Reinforcement Learning
Homework 2

Amine Kheldouni

26 November 2018

1 Stochastic Multi-Armed Bandits on Simulated Data

1.1 Bernoulli bandit models

Question 1

We consider two different Bernoulli bandit problems as follows :

— One easy-case Bernoulli multi-armed bandit where complexity is relatively small ($C(p) = 1.47$) :

$$MAB_1 = \{\mathcal{B}(0.8), \mathcal{B}(0.3), \mathcal{B}(0.1)\}$$

— One hard-case Bernoulli multi-armed bandit where complexity is higher ($C(p) = 141.64$) :

$$MAB_2 = \{\mathcal{B}(0.48), \mathcal{B}(0.47), \mathcal{B}(0.50), \mathcal{B}(0.49), \mathcal{B}(0.49)\}$$

Those two models have different complexities in terms of arms numbers and means adjacency. Running our UCB1 and Thompson algorithms, we can plot the average cumulative regret over $N = 20$ simulations, as well as Lai and Robbin's lower bound.

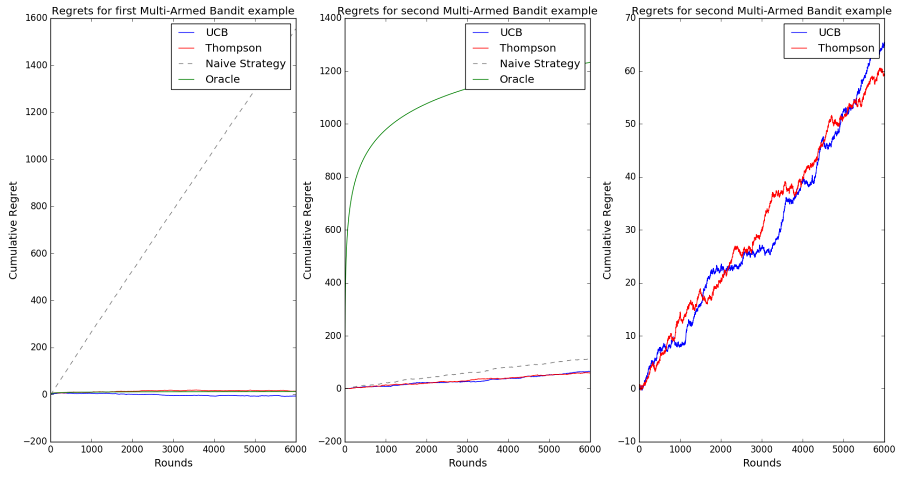

FIGURE 1 – Cumulative regrets for UCB1 and Thompson sampling algorithms for the easy-case MAB (*left*) and hard-case MAB (*center* and *right*) with Lai Robbins boundary

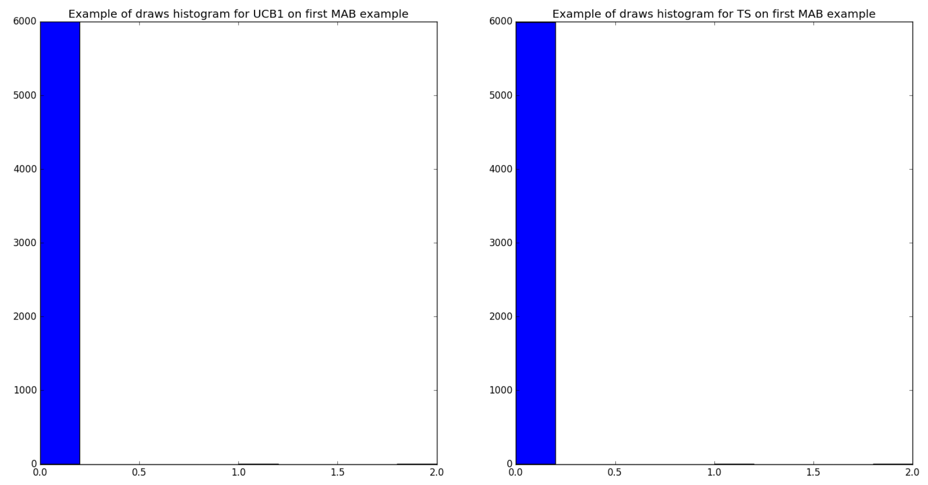

FIGURE 2 – Histogram of first MAB draws for both UCB1 and Thompson Sampling algorithms

