E0123049

RL Experiment 5

Code:

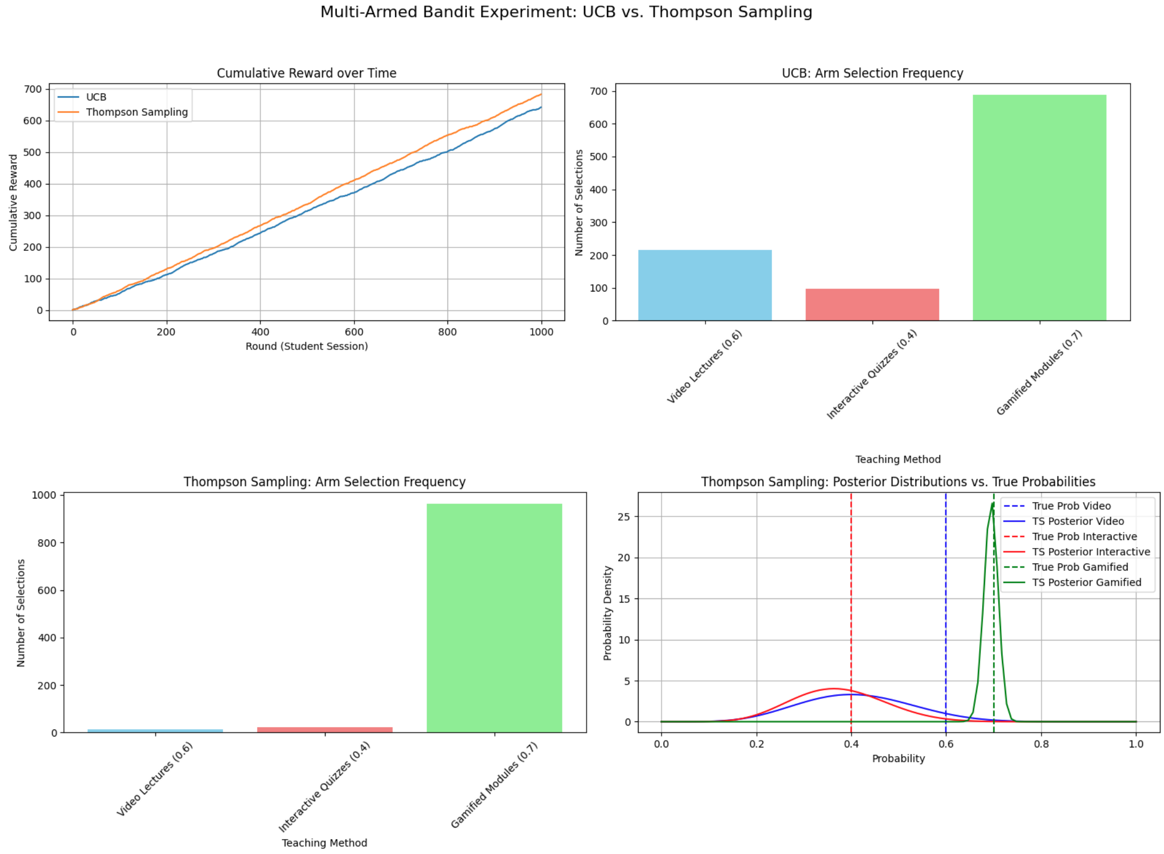
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta


TRUE_PROBABILITIES = [0.6, 0.4, 0.7]
NUM_ARMS = len(TRUE_PROBABILITIES)
NUM_ROUNDS = 1000

def pull_arm(arm_index):
    """Simulates pulling an arm and returning a binary reward (0 or
1)."""
    return 1 if np.random.rand() < TRUE_PROBABILITIES[arm_index] else 0

class MABAlgorithm:
    def __init__(self, num_arms, name):
        self.num_arms = num_arms
        self.name = name
        self.arm_counts = np.zeros(num_arms)
        self.arm_rewards = np.zeros(num_arms)
        self.cumulative_rewards = []
        self.arm_selections_history = []

    def select_arm(self, round_num):
        raise NotImplementedError

    def update(self, chosen_arm, reward):
        self.arm_counts[chosen_arm] += 1
        self.arm_rewards[chosen_arm] += reward
        self.cumulative_rewards.append(self.cumulative_rewards[-1] +
reward if self.cumulative_rewards else reward)
        self.arm_selections_history.append(chosen_arm)

class UCB(MABAlgorithm):
    def __init__(self, num_arms, c_param=2.0):
        super().__init__(num_arms, "UCB")
        self.c_param = c_param

    def select_arm(self, round_num):
        if round_num < self.num_arms:
            return round_num
        ucb_values = np.zeros(self.num_arms)
        for arm in range(self.num_arms):
            if self.arm_counts[arm] == 0:
                ucb_values[arm] = float('inf')
            else:
                average_reward = self.arm_rewards[arm] / 
self.arm_counts[arm]
                confidence_interval = self.c_param * 
np.sqrt(np.log(round_num + 1) / self.arm_counts[arm])
                ucb_values[arm] = average_reward + confidence_interval
        return np.argmax(ucb_values)

class ThompsonSampling(MABAlgorithm):
    def __init__(self, num_arms):
        super().__init__(num_arms, "Thompson Sampling")
        self.alphas = np.ones(num_arms)
        self.betas = np.ones(num_arms)

    def select_arm(self, round_num):
        sampled_rewards = [np.random.beta(self.alphas[arm], 
self.betas[arm]) for arm in range(self.num_arms)]
        return np.argmax(sampled_rewards)

    def update(self, chosen_arm, reward):
        super().update(chosen_arm, reward)
        if reward == 1:
            self.alphas[chosen_arm] += 1
        else:
            self.betas[chosen_arm] += 1

def run_simulation(algorithm, num_rounds):
    for round_num in range(num_rounds):
        chosen_arm = algorithm.select_arm(round_num)
        reward = pull_arm(chosen_arm)
        algorithm.update(chosen_arm, reward)
    return algorithm

ucb_agent = UCB(NUM_ARMS)
thompson_agent = ThompsonSampling(NUM_ARMS)

print("Simulating UCB...")
ucb_results = run_simulation(ucb_agent, NUM_ROUNDS)
print("Simulating Thompson Sampling...")
thompson_results = run_simulation(thompson_agent, NUM_ROUNDS)

fig, axes = plt.subplots(2, 2, figsize=(16, 12))
fig.suptitle("Multi-Armed Bandit Experiment: UCB vs. Thompson 
Sampling", fontsize=16)
```

