# CS747 Assignment 1

Sarthak Mittal

September 9, 2022

## Contents

# 1 Task 1: Bandit Problem

## 1.1 Upper Confidence Bounds

### 1.1.1 Algorithm

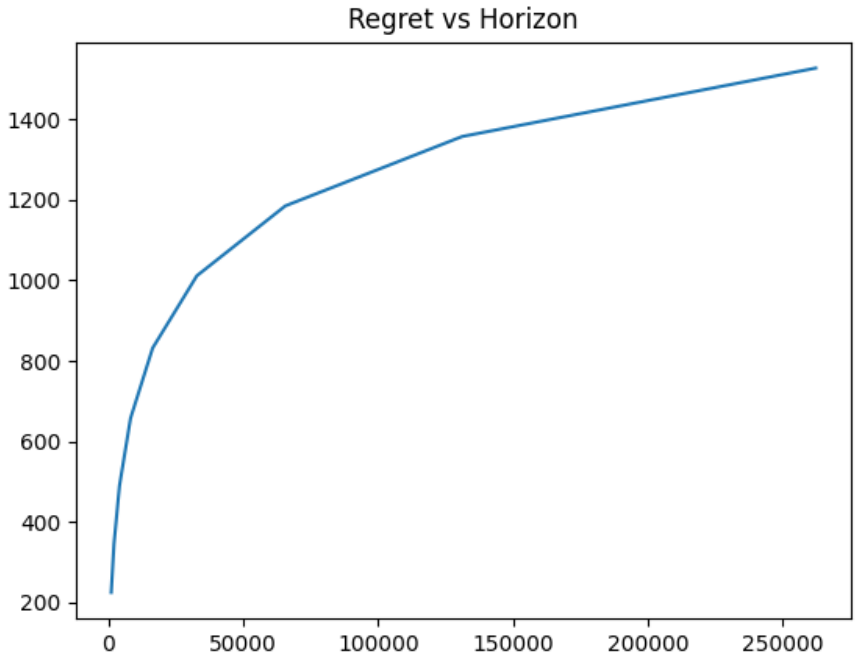
At time $t$, for every arm $a$, if $\hat{p}_a^t$ is the empirical mean of rewards from arm $a$ and $u_a^t$ is the number of times $a$ has been sampled at time $t$, define

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

We pull arm $a$ for which $\text{ucb}_a^t$ is maximum. More information on the UCB Algorithm.

### 1.1.2 Plot

Following is the plot for regret v/s horizon obtained by running `simulator.py` for UCB:




UCB: Regret v/s Horizon

### 1.1.3 Inference

The additional factor apart from empirical mean ensures that all arms are pulled at some time. Also, the more some arm is pulled, the lesser the influence of the second term. This helps in achieving sub-linear regret. It follows from the plot that regret is $\mathcal{O}(log(T))$, as expected from the derivation and analysis of UCB in class, which proved the regret bound by expanding the summations.

### 1.1.4 Comments

We assume that initially each arm has infinite empirical mean, so each arm gets pulled once, after which the algorithm (implemented using functions to calculate empirical mean and uncertainty) leads the pulls. The rewards are accumulated and regret is then calculated. Functions have been explained in comments in code.

