Markov Decision Processes

Bellman Equations for State and Action Values

Sarwan Ali

Department of Computer Science
Georgia State University

Understanding Markov Decision Processes

Today's Learning Journey

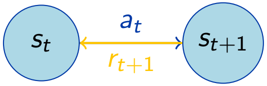
What is a Markov Decision Process?

Definition

A **Markov Decision Process (MDP)** is a mathematical framework for modeling decision-making problems where outcomes are partly random and partly under the control of a decision maker.

Key Components:

- States (S): Environment configurations
- Actions (A): Available choices
- Transitions (P): State change probabilities
- Rewards (R): Immediate feedback
- Discount Factor (γ): Future value weighting



MDP Formal Definition

An MDP is formally defined as a 5-tuple: $\langle S, A, P, R, \gamma \rangle$

$$
\begin{align}
S &: \text{ Set of states} \tag{1}\\
A &: \text{ Set of actions} \tag{2}\\
P &: S \times A \times S \to [0,1] \text{ (Transition probabilities)} \tag{3}\\
R &: S \times A \times S \to \mathbb{R} \text{ (Reward function)} \tag{4}\\
\gamma &: \text{ Discount factor, } \gamma \in [0,1] \tag{5}
\end{align}
$$

Markov Property

$$P(S_{t+1} = s' | S_t = s, A_t = a, S_{t-1}, A_{t-1}, \ldots) = P(S_{t+1} = s' | S_t = s, A_t = a)$$

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

Policy Definition

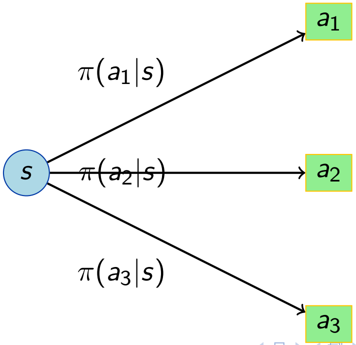
Policy

A **policy** π is a mapping from states to actions (or action probabilities):

$$\pi : S \to A \quad \text{or} \quad \pi : S \times A \to [0, 1]$$

Types of Policies:

- Deterministic: $\pi(s) = a$
- Stochastic: $\pi(a|s) = P(A_t = a|S_t = s)$
- Stationary: Time-independent
- Non-stationary: Time-dependent



Value Functions: State Value

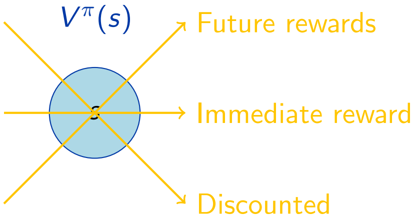
State Value Function

The **state value function** $V^\pi(s)$ represents the expected cumulative reward starting from state s and following policy π:

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s\right]$$

Interpretation:

- Expected return from state s
- Depends on the policy π
- Discounted future rewards
- γ controls importance of future vs. immediate rewards



Value Functions: Action Value

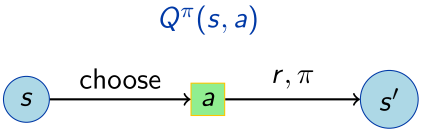
Action Value Function (Q-function)

The **action value function** $Q^\pi(s,a)$ represents the expected cumulative reward starting from state s, taking action a, then following policy π:

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s, A_0 = a\right]$$

Relationship with State Value:

$$V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s,a)$$



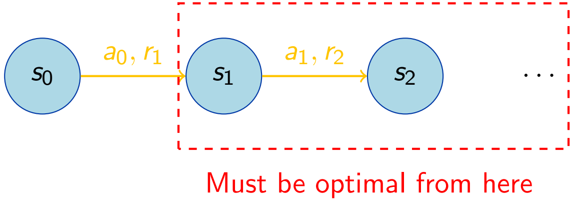
Alternative form:

$$Q^\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s, A_t = a]$$

The Bellman Principle

Principle of Optimality (Richard Bellman, 1957)

"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."



This principle leads to recursive relationships called **Bellman equations**.

