Multi-Armed Bandit Problem and its Foundational Algorithms

Santhosh Kumar G

Tuesday 5th August, 2025

1 The Multi-Armed Bandit Problem: A Foundational Overview

1.1 Defining the Problem: The Gambler's Dilemma

The Multi-Armed Bandit (MAB) problem is a classic model in machine learning and probability theory that addresses sequential decision-making under uncertainty. The name originates from a gambling analogy: a player, or "agent," is faced with a row of slot machines, colloquially known as "one-armed bandits," each with a lever or "arm" to pull. The problem is a stylized representation of a broad class of challenges where a decision-maker must iteratively choose from a set of fixed options, or "arms," where the properties of each option are initially unknown.

The core components of a standard MAB problem are as follows:

- **K Arms:** A finite set of options, denoted as $\{1, 2, ..., K\}$, which the agent can choose to activate at each time step. Each arm is associated with a specific, independent, and unknown probability distribution from which rewards are drawn. For instance, a reward could be binary, such as 1 for a win and 0 for a loss, a scenario known as a Bernoulli Bandit problem.
- **An Agent:** The decision-maker who, at each discrete time step $t = 1, 2, ..., T$, must select and pull one of the arms, denoted as a_t.
- **Rewards:** After pulling an arm a_t, the agent observes a random reward r_t drawn from that arm's unknown reward distribution. The reward distribution does not change over time, a characteristic of a "stationary" bandit problem.
- **Time Horizon (T):** The total number of arm pulls or decisions to be made.

The ultimate objective of the agent is to devise a strategy that maximizes the total cumulative expected reward over the entire time horizon T. This is a fundamentally challenging problem because the agent does not possess prior knowledge of the reward distributions for any of the arms. A seemingly simple approach, such as always pulling the arm that has provided the highest reward in a single initial test pull, is prone to catastrophic failure. An arm that is, in fact, the most profitable might yield a low reward by chance during an initial trial, leading the agent to permanently disregard it. Therefore, the agent must develop a policy that not only utilizes its current knowledge but also actively seeks to improve that knowledge.

1.2 The Central Trade-off: Exploration vs. Exploitation

The fundamental challenge inherent in the MAB problem is the delicate balance between exploration and exploitation. This trade-off is central to a wide range of decision-making scenarios where a finite set of resources must be allocated among competing choices to maximize an expected gain.

- **Exploitation:** This is the act of leveraging existing knowledge to maximize immediate gain. In the context of the MAB problem, an agent practicing pure exploitation would, at every time step, pull the arm that has yielded the highest average reward up to that point. This strategy is focused on capitalizing on what is already known to be "good" or promising.
- **Exploration:** This is the act of gathering new information. It involves choosing an arm that has been pulled less frequently or has not performed as well in the past. The motivation for exploration is the possibility of discovering an arm with a higher true reward mean that was initially underestimated due to early chance outcomes.

A naive strategy of pure exploitation risks becoming trapped in a suboptimal local maximum, consistently pulling an arm that is good but not the best, and entirely missing out on a superior option. Conversely, a strategy of pure exploration, which involves randomly pulling arms without regard for past performance, would waste valuable opportunities to accumulate rewards by repeatedly choosing inferior arms. The core of the MAB problem is to design an algorithm that intelligently navigates this trade-off, shifting the balance from exploration to exploitation as more information is accumulated over time. The ultimate goal is to find a policy that maximizes the total reward by making informed, sequential decisions.

1.3 The Ultimate Metric: Maximizing Reward by Minimizing Regret

To quantify the performance of an MAB algorithm, a key metric known as "regret" is used. Regret provides a formal measure of how much a particular strategy falls short of the ideal, optimal strategy. It can be seen as the opportunity cost incurred during the learning process, capturing the cumulative loss from not having perfect knowledge from the outset.

Regret is defined as the difference between the best possible expected reward and the actual reward obtained by the agent after T time steps. The best possible expected reward is a hypothetical value achieved by always pulling the single arm with the highest true mean reward, which is unknown to the agent.

Mathematically, the cumulative regret $R(T)$ after T rounds is expressed as:

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

where μ^* represents the mean reward of the optimal arm (the arm with the highest true expected reward) and μ_{a_t} is the mean reward of the arm chosen by the agent at time step t.

