A Comprehensive Analysis of Multi-Armed Bandit Algorithms

By Taha Majlesi

July 17, 2025

Contents

1 The Multi-Armed Bandit Problem: A Foundational Framework

The Multi-Armed Bandit (MAB) problem is a cornerstone model in reinforcement learning and sequential decision-making. It masterfully simplifies a complex learning environment to isolate a fundamental challenge: the trade-off between **exploration** (gathering new information) and **exploitation** (using existing information to maximize immediate gains).

1.1 Defining the K-Armed Bandit Problem

Imagine a gambler facing a row of 'k' slot machines, each with its own hidden payout probability. The gambler's goal is to devise a strategy to maximize their total winnings over a series of plays. This analogy perfectly captures the essence of the k-armed bandit problem.

Definition 1.1 (The K-Armed Bandit Problem)**.** *The problem is formally defined by:*

- ***An agent**: The decision-maker.*
- ***A set of k actions (arms)**: $A = \{1, 2, \ldots, k\}$.*
- ***A set of reward distributions**: $\{D_1, D_2, \ldots, D_k\}$, where each action a is associated with an unknown probability distribution D_a from which rewards are drawn.*

The agent's objective is to create a policy to select actions over a sequence of time steps, $t = 1, 2, \ldots, T$, to maximize the total cumulative reward.

The MAB problem is equivalent to a one-state Markov Decision Process (MDP). By removing state transitions, it allows for a focused study of the exploration-exploitation dilemma. The challenge is not navigating a changing environment, but efficiently learning the value of the single best action from noisy feedback.

1.2 The True Action-Value, q*(a): Quantifying the Optimal Choice

To make informed decisions, the agent must quantify the value of each action. This is done using the concept of the **true action-value**, denoted as $q_*(a)$.

Definition 1.2 (True Action-Value)**.** *The true action-value is the expected (mean) reward for selecting action a. It is formally defined as:*

$$q_*(a) \doteq \mathbb{E}[R_t | A_t = a] = \sum_r p(r|a) r \tag{1}$$

where R_t is the reward at time step t, A_t is the action taken, and $p(r|a)$ is the probability of receiving reward r given action a.

The agent's ultimate goal is to identify the optimal action, $a_* = \arg\max_a q_*(a)$. The core difficulty is that $q_*(a)$ is unknown and must be estimated from experience. This gap between the unknown true value and the agent's estimated value, $Q_t(a)$, is the central challenge.

1.3 The Core Conundrum: The Exploration-Exploitation Dilemma

This need to estimate values gives rise to a classic conflict in decision-making.

- **Exploitation**: Leveraging current knowledge to maximize immediate reward. This involves choosing the action with the highest estimated value, a "greedy" strategy.
- **Exploration**: Trying different actions, even those that seem suboptimal, to gather more information and improve the accuracy of value estimates. This is a long-term strategy.

A successful bandit algorithm must intelligently balance these competing objectives to maximize cumulative reward over the long run.

1.4 Applications in Practice

The MAB framework is widely applicable to real-world problems:

- **Clinical Trials**: Arms are different treatments. The goal is to quickly find the most effective treatment, maximizing patient recovery.
- **Recommender Systems & Online Advertising**: Arms are articles, products, or ads. The reward is user engagement (clicks, purchases). The system learns to display the most appealing content.
- **Other Domains**: Dynamic pricing, financial trading, and resource allocation in research.

2 Estimating Action-Values from Experience

The agent must estimate the true action-values, $q_*(a)$, from the rewards it observes.

2.1 The Sample-Average Method

The most intuitive estimation technique is the sample-average method. The estimated value of an action, $Q_t(a)$, is the average of all rewards received from that action.

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{N_t(a)} R_i}{N_t(a)} \tag{2}$$

where $N_t(a)$ is the number of times action a has been chosen. By the law of large numbers, $Q_t(a)$ converges to $q_*(a)$ as $N_t(a) \to \infty$. However, this method can be memory-intensive.

2.2 The Incremental Update Rule: Efficient Learning

A more efficient method is the incremental update rule, which updates estimates without storing all past rewards. For the (n)-th reward R_n for a given action, the new estimate Q_{n+1} is calculated from the previous estimate Q_n:

$$Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n) \tag{3}$$

This formula represents a general learning principle:

NewEstimate ← *OldEstimate* + *StepSize*(*Target* - *OldEstimate*)

Here, the **Target** is the new reward R_n, and the **StepSize** is $\frac{1}{n}$. The term $(R_n - Q_n)$ is the prediction error.

2.3 Addressing Non-Stationary Environments

In many real-world scenarios, the true action-values $q_*(a)$ can change over time (a non-stationary problem). The standard sample-average method is ill-suited for this, as the step size $\frac{1}{n}$ diminishes, effectively halting learning.

