Markov Decision Processes

DEPARTMENT OF COMPUTER SCIENCE
ST. FRANCIS XAVIER UNIVERSITY

CSCI-531 - Reinforcement Learning

Fall 2025

Beyond Multi-Armed Bandits

Bandits vs. Real-World Decisions

- **Bandits**: Each action is independent

Beyond Multi-Armed Bandits

Bandits vs. Real-World Decisions

- **Bandits**: Each action is independent
- **Real World**: Actions have lasting consequences

Beyond Multi-Armed Bandits

Bandits vs. Real-World Decisions

- **Bandits**: Each action is independent
- **Real World**: Actions have lasting consequences

Real-World Examples

- **Robot Learning**: Each step affects balance for future steps

Beyond Multi-Armed Bandits

Bandits vs. Real-World Decisions

- Bandits: Each action is independent
- Real World: Actions have lasting consequences

Real-World Examples

- **Robot Learning**: Each step affects balance for future steps
- **Drug Discovery**: Molecular modifications affect entire research trajectory

Beyond Multi-Armed Bandits

Bandits vs. Real-World Decisions

- **Bandits**: Each action is independent
- **Real World**: Actions have lasting consequences

Real-World Examples

- **Robot Learning**: Each step affects balance for future steps
- **Drug Discovery**: Molecular modifications affect entire research trajectory
- **Language Model Training**: Each decision influences thousands of future steps

Beyond Multi-Armed Bandits

Bandits vs. Real-World Decisions

- **Bandits**: Each action is independent
- **Real World**: Actions have lasting consequences

Real-World Examples

- **Robot Learning**: Each step affects balance for future steps
- **Drug Discovery**: Molecular modifications affect entire research trajectory
- **Language Model Training**: Each decision influences thousands of future steps
- **Algorithmic Trading**: Every trade changes portfolio state and risk

Beyond Multi-Armed Bandits

Bandits vs. Real-World Decisions

- **Bandits**: Each action is independent
- **Real World**: Actions have lasting consequences

Real-World Examples

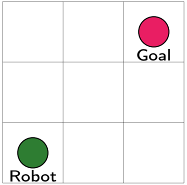
- **Robot Learning**: Each step affects balance for future steps
- **Drug Discovery**: Molecular modifications affect entire research trajectory
- **Language Model Training**: Each decision influences thousands of future steps
- **Algorithmic Trading**: Every trade changes portfolio state and risk

These are sequential decision processes with ripple effects through time

The Challenge of Sequential Decisions

Key Differences from Bandits

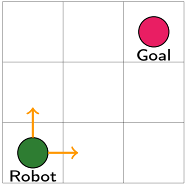
- **State matters**: Current situation affects future options



The Challenge of Sequential Decisions

Key Differences from Bandits

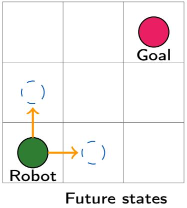
- **State matters**: Current situation affects future options
- **Actions have consequences**: Choices change the world



The Challenge of Sequential Decisions

Key Differences from Bandits

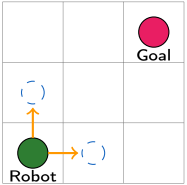
- **State matters**: Current situation affects future options
- **Actions have consequences**: Choices change the world
- **Delayed rewards**: Must trade off immediate vs future gains



The Challenge of Sequential Decisions

Key Differences from Bandits

- **State matters**: Current situation affects future options
- **Actions have consequences**: Choices change the world
- **Delayed rewards**: Must trade off immediate vs future gains
- **Uncertainty**: Outcomes are probabilistic

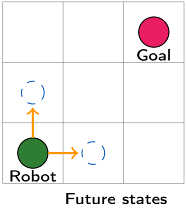


Future states

The Challenge of Sequential Decisions

Key Differences from Bandits

- **State matters**: Current situation affects future options
- **Actions have consequences**: Choices change the world
- **Delayed rewards**: Must trade off immediate vs future gains
- **Uncertainty**: Outcomes are probabilistic



We need a mathematical framework for sequential decisions

What is a Markov Decision Process?

MDP: Mathematical Framework

A Markov Decision Process models sequential decision-making under uncertainty

What is a Markov Decision Process?

MDP: Mathematical Framework

A Markov Decision Process models sequential decision-making under uncertainty

Key Properties

- **Sequential**: Multiple decisions over time

What is a Markov Decision Process?