## 1.2 Kullback-Leibler Upper Confidence Bounds

### 1.2.1 Algorithm

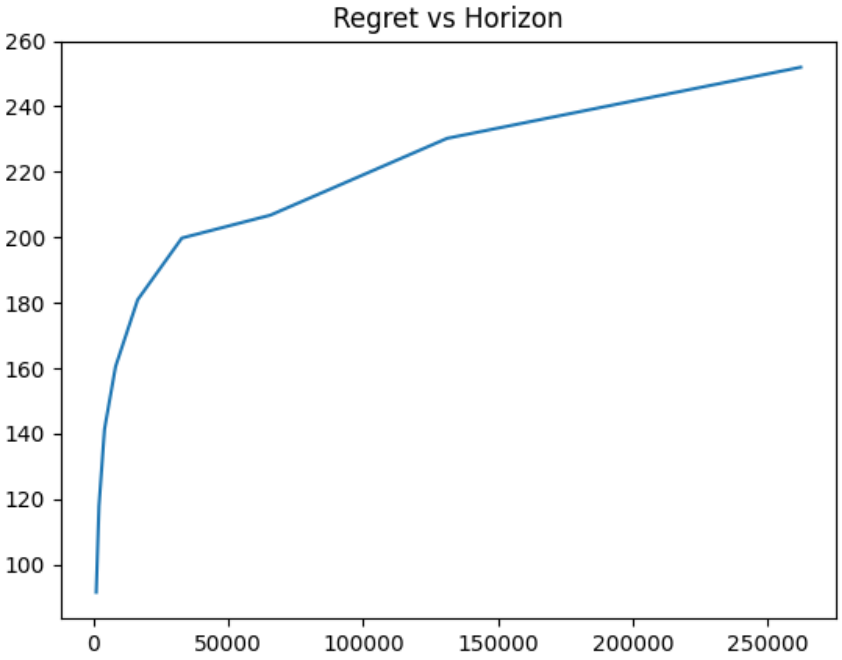
Let $c \geq 3$ be a constant. At time $t$, for every arm $a$, if $\hat{p}_a^t$ is the empirical mean of rewards from arm $a$ and $u_a^t$ is the number of times $a$ has been sampled at time $t$, define

$$\text{ucb-kl}_a^t := \max\left(q \in [\hat{p}_a^t, 1] \text{ s. t. } \text{KL}(\hat{p}_a^t, q) \leq \frac{\ln(t) + c\ln(\ln(t))}{u_a^t}\right)$$

We pull arm $a$ for which $\text{ucb-kl}_a^t$ is maximum. More information on the KL−UCB Algorithm.

### 1.2.2 Plot

Following is the plot for regret v/s horizon obtained by running `simulator.py` for KL−UCB:




KL−UCB: Regret v/s Horizon

### 1.2.3 Inference

The value of $\text{ucb-kl}_a^t$ can be computed using binary search. Since $\text{KL}(p, q)$ is monotonically increasing with $q$, the solution corresponds to the maximum value in the range. The additional sub-sub-linear factor in the expression leads to KL−UCB being a tighter bound, as was explained in class. It follows from the plot that regret is $\mathcal{O}(log(T))$.

### 1.2.4 Comments

We assume that initially each arm has infinite empirical mean, so each arm gets pulled once, after which the algorithm (implemented using functions to calculate empirical mean, KL-divergence and binary search for the value) leads the pulls. The rewards are accumulated and regret is then calculated. Consequently, due to binary search at every time and for every arm, KL−UCB takes much longer to simulate. Functions have been explained in comments in code.