To handle this, a constant step-size parameter, $\alpha \in (0, 1]$, is used:

$$Q_{n+1} = Q_n + \alpha(R_n - Q_n) \tag{4}$$

This update rule creates an **exponential recency-weighted average**. The weight given to past rewards decreases exponentially with their age, allowing the agent to "forget" old information and adapt to changes.

3 Balancing the Dilemma: Action Selection Strategies

Having a value estimation method is not enough. The agent needs a policy to select actions.

3.1 The ϵ-Greedy Strategy

A simple and popular method for balancing exploration and exploitation.

- With probability $1 - \epsilon$, the agent **exploits** by choosing the action with the highest current estimated value: $\arg\max_a Q_t(a)$.
- With probability ϵ, the agent **explores** by choosing an action at random from all k actions.

While simple, its exploration is "undirected" or "blind," which can be inefficient.

3.2 Optimistic Initial Values

This method encourages early, systematic exploration by initializing all action-value estimates $Q_1(a)$ to a value known to be higher than any possible reward. A greedy agent will then be forced to try every arm at least once, as the reward from a chosen arm will be lower than the optimistic initial value of the unexplored arms. This is a simple but effective initial exploration strategy.

3.3 Upper Confidence Bound (UCB)

UCB provides a more sophisticated, deterministic approach to exploration based on the principle of "optimism in the face of uncertainty." It selects the action that maximizes a combination of the current value estimate and an uncertainty bonus.

$$A_t \doteq \arg\max_a \left[\underbrace{Q_t(a)}_{\text{Exploitation}} + \underbrace{c\sqrt{\frac{\ln t}{N_t(a)}}}_{\text{Exploration (Uncertainty Bonus)}} \right] \tag{5}$$

- $Q_t(a)$: The current estimated value (exploitation term).
- The second term is the uncertainty bonus. It is large for actions that have been tried infrequently ($N_t(a)$ is small). The $\ln t$ term ensures that all arms are eventually explored. The parameter $c > 0$ controls the degree of exploration.

UCB directs exploration towards actions where the value estimate is least certain, making it more efficient than ϵ-greedy's random exploration.

4 Introducing Context: The Advent of Contextual Bandits

The simple bandit framework assumes the best action is always the same. Contextual bandits relax this assumption, allowing for personalized decision-making.

4.1 The Role of Contextual Information

In a contextual bandit problem, the agent first observes a "context" (a feature vector, x_t) before choosing an action. The reward for an action now depends on the context in which it was chosen. The goal is to learn a policy that maps contexts to the best actions.

4.2 The LinUCB Algorithm

LinUCB combines the UCB principle with a linear model to handle contextual information.

4.2.1 The Linear Payoff Assumption

LinUCB assumes that the expected reward of an arm a is a linear function of its d-dimensional context feature vector $x_{t,a}$:

$$\mathbb{E}[r_{t,a}|x_{t,a}] = x_{t,a}^T \theta_{a,*} \tag{6}$$

where $\theta_{a,*}$ is an unknown coefficient vector for each arm a that the algorithm must learn. This vector is estimated using ridge regression:

$$\hat{\theta}_a = (D_a^T D_a + I_d)^{-1} D_a^T b_a \tag{7}$$

Here, D_a is a matrix of past context vectors for arm a, b_a is a vector of corresponding rewards, and I_d is an identity matrix for regularization.

4.2.2 The LinUCB Selection Rule

The action selection rule extends the UCB principle to this contextual setting:

$$a_t = \arg\max_{a \in A_t} \left[\underbrace{x_{t,a}^T \hat{\theta}_a}_{\text{Predicted Reward}} + \underbrace{\alpha \sqrt{x_{t,a}^T (D_a^T D_a + I_d)^{-1} x_{t,a}}}_{\text{Uncertainty Bound}} \right] \tag{8}$$

The exploration term is now the standard deviation of the reward prediction. This means exploration is driven by the uncertainty of the prediction for the *current context*, making it highly targeted and efficient.

4.2.3 The Complete Online Algorithm

Algorithm 1 LinUCB Algorithm

Initialize: For each arm a, set $A_a \leftarrow I_d$ (a $d \times d$ identity matrix) and $b_a \leftarrow 0_{d \times 1}$ (a zero vector).
for t = 1, 2, 3, ... **do**
    Observe context vectors $x_{t,a}$ for all arms a.
    for each arm a **do**
        Calculate $\hat{\theta}_a \leftarrow A_a^{-1} b_a$.
        Compute UCB score $p_{t,a} \leftarrow x_{t,a}^T \hat{\theta}_a + \alpha \sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$.
    end for
    Choose arm $a_t \leftarrow \arg\max_a p_{t,a}$.
    Observe reward r_t for the chosen arm a_t.
    Update:
    $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^T$.
    $b_{a_t} \leftarrow b_{a_t} + r_t x_{t,a_t}$.
end for

5 A Bayesian Perspective: Thompson Sampling

Thompson Sampling (TS) offers an elegant Bayesian alternative to UCB-style algorithms.

