Reinforcement learning - MVA 2017/2018
Homework 2

Rémi Lespinet

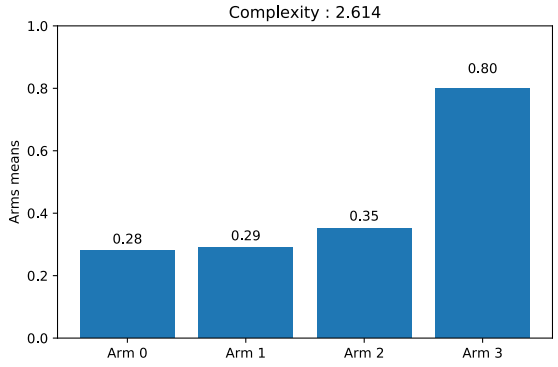
1 Stochastic Multi-Armed Bandits on Simulated Data

1.1 Bernoulli bandit models

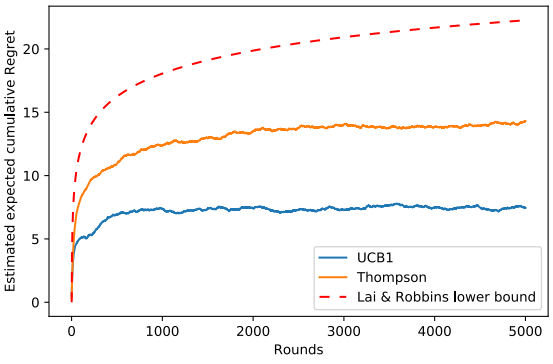
• Q1: The figures 1 and 2 represent two different Bernoulli bandit problems with different complexity. For each problem, the left subfigure represents the repartition of the arms and the complexity $C(p)$ as defined by the *Lai and Robbins* formula [3]. The right figure represents the estimation of the expected cumulated regret for both algorithm (UCB1 and Thompson Sampling). The value ρ for UCB1 is equal to 0.5 for both problems, and the expected cumulated regret is computed over 1000 simulations. Each simulation has a horizon $T = 5000$.

The first chosen problem has 4 arms of means. The complexity $C(p) = 2.614$

The first chosen problem has 5 arms of means $[0.64, 0.18, 0.80, 0.81, 0.29]$ and a complexity $C(p) = 34.825$

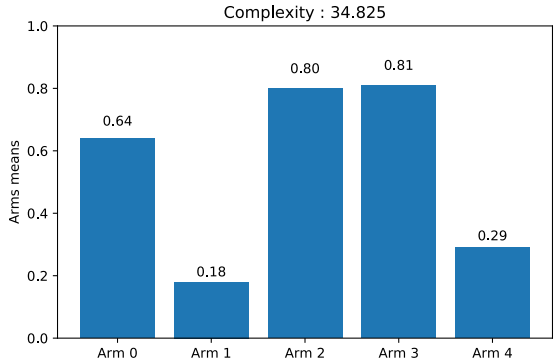


(a) Representations of the parameters of each arms

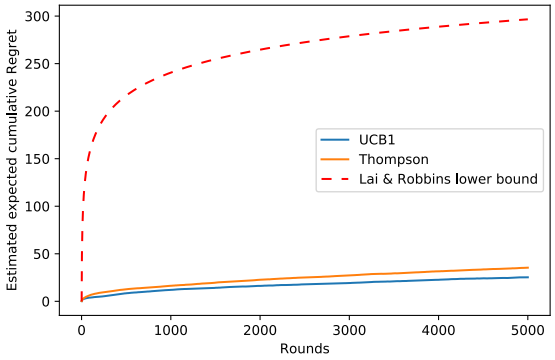


(b) Estimated expected cumulative regret for UCB1 and TS, and Lai-Robbins lower bound

Figure 1: Representation of the arms means and expected cumulative regret for the first chosen Bernoulli bandit



(a) Representations of the parameters of each arms

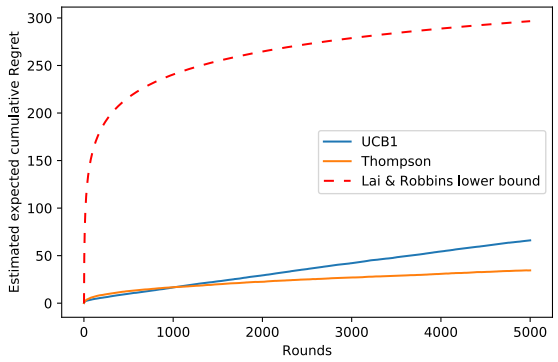


(b) Estimated expected cumulative regret for UCB1 and TS, and Lai-Robbins lower bound

