CS747: Programming Assignment 1

Report by: Atharva Mete | 190010012

Introduction

In this assignment, I implemented and compared different algorithms for sampling the arms of a stochastic multi-armed bandit. The implemented algorithms are epsilon-greedy exploration, UCB, KL-UCB, Thompson Sampling, and a variation of the Thompson Sampling for tasks 3 and 4.

Tasks

Task 1: Implementation of basic sampling algorithms

Epsilon-greedy

In this, I implemented the epsilon-greedy-3 algorithm. Here, I'm not initializing the arms, so starting empirical mean of each arm is zero. Then I draw a random number between [0,1] (uniformly) and if the number is less than epsilon then we explore (sample arms uniformly), if the number is greater than epsilon then we exploit (pull the arm with highest empirical mean).
Tiebreaker: If more than one arms have the same max empirical mean, then we draw one of them uniformly at random.

UCB

Here, I'm allotting first n pulls for initializing the arms (where n is the total number of arms). Thus the empirical mean of all the arms gets initialized. From t=n+1, we start pulling the arm with maximum UCB. If more than one arms have the same max UCB, then we draw one of them uniformly at random.

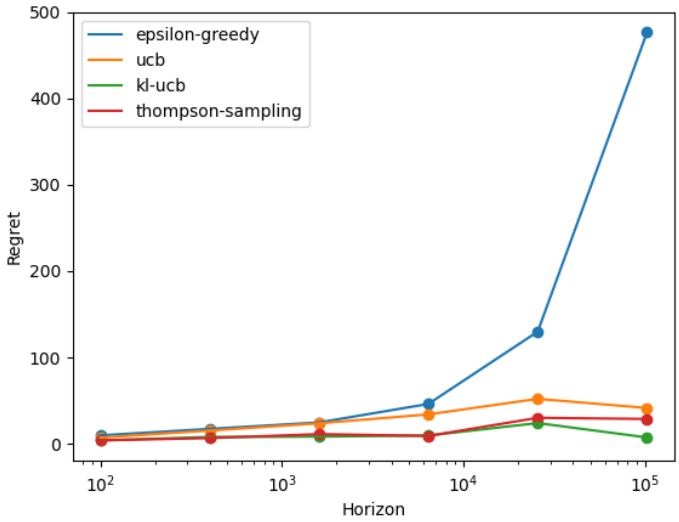
KL-UCB

Like UCB, here too I'm allotting first n pulls for initializing the arms. I have taken the parameter c = 3. For calculating the ucb-kl (max value of q in [emperical-mean,1]), that satisfies the inequality, I performed the binary search algorithm. The algorithm reaches the optimum value within 10 iterations.
I have added two checks to avoid errors like 'division by zero' and 'invalid value in log', encountered while calculating KL value. In order to rectify these errors, I added a small

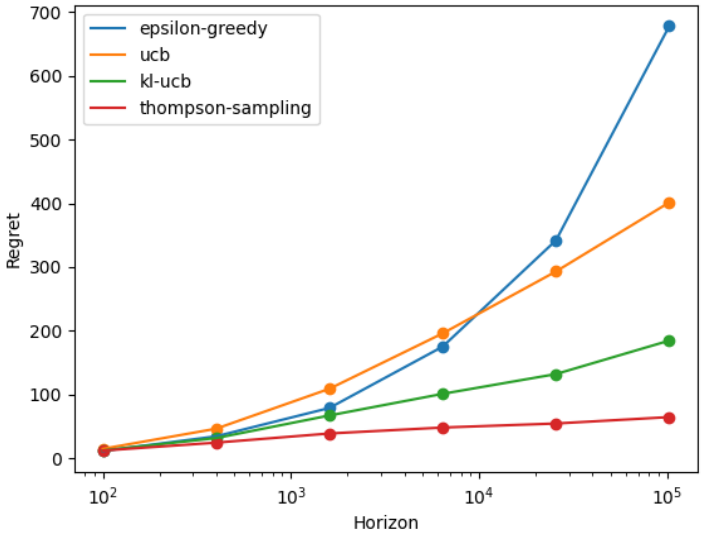
number ‘0.00001’ to the empirical mean. After we calculate ucb-kl of each arm, we pull the one with maximum ucb-kl. Tiebreaker is the same as in UCB.

