Reinforcement Learning

MDP, Q-learning

2021.03.11

최영제

Index

Components of Reinforcement Learning

상태(State)

- 상태(State)란 에이전트가 앞으로의 미래를 결정하기 위한 정보 혹은 에이전트가 처한 상황을 뜻함
- 모든 상태의 집합을 상태 공간(State Space)이라고 부름

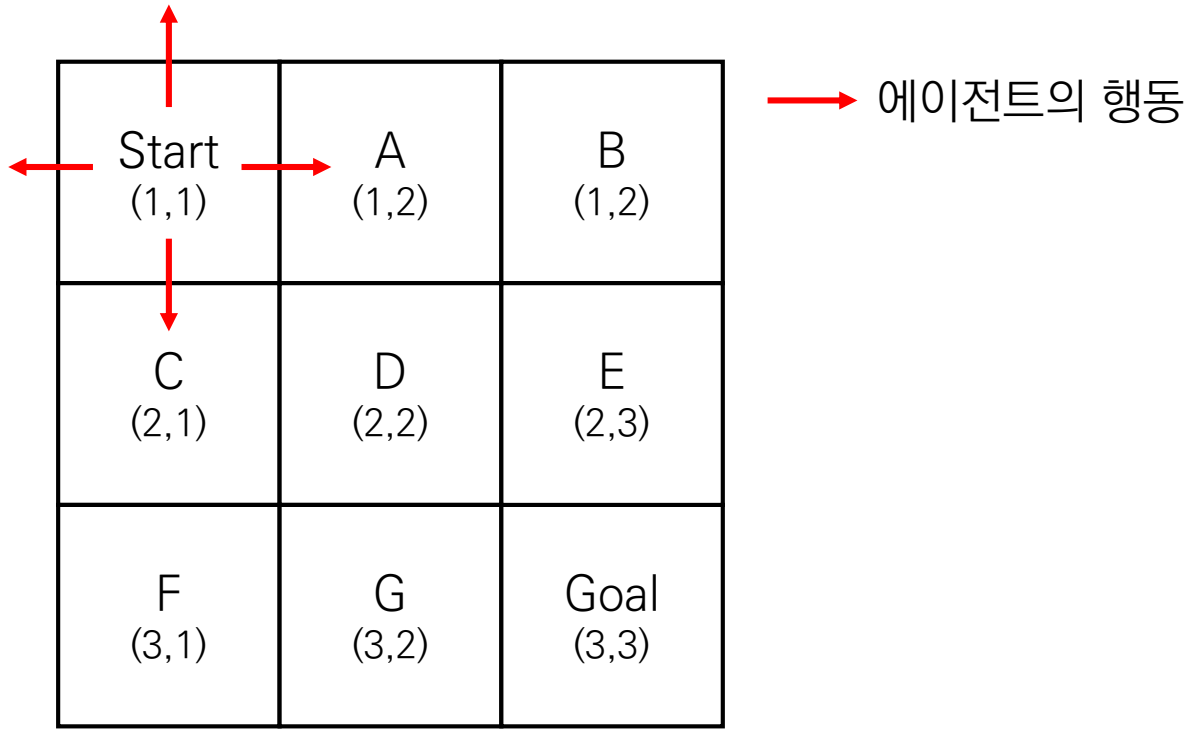
Start (1,1)	A (1,2)	B (1,2)
C (2,1)	D (2,2)	E (2,3)
F (3,1)	G (3,2)	Goal (3,3)

O 에이전트의 현재 위치

Components of Reinforcement Learning

행동(Action)

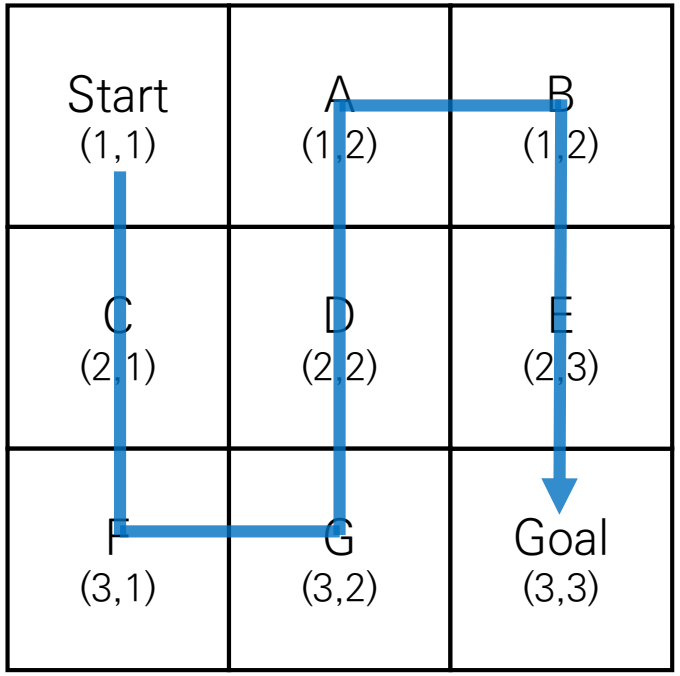
- 행동(Action)이란 에이전트가 상태에 기반하여 취할 수 있는 가능한 선택지를 뜻함
- 모든 행동 집합을 행동 공간(Action Space)이라고 부름



Components of Reinforcement Learning

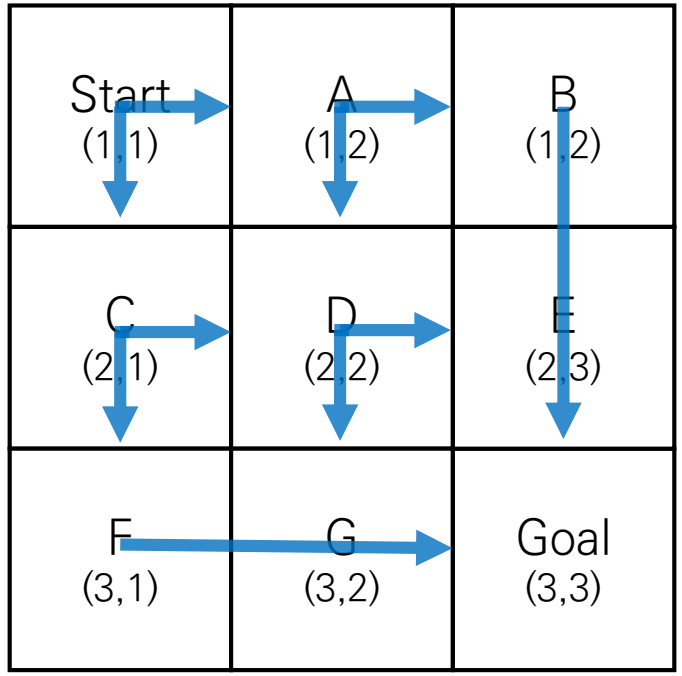
정책(Policy)

- 정책(Policy)이란, 특정 상태에서 에이전트가 행동을 취하는 규칙을 의미하며 상태를 Input으로 받아 행동에 대한 확률 분포를 출력하는 함수
- 정책은 주로 π로 표기하며 특정 정책 (π_θ)을 따랐을 때의 상태와 행동의 관계식은 $A_t = \pi_\theta(S_t)$와 같이 표기



정책 1을 따랐을 때 에이전트의 경로

Which policy is more valuable?