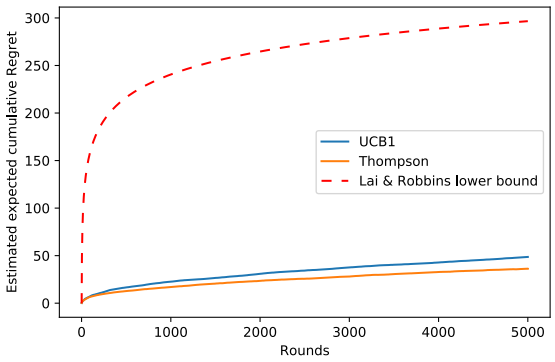
Figure 2: Representation of the arms means and expected cumulative regret for the second chosen Bernoulli bandit

UCB1 vs Thompson

In the two presented cases, the estimated expected cumulative regret is lower for the UCB1 algorithm than for the Thompson algorithm. But we see that this depends on the value of ρ chosen. The figure 3 presents the results obtained for the second Bernoulli bandit problem for two other values of ρ.



(a) $\rho = 0.2$



(b) $\rho = 1.0$

Figure 3: Comparison of UCB1 and Thompson sampling algorithm for 2 different values of ρ on the second chosen Bernoulli bandit problem

This can be explained easily, for the UCB algorithm, we have that By Chernoff-Hoeffling

$$P\left(\bar{X} - \mathbb{E}\left[X\right] \leq \sqrt{\frac{\ln 1/\delta}{2n}}\right) \geq 1 - \delta$$

If we take δ such that $\ln 1/\delta = \rho^2 \ln t$, we have

$$P\left(\bar{X} - \mathbb{E}\left[X\right] \leq \rho\sqrt{\frac{\ln t}{2n}}\right) \geq 1 - \frac{1}{t^{\rho^2}}$$

- If we take ρ too small, the confidence interval length is small but we are less confident that $\bar{X} - \mathbb{E}[X]$ lies in this interval. Given that the dependency in time is in $\log t$, this means that if we evaluate a sub-optimal arm as optimal in the early iterations of the algorithm, and we discard the optimal one, we will end up pulling this sub optimal arm for a very long time until the confidence interval eventually grows higher (because of the $\log(t)$ term) and we have a chance to correct our mistake, and this will have a negative impact on the expected cumulative regret.
- If we take ρ too large, the confidence intervals for each arm will be large, and we will end up exploring a lot more than necessary, which also causes the expected cumulative regret to be large.

For a Bernoulli bandit problem with 4 arms (means 0.20, 0.25, 0.30, 0.35), I've run the UCB algorithm 1000 times for different values of ρ and a horizon $T = 5000$ and measured the frequency of success (that is the best arm being pulled more often than the others). The figure 4 presents the curve obtained.

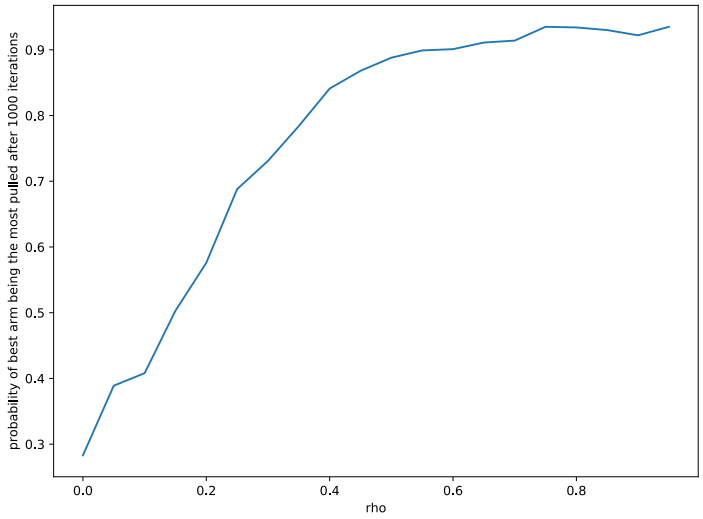


Figure 4: Empirical estimation of the probability of the arm with maximum mean mu_{i*} being pulled the most as a function of ρ for a Bernoulli bandit problem with means (0.20, 0.25, 0.30, 0.35)

Advantages of Thompson sampling

The main advantages of the Thompson sampling is that it does not need a parameter and that the random sampling behavior allow it to avoid getting trapped like the UCB1 algorithm due to wrong estimations in the early iterations of the algorithm. Another advantage is that it provides estimations of the means as a continuous probability density function (which allows to compute confidence intervals etc ...)

