Lecture 1: Introduction to Reinforcement Learning

David Silver

Outline

1 Admin

2 About Reinforcement Learning

3 The Reinforcement Learning Problem

4 Inside An RL Agent

5 Problems within Reinforcement Learning

Class Information

- Thursdays 9:30 to 11:00am
- Website:
  http://www.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html
- Group:
  http://groups.google.com/group/csml-advanced-topics
- Contact me: d.silver@cs.ucl.ac.uk

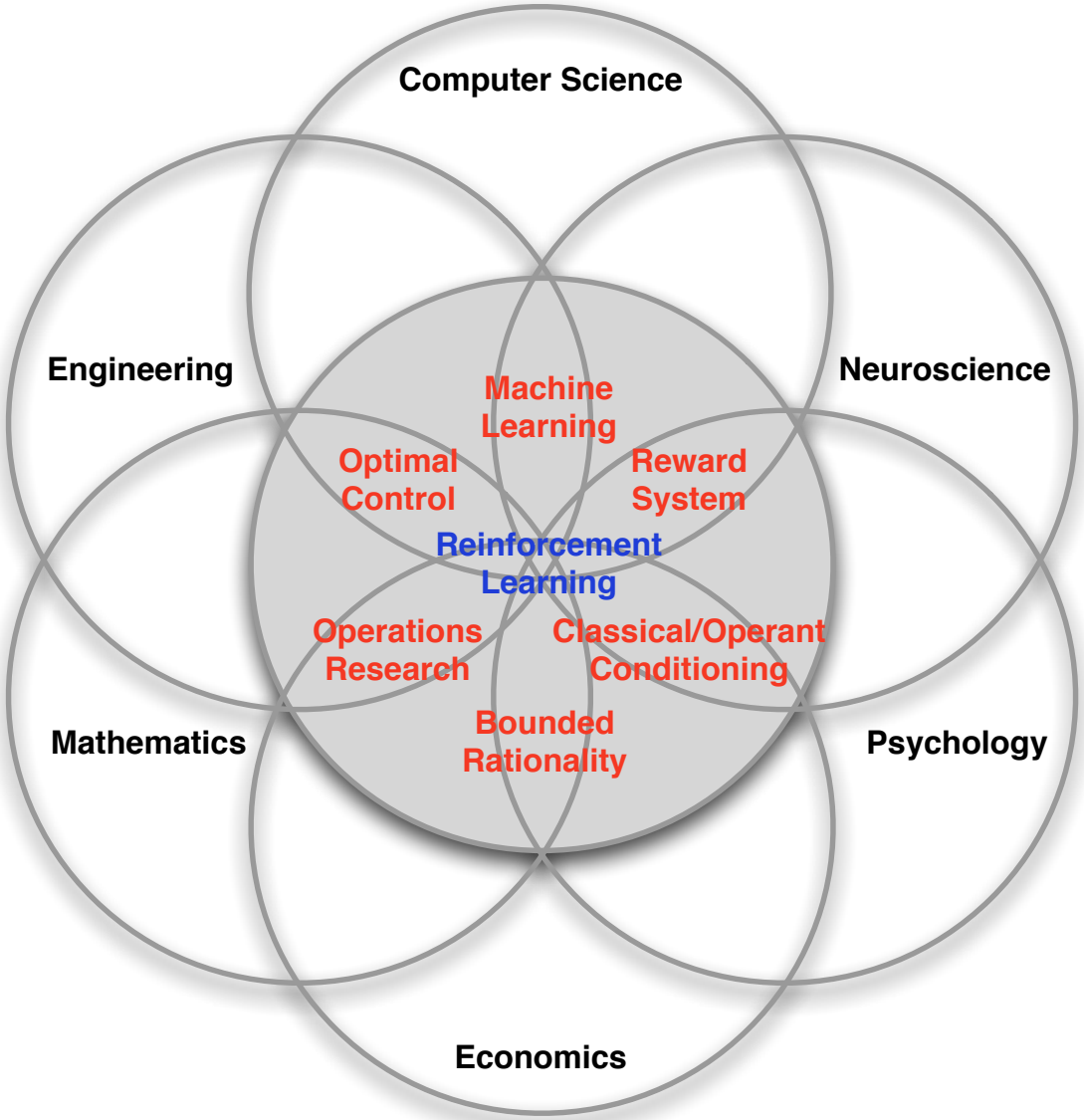
Assessment

- Assessment will be 50% coursework, 50% exam
- Coursework
  - Assignment A: RL problem
  - Assignment B: Kernels problem
  - Assessment $= max(assignment1, assignment2)$
- Examination
  - A: 3 RL questions
  - B: 3 kernels questions
  - Answer any 3 questions

