Introduction to Reinforcement Learning

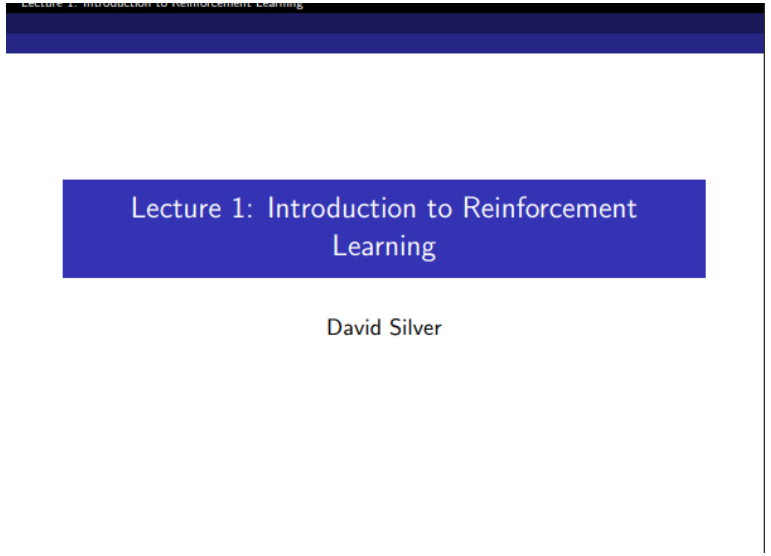
Younghoon Kim
nongaussian@gmail.com

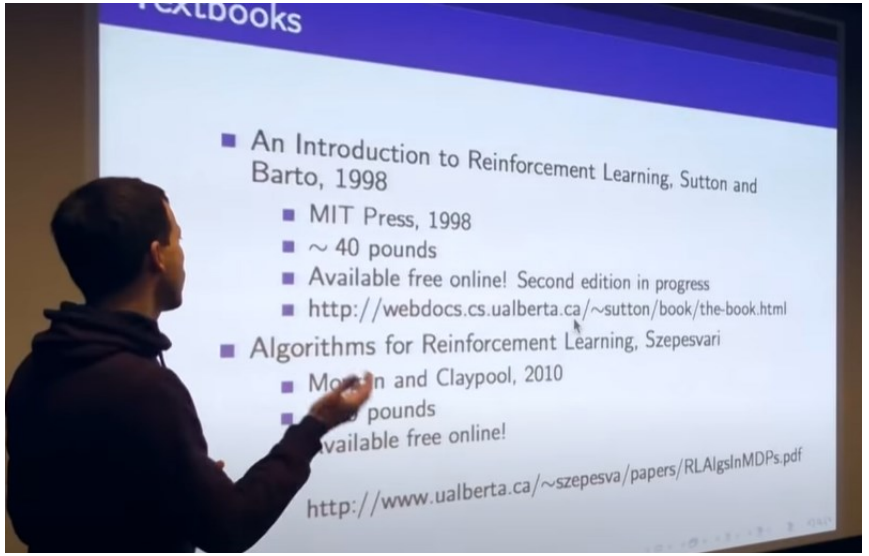
소개

- 김영훈
- 한양대학교ERICA 소프트웨어융합대학 인공지능학과
- 전공
  - 데이터베이스 및 데이터마이닝
  - 서울대학교 학사(컴퓨터공학과)/박사(전기컴퓨터공학부)
- 연구분야
  - 데이터마이닝
  - 기계학습/딥러닝

Source of The Slides

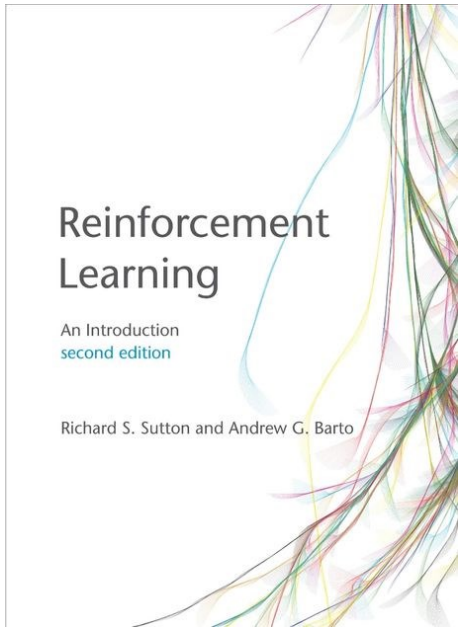
- [Introduction to Reinforcement Learning with David Silver (deepmind.com)]
  - https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver





Reference

- An Introduction to Reinforcement Learning, Sutton and Barto
  - MIT Press, 1998
  - Available free online!
  - http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html



Schedule of The Course

- Part 1. Intro. to reinforcement learning
- Part 2. Markov decision process (dynamic programming)
- Part 3. Monte-Carlo RL
- Part 4. Temporal-difference RL
- Part 5. Deep RL

Goal of This Course

- 강화학습의 중요개념을 이해한다.
- Deep Q-learning 코드를 이해한다.
- Gym을 기반으로 custom environment를 구현한다.
- 간단한 보드게임(오목)을 학습시켜본다.

LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

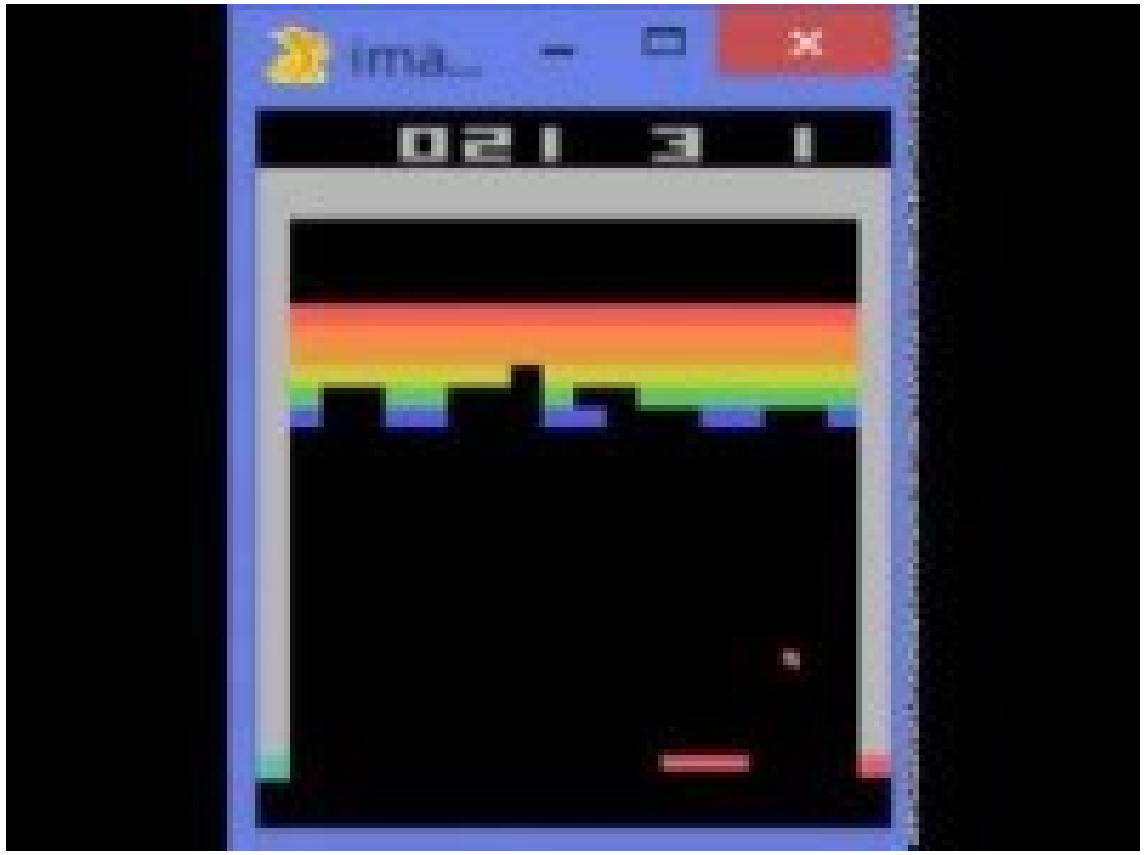
The theory of reinforcement learning provides a normative account[1], deeply rooted in psychological[2] and neuroscientific[3] perspectives on animal behaviour, of how agents may optimize their control of an environment. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations. Remarkably, humans and other animals seem to solve this problem through a harmonious combination of reinforcement learning and hierarchical sensory processing systems[4,5], the former evidenced by a wealth of neural data revealing notable parallels between the phasic signals emitted by dopaminergic neurons and temporal difference reinforcement learning algorithms[3]. While reinforcement learning agents have achieved some successes in a variety of domains[6–8], their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. Here we use recent advances in training deep neural networks[9–11] to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. We tested this agent on the challenging domain of classic Atari 2600 games[12]. We demonstrate that the deep Q-network agent, receiving only the pixels and

agent is to select actions in a fashion that maximizes cumulative future reward. More formally, we use a deep convolutional neural network to approximate the optimal action-value function

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s,\, a_t = a,\, \pi\right],$$

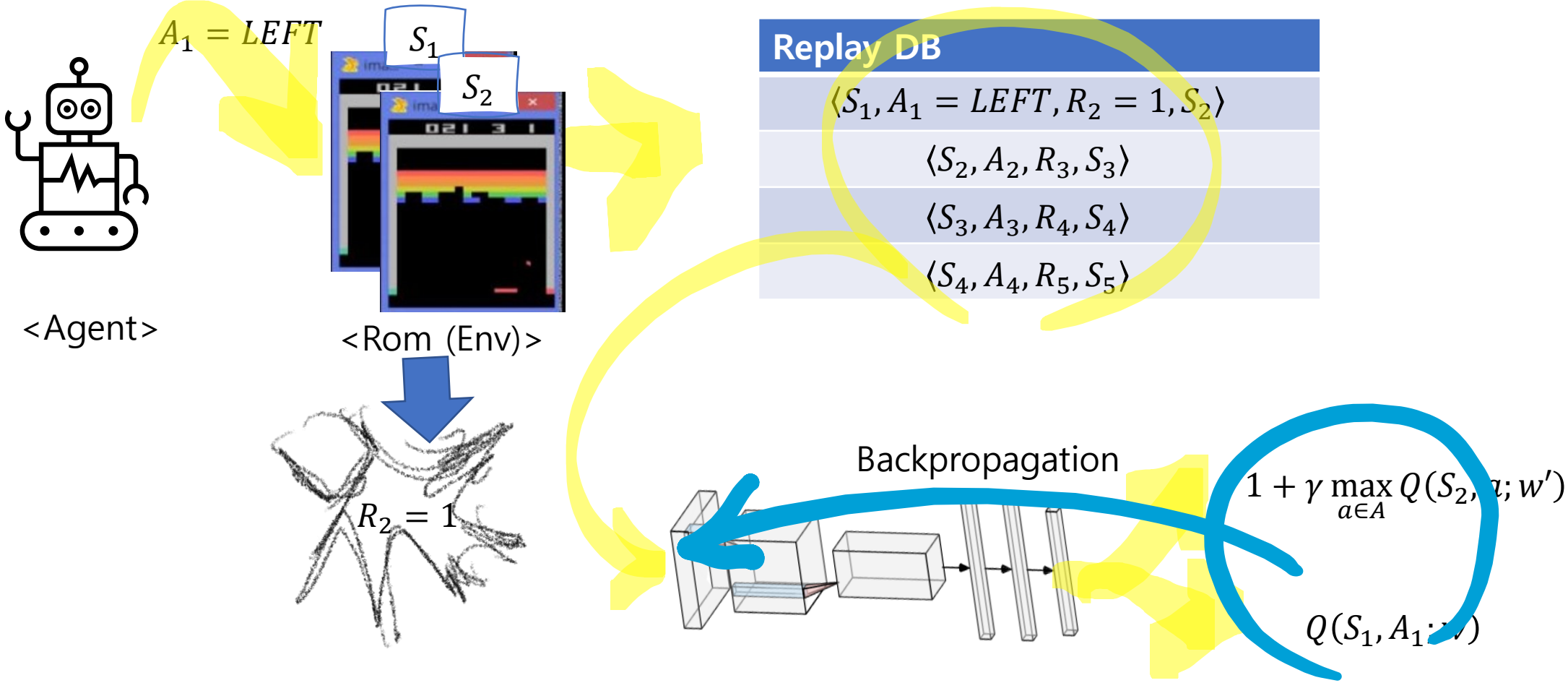
which is the maximum sum of rewards r_t discounted by γ at each time-step t, achievable by a behaviour policy $\pi = P(a|s)$, after making an observation (s) and taking an action (a) (see Methods)[19].

