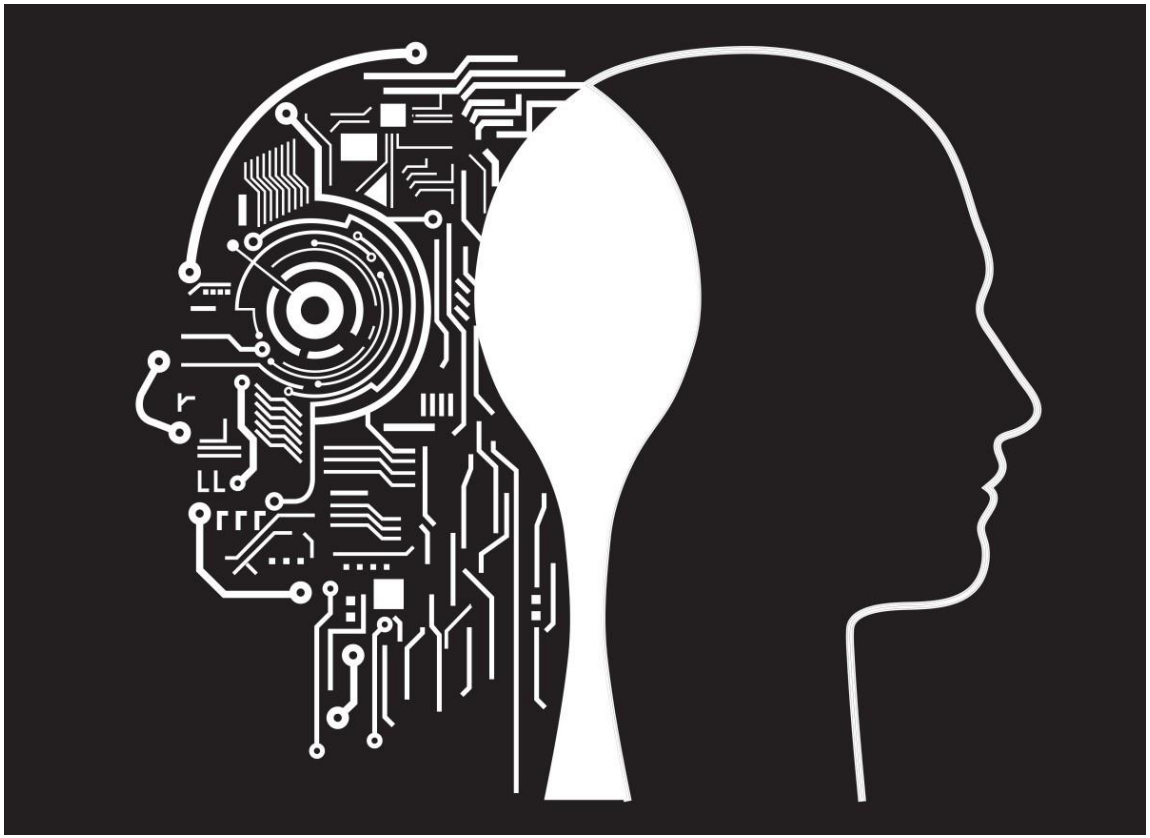
Markov Decision Processes

석사과정 이기창

Contents

강화학습 개요

어떤 기계를 만들 것인가



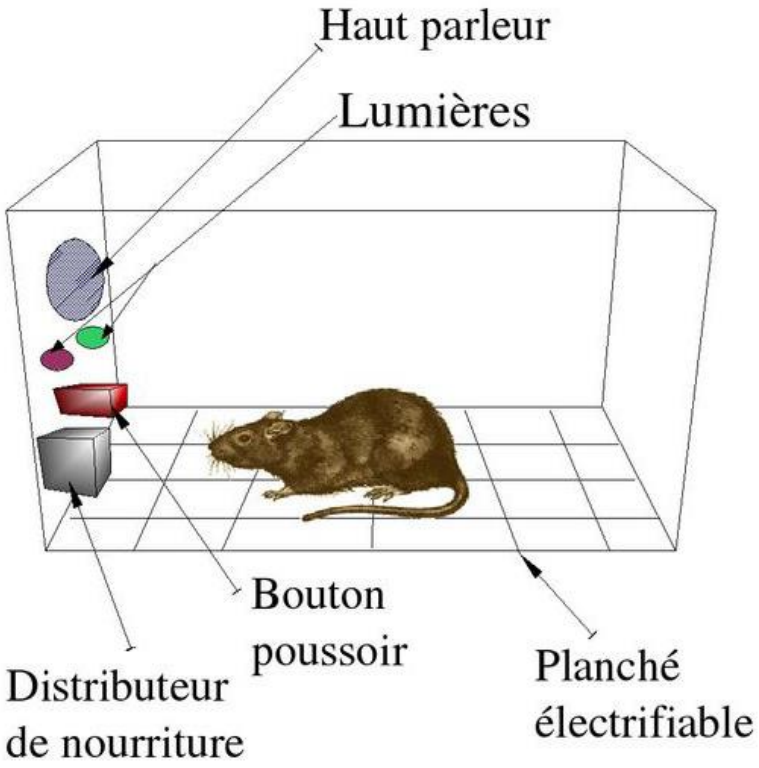
위키피디아

강화학습 개요

강화학습?

강화학습 개요

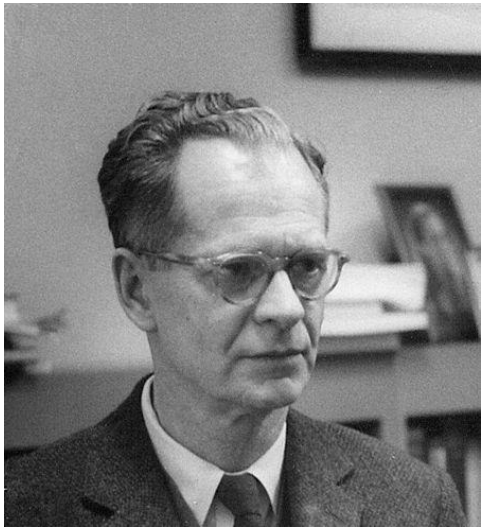
스키너 상자



위키피디아

1. 배고픈 흰 쥐를 스키너 상자에 넣는다.
2. 흰 쥐는 스키너 상자 안에서 돌아다니다 우연히 지렛대를 누르게 된다.
3. 지렛대를 누르자 먹이가 나온다.
4. 지렛대와 먹이 간의 상관관계를 알지 못하는 쥐는 다시 상자 안을 돌아다닌다.
5. 다시 우연히 지렛대를 누른 흰 쥐는 또 먹이가 나오는 것을 보고 지렛대를 누르는 행동을 자주 하게 된다.
6. 이러한 과정이 반복되면서 흰 쥐는 지렛대를 누르면 먹이가 나온다는 사실을 학습하게 된다.

강화학습 개요



위키피디아

Burrhus Frederic Skinner (1904~1990)

"유기체가 어떤 행동을 한 뒤에 유기체가 원하는 것을 제공하는 행위를 강화(reinforcement)라고 한다. 어떤 행동을 한 결과가 유기체 스스로에게 유리하면 유기체는 그 행동을 더 자주 한다."

강화학습 개요

에이전트 agent

의사결정자 decision maker

환경 environment

에이전트 바깥에 있는 모든 것

행동 action

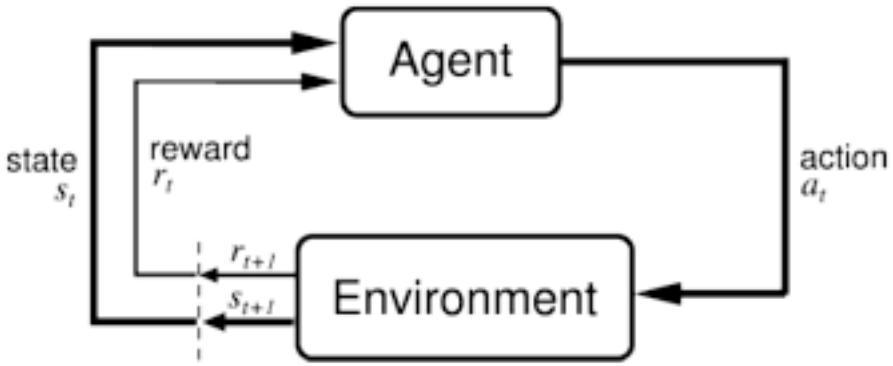
에이전트의 의사결정

상태 state

에이전트가 환경을 관측한 결과
(마코프 프로세스)

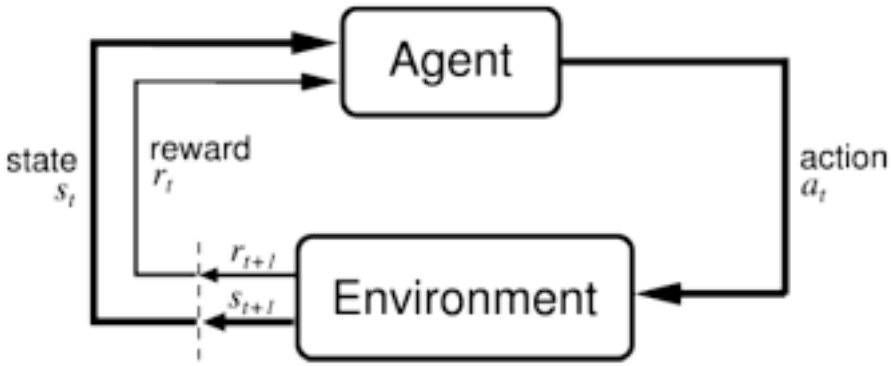
강화학습 개요

강화학습 매커니즘



강화학습 개요

강화학습 매커니즘



관측observation, 행동action, 보상reward

보상reward을 최대화!

