CS747
Programming Assignment 1 - Report

submitted by

Dhakne Ajay Sopan

190050033

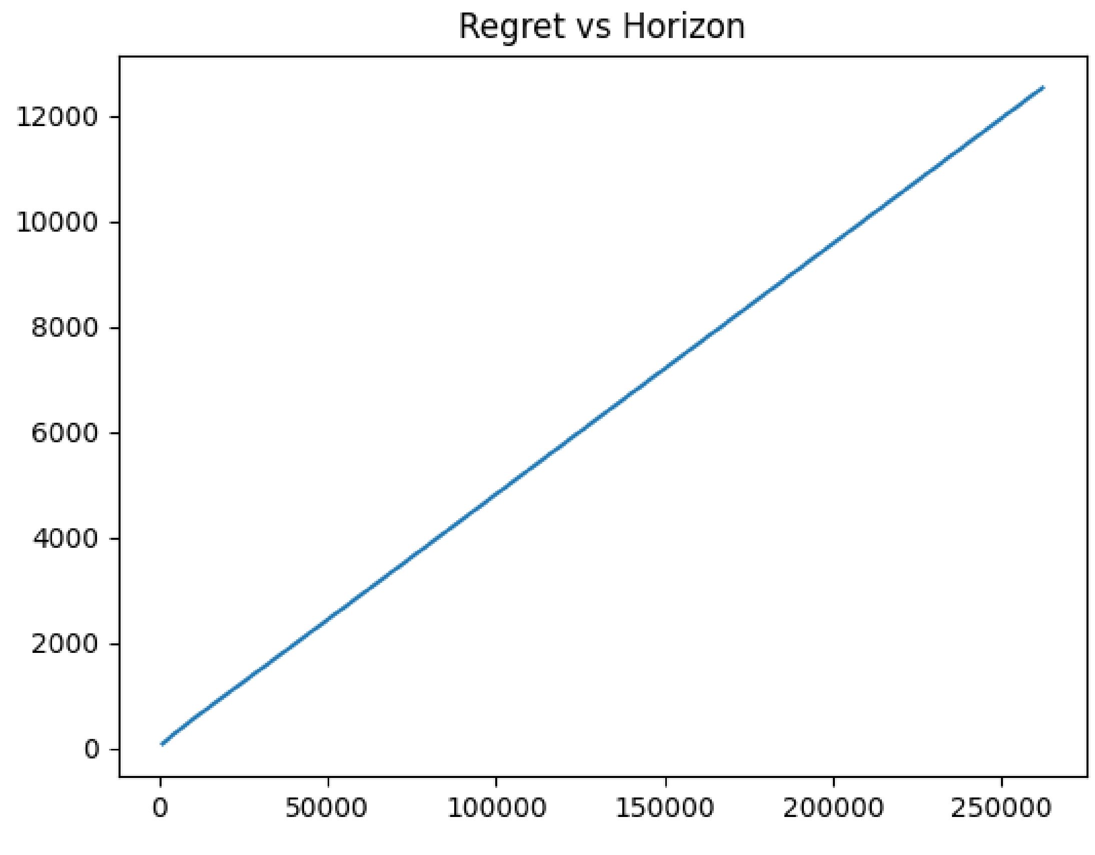
Task1:

Epsilon Greedy:

Exploration is done with probability ε and exploitation with probability 1 - ε.

As seen from the plot above the regret achieved is linear.
We need to keep ε small for large horizon since we should exploit arm with higher mean reward i.e we need less exploration in case of larger horizon.

Plot1:



UCB:

Assuming that all arms are pulled initially at once.

