동적 프로그래밍을 통한 마르코프 결정 과정

Dynamic programming

1. Optimal substructure

- Optimal solution can be decomposed into subproblems

2. Overlapping subproblems

- Subproblems recur many times
- Solutions can be cached and reused

주어진 문제를 여러 개의 sub-problem으로 나눌 수 있고, 나눈 문제들에 대해 solution을 구하면 그 solution이 전체 문제를 푸는데 사용됨

Iterative Policy Evaluation

- **Iteratively compute until convergence**

$$\boldsymbol{V}^{k+1} = \boldsymbol{R}^{\pi} + \gamma \boldsymbol{P}^{\pi} \boldsymbol{V}^{k}$$

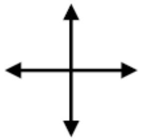
► Matrix form of Bellman expectation equation

$$V_\pi(s) = \sum_{a \in A} \pi(a|s)(R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_\pi(s'))$$

State가 10개. V = 10x1 (1로만 이루어진)
MDP 주어짐 = R이 주어졌고, P도 주어졌고, γ도 주어짐
수렴된 값이 value function.

Evaluating Random Policy in Small Gridworld

Problem setup



actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

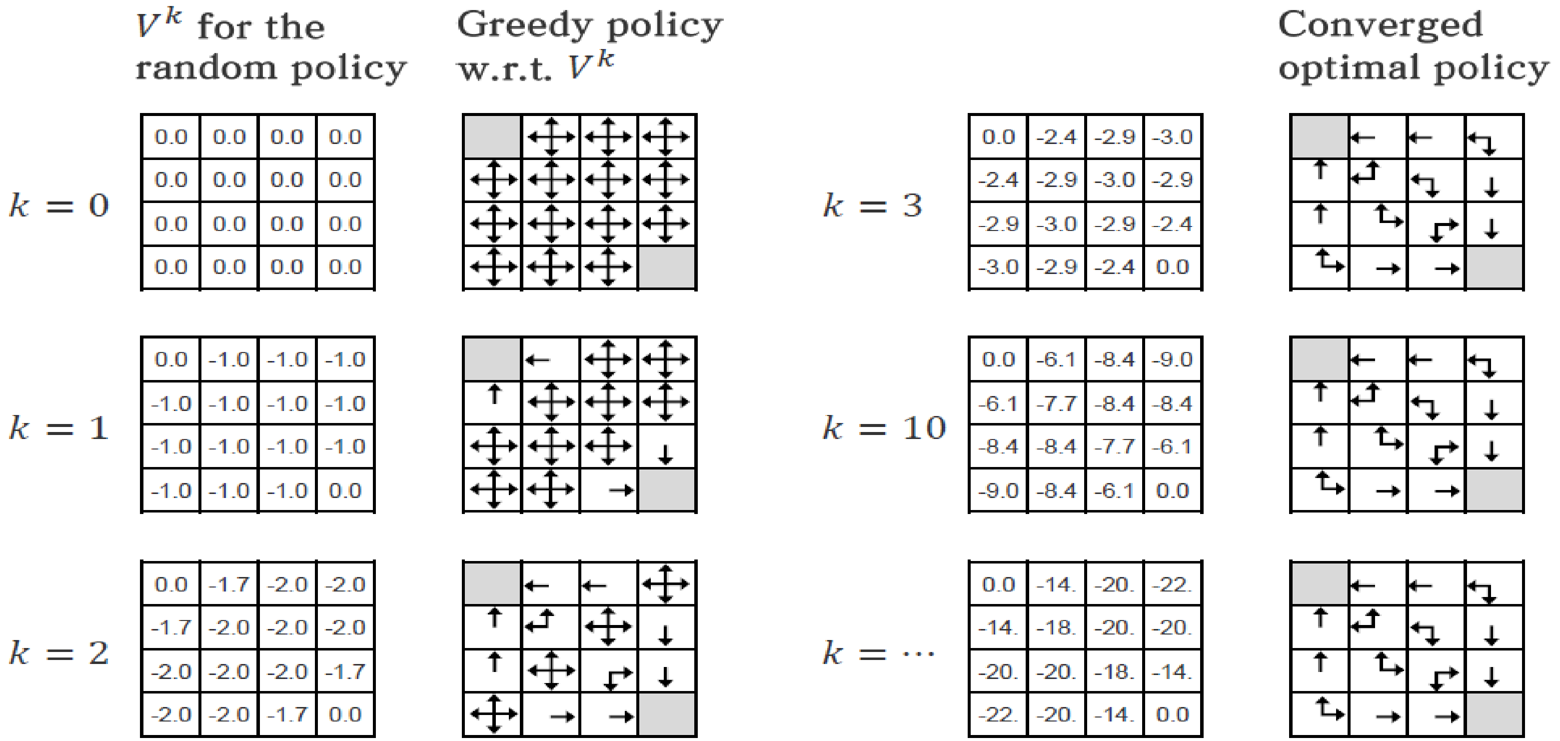
- Undiscounted episodic MDP ($\gamma = 1$)
- Terminal state: two shaded squares
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

1~14의 value function 구하기
모든 방향 0.25 확률 같음
Reward : 한 번 갈 때마다 -1
Policy : Random Policy

Evaluating Random Policy in Small Gridworld

- **V = Reward+γ+state transition probability P+V.** $R^{\pi} + \gamma P^{\pi} V^{k}$



Policy를 하나 정한 다음(ex, Random) 그 policy에 대한 value function을 구한 다음 Greedy 아이디어(value function이 최대가 되는 action을 하도록 지정. 계속 반복하면 Optimal policy 구할 수 있음.

DP Algorithms

Problem	Bellman equation	Algorithm
Prediction	Bellman expectation equation	Iterative policy evaluation
Control	Bellman expectation equation + Greedy policy improvement	Policy iteration

Prediction : policy가 주어졌을 때, MDP가 주어졌을 때 value function을 구하는 것
Control : Iterative policy Evaluation해서 value function이 구해지면 Greedy Policy Improvement를 한 뒤 Policy update 시킨 다음 반복

- Algorithms are based on state-value function $V_\pi(s)$ or $V_*(s)$
- Complexity $O(mn^2)$ per iteration, for m actions and n states
- Could also apply to action-value function $Q_\pi(s,a)$ or $Q_*(s,a)$
- Complexity $O(m^2n^2)$ per iteration

V를 주로 다루는 이유 :
Complexity가 더 적음.
V : nx1 벡터(각 state)
Q : nxm 벡터(각 state와 각 action 정의)