Multi-Armed Bandit

DEPARTMENT OF COMPUTER SCIENCE
ST. FRANCIS XAVIER UNIVERSITY

CSCI-531 - Reinforcement Learning

Fall 2025

From Full RL to Bandit Problems

Why Start with Bandits?

- Multi-armed bandits are a **simplified version** of full RL

From Full RL to Bandit Problems

Why Start with Bandits?

- Multi-armed bandits are a **simplified version** of full RL
- Focus on one key aspect: the **exploration-exploitation trade-off**

From Full RL to Bandit Problems

Why Start with Bandits?

- Multi-armed bandits are a **simplified version** of full RL
- Focus on one key aspect: the **exploration-exploitation trade-off**
- Foundation for understanding more complex RL scenarios

From Full RL to Bandit Problems

Why Start with Bandits?

- Multi-armed bandits are a **simplified version** of full RL
- Focus on one key aspect: the **exploration-exploitation trade-off**
- Foundation for understanding more complex RL scenarios

Relationship to Full RL

- **Bandit**: Single state, multiple actions, immediate rewards

From Full RL to Bandit Problems

Why Start with Bandits?

- Multi-armed bandits are a **simplified version** of full RL
- Focus on one key aspect: the **exploration-exploitation trade-off**
- Foundation for understanding more complex RL scenarios

Relationship to Full RL

- **Bandit**: Single state, multiple actions, immediate rewards
- **Full RL**: Multiple states, multiple actions, delayed rewards

From Full RL to Bandit Problems

Why Start with Bandits?

- Multi-armed bandits are a **simplified version** of full RL
- Focus on one key aspect: the **exploration-exploitation trade-off**
- Foundation for understanding more complex RL scenarios

Relationship to Full RL

- **Bandit**: Single state, multiple actions, immediate rewards
- **Full RL**: Multiple states, multiple actions, delayed rewards
- **Key insight**: Every state in full RL = separate bandit problem

From Full RL to Bandit Problems

Why Start with Bandits?

- Multi-armed bandits are a **simplified version** of full RL
- Focus on one key aspect: the **exploration-exploitation trade-off**
- Foundation for understanding more complex RL scenarios

Relationship to Full RL

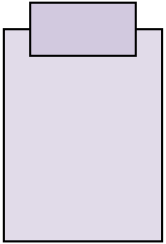
- **Bandit**: Single state, multiple actions, immediate rewards
- **Full RL**: Multiple states, multiple actions, delayed rewards
- **Key insight**: Every state in full RL = separate bandit problem

Understanding bandits $\rightarrow$ *Foundation for all RL*

What is the k-Armed Bandit Problem?

The Setup

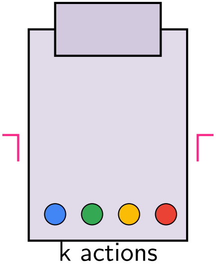
- Agent has k different actions



What is the k-Armed Bandit Problem?

The Setup

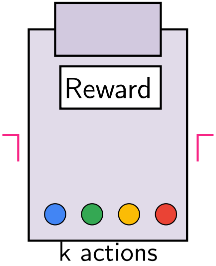
- Agent has k different actions
- After taking action $\rightarrow$ receive reward



What is the k-Armed Bandit Problem?

The Setup

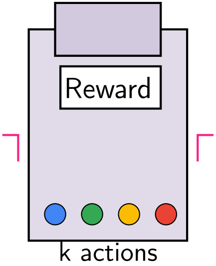
- Agent has k different actions
- After taking action $\rightarrow$ receive reward
- Rewards from stationary probability distribution



What is the k-Armed Bandit Problem?

The Setup

- Agent has k different actions
- After taking action $\rightarrow$ receive reward
- Rewards from stationary probability distribution
- **Objective**: Maximize expected total reward



Mathematical Formulation

Key Variables

- **Action at time** t: $A_t \in \{1, 2, \ldots, k\}$

Mathematical Formulation

Key Variables

- **Action at time** t: $A_t \in \{1, 2, \ldots, k\}$
- **Corresponding reward**: R_t

Mathematical Formulation

Key Variables

- **Action at time** t: $A_t \in \{1, 2, \ldots, k\}$
- **Corresponding reward**: R_t
- **Reward distribution**: $R_t \sim P(r|A_t)$

Mathematical Formulation

Key Variables

- **Action at time** t: $A_t \in \{1, 2, \dots, k\}$
- **Corresponding reward**: R_t
- **Reward distribution**: $R_t \sim P(r|A_t)$

True vs Estimated Values

True value of action a**:**

$$q_*(a) = \mathbb{E}[R_t|A_t = a]$$

Estimated value at time t**:**

$$Q_t(a) \approx q_*(a)$$

Mathematical Formulation

Key Variables

- **Action at time** t: $A_t \in \{1, 2, \ldots, k\}$
- **Corresponding reward**: R_t
- **Reward distribution**: $R_t \sim P(r|A_t)$

True vs Estimated Values

True value of action a:

$$q_*(a) = \mathbb{E}[R_t | A_t = a]$$

Estimated value at time t:

$$Q_t(a) \approx q_*(a)$$

Challenge: We don't know $q_*(a)$ - must estimate from experience!

