Reinforcement Learning

Dr Stephen James

Autumn Term 2025

Imperial College London

Reinforcement Learning
Lecture 1: Markov Decision Processes

Dr Stephen James

Autumn Term 2025

Imperial College London

Outline

1. Introduction
2. Markov Decision Processes
3. Policies and Returns
4. Value Functions
5. Bellman Equations
6. Optimal Policies and Bellman Optimality

How does reinforcement learning actually work?

Intro: Why RL is amazing and will change the world

⬇️

Lecture 1: How does RL actually work?

Prerequisites

- Linear algebra, probability theory, calculus

Learning Outcomes

- Understand what makes RL different from other ML
- Master the mathematical framework: MDPs
- Learn the core concepts: policies, value functions, Bellman equations

What mathematical tools do we need for RL?

Probability Essentials

- **Conditional probability**: $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- **Chain rule**: $P(A \cap B) = P(A|B)P(B)$

Expectation

- **Definition**: $\mathbb{E}[X] = \sum_x x \cdot P(X = x)$
- **Linearity**: $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$
- **Conditional**: $\mathbb{E}[X|Y] = \sum_x x \cdot P(X = x|Y)$
- **Law of total expectation**: $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$

How does RL fit with other types of machine learning?

Paradigm	Learning Signal	Goal	Examples
Supervised	Labeled examples (x, y) pairs	Predict labels $y = f(x)$	Image classification Language translation
Unsupervised	Unlabeled data x only	Find patterns Density, structure	Clustering Dimensionality reduction
Reinforcement	Rewards (s, a, R, s')	Maximize return Optimal actions	Game playing Robot control

What makes RL uniquely challenging?

Key RL Challenges

- **Delayed consequences**: Actions have long-term effects
- **Exploration vs exploitation**: Try new things vs use what works
- **Credit assignment**: Which actions led to rewards?
- **Non-stationary**: Your actions change the data distribution

Why These Don't Exist in Supervised Learning

- **Immediate feedback**: Each (x, y) pair gives direct feedback
- **Fixed dataset**: No exploration needed, all data available
- **Clear attribution**: Each input directly maps to output
- **Stationary**: Data distribution doesn't change during training

How do we formalize sequential decision making problems?

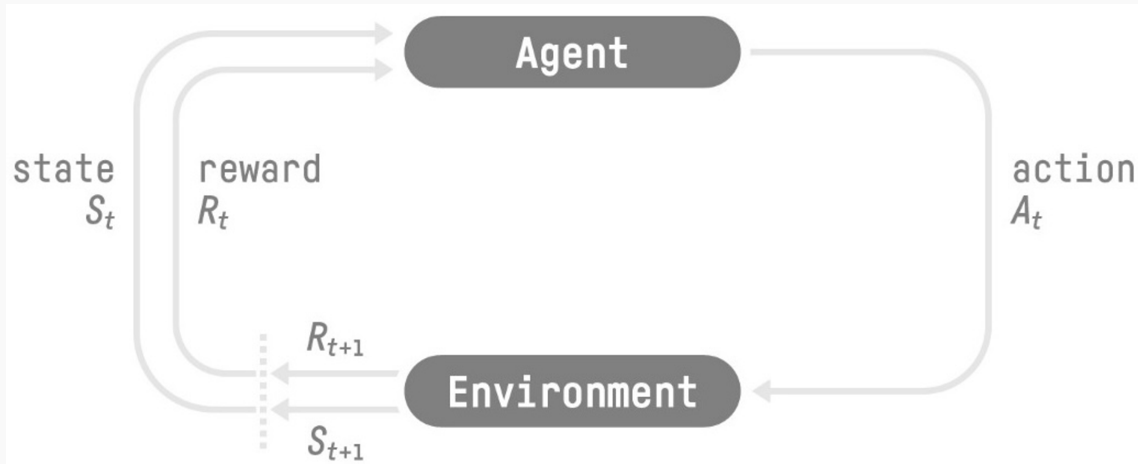
Definition (Markov Decision Process (MDP))
A model for sequential decision making when outcomes are uncertain.

Formally defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$:

- $\mathcal{S}$: Set of **states** - all possible situations
- $\mathcal{A}$: Set of **actions** - all possible decisions
- $P(s'|s, a)$: **Transition probabilities** - how actions change states
- $R(s, a, s')$: **Reward function** - immediate feedback for transitions

If you can specify $(\mathcal{S}, \mathcal{A}, P, R)$, you can apply RL!