A successful MAB algorithm does not aim for a cumulative regret of zero, which is impossible due to the initial uncertainty. Instead, a successful algorithm is one whose average regret per round, $R(T)/T$, asymptotically approaches zero as the number of time steps T grows infinitely large. This is typically achieved with a "sub-linear" cumulative regret, such as one that grows logarithmically, denoted as $O(\log T)$. A linear regret, $O(T)$, indicates a fundamentally flawed strategy that fails to converge on the optimal arm, leading to a fixed, non-zero average reward gap at every time step. Minimizing regret is mathematically equivalent to maximizing the total reward, as both metrics are two sides of the same coin.

2 Classic MAB Algorithms: From Naive to Probabilistic

2.1 The Greedy Approach: Simplicity and Sub-optimality

The simplest and most straightforward approach to the MAB problem is the greedy algorithm. This strategy is purely exploitative and makes decisions based solely on the current knowledge of past rewards. At each time step, the agent calculates the average reward for each arm pulled so far and then selects the arm with the highest average.

The fundamental flaw of the greedy algorithm is that it lacks any mechanism for exploration. If, by chance, an arm with a high true mean reward yields a low or zero reward during its initial pulls, its average reward will be low, and the algorithm will likely never select it again. The initial data can "poison" the agent's perception of a potentially optimal arm, trapping the agent in a suboptimal state for the remainder of the experiment. This failure to explore ensures that the algorithm's cumulative regret will grow linearly with the number of time steps, as it consistently makes suboptimal decisions after its initial misstep.

2.2 Epsilon-Greedy: Introducing Controlled Exploration

To address the fatal flaw of the greedy approach, the epsilon-greedy algorithm introduces a simple but effective mechanism to balance exploitation and exploration. This strategy incorporates a small, predefined probability, denoted as ϵ (epsilon), to govern the decision-making process at each time step.

The algorithm operates as follows:

- With a small probability ϵ (e.g., 0.1), the agent explores by selecting an arm uniformly at random from the set of K arms. This ensures that even seemingly poor-performing arms have a chance to be pulled, potentially revealing a hidden high reward.
- With the complementary probability $1 - \epsilon$, the agent exploits by selecting the arm with the highest estimated average reward from all the pulls up to that point.

The value of ϵ is a critical hyperparameter that directly controls the trade-off. A higher ϵ promotes more exploration, while a lower value favors exploitation. A common variant of this algorithm is to use a dynamic ϵ that decreases over time. By starting with a high ϵ to encourage initial exploration and then reducing it, the algorithm can

gradually shift its focus to exploitation as it gathers more data and gains confidence in its estimates. Despite its simplicity and effectiveness in many scenarios, the epsilon-greedy algorithm still has a limitation: it explores indiscriminately, treating all underperforming or less-pulled arms equally during the exploration phase. This can still lead to a linear regret over time, as exploration never fully ceases.

3 Upper Confidence Bound (UCB): The Principle of Optimism

3.1 Conceptual Foundation: Optimism in the Face of Uncertainty

The Upper Confidence Bound (UCB) algorithm represents a more sophisticated and principled approach to the exploration-exploitation trade-off. It is a deterministic, frequentist method based on the principle of "optimism in the face of uncertainty". The core idea is that, rather than making decisions based solely on the observed average reward, the algorithm should also consider the uncertainty surrounding that estimate. UCB prioritizes arms that have a high estimated reward or a high degree of uncertainty, with the assumption that the "true" mean of an under-explored arm might be much higher than its current, limited sample suggests.

The algorithm calculates a score for each arm by combining two key components: the exploitation term, which is the arm's average reward, and an exploration bonus, which is a measure of its uncertainty. At each time step, the agent simply selects the arm with the highest UCB score. The brilliance of this approach is that exploration is "baked in" to the decision-making process; the algorithm naturally and systematically explores options with the most potential to be better than the current best-known arm.

3.2 The UCB Formula and its Components

The UCB score for an arm a at time step t is calculated using the following formula:

$$UCB(a) = \text{average\_reward}(a) + \sqrt{\frac{2 \ln T}{n(a)}}$$

This formula captures the core tension between exploitation and exploration through its two distinct terms.

- **Exploitation Term (average_reward(a)):** This is the mean reward of arm a calculated from all its previous pulls. This term favors arms that have performed well in the past, driving the agent to exploit known good options.
- **Exploration Bonus ($\sqrt{\frac{2 \ln T}{n(a)}}$):** This term quantifies the uncertainty of the reward estimate for arm a. The numerator, $\ln T$, increases with time, ensuring that the exploration bonus for all arms, even those not yet pulled, continues to grow, thereby preventing any arm from being completely ignored over the long term. The denominator, $n(a)$, represents the number of times arm a has been pulled. As an arm is pulled more frequently, $n(a)$ increases, causing the exploration bonus to shrink.

