Introduction to (Deep) Reinforcement Learning

Insoon Yang

Department of Electrical and Computer Engineering
Seoul National University

CORE
Control + Optimization Research Lab

How do we build intelligent systems/machines?

- Data $\underbrace{\rightarrow}_{\text{intelligence}}$ Information
- Information $\underbrace{\rightarrow}_{\text{intelligence}}$ Decision

강화학습: intelligent Decsion making framework

Intelligent systems must be able to adapt to uncertainty

Uncertainty:

Intelligent systems must be able to adapt to uncertainty

Uncertainty:

- Other cars

Intelligent systems must be able to adapt to uncertainty

Uncertainty:

- Other cars
- Weather

Intelligent systems must be able to adapt to uncertainty

Uncertainty:

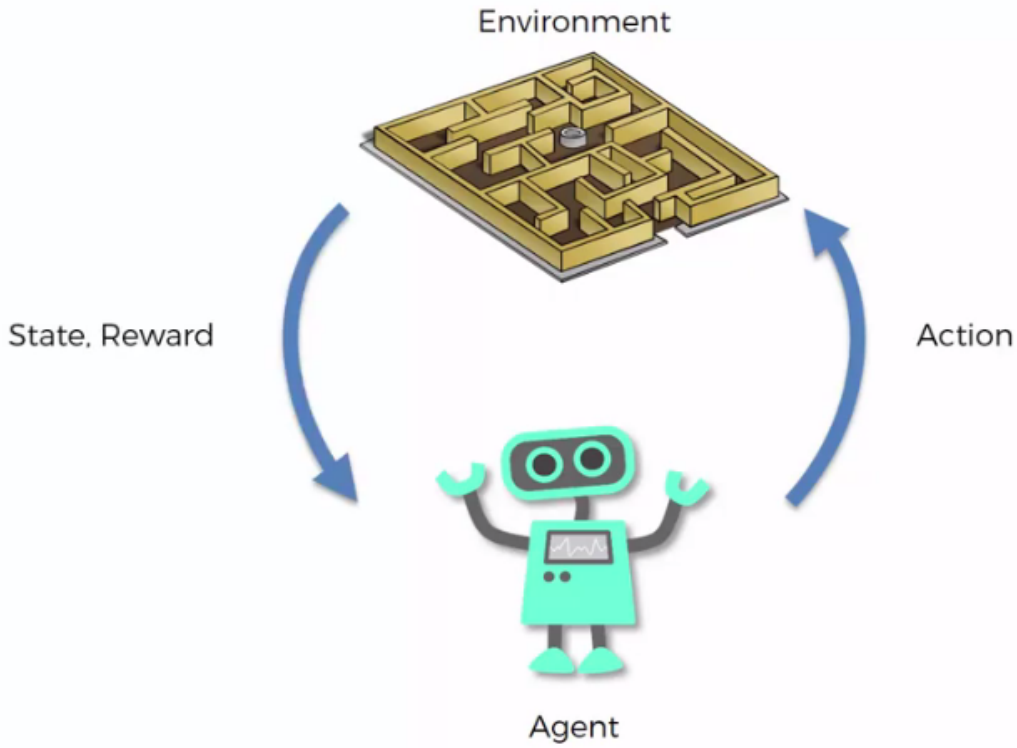
- Other cars
- Weather
- Road construction

Intelligent systems must be able to adapt to uncertainty

Uncertainty:

- Other cars
- Weather
- Road construction
- Passenger

RL provides a formalism for behaviors

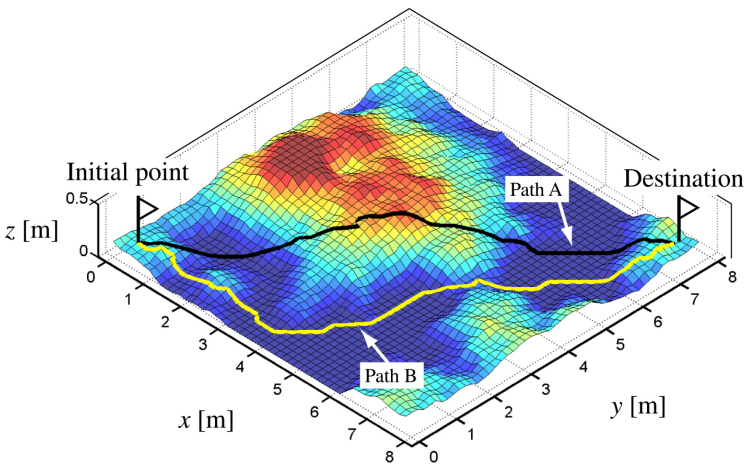


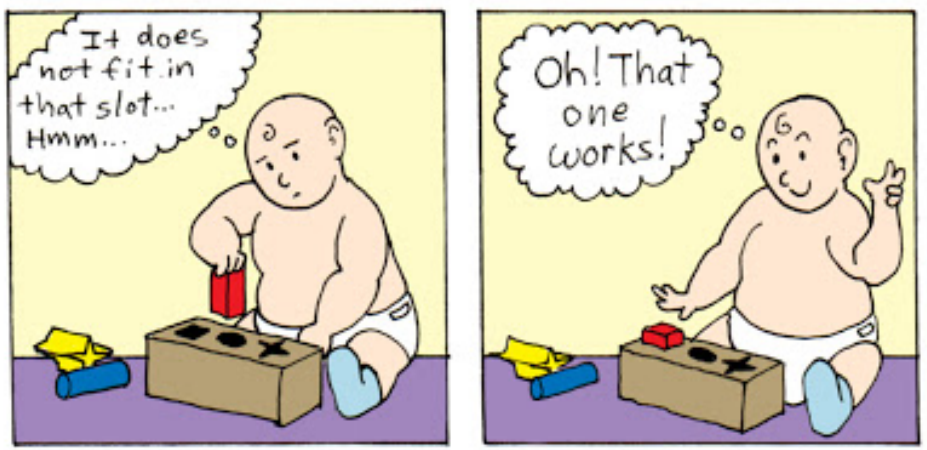
- "Problem of a goal-directed agent $\underbrace{\textit{interacting with}}_{\text{feedback \& decision}}$ an uncertain environment"
- Interaction $\Longrightarrow$ adaptation

Brief History: Roots of RL

Two threads:

1. Optimal Control
2. Psychology (trial-and-error learning)





Thread I: Optimal Control

"Problem of designing a controller to maximize the performance of a dynamical system's behavior over time"

- 1950's: Dynamic programming (Richard Bellman)
- Two limitations:
  1. Knowledge about models
  2. Scalability: "Curse of dimensionality"

Thread I: Optimal Control

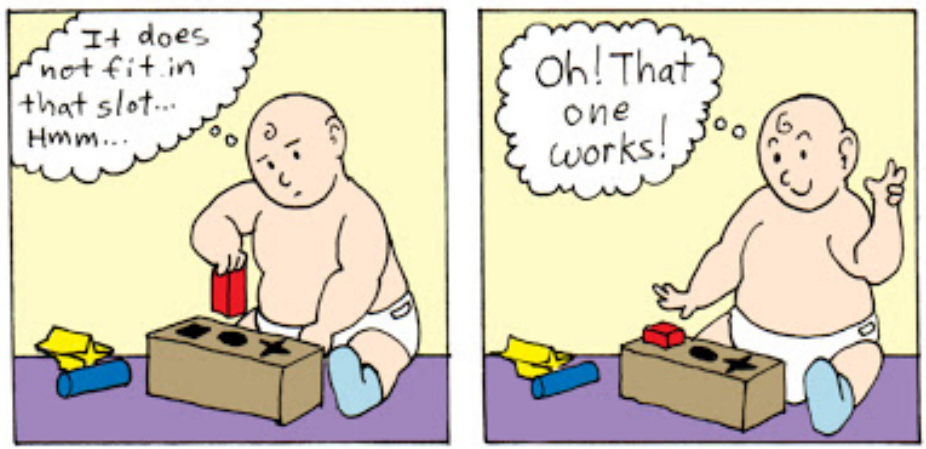
"Problem of designing a controller to maximize the performance of a dynamical system's behavior over time"

- 1950's: Dynamic programming (Richard Bellman)
- Two limitations:
  1. Knowledge about models
  2. Scalability: "Curse of dimensionality"

$\Longrightarrow$ RL aims to resolve these issues in (stochastic) optimal control

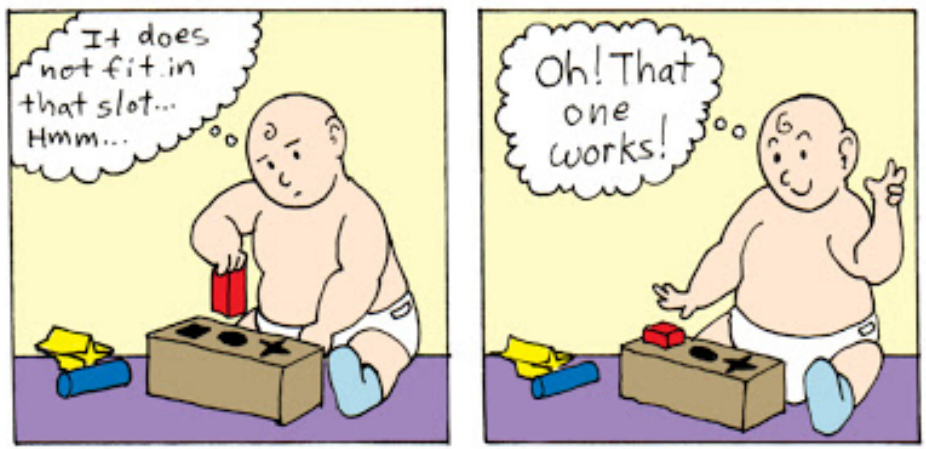
Thread II: Psychology (trial-and-error learning)

- Edward Thorndike (1911) "Law of Effect": Of several responses made to the same situation, those which are accompanied by satisfaction to the animal will be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied by discomfort to the animal will have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur.



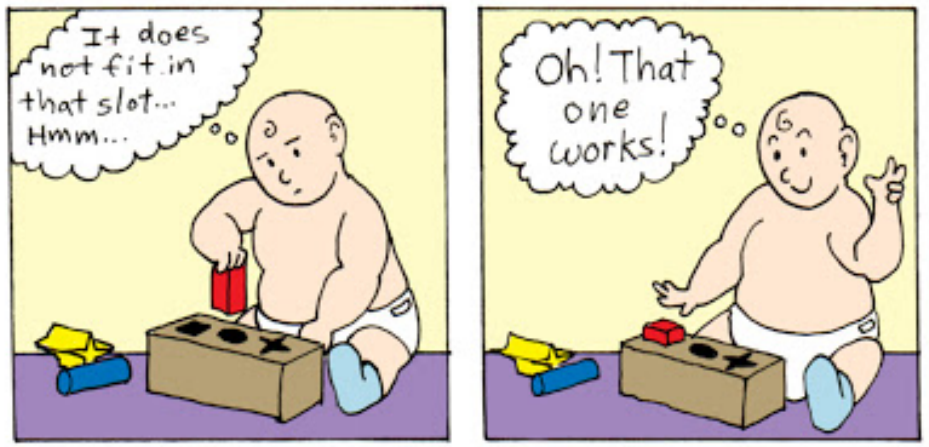
Thread II: Psychology (trial-and-error learning)

- Edward Thorndike (1911) "Law of Effect": Of several responses made to the same situation, those which are accompanied by satisfaction to the animal will be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied by discomfort to the animal will have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur.



1. Selectional (search): trying alternatives and selecting among them by comparing their consequences
2. Associative (memory): the alternatives found by selection are associated with particular situations

Thread II: Psychology (trial-and-error learning)



- Klopf (1972) linked the idea with trial-and-error learning and related it to the massive empirical database of animal learning psychology
- Sutton and Barto (70's-80's): temporal-difference learning

Bringing the two together: RL

"Trial-and-error learning approach to optimal control"

Bringing the two together: RL

"Trial-and-error learning approach to optimal control"

- Q-Learning: Chris Watkins (1989)

Bringing the two together: RL

"Trial-and-error learning approach to optimal control"

- Q-Learning: Chris Watkins (1989)
- Policy gradient: Williams (1987), Marbach & Tsitsiklis (1998)

Bringing the two together: RL

"Trial-and-error learning approach to optimal control"

- Q-Learning: Chris Watkins (1989)
- Policy gradient: Williams (1987), Marbach & Tsitsiklis (1998)
- Actor-critic: Witten, Barto, Sutton, Degris, White, etc.

