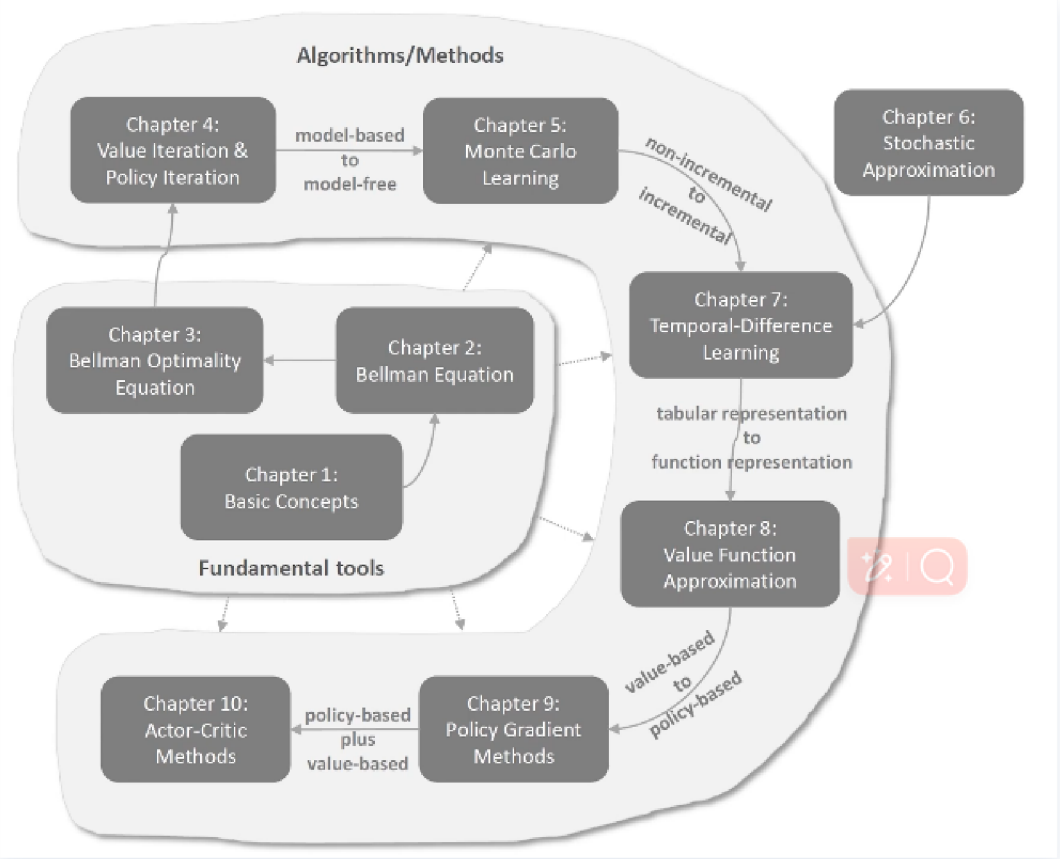
Reinforcement Learning

- [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.

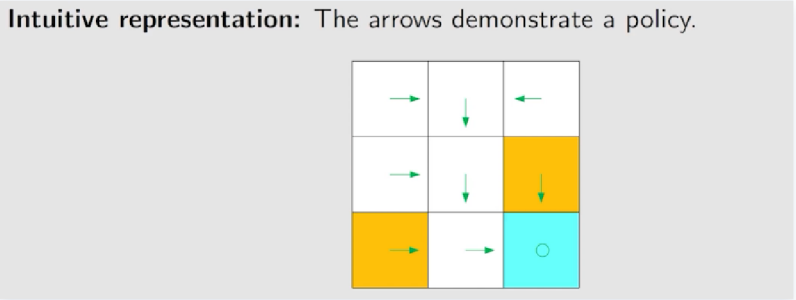
(Tabular representation, probability representation)

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



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.

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

```
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

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

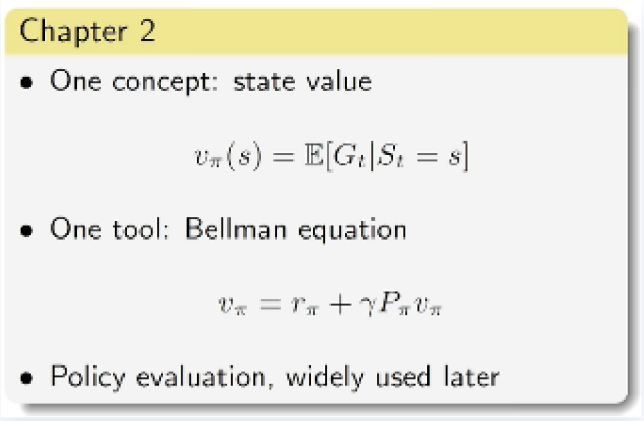
Markov property: memoryless property

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



Consider the following single-step process:

$$S_t \xrightarrow{A_t} R_{t+1},\ S_{t+1}$$

- $t, t+1$: discrete time instances
- S_t: state at time t
- A_t: the action taken at state S_t
- R_{t+1}: the reward obtained after taking A_t

- S_{t+1}: the state transited to after taking A_t

Note that S_t, A_t, R_{t+1} are all *random variables*.

This step is governed by the following probability distributions:

- $S_t \to A_t$ is governed by

$$\pi(A_t = a \mid S_t = s)$$

- $S_t, A_t \to R_{t+1}$ is governed by

$$p(R_{t+1} = r \mid S_t = s,\ A_t = a)$$

- $S_t, A_t \to S_{t+1}$ is governed by

$$p(S_{t+1} = s' \mid S_t = s,\ A_t = a)$$

- $S_t \to S_{t+1}$ is governed by (marginalizing over actions):

$$p(S_{t+1} = s' \mid S_t = s) = \sum_a \pi(a \mid s)\, p(s' \mid s, a)$$

> At this moment, we assume we know the model (i.e., the probability distributions)!

