided Error Randomized Polynomial (RP) is the class of languages that are decided by probabilistic polynomial time Turing machines where inputs in the language are accepted with a probability of at least $1/2$, and inputs not in the language are rejected with a probability of 1. [Sip06]

One Sided Error Polynomial Time

Definition

One Sided Error Randomized Polynomial (RP) is the class of languages that are decided by probabilistic polynomial time Turing machines where inputs in the language are accepted with a probability of at least $1/2$, and inputs not in the language are rejected with a probability of 1. [Sip06]

Theorem

If L is a language in RP, then for every polynomial q there is a PTM M such that

- *If $x \in L$ then $\Pr[M(x) = 1] \geq 1 - 2^{-q(|x|)}$*
- *If $x \notin L$ then $\Pr[M(x) = 1] = 0$*

By calling the RP algorithm $q(|x|)$ [PS12]

One Sided Error Polynomial Time

Definition

co-RP Class

- $co - RP := \{L \mid \bar{L} \in RP\}$
- s the class of languages that are decided by probabilistic polynomial time Turing machines where inputs in the language are accepted with a probability 1 , and inputs not in the language are rejected with a probability of at least $1/2$.

One Sided Error Polynomial Time

Definition

co-RP Class

- $co - RP := \{L \mid \bar{L} \in RP\}$
- s the class of languages that are decided by probabilistic polynomial time Turing machines where inputs in the language are accepted with a probability 1 , and inputs not in the language are rejected with a probability of at least $1/2$.

Theorem

The above definitions are equivalent.

Zero Sided Error Polynomial Time

Definition

ZPP Class, contains all the languages L for which there is a machine M that runs in an expected-time $\mathcal{O}(T(n))$ such that for every input x, whenever M halts on x, the output $M(x)$ it produces is exactly $L(x)$, where $T(n)$ is polynomial w.r.t. n. [AB09]

Zero Sided Error Polynomial Time

Definition

ZPP Class, contains all the languages L for which there is a machine M that runs in an expected-time $\mathcal{O}(T(n))$ such that for every input x, whenever M halts on x, the output $M(x)$ it produces is exactly $L(x)$, where $T(n)$ is polynomial w.r.t. n. [AB09]

Theorem

$ZPP = RP \cap co - RP$

Sketch of Proof.

Create a new machine M which iteratively i steps from each RP and $co - RP$ algorithms. $\square$

Probabilistic Polynomials Inclusions

RP, co-RP, ZPP and PP

Theorem

- $RP \subseteq NP$
- $co - RP \subseteq co - NP$

RP, co-RP, ZPP and PP

Theorem

- $RP \subseteq NP$
- $co - RP \subseteq co - NP$
- $P \subseteq ZPP$

RP, co-RP, ZPP and PP

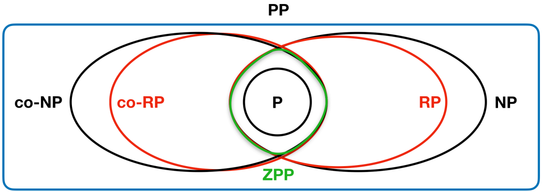
Theorem

- $RP \subseteq NP$
- $co - RP \subseteq co - NP$
- $P \subseteq ZPP$
- $co - NP \subseteq PP$ *(Same as theorem for $NP \subseteq PP$)*

RP, co-RP, ZPP and PP

Theorem

- $RP \subseteq NP$
- $co - RP \subseteq co - NP$
- $P \subseteq ZPP$
- $co - NP \subseteq PP$ *(Same as theorem for* $NP \subseteq PP$*)*



Where is BPP?

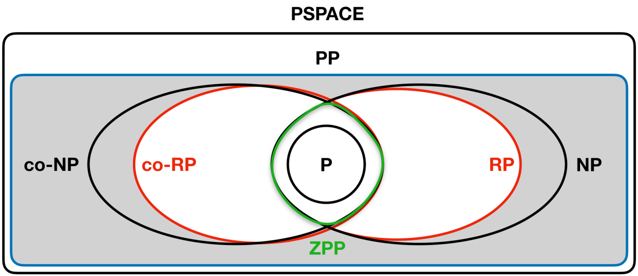
Theorem

- $RP \subseteq BPP$
- $co - RP \subseteq BPP$

Where is BPP?

