COMP 138 RL: Homework 1

Nanxi Liu

February 14, 2025

1 Introduction

In reinforcement learning, two of the most popular action-value methods are calculating the sample averages of learned rewards for stationary problems (meaning the reward probabilities don't change over time) and calculating the rewards using a step-size parameter for non-stationary problems (some parameters of the environment is changing). Action-value methods are used by agents to estimate the reward of taking a certain action in a given state, and it is crucial to the success of an agent's learning process. For the sample-average method, the value of each action is calculated by obtaining the average of all the past rewards gained by that particular action.[1] During implementation, this is done by adding to the previous reward of the action the result of the actual reward of the action subtracting the learned reward of the action and dividing by the number of times this action has been taken. The step method, on the other hand, keeps updating the learned reward by adding the product of a step-size parameter and the result of subtracting the previous learned reward from the actual reward of the action. [1]

2 Goal

The goal of this research is to first compare the performance of the two methods in a 10-armed non-stationary problem, then look into how having an increasing number of arms affect their performances, thus gaining more information about the applications of these methods.

3 Problem Description

The agents are trained to learn the best option in a k-armed bandit problem. The agent is given k options to choose from, and at each time step, the agent can only choose to take one action. The ultimate goal of the agent is to maximize the expected total reward over the training time period. Each of the k actions has an expected mean reward, which is called the value of the action, that the agent is not aware of at first, and learns gradually during its exploration. However,

the agent will never know the exact value of the rewards; instead, it estimates the reward of each action by keeping track of the learned rewards at each time step. The agent will also need to decide between exploration and exploitation. Exploration refers to the agent's willingness to try other actions even though there might be a known optimal action in its records of reward estimates. Exploitation refers to the agent taking the current optimal action in its reward estimates. While exploitation allows the agent to maximize its rewards in the short-term, exploration offers the agent more opportunities to discover new actions that will potentially bring more rewards. The agent's learning result will be optimized when exploration and exploitation is in a balanced state.

4 Experiment Design

This experiment is run on multiple sets of parameters to test out different hypotheses. An agent is designed as a Python class, and takes in several parameters - the number of arms, the value of the epsilon variable that decides the extent of exploration, a seed that allows users to recreate a random initiation of the rewards, the number of epochs to train, a step-size parameter if an agent uses the step method, and also a variable called 'method' that informs the system what type of action-value methods we wish to use. In this study, each agent is trained in a 10000 time-step session with the exception of the 100-arm bandit agent that was trained both 10000 and 20000 times for better comparison between the two action-value methods. For each set of parameters, its corresponding experiment will be run for 900 times, whose average will then be used to plot graphs for observation. All sets of parameters are tested on both action value methods using epsilon = 0.1 and step-size = 0.1 (if needed). To mimic a non-stationary environment, the rewards of all actions are increased by a normally distributed increment with mean 0 and standard deviation 0.01 at each time step.

5 Hypothesis

My first hypothesis is that the step-size method will outperform the average method in all experiments due to the non-stationary nature of the environment. My second hypothesis is that it will take longer for the proportion of the optimal actions of the agents to converge if dealing with more arms.

6 Results

The result of the experiment is mostly the same as predicted in the hypothesis, with the step-size methods out performing the average methods in all experiments. However, the more arms there are, the longer it takes for the step-size methods to outperform the average methods. The rewards are calculated by first gathering a list of rewards learned in each experiments, then the list is averaged

to produce a single list of average reward. Subsequently, the list is convolved using a 100 time-step moving window through the numpy library to produce a smoother output. The proportion of optimal action overtime is calculated using the same method. However, when keeping track of optimal actions, each action the agent takes is compared to the optimal action using the actual reward list, then recorded "1" if the action is an optimal action, or "0" if the action is not.