강화학습 개요

동전던지기

에이전트?

환경?

행동?

상태?

강화학습 개요

동전던지기

에이전트 : 나

환경 : 게임룰 전체

행동 : 동전던지기

상태 : 앞면, 뒷면

마코프 프로세스

마코프 프로세스 Markov process

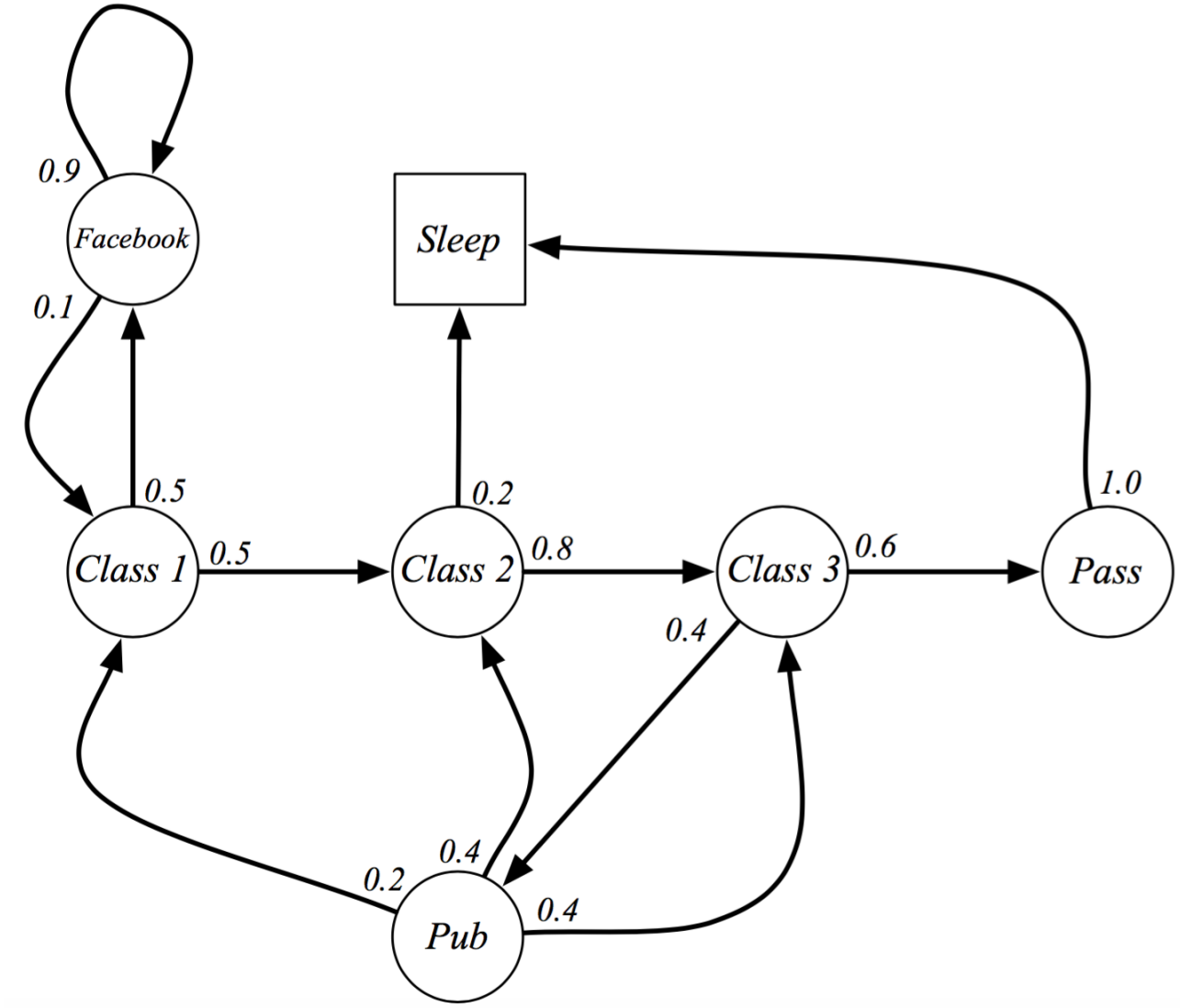
$$P[S_{t+1}|S_t] = \mathrm{P}[\mathrm{S_{t+1}}|\mathrm{S_1}, \ldots, \mathrm{S_t}]$$

일련의 시간적 사건이 있을 때 현재 단계의 상태에서 예상되는
다음 단계의 상태는 과거의 사건과는 무관하다.

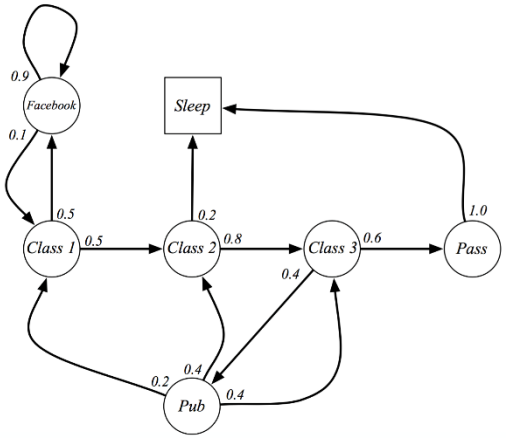
Agent가 지금 당장 취할 행동에 영향을 주는 요인은 과거의 사건이 아니라 미래의 상태와 그에 따른 보상!

Memoryless process!

마코프 프로세스


0.9
Facebook
0.1
Sleep
0.5
0.2
1.0
Class 1
0.5
Class 2
0.8
Class 3
0.6
Pass
0.4
0.4
0.2
0.4
Pub

마코프 프로세스



현재 구성요소

〈S, P〉 S=상태state
P=상태전이확률state transition probability

상태전이행렬 State Transition Matrix

$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

마코프 보상 프로세스

보상 reward

Agent가 어떤 상태에서 다음 단계로 이동하는 행동을 취할 때
환경으로부터 피드백 받는 스칼라 실수값

마코프 보상 프로세스

감쇄계수 discount factor

미래에 있을 보상값을 현재가치로 환산(할인)하기 위한 값. [0, 1]
감쇄계수가 1이면 모든 미래가치를 현재와 같은 비중으로 고려한다.
현실을 반영하는 모델링이지만 수학적 장점(수렴)도 있다.

마코프 보상 프로세스

t시점에서의 return

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

기준시점(t) 이후 미래 보상을 현재가치로 환산해 모두 더한 값.
마코프 프로세스는 과거는 무시하기 때문에 R_{t+1}부터 고려한다.

