COMP 138 RL: Homework 1

Ramtin Hosseini

September 25, 2020

1 Introduction

In all reinforcement learning problems, we are interested into maximizing the reward function. In chapter 2 of the book [1], we investigate different methods in the context of a k-arm bandit problem to achieve this goal.
In the first section of the book, we study that the averaging methods for defining the action-value functions are appropriate for stationary bandit problems, that is, for bandit problems in which the reward probabilities do not change over time.
In the programming assignment 2.5 we would like to compare and contrast another method called recency weighted average with sample average in a non-stationary problem. The motivation behind using recency weighted average method is that in a reinforcement learning setting, we often encounter problems which are effectively non-stationary. In such cases it makes sense to give more weight to recent rewards than to long-past rewards.

2 Learning Methodology

In the following subsections we investigate both the sample-average and receceny weighted average methods.

2.1 Sample-average

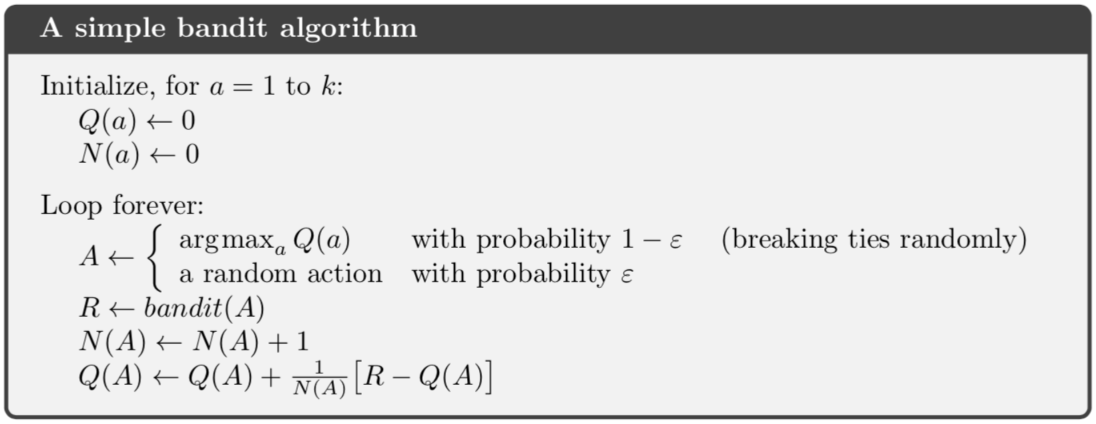
In the sample-average method, the action-value estimate is considered as averages of observed rewards. To compute these averages in long turn which is computationally prohibitive, we need to find a way to efficiently calculate it. Finding a recursive algorithm to use dynamic programming would be very helpful. To do so, we devise incremental formula for updating averages with small, constant computation. Thus we get the following expression:

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) = Q_n + \frac{1}{n}[R_n - Q_n]$$

The general form is:

NewEstimate $\leftarrow$ *OldEstimate + StepSize [Target - OldEstiamte]*

The pseudocode for a bandit algorithm using incrementally computed sample averages and ϵ-greedy action selection is shown in figure 1. The function *bandit(a)* is assumed to take an action and return a corresponding reward.



A simple bandit algorithm

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

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

Figure 1: Pseudocode for a complete sample-average bandit algorithm

2.2 Recency Weighted Average

As we mentioned earlier, we usually encounter non-stationary problems in reinforcement learning. In such cases it is better to give more weight to recent rewards than to rewards along time ago. This way we encourage the learning agent to adapt to the recent changes (e.g in the environment) which is reflected in the rewards. One of the most popular ways of doing this is to use a constant step-size parameter. For example, the incremental update rule for updating an average Q_n of the $n-1$ past rewards is modified to be

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

where the step-size parameter $\alpha = (0, 1]$ is constant. This results in Q_{n+1} being a weighted average of past rewards and the initial estimate Q_1:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n] = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

This is called a weighted average because the sum of the weights is equal to 1. Let $\alpha_n(a)$ denote the step-size parameter used to process the reward received after the nth selection of action a. A well-known result in stochastic approximation theory gives us the conditions required to assure convergence with probability 1:

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \text{ and } \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty.$$

The first condition is required to guarantee that the steps are large enough to eventually overcome any initial conditions or random fluctuations. The second condition guarantees that eventually the steps become small enough to assure convergence.

3 Experimental Results

We conduct experiments for both recency weighted average and the sample average methods on a 10-armed non-stationary bandit problem for 10000 steps and 2000 iterations. Figure 2 shows the comparison between the recency weighted average and sample-average ϵ-greedy action-value methods. The experiments demonstrate that recency weighted average outperforms the sample average method in the non-stationary setting. The reason is that by considering higher weight for the recent rewards we encourage the agent to adapt itself better to the latest environment dynamics.