Bringing the two together: RL

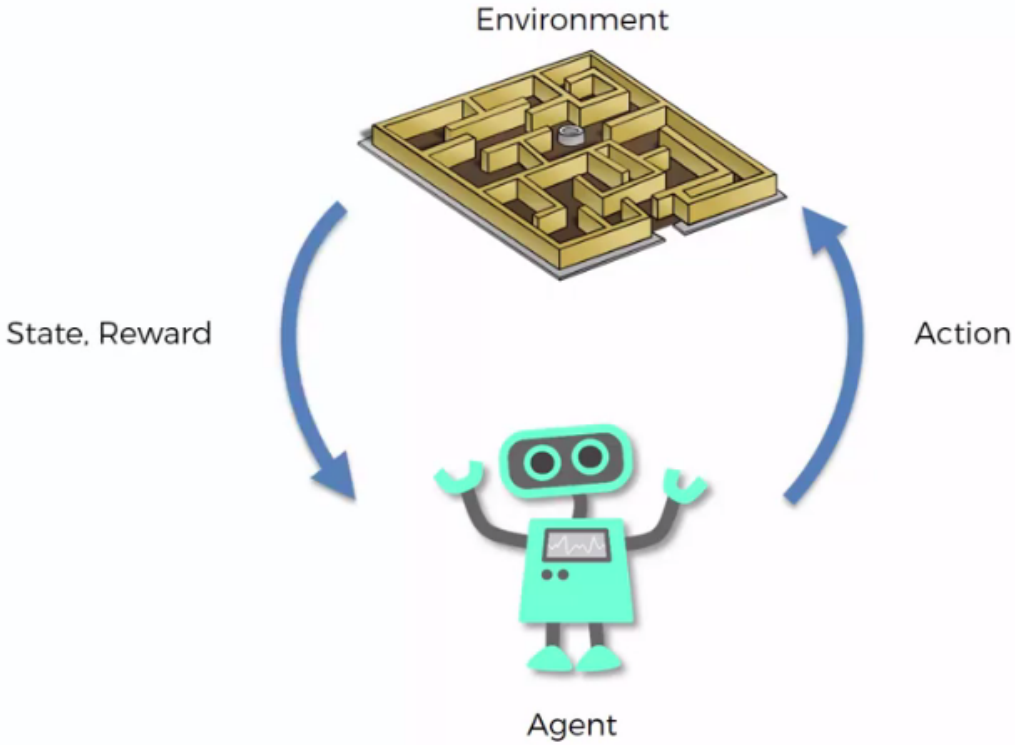
"Trial-and-error learning approach to optimal control"

- Q-Learning: Chris Watkins (1989)
- Policy gradient: Williams (1987), Marbach & Tsitsiklis (1998)
- Actor-critic: Witten, Barto, Sutton, Degris, White, etc.
- Applications ($\sim$2010)
  - Game: Tetris, Chess
  - Robotics

Application I: Games

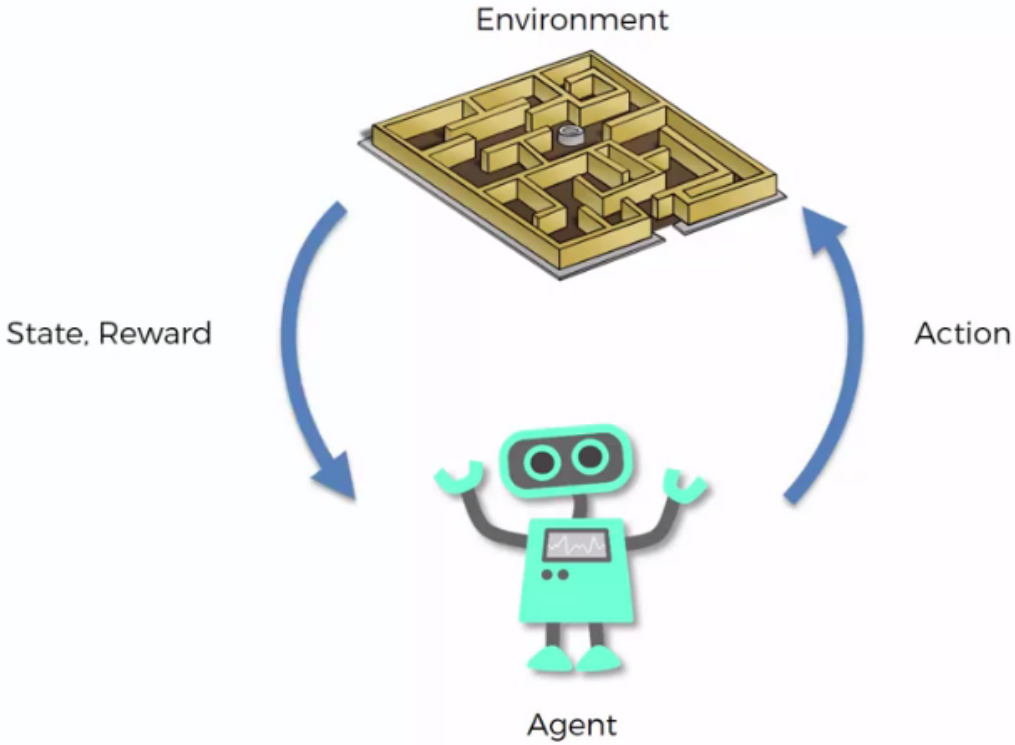
Application II: Robotics

Distinct Features of RL



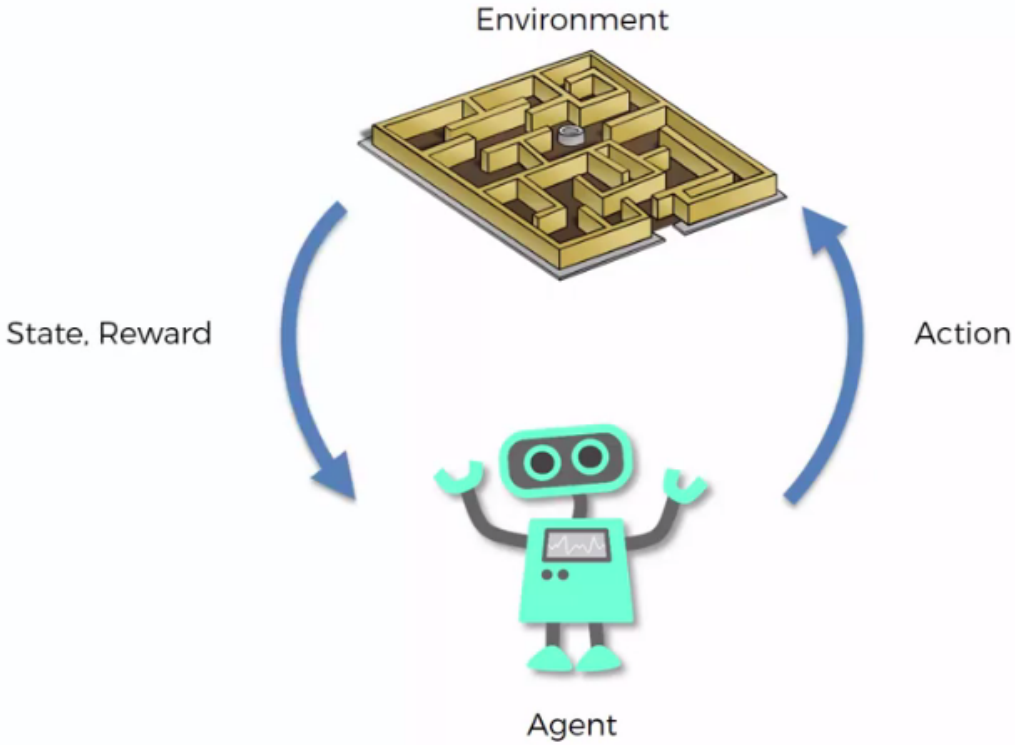
1. There is no supervisor, only a reward signal

Distinct Features of RL



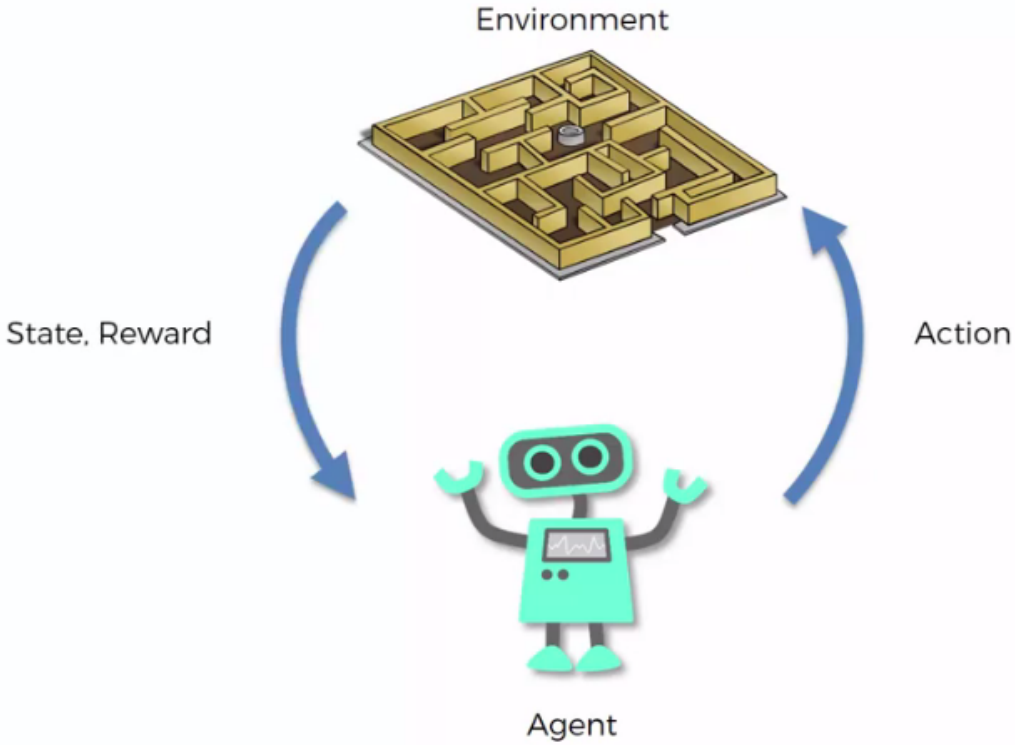
1. There is no supervisor, only a reward signal
2. Feedback is delayed, not instantaneous

Distinct Features of RL



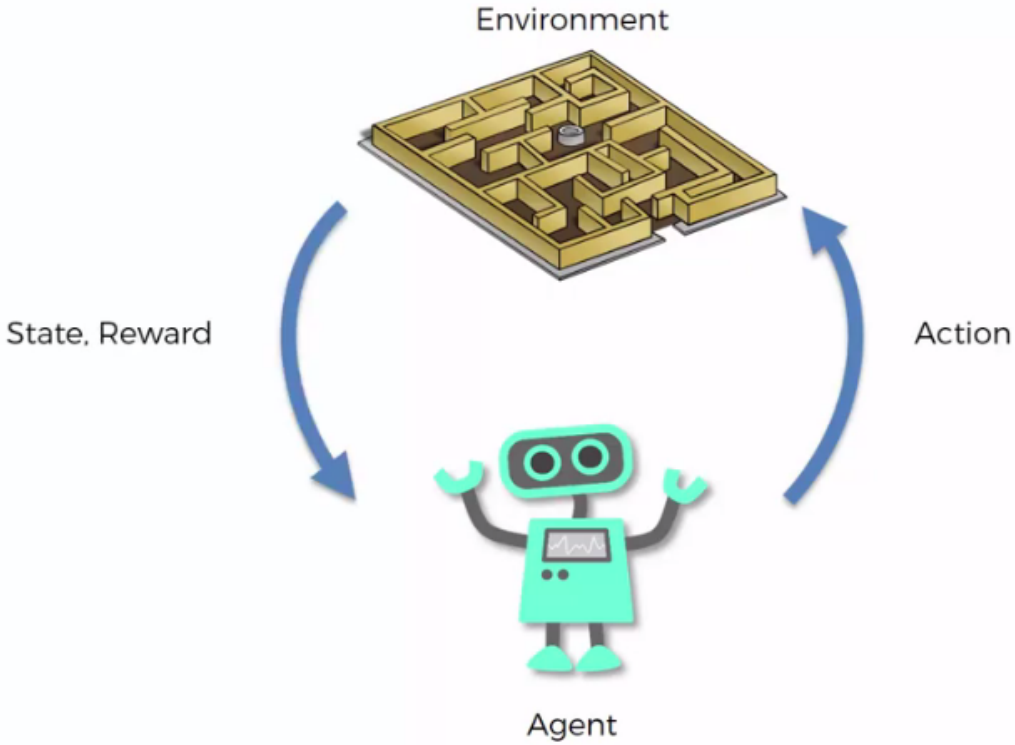
1. There is no supervisor, only a reward signal
2. Feedback is delayed, not instantaneous
3. Time really matters (sequential, non i.i.d data)