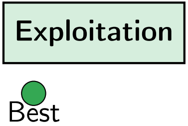
The Exploration-Exploitation Dilemma

When you know $Q_t(a)$ for all actions:

The Exploration-Exploitation Dilemma

When you know $Q_t(a)$ for all actions:

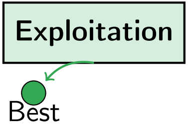
- **Exploitation**: Choose $\arg\max_a Q_t(a)$
  - Greedy action selection



The Exploration-Exploitation Dilemma

When you know $Q_t(a)$ for all actions:

- **Exploitation**: Choose $\arg\max_a Q_t(a)$
  - Greedy action selection
  - Maximizes immediate reward



The Exploration-Exploitation Dilemma

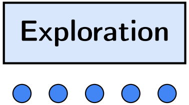
When you know $Q_t(a)$ for all actions:

- **Exploitation**: Choose $\arg\max_a Q_t(a)$
  - Greedy action selection
  - Maximizes immediate reward
  - But estimates might be wrong!

The Exploration-Exploitation Dilemma

When you know $Q_t(a)$ for all actions:

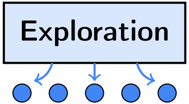
- **Exploitation**: Choose $\arg\max_a Q_t(a)$
  - Greedy action selection
  - Maximizes immediate reward
  - But estimates might be wrong!
- **Exploration**: Choose non-greedy actions
  - Improves action-value estimates



The Exploration-Exploitation Dilemma

When you know $Q_t(a)$ for all actions:

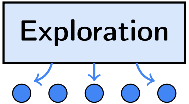
- **Exploitation**: Choose $\arg\max_a Q_t(a)$
  - Greedy action selection
  - Maximizes immediate reward
  - But estimates might be wrong!
- **Exploration**: Choose non-greedy actions
  - Improves action-value estimates
  - Necessary for long-term success



The Exploration-Exploitation Dilemma

When you know $Q_t(a)$ for all actions:

- **Exploitation**: Choose $\arg\max_a Q_t(a)$
  - Greedy action selection
  - Maximizes immediate reward
  - But estimates might be wrong!
- **Exploration**: Choose non-greedy actions
  - Improves action-value estimates
  - Necessary for long-term success
  - But sacrifices immediate reward



Cost: Lower immediate reward

The Exploration-Exploitation Dilemma

When you know $Q_t(a)$ for all actions:

- **Exploitation**: Choose $\arg\max_a Q_t(a)$
  - Greedy action selection
  - Maximizes immediate reward
  - But estimates might be wrong!
- **Exploration**: Choose non-greedy actions
  - Improves action-value estimates
  - Necessary for long-term success
  - But sacrifices immediate reward

The Challenge: Balance exploration and exploitation

Action-Value Methods

The Approach

1. **Estimate** action values from experience

Action-Value Methods

The Approach

1. **Estimate** action values from experience
2. **Use** these estimates to select actions

Action-Value Methods

The Approach

1. **Estimate** action values from experience
2. **Use** these estimates to select actions
3. **Update** estimates as we gain more experience

Action-Value Methods

The Approach

1. **Estimate** action values from experience
2. **Use** these estimates to select actions
3. **Update** estimates as we gain more experience

Why This Works

- We observe rewards for each action we take

Action-Value Methods

The Approach

1. **Estimate** action values from experience
2. **Use** these estimates to select actions
3. **Update** estimates as we gain more experience

Why This Works

- We observe rewards for each action we take
- More samples $\rightarrow$ better estimates

Action-Value Methods

The Approach

1. **Estimate** action values from experience
2. **Use** these estimates to select actions
3. **Update** estimates as we gain more experience

Why This Works

- We observe rewards for each action we take
- More samples $\rightarrow$ better estimates
- Can balance exploration (trying new actions) vs exploitation (using best known action)

Action-Value Methods

The Approach

1. **Estimate** action values from experience
2. **Use** these estimates to select actions
3. **Update** estimates as we gain more experience

Why This Works

- We observe rewards for each action we take
- More samples $\rightarrow$ better estimates
- Can balance exploration (trying new actions) vs exploitation (using best known action)

How do we estimate action values?

Sample Average Method

Natural Estimation Approach

One natural way to estimate expected reward:

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

Sample Average Method

Natural Estimation Approach

One natural way to estimate expected reward:

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

Mathematical Formulation

More formally:

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

where $\mathbb{1}_{A_i=a}$ is 1 if action a was taken at time i, 0 otherwise.

Sample Average Method

Natural Estimation Approach

One natural way to estimate expected reward:

$$Q_t(a) = \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t}$$

Mathematical Formulation

More formally:

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

where $\mathbb{1}_{A_i=a}$ is 1 if action a was taken at time i, 0 otherwise.