Reinforcement learning is known to be unstable or even to diverge when a nonlinear function approximator such as a neural network is used to represent the action-value (also known as Q) function[20]. This instability has several causes: the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and therefore change the data distribution, and the correlations between the action-values (Q) and the target values $r + \gamma \max_{a'} Q(s', a')$. We address these instabilities with a novel variant of Q-learning, which uses two key ideas. First, we used a biologically inspired mechanism termed experience replay[21–23] that randomizes over the data, thereby removing correlations in the observation sequence and smoothing over changes in the data distribution (see below for details). Second, we used an iterative update that adjusts the action-values (Q) towards target values that are only periodically updated, thereby reducing correlations with the target.



What Can Reinforcement Learning Do?

Learning To Play Atari Game with DQN



Applications of Reinforcement Learning

- 헬리콥터에서 스턴트 기동 비행하기
- 바둑, 체스 등 퍼즐게임 배우기
- 투자 포트폴리오 관리
- 발전소 제어하기
- 휴머노이드 로봇 걷기
- 비디오 게임 플레이하기
- ChatGPT 대화 학습시키기

Traditional Categories of Machine Learning

Supervised learning

- 정확한 목표 클래스를 가진 훈련 데이터가 주어지면 입력 테스트 데이터의 목표 클래스를 예측하는 함수를 훈련

Unsupervised learning

- 유사도에 따라 입력 데이터를 분류하는 함수, 즉, 유사한 데이터는 동일한 그룹에 넣고 유사하지 않은 데이터는 다른 그룹으로 분리하는 함수를 학습

