

Deep Reinforcement Learning

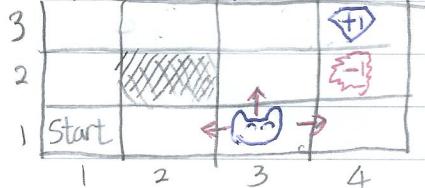
목차

1. Markov Decision Process
2. Bellman Equation
3. Dynamic Programming - Value ~~Fit~~ Iteration, Policy Iteration
4. Reinforcement Learning - Monte Carlo, Sarsa, Q-learning
5. Deep Reinforcement Learning - DQN, REINFORCE, A3C
6. Policy Gradient DRL - DDPG, TRPO, PPO
7. Distributional Reinforcement Learning - C51, QR-DQN, IQN
8. MAP Policy Optimisation (MPO) - Bayesian statistics, ELBO, EM

Markov Decision Process

마코프 의사결정 과정

Grid World example



4x3칸 (grid)

로봇 = agent = 강화학습으로 스스로 학습하는 컴퓨터 (학습의 주체)

Agent는 grid 안에서 벽 밖을 나갈 수 없음.

State : $S = \{(1,1), (1,2), (1,3), \dots, (4,2), (4,3)\}$ (2,2)는 존재X, 117개 state

Action : $A = \{\text{north}, \text{south}, \text{east}, \text{west}\}$ 4 actions

Reward : - big rewards at the end (+1, -1) 즉, 승점과 벌점

- small negative reward C for the other case 즉, 한번에 움직임마다 -0.1 [물결방지]

Agents는 계획한대로 항상 움직이진 않는다. (noisy movement). 랜덤성

State transition probability (상태 전이 확률)

- 각 상태에서 각 상태로 이동할 확률

상태를 움직일 때

- 예) Action north로 취함, agent는 north(80%), west(10%), east(10%) - 10% (south)

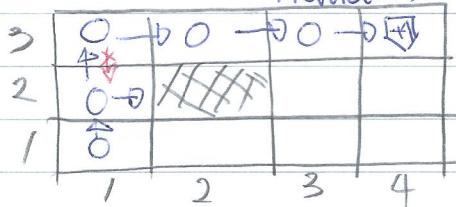
- Agent가 이동하는 방향에 Wall이 있으면 가만히 있다.

- 예) Episode (1,1) $\xrightarrow{\text{north}} (1,2) \xrightarrow{\text{east}} (1,2) \xrightarrow{\text{north}} (1,3) \xrightarrow{\text{south}} (2,3) \xrightarrow{\text{east}} (3,3) \xrightarrow{\text{east}} (4,3)$

80% 확률
North
/ reward (small) 2 reward
가만히

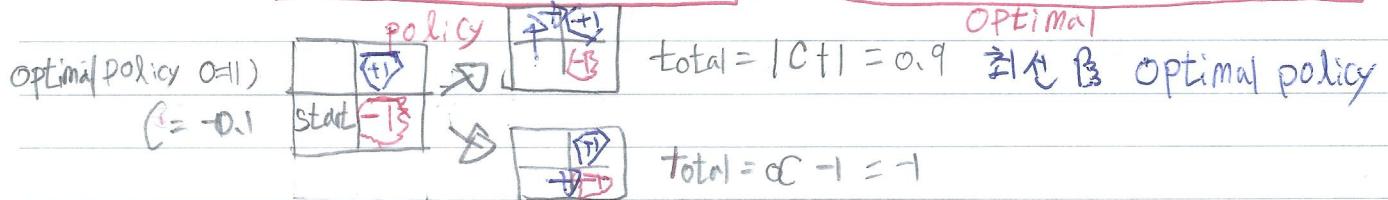
10% 확률
East
4 reward
South

5 reward
East
big reward



Total = 5 small + 1 big = 5C + 1

목표 : each state마다 정해준 action의 total sum of rewards를 maximize함.



강화학습의 핵심 목표 : optimal policy 찾기

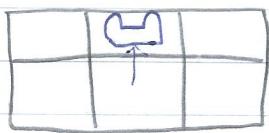
Actions in grid world

Deterministic grid world

결정



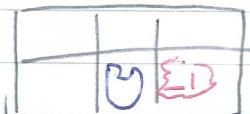
↓ 100%



model output of present state와 action은
fully determined한데 100% 차시한대로

policy 즉 one episode
정해진 뒤

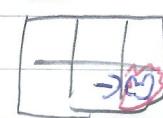
Stochastic grid world



10%

80%

10%



같은 state 와 action에서 다른 action을
랜덤하게 가져감. randomness
many episode

Markov property

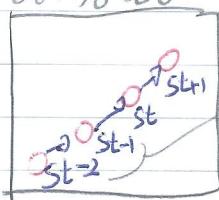
◎ Stochastic or random process : Collection of random Variables indexed by time set
'확률과정' - random variable 예) 확률 (Probability) $P(X=3)$, random variable X 가 3이 될 확률

- 랜덤 Variable의 색인은 time set
예) discrete (이산) time random process $S_0, S_1, \dots, S_{t-1}, S_t, S_{t+1}, \dots$
Time set이 1초 2초 discrete이며
같은 유연한 시점 | 현재 | 미래 ... 이 random variable
같이 있는 것

- Continuous time random process $\{S_t | t \geq 0\}$

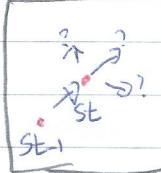
◎ Stochastic process is Markov property \Rightarrow Markov process (Markov chain)
- Markov property란?

