CS0138 Assignment 1: Multiarm Bandit Problem

Peter Nadel

September 2025

1 Introduction

The multiarm bandit problem imagines k levers or slot machines, which when pulled return to the user some reward. Given this situation, our task is to maximize the amount of reward we receive from the k arms by trying to pull the most optimal arm at any given step. To do so, we must learn the underlying distribution of rewards for each arm, often through trial and error.

Solutions to the multiarm bandit problem are diverse, but what happens to them if each arm does not have a stable probability distribution from which it samples rewards? Instead, this assignment asks us to consider a *non-stationary* variant of the problem, in which the rewards are sampled from a distribution which adds a random residual each time it is sampled. In the following discussion, we will simulate this non-stationary formulation of the multiarm bandit problem, vary certain sampling distributions, and draw conclusions about reinforcement learning in these non-stationary environments.

2 Methods

2.1 Standard Non-stationary Multiarm Bandit Problem

This iteration of the multiarm bandit problem asks for a testbed of ten (k = 10) arms which are all initialized at the same value, in our case, 0. In our implementation, these arms are an array, whose values are updated at each pull. Rewards for each of these arms are sampled from a Gaussian with variance of 1 and a mean of the current value of the arm, which would be 0 at initialization. In this non-stationary case, random residuals are added the to the value of each arm once it is pulled. To learn about the effects of the non-stationary environment, we compare stationary and non-stationary learners. Below are the algorithms that we follow for each.

Note that the only difference between these two is how we do the update of an arm's value. In the former, we use an average to compute the current value of an arm, while in the latter, we use a small step size. In both cases, the scale multiplied by the net reward is a number between 0 and 1. In the stationary case, however, this value, $\frac{1}{N(A)}$ gets smaller and smaller as $N(A)$ gets larger and

Algorithm 1 Stationary learner algorithm

Initialize, for $a = 1$ to k
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$
$\epsilon \leftarrow 0.1$
$X \leftarrow \text{random\_float}(0.0, 1.0)$
loop
  if $X > \epsilon$ **then**
    $A \leftarrow \arg\max_a Q(a)$
  else
    $A \leftarrow \text{random\_int}(0, k)$
  end if
  $R \leftarrow bandit(A)$
  $N(A) + 1$
  $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$
end loop

Algorithm 2 Non-stationary learner algorithm

Initialize, for $a = 1$ to k
$Q(a) \leftarrow 0$
$\alpha \leftarrow 0.1$
$\epsilon \leftarrow 0.1$
$X \leftarrow \text{random\_float}(0.0, 1.0)$
loop
  if $X > \epsilon$ **then**
    $A \leftarrow \arg\max_a Q(a)$
  else
    $A \leftarrow \text{random\_int}(0, k)$
  end if
  $R \leftarrow bandit(A)$
  $Q(A) \leftarrow Q(A) + \alpha[R - Q(A)]$
end loop

larger. As a result, past rewards are less and less important to the calculation of $Q(A)$. Whereas in the non-stationary case, a constant α step size means that past rewards are weighed at the same importance from step to step. Our hypothesis, thus, is that using the non-stationary update will results in higher rewards over a large number of steps, as it will better learn the non-stationary distribution from which the arm samples its rewards.

2.2 Extra experiment

In addition to the test bed described above, we also chose to vary the distribution from which the residual is sampled. In the original formulation, this random walk is drawn from a Gaussian with a mean of one and standard deviation of 0.01. We ran extra experiments where residuals were drawn from two more distributions: lognormal and exponential. As these two distributions are vastly different from the Gaussian, behavior of the test bed changes significantly, giving us more insight into the non-stationary case of the multiarm bandit problem.

3 Results

3.1 Standard Non-stationary Multiarm Bandit Problem

Figures 1 and 2 below show the results for the standard test bed.

