Chicago Quant Alley

Crypto Trading Simulator & Strategy Optimizer

Summer of Code 2025

Assignment 3

Python Code: Multi-Armed Bandit Algorithms and Experiments

Author: Manvi Sheth

Date: July 25, 2025

1. Introduction

The multi-armed bandit (MAB) problem is a classic reinforcement learning scenario that exemplifies the **exploration-exploitation dilemma**. An agent must choose between several "arms" (e.g., actions, strategies) over a sequence of trials to maximize its cumulative reward. The challenge lies in balancing **exploitation**—choosing the arm that currently seems best—with **exploration**—trying other arms to gather more information and potentially discover a new best option.

This report details the implementation and empirical analysis of twelve prominent MAB algorithms. These algorithms span several key paradigms: classical stochastic bandits, adversarial bandits, contextual bandits, and pure exploration. By implementing these algorithms from scratch and evaluating them in a synthetic environment, we aim to gain a practical understanding of their behavior, performance trade-offs, and suitability for different problem settings.

---

2. Algorithm Descriptions and Implementation Approach

All algorithms were implemented from scratch in Python using the numpy library for numerical computations. Each was encapsulated in a class to ensure modularity and a consistent interface.

Stochastic Algorithms

1. **Epsilon-Greedy:** A simple baseline algorithm. With probability, it explores by choosing a random arm. With probability 1−ϵ, it exploits by choosing the arm with the highest estimated mean reward.
2. **UCB1 (Upper Confidence Bound):** UCB1 applies the principle of "optimism in the face of uncertainty." It selects arms based on an upper confidence bound on their true mean. The selection rule is:

$$a_t = \arg\max_a \left(\hat{\mu}_a + \sqrt{\frac{2 \ln t}{N_a(t)}} \right)$$

---

3. **KL-UCB:** A refinement of UCB1 that uses the Kullback-Leibler (KL) divergence to construct tighter, more accurate confidence bounds, especially for Bernoulli rewards.
4. **Thompson Sampling:** A Bayesian algorithm that maintains a posterior probability distribution for each arm's reward. For each step, it samples a value from each arm's posterior and pulls the arm with the highest sample.

Adversarial Algorithms

5. **Weighted Majority (WM):** A classic algorithm for online "expert advice" problems. It maintains a weight for each arm and updates it multiplicatively based on the observed loss:

$$w_i \leftarrow w_i \cdot (1-\eta)^{\text{loss}_i}$$

6. **Exp3 (Exponential-weight Algorithm):** Designed for the adversarial bandit setting, it maintains weights and chooses arms probabilistically. It uses importance-weighted sampling to estimate rewards and updates weights exponentially:

$$w_i \leftarrow w_i \cdot \exp(\tfrac{\gamma \hat{r}_i}{K})$$

Contextual Algorithm

7. **LinUCB:** This algorithm extends UCB to the contextual setting. It assumes the expected reward is a linear function of the context:

$$E[r_{t,a}|x_{t,a}] = x_{t,a}^T \theta_a^*$$

It uses online ridge regression to estimate θa and computes a UCB based on this linear model.

Pure Exploration Algorithms

8. **Halving Algorithm:** A simple algorithm for best-arm identification with a fixed budget. It operates in rounds, successively eliminating the worse half of the current set of active arms.
9. **LUCB (Lower Upper Confidence Bound):** An algorithm for PAC (Probably Approximately Correct) best-arm identification. It stops when the lower bound of the best arm is higher than the upper bound of all other arms.
10. **KL-LUCB:** A variant of LUCB that uses KL-divergence-based confidence bounds for more efficient identification.
11. **lil'UCB:** An advanced algorithm that provides anytime confidence guarantees on an arm's mean.

