PROGRAMMING ASSIGNMENT 1

An assignment report,

submitted to **Prof. Shivaram Kalyanakrishnan**

in the subject of Foundations of Intelligent and Learning Agents

by

Sandarbh Yadav
(22D0374)

INDIAN INSTITUTE OF TECHNOLOGY
BOMBAY

2022

TABLE OF CONTENTS

Task 1

The UCB, KL-UCB and Thomson Sampling algorithms have been implemented and generated plots are attached below.

1.1 UCB Algorithm

At time t, for every arm a, define

$$ucb_a^t = p_a^t + \sqrt{\frac{2\ln(t)}{u_a^t}} \tag{1.1}$$

where p_a^t denotes the empirical mean of rewards from arm a and u_a^t denotes the number of times arm a has been sampled at time t. In UCB algorithm, we pull an arm a for which ucb_a^t is maximum. The submitted code implements above algorithm and the generated plot is attached below. It is clearly seen from the plot that regret has logarithmic dependence on horizon which is optimal.

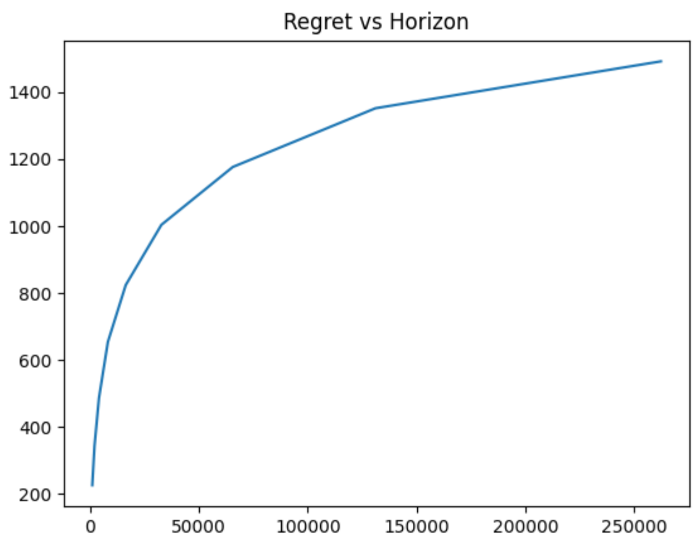

Figure 1.1: Generated plot for UCB algorithm

1.2 KL-UCB Algorithm

KL-UCB algorithm is similar to UCB algorithm, except for a different definition of upper confidence bound.

$$ucb\text{-}kl_a^t = \max\{\ q \in [p_a^t, 1] \ \ s.t.\ \ u_a^t\ KL(p_a^t, q) \leq \ln(t) + c\ln(\ln(t))\ \} \tag{1.2}$$

where $c \leq 3$. Equivalently, $ucb\text{-}kl_a^t$ is the solution $q \in [p_a^t, 1]$ to

$$KL(p_a^t, q) = \frac{\ln(t) + c\ln(\ln(t))}{u_a^t} \tag{1.3}$$

In KL-UCB algorithm, we pull an arm a for which $ucb\text{-}kl_a^t$ is maximum. It is easy to compute the value of $ucb\text{-}kl_a^t$ numerically. In this assignment, binary search is used to calculate value of $ucb\text{-}kl_a^t$.

The submitted code implements above algorithm and the generated plot is attached below. It is clearly seen from the plot that regret has logarithmic dependence on horizon which is optimal. Another observation is that the regret incurred by KL-UCB algorithm is much less compared to UCB algorithm. For UCB algorithm, regret is close to 1400 for horizon of 250000 while for KL-UCB algorithm it is close to 275. This is because KL-UCB algorithm matches the lower bound on regret given by Lai and Robbins. [1]

