CS747: Assignment 1

Yashvardhan

September 2022

1 Task1

1.1 UCB

- I used 3 state variables to determine next arm to pull
- UCB was calculated using vec_UCB function using the formula
- Arm with maximun UCB was pulled

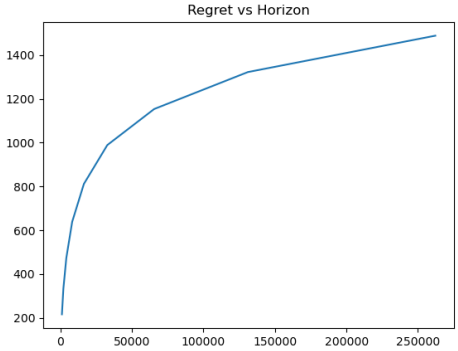

Figure 1: Task3 get$_r eward()$

1.2 KL$_U CB$

- I used 3 state variables to determine next arm to pull
- For determining optimal $q*$ in the KL-divergence I iterated over q till q converged to a value
- Arm with maximum optimal $q*$ is then pulled

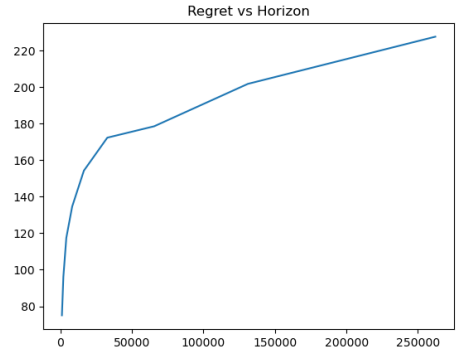

Figure 2: Task3 get$_r eward()$

1.3 Thompson

- I used **np.random.beta()** function to get random value of optimistic-mean from each arm given there previous successes and failure
- Arm with maximun optimistic mean was pulled

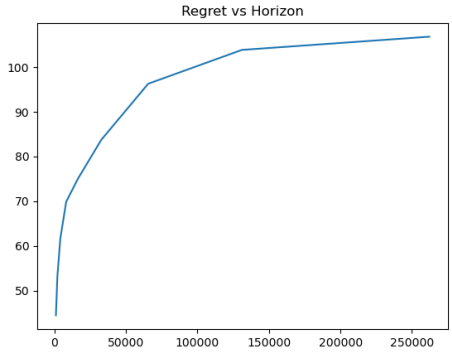

Figure 3: Task3 get$_reward()$

Inference: I got best performance in Thompson sampling followed by KL-UCB and UCB. KL-UCB performs better than UCB as it achieves lower bound on defiened in Lai and Robbins lower bound. Thompson Sampling also achieves this bound but seems to perform better in practice

2 Task2

- I used *Thompson Sampling* for this task
- I iterated over batch size such that every iteration returned the arm to be sampled
- I stored and returned this stored computation

```
def give_pull(self):
    # START EDITING HERE
    mp = {}
    for i in range(self.batch_size):
        tomp = vec_tomp(self.succ, self.tot)
        m = np.argmax(tomp)
        if str(m) in mp.keys():
            mp[str(m)] += 1
        else:
            mp[str(m)] = 1

    keys = []
    values = []
    for k in mp.keys():
        self.tot[int(k)] += mp[k]
        keys += [int(k)]
        values += [mp[k]]

    # return [0], [self.batch_size]
    return np.array(keys), np.array(values)
```

Figure 4: task2 get$_pulls()$

Inference: As the batch size increase, the regret increases linearly. This can be justified by saying that we are not getting immediate feedback from our system for tuning the parameters of our algorithm. Which results in increasing regret.

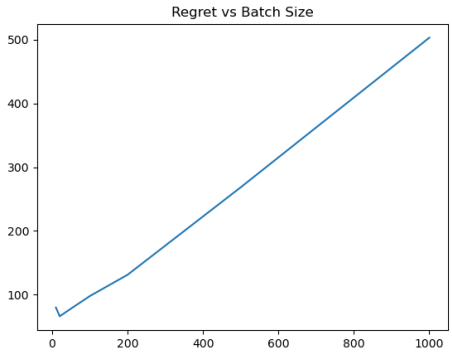

Figure 5: Task3 get$_r eward()$

3 Task3

- Here I basically sampled arms at random till there empirical mean dropped below a certain threshold
- If the e-mean dropped below, I randomly chose another armed to sample
- Here threshold was determined using trial and error
- This was done so that I could settle down at some arm with empirical mean greater the threshold.
- From that point onward I start exploit till empirical mean drops below threshold

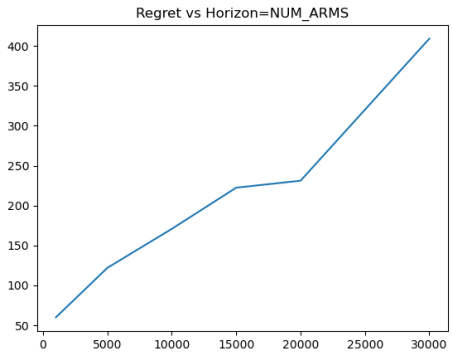

Figure 6: Task3 get$_r eward()$

Inference: Regret here also increase linearly with horizon.

```
def get_reward(self, arm_index, reward):
    # START EDITING HERE
    if reward == 0:
        self.fail[arm_index] += 1
    else:
        self.succ[arm_index] += 1

    if self.succ[arm_index]/(self.succ[arm_index]+ self.fail[arm_index]) < 0.97 :
        self.arm_to_sample = np.random.randint(self.num_arms)
    # raise NotImplementedError
    # END EDITING HERE
```

Figure 7: Task3 get$_r eward()$