Reinforcement Learning: K-Armed Bandits

Mateo Guaman Castro

September 18, 2018

Abstract

In this homework assignment, the problem of non-stationary k-armed bandits was explored in the context of different action-value estimate methods. The two methods explored were action-value estimation using incrementally computed sample averages and action-value estimation using incrementally computed weighted averages. The action-selection methods used were epsilon-greedy and greedy methods. Overall, it was shown in this assignment that the incrementally computed weighted averages performed better than the incrementally computed sample averages in this context of non-stationary k-armed bandit problems.

1 Background

Multi-arm bandit problems, also known as k-armed bandit problems, are a simple type of reinforcement learning problems where there is only one state and k available actions for the agent to choose from. Additionally, in these types of problems, the agent's actions do not influence the state of the environment. These types of problems, although not very common in the broader domain of reinforcement learning problems, do present a good way to observe the difference in performance of different action-selection and action-value-approximation methods. In reinforcement learning, action-selection methods are how the agent decides which action to do next, and action-value-approximation methods are what the agent does to update the knowledge it has of its environment and of the expected values for the available actions. In non-stationary problems, these expected values change over time. The goal of the agent is to maximize the total reward it gets from doing the available actions.

Two different types of action-selection methods are used in this assignment. The first one is a purely greedy action-selection method, as a guideline for the worst performance the agent could have. Later, epsilon-greedy action-selection methods are used. The main focus of the assignment is, however, to show the difference in performance of two action-value-estimation methods, namely incrementally computed sample averages and incrementally computed weighted averages.

Sutton and Barto [1] provide formulas for the two action-selection methods that are used in this assignment, as well as for the two different action-value-estimation methods. For greedy action-selection methods, the agent chooses the action with the highest estimated action value, according to:

$$A_t = \arg\max_x Q_t(a) \tag{1}$$

On the other hand, in epsilon-greedy methods, the action selected by the agent is determined according to:

$$A_t = \begin{cases} \text{random}\, a \in A, & \epsilon \\ \underset{a}{\text{argmax}}\ \ Q(a), & 1 - \epsilon \end{cases} \tag{2}$$

Additionally, the formula used to compute incrementally computed sample averages, used for the estimation of action-values by the agent, is the following:

$$Q_{n-1} = Q_n + \frac{1}{n}[R_n - Q_n] \tag{3}$$

The formula for incrementally computed weighted averages is a generalization of (3) , except that $\frac{1}{n}$ is replaced by α, as follows:

$$Q_{n-1} = Q_n + \alpha[R_n - Q_n] \tag{4}$$

This equation ensures that the latest rewards obtained get a larger weight when computing the average than the first ones.

In order to quantify the performance of the different methods, the rewards obtained by the agent and the agent's choice of action over multiple runs were used as guidelines.

2 Motivation

Non-stationary problems present an interesting challenge for reinforcement learning methods, as the agent needs to not only estimate the expected action values for each of the actions, but also track how these values change over time. These types of problem appear often in the real world, where the values of certain actions can drift over time due to the physical nature of the agent's environment, or simply because of error in the observations of the agent. Therefore, it is important to determine whether changing the action-value-estimation method produces better results. As discussed in [1], for most problems, the rewards that the agent obtains at the beginning of its lifetime should not remain equally relevant to the other rewards obtained in the entire lifetime of the agent, but rather the latest rewards should be more significant to the agent when deciding which action to pursue next.

3 Experiment

3.1 Hypothesis

I believe that the recency-weighed average method will perform the same as the samples average method when greedy methods are used. However, when the epsilon-greedy action-selection method is used, the recency-weighed averages will perform better than the samples averages method for all the different hyperparameters tested.

3.2 Methodology

There are two parts to the experiment. The first one is the testbed for the experiment, and the second one is the agent. The testbed of the experiment consists of k arms, and in this case $k = 10$. Each of these arms has an expected action value, and the starting action value for all of the arms is the same. When the agent "pulls" each one of these arms, it obtains a reward. The reward of each arm is determined according to a normal distribution with mean equal to the expected action value for that arm and an initial variance of 1 that is then changed to 0.1 in one of the experiments. After the agent "pulls" an arm, all the expected action values take a "random walk" by adding to them a value from a normal distribution with mean 0 and initial variance 0.01, subject to change for different experiments. This represents the expected action values for the arms drifting over time.