마코프 보상 프로세스

상태가치함수 state value function

$$v(s) = E[G_t|S_t = s]$$

$$= \mathrm{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots |\mathrm{S_t} = \mathrm{s}]$$

$$= \mathrm{E}[\mathrm{R_{t+1}} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots)|S_t = s]$$

$$= \mathrm{E}[\mathrm{R_{t+1}} + \gamma \mathrm{G_{t+1}}|\mathrm{S_t} = \mathrm{s}]$$

마코프 보상 프로세스

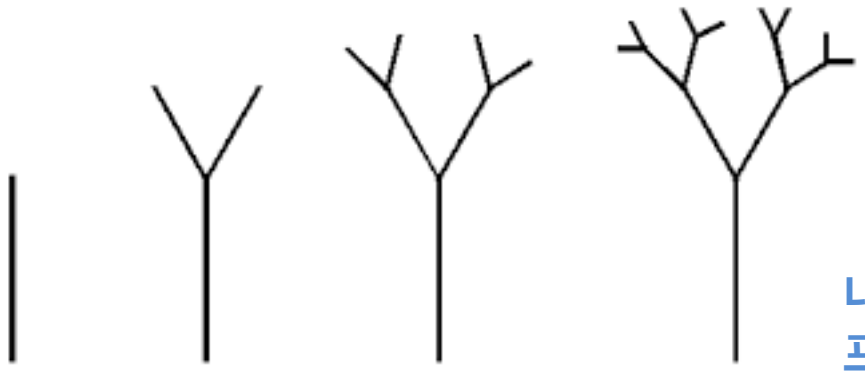
상태가치함수 state value function

$$v(s) = E[G_t | S_t = s]$$

$$= \mathrm{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots | \mathrm{S_t} = \mathrm{s}]$$

$$= \mathrm{E}[\mathrm{R_{t+1}} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) | S_t = s]$$

$$= \mathrm{E}[\mathrm{R_{t+1}} + \gamma \mathrm{G_{t+1}} | \mathrm{S_t} = \mathrm{s}]$$

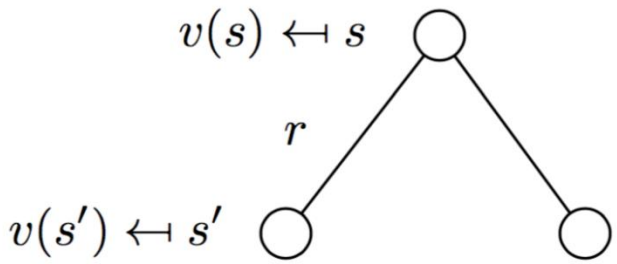


내 안에 너 있다!
프랙탈 구조?

마코프 보상 프로세스

상태가치함수 state value function

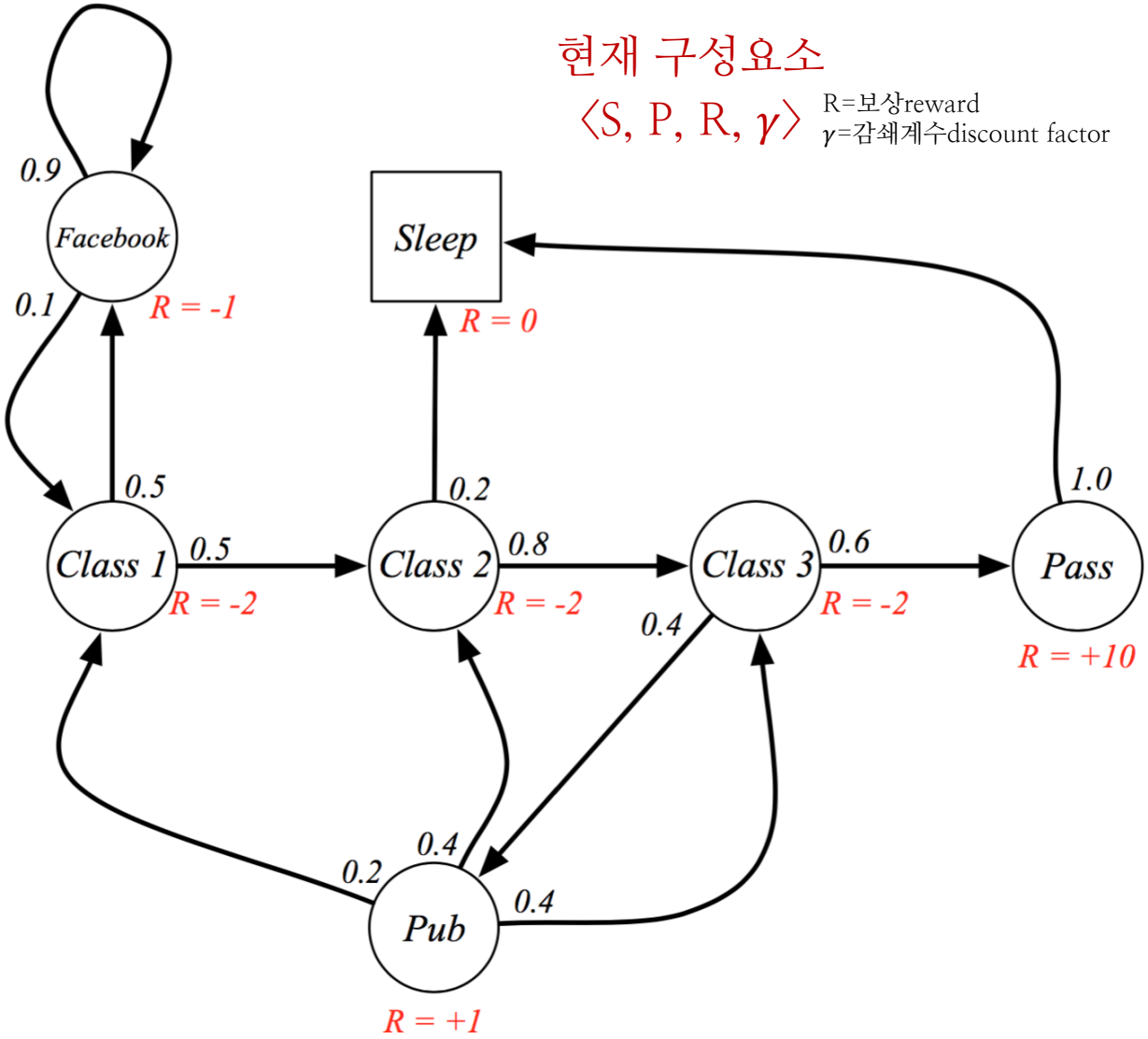
$$v(s) = \mathrm{R_s} + \gamma \sum_{s' \in S} P_{SS'} v(s')$$



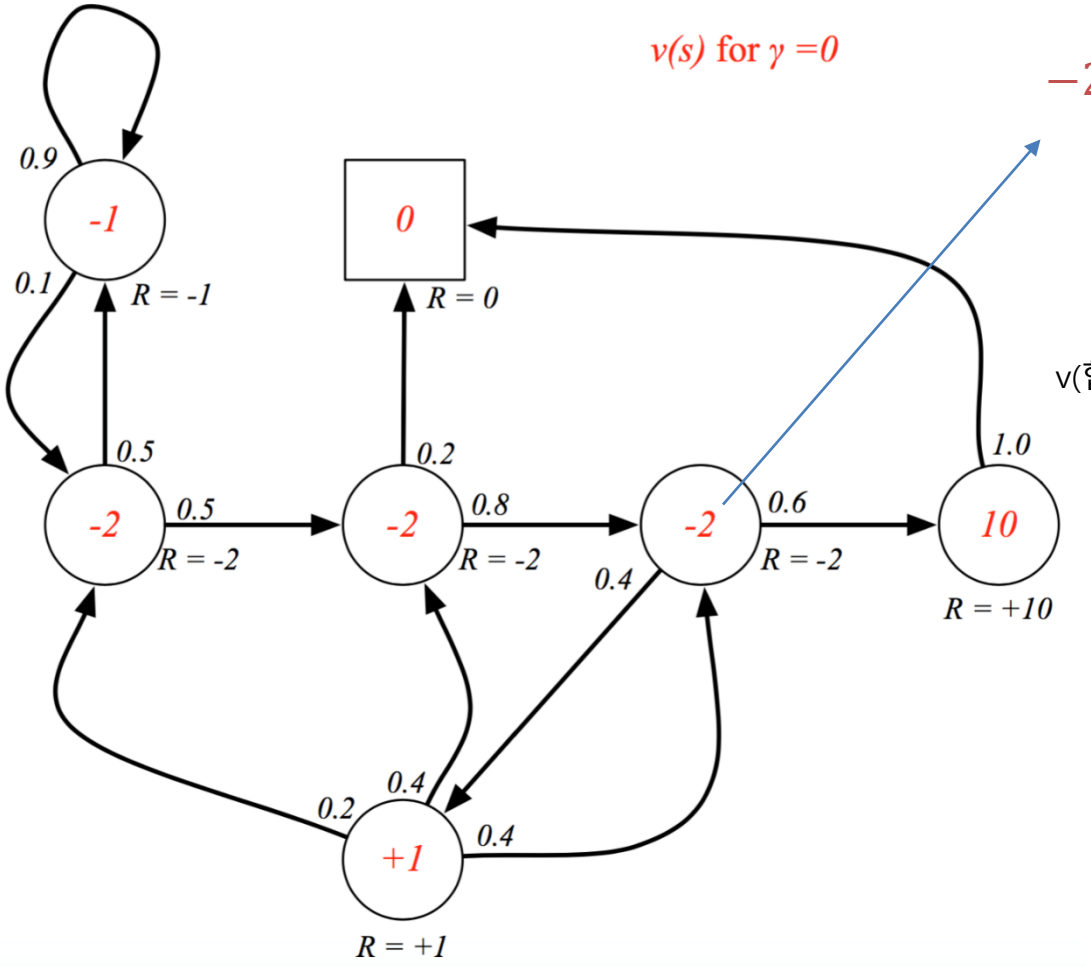
이처럼 상태가치는 1)즉시보상과 2) t+1이후의 모든 미래가치를 현재 가치로 환산한 값으로 분해할 수 있다.
다시 말해 즉시보상과 다음 단계 상태가치를 더하면 현재 상태가치가 된다.