Distinct Features of RL



1. There is no supervisor, only a reward signal
2. Feedback is delayed, not instantaneous
3. Time really matters (sequential, non i.i.d data)
4. Agent's actions affect the subsequent data it receives

Distinct Features of RL

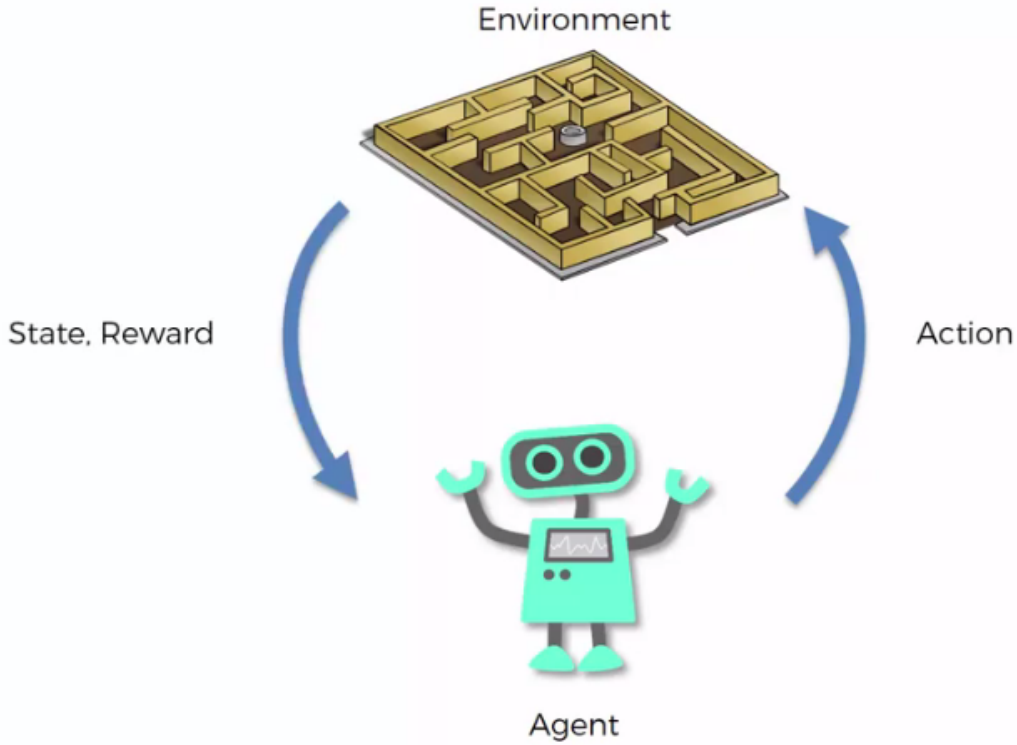


1. There is no supervisor, only a reward signal
2. Feedback is delayed, not instantaneous
3. Time really matters (sequential, non i.i.d data)
4. Agent's actions affect the subsequent data it receives
5. Tradeoff between exploration and exploitation

탐색결과에 대한 이득

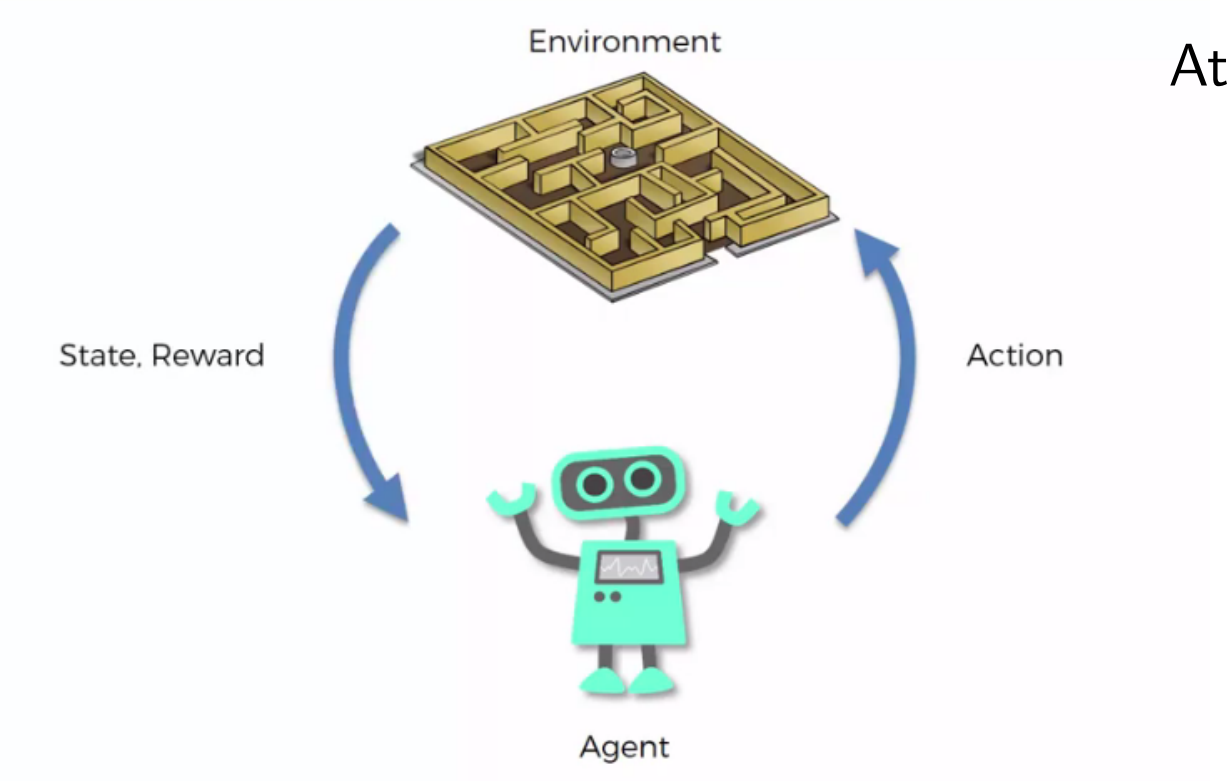
눈앞의 이득

Components of the RL Problem



- State of the environment/system
- Action determined by the agent: affects the state
- Reward: evaluates the benefit of state and action

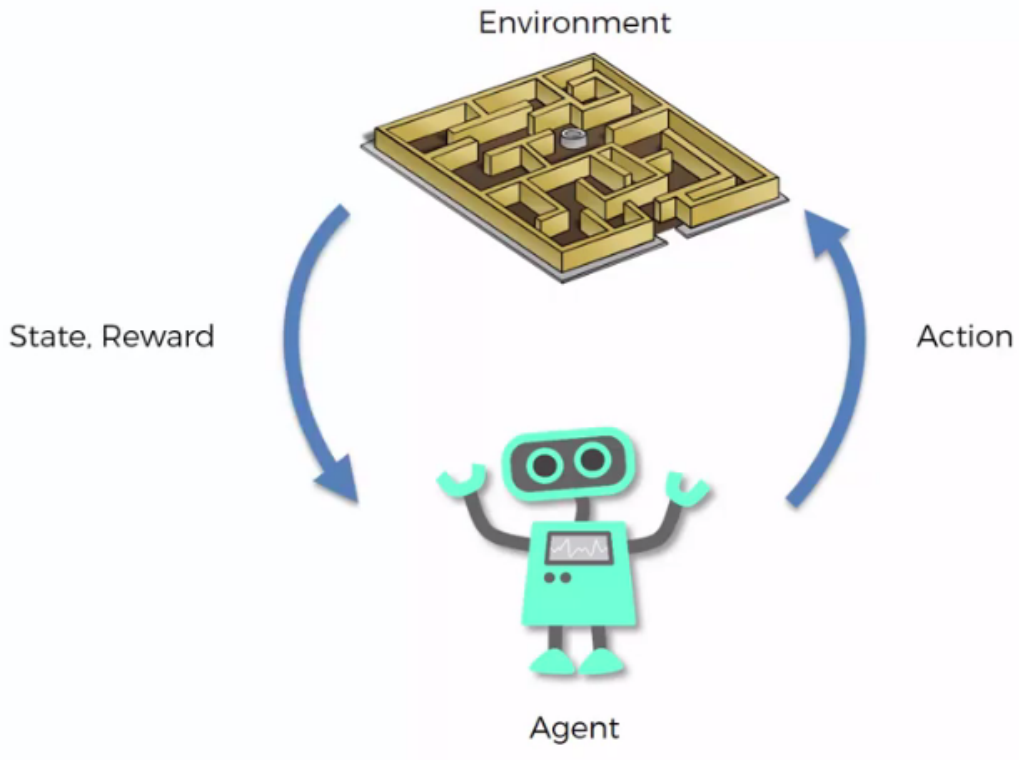
Agent and Environment



At each time step t:

- Agent

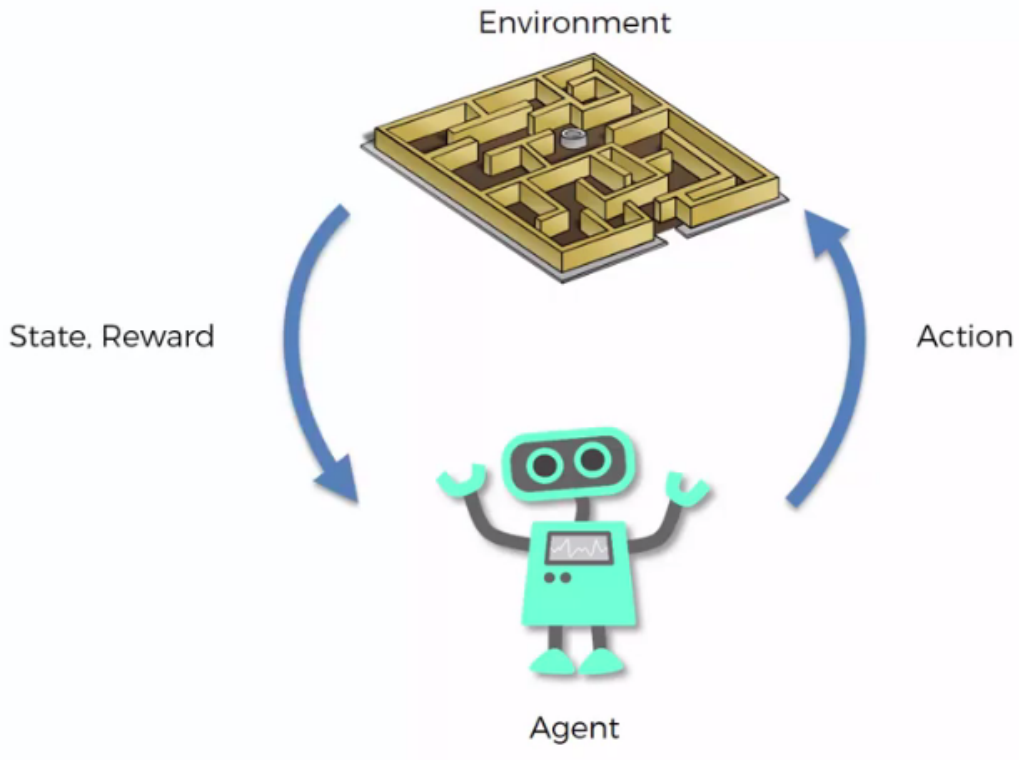
Agent and Environment



At each time step t:

- Agent
  1. observes state s_t

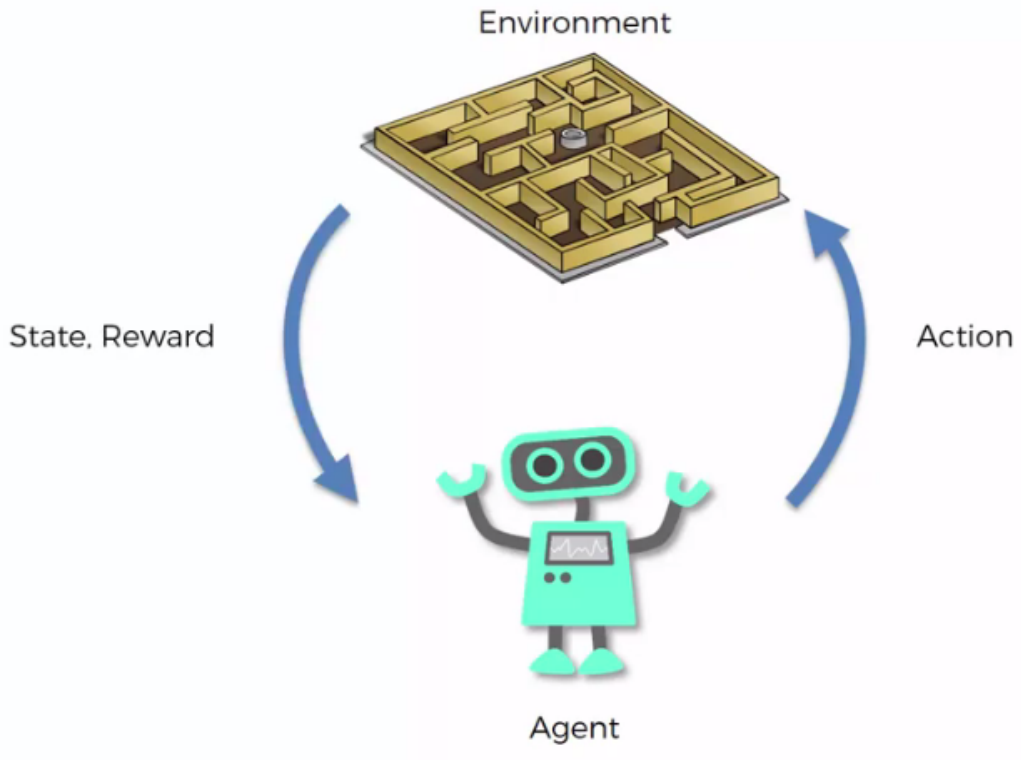
Agent and Environment



At each time step t:

- Agent
  1. observes state s_t
  2. executes action a_t

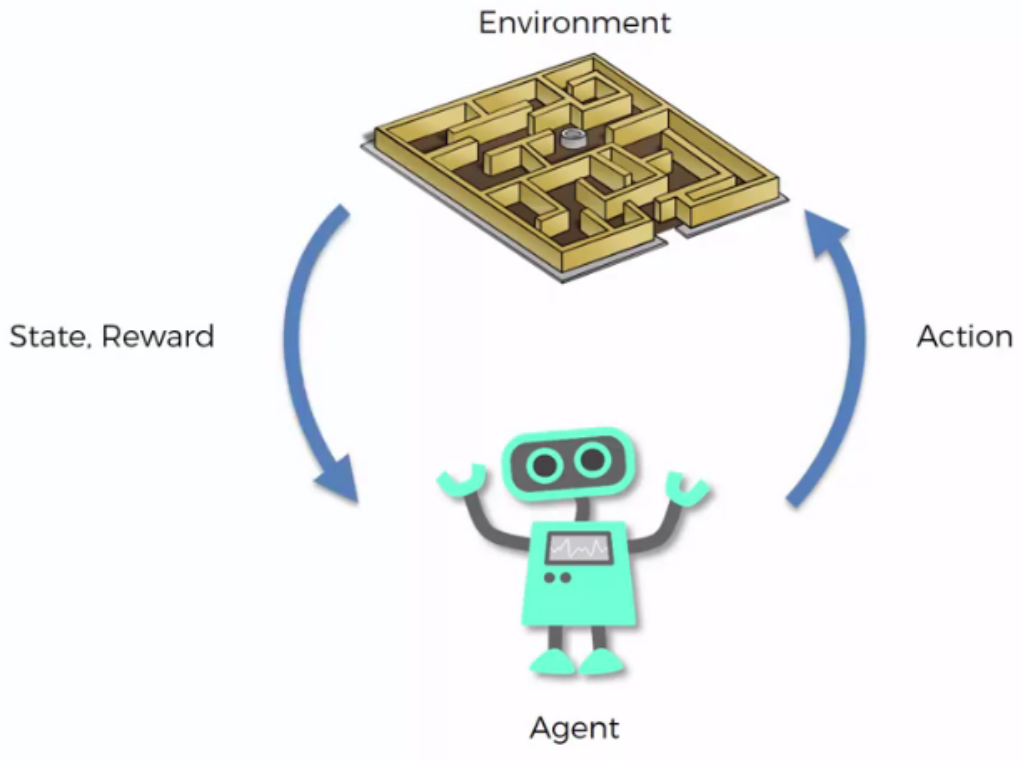
Agent and Environment



At each time step t:

- Agent
  1. observes state s_t
  2. executes action a_t
  3. receives reward r_t

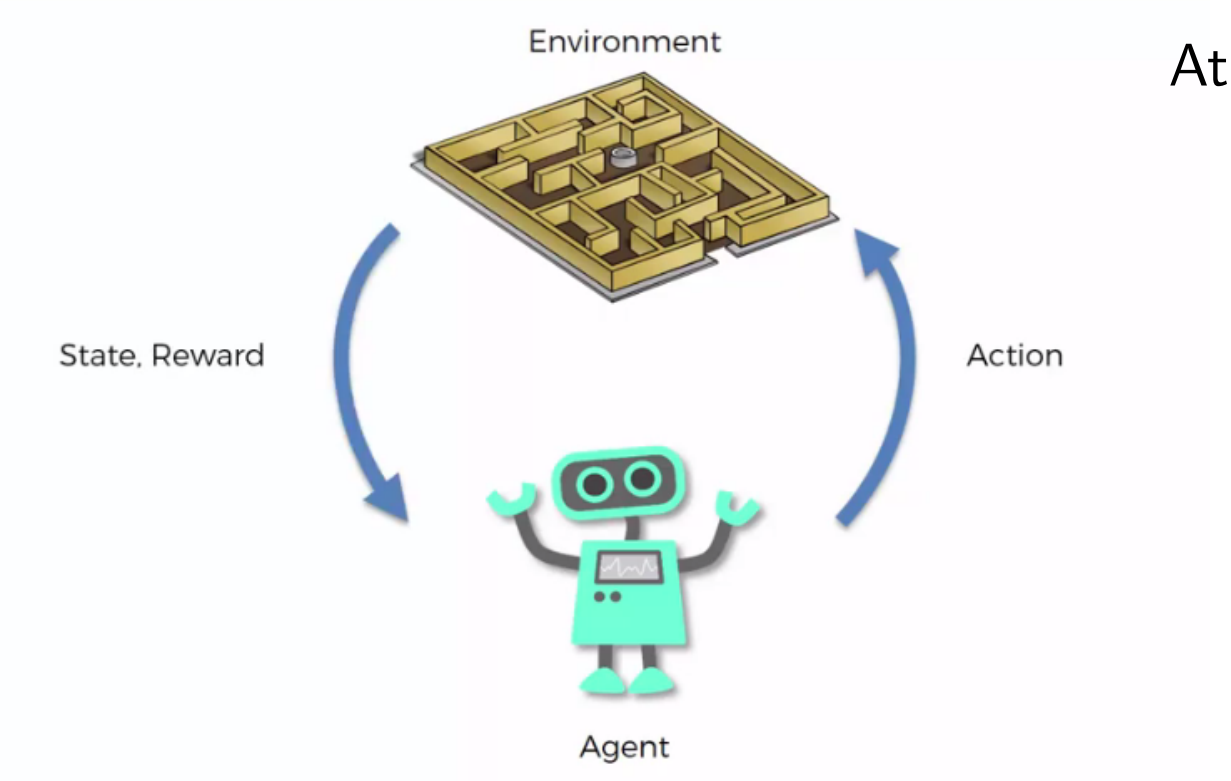
Agent and Environment



At each time step t:

- Agent
  1. observes state s_t
  2. executes action a_t
  3. receives reward r_t
- Environment

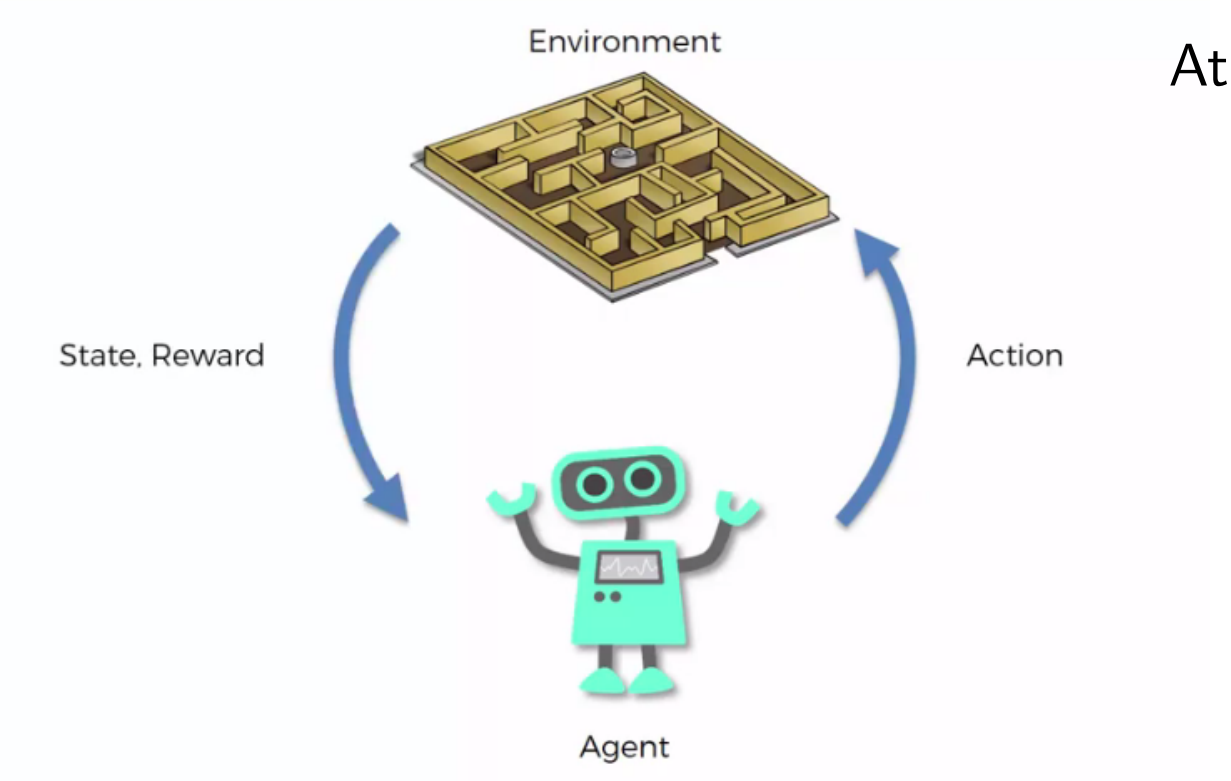
Agent and Environment



At each time step t:

- Agent
  1. observes state s_t
  2. executes action a_t
  3. receives reward r_t
- Environment
  1. receives action a_t

