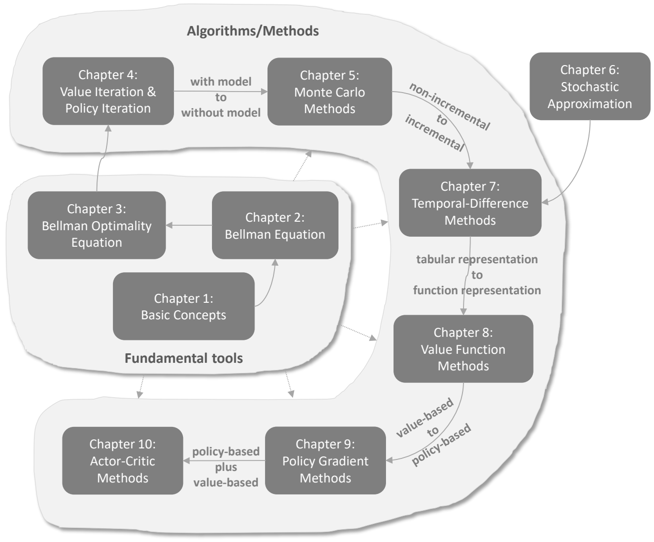
Lecture 4: Value Iteration and Policy Iteration

Shiyu Zhao

Department of Artificial Intelligence

Westlake University

Outline



Algorithms/Methods
Chapter 4: Value Iteration & Policy Iteration
with model to without model
Chapter 5: Monte Carlo Methods
non-incremental to incremental
Chapter 6: Stochastic Approximation
Chapter 7: Temporal-Difference Methods
tabular representation to function representation
Chapter 8: Value Function Methods
value-based to policy-based
Chapter 9: Policy Gradient Methods
policy-based plus value-based
Chapter 10: Actor-Critic Methods
Chapter 3: Bellman Optimality Equation
Chapter 2: Bellman Equation
Chapter 1: Basic Concepts
Fundamental tools

Outline

1. Value iteration algorithm
2. Policy iteration algorithm
3. Truncated policy iteration algorithm

Outline

1. Value iteration algorithm
2. Policy iteration algorithm
3. Truncated policy iteration algorithm

Value iteration algorithm

▷ How to solve the Bellman optimality equation?

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

▷ The **contraction mapping theorem** suggests an iterative algorithm:

$$v_{k+1} = f(v_k) = \max_{\pi}(r_\pi + \gamma P_\pi v_k), \quad k = 0, 1, 2, 3 \dots$$

where v_0 can be arbitrary. This algorithm can eventually find the optimal state value and an optimal policy.

▷ This algorithm is called the value iteration algorithm!

▷ We next study the **implementation details** of this algorithm.

Value iteration algorithm

- ▷ How to solve the Bellman optimality equation?

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

- ▷ The **contraction mapping theorem** suggests an iterative algorithm:

$$v_{k+1} = f(v_k) = \max_{\pi}(r_\pi + \gamma P_\pi v_k), \quad k = 0, 1, 2, 3 \ldots$$

where v_0 can be arbitrary. This algorithm can eventually find the optimal state value and an optimal policy.

- ▷ This algorithm is called the value iteration algorithm!
- ▷ We next study the **implementation details** of this algorithm.

Value iteration algorithm

- How to solve the Bellman optimality equation?

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

- The **contraction mapping theorem** suggests an iterative algorithm:

$$v_{k+1} = f(v_k) = \max_{\pi}(r_\pi + \gamma P_\pi v_k), \quad k = 0, 1, 2, 3 \dots$$

where v_0 can be arbitrary. This algorithm can eventually find the optimal state value and an optimal policy.

- This algorithm is called the value iteration algorithm!
- We next study the **implementation details** of this algorithm.

Value iteration algorithm

The algorithm (matrix-vector form)

$$v_{k+1} = f(v_k) = \max_\pi (r_\pi + \gamma P_\pi v_k), \quad k = 0, 1, 2, 3 \ldots$$

can be decomposed to two steps.

- Step 1: policy update. This step is to solve

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

  where v_k is given.

- Step 2: value update. This step is to compute the following equation:

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

Question: is v_k a state value?

Answer: No, because v_k may not satisfy any Bellman equation.

Value iteration algorithm

The algorithm (matrix-vector form)

$$v_{k+1} = f(v_k) = \max_\pi (r_\pi + \gamma P_\pi v_k), \quad k = 0, 1, 2, 3 \ldots$$

can be decomposed to two steps.

- Step 1: policy update. This step is to solve

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

  where v_k is given.

- Step 2: value update. This step is to compute the following equation:

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

Question: is v_k a state value?

Answer: No, because v_k may not satisfy any Bellman equation.

Value iteration algorithm

The algorithm (matrix-vector form)

$$v_{k+1} = f(v_k) = \max_\pi (r_\pi + \gamma P_\pi v_k), \quad k = 0, 1, 2, 3 \dots$$

can be decomposed to two steps.

- Step 1: policy update. This step is to solve

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

  where v_k is given.

- Step 2: value update. This step is to compute the following equation:

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

Question: is v_k a state value?