Law of Large Numbers: As $n \to \infty$, $Q_t(a) \to q_*(a)$

Example: Estimating Action Values

Time:	1	2	3	4	5	6
Action:	A_2	A_1	A_3	A_4	A_3	A_1
Reward:	5	-1	2	10	1.5	1

Estimates:

$$Q_7(A_1) = \frac{-1+1}{2} = 0 \qquad Q_7(A_2) = \frac{5}{1} = 5$$

$$Q_7(A_3) = \frac{2+1.5}{2} = 1.75 \qquad Q_7(A_4) = \frac{10}{1} = 10$$

Best action so far: A_4 with $Q_7(A_4) = 10$

ϵ-Greedy Action Selection

The Strategy

Balance exploration and exploitation with a simple rule:

ϵ-Greedy Action Selection

The Strategy

Balance exploration and exploitation with a simple rule:

- With probability $(1-\epsilon)$: Choose **greedy action**

$$A_t = \arg\max_a Q_t(a)$$

ϵ-Greedy Action Selection

The Strategy

Balance exploration and exploitation with a simple rule:

- With probability $(1-\epsilon)$: Choose greedy action

$$A_t = \arg\max_a Q_t(a)$$

- With probability ϵ: Choose random action

$$A_t = \text{uniform random from all actions}$$

ϵ-Greedy Action Selection

The Strategy

Balance exploration and exploitation with a simple rule:

- With probability $(1 - \epsilon)$: Choose **greedy action**

$$A_t = \arg\max_a Q_t(a)$$

- With probability ϵ: Choose **random action**

$$A_t = \text{uniform random from all actions}$$

Simple but effective approach to exploration-exploitation

ϵ-Greedy Parameter Analysis

Parameter ϵ

- $\epsilon = 0$: Pure exploitation (greedy)

ϵ-Greedy Parameter Analysis

Parameter ϵ

- $\epsilon = 0$: Pure exploitation (greedy)
- $\epsilon = 1$: Pure exploration (random)

ϵ-Greedy Parameter Analysis

Parameter ϵ

- $\epsilon = 0$: Pure exploitation (greedy)
- $\epsilon = 1$: Pure exploration (random)
- $\epsilon \in (0, 1)$: Balanced approach

ϵ-Greedy Parameter Analysis

Parameter ϵ

- $\epsilon = 0$: Pure exploitation (greedy)
- $\epsilon = 1$: Pure exploration (random)
- $\epsilon \in (0, 1)$: Balanced approach

Properties

- **Guarantees convergence** to optimal action

ϵ-Greedy Parameter Analysis

Parameter ϵ

- $\epsilon = 0$: Pure exploitation (greedy)
- $\epsilon = 1$: Pure exploration (random)
- $\epsilon \in (0, 1)$: Balanced approach

Properties

- **Guarantees convergence** to optimal action
- **But** doesn't guarantee optimal performance

ϵ-Greedy Parameter Analysis

Parameter ϵ

- $\epsilon = 0$: Pure exploitation (greedy)
- $\epsilon = 1$: Pure exploration (random)
- $\epsilon \in (0, 1)$: Balanced approach

Properties

- **Guarantees convergence** to optimal action
- **But** doesn't guarantee optimal performance
- Simple to implement and understand

ϵ-Greedy Parameter Analysis

Parameter ϵ

- $\epsilon = 0$: Pure exploitation (greedy)
- $\epsilon = 1$: Pure exploration (random)
- $\epsilon \in (0, 1)$: Balanced approach

Properties

- **Guarantees convergence** to optimal action
- **But** doesn't guarantee optimal performance
- Simple to implement and understand

Trade-off: Higher ϵ = more exploration, lower immediate reward

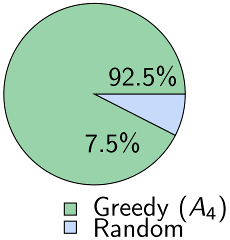
ϵ-Greedy Example

Current estimates:

$$Q_t(A_1) = 0$$
$$Q_t(A_2) = 5$$
$$Q_t(A_3) = 1.75$$
$$Q_t(A_4) = 10 \quad \leftarrow \text{Best}$$

With $\epsilon = 0.1$:

- Probability of choosing A_4: $0.9 + 0.1 \times 0.25 = 0.925$
- Probability of choosing any other action: $0.1 \times 0.25 = 0.025$



The Computational Challenge

Naive Implementation

Store all rewards and recompute average each time:

$$Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

Problems:

The Computational Challenge

Naive Implementation

Store all rewards and recompute average each time:

$$Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

Problems:

- Memory grows linearly with time

The Computational Challenge

Naive Implementation

Store all rewards and recompute average each time:

$$Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

Problems:

- Memory grows linearly with time
- Computation time increases with each update

The Computational Challenge

Naive Implementation

Store all rewards and recompute average each time:

$$Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

Problems:

- Memory grows linearly with time
- Computation time increases with each update

Better approach needed.

