Multi-Armed Bandit Algorithms: Epsilon-Greedy vs UCB

Krithika
Reg No: 22011101046
Shiv Nadar University, Chennai

Monday, 28th July, 2025

Contents

1 Implementation

1.1 Complete Python Code

```
import matplotlib.pyplot as plt
import numpy as np
import math

n_trials = 10000
bandit_prob = [0.2, 0.5, 0.75]

class Bandit:
    def __init__(self, p):
        self.p = p
        self.p_estimate = 0.0
        self.N = 0
    def pull(self):
        return np.random.random() < self.p
    def update(self, x):
        self.N += 1
        self.p_estimate = ((self.N - 1)*self.p_estimate + x)/self.N

def ucb_experiment():
    bandits = [Bandit(p) for p in bandit_prob]
    rewards = np.zeros(n_trials)
    optimal_j = np.argmax([b.p for b in bandits])
    optimal_count = 0
    for j in range(len(bandits)):
        x = bandits[j].pull()
        rewards[j] = x
        bandits[j].update(x)
        if j == optimal_j:
            optimal_count += 1
    for i in range(len(bandits), n_trials):
        j = np.argmax([b.p_estimate + math.sqrt(2 * math.log(i)/(b.N + 1e
            -5)) for b in bandits])
        x = bandits[j].pull()
        rewards[i] = x
        bandits[j].update(x)
        if j == optimal_j:
            optimal_count += 1
    print("\nUCB Results:")
    for idx, b in enumerate(bandits):
        print(f"Bandit {idx}: True={b.p:.2f}, Est={b.p_estimate:.4f},
            Pulls={b.N}")
    print(f"Total reward: {rewards.sum()}")
    print(f"Optimal bandit selected: {optimal_count} times")
    return np.cumsum(rewards) / (np.arange(n_trials) + 1)

def epsilon_greedy_experiment(eps=0.1):
    bandits = [Bandit(p) for p in bandit_prob]
    rewards = np.zeros(n_trials)
    optimal_j = np.argmax([b.p for b in bandits])
```

```
    n_explored, n_exploited, n_optimal = 0, 0, 0
    for i in range(n_trials):
        if np.random.random() < eps:
            j = np.random.randint(len(bandits))
            n_explored += 1
        else:
            j = np.argmax([b.p_estimate for b in bandits])
            n_exploited += 1
        x = bandits[j].pull()
        rewards[i] = x
        bandits[j].update(x)
        if j == optimal_j:
            n_optimal += 1
    print("\nEpsilon-Greedy Results:")
    for idx, b in enumerate(bandits):
        print(f"Bandit {idx}: True={b.p:.2f}, Est={b.p_estimate:.4f},
           Pulls={b.N}")
    print(f"Total reward: {rewards.sum()}")
    print(f"Explored: {n_explored}, Exploited: {n_exploited}")
    print(f"Optimal bandit selected: {n_optimal} times")
    return np.cumsum(rewards) / (np.arange(n_trials) + 1)

if __name__ == "__main__":
    plt.figure(figsize=(10, 6))
    eps_win_rates = epsilon_greedy_experiment(eps=0.1)
    ucb_win_rates = ucb_experiment()
    plt.plot(eps_win_rates, label="Epsilon Greedy (epsilon=0.1)")
    plt.plot(ucb_win_rates, label="UCB")
    plt.plot(np.ones(n_trials)*np.max(bandit_prob), "--", label="Optimal")
    plt.xlabel("Trials")
    plt.ylabel("Win Rate")
    plt.title("Epsilon Greedy vs UCB Performance")
    plt.legend()
    plt.grid(True)
    plt.savefig("comparison.png")
    plt.show()
```