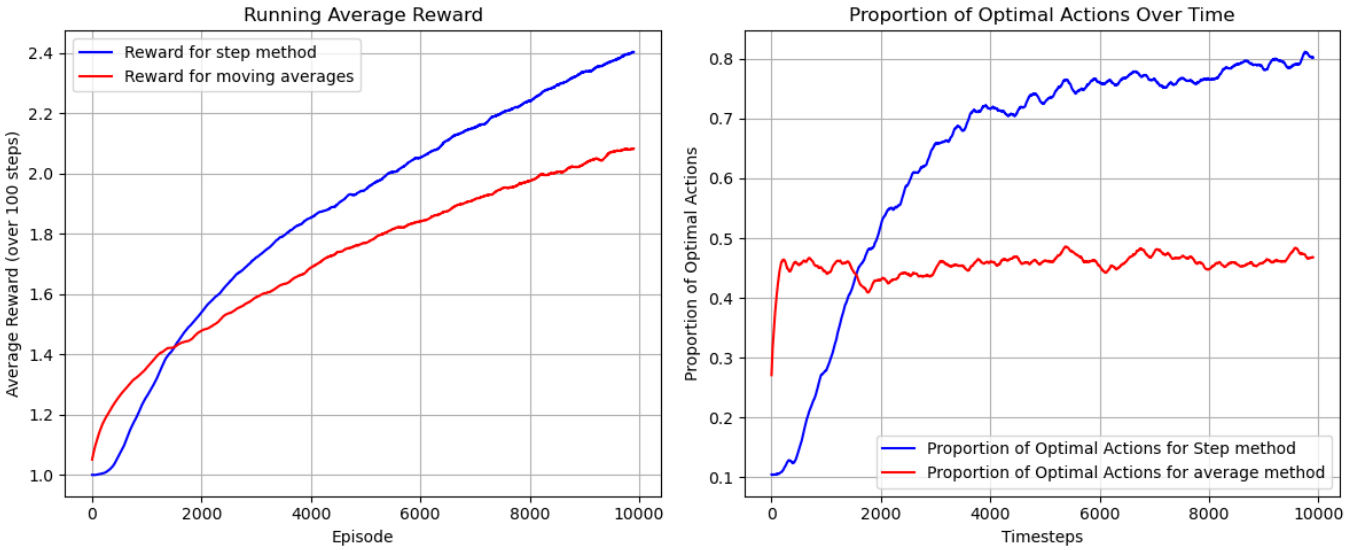

Figure 1: Average reward and optimal actions for the 10-armed bandit problem

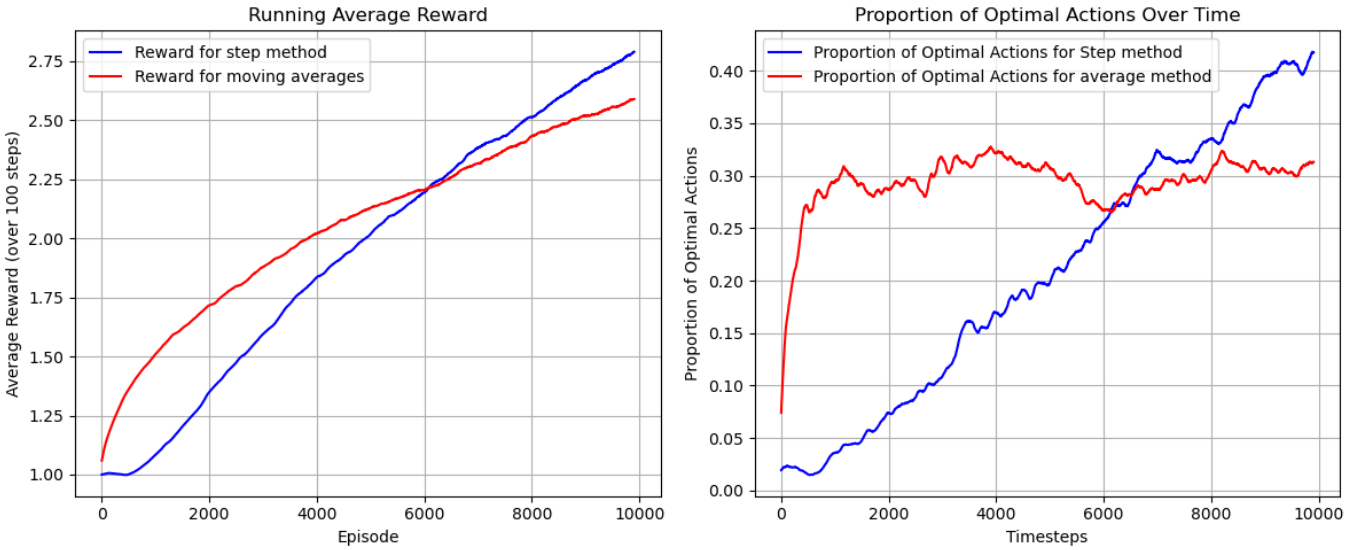

Figure 2: Average reward and optimal actions for the 50-armed bandit problem

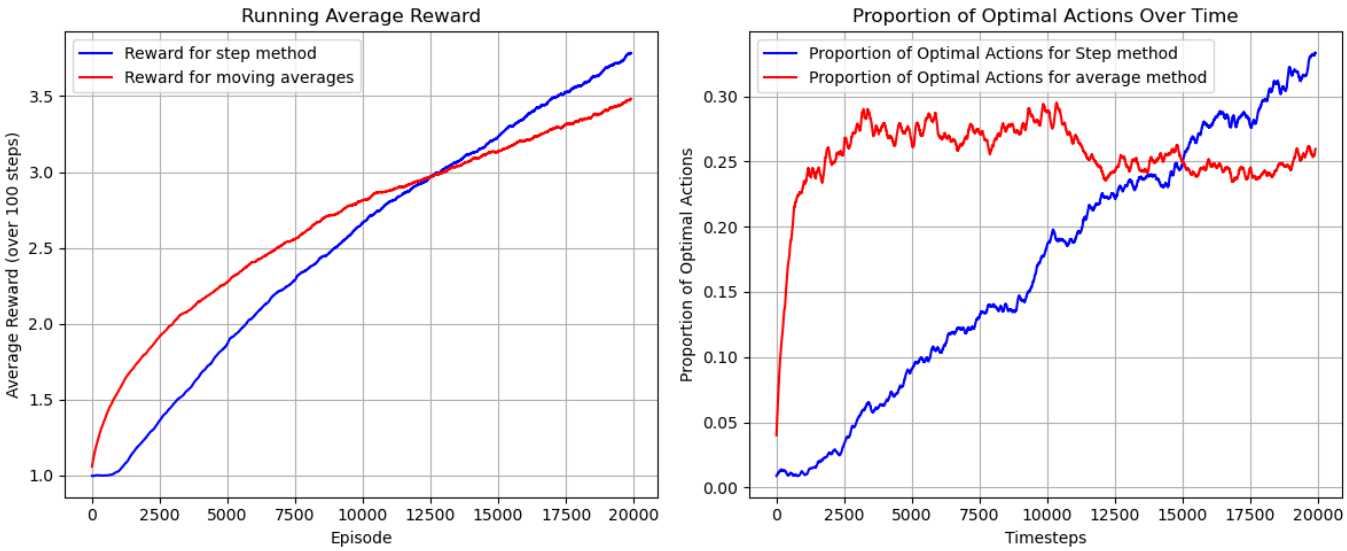

Figure 3: Average reward and optimal actions for the 100-armed bandit problem

In the 10-arm bandit problem (Figure 1), it takes the step-size method less than 2000 time steps to surpass the average method in average reward and average optimal action; in the 50-arm bandit problem (Figure 2), it takes the step-size method around 6000 time steps; finally in the 100-arm bandit problem (Figure 3), it takes the step-size method around 12500 time steps.

7 Discussion

The author is curious about why it takes the step-size method longer to surpass the average method when facing increasing dimensionality in the arms. This is probably due to the nature of how both methods calculate their learned values. While both start out slow due to lack of knowledge of the actual rewards, the step-size method is more volatile in the sense that it depends on more recently learned rewards while the average method learns the rewards more stably by keeping track of all the past rewards. This could potentially lead to more learning time for the step-size methods, and the more arms there are, the longer it takes for the agent to learn

8 Conclusion

This paper discussed the properties of the step-size and the average action-value method in the applications of a reinforcement learning agent solving a non-stationary k-armed bandit problem. The step-size method has proven to

perform better in all experiments, albeit taking longer to surpass the average method in higher dimensional problems.

References

[1] Andrew B Richard S. *Reinforcement Learning, An Introduction*. MIT Press, 2018.

9 Appendix

See the Python script and instructions on how to replicate the experiment using the following link: `https://github.com/Nanxi-Flaneuse/Reinforcement-Learning/tree/main/assignments/HW1`