Introduction to Reinforcement Learning

March 25

Tailin Wu, Westlake University

Website: ai4s.lab.westlake.edu.cn/course

Image from: DeepMind

Some materials are from Shiyu Zhao

Class 2: Deep learning fundamentals

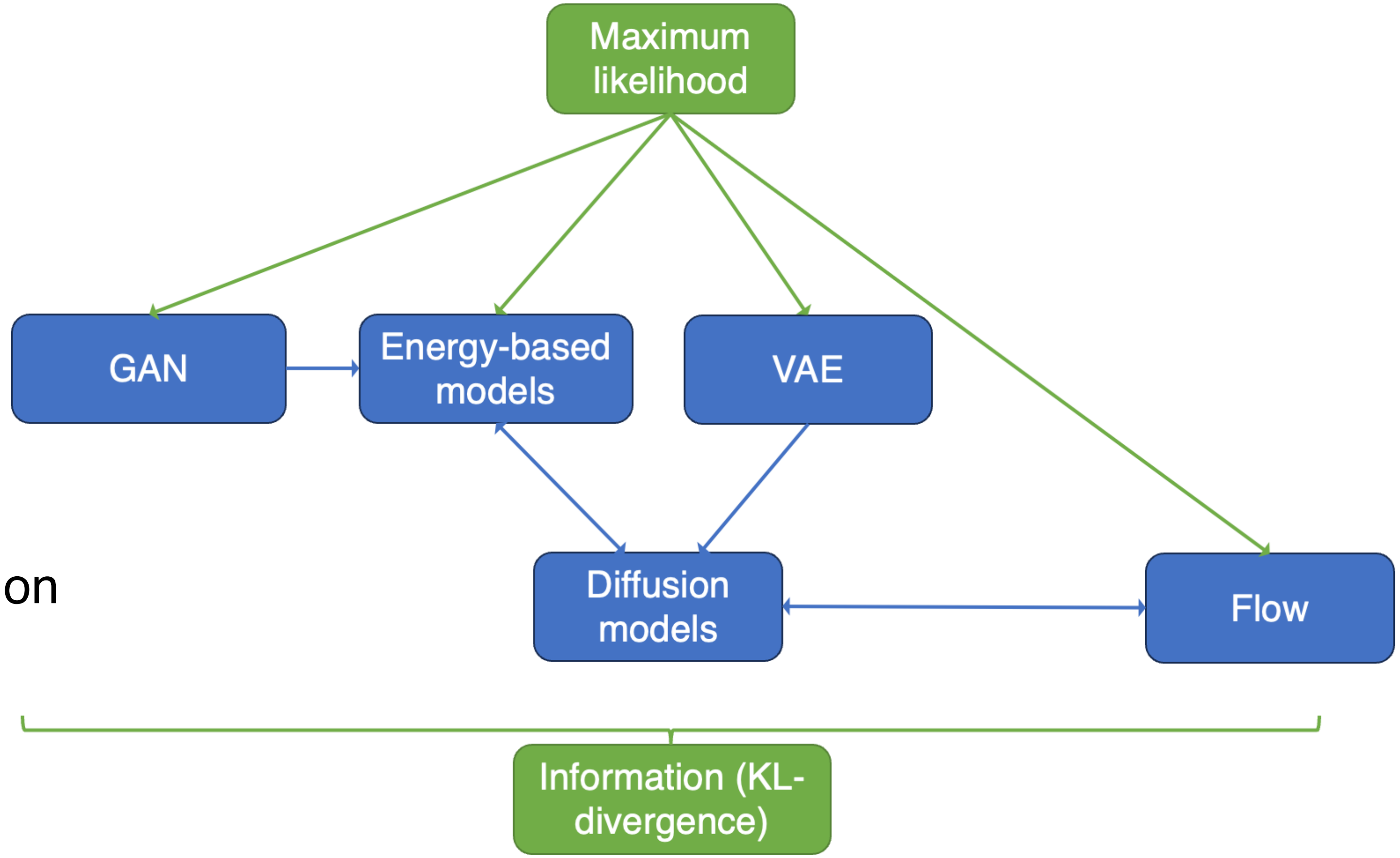
1. **Principle 1:** Model a hard transformation by composing simple transformations:
    - Multilayer Perceptron (MLP)
    - Backpropagation

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): $\max_{\boldsymbol{\epsilon} \in N_{\boldsymbol{\theta}}} \ell(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ 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

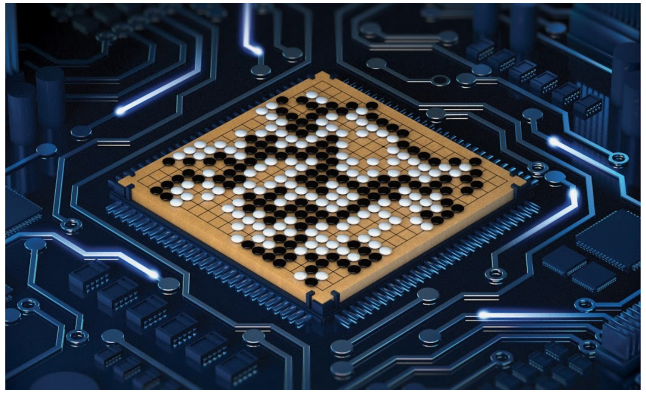


Class 4: Frontiers in Foundation Models

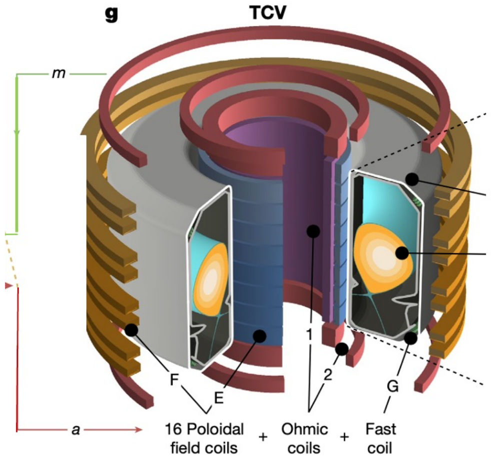
- **Principle 3** (the scaling law): AI methods that leverage computation are ultimately the most effective way of improvements (from "The bitter lesson" by Rich Sutton)
- **Principle 4** (the data law): Data is the ultimate way of regularization
- **Principle 5** (consistency law): The more consistent between training and testing, the better the performance

1. Transformer and its improvements
2. Different kinds of SSL methods
3. Application of foundation models

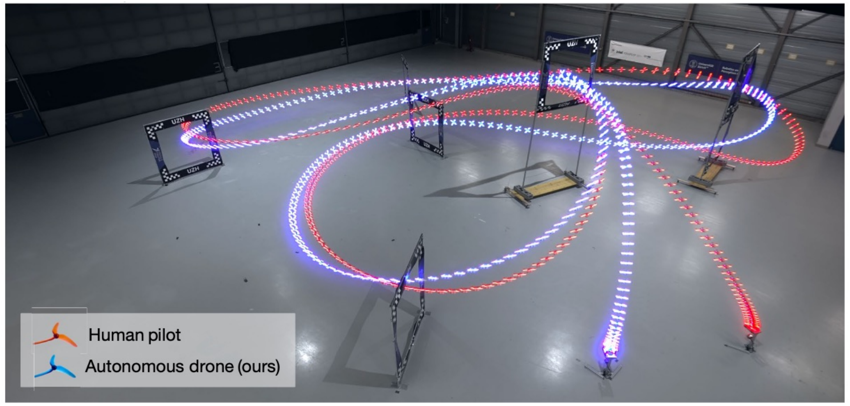
Reinforcement learning



AlphaGo [1]



Controlled nuclear fusion [2]

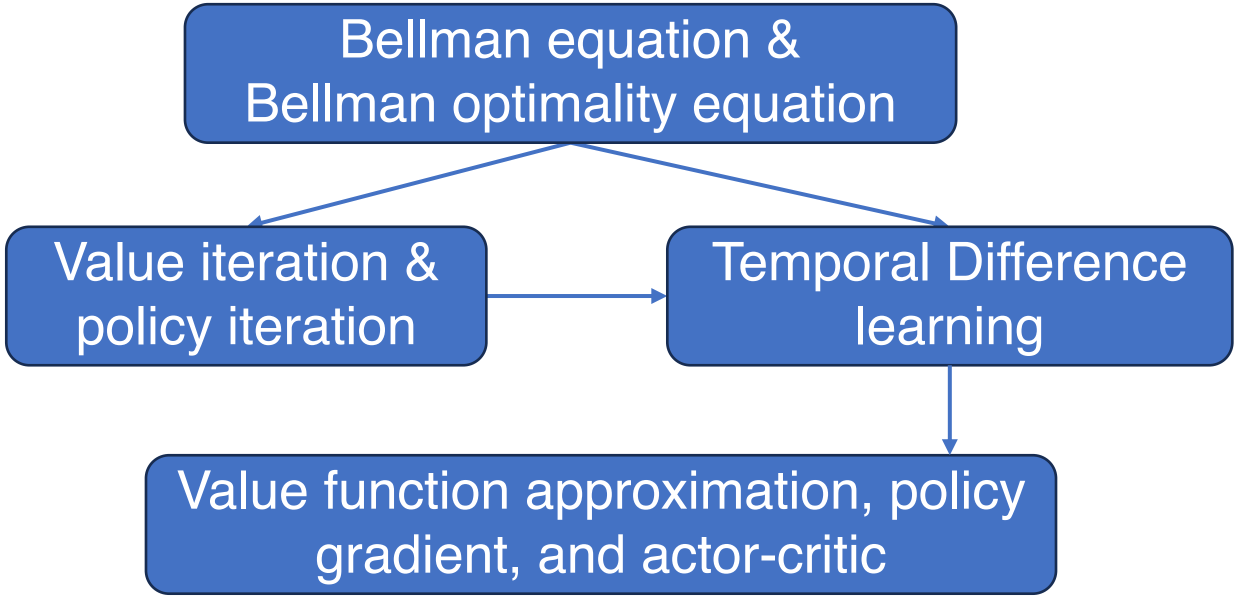


Drone racing [3]

