CS533-Assignment5-Report

Chunxiao Wang

1 Part I: Create Bandit Algorithms

Bandit algorithms, incremental uniform, UCB and ϵ-greedy, are implemented in bandit_algorithm.py. Bandit problems are implemented in bandit.py. The code is run in Python 3.6.

2 Part II: Bandit Design

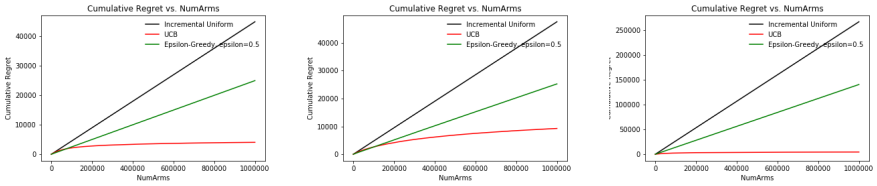
The self defined bandit is created with 20 arms. The first 10 arms has parameters (0.01, 1). The next 9 arms has parameters (0.9, 0.5). The last arm has parameter (1, 0.5). The optimal arm here should be the last arm with parameter (1, 0.5) since it has the highest expected reward 0.5(the other two are 0.01 and 0.45 respectively). But since 0.45 and 0.5 are closed, it will be interesting to test whether the bandit algorithms can pick up the optimal arm as the best arm or end up picking a nearly optimal arm as the best arm.

3 Part III: Evaluating Cumulative Regret

We compute the cumulative regret of the bandit algorithms for the three bandits created in Part II. Cumulative regret after N arm pulls is $E[Reg_N] = NR^* - \sum_{i=1}^{N} E[r_i]$, where R^* is the optimal expected reward, and $E[r_i]$ is the expected reward at time step i equal to $p_j r_j$ with j indicating the j-th arm pulled at time step i.

For each pair of algorithm A and bandit B, we choose $T = 100$ and $N = 100000$ to produce the curve.

For bandit 1, the best arm found from all three algorithms is always arm 10. For bandit 2, the best arm found from all three algorithms is always arm 19 and for bandit 3, the best arm found from all three algorithms is always arm 19. The first two match our expectation. For bandit 3, we don't see the nearly optimal arms (11-19) selected as the best arm. There might be two reasons. The first one is the algorithms are smart enough to choose the exact optimal arm. The second one might be the difference between 0.45 and 0.5 is too large to treat the i-th arm ($i = 11, \cdots, 19$) as the nearly optimal arms. Because of the time constraint, I stopped the exploration here. The cumulative regret curves for each bandit are as follows,



(a) bandit1 (b) bandit2 (c) self defined bandit

Figure 1: Evaluating Cumulative Regret

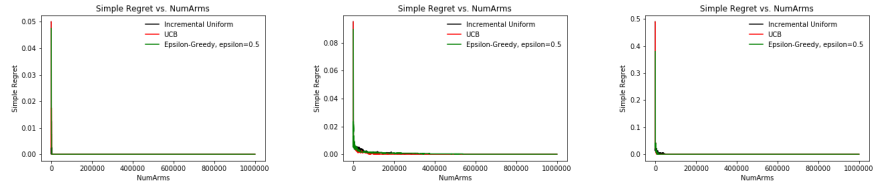
The figure shows that cumulative regret increases over time, which matches our expectations since at each time step it accumulates the sum of regrets. Also for all three bandits, the curves of UCB algorithm look logarithmic and have the best performance among the three algorithms, which also matches the expectation since cumulative regret grows $O(\log N)$ with N indicating the total number of arm pulls from the UCB theoretical bounds and it is designed specifically for the cumulative regret objective.

4 Part IV: Evaluating Simple Regret

We compute the cumulative regret of the bandit algorithms for the three bandits created in Part II. Simple regret is defined as $E[SReg_N] = R^* - E[R(a_{i_N})]$ after N pulls with R^* being the optimal expected reward and $E[R(a_{i_N})]$ being the expected reward of the best arm tracked by the algorithm, which in the case of SBRD bandits would be $p_j r_j$ where j is the best arm tracked so far. Thus, simple regret is the difference between the optimal expected reward and expected reward of the best arm selected by algorithm after N pulls.

For each pair of algorithm A and bandit B, we choose $T = 100$ and $N = 100000$ to produce the curve.

For bandit 1, the best arm found from all three algorithms is always arm 10. For bandit 2, the best arm found from all three algorithms is always arm 19 and for bandit 3, the best arm found from all three algorithms is always arm 19. The explanation is the same as in Part III. The simple regret curves for each bandit are as follows,



(a) bandit1 (b) bandit2 (c) self defined bandit

Figure 2: Evaluating Simple Regret

The plot shows that the simple regret is decreasing over time, which matches our expectations since simple regret doesn't accumulate sum of regrets over time but just compares the difference between the optimal expected reward and the expected reward of the best arm selected by the algorithm. For bandit1, the performance of the three algorithms are very closed. For bandit2 and self defined bandit, from the zoomed plot, UCB reduces the regret fastest, and then the ϵ-greedy algorithm while the incremental uniform performs the worst. This result matches our expectations since UCB reduces regret at a polynomial rate, incremental uniform reduces at an exponential rate and 0.5-greedy reducew at a better exponential rate. $N = 1000000$ is too large. We might need to use a smaller number of pulls to see more clear comparison.