## 1.3 Thompson Sampling

### 1.3.1 Algorithm

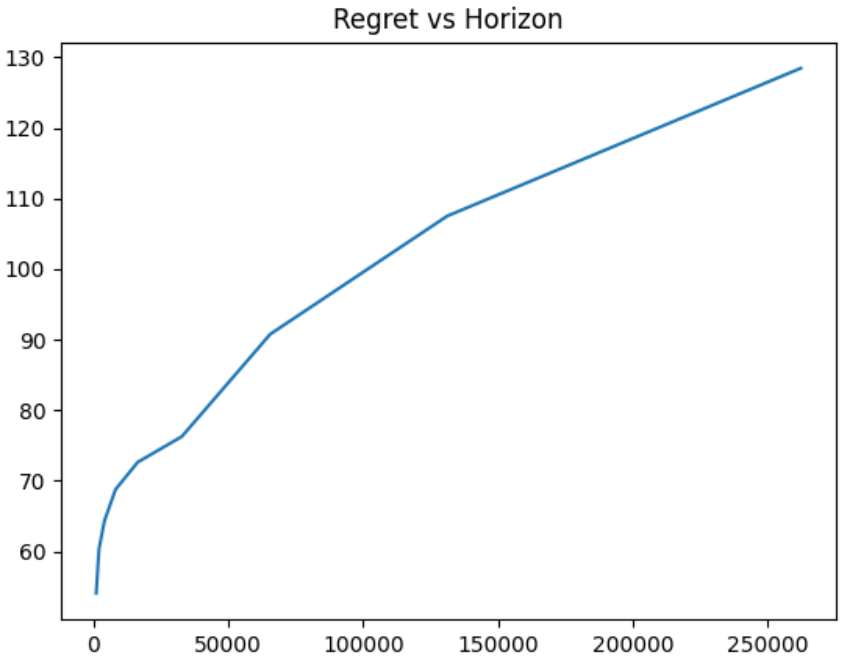
At time $t$, let arm $a$ have $s_a^t$ successes (1's or heads) and $f_a^t$ failures (0's or tails). Then $\text{Beta}(s_a^t + 1, f_a^t + 1)$ represents the belief about true mean of arm $a$. We draw a sample $x_a^t \sim \text{Beta}(s_a^t + 1, f_a^t + 1)$, where

$$\text{Beta}(\alpha, \beta) := \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)}$$

We pull arm $a$ for which $x_a^t$ is maximum. More information on Thompson Sampling and Beta Distribution.

### 1.3.2 Plot

Following is the plot for regret v/s horizon obtained by running `simulator.py` for Thompson Sampling:




Thompson Sampling: Regret v/s Horizon

### 1.3.3 Inference

The regret achieved is sub-linear, and it is very easy to implement. The algorithm is excellent in practice. The random sampling using Beta distribution accounts for the empirical mean and associated uncertainty with it. We add the constant so that the coefficients are positive. The bound for the given dataset has turned out to be tighter than both the UCB bounds. It follows from the plot that regret is $\mathcal{O}(log(T))$.

### 1.3.4 Comments

At each turn, we sample an instance from the Beta distribution depending on the number of positive/negative rewards, and the maximum out of them leads the pulls. The rewards are accumulated and regret is then calculated. Functions have been explained in comments in code.

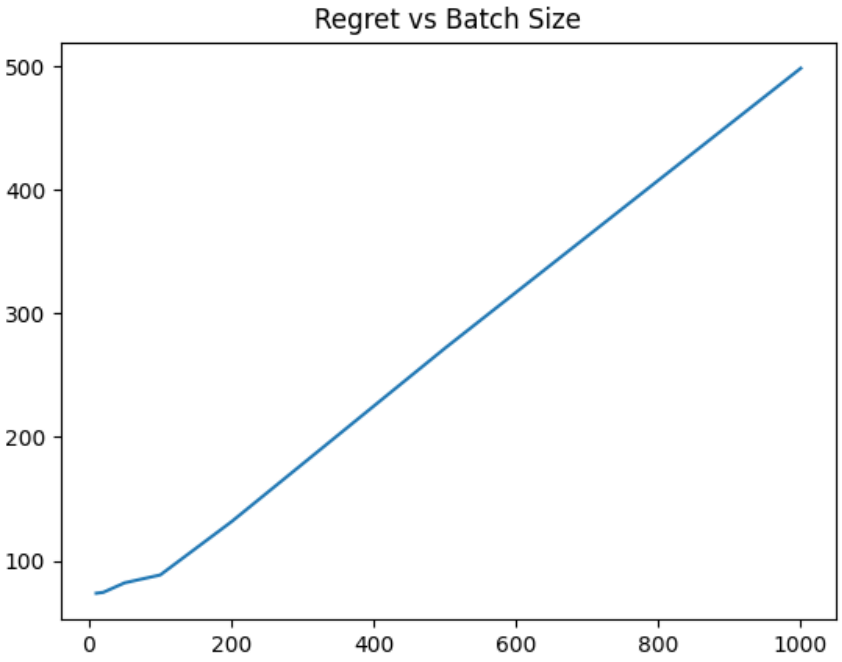
# 2 Task 2: Batched Bandits

## 2.1 Algorithm

We choose optimal arm based on Thompson Sampling for each iteration of the batch size $B$. We record the optimal arms (and add to their number of pulls) into a dictionary.