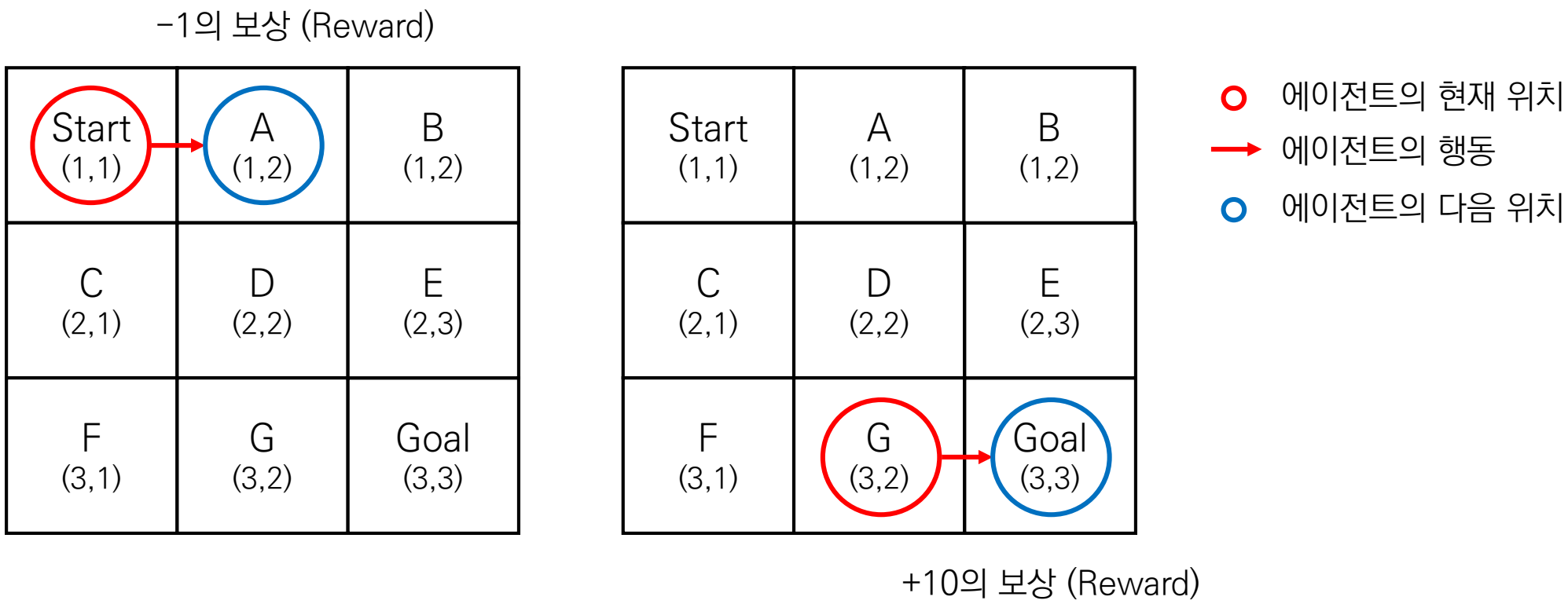


최단 경로 정책 시 에이전트의 경로

Components of Reinforcement Learning

보상(Reward)

- 현재 에이전트가 내리는 의사결정이 최적인지 판단하기 위해서는 행동에 대한 평가를 할 수 있어야 함
- 보상(Reward)이란, 에이전트가 행동을 취했을 때 얻을 수 있는 즉각적인 이득을 말함



1 SIL Components of Reinforcement Learning

감가율과 감가 누적 보상(Discount Factor & Return)

- 높은 보상을 추구하는 것은 합리적이지만 단기적인 보상이 높은 행동 만을 추구하면 장기적 관점에서의 큰 이익을 불러오는 행동을 놓칠 수 있음
- 우리가 구축하고자 하는 에이전트는 즉각적인 이익에 더하여 앞으로 얻을 수 있는 이익이 큰 행동을 취해야만 함
- 우선 에이전트가 얻은 보상들의 합을 누적 보상으로 정의하면 $\sum_{t=1}^{T} R_t$ 와 같이 표기할 수 있음
- 이를 통해 앞에서 정책1과 최단 경로 정책 중 어떤 정책이 더 나은지 평가할 수 있음(-3과 +7의 누적 보상)
- 그러나 사람은 먼 미래의 보상보다는 즉각적인 보상을 선호함
- 동일한 크기의 이익을 즉각적으로 취할 수 있다면 굳이 긴 시간 기다려서 받을 필요가 없기 때문
- 따라서 미래 가치는 감가하여 더하는 방법을 사용함
- 감가율(discount factor, γ)은 0과 1사이의 값이며 1인 경우 미래의 보상과 즉각적인 보상을 동일시하겠다는 것과 같음

$$G_t = \sum_{t=1}^{T} \gamma^{t-1} R_t$$

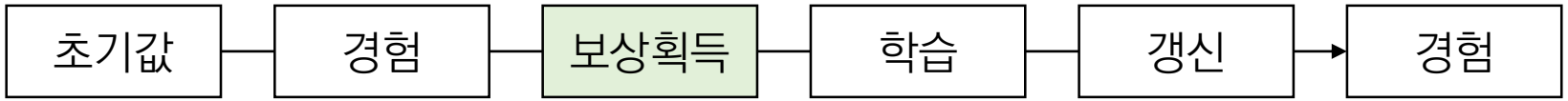
1 SIL Components of Reinforcement Learning

상태 가치 함수(State Value Function)

- 보상과 감가 누적 보상을 정의한 이유는 에이전트가 어떤 행동을 취해야 높은 보상을 취득할 수 있는지를 알기 위함
- 에이전트의 행동을 평가하는 방법은 두 가지 관점으로 나뉘는데 첫 번째는 **상태의 관점에서 특정 상태에 도달 후 앞으로 얻을 수 있는 감가 누적 보상을 확인**하는 것
- 이를 상태 가치 함수라고 부르며 다음과 같이 표기함

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$

- 만약 에이전트가 모든 상태들에 대해서 상태 가치 함수의 값을 정확히 알 수 있다면 최적의 의사결정을 할 수 있음
- T+1 시점에 도달할 수 있는 상태 중 가장 높은 가치를 갖는 상태에 도달하면 되기 때문
- 상태 가치 함수가 큰 값을 갖는다는 것은 해당 상태에 도달했을 때 앞으로 얻을 수 있는 보상이 최대라는 의미
- 그런데 미래의 보상이 포함되어 있는 G_t는 현재 시점(t)에서 알 수 없지 않나?



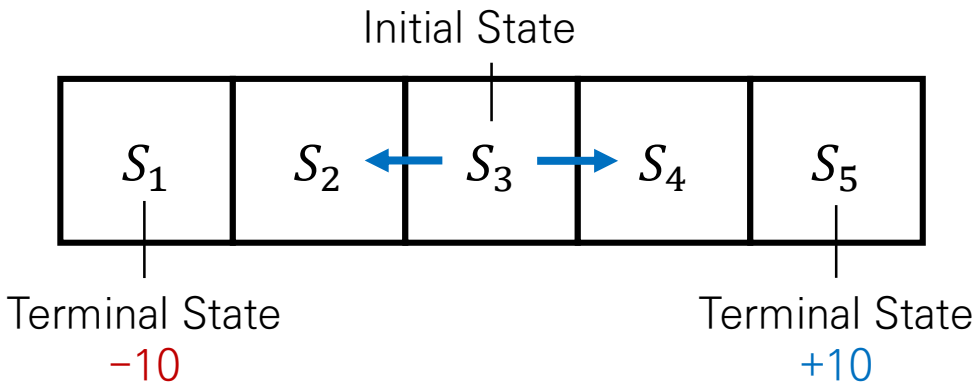
Components of Reinforcement Learning

상태 가치 함수(State Value Function)

- 에이전트가 Random Policy에 따라서 모든 상태에서 좌측 이동(Episode 01)과 모든 상태에서 우측 이동(Episode 02), 두 번의 에피소드를 진행
- 감가율이 1이라고 가정하면 Terminal State에서 얻은 보상(-10과 +10)으로 각각의 상태 가치 함수를 갱신
- Initial State에서 오른쪽으로 가는 행동을 취해야 최적임을 알 수 있음

$$G_t = \sum_{t=1}^{T} \gamma^{t-1} R_t$$

$$V_\pi(s) = E_\pi[G_t | S_t = s]$$



Initial State Value Function

$V(S) = (0, 0, 0, 0, 0)$

	Transition	Gain	State Value Function
Episode 01	$< S_3, L, S_2, L, S_1 >$	$G_1 = -10$	$V(S) = (-10, -10, -10, 0, 0)$
Episode 02	$< S_3, R, S_4, R, S_5 >$	$G_2 = +10$	$V(S) = (-10, -10, 0, 10, 10)$

Components of Reinforcement Learning

행동 가치 함수(Action Value Function)

- 에이전트의 행동의 좋고 나쁨을 평가하는 두 번째 관점은 행동의 가치를 확인하는 것

$$Q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$$

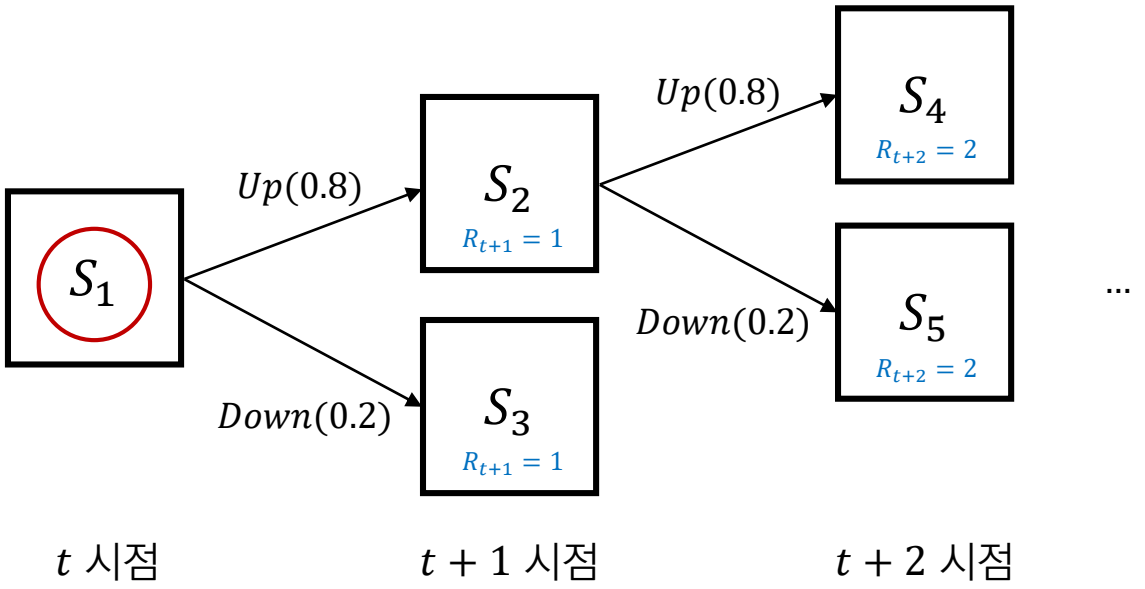
- 상태 가치 함수와 마찬가지로 에이전트가 행동 가치 함수를 정확히 알 수 있다면 최적의 의사결정이 가능
- 앞선 예제에서는 모든 상태에서 오른쪽으로 가는 행동을 취했을 때의 행동 가치 함수가 왼쪽으로 가는 행동 가치 함수보다 높은 값을 갖게 됨

