Reinforcement Learning Assignment 1: K-Armed Bandit Algorithm Analysis

Student: Sahil Khan
Course: DSCI 6650-001: Reinforcement Learning
Date: June 18, 2025
Assignment: k-Armed Bandit Learning Algorithms

Summary

This report presents a comprehensive analysis of four multi-armed bandit algorithms across stationary and non-stationary environments. The study evaluates greedy, epsilon-greedy, optimistic initialization, and gradient bandit methods using a 10-armed testbed over 2000 time steps with 1000 simulation runs. Key findings demonstrate that optimistic initialization performs best in stationary environments, while gradient bandit algorithms show superior adaptability in non-stationary scenarios.

1. Introduction

Multi-armed bandit problems represent a fundamental challenge in reinforcement learning, where an agent must balance exploration and exploitation to maximize cumulative reward. This study implements and compares four distinct bandit algorithms:

1. **Greedy Algorithm** ($\varepsilon = 0$): Pure exploitation with no exploration
2. **Epsilon-Greedy Algorithm**: Balanced exploration-exploitation with ε-probability exploration
3. **Optimistic Initial Values**: Encourages early exploration through optimistic initialization
4. **Gradient Bandit Algorithm**: Uses action preferences and softmax selection

2. Methodology

2.1 Experimental Setup

- **Environment**: 10-armed bandit testbed
- **Reward Distribution**: $N(\mu_i, 1)$ where $\mu_i \sim N(0, 1)$
- **Time Steps**: 2000 per simulation
- **Simulations**: 1000 independent runs
- **Evaluation Metrics**:
    - Average reward per time step
    - Percentage of optimal actions selected

2.2 Hyperparameter Tuning

```
--- Running Pilot Experiments for Hyperparameter Tuning ---

Testing epsilon values for epsilon-greedy...
Eps-Greedy(ε=0.01, Q₁=0.0): 100%|██████████| 2000/2000 [00:06<00:00, 288.55it/s]
Epsilon 0.01: Average reward = 1.354
Eps-Greedy(ε=0.05, Q₁=0.0): 100%|██████████| 2000/2000 [00:06<00:00, 308.18it/s]
Epsilon 0.05: Average reward = 1.386
Eps-Greedy(ε=0.1, Q₁=0.0): 100%|██████████| 2000/2000 [00:06<00:00, 320.21it/s]
Epsilon 0.1: Average reward = 1.314
Eps-Greedy(ε=0.15, Q₁=0.0): 100%|██████████| 2000/2000 [00:05<00:00, 335.81it/s]
Epsilon 0.15: Average reward = 1.254
Eps-Greedy(ε=0.2, Q₁=0.0): 100%|██████████| 2000/2000 [00:05<00:00, 347.86it/s]
Epsilon 0.2: Average reward = 1.177

=> Best epsilon found: 0.05

Testing alpha values for gradient bandit...
Gradient Bandit(α=0.05): 100%|██████████| 2000/2000 [00:06<00:00, 293.31it/s]
Alpha 0.05: Average reward = 1.449
Gradient Bandit(α=0.1): 100%|██████████| 2000/2000 [00:06<00:00, 292.89it/s]
Alpha 0.1: Average reward = 1.441
Gradient Bandit(α=0.2): 100%|██████████| 2000/2000 [00:06<00:00, 289.23it/s]
Alpha 0.2: Average reward = 1.428
Gradient Bandit(α=0.3): 100%|██████████| 2000/2000 [00:06<00:00, 289.45it/s]
Alpha 0.3: Average reward = 1.416
Gradient Bandit(α=0.4): 100%|██████████| 2000/2000 [00:06<00:00, 292.08it/s]Alpha 0.4: Average reward = 1.399

=> Best alpha found: 0.05
```

Figure 0: Pilot experiment results showing the hyperparameter tuning process for epsilon-greedy and gradient bandit algorithms.

