Reinforcement Learning Assignment 1

Thomas Klimek

September 18, 2018

1 Introduction

This paper analyzes the performance of simple reinforcement learning algorithms on a non-stationary k arm bandit problem with 10 arms. The k arm bandit problem is an analogy to a slot machine with k arms, or actions, each rewarding a random number from a different distribution. The goal is to maximize the reward over a large amount of trials. The bandit is non-stationary, meaning the true values of the actions change over time. This paper considers the performance of epsilon greedy algorithms using a sample average with a varying step size, and a sample average with a fixed step size also know as exponential recency-weight average.

2 Problem Statement

We will be responding to **Exercise 2.5 (programming)** from *Suton & Barto*. Summarized, this problem states to design and conduct an experiment comparing two sample-average methods for a nonstationary problem. We use a 10-armed testbed in which all the $q(a)$ start out equal and then take independent random walks by adding a normally distributed increment with mean zero and standard deviation 0.01 on each step. The two action-value methods are using sample averages, one incrementally computed, and one with a constant step-size parameter, $\alpha = 0.1$, each with learning rate $\epsilon = 0.1$. Finally a plot is required of the performance of each method over 10,000 steps.

3 Methodology

3.1 Bandit

The bandit is designed to receive an initial value, chosen randomly from a Gaussian distribution centered at 0 with a standard deviation of 0.1. All arms of the bandit are initialized to this value to start. Then with each action, the true values of the arms are incremented randomly from another Gaussian distribution with a standard deviation of 0.01. This is how the random-walk occurs, and what makes the problem non-stationary. The true values are reset between each trial back to the initial value.

3.2 Agent

Two agents are designed to apply epsilon greedy algorithms using a sample-average with varying and fixed step size. The algorithms follow from the "simple bandit algorithm" found on page 24 of *Suton & Barto*. The average rewards and action counts are reset between trials. The algorithms are implemented with exploration rates of 0.1, however I change the variables in the results to better compare the performance of the two algorithms.

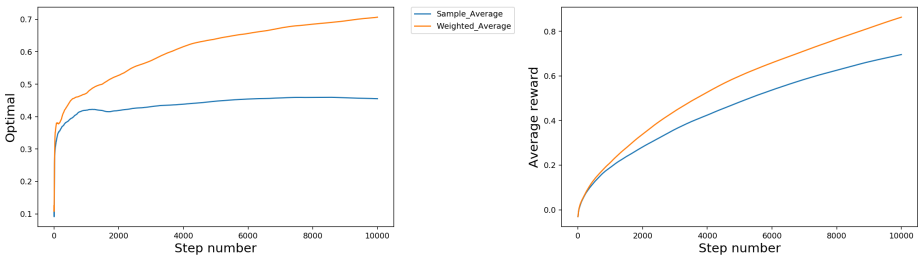
3.3 Data & Plots

Lastly, I designed a data class to keep track of optimal action percentage, and average reward for each algorithm over a large number of trials. These values are calculated using an incremental average at each trial, and averaged over the trials to reveal trends in the performance of the algorithms. For this paper we will analyze 200 trails of 10,0000 steps with the same bandit, taking independent random walks each trial.

4 Results

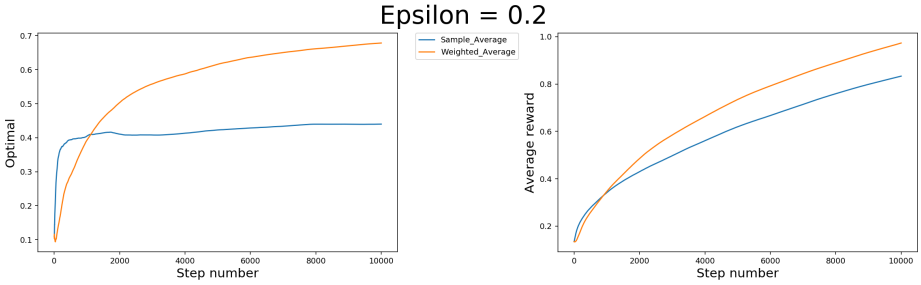
4.1 Initial Experiment

The initial analysis reveals that the weighted averages already begin to outperform the standard averages before 2,000 steps, and overtime reach an optimal percentage of 70%, compared to 40% reached from the incremental average.



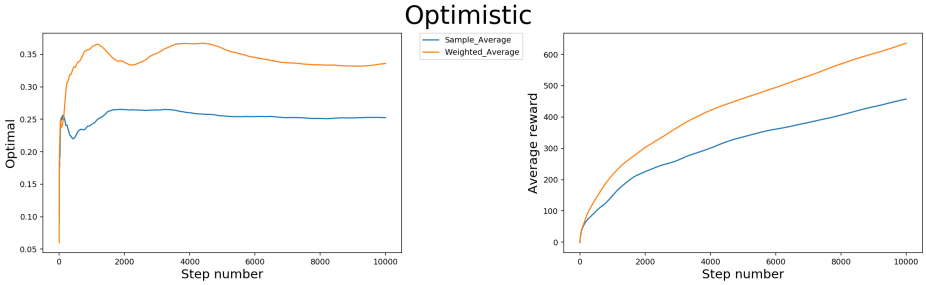
4.2 Changing Epsilon

Now lets take a look at how varying the exploration rate affects the comparison between these two algorithms. Specifically, let's see the effects of increasing the exploration rate to $\varepsilon = 0.2$ to see if more exploration will help our naive incremental average algorithm.



