In the Name of God

Iran University of Science and Technology

Reinforcement Learning in Control

Dr. Saeed Shamaghdari

Electrical Engineering Department
Control Group

Fall 2025 | 4041

Finite Markov Decision Processes

Markov Decision Processes

It implies that the action taken by our agent is conditional solely on the present state-action and independent of the past states and actions.

Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}\left[S_{t+1} \mid S_t\right] = \mathbb{P}\left[S_{t+1} \mid S_1, \ldots, S_t\right]$$

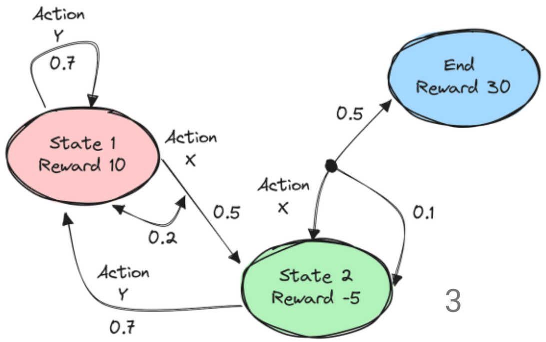
Computing the state trajectory

The effect of an action on both immediate and future rewards

Value Function:

Multi-armed bandit : $q_*(a)$

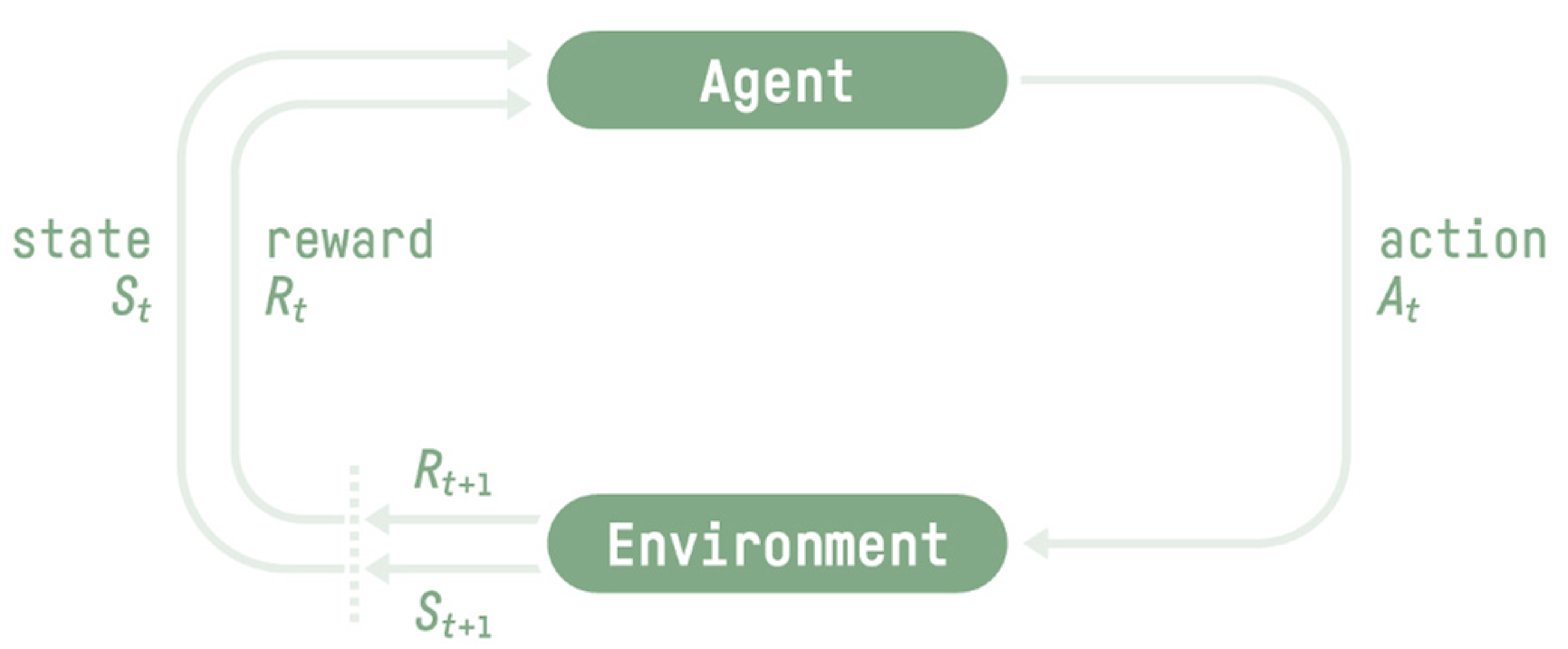
MDP in general : $q_*(s, a)\ or\ V_*(s)$ **RL and MDP?**



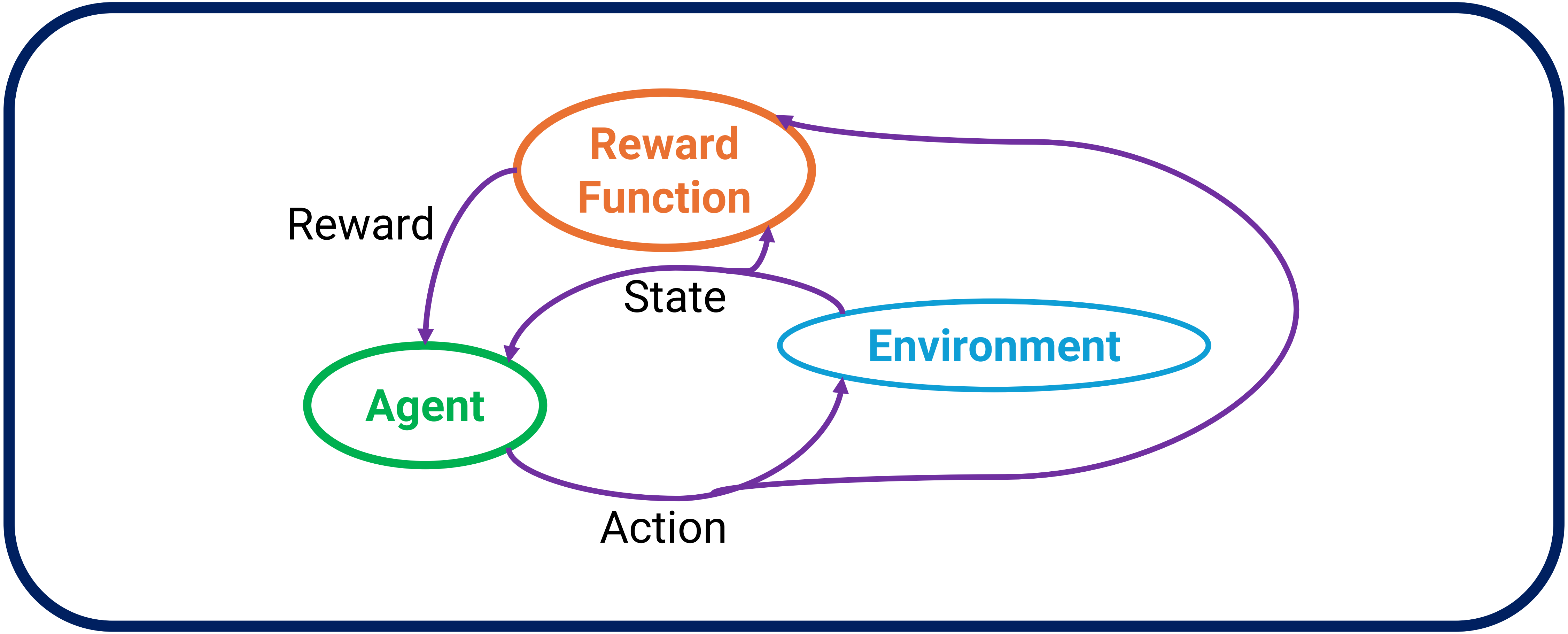
Agent and Environment

Agent?

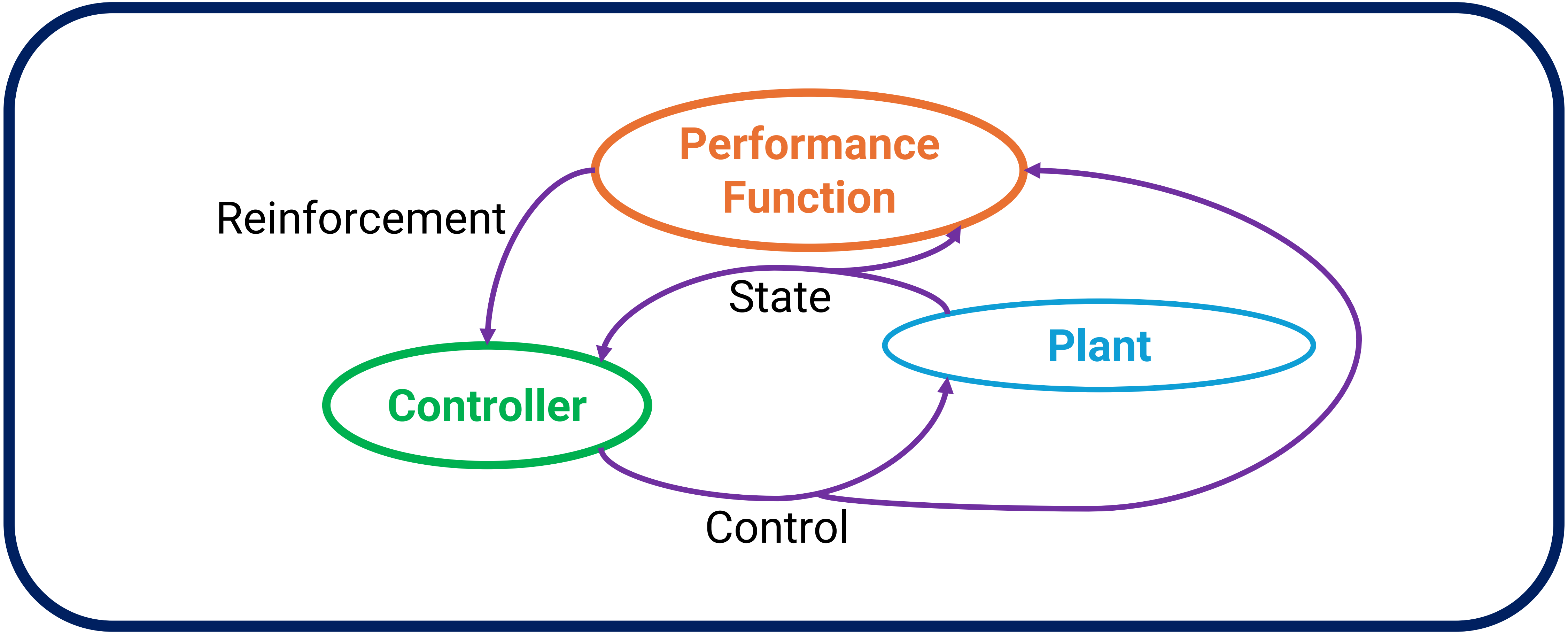
Environment?



