Price Optimization Using Bandits

Student Michalis Lamprakis
AM 2020030077

Problem context

I have a service that i don't know how to price. In my setup there are these options:

$$p_0 = 0.6, p_1 = 1.21, p_2 = 2.42, p_3 = 3.63, p_4 = 5.44$$

The competitor of this service gives a random price $Q \sim [0, 4]$ to each user. In **Subtask 1** user picks randomly between the 2 sellers. On the other hand, in **Subtask 2** after $T/4$ rounds the user picks the seller with the lowest price (the cheapest one!). If the user picks our service, we get a reward equal to the price of the current service, otherwise the reward is 0.

Implementantion of the assignment (In Google Colab):
https://colab.research.google.com/drive/1KPvHqkYBGa5-dhWF5EYeK0ixty2ZK4gm?usp=sharing
The code contains detailed comments in which I explain everything.

Implementantion

In order to use the Multiplicative Weights algorithm in a Bandit setup, we need to use a modified version of the algorithm. First of all, as the problem gives rewards, we turn them into losses by comparing them to the maximum reward (which in our seed is $p_4 = 5.44$) and normalizing it so $l_i^t \epsilon [0, 1]$. We use the formula:

$$l_i^t = \frac{p_4 - r^t}{p_4}$$

Where r^t is the reward of the current round. Estimated loss is calculated using the formula:

$$\hat{l}_i^t = \begin{cases} \frac{l_i^t}{q_i} & \text{for chosen arm } i \\ 0 & \text{otherwise} \end{cases}$$

The q term adds exploration to the environment and is equal to $q_i = (1 - \varepsilon) \cdot p_i^t + \frac{\varepsilon}{k}$. Then we use $\hat{l}_i^t$ to update the weights of all arms $w_i^{t+1} = (1 - \eta)^{\hat{l}_i^t} \cdot w_i^t$. The parameters η and ε as mentioned in the Notes are set to $\eta = \varepsilon = \sqrt{\frac{\ln k}{T}}$ so as to achieve sublinear regret.

The regret is calculated using the formula:

$$R^T = L_{ALG} - L_{OPT}.$$

Where L_{ALG} is the loss of the algorithm and L_{OPT} is the loss of the optimal arm (chosen in the end of the 'game').

Subtask 1

In Subtask 1 the user doesn't care about the price and picks randomly between the 2 prices. As a result the MW algorithm converges to a policy that heavily favors the highest price among the 5 choices. Initially, all prices are explored uniformly due to equal weights. Over time, weights for higher prices grow faster because their rewards (when selected) are larger. This behaviour is verified in **Figure 1**, where we can see that the algorithm gives the highest chosen probability to the highest price and lower prices are gradually deprioritized (they still have a small probability due to factor ε).

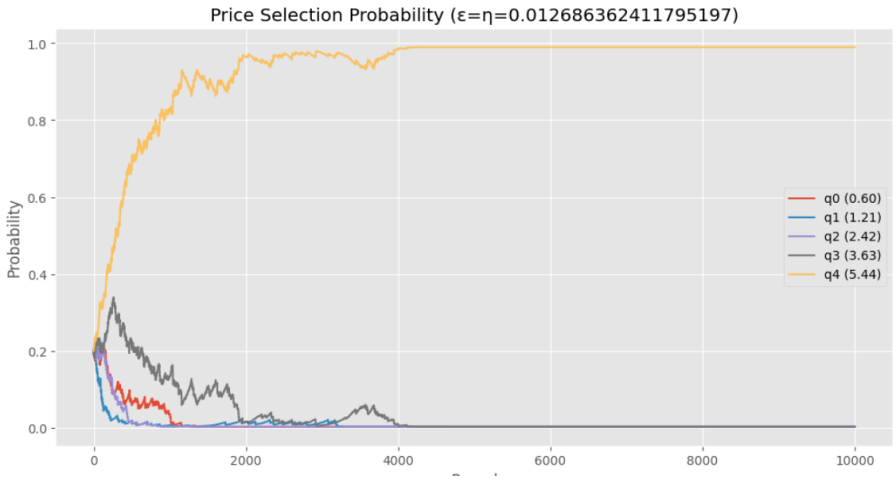

Figure 1: Probability distribution of prices over time.

As regards the rewards, keeping in mind that the algoorithm converges to the highest price, we can easily estimate that the total profit approaches the maximum reward of the optimal devided in half(as the chosen probability is 0.5).

$$profit_{MW} = \frac{p_4}{2} \cdot T = \frac{5.44}{2} \cdot 10000 = 27200$$

This is also verified in **Figure 2**.

