Reinforcement learning Project

Objective - 1

To evaluate the performance of different bandit algorithms for the 10 – armed bandit problem. The focus is on learning the action values q*(a) and assessing the effectiveness of each algorithm in terms of average rewards and the frequency of selecting the optimal action.

The methods studied are:

1. Greedy with non-optimistic values
2. Epsilon-Greedy
3. Optimistic Initial Values with Greedy Approach
4. Gradient Bandit Algorithm

Problem Description

We simulate a 10-armed bandit problem where each arm has rewards normally distributed with unknown means and variance of 1. The means are generated from a normal distribution N(0,1). The objective is to learn the action values $q*(a)$ for each arm using different methods and to evaluate their performance in terms of:

1.Average reward acquired by the algorithm at each time step.

2.Percentage of time the optimal action is taken by the algorithm at each time step.

Methods

1.Greedy with non-optimistic values
This approach exploits current knowledge to maximize immediate rewards It spends no time at all sampling *apparently* inferior actions to see if they might really be better. Initialize the action value estimates to 0 and use a greedy strategy to select actions. We did initialization at Q(a) = 0 for all a. Always select the arm with the highest estimated value Q(a)

2.Epsilon-Greedy
With probability ϵ, select a random arm where basically to explore. With probability 1 - ϵ, select the arm with the highest estimated value Q(a)
Same as the greedy method mentioned above Q(a) = 0 for all a. Tuned using pilot runs, with 0.5. and decay rate 0.0004 the min value set was 0.1.

3.Optimistic Initial Values with Greedy Approach

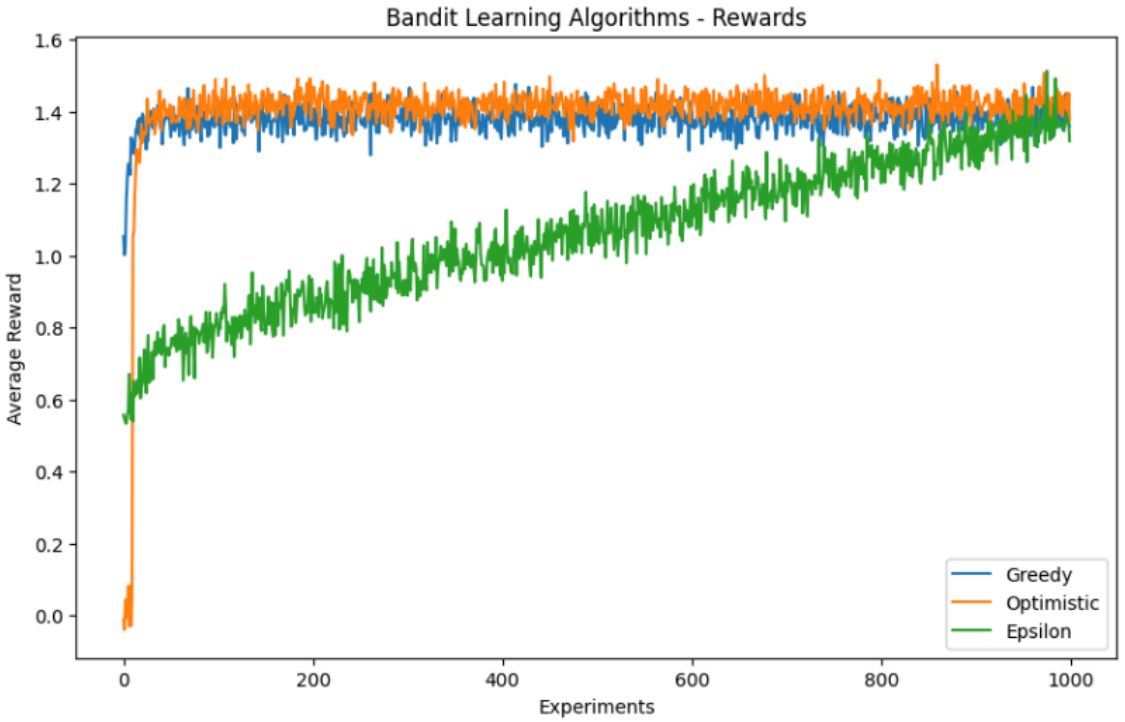
We initialize the action value estimates to a high value to promote more exploration in the beginning until we have some estimates for action values then we can benefit from our greedy choices. Initialization by Q(a) = 10 for all ‘a’ that is the optimistic starting value.