	$Action$	
$State$	L	R
S_1	$Q(S_1, L) = 0$	$Q(S_1, R) = 0$
S_2	$Q(S_2, L) = -10$	$Q(S_2, R) = 0$
S_3	$Q(S_3, L) = -10$	$Q(S_3, R) = +10$
S_4	$Q(S_4, L) = 0$	$Q(S_4, R) = +10$
S_5	$Q(S_5, L) = 0$	$Q(S_5, R) = 0$

Components of Reinforcement Learning

상태 가치 함수와 행동 가치 함수의 변환

- 모든 상태에 대해서 특정 행동을 취했을 때 도달하는 다음 상태가 결정적이고 단일 상태만 존재하기에 상태 가치 함수와 행동 가치 함수가 동일함
- S_3에서 행동 R을 취했을 때의 행동 가치 함수는 $\mathrm{Q}(S_3, R) = 10$으로 $V(S_4)$ 와 같게 됨
- 상태 가치 함수와 행동 가치 함수 모두 G_t를 사용하기 때문에 당연한 결과이며 **상태 가치 함수와 행동 가치 함수는 서로 변환이 가능**함



○ 현재 시점(t)의 에이전트 위치

Components of Reinforcement Learning

상태 가치 함수와 행동 가치 함수의 변환

- 행동 가치 함수 $Q_\pi(S_1, Up)$를 구해보면 즉각적인 보상 1과 다음 시점의 감가된 상태 가치 함수 $\gamma V_\pi(S_2)$의 합으로 표현이 가능

$$Q_\pi(S_1, Up) = 1 + \gamma V_\pi(S_2)$$

- 동일하게 $Q_\pi(S_1, Down)$는 다음과 같이 표현이 가능

$$Q_\pi(S_1, Down) = 1 + \gamma V_\pi(S_3)$$

- 행동 가치 함수는 즉각적인 보상과 다음 시점의 감가된 상태 가치 함수의 합으로 표현이 가능함($\mathcal{R}_s^a = R_{t+1} = r',\ S_{t+1} = s'$)

$$Q_\pi(s, a) = r' + \gamma V_\pi(s')$$

- 마찬가지로 상태 가치 함수도 행동 가치 함수로 변환이 가능하며, 두 가지 행동이 가능한 S_1의 가치 함수는 다음과 같이 나타낼 수 있음

$$V_\pi(S_1) = 0.8 * Q_\pi(S_1, Up) + 0.2 * Q_\pi(S_1, Down)$$

- 마찬가지로 S_2의 가치 함수는 다음과 같음

$$V_\pi(S_2) = 0.8 * Q_\pi(S_4, Up) + 0.2 * Q_\pi(S_5, Down)$$

- 일반화 시키면 다음과 같음

$$V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s, a)$$

Components of Reinforcement Learning

상태 가치 함수와 행동 가치 함수의 변환

- 즉 상태 가치 함수는 다음과 같이 표현할 수 있음

$$V_\pi(S_1) = 0.8 * Q_\pi(S_1, Up) + 0.2 * Q_\pi(S_1, Down)$$
$$= 0.8 * \big(1 + \gamma V_\pi(S_2)\big) + 0.2 * \big(1 + \gamma V_\pi(S_3)\big)$$

$$Q_\pi(S_1, Up) = 1 + \gamma V_\pi(S_2)$$
$$Q_\pi(S_1, Down) = 1 + \gamma V_\pi(S_3)$$

- 변경된 식을 살펴보면 t 시점의 상태 가치 함수 $V_\pi(S_1)$는 즉각적인 보상과 다음 시점(t+1)의 상태 가치 함수 $V_\pi(S_2)$, $V_\pi(S_3)$의 가중합으로 정의됨
- 따라서 t+1 시점의 가치 함수인 $V_\pi(S_2)$, $V_\pi(S_3)$값을 알면 $V_\pi(S_1)$ 값을 도출할 수 있게 됨
- 동일한 원리로 $V_\pi(S_2)$는 t+2 시점의 보상과 가치 함수인 $V_\pi(S_4)$, $V_\pi(S_5)$를 사용하여 정의할 수 있으며 $V_\pi(S_4), V_\pi(S_5)$ 값을 알면 $V_\pi(S_2)$ 값 또한 쉽게 구할 수 있음
- 만약 **종료가 확실시되는 환경이라면 마지막 시점에서 획득한 보상을 토대로 시점을 거슬러 올라가 모든 상태 가치 함수 값을 갱신**할 수 있게됨
- 위와 같이 특정 시점(t)의 가치 함수는 보상과 다음 시점(t+1)의 가치 함수로 정의할 수 있으며 이를 토대로 가치 함수를 도출하는 과정이 바로 벨만 방정식의 핵심 내용

Components of Reinforcement Learning

벨만 방정식

- 벨만 방정식이란 **현재 시점과 다음 시점 사이의 가치 함수의 관계식이자 가치 함수를 도출할 수 있는 방정식**임
- 벨만 방정식의 도출 과정은 다음과 같음

$$V_\pi(s) = E_\pi[G_t|S_t = s]$$
$$= E_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots |S_t = s\right]$$

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} \ldots)|S_t = s]$$
$$= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s]$$
$$= E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1})|S_t = s]$$

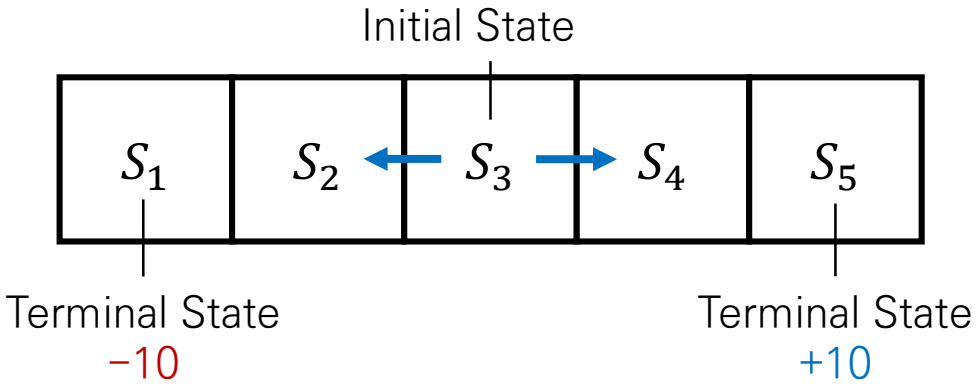
- 우측 식을 확인하면 t시점의 가치 함수는 보상과 t+1시점의 가치 함수로 구성되어 있음
- 즉 t+1 시점의 가치 함수와 보상 값을 알 수 있다면 t시점의 가치 함수를 갱신할 수 있으며 이는 행동 가치 함수도 마찬가지임

$$Q_\pi(s, a) = E_\pi[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

Components of Reinforcement Learning

벨만 방정식

- 앞의 예에서 모두 좌측으로 이동한 에피소드는 다음과 같은 벨만 방정식을 통해서 지나온 상태들에 대해서 값을 갱신함



$$V(S_3) = 0 + \gamma V(S_2)$$

$$V(S_2) = 0 + \gamma V(S_1)$$

$$V(S_1) = -10$$

- 이러한 가치 함수는 정책에 따라서 다양한 값을 가질 수 있는데 존재하는 모든 가치 함수 중 최대값을 갖는 가치 함수를 최적 가치 함수라고 함
- **최적 가치 함수를 도출할 수 있는 정책을 최적 정책**이라고 하며 강화학습에서 풀고자하는 목적에 해당함
- 벨만 방정식이 가치 함수를 도출할 수 있는 식이라면 **벨만 최적 방정식은 최적 가치 함수를 도출할 수 있는 식**임

