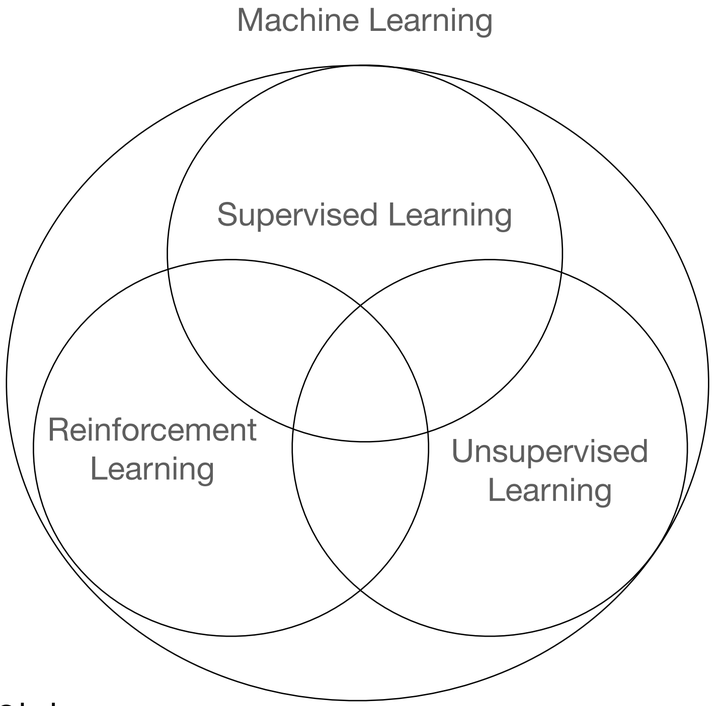
Reinforcement Learning introduction #1

정규헌

Reinforcement Learning
강화학습의 특징

- **No supervisor (Self-learning)**
  - 정답이 없고, 오직 **reward signal/state**만 존재
  - 알파고가 사람을 뛰어넘는 것이 가능했던 이유
- **Time really matters (sequential)**
- **Feedback is delayed, not instantaneous**
  - 행동 후에 보상이 뒤늦게 나타나는 경우가 존재
  - 피드백이 지연될 수 있음
- **Reinforcement learning is based on the reward hypothesis**
  - **Reward Hypothesis** : 모든 목적은 **cumulative reward**를 극대화하는 것으로 정의할 수 있다.
  - 모든 문제에서 강화학습이 적합할 수 없음



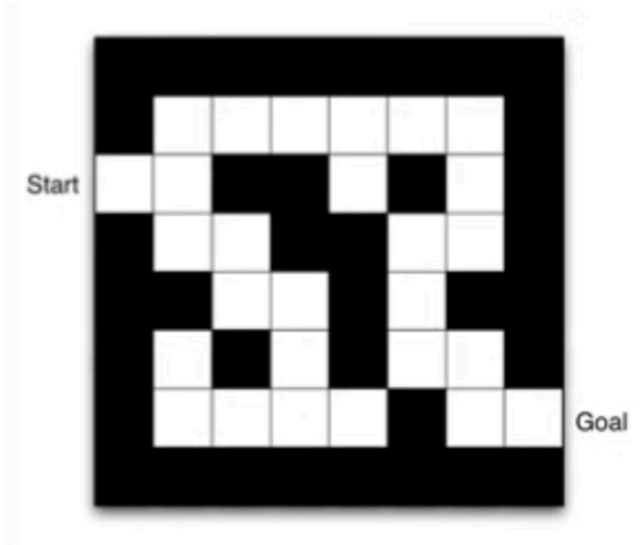
Reinforcement Learning
강화학습 문제

- **강화학습 문제**
  - 주어진 상황이 있을 때, 가장 최적의 행동은?
  - **Agent**가 환경과의 상호작용(시행착오)을 통해 목표를 달성하는 방법을 배우는 문제
  - **순차적 의사결정 문제(Sequential decision making)**
    - 행동을 하면, 그로 인해 상황이 바뀌고, 다시 어떤 행동을 하고..
    - 연속적인 행동을 잘 선택해야 하는 문제
    - **Agent**의 **action**에 따라 뒤에 받게될 데이터가 달라짐

