

### 1. Introduction

### 2. Markov Process (Markov Chain) $\langle S, P \rangle$

#### 2-1. Markov Property

#### 2-2. State Transition Matrix

#### 2-3. Definition

#### 2-4. Example

### 3. Markov Reward Process $\langle S, P, R, r \rangle$

#### 3-1. Definition

#### 3-2. Return

#### 3-3. Value Function

#### 3-4. Bellman Equation for MRP

##### 3-4-1. Bellman Equation의 의미

##### 3-4-2. Bellman Equation for MRP

#### 3-5. Bellman Equation in Matrix Form

### 4. Markov Decision Process $\langle S, A, P, R, r \rangle$

#### 4-1. Definition

#### 4-2. Policy

#### 4-3. Value Functions

##### 4-3-1. State Value Function

##### 4-3-2. Action Value Function

#### 4-4. Bellman Expectation Equation

##### 4-4-1. $v$ 을 $q$ 로, $q$ 을 $v$ 로 표현하기

##### 4-4-2. $v$ 을 $v$ 로, $q$ 을 $q$ 로 recursive하게 표현

##### 4-4-3. Solving the Bellman Expectation Equation

#### 4-5. Optimal Value Function

#### 4-6. Optimal Policy

#### 4-7. Bellman Optimality Equation

##### 4-7-1. $v$ 을 $q$ 로, $q$ 을 $v$ 로 표현하기

##### 4-7-2. $v$ 을 $v$ 로, $q$ 을 $q$ 로 recursive하게 표현

##### 4-7-3. Solving the Bellman Optimality Equation

### 5. Summary

## 1. Introduction

---

- #. Agent가 환경을 모두 관측 가능한 상황을 MDP라고 하며, 이러한 환경(문제)에서 순차행동결정 문제를 어떻게 풀어나갈 것인지가 강화학습의 핵심이다. 순차행동결정 문제를 푼다는 것은 Bellman Equation을 사용하여 Optimal Value Function과 Optimal Policy를 구하는 것을 의미한다.
- a. 환경을 모두 관측 가능하다는 의미는 MDP의 구성요소인  $\langle S, A, P, R, r \rangle$  중  $S, A, r$ 은 주어져 있다고 항상 가정하므로  $P, R$ 을 알고 있다는 의미이다. 따라서  $P, R$ 을 이용하여 Bellman Equation을 풀 수 있다.
- b. MDP 전 까지는 Action이 없다는 사실에 주목해야한다. (No Policy)

## 2. Markov Process (Markov Chain) $\langle S, P \rangle$

### 2-1. Markov Property

#.  $P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_t]$ 가 성립하면 State  $s$ 는 Markov하다고 하며, 이를 Markov Property라고도 한다.

### 2-2. State Transition Matrix

#. Action없이 매 Time Step  $t$  마다 정해진 확률로 다음 State로 이동하는 것을 표현한 행렬로, 아래와 같다.

State Transition Probability	State Transition Matrix (Square Matrix)
$P_{ss'} = P[S_{t+1} = s'   S_t = s]$	$P = \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{1n} & \dots & P_{nn} \end{bmatrix}$

a. 각 Row의 합은 1이며  $n$ 개의 State가 있으면  $n \times n$  크기가 된다.

### 2-3. Definition

#. State Set  $S$ 와 State Transition Matrix  $P$ 을 Tuple 형식  $\langle S, P \rangle$ 로 표현한 것이며, Action없이 매 Time Step  $t$ 마다  $P$ 에 의해  $s'$ 로 이동하는 Memoryless Random Process이다.

a. Memoryless는 어느 경로를 통해서 현재 State에 왔는지 저장할 필요 없이, 현재 State만 있으면  $P$ 을 이용하여 다음 State를 결정할 수 있다는 의미이다.

b. Random Process는 확률적 요소인 State Transition Matrix를 바탕으로 Random적인 속성으로 State간의 Process, 즉 Sequence가 이루어졌다는 것을 의미한다. 다른 말로 Sampling<sup>1)</sup>이 가능하다는 것이다.

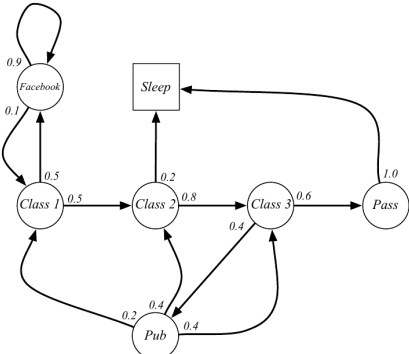
c. State Transition Matrix가 어떤 조건들을 만족하면, Markov Process의 최종 분포가 Stationary<sup>2)</sup>하다.

1) Sampling은 어느 환경에서 Terminal State까지의 State Sequence를 직접 해볼 수 있다는 것이다.

2) 각 State 마다 고르게 분포되어 있는 상태.

