Complexité avancée - TD 7

Benjamin Bordais

November 25, 2020

Exercise 1 RP^*

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

- If $x \in L$ then $Pr[M(x,r) \text{ reject}] < 1$
- If $x \notin L$ then $Pr[M(x,r) \text{ accept}] = 0$

Do you recognize this class?

Solution:
This is in fact NP.

- $\mathsf{RP}^* \subseteq \mathsf{NP}$: For the same reason than $\mathsf{RP} \subseteq \mathsf{NP}$
- $\mathsf{NP} \subseteq \mathsf{RP}^*$: Let us show that $\mathsf{SAT} \in \mathsf{RP}^*$. Let M be the probabilistic Turing machine that, on a formula ϕ with p free variable, and r a random tape of bits (of length $\geq p$), evaluates ϕ on r. We have that M runs in polynomial time. In addition, if we denote by $eval(\phi)$ the proportion of valuations that satisfy ϕ, we have $Pr[M(\phi,r) = \top] = eval(\phi)$ and $Pr[M(\phi,r) = \bot] = 1 - eval(\phi)$. Therefore:
  - If $\phi \in \mathsf{SAT}$, $eval(\phi) > 0$ and we have $Pr[M(\phi,r) = \bot] < 1$.
  - If $\phi \notin \mathsf{SAT}$, $eval(\phi) = 0$ and $Pr[M(\phi,r) = \top] = 0$.

  It follows that $\mathsf{SAT} \in \mathsf{RP}^*$. As RP^* is closed under logspace reduction, we have $\mathsf{NP} \subseteq \mathsf{RP}^*$.

Exercise 2 BPP and oracle machines

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

Solution:

- $\mathsf{BPP} \subseteq \mathsf{P}^{\mathsf{BPP}}$: This is straightforward, since one can ask the oracle the answer.
- $\mathsf{P}^{\mathsf{BPP}} \subseteq \mathsf{BPP}$: Let $L \in \mathsf{P}^{\mathsf{BPP}}$. By definition, there exists $B \in \mathsf{BPP}$ and M a TM (of execution time lower than a polynomial p) which decides L by calling the oracle B. We know that for all polynomial q, there exists a probabilistic Turing machine $\mathcal{M}_q$ running in polynomial time which decides B with a two-sided error lower than $2^{-q(n)}$. Consider now the probabilistic TM M' that executes M and

simulates all calls to the oracle B by simulating the execution of the TM $\mathcal{M}_q$. Note that the complete size of random words we need is polynomial as we make at most $p(n)$ calls to $\mathcal{M}_q$ which uses polynomial size random tapes. Furthermore, if no mistake is made in all the calls to $\mathcal{M}_q$, then M' does not make a mistake and correctly accepts or rejects inputs belonging or not to L. Hence, for $n = |x|$, we have $Pr[M'(x, r) \text{ errs}] \leq (1 - 2^{-q(n)})^{p(n)} \leq 1 - 2^{-q(n)} \cdot p(n) \leq 1 - 2^{p(n)-q(n)} \leq 1/3$ for $p(n) = q(n) + 2$. Note that $q(n)$ can be chosen as a function of $p(n)$ since p is given by the Turing machine M. Therefore $L \in \mathsf{BPP}$.

Exercise 3 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 now 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 ?

1.
   - $L \in \mathsf{NP}$. Let M be the code of a non-deterministic Turing machine, x an input of M and t a natural number. Notice that the timeout t we set for the execution of $M(x)$ is 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 and runs in polynomial time. 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, $L \in \mathsf{NP}$.
   - L is NP-hard. Given $L' \in \mathsf{NP}$, M a NDTM for L', and p a polynomial associated. For an instance x of L' we can build (in logspace) the instance $(M, x, 1^{p(|x|)})$. Then, 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 $L' \in \mathsf{BPP}$, M a probabilistic Turing machine for L', and p his polynomial associated. For an instance x of L' we can build (in logspace) the instance $(M, x, 1^{p(|x|)})$. And, by definition of L, $(M, x, 1^{p(|x|)}) \in L \Leftrightarrow x \in L'$
   - It is not known if this problem is in BPP. However, we know that we can not use the same idea than with NP, that is simulating the machine M. Indeed, if M accepts the input x with probability $1/2$, so will the simulation, whereas it should accept with probability lower than or equal to $1/3$.

