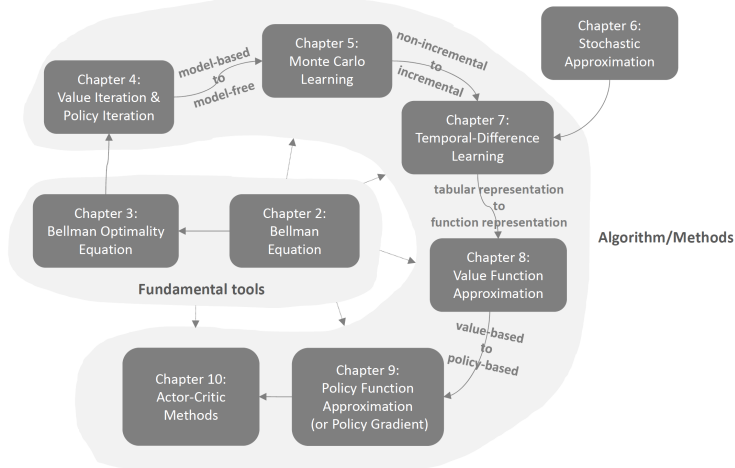


Lecture 4: Value Iteration and Policy Iteration

Shiyu Zhao

Outline



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

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

Value iteration algorithm

- ▷ How to solve the Bellman optimality equation?

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

- ▷ In the last lecture, we know that the contraction mapping theorem suggest an iterative algorithm:

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

where v_0 can be arbitrary.

- ▷ This algorithm can eventually find the optimal state value and an optimal policy.
- ▷ This algorithm is called *value iteration*!
- ▷ We will see that the math about the BOE that we have learned finally pays off!

Value iteration algorithm

The algorithm

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

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.**

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

Question: is v_k a state value? No, because it is not ensured that v_k satisfies a Bellman equation.

Value iteration algorithm

- ▷ Next, we need to study the elementwise form in order to implement the algorithm.
- Matrix-vector form is useful for theoretical analysis.
- Elementwise form is useful for implementation.

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 - Pseudocode

▷ Procedure summary:

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

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 k th iteration, do

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

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

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

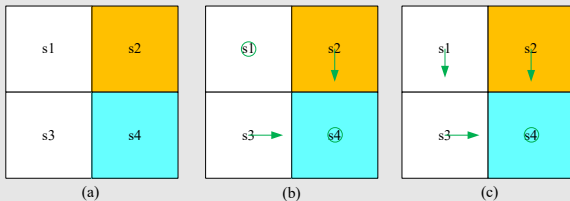
Maximum action value: $a_k^*(s) = \arg \max_a q_k(a, s)$

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

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

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 $q(s, a)$.

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:

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

This policy is visualized in Figure (b).

Step 2: Value update:

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

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!!

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

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

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!

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 such an iterative algorithm can finally reach an optimal policy?
- Q4: What is the relationship between this policy iteration algorithm and the previous value iteration algorithm?

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:

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

- ▷ Already studied in the lecture on 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 ?**

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

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

As a result, v_{π_k} keeps increasing and will converge. Still need to prove it converges to 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.*

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

Related to the answer to Q3 and will be explained in detail later.

Policy iteration algorithm - Elementwise form

Step 1: Policy evaluation

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

▷ 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 $j \rightarrow \infty$ or 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

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

Policy iteration algorithm - Implementation

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 k th iteration, do

Policy evaluation:

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

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

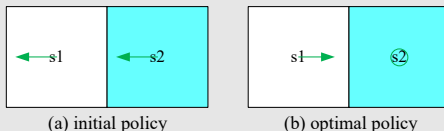
For every action $a \in \mathcal{A}(s)$, do

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

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

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

Policy iteration algorithm - Simple example



- ▷ 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 go left, stay unchanged, and go right.
- ▷ Aim: use policy iteration to find out the 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

$$v_{\pi_0}^{(0)}(s_1) = v_{\pi_0}^{(0)}(s_2) = 0:$$

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

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

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

...

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 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

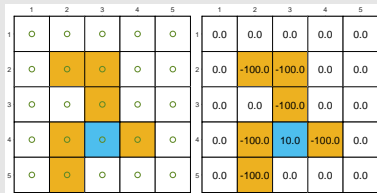
Excise! Set the left cell as the target area.

Now you know another powerful algorithm searching for optimal policies!

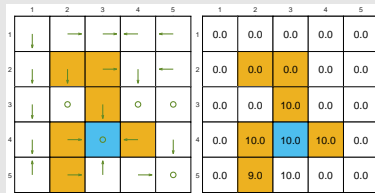
Now let's apply it and see what we can find.

Policy iteration algorithm - Complicated example

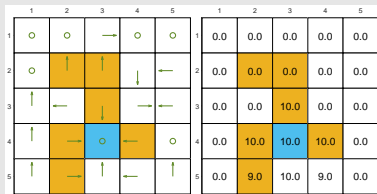
- ▷ Setting: $r_{\text{boundary}} = -1$, $r_{\text{forbidden}} = -10$, $r_{\text{target}} = 1$, $\gamma = 0.9$.
- ▷ Let's check out the intermediate policies and state values.



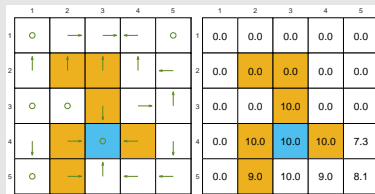
π_0 and v_{π_0}



π_1 and v_{π_1}



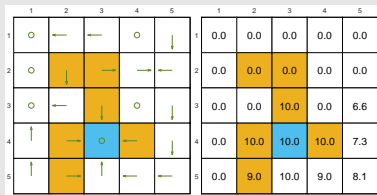
π_2 and v_{π_2}



π_3 and v_{π_3}

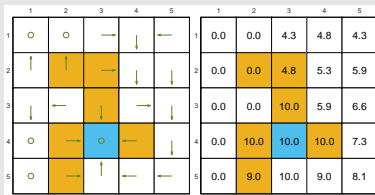
Policy iteration algorithm - Complicated example

- ▷ Interesting pattern of the policies and state values



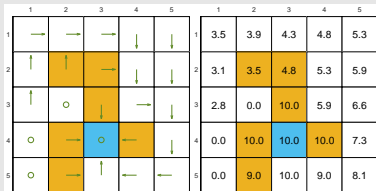
π_4 and v_{π_4}

⋮

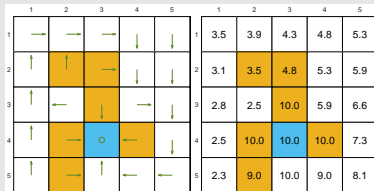


π_5 and v_{π_5}

⋮



π_9 and v_{π_9}



π_{10} and $v_{\pi_{10}}$

Outline

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

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:

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

Value iteration: $u_0 \xrightarrow{PU} \pi'_1 \xrightarrow{VU} u_1 \xrightarrow{PU} \pi'_2 \xrightarrow{VU} u_2 \xrightarrow{PU} \dots$

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:	$v_{\pi_0} = r_{\pi_0} + \gamma P_{\pi_0} v_{\pi_0}$	$v_0 := v_{\pi_0}$	
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

- 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

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

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

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

$$v_{\pi_1}^{(2)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(1)}$$

$$\vdots$$

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

$$\vdots$$

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

- 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 the policy has not converged, for the k th 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

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

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

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

Truncated policy iteration - Convergence

▷ Will the truncation undermine convergence?

Proposition (Value Improvement)

Consider the iterative algorithm for solving the policy evaluation step:

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

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 $j = 0, 1, 2, \dots$.

For the proof, see the book.

Truncated policy iteration - Convergence

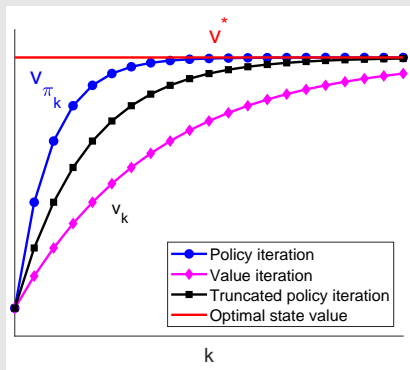



























Figure: Illustration of the relationship among value iteration, policy iteration, and truncated policy iteration.

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

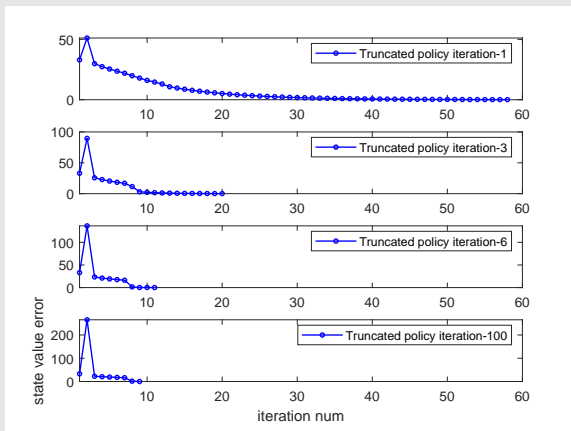
Truncated policy iteration - Example

▷ Setup: The same as the previous example. Below is the initial policy.

	1	2	3	4	5
1					
2					
3					
4					
5					

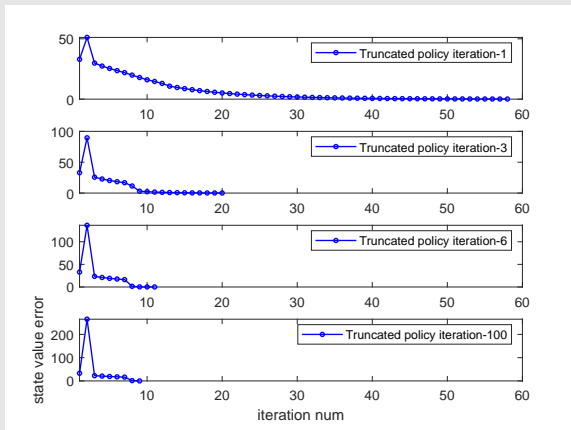
▷ Define $\|v_k - v^*\|$ as the state value error at time k . The stop criterion is $\|v_k - v^*\| < 0.01$.

Truncated policy iteration - Example



▷ “Truncated policy iteration- x ” where $x = 1, 3, 6, 100$ refers to a truncated policy iteration algorithm where the policy evaluation step runs x iterations.

Truncated policy iteration - Example



- ▷ The greater the value of x is, the faster the value estimate converges.
- ▷ However, the benefit of increasing x drops quickly when x is large.
- ▷ In practice, run a few number of iterations in the policy evaluation step.

Summary

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

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



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

- ▷ 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