

# Reinforcement learning

---

Geonhee Lee  
gunhee6392@gmail.com

## Outline

- Introduction to Reinforcement learning
- Markov Decision Process(MDP)
- Dynamic Programming(DP)
- Monte Carlo Method(MC)
- Temporal Difference Method(TD)
  - SARSA
  - Q-Learning
- Planning and Learning with Tabular Methods
- On-policy Control with with Approximation
- On-policy Prediction with Approximation
- Policy Gradient Method
- Actor Critic Method

# Introduction to Reinforcement learn

## RL 특성

다른 ML paradigms과의 차이점

- No supervisor, 오직 reward signal.
- Feedback이 즉각적이지 않고 delay 된다.
- Time이 큰 문제가 된다(연속적인, Independent and Identically Distributed(i.i.d, 독립항등분포) data가 아니다).
- Agent의 행동이 agent가 수용하는 연속적인 data에 영향을 준다.

## Reward

- **Reward**: scalar feedback signal.
- agent가 step t에서 얼마나 잘 수행하는 지 나타냄.
- agent의 목표는 전체 reward의 합을 최대화하는 것

## Sequential Decision Making

- Goal: Total future reward를 최대화하는 action 선택.
- Action들은 long term 결과들을 가질 것.
- Reward는 지연될 것.
- long-term reward를 더 크게 얻기 위해 즉각적인 reward를 희생하는 것이 나올 수도 있음.

## History and State

- history: observations, actions, rewards의 연속.
- State: 다음에 어떤 일이 일어날 것인지 결정하기 위해 사용된 정보(다음 수식을 위한 정의로 보임)
- 공식으로는, state는 history의 함수이다.

$$S_t = f(H_t)$$

## Information State

- Information state(a.k.a. Markov state)는 history로부터 모든 유용한 정보를 포함한다.
- 정보이론 관점에서의 information state 혹은 Markov state라는 상태가 있다. 데이터 관점에서 history의 유용한 정보들을 포함하고 있는 state를 의미한다.

### Definition

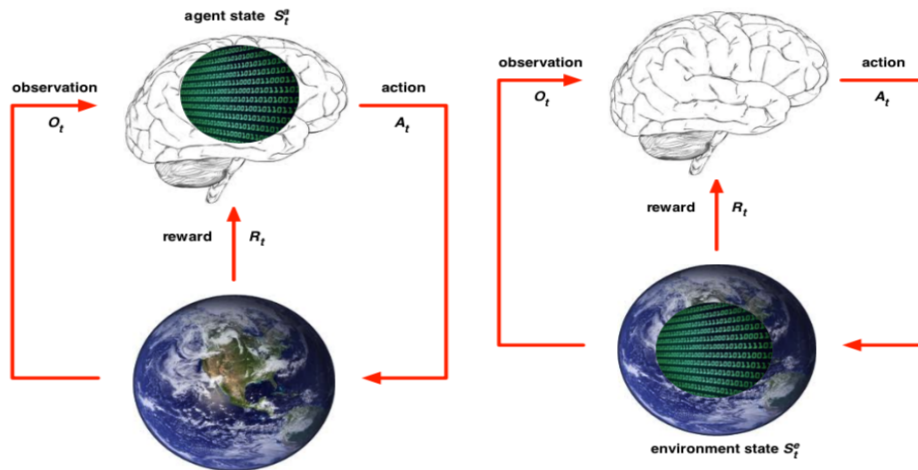
state  $S_t$ 는 Markov 이다 if and only if  $P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_t]$

- 미래는 현재의 과거와 독립적이다.
- State가 주어지면, history는 버려질 수 있다.

## Fully Observable Environments

- **Full observability:** agent는 직접적으로 enviroment state를 관찰한다.

$$O_t = S_t^a = S_t^e$$



- Agent state = environment state = information state.
- 형식적으로, 이것은 Markov decision precess(MDP).