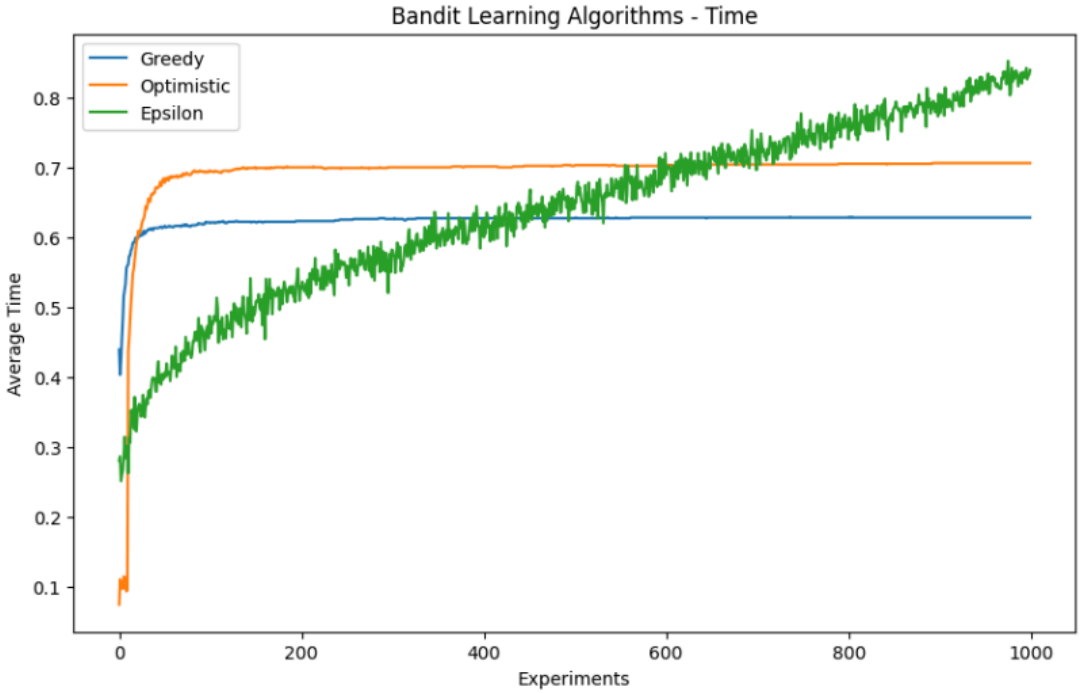
4.Gradient Bandit Algorithm
Here the model learns the preferences for each action and updates them using a gradient ascent. The preferences H(a) = 0 for all a. For learning rate is tuned using pilot runs, chosen alpha = 0.1

Experiment Setup

For the 10-armed bandit problem at hand we used 1000 number of steps for the 1000 bandit problems, using each method mentioned above in order to calculate the Average reward at each step and Percentage of time the optimal action taken at each time step.

Graphs





Comparison between Greedy, Optimistic and Epsilon methods

Greedy with Non-Optimistic

The percentage of optimal actions starts low and increases over time.It stabilizes around 60%, indicating that the lack of exploration prevents the algorithm from consistently finding and sticking to the optimal action.

Optimistic Initial Values with Greedy Approach

The percentage of optimal actions starts very high due to the optimistic initial values.It quickly stabilizes later on, suggesting that the algorithm was able to explore effectively early on and find the optimal action.This method shows the highest and quickest convergence to the optimal action percentage, which reflects its effectiveness in optimistic initial values to encourage exploration.

Epsilon-Greedy

The percentage of optimal actions starts low and gradually increases due to exploration. It converges more slowly and stabilizes .The fixed exploration rate means that even after finding the optimal action, the algorithm continues to explore, which can reduce the percentage of time the optimal action is taken.

