Comparing Action-Value Methods for Non-Stationary Reinforcement Learning Problems

Cooper Golemme

February 19, 2025

1 Motivation

Reinforcement learning distinguishes itself from other forms of machine learning by evaluating actions taken by the machine, sometimes called the *agent*, to create insights to inform what the machine should do in the future rather than explicitly instructing the machine on what actions to take based on training data. One of the crucial elements of a reinforcement learning framework is its *action-value* method – a method for estimating the reward received for performing a specific action in a particular scenario. A well calibrated action value method informs the machine on what action to do given its experience performing past actions.

In stationary problems, the environment the agent is in is unchanging. In these situations, it would make sense for the agent, in a given position, to perform the action that has yielded the best results in past. This makes sense intuitively. There is nothing changing about the environment, so eventually, the agent will learn everything there is to learn and choose the optimal action in every situation. Thus, mathematically, a simple sample average action-value method can be shown to converge in probability to the optimal solution.

In non-stationary problems, however, the environment the agent is in is dynamic. In these situations, some more thought is required for constructing a good action-value method. A sample average method will no longer suffice as new information is constantly being added to the environment. In this paper, we will demonstrate the shortcomings of using a sample average method for non-stationary machine learning problems, offer a better method using a constant step-size parameter, and offer an even better method that is both unbiased to its initial guess for the environment dynamics and behaves like a constant step-size method.

2 Introduction

To begin, we will outline the experimental setup and context in which we will evaluate various action value methods. The problem we will introduce is called the k-armed bandit problem.

2.1 Context

The k-armed bandit problem is a classic problem in reinforcement learning.[1] Its formulation goes something like this. Consider an environment where an agent is repeatedly faced with choosing among k different options. After each choice, the agent receives a numerical reward chosen from a probability distribution that depends on the action that the agent selected. The objective of the problem is to maximize the expected total reward over some period of time, 1000 action selections for example.

Ideally, the agent should learn which actions are the "best" to maximize its reward. The agent should then exploit this action, using it frequently, to achieve the greatest reward. But, the agent should balance exploitation and exploration. Just because the agent gets a very high reward from a specific action does not mean that it is necessarily the best. In such a case, the agent should try other actions, exploring other options before determining which action is the best an exploiting it. In order to determine which actions are the best, the agent needs some way of estimating the expected reward associated with performing an action. We call the estimation of the reward the agent's action value method.

In this paper, we will call this problem, the *stationary k-armed bandit problem* because the reward associated with each action comes from a stationary probability distribution. This means the expected reward for each action is stationary; that $q_*(a)$ in the equation below is a single, pre-defined value.

$$q_*(a) = \mathbb{E}(R_t | A_t = a)$$

A more interesting problem and one that pertains more broadly to the field of reinforcement learning comes when $q_*(a)$ is not stationary. In the field, this type of problem is called a *non-stationary* problem. Non-stationary problems occur much more often in real life than stationary ones. Agents designed to predict financial markets have to deal with constantly changing, non-stationary rewards, for example. Often times the environment we are in is changing and we should adjust our learning accordingly.

2.2 Problem

Non-stationary problems pose an increased degree of difficultly as we will discuss in this paper. The main problem is that traditional sample-average methods for estimating the rewards for a given action assume stationarity. They fail in subtle ways when $q_*(a)$ changes over time.

2.3 Objectives

In this paper, we want to demonstrate improvements that can be made to the sample average action-value method in non-stationary settings. We will do this by comparing sample average methods, constant step-size methods and an unbiased constant step-size method in a controlled non-stationary setting.

3 Experimental design

3.1 Environment

The environment in which we will run the experiment to test the sample average method and constant step-size for calculating the estimate for the reward function is a non-stationary 10-armed bandit problem.

Starting with some notation, $q_*(a)$, is the expected reward given that action a is selected. The actual reward received at a given time step t, denoted R_t, depends on the action selected at that time step A_t and is sampled from a normal distribution with mean $q_*(A_t)$ and variance 1.