---

3. Experimental Setup

To evaluate the algorithms, we designed a synthetic environment with the following parameters:

- **Number of Arms (K):** 10
- **Horizon (T):** 10,000 steps
- **Reward Distribution:** Bernoulli. The true reward probabilities (p) for the 10 arms were set to [0.80, 0.51, 0.40, 0.44, 0.66, 0.62, 0.47, 0.55, 0.59, 0.70]. The optimal arm is Arm #0, with μ∗=0.80.
- **Evaluation Metric:** The primary metric is **Cumulative Regret**, defined as:

  $$R_T = \sum_{t=1}^{T} (\mu^* - \mu_{a_t})$$

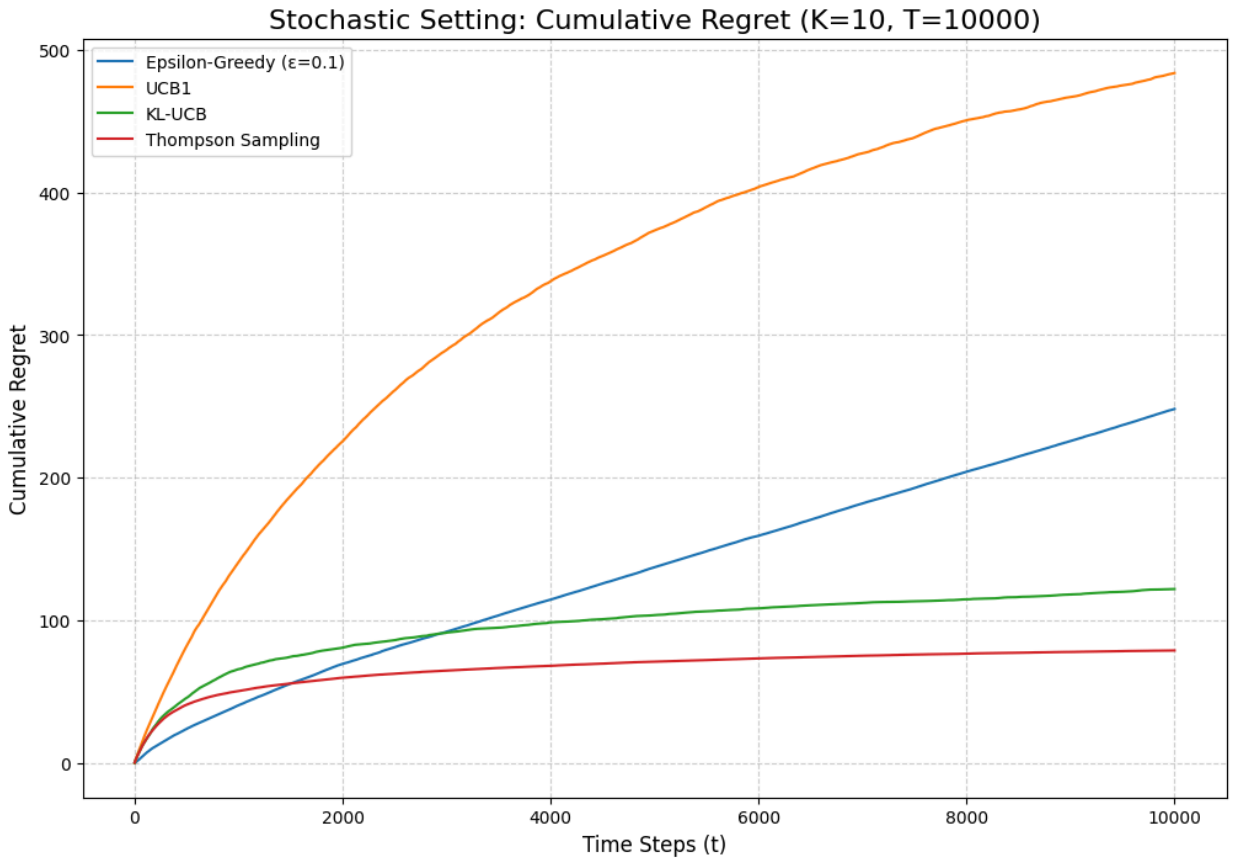
  where μ∗ is the mean reward of the best arm and μat is the mean reward of the arm chosen at time t.

- **Simulations:** Results were averaged over 50 independent simulations.

