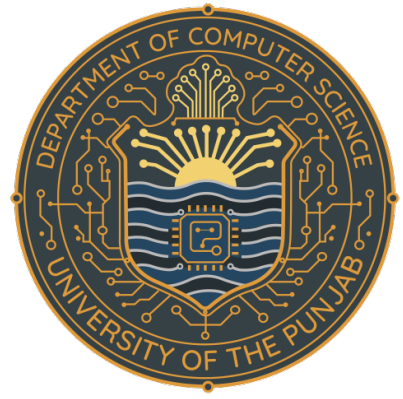
CS-866 Deep Reinforcement Learning

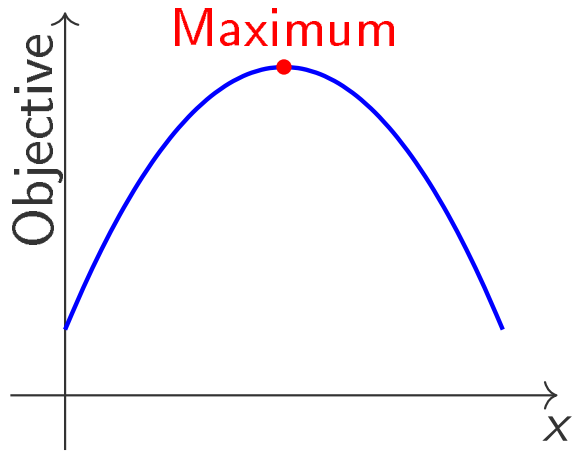
MDP Objective Function



Nazar Khan
Department of Computer Science
University of the Punjab

What Does "Objective" Mean?

- In everyday language, *objective* refers to a goal or target we want to achieve.
- In mathematics/optimization:
  - The *objective function* is the quantity we want to maximize or minimize.
  - Example: In regression, objective is to minimize mean squared error.
  - In RL, objective is to maximize expected cumulative reward.
- In this lecture: *objective* is "what the agent is trying to optimize".



MDP Objective

- Goal of Reinforcement Learning (RL): find the **optimal policy function**.
- Many algorithms exist under different assumptions.
- Before defining the RL objective, we need to understand:
  - Traces
  - Return
  - Value functions

Trace τ (Trajectory)

- As we start interacting with the MDP, at each timestep t:
  - Observe state s_t
  - Take an action a_t
  - Observe next state $s_{t+1} \sim T_{a_t}(s_t)$
  - Receive reward $r_t = R_{a_t}(s_t, s_{t+1})$
- Repeating this process leads to a sequence (trace/trajectory/episode).

$$\tau_t^n = \{s_t, a_t, r_t, s_{t+1}, \ldots, a_{t+n}, r_{t+n}, s_{t+n+1}\}$$

Finite vs Infinite Trace

- $n =$ length of the trace.
- Often assume $n = \infty$, i.e., run until termination.
- In that case, write:

$$\tau_t = \tau_t^{\infty}$$

- Traces are fundamental in RL:
  - A single full rollout of decisions
  - Also called trajectory, episode, or sequence

Trace Visualization

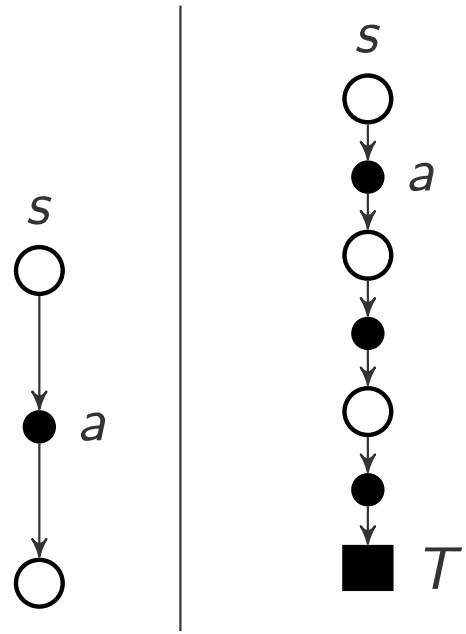


Figure: Single Transition Step vs. Full 3-Step Trace/Episode/Trajectory

Example of a Trace

Example: A short trace with three actions:

$$\begin{aligned}\tau_0^2 = \{&s_0 = 1, a_0 = \text{up}, r_0 = -1,\\ &s_1 = 2, a_1 = \text{up}, r_1 = -1,\\ &s_2 = 3, a_2 = \text{left}, r_2 = 20,\\ &s_3 = 5\}\end{aligned}$$

Trace τ: Step-by-Step Expansion

Recall: A trace (trajectory/episode) unfolds step by step:

- At each timestep t, observe s_t,
- Take action a_t,
- Observe reward r_t and next state s_{t+1}.

