Programming Assignment 1

CS 747: Foundations of Intelligent & Learning Agents

(Autumn 2023)

Atishay Jain (210050026)

210050026@iitb.ac.in

Contents

PART

I

Task 1

Section 1

UCB

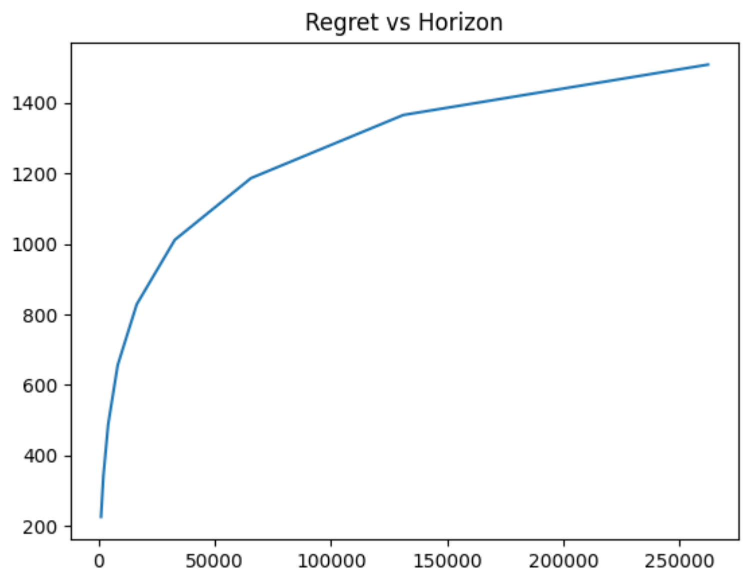

Figure 1. Regret vs Horizon for UCB Algorithm

Subsection 1.1

Implementation

Definition 1
UCB Algorithm -

- At time t, for every arm a, define $ucb_a^t = \hat{p}_a^t + \sqrt{\frac{2\ln t}{u_a^t}}$
- $\hat{p}_a^t$ is the **empirical** mean of rewards from arm a
- u_a^t is the number of times a has been sampled at time t
- Pull an arm a for which ucb_a^t is **maximum**

I created 3 arrays (`ucb, values, count`) for keeping the UCB values, empirical mean and count of pulls of an arm respectively for each arm, and a variable `time`. In `give_pull()`, I returned the arm with maximum UCB value. In `get_reward()`, I incremented `time` and `count[arm]` for the arm pulled, calculated the new empirical mean reward for the arm pulled, and updated `ucb` value for **every** arm. From the plot, we see that UCB achieves sub-linear regret, with regret of ~ 1400 at large horizon

SECTION 2

KL-UCB

Figure 2. Regret vs Horizon for KL-UCB Algorithm

SUBSECTION 2.1

Implementation

Definition 2
For each arm a at time t, we define

$$\text{ucb-kl}_a^t = max\{q \in [\hat{p}_a^t, 1] \text{ s.t. } u_a^t KL(\hat{p}_a^t, q) \leq \ln t + c \ln(\ln t)\}$$

where $c \geq 3$, and at step t pull the arm for which ucb-kl$_a^t$ is maximum. But as proved in a recent paper, that $c = 0$ gives better results, so I used $c = 0$

Method

- I used the arrays and variables almost similar to UCB
- I wrote a separate function (`KL(x,y)`) for finding KL-divergence and added a small term of 1^{-10} to y in order to avoid division by 0
- For finding the KL-UCB value for each arm, I used Binary search on an array with values ranging in $[p_a^t, 1]$ with a gap of 0.01 between its elements, with the binary search condition as $u_a^t \cdot \text{KL}(p_a^t, arr[mid]) \leq \ln(t) + c\ln(\ln(t))$
- I also implemented the bisection method for this purpose using similar conditions but commented it in the final submissions and results.