The second part of the experiment is the modeling of the agent and it's behavior. The agent only has access to the rewards set in the testbed, and it generates an estimated action value, $Q(a)$, for each of the arms. This $Q(a)$ is updated using either the samples average method or the recency-weighed method shown in the Background section of this assignment. The algorithm for choosing which action to take, either greedily or epsilon-greedily, is shown in Figure 1 taken from [1]. This algorithm works for recency-weighed averages as well, as mentioned in the footnote.

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Repeat forever:
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Figure 1: Algorithm for epsilon-greedy action-selection method. To get greedy method, change epsilon to be 0. To change it from a samples average to a recency average, change $\frac{1}{n}$ to α.

Finally, a benchmark function was added to interpret the results. Two quantitative results were obtained from the experiment. The first one is the average reward obtained in each time step averaged over the total number of runs of the experiment (2000 in this case). The second one is the percentage of the number of times the agent selected the optimal action that had the highest action value. In order to measure this, the agent had access to the true expected action value for each arms, but this was only used for testing purposes and not as part of the agent's learning.

In order to compare the performance of the different action-selection methods and action-value-estimation methods, several runs of the experiment with variations in the parameters were done. In the first run, a baseline experiment was run where the epsilon-greedy method was used with a reward variance of 1, $\epsilon = 0.1$, and $\alpha = 0.1$. The, the greedy method was used with a reward variance

of 1 and $\alpha = 0.1$. In the third run, the epsilon-greedy method was used with a reward variance of 1, $\epsilon = 0.1$, and $\alpha = 0.5$. Finally, in the fourth run, the epsilon-greedy method was used with a reward variance of 0.1, $\epsilon = 0.1$, and $\alpha = 0.1$.

3.3 Implementation

The experiment was done using Python 3 and two libraries: NumPy for math operations and Matplotlib to visualize results. The program was separated into two classes, one for the environment and one for the agent, each with its respective methods. Each run of the experiment consisted of the agent "pulling" the arms 10000 times according to the action-selection method used. Additionally, the program did 2000 runs in order to obtain a good approximation of the performance of the algorithm.

4 Results and interpretation

The experiments were run with different parameters of α, ϵ and reward variance in order to explore the difference in performance between different action-selection methods, action-value-estimation methods, and to explore the difference in performance in different environments. Namely, the two action-selection methods compared were greedy and epsilon-greedy methods; the two different action-value-estimation methods compared were incrementally computed sample averages and incrementally computed recency-weighted averages with different parameters α; and two types of environment were tested, one with rewards with a variance of 1, and another with rewards with a variance of 0.1.

4.1 Baseline

A baseline performance test was run in order to not only observe the differences between the two action-value-estimation methods but also to compare it to other runs of the experiment with different parameters. The results for this run can be seen in Figure 2 . They show clearly that the recency-weighted average method performs better both in average reward achieved over time as well as in choosing the optimal action a larger percentage of the time. This was expected, as the agent should have a better idea of its environment as time goes on, and therefore it should also be able to choose the optimal value more often and get higher rewards.

4.2 Greedy vs. Epsilon-Greedy

The first experiment done after the baseline experiment consisted of comparing the performance of the two different action-value-estimation methods using a greedy method to estimate the action values. Although it seemed like there was no reason for the recency-weighted averages method to perform better than the samples average method, it is clear from Figure 3 that the recency-weighted averages method performed much better than the average samples method. However, the recency-weighted averages method still performs poorly compared to the baseline performance shown in Figure 2. It gets lower average rewards and

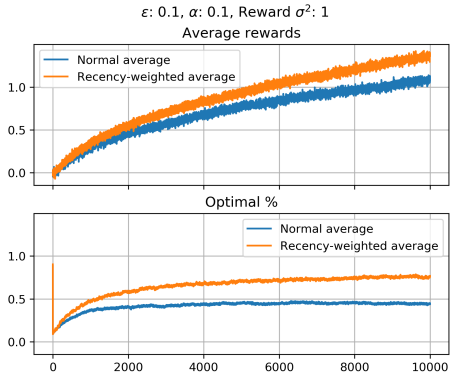

Figure 2: Baseline experiment

chooses the optimal action much less often than when the agent used epsilon-greedy. For instance, the recency-weighted averages performs similarly to the normal averages method in the baseline experiment.