Consider the following multi-step trajectory:

$$S_t \xrightarrow{A_t} R_{t+1},\ S_{t+1} \xrightarrow{A_{t+1}} R_{t+2},\ S_{t+2} \xrightarrow{A_{t+2}} R_{t+3},\ \ldots$$

The discounted return is:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$$

- $\gamma \in [0, 1)$ is a discount rate.
- G_t is also a random variable since $R_{t+1}, R_{t+2}, \ldots$ are random variables.

State value

The expectation (or called expected value or mean) of G_t is defined as the *state-value function* or simply *state value*:

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

Remarks:

- It is a function of s. It is a conditional expectation with the condition that the state starts from s.
- It is based on the policy π. For a different policy, the state value may be different.
- It represents the "value" of a state. **If the state value is greater, then the policy is better because greater cumulative rewards can be obtained.**

Q: What is the relationship between **return** and **state value**?

A: The state value is the mean of all possible returns that can be obtained starting from a state. If everything — $\pi(a \mid s), p(r \mid s, a), p(s' \mid s, a)$ — is deterministic, then state value is the same as return.

Deriving the Bellman equation

Consider a random trajectory:

$$S_t \xrightarrow{A_t} R_{t+1},\ S_{t+1} \xrightarrow{A_{t+1}} R_{t+2},\ S_{t+2} \xrightarrow{A_{t+2}} R_{t+3},\ \dots$$

The return G_t can be written as:

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

Then, it follows from the definition of the state value that:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} \mid S_t = s] + \gamma \mathbb{E}[G_{t+1} \mid S_t = s] \end{aligned}$$

$$\begin{aligned} \mathbb{E}[R_{t+1} \mid S_t = s] &= \sum_a \pi(a \mid s)\, \mathbb{E}[R_{t+1} \mid S_t = s,\ A_t = a] \\ &= \sum_a \pi(a \mid s) \sum_r p(r \mid s, a)\, r \end{aligned}$$

Note that:

- This is the mean of *immediate rewards*.

$$\begin{aligned}\mathbb{E}[G_{t+1} \mid S_t = s] &= \sum_{s'} \mathbb{E}[G_{t+1} \mid S_t = s,\ S_{t+1} = s'] \cdot p(s' \mid s) \\ &= \sum_{s'} \mathbb{E}[G_{t+1} \mid S_{t+1} = s'] \cdot p(s' \mid s) \\ &= \sum_{s'} v_\pi(s') \cdot p(s' \mid s) \\ &= \sum_{s'} v_\pi(s') \sum_{a} p(s' \mid s, a) \cdot \pi(a \mid s)\end{aligned}$$

Note that:

- This is the **mean of future rewards**.
- $\mathbb{E}[G_{t+1} \mid S_t = s,\ S_{t+1} = s'] = \mathbb{E}[G_{t+1} \mid S_{t+1} = s']$ due to the **memoryless Markov property**.

Bellman Expectation Equation for State Value

$$\begin{aligned}v_\pi(s) &= \mathbb{E}[R_{t+1} \mid S_t = s] + \gamma \mathbb{E}[G_{t+1} \mid S_t = s] \\ &= \underbrace{\sum_a \pi(a \mid s) \sum_r p(r \mid s, a) r}_{\text{mean of immediate rewards}} + \underbrace{\gamma \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) v_\pi(s')}_{\text{mean of future rewards}} \\ &= \sum_a \pi(a \mid s) \left[\sum_r p(r \mid s, a) r + \gamma \sum_{s'} p(s' \mid s, a) v_\pi(s') \right], \quad \forall s \in \mathcal{S}\end{aligned}$$

Highlights:

- The above equation is called the *Bellman equation*, which characterizes the relationship among the **state-value functions** of different states.
- which is depended on policy. Solving the equation is call **policy evaluation.**
- $p(r \mid s, a)$ and $p(s' \mid s, a)$ represent the **dynamic model**.

Matrix-vector form of the Bellman equation

Recall that:

$$\begin{aligned}v_\pi(s) &= \mathbb{E}[R_{t+1} \mid S_t = s] + \gamma \mathbb{E}[G_{t+1} \mid S_t = s] \\ &= \sum_a \pi(a \mid s) \left[\sum_r p(r \mid s, a) r + \gamma \sum_{s'} p(s' \mid s, a) v_\pi(s') \right]\end{aligned}$$

Rewrite the Bellman equation as:

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

where

$$r_\pi(s) \triangleq \sum_a \pi(a \mid s) \sum_r p(r \mid s, a) r,$$

$$p_\pi(s' \mid s) \triangleq \sum_a \pi(a \mid s) p(s' \mid s, a)$$

Suppose the states could be indexed as s_i $(i = 1, \ldots, n)$. For state s_i, the Bellman equation is:

$$v_\pi(s_i) = r_\pi(s_i) + \gamma \sum_{s_j} p_\pi(s_j \mid s_i) v_\pi(s_j)$$

Put all these equations for all the states together and rewrite to a matrix-vector form:

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

---

where

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

If there are four states, $v_\pi = r_\pi + \gamma P_\pi v_\pi$ can be written out as:

$$\underbrace{\begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ v_\pi(s_3) \\ v_\pi(s_4) \end{bmatrix}}_{v_\pi} = \underbrace{\begin{bmatrix} r_\pi(s_1) \\ r_\pi(s_2) \\ r_\pi(s_3) \\ r_\pi(s_4) \end{bmatrix}}_{r_\pi} + \gamma \underbrace{\begin{bmatrix} p_\pi(s_1 \mid s_1) & p_\pi(s_2 \mid s_1) & p_\pi(s_3 \mid s_1) & p_\pi(s_4 \mid s_1) \\ p_\pi(s_1 \mid s_2) & p_\pi(s_2 \mid s_2) & p_\pi(s_3 \mid s_2) & p_\pi(s_4 \mid s_2) \\ p_\pi(s_1 \mid s_3) & p_\pi(s_2 \mid s_3) & p_\pi(s_3 \mid s_3) & p_\pi(s_4 \mid s_3) \\ p_\pi(s_1 \mid s_4) & p_\pi(s_2 \mid s_4) & p_\pi(s_3 \mid s_4) & p_\pi(s_4 \mid s_4) \end{bmatrix}}_{P_\pi} \underbrace{\begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ v_\pi(s_3) \\ v_\pi(s_4) \end{bmatrix}}_{v_\pi}$$

The closed-form solution is:

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

An *iterative solution* is:

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

This algorithm leads to a sequence:

$$\{v_0, v_1, v_2, \ldots\}$$

We can show that:

$$v_k \to v_\pi = (I - \gamma P_\pi)^{-1} r_\pi, \quad \text{as } k \to \infty$$

Proof.

