Monte Carlo Methods

DEPARTMENT OF COMPUTER SCIENCE
ST. FRANCIS XAVIER UNIVERSITY

CSCI-531 - Reinforcement Learning

Fall 2025

From Model-Based to Model-Free Learning

Dynamic Programming Limitations

- Requires **complete knowledge** of the environment

From Model-Based to Model-Free Learning

Dynamic Programming Limitations

- Requires **complete knowledge** of the environment
- Needs full **transition function** $p(s'|s, a)$

From Model-Based to Model-Free Learning

Dynamic Programming Limitations

- Requires **complete knowledge** of the environment
- Needs full **transition function** $p(s'|s, a)$
- Often impractical in real-world scenarios

From Model-Based to Model-Free Learning

Dynamic Programming Limitations

- Requires **complete knowledge** of the environment
- Needs full **transition function** $p(s'|s,a)$
- Often impractical in real-world scenarios

Monte Carlo Solution

Learn from experience, not models!

From Model-Based to Model-Free Learning

Dynamic Programming Limitations

- Requires **complete knowledge** of the environment
- Needs full **transition function** $p(s'|s, a)$
- Often impractical in real-world scenarios

Monte Carlo Solution

Learn from experience, not models!

- Use **episodes** (complete trajectories)

From Model-Based to Model-Free Learning

Dynamic Programming Limitations

- Requires **complete knowledge** of the environment
- Needs full **transition function** $p(s'|s, a)$
- Often impractical in real-world scenarios

Monte Carlo Solution

Learn from experience, not models!

- Use **episodes** (complete trajectories)
- Average returns to estimate value functions

From Model-Based to Model-Free Learning

Dynamic Programming Limitations

- Requires **complete knowledge** of the environment
- Needs full **transition function** $p(s'|s,a)$
- Often impractical in real-world scenarios

Monte Carlo Solution

Learn from experience, not models!

- Use **episodes** (complete trajectories)
- Average returns to estimate value functions
- No need for transition probabilities

Real-World Motivation

Think About...

How would you learn to play a new video game?

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics
- Calculate optimal strategy

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics
- Calculate optimal strategy
- Then start playing

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics
- Calculate optimal strategy
- Then start playing

Monte Carlo Approach

- Jump in and play!

Real-World Motivation

Think About...
How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics
- Calculate optimal strategy
- **Then** start playing

Monte Carlo Approach

- Jump in and play!
- Try different actions

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics
- Calculate optimal strategy
- **Then** start playing

Monte Carlo Approach

- Jump in and play!
- Try different actions
- See what happens

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics
- Calculate optimal strategy
- Then start playing

Monte Carlo Approach

- Jump in and play!
- Try different actions
- See what happens
- Learn from experience

Real-World Motivation

Think About...

How would you learn to play a new video game?

Traditional DP Approach

- Read the entire manual
- Study all game mechanics
- Calculate optimal strategy
- Then start playing

Monte Carlo Approach

- Jump in and play!
- Try different actions
- See what happens
- Learn from experience

Key Insight

Most real problems are more like the video game scenario!

The Monte Carlo Approach

Key Insight

Value = Expected Return

If we can collect many returns from a state, we can **average** them to estimate the true value!

The Monte Carlo Approach

Key Insight

Value = Expected Return

If we can collect many returns from a state, we can **average** them to estimate the true value!

Requirements

- Problems must be **episodic**

The Monte Carlo Approach

Key Insight

Value = Expected Return

If we can collect many returns from a state, we can **average** them to estimate the true value!

Requirements

- Problems must be **episodic**
- Episodes must eventually **terminate**

The Monte Carlo Approach

Key Insight

Value = Expected Return

If we can collect many returns from a state, we can **average** them to estimate the true value!

Requirements

- Problems must be **episodic**
- Episodes must eventually **terminate**
- Can still interact with environment (but don't need model)

The Monte Carlo Approach

Key Insight

Value = Expected Return

If we can collect many returns from a state, we can **average** them to estimate the true value!

Requirements

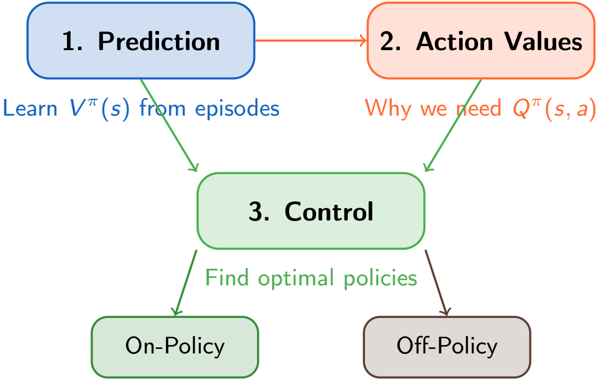
- Problems must be **episodic**
- Episodes must eventually **terminate**
- Can still interact with environment (but don't need model)

Important Note

Model still needed for interaction, but not for learning!

Learning Roadmap: Our Journey Through Monte Carlo

Where We're Going



Monte Carlo Prediction: The Basic Idea

The Process

1. Follow policy π to generate **episodes**

Monte Carlo Prediction: The Basic Idea

The Process

1. Follow policy π to generate **episodes**
2. For each visit to state s, record the **return** G_t

Monte Carlo Prediction: The Basic Idea

The Process

1. Follow policy π to generate **episodes**
2. For each visit to state s, record the **return** G_t
3. Estimate $v_\pi(s)$ as the **average** of all returns from s

Monte Carlo Prediction: The Basic Idea

The Process

1. Follow policy π to generate **episodes**
2. For each visit to state s, record the **return** G_t
3. Estimate $v_\pi(s)$ as the **average** of all returns from s

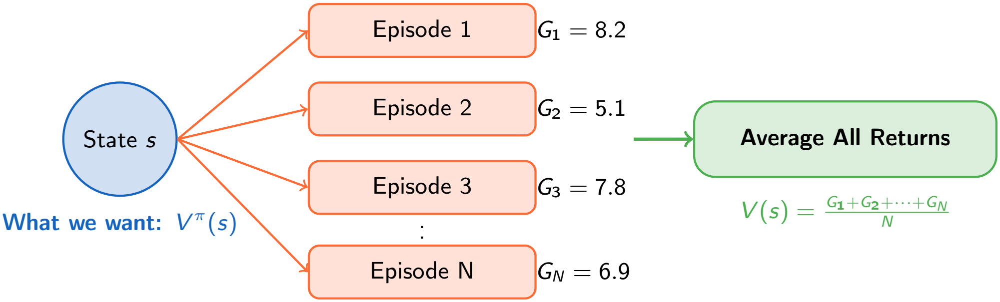
Mathematical Foundation

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

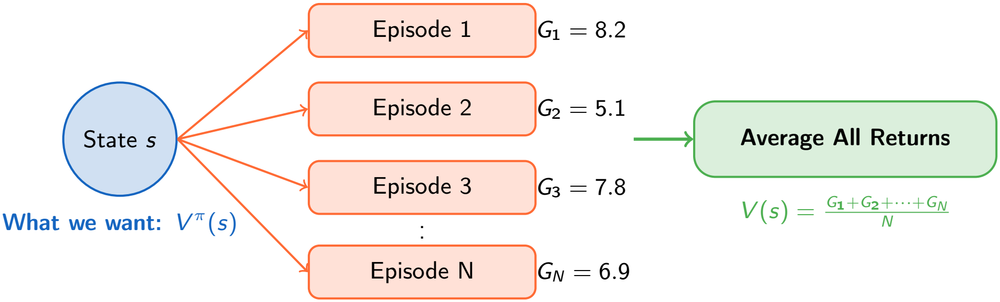
$$V(s) \approx \frac{1}{N} \sum_{i=1}^{N} G_i$$

where G_i are returns from N visits to state s.

Episode Collection: The Visual Process



Episode Collection: The Visual Process



Key Insight

More episodes $\Rightarrow$ Better estimate of true value!

Monte Carlo Prediction: The Basic Idea

Mathematical Foundation

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

$$V(s) \approx \frac{1}{N} \sum_{i=1}^{N} G_i$$

where G_i are returns from N visits to state s.

Monte Carlo Prediction: The Basic Idea

Mathematical Foundation

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

$$V(s) \approx \frac{1}{N} \sum_{i=1}^{N} G_i$$

where G_i are returns from N visits to state s.

