TP1: The stochastic multi-armed bandit

odalric.maillard@inria.fr

January, 2018

I) Setting-up experiments

You are given a python project with 4 packages.

- Algorithms: code of various bandit strategies
- Environments: code of the stochastic bandit environment
- Experiments: code to generate numerical experiments
- Figures: plots output by the experiments

A numerical experiment running a bandit strategy on a bandit problem has the following form

```
for t in range(0,timeHorizon):
        arm = learner.chooseArmToPlay()
        reward,expectedInstantaneousRegret=bandit.GenerateReward(arm)
        learner.receiveReward(arm,reward)
```

You can find a simple example of experiment in the function `OneBanditOneLearnerOneRun` from the file `Experiments_MakeBanditExperiments`.

The class `Environments_StochasticBandits` enables to instantiate a bandit problem:

- a method `createBernoulliArms(minGap)` creates an instance of a bandit problem with Bernoulli distributions on each arm, where `minGap` denotes the minimum gap between the mean of two arms.
- a method `reward,expectedInstantaneousRegret=GenerateReward(arm)` outputs an instantaneous reward as well as the corresponding instantaneous regret when we choose to pull arm `arm`.

The file `Algorithms_MyStrategy` contains a dummy class showing the skeleton of a bandit strategy.

1. Take a look at the file `Experiments_Demo`, and go over each function that is called.
2. Run the file and observe the result of a single experiment with given time horizon T.
3. Create a novel method in the file `Experiments_MakeBanditExperiments` in order to collect data over N independent runs instead of a single one. We are especially interested in getting the cumulative regret $R_{t,n}$ after $t = 1, \dots, T$ steps in run $n = 1, \dots N$
4. Create novel method to visualize
   (a) the histogram of the values $R_{T,n}$, $n = 1, \dots, N$, and
   (b) the average regret $\frac{1}{N}\sum_{n=1}^{N} R_{t,n}$ as a function of $t = 1, \dots, T$.

(bonus) Show also the error bars, or quantiles.

II) The UCB algorithm

We focus here on $[0,1]$-bounded bandits.

We recall that UCB-α pulls at time $t+1$ the arm with highest index:

$$\mu^+_{a,t} = \tilde{\mu}_{a,t} + \sqrt{\frac{\alpha\log(t)}{2N_a(t)}}$$

where $\tilde{\mu}_{a,t}$ is the empirical mean of the rewards gathered from arm a up to time t, and $N_a(t)$ denotes the number of pulls of arm at up to time t. The α-UCB algorithm starts with an initialization phase that draws each arm once, and for $t \geqslant K$, chooses at time $t+1$ arm

$$\boxed{A_{t+1} = \underset{a\in\mathcal{A}}{\text{argmax}}\ \mu^+_{a,t}}$$

1. Implement the standard UCB-strategy that is using $\alpha = 3$.
2. Based on many simulations ($N = 100$ or more), estimate the expected regret of the naive strategy and of the UCB algorithm.
3. Compare the histogram of the cumulative regrets for FTL ($\alpha = 0$) and UCB on a simple Bernoulli arm problem obtained using sufficiently many runs ($N = 100$ or more).
4. Do you think that FTL is a ”safe” strategy?
5. Study the influence of the value of the minimal gap.
6. For a bandit problem of your choice (to specify), draw (on the same plot) regret curves for the naive strategy and the UCB algorithm for several values of α. Which value of α seems to be the best?

III) Other UCB algorithms

Try and compare the following strategies (using regret histogram and mean regret as a function of time)

1. For a Bandit with reward in $[0,1]$, 1 is a valid upper bound on the mean of each arm, thus it makes sense to choose instead: $\text{Argmax}\{\min(\mu^+_{a,t}, 1), a \in \mathcal{A}\}$.

   How does it compare to the vanilla algorithm? What do you conclude?
2. One may try instead using the following index: $\text{Argmin}\{N_a(t) : a \in \text{Argmax}\{\min(\mu^+_{a,t}, 1), a \in \mathcal{A}\}\}$

   What do you observe? What do you conclude?
3. One may consider improved confidence intervals. Try using the following indexes

   - $\mu^+_{a,t} = \tilde{\mu}_{a,t} + \sqrt{\frac{\alpha}{2N_a(t)}\log\left(\lceil\frac{\log(t)}{\log(\alpha)}\rceil\frac{1}{\delta}\right)}$, for $\alpha > 1$ and a small value of δ (say 0.01).
   - $\mu^+_{a,t} = \tilde{\mu}_{a,t} + \sqrt{\frac{(1+\frac{1}{N_a(t)})}{2N_a(t)}\log(\sqrt{N_a(t)+1}/\delta)}$.

   What do you observe? What do you conclude?

IV) Complexity of a bandit problem

Lai and Robbins proved in 1985 that in a bandit problem with arms $\nu_1, \ldots, \nu_A$ that are parametric distributions, the regret is lower bounded, for large values of T as

$$\frac{\mathbb{E}[R_T]}{\log T} \geqslant \sum_{a\neq a^*}\frac{(\mu^* - \mu_a)}{\text{KL}(\nu_a, \nu^*)},$$

where $\mathrm{KL}(\nu, \nu') = \int \log(d\nu(x)/d\nu'(x))d\nu(x)$ is the Kullback-Leibler (KL) divergence between distributions ν and ν'. The Kullback-Leibler divergence between two Bernoulli distribution $\mathcal{B}(x)$ and $\mathcal{B}(y)$ is given by

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

1. Write a function that returns the complexity term of a multi-armed bandit problem with Bernoulli arms. The complexity term c is such that the optimal regret at time T should be $c \log(T)$.

```
c=complexity(MAB)
```

2. Plot the function $t \mapsto c \log(t)$.

V) The family of KL-UCB algorithms and Thompson Sampling algorithms

KL-UCB refers and Thompson Sampling both refer to families of algorithms that explicitly make use of the type of distributions available on each arm. Thus, the algorithm for Bernoulli distribution is different than for Gaussian distributions, or Expoential distributions and so on.

Cappé et al. (2013)

The function `klucbBern(mu, f(t)/ N_t(a), precision)` from the file `Algorithms_kullback` computes the upper-confidence index of KL-UCB at precision `precision`.

1. Use this function in order to code KL-UCB for Bernoulli distributions.
2. Test it against UCB. Conclusion?
3. Implement as well KL-UCB for Gaussian, Exponential and Poisson distributions, using other functions from the file `Algorithms_kullback`.
4. Do not forget to modify the file `StochasticBandit` to generate bandits with Gaussian, Exponential and Poisson distributions (make sure the rewards stay in $[0, 1]$ or are $1/2$-sub-Gaussian).
5. Test different KL-UCB on different bandit problems, adapted or not to the type of distribution.
6. What do you observe? Is there a strategy that approximately dominates all others on all problems?

VI) A Bayesian idea for Bernoulli bandit problems

Consider a bandit problem with A arms that are Bernoulli distributions with means $\theta_1, \ldots, \theta_A$. The UCB algorithm uses confidence intervals on the unknown mean of each arm to make its decision.

In a *Bayesian* view on the MAB, the θ_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 θ_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.

$$\begin{array}{rl} \textbf{Prior distribution} & \theta_a \sim \mathcal{U}([0,1]) \\ \textbf{Posterior distribution} & \theta_a | X_1, ..., X_{N_a(t)} \sim \text{Beta}\left(S_a(t)+1, N_a(t) - S_a(t) + 1\right) \end{array}$$

where $X_1, ..., X_{N_a(t)}$ are the rewards from arm a gathered up to time t, and $S_a(t)$ is the number of rewards equal to 1 received until time t when pulling arm a.

Bayesian bandit algorithms choose an action based on the current posterior distributions over the parameters of the arms,

$$\pi_a(t) = \text{Beta}(S_a(t)+1, N_a(t) - S_a(t) + 1).$$

Thompson Sampling is a simple, randomized, Bayesian algorithm. See also Kaufmann et al. (2012)

At time $t + 1$,

- for each arm a, draw a sample $\theta_a(t)$ from $\pi_a(t)$

- choose
$$A_{t+1} = \underset{a \in Arms}{\text{argmax}}\ \theta_a(t)$$
- update the posterior on arm A_t (posterior distributions on the other arms are unchanged)

1. Implement the TS algorithm for Bernoulli distributions. You can make use of the `beta` distribution from the library `scipy.stats`.
2. Can Thompson Sampling also be called an 'optimistic algorithm'?
3. Test it on a bandit with Bernoulli distributions. For two different bandit problems with Bernoulli arms (that you specify), one 'easy' (small complexity term), and one 'difficult' (large complexity term), compare the regret of Thompson Sampling with that of UCB. Add Lai and Robbins' lower bound on your plots.
4. Test it on other bandits problems, such as bandit with standard deviation $\sigma = 1/2$ and mean $\mu \in [0, 1]$.

VII) A puzzling sub-sampling strategy

The best-empirical subsampled arm (BESA) strategy introduced in the paper Baransi et al. (2014). It is a very simple strategy that proceeds as follows:

For 2 arms: If arm a has been pulled 3 times at time t, and arm b has been pulled 10 times, the algorithm sub-samples 3 observations out of the 10 of arm b, then compares the empirical mean built from b with these 3 samples, to the empirical mean built from a. The chosen arm is the one with the highest such empirical mean, and is called the "winner".

Formally, the winner of the tournament between arm a_1 and a_2 is

$$\underset{a \in \{a_1, a_2\}}{\text{argmax}}\ \tilde{\mu}^{sub}_{a,t}\Big(\min_{a \in \{a_1, a_2\}} N_a(t)\Big)$$

where $\tilde{\mu}^{sub}_{a,t}(n)$ denotes the empirical mean built from n sub-samples chosen uniformly at random amongst the $N_t(a)$ rewards that are available for arm a at time t. See Baransi et al. (2014) for further details.

For A arms, it uniformly randomly shuffles the arm at time t, then organize a pairwise tournament between the arm: each arm competes another one, then we proceed similarly using the $A/2$ winners, and proceed similarly until there is a single winner.

1. Implement the BESA strategy from Baransi et al. (2014) and for 2 arms only.
2. Compare it to KL-UCB and TS for Bernoulli distributions on a bandit with Bernoulli arms.
3. Proceed similarly using other type of distributions (Gaussian, Poisson, etc) and corresponding best strategies.
4. What do you observe? Is there a strategy that approximately dominates all others on all problems?
5. Implement the algorithm for $A = 4$ arms.
6. Implement another variant that does not use random shuffling of arms. What do you observe?

References

Baransi, A., Maillard, O.-A., and Mannor, S. (2014). Sub-sampling for multi-armed bandits. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 115–131. Springer.

Cappé, O., Garivier, A., Maillard, O.-A., Munos, R., Stoltz, G., et al. (2013). Kullback–leibler upper confidence bounds for optimal sequential allocation. *The Annals of Statistics*, 41(3):1516–1541.

Kaufmann, E., Korda, N., and Munos, R. (2012). Thompson sampling: An asymptotically optimal finite-time analysis. In *ALT*, volume 12, pages 199–213. Springer.