Policy Iteration

24.01.18
정상혁

Markov Decision Process (MDP)

- MDP를 푸는 방법에는 크게 2가지가 존재한다. >> 여기에서 ‘푼다’ 라는 것은 **optimal policy**를 찾는 것이다.

  - Dynamic Programming (DP)

  - Reinforcement Learing (RL)

- MDP를 DP으로 푸는 방법에는 크게 2가지가 존재한다.

  - Value Iteration

  - Policy Iteration

Value Iteration

(1) $$V_{k+1}(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V_k(s')]$$

$$v_*(s) \ \Big\downarrow$$

(2) $$\pi_*(s) = arg \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$$

Bellman optimality equation을 사용한다.

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$$

단점

- 한번의 iteration 마다 $O(S^2A)$ 의 시간복잡도를 가진다.
  더불어 convergence를 위해 많은 iteration을 반복해야한다.

  >> 계산량이 너무 많다.

- optimal policy가 이미 고정된 이후에도, optimal state-value function이 수렴할때까지 계속해서 계산을 진행한다.

  >> 불필요한 추가 계산이 존재할 수 있다.

Policy Iteration

- Policy가 수렴할때까지 policy evaluation 과 policy improvement를 반복한다.

- Policy Evaluation : policy π에 대해서 $V^{\pi}(s)$를 계산한다.

$$V_{k+1}(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))[r + \gamma V_k(s')]$$

- Policy Improvement : $V^{\pi}(s)$를 이용하여 policy 를 update 한다. ($\pi \rightarrow \pi'$)

$$\pi'(s) = arg\max_{a} \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^{\pi}(s')] = arg\max_{a} Q^{\pi}(s, a)$$

Bellman expectation equation을 사용한다.

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

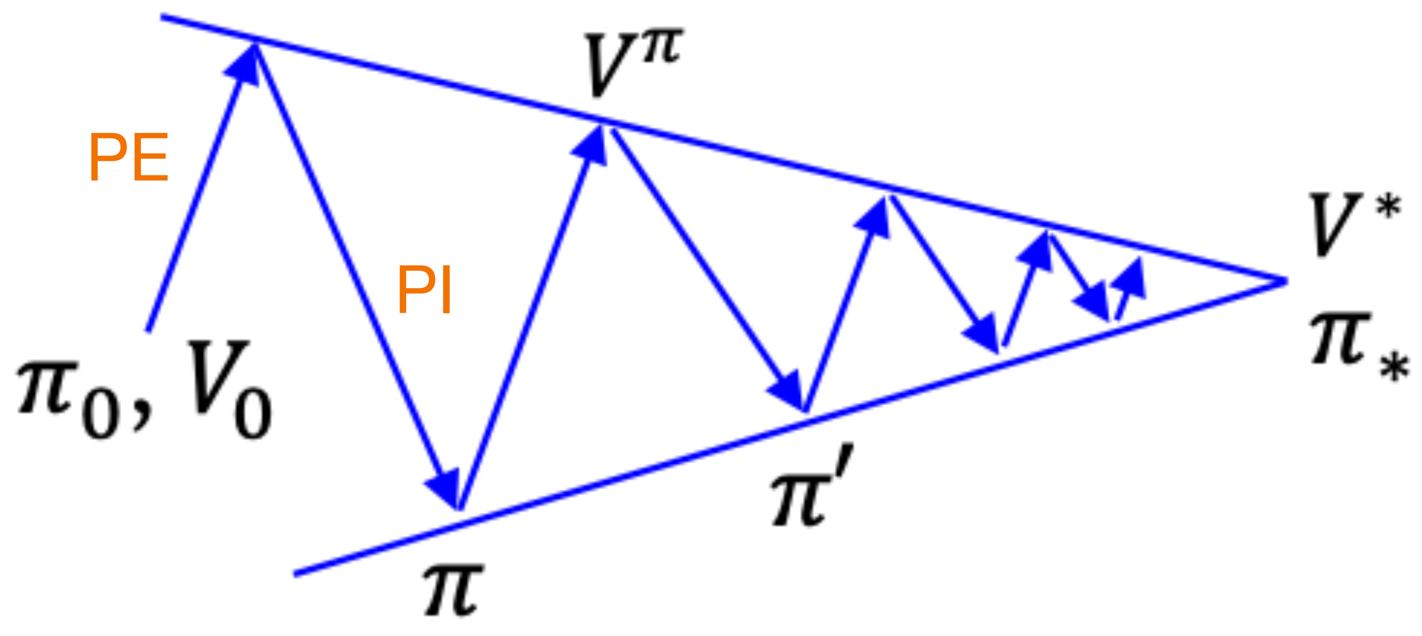
(stochastic)

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

(deterministic)

- 질문

1. policy improvement 과정에서 $\pi \leq \pi'$ 이 보장되는가?
2. $\pi = \pi'$ 일때, π' 가 optimal policy 인가?
3. 항상 converge 하는가?



Policy Improvement Theorem

1. policy improvement 과정에서 $\pi \le \pi'$ 이 보장되는가?

[Policy Improvement Theorem]

Let π and π' be two policies.
If $Q^{\pi}(s, \pi'(s)) \ge V^{\pi}(s)$ for all $s \in S$, $V^{\pi'}(s) \ge V^{\pi}(s)$ for all $s \in S$.
This implies that π' is a better policy than π.

$$
\begin{aligned}
\because v_{\pi}(s) &\le q_{\pi}(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma\, v_{\pi}(S_{t+1}) \,|\, S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma\, v_{\pi}(S_{t+1}) \,|\, S_t = s] \\
&\le \mathbb{E}_{\pi'}[R_{t+1} + \gamma\, q_{\pi}(S_{t+1}, \pi'(S_{t+1})) \,|\, S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma\, \mathbb{E}_{\pi'}[R_{t+2} + \gamma\, v_{\pi}(S_{t+2}) \,|\, S_{t+1}, A_{t+1} = \pi'(S_{t+1})] \,|\, S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma\, R_{t+2} + \gamma^2 v_{\pi}(S_{t+2})] \,|\, S_t = s] \\
&\vdots \\
&\le \mathbb{E}_{\pi'}[R_{t+1} + \gamma\, R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \,|\, S_t = s] \\
&= v_{\pi'}(s)
\end{aligned}
$$

PE : $V_{k+1}(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s))[r + \gamma V_k(s')]$

PI : $\pi'(s) = arg\ \max_a \sum_{s',r} p(s', r \,|\, s, a)[r + \gamma V^{\pi}(s')] = arg\ \max_a Q^{\pi}(s, a)$

$$
Q^{\pi}(s, \pi'(s)) \;\ge\; V^{\pi}(s) = \sum_a \pi(a \,|\, s)\; Q^{\pi}(s, a)
$$

Policy Improvement Theorem

2. $\pi = \pi'$ 일때, π' 가 optimal policy 인가?

$\downarrow$

$v_{\pi'}$ 가 bellman optimality equation 을 만족하는가?

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$$

(Bellman optimality equation)

$$\pi = \pi' \quad \leftrightarrow \quad v_\pi = v_{\pi'}$$

$$\begin{aligned}
V^\pi(s) &= \sum_{s',r} p(s', r \mid s, \pi(s))[r + \gamma V^\pi(s')] && \text{(Bellman expectation equation)} \\
&= \sum_{s',r} p(s', r \mid s, \pi'(s))[r + \gamma V^\pi(s')] && \leftarrow \quad \pi'(s) = arg \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^\pi(s')] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^\pi(s')] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V^{\pi'}(s')] \\
&= V^{\pi'}(s)
\end{aligned}$$

3. 항상 converge 하는가?

A finite MDP has finitely many policies.
>> converges to optimal policy in finitely many iterations

Policy Iteration

- **Value Iteration** 의 단점

  - 한번의 iteration 마다 $O(S^2A)$ 의 시간복잡도를 가진다.
  더불어 convergence를 위해 많은 iteration을 반복해야한다.

  - Optimal policy가 이미 고정된 이후에도, optimal state-value function이
  수렴할때까지 계속해서 계산을 진행한다.

- **Policy Iteraion** 은 Value Iteration의 두가지 단점을 보완한다.

  - 한번의 iteration 마다 $O(S^2)$ 의 시간복잡도를 가진다.

  - Optimal policy가 이미 고정되면 policy iteration이 끝난다.

Value / Policy Iteration

value iteration	policy iteraion
bellman optimality equation 사용	bellman expectation equation 사용
$V_{k+1}(s) \leftarrow \max_a \sum_{s',r} p(s',r \mid s,a)[r + \gamma V_k(s')]$	$V_{k+1}(s) \leftarrow \sum_{s',r} p(s',r \mid s,\pi(s))[r + \gamma V_k(s')]$
$\pi_*(s) = arg\max_a \sum_{s',r} p(s',r \mid s,a)[r + \gamma v_*(s')]$	$\pi'(s) = arg\max_a \sum_{s',r} p(s',r \mid s,a)[r + \gamma V^{\pi}(s')]$
$O(S^2A)$ per iteration	$O(S^2)$ per iteration
optimal state-value function 수렴할때까지 반복	optimal policy 수렴할때까지 반복

공통점

- Bellman optimality equation을 만족시키는 $V_*(s)$ 를 구한다.
- Known MDP 상황에서 계산하는 것이기 때문에 transition probability를 알아야 한다.

Dynamic Programming

- MDP를 푸는 방법에는 크게 2가지가 존재한다.

  - **Dynamic Programming (DP)** — Value Iteration / Policy Iteration

  - Reinforcement Learing (RL)

- Dynamic Programming (DP) - **model based** (known MDP)

  - Value function을 table에 저장한 후에 full-backup을 이용하여 계속해서 update 하여,
    bellman optimality equation을 만족시키도록 한다.

  - 계산량을 고려하여 $Q(s, a)$ 보다 $V(s)$ 를 사용한다. ($\because |\{s\}| \ll |\{s, a\}|$)

  - State의 개수가 많아지면 계산량이 많아진다는 단점이 존재한다 (= curse of dimensionality).