Convergence Guarantee

As $N \to \infty$, $V(s) \to v_\pi(s)$ by the **Law of Large Numbers**

Concrete Example: Blackjack

Consider playing Blackjack with a simple policy:

- Hit if hand value < 17
- Stand if hand value ≥ 17

Concrete Example: Blackjack

Consider playing Blackjack with a simple policy:

- Hit if hand value < 17
- Stand if hand value ≥ 17

Sample Episodes from State "Hand = 16"

Episode	Outcome	Return
1	Bust	-1
2	Win	+1
3	Bust	-1
4	Bust	-1
5	Lose	-1

Concrete Example: Blackjack

Worked Example and Value Estimate

Worked episodes (hand = 16), assume $\gamma = 1$:

- Episode 1: Bust $\rightarrow R = -1 \Rightarrow G = -1$
- Episode 2: Win $\rightarrow R = +1 \Rightarrow G = +1$
- Episode 3: Bust $\rightarrow R = -1 \Rightarrow G = -1$
- Episode 4: Bust $\rightarrow R = -1 \Rightarrow G = -1$
- Episode 5: Lose $\rightarrow R = -1 \Rightarrow G = -1$

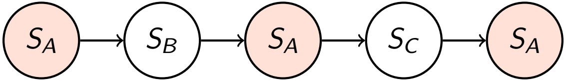
Averaging these returns:

$$V(\text{Hand} = 16) \approx \frac{-1 + 1 + (-1) + (-1) + (-1)}{5} = \textbf{-0.6}.$$

First-Visit vs Every-Visit Monte Carlo

A Key Decision

If we visit the same state multiple times in an episode, should we count all visits or just the first one?



State S_A visited 3 times

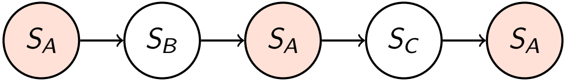
First-Visit vs Every-Visit Monte Carlo

A Key Decision

If we visit the same state multiple times in an episode, should we count all visits or just the first one?

First-Visit MC

- Only count the **first** visit to state s in each episode



State S_A visited 3 times

First-Visit vs Every-Visit Monte Carlo

A Key Decision

If we visit the same state multiple times in an episode, should we count all visits or just the first one?

First-Visit MC

- Only count the **first** visit to state s in each episode
- Simpler to analyze theoretically

$$S_A \rightarrow S_B \rightarrow S_A \rightarrow S_C \rightarrow S_A$$

State S_A visited 3 times

First-Visit vs Every-Visit Monte Carlo

A Key Decision

If we visit the same state multiple times in an episode, should we count all visits or just the first one?

First-Visit MC

- Only count the **first** visit to state s in each episode
- Simpler to analyze theoretically
- Guaranteed to converge

$$S_A \rightarrow S_B \rightarrow S_A \rightarrow S_C \rightarrow S_A$$

State S_A visited 3 times

First-Visit vs Every-Visit Monte Carlo

A Key Decision

If we visit the same state multiple times in an episode, should we count all visits or just the first one?

First-Visit MC

- Only count the **first** visit to state s in each episode
- Simpler to analyze theoretically
- Guaranteed to converge

Every-Visit MC

- Count **every** visit to state s in each episode

$$S_A \rightarrow S_B \rightarrow S_A \rightarrow S_C \rightarrow S_A$$

State S_A visited 3 times

First-Visit vs Every-Visit Monte Carlo

A Key Decision

If we visit the same state multiple times in an episode, should we count all visits or just the first one?

First-Visit MC

- Only count the **first** visit to state s in each episode
- Simpler to analyze theoretically
- Guaranteed to converge

Every-Visit MC

- Count **every** visit to state s in each episode
- Often converges faster in practice

$$S_A \rightarrow S_B \rightarrow S_A \rightarrow S_C \rightarrow S_A$$

State S_A visited 3 times

First-Visit vs Every-Visit Monte Carlo

A Key Decision

If we visit the same state multiple times in an episode, should we count all visits or just the first one?

First-Visit MC

- Only count the **first** visit to state s in each episode
- Simpler to analyze theoretically
- Guaranteed to converge

Every-Visit MC

- Count **every** visit to state s in each episode
- Often converges faster in practice
- More data per episode

$$S_A \rightarrow S_B \rightarrow S_A \rightarrow S_C \rightarrow S_A$$

State S_A visited 3 times

First-Visit vs Every-Visit — Small numeric example

Episode

Consider the episode (with $\gamma = 1$):

$$S_A \xrightarrow{2} S_B \xrightarrow{1} S_A \xrightarrow{3} S_C \xrightarrow{0}$$

- First-Visit for S_A: use return from its first occurrence: $G_0 = 2 + 1 + 3 + 0 = 6$.
- Every-Visit for S_A: include returns from each visit: first visit $G_0 = 6$, second visit (at time 2) $G_2 = 3 + 0 = 3$.

Discussion: First-Visit yields one sample per episode (independent), Every-Visit yields multiple correlated samples but more data per episode.

Why Does This Choice Matter?

Different Perspectives

- **First-Visit:** Each episode provides one independent sample per state

Why Does This Choice Matter?

Different Perspectives

- **First-Visit:** Each episode provides one independent sample per state
- **Every-Visit:** Each visit provides additional information about the state

Why Does This Choice Matter?

Different Perspectives

- **First-Visit:** Each episode provides one independent sample per state
- **Every-Visit:** Each visit provides additional information about the state

Theoretical Implications

- Both are unbiased estimators of $v_\pi(s)$

Why Does This Choice Matter?

Different Perspectives

- **First-Visit:** Each episode provides one independent sample per state
- **Every-Visit:** Each visit provides additional information about the state

Theoretical Implications

- Both are unbiased estimators of $v_\pi(s)$
- **First-Visit:** Independent samples, proven convergence

Why Does This Choice Matter?

Different Perspectives

- **First-Visit:** Each episode provides one independent sample per state
- **Every-Visit:** Each visit provides additional information about the state

Theoretical Implications

- Both are unbiased estimators of $v_\pi(s)$
- **First-Visit:** Independent samples, proven convergence
- **Every-Visit:** Correlated samples, often faster convergence

Why Does This Choice Matter?

Different Perspectives

- **First-Visit:** Each episode provides one independent sample per state
- **Every-Visit:** Each visit provides additional information about the state

Theoretical Implications

- Both are unbiased estimators of $v_\pi(s)$
- **First-Visit:** Independent samples, proven convergence
- **Every-Visit:** Correlated samples, often faster convergence

Practical Guidelines

- **Use First-Visit** for theoretical guarantees and textbook implementations

Why Does This Choice Matter?

Different Perspectives

- **First-Visit:** Each episode provides one independent sample per state
- **Every-Visit:** Each visit provides additional information about the state

Theoretical Implications

- Both are unbiased estimators of $v_\pi(s)$
- **First-Visit:** Independent samples, proven convergence
- **Every-Visit:** Correlated samples, often faster convergence

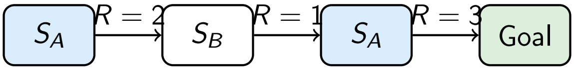
Practical Guidelines

- **Use First-Visit** for theoretical guarantees and textbook implementations
- **Use Every-Visit** for faster learning and limited computational budget

Interactive Example: Computing Returns

Activity

Given this episode with $\gamma = 0.9$:



Interactive Example: Computing Returns

Activity

Given this episode with $\gamma = 0.9$:

$$S_A \xrightarrow{R=2} S_B \xrightarrow{R=1} S_A \xrightarrow{R=3} \text{Goal}$$

Calculate Returns

- From first visit to S_A: $G_0 = 2 + 0.9 \times 1 + 0.9^2 \times 3 = 4.33$

Interactive Example: Computing Returns

Activity

Given this episode with $\gamma = 0.9$:

$$S_A \xrightarrow{R=2} S_B \xrightarrow{R=1} S_A \xrightarrow{R=3} \text{Goal}$$

Calculate Returns

- From first visit to S_A: $G_0 = 2 + 0.9 \times 1 + 0.9^2 \times 3 = 4.33$
- From second visit to S_A: $G_2 = 3$

Interactive Example: Computing Returns

Activity

Given this episode with $\gamma = 0.9$:

$$S_A \xrightarrow{R=2} S_B \xrightarrow{R=1} S_A \xrightarrow{R=3} \text{Goal}$$

Calculate Returns

- From first visit to S_A: $G_0 = 2 + 0.9 \times 1 + 0.9^2 \times 3 = 4.33$
- From second visit to S_A: $G_2 = 3$

Value Estimates

- First-Visit MC: $V(S_A) = 4.33$

Interactive Example: Computing Returns

Activity

Given this episode with $\gamma = 0.9$:

$$S_A \xrightarrow{R=2} S_B \xrightarrow{R=1} S_A \xrightarrow{R=3} \text{Goal}$$

Calculate Returns

- From first visit to S_A: $G_0 = 2 + 0.9 \times 1 + 0.9^2 \times 3 = 4.33$
- From second visit to S_A: $G_2 = 3$

Value Estimates

- First-Visit MC: $V(S_A) = 4.33$
- Every-Visit MC: $V(S_A) = \frac{4.33+3}{2} = 3.67$

Monte Carlo Prediction Algorithm

Algorithm First-Visit Monte Carlo Prediction

Input: Policy π, number of episodes N
Initialize: $V(s) \in \mathbb{R}$ arbitrarily, $\forall s \in S$; $Returns(s) \leftarrow$ empty list, $\forall s \in S$
for $episode = 1$ to N **do**
    Generate episode using π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    for $t = T-1, T-2, \ldots, 0$ **do**
        $G \leftarrow \gamma G + R_{t+1}$
        if $S_t \notin \{S_0, S_1, \ldots, S_{t-1}\}$ **then**
            Append G to $Returns(S_t)$
            $V(S_t) \leftarrow \text{average}(Returns(S_t))$
        end if
    end for
end for

Common Mistakes and Pitfalls

Pitfall #1: Wrong Direction

Why do we go backwards through the episode?