It is important to note what here is deterministic and what is not. The reward, R_t, not deterministic as it is sampled from a normal distribution, given an action selected at time step t, A_t. The $q_*(a)$, under the current construction is deterministic. It is a fixed value which acts expected reward for the underlying stochastic reward function. A deterministic q_* like this, makes this a stationary problem; the optimal action, defined as the action which yields the best expected reward does not change.

In the non-stationary case, q_* is not deterministic. In this paper, to simulate non-stationarity, each $q_*(a)$ takes a random walk achieved by adding a small step ϵ_t to the value $q_*(a)$ at each time step t. This step, ϵ_t, is generated by sampling from a normal distribution, with mean 0 and variance 0.01. This has the effect of potentially changing the optimal action by a small amount each time we select an action.

3.2 Action Value Methods and Algorithm

Now that we have outlined the environment, we will introduce the methods used to estimate the underlying reward function.

We denote our estimate, Q_n, with $Q_n(A)$ being our best guess for the reward for picking action A after n time steps. To allow for exploration, instead of always choosing the best action according to our action value estimate, $Q_n(a)$, we choose the best action with $1 - \epsilon$ probability and randomly choose an action from the other actions with ϵ probability, where ϵ is a tuneable parameter that is greater than 0 and less than 1.

This strategy of choosing an action is called an ϵ-greedy approach. This approach allows for exploration at a tunable rate so that we can explore all possible actions to find the best ones rather than finding one that may be suboptimal and exploiting it.

We will now show the three ways we will estimate $Q_n(a)$ in this paper.

3.2.1 Sample Average

The sample average method calculates Q_n by taking the average reward we observed over the past $n-1$ time steps for choosing a given action. Formulaically, Q_{n+1} is

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i$$

where R_i is the reward we got at the ith time step.

With some effort, it can be shown that we do not need to keep track of all previous rewards in order to update our estimate. We need to only know the previous reward and the current time step. Q_{n+1} can be represented as

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

where Q_n is our previous estimate and R_n is the previous reward received. We will use this formula to update our estimate for the reward function.

3.2.2 Constant Step-size

The other method that we will compare involves using a constant step-size rather than a sample average to estimate the rewards. Rather than each previous reward being weighed equally, as they are in the sample average method, this method gives more weight to recent rewards than to long-past rewards. Theoretically, giving more weight to recent rewards should perform better in non-stationary settings where recent rewards are a better indicator for the true underlying reward function.

We can modify the incremental update rule given as (1) to use this step-size parameter instead of the average by replacing the $\frac{1}{n}$ term with a positive constant, α, as follows

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

Again, with some effort, we can give an explicit formula which results in Q_{n+1} being a weighted average of past rewards and the initial estimate Q_1:

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i \tag{2}$$

3.2.3 Unbiased Constant Step-size

The third method we will use to estimate the reward function will be an unbiased constant step-size method. If we examine Equations 1 and 2, we notice that both equations depend on the original estimate for the reward function, Q_1. Thus, both methods are biased as to our original estimate for Q_1, which is chosen a priori. The bias in the estimate leads to arbitrarily selected initial estimates performing better than other estimates. Ideally, we would want an unbiased estimator to achieve the optimal results.

With some subtle changes, we can modify our equation for the constant step-size estimate to achieve an unbiased, constant step-size estimate for the reward function. We do this by selecting a step size, β_n to process the n-th reward like

$$\beta_n = \frac{\alpha}{\bar{o}_n}$$

where $\alpha > 0$ is the constant step size parameter and $\bar{o}_n$ is defined as

$$\bar{o}_n := \bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1}) \quad \text{for } n > 0 \quad \text{and} \quad \bar{o}_0 = 0$$

We then can derive an update rule similar to Equation 1 and Equation 2 , which is done in Section 4.3 of this paper. We get

$$Q_{n+1} = \sum_{k=1}^{n} \beta_k R_k \prod_{i=k+1}^{n} (1 - \beta_i) \tag{3}$$

We can see that this equation gives the same exponential weights to more recent rewards like in Equation 2, but without the Q_1 term which gives Equation 2 its bias. Theoretically, this method should perform better when we do not a prior guess the starting reward value that then takes a random walk. We will verify this through experiments.