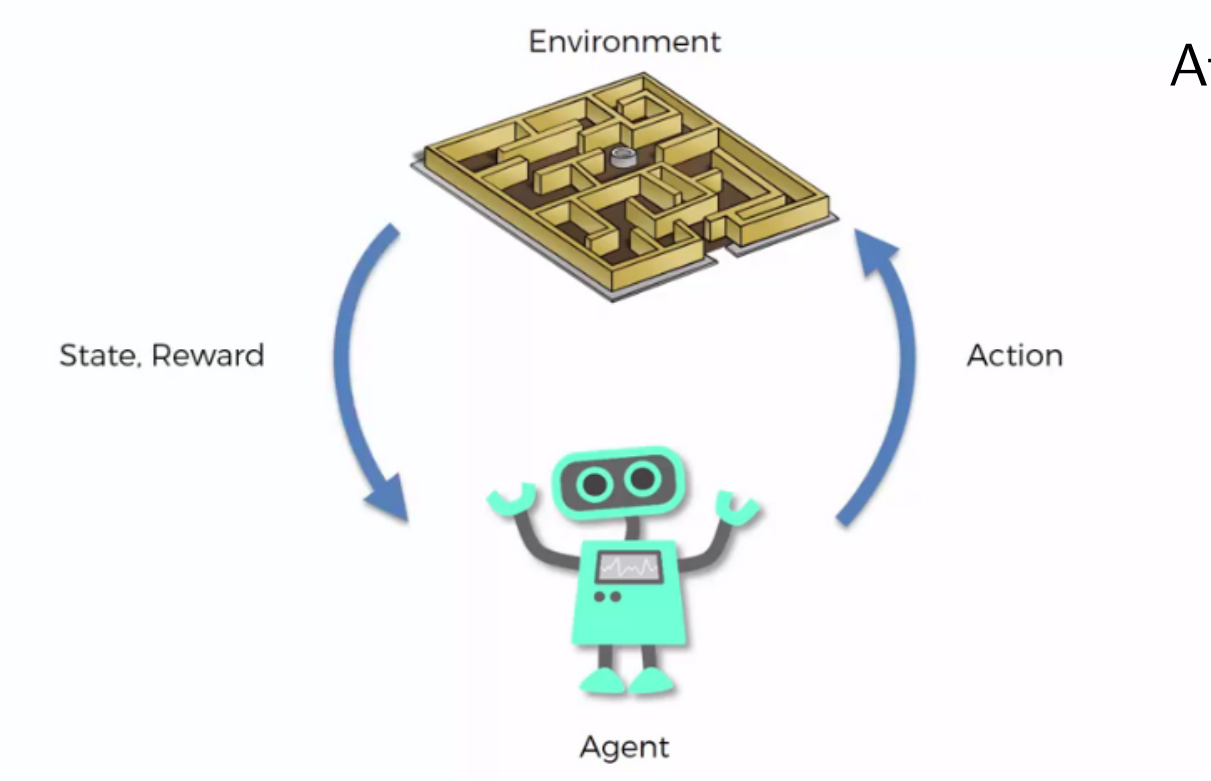
Agent and Environment



At each time step t:

- Agent
  1. observes state s_t
  2. executes action a_t
  3. receives reward r_t
- Environment
  1. receives action a_t
  2. emits reward r_t

Agent and Environment



At each time step t:

- Agent
  1. observes state s_t
  2. executes action a_t
  3. receives reward r_t
- Environment
  1. receives action a_t
  2. emits reward r_t
  3. updates state to s_{t+1}

Rewards

- Reward $r_t \in \mathbb{R}$ is a scalar feedback signal

Rewards

- Reward $r_t \in \mathbb{R}$ is a scalar feedback signal
- Indicates how well the agent is doing at step t

Rewards

- Reward $r_t \in \mathbb{R}$ is a scalar feedback signal
- Indicates how well the agent is doing at step t
- The agent's job is to maximize the expected cumulative reward

Rewards

- Reward $r_t \in \mathbb{R}$ is a scalar feedback signal
- Indicates how well the agent is doing at step t
- The agent's job is to maximize the expected cumulative reward

Assumptions

All goals can be described by the maximization of expected cumulative rewards.

Examples of Rewards

1. Control of a drone
   - $+$ reward for following desired trajectory
   - $-$ reward for crashing

Examples of Rewards

1. Control of a drone
   - $+$ reward for following desired trajectory
   - $-$ reward for crashing
2. Control of a humanoid robot to walk
   - $+$ reward for forward walking
   - $-$ reward for falling over

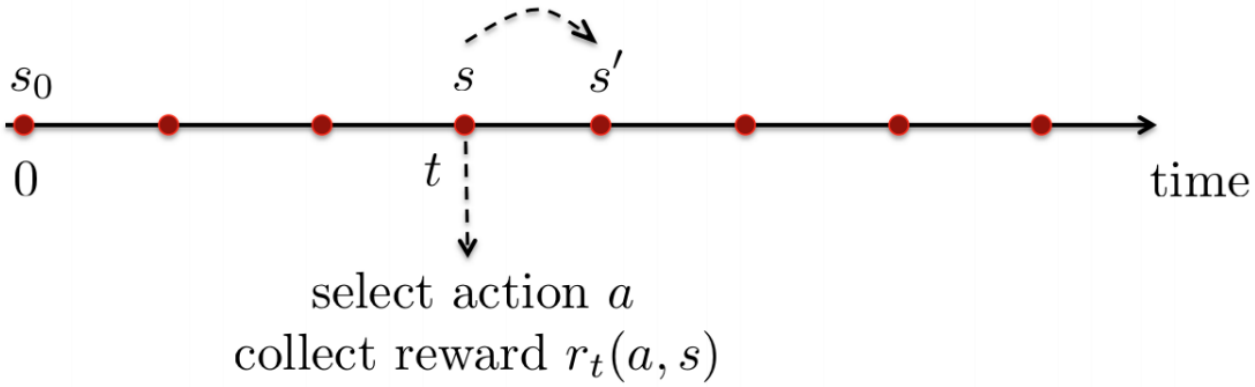
Examples of Rewards

1. Control of a drone
   - $+$ reward for following desired trajectory
   - $-$ reward for crashing
2. Control of a humanoid robot to walk
   - $+$ reward for forward walking
   - $-$ reward for falling over
3. Portfolio management
   - $+$ reward for earning money
   - $-$ reward for losing money

Examples of Rewards

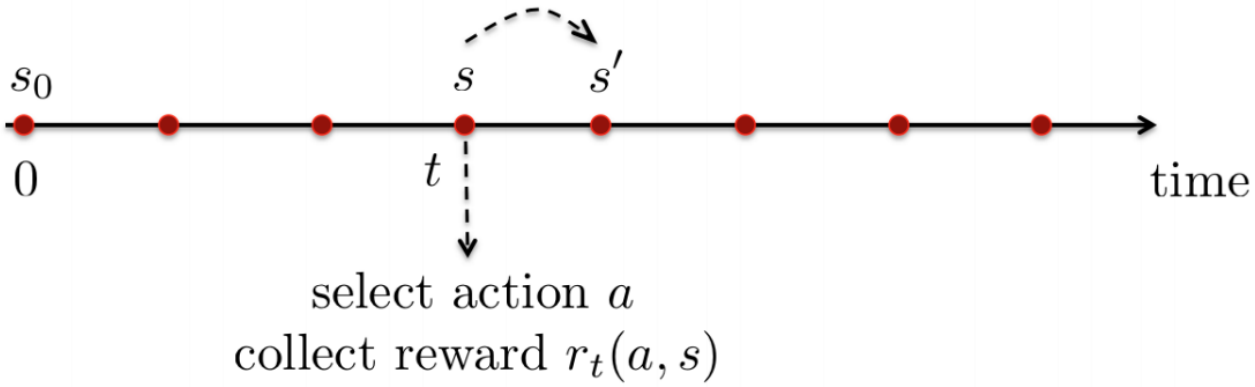
1. Control of a drone
   - $+$ reward for following desired trajectory
   - $-$ reward for crashing
2. Control of a humanoid robot to walk
   - $+$ reward for forward walking
   - $-$ reward for falling over
3. Portfolio management
   - $+$ reward for earning money
   - $-$ reward for losing money
4. (Computer) Games
   - $+$ reward for increasing score
   - $-$ reward for decreasing score

Sequential Decision Making



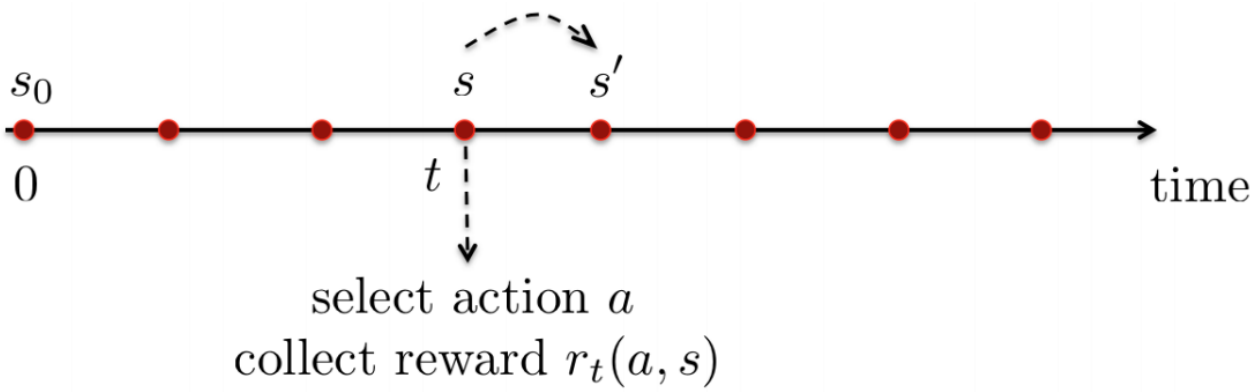
- Goal: select actions **over time** to maximize the expected cumulative reward

Sequential Decision Making



- Goal: select actions **over time** to maximize the expected cumulative reward
- Actions may have long term effects. *Q) Why?*

Sequential Decision Making



- Goal: select actions **over time** to maximize the expected cumulative reward
- Actions may have long term effects. *Q) Why?*
- It may be better to sacrifice immediate reward to gain more long-term reward. *Q) How?*

State, Action, and Policy

- State: $s_t \in S$
- Action: $a_t \in A$

State, Action, and Policy

- State: $s_t \in S$
- Action: $a_t \in A$
- **Policy** (mapping from state to action): π

State, Action, and Policy

- State: $s_t \in S$
- Action: $a_t \in A$
- **Policy** (mapping from state to action): π
  - Deterministic

$$\pi(s_t) = a_t$$

State, Action, and Policy

- State: $s_t \in S$
- Action: $a_t \in A$
- **Policy** (mapping from state to action): π
  - Deterministic

$$\pi(s_t) = a_t$$

  - Stochastic (randomized)

$$\pi(a|s) = \text{Prob}(a_t = a|s_t = s)$$

state는 불확실성을 포함한다

State, Action, and Policy

- State: $s_t \in S$
- Action: $a_t \in A$
- **Policy** (mapping from state to action): π
  - Deterministic

$$\pi(s_t) = a_t$$

  - Stochastic (randomized)

$$\pi(a|s) = \text{Prob}(a_t = a|s_t = s)$$

state는 불확실성을 포함한다

$\Longrightarrow$ **Policy is what we want to optimize!**

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

- S: set of states (state space)

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

- S: set of states (state space)
  e.g., $S = \{1, \ldots, n\}$ (discrete), $S = \mathbb{R}^n$ (continuous)

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

- S: set of states (state space)
  e.g., $S = \{1, \ldots, n\}$ (discrete), $S = \mathbb{R}^n$ (continuous)
- A: set of actions (action space)

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

- S: set of states (state space)
  e.g., $S = \{1, \ldots, n\}$ (discrete), $S = \mathbb{R}^n$ (continuous)
- A: set of actions (action space)
  e.g., $A = \{1, \ldots, m\}$ (discrete), $A = \mathbb{R}^m$ (continuous)

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

- S: set of states (state space)
  e.g., $S = \{1, \ldots, n\}$ (discrete), $S = \mathbb{R}^n$ (continuous)
- A: set of actions (action space)
  e.g., $A = \{1, \ldots, m\}$ (discrete), $A = \mathbb{R}^m$ (continuous)
- p: state transition probability
  $p(s'|s, a) := \mathrm{Prob}(s_{t+1} = s'|s_t = s, a_t = a)$

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

- S: set of states (state space)
  e.g., $S = \{1, \ldots, n\}$ (discrete), $S = \mathbb{R}^n$ (continuous)
- A: set of actions (action space)
  e.g., $A = \{1, \ldots, m\}$ (discrete), $A = \mathbb{R}^m$ (continuous)
- p: state transition probability
  $p(s'|s, a) := \text{Prob}(s_{t+1} = s'|s_t = s, a_t = a)$
- r: reward function
  $r(s_t, a_t) = r_t$

Markov Decision Processes (MDPs)

Definition

A Markov decision process (MDP) is a tuple $\langle S, A, P, R, \gamma \rangle$, consisting of

- S: set of states (state space)
  e.g., $S = \{1, \ldots, n\}$ (discrete), $S = \mathbb{R}^n$ (continuous)
- A: set of actions (action space)
  e.g., $A = \{1, \ldots, m\}$ (discrete), $A = \mathbb{R}^m$ (continuous)
- p: state transition probability
  $p(s'|s, a) := \mathrm{Prob}(s_{t+1} = s'|s_t = s, a_t = a)$
- r: reward function
  $r(s_t, a_t) = r_t$
- $\gamma \in (0, 1]$: discount factor

The MDP problem

To find an **optimal policy** that *maximizes the expected cumulative reward*:

$$\max_{\pi \in \Pi} \mathbb{E}^{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

The MDP problem

To find an **optimal policy** that *maximizes the expected cumulative reward*:

$$\max_{\pi \in \Pi} \mathbb{E}^{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

- Difficult to solve *Q) Why?*

policy 라는 "함수"를 최적화 하는 문제라서

gamma: discounting for 미래
if gamma ==1:
과거, 현재, 미래 가중치 동일

The RL Problem

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?
  - **Flexibility**: no knowledge about model

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?
  - **Flexibility**: no knowledge about model
  - **Adaptivity**: no reliance on fixed model

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?
  - **Flexibility**: no knowledge about model
  - **Adaptivity**: no reliance on fixed model
  - **Scalability**: approximate solution to large-scale problems

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?
  - **Flexibility**: no knowledge about model
  - **Adaptivity**: no reliance on fixed model
  - **Scalability**: approximate solution to large-scale problems
- Why difficult?

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?
  - **Flexibility**: no knowledge about model
  - **Adaptivity**: no reliance on fixed model
  - **Scalability**: approximate solution to large-scale problems
- Why difficult?
  - **Limited knowledge**: no knowledge about model

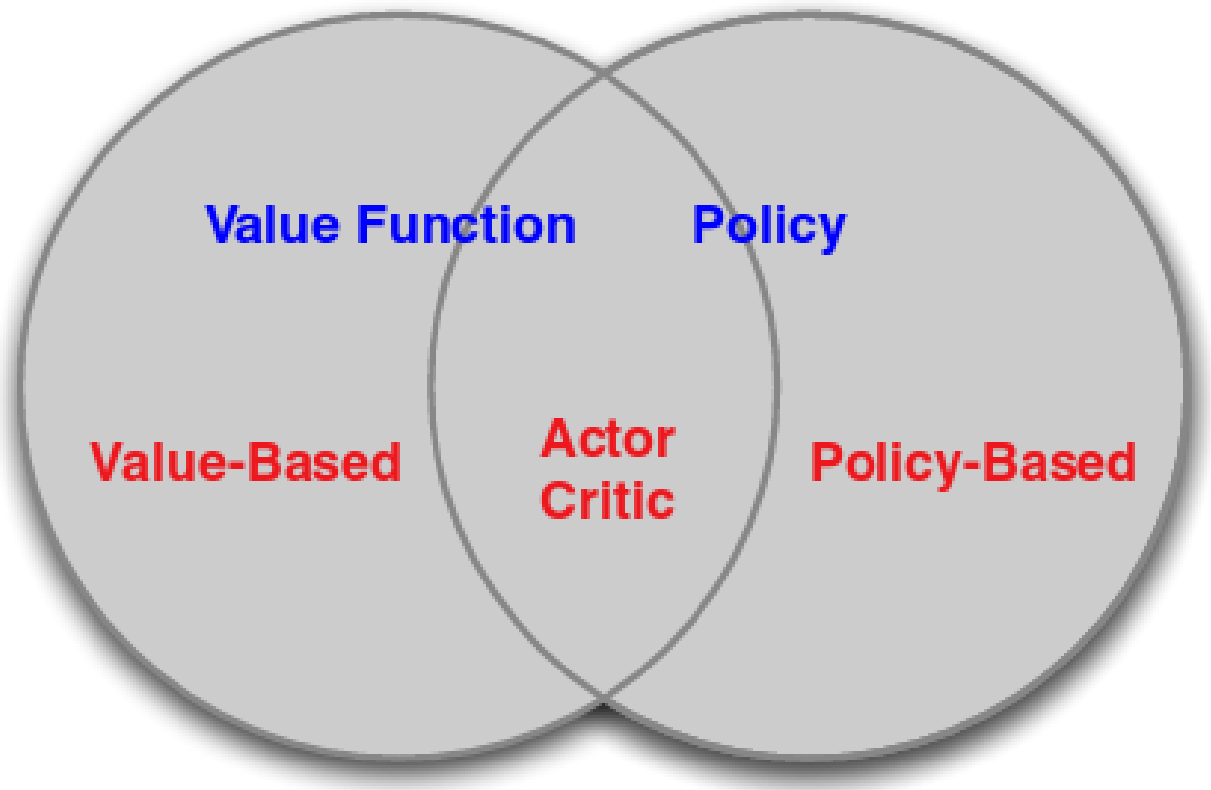
The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?
  - **Flexibility**: no knowledge about model
  - **Adaptivity**: no reliance on fixed model
  - **Scalability**: approximate solution to large-scale problems
- Why difficult?
  - **Limited knowledge**: no knowledge about model
  - **Exploration vs exploitation**: unclear what to try

The RL Problem

- Goal: Solve the MDP problem **when the model** (p, r) **is unknown**
- Approach: "**Trial-and-error**" using experiences (data)
- Why useful?
  - **Flexibility**: no knowledge about model
  - **Adaptivity**: no reliance on fixed model
  - **Scalability**: approximate solution to large-scale problems
- Why difficult?
  - **Limited knowledge**: no knowledge about model
  - **Exploration vs exploitation**: unclear what to try
  - **Limited representation**: unclear how to approximately represent the value function or policy of complex problems

Categorizing RL algorithms



Ok, reinforcement learning seems to work well in solving simple problems

- Structured environments
- Structured inputs (observations, states)
- Static environments
- Failure allowed

Ok, reinforcement learning seems to work well in solving simple problems

- Structured environments
- Structured inputs (observations, states)
- Static environments
- Failure allowed

Q) But, what about this?

Can reinforcement learning solve real-world sequential decision making problems?

Many challenges:

Can reinforcement learning solve real-world sequential decision making problems?

Many challenges:

- Unstructured environments

Can reinforcement learning solve real-world sequential decision making problems?

Many challenges:

- Unstructured environments
- Complex sensory inputs

Can reinforcement learning solve real-world sequential decision making problems?

Many challenges:

- Unstructured environments
- Complex sensory inputs
- Fast adaptation

Can reinforcement learning solve real-world sequential decision making problems?

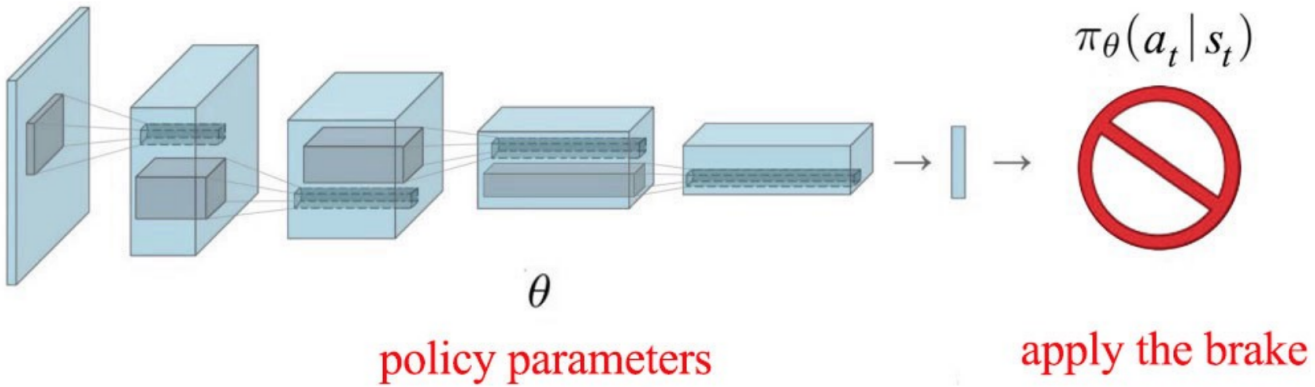
Many challenges:

- Unstructured environments
- Complex sensory inputs
- Fast adaptation
- Reliability, safety

Is there a hope?

Is there a hope?

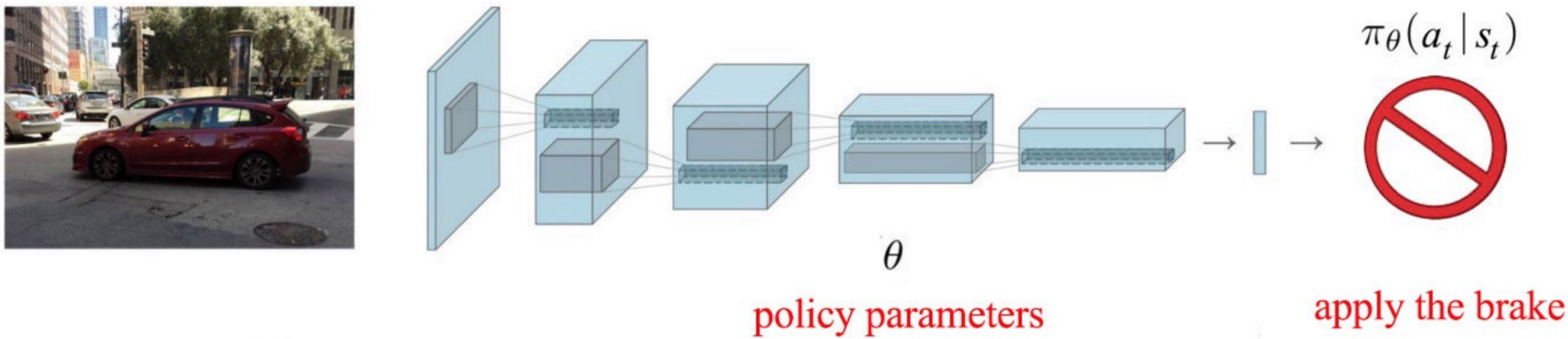
- Yes, can combine RL with the new *deep learning* technology



[S. Levine, CS285]

Is there a hope?

- Yes, can combine RL with the new *deep learning* technology

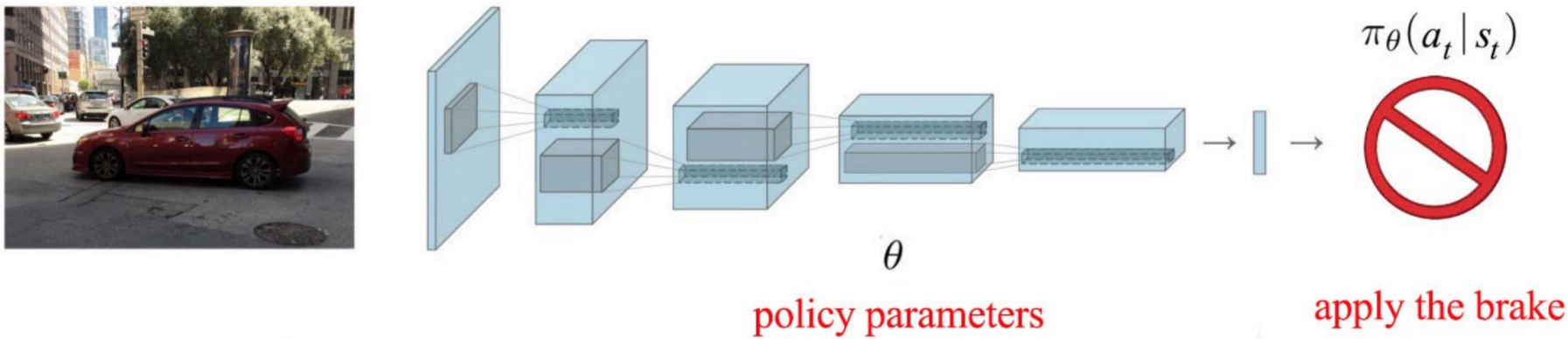


[S. Levine, CS285]

Advantages of deep learning:

Is there a hope?

- Yes, can combine RL with the new *deep learning* technology



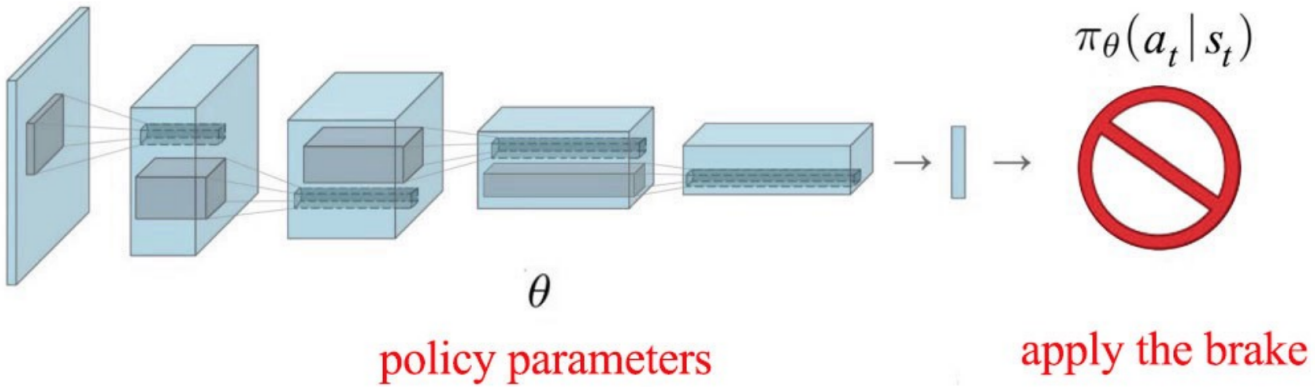
[S. Levine, CS285]

Advantages of deep learning:

- Can process complex sensory inputs

Is there a hope?

- Yes, can combine RL with the new *deep learning* technology



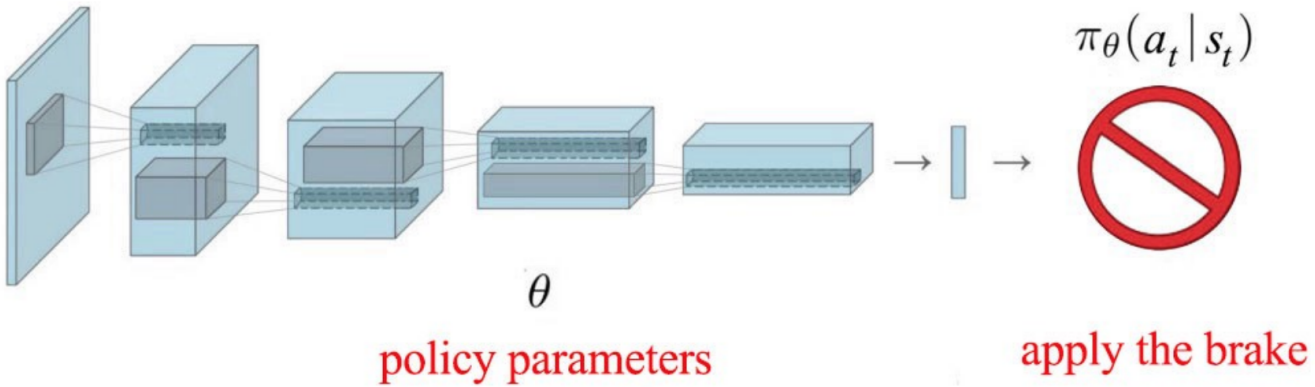
[S. Levine, CS285]

Advantages of deep learning:

- Can process complex sensory inputs
- Can handle unstructured environments

Is there a hope?

- Yes, can combine RL with the new *deep learning* technology



[S. Levine, CS285]

Advantages of deep learning:

- Can process complex sensory inputs
- Can handle unstructured environments
- Powerful computing technology and algorithms for training

What is deep RL?

Deep RL = RL + Deep learning

What is deep RL?

Deep RL = RL + Deep learning

- RL: sequential decision making interacting with environments

What is deep RL?

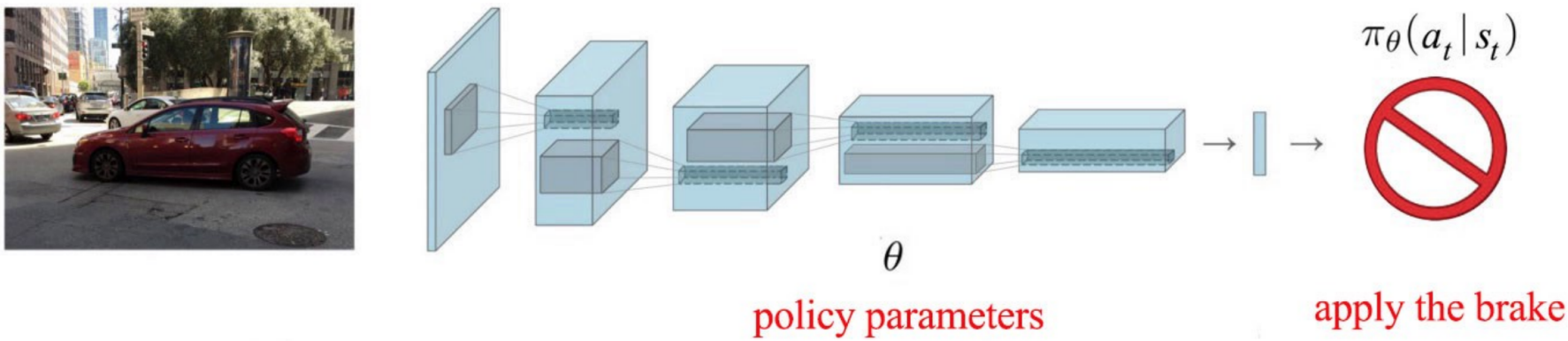
Deep RL = RL + Deep learning

- RL: sequential decision making interacting with environments
- Deep learning: representation of policies and value functions

What is deep RL?

Deep RL = RL + Deep learning

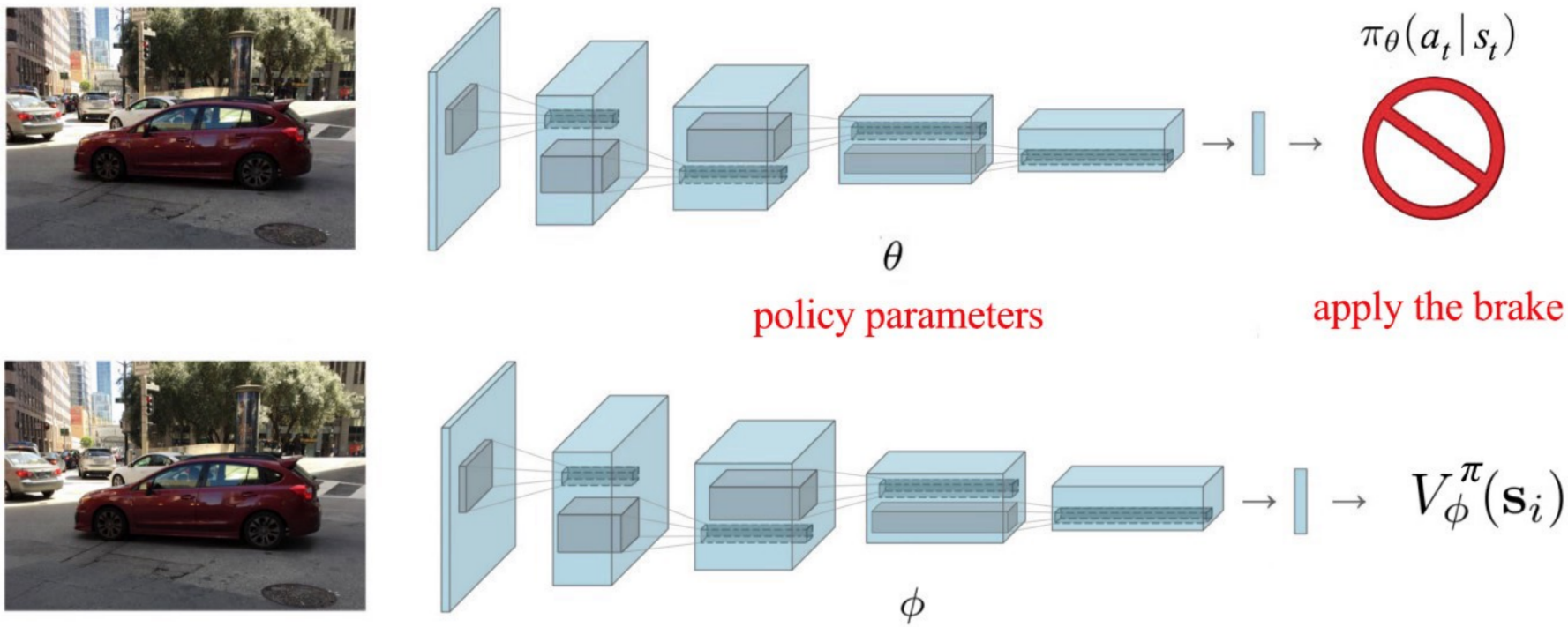
- RL: sequential decision making interacting with environments
- Deep learning: representation of policies and value functions



What is deep RL?

Deep RL = RL + Deep learning

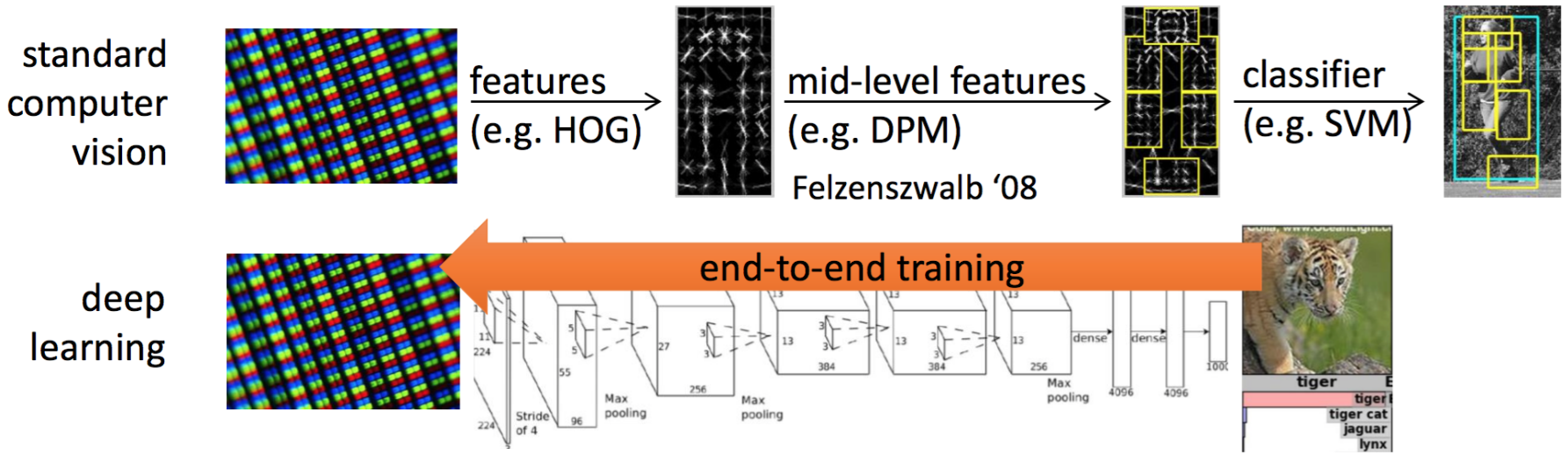
- RL: sequential decision making interacting with environments
- Deep learning: representation of policies and value functions



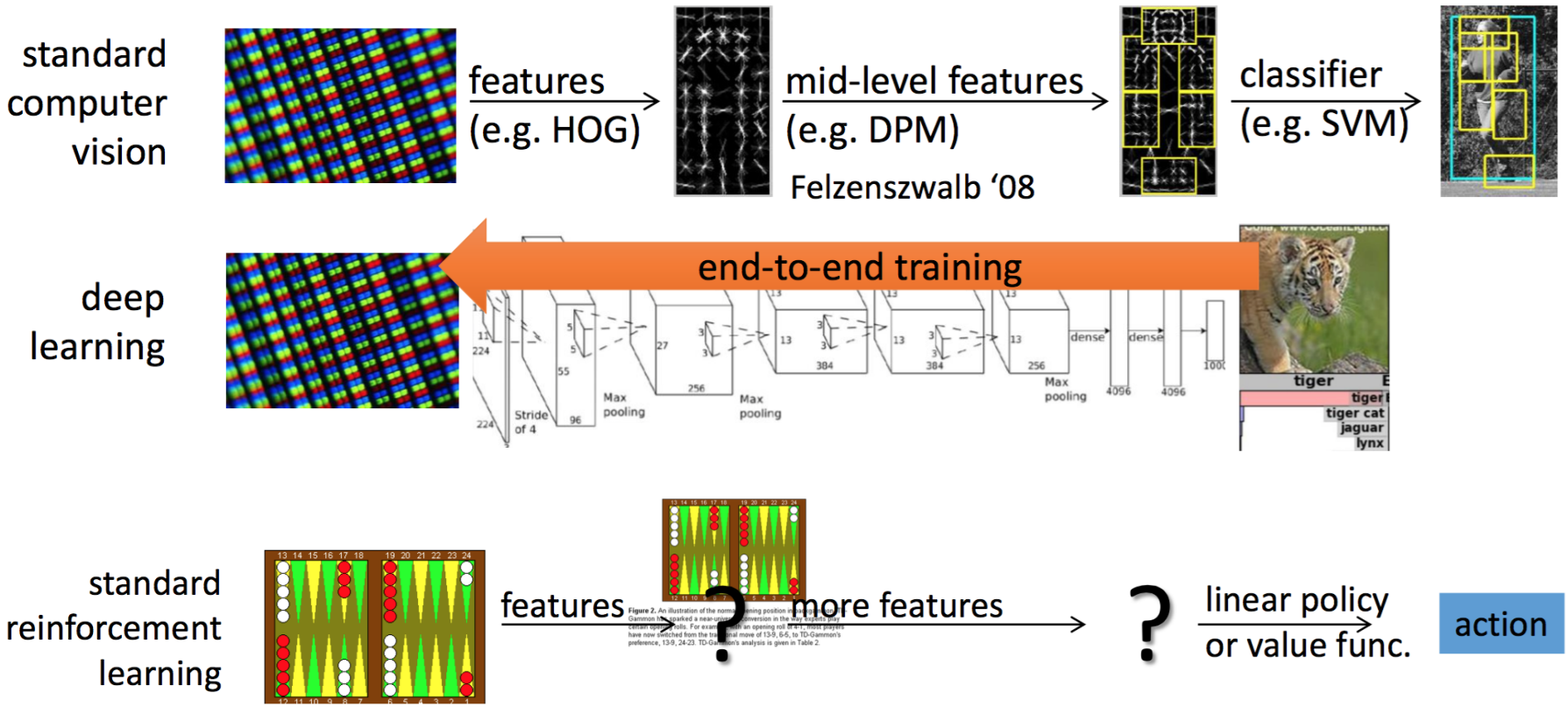
[S. Levine, CS285]

End-to-end learning

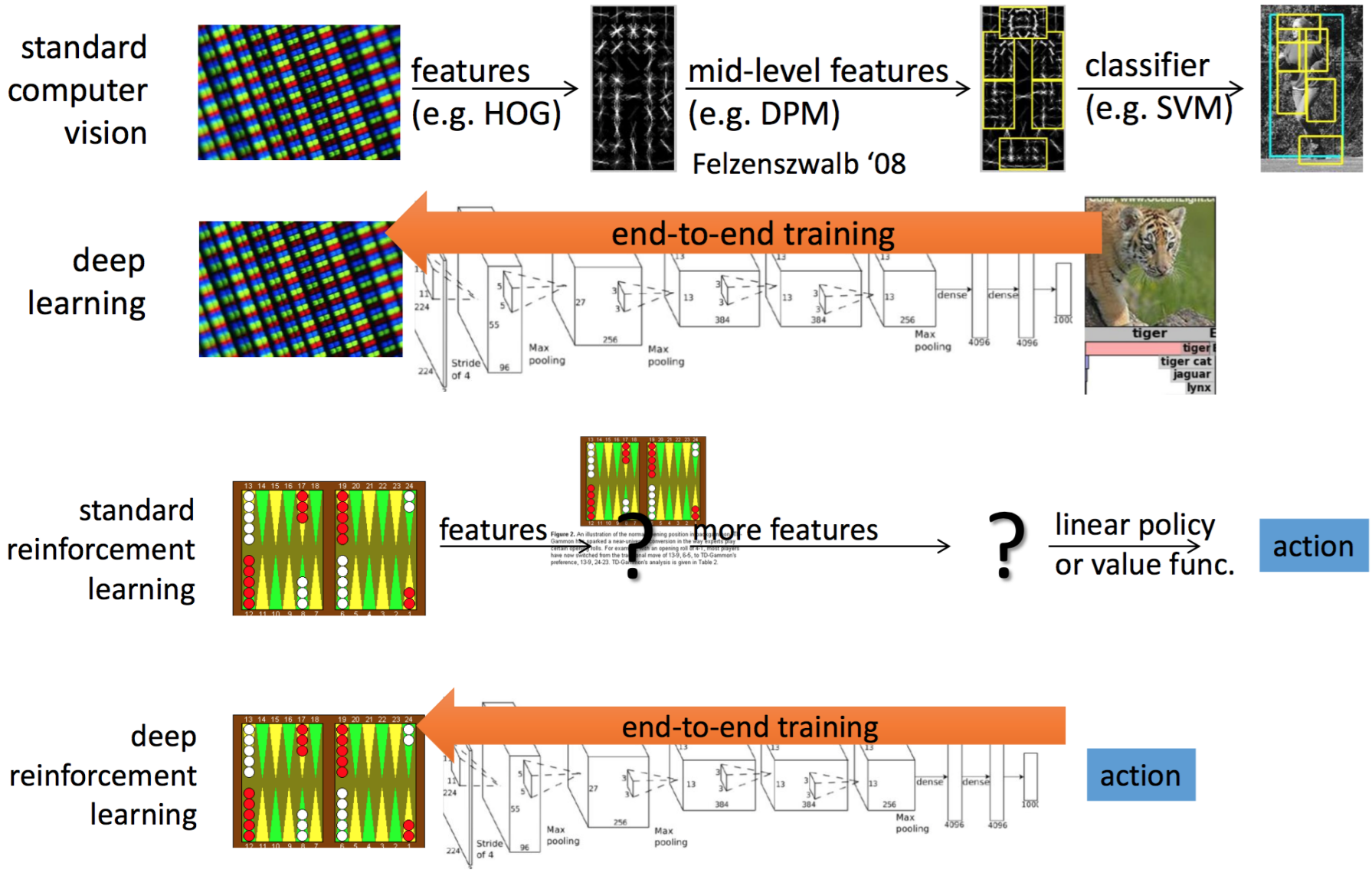
End-to-end learning



End-to-end learning



End-to-end learning



[S. Levine, CS285]

Why do we study this now?

- Advances in deep learning

Why do we study this now?

- Advances in deep learning
- Advances in reinforcement learning

Why do we study this now?

- Advances in deep learning
- Advances in reinforcement learning
- Advances in computing power

Why do we study this now?

- Advances in deep learning
- Advances in reinforcement learning
- Advances in computing power
- **High-performance simulators**

Some deep RL applications

What are the challenges in deep RL?

What are the challenges in deep RL?

- Huge # of samples: millions

What are the challenges in deep RL?

- Huge # of samples: millions
- Fast, stable learning

What are the challenges in deep RL?

- Huge # of samples: millions
- Fast, stable learning
- Hyperparameter tuning

What are the challenges in deep RL?

- Huge # of samples: millions
- Fast, stable learning
- Hyperparameter tuning
- Exploration

What are the challenges in deep RL?

- Huge # of samples: millions
- Fast, stable learning
- Hyperparameter tuning
- Exploration
- Sparse reward signals

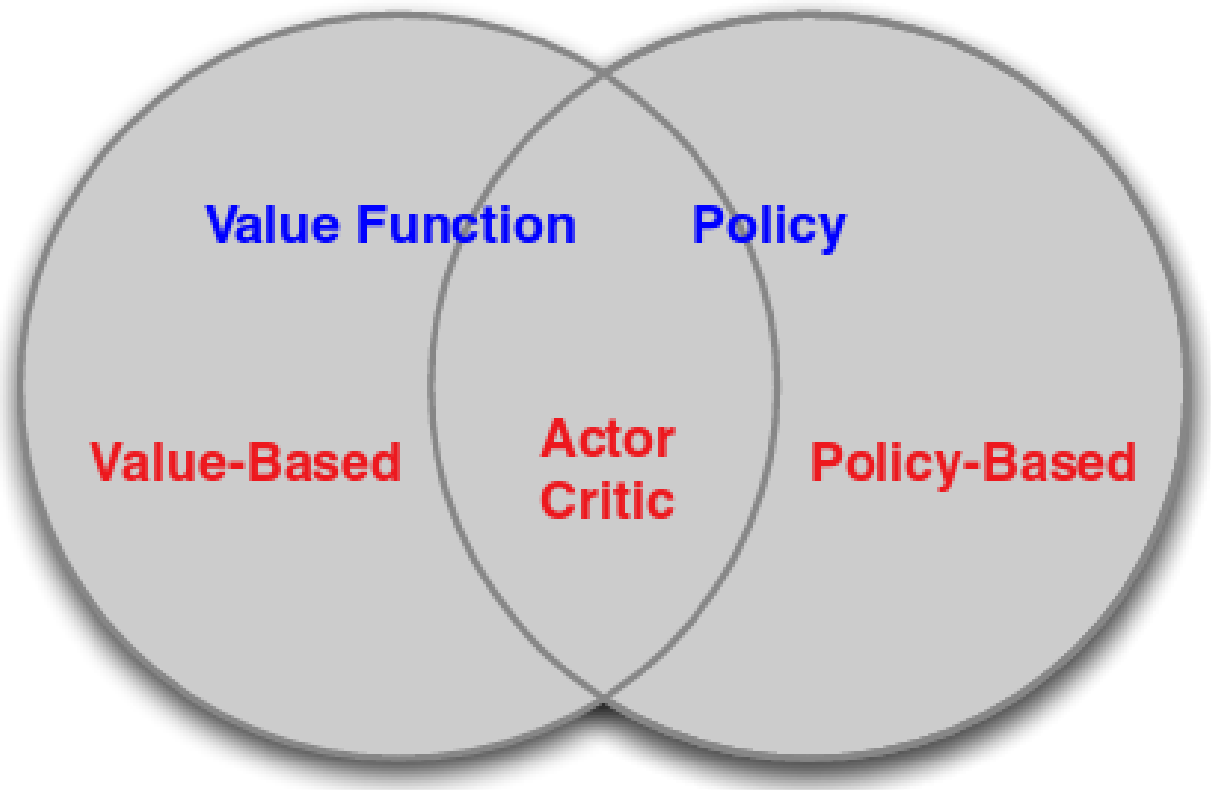
What are the challenges in deep RL?

- Huge # of samples: millions
- Fast, stable learning
- Hyperparameter tuning
- Exploration
- Sparse reward signals
- Safety/reliability

What are the challenges in deep RL?

- Huge # of samples: millions
- Fast, stable learning
- Hyperparameter tuning
- Exploration
- Sparse reward signals
- Safety/reliability
- **Simulator**

Roadmap


Value Function
Policy
Value-Based
Actor
Critic
Policy-Based