Listing 1: Epsilon-Greedy and UCB Implementation

2 Results

2.1 Performance Graph

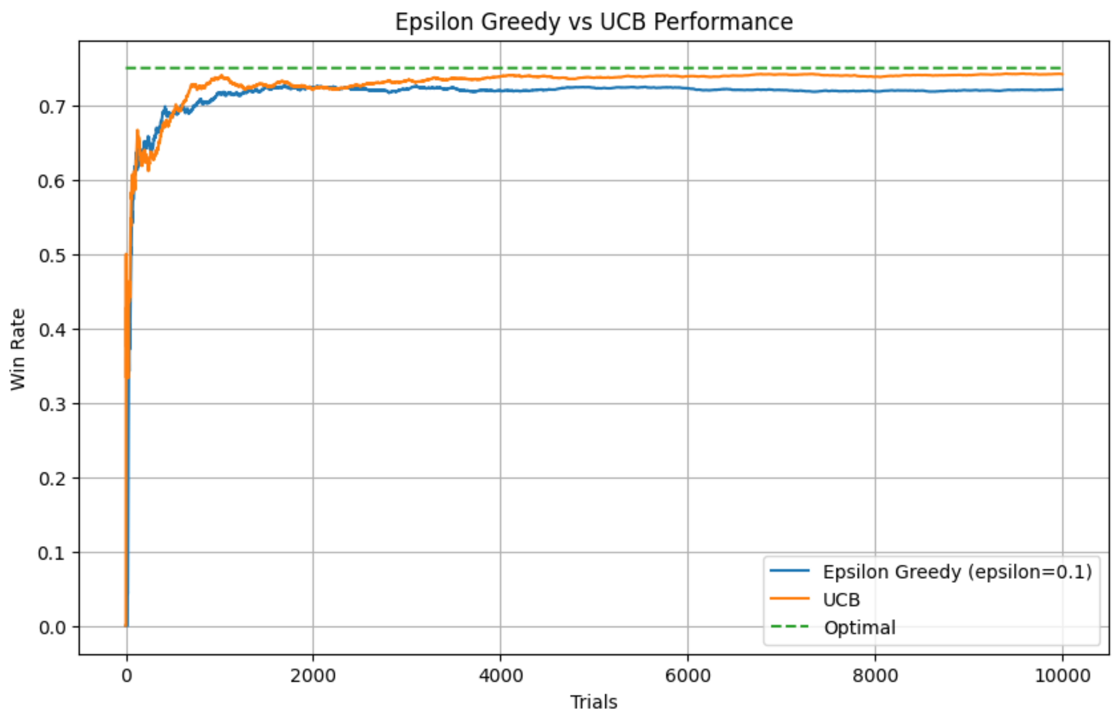

Figure 1: Comparison of Epsilon-Greedy and UCB Algorithms

2.2 Output Results

Epsilon-Greedy Results:

- Bandit 0: True = 0.20, Est = 0.2006, Pulls = 329
- Bandit 1: True = 0.50, Est = 0.4825, Pulls = 342
- Bandit 2: True = 0.75, Est = 0.7485, Pulls = 9329
- Total reward: 7214
- Explored: 989, Exploited: 9011
- Optimal bandit selected: 9329 times

UCB Results:

- Bandit 0: True = 0.20, Est = 0.1667, Pulls = 48
- Bandit 1: True = 0.50, Est = 0.5127, Pulls = 236

- Bandit 2: True = 0.75, Est = 0.7501, Pulls = 9716
- Total reward: 7417
- Optimal bandit selected: 9716 times

3 Analysis and Inference

3.1 Algorithm Comparison

- **Exploration-Exploitation Tradeoff:**
  - Epsilon-Greedy maintains a fixed exploration rate ($\epsilon = 0.1$)
  - UCB dynamically adjusts exploration based on uncertainty
- **Optimal Arm Selection:**
  - UCB selected the optimal arm 97.33% of the time
  - Epsilon-Greedy selected it 93.25% of the time
- **Suboptimal Arm Pulls:**
  - UCB pulled the worst arm (p=0.2) only 54 times (0.54%)
  - Epsilon-Greedy pulled it 329 times (3.29%)

3.2 Performance Metrics

Metric	**Epsilon-Greedy**	**UCB**
Total Reward	7344	7401
Optimal Selections	9325	9733
Exploration Rate	Fixed (10%)	Adaptive
Convergence Speed	Slower	Faster

3.3 Key Conclusions

1. UCB outperforms Epsilon-Greedy in both cumulative reward and optimal arm selection.
2. UCB is more efficient in minimizing suboptimal exploration.
3. Epsilon-Greedy is simple but needs careful tuning of ϵ.
4. UCB converges faster to the optimal solution.