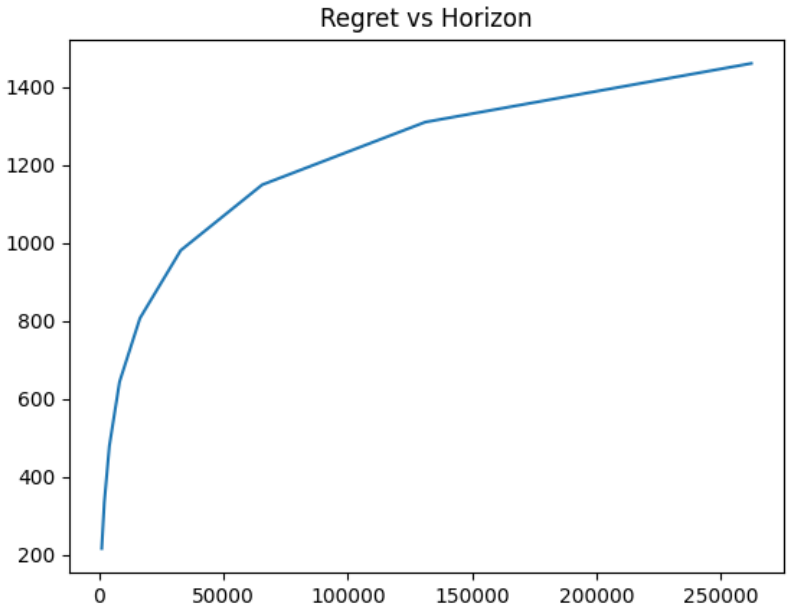
CS 747 Assignment 1
Multi-armed Bandits

Amruta Mahendra Parulekar, 20d070009

September 10, 2023

1 Task 1

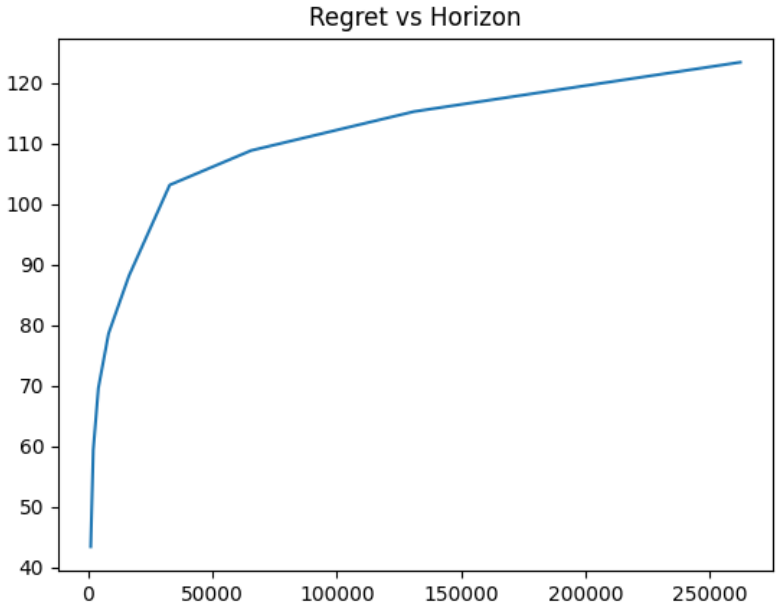
1.1 UCB



- The variables initialised were: an array to store the UCB for each arm (initially zero), an array to store the empirical mean for each arm(initially zero), an array to store the pull counts (initially zero) and a variable to store timestep.
- First every arm is pulled once. After these cases are over, the arm with max UCB is pulled. After this, for every timestep, the the arm with maximum ucb is chosen.
- Once reward is obtained, the pull count for the arm pulled is updated and its empirical mean is updated.
- Then, the timestep is updated and if each arm hasnt been pulled once, only the UCB for that arm is updated. Else,for every arm, the UCB is calculated using the formula:

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

1.2 KL UCB



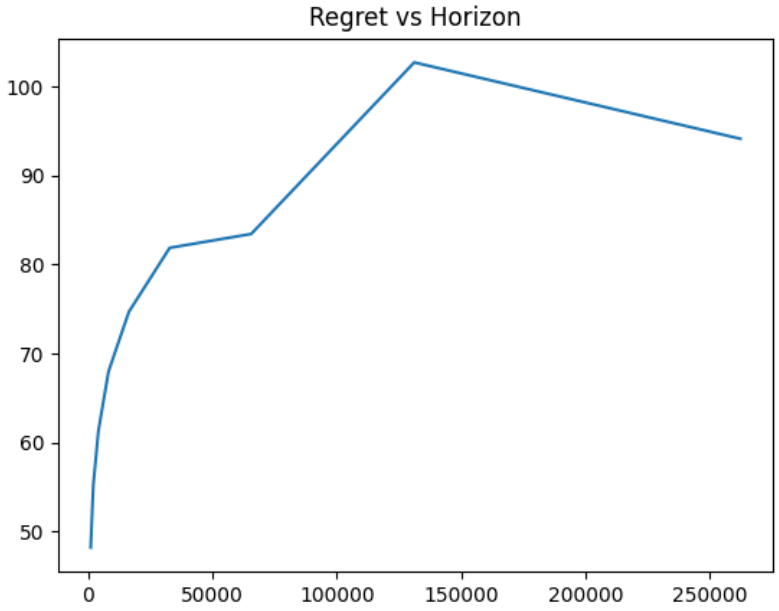
- The variables initialised were: an array to store the UCB for each arm (initially zero), an array to store the empirical mean for each arm(initially very small value close to zero as it is used inside log so the limit value was taken), an array to store the pull counts (initially taken as 1 because it is in the denominator in ucb equation, which wont matter where horizon is large) and a variable to store timestep.
- First every arm is pulled once. After these cases are over, the arm with max UCB is pulled. After this, for every timestep, the the arm with maximum ucb is chosen.
- Once reward is obtained, the pull count for the arm pulled is updated and its empirical mean is updated.
- Then, the timestep is updated and if each arm hasnt been pulled once, only the UCB for that arm is updated.Else, for every arm, the UCB is calculated using the formula:

$$\text{for } x, y \in [0,1),\ KL(x,y) \stackrel{\text{def}}{=} x \ln \frac{x}{y} + (1-x) \ln \frac{1-x}{1-y}.$$

$$\text{ucb-kl}_a^t = \max\{q \in [\hat{p}_a^t, 1] \text{ such that } u_a^t KL(\hat{p}_a^t, q) \leq \ln(t) + c \ln(\ln(t))\}$$

- Care is taken that when mean is 1, q is taken as 1 as q (ucb) cant be more than that
- C=0 and when rhs is taken to lhs, we get a monotonically increasing function.
- We want to maximise q between p and 1 so we can take it as an equality and just find the value of q where RHS becomes zero.
- To find q, I used the binary search algorithm with upper and lower limits 1 and p_a respectively.
- The algorithm divides the interval in half and checks the value of the function at the midpoint. If more than zero, it searches in the left half and if less than zero, it searches in the right half. It stops when the low and high interval become so close that they are almost one number.

1.3 Thompson Sampling

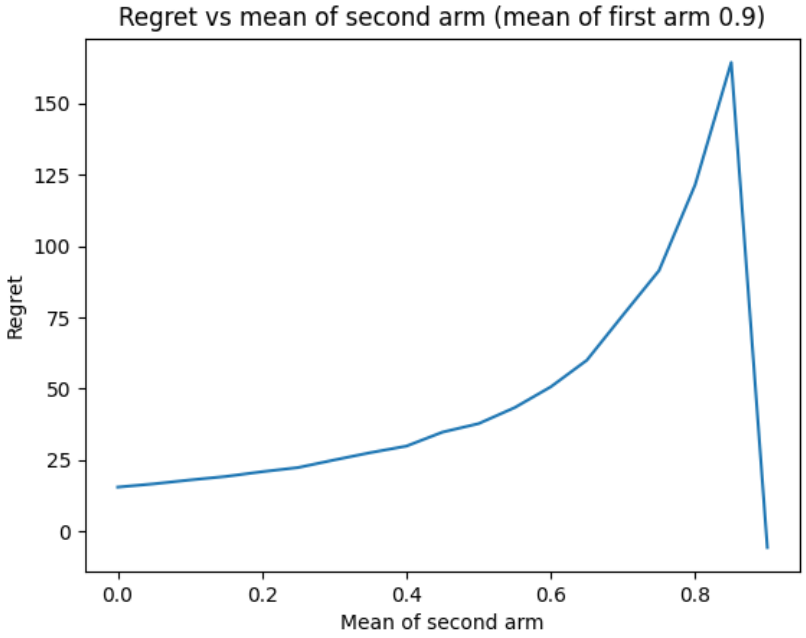


- The variables initialised were: an array to store the beta for each arm (initially zero), an array to store the successes of the arms(initially zero) and one to store the failures of the arms(initially zero) and a variable to store timestep.
- First arm is chosen randomly. After this, for every timestep, the the arm with maximum beta is chosen.
- Once reward is obtained, the success value or failure value of that arm is updated.
- Then, the timestep is updated and for every arm, the a random sample from the beta distribution is calculated using the formula:

$$x_a^t \sim Beta(s_a^t + 1, f_a^t + 1).$$

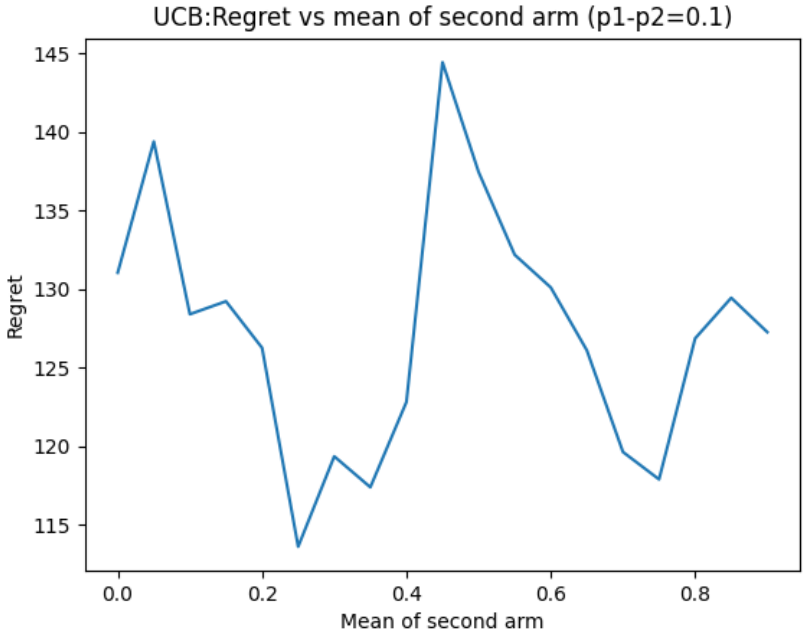
2 Task 2

2.1 Part a

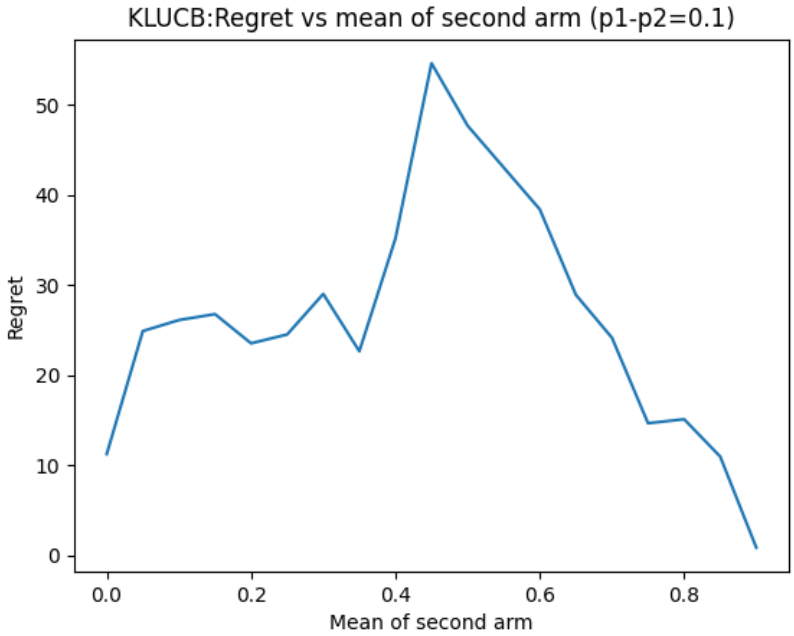


- When the difference in empirical means of the two arms is high, it is easier for the algorithm to identify which arm is better, i.e. learning is quicker. Thus, regret is lower.
- As the difference between the empirical means of the arms decreases, it becomes more difficult to identify the better arm so regret increases.
- When the means of the two arms are equal however, it will not matter which arm you pick. Thus regret becomes zero.
- Thus, the graph has a steady increase as p2-p1 decreases and then a nosedive to zero when p1=p2

2.2 Part b



- UCB is a more loosely bound algorithm than KL UCB. It follows the Lai-Robbins lower bound, which has a central peak too.
- Since the difference between the empirical means of the two arms is the same in all cases and it is small, it is almost equally hard for the algorithm to learn which arm is better.
- The regret is low when both arms have a very high mean as whichever arm you choose, it will most likely give a good outcome. i.e. risk to reward is minimised.
- The regret is the higher when both arms have a very low mean as whichever arm you choose, it will most likely give a bad outcome. i.e. risk to reward is higher.(risk same reward lower)
- However, when the means are close to 0.5 for both arms, it becomes equivalent to a coin flip, i.e a random outcome for both arms, and hence regret is the highest.
- Other than that it is mostly randomized, due to the loosely bounded nature of the algorithm