Common Mistakes and Pitfalls

Pitfall #1: Wrong Direction

Why do we go backwards through the episode?

- Returns are calculated from **future rewards**

Common Mistakes and Pitfalls

Pitfall #1: Wrong Direction

Why do we go backwards through the episode?

- Returns are calculated from **future rewards**
- Working backwards ensures we have all future rewards available

Common Mistakes and Pitfalls

Pitfall #1: Wrong Direction

Why do we go backwards through the episode?

- Returns are calculated from **future rewards**
- Working backwards ensures we have all future rewards available
- More efficient than recalculating from scratch each time

Common Mistakes and Pitfalls

Pitfall #2: Insufficient Episodes

How many episodes do we need?

Common Mistakes and Pitfalls

Pitfall #2: Insufficient Episodes

How many episodes do we need?

- Depends on state space size and variance

Common Mistakes and Pitfalls

Pitfall #2: Insufficient Episodes

How many episodes do we need?

- Depends on state space size and variance
- Start with 1000+ episodes for simple problems

Common Mistakes and Pitfalls

Pitfall #2: Insufficient Episodes

How many episodes do we need?

- Depends on state space size and variance
- Start with 1000+ episodes for simple problems
- Monitor convergence of value estimates

From Prediction to Control: What's Missing?

What We've Learned So Far

- We can estimate **state values** $V^{\pi}(s)$ using episodes

From Prediction to Control: What's Missing?

What We've Learned So Far

- We can estimate **state values** $V^{\pi}(s)$ using episodes
- We know how good each state is under policy π

From Prediction to Control: What's Missing?

What We've Learned So Far

- We can estimate **state values** $V^{\pi}(s)$ using episodes
- We know how good each state is under policy π
- We have a working Monte Carlo prediction algorithm

From Prediction to Control: What's Missing?

What We've Learned So Far

- We can estimate **state values** $V^\pi(s)$ using episodes
- We know how good each state is under policy π
- We have a working Monte Carlo prediction algorithm

The Next Challenge: Finding Better Policies

Question: If we know $V^\pi(s)$ for all states, can we immediately find a better policy?

From Prediction to Control: What's Missing?

What We've Learned So Far

- We can estimate **state values** $V^\pi(s)$ using episodes
- We know how good each state is under policy π
- We have a working Monte Carlo prediction algorithm

The Next Challenge: Finding Better Policies

Question: If we know $V^\pi(s)$ for all states, can we immediately find a better policy?

The Problem

- In DP: $\pi'(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r + \gamma V^\pi(s')]$

From Prediction to Control: What's Missing?

What We've Learned So Far

- We can estimate **state values** $V^\pi(s)$ using episodes
- We know how good each state is under policy π
- We have a working Monte Carlo prediction algorithm

The Next Challenge: Finding Better Policies

Question: If we know $V^\pi(s)$ for all states, can we immediately find a better policy?

The Problem

- In DP: $\pi'(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r + \gamma V^\pi(s')]$
- **But we don't know** $p(s'|s,a)$ in model-free settings!

From Prediction to Control: What's Missing?

What We've Learned So Far

- We can estimate **state values** $V^{\pi}(s)$ using episodes
- We know how good each state is under policy π
- We have a working Monte Carlo prediction algorithm

The Next Challenge: Finding Better Policies

Question: If we know $V^{\pi}(s)$ for all states, can we immediately find a better policy?

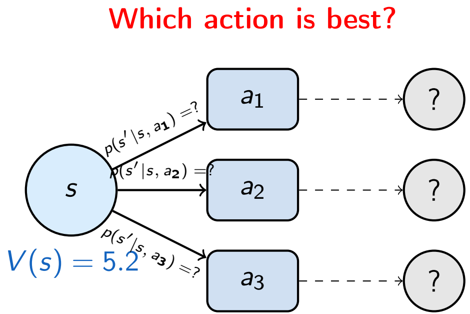
The Problem

- In DP: $\pi'(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r + \gamma V^{\pi}(s')]$
- **But we don't know** $p(s'|s,a)$ in model-free settings!
- We need a different approach...

Why Action Values in Model-Free Settings?

Discussion Question

If we have perfect state values $V^*(s)$, can we immediately determine the optimal policy without knowing the model?



Why Action Values in Model-Free Settings?

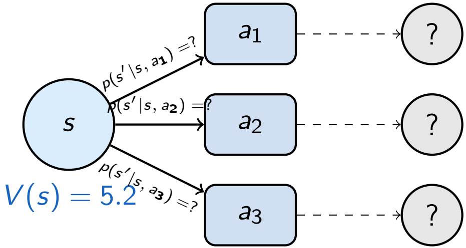
Discussion Question

If we have perfect state values $V^*(s)$, can we immediately determine the optimal policy without knowing the model?

Which action is best?

The Problem with State Values

- In DP: $\pi(s) = \arg\max_a \sum_{s'} p(s'|s, a)[r + \gamma V(s')]$



Why Action Values in Model-Free Settings?

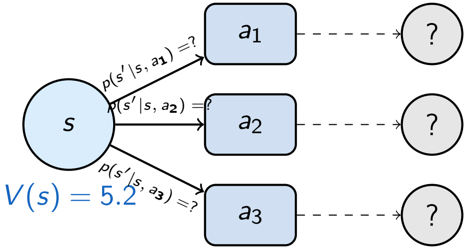
Discussion Question

If we have perfect state values $V^*(s)$, can we immediately determine the optimal policy without knowing the model?

The Problem with State Values

- In DP: $\pi(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r + \gamma V(s')]$
- **Requires** transition probabilities $p(s'|s,a)$

Which action is best?



Why Action Values in Model-Free Settings?

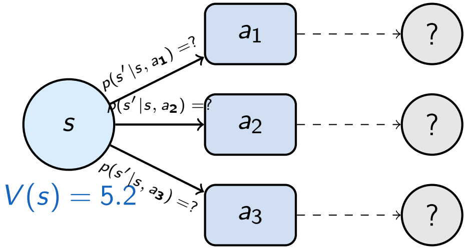
Discussion Question

If we have perfect state values $V^*(s)$, can we immediately determine the optimal policy without knowing the model?

The Problem with State Values

- In DP: $\pi(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r + \gamma V(s')]$
- **Requires** transition probabilities $p(s'|s,a)$
- Not available in model-free settings!

Which action is best?



The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Question

What should you do next?

The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Question

What should you do next?

Available Actions

- **Action A**: Move left

The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Question

What should you do next?

Available Actions

- **Action A**: Move left
- **Action B**: Move right

The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Question

What should you do next?

Available Actions

- **Action A**: Move left
- **Action B**: Move right
- **Action C**: Jump

The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Question

What should you do next?

Available Actions

- Action A: Move left
- Action B: Move right
- Action C: Jump

The Problem

- You know the state is good

The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Question

What should you do next?

Available Actions

- Action A: Move left
- Action B: Move right
- Action C: Jump

The Problem

- You know the state is good
- But which action leads to the best outcome?

The Fundamental Problem: State Values Aren't Enough

Imagine This Scenario

You're playing a game and reach this state:

You have **perfect knowledge** that being here is worth +5.2 points on average

The Question

What should you do next?