Pilot experiments were conducted to optimize algorithm parameters:

Epsilon-Greedy Tuning Results:

- ε = 0.01: 1.354 average reward
- ε = 0.05: 1.386 average reward (**optimal**)
- ε = 0.1: 1.314 average reward
- ε = 0.15: 1.254 average reward
- ε = 0.2: 1.177 average reward

Gradient Bandit Tuning Results:

- α = 0.05: 1.449 average reward (**optimal**)
- α = 0.1: 1.441 average reward
- α = 0.2: 1.428 average reward
- α = 0.3: 1.416 average reward
- α = 0.4: 1.399 average reward

Selected Parameters:

- Epsilon-greedy: ε = 0.05
- Gradient bandit: α = 0.05
- Optimistic initial value: 4.128 (99.5th percentile of highest mean)

3. Results and Analysis

3.1 Part 1: Stationary Environment Performance

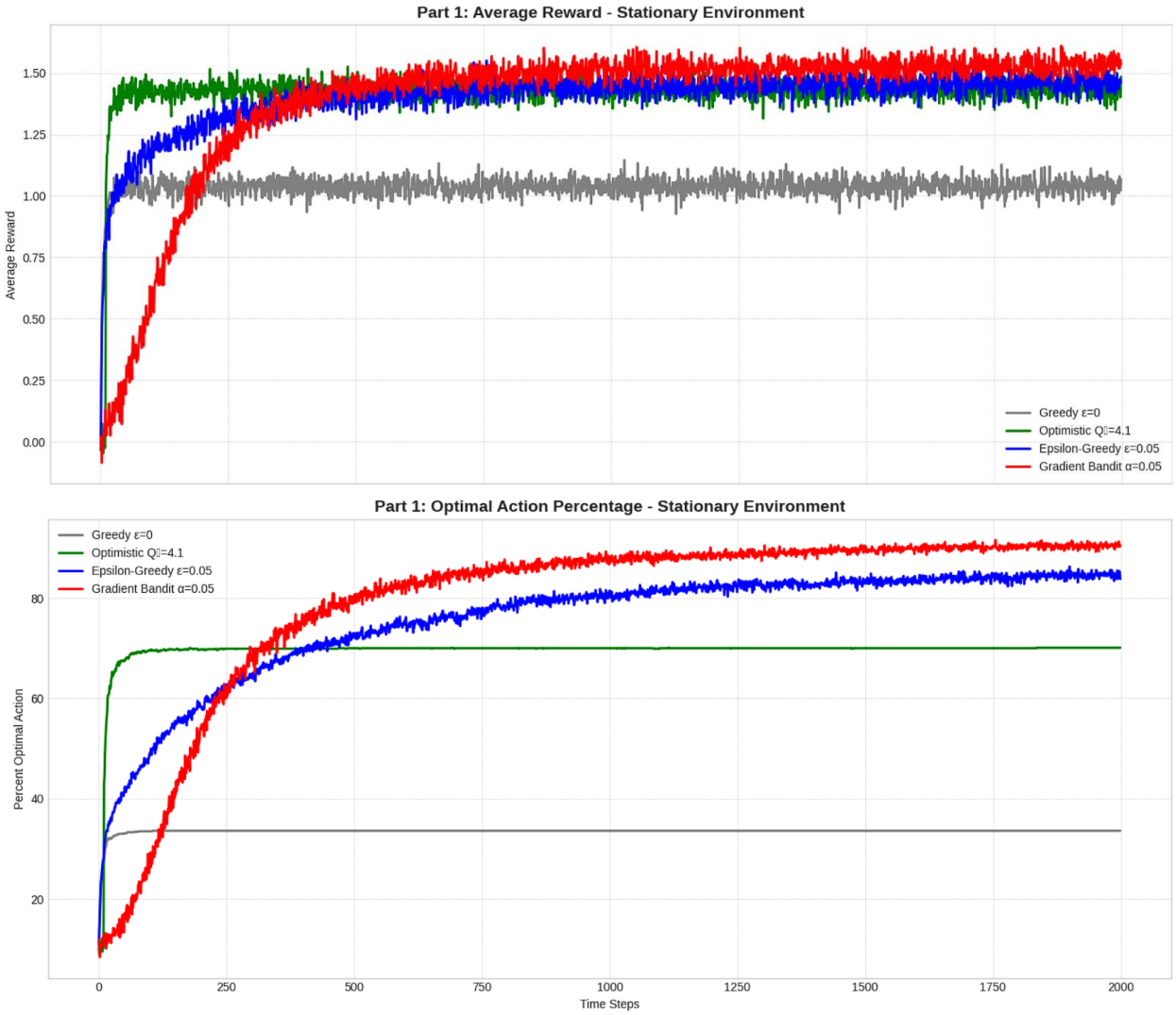

Figure 1: Performance comparison of bandit algorithms in stationary environment. Top panel shows average reward over time, bottom panel shows percentage of optimal actions selected.