$$V_*(s) = \max_a [s' + \gamma V_*(s')]$$

$$Q_*(s,a) = r' + \gamma \max_{a'} Q_\pi(s', a')$$

1 SIL Components of Reinforcement Learning

벨만 최적 방정식

- 벨만 최적 방정식을 풀기 위해서는 경험적인 방법을 통해서 값을 찾아가야 함
- 경험적인 방법이란 임의의 가치 함수로 초기화 후 에이전트의 경험에 따라 가치 함수를 업데이트하는 과정을 반복하는 것
- 에이전트의 경험을 사용하는 방법에 따라 동적 계획법(Dynamic Programming, DP), 몬테 카를로 학습(Monte-Carlo Learning, MC), 그리고 시간차 학습(Temporal Difference Learning, TD)으로 구분할 수 있음
- 세 가지 방법은 강화학습에서 환경에 대한 가정인 모델(Model)을 알고 있는 경우와 모르는 경우에 따라서 다르게 적용됨
- 환경에 대한 가정, **모델이란 상태 전이 행렬**을 뜻하며, 넓게 보면 **에이전트가 특정 상태에 놓여 있을 분포(Stationary Distribution)**를 말함
- 모델을 알고 있을 때 사용하는 방법은 DP이며 그렇지 않은 경우에 사용하는 것은 MC와 TD임
- 바둑에서 첫 수를 두었을 때 존재할 수 있는 모든 상태에 대해서 분포를 미리 모델링 한 후에 강화학습을 진행하지 않듯이 모델을 모르는 경우가 많음
- 따라서 MC와 TD를 집중적으로 다루며 이를 Model-Free RL이라고 부름

Components of Reinforcement Learning

몬테 카를로 학습과 시간차 학습

- 몬테 카를로 학습과 시간차 학습의 차이는 **학습의 대상이 되는 Target**을 무엇으로 사용하느냐임
- 몬테 카를로 학습은 학습 대상에 감가 누적 보상인 G_t를 사용함
- 몬테 카를로 학습에서 가치 함수를 갱신하는 식은 다음과 같음

1) 에이전트가 상태 s를 방문할 경우 회수를 센다 $N(s) = N(s) + 1$
2) 각 Episode 에서 상태 s 발생한 감가 누적 보상을 합한다 $S(s) = S(s) + G_t$
3) 이를 나눔으로써 평균을 구하여 $V(s)$를 갱신한다 $V(s) = \frac{S(s)}{N(s)}$

- 즉 에이전트가 경험을 통해서 획득한 G_t의 평균값으로 가치 함수를 갱신함
- 다만 위의 평균을 구하는 공식을 다음과 같이 변형하여 사용함

$$V(S_i) = \frac{G_1 + G_2 + \cdots + G_{i-1} + G_i}{n}$$

$$= \frac{G_1 + G_2 + \cdots + G_{i-1}}{n} + \frac{G_i}{n}$$

$$= \frac{n-1}{n}\frac{G_1 + G_2 + \cdots + G_{i-1}}{n-1} + \frac{G_i}{n}$$

$$= \frac{n-1}{n}V(S_{i-1}) + \frac{G_i}{n}$$

$$= V(S_{i-1}) - \frac{1}{n}V(S_{i-1}) + \frac{G_i}{n}$$

$$= V(S_{i-1}) + \frac{1}{n}[G_i - V(S_{i-1})]$$

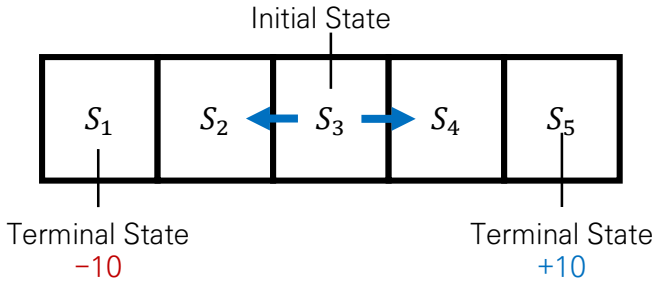
Components of Reinforcement Learning

몬테 카를로 학습과 시간차 학습

- 변경된 식을 사용하면 직전 Episode의 상태 가치 함수 $V(S_{i-1})$와 현재 Episode의 감가 누적 보상 을 이용하여 상태 가치 함수 G_i를 구할 수 있음

$$V(S_t) = V(S_t) + \frac{1}{N(s)}(G_t - V(S_t)$$
$$= V(S_t) + \alpha(G_t - V(S_t))$$

- 이렇게 평균을 구하는 방법을 Incremental mean이라고 하며 위식을 해석하면 현재의 가치 함수 + step-size*(감가 누적 보상과 현재 가치와의 차이) 만큼 갱신을 한다고 보면 됨



	Transition	Gain	State Value Function
Episode 01	$< S_3, L, S_2, L, S_1 >$	$G_1 = -10$	$V(S) = (-10, -10, -10, 0, 0)$
Episode 02	$< S_3, R, S_4, R, S_5 >$	$G_2 = +10$	$V(S) = (-10, -10, 0, 10, 10)$

Components of Reinforcement Learning

몬테 카를로 학습과 시간차 학습

- 다만 몬테 카를로 학습은 G_t를 사용하기에 에피소드가 끝나야지만 학습이 가능하다는 단점이 있음
- 시간차 학습 TD-learning은 이러한 단점에서 벗어나 G_t를 사용하는 대신 다음 시점의 가치 함수를 target으로 삼아 학습을 진행함
- 즉 식이 다음처럼 변함

$$V(S_t) = V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

- 이러한 시간차 학습은 에피소드가 끝나지 않아도 학습이 가능하여 보다 다양한 상황에서 적용이 가능하나
- 실제 값 G_t를 사용하는 MC와 비교하여 가치 함수의 초깃값에 민감하게 영향을 받으며 학습 대상으로 실제 값이 아닌 기존의 가치 함수를 사용하기 때문에 오차가 발생할 수 있는(=Bias가 높은) 특징을 가지고 있음
- 추가로 MC는 low bias high variance, TD는 high bias low variance

Q-learning

Q-learning

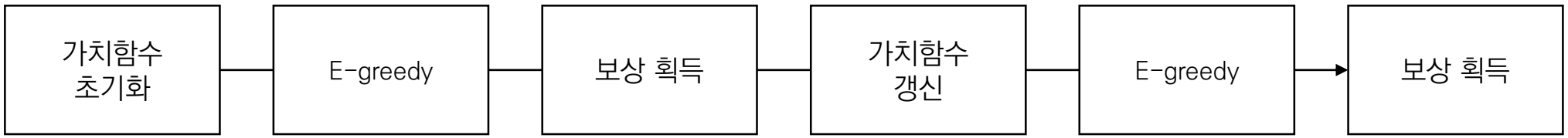
- Q-learning은 TD-learning의 대표적인 방법이며 가치 함수를 갱신하는 식은 다음과 같음

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \underset{\acute{a}}{max} Q(S_{t+1}, \acute{a}) - Q(S_t, A_t)]$$

- 기존 TD-Prediction과 구조가 동일하며, 다른 점은 상태 가치 함수가 행동 가치 함수로 변경되었다는 점과 TD-Target에 해당하는 감가된 다음 가치 함수에 max 연산자가 들어갔다는 것

- Q-learning은 행동 가치 함수를 갱신할 때, 즉각적인 보상 R_{t+1} 과 감가된 다음 상태에서의 최대 행동 가치(Maximum Q-Value), $\gamma \underset{\acute{a}}{max} Q(S_{t+1}, \acute{a})$ 를 목표로 삼아 학습을 진행

