INTRODUCTION TO REINFORCEMENT LEARNING EXPLORATION/EXPLOITATION ON BANDITS

Irie Railton
S3292037
i.r.railton@umail.leidenuniv.nl

Pablo Ubilla Pavez
S3430839
p.ubilla.pavez@umail.leidenuniv.nl

ABSTRACT

This report investigates the use of three different algorithms to maximize reward in a simulated bandit environment where the probability of pulling a reward for each arm of the bandit is unknown, but comes from a standard uniform distribution. The algorithms are ϵ-greedy, optimistic initialization, and upper confidence bounds. Each of these algorithms offers some exploration and exploitation of the bandit problem which can be tuned with hyper-parameters to increase the reward. The hyper-parameters that were experimented with for each algorithm are, respectively, the ϵ value, the initial value, and the exploration constant. The experiments showed that the upper confidence bounds algorithm is the most effective at maximizing the reward and any setting of the exploration constant between 0 and 0.25 will yield a higher reward than the other two algorithms. The most effective exploration constant value we discovered was 0.25.

1 INTRODUCTION

In the following report we will study three bandit algorithms: ϵ - greedy, optimistic initialization and upper confidence bounds. The main idea of these algorithms is to find a way to explore and exploit a multi-armed bandit. Specifically, we will work with a 10 armed bandit, $a \in \mathcal{A}$ (set of arms), where each arms pulls a reward, r_a, that distributes Bernoulli with paramater μ_a. We don't know the parameter μ_a for each $a \in \mathcal{A}$, but we do know that: $\mu_a \sim U(0, 1)$. Hence, the problem itself consists in studying how the algorithms explore the different arms in order to determine a good estimate for the mean values that eventually exploits the Bandit configuration (and we will see how efficiently different algorithms get to an optimum action selection). This whole idea is based on a trade off between exploration and exploitation, were the algorithm should find a balance between those two in order to maximize the reward in the fewest steps possible.

For every algorithm we will be studying average rewards, meaning that we will run 500 repetitions of every experiment done and keep the value of the mean within those repetitions.

2 ϵ-GREEDY

2.1 METHOD

This methods consists of a greedy selection of the optimal action. This means, by default we would choose the action $a \in \mathcal{A}$ (where the action refers to the arm selected) that provides the maximum mean estimate: $Q(a)$. However, there would be a probability ϵ of choosing an action that doesn't have the maximum mean estimate. This means that there would be a fixed probability ϵ of exploring in each time-step, meaning that after many repetitions we would eventually have tried all the actions a certain amount of times (and this would give an estimate of the mean reward for that arm).

$$\pi_{\epsilon-\text{ greedy}}(a) = \begin{cases} \frac{1-\epsilon}{|\mathcal{A}^*|}, & \text{if } a \in \mathcal{A}^* \\ \frac{\epsilon}{|\mathcal{A} \setminus \mathcal{A}^*|}, & \text{otherwise} \end{cases}$$

Where $\mathcal{A}$ is the set of all actions available and $\mathcal{A}^* = \arg\max_{\mathcal{A}} Q$.

In this algorithm we will initialize the mean values $Q(a)$ for each a to 0, and we will also keep a counter $n(a)$ that indicates how many times have we chosen action a. They would follow the following update rule for every step.

$$n(a) \leftarrow n(a) + 1$$
$$Q(a) \leftarrow Q(a) + \frac{1}{n(a)}[r(a) - Q(a)]$$

We will study the reward over different configurations of the algorithm, changing the ϵ value within: 0.01, 0.05, 0.1, 0.25.

2.2 Result

As we can see in 1, there is a difference in the average reward curves for different ϵ values in this algorithm. We can see that when the curves plateau there is a noticeable gap between the average rewards, this implies that this method is sensitive to hyper-parameter tuning. This is expected due to the fact that the ϵ value is constant through the evolution of the algorithm, meaning that even after many timesteps, the policy could still choose an action that has a low mean estimate, and this would decrease the average reward through the repetitions.