RL Base Knowledge

Term

- Reinforcement Learning
- Reward
- Agent
  - Policy
  - Value function
- Environment
- state

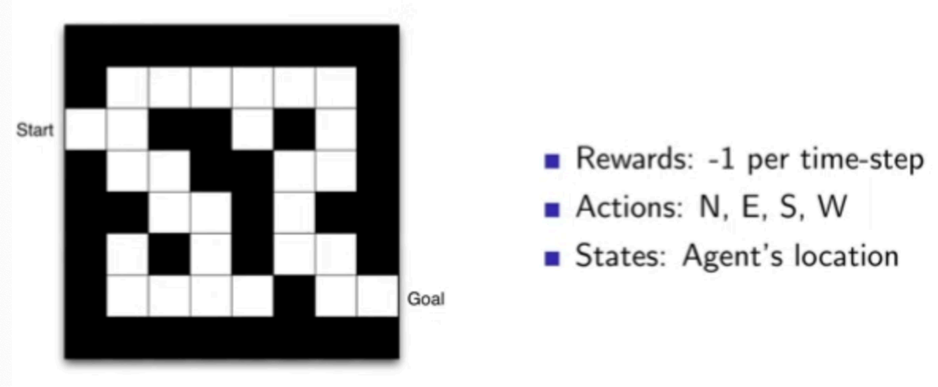


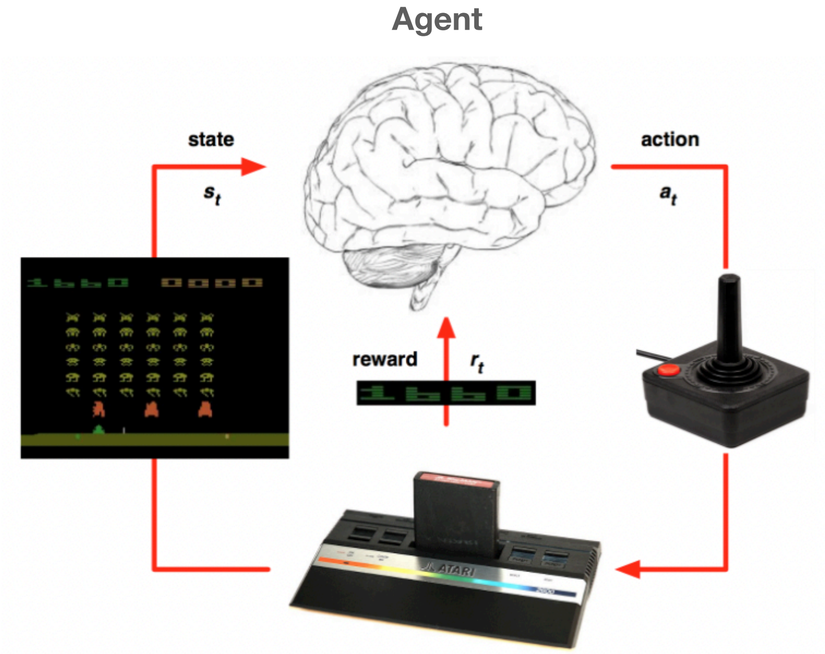
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

RL Base Knowledge

Reward

- RL은 **Reward** Hypothesis를 기반으로 한다!
- **Scalar feedback signal**
  - Vector가 아닌 +/-로만 이루어져 있음
- Agent가 행동을 잘했는지? 못했는지?
- Rt로 표현
  - Timestep t에서 Agent의 Action에 따른 reward
- Example
  - Atari game의 경우 점수가 오를때 reward +1
  - 점수가 내리거나 끝나면 -1