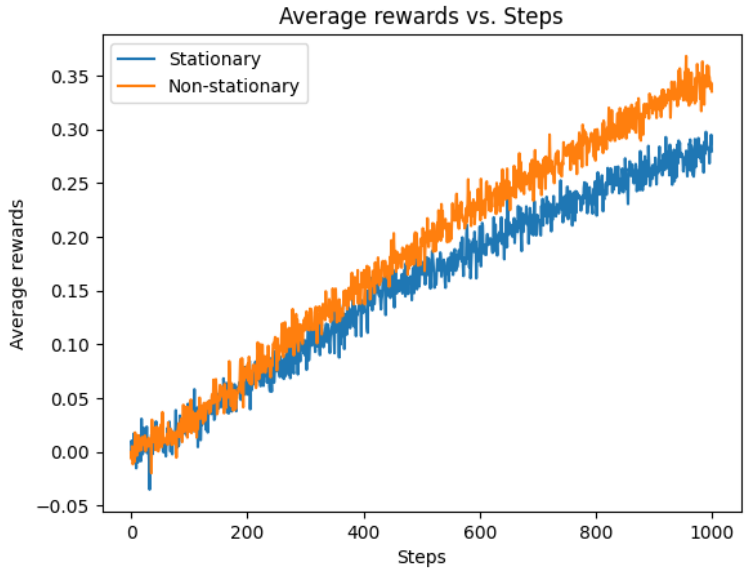

Figure 1: Normal: Average rewards vs. steps

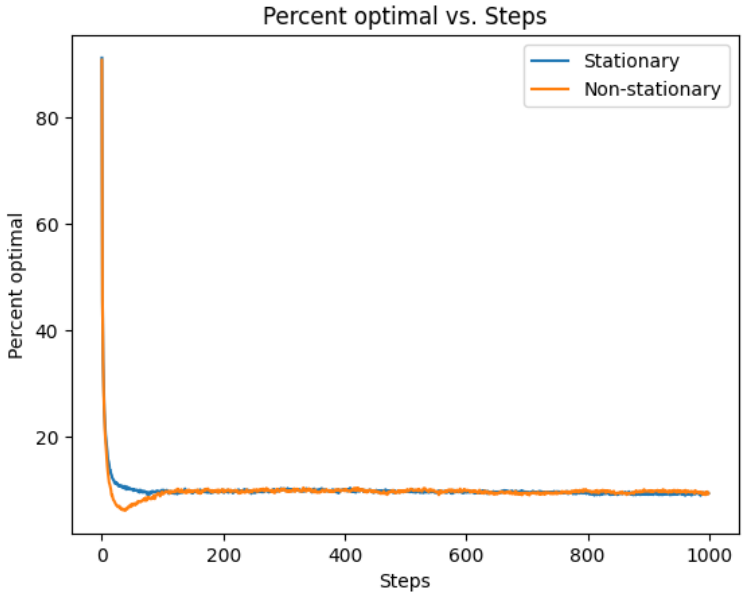

Figure 2: Normal: Percent optimal choice vs. steps

We had hypothesized that the non-stationary update would result in higher rewards over a large number of steps. We see in Figure 1 that rewards begin to diverge between the non-stationary and the stationary learners around 250 steps. After this, we see that the non-stationary learner's rewards increase linearly, while the stationary learner's rewards plateaus. As mentioned in the Methods sections, this discrepancy comes from the different ways that each update rule considers past results. In the stationary learner, the simple average is used, which heavily discounts past results when $N(A)$ is large. This is in opposition to the non-stationary learner, which is better able to adapt to the constantly shifting distributions with a set step size α.

Interestingly, both learners do a poor job at choosing the optimal arm. This result is consistent with the test bed set up. At the beginning, all of the arms start out at 0, so every action is optimal, resulting in both learners having a 100% optimal action selection rate in the first few steps. Quickly though, the random walks make the values diverge, meaning that at any given step only 1 out of the $k = 10$ arms is optimal. With the ϵ-greedy ($\epsilon = 0.1$) exploration policy, we choose an arm randomly 10% of the time. In the other 90% of times, only 10% of the values are optimal, leading to a global 10% optimal action selection rate, regardless of learner.

