Complexité avancée - TD 9

Rémy Poulain

20 novembre 2019

Exercise 0 : Complexity and complexity : RP^*

RP^* is the class of all languages L for which there exists a probabilistic Turing machine M running in time polynomial, such that :

— If $x \in L$ then $Pr[M(x,r) = \bot] < 1$

— If $x \notin L$ then $Pr[M(x,r) = \top] = 0$

Do you recognize this class ?

Solution:

It's NP.

— $\mathsf{RP}^* \subseteq \mathsf{NP}$: Exactly the same proof that $\mathsf{RP} \subseteq \mathsf{NP}$

— $\mathsf{NP} \subseteq \mathsf{RP}^*$: Let M be the PTM that, on ϕ a formulae with p free variable, and r a random tape (of length $> p$), evaluates ϕ on r. Then M is obviously polynomial and :

— If $\phi \in \mathsf{SAT}$ then $Pr[M(\phi,r) = \bot] = 1 - eval(\phi)$.
Where $eval(\phi)$ is the proportion of evaluations that satisfy ϕ. There is at list 1 evaluation that satisfies ϕ, then $Pr[M(\phi,r) = \bot] < 1$.

— If $\phi \notin \mathsf{SAT}$ then $Pr[M(\phi,r) = \top] = 0$

Then $SAT \in \mathsf{RP}^*$ Obviously RP^* is closed by polynomial reduction. Therefore $NP \subseteq \mathsf{RP}^*$. ■

Exercise 1 : Alternative definition of probabilistic TM

1. Show that, for $K = 2$, the notion of probabilistic Turing machines defined below is equivalent to the one given in introduction.

   A PTM is a non-deterministic Turing machine M with a fixed degree of non-determinism $K \geq 2$ (*i.e.* in each configuration, the machine has exactly K possible choices, not necessarily all bringing to distinct configurations).

   We associate a probability $1/K$ to each non-deterministic choice. In other words, in each configuration the machine chooses with equal probability which transition to follow, among the possible ones. The choices of the machine at any two different steps are assumed independent.

   To a run R of M, we assign the probability $Pr[R]$ that the machine makes a sequence of choices that produces the run R. The probability that M accepts the input x is given by :

   $$Pr[M(x) = \top] = \sum_{R \text{ accepting run of } M \text{ on } x} Pr[R]$$

   and the probability that it rejects x is $Pr[M(x) = \bot] = 1 - Pr[M(x) = \top]$.

   The class $\mathsf{RTIME}'(f(n), f(n), accerr(n), rejerr(n))$ is defined as the class of languages L for which there exists a PTM M running in time $f(n)$, such that : if $x \in L$ then $Pr[M(x) = \bot] \leq rejerr(n)$, and if $x \notin L$ then $Pr[M(x) = \top] \leq accerr(n)$.

2. In the lecture, we have restricted our attention to probabilistic Turing machines with random-tape alphabet $\Sigma = \{0,1\}$. Show that in the simple case of RP, this is not a restriction. Only describe the equivalence between a 2-symbol alphabet and a 3-symbol alphabet : $\mathsf{RP}^{\{0,1\}} = \mathsf{RP}^{\{0,1,2\}}$.

3. bonus

   One can also show that assuming uniform probability is not a restriction. More precisely, consider the following third variant of probabilistic Turing machines :

   Given $\rho \in]0,1[$, a ρ-PTM is a PTM of degree of non-determinism 2, but such that the first choice is made with probability ρ and the second one with probability $1-\rho$. Assuming ρ is computable in polynomial time, that is one can compute the i-th bit of ρ in time i^k for some constant k, show that :

   (a) A ρ-PTM can be simulated by a PTM in expected time $O(1)$.
   Hint : pick a number in]0,1[at random

   (b) Conversely, a PTM can be simulated by a ρ-PTM in expected time $O(\frac{1}{\rho(1-\rho)})$
   Hint : pick pairs (r_1, r_2) of random bits until $r_1 \neq r_2$

   Curiosity : if ρ is not computable, then a ρ-PTM can compute indecidable languages.

Solution:

Idea :

1. Simulate the choice of one transition by reading a letter and reciprocally.
2. The idea is to use the exercice 1 of the first TD, we don't need to be in polynomial time but just in expected running time. So to prove that $\mathsf{RP}^{\{0,1\}} \subseteq \mathsf{RP}^{\{0,1,2\}}$ you just need to read the next letter if the letter is 2.