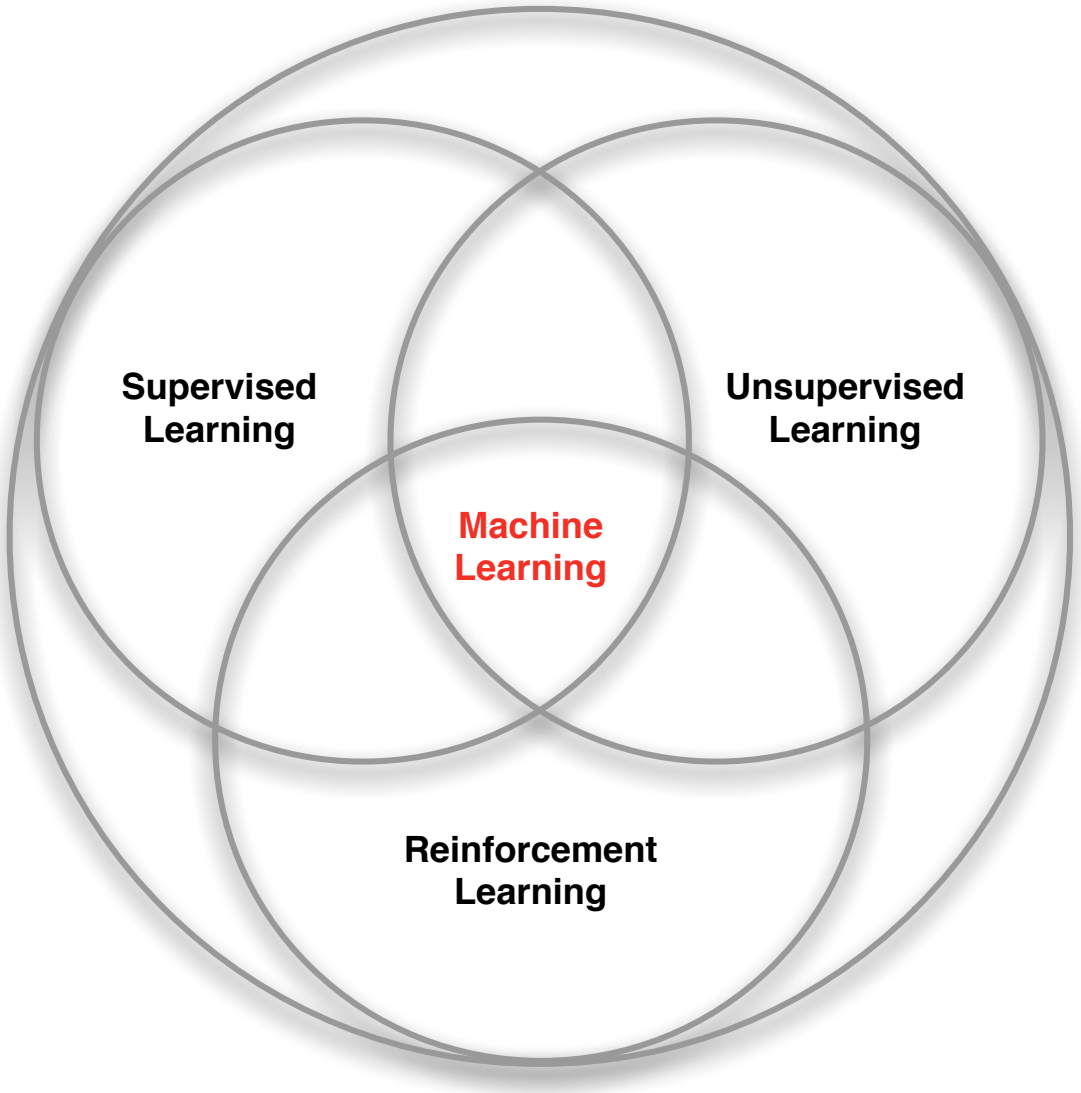
Textbooks

- An Introduction to Reinforcement Learning, Sutton and Barto, 1998
  - MIT Press, 1998
  - $\sim$ 40 pounds
  - Available free online!
  - http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html
- Algorithms for Reinforcement Learning, Szepesvari
  - Morgan and Claypool, 2010
  - $\sim$ 20 pounds
  - Available free online!
  - http://www.ualberta.ca/~szepesva/papers/RLAlgsInMDPs.pdf

Many Faces of Reinforcement Learning



Branches of Machine Learning



Characteristics of Reinforcement Learning

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a *reward* signal 그냥 reward 한줄뿐임.
- Feedback is delayed, not instantaneous 행동을 지나고서 결과를 얻는다. 즉각적으로 Feedback하지 않음.
- Time really matters (sequential, non i.i.d data) 시간에 따른 sequential한 system이다. iid data 아닌 sequential data.
- Agent's actions affect the subsequent data it receives 행동이 뒤이어 나올 data에 영향을 준다.

*i.i.d : independent identically distribution

Examples of Reinforcement Learning

- Fly stunt manoeuvres in a helicopter
- Defeat the world champion at Backgammon
- Manage an investment portfolio
- Control a power station
- Make a humanoid robot walk
- Play many different Atari games better than humans

Helicopter Manoeuvres

Video

Bipedal Robots

Atari

Atari game playing video

Rewards

Agent의 목적 maximum total reward 받는 Policy 찾기. (=optimal policy)

- A reward R_t is a scalar feedback signal
- Indicates how well agent is doing at step t
- The agent's job is to maximise cumulative reward

Reinforcement learning is based on the reward hypothesis

Definition (Reward Hypothesis)

All goals can be described by the maximisation of expected cumulative reward

Do you agree with this statement? RL은 Reward를 얼마나 잘 주는지에 따라 다르다.

Examples of Rewards

- Fly stunt manoeuvres in a helicopter
  - +ve reward for following desired trajectory
  - −ve reward for crashing
- Defeat the world champion at Backgammon
  - +/−ve reward for winning/losing a game
- Manage an investment portfolio
  - +ve reward for each $ in bank
- Control a power station
  - +ve reward for producing power
  - −ve reward for exceeding safety thresholds
- Make a humanoid robot walk
  - +ve reward for forward motion
  - −ve reward for falling over
- Play many different Atari games better than humans
  - +/−ve reward for increasing/decreasing score

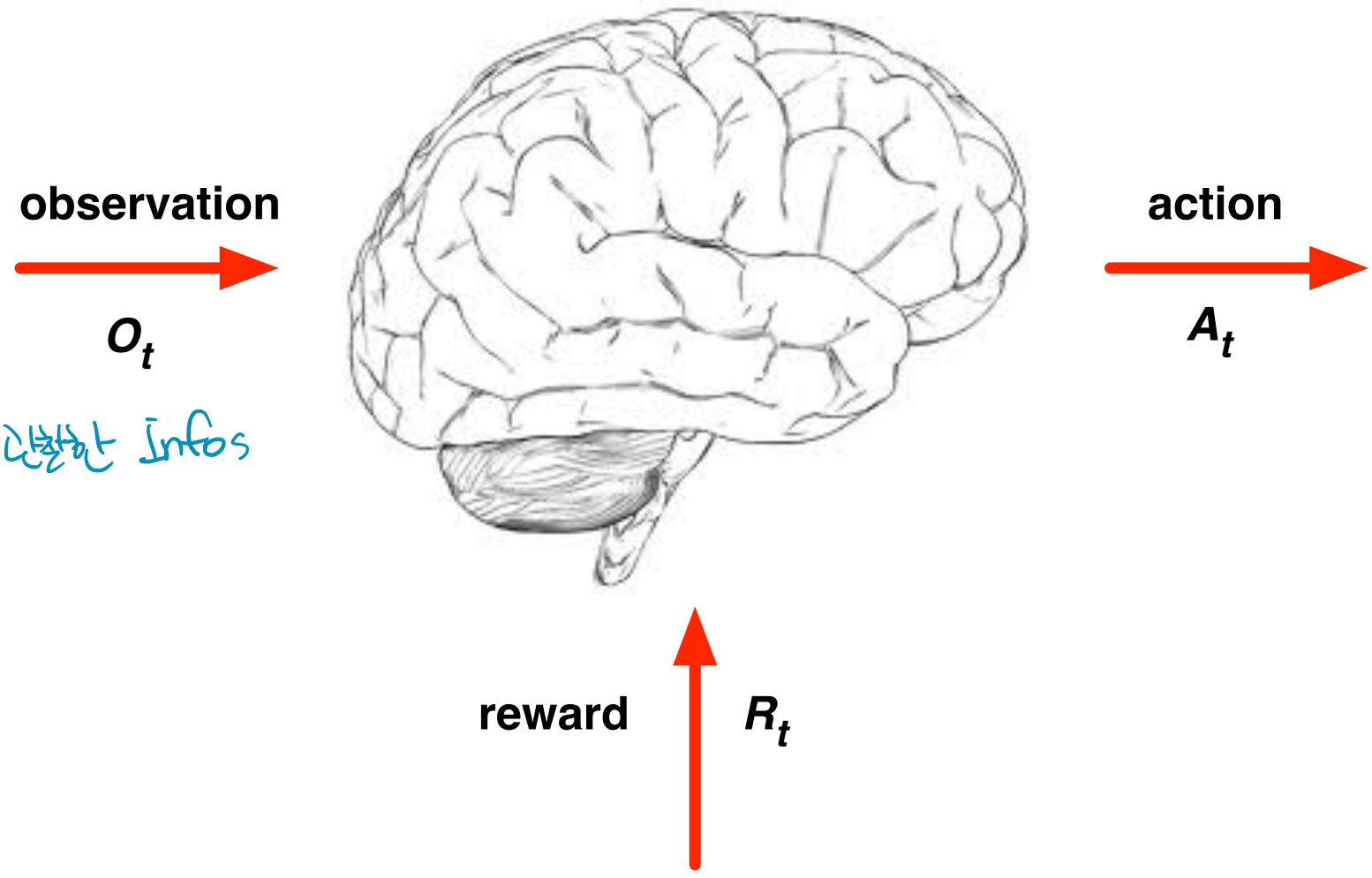
* agent가 환가 user의 의도대로 행동을 해서 좋은 결과를 얻으면 positive reward, 아니면 negative 값을 준다.

