MVA: Reinforcement Learning **TP2**

Q-Learning / Stochastic Bandit algorithms

Lecturer: *Emilie Kaufmann* *emilie.kaufmann@inria.fr*

Deadline: November 27th, midnight

Note: Your report should be *short* and based on the answers to questions Q1-Q3

Report and code should be sent by e-mail to emilie.kaufmann@inria.fr (one pdf file *yourname_TP2.pdf* + one archive *yourname_TP2.zip*), with [MVA 2016] in the title.

1 Q-Learning

Using your code for TP1, the goal is to implement Q-Learning to learn the optimal policy for the tree-cutting problem of TP1 (use your previous code !).

The Q-Learning algorithm. Q-Learning estimates simulateously the quantities $(Q^*(x,a))$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$. The algorithm simulates trajectories, i.e. sequences of transitions (x_t, a_t, r_t, x_{t+1}) and updates its estimates along the way using

$$Q_{t+1}(x_t,a_t) = \left(1 - \alpha_{N(x_t,a_t)}(x_t,a_t)\right) Q_t(x_t,a_t) + \alpha_{N(x_t,a_t)}(x_t,a_t) \left[r_t + \gamma \max_{b \in \mathcal{A}} Q_t(x_{t+1},b) \right],$$

where

- $N(x,a)$ is the number of visits of the state-action (x,a)
- for each (x,a), $\alpha_i(x,a)$ is a sequence of *stepsizes*
- in state x_t, a_t is choosen based on some *exploration policy*

For the algorithm to converge, the stepsizes and exploration policy should be chosen such that all state-action pairs are visited infinitely often and $\alpha_i(x,a)$ satisfies the usual stochastic approximation requirements:

$$\sum_i \alpha_i(x,a) = +\infty \quad \text{and} \quad \sum_i \alpha_i^2(x,a) < +\infty.$$

You will implement Q-Learning in episodes (each starting at the initial state 1) of leangth $T_{\max} = 1000$, with an ϵ-greedy exploration policy within each episode, i.e. select an action as $a_t = \arg\max Q(x_t,a)$ with probability $1-\epsilon$ and randomize with probability ϵ.

Q1: Describe the (parameters of the) exploration policy and learning rate chosen, and illustrate the convergence of Q-Learning using some of the following performance metrics:

- Performance in the initial state $|V^*(I) - V^{\pi_n}(I)|$, where π_n is the greedy policy w.r.t. Q_n at the end of the n-th episode
- Performance over all the other state $||V^* - V^{\pi_n}||_\infty$
- Reward cumulated over the episode

2 Stochastic Multi-Armed Bandits on simulated data

In a stochastic multi-armed bandit model, each arm is a probability distribution and drawing an arm means observing a sample from this distribution. We will consider only distributions supported in $[0, 1]$. Arms can be implemented as objects, and we give the following classes (you can create others):

`armBernoulli.m` `armBeta.m` `armFinite.m` `armExp.m`

For each object *Arm* belonging to one of these classes, we have the following methods:

- *Arm.mean()* returns the mean of the arm
- *Arm.sample()* produces a sample from the arm

A multi-armed bandit model is a collection of arms:

```
MAB = {Arm1, Arm2, ...,ArmK}
```

2.1 Bernoulli bandit models

Start by defining your own Bernoulli bandit model with K arms of means $p_1, \dots, p_K$ (see *mainTP2.m*).

We denote by :

- $N_a(t)$ the number of draws of arm a up to time t
- $S_a(t)$ the sum of rewards gathered up to time t
- $\hat{\mu}_a(t) = \frac{S_a(t)}{N_a(t)}$ the empirical mean of the rewards gathered from arm a up to time t.

UCB1. The UCB1 algorithm starts with an initialization phase that draws each arm once, and for $t \geq K$, chooses at time $t+1$ arm

$$\boxed{A_{t+1} = \underset{a \in \{1,\dots,K\}}{\operatorname{argmax}} \ \hat{\mu}_a(t) + \sqrt{\frac{\log(t)}{2N_a(t)}}}$$

Thompson Sampling. In a *Bayesian* view on the MAB, the means p_a are no longer seen as unknown parameters but as (independent) random variables following a uniform distribution. The *posterior distribution* on the arm a at time t of the bandit game is the distribution of p_a conditional to the observations from arm a gathered up to time t. Each sample from arm a leads to an update of this posterior distribution.

$$\begin{aligned} \textbf{Prior distribution} \quad & p_a \sim \mathcal{U}([0,1]) \\ \textbf{Posterior distribution} \quad & p_a | X_1, ..., X_{N_a(t)} \sim \pi_a(t) := \text{Beta}\left(S_a(t)+1, N_a(t) - S_a(t) + 1\right) \end{aligned}$$

where $X_1, ..., X_{N_a(t)}$ are the rewards from arm a gathered up to time t. Thompson Sampling chooses

$$\boxed{A_{t+1} = \underset{a \in \{1,\dots,K\}}{\operatorname{argmax}} \ \theta_a(t), \quad \text{where} \quad \theta_a(t) \sim \pi_a(t)}$$

1. Write two functions

   `[rew,draws]=UCB1(T,MAB)` `[rew,draws]=TS(T,MAB)`

   simulating a bandit game of length T with the UCB1 and Thompson Sampling strategy on the bandit model MAB: *rew* and *draws* are the sequence of the T rewards obtained and of the T the arms drawn.

2. Based on many simulations, estimate the expected regret of UCB1 and Thompson Sampling and display regret curves.

   $$\mathbb{E}[R_T] = Tp^* - \mathbb{E}\left[\sum_{t=1}^{T} x_t\right]$$

   You may also add the naive strategy that selects the empirical best arm at each round.

3. In a Bernoulli bandit model, Lai and Robbins lower bound tells that

   $$\liminf_{T\to\infty} \frac{R_T}{\log(T)} \geq \sum_{a:p_a<p^*} \frac{p^* - p_a}{\mathrm{kl}(p_a, p^*)} := C(p),$$

   with

   $$\mathrm{kl}(x,y) = \mathrm{KL}(\mathcal{B}(x), \mathcal{B}(y)) = x\log(x/y) + (1-x)\log((1-x)/(1-y)).$$

   $C(p)$ may be called the complexity of the problem.

   Add the "oracle" regret curve $t \mapsto C(p)\log(t)$ on your graph

Q2: For two different Bernoulli bandit problems (that you specify), with different complexity, compare the regret of Thompson Sampling with that of UCB1. Add Lai and Robbins' lower bound on your plots.

2.2 Non-parametric bandits (bounded rewards)

The UCB1 algorithm can be used in any bandit model such that each arm is bounded on $[0,1]$, without modification.

1. Using the other classes of arms, build a MAB with arms that are not only Bernoulli.
2. Propose an adaptation of Thompson Sampling to handle non-binary rewards, and implement it together with UCB1 on your bandit model.

Q3: Describe the proposed implementation of Thompson Sampling, and present a regret curve in a bandit model that you specify. Does the notion of complexity still make sense?