4 Experiment Results

4.1 Experiment 1: Sample Average vs. Constant Step-Size Method

For our first experiment, we implemented a non-stationary, 10-armed bandit environment. The rewards for each step are determined by sampling a normal distribution with mean 0 and variance 1. At each step, each mean take a random walk by adding a small value sampled from a normal distribution with mean 0 and variance 0.01.

The results of the two action value method's performance in this environment are shown below.

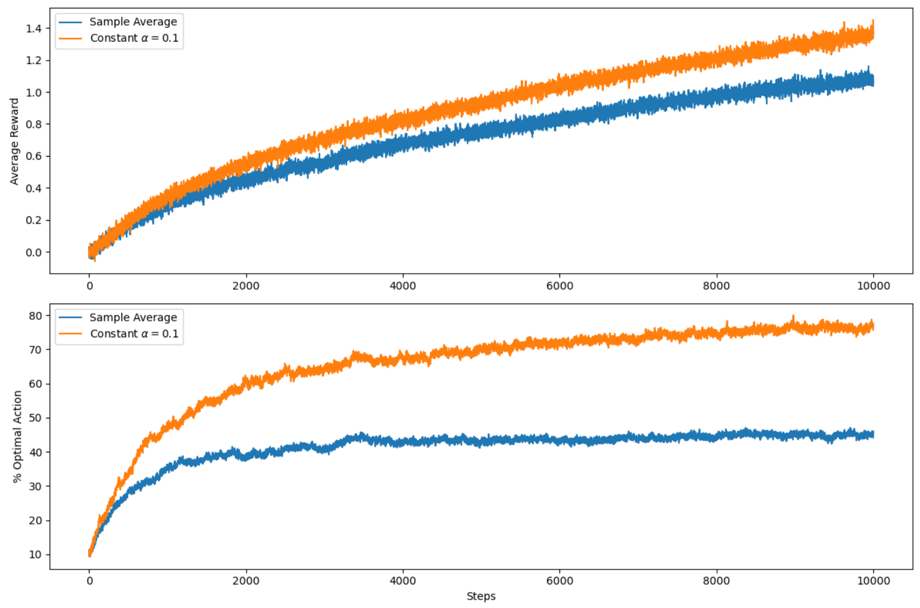

Figure 1: Average reward and percent optimal action for both the sample average and constant step-size methods of action value functions. The constant step size method appears to be a better way to estimate the reward function in this non-stationary context

4.2 Experiment 2: Bias in the Constant Step-Size Method

For the next experiment, we wanted to test the effects of the bias of the constant step-size method compared to the unbiased sample average method. Instead of starting each reward function at mean 0, we started each at 10, still performing the same random walks as before. In this case, the constant step-size method still guessed 0 for all actions, simulating not knowing the scale of the underlying distribution a priori. Here are the results:

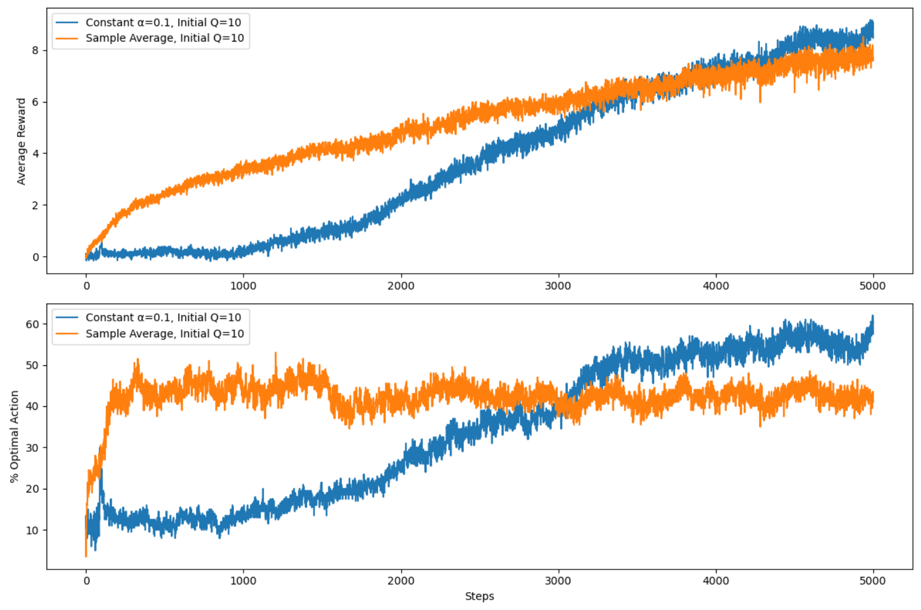

Figure 2: Comparing sample average method and constant step-size method for tracking non-stationary problems. This case highlights the effect of the bias term for the constant step-size method. When we do not guess starting values of the random walk, it takes much longer for the constant step-size method to perform better than the sample average method.

From this experiment, we can see the issues that come with biased estimators; it takes a long time for the agent to overcome the bias it brings to the environment. While the sample average method is unbiased, it isn't optimal when we know the underlying conditions of the environment, as shown in Figure 1.

4.3 Experiment 3: Unbiased Constant Step-Size vs. Biased Constant Step-Size vs. Sample Average

In Section 3.2.3, we outlined a procedure that would yield an unbiased, constant step-size method for estimating the reward function. This ideally would perform better than the biased estimate initially and outperform the sample average method quicker than the biased estimate as well because the unbiased estimate does not need to "overcome" the bias. When we introduced this agent that uses the unbiased constant step-size method into this environment, we notice exactly this.

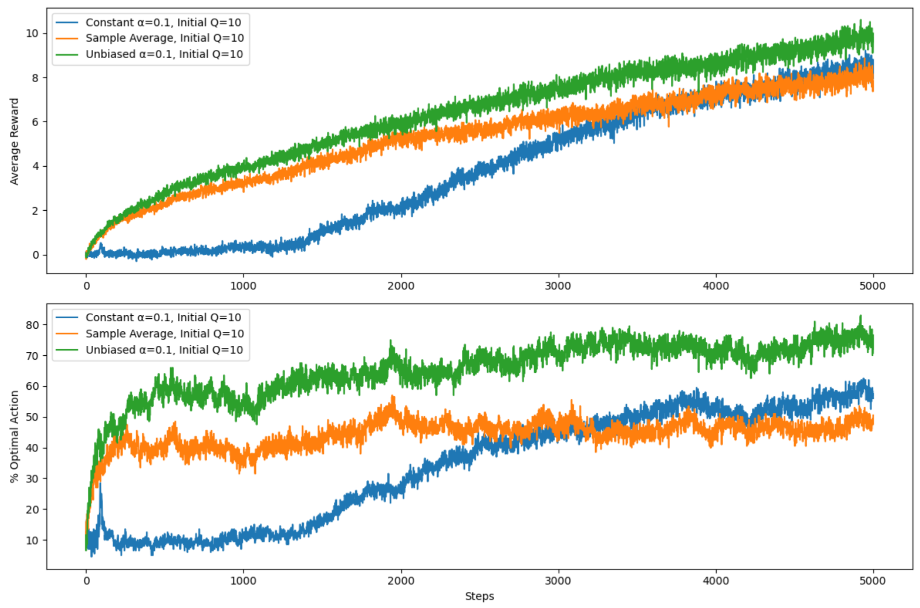

Figure 3: Comparing the biased constant step-size, unbiased constant step-size, and sample average methods for action value functions in our non-stationary environment. We find that the unbiased constant step-size method performs the best overall. It eliminates the slow adaptation problem of the sample average method and avoid the bias problem of the constant step-size method

5 Conclusion

In this paper, we investigated different action-value methods for reinforcement learning in non-stationary environments, specifically within the context of the k-armed bandit problem. We compared the performance of the sample average method, the constant step-size method, and the unbiased constant step-size method through controlled experiments.

The results of the paper clearly demonstrate that the sample average method is not suited for non-stationary settings because it weighs all past rewards equally, making it slow to adapt in dynamic environments. The constant step-size method, gives exponentially more weight to recent rewards which is better suited to dynamic environments and outperforms the sample average method in non-stationary environments. One problem it has, however, is that is dependent on an initial action value estimate and therefore has bias, which leads it to perform sub-optimally in early stages.

