MVA: Reinforcement Learning **TP2**

The Exploration-Exploitation Dilemma

Lecturer: *Matteo Pirotta* *matteo.pirotta@inria.fr*

Deadline: December 1, midnight

Note: Your report should be *short* and based on the answers to questions Q1-Q3.
The code is not optional!

The solution (single archive with name *lastname_firstname_TP2.zip*) should be sent by e-mail to matteo.pirotta@inria.fr, with "[MVA 2017] TP2" as object.

We provide support for Matlab and Python but you can solve the homework using any language. You can choose between the following formats for the solution:

- Report and code: a short report (PDF) based on the answers to questions and the source code.
- Code with comments: integrate answers and comments in the code. For example, you can use a Matlab report or a Python notebook.

1 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`

In the python code, the arm models are specified in `arms.py`.

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}
```

1.1 Bernoulli bandit models

Start by defining your own Bernoulli bandit model with K arms of means $p_1, \dots, p_K$ (see *mainTP2_SMAB*).

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

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

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.

$$\begin{aligned} \textbf{Prior distribution} \quad & p_a \sim \mathcal{U}([0,1]) \\ \textbf{Posterior distribution} \quad & p_a | R_1, ..., R_{N_a(t)} \sim \pi_a(t) := \text{Beta}\left(S_a(t)+1, N_a(t) - S_a(t) + 1\right) \end{aligned}$$

where $R_1, ..., R_{N_a(t)}$ are the rewards from arm a gathered up to time t. Thompson Sampling chooses

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

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.

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

   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 [Lai and Robbins, 1985] tells that

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

   with

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

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

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

Q1: 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.

1.2 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.
   Why non-parametric? because we consider a MAB with different classes of arms (which can even be non parametric).
2. Propose an adaptation of Thompson Sampling to handle non-binary rewards and implement it together with UCB1 on your bandit model (see e.g., [Agrawal and Goyal, 2012]).

Q2: 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? Suggestion: [Burnetas and Katehakis, 1996].

2 Linear Bandit on Real Data

In this section, we investigate a real-world application of linear bandits. In particular, we consider a recommendation problem where we need to discover the preferences of a user (cold start).

2.1 Linear Bandit

We start briefly reviewing the settings of linear bandits. We assume that each action (arm) a can be represented by a feature vector $\phi_a \in \mathbb{R}^d$ and that the reward can be expressed as a linear combination of the arm and an unknown parameter $\theta^* \in \mathbb{R}^d$: $E[r_a] = \phi_a^\intercal \theta^*$.

The linear bandit problem (LinUCB) can be formalized as follows. At each round $t = 1, 2, \ldots$

1. The algorithm selects the action a_t that maximises the upper bound on the expected reward
$$a_t \in \arg\max_{a \in \mathcal{A}} \phi_a^\intercal \hat{\theta}_t + \beta_t(a)$$
   where $\hat{\theta}_t \in \mathbb{R}^d$ is an estimate of the unknown θ^*.
2. Observe a reward $r_{a_t} = r_t(a_t)$

Upper confidence bound. In order to select the action a_t we need to compute an estimate $\hat{\theta}_t$ and a confidence bound $\beta_t(a)$ for each arm. The term $\hat{r}_t(a) = \phi_a^\intercal \hat{\theta}_t$ represents the estimated mean reward for action a, where $\hat{\theta}_t$ can be computed via regularized least squares using the observed rewards:

$$\hat{\theta}_t = \arg\min_{\theta} \sum_{i=0}^{t} \left(\phi_{a_i}^\intercal \theta - r_{a_i}\right)^2 + \lambda \|\theta\|_2^2$$

that can be solved in closed-form:

$$\hat{\theta}_t = (Z_t^\intercal Z_t + \lambda I)^{-1} Z_t^\intercal y_t$$ rt

where $Z_t = [\phi_{a_1}^\intercal; \ldots; \phi_{a_t}^\intercal] \in \mathbb{R}^{t \times d}$ and $y_t = [r_1; \ldots; r_t] \in \mathbb{R}^t$ are obtained by vertically stacking the observed features and rewards. The associated confidence intervals can be computed as follows:

$$|\hat{r}_t(a) - r(a)| \leq \alpha_t \sqrt{\phi_a^\intercal (Z_t^\intercal Z_t + \lambda I)^{-1} \phi_a} = \beta_t(a)$$

Note that the algorithm can be efficiently written in order to store an incremental estimate of the matrices $Z_t^\intercal Z_t + \lambda I$ and $Z_t^\intercal y_t$ (see lecture slides).

Note: there is a theoretical value for α. However, such a value may be conservatively large in some applications, and so optimizing this parameter may result in higher total payoffs in practice.

2.2 The model

One of the most common datasets that are available for building a recommender system is the MovieLens dataset. There are several versions of this dataset, we consider the one composed of 1M of entries. This dataset contains $1,000,209$ anonymous ratings of approximately $3,900$ movies made by $6,040$ MovieLens users who joined MovieLens in 2000 [Harper and Konstan, 2015].

The dataset has already been preprocessed to extract the movies with at least $N_{\min}$ ratings. These entries are used to build our model M of users, the rating that user i assigns to movie j. We have factored the user-movie matrix ($M \in \mathbb{R}^{n_u \times n_m}$) using a low-rank matrix factorization as $M \approx UV^\intercal$ ($U \in \mathbb{R}^{n_u \times d}$ and $V = \mathbb{R}^{n_m \times d}$). The rating that the user i assigns to movie j is estimated as $\hat{r}_{i,j} = u_i v_j^\intercal$ where u_i is the i-th row of U and v_j is the j-th row of V. The rating $\hat{r}_{i,j}$ is the payoff of pulling arm j when recommending to user i. If you are interested in the preprocessing phase you can find the code in the archive.

Cold start problem. We use linear bandit to solve the cold start problem. A new user arrives and we need to learn his/her features (preferences). We can use the features extracted via low-rank matrix factorization to represent the actions[1] (i.e., movies to suggest to the new user). Formally, we are interested in recovering the vector u^*_{new} under the assumption that $E[r(a)] = u^*_{new} v_a^\intercal$ where v_a is the row of V associated to action a.

Detailed description. We report a detailed description of the model

- $N_{\min} = 1000$.
- Number of users: $n_u = 6040$.
- Number of movies: $n_m = 207$.
- Feature dimension: $d = 30$.
- Arms: the arms correspond to the movies.
- Goal: recover the vector $u^*_{new} = \theta^*$.

We provide a simulator implementing the following function:

- **reward**: receives the index of the arm to execute $a \in \{0, \dots, d-1\}$ and it returns the reward. Note that for the cold start problem the reward is generated from a single user.

These are all the information required for implementing the model. We provide a partial structure of the exercise in *mainTP2_linMAB*. Here you can find even a simple domain (synthetic data) for testing.

2.3 What to do.

Q3: Implement the LinUCB algorithm for the provided MovieLens problem. Compare the performance against the following algorithms (consider a horizon $T = 6000$):

- **random**: A random policy always chooses one of the candidate movies from the pool with equal probability.

[1] This is a common practice in recommender systems.

- **ϵ-greedy**: it estimates each movie's rating; then it chooses a random movie with probability ϵ and chooses the movie of the highest rating estimate with probability $1 - \epsilon$.

Show the convergence of the algorithm using the following metrics

- ℓ_2-norm of the estimated theta w.r.t. the true one (provided as part of the model, property *real_theta*):

$$\|\hat{\theta}_t - \theta^*\|_2$$

- Expected Cumulative Regret

$$\mathbb{E}[R_n] = \mathbb{E}\left[\sum_{t=1}^{n} r_t(a^*)\right] - \mathbb{E}\left[\sum_{t=1}^{n} r_t(a_t)\right]$$

Play with the parameters of each algorithm and report your choice. Remember to average the regret on multiple simulations as done for the Bernoulli exercise.

References

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

Apostolos N Burnetas and Michael N Katehakis. Optimal adaptive policies for sequential allocation problems. *Advances in Applied Mathematics*, 17(2):122–142, 1996.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015. ISSN 2160-6455. doi: 10.1145/2827872.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.