What does the agent-environment interaction look like?



The RL Loop:

1. Agent observes **state** s_t
2. Agent takes **action** a_t
3. Environment gives **reward** r_{t+1}
4. Environment transitions to new **state** s_{t+1}
5. Repeat...

Goal: Learn to maximize **cumulative reward**

Do tasks have natural start and end points?

Definition (Episode)
A complete sequence of interaction from start to terminal state:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots, S_T$$

where S_T is a terminal state.

Types of MDPs

- **Episodic**: Has clear start/end states (games, tasks)
  - Chess game, robot reaching goal, Atari game
- **Continuing**: No terminal states (ongoing processes)
  - Stock trading, server management, autonomous driving

Why this matters: Affects how we define returns and value functions

What do states, actions, and rewards look like in practice?

Atari Breakout

- States: ?
- Actions: ?
- Rewards: ?

Robot Manipulation

- States: ?
- Actions: ?
- Rewards: ?

Autonomous Driving

- States: ?
- Actions: ?
- Rewards: ?

Recommendation Systems

- States: ?
- Actions: ?
- Rewards: ?

What do states, actions, and rewards look like in practice?

Atari Breakout

- **States**: Pixel observations
- **Actions**: Left, Right, Fire
- **Rewards**: +1 for brick hit

Robot Manipulation

- **States**: ?
- **Actions**: ?
- **Rewards**: ?

Autonomous Driving

- **States**: ?
- **Actions**: ?
- **Rewards**: ?

Recommendation Systems

- **States**: ?
- **Actions**: ?
- **Rewards**: ?

What do states, actions, and rewards look like in practice?

Atari Breakout

- **States**: Pixel observations
- **Actions**: Left, Right, Fire
- **Rewards**: +1 for brick hit

Robot Manipulation

- **States**: ?
- **Actions**: ?
- **Rewards**: ?

Autonomous Driving

- **States**: Camera, lidar, GPS
- **Actions**: Steering, throttle, brake
- **Rewards**: Safe, efficient driving

Recommendation Systems

- **States**: ?
- **Actions**: ?
- **Rewards**: ?

What do states, actions, and rewards look like in practice?

Atari Breakout

- **States**: Pixel observations
- **Actions**: Left, Right, Fire
- **Rewards**: +1 for brick hit

Robot Manipulation

- **States**: Joint angles, object poses
- **Actions**: Joint torques/velocities
- **Rewards**: Task completion

Autonomous Driving

- **States**: Camera, lidar, GPS
- **Actions**: Steering, throttle, brake
- **Rewards**: Safe, efficient driving

Recommendation Systems

- **States**: ?
- **Actions**: ?
- **Rewards**: ?

What do states, actions, and rewards look like in practice?

Atari Breakout

- **States**: Pixel observations
- **Actions**: Left, Right, Fire
- **Rewards**: +1 for brick hit

Robot Manipulation

- **States**: Joint angles, object poses
- **Actions**: Joint torques/velocities
- **Rewards**: Task completion

Autonomous Driving

- **States**: Camera, lidar, GPS
- **Actions**: Steering, throttle, brake
- **Rewards**: Safe, efficient driving

Recommendation Systems

- **States**: User history, context
- **Actions**: Recommend items
- **Rewards**: Clicks, purchases, ratings

Why start with finite state and action spaces?

Definition (Finite MDP)
An MDP where both state and action sets are finite:

- $|\mathcal{S}| = n < \infty$ (finite number of states)
- $|\mathcal{A}| = m < \infty$ (finite number of actions)

Why Start With Finite MDPs?

- **Mathematical tractability**: Can represent everything as matrices/tables
- **Exact solutions**: Can find optimal policies exactly
- **Clear intuition**: Easy to visualize and understand
- **Foundation for continuous**: Continuous methods often discretize or generalize these ideas

How do we model uncertainty in state transitions?

Definition (Dynamics/Transition Function)

$$P(s'|s,a) = \Pr[S_{t+1} = s'|S_t = s, A_t = a]$$

The probability of transitioning to state s' given current state s and action a.

Key Properties

- **Probability distribution**: $\sum_{s' \in \mathcal{S}} P(s'|s,a) = 1$ for all s, a
- **Stationary**: Transition probabilities don't change over time
- **Markovian**: Next state depends only on current state and action

Deterministic Example:

- GridWorld: $P(s'|s, \text{Right}) = 1$ if s' is right of s

Stochastic Example:

- Sticky GridWorld: $P(s'|s, \text{Right}) = 0.9$ if s' is right, $P(s|s, \text{Right}) = 0.1$

Why is the Markov property so important?