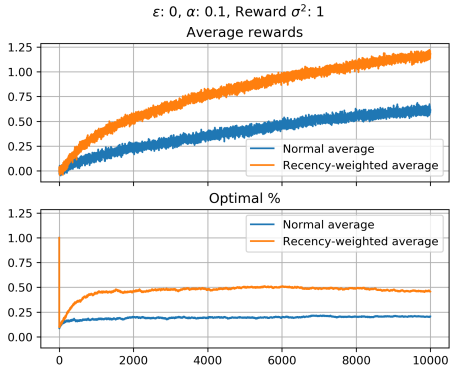

Figure 3: Comparison between greedy and non-greedy methods using incrementally computed sample averages and incrementally computed recency-weighted averages

4.3 Different values of α

One of the features of the recency-weighted averages method is that the parameter α can be tuned to add or reduce weight to the most recent values obtained. Here, two values of α were compared: $\alpha = 0.1$ (used in the baseline experiment) and $\alpha = 0.5$. It can be seen in Figure 4 that while $\alpha = 0.5$ still resulted in the recency-weighted method achieving better results than the normal averages method, the difference in performance was not very significant. Additionally, when comparing this run of the experiment with the baseline experiment, it can be seen that the lower value of $\alpha = 0.1$ used in the baseline experiment performed much better. This leads to believe that lower values of α perform better than larger values. Intuitively, this means that while it is good to give

more weight to recently obtained values, it is still important to keep a significant knowledge of the past.

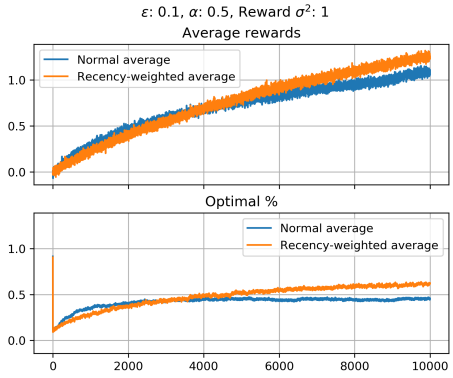

Figure 4: Comparison between different values of α

4.4 Different Reward Variances

Finally, in this experiment, two different environments were tested. Namely, the variance in the reward was modified to be 10 times lower than the reward variance in the baseline experiment. The results of this experiment can be seen in figure 5. The difference between the two action-value-estimation methods is more evident with lower reward variances. Additionally, the performance of the recency-weighted method increases, albeit not by much.

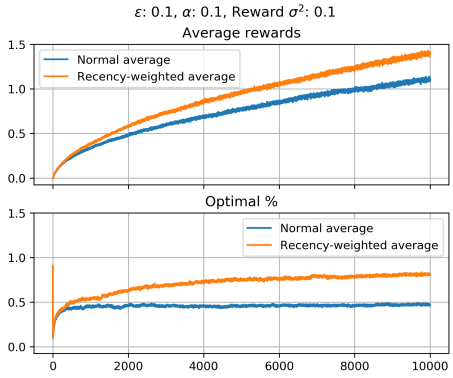

Figure 5: Comparison between different environments (reward variances)

5 Conclusion

The hypothesis that the recency-weighted averages action-value-estimation method would perform better than the samples average method in all the epsilon-greedy cases was correct. However, they also performed better in the purely greedy case

as well.This makes sense, because of the randomly changing nature of the expected action values. Since the values change randomly, they can decrease as well, meaning that the agent is more likely to pursue the actions that have given the highest reward lately instead of the highest average reward over its entire lifetime. From the experiments done here, we can safely argue that the recency-weighted averages method is better than the normal averages method for non-stationary problems. Could it be that the recency-weighted averages method performs better for stationary problems as well? The evidence presented in this assignment leads to believe that it would

References

[1] Sutton, Richard S., and Andrew G. Barto. *Reinforcement Learning: an Introduction.* 2nd ed., The MIT Press, 2018.