CS 747 Assignment 1

Teja Bale, 200050020

Autumn 2023

Contents

1 Task1

Implemented UCB, KL-UCB, and Thompson Sampling algorithms in Task 1.

1.1 UCB Algorithm

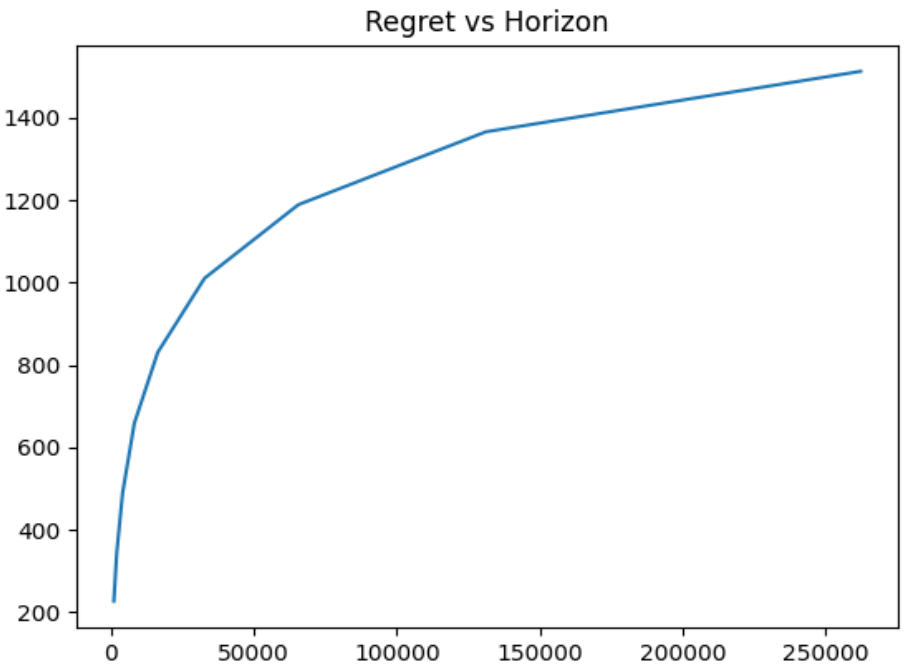

Figure 1: Regret vs Horizon for UCB Algorithm

Initially, I pulled every arm once. After that, for each pull, I calculated the ucb values for all the arms and returned the arm with the highest ucb value. The value of ucb is given by

$$ucb_a^t = \hat{p}_a^t + \sqrt{\frac{2\ln(t)}{u_a^t}}$$

In the above equation, $\hat{p}_a^t$ represents the empirical probability for an arm a at time t, and u_a^t represents the number of times an arm a has been sampled upto time t.

After the pull, we will get the reward. From the reward value, I have updated the values for $\hat{p}_a$ for pulled arm and the number of pulls in the get_reward function.

We can see that the UCB performs significantly better when compared to the Epsilon greedy algorithm.

