Reinforcement Learning

Lecture 2: The Multi-Armed Bandit Problem

Niranjan Deshpande

Minor in AI, IIT Ropar

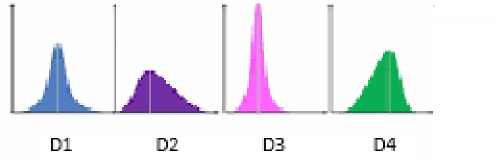
April 15, 2025

Today's Agenda

The One-Armed Bandit

A simplified problem: A single slot machine returns rewards probabilistically.

The Multi-Armed Bandit



Which arm should you pull to maximize rewards over time?

What is the Multi-Armed Bandit (MAB) Problem?

- You are faced with multiple options (arms), each giving rewards drawn from an unknown distribution.
- At each time step, you choose one arm to pull.
- Goal: **maximize the total cumulative reward over time**.
- Challenge: Balance **exploration** (trying new arms) and **exploitation** (choosing the best-known arm).

Real-World Applications

- Clinical trials (which treatment to assign).
- Network Routing.
- Online advertising (which ad to show).
- Game AI Designing and dynamic pricing.
- Recommender systems (which product/content to recommend)

Popular Bandit Algorithms

- **Epsilon-Greedy:** Simple and intuitive with fixed exploration.
- **Upper Confidence Bound (UCB):** Balances value and uncertainty.
- **Thompson Sampling:** Bayesian approach using probability distributions.

MAB Solution

From Uncertainty to Estimates

- Each arm (action) has an unknown reward distribution.
- When we choose an arm, we get one sample from its distribution.
- With complete knowledge, we'd always pick the highest expected reward.
- But we must summarize experience using a single informative number.

That number is the mean.

Why Use the Mean?

- The mean is the simplest estimate of expected reward.
- It captures the central tendency and improves with more samples.

$$Q(a) = \mathbb{E}[R_t \mid A_t = a], \quad Q_t(a) = \text{Estimate at time } t$$

Understanding the Mean

- Each arm produces a sequence of rewards.
- Rewards come from an unknown distribution.
- The true value:

$$q(a) = \mathbb{E}[R_t \mid A_t = a]$$

- We approximate it from observed rewards.

Estimating Action Values

The true value $q(a)$, estimated at time t as $Q_t(a)$, is the mean of rewards received when selecting action a **[Sutton & Barto]**.

$$Q_t(a) = \frac{R_1 + R_2 + \cdots + R_{N_t(a)}}{N_t(a)} = \frac{1}{N_t(a)} \sum_{i=1}^{N_t(a)} R_i$$

Practical Example

- Arm a pulled 3 times: $R_1 = 2, R_2 = 5, R_3 = 3$

$$Q(a) = \frac{2+5+3}{3} = \frac{10}{3} \approx 3.33$$

- This becomes our current belief.
- We update it after every new reward.

Alternative Form: Indicator Function

$$Q_t(a_i) = \frac{\sum_{j=1}^{t} \mathbf{1}\{A_j = a_i\} R_j}{\sum_{j=1}^{t} \mathbf{1}\{A_j = a_i\}}$$

$$\mathbf{1}\{A_j = a_i\} = \begin{cases} 1 & \text{if } A_j = a_i \\ 0 & \text{otherwise} \end{cases}$$

- Numerator: sum of rewards when a_i was selected.
- Denominator: number of times a_i was selected.

Formal Notation and Setup

- N: Number of actions (arms)
- a_i: i-th action
- A_t: Action at time t
- R_t: Reward at time t
- $q(a)$: True value of a
- $Q_t(a)$: Estimated value at t
- $N_t(a)$: Times a selected by t

Illustrative Example

$$A_1 = a_1,\ R_1 = 1$$
$$A_2 = a_1,\ R_2 = 2$$
$$A_3 = a_2,\ R_3 = 5$$
$$A_4 = a_1,\ R_4 = 4$$

$$N_5(a_1) = 3, \quad Q_5(a_1) = \frac{1+2+4}{3} = 2.33$$

$$N_5(a_2) = 1, \quad Q_5(a_2) = \frac{5}{1} = 5$$

Why Use a Recursive Update?

- Storing all past rewards is impractical.
- We can update the estimate incrementally.

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

Deriving Recursive Update

$$Q_n = \frac{1}{n}\sum_{i=1}^{n} R_i$$

Expressing Q_n in terms of Q_{n-1}:

$$Q_n = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right)$$

Note that the sum (by definition of Q_{n-1}):

$$\sum_{i=1}^{n-1} R_i = (n-1)Q_{n-1}$$

Substituting:

$$Q_n = \frac{1}{n}\left(R_n + (n-1)Q_{n-1}\right)$$

$$\boxed{Q_n = Q_{n-1} + \frac{1}{n}\left(R_n - Q_{n-1}\right)}$$

- Use sample means to estimate action values.
- Update estimates incrementally:

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

- This forms the foundation of bandit algorithms.