Interpretation and Comparison

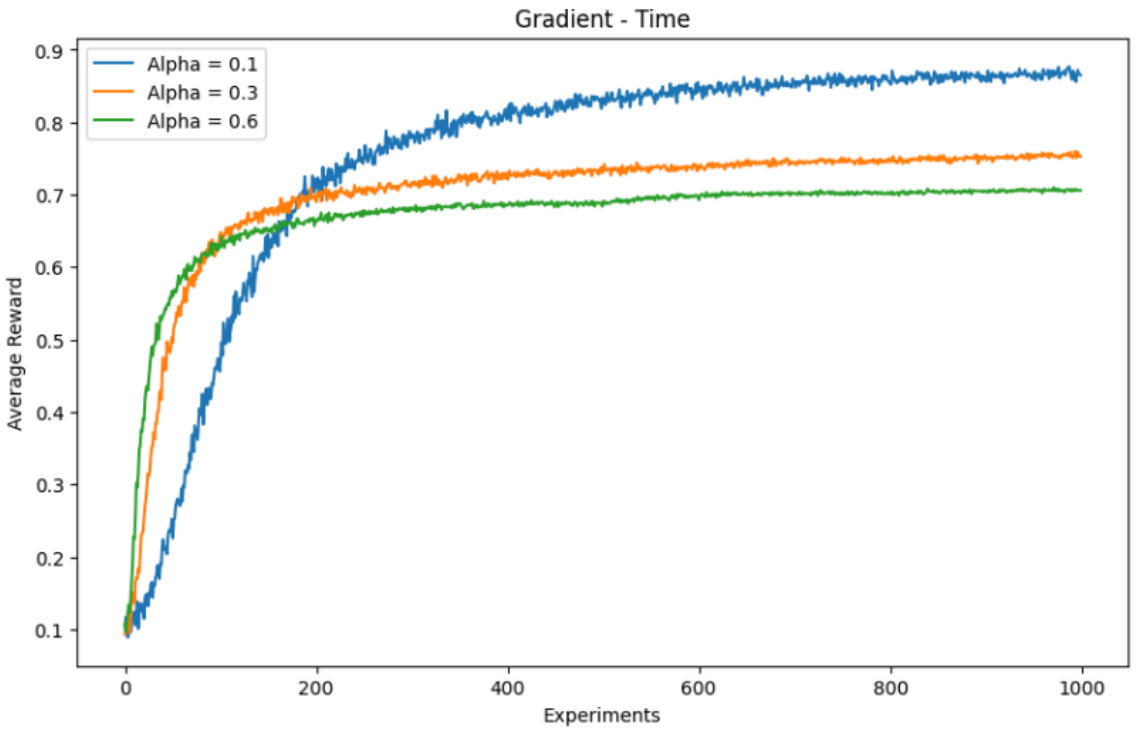
Average Reward

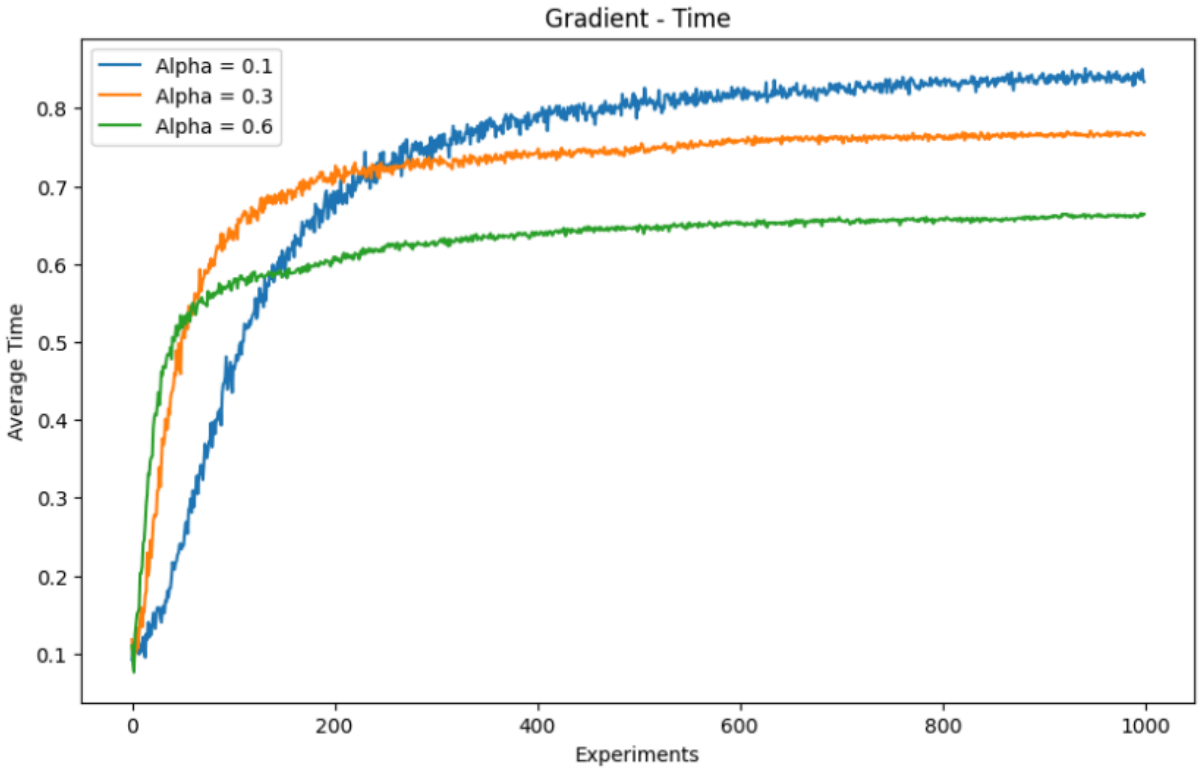
1. Optimistic Greedy performs the best in terms of average reward, quickly converging to a high reward and maintaining it.
2. Greedy performs moderately well but does not explore enough to consistently find the optimal actions.
3. Epsilon-Greedy explores more but converges more slowly, resulting in a lower average reward compared to the optimistic method.

Optimal Action Percentage

1. Optimistic Greedy also performs the best in terms of optimal action percentage, quickly finding and sticking to the optimal action.
2. Greedy performs moderately well, with a reasonable percentage of optimal actions but not as high as the optimistic method.
3. Epsilon-Greedy has a lower optimal action percentage due to continued exploration, even after finding the optimal action.

Gradient bandit algorithm.





Interpretation and Comparison

Average Reward

Alpha = 0.1 showed the highest average reward over time, indicating that a lower learning rate provides more stable and effective learning.

Alpha = 0.3 achieves faster initial learning but stabilizes at a slightly lower average reward.

Alpha = 0.6 converges quickly but at a lower average reward, suggesting that high learning rates can lead to instability.

Optimal Action Percentage

Alpha = 0.1 maintains the highest percentage of optimal actions, reflecting its stable learning process.

Alpha = 0.3 also achieves high optimal action percentage but is slightly lower than α=0.1

Alpha = 0.6 has the lowest percentage of optimal actions, indicating that the higher learning rate may cause the algorithm to struggle in consistently identifying the optimal action.

Conclusion

When , α=0.1 shows the best performance in both average reward and optimal action percentage, suggesting that a lower learning rate allows for more stable and effective learning in the gradient bandit algorithm.

Objective - 2

To evaluate the performance of different bandit algorithms under non-stationary conditions, including gradual and abrupt changes to reward distributions.

Introduction

The k-armed bandit problem is a classic reinforcement learning scenario where an agent must choose between k different actions (arms), each providing rewards from a probability distribution. In real-world scenarios, reward distributions can change over time due to various factors, making it important to consider non-stationary environments. This report evaluates the performance of different bandit algorithms under both gradual and abrupt changes to reward distributions.

Methods

10-armed bandit problem with normally distributed rewards.True means of rewards are drawn from a normal distribution N(0,1). Algorithms aim to learn the action values $q*(a)$ over time.