Answer: No, because v_k may not satisfy any Bellman equation.

Value iteration algorithm

The algorithm (matrix-vector form)

$$v_{k+1} = f(v_k) = \max_\pi (r_\pi + \gamma P_\pi v_k), \quad k = 0, 1, 2, 3 \ldots$$

can be decomposed to two steps.

- Step 1: policy update. This step is to solve

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

  where v_k is given.

- Step 2: value update. This step is to compute the following equation:

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

Question: is v_k a state value?

Answer: No, because v_k may not satisfy any Bellman equation.

▷ We next study the elementwise form in order to implement the algorithm.

- Matrix-vector form is useful for **theoretical analysis**.
- Elementwise form is useful for **implementation**.

▷ Step 1: Policy update

The elementwise form of

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

is

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

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.

Value iteration algorithm - Elementwise form

▷ Step 1: Policy update

The elementwise form of

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

is

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

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.

Value iteration algorithm - Elementwise form

▷ Step 2: Value update

The elementwise form of

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

is

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

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

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

Value iteration algorithm - Elementwise form

▷ Step 2: Value update

The elementwise form of

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

is

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

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

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

Value iteration algorithm - Pseudocode

▷ Procedure summary:

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

Value iteration algorithm - Pseudocode

▷ Procedure summary:

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

Pseudocode: Value iteration algorithm

Initialization: The probability models $p(r|s,a)$ and $p(s'|s,a)$ for all (s,a) are known. Initial guess v_0.
Aim: Search for the optimal state value and an optimal policy by 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

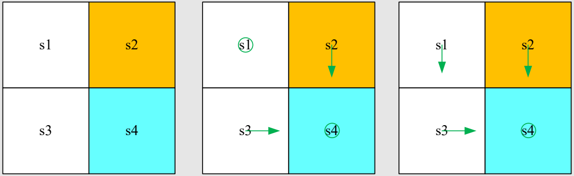
q-value: $q_k(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s')$

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

▷ The reward setting is $r_{\text{boundary}} = r_{\text{forbidden}} = -1$, $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$.



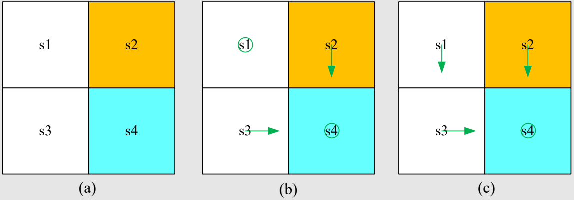
(a) (b) (c)

q-table: The expression of $q(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s')$:

q-value	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	$-1+\gamma v(s_2)$	$-1+\gamma v(s_4)$	$-1+\gamma v(s_4)$	$0+\gamma v(s_3)$	$1+\gamma v(s_4)$

Value iteration algorithm - Example

▷ The reward setting is $r_{\text{boundary}} = r_{\text{forbidden}} = -1$, $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$.



q-table: The expression of $q(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v_k(s')$:

q-value	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	$-1 + \gamma v(s_2)$	$-1 + \gamma v(s_4)$	$-1 + \gamma v(s_4)$	$0 + \gamma v(s_3)$	$1 + \gamma v(s_4)$

Value iteration algorithm - Example

- $k = 0$: 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:

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

Step 2: Value update:

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

This policy is visualized in Figure (b).

- $k = 0$: 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:

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

Step 2: Value update:

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

This policy is visualized in Figure (b).

Value iteration algorithm - Example

- $k = 0$: 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:

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

Step 2: Value update:

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

This policy is visualized in Figure (b).

Value iteration algorithm - Example

- $k = 1$: 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.

Value iteration algorithm - Example

- $k = 1$: 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.

Value iteration algorithm - Example

- $k = 1$: 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.

Value iteration algorithm - Example

- $k = 1$: 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.

Value iteration algorithm - Example

- $k = 1$: 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.

Outline

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!

Similar to the value iteration algorithm? Be patient. We will compare them later.

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!

Similar to the value iteration algorithm? Be patient. We will compare them later.

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!

Similar to the value iteration algorithm? Be patient. We will compare them later.

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!

Similar to the value iteration algorithm? Be patient. We will compare them later.

Policy iteration algorithm

▷ The algorithm leads to a sequence:

$$\pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} v_{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} v_{\pi_2} \xrightarrow{PI} \dots$$

PE=policy evaluation, PI=policy improvement

▷ Questions:

- Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?
- Q2: In the policy improvement step, why is the new policy π_{k+1} better than π_k?
- Q3: Why can such an iterative algorithm finally reach an optimal policy?
- Q4: What is the relationship between this policy iteration algorithm and the previous value iteration algorithm?

Policy iteration algorithm

▷ The algorithm leads to a sequence:

$$\pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} v_{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} v_{\pi_2} \xrightarrow{PI} \dots$$

PE=policy evaluation, PI=policy improvement

▷ Questions:

- Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?
- Q2: In the policy improvement step, why is the new policy π_{k+1} better than π_k?
- Q3: Why can such an iterative algorithm finally reach an optimal policy?
- Q4: What is the relationship between this policy iteration algorithm and the previous value iteration algorithm?