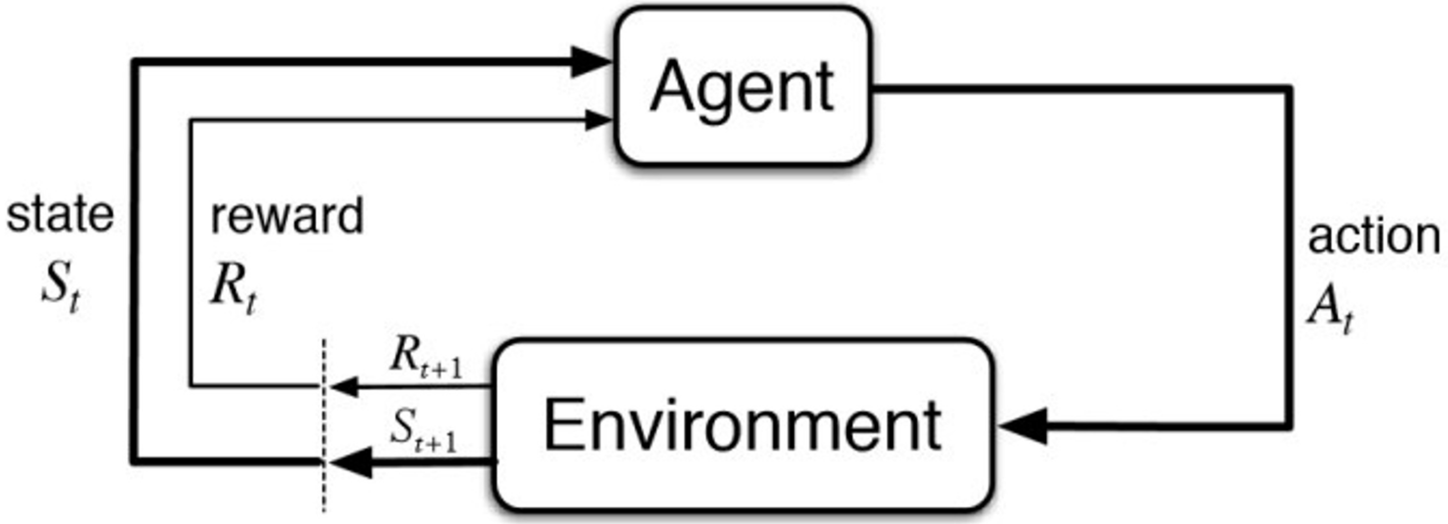
[1] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.
[2] Degrave, Jonas, et al. "Magnetic control of tokamak plasmas through deep reinforcement learning." *Nature* 602.7897 (2022): 414-419.
[3] Kaufmann, Elia, et al. "Champion-level 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 $\pi(A_t|S_t)$

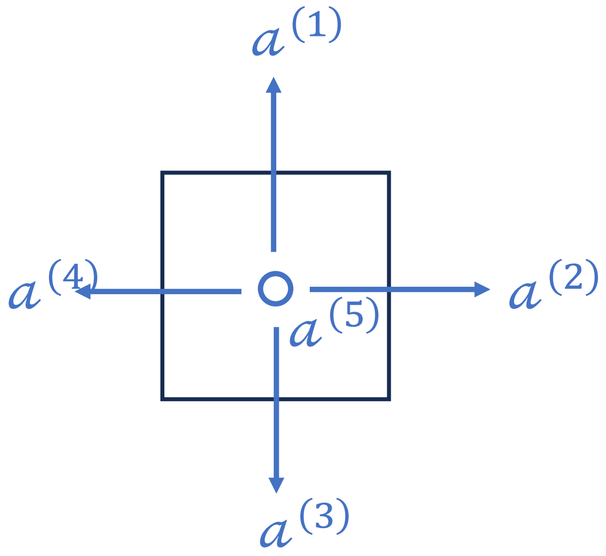
$$\max_{\pi(A_t|S_t)} \mathbb{E}_t[R_t]$$

Markov Decision Process (MDP): State and action

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

Action:

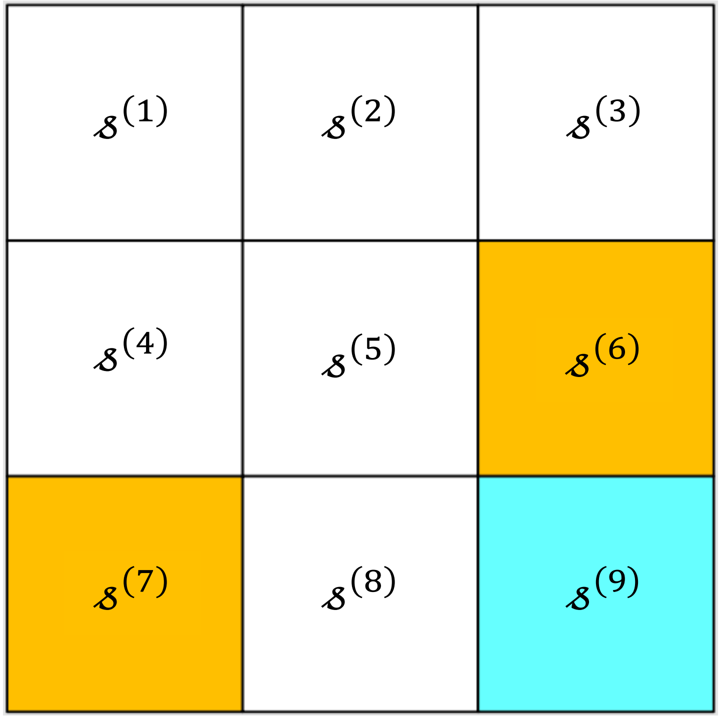
$a^{(1)}$: move upward
$a^{(2)}$: move rightward
$a^{(3)}$: move downward
$a^{(4)}$: move leftward
$a^{(5)}$: stay unchanged



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.,

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



Grid world

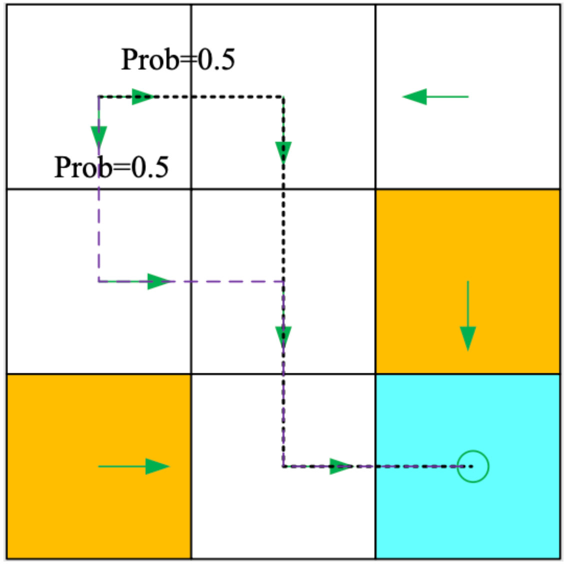
*We use subscript t in s_t, a_t to denote step t, and use superscript in $s^{(i)}, a^{(i)}$ to denote different choices of state or action.

Markov Decision Process (MDP): Policy

Policy $\pi(a|s)$:

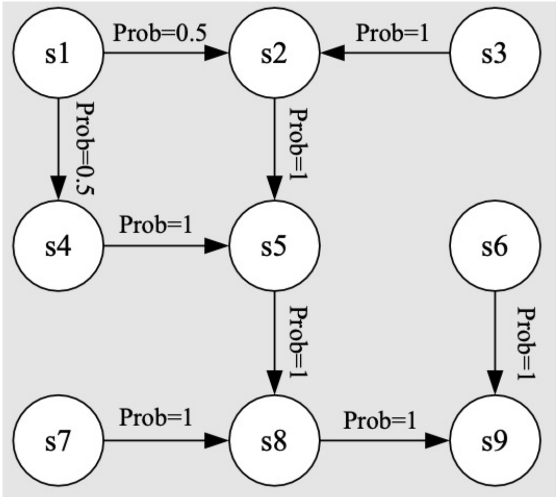
E.g., for $s = s^{(1)}$,

$$\pi\left(a = a^{(1)} \middle| s = s^{(1)}\right) = 0$$
$$\pi\left(a = a^{(2)} \middle| s = s^{(1)}\right) = 0.5$$
$$\pi\left(a = a^{(3)} \middle| s = s^{(1)}\right) = 0.5$$
$$\pi\left(a = a^{(4)} \middle| s = s^{(1)}\right) = 0$$
$$\pi\left(a = a^{(5)} \middle| s = s^{(1)}\right) = 0$$



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



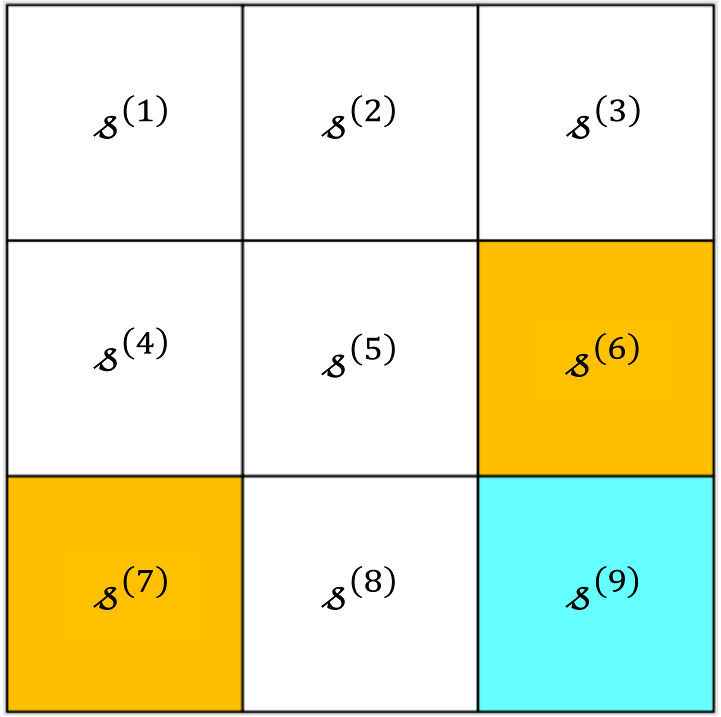
Markov Decision Process (MDP): Reward

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:

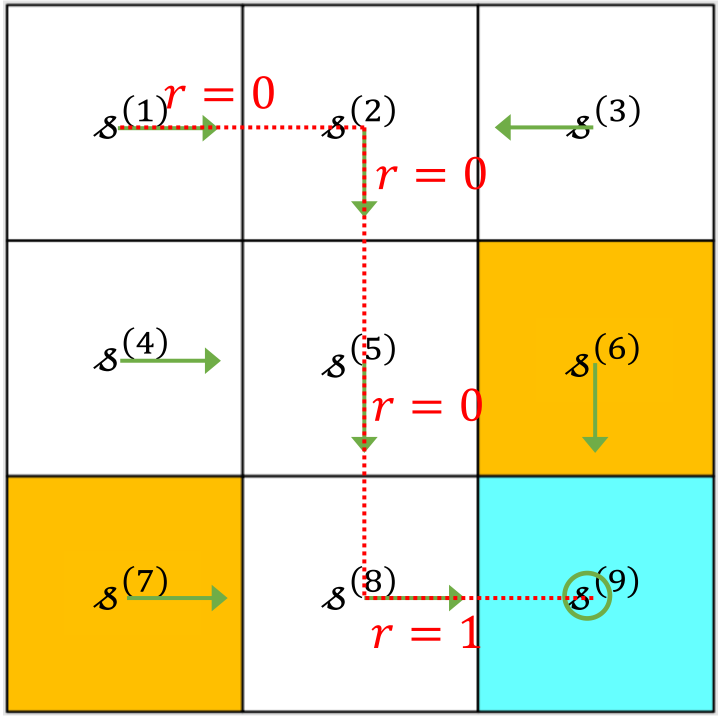
$$\text{return} = 0 + 0 + 0 + 1 = 1$$

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 γ,

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



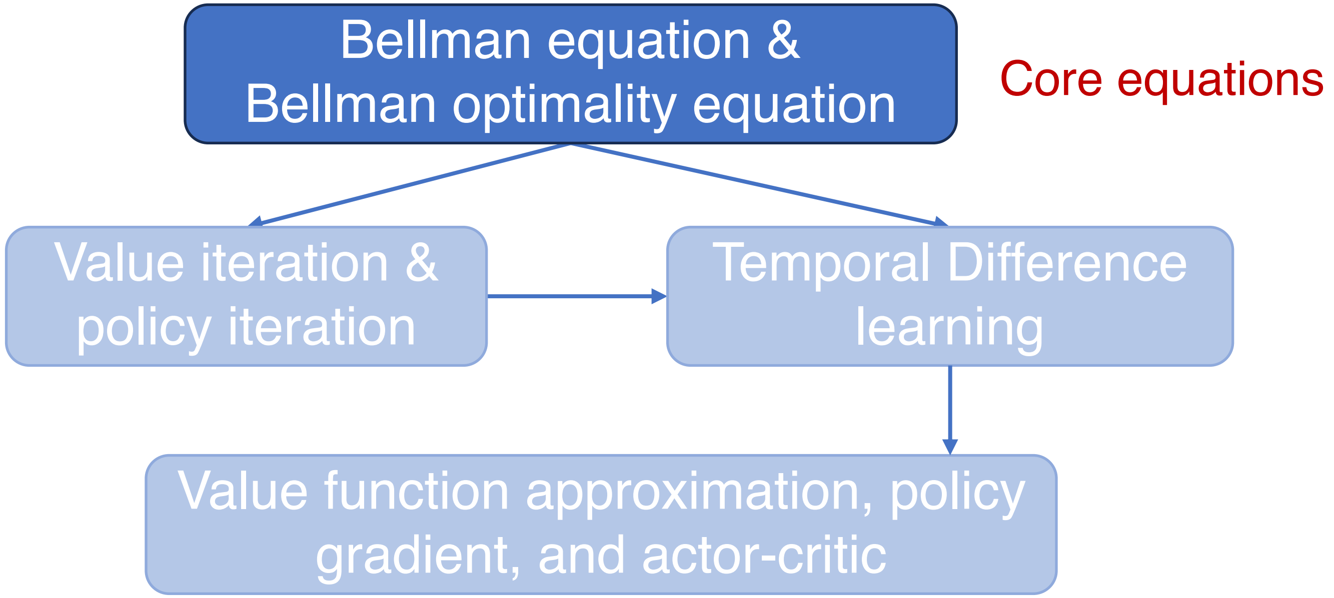
Trajectory

Markov Decision Process (MDP): Full components

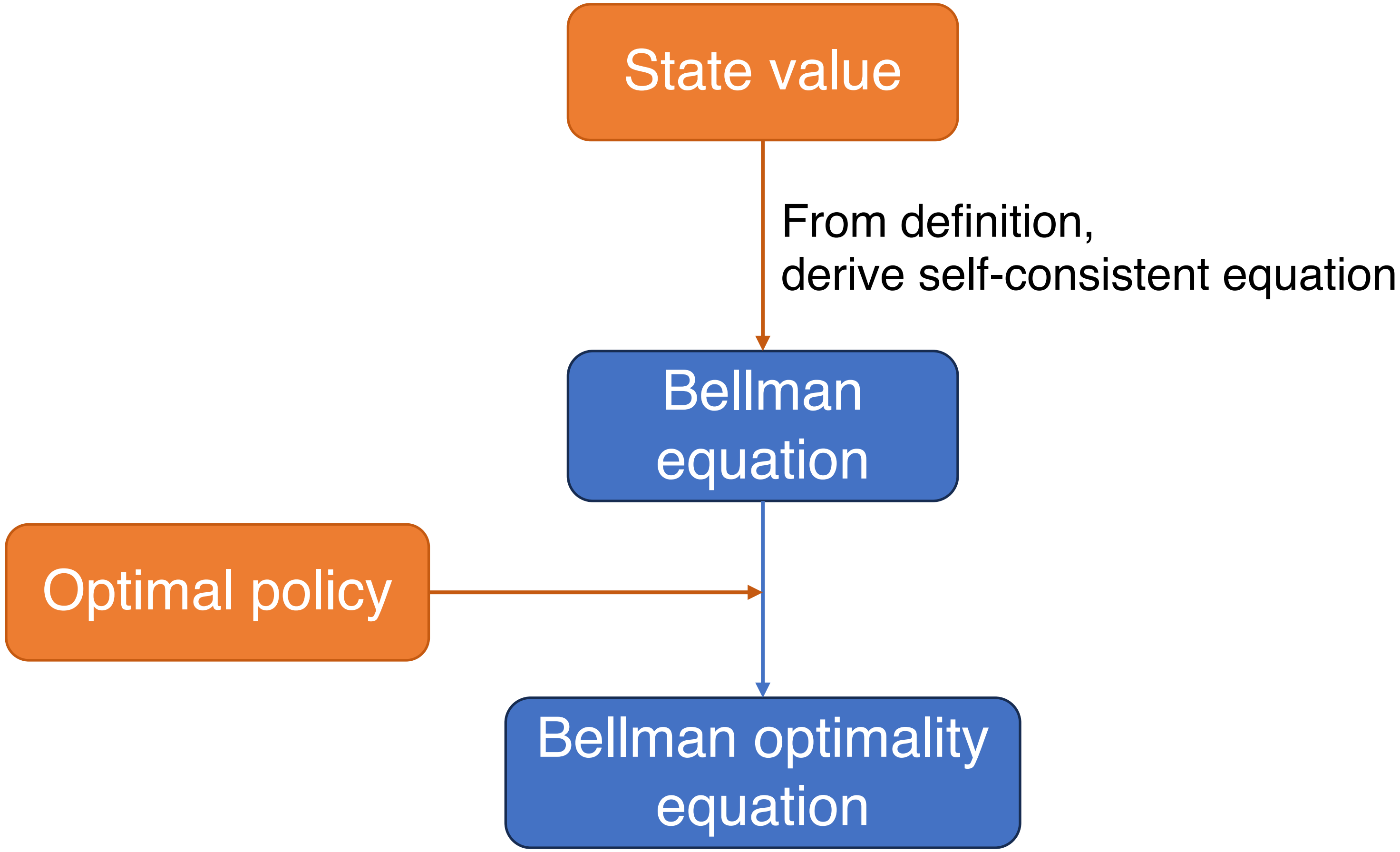
- **Sets:**
  - **State:** the set of states $\mathcal{S}$
  - **Action:** the set of actions $\mathcal{A}(s)$ is associated for state $s \in \mathcal{S}$.
  - **Reward:** the set of rewards $\mathcal{R}(s, a)$.
- **Probability distribution** (or called system model):
  - **State transition probability:** 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
  - $p(s_{t+1}|a_t, s_t, \ldots, a_0, s_0) = p(s_{t+1}|a_t, s_t)$
  - $p(r_{t+1}|a_t, s_t, \ldots, a_0, s_0) = p(r_{t+1}|a_t, s_t)$.

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

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

The state value function evaluates how good the policy π is.

Bellman Equation: Derivation from state value definition

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

Here

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

Bellman Equation: Derivation

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

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

Here

$$p_\pi(s'|s) \coloneqq \sum_a \pi(a|s) p(s'|s,a)$$

is the transition probability under policy π.