Incremental Update Rule

Mathematical Derivation

Starting with the sample average:

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

Incremental Update Rule

Mathematical Derivation

Starting with the sample average:

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

Incremental Update Rule

Mathematical Derivation

Starting with the sample average:

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

Incremental Update Rule

Mathematical Derivation

Starting with the sample average:

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

Incremental Update Rule

Mathematical Derivation

Starting with the sample average:

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

Incremental Update Rule

Mathematical Derivation

Starting with the sample average:

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)Q_n\right) = \frac{1}{n}R_n + \frac{n-1}{n}Q_n \\
&= Q_n + \frac{1}{n}\left[R_n - Q_n\right]
\end{aligned}
$$

Incremental Update Rule

Mathematical Derivation

Starting with the sample average:

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i = \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)Q_n\right) = \frac{1}{n}R_n + \frac{n-1}{n}Q_n \\
&= Q_n + \frac{1}{n}\left[R_n - Q_n\right]
\end{aligned}
$$

$$Q_{n+1} = Q_n + \tfrac{1}{n}[R_n - Q_n]$$

The General Update Form

Key RL Update Pattern

$$\text{NewEstimate} \leftarrow \text{OldEstimate} \quad + \text{StepSize} \times [\text{Target} - \text{OldEstimate}]$$

The General Update Form

Key RL Update Pattern

$$\text{NewEstimate} \leftarrow \text{OldEstimate} \quad + \text{StepSize} \times [\text{Target} - \text{OldEstimate}]$$

Components

- **Target**: R_n (the observed reward)

The General Update Form

Key RL Update Pattern

$$\text{NewEstimate} \leftarrow \text{OldEstimate} \quad + \text{StepSize} \times [\text{Target} - \text{OldEstimate}]$$

Components

- **Target**: R_n (the observed reward)
- **Prediction Error**: $[R_n - Q_n]$

The General Update Form

Key RL Update Pattern

$$\text{NewEstimate} \leftarrow \text{OldEstimate} \quad + \text{StepSize} \times [\text{Target} - \text{OldEstimate}]$$

Components

- **Target**: R_n (the observed reward)
- **Prediction Error**: $[R_n - Q_n]$
- **Step Size**: $\frac{1}{n}$ (how much to adjust)

The General Update Form

Key RL Update Pattern

$$\text{NewEstimate} \leftarrow \text{OldEstimate} \quad + \text{StepSize} \times [\text{Target} - \text{OldEstimate}]$$

Components

- **Target**: R_n (the observed reward)
- **Prediction Error**: $[R_n - Q_n]$
- **Step Size**: $\frac{1}{n}$ (how much to adjust)

This pattern appears throughout all of reinforcement learning!

Complete k-Armed Bandit Algorithm

Algorithm ϵ-Greedy k-Armed Bandit

Initialize:
for $a = 1$ to k **do**
    $Q(a) \leftarrow 0$
    $N(a) \leftarrow 0$
end for
Loop forever:
$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \end{cases}$$
$R \leftarrow \text{bandit}(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$

Beyond Sample Averages: Constant Step-Size

Limitation of Sample Average

Sample average method: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$

Beyond Sample Averages: Constant Step-Size

Limitation of Sample Average

Sample average method: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$

- Gives **equal weight** to all past rewards

Beyond Sample Averages: Constant Step-Size

Limitation of Sample Average

Sample average method: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$

- Gives **equal weight** to all past rewards
- Step size $\frac{1}{n}$ decreases over time

Beyond Sample Averages: Constant Step-Size

Limitation of Sample Average

Sample average method: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$

- Gives **equal weight** to all past rewards
- Step size $\frac{1}{n}$ decreases over time
- **Problem**: Old rewards have too much influence

Beyond Sample Averages: Constant Step-Size

Limitation of Sample Average

Sample average method: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$

- Gives **equal weight** to all past rewards
- Step size $\frac{1}{n}$ decreases over time
- **Problem**: Old rewards have too much influence

Constant Step-Size Alternative

Replace $\frac{1}{n}$ with constant $\alpha \in (0, 1]$:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

Benefits:

Beyond Sample Averages: Constant Step-Size

Limitation of Sample Average

Sample average method: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$

- Gives **equal weight** to all past rewards
- Step size $\frac{1}{n}$ decreases over time
- **Problem**: Old rewards have too much influence

Constant Step-Size Alternative

Replace $\frac{1}{n}$ with constant $\alpha \in (0, 1]$:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

Benefits:

- More weight to recent rewards

Beyond Sample Averages: Constant Step-Size

Limitation of Sample Average

Sample average method: $Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$

- Gives **equal weight** to all past rewards
- Step size $\frac{1}{n}$ decreases over time
- **Problem**: Old rewards have too much influence

Constant Step-Size Alternative

Replace $\frac{1}{n}$ with constant $\alpha \in (0, 1]$:

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

Benefits:

- More weight to recent rewards
- Called **weighted average**

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$Q_{n+1} = Q_n + \alpha[R_n - Q_n]$$

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\
&= (1-\alpha)Q_n + \alpha R_n
\end{aligned}
$$

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\
&= (1-\alpha)Q_n + \alpha R_n \\
&= (1-\alpha)[(1-\alpha)Q_{n-1} + \alpha R_{n-1}] + \alpha R_n
\end{aligned}
$$

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\
&= (1-\alpha)Q_n + \alpha R_n \\
&= (1-\alpha)[(1-\alpha)Q_{n-1} + \alpha R_{n-1}] + \alpha R_n \\
&= (1-\alpha)^2 Q_{n-1} + \alpha(1-\alpha)R_{n-1} + \alpha R_n
\end{aligned}
$$

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\
&= (1-\alpha)Q_n + \alpha R_n \\
&= (1-\alpha)[(1-\alpha)Q_{n-1} + \alpha R_{n-1}] + \alpha R_n \\
&= (1-\alpha)^2 Q_{n-1} + \alpha(1-\alpha)R_{n-1} + \alpha R_n \\
&= \cdots
\end{aligned}
$$