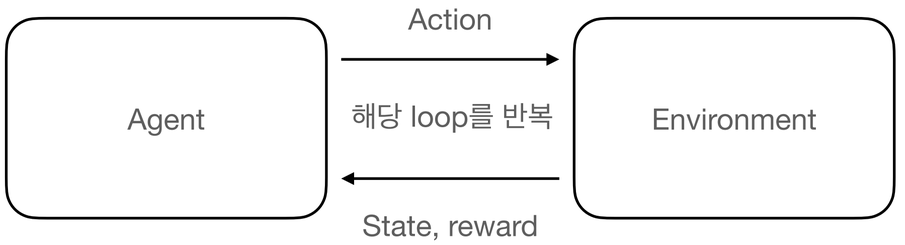




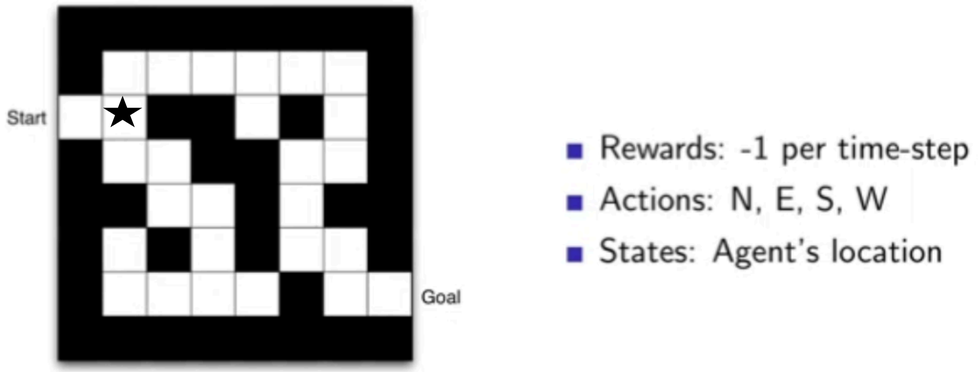
RL Base Knowledge

Agent and Environment

- **Agent** : action의 주체
  - **Policy**
  - **Value function**
  - **Model**
- **Environment**
  - Agent가 action을 취했을때 두가지 signal을 줌
  - **Reward** : 보상
  - **State** : 현재 상태에 대한 정보를 숫자로 표현해 놓은 것



목표 : Start로 부터 Goal 지점까지 한 칸씩 이동

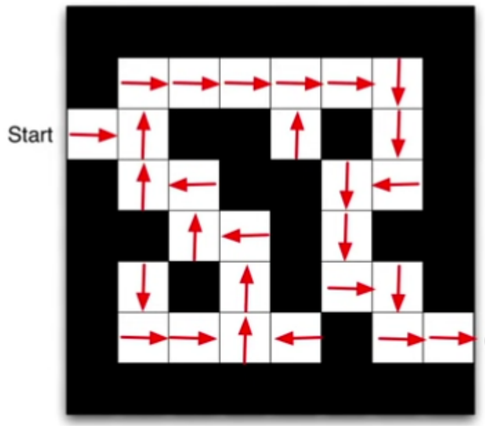


RL Base Knowledge

Agent의 구성요소

- **Policy** : Agent의 행동 정책
  - **Deterministic Policy** : state를 받고 Agent의 다음 Action을 결정
  - **Stochastic Policy** : state를 받고 여러 가능한 Action들의 확률을 계산
- **Value function** : policy을 따랐을 때 얻을 수 있는 reward의 총합의 **기댓값** 계산
  - 같은 policy라도 항상 같은 reward를 얻지 못함
    - Policy, environment에 확률적 요소들이 존재