Definition (Markov Property)
The future is independent of the past given the present:

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = P(s_{t+1}|s_t, a_t)$$

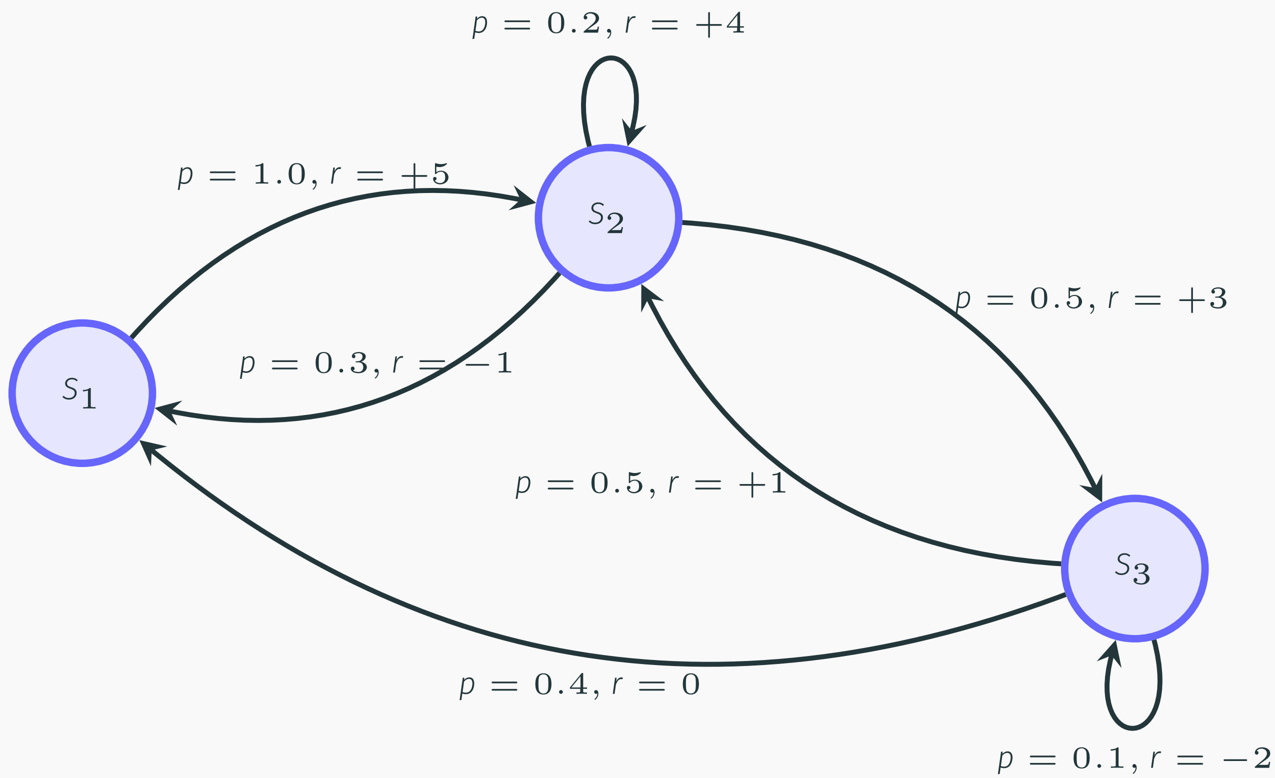
What This Means

- The current state contains all information needed to predict the future
- We don't need to remember the entire history
- Makes the problem mathematically tractable

Markovian vs Non-Markovian Examples

- **Markovian**: Chess position, robot joint angles + velocities
- **Non-Markovian**: Single Atari frame (no velocity), stock prices, poker (hidden cards)

How can we visualize MDP structure?



State Transition Diagram:

- Nodes = states, Arrows = transitions
- Labels show probability (p) and reward (r)
- Self-loops = staying in same state

How do we represent MDPs mathematically?

Transition Matrix P

$P_{ij} = P(S_{t+1} = j | S_t = i)$

$\mathbf{P} =$

	S_1	S_2	S_3
S_1	0.0	1.0	0.0
S_2	0.3	0.2	0.5
S_3	0.4	0.5	0.1

Each row sums to 1.0

Reward Matrix R

$R_{ij} = E[R_{t+1} | S_t = i, S_{t+1} = j]$

$\mathbf{R} =$

	S_1	S_2	S_3
S_1	0	+5	0
S_2	−1	+4	+3
S_3	0	+1	−2

Expected reward per transition

This can be ofc. made only for discrete and finite space

How do agents choose actions?