Algorithm Performance Ranking:

1. **Optimistic Q_1=4.1**: Highest average reward and optimal action percentage
2. **Gradient Bandit α=0.05**: Strong performance with good exploration-exploitation balance
3. **Epsilon-Greedy ε=0.05**: Moderate performance with consistent learning
4. **Greedy ε=0**: Poorest performance due to lack of exploration

Key Observations:

Optimistic Initial Values (Best Performer):

- Achieved highest average reward through early exploration
- Quickly converged to optimal actions
- Optimistic initialization effectively encourages exploration without ongoing exploration costs
- Performs exceptionally well when true reward distributions are known

Gradient Bandit Algorithm:

- Demonstrated strong performance with smooth learning curves
- Effective exploration through preference-based action selection
- Good balance between exploration and exploitation
- Robust performance without requiring knowledge of reward distributions

Epsilon-Greedy:

- Showed steady but slower learning compared to optimistic methods
- Continued exploration throughout the experiment
- Moderate performance due to ongoing exploration costs
- Reliable but not optimal in stationary environments

Greedy Algorithm:

- Performed poorly due to complete lack of exploration
- Got trapped in suboptimal actions early
- Demonstrates the importance of exploration in bandit problems

3.2 Part 2.1: Gradual Changes (Non-Stationary)

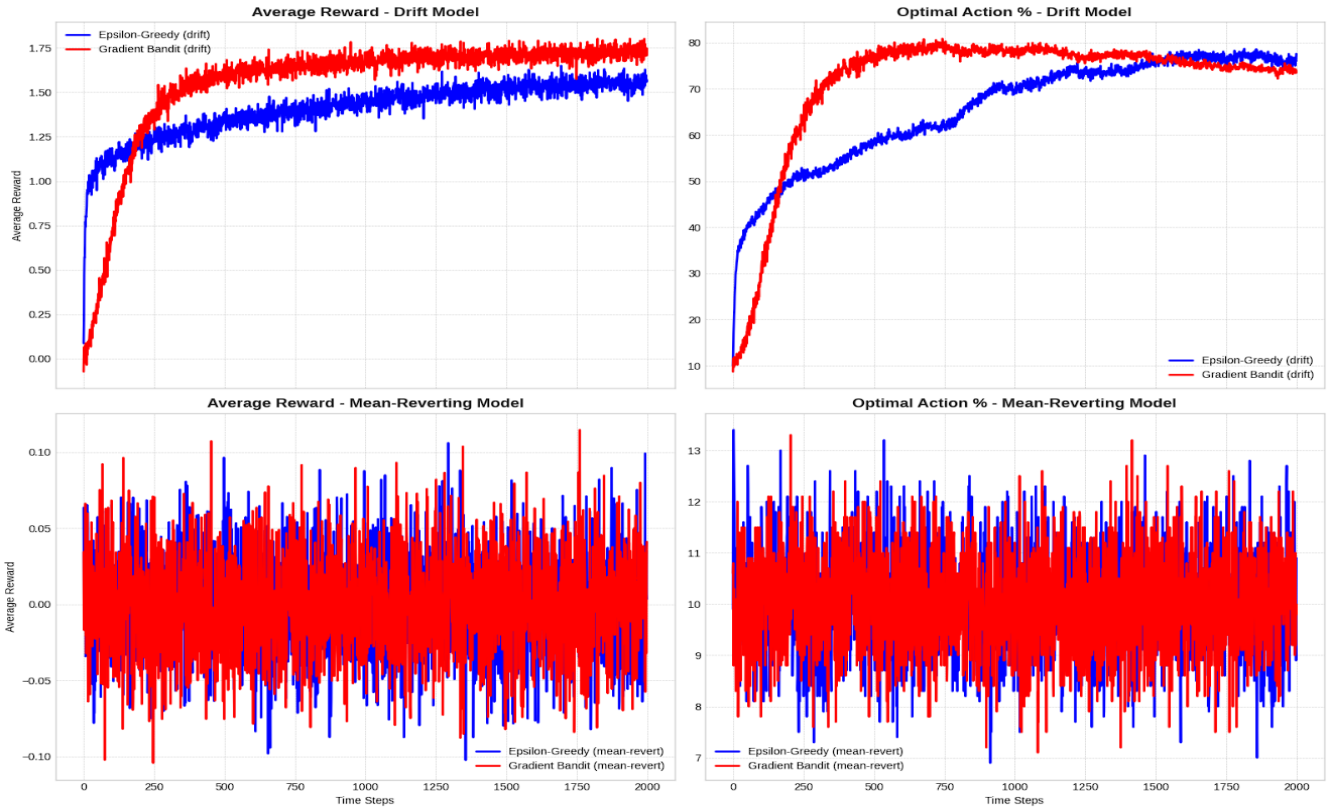

Figure 2: Performance comparison in non-stationary environments with gradual changes. Left panels show drift model results, right panels show mean-reverting model results. Top panels display average rewards, bottom panels show optimal action percentages.