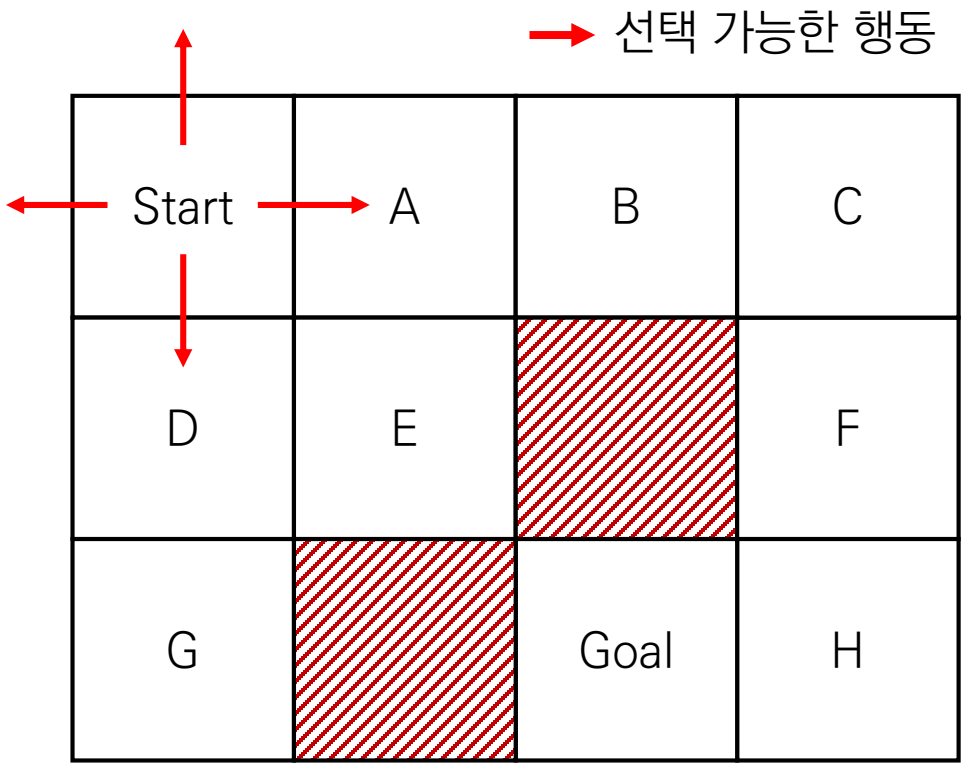
- Q-learning의 동작 방식은 다음과 같음



Q-learning

Q-learning

- 다음과 같은 환경이 있다고 가정



- Task : 최단 경로로 Goal에 도달
- State : Grid World의 좌표 e.g. (1,1)
- Action : 상, 하, 좌, 우
- Reward
  - Time Penalty : -1
  - 위치 변화가 없을 경우 : -2
  - 붉은색 장애물에 부딪힐 경우 : -10

Q-learning

Q-learning

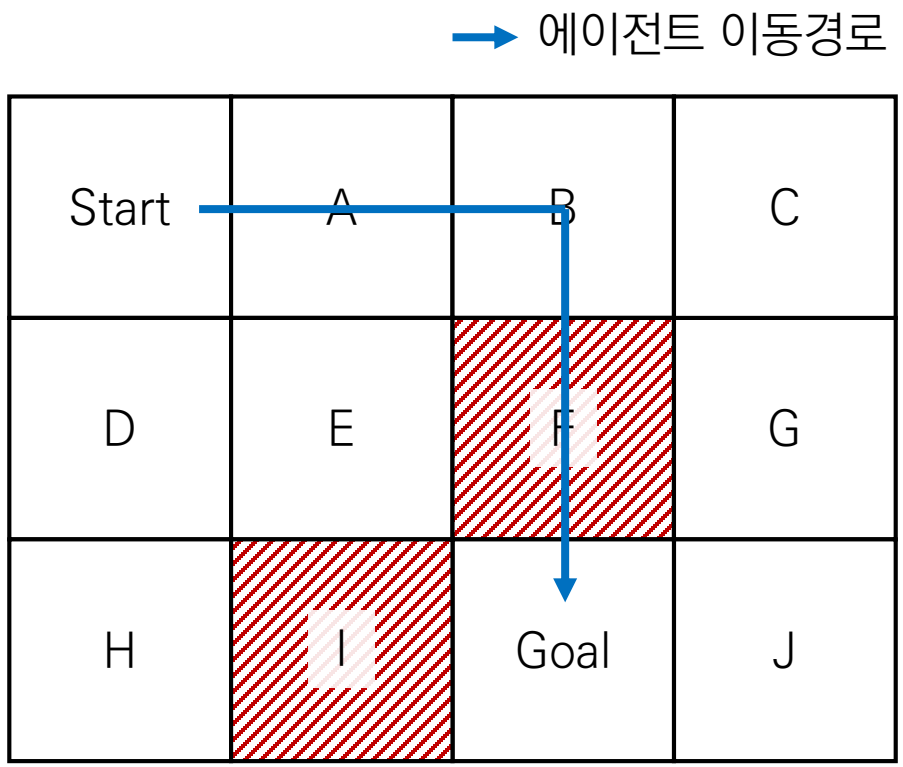
- 가치 함수는 다음과 같이 초기화 되어있음

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	0.5	0.5	0.5	0.5
A	0.5	0.5	0.5	0.5
B	0.5	0.5	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	0.5	0.5	0.5
E	0.5	0.5	0.5	0.5
F	0.5	0.5	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	0.5
I	0.5	0.5	0.5	0.5
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

Q-learning

Q-learning

- 초기에는 모든 행동 가치가 동일하기 때문에 에이전트는 랜덤하게 움직일 것
- 만약 에이전트의 움직임이 다음과 같이 [Start-A-B-F-Goal]로 움직였을 때 Q-Value의 업데이트 과정은 다음과 같음



- 상태-행동 (Start,R)의 Q-Value 업데이트
  $Q(Start, R) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (A,R)의 Q-Value 업데이트
  $Q(A, R) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (B,D)의 Q-Value 업데이트
  $Q(B, D) = 0.5 + 1 * (-10 + 0.9 * 0.5 - 0.5) = -9.55$
- 상태-행동 (F,D)의 Q-Value 업데이트
  $Q(F, D) = 0.5 + 1 * (1 + 0.9 * 0 - 0.5) = 1$

Q-learning

Q-learning

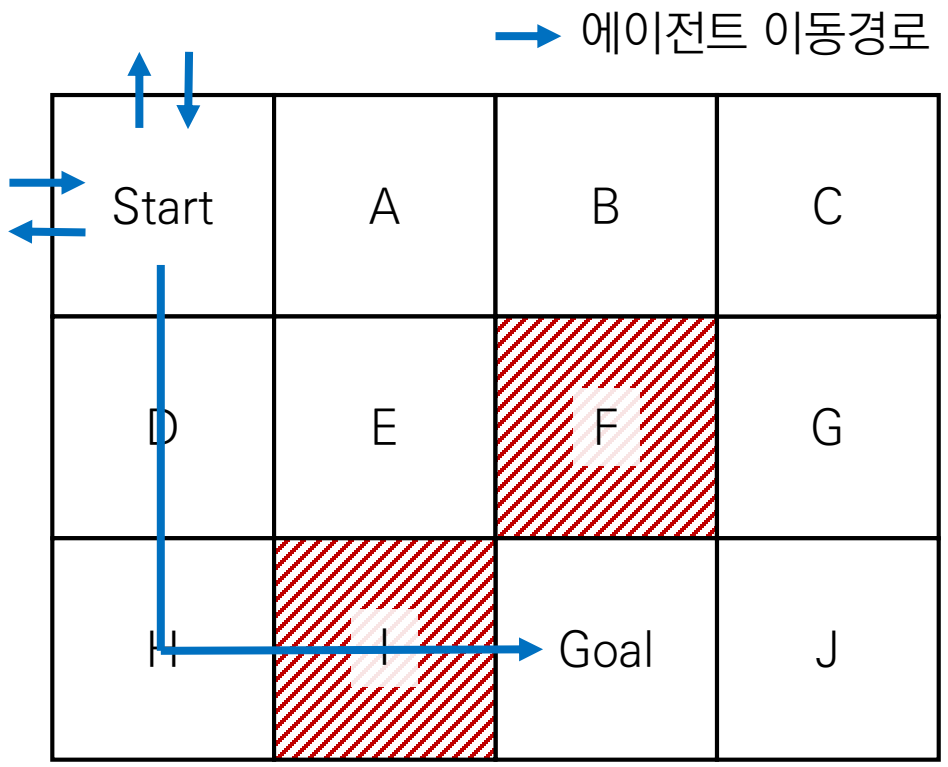
- 이를 토대로 가치 함수(Q-table)를 갱신함

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	0.5	0.5	0.5	0.5
A	0.5	0.5	0.5	0.5
B	0.5	0.5	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	0.5	0.5	0.5
E	0.5	0.5	0.5	0.5
F	0.5	0.5	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	0.5
I	0.5	0.5	0.5	0.5
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	0.5	0.5	0.5	-0.55
A	0.5	0.5	0.5	-0.55
B	0.5	-9.55	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	0.5	0.5	0.5
E	0.5	0.5	0.5	0.5
F	0.5	1	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	0.5
I	0.5	0.5	0.5	0.5
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

Q-learning

- 갱신된 Q-table을 토대로 e-greedy 정책을 따라 에이전트가 [Start-Start-D-H-I-Goal]로 움직였다고 가정



- 상태-행동 (Start,U)의 Q-Value 업데이트
  $Q(Start, U) = 0.5 + 1 * (-2 + 0.9 * 0 - 0.5) = -1.55$
- 상태-행동 (Start,L)의 Q-Value 업데이트
  $Q(Start, L) = 0.5 + 1 * (-2 + 0.9 * 0 - 0.5) = -1.55$
- 상태-행동 (Start,D)의 Q-Value 업데이트
  $Q(Start, D) = 0.5 + 1 * (-1 + 0.9 * 0 - 0.5) = -0.55$
- 상태-행동 (D,D)의 Q-Value 업데이트
  $Q(D, D) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (H,R)의 Q-Value 업데이트
  $Q(H, R) = 0.5 + 1 * (-10 + 0.9 * 0.5 - 0.5) = -9.55$
- 상태-행동 (I,R)의 Q-Value 업데이트
  $Q(I, R) = 0.5 + 1 * (1 + 0.9 * 0 - 0.5) = 1$

