Value Iteration for Dynamic Programming

Sangkeun Jung

Copyrightⓒ2019, Sangkeun Jung, All Rights Reserved.

Bellman Equations

Bellman Expectation Equation

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(s_{t+1}) | S_t = s]$$

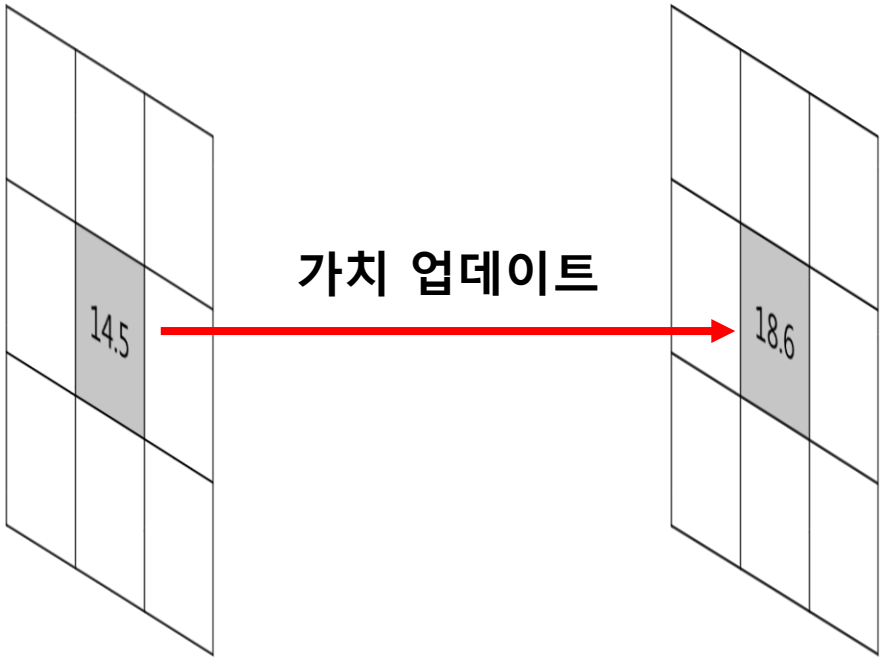
최적의 정책을 찾은 후
이를 통해 계산된
최적 Value function

Bellman Optimality Equation

$$v_*(s) = \max_a E_\pi[R_{t+1} + \gamma v_*(s_{t+1}) | S_t = s, A_t = a]$$

Copyright©2019, Sangkeun Jung, All Rights Reserved.

(Remind)



현재 (땅의) 가치 **보다 그럴듯한 (땅의) 가치**

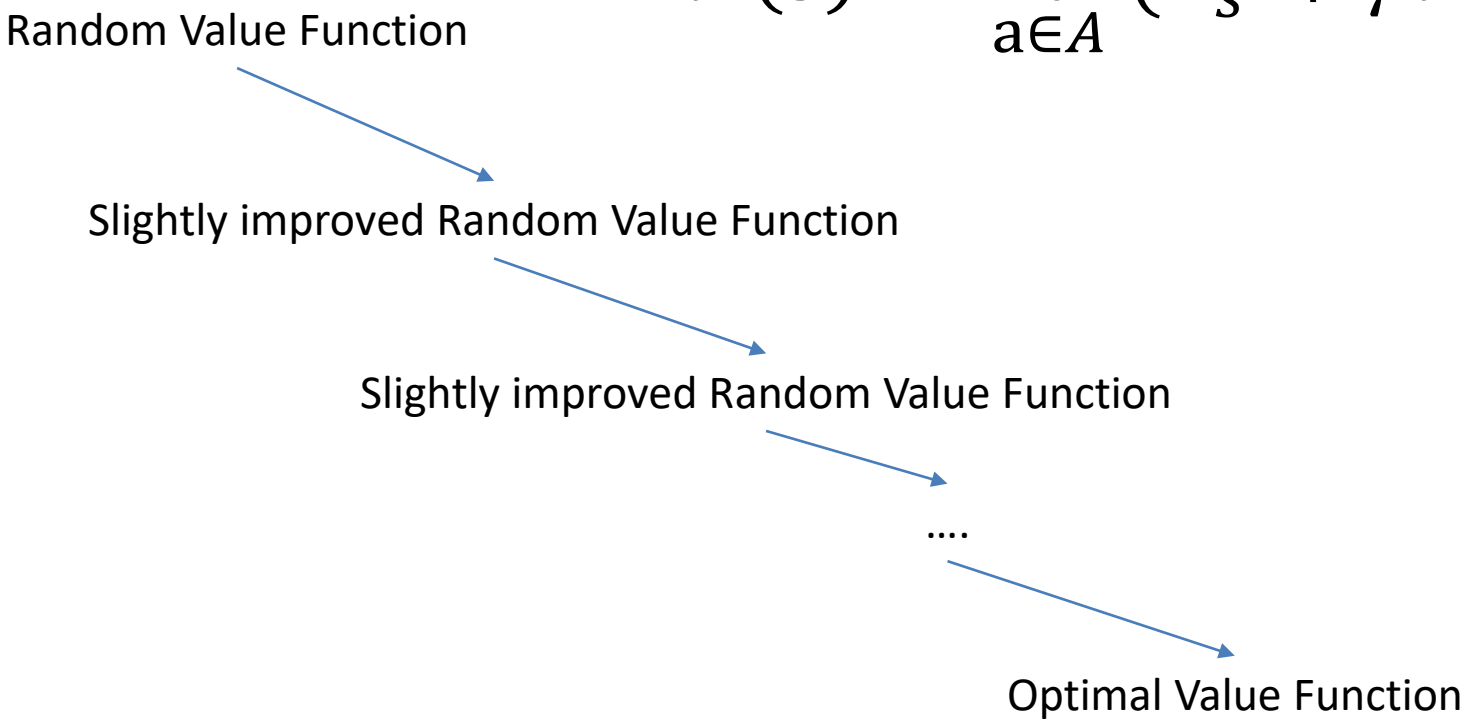
- 최초에는 땅의 가치를 모르기 때문에(동등, uniform)
- 목표점까지 갈 수 있는 땅들이 높은 값을 가질 수 있게
- 가치를 업데이트 할 수 있는 방법이 필요하다.

Copyrightⓒ2019, Sangkeun Jung, All Rights Reserved.

Value Iteration

[Insight]

$$v'(s) = \max_{a \in A}(R_s^a + \gamma v(s'))$$



Copyright ⓒ 2019, Sangkeun Jung, All Rights Reserved.

Coding

VALUE ITERATION METHOD

Copyrightⓒ2019, Sangkeun Jung, All Rights Reserved.

Implementation

- Implement
  - `def update_value(self)`
  - `def get_action(self, state)`

- Useful Methods

Method or Variables	Description
`self.env.possible_actions()`	return all possible actions in current state
`self.env.state_after_action(state, action)`	retrieve next state
`self.get_value(state)`	return value of the state
`self.discount_factor`	discount factor
`self.env.get_reward(state, action)`	get reward

Copyright ⓒ 2019, Sangkeun Jung, All Rights Reserved.

Download code

https://bit.ly/2m7jdEM

Copyrightⓒ2019, Sangkeun Jung, All Rights Reserved.