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:

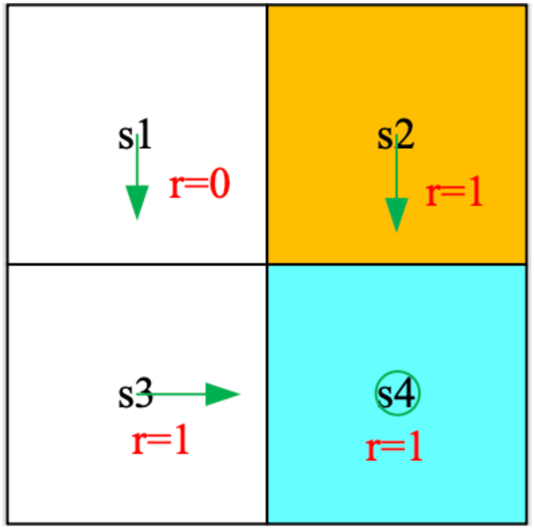
$$v_\pi = r_\pi + \gamma P_\pi v_\pi$$

$v_\pi = [v_\pi(s_1), \ldots v_\pi(s_n)]^T \in R^n$
$r_\pi = [r_\pi(s_1), \ldots r_\pi(s_n)]^T \in R^n$
$P_\pi \in R^{n \times n}, [P_\pi]_{ij} = p_\pi(s_j|s_i)$ is the state transition matrix

Bellman Equation: Example

Bellman equation:

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



2×2 grid world

For the specific policy in this grid world:

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

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:

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

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

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 and taking an action, and then following the policy π.

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 ⇒ 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 ⇒ action value

Optimal policy: Definition

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

Bellman Optimality Equation

Recall Bellman Equation: $v_\pi = r_\pi + \gamma P_\pi v_\pi$

Bellman Optimality Equation:

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

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

	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 π^* is an optimal policy.

Bellman Optimality Equation: Proof

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

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

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:

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

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 π^* is

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

Bellman (optimality) equation: Summary

- State value:

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

- Action value:

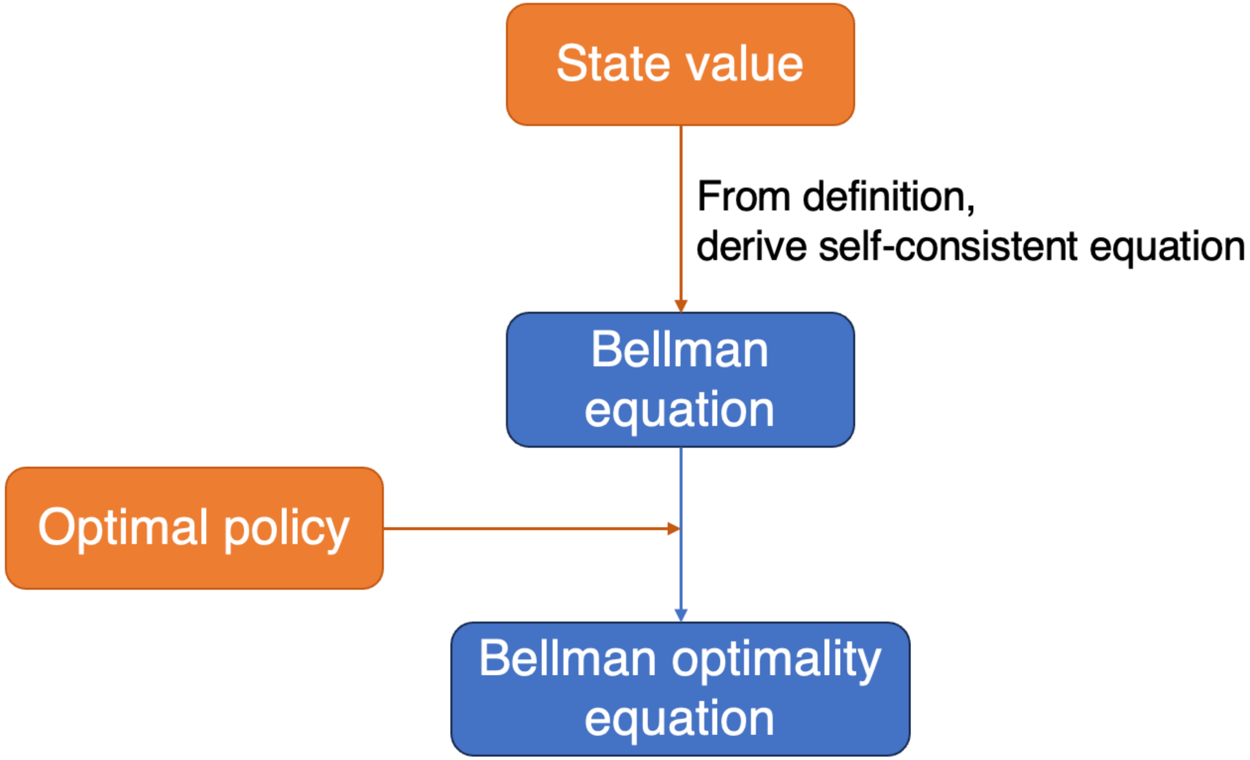
$$q_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a]$$

- Bellman Equation:

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

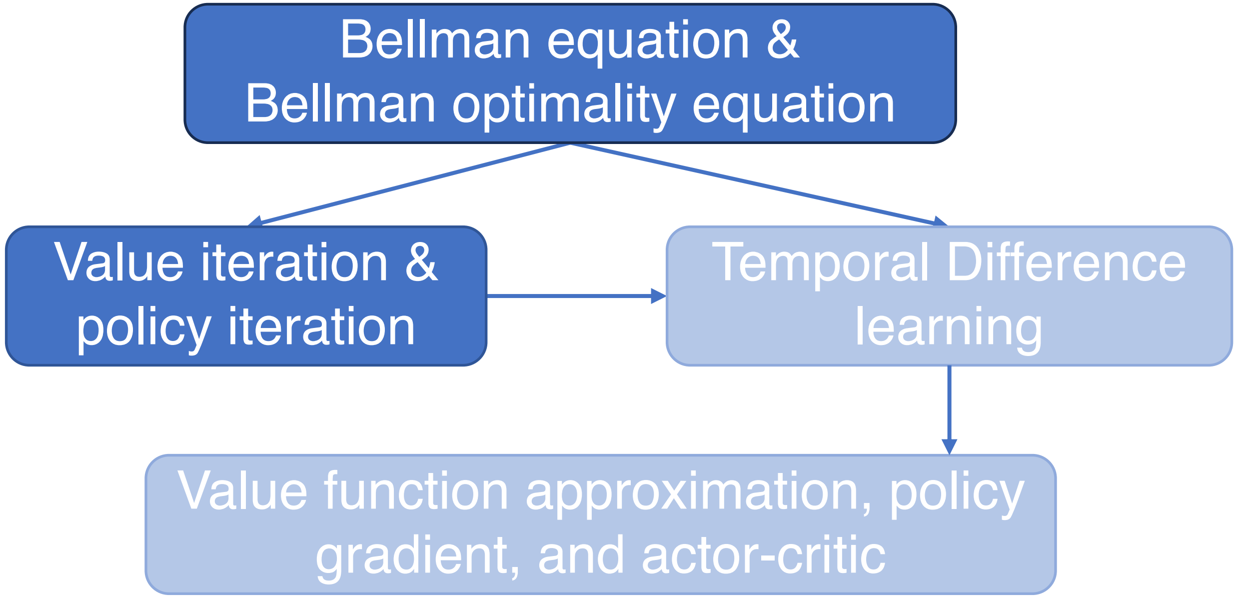
- Bellman Optimality Equation:

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

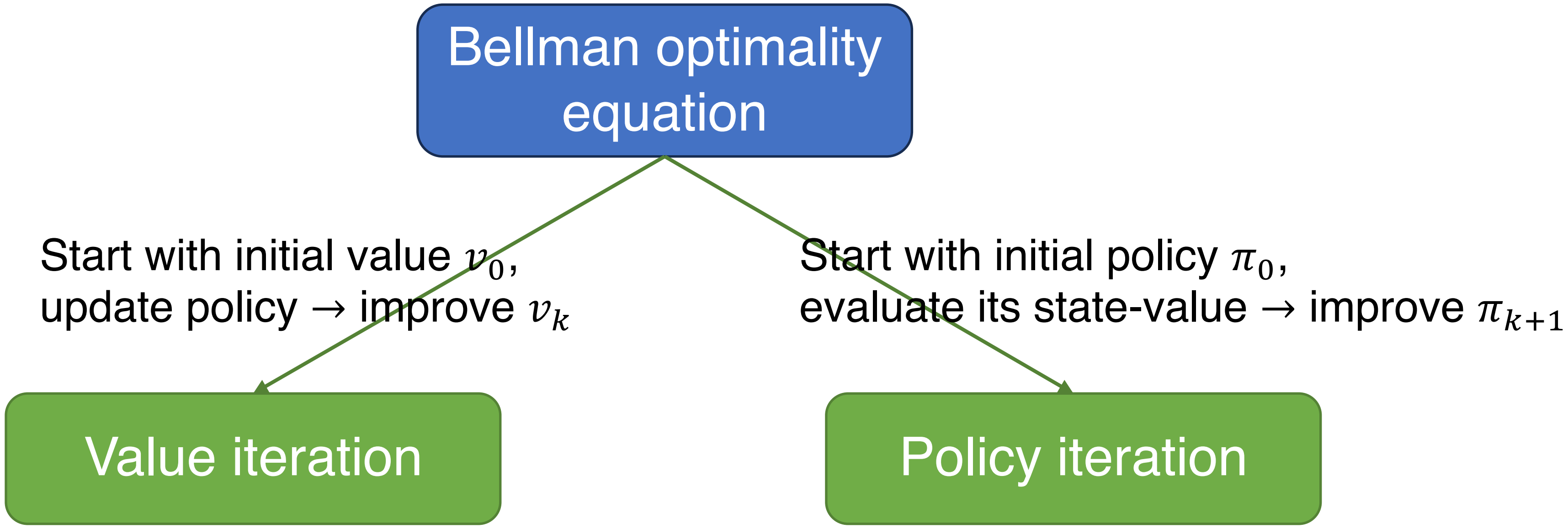


Outline

- 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 π_{k+1}.