Available Actions

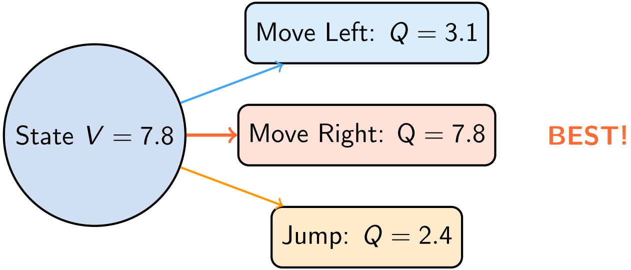
- **Action A**: Move left
- **Action B**: Move right
- **Action C**: Jump

The Problem

- You know the state is good
- But **which action** leads to the best outcome?
- State value doesn't tell you!

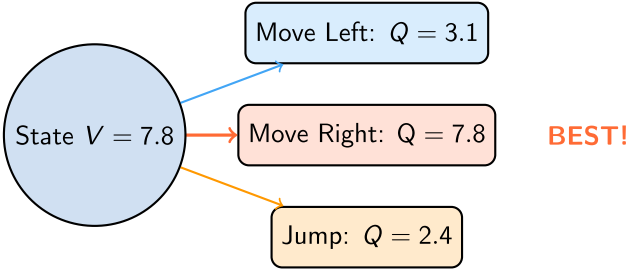
Action Values: The Missing Piece

What We Really Need



Action Values: The Missing Piece

What We Really Need

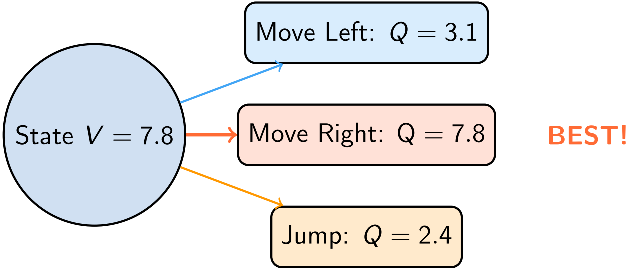


Now We Can Decide!

- **Move Right** has highest Q-value (7.8)

Action Values: The Missing Piece

What We Really Need



Now We Can Decide!

- **Move Right** has highest Q-value (7.8)
- **Simple policy**: $\pi(s) = \arg\max_a Q(s, a)$

Why This Changes Everything

The Key Insight

Action values give us direct decision-making power!

Why This Changes Everything

The Key Insight

Action values give us direct decision-making power!

The Difference

- **State values $V(s)$**: Know how good states are, still need model for decisions

Why This Changes Everything

The Key Insight

Action values give us direct decision-making power!

The Difference

- **State values** $V(s)$: Know how good states are, still need model for decisions
- **Action values** $Q(s, a)$: Know how good actions are, can decide immediately

Why This Changes Everything

The Key Insight

Action values give us direct decision-making power!

The Difference

- **State values** $V(s)$: Know how good states are, still need model for decisions
- **Action values** $Q(s, a)$: Know how good actions are, can decide immediately
- **Result**: Model-free decision making!

Why This Changes Everything

The Key Insight

Action values give us direct decision-making power!

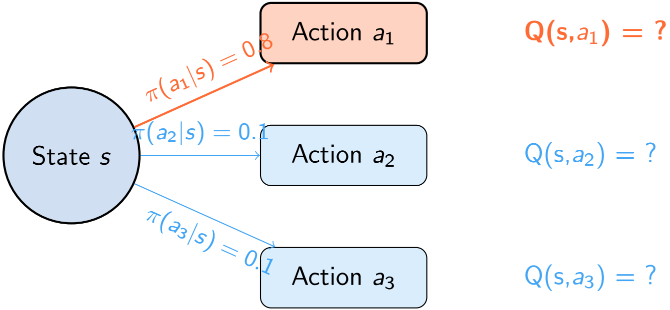
The Difference

- **State values** $V(s)$: Know how good states are, still need model for decisions
- **Action values** $Q(s, a)$: Know how good actions are, can decide immediately
- **Result**: Model-free decision making!

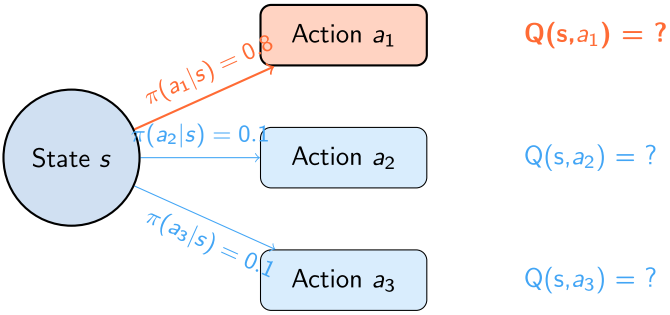
New Challenge

Exploration Problem: All actions must be visited to get good estimates!

The Exploration Problem



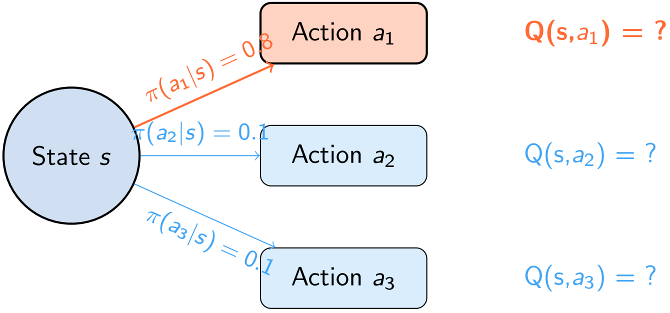
The Exploration Problem



The Dilemma

- **Greedy action** gets lots of data → good estimate

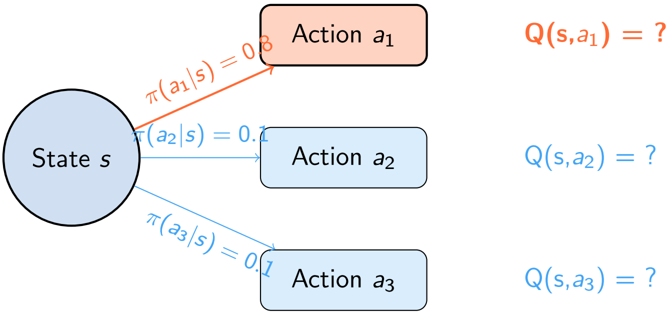
The Exploration Problem



The Dilemma

- **Greedy action** gets lots of data $\rightarrow$ good estimate
- **Other actions** get little data $\rightarrow$ poor estimates

The Exploration Problem



The Dilemma

- **Greedy action** gets lots of data $\rightarrow$ good estimate
- **Other actions** get little data $\rightarrow$ poor estimates
- How do we know if unexplored actions are actually better?

Solutions to the Exploration Problem

Option 1: Exploring Starts

- Every episode starts with random state-action pair

Solutions to the Exploration Problem

Option 1: Exploring Starts

- Every episode starts with random state-action pair
- Guarantees all (s, a) pairs are visited

Solutions to the Exploration Problem

Option 1: Exploring Starts

- Every episode starts with random state-action pair
- Guarantees all (s, a) pairs are visited
- **Problem:** Often not practical in real applications

Solutions to the Exploration Problem

Option 1: Exploring Starts

- Every episode starts with random state-action pair
- Guarantees all (s, a) pairs are visited
- **Problem:** Often not practical in real applications

Option 2: Soft Policies (Our Focus)

- Maintain **non-zero probability** for all actions

Solutions to the Exploration Problem

Option 1: Exploring Starts

- Every episode starts with random state-action pair
- Guarantees all (s, a) pairs are visited
- **Problem:** Often not practical in real applications

Option 2: Soft Policies (Our Focus)

- Maintain **non-zero probability** for all actions
- Balance exploration and exploitation

Solutions to the Exploration Problem

Option 1: Exploring Starts

- Every episode starts with random state-action pair
- Guarantees all (s, a) pairs are visited
- **Problem:** Often not practical in real applications

Option 2: Soft Policies (Our Focus)

- Maintain **non-zero probability** for all actions
- Balance exploration and exploitation
- Example: ϵ**-greedy policies**

On-Policy Monte Carlo Control

The Approach

- Learn the value of the same policy we're following

On-Policy Monte Carlo Control

The Approach

- Learn the value of the **same policy** we're following
- Use **ϵ-greedy** policy for exploration

On-Policy Monte Carlo Control

The Approach

- Learn the value of the **same policy** we're following
- Use ϵ-**greedy** policy for exploration
- Gradually improve policy based on learned action values

On-Policy Monte Carlo Control

The Approach

- Learn the value of the **same policy** we're following
- Use **ϵ-greedy** policy for exploration
- Gradually improve policy based on learned action values

