CS747 Programming Assignment 1 Report

Tezan Sahu — 170100035

September 24, 2020

---

1 Task 1

In the code, we define a class `Bandit`, that gets initialized with the true mean arm rewards & has a `pull_arm()` method that simulates the pulling of an arm to generate & return a reward based on the true mean of that arm.

For each learning agent, we define a separate class, each having:

- `__init__(...)` method, to initialize the state of the agent
- `act(...)` method, to choose an action (select an arm to be pulled) based on the strategy
- `feedback(...)` method, to receive feedback from the bandit instance after pulling an arm & update the state of the agent

The `Experiment` class initializes the bandit instance along with desired agent, and uses the `run_bandit()` method to actually simulate the agent pulling an arm & receiving feedback along a defined horizon.

Here, we discuss the approaches used to implement the different strategies for the agent.

1.1 Epsilon-Greedy

The `epsilon-greedy` algorithm has been implemented in the following manner:

- Based on the `epsilon` value (ϵ in the range [0, 1]) provided as an argument, the arm with highest empirical mean is chosen with probability $1 - \epsilon$ (*exploitation*), while with ϵ probability, a random arm is chosen (*exploration*)
- $\epsilon = 1$ indicates pure exploration, while $\epsilon = 0$ indicates pure exploitation
- Initially, when an arm has not been pulled even once, the initial value of empirical mean for that arm is set to a slightly optimistic value (0.5 in our implementation)
- Ties are resolved automatically using `numpy.argmax()` function, which chooses the index of first occurring maximum value in an array in case there are 2 or more maximum values present. *This is also done in the remaining algorithms.*

1.2 UCB

The `ucb` algorithm has been implemented as follows:

- For a k-arm bandit, each arm is pulled once in a *round robin* fashion for the first k rounds
- For further rounds, we first compute the *upper confidence bound (UCB)*, which is the sum of the empirical mean & *exploration bonus* for each arm:

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

- Arm with the highest UCB is chosen at every instance to be pulled

1.3 KL-UCB

The `kl-ucb` algorithm has been implemented as follows:

- For a k-arm bandit, each arm is pulled once in a *round robin* fashion for the first k rounds
- Here, the upper confidence bound is defined in a slightly different manner such that $ucb - kl_a^t$ is given by the maximum value of $q \in [\hat{p}_a^t, 1]$ such that:

$$u_a^t KL(\hat{p}_a^t, q) \leq ln(t) + c \cdot ln(ln(t))$$

- Since KL-Divergence is a monotonically increasing function, for implementation, we used the *bisection method* as a numerical method to solve the above equation, with a tolerance of 0.0001. Also, to speed up the execution, early stopping was implemented so as to terminate the search in case the tolerance value wasn't reached within 5 iterations.
- We chose the value of $c = 3$ because the paper by Garivier and Capp specify that c shoule be greater than or equal to 3 for the proof of KL-UCB algorithm, and also it showed the best performance among some values that I tried out.
- Similar to the UCB algorithm, we pull the arm with the highest UCB.

1.4 Thompson Sampling

The `thompson-sampling` algorithm has been implemented as follows:

- Number of successes & failures for each arm are initialized to 1
- After each pull, we note the success & failures for the arms, & create a *beta distribution* for each arm with the parameters $\alpha = successes + 1$ & $\beta = failures + 1$.
- Now, we sample from each of these beta distributions & pull the arm whose sampled value is the highest.

2 Task 2

2.1 Thompson Sampling with Hint

In this problem, we are given a hint about the true means of the arms in the form of a permutation of the true means by sorting them & using them in the agent.

The `thompson-sampling-with-hint` algorithm is implemented as follows:

- Here, instead of modelling the true means using a beta distribution (that could take any value in the range $[0, 1]$, we try to come up with a discrete distribution over the n possible values of the arm means.
- Let n be the number of arms & p_0, p_1, ..., p_{n-1} be the true arm means in ascending order. We assume that every arm i has a certain probability q_{ij} that the true mean of that arm is p_j.
- We create a matrix $Q_{n \times n}$ of probabilities q_{ij} defined above & initialize it with values $\frac{1}{n}$. Each row of the matrix corresponds to a particular arm i, while each column corresponds to the probability that a particular arm has the mean p_j according to the hint. *[implemented in `__init__(...)`]*
- If we choose an arm l, which yield a reward r, the posteriors for that arm is given by (using Bayes Rule): *[implemented in `feedback(...)`]*

$$\forall k \in \{0, ..., n-1\},\ P(a_l = p_k | r_l = r) = \frac{P(r_l = r | a_l = p_k) \cdot P(a_l = p_k)}{\sum_{k=0}^{n-1} P(r_l = r | a_l = p_k) \cdot P(a_l = p_k)}$$