Categories of Machine Learning

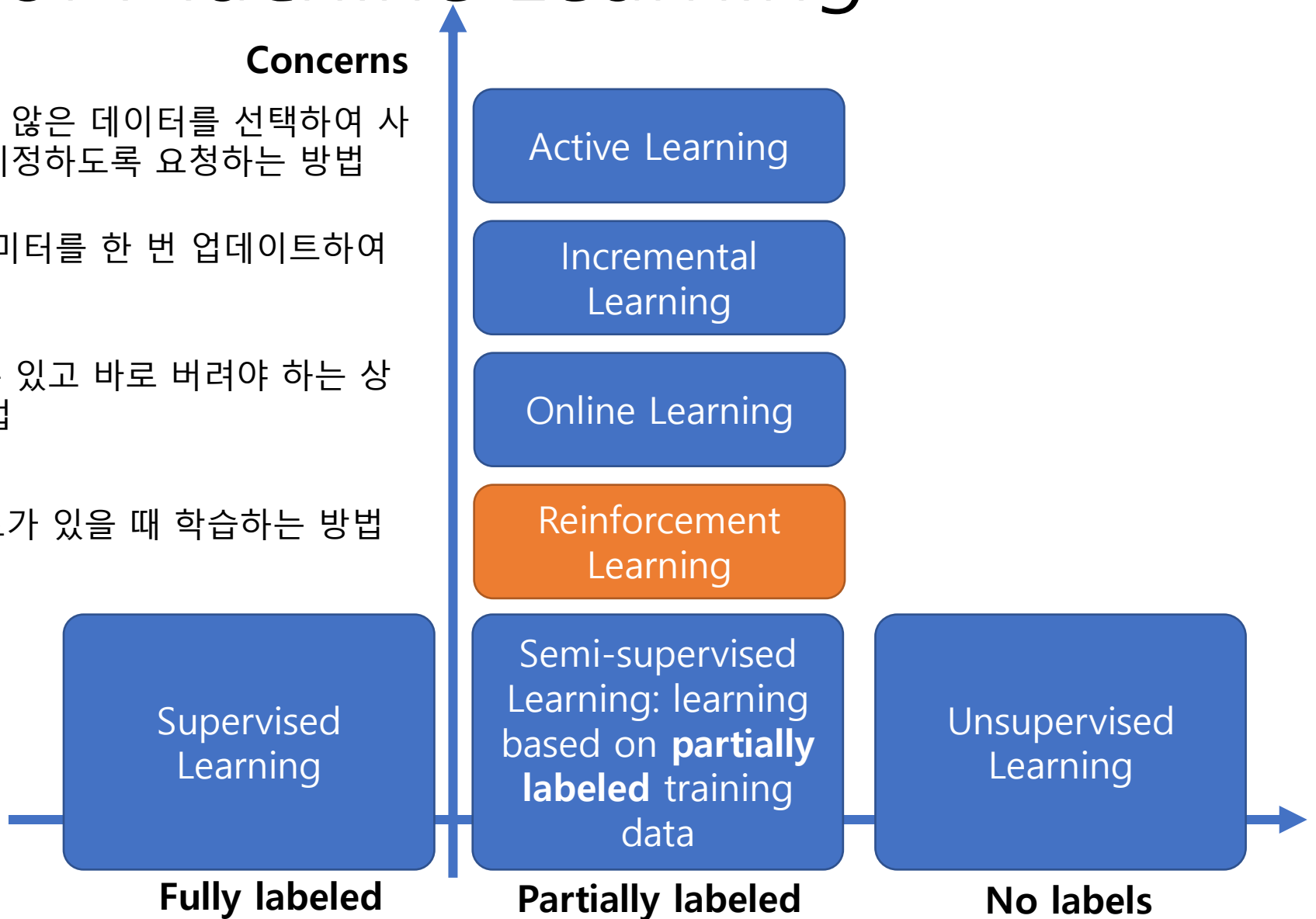
Concerns

라벨이 지정되지 않은 데이터를 선택하여 사람에게 라벨을 지정하도록 요청하는 방법

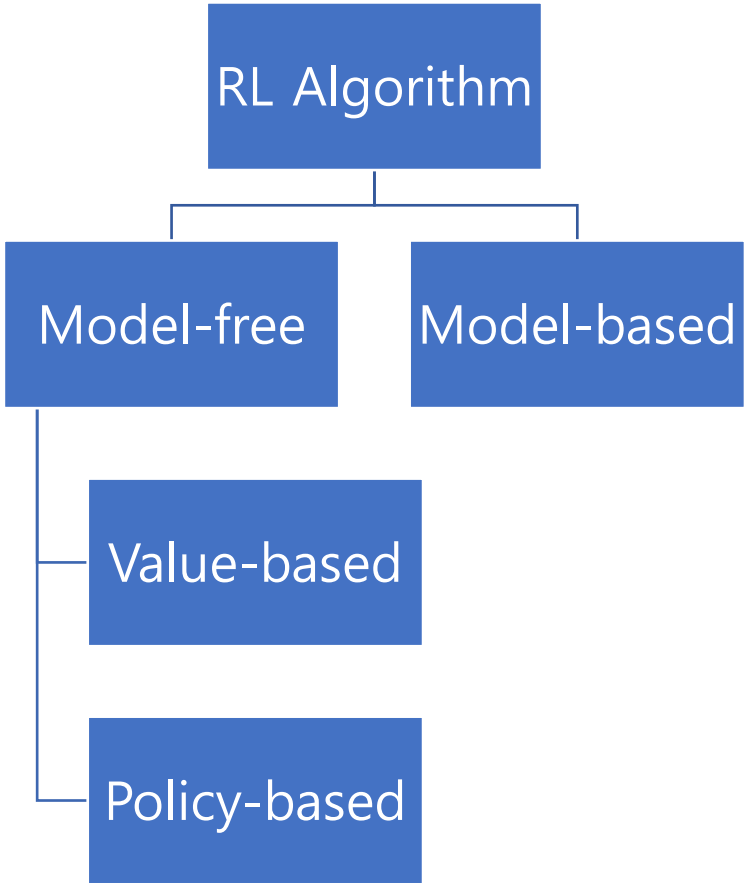
샘플(또는 샘플 세트)로 모델 파라미터를 한 번 업데이트하여 모델을 학습시키는 방법