ϵ-Greedy Policy

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \arg\max_a Q(s,a) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

On-Policy Monte Carlo Control

ϵ-Greedy Policy

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \arg\max_a Q(s, a) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

On-Policy Monte Carlo Control

ϵ-Greedy Policy

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \arg\max_a Q(s, a) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

Key Properties

- **Soft policy:** All actions have non-zero probability

On-Policy Monte Carlo Control

ϵ-Greedy Policy

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = \arg\max_a Q(s, a) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

Key Properties

- **Soft policy:** All actions have non-zero probability
- **Improvement guarantee:** Any ϵ-greedy policy w.r.t. Q_π is better than π

ϵ-Greedy Policy Example

Interactive Example

State s has 3 actions with $\epsilon = 0.2$:

$$Q(s, a_1) = 5, \; Q(s, a_2) = 3, \; Q(s, a_3) = 1$$

ϵ-Greedy Policy Example

Interactive Example

State s has 3 actions with $\epsilon = 0.2$:

$$Q(s, a_1) = 5, \; Q(s, a_2) = 3, \; Q(s, a_3) = 1$$

Calculate ϵ-greedy probabilities:

- Best action: a_1 (greedy action)

ϵ-Greedy Policy Example

Interactive Example

State s has 3 actions with $\epsilon = 0.2$:

$$Q(s, a_1) = 5,\ Q(s, a_2) = 3,\ Q(s, a_3) = 1$$

Calculate ϵ-greedy probabilities:

- Best action: a_1 (greedy action)
- $\pi(a_1|s) = 1 - 0.2 + \frac{0.2}{3} = 0.8 + 0.067 = 0.867$

ϵ-Greedy Policy Example

Interactive Example

State s has 3 actions with $\epsilon = 0.2$:

$$Q(s, a_1) = 5,\ Q(s, a_2) = 3,\ Q(s, a_3) = 1$$

Calculate ϵ-greedy probabilities:

- Best action: a_1 (greedy action)
- $\pi(a_1|s) = 1 - 0.2 + \frac{0.2}{3} = 0.8 + 0.067 = 0.867$
- $\pi(a_2|s) = \frac{0.2}{3} = 0.067$

ϵ-Greedy Policy Example

Interactive Example

State s has 3 actions with $\epsilon = 0.2$:

$$Q(s, a_1) = 5,\ Q(s, a_2) = 3,\ Q(s, a_3) = 1$$

Calculate ϵ-greedy probabilities:

- Best action: a_1 (greedy action)
- $\pi(a_1|s) = 1 - 0.2 + \frac{0.2}{3} = 0.8 + 0.067 = 0.867$
- $\pi(a_2|s) = \frac{0.2}{3} = 0.067$
- $\pi(a_3|s) = \frac{0.2}{3} = 0.067$

ϵ-Greedy Policy Example

Interactive Example

State s has 3 actions with $\epsilon = 0.2$:

$$Q(s, a_1) = 5,\ Q(s, a_2) = 3,\ Q(s, a_3) = 1$$

Calculate ϵ-greedy probabilities:

- Best action: a_1 (greedy action)
- $\pi(a_1|s) = 1 - 0.2 + \frac{0.2}{3} = 0.8 + 0.067 = 0.867$
- $\pi(a_2|s) = \frac{0.2}{3} = 0.067$
- $\pi(a_3|s) = \frac{0.2}{3} = 0.067$

ϵ-Greedy Policy Example

Interactive Example

State s has 3 actions with $\epsilon = 0.2$:

$$Q(s, a_1) = 5,\ Q(s, a_2) = 3,\ Q(s, a_3) = 1$$

Calculate ϵ-greedy probabilities:

- Best action: a_1 (greedy action)
- $\pi(a_1|s) = 1 - 0.2 + \frac{0.2}{3} = 0.8 + 0.067 = 0.867$
- $\pi(a_2|s) = \frac{0.2}{3} = 0.067$
- $\pi(a_3|s) = \frac{0.2}{3} = 0.067$

What happens as we learn and Q values change?

Algorithm On-Policy First-Visit MC Control

Initialize: $Q(s, a) \in \mathbb{R}$ arbitrarily, $\forall s \in S, a \in A$
Initialize: $\pi \leftarrow$ arbitrary ϵ-soft policy; $Returns(s, a) \leftarrow$ empty list, $\forall s \in S, a \in A$
for each episode **do**
  Generate episode using π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for $t = T-1, T-2, \ldots, 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if $(S_t, A_t) \notin \{(S_0, A_0), \ldots, (S_{t-1}, A_{t-1})\}$ **then**
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
      $A^* \leftarrow \arg\max_a Q(S_t, a)$
      for all $a \in A$ **do**

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \text{if } a = A^* \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

      end for

Concrete Example Part 1: Q-Value Update

Current Situation

State: Hand = 16, Actions: Hit or Stand

Action	**Current Q-value**	**Returns History**
Hit	$Q(16, \text{Hit}) = 0.2$	$[0.1, 0.3]$
Stand	$Q(16, \text{Stand}) = 0.5$	$[0.5]$

Concrete Example Part 1: Q-Value Update

Current Situation

State: Hand $= 16$, Actions: Hit or Stand

Action	Current Q-value	Returns History
Hit	$Q(16, \text{Hit}) = 0.2$	$[0.1, 0.3]$
Stand	$Q(16, \text{Stand}) = 0.5$	$[0.5]$

Episode Update

- Chose Hit, got return $G = 0.7$

Concrete Example Part 1: Q-Value Update

Current Situation

State: Hand = 16, Actions: Hit or Stand

Action	Current Q-value	Returns History
Hit	$Q(16, \text{Hit}) = 0.2$	$[0.1, 0.3]$
Stand	$Q(16, \text{Stand}) = 0.5$	$[0.5]$

Episode Update

- Chose Hit, got return $G = 0.7$
- $\mathit{Returns}(16, \text{Hit}) = [0.1, 0.3, 0.7]$

Concrete Example Part 1: Q-Value Update

Current Situation

State: Hand = 16, Actions: Hit or Stand

Action	Current Q-value	Returns History
Hit	$Q(16, \text{Hit}) = 0.2$	$[0.1, 0.3]$
Stand	$Q(16, \text{Stand}) = 0.5$	$[0.5]$

Episode Update

- Chose Hit, got return $G = 0.7$
- $Returns(16, \text{Hit}) = [0.1, 0.3, 0.7]$
- $Q(16, \text{Hit}) = \frac{0.1+0.3+0.7}{3} = 0.37$

Concrete Example Part 1: Q-Value Update

Current Situation

State: Hand = 16, Actions: Hit or Stand

Action	Current Q-value	Returns History
Hit	$Q(16, \text{Hit}) = 0.2$	$[0.1, 0.3]$
Stand	$Q(16, \text{Stand}) = 0.5$	$[0.5]$

Episode Update

- Chose Hit, got return $G = 0.7$
- $Returns(16, \text{Hit}) = [0.1, 0.3, 0.7]$
- $Q(16, \text{Hit}) = \frac{0.1+0.3+0.7}{3} = 0.37$
- **Result:** Hit improved from 0.2 to 0.37, Stand stays 0.5

Concrete Example Part 2: Policy Update

Current Q-Values

Action	Q-value
Hit	$Q(16, \text{Hit}) = 0.37$
Stand	$Q(16, \text{Stand}) = 0.5$

Concrete Example Part 2: Policy Update

Current Q-Values

Action	Q-value
Hit	$Q(16, \text{Hit}) = 0.37$
Stand	$Q(16, \text{Stand}) = 0.5$

Policy Update with $\epsilon = 0.1$:

- Best action: $\arg\max\{0.37, 0.5\} =$ Stand

Concrete Example Part 2: Policy Update

Current Q-Values

Action	**Q-value**
Hit	$Q(16, \text{Hit}) = 0.37$
Stand	$Q(16, \text{Stand}) = 0.5$

Policy Update with $\epsilon = 0.1$:

- Best action: $\arg\max\{0.37, 0.5\} =$ Stand
- $\pi(\text{Stand}|16) = 1 - 0.1 + \frac{0.1}{2} = 0.95$

Concrete Example Part 2: Policy Update

Current Q-Values

Action	Q-value
Hit	$Q(16, \text{Hit}) = 0.37$
Stand	$Q(16, \text{Stand}) = 0.5$

Policy Update with $\epsilon = 0.1$:

- Best action: $\arg\max\{0.37, 0.5\} = \text{Stand}$
- $\pi(\text{Stand}|16) = 1 - 0.1 + \frac{0.1}{2} = 0.95$
- $\pi(\text{Hit}|16) = \frac{0.1}{2} = 0.05$