Define the error as $\delta_k = v_k - v_\pi$. We only need to show $\delta_k \to 0$. Substituting $v_{k+1} = \delta_{k+1} + v_\pi$ and $v_k = \delta_k + v_\pi$ into $v_{k+1} = r_\pi + \gamma P_\pi v_k$ gives

$$\delta_{k+1} + v_\pi = r_\pi + \gamma P_\pi (\delta_k + v_\pi),$$

which can be rewritten as

$$\delta_{k+1} = -v_\pi + r_\pi + \gamma P_\pi \delta_k + \gamma P_\pi v_\pi = \gamma P_\pi \delta_k.$$

As a result,

$$\delta_{k+1} = \gamma P_\pi \delta_k = \gamma^2 P_\pi^2 \delta_{k-1} = \cdots = \gamma^{k+1} P_\pi^{k+1} \delta_0.$$

Note that $0 \leq P_\pi^k \leq 1$, which means every entry of P_π^k is no greater than 1 for any $k = 0, 1, 2, \ldots$. That is because $P_\pi^k \mathbf{1} = \mathbf{1}$, where $\mathbf{1} = [1, \ldots, 1]^T$. On the other hand, since $\gamma < 1$, we know $\gamma^k \to 0$ and hence $\delta_{k+1} = \gamma^{k+1} P_\pi^{k+1} \delta_0 \to 0$ as $k \to \infty$. □

> 本质是 $T(v) = r_\pi + \gamma P_\pi v$ 的 T 是 contracted mapping.

From State Value to Action Value

- **State value**: The average return the agent can get *starting from a state*.

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

- **Action value**: The average return the agent can get *starting from a state and taking an action*.

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

  - $q_\pi(s, a)$ is a function of the state-action pair (s, a)

From Conditional Expectation:

Using the law of total expectation, we get:

$$\mathbb{E}[G_t \mid S_t = s] = \sum_a \mathbb{E}[G_t \mid S_t = s, A_t = a] \cdot \pi(a \mid s)$$

This shows that the **state-value function** $v_\pi(s)$ is a weighted average of the action-value function $q_\pi(s, a)$ over the policy π.

Recall that the state value is given by:

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

we have the **action-value function** as:

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

Key concepts and results:

- State value: $v_\pi(s) = \mathbb{E}[G_t | S_t = s]$
- Action value: $q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$
- The Bellman equation (elementwise form):

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) \Big[\underbrace{\sum_r p(r|s,a) r + \gamma \sum_{s'} p(s'|s,a) v_\pi(s')}_{q_\pi(s,a)} \Big] \\ &= \sum_a \pi(a|s) q_\pi(s,a) \end{aligned}$$

- The Bellman equation (matrix-vector form):

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

- How to solve the Bellman equation: closed-form solution, iterative solution

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

Optimal Policy

The state value could be used to evaluate if a policy is good or not: If

$$v_{\pi_1}(s) \geq v_{\pi_2}(s) \quad \text{for all } s \in \mathcal{S}$$

then π_1 is "better" than π_2.

Definition

A policy π^* is **optimal** if

$$v_{\pi^*}(s) \geq v_\pi(s) \quad \text{for all } s \text{ and for any other policy } \pi.$$

Bellman Optimality Equation (Elementwise Form)

$$v(s) = \max_{\pi} \sum_a \pi(a \mid s) \left(\sum_r p(r \mid s,a) r + \gamma \sum_{s'} p(s' \mid s,a) v(s') \right), \quad \forall s \in \mathcal{S}$$

$$= \max_{\pi} \sum_a \pi(a \mid s) q(s,a), \quad s \in \mathcal{S}$$

Remarks

- $p(r \mid s,a),\ p(s' \mid s,a)$ are **known**.
- $v(s),\ v(s')$ are **unknown** and **to be calculated**.

