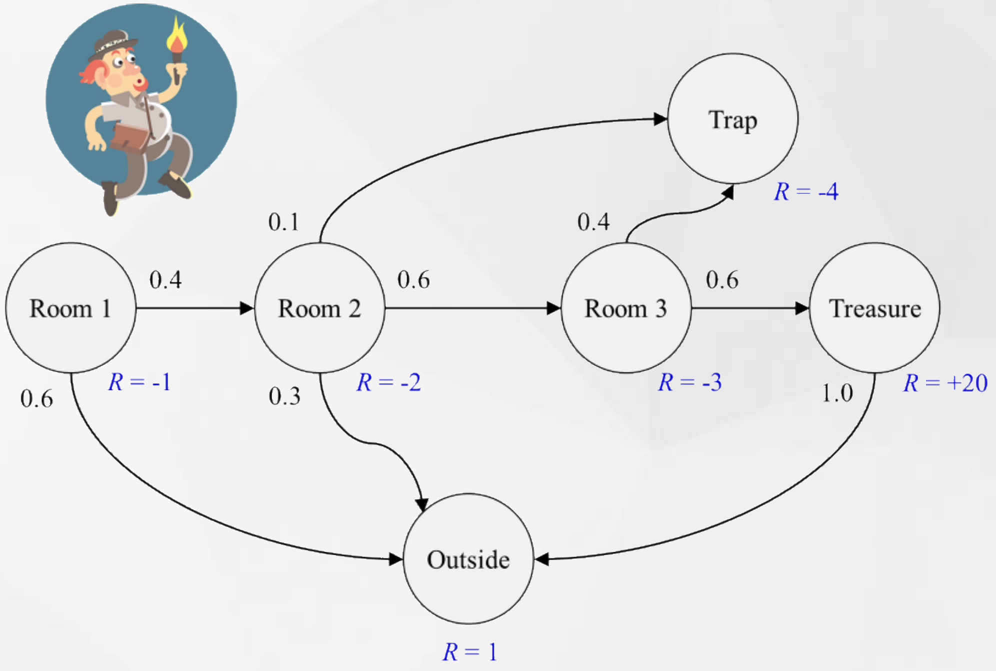
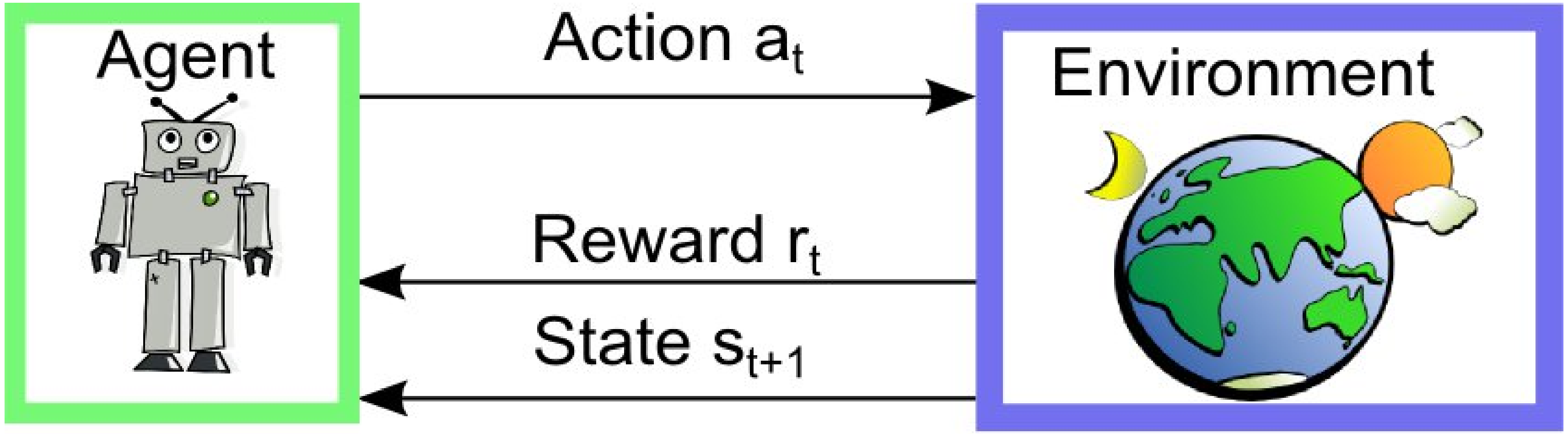
Chapter 06. 스스로 전략을 짜는 강화학습 (Reinforcement Learning)

