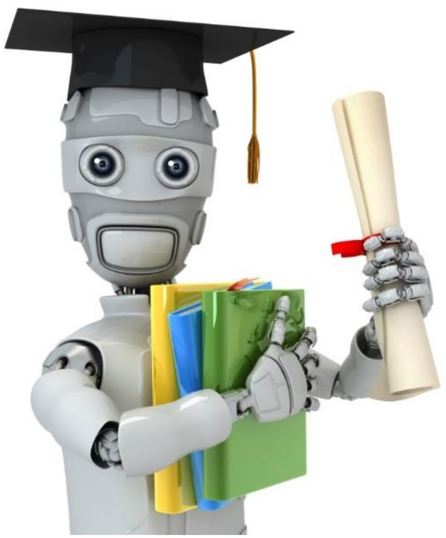


Machine Learning

强化学习 (Reinforcement Learning)

梁毅雄

yxliang@csu.edu.cn

Some materials from Hong-yi Lee, Andrew Ng and others

强化学习

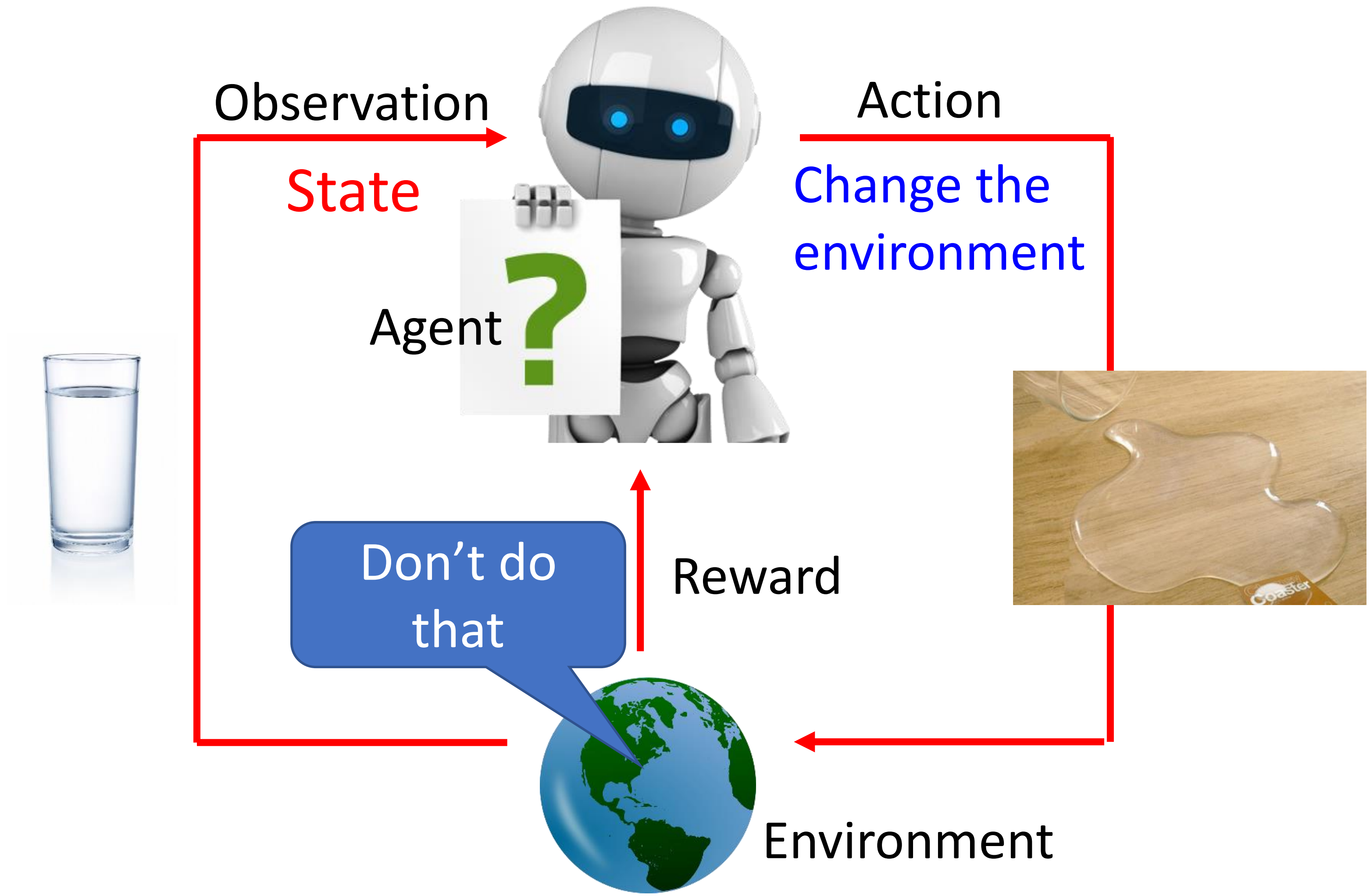
- 监督学习中对任意的x均给定了正确的标签y, 但在许多实际的应用中无法即时给出标准的正确答案，而需要反复去尝试后方可得到好的策略(实践出真知)

- 如何种西瓜: 选种，浇水、施肥、除草、杀虫等，我们其实不知道何时进行哪种操作能给出最正确的指导，因此也无法即时提供正确的监督信息去进行学习Action

- 我们并没有直接告诉Agent采取何种Action, 而是让Agent来自己发现最佳的Action。方案：取消监督信息，用reward奖励函数代替，表示对应action产生反馈

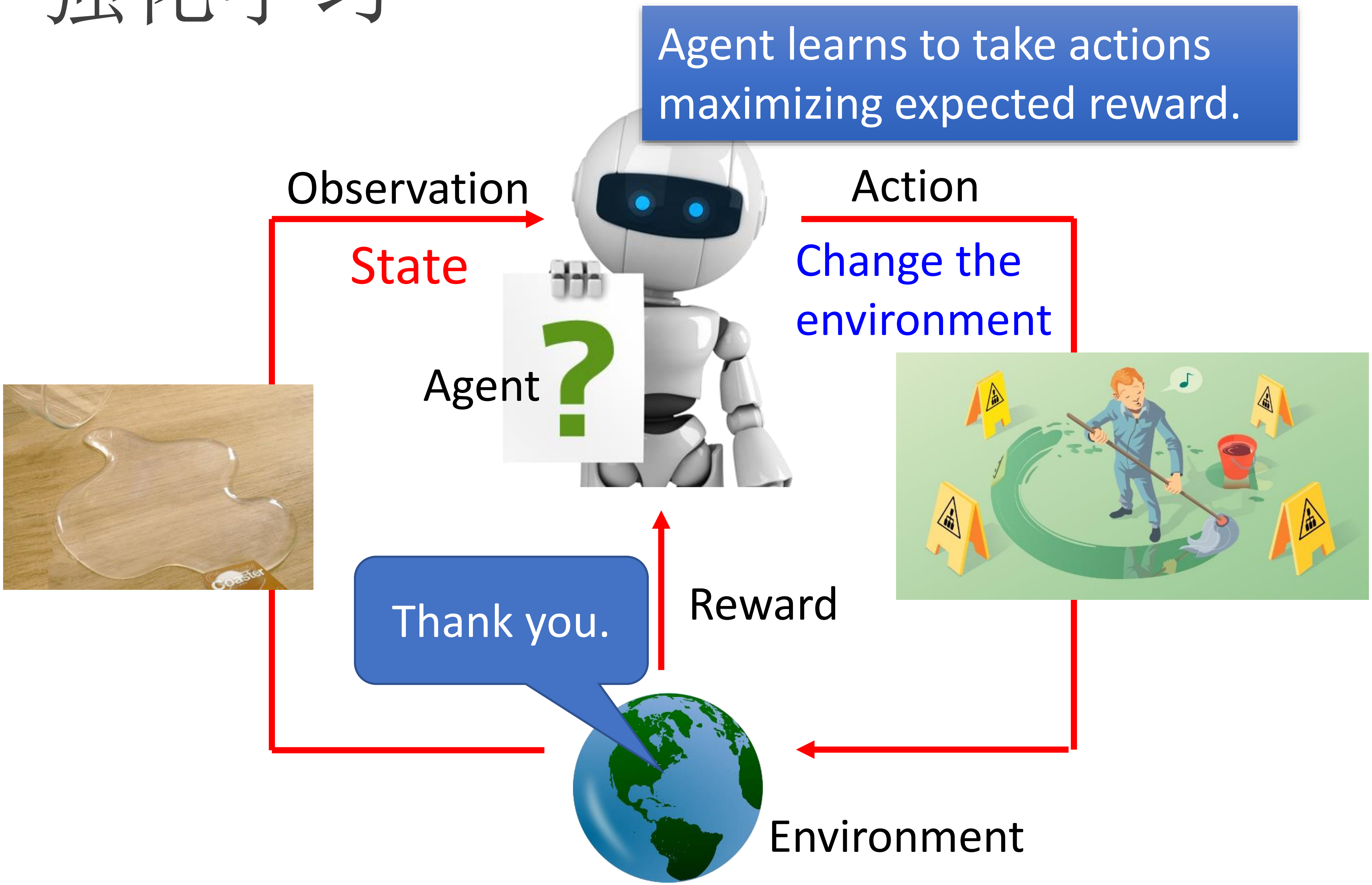
- 强化（reinforcement）学习是指从环境状态到行为映射的学习，以使系统行为从环境中获得的累积奖励值最大。在强化学习中，我们设计算法来把外界环境转化为最大化奖励量的方式的动作。

强化学习

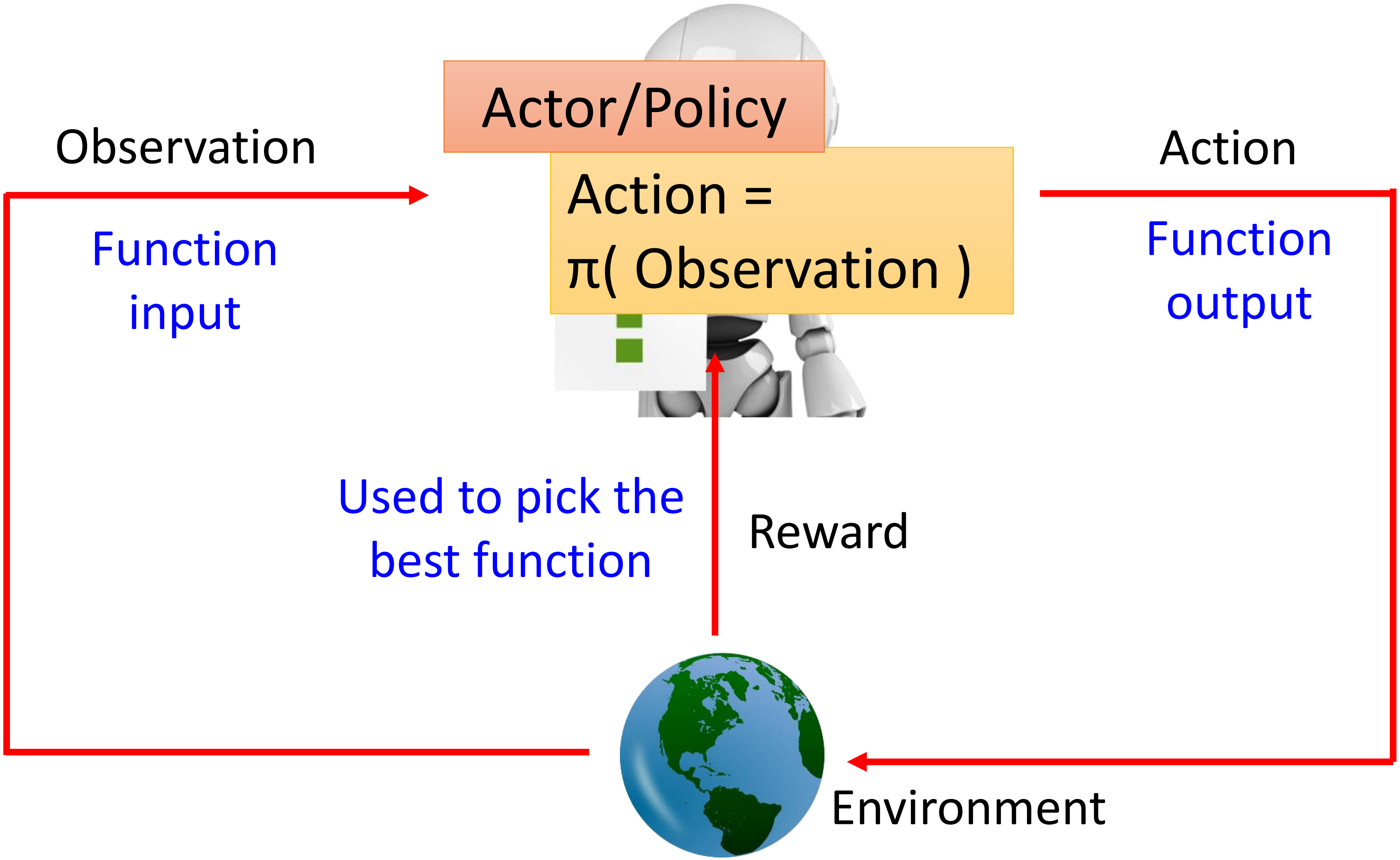


Observation
State
Agent
Action
Change the environment
Don’t do that
Reward
Environment

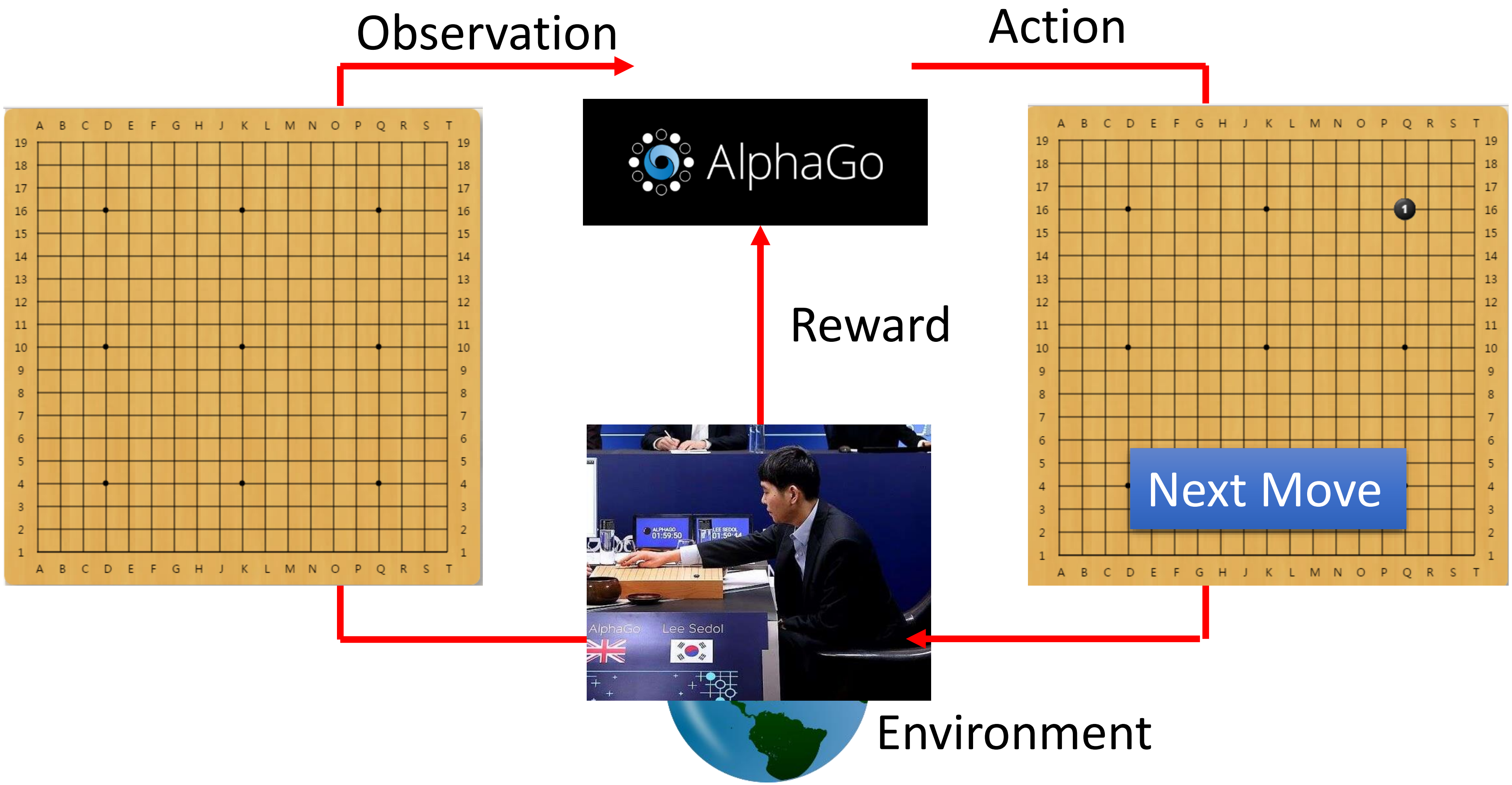
强化学习


Agent learns to take actions maximizing expected reward.
Observation
State
Agent
Action
Change the environment
Thank you.
Reward
Environment

Machine Learning
≈ Looking for a Function

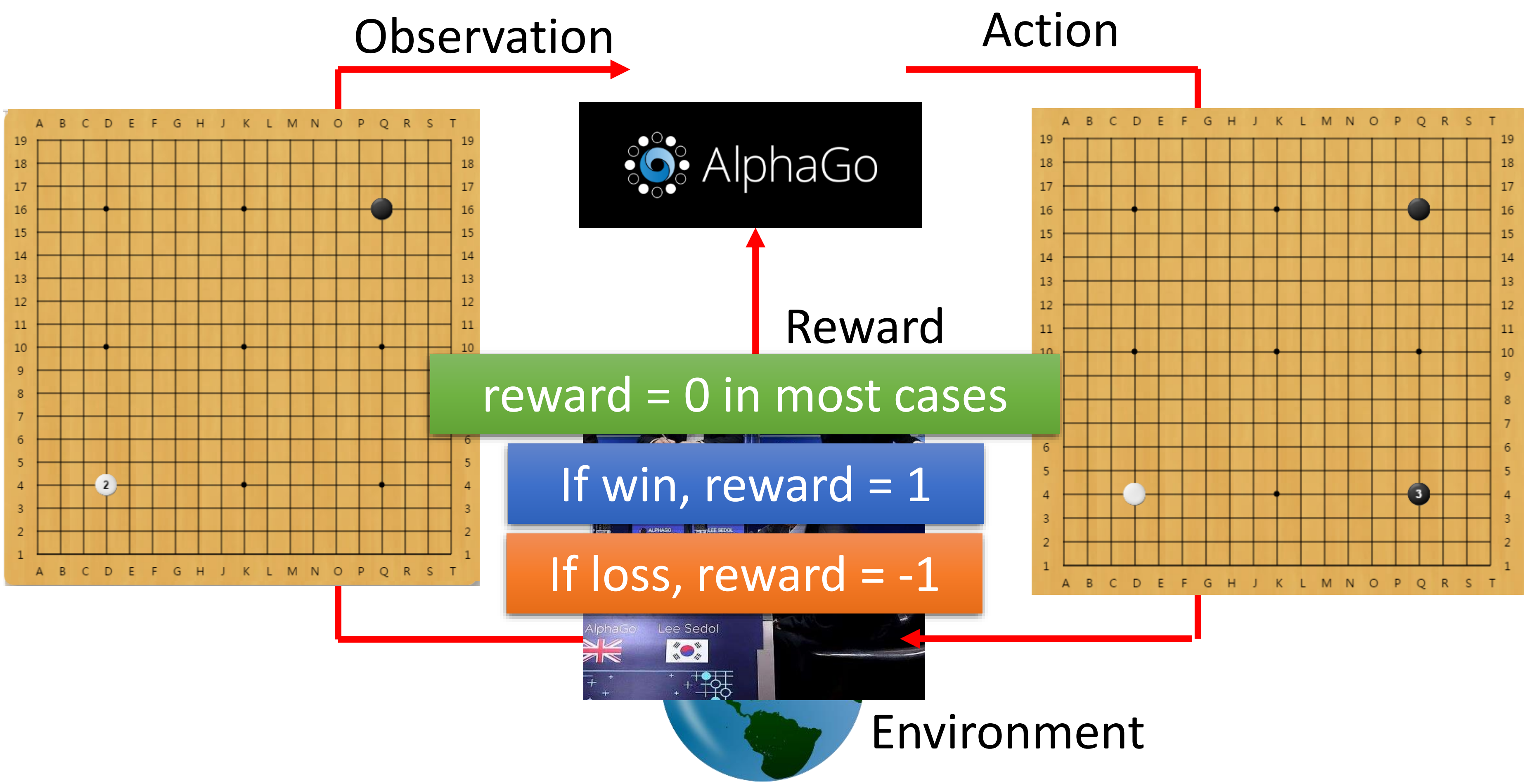


Learning to play Go



Learning to play Go

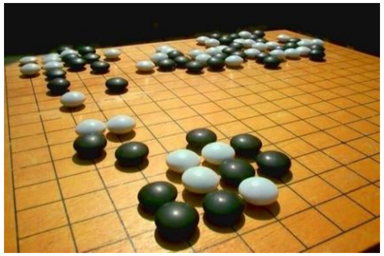
Agent learns to take actions maximizing expected reward.



Learning to play 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.)

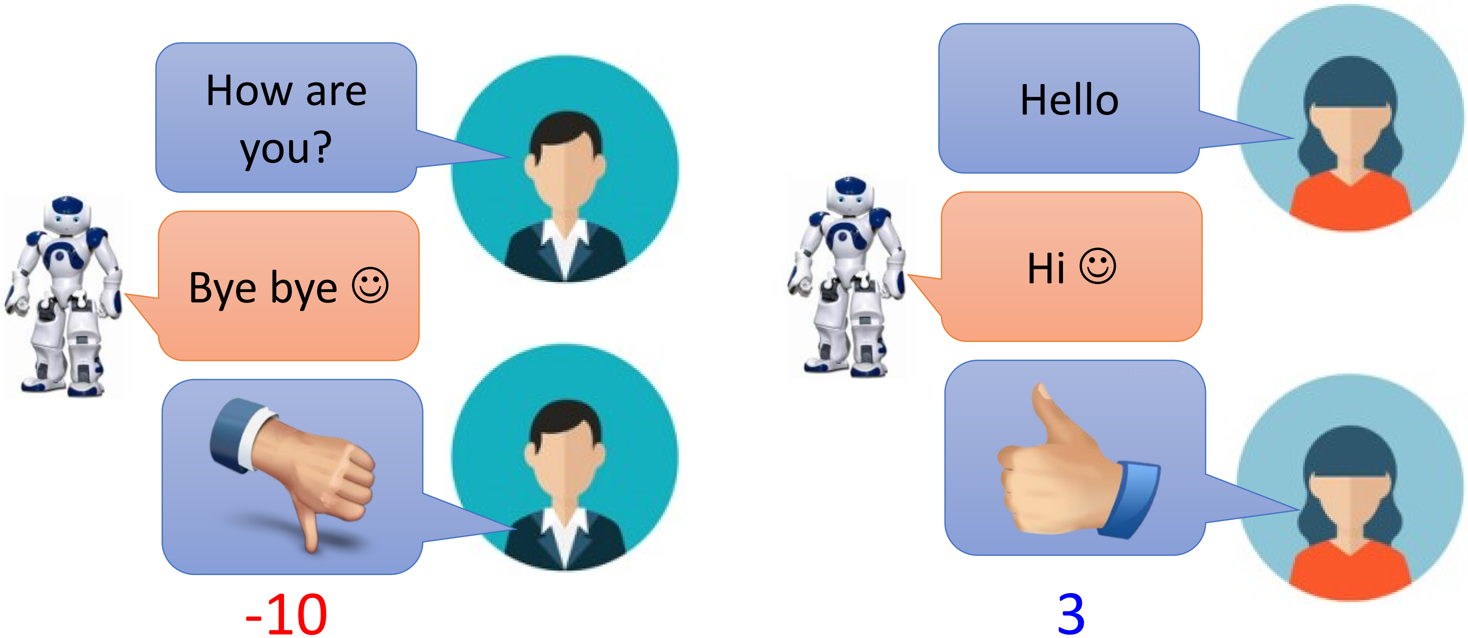
Alpha Go is supervised learning + reinforcement learning.

Learning a chat-bot

https://image.freepik.com/free-vector/variety-of-human-avatars_23-2147506285.jpg

http://www.freepik.com/free-vector/variety-of-human-avatars_766615.htm

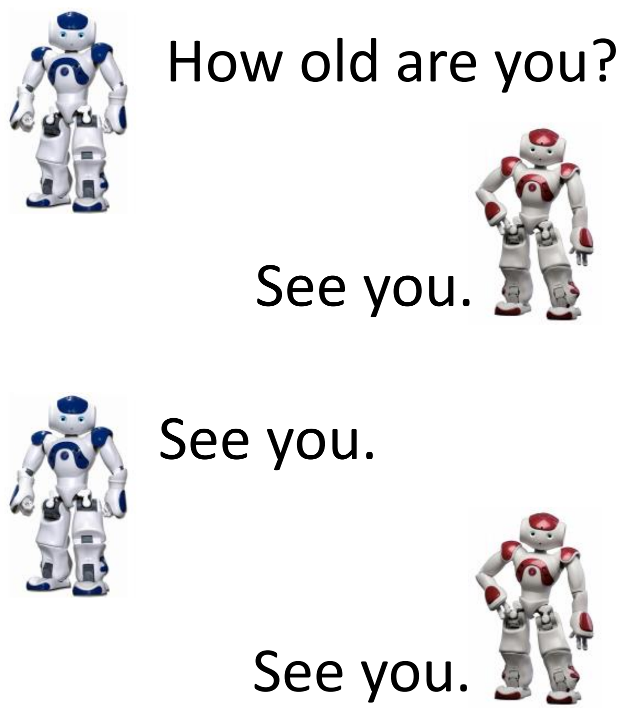
- Machine obtains feedback from user

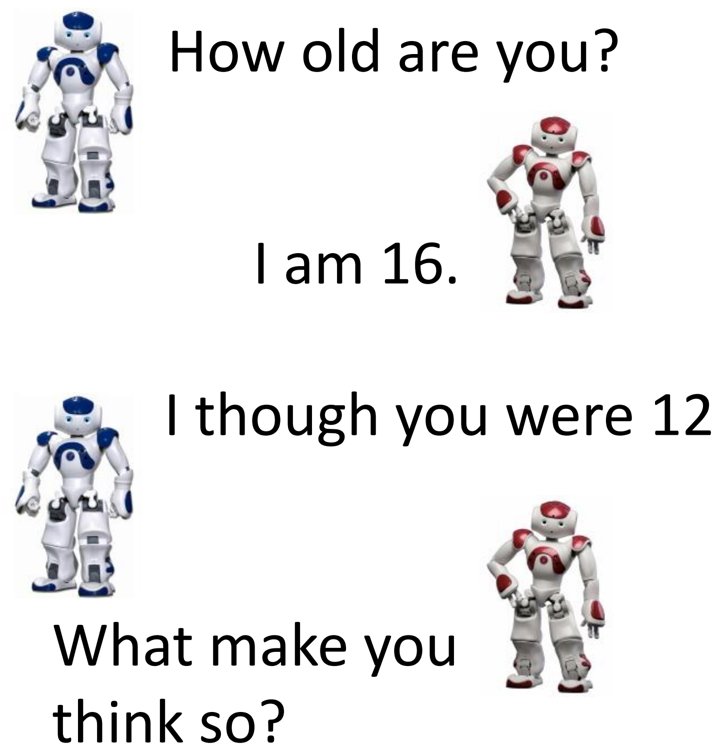


- Chat-bot learns to maximize the ***expected reward***

Learning a chat-bot

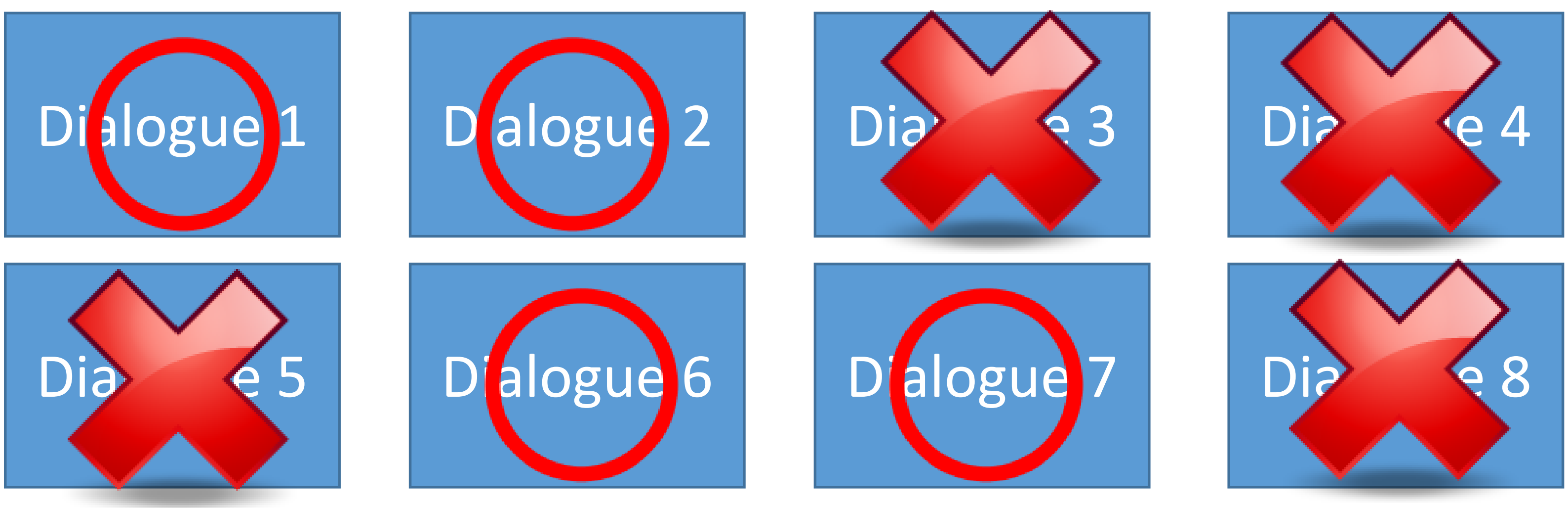
- Let two agents talk to each other (sometimes generate good dialogue, sometimes bad)





Learning a chat-bot

- 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

Learning a chat-bot

- Supervised

"Hello" — Say "Hi"

"Bye bye" — Say "Good bye"

- Reinforcement

....

....

......

Hello ☺

Agent

......

Agent

Bad

More applications

- Flying Helicopter
  - https://www.youtube.com/watch?v=0JL04JJjocc
- Driving
  - https://www.youtube.com/watch?v=0xo1Ldx3L5Q
- Robot
  - https://www.youtube.com/watch?v=370cT-OAzzM
- 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
  - https://www.youtube.com/watch?v=pbQ4qe8EwLo

Example: Playing Video Game

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

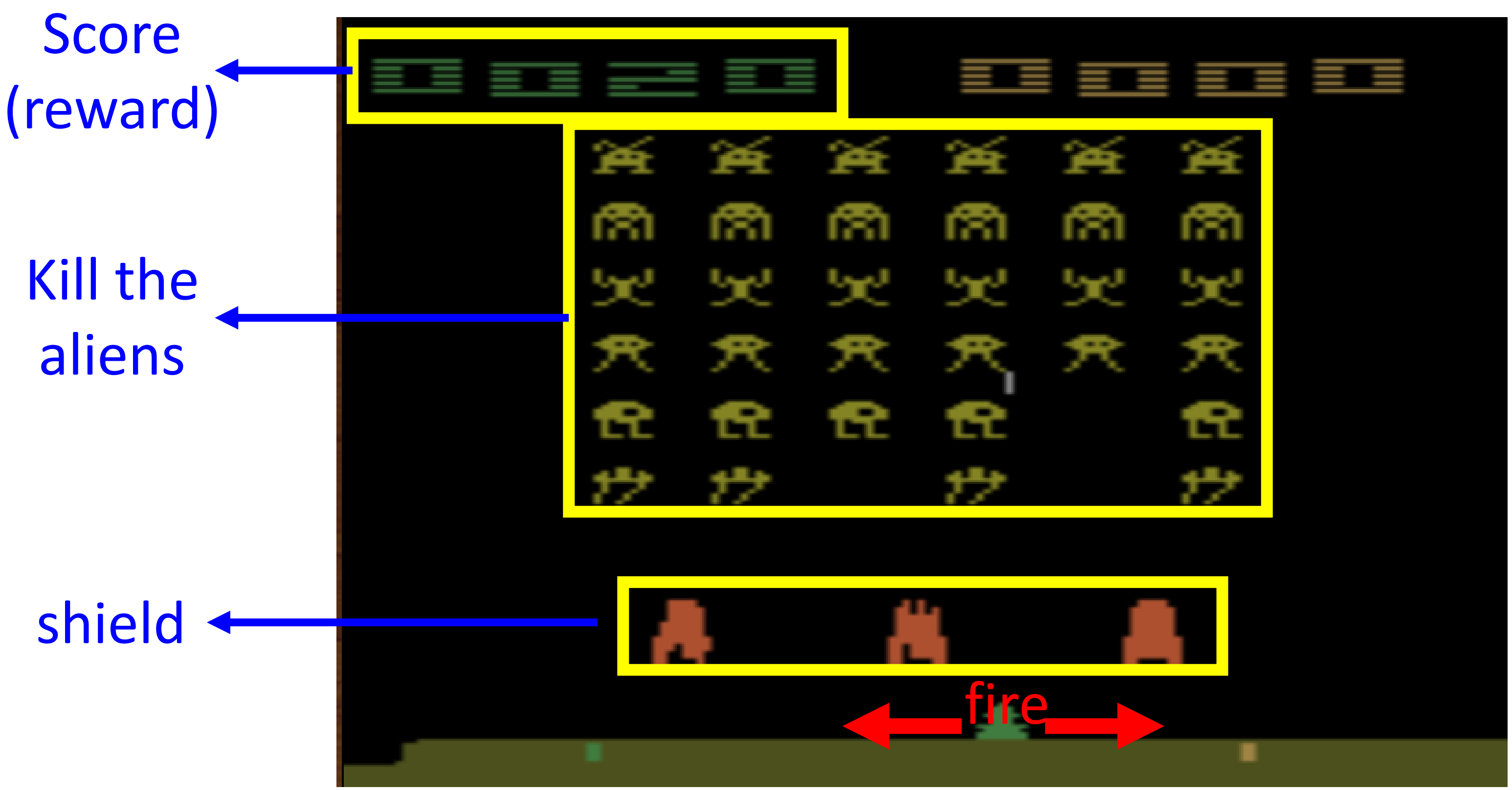
Machine learns to play video games as human players

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

Example: Playing Video Game

- Space invader

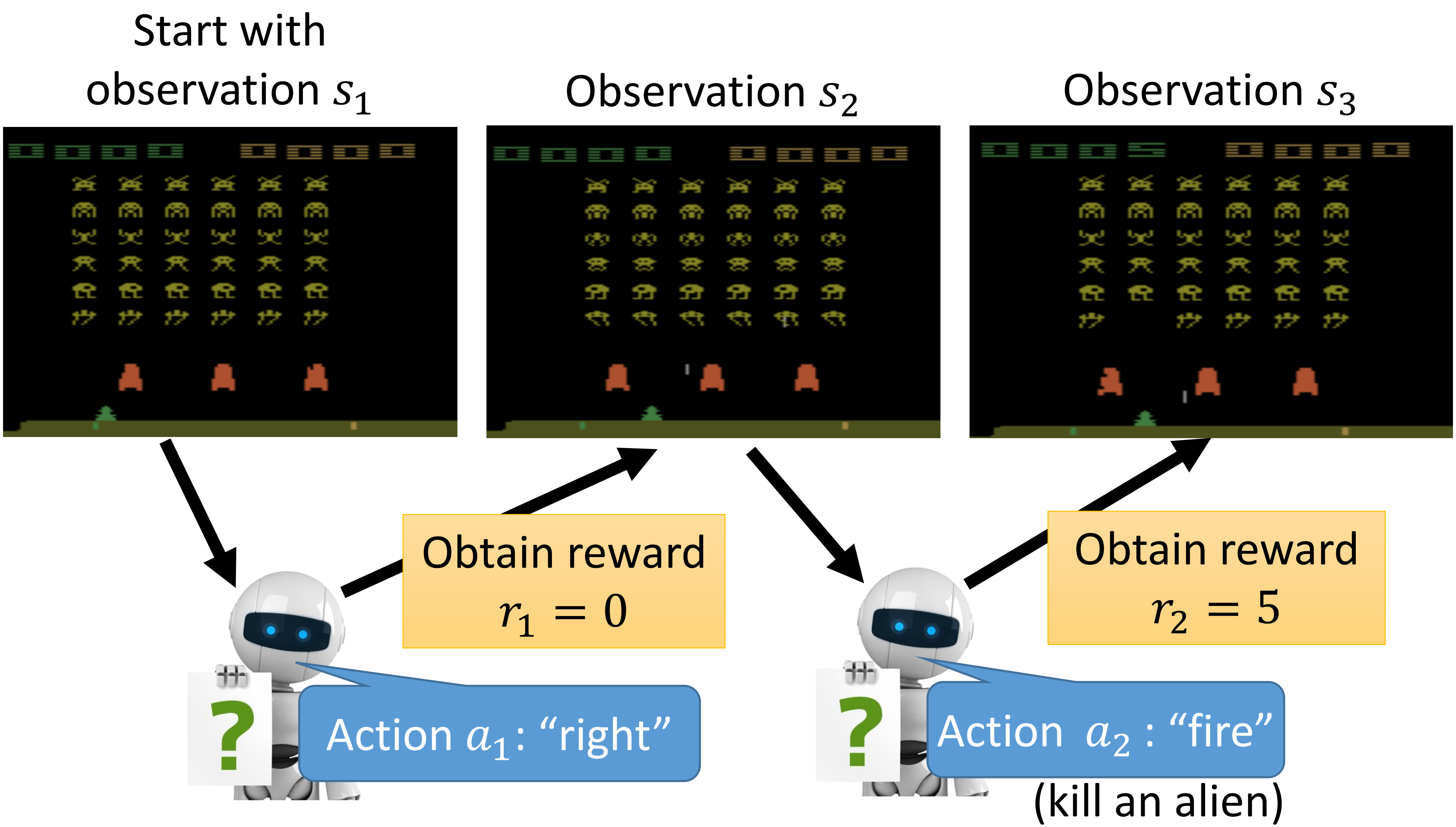
Termination: all the aliens are killed, or your spaceship is destroyed.



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

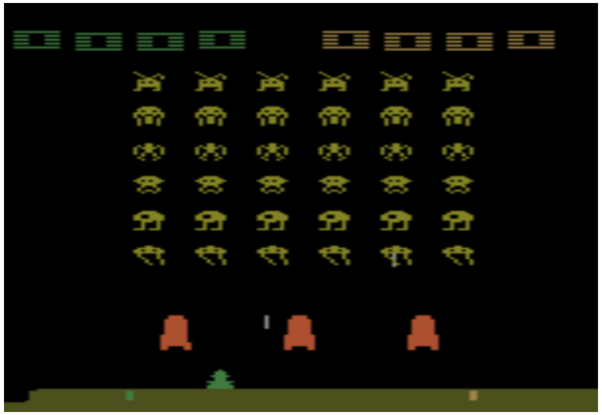


Usually there is some randomness in the environment

Example: Playing Video Game

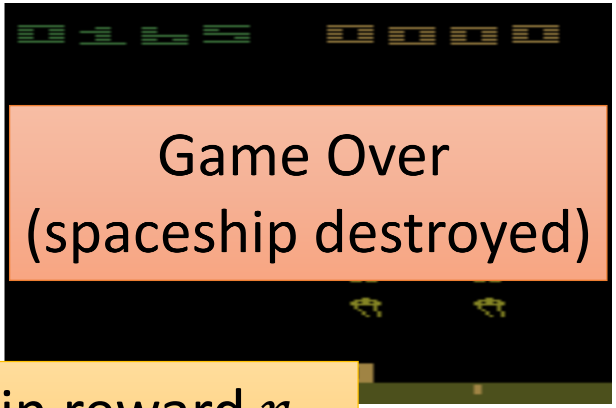
Start with observation s_1

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

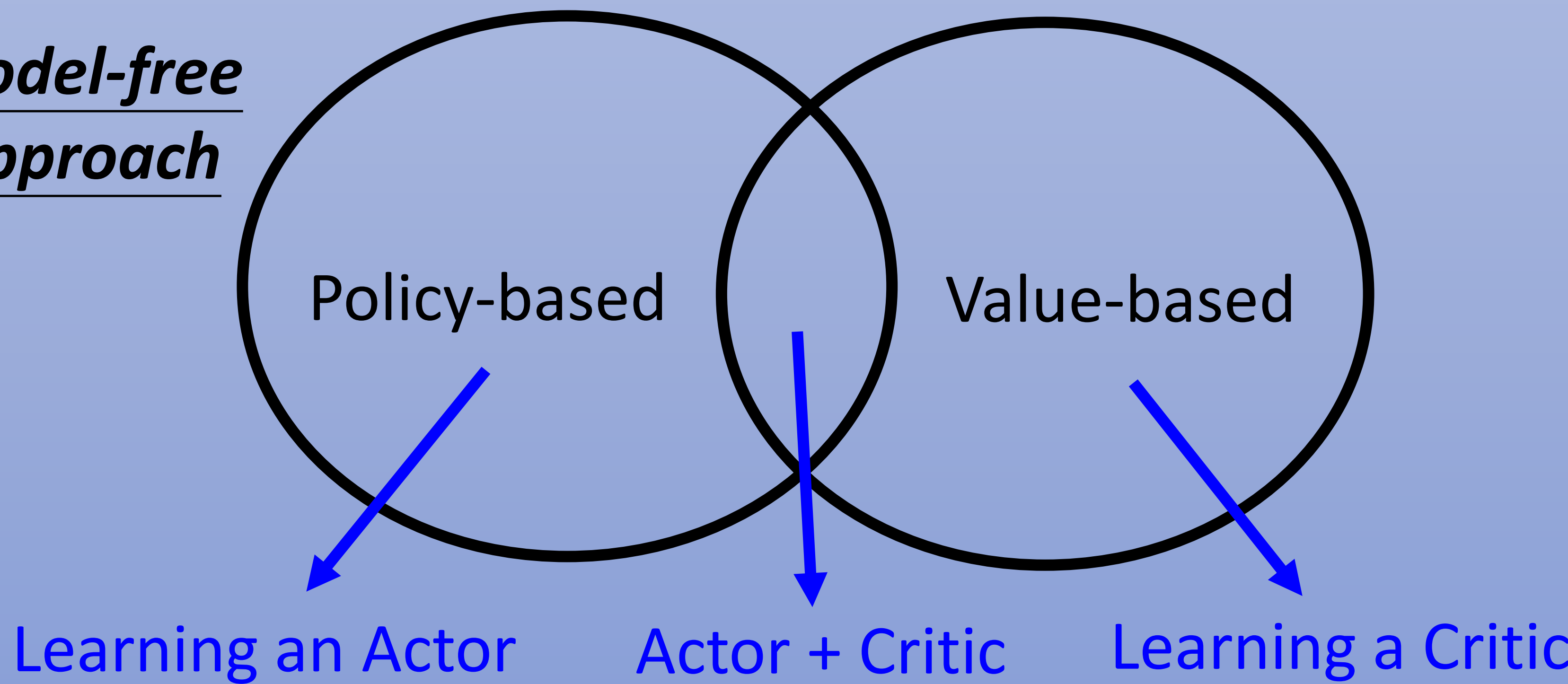
Properties of Reinforcement Learning

- **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

Model-free Approach



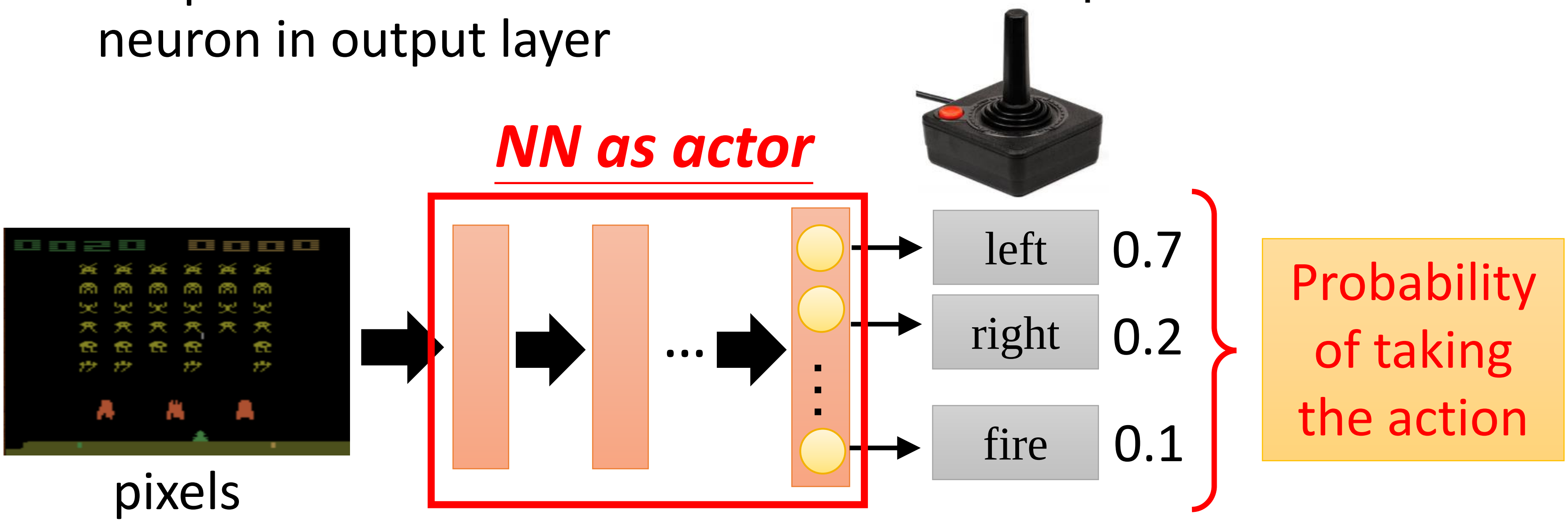
Model-based Approach

Policy-based Approach

Learning an Actor

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?

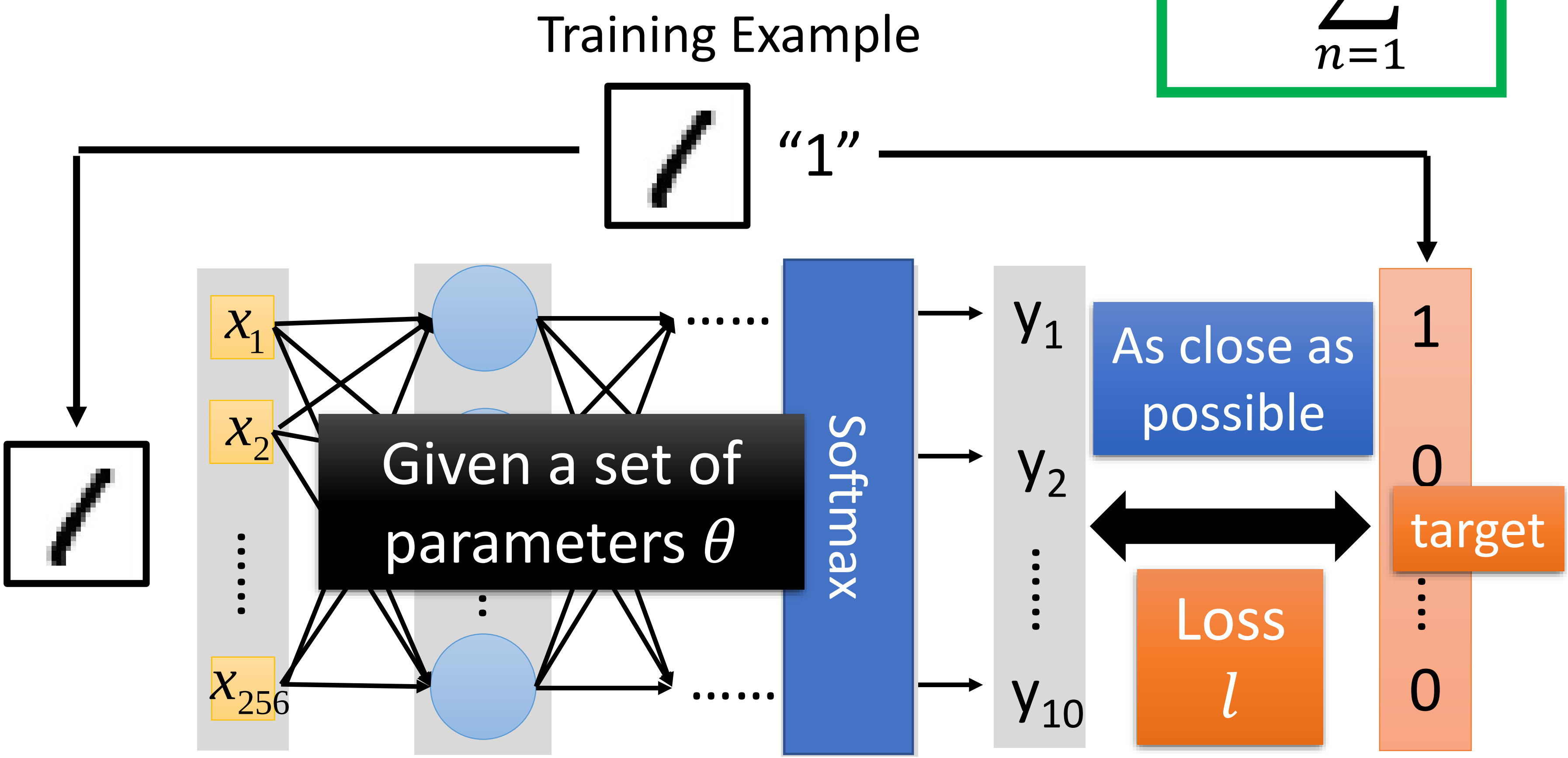
generalization

Goodness of Actor

- Review: Supervised learning

Total Loss:

$$L = \sum_{n=1}^{N} l_n$$



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

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

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

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

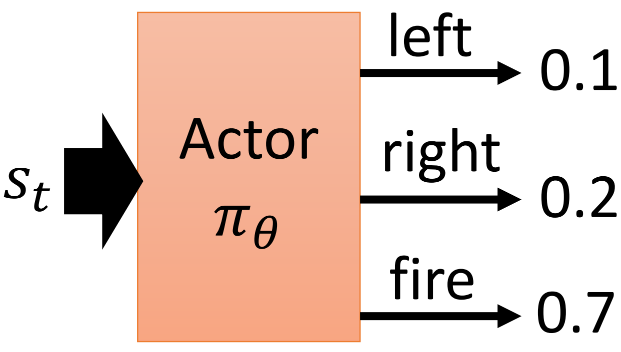
$$p(s_1)p(a_1|s_1,\theta)p(r_1,s_2|s_1,a_1)p(a_2|s_2,\theta)p(r_2,s_3|s_2,a_2)\cdots$$

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

not related to your actor

Control by your actor π_θ

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



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 $\{\tau^1, \tau^2, \cdots, \tau^N\}$

Sum over all possible trajectory

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

Gradient Ascent

- Problem statement

$$\theta^* = arg \max_{\theta} \bar{R}_\theta$$

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

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

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

Policy Gradient

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

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

$R(\tau)$ 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 $\{\tau^1, \tau^2, \cdots, \tau^N\}$

Policy Gradient

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

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

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

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

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

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

Ignore the terms not related to θ

Policy Gradient

$$\nabla logP(\tau|\theta) = \sum_{t=1}^{T} \nabla logp(a_t|s_t,\theta)$$

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

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

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

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

Policy Gradient

Given actor parameter 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) $R(\tau^2)$

$\vdots$ $\vdots$

Update Model

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

$$\nabla \bar{R}_\theta =$$

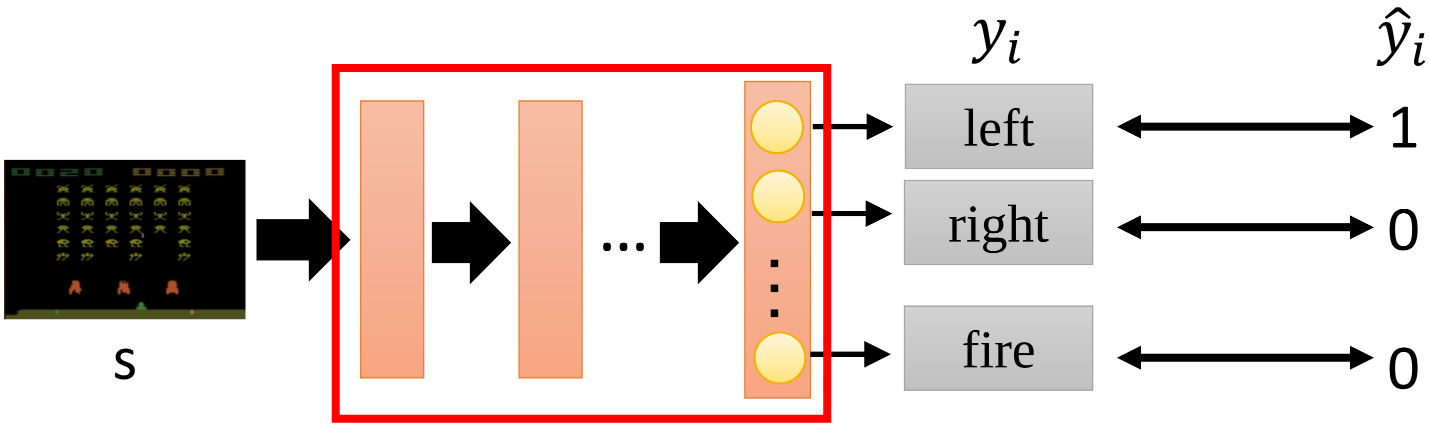
$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla \log p(a_t^n | s_t^n, \theta)$$

Data Collection

Policy Gradient

Considered as Classification Problem

Minimize: $-\sum_{i=1}^{3} \hat{y}_i log y_i$



Maximize: $log y_i =$

$log P("left"|s)$

$$\theta \leftarrow \theta + \eta \nabla log P("left"|s)$$

Policy Gradient

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

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

Given actor parameter θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

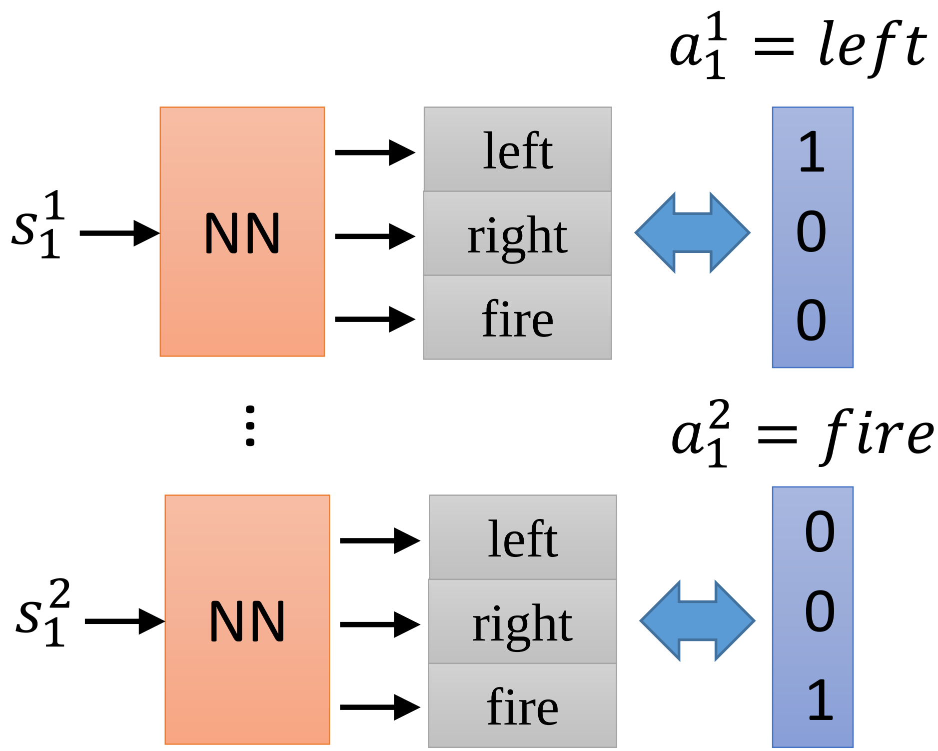
(s_2^1, a_2^1) $R(\tau^1)$

⋮ ⋮

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

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

⋮ ⋮



Policy Gradient

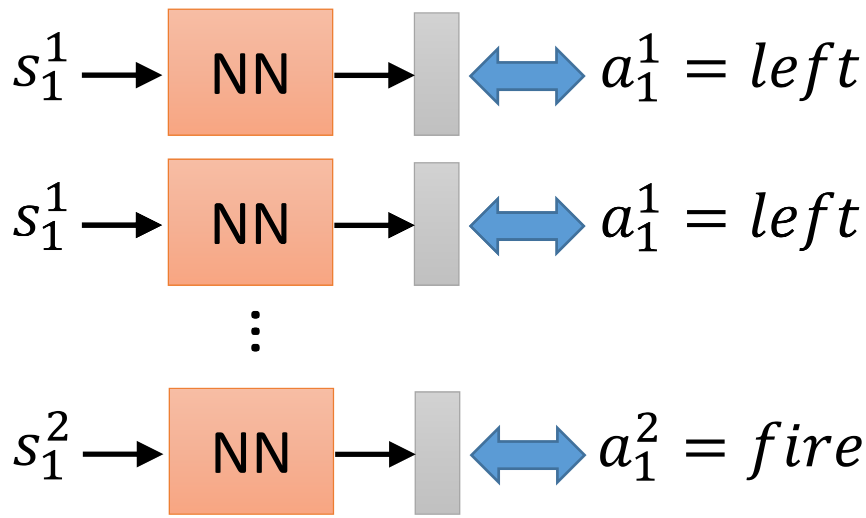
Given actor parameter θ

τ^1:	(s_1^1, a_1^1)	$R(\tau^1)$ 2
	(s_2^1, a_2^1)	$R(\tau^1)$ 2
	⋮	⋮
τ^2:	(s_1^2, a_1^2)	$R(\tau^2)$ 1
	(s_2^2, a_2^2)	$R(\tau^2)$ 1
	⋮	⋮

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

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

Each training data is weighted by $R(\tau^n)$

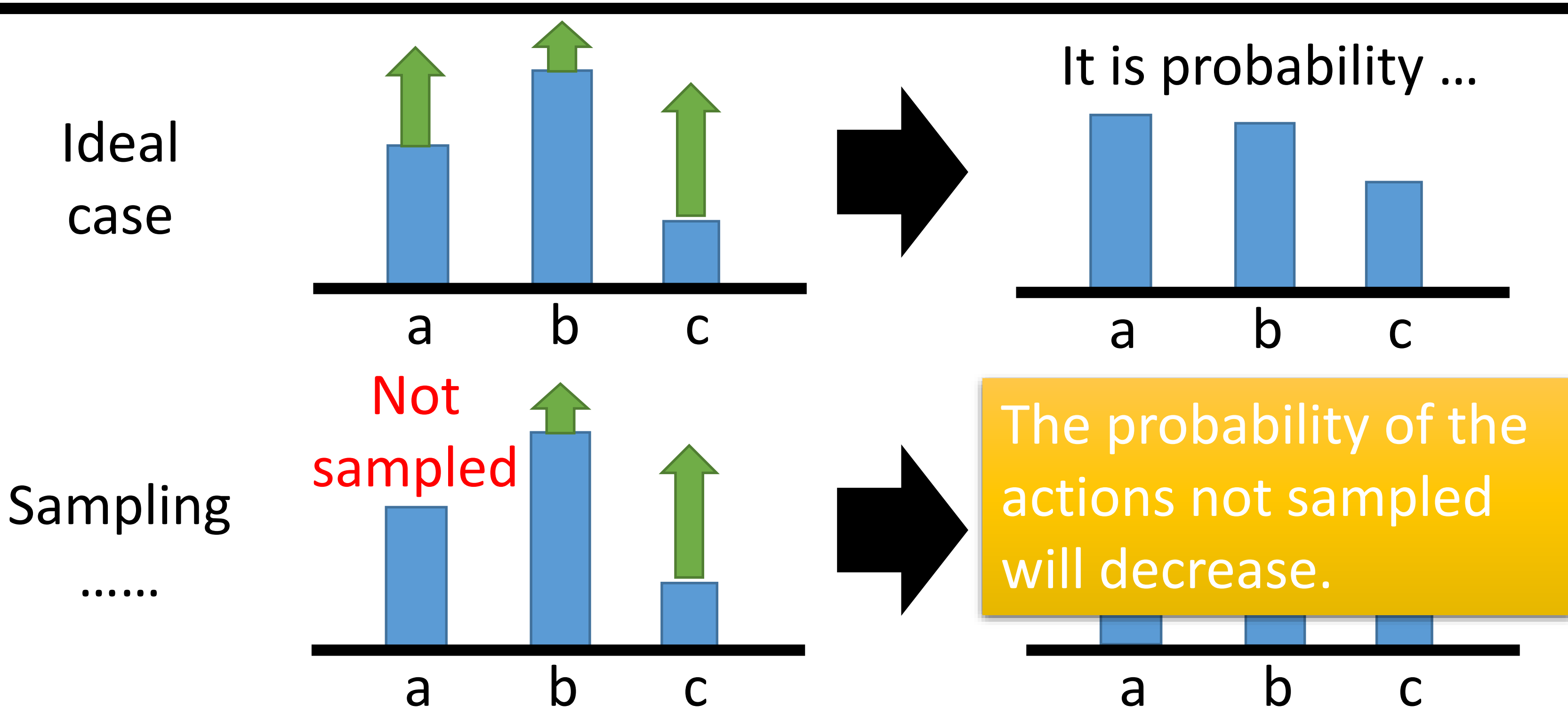


Add a Baseline

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

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

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



Value-based Approach

Learning a Critic

Critic

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

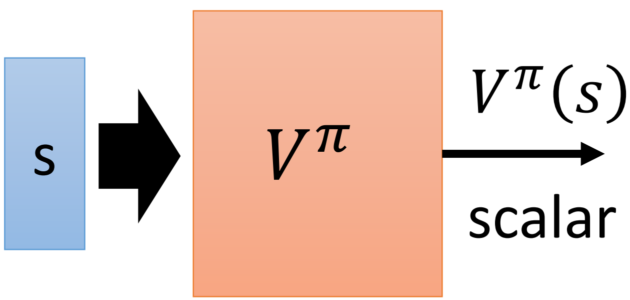
An actor can be found from a critic.

e.g. Q-learning

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

Critic

- 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

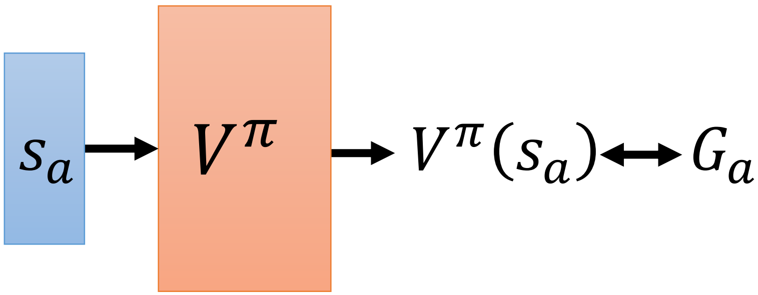
$V^{\pi}(s)$ is smaller

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

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

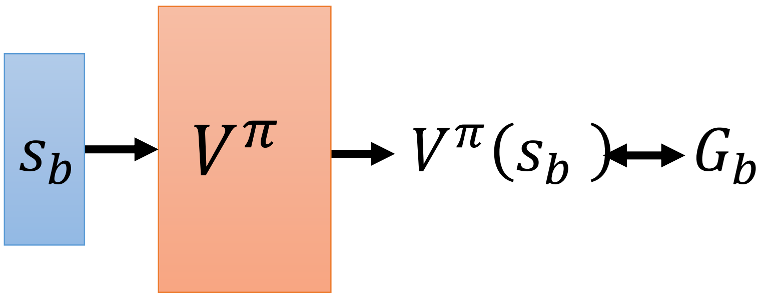
After seeing s_a,

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

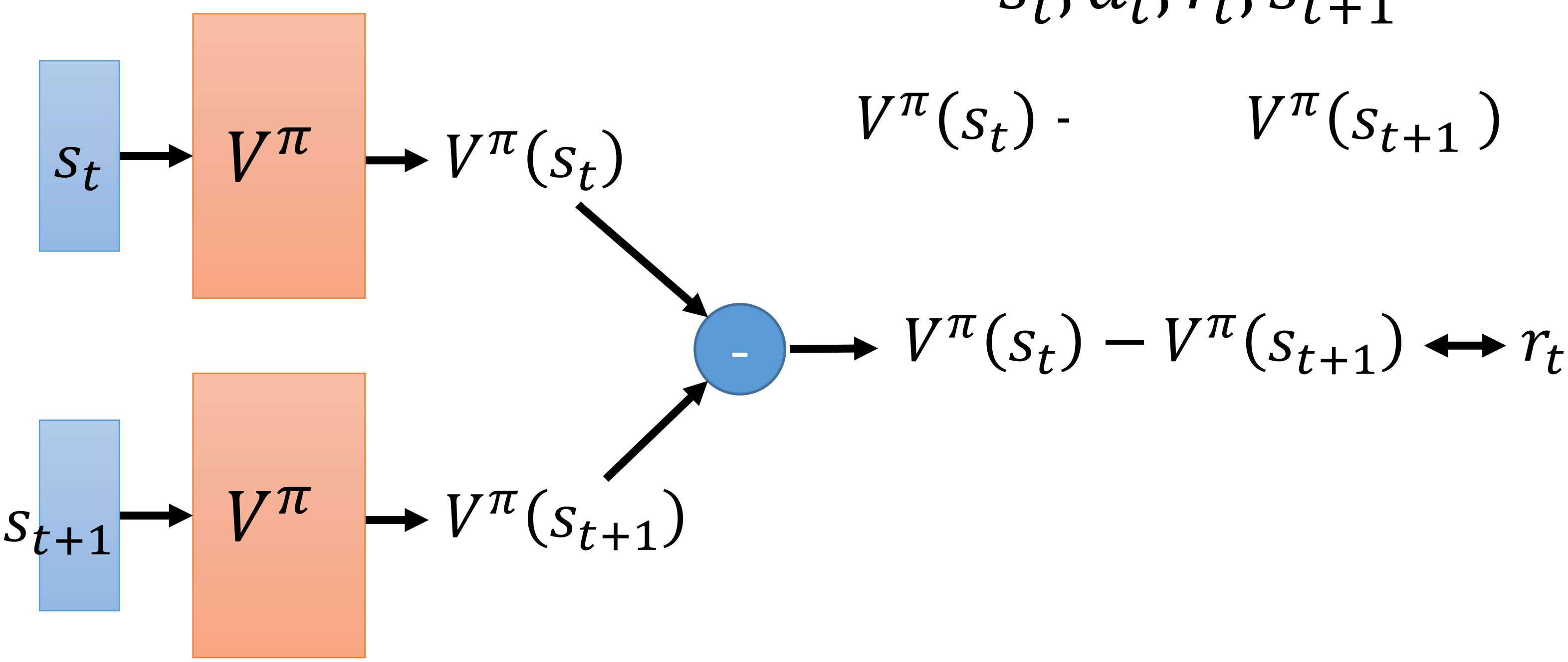


How to estimate $V^{\pi}(s)$

- Temporal-difference approach

$$\cdots s_t, a_t, r_t, s_{t+1} \cdots$$

$$V^{\pi}(s_t) = V^{\pi}(s_{t+1}) + r_t$$



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

MC v.s. TD

$s_a \rightarrow V^\pi \rightarrow V^\pi(s_a) \leftrightarrow G_a$

Larger variance

unbiased

$s_t \rightarrow V^\pi \rightarrow V^\pi(s_t) \longleftrightarrow r + V^\pi(s_{t+1}) \leftarrow V^\pi \leftarrow s_{t+1}$

Smaller variance

May be biased

MC v.s. TD

[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:

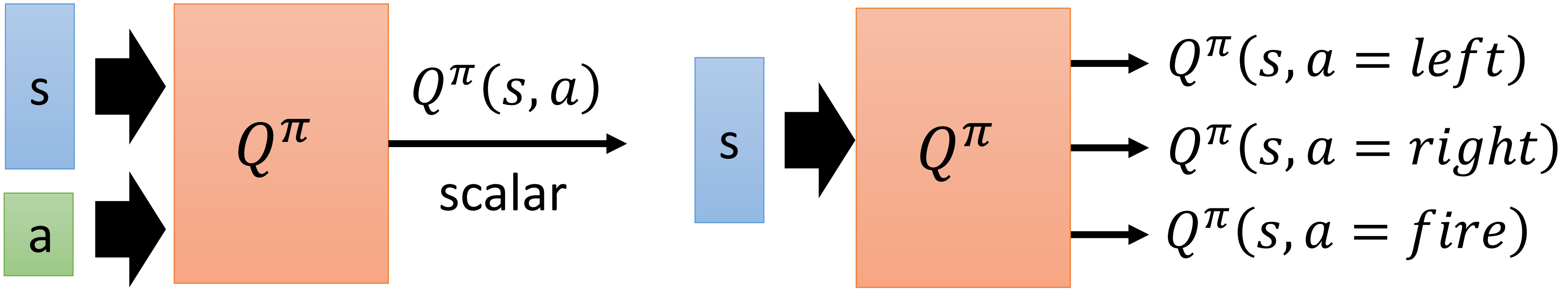
$$V^\pi(s_b) + r = V^\pi(s_a)$$

3/4   0   3/4

(The actions are ignored here.)

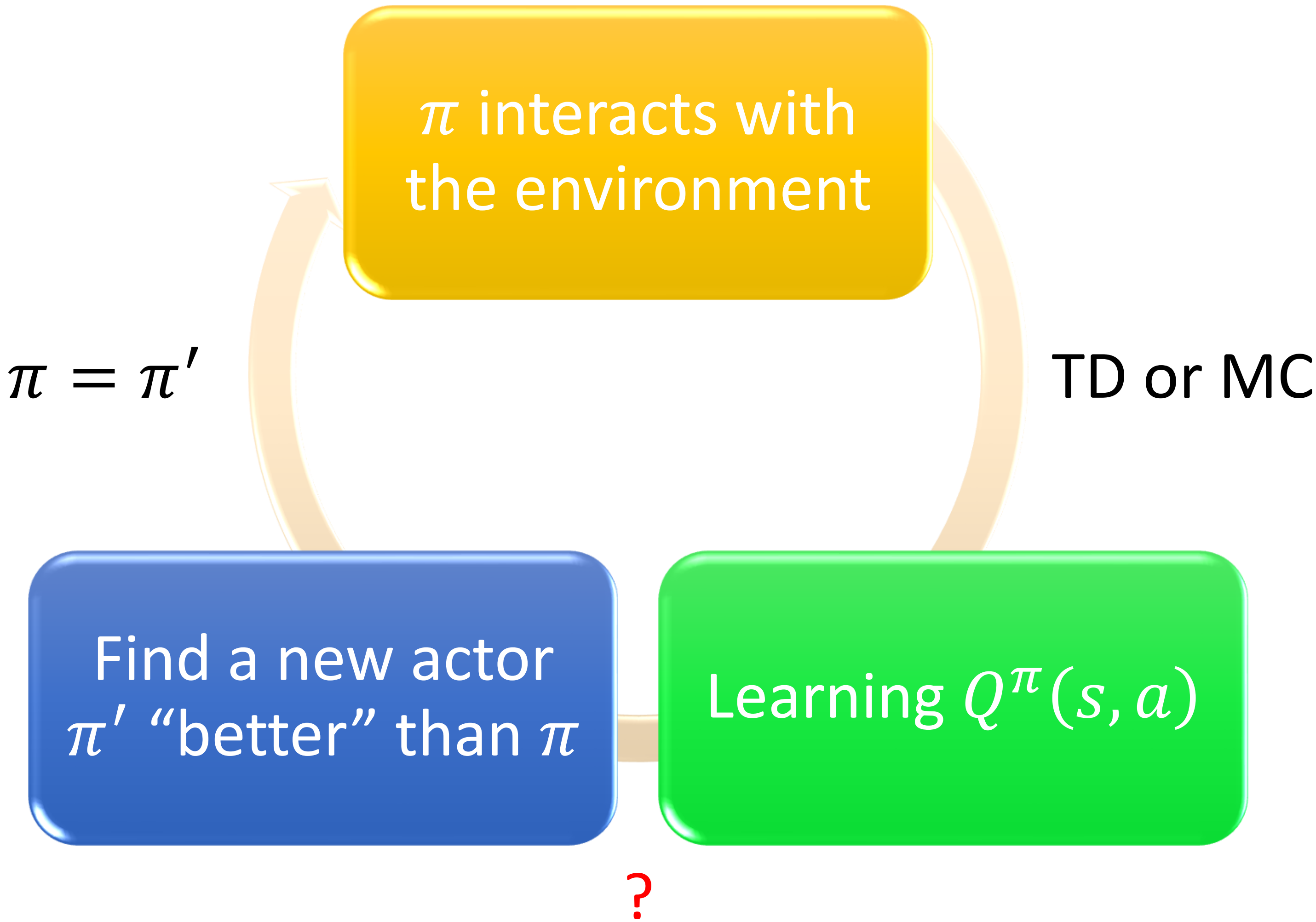
Another Critic

- 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



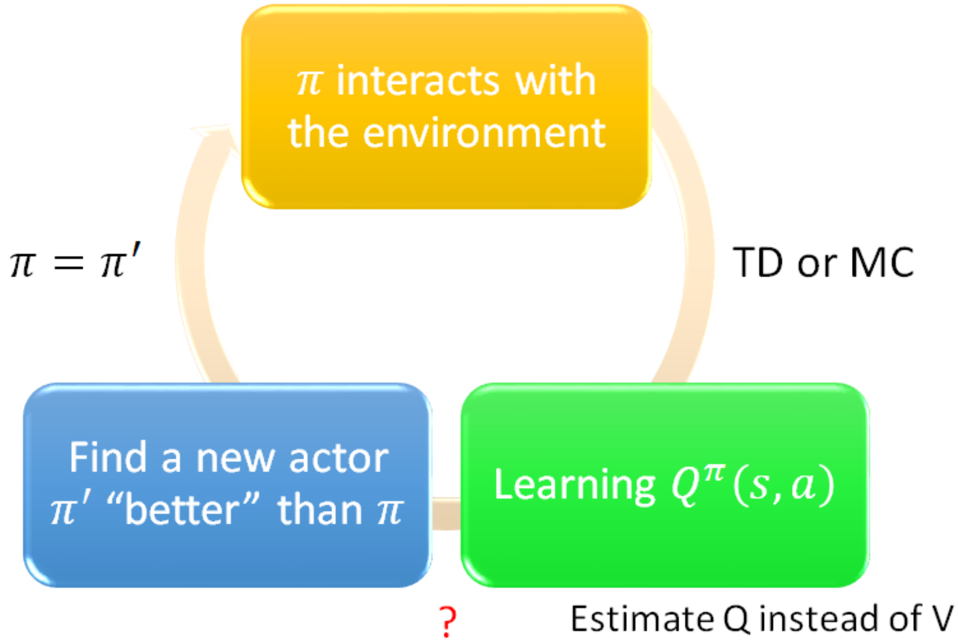
for discrete action only

Q-Learning



π interacts with the environment
TD or MC
Learning $Q^{\pi}(s,a)$
?
Find a new actor π' "better" than π
$\pi = \pi'$

Q-Learning



- Given $Q^{\pi}(s,a)$, find a new actor π' "better" than π
  - "Better": $V^{\pi'}(s) \geq V^{\pi}(s)$, for all state s

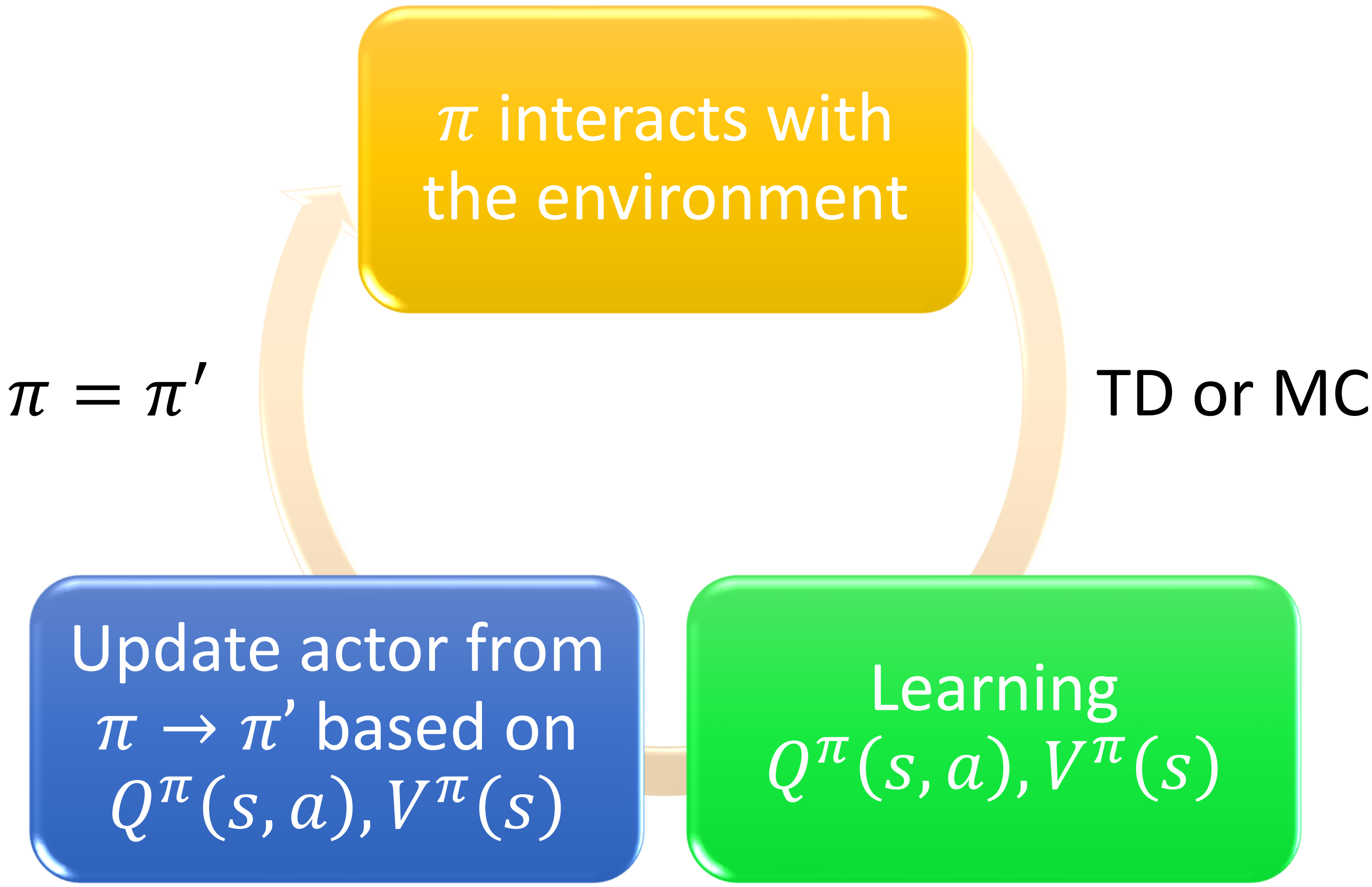
$$\pi'(s) = arg \max_{a} Q^{\pi}(s,a)$$

- ➢ π' does not have extra parameters. It depends on Q
- ➢ Not suitable for continuous action a

Deep Reinforcement Learning

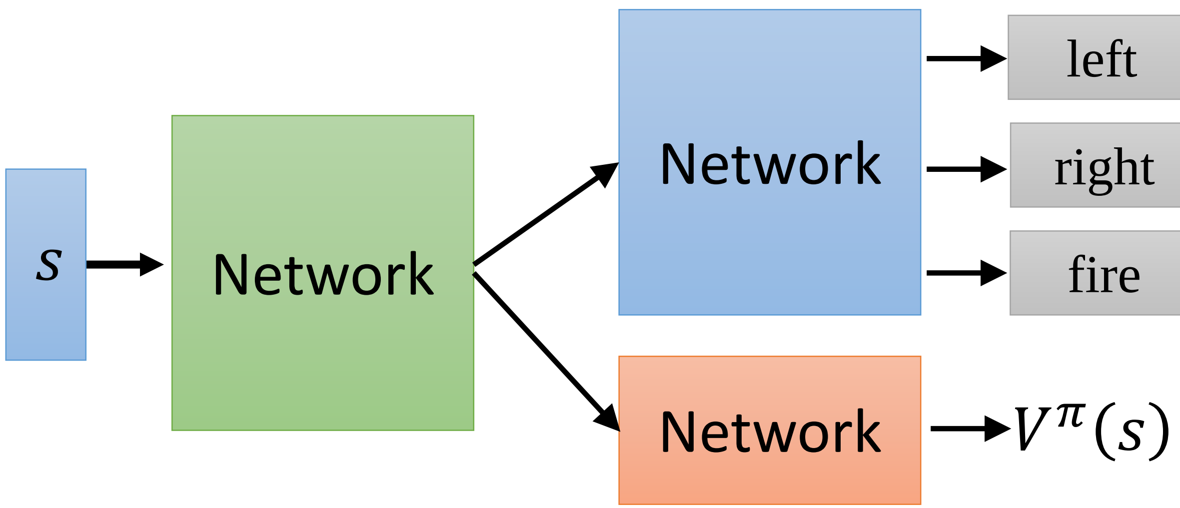
Actor-Critic

Actor-Critic



Actor-Critic

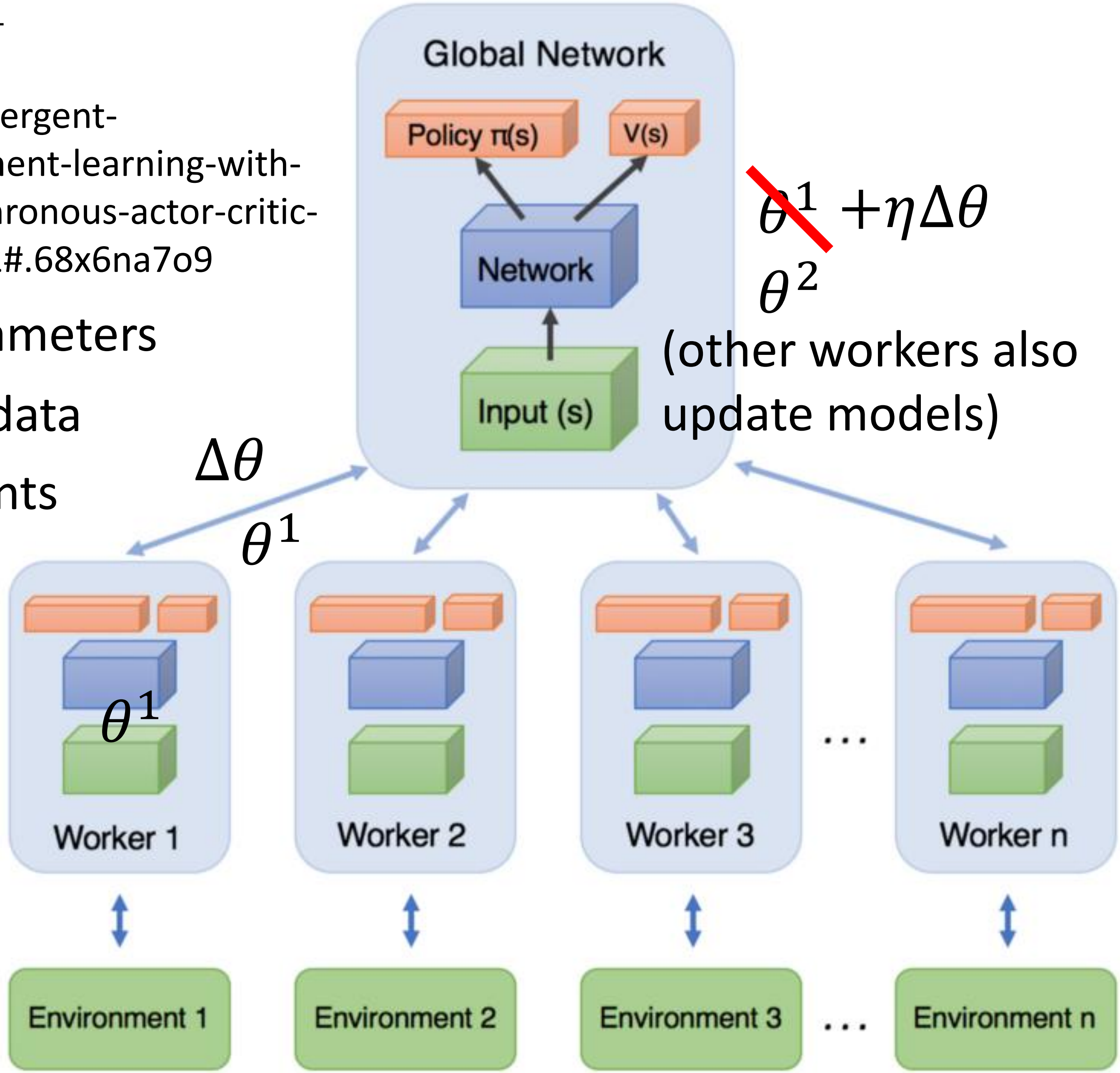
- Tips
  - The parameters of actor $\pi(s)$ and critic $V^{\pi}(s)$ can be shared



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)

$\Delta\theta$

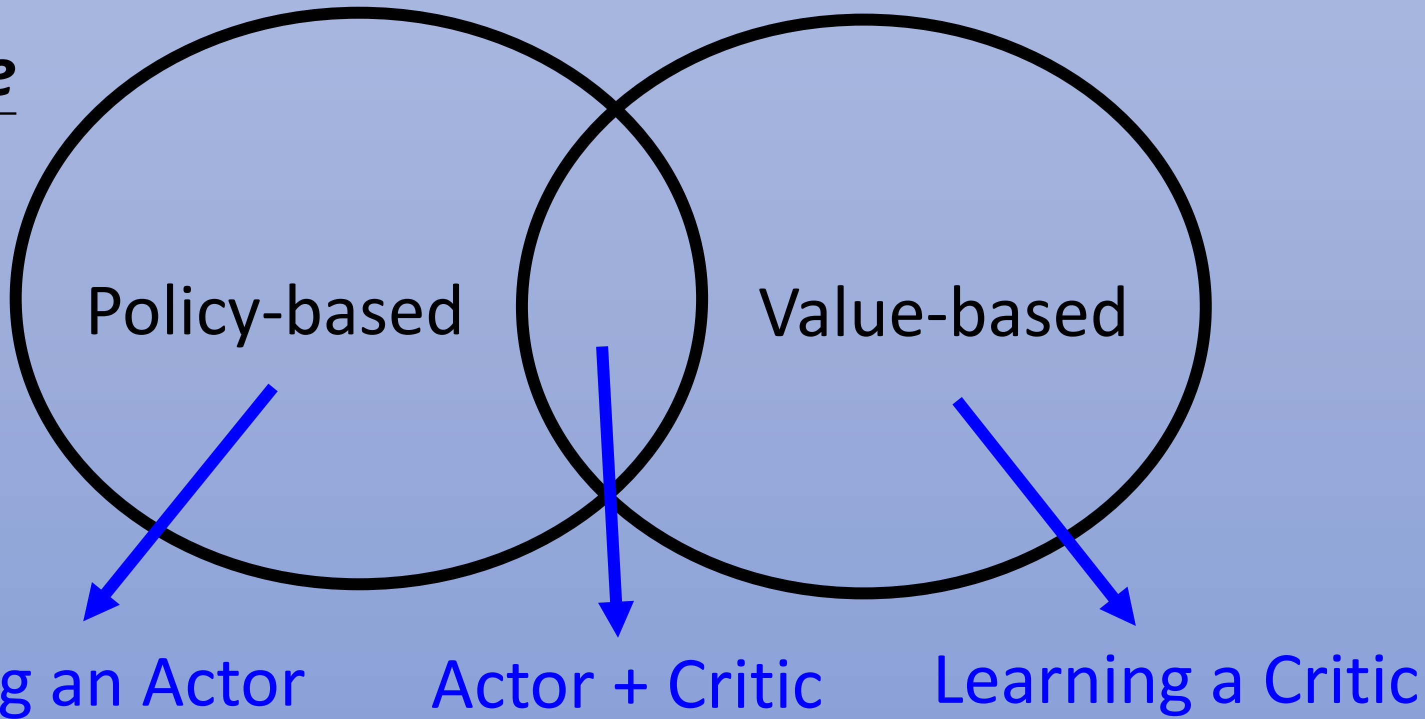
θ^1

θ^1

$\Delta\theta$

Concluding Remarks

Model-free Approach



Model-based Approach

Thanks!

Any questions?