MDP: Mathematical Framework

A Markov Decision Process models sequential decision-making under uncertainty

Key Properties

- **Sequential**: Multiple decisions over time
- **Consequential**: Each action affects future situations

What is a Markov Decision Process?

MDP: Mathematical Framework

A Markov Decision Process models sequential decision-making under uncertainty

Key Properties

- **Sequential**: Multiple decisions over time
- **Consequential**: Each action affects future situations
- **Uncertain**: Outcomes are probabilistic

What is a Markov Decision Process?

MDP: Mathematical Framework

A Markov Decision Process models sequential decision-making under uncertainty

Key Properties

- **Sequential**: Multiple decisions over time
- **Consequential**: Each action affects future situations
- **Uncertain**: Outcomes are probabilistic
- **Goal-oriented**: Maximize long-term reward

What is a Markov Decision Process?

MDP: Mathematical Framework

A Markov Decision Process models sequential decision-making under uncertainty

Key Properties

- **Sequential**: Multiple decisions over time
- **Consequential**: Each action affects future situations
- **Uncertain**: Outcomes are probabilistic
- **Goal-oriented**: Maximize long-term reward

MDPs provide the foundation for virtually all modern RL algorithms

Formal MDP Definition

MDP Tuple (S, A, T, R)

- S: Finite set of **states** (all possible situations)

Formal MDP Definition

MDP Tuple (S, A, T, R)

- S: Finite set of states (all possible situations)
- A: Finite set of actions (all choices available)

Formal MDP Definition

MDP Tuple (S, A, T, R)

- S: Finite set of states (all possible situations)
- A: Finite set of actions (all choices available)
- $T : S \times A \times S \rightarrow [0, 1]$: Transition function (world's physics)

Formal MDP Definition

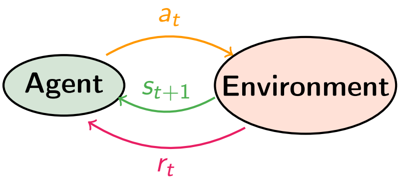
MDP Tuple (S, A, T, R)

- S: Finite set of **states** (all possible situations)
- A: Finite set of **actions** (all choices available)
- $T : S \times A \times S \to [0, 1]$: **Transition function** (world's physics)
- $R : S \times A \to \mathbb{R}$: **Reward function** (immediate feedback)

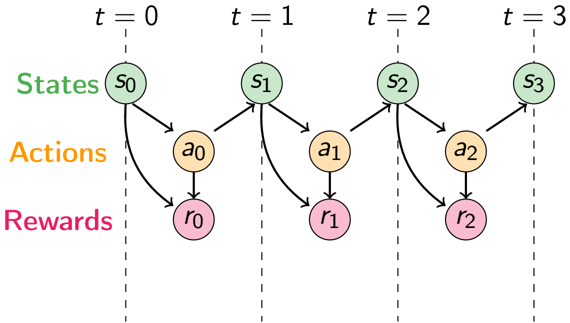
Formal MDP Definition

MDP Tuple (S, A, T, R)

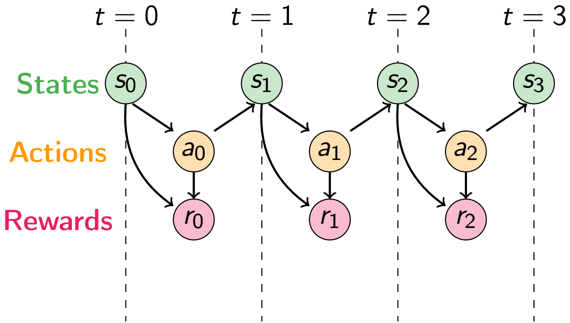
- S: Finite set of **states** (all possible situations)
- A: Finite set of **actions** (all choices available)
- $T : S \times A \times S \to [0, 1]$: **Transition function** (world's physics)
- $R : S \times A \to \mathbb{R}$: **Reward function** (immediate feedback)



The Agent-Environment Interaction Loop



The Agent-Environment Interaction Loop



Trajectory Generation

This process generates a **trajectory** or **episode**:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots$$

Transition Function: The World's Physics

Probabilistic Transitions

After taking action a_t in state s_t, next state s_{t+1} is determined by:

$$p(s'|s, a) = \text{Probability of moving to state } s' \text{ given state } s \text{ and action } a$$

Transition Function: The World's Physics

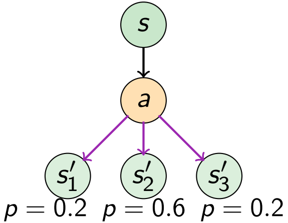
Probabilistic Transitions

After taking action a_t in state s_t, next state s_{t+1} is determined by:

$$p(s'|s,a) = \text{Probability of moving to state } s' \text{ given state } s \text{ and action } a$$

Why Probabilistic?