Intuition of the lower bound

Intuitively, when the distributions of each arm are close to each other (in the case of a Bernoulli, when the means are close), the Kullback-Leibler divergence is close to zero, and hence the complexity $C(p)$ is very high, which traduces the fact that it is hard to differentiate the arms.

1.2 Non-parametric bandits

- Q2: First the Thompson sampling algorithm for Bernoulli bandits relies on the fact that if the prior distribution of the mean is Beta(α, β) then the posterior distribution after an observation of a Bernoulli sample x_i is also a beta distribution Beta$(\alpha + x_i, \beta + (1 - x_i))$

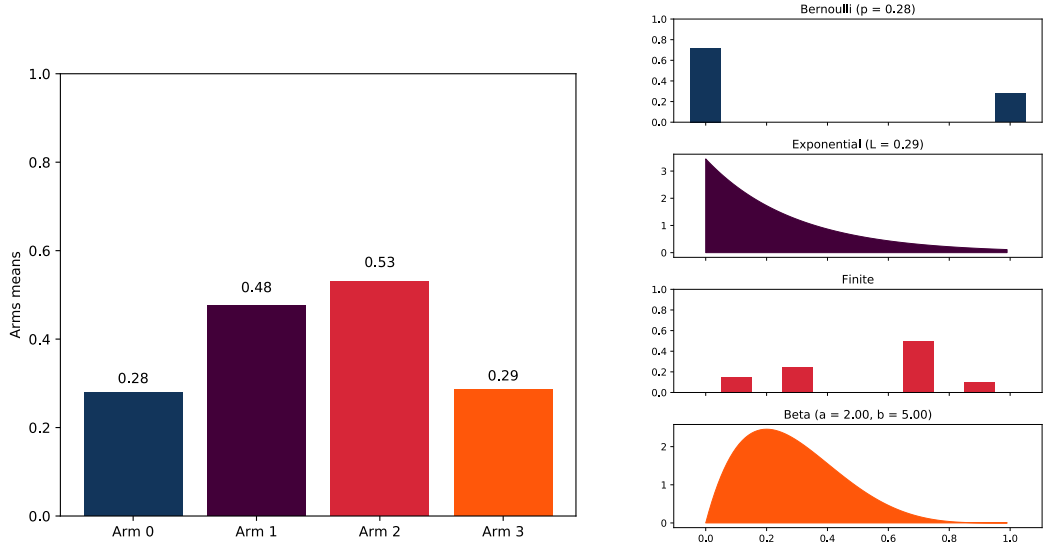
$$p(\mu|x_i) \propto p(x_i|\mu)p(\mu) = \mu^{x_i}(1-\mu)^{1-x_i}\mu^{\alpha-1}(1-\mu)^{\beta-1} = \mu^{\alpha-1+x_i}(1-\mu)^{\beta-x_i}$$

In general, when the MAB contains at least an arm that does not follow a Bernoulli distribution, because the conjuguate prior distribution for such an arm is not the Beta distribution in general, we cannot apply the exact same algorithm. The idea proposed in [1] is to pull the arm, observe the reward $\tilde{r}_t$ which is drawn from a probability distribution with density f taking values in $[0, 1]$, and then perform a Bernoulli trial with probability $\tilde{r}_t$, we note r_t the outcome. r_t is a random variable following a Bernoulli distribution, and we have

$$p(r_t = 1) = \int_0^1 p(r_t = 1|\tilde{r}_t)f(\tilde{r}_t)d\tilde{r}_t = \int_0^1 \tilde{r}_t f(\tilde{r}_t)d\tilde{r}_t = \mathbb{E}\left[\tilde{r}_t\right] = \mu$$

Which means that we have constructed a Bernoulli random variable with the same mean as the general distribution of the arm and we can hence apply the same algorithm using this random variable to update the Beta parameters. We can also notice that if an arm follows a Bernoulli distribution, the added step has no effect and we then recover the Bernoulli Thompson sampling as we can expect.

