Programming Assignment 2

Reinforcement Learning and Dynamic Optimization

Ιωάννης Χαριομπόλης 2017030054 *April 28, 2023*

You are given a dataset of real traffic loads over time (normalized), for a number of servers. These demands are non stationary.

Your goal is to devise algorithms that learn to predict the least loaded server at every time round. Imagine, for example, that you had to offload computing tasks, one after another, to one of these servers: the higher the load of the server at that time, the longer the delay for your own offloaded task as well.

Part I

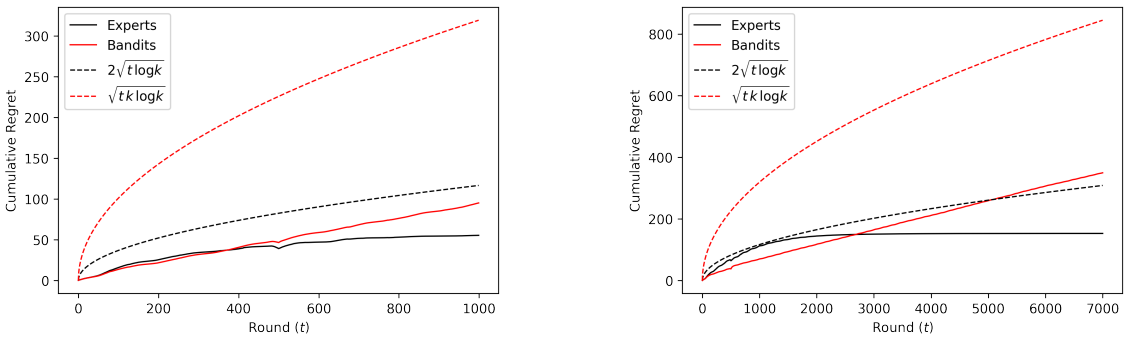
- Implement the Multiplicative Weights algorithm assuming an "Experts" environment (i.e., you get to learn the load of other servers at every round, not just the one you chose).

  We chose $\eta = \sqrt{\frac{\ln k}{T}}$.

- Implement the Multiplicative Weights algorithm assuming a "Bandit" environment.

  We chose $\eta = \epsilon = \sqrt[3]{\frac{k \ln k}{T}}$.

- Compare the cumulative regret of the two algorithms, for horizon values $T = 1000$, $T = 7000$ (entire duration of dataset).



(a) $T = 1000$ (b) $T = 7000$

Figure 1: Performance for the two implementations of the MW algorithm.

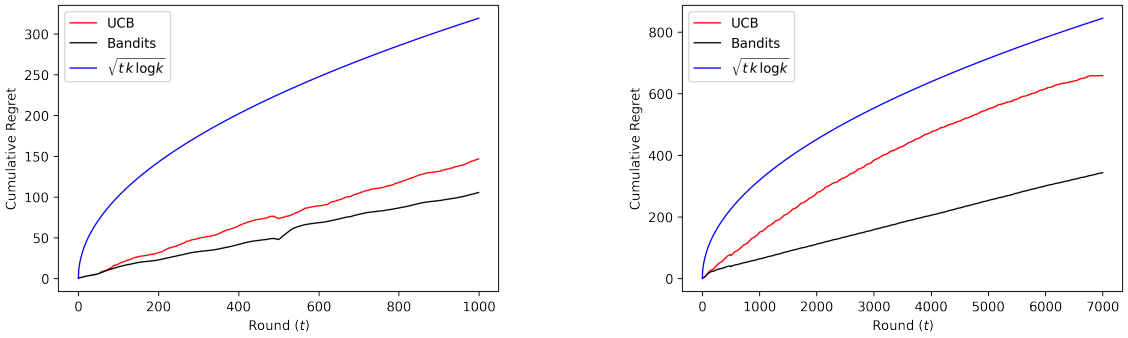
Both algorithms stay well below their respective theoretical regret bounds, represented by color coded dashed curves. We can see that even though the bandits algorithm for the case where $T = 7000$ has a smaller regret initially it eventually even surpasses the regret bound for the experts algorithm, clearly showing its inferior performance. Although this doesn't necessarily happen for $T = 1000$, it performs worse there as well, especially after the midpoint of the horizon.

Part II

- Adapt UCB to this problem as well (remember, you now have losses, not rewards as in the previous assignment).

  We set as the reward $r_i^{(t)}$ equal to the loss but negated, $-l_i^{(t)}$, which then makes the i with the largest $\text{ucb}_i(t) = \hat{\mu}_i + \sqrt{\frac{\ln T}{Q_i(t)}}$ be either the one with the smallest loss or the one that needs to be explored more, as we require.

- Compare the performance of UCB to that of MW algorithm for the bandit setting (again for $T = 1000, T = 7000$).



(a) $T = 1000$ (b) $T = 7000$

Figure 2: Performance of the UCB and the bandit MW algorithms.

As expected the UCB algorithm performs worse than the bandit multiplicative weights algorithm, nonetheless it never surpasses the upper regret bound, even for the case where $T = 7000$.