Definition (Policy)
A policy π is a mapping from states to actions:

- **Deterministic**: $a = \pi(s)$
- **Stochastic**: $a \sim \pi(a|s)$

Example Policies:

- **Random**: Choose actions uniformly
- **Greedy**: Always go toward goal
- **Safe**: Avoid traps at all costs
- **Optimal**: Maximize cumulative reward

The goal of RL is to find the optimal policy π^*!

How do we measure total reward over time?

The Problem
How do we measure ”total reward” when actions have long-term consequences?

Simple Example: Saving Money

- Today: Save \$100 (reward = -\$100)
- Tomorrow: Earn \$5 interest (reward = +\$5)
- Next day: Earn \$5 more interest (reward = +\$5)
- ...

Question: Was saving worth it? We need to sum ALL future rewards.

Why do we discount future rewards?

Definition (Discounted Return)
The return G_t is the cumulative discounted reward from time t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

↳ what I expect to get in future, with respect with my "forgetting"

Why Discounting? ($\gamma < 1$)

- **Mathematical convenience**: Ensures finite returns
- **Uncertainty**: Future rewards are less certain
- **Preference**: We prefer immediate rewards
- **Computational**: Avoids infinite planning horizons

Examples: $\gamma = 0$ (myopic), $\gamma = 0.9$ (balanced), $\gamma = 1$ (far-sighted)

How do we predict future success?

Definition (State Value Function)
The value of state s under policy π is the expected return:

$$V^{\pi}(s) = \mathbb{E}[G_t | s_t = s, \pi]$$

Definition (Action Value Function (Q-function))
The value of taking action a in state s under policy π:

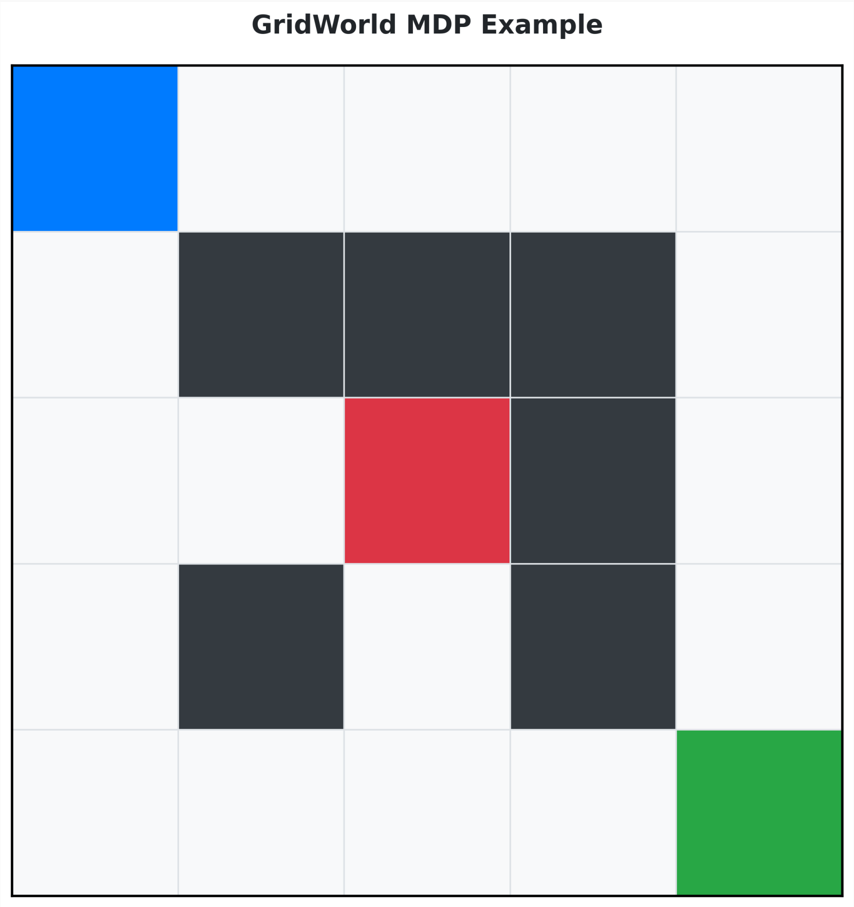
$$Q^{\pi}(s, a) = \mathbb{E}[G_t | s_t = s, a_t = a, \pi]$$

Intuition

- $V^{\pi}(s)$: "How good is it to be in state s?"
- $Q^{\pi}(s, a)$: "How good is it to take action a in state s?"
- Both measure expected future reward, not immediate reward!