Epsilon-Greedy – Concept

- At each time step:
  - With probability ε, explore (choose a random arm).
  - With probability $1 - \varepsilon$, exploit (choose best estimated arm).
- Simple, effective baseline strategy.

Epsilon-Greedy - Intution

- A random number between 0 and 1 is generated.
- If this number is less than ϵ, the algorithm explores by choosing an arm randomly.
- Otherwise, exploits by selecting the arm with the highest current average reward estimate.
- The value of the hyperparameter $\epsilon \in [0, 1]$ controls how often exploration occur:
  - A small value (e.g., $\epsilon = 0.1$) allows greedy behavior with occasional exploration.
  - A larger value (e.g., $\epsilon = 0.5$) increases exploration, useful during the early stages of learning.

Epsilon-Greedy – Formula

$$Q_a \leftarrow Q_a + \frac{1}{N_a}(R_t - Q_a)$$

- Q_a: Estimated reward for arm a.
- N_a: Number of times arm a has been selected.
- R_t: Reward received at time t.

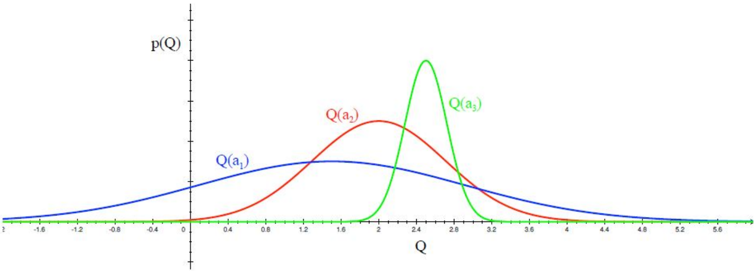
Epsilon-Greedy - Code

Upper Confidence Bound (UCB) – Introduction

- UCB is a popular solution to the Multi-Armed Bandit problem.
- Prefer arms with high average reward or high uncertainty.
- Principle: **Optimism in the face of uncertainty.**

"Instead of only trusting the average reward, give a bonus to actions we are less sure about."

UCB - Intuition



- Three arms a_1, a_2, a_3 with different uncertainty levels.
- UCB favors a_1 if it has highest variance, even if its mean is lower.
- Outcome:
  - If optimism pays off $\rightarrow$ high reward.
  - If not $\rightarrow$ uncertainty is reduced.

UCB - Summary of Intuition

- Optimism drives exploration.
- Helps balance between:
  - Exploiting known good arms.
  - Exploring uncertain but potentially better arms.
- Reduces wasted trials over time.

UCB1 - Algorithm Overview

1. **Initialization:** Play each arm once to initialize estimates.

2. **For each time step $t \geq K$:**
   - Compute UCB score for each arm:

$$\text{UCB}_t(a) = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$$

   - Select the arm with highest UCB value.
   - Update average reward $Q_t(a)$.

UCB – Formula

$$\mathrm{UCB}_t(a) = Q_t(a) + c \cdot \sqrt{\frac{\ln t}{N_t(a)}}$$

- $Q_t(a)$: Estimated average reward.
- $N_t(a)$: Number of times arm a was selected.
- t: Current time step.
- c: Exploration constant.

Interpreting the Formula

- $Q_t(a) \rightarrow$ Encourages exploitation.
- $\sqrt{\frac{\ln t}{N_t(a)}} \rightarrow$ Confidence bonus.
- As $N_t(a)$ increases, uncertainty shrinks.

$$N_t(a) \Uparrow \qquad \sqrt{\frac{2 \log t}{N_t(a)}} \Downarrow$$

- If $N_t(a)$ stays small and t grows, uncertainty grows.

$$t \Uparrow \qquad \text{With } N_t(a) \text{ constant} \qquad \sqrt{\frac{2 \log t}{N_t(a)}} \Uparrow$$

UCB - Code

```
import numpy as np

def ucb1(num_arms, num_episodes, c=1):
    Q = np.zeros(num_arms) # Average rewards
    N = np.zeros(num_arms) # Number of times each arm was played
    rewards = []

    for t in range(1, num_episodes + 1):
        if t <= num_arms:
            action = t - 1 # Play each arm once
        else:
            ucb_values = Q + c * np.sqrt(np.log(t) / (N + 1e-5))
            action = np.argmax(ucb_values)

        reward = np.random.binomial(1, true_probs[action]) # Simulated reward
        N[action] += 1
        Q[action] += (reward - Q[action]) / N[action]
        rewards.append(reward)

    return Q, rewards
```

UCB vs Epsilon-Greedy

Algorithm	**Exploration Strategy**
Epsilon-Greedy	Random arm with probability ε
UCB1	High reward $+$ high uncertainty arms
Adaptivity	Fixed exploration
Adaptivity (UCB)	Dynamically reduces exploration

UCB - Limitations and Considerations

- Assumes bounded, independent rewards.
- May struggle in nonstationary environments.
- Exploration constant c impacts convergence.