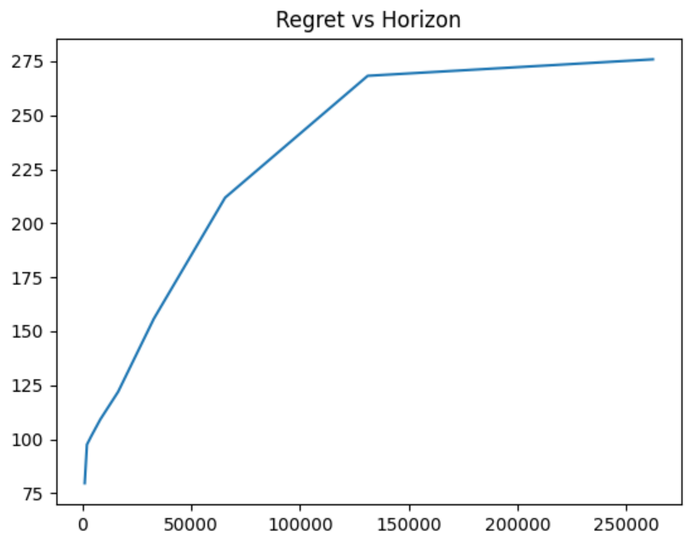

Figure 1.2: Generated plot for KL-UCB algorithm

1.3 Thomson Sampling

At time t, let arm a have s_a^t successes and f_a^t failures. $\beta(s_a^t + 1, f_a^t + 1)$ represents a belief about the true mean of arm a. In Thompson Sampling, for every arm a, we draw a sample from the beta distribution

$$x_a^t \sim \beta(s_a^t + 1, f_a^t + 1) \tag{1.4}$$

and pull the arm a for which x_a^t is maximum.

The submitted code implements above algorithm and the generated plot is attached below. It is clearly seen from the plot that regret has logarithmic dependence on horizon which is optimal. Another observation is that the regret incurred by Thompson Sampling is much less compared to UCB and KL-UCB algorithms. For UCB algorithm, regret is close to 1400 for horizon of 250000 while for KL-UCB algorithm it is close to 275. For Thomson Sampling, this regret is a meagre 110 for horizon of 250000. This is in accordance with Chapelle and Li [2] which states that Thompson Sampling is excellent in practice.

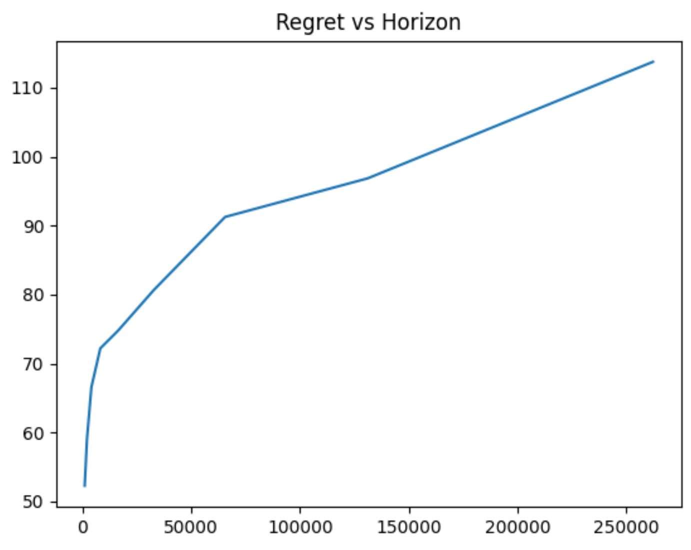

Figure 1.3: Generated plot for Thomson Sampling

Task 2

It was observed in task 1 that Thompson Sampling performed significantly better than UCB and KL-UCB algorithms. So, a modified version of Thompson Sampling is used for this task. The horizon is divided into a number of batches of size *batch_size* which is a factor of horizon. For every batch, Thompson Sampling is performed but instead of pulling the arms instantly we store the arm numbers and their corresponding number of pulls in arrays *index* & *pulls*. At the end of each batch, these arrays are sent to the bandit instance which returns the rewards for the batch according to the number of pulls. After receiving these rewards, the algorithm updates the beta belief distributions and initialises the array *pulls* for next batch. This process is repeated for every batch.