- If $r = 1$, $P(r_l = r | a_l = p_k) = p_l$
- If $r = 0$, $P(r_l = r | a_l = p_k) = 1 - p_l$
- The choice of arm to be pulled is made by selecting the arm with the largest value of $q_{i(n-1)}$, i.e., the arm which has the maximum probability of being the arm with the largest true mean p_{n-1}. *[implemented in `act(...)`]*

3 Task 3

For a given horizon, it is indeed possible to find ϵ_1, ϵ_2 & ϵ_3 such that $\epsilon_1 < \epsilon_2 < \epsilon_3$ & the regret of ϵ_2 is less compared to the other two. This is because for a large value of ϵ, more exploration would be done, leading to potentially larger regrets. As the value of ϵ is reduced, we go on exploiting more, thus reducing the regret. Now, for a given horizon, beyond a value of ϵ, reducing it further would lead to insufficient exploration & chances of the agent getting hooked on to a sub-optimal arm for a large period of exploitation, thus increasing the regret.

However, if the horizon is sufficiently increased (or taken up to ∞), the regret for this ϵ would eventually become the minimum out of all the 3 cases because in the limit, this will exploit more compared to the other 2.

Following are the values for ϵ_1, ϵ_2 & ϵ_3 for each of the instances when the horizon was 10000. The regret values are averaged over 50 unique seed values.

- **Instance 1:**

	$\epsilon_1 = 0.005$	$\epsilon_2 = 0.01$	$\epsilon_3 = 0.05$
Regret	58.60	47.34	103.32

- **Instance 2:**

	$\epsilon_1 = 0.01$	$\epsilon_2 = 0.05$	$\epsilon_3 = 0.1$
Regret	331.98	146.54	232.12

- **Instance 3:**

	$\epsilon_1 = 0.01$	$\epsilon_2 = 0.05$	$\epsilon_3 = 0.1$
Regret	443.19	338.75	492.39

4 Task 4

4.1 For Task 1

Following are the plots showing the regret for each of the 3 bandit instances with the horizon. Each point here gives the average regret from the 50 random runs at the particular horizon for the respective algorithm. Some observations that could be made from this exercise & the plots are mentioned below:

- Clearly, for all the 3 instances, the $\epsilon-$greedy algorithm does NOT give a sublinear regret (consistent with the theory mentioned in class).
- The KL-UCB algorithm performs better than the UCB algorithm (due to its tighter theoretical upper confidence bound, as mentioned in theory) in most cases (except some of the initial initial horizons for Instance 1.
- Thompson sampling performs the significantly better in all 3 cases, ensuring that the agent incurs very low regrets even at very large horizons
- For all the 3 instance, it may appear that initially the $\epsilon-$greedy algorithm may have lower regrets, but in the long run, UCB, KL-UCB & Thompson sampling manage to get sublinear regret values. Though this is not very evident for Instance 3 (25-armed bandit), especially for UCB, it could be examined that on extrapolating the plots beyond the horizon of 102400, it is quite evident that the regret for UCB will become lesser than that for $\epsilon-$greedy algorithm.
- As the number of arms increased, the overall regret at a given horizon increased (due to larger number of options to slect an arm from) & also, the algorithms took longer to explore & ascertain the arm with the highest true mean.
- The average execution times for the algorithms was also somewhat different, & became very prominent while running for a horizon of 102400. This is qualitative shown as follows:

$$\epsilon - greedy < ThompsonSampling < UCB << KL - UCB$$

4.2 For Task 2

- Clearly, the knowledge of the true means as a hint has immensely helped in minimizing the regret in all the 3 cases.
- It is interesting to note that for instances 2 & 3, after around 2000 timesteps, the cumulative regret values have actually started to show a decreasing trend. This could be because the hint has enable the algorithm to actually narrow down on the optimal arm in such a way that it consistently pulls that arm, without actually *exploring* other arms after a sufficiently long period of time.