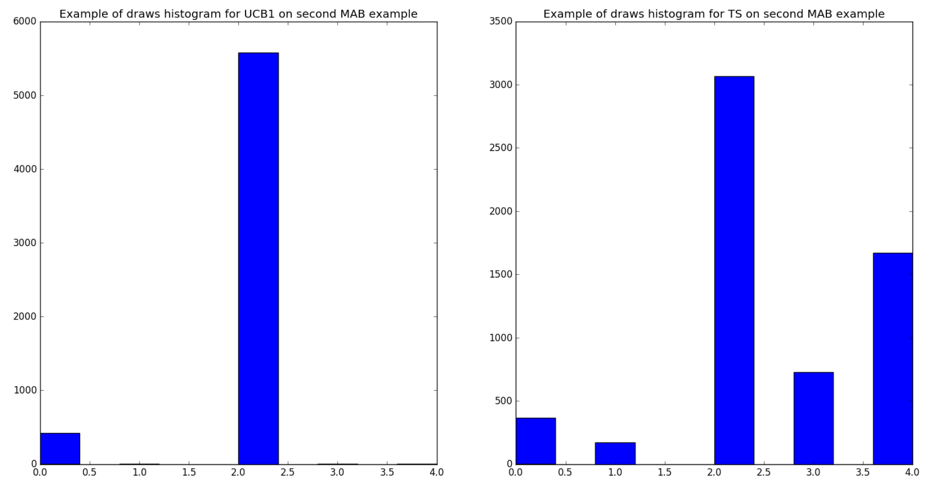

FIGURE 3 – Histogram of second MAB draws for both UCB1 and Thompson Sampling algorithms

We notice that for the easy scenario bandit problem, the naive solution does not explore at all, being too greedy, and commits a huge cumulative regret compared to UCB1 and Thompson Sampling (TS). For the second bandit scenario constructed (MAB_2), we notice that UCB1 performs as well as TS in average which may not be the case in general. The histograms display the number of times each arm is pulled by the algorithms. We notice that in the first case, it is easy for both algorithms to explore quickly the actions and exploit the best one, in the second case UCB1 settles faster than TS algorithm on the arm that maximizes the reward while the Bayesian approach takes a little more time to explore the environment.

The UCB1 algorithm, as we acknowledge it, uses ρ as a parameter of regularization between exploration and exploitation. The exploration and exploitation trade-off is an important issue of MAB problems, since without exploration as seen on the naive strategy (same as setting $\rho = 0$), we underestimate the risk of exploiting a sub-optimal arm. On the other hand, exploring too much generates marginal regrets of not stopping the exploration to take advantage of the best empirical arm.

Therefore, we thought of tuning the ρ parameter in the UCB1 algorithm (to set its exploration/exploitation trade-off), and we plot in the following figure the cumulative regret for different values of ρ [4]. Hence we chose $\rho = 0.2$ (as recommanded in the course's lecture) because the value gives a good settlement between exploration and exploitation.

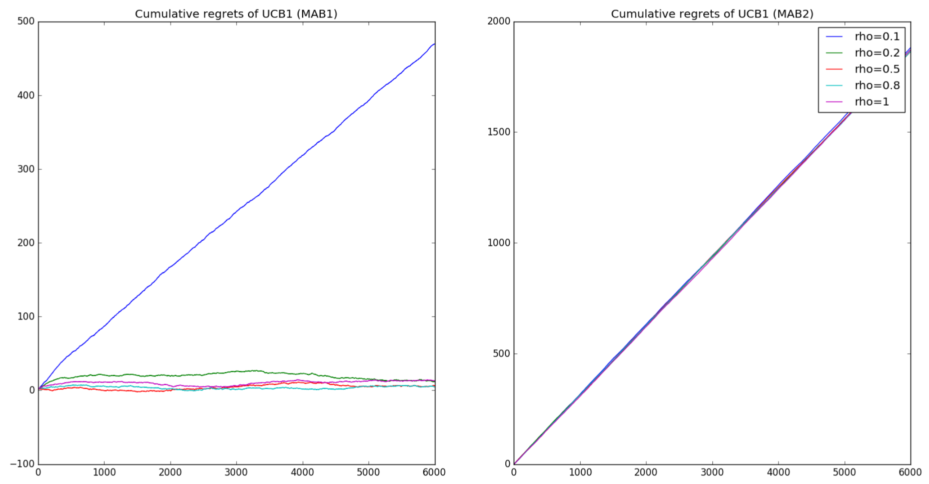

FIGURE 4 – Tuning the parameter ρ of UCB1 to explore enough but not too much ($\rho = 0.2$ seems a good strategy)