## 2-4. Examples

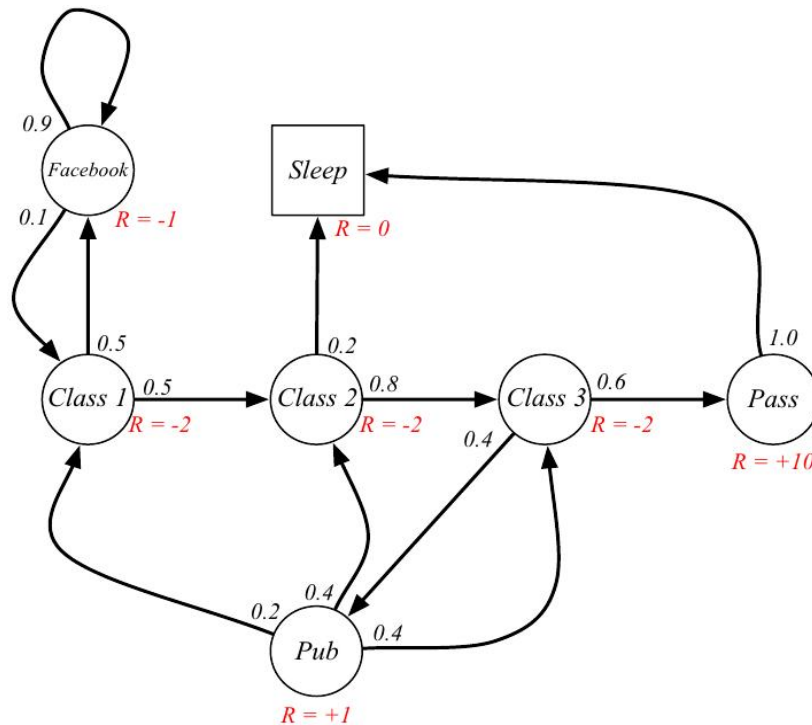
#.

Markov Process	Markov Chain Episodes	State Transition Matrix
<ul style="list-style-type: none"> <li>7개의 States와 State Transition Probabilities</li> </ul>	<ul style="list-style-type: none"> <li>Sampling의 결과</li> </ul>	<ul style="list-style-type: none"> <li>맨 왼쪽그림에서 화살표를 이용하여 Matrix형태로 표현</li> </ul>
	<p>C1 C2 C3 Pass Sleep</p> <p>C1 FB FB C1 C2 Sleep</p> <p>C1 C2 C3 Pub C2 C3 Pass Sleep</p> <p>C1 FB FB C1 C2 C3 Pub C1 FB FB</p> <p>FB C1 C2 C3 Pub C2 Sleep</p>	$P = \begin{matrix} & \begin{matrix} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{matrix} \\ \begin{matrix} C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{matrix} & \begin{bmatrix} & & & & & & \\ & 0.5 & & & & & \\ & & 0.8 & & & & \\ & & & 0.6 & 0.4 & & \\ 0.2 & 0.4 & 0.4 & & & & 1.0 \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{bmatrix}$

#### 3-1. Definition

#. Markov Process 정의에 Reward Function  $R_s = E[R_{t+1} | S_t = s]$ 과 Discount Factor  $\gamma \in [0, 1]$ 가 포함된 것이며  $\langle S, P, R, \gamma \rangle$  형태로 표현한다. 마찬가지로 Action없이 진행되는데, Action이 없으므로 각각의 State에 Reward<sup>3)</sup>을 주게 된다.

a. Example :



#### 3-2. Return

#. 특정 시점  $t$ 에서부터 Terminal State(없는 경우도 있음)까지 받을 수 있는 Cumulation of Discounted Rewards를 의미하여 아래와 같이 정의한다.

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

a. Discount Factor 덕분에 수렴성이 증명이 된다고 하지만, 문제에 따라서 Discount Factor를 사용하지 않을 수 있다.

### 3-3. Value Function

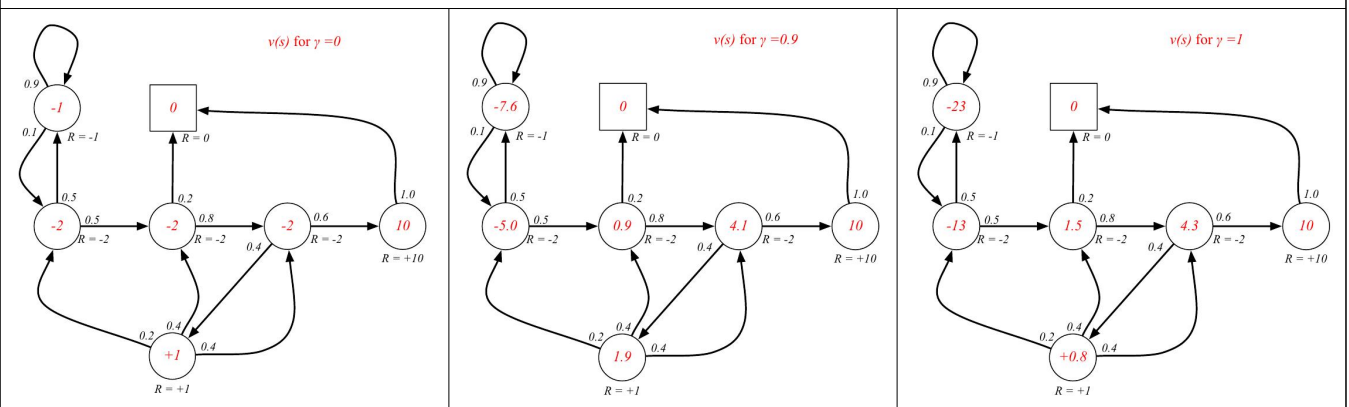
#. 어떤 State  $s$  자체의 Long-Term Value이며, 아직 Action이 없기 때문에  $v(s) = E[G_t | S_t = s]$ 으로 표현하고 구체적으로는  $s$ 을 시작점으로 하는 Returns of Many Sampled Episodes의 산술평균값이다.

- Action이 아직 없으므로 State마다 Reward가 존재한다. 그래서 한 Episode가 끝나면 그 때 마다 Return 값이 고정되어 나온다.
- MDP인 경우에서도 이런 식으로 Value Function을 구한다. 하지만 수학적으로 구하려면 각 State마다의 Action을 고려하여 해당 State에서 각각의 Action을 선택할 확률 값을 곱해주고 더하는 과정이 필요하다.
- Example :

Value Function	
C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8} = -2.25$
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} = -3.125$
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots = -3.41$
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots = -3.20$
FB FB FB C1 C2 C3 Pub C2 Sleep	

• State C1의 경우 위의 값을 바탕으로 계산하면, 위의 값들의 산술 평균값이 된다.

$\gamma$  값의 변화에 따른 Value Function의 변화



### 3-4. Bellman Equation for MRP

#### 3-4-1. Bellman Equation의 의미

#. 강화학습과 같이 순차행동결정 문제에서 현재 상태와 다음 상태의 관계를 표현한 방정식이며 반복적인 Update연산을 통해 특정 Time Step  $t$ 에서의 (Action) State Value Function을 구할 수 있다.

- 현재 상태와 다음 상태의 관계를 표현한 식이므로 같은 Term이 점화식처럼 Recursive하게 표현된다.
- 이 경우 연산량은  $O(n^3)$ 이므로 작은 경우에는 Matrix Form으로 풀 수 있지만, State 개수가 많은 경우에는 Iterative 방법<sup>4)</sup>으로 푼다.

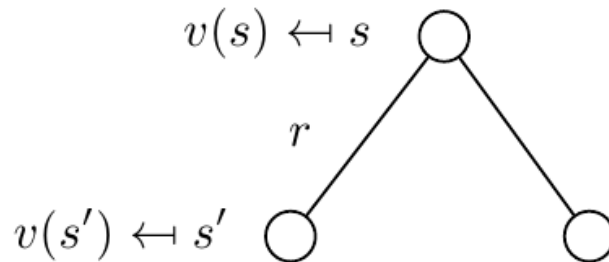
서강대학교 머신러닝 연구실

4) Policy Iteration, Value Iteration (Dynamic Programming), Q-learning, SARSA 등이 있다.

### 3-4-2. Bellman Equation for MRP

#.  $v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s')$ 으로 표현된다. 즉 현재 State  $s$ 에서의 Reward에다 한 Step 더 갔을 때, 가능한 모든  $s'$ 의 Value Functions의 기댓값(평균값)을 Discount하여 더한 값이다.

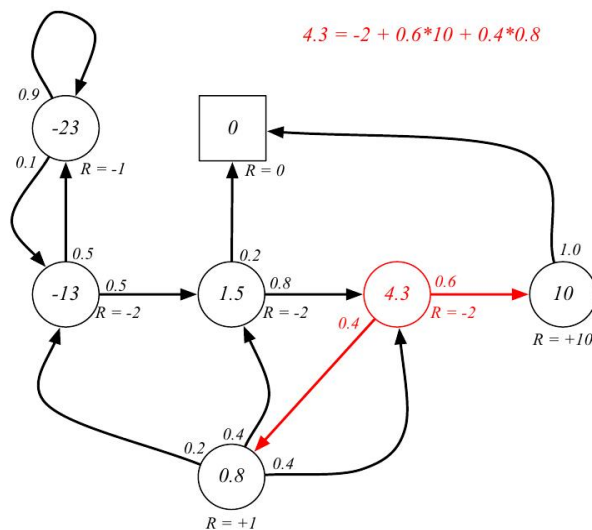
- a. 예를 들어 아래와 같이 한 Step을 더 가면 왼쪽, 오른쪽 둘 다 갈 수 있는 상황에서 왼쪽으로 갈 확률은 0.2, 오른쪽은 0.8이라 했을 때,  
'현재 State의 Reward' + (0.2×왼쪽 State의 Value Function + 0.8×오른쪽 State의 Value Function)'으로 계산된다.



- b. MRP에서의 Value Function의 정의를 이용하여 Bellman Equation을 도출하는 과정은 아래와 같다.  
즉 위에서  $E[]$  연산을 떼어내어 계산하기 위한 식으로 만든 것이다.

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

- c. Example :



### 3-5. Bellman Equation in Matrix Form

#. Matrix Form을 사용하여 선형방정식으로 풀이할 수 있는데, MRP의 정의대로  $\langle S, P, R, \gamma \rangle$  만 주어지면 작은 경우에 한해서만 아래와 같이 Linear하게 Solution( $v$ )을 구할 수 있다.

$$v = R + \gamma P v \quad (v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s'))$$

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

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

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

a. 위의 방정식에서 하나의 Row가 하나의 State를 의미한다.

b. State가  $n$ 개 있을 때, 필요한 연산량은  $O(n^3)$ 이므로 State가 조금만 많아져도 계산하기 어렵다.



#### 4-1. Definition

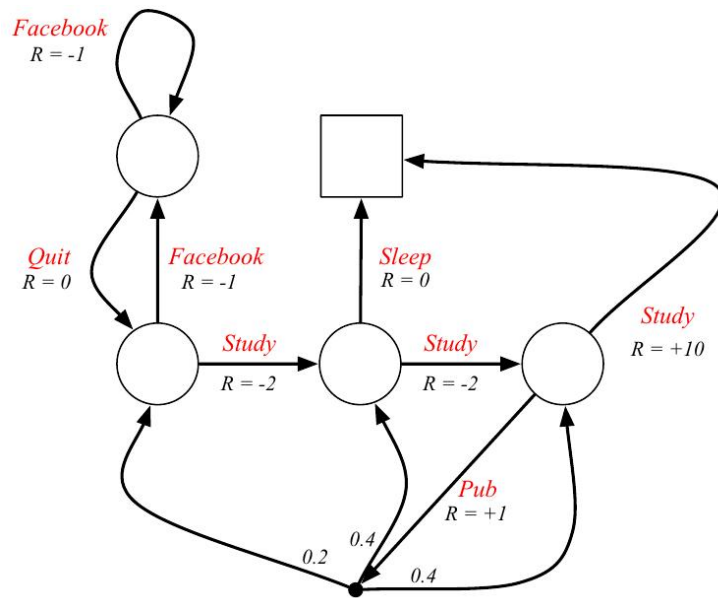
#. MRP에서 Action이 추가된 상태이며  $\langle S, A, P, R, \gamma \rangle$ 로 표현한다. 그리고 Action이 추가됨으로써  $P, R$ 의 정의가 아래와 같이 달라지며,  $P$ 는  $(a \times n \times n)$ 크기가 된다. 또한  $s$ 에서 확률적으로 Action을 선택하게 되고 Action을 통해서 확률적으로 다음 State로 이동하게 된다.

$$A = \text{set of actions}$$

$$P_{ss'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$$

$$R_s^a = E[R_{t+1} | S_t = s, A_t = a]$$

a. Example<sup>5)</sup> : 이제 State말고 Action을 뜻하는 Edge에 Reward가 할당되고, pub에 가는 Action을 취하면 다음 State는 0.2, 0.4, 0.4 각각의 확률로 이동하게 된다.

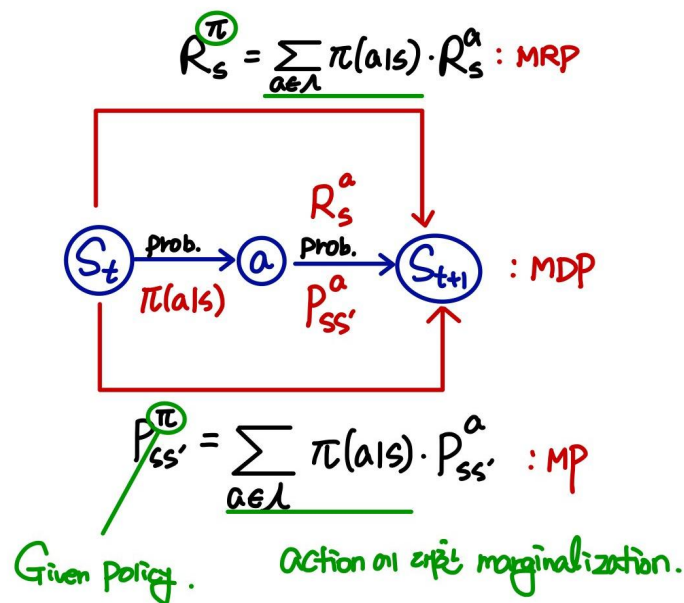


## 4-2. Policy

#. Policy는 주어진 State  $s$ 가 있을 때, 선택가능한 각각의 Action  $a$ 을 선택할 확률 값으로 Mapping하는 것이며, 하나의 확률 분포라 볼 수 있다. Deep Neural Net, Tabular Method 또는 Linear Function Approximation을 이용하여 구현하고, 아래와 같이 정의한다.

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

- a. 모든 강화학습 Algorithm은 Env.와 상호작용의 결과로 Policy를 변화(Update)시키는 것이다.
- b. MDP를 푸는 것은 Return을 최대화하는 Optimal Policy를 찾아내는 것을 의미한다.
- c. 확률분포함수이므로 Input에는 State가 들어가게 되고 Output은 Action의 가짓수만큼 길이의 Vector이고 각 원소는 Action을 취할 확률을 의미한다.
- d. MP, MRP, MDP의 재정의 :



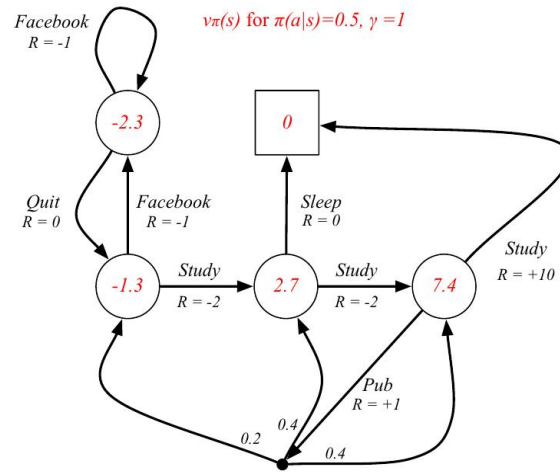
### 4-3. Value Functions

#### 4-3-1. State Value Function

#. 특정 State  $s_t$ 에서 Policy( $\pi$ )을 따라서 Episode가 끝날 때까지 움직이면 하나의 Return값이 나오는데 이를 여러 번 반복(Sampling)하여 평균값을 내는 것이 State Value Function이며 아래와 같이 정의한다.

$$v_{\pi}(s) = E_{\pi}[G_t | S_t = s]$$

- Input 값으로 State  $s_t$  하나만 들어간다.
- State Value Function을 정의하기 위해서는 Parameter인 Policy( $\pi$ )가 정의되어야 한다.
- Example : Uniform Random Policy로 설정.



#### 4-3-2. Action Value Function

#. 특정 State  $s_t$ 에서 Policy( $\pi$ )을 따라 특정 Action  $a_t$ 을 하고 난 이후 그 Policy( $\pi$ )을 따라서 Episode가 끝날 때까지 움직이면 하나의 Return값이 나오는데 이를 여러 번 반복(Sampling)하여 평균값을 내는 것이 Action Value Function이며 아래와 같이 정의한다.

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

- Input 값으로 State  $s_t$ 와 Action  $a_t$ 가 들어간다.

#### 4-4. Bellman Expectation Equation

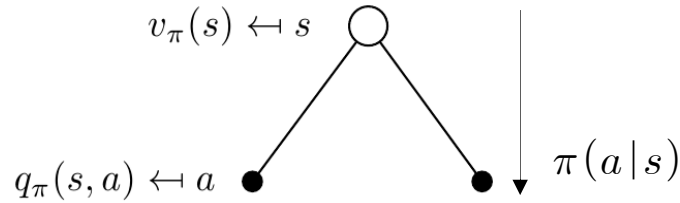
- Bellman Equation에 Expectation Term이 들어간 식이며, MRP에서 했던 것과 같이, 같은 의미의 Term을 이용하여 현재 Step과 다음 Step과의 관계를 수식으로 표현하기 위함이다.
- 이를 위해 (Action) State Value Function의 정의에서 아래와 같이 한 Step 분해할 수 있는데, 계산을 위해  $E[]$  연산을 풀어낼 것이다.

$$v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$$

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

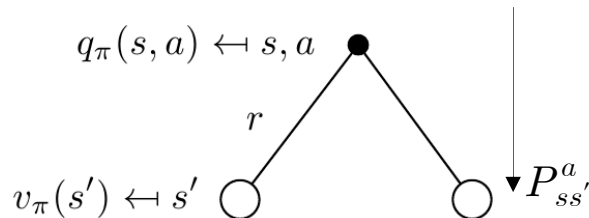
4-4-1.  $v$ 을  $q$ 로,  $q$ 을  $v$ 로 표현하기

- #. State  $s_t$ 에서는 Policy( $\pi$ )을 통해 다양한 Action  $a$ 을 할 수 있으므로 아래와 같이 State  $s$ 에서 선택할 수 있는 모든 Action에 대해 Marginalization을 한다.



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

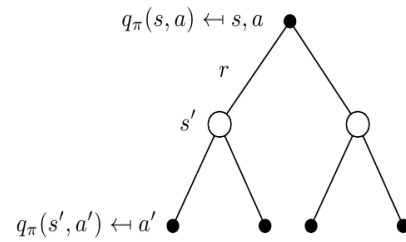
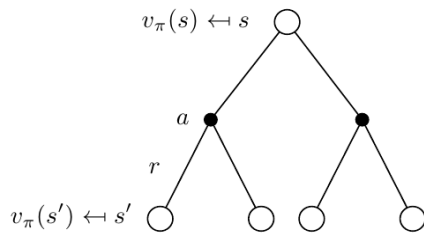
- #. Action  $a$ 을 했을 때, 이에 대한 Reward( $R_s^a$ )를 받고 State Transition Probability에 의해  $s'$ 으로 넘어가므로 아래와 같이 Action  $a$ 을 했을 때, 옮겨갈 수 있는 모든 State에 대해 Marginalization한다.



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

4-4-2.  $v$ 을  $v$ 로,  $q$ 을  $q$ 로 Recursive하게 표현

#. 4-4-1.에서 표현한 식을 서로 대입하면 Bellman Equation의 정의대로 같은 Term끼리 현재( $t$ )와 미래( $t+1$ )의 관계식을 표현할 수 있다.

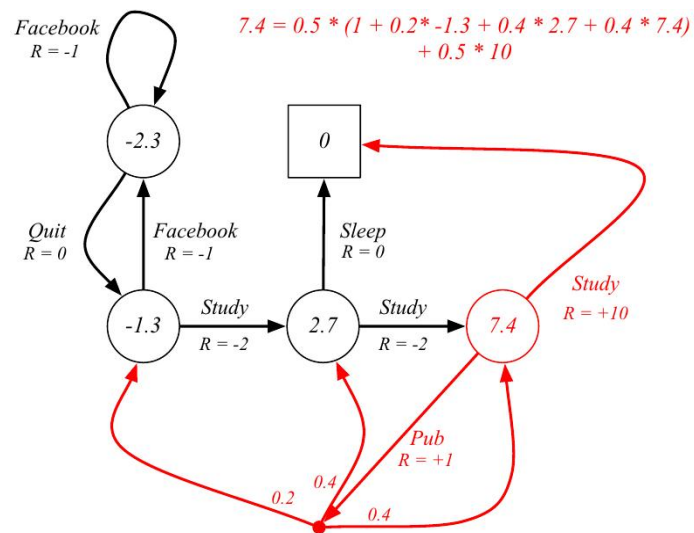


$$v_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_{\pi}(s') \right)$$

$$q_{\pi}(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_{\pi}(s', a')$$

a. 위의 식은 2-step을 표현한 것이다. 왜냐하면  $s_t$ 과  $s_{t+1}$  사이에는  $a_t$ 가,  $a_t$ 과  $a_{t+1}$ 사이엔  $s$ 가 존재하기 때문이다.

b. Example :



c. 이런 식으로 표현한 이유는 3-5.에서처럼 Matrix Form을 이용하여 Solution(Value Functions)을 구하기 위함이다.

#### 4-4-3. Solving the Bellman Expectation Equation

#. 아래와 같이 Value Function과 Action Value Function 각각에 대해 행렬을 이용하여 Linear Equation을 풀면 MDP를 풀 수 있다.

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

$$q_{\pi} = R^{\pi} + \gamma P^{\pi} q_{\pi}$$

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

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

#### 4-5. Optimal Value Function

#. 가능한 모든 Policies에 대해 각각 가능한 Value Function중 값이 가장 큰 것을 Optimal Value Function이라 하며 아래와 같이 각각 표현한다.

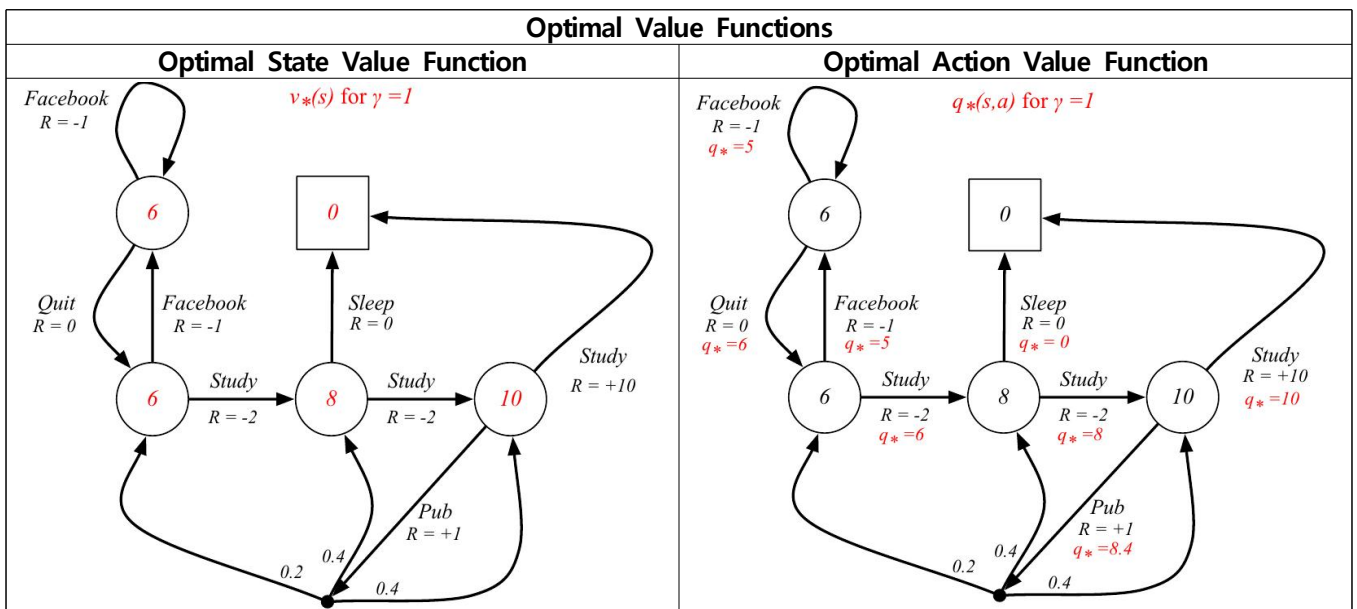
$$v_{*}(s) = \max_{\pi} v_{\pi}(s)$$

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

a. Value Function의 Parameter는 Policy이다.

b. 이를 구하면 MDP를 풀었다고 하지만 풀기가 쉽지 않다. 작은 문제여도 max연산 때문에 Matrix Form으로 풀 수 없는 Non-Linear이기 때문이다.

c. Example



## 4-6. Optimal Policy

#. Policy 정의<sup>6)</sup>상 직접 비교할 수 없으므로 Partial-Ordering을 이용하여 정의하면 아래와 같으며, Policy는 Value Function의 Parameter<sup>7)</sup>이므로 Optimal Policy<sup>7)</sup>을 넣으면 모두 Optimal Value Function이 성립한다.

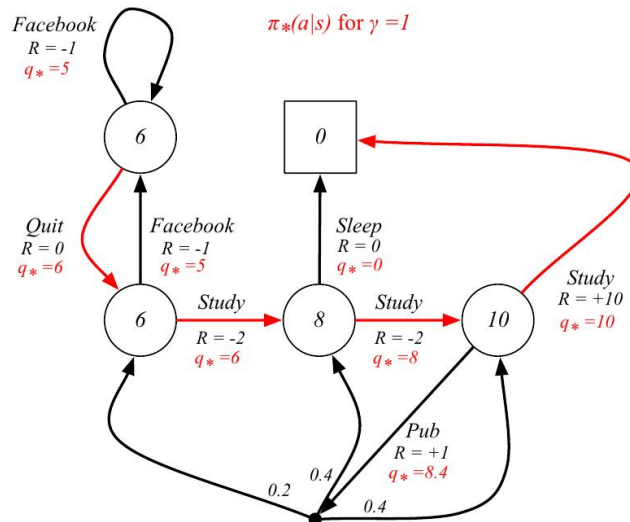
Partial-Ordering
$\pi \geq \pi' \text{ if } v_{\pi}(s) \geq v_{\pi'}(s), \forall s$
$v_{\pi_*}(s) = v_*(s) \quad q_{\pi_*}(s, a) = q_*(s, a)$

a. Optimal Policy는  $q_*$ 을 알면 찾을 수 있다. 왜냐하면  $q_*$ 는  $s$ 에서 가능한 Actions 중 높은 값의  $a$ 을 알려주기 때문에 Deterministic하게 그 Action을 선택하기만 하면 되기 때문이다.

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

b. Stochastic Env.임에도 Optimal Policy는 Deterministic하니 신기한 느낌이 든다. 예를 들어 가위, 바위, 보 게임에서 Optimal Policy는 하나만 내는 것이기 때문이다. 즉 승률 1/3보다 높을 수는 없다.

c. Example



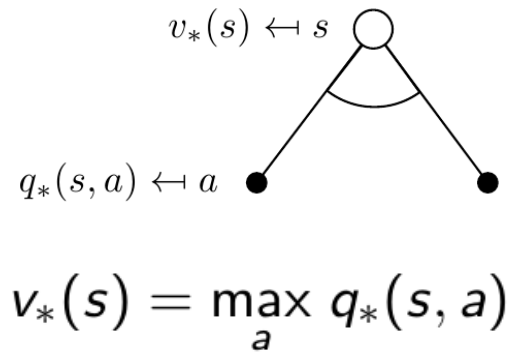
6) Policy는 확률분포(Deep Neural Net., Tabular Method, 또는 Linear Function Approximation으로 표현)이므로 직접 비교하기가 불가능하다.  
7) 여러 Optimal Policies가 존재할 수 있다.

## 4-7. Bellman Optimality Equation

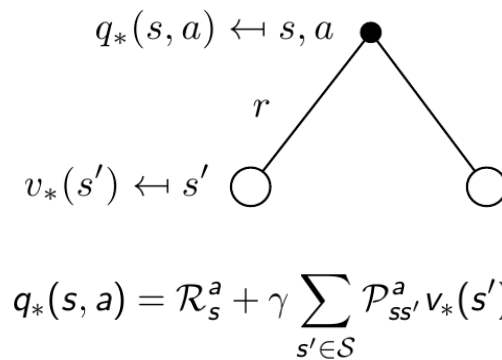
a. Bellman Expectation Equation과 동일한 전개를 하고, Marginalization 대신에  $\max$  연산이 들어가게 된다.

4-7-1.  $v$ 을  $q$ 로,  $q$ 을  $v$ 로 표현하기

#. State  $s$ 에서 가능한 각각의 Action  $a$ 에 대한  $q$  값 중 가장 큰 값이  $v_*(s)$ 이며 아래와 같이 표현된다.



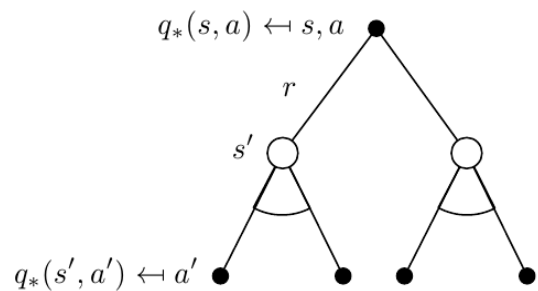
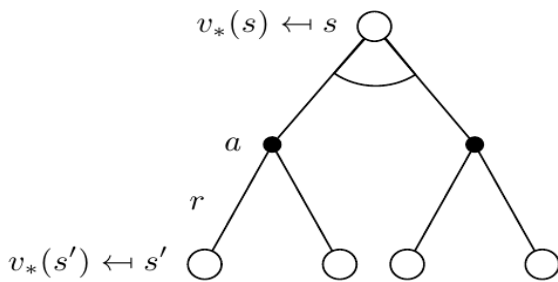
#. Action  $a$ 에 대한 Reward( $R_s^a$ )에다  $a$ 을 통해 이동할 수 있는  $s'$ 의 Optimal Value Function의 기댓값을 더한 값이며 아래와 같이 표현된다.





4-7-2.  $v$ 을  $v$ 로,  $q$ 을  $q$ 로 Recursive하게 표현

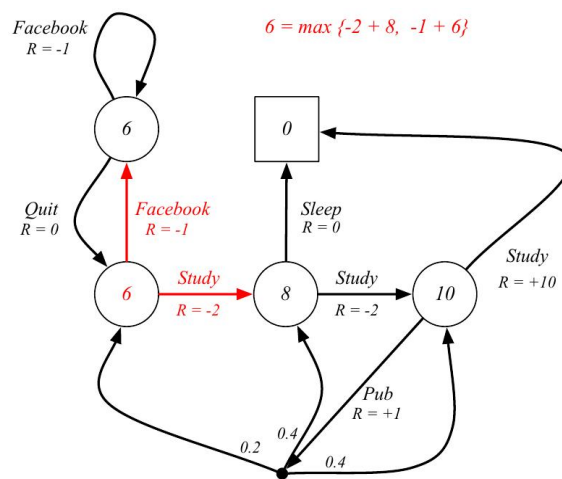
#. 4-7-1.에서 표현한 식을 서로 대입하면 Bellman Equation의 정의대로 같은 Term끼리 현재( $t$ )와 미래 ( $t+1$ )의 관계식을 표현할 수 있다.



$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s') \quad q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s', a')$$

a. 작은 경우에도 'max' 연산 때문에 Non-Linear가 되어 Matrix Form<sup>8)</sup>으로 표현할 수도 없고 풀 수도 없다.

b. Example

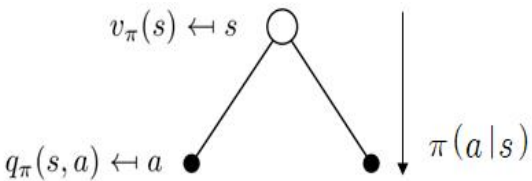
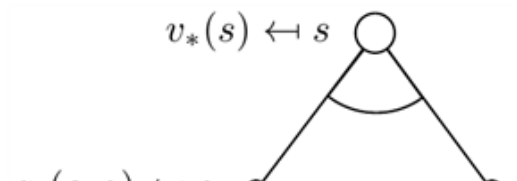
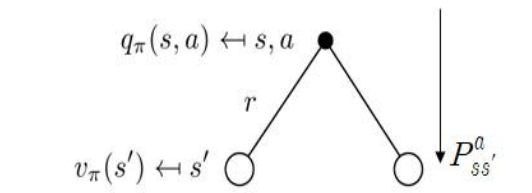
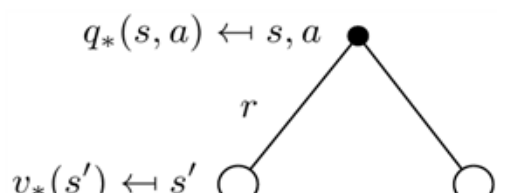


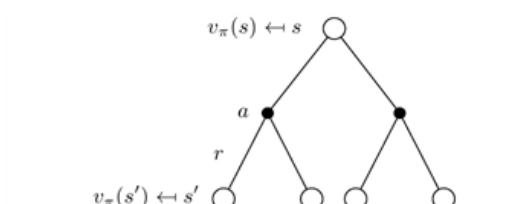
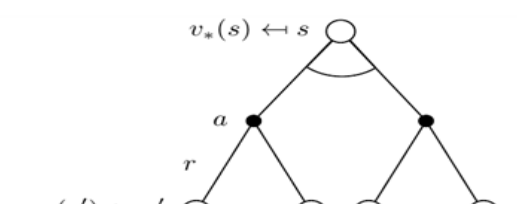
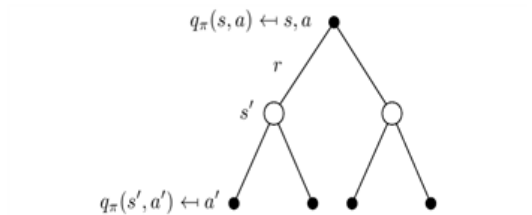
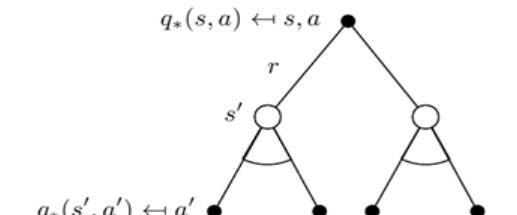
### 4-7-3. Solving the Bellman Optimality Equation

a. Bellman Optimality Equation은 Non-Linear이므로 Closed Form이 없어서 Iterative 방법으로 풀어야 한다.

b. Iterative 방법에는 Dynamic Programming(Policy Iteration, Value Iteration), Q-learning, SARSA가 있다.

## 5. Summary

	Bellman Expectation Equation	Bellman Optimality Equation
$v$ 을 $q$ 로 표현	 $v_{\pi}(s) = \sum_{a \in A} \pi(a s) q_{\pi}(s, a)$	 $v_{*}(s) = \max_a q_{*}(s, a)$
$q$ 을 $v$ 로 표현	 $q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s')$	 $q_{*}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{*}(s')$

	Bellman Expectation Equation	Bellman Optimality Equation
$v$ 을 $v$ 로 표현	 $v_{\pi}(s) = \sum_{a \in A} \pi(a s) \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{\pi}(s') \right)$	 $v_{*}(s) = \max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_{*}(s')$
$q$ 을 $q$ 로 표현	 $q_{\pi}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a' s') q_{\pi}(s', a')$	 $q_{*}(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \max_{a'} q_{*}(s', a')$

서강대학교 머신러닝 연구실

서강현