Non-stationary environments

Gradual changes

1. Drift change: $\mu t=\mu t-1+\epsilon t$, where $\epsilon t \sim N(0,0.001^2)$

2. Mean-reverting change: $\mu t=\kappa \mu t-1+\epsilon t$, where κ=0.5and $\epsilon t \sim N(0,0.01^2)$

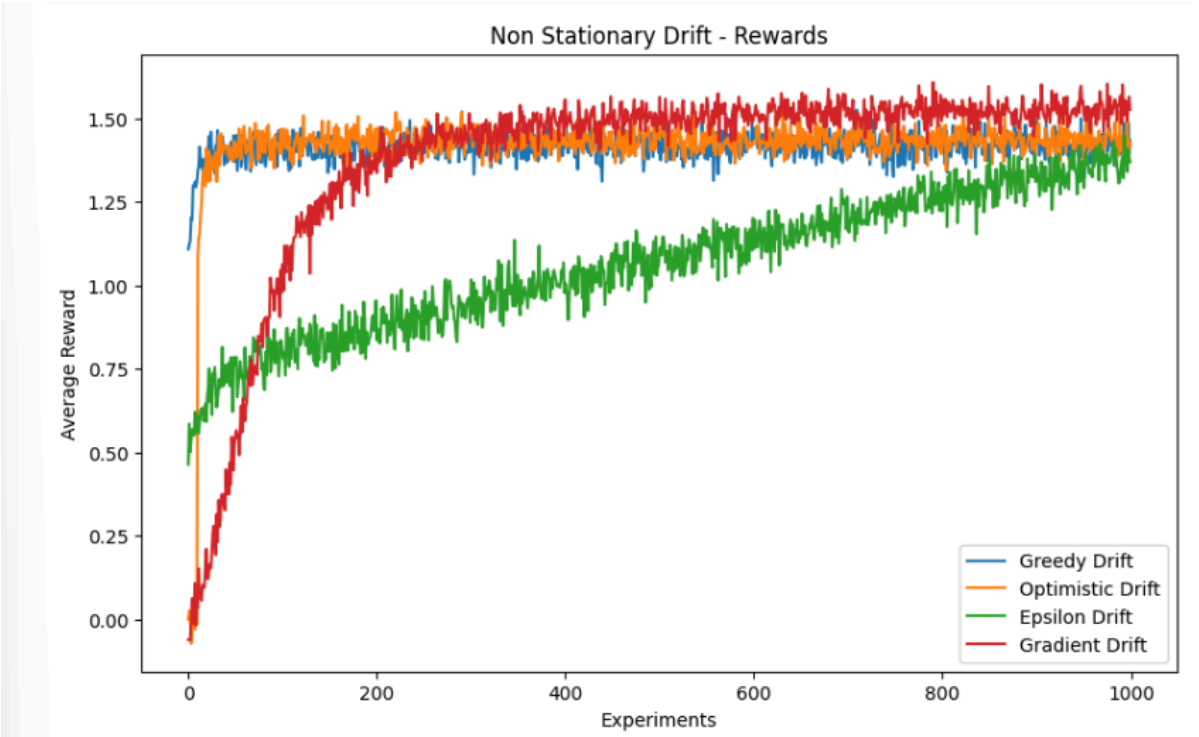
Abrupt changes

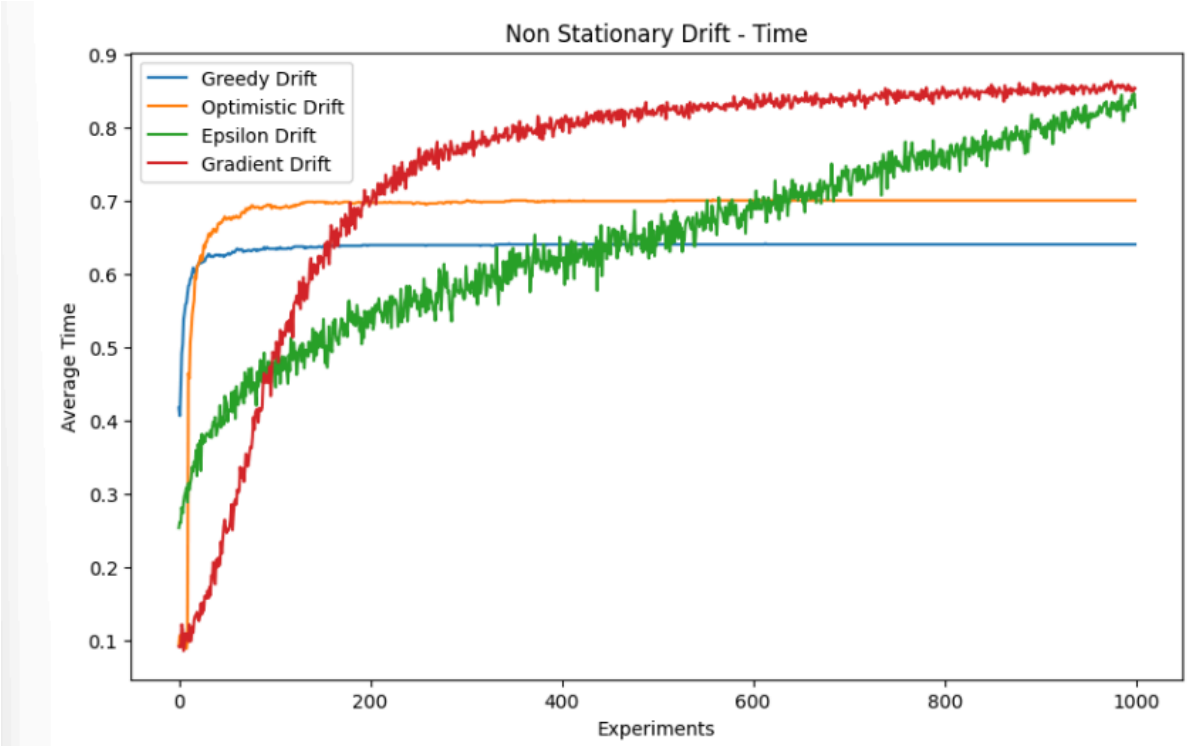
With a probability of 0.005 at each time step, permute the means of the reward distributions.