What is the equivalent of Q^{π} in continuous space (like car driving)

What do value functions look like visually?

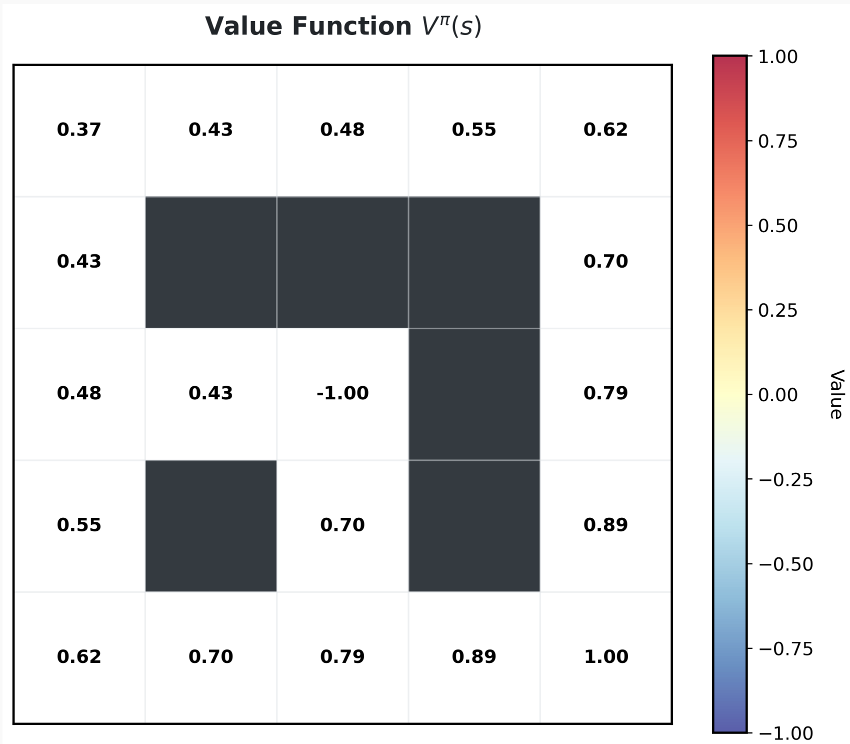


Simple 5×5 GridWorld

MDP Components:

- $\mathcal{S}$: Grid positions (i, j)
- $\mathcal{A}$: {Up, Down, Left, Right}
- P: Deterministic movement
- R: Goal=+1, Trap=-1, Step=-0.01
- $\gamma = 0.9$

How do values spread through the state space?



State values $V^\pi(s)$ for a specific policy

What We See:

- **Goal state**: High value (close to +1)
- **Trap state**: Low value (close to -1)
- **Gradient**: Values decrease with distance from goal

Key Insight: Value functions tell us which states are "good" to be in!

🎮 Mentimeter Poll

menti.com - Code: XXXX XXXX

In the GridWorld example, if $\gamma = 0$ (no discounting), what happens?

- A) The agent becomes more patient
- B) Only immediate rewards matter
- C) The Bellman equation becomes invalid
- D) All states have the same value
- E) The policy becomes deterministic

Bellman Equations

What is the fundamental relationship in value functions?

Value functions satisfy a **recursive relationship** - we can define the value of a state in terms of the values of its successor states.

Theorem (Bellman Equation)

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

What is the fundamental relationship in value functions?

Value functions satisfy a **recursive relationship** - we can define the value of a state in terms of the values of its successor states.

Theorem (Bellman Equation)

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

What is the fundamental relationship in value functions?

Value functions satisfy a **recursive relationship** - we can define the value of a state in terms of the values of its successor states.

Theorem (Bellman Equation)

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

What is the fundamental relationship in value functions?

Value functions satisfy a **recursive relationship** - we can define the value of a state in terms of the values of its successor states.

Theorem (Bellman Equation)

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

What is the fundamental relationship in value functions?

Value functions satisfy a **recursive relationship** - we can define the value of a state in terms of the values of its successor states.