Proof :

1. The idea is sufficient
2. $RP^{\{0,1\}} \subseteq RP^{\{0,1,2\}}$: Let $L \in \mathsf{RP}^{\{0,1\}}$, and M which decides L in time p, and let M' which simulates it on the alphabet $\{0,1,2\}$, for each step, if it reads a 2, it just read the next letter.
   For x an input and r in $\{0,1,2\}^*$, $T_{M'}(x,r) = T_M(x, w_2(r)) + |r|_2$, where $w_2(r)$ is r without 2 and $|r|_2$ is the number of 2 in r. For all x, $E[T_{M'}(x,r)] \leq p(|x|) + \frac{p(|x|)}{3}$, and $P[T_{M'}(x,r) = \infty] = 0$ (this is the probability of r to be in $\{0,1,2\}^* 2^\omega$). So according to the previous TD, $L \in \mathsf{RP}^{\{0,1,2\}}$. ■

   $RP^{\{0,1,2\}} \subseteq RP^{\{0,1\}}$: Reciprocally we do exactly the same thing, reading two letters every step and if it's 11 reading the next two letters.

Exercise 2 : BPP and oracle machines

Prove that $\mathsf{P}^{\mathsf{BPP}} = \mathsf{BPP}$.

Solution:

Idea : To show that $\mathsf{P}^{\mathsf{BPP}} \subseteq \mathsf{BPP}$ we need to simulate all calls to the oracle BPP by our language BPP, so the difficulty is in the duplication of random words. We have already seen this in the exercice with the self-reduction. The main idea is to use error reduction on the oracle and not on our built language.

Bad idea : Care if you say something like : $\mathsf{BPP}^{\mathsf{BPP}} \subseteq \mathsf{BPP}$ that's true here but that's not obvious, think about $\mathsf{NP}^{\mathsf{NP}}$ and NP.

Proof (scheme) :
$\mathsf{BPP} \subseteq \mathsf{P}^{\mathsf{BPP}}$: Obvious, cause you can just ask to the oracle the answer.

$\mathsf{BPP} \subseteq \mathsf{P}^{\mathsf{BPP}}$: Let $L \in \mathsf{P}^{\mathsf{BPP}}$, by definition $\exists B \in \mathsf{BPP}$ an oracle and M a TM (of execution time lower than p a poly.) which decide together L. We know that for q a poly., we can have M_q a PTM such that M_q decide B with an error lower than $e^{-q(.)}$. So, if we simulate all calls to the oracle by BPP, the error is lower than $p(.)e^{-q(.)}$. Notice that we can set q after that p is given, so we can set q to have a good error. Moreover, for an input of length n, we have a number of random words polynomial (of n) and all are of lenght polynomial (of $p(n)$, so of n), then our big random word used to simulate all random words have a length polynomial of n. Therefore $L \in \mathsf{BPP}$ ■

Exercise 3 : Probabilistic Logarithmic Space

Propose a definition of RSPACE.

Let $\mathsf{RL} = \bigcup_{k\in\mathbb{N}} \mathsf{RSPACE}(k \cdot log(n), \infty, 0, 1/2)$ be the class of languages that can be decided in probabilistic logarithmic space (the machine does not necessarily halt).

Show that :

1. For L in RL and M a PTM which decides L , If $x \notin L$, then $\forall r$, $M(x, r) \neq \top$
2. $\mathsf{RL} \subseteq \mathsf{NL}$
3. $\mathsf{RL} \subseteq \mathsf{RP}$

Solution:

Idea : What's difficult here is the possibility that the Machine doesn't stop. Moreover (for the same reason) we don't have a boundary on our random word so we have a probability on an infinite set (as in ZPP). For the definition we won't use the usual definition by contraposition because there is not two but three cases.

Proof : Definition : $\mathsf{RSPACE}(p(n), \infty, reject(n), accept())$ is the class of all languages L such that there is a randomized Turing machine M, working in space $O(p(n))$, that terminates with probability 1, and such that :

— If $x \in L$, then $Pr[M(x, r) = \top] \geq 1 - accept(n)$

— If $x \notin L$, then $Pr[M(x, r) = \bot] \geq 1 - reject(n)$

1. For $x \notin L$, if $\exists r_0$ s.t $M(x, r_0) = \top$ then for all word w $M(x, r_0 w) = \top$ so $Pr[M(x, r) = \top] > 0$, then $Pr[M(x, r) = \bot] < 1$: $\lightning$. ■
2. $\mathsf{RL} \subseteq \mathsf{NL}$:

   Given $L \in \mathsf{RL}$, M a RTM which decides L, we build the NDMT M' which follows the algorithm :

   Input: x a word
   Guess r a random word (bit by bit) ;
   return Simulate $M(x, r)$

   Therefore :

   — if $x \in L$ then $Pr[M(x, r) = \top] \geq \frac{1}{2}$, so $(\exists r, M(x, r) = \top)$ then $M'(x) = \top$

   — if $x \notin L$ then $(\nexists r, M(x, r) = \top)$ then $M'(x) = \bot$

   Notice that the first assertion is true if we take $|r| > 1$. Therefore : M' recognize L. Moreover, the resulting NL machine runs in space $klog(n)$ for some k, but may fail to terminate. As in the lectures, we know that any run of more than $a^{klog(n)}$ (where a is the alphabet size) will visit the same configuration twice. So we can stop any run when it exceeds that number of steps, and reject. This requires an counter of size $klog(n)$. Then $L \in \mathsf{NL}$. ■
3. $\mathsf{RL} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{RP}$ ■

Exercise 4 : BPP and PSPACE

Give a direct proof that $\mathsf{BPP} \subseteq \mathsf{PSPACE}$.

Solution:

Idea : The idea is to try all random words and count the rejects and the accepts.

Proof : Given M a PTM for a language L in BPP. By definition, we have $c \in \mathbb{N}$, such as $T_M(x,r) \leq |x|^c$, for all r of length lower than $|x|^c$. Let x be a word and $n = |x|$. There are $Max(x) = 2^{n^c}$ different r to test. So we use the following pseudocode :

```
Simulation(x):
    let nacc = 0
    let nrej = 0
    for r = 0 to Max(x) - 1 do
        res = Simulate M(x,r)
        if (res)
           then nacc ++
           else nrej ++
        end if
    endfor
    return (nacc > nrej)
```

r,$nacc$ and $nrej$ have a length lower than n^c. Moreover, by definition, we can simulate $M(x,r)$ in a polynomial time, so a fortiori, in a polynomial space. Then $L \in$ PSPACE. ■

Exercise 5 : BPP-completeness

1. Show that the language $L = \{(M, x, 1^t) \mid M \text{ accepts on input } x \text{ in time at most } t\}$, where M is the code of a non-deterministic Turing machine, x an input of M and t a natural number, is NP-complete.
2. Let L be the language of words $(M, x, 1^t)$ where M designates the encoding of a probabilistic Turing machine and x a string on $M's$ alphabet such that M accepts x in at most t steps, for at least $2/3$ of the possible random tapes of size t.
   Is L BPP-hard ? Is it in BPP ?

Solution:

1. — $L \in$ NP :
   Let M be the code of a non-deterministic Turing machine, x an input of M and t a natural number.
   Notice that the execution time of $M(x)$ is t so lower than the length of $(M, x, 1^t)$. So the algorithm which simulates M on x on the input $(M, x, 1^t)$ is non-deterministic polynomial. Then we can check that $(M, x, 1^t) \in \{(M, x, 1^t) \mid M \text{ accepts on input } x \text{ in time at most } t\}$.
   Therefore, $L \in$ NP.
   — L is NP-hard :
   Given $L' \in$ NP, M a NDTM for L', and p a polynomial associated.
   For an instance x of L' we can build (polynomially) the instance $(M, x, 1^{p(|x|)})$
   And, by definition of L, $(M, x, 1^{p(|x|)}) \in L \Leftrightarrow x \in L'$
   ■

2. — L is BPP-hard : (for exactly the same reasons).
   Given $L' \in$ BPP, M a NDTM for L', and p his polynomial associated.
   For an instance x of L' we can build (polynomially) the instance $(M, x, 1^{p(|x|)})$
   And, by definition of L, $(M, x, 1^{p(|x|)}) \in L \Leftrightarrow x \in L'$
   ■
   — Can't say the same thing cause bpp is not syntactic.

Exercise (bonus) 6 : Probabilistic Logarithmic Space

Let BPL be the class of languages L for which there exists a probabilistic Turing machine running in polynomial time and logarithmic space (the random tape does not count in terms of space) such that :

— if $x \in L$ then $Pr[M(x,r) = \bot] \leq \frac{1}{3}$

— if $x \notin L$ then $Pr[M(x,r) = \top] \leq \frac{1}{3}$

Show that $\mathsf{BPL} \subseteq P$.