Concerning the Lai and Robbins' lower bound, we notice that both algorithms (UCB1 and TS) may take some time before getting higher values than the oracle regret curve. In other words, the lower bound underestimates our algorithms and tends to believe that the cumulative regret will be much higher. This is an illustration of the asymptotic property of the inequality, which may take more rounds to be effective as T grows to infinity.

1.2 Non-parametric bandits

Question 2

This time we build a Multi-Armed Bandit with different classes of arms.

$$MAB = \{\mathcal{B}(0.5), \beta(0.3, 0.45), \mathcal{E}(0.2), \mathcal{E}(0.1), \mathcal{B}(0.1), \mathcal{F}(X, P)\}$$

where $X = (0.1, 0.3, 0.7, 0.8)$ and $P = (0.2, 0.4, 0.1, 0.3)$. The means vector of our diversified MAB is the following :

$$\mu_{MAB} = (0.5, 0.4, 0.48, 0.49, 0.1, 0.45)$$

We propose an adaptive Thompson sampling algorithm which handles non-binary rewards for multi-class MAB. This adaptation follows the same sketch as in the first Bernoulli case, but still operates a transformation on the reward r_t, converting it into a binary value by comparing it to a random number

$p \in [0, 1]$. In other words, we perform a Bernoulli trial with success probability r_t and we use the observed output to update our success or failure counters. This procedure is explained in [1]. We display in the following figure the non-binary Thompson Sampling algorithm and the UCB1 algorithm. The notion of

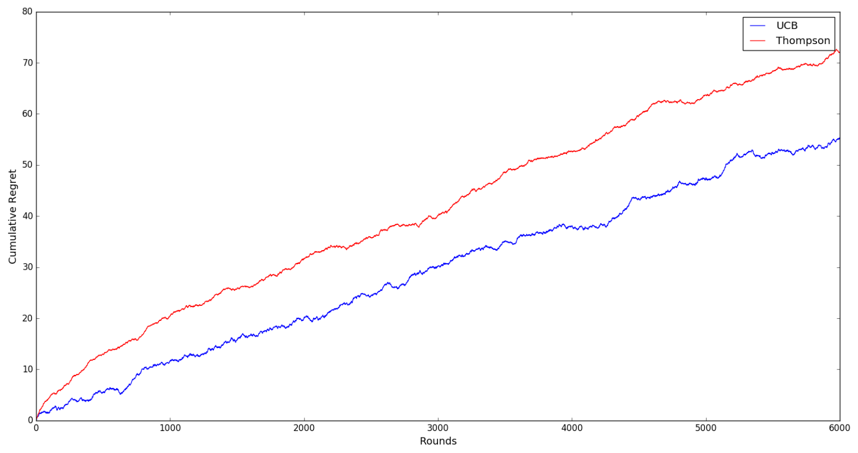

FIGURE 5 – Cumulative regrets for non-binary TS and UCB1 in a non-parametric bandits example

complexity does not make sense because we are using other classes of arms to build our multi-armed bandit which can even be non parametric.

2 Linear Bandit on Real Data

Question 3

Discussing the algorithms

First and foremost, we implemented the three algorithms (Random policy, Epsilon Greedy and LinUCB) for a *ToyLinearModel* with much less parameters ($n_{actions} = 20$, $n_{features} = 8$) than the MovieLens problem. Our primary goal was to test the algorithms and demonstrate their convergence.