Game 외 다른 문제에도 적용될 곳이 많다.

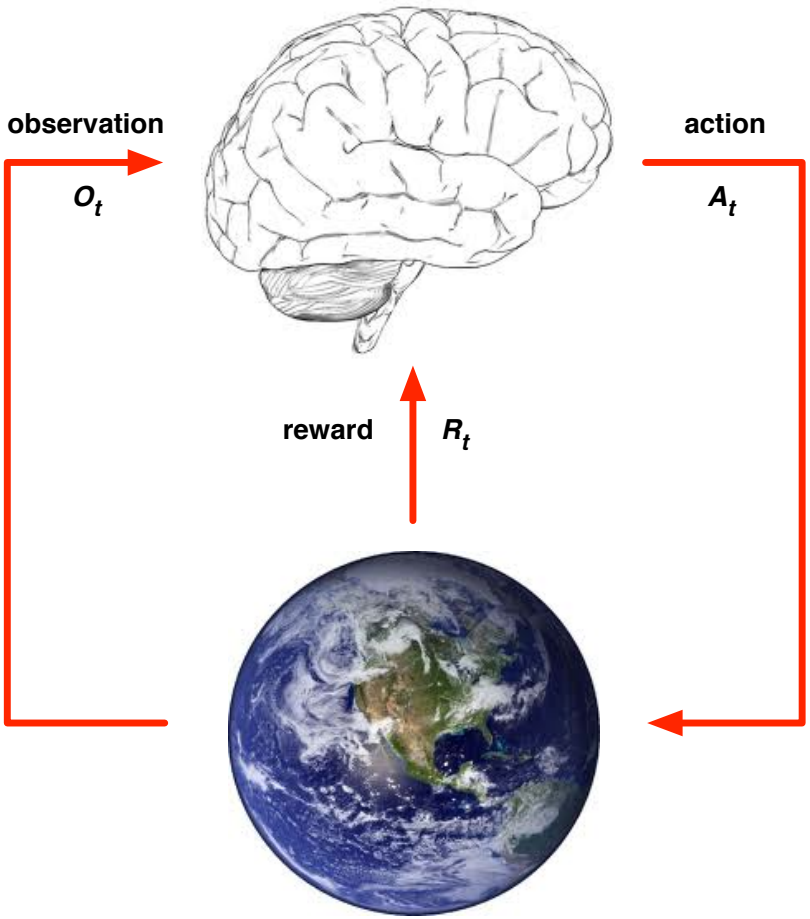
Sequential Decision Making

- Goal: *select actions to maximise total future reward*
- Actions may have long term consequences
- Reward may be delayed 현 state 에서 action a 했을때 다음 state로 넘어가면서 reward를 받는다.
- It may be better to sacrifice immediate reward to gain more long-term reward 즉각적인 reward 버리고 long-term을 얻는 것도 고려해야함
- Examples:
  - A financial investment (may take months to mature)
  - Refuelling a helicopter (might prevent a crash in several hours)
  - Blocking opponent moves (might help winning chances many moves from now)

Agent and Environment



Agent and Environment



environment (Agent 이 보는 것)

- At each step t the agent:
  - Executes action A_t
  - Receives observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at env. step

History and State

- The history is the sequence of observations, actions, rewards

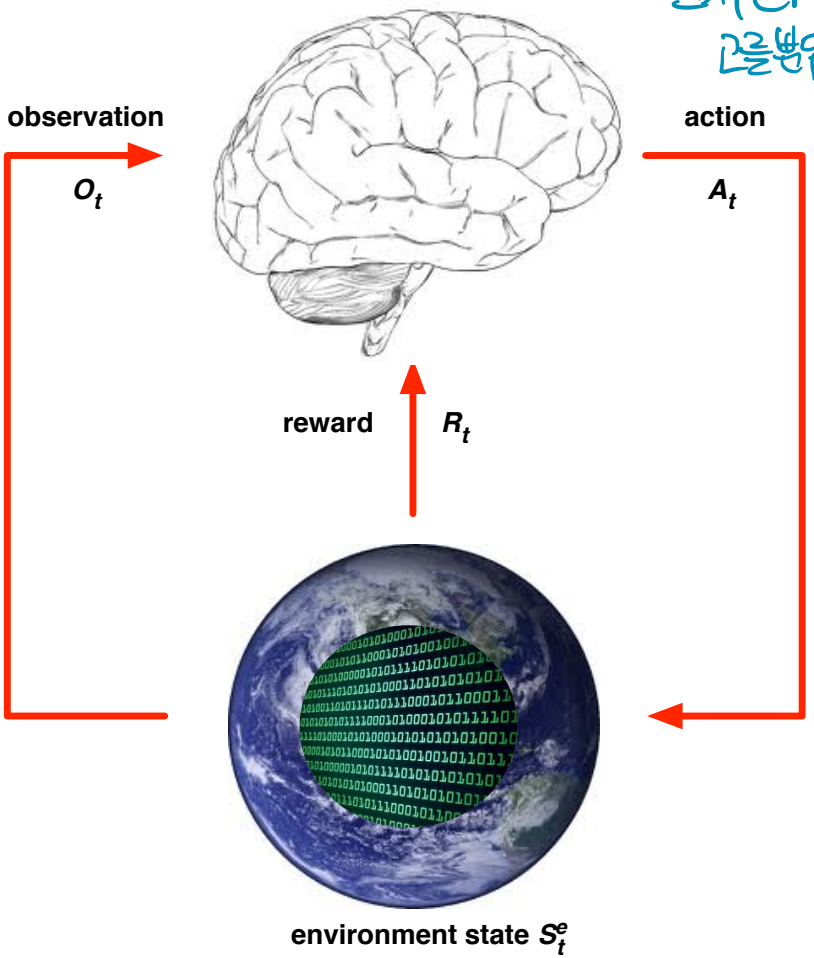
$$H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$$