- **Robot motors**: Have noise and uncertainty



Transition Function: The World's Physics

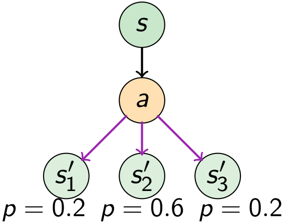
Probabilistic Transitions

After taking action a_t in state s_t, next state s_{t+1} is determined by:

$$p(s'|s,a) = \text{Probability of moving to state } s' \text{ given state } s \text{ and action } a$$

Why Probabilistic?

- **Robot motors**: Have noise and uncertainty
- **Market conditions**: Can change unexpectedly



Transition Function: The World's Physics

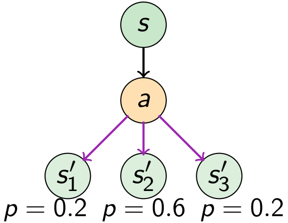
Probabilistic Transitions

After taking action a_t in state s_t, next state s_{t+1} is determined by:

$$p(s'|s,a) = \text{Probability of moving to state } s' \text{ given state } s \text{ and action } a$$

Why Probabilistic?

- **Robot motors**: Have noise and uncertainty
- **Market conditions**: Can change unexpectedly
- **Experiments**: May have measurement errors



Transition Function: The World's Physics

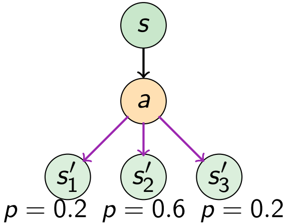
Probabilistic Transitions

After taking action a_t in state s_t, next state s_{t+1} is determined by:

$$p(s'|s, a) = \text{Probability of moving to state } s' \text{ given state } s \text{ and action } a$$

Why Probabilistic?

- **Robot motors**: Have noise and uncertainty
- **Market conditions**: Can change unexpectedly
- **Experiments**: May have measurement errors
- **Real world**: Is inherently stochastic



Transition Function: The World's Physics

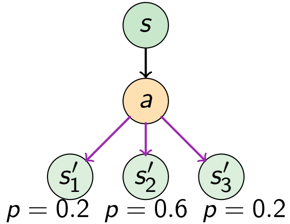
Probabilistic Transitions

After taking action a_t in state s_t, next state s_{t+1} is determined by:

$$p(s'|s, a) = \text{Probability of moving to state } s' \text{ given state } s \text{ and action } a$$

Why Probabilistic?

- **Robot motors**: Have noise and uncertainty
- **Market conditions**: Can change unexpectedly
- **Experiments**: May have measurement errors
- **Real world**: Is inherently stochastic



Note: $\sum_{s' \in S} p(s'|s, a) = 1$ for all $s \in S, a \in A$

The Markov Property

Markov Property Definition

The future depends only on the present, not on the past

Formally: All information needed to predict future states must be contained in the current state.

The Markov Property

Markov Property Definition

The future depends only on the present, not on the past

Formally: All information needed to predict future states must be contained in the current state.

Mathematical Expression

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, \ldots, S_0 = s_0) = P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

The Markov Property

Markov Property Definition

The future depends only on the present, not on the past

Formally: All information needed to predict future states must be contained in the current state.

Mathematical Expression

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, \ldots, S_0 = s_0) = P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

Examples

- **Markovian**: Chess position (complete board state)

The Markov Property

Markov Property Definition

The future depends only on the present, not on the past

Formally: All information needed to predict future states must be contained in the current state.

Mathematical Expression

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, \ldots, S_0 = s_0) = P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

Examples

- **Markovian**: Chess position (complete board state)
- **Non-Markovian**: "Take the third turn" (depends on path history)

Reward Function: The Learning Signal

Reward Function Purpose

$r(s, a)$ provides immediate feedback that guides the agent toward its goal

Reward Function: The Learning Signal

Reward Function Purpose

$r(s, a)$ provides immediate feedback that guides the agent toward its goal

Design Principles

- Rewards communicate **what** to achieve, not **how**

Reward Function: The Learning Signal

Reward Function Purpose

$r(s, a)$ provides immediate feedback that guides the agent toward its goal

Design Principles

- Rewards communicate **what** to achieve, not **how**
- Given immediately after each action

Reward Function: The Learning Signal

Reward Function Purpose

$r(s, a)$ provides immediate feedback that guides the agent toward its goal

Design Principles

- Rewards communicate **what** to achieve, not **how**
- Given immediately after each action
- Should guide toward desired behavior

Reward Function: The Learning Signal

Reward Function Purpose

$r(s, a)$ provides immediate feedback that guides the agent toward its goal

Design Principles

- Rewards communicate **what** to achieve, not **how**
- Given immediately after each action
- Should guide toward desired behavior
- Warning: Poor design can lead to unintended behaviors!