We then apply this algorithm as well as UCB1 to a more general bandit model where arms are not Bernoulli. The chosen model contains 4 arms, whose distributions are represented in the figure 5.



(a) Representation of the mean of each arm

(b) Distribution of each arm

Figure 5: Example of a more general bandit model

The regret curve for this bandit model is shown in figure 6.

The theorem introducing the lower bound [3] only holds when the distribution of each arms follow the same univariate distribution with a single parameter, thus it is not valid for this

new bandit model. *Burnetas and Katehakis* [2] proved that this bound can be extended to multiparameter or nonparametric models. It shows that under mild regularity condition, any policy satisfies

$$\liminf_{n\to\infty} \frac{R_T}{\log(T)} \geq \sum_{a:\mu_a<\mu^*} \frac{\mu^* - \mu_i}{KL_{\inf}(F_a, \mu^*)}$$

with

$$KL_{\inf}(F, \mu) = \inf_{G\in\mathcal{F}:E[G]>\mu} KL(F\|G)$$

It is a little more difficult to compute, and hence the figure 6 does not show this lower bound. We can still notice that the intuition behind the complexity expression remains unchanged, and that this formula is indeed more general than the previous one (if we take only arms following Bernoulli distributions, we recover the Lai and Robbins lower bound expression).

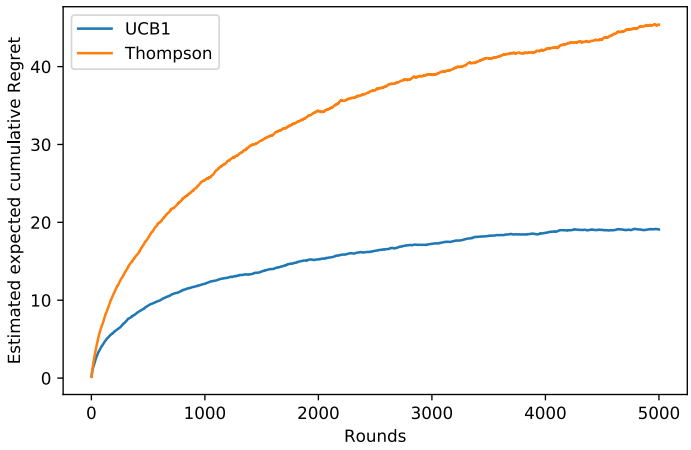


Figure 6: Estimated expected cumulative regret for UCB1 and TS, and Lai-Robbins lower bound

2 Linear Bandit on Real Data

- Q3: To write the algorithm efficiently (as suggested in the homework), we use the following relations

$$Z_{t+1}^T Z_{t+1} = \left[\overbrace{Z_t^T}^{t} \;\middle|\; \phi_{a_{t+1}}^T\right] \cdot \begin{bmatrix} \overbrace{Z_t}^{d} \\ \hline \phi_{a_{t+1}} \end{bmatrix} = Z_t^T Z_t + \phi_{a_{t+1}}^T \phi_{a_{t+1}}$$

and

$$Z_{t+1}^T y_{t+1} = \left[\begin{array}{c|c} \overbrace{\quad Z_t^T \quad}^{t} & \phi_{a_{t+1}}^T \end{array}\right] \cdot \left[\begin{array}{c} y_t \\ \hline r_{t+1} \end{array}\right] = Z_t^T y_t + r_{t+1}\phi_{a_{t+1}}^T$$

Hence, if we note

$$W_{t+1} = Z_{t+1}^T Z_{t+1} + \lambda I_d$$

and

$$V_{t+1} = Z_{t+1}^T y_{t+1}$$

We have the update formulas

$$\begin{aligned} W_{t+1} &= W_t + \phi_{a_{t+1}}^T \phi_{a_{t+1}} \\ V_{t+1} &= V_t + r_{t+1}\phi_{a_{t+1}}^T \end{aligned}$$

The figure 7 compares the performances of linUCB against the two following algorithm (in term of estimated expected regret)

- Random policy : (Pure exploration) Take a random arm (propose a random movie) uniformly at each iteration
- Epsilon greedy policy : Initialize by taking all arms, explore with probability ϵ and exploit with probability $1 - \epsilon$

Notice that these two algorithms do not use the context.