- i.e. all observable variables up to time t
- i.e. the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next
  State는 다음에 무엇이 일어나는가를 결정하기 위해 사용되는 info.
- Formally, state is a function of the history:

$$S_t = f(H_t)$$

Environment State

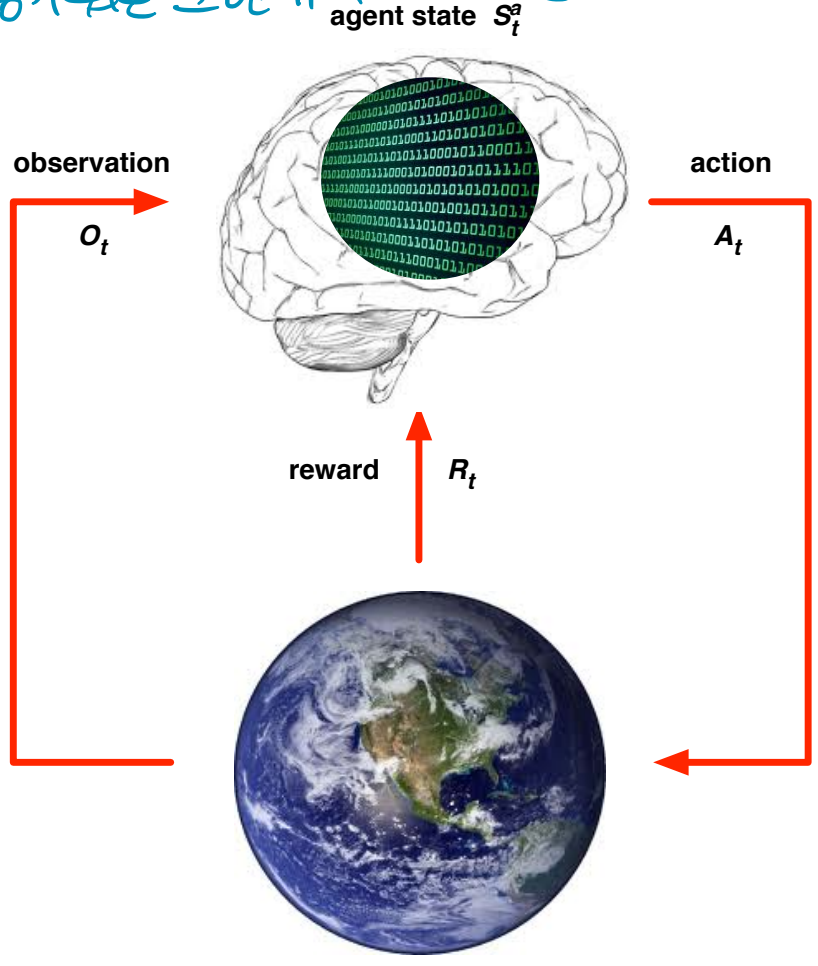
environment의 state는 agent가 볼수 없게 되어있음.
그래서 next observation과 reward를 만들뿐임.



- The environment state S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

Agent State

Agent의 state는 agent의 내재된 표현이다.
info가 있는 그안에서 agent는 next action을 고른다.



- The agent state S_t^a is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f(H_t)$$

Information State

An information state (a.k.a. Markov state) contains all useful information from the history.

Definition

A state S_t is Markov if and only if

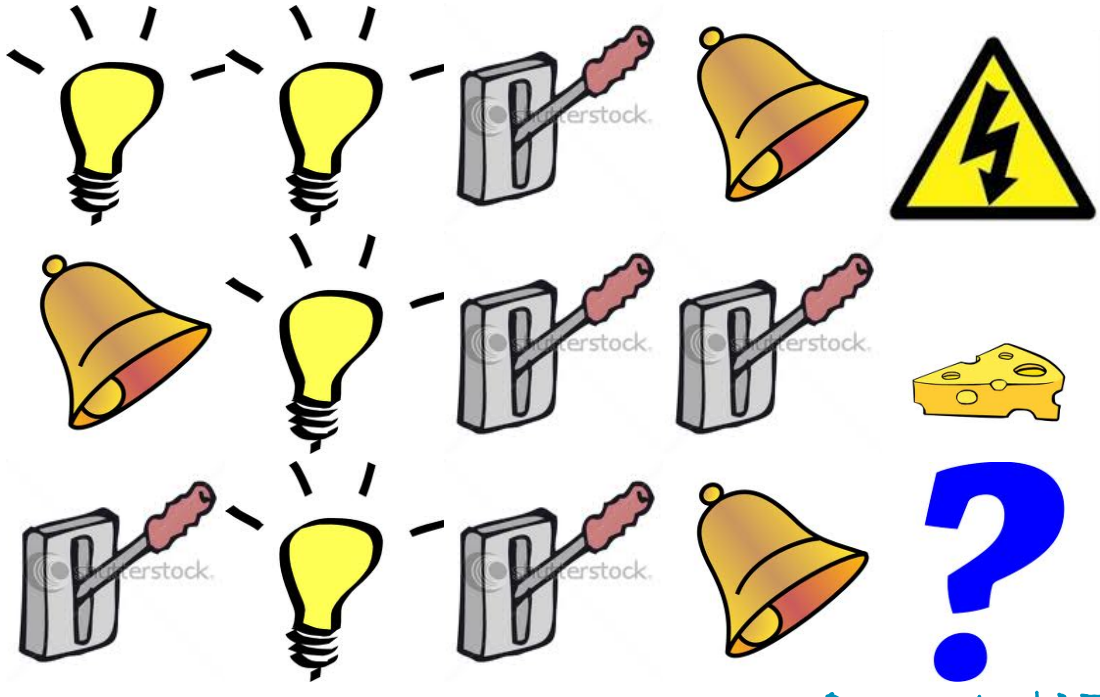
$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

- "The future is independent of the past given the present"