Q-learning

Q-learning

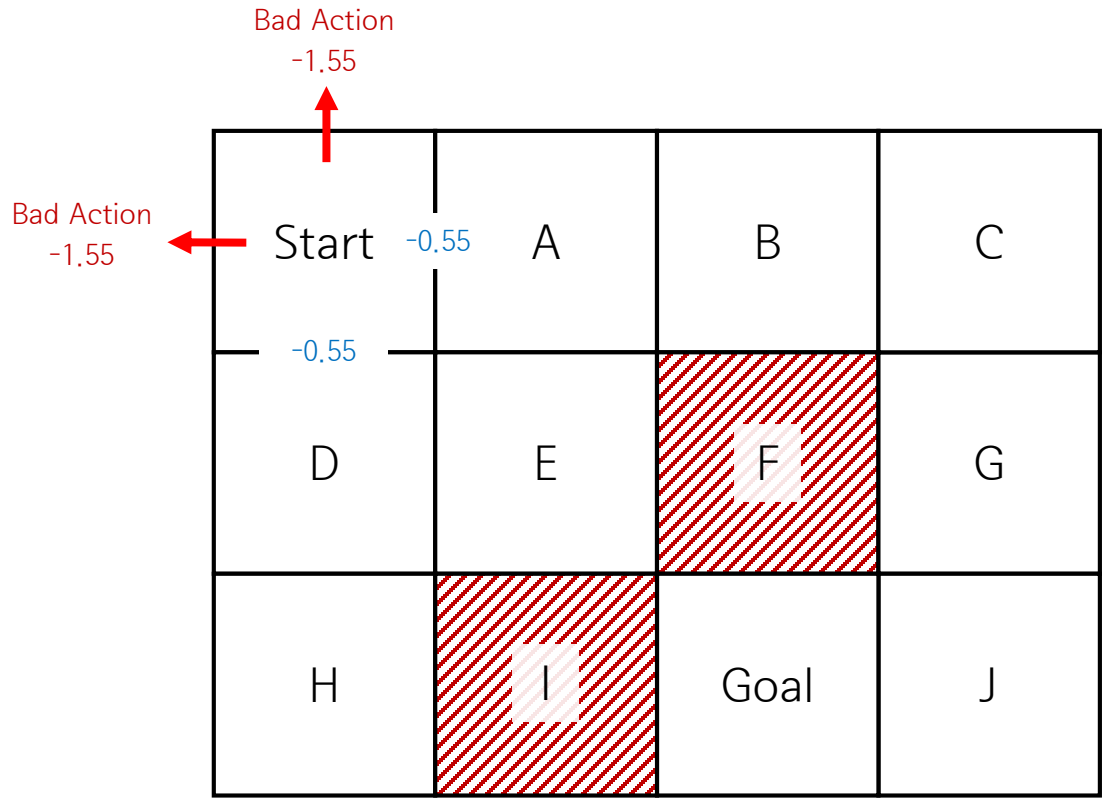
- 이를 토대로 가치 함수(Q-table)를 갱신함

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	0.5	0.5	0.5	-0.55
A	0.5	0.5	0.5	-0.55
B	0.5	-9.55	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	0.5	0.5	0.5
E	0.5	0.5	0.5	0.5
F	0.5	1	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	0.5
I	0.5	0.5	0.5	0.5
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	-1.55	-0.55	-1.55	-0.55
A	0.5	0.5	0.5	-0.55
B	0.5	-9.55	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	-0.55	0.5	0.5
E	0.5	0.5	0.5	0.5
F	0.5	1	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	-9.55
I	0.5	0.5	0.5	1
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

Q-learning

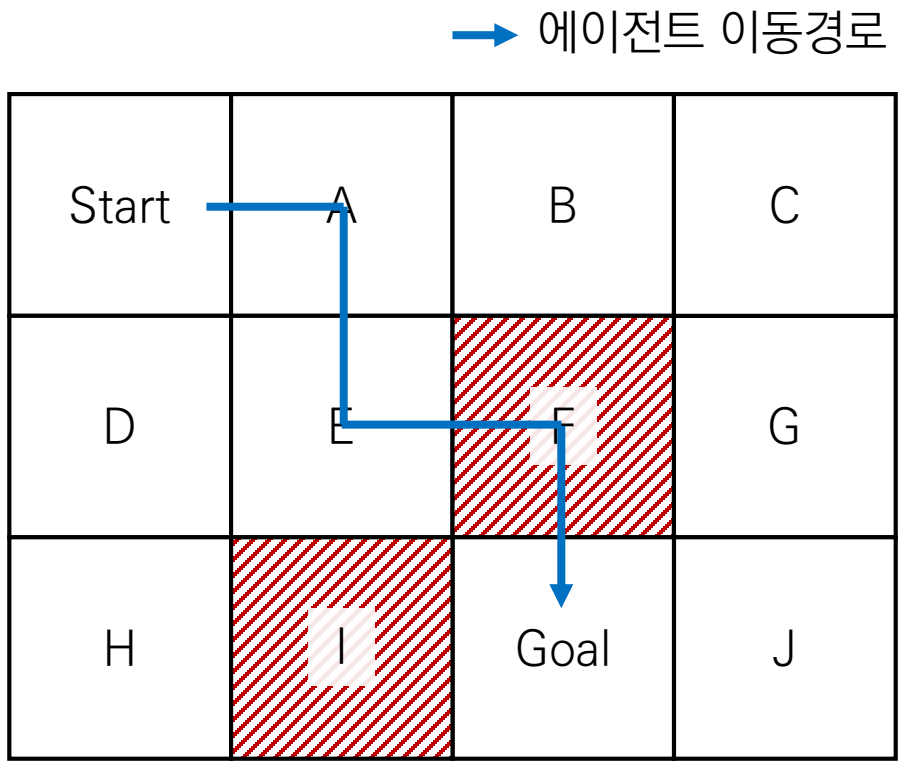
- 현재까지 갱신된 Q-table로 봤을 때 Start 상태에서 위로 가는 행동과 왼쪽으로 가능 행동은 좋지 않다는 것을 에이전트는 알 수 있음