Policy iteration algorithm

▷ The algorithm leads to a sequence:

$$\pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} v_{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} v_{\pi_2} \xrightarrow{PI} \dots$$

PE=policy evaluation, PI=policy improvement

▷ Questions:

- Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?
- Q2: In the policy improvement step, why is the new policy π_{k+1} better than π_k?
- Q3: Why can such an iterative algorithm finally reach an optimal policy?
- Q4: What is the relationship between this policy iteration algorithm and the previous value iteration algorithm?

Policy iteration algorithm

▷ The algorithm leads to a sequence:

$$\pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} v_{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} v_{\pi_2} \xrightarrow{PI} \dots$$

PE=policy evaluation, PI=policy improvement

▷ Questions:

- Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?
- Q2: In the policy improvement step, why is the new policy π_{k+1} better than π_k?
- Q3: Why can such an iterative algorithm finally reach an optimal policy?
- Q4: What is the relationship between this policy iteration algorithm and the previous value iteration algorithm?

Policy iration algorithm

▷ The algorithm leads to a sequence:

$$\pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} v_{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} v_{\pi_2} \xrightarrow{PI} \dots$$

PE=policy evaluation, PI=policy improvement

▷ Questions:

- Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?
- Q2: In the policy improvement step, why is the new policy π_{k+1} better than π_k?
- Q3: Why can such an iterative algorithm finally reach an optimal policy?
- Q4: What is the relationship between this policy iteration algorithm and the previous value iteration algorithm?

▷ Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?

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

- Closed-form solution:

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

- Iterative solution:

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

Already studied in the lecture about Bellman equation.

▷ Policy iteration is an iterative algorithm with another iterative algorithm embedded in the policy evaluation step!

Policy iteration algorithm

▷ Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?

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

- Closed-form solution:

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

- Iterative solution:

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

Already studied in the lecture about Bellman equation.

▷ Policy iteration is an iterative algorithm with another iterative algorithm embedded in the policy evaluation step!

Policy iteration algorithm

▷ Q1: In the policy evaluation step, how to get the state value v_{π_k} by solving the Bellman equation?

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

- Closed-form solution:

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

- Iterative solution:

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

Already studied in the lecture about Bellman equation.

▷ Policy iteration is an iterative algorithm with another iterative algorithm embedded in the policy evaluation step!

Policy iteration algorithm

▷ Q2: In the policy improvement step, why is the new policy π_{k+1} better than π_k?

See the proof in the book.

▷ Q2: In the policy improvement step, why is the new policy π_{k+1} better than π_k?

Lemma (Policy Improvement)

If $\pi_{k+1} = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_k})$, *then* $v_{\pi_{k+1}} \geq v_{\pi_k}$ *for any* k.

See the proof in the book.

Policy iteration algorithm

▷ Q3: Why can such an iterative algorithm finally reach an optimal policy?

Since every iteration would improve the policy, we know

$$v_{\pi_0} \le v_{\pi_1} \le v_{\pi_2} \le \cdots \le v_{\pi_k} \le \cdots \le v^*.$$

As a result, v_{π_k} keeps **increasing** and will converge.

Still need to prove that it converges to the optimal value v^*.

The proof is given in my book.

Policy iteration algorithm

▷ Q3: Why can such an iterative algorithm finally reach an optimal policy?

Since every iteration would improve the policy, we know

$$v_{\pi_0} \leq v_{\pi_1} \leq v_{\pi_2} \leq \cdots \leq v_{\pi_k} \leq \cdots \leq v^*.$$

As a result, v_{π_k} keeps **increasing** and will converge.

Still need to prove that it converges to the optimal value v^*.

The proof is given in my book.

Policy iteration algorithm

▷ Q3: Why can such an iterative algorithm finally reach an optimal policy?

Since every iteration would improve the policy, we know

$$v_{\pi_0} \leq v_{\pi_1} \leq v_{\pi_2} \leq \cdots \leq v_{\pi_k} \leq \cdots \leq v^*.$$

As a result, v_{π_k} keeps **increasing** and will converge.

Still need to prove that it converges to the optimal value v^*.

Theorem (Convergence of Policy Iteration)

The state value sequence $\{v_{\pi_k}\}_{k=0}^{\infty}$ generated by the policy iteration algorithm converges to the optimal state value v^*. As a result, the policy sequence $\{\pi_k\}_{k=0}^{\infty}$ converges to an optimal policy.

The proof is given in my book.

Policy iteration algorithm

▷ Q4: What is the relationship between policy iteration and value iteration algorithms?

Will be explained in detail later.

Policy iteration algorithm - Elementwise form

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:

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

Stop when j is sufficiently large or $\|v_{\pi_k}^{(j+1)} - v_{\pi_k}^{(j)}\|$ is sufficiently small.