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha[R_n - Q_n] \\
&= (1-\alpha)Q_n + \alpha R_n \\
&= (1-\alpha)[(1-\alpha)Q_{n-1} + \alpha R_{n-1}] + \alpha R_n \\
&= (1-\alpha)^2 Q_{n-1} + \alpha(1-\alpha)R_{n-1} + \alpha R_n \\
&= \cdots \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned}
$$

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

Weighted Average Derivation

Expanding the Constant Step-Size Update

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

Weight of reward R_i: $\alpha(1-\alpha)^{n-i}$

- Recent rewards have weight close to α
- Weights decay exponentially into the past
- Sum of all weights $= 1$ (true weighted average)

Upper Confidence Bound (UCB) Action Selection

Limitation of ϵ-Greedy

- Random exploration among **all** non-greedy actions

Upper Confidence Bound (UCB) Action Selection

Limitation of ϵ-Greedy

- Random exploration among **all** non-greedy actions
- No preference for actions with high uncertainty

Upper Confidence Bound (UCB) Action Selection

Limitation of ϵ-Greedy

- Random exploration among **all** non-greedy actions
- No preference for actions with high uncertainty
- Doesn't consider **how uncertain** our estimates are

Upper Confidence Bound (UCB) Action Selection

Limitation of ϵ-Greedy

- Random exploration among **all** non-greedy actions
- No preference for actions with high uncertainty
- Doesn't consider **how uncertain** our estimates are

UCB Idea

Explore based on uncertainty!

Upper Confidence Bound (UCB) Action Selection

Limitation of ϵ-Greedy

- Random exploration among **all** non-greedy actions
- No preference for actions with high uncertainty
- Doesn't consider **how uncertain** our estimates are

UCB Idea

Explore based on uncertainty!

- High uncertainty $\rightarrow$ more exploration

Upper Confidence Bound (UCB) Action Selection

Limitation of ϵ-Greedy

- Random exploration among **all** non-greedy actions
- No preference for actions with high uncertainty
- Doesn't consider **how uncertain** our estimates are

UCB Idea

Explore based on uncertainty!

- High uncertainty $\rightarrow$ more exploration
- Low uncertainty $\rightarrow$ less exploration

Upper Confidence Bound (UCB) Action Selection

Limitation of ϵ-Greedy

- Random exploration among **all** non-greedy actions
- No preference for actions with high uncertainty
- Doesn't consider **how uncertain** our estimates are

UCB Idea

Explore based on uncertainty!

- High uncertainty $\rightarrow$ more exploration
- Low uncertainty $\rightarrow$ less exploration
- Select action with highest **upper confidence bound**

Understanding the UCB Formula

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

where $c > 0$ controls the degree of exploration

Exploration Bonus Analysis

Understanding the UCB Formula

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

where $c > 0$ controls the degree of exploration

Exploration Bonus Analysis

- $N_t(a)$ **increases**: Uncertainty decreases $\rightarrow$ bonus decreases

Understanding the UCB Formula

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

where $c > 0$ controls the degree of exploration

Exploration Bonus Analysis

- $N_t(a)$ **increases**: Uncertainty decreases $\rightarrow$ bonus decreases
- t **increases**: More total trials $\rightarrow$ should explore more

Understanding the UCB Formula

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

where $c > 0$ controls the degree of exploration

Exploration Bonus Analysis

- $N_t(a)$ **increases**: Uncertainty decreases $\rightarrow$ bonus decreases
- t **increases**: More total trials $\rightarrow$ should explore more
- c **parameter**: Controls exploration vs exploitation trade-off

Understanding the UCB Formula

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

Intuition

Understanding the UCB Formula

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

Intuition

- Actions with few trials get high bonus (high uncertainty)

Understanding the UCB Formula

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

Intuition

- Actions with few trials get high bonus (high uncertainty)
- Actions with many trials get low bonus (low uncertainty)

Understanding the UCB Formula

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

Intuition

- Actions with few trials get high bonus (high uncertainty)
- Actions with many trials get low bonus (low uncertainty)
- Automatically balances exploration and exploitation

Understanding the UCB Formula

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

Intuition

- Actions with few trials get high bonus (high uncertainty)
- Actions with many trials get low bonus (low uncertainty)
- Automatically balances exploration and exploitation

No parameters to tune over time - UCB adapts automatically!

UCB Example

Current state after 8 time steps:

$$Q_8(A_1) = 0, \quad N_8(A_1) = 2 \quad Q_8(A_2) = 5, \quad N_8(A_2) = 1$$
$$Q_8(A_3) = 1.75, \quad N_8(A_3) = 3 \quad Q_8(A_4) = 10, \quad N_8(A_4) = 2$$

UCB values with $c = 2$:

$$UCB_8(A_1) = 0 + 2\sqrt{\frac{\ln 8}{2}} = 0 + 2\sqrt{1.04} = 2.04$$

$$UCB_8(A_2) = 5 + 2\sqrt{\frac{\ln 8}{1}} = 5 + 2\sqrt{2.08} = 7.88$$

$$UCB_8(A_3) = 1.75 + 2\sqrt{\frac{\ln 8}{3}} = 1.75 + 2\sqrt{0.69} = 3.42$$

$$UCB_8(A_4) = 10 + 2\sqrt{\frac{\ln 8}{2}} = 10 + 2\sqrt{1.04} = 12.04$$

Gradient Bandit Algorithms

Different Approach

Instead of estimating action values, learn **action preferences**:

Gradient Bandit Algorithms

Different Approach

Instead of estimating action values, learn **action preferences**:

- Maintain preference $H_t(a)$ for each action a

Gradient Bandit Algorithms

Different Approach

Instead of estimating action values, learn **action preferences**:

- Maintain preference $H_t(a)$ for each action a
- Higher preference $\rightarrow$ action selected more often

Gradient Bandit Algorithms

Different Approach

Instead of estimating action values, learn **action preferences**:

- Maintain preference $H_t(a)$ for each action a
- Higher preference $\rightarrow$ action selected more often
- Use preferences in **softmax distribution**

Gradient Bandit Algorithms

Different Approach

Instead of estimating action values, learn **action preferences**:

- Maintain preference $H_t(a)$ for each action a
- Higher preference $\rightarrow$ action selected more often
- Use preferences in **softmax distribution**

Softmax Action Selection

$$\pi_t(a) = P(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

where $\pi_t(a)$ is the probability of selecting action a at time t.

Gradient Bandit Algorithms

Softmax Action Selection

$$\pi_t(a) = P(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

where $\pi_t(a)$ is the probability of selecting action a at time t.

Gradient Bandit Algorithms

Softmax Action Selection

$$\pi_t(a) = P(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

where $\pi_t(a)$ is the probability of selecting action a at time t.

Initial Conditions

Gradient Bandit Algorithms

Softmax Action Selection

$$\pi_t(a) = P(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

where $\pi_t(a)$ is the probability of selecting action a at time t.

Initial Conditions

- Start with $H_1(a) = 0$ for all actions

Gradient Bandit Algorithms

Softmax Action Selection

$$\pi_t(a) = P(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}}$$

where $\pi_t(a)$ is the probability of selecting action a at time t.

Initial Conditions

- Start with $H_1(a) = 0$ for all actions
- Initially: $\pi_1(a) = \frac{1}{k}$ (uniform distribution)

Gradient Bandit: Action Preferences

Key Insight

- No need to estimate action values $q_*(a)$

Gradient Bandit: Action Preferences

Key Insight

- No need to estimate action values $q_*(a)$
- Instead, learn relative preferences between actions

Gradient Bandit: Action Preferences

Key Insight

- No need to estimate action values $q_*(a)$
- Instead, learn relative preferences between actions
- Preferences determine selection probabilities

Gradient Bandit: Action Preferences

Key Insight

- No need to estimate action values $q_*(a)$
- Instead, learn relative preferences between actions
- Preferences determine selection probabilities

Preference Properties

- Only **relative** differences matter

Gradient Bandit: Action Preferences

Key Insight

- No need to estimate action values $q_*(a)$
- Instead, learn relative preferences between actions
- Preferences determine selection probabilities

Preference Properties

- Only **relative** differences matter
- Adding constant to all preferences: no effect on probabilities

Gradient Bandit: Action Preferences

Key Insight

- No need to estimate action values $q_*(a)$
- Instead, learn relative preferences between actions
- Preferences determine selection probabilities

Preference Properties

- Only **relative** differences matter
- Adding constant to all preferences: no effect on probabilities
- If $H_t(a) > H_t(b)$, then $\pi_t(a) > \pi_t(b)$

Gradient Bandit Update Rules

Stochastic Gradient Ascent Updates

For the selected action:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

For all other actions $a \neq A_t$**:**

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a)$$

where $\bar{R}_t$ is the average of all rewards up to time t (baseline).

Gradient Bandit Update Rules

Stochastic Gradient Ascent Updates

For the selected action:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

For all other actions $a \neq A_t$:

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a)$$

where $\bar{R}_t$ is the average of all rewards up to time t (baseline).

Intuition

Gradient Bandit Update Rules

Stochastic Gradient Ascent Updates

For the selected action:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

For all other actions $a \neq A_t$:

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a)$$

where $\bar{R}_t$ is the average of all rewards up to time t (baseline).

Intuition

- $R_t > \bar{R}_t$: Increase preference for A_t, decrease for others

Gradient Bandit Update Rules

Stochastic Gradient Ascent Updates

For the selected action:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

For all other actions $a \neq A_t$:

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a)$$

where $\bar{R}_t$ is the average of all rewards up to time t (baseline).

Intuition

- $R_t > \bar{R}_t$: Increase preference for A_t, decrease for others
- $R_t < \bar{R}_t$: Decrease preference for A_t, increase for others

Gradient Bandit Update Rules

Stochastic Gradient Ascent Updates

For the selected action:

$$H_{t+1}(A_t) = H_t(A_t) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(A_t))$$

For all other actions $a \neq A_t$:

$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a)$$

where $\bar{R}_t$ is the average of all rewards up to time t (baseline).