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

The Issue

- Can only find best policy among **ϵ-soft policies**

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

The Issue

- Can only find best policy among ϵ-**soft policies**
- Always includes some exploration (suboptimal actions)

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

The Issue

- Can only find best policy among **ϵ-soft policies**
- Always includes some exploration (suboptimal actions)
- Cannot learn truly **optimal deterministic** policy

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

The Issue

- Can only find best policy among **ϵ-soft policies**
- Always includes some exploration (suboptimal actions)
- Cannot learn truly **optimal deterministic** policy

Off-Policy Solution

- Use two policies:

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

The Issue

- Can only find best policy among **ϵ-soft policies**
- Always includes some exploration (suboptimal actions)
- Cannot learn truly **optimal deterministic** policy

Off-Policy Solution

- Use two policies:
  - Target Policy π: The policy we want to learn (can be deterministic)

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

The Issue

- Can only find best policy among **ϵ-soft policies**
- Always includes some exploration (suboptimal actions)
- Cannot learn truly **optimal deterministic** policy

Off-Policy Solution

- Use two policies:
  - Target Policy π: The policy we want to learn (can be deterministic)
  - Behavior Policy b: The policy we follow for exploration

Off-Policy Monte Carlo: Motivation

Limitation of On-Policy Methods

What's the best policy on-policy MC can find?

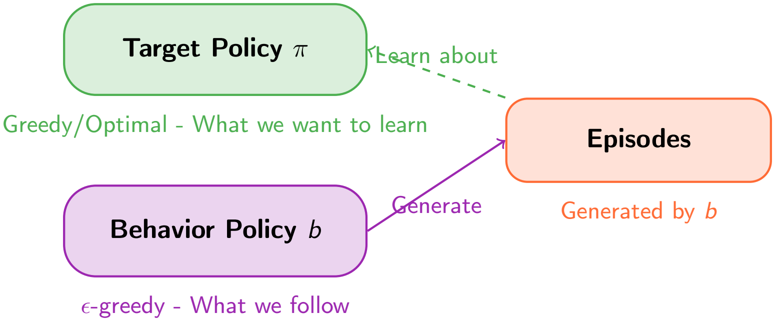
The Issue

- Can only find best policy among **ϵ-soft policies**
- Always includes some exploration (suboptimal actions)
- Cannot learn truly **optimal deterministic** policy

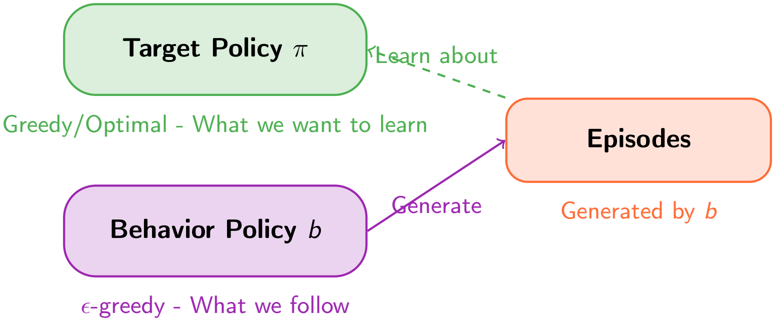
Off-Policy Solution

- Use two policies:
  - Target Policy π: The policy we want to learn (can be deterministic)
  - Behavior Policy b: The policy we follow for exploration
- Learn about π using data generated by b

Off-Policy Concept Visualization



Off-Policy Concept Visualization



Coverage Assumption

$\pi(a|s) > 0 \Rightarrow b(a|s) > 0$ for all s, a

Why is this assumption crucial?

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

The Problem

- You mostly have gameplay videos from **casual players** (not speedrunners)

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

The Problem

- You mostly have gameplay videos from **casual players** (not speedrunners)
- Casual players use "safe strategy" 80% of the time

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

The Problem

- You mostly have gameplay videos from **casual players** (not speedrunners)
- Casual players use "safe strategy" 80% of the time
- Speedrunners would use "risky strategy" 80% of the time

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

The Problem

- You mostly have gameplay videos from **casual players** (not speedrunners)
- Casual players use "safe strategy" 80% of the time
- Speedrunners would use "risky strategy" 80% of the time
- How do you estimate speedrunner performance?

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

The Problem

- You mostly have gameplay videos from **casual players** (not speedrunners)
- Casual players use "safe strategy" 80% of the time
- Speedrunners would use "risky strategy" 80% of the time
- How do you estimate speedrunner performance?

The Solution: Weight the Gameplay

- Casual player videos are **over-represented** for safe moves

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

The Problem

- You mostly have gameplay videos from **casual players** (not speedrunners)
- Casual players use "safe strategy" 80% of the time
- Speedrunners would use "risky strategy" 80% of the time
- How do you estimate speedrunner performance?

The Solution: Weight the Gameplay

- Casual player videos are **over-represented** for safe moves
- Weight risky moves by $\frac{0.80}{0.20} = 4.0$

Simple Analogy: YouTube Gaming Videos

Imagine You Want to Know...

How good are speedrunners at this game?

The Problem

- You mostly have gameplay videos from **casual players** (not speedrunners)
- Casual players use "safe strategy" 80% of the time
- Speedrunners would use "risky strategy" 80% of the time
- How do you estimate speedrunner performance?

The Solution: Weight the Gameplay

- Casual player videos are **over-represented** for safe moves
- Weight risky moves by $\frac{0.80}{0.20} = 4.0$
- This "up-weights" the rare risky gameplay we want to learn about

From Gaming to Reinforcement Learning

The Connection

YouTube Gaming	**RL Off-Policy**
Casual players	Behavior policy b
Speedrunners	Target policy π
Strategy choices	Episode actions
Game scores	Episode returns
Weight $= 0.80/0.20$	Weight $= \pi(a\|s)/b(a\|s)$

From Gaming to Reinforcement Learning

The Connection

YouTube Gaming	**RL Off-Policy**
Casual players	Behavior policy b
Speedrunners	Target policy π
Strategy choices	Episode actions
Game scores	Episode returns
Weight $= 0.80/0.20$	Weight $= \pi(a\|s)/b(a\|s)$

Key Insight

When gameplay comes from the "wrong" player type, we reweight it to match the strategy we want to learn!

Importance Sampling Intuition

The Problem

We have episodes from policy b, but want to learn about policy π. How do we account for the difference?

Importance Sampling Intuition

The Problem

We have episodes from policy b, but want to learn about policy π. How do we account for the difference?

Intuitive Example

- Suppose b takes action a_1 with probability 0.8

Importance Sampling Intuition

The Problem

We have episodes from policy b, but want to learn about policy π. How do we account for the difference?

Intuitive Example

- Suppose b takes action a_1 with probability 0.8
- But π would take a_1 with probability 0.2

Importance Sampling Intuition

The Problem

We have episodes from policy b, but want to learn about policy π. How do we account for the difference?

Intuitive Example

- Suppose b takes action a_1 with probability 0.8
- But π would take a_1 with probability 0.2
- Episodes with a_1 are over-represented in our data

Importance Sampling Intuition

The Problem

We have episodes from policy b, but want to learn about policy π. How do we account for the difference?

Intuitive Example

- Suppose b takes action a_1 with probability 0.8
- But π would take a_1 with probability 0.2
- Episodes with a_1 are over-represented in our data
- We need to down-weight these episodes

Importance Sampling Intuition

The Problem

We have episodes from policy b, but want to learn about policy π. How do we account for the difference?

Intuitive Example

- Suppose b takes action a_1 with probability 0.8
- But π would take a_1 with probability 0.2
- Episodes with a_1 are over-represented in our data
- We need to down-weight these episodes

The Solution: Importance Sampling

Weight each episode by the ratio of its probability under π vs b

Importance Sampling Mathematics

Importance Sampling Ratio

$$
\begin{aligned}
\rho_{t:T-1} &= \frac{\text{Prob of trajectory under } \pi}{\text{Prob of trajectory under } b} \\
&= \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} \\
&= \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}
\end{aligned}
$$

Importance Sampling Mathematics

Importance Sampling Ratio

$$
\begin{aligned}
\rho_{t:T-1} &= \frac{\text{Prob of trajectory under } \pi}{\text{Prob of trajectory under } b} \\
&= \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} \\
&= \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}
\end{aligned}
$$

Key Insight

Transition probabilities cancel out. Only depends on policies.