Theorem (Bellman Equation)

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}[G_t | S_t = s, \pi] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, \pi] \\
&= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}[G_{t+1} | S_{t+1} = s', \pi] | S_t, \pi] \\
&= \mathbb{E}[R_{t+1} + \gamma V^{\pi}(S_{t+1}) | S_t = s, \pi] \\
&= \sum_{a} \pi(a|s) \sum_{s', r} P(s', r | s, a)[r + \gamma V^{\pi}(s')]
\end{aligned}
$$

What is the fundamental relationship in value functions?

Value functions satisfy a **recursive relationship** - we can define the value of a state in terms of the values of its successor states.

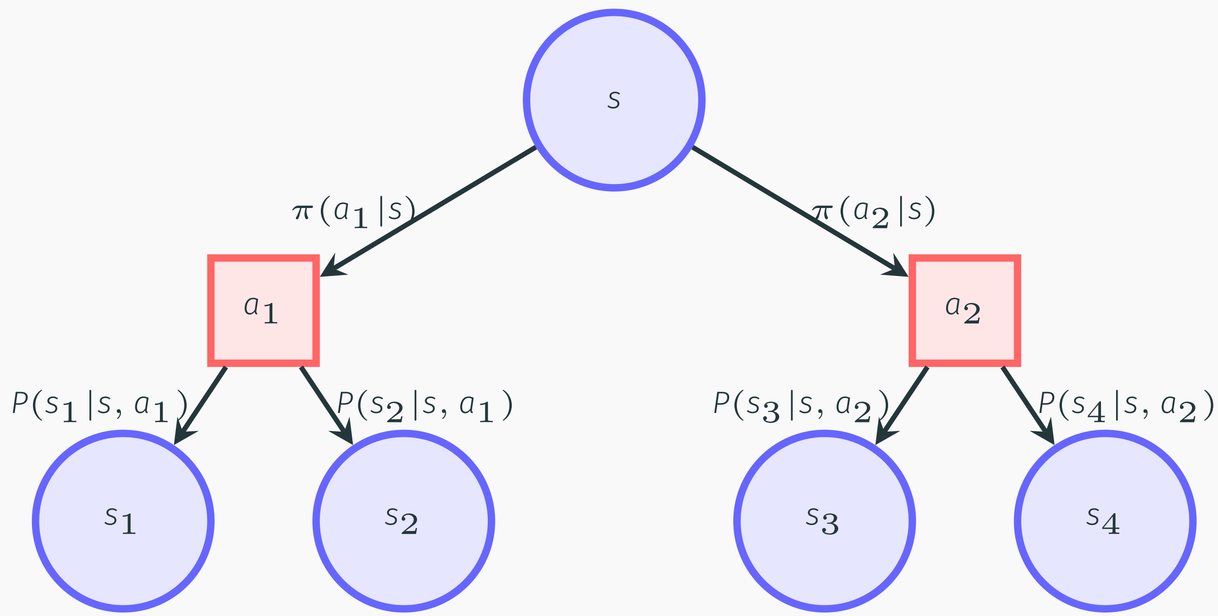
Theorem (Bellman Equation)

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

This equation is the foundation of ALL reinforcement learning algorithms!

How can we visualize the Bellman equation?

Backup Diagram for $V^\pi(s)$:



Key: States (circles), Actions (squares), Values flow backwards through tree

menti.com - Code: XXXX XXXX

The Bellman equation expresses:

- A) How to compute immediate rewards
- B) The relationship between current and future values
- C) How to choose optimal actions
- D) The transition probabilities
- E) The discount factor

menti.com - Code: XXXX XXXX

Which of these is **NOT** a correct step in deriving the Bellman equation?

A) $V^{\pi}(s) = \mathbb{E}[G_t | s_t = s, \pi]$

B) $V^{\pi}(s) = \mathbb{E}[R_{t+1} + \gamma G_{t+1} | s_t = s, \pi]$

C) $V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[r + \gamma V^{\pi}(s')]$

D) $V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} P(s', r|s,a)\gamma[r + V^{\pi}(s')]$

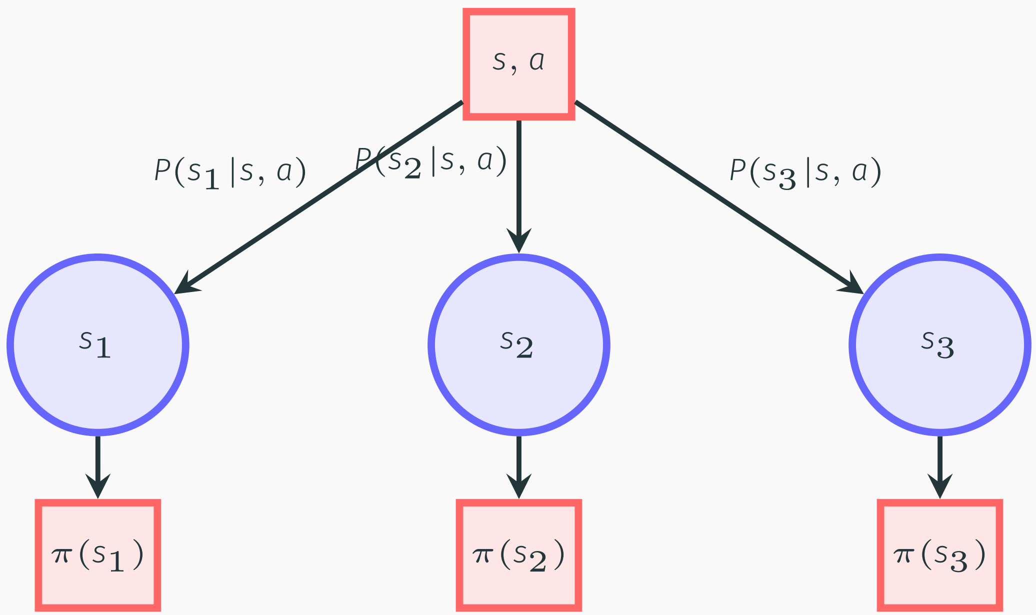
E) $V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} P(s', r|s,a)[r + \gamma V^{\pi}(s')]$

What about Bellman equations for action values?

Theorem (Bellman Equation for Action Values)
The action value function $Q^\pi(s,a)$ satisfies:

$$Q^\pi(s,a) = \sum_{s',r} P(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s')Q^\pi(s',a')]$$

Backup Diagram for $Q^\pi(s,a)$:



What defines the best possible policy?

Definition (Optimal Value Functions)

↗ we get this by exploring the environment

$$V^*(s) = \max_\pi V^\pi(s) \quad (1)$$

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \quad (2)$$

Relationship Between Optimal Functions

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma V^*(s')|s_t = s, a_t = a]$$

↗ expectation of
- what I get
- what comes in future

$$V^*(s) = \max_a Q^*(s,a)$$

Definition (Optimal Policy)

An optimal policy π^* satisfies:

$$\pi^*(s) = \text{argmax}_a Q^*(s,a)$$

How do we find the optimal value function?

We want the optimal policy π^* that maximizes value, not just evaluate a given policy.

Key insight: If we act optimally, we choose the action with highest Q-value:

$$\pi^*(s) = \text{argmax}_a Q^*(s, a)$$

this evaluates

This means: $V^*(s) = \max_a Q^*(s, a)$

this performs the action! :-)

Can we write a recursive equation for $V^*(s)$ like we did for $V^\pi(s)$?

What are the Bellman optimality equations?

The Bellman Optimality Equation expresses the optimal value function recursively:

Theorem (Bellman Optimality Equation for V^*)

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s,a) && \text{(choose best action)} \\
&= \max_a \mathbb{E}[G_t | S_t = s, A_t = a, \pi^*] && \text{(expected return under optimal policy)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a, \pi^*] && \text{(break into immediate + future)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] && \text{(future return is optimal)} \\
&= \max_a \sum_{s',r} P(s', r | s, a)[r + \gamma V^*(s')] && \text{(expand expectation)}
\end{aligned}
$$

What are the Bellman optimality equations?

The Bellman Optimality Equation expresses the optimal value function recursively:

Theorem (Bellman Optimality Equation for V^*)

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s,a) && \text{(choose best action)} \\
&= \max_a \mathbb{E}[G_t | S_t = s, A_t = a, \pi^*] && \text{(expected return under optimal policy)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a, \pi^*] && \text{(break into immediate + future)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] && \text{(future return is optimal)} \\
&= \max_a \sum_{s',r} P(s', r | s, a)[r + \gamma V^*(s')] && \text{(expand expectation)}
\end{aligned}
$$

What are the Bellman optimality equations?

The Bellman Optimality Equation expresses the optimal value function recursively:

Theorem (Bellman Optimality Equation for V^*)

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s, a) && \text{(choose best action)} \\
&= \max_a \mathbb{E}[G_t | S_t = s, A_t = a, \pi^*] && \text{(expected return under optimal policy)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a, \pi^*] && \text{(break into immediate + future)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] && \text{(future return is optimal)} \\
&= \max_a \sum_{s', r} P(s', r | s, a)[r + \gamma V^*(s')] && \text{(expand expectation)}
\end{aligned}
$$

What are the Bellman optimality equations?

The Bellman Optimality Equation expresses the optimal value function recursively:

Theorem (Bellman Optimality Equation for V^*)

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s,a) && \text{(choose best action)} \\
&= \max_a \mathbb{E}[G_t | S_t = s, A_t = a, \pi^*] && \text{(expected return under optimal policy)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a, \pi^*] && \text{(break into immediate + future)} \\
&= \max_a \mathbb{E}[R_{t+1} + \underline{\gamma V^*(S_{t+1})} | S_t = s, A_t = a] && \text{(future return is optimal)} \\
&= \max_a \sum_{s',r} P(s', r | s, a)[r + \gamma V^*(s')] && \text{(expand expectation)}
\end{aligned}
$$

recursion

What are the Bellman optimality equations?

The Bellman Optimality Equation expresses the optimal value function recursively:

Theorem (Bellman Optimality Equation for V^*)

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s, a) && \text{(choose best action)} \\
&= \max_a \mathbb{E}[G_t | S_t = s, A_t = a, \pi^*] && \text{(expected return under optimal policy)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a, \pi^*] && \text{(break into immediate + future)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] && \text{(future return is optimal)} \\
&= \max_a \sum_{s', r} P(s', r | s, a)[r + \gamma V^*(s')] && \text{(expand expectation)}
\end{aligned}
$$

What are the Bellman optimality equations?

The Bellman Optimality Equation expresses the optimal value function recursively:

Theorem (Bellman Optimality Equation for V^*)

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s,a) && \textbf{(choose best action)} \\
&= \max_a \mathbb{E}[G_t | S_t = s, A_t = a, \pi^*] && \textbf{(expected return under optimal policy)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a, \pi^*] && \textbf{(break into immediate + future)} \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma V^*(S_{t+1}) | S_t = s, A_t = a] && \textbf{(future return is optimal)} \\
&= \max_a \sum_{s',r} P(s', r | s, a)[r + \gamma V^*(s')] && \textbf{(expand expectation)}
\end{aligned}
$$

Theorem (Bellman Optimality Equation for Q^*)

$$
\begin{aligned}
Q^*(s,a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(s', a') | s_t = s, a_t = a] \\
&= \sum_{s',r} P(s', r | s, a)[r + \gamma \max_{a'} Q^*(s', a')]
\end{aligned}
$$

The **max operator** makes these equations nonlinear!

menti.com - Code: XXXX XXXX

If you got lost today, at what point did you lose track?

- A) Markov property and MDPs
- B) Bellman equations for state values
- C) Bellman equations for action values
- D) Backup diagrams
- E) Optimal value functions and policies
- F) Bellman optimality equations
- G) All concepts are clear

How does this connect to modern deep reinforcement learning?

The Bridge from Classical to Modern

- Same math, different representation
- Tables → Neural networks ⇝ the tables are only predicted by a network
- Exact solutions → Approximate solutions ⇝ because of predictions...
- Small problems → Complex problems ⇝ :)

Tabular RL:

- $V(s)$ stored in table
- $Q(s, a)$ stored in table
- Exact Bellman updates
- Works for small $|\mathcal{S}|, |\mathcal{A}|$

Deep RL:

- $V(s) \approx V_\theta(s)$ neural net
- $Q(s, a) \approx Q_\theta(s, a)$ neural net
- Approximate Bellman updates
- Scales to huge state spaces

What are the key concepts we've learned today?

Key Takeaways

- **MDPs**: Formalize RL problems with states, actions, rewards
- **Policies**: Define agent's behavior, can be deterministic or stochastic
- **Returns**: Measure long-term success using discounted rewards
- **Value Functions**: Predict future success from states or state-action pairs
- **Bellman Equations**: Capture recursive relationships in value functions
- **Optimal Policies**: Achieve maximum expected return, defined by optimal value functions

Next Week: Dynamic Programming - How to compute optimal policies and value functions using Bellman equations

Reading and Resources

📚 Essential Reading

- **Sutton & Barto**: Chapters 1 to 3 (MDPs and Bellman Equations)
- Focus especially on: Section 3.3 (Returns), 3.5 (Policies and Value Functions), 3.6 (Optimal Policies)