Therefore, we computed the $\hat{\theta}$ estimator and the cumulative regret for multiple simulations to get an average estimation of our algorithms performances. For $T = 6000$ rounds, we display in the following figure for each of our three algorithms the l_2-norm of $(\hat{\theta} - \theta^*)$, and the expected cumulative regret. There are many results shown in the figure (6). First, concerning the evolution over rounds of $\|\theta - \hat{\theta}\|_2$, we notice that all three algorithms tend to converge to small values of the difference. Again, the exploration/exploitation trade-off is symbolized in the estimated mean reward that we tend to maximize :

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

The term $\phi_a^T \hat{\theta}_t$ favors the exploitation and the upper bound term $\beta_t(a)$ favors the exploration. $\hat{\theta}$ describes how well we are aware of the environment. This taken into consideration, the random strategy explores too much without exploiting and ends to have an accurate representation of the environment. This explains why $\|\theta^* - \hat{\theta}\|_2$ tends to 0 for this strategy. On the other hand, without exploitation, the random strategy accumulates more regret since it doesn't learn from its exploration.
The Epsilon Greedy algorithm performs a better trade between exploration and exploitation since it is

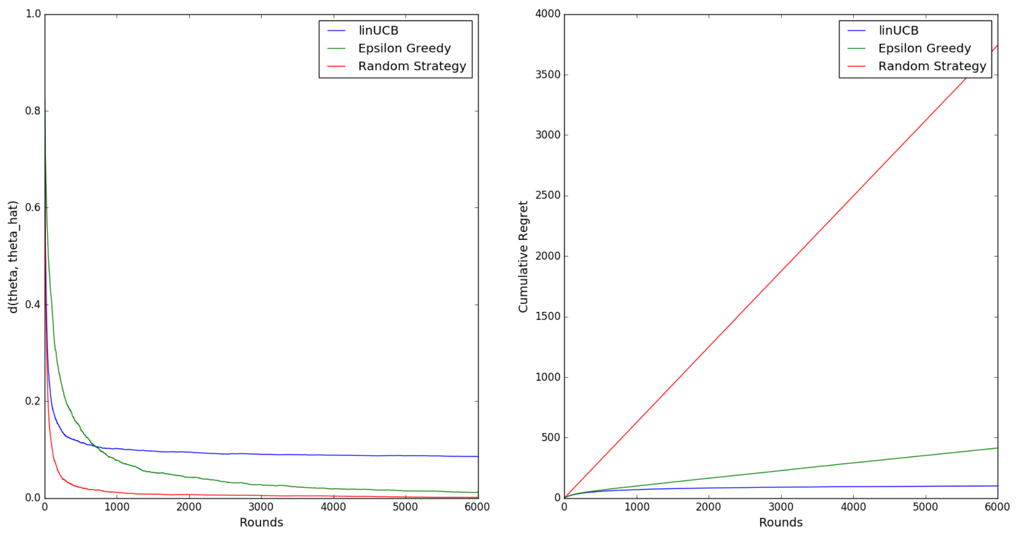

FIGURE 6 – Convergence of algorithms and expected cumulative regret

greedy with probability $1 - \epsilon$ and it explores the situations with probability ϵ. Moreover, LinUCB algorithm performs much better than both random and epsilon greedy algorithms. The LinUCB algorithm explores thanks to its upper bound confidence term and exploits over $\phi_a^T \hat{\theta}$. Even though it does not learn θ^* as well as the others, this algorithm has the lowest cumulative regret over the episodes.

Results on MovieLens dataset

Running the implemented algorithms on MovieLens dataset, we average the cumulative regret over $N = 20$ simulations and display the results below.