Policy iteration algorithm - Elementwise form

Step 1: Policy evaluation

- Matrix-vector form: $v_{\pi_k}^{(j+1)} = r_{\pi_k} + \gamma P_{\pi_k} v_{\pi_k}^{(j)}, \quad j = 0, 1, 2, \dots$
- Elementwise form:

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

Stop when j is sufficiently large or $\|v_{\pi_k}^{(j+1)} - v_{\pi_k}^{(j)}\|$ is sufficiently small.

Policy iteration algorithm - Elementwise form

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:

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

Stop when j is sufficiently large or $\|v_{\pi_k}^{(j+1)} - v_{\pi_k}^{(j)}\|$ is sufficiently small.

Policy iteration algorithm - Elementwise form

Step 2: Policy improvement

▷ Matrix-vector form: $\pi_{k+1} = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_k})$

▷ Elementwise form

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

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

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

Step 2: Policy improvement

▷ Matrix-vector form: $\pi_{k+1} = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_k})$

▷ Elementwise form

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

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

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

Step 2: Policy improvement

- Matrix-vector form: $\pi_{k+1} = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_k})$
- Elementwise form

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

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

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

Policy iteration algorithm - Implementation

Pseudocode: Policy iteration algorithm

Initialization: The probability models $p(r|s,a)$ 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 v_{π_k} 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

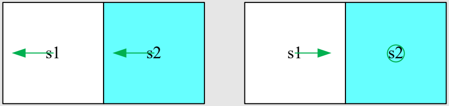
For every action $a \in \mathcal{A}$, do

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

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

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

Policy iteration algorithm - Simple example



(a) initial policy (b) optimal policy

- The reward setting is $r_{\text{boundary}} = -1$ and $r_{\text{target}} = 1$. The discount rate is $\gamma = 0.9$.
- Actions: a_ℓ, a_0, a_r represent going left, staying still, and going right, respectively.
- Goal: use the policy iteration algorithm to find out an optimal policy.

Policy iteration algorithm - Simple example

▷ Iteration $k = 0$: Step 1: policy evaluation

π_0 is selected as the policy in Figure (a). The Bellman equation is

$$v_{\pi_0}(s_1) = -1 + \gamma v_{\pi_0}(s_1),$$
$$v_{\pi_0}(s_2) = 0 + \gamma v_{\pi_0}(s_1).$$

- Solve the equations directly:

$$v_{\pi_0}(s_1) = -10, \quad v_{\pi_0}(s_2) = -9.$$

- Solve the equations iteratively. Select the initial guess as $v_{\pi_0}^{(0)}(s_1) = v_{\pi_0}^{(0)}(s_2) = 0$:

$$\begin{cases} v_{\pi_0}^{(1)}(s_1) = -1 + \gamma v_{\pi_0}^{(0)}(s_1) = -1, \\ v_{\pi_0}^{(1)}(s_2) = 0 + \gamma v_{\pi_0}^{(0)}(s_1) = 0, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(2)}(s_1) = -1 + \gamma v_{\pi_0}^{(1)}(s_1) = -1.9, \\ v_{\pi_0}^{(2)}(s_2) = 0 + \gamma v_{\pi_0}^{(1)}(s_1) = -0.9, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(3)}(s_1) = -1 + \gamma v_{\pi_0}^{(2)}(s_1) = -2.71, \\ v_{\pi_0}^{(3)}(s_2) = 0 + \gamma v_{\pi_0}^{(2)}(s_1) = -1.71, \end{cases}$$

$$\ldots$$

Policy iteration algorithm - Simple example

▷ Iteration $k = 0$: Step 1: policy evaluation

π_0 is selected as the policy in Figure (a). The Bellman equation is

$$v_{\pi_0}(s_1) = -1 + \gamma v_{\pi_0}(s_1),$$
$$v_{\pi_0}(s_2) = 0 + \gamma v_{\pi_0}(s_1).$$

- Solve the equations directly:

$$v_{\pi_0}(s_1) = -10, \quad v_{\pi_0}(s_2) = -9.$$

- Solve the equations iteratively. Select the initial guess as $v_{\pi_0}^{(0)}(s_1) = v_{\pi_0}^{(0)}(s_2) = 0$:

$$\begin{cases} v_{\pi_0}^{(1)}(s_1) = -1 + \gamma v_{\pi_0}^{(0)}(s_1) = -1, \\ v_{\pi_0}^{(1)}(s_2) = 0 + \gamma v_{\pi_0}^{(0)}(s_1) = 0, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(2)}(s_1) = -1 + \gamma v_{\pi_0}^{(1)}(s_1) = -1.9, \\ v_{\pi_0}^{(2)}(s_2) = 0 + \gamma v_{\pi_0}^{(1)}(s_1) = -0.9, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(3)}(s_1) = -1 + \gamma v_{\pi_0}^{(2)}(s_1) = -2.71, \\ v_{\pi_0}^{(3)}(s_2) = 0 + \gamma v_{\pi_0}^{(2)}(s_1) = -1.71, \end{cases}$$

$$\ldots$$

Policy iteration algorithm - Simple example

▷ Iteration $k = 0$: Step 1: policy evaluation

π_0 is selected as the policy in Figure (a). The Bellman equation is

$$v_{\pi_0}(s_1) = -1 + \gamma v_{\pi_0}(s_1),$$
$$v_{\pi_0}(s_2) = 0 + \gamma v_{\pi_0}(s_1).$$