위의 식이 바로 벨만Bellman 방정식이며, MDP문제는 곧 이 방정식을 푸는 것을 뜻한다.

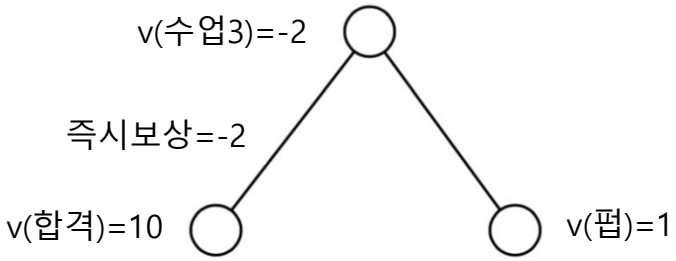
마코프 보상 프로세스


현재 구성요소
〈S, P, R, γ〉
R=보상reward
γ=감쇄계수discount factor
0.9
Facebook
0.1
R = -1
Sleep
R = 0
0.5
0.2
1.0
Class 1
0.5
R = -2
Class 2
0.8
R = -2
Class 3
0.6
R = -2
Pass
R = +10
0.4
0.4
0.2
Pub
0.4
R = +1

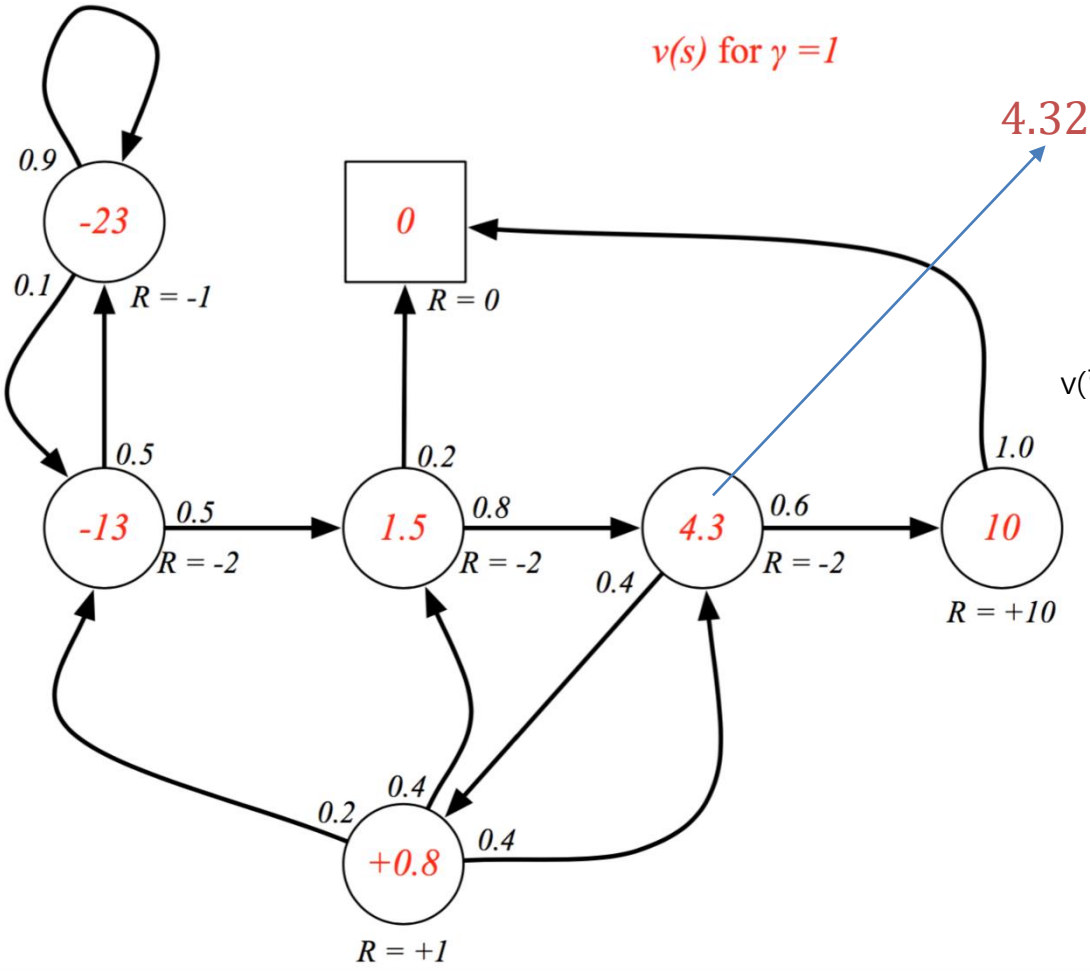
마코프 보상 프로세스



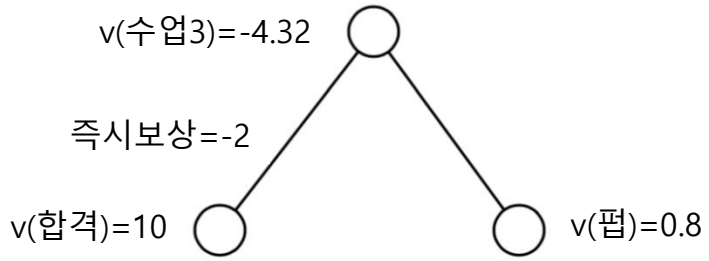
$$-2 = -2 + 0 \times (0.6 \times 10 + 0.4 \times 1)$$



마코프 보상 프로세스



$$4.32 = -2 + 1 \times (0.6 \times 10 + 0.4 \times 0.8)$$



마코프 보상 프로세스

행렬을 이용한 마코프 보상 프로세스의 풀이

$$v = R + \gamma P v$$

$$\begin{bmatrix} v(1) \\ \dots \\ v(n) \end{bmatrix} = \begin{bmatrix} R_1 \\ \dots \\ R_n \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \dots & & \dots \\ P_{n1} & \dots & P_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \dots \\ v(n) \end{bmatrix}$$