From the plot, we see that KL-UCB provides a more tougher bound than UCB and also achieves sub-linear regret. The regret achieved is ~ 140 for a large horizon

Section 3
Thompson Sampling

Figure 3. Regret vs Horizon for Thompson Sampling

Subsection 3.1
Implementation

Definition 3
Thompson Sampling -

- At time t, let arm a have s_a^t successes (1's) and f_a^t failures (0's)
- Computational step: For every arm a, draw a sample (in agent's mind) from Beta distribution, that is, $x_a^t \sim Beta(s_a^t + 1, f_a^t + 1)$
- Sampling step: Pull (in real world) arm a for which x_a^t is **maximum**

Method

- I created 3 numpy arrays (`values, success, fail`) for storing samples from Beta distribution, count of 1-rewards, count of 0-rewards for each arm
- In `give_pull()`, I sampled Beta$(s_a^t + 1, f_a^t + 1)$ for each arm and returned the arm having the maximum value sampled
- In `get_reward()`, I updated the `success` and `fail` array values for the arm pulled according to the reward

As seen from the plot, Thompson Sampling performs even better than UCB or KL-UCB and achieves sub-linear regret. The regret is ~ 110 at large horizon

PART

II

Task 2

Section 4

Part A

Figure 4. Regret vs p_2 with $p_1 = 0.9$ using UCB

Subsection 4.1

Observations and Reasoning

- From the above plot, we obverse that as p_2 is increased from 0 to 0.9, keeping $p1 = 0.9$, regret also increased. This is because
  - the UCB algorithm balances exploration and exploitation. When the difference $p_1 - p_2$ is greater, it is easier for the algorithm to identify the best arm quickly, resulting in lower regret
  - As the means of the arms become closer (smaller $p_1 - p_2$), the algorithm takes longer to distinguish between them, leading to higher regret during the exploration phase
- At $p_2 = 0.9 = p_1$ exactly, there is a sharp drop in the plot and the regret is ~ 0, which is justifiable because this means that both arms are almost identical and the algorithm converges to selecting the best arm with minimal regret. Since both arms have identical mean and that too 0.9 (close to 1), it does not matter much which one is selected.
- We know that Regret is lower bounded by Lai and Robins bound as $T \to \infty$

$$\frac{R_T}{\ln(T)} \geq \sum_{a:p_a \neq p_a^*}^{A} \frac{p_a^* - p_a}{KL(p_a, p_a^*)}$$

Also, we have

$$R_T = \sum_{a:p_a \neq p_a^*}^{A} u_a^T(p_a^* - \hat{p}_a)$$

According to the infinite exploration property of each arm in UCB as $T \to \infty$,

$$\lim_{T\to\infty} \hat{p}_a^t = p_a$$

And by using the property of UCB at $T \to \infty$,

$$\lim_{T\to\infty} \hat{p}_a^T + \sqrt{\frac{2\ln(T)}{u_a^T}} = p_a^*$$

$$\implies \lim_{T\to\infty} u_a^T = \frac{2\ln(T)}{(p_a^* - p_a)^2}$$

Putting this in the Regret equation, we also get an upper bound on regret, so we have

$$\sum_{a:p_a \neq p_a^*}^{A} \frac{p_a^* - p_a}{KL(p_a, p_a^*)} \leq \frac{R_T}{\ln(T)} \leq \sum_{a:p_a \neq p_a^*}^{A} \frac{2}{(p_a^* - \hat{p_a})}$$

So, from this bound on regret (for a large horizon), we also see mathematically that as $p_a^* - p_a$ is decreased, regret increases. In our case, $p_a^* = 0.9$, and when we vary p_1 from 0 to 0.9, the difference decreases.