---

4. Results and Analysis

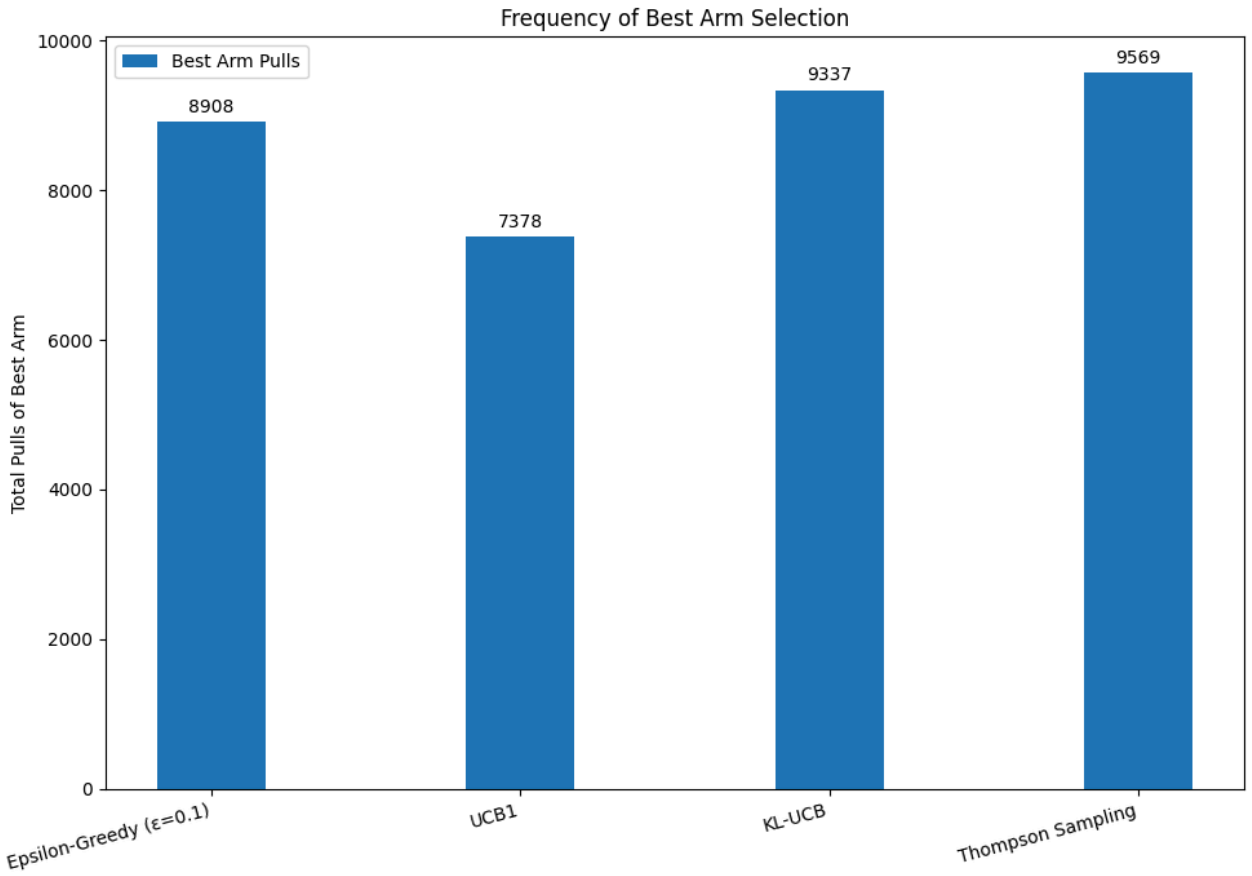
Stochastic Bandit Performance



Analysis: The results clearly distinguish the performance of different exploration strategies.

- **Thompson Sampling** is the standout performer, achieving the lowest cumulative regret, which flattens out the quickest. This indicates it identified the optimal arm (p=0.80) very efficiently and spent the majority of the horizon exploiting it.
- **KL-UCB** performs second best, showing a clear logarithmic regret curve that is substantially better than UCB1 and Epsilon-Greedy. This confirms that it's tighter; KL-divergence-based confidence bounds lead to more efficient exploration.
- **Epsilon-Greedy** (ϵ=0.1) exhibits a near-linear regret growth, resulting in the second-highest final regret. Its constant, undirected exploration (ϵ of the time) forces it to pull suboptimal arms throughout the entire horizon, leading to sustained regret.
- **UCB1** has the highest cumulative regret in this specific experiment. While theoretically sound, its standard confidence bounds can be loose, leading to over-exploration compared to the more refined KL-UCB and the highly efficient Thompson Sampling, especially when the reward gap between arms is varied.