- Solve the equations directly:

$$v_{\pi_0}(s_1) = -10, \quad v_{\pi_0}(s_2) = -9.$$

- Solve the equations iteratively. Select the initial guess as $v_{\pi_0}^{(0)}(s_1) = v_{\pi_0}^{(0)}(s_2) = 0$:

$$\begin{cases} v_{\pi_0}^{(1)}(s_1) = -1 + \gamma v_{\pi_0}^{(0)}(s_1) = -1, \\ v_{\pi_0}^{(1)}(s_2) = 0 + \gamma v_{\pi_0}^{(0)}(s_1) = 0, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(2)}(s_1) = -1 + \gamma v_{\pi_0}^{(1)}(s_1) = -1.9, \\ v_{\pi_0}^{(2)}(s_2) = 0 + \gamma v_{\pi_0}^{(1)}(s_1) = -0.9, \end{cases}$$

$$\begin{cases} v_{\pi_0}^{(3)}(s_1) = -1 + \gamma v_{\pi_0}^{(2)}(s_1) = -2.71, \\ v_{\pi_0}^{(3)}(s_2) = 0 + \gamma v_{\pi_0}^{(2)}(s_1) = -1.71, \end{cases}$$

$$\ldots$$

Policy iteration algorithm - Simple example

▷ Iteration $k = 0$: Step 2: policy improvement

The expression of $q_{\pi_k}(s, a)$:

$q_{\pi_k}(s, a)$	a_ℓ	a_0	a_r
s_1	$-1 + \gamma v_{\pi_k}(s_1)$	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$
s_2	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$	$-1 + \gamma v_{\pi_k}(s_2)$

Substituting $v_{\pi_0}(s_1) = -10, v_{\pi_0}(s_2) = -9$ and $\gamma = 0.9$ gives

$q_{\pi_0}(s, a)$	a_ℓ	a_0	a_r
s_1	-10	-9	-7.1
s_2	-9	-7.1	-9.1

By seeking the greatest value of q_{π_0}, the improved new policy is:

$$\pi_1(a_r|s_1) = 1, \quad \pi_1(a_0|s_2) = 1.$$

This policy is optimal after one iteration! In your programming, should continue until the stopping criterion is satisfied.

Policy iteration algorithm - Simple example

▷ Iteration $k=0$: Step 2: policy improvement

The expression of $q_{\pi_k}(s,a)$:

$q_{\pi_k}(s,a)$	a_ℓ	a_0	a_r
s_1	$-1+\gamma v_{\pi_k}(s_1)$	$0+\gamma v_{\pi_k}(s_1)$	$1+\gamma v_{\pi_k}(s_2)$
s_2	$0+\gamma v_{\pi_k}(s_1)$	$1+\gamma v_{\pi_k}(s_2)$	$-1+\gamma v_{\pi_k}(s_2)$

Substituting $v_{\pi_0}(s_1)=-10, v_{\pi_0}(s_2)=-9$ and $\gamma=0.9$ gives

$q_{\pi_0}(s,a)$	a_ℓ	a_0	a_r
s_1	-10	-9	-7.1
s_2	-9	-7.1	-9.1

By seeking the greatest value of q_{π_0}, the improved new policy is:

$$\pi_1(a_r|s_1)=1, \quad \pi_1(a_0|s_2)=1.$$

This policy is optimal after one iteration! In your programming, should continue until the stopping criterion is satisfied.

Policy iteration algorithm - Simple example

▷ Iteration $k = 0$: Step 2: policy improvement

The expression of $q_{\pi_k}(s, a)$:

$q_{\pi_k}(s,a)$	a_ℓ	a_0	a_r
s_1	$-1 + \gamma v_{\pi_k}(s_1)$	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$
s_2	$0 + \gamma v_{\pi_k}(s_1)$	$1 + \gamma v_{\pi_k}(s_2)$	$-1 + \gamma v_{\pi_k}(s_2)$

Substituting $v_{\pi_0}(s_1) = -10, v_{\pi_0}(s_2) = -9$ and $\gamma = 0.9$ gives

$q_{\pi_0}(s,a)$	a_ℓ	a_0	a_r
s_1	-10	-9	-7.1
s_2	-9	-7.1	-9.1

By seeking the greatest value of q_{π_0}, the improved new policy is:

$$\pi_1(a_r|s_1) = 1, \quad \pi_1(a_0|s_2) = 1.$$

This policy is optimal after one iteration! In your programming, should continue until the stopping criterion is satisfied.

Exercise: Set the left cell as the target area.

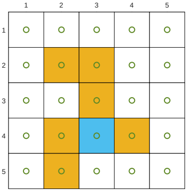
- Now we have another powerful algorithm searching for optimal policies!
- Next let's apply it and see what we can find.

Exercise: Set the left cell as the target area.

- Now we have another powerful algorithm searching for optimal policies!
- Next let's apply it and see what we can find.