Importance Sampling: Numerical Example

Episode Trajectory

Episode: $S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2$

Importance Sampling: Numerical Example

Episode Trajectory

Episode: $S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2$

Policy Probabilities

- At S_0: $\pi(A_0|S_0) = 0.1$, $b(A_0|S_0) = 0.3$

Importance Sampling: Numerical Example

Episode Trajectory

Episode: $S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2$

Policy Probabilities

- At S_0: $\pi(A_0|S_0) = 0.1$, $b(A_0|S_0) = 0.3$
- At S_1: $\pi(A_1|S_1) = 0.8$, $b(A_1|S_1) = 0.4$

Importance Sampling: Numerical Example

Episode Trajectory

Episode: $S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2$

Policy Probabilities

- At S_0: $\pi(A_0|S_0) = 0.1$, $b(A_0|S_0) = 0.3$
- At S_1: $\pi(A_1|S_1) = 0.8$, $b(A_1|S_1) = 0.4$

Importance Sampling Ratio

$$\begin{aligned}\rho &= \frac{\pi(A_0|S_0)}{b(A_0|S_0)} \times \frac{\pi(A_1|S_1)}{b(A_1|S_1)} \\ &= \frac{0.1}{0.3} \times \frac{0.8}{0.4} = \frac{1}{3} \times 2 = \frac{2}{3}\end{aligned}$$

Algorithm Off-Policy Monte Carlo Prediction

Initialize: $Q(s, a) \in \mathbb{R}$ arbitrarily, $\forall s \in S, a \in A$
Initialize: $C(s, a) \leftarrow 0$, $\forall s \in S, a \in A$
for each episode **do**
    $b \leftarrow$ any policy with coverage of π
    Generate episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$, $W \leftarrow 1$
    for $t = T-1, T-2, \ldots, 0$ and $W \neq 0$ **do**
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
        $W \leftarrow W \cdot \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
    end for
end for

Weighted Importance Sampling: Key Features

Implementation Details

- Uses **incremental implementation** with weighted average

Weighted Importance Sampling: Key Features

Implementation Details

- Uses **incremental implementation** with weighted average
- Stops if $W = 0$ (trajectory impossible under target policy)

Weighted Importance Sampling: Key Features

Implementation Details

- Uses **incremental implementation** with weighted average
- Stops if $W = 0$ (trajectory impossible under target policy)

Why This Matters

- **Efficiency**: No need to store all returns

Weighted Importance Sampling: Key Features

Implementation Details

- Uses **incremental implementation** with weighted average
- Stops if $W = 0$ (trajectory impossible under target policy)

Why This Matters

- **Efficiency**: No need to store all returns
- **Robustness**: Handles impossible trajectories gracefully

Weighted Importance Sampling: Key Features

Implementation Details

- Uses **incremental implementation** with weighted average
- Stops if $W = 0$ (trajectory impossible under target policy)

Why This Matters

- **Efficiency**: No need to store all returns
- **Robustness**: Handles impossible trajectories gracefully
- **Convergence**: Weighted averaging often has better properties

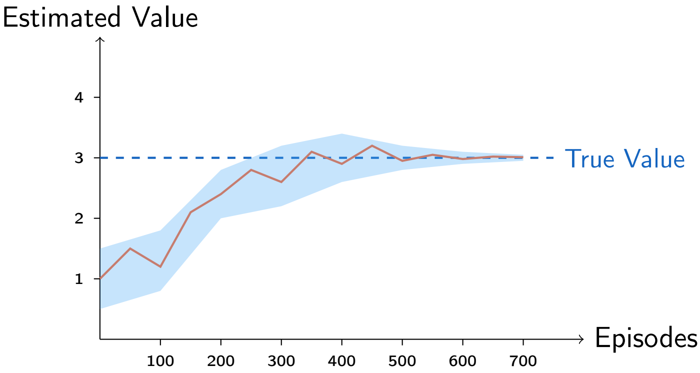
Algorithm Off-Policy Monte Carlo Control

Initialize: $\pi(s) \leftarrow \arg\max_a Q(s, a)$ (greedy policy)
for each episode **do**
  $b \leftarrow$ any policy with coverage of π
  Generate episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$, $W \leftarrow 1$
  for $t = T-1, T-2, \ldots, 0$ and $W \neq 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)}[G - Q(S_t, A_t)]$
    $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
    if $A_t \neq \pi(S_t)$ **then**
      break (exit inner loop)
    end if
    $W \leftarrow W \cdot \frac{1}{b(A_t|S_t)}$
  end for

Learning Progress: How Monte Carlo Converges

What Does Learning Look Like?

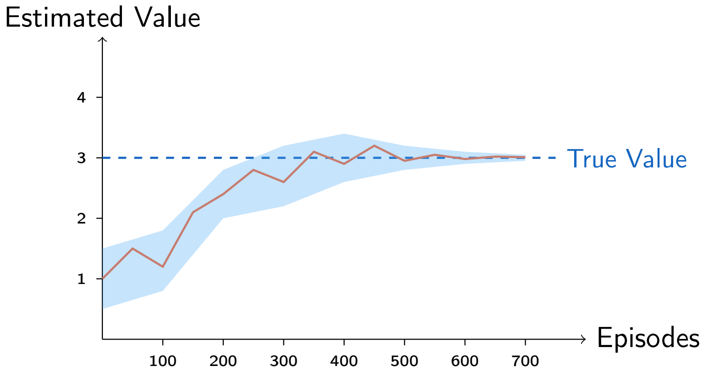
Let's visualize how our value estimates improve over time



Learning Progress: How Monte Carlo Converges

What Does Learning Look Like?

Let's visualize how our value estimates improve over time

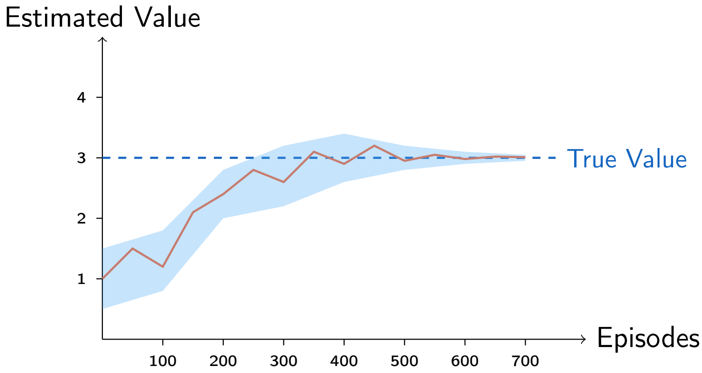


Key Observations

Learning Progress: How Monte Carlo Converges

What Does Learning Look Like?

Let's visualize how our value estimates improve over time

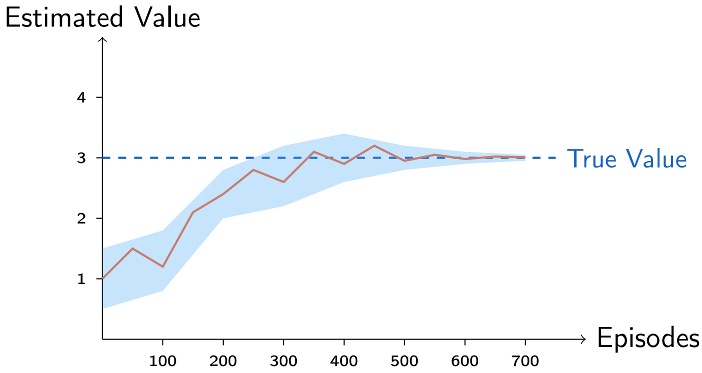


Key Observations

Learning Progress: How Monte Carlo Converges

What Does Learning Look Like?

Let's visualize how our value estimates improve over time



Key Observations

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

Slower Convergence:

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns

Slower Convergence:

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns
- Frequent state visits

Slower Convergence:

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns
- Frequent state visits
- Shorter episodes

Slower Convergence:

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns
- Frequent state visits
- Shorter episodes
- Deterministic rewards

Slower Convergence:

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns
- Frequent state visits
- Shorter episodes
- Deterministic rewards

Slower Convergence:

- High variance returns

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns
- Frequent state visits
- Shorter episodes
- Deterministic rewards

Slower Convergence:

- High variance returns
- Rare state visits

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns
- Frequent state visits
- Shorter episodes
- Deterministic rewards

Slower Convergence:

- High variance returns
- Rare state visits
- Long episodes

Practical Learning Considerations

What Affects Convergence Speed?

