EXERCISE 4

Instructor: Yuan Zhou

1 Description

In this exercise, we are going to compare the performance of different policies for multi-armed bandits. Suppose we only have two Bernoulli arms denoted by $\mathcal{B}(\mu_1)$ and $\mathcal{B}(\mu_2)$, where μ_1 and μ_2 are *unknown* beforehand. After each play of arm i ($i = 1$ or 2), you will get the reward which is independently generated from distribution $\mathcal{B}(\mu_i)$. (Recall that the Bernoulli distribution $\mathcal{B}(\mu)$ has probability mass μ on outcome 1 and probability mass $1 - \mu$ on outcome 0.) The goal of the policy is to collect as much reward as possible. We assume the total number of plays reaches T.

We will focus on the *regret* of different policies. For each policy, the regret is defined as the expected difference between the optimal rewards assuming μ_1 and μ_2 are known and the rewards collected by the policy. Let π denote the policy. Suppose at time t ($1 \le t \le T$), the choice made by π is i_t and the corresponding reward is X_{i_t}. To avoid clutter, we have removed the dependency on policy in the notations. Under this scenario, the *regret* is defined as

$$\mathcal{R}_T^\pi \overset{\text{def}}{=} \mathbb{E}\left[T \cdot \max\{\mu_1, \mu_2\} - \sum_{t=1}^{T} X_{i_t}\right].$$

It is straightforward to see that to maximize the expected reward is equivalent to minimizing the regret.

2 Policies

In this section, we will introduce the policies you are going to implement and compare. From now on, we will use $\bar{\mu}_i(t)$ to denote the empirical average rewards of arm i before the tth time step (exclusive).

2.1 Greedy Algorithm

As the name suggests, we greedily decide which arm to play at time t according to $\bar{\mu}_1(t)$ and $\bar{\mu}_2(t)$. The details are described in Algorithm 1.

Algorithm 1: Greedy Algorithm

```
1 initialization: play each arm once
2 for time $t = 3$ to $T$ do
3     if $\bar{\mu}_1(t) > \bar{\mu}_2(t)$ then play arm 1
4     else play arm 2
```

2.2 Epsilon-Greedy Algorithm

This algorithm can be seen as a variant of the Greedy algorithm. During each time step t, with probability $(1 - \epsilon/t)$, we greedily decide which arm to play according to $\bar{\mu}_1(t)$ and $\bar{\mu}_2(t)$ and for the left cases, we just randomly play an arm. The details are described in Algorithm 2.

Algorithm 2: Epsilon-Greedy Algorithm (ϵ)

```
1  initialization: play each arm once
2  for time $t = 3$ to $T$ do
3      uniformly sample a random number $a$ from $[0, 1]$
4      if $a \leq (1 - \epsilon/t)$ then
5          if $\bar{\mu}_1(t) > \bar{\mu}_2(t)$ then play arm 1
6          else play arm 2
7      else
8          uniformly sample a random number $b$ from $[0, 1]$
9          if $b \leq 0.5$ then play arm 1
10         else play arm 2
```

You are suggested to set $\epsilon = 1$.

2.3 Explore-Then-Commit

As the name suggests, we will first play each arm the same number of $\tau/2$ times and then play the arm with the maximum empirical average rewards throughout the remaining period. The details are described in Algorithm 3.

Algorithm 3: Explore-Then-Commit (τ)

```
   /* Explore                                                */
1  play each arm $\tau/2$ times
   /* Commit                                                 */
2  for time $t = (\tau + 1)$ to $T$ do
3      play arm $\text{argmax}_{i=1,2}\, \bar{\mu}_i(\tau + 1)$
```

You are suggested to set τ as $\lceil C \cdot T^{2/3} \rceil$ where C is a positive real number. It is also suggested to set $C = 1$.

2.4 UCB

UCB stands for *Upper Confidence Bound*. During each time step t, instead of playing the arm with the maximum empirical average rewards, it plays the arm with the maximum upper confidence bound. Such bound serves as an optimistic estimate of the arm's real mean rewards. The details are described in Algorithm 4. Note that $T_i(t)$ denotes the number of times arm i is played before time step t (exclusive).

Algorithm 4: UCB (α)

```
1  initialization: play each arm once
2  for time $t = 3$ to $T$ do
3      play arm $\text{argmax}_{i=1,2} \left( \bar{\mu}_i(t) + \alpha \cdot \sqrt{\frac{2\ln(t-1)}{T_i(t)}} \right)$
```

You are suggested to set $\alpha = 0.5$.

2.5 Thompson Sampling

Thompson Sampling is a kind of Bayesian algorithm. Before the algorithm starts, it assumes a prior distribution over μ_1 and μ_2. Here we choose uniform distribution i.e, $\mu_1 \sim \text{Beta}(1, 1)$ and $\mu_2 \sim \text{Beta}(1, 1)$, where Beta denotes the Beta distribution.

At time t, we will use the history to derive a posterior distribution over μ_1 and μ_2 which is $\mu_1 \sim \mathrm{Beta}(1+T_1^P(t), 1+T_1^N(t))$ and $\mu_2 \sim \mathrm{Beta}(1+T_2^P(t), 1+T_2^N(t))$, where we have used $T_i^P(t)$ and $T_i^N(t)$ to represent the number of times we get a reward 1 and a reward 0 from arm i respectively before the tth time step. Then, we will sample X_t and Y_t from the posterior distribution and make the decision according to the sampled X_t and Y_t. The details are described in Algorithm 5.

Algorithm 5: Thompson Sampling

```
1 for time $t = 1$ to $T$ do
2     get a sample $X_t$ from $\mathrm{Beta}(1 + T_1^P(t), 1 + T_1^N(t))$
3     get a sample $Y_t$ from $\mathrm{Beta}(1 + T_2^P(t), 1 + T_2^N(t))$
4     if $X_t > Y_t$ then play arm 1
5     else play arm 2
```

3 Experiment Setup

In the experiment, we will use $T = 1000$. Let the *empirical regret* be

$$R = T \cdot \max\{\mu_1, \mu_2\} - \sum_{t=1}^{T} X_{i_t}.$$

For each policy, you will need to run the experiment for 100 trials and report the average empirical regret which is

$$\frac{R_1 + \cdots + R_{100}}{100},$$

where R_k is the empirical regret during the kth trial as the approximation of the real regret.

Let $\mathcal{S} = \{(0.2, 0.8), (0.25, 0.75), (0.3, 0.7), (0.35, 0.65), (0.4, 0.6)\}$. Every item in $\mathcal{S}$ denotes an instance of (μ_1, μ_2). You will need to produce 6 figures.

Data-dependent Regret For every input in $\mathcal{S}$, plot the average empirical regret at time $50, 100, \ldots, 1000$ for every policy.

Minimax Regret For the 6th figure, plot the maximum average empirical regret among all the inputs in $\mathcal{S}$ at time $50, 100, \ldots, 1000$ for every policy.

For the Python language, we have implemented the Uniform Sampling algorithm, which just samples each arm in a round-robin way and generated the figure for data-dependent regret (see Figure 1). To implement a new policy, you only need to inherit the **Learner** class and change the corresponding part of the **main** method. For more details, please check the **code** folder.

If you decide to use another programming language, you need to start from scratch.

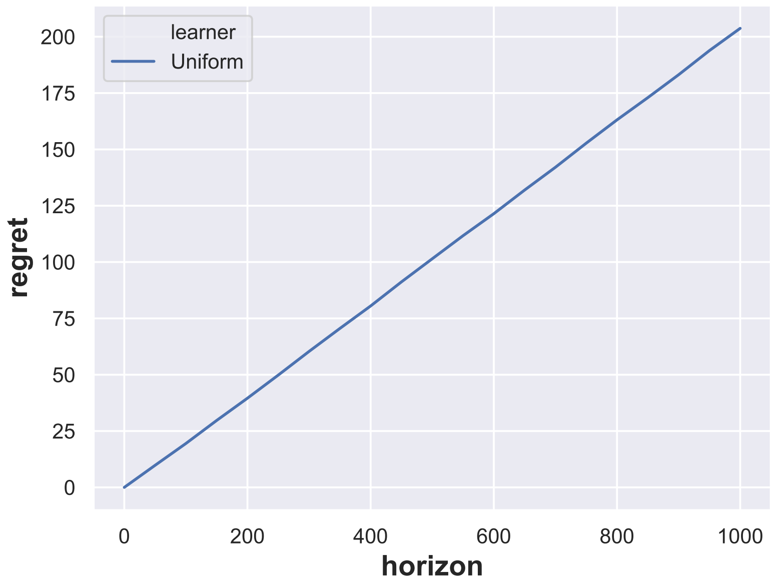

Figure 1: Uniform Sampling

4 Submitting Your Solution

1. Please write a short report with the requested plots and some discussion and put this into a PDF file. In particular, please explain i) do the plots for the UCB algorithm in the data-dependent regret section represent the $\sqrt{T}$-type curve, and why? ii) do the plots for the UCB algorithm in the minimax regret section represent the $\sqrt{T}$-type curve, and why?

2. Please write your code clearly so that someone else reading the code can understand it without significant effort, i.e., structure it and include enough documentation.

3. Please write a README file that explains how the code should be executed.

Finally pack all files included in items (1)-(3) into a zip archive and upload the archive.