Exercise 4 NP and randomized classes

Show that if $\mathsf{NP} \subseteq \mathsf{BPP}$ then $\mathsf{NP} = \mathsf{RP}$.
Hint: you may use the self-reducibility of SAT.

Solution:

In any case, we have $\mathsf{RP} \subseteq \mathsf{NP}$. Let's now assume that $\mathsf{NP} \subseteq \mathsf{BPP}$. So $\mathsf{SAT} \in \mathsf{BPP}$. We know that, for all polynomial q, we have M a probabilistic Turing machine running in polynomial time which recognizes SAT, with an error lower than or equal to $2^{-q(n)}$. We will define the M' a PTM which works as the following pseudocode:

Input: ϕ a formulae with p free variables; r randoms ; r' randoms
$\psi := \phi$;
for $i < p$ **do**
  if $M(\psi[x_i = \top], r_i) = \top$ **then**
    $\psi := \psi[x_i = \top]$
  else
    if $M(\psi[x_i = \bot], {r'}_i) = \top$ **then**
      $\psi := \psi[x_i = \bot]$
    else
      return $\bot$
    end
  end
end
return ψ is satisfied

Notice that $p < |\phi|$. There is a at most $2p$ calls to M. Hence, the running time of this algorithm is polynomial and total length of random word used is also polynomial. Therefore, for ϕ a formulae with p free variables, $|\phi| = n$ and $x = 2^{-q(n)}$:

- if $\phi \notin L$ then $Pr[M'(\phi, r) = \top] = 0$ (since we check that the last ψ is satisfied, which implies that the valuation chosen satisfies ϕ).
- if $\phi \in L$ then $Pr[M'(\phi, r) = \bot] \leq \sum_{i=0}^{i=2p-1}(1-x)^i x$ (it's the probability that one simulation of M fails). That is, $Pr[M'(\phi, r) = \bot] \leq \sum_{i=0}^{i=2p-1} x = 2p \cdot x = 2n \cdot 2^{-q(n)} \leq 2^{2n-q(n)}$.

  So, with $q(n) = 2n+1$: if $\phi \in L$ then $Pr[M'(\phi, r) = \bot] \leq \frac{1}{2}$

Then: $\mathsf{SAT} \in \mathsf{RP}$.

Exercise 5 The PP class

The class PP is the class of languages L for which there exists a polynomial time probabilistic Turing machine M such that:

- if $x \in L$ then $Pr[M(x, r)$ accepts $] > \frac{1}{2}$
- if $x \notin L$ then $Pr[M(x, r)$ accepts $] \leq \frac{1}{2}$

Also define $\mathsf{PP}_<$ as the class of languages L for which there exists a polynomial time probabilistic Turing machine M such that:

- if $x \in L$ then $Pr[M(x, r)$ accepts $] > \frac{1}{2}$
- if $x \notin L$ then $Pr[M(x, r)$ accepts $] < \frac{1}{2}$

1. Show that $\mathsf{BPP} \subseteq \mathsf{PP}$ and $\mathsf{NP} \subseteq \mathsf{PP}$;
2. Show that $\mathsf{PP} = \mathsf{PP}_<$ and that PP is closed under complement;
3. Consider the decision problem MAJSAT:
   (a) Input: a boolean formula ϕ on n variables
   (b) Output: the (strict) majority of the 2^n valuations satisfy ϕ.

Show that $\mathsf{MAJSAT} \in \mathsf{PP}$. In fact, MAJSAT is PP-complete.

One may also consider the decision problem MAXSAT:

(a) Input: a boolean formula ϕ on n variables, a number K

(b) Output: more than K valuations satisfy ϕ.

Show that MAXSAT is also PP-complete (to prove that $\mathsf{MAXSAT} \in \mathsf{PP}$ one may reduce MAXSAT to MAJSAT).