Policy



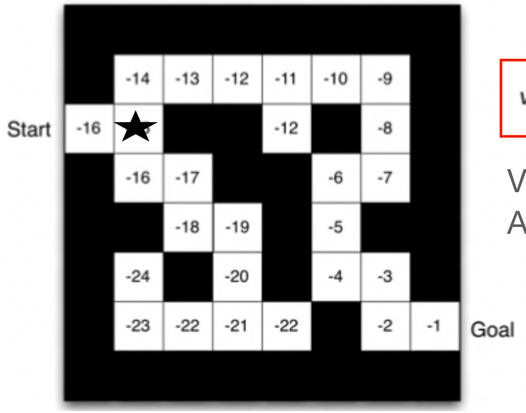
Deterministic policy: $a = \pi(s)$

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

화살표 : Agent의 최적 Policy

Agent의 state(위치)에 따라 할수 있는 최적의 행동

Value function



$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$

Value function을 통해 얻은

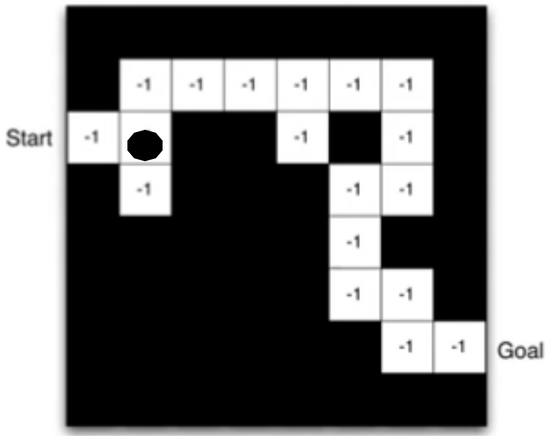
Agent의 state(위치)에 따른 누적 reward 기대값

RL Base Knowledge

Agent의 구성요소

- **Model** : Environment가 어떻게 변화할지 예측
  - Environment에서 받을 수 있는 reward, state 을 Agent의 model이 예측
  - reward를 예측 $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
    action a를 하고 State s에서 다음 state에 대한 정보
  - State를 예측 $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
    action a를 하고 State s -> s' 로 갈때 얻는 reward