Trace Example:

$$\tau_0^2 = \{s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, s_3\}$$

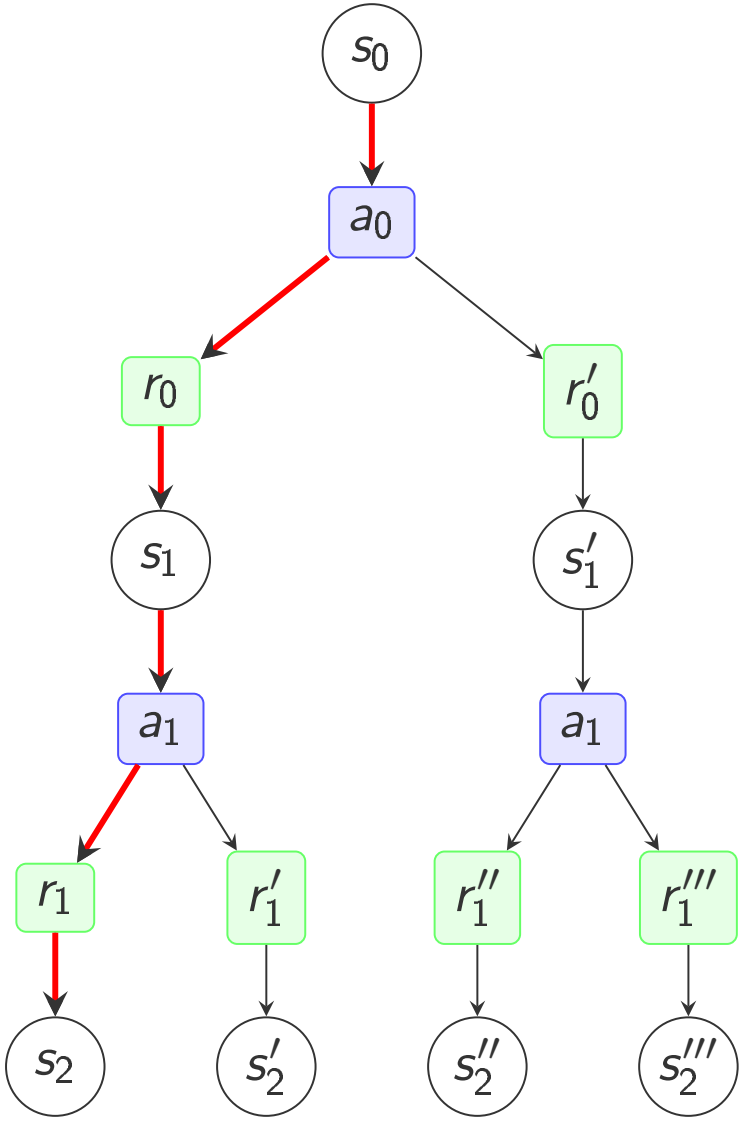
Trace τ: Branching Expansion

Trace as a sequence:

$$\tau_0 = \{s_0, a_0, r_0, s_1, a_1, r_1, s_2, \dots\}$$

But in practice:

- From each state s_t and action a_t,
- The environment may branch into different s_{t+1}.
- So a trace is one *path* through this tree.



Stochastic Spaces and Distribution over Traces

- Both the policy π and transitions T can be stochastic.
- So proceeding from the start start will not always produce the same trace.
- Instead, we get a **distribution over traces**:

$$p(\tau_0) = \text{probability of complete trace from start state } s_0$$

- Probability of a trace = product of probabilities of its transitions:

$$p(\tau_0) = p_0(s_0) \cdot \prod_{t=0}^{\infty} \pi(a_t|s_t) \cdot T_{a_t}(s_t, s_{t+1})$$

Distribution over Traces: Breaking Down the Equation

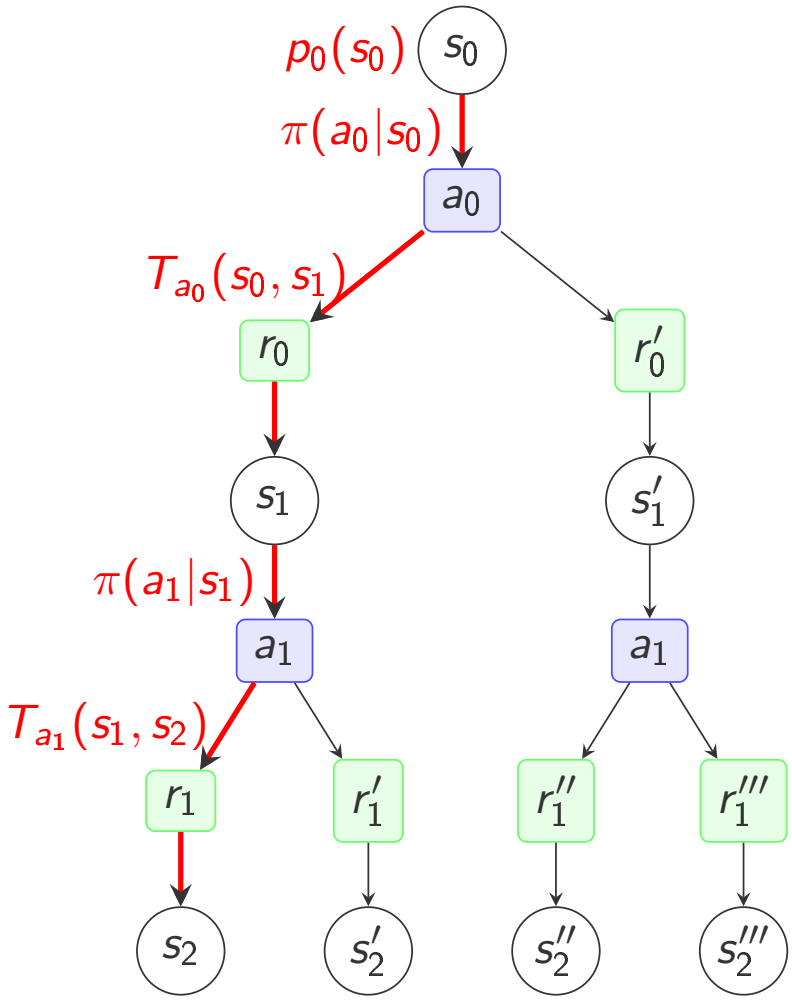
- The trace $\tau_0 = \{s_0, a_0, r_0, s_1, a_1, r_1, \dots\}$ is one possible path.
- Its probability depends on:
  - $p_0(s_0)$ = probability of starting in state s_0,
  - $\pi(a_t|s_t)$ = probability of choosing action a_t in state s_t,
  - $T_{a_t}(s_t, s_{t+1})$ = probability of transitioning to s_{t+1} after action a_t.
- Multiply these step probabilities together for the full trace:

$$p(\tau_0) = p_0(s_0)\, \pi(a_0|s_0)\, T_{a_0}(s_0, s_1)\, \pi(a_1|s_1)\, T_{a_1}(s_1, s_2)\, \cdots$$

- Compact notation:

$$p(\tau_0) = p_0(s_0) \cdot \prod_{t=0}^{\infty} \pi(a_t|s_t)\, T_{a_t}(s_t, s_{t+1})$$

Distribution over Traces: Breaking Down the Equation



A trace is just one path in the MDP tree, and its probability is the product of the branching probabilities along that path.

Traces in RL

- **Policy-based RL**: depends heavily on full traces.
- **Value-based RL**: often uses single transition steps.
- $\Rightarrow$ Both approaches build on the idea of traces.

Return R: What Are We Optimizing?

- Goal of sequential decision-making: **Find the best policy**.
- To evaluate a policy, we need a measure of *long-term success*.
- This measure is the **return**: the sum of rewards collected along a trace.

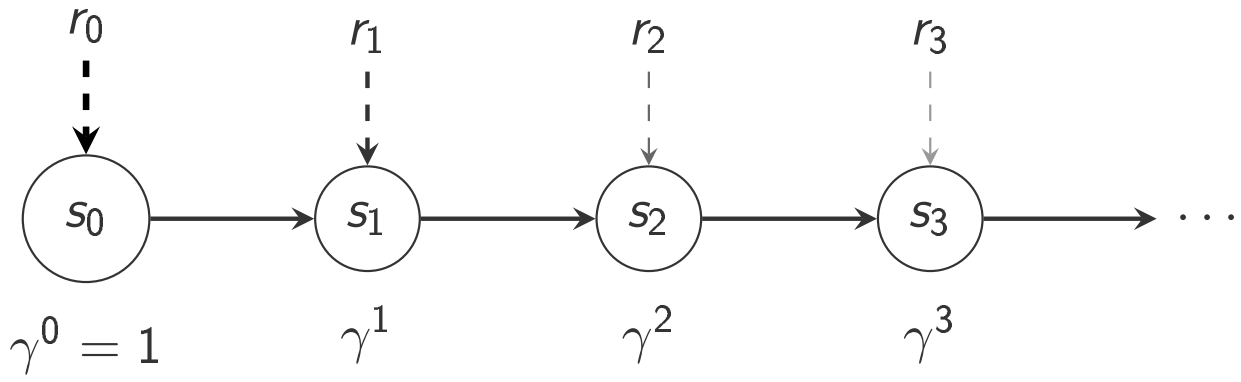
Definition of Return R

- For a trace $\tau_t = (s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots)$
- The **return starting at time** t is:

$$R(\tau_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

- Compact form:

$$R(\tau_t) = r_t + \sum_{i=1}^{\infty} \gamma^i r_{t+i}$$



Discount Factor γ

- $\gamma \in [0, 1]$: balances short-term vs. long-term rewards.
- Two extremes:
  - $\gamma = 0$: **Myopic agent** (only immediate reward matters)
  - $\gamma = 1$: **Far-sighted agent** (all rewards equally important)
- In *infinite-horizon* tasks:
  - $\gamma = 1$ can lead to unbounded returns.
  - Typically use $\gamma \approx 0.99$ to keep returns finite.

Example: Computing Return

Example: Recall the sample trace.

- Assume $\gamma = 0.9$
- Rewards along trace: $r_0 = -1,\ r_1 = -1,\ r_2 = 20$
- Return is:

$$R(\tau_0^2) = -1 + 0.9(-1) + 0.9^2(20)$$

$$= -1 - 0.9 + 16.2 = 14.3$$

Intuition: Why Discounting?

- Ensures **mathematical stability**: infinite sum remains bounded.
- Captures the idea of **time preference**:
  - Immediate rewards are more certain/valuable.
  - Future rewards are less predictable.
- γ lets us trade off:
  - Short-term exploitation
  - Long-term exploration

From Traces to Expectations

- Return of a single trace is not enough.
- The environment can be stochastic $\Rightarrow$ different traces possible.
- The policy π can also be stochastic $\Rightarrow$ actions may vary.
- Therefore: we care about the **expected cumulative reward**.