Solution:

1.
   - A language $L \in \mathsf{BPP}$ is recognized by a PTM M such that if $x \in L$ then $Pr[M(x,r) \text{ accepts }] \geq \frac{2}{3}$ and if $x \notin L$ then $Pr[M(x,r) \text{ accepts }] \leq \frac{1}{3}$. It follows that $L \in \mathsf{PP}$.
   - The class PP is closed under logspace reduction. It suffice to show that $\mathsf{SAT} \in \mathsf{PP}$. Consider now a probabilistic Turing machine with an input that is a formula ϕ. According to the first bit of the random tape, it either accepts or reads what remains of the random tape for a valuation and accepts if and only if it satisfies ϕ. Then, if $\phi \in \mathsf{SAT}$, we have $Pr[M(x,r) \text{ accepts }] > \frac{1}{2}$, otherwise $Pr[M(x,r) \text{ accepts }] = \frac{1}{2}$.

2. Trivially, we have $\mathsf{PP}_{<} \subseteq \mathsf{PP}$. Now, consider $L \in \mathsf{PP}$ and its associated Turing machine M running in polynomial time p. Without loss of generality, we assume that the alphabet of the random tape is of size 2, hence the probability of a random word for M on an input x such that $|x| = n$ is $2^{-p(n)}$. Therefore, if $x \in L$ then $Pr[M(x,r) \text{ accepts }] \geq \frac{1}{2} + \frac{1}{2^{p(n)}}$. Now, we construct another Turing machine M' that runs M on an input. If M would reject, M' rejects too, and if M would accept then M' rejects with probability $\frac{1}{2^{p(n)}}$ (for instance, by reading a word in the random tape of length $p(n)$and accepting only if there are only 0s). Then:

   - if $x \in L$: $Pr[M(x,r) \text{ accepts}] \geq (\frac{1}{2} + \frac{1}{2^{p(n)}}) \cdot (1 - \frac{1}{2^{p(n)}}) = \frac{1}{2} + \frac{1}{2^{p(n)+1}} - \frac{1}{2^{2 \cdot p(n)}} > \frac{1}{2}$
   - if $x \notin L$: $Pr[M(x,r) \text{ accepts}] \leq \frac{1}{2} \cdot (1 - \frac{1}{2^{p(n)}}) < \frac{1}{2}$

   That is, $L \in \mathsf{PP}_{<}$. The stability under complement then follows by inverting the accepting and rejecting states.

3. A probabilistic Turing machine that checks that a valuation read on the random tape satisfies the formula decides MAJSAT for PP. Then, MAJSAT can be reduced to MAXSAT in logarithmic (as one has to write on the output tape the number $2^{n-1}+1$ in binary, which consists in a 1, $n-2$ 0s and then a 1). Therefore, MAXSAT is also PP-hard. Let us now show that $\mathsf{MAXSAT} \in \mathsf{PP}$. To do so, let us reduce MAXSAT to MAJSAT. Consider an instance (ϕ, i) of MAXSAT with $0 \leq r_1 < r_2 < \ldots < r_k \leq n$ such that $2^n - i = 2^{n-r_1} + \ldots + 2^{n-r_k}$ (the values $n - r_j$ refers to the 1s in the binary decomposition of $2^n - i$). Let us denote $x_1, \ldots, x_n$ the variables of ϕ. Then, we consider the formula ψ as:

$$\begin{aligned}
\psi = &(x_1 \wedge \ldots \wedge x_{r_1}) \\
&\vee (\neg x_1 \wedge \ldots \wedge \neg x_{r_1} \wedge x_{r_1+1} \wedge \ldots \wedge x_{r_2}) \\
&\vee \cdots \\
&\vee (\neg x_1 \wedge \ldots \wedge \neg x_{r_{k-1}} \wedge x_{r_{k-1}+1} \wedge \ldots \wedge x_{r_k})
\end{aligned}$$

We can see there are exactly 2^{n-r_j} valuations satisfying the j-th line of ψ. With the negation at beginning of the lines, no valuation satisfies two lines of ψ. Therefore, there are exactly $2^{n-r_1}+\ldots+2^{n-r_k} = 2^n - i$ valuations satisfying ψ. Consider now a fresh variable y and the formula: $\phi' = (y \wedge \phi) \vee (\neg y \wedge \psi)$. Then, we have ϕ' computable in polynomial time from ϕ and ϕ is satisfied by more than i valuations if and only if ϕ' is satisfied by more than half of valuations, i.e. $\phi \in \mathsf{MAXSAT} \Leftrightarrow \phi' \in \mathsf{MAJSAT}$.