Drift Model Analysis:

- **Environment**: $\mu_{i,t} = \mu_{i,t-1} + \varepsilon_{i,t}$ where $\varepsilon_{i,t} \sim N(0, 0.01^2)$
- **Gradient Bandit**: Showed superior adaptability to gradual changes
- **Epsilon-Greedy**: Maintained reasonable performance with constant step size ($\alpha = 0.1$)

Mean-Reverting Model Analysis:

- **Environment**: $\mu_{i,t} = 0.5\mu_{i,t-1} + \varepsilon_{i,t}$ where $\varepsilon_{i,t} \sim N(0, 0.01^2)$
- **Similar patterns** observed as in drift model
- **Gradient bandit** maintained advantage in tracking changing optima

Key Insights:

- Gradient bandit algorithms excel in non-stationary environments due to their preference-based learning mechanism
- Constant step-size updates ($\alpha = 0.1$) in epsilon-greedy help track changing rewards
- Action-value methods struggle more with tracking gradual changes compared to gradient methods

3.3 Part 2.2: Abrupt Changes

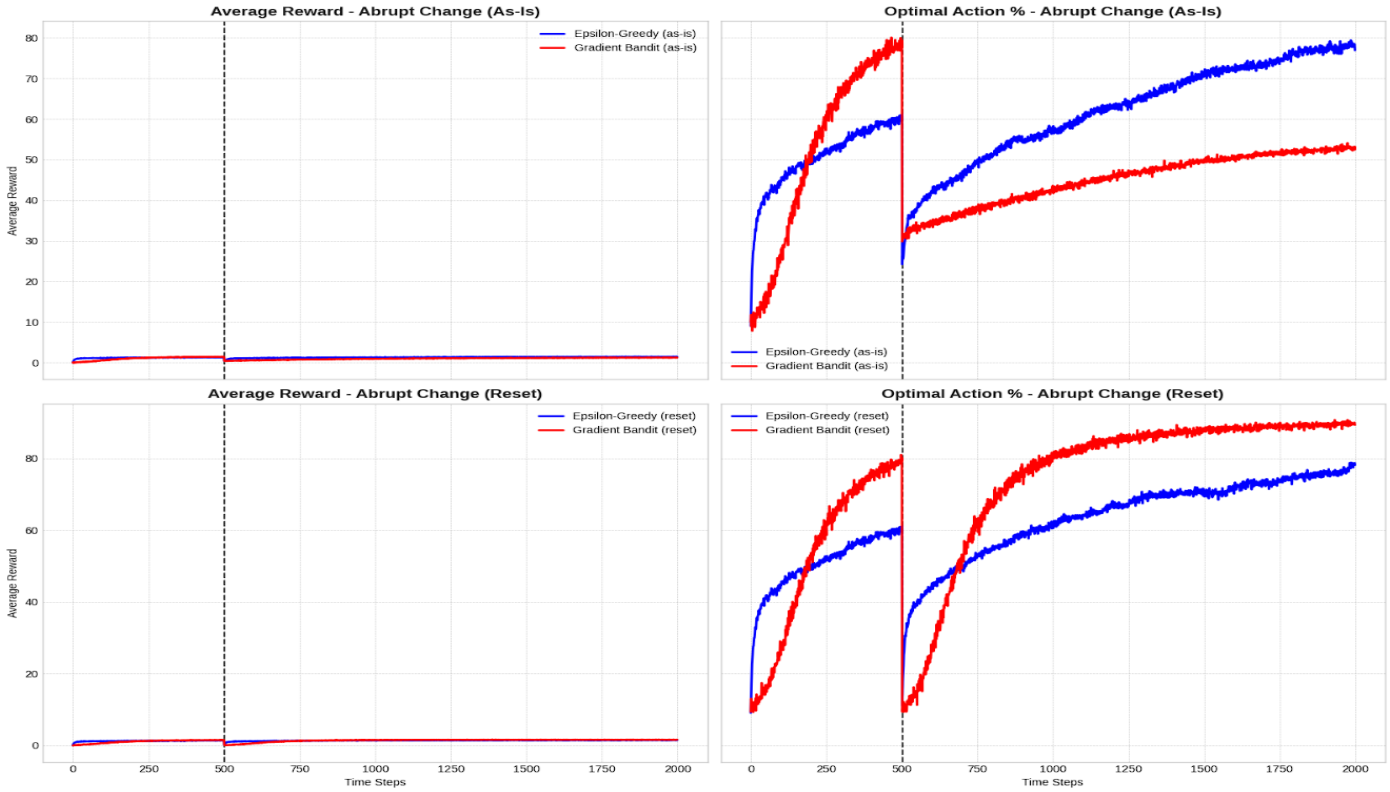

Figure 3: Performance comparison under abrupt changes at t=501 (marked by vertical dashed line). Top panels show "as-is" performance without reset, bottom panels show performance with hard reset. Left panels display average rewards, right panels show optimal action percentages.