Faster Convergence:

- Low variance returns
- Frequent state visits
- Shorter episodes
- Deterministic rewards

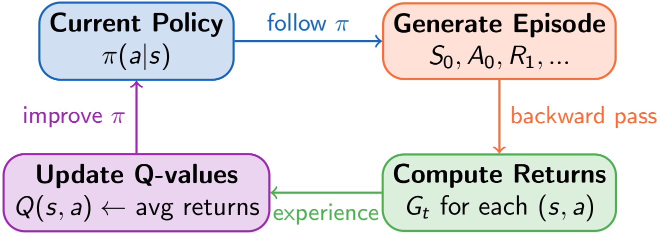
Slower Convergence:

- High variance returns
- Rare state visits
- Long episodes
- Stochastic environments

Monte Carlo Control: The Complete Loop

The Big Picture

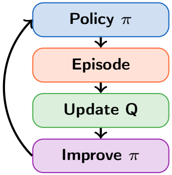
How do all the pieces fit together?



Repeat until convergence

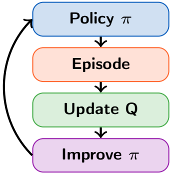
On-Policy vs Off-Policy Control Flow

On-Policy Monte Carlo



On-Policy vs Off-Policy Control Flow

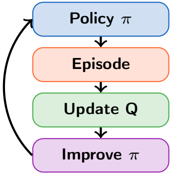
On-Policy Monte Carlo



- Same policy for **acting** and **learning**

On-Policy vs Off-Policy Control Flow

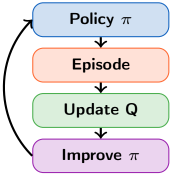
On-Policy Monte Carlo



- Same policy for **acting** and **learning**
- Simpler, more stable

On-Policy vs Off-Policy Control Flow

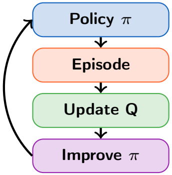
On-Policy Monte Carlo



- Same policy for acting and learning
- Simpler, more stable
- Slower convergence

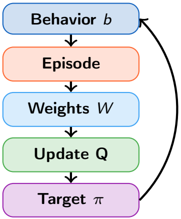
On-Policy vs Off-Policy Control Flow

On-Policy Monte Carlo



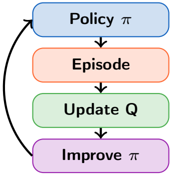
- Same policy for **acting** and **learning**
- Simpler, more stable
- Slower convergence

Off-Policy Monte Carlo



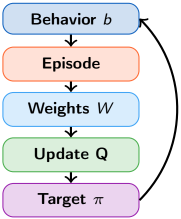
On-Policy vs Off-Policy Control Flow

On-Policy Monte Carlo



- Same policy for **acting** and **learning**
- Simpler, more stable
- Slower convergence

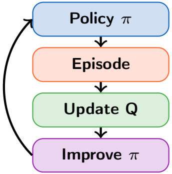
Off-Policy Monte Carlo



- Different policies for **acting** vs **learning**

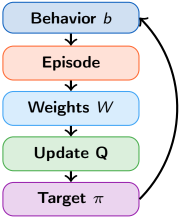
On-Policy vs Off-Policy Control Flow

On-Policy Monte Carlo



- Same policy for **acting** and **learning**
- Simpler, more stable
- Slower convergence

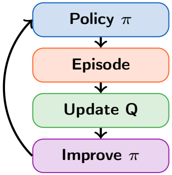
Off-Policy Monte Carlo



- Different policies for **acting** vs **learning**
- More complex (importance sampling)

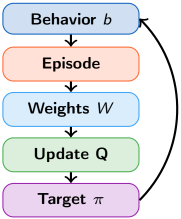
On-Policy vs Off-Policy Control Flow

On-Policy Monte Carlo



- Same policy for **acting** and **learning**
- Simpler, more stable
- Slower convergence

Off-Policy Monte Carlo



- Different policies for **acting** vs **learning**
- More complex (importance sampling)
- Better exploration potential

On-Policy vs Off-Policy: Trade-offs

On-Policy

Advantages:

- Simpler to understand
- Lower variance
- Faster convergence
- Easier to implement

Disadvantages:

- Limited to ϵ-soft policies
- Cannot learn deterministic optimal policy

Off-Policy

Advantages:

- Can learn optimal deterministic policy
- More general and powerful
- Can reuse data from any policy

Disadvantages:

- Higher variance
- Slower convergence
- More complex (importance sampling)

On-Policy vs Off-Policy: Trade-offs

On-Policy

Advantages:

- Simpler to understand
- Lower variance
- Faster convergence
- Easier to implement

Disadvantages:

- Limited to ϵ-soft policies
- Cannot learn deterministic optimal policy

Off-Policy

Advantages:

- Can learn optimal deterministic policy
- More general and powerful
- Can reuse data from any policy

Disadvantages:

- Higher variance
- Slower convergence
- More complex (importance sampling)

Discussion

When would you choose each approach?

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Choosing ϵ

- Start with $\epsilon = 0.1$ (10% exploration)

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Choosing ϵ

- Start with $\epsilon = 0.1$ (10% exploration)
- Decay ϵ over time: $\epsilon_t = \epsilon_0 / t$

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Choosing ϵ

- Start with $\epsilon = 0.1$ (10% exploration)
- Decay ϵ over time: $\epsilon_t = \epsilon_0 / t$
- Balance exploration vs exploitation needs

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Choosing ϵ

- Start with $\epsilon = 0.1$ (10% exploration)
- Decay ϵ over time: $\epsilon_t = \epsilon_0 / t$
- Balance exploration vs exploitation needs

Number of Episodes

- Monitor convergence of value estimates

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Choosing ϵ

- Start with $\epsilon = 0.1$ (10% exploration)
- Decay ϵ over time: $\epsilon_t = \epsilon_0 / t$
- Balance exploration vs exploitation needs

Number of Episodes

- Monitor convergence of value estimates
- Plot average returns over time

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Choosing ϵ

- Start with $\epsilon = 0.1$ (10% exploration)
- Decay ϵ over time: $\epsilon_t = \epsilon_0 / t$
- Balance exploration vs exploitation needs

Number of Episodes

- Monitor convergence of value estimates
- Plot average returns over time
- Consider computational budget

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Common Issues

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Common Issues

- High variance in off-policy methods

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Common Issues

- High variance in off-policy methods
- Slow convergence for large state spaces

Practical Considerations

Implementation Tips

What should you consider when implementing Monte Carlo methods?

Common Issues

- High variance in off-policy methods
- Slow convergence for large state spaces
- Need sufficient exploration

Key Takeaways

What We've Learned

1. **Model-Free Learning:** Can learn optimal policies without knowing environment dynamics

Key Takeaways

What We've Learned

1. **Model-Free Learning:** Can learn optimal policies without knowing environment dynamics
2. **Experience-Based:** Use episodes to estimate value functions through averaging

Key Takeaways

What We've Learned

1. **Model-Free Learning:** Can learn optimal policies without knowing environment dynamics
2. **Experience-Based:** Use episodes to estimate value functions through averaging
3. **Exploration is Critical:** Must visit all state-action pairs sufficiently often

Key Takeaways

What We've Learned

1. **Model-Free Learning:** Can learn optimal policies without knowing environment dynamics
2. **Experience-Based:** Use episodes to estimate value functions through averaging
3. **Exploration is Critical:** Must visit all state-action pairs sufficiently often
4. **Two Approaches:** On-policy (simple) vs Off-policy (powerful)

Key Takeaways

What We've Learned

1. **Model-Free Learning:** Can learn optimal policies without knowing environment dynamics
2. **Experience-Based:** Use episodes to estimate value functions through averaging
3. **Exploration is Critical:** Must visit all state-action pairs sufficiently often
4. **Two Approaches:** On-policy (simple) vs Off-policy (powerful)

Looking Ahead

- Monte Carlo requires **complete episodes**

Key Takeaways

What We've Learned

1. **Model-Free Learning:** Can learn optimal policies without knowing environment dynamics
2. **Experience-Based:** Use episodes to estimate value functions through averaging
3. **Exploration is Critical:** Must visit all state-action pairs sufficiently often
4. **Two Approaches:** On-policy (simple) vs Off-policy (powerful)

Looking Ahead

- Monte Carlo requires **complete episodes**
- What if episodes are very long or infinite?

Key Takeaways

What We've Learned

1. **Model-Free Learning:** Can learn optimal policies without knowing environment dynamics
2. **Experience-Based:** Use episodes to estimate value functions through averaging
3. **Exploration is Critical:** Must visit all state-action pairs sufficiently often
4. **Two Approaches:** On-policy (simple) vs Off-policy (powerful)

Looking Ahead

- Monte Carlo requires **complete episodes**
- What if episodes are very long or infinite?
- Next: **Temporal Difference Learning** - combine MC and DP benefits