## Partially Observable Environments

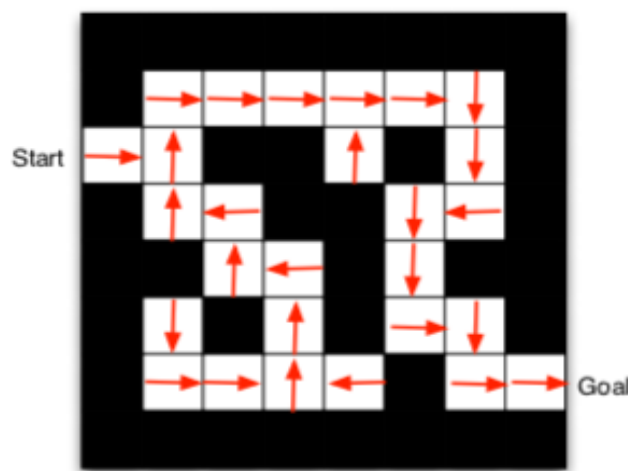
- Partial observability: agent는 간접적으로 environment를 관찰.
  - (ex)robot이 카메라를 가지고 절대적인 위치를 알지못하는 것.
  - (ex)포커를 하는 agent는 오직 오픈한 card들만 볼 수 있는 것.
- 여기서, agent state  $\neq$  environment state.
- 형식적으로, 이것을 partially observable Markob decision process(POMDP).
- Agent는 자체 state representation  $S_t^a$ 을 구성해야만 한다.
  - 다음과 같은 방법으로 만들 수 있다(1. 전체 history 사용, 2. 확률을 사용, 3. RNN 방식 사용).
    - Complete history:  $S_t^a = H_t$ .
    - **Beliefs** of environment state:  $S_t^a = \mathbb{P}[S_t^e = s^1], \dots, \mathbb{P}[S_t^e = s^n]$ .
    - Recurrent neural network:  $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$ .

## RL Agent의 주요 성분

- **Policy:** agent의 행동 함수.
- **Value function:** 각 state 및/혹은 action이 얼마나 좋은지.
- **Model:** agent's representation of the environment.

### Policy

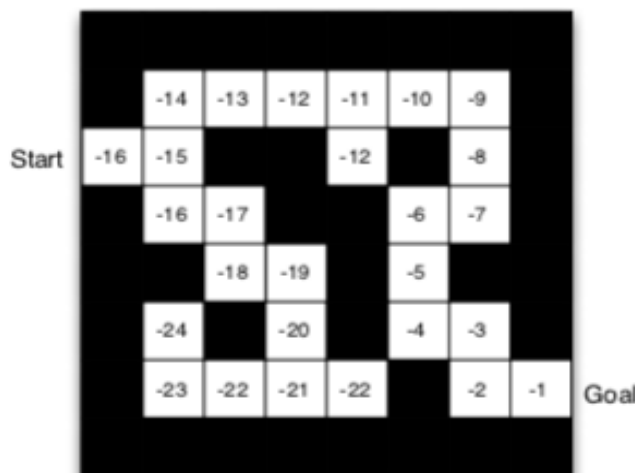
- **Policy:** Agent의 행동.
- State에서 action으로 매핑.
  - Deterministic policy:  $a = \pi(s)$ .
  - Stochastic policy:  $\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$ .



### Value function

- **Value function:** Future reward 예측 값.
- State의 좋은것/나쁜것인지 판단하기 위해 사용.
- Value function을 이용하여 action 선택

$$V_{\pi} = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

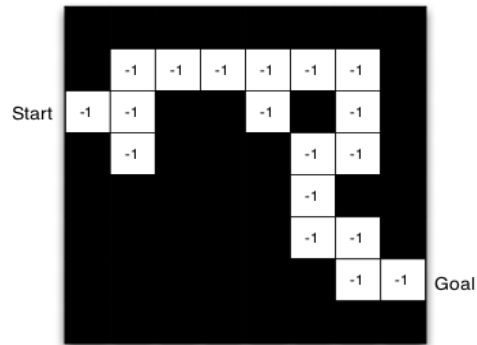


## Model

- **Model:** environment에서 다음에 행해질게 무엇인지 예측.
- $P$ : 다음 state를 예측.
- $R$ : 다음(즉각적인) reward를 예측.