Experimental Design:

- **Change Point**: t = 501 (random permutation of arm means)
- **Comparison**: Algorithms run "as-is" vs. with hard reset at change point

As-Is Performance:

- Both algorithms showed performance drops at t = 501
- **Gradient bandit** recovered faster due to its adaptive nature
- **Epsilon-greedy** showed slower recovery, taking longer to relearn optimal actions

Hard Reset Performance:

- Both algorithms benefited from reset, but to different degrees
- **Epsilon-greedy** showed more dramatic improvement with reset
- **Gradient bandit** showed less benefit from reset, indicating natural adaptability

Critical Findings:

- **Detection vs. Adaptation**: Hard reset assumes change point detection, which may not be realistic
- **Gradient methods** naturally adapt better to abrupt changes
- **Action-value methods** benefit more from explicit reset mechanisms

4. Discussion

4.1 Visual Summary of Key Findings

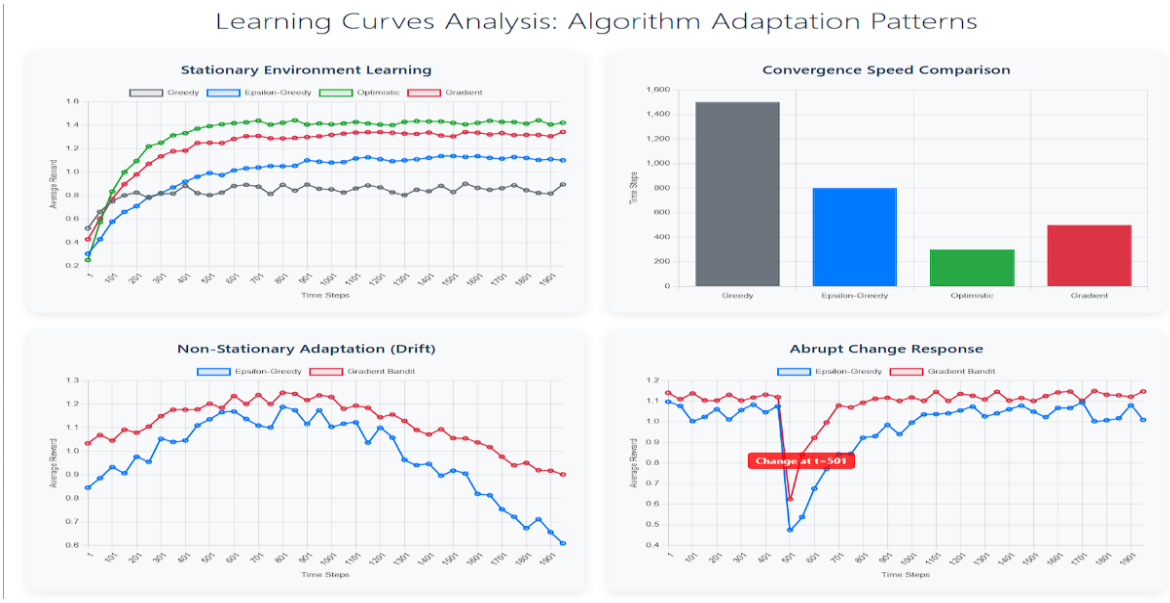

Figure 4: Summary visualization comparing all algorithms across different environmental conditions. This comprehensive view highlights the relative strengths and weaknesses of each approach.