Reward Function: The Learning Signal

Reward Function Purpose

$r(s, a)$ provides immediate feedback that guides the agent toward its goal

Design Principles

- Rewards communicate **what** to achieve, not **how**
- Given immediately after each action
- Should guide toward desired behavior
- **Warning**: Poor design can lead to unintended behaviors!

Common Examples

- **Robot navigation**: $r = -1$ per step, $+100$ for goal

Reward Function: The Learning Signal

Reward Function Purpose

$r(s, a)$ provides immediate feedback that guides the agent toward its goal

Design Principles

- Rewards communicate **what** to achieve, not **how**
- Given immediately after each action
- Should guide toward desired behavior
- Warning: Poor design can lead to unintended behaviors!

Common Examples

- **Robot navigation**: $r = -1$ per step, $+100$ for goal
- **Game playing**: $r = +1$ win, -1 loss, 0 otherwise

From Immediate to Cumulative Rewards

The Challenge

- Actions have lasting consequences

From Immediate to Cumulative Rewards

The Challenge

- Actions have lasting consequences
- We care about **total** reward over time, not just immediate

From Immediate to Cumulative Rewards

The Challenge

- Actions have lasting consequences
- We care about **total** reward over time, not just immediate
- Future outcomes are uncertain due to stochastic transitions

From Immediate to Cumulative Rewards

The Challenge

- Actions have lasting consequences
- We care about **total** reward over time, not just immediate
- Future outcomes are uncertain due to stochastic transitions

$$r_0 + r_1 + r_2 + r_3 + r_4 = G_t$$

Expected Return

From Immediate to Cumulative Rewards

The Challenge

- Actions have lasting consequences
- We care about **total** reward over time, not just immediate
- Future outcomes are uncertain due to stochastic transitions

$$r_0 + r_1 + r_2 + r_3 + r_4 = G_t$$

Expected Return

Return Definition

The **return** G_t is the cumulative reward from time t:

$$G_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_T$$

The Problem with Infinite Horizons

What if $T = \infty$?

The Problem with Infinite Horizons

What if $T = \infty$?

- Sum might blow up to infinity

The Problem with Infinite Horizons

What if $T = \infty$?

- Sum might blow up to infinity
- Cannot compare different strategies

The Problem with Infinite Horizons

What if $T = \infty$?

- Sum might blow up to infinity
- Cannot compare different strategies
- Need a solution for continuing tasks

The Problem with Infinite Horizons

What if $T = \infty$?

- Sum might blow up to infinity
- Cannot compare different strategies
- Need a solution for continuing tasks

Solution: Discounted Return

Introduce **discount factor** $\gamma \in [0, 1]$:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

The Problem with Infinite Horizons

Discounted Return

Introduce **discount factor** $\gamma \in [0, 1]$:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

The Problem with Infinite Horizons

Discounted Return

Introduce **discount factor** $\gamma \in [0, 1]$:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

Discount Factor Interpretation

The Problem with Infinite Horizons

Discounted Return

Introduce **discount factor** $\gamma \in [0, 1]$:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

Discount Factor Interpretation

- $\gamma = 0$: Only immediate rewards matter (completely myopic)

The Problem with Infinite Horizons

Discounted Return

Introduce **discount factor** $\gamma \in [0, 1]$:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

Discount Factor Interpretation

- $\gamma = 0$: Only immediate rewards matter (completely myopic)
- $\gamma = 1$: All future rewards equally important (completely farsighted)

The Problem with Infinite Horizons

Discounted Return

Introduce **discount factor** $\gamma \in [0, 1]$:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

Discount Factor Interpretation

- $\gamma = 0$: Only immediate rewards matter (completely myopic)
- $\gamma = 1$: All future rewards equally important (completely farsighted)
- $\gamma = 0.9$: Future rewards matter, but less than immediate ones

Discount Factor Example

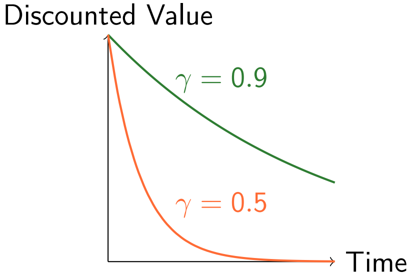
Scenario: Receive reward of 10 at each time step

With $\gamma = 0.5$:

$$\begin{aligned} G_0 &= 10 + 0.5 \times 10 + 0.5^2 \times 10 + \cdots \\ &= 10 + 5 + 2.5 + 1.25 + \cdots \\ &= \frac{10}{1 - 0.5} = 20 \end{aligned}$$

With $\gamma = 0.9$:

$$\begin{aligned} G_0 &= 10 + 9 + 8.1 + 7.29 + \cdots \\ &= \frac{10}{1 - 0.9} = 100 \end{aligned}$$

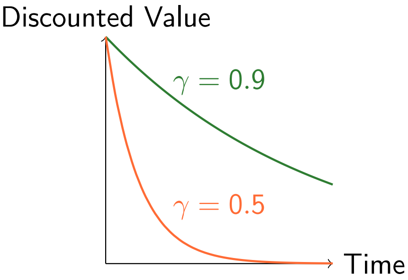


Discount Factor Example

Scenario: Receive reward of 10 at each time step

With $\gamma = 0.5$:

$$G_0 = 10 + 0.5 \times 10 + 0.5^2 \times 10 + \cdots$$
$$= 10 + 5 + 2.5 + 1.25 + \cdots$$
$$= \frac{10}{1 - 0.5} = 20$$



With $\gamma = 0.9$:

$$G_0 = 10 + 9 + 8.1 + 7.29 + \cdots$$
$$= \frac{10}{1 - 0.9} = 100$$

Higher γ values future rewards more

Policies: Decision-Making Strategies

Policy Definition

A **policy** π is a mapping from states to action probabilities:

$$\pi(a|s) = \text{Probability of selecting action } a \text{ in state } s$$

Policies: Decision-Making Strategies

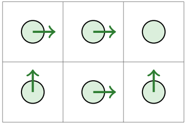
Policy Definition

A **policy** π is a mapping from states to action probabilities:

$$\pi(a|s) = \text{Probability of selecting action } a \text{ in state } s$$

Types of Policies

- **Deterministic**: $\pi(a|s) \in \{0, 1\}$
  - Always same action in same state



Policies: Decision-Making Strategies

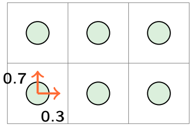
Policy Definition

A **policy** π is a mapping from states to action probabilities:

$$\pi(a|s) = \text{Probability of selecting action } a \text{ in state } s$$

Types of Policies

- **Deterministic**: $\pi(a|s) \in \{0, 1\}$
  - Always same action in same state
- **Stochastic**: $\pi(a|s) \in (0, 1)$
  - Probabilistic action selection



Policies: Decision-Making Strategies

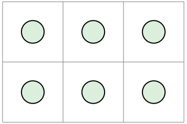
Policy Definition

A **policy** π is a mapping from states to action probabilities:

$$\pi(a|s) = \text{Probability of selecting action } a \text{ in state } s$$

Types of Policies

- **Deterministic**: $\pi(a|s) \in \{0, 1\}$
  - Always same action in same state
- **Stochastic**: $\pi(a|s) \in (0, 1)$
  - Probabilistic action selection



A policy completely specifies the agent's behavior

Value Functions: Evaluating Policies

State Value Function

The **value** of state s under policy π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| S_t = s\right]$$

"Expected return when starting in state s and following policy π"

Value Functions: Evaluating Policies

State Value Function

The **value** of state s under policy π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| S_t = s\right]$$

"Expected return when starting in state s and following policy π"

Action-Value Function (Q-Function)

The **value** of taking action a in state s under policy π:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| S_t = s, A_t = a\right]$$

"Expected return when taking action a in state s, then following policy π"

Value Functions: Evaluating Policies

State Value Function

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| S_t = s \right]$$

Action-Value Function (Q-Function)

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| S_t = s, A_t = a \right]$$

Value Functions: Evaluating Policies

State Value Function

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| S_t = s\right]$$

Action-Value Function (Q-Function)

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| S_t = s, A_t = a\right]$$

Relationship: $v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

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

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[r_t + \gamma G_{t+1} | S_t = s]
\end{aligned}
$$

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[r_t + \gamma G_{t+1} | S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']]
\end{aligned}$$

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\
&= \mathbb{E}_\pi[r_t + \gamma G_{t+1} | S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\
&= \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_\pi(s')]
\end{aligned}
$$

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_\pi(s')]$$

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_\pi(s')]$$

Interpretation

- **Immediate reward**: $r(s,a)$

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_\pi(s')]$$

Interpretation

- **Immediate reward**: $r(s,a)$
- **Future value**: $\gamma v_\pi(s')$ (discounted)

Bellman Equation: The Key Recursive Relationship

Bellman Equation Derivation

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v_\pi(s')]$$

Interpretation

- **Immediate reward**: $r(s, a)$
- **Future value**: $\gamma v_\pi(s')$ (discounted)
- **Weighted by probabilities**: Policy and transitions

Optimal Policies and Value Functions

Policy Comparison

Policy π is **better than** policy π' if:

$$v_\pi(s) \geq v_{\pi'}(s) \text{ for all states } s \in S$$

Optimal Policies and Value Functions

Policy Comparison

Policy π is **better than** policy π' if:

$$v_\pi(s) \geq v_{\pi'}(s) \text{ for all states } s \in S$$

Fundamental Theorem

There always exists at least one optimal policy that is better than or equal to all other policies.

Optimal Policies and Value Functions

Optimal Value Functions

Optimal state value function:

$$v^*(s) = \max_\pi v_\pi(s) \text{ for all } s \in S$$

Optimal action-value function:

$$q^*(s, a) = \max_\pi q_\pi(s, a) \text{ for all } s \in S, a \in A$$

Bellman Optimality Equations

Bellman Optimality Equation for v^*

$$
\begin{aligned}
v^*(s) &= \max_a q^*(s, a) \\
&= \max_a \mathbb{E}[r_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a] \\
&= \max_a \sum_{s'} p(s' | s, a)[r(s, a) + \gamma v^*(s')]
\end{aligned}
$$

Bellman Optimality Equations

Bellman Optimality Equation for v^*

$$\begin{aligned} v^*(s) &= \max_a q^*(s, a) \\ &= \max_a \mathbb{E}[r_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s'} p(s'|s, a)[r(s, a) + \gamma v^*(s')] \end{aligned}$$

Key Insight

Instead of averaging over policy probabilities, we take the **maximum** over all possible actions.

Bellman Optimality Equations

Bellman Optimality Equation for v^*

$$
\begin{aligned}
v^*(s) &= \max_a q^*(s, a) \\
&= \max_a \sum_{s'} p(s'|s, a)[r(s, a) + \gamma v^*(s')]
\end{aligned}
$$

Bellman Optimality Equations

Bellman Optimality Equation for v^*

$$\begin{aligned} v^*(s) &= \max_a q^*(s,a) \\ &= \max_a \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v^*(s')] \end{aligned}$$

Finding the Optimal Policy

Once we have v^*, the optimal policy is:

$$\pi^*(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r(s,a) + \gamma v^*(s')]$$

Choose the action that achieves the maximum in the Bellman optimality equation

MDP Solution Methods

Dynamic Programming Methods

When we know the MDP model (S, A, T, R):

- **Policy Iteration**: Alternate between policy evaluation and improvement

MDP Solution Methods

Dynamic Programming Methods

When we know the MDP model (S, A, T, R):

- **Policy Iteration**: Alternate between policy evaluation and improvement
- **Value Iteration**: Iteratively update value function using Bellman optimality equation

MDP Solution Methods

Dynamic Programming Methods

When we know the MDP model (S, A, T, R):

- **Policy Iteration**: Alternate between policy evaluation and improvement
- **Value Iteration**: Iteratively update value function using Bellman optimality equation
- **Guaranteed convergence** to optimal policy

MDP Solution Methods

Dynamic Programming Methods

When we know the MDP model (S, A, T, R):

- **Policy Iteration**: Alternate between policy evaluation and improvement
- **Value Iteration**: Iteratively update value function using Bellman optimality equation
- **Guaranteed convergence** to optimal policy

When Model is Unknown

Real-world applications often don't know transition probabilities:

- **Model-free RL**: Learn from experience without knowing T or R

MDP Solution Methods

Dynamic Programming Methods

When we know the MDP model (S, A, T, R):

- **Policy Iteration**: Alternate between policy evaluation and improvement
- **Value Iteration**: Iteratively update value function using Bellman optimality equation
- **Guaranteed convergence** to optimal policy

When Model is Unknown

Real-world applications often don't know transition probabilities:

- **Model-free RL**: Learn from experience without knowing T or R
- **Temporal Difference Learning**: Update estimates using observed transitions

MDP Solution Methods

Dynamic Programming Methods

When we know the MDP model (S, A, T, R):

- **Policy Iteration**: Alternate between policy evaluation and improvement
- **Value Iteration**: Iteratively update value function using Bellman optimality equation
- **Guaranteed convergence** to optimal policy

When Model is Unknown

Real-world applications often don't know transition probabilities:

- **Model-free RL**: Learn from experience without knowing T or R
- **Temporal Difference Learning**: Update estimates using observed transitions
- **Q-Learning, SARSA**: Popular algorithms for this setting

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties
- **Actions**: Add/remove functional groups, modify structure

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties
- **Actions**: Add/remove functional groups, modify structure
- **Reward**: Binding affinity, toxicity, synthesis complexity

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties
- **Actions**: Add/remove functional groups, modify structure
- **Reward**: Binding affinity, toxicity, synthesis complexity
- **Challenge**: Massive state spaces, sparse rewards

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties
- **Actions**: Add/remove functional groups, modify structure
- **Reward**: Binding affinity, toxicity, synthesis complexity
- **Challenge**: Massive state spaces, sparse rewards

Advanced Robotics

- **State**: Joint positions, velocities, environment perception

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties
- **Actions**: Add/remove functional groups, modify structure
- **Reward**: Binding affinity, toxicity, synthesis complexity
- **Challenge**: Massive state spaces, sparse rewards

Advanced Robotics

- **State**: Joint positions, velocities, environment perception
- **Actions**: Motor commands, grasp planning, trajectory selection

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties
- **Actions**: Add/remove functional groups, modify structure
- **Reward**: Binding affinity, toxicity, synthesis complexity
- **Challenge**: Massive state spaces, sparse rewards

Advanced Robotics

- **State**: Joint positions, velocities, environment perception
- **Actions**: Motor commands, grasp planning, trajectory selection
- **Reward**: Task completion, energy efficiency, safety

MDP Applications in Modern AI

Computational Biology & Drug Discovery

- **State**: Molecular configuration, binding sites, chemical properties
- **Actions**: Add/remove functional groups, modify structure
- **Reward**: Binding affinity, toxicity, synthesis complexity
- **Challenge**: Massive state spaces, sparse rewards

Advanced Robotics

- **State**: Joint positions, velocities, environment perception
- **Actions**: Motor commands, grasp planning, trajectory selection
- **Reward**: Task completion, energy efficiency, safety
- **Challenge**: Continuous spaces, real-time constraints

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape
- **Actions**: Learning rate adjustments, architecture modifications

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape
- **Actions**: Learning rate adjustments, architecture modifications
- **Reward**: Validation performance, computational efficiency

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape
- **Actions**: Learning rate adjustments, architecture modifications
- **Reward**: Validation performance, computational efficiency
- **Challenge**: Non-stationary environments, delayed feedback

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape
- **Actions**: Learning rate adjustments, architecture modifications
- **Reward**: Validation performance, computational efficiency
- **Challenge**: Non-stationary environments, delayed feedback

Research Strategy Optimization

- **State**: Current knowledge, available resources, progress

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape
- **Actions**: Learning rate adjustments, architecture modifications
- **Reward**: Validation performance, computational efficiency
- **Challenge**: Non-stationary environments, delayed feedback

Research Strategy Optimization

- **State**: Current knowledge, available resources, progress
- **Actions**: Experiment design, resource allocation, collaborations

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape
- **Actions**: Learning rate adjustments, architecture modifications
- **Reward**: Validation performance, computational efficiency
- **Challenge**: Non-stationary environments, delayed feedback

Research Strategy Optimization

- **State**: Current knowledge, available resources, progress
- **Actions**: Experiment design, resource allocation, collaborations
- **Reward**: Scientific impact, knowledge gain, progress metrics

Advanced MDP Applications

Large Language Model Training

- **State**: Model parameters, training data batch, loss landscape
- **Actions**: Learning rate adjustments, architecture modifications
- **Reward**: Validation performance, computational efficiency
- **Challenge**: Non-stationary environments, delayed feedback

Research Strategy Optimization

- **State**: Current knowledge, available resources, progress
- **Actions**: Experiment design, resource allocation, collaborations
- **Reward**: Scientific impact, knowledge gain, progress metrics
- **Challenge**: Extremely long horizons, uncertain outcomes

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?
2. **Action Space**: What decisions can you make? Discrete vs continuous?

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?
2. **Action Space**: What decisions can you make? Discrete vs continuous?
3. **Transition Dynamics**: How do actions change states? What's uncertain?

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?
2. **Action Space**: What decisions can you make? Discrete vs continuous?
3. **Transition Dynamics**: How do actions change states? What's uncertain?
4. **Reward Design**: How do you quantify progress toward goals?

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?
2. **Action Space**: What decisions can you make? Discrete vs continuous?
3. **Transition Dynamics**: How do actions change states? What's uncertain?
4. **Reward Design**: How do you quantify progress toward goals?
5. **Scalability**: What makes this computationally challenging?

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?
2. **Action Space**: What decisions can you make? Discrete vs continuous?
3. **Transition Dynamics**: How do actions change states? What's uncertain?
4. **Reward Design**: How do you quantify progress toward goals?
5. **Scalability**: What makes this computationally challenging?

Critical Questions

- Does your state representation satisfy the Markov property?

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?
2. **Action Space**: What decisions can you make? Discrete vs continuous?
3. **Transition Dynamics**: How do actions change states? What's uncertain?
4. **Reward Design**: How do you quantify progress toward goals?
5. **Scalability**: What makes this computationally challenging?

Critical Questions

- Does your state representation satisfy the Markov property?
- Does your reward function capture what you actually want?

Key Skills: MDP Formulation

Problem Formulation Process

1. **Define State Space**: What information is crucial for decisions?
2. **Action Space**: What decisions can you make? Discrete vs continuous?
3. **Transition Dynamics**: How do actions change states? What's uncertain?
4. **Reward Design**: How do you quantify progress toward goals?
5. **Scalability**: What makes this computationally challenging?

Critical Questions

- Does your state representation satisfy the Markov property?
- Does your reward function capture what you actually want?
- Where might the MDP framework break down?

Current Research Challenges

Computational Challenges

- **Curse of Dimensionality**: State spaces grow exponentially

Current Research Challenges

Computational Challenges

- **Curse of Dimensionality**: State spaces grow exponentially
- **Continuous Spaces**: Real problems rarely have discrete states/actions

Current Research Challenges

Computational Challenges

- **Curse of Dimensionality**: State spaces grow exponentially
- **Continuous Spaces**: Real problems rarely have discrete states/actions
- **Partial Observability**: Perfect state information often unrealistic

Current Research Challenges

Computational Challenges

- **Curse of Dimensionality**: State spaces grow exponentially
- **Continuous Spaces**: Real problems rarely have discrete states/actions
- **Partial Observability**: Perfect state information often unrealistic

Research Frontiers

- **Transfer Learning**: How do solutions generalize across problems?

Current Research Challenges

Computational Challenges

- **Curse of Dimensionality**: State spaces grow exponentially
- **Continuous Spaces**: Real problems rarely have discrete states/actions
- **Partial Observability**: Perfect state information often unrealistic

Research Frontiers

- **Transfer Learning**: How do solutions generalize across problems?
- **Multi-Agent MDPs**: What happens when multiple agents interact?

Current Research Challenges

Computational Challenges

- **Curse of Dimensionality**: State spaces grow exponentially
- **Continuous Spaces**: Real problems rarely have discrete states/actions
- **Partial Observability**: Perfect state information often unrealistic

Research Frontiers

- **Transfer Learning**: How do solutions generalize across problems?
- **Multi-Agent MDPs**: What happens when multiple agents interact?
- **Inverse RL**: Can we infer reward functions from behavior?

Current Research Challenges

Computational Challenges

- **Curse of Dimensionality**: State spaces grow exponentially
- **Continuous Spaces**: Real problems rarely have discrete states/actions
- **Partial Observability**: Perfect state information often unrealistic

Research Frontiers

- **Transfer Learning**: How do solutions generalize across problems?
- **Multi-Agent MDPs**: What happens when multiple agents interact?
- **Inverse RL**: Can we infer reward functions from behavior?
- **Safe RL**: How to ensure safety during learning?

Summary

What We've Learned

- **MDP Framework**: States, actions, transitions, rewards
- **Sequential Decision Making**: Actions have lasting consequences
- **Value Functions**: Evaluate policies and states
- **Optimal Solutions**: Bellman equations and dynamic programming

Next Steps

- Dynamic Programming algorithms (Policy/Value Iteration)
- Model-free reinforcement learning
- Temporal difference learning
- Deep reinforcement learning

From bandits to MDPs: The foundation is complete!