To address this issue, we introduced the unbiased constant step-size method, which removes the dependence on the initial estimate, through some clever math, while still prioritizing recent rewards. Through experiments, we confirmed that this method achieves faster adaption and better long term performance compared to both the sample average and biased constant step-size methods.

These findings highlight the importance of learning strategies in reinforcement learning, particularly in non-stationary contexts where the reward distribution changes over time. These principles extend past this paper and into real world applications. More thought could be put in to these methods to adapt to multi-agent environments or more complex environments like those in contextual bandit problems. Nonetheless, the principles of this paper display crucial elements of action value estimation in reinforcement learning.

6 Extra Credit

In this paper, when we investigated the non-stationary k-armed bandit problem, we used a constant step-size parameter, and updated our estimate for the reward for the action, $Q_n(A)$, using the following formula:

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i \tag{4}$$

This equation holds for a constant step size parameter $\alpha \in (0,1]$. Exercise 2.4 in the book challenges the reader to derive the formula for updating the estimates for an arbitrary sequence of step parameters a_n that are not constant. This formula is the useful as it applies more broadly to all sequences of step-size parameters (α_n) and not just constant ones.

6.1 Estimate for the Reward Function with Non-constant Step-sizes

We can start this derivation by noting that an arbitrary step Q_{n+1} can be expressed in terms of the previous step plus the difference between the reward coming from the previous step and the estimate times some constant.

$$Q_{n+1} = Q_n + \alpha_n[R_n - Q_n] \tag{5}$$

From here, we can continue expressing Q_n in terms of the rewards, estimates and step sizes of the previous step, to eventually arrive at our derivation.

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha_n[R_n - Q_n] && (6)\\
&= \alpha_n R_n + (1-\alpha_n)Q_n && (7)\\
&= \alpha_n R_n + (1-\alpha_n)(\alpha_{n-1}R_{n-1} + (1-\alpha_{n-1})Q_{n-1}) && (8)\\
&\{\text{Substituting } Q_n \text{ using (5)}\}\\
&= \alpha_n R_n + (1-\alpha_n)(\alpha_{n-1})R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})Q_{n-1} && (9)\\
&= \alpha_n R_n + (1-\alpha_n)(\alpha_{n-1})R_{n-1} && (10)\\
&+ (1-\alpha_n)(1-\alpha_{n-1})(\alpha_{n-2}R_{n-2} + (1-\alpha_{n-2})Q_{n-2})\\
&= \alpha_n R_n + (1-\alpha_n)(\alpha_{n-1})R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})(\alpha_{n-2}R_{n-2}) && (11)\\
&+ (1-\alpha_n)(1-\alpha_{n-1})(1-\alpha_{n-2})Q_{n-2}) + \cdots +\\
&(1-\alpha_n)(1-\alpha_{n-2})\ldots(1-\alpha_2)(\alpha_1)R_1 +\\
&(1-\alpha_n)(1-\alpha_{n-1})\ldots(1-\alpha_1)Q_1
\end{aligned}
$$

Continuing this pattern down to Q_1, we notice that there is a long chain of weights multiplied together. The coefficient of the Q_1 term is a product of $n-1$ terms. Each term is the $1-\alpha$ weight associated with each step.

The other terms can be expressed as the sum of the rewards at each step multiplied by a coefficient that is the product of the weights of each future step.

Thus our final derivation looks like:

$$
\begin{aligned}
Q_{n+1} &= (1-\alpha_n)Q_n + \alpha_n R_n\\
&= \underbrace{(1-\alpha_n)(1-\alpha_{n-1})\cdots(1-\alpha_1)Q_1}_{\text{Weight on initial estimate}} + \underbrace{\sum_{k=1}^{n} \alpha_k (1-\alpha_n)\cdots(1-\alpha_{k+1})R_k}_{\text{Weighted sum of rewards}}.
\end{aligned}
$$