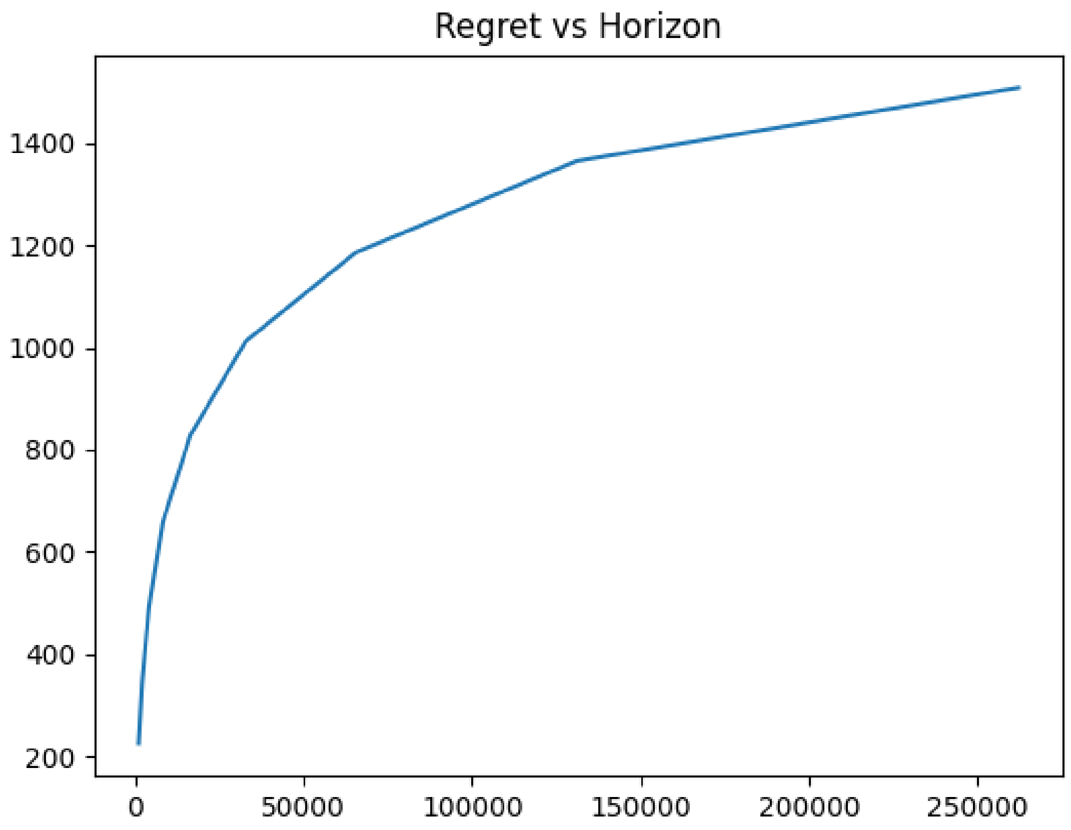
$$ucb_a^t = \hat{p}_a^t + \sqrt{\frac{2*ln(t)}{u_a^t}}$$

Where:

$\hat{p}_a^t$ is the empirical mean for arm a till time t

u_a^t *is the number of times an arm a has been pulled till time t*

Plot2:



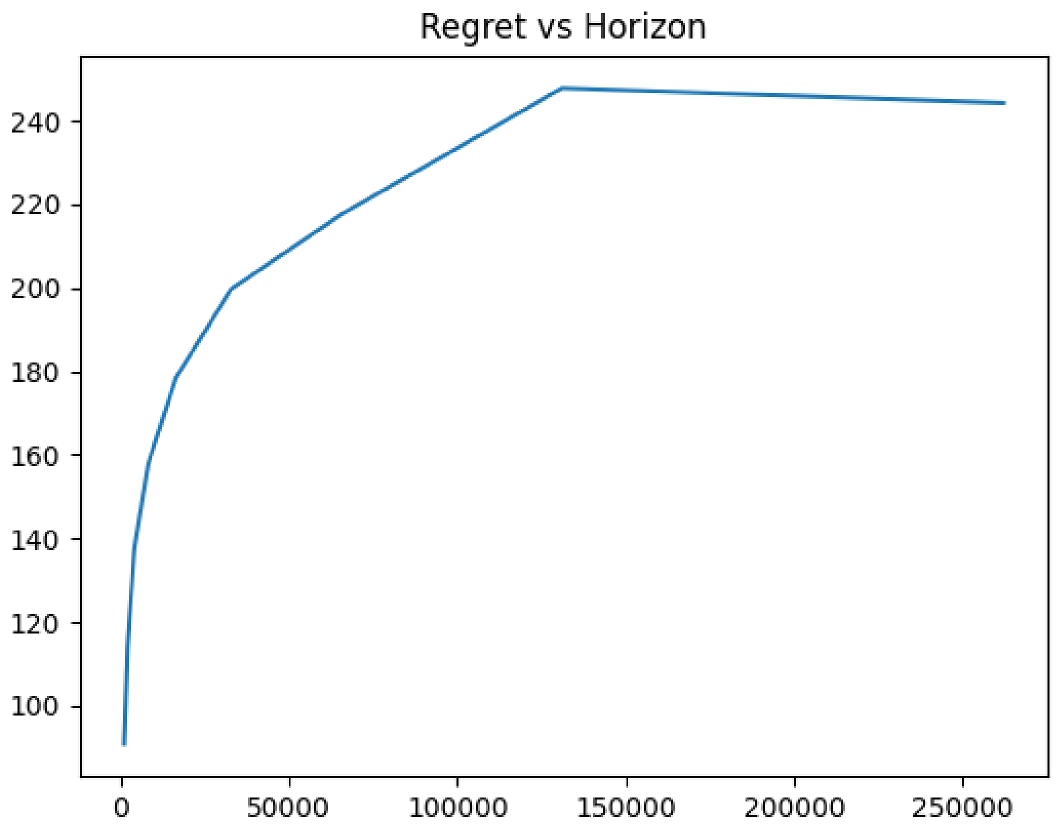
As evident from the plot2, UCB performs better than ε-Greedy(plot1) both for small horizon and also the large.The graph is in similar to logarithmic shape since we know that UCB achieves logarithmic regret. UCB will not work well when horizon or number of arms are less.
Eg. when number of arms are 2 then it will work like round robin.
Also when the horizon is less the arm whose ucb was less earlier might get a chance to be pulled when t=Horizon occurs just before ucb of that arm was maximum.