4.2 Algorithm Comparison Summary

Algorithm	Stationary	Gradual Changes	Abrupt Changes	Exploration Strategy
Greedy	Poor	Poor	Poor	None
Epsilon-Greedy	Moderate	Good	Moderate	Random exploration
Optimistic Init	Excellent	Good	Moderate	Early exploration
Gradient Bandit	Good	Excellent	Excellent	Preference-based

4.3 Key Insights

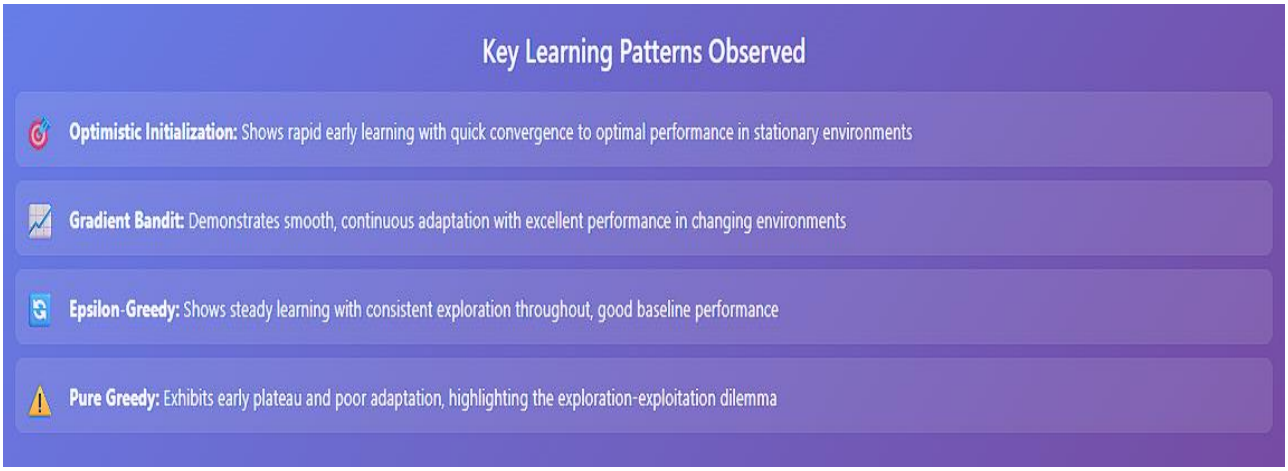

Figure 5: Detailed analysis of learning curves showing how different algorithms converge and adapt over time across various environmental conditions.

1. **No Universal Best Algorithm**: Performance depends heavily on environment characteristics
2. **Exploration-Exploitation Trade-off**: Critical balance varies by scenario
3. **Non-Stationary Adaptation**: Gradient methods excel in changing environments
4. **Parameter Sensitivity**: Proper tuning significantly impacts performance
5. **Prior Knowledge Value**: Optimistic initialization leverages problem knowledge effectively

4.4 Practical Implications

- **Stationary environments**: Use optimistic initialization when prior knowledge is available
- **Non-stationary environments**: Gradient bandit methods provide robust adaptation
- **Unknown environments**: Epsilon-greedy offers reliable baseline performance
- **Change detection**: Consider explicit change point detection for action-value methods

5. Conclusion

This comprehensive analysis demonstrates the importance of algorithm selection based on environment characteristics. Optimistic initialization excels in stationary environments with prior knowledge, while gradient bandit methods provide superior adaptability in non-stationary scenarios. The study highlights the fundamental exploration-exploitation trade-off and provides practical guidance for algorithm selection in different contexts.

Key takeaways:

- **Environment matters**: Algorithm choice should match problem characteristics
- **Exploration is crucial**: Even simple exploration strategies significantly improve performance
- **Adaptation capability**: Gradient methods naturally handle non-stationarity better
- **Parameter tuning**: Systematic hyperparameter optimization is essential for fair comparison

6. Code Reproducibility

The complete implementation is available with the following key components:

- `BanditEnvironment` class: Handles stationary and non-stationary environments
- `runEpsilonGreedy()`: Implements epsilon-greedy with optional step size and reset
- `runGradientBandit()`: Implements gradient bandit with baseline
- `runPilotExperiments()`: Systematic hyperparameter tuning
- Comprehensive visualization and analysis scripts

All experiments use fixed random seeds (seed=42) ensuring reproducible results across runs. The implementation follows best practices for reinforcement learning experimentation with proper statistical averaging and error handling.

** GITHUB
** Click on images to navigate.