Policy iteration algorithm - Complicated example

- Setting: $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, $r_{\text{target}} = 1$, $\gamma = 0.9$.
- Let's check out the intermediate policies and state values.

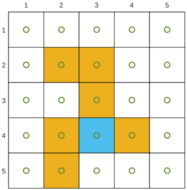


	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	-100.0	-100.0	0.0	0.0
3	0.0	0.0	-100.0	0.0	0.0
4	0.0	-100.0	10.0	-100.0	0.0
5	0.0	-100.0	0.0	0.0	0.0

π_0 and v_{π_0}

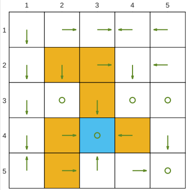
Policy iteration algorithm - Complicated example

- Setting: $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, $r_{\text{target}} = 1$, $\gamma = 0.9$.
- Let's check out the intermediate policies and state values.



	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	-100.0	-100.0	0.0	0.0
3	0.0	0.0	-100.0	0.0	0.0
4	0.0	-100.0	10.0	-100.0	0.0
5	0.0	-100.0	0.0	0.0	0.0

π_0 and v_{π_0}

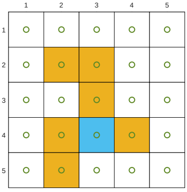


	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	0.0
4	0.0	10.0	10.0	10.0	0.0
5	0.0	9.0	10.0	0.0	0.0

π_1 and v_{π_1}

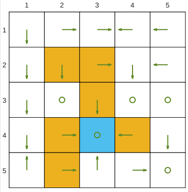
Policy iteration algorithm - Complicated example

- Setting: $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, $r_{\text{target}} = 1$, $\gamma = 0.9$.
- Let's check out the intermediate policies and state values.



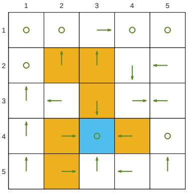
	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	-100.0	-100.0	0.0	0.0
3	0.0	0.0	-100.0	0.0	0.0
4	0.0	-100.0	10.0	-100.0	0.0
5	0.0	-100.0	0.0	0.0	0.0

π_0 and v_{π_0}



	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	0.0
4	0.0	10.0	10.0	10.0	0.0
5	0.0	9.0	10.0	0.0	0.0

π_1 and v_{π_1}

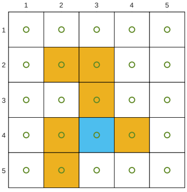


	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	0.0
4	0.0	10.0	10.0	10.0	0.0
5	0.0	9.0	10.0	9.0	0.0

π_2 and v_{π_2}

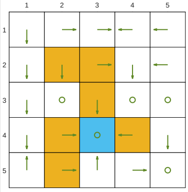
Policy iteration algorithm - Complicated example

- ▷ Setting: $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, $r_{\text{target}} = 1$, $\gamma = 0.9$.
- ▷ Let's check out the intermediate policies and state values.



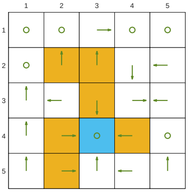
	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	-100.0	-100.0	0.0	0.0
3	0.0	0.0	-100.0	0.0	0.0
4	0.0	-100.0	10.0	-100.0	0.0
5	0.0	-100.0	0.0	0.0	0.0

π_0 and v_{π_0}



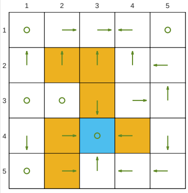
	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	0.0
4	0.0	10.0	10.0	10.0	0.0
5	0.0	9.0	10.0	0.0	0.0

π_1 and v_{π_1}



	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	0.0
4	0.0	10.0	10.0	10.0	0.0
5	0.0	9.0	10.0	9.0	0.0

π_2 and v_{π_2}

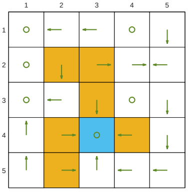


	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	0.0
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_3 and v_{π_3}

Policy iteration algorithm - Complicated example

▷ Interesting pattern of the policies and state values

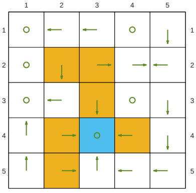


	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_4 and v_{π_4}

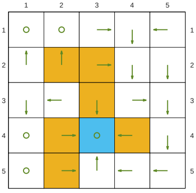
Policy iteration algorithm - Complicated example

▷ Interesting pattern of the policies and state values



	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_4 and v_{π_4}

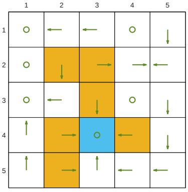


	1	2	3	4	5
1	0.0	0.0	4.3	4.8	4.3
2	0.0	0.0	4.8	5.3	5.9
3	0.0	0.0	10.0	5.9	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_5 and v_{π_5}

Policy iteration algorithm - Complicated example

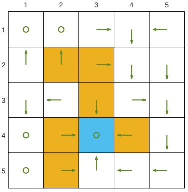
- Interesting pattern of the policies and state values