KL_UCB:

All same as UCB except $ucb\_kl_a^t$ is calculated as follows:

$$ucb\_kl_a^t = \{q \in \left[\hat{p}_a^t,\ 1\right] |\ u_a^t KL\left(\hat{p}_a^t,\ q\right) \leq \ln ln\,(t)\ +\ cln(\ln ln\,(t)\,)\}$$

Plot:



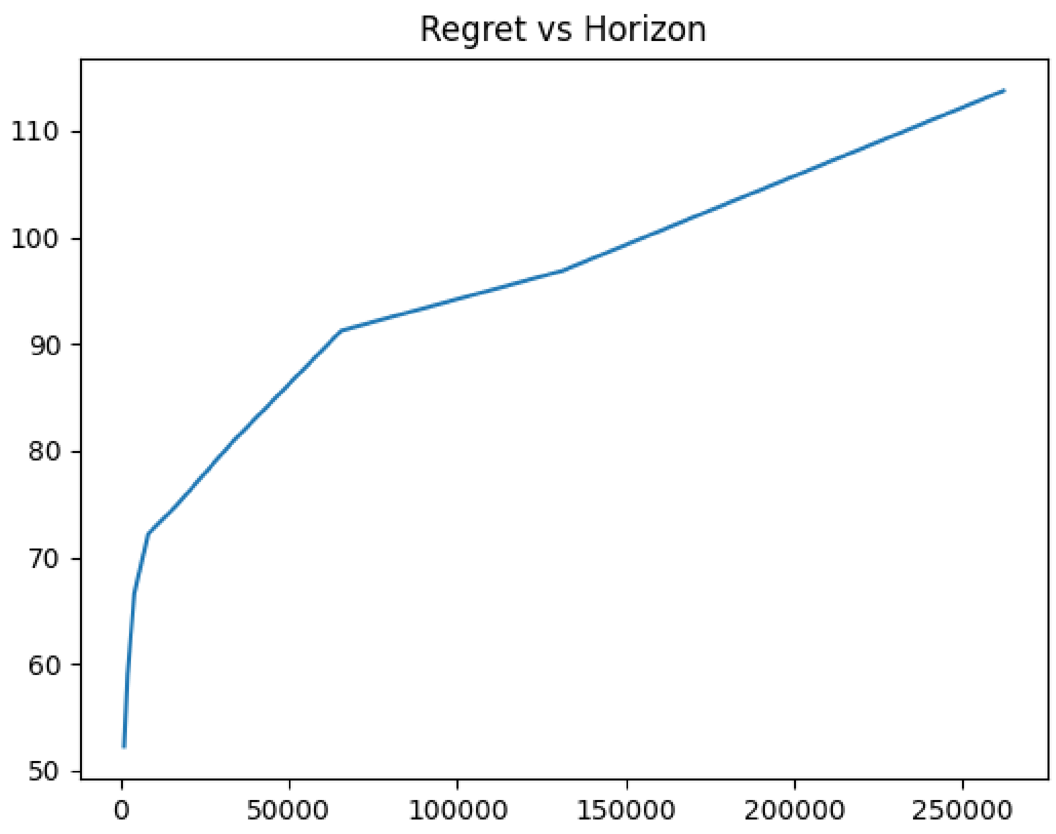
Cleary KL_UCB performs better than ε-Greedy and UCB .
Since we know kl_ucb has tighter confidence bounds than ucb the regret values are almost 1/10 th of that of UCB. Also regret achieves saturation early as compared to UCB.