We can see the static step size still outperforms incremental step size. With a larger exploration rate, the incremental step size will initially beat the static step size due to the the weights updating more responsively short-term with a standard average. However as the arms take there random walks, the incremental algorithm is not able to account for the non-stationary values and is eventually beat by the static step size.

I also wanted to see what effects decreasing the epsilon might have, so I simulated a greedy algorithm with no exploration, and optimistic initial estimates.

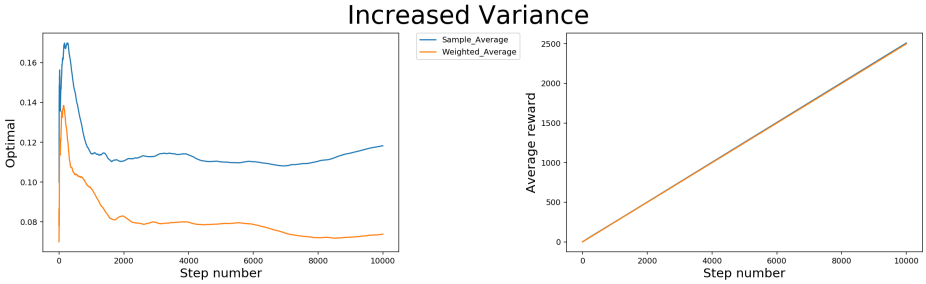


As we can see the static step size algorithm continues to outperform the incremental step size to this point. It is also worth noticing that the optimistic and greedy algorithms did not do nearly as well as the exploration based ones. For stationary problems, the optimistic algorithm may be a good choice, however even with exponential recency weighting, these algorithms are not nearly as effective ones that utilize epsilon exploration.

4.3 Changing Variance

The next idea I had was to increase the variance of the random walks to make the problem even less stationary. Instead of a random walk coming from a standard distribution centered at 0 with standard deviation 0.01 as suggested in the book, we take a random walk coming centered at a new

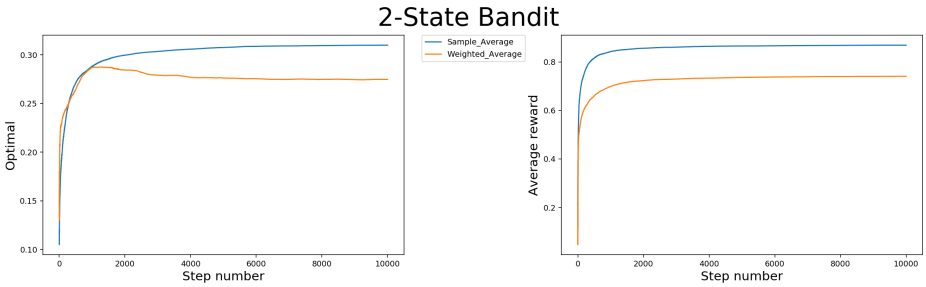
random number between 0 and 1, with a random standard deviation, each changing each time. This is an experiment to track a large amount of randomness in each arm.



These results show that neither algorithm as implemented does a great job in tracking this much randomness. Despite that in terms of optimal percentages the incremental algorithm appears higher, in terms of average reward these algorithms perform more or less the same.

4.4 2-State Bandit

The last experiment I devised is a simple 2 state bandit problem. Instead of using a random walk, we reverse the order of the array of rewards between pulls. Thus there are two states for this bandit, an initial state, and a reversed state.



From the figure we can see that over time the incremental step average outperforms the static step average. This is because the recency of an observed result does not affect the predicted value, so using an exponential recency average has no benefit. We also see that neither algorithm functioned very well in this condition. I used this example to show how bandit algorithms are not effective in tracking reinforcement learning problems that are state dependent. If the states had been utilized, we could have easily learned this problem using two bandits, however relying on one bandit is an insufficient method to learn this problem.

5 Conclusions

In summary this paper compares two algorithms for the bandit problem, one of which takes into account recency, and the other ignores it. We specifically analyze non-stationary bandits and see that as the distributions take random walks, the recency tracking algorithm significantly improves performances. We also see that these two algorithms fail to learn effectively in certain environments. This shows us that the bandit problem has limited utility in solving more complicated reinforcement learning problems. It also demonstrates the importance of thoroughly crafting an algorithm to perform for its use scenario. The recency weighted average performed well for the example given in the book, however in general it did not always beat the regular average method, and both of these performed poorly in a 2-state problem.