Recommending Pairs of News Stories using Bandits

Student Michalis Lamprakis
AM 2020030077

Problem context

Display a pair of articles (i, j), where i is placed on top and j on bottom. The user clicks either i (with probability p_i) or j (with probability $(1 - p_i)p_j$, but not both. It is clear that the total expected reward is influenced by their joint configuration and given by the formula $p_i + (1 - p_i)p_j$. So the goal is to maximize the expected reward by selecting the best pair of articles. So we made some slight modifications to our algorithms to fit the evironment.

$$\mu^* = p_i^* + (1 - p_i^*)p_j^*$$

$$\mu_{(i,j)} = p_i + (1 - p_i)p_j$$

$$\Delta_{(i,j)} = \mu^* - \mu_{(i,j)}$$

Implementantion of the assignment (In Google Colab):
https://colab.research.google.com/drive/1S4G9UBfirLocYdTSE6vLyiHGKPBdcLDo?usp=sharing
The code contains detailed comments in which I explain everything.

Subtask 1

In this task i made slight modifications to the given Explore & Exploit code to fit the environment. Also, i implemented the UCB algorithm (i tried to follow the same structure as the first algorythm). So the main difference here compared to a UCB algorithm for single-arm bandits is that when a pair (i, j) is selected, we update the statistics ($\hat{\mu}_i(t)$ and $N_i(t)$) for both arms no matter which one was clicked. This is because the reward is a function of both arms.

In the UCB algorithm i use o formula to calculate the mean reward $\hat{\mu}_i$ when a new reward is recieved. Here is the proof of this:

1. The current $\hat{\mu}_i$ after n selections:
$$\hat{\mu}_n = \frac{r_1 + r_2 + ...r_n}{n}$$
2. When a new reward r_{n+1} is recieved, the new $\hat{\mu}_{n+1}$ is:
$$\hat{\mu}_{n+1} = \frac{r_1 + r_2 + ...r_n + r_{n+1}}{n+1} = \frac{n\hat{\mu}_n + r_{n+1}}{n+1}$$
3. We substract $\hat{\mu}_n$ from both sides:
$$\hat{\mu}_{n+1} - \hat{\mu}_n = \frac{n\hat{\mu}_n + r_{n+1}}{n+1} - \hat{\mu}_n = \frac{n\hat{\mu}_n + r_{n+1} - (n+1)\hat{\mu}_n}{n+1} = \frac{r_{n+1} - \hat{\mu}_n}{n+1} \Rightarrow \boxed{\hat{\mu}_{n+1} = \hat{\mu}_n + \frac{r_{n+1} - \hat{\mu}_n}{n+1}}$$

Results:
After implementing the two algorithms, we plot the accumulated rewards for both of them, comparing them to the optimal reward (the reward achieved by displaying only the optimal pair).

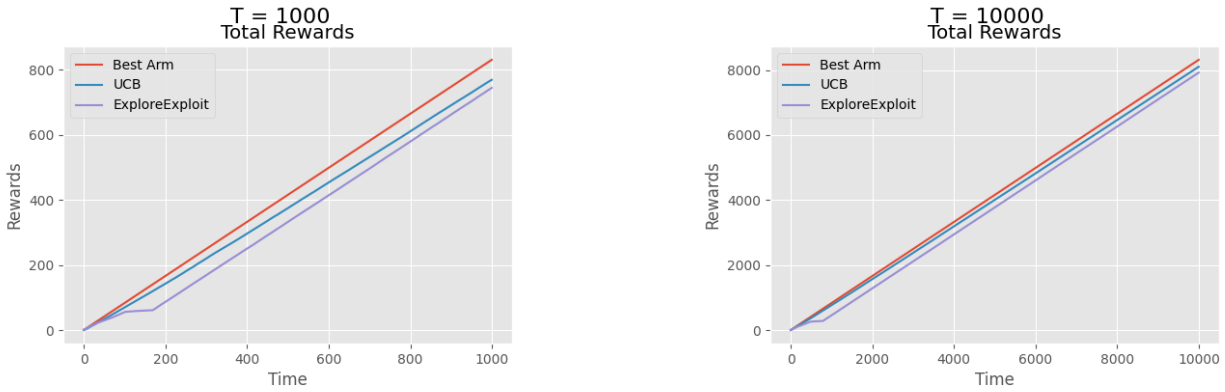

Figure 1: Comparison of rewards across different horizons.

