Homework 3 Data Science II
Multi-armed Bandits

Instructions

Please prepare a writeup following the general writeup instructions and submit on Brightspace.

Please show your work! This means (i) provide the code you have produced by placing it in an appendix, (ii) next to any answers in your writeup include a parenthetical statement pointing out where in your code the answer was computed. For example: "(See App. 2: plot_timeseries.py, lines 100-110.)" or "(See App. 2: plot_timeseries.py, comment "`#PROB01`") if you cannot include line numbers.

Problem 1. **Simulating MABs**

Implement the MAB model so that you may simulate the use of strategies (see below). The lecture notes and code are helpful here. To do this, you will need to construct well-documented, readable code that:

- sets up the probability distributions and (true) mean rewards for each of the N arms;
- returns a reward r_i taken from distribution D_i when arm i is played;
- implements **gambles**, the sequential playing of arms over T timesteps;
- computes the **regret** R of a strategy as a function of time over the course of a gamble;
- runs efficiently so that you can average over many gambles.

While the *expected* regret discussed in class is a mathematical entity because the expectation is over all possible gambles, here in simulations you can "sample" the expected regret by performing many simulations and averaging R for each one.

Distributions of reward Choose probability distributions D for the rewards that have support on the interval $[0, 1]$. A good choice is the Beta distribution, which has two parameters that let you tune the mean reward value.

In your writeup:

- Describe the multi-armed bandit problem in your own words, why it is a good model for dynamic A/B/n testing, and what limitations it may have for practical problems.
- Describe your implementation of MAB simulations.
- Discuss and motivate your choice of reward distribution. You should probably include plots to visualize your reward distributions.

Problem 2. **Strategies for playing MABs**

Implement with your own code the following algorithms/strategies/policies for playing arms during the course of a gamble:

1. Random
2. (Naive) greedy
3. ϵ-first greedy
4. ϵ greedy

These strategies were discussed in class or described in the lecture notes. Some of these strategies require data structures that store the history of rewards received per arm so that you can compute estimates $\hat{r_i}$ of the mean reward $\bar{r_i}$.

- **Note**: While ***you*** have access to the true parameters of the arms, and you need that information to evaluate performance (Part 3), **the *strategies* must not see that information**. They can only have access to the rewards they receive after they play an arm. Otherwise, you are not implementing a realistic model of the

uncertainty inherent in dynamic A/B testing or, more generally, reinforcement learning. Put another way, if you already know what version of the website is best, why would you be doing A/B testing in the first place?

In your writeup:

- Describe these strategies.
- Describe and document your code implementation.
- Discuss how you made sure the strategies are not "cheating" by seeing information they shouldn't.

Problem 3. **Evaluate MAB performance**

Use what you have built in Problem 1 to implement two different six-armed bandits (i.e., $N = 6$):

1. An **easy** bandit, where the true rewards are distinct enough that you expect it is possible to identify the optimal arm;
2. A **hard** bandit, where the true rewards are such that strategies are likely to struggle finding which arm is optimal. (A hard bandit, but not an impossible bandit.)

In your writeup:

- Evaluate the performances of the strategies (Problem 2) over the course of gambles of length $T = 1000$. For each strategy, include one or plots in your writeup showing:

  1. The regret of an individual gamble (plot $R(t)$ vs. t),
  2. The expected regret averaged over 100 gambles (plot average $R(t)$ vs. t),
  3. Another *metric* of performance different from regret of your own choosing, also averaged over 100 gambles and as a function of time. Put some thought into what metric best evaluates how efficient/accurate the strategies are.

  Any time a reward is received, the regret must be accounted for. Otherwise, strategies could "play for free".

  Organize the above plot(s) to make your writeup easy to read; A single plot with 50 curves on it will be too hard to understand, for example. Include informative captions with all plots.

- Discuss what did you do to make the **easy** bandit easy and the **hard** bandit hard. Include plots of the true reward distributions, one for the easy bandit and one for the hard bandit.

- Describe the second metric you introduced (what it measures and why it is appropriate), and interpret your results and plots to determine what is the best method from Part 2 and why.

Problem 4. **But suddenly, a new contender has emerged**

As per Problem 2, implement the UCB1 algorithm discussed in the reading. Then use the code from Problem 1 to evaluate the performance of this new method as per Problem 3. Make new plots directly comparing this algorithm to the "best" algorithm identified during Problem 3.

In your writeup:

- Describe what UCB1 does in your own words.
- Determine how well it works.
- Discuss why UCB1 works as well (or as badly) as it does.