샘플을 한 번만 볼 수 있고 바로 버려야 하는 상황에서 학습하는 방법

평가될 수 있는 일련의 이벤트가 있을 때 학습하는 방법



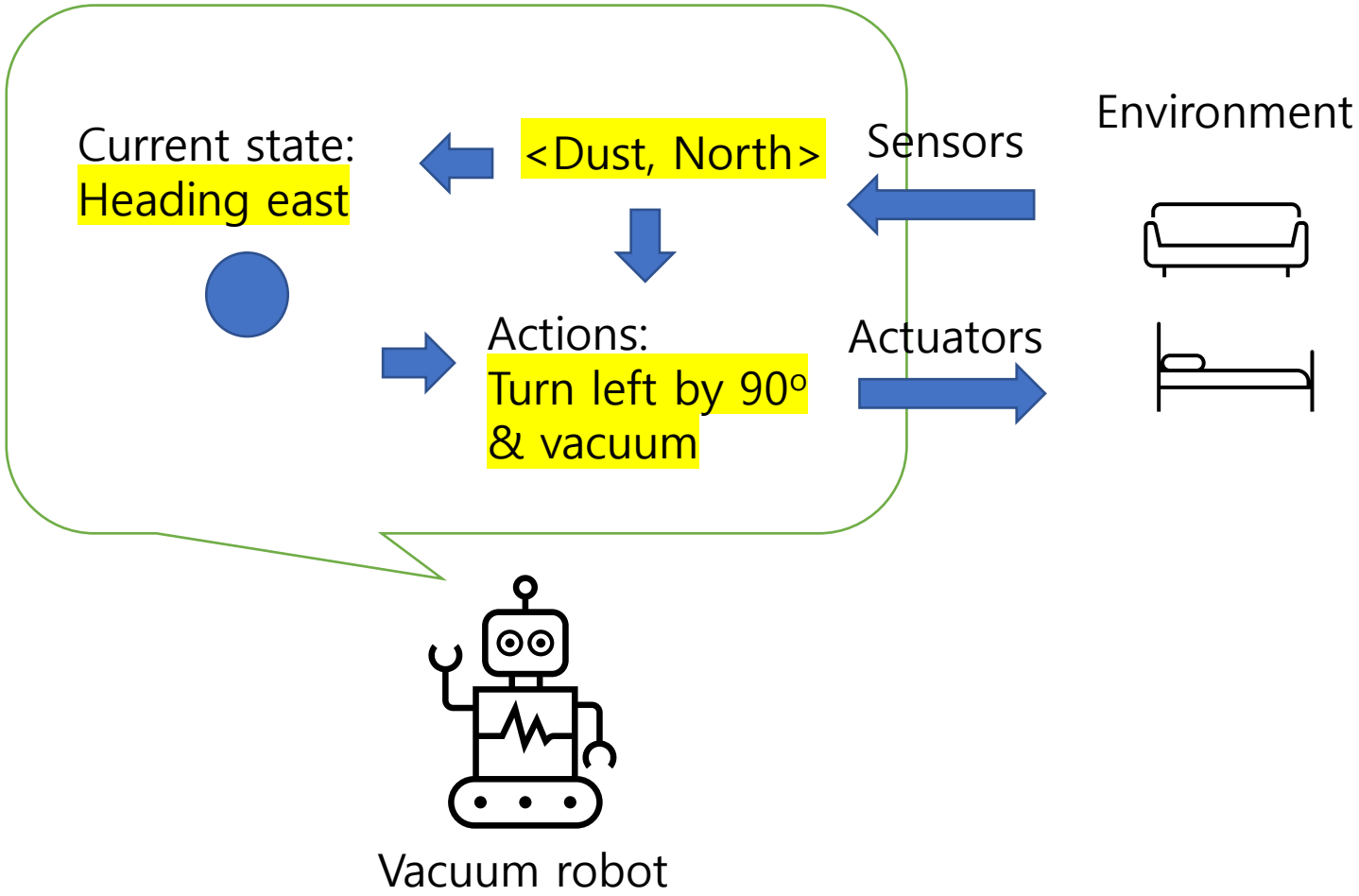
Taxonomy


RL Algorithm
Model-free
Model-based
Value-based
Policy-based

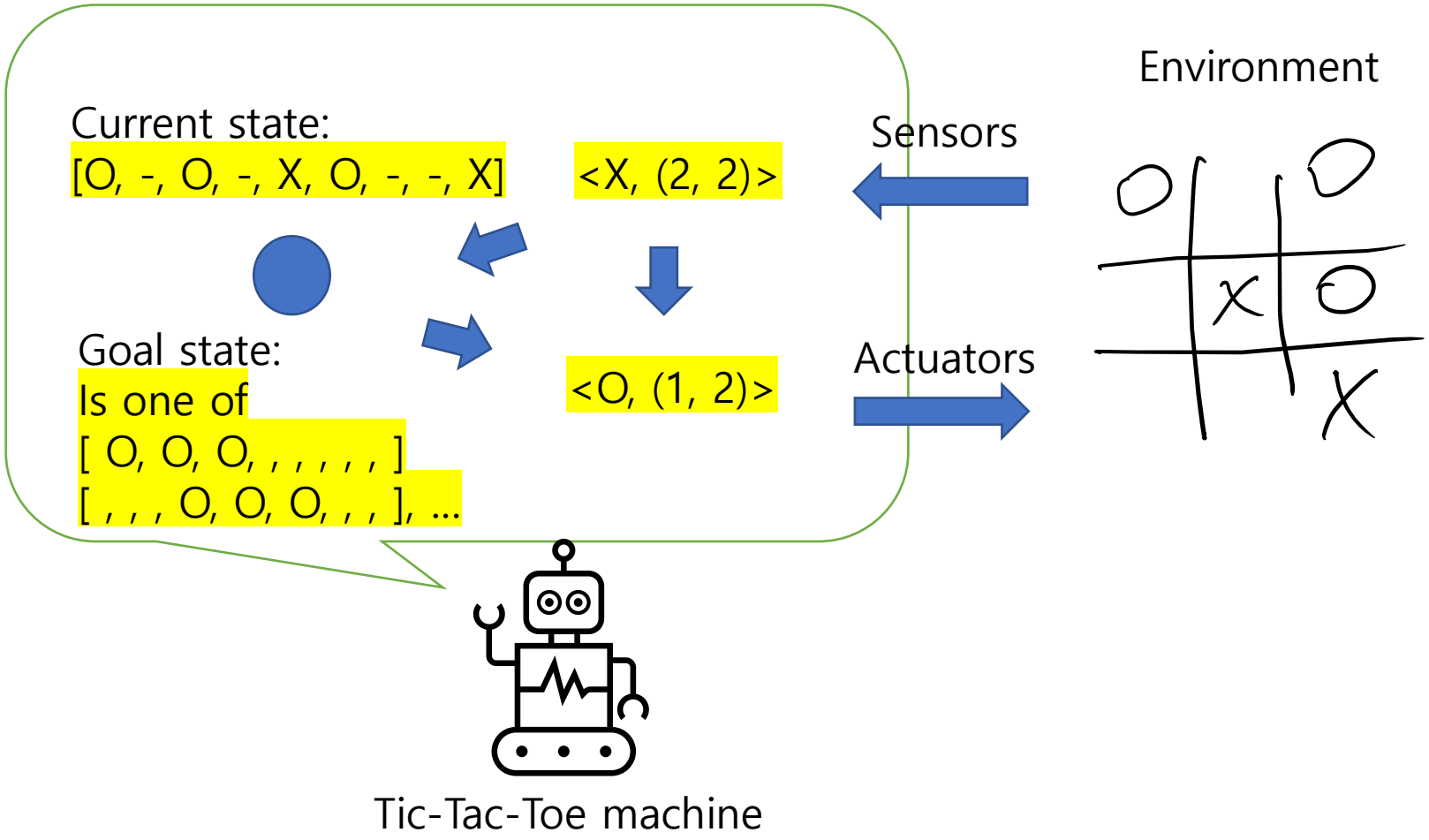
PEAS

- 새로운 AI 에이전트를 개발했다고 가정
- 어떤 요소를 정의해야 하는가?
- Consider (e.g., automatic driving)
  - Performance measure (time, safety, comfort)
  - Environment (Navigation path, traffic, pedestrians)
  - Actuator (accelerator, brake, horn)
  - Sensors (Lidar, video, GPS, gyroscope)