Expectation

- The **expected value** of a random variable X is the long-run average outcome.
- Definition:

$$\mathbb{E}[X] = \sum_x p(x) \cdot x \quad \text{(discrete)}$$

$$\mathbb{E}[X] = \int x\, p(x)\, dx \quad \text{(continuous)}$$

- Think of expectation as a **weighted average**, where outcomes are weighted by their probability.

Expectation Example (Unfair Die)

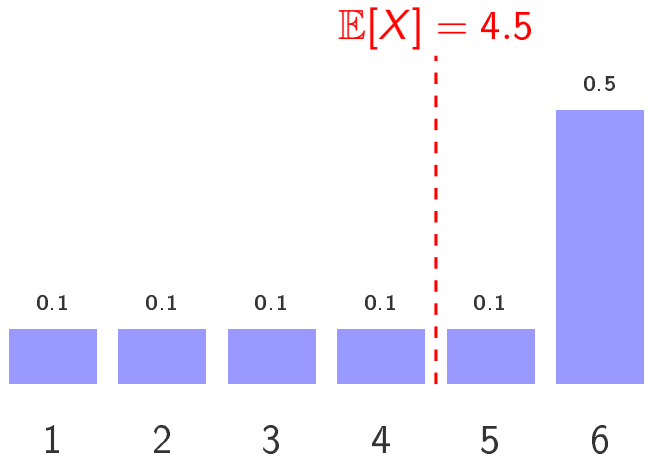
- Suppose a die is biased:

X	1	2	3	4	5	6
$p(X)$	0.1	0.1	0.1	0.1	0.1	0.5
$X \cdot p(X)$	0.1	0.2	0.3	0.4	0.5	3.0

- Expected value:

$$\mathbb{E}[X] = 0.1(1 + 2 + 3 + 4 + 5) + 0.5 \cdot 6 = 0.1 \cdot 15 + 3 = 4.5$$

- Unlike the fair die ($\mathbb{E}[X] = 3.5$), our unfair die's expectation is skewed towards 6.



Expectation Example in RL

- In RL, rewards are random because:
  - The policy π may be stochastic.
  - The environment transitions T may be stochastic.
- Just like the number on the die, cumulative reward is a random variable.
- Therefore, the *value function* is an **expectation over returns**:

$$V^{\pi}(s) = \mathbb{E}\Big[R(\tau) \,|\, s_0 = s, \pi\Big]$$

- Just like the unfair die rolls average to 4.5, the value function averages over many possible traces.

State Value: Definition

- The **state value function** $V^{\pi}(s)$:

$$V^{\pi}(s) = \mathbb{E}_{\tau_t \sim p(\tau_t)} \left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \;\middle|\; s_t = s \right]$$

- Meaning:
  - Start in state s.
  - Follow policy π.
  - Value = expected return.

> $V^{\pi}(s)$ is the *expected return* when you *start from state s* and *follow policy π*.

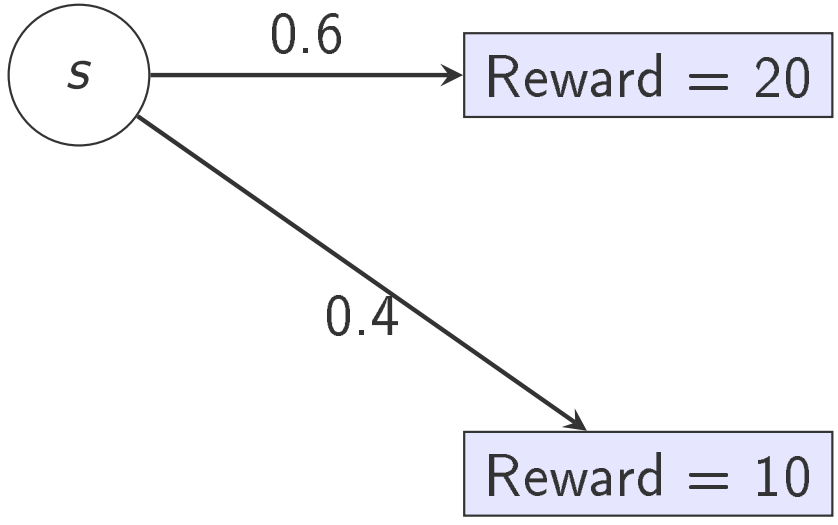
Example: Two Possible Traces

Imagine from state s under policy π:

- With probability 0.6 cumulative reward is 20
- With probability 0.4 cumulative reward is 10

Then:

$$V^{\pi}(s) = 0.6 \cdot 20 + 0.4 \cdot 10 = 16$$



State Value Function

- Every policy π defines a unique value function $V^\pi(s)$.
- Often we drop π and just write $V(s)$.
- $V(s)$ assigns a real number (expected return) to each state.

Example: Tabular state values (discrete state space)

State s	$V^\pi(s)$
1	2.0
2	4.0
3	1.0
etc.	...

Terminal States

- By definition:

$$s = \text{terminal} \quad \Rightarrow \quad V(s) = 0$$

- Once the episode ends, no further rewards are possible.

From $V(s)$ to $Q(s, a)$

- State value $V^\pi(s)$: expected return starting from state s and following π.
- But sometimes we want to know the value of a *specific action*.
- So we can define the *state-action value* $Q^\pi(s, a)$:

$$Q^\pi(s, a) = \mathbb{E}_{\tau_t \sim p(\tau_t)}\Big[\sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} \;\Big|\; s_t = s, a_t = a\Big]$$

- Meaning: the expected return if we take action a in s, then follow π.

