Reinforcement Learning and Dynamic Optimization
Assignment 2: Experts and Adversarial Bandits
Report Delivery Date: Friday, April 28, 2023

Student: Alevrakis Dimitrios 2017030001

Assignment Description
We are given a dataset of real traffic loads over time (normalized), for a number of servers. These demands are non stationary. The dataset was given in csv file

The 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 We implement the Multiplicate Weights algorithm assuming an experts or a bandits environment.

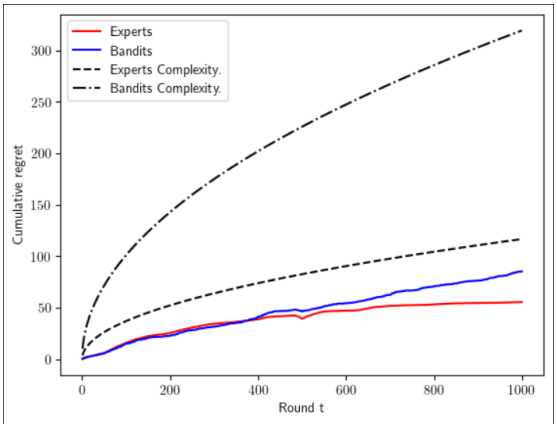
Following the suggestions of the uploaded lecture notes the parameters chosen where: For the Experts environment:

$$\eta = \sqrt{\frac{lnk}{T}} \tag{1}$$

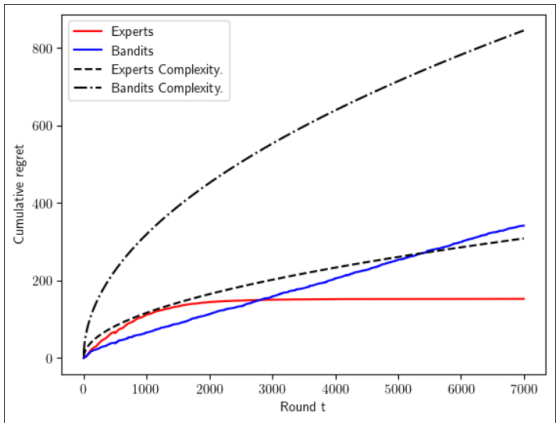
and for the Bandits environment:

$$\eta = \epsilon = \sqrt[3]{\frac{lnk}{T}} \tag{2}$$

Comparison of the cumulative regret of each algorithm, for horizon values $T = 1000, 7000$.



(a) $T = 1000$



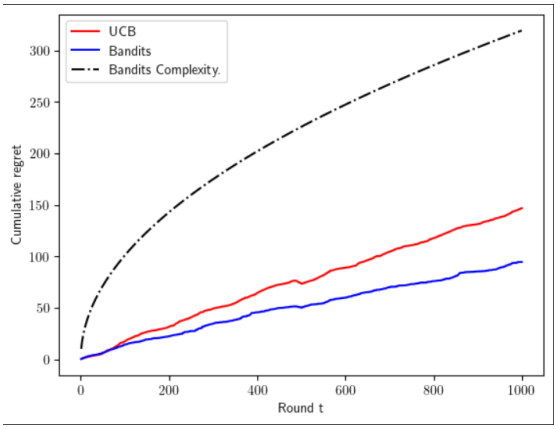
(b) $T = 7000$

By observing the graphs we can see that that in both cases the algorithm achieve a Regret lower than their upper bounds. We can also observe that while for a lower horizon the two algorithms have comparable performance. Although for a bigger horizon the hostile environment of the bandits algorithm is made obvious by its significantly worse performance than the experts algorithm.

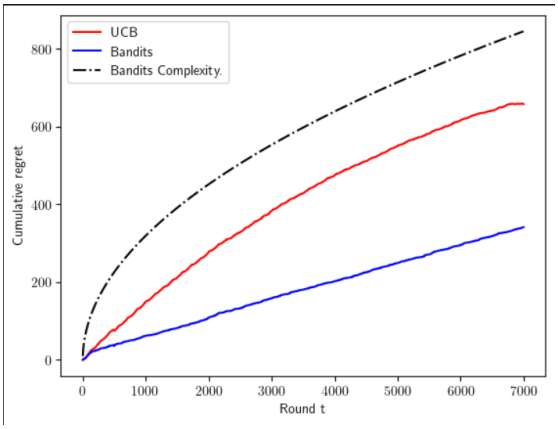
The respective algorithm upper bounds:

- Experts: $O(2\sqrt{tlogk})$
- Bandits: $O(\sqrt{tklogk})$

Part II We adapt the UCB algorithm to the above problem.
In order for the algorithm to be adapted we consider the reward as the loss and calculating $ucb_i = -\mu_i + \sqrt{\frac{lnT}{Q_i(t)}}$ in order to choose the arm yielding the lower loss or the arm chosen the least.



(a) $T = 1000$



(b) $T = 7000$

Since the UCB algorithm is deterministic it fails in a bandit environment thus performing worse than the bandit algorithm.