$$\pi_{k+1} = \text{argmax}_\pi (r_\pi + \gamma P_\pi v_k)$$

2. Value update: Given π_{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 π_k, solve v_{π_k}.

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

2. Policy improvement: Given v_{π_k}, update π_{k+1}.

$$\pi_{k+1} = \text{argmax}_\pi \left(r_\pi + \gamma P_\pi v_{\pi_k} \right)$$

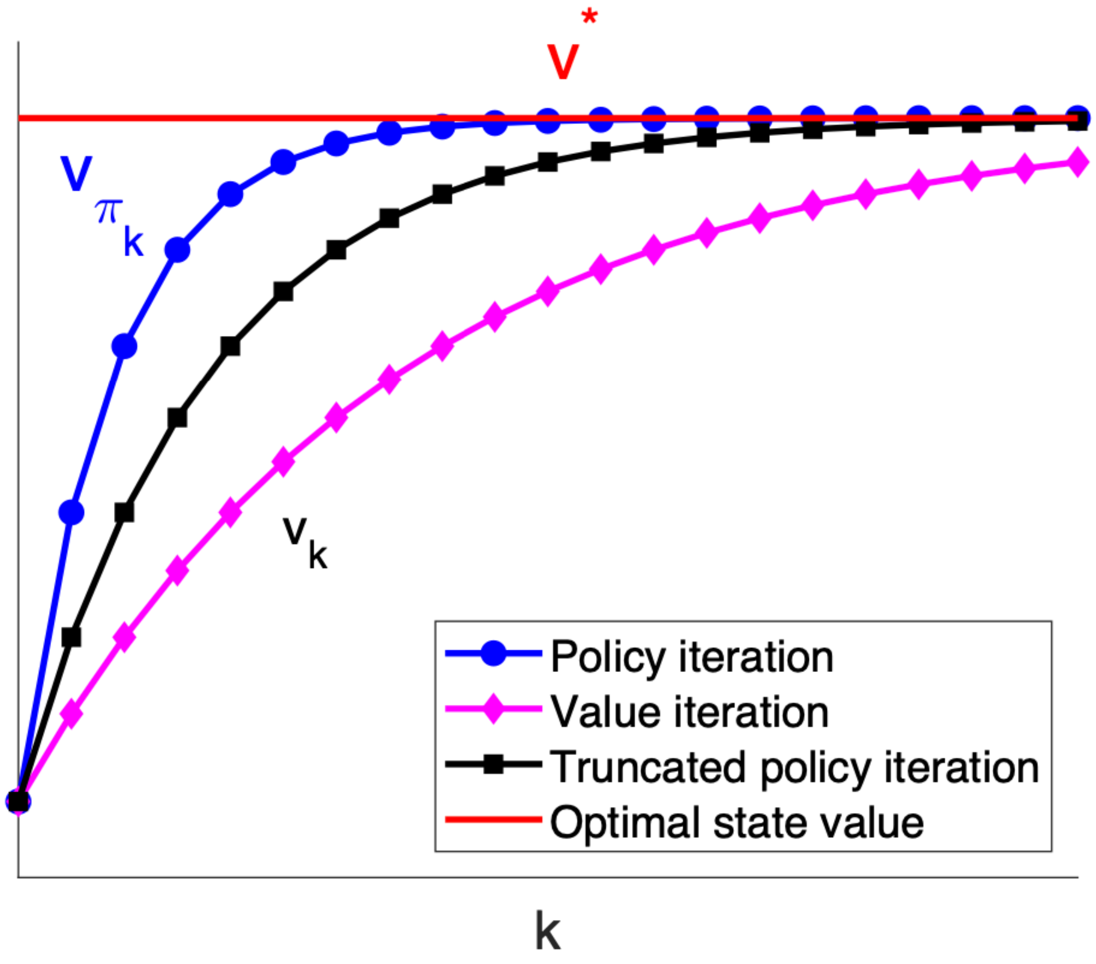
Policy iteration & value iteration: Comparison

	Policy iteration algorithm	Value iteration algorithm
1) Policy	π_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	$\pi_1 = \text{argmax}_\pi (r_\pi + \gamma P_\pi v_{\pi_0})$	$\pi_1 = \text{argmax}_\pi (r_\pi + \gamma P_\pi v_0)$
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	$\pi_2 = \text{argmax}_\pi (r_\pi + \gamma P_\pi v_{\pi_1})$	$\pi_2' = \text{argmax}_\pi (r_\pi + \gamma P_\pi v_1)$
…	…	…

The 4th step becomes different:

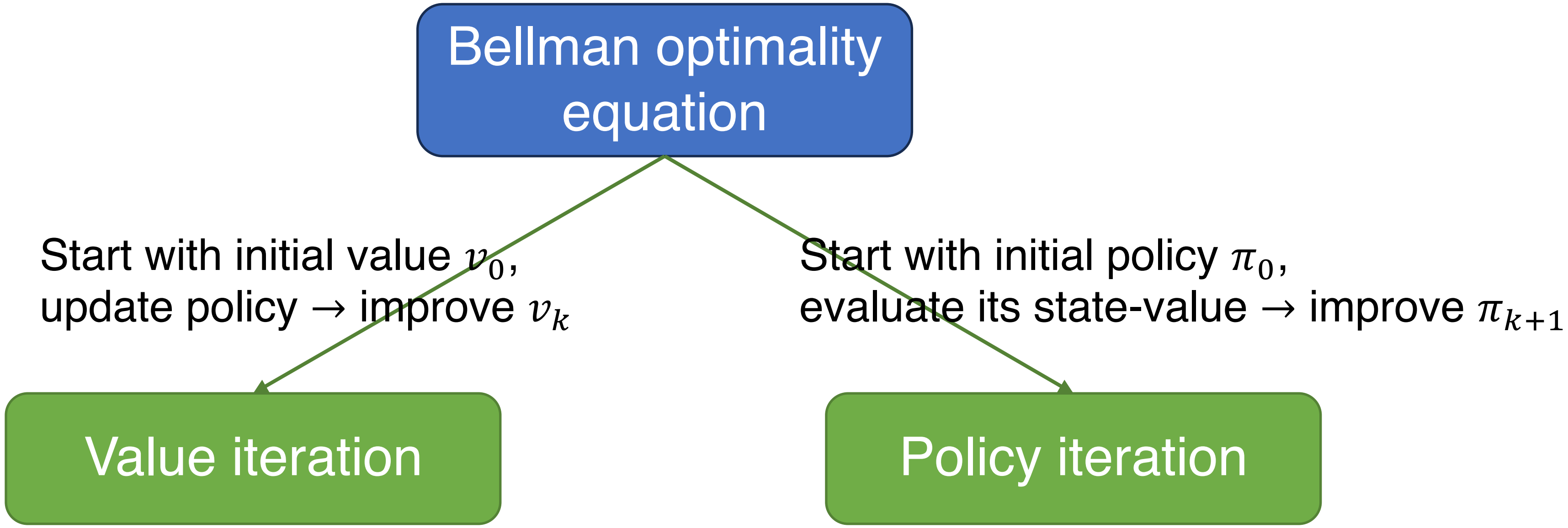
- In policy iteration, solving $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$ requires an iterative algorithm (an infinite number of iterations)
- In value iteration, $v_1 = r_{\pi_1} + \gamma P_{\pi_1} v_0$ is a one-step iteration

Policy iteration & value iteration: Comparison



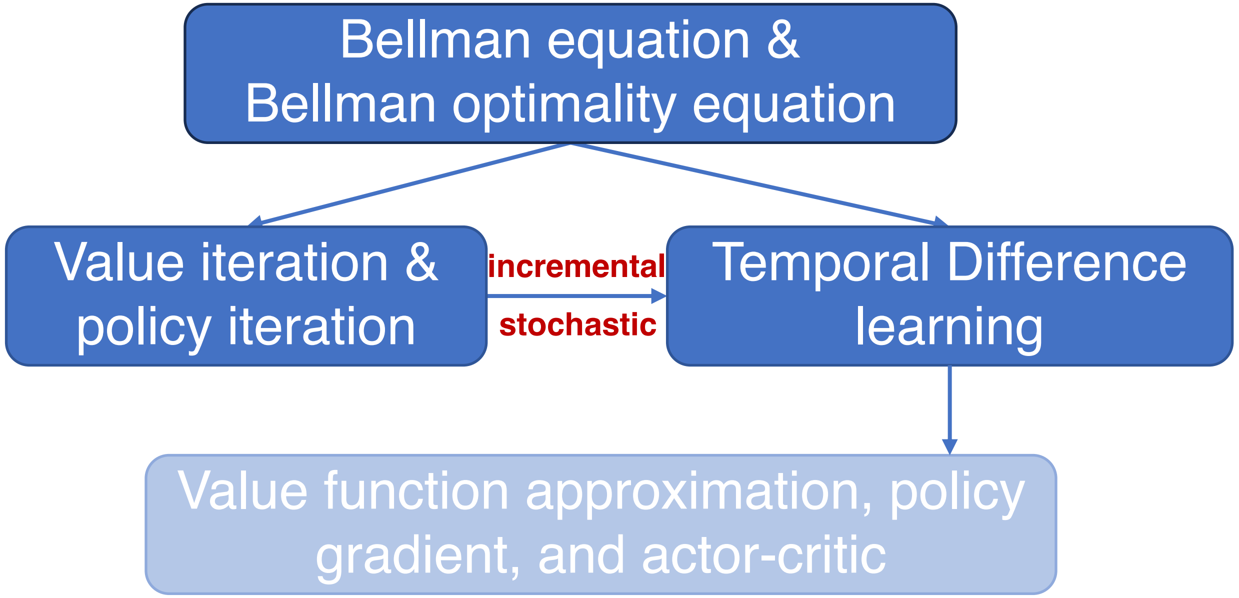
*The truncated policy iteration algorithm computes a finite number of iterations