Intuition

- $R_t > \bar{R}_t$: Increase preference for A_t, decrease for others
- $R_t < \bar{R}_t$: Decrease preference for A_t, increase for others
- Baseline $\bar{R}_t$ provides context - what's "good" vs "bad"

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values
- Use **uncertainty** to guide exploration naturally

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values
- Use **uncertainty** to guide exploration naturally
- No exploration parameters to tune!

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values
- Use **uncertainty** to guide exploration naturally
- No exploration parameters to tune!

Thompson Sampling Algorithm

At each time step:

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values
- Use **uncertainty** to guide exploration naturally
- No exploration parameters to tune!

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values
- Use **uncertainty** to guide exploration naturally
- No exploration parameters to tune!

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values
- Use **uncertainty** to guide exploration naturally
- No exploration parameters to tune!

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$
3. **Observe** reward R_t

Thompson Sampling: Bayesian Approach

The Bayesian Philosophy

- Maintain **belief distributions** over true action values
- Use **uncertainty** to guide exploration naturally
- No exploration parameters to tune!

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$
3. **Observe** reward R_t
4. **Update** belief distribution for A_t using Bayesian inference

Thompson Sampling: Bayesian Approach

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$
3. **Observe** reward R_t
4. **Update** belief distribution for A_t using Bayesian inference

Thompson Sampling: Bayesian Approach

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$
3. **Observe** reward R_t
4. **Update** belief distribution for A_t using Bayesian inference

Why It Works

Thompson Sampling: Bayesian Approach

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$
3. **Observe** reward R_t
4. **Update** belief distribution for A_t using Bayesian inference

Why It Works

- **Probability matching**: Prob(select action) = Prob(action is optimal)

Thompson Sampling: Bayesian Approach

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$
3. **Observe** reward R_t
4. **Update** belief distribution for A_t using Bayesian inference

Why It Works

- **Probability matching**: Prob(select action) = Prob(action is optimal)
- **Natural exploration**: High uncertainty $\rightarrow$ more sampling variation

Thompson Sampling: Bayesian Approach

Thompson Sampling Algorithm

At each time step:

1. **Sample** θ_a from belief distribution for each action a
2. **Select** action $A_t = \arg\max_a \theta_a$
3. **Observe** reward R_t
4. **Update** belief distribution for A_t using Bayesian inference

Why It Works

- **Probability matching**: Prob(select action) = Prob(action is optimal)
- **Natural exploration**: High uncertainty $\rightarrow$ more sampling variation
- **Theoretically optimal** for many reward distributions

Bandit Algorithm Comparison

Algorithm	**Exploration**	**Parameters**	**Cost**	**Best Use**
ϵ-greedy	Random	ϵ	Very Low	Simple baseline
UCB	Confidence	c	Low	Principled exploration
Gradient	Preference	α, baseline	Medium	Varying scales
Thompson	Bayesian	Prior params	Medium	Optimal exploration

Bandit Algorithm Comparison

Algorithm	**Exploration**	**Parameters**	**Cost**	**Best Use**
ϵ-greedy	Random	ϵ	Very Low	Simple baseline
UCB	Confidence	c	Low	Principled exploration
Gradient	Preference	α, baseline	Medium	Varying scales
Thompson	Bayesian	Prior params	Medium	Optimal exploration

When to Use Which?

- **ϵ-greedy**: Need interpretable, simple baseline

Bandit Algorithm Comparison

Algorithm	**Exploration**	**Parameters**	**Cost**	**Best Use**
ϵ-greedy	Random	ϵ	Very Low	Simple baseline
UCB	Confidence	c	Low	Principled exploration
Gradient	Preference	α, baseline	Medium	Varying scales
Thompson	Bayesian	Prior params	Medium	Optimal exploration

When to Use Which?

- ϵ-**greedy**: Need interpretable, simple baseline
- **UCB**: Want theoretical guarantees without parameter tuning

Bandit Algorithm Comparison

Algorithm	**Exploration**	**Parameters**	**Cost**	**Best Use**
ϵ-greedy	Random	ϵ	Very Low	Simple baseline
UCB	Confidence	c	Low	Principled exploration
Gradient	Preference	α, baseline	Medium	Varying scales
Thompson	Bayesian	Prior params	Medium	Optimal exploration

When to Use Which?

- **ϵ-greedy**: Need interpretable, simple baseline
- **UCB**: Want theoretical guarantees without parameter tuning
- **Gradient**: Action values have very different scales

Bandit Algorithm Comparison

Algorithm	**Exploration**	**Parameters**	**Cost**	**Best Use**
ϵ-greedy	Random	ϵ	Very Low	Simple baseline
UCB	Confidence	c	Low	Principled exploration
Gradient	Preference	α, baseline	Medium	Varying scales
Thompson	Bayesian	Prior params	Medium	Optimal exploration

When to Use Which?

- ϵ**-greedy**: Need interpretable, simple baseline
- **UCB**: Want theoretical guarantees without parameter tuning
- **Gradient**: Action values have very different scales
- **Thompson**: Want near-optimal exploration

Theoretical Performance Guarantees

Regret Analysis

Regret = Total reward loss compared to always choosing optimal action

$$\text{Regret}_T = T \cdot r^* - \sum_{t=1}^{T} R_t$$

where $r^* = \max_a q_*(a)$ is the optimal expected reward.