Bellman Optimality Equation (Matrix-Vector Form)

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

where the elements corresponding to (s) or (s') are:

$$[r_\pi]_s \triangleq \sum_a \pi(a \mid s) \sum_r p(r \mid s,a) r,$$

$$[P_\pi]_{s,s'} = p(s' \mid s) \triangleq \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s,a)$$

Maximization on the right-hand side of BOE

Fix $v'(s)$ first and solve π:

$$\begin{aligned} v(s) &= \max_{\pi} \sum_a \pi(a \mid s) \left(\sum_r p(r \mid s,a) r + \gamma \sum_{s'} p(s' \mid s,a) v(s') \right), \quad \forall s \in \mathcal{S} \\ &= \max_{\pi} \sum_a \pi(a \mid s) q(s,a) \end{aligned}$$

considering that $\sum_a \pi(a \mid s) = 1$, we have

$$\max_{\pi} \sum_a \pi(a \mid s) q(s,a) = \max_{a \in \mathcal{A}(s)} q(s,a),$$

where the optimality is achieved when

$$\pi(a \mid s) = \begin{cases} 1 & a = a^* \\ 0 & a \neq a^* \end{cases}$$

where $a^* = \arg\max_a q(s,a)$.

Solve the Bellman optimality equation

The BOE is $v = \max_\pi (r_\pi + \gamma P_\pi v)$. Let

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

Then, the Bellman optimality equation becomes

$$v = f(v)$$

where

$$[f(v)]_s = \max_\pi \sum_a \pi(a \mid s) q(s,a), \quad s \in \mathcal{S}$$

Theorem (Contraction Property)

$f(v)$ is a *contraction mapping* satisfying

$$\|f(v_1) - f(v_2)\| \leq \gamma \|v_1 - v_2\|$$

where γ is the discount rate!

Consider any two vectors $v_1, v_2 \in \mathbb{R}^{|\mathcal{S}|}$, and suppose that $\pi_1^* \doteq \arg\max_\pi (r_\pi + \gamma P_\pi v_1)$ and $\pi_2^* \doteq \arg\max_\pi (r_\pi + \gamma P_\pi v_2)$. Then,

$$f(v_1) = \max_\pi (r_\pi + \gamma P_\pi v_1) = r_{\pi_1^*} + \gamma P_{\pi_1^*} v_1 \geq r_{\pi_2^*} + \gamma P_{\pi_2^*} v_1,$$
$$f(v_2) = \max_\pi (r_\pi + \gamma P_\pi v_2) = r_{\pi_2^*} + \gamma P_{\pi_2^*} v_2 \geq r_{\pi_1^*} + \gamma P_{\pi_1^*} v_2,$$

where $\geq$ is an elementwise comparison. As a result,

$$\begin{aligned} f(v_1) - f(v_2) &= r_{\pi_1^*} + \gamma P_{\pi_1^*} v_1 - (r_{\pi_2^*} + \gamma P_{\pi_2^*} v_2) \\ &\leq r_{\pi_1^*} + \gamma P_{\pi_1^*} v_1 - (r_{\pi_1^*} + \gamma P_{\pi_1^*} v_2) \\ &= \gamma P_{\pi_1^*} (v_1 - v_2). \end{aligned}$$

Theorem (Existence, Uniqueness, and Algorithm)

Applying the contraction mapping theorem gives the following results.

For the BOE

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

there always exists a solution v^* *and the solution is* **unique**. The solution could be solved iteratively by

$$v_{k+1} = f(v_k) = \max_\pi (r_\pi + \gamma P_\pi v_k)$$

This sequence $\{v_k\}$ converges to v^* **exponentially fast** given any initial guess v_0.

- **Solving** π^*: Once the value of v^* has been obtained, we can easily obtain π^* by solving

$$\pi^* = \arg\max_{\pi \in \Pi}(r_\pi + \gamma P_\pi v^*).$$

The value of π^* will be given in Theorem 3.5. Substituting above into the BOE yields

$$v^* = r_{\pi^*} + \gamma P_{\pi^*} v^*.$$

Therefore, $v^* = v_{\pi^*}$ is the state value of π^*, and the BOE is a special Bellman equation whose corresponding policy is π^*.

Theorem (Policy Optimality)

Suppose that v^* is the unique solution to

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

and v_π is the state value function satisfying

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

for any given policy π, then

$$v^* \geq v_\pi, \quad \forall \pi$$

Theorem (Greedy Optimal Policy)

For any $s \in \mathcal{S}$, the deterministic greedy policy

$$\pi^*(a \mid s) = \begin{cases} 1 & a = a^*(s) \\ 0 & a \neq a^*(s) \end{cases}$$

is an optimal policy solving the BOE. Here,

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

where

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

Proof: simple.

$$\pi^*(s) = \arg\max_\pi \sum_a \pi(a \mid s) \left(\underbrace{\sum_r p(r \mid s, a) r + \gamma \sum_{s'} p(s' \mid s, a) v^*(s')}_{q^*(s,a)} \right)$$

Theorem (Optimal Policy Invariance)

Consider a Markov decision process with $v^* \in \mathbb{R}^{|\mathcal{S}|}$ as the optimal state value satisfying

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

If every reward r is changed by an affine transformation to $ar + b$, where $a, b \in \mathbb{R}$ and $a \neq 0$, then the corresponding optimal state value v' is also an affine transformation of v^*:

$$v' = av^* + \frac{b}{1-\gamma}\mathbf{1},$$

where $\gamma \in (0, 1)$ is the discount rate and $\mathbf{1} = [1, \ldots, 1]^T$.

Consequently, the optimal policies are invariant to the affine transformation of the reward signals.