1.2 KL-UCB Algorithm

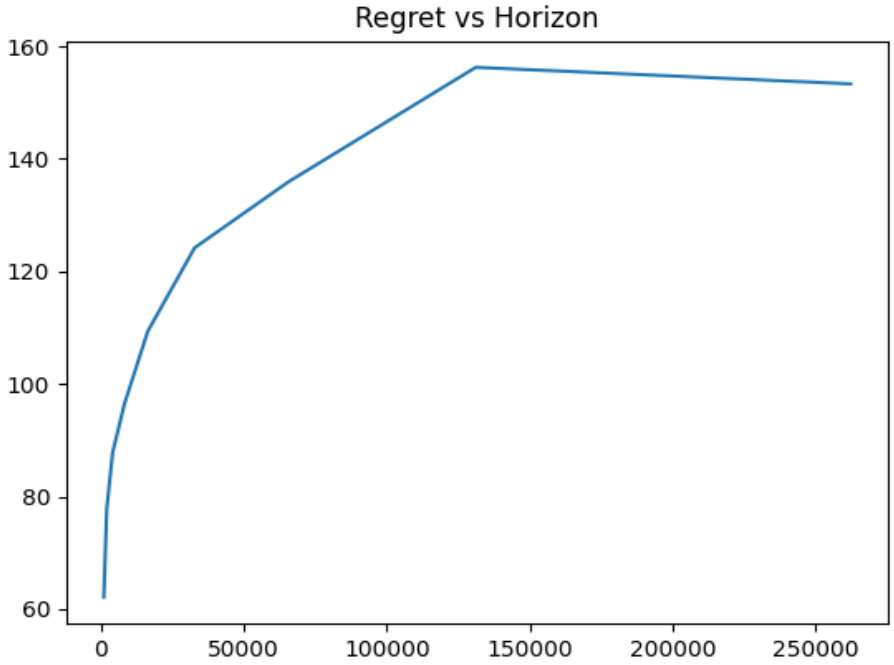

Figure 2: Regret vs Horizon for KL-UCB Algorithm

Similar to UCB, Initially, I pulled every arm once. After that, for the subsequent pulls, I calculated the kl-ucb values for all the arms using the below equation and returned the arm with the maximum value.

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

Here, I have taken the value of the $c = 0$. Also, $KL(p, q)$ denotes the KL divergence between p, q and is defined as $KL(p, q) = p\ln(\frac{p}{q}) + (1-p)\ln(\frac{1-p}{1-q})$.

Basically, the kl-ucb value for an arm can be found as the solution to the equation :

$$KL(\hat{p}_a^t, q) = \frac{\ln(t) + c\ln(\ln(t))}{u_a^t}$$

I implemented a function get_klucb to solve the above equation. I used Binary search to solve the equation. Initially, the interval for q will be $[\hat{p}_a^t, 1]$. We calculate the value of the difference in LHS and RHS of the above equation at the midpoint of the interval, and based on the sign of the difference, we update the interval accordingly. We do this until we get an interval of size $1e^{-6}$.

We can see KL-UCB gives us a great improvement in performance. Not only is the regret sub-linear, but the rate of increase in regret with horizon also decreases greatly. However, there is a tradeoff. KL-UCB takes a considerably higher amount of time compared to other algorithms because of the equation solving involved.

1.3 Thompson Sampling

Figure 3: Regret vs Horizon for Thompson Sampling

The implementation of Thompson sampling was pretty straightforward. I maintained two arrays, alphas and betas, storing the number of successes+1 and failures+1 for each arm so far. For each pull, I obtained a sample from the following distribution for all the arms in an array and then returned the arm with the highest value obtained: Beta(alphas, betas). This returns the samples taken from all the arms in an array.

So far, Thompson sampling has outperformed the other algorithms in terms of performance. Even at large values of the horizon, the regret incurred by Thompson sampling is far less than the other algorithms.