Q-learning

Q-learning

- 갱신된 Q-table을 토대로 e-greedy 정책을 따라 에이전트가 [Start-A-E-F-Goal]의 경로로 움직였다고 가정



- 상태-행동 (Start,R)의 Q-Value 업데이트

$$Q(Start, R) = -0.55 + 1 * (-1 + 0.9 * 0.5 - (-0.55)) = -0.55$$

- 상태-행동 (A,D)의 Q-Value 업데이트

$$Q(A, D) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$$

- 상태-행동 (E,R)의 Q-Value 업데이트

$$Q(E, R) = 0.5 + 1 * (-10 + 0.9 * 1 - 0.5) = -9.1$$

- 상태-행동 (F,D)의 Q-Value 업데이트

$$Q(F, D) = 1 + 1 * (1 + 0.9 * 0 - 1) = 1$$

Q-learning

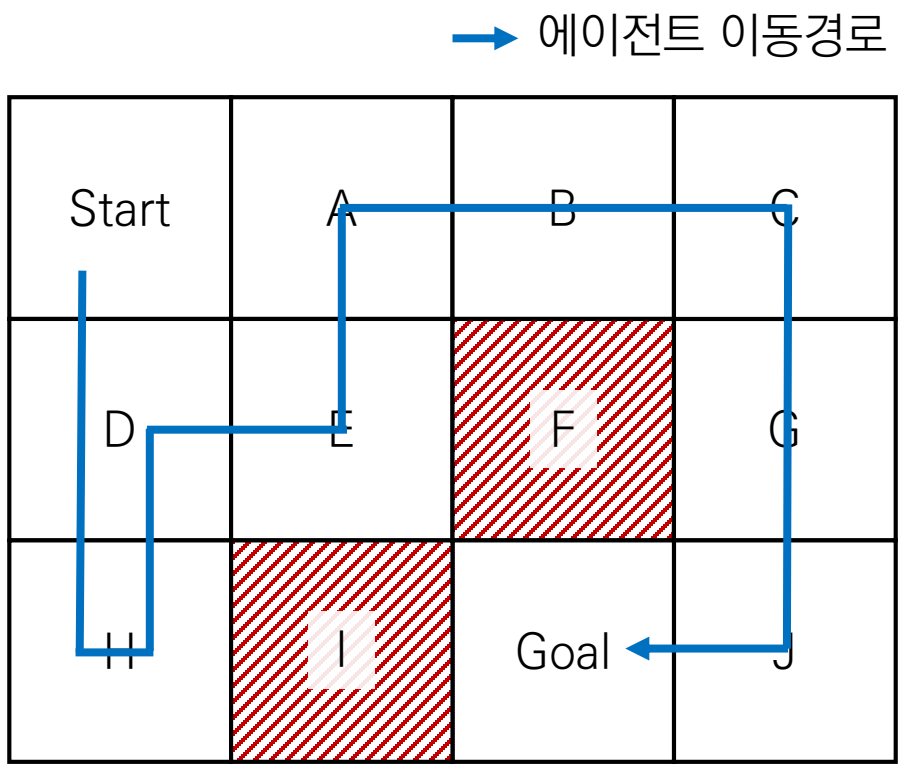
- 이를 토대로 가치 함수(Q-table)를 갱신함

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	-1.55	-0.55	-1.55	-0.55
A	0.5	0.5	0.5	-0.55
B	0.5	-9.55	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	-0.55	0.5	0.5
E	0.5	0.5	0.5	0.5
F	0.5	1	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	-9.55
I	0.5	0.5	0.5	1
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	-1.55	-0.55	-1.55	-0.55
A	0.5	-0.55	0.5	-0.55
B	0.5	-9.55	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	-0.55	0.5	0.5
E	0.5	0.5	0.5	-9.1
F	0.5	1	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	-9.55
I	0.5	0.5	0.5	1
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

Q-learning

- 갱신된 Q-table을 토대로 e-greedy 정책을 따라 에이전트가 [Start-D-H-D-E-A-B-C-G-J-Goal]의 경로로 움직였다고 가정



- 상태-행동 (Start,D)의 Q-Value 업데이트
  $Q(Start, D) = -0.55 + 1 * (-1 + 0.9 * 0.5 - (-0.55)) = -0.55$
- 상태-행동 (D,D)의 Q-Value 업데이트
  $Q(D, D) = -0.55 + 1 * (-1 + 0.9 * 0.5 - (-0.55)) = -0.55$
- 상태-행동 (H,U)의 Q-Value 업데이트
  $Q(H, U) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (D,R)의 Q-Value 업데이트
  $Q(D, R) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (E,U)의 Q-Value 업데이트
  $Q(E, U) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (A,R)의 Q-Value 업데이트
  $Q(A, R) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (B,R)의 Q-Value 업데이트
  $Q(B, R) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (C,D)의 Q-Value 업데이트
  $Q(C, D) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (G,D)의 Q-Value 업데이트
  $Q(G, D) = 0.5 + 1 * (-1 + 0.9 * 0.5 - 0.5) = -0.55$
- 상태-행동 (J,L)의 Q-Value 업데이트
  $Q(J, L) = 0.5 + 1 * (1 + 0.9 * 0 - 0.5) = 1$

Q-learning

- 이를 토대로 가치 함수(Q-table)를 갱신함

State	Up (U)	Down (D)	Left (L)	Right (R)
Start	-1.55	-0.55	-1.55	-0.55
A	0.5	-0.55	0.5	-0.55
B	0.5	-9.55	0.5	0.5
C	0.5	0.5	0.5	0.5
D	0.5	-0.55	0.5	0.5
E	0.5	0.5	0.5	-9.1
F	0.5	1	0.5	0.5
G	0.5	0.5	0.5	0.5
H	0.5	0.5	0.5	-9.55
I	0.5	0.5	0.5	1
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	0.5	0.5

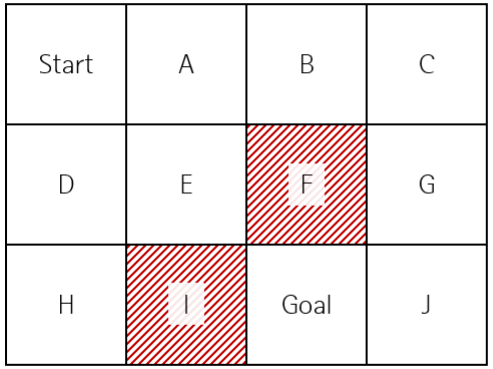
State	Up (U)	Down (D)	Left (L)	Right (R)
Start	-1.55	-0.55	-1.55	-0.55
A	0.5	-0.55	0.5	-0.55
B	0.5	-9.55	0.5	-0.55
C	0.5	-0.55	0.5	0.5
D	0.5	-0.55	0.5	-0.55
E	-0.55	0.5	0.5	-9.1
F	0.5	1	0.5	0.5
G	0.5	-0.55	0.5	0.5
H	-0.55	0.5	0.5	-9.55
I	0.5	0.5	0.5	1
Goal	0.5	0.5	0.5	0.5
J	0.5	0.5	1	0.5

Q-learning

- 현재까지 도출한 상황으로 봤을 때 다음과 같은 갱신 규칙을 알 수 있음

1) 에이전트의 다음 상태가 장애물이나 Goal 이 아닌 행동이라면 -0.55의 Q-Value 로 갱신됨
2) 에이전트의 다음 상태가 제자리일 경우 -1.55의 Q-Value 로 갱신됨
3) 에이전트의 다음 상태가 장애물일 경우 -9.1의 Q-Value 로 갱신됨.

- 따라서 일괄 갱신하면 다음과 같음

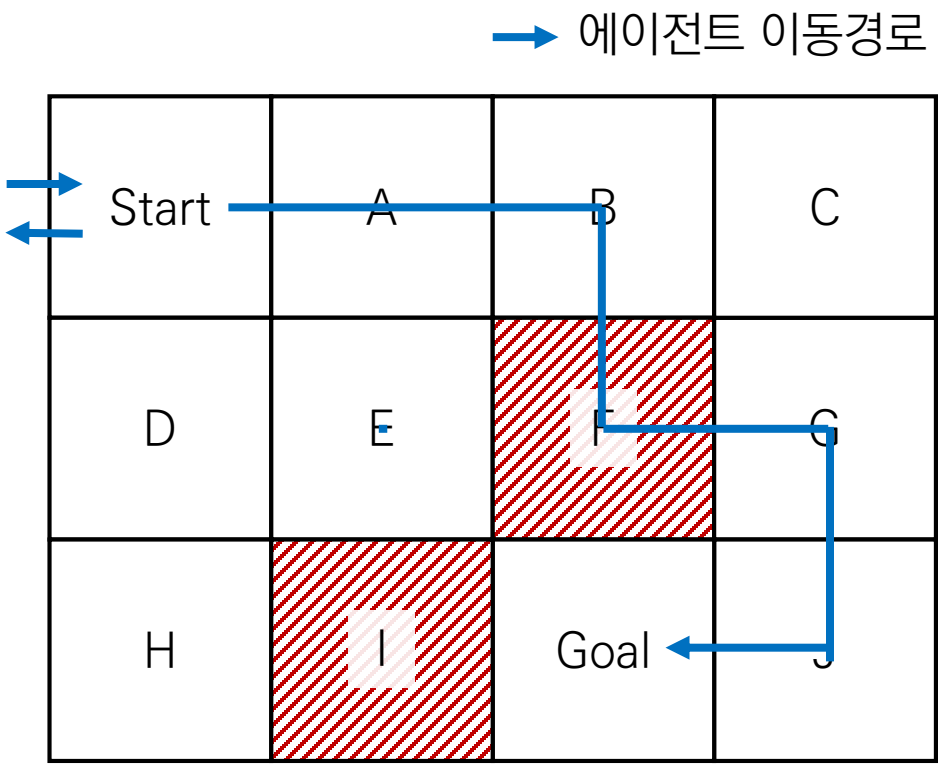


State	Up (U)	Down (D)	Left (L)	Right (R)
Start	-1.55	**-0.55**	-1.55	**-0.55**
A	-1.55	**-0.55**	**-0.55**	**-0.55**
B	-1.55	-9.1	**-0.55**	**-0.55**
C	-1.55	**-0.55**	**-0.55**	-1.55
D	**-0.55**	**-0.55**	-1.55	**-0.55**
E	**-0.55**	-9.1	**-0.55**	-9.1
F	-0.55	**1**	-0.55	-0.55
G	-0.55	**-0.1**	-9.1	-1.55
H	**-0.55**	-1.55	-1.55	-9.1
I	-0.55	-1.55	-0.55	**1**
Goal	0.5	0.5	0.5	0.5
J	-0.55	-1.55	**1**	-1.55

Q-learning

Q-learning

- 갱신된 Q-table을 토대로 e-greedy 정책을 따라 에이전트가 [Start-Start-A-B-F-G-J-Goal]의 경로로 움직였다고 가정