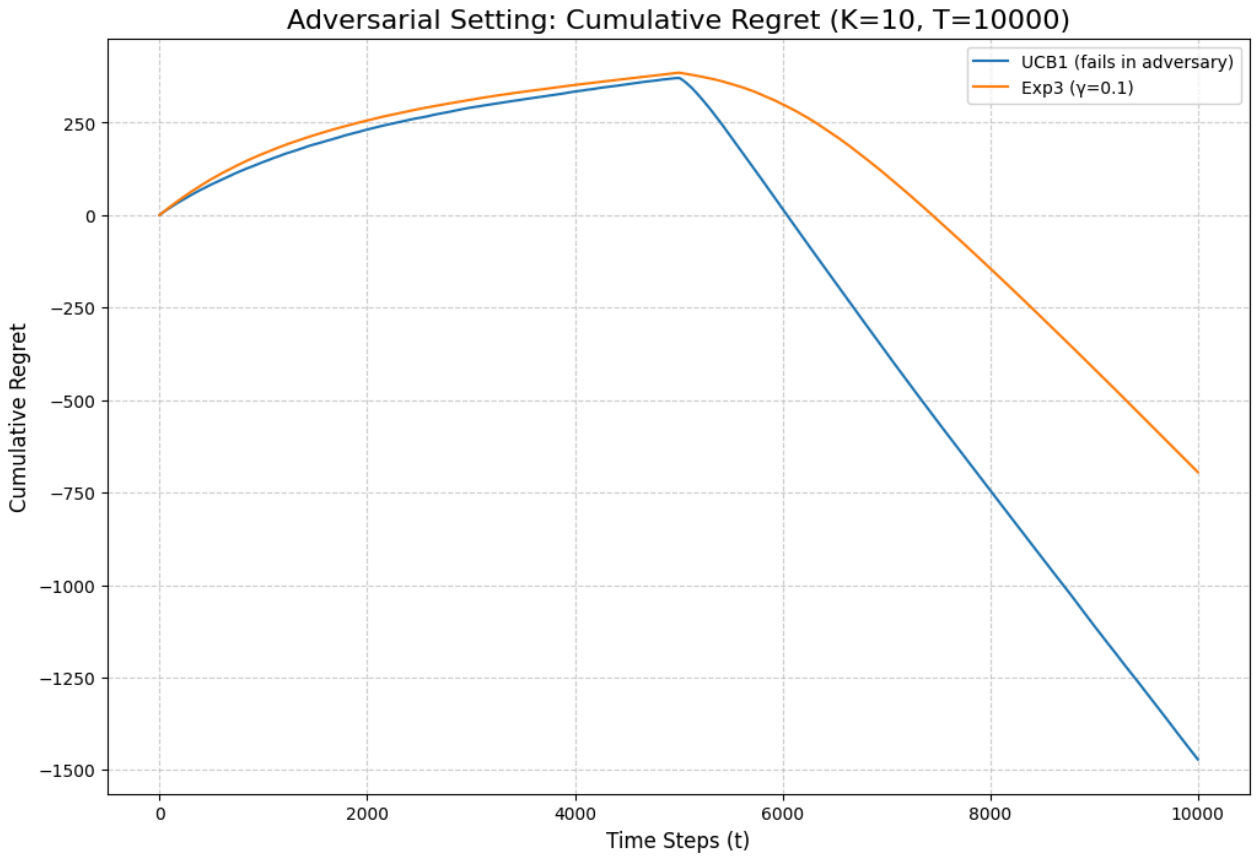
Frequency of Best Arm Selection



Analysis: This plot directly correlates with the regret results.