Algorithms used for comparison

1. Greedy with non-optimistic values:
2. Epsilon-Greedy:
3. Optimistic Initial Values with Greedy Approach
4. Gradient Bandit Algorithm

Drift change





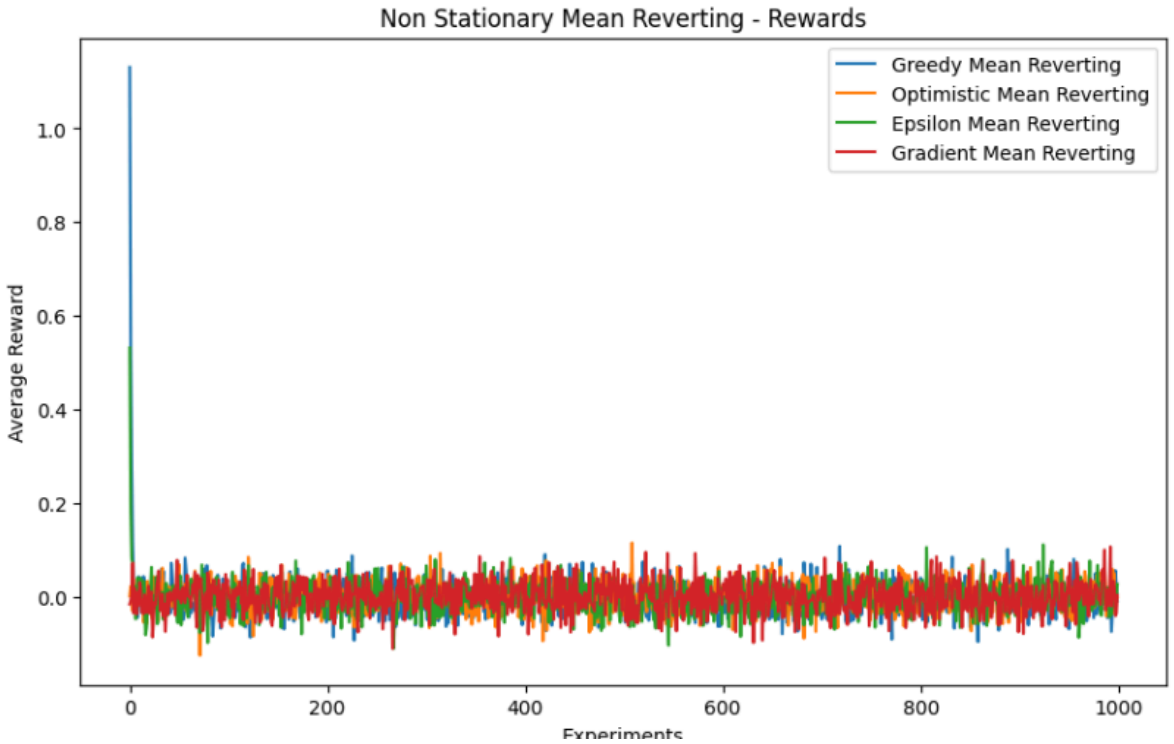
Conclusions

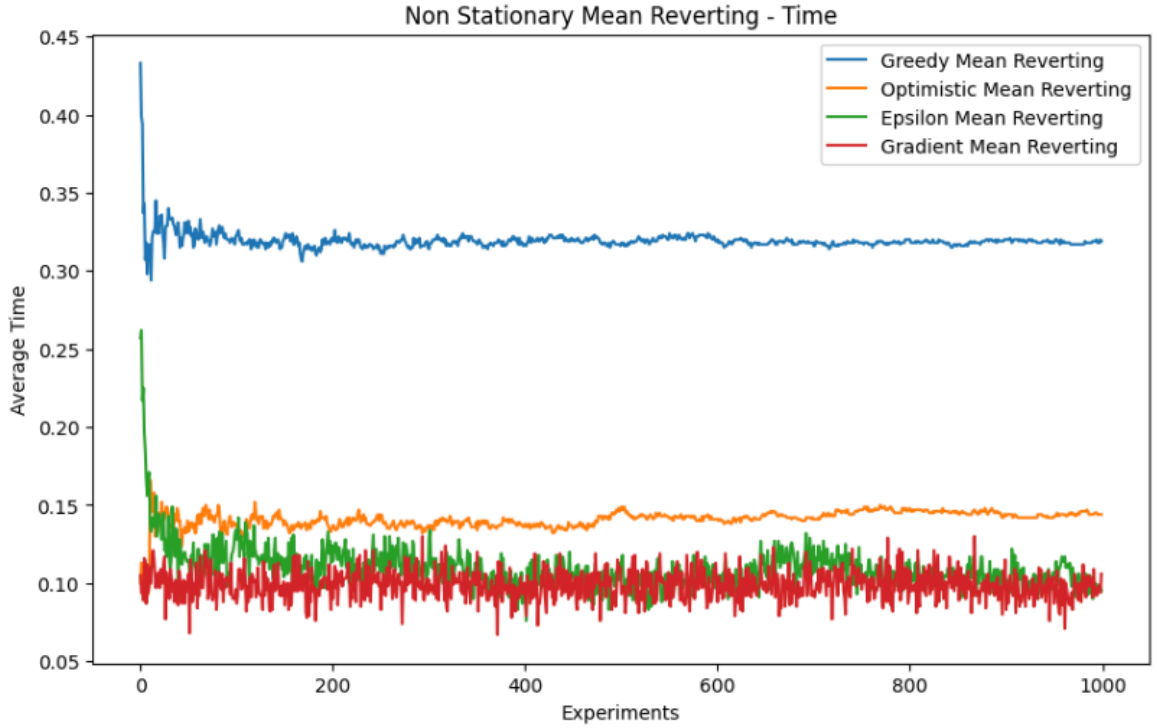
Best Performing Algorithm: The Gradient Bandit Algorithm demonstrates the best performance in terms of both average reward and optimal action percentage. Its ability to continuously adapt preferences based on observed rewards allows it to effectively handle the non-stationary environment.