I could have also compared the evolution of the estimated distance to the real theta for each algorithm, in fact, we can computed an $\hat{\theta}_t$ for the epsilon greedy algorithm from the equality

$$\hat{r}_t(a) = \phi_a^T \hat{\theta}_t$$

by computing the pseudo-inverse of ϕ_a^T and right multiply this matrix by our current estimate of the rewards. But since the Epsilon Greedy algorithm starts by taking all arms (207), it already has a good estimate of θ^* before actually starting the greedy strategy.

Performances of linUCB

We see that the linUCB algorithm is very effective, experimentally for a value of $\alpha = 1.0$, over 100 run, the algorithm found the best arm 100% of the time and started to exploit it after about 12 draws in average. This would of course be impossible if we would not use the context (there are 207 arms, so this is impossible to pretend to know the optimal arm if we don't at least pull one time each arm).

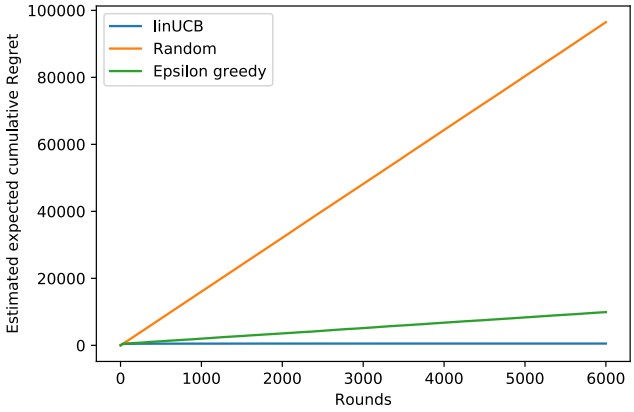


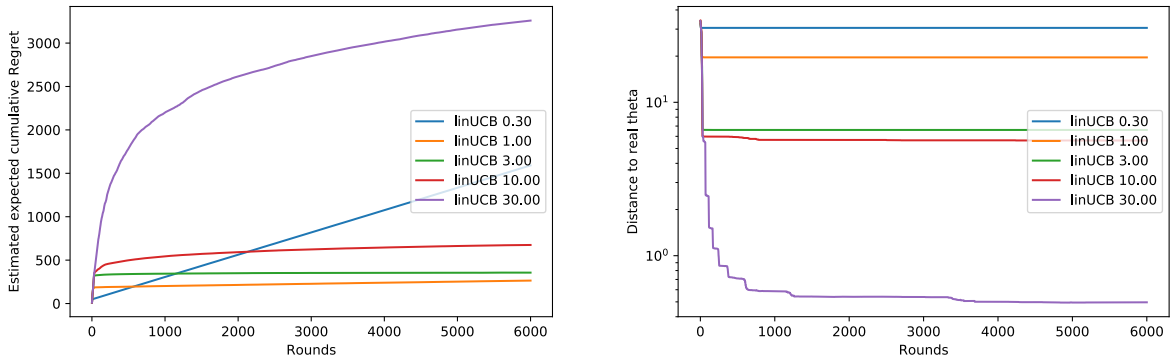
Figure 7: Comparison of the estimated expected regret for the different algorithm implemented (regularization parameter $\lambda = 0.1$)

Trying different values of α

The figure 8 shows the impact of α on the estimated expected regret and on the distance $\|\theta_t - \theta^*\|$ between the estimate of θ_t and the real value θ^* (provided)

As expected, as α increases, we give more importance to arms which are the most incertain (and less importance to the information we gain on the rewards of the arms), this means we explore more and we exploit less. In the figure 8, the consequences are that the regret is higher (we exploit more), and the aveage distance to the real theta decreases (we gain information about the distributions of the arms).

As in the UCB1 case, if we take a very small value of α, the algorithm will try to estimate the value of θ^* with a very small number of draws, and might then conclude about the optimality of a sub-optimal arm, which it will then pull for the rest of the simulation. For example if $\alpha = 0.3$, we see that the algorithm draws in average 3 arms before concluding about the optimality of an arm and then pull it to the end of time. This explains the observed behavior in the figure 8a for $\alpha = 0.3$ (the algorithm pulls the same sub-optimal arm after the third iteration, which induces a regret at each iteration).



(a) Impact on the regret

(b) Impact on $\|\theta_t - \theta^*\|$

Figure 8: Impact of α on the regret and the distance to the real theta (average over 20 simulations of horizon $T = 6000$ with $\lambda = 0.1$)

References

[1] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. 2012.

[2] A. N. Burnetas and M. N. Katehakis. Optimal adaptive policies for sequential allocation problems. 1996.

[3] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. 1985.