관심이 적용되는 곳



현재 St-2, St-1 곳의 위치를 보고 St+1 유동 가능

관심이 적용되는 곳 (Mark)



St-1에서 St로 이동함

하지만 St+1 유동 여부 (Infix)



현재 St+1과 St+2가 어디로 생성될지
알기 어려움 (관심 방향을 모름)

미래가 유동 여부는 모름 \Rightarrow Brownian motion이라 함

즉 present state에서 future state으로 이동할 때

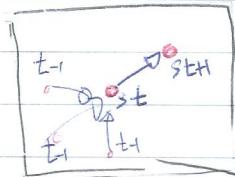
Past state가 전혀 영향을 미치지 않음 = Markov Property

Markov property 이어서

Markov property를 수학적으로 표현하면

$$P(S_{t+1} = S' | S_t = S) = P(S_{t+1} = S' | S_0 = s_0, S_1 = s_1, \dots, S_t = s)$$

T 첫시작 $s_0 \rightarrow s_1 \rightarrow \dots \rightarrow$ 현재 s 까지 옴
미래 state로 s' 이 될 확률

현재 state s 와 Next state s' 과 계산할 확률이 일치한다.즉 Past state (s_0, s_1, \dots, s_{t-1})을 고려하지 않아도 확률이 항상 일치.

past state가 어디서 오던 present와 next의 관계에 영향미치기 않는다.

정리: 위를 Markov property이며, Markov Property를 만족하는 Stochastic Process를 Markov process라고 함. Brownian motion의 대표적인 Markov property

- Markov property 장점 1. (과거 past state에 depend하지 않아) 과거 past를 기록하지 않음.
- (Memoryless) 2. Past state를 계산하지 않아 어려운 조건부 확률(여러음)을 많은 계산 없음.

추가 내용 1. $P(S_{t+1} = S' | S_t = S)$ 2. State of S_t 에서 S' 으로 전이 state transition.3. P 는 사용된 확률값 (probability)

4. 즉, 합쳐서 state transition probability라고 함

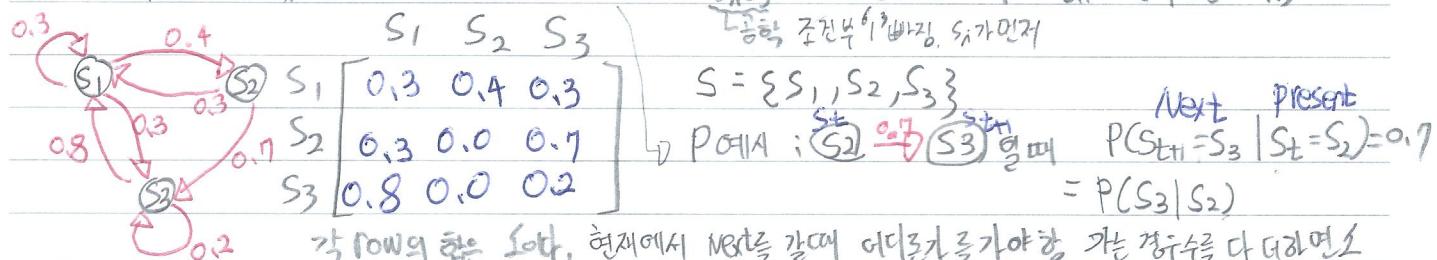
결론 1. Markov process는 tuple (S, P) 로 이루어짐

S: state set

P: state transition probability. Matrix로 표현하면 $[P_{ij}]$

Markov process 예)

$$P_{ij} = P_{S_i, S_j} = P(S_j | S_i) = P(S_{t+1} = S_j | S_t = S_i)$$

공학 조건부 확률, S_i 가 먼저

Markov process 2번째 예)

$$S_0, S_1, S_2, \dots, S_{t-2}, S_{t-1}, S_t, S_{t+1}, \dots$$

$$\begin{matrix} \parallel & \parallel & \parallel \\ S_1 & S_2 & S_1 \end{matrix} \quad \begin{matrix} \parallel & \parallel & \parallel \\ S_3 & S_3 & S_1 & S_2 \end{matrix} \quad \dots$$

T ↑
S3에서 S3로 이동 0.2 (위 그림과) $\textcolor{red}{S_2}$ 0.2

Photo: Markov property가 적용 안되면 S_t 을 계산하기 위해 $(S_0 \sim S_{t-1})$ 까지 depend한 계산해야 함(복잡)
Markov property를 적용하면 Matrix 계산 가능(단순)

Markov Decision Process (MDP)

MDP tuple (S, A, P, R, γ) Markov Property를 모든 state에서 만족해야 함

↑ 일을 때도, 일을 때도

 $S = \text{State}$, $A = \text{action}$, $P = (\text{State}) \text{ transition probability from } s \text{ to } s' \text{ given } a$ $R = \text{Reward function}$

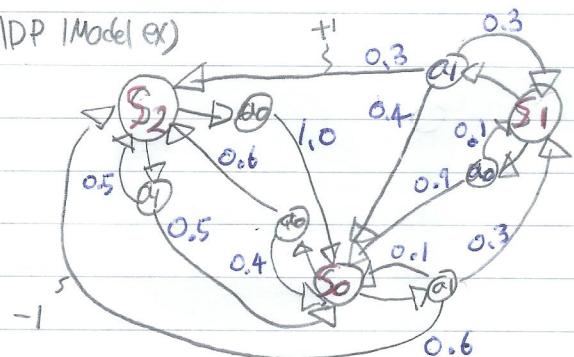
$P_{s,s'}^a = P(s'|s, a) = P(s_{t+1} = s' | s_t = s, A_t = a)$

 $R_{ss'}^a$ ↗
3 가지 다른 표현 R_s : 특정 state에서 Reward를 때
각 행식조건부 P(공학, s (현재 state)에서 s' 로 가는 확률) \rightarrow s 에서 s' 를 갈 때 a 액션을 취함.ac \rightarrow s' 를 갈 때 a 액션을 취함.
 s' (Next state)의 확률 $P(s')$ R_s^a : 현재 state에서 action을 취하자면 Reward $R_{ss'}^a$: 현재 state에서 action을 취하자면 Next state

도착해야 만족하는 Reward

 $\gamma \in [0, 1]$: discount factor \rightarrow 다음 강의 차세한 설명

MDP (Model ex)

3 States = S_0, S_1, S_2 2 actions : a_0, a_1

2 reward : +1, -1

$P(s_0, a_1) = 0.3, R_{s_1, s_2} = +1$

가정: MDP 모델을 하나 제작함. 제작자는 S, A, R 를 정해줘야 함. 어떤 모델은 P (전이 확률) 계산이 되는 문제가 있고, 어떤 모델은 P 가 계산이 안됨.

MDP에서 Environment model (주변 모델)을 transition probability로 찾

transition probability를 알 때

Model-based : known MDP (transition probability)

$P(s_{t+1} = s' | s_t = s, A_t = a)$

(현재, 미래) state과 action의 확률을

전부 알고 있음 = 방대한 데이터 (transition probability)

해결방법 \rightarrow Dynamic programming 사용

↳ 세부화시켜 효율적으로 푸는 방식

transition probability를 모를 때

Model-free : unknown MDP

transition probability를 알 수 없어.

Sample data를 기반으로 $P(a_t | s_t)$ 을 계산

↳ Reinforcement Learning

↳ 실제 환경에서 얻어진 데이터 or 사용자 입력 데이터

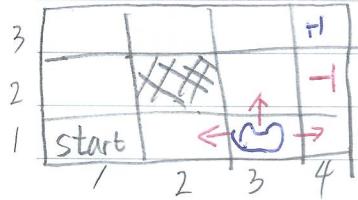
로 transition probability를 대체하여 Policy를 계산함.

The theory of MDP: does not required that S and A are finite (continuous도 관계X)

10/163

11/163

MDP in grid world



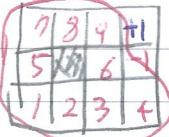
- state $S = \{(1,1), (1,2), (1,3), \dots, (4,2), (4,3)\}$ // states $\binom{2^2}{2,1,1}$

- Action set $A = \{\text{North}, \text{South}, \text{East}, \text{West}\}$ 4 actions

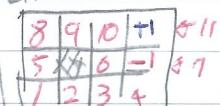
- State transition probability

$$P_{(3,1)}^{(\text{North})} = 0.8, P_{(3,2)}^{(\text{North})} = 0.1, P_{(3,3)}^{(\text{North})} = 0.1$$

S_t (현재 state) 수 9 가지, A_t 가능한 state 수 4 가지, S_{t+1} (Next state) 수 4 가지



취할 수 있는 액션 4 가지



$$\text{총 transition probability} = S_t \times A_t \times S_{t+1} = 9 \times 4 \times 11 = 396$$

Reward

Reward 예.)

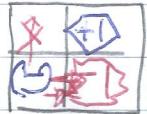
big reward

$$R_{(3,3),(4,3)}^{\text{East}(우)} = +1$$

$$R_{(3,2),(4,2)}^{\text{North}(위)} = +1$$

explain: 명령을 위로

행: 4 0.1 확률로 우회(East)로 가



$$R_{(4,1),(4,2)}^{\text{North}} = -1$$

small negative value



무한정 반복을 방지하기 위해, big reward가 아닌 케이스에선 $R_{ss}^a = c$

에서 a 를 취해서 s' 로 전하는 경우에 c 리워드를 줌 (c 는 음수, 예) > 0.1)

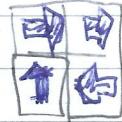
그리드 월드에서는 c 값에 따라 경로가 바뀜.

Optimal Policy in grid world ('c'값을 변화시킴)

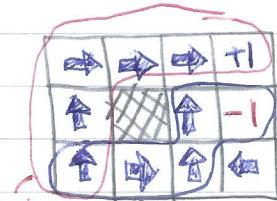
정책

- Policy π 는 각 state마다 어떤 action을 취해야 할지 정함 ex)

I policy

 $\pi: S \rightarrow A$

↑ state와 action 중



2 episode.

10% 확률로 다른 방향으로 가(동작)

$$3c - 1 = -2.2$$

$$1 \text{ episode } R_{ss}^a = -0.4$$

appropriate 정향

$$4 \times (-0.4) + 1$$

$$= -0.6 \text{ Total Reward}$$

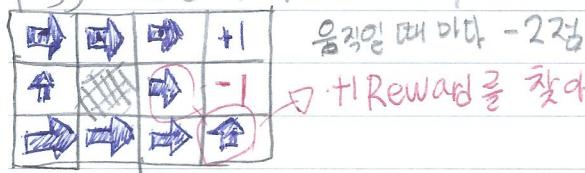
- Optimal Policy란(최적정책) 최대 expected total rewards를 가진다. π^* 로 표기

Optimal policy 계속

11/163

optimal policy 예시들

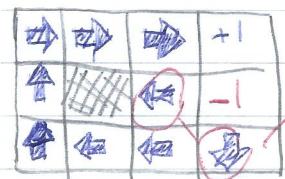
$$R_{ss'}^a = -2 \text{ 일 때 finish asap}$$



움직일 때마다 -2점

+1 Reward를 찾아가는 것보다, -1 Reward로 끝내는 것이 더 높은 점수

$$R_{ss'}^a = \begin{cases} -0.01 \\ (=c) \end{cases} \text{ slow but safe.}$$



움직여도 많이 손해보지 않음, 느려도 안전한 길을 선택

-1을 가지 않기 위해 동쪽(반대) 방향으로 강 10% 확률로 좌우 0% 이동

MDP의 목표는 optimal policy π^* : $S \times A$ 를 찾는 것이다.

DP, RL에서 원하는 목표 = optimal policy

Markov Decision process

Reward and Policy

12/16/3
13/16/3

Reward 정의: Reward R_t 현재 t step에서 agent가 한 행동이 얼마나 좋은지 안 좋은지를 평가하는 scalar feedback

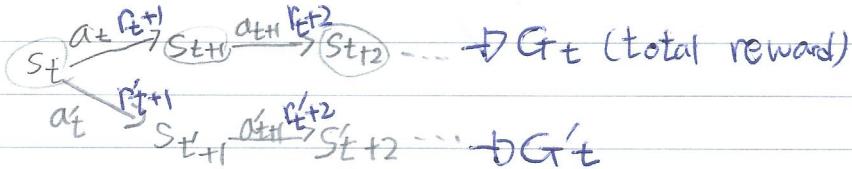
↳ Reward 값이 산수

agent의 일: Cumulative sum of rewards (total rewards)를 최대화하는 것.

RL is base on the Reward Hypothesis.

Reward Hypothesis: 모든 목표는 최대 total reward를 얻는 것

예1)



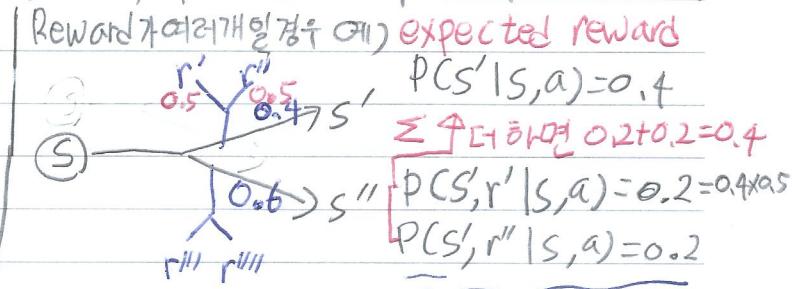
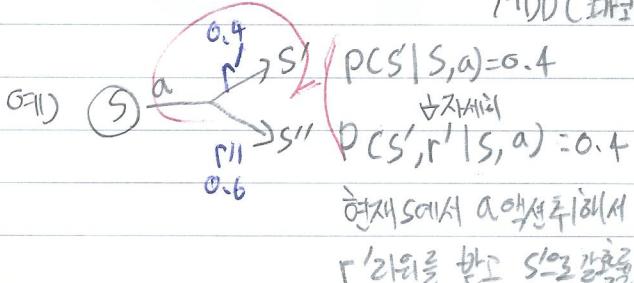
a_t $\xrightarrow{P_{t+1}^{a_t+1}} S_{t+1} \xrightarrow{a_{t+1} P_{t+2}^{a_t+2}} S_{t+2} \dots \xrightarrow{a_{t+k} P_{t+k}^{a_t+k}} S_{t+k} \rightarrow G_t$ (total reward)

Reward functions $R = R(s), R(s, a), R(s, a, s')$

예) 바둑: 게임이 끝난 뒤 Reward, Ping-pong: 주고 받을 때마다 (0.16%) Reward

여태 가정한 상황은 현재 State에서 하나의 Reward. 앞으로 여러개의 Reward를 갖는다

State transition probability: MPC(마코프 프로세스) $P(S'|S)$ 현재 state에서 next state로 넘어갈 확률
MDP(마코프 결정프로세스) $P(S'|S, a)$ 현재 state에서 action을 취했을 때 다음 state로



세부화된 Reward 경로를 전부 알면 Known dynamics

↳ Known dynamics $P(S', r | S, a)$

같음

State transition probability: $P_{S/S'}^a = P(S' | S, a) = P(S_{t+1} = S' | S_t = S, A_t = a) = \sum_{r \in R} P(S', r | S, a)$
 $R = \text{Reward set}, r = \text{определен reward set}; \sum_{r \in R}$

expected reward for state-action pair: $R_S^a = r(S, a) = E[R_{t+1} | S_t = S, A_t = a] = \sum_r \sum_{S' \in S} P(S' | S, a) r$

Reward의 세종류: 1. State가 정해지면 주어지는 리워드 R_S (마코프에서만 가능)

2. 특정한 state에서 액션 a를 취했을 때 얻는 리워드 R_S^a

3. 현재 state S에서 a 액션을 취해서 S'이 될 때 얻어지는 리워드 $R_{S/S'}^a$

1 MDP

Continue Reward and policy

◦ expected reward for state-action pair

$$R_s^a = r(s, a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] = \sum_{r \in R} r \sum_{s' \in S} P(s', r | s, a)$$

- State action pair를 고려 $R_s^a = r(s, a)$

↳ $s \xrightarrow{a} s'$ 현재 state에서 a 를 취했을 때 s' 도달. r 의 라워드를 얻음. 이때 리워드가 $r(s, a)$

↳ 만약 ⑤ $a \xrightarrow{r_1, r_2} s'$ 현재 state에서 a 액션을 취했을 때 reward₁=0.4, reward₂=0.6이다.

이때는 기대값을 계산해야 함

$\mathbb{E}[R_{t+1} | s, a]$: 현재 state에서 a 액션을 취했을 때 나올 수 있는 리워드 R_{t+1} 가 여러가지면 \mathbb{E} 기대값을 취합니다.

현재 state에서 액션 a 를 취했을 때 reward가 많다.

각각에 대한 확률을 알아야 함.

$\mathbb{E}[R_{t+1} | s, a]$ 리워드에 대한 기대값을 얻기 위해

$\sum_{r \in R} r \sum_{s' \in S} P(s', r | s, a)$
↳ 모든 리워드에 대해서 $\sum_{s' \in S} P(s', r | s, a)$
↳ 리워드와 리워드가 나올 확률을 계산함

$\sum_{r \in R} r \sum_{s' \in S} P(s', r | s, a)$

◦ $\mathbb{E}[R_{t+1} | S_t = s, A_t = a] = \sum_{r \in R} r \sum_{s' \in S} P(s', r | s, a)$ 의 식 유도

↳ 1. $S_t = s, A_t = a$ expectation(예측)을 sum으로 바꿔기

$$\sum_{r \in R} r \cdot P(r | s, a)$$

모든 리워드에 대해서, 각 리워드 와 주어진 state-action pair가 나올 확률

$$\sum_{r \in R} r \cdot P(r | s, a)$$

2. $P(r | s, a)$ 를 $\sum_{s' \in S} P(s', r | s, a)$ 로 바꿔주기 위해 state로 sum.

↳ reward r 과 함께 s (현재)에서 a (액션)을 취하면 s' (next)로 전이하는 확률이 $P(s', r | s, a)$
+ 모든 가능한 상태가 reward를 도려내기 위한 state sum.

$$\text{즉, } P(r | s, a) = \sum_{s' \in S} P(s', r | s, a)$$

$$\text{총정리 } \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \rightarrow \sum_{r \in R} r \cdot P(r | s, a) \rightarrow \sum_{r \in R} r \sum_{s' \in S} P(s', r | s, a)$$

주어진 state-action pair reward가 여러가지 일 경우 사용. (expected reward)

MDP

Reward

• Expected reward for state-action-next-state triple

↳ Expected reward for state-action pair 보다 세분화.

$$R_{ss'}^a = r(s, a, s') = \mathbb{E}[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s'] = \frac{\sum_{r \in R} r P(s', r | s, a)}{P(s' | s, a)}$$

계산목표: state-action-nextstate

이번식은 조건부론에 s, a, s' 까지 포함된 상태에서 Reward(R_{t+1})에 대한 expectation을 계산

$$\mathbb{E}[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s'] = \frac{\sum_{r \in R} r P(s', r | s, a)}{P(s' | s, a)} \text{ 를 유도.}$$

↳ $[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s']$ 을 reward sum으로 바꿔줌 (reward에 대한 expectation이 2)

$$\sum_{r \in R} r \cdot P(r | s, a, s')$$

각각의 r에 리워드가 나올 확률에 조건부에 s' 가 포함된 확률을 곱함↳ $P(r | s, a, s')$ 변형

$$\text{간단한 예 } P(r | s') = \frac{P(s', r)}{P(s')}$$

$$= \frac{\text{상태 } s' \text{ 을 방문한 } r \text{ 리워드를 가지는 확률}}{\text{상태 } s' \text{ 을 방문할 확률}}$$

여기 (s') 이 '내가 오늘 날'

(R)이 '사람들이 우산을 쓸 것'

$$\text{조건부 확률 } P(r | s) = \frac{\text{두 사건이 동시에 발생할 확률}}{s' \text{ 사건이 발생할 확률}}$$

 $P(s', r)$ 은 '오늘 비가 와 사람들이 우산을 쓸 확률' $P(s')$ 은 '오늘 비가 올 확률' $P(r | s')$ 은 '오늘 비가 와 때 사람들이 우산을 쓸 확률'

$$P(r | s') = \frac{P(s', r)}{P(s')} \text{ 에 동일한 조건을 추가할 수 있음}$$

$$\text{조건부 추가 } P(r | s, a, s') = \frac{P(s', r | s, a)}{P(s' | s, a)}$$

$$\text{즉, } [R_{t+1} | S_t = s, A_t = a, S_{t+1} = s'] \rightarrow \sum_{r \in R} r \cdot P(r | s, a, s') \rightarrow \sum_{r \in R} r \cdot \frac{P(s', r | s, a)}{P(s' | s, a)}$$

known dynamic $P(s', r | s, a)$ 경로적으로 1. State transition probability $P_{ss'}^a = P(s' | s, a) = \sum_{r \in R} P(s', r | s, a)$ 2. Expected reward for state-actionpair $R_s^a = r(s, a) = \sum_{r \in R} r \sum_{s' \in S} P(s', r | s, a)$ 3. Expected reward for state-action-next state triple - $R_{ss'}^a = r(s, a, s') = \frac{\sum_{r \in R} r P(s', r | s, a)}{P(s' | s, a)}$ 여러 리워드에
나올 때
expected

MDP

return

Return G_t : 현재 time step에서 이후에 얻어진 total discount reward

$$(S_t) \xrightarrow{a_t} S_{t+1} \xrightarrow{a_{t+1}} S_{t+2}$$

$R_{t+1} \rightarrow G_t$ t 사용 이유: 현재 state G_t 이후 모든 reward들의 sum

$$G_t = \cancel{R_{t+1}} + \cancel{\gamma R_{t+2}} + \cancel{\gamma^2 R_{t+3}} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Return G_t 는 리워드들을 sum을 해서 얻어짐 + γ (감가율)을 곱해줌. (r^k 는 후에 있는 리워드를 만날 확률)

Discount $\gamma \in [0, 1]$ 으로 1 사이의 값. 예) $\gamma = 0.9, R=1 \rightarrow \gamma^1 R = 0.9, \gamma^{10} R = 0.348$

γ 의 이유: 뒤로 갈수록 일어날 확률이 불확실(부정확한 리워드) 뒤로 갈수록 더 많은 감가.

더 적은 값을 곱해서 리워드들을 작게 만듬. 예가가 두면 바로 알겠죠, 예가가 가면 멀리 있는 것도 중요해

(MDP에서 γ 의 사용 이유 자세히)

1. 주학적으로 안전한데 만약 $G_t = \sum_{k=0}^{\infty} R_{t+k+1}$ (즉 γ 가 없을 때)

R_{t+k+1} 은 리워드가 계속 커져서 infinite return을 가질 수 있음

$G_t = \gamma^k R_{t+k+1}$ 이면 infinite return은 일어나지 않음

2. 미래에 대한 불확실성 - 예상 state 이후 reward는 불확실해서, 지수 γ time state가 흐를수록 reward의 가치가 점점 적어집니다. (exponentially fast)

3. in practice (실용적) - 가까이에 있는 reward가 멀리 있는 reward보다 중요

4. 만약 유한한 (finite)하게 끝나는 조건이면 discount를 사용하지 않아도 됨

Policy란 각 state마다 정해진 actions

Stochastic Policy (확률적) π : 각 state마다 action에 대한 확률 분포

$\pi(a|s) = P(A_t = a | S_t = s)$: 주어진 state에 대해 action a가 될 확률 (모든 state에)

Deterministic Policy : $\pi(s) = a$ Action을 1개뿐. Deterministic = 결정론적 예측한 대로.

Deterministic Policy 예) $a_1 \rightarrow s'_1 \rightarrow \dots G_1 \} \mathbb{E}[G_1] = 7 \leftarrow 0$

$\pi(s) = a_2$ $\begin{cases} a_2 \rightarrow s'_2 \rightarrow \dots G_2 \} \mathbb{E}[G_2] = 10 \leftarrow 1.0 \text{ 확률값} \\ a_3 \rightarrow s'_3 \rightarrow \dots G_3 \} \mathbb{E}[G_3] = 1 \leftarrow 0 \end{cases}$

Stochastic Policy 예) $a_1 \rightarrow s'_1 \rightarrow \dots G_1 \} \mathbb{E}[G_1] = 7 \leftarrow 0.15$

$\begin{cases} a_2 \rightarrow s'_2 \rightarrow \dots G_2 \} \mathbb{E}[G_2] = 10 \leftarrow 0.8 \\ a_3 \rightarrow s'_3 \rightarrow \dots G_3 \} \mathbb{E}[G_3] = 1 \leftarrow 0.05 \end{cases}$

확률이 분포되어 있음

$\pi(a_1|s) = 0.15, \pi(a_2|s) = 0.8, \pi(a_3|s) = 0.05$ 위 값은 표현 Stoch-Policy

Policy π 역할: total discounted reward (return) 값을 최대로 하는 것.

return을 최대화하기 위해 each state에 optimal action a (여러 액션)이 가장 좋은지 알려주는 guideline

↳ Past state 고려

증명 know MDP state probability 알고 있음. DP를 사용하여 해결, 모든 값을 전부 알고 있어 계산 가능.

계산을 도대로 expectation $\mathbb{E}[G]$ 가장 큰 값을 고를 수 있음

즉, optimal한 Policy $\pi^*(s)$ 를 계산한 선택 = Deterministic Policy

RL (Unknown MDP) transition probability를 모른다. Sample data만 가진 한정적 데이터만 있음. 정확한 $\mathbb{E}[G]$ 값 계산 X.

↳ 사용 가능 E-greedy policy (stochastic policy) (1- ϵ : choose optimal value, ϵ : choose randomly)

예) 2가지 Sample만 가지고 있음 $(a_1, s'_1), (a_2, s'_2)$

a_2 action을 취한 후, 얻어진 Samples를

$a_1 \rightarrow s'_1$ 정보임은 if 학습후 a_2 가 베스트 action : return값 평균이 가장 높음.
 $a_2 \rightarrow s'_2$ (greedy) 정보임은 a_2 쪽에만 1.0 확률을 주고 a_1, a_3, a_4, a_5 는 0을 주면 a_3 는 데이터가 없어 a_2 로 무시됨으로 끝됨.
 $a_3 \rightarrow s'_3$ 정보임은 a_2 쪽에만 1.0 확률을 주고, $\epsilon = 0.1$ 을 5개에 나눠줌.
 $a_4 \rightarrow s'_4$ (중립적) 정보임은 a_2 쪽에만 0.9 확률을 주고, a_3, a_5 가 베스트로 아니지만, 좋을 수 있다. 가능성 open.
 $a_5 \rightarrow s'_5$ 정보임은 a_2 쪽에만 0.9 확률을 주고, 나머지 $\epsilon = 0.1$ 을 4개로 나눠줌.
E-greedy는 best action = a_2 선택, E-greedy는 같은 랜덤하게 다른 선택

Summary of notation (자주 사용할 기호 설명)

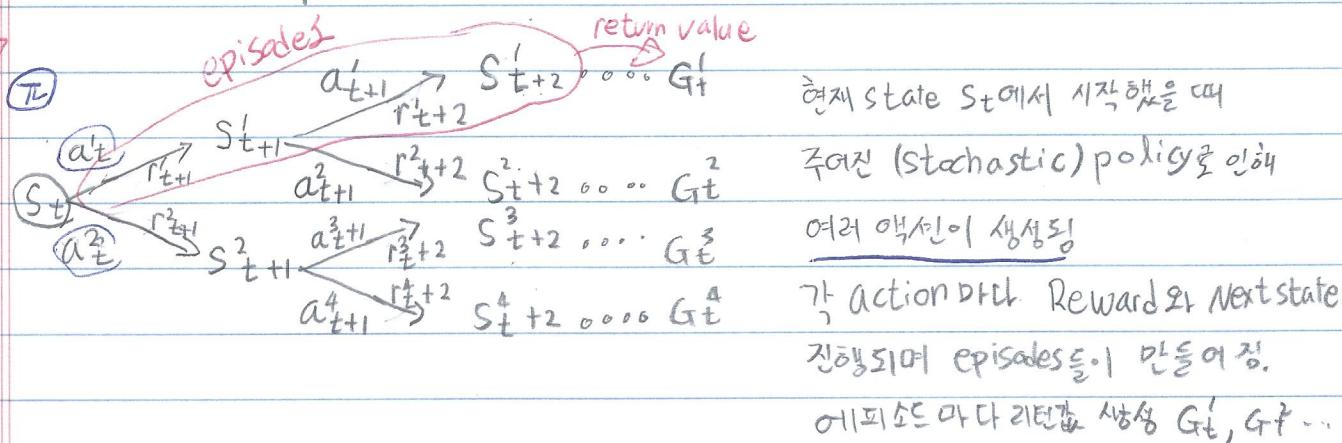
16/168

 $P(X=x)$ X 는 random variable, x 는 X 의 특징값 $P(X=x)$ 는 X 가 x 를 갖는 확률, 간략히 $P(x)$ 로도 씀 $E[X]$ random variable X 의 expectation(기대값)을 나타냄. 정의: $E[X] = \sum_x p(x)x$ \sum_x random variable X 가 갖는 모든 x 값에 대해서, $p(x)x$ x 값과 x 값의 확률을 곱한 것을 더해줌
기대값이라고 함. $p(x)$ 가 $\frac{1}{N}$ 일 때 (N 는 x 들의 총 개수) 평균값이라고 함. $\max_a f(a)$ set $\{a\}$ 위에서 $f(a)$ 의 최대값. $\arg \max_a f(a)$ $f(a)$ 라는 함수가 최대값을 가질 때 a 값. $\arg = \text{argument of the maximum}$ S_t, A_t, R_t time t 에서 state, action, reward. S_{t-1} 에서 A_t 취했을 때 stochastically(확률적으로) 일어지는 state, action, reward를 나타냄 G_t return: time t 이후에 일어지는 total discounted reward $P(s'|s, a)$ transition probability $\frac{p(s'|s, a)}{2}$ 의미. 현재 state s 에서 action a 를 취했을 때 next state이 s' 이 될 확률. $P(s', r|s, a)$ reward까지 고려한 transition probability. 현재 state s 에서 action a 를 취했을 때 next state이 s' 이며 reward가 r 인 π

policy: 모든 state에서 어떤 action을 취할지 정하는 정책. decision-making rule

 $\pi(a|s)$ stochastic policy: policy π 가 주어질 때, state s 에서 action a 를 취할 확률 $\pi(s)$ deterministic policy: policy π 가 주어졌을 때, state s 에서 취할 액션을 나타냄

추후 배울 내용

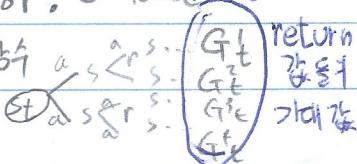


좋은 Policy란

- 현재 state에서 만들어진 episode들이 있다. 각 episode에서 일어지는 return 값이 크다.
- State가 좋은 위치에 있다면 일어지는 return 값도 큼.

알고 싶은 것

- Policy가 주어졌을 때 state t 가 좋은 위치인지 평가하고 싶음.
- state t 에서 policy π 에서 나온 모든 episode의 return 값의 기대값 (expectation)이 좋은지.

 $V_\pi(s)$ state-value function: 주어진 policy π 안에서 state의 가치평가. Expected return즉, 주어진 policy π 안에서 state s 에 대한 가치 V 를 나타내는 함수 $a, s \xrightarrow{\pi} r, s' \xrightarrow{\pi} G_t^1, G_t^2, G_t^3, G_t^4$ 

Continue Summary of notation

 $V_{\pi}(s)$

두 가지 명칭

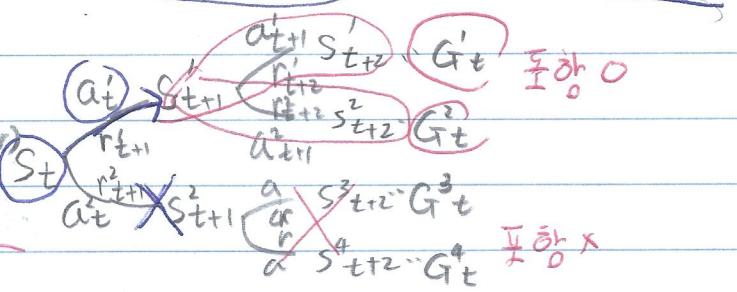
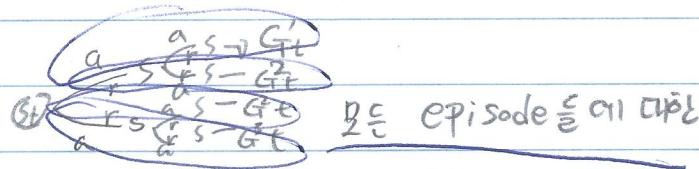
 $V_{*}(s)$

는 ^{best} optimal case를 의미。 optimal state-value function: policy가 optimal, optimal policy π_{} 일 때 states의 value (expected return)을 의미

$q_{\pi}(s, a)$ (state) action-value function: policy π 가 주어져 있을 때, 특정한 state s 에서 특정한 actiona를 취했을 때, 얻어지는 value.

state s 가 $\Rightarrow V_{\pi}(s)$ state value function은
증오한지 가치로는 return의 expectation 값.

State s $\Rightarrow q_{\pi}(s, a)$ action-value function은
현재 state과 액션은 고정 (state-action pair)
이후에 얻어지는 episode return 값들의
expectation을 의미



즉, 특정한 state s 에서 action a 를 취했을 때 앞으로 얻어질 return 값들의 기대값
평균적으로 이정도의 return을 얻을 것이다. state action pair 가치 나타내는 함수

 $q_{*}(s, a)$ $q_{\pi}(s, a)$ 차이점은 policy π 대신 optimal policy π_{*} 를 받음.optimal policy π_{*} 에 대해 action a 를 취했을 때 얻는 value (expected return)DP는 $V_{\pi}(s)$, $V_{*}(s)$ RL은 $q_{\pi}(s, a)$, $q_{*}(s, a)$