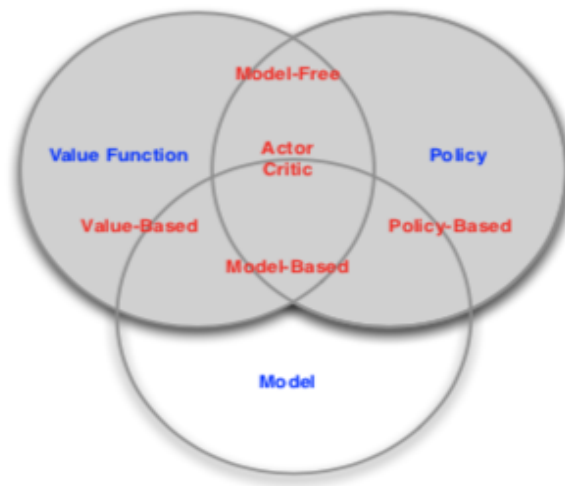
$$P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

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



- Agent는 env의 내부 모델을 가지고 있다고 가정.
  - Dynamics: action들이 state를 변화시키는 방법.
  - Rewards: 각 state으로부터 얼마의 reward를 받는 지.
  - Model은 불완전할 것.
- Grid layout은 transition model ( $P_{ss'}^a$ )를 나타낸다.
- 숫자들은 (모든 행동에 동일한) 각 state s로부터 즉각적인 reward ( $R_s^a$ )를 나타낸다.

## RL Agent 분류



## Learnign and Planning

Sequential decision making에서 두 가지 근본적인 문제

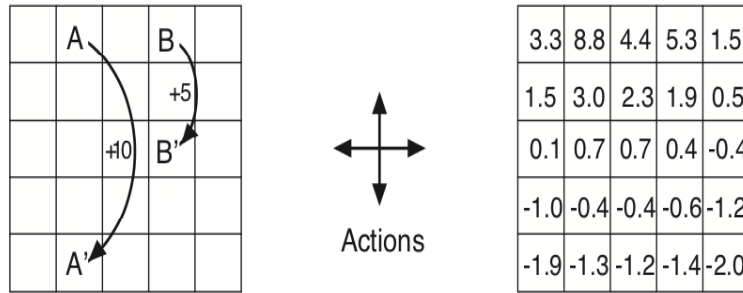
- Reinforcement Learning:
  - Env는 초기에 알려져있지 않음.
  - Agent는 Env와 상호작용.
  - Agent는 policy를 향상시킴.
- Planning:
  - Env 모델은 알려져 있음.
  - Agent는 (어떠한 외부 상호작용 없이) 모델과 계산을 수행.
  - Agent는 policy를 향상시킴.
  - a.k.a. deliberation, reasoning, introspection, pondering, thought, search.

## Exploration and Exploitation

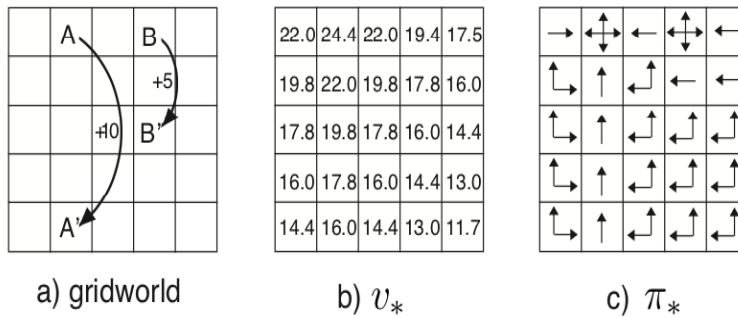
- RL은 trial-and-error learning과 유사.
- Agent는 good policy를 발견해야만 한다.
  - Env의 경험으로부터
  - 도중에 많은 reward를 잃지 않도록
- **Exploration**은 Env에 대한 더 많은 정보를 찾는다.
- **Exploitation**은 reward를 최대화하기 위해 알려진 정보를 exploit.
- Exploit만큼 explore도 일반적으로 중요하다.

## Prediction and Control

- **Prediction:** future를 평가.
  - 주어진 policy를 이용하여 계산 및 평가.
    - (아래그림)Uniform random policy의 value function은 무엇인가?



- **Control:** future를 최적화.
  - best policy를 찾는 것.
    - (아래그림)모든 가능한 정책들에서 optimal value function은 무엇인가?
    - (아래그림)Optimal policy는 무엇인가?



