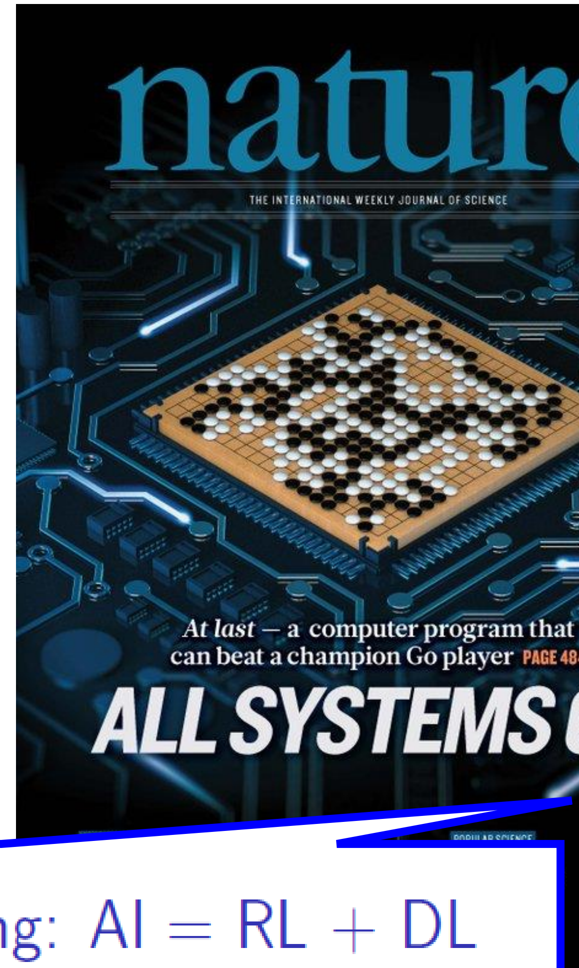
Deep Reinforcement Learning

Scratching the surface

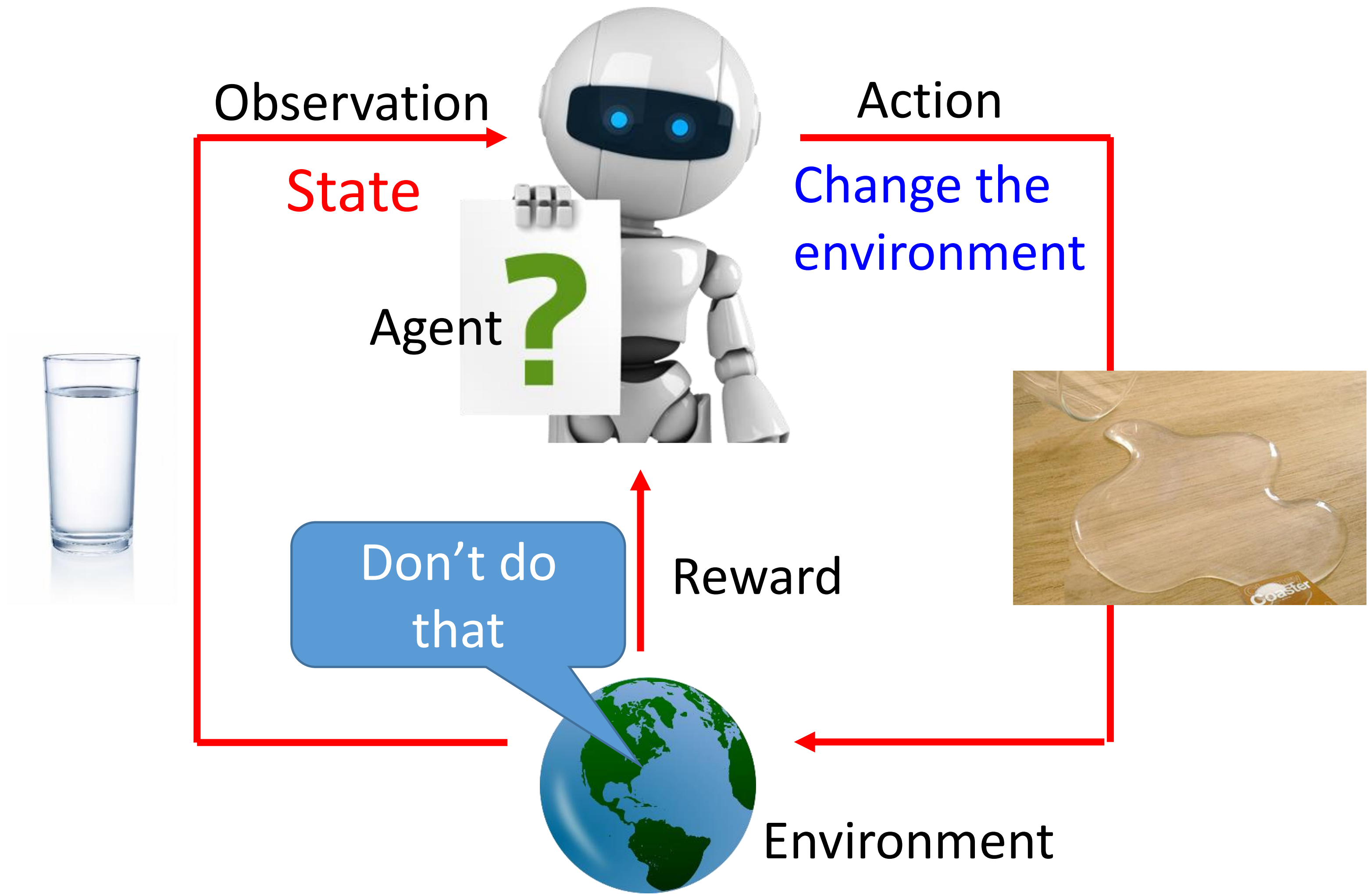
Deep Reinforcement Learning



Deep Reinforcement Learning: AI = RL + DL

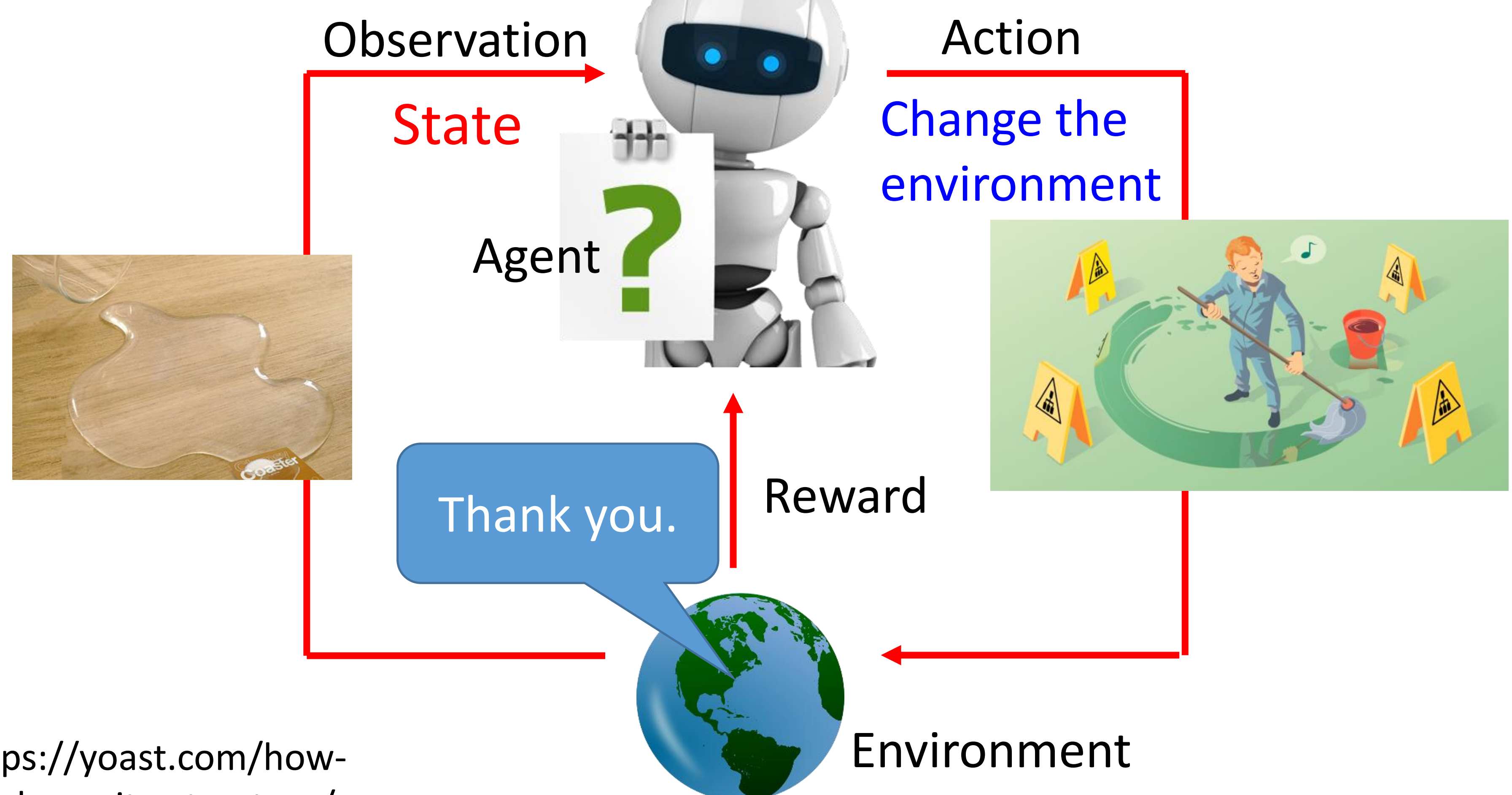
David Silver

Scenario of Reinforcement Learning



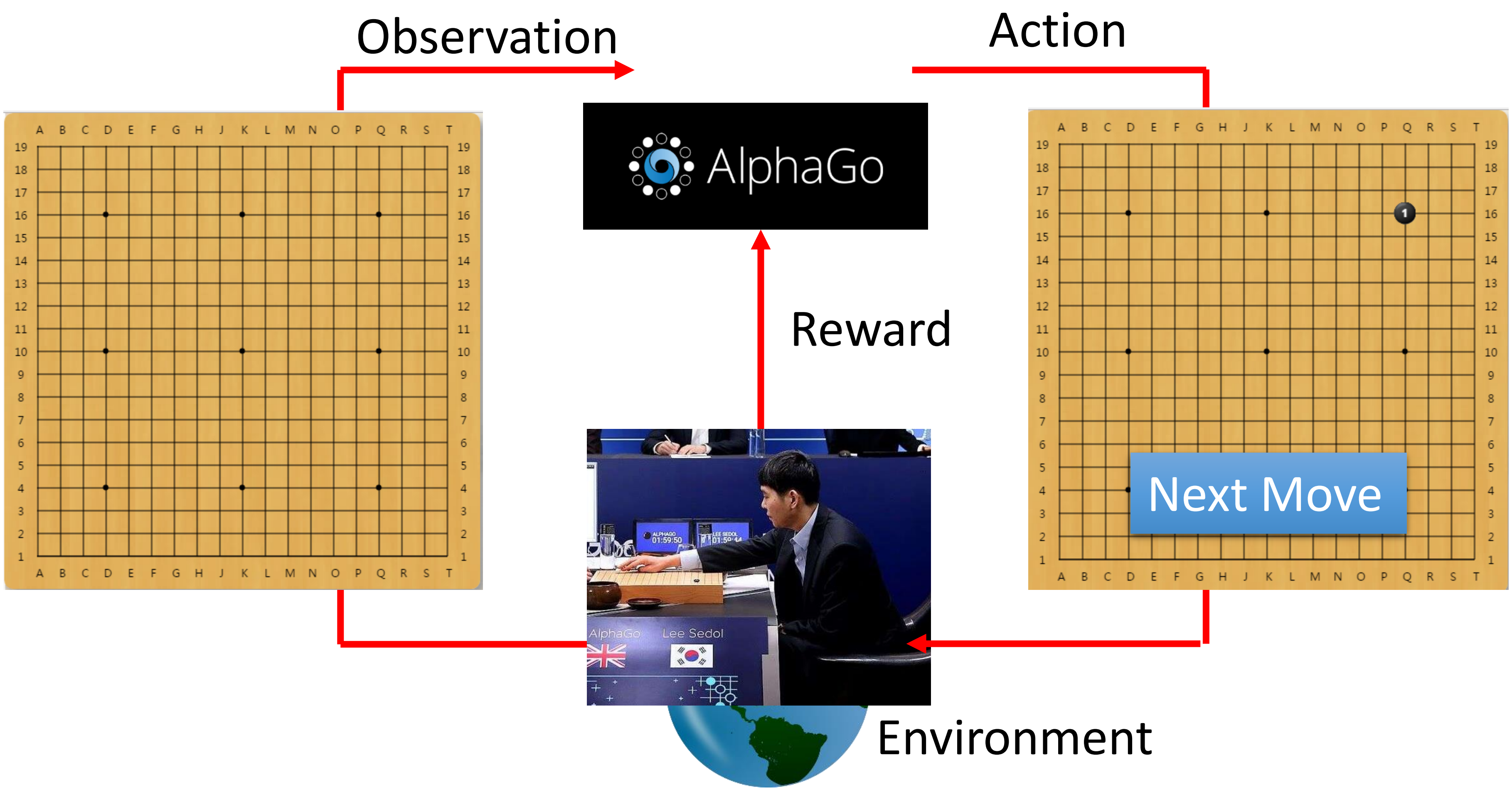
Scenario of Reinforcement Learning

Agent learns to take actions to maximize expected reward.



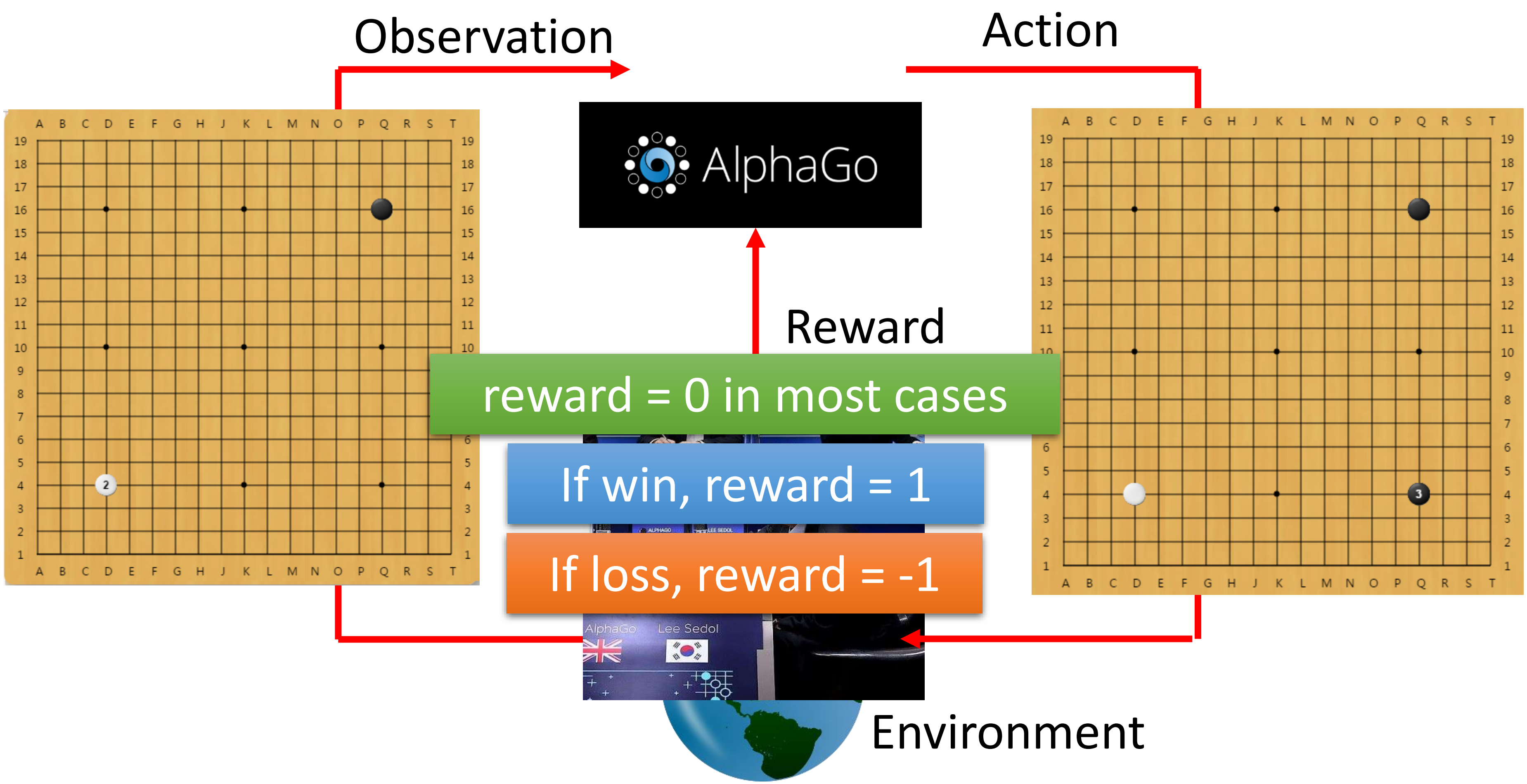
https://yoast.com/how-to-clean-site-structure/

Learning to paly Go



Learning to paly Go

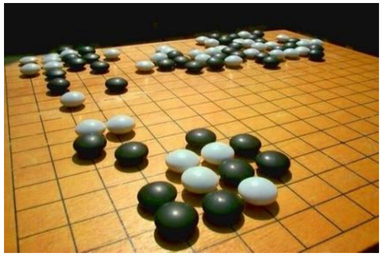
Agent learns to take actions to maximize expected reward.



Learning to paly Go
- Supervised v.s. Reinforcement

- Supervised: **Learning from teacher**

Next move: "5-5"



Next move: "3-3"

- Reinforcement Learning **Learning from experience**

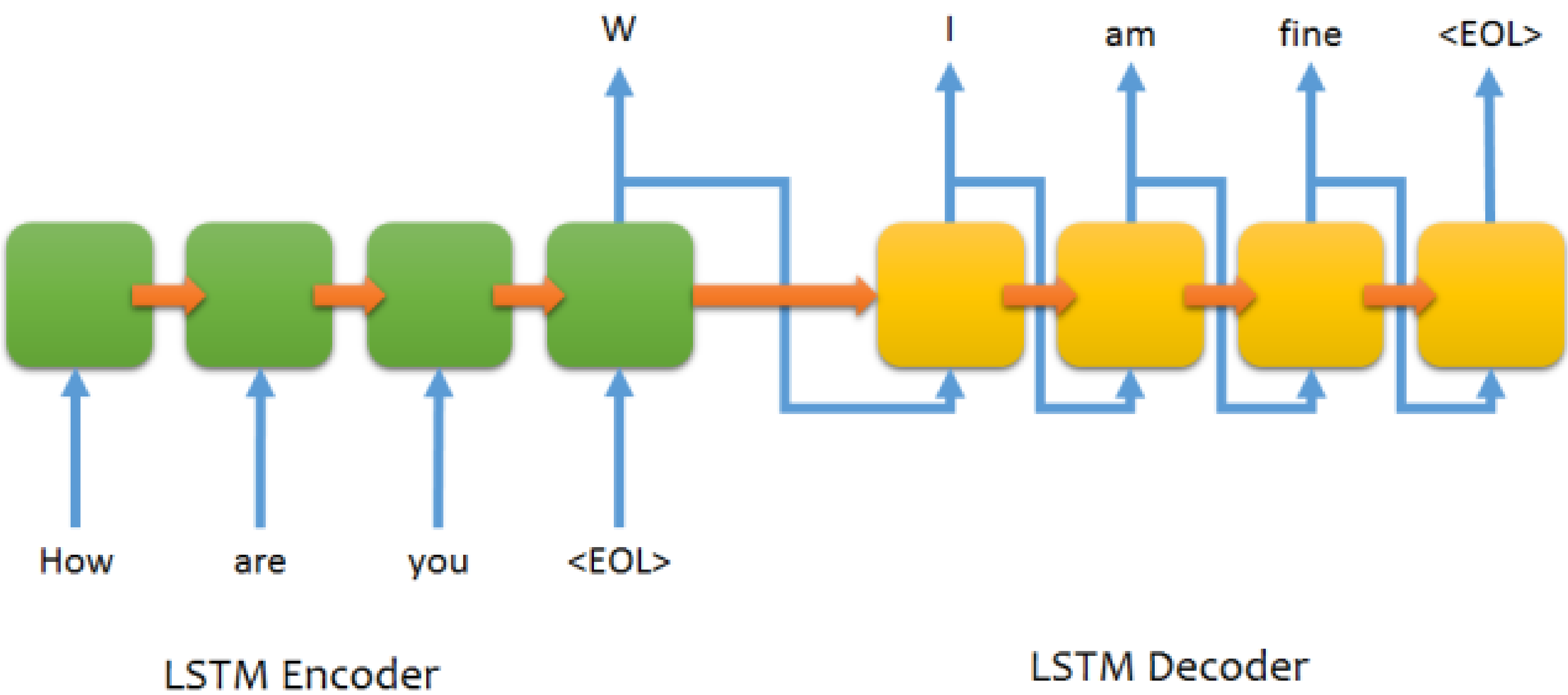
First move → …… many moves …… → Win!

(Two agents play with each other.)

Alpha Go is supervised learning + reinforcement learning.

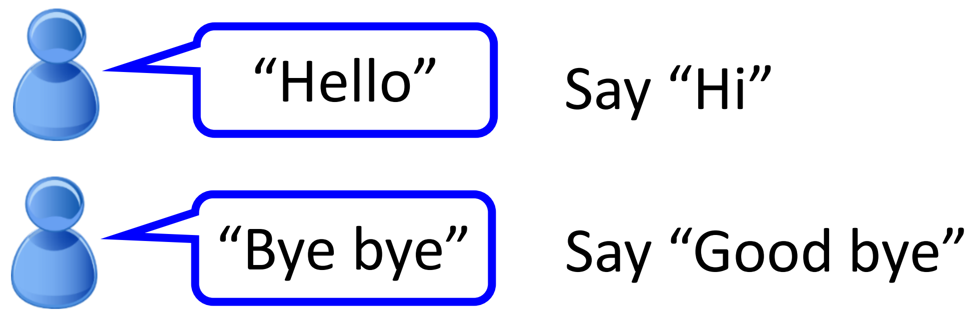
Learning a chat-bot

- Sequence-to-sequence learning

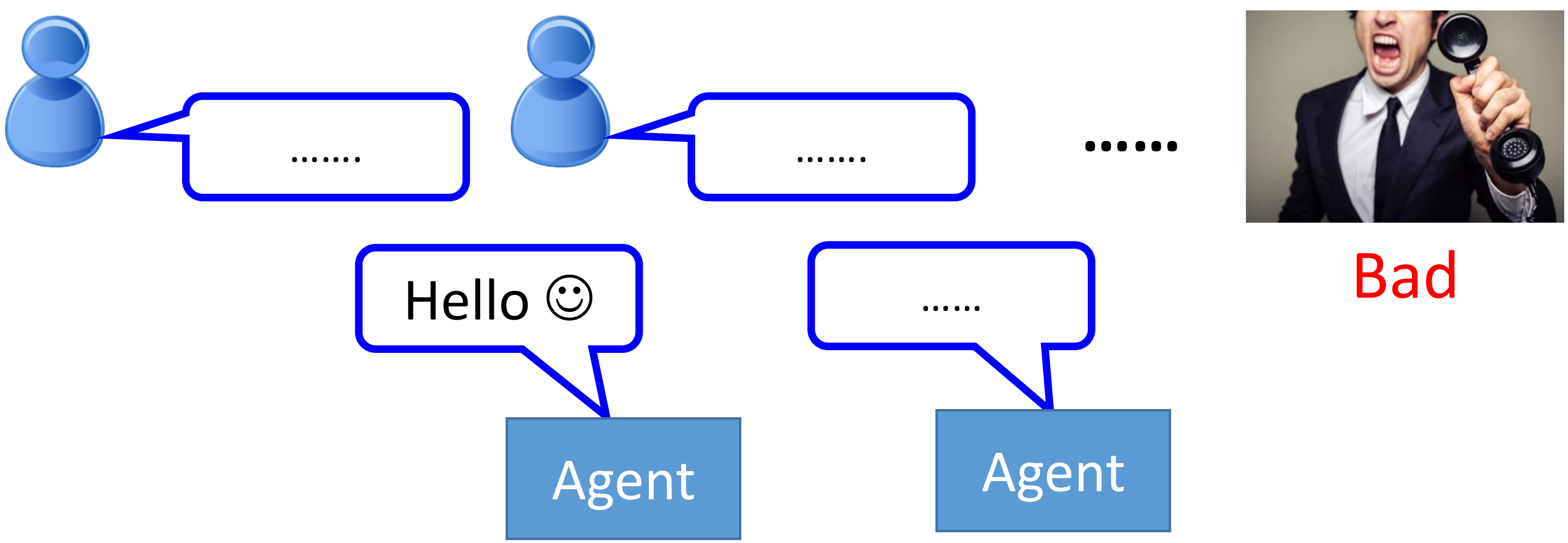


Learning a chat-bot
- Supervised v.s. Reinforcement

- Supervised



- Reinforcement



Learning a chat-bot - Reinforcement Learning

- Let two agents talk to each other (sometimes generate good dialogue, sometimes bad)

How old are you?

See you.

See you.

See you.

How old are you?

I am 16.

I though you were 12.

What make you think so?

Learning a chat-bot
- Reinforcement Learning

- By this approach, we can generate a lot of dialogues.
- Use some pre-defined rules to evaluate the goodness of a dialogue

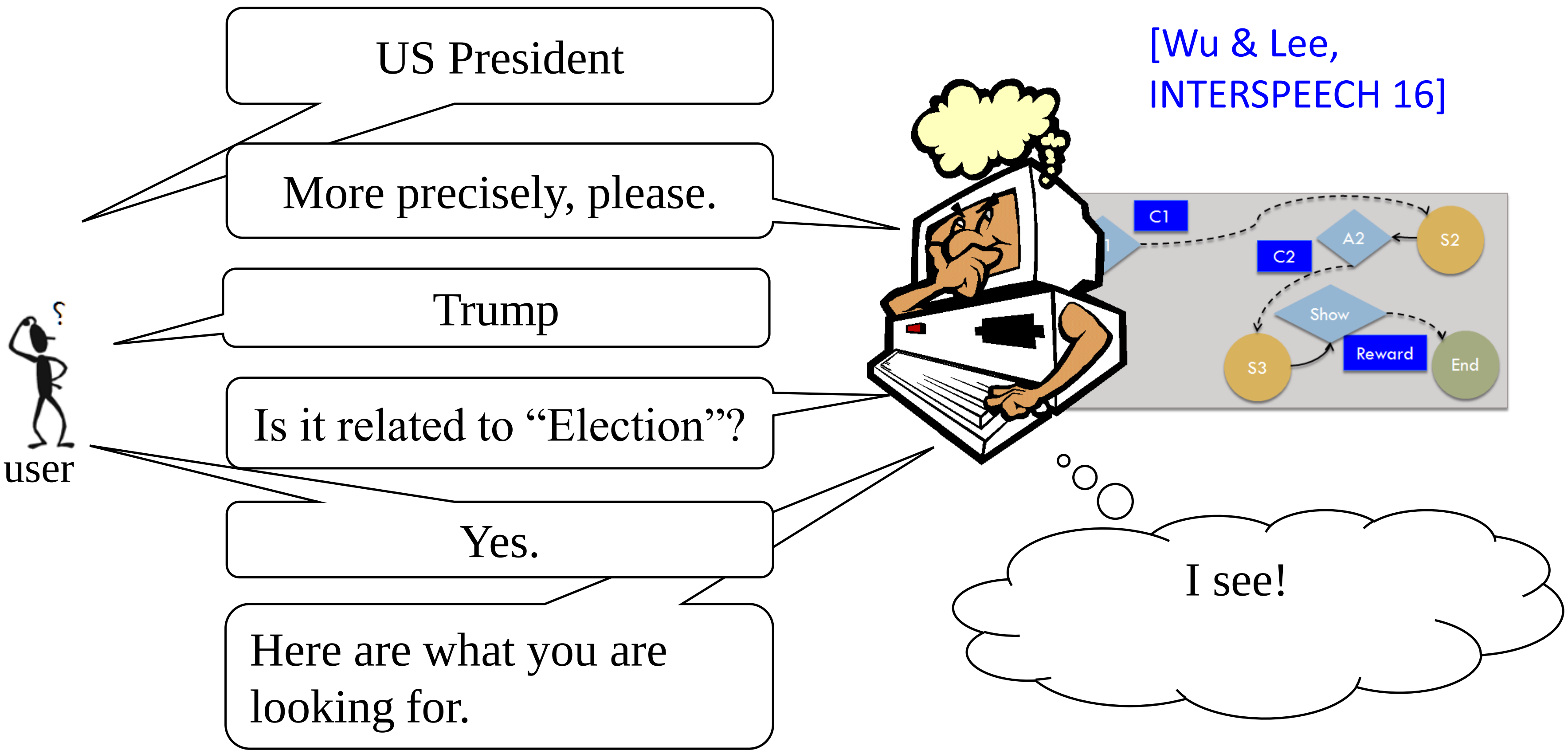
Machine learns from the evaluation

Deep Reinforcement Learning for Dialogue Generation
https://arxiv.org/pdf/1606.01541v3.pdf

More applications

- Interactive retrieval

[Wu & Lee, INTERSPEECH 16]



More applications

- Flying Helicopter
  - https://www.youtube.com/watch?v=0JL04JJjocc
- Driving
  - https://www.youtube.com/watch?v=0xo1Ldx3L5Q
- Google Cuts Its Giant Electricity Bill With DeepMind-Powered AI
  - http://www.bloomberg.com/news/articles/2016-07-19/google-cuts-its-giant-electricity-bill-with-deepmind-powered-ai
- Text generation
  - Hongyu Guo, “Generating Text with Deep Reinforcement Learning”, NIPS, 2015
  - Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba, “Sequence Level Training with Recurrent Neural Networks”, ICLR, 2016

Example: Playing Video Game

- Widely studies:
  - Gym: https://gym.openai.com/
  - Universe: https://openai.com/blog/universe/

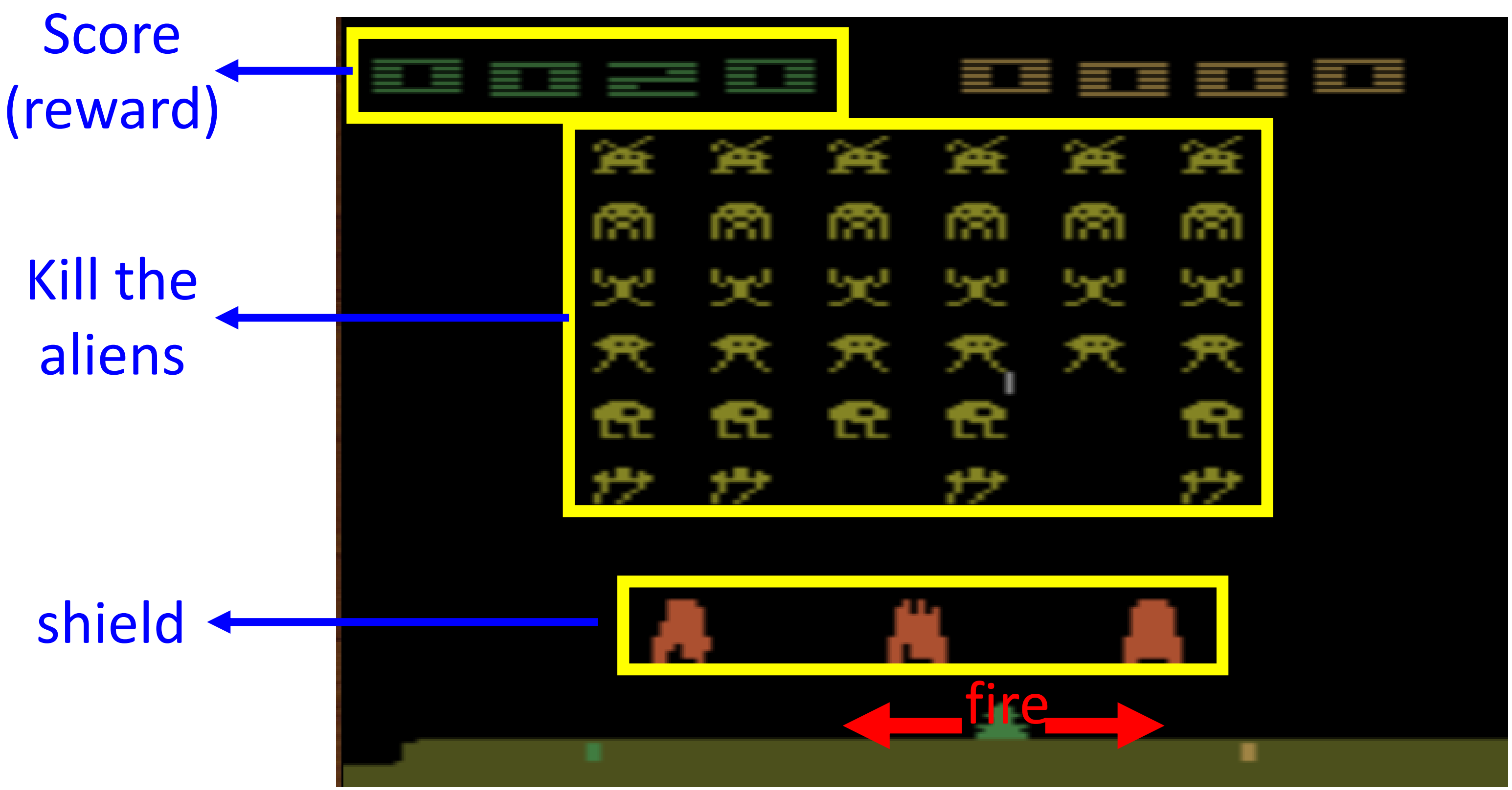
Machine learns to play video games as human players

- ➢ What machine observes is pixels
- ➢ Machine learns to take proper action itself

Example: Playing Video Game

- Space invader

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



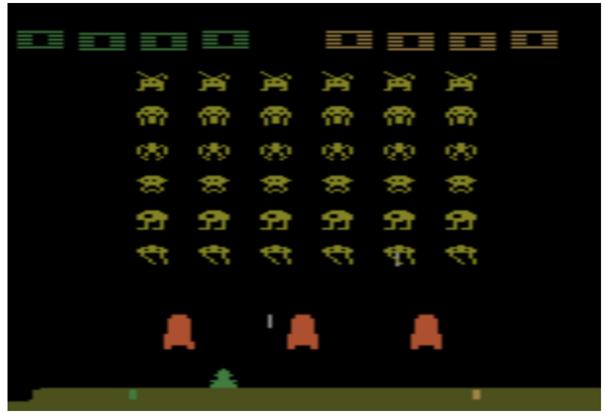
Example: Playing Video Game

- Space invader
  - Play yourself: http://www.2600online.com/spaceinvaders.html
  - How about machine: https://gym.openai.com/evaluations/eval_Eduozx4HRyqgTCVk9ltw

Example: Playing Video Game

Start with observation s_1

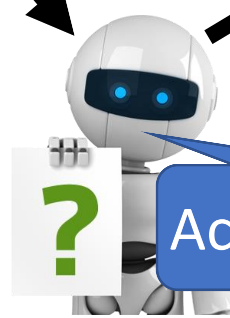
Observation s_2



Observation s_3

Action a_1: "right"

Obtain reward $r_1 = 0$



Action a_2 : "fire"

(kill an alien)

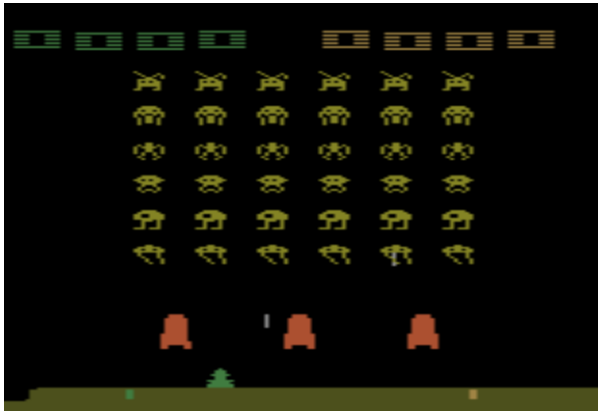
Obtain reward $r_2 = 5$

Usually there is some randomness in the environment

Example: Playing Video Game

Start with observation s_1

Observation s_2



Observation s_3

After many turns

Game Over
(spaceship destroyed)

Obtain reward r_T

Action a_T

This is an ***episode***.

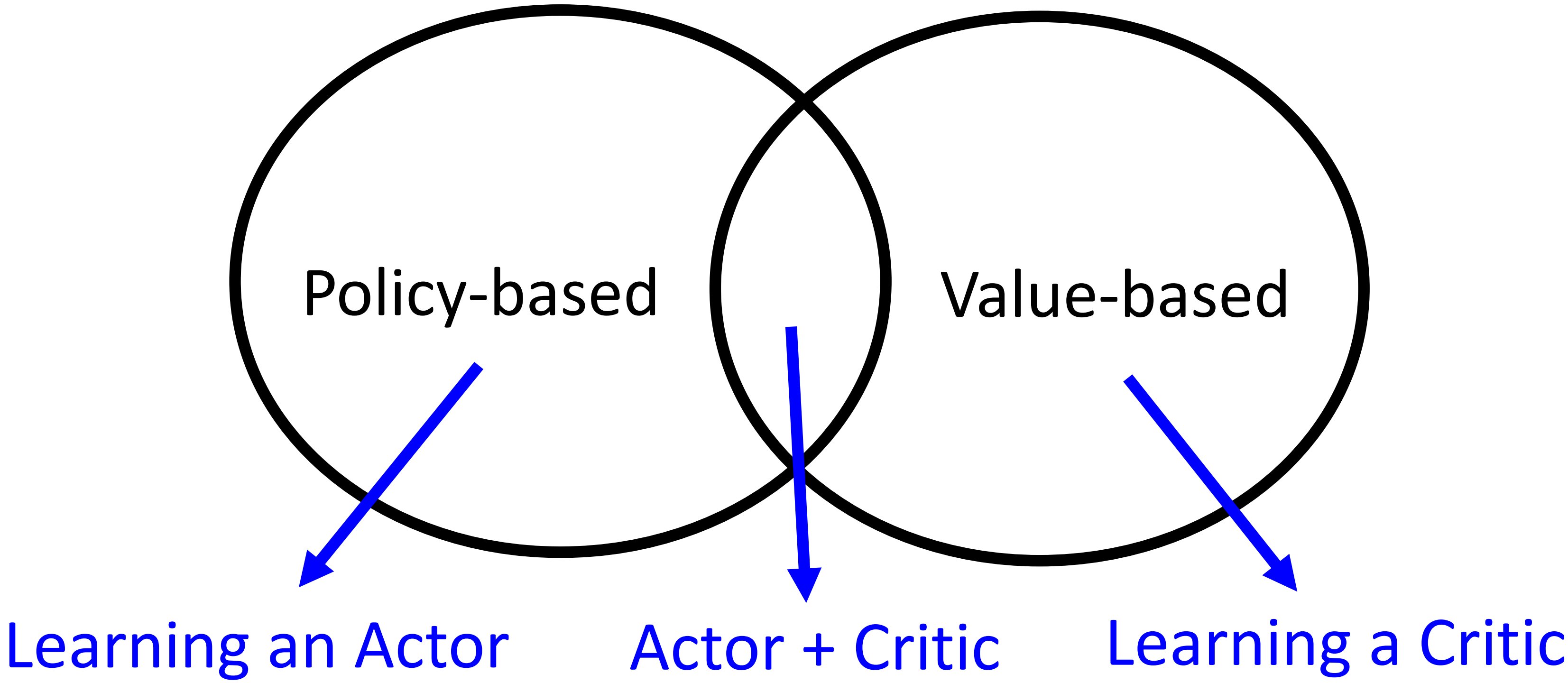
Learn to maximize the expected cumulative reward per episode

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

Outline

Alpha Go: policy-based + value-based + model-based



Asynchronous Advantage Actor-Critic (A3C)

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning", ICML, 2016

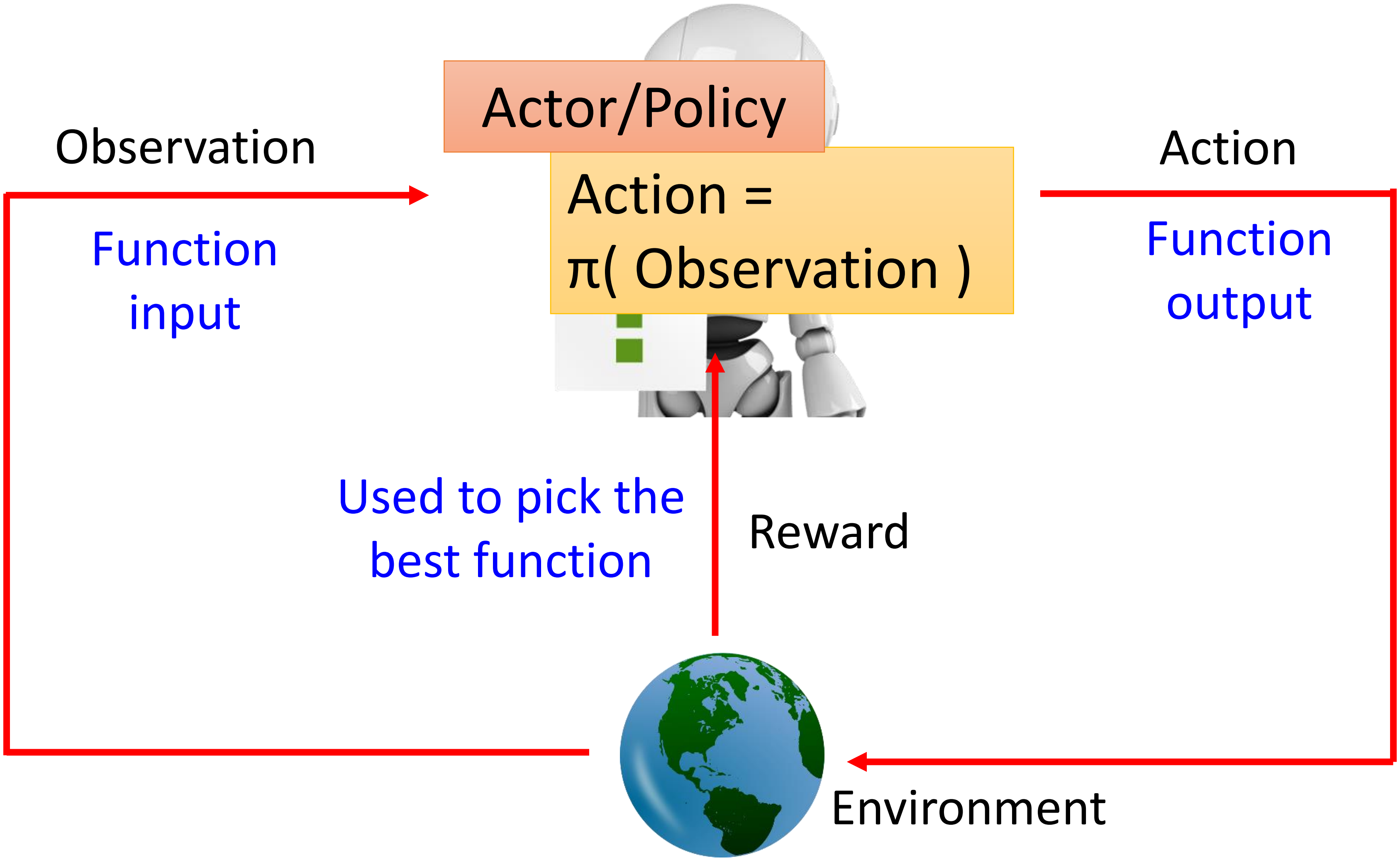
To learn deep reinforcement learning ……

- Textbook: Reinforcement Learning: An Introduction
  - https://webdocs.cs.ualberta.ca/~sutton/book/the-book.html
- Lectures of David Silver
  - http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html (10 lectures, 1:30 each)
  - http://videolectures.net/rldm2015_silver_reinforcement_learning/ (Deep Reinforcement Learning)
- Lectures of John Schulman
  - https://youtu.be/aUrX-rP_ss4

Policy-based Approach

Learning an Actor

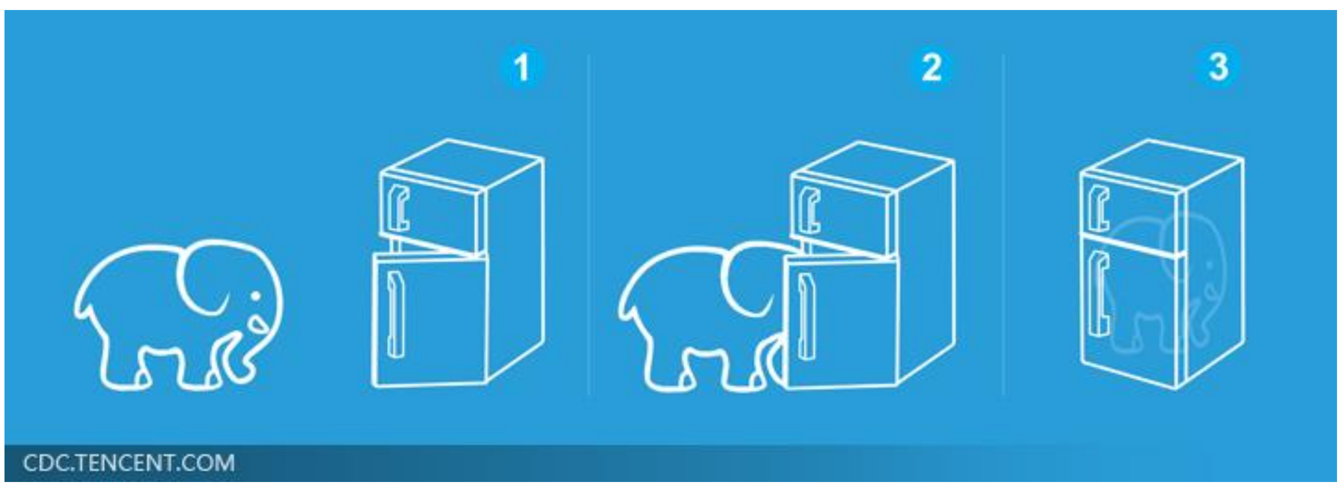
Machine Learning
≈ Looking for a Function



Three Steps for Deep Learning

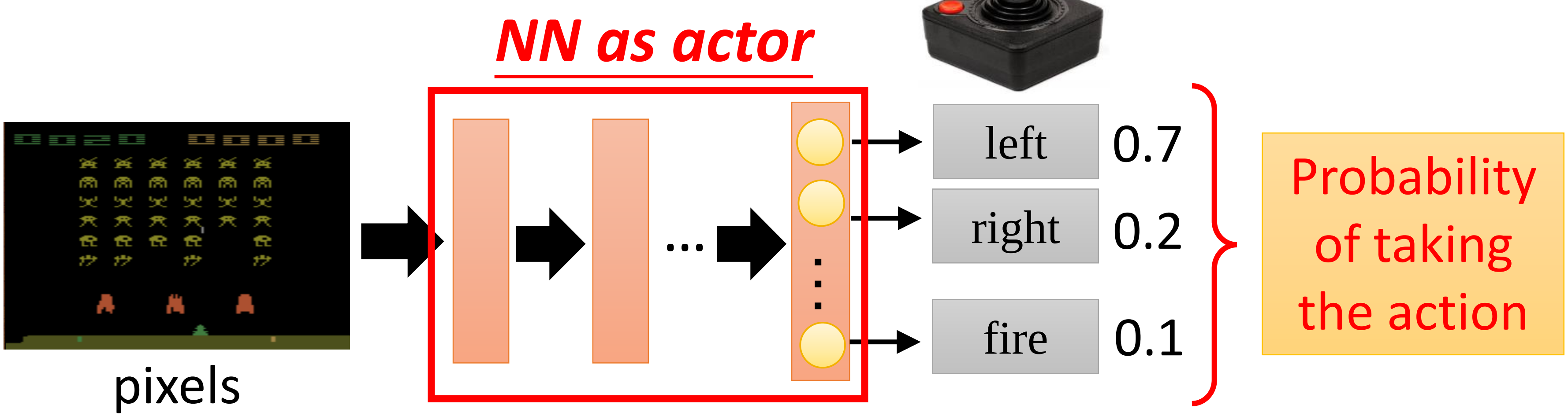
Step 1: Neural Network as Actor → Step 2: goodness of function → Step 3: pick the best function

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



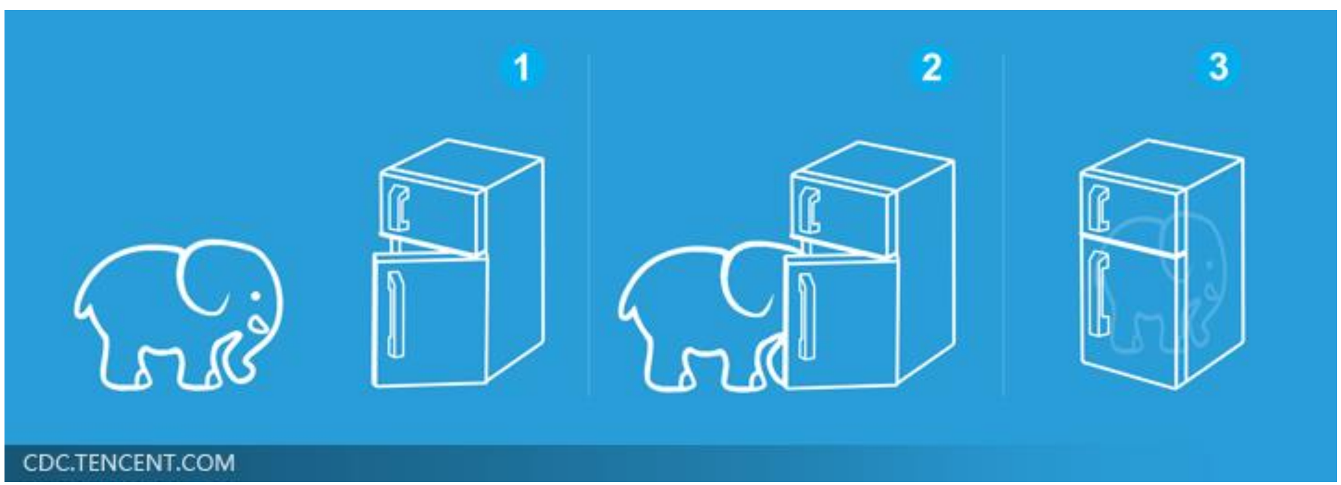
What is the benefit of using network instead of lookup table?

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 ……



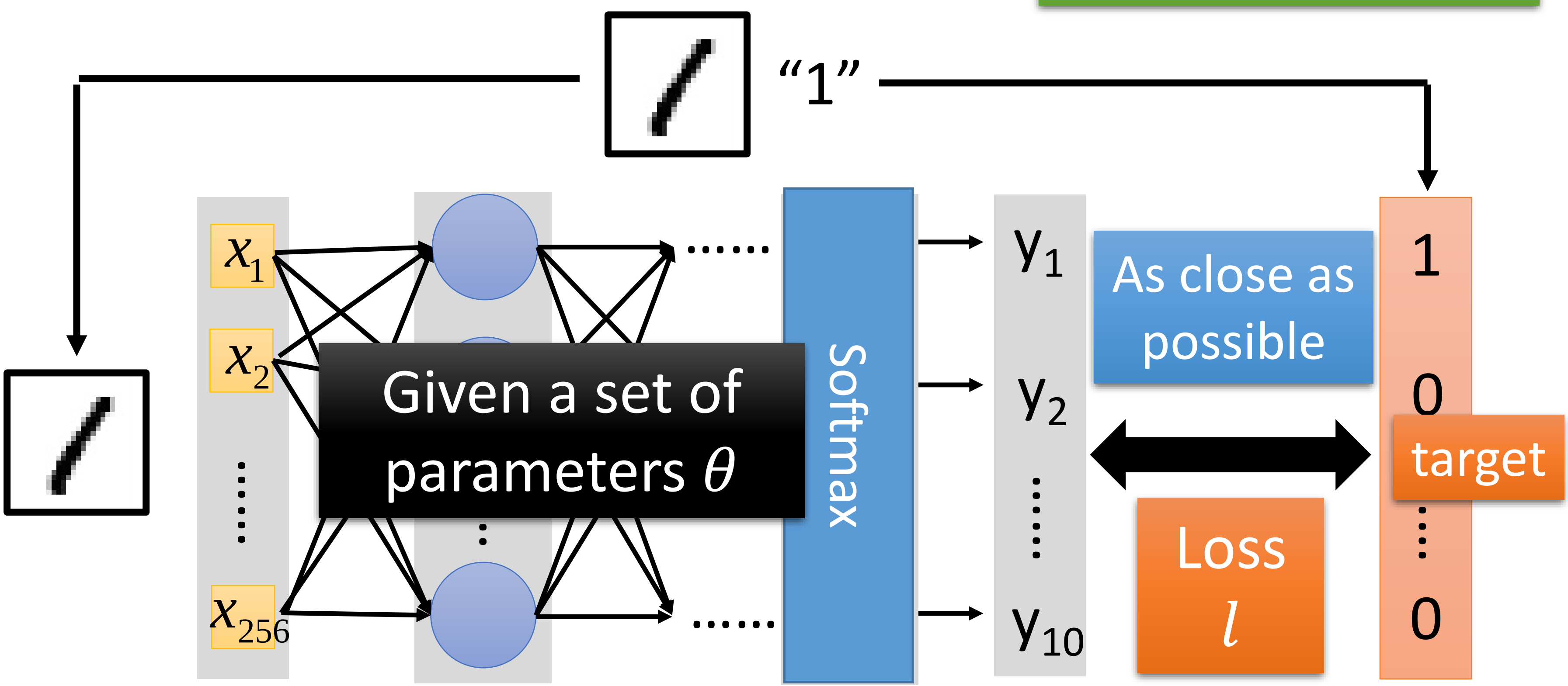
Goodness of Actor

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

- Review: Supervised learning

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

Training Example



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

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

$\bar{R}_\theta$ 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: $P(\tau|\theta)$

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

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

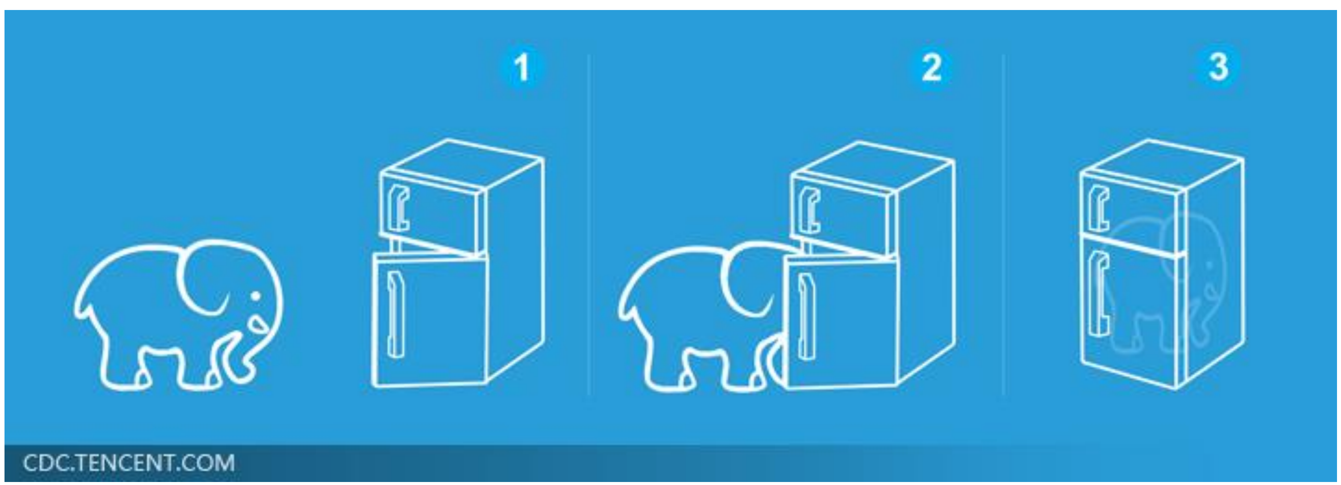
Sampling τ from $P(\tau|\theta)$ N times

Sum over all possible trajectory

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.

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

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

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

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

Gradient Ascent

$\nabla logP(\tau|\theta) = ?$

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

$P(\tau|\theta) =$

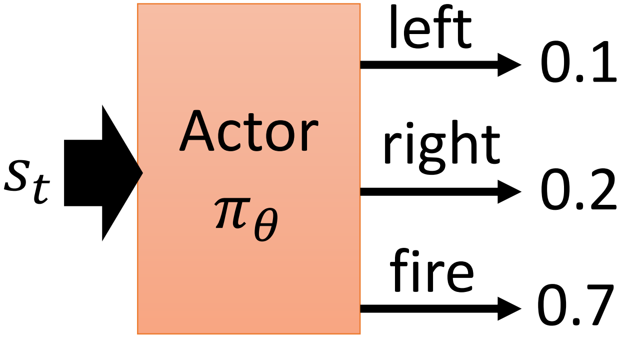
$$p(s_1)p(a_1|s_1,\theta)p(r_1,s_2|s_1,a_1)p(a_2|s_2,\theta)p(r_2,s_3|s_2,a_2)\cdots$$

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

not related to your actor

Control by your actor π_θ

$p(a_t = \text{"}fire\text{"}|s_t,\theta)$
$= 0.7$



Gradient Ascent

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

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

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

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

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

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

Ignore the terms not related to θ

Gradient Ascent

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

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

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

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

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

Gradient Ascent

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

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

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

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

Why divided by $p(a_t^n|s_t^n,\theta)$?

e.g. in the sampling data … s has been seen in $\tau^{13}, \tau^{15}, \tau^{17}, \tau^{33}$

In τ^{13}, take action a	$R(\tau^{13}) = 2$	In τ^{15}, take action b	$R(\tau^{15}) = 1$
In τ^{17}, take action b	$R(\tau^{17}) = 1$	In τ^{33}, take action b	$R(\tau^{33}) = 1$

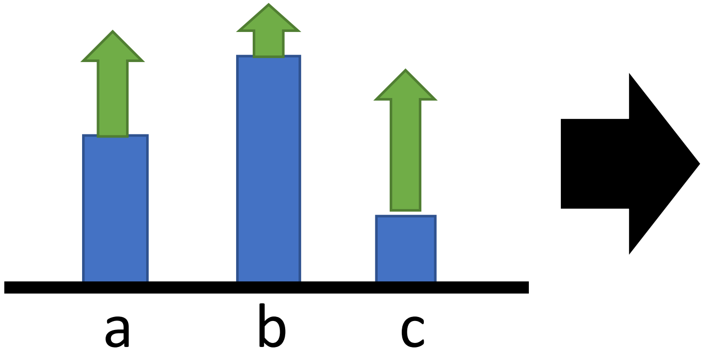
Add a Baseline

It is possible that $R(\tau^n)$ is always positive.

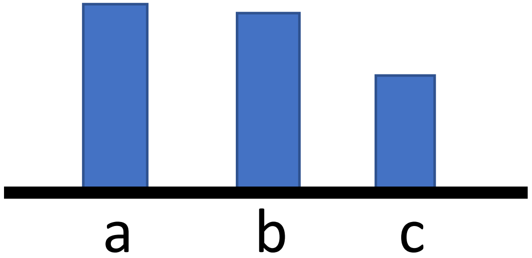
$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

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

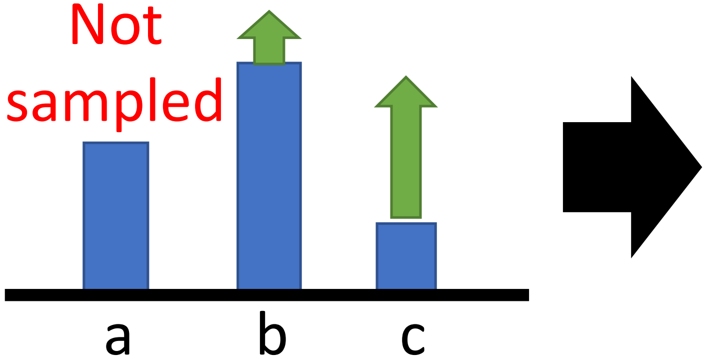
Ideal case

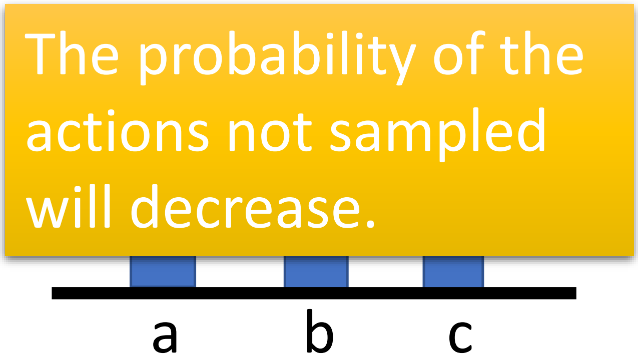


It is probability …



Sampling ……





Value-based Approach

Learning a Critic

Critic

- A critic does not determine the action.
- Given an actor, it evaluates the how good the actor is

An actor can be found from a critic.

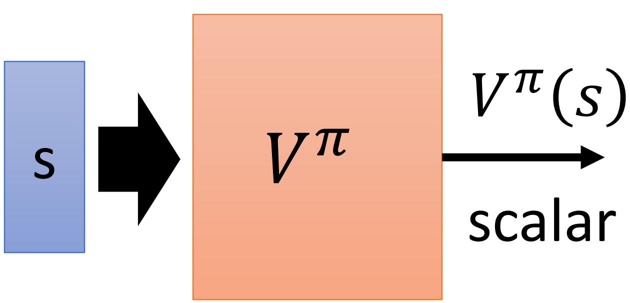
e.g. Q-learning

(not today)

http://combiboilersleeds.com/picaso/critics/critics-4.html

Three kinds of Critics

- A critic is a function depending on the actor π it is evaluated
  - The function is represented by a neural network
- State value function $V^\pi(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation (state) s

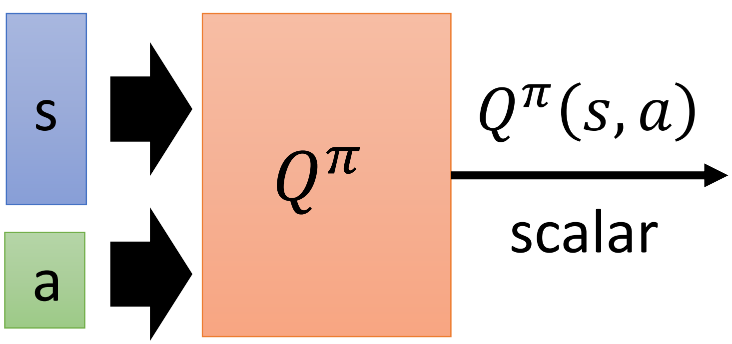


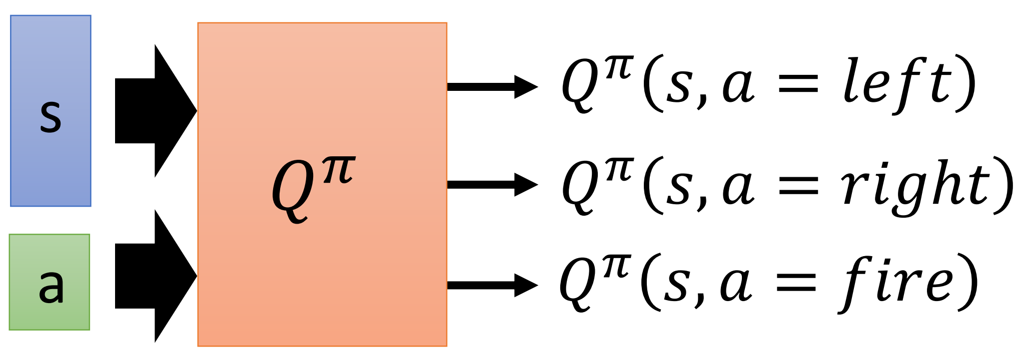
$V^\pi(s)$ is large

$V^\pi(s)$ is smaller

Three kinds of Critics

- State-action value function $Q^{\pi}(s,a)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation s and taking a





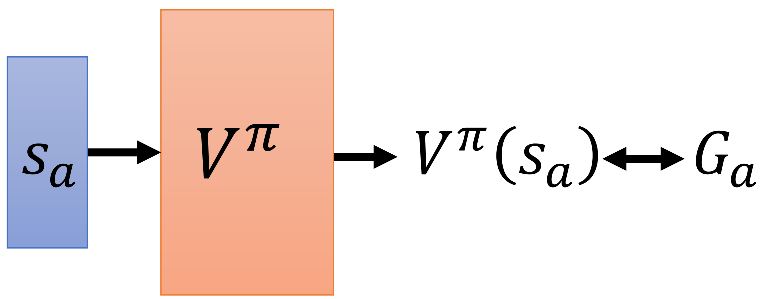
for discrete action only

How to estimate $V^{\pi}(s)$

- Monte-Carlo based approach
  - The critic watches π playing the game

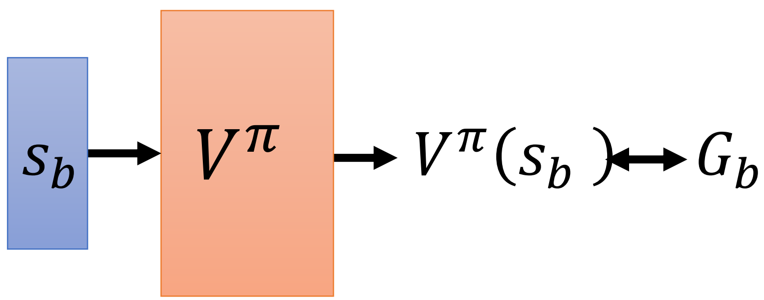
After seeing s_a,

Until the end of the episode, the cumulated reward is G_a



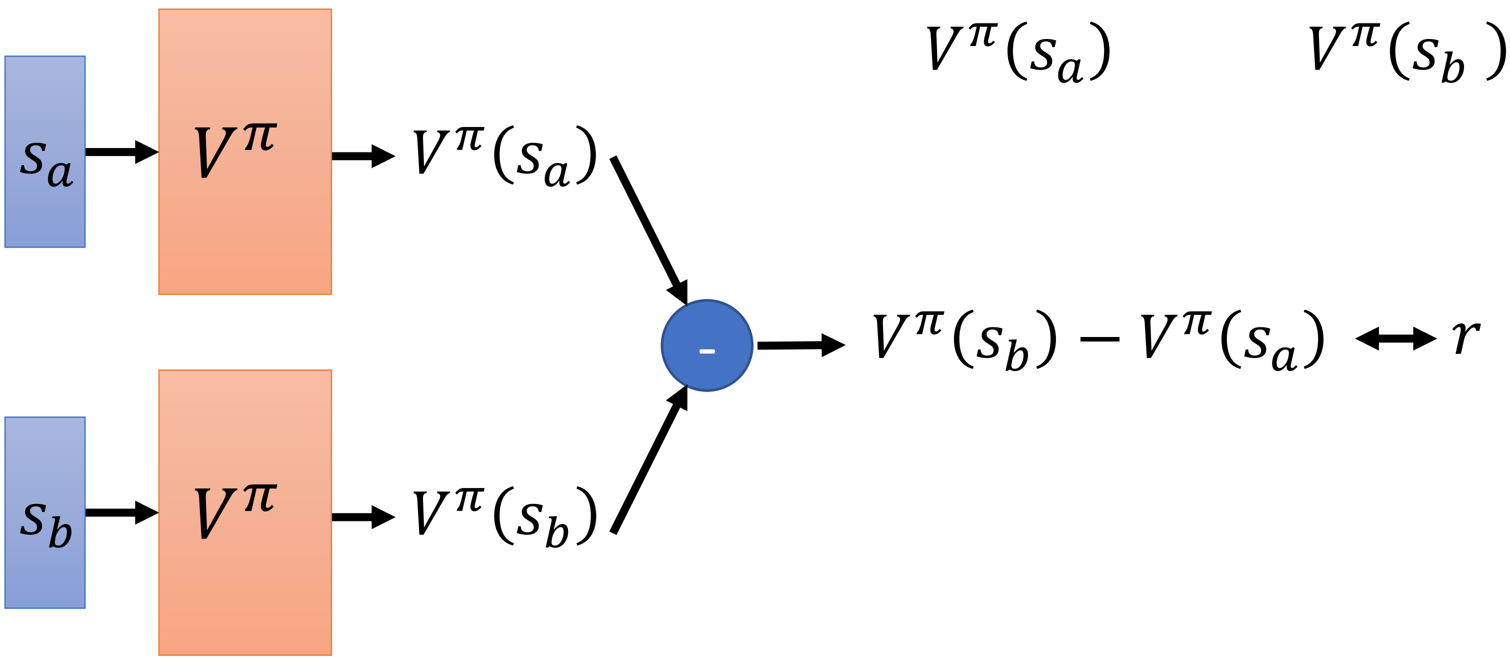
After seeing s_b,

Until the end of the episode, the cumulated reward is G_b



How to estimate $V^\pi(s)$

- Temporal-difference approach $\cdots s_a, a, r, s_b \cdots$



Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.

How to estimate $V^{\pi}(s)$

[Sutton, v2, Example 6.4]

- The critic has the following 8 episodes
  - $s_a, r = 0, s_b, r = 0$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 0$, END

$$V^{\pi}(s_b) = 3/4$$

$V^{\pi}(s_a) = ?$ 0? 3/4?

Monte-Carlo: $V^{\pi}(s_a) = 0$

Temporal-difference:

$$\underset{3/4}{V^{\pi}(s_a)} + \underset{0}{r} = \underset{3/4}{V^{\pi}(s_b)}$$

(The actions are ignored here.)

Deep Reinforcement Learning

Actor-Critic

Actor-Critic

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

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

Evaluated by critic

Advantage Function: $r_t^n - \left(V^{\pi_\theta}(s_t^n) - V^{\pi_\theta}(s_{t+1}^n)\right)$

Baseline is added

The reward r_t^n we truly obtain when taking action a_t^n

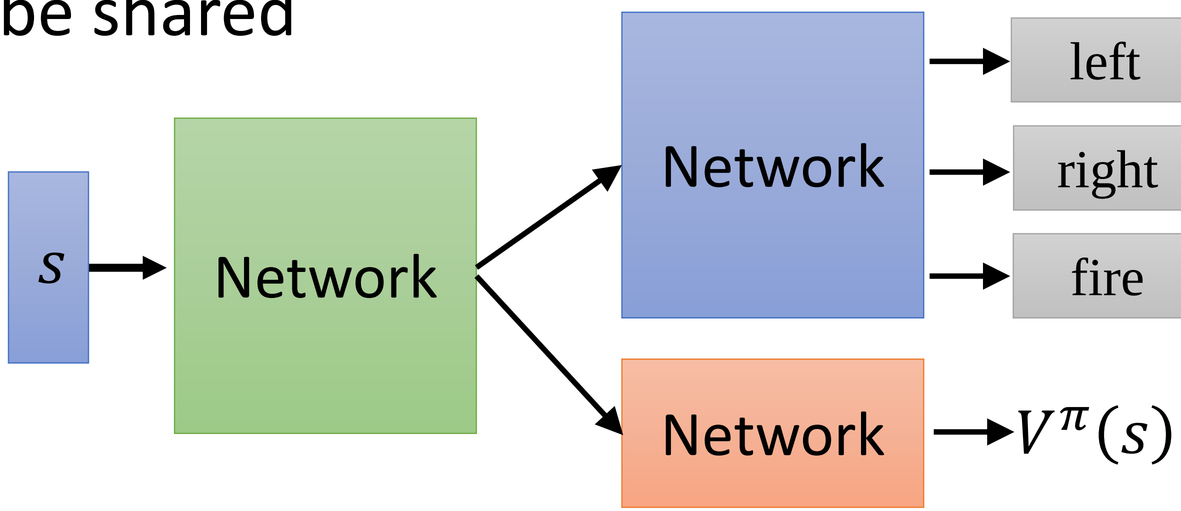
Expected reward r_t^n we obtain if we use actor π_θ

Positive advantage function $\Leftrightarrow$ Increasing the prob. of action a_t^n

Negative advantage function $\Leftrightarrow$ decreasing the prob. of action a_t^n

Actor-Critic

- Tips
  - The parameters of actor $\pi(s)$ and critic $V^\pi(s)$ can be shared

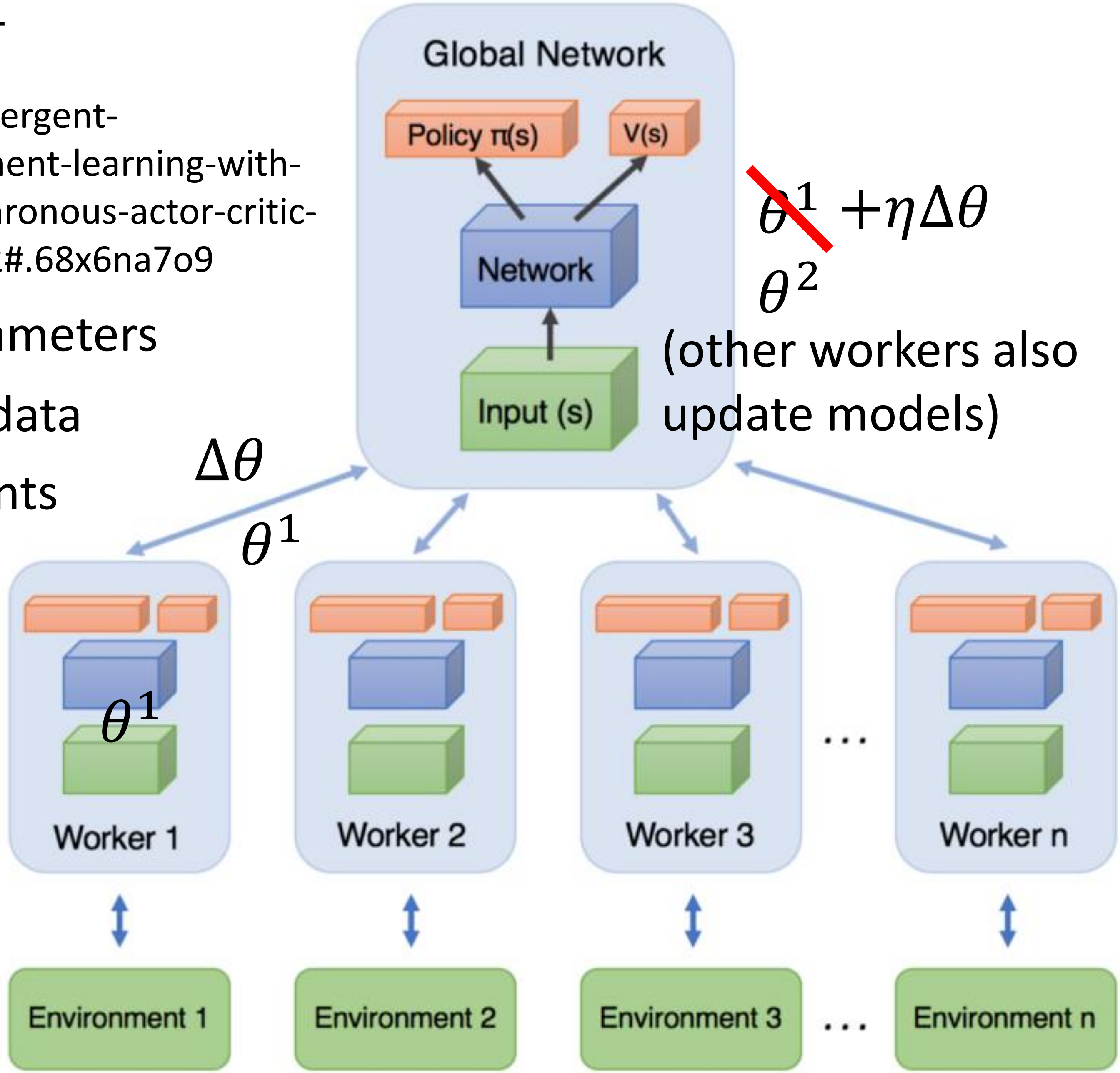


  - Use output entropy as regularization for $\pi(s)$
    - Larger entropy is preferred → exploration

Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



$\cancel{\theta^1} + \eta\Delta\theta$

θ^2

(other workers also update models)

$\Delta\theta$

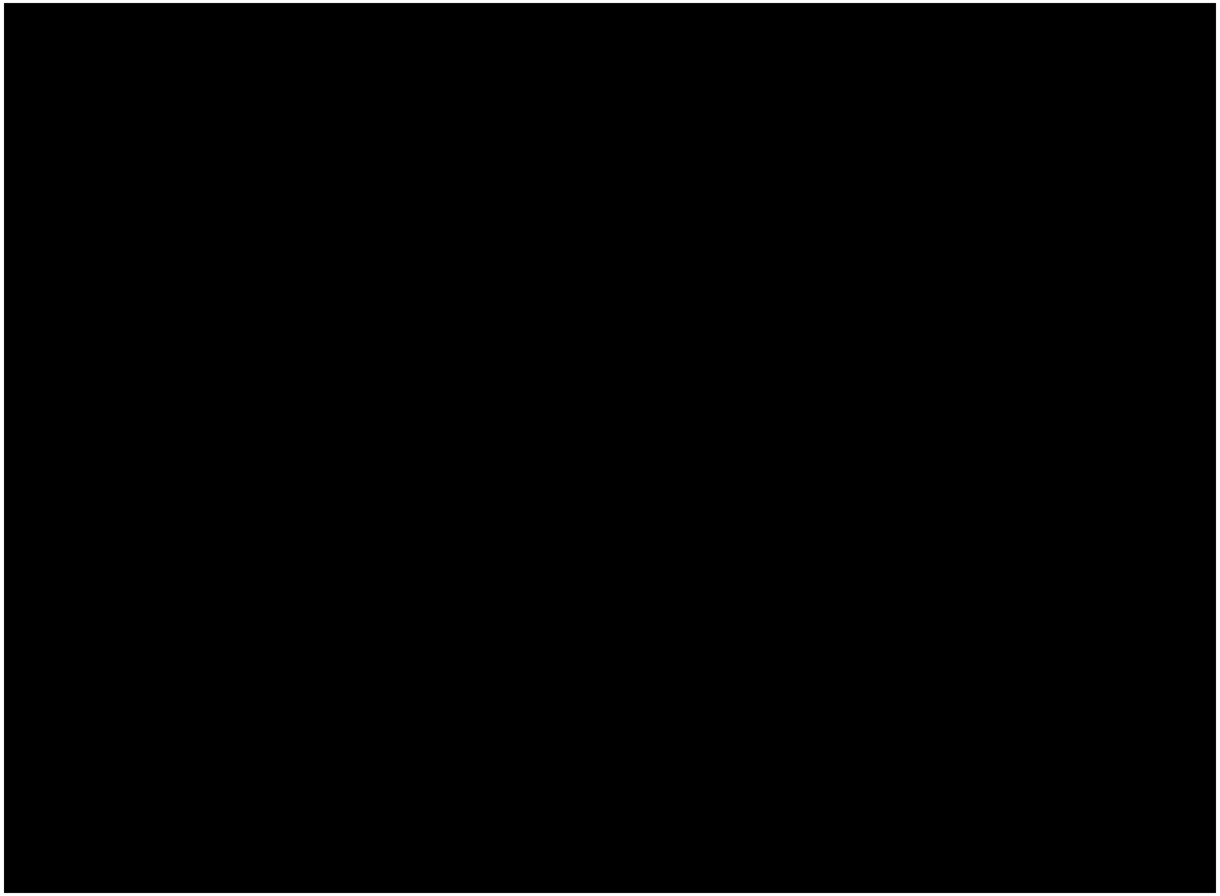
θ^1

θ^1

$\Delta\theta$

Demo of A3C

- DeepMind https://www.youtube.com/watch?v=nMR5mjCFZCw

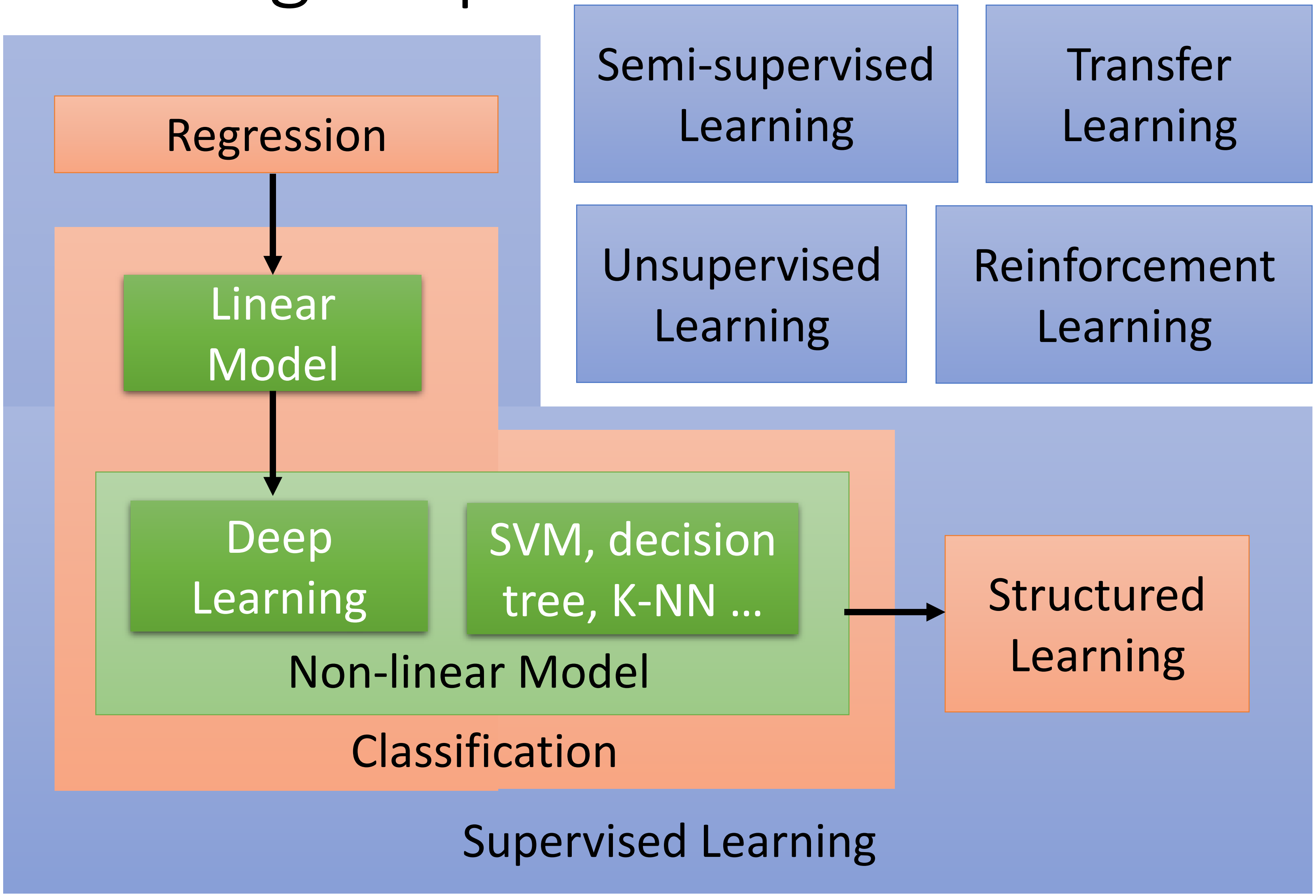


Demo of A3C

- DeepMind https://www.youtube.com/watch?v=0xo1Ldx3L5Q

Conclusion of This Semester

Learning Map



Acknowledgment

- 感謝 Larry Guo 同學發現投影片 上的符號錯誤