Theorem

- $RP \subseteq BPP$
- $co - RP \subseteq BPP$
- *Items above imply that* $P \subseteq BPP$
- $BPP \subseteq PP$



Where is BPP?

Lemma

$BPP = co - BPP$ in other words means that if a language L is in BPP then $\bar{L}$ is also in BPP.

Where is BPP?

Lemma

$BPP = co - BPP$ in other words means that if a language L is in BPP then $\bar{L}$ is also in BPP.

Theorem

$BPP \subseteq \Sigma_2^p \cap \Pi_2^p$

Proof.

[AB09] First we prove that $BPP \subseteq \Sigma_2^p$ and the lemma above shows that $BPP \subseteq \Pi_2^p$, $\square$

Where is BPP?

Continuation of Proof.

Suppose $L \in BPP$ means that there is a polynomial time deterministic Turing machine for L that on input of length n uses $m = poly(n)$ random bits and by amplification theorem,

$$x \in L \Rightarrow \Pr_r[M(x,r) accepts] \geq 1 - 2^{-n}$$
$$x \notin L \Rightarrow \Pr_r[M(x,r) accepts] \leq 2^{-n}$$

Where is BPP?

Continuation of Proof.

Suppose $L \in BPP$ means that there is a polynomial time deterministic Turing machine for L that on input of length n uses $m = poly(n)$ random bits and by amplification theorem,

$$x \in L \Rightarrow \Pr_r[M(x,r)accepts] \geq 1 - 2^{-n}$$
$$x \notin L \Rightarrow \Pr_r[M(x,r)accepts] \leq 2^{-n}$$

For $x \in \{0,1\}^n$ define S_x as the set of all random rs that M accepts $\langle x, r \rangle$ then

$$x \in L \Rightarrow |S_x| \geq (1 - 2^{-n})2^m$$
$$x \notin L \Rightarrow |S_x| \leq 2^{-n}2^m$$

$\square$

Where is BPP?

Proof.

Defining shift operator as $S + u = \{x \oplus u \mid x \in S\}$ if $S \subseteq \{0,1\}^m$, $u \in \{0,1\}^m$ we **claim** that $x \in L$ if and only if

$$\exists u_1, ..., u_k \in \{0,1\}^m \forall r \in \{0,1\}^m r \in \cup_{i=1}^k (S_x + u_i)$$

which means

$$\exists u_1, ..., u_k \in \{0,1\}^m \forall r \in \{0,1\}^m \vee_{i=1}^k M(x, r \oplus u_i) accepts$$

For $k = \lceil \frac{m}{n} \rceil + 1$ Which is the definition of Σ_2^p. □

Where is BPP?

Proof of the Claim.

Following items implies the claim:

- For every set $S \in \{0,1\}^m$ that $|S| \leq 2^{n-m}$ and every k vectors $u_1, ..., u_k$ we have
$$\cup_{i=1}^{k}(S + u_i) \neq \{0,1\}^m$$
Use union bound.

Where is BPP?

Proof of the Claim.

Following items implies the claim:

- For every set $S \in \{0,1\}^m$ that $|S| \leq 2^{n-m}$ and every k vectors $u_1, ..., u_k$ we have
$$\cup_{i=1}^{k}(S + u_i) \neq \{0,1\}^m$$
Use union bound.
- For every set $S \in \{0,1\}^m$ that $|S| \geq (1 - 2^{-n})2^m$ and there exist k vectors $u_1, ..., u_k$ that we have
$$\cup_{i=1}^{k}(S + u_i) = \{0,1\}^m$$
Use union bound for complement and show $\Pr[\cup_{i=1}^{k}(S + u_i) = \{0,1\}^m] > 0$

□

BPP vs NP

Theorem

If $NP \subseteq BPP$ *then* $NP = RP$.
[DK11]

References

- Sanjeev Arora and Boaz Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.
- Ding-Zhu Du and Ker-I Ko, *Theory of computational complexity*, vol. 58, John Wiley & Sons, 2011.
- Alejandro López-Ortiz, *Probabilistic complexity classes*, Master's thesis, Citeseer.
- Hans Jürgen Prömel and Angelika Steger, *The steiner tree problem: a tour through graphs, algorithms, and complexity*, Springer Science & Business Media, 2012.
- Michael Sipser, *Introduction to the theory of computation*, vol. 2, Thomson Course Technology Boston, 2006.