---

# Markov Decision Process(MDP)

---

## Outline

- Markov Processes
- Markov Reward Processes
- Markov Decision Processes
- Extensions to MDPs



## Introduction to MDPs

- Markov decision processes(MDP)는 RL에서 Env를 형식적으로 기술.
  - 여기서 Env는 fully observable.
  - i.e., 현재 state는 완전하게 process의 특성을 나타냄.
- 대부분 모든 RL 문제들은 MDPs 로 공식화될 수 있다.
  - Optimal control은 주로 continuous MDPs를 다룬다.
  - Partially Observable problem은 MDPs로 변환을 할 수 있다.
  - Bandits은 하나의 state를 가진 MDPs이다.

## Markov Property

"미래는 현재에서의 과거와 독립적이다."

### Definition

state  $S_t$ 는 *Markov* if and only if

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

- State는 history로부터 모든 관련정보를 수집한다.
- State가 알려졌다면, history는 버릴 수 있다.
  - i.e. State는 미래의 sufficient statistic.

## State Transition Matrix

Markov state  $s$  및 successor state  $s'$ 에 대하여, **state transition probability** 는 다음과 같이 정의된다.

$$P_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s]$$

**State transition matrix**  $P$ 는 모든 state  $s$ 에서 모든 successor  $s'$ 로의 transition probabilities를 다음과 같이 정의한다.

$$\text{to} \quad P = \text{from} \begin{bmatrix} P_{11} & \dots & P_{1n} & \vdots & \ddots & \vdots & P_{n1} & \dots & P_{nn} \end{bmatrix}$$

여기서 각 행렬의 행의 합은 1이다.

## Markov Process

**Markov process** 는 memoryless random process, i.e. Markov property를 가진 random states  $S_1, S_2, \dots$  의 sequence.

#### Definition

A *Markov Process* (or *Markov Chain*) is a tuple  $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$  is a (finite) set of states
- $\mathcal{P}$  is a state transition probability matrix,  
 $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$

## Markov Reward Process

**Markov reward process** 는 value를 가진 Markov chain(Markov Process)

#### Definition

A *Markov Reward Process* is a tuple  $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$  is a finite set of states
- $\mathcal{P}$  is a state transition probability matrix,  
 $\mathcal{P}_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s]$
- $\mathcal{R}$  is a reward function,  $\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$
- $\gamma$  is a discount factor,  $\gamma \in [0, 1]$

## Return

#### Definition

The *return*  $G_t$  is the total discounted reward from time-step  $t$ .

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

- Discount  $\gamma \in [0, 1]$ 은 미래 보상들의 현재 값.
- $k+1$  time-step 이후에 받게되는 reward의 값은  $\gamma^k R$ .
- 지연되는 reward는 즉각적인 reward를 중요시한다.
  - $\gamma$ 가 0에 가까우면, "myopic(근시안적)" 평가를 도출.
  - $\gamma$ 가 1에 가까우면, "far-sighted(미래를 내다보는)" 평가를 도출.

## Discount

대부분 **Markov reward** 및 **decision process** 는 discount된다. 왜?

- 수학적으로 discount reward에 대해 편리하다.
- Cyclic Markov process에서 infinite return을 피한다.
- 미래에 대한 uncertainty는 fully representation 하지 않아도 된다.
- 동물/인간의 행동은 즉각적인 보상에 대해 선호하는 것을 볼 수 있다.

## Value Function

**Value function**  $v(s)$ 는 state  $s$ 의 long-term 값을 제공한다.

### Definition

The state value function  $v(s)$  of an MRP is the expected return starting from state  $s$

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

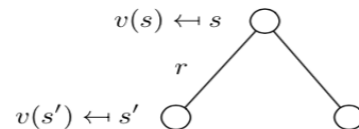
## Bellman Equation for MRPs

Value function은 두 개의 part로 분리할 수 있다.

- 즉각적인 reward  $R_{t+1}$
- Successor state  $\gamma v_{(S_{t+1})}$ 의 discounted value.

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

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



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

## Solving the Bellman Equation

- Bellman Equation은 linear equation.
- 이 식은 다음과 같이 직접 풀 수 있다:

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

- Computational complexity는  $n$  state에 대해  $O(n^3)$ .
- **Small MRPs**에 대해서만 직접 풀 수 있다.
- **Large MRPs**에 대해 여러 iterative method가 있다.

- Dynamic programming(DP)
- Monte-Carlo evaluation(MC)
- Temporal-Difference learning(TD)

## Markov Decision Process

**Markov decision process(MDP)** 는 **decision** 을 가진 Markov reward process이다. 모든 state들이 Markov인 Environment이다.

**Definition**

A Markov Decision Process is a tuple  $\langle S, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $S$  is a finite set of states
- $\mathcal{A}$  is a finite set of actions
- $\mathcal{P}$  is a state transition probability matrix,  
 $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$  is a reward function,  $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- $\gamma$  is a discount factor  $\gamma \in [0, 1]$ .

## Policies

**Definition**

A policy  $\pi$  is a distribution over actions given states,

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

- Policy는 agent의 behavior를 완벽히 정의.
- MDP policy는 현재 state에 의존(not the history).
- i.e. Policy는 stationary(time-independent).  $A_t \sim \pi(\cdot | S_t), \forall t > 0$
- MDP  $M = \langle S, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$  및 policy  $\pi$  주어지고
  - State sequence  $(S_1, S_2, \dots)$ 는 Markov process  $\langle S, P^\pi \rangle$
  - State 및 reward sequence  $(S_1, R_2, S_2, \dots)$ 는 Markov reward process  $\langle S, P^\pi R^\pi, \gamma \rangle$
  - 여기서,

$$P_{s,s'}^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) P_{ss'}^a$$

$$R_s^\pi = \sum_{a \in \mathcal{A}} \pi(a|s) R_s^a$$

## Value Function

**Definition**

The *state-value function*  $v_{\pi}(s)$  of an MDP is the expected return starting from state  $s$ , and then following policy  $\pi$

$$v_{\pi}(s) = \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

**Definition**

The *action-value function*  $q_{\pi}(s, a)$  is the expected return starting from state  $s$ , taking action  $a$ , and then following policy  $\pi$

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a]$$

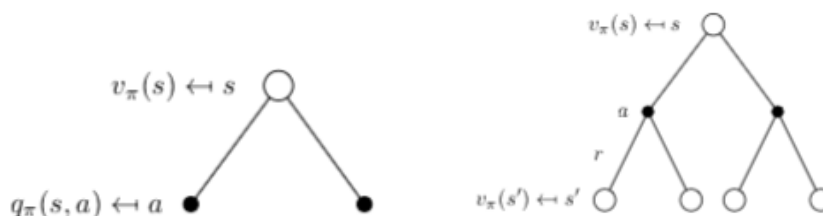
## Bellman Expectation Equation

**State-value( $V$ ) function**은 즉각적인 reward와 successor state의 discounted value의 합으로 다시 분해할 수 있다.

$$v_{\pi}(s) = \mathbb{E}_{\pi} [R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s]$$

**Action-value( $Q$ ) function** 은 유사하게 분해할 수 있다.

$$q_{\pi}(s, a) = \mathbb{E}_{\pi} [R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a]$$

Bellman Expectation Equation for  $V^{\pi}$ 

$$v_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_{\pi}(s, a) \quad v_{\pi}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left( R_s^a + \gamma \sum_{s' \in \mathcal{S}} P_{ss'}^a v_{\pi}(s') \right)$$

Bellman Expectation Equation for  $Q^{\pi}$



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

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

## Optimal 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

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

- Optimal value function은 MDP에서 best possible performance를 명시한다.
- MDP는 optimal value function을 안다면 풀 수 있다.

## Optimal Policy

Policy의 부분 순서를 정의

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

### Theorem

For any Markov Decision Process

- There exists an optimal policy  $\pi_*$  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,  $q_{\pi_*}(s, a) = q_*(s, a)$

## Finding an Optimal policy

Optimal policy는  $q_*(s, a)$ 를 최대화하여 찾을 수 있다.

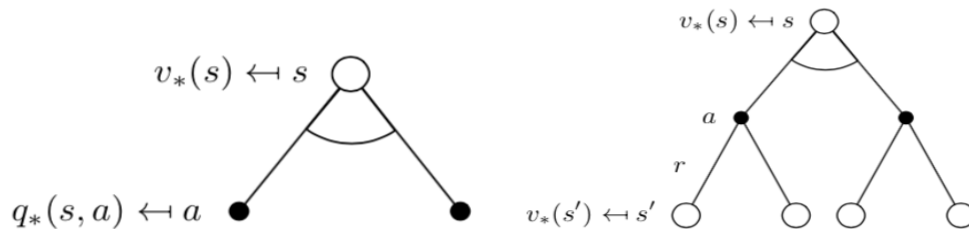
$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in A}{\operatorname{argmax}} q_*(s, a) \\ 0 & \text{otherwise} \end{cases}$$

- 어떤 MDP에 대해서도 항상 deterministic optimal policy.
- 만약  $q_*(s, a)$ 를 알고 있다면, 즉시 optimal policy를 가질 수 있다.

## Bellman Optimality Equation

### Bellman Optimality Equation for $V^*$

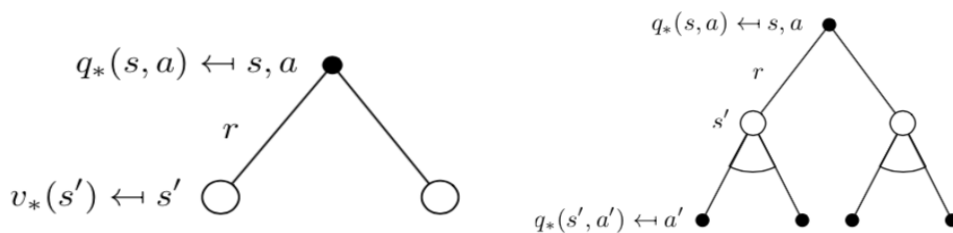
Optimal value function은 재귀적으로 bellman optimality equation과 관련 있다.



$$v_*(s) = \max_a q_*(s, a)$$

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

### Bellman Optimality Equation for $Q^*$



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

## Solving the Bellman Optimality Equation

- Bellman Optimality Equation은 non-linear.
- (일반적으로) closed form solution은 없다.
- 많은 iterative solution 방법:
  - Value Iteration.
  - Policy Iteration.
  - Q-learning.
  - Sarse.

## Extensions to MDPs

- Infinite and continous MSPs



- Partially observable MDPs
- Undiscounted, average reward MDPs

## Infinite MDPs

다음 extension은 모두 이용가능하다:

- Countably infinite state and/or action spaces
  - Straightforward(명확, 간단한)
- Continuous state and/or action spaces
  - Closed form for linear quadratic model(LQR)
- Continuous time
  - Requires partial differential equations
  - Hamilton-Jacobi-Bellman (HJB) equation
  - Limiting case of Bellman equation as time-step  $\rightarrow 0$

## Partially Observable Markov Decision Process(POMDPs)

POMDPs: Hidden state들을 가진 MDP. action을 가진 hidden Markov model.

**Definition**

A POMDP is a tuple  $\langle S, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$

- $S$  is a finite set of states
- $\mathcal{A}$  is a finite set of actions
- $\mathcal{O}$  is a finite set of observations
- $\mathcal{P}$  is a state transition probability matrix,  
 $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$  is a reward function,  $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- $\mathcal{Z}$  is an observation function,  
 $\mathcal{Z}_{s'o}^a = \mathbb{P}[O_{t+1} = o \mid S_{t+1} = s', A_t = a]$
- $\gamma$  is a discount factor  $\gamma \in [0, 1]$ .

## Belief States

- History: action, observation, reward의 sequence.
- Belief state: history 조건에서의 state의 확률 분포.

**Definition**

A history  $H_t$  is a sequence of actions, observations and rewards,

$$H_t = A_0, O_1, R_1, \dots, A_{t-1}, O_t, R_t$$

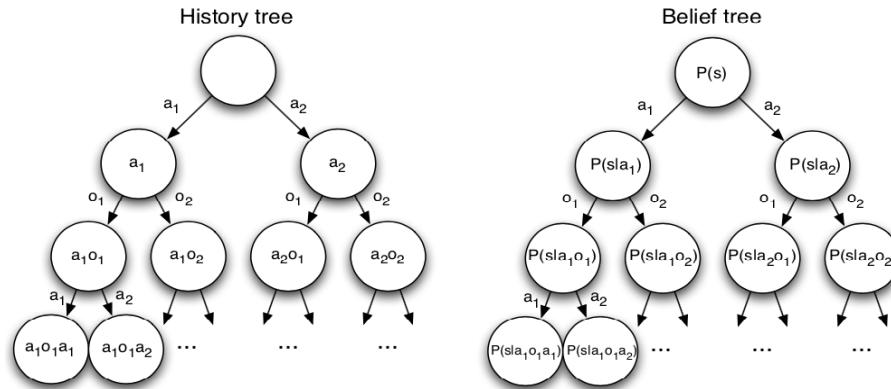
**Definition**

A belief state  $b(h)$  is a probability distribution over states, conditioned on the history  $h$

$$b(h) = (\mathbb{P}[S_t = s^1 \mid H_t = h], \dots, \mathbb{P}[S_t = s^n \mid H_t = h])$$

## Reductions of POMDPs

- History  $H_t$ 는 Markov property 만족.
- Belief state  $b(H_t)$ 는 Markov property 만족.



- POMDP는 (infinite) history tree로 축소될 수 있음.
- POMDP는 (infinite) belief state로 축소될 수 있음.

## Ergodic Markov Process

- Ergodic Markov process는
  - Recurrent: 각 state는 무한히 방문됨
  - Aperiodic: 각 state는 systematic 주기없이 방문됨.
- Ergodic Markov process는 제한된 고정 분포  $d^\pi(s)$ 를 가짐.
- 정의:
  - 어떠한 policy에 의해 유도된 Markov chain이 ergodic이라면 MDP는 ergodic.
- 어떠한 policy  $\pi$  에 대해서, ergodic MDP는 start state의 독립적인 time-step  $\rho^\pi$  분의(per) average reward를 가짐.

### Theorem

An ergodic Markov process has a limiting stationary distribution  $d^\pi(s)$  with the property

$$d^\pi(s) = \sum_{s' \in S} d^\pi(s') P_{s's}$$

### Definition

An MDP is ergodic if the Markov chain induced by any policy is ergodic.

For any policy  $\pi$ , an ergodic MDP has an average reward per time-step  $\rho^\pi$  that is independent of start state.

$$\rho^\pi = \lim_{T \rightarrow \infty} \frac{1}{T} \mathbb{E} \left[ \sum_{t=1}^T R_t \right]$$

---

## Average Reward Value Function

- Undiscounted의 value function, ergodic MDP는 average reward의 관점으로 표현될 수 있다.
- $\tilde{v}_\pi(s)$ 는 state  $s$  에서 시작하기 때문에 extra reward.

$$\tilde{v}_\pi(s) = \mathbb{E}_\pi \left[ \sum_{k=1}^{\infty} (R_{t+k} - \rho^\pi) \mid S_t = s \right]$$

이것은 average reward Bellman equation과 상응한다.

$$\begin{aligned} \tilde{v}_\pi(s) &= \mathbb{E}_\pi \left[ (R_{t+1} - \rho^\pi) + \sum_{k=1}^{\infty} (R_{t+k+1} - \rho^\pi) \mid S_t = s \right] \\ &= \mathbb{E}_\pi [(R_{t+1} - \rho^\pi) + \tilde{v}_\pi(S_{t+1}) \mid S_t = s] \end{aligned}$$

---

## Dynamic Programming(DP)

---

---

# Monte Carlo Method(MC)

---

---

# Temporal Difference Method(TD)

---

---

# Planning and Learning with Tabular Methods

---

---

# On-policy Control with with Approximation

---

---

# Policy Gradient Method

---



---

# Actor Critic Method

---

# Reference

---

- [1] [UCL Course on RL](#)
- [2] [Reinforcement Learning: Tutorial](#)(Seoul National University of Science and Technology)
- [3] [Reinforcement Learning : An Introduction](#), Sutton
- [4] [jay.tech.blog](#)
- [5] [대손의 스마트 웹](#)