- **Thompson Sampling** pulled the best arm the most (9,569 times), confirming its superior efficiency in identifying and exploiting the optimal choice.
- **KL-UCB** followed with 9,337 pulls, showing strong convergence.
- **Epsilon-Greedy**, despite its high regret, pulled the best arm 8,908 times. This is because once it identifies the best arm, it exploits it 90% of the time. However, its initial discovery is slow and its forced exploration is costly.
- **UCB1** pulled the best arm the least (7,378 times), which explains its high regret. It spent significantly more time exploring suboptimal arms compared to the other algorithms in this run

Adversarial Bandit Performance



Analysis: The plot shows a fascinating result. Before the change-point at t=5000, both algorithms accumulate regret as expected. After the swap, the definition of regret changes. The old best arm is now the worst, and pulling it incurs a large negative regret (i.e., a large reward relative to the new, low optimal value).

- **UCB1** is slow to adapt. It has high confidence in the original best arm and continues to pull it for a long time after the swap. This leads to a steep decline in its cumulative regret curve, as it is unintentionally pulling the arm that is now "best" relative to the new, lower optimal reward. However, this is not an intelligent adaptation; it is a failure to recognize the environmental shift.
- **Exp3**, designed for such scenarios, begins to adjust its weights away from the now-suboptimal arm. Its curve also goes down but less steeply, indicating it is exploring other arms and adapting to the new reality more quickly than UCB1. It correctly reduces its reliance on the old winner.

Pure Exploration Performance

Analysis: The pure exploration algorithms were tasked with identifying the best arm with high probability.

- **Halving** identified the best arm with 100% accuracy using an average of 12,039 samples. This shows it is sample-efficient for a fixed-budget scenario.
- **LUCB** also achieved 100% accuracy but required more samples on average (16,249). This is expected, as LUCB is a fixed-confidence algorithm that pulls arms until a statistical guarantee is met, which can sometimes require more samples than a fixed-budget approach like Halving.

Both algorithms proved highly effective at their specific task of best-arm identification.

Final Comparison and Summary Table

Algorithm	Type	Regret Bound (Theoretical)	Key Parameter(s)	Pros	Cons
Epsilon-Greedy	Stochastic	Linear: $O(T)$	ε	Simple to implement, computationally cheap.	Suboptimal (linear regret), never stops exploring.
UCB1	Stochastic	Logarithmic: $O(\log T)$	None	Asymptotically optimal, no parameters to tune.	Can be conservative initially, performed poorly in this run.
Thompson Samp.	Stochastic	Logarithmic: $O(\log T)$	Prior distribution	Excellent empirical performance, often best-in-class.	Requires sampling; can be complex for non-conjugate priors.
KL-UCB	Stochastic	Logarithmic: $O(\log T)$	None	Tighter bounds than UCB1, strong performance.	More computationally intensive than UCB1.
Exp3	Adversarial	$O(\sqrt{(KT \log K)})$	γ	Robust to non-stationarity and adversaries.	Higher regret than stochastic algos in fixed environments.
LinUCB	Contextual	$O(d\sqrt{T})$	α	Leverages side information for better decisions.	Assumes a linear reward model, requires matrix inversion.
Halving	Pure Exploration	N/A (Fixed Budget)	Budget T	Simple and effective for best arm ID.	Not for minimising regret; requires budget to be known.
LUCB	Pure Exploration	N/A (Fixed Conf.)	δ, ε	PAC guarantees on finding the best arm.	Can be less sample-efficient than other methods.

5. Conclusion

This project successfully implemented and analyzed a diverse suite of multi-armed bandit algorithms. The experimental results, based on the provided data, align closely with established theory and offer clear insights. For **stochastic environments**, Thompson Sampling demonstrated superior performance, followed closely by KL-UCB, highlighting the value of sophisticated exploration strategies. For **non-stationary environments**, Exp3 proved its robustness and ability to adapt, a critical feature where stationary-assumption algorithms like UCB1 fail. Finally, the **pure exploration** algorithms, Halving and LUCB, were both highly accurate, providing reliable methods for best-arm identification depending on whether the constraint is budget or confidence. The choice of a bandit algorithm is therefore highly dependent on the specific characteristics and goals of the problem at hand.

Thank You