Reinforcement Learning in Artificial Intelligence



Reinforcement Learning in Control Engineering



At each time step t=0,1,2,3,…

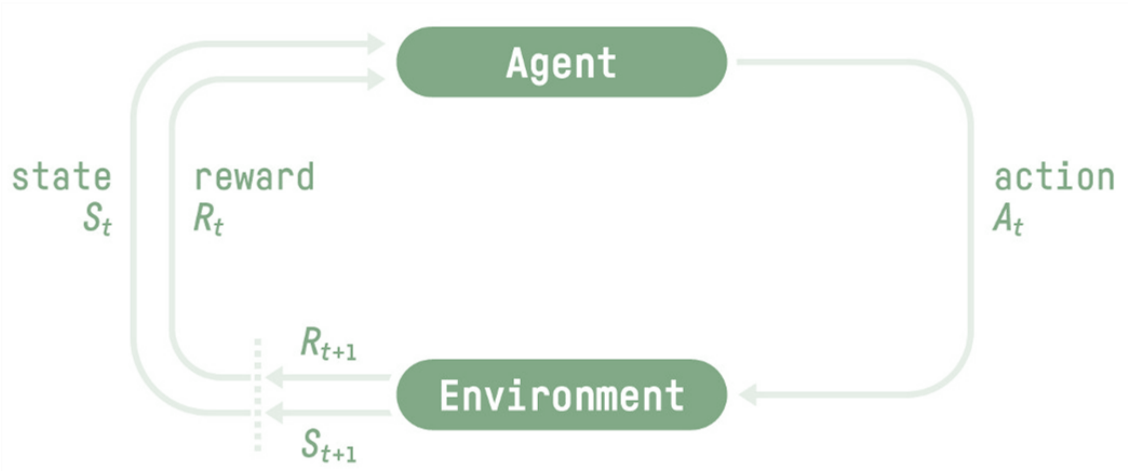
The agent receives the environment state: $S_t \in \mathcal{S}$

It selects an action based on a policy: $A_t \in \mathcal{A}(s)$

In response to A_t:

the environment transitions to a new state S_{t+1}, and the agent receives a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

A_t, S_t: random variables



System trajectory:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

MDP Dynamic

Definition

The function *p* defines the dynamics of the MDP:

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

Computing the probability of transitioning to s' as the next state and receiving reward r

$$s', s \in \mathcal{S},\ r \in \mathcal{R},\ \text{and}\ a \in \mathcal{A}(s)$$

$$p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$$

P: A complete description of the environment's dynamics!

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Deriving various state-transition functions from the transition probability function p

$$p(s' \mid s, a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

$$p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1])$$

The expected rewards for state–action pairs:

$$r(s, a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

$$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

Deriving various state-transition functions from the transition probability function p

The expected rewards for state−action−next-state triples:

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1}=s, A_{t-1}=a, S_t = s'] = \sum_{r\in\mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

$$r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$$

If we want to derive the probability density function of one variable from the joint probability density function while knowing the other variable, it is called the **conditional probability function**.

$P(s', r \mid s, a)$ is known so $p(r \mid s, a, s') = \frac{P(s', r \mid s, a)}{P(s' \mid s, a)}$

Now, if we want to compute $E[R_t]$:

$$E[R_t \mid s_{t-1} = s, R_{t-1} = a, s_t = s'] = \sum_{r\in R} rP(r \mid s, a, s')$$

Determining Action and State

Action? Low-level Control → High-level Control

Motor voltage of a robot arm

Robot movement (left/right)

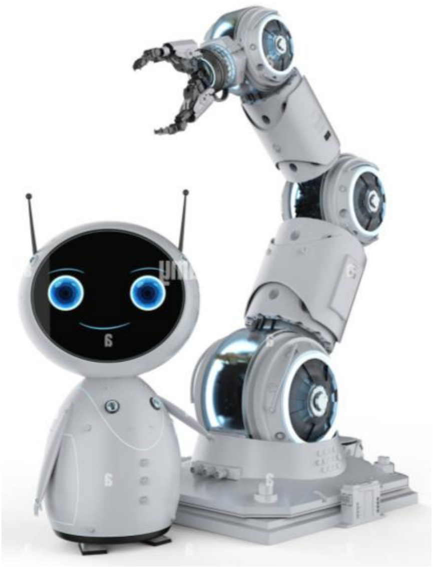
Decision-making to achieve the goal

State? Low-level State → High-level State (sensor output)

Position of robot arm

Robot mobility status (moving/stationary)

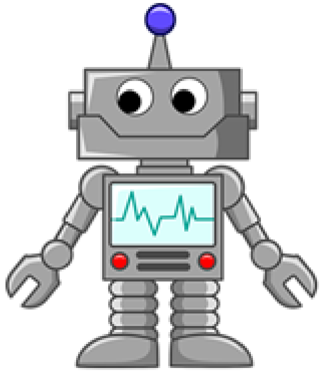
Required knowledge for goal achievement



Boundary between the environment and the agent?

Robot
Animal

Environment → non-agent (cannot be changed by the agent)



What information does the agent have about the environment?

Complete
Partial (e.g., how rewards are calculated, how states change)
None

Boundary between the environment and the agent:

The limit of the agent's control (may vary depending on the objective)

RL and MDP

Modeling the RL problem in the MDP framework:

- Action: Agent's control signal
- State: Effect of the agent's signal
- Reward: Signal representing the agent's goal

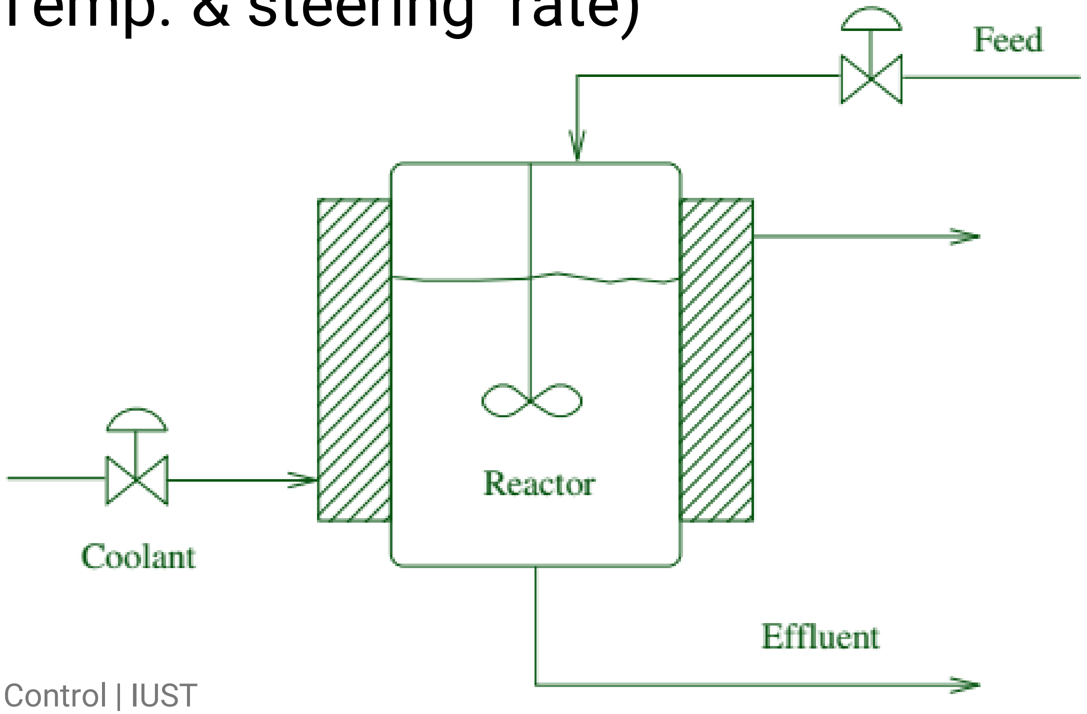
RL performance ⇆ appropriate selection of action and state

Examples

Bioreactor

States: Temperature , Concentrations

Action: (control valve) → set points (Temp. & steering rate)
Reward: chemical product rate



Examples

Pick and Place

Goal: Fast and smooth movement!

Action?

Motor voltage for each joint

State?

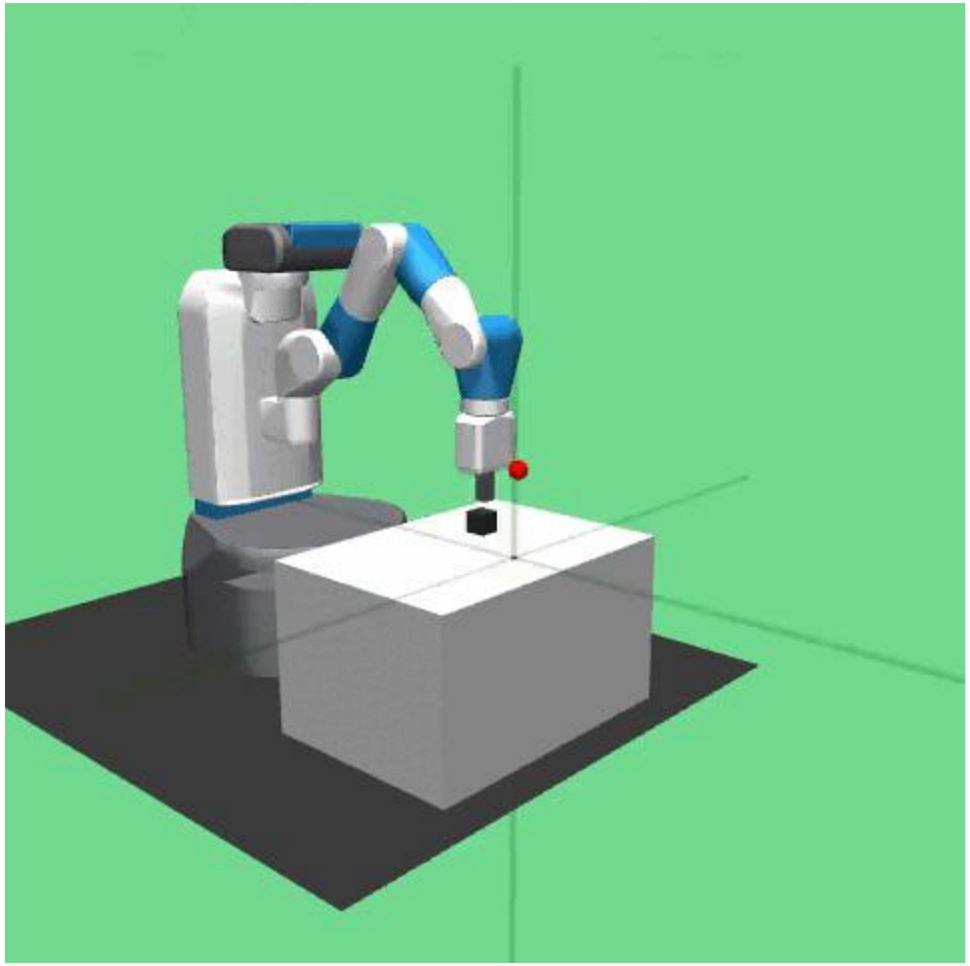
Joint angles and angular velocities

Reward?

+1 for each successful optimal movement

$-\epsilon$ for each collision or impact

$-\beta$ for each sample time



Examples

Action in driving?

Wheel torque (in both directions)
Brake, accelerator, steering
Driver's hand and foot commands
Which path to drive on

Which one do you prefer?

Examples

Recycling Robot

Sensors...
Actuators...
Navigation and control...
Rechargeable battery
Action?
State?

High-level definition:
State: Battery charge level: *low – high*
Action: {Search – Return to charging station – Wait} or {Search – Wait}

Examples

Recycling Robot

System Dynamics:

Search

- If the battery is *high*, it stays *high* with probability α, and drops to *low* with probability $1-\alpha$
- If the battery is *low*, it stays *low* with probability β, and becomes *empty* with probability $1-\beta$

Wait

- No battery consumption

Recharge

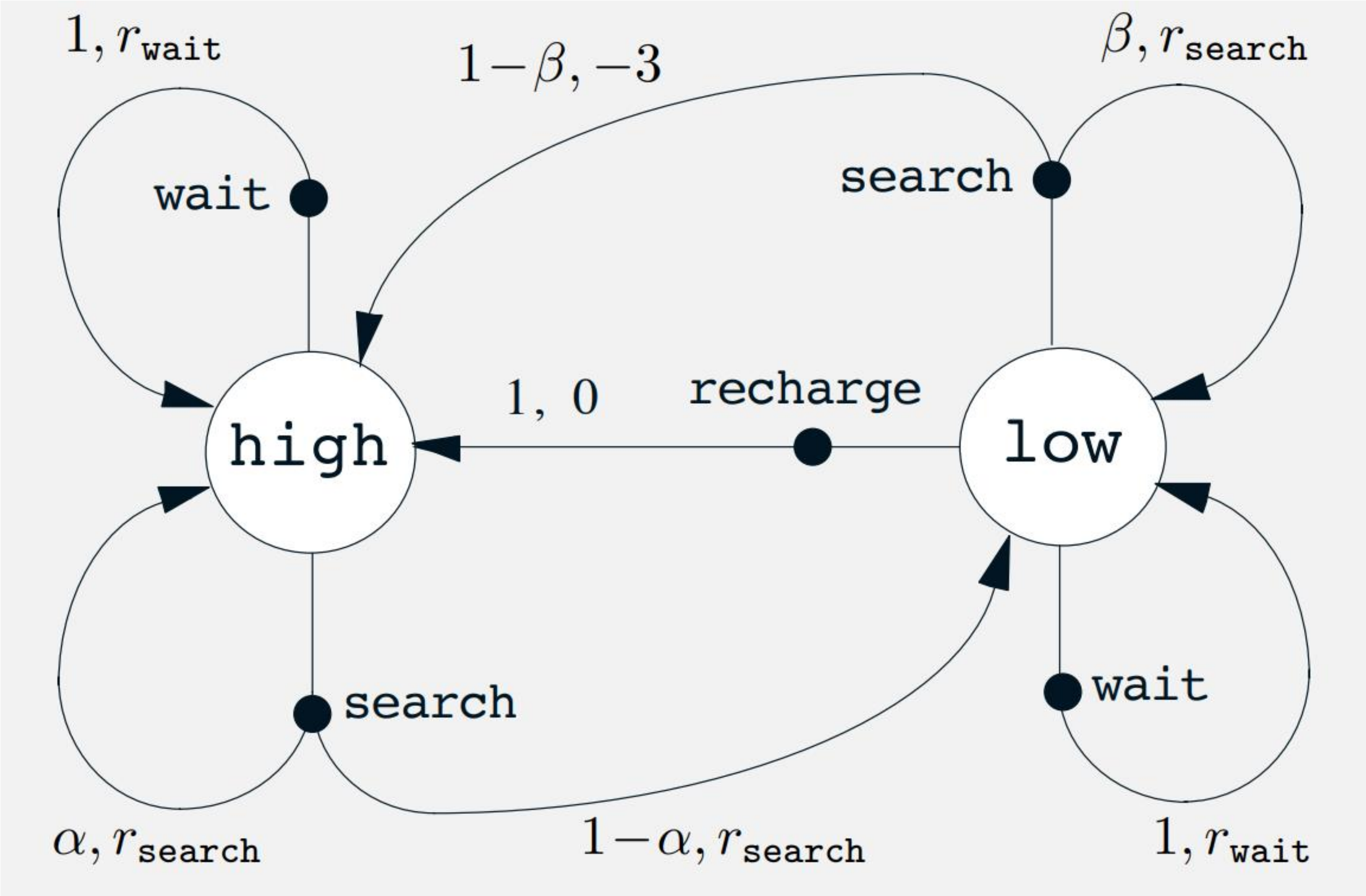
- Transitions from *low* to *high*

Reward:

- Trash placed in the bin:
  +1 × (*r_search* or *r_wait*)
- Battery becomes empty: −3

Examples

Finite MDP Dynamics Diagram for the Robot:



Examples

Transition Probability and Reward Function for a Finite MDP:

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1 - \alpha$	$r_{\texttt{search}}$
low	search	high	$1 - \beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-

Defining the Goal for the Agent

A numerical reward signal that guides the agent toward its goal by maximizing its expected value.
(or cumulative reward).

Limitations?

Definition of Reward

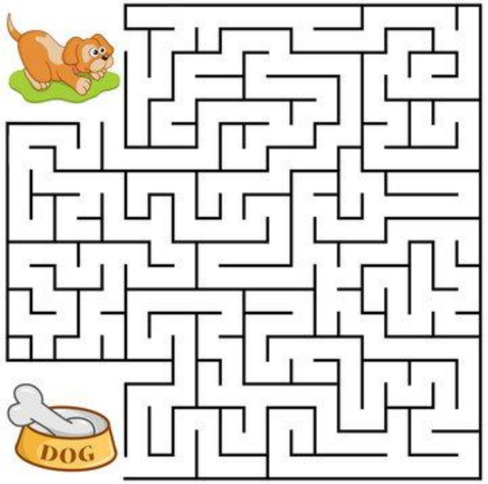
- ❑ Represents *what* we want to achieve, not *how* to achieve it.
- ❑ Does not provide prior knowledge (i.e., initialize the policy or value)
- ❑ A way to communicate with the robot (or agent)

If the agent maximizes the reward, it should reach the goal.

Examples

Chess

Maze



Examples

Go (one of the oldest and hardest board games)

Agent: Player

Environment: Opponent

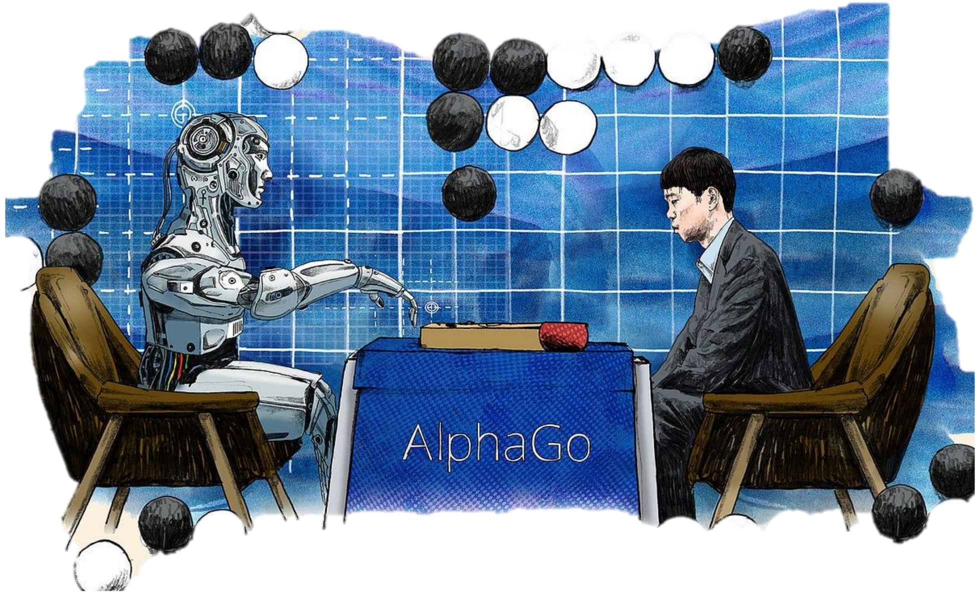
State: Board configuration

Action: Next stone placement

Reward:

+1 for win

-1 for loose



2016 Milestone: AlphaGo defeats world champion Lee Sedol (4-1).

Return and Episodes

Formulating Long-Term Reward

In the simplified form: the sum of rewards from time step *t* onward

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

Objective: Maximize the expected return.

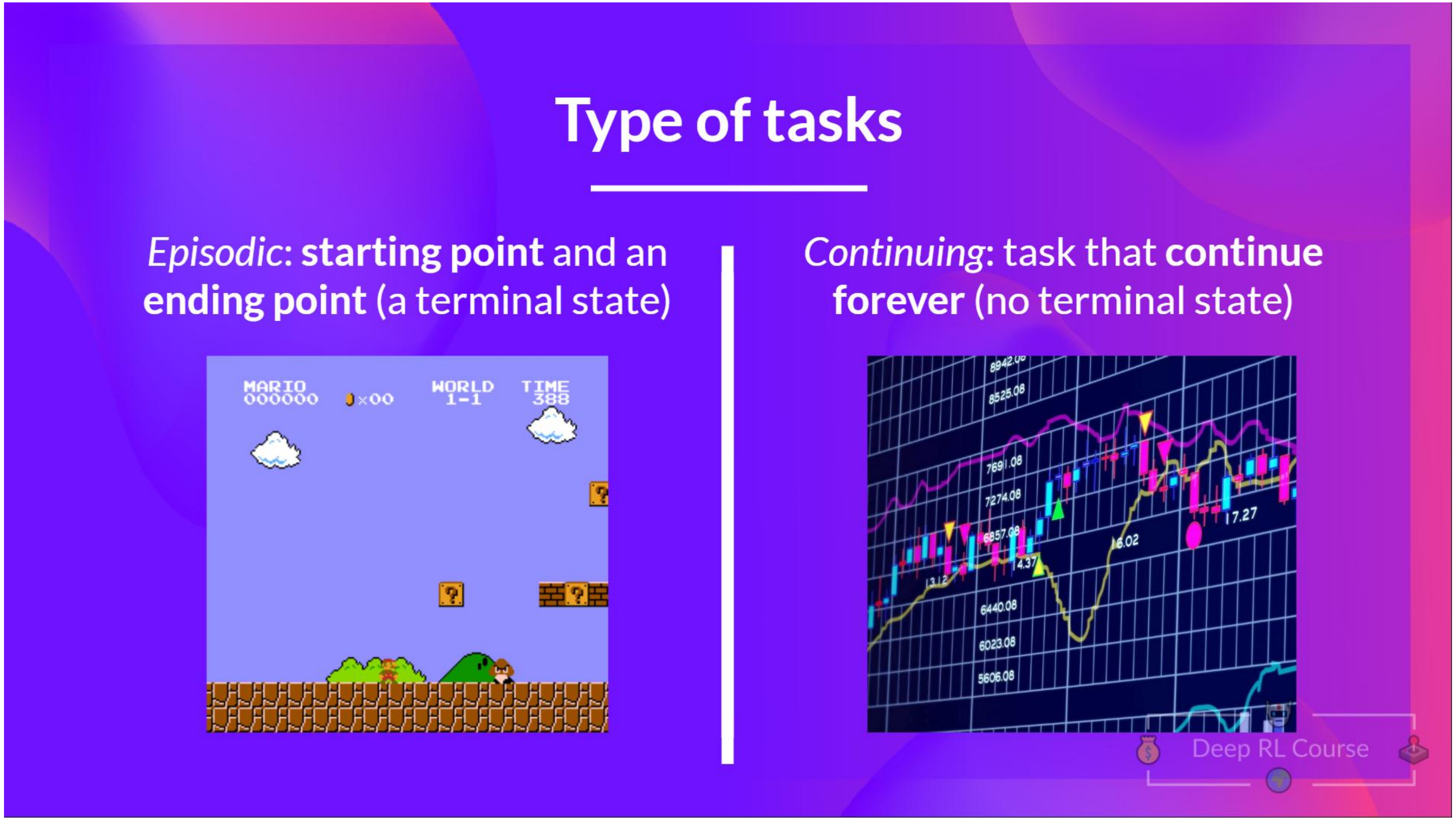
Episode:

An interaction between the agent and the environment that has an end. It includes a **terminal state**.

Continuing Task:

An ongoing interaction between the agent and the environment with **no terminal state**.

To Recap …


Type of tasks
Episodic: **starting point** and an **ending point** (a terminal state)
Continuing: task that **continue forever** (no terminal state)
MARIO
000000
x00
WORLD
1-1
TIME
388
Deep RL Course

Discounted Return

Definition

The *return* G_t is the total discounted reward from time-step t.

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

$0 < \gamma < 1 \quad \rightarrow \quad G_t < \infty$

$\gamma \to 0?$ γ close to 0 leads to "myopic" evaluation

$\gamma \to 1?$ γ close to 1 leads to "far-sighted" evaluation

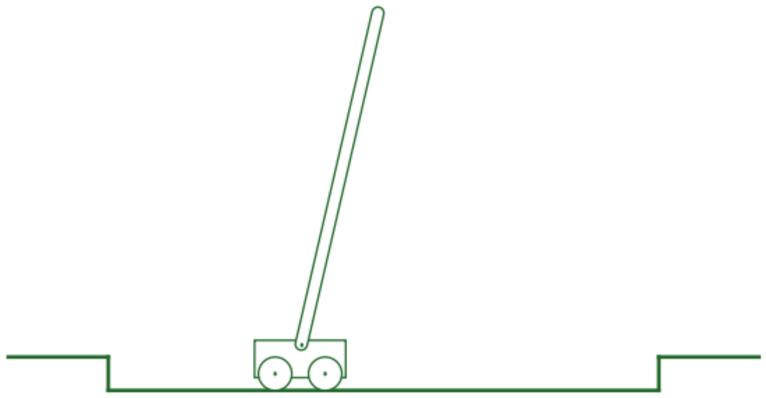
Recursive Calculation of Return

$$
\begin{aligned}
G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&= R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots\right) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

Example

Constrained Inverted Pendulum

$$\theta_{min} < \theta < \theta_{max}$$



Episodic or Continuing?

Episodic:

Execution of each task from the start until the moment the pendulum falls.

Reward:

+1 for every step without failure.

Return:

Sum of rewards from the start until failure.

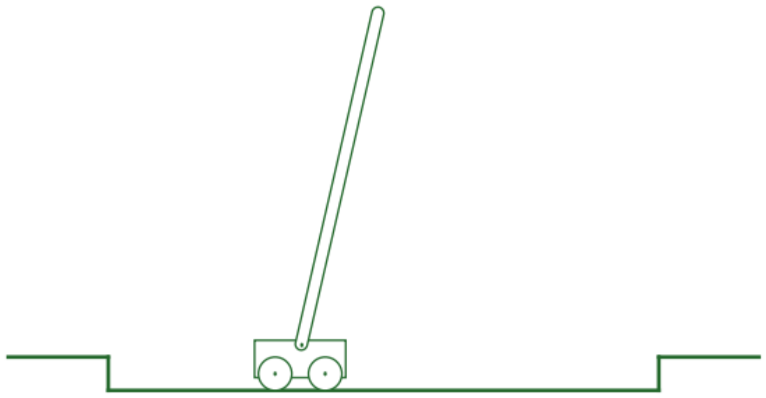
Successful balancing:

Return $\to \infty$

Example

Constrained Inverted Pendulum

$$\theta_{min} < \theta < \theta_{max}$$



Episodic or Continuing?

Continuing:

Consecutive tasks that restart after each failure, beginning from a mid-state.

Reward:

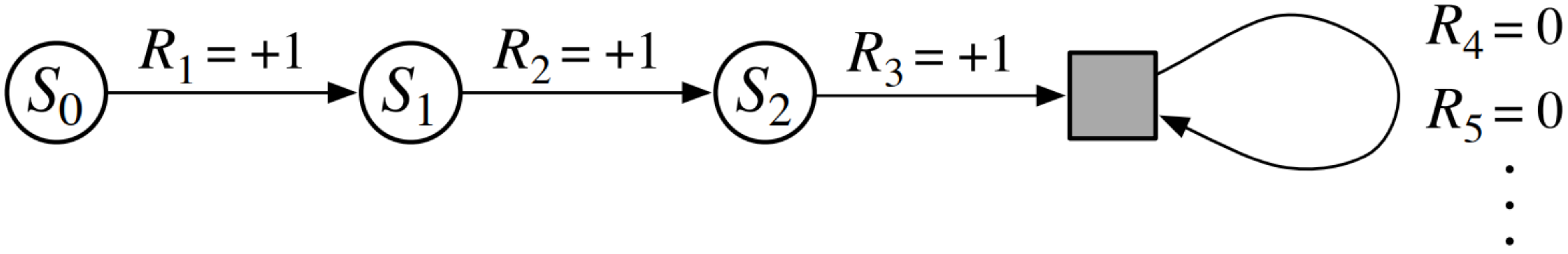
−1 for each failure.

Discounted Return:

Up to $-\gamma^K$

(*K: time step at which failure occurs*)

Unified Notation for Episodic and Continuing Tasks

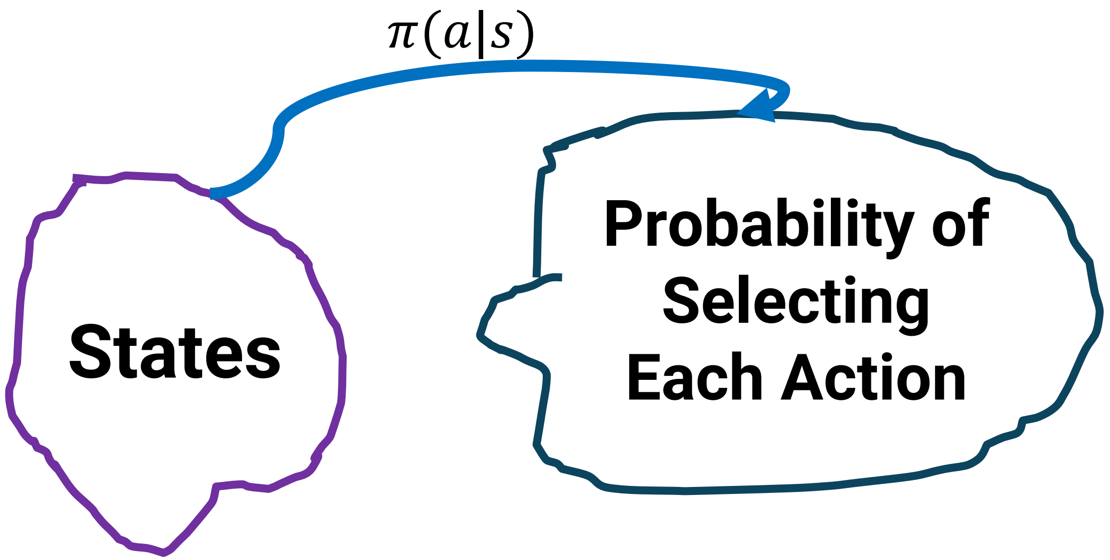


$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$T = \infty \text{ or } \gamma = 1 \text{ (but not both)}$$

Reinforcement Learning Problems

Value Function Estimation $\begin{cases} \text{State} \\ \text{State \& Action} \end{cases}$



Value: Expected return

Policy: Action selection methods

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

Value Functions

If we are in state : s and policy π has been selected,

Definition

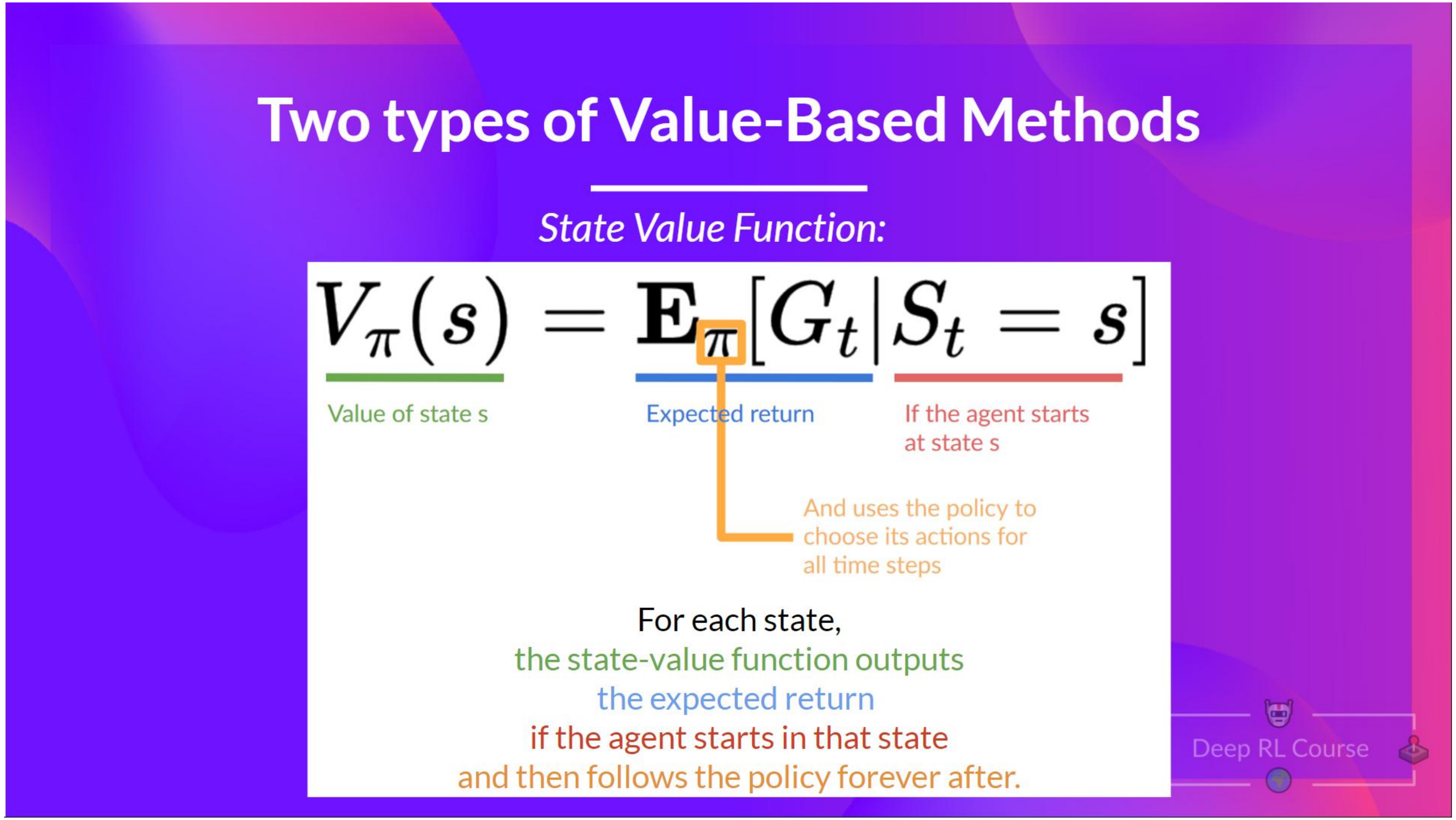
The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

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

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}$$

State-value function

Value Functions



Value Functions

If we are in state s, action a is taken, and thereafter policy π is followed,

Definition

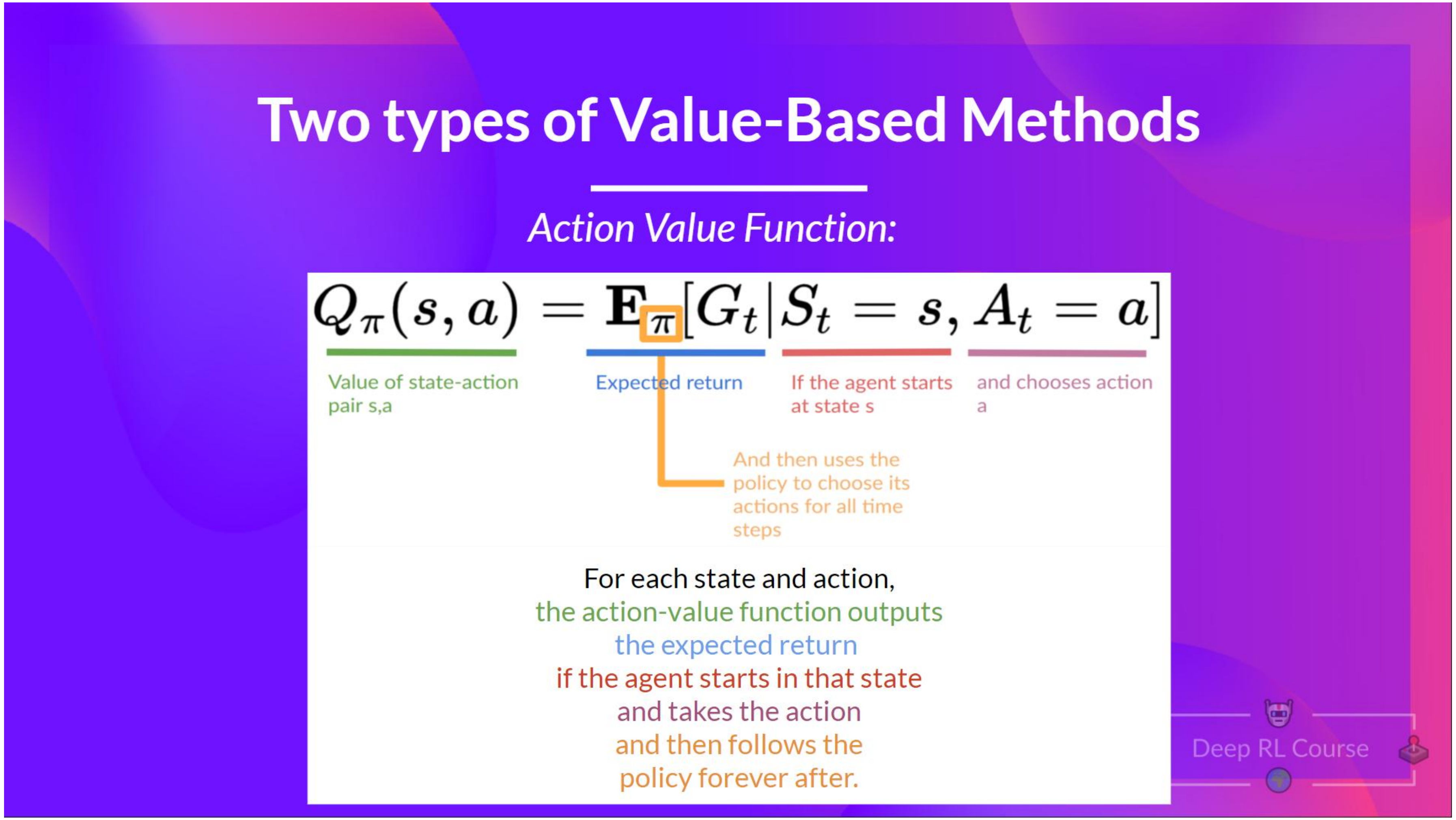
The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a\right]$$

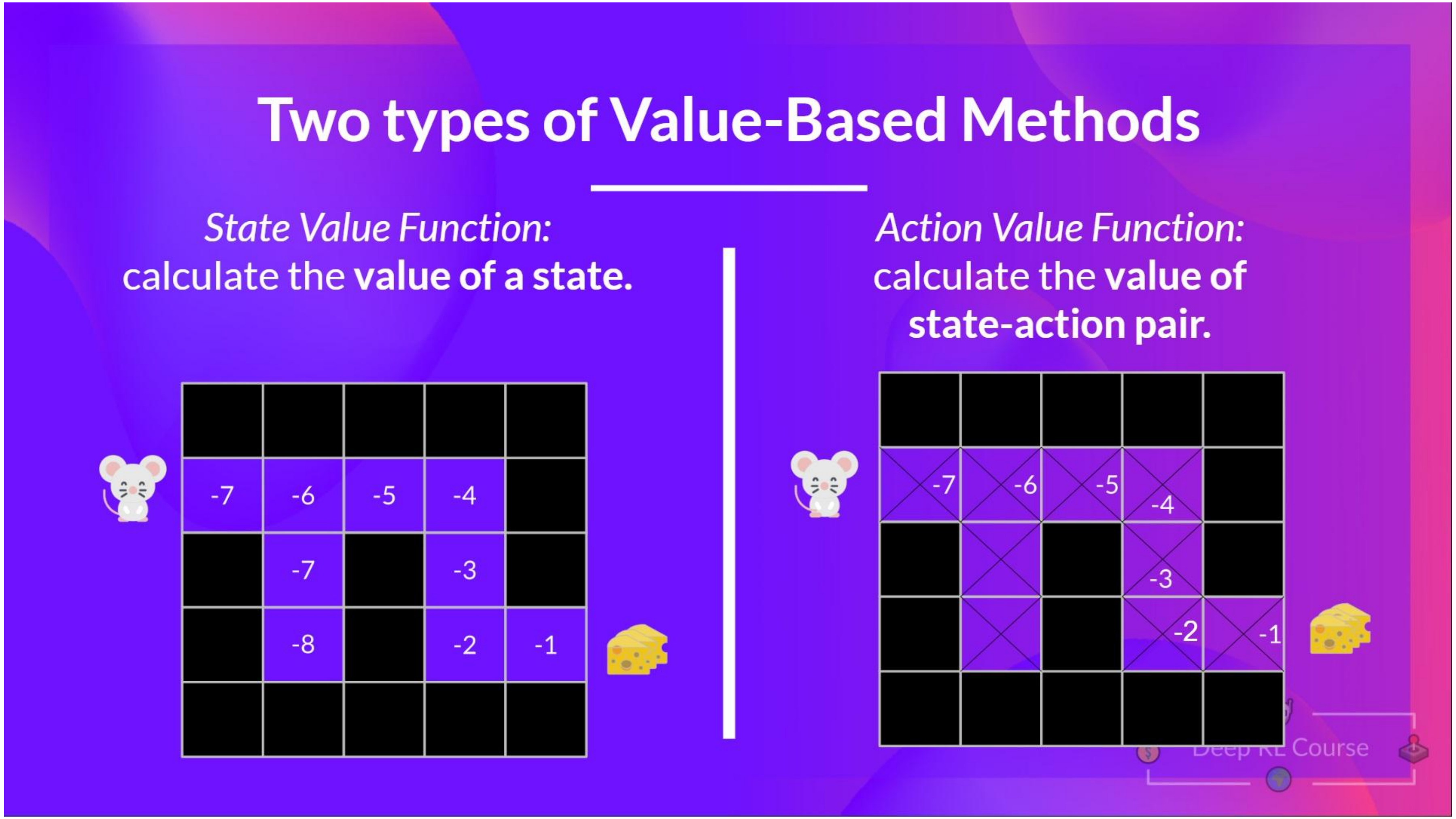
$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right]$$

Action-value function

Value Functions



To Recap …



Two types of Value-Based Methods
State Value Function:
calculate the value of a state.
Action Value Function:
calculate the value of
state-action pair.
-7
-6
-5
-4
-7
-3
-8
-2
-1
-7
-6
-5
-4
-3
-2
-1
Deep RL Course

Value Functions

Recursive Estimation of the Value Function:

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

Value Functions

Recursive Estimation of the Value Function:

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S},
\end{aligned}
$$

Bellman Equation

Expresses the relationship between the estimated state-value function at time t and t+1

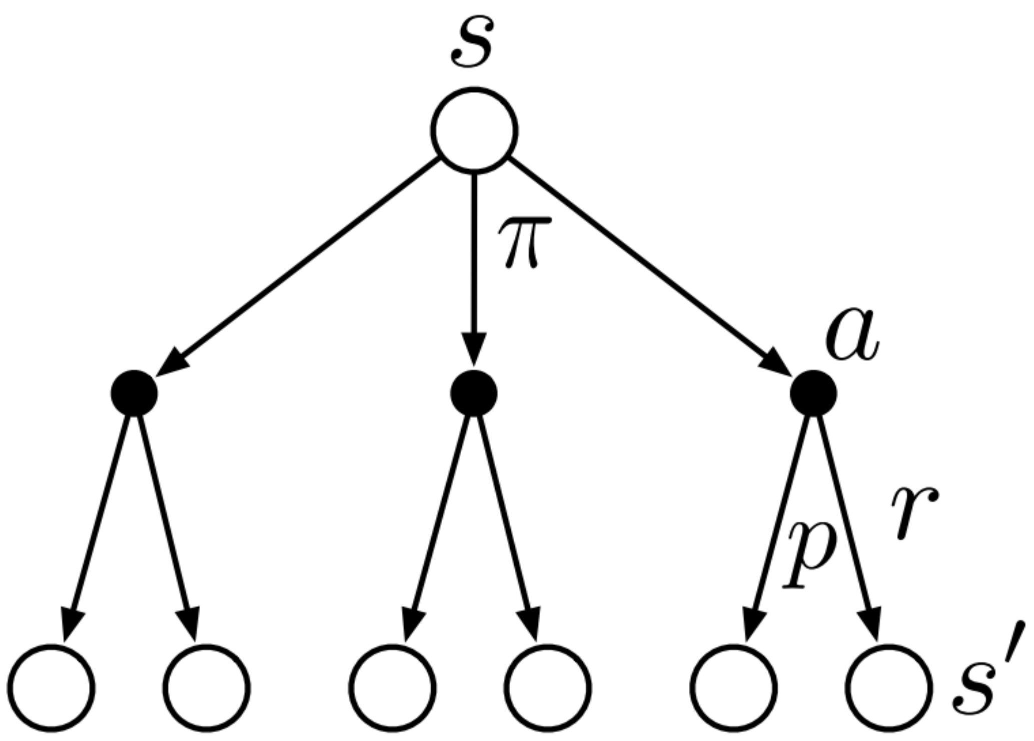
Unique solution to the Bellman equation: V_π

Description of the Bellman Equation

State:

State – Action:

State:



Backup diagram for v_π

Computing the Value Function from the Bellman Equation

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)\Big[r + \gamma v_\pi(s')\Big], \quad \text{for all } s \in \mathcal{S} \end{aligned}$$

Solving the Above System of Linear Equations

(*n equations with n unknowns* $V_\pi(s)$ *for all* $s \in S$)

Unique solution to the Bellman equation: V_π

Example

State: cell

Action: up-down-left-right

Reward:

Hitting the wall:

-1 ($s' = s$)

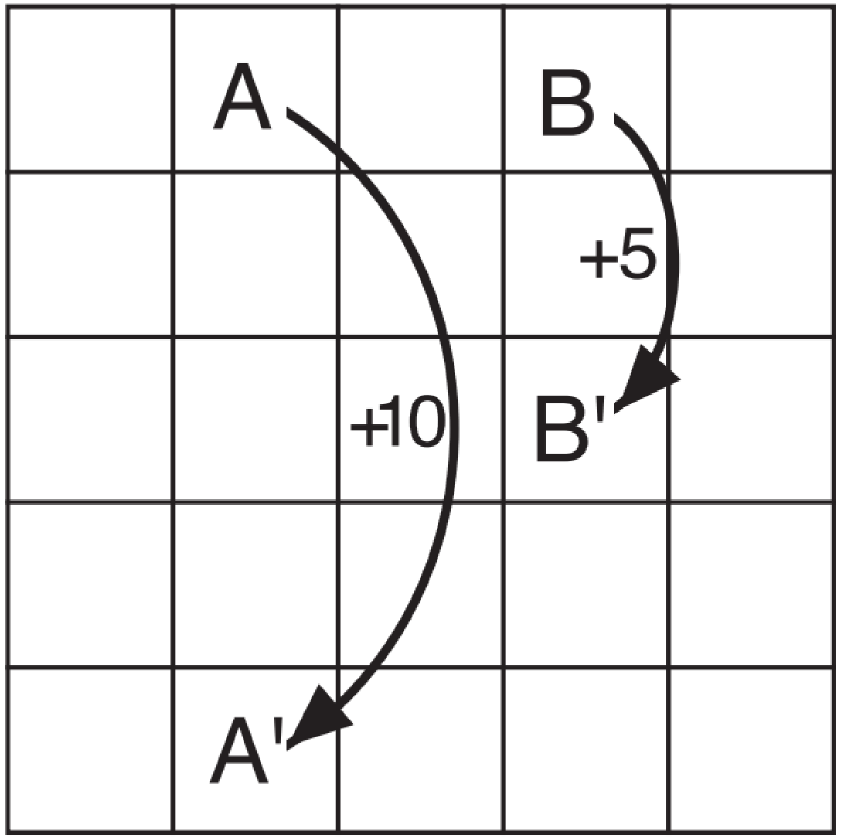
Exiting region A:

+10 (for any action) → transitions to A'

Exiting region B:

+5 (for any action) → transitions to B'

Others: 0



Example

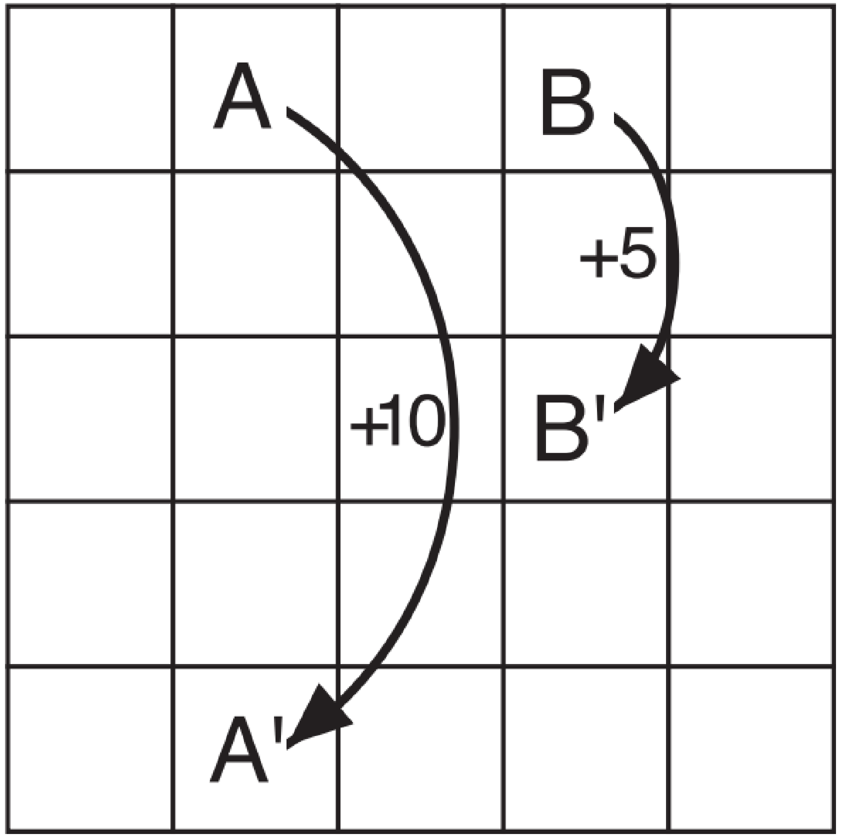
If we go only to A:

$$G_t = 10 + 10\gamma^5 + 10\gamma^{10} + \cdots = \frac{10}{1-\gamma^5}$$

If we go only to B:

$$G_t = 5 + 5\gamma^3 + 5\gamma^6 + \cdots = \frac{5}{1-\gamma^3}$$

$$\gamma = 0.9 \quad \rightarrow G_t = ?$$



Solving the Bellman Equation System for All Table Cells

(Determining $V_\pi(s)$ for each state)

Effect of γ on $V_\pi(s)$?

Function $p(s', r|s, a)$?

Function $\pi(a|s)$?

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

State-value function for the equiprobable random policy

Solving the Bellman Equation System for All Table Cells

(Determining $V_\pi(s)$ for each state)

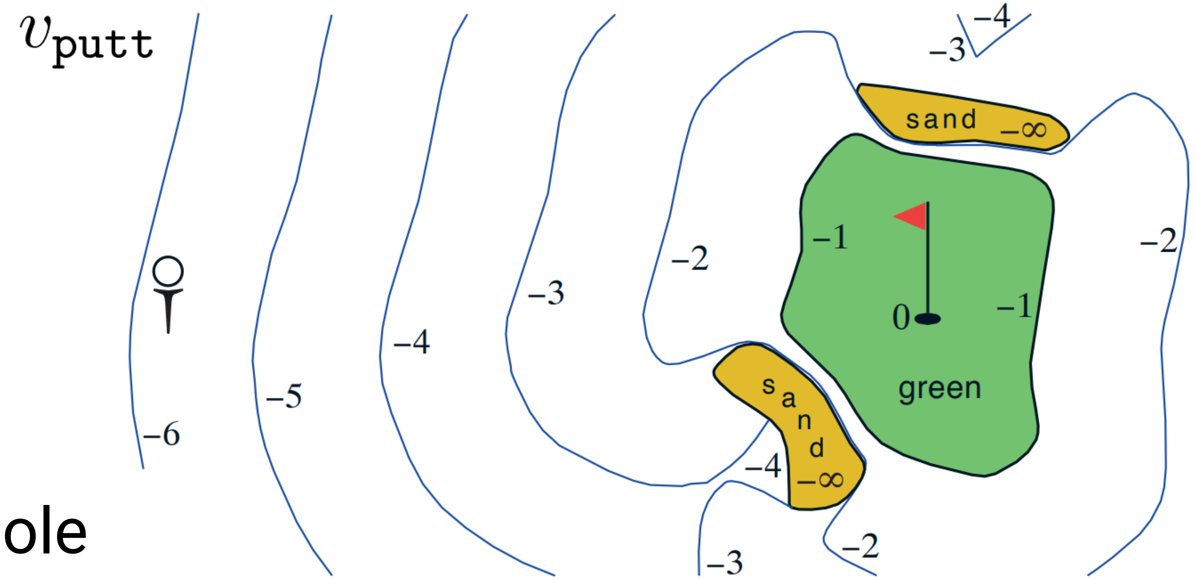
1. These numbers are the solution to the Bellman equation for a random policy with a uniform distribution.
2. The edge rows are negative due to collisions with the wall and the −1 penalty.
3. The value of A is less than 10 because it leads toward the wall.
4. The value of B is greater than 5 because it leads toward a value greater than 0.

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

State-value function for the equiprobable random policy

Example

State: Distance to the hole
Action: Driver – Putter
Reward: −1 for each stroke
Value for each state:
Negative number of strokes to the hole
Assumption:
Accurate and deterministic aiming, but limited shot distance



Optimal Policy

$$\pi \geq \pi'$$

$$v_\pi(s) \geq v_{\pi'}(s)$$

Definition

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

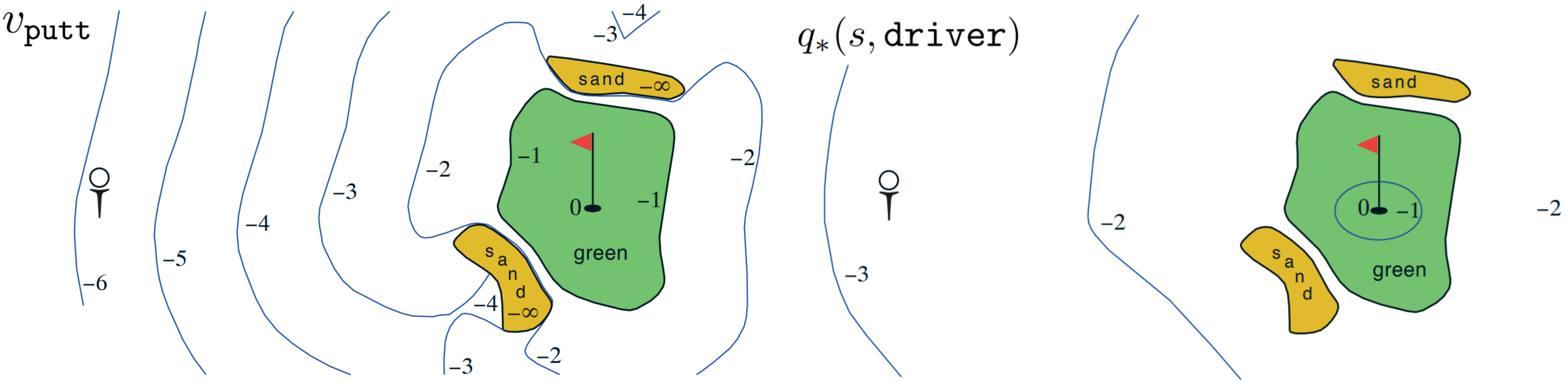
The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

We can write $q_*(s, a)$ in terms of $v_*(s)$ as follows:

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

Example



Assumption: Starting with the driver

Question: $q_*(s, driver) =?$

Optimal action for distant points: two drivers and one putter

Bellman Optimality Equation for State Value Function

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma v_*(s')\big].
\end{aligned}
$$

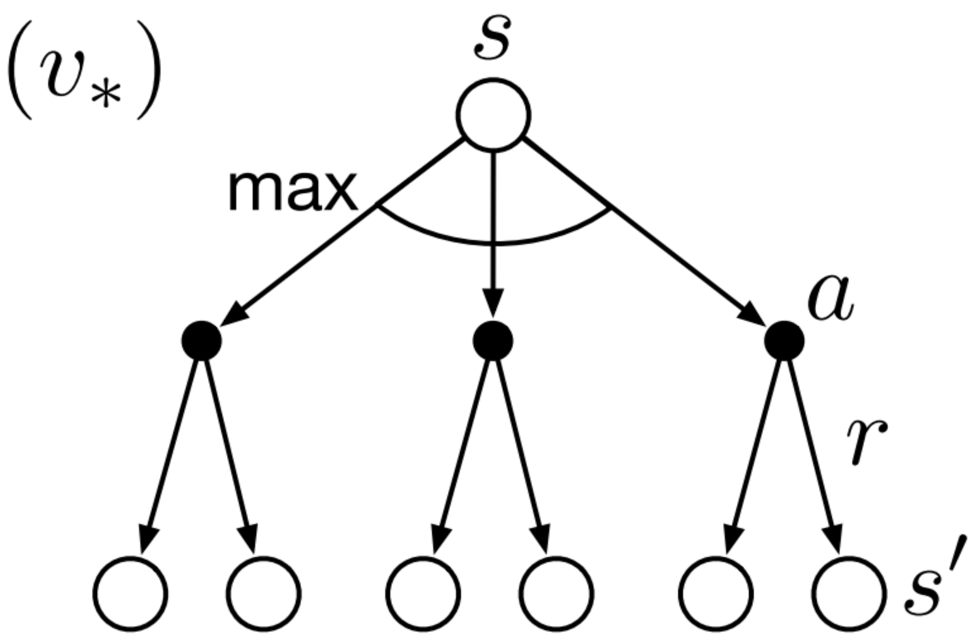
Solving the Above System of Linear Equations

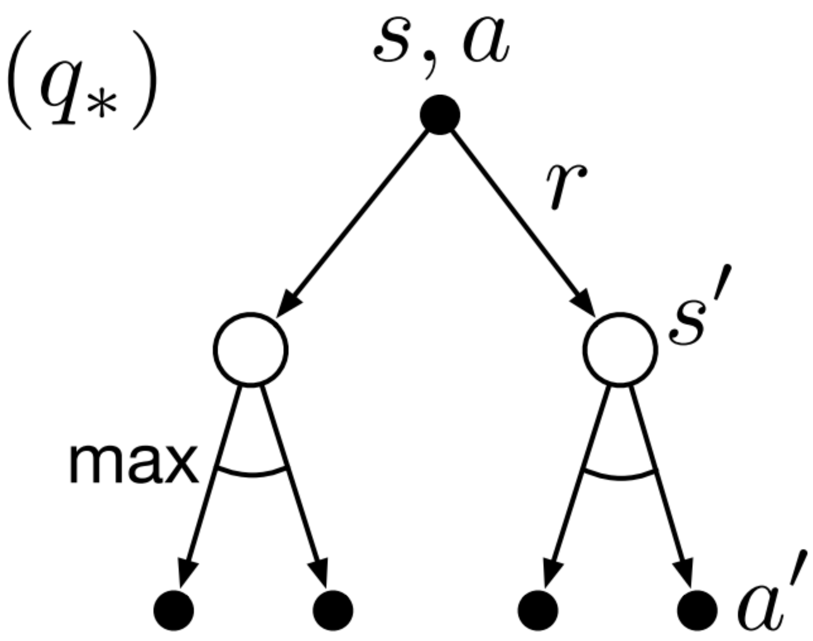
(*n equations with n unknowns* $V_\pi(s)$ *for all* $s \in S$)

Bellman Optimality Equation for Action Value Function

$$
\begin{aligned}
q_*(s, a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\
&= \sum_{s', r} p(s', r \,|\, s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big].
\end{aligned}
$$

Backup Diagrams for v_* and q_*





Optimal State Value Function

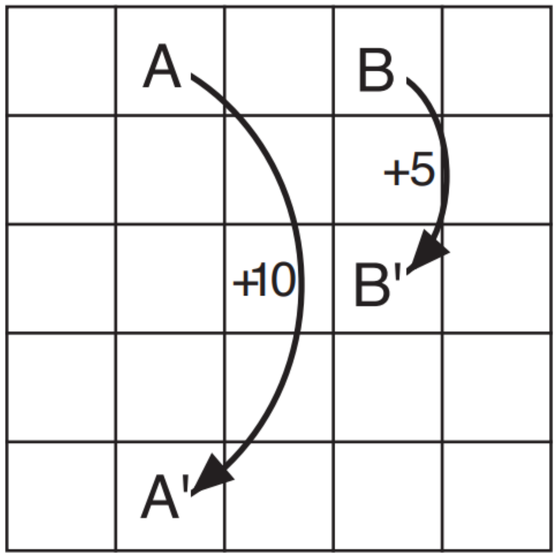
Determining the optimal policy is easily done from $V_*(s)$ (simple search).
Optimal policy: Any policy that is **greedy** with respect to $V_*(s)$.
Note: A policy selected greedily remains optimal in the long run.

Optimal Action Value Function

Determining the optimal policy from $q_*(s, a)$:
A policy that **maximizes** $q_*(s, a)$. (even simpler than before)!

Gridworld

Solving the Bellman Equation System for All Table Cells



Gridworld

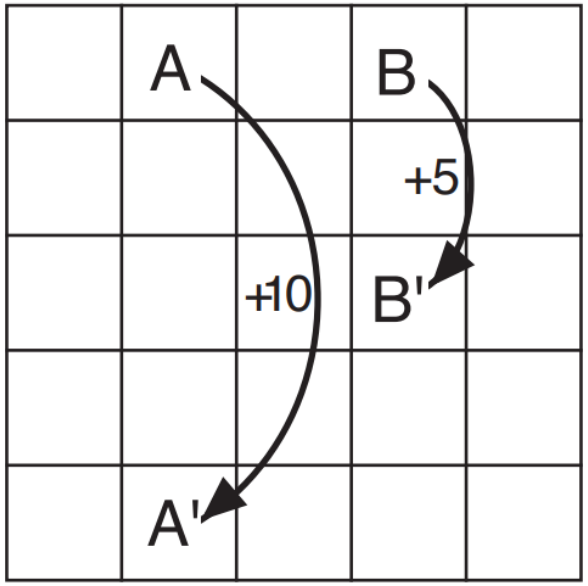
22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*

Solving the System of Bellman Optimality Equations (Optimal State-Value Function with 25 Nonlinear Equations)

Gridworld

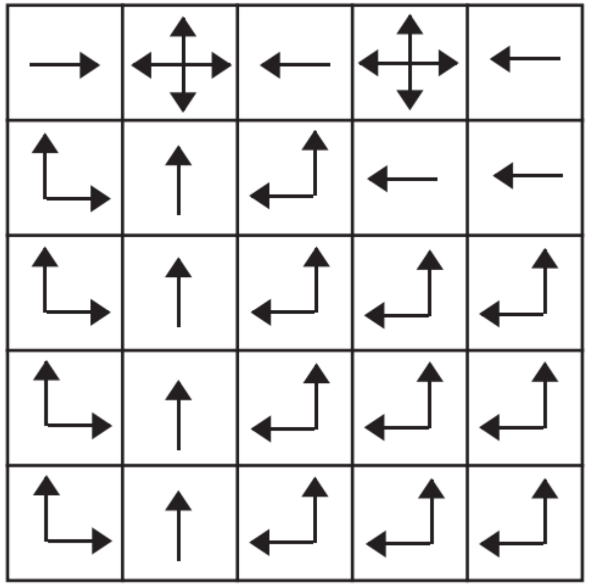
Determining $\pi_*(s)$ from $V_*(s)$:



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

Effect of Gamma (γ) on the Above Table?

Recycling Robot

Bellman Optimality Equation?

States: Low – High

→ Computing $V_*(s)$ by taking the **max** over the possible actions

Actions:

High → search & wait

Low → search & wait & recharge

S=High:

Action	S'	p	r
Search	High	α	r_s
Search	Low	$1-\alpha$	r_s
Wait	High	1	r_w

S=High:

Action	S'	p	r
Search	High	α	r_s
Search	Low	$1-\alpha$	r_s
Wait	High	1	r_w

Recycling Robot

Bellman Optimality Equation?

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right]$$

$$\begin{aligned}
v_*(\mathtt{h}) &= \max\left\{\begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l}) + \gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l}) + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} \alpha[r_\mathtt{s} + \gamma v_*(\mathtt{h})] + (1-\alpha)[r_\mathtt{s} + \gamma v_*(\mathtt{l})], \\ 1[r_\mathtt{w} + \gamma v_*(\mathtt{h})] + 0[r_\mathtt{w} + \gamma v_*(\mathtt{l})] \end{array}\right\} \\
&= \max\left\{\begin{array}{l} r_\mathtt{s} + \gamma[\alpha v_*(\mathtt{h}) + (1-\alpha)v_*(\mathtt{l})], \\ r_\mathtt{w} + \gamma v_*(\mathtt{h}) \end{array}\right\}.
\end{aligned}$$

For any choice of $r_\mathtt{s}$, $r_\mathtt{w}$, α, β, and γ, with $0 \le \gamma < 1$, $0 \le \alpha, \beta \le 1$

We can calculate $V_*(high)$

Recycling Robot

Bellman Optimality Equation?

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right]$$

S=Low:

Action	S'	p	r
Search	Low	β	r_s
Search	High	$1-\beta$	r_s
Wait	Low	1	r_w
Recharge	High	1	0

$$v_*(\mathtt{l}) = \max \left\{ \begin{array}{l} \beta r_{\mathtt{s}} - 3(1-\beta) + \gamma[(1-\beta)v_*(\mathtt{h}) + \beta v_*(\mathtt{l})], \\ r_{\mathtt{w}} + \gamma v_*(\mathtt{l}), \\ \gamma v_*(\mathtt{h}) \end{array} \right\}$$

Golf

Bellman Optimality Equation?

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma v_*(s')\big]$$

$$V_*(s) = \max \left[\sum_{r,s'} P\left(s', -1 \mid s,\ \text{Putter}\right)\left[-1 + \gamma V_*\left(s'\right)\right] \right.$$
$$\left. , \sum_{r,s'} P\left(s', -1 \mid s,\ \text{driver}\right)\left[-1 + \gamma V_*\left(s'\right)\right] \right]$$

To Recap …

Solving the Bellman Optimality Problem ⇔ Reinforcement Learning Solution

Requirements:

- Knowing environment dynamics: *p*
- Computational load!
- Markov property of the problem

Challenge:
Calculating V_* , q_* for states (e.g., 10^{20} in backgammon)

Solution: Approximation
Dynamic Programming