CS747 Assignemnt #1

Hanish Dhanwalkar

February 18, 2025

TASK 1

1 Explaination of code

1.1 Upper Confidence Bound (UCB) Algorithm

1. **Initialization:**
   - Keeps track of the number of times each arm is pulled (counts) and the estimated value (values) of each arm.
   - t keeps track of the total number of pulls.
2. **Arm Selection:** ***give_pull*:**
   - For each arm, calculate the upper confidence bound (UCB) for that arm. Using
     $UCB(i) = R_m + sqrt(\frac{2\times\log(t)}{n})$, where R_m is mean reward, n= times arm i was pulled.
   - The arm with the highest UCB value is selected.
3. **Updating Rewards** ***get_reward*:**
   - Updates the mean reward estimate for the selected arm using an incremental formula:
     $Q_n = \frac{(n-1)Q_{n-1}+R_n}{n}$, where Q = value of the arm.
4. Regret vs. Horizon in Figure 2. (plot generated by simulator.py)

1.2 KL-UCB Algorithm

1. **Initialization:**
   - Similar to UCB but uses KL divergence instead of a direct confidence bound.
   - Keeps track of counts and values (mean rewards).
   - ϵ (epsilon) is used as a precision threshold for numerical root-finding.

2. **Arm Selection:** *give_pull*:

   - Uses Kullback-Leibler (KL) divergence to compute the upper confidence bound.
   - Defines a function that measures how much the estimated mean p deviates from an optimistic bound q.
   - Uses the bisection method to find q such that: $KL(p,q) = \frac{\ln t + 3 \ln \ln t}{u_t}$, where u_t is the number of times the arm has been pulled.

3. **Updating Rewards** *get_reward*:

   - Updates the running mean estimate of the selected arm in the same way as UCB.

4. Regret vs. Horizon in Figure 3. (plot generated by simulator.py)

1.3 Thompson Sampling

1. **Initialization:**

   - Uses Beta distributions to model uncertainty in each arm's reward probability.
   - initiates α (successes) and β (failures) for each arm, both initialized to 1.

2. **Arm Selection:** *give_pull*:

   - Samples a random value from each arm's $Beta(\alpha, \beta)$ distribution.
   - The arm with the highest sampled value is selected.

3. **Updating Rewards** *get_reward*:

   - If the selected arm gives a reward of 1, its α value is increased.
   - If the reward is 0, its β value is increased.

4. Regret vs. Horizon in Figure 4. (plot generated by simulator.py)

2 Results

Figure 1: Eps Greedy Algorithm. Regret vs. Horizon

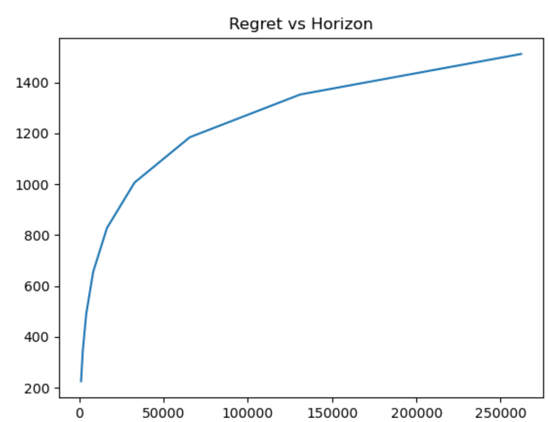

Figure 2: UCB Algorithm. Regret vs. Horizon

Figure 3: KL-UCB Algorithm. Regret vs. Horizon

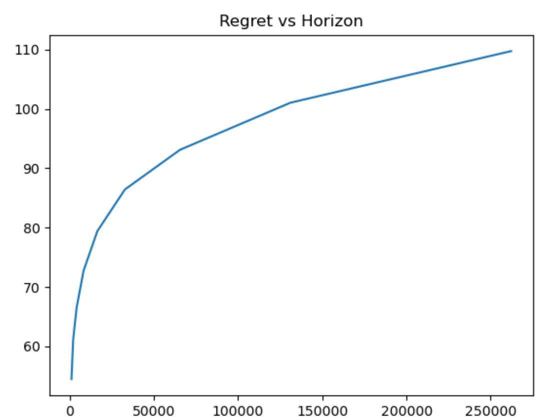

Figure 4: Thompson Sampling Algorithm. Regret vs. Horizon

TASK 2

2.1 Approach

1. **Tracking Rewards and Pulls:**
   - Maintains 2 arrays
     - *sum_rewards*: Keeps track of the total rewards obtained for each arm.
     - pulls: Stores how many times each arm has been pulled.
   - maintains a time step counter t to adjust confidence estimates over time.

2. **Upper Confidence Bound (UCB) Estimation:**
   - Each arm's expected value is estimated using UCB, which balances exploration and exploitation:

$$UCB(i) = R_m + sqrt(\frac{2 \times \log(t)}{n}) \tag{1}$$

   - If an arm has never been pulled, it is given a high UCB value (1e5) to encourage initial exploration.

3. **Choosing the Query Set:**
   - The arms are sorted in descending order of UCB values.
   - evaluates different possible sizes (m) for the query set and selects the one that maximizes the expected net reward:

$$\frac{\sum UCB\ values\ of\ chosen\ arm - 1}{m} \tag{2}$$

4. When the oracle returns a pulled arm and its reward, the algorithm updates the reward and pull count for that specific arm.

2.2 Explaination of the approach

1. The UCB-based selection ensures that the algorithm prioritizes arms that are promising while still exploring less-tested ones.

2. By choosing the query set dynamically, it optimizes the trade-off between expected reward and cost.

3. It avoids unnecessarily large query sets while still gaining useful information.

TASK 3

Observations:

1. **Zero Epsilon:** When epsilon is very zero, algorithm predominantly exploits the best-known action. This can lead to lowest regret as no exploration is done and always pulls optimal arm here.
2. **Medium Epsilon $0 < \epsilon < 1$:** When epsilon is between zero and one, the algorithm explores more frequently but also explores suboptimal actions, leading to higher regret.
3. **1 Epsilon:** When epsilon is high, the algorithm explores more frequently but also explores suboptimal actions, leading to higher regret.

Result:

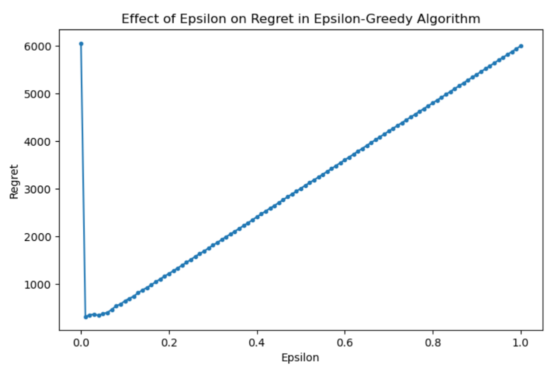

Figure 5: Eps Greedy Algorithm. Regret vs. Horizon