Chapter 4 Value Iteration and Policy Iteration

Value Iteration, Policy Iteration, policy update, value update, Truncated Policy Iteration

Value Iteration:

1. Explain the process of value iteration.
2. In every iteration, when calculating based on every possible policy, can this be infinite because there are infinite policies?
3. What is policy update and value update?
4. In every iteration, is v(s) satisfy Bellman Equation?
5. What is Q-Table?

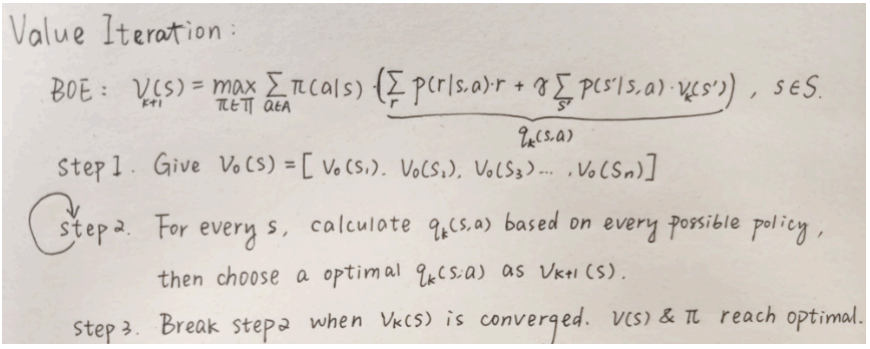
Policy Iteration:

6. Explain the process of policy iteration.
7. What is the fundamental difference between policy iteration and value iteration?
8. In every iteration, is v(s) satisfy Bellman Equation?

Truncated Policy Iteration:

9. Explain the process of truncated policy iteration.
10. Explain the difference and common between truncated policy iteration, policy iteration, and value iteration. What are their advantages?
11. Any tip to make policy evaluation more efficient?

1.



Algorithm 4.1: 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.
Goal: Search for the optimal state value and an optimal policy for 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: $a_k^*(s) = \arg\max_a q_k(s,a)$
    Policy update: $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$, and $\pi_{k+1}(a|s) = 0$ otherwise
    Value update: $v_{k+1}(s) = \max_a q_k(s,a)$

2. No. From BOE, we can understand that only in this condition can BOE reach its optimal:

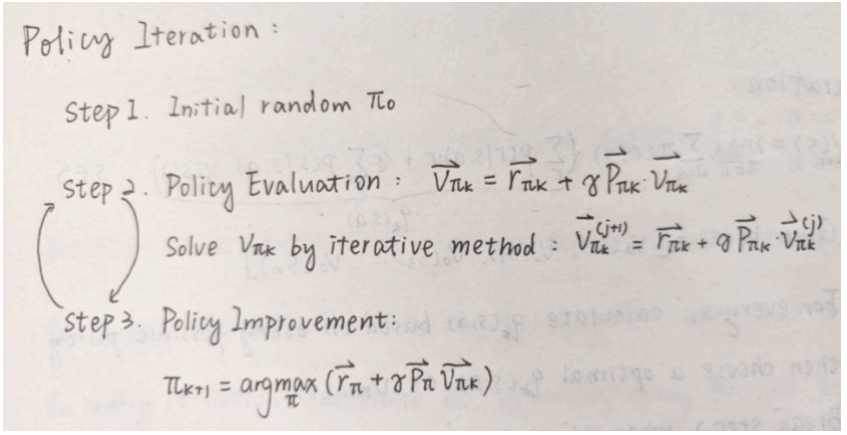
$$\pi(a|s) = \begin{cases} 1, & a = a^*, \\ 0, & a \neq a^*. \end{cases}$$

Thus, the policy space is finite. It is equal to, in every state, we just need calculate every action's action value and choose the largest one.

3. Policy update is when π is assigned a new value. Value update is when *v* is assigned a new value.
4. No. Cannot ensure it satisfies the Bellman Equation. It is only an intermediate value mathematically generated by the algorithm. When it is converged, v(s) then satisfies the Bellman Equation.
5.

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

6.



Algorithm 4.2: Policy iteration algorithm

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

7. Policy iteration used infinite iters to calculate v_k, but value iteration only use 1 iter to get a new v_k.
8. Yes. Because these v(s) is the precise solution of the Bellman Equation. However, in truncated policy iteration, v(s) is not the solution of Bellman Equation.
9.

Algorithm 4.3: Truncated 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.
Goal: 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 number of iterations is set as 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

10. Value iteration use 1 iter to estimate v(s), policy iteration use infinite iters to estimate a precise v(s), truncated policy iteration use finite iters to estimate v(s). VI has the least calculation but the slowest to converge. Policy iteration has the most calculation but the fast to converge.

    A common property of the three algorithms is that every iteration has two steps. One step is to update the value, and the other step is to update the policy. The idea of interaction between value and policy updates widely exists in reinforcement learning algorithms. This idea is also called generalized policy iteration
11. In policy iteration, choose $v_{\pi_{k-1}}$ as the initial value v v_{π_0} in this iter. However, in truncated policy iteration, we cannot get the precise $v_{\pi_{k-1}}$, but it can still make the process more efficient.

evaluation step:

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

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