$$(I - \gamma P)v = \mathrm{R}$$

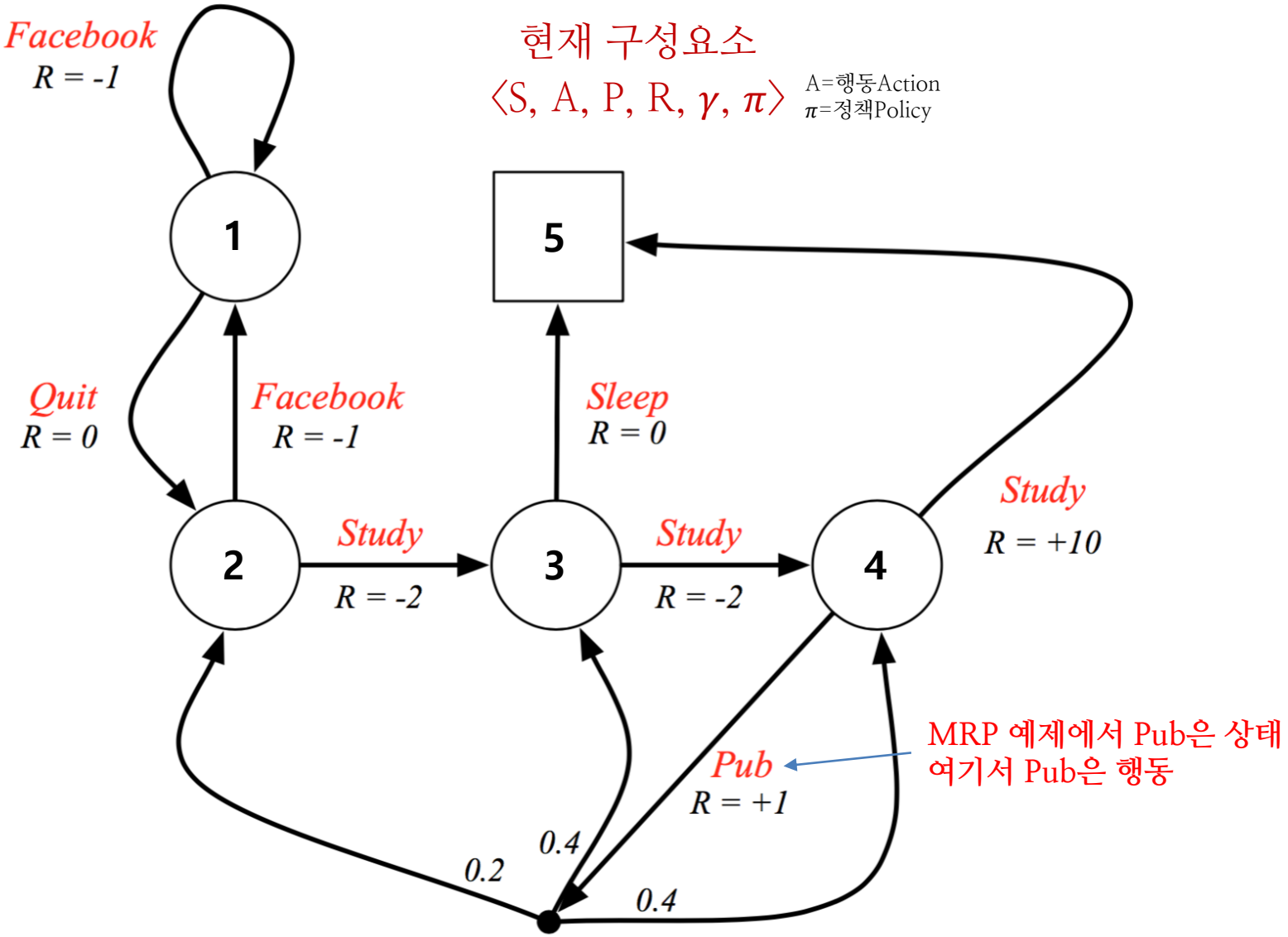
$$v = (I - \gamma P)^{-1}\mathrm{R}$$

마코프 결정 프로세스

행동 action

에이전트의 의사결정. 마코프 보상 프로세스(MRP)는 보상 중심,
'행동'이 추가된 마코프 결정 프로세스(MDP)에선 행동 중심 가치 평가.

마코프 결정 프로세스


Facebook
R = -1
현재 구성요소
〈S, A, P, R, γ, π〉
A=행동Action
π=정책Policy
1
5
Quit
R = 0
Facebook
R = -1
Sleep
R = 0
Study
R = +10
2
Study
R = -2
3
Study
R = -2
4
MRP 예제에서 Pub은 상태
여기서 Pub은 행동
Pub
R = +1
0.4
0.2
0.4

마코프 결정 프로세스

정책 policy

상태state와 행동action을 연결해주는 함수.
상태 s가 주어졌을 때 행동 a를 할 확률로 정의.

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

제약조건 두가지

$$\sum_{a \in A} \pi(a|s) = 1$$

정책은 시간의 흐름과 무관(stationary, time-dependent)하다.

마코프 결정 프로세스

행동가치함수 action-value function

상태 s에서 행동 a를 취하고 정책 π를 따랐을 때 기대되는 가치의 총합

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

정책이 고려된 상태가치함수

상태 s에서 정책 π를 따랐을 때 기대되는 가치의 총합

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$

마코프 결정 프로세스의 목적은 가장 좋은 정책, 즉 상태가치가 가장 큰 정책을 찾는 것이다.

마코프 결정 프로세스

행동가치함수 action-value function

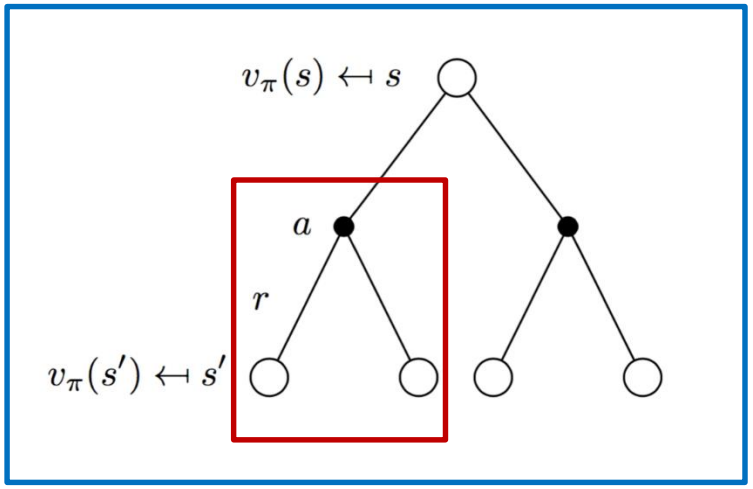
상태 s에서 행동 a를 취하고 정책 π를 따랐을 때 기대되는 가치의 총합