Formal Definition

- Every policy π has exactly one associated Q-function.
- Domain and codomain:

$$Q : S \times A \to \mathbb{R}$$

- Each state-action pair (s, a) is mapped to the expected return.
- Terminal states: by convention

$$s = \text{terminal} \quad \Rightarrow \quad Q(s, a) := 0, \quad \forall a$$

Intuition: Why $Q(s, a)$?

- $V(s)$ tells us how good a state is, on average, under π.
- $Q(s, a)$ tells us how good it is to *take action a* in state s.
- If we know Q, we can easily pick the best action:

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

- This is why Q is central in reinforcement learning.

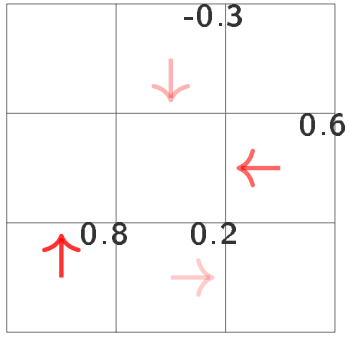
Example: Tabular Representation of $Q(s, a)$

For discrete S and A, Q can be stored in a table of size $|S| \times |A|$.

	a=up	a=down	a=left	a=right
s=1	4.0	3.0	7.0	1.0
s=2	2.0	-4.0	0.3	1.0
s=3	3.5	0.8	3.6	6.2
etc.	.	.	.	.

Visual Intuition: Grid World

- In a grid world:
  - Each state s = cell in the grid.
  - Each action a = arrow direction.
  - $Q(s, a)$ = expected value of moving in that direction.



Numerical Example: Computing $Q^\pi(s, a)$

Setup. From state s we evaluate action a_1.

- Discount: $\gamma = 0.9$
- After taking a_1 in s:
  - With prob. 0.6: get $r_0 = 2$, go to s_1; under π, next step gives $r_1 = 5$, then terminal.
  - With prob. 0.4: get $r_0 = -1$, go to s_1'; under π, next step gives $r_1 = 10$, then terminal.

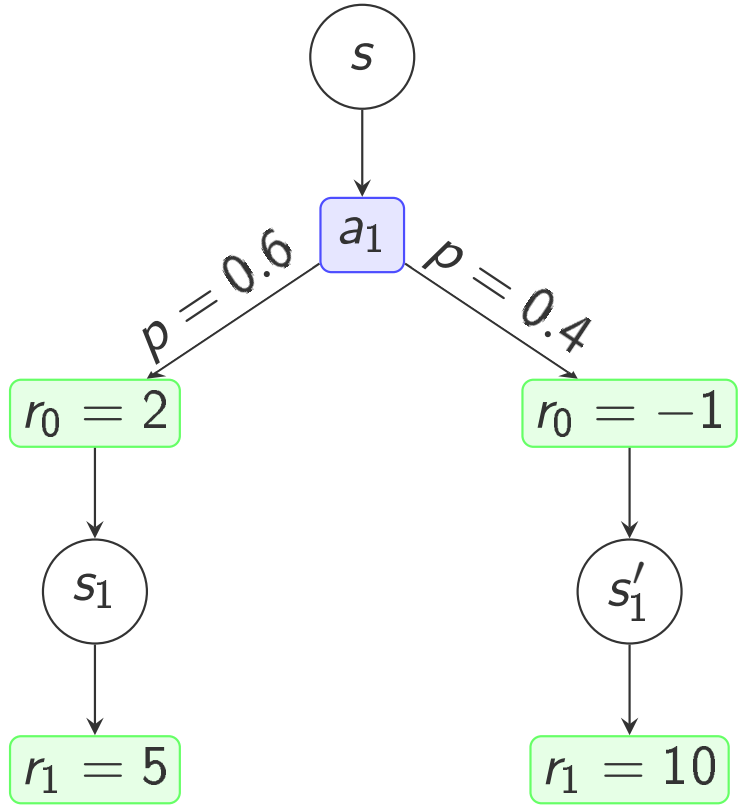
Branch returns (2-step horizon for clarity):

$$G_1 = r_0 + \gamma r_1 = 2 + 0.9 \times 5 = 6.5, \qquad G_2 = r_0 + \gamma r_1 = -1 + 0.9 \times 10 = 8.0$$

Expected return (definition of Q^π):

$$Q^\pi(s, a_1) = 0.6 \times G_1 \; + \; 0.4 \times G_2 \; = \; 0.6 \times 6.5 \; + \; 0.4 \times 8 \; = \; 7.1$$

Visualization of the Example (Branches & Probabilities)



$G_1 = 2 + 0.9 \times 5 = 6.5$

$G_2 = -1 + 0.9 \times$

$$\boxed{Q^\pi(s, a_1) = 0.6 \times 6.5 \; + \; 0.4 \times 8 \; = \; \mathbf{7.1}}$$

Compare Two Actions via Q: Choose $\arg\max_a Q(s,a)$

Action a_1 (from previous slide): $Q^\pi(s, a_1) = 7.1$.