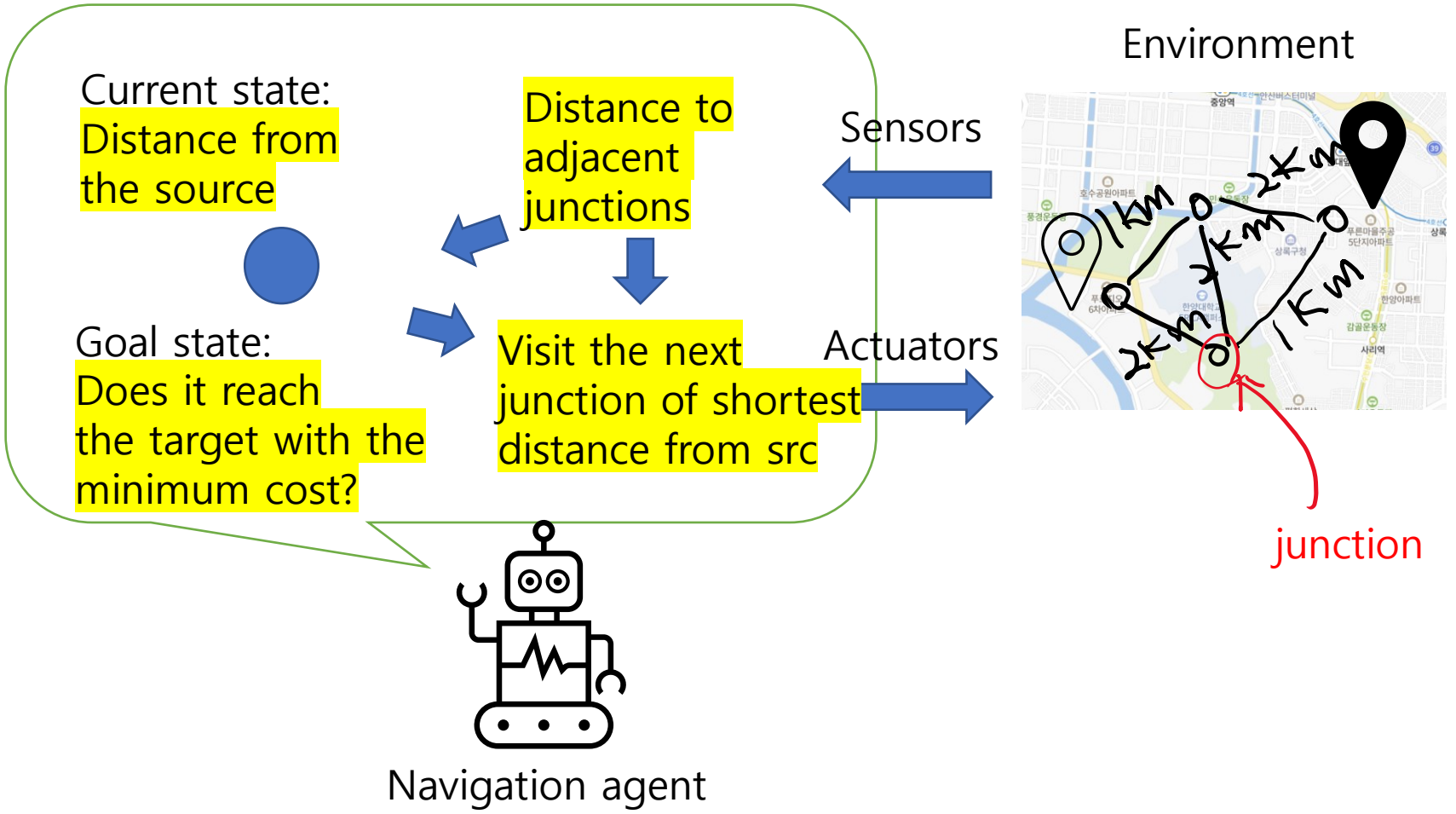
Reflex Agents 반사적 에이전트



Goal-based Agents 목표기반 에이전트

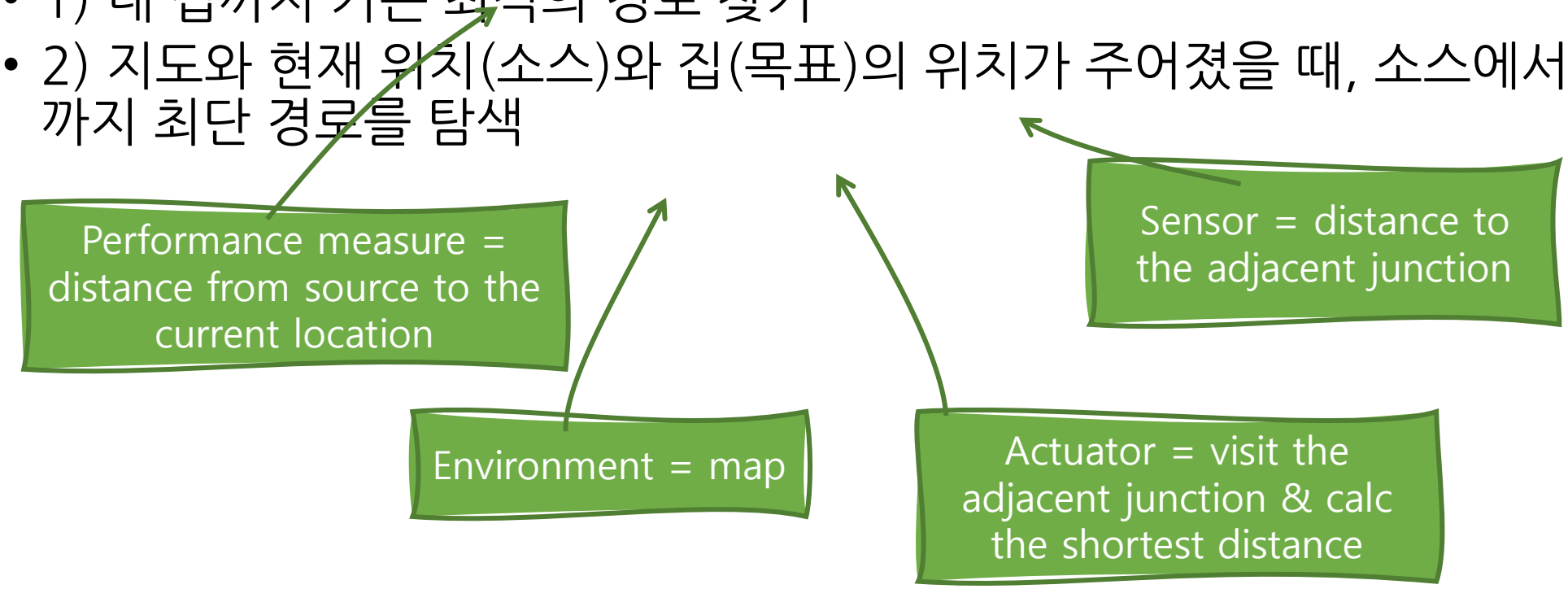


Utility-based Agents 평가지표기반 에이전트



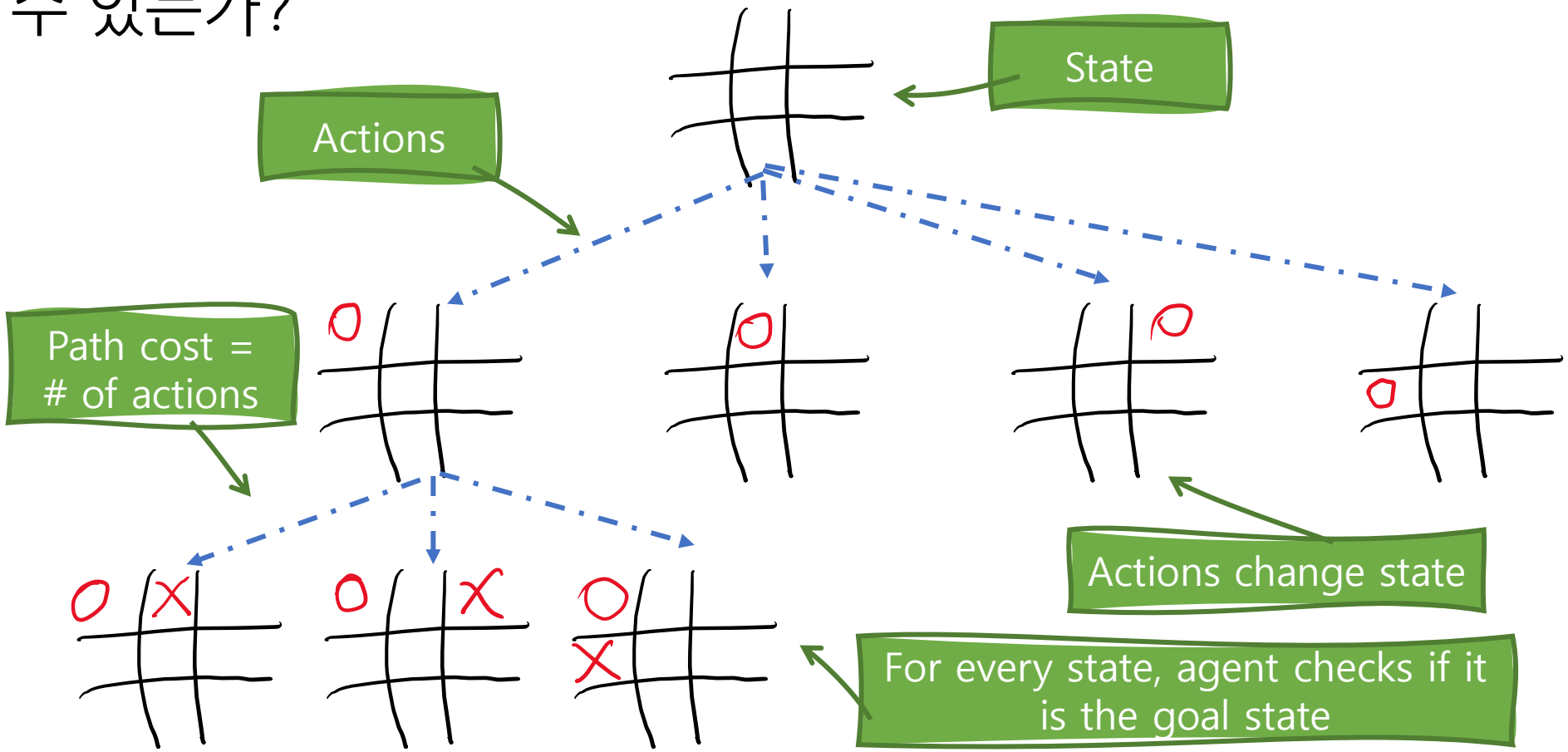
AI Agent Problem

- PEAS: 지능형 에이전트의 문제를 어떻게 정의할 수 있을까?
  - 1) 내 집까지 가는 최적의 경로 찾기
  - 2) 지도와 현재 위치(소스)와 집(목표)의 위치가 주어졌을 때, 소스에서 목표까지 최단 경로를 탐색



Solving Problems

- 틱택토 게임도 최단 경로 문제를 풀 때와 동일한 알고리즘을 사용해 풀 수 있는가?



Solving Problems Using Tree Search

- To solve problems using tree search, we define
  - State
  - Actions
  - Goal test
  - Path cost

여기에서는 동작(action)을 선택하는 것이나 리워드는 정의하지 않는다. 어떤 차이인가?

Goal test와 리워드가 같은 것 아닌가?

Characteristics of Reinforcement Learning

- 감독자는 없고 보상 신호만 있다
- 보통, 피드백은 즉각적이지 않고 지연됨
- 시간이 정말 중요함 (순차적, Non-i.i.d 데이터)