📌 总结：影响最优策略的因素

BOE 的逐元素形式表明，**最优状态值函数** $v(s)$ 和 **最优策略** π^* 取决于以下三个主要因素：

1. **即时奖励** r
2. **折扣因子** γ
3. **系统模型** $p(s'|s,a)$ 与 $p(r|s,a)$（即环境的转移概率和奖励分布）

🎯 折扣因子 γ 的影响

- **较大 γ（如 0.9）**：策略更具“远见”。即使短期有损失（如进入惩罚区域），只要长期回报高，也会选择这样做。
- **较小 γ（如 0.5）**：策略更“短视”，倾向于避免即时惩罚，即使因此导致路径更长。
- **极端情况 $\gamma = 0$**：策略完全忽略未来，仅追求即时奖励。这时策略无法“规划”路径，仅选取当下最优动作。

👉 折扣因子鼓励智能体尽快完成目标（如到达目标区），即使没有负奖励，策略也不会走无意义的绕路（detour）。

💰 奖励函数的影响

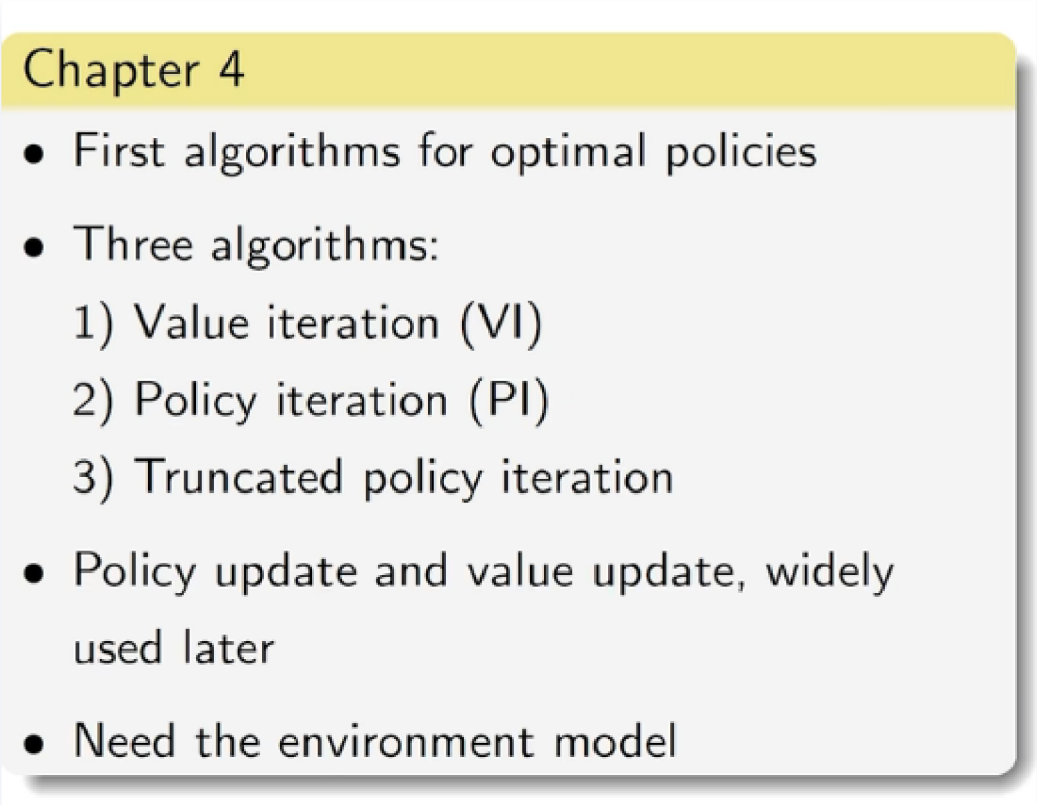
- 奖励值越大，状态值越高。
- 如果希望策略完全避免某些区域（如“禁区”），可通过**加大惩罚值（如将 $r_{\text{forbidden}}$ 从 -1 改为 -10）** 实现。

- **策略对奖励仿射变换保持不变**.

🚫 关于负步长奖励的误解

- 初学者常认为需要对每一步施加负奖励（如 -1）来"鼓励"尽快到达目标。
- 实际上：
  - 即使 $r_{\text{other}} = 0$，折扣因子 γ 也已隐含这种"快速完成任务"的动机。
  - 给所有奖励加同一个常数值只会平移状态值函数，不会改变策略。

Value Iteration & Policy Iteration



🔁 Value Iteration Algorithm

算法迭代式：

$$v_{k+1} = f(v_k) = \max_{\pi}(r_\pi + \gamma P_\pi v_k), \quad k = 1, 2, 3, \ldots$$

◆ Step 1: Policy Update（策略更新）

在当前值函数 v_k 给定的条件下，更新策略为：

$$\pi_{k+1} = \arg\max_{\pi}(r_\pi + \gamma P_\pi v_k)$$

这一步是“贪婪策略更新”（greedy w.r.t. v_k）。

◆ Step 2: Value Update（值函数更新）

然后使用新的策略 π_{k+1} 来更新值函数：

$$v_{k+1} = r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k$$

注意，这里 **不是** 直接解 Bellman 方程，而是前向推进一步。

❓ 问题：v_k 是状态值函数吗?

> 答：**不是**，因为它**不一定满足 Bellman 方程**。

换言之，只有当 $v_k \to v^*$ 收敛时，才是最优状态值函数，才满足：

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

▷ Step 1: Policy update

The elementwise form of

$$\pi_{k+1} = \arg\max_{\pi} (r_\pi + \gamma P_\pi v_k)$$

is

$$\pi_{k+1}(s) = \arg\max_{\pi} \sum_a \pi(a|s) \left(\sum_r p(r|s,a) r + \gamma \sum_{s'} p(s'|s,a) v_k(s') \right), \quad s \in \mathcal{S}$$

The optimal policy solving the above optimization problem is

$$\pi_{k+1}(a|s) = \begin{cases} 1, & a = a_k^*(s) \\ 0, & a \neq a_k^*(s) \end{cases}$$

where

$$a_k^*(s) = \arg\max_a q_k(a, s)$$

π_{k+1} is called a **greedy policy**, since it simply selects the greatest q-value.