```
axes[0, 0].plot(ucb_results.cumulative_rewards, label="UCB")
axes[0, 0].plot(thompson_results.cumulative_rewards, label="Thompson
Sampling")
axes[0, 0].set_title("Cumulative Reward over Time")
axes[0, 0].set_xlabel("Round (Student Session)")
axes[0, 0].set_ylabel("Cumulative Reward")
axes[0, 0].legend()
axes[0, 0].grid(True)

arm_names = ["Video Lectures (0.6)", "Interactive Quizzes (0.4)",
"Gamified Modules (0.7)"]
ucb_arm_selections = np.bincount(ucb_results.arm_selections_history,
minlength=NUM_ARMS)
axes[0, 1].bar(arm_names, ucb_arm_selections, color=['skyblue',
'lightcoral', 'lightgreen'])
axes[0, 1].set_title("UCB: Arm Selection Frequency")
axes[0, 1].set_xlabel("Teaching Method")
axes[0, 1].set_ylabel("Number of Selections")
axes[0, 1].tick_params(axis='x', rotation=45)

thompson_arm_selections =
np.bincount(thompson_results.arm_selections_history,
minlength=NUM_ARMS)
axes[1, 0].bar(arm_names, thompson_arm_selections, color=['skyblue',
'lightcoral', 'lightgreen'])
axes[1, 0].set_title("Thompson Sampling: Arm Selection Frequency")
axes[1, 0].set_xlabel("Teaching Method")
axes[1, 0].set_ylabel("Number of Selections")
axes[1, 0].tick_params(axis='x', rotation=45)

x = np.linspace(0, 1, 100)
for i in range(NUM_ARMS):

    axes[1, 1].axvline(TRUE_PROBABILITIES[i], color=['blue', 'red',
'green'][i], linestyle='--', label=f'True Prob {arm_names[i].split("
")[0]}')

    a_ts = thompson_results.alphas[i]
    b_ts = thompson_results.betas[i]
    axes[1, 1].plot(x, beta.pdf(x, a_ts, b_ts), color=['blue', 'red',
'green'][i], label=f'TS Posterior {arm_names[i].split(" ")[0]}')

axes[1, 1].set_title("Thompson Sampling: Posterior Distributions vs.
True Probabilities")
axes[1, 1].set_xlabel("Probability")
axes[1, 1].set_ylabel("Probability Density")
axes[1, 1].legend()
```

```
axes[1, 1].grid(True)


plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

print("\n--- Comparison ---")
print(f"True Probabilities: {TRUE_PROBABILITIES}")

print("\nUCB Results:")
for i in range(NUM_ARMS):
    print(f"  Arm {i} ({arm_names[i]}): Pulled
{ucb_results.arm_counts[i]} times, Avg Reward =
{ucb_results.arm_rewards[i] / ucb_results.arm_counts[i] if
ucb_results.arm_counts[i] > 0 else 0:.4f}")
print(f"  Total Cumulative Reward (UCB):
{ucb_results.cumulative_rewards[-1]}")
print(f"  Best Arm identified by UCB (based on highest final average
reward): Arm {np.argmax(ucb_results.arm_rewards /
ucb_results.arm_counts)}")

print("\nThompson Sampling Results:")
for i in range(NUM_ARMS):
    print(f"  Arm {i} ({arm_names[i]}): Pulled
{thompson_results.arm_counts[i]} times, Avg Reward =
{thompson_results.arm_rewards[i] / thompson_results.arm_counts[i] if
thompson_results.arm_counts[i] > 0 else 0:.4f}")
print(f"  Total Cumulative Reward (Thompson Sampling):
{thompson_results.cumulative_rewards[-1]}")
print(f"  Best Arm identified by Thompson Sampling (based on highest
final average reward): Arm {np.argmax(thompson_results.arm_rewards /
thompson_results.arm_counts)}")
```

Output:

```
Simulating UCB...
Simulating Thompson Sampling...
```



```
--- Comparison ---
True Probabilities: [0.6, 0.4, 0.7]

UCB Results:
  Arm 0 (Video Lectures (0.6)): Pulled 215.0 times, Avg Reward = 0.5488
  Arm 1 (Interactive Quizzes (0.4)): Pulled 97.0 times, Avg Reward = 0.3711
  Arm 2 (Gamified Modules (0.7)): Pulled 688.0 times, Avg Reward = 0.7093
  Total Cumulative Reward (UCB): 642
  Best Arm identified by UCB (based on highest final average reward): Arm 2

Thompson Sampling Results:
  Arm 0 (Video Lectures (0.6)): Pulled 15.0 times, Avg Reward = 0.4000
  Arm 1 (Interactive Quizzes (0.4)): Pulled 22.0 times, Avg Reward = 0.3636
  Arm 2 (Gamified Modules (0.7)): Pulled 963.0 times, Avg Reward = 0.6947
  Total Cumulative Reward (Thompson Sampling): 683
  Best Arm identified by Thompson Sampling (based on highest final average reward): Arm 2
```