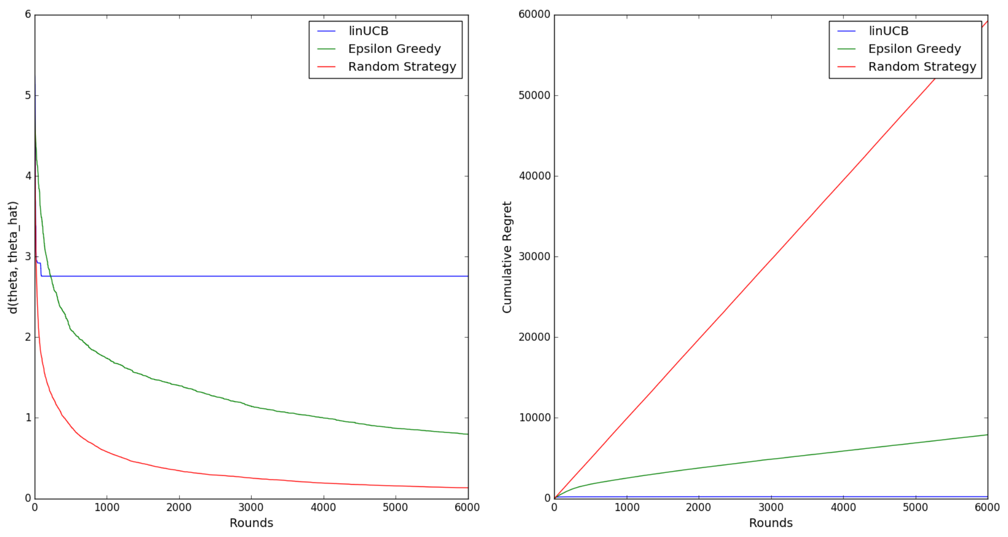

FIGURE 7 – Convergence of algorithms and expected cumulative regret (MovieLens dataset)

The same comments hold for the *ColdStartMovieLens* model. The LinUCB algorithm performs really well in minimizing the cumulative regret but explores lesser than the other algorithms and therefore the estimate $\hat{\theta}_{LinUCB}$ does not converge well towards θ^*.

Sensitivity to parameters

To discuss the sensitivity of each algorithm over the different parameters (mainly ϵ for Epsilon Greedy and λ, α for LinUCB), we plot the cumulative regret for different values of ϵ and α in the following figure.

For LinUCB algorithm, We fix $\lambda = 1$ for convenience and we choose $\alpha = 1 + \sqrt{\frac{log(2/\delta)}{2}}$ as explained in [2] (with $\delta = 1\%$) so that the confidence interval inequality over $|\hat{r}_t(a) - r(a)|$ is true with probability $1 - \delta = 99\%$. Therefore, $\alpha \approx 2.63$.
For Epsilon Greedy algorithm, we need to take ϵ small enough to exploit and minimize the regret and large enough to explore other states. Looking at figure [8] we choose $\epsilon = 0.1$.

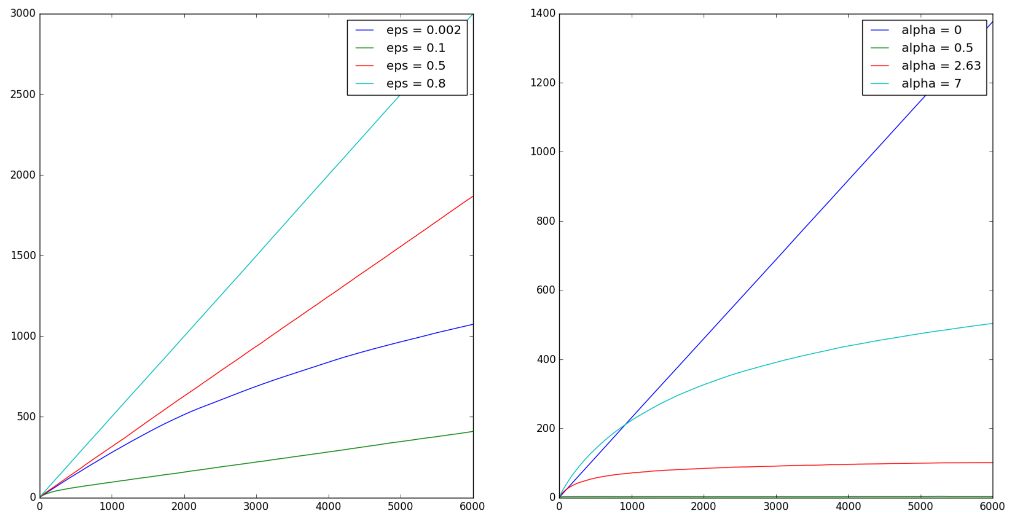

FIGURE 8 – Sensitivity of Epsilon Greedy (*left*) and LinUCB algorithm (*right*) to parameters ϵ and α.

Références

[1] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

[2] L. Li, W. Chu, J. Langford, R. E. Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation, pages 662-664, 2010.