Effectiveness of Optimistic Initial Values: The optimistic initial values with the greedy approach also perform well, leveraging early exploration to adapt to changes. This method strikes a good balance between exploration and exploitation in non-stationary settings.

Limitations of Greedy and Epsilon-Greedy Methods: The greedy algorithm with non-optimistic values performs poorly due to its lack of exploration. The epsilon-greedy method, while better, still struggles with a fixed exploration rate, leading to suboptimal performance in highly dynamic environments

Mean Reverting change


Non Stationary Mean Reverting - Rewards
Greedy Mean Reverting
Optimistic Mean Reverting
Epsilon Mean Reverting
Gradient Mean Reverting
Average Reward
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000


Non Stationary Mean Reverting - Time
Greedy Mean Reverting
Optimistic Mean Reverting
Epsilon Mean Reverting
Gradient Mean Reverting
Average Time
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
200
400
600
800
1000
Experiments

Conclusions

Overall Performance: All algorithms show poor performance under mean-reverting changes, with average rewards and optimal action percentages stabilizing around very low values.

Greedy with Non-Optimistic : This method performs the worst due to its lack of exploration, resulting in quick convergence to suboptimal actions.

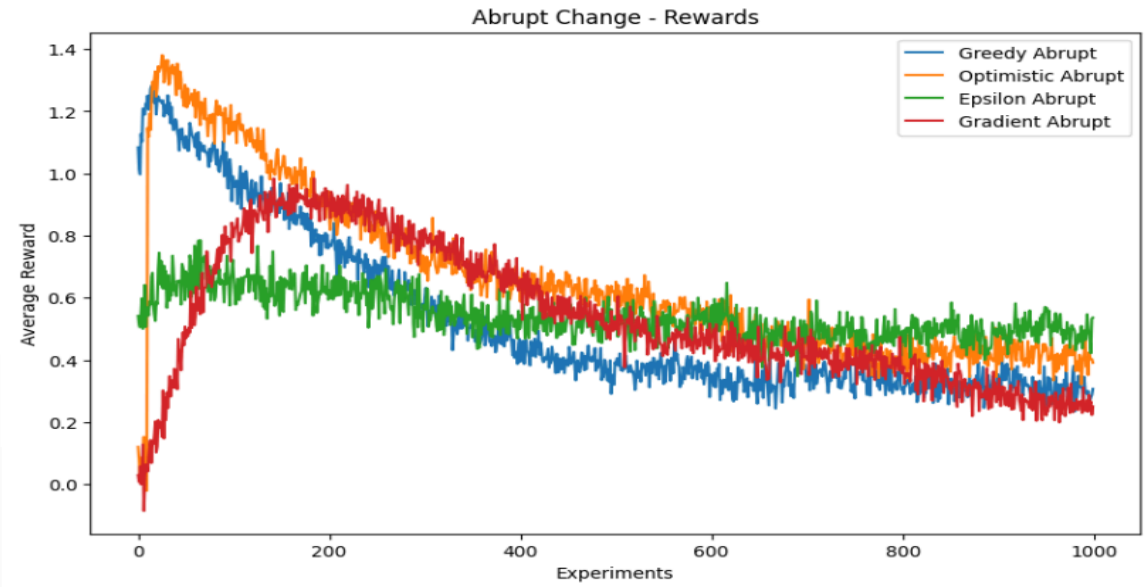
Epsilon-Greedy: This method performs slightly better than the greedy method due to its exploration component, but still struggles to adapt effectively.

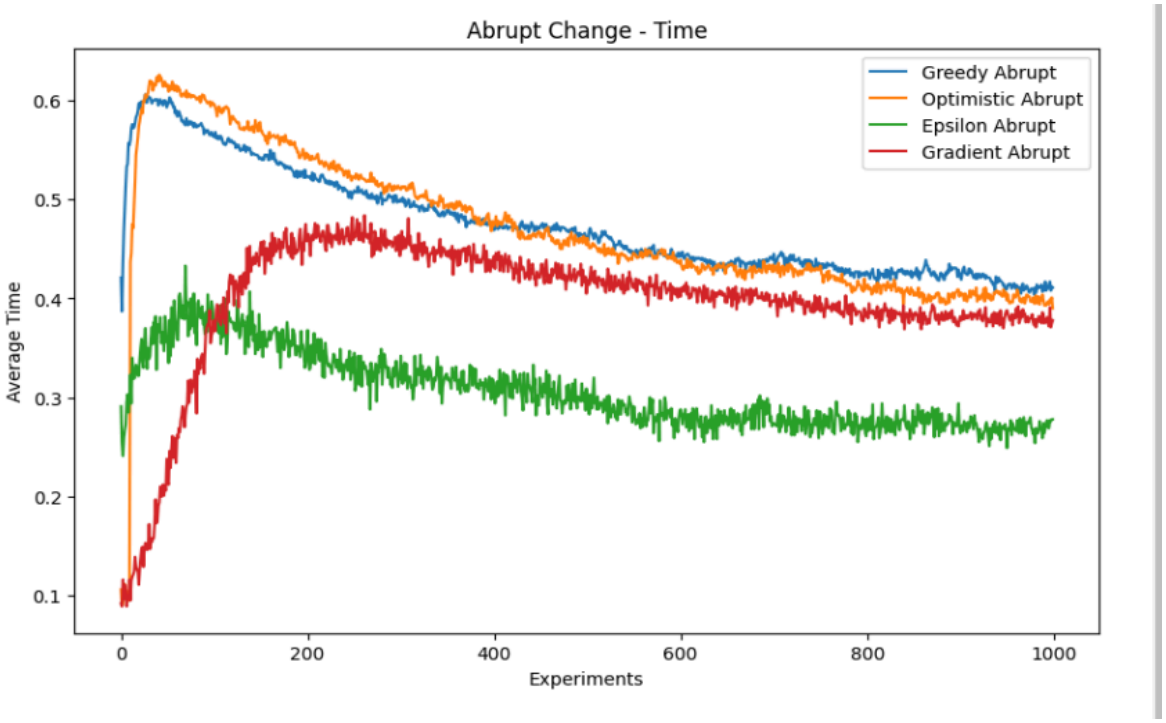
Optimistic Initial Values with Greedy Approach: This method shows initial promise due to early exploration, but ultimately fails to maintain performance as the environment changes.