Agent의 internal model



각 state에서 받는 reward : -1

Transition model : Grid layout

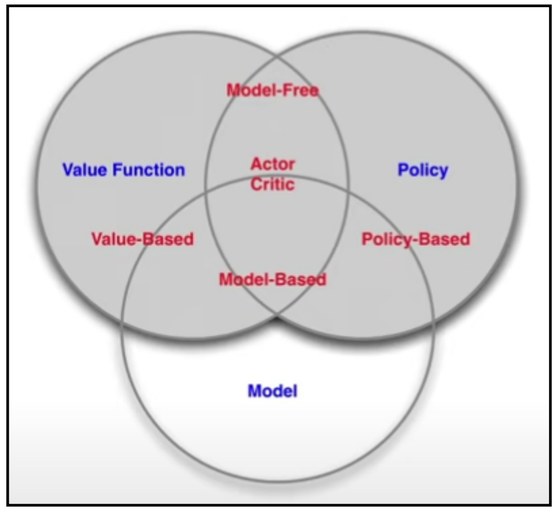
Transition model : immediate reward

Model의 일부가 드러나지 않은 이유?
Agent가 경험한 environment만 학습되었기 때문

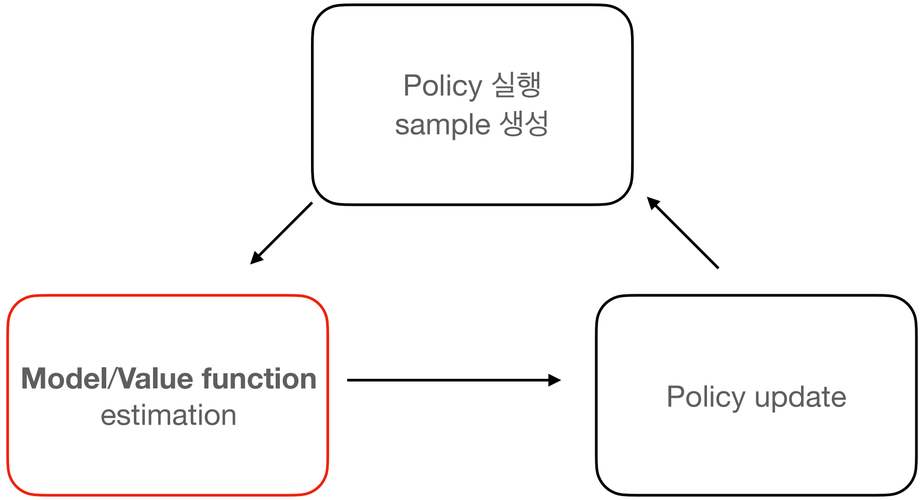
RL Base Knowledge

Categorizing RL

- **Value Based**
  - No Policy, Value Function
  - Value function을 추정해 최대 reward 계산
- **Policy Based (Policy Gradient)**
  - Policy, No Value Function
  - reward의 기댓값을 최대화하기 위해 policy를 최적화
  - Policy parameter를 계산
- **Actor Critic**
  - Policy, Value Function 모두 사용

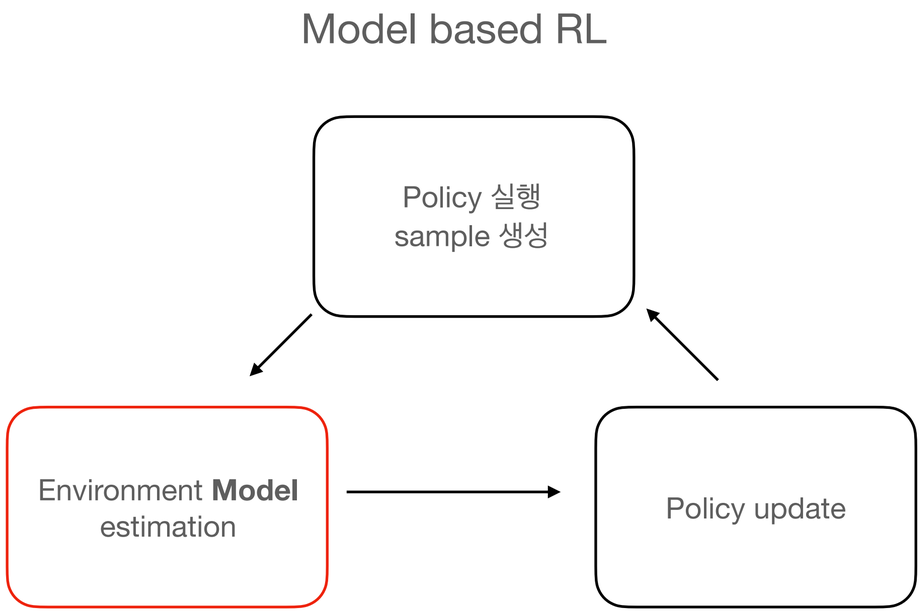


강화학습의 공통적인 iteration



RL Base Knowledge
Categorizing RL

- Model 유무로 RL을 나누기도 함
  - **Model-free RL** : model 존재 x
    - Policy and/or Value Function
  - **Model-based RL** : model 존재
    - Policy and/or Value Function
    - 간단하고 효율적
    - 로봇 제어, 드론 제어 분야에서 인기
    - model을 정의하는 방식에 따라 사용되는 policy update 방식이 다름



RL Base Knowledge

Learning & Planning

- 순차적 의사결정 문제 (sequential decision making)

    1. **Learning** : **Reinforcement Learning**
        - 알려지지 않은 environment : reward를 어떻게 주고, state가 어떻게 바뀌는지 모름
        - Agent가 environment와 상호작용 - policy update
    2. **Planning**
        - environment의 model이 알려져 있음 : reward, state의 변화를 알고 있음
        - Agent는 model을 활용해 계산을 수행 (environment와의 상호작용 없이)
        - Agent의 policy update