▷ Step 2: Value update

The elementwise form of

$$v_{k+1} = r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k$$

is

$$v_{k+1}(s) = \sum_a \pi_{k+1}(a|s) \left(\underbrace{\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v_k(s')}_{q_k(s,a)} \right), \quad s \in \mathcal{S}$$

Since π_{k+1} is greedy, the above equation is simply

$$v_{k+1}(s) = \max_a q_k(a, s)$$

▷ Procedure summary:

$v_k(s) \rightarrow q_k(s,a) \rightarrow$ greedy policy $\pi_{k+1}(a|s) \rightarrow$ new value $v_{k+1} = \max_a q_k(s,a)$

Pseudocode: Value iteration algorithm

Initialization: The probability model $p(r|s,a)$ and $p(s'|s,a)$ for all (s,a) are known. Initial guess v_0.

Aim: Search the optimal state value and an optimal policy solving the Bellman optimality equation.

While v_k has not converged in the sense that $\|v_k - v_{k-1}\|$ is greater than a predefined small threshold, for the kth iteration, do

  For every state $s \in \mathcal{S}$, do

    For every action $a \in \mathcal{A}(s)$, do

      q-value: $q_k(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v_k(s')$

    Maximum action value: $a_k^*(s) = \arg\max_a q_k(a,s)$

    Policy update: $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$, and $\pi_{k+1}(a|s) = 0$ otherwise

    Value update: $v_{k+1}(s) = \max_a q_k(a,s)$

example

q-table	a_1	a_2	a_3	a_4	a_5
s_1	$-1+\gamma v(s_1)$	$-1+\gamma v(s_2)$	$0+\gamma v(s_3)$	$-1+\gamma v(s_1)$	$0+\gamma v(s_1)$
s_2	$-1+\gamma v(s_2)$	$-1+\gamma v(s_2)$	$1+\gamma v(s_4)$	$0+\gamma v(s_1)$	$-1+\gamma v(s_2)$
s_3	$0+\gamma v(s_1)$	$1+\gamma v(s_4)$	$-1+\gamma v(s_3)$	$-1+\gamma v(s_3)$	$0+\gamma v(s_3)$
s_4	$-1+\gamma v(s_2)$	$-1+\gamma v(s_4)$	$-1+\gamma v(s_4)$	$0+\gamma v(s_3)$	$1+\gamma v(s_4)$

Table 4.1: The expression of $q(s,a)$ for the example as shown in Figure 4.2.

ure 4.2). The target area is s_4. The reward settings are $r_{\text{boundary}} = r_{\text{forbidden}} = -1$ and $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$.

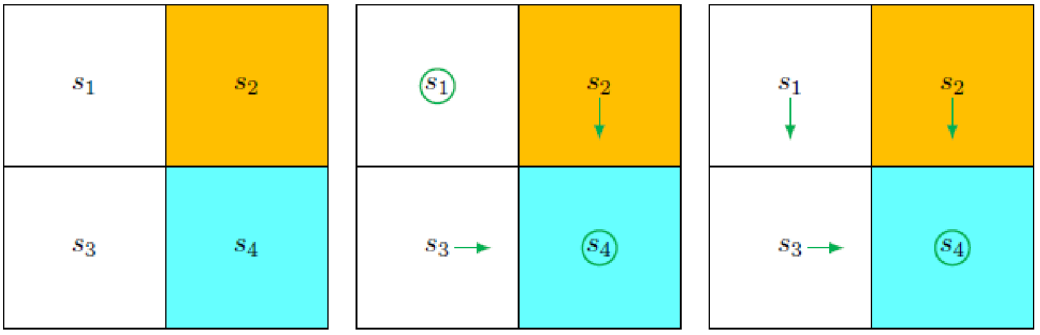


Figure 4.2: An example for demonstrating the implementation of the value iteration algorithm.

- $k = 0$: let $v_0(s_1) = v_0(s_2) = v_0(s_3) = v_0(s_4) = 0$

q-value	a_1	a_2	a_3	a_4	a_5
s_1	-1	-1	0	-1	0
s_2	-1	-1	1	0	-1
s_3	0	1	-1	-1	0
s_4	-1	-1	-1	0	1

Step 1: Policy update:

$$\pi_1(a_5|s_1) = 1, \pi_1(a_3|s_2) = 1, \pi_1(a_2|s_3) = 1, \pi_1(a_5|s_4) = 1$$

This policy is visualized in Figure (b).

Step 2: Value update:

$$v_1(s_1) = 0, v_1(s_2) = 1, v_1(s_3) = 1, v_1(s_4) = 1.$$

- $k = 1$: since $v_1(s_1) = 0, v_1(s_2) = 1, v_1(s_3) = 1, v_1(s_4) = 1$, we have

q-table	a_1	a_2	a_3	a_4	a_5
s_1	$-1+\gamma 0$	$-1+\gamma 1$	$0+\gamma 1$	$-1+\gamma 0$	$0+\gamma 0$
s_2	$-1+\gamma 1$	$-1+\gamma 1$	$1+\gamma 1$	$0+\gamma 0$	$-1+\gamma 1$
s_3	$0+\gamma 0$	$1+\gamma 1$	$-1+\gamma 1$	$-1+\gamma 1$	$0+\gamma 1$
s_4	$-1+\gamma 1$	$-1+\gamma 1$	$-1+\gamma 1$	$0+\gamma 1$	$1+\gamma 1$

  Step 1: Policy update:

$$\pi_2(a_3|s_1) = 1, \pi_2(a_3|s_2) = 1, \pi_2(a_2|s_3) = 1, \pi_2(a_5|s_4) = 1.$$

  Step 2: Value update:

$$v_2(s_1) = \gamma 1, v_2(s_2) = 1 + \gamma 1, v_2(s_3) = 1 + \gamma 1, v_2(s_4) = 1 + \gamma 1.$$

  This policy is visualized in Figure (c).