Value iteration and policy iteration: Summary

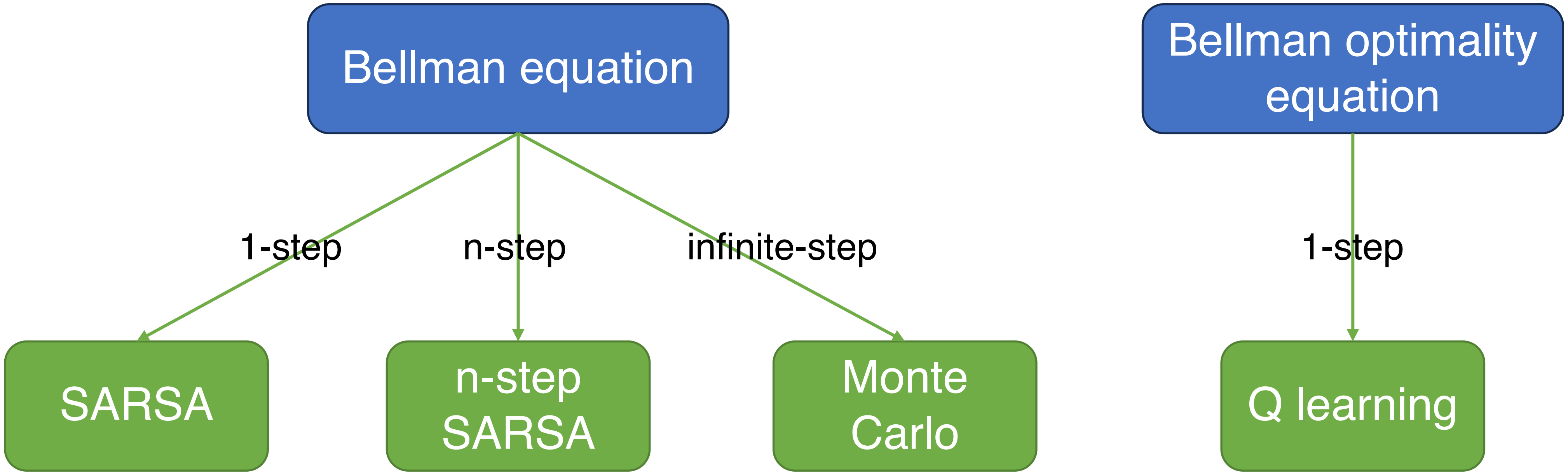


Outline

- 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:

$$v_{t+1}(s_t) = v_t(s_t) + \alpha_t(s_t)\Big[\big(r_{t+1} + \gamma v_t(s_{t+1})\big) - v_t(s_t)\Big]$$

- It uses boostrapping, 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:

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

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

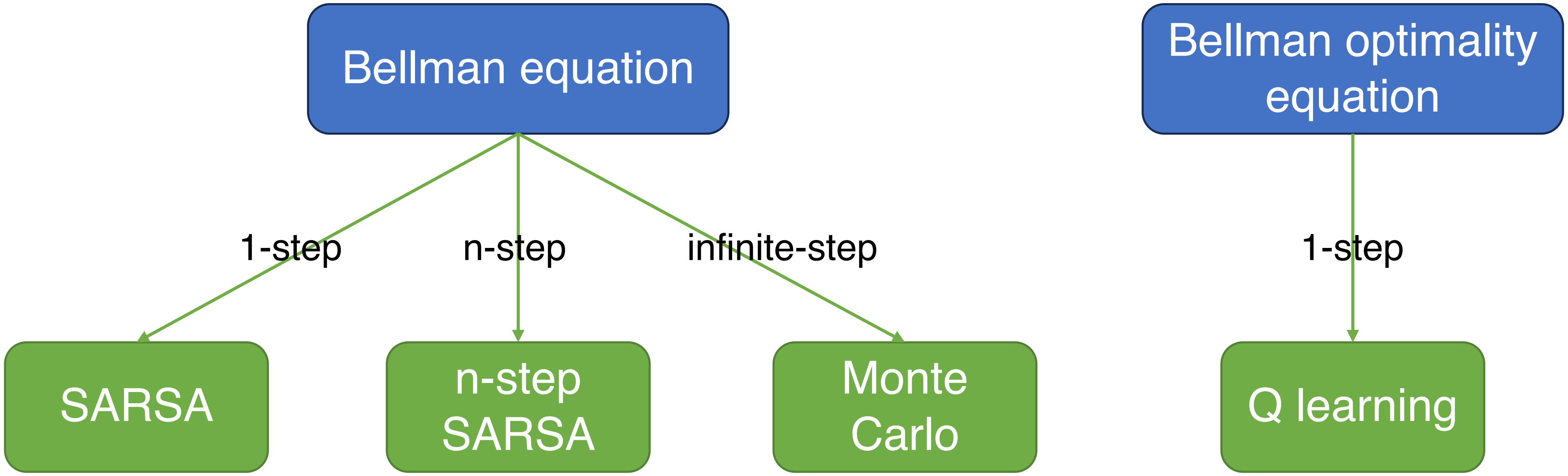
Other temporal difference learning: Unified view

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

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

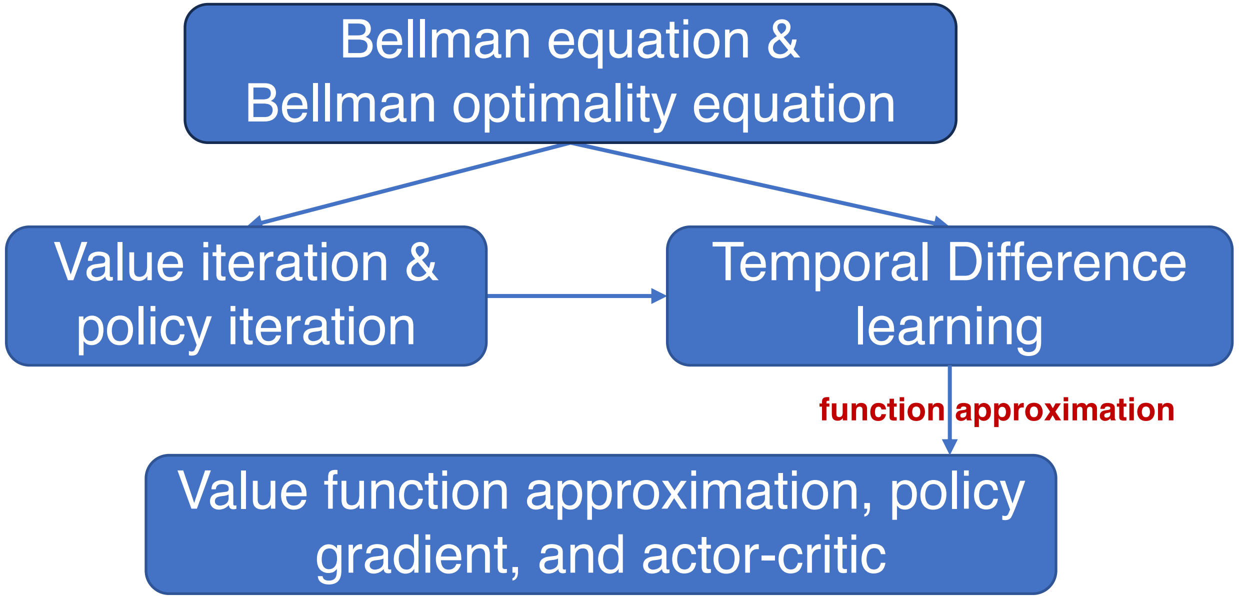
Algorithm	Equation and expression of $\bar{q}_t$
Sarsa	BE: $q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1},A_{t+1})\|S_t = s, A_t = a]$
	$\bar{q}_t = r_{t+1} + \gamma q(s_{t+1},a_{t+1})$
n-step Sarsa	BE: $q_\pi(s,a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^n q_\pi(S_{t+n},A_{t+n})\|S_t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

Temporal difference learning: Summary



Outline

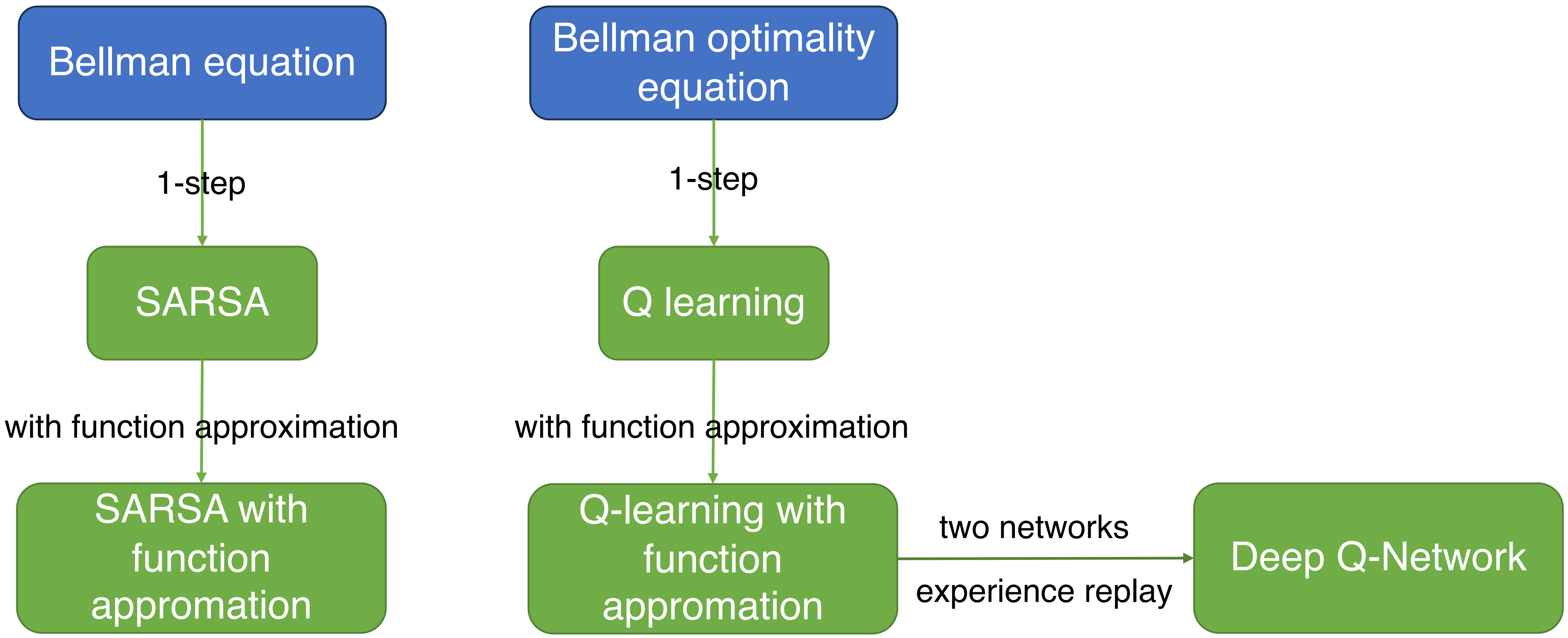
- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- Value function approximation, policy gradient methods, and actor-critic



Outline

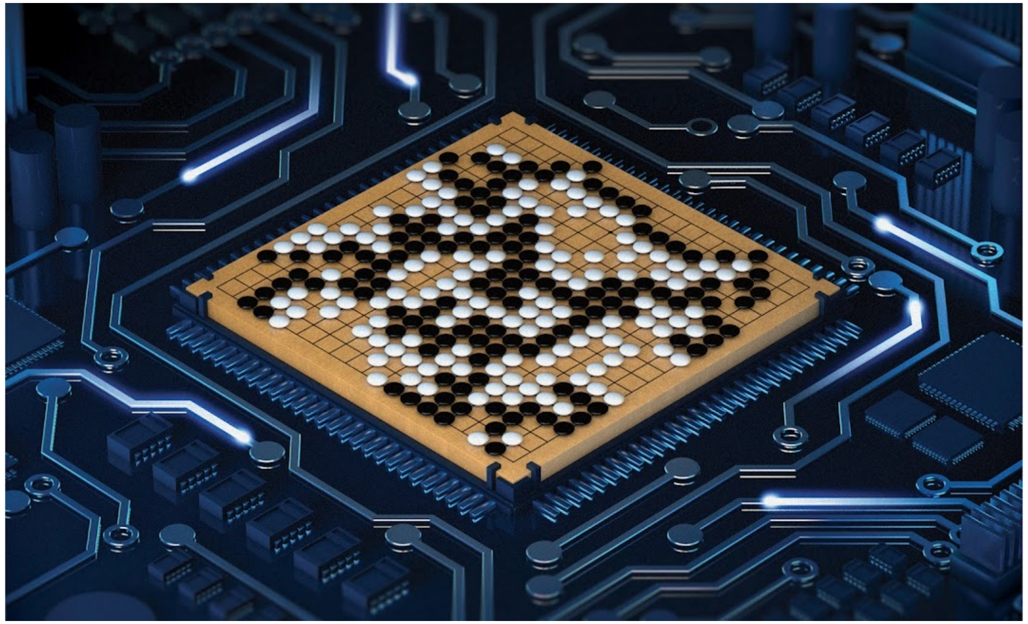
- Markov Decision Process (MDP) setup
- Bellman equation and Bellman optimality equation
- Value iteration and policy iteration
- Temporal difference learning
- Value function approximation, policy gradient methods, and actor-critic
  - Value function approximation: $\hat{v}(s; w)$
  - Policy gradient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

$$\underbrace{v_{t+1}(s_t)}_{\text{new estimate}} = \underbrace{v_t(s_t)}_{\text{current estimate}} + \alpha_t(s_t)\underbrace{\Big[\overbrace{\big(r_{t+1} + \gamma v_t(s_{t+1})\big)}^{\text{TD target}} - v_t(s_t)\Big]}_{\text{TD error}}$$

TD learning (with function approximation):

$$\underbrace{w_{t+1}}_{\text{new estimate}} = \underbrace{w_t}_{\text{current estimate}} + \alpha_t\underbrace{\Big[\overbrace{\big(r_{t+1} + \gamma \hat{v}(s_{t+1}; w_t)\big)}^{\text{TD target}} - \hat{v}(s_t; w_t)\Big]}_{\text{TD error}}\nabla_w \hat{v}(s_t; w_t)$$

SARSA with function approximation

SARSA (tabular):

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

SARSA (with function approximation):

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

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):

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

Policy update:

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

$\pi_{t+1}(a|s_t) = \frac{\varepsilon}{|\mathcal{A}(s)|}$ 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:

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

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:

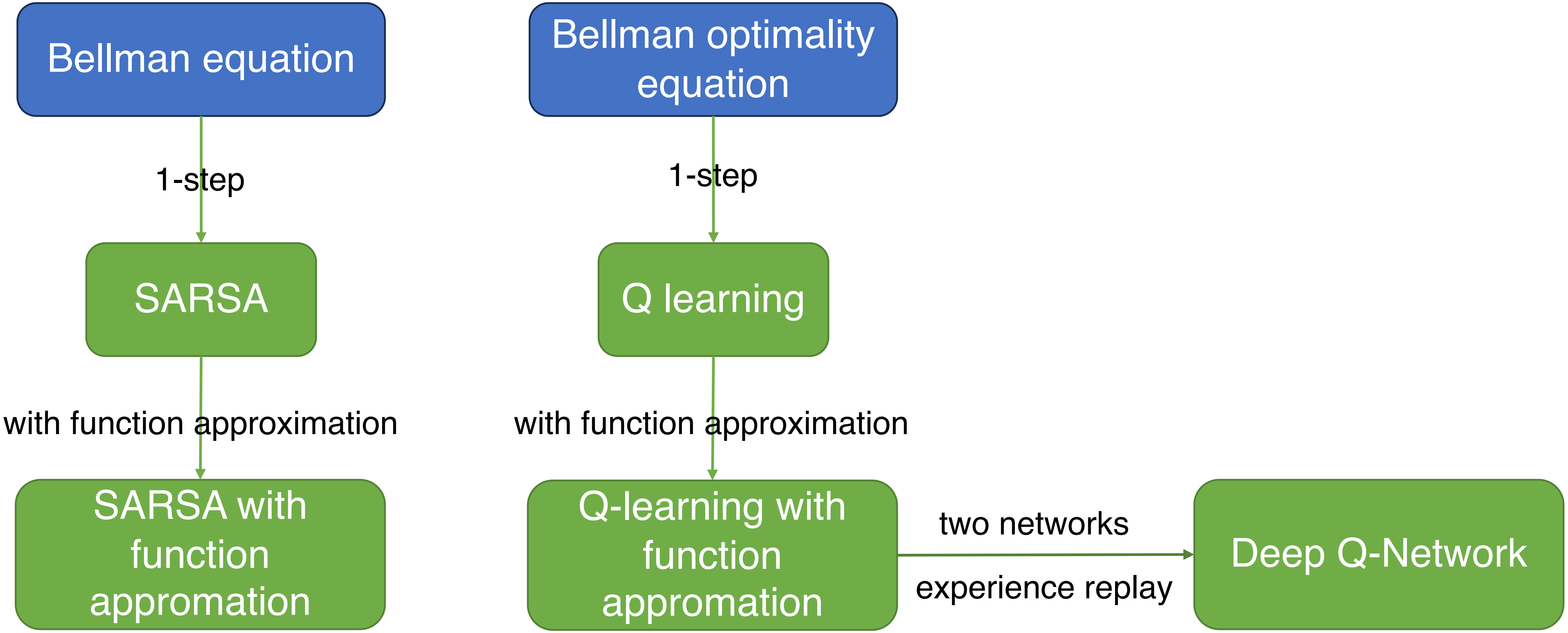
$$J(w) = \mathbb{E}\left[\left(R + \gamma \max_a \hat{q}(S_{t+1}, a; w_T) - \hat{q}(S_t, a; w)\right)^2\right]$$

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.

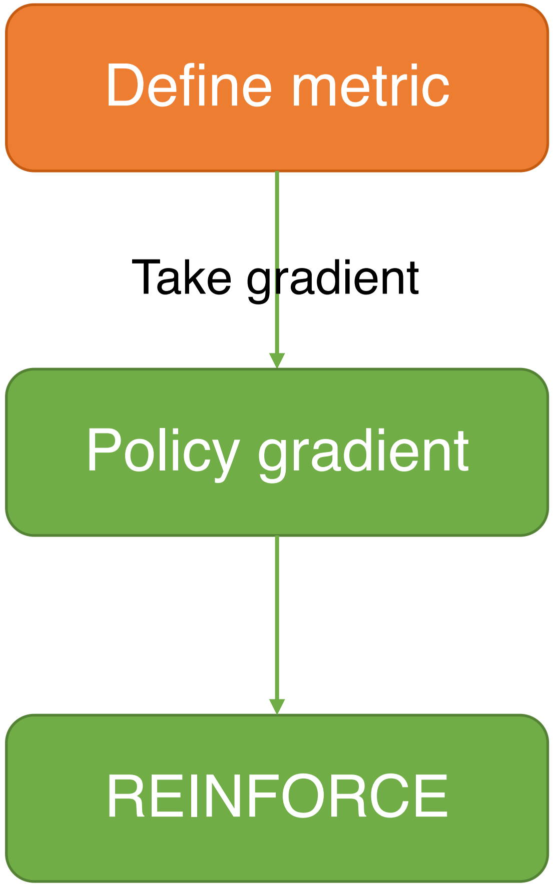
Value function approximation: Summary



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

Recall in Bellman Equation section section**:**

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

Policy gradient: Gradient expression

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

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 θ; $\gamma \in (0,1)$; $\alpha > 0$.
Goal: Learn an optimal policy to maximize $J(\theta)$.

For each episode, do
  Generate an episode $\{s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T\}$ following $\pi(\theta)$.
  For $t = 0, 1, \ldots, T-1$:
    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

TD learning with function approximation:

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

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

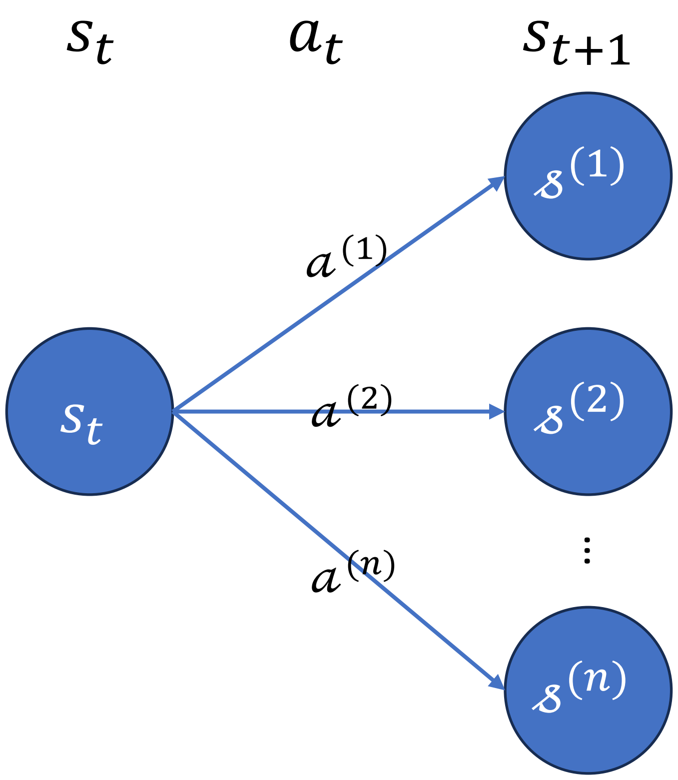
Value update: $w_{t+1} = w_t + \alpha_t\big[\big(r_{t+1} + \gamma \hat{v}(s_{t+1};w_t)\big) - \hat{v}(s_t;w_t)\big]\nabla_w \hat{v}(s_t;w_t)$

This value update is the same as in TD learning with function approximation

Actor-critic: Interpretation

Advantage δ_t

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



δ_t

0.5 — $\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi\left(a_t = a^{(1)} \middle| s_t;\theta_t\right) \cdot 0.5$ ↑

increase likelihood

-0.2 — $\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi\left(a_t = a^{(2)} \middle| s_t;\theta_t\right) \cdot (-0.2)$ ↓

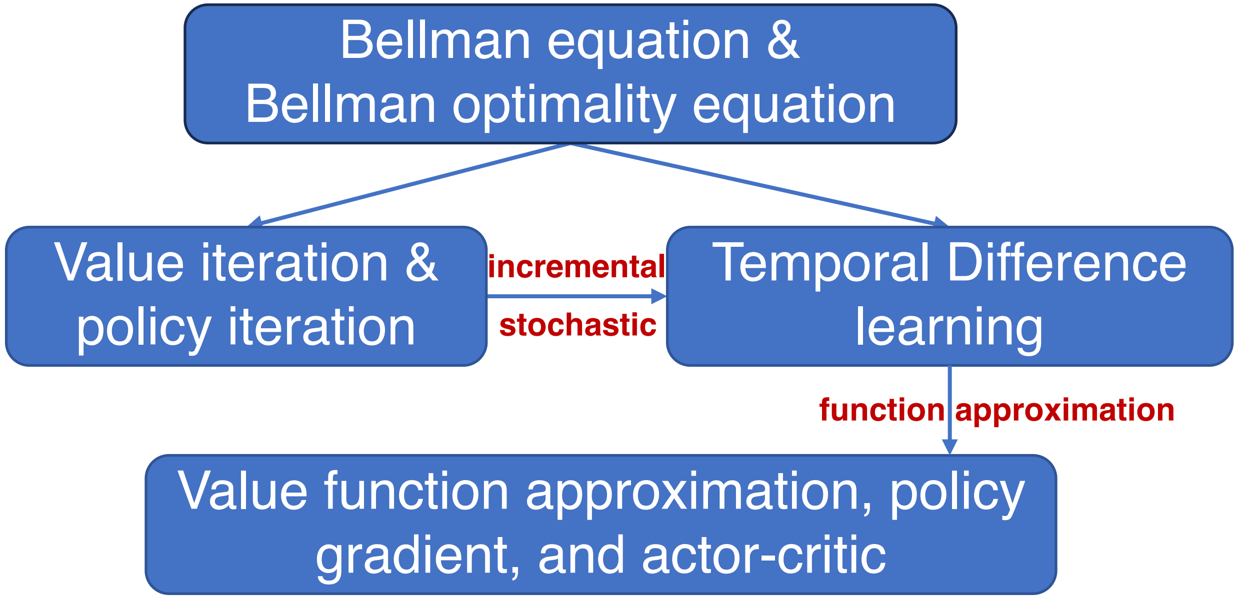
decrease likelihood

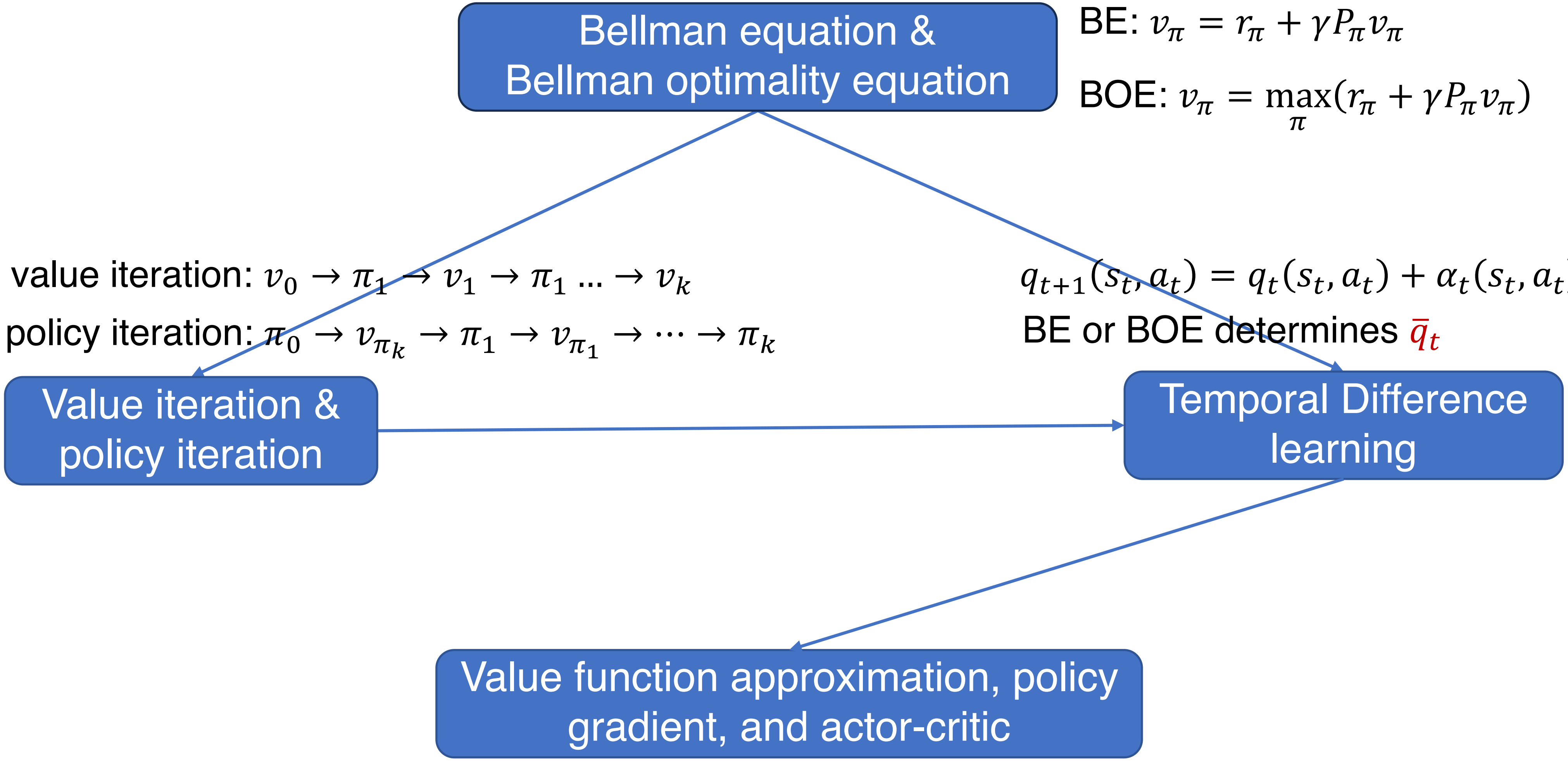
-1 — $\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi\left(a_t = a^{(n)} \middle| s_t;\theta_t\right) \cdot (-1)$ ↓

decrease likelihood

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





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

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

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

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

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

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

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

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

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-Mathmatical-Foundation-of-Reinforcement-Learning
- Useful code: https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail