



Chapter

6\_2

# 강화학습(Reinforcement Learning)

1. Markov Model
2. Markov Decision Process (MDP)
3. Q-Learning
4. Deep Q-Learning(DQN)

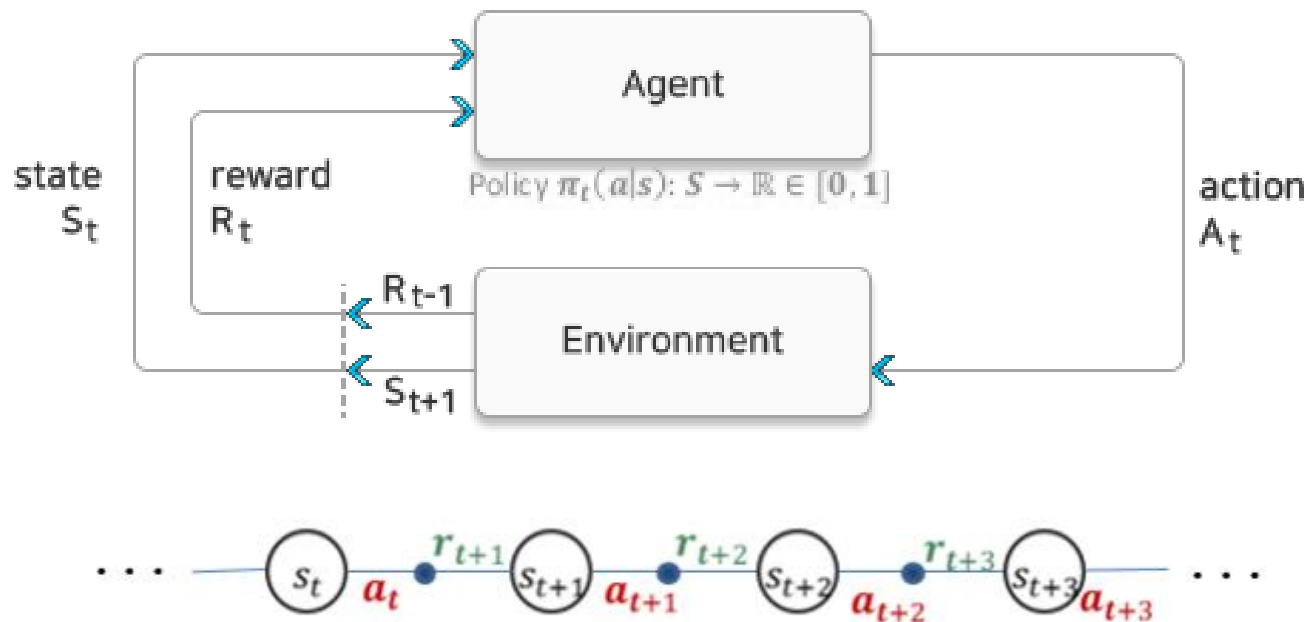
# 강의에 앞서서..

---

- ❖ 본 문서는 아래의 자료들을 활용하여 만들어 졌음을 알립니다
- ❖ 모두를 위한 딥러닝 강좌
  - <https://hunkim.github.io/ml/>
- ❖ Hands on Machine Learning
  - <https://github.com/ugaemi/handson-ml-1>

# 강화학습(Reinforcement Learning)

- ❖ 모르는 환경에서 동작하는 에이전트가 있을 때, 에이전트가 현재 상태 (state)에서 향후 기대되는 누적 보상값(reward)이 최대가 되도록 행동 (action)을 선택하는 정책(policy)을 찾는 문제
- ❖ 마르코프 결정 과정(MDP) 확률 모델로 표현



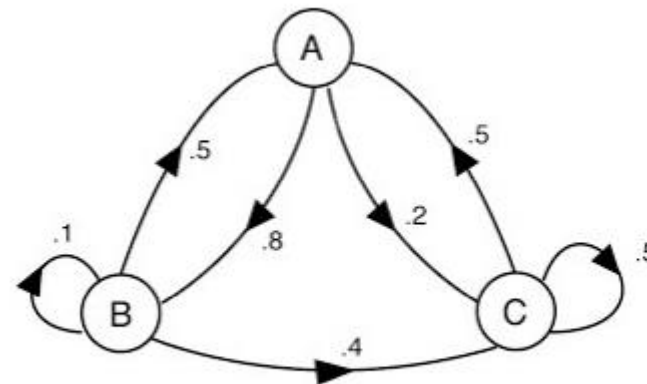
# Markov Model

- ❖ 상태 전이(state transition)가 현재 상태에 의해서 확률적으로 결정
- ❖ 상태(state)
  - $V = \{v_1, \dots, v_m\}$
- ❖ 상태 전이 확률(State transition Probability)
  - 확률  $a_{ij}$ 는 상태  $v_i$ 에서 상태  $v_j$ 로 이동할 확률