	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_4 and v_{π_4}

$\vdots$

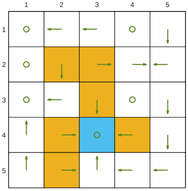


	1	2	3	4	5
1	0.0	0.0	4.3	4.8	4.3
2	0.0	0.0	4.8	5.3	5.9
3	0.0	0.0	10.0	5.9	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_5 and v_{π_5}

$\vdots$

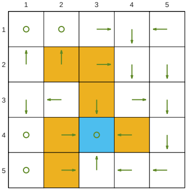
- Interesting pattern of the policies and state values



	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_4 and v_{π_4}

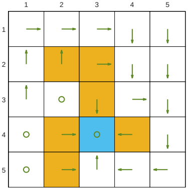
$\vdots$



	1	2	3	4	5
1	0.0	0.0	4.3	4.8	4.3
2	0.0	0.0	4.8	5.3	5.9
3	0.0	0.0	10.0	5.9	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_5 and v_{π_5}

$\vdots$

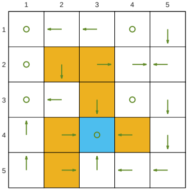


	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	0.0	10.0	5.9	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_9 and v_{π_9}

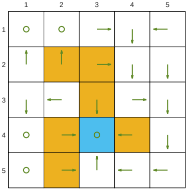
Policy iteration algorithm - Complicated example

▷ Interesting pattern of the policies and state values



	1	2	3	4	5
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	10.0	0.0	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

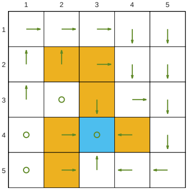
π_4 and v_{π_4}



	1	2	3	4	5
1	0.0	0.0	4.3	4.8	4.3
2	0.0	0.0	4.8	5.3	5.9
3	0.0	0.0	10.0	5.9	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

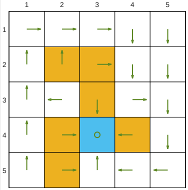
π_5 and v_{π_5}

$\vdots$



	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	0.0	10.0	5.9	6.6
4	0.0	10.0	10.0	10.0	7.3
5	0.0	9.0	10.0	9.0	8.1

π_9 and v_{π_9}



	1	2	3	4	5
1	3.5	3.9	4.3	4.8	5.3
2	3.1	3.5	4.8	5.3	5.9
3	2.8	2.5	10.0	5.9	6.6
4	2.5	10.0	10.0	10.0	7.3
5	2.3	9.0	10.0	9.0	8.1

π_{10} and $v_{\pi_{10}}$

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):

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

Value iteration: start from v_0

- Policy update (PU):

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

- Value update (VU):

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

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):

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

Value iteration: start from v_0

- Policy update (PU):

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

- Value update (VU):

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

Compare value iteration and policy iteration

▷ The two algorithms are very similar:

$$\text{Policy iteration: } \pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} v_{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} v_{\pi_2} \xrightarrow{PI} \dots$$

$$\text{Value iteration: } \quad u_0 \xrightarrow{PU} \pi_1' \xrightarrow{VU} u_1 \xrightarrow{PU} \pi_2' \xrightarrow{VU} u_2 \xrightarrow{PU} \dots$$

- PE=policy evaluation. PI=policy improvement.
- PU=policy update. VU=value update.

Compare value iteration and policy iteration

▷ Let's compare the steps carefully:

	Policy iteration algorithm	Value iteration algorithm	Comments
1) Policy:	π_0	N/A	
2) Value:	$v_{\pi_0} = r_{\pi_0} + \gamma P_{\pi_0} v_{\pi_0}$	$v_0 \doteq v_{\pi_0}$	
3) Policy:	$\pi_1 = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_0})$	$\pi_1 = \arg\max_\pi (r_\pi + \gamma P_\pi v_0)$	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:	$\pi_2 = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_1})$	$\pi'_2 = \arg\max_\pi (r_\pi + \gamma P_\pi v_1)$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Remarks:

- 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)
  - In value iteration, $v_1 = r_{\pi_1} + \gamma P_{\pi_1} v_0$ is a one-step iteration

Compare value iteration and policy iteration

▷ Let's compare the steps carefully:

	Policy iteration algorithm	Value iteration algorithm	Comments
1) Policy:	π_0	N/A	
2) Value:	$v_{\pi_0} = r_{\pi_0} + \gamma P_{\pi_0} v_{\pi_0}$	$v_0 \doteq v_{\pi_0}$	
3) Policy:	$\pi_1 = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_0})$	$\pi_1 = \arg\max_\pi (r_\pi + \gamma P_\pi v_0)$	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:	$\pi_2 = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_1})$	$\pi_2' = \arg\max_\pi (r_\pi + \gamma P_\pi v_1)$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Remarks:

- 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)
  - In value iteration, $v_1 = r_{\pi_1} + \gamma P_{\pi_1} v_0$ is a one-step iteration

More specifically, examine the step of solving $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$:

$$
\begin{aligned}
v_{\pi_1}^{(0)} &= v_0 \\
v_{\pi_1}^{(1)} &= r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(0)} \\
v_{\pi_1}^{(2)} &= r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(1)} \\
&\vdots \\
v_{\pi_1}^{(j)} &= r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(j-1)} \\
&\vdots \\
v_{\pi_1}^{(\infty)} &= r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(\infty)}
\end{aligned}
$$