There was one difficulty I faced which was the calculation q. If not done efficiently the code will take quite some time. Needed to use Binary search for this in order to do it efficiently(Log(N) time).

Thompson Sampling:

Mean of each arm a is assigned a prior belief in terms of successes(s_a^t) and failures (f_a^t) till time t as Beta($s_a^t + 1, f_a^t + 1$) . Then while selecting an arm, sample **X** is drawn from the distribution of each arm and that arm is pulled for which **X** is maximum. s_a^t *and* f_a^t is initially assigned zero values (In the implementation $s_a^t + 1$ *and* $f_a^t + 1$ is maintained).

Plot:



Performs better than ε-Greedy and UCB and also KL_UCB. The regret is almost half that compared to KL_UCB. For most of the instances Thompson sampling performs better that that of other algorithms. It seems Thompson sampling learns faster as compared to the other algorithms, due to which the regret is saturated earlier as compared to the others.

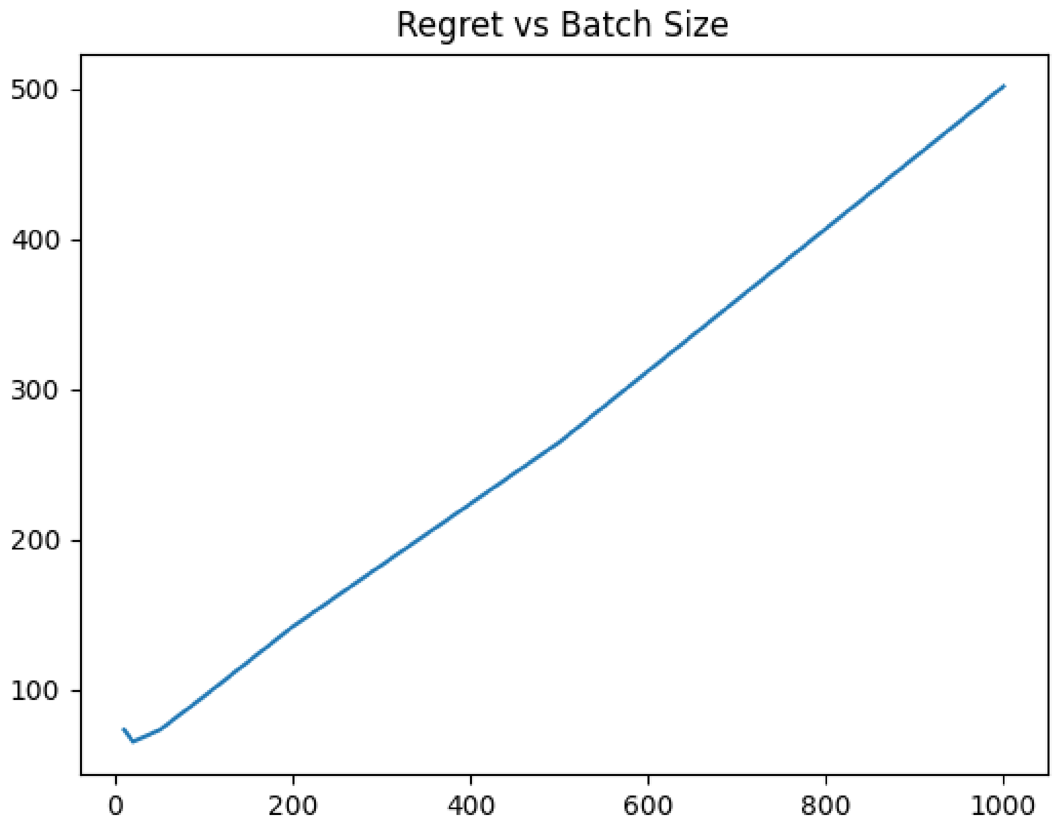
Task2:

Batched Sampling:

Algorithm:

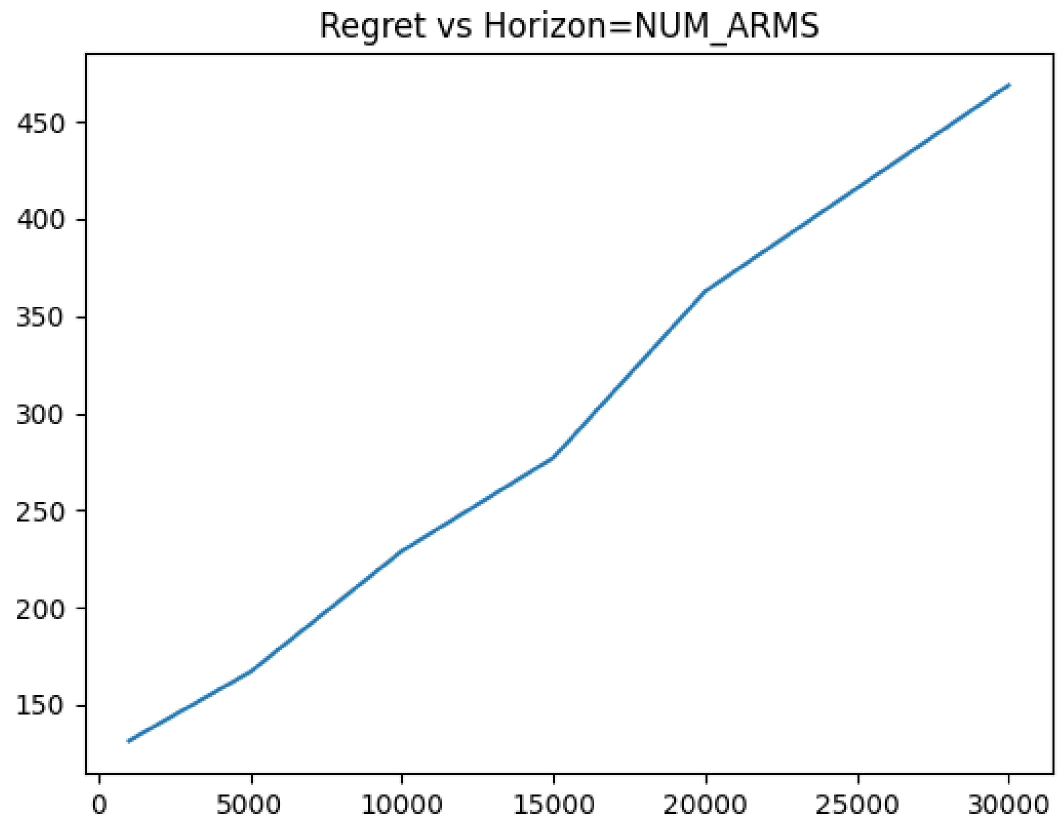
Algorithm works as :

```
    for t from (1,batchsize) do
        Pull an arm with highest value of Beta distribution
        (Beta distribution is same as mentioned in Thompson's sampling for each arm)
    At the end of this for loop we will have N=batchsize number of arm indices which may
    or may not be the same.
```



As we know thompson sampling performs best for most of the instances as compared to the other algorithms. I used Thompson Sampling in order to determine the arms which need to be pulled. Since the algorithm has two for loops for larger batch size it will take some time to give output. For smaller batch size the algorithm will perform almost similar to the thompson's sampling where batch_size = 1. For larger batch size algorithm may not perform well as compared to thompson as in for earlier stages we will have less prior information about the arms in order to use in Beta distribution and we might end up exploring more rather than exploiting better arms. The algorithm performs better than epsilon-Greedy.

TASK 3:



Since we have a very less amount of exploration. We need to explore more carefully or chose only those arms which are performing better.

The algorithm goes as:

```
T= horizon
C = exploration factor
C = t/εT
for t =1 to T
  If np.random.random() > c then:
        Draw an arm using thompson's sampling. –{Explore using Thompsom's Sampling }
  Else
        Draw an arm with highest empirical mean. {Exploit}
Update the data structures accordingly.
```

Clearly from the plot algorithm performs better than ε - Greedy.
Compared to UCB the regret is almost ⅓ rd of UCB.
Compared to thompson and KL_UCB regret is larger (almost twice and thrice resp.).
This may be due to the fact that we have less number of arms in thompson (task1) so it will get more training to learn the best/optimal arms so will have less regret as compared to our case.