Gradient Bandit Algorithm: This method shows some adaptation capabilities due to continuous preference updates, but overall performance remains low in a mean-reverting environment.

Abrupt Changes

Thee means of the reward distributions are permuted with a probability of 0.005 at each time step





Conclusion

Overall Performance: All algorithms show a decline in performance under abrupt changes, with average rewards and optimal action percentages stabilizing at lower values.

Greedy with Non-Optimistic Values: This method performs poorly due to its lack of exploration, resulting in quick convergence to suboptimal actions.

Epsilon-Greedy: This method performs slightly better than the greedy method due to its exploration component, but still struggles to adapt effectively to frequent changes.

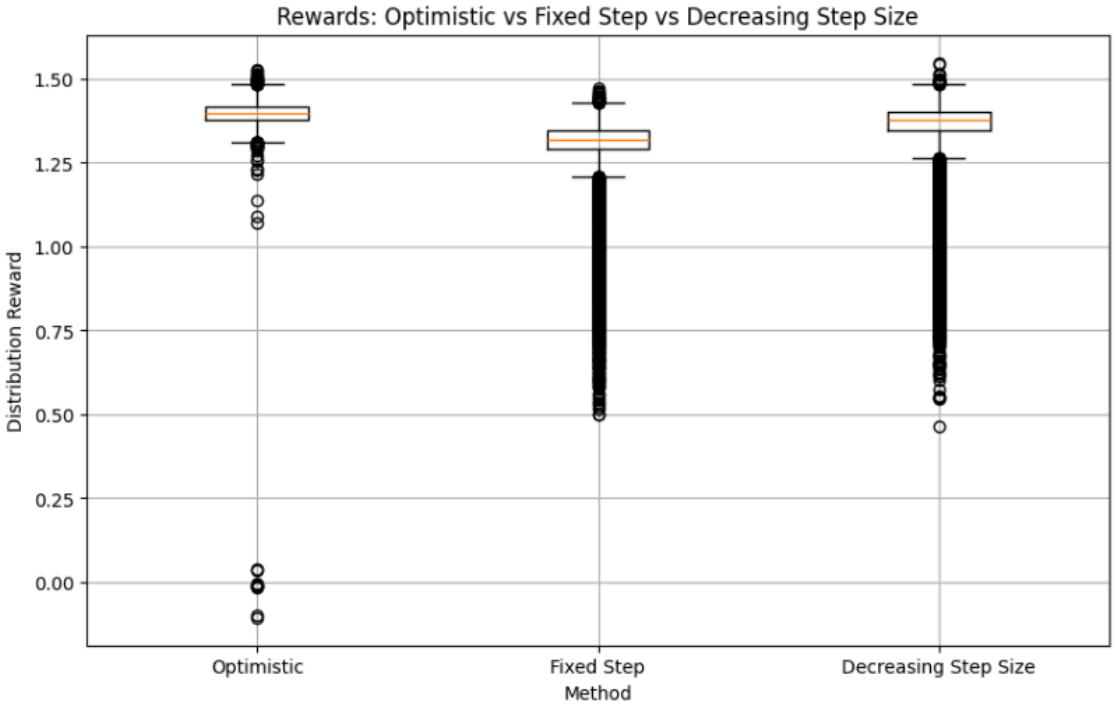
Optimistic Initial Values with Greedy Approach: This method shows initial promise due to early exploration, but ultimately fails to maintain performance as the environment changes frequently.

Gradient Bandit Algorithm: This method shows some adaptation capabilities due to continuous preference updates, but overall performance remains low in an environment with frequent abrupt changes.