This creates a self-correcting feedback loop: an arm is pulled because it has a high score, its count $n(a)$ increases, its uncertainty bonus decreases, and it becomes less likely to be chosen in subsequent rounds, which naturally directs the algorithm to explore other options with higher uncertainty.

The dynamic interplay between these two terms is what allows UCB to be so effective. It starts with a high exploration bonus for all arms, encouraging initial exploration, but as data is gathered for a particular arm, its bonus decreases, and the algorithm shifts its focus toward exploitation of the best-performing arms. This deterministic process provides a systematic way to balance the two competing objectives.

3.3 Reconstructing the UCB Example

The following example, reconstructed from the provided materials, illustrates the step-by-step operation of the UCB algorithm with three arms. The reward for each arm is binary, based on a Bernoulli distribution.

Time (t)	Arm Pulled (a_t)	Reward (r_t)	# Times Pulled	Total Reward	UCB (Arm 1)	UCB (Arm 2)	UCB (Arm 3)
			Arm 1				
			0	0	-	-	-
			Arm 2				
			0	0	-	-	-
			Arm 3				
			0	0	-	-	-
1	1	0	Arm 1: 1	0	-	-	-
2	2	0	Arm 2: 1	0	-	-	-
3	3	1	Arm 3: 1	1	-	-	-
4	3	0	Arm 3: 2	1	1.665	1.665	**2.665**
5	1	0	Arm 1: 2	0	**1.794**	1.794	1.769
6	2	0	Arm 2: 2	0	1.954	**1.954**	1.879

Table 1: UCB Example Walkthrough

Step-by-step logic:

- **Time steps 1-3:** The algorithm pulls each arm once to establish initial observations.
  - At $t = 1$, Arm 1 is pulled, and the reward is 0.
  - At $t = 2$, Arm 2 is pulled, and the reward is 0.
  - At $t = 3$, Arm 3 is pulled, and the reward is 1.
- **Time step 4:** The algorithm must now make a decision based on the UCB scores.
  - **Arm 1:** Average reward is $0/1 = 0$. UCB score is $0 + \sqrt{\frac{2 \ln 4}{1}} \approx 1.665$.
  - **Arm 2:** Average reward is $0/1 = 0$. UCB score is $0 + \sqrt{\frac{2 \ln 4}{1}} \approx 1.665$.
  - **Arm 3:** Average reward is $1/1 = 1$. UCB score is $1 + \sqrt{\frac{2 \ln 4}{1}} \approx 2.665$.
  - **Decision:** Arm 3 has the highest UCB score, so it is pulled. The observed reward is 0. The total reward for Arm 3 is now 1, and its number of pulls is 2.
- **Time step 5:** The UCB scores are recalculated based on the new data.
  - **Arm 1:** Average reward is still $0/1 = 0$. UCB score is now $0 + \sqrt{\frac{2 \ln 5}{1}} \approx 1.794$.
  - **Arm 2:** Average reward is still $0/1 = 0$. UCB score is now $0 + \sqrt{\frac{2 \ln 5}{1}} \approx 1.794$.
  - **Arm 3:** Average reward is now $1/2 = 0.5$. UCB score is $0.5 + \sqrt{\frac{2 \ln 5}{2}} \approx 1.769$.
  - **Decision:** Arm 1 and Arm 2 now have the highest UCB scores. The exploration bonus for Arm 3 has shrunk due to its second pull. The algorithm breaks the tie by choosing the lower index arm, Arm 1. The observed reward is 0.
- **Time step 6:** The process continues with a new set of UCB scores and a new decision. The number of pulls for Arm 1 has increased, causing its UCB score to decrease relative to Arm 2, which has a similar average reward. This systematic reduction in the uncertainty bonus ensures that the algorithm continually explores less-pulled arms, providing a robust mechanism for balancing exploration and exploitation.

4 Thompson Sampling: A Bayesian Perspective

4.1 Conceptual Foundation: Probability Matching

Thompson Sampling, also known as Bayesian Bandits, offers a powerful alternative to frequentist methods like UCB by adopting a Bayesian approach. The core principle of Thompson Sampling is "probability matching". Instead of relying on a deterministic formula, the algorithm maintains a probabilistic belief about the true mean reward of each arm, represented by a probability distribution. At each time step, the agent samples a

value from the belief distribution for each arm and then pulls the arm that yields the highest sampled value.