Theoretical Performance Guarantees

Algorithm Regret Bounds

Algorithm	**Regret Bound**
Random	$O(T)$
ϵ-greedy (fixed ϵ)	$O(T)$
ϵ-greedy (decreasing ϵ)	$O(\sqrt{T \ln T})$
UCB	$O(\sqrt{kT \ln T})$
Thompson Sampling	$O(\sqrt{kT})$

Theoretical Performance Guarantees

Algorithm Regret Bounds

Algorithm	**Regret Bound**
Random	$O(T)$
ϵ-greedy (fixed ϵ)	$O(T)$
ϵ-greedy (decreasing ϵ)	$O(\sqrt{T \ln T})$
UCB	$O(\sqrt{kT \ln T})$
Thompson Sampling	$O(\sqrt{kT})$

Key insight: Sublinear regret bounds ($O(T)$) mean the algorithm eventually finds the optimal action!

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off
- **Value estimation** from experience

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off
- **Value estimation** from experience
- **Action selection** strategies

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off
- **Value estimation** from experience
- **Action selection** strategies
- **Learning from rewards**

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off
- **Value estimation** from experience
- **Action selection** strategies
- **Learning from rewards**

Limitations of Bandits

- **Single state**: All decisions in same context

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off
- **Value estimation** from experience
- **Action selection** strategies
- **Learning from rewards**

Limitations of Bandits

- **Single state**: All decisions in same context
- **Immediate rewards**: No delayed consequences

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off
- **Value estimation** from experience
- **Action selection** strategies
- **Learning from rewards**

Limitations of Bandits

- **Single state**: All decisions in same context
- **Immediate rewards**: No delayed consequences
- **Independent actions**: Choices don't affect future situations

From Bandits to Full Reinforcement Learning

What Bandits Taught Us

- **Exploration vs Exploitation** trade-off
- **Value estimation** from experience
- **Action selection** strategies
- **Learning from rewards**

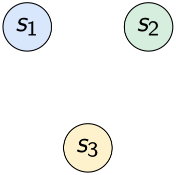
Limitations of Bandits

- **Single state**: All decisions in same context
- **Immediate rewards**: No delayed consequences
- **Independent actions**: Choices don't affect future situations
- **Stationary environment**: World doesn't change

The Jump to Full Reinforcement Learning

Full RL Removes Limitations

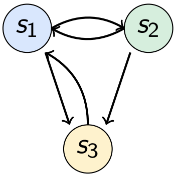
- **Multiple States**: Different situations



The Jump to Full Reinforcement Learning

Full RL Removes Limitations

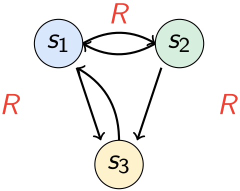
- **Multiple States**: Different situations
- **Sequential Decisions**: Actions affect future states



The Jump to Full Reinforcement Learning

Full RL Removes Limitations

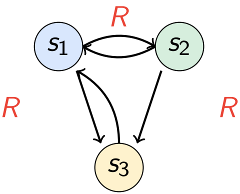
- **Multiple States**: Different situations
- **Sequential Decisions**: Actions affect future states
- **Delayed Rewards**: Long-term consequences



The Jump to Full Reinforcement Learning

Full RL Removes Limitations

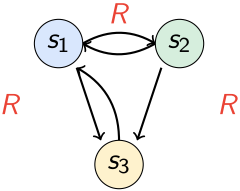
- **Multiple States**: Different situations
- **Sequential Decisions**: Actions affect future states
- **Delayed Rewards**: Long-term consequences
- **Dynamic Environment**: World changes with actions



The Jump to Full Reinforcement Learning

Full RL Removes Limitations

- **Multiple States**: Different situations
- **Sequential Decisions**: Actions affect future states
- **Delayed Rewards**: Long-term consequences
- **Dynamic Environment**: World changes with actions



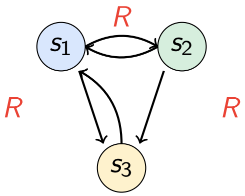
Mathematical Connection

- **Bandit**: $Q(a)$ - value of action a

The Jump to Full Reinforcement Learning

Full RL Removes Limitations

- **Multiple States**: Different situations
- **Sequential Decisions**: Actions affect future states
- **Delayed Rewards**: Long-term consequences
- **Dynamic Environment**: World changes with actions



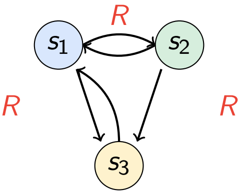
Mathematical Connection

- **Bandit**: $Q(a)$ - value of action a
- **Full RL**: $Q(s, a)$ - value of action a in state s

The Jump to Full Reinforcement Learning

Full RL Removes Limitations

- **Multiple States**: Different situations
- **Sequential Decisions**: Actions affect future states
- **Delayed Rewards**: Long-term consequences
- **Dynamic Environment**: World changes with actions



Mathematical Connection

- **Bandit**: $Q(a)$ - value of action a
- **Full RL**: $Q(s, a)$ - value of action a in state s

Every state in RL = separate bandit problem!

Summary

Next: Markov Decision Processes

- States, actions, transitions, and rewards
- Policies and value functions
- Bellman equations
- Dynamic programming solutions

The exploration-exploitation foundation is set!