3.2 Extra experiment

As we can see from Figures 3, 4, 5 and 6, the nature of the non-stationary problem changes when we vary the distribution from which residuals are sampled. In particular, lognormal and exponential distributions are both positive and right skewed, meaning that their probability weight is not symmetric about their means. We observe in both the exponential and lognormal reward plots (Figures 3 and 5) that the non-stationary and stationary learners perform equally. Because both of the distributions are positive, they can only emit positive values as residuals, meaning that all arms are updated at roughly the same rate. In this "rising tides raise all boats" situation, the step size in the update rule makes little difference. $N(a)$ and α determine how much the update should consider past results, but in this case, past results do not vary in the same way as in the Gaussian case. In fact, past results are consistent enough that the simple average, that is, the stationary update rule is sufficient to match the results of the non-stationary update rule. Where the story becomes more complex is in the percent optimal selection plots. Though the exponential plot follows a similar, though not the same, pattern as the normal plot, the lognormal non-stationary learner is able to choose the optimal arm 40% of the time, as opposed to about 10% of the time in the normal and exponential cases. We believe that the lognormal non-stationary learner is better able to determine which arms are best, compared to the stationary learner, whereas in the normal and exponential cases, there are no best arms. The lognormal distribution does not just create this rising tide, it creates an upward trend, where winning arms tend to continue to win. The adaptability of the non-stationary learner latches on to this and can accurately learn it.