The exploration component of Thompson Sampling is a natural and inherent part of this process. Arms that have been pulled less frequently have a wider, more uncertain belief distribution. This wider distribution increases the chance of drawing a high random sample from it, thereby promoting its selection and ensuring that under-explored arms are not ignored. Conversely, arms that have been pulled many times have a tighter distribution, concentrated around their true mean, which makes it more likely for them to be selected only if they have a genuinely high mean reward.

4.2 The Bayesian Framework for Bernoulli Bandits

For Bernoulli Bandit problems, where rewards are binary (1 or 0), the Beta distribution is the ideal choice for modeling the belief about the unknown reward probability of each arm. This is because the Beta distribution is a *conjugate prior* for the Bernoulli distribution. This means that when an observation (a reward of 1 or 0) is made, the posterior distribution (the updated belief) remains a Beta distribution, simplifying the computational updates.

A Beta distribution is parameterized by two positive numbers, α and β, written as $\text{Beta}(\alpha, \beta)$. In the context of MAB, α corresponds to the number of successes plus one, and β corresponds to the number of failures plus one.

The algorithm is initialized by setting the prior belief for each arm to a Beta distribution with parameters $\alpha = 1$ and $\beta = 1$. The $\text{Beta}(1, 1)$ distribution is a uniform distribution over the interval $(0, 1)$, which represents a state of complete ignorance about the true reward probability of an arm—every value is considered equally likely. As the agent pulls an arm and observes a reward, the parameters are updated: if the reward is a success (1), α is incremented by 1; if it is a failure (0), β is incremented by 1. The Beta distribution's density function then shifts and tightens around the observed mean, reflecting the agent's updated belief.

4.3 Reconstructing the Thompson Sampling Example

The following example, reconstructed from the provided materials, demonstrates the step-by-step operation of the Thompson Sampling algorithm.

Time (t)	Arm Pulled (a_t)	Reward (r_t)	Arm 1	Arm 2	Arm 3	Sample (Arm 1)	Sample (Arm 2)	Sample (Arm 3)
			Successes, Failures (0,0) **Parameters** Beta$(1,1)$	(0,0) Beta$(1,1)$	(0,0) Beta$(1,1)$			
1	2	0	Beta$(1,1)$	Beta$(1,2)$	Beta$(1,1)$	0.43	**0.75**	0.11
2	3	1	Beta$(1,1)$	Beta$(1,2)$	Beta$(2,1)$	0.52	0.05	**0.67**
3	3	0	Beta$(1,1)$	Beta$(1,2)$	Beta$(2,2)$	0.44	0.27	**0.86**
4	1	1	Beta$(2,1)$	Beta$(1,2)$	Beta$(2,2)$	**0.63**	0.15	0.44

Table 2: Thompson Sampling Example Walkthrough

Step-by-step logic:

- **Initial State:** All three arms are initialized with a prior distribution of Beta$(1, 1)$, representing no prior knowledge.
- **Time step 1:**
  - The agent samples a random value from the Beta$(1, 1)$ distribution for each arm.
  - Arm 1 Sample: 0.43
  - Arm 2 Sample: 0.75
  - Arm 3 Sample: 0.11
  - **Decision:** The agent chooses the arm with the highest sample, which is Arm 2. It observes a reward of 0.
  - **Update:** The number of failures for Arm 2 is incremented, and its belief distribution is updated to Beta$(1, 2)$. The distributions for Arms 1 and 3 remain Beta$(1, 1)$.
- **Time step 2:**
  - The agent samples from the updated belief distributions.
  - Arm 1 Sample (from Beta$(1, 1)$): 0.52
  - Arm 2 Sample (from Beta$(1, 2)$): 0.05
  - Arm 3 Sample (from Beta$(1, 1)$): 0.67
  - **Decision:** The highest sample is from Arm 3, so it is pulled. A reward of 1 is observed.
  - **Update:** The number of successes for Arm 3 is incremented, and its distribution becomes Beta$(2, 1)$.
- **Time step 3:**
  - The agent samples again.
  - Arm 1 Sample (from Beta$(1, 1)$): 0.44
  - Arm 2 Sample (from Beta$(1, 2)$): 0.27
  - Arm 3 Sample (from Beta$(2, 1)$): 0.86
  - **Decision:** Arm 3 is chosen again due to its high sample. A reward of 0 is observed.
  - **Update:** The number of failures for Arm 3 is incremented, and its distribution becomes Beta$(2, 2)$. This distribution is now centered closer to a mean of 0.5 but is still relatively wide, reflecting the low number of total pulls.
- **Time step 4:** The process continues. Note that even though Arm 3 is performing well, there is still a chance that a less-pulled arm (like Arm 1) could yield a high sample from its wider distribution, leading to its selection. This illustrates how the probabilistic nature of the algorithm naturally handles exploration.