$$H_{1:t} \to S_t \to H_{t+1:\infty}$$

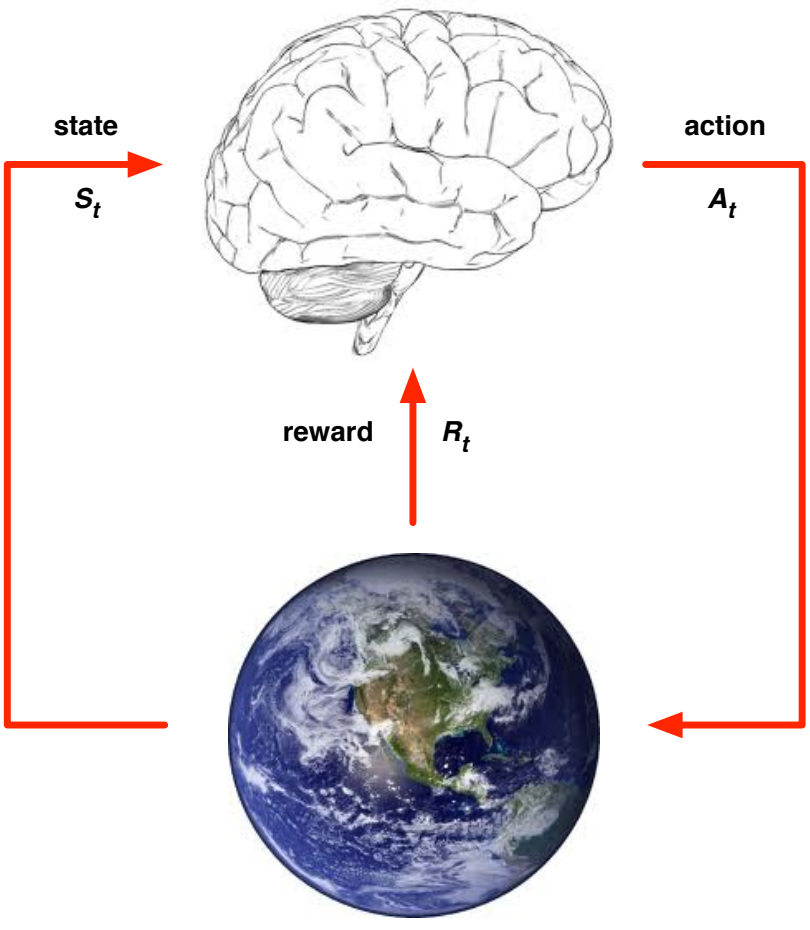
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future
- The environment state S_t^e is Markov
- The history H_t is Markov

Rat Example



- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

Fully Observable Environments



Full observability: agent directly observes environment state

agent state = env — state

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

- Agent state = environment state = information state
- Formally, this is a Markov decision process (MDP)
- (Next lecture and the majority of this course)

Partially Observable Environments

agent가 부분적으로 env-info를 관찰하는 경우.
agent state ≠ env-state

- Partial observability: agent indirectly observes environment:
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards
- Now agent state ≠ environment state
- Formally this is a partially observable Markov decision process (POMDP)
- Agent must construct its own state representation S_t^a, e.g.
  - Complete history: $S_t^a = H_t$
  - Beliefs of environment state: $S_t^a = (\mathbb{P}[S_t^e = s^1], ..., \mathbb{P}[S_t^e = s^n])$
  - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Major Components of an RL Agent

- An RL agent may include one or more of these components:
  - Policy: agent's behaviour function
  - Value function: how good is each state and/or action
  - Model: agent's representation of the environment

① Policy : 어떻게 action 할지

② value function : 그 state, action이 얼마나 좋은지

③ model : agent의 env 표현.

Policy

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

결정론적인 Policy는 action = π(state) π에서 생성한 action 을바로 수행
Policy

확률적인 Policy는 π(action | state) → state s에서 action a가 확률을 따라감.
Policy

Value Function

Value function : future reward 예측.
= expected total reward.

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

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

Policy
expectation
discount factor : 시간이 길어질수록 Reward 영향력 줄어든다
total reward

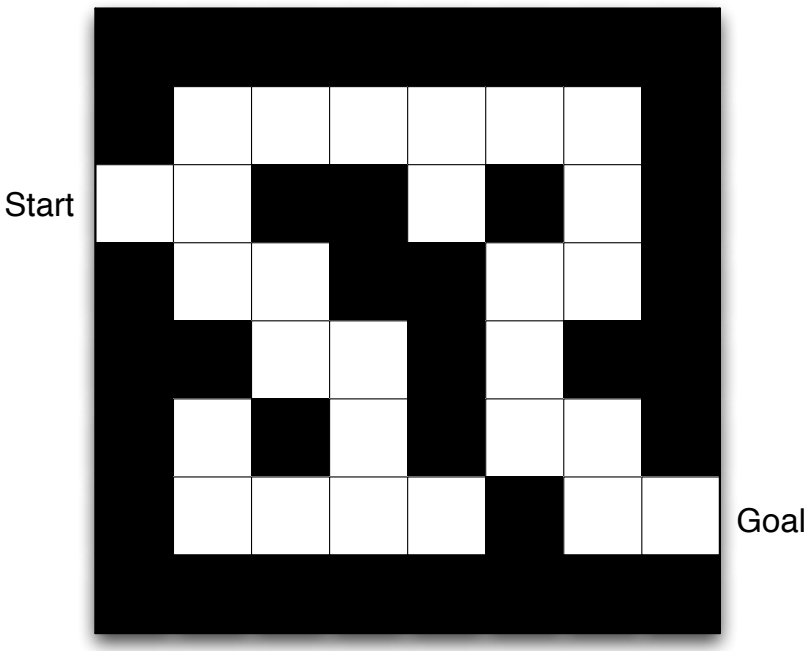
Example: Value Function in Atari

Model

- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state ⇒ Transition
- $\mathcal{R}$ predicts the next (immediate) reward, e.g. ⇒ Reward

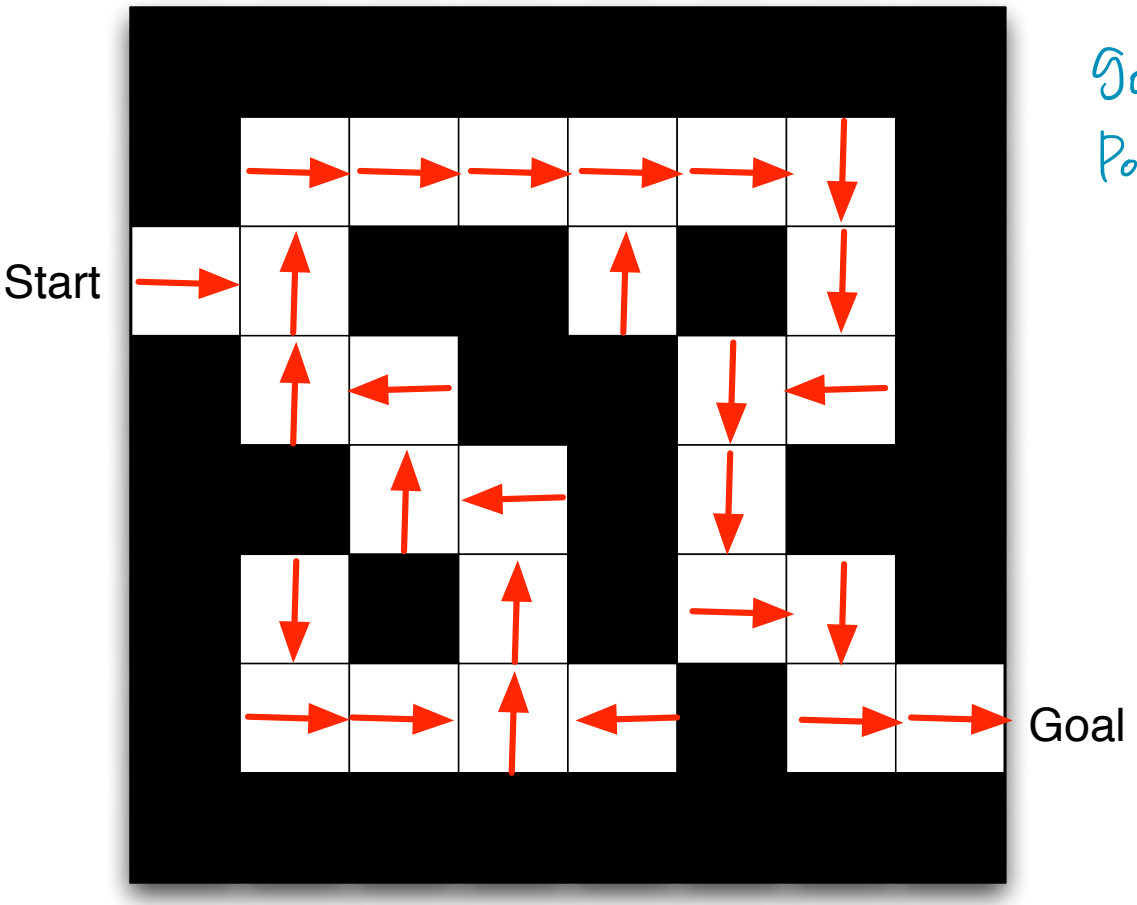
$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

Maze Example



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

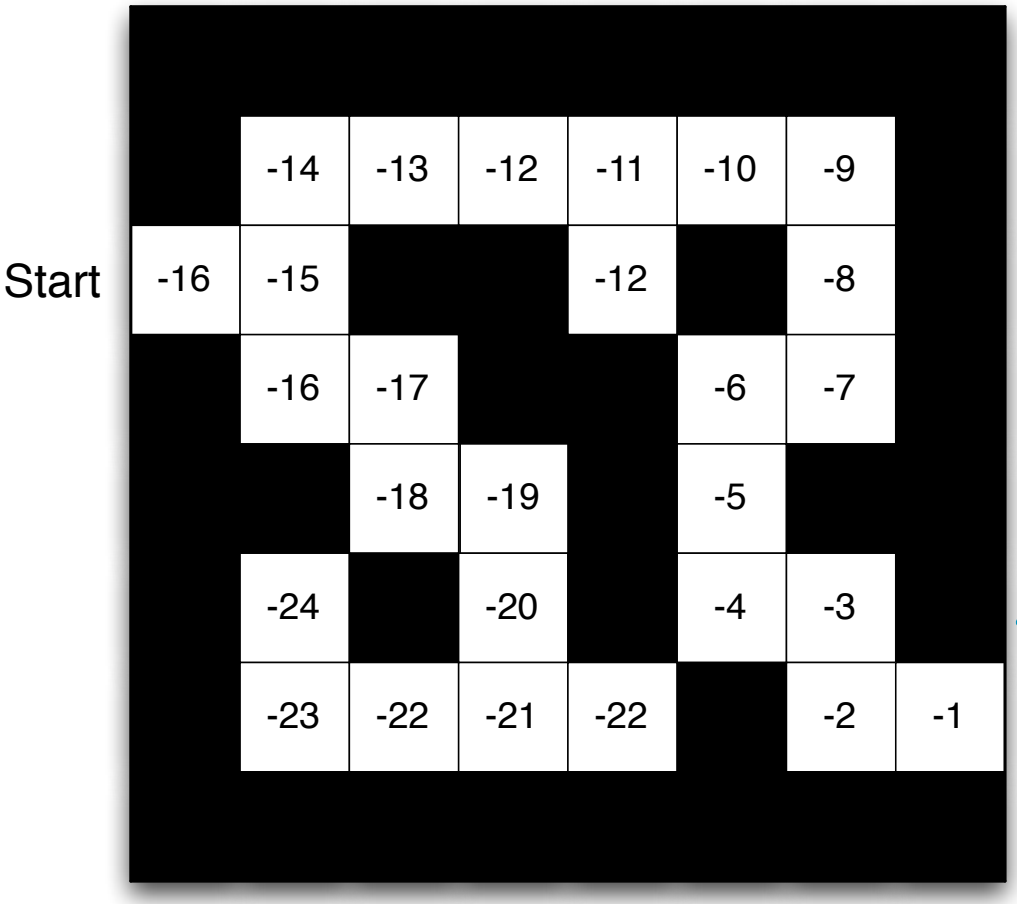
Maze Example: Policy



Goal을 가지는 Policy $\pi(s)$

- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function



Goal까지 받는 expected total reward

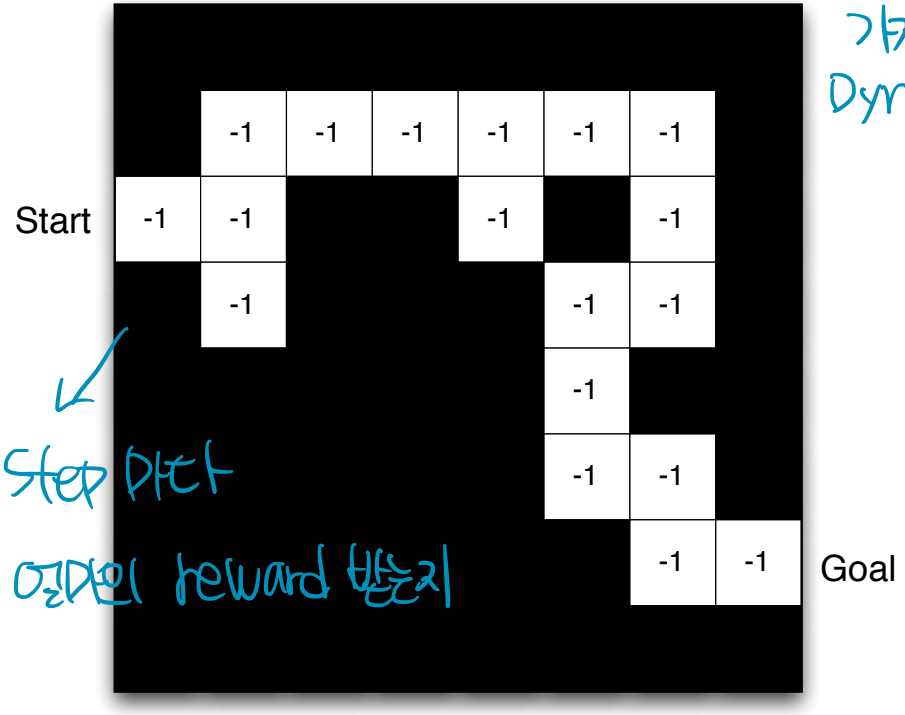
time step당 -1 받으니까

$R_{t+1} + \gamma R_{t+2} + \cdots$

← 각각의 상태에서 이것에 대한 value.

- Numbers represent value $v_\pi(s)$ of each state s

Maze Example: Model



Agent는 내부적으로 env-의 model을 가지고 있을 수임.
Dynamics: 어떻게 action이 state 변화시킬지

- Agent may have an internal model of the environment
- Dynamics: how actions change the state
- Rewards: how much reward from each state
- The model may be imperfect

*model은 불완전할 수임.

각 step 마다 얻게되는 reward 받는지

- Grid layout represents transition model $\mathcal{P}^a_{ss'}$
- Numbers represent immediate reward $\mathcal{R}^a_s$ from each state s (same for all a)

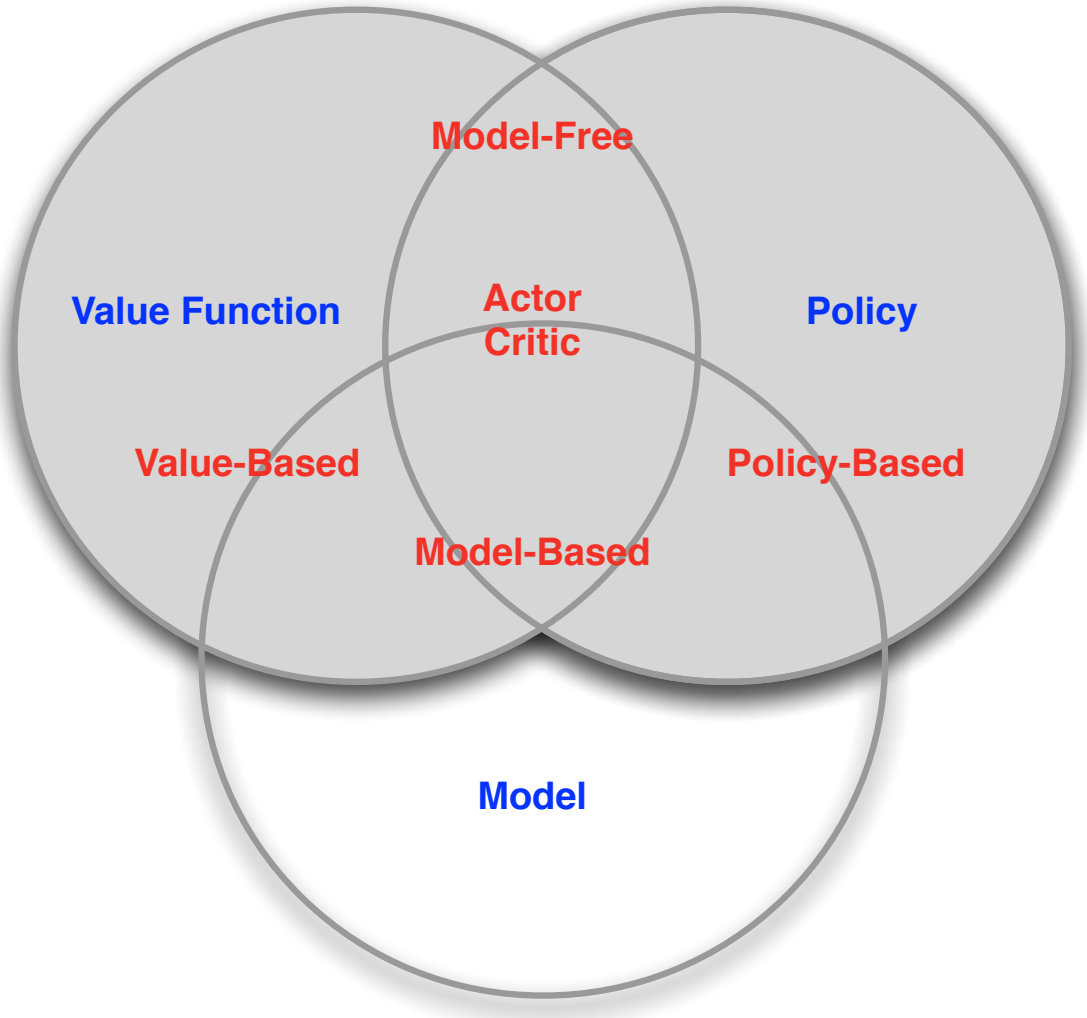
Categorizing RL agents (1)

- Value Based
  - No Policy (Implicit) → Policy 내부적으로 있음.
  - Value Function
- Policy Based
  - Policy
  - No Value Function
- Actor Critic
  - Policy → 두개 결합형태
  - Value Function

Categorizing RL agents (2)

- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

RL Agent Taxonomy



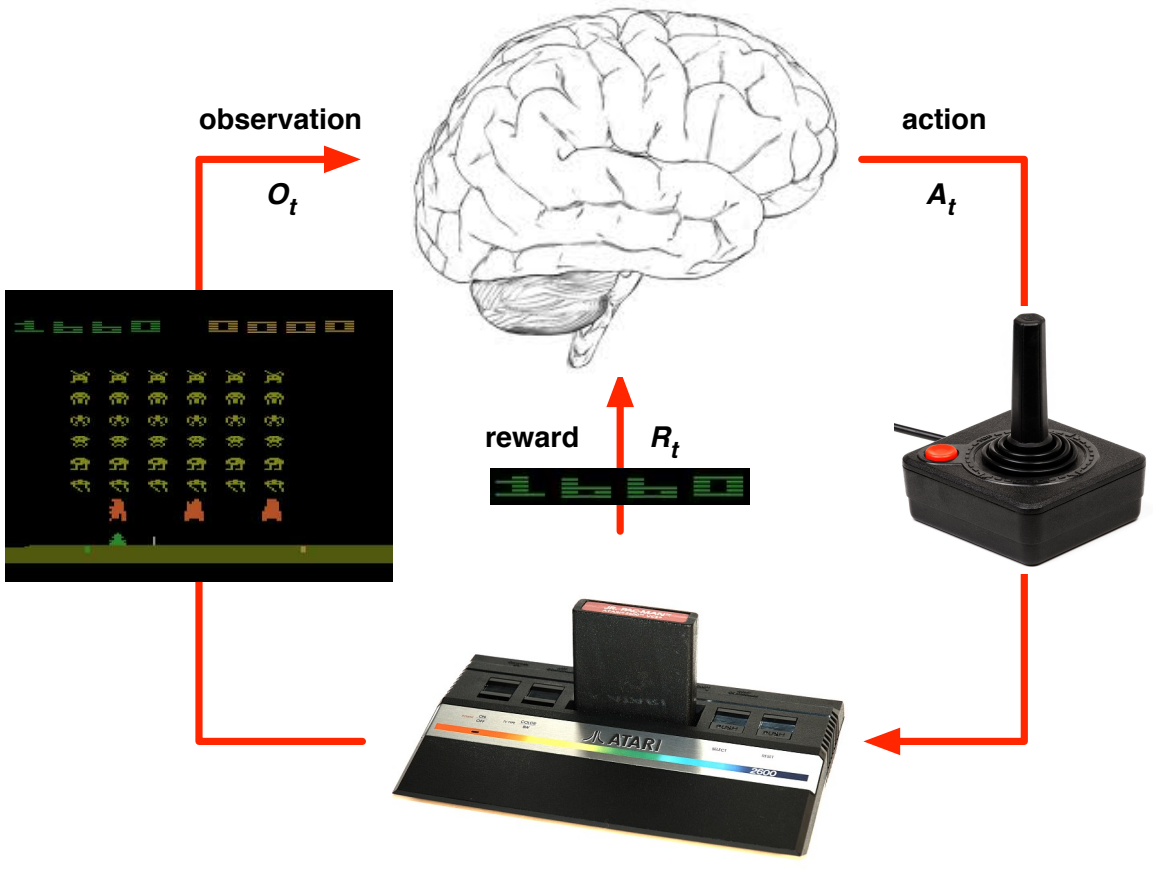
Learning and Planning

Two fundamental problems in sequential decision making

- Reinforcement Learning:
  - The environment is initially unknown
  - The agent interacts with the environment
  - The agent improves its policy
- Planning:
  - A model of the environment is known
  - The agent performs computations with its model (without any external interaction)
  - The agent improves its policy
  - a.k.a. deliberation, reasoning, introspection, pondering, thought, search