Alternative action a_2:

- With prob. 0.7: $r_0 = 3$, next $r_1 = 2$ (then terminal)
- With prob. 0.3: $r_0 = 3$, next $r_1 = 0$ (then terminal)

$$G_1^{(a_2)} = 3 + 0.9 \times 2 = 4.8, \qquad G_2^{(a_2)} = 3 + 0.9 \times 0 = 3.0$$

$$Q^\pi(s, a_2) = 0.7 \times 4.8 + 0.3 \times 3.0 = 3.36 + 0.9 = 4.26$$

$$\pi^*(s) = \arg \max_{a \in \{a_1, a_2\}} Q^\pi(s, a)$$

$$= a_1 \text{ (since } 7.1 > 4.26)$$

Trace View and Factorization of Probabilities

Each branch is a partial trace (here: 2 time steps after taking a in s).
For longer horizons,

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \;\middle|\; s_t = s, a_t = a\right]$$

If transitions and policy are stochastic, each *full* trace probability factors as:

$$p(\tau_t) = \pi(a_t|s_t) \cdot T_{a_t}(s_t, s_{t+1}) \cdot \pi(a_{t+1}|s_{t+1}) \cdot T_{a_{t+1}}(s_{t+1}, s_{t+2}) \cdots$$

Then $Q^{\pi}(s,a)$ is the expectation of discounted returns over all such traces conditioned on $(s_t = s, a_t = a)$.

Reinforcement Learning Objective

- We now have the ingredients to formally state the **objective** $J(\cdot)$ of reinforcement learning.
- The objective: achieve the highest possible **average return** from the start state.

$$J(\pi) = V^{\pi}(s_0) = \mathbb{E}_{\tau_0 \sim p(\tau_0|\pi)}\Big[R(\tau_0)\Big].$$

- $V^{\pi}(s_0)$: value of the start state under policy π
- Expectation is taken over trajectories τ_0 sampled from $p(\tau_0|\pi)$

Optimal Policy

- There exists one **optimal value function**, which achieves higher or equal value than all others.
- The corresponding policy is called the **optimal policy** $\pi^\star$.

$$\pi^\star(a|s) = \arg\max_\pi V^\pi(s_0)$$

- arg max selects the policy π that maximizes the expected return.
- Goal in RL: find $\pi^\star$ for the start state s_0.

State Values vs. State-Action Values

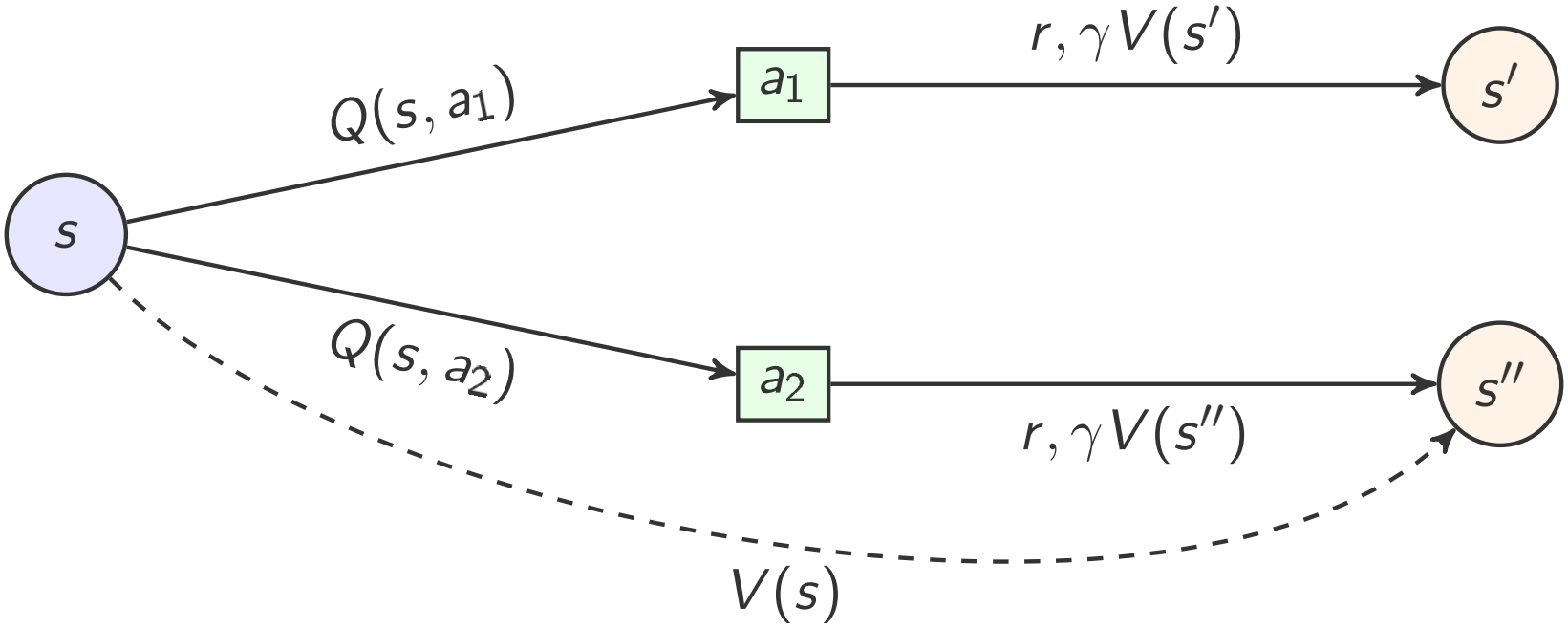
- Potential benefit of using **state-action values** Q over state values V:
  - Q directly tells what every action is worth.
  - Useful for action selection in discrete action spaces.