가치함수, 벨만방정식, MDP



Reinforcement Learning



Reinforcement Learning Setup

•에이전트(Agent) : 상태를 관찰, 행동을 선택, 목표지향
•환경(Environment) : 에이전트를 제외한 나머지 (물리적으로 정의하기 힘듦)
•상태(State) : 현재 상황을 나타내는 정보
•행동(Action) : 현재 상황에서 에이전트가 하는 것
•보상(Reward) : 행동의 좋고 나쁨을 알려주는 정보

Value Funtion

•**보상 (R, Reward): 보상**이란 에이전트의 행동에 대한 성공이나 실패를 측정하는 피드백. 에이전트의 행동에 의해 평가된 보상은 즉시 주어질 수도, 지연될 수 도 있음.

•**할인율(γ , Discount factor): 할인율**은 보통 0과 1 사이의 값으로 즉각적으로 주어지는 보상보다 상대적으로 가치가 낮은 미래의 보상을 만들기 위해 고안. γ가 0.8이고 3단계를 거쳐 10점의 보상을 받는다면 보상의 현재가치는 $0.8^3 \times 10$이다.

•**Return (G, total discount reward) :** 전체 reward를 시간에 따른 감가상각을 포함하여 합산 한 것

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

Value Funtion

•**가치 (Value Function):** expected return starting from state s
현재 상태 s 에서부터 시작할 때 기대되는 return 값을 말합니다.

$$V(s)=\mathbb{E}[R(s0)+\gamma R(s1)+\gamma 2R(s2)+\dots|s0=s].$$

cf. 이웅원님 글

Value function은 return(실재 경험을 통해서 받은 reward의 discounted amount)의 expectation이기 때문에 마치 주사위를 던져 보듯이 던져 보면서 expectation 값을 구할 수 있습니다.
계속 그 state로부터 시작되거나 그 state를 지나가는 episode를 try해보면서 얻어진 reward들에 대한 data들로 그 value function에 점점 다가갈 수 있는데 사실 주사위도 무한번 던져야 1/3이라는 true expectation값을 가지듯이 value function 또한 무한 히 try를 해봐야 true value function을 찾을 수 있을 것 입니다. 그렇다면 어떻게 적 당한 선을 찾아서 "이 정도는 true 값이라고 하자"라고 결정을 내릴까요? 이 또한 생각해봐야 할 점인 것 같습니다.

Bellman Equation

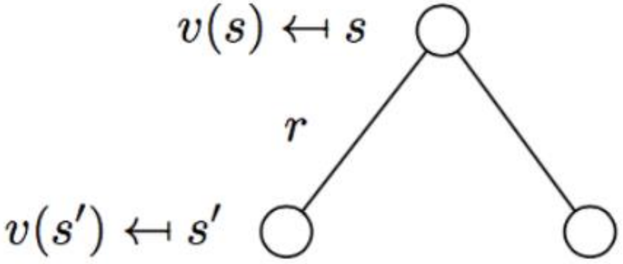
The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

Solving Bellman Equation

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \boxed{\mathcal{P}_{ss'}} v(s')$$