The submitted code implements above approach and the generated plot is attached below. A linear dependence is seen between regret and batch size. Regret increases as the batch size increases. This was expected because for batch size = 1, this algorithm reduces to Thompson Sampling of task 1. As the batch size increases, the delay in updating the beta belief distributions keeps on increasing. Hence, the old belief values of beta belief distributions are used for a longer duration which results in increased regret.

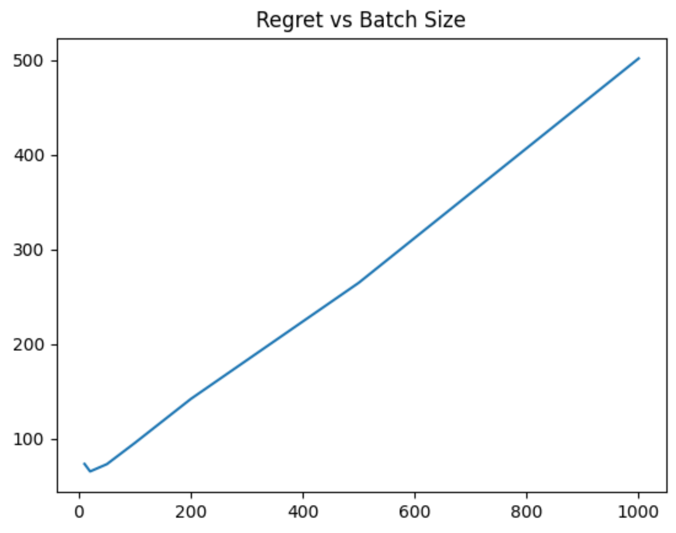

Figure 2.1: Generated plot for Task 2

Task 3

It is given that the arm means are distributed regularly (in arithmetic progression) between 0 and (1 - 1/*numArms*). This fact is exploited for task 3. It is known with certainty that there will always be some arms having mean above a certain threshold value (suppose 0.9). Since the number of arms is equal to the horizon, the aim of this approach is not to find the most optimal arm but to find near optimal arms and exploit them. The algorithm attempts to find one such near optimal arm and keeps on pulling it until its empirical mean drops below the threshold and when that happens it starts looking for another near optimal arm. The process keeps repeating in similar manner.

The submitted code implements above approach and the generated plot is attached below. This plot corresponds to a threshold of 0.95. Clearly, regret increases almost linearly with horizon (= number of arms). This dependence was expected because as the horizon increases, number of arms also increase which increases the complexity of problem reuslting in increased regret. This algorithm performs significantly better than UCB algorithm upto horizon of 30000 despite having to deal with a lot of arms. Moreover, it is observed that a threshold value of around 97-98 yields best results in this setting.

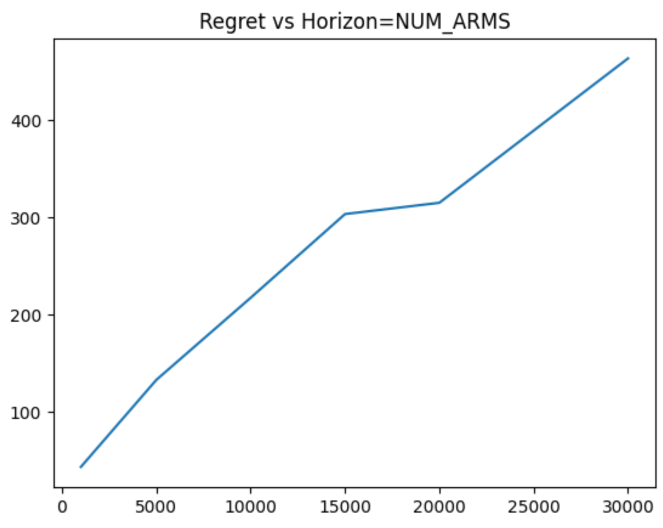

Figure 3.1: Generated plot for Task 3

REFERENCES

[1] T. L. Lai, H. Robbins, *et al.*, “Asymptotically efficient adaptive allocation rules,” *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[2] O. Chapelle and L. Li, “An empirical evaluation of thompson sampling,” *Advances in neural information processing systems*, vol. 24, 2011.