  The policy is already optimal!!

- $k = 2, 3, \ldots$. Stop when $\|v_k - v_{k+1}\|$ is smaller than a predefined threshold.

Policy iteration algorithm

▶ **Algorithm description:**

Given a random initial policy π_0,

- **Step 1: [policy evaluation (PE)]**

  This step is to calculate the state value of π_k:

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

  Note that v_{π_k} is a state value function.

- **Step 2: [policy improvement (PI)]**

$$\pi_{k+1} = \arg\max_{\pi} (r_\pi + \gamma P_\pi v_{\pi_k})$$

  The maximization is componentwise!

Step 1: Policy evaluation

- ▶ **Matrix-vector form**:

$$v_{\pi_k}^{(j+1)} = r_{\pi_k} + \gamma P_{\pi_k} v_{\pi_k}^{(j)}, \quad j = 0, 1, 2, \ldots$$

- ▶ **Elementwise form**:

$$v_{\pi_k}^{(j+1)}(s) = \sum_a \pi_k(a \mid s) \left(\sum_r p(r \mid s, a) r + \gamma \sum_{s'} p(s' \mid s, a) v_{\pi_k}^{(j)}(s') \right), \quad s \in \mathcal{S}$$

Step 2: Policy improvement

- ▶ **Matrix-vector form**:

$$\pi_{k+1} = \arg\max_{\pi} \left(r_\pi + \gamma P_\pi v_{\pi_k} \right)$$

- ▶ **Elementwise form**:

$$\pi_{k+1}(s) = \arg\max_{\pi} \sum_a \pi(a \mid s) \left(\sum_r p(r \mid s, a) r + \gamma \sum_{s'} p(s' \mid s, a) v_{\pi_k}(s') \right), \quad s \in \mathcal{S}$$

- Here, $q_{\pi_k}(s, a)$ is the action value under policy π_k. Let

$$a_k^*(s) = \arg\max_a q_{\pi_k}(a, s)$$

- Then, the **greedy policy** is:

$$\pi_{k+1}(a \mid s) = \begin{cases} 1, & a = a_k^*(s) \\ 0, & a \neq a_k^*(s) \end{cases}$$

Pseudocode: Policy iteration algorithm

Initialization: The probability model $p(r|s,a)$ and $p(s'|s,a)$ for all (s,a) are known. Initial guess π_0.

Aim: Search for the optimal state value and an optimal policy.

While the policy has not converged, for the kth iteration, do

Policy evaluation:

Initialization: an arbitrary initial guess $v_{\pi_k}^{(0)}$

While $v_{\pi_k}^{(j)}$ has not converged, for the jth iteration, do

For every state $s \in \mathcal{S}$, do

$$v_{\pi_k}^{(j+1)}(s) = \sum_a \pi_k(a|s) \left[\sum_r p(r|s,a) r + \gamma \sum_{s'} p(s'|s,a) v_{\pi_k}^{(j)}(s') \right]$$

Policy improvement:

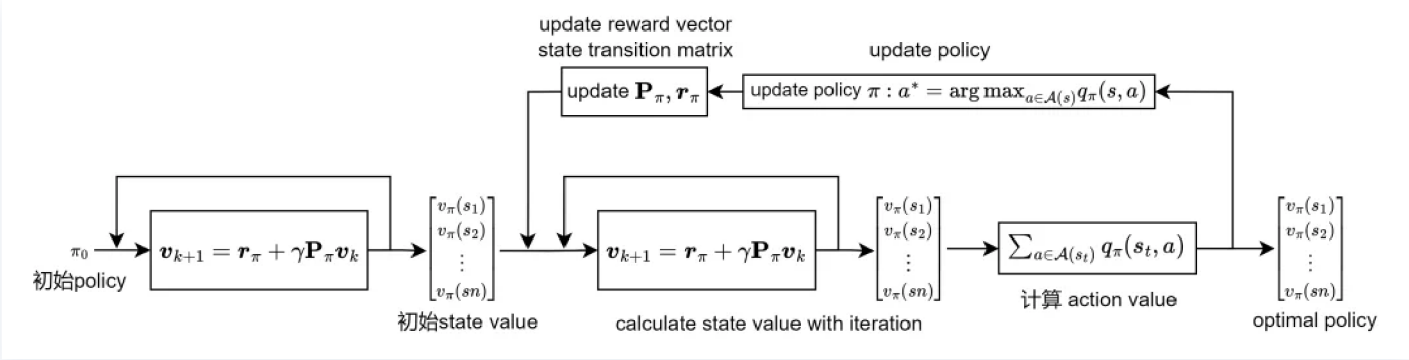
For every state $s \in \mathcal{S}$, do

For every action $a \in \mathcal{A}(s)$, do

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

$$a_k^*(s) = \arg\max_a q_{\pi_k}(s,a)$$

$$\pi_{k+1}(a|s) = 1 \text{ if } a = a_k^*, \text{ and } \pi_{k+1}(a|s) = 0 \text{ otherwise}$$



Compare value iteration and policy iteration

Policy iteration: start from π_0

- **Policy evaluation (PE)**:

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

- **Policy improvement (PI)**:

$$\pi_{k+1} = \arg\max_{\pi} \left(r_\pi + \gamma P_\pi v_{\pi_k}\right)$$

Value iteration: start from v_0

- **Policy update (PU)**:

$$\pi_{k+1} = \arg\max_{\pi} \left(r_\pi + \gamma P_\pi v_k\right)$$

- **Value update (VU)**:

$$v_{k+1} = r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k$$

> The two algorithms are very similar:

Policy iteration:

$$\pi_0 \xrightarrow{\text{PE}} v_{\pi_0} \xrightarrow{\text{PI}} \pi_1 \xrightarrow{\text{PE}} v_{\pi_1} \xrightarrow{\text{PI}} \pi_2 \xrightarrow{\text{PE}} v_{\pi_2} \xrightarrow{\text{PI}} \cdots$$

Value iteration:

$$u_0 \xrightarrow{\text{PU}} \pi_1' \xrightarrow{\text{VU}} u_1 \xrightarrow{\text{PU}} \pi_2' \xrightarrow{\text{VU}} u_2 \xrightarrow{\text{PU}} \cdots$$

Let's compare the steps carefully:

Step	Policy iteration algorithm	Value iteration algorithm	Comments
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	$\pi_1 = \arg\max_\pi (r_\pi + \gamma P_\pi v_0)$	$\pi_1 = \arg\max_\pi (r_\pi + \gamma P_\pi v_0)$	The two policies are the same
4) Value	$v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$	$v_1 = r_{\pi_1} + \gamma P_{\pi_1} v_0$	$v_{\pi_1} \geq v_1$ since $v_{\pi_1} \geq v_{\pi_0}$
5) Policy	$\pi_2 = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_1})$	$\pi_2' = \arg\max_\pi (r_\pi + \gamma P_\pi v_1)$	

Step	Policy iteration algorithm	Value iteration algorithm	Comments
⋮	⋮	⋮	⋮

Key observations:

- ✅ They start from the same initial condition. ✅ The first three steps are the same. ⚠ The fourth 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).

Consider the step of solving: $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$

Value Iteration:

Start from initial value:

$$v_{\pi_1}^{(0)} = v_0$$

One-step update:

$$v_{\pi_1}^{(1)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(0)} = v_1$$

Truncated Policy Iteration:

Iterate j steps:

$$\bar{v}_1 \leftarrow v_{\pi_1}^{(j)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(j-1)}$$

⋮

Policy Iteration:

Solve to convergence:

$$v_{\pi_1} \leftarrow v_{\pi_1}^{(\infty)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(\infty)}$$

Pseudocode: Truncated policy iteration algorithm

Initialization: The probability model $p(r|s,a)$ and $p(s'|s,a)$ for all (s,a) are known. Initial guess π_0.

Aim: Search for the optimal state value and an optimal policy.

While the policy has not converged, for the kth iteration, do

Policy evaluation:

Initialization: select the initial guess as $v_k^{(0)} = v_{k-1}$. The maximum iteration is set to be j_{truncate}.

While $j < j_{\text{truncate}}$, do

For every state $s \in \mathcal{S}$, do

$$v_k^{(j+1)}(s) = \sum_a \pi_k(a|s) \left[\sum_r p(r|s,a) r + \gamma \sum_{s'} p(s'|s,a) v_k^{(j)}(s') \right]$$

Set $v_k = v_k^{(j_{\text{truncate}})}$

Policy improvement:

For every state $s \in \mathcal{S}$, do

For every action $a \in \mathcal{A}(s)$, do

$$q_k(s,a) = \sum_r p(r|s,a) r + \gamma \sum_{s'} p(s'|s,a) v_k(s')$$

$a_k^*(s) = \arg\max_a q_k(s,a)$

$\pi_{k+1}(a|s) = 1$ if $a = a_k^*$, and $\pi_{k+1}(a|s) = 0$ otherwise

✅ 总结对比：

方法	特点
Value Iteration	只做一次 Bellman 更新
Truncated Policy Iteration	只迭代有限步 Bellman 更新
Policy Iteration	精确求解固定策略下的值函数（直到收敛）

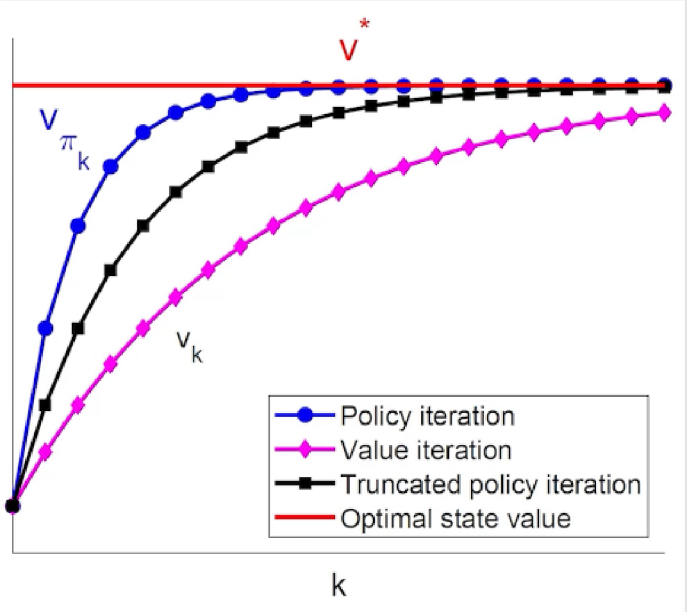


Figure: Illustration of the relationship among value iteration, policy iteration, and truncated policy iteration.

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)