Regret Bounds and the UCB Algorithm: A Theoretical Analysis

By Taha Majlesi

July 17, 2025

Abstract

This document provides a comprehensive theoretical analysis of the Upper Confidence Bound (UCB) algorithm within the framework of the Multi-Armed Bandit (MAB) problem. We begin by formalizing the MAB problem as a model for sequential decision-making, highlighting the fundamental exploration-exploitation dilemma. We then introduce the concept of regret as the primary metric for evaluating algorithm performance, aiming for sublinear regret growth as a hallmark of effective learning. The core of the analysis is a detailed examination of the UCB algorithm, which is guided by the principle of "Optimism in the Face of Uncertainty." We derive its action-selection formula by leveraging the Chernoff-Hoeffding inequality, a powerful tool from concentration inequalities. The report culminates in a rigorous, step-by-step proof of the UCB algorithm's regret bound, demonstrating that it achieves sublinear regret of the order $O(\sqrt{mT \log T})$. Finally, we discuss the implications of this theoretical guarantee and briefly touch upon extensions to more complex scenarios like non-stationary and contextual bandits.

1 The Multi-Armed Bandit Problem: A Framework for Sequential Decision-Making

The Multi-Armed Bandit (MAB) problem serves as a canonical model within the field of reinforcement learning for studying sequential decision-making under uncertainty. Its name evokes the vivid image of a gambler facing a row of slot machines (or "one-armed bandits"), each with a different, unknown probability of paying out a reward. The gambler's objective is to devise a strategy for pulling the arms—deciding which machine to play, in what order, and how many times—to maximize their total winnings over a series of plays. This seemingly simple scenario elegantly encapsulates a fundamental challenge that permeates not only machine learning but also numerous real-world domains: the trade-off between exploiting known good options and exploring new ones to discover potentially better alternatives.

1.1 The Fundamental Dilemma: Exploration vs. Exploitation

At the heart of the MAB problem lies a core tension known as the **exploration-exploitation dilemma**. This trade-off is a fundamental aspect of any learning system that must make decisions based on incomplete information while simultaneously gathering new data to improve future decisions.

- **Exploitation** is the strategy of leveraging existing knowledge to maximize immediate reward. It involves choosing the action that, based on past experience, is believed to be the best one right now. Consider the analogy of choosing a restaurant in a new city where you plan to stay for an extended period. If you find a restaurant that provides a satisfying meal on your first few visits, the exploitation strategy would be to return to that same restaurant every day. This approach guarantees a known, satisfactory outcome, thereby maximizing your short-term dining satisfaction based on the information you have already acquired.

- **Exploration**, in contrast, is the strategy of gathering new information. It involves trying different, less-known actions with the aim of discovering options that might yield superior rewards in the long run. In the restaurant analogy, exploration means trying a new restaurant you have never visited before. This action carries an immediate risk; the new restaurant could be a disappointment, resulting in a suboptimal meal for that day. However, it also carries the potential for a significant long-term gain: you might discover a new favorite restaurant that is far better than the one you were previously exploiting, leading to a much higher cumulative satisfaction over your entire stay.

The dilemma arises because these two strategies are inherently in conflict. An agent cannot explore and exploit with the same action at the same time. Over-emphasizing exploitation can lead to a suboptimal state where the agent gets stuck with a "good enough" option, forever missing out on the truly optimal one. Conversely, over-emphasizing exploration can be wasteful. The agent might spend too much time trying inferior options, accumulating low rewards and failing to capitalize on the knowledge it has gained about better choices.

1.2 Formalizing the Problem: Arms, Rewards, and the Horizon

To analyze the MAB problem with mathematical rigor, it is essential to establish a formal framework. The stochastic multi-armed bandit problem is typically defined as a game played over a series of discrete time steps.

The key components of this formalization are:

- **Arms (Actions):** There is a finite set of m possible actions, commonly referred to as "arms." Let this set be denoted by $\mathcal{A} = \{1, 2, \ldots, m\}$.

- **Time Horizon:** The game is played for a total of T rounds, where T is the time horizon. At each time step $t \in \{1, 2, \ldots, T\}$, the decision-making agent selects one arm to pull, denoted as $a_t \in \mathcal{A}$.

- **Rewards:** Upon pulling arm a_t at time t, the agent receives a numerical reward, r_t. This reward is a random variable drawn from a probability distribution ν_a that is specific to the chosen arm a_t. These reward distributions are initially unknown to the agent.

- **Stationarity Assumption (i.i.d. Rewards):** A critical assumption in the standard MAB formulation is that the rewards for each arm are independent and identically distributed (i.i.d.). This means that the reward r_t obtained from pulling arm a_t is drawn from the fixed distribution ν_{a_t}, independent of all previous actions and rewards.

1.3 Context and Applications

Despite its simplified structure, the MAB framework has proven to be remarkably effective for modeling a wide array of real-world problems.

- **Online Advertising and Content Curation:** News websites and advertising platforms use bandit algorithms to decide which content or ads to display to users.

- **Healthcare and Clinical Trials:** MAB algorithms offer a powerful paradigm for adaptive clinical trials, dynamically allocating patients to more promising treatments.

- **Recommender Systems:** E-commerce sites and streaming services use bandit algorithms to personalize recommendations and solve the "cold-start" problem.

- **Dynamic Pricing and Resource Allocation:** Companies can use bandit algorithms to optimize pricing strategies or allocate computational resources.

2 Quantifying Performance: The Concept of Regret

To evaluate and compare the performance of different MAB algorithms, the standard measure used is **regret**, which quantifies the cumulative opportunity cost incurred by the agent for not knowing the optimal action from the very beginning.

Table 1: Glossary of Notation

Symbol	**Description**
T	The total time horizon or number of rounds.
m	The total number of arms (actions).
a	A specific arm or action.
a_t	The action chosen by the algorithm at time step t.
a^*	The single best action, i.e., the one with the highest true expected reward.
r_t	The stochastic reward received at time step t.
$Q(a)$	The true, unknown expected reward of arm a (Action-Value).
V^*	The true expected reward of the optimal arm, $Q(a^*)$ (Optimal Value).
$n_t(a)$	The number of times arm a has been pulled up to the beginning of time step t.
$\hat{Q}_t(a)$	The empirical mean (sample average) of rewards from arm a based on $n_t(a)$ pulls.
$U_t(a)$	The Upper Confidence Bound estimate for arm a at time t.
δ	A confidence parameter, typically small, controlling the probability of failure.
R_T	The total cumulative regret over the horizon T.
Δ_a	The sub-optimality gap for a non-optimal arm a, defined as $V^* - Q(a)$.

2.1 Essential Terminology: Action-Value and Optimal Value

- **Action-value $Q(a)$:** For each arm a, its action-value, denoted $Q(a)$, is its true, unknown expected reward. Formally, it is the conditional expectation of the reward r given the action a:

  $$Q(a) = \mathbb{E}[r|a]$$

  This is the ground-truth value that the learning algorithm seeks to estimate for each arm.

- **Optimal value V^*:** The optimal value, denoted V^* (or sometimes μ^*), is the action-value of the single best arm. If we denote the optimal arm as a^*, then V^* is the maximum possible expected reward achievable in any single round. It is defined as:

  $$V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$$

2.2 Defining Regret: The Cumulative Opportunity Cost

Regret measures the total loss accumulated over the time horizon T due to the algorithm's failure to select the optimal arm a^* at every step. The total cumulative regret, denoted R_T, is the sum of these instantaneous regrets over all time steps.

$$R_T = \sum_{t=1}^{T} (V^* - Q(a_t)) = T \cdot V^* - \sum_{t=1}^{T} Q(a_t)$$

The goal of a well-designed bandit algorithm is to minimize this cumulative regret.

2.3 The Goal of Learning: An Analysis of Sublinear Regret

The behavior of the cumulative regret R_T as the time horizon T grows provides a rigorous way to classify the learning performance of an algorithm.

- **Linear Regret:** An algorithm has linear regret if $R_T = O(T)$. This implies that the average regret per time step, R_T/T, converges to a non-zero constant. An algorithm with linear regret is not truly learning, as it continues to make suboptimal choices at a roughly fixed rate.
- **Sublinear Regret:** An algorithm achieves the goal of learning if its cumulative regret grows slower than linearly with T, i.e., $R_T = o(T)$. Common sublinear regret bounds take forms like $O(\log T)$ or $O(\sqrt{T})$. The critical implication is that the average regret per step approaches zero as the time horizon grows:

  $$\lim_{T \to \infty} \frac{R_T}{T} = 0$$

  This property is the hallmark of a successful learning algorithm.

3 The Upper Confidence Bound (UCB) Algorithm

The Upper Confidence Bound (UCB) family of algorithms provides an elegant and effective solution to the exploration-exploitation dilemma based on the principle of **”Optimism in the Face of Uncertainty”**.

3.1 The UCB Principle and Action-Selection Strategy

The core mechanism of the UCB algorithm is as follows:

1. For each arm a at each time step t, compute an Upper Confidence Bound, $U_t(a)$, such that the true mean reward $Q(a)$ is less than or equal to $U_t(a)$ with high probability.
2. At each time step t, select the arm that has the highest upper confidence bound:

   $$a_t = \arg\max_{a \in \mathcal{A}} U_t(a)$$

This simple rule naturally balances exploration and exploitation.

3.2 Deconstructing the UCB Formula

The specific UCB formula analyzed here is a variant of the popular UCB1 algorithm:

$$U_t(a) = \underbrace{\hat{Q}_t(a)}_{\text{Exploitation Term}} + \underbrace{\sqrt{\frac{\log(t^2/\delta)}{n_t(a)}}}_{\text{Exploration Bonus}}$$

This formula can be broken down into two key components:

- $\hat{Q}_t(a)$ **(The Exploitation Term):** This is the empirical mean reward of arm a. It drives exploitation by favoring arms with high observed average rewards.
- **The Exploration Bonus:** This term quantifies the uncertainty in our estimate $\hat{Q}_t(a)$. Its behavior is governed by:
  - $n_t(a)$: The number of times arm a has been pulled. As $n_t(a)$ increases, the bonus shrinks, reducing exploration for well-understood arms.
  - t: The total number of rounds. As t increases, the bonus grows (slowly, due to the logarithm), ensuring that no arm is permanently abandoned.

3.3 A Comparative Perspective: UCB vs. Naive Strategies

- **Greedy Algorithm:** Forgoes exploration entirely, making it highly susceptible to locking onto a suboptimal arm and suffering linear regret.
- **ϵ-Greedy Algorithm:** Introduces random exploration with probability ϵ. Its exploration is undirected and inefficient compared to UCB. A fixed ϵ leads to linear regret.
- **UCB Algorithm:** Improves upon these methods by making exploration intelligent and directed, using confidence bounds to systematically reduce uncertainty where it is highest.

4 Probabilistic Foundations: The Chernoff-Hoeffding Inequality

The "optimism" of the UCB algorithm is grounded in rigorous probability theory, specifically concentration inequalities.

4.1 Concentration Inequalities in Machine Learning

Concentration inequalities provide bounds on the probability that a random variable deviates from its expected value. They are fundamental to statistical learning theory, as they justify making inferences from finite samples. They answer the question: "Given a sample average, how confident can we be that it is close to the true average?"

4.2 The Chernoff-Hoeffding Bound: Formal Statement

The Chernoff-Hoeffding inequality applies to sums of bounded, independent random variables. [Chernoff-Hoeffding Inequality] Let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables bounded in $[0, 1]$. Let $\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ be the sample mean and $\mathbb{E}[X]$ be the true mean. For any $u > 0$, the following one-sided bound holds:

$$P(\mathbb{E}[X] > \bar{X}_n + u) \leq \exp(-2nu^2)$$

Intuition: This inequality guarantees that the probability of the sample mean underestimating the true mean by more than u decays exponentially fast with the number of samples n and the square of the deviation u^2.

4.3 Application: Deriving the UCB Confidence Term

The UCB algorithm leverages this inequality to construct its confidence term in four logical steps.

1. **Re-interpret the Bound for the Bandit Problem:** We map the variables to the MAB problem: X_i are the rewards from arm a, $\mathbb{E}[X]$ is $Q(a)$, $\bar{X}_n$ is $\hat{Q}_t(a)$, and n is $n_t(a)$. The inequality becomes:

$$P(Q(a) > \hat{Q}_t(a) + u) \leq \exp(-2n_t(a)u^2)$$

2. **Set a Desired Failure Probability:** We want the probability of our bound being wrong to be very small. We set this failure probability to a value that decreases with time:

$$\exp(-2n_t(a)u^2) = \frac{\delta}{t^2}$$

Here, δ is a small confidence parameter. The t^{-2} form is chosen because the series $\sum t^{-2}$ converges, which is crucial for the final proof.

3. **Solve for the Deviation u:** We now solve for u, which will become our exploration bonus.

$$\begin{aligned} -2n_t(a)u^2 &= \ln\left(\frac{\delta}{t^2}\right) \\ 2n_t(a)u^2 &= \ln\left(\frac{t^2}{\delta}\right) \\ u^2 &= \frac{\ln(t^2/\delta)}{2n_t(a)} \\ u &= \sqrt{\frac{\ln(t^2/\delta)}{2n_t(a)}} \end{aligned}$$

The presentation simplifies this to $u = \sqrt{\frac{\log(t^2/\delta)}{n_t(a)}}$, likely by absorbing the constant factor of 2 into the definition of δ for clarity. We will adhere to the formula as presented.

4. **Construct the Upper Confidence Bound:** The final UCB estimate is the sum of the empirical mean and this confidence term:

$$U_t(a) = \hat{Q}_t(a) + u = \hat{Q}_t(a) + \sqrt{\frac{\log(t^2/\delta)}{n_t(a)}}$$

5 A Rigorous Analysis of UCB: Derivation of the Regret Bound

This section provides a step-by-step reconstruction of the regret bound proof for the UCB algorithm.

5.1 Bounding the Instantaneous Regret

The proof begins by decomposing the total regret:

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

We analyze the regret conditioned on the "good event," which is the event that all of our UCB estimates are simultaneously valid for all arms and at all time steps, i.e., $Q(a) \leq U_t(a)$ for all a, t. Under this assumption, we can bound the regret. By the UCB selection rule, $U_t(a_t) \geq U_t(a^*)$. In the "good event," we also have $Q(a^*) \leq U_t(a^*)$. Combining these gives:

$$Q(a^*) \leq U_t(a^*) \leq U_t(a_t)$$

This implies that $Q(a^*) - U_t(a_t) \leq 0$. Now, we can bound the regret by adding and subtracting $U_t(a_t)$:

$$\begin{aligned} R_T &= \sum_{t=1}^{T} (Q(a^*) - U_t(a_t) + U_t(a_t) - Q(a_t)) \\ &= \sum_{t=1}^{T} \underbrace{(Q(a^*) - U_t(a_t))}_{\leq 0} + \sum_{t=1}^{T} (U_t(a_t) - Q(a_t)) \\ &\leq \sum_{t=1}^{T} (U_t(a_t) - Q(a_t)) \end{aligned}$$

The problem is now reduced to bounding this new sum.

5.2 Probabilistic Consistency: Ensuring the "Good Event"

We must show that the "good event" holds with high probability. The probability of failure for a single arm i at a single time step t is $P(Q(i) > U_t(i)) \leq \delta/t^2$. The "bad event" is that at least one bound fails. Using the union bound across all m arms and all T time steps:

$$P(\text{any bound fails}) \leq \sum_{t=1}^{T} \sum_{i=1}^{m} P(Q(i) > U_t(i)) \leq \sum_{t=1}^{T} \sum_{i=1}^{m} \frac{\delta}{t^2} = m\delta \sum_{t=1}^{T} \frac{1}{t^2}$$

For an infinite horizon, this is bounded because the series converges: $\sum_{t=1}^{\infty} \frac{1}{t^2} = \frac{\pi^2}{6} < 2$.

$$P(\text{any bound ever fails}) < 2m\delta$$

Thus, the "good event" holds with probability at least $1 - 2m\delta$.

5.3 Summation of Bounds: The Path to the Final Regret Formula

We now bound the remaining sum, $\sum_{t=1}^{T} (U_t(a_t) - Q(a_t))$.

1. **Bound the summand:** With high probability, the estimation error $|\hat{Q}_t(a_t) - Q(a_t)|$ is also bounded by the confidence width. This leads to:

$$U_t(a_t) - Q(a_t) = (\hat{Q}_t(a_t) - Q(a_t)) + \sqrt{\frac{\log(t^2/\delta)}{n_t(a_t)}} \leq 2\sqrt{\frac{\log(t^2/\delta)}{n_t(a_t)}}$$

   So, $R_T \leq \sum_{t=1}^{T} 2\sqrt{\frac{\log(t^2/\delta)}{n_t(a_t)}}$.

2. **Replace the time-dependent logarithm:** Since $t \leq T$, we have $\log(t^2/\delta) \leq \log(T^2/\delta)$. We can pull this constant factor out:

$$R_T \leq 2\sqrt{\log(T^2/\delta)} \sum_{t=1}^{T} \frac{1}{\sqrt{n_t(a_t)}}$$

3. **Regroup the sum by arms:** Let $n_i(T)$ be the total pulls of arm i. The sum over time can be rewritten as a sum over arms and their pull counts:

$$\sum_{t=1}^{T} \frac{1}{\sqrt{n_t(a_t)}} = \sum_{i=1}^{m} \sum_{n=1}^{n_i(T)} \frac{1}{\sqrt{n}}$$

4. **Bound the inner sum:** The sum $\sum_{n=1}^{k} \frac{1}{\sqrt{n}}$ can be bounded by an integral: $\sum_{n=1}^{k} \frac{1}{\sqrt{n}} \leq \int_0^k \frac{1}{\sqrt{x}} dx = 2\sqrt{k}$.

$$\sum_{i=1}^{m} \sum_{n=1}^{n_i(T)} \frac{1}{\sqrt{n}} \leq \sum_{i=1}^{m} 2\sqrt{n_i(T)}$$

5. **Bound the sum over arms:** We need to bound $\sum_{i=1}^{m} \sqrt{n_i(T)}$. Using the Cauchy-Schwarz inequality with vectors $\mathbf{u} = (1, \ldots, 1)$ and $\mathbf{v} = (\sqrt{n_1(T)}, \ldots, \sqrt{n_m(T)})$:

$$\left(\sum_{i=1}^{m} \sqrt{n_i(T)}\right)^2 \leq \left(\sum_{i=1}^{m} 1^2\right)\left(\sum_{i=1}^{m} (\sqrt{n_i(T)})^2\right) = m\sum_{i=1}^{m} n_i(T) = mT$$

Taking the square root gives: $\sum_{i=1}^{m} \sqrt{n_i(T)} \leq \sqrt{mT}$.

6. **Combine all terms:** Substituting everything back:

$$\begin{aligned} R_T &\leq 2\sqrt{\log(T^2/\delta)} \left(\sum_{i=1}^{m} 2\sqrt{n_i(T)}\right) \\ &\leq 4\sqrt{\log(T^2/\delta)} \left(\sum_{i=1}^{m} \sqrt{n_i(T)}\right) \\ &\leq 4\sqrt{\log(T^2/\delta)}\sqrt{mT} \\ R_T &\leq 4\sqrt{mT\log(T^2/\delta)} \end{aligned}$$

5.4 Interpreting the Result: The Implications of a Sublinear Bound

The final regret bound is of the order $O(\sqrt{mT \log T})$.

- **Sublinear in T:** The regret grows with $\sqrt{T \log T}$, which is sublinear. This formally proves that UCB learns, as the average regret per round, $R_T/T \approx O(\sqrt{m \log T/T})$, approaches zero as $T \to \infty$.
- **Dependency on m:** The regret grows with $\sqrt{m}$. This is intuitive; problems with more arms are inherently harder.

6 Concluding Remarks and Future Directions

6.1 Summary of UCB and its Theoretical Guarantees

The UCB algorithm offers an elegant solution to the exploration-exploitation dilemma. Its principle, "Optimism in the Face of Uncertainty," is grounded in the Chernoff-Hoeffding inequality and leads to a deterministic, intelligent exploration strategy. The key result is a sublinear regret bound of $O(\sqrt{mT \log T})$, which formally proves that UCB is a true learning algorithm and quantifies its performance.

6.2 Beyond the Scope: Non-Stationary and Contextual Bandits

The analysis presented here relies on the assumption of a stationary environment. Many real-world applications violate this assumption, which has motivated more advanced bandit formulations.

- **Non-Stationary Bandits:** In environments where reward distributions change over time, standard UCB can fail. Algorithms like Discounted UCB and Sliding-Window UCB address this by giving more weight to recent observations, allowing them to adapt.

- **Contextual Bandits:** This is a significant extension where the agent receives side information, or "context," before making a decision. The goal is to learn a policy that maps contexts to optimal actions. Algorithms like LinUCB, which models the reward as a linear function of the context, are powerful tools for this more complex and practical setting.