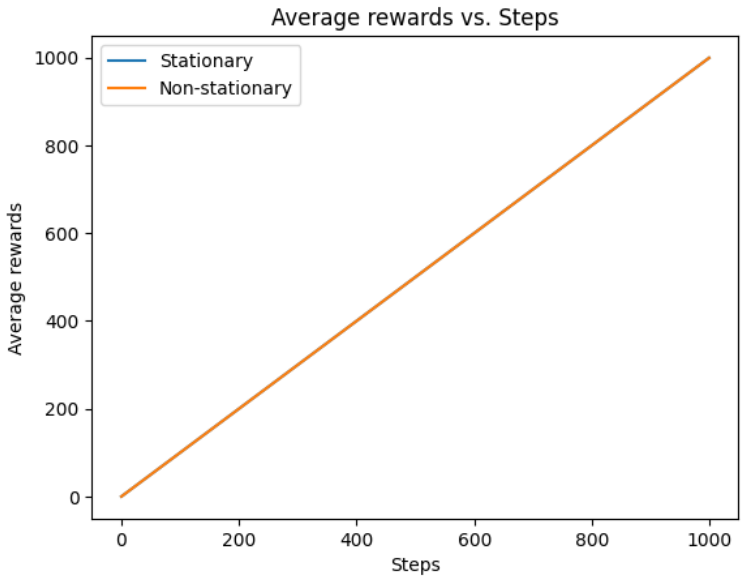

Figure 3: Lognormal: Average rewards vs. steps

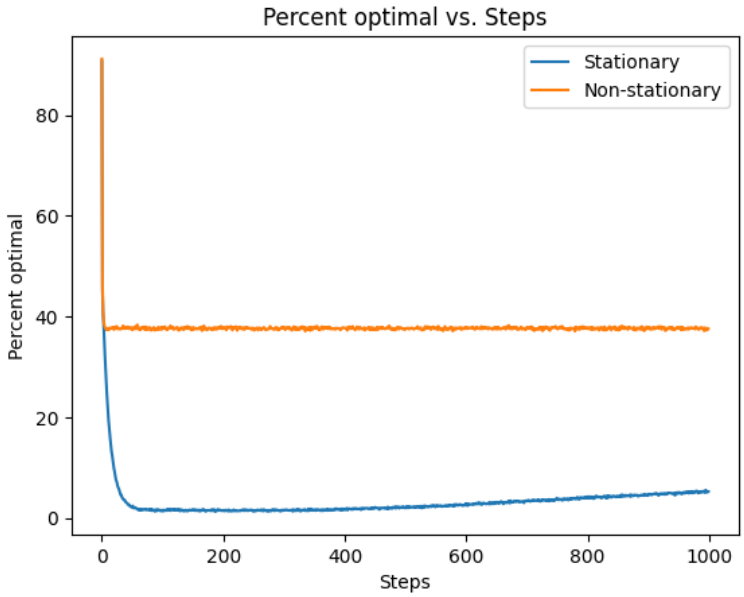

Figure 4: Lognormal: Percent optimal choice vs. steps

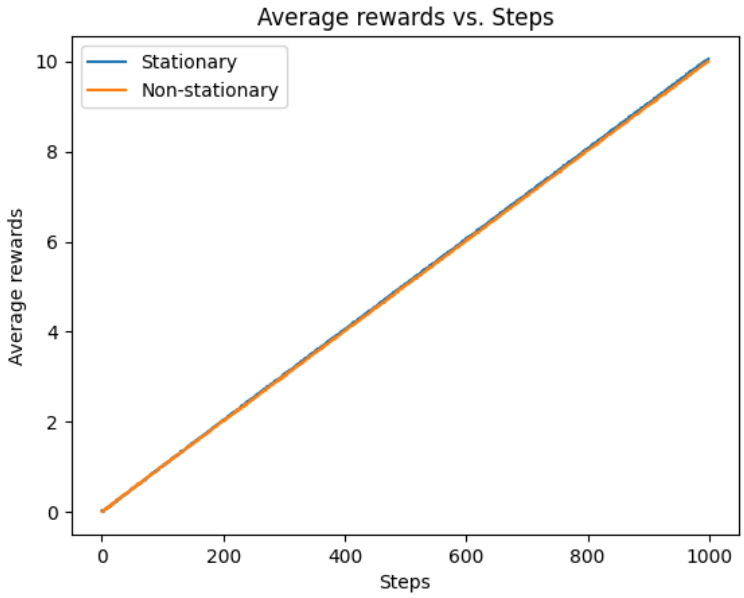

Figure 5: Exponential: Average rewards vs. steps

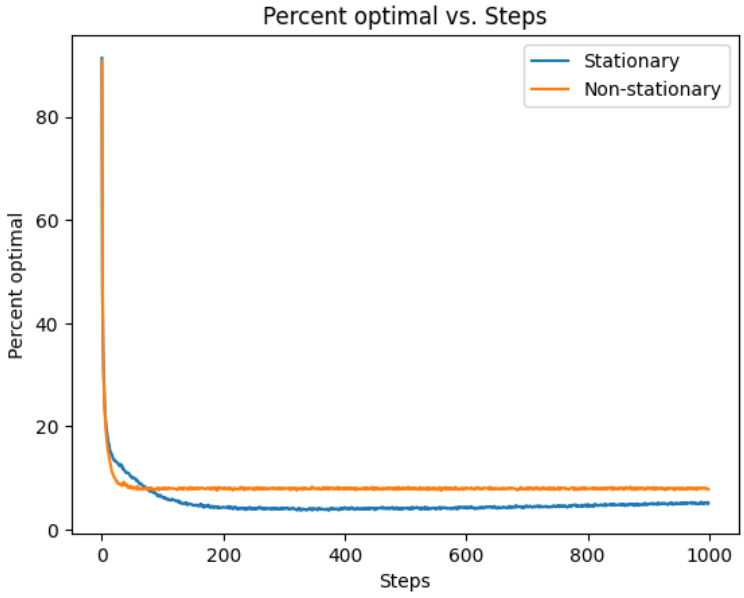

Figure 6: Exponential: Percent optimal choice vs. steps