Section 5

Part B

Subsection 5.1

Plot - UCB

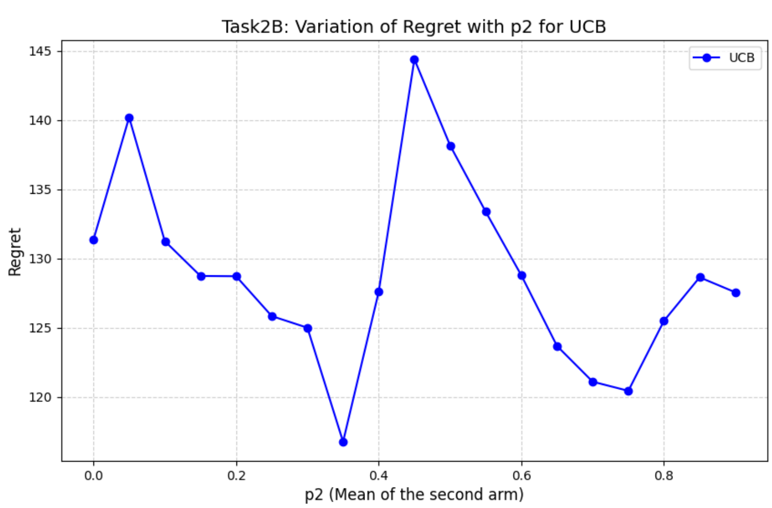

Figure 5. Regret vs p_2 with $p_1 - p_2 = 0.1$ using UCB

SUBSECTION 5.2
Observations and Reasoning - UCB

- From the plot, we observe that the regret does not follow a very regular trend. It sometimes goes to a little high or low value but is almost around the average value of ~ 130.
- This is because as we have seen in the upper bound expression of regret in UCB in task 2(a), at high horizons, regret depends on the difference between the means of arms ($\frac{R_T}{\ln(T)} \leq \sum_{a:p_a \neq p_a^*}^{A} \frac{2}{(p_a^* - \hat{p_a})}$)
- Here, the difference is kept constant ($p_1 = p_2 + 0.1$) and plot is plotted against p_2
- Since the difference is same, we expect the regret to be nearly same, the slight fluctuations are maybe due to noise and limited number of horizons used (30000).
- On increasing the number of simulations and horizon, the regret goes closer to a value and does not vary much because the difference between the means is kept same.

SUBSECTION 5.3
Plot - KL_UCB

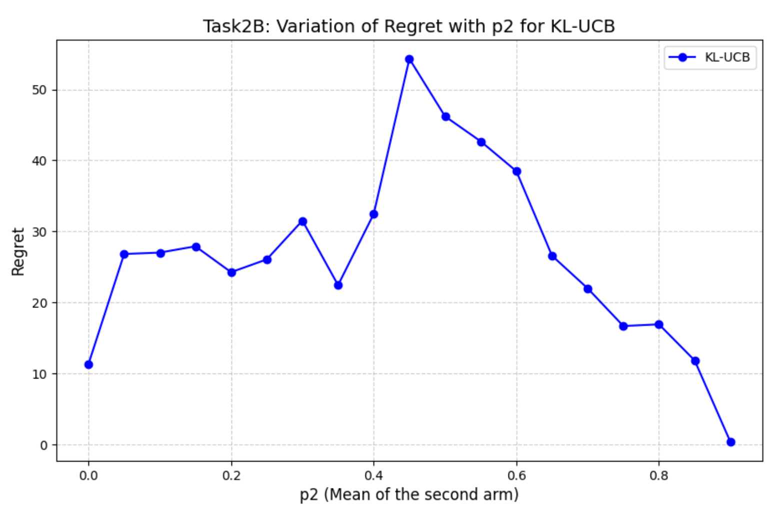

Figure 6. Regret vs p_2 with $p_1 - p_2 = 0.1$ using KL-UCB

SUBSECTION 5.4
Observations and Reasoning - KL_UCB

- From the plot, we observe that the regret increases when p_2 goes from 0 to ~ 0.5 and then decreases as it goes further from ~ 0.5 to 0.9, and it is almost zero at $p2 = 0.9$
- This is because we know that Regret of KL-UCB asymptotically matches the Lai and Robbin's lower bound, whose curve also goes in the same way

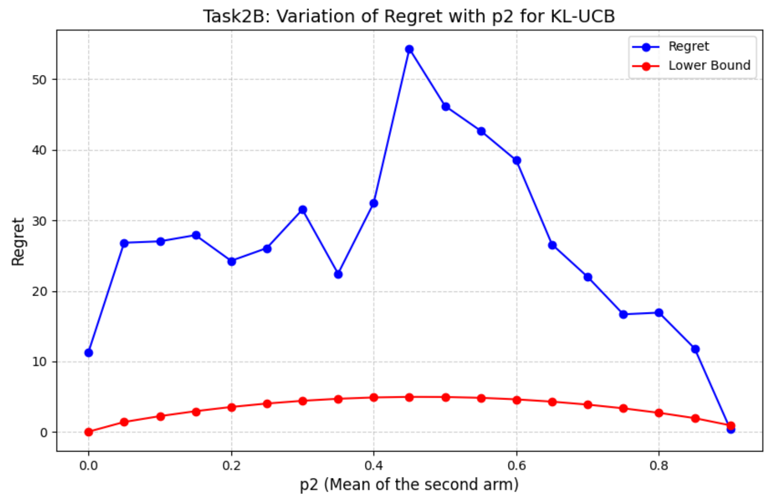

Figure 7. Regret vs p_2 with $p_1 - p_2 = 0.1$ using KL-UCB along with Lower Bound

The above plot also shows the Lai and Robbins lower bound along with KL-UCB's regret. So, we know that KL-UCB approaches this bound as $T \to \infty$, this leads to the kind of curve we obtained for KL-UCB, in which a little peak is around the middle and very less values at corners.
I wrote code for this plot also in `task2.py`, and commented it in the final submission as it was an additional plot.