Bellman Equation for State Values

Bellman Equation for $V^\pi(s)$

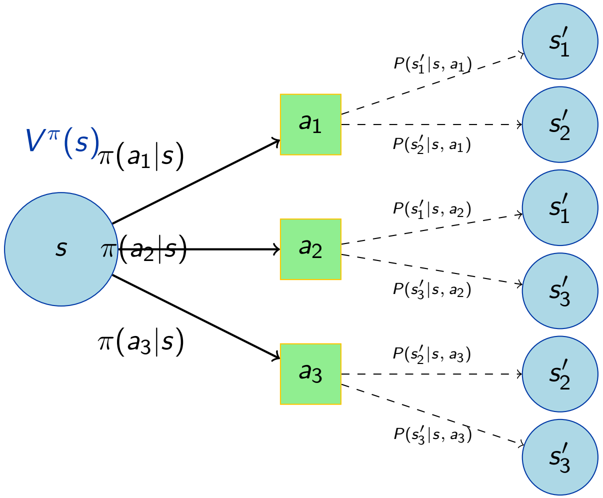
$$V^\pi(s) = \sum_{a\in A} \pi(a|s) \sum_{s'\in S} P(s'|s,a) \left[R(s,a,s') + \gamma V^\pi(s')\right]$$

Intuitive Breakdown:

$$\begin{aligned}
V^\pi(s) &= \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s] && (6)\\
&= \sum_a \pi(a|s)\mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1})|S_t = s, A_t = a] && (7)\\
&= \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^\pi(s')] && (8)
\end{aligned}$$

Value of current state = Expected immediate reward + Discounted future value

Bellman Equation: Visual Representation



Each path contributes: $\pi(a|s) \times P(s'|s,a) \times [R(s,a,s') + \gamma V^\pi(s')]$

Bellman Equation for Action Values

Bellman Equation for $Q^\pi(s, a)$

$$Q^\pi(s, a) = \sum_{s' \in S} P(s'|s, a) \left[R(s, a, s') + \gamma \sum_{a' \in A} \pi(a'|s') Q^\pi(s', a') \right]$$

Alternative compact form:

$$Q^\pi(s, a) = \sum_{s' \in S} P(s'|s, a) \left[R(s, a, s') + \gamma V^\pi(s') \right]$$

Key Relationships:

$$V^\pi(s) = \sum_{a \in A} \pi(a|s) Q^\pi(s, a) \tag{9}$$

$$Q^\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma V^\pi(S_{t+1}) | S_t = s, A_t = a] \tag{10}$$

Optimal Value Functions

Optimal State Value Function

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s \in S$$

Optimal Action Value Function

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) \quad \forall s \in S, a \in A$$

Relationship:

$$V^*(s) = \max_{a \in A} Q^*(s, a)$$

Optimal Policy

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s, a)$$

An optimal policy π^* satisfies $V^{\pi^*}(s) = V^*(s)$ for all $s \in S$.

Bellman Optimality Equations

Bellman Optimality Equation for V^*

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} P(s'|s,a) \left[R(s,a,s') + \gamma V^*(s') \right]$$

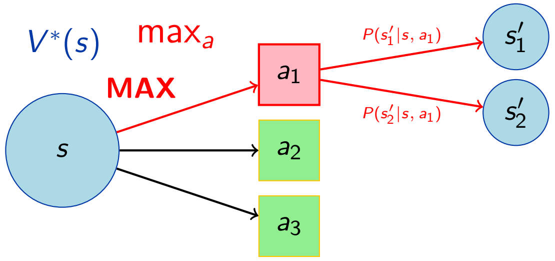
Bellman Optimality Equation for Q^*

$$Q^*(s,a) = \sum_{s' \in S} P(s'|s,a) \left[R(s,a,s') + \gamma \max_{a' \in A} Q^*(s',a') \right]$$

Key Differences from Policy Evaluation:

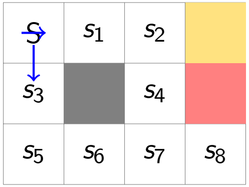
- Expectation over policy $\rightarrow$ Maximization over actions
- Non-linear system of equations
- Unique solution under standard conditions

Bellman Optimality: Visual Representation



The optimal value is achieved by selecting the **best action** at each state.

Example: Grid World MDP



MDP Components:

- **States**: Grid positions
- **Actions**: $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$
- **Rewards**:
  - Goal: $+1$
  - Pit: -1
  - Step: -0.04
- **Transitions**:
  - 80% intended
  - 10% each side
- $\gamma = 0.9$

Bellman Equation Example

For state s_1:

$$V^*(s_1) = \max_a \sum_{s'} P(s'|s_1, a)[R(s_1, a, s') + 0.9 \cdot V^*(s')]$$

Grid World: Value Iteration Results

Optimal Values $V^*(s)$:

0.81	0.87	0.92	+1.0
0.76	▨	0.66	-1.0
0.70	0.66	0.61	0.39

Optimal Policy $\pi^*(s)$:

$\rightarrow$	$\rightarrow$	$\rightarrow$	GOAL
$\uparrow$	▨	$\uparrow$	PIT
$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$

The optimal policy shows the best action to take from each state to maximize expected return.

Solving Methods for Bellman Equations

Policy Evaluation:

- Input: Policy π
- Output: $V^\pi(s)$
- Method: Iterative solution

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]$$

Value Iteration:

- Output: $V^*(s)$ and π^*
- Method: Dynamic programming

$$V_{k+1}(s) = \max_a \textstyle\sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]$$

Policy Iteration:

1. Policy Evaluation: Compute V^π
2. Policy Improvement: $\pi'(s) = \arg\max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^\pi(s')]$
3. Repeat until convergence

$$\begin{tikzcd}\boxed{\text{Evaluate}} \arrow[r, leftrightarrow] & \boxed{\text{Improve}} \\ V^\pi & \pi'\end{tikzcd}$$

Properties of Bellman Equations

Existence and Uniqueness

Under standard conditions (finite MDP, $\gamma < 1$):

- The Bellman equations have a unique solution
- There exists at least one optimal policy
- All optimal policies share the same value functions V^* and Q^*

Contraction Property:

- Bellman operator is a contraction
- Convergence rate: γ^k
- Guaranteed convergence for $\gamma < 1$

$$\|TV - TU\|_\infty \leq \gamma \|V - U\|_\infty$$

Computational Complexity:

- Value Iteration: $O(|S|^2|A|)$ per iteration
- Policy Iteration: $O(|S|^3 + |S|^2|A|)$
- Policy Evaluation: $O(|S|^3)$

Challenge: Curse of dimensionality for large state spaces

Convergence Analysis

Value Iteration Convergence

For value iteration with discount factor $\gamma < 1$:

$$\|V_k - V^*\|_\infty \leq \gamma^k \|V_0 - V^*\|_\infty$$

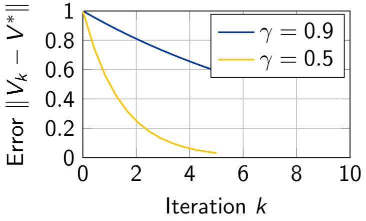
Stopping Criteria:

$$\|V_{k+1} - V_k\|_\infty < \varepsilon \frac{1-\gamma}{2\gamma}$$

This guarantees: $\|V_k - V^*\|_\infty < \varepsilon$

Policy Iteration:

- Finite convergence
- Each iteration improves policy
- Monotonic improvement



Exponential convergence rate

Extensions of Basic MDPs

Partially Observable MDPs (POMDPs):

- Agent doesn't observe full state
- Observation model needed
- Belief state representation
- Much more complex to solve

Continuous MDPs:

- Continuous state/action spaces
- Function approximation needed
- Bellman equations become functional equations

Multi-Agent MDPs:

- Multiple decision makers
- Game-theoretic considerations
- Nash equilibria

Hierarchical MDPs:

- Temporal abstraction
- Options and semi-MDPs
- Multi-level planning

Factored MDPs:

- Structured state representation
- Exploit independence
- Scalability improvements

Modern Applications

Reinforcement Learning:

- Q-Learning approximates Q^*
- Policy gradient methods
- Actor-critic algorithms
- Deep reinforcement learning

Operations Research:

- Inventory management
- Resource allocation
- Maintenance scheduling
- Financial portfolio optimization

Robotics:

- Path planning
- Motion control
- Task scheduling
- Human-robot interaction

Game AI:

- Strategic decision making
- Resource management games
- Real-time strategy
- Board game AI

Key Insight: MDPs provide the theoretical foundation for sequential decision making

Key Takeaways

Markov Decision Processes

- **Framework**: Mathematical model for sequential decision making under uncertainty
- **Components**: States, actions, transitions, rewards, discount factor
- **Goal**: Find optimal policy to maximize expected cumulative reward

Bellman Equations

- **Principle**: Optimal decisions have optimal sub-decisions
- **Policy Evaluation**: $V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[R + \gamma V^\pi(s')]$
- **Optimality**: $V^*(s) = \max_a \sum_{s'} P(s'|s,a)[R + \gamma V^*(s')]$

Solution Methods

- **Value Iteration**: Direct optimization of value function
- **Policy Iteration**: Alternating evaluation and improvement
- **Convergence**: Guaranteed under standard conditions

Next Steps and Further Reading

Recommended Next Topics:

- Reinforcement Learning: Model-free approaches (Q-learning, SARSA)
- Function Approximation: Handling large state spaces
- Policy Gradient Methods: Direct policy optimization
- Deep RL: Neural network approximation

Questions?

Thank you for your attention!

sali85@student.gsu.edu