In general we can observe that within the ϵ values explored the method always get to an average reward of at least 0.7, which is indeed better than the expected average of choosing an action at random (0.5).

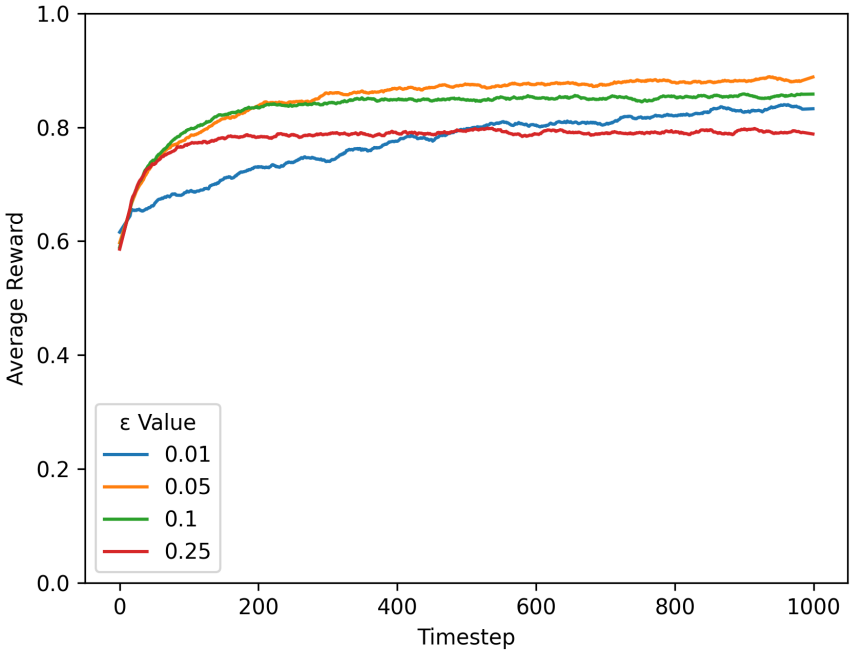

Figure 1: ϵ-Greedy algorithm average reward over 500 repetitions for 1000 timesteps compared for different values of ϵ

3 Optimistic Initialization

3.1 Method

The optimistic initialization policy starts by initializing all arms with a mean value above 0. We will experiment with this hyper-parameter by using values of 0.1, 0.5, 1.0, and 2.0 as our initial mean

estimates (Q_0). This policy also uses greedy action selection. However, unlike the ϵ-greedy policy, there is no chance of selecting an action that doesn't have the highest mean estimate. If there is more than one highest mean estimate it will randomly sample an action from the list of all highest mean estimates. This equation for selecting an action is as follows:

$$\pi(a) = \begin{cases} \frac{1}{|\mathcal{A}^*|}, & \text{if } a \in \mathcal{A}^* \\ 0, & \text{otherwise} \end{cases}$$

The update rule for the mean, $Q(a)$, is learning based and uses a fixed learning rate of $\alpha = 0.1$. The update rule is as followed:

$$Q(a) \leftarrow Q(a) + \alpha[r - Q(a)]$$

3.2 Result

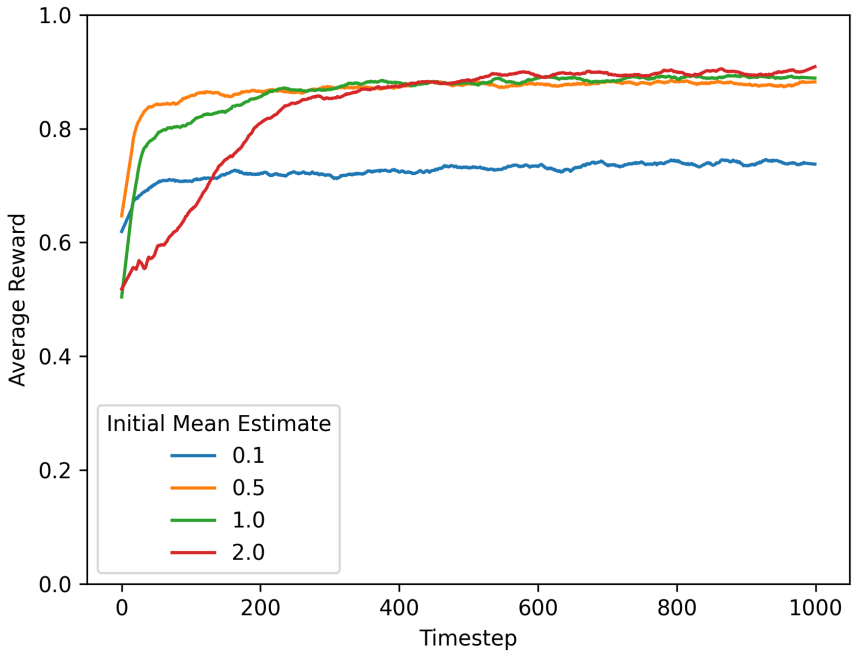

Figure 2: Optimistic initialization algorithm average reward over 500 repetitions for 1000 timesteps compared for different values of ϵ

The results, as seen in 2, show a clear difference in the average reward curves over time for different values of the initial mean estimate. The curve for an initial mean of 0.1 quickly plateaus at a lower average reward than the other curves. This shows that 0.1 is not a high enough value for sufficient exploration. The trend of a higher initial mean estimate resulting in a higher average reward after 1000 timesteps continues for the other values of 0.5, 1.0, and 2.0, although it is a very small difference.

All 3 of the higher initial mean estimates converge on roughly 0.85 average reward, with some minor fluctuation. However, the higher the value for the initial mean estimate, the longer it takes for the reward to converge on this value. This can be most clearly seen with the 2.0 curve that doesn't plateau until about 500 timesteps as compared to the 0.5 curve which arrives at its peak after about 100 timesteps. This shows that 0.5 is the optimal value for an initial mean estimate.

4 Upper Confidence Bounds (UCB)

4.1 Method

The idea of this method is to construct a confidence interval for the reward distribution of every arm. We would choose the actions based on a good case scenario for this intervals, choosing the one that grants the best mean estimate, but in addition to a confidence term that decreases with the number of time we have chosen the action.

$$\pi(a) = \begin{cases} 1, & \text{if } a \in \mathcal{A}' \\ 0, & \text{otherwise} \end{cases}$$

Where $\mathcal{A}' = \arg\max_{\mathcal{A}} h$, considering: $h(a) = \begin{cases} \infty, & \text{if } n(a) = 0 \\ Q(a) + c\sqrt{\frac{ln(t)}{n(a)}}, & \text{otherwise} \end{cases}$.

We also consider that $c \in \mathbf{R}^+$ is an exploration constant, $n(a)$ is the number of times we have chosen action a and t is the timestep.

The update rule would be the same used in the ϵ-greedy algorithm.

We will study the reward over different configurations of the algorithm, changing the exploration constant value within: 0.01, 0.05, 0.1, 0.25, 0.5, 1.