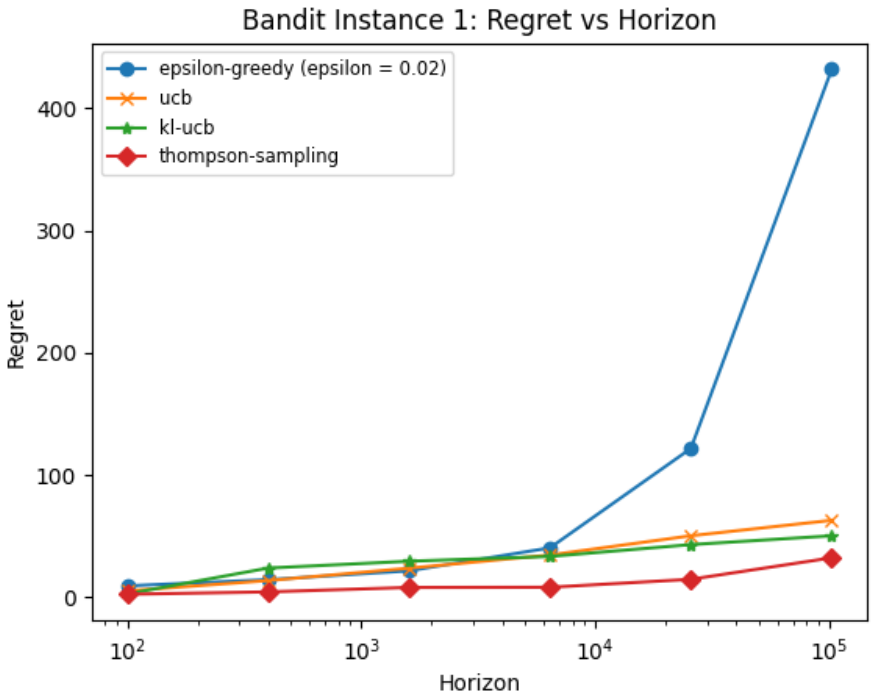

Figure 1: Regret vs Horizon Plot for Instance 1 (2-armed bandit) using all 4 algorithms

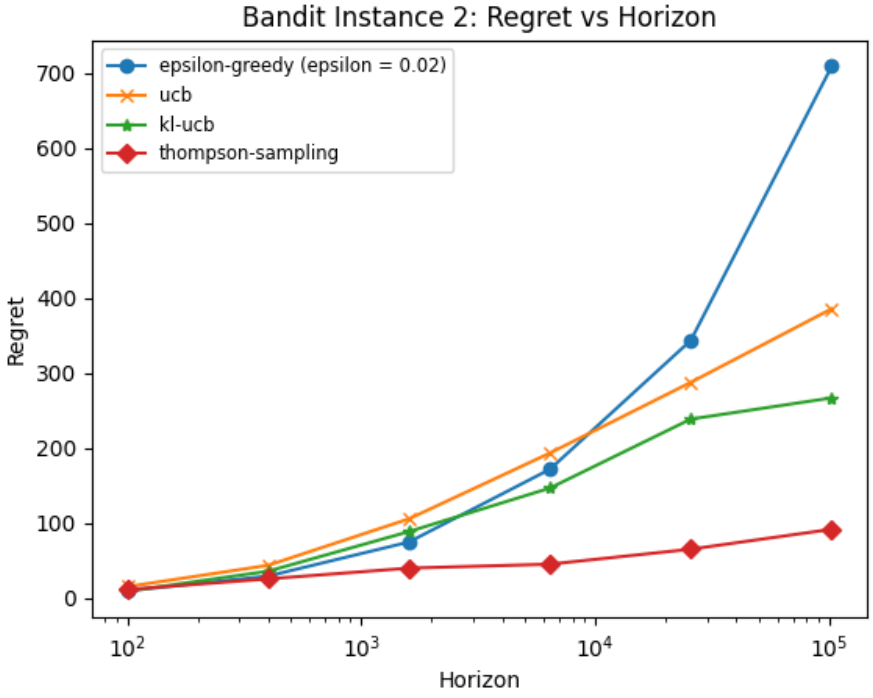

Figure 2: Regret vs Horizon Plot for Instance 2 (5-armed bandit) using all 4 algorithms

