COMP423 - Reinforcement Learning and Dynamic Optimization
2nd Programming Assignment Report:
Experts and Adversarial Bandits

Pantourakis Michail AM 2015030185
School of Electrical and Conputer Engineering
Technical University of Crete

Date submitted: 16 April 2023

In this report I analyze the performance of Multiplicative Weights (MW) algorithms in "Experts" and "Adversarial Bandits" environments. As an example, a normalized dataset of real traffic loads for $k = 30$ servers and a duration of $T = 7000$ rounds was provided. The goal of the algorithms is to predict and pick the least loaded server at each time round.

Part I

1.1 MW and hyperparameter values

The implementation between the MW variants has a shared base structure, varying only in the amount of information acquired per round (Experts vs Bandits environment) and how this assumption is then applied to updating the weights for the selection process.

More specifically, for the Experts environment, the learning algorithm acquires the normalized load (loss) of every server at each round t. Following our lecture notes [1], I implemented the updating algorithm and set the hyperparameter $\eta = \sqrt{\frac{\ln k}{T}}$, which theoretically gives sublinear regret over the horizon ($O(\sqrt{T \ln k})$).

In regards to the Bandits' environment, the algorithm only gets the load of the selected server at round t. Following the respective analysis in our lecture notes [2], we now have to set an exploration hyperparameter ϵ, which is a balancing factor between exploration and exploitation. In this report, I present results for two values: 1) $\eta = \epsilon = \sqrt{\frac{\ln k}{T}}$ (titled "explore less"), and 2) $\eta = \epsilon = \sqrt[3]{\frac{n \ln k}{T}}$ (titled "explore more").

1.2 Experts vs Bandits

Figure 1 shows the results of one realization per algorithm for $T = 1000$ (Subfigures a-c) and $T = 7000$ (Subfigures d-f). Subfigures (a,d) show the total loss accumulated at each round t, (b,e) show the cumulative regret at each t, and (c,f) show the average regret over t. For the calculation of regrets, I used the loss provided by picking the best server over the horizon (server i = 12 in both cases).

Expectedly due to full observability, the Experts setup has the superior performance over both small and large horizon T. At $T = 7000$ it seems it has reached a point where the cumulative regret no longer significantly increases. On the contrary, the adversarial bandits setup accumulates more regret over time, and still shows accumulation at the end of the horizon. Nevertheless, the curve for the algorithm with less exploration has a shape similar to that of Experts, which is indicative of the similar regret bound of $O(\sqrt{kT \ln k})$ (the extra k term being the penalty of partial observability).

[1] lecture19, Expert_Adv_Bandit_Lecture
[2] lecture20, Expert_Adv_Bandit_Lecture

1.3 Exploration with Bandits

Interesting observations are made by inspecting the behavior of the "explore more" adversarial bandits case. Since η is also higher for this implementation, it is obvious that bad servers are more severely penalized, resulting in better server choices in the short term. However, since the exploration probability is higher too, the "explore more" Bandits variant eventually accumulates more regret than both the Experts and the "explore less" variants. While the "explore more" variant seems at a long-term disadvantage, we should factor in the purpose of this algorithm: high value of ϵ may lead to more bad options, but the reasoning behind it is that under an adversarial setting, these so-far bad options may turn out to be good. Therefore, we should expect this algorithm to adapt faster to sudden changes of the best server over time.

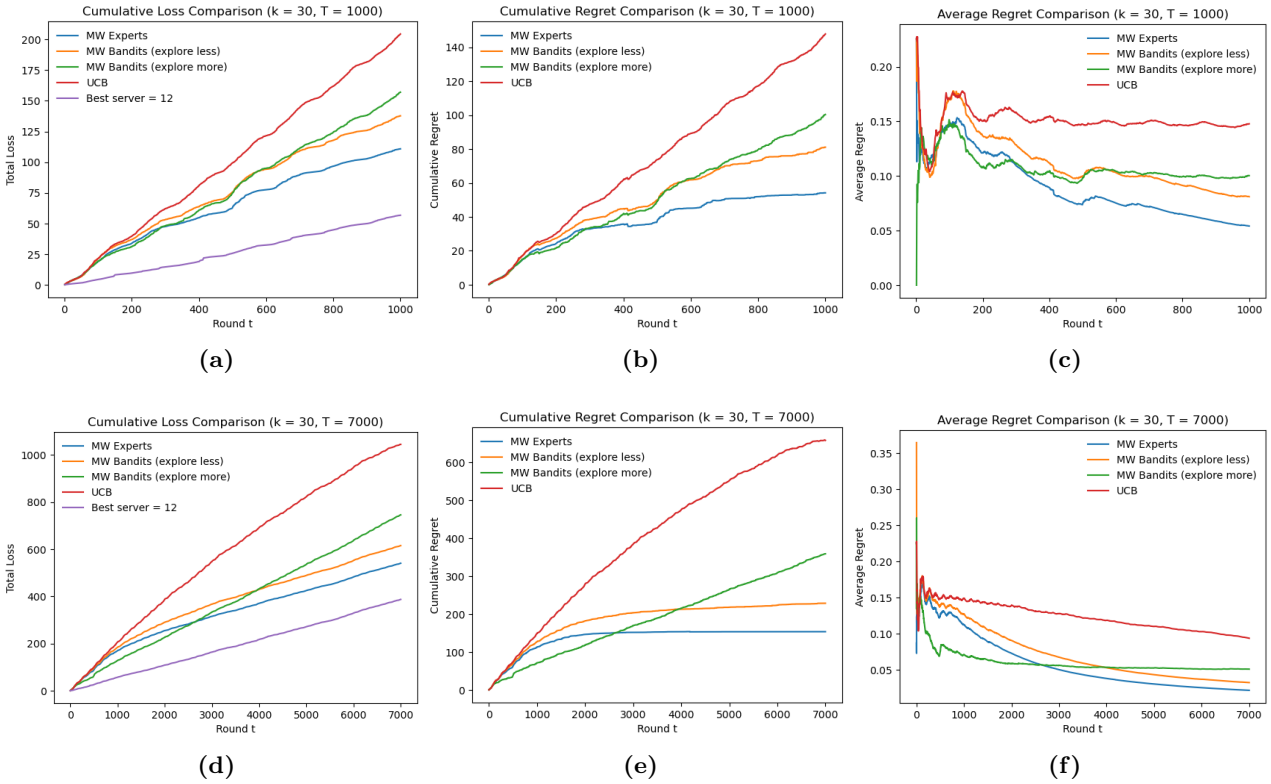

Figure 1: Performance of Multiplicative Weights (MW) and Upper Confidence Bound (UCB) algorithms on predicting the least loaded server out of $k = 30$ at every time round t.

Part II

2.1 Modifications to UCB for losses instead of rewards

In this part, I adapted the UCB algorithm of programming assignment 1 to this assignment's problem. Since in assignment 1 the algorithm was maximizing rewards instead of minimizing losses, I simply negated the provided dataset values. As a result, no modification was required for the algorithm internally, and the algorithm simply maximized negative losses, which is equivalent to minimizing the losses (as positive values).

2.2 UCB vs Bandits

As expected for adversarial bandits, UCB consistently exhibits the worst performance, which agrees with the theoretical linear regret bound. Nevertheless, for this particular dataset, it is clear that even UCB shows a curve saturation over time. If a particular server is consistently and/or significantly better than the others over the horizon, which seems to be the case for server $i = 12$, then the environment is practically less "adversarial" and more "stochastic" in nature, and UCB will predictably perform better over time.