State Transition Matrix (s->s')

$$\begin{bmatrix} v(s_1) \\ \vdots \\ v(s_N) \end{bmatrix} = \begin{bmatrix} r(s_1) \\ \vdots \\ r(s_N) \end{bmatrix} + \gamma \begin{bmatrix} p(s_1|s_1) & \cdots & p(s_N|s_1) \\ \vdots & \ddots & \vdots \\ p(s_1|s_N) & \cdots & p(s_N|s_N) \end{bmatrix} \begin{bmatrix} v(s_1) \\ \vdots \\ v(s_N) \end{bmatrix} \tag{6}$$

$$\begin{aligned} v &= \mathcal{R} + \gamma \mathcal{P} v \\ (I - \gamma \mathcal{P})\, v &= \mathcal{R} \\ v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R} \end{aligned}$$

https://untitledtblog.tistory.com/139

Markov Reward Process

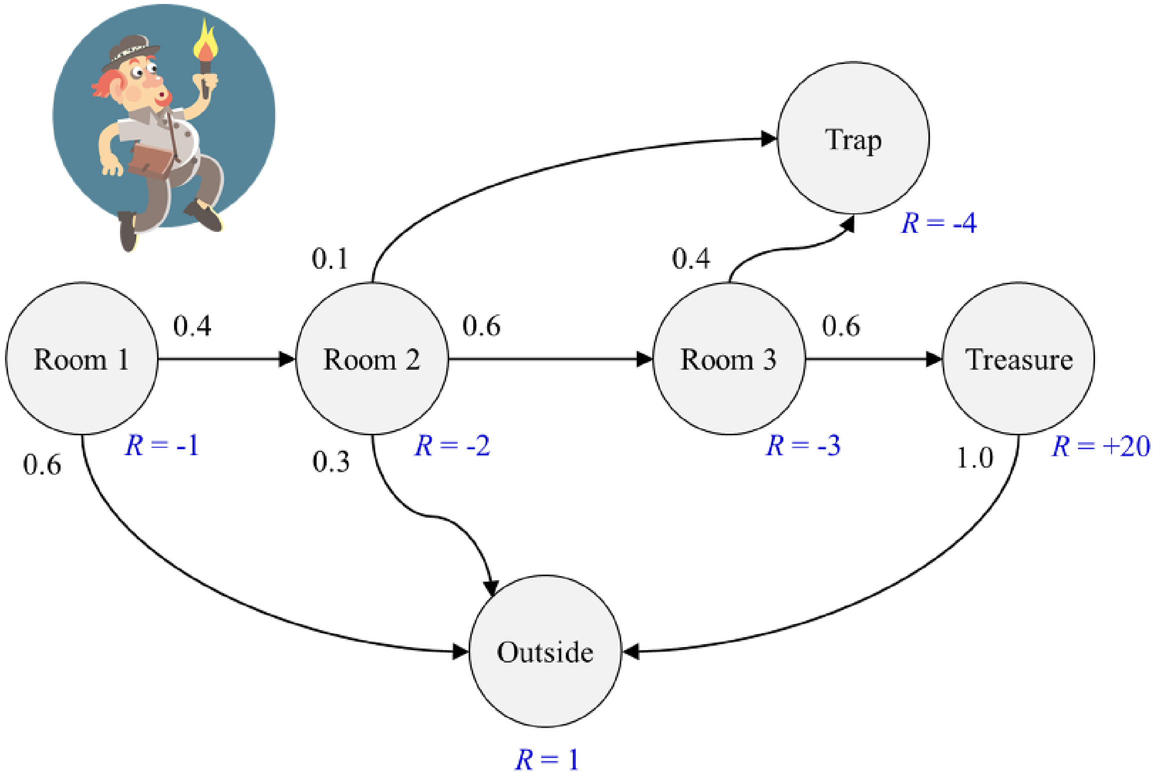
Markov Reward Process :

Markov reward process는 Markov process의 각 state에 reward를 추가하여 확장한 것이다. Markov reward process는 아래와 같이 정의된 <S,P,R,γ>라는 4-tuple로 표현