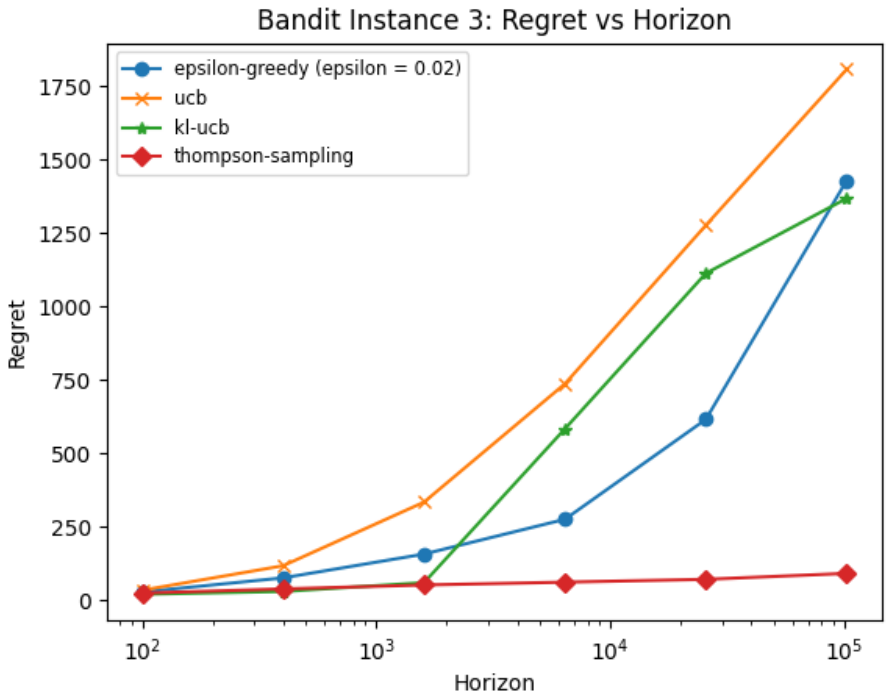

Figure 3: Regret vs Horizon Plot for Instance 3 (25-armed bandit) using all 4 algorithms

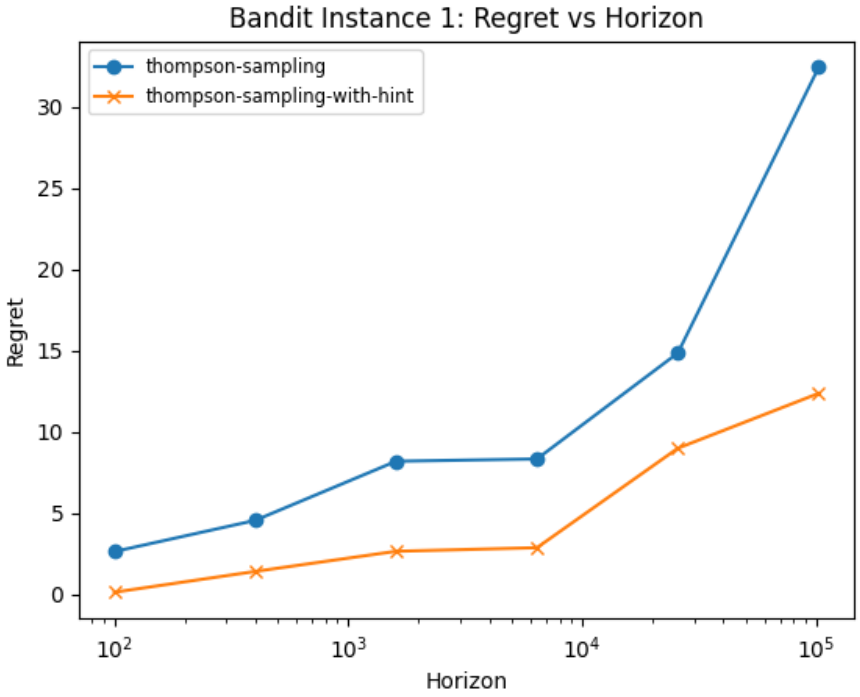

Figure 4: Regret vs Horizon Plot for Instance 1 (2-armed bandit)