4.2 Result

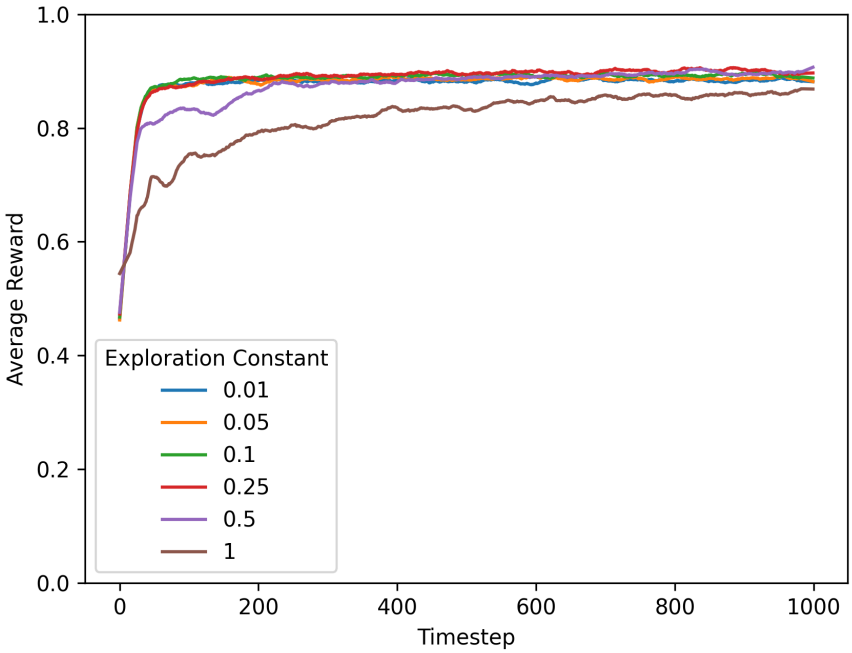

Figure 3: Upper confidence bounds algorithm average reward over 500 repetitions for 1000 timesteps compared for different values of ϵ

In figure 3 we can see how the average reward evolution behaves for different exploration constant values. We can notice than most curves have a very similar shape except for exploration constants equal 1 and 0.5. This could imply that algorithm works well with a wide range of values for the exploration constant (at least from 0.01 to 0.25). Probably when the exploration constant gets to high (like 0.5 or 1), the algorithm takes more time to start fully exploiting the best bandits, since the actions that grant a lower expected reward could still be compensated since the added argument to

the expected means in the action selection decreases with the amount of times the action has been selected.

5 Discussion

In this section we will study how the different algorithms vary depending on the value of the different parameters utilized. In figure 4 we can observe how the performance varies in every method studied with different values for the parameters, where the average reward is calculated as follows: $\bar{r} = \frac{1}{N \cdot T} \sum_{n=1}^{N} \sum_{t=1}^{T} r_{t,n}$. We will use the results that were gathered in the three experiments for comparison.

We should keep in mind that changes in parameter can only be compared inside one method, since the parameters have different implications for each algorithm. Therefore we shouldn't make any kind of comparison between parameter values of different algorithms.

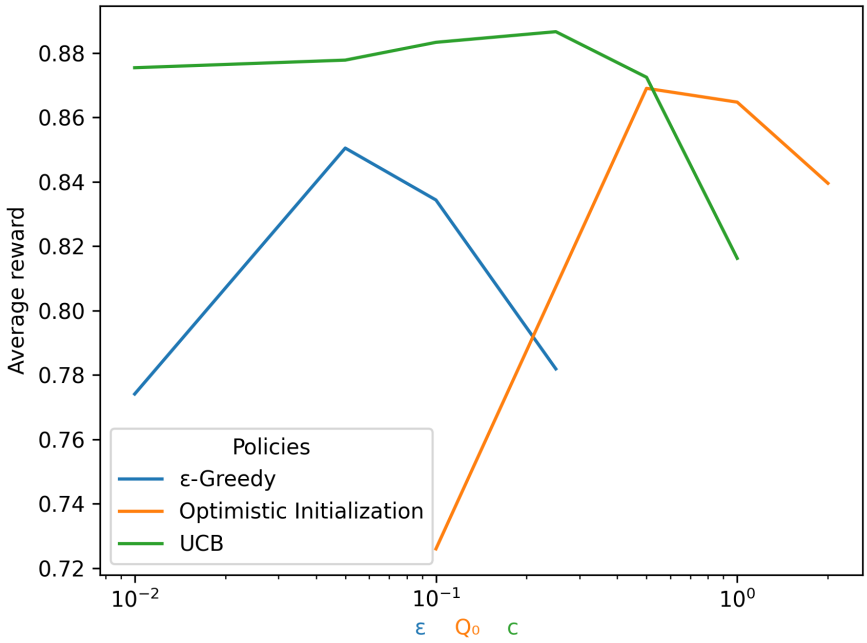

Figure 4: Average reward over 1000 timesteps and 500 repetitions for different parameter values in each algorithm

In general we can observe that every method performance has a clear dependence on the parameter values chosen. The 3 curves observed have a 'somewhat' concave form, which implies the presence of at least a local optimum regarding the average reward in terms of the parameter. It should be noted that since the list of parameter values for each method is not extensive, this wouldn't imply that the best average reward obtained should be an optimum (just the best performance within the values explored). The best hyper-parameters studied for each method are the following: ϵ-greedy has $\epsilon = 0.05$, OI has $Q_0 = 0.5$ and UCB has $c = 0.25$.

