CS747 Assignment 1: Multi-Armed Bandits

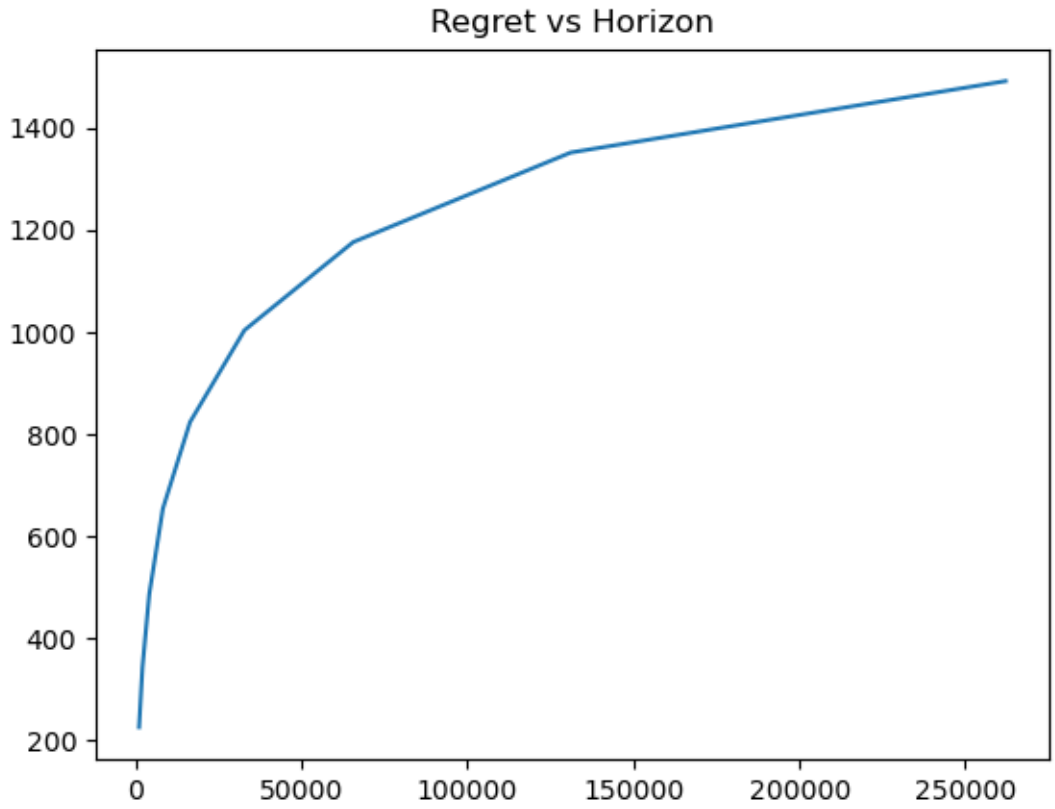
Ninad Chaphekar, roll no. 200010016

September 10, 2022

TASK 1:

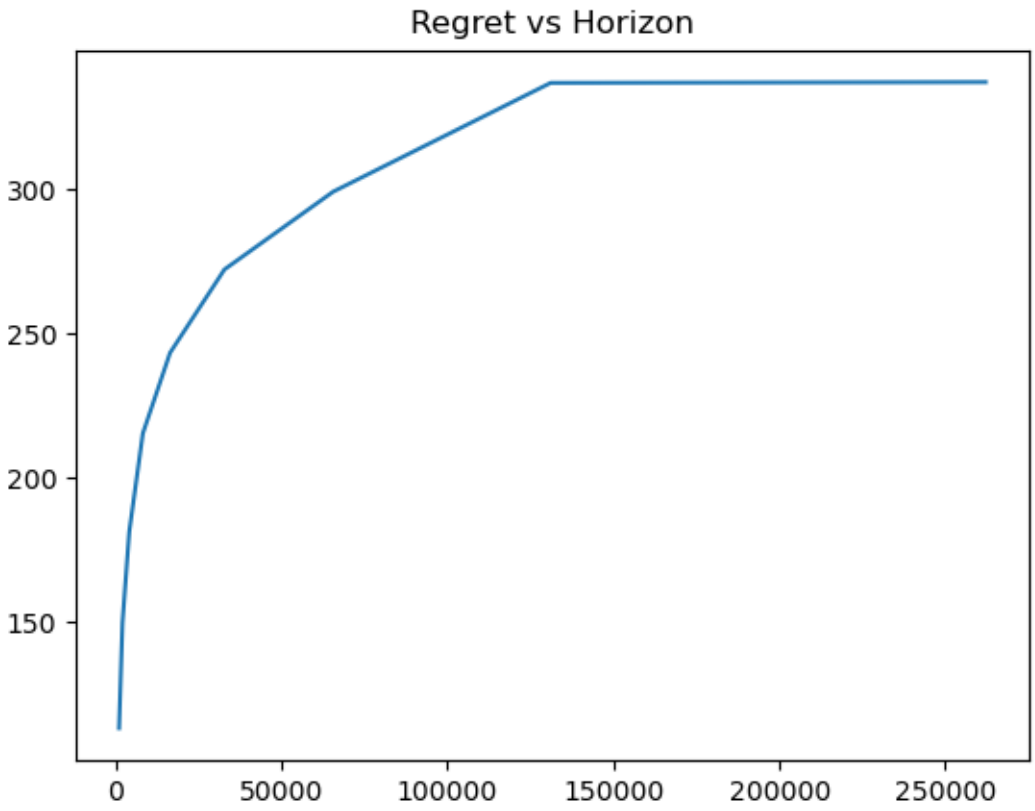
Implemented some standard Multi-Armed Bandit Algorithms discussed in class, like
(1) **Eps-Greedy** (already implemented)
(2) Upper Confidence Bound (**UCB**)
(3) KL Divergence -Upper Confidence Bound (**KL-UCB**)
(4) **Thompson Sampling**.

- **UCB:** At time t, for each arm a, $ucb_a^t = p_a^t + \sqrt{\frac{2\log(t)}{u_a^t}}$ where p_a^t and u_a^t are the emperical mean and the number of times sampled for arm a at time t. We pull the arm a with the highest value of $ucb_a^t \in (0, 1)$. Note, p_a^t changes only for the arm which is pulled, whereas ucb_a^t changes for all. Since ucb_a^t is undefined for arms who haven't been pulled atleast once before ($u_a^t = 0$), we pull all the arms once, in a Round-Robin fashion and then employ the UCB algorithm henceforth. As discussed in the class, this algorithm achieves Greed in the Limit (GLIE) <==> sub-linear regret. We can confirm this by looking at simulator plot of Regret vs Horizon.



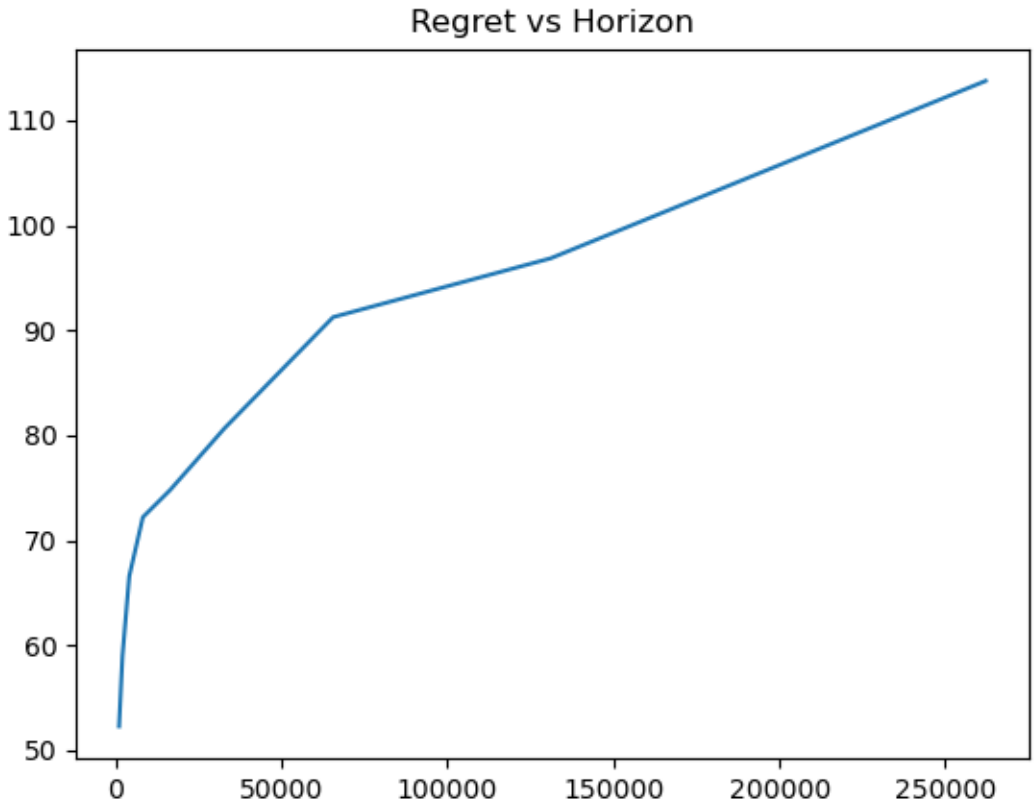
UCB Algorithm

- **KL-UCB:** Similar to the UCB algorithm, at time t, for each arm a, $kl - ucb_a^t = q \in [p_a^t, 1] \ni KL(p_a^t, q) = \frac{\log(t)+C*\log(\log(t))}{u_a^t}$, where KL is KL-divergence, given by $KL(x, y) = x\log(\frac{x}{y}) + (1-x)\log(\frac{1-x}{1-y})$. Finding a solution to this equation analytically is very difficult, hence we resort to numerical methods. One such numerical method used in this implementation is the binary search method. With $f(q) = KL(p_a^t, q) - \frac{\log(t)+C*\log(\log(t))}{u_a^t}$, we find an approximate root to $f(q)$ using an iterative approach, till we reach at a point where $f(q) <= threshold$ (set as 10^{-4}). $KL(p_a^t, q)$ increases monotonically with q, hence a q does exist which satisfies the above equation. Reason since binary search is used is that it is simple and converges quickly in the sense that other methods like Newton-Rhapson involve computation of derivatives which is a tedious task in itself. KL-UCB gives a tighter bound than UCB and achieves sub-linear regret, confirmed from the simulator plot below.



KL-UCB Algorithm

- **Thompson Sampling:** At time t, let arm a have s_a^t successes and f_a^t failures. $s_a^t + f_a^t = u_a^t$ Beta($s_a^t + 1, f_a^t + 1$) represents a 'belief' about the true mean of arm a, where Beta(α,β) represents the Beta distribution with parameters α and β. We pick a random sample $x_a^t \sim Beta(s_a^t + 1, f_a^t + 1)$ We pull the arm which has the highest value of x_a^t. As discussed in class, this algorithm achieves sub-linear regret, confirmed by the simulator plot below



Thompson Sampling Algorithm

TASK 2:

Implemented a batched-algorithm for Multi-Armed Bandits. Here, the arms are pulled in batches instead of one by one in TASK 1. Thus, batch size evenly divides the horizon. Each batch pull consists of the arms that need to be pulled at that time step, and the number of times those arms need to be pulled.

Approach: Run Thompson Sampling batch-size number of times. This gives us the batch of arms to be pulled and the number of times they need to be pulled. The logic for this approach is that, in the early stages of the pulls, each Beta distribution is flatter and away from their true means p_a^t. This leads to batch having more number of distinct arms to be pulled, encouraging exploration. Once we reach a sufficient amount of pulls and the Beta distribution closely represents the true mean, we will get a single arm obtained from the sampling more number of times, encouraging exploitation and obtaining more reward. This is all possible due to the nature of the Beta distribution and Thompson Sampling. Note, this is still not as good as Thompson Sampling itself, as we do not ”pull the argmax or best arm” every pull, rather do it after every batch-size number of pulls.

In this approach, we achieve linear plot for regret vs batch-size, as we expect.

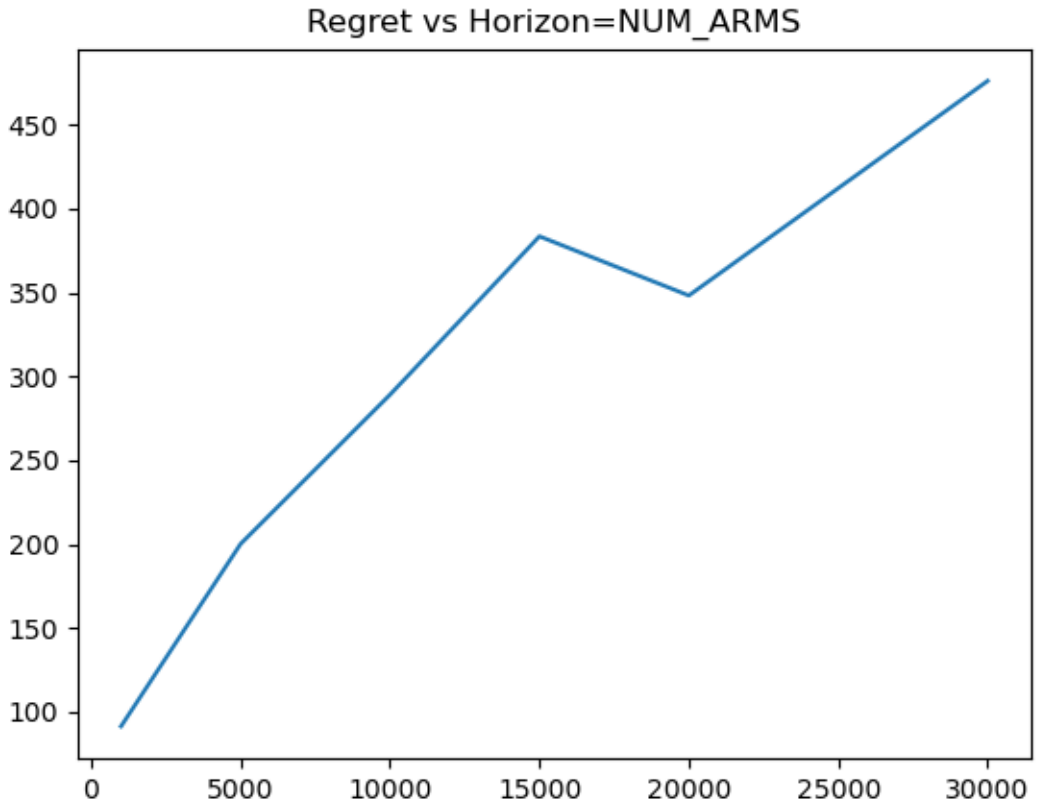
Batched Multi-Arm bandit

TASK 3:

Implemented an algorithm when the horizon is equal to the number of arms. Here, we do not have enough time for optimal exploration and exploitation.

Approach: We select a certain number of arms randomly and run Thompson Sampling on them. The idea behind this is with a appropriately selected number of arms, there is good chance that those arms will include a few arms having very high true means, which will eventually be captured by Thompson Sampling. The number of arms to be selected randomly is a parameter that can be optimized. Here, we have chosen it to be the square root of number of arms. When the number of arms are small, the square root will still be quite a comparable number, and thus have more chance that the optimal arm is chosen, exploiting the fact that the number of arms is small. When the number arms are large, its square root will be small enabling better convergence for Thompson Sampling and pulling the 'best arm' from the selected bunch, even if it isn't the actual optimal arm, but is close.

In this approach, we achieve the following plot for regret vs horizon:



When Horizon = Number of arms

THE END