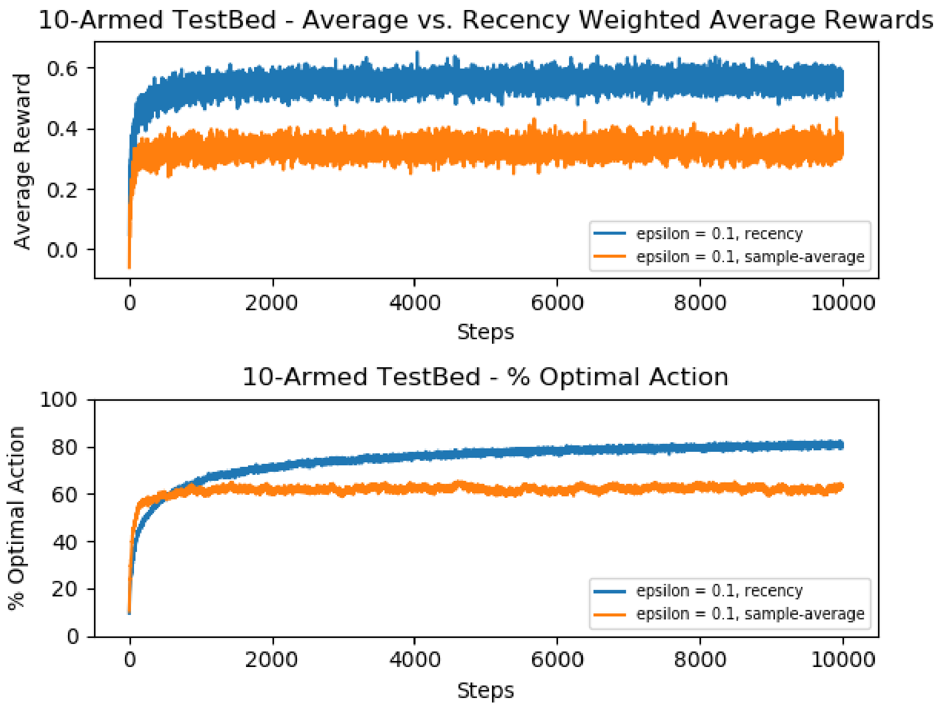


Figure 2: 10-Armed TestBed - Average vs. Recency Weighted Average Rewards

4 Extra Credit

In the following subsections, I did three additional exercises. The last one is a programming exercise.

4.1 Extra Credit 1: Exercise 2.4

$$
\begin{aligned}
Q_{n+1} = Q_n + \alpha[R_n - Q_n] \\
= (1-\alpha_n)Q_n + \alpha_n R_n \\
= (1-\alpha_n)((1-\alpha_{n-1})Q_{n-1} + \alpha_{n-1}R_{n-1}) + \alpha_n R_n \\
= (1-\alpha_n)((1-\alpha_{n-1})Q_{n-1} + (1-\alpha_n)\alpha_{n-1}R_{n-1} + \alpha_n R_n \\
= Q_1 \prod_i^n (1-\alpha_i) + \sum_i^n (\alpha_i R_i \prod_{j=i}^{n-1} (1-\alpha_j))
\end{aligned}
$$

They look very similar. In fact the expression in section 2.5 is a special case of this general form.

4.2 Extra Credit 2: Exercise 2.6 - Mysterious Spikes

At the beginning, most agents go through all actions once and be disappointed in all the rewards they receive. Afterwards in their next move, depending on the rewards they obtain, they have a high chance of selecting the optimal action. Because many agents select the optimal action at that time step we observe a spike in the graph. However, after having selected it, most agents will have been disappointed, and begin selecting other actions again.

4.3 Extra Credit 3: Programming Exercise 2.11

In this subsection, we investigate the effect of the action value initialization in a non-stationary ϵ-greedy bandit problem by using weighted average reward algorithm. Figure 3 depicts this effect when the algorithm has been run for 2000 iterations and 10000 steps. Note, based on the spec, for plotting this figure, the reward in the steps in the second half part is considered.

5 Conclusion

In this exercise we consider a non-stationary bandit problem as our case study and investigate the performance of two methods. 1. sample-average reward, 2. recency weighted reward. The experiments show that in long turn, recency outperforms. The reason is that by giving more weights to the recent actions, the agent adapts itself to the recent dynamic in the environment.

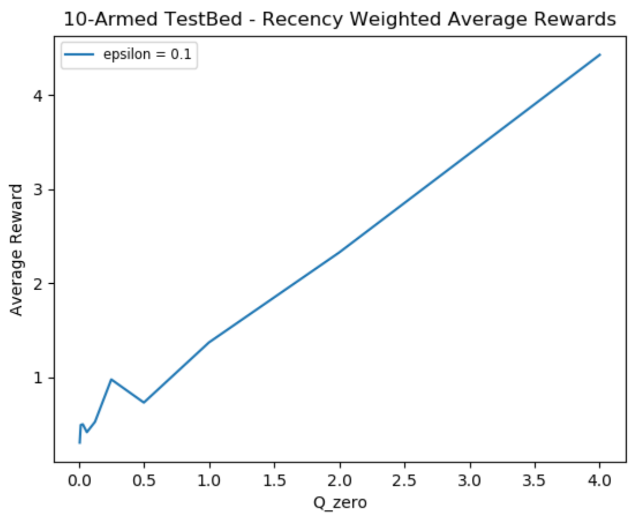


Figure 3: Average rewards obtained in a 10-armed bandit problem based on different action values estimates' initialization in 10000 steps

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.