RL Base Knowledge

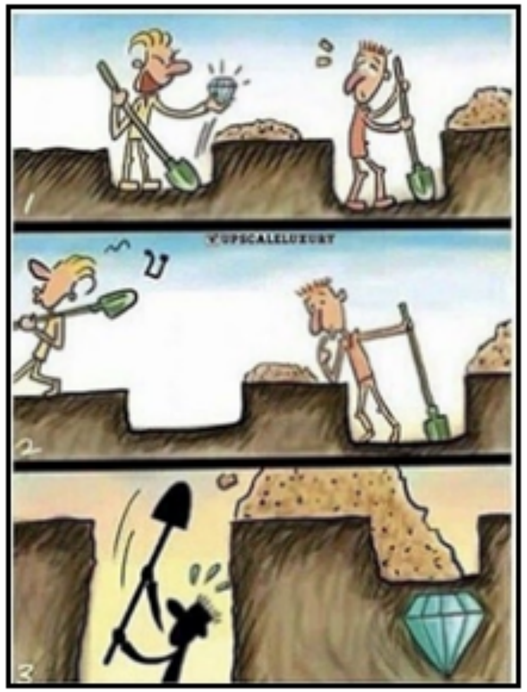
Exploration and Exploitation

- **Exploration** : 탐사,탐구
  - Environment에서 정보를 받아 이해하는 과정
  - 미지의 경험을 계속해 더 좋을지도 모르는 방법을 찾음
    - Ex. 새로운 음식에 도전
- **Exploitation** : 개척,착취
  - 현재 가진 정보를 활용해 최적의 행동을 찾음
    - Ex. 아는 음식중에서 맛있는 음식을 선택

RL Base Knowledge

Exploration and Exploitation

- Agent는 map에 대한 정보가 없음
  - Action을 통해 state , reward를 수집 해야함
  - Action을 어떻게 진행?
- **Greedy action** : 점수가 가장 큰 쪽으로 움직임
  - 미래를 생각하지 않고 각 단계에서 가장 최선의 선택
  - Exploration이 충분하지 않아 최상의 결과가 나오기 힘듬
- **Epsilon-greedy** : 점수가 가장 큰 쪽으로 움직이되, **epsilon 확률값 0.2** 만큼은 random
  - 부족한 Exploration을 보충
- **Decaying epsilon-greedy** : epsilon 값을 0에 가깝게 점점 줄여나감
  - Exploration + Exploitation

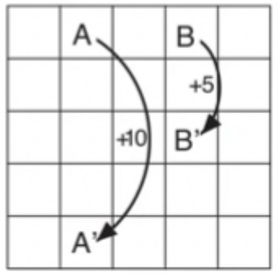


RL Base Knowledge

Prediction and Control

- 그래서 강화학습으로 뭘할 수 있는데?
- **Prediction** : evaluate the future
  - Value function을 잘 학습시키는 것
  - 어떤 state의 value가 높은지 Prediction
  - Ex. Monte-Carlo prediction
- **Control** : optimize the future
  - Find the best policy
  - agent가 최적의 행동을 하게끔 control
  - Ex. SARSA, Q-learning

Example : A,B지점에서 reward +10,+5와 함께 위치를 이동시킴
나머지 지점에서는 reward -1



(a)

Prediction

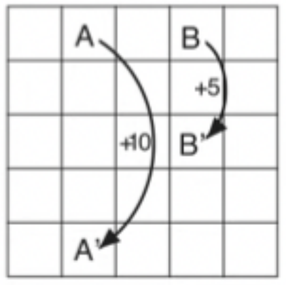
Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

각 state의 value를 평가

Control

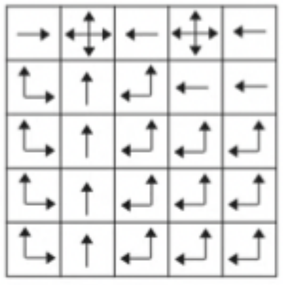


a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) v_*

optimal value function



c) π_*

optimal policy

다음시간 다룰내용

DQN을 위하여

- MDP : 순차적 의사결정 문제 - 수식화
  - Markov property
  - Markov process
- Q-Learning : value function을 어떻게 최대화 할지
  - Q-value
  - Bellman equation
- 실습 까지

감사합니다