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

- The Q function identifies the best action directly.
- Equivalently:

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

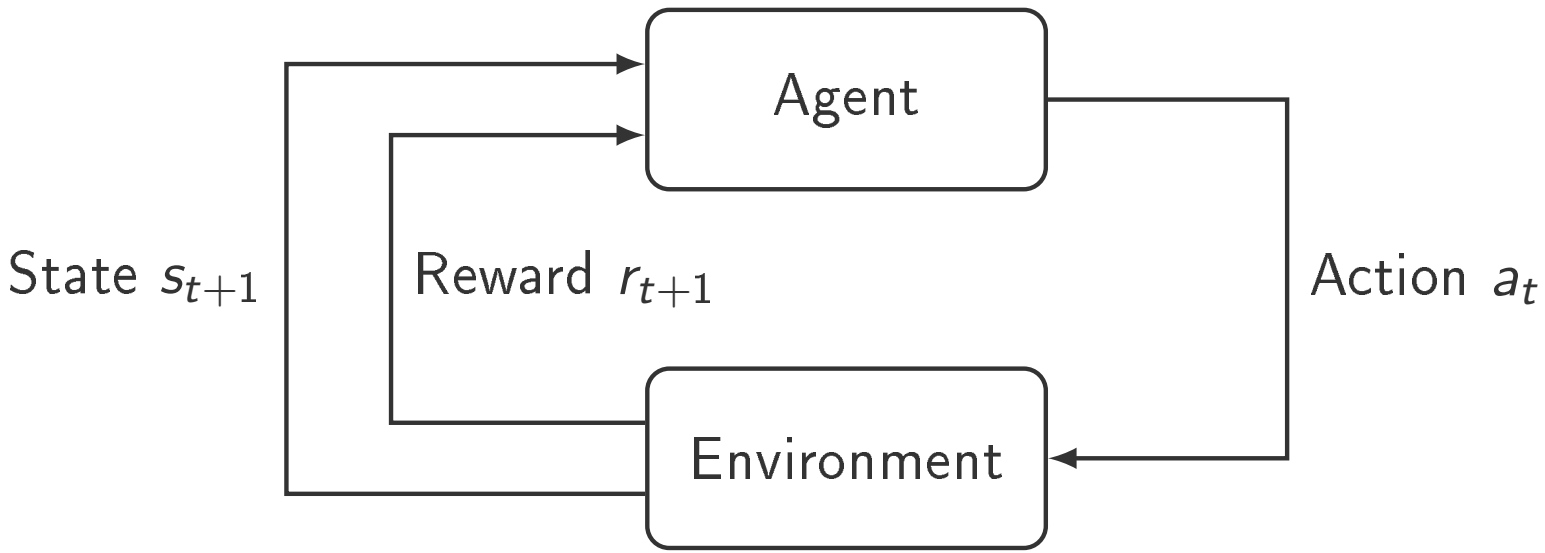
Visualizing $V(s)$ vs. $Q(s, a)$



- $V(s)$: value of a state = expected return from that state.
- $Q(s, a)$: value of taking action a in state s.

RL Objective

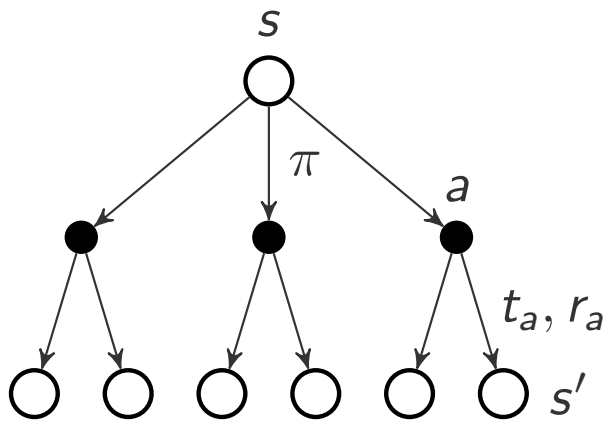
- Objective: maximize expected cumulative reward from the start state.
- Optimal policy $\pi^\star$ achieves this maximum.
- Q-values make it easy to select optimal actions directly.



Agent interacting with environment to maximize objective $J(\pi)$.

The Bellman Equation

- To compute the value function $V(s)$:
  - Imagine the state-space tree, expanded to cover all possible states.



  - The value of a parent node depends on:
    - Immediate reward at that step.
    - Discounted sum of its children's values.
- If you know all $V(s')$, you can compute $V(s)$.
- Recursive structure $\Rightarrow$ dynamic programming.

Richard Bellman

- Richard Bellman formalized this recursive approach in 1957.
- Introduced the term *dynamic programming*.
- Showed that many discrete optimization problems can be solved by *backward induction*.

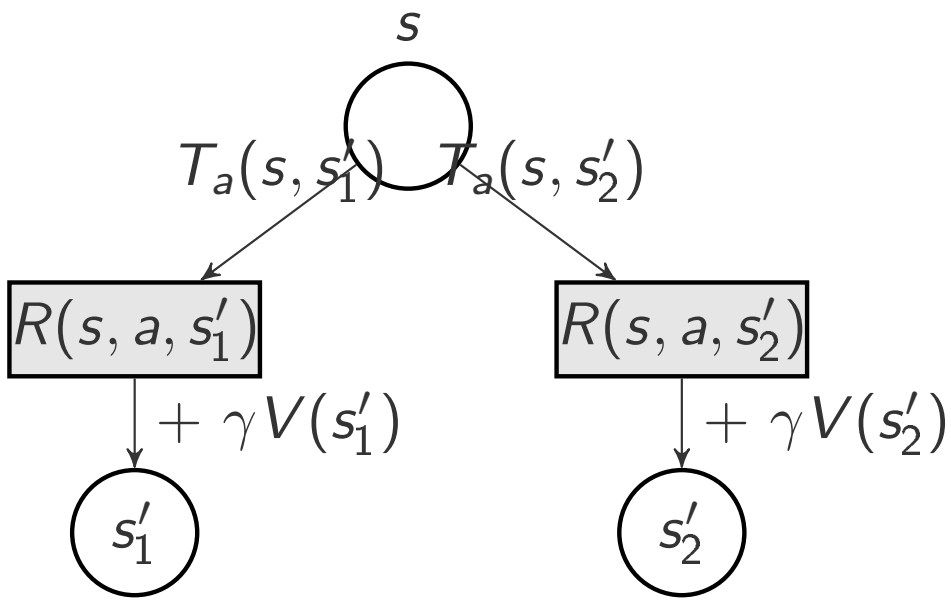
Richard Bellman (1920–1984)

Bellman Equation (Intuition)

- The value of a state s depends on:
  - The policy π: how likely we are to take action a in s.
  - The transition function T: probability of moving to s'.
  - The reward R: immediate gain for transition (s, a, s').
  - The discount factor γ: importance of future rewards.
- Recursive definition:

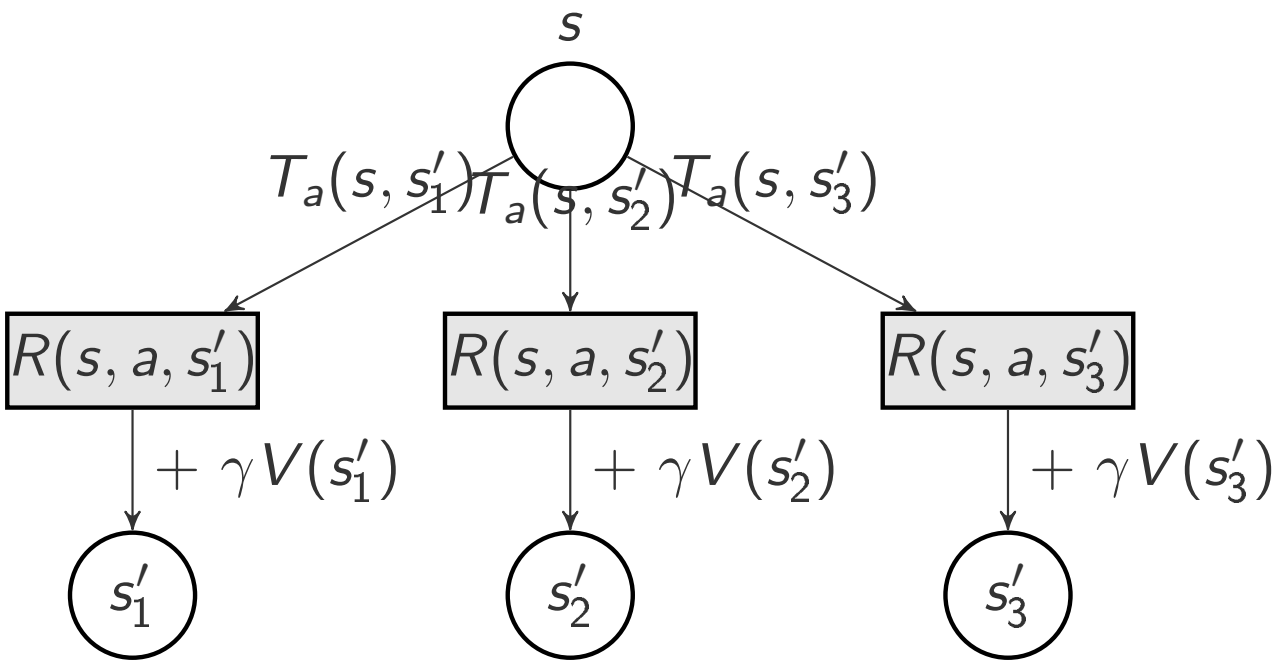
$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} T_a(s, s') \Big[R_a(s, s') + \gamma V^{\pi}(s') \Big]$$

Bellman Equation as a Recursive Tree



Each child contributes: $T_a(s, s')\big[R(s, a, s') + \gamma V(s')\big]$

Bellman Equation as Recursive Expansion



General form: $V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} T_a(s, s') \Big[R_a(s, s') + \gamma V^\pi(s') \Big]$

Key Notes

- **Recursion:** The value of s depends on future values $V^{\pi}(s')$.
- **Dynamics model:** Requires knowledge of T (transitions) and R (rewards).
- Often, these are **unknown** in practice $\Rightarrow$ model-free RL.
- Bellman equation is the foundation of RL algorithms:
  - Value iteration
  - Policy iteration
  - Q-learning
- We will study them in the next lecture.