Chapter 4: Value Iteration & Policy Iteration

(Model Based)

(Dynamic programming)

1. Outline

① Value Iteration algorithm

② Policy Iteration algorithm

③ Truncated policy Iteration algorithm.

2. Value Iteration

▷ 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:

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

where v_0 can be arbitrary. 任取的.

▷ 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!

The algorithm

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

can be decomposed to two steps.

① Step 1: policy update:

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

v_k is given

② Step2: Value update:

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

v_k is not a state value

v_k is not ensured satisfies a Bellman equation.

- Matrix-vector form useful for theoretical analysis
- Elementwise form ---- implementation

Step 1: Policy update:

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

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.

Zhao 7

Step 2: Value update:

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

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

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

procedure summary:

$v_k(s) \rightarrow q_k(s,a) \rightarrow$ greedy policy $\pi_{k+1}(a|s)$

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

是否收敛

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

< 0.001

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

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

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

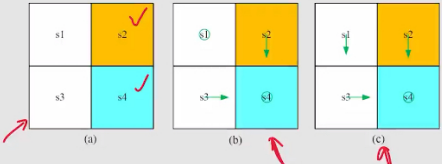
Maximum action value: $a_k^*(s) = \arg\max_a q_k(a,s)$

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:

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

- $k = 0$: 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!!

局部最优?

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

3. 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 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?

▷ **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!

Shiyu Zhao

▷ **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.

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

Since every iteration would improve the policy, we know

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

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.

两个极端.

Elementwise form:

Step 1:

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

Step 2:

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

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

pseudocode :

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

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

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

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

Yuyu Zhao 2

Example (simple)

(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 go left, stay unchanged, and go right.

▷ Aim: use policy iteration to find out the optimal policy.

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

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

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

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

...

Zhao

▷ 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.

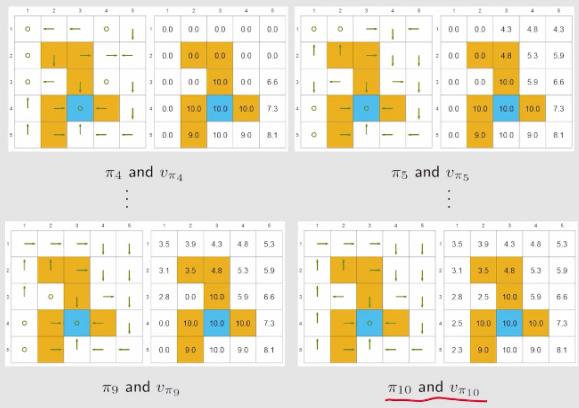
Complicated example:

▷ 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.

Zhao

▷ Interesting pattern of the policies and state values



策略：越来越好、先距离目标最近的策略先变好.

像人找迷宫一样，从出口往回找.

3. Truncated policy iteration algorithm

general algorithm of 4.1 and 4.2. < value / policy

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):

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

▷ 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.

different:

▷ 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:	$\pi_2 = \arg\max_\pi (r_\pi + \gamma P_\pi v_{\pi_1})$	$\pi_2' = \arg\max_\pi (r_\pi + \gamma P_\pi v_1)$	
⋮	⋮	⋮	⋮

关键是在于第4步

Compare value iteration and policy iteration

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

只计算3步 → value iteration ← v_1 ←

$$v_{\pi_1}^{(0)} = v_0$$
$$v_{\pi_1}^{(1)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(0)} \quad j=1$$
$$v_{\pi_1}^{(2)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(1)}$$
$$\vdots$$
truncated policy iteration ← $v_{\pi_1}^{(j)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(j-1)}$ 有限j步
$$\vdots$$
计算3无穷步 → policy iteration ← v_{π_1} ← $v_{\pi_1}^{(\infty)} = r_{\pi_1} + \gamma P_{\pi_1} v_{\pi_1}^{(\infty)}$ $j=\infty$

⇓

实际上是截断的

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

will the truncation undermine convergence?

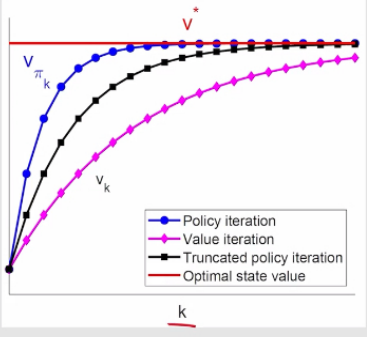


Figure: Illustration of the relationship among value iteration, policy iteration, and truncated policy iteration.

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

4. Summary.

① Value iteration

② policy iteration

③ Truncated policy iteration.

▷ 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