Thompson Sampling

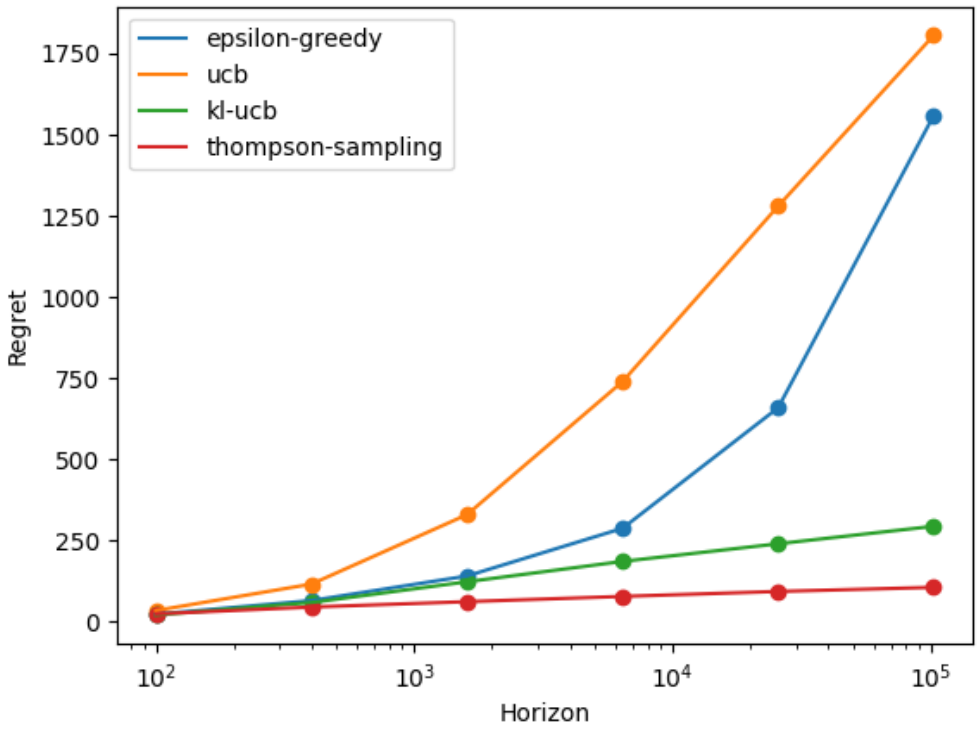
In this, it’s not required to initialize the arms. For every arm, we draw a sample (belief of the true mean) according to the beta distribution and then pull the one with the maximum value of the sample. I update success and failure accordingly. Tiebreaker is the same as in UCB.



Instance 1



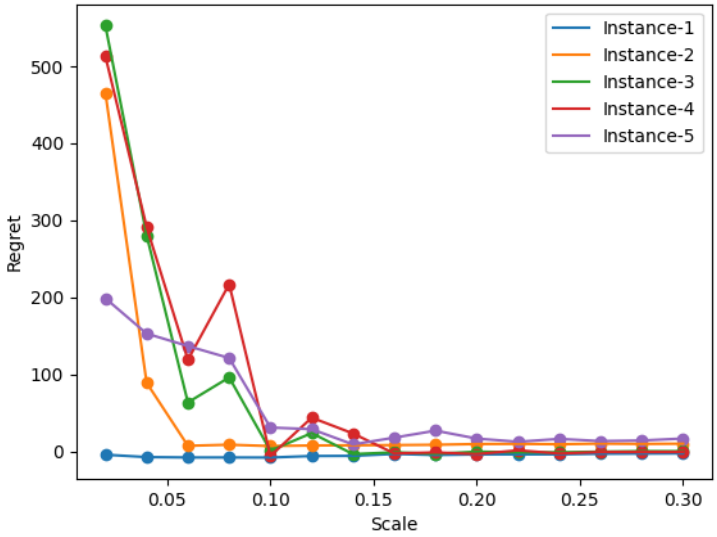
Instance 2



Instance 3

Task 2: Optimise the scale ‘c’

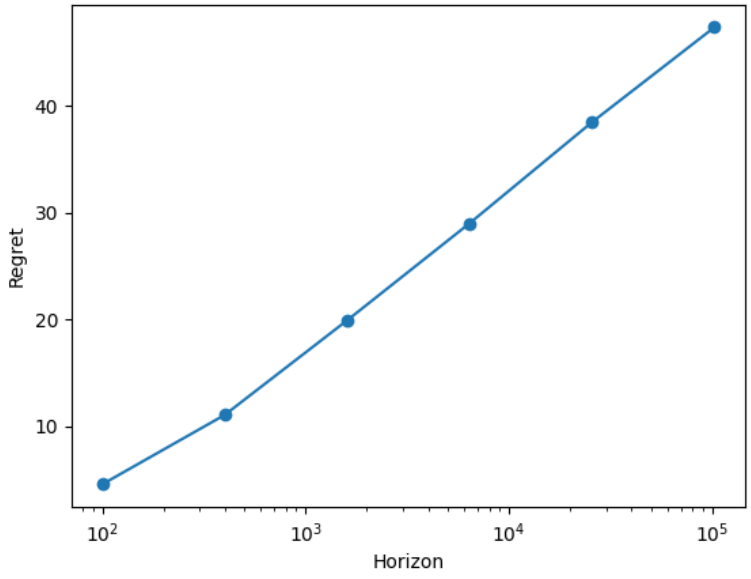
In task 1 we took the value of c to be 2, here we will see the variation of regret with the scale. The value of c controls the amount of exploration.



From graph we observe that c = 0.3 gave lowest regret and regret decreases with increase in c

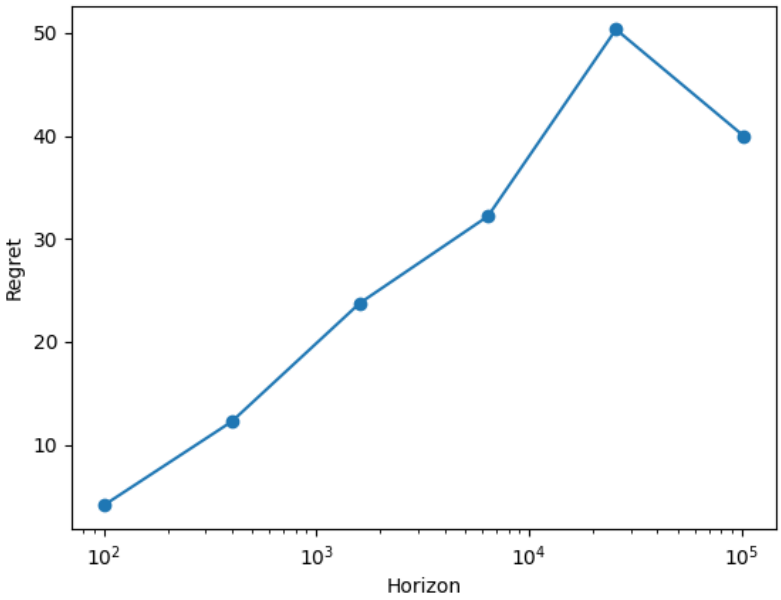
Task 3: Reward with distribution from finite support

For this, I used the Thompson sampling algorithm with a bit of modification. I changed the pull arm function so as to give output from the support according to the given distribution. The reward is added to the reward and ‘1-reward’ is added to the failure. This is analogous to what we did earlier in Thompson sampling where ‘1-reward = 0’ Then, to find the maximum expected cumulative reward, I find the expected value of reward for each arm and then choose the one with the maximum expected value. The tiebreaker is same as mentioned earlier.

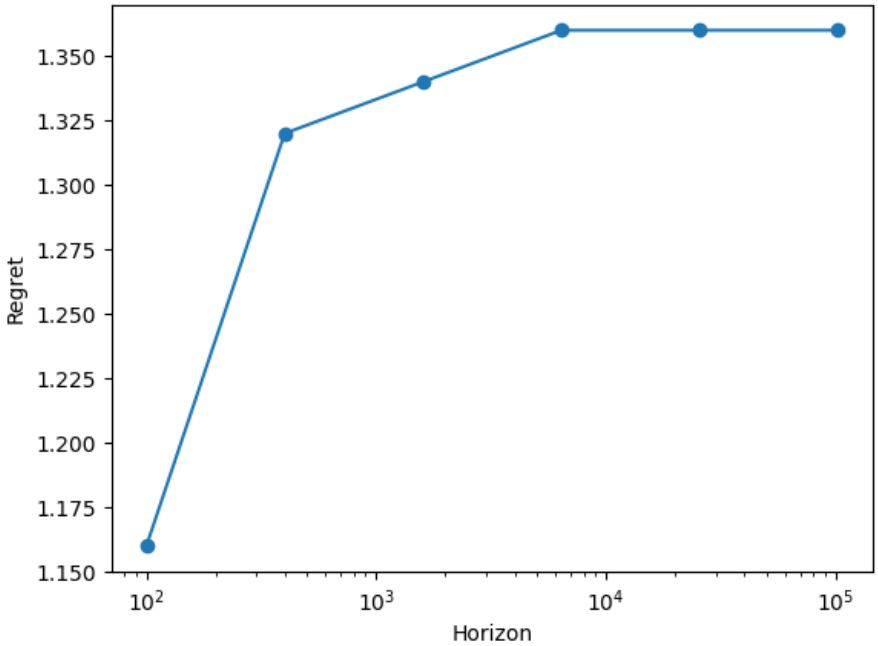


Task 4: Maximise reward exceeding a threshold

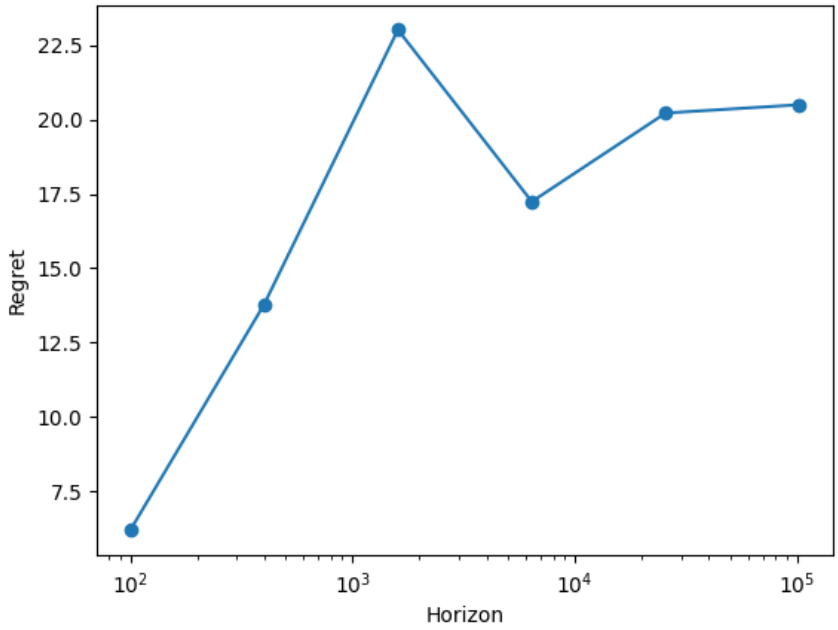
For this, I used the Thompson sampling algorithm with a bit of modification. The pull arm function is the same as task 3. The success and failure were updated according to the following strategy where if the reward is greater than the threshold then add one to the success else add one to the failure. And to calculate maximum cumulative reward, only reward greater than the threshold were considered in calculating the expected reward of each arm.



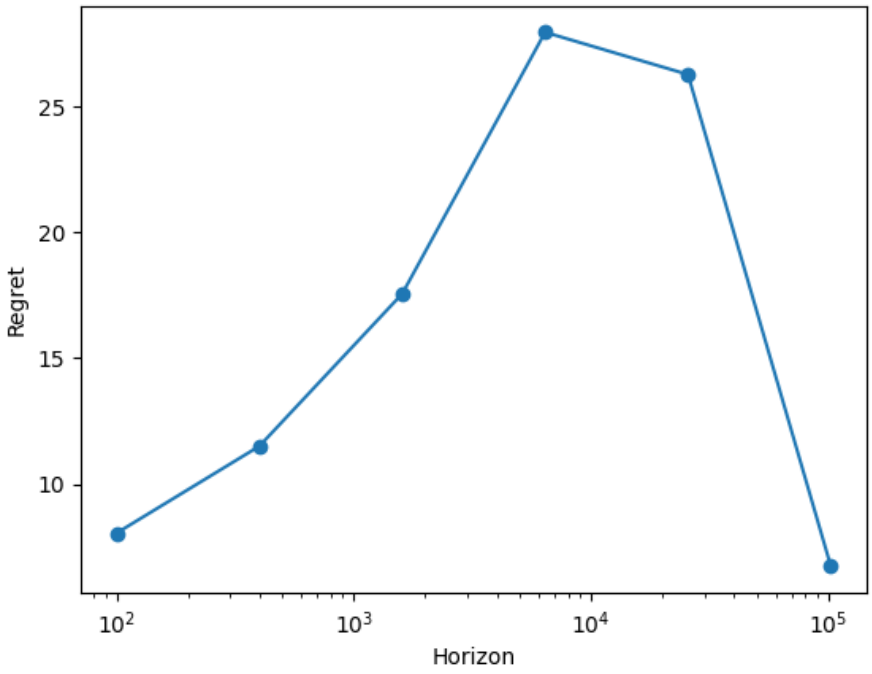
Instance 2 | Threshold 0.2



Instance 1 | Threshold 0.2



Instance 1 | Threshold 0.6



Instance 2 | Threshold 0.6