env - 처음에 모름
agent가 env-와 상호작용
agent는 policy 개선한다.
env-model을 알고있다.
agent는 model통해서 계산한다.
agent가 policy 개선한다.

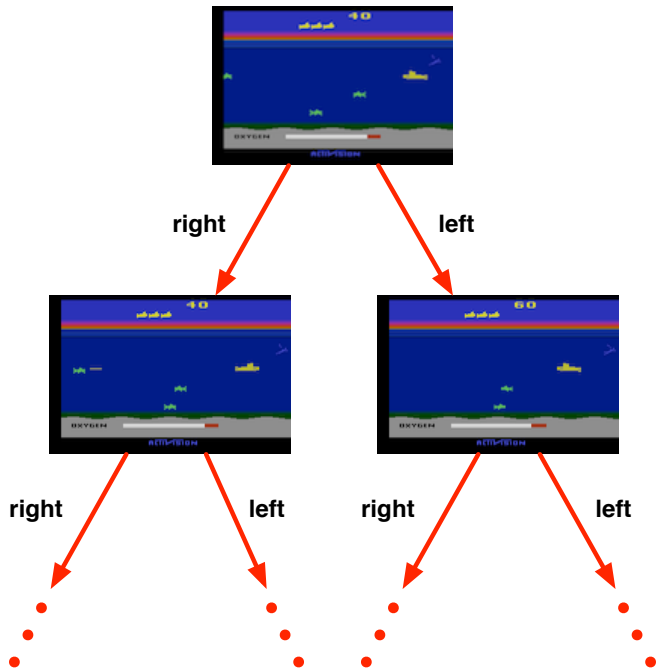
Atari Example: Reinforcement Learning



- Rules of the game are unknown
- Learn directly from interactive game-play
- Pick actions on joystick, see pixels and scores

Atari Example: Planning

- Rules of the game are known
- Can query emulator
  - perfect model inside agent's brain
- If I take action a from state s:
  - what would the next state be?
  - what would the score be?
- Plan ahead to find optimal policy
  - e.g. tree search



Exploration and Exploitation (1)

- Reinforcement learning is like trial-and-error learning
- The agent should discover a good policy
- From its experiences of the environment
- Without losing too much reward along the way

Exploration and Exploitation (2)

exploration : env-에 대해 더 정보찾는가.

exploitation : reward 최대화하기 위해 아는 정보를 이용함.

- *Exploration* finds more information about the environment
- *Exploitation* exploits known information to maximise reward
- It is usually important to explore as well as exploit

Examples

- Restaurant Selection
  - Exploitation Go to your favourite restaurant
  - Exploration Try a new restaurant
- Online Banner Advertisements
  - Exploitation Show the most successful advert
  - Exploration Show a different advert
- Oil Drilling
  - Exploitation Drill at the best known location
  - Exploration Drill at a new location
- Game Playing
  - Exploitation Play the move you believe is best
  - Exploration Play an experimental move

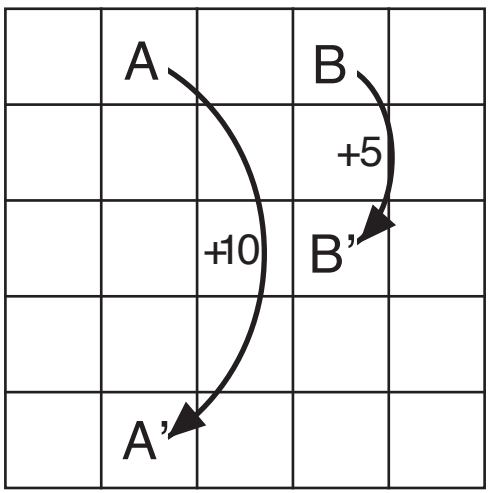
Prediction and Control

- Prediction: evaluate the future
  - Given a policy
- Control: optimise the future
  - Find the best policy

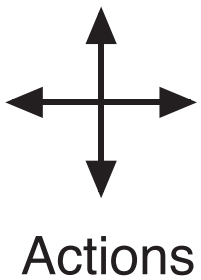
Prediction
- Policy 주고 얼마나 좋은지 평가하는것.

Control
- 최적의 policy 찾기.

Gridworld Example: Prediction



(a)

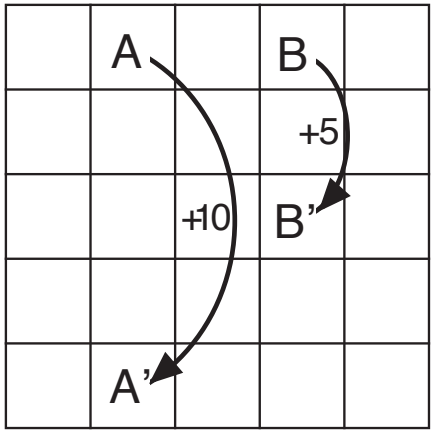


3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

What is the value function for the uniform random policy?

Gridworld Example: Control

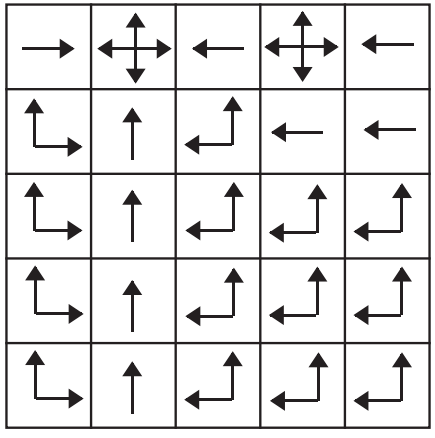


a) gridworld

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 fn



c) π_*

optimal policy

What is the optimal value function over all possible policies?
What is the optimal policy?

Course Outline

- Part I: Elementary Reinforcement Learning
  1. Introduction to RL
  2. Markov Decision Processes
  3. Planning by Dynamic Programming
  4. Model-Free Prediction
  5. Model-Free Control
- Part II: Reinforcement Learning in Practice
  1. Value Function Approximation
  2. Policy Gradient Methods
  3. Integrating Learning and Planning
  4. Exploration and Exploitation
  5. Case study - RL in games