- 에이전트의 행동이 이후 관측되는 데이터에 영향을 미침

Problem Formulation of Reinforcement Learning

- Data
  - A sequence of state, action and reward

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, \dots, S_{T-1}, A_{T-1}, R_T, S_T$$

  - where
    - S_t is state at step t
    - A_t is the action which the agent takes at step t
    - R_t is a reward representing how good the state S_t
- Goal
  - 에이전트의 누적 리워드를 최대화하라!

Examples of Rewards

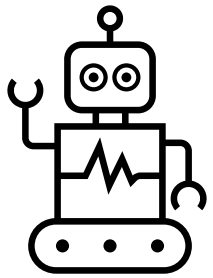
- 헬리콥터에서 스턴트 기동 비행
  - 원하는 궤도 따라 비행 시 +5배 보상
  - -추락 시 보상
- 바둑, 체스 등 보드 게임
  - 게임 승패에 따라 +/-보상 제공
- 투자 포트폴리오 관리
  - 계좌의 $ 증감 분에 대해 보상
- 발전소 제어
  - 전력 생산에 대한 + 보상
  - 안전 임계값 초과에 대한 - 보상
- 휴머노이드 로봇 걷기
  - 앞으로 나아갈 때 + 보상
  - 넘어지면 - 보상
- 비디오 게임
  - 점수 증가/감소에 따른 보상 +/-

Input Data for Reinforcement Learning

- The sequence of observations, actions and rewards

$$H = \langle O_1, A_1, R_2, O_2, A_2, \ldots, A_{T-1}, R_T, O_T \rangle$$

How the agent acts



에이전트가 환경으로부터 O_t를 관측

행동 A_t

Input Data for Reinforcement Learning

- The sequence of observations, actions and rewards

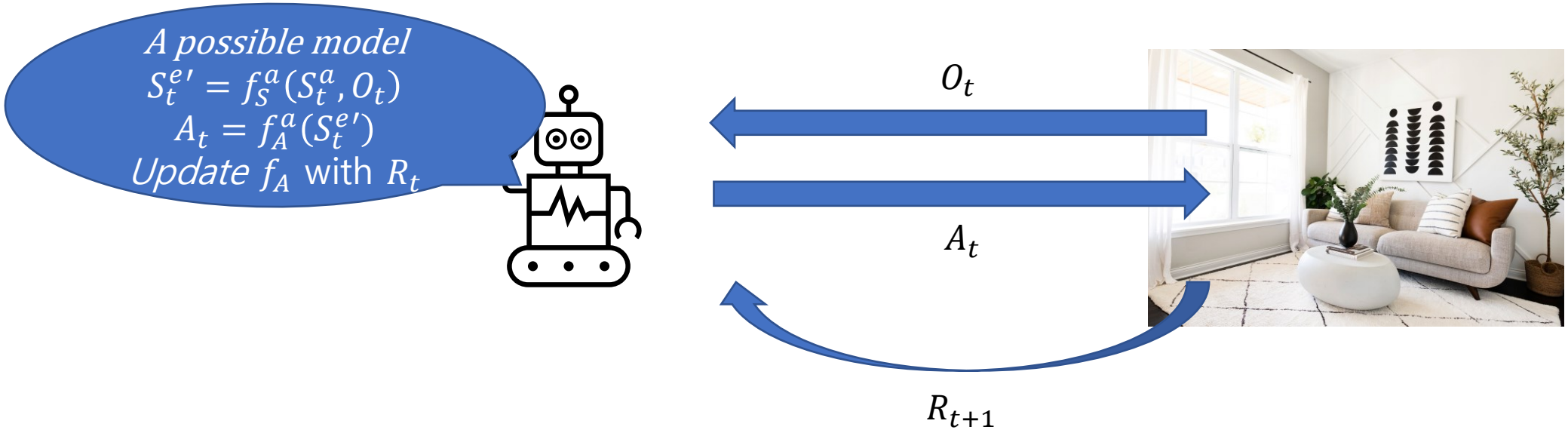
$$H = \langle O_1, A_1, R_2, O_2, A_2, \ldots, A_{T-1}, R_T, O_T \rangle$$

Environment State

- 환경의 상태를 표현
  - E.g.,
    - Location of all furniture
    - Amount of dust over the floor
- 환경은 환경 상태 S_t^e 에 따라 다음 관찰 및 보상을 계산

Agent State

- A variable S_t^a representing
  - 에이전트가 환경을 인식하는 방법
  - 또는 에이전트가 다음 행동을 선택하는 데 사용하는 모든 정보



Fully Observable Environments

- An assumption that
  - An agent can observe the environment state
  - i.e.,

$$S_t^{e\prime} = S_t^e \text{ with } S_t^{e\prime} = f_S^a(S_t^a, O_t)$$

That is, O_t includes all information the agent needs to know S_t^e

- 이와 같은 일련의 과정에 대한 모델이 Markov Decision Process (MDP)

A state S_t is called (first order) Markov if and only if

$$\Pr(S_{t+1}|S_t) = \Pr(S_{t+1}|S_1, S_2, \dots, S_t)$$

Partially Observable Environments

- 일반적으로 에이전트는 환경 상태 S_t^e 를 완전히 관찰할 수 없다
- That is

$$S_t^{e\prime} \neq S_t^e \text{ with } S_t^{e\prime} = f_S^a(S_t^a, O_t)$$

That is, an agent cannot perceive S_t^e using the observation O_t

- 이와 같은 경우를 Partially Observable Markov Decision Process (POMDP) 라고 함

Let us define a reinforcement learning agent under the POE assumption!

Components of an RL Agent

Policy

- 에이전트의 다음 행동을 계산하는 함수

Value function

- 어떤 상태나 행동이 **장기적으로** 얼마나 좋은지 추정하는 함수

Model

- 자신의 행동에 대한 환경의 반응을 계산하기 위해 에이전트가 알고 있는 함수

Policy

- 에이전트의 행동을 선택하기 위한 함수
- Two types of policy functions:
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a|s) = \Pr(A_t = a|S_t = s)$

Then, it chooses an action a^* with the maximum probability. That is, $a^* = \operatorname{argmax} \pi(a|s)$

Value Function

- 최종적으로 현재 상태가 미래에 얼마나 좋은지 평가하기 위한 함수
  - Value function을 기반으로 최선의 다음 행동을 선택
  - 보상 (리워드)은 즉각적인 의미에서 무엇이 좋은지를 나타냄
- *통상적 계산 방법*
  - 가치 함수는 다음과 같이 기하급수적으로 감소하는 가중치를 가진 미래 보상의 합을 기대하는 것으로 정의

$$v_\pi(s) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s] \text{ with } 0 \leq \gamma \leq 1$$

(Optional) Model

- 환경이 다음에 어떻게 반응할지 예측
- 예를 들어
  - 다음과 같은 확률 분포로 표현

$$\Pr(S_{t+1}|S_t = s, A_t = a)$$

gym

- Gym은 RL 알고리즘 개발 및 테스트를 위한 툴킷
  - OpenAI
  - https://www.gymlibrary.dev/index.html
  - 학습 알고리즘과 환경 간의 통신을 위한 표준 API
  - 걷기부터 탁구나 핀볼과 같은 게임까지 에이전트를 학습시키기 위한 환경(반응)을 제공

Practice: Atari Breakout Game

- 복붙 & 타이핑하며 블럭단위로 코드 이해
- 파일: pytorch-dqn-atari-practice.pdf