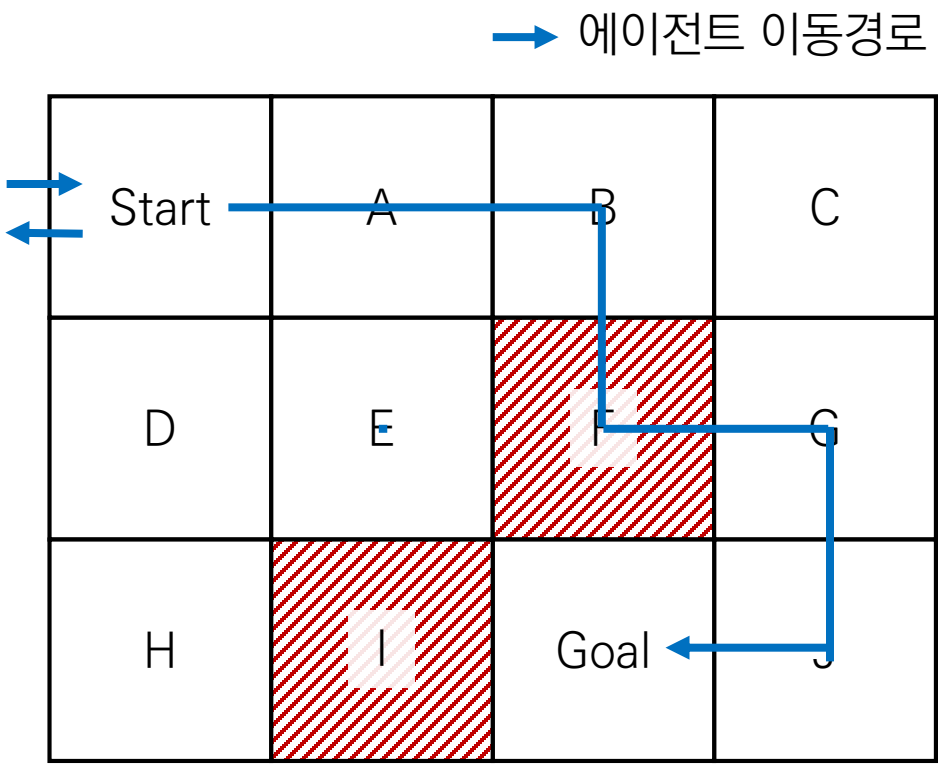


- 상태-행동 (Start,L)의 Q-Value 업데이트
  $Q(Start, L) = -1.55 + 1 * (-2 + 0.9 * -0.55 - (-1.55)) = -2.495$
- 상태-행동 (Start,R)의 Q-Value 업데이트
  $Q(Start, R) = -0.55 + 1 * (-1 + 0.9 * -0.55 - (-0.55)) = -1.495$
- 상태-행동 (A,R)의 Q-Value 업데이트
  $Q(A, R) = -0.55 + 1 * (-1 + 0.9 * -0.55 - (-0.55)) = -1.495$
- 상태-행동 (B,D)의 Q-Value 업데이트
  $Q(B, D) = -9.1 + 1 * (-10 + 0.9 * 1 - (-9.1)) = -9.1$
- 상태-행동 (F,R)의 Q-Value 업데이트
  $Q(F, R) = -0.55 + 1 * (-1 + 0.9 * -0.1 - 0.55 - (-0.55)) = -1.09$
- 상태-행동 (G,D)의 Q-Value 업데이트
  $Q(G, D) = -0.1 + 1 * (-1 + 0.9 * 1 - (-0.1)) = -0.1$
- 상태-행동 (J,L)의 Q-Value 업데이트
  $Q(J, L) = 1 + 1 * (1 + 0.9 * 0 - 1) = 1$

Q-learning

Q-learning

- 갱신된 Q-table을 토대로 e-greedy 정책을 따라 에이전트가 [Start-Start-A-B-F-G-J-Goal]의 경로로 움직였다고 가정



- 에이전트는 상태 B에서 가장 안좋은 행동인 “Down”을 선택
- 만약 Q-Learning이 다음 시점의 최대 행동 가치를 사용하지 않고 에이전트의 실제 행동을 사용하였다면 Q(A,R)는 -9.55의 값으로 갱신되었을 것

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- 이후의 Episode에서 에이전트는 Q(A,R)을 선택하지 않을 가능성이 높아져 실제 Goal 상태에 도달하기까지 무수히 많은 Episode가 필요할 것
- Q-Learning은 다음 시점의 최대 행동 가치를 사용하기 때문에 실제 상태 B에서 안좋은 행동을 택했을 지라도 Q(A,R)을 갱신하는 데에는 어떠한 영향도 끼치지 않음

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_{\acute{a}} Q(S_{t+1}, \acute{a}) - Q(S_t, A_t)]$$

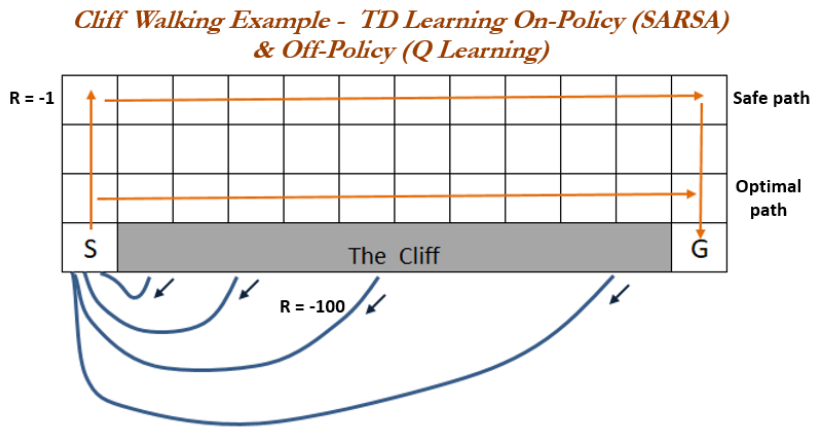
Q-learning

Off policy와 on policy

- Q-Learning과 같이 에이전트가 움직이는 정책(=행동 정책, Behavior Policy)과 가치 함수를 갱신하는 정책(=목표 정책, Target Policy)이 다른 경우를 Off-Policy 알고리즘이라고 함
- 반대로 두 정책이 같은 경우를 On-Policy 알고리즘이라고 하며 아래는 on-policy 알고리즘인 SARSA의 가치 함수 갱신의 식

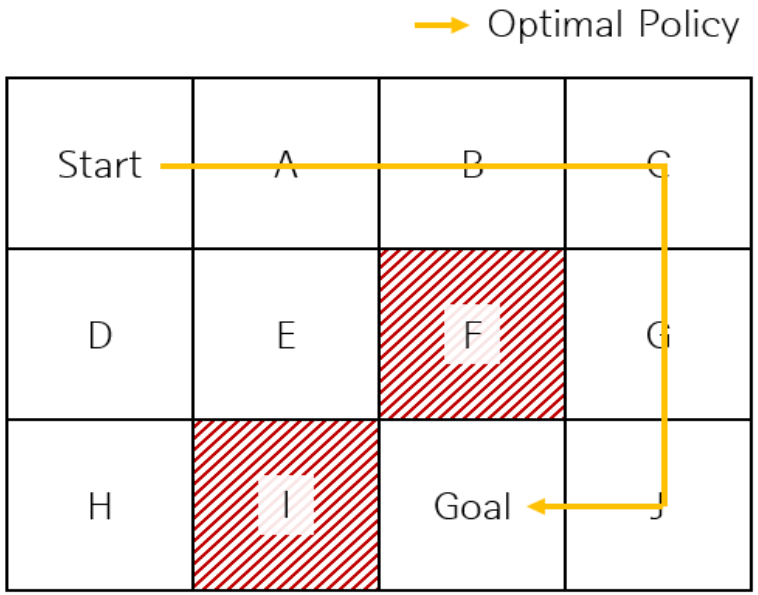
$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

- Q-Learning은 행동 정책으로 Epsilon Greedy Policy를, 목표 정책으로 Greedy Policy를 사용하는 Off-Policy 알고리즘에 해당함



Q-learning

- 이와 같은 방식으로 갱신을 계속하게 되면 최적의 정책을 찾을 수 있음



State	Up (U)	Down (D)	Left (L)	Right (R)
Start	-4.8	-3.8	-4.8	-3.5
A	-4.8	-3.8	-3.8	-2.78
B	-4.8	-9.1	-3.8	-1.981
C	-4.8	-1.09	-3.8	-4.8
D	-3.8	-3.8	-4.8	-3.8
E	-3.8	-9.1	-3.8	-9.1
F	-3.8	1	-3.8	-1.09
G	-1.981	-0.1	-9.1	-4.8
H	-3.8	-4.8	-4.8	-9.1
I	-3.8	-4.8	-3.8	1
Goal	0.5	0.5	0.5	0.5
J	-1.09	-4.8	1	-4.8

Q&A