Summary

- UCB balances exploration and exploitation.
- Confidence bounds guide arm selection.
- Over time, focus shifts toward best arms.
- Performs well in many practical applications.

Thompson Sampling - Introduction

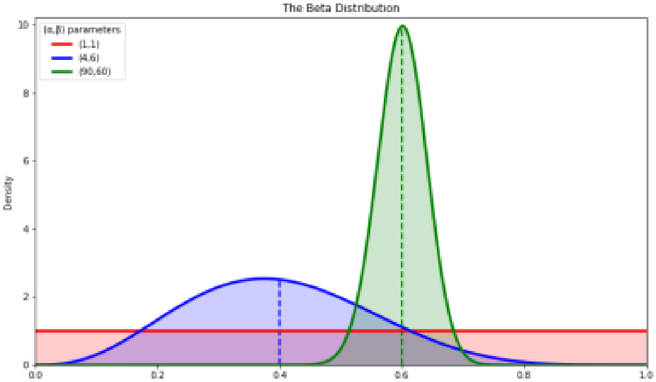
- Bayesian approach to balancing exploration and exploitation.
- Maintain a Beta distribution over each arm's reward probability.
- Focus on estimating the **probability of success** for each arm.
- Sample from each arm's distribution and choose the one with the highest sample.

Thompson Sampling - Modeling Belief

- Use Bayesian approach: maintain belief over success probability.
- Use **Beta distribution** with parameters:
  - α: number of successes.
  - β: number of failures.
- Mean:

$$\mathbb{E}[\theta] = \frac{\alpha}{\alpha + \beta}$$

Thompson Sampling - Visualizing Beta Distribution



- Shows probability curve over possible success rates.
- Distribution sharpens as more data is observed.

Thompson Sampling - Prior and Posterior

- Start with uniform prior:

$$\theta \sim \text{Beta}(1, 1)$$

- Update after observing outcome:
  - Reward $= 1$: $\alpha \leftarrow \alpha + 1$
  - Reward $= 0$: $\beta \leftarrow \beta + 1$
- Posterior reflects updated belief after each trial.

Thompson Sampling - Working

1. For each arm, sample θ_a from $\text{Beta}(\alpha_a, \beta_a)$.
2. Select arm with highest sampled value.
3. Pull arm, observe reward.
4. Update α, β for that arm.

Thompson Sampling - Exploration vs Exploitation

- Arms with many successes $\rightarrow$ narrow, tall distributions.
- Arms with few trials $\rightarrow$ wide distributions.
- Random sampling balances:
  - **Exploitation**: favoring best-known arms.
  - **Exploration**: occasionally trying uncertain arms.

Key Takeaways

- Thompson Sampling is a **Bayesian** method.
- It balances exploration and exploitation **automatically**.
- Works well with **binary rewards** and scales to more complex cases.
- Needs just two counters per arm: α and β.

Thompson Sampling – Concept

- Bayesian approach to balancing exploration and exploitation.
- Maintain a Beta distribution over each arm's reward probability.
- Sample from each arm's distribution and choose the one with the highest sample.

Thompson Sampling – Formula

$$\theta_a \sim \text{Beta}(\alpha_a, \beta_a)$$

- α_a: Number of observed successes for arm a.
- β_a: Number of observed failures.
- Select arm with highest sampled θ_a.

Optional: Applications

- Online advertising and click-through optimization.
- A/B testing and website design.
- Personalized recommendations.
- Clinical trials and adaptive experimentation.

Thompson Sampling – Concept

- Bayesian approach to balancing exploration and exploitation.
- Maintain a Beta distribution over each arm's reward probability.
- Sample from each arm's distribution and choose the one with the highest sample.

Thompson Sampling – Formula

$$\theta_a \sim \text{Beta}(\alpha_a, \beta_a)$$

- α_a: Number of observed successes for arm a.
- β_a: Number of observed failures.
- Choose arm with highest sampled θ_a

Summary

- The MAB problem is a classic setting for decision-making under uncertainty.
- Effective algorithms must balance exploring new arms vs. exploiting known rewards.
- **Epsilon-Greedy** is simple but static.
- **UCB** dynamically adjusts based on confidence.
- **Thompson Sampling** takes a probabilistic Bayesian approach.