## 2.2 Plot

Following is the plot for regret v/s horizon obtained by running `simulator.py` for batched algorithm:




Batched Sampling: Regret v/s Batch Size

## 2.3 Inference

The regret is linearly dependent on the batch size. The larger the batch size, more the number of times we need to make decision for optimal arm without actually upating the rewards and pulls. This results in a $\mathcal{O}(\mathcal{B})$ number of sub-optimal pulls being made, thus introducing a linear factor into the regret. It follows from the plot that regret is $\mathcal{O}(B)$.

## 2.4 Comments

We draw samples for each arm from the Beta distribution as many times as the batch size, and record the optimal choices. Then we collectively update the pulls and rewards. The rewards are accumulated and regret is then calculated. Functions have been explained in comments in code.

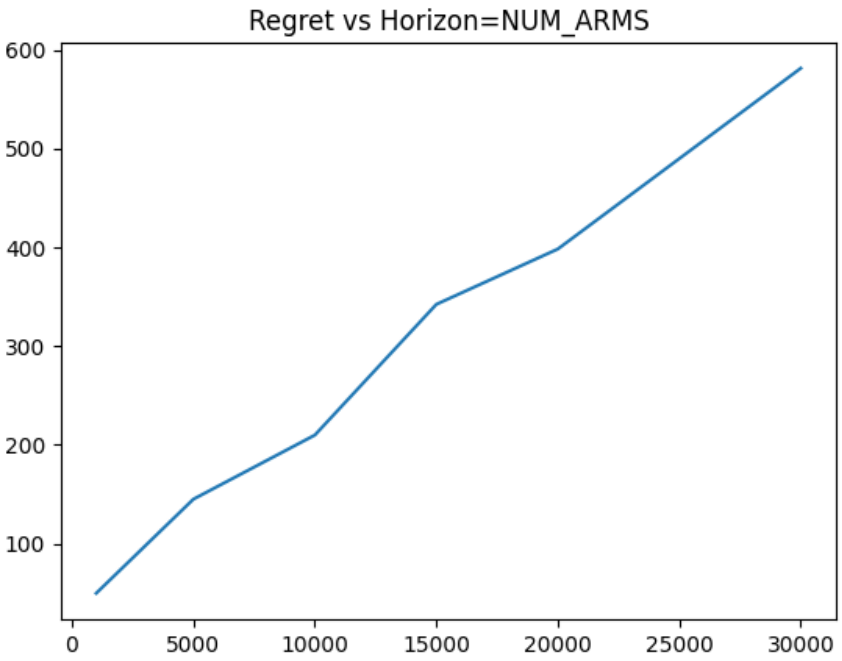
# 3 Task 3: Multi-Armed Bandits

## 3.1 Algorithm

We decide a threshold factor $\epsilon$ which essentially accounts for the explore-exploit ratio. If the mean of the current arm is within the threshold amount of the real optimal arm, we continue to exploit, otherwise we explore and choose another arm randomly. More information on Explore-Exploit with Threshold.

## 3.2 Plot

Following is the plot for regret v/s horizon obtained by running `simulator.py` for multi-armed algorithm:




Multi-Armed Bandits: Regret v/s Horizon

## 3.3 Inference

The regret turns out as being linearly proportional to the horizon. Due to the number of arms being equal to the horizon, the search for the optimal arm using empirical means causes the overall regret to become linearly dependent on the horizon value. The threshold ensures that we do not stay on a sub-optimal arm for very long.

## 3.4 Comments

We check if the empirical mean of the current arm satisfies the threshold. If it does then our choice stays and we continue to exploit. Otherwise, we randomly choose another arm and proceed to find the reward and empirical mean in the next pull. The rewards are accumulated and regret is then calculated. Functions have been explained in comments in code.