- S: state의 집합을 의미한다. State는 바둑에서 바둑판에 돌이 어떻게 놓여져 있는가를, 미로를 탈출하는 문제에서는 현재의 위치를 나타낸다.
- P: 각 요소가 p(s′|s)=Pr(St+1=s′|St=s)인 집합이다. p(s′|s)는 현재 상태 s에서 s′으로 이동할 확률을 의미하며, transition probability라고 한다.
- R: 각 요소가 r(s)=E [Rt+1|St=s]인 집합이다. r(s)는 state s에서 얻는 reward를 의미한다.
- γ: 즉각적으로 얻는 reward와 미래의 얻을 수 있는 reward 간의 중요도를 조절하는 변수이다. 주로 [0, 1] 사이의 값을 가지며, discount factor라고 한다.

Markov Reward Process

https://untitledtblog.tistory.com/139



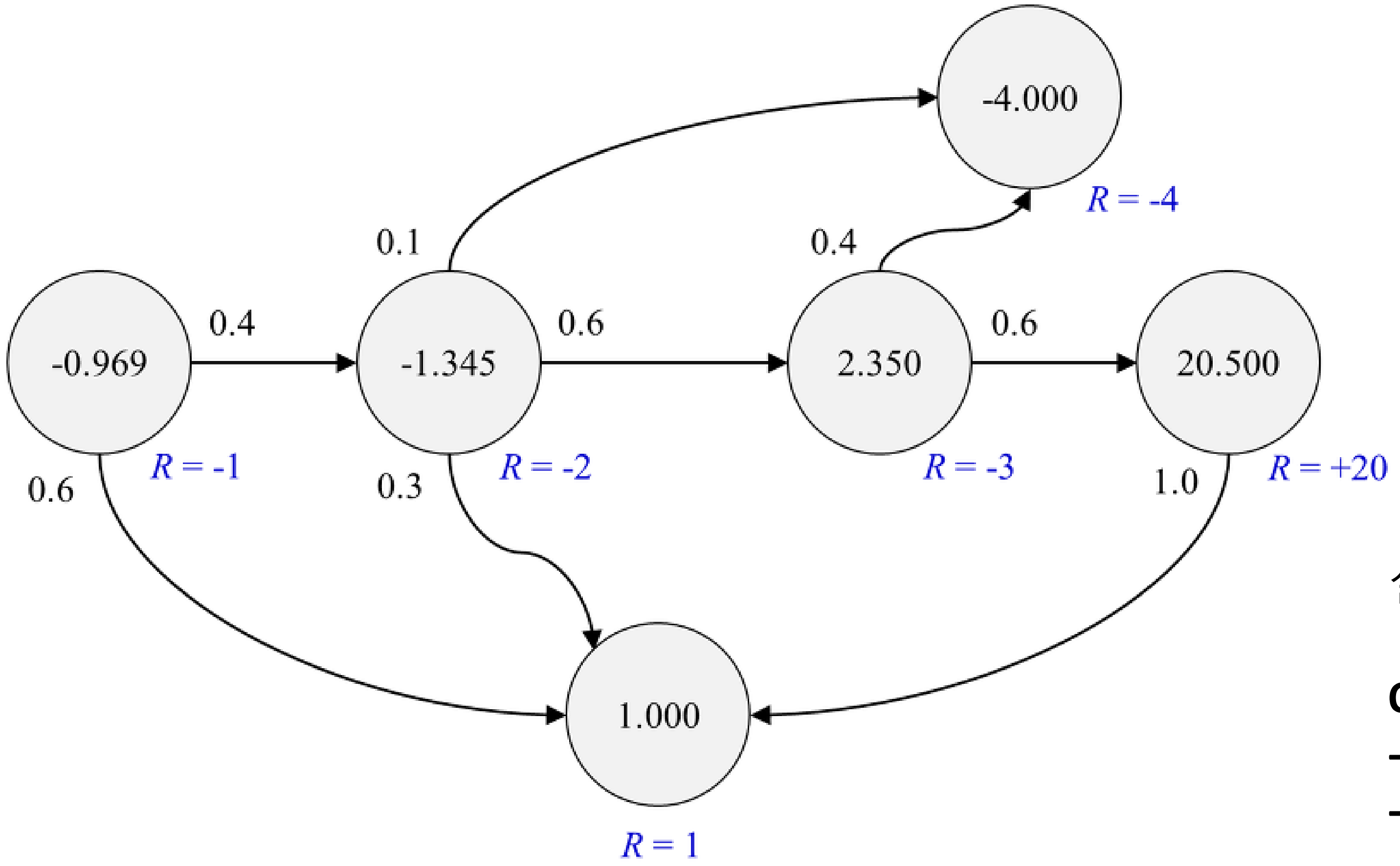
$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