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

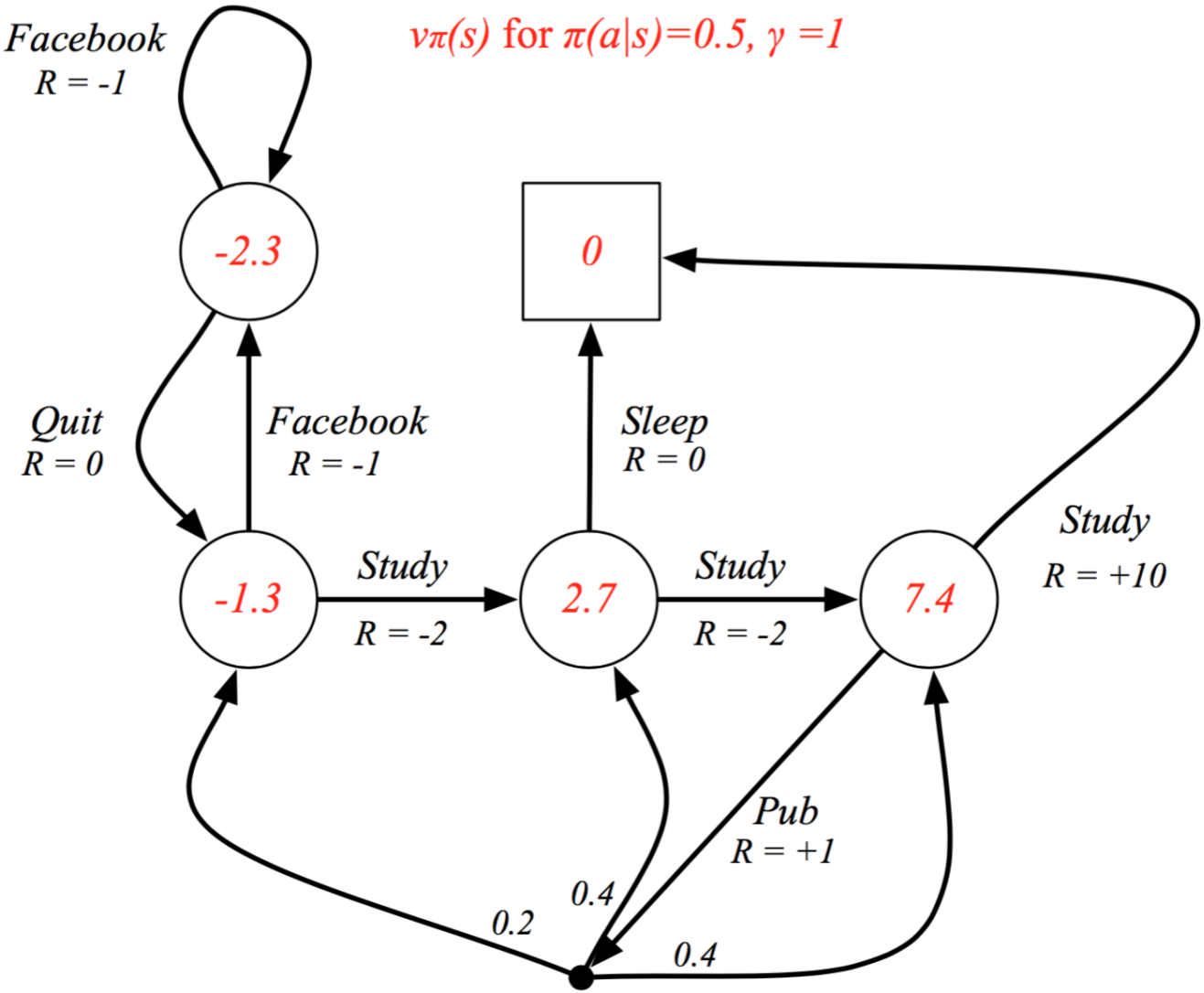
정책이 고려된 상태가치함수

상태 s에서 정책 π를 따랐을 때 기대되는 가치의 총합

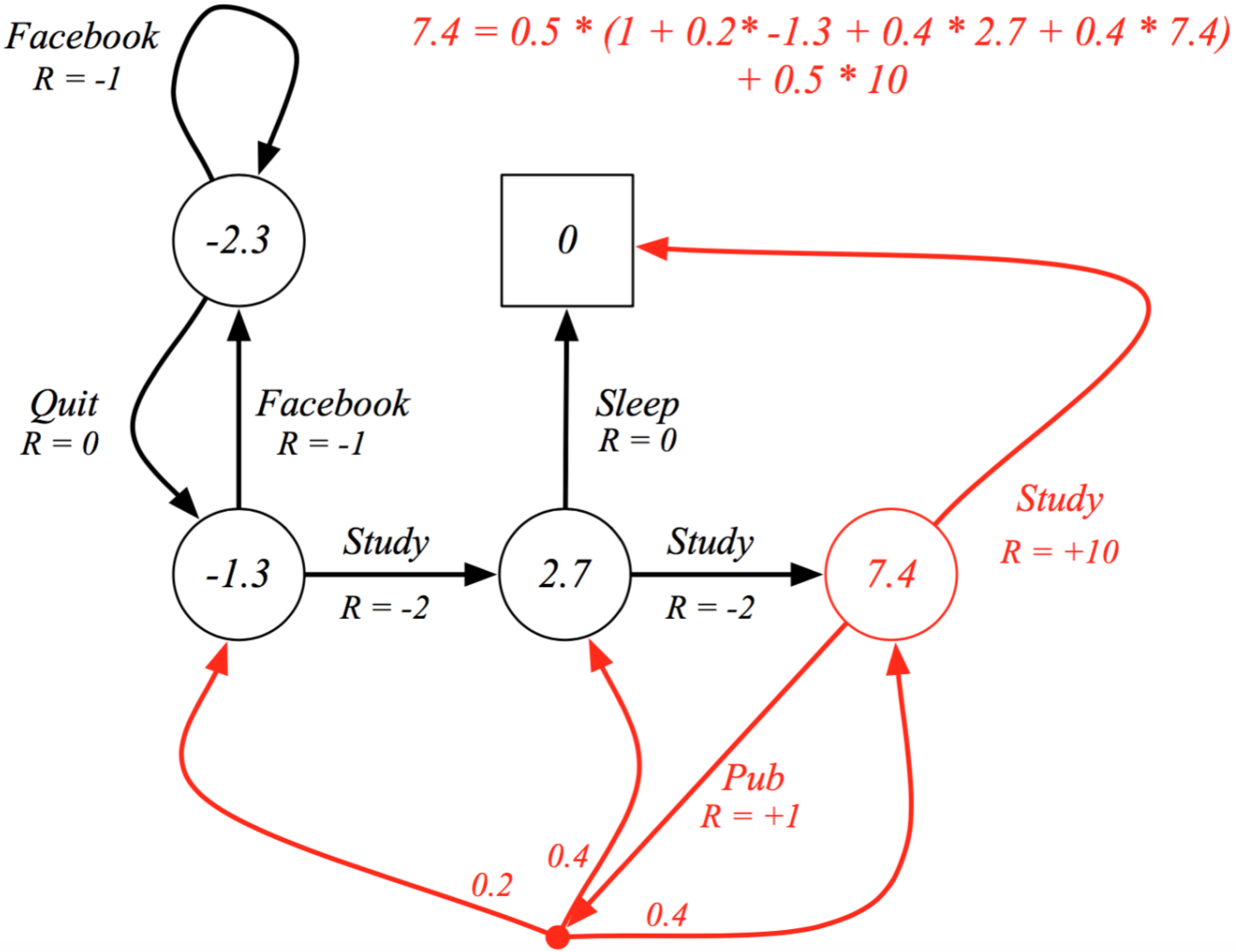
$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$



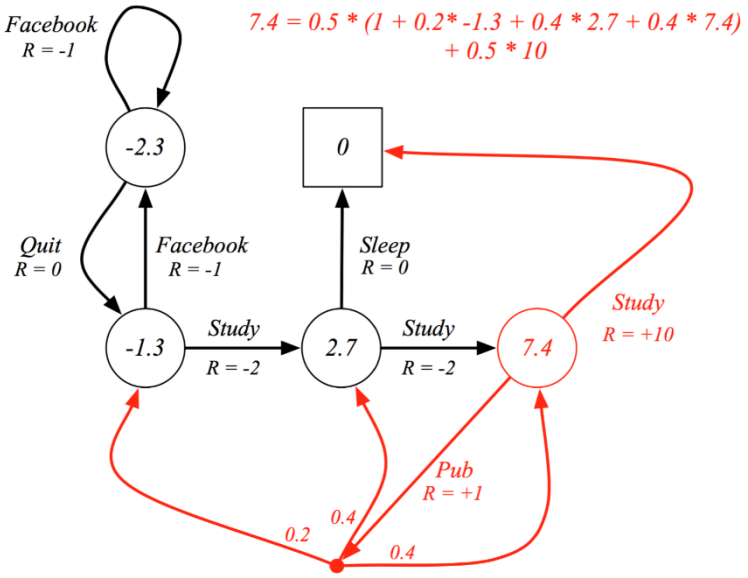
마코프 결정 프로세스


Facebook
R = -1
vπ(s) for π(a|s)=0.5, γ =1
-2.3
0
Quit
R = 0
Facebook
R = -1
Sleep
R = 0
Study
R = +10
-1.3
Study
R = -2
2.7
Study
R = -2
7.4
Pub
R = +1
0.4
0.2
0.4

마코프 결정 프로세스


Facebook
R = -1
7.4 = 0.5 * (1 + 0.2* -1.3 + 0.4 * 2.7 + 0.4 * 7.4)
+ 0.5 * 10
-2.3
0
Quit
R = 0
Facebook
R = -1
Sleep
R = 0
Study
R = +10
-1.3
Study
R = -2
2.7
Study
R = -2
7.4
Pub
R = +1
0.4
0.2
0.4