- KL UCB is a more tightly bound algorithm than UCB. It follows the Lai-Robbins lower bound, which has a central peak too.
- Since the difference between the empirical means of the two arms is the same in all cases and it is small, it is almost equally hard for the algorithm to learn which arm is better.
- However, when the means are close to 0.5 for both arms, it becomes equivalent to a coin flip, i.e a random outcome for both arms, and hence regret is the highest.
- The regret is the lowest when both arms have a very high mean as whichever arm you choose, it will most likely give a good outcome. i.e. risk to reward is minimised.
- The regret of KL UCB is much lower than that of UCB.

3 Task 3

- We know that the bandit arms give faulty predictions with a fixedf probability, and otherwise give accurate predictions based on how good the arm is.
- We cannot identify which output is faulty and which is accurate, but if we use all data points for learning we will be learning a lot of faulty data points.
- If we use too few data points for learning, the learning will be slow, so we want an optimal number of data points to ignore and ones to learn from.
- I hypothesized that if with the same probability as the fault probability, we decide to ignore the outputs of the arms, we will maximise the points where we are learning from a non faulty output.
- I chose to use Thompson sampling as my base algorithm as it had given the best results in task 1 and it is also a fast algorithm.
- Then I used a function that returns 'True' with the probability,

$$P_{True} = 1 - faultprob \tag{1}$$

  and otherwise returns false.
- If false, the reward will be ignored and it wont be learnt from.
- If true, the reward and the arm which gave the reward will be considered in the calculation of success and failure and a new set of beta values for all arms will be calculated.
- The next arm pull index is the one which has the highest beta.

4 Task 4

- We know that there are two bandit instances and how they are chosen is completely random.
- I hypothesized that the best or preferred arm number to choose, would be where both arms have a good empirical mean or when one arm is so good that the other arm failing wouldn't matter.
- This means that the sum of the individual empirical means of the two bandit instances for that arm index would be maximum.
- This is because we want to maximise the arm results over both bandit instances, only then can we say that even if any one instance is chosen randomly, it will give the best outcome.
- Basically,

$$Maximise(p_a\beta_a + p_b\beta_b) \tag{2}$$

  to get the arm index to pull. Here, since choice is uniform between the two instances, pa=pb
- I chose to use Thompson sampling as my base algorithm as it had given the best results in task 1 and it is also a fast algorithm.
- I calculated the beta for both the bandit instances separately, by giving an if condition, such that when a specific instance was used, the success and failure arrays for that instance were updated.
- The net beta for an arm index was taken as the sum of the individual betas of both bandit instances for that index.
- The next arm pull index was the one which had the highest net beta.

5 Simulations

All simulations were run and then the autograder was run. All test-cases were passed.

```
amruta@amruta-OMEN-Laptop-15-ek0xxx:~$ cd code
amruta@amruta-OMEN-Laptop-15-ek0xxx:~/code$ source env747/bin/activate
(env747) amruta@amruta-OMEN-Laptop-15-ek0xxx:~/code$ python3 autograder.py --task all
================== Task 1 ==================
Testcase 1
UCB               : PASSED. Regret: 42.28
KL-UCB            : PASSED. Regret: 12.90
Thompson Sampling : PASSED. Regret: 7.00

Testcase 2
UCB               : PASSED. Regret: 297.20
KL-UCB            : PASSED. Regret: 62.70
Thompson Sampling : PASSED. Regret: 52.14

Testcase 3
UCB               : PASSED. Regret: 408.34
KL-UCB            : PASSED. Regret: 65.45
Thompson Sampling : PASSED. Regret: 70.38

================== Task 3 ==================
Testcase 1
Faulty Bandit Algorithm: PASSED. Reward: 8627.98

Testcase 2
Faulty Bandit Algorithm: PASSED. Reward: 1909.04

Testcase 3
Faulty Bandit Algorithm: PASSED. Reward: 3207.18

================== Task 4 ==================
Testcase 1
MultiBandit Algorithm: PASSED. Reward: 6475.08

Testcase 2
MultiBandit Algorithm: PASSED. Reward: 1494.42

Testcase 3
MultiBandit Algorithm: PASSED. Reward: 10997.46

Time elapsed: 78.51 seconds
(env747) amruta@amruta-OMEN-Laptop-15-ek0xxx:~/code$
```