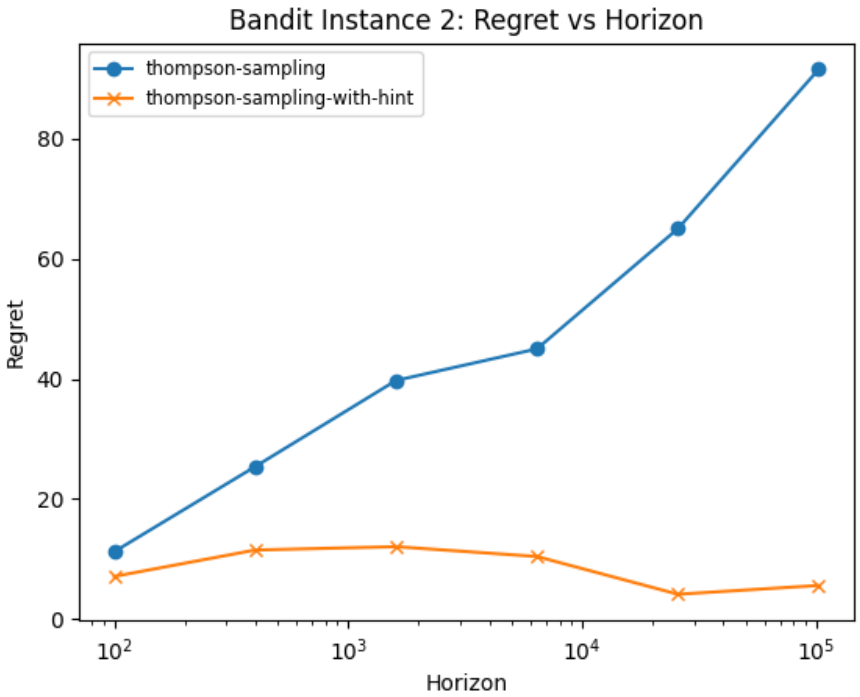

Figure 5: Regret vs Horizon Plot for Instance 2 (5-armed bandit)

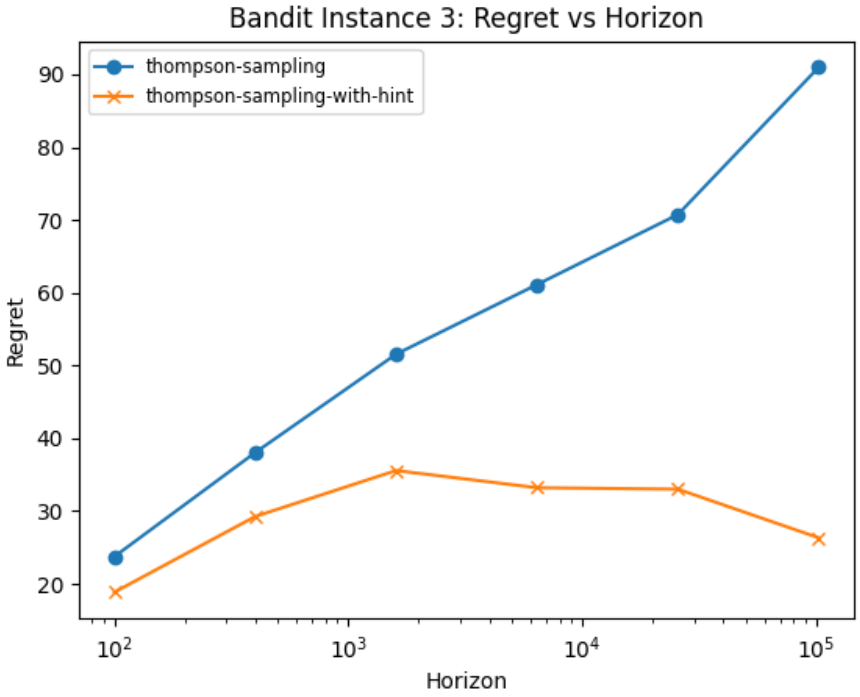

Figure 6: Regret vs Horizon Plot for Instance 3 (25-armed bandit)