The results validate UCB's superiority in stochastic Multi-Armed Bandit environments like the one in our problem, particularly for large horizons. UCB dynamically adjusts exploration, allowing it to focus on high-reward pairs early. On the other hand Explore & Exploit suffers from a fixed exploration phase, leading to slower adaptation. Especially for large horizons (like T=10000) UCB likely approaches the theoretical rewards bound, making it (almost) optimal. Also, in real-world scenarions where Horizon T is uknown, UCB seems the only choice.

Subtask 2

From our lesson we have proven the (instance-dependent) theoretical regret bound of UCB:

$$E[R(T)] \leq \sum_{i=1}^{K} \frac{8 \log(T)}{\Delta_i}$$

Since our problem selects pairs of articles (i, j), we extend this to all suboptimal pairs.Thus, regret is upper-bounded by:

$$E[R(T)] \leq \sum_{j=1}^{K} \sum_{i=1, i \neq j}^{K} \frac{8 \log(T)}{\Delta_{(i,j)}}$$

This extends the single-arm regret bound to our case, where we optimize for pairs instead of single choices.

Subtask 3

For subtask 3 we calculate the theoretical regret bound for the UCB algorithm in our problem.

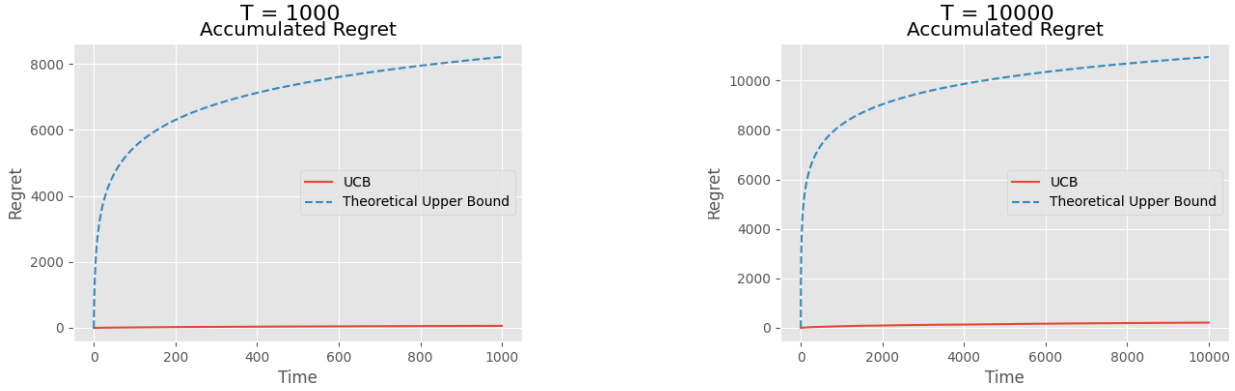

Figure 2: Comparison of regret bounds across different horizons.

In these plot the the theoretical regret curve is significantly larger than UCB's experimental regret at both horizons. This is expected because theoretical bounds are worst-case guarantees while the experimental are instance-dependent. While the experimental regret appears flat compared to the theoretical curve, its growth rate is consistent with logarithmic scaling ($O(\log T)$). What we can conclude is that the suboptimality gaps ($\Delta_{(i,j)}$) in this problem instance are, allowing UCB to quickly identify and exploit the optimal pair. We can confirm this by looking at the true probabilities of each arm ($p_1 = 0.64, p_2 = 0.45, p_3 = 0.53, p_4 = 0.1, p_5 = 0.06$). Therefore both grow slower than linear, indicating the efficiency of UCB.