It is noted that UCB performance is particularly good compared to the other algorithms, within different parameter values. This could imply that this is a good choice, considering not much variation regarding parameter-tuning and a consistent better performance regarding the methods studied.

Plotting the average reward for the 3 bandit algorithms 5, with only the optimal parameter (of the values explored) for each policy, we can clearly see that the UCB policy performs the best. The ϵ-greedy and optimistic initialization algorithms both converge on a similar value after 500 timesteps. However, the ϵ-greedy policy takes longer to reach this due to its exploration with the possibility of choosing a random action instead of the optimal one.

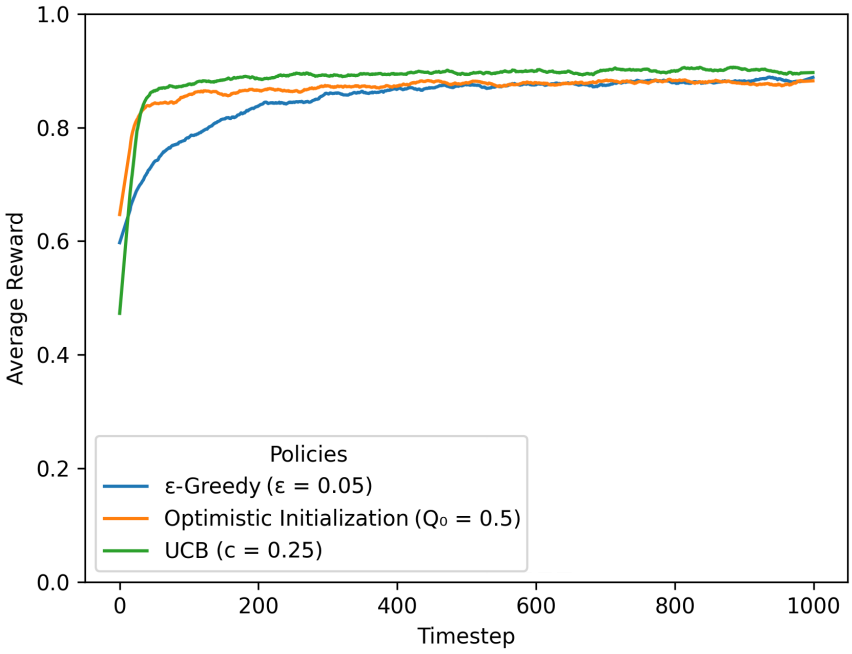

Figure 5: Average reward over 500 repetitions for 1000 timesteps compared for 3 bandit policies with optimized parameters

6 Conclusion

After experimenting with the ϵ-greedy, optimistic initialization, and upper confidence bounds algorithms we can conclude that the UCB algorithm yields the highest average reward within the repetitions and timesteps. It works well for several low values of the exploration constant hyper-parameter but the most effective value we discovered was 0.25. This should indicate that UCB is in general a good algorithm for this bandit environment, regarding it does not take much exploration to start an exploitation phase with an average reward around 0.85. All algorithms studied grant average rewards of at least 0.7 when optimizing the hyper-parameters, however, it is clearly seen that the different algorithms, and the different configuration of these algorithms, grant changes in how much time it takes for the curves to reach its plateau and how good the average reward is when this happens. This shows that is important to do hyper-parameter tuning and that more complex algorithms can make a significant improvement in the reward obtained and the time it takes to get to an exploitation phase.

For future research one could experiment with this hyper-parameter to fine tune the value further or explore other algorithms entirely. We could also explore further the trade-off between exploration and exploitation in each algorithm.