$$a_{ij} = P(o_t = v_j \mid o_{t-1} = v_i)$$

$$a_{ij} > 0 \text{ and } \sum_{j=1}^m a_{ij} = 1$$

- ❖ 상태 전이도(state transition diagram)



Markov graph of transition probabilities  
between states A, B and C

# Markov Decision Process (MDP)

## ❖ First-order Markov assumption 기반으로 고안

- 시간  $t$ 에서의 상태는  $t-1$  상태에서만 영향을 받는다

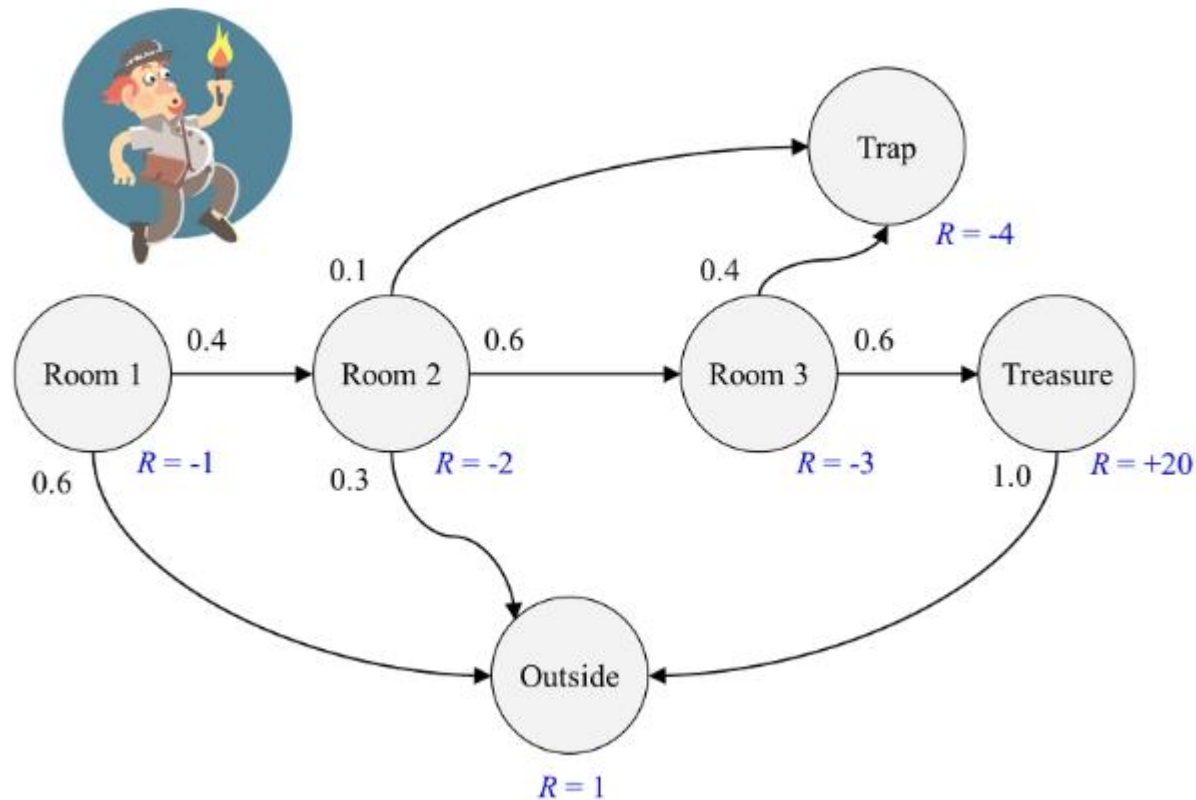
$$p(s_t | s_0, s_1, \dots, s_{t-1}) = p(s_t | s_{t-1})$$

## ❖ Markov reward process : $\langle S, P, R, \gamma \rangle$

- $S$ : state의 집합을 의미한다. State는 바둑에서 바둑판에 돌이 어떻게 놓여져 있는가를, 미로를 탈출하는 문제에서는 현재의 위치를 나타낸다.
- $P$ : 각 요소가  $p(s' | s) = \Pr(S_{t+1} = s' | S_t = s)$ 인 집합이다.  $p(s' | s)$ 는 현재 상태  $s$ 에서  $s'$ 으로 이동할 확률을 의미하며, transition probability라고 한다.
- $R$ : 각 요소가  $r(s) = \mathbb{E}[R_{t+1} | S_t = s]$ 인 집합이다.  $r(s)$ 는 state  $s$ 에서 얻는 reward를 의미한다.
- $\gamma$ : 즉각적으로 얻는 reward와 미래의 얻을 수 있는 reward 간의 중요도를 조절하는 변수이다. 주로  $[0, 1]$  사이의 값을 가지며, discount factor라고 한다.

# Markov Decision Process (MDP)

❖ 유적을 탐험하는 모험가의 상황을 MDP로 표현



# Markov Decision Process (MDP)

## ❖ 누적 보상치(Return)

- t시간 이후부터 얻을 수 있는 보상(reward)의 합

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

- 예:  $\gamma=0.5, S_1 = \text{Room 1} \rightarrow \text{Room 2} \rightarrow \text{Outside}$ 의 순서로 탐험하였을 때  $G_1$

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

## ❖ 상태가치함수(State-value function)

- 목표를 도달하는데 있어서 state  $s$ 가 얼마나 좋은 상태인지를 표현
- $V(s)$ 는 state  $s$ 에서 시작했을 때 얻을 수 있는 누적 보상치의 기댓값

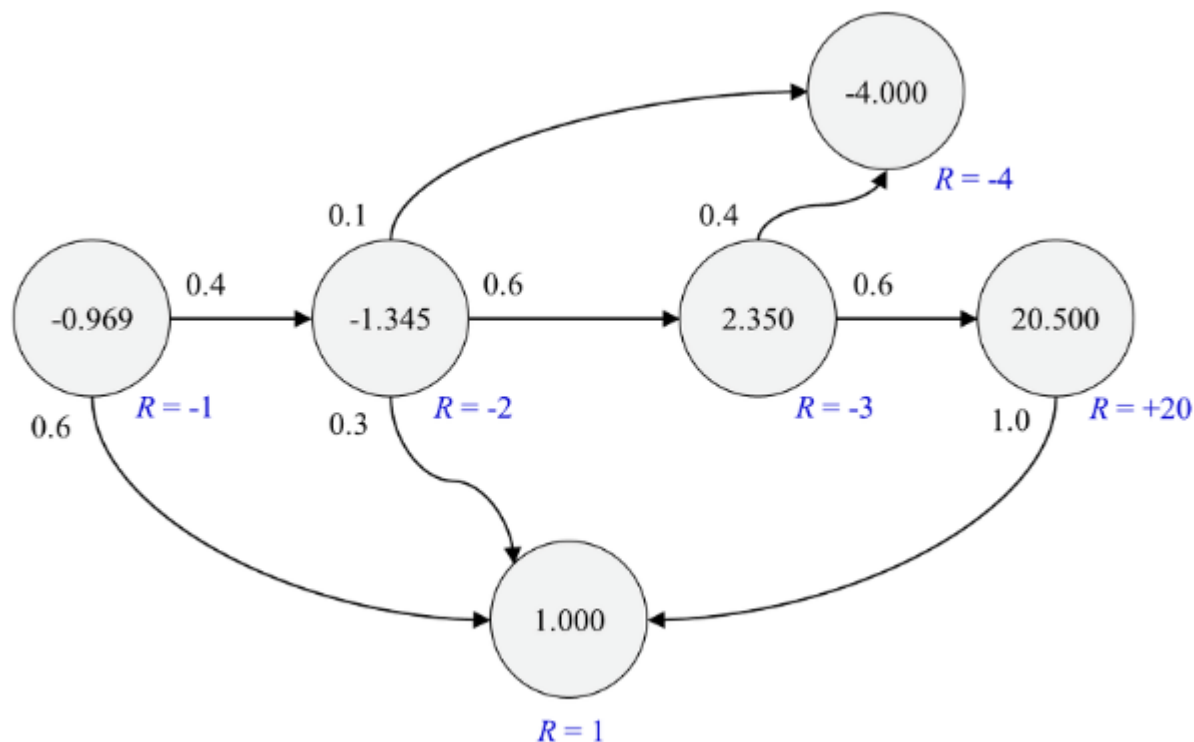
$$v(s) = \mathbb{E}[G_t | S_t = s]$$

$$v(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] = E[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]$$

$$v(s) = R_{t+1} + \gamma \sum_{s' \in S} p(s' | s) v(s')$$

# Markov Decision Process (MDP)

## ❖ 탐험가 MDP에서의 각 state의 State-value function

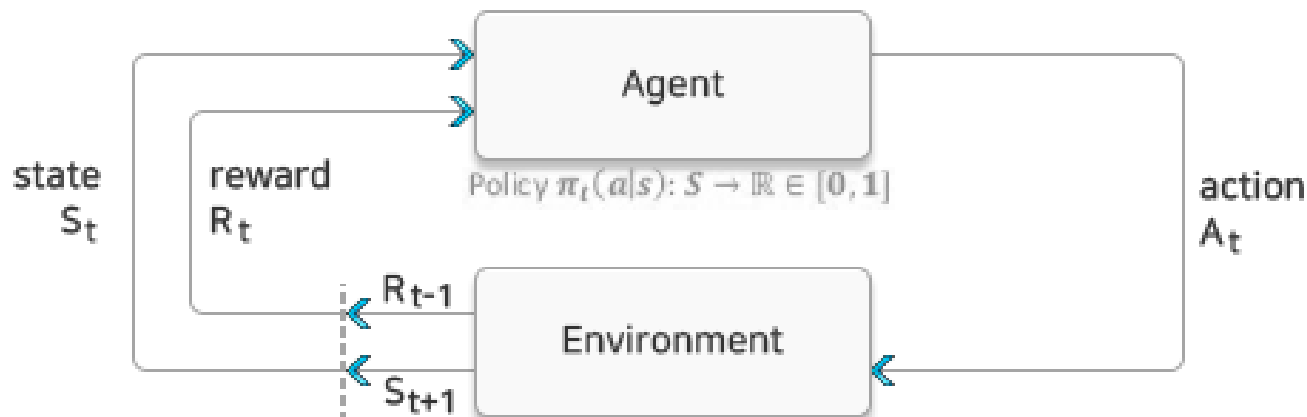




# Markov Decision Process (MDP)

## ❖ 마르코프 결정 과정(Markov Decision Process ,MDP)

- 전이학습 모델
- 상태 전이(state transition)가 현재 상태  $S_t$ 와 행동  $A_t$ 에 의해서 확률적으로 결정되는 마르코프 모델(Markov model)
- Markov reward process에 action이라는 요소가 추가된 모델  $\langle S, A, P, R, \gamma \rangle$



## ❖ Reward의 합의 기댓값을 최대로 만드는 policy를 찾는 문제

$$\max_{\pi} \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t | \pi]$$

# Markov Decision Process (MDP)

❖ 강화학습 목적 : MDP로 정의된 문제에 대해 각 state마다 전체적인 reward를 최대화하는 action이 무엇인지를 결정하는 것

❖ 정책(Policy)  $\pi$

- 각각의 state마다 action이 선택될 확률을 표현하는 함수

$$\pi(a|s) = \Pr(A_t=a|S_t=s)$$

❖ 정책을 포함한 상태가치함수(State-value function with policy)

- 어떠한 state가 더 많은 reward를 얻을 수 있는지를 알려준다

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

# Markov Decision Process (MDP)

## ❖ 상태-행동 가치함수(Action-value function)

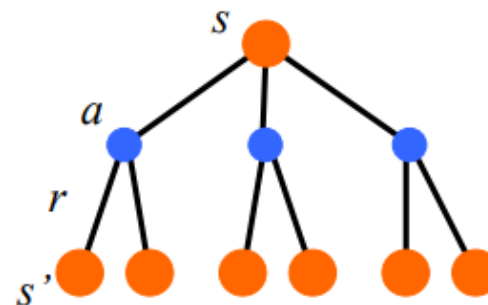
- state  $s$ 에서 시작하여  $a$ 라는 action을 취했을 때 얻을 수 있는 누적보상치의 기댓값
- 어떠한 state에서 어떠한 action을 취해야 더 많은 reward를 얻을 수 있는지 알려준다

$$\begin{aligned} q_{\pi}(s,a) &= E_{\pi}[G_t | S_t=s, A_t=a] = E_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t=s] \\ &= r(s,a) + \gamma \sum_{s' \in S} p(s' | s,a) \sum_{a' \in A} \pi(a' | s') q_{\pi}(a' | s') \end{aligned}$$

# Markov Decision Process (MDP)

## ❖ 상태 가치 함수와 상태-행동 가치 함수의 관계

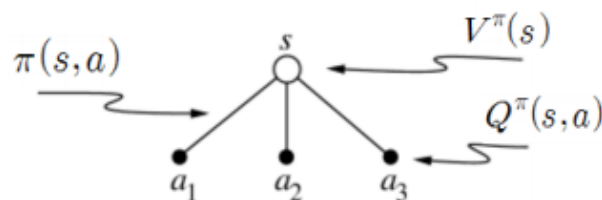
$$\begin{aligned}
 V^\pi(s) &= \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi\right] \\
 &= \mathbb{E}\left[r_{t+1} + \gamma \sum_{s'} P_{ss'}^a \left[r_{ss'}^a + \gamma \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right]\right]\right] \\
 &= \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a \left[r_{ss'}^a + \gamma \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_{t+1} = s'\right]\right] \\
 &= \sum_a \pi(s, a) \left[ \sum_{s'} P_{ss'}^a \left[r_{ss'}^a + \gamma V^\pi(s')\right] \right] \\
 &= \sum_a \pi(s, a) Q^\pi(s, a)
 \end{aligned}$$



- $\pi(s, a)$ : 정책  $\pi$ 가 상태  $s$ 에서 행동  $a$ 를 선택할 확률
- $P_{ss'}^a$ : 상태  $s$ 에서 행동  $a$ 를 할 때, 상태  $s'$ 이 될 확률
- $r_{ss'}^a$ : 상태  $s$ 에서 행동  $a$ 를 할 때, 보상값
- $\gamma$ : 할인율

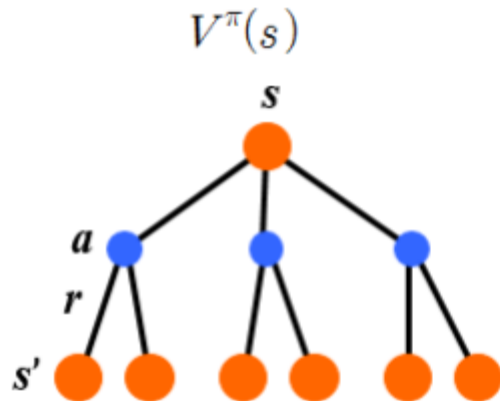
$$V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a)$$

$$Q^\pi(s, a) = \sum_{s'} P_{ss'}^a \left[ r_{ss'}^a + \gamma V^\pi(s') \right]$$

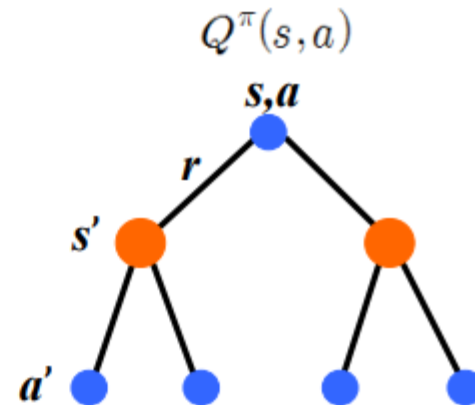


# Markov Decision Process (MDP)

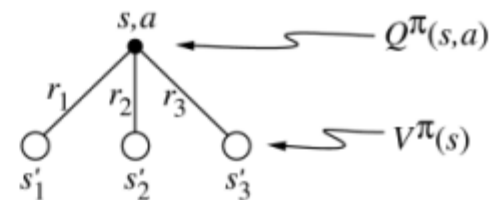
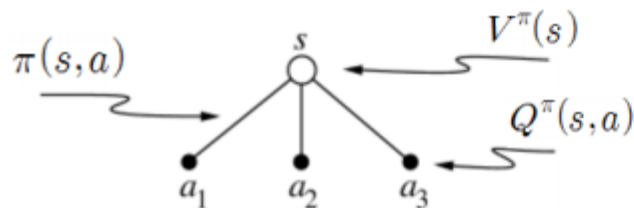
## ❖ 상태 가치 함수와 상태-행동 가치 함수의 관계



$$V^{\pi}(s) = \sum_a \pi(s,a) Q^{\pi}(s,a)$$



$$Q^{\pi}(s,a) = \sum_{s'} P_{ss'}^a [r_{ss'}^a + \gamma V^{\pi}(s')]$$



# Markov Decision Process (MDP)

## ❖ 최적의 상태가치함수(optimal state-value function)

- 주어진 모든 policy에 대한 state-value function의 최대값

$$V^*(s) = \max_{\pi} V_{\pi}(s)$$

## ❖ 최적의 상태-행동 가치 함수(optimal action-value function )

- 주어진 모든 policy에 대한 action-value function의 최대값

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a)$$

## ➔ 최적의 정책(optimal policy)

$$\pi^*(a|s) = \begin{cases} 1, & a = \arg \max_{a'} Q^*(s, a') \\ 0, & \text{otherwise} \end{cases}$$

# Markov Decision Process (MDP)

## ❖ 벨만 최적 방정식(Bellman Optimality Equation)

- 에이전트가 상태  $s$ 에 도달한 후 최적으로 행동한다는 가정하에 평균적으로 기대할 수 있는 할인된 미래의 보상의 합

$$V^*(s) = \max_a \sum_{s'} [r(s, a, s') + \gamma V^*(s')] P(s'|s, a)$$

$$Q^*(s, a) = \sum_{s'} [r(s, a, s') + \gamma \max_{a'} Q^*(s', a')] P(s'|s, a)$$

$$\pi^*(a|s) = \begin{cases} 1, & a = \arg \max_{a'} Q^*(s, a') \\ 0, & \text{otherwise} \end{cases}$$

Bellman Optimality Equation

$P(s'|s, a)$  에이전트가 행동  $a$ 를 선택했을 때, 상태  $s$ 에서 상태  $s'$ 으로 전이될 확률

$r(s, a, s')$  에이전트가 행동  $a$ 를 선택해서 상태  $s$ 에서 상태  $s'$ 으로 이동되었을 때 에이전트가 받을 수 있는 보상

# Markov Decision Process (MDP)

## ❖ 가치반복(Value Iteration) 알고리즘

- 모든 상태가치를 반복적으로 수정하여 최적의 상태가치를 추정

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} [r(s, a, s') + \gamma V_k(s')] P(s'|s, a)$$

## ❖ Q-가치반복(Q-Value Iteration) 알고리즘

- 모든 Q-가치를 반복적으로 수정하여 최적의 Q-가치를 추정

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \cdot \max_{a'} Q_k(s', a') \right] \quad \text{for all } (s, a)$$

## ❖ 최적의 Q-가치를 이용하여 최적의 정책 정의

- 에이전트는 가장 높은 Q값을 가진 행동을 선택하는 것이 가장 좋은 전략

$$\pi_{k+1}(a|s) = \begin{cases} 1, & a = \arg \max_{a'} Q_k(s, a') \\ 0, & \text{otherwise} \end{cases}$$



# Q-Learning

## ❖ Model-based & Model-free

- Model-based : 가치반복(Value Iteration) 알고리즘은 정확한 MDP모델이 필요하나 실제 상황에서는 정확한 모델을 모름
- Model-free : Q-Learning은 모델이 없이 학습하는 강화학습 알고리즘

## ❖ Q-Learning

- 최적의 상태(s)-행동(a) 가치를 추정하여 최적의 정책을 정의
- 가치함수 갱신을 정책이 아닌 max 연산자로 결정
- 에이전트가 상태 s에 도달해서 행동 a를 선택한 후 행동의 결과를 얻기 전에 평균적으로 기대할 수 있는 할인된 미래 보상의 합

$$\hat{Q}(s, a) \leftarrow r(s, a) + \gamma \max_{a'} \hat{Q}(s', a')$$

# Q-Learning

## ❖ Learning rate( $\alpha$ )를 이용하여 점진적으로 학습

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')]$$

### Q-learning algorithm

For each  $s, a$  initialize table entry  $\hat{Q}(s, a) \leftarrow 0$

Observe current state  $s$

Do forever:

- Select an action  $a$  and execute it
- Receive immediate reward  $r$
- Observe the new state  $s'$
- Update the table entry for  $\hat{Q}(s, a)$  as follows:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')]$$

- $s \leftarrow s'$

*Machine Learning*, T. Mitchell, McGraw Hill, 1997

# Q-Learning

---

## Exploit VS Exploration: E-greedy

$e = 0.1$

if  $\text{rand} < e$ :

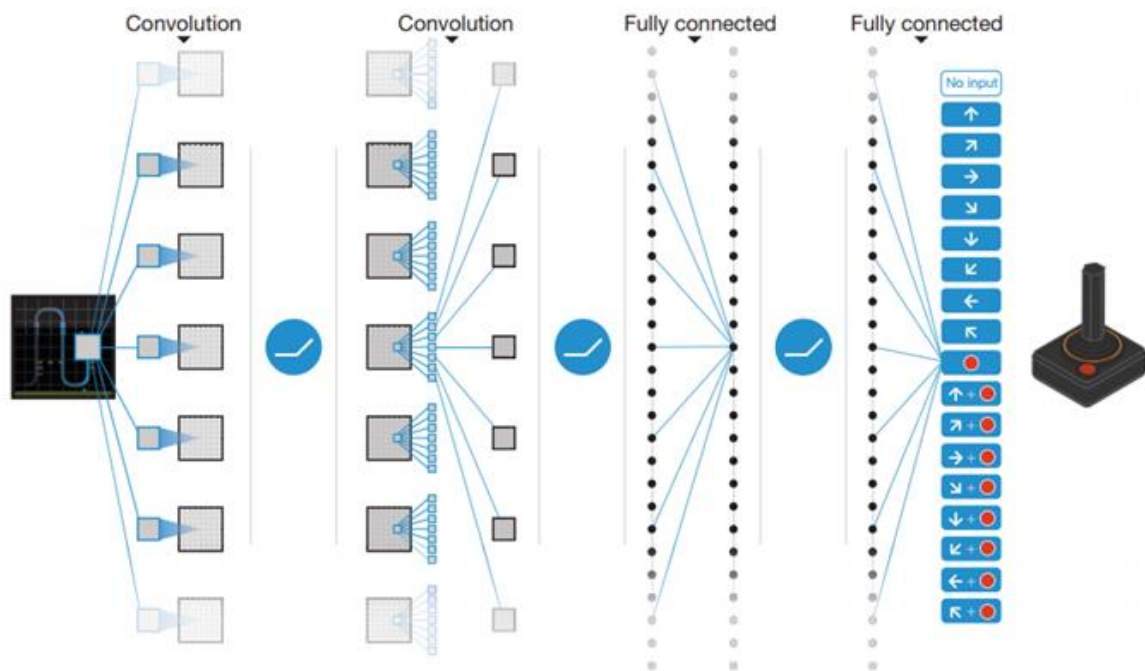
$a = \text{random}$

else:

$a = \text{argmax}(Q(s, a))$

# Deep Q-Learning(DQN)

- ❖ “Playing Atari with Deep Reinforcement Learning” 논문 에서 발표 (Google Deepmind)
- ❖ atari video game에 적용
  - NN과 Q-Learning을 결합한 DQN으로 상태를 입력하고, 게임 종류마다 필요한 만큼 출력 노드를 결정



# Deep Q-Learning(DQN)

- ❖ 상태-행동 쌍의 Q-value에 근사 하는 함수의 가중치를 찾는 문제
- ❖ Q-value 추정을 위해 DNN 사용 (deep Q-network, DQN)
  - 미래의 할인된 가치를 추정하기 위해 다음 상태  $s'$  과 모든 사용 가능한 행동  $a'$  에 대해 DQN실행으로 모든 가능한 행동에 대한 미래 근사 Q-value를 얻을 수 있다
  - 미래 근사 Q-value 에서 가장 큰 값을 골라 할인을 적용하면 할인된 미래보상 추정을 얻을 수 있다
  - 할인된 미래보상 추정과 보상  $r$ 을 더하여 상태-행동쌍( $s,a$ )에 대한 목표 Q-value  $y(s,a)$ 를 얻게 된다.
  - 목표 Q-value로 손실을 최소화할 수 있도록 경사하강법을 사용해 훈련단계를 수행

$$Q_{\text{target}}(s, a) = r + \gamma \cdot \max_{a'} Q_{\theta}(s', a')$$

# Deep Q-Learning(DQN)

---

## ❖ 예제파일

- Lab16\_2\_reinforcement\_learning.ipynb