마코프 결정 프로세스



$v_\pi(\text{상태}4) = 7.4$

공부

펍

$R_{\text{상태}_4}\text{공부} = 10$

$R_{\text{상태}_4}\text{펍} = 1$

$v_\pi(\text{상태}5) = 0$

$v_\pi(\text{상태}2) = -1.3$

$v_\pi(\text{상태}3) = 2.7$

$v_\pi(\text{상태}4) = 7.4$

마코프 결정 프로세스

행렬을 활용한 계산

$$v_\pi = R^\pi + \gamma P^\pi v_\pi$$

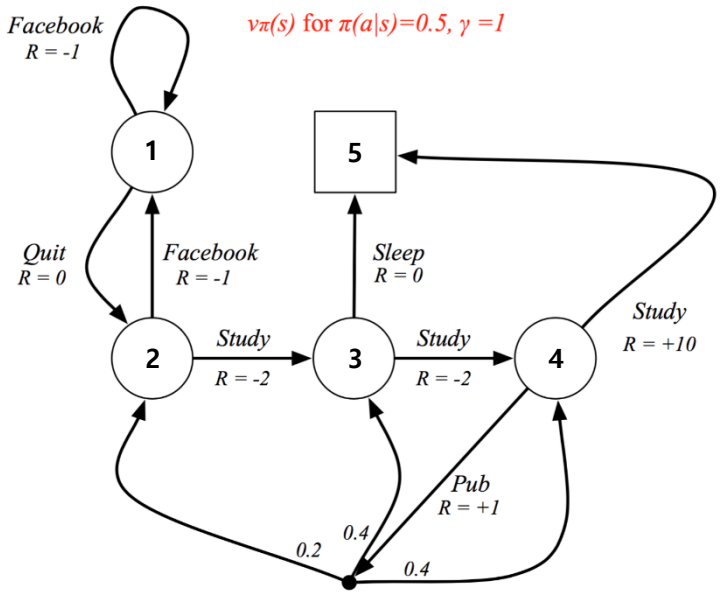
$$v_\pi - \gamma P^\pi v_\pi = R^\pi$$

$$(I - \gamma P^\pi) v_\pi = R^\pi$$

$$v_\pi = (I - \gamma P^\pi)^{-1} R^\pi$$

마코프 결정 프로세스

행렬을 활용한 계산



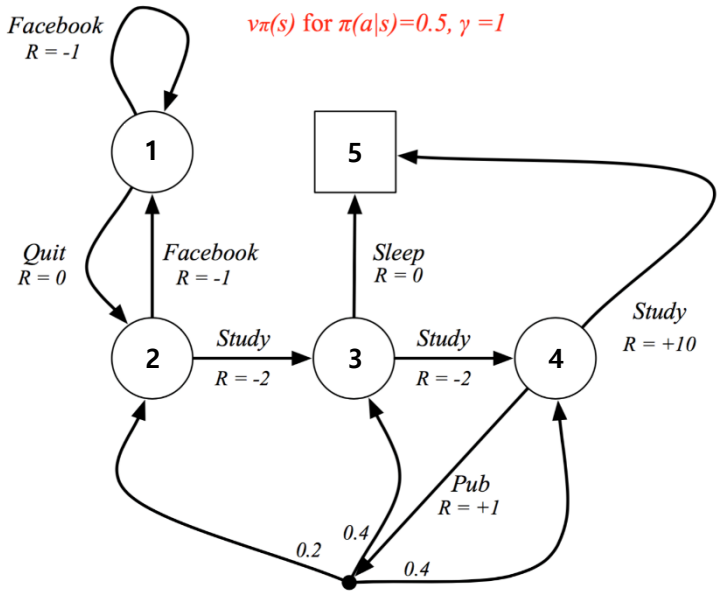
$$\pi(s,a) = \begin{array}{c} \\ \text{상태1} \\ \text{상태2} \\ \text{상태3} \\ \text{상태4} \\ \text{상태5} \end{array} \begin{array}{c} \begin{array}{ccccccccc} \text{공부1} & \text{공부2} & \text{공부3} & \text{펍} & \text{페북1} & \text{페북2} & \text{중지} & \text{취침} \end{array} \\ \begin{bmatrix} & & & & 0.5 & & 0.5 & \\ 0.5 & & & & & 0.5 & & \\ & 0.5 & & & & & & 0.5 \\ & & 0.5 & 0.5 & & & & \\ & & & & & & & \end{bmatrix} \end{array}$$

마코프 결정 프로세스

행렬을 활용한 계산

$$P^{a}_{ss'} =$$

	상태1	상태2	상태3	상태4	상태5
공부1			1		
공부2				1	
공부3					1
펍		0.2	0.4	0.4	
페북1	1				
페북2	1				
중지		1			
취침					1



$$R^{a}_{s} = [-2 \quad -2 \quad 10 \quad 1 \quad -1 \quad -1 \quad 0 \quad 0]^{\mathrm{T}}$$

공부1 공부2 공부3 펍 페북1 페북2 중지 취침

마코프 결정 프로세스

행렬을 활용한 계산

$$P_\pi = \pi(s,a) \times P_{ss'}^a,$$

$$= \begin{bmatrix} & & & & 0.5 & & 0.5 & \\ 0.5 & & & & & 0.5 & & \\ & 0.5 & & & & & & \\ & & 0.5 & 0.5 & & & & \\ & & & & & & & 0.5 \end{bmatrix} \times \begin{bmatrix} & & 1 & & & \\ & & & & 1 & \\ & & & & & 1 \\ & 0.2 & 0.4 & 0.4 & & \\ 1 & & & & & \\ 1 & & & & & \\ & 1 & & & & \\ & & & & & 1 \end{bmatrix}$$

$$R_\pi = \pi(s,a) \times R_s^a$$

$$= \begin{bmatrix} & & & & 0.5 & & 0.5 & \\ 0.5 & & & & & 0.5 & & \\ & 0.5 & & & & & & \\ & & 0.5 & 0.5 & & & & \\ & & & & & & & 0.5 \end{bmatrix} \times [-2 \quad -2 \quad 10 \quad 1 \quad -1 \quad -1 \quad 0 \quad 0]^{\mathrm{T}}$$

마코프 결정 프로세스

행렬을 활용한 계산

$$P_\pi = \pi(s,a) \times P_{ss'}^a$$

	V1	V2	V3	V4	V5
1	0.5	0.5	0.0	0.0	0.0
2	0.5	0.0	0.5	0.0	0.0
3	0.0	0.0	0.0	0.5	0.5
4	0.0	0.1	0.2	0.2	0.5
5	0.0	0.0	0.0	0.0	0.0