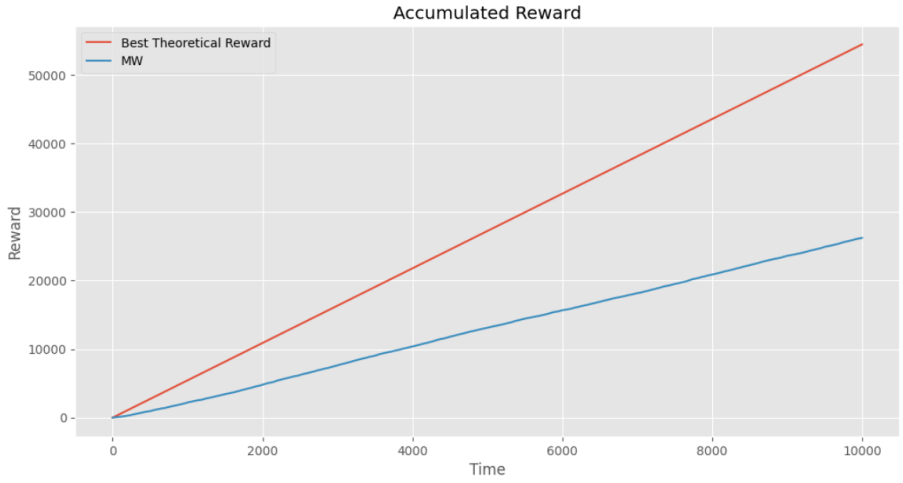

Figure 2: Accumulated reward over time.

Lastly, we observe sublinear regret, indicating that the algorithm is efficient.

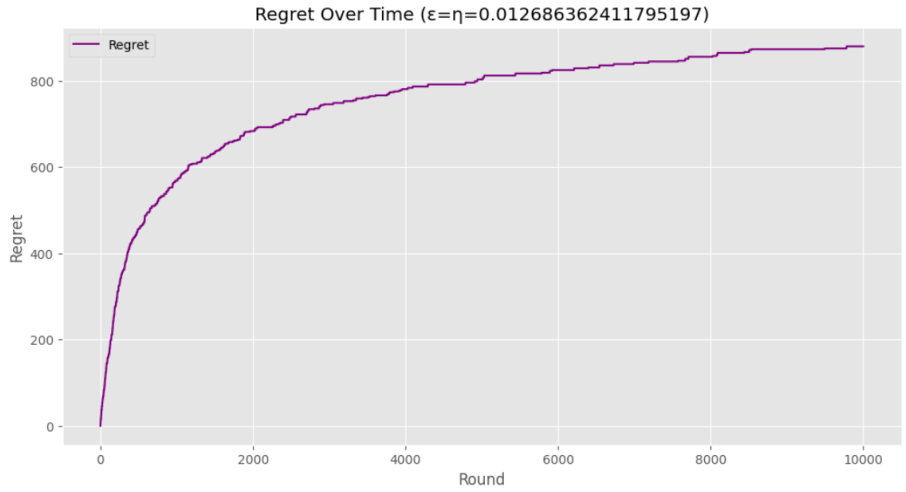

Figure 3: Accumulated regret over time.

Subtask 2

So, in the second subtask, the user fot the first T/4 rounds picks randomly between the 2 prices and then picks the cheapest one. For the first T/4 rounds, the algorithm behaves similarly to subtask 1. However, for the rest of the rounds, if MW continues favoring high prices (learned from the initial phase), the seller frequently loses to competitors offering lower prices (our max price is 5.44 while the competitor's is 4). This means that the algorithm needs to adapt to the new environment and converge to a new price. This new price must be lower so as to be competitive with the competitor's price. This is exactly what we observe in **Figure 4**, where the algorithm after T/4 changes behaviour.