5.1 The Bayesian Heuristic: Representing Beliefs as Distributions

Instead of a single point estimate for an action's value, TS maintains a full probability distribution (the posterior) over the possible true values. This distribution represents the agent's "belief" about the arm's quality.

5.2 The Thompson Sampling Algorithm: Posterior Sampling

The action selection mechanism is remarkably simple:

1. **Sample**: At each time step, draw one random sample from each arm's current posterior distribution.

2. **Select**: Choose the arm that yielded the highest sample.

3. **Update**: Observe the reward and use it to update that arm's posterior distribution using Bayes' rule.

This process naturally balances exploration and exploitation. An arm with high uncertainty (a wide posterior) has a chance to produce a high sample, encouraging exploration. An arm with high certainty (a narrow posterior) will be exploited if its mean is high and ignored if its mean is low.

5.3 A Canonical Example: The Beta-Bernoulli Model

For Bernoulli bandits (binary rewards like 0 or 1), the Beta distribution is a perfect choice for the prior belief about the success probability $\theta \in [0, 1]$. The Beta distribution is the **conjugate prior** for the Bernoulli likelihood, which means the posterior is also a Beta distribution, making updates computationally trivial.

The update rule is simple:

- Start with a prior belief: Beta(α, β).
- Observe a success (reward = 1): The posterior becomes Beta($\alpha + 1$, β).
- Observe a failure (reward = 0): The posterior becomes Beta(α, $\beta + 1$).

5.3.1 The Algorithm in Practice

Algorithm 2 Thompson Sampling for Bernoulli Bandits

Initialize: For each arm i, set success count $S_i \leftarrow 0$ and failure count $F_i \leftarrow 0$.
for t = 1, 2, 3, ... **do**
    for each arm i **do**
        Draw a random sample $\theta_i(t)$ from its current posterior, Beta$(S_i + 1, F_i + 1)$.
    end for
    Play the arm $i(t) \leftarrow \arg\max_i \theta_i(t)$.
    Observe the reward $r_t \in \{0, 1\}$.
    if $r_t = 1$ **then**
        $S_{i(t)} \leftarrow S_{i(t)} + 1$.
    else
        $F_{i(t)} \leftarrow F_{i(t)} + 1$.
    end if
end for

6 Comparative Analysis and Concluding Remarks

The choice of a bandit algorithm depends on the problem's characteristics.

6.1 Algorithmic Performance

- **Necessity of Exploration**: Any form of exploration is vastly superior to a purely greedy strategy ($\epsilon = 0$).
- **Directed vs. Undirected Exploration**: Directed exploration (UCB) outperforms undirected exploration (ϵ-greedy).
- **UCB vs. Thompson Sampling**: In many empirical studies, Thompson Sampling often shows superior performance, being less deterministic and quicker to discard suboptimal arms.
- **Contextual vs. Non-Contextual**: Contextual algorithms (LinUCB) outperform non-contextual ones if the context is predictive of rewards. If not, their added complexity can be a disadvantage.

6.2 Practical Considerations and Selection Guidance

There is no single "best" algorithm. The choice involves trade-offs between complexity, cost, and performance.

Table 1: Comparative Summary of Bandit Algorithms

Algorithm	Core Mechanism	Key Parameter(s)	Handles Context?	Primary Advantage	Primary Disadvantage
ϵ-Greedy	Random exploration with probability ϵ.	ϵ (exploration rate)	No	Extremely simple to implement.	Undirected and inefficient exploration.
UCB	Selects based on value + uncertainty bonus.	c (exploration constant)	No	Principled, directed exploration.	Can be overly conservative; sensitive to c.
LinUCB	UCB principle on a linear reward model.	α (confidence), λ (regularization)	Yes	Personalizes decisions using features.	Assumes linear rewards; more complex.
Thompson Sampling	Samples from posterior belief distributions.	Prior distribution params (e.g., α, β)	No (but extendable)	Excellent empirical performance. Naturally balances trade-off.	Can be computationally intensive. Bayesian concepts can be less intuitive.

In conclusion, the evolution from simple ϵ-greedy methods to sophisticated contextual and Bayesian models like LinUCB and Thompson Sampling highlights a clear progression in intelligently managing the exploration-exploitation trade-off. The best choice depends on the specific problem, but superior performance is generally achieved through more directed and principled exploration strategies.