$$R_\pi = \pi(s,a) \times R_s^a$$

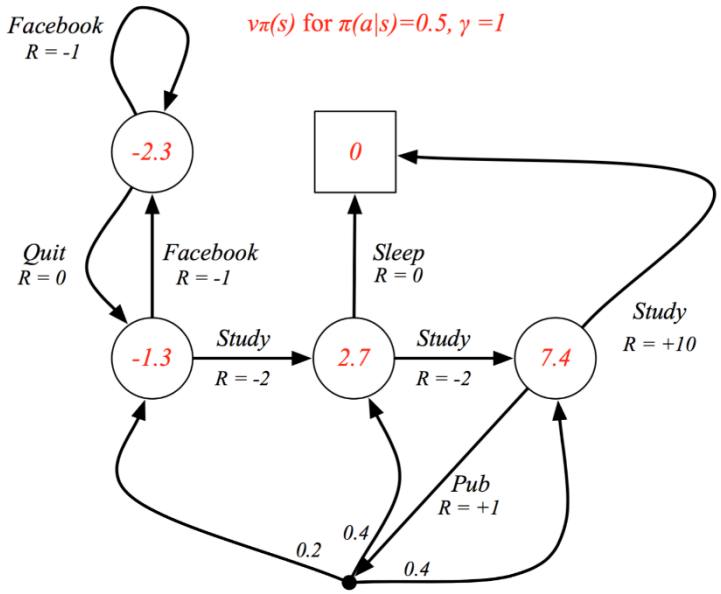
	V1
1	-0.5
2	-1.5
3	-1.0
4	5.5
5	0.0

마코프 결정 프로세스

행렬을 활용한 계산

$$v_\pi = (I - \gamma P^\pi)^{-1} R^\pi$$

	V1
1	-2.307692
2	-1.307692
3	2.692308
4	7.384615
5	0.000000



마코프 결정 프로세스

최적가치함수 optimal value function

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

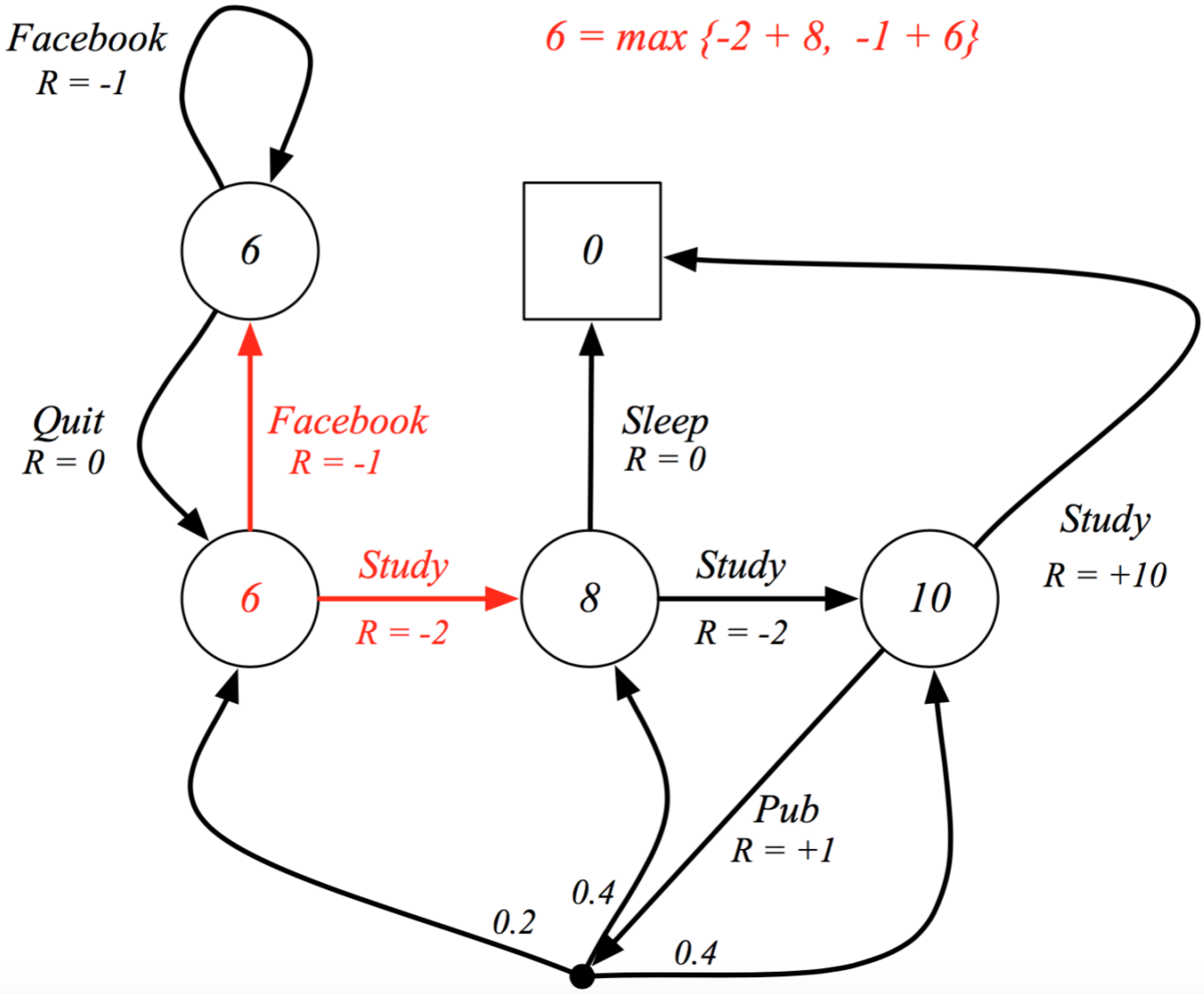
최적행동가치함수 optimal action−value function

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

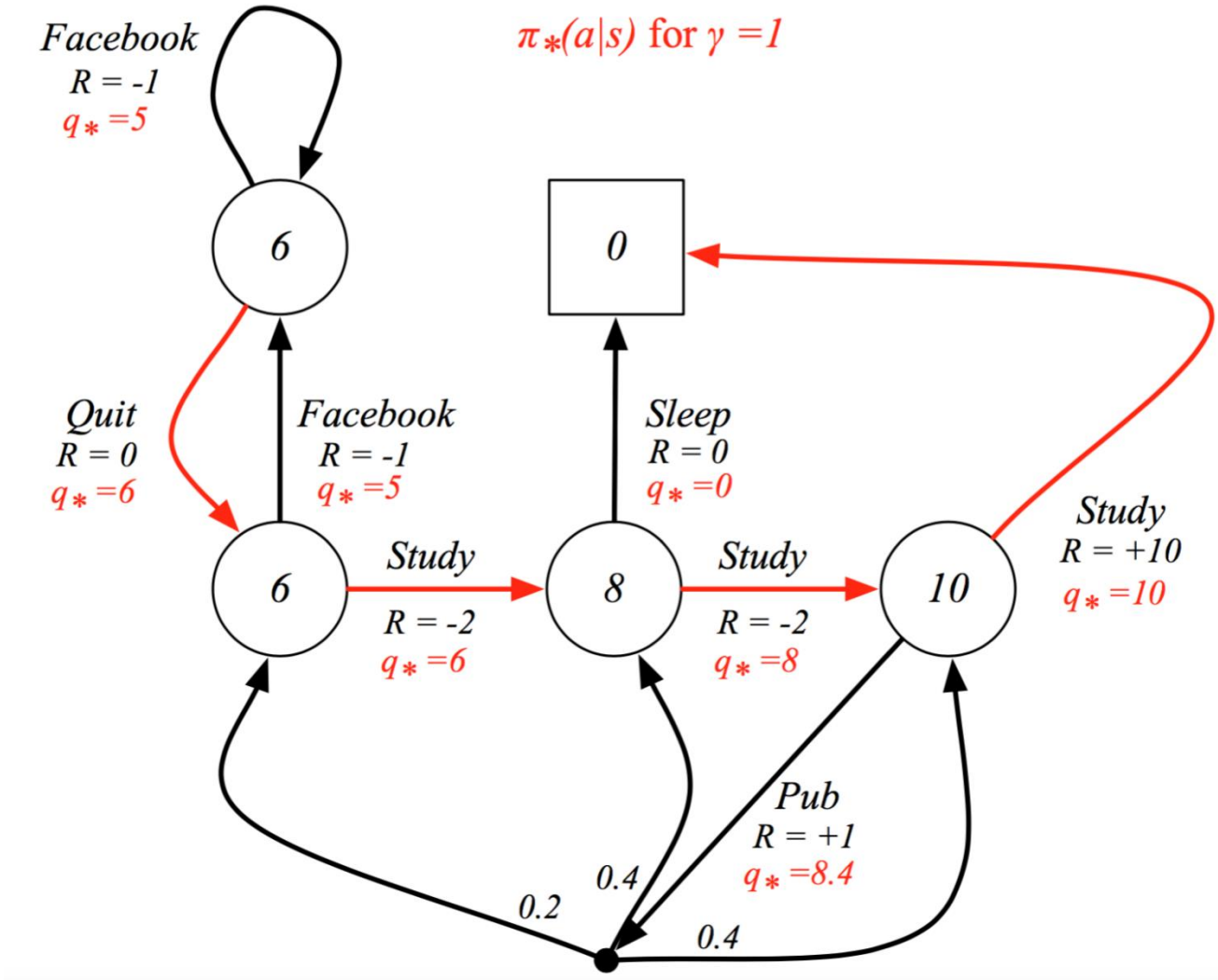
최적 정책 찾기

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, q_*(s,a) \\ 0 & \textit{otherwise} \end{cases}$$

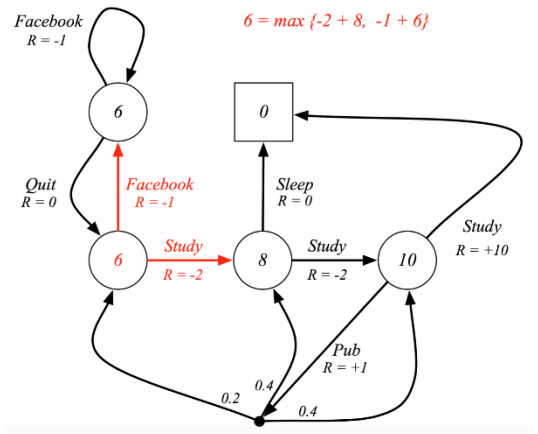
마코프 결정 프로세스

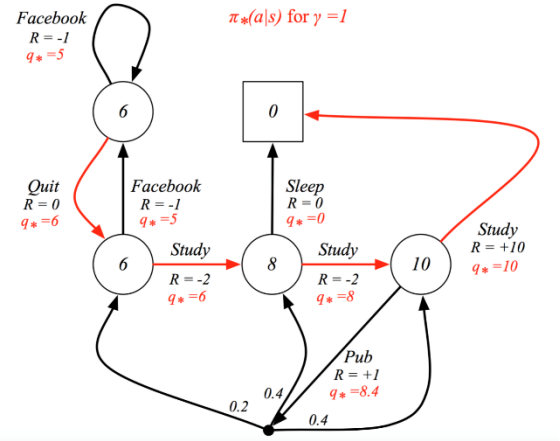

Facebook
R = -1
6 = max {-2 + 8, -1 + 6}
6
0
Quit
R = 0
Facebook
R = -1
Sleep
R = 0
Study
R = +10
6
Study
R = -2
8
Study
R = -2
10
Pub
R = +1
0.4
0.2
0.4

마코프 결정 프로세스


Facebook
R = -1
q* =5
π*(a|s) for γ =1
6
0
Quit
R = 0
q* =6
Facebook
R = -1
q* =5
Sleep
R = 0
q* =0
Study
R = +10
q* =10
6
Study
R = -2
q* =6
8
Study
R = -2
q* =8
10
Pub
R = +1
q* =8.4
0.4
0.2
0.4

마코프 결정 프로세스





공부1 공부2 공부3 펍 페북1 페북2 중지 취침

$$\pi_*(s,a) = \begin{matrix} \text{상태1} \\ \text{상태2} \\ \text{상태3} \\ \text{상태4} \\ \text{상태5} \end{matrix} \begin{bmatrix} & & & & 1 & & \\ 1 & & & & & & \\ & 1 & & & & & \\ & & & 1 & & & \\ & & & & & & \end{bmatrix}$$

$v_*(s) =$

	V1
1	6
2	6
3	8
4	10
5	0

$q_*(s) =$

	V1
1	6.0
2	8.0
3	10.0
4	9.4
5	5.0
6	5.0
7	6.0
8	0.0

마코프 결정 프로세스

뫼비우스의 띠?

벨만방정식은 최적 상태가치, 최적 행동가치, 최적 정책이
모두 맞물려 있는 형태, 어느 하나를 정확히 알면 모든 문제가 풀림

```
p_pi <- pi_star %*% p
r_pi <- pi_star %*% r

v_star <- solve(I-p_pi) %*% r_pi
q_star <- r + p %*% v_star
```

마코프 결정 프로세스

벨만방정식 특성

벨만방정식은 반복적인 방식으로 해를 찾아야 함
해를 구하기 위해서 다음과 같은 알고리즘이 일반적으로 사용됨

동적프로그래밍
- 가치반복법 value iteration
- 정책반복법 policy iteration

시간차방법
- Q-learing
- State Action Reward State Action

몬테카를로방법

감사합니다.