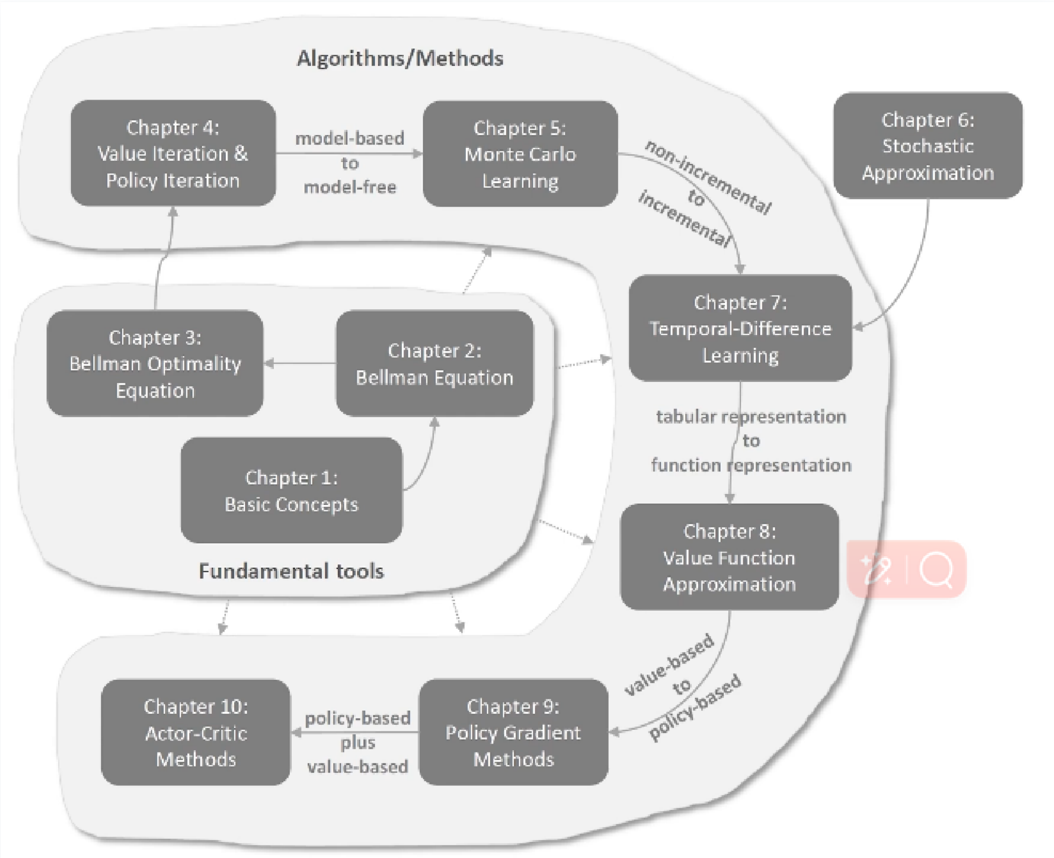
Reinforcement Learning

- [RL]https://www.bilibili.com/video/BV1r3411Q7Rr?vd_source=5002bb146c6f86977323df7568cda7ca



Basic Concepts

Chapter 1

- Concepts: state, action, reward, return, episode, policy,…
- Grid-world examples
- Markov decision process (MDP)
- Fundamental concepts, widely used later

- **State:** The status of the agent with respect to the environment (locations, ...)
- **State space:** the set of all states $\mathcal{S} = \{s_i\}$
- **Action:** For each state, actions that can be taken a_i
- **Action space of a state:** the set of all possible actions of a state $\mathcal{A}(s_i) = \{a_i\}$
- **state transition:** moving from one state to another, e.g. $s_1 \xrightarrow{a_i} s_2$

Forbidden area: At state s_5, if we choose action a_2, then what is the next state?

- Case 1: the forbidden area is accessible but with penalty. Then,

$$s_5 \xrightarrow{a_2} s_6$$

- Case 2: the forbidden area is inaccessible (e.g., surrounded by a wall)

$$s_5 \xrightarrow{a_2} s_5$$

We consider the first case, which is more general and challenging.

