Reinforcement Learning: An Overview

Outline

- Overview
- Policy Gradient
- On-Policy vs Off-Policy
- Q-Learning
- Key Issues and Application

Origin

- **The trial and error method (trial-and-error) originated from animal learning psychology.**
- **Optimal control - value function, dynamic programming, temporal difference.**
- **February 2015: Google's article on nature, using RL to play Atari games, can surpass the performance of human players.**
- **2016 spring: Alpha go based on RL sweeps human chess players.**
    - Reinforcement Learning: An Introduction, Richard S. Sutton and Andrew G. Barto c 2014, 2015, 2016

Deep Reinforcement Learning

Deep Reinforcement Learning: AI = RL + DL

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.

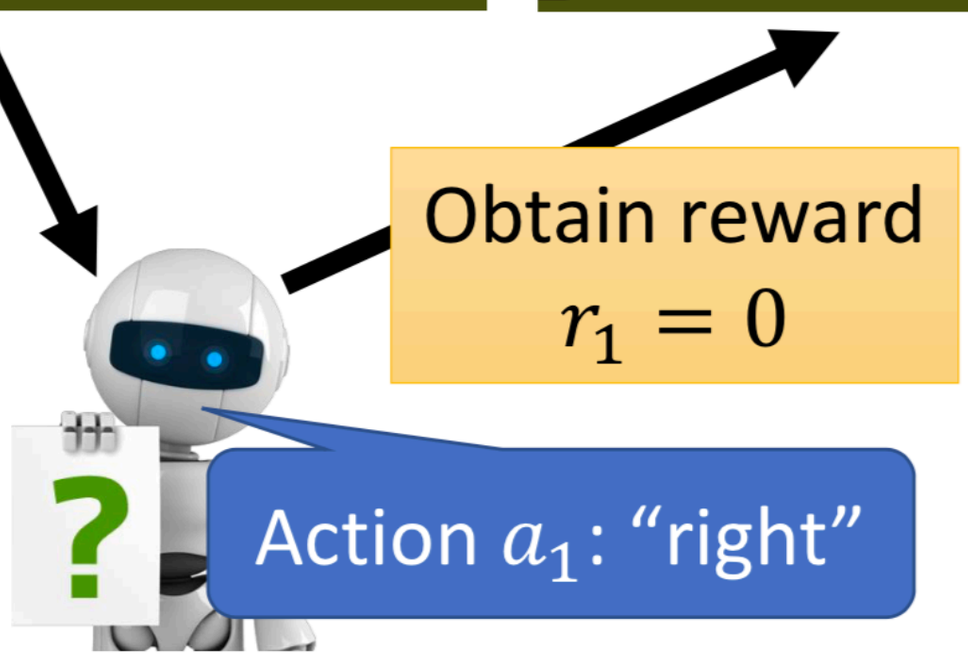
http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML17_2.html