γ=0.5, S1= Room 1이고, Room 2, Outside 의 순서로 탐험하였을 때

$$G_1 = -1 + (0.5) \times (-2) + (0.5)^2 \times 1 = -1.75$$

Markov Reward Process

https://untitledtblog.tistory.com/139



실행 시간 복잡도 : **O (n^3)**

Other iterative solving method
-Dynamic Programming
-Temporal Difference Learning
-Monte-Carlo Method

Markov Decision Process

MDP: MDP는 Markov reward process에 action이라는 요소가 추가된 모델로써, <S,A,P,R,γ>라는 tuple로 정의

Policy 정책 (π) : 정책은 각 상태 (s ∈ S)에 대해 Actions (a∈ A)에 대한 **확률 분포** 를 정의하는 함수

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

State Transition (P) : MDP가 주어진 π를 따를 때, s에서 s′으로 이동할 확률

$$p_\pi(s'|s) = \sum_{a \in A} \pi(a|s) p(s'|s,a) \qquad (9)$$

Reward(P) : s에서 얻을 수 있는 reward

$$r_\pi(s) = \sum_{a \in A} \pi(a|s) r(s,a) \qquad (10)$$

fast campus

Markov Decision Process

State-value function with policy

MDP에서 state-value function v(s)는 Markov reward process의 state-value function과 마찬가지로 state s에서 시작했을 때 얻을 수 있는 return의 기댓값을 의미한다. 그러나 MDP는 주어진 policy π를 따라 action을 결정하고, state를 이동하기 때문에 MDP에서의 state-value function은 다음과 같이 정의된다

$$\begin{aligned} v_\pi(s) &= \mathrm{E}_\pi[G_t | S_t = s] = \mathrm{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\ &= \sum_{a \in A} \pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_\pi(s') \right) \end{aligned} \tag{11}$$

Markov Decision Process

Action-value function

Action-value function qπ(s,a) 는 state s에서 시작하여 a라는 action을 취했을 때 얻을 수 있는 return의 기댓값을 의미한다. Action-value function qπ(s,a) 는 아래 식과 같이 정의된다.

$$q_\pi(s,a) = \mathrm{E}_\pi[G_t | S_t = s,\ A_t = a] = \mathrm{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1},\ A_{t+1}) | S_t = s]$$
$$= r(s,a) + \gamma \sum_{s' \in S} p(s' | s,a) \sum_{a' \in A} \pi(a' | s') q_\pi(a' | s') \qquad (12)$$

State-value function은 어떠한 state가 더 많은 reward를 얻을 수 있는지를 알려준다면, action-value function은 어떠한 state에서 어떠한 action을 취해야 더 많은 reward를 얻을 수 있는지 알려줌

Markov Decision Process

Optimal state-value function and optimal action-value function

Definition

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

The *optimal action-value function* $q_*(s,a)$ is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is "solved" when we know the optimal value fn.

fast campus

Markov Decision Process

optimal policy

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- *There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$*
- *All optimal policies achieve the optimal value function, $v_{\pi_*}(s) = v_*(s)$*
- *All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s,a) = q_*(s,a)$*

- *Thank you*