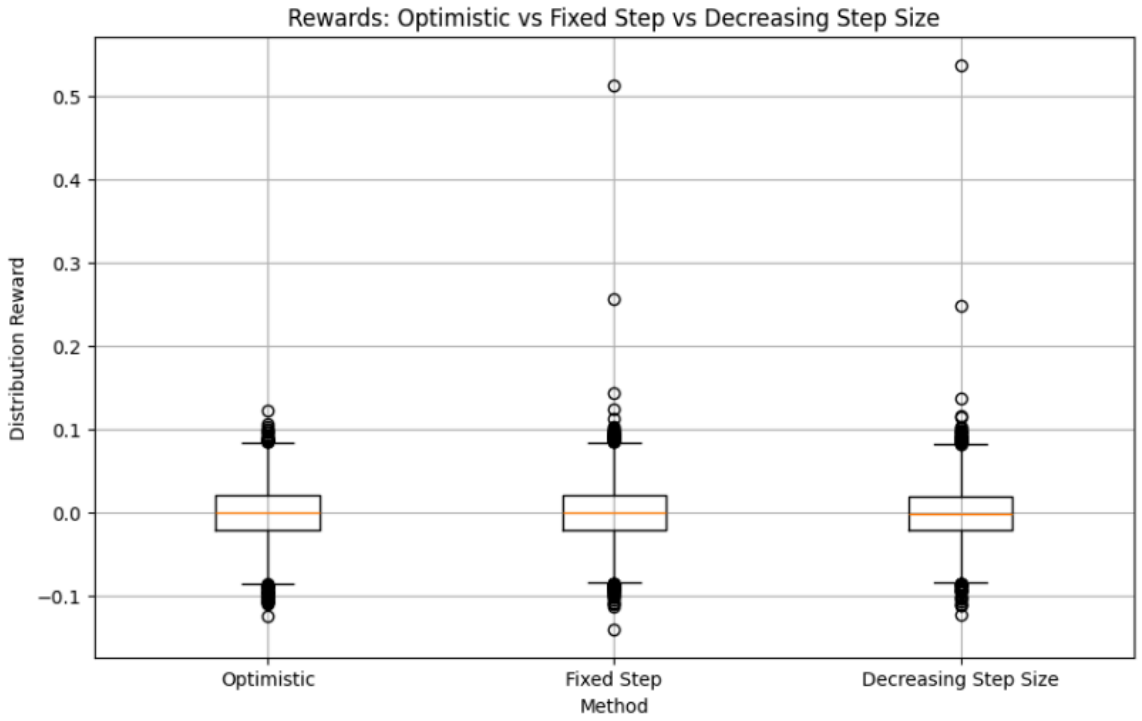
Terminal Reward Distributions

We analyze the terminal reward distributions for the three different bandit algorithms: Optimistic, Fixed Step, and Decreasing Step Size. The box plots below provide a visual representation of the reward distributions at the terminal step. The algorithm

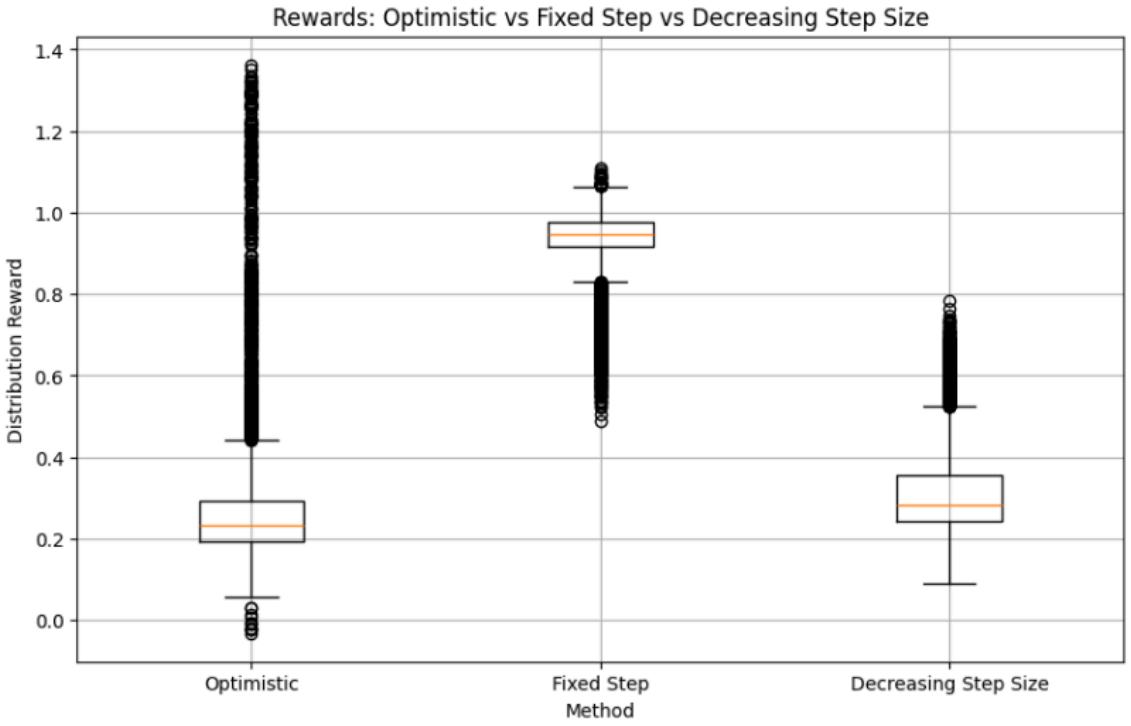
producing the most favorable distribution of rewards at the terminal step is considered preferable.



Optimistic method shows a high concentration of rewards around 1.25 with few outliers. Whereas the Fixed Step Exhibits a tighter distribution around 1.25, indicating consistent performance. And Decreasing Step Displays a similar distribution to Fixed Step but with more variability.



In the Optimistic method, the rewards are tightly clustered around zero, indicating poor performance in some trials.Whereas Fixed Step Similar distribution to Optimistic but with slightly higher variability. And Decreasing Step Size: Exhibits more variability in the rewards, suggesting instability.



The Optimist Shows a wide distribution of rewards with a median around 0.4. Whereas Fixed Step displays a tighter distribution around 1.0, indicating consistent high performance. And Decreasing Step Size shows a median around 0.6 with more spread, suggesting inconsistent performance.

Conclusions

Based on the box plots, we can draw the following conclusions:

Fixed Step Size This algorithm consistently produces high rewards with less variability, making it the most favorable in terms of terminal reward distribution. The tight clustering around higher reward values suggests it can reliably find optimal actions.

Optimistic Algorithm: This algorithm shows a wide range of rewards, with high rewards in some instances but also low rewards in others. The variability suggests it might be less reliable compared to the Fixed Step Size algorithm.

Decreasing Step Size Algorithm: This algorithm shows the most variability in rewards, indicating inconsistent performance. Although it achieves high rewards in some trials, it also produces lower rewards more frequently than the Fixed Step Size algorithm.