(Tabular representation, probability representation)

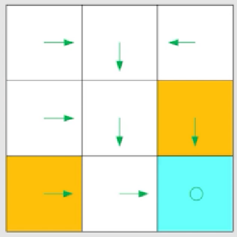
Math:

$$\begin{cases} p(s_2 \mid s_1, a_2) = 1 \\ p(s_i \mid s_1, a_2) = 0, \quad \forall i \neq 2 \end{cases}$$

The state transition could be **stochastic**

- **Policy:** tell the agent what actions to take at a state

Intuitive representation: The arrows demonstrate a policy.



Mathematical Representation Using conditional probability.

For example, for state s_1:

$$\begin{aligned}
\pi(a_1 \mid s_1) &= 0 \\
\pi(a_2 \mid s_1) &= 1 \\
\pi(a_3 \mid s_1) &= 0 \\
\pi(a_4 \mid s_1) &= 0 \\
\pi(a_5 \mid s_1) &= 0
\end{aligned}$$

It is a **deterministic** policy. In a stochastic situation, use sampling.

> **Policy** is how the agent chooses actions.
> **State transition** is how the environment responds to actions.
>
> The **agent** uses its **policy** to pick actions.
>
> The **environment** uses **state transitions** to return the next state.

- **Reward:** a real number we get after taking an action.

$$R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \quad r = R(s, a)$$

```
A **positive** reward represents **encouragement** to take such actions.
```

A **negative** reward represents **punishment** to take such actions.

Reward depends on the state and action but not the next state.

Mathematical Description: Conditional Probability

Intuition: At state s_1, if we choose action a_1, the reward is -1.

Math: $p(r = -1 \mid s_1, a_1) = 1$ and $p(r \neq -1 \mid s_1, a_1) = 0$ ....

- **trajectory:** a state-action-reward chain (evaluate whether a policy is good or not):

$$s_1 \xrightarrow[r=0]{a_2} s_2 \xrightarrow[r=0]{a_3} s_5 \xrightarrow[r=0]{a_3} s_8 \xrightarrow[r=1]{a_2} s_9$$

- **return:** the sum of all the rewards collected along the trajectory:

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

A trajectory may be infinite:

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_3} s_8 \xrightarrow{a_2} s_9 \xrightarrow{a_5} s_9 \xrightarrow{a_5} s_9 \cdots$$

The return is:

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

The definition is invalid since the return diverges!

- **discount rate** $\gamma \in [0, 1)$, **Discounted return**:

$$\begin{aligned} \text{discounted return} &= 0 + \gamma 0 + \gamma^2 0 + \gamma^3 1 + \gamma^4 1 + \gamma^5 1 + \cdots \\ &= \gamma^3(1 + \gamma + \gamma^2 + \cdots) = \gamma^3 \cdot \frac{1}{1-\gamma} \end{aligned}$$

Roles of discount factor γ:

1. The sum becomes finite.
2. It balances far and near future rewards.

If γ is close to 0, the value of the discounted return is dominated by the rewards obtained in the **near future**.

If γ is close to 1, the value of the discounted return is dominated by the rewards obtained in the **far future**.

- **episode:** usually assumed to be a **finite** trajectory. Tasks with episodes are called **episodic tasks**.

Key elements of Markov Decision Process (MDP):

- **Sets**:
  - **State**: the set of states $\mathcal{S}$
  - **Action**: the set of actions $\mathcal{A}(s)$ is associated with state $s \in \mathcal{S}$
  - **Reward**: the set of rewards $\mathcal{R}(s, a)$
- **Probability distribution**:

- **State transition probability**:
  At state s, taking action a, the probability to transition to state s' is

$$p(s' \mid s, a)$$

- **Reward probability**:
  At state s, taking action a, the probability to get reward r is

$$p(r \mid s, a)$$

- **Policy**:
  At state s, the probability to choose action a is

$$\pi(a \mid s)$$

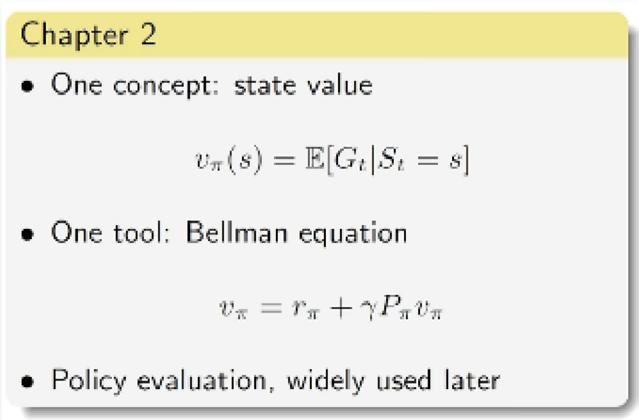
Markov property: memoryless property

The next state and reward depend **only on the current state and action**, not the full history:

$$p(s_{t+1} \mid a_{t+1}, s_t, \ldots, a_1, s_0) = p(s_{t+1} \mid a_{t+1}, s_t)$$

$$p(r_{t+1} \mid a_{t+1}, s_t, \ldots, a_1, s_0) = p(r_{t+1} \mid a_{t+1}, s_t)$$

Bellman Equation



Bellman Optimality Equation

Chapter 3

- A special Bellman equation
- Two concepts: optimal policy π^* & optimal state value
- One tool: Bellman optimality equation

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

1) Fixed-point theorem
2) Fundamental problems
3) An algorithm solving the equation

- Optimality, widely used later

Value Iteration & Policy Iteration

Chapter 4

- First algorithms for optimal policies
- Three algorithms:

  1) Value iteration (VI)

  2) Policy iteration (PI)

  3) Truncated policy iteration
- Policy update and value update, widely used later
- Need the environment model

Monta Carlo Learning

Chapter 5

- **Gap:** how to do model-free learning?
- Mean estimation with sampling data

$$\mathbb{E}[X] \approx \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- First model-free RL algorithms
- Algorithms:

  1) MC Basic

  2) MC Exploring Starts

  3) MC ϵ-greedy

Stochastic Approximation

Chapter 6

- **Gap:** from non-incremental to incremental
- Mean estimation
- Algorithms:

  1) Robbins-Monro (RM) algorithm
  2) Stochastic gradient descent (SGD)
  3) SGD, BGD, MBGD
- Incremental manner and SGD, widely used later

Temporal-Difference Learning

Chapter 7

- Classic RL algorithms
- Algorithms:

  1) TD learning of state values

  2) Sarsa: TD learning of action values

  3) Q-learning: TD learning of optimal action values

     - on-policy & off-policy

  4) Unified point of view

Value Function Approximation

Chapter 8

- **Gap:** tabular representation to function representation
- Algorithms:

  1) State value estimation with value function approximation (VFA):

  $$\min_{w} J(w) = \mathbb{E}[v_\pi(S) - \hat{v}(S, w)]$$

  2) Sarsa with VFA

  3) Q-learning with VFA

  4) Deep Q-learning
- Neural networks come into RL

Policy Gradient Method

Chapter 9

- **Gap:** from value-based to policy-based
- Contents:

  1) Metrics to define optimal policies:

  $$J(\theta) = \bar{v}_\pi, \bar{r}_\pi$$

  2) Policy gradient:

  $$\nabla J(\theta) = \mathbb{E}[\nabla_\theta \ln \pi(A|S,\theta) q_\pi(S,A)]$$

  3) Gradient-ascent algorithm (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 Method

Chapter 10

- **Gap:** policy-based + value-based

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

- Algorithms:

  1) The simplest actor-critic (QAC)

  2) Advantage actor-critic (A2C)

  3) Off-policy actor-critic

     - Importance sampling

  4) Deterministic actor-critic (DPG)