

# Introduction to Reinforcement Learning

May 8

[Tailin Wu](#), Westlake University

Website: [ai4s.lab.westlake.edu.cn/course](http://ai4s.lab.westlake.edu.cn/course)



Image from: DeepMind

Some materials are from [Shiyu Zhao](#)

# Class 2: Deep learning fundamentals

## 1. **Principle 1:** Model a hard transformation by **composing simple** transformations:

- Multilayer Perceptron (MLP)
- Backpropagation

## 2. **Principle 2:** Directly optimizing the final objective using **maximum likelihood** and **information theory**:

- Maximum likelihood: MSE, uncertainty estimation
- Information: cross-entropy, Information Bottleneck

## 3. Optimization

- Adam: combining **momentum** and **per-dimension magnitude**
- SAM (sharpness-aware minimization):  $\max_{\epsilon \in N_{\theta}} \ell(\theta + \epsilon)$  finds flat and robust minima
- Federative learning: improves the data privacy by only sharing client models

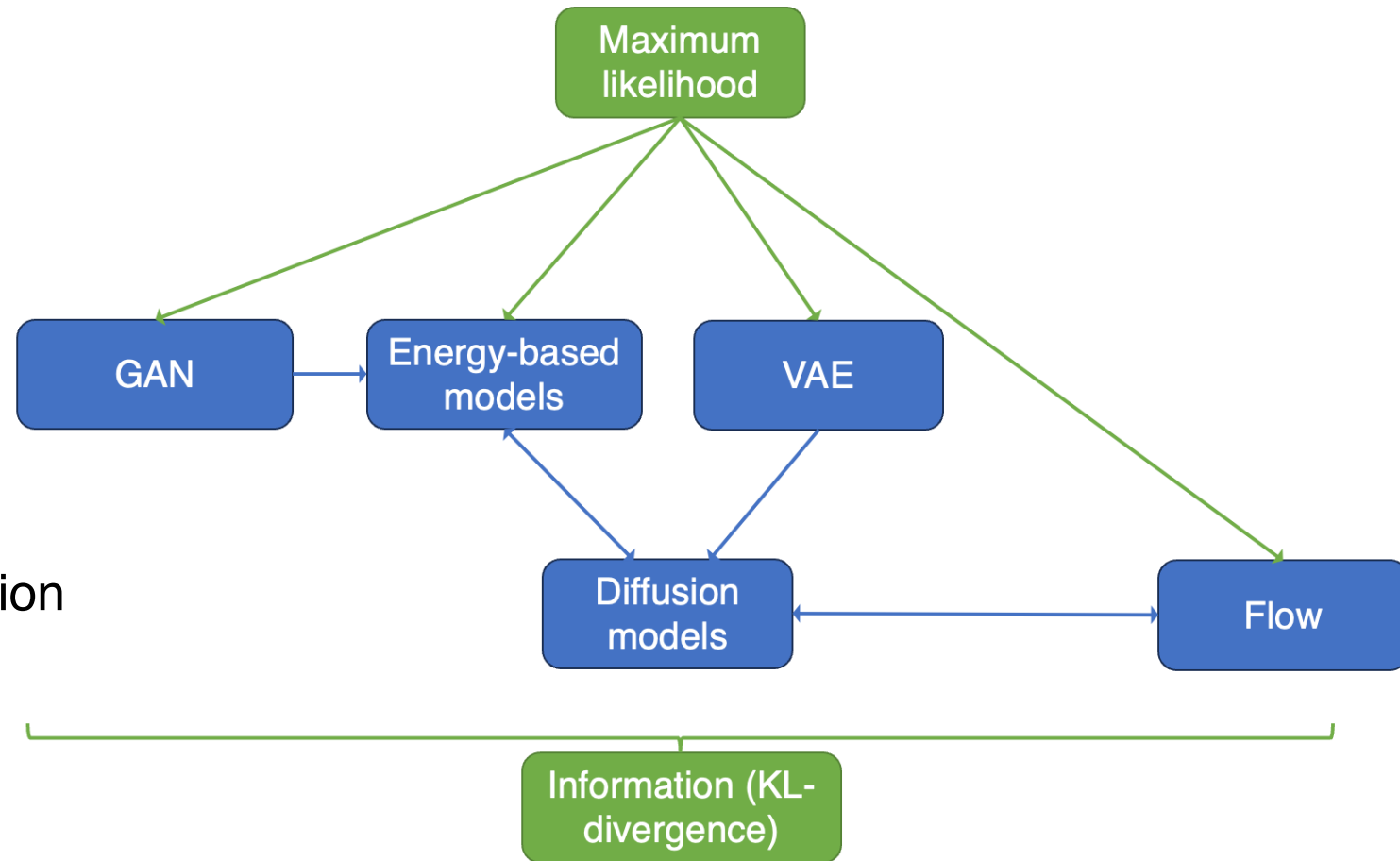
# Class 3: Frontiers in Generative modeling

- Generative models

- VAE
- GAN
- Energy-based models
- Diffusion models
- Flows

- Application of diffusion models

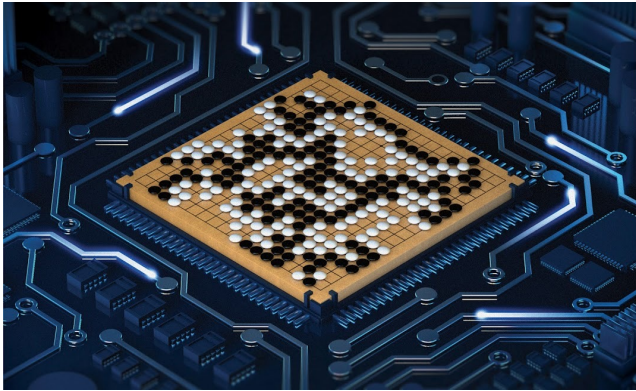
- Image, video, and shape generation
- Simulation
- Inverse design/inverse problem
- Control/planning



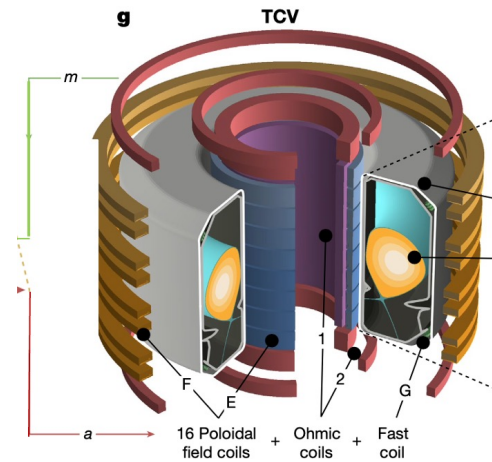
# Class 4: Graph Neural Networks

- Graph representation
- Tasks
- Graph Neural Networks: Fundamentals
- Useful Techniques
  - Aggregation
  - GCN
  - GraphSage
  - GAT
- Advanced topics
  - Neural subgraph matching
  - Graph Transformers

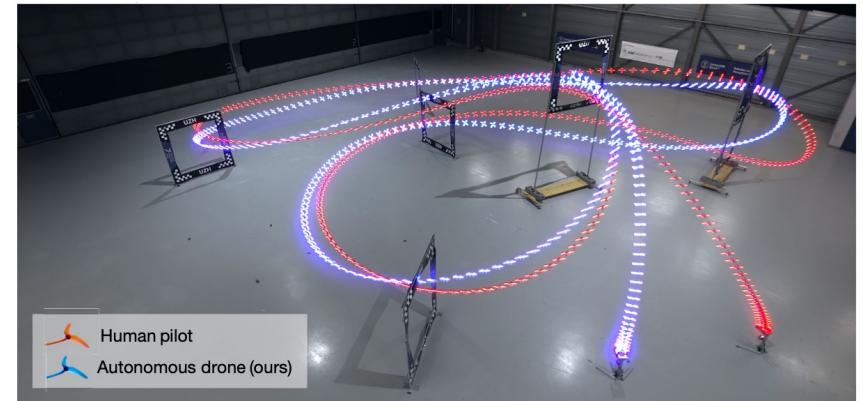
# Reinforcement learning



AlphaGo [1]



Controlled nuclear fusion [2]



Drone racing [3]

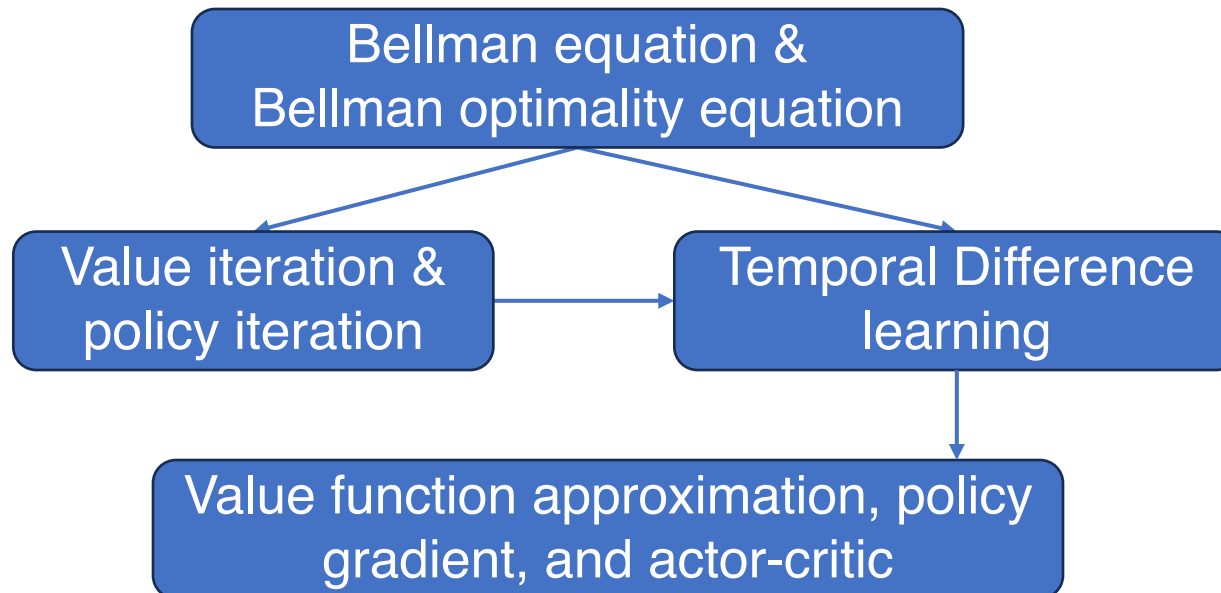
[1] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.

[2] Degraeve, Jonas, et al. "Magnetic control of tokamak plasmas through deep reinforcement learning." *Nature* 602.7897 (2022): 414-419.

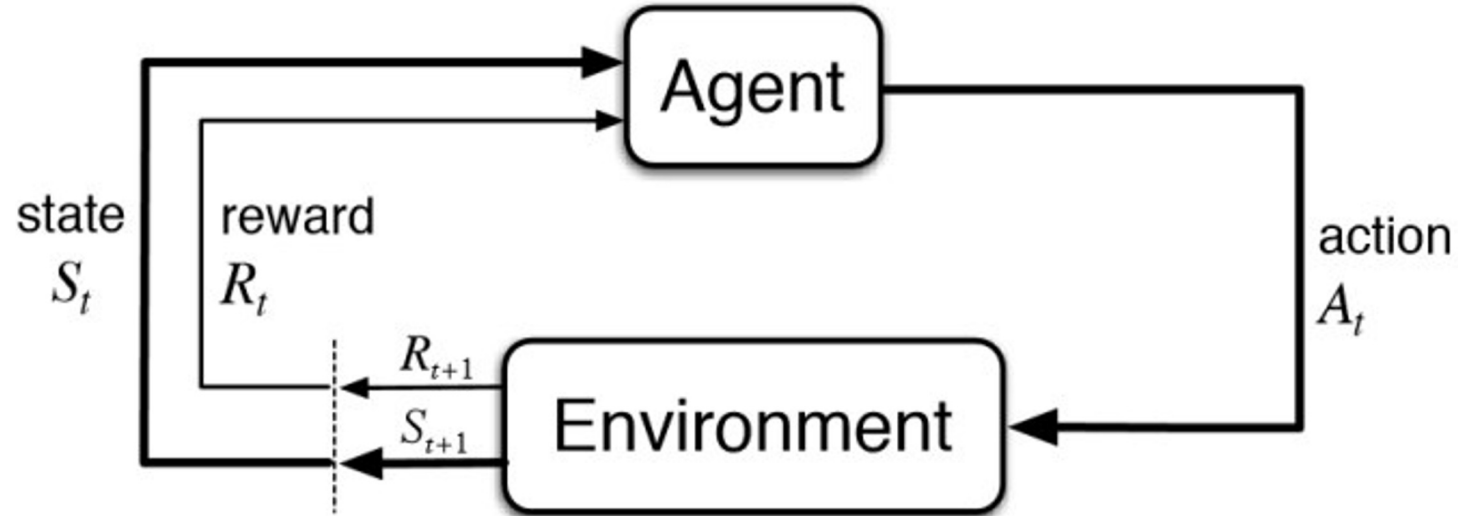
[3] Kaufmann, Elia, et al. "Champion-level drone racing using deep reinforcement learning." *Nature* 620.7976 (2023): 982-987.

# Outline

- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- Value function approximation, policy gradient, and actor-critic



# Markov Decision Process (MDP): Setup



**Goal:** maximize the long-term expected reward w.r.t. to the policy  $\pi(A_t|S_t)$

$$\max_{\pi(A_t|S_t)} \mathbb{E}_t[R_t]$$

# Markov Decision Process (MDP): State and action

**State:**  $\{s^{(1)}, s^{(2)}, \dots, s^{(9)}\}$

**Action:**

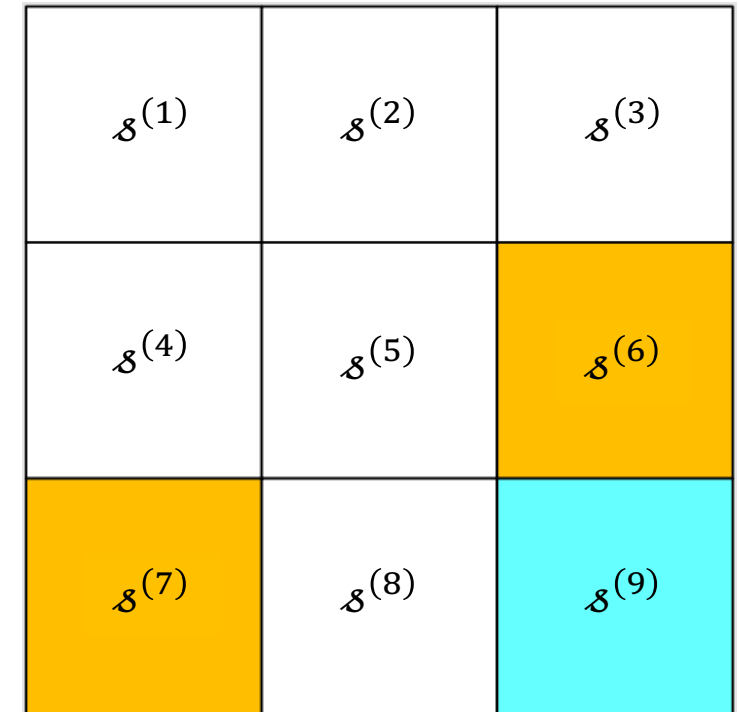
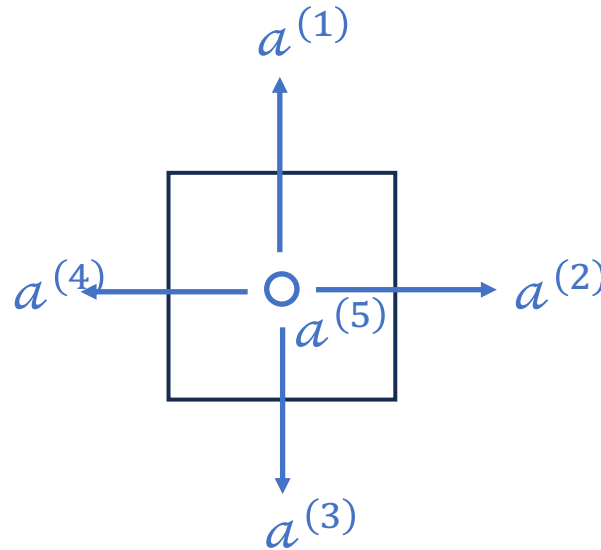
$a^{(1)}$ : move upward

$a^{(2)}$ : move rightward

$a^{(3)}$ : move downward

$a^{(4)}$ : move leftward

$a^{(5)}$ : stay unchanged



Grid world

**Action space of a state:** the set of all possible actions of a state.  $\mathcal{A}(s^{(i)}) = \{a^{(i)}\}_{i=1}^5$ .

**State transition probability:** e.g.,

$$p(s^{(2)} | s^{(1)}, a^{(2)}) = 1.$$

$$p(s^{(i)} | s^{(1)}, a^{(2)}) = 0, \quad \forall i \neq 2$$

\*We use subscript  $t$  in  $s_t, a_t$  to denote step  $t$ , and use superscript in  $s^{(i)}, a^{(i)}$  to denote different choices of state or action. 8

# Markov Decision Process (MDP): Policy

**Policy**  $\pi(a|s)$ :

E.g., for  $s = s^{(1)}$ ,

$$\pi(a = a^{(1)} | s = s^{(1)}) = 0$$

$$\pi(a = a^{(2)} | s = s^{(1)}) = 0.5$$

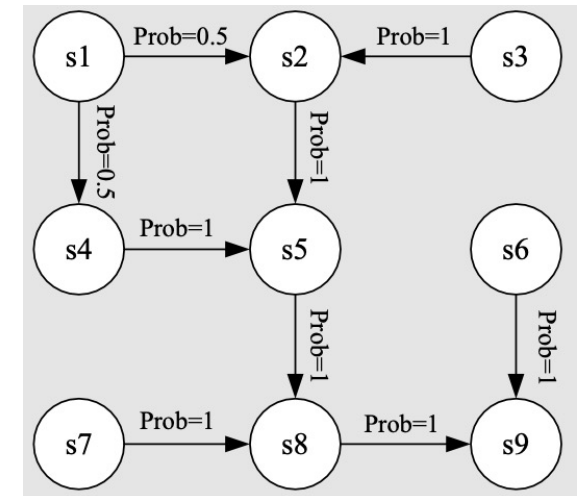
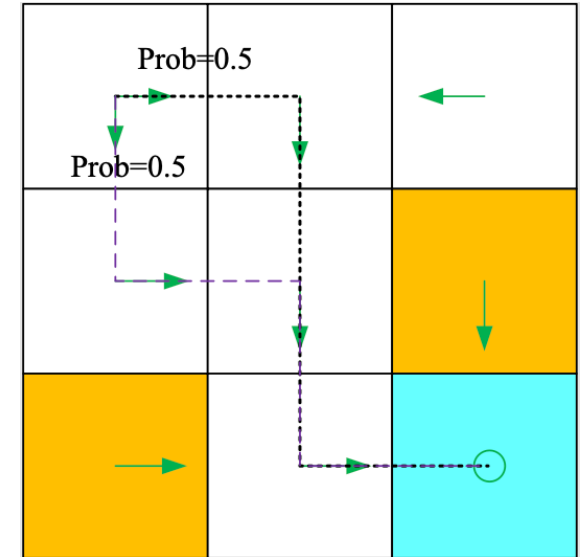
$$\pi(a = a^{(3)} | s = s^{(1)}) = 0.5$$

$$\pi(a = a^{(4)} | s = s^{(1)}) = 0$$

$$\pi(a = a^{(5)} | s = s^{(1)}) = 0$$

Tabular representation:

	$a_1$ (upward)	$a_2$ (rightward)	$a_3$ (downward)	$a_4$ (leftward)	$a_5$ (unchanged)
$s_1$	0	0.5	0.5	0	0
$s_2$	0	0	1	0	0
$s_3$	0	0	0	1	0
$s_4$	0	1	0	0	0
$s_5$	0	0	1	0	0
$s_6$	0	0	1	0	0
$s_7$	0	1	0	0	0
$s_8$	0	1	0	0	0
$s_9$	0	0	0	0	1



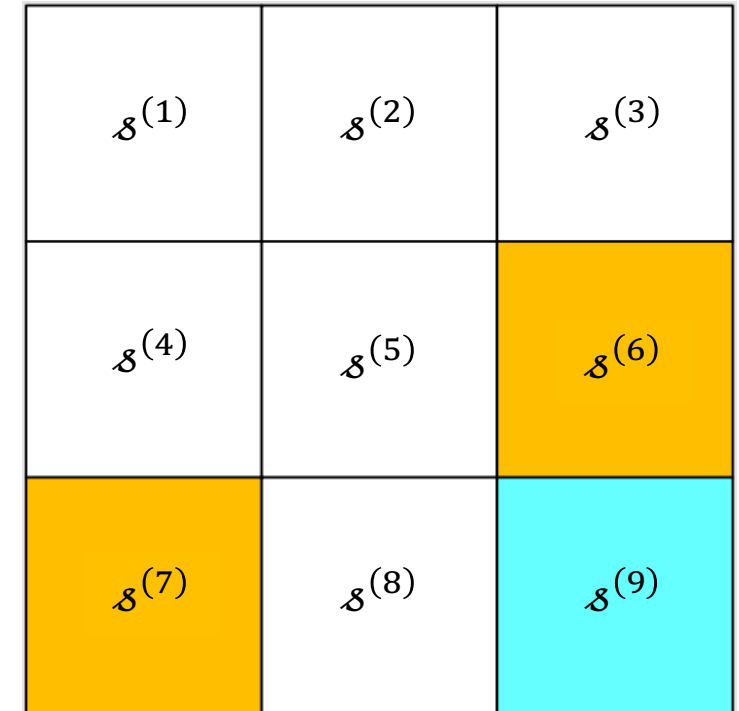
# Markov Decision Process (MDP): Reward

## Reward:

- If the agent attempts to get out of the boundary,  $r_{bound} = -1$
- If the agent attempts to enter a forbidden cell,  $r_{forbid} = -1$
- If the agent reaches the target cell,  $r_{target} = +1$
- Otherwise, the agent gets a reward of  $r = 0$ .

## Tabular representation:

	$a_1$ (upward)	$a_2$ (rightward)	$a_3$ (downward)	$a_4$ (leftward)	$a_5$ (unchanged)
$s_1$	$r_{bound}$	0	0	$r_{bound}$	0
$s_2$	$r_{bound}$	0	0	0	0
$s_3$	$r_{bound}$	$r_{bound}$	$r_{forbid}$	0	0
$s_4$	0	0	$r_{forbid}$	$r_{bound}$	0
$s_5$	0	$r_{forbid}$	0	0	0
$s_6$	0	$r_{bound}$	$r_{target}$	0	$r_{forbid}$
$s_7$	0	0	$r_{bound}$	$r_{bound}$	$r_{forbid}$
$s_8$	0	$r_{target}$	$r_{bound}$	$r_{forbid}$	0
$s_9$	$r_{forbid}$	$r_{bound}$	$r_{bound}$	0	$r_{target}$



Grid world

# Markov Decision Process (MDP): Return

**Trajectory:** a state-action-reward chain

$$s^{(1)} \xrightarrow[r=0]{a^{(2)}} s^{(2)} \xrightarrow[r=0]{a^{(3)}} s^{(5)} \xrightarrow[r=0]{a^{(3)}} s^{(8)} \xrightarrow[r=1]{a^{(2)}} s^{(9)}$$

**Return:** the sum of all the rewards collected along the trajectory:

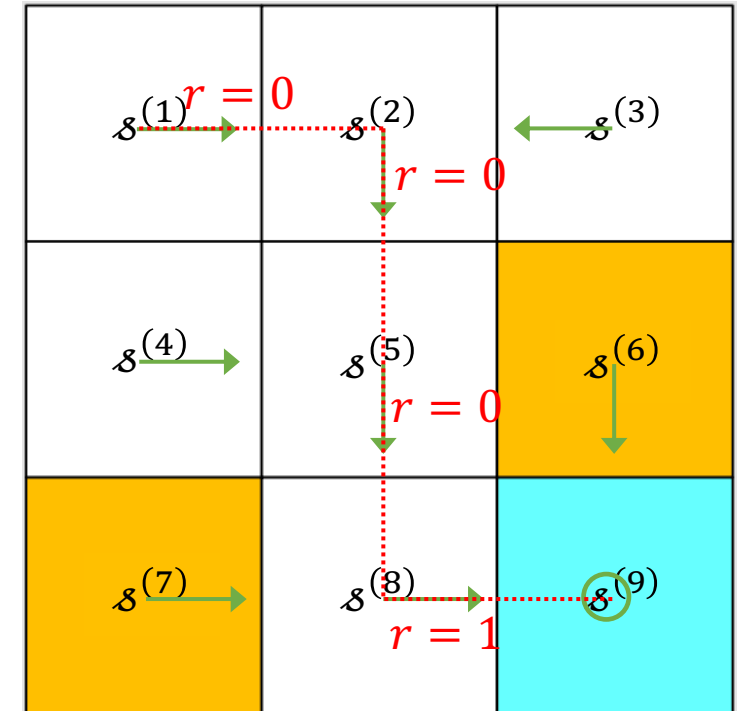
$$\text{return} = 0 + 0 + 0 + 1 = 1$$

The trajectory can be infinite,

$$s^{(1)} \xrightarrow[r=0]{a^{(2)}} s^{(2)} \xrightarrow[r=0]{a^{(3)}} s^{(5)} \xrightarrow[r=0]{a^{(3)}} s^{(8)} \xrightarrow[r=1]{a^{(2)}} s^{(9)} \xrightarrow[r=1]{a^{(5)}} s^{(9)} \xrightarrow[r=1]{a^{(5)}} s^{(9)} \dots$$

**Discounted return:** given discounted rate  $\gamma$ ,

$$\text{discounted return} = 0 + \gamma 0 + \gamma^2 0 + \gamma^3 1 + \gamma^4 1 \dots = \frac{\gamma^3}{1-\gamma}$$



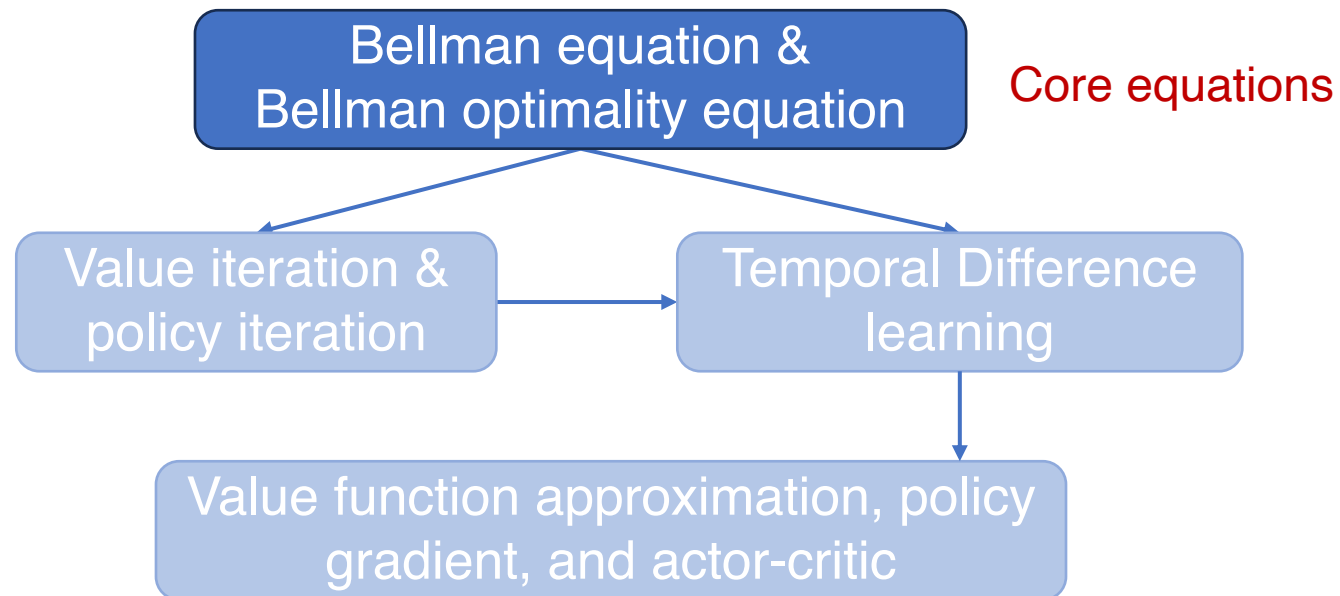
Trajectory

# Markov Decision Process (MDP): Full components

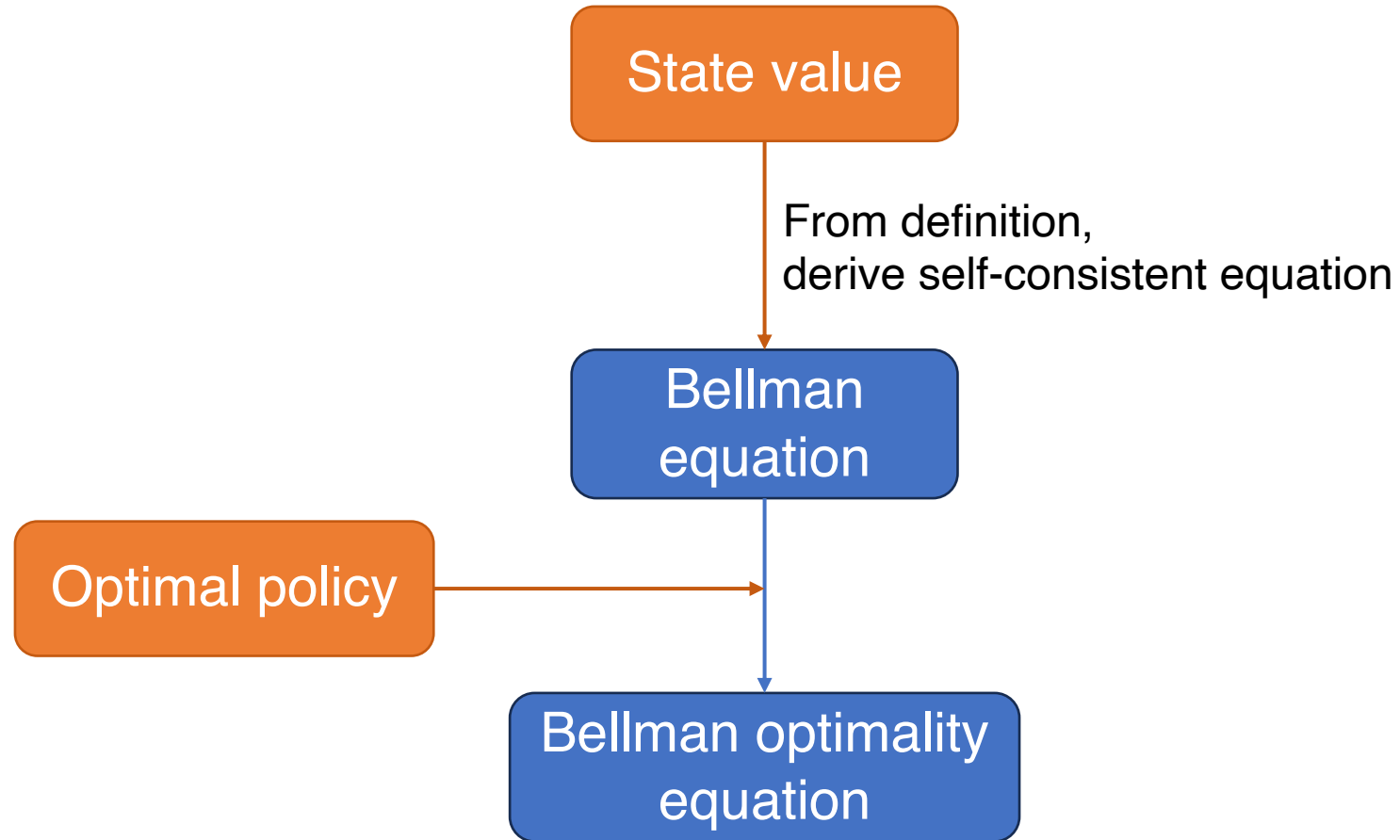
- **Sets:**
  - **State:** the set of states  $\mathcal{S}$
  - **Action:** the set of actions  $\mathcal{A}(s)$  is associated for state  $s \in \mathcal{S}$ .
  - **Reward:** the set of rewards  $\mathcal{R}(s, a)$ .
- **Probability distribution** (or called **system model**):
  - **State transition probability:** at state  $s$ , taking action  $a$ , the probability to transit to state  $s'$  is  $p(s'|s, a)$
  - **Reward probability:** at state  $s$ , taking action  $a$ , the probability to get reward  $r$  is  $p(r|s, a)$
- **Policy:** at state  $s$ , the probability to choose action  $a$  is  $\pi(a|s)$
- **Markov property:** memoryless property
  - $p(s_{t+1}|a_t, s_t, \dots, a_0, s_0) = p(s_{t+1}|a_t, s_t)$
  - $p(r_{t+1}|a_t, s_t, \dots, a_0, s_0) = p(r_{t+1}|a_t, s_t)$ .

# Outline

- Markov Decision Process (MDP) setup
- **Bellman equation and Bellman optimality equation**
- Value iteration and policy iteration
- Temporal difference learning
- Value function approximation, policy gradient methods, and actor-critic



# How we approach it



# State value

**State value:** Suppose that we start with a state  $s$  and follow the policy  $\pi(a|s)$ , state value is the **expected discounted return**

$$v_{\pi}(s) := \mathbb{E}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

The state value function evaluates how good the policy  $\pi$  is.

# Bellman Equation: Derivation from state value definition

$$\begin{aligned}v_{\pi}(s) &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} \dots) | S_t = s] \\ &= \underbrace{\mathbb{E}[R_{t+1} | S_t = s]}_{\text{expected immediate reward}} + \gamma \underbrace{\mathbb{E}[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} \dots | S_t = s]}_{\text{remaining future reward}}\end{aligned}$$

Here

$$\mathbb{E}[R_{t+1} | S_t = s] := r_{\pi}(s) = \sum_a \pi(a|s) \sum_r p(r|s, a)r$$

# Bellman Equation: Derivation

$$\begin{aligned}v_{\pi}(s) &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\&= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} \dots) | S_t = s] \\&= \mathbb{E}[R_{t+1} | S_t = s] + \gamma \mathbb{E}[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} \dots | S_t = s]\end{aligned}$$

$$\begin{aligned}\mathbb{E}[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} \dots | S_t = s] \\&= \sum_{s'} p_{\pi}(s'|s) \mathbb{E}[G_{t+1} | S_t = s, S_{t+1} = s'] \\&= \sum_{s'} p_{\pi}(s'|s) \mathbb{E}[G_{t+1} | S_{t+1} = s'] \\&= \sum_{s'} p_{\pi}(s'|s) v_{\pi}(s')\end{aligned}$$

Here

$$p_{\pi}(s'|s) := \sum_a \pi(a|s) p(s'|s, a)$$

is the transition probability under policy  $\pi$ .

# Bellman Equation

$s$ : state  
 $a$ : action  
 $r$ : reward

$$v_{\pi}(s) = r_{\pi}(s) + \gamma \sum_{s'} p_{\pi}(s'|s) v_{\pi}(s')$$

Expanded form:

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

Matrix form:

$$v_{\pi} = r_{\pi} + \gamma P_{\pi} v_{\pi}$$

$$v_{\pi} = [v_{\pi}(s_1), \dots, v_{\pi}(s_n)]^T \in R^n$$

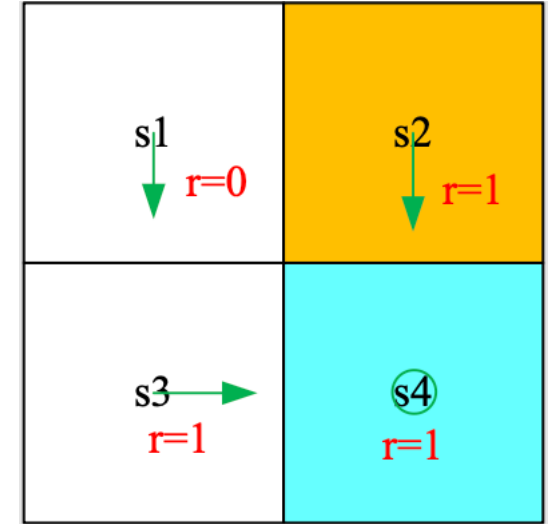
$$r_{\pi} = [r_{\pi}(s_1), \dots, r_{\pi}(s_n)]^T \in R^n$$

$P_{\pi} \in R^{n \times n}$ ,  $[P_{\pi}]_{ij} = p_{\pi}(s_j | s_i)$  is the state transition matrix

# Bellman Equation: Example

Bellman equation:

$$\underbrace{\begin{bmatrix} v_{\pi}(\mathcal{s}^{(1)}) \\ v_{\pi}(\mathcal{s}^{(2)}) \\ v_{\pi}(\mathcal{s}^{(3)}) \\ v_{\pi}(\mathcal{s}^{(4)}) \end{bmatrix}}_{v_{\pi}} = \underbrace{\begin{bmatrix} r_{\pi}(\mathcal{s}^{(1)}) \\ r_{\pi}(\mathcal{s}^{(2)}) \\ r_{\pi}(\mathcal{s}^{(3)}) \\ r_{\pi}(\mathcal{s}^{(4)}) \end{bmatrix}}_{r_{\pi}} + \gamma \underbrace{\begin{bmatrix} p_{\pi}(\mathcal{s}^{(1)}|\mathcal{s}^{(1)}) & p_{\pi}(\mathcal{s}^{(2)}|\mathcal{s}^{(1)}) & p_{\pi}(\mathcal{s}^{(3)}|\mathcal{s}^{(1)}) & p_{\pi}(\mathcal{s}^{(4)}|\mathcal{s}^{(1)}) \\ p_{\pi}(\mathcal{s}^{(1)}|\mathcal{s}^{(2)}) & p_{\pi}(\mathcal{s}^{(2)}|\mathcal{s}^{(2)}) & p_{\pi}(\mathcal{s}^{(3)}|\mathcal{s}^{(2)}) & p_{\pi}(\mathcal{s}^{(4)}|\mathcal{s}^{(2)}) \\ p_{\pi}(\mathcal{s}^{(1)}|\mathcal{s}^{(3)}) & p_{\pi}(\mathcal{s}^{(2)}|\mathcal{s}^{(3)}) & p_{\pi}(\mathcal{s}^{(3)}|\mathcal{s}^{(3)}) & p_{\pi}(\mathcal{s}^{(4)}|\mathcal{s}^{(3)}) \\ p_{\pi}(\mathcal{s}^{(1)}|\mathcal{s}^{(4)}) & p_{\pi}(\mathcal{s}^{(2)}|\mathcal{s}^{(4)}) & p_{\pi}(\mathcal{s}^{(3)}|\mathcal{s}^{(4)}) & p_{\pi}(\mathcal{s}^{(4)}|\mathcal{s}^{(4)}) \end{bmatrix}}_{P_{\pi}} \underbrace{\begin{bmatrix} v_{\pi}(\mathcal{s}^{(1)}) \\ v_{\pi}(\mathcal{s}^{(2)}) \\ v_{\pi}(\mathcal{s}^{(3)}) \\ v_{\pi}(\mathcal{s}^{(4)}) \end{bmatrix}}_{v_{\pi}}$$



2x2 grid world

For the specific policy in this grid world:

$$\begin{bmatrix} v_{\pi}(\mathcal{s}^{(1)}) \\ v_{\pi}(\mathcal{s}^{(2)}) \\ v_{\pi}(\mathcal{s}^{(3)}) \\ v_{\pi}(\mathcal{s}^{(4)}) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \gamma \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{\pi}(\mathcal{s}^{(1)}) \\ v_{\pi}(\mathcal{s}^{(2)}) \\ v_{\pi}(\mathcal{s}^{(3)}) \\ v_{\pi}(\mathcal{s}^{(4)}) \end{bmatrix}$$

# Bellman Equation: Solution

Bellman Equation:

$$v_{\pi} = r_{\pi} + \gamma P_{\pi} v_{\pi}$$

Solution:

$$v_{\pi} = (1 - \gamma P_{\pi})^{-1} r_{\pi}$$

Iterative solution:

$$v_{k+1} = r_{\pi} + \gamma P_{\pi} v_k, k = 1, 2, \dots$$

It can be proved that when  $k \rightarrow +\infty$ ,  $v_k \rightarrow v_{\pi}$ .

# Action value: Definition

$$q_{\pi}(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

The **average return** the agent can get starting from a state  $s$  and taking an action  $a$ , and then following the policy  $\pi$ .

# Action value: Relation with state value

$s$ : state  
 $a$ : action  
 $r$ : reward

Bellman equation:

$$v_{\pi}(s) = \sum_a \pi(a|s) \underbrace{\left( \sum_r p(r|s, a)r + \gamma \sum_{s'} p(s'|s, a)v_{\pi}(s') \right)}_{q_{\pi}(s, a)}$$

So we have:

$$v_{\pi}(s) = \sum_a \pi(a|s) q_{\pi}(s, a)$$

action value  $\Rightarrow$  state value

$$q_{\pi}(s, a) = \sum_r p(r|s, a)r + \gamma \sum_{s'} p(s'|s, a)v_{\pi}(s')$$

state value  $\Rightarrow$  action value

# Optimal policy: Definition

A policy  $\pi^*$  is **optimal** if  $v_{\pi^*}(s) \geq v_{\pi}(s)$  for **all** states  $s$  and for **any other** policy  $\pi$ .

# Bellman Optimality Equation

Recall Bellman Equation:

$$v_{\pi} = r_{\pi} + \gamma P_{\pi} v_{\pi}$$

Here the maximization of  $\pi(a|s)$  is performed elementwise:

Bellman Optimality Equation:

$$v_{\pi} = \max_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi})$$

	$a_1$ (upward)	$a_2$ (rightward)	$a_3$ (downward)	$a_4$ (leftward)	$a_5$ (unchanged)
$s_1$	0	0.5	0.5	0	0
$s_2$	0	0	1	0	0
$s_3$	0	0	0	1	0
$s_4$	0	1	0	0	0
$s_5$	0	0	1	0	0
$s_6$	0	0	1	0	0
$s_7$	0	1	0	0	0
$s_8$	0	1	0	0	0
$s_9$	0	0	0	0	1

The solution  $v^*$  is the optimal state value, and  $\pi^*$  is an optimal policy.

# Bellman Optimality Equation: Proof

For any policy  $\pi$ :  $v_\pi = r_\pi + \gamma P_\pi v_\pi$

$$v^* = \max_{\pi} (r_\pi + \gamma P_\pi v^*) = r_{\pi^*} + \gamma P_{\pi^*} v^* \geq r_\pi + \gamma P_\pi v^*$$

Therefore:

$$v^* - v_\pi \geq (r_\pi + \gamma P_\pi v^*) - (r_\pi + \gamma P_\pi v_\pi) = \gamma P_\pi (v^* - v_\pi)$$

Repeatedly applying:

$$v^* - v_\pi \geq \lim_{n \rightarrow \infty} \gamma^n P_\pi^n (v^* - v_\pi) = 0$$

# Greedy optimal policy

$$v_{\pi} = \max_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi})$$

After obtaining  $v^*$ , we can obtain **action value** from **state value**:

$$q^*(s, a) = \sum_r p(r|s, a)r + \gamma \sum_{s'} p(s'|s, a)v^*(s')$$

Then the optimal policy  $\pi^*$  is

$$\pi^*(a|s) = \begin{cases} 1, & a = \operatorname{argmax}_a q^*(s, a) \\ 0, & a \neq \operatorname{argmax}_a q^*(s, a) \end{cases}$$

# Bellman (optimality) equation: Summary

- State value:

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

- Action value:

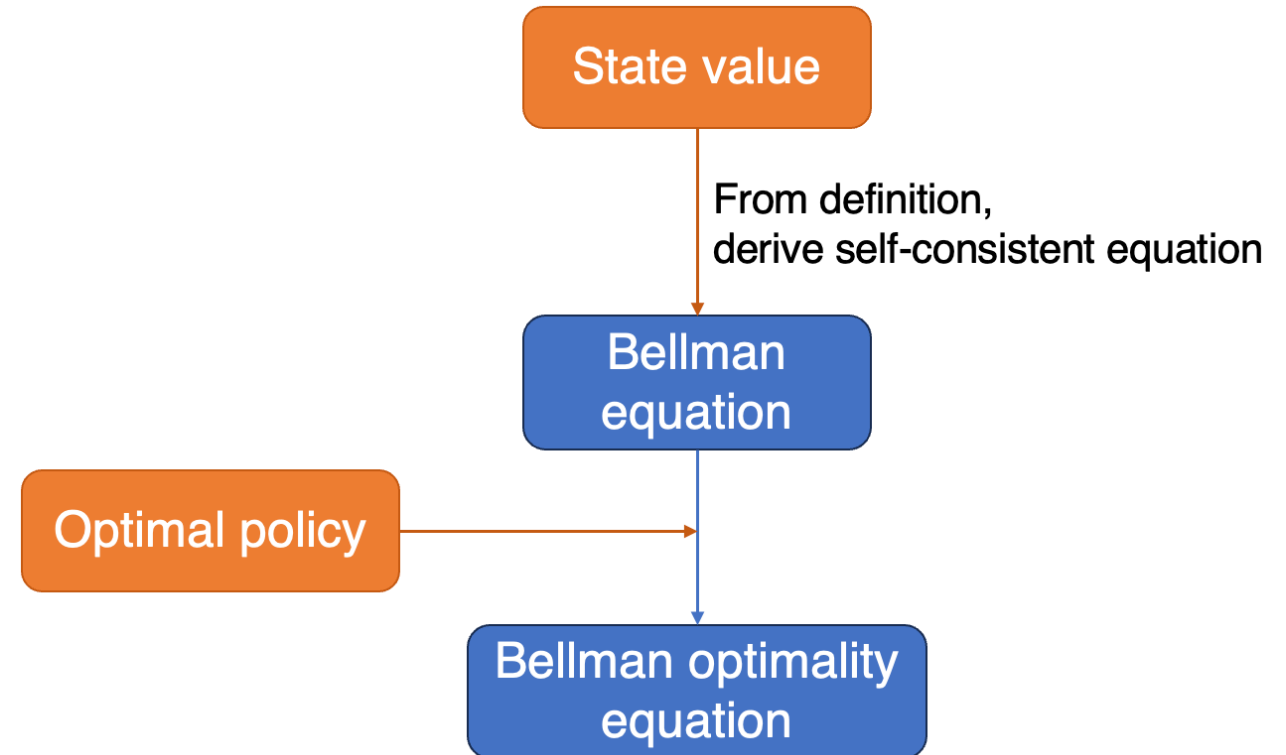
$$q_{\pi}(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

- Bellman Equation:

$$v_{\pi} = r_{\pi} + \gamma P_{\pi} v_{\pi}$$

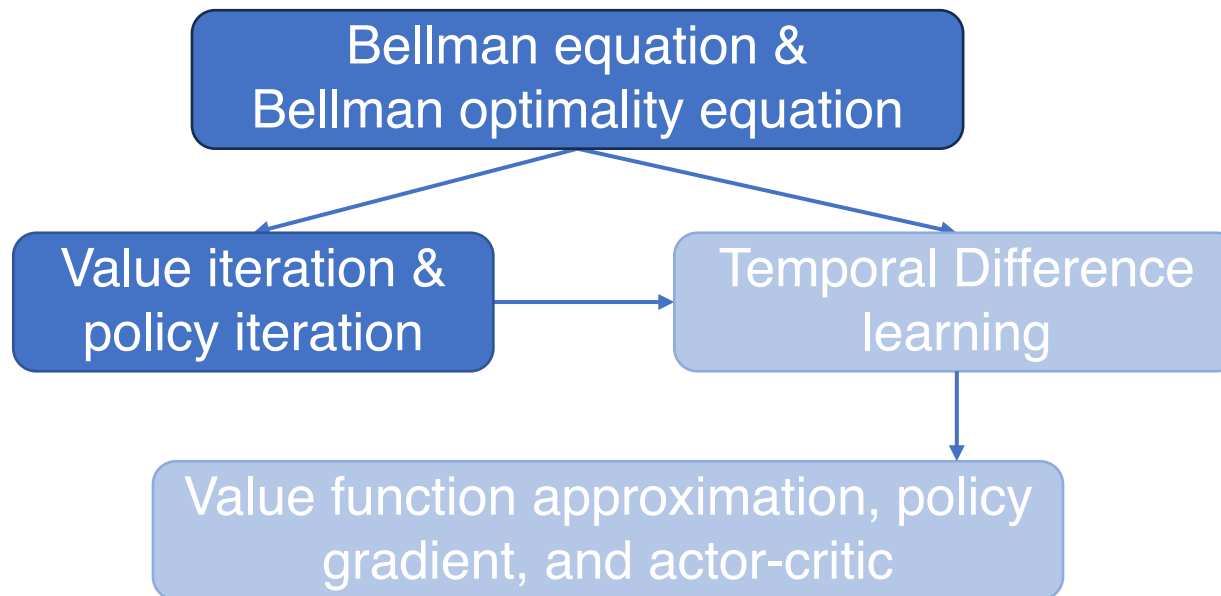
- Bellman Optimality Equation:

$$v_{\pi} = \max_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi})$$

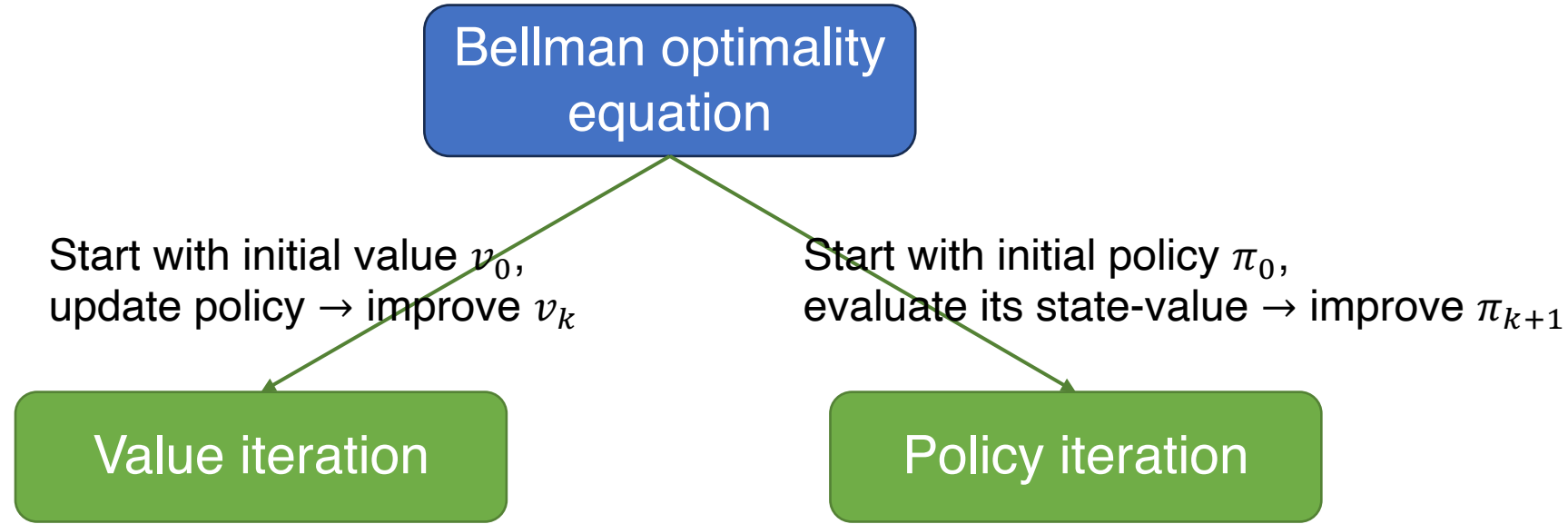


# Outline

- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- **Value iteration and policy iteration**
- Temporal difference learning
- Value function approximation, policy gradient methods, and actor-critic



# Value iteration and policy iteration: How we approach it



# Value iteration

## Bellman Optimality Equation:

$$v_{\pi} = \max_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi})$$

## Value iteration:

- Iterate
1. Policy update: Given  $v_k$ , update  $\pi_{k+1}$ .  
$$\pi_{k+1} = \operatorname{argmax}_{\pi} (r_{\pi} + \gamma P_{\pi} v_k)$$
  2. Value update: Given  $\pi_{k+1}$ , update  $v_{k+1}$ .  
$$v_{k+1} = r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k$$

# Policy iteration

## Bellman Optimality Equation:

$$v_{\pi} = \max_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi})$$

## Policy iteration:

- Iterate
1. Policy evaluation: Given  $\pi_k$ , solve  $v_{\pi_k}$ .  
$$v_{\pi_k} = r_{\pi_k} + \gamma P_{\pi_k} v_{\pi_k}$$
  2. Policy improvement: Given  $v_{\pi_k}$ , update  $\pi_{k+1}$ .  
$$\pi_{k+1} = \operatorname{argmax}_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi_k})$$

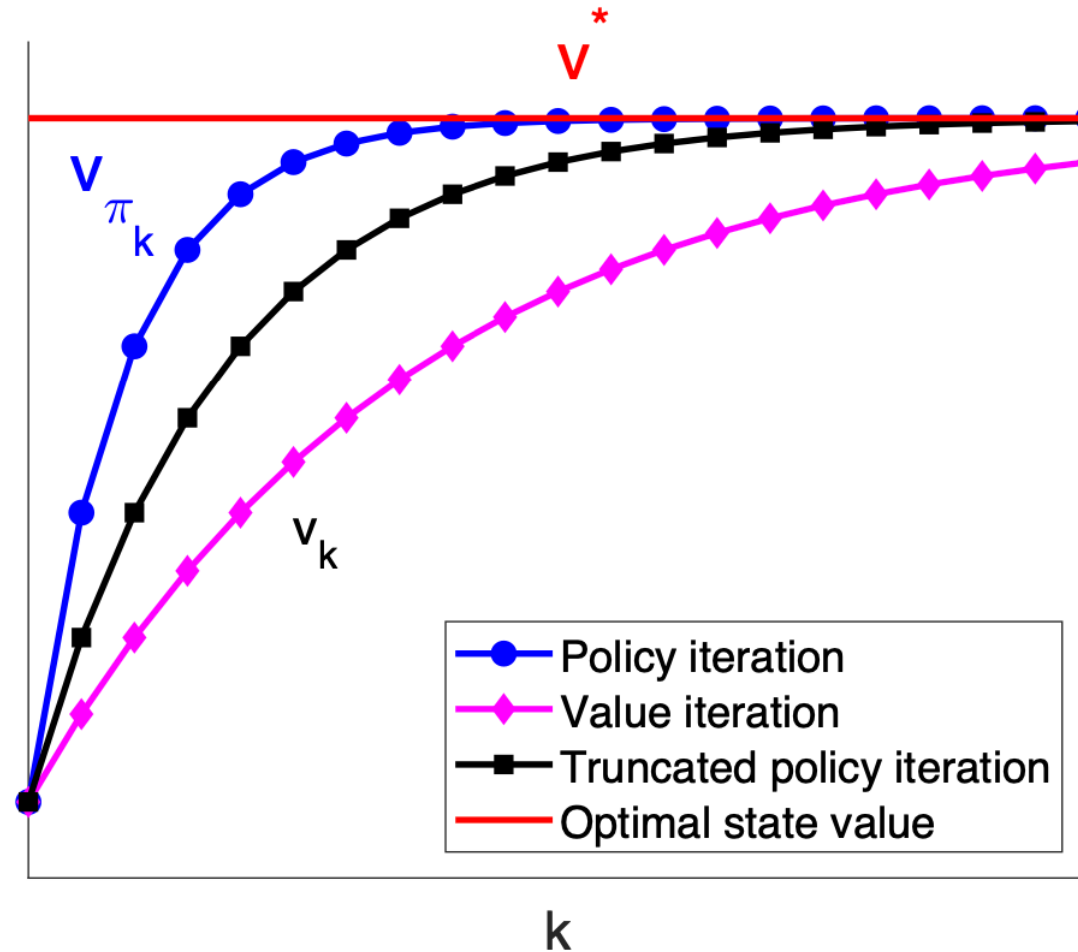
# Policy iteration & value iteration: Comparison

	Policy iteration algorithm	Value iteration algorithm
1) Policy	$\pi_0$	N/A
2) Value	$v_{\pi_0} = r_{\pi_0} + \gamma P_{\pi_0} v_{\pi_0}$	$v_0 := v_{\pi_0}$
3) Policy	$\pi_1 = \operatorname{argmax}_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi_0})$	$\pi_1 = \operatorname{argmax}_{\pi} (r_{\pi} + \gamma P_{\pi} v_0)$
4) Value	$v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$	$v_1 = r_{\pi_1} + \gamma P_{\pi_1} v_0$
5) Policy	$\pi_2 = \operatorname{argmax}_{\pi} (r_{\pi} + \gamma P_{\pi} v_{\pi_1})$	$\pi'_2 = \operatorname{argmax}_{\pi} (r_{\pi} + \gamma P_{\pi} v_1)$
...	...	...

The 4<sup>th</sup> step becomes different:

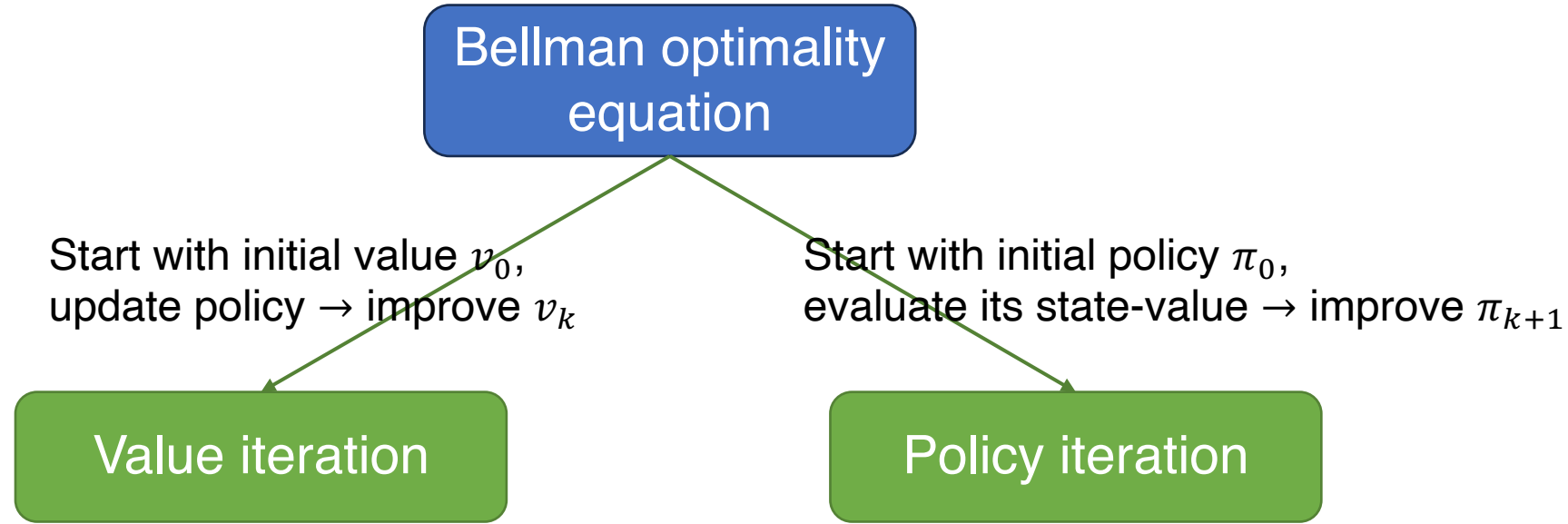
- In policy iteration, solving  $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$  requires an iterative algorithm (an infinite number of iterations)
- In value iteration,  $v_1 = r_{\pi_1} + \gamma P_{\pi_1} v_0$  is a one-step iteration

# Policy iteration & value iteration: Comparison



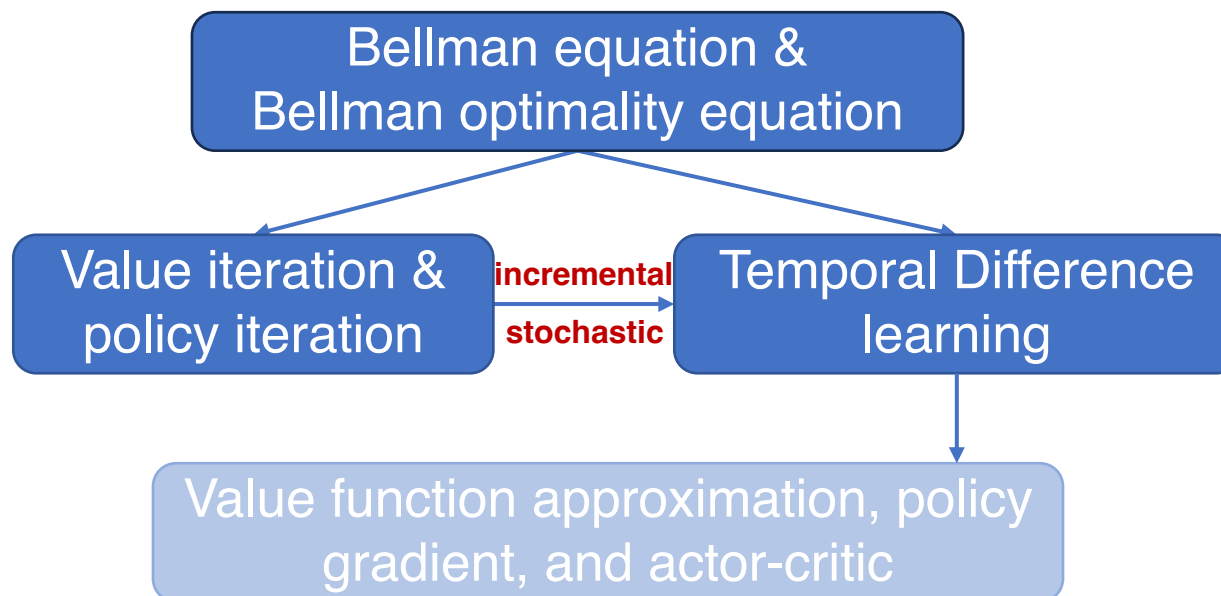
\*The truncated policy iteration algorithm computes a finite number of iterations

# Value iteration and policy iteration: Summary

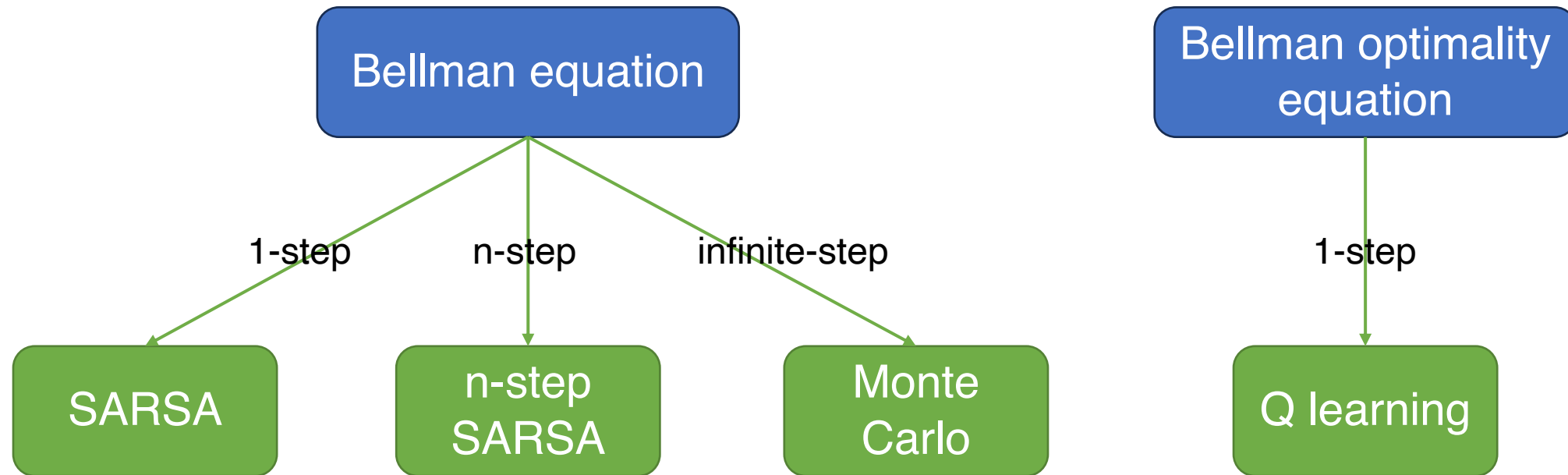


# Outline

- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- **Temporal difference learning**
- Value function approximation, policy gradient methods, and actor-critic



# Temporal difference learning: How we approach it



# Bellman equation (BE): An expectation form

Recall Bellman equation:

$$v_{\pi}(s) = r_{\pi}(s) + \gamma \sum_{s'} p_{\pi}(s'|s)v_{\pi}(s')$$

Bellman equation (expectation form):

$$v_{\pi}(s) = \mathbb{E}[R + \gamma v_{\pi}(S') | S = s]$$

# Temporal difference learning

Bellman equation (expected form):

$$v_{\pi}(s) = \mathbb{E}[R + \gamma v_{\pi}(S') | S = s]$$

Temporal difference (TD) learning:

$$v_{t+1}(s_t) = v_t(s_t) + \alpha_t(s_t) \left[ \underbrace{(r_{t+1} + \gamma v_t(s_{t+1}))}_{\text{TD target}} - v_t(s_t) \right]$$

TD error

- It uses bootstrapping, since  $r_{t+1} + \gamma v_t(s_{t+1})$  is a better estimate of  $v_{\pi}(s_t)$  than  $v_t(s_t)$ .

# SARSA

Bellman expectation equation:

$$q_{\pi}(s, a) = \mathbb{E}[R + \gamma q_{\pi}(S', A') | S = s, A = a]$$

SARSA:

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \alpha_t(s_t, a_t) \left[ (r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})) - q_t(s_t, a_t) \right]$$

It is named SARSA because each step of the algorithm involves  $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ .

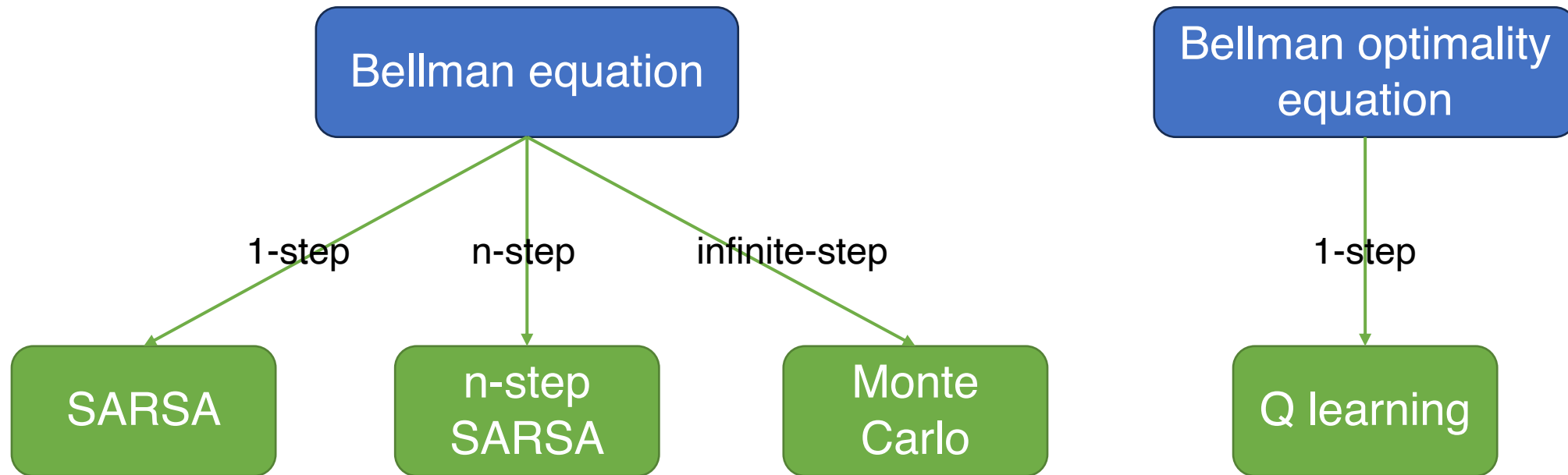
# Other temporal difference learning: Unified view

We also have Bellman equation (BE) and Bellman optimality equation (BOE) for the action value  $q(s, a)$ , and have corresponding TD learning algorithm:

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \alpha_t(s_t, a_t)[\bar{q}_t - q_t(s_t, a_t)]$$

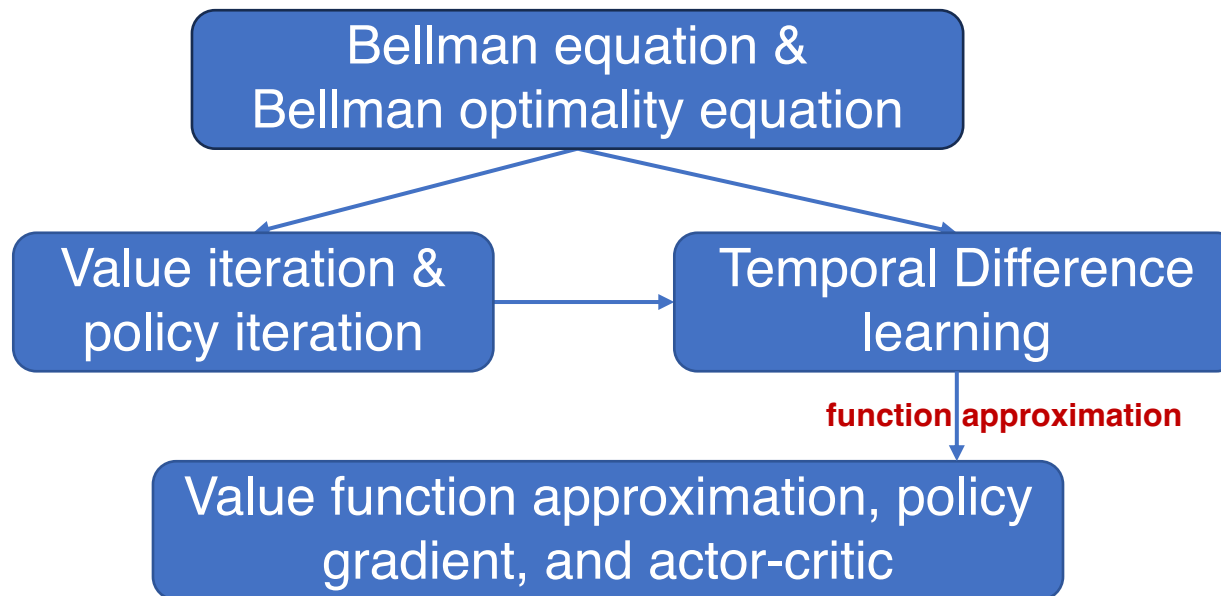
Algorithm	Equation and expression of $\bar{q}_t$
Sarsa	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})   S_t = s, A_t = a]$ $\bar{q}_t = r_{t+1} + \gamma q(s_{t+1}, a_{t+1})$
n-step Sarsa	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \dots + \gamma^n q_\pi(S_{t+n}, A_{t+n})   S_t = s, A_t = a]$ $\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \dots + \gamma^n q(s_{t+n}, a_{t+n})$
Q-learning	BOE: $q_\pi(s, a) = \mathbb{E} \left[ R_{t+1} + \gamma \max_a q_\pi(S_{t+1}, a)   S_t = s, A_t = a \right]$ $\bar{q}_t = r_{t+1} + \gamma \max_a q(s_{t+1}, a)$
Monte Carlo	BE: $q_\pi(s, a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \dots   S_t = s, A_t = a]$ $\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \dots$

# Temporal difference learning: Summary



# Outline

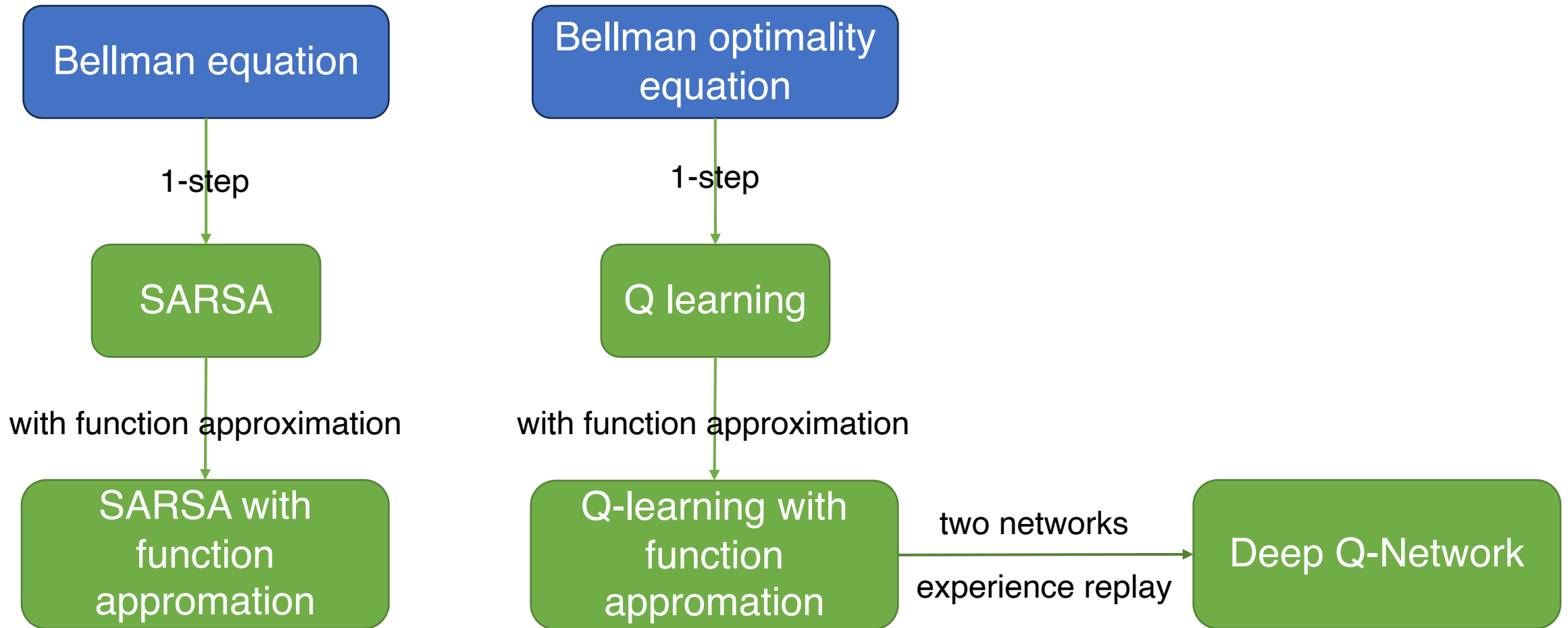
- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- Value function approximation, policy gradient methods, and actor-critic



# Outline

- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- **Value function approximation, policy gradient methods, and actor-critic**
  - Value function approximation:  $\hat{v}(s; w)$
  - Policy gradient methods
  - Actor-critic

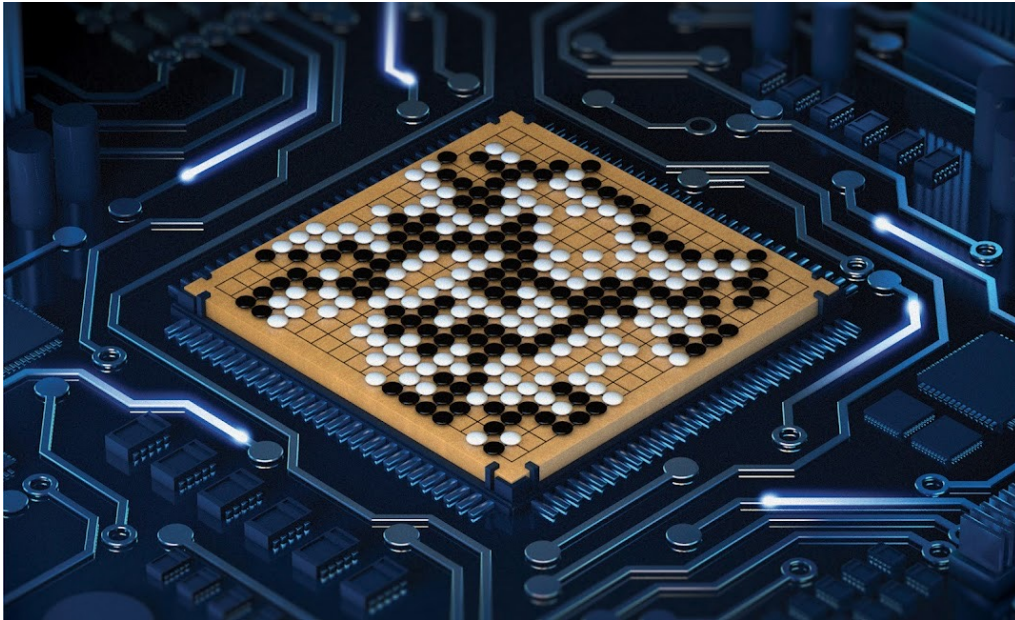
# Value function approximation: How we approach it



# DQN by DeepMind (2015): A new paradigm emerges



# AlphaGo by DeepMind (2016): A historical breakthrough



[AlphaGo - The Movie | Full award-winning documentary](#)

# Value function approximation

Previous

Now

$$v(s) \rightarrow \hat{v}(s; w)$$

State  $s$ : discrete  $\rightarrow$  continuous

Representation: tabular  $\rightarrow$  function

$\hat{v}(s; w)$  can be a neural network (MLP, transformer, CNN, GNN, etc.),  $w$  is its weights

# TD learning with function approximation

TD learning (tabular):

$$v_{t+1}(s_t) = v_t(s_t) + \alpha_t(s_t) \left[ \underbrace{(r_{t+1} + \gamma v_t(s_{t+1}))}_{\text{TD target}} - v_t(s_t) \right]$$

new estimate      current estimate      TD error

TD learning (with function approximation):

$$w_{t+1} = w_t + \alpha_t \left[ \underbrace{(r_{t+1} + \gamma \hat{v}(s_{t+1}; w_t))}_{\text{TD target}} - \hat{v}(s_t; w_t) \right] \nabla_w \hat{v}(s_t; w_t)$$

new estimate      current estimate      TD error

# SARSA with function approximation

SARSA (tabular):

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \alpha_t(s_t, a_t) \left[ (r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})) - q_t(s_t, a_t) \right]$$

SARSA (with function approximation):

$$w_{t+1} = w_t + \alpha_t \left[ (r_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}; w_t)) - \hat{q}(s_t, a_t; w_t) \right] \nabla_w \hat{q}(s_t, a_t; w_t)$$

# SARSA with function approximation: Algorithm

**Aim:** Search a policy that can lead the agent to the target from an initial state-action pair  $(s_0, a_0)$ .

For each episode, do

If the current  $s_t$  is not the target state, do

Take action  $a_t$  following  $\pi_t(s_t)$ , generate  $r_{t+1}, s_{t+1}$ , and then take action  $a_{t+1}$  following  $\pi_t(s_{t+1})$

**Value update (parameter update):**

$$w_{t+1} = w_t + \alpha_t \left[ r_{t+1} + \gamma \hat{q}(s_{t+1}, a_{t+1}, w_t) - \hat{q}(s_t, a_t, w_t) \right] \nabla_w \hat{q}(s_t, a_t, w_t)$$

**Policy update:**

$$\pi_{t+1}(a|s_t) = 1 - \frac{\epsilon}{|\mathcal{A}(s)|} (|\mathcal{A}(s)| - 1) \text{ if } a = \arg \max_{a \in \mathcal{A}(s_t)} \hat{q}(s_t, a, w_{t+1})$$
$$\pi_{t+1}(a|s_t) = \frac{\epsilon}{|\mathcal{A}(s)|} \text{ otherwise}$$

# Q-learning with function approximation

Q-learning (tabular):

$$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \alpha_t(s_t, a_t) \left[ r_{t+1} + \gamma \max_a q_t(s_{t+1}, a) - q_t(s_t, a_t) \right]$$

Q-learning (with function approximation):

$$w_{t+1} = w_t + \alpha_t \left[ r_{t+1} + \gamma \max_a \hat{q}(s_{t+1}, a; w_t) - \hat{q}(s_t, a_t; w_t) \right] \nabla_w \hat{q}(s_t, a_t; w_t)$$

# Deep Q-Network (DQN) for human-level Atari games

Starting from the Bellman equation:

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

It aims to minimize the Bellman optimality error:

$$J(w) = \mathbb{E} \left[ \left( R + \gamma \max_a \hat{q}(S_{t+1}, a; w) - \hat{q}(S_t, a; w) \right)^2 \right]$$

# Deep Q-Network (DQN): Challenges

Objective:

$$J(w) = \mathbb{E} \left[ \left( R + \gamma \max_a \hat{q}(S_{t+1}, a; w) - \hat{q}(S_t, a; w) \right)^2 \right]$$

How to optimize? Challenges:

1.  $w$  appears in both  $\hat{q}(S_t, a; w)$  and the target  $R + \gamma \max_a \hat{q}(S_{t+1}, a; w)$ .

Furthermore,  $\nabla_w \left( R + \gamma \max_a \hat{q}(S_{t+1}, a; w) \right) \neq \max_a \nabla_w \hat{q}(S_{t+1}, a; w)$ .

2. Expectation  $\mathbb{E}$  assumes that the state-action pair  $(S, A)$  is uniform. However, they are generated consecutively by certain policies, and have **strong correlations**.

# Deep Q-Network (DQN): Innovation

Objective:

$$J(w) = \mathbb{E} \left[ \left( R + \gamma \max_a \hat{q}(S_{t+1}, a; w_T) - \hat{q}(S_t, a; w) \right)^2 \right]$$

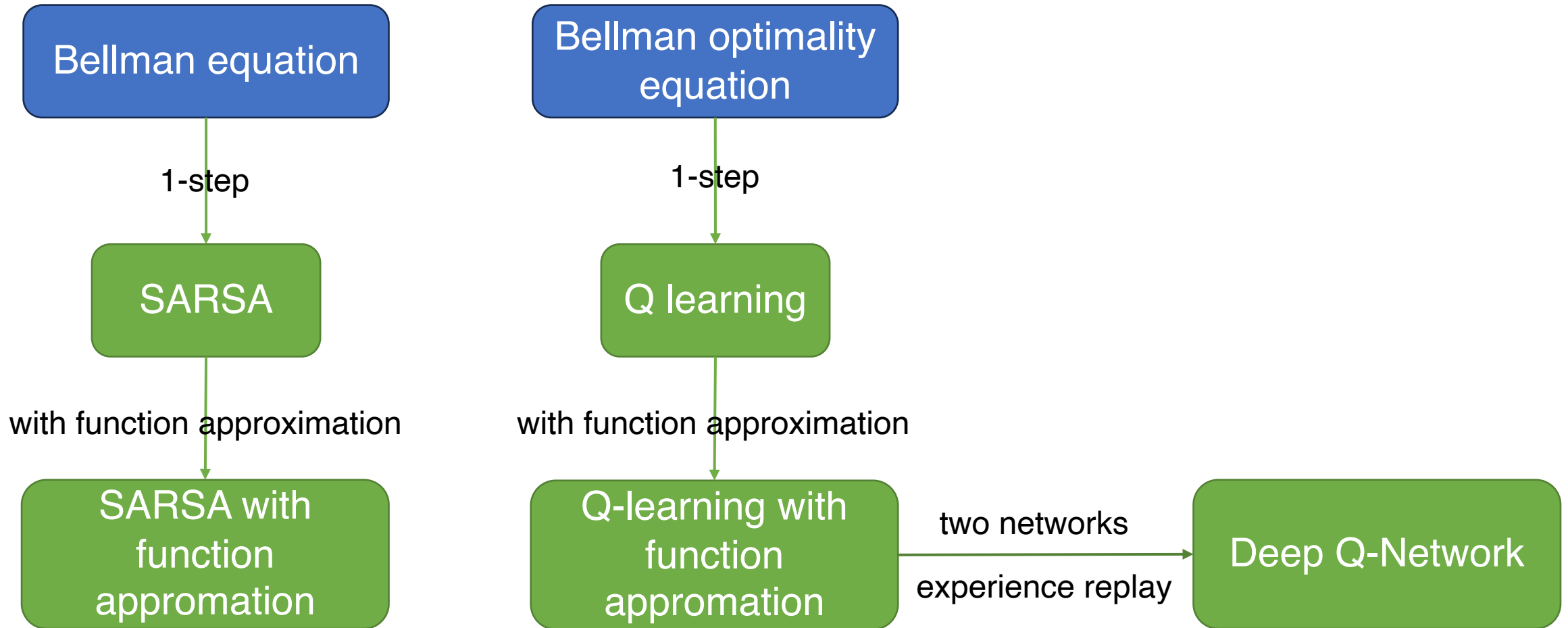
Two key innovations:

1. **Two networks:** Use a main network  $\hat{q}(S_t, a; w)$  for learning, and a target network  $\hat{q}(S_t, a; w_T)$  as target which stops gradient from passing through. Every K steps, update the  $w_T \leftarrow w$ .

## 2. Experience replay

Store samples  $\{(s, a, r, s')\}$  in a buffer, and in each step of training we draw a minibatch from the replay buffer.

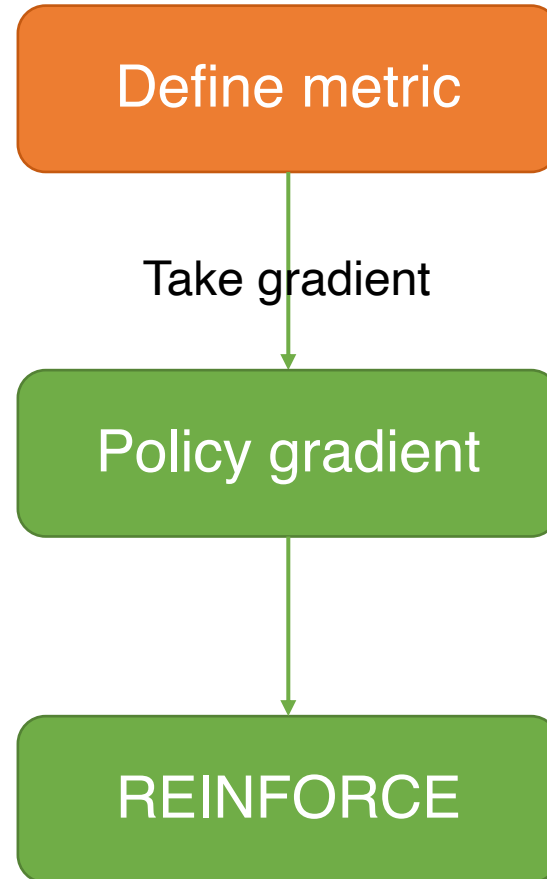
# Value function approximation: Summary



# Outline

- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- **Value function approximation, policy gradient methods, and actor-critic**
  - Value function approximation:  $\hat{v}(s; w)$
  - Policy gradient methods:  $\pi(a|s; \theta)$
  - Actor-critic

# Policy gradient: How we approach it



# Policy gradient: Define metrics

$d(s)$  is the weight for state  $s$ ,  $\sum_{s \in \mathcal{S}} d(s) = 1$

Metric	Expression 1	Expression 2	Expression 3
average value $\bar{v}_\pi$	$\sum_{s \in \mathcal{S}} d(s) v_\pi(s)$	$\mathbb{E}_{S \sim d}[v_\pi(S)]$	$\lim_{n \rightarrow \infty} \mathbb{E} \left[ \sum_{t=0}^n \gamma^t R_{t+1} \right]$
average reward $\bar{r}_\pi$	$\sum_{s \in \mathcal{S}} d(s) r_\pi(s)$	$\mathbb{E}_{S \sim d}[r_\pi(S)]$	$\lim_{n \rightarrow \infty} \frac{1}{n} \mathbb{E} \left[ \sum_{t=0}^{n-1} R_{t+1} \right]$

**Recall** in [Bellman Equation section](#) section:

- State value:  $v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$
- Immediate reward:  $r_\pi(s) = \mathbb{E}[R_{t+1} | S_t = s]$

# Policy gradient: Gradient expression

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \sum_{s \in \mathcal{S}} \eta(s) \sum_{a \in \mathcal{A}} \nabla_{\theta} \pi(a|s; \theta) q_{\pi}(s, a) \\ &= \mathbb{E}_{S \sim \eta, A \sim \pi(S; \theta)} [\nabla_{\theta} \ln \pi(A|S; \theta) q_{\pi}(S, A)]\end{aligned}$$

Here  $\eta(s)$  is a distribution of the states

# REINFORCE algorithm

From  $\nabla_{\theta} J(\theta) = \mathbb{E}_{S \sim \eta, A \sim \pi(S; \theta)} [\nabla_{\theta} \ln \pi(A|S; \theta) q_{\pi}(S, A)]$

We let

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} \ln \pi(a_t | s_t; \theta_t) q_t(s_t, a_t)$$

**Initialization:** Initial parameter  $\theta$ ;  $\gamma \in (0, 1)$ ;  $\alpha > 0$ .

**Goal:** Learn an optimal policy to maximize  $J(\theta)$ .

For each episode, do

Generate an episode  $\{s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T\}$  following  $\pi(\theta)$ .

For  $t = 0, 1, \dots, T - 1$ :

*Value update:*  $q_t(s_t, a_t) = \sum_{k=t+1}^T \gamma^{k-t-1} r_k$

*Policy update:*  $\theta \leftarrow \theta + \alpha \nabla_{\theta} \ln \pi(a_t | s_t, \theta) q_t(s_t, a_t)$

# Outline

- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- **Value function approximation, policy gradient methods, and actor-critic**
  - Value function approximation:  $\hat{v}(s; w)$
  - Policy gradient methods:  $\pi(a|s; \theta)$
  - Actor-critic: Combine  $\pi(a|s; \theta)$  and  $\hat{v}(s; w)$

# From REINFORCE to actor-critic

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{S \sim \eta, A \sim \pi(S; \theta)} [\nabla_{\theta} \ln \pi(A|S; \theta) q_{\pi}(S, A)] \\ &= \mathbb{E}_{S \sim \eta, A \sim \pi(S; \theta)} [\nabla_{\theta} \ln \pi(A|S; \theta) (q_{\pi}(S, A) - b(S))] \end{aligned}$$

here  $b(S)$  can be any function. We choose one that **reduces variance**.

REINFORCE:  $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} \ln \pi(a_t | s_t; \theta_t) q_t(s_t, a_t)$

Advantage Actor-critic (A2C):

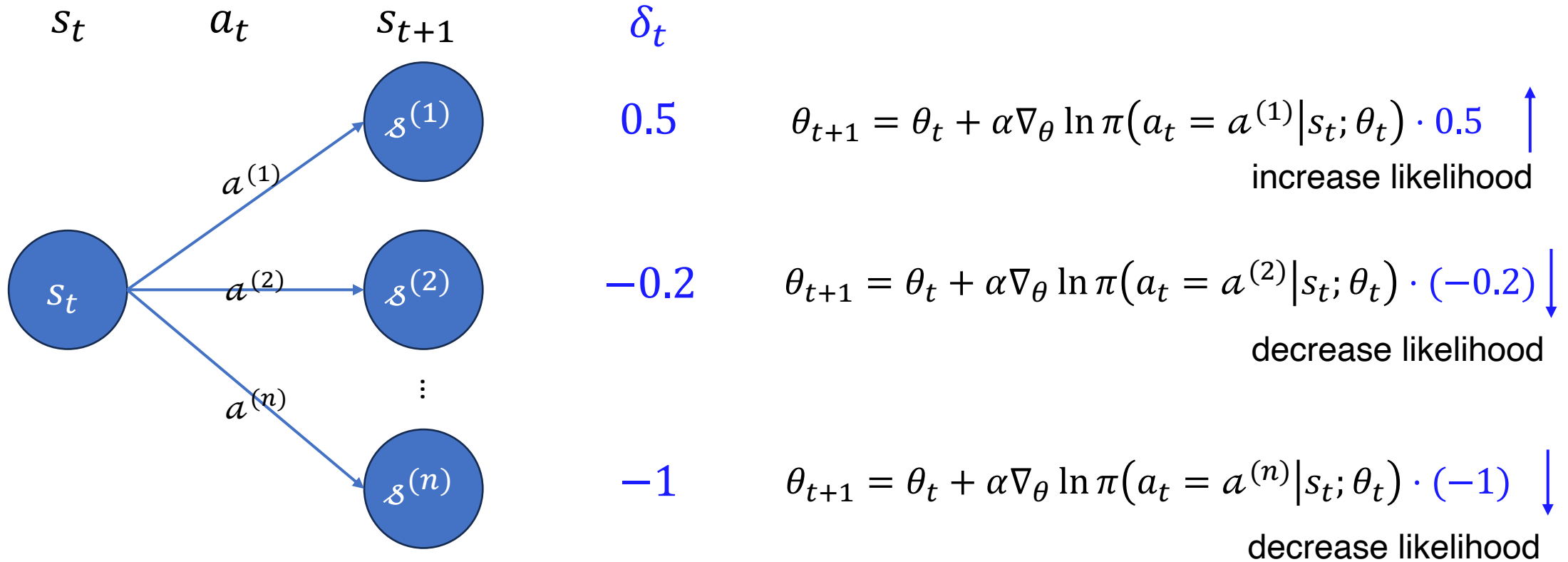
Policy update:  $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} \ln \pi(a_t | s_t; \theta_t) \underbrace{(r_{t+1} + \gamma v(s_{t+1}; w_t) - v_t(s_t; w_t))}_{\text{Advantage}}$

Value update:  $w_{t+1} = w_t + \alpha_t [(r_{t+1} + \gamma \hat{v}(s_{t+1}; w_t)) - \hat{v}(s_t; w_t)] \nabla_w \hat{v}(s_t; w_t)$

This value update is the same as in [TD learning with function approximation](#)

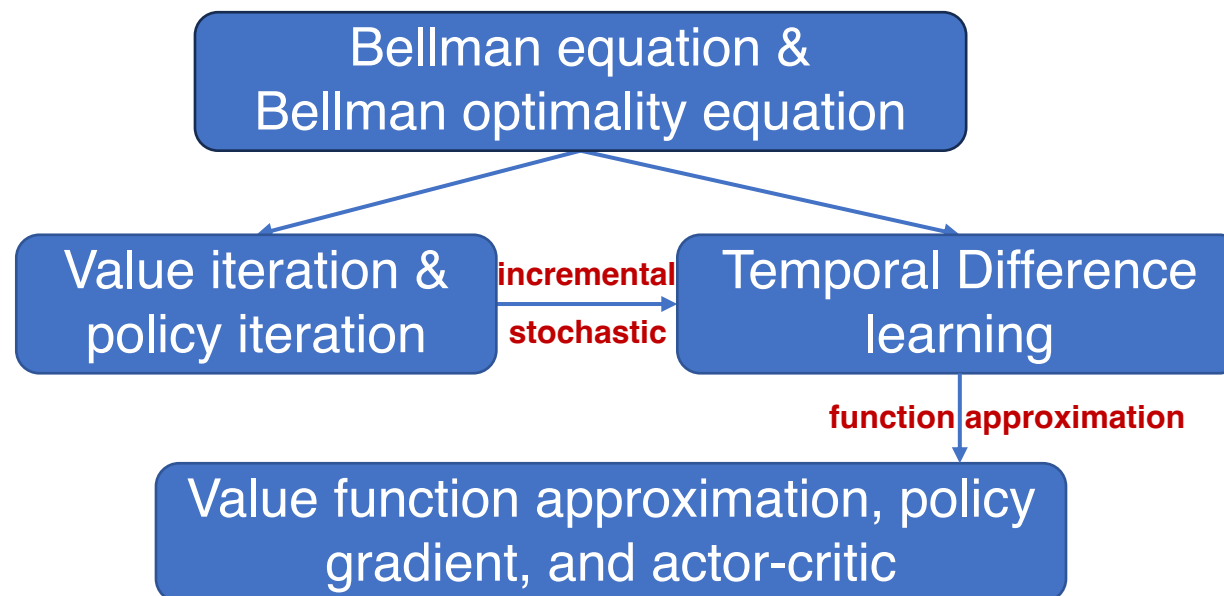
# Actor-critic: Interpretation

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} \ln \pi(a_t | s_t; \theta_t) \overbrace{\left( r_{t+1} + \gamma v(s_{t+1}; w_t) - v_t(s_t; w_t) \right)}^{\text{Advantage } \delta_t}$$



# Summary

- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- Value function approximation, policy gradient methods, and actor-critic



Bellman equation & Bellman optimality equation

BE:  $v_\pi = r_\pi + \gamma P_\pi v_\pi$

BOE:  $v_\pi = \max_\pi (r_\pi + \gamma P_\pi v_\pi)$

value iteration:  $v_0 \rightarrow \pi_1 \rightarrow v_1 \rightarrow \pi_1 \dots \rightarrow v_k$

policy iteration:  $\pi_0 \rightarrow v_{\pi_k} \rightarrow \pi_1 \rightarrow v_{\pi_1} \rightarrow \dots \rightarrow \pi_k$

$q_{t+1}(s_t, a_t) = q_t(s_t, a_t) + \alpha_t(s_t, a_t)[\bar{q}_t - q_t(s_t, a_t)]$

BE or BOE determines  $\bar{q}_t$

Value iteration & policy iteration

Temporal Difference learning

Value function approximation, policy gradient, and actor-critic

TD with function approximation:  $w_{t+1} = w_t + \alpha_t [(r_{t+1} + \gamma \hat{v}(s_{t+1}; w_t)) - \hat{v}(s_t; w_t)] \nabla_w \hat{v}(s_t; w_t)$

REINFORCE:  $\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi(a_t | s_t; \theta_t) q_t(s_t, a_t)$

Actor-critic:  $\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi(a_t | s_t; \theta_t) (r_{t+1} + \gamma v(s_{t+1}; w_t) - v_t(s_t; w_t))$

# Useful materials

- 课程：强化学习的数学原理（西湖大学 赵世钰）：  
<https://github.com/MathFoundationRL/Book-Mathematical-Foundation-of-Reinforcement-Learning>
- 上海交大强化学习课程：<https://wnzhang.net/teaching/sjtu-rl-2024/index.html>
- 张伟楠：《动手学强化学习》
- Useful code: <https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail>