SE125 Machine Learning

Introduction to Reinforcement Learning

Yue Ding

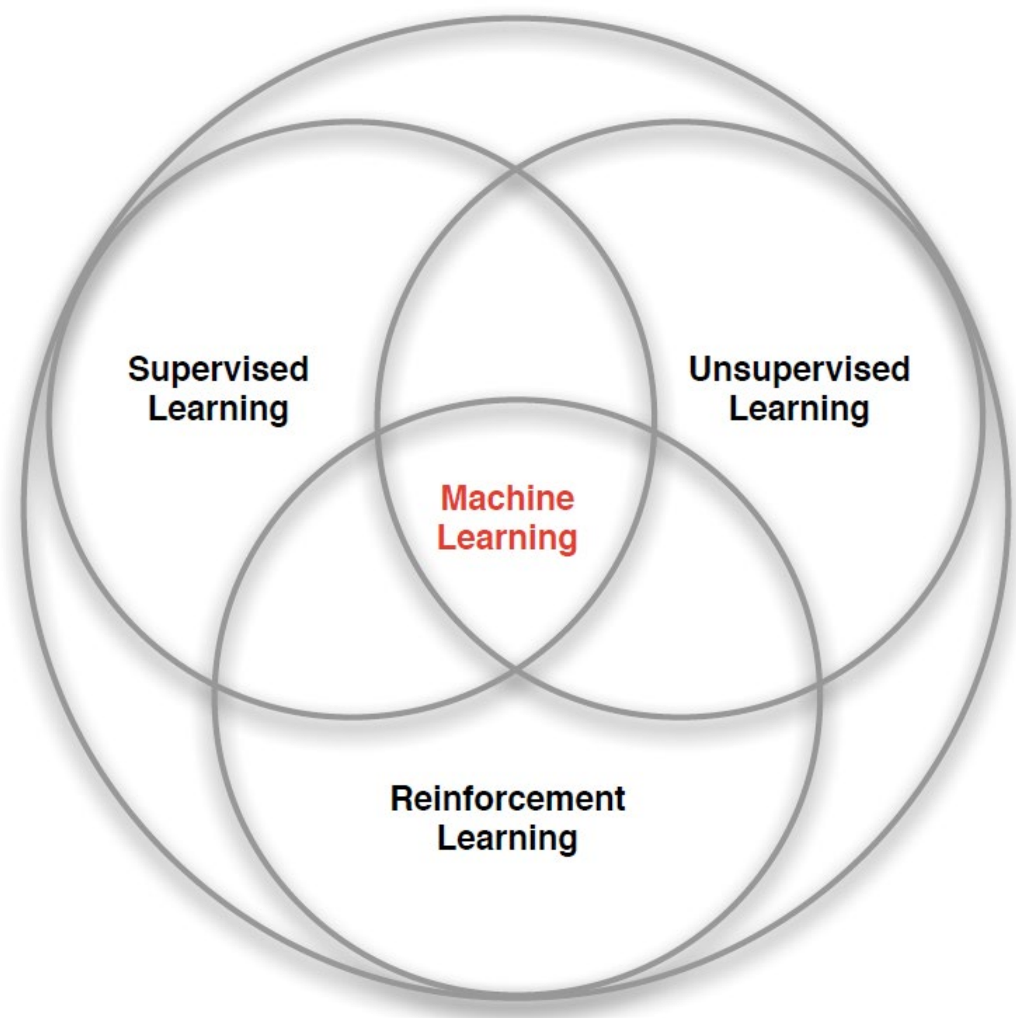
School of Software, Shanghai Jiao Tong University
dingyue@sjtu.edu.cn

Introduction to Reinforcement Learning

- 课程难度:

- 掌握程度:

Branches of Machine Learning



Branches of Machine Learning

- Supervised Learning $p(y|x)$
  - To perform the desired output given the data and labels
- Unsupervised Learning
  - To analyze and make use of the underlying data patterns/structures $p(x)$
- Reinforcement Learning
  - To learn a policy of taking actions in a dynamic environment and acquire rewards $\pi(a|x)$

Deep Reinforcement Learning: $\mathrm{AI} = \mathrm{RL} + \mathrm{DL}$

- We seek a single agent which can solve any human-level task
  - RL defines the objective
  - DL gives the mechanism
  - RL + DL = general intelligence

Google DeepMind
London
davidsilver@google.com

Professor of Computer Science
University College London
d.silver@cs.ucl.ac.uk

I am on indefinite leave of absence from UCL
and not currently accepting any new students.

ABOUT

David Silver is a principal research scientist at DeepMind and a professor at University College London.

David's work focuses on artificially intelligent agents based on reinforcement learning. David co-led the project that combined deep learning and reinforcement learning to play Atari games directly from pixels (Nature 2015).

He also led the AlphaGo project, culminating in the first program to defeat a top professional player in the full-size game of Go (Nature 2016), and the AlphaZero project, which learned by itself to defeat the world's strongest chess, shogi and Go programs (Nature 2017, Science 2018).

Most recently he co-led the AlphaStar project, which led to the world's first grandmaster level StarCraft player (Nature 2019).

His work has been recognised by the Marvin Minsky award, Mensa Foundation Prize, and Royal Academy of Engineering Silver Medal.

DeepMind > About

AI could be one of humanity's most useful inventions

When we started DeepMind in 2010, there was far less interest in the field of AI than there is today. To accelerate the field, we took an interdisciplinary approach, bringing together new ideas and advances in machine learning, neuroscience, engineering, mathematics, simulation and computing infrastructure, along with new ways of organising scientific endeavour.

We achieved early success in computer games, which researchers often use to test AI. One of our programs learned to play 49 different Atari games from scratch, just from seeing the pixels and score on the screen. Our AlphaGo program was also the first to beat a professional Go player, a feat described as a decade ahead of its time.

AlphaGo

- Despite decades of work, the strongest Go computer programs could only play at the level of human amateurs.
- Standard AI methods, which test all possible moves and positions using a search tree, ***can't handle the sheer number of possible Go moves or evaluate the strength of each possible board position***.

There are an astonishing 10 to the power of 170 possible board configurations.

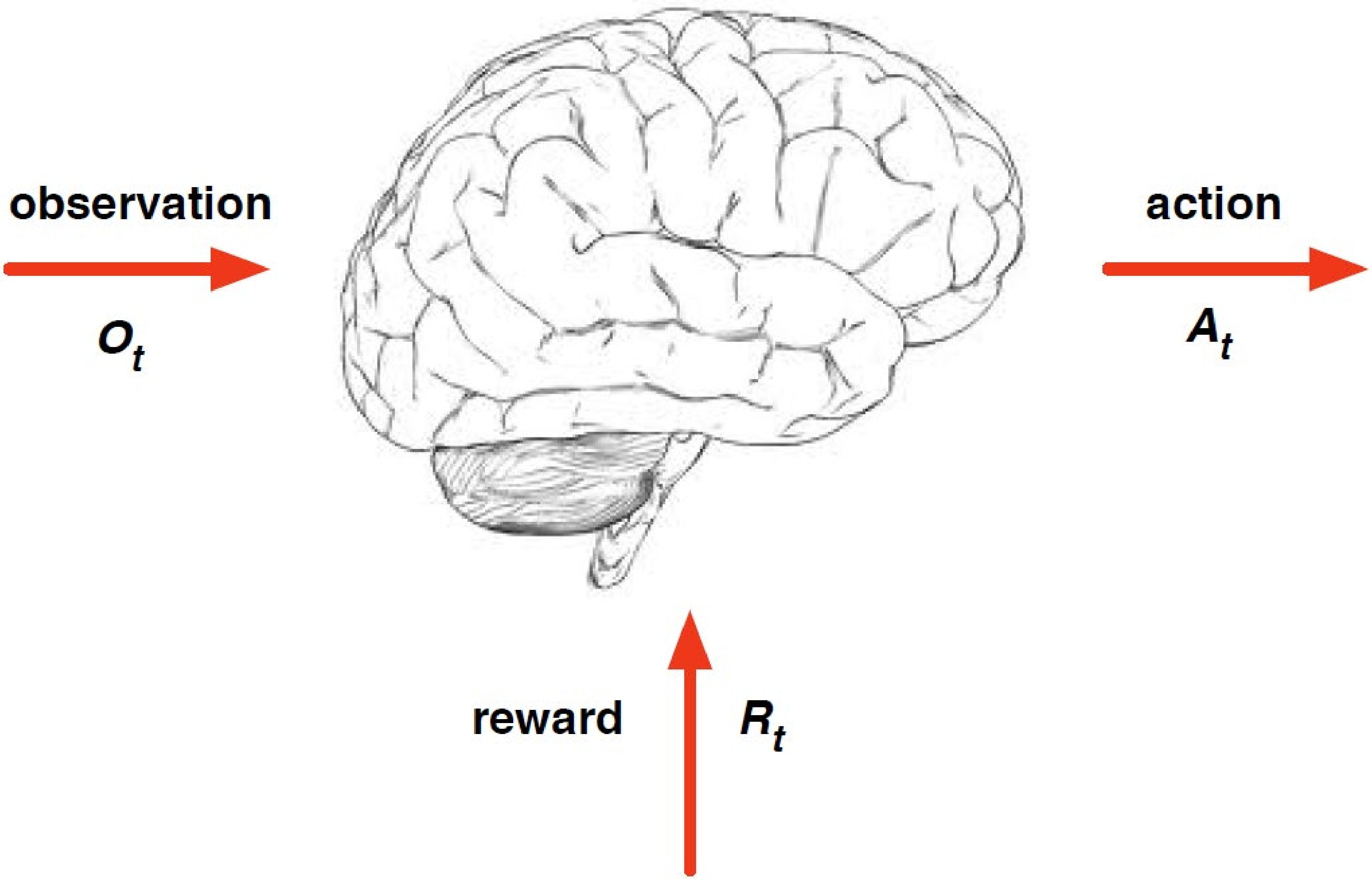
AlphaGo

版本	硬件	等级分	赛况
AlphaGo Fan	176个GPU、[4]分布式	3144[1]	5：0 对阵 樊麾
AlphaGo Lee	48个TPU、[4]分布式	3739[1]	4：1 对阵 李世石
AlphaGo Master	4个第二代TPU[4]、单机	4858[1]	网棋 60:0 对阵 44位职业棋手 中国乌镇围棋峰会 3:0 对阵 柯洁；1:0 对阵 五位顶尖棋手联队
AlphaGo Zero	4个第二代TPU[4]、单机	5185[1]	100：0 对阵AlphaGo Lee 89：11 对阵AlphaGo Master

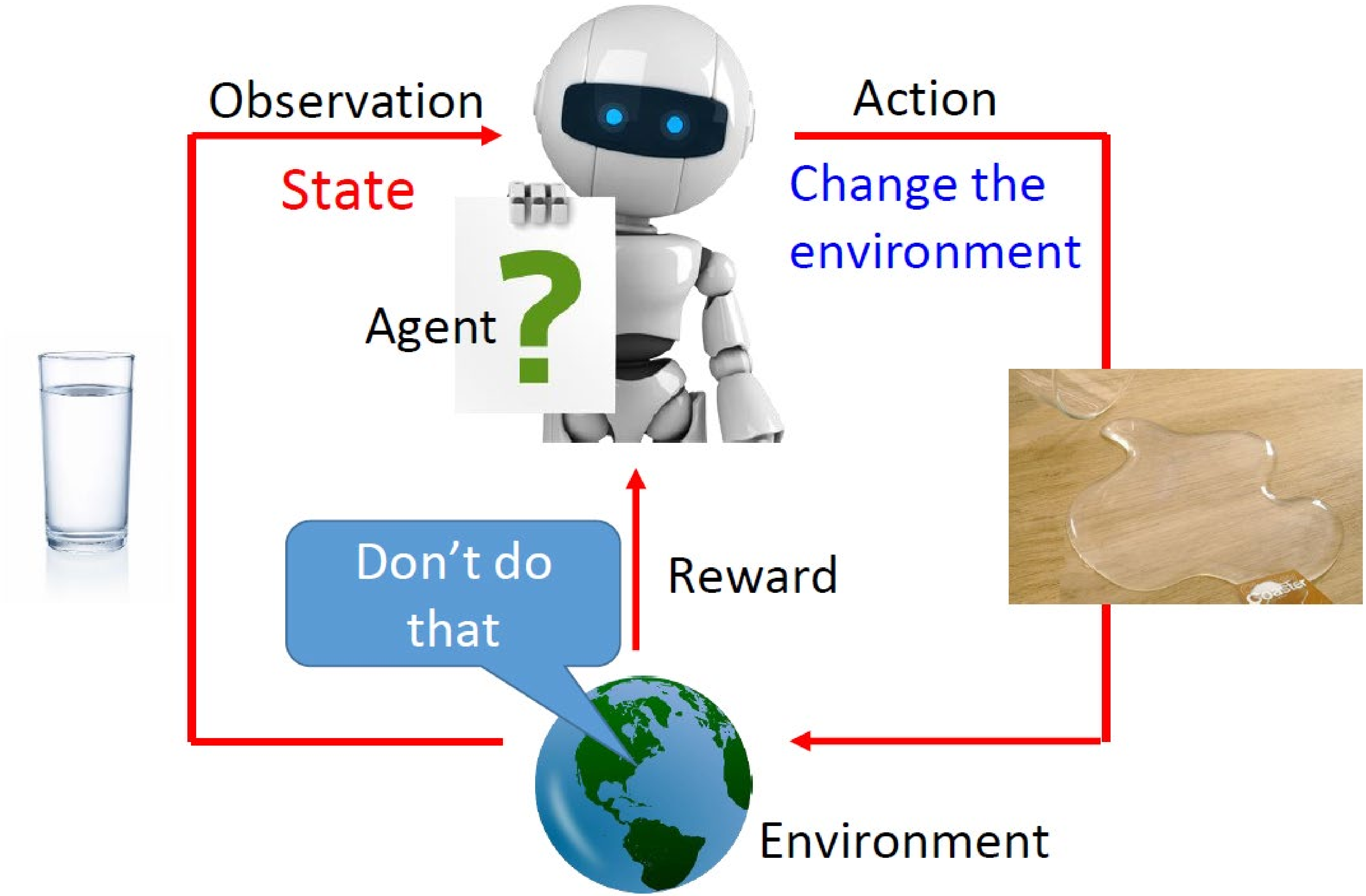
AlphaStar

StarCraft II, created by [Blizzard Entertainment](), is set in a fictional sci-fi universe and features rich, multi-layered gameplay designed to challenge human intellect. Along with the original title, it is among the biggest and most successful games of all time, with players competing in esports tournaments for more than 20 years.

Agent and Environment

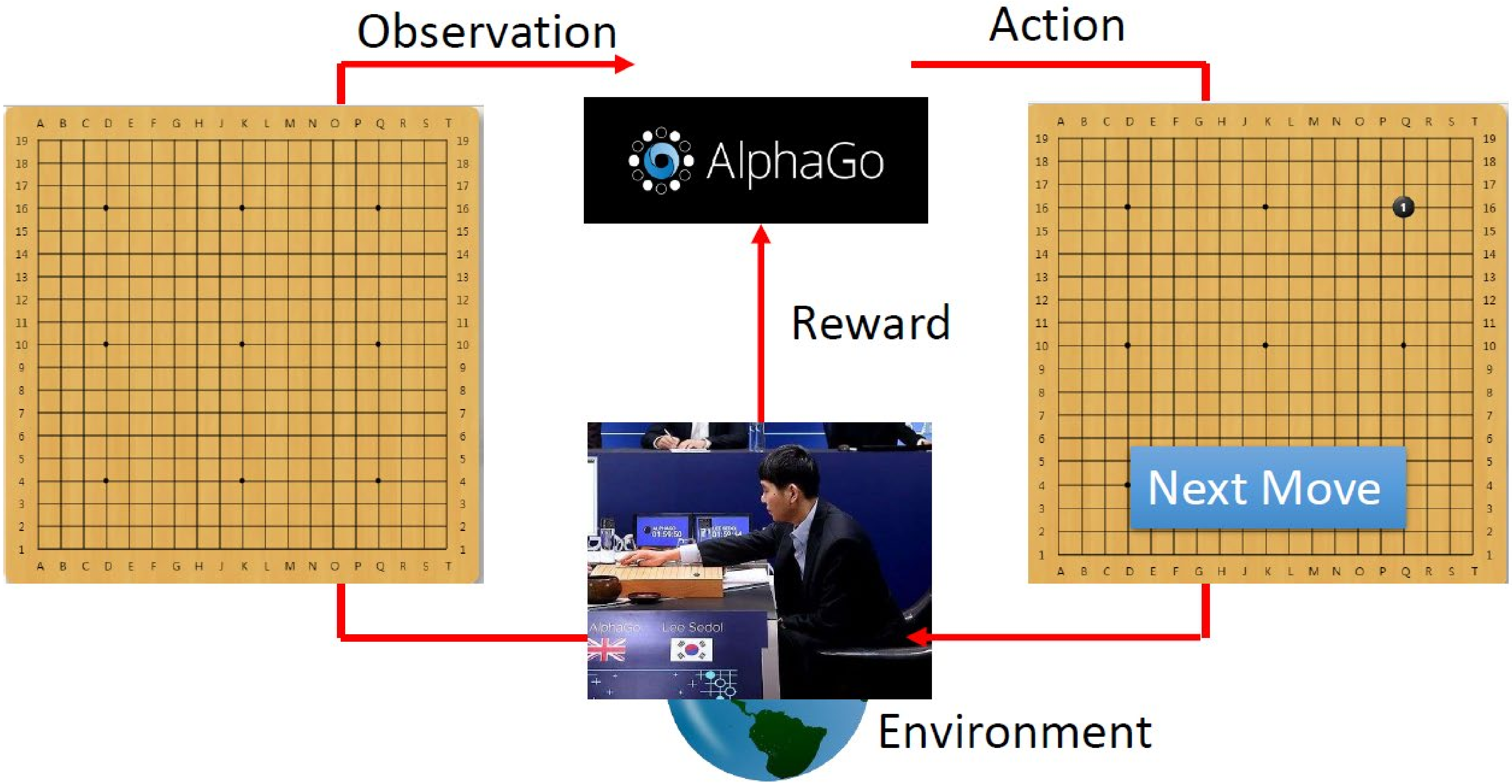


Scenario of Reinforcement Learning



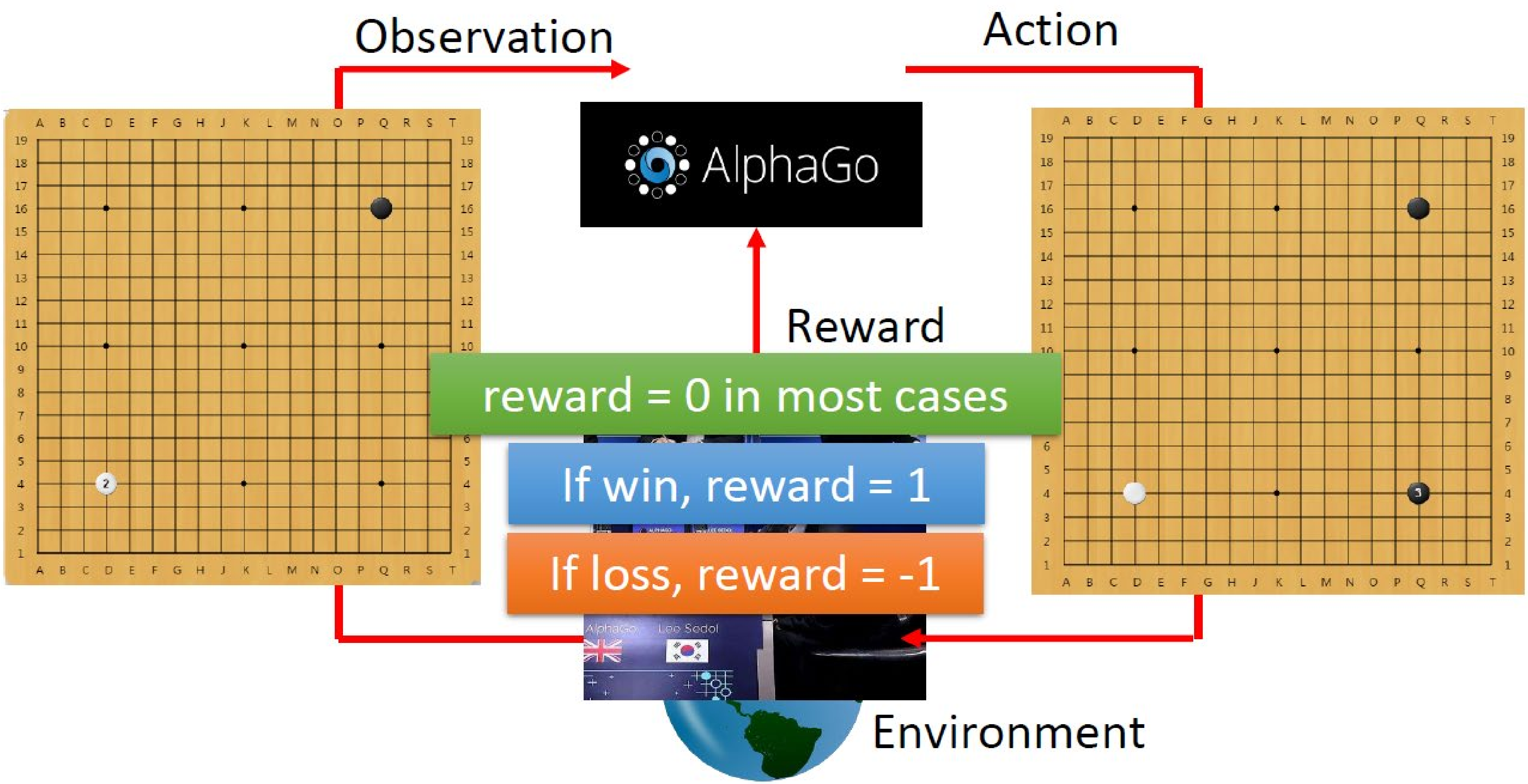
Scenario of Reinforcement Learning

Learning to Play Go



Learning to Play Go

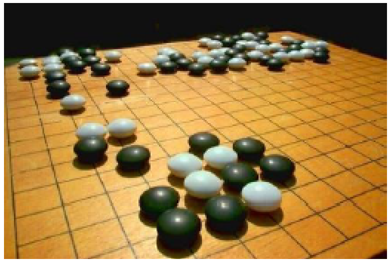
Agent learns to take actions to maximize expected reward.



Learning to paly Go

- Supervised: Learning from teacher

Next move: "5-5"



Next move: "3-3"

- Reinforcement Learning Learning from experience

First move → ...... many moves ...... → Win!

(Two agents play with each other.)

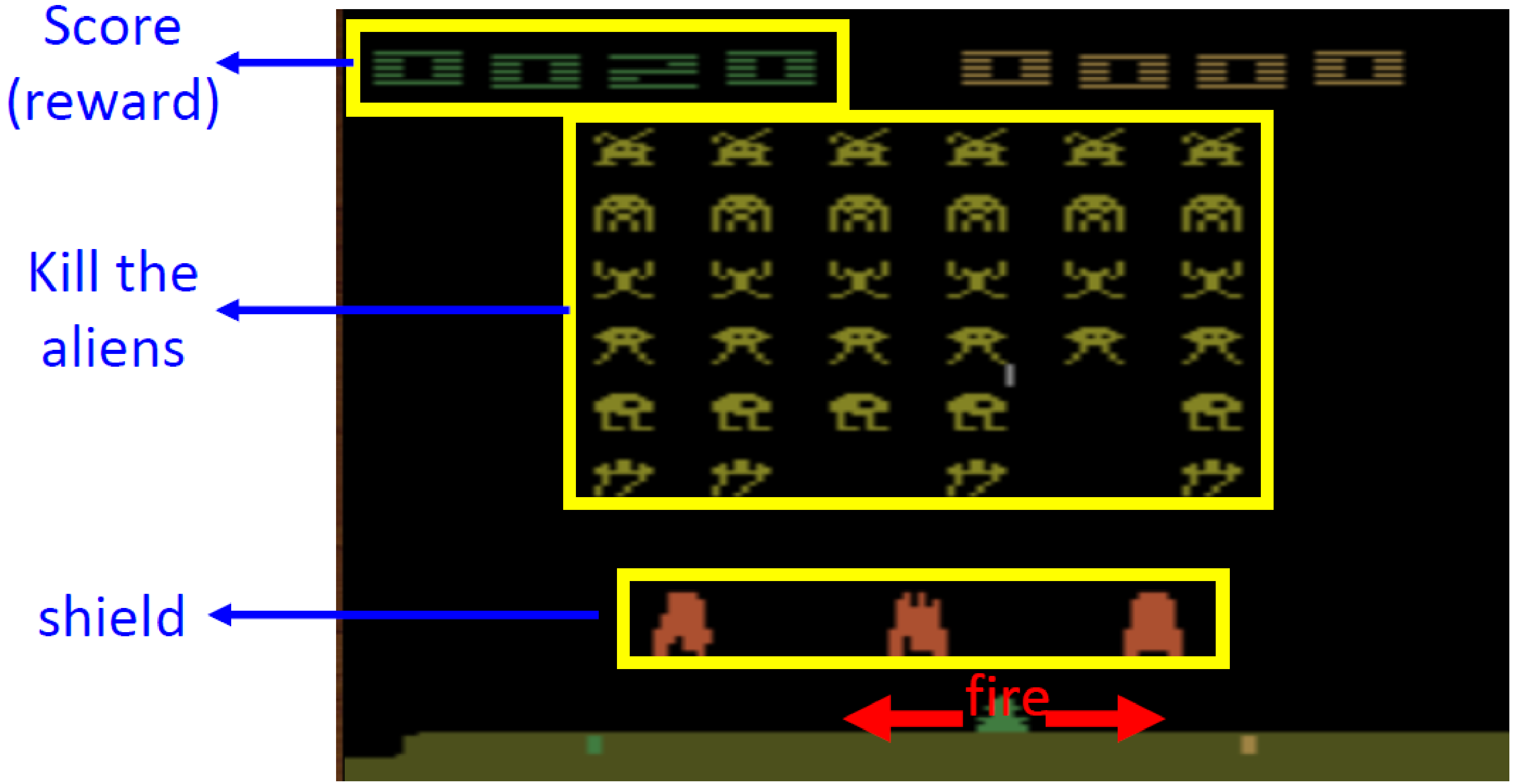
Reinforcement Learning in a Nutshell

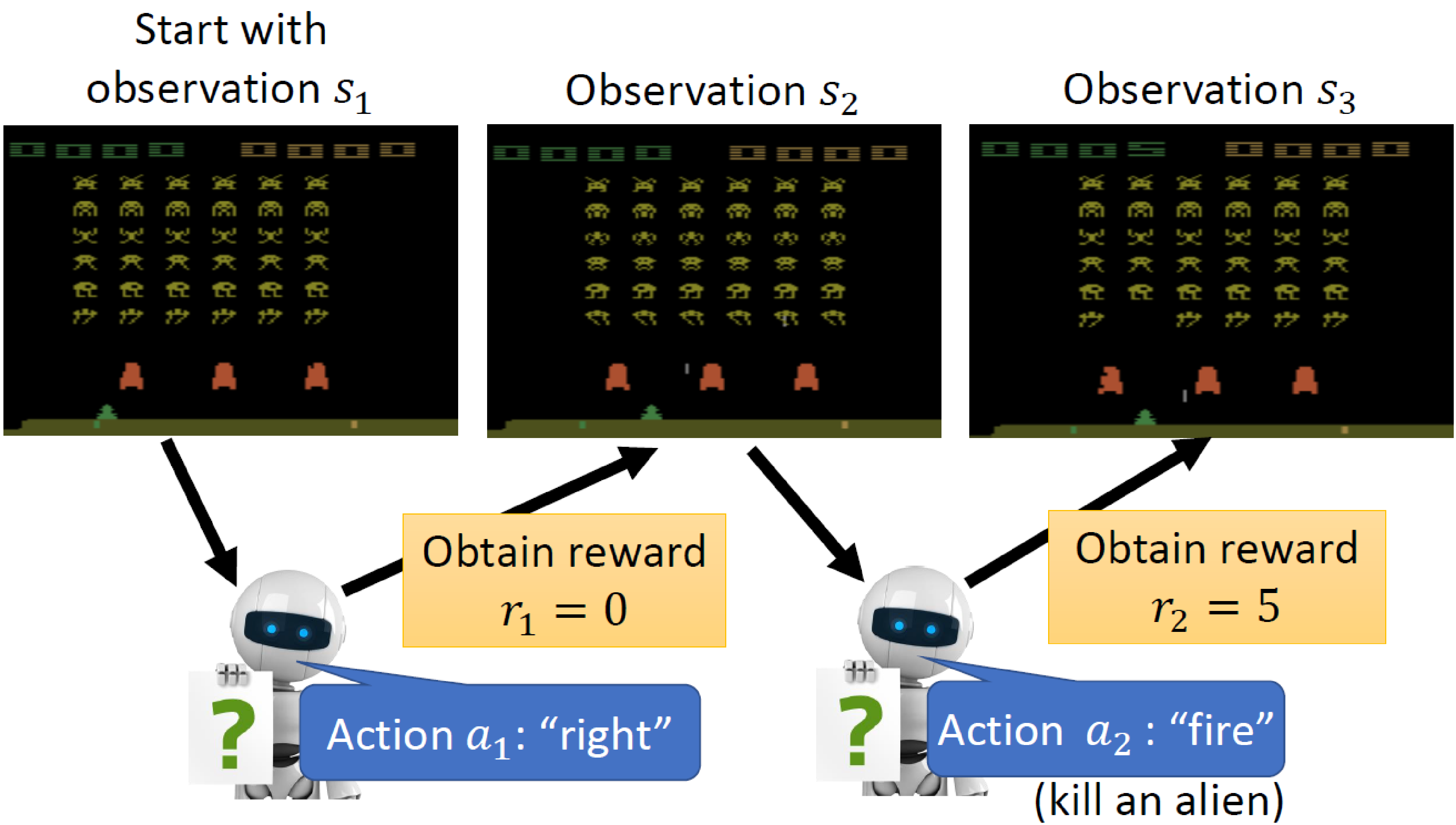
- RL is a general-purpose framework for decision-making
  - RL is for an agent with the capacity to act
  - Each action influences the agent's future state
  - Success is measured by a scalar reward signal
  - Goal: select actions to maximize future reward

Playing Video Game

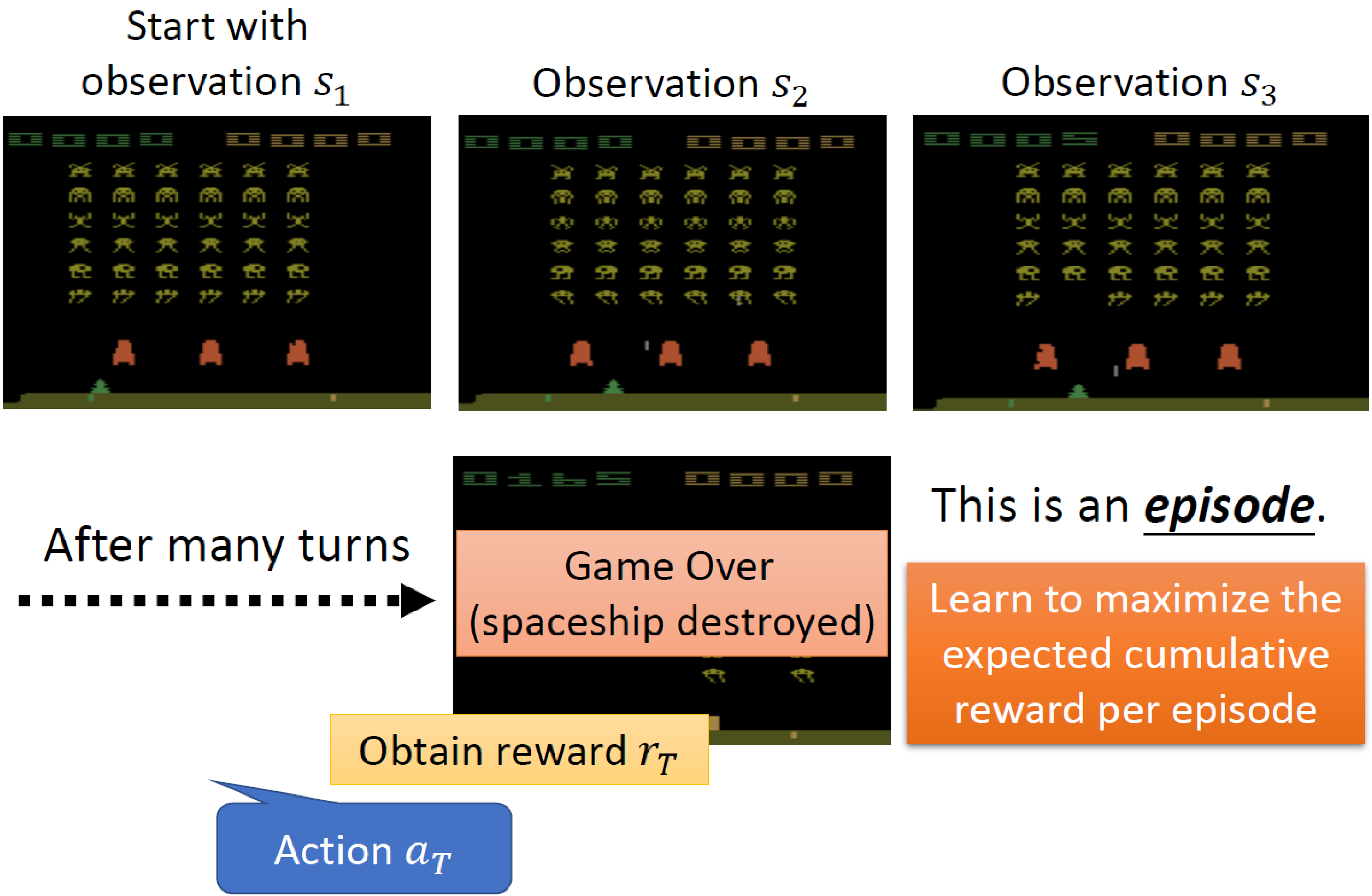
- Space invader

Termination: all the aliens are killed, or your spaceship is destroyed.





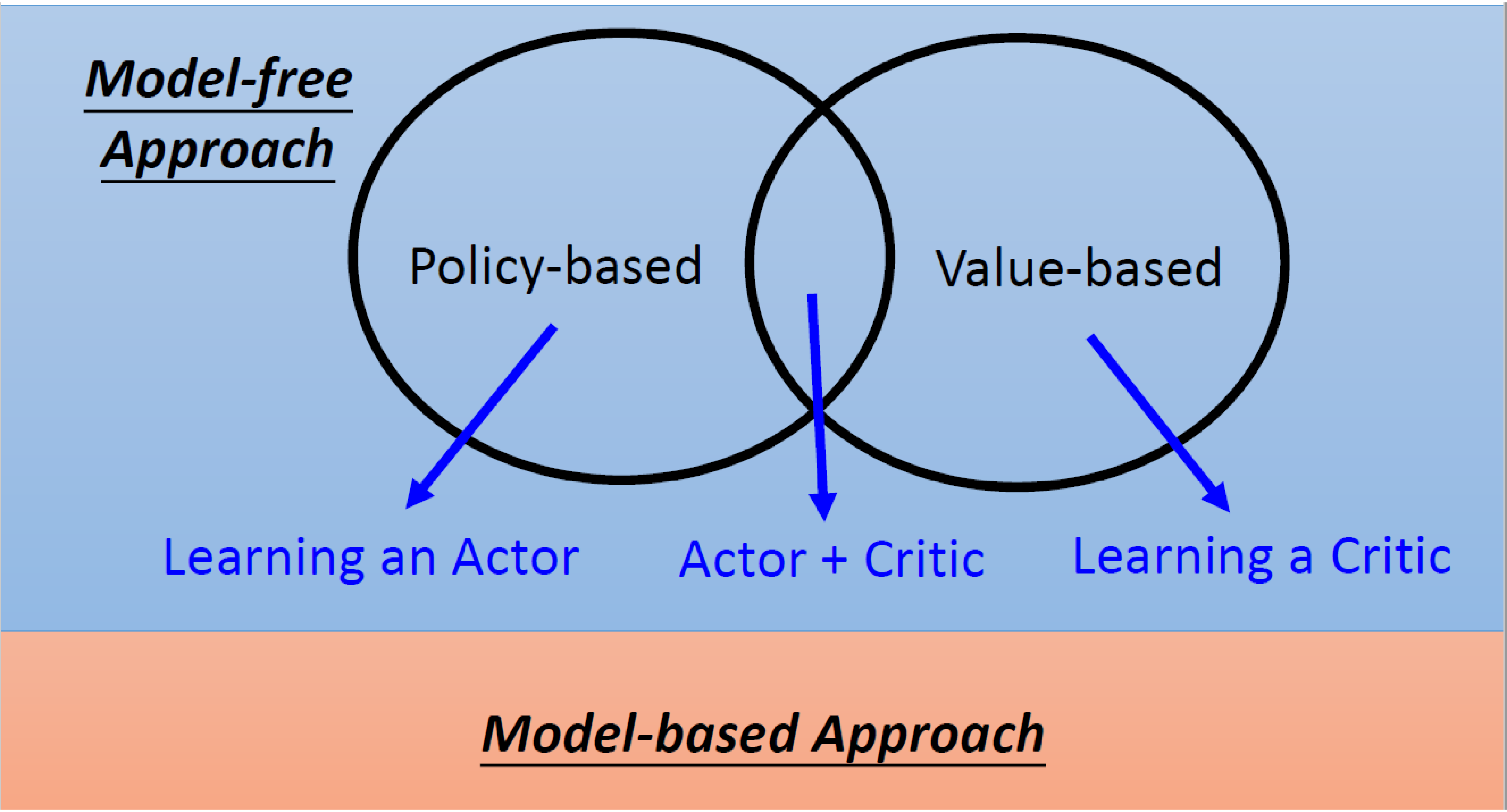
Usually there is some randomness in the environment


Start with observation s_1
Observation s_2
Observation s_3
After many turns
Game Over
(spaceship destroyed)
This is an ***episode***.
Learn to maximize the expected cumulative reward per episode
Obtain reward r_T
Action a_T

Difficulties of Reinforcement Learning

- Reward delay
  - In space invader, only “fire” obtains reward
    - Although the moving before “fire” is important
  - In Go playing, it may be better to sacrifice immediate reward to gain more long-term reward
- Agent’s actions affect the subsequent data it receives
  - E.g. Exploration

Approaches to Reinforcement Learning



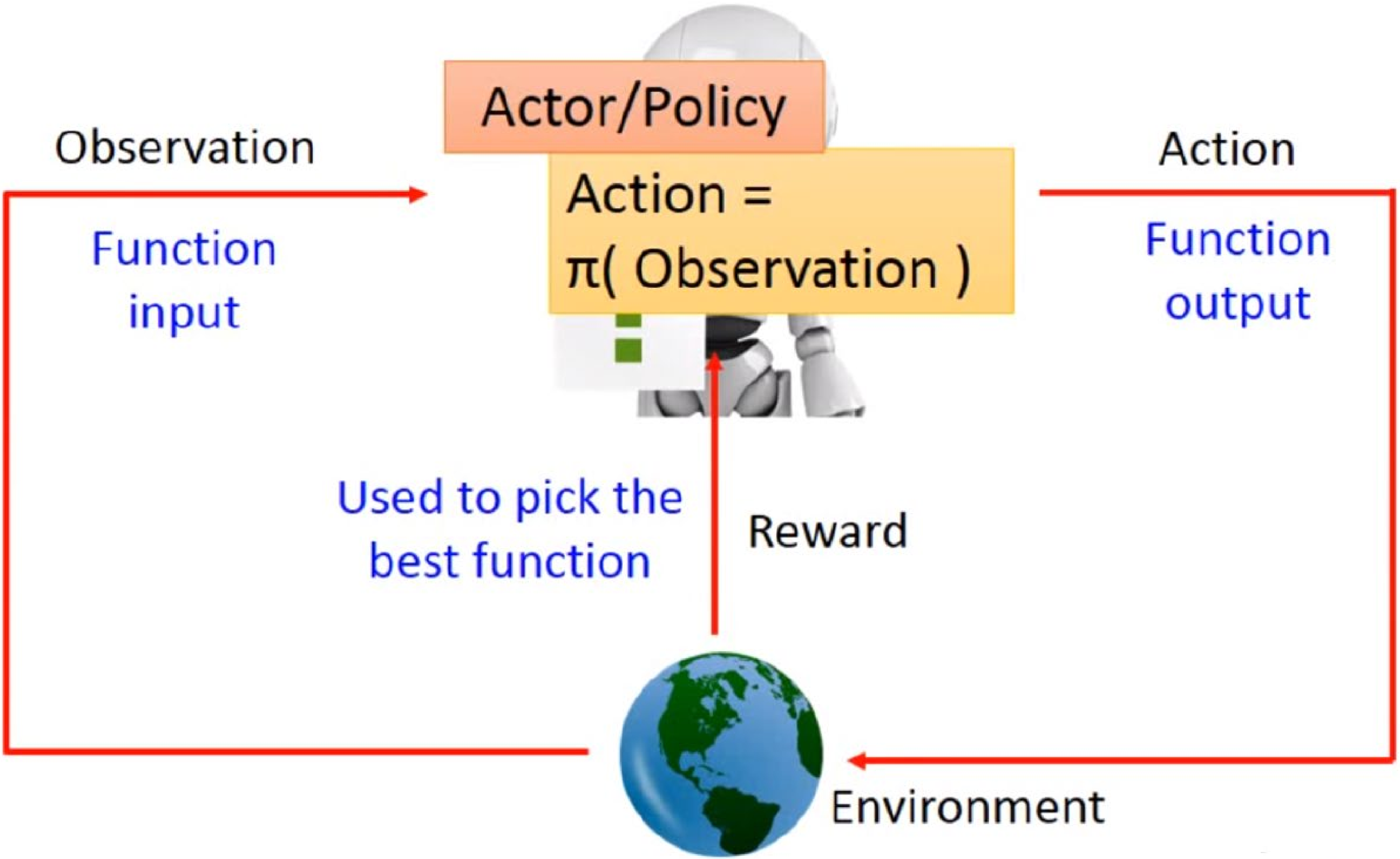
Major Components of an RL Agent

- An RL agent may include one or more of these components:
  - Policy: agent's behavior function
    - It is a map from state to action
      - Deterministic policy: $a = \pi(s)$
      - Stochastic policy: $\pi(a|s) = \mathbb{P}[a|s]$
  - Value function: a prediction of future reward
  - Model: agent's representation of the environment
    - Model is learnt from experience
    - Acts as proxy for environment

Policy-based Approach

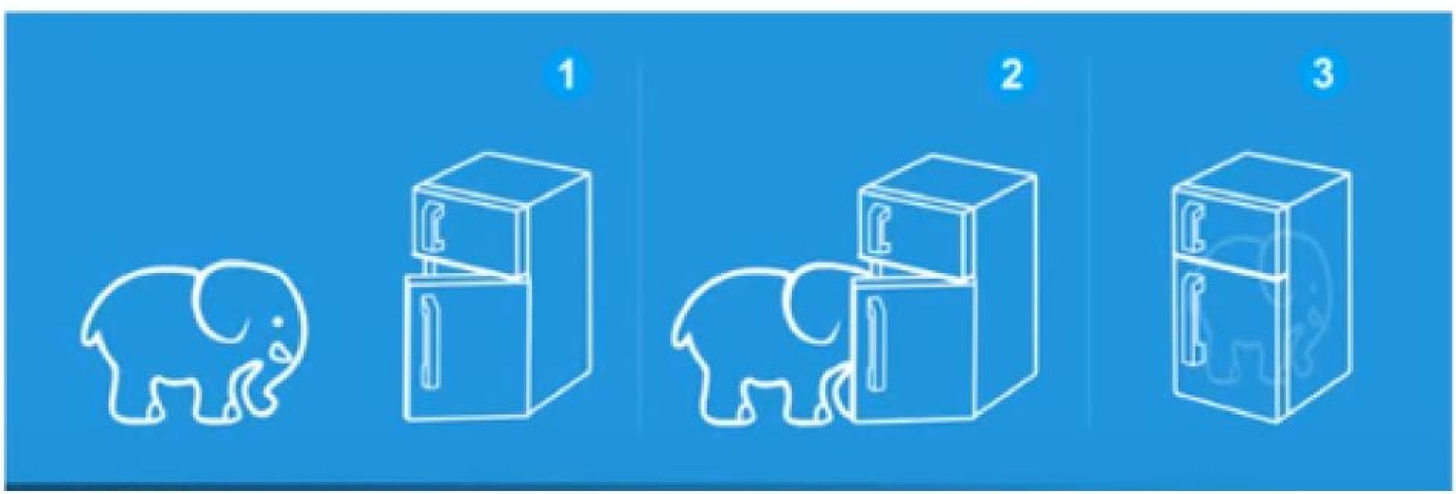
Learning an Actor

Machine Learning ≈ Looking for a Function



Three Steps for Deep Learning

Deep Learning is so simple ……



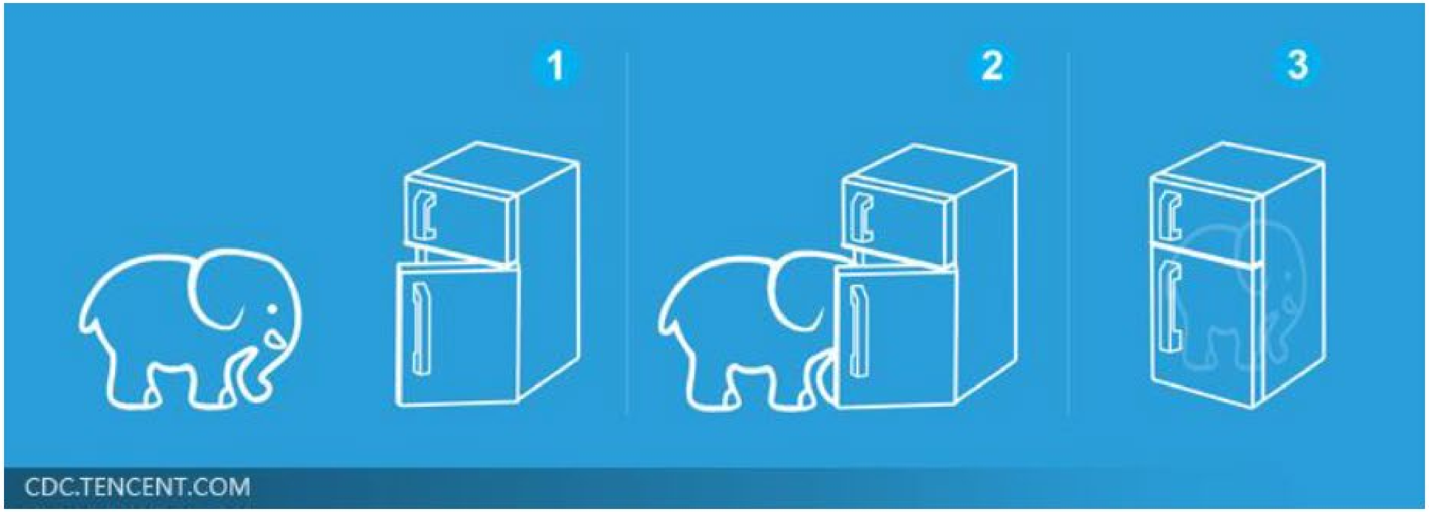
Three Steps for Deep Learning

Step 1: Neural Network as Actor

Step 2: goodness of function

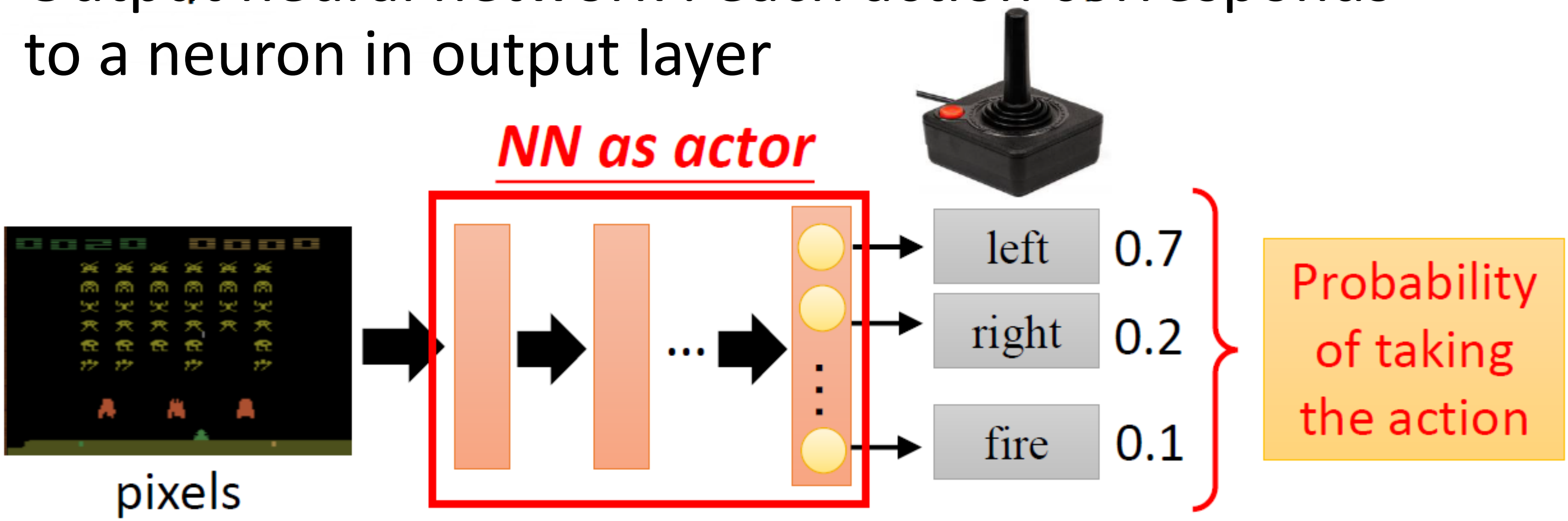
Step 3: pick the best function

Deep Learning is so simple ……



Neural Network as Actor

- Input of neural network: the observation of machine represented as a vector or a matrix
- Output neural network : each action corresponds to a neuron in output layer



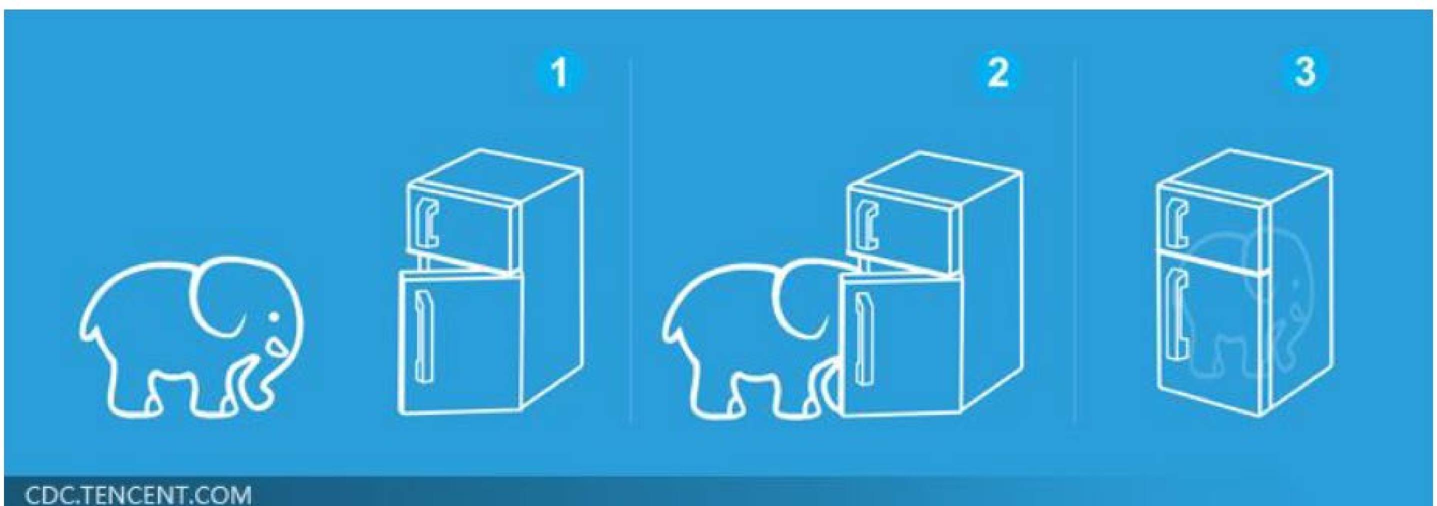
What is the benefit of using network instead of lookup table?

generalization

Three Steps for Deep Learning

Step 1: Neural Network as Actor → **Step 2: goodness of function** → Step 3: pick the best function

Deep Learning is so simple ……

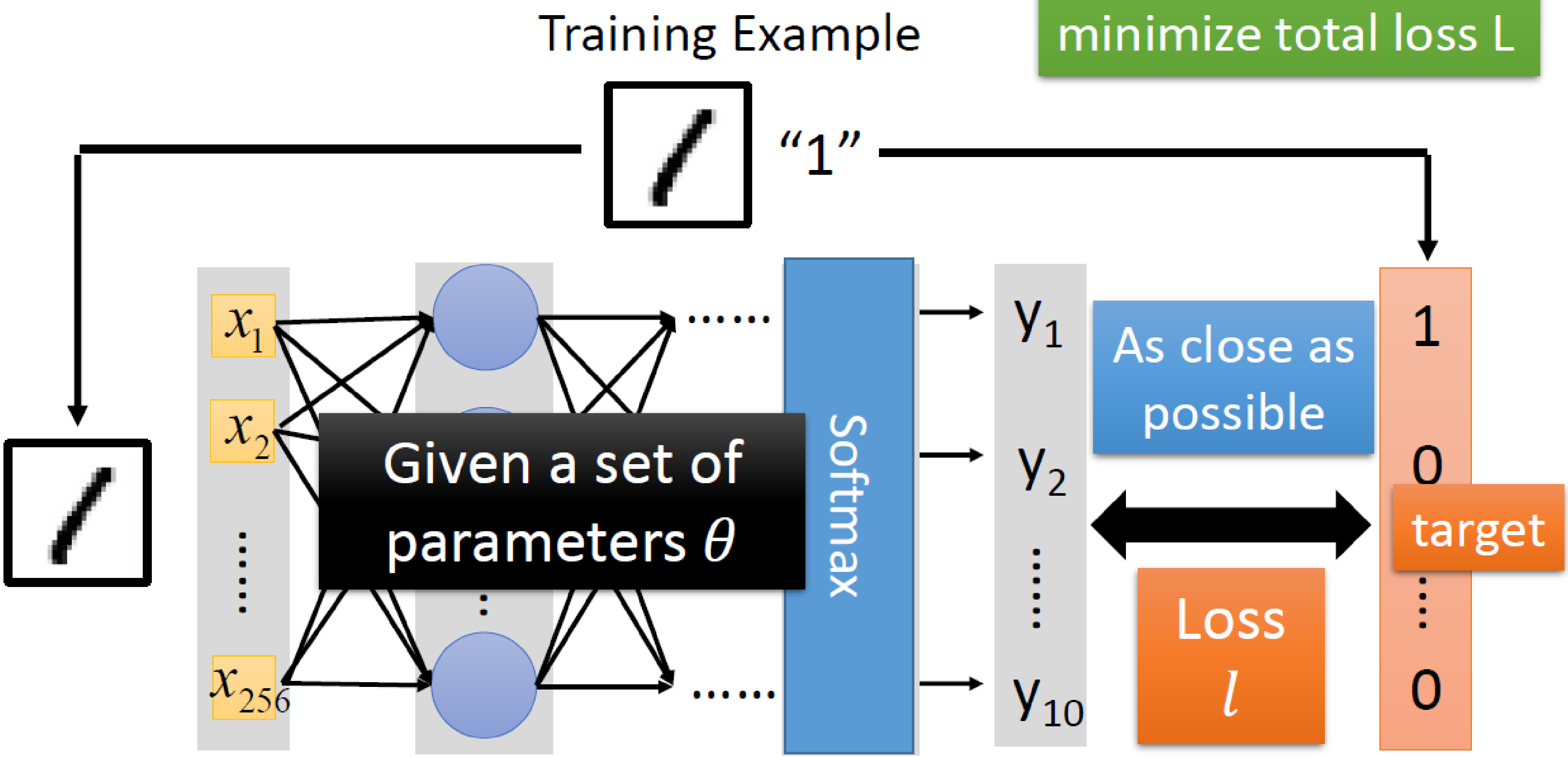


Goodness of Actor

Total Loss: $\boxed{L = \sum_{n=1}^{N} l_n}$

Find ***the network parameters θ^**** that minimize total loss L

- Review: Supervised learning



Goodness of Actor

Step 2: goodness of ~~function~~ Actor

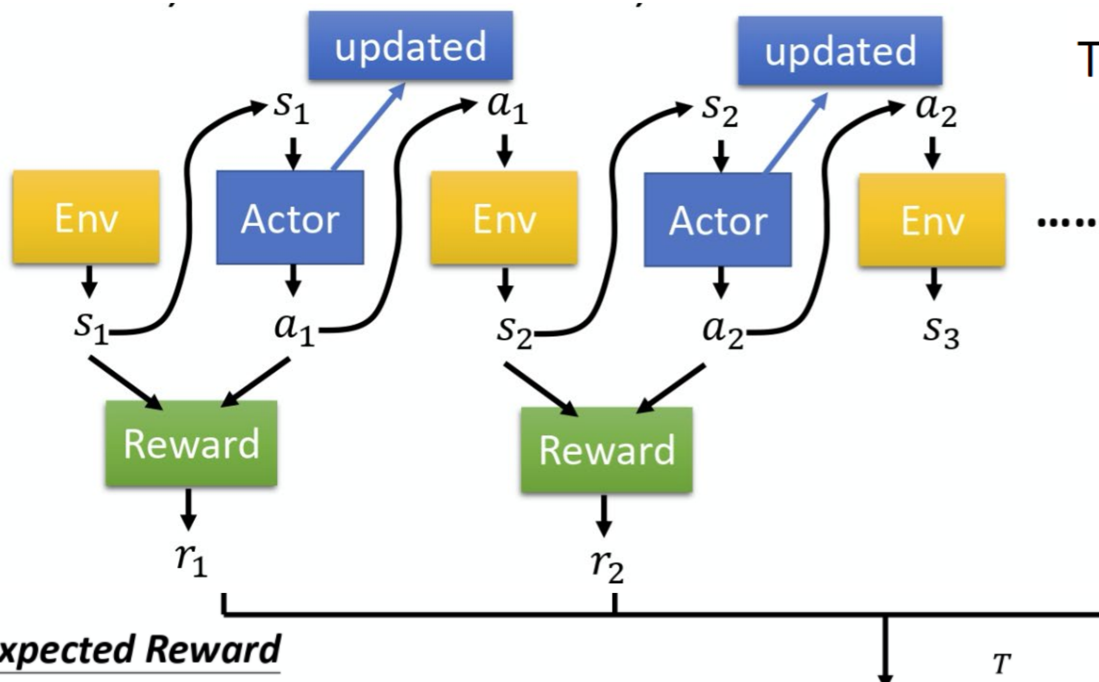
- Given an actor $\pi_\theta(s)$ with network parameter θ
- Use the actor $\pi_\theta(s)$ to play the video game

Goodness of Actor

- Given an actor $\pi_\theta(s)$ with network parameter θ
- Use the actor $\pi_\theta(s)$ to play the video game
  - Start with observation s_1
  - Machine decides to take a_1
  - Machine obtains reward r_1
  - Machine sees observation s_2
  - Machine decides to take a_2
  - Machine obtains reward r_2
  - Machine sees observation s_3
  - ……
  - Machine decides to take a_T
  - Machine obtains reward r_T END

Goodness of Actor

- Given an actor $\pi_\theta(s)$ with network parameter θ
- Use the actor $\pi_\theta(s)$ to play the video game



Total reward: $R_\theta = \sum_{t=1}^{T} r_t$

Even with the same actor, R_θ is different each time

Randomness in the actor and the game

We define $\bar{R}_\theta$ as the *expected value* of R_θ

Expected Reward

$$\bar{R}_\theta = \sum_{\tau} R(\tau) p_\theta(\tau) = E_{\tau \sim p_\theta(\tau)}[R(\tau)]$$

$$R(\tau) = \sum_{t=1}^{T} r_t$$

$\bar{R}_\theta$ evaluates the goodness of an actor $\pi_\theta(s)$

Goodness of Actor

- An episode is considered as a trajectory τ
  - $\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$
  - $R(\tau) = \sum_{t=1}^{T} r_t$
  - If you use an actor to play the game, each τ has a probability to be sampled
    - The probability depends on actor parameter θ: $P(\tau|\theta)$

$$\bar{R}_\theta = \boxed{\sum_{\tau}} R(\tau) \boxed{P(\tau|\theta)} \approx \boxed{\frac{1}{N}\sum_{n=1}^{N}} R(\tau^n)$$

Sum over all possible trajectory

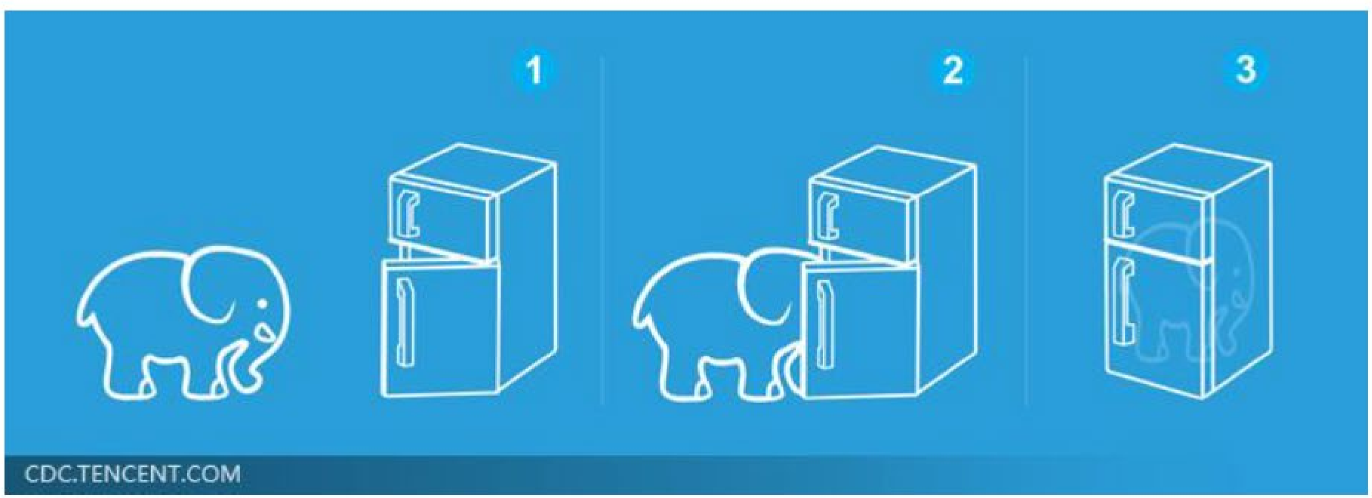
Use π_θ to play the game N times, obtain $\{\tau^1, \tau^2, \cdots, \tau^N\}$

Sampling τ from $P(\tau|\theta)$ N times

Three Steps for Deep Learning

Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

Deep Learning is so simple ……



Gradient Ascent

- Problem statement

$$\theta^* = arg \max_\theta \bar{R}_\theta \qquad \bar{R}_\theta = \sum_\tau R(\tau)P(\tau|\theta)$$

- Gradient ascent
  - Start with θ^0
  - $\theta^1 \leftarrow \theta^0 + \eta\nabla\bar{R}_{\theta^0}$
  - $\theta^2 \leftarrow \theta^1 + \eta\nabla\bar{R}_{\theta^1}$
  - ……

$$\theta = \{w_1, w_2, \cdots, b_1, \cdots\}$$

$$\nabla\bar{R}_\theta = \begin{bmatrix} \partial\bar{R}_\theta/\partial w_1 \\ \partial\bar{R}_\theta/\partial w_2 \\ \vdots \\ \partial\bar{R}_\theta/\partial b_1 \\ \vdots \end{bmatrix}$$

Gradient Ascent

$$\bar{R}_\theta = \sum_\tau R(\tau)P(\tau|\theta) \qquad \nabla\bar{R}_\theta = ?$$

$$\nabla\bar{R}_\theta = \sum_\tau R(\tau)\nabla P(\tau|\theta) = \sum_\tau R(\tau)P(\tau|\theta)\frac{\nabla P(\tau|\theta)}{P(\tau|\theta)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

$$= \boxed{\sum_\tau} R(\tau)\boxed{P(\tau|\theta)}\nabla log P(\tau|\theta)$$

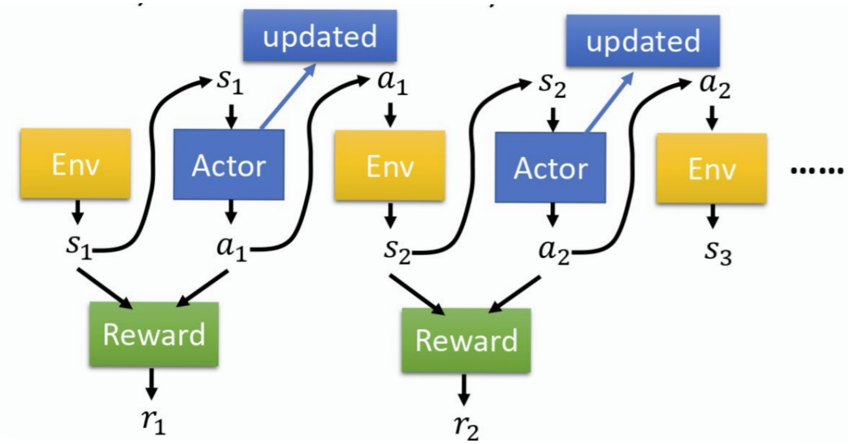
$$\boxed{\frac{d log\big(f(x)\big)}{dx} = \frac{1}{f(x)}\frac{df(x)}{dx}}$$

$$\approx \frac{1}{N}\sum_{n=1}^{N} R(\tau^n)\underline{\nabla log P(\tau^n|\theta)}$$

Use π_θ to play the game N times,
Obtain $\{\tau^1, \tau^2, \cdots, \tau^N\}$

$$\nabla log P(\tau|\theta) = ?$$

$$\tau = \{s_1, a_1, r_1, s_2, a_2, r_2, \cdots, s_T, a_T, r_T\}$$



$$P(\tau|\theta) =$$

$$p(s_1)p(a_1|s_1,\theta)p(r_1,s_2|s_1,a_1)p(a_2|s_2,\theta)p(r_2,s_3|s_2,a_2)\cdots$$

$$= p(s_1)\prod_{t=1}^{T} p(a_t|s_t,\theta)p(r_t,s_{t+1}|s_t,a_t)$$

$$log P(\tau|\theta)$$

$$= log p(s_1) + \sum_{t=1}^{T} log p(a_t|s_t,\theta) + log p(r_t,s_{t+1}|s_t,a_t)$$

$$\nabla log P(\tau|\theta) = \sum_{t=1}^{T} \nabla log p(a_t|s_t,\theta)$$

Ignore the terms not related to θ

Policy Gradient

$$\nabla \log P(\tau|\theta) = \sum_{t=1}^{T} \nabla \log p(a_t|s_t,\theta)$$

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N} R(\tau^n) \nabla \log P(\tau^n|\theta) = \frac{1}{N}\sum_{n=1}^{N} R(\tau^n) \sum_{t=1}^{T_n} \nabla \log p(a_t^n|s_t^n,\theta)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n) \nabla \log p(a_t^n|s_t^n,\theta)$$

What if we replace $R(\tau^n)$ with r_t^n ……

If in τ^n machine takes a_t^n when seeing s_t^n in

$R(\tau^n)$ is positive → Tuning θ to increase $p(a_t^n|s_t^n)$

$R(\tau^n)$ is negative → Tuning θ to decrease $p(a_t^n|s_t^n)$

It is very important to consider the cumulative reward $R(\tau^n)$ of the whole trajectory τ^n instead of immediate reward r_t^n

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla log p_\theta(\tau)]$$

Policy Gradient

Given policy π_θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

$\vdots$ $\vdots$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

$\vdots$ $\vdots$

only used once

Update Model

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p_\theta(a_t^n | s_t^n)$$

Data Collection

References

- Hung-yi Lee, Machine learning courses on NTU, 2020
  - https://speech.ee.ntu.edu.tw/~hylee/ml/2020-spring.html
- David Silver, UCL courses on RL
  - https://www.davidsilver.uk/teaching/