5 Comparative Analysis and Broader Context

5.1 Fundamental Differences: Frequentist vs. Bayesian

The UCB and Thompson Sampling algorithms represent two fundamentally different philosophical approaches to the MAB problem.

- **Upper Confidence Bound (UCB):** This is a frequentist approach. It is based on point estimates (average reward) and a statistical confidence interval. The UCB algorithm is entirely deterministic; given the same history of rewards, it will always make the same decision. It is based on a well-defined theoretical framework that provides strong guarantees on its regret bounds.
- **Thompson Sampling:** This is a Bayesian approach. It operates on a probabilistic belief system, where the agent maintains a distribution over the unknown parameters. The decision-making process is randomized; the same reward history can lead to different decisions because the choice depends on a random sample drawn from the posterior distribution.

5.2 Performance & Robustness

While UCB has well-established theoretical guarantees for its performance, Thompson Sampling has been shown to consistently outperform it in real-world simulations and applications. The reasons for this disparity highlight the practical advantages of the Bayesian approach.

The randomized nature of Thompson Sampling is often more robust to complexities found in real-world scenarios, such as delayed feedback. The algorithm is less likely to become "stuck" in a poor decision because the probabilistic sampling mechanism provides a continuous, organic form of exploration that is not reliant on a specific formula or a fixed exploration schedule. This natural exploration can be more efficient than UCB's explicit, formula-based bonus. For instance, in a large, dynamic system, a UCB algorithm might become too focused on a few arms, while Thompson Sampling's randomization is more likely to explore a broader range of options, thus discovering the truly optimal arm more quickly.

The visual comparison of the two algorithms often shows that Thompson Sampling's average regret curve drops more steeply than UCB's, indicating a faster convergence to the optimal solution. Similarly, plots of the proportion of times each arm is pulled over time show Thompson Sampling quickly moving most of its pulls to the single optimal arm, while UCB may take longer to fully abandon suboptimal arms.

5.3 Real-World Applications

The MAB problem is not merely a theoretical exercise; it is a foundational model with wide-ranging applications across various industries where decisions must be made under uncertainty.

- **Clinical Trials:** The MAB model is a natural fit for clinical trials, where the "arms" are different experimental treatments and the "rewards" are positive patient outcomes. An MAB algorithm can be used to dynamically allocate more patients

to the most effective treatments while minimizing the number of patients receiving less successful ones, thereby maximizing the total number of patients healed over the course of the trial.

- **Digital Marketing and Recommender Systems:** In online environments, MAB algorithms are used to optimize content, ads, and product recommendations. For example, a news website can use MAB to dynamically choose which article headlines to display to visitors to maximize click-through rates. Similarly, e-commerce platforms can apply the model to personalize product recommendations, and ad platforms can optimize ad placements by dynamically allocating traffic to the best-performing creative variants.

- **Other Applications:** The MAB framework also extends to adaptive network routing, where different network paths are the arms and delays are the rewards. In financial portfolio management, the arms can be different assets, and the goal is to maximize total returns. The model has even been applied to problems in game theory and computer game playing to explore possible continuations more efficiently by focusing on the most promising subtrees.

6 Conclusion

The Multi-Armed Bandit problem serves as a canonical framework for understanding the fundamental challenge of decision-making under uncertainty. The core tension between exploiting known information for immediate gain and exploring new options to discover better long-term opportunities is a dilemma that permeates various fields, from scientific research to modern data-driven systems.

While naive approaches like the greedy algorithm are fundamentally flawed, more sophisticated methods provide elegant solutions. The UCB algorithm, with its deterministic and optimistic principle, provides a systematic and provably effective way to navigate the exploration-exploitation trade-off. It balances its decisions by adding an uncertainty bonus to under-explored options, ensuring a thorough search for the optimal solution. In contrast, Thompson Sampling offers a more flexible, probabilistic, and often more robust solution. By maintaining a continuous belief distribution and making decisions based on random samples, it provides a natural and powerful mechanism for exploration that often outperforms UCB in practice, particularly in complex or non-stationary environments.

The study of these algorithms is not merely academic. The insights they provide are crucial for the development of real-time, data-driven systems that can learn and adapt to dynamic environments. From optimizing clinical trials to personalizing online experiences, the principles of MAB provide a foundation for designing intelligent agents that can maximize rewards by embracing the necessary uncertainty of the learning process.