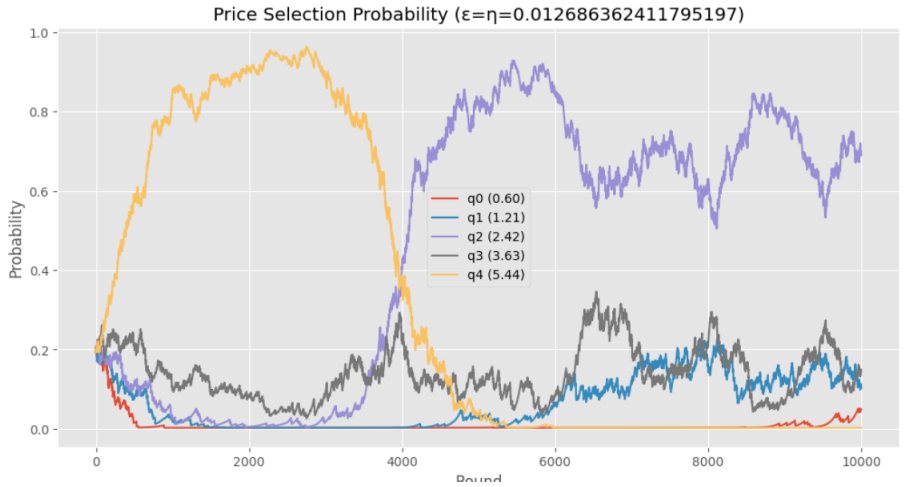

Figure 4: Probability distribution of prices over time.

As a result, after the first T/4 rounds, the algorithm starts to favor lower prices and we expect lower profits, compared to subtask 1.

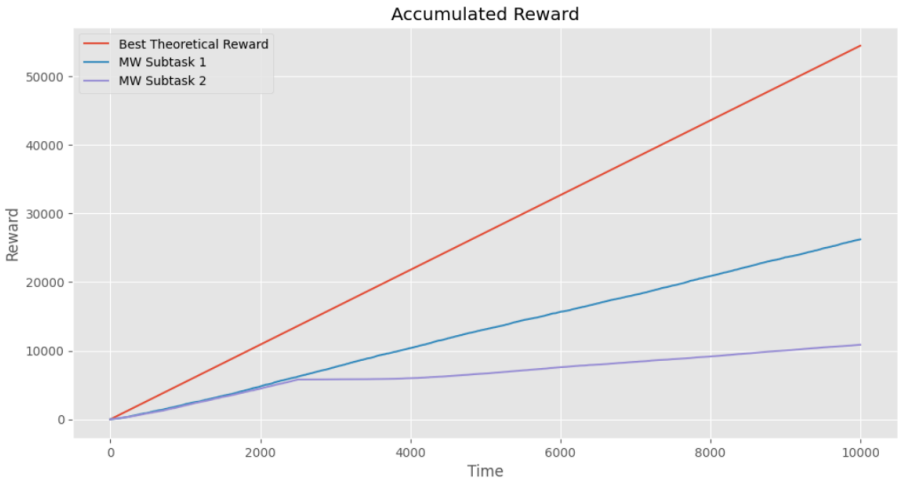

Figure 5: Accumulated rewards of the 2 subtasks.

Last but not least, the regret is more complicated here. For the first T/4 rounds the regret starts decreasing and that means $L_{ALG} < L_{OPT}$. This is reasonable beacause at the first rounds the algorithm converges to the highest price(as the user picks randomly). The highest price is not the optimal one and thats why the regret is negative. After the first T/4 rounds, the regret starts increasing sublinear over T and that means that the algorithm adapts to the new environment.

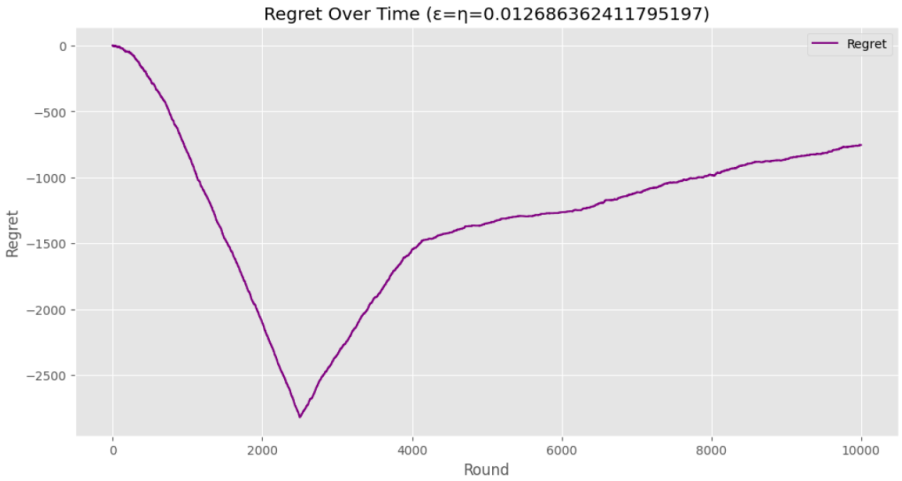

Figure 6: Accumulated regret.