2 Task2

2.1 Part A

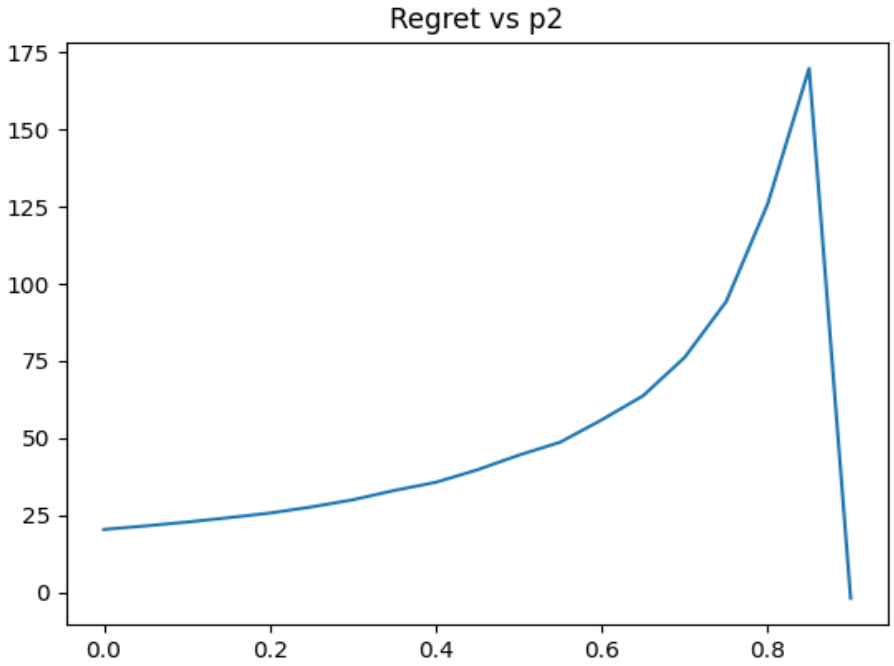

Figure 4: Regret vs p2

The above graph of regret vs p2 for UCb algorithm is generated by taking the number of simulations equal to 50 and p1 = 0.9

Observations: In the graph, as the p2 increases the regret increases and when p2 becomes equal to p1, there is a sudden decrease in the regret, and it becomes almost zero.

Reasons:

1. When p2 is zero even after some rounds, the $\hat{p}_a^t$ is also zero, so every time when we calculate the value of ucb_a^t is high for the p1 and we pull the arm 1 in every round and it has very high probability of reward 1, so the regret is very less in this case.
2. So as the p2 increases from zero, the ucb values of arm 2 also become comparable to arm 1, and we have to pull arm 2 also, so when we are pulling arm 2, there is a more probability of getting reward 0 (since $p2 < p1$), therefore, the regret increases the p2 increases.
3. when p2 becomes equal to p1, it doesn't matter choosing which arm to pull since both arms have an equal probability of getting reward 1, so the regret becomes much less in this case.

2.2 Part B

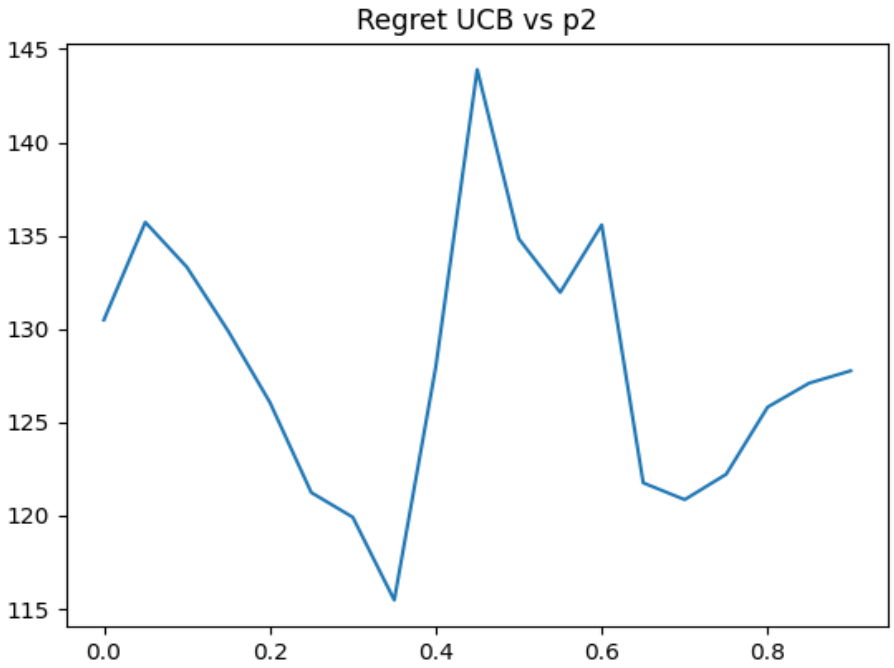

Figure 5: Regret UCB vs p2

The above graph of regret vs p2 for the UCb algorithm is generated by taking the number of simulations equal to 50 and keeping p1 - p2 = 0.1

Observations: The regret value change is not regular. First, it decreases, then increases, then decreases, and finally, it is the same as the initial value. Despite the increase in the probability values (p1 and p2), the regret still increased.

Reason: Maybe this happened because the $\hat{p}_a^t$ values for both arms are close values, so the values of the ucb are also very close to each other (I have checked these differences in the code by printing the values), choosing the arms depending upon the ucb values, but ucb values depend upon a number of times it has been chosen before more than on the $\hat{p}_a^t$, so we can't say which arm will be pulled more with this.

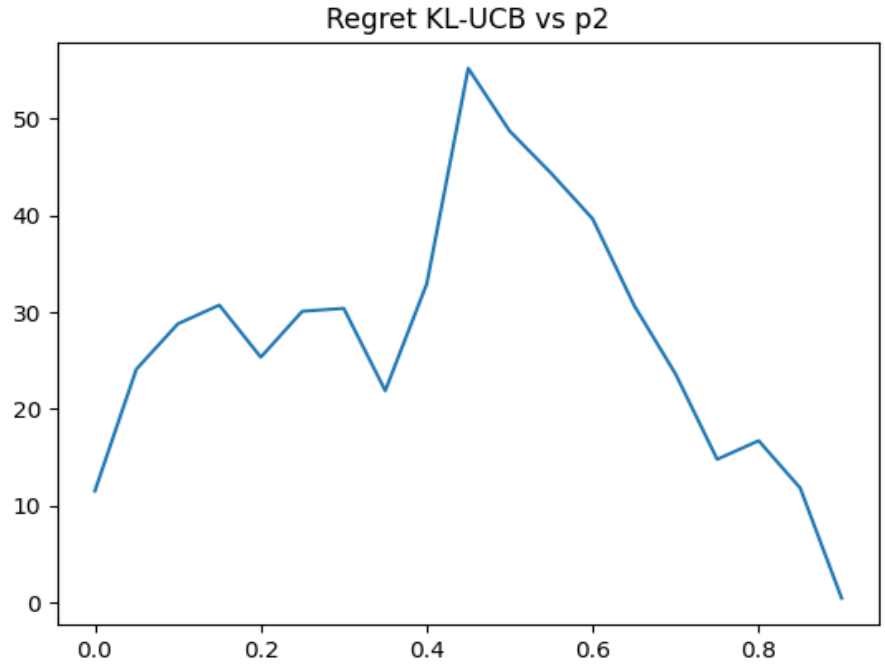

Figure 6: Regret KL-UCB vs p2

The above graph of regret vs p2 for KL-UCb algorithm is generated by taking the number of simulations equal to 50 and keeping p1 - p2 = 0.1

Observations: This is similar to the case with the UCB algorithm, but the regret values are low compared to the UCB algorithm, but when the final p2 becomes 0.9, the regret became very low.

Reasons: Maybe this also happened because of close values of kl-ucb for both arms. The choice of arms from kl-ucb values is in the difference around 0.001. But when the probabilities became high, the regret started to decrease. Maybe this dominated the choice arms (since both have almost the same values)

3 Task 3

For this problem, the algorithm I implemented is almost similar to the Thompson sampling. But when there is success or failure, the increase in the values of the number of successes or failures is slightly changed. Instead of an increase in 1, I have increased with $\frac{1-\frac{f}{2}}{1-f}$.

The probability of getting reward 1 is $f(0.5) + (1-f)p_a$, But after N rounds, the successes are $N(f(0.5) + (1-f)p_a)$, but the original number of success are Np_a, therefore I calculated the $N * p_a$ from the above value, which is $\frac{\alpha^{'} - (Nf/2)}{1-f}$, so incremented $\frac{1-\frac{f}{2}}{1-f}$ instead of 1

After this, the reward has little increase compared to the normal method.

4 Task 4

In this problem, the algorithm I implemented is similar to the Thompson sampling method, But here, the bandit instance is chosen randomly (with a probability of 0.5), so choosing the arm based on just one instance of bandit is not sufficient. Therefore, after samples were drawn from the Beta distribution for both instances, I have taken the average values for all the arms for both bandit instances (since both are similar). I have pulled the arm with the maximum value of the averages.