PART

III

Task 3

Section 6

Algorithm and Approach

I used **Thompson Sampling** algorithm with a slight modification according to the faulty bandit problem.
Let the probability that the bandit returns a faulty pull be p. Using the same notation as used in slides, we maintain a belief distribution over $w \in W$ (viewing each arm's mean p_a as world w) with the evidence samples (rewards from arms) $e_1, e_2, \ldots e_t$ being produced by unknown world w. Now, the expression for our belief after refining it based on incoming e_{t+1} evidence is -

$$\text{Belief}_{t+1}(w) = \frac{\text{Belief}_t(w)P(e_{t+1}|w)}{\sum_{w' \in W} \text{Belief}_t(w')P(e_{t+1}|w')}$$

Now, when we received the evidence e_{t+1} with faulty probability p, if e_{t+1} is a 1-reward, we must set for $w \in [0,1]$

$$\begin{aligned} P(e_{t+1}|w) &= (1-p)w + p\Big(\frac{1}{2}\cdot 1 + \frac{1}{2}\cdot 0\Big) \\ &= (1-p)w + \frac{p}{2} \\ \implies \text{Belief}_{t+1}(w) &= \frac{\text{Belief}_t(w)\big((1-p)w + \frac{p}{2}\big)}{\int_{y=0}^{1} \text{Belief}_t(y)\big((1-p)y + \frac{p}{2}\big)} \end{aligned}$$

and if e_{t+1} is a 0-reward, we must set for $w \in [0,1]$

$$\begin{aligned} P(e_{t+1}|w) &= (1-p)(1-w) + p\Big(\frac{1}{2}\cdot 1 + \frac{1}{2}\cdot 0\Big) \\ &= (1-p)(1-w) + \frac{p}{2} \\ \implies \text{Belief}_{t+1}(w) &= \frac{\text{Belief}_t(w)\big((1-p)(1-w) + \frac{p}{2}\big)}{\int_{y=0}^{1} \text{Belief}_t(y)\big((1-p)(1-y) + \frac{p}{2}\big)} \end{aligned}$$

which we can achieve by taking

$$\text{Belief}_t(w) = \text{Beta}_{s+1,f+1}\Big((1-p)w + \frac{p}{2}\Big), \quad w \in [0,1]$$

Since $w \in [0,1]$, the above Belief can be taken as

$$\text{Belief}_t(w) = \text{Beta}_{s+1,f+1}(\hat{w}), \quad \text{where } \hat{w} \in \Big[\frac{p}{2}, 1 - \frac{p}{2}\Big]$$

So, this is like sampling from a Beta distribution with a cutoff range of samples, that is the samples should be in $\left[\frac{p}{2}, 1 - \frac{p}{2}\right]$. Therefore, I used Thompson sampling, with a modification that the samples drawn from Beta distribution should be in this range. To implement this, I did that if a sample that I got lies in this range, then go ahead, else keep sampling until you get the sample in this range.

PART

IV

Task 4

SECTION 7

Algorithm and Approach

- I used **Thompson Sampling** algorithm for each bandit instance, but when selecting the arm to pull, I took the average of the samples drawn from beta distribution.
- This is because the problem of having two bandit instances at once involves a dilemma. When we select an arm, the arm of that index can be pulled randomly from any of the two bandit instances. As we pull the arm that has the maximum value sampled from beta distribution, it can happen that the maximum of one bandit instance may or may not align with the second bandit instance.
- According to Thompson Sampling, I will sample the arm with maximum Belief. Say for the first bandit instance, I have $\text{Belief1}_t(w)$ and for other $\text{Belief2}_t(w)$
- For maximizing my belief over both instances by taking them as a single system, I calculated
$$\frac{\text{Belief1}_t(w) + \text{Belief2}_t(w)}{2}$$
for each arm. This ensures that when anyone of the two arms is selected at random for the arm for which this term is maximized, the probability of getting a reliable maximum score from both is more.
- And these Belief terms are drawn from Beta distribution for each arm, as done in normal Thompson sampling, just the selection of arm to pulled is done based on maximizing the **average sampling** of the two instances for respective arms in them.