Compare value iteration and policy iteration

More specifically, examine the step of solving $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$:

$$
\begin{aligned}
& v_{\pi_1}^{(0)} = v_0 \\
\text{value iteration} \leftarrow v_1 \longleftarrow\, & v_{\pi_1}^{(1)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(0)} \\
& v_{\pi_1}^{(2)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(1)} \\
& \quad \vdots \\
& v_{\pi_1}^{(j)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(j-1)} \\
& \quad \vdots \\
& v_{\pi_1}^{(\infty)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(\infty)}
\end{aligned}
$$

Compare value iteration and policy iteration

More specifically, examine the step of solving $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$:

$$
\begin{aligned}
& v_{\pi_1}^{(0)} = v_0 \\
\text{value iteration} \leftarrow v_1 \longleftarrow\ & v_{\pi_1}^{(1)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(0)} \\
& v_{\pi_1}^{(2)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(1)} \\
& \vdots \\
& v_{\pi_1}^{(j)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(j-1)} \\
& \vdots \\
\text{policy iteration} \leftarrow v_{\pi_1} \longleftarrow\ & v_{\pi_1}^{(\infty)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(\infty)}
\end{aligned}
$$

Compare value iteration and policy iteration

More specifically, examine the step of solving $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$:

$$
\begin{aligned}
& v_{\pi_1}^{(0)} = v_0 \\
\text{value iteration} \leftarrow v_1 \longleftarrow & v_{\pi_1}^{(1)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(0)} \\
& v_{\pi_1}^{(2)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(1)} \\
& \vdots \\
\text{truncated policy iteration} \leftarrow \bar{v}_1 \longleftarrow & v_{\pi_1}^{(j)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(j-1)} \\
& \vdots \\
\text{policy iteration} \leftarrow v_{\pi_1} \longleftarrow & v_{\pi_1}^{(\infty)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(\infty)}
\end{aligned}
$$

Compare value iteration and policy iteration

More specifically, examine the step of solving $v_{\pi_1} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}$:

$$
\begin{aligned}
& v_{\pi_1}^{(0)} = v_0 \\
\text{value iteration} \leftarrow v_1 \longleftarrow\; & v_{\pi_1}^{(1)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(0)} \\
& v_{\pi_1}^{(2)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(1)} \\
& \vdots \\
\text{truncated policy iteration} \leftarrow \bar{v}_1 \longleftarrow\; & v_{\pi_1}^{(j)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(j-1)} \\
& \vdots \\
\text{policy iteration} \leftarrow v_{\pi_1} \longleftarrow\; & v_{\pi_1}^{(\infty)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(\infty)}
\end{aligned}
$$

- The value iteration algorithm computes once.
- The policy iteration algorithm computes an infinite number of iterations.
- The truncated policy iteration algorithm computes a finite number of iterations (say j). The rest iterations from j to ∞ are truncated.

Truncated policy iteration - Pseudocode

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 v_k 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

Truncated policy iteration - Convergence

▷ Will the truncation undermine convergence?

Proposition (Value Improvement)

Consider the iterative algorithm for solving the policy evaluation step:

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

If the initial guess is selected as $v_{\pi_k}^{(0)} = v_{\pi_{k-1}}$, *it holds that*

$$v_{\pi_k}^{(j+1)} \geq v_{\pi_k}^{(j)}$$

for every $j = 0, 1, 2, \ldots$.

For the proof, see the book.

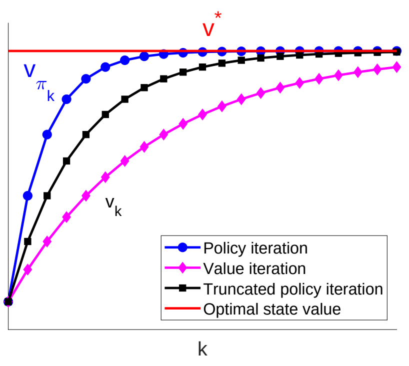


Figure: Illustration of the relationship among value iteration, policy iteration, and truncated policy iteration.

The convergence proof of PI is based on that of VI. Since VI converges, we know PI converges.

Summary

▷ Value iteration: it is the iterative algorithm solving the Bellman optimality equation: given an initial value v_0,

$$v_{k+1} = \max_\pi (r_\pi + \gamma P_\pi v_k)$$
$$\Updownarrow$$
$$\begin{cases} \text{Policy update: } \pi_{k+1} = \arg\max_\pi (r_\pi + \gamma P_\pi v_k) \\ \text{Value update: } v_{k+1} = r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k \end{cases}$$

▷ Policy iteration: given an initial policy π_0,

$$\begin{cases} \text{Policy evaluation: } v_{\pi_k} = r_{\pi_k} + \gamma P_{\pi_k} v_{\pi_k} \\ \text{Policy improvement: } \pi_{k+1} = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_k}) \end{cases}$$

▷ Truncated policy iteration