And we get the following equation:

$$
Q_{n+1} = Q_1 \prod_{i=1}^{n}(1-\alpha_i) + \sum_{k=1}^{n}\left[\alpha_k R_k \prod_{i=k+1}^{n}(1-\alpha_i)\right] \qquad (12)
$$

6.2 Connection to constant step-sizes

As a sanity check, lets see if our equation holds for a constant sequence of step-sizes i.e. $(a_n) = \alpha \quad \forall n \in \mathbb{N}$. Notice that for the first term, the alphas don't depend on i anymore and thus becomes:

$$
Q_1 \prod_{i=1}^{n}(1-\alpha) = (1-\alpha)^n Q_1
$$

Similarly, in the second term, the weighted sum of rewards, the α's in the product again no longer depend on the indexing and thus becomes:

$$\sum_{k=1}^{n} \alpha R_k \prod_{i=k+1}^{n} (1-\alpha) = \sum_{k=1}^{n} \alpha R_k (1-\alpha)^{n-k}$$

6.3 Unbiased Constant-Step-Size Trick

The constant step-size formula as well as the general formula for any sequence of step-sizes are biased by the initial estimates Q_1 as evidenced in (12) and (2). Both equations have the term Q_1. The sample average estimate for the rewards does not have this problem. It is unbiased, but as we demonstrated in this paper, it applies poorly to non-stationary problems. In non-stationary problems, using a constant step-size estimate performs much better than the sample average. Unfortunately, unlike the sample average estimate, the constant step-size estimate is biased to the initial estimate Q_1. The book gives a way to avoid the bias of constant step sizes while retaining their advantages on non-stationary problems.

The book defines the step sizes depending on n, to process the nth reward for a particular action.

$$\beta_n := \frac{\alpha}{\bar{o}_n}$$

where $\alpha > 0$ is the constant step size parameter and $\bar{o}_n$ is defined as

$$\bar{o}_n := \bar{o}_{n-1} + \alpha(1 - \bar{o}_{n-1}) \quad \text{for } n > 0 \quad \text{and} \quad \bar{o}_0 = 0$$

To prove that this construction leads to an unbiased, constant step-size estimate, we will use the general formula that we derived in the first section and show that, unlike (12), the equation for the estimate does not depend on Q_1.

Proof. Starting with the definition for $\bar{o}_n$ given above:

$$\begin{aligned} \bar{o}_1 &= \bar{o}_0 + \alpha(1 - \bar{o}_0) && (13) \\ &= 0 + \alpha(1 - 0) && (14) \\ &= \alpha && (15) \end{aligned}$$

Using the definition for β_1:

$$\beta_1 = \frac{\alpha}{\bar{o}_1} = \frac{\alpha}{\alpha} = 1 \quad \alpha > 0 \qquad (16)$$

Continuing with the general formula (12):

$$Q_{n+1} = Q_1 \prod_{i=1}^{n}(1-\beta_i) + \sum_{k=1}^{n} \beta_k R_k \prod_{i=k+1}^{n}(1-\beta_i) \tag{17}$$

$$Q_{n+1} = Q_1(1-\beta_1) \prod_{i=2}^{n}(1-\beta_i) + \sum_{k=1}^{n} \beta_k R_k \prod_{i=k+1}^{n}(1-\beta_i) \tag{18}$$

$$Q_{n+1} = Q_1(1-1) \prod_{i=2}^{n}(1-\beta_i) + \sum_{k=1}^{n} \beta_k R_k \prod_{i=k+1}^{n}(1-\beta_i) \tag{19}$$

$$Q_{n+1} = Q_1 \cdot 0 \cdot \prod_{i=2}^{n}(1-\beta_i) + \sum_{k=1}^{n} \beta_k R_k \prod_{i=k+1}^{n}(1-\beta_i) \tag{20}$$

$$Q_{n+1} = \sum_{k=1}^{n} \beta_k R_k \prod_{i=k+1}^{n}(1-\beta_i) \tag{21}$$

$\square$

We have shown that to update the estimate for the reward received for an action, Q_{n+1}, we need only know the previous rewards received and not our initial estimate Q_1. Thus, we have shown that the estimate is not biased to our initial estimate for the reward function, Q_1.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, 1995.