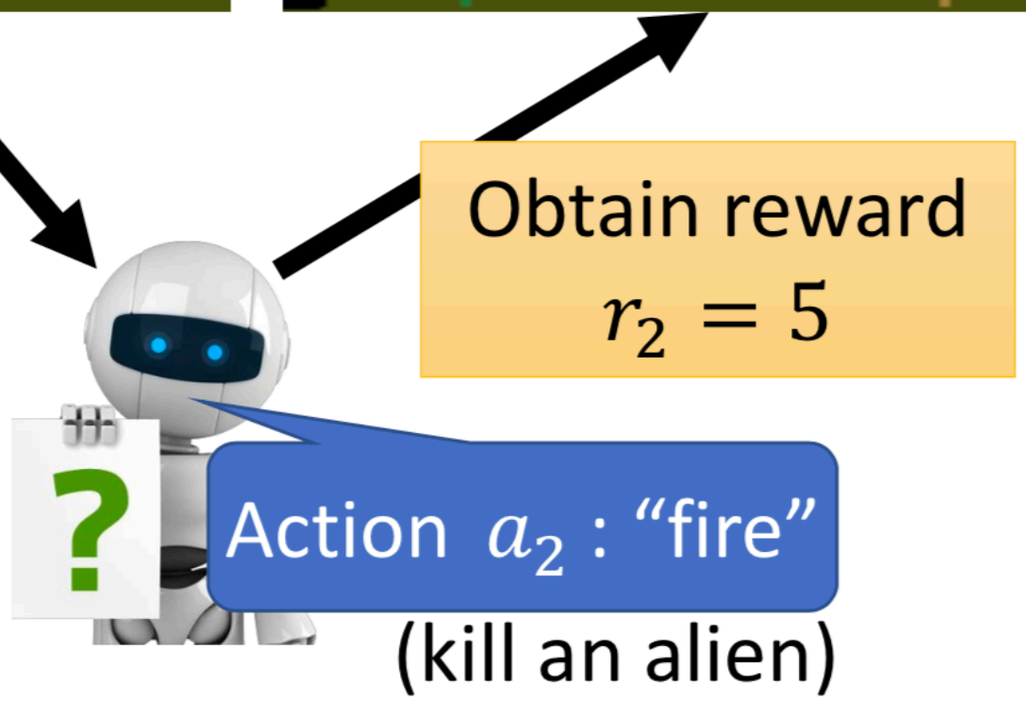
Example: Playing Video Game

Start with observation s_1

Observation s_2

Observation s_3

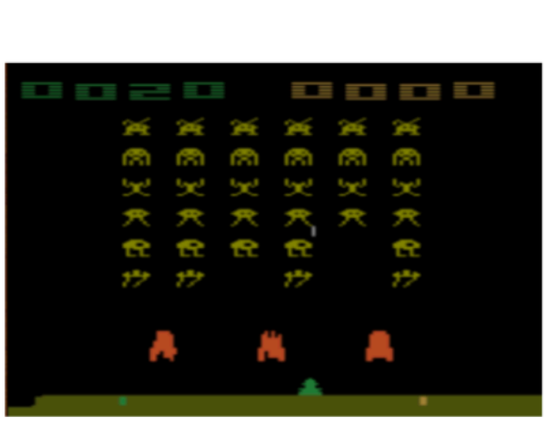




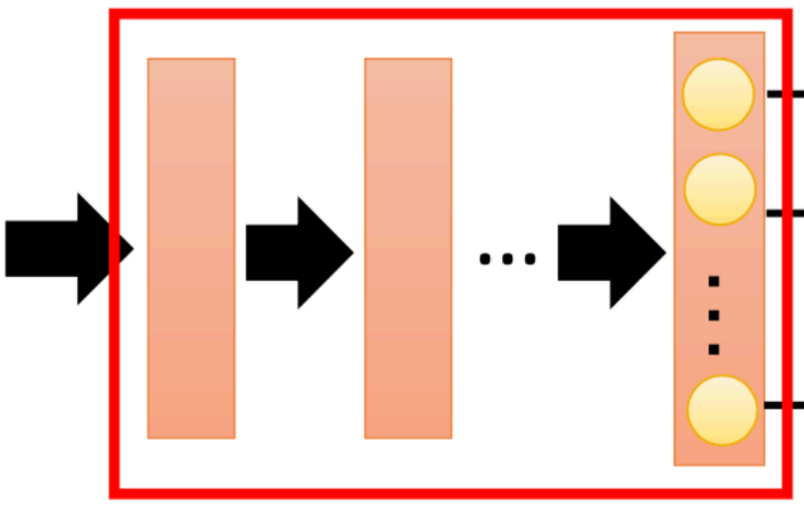
Policy Gradient

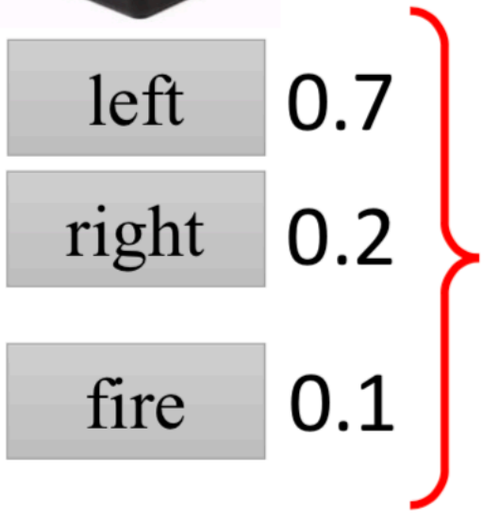
- Policy π is a network with parameter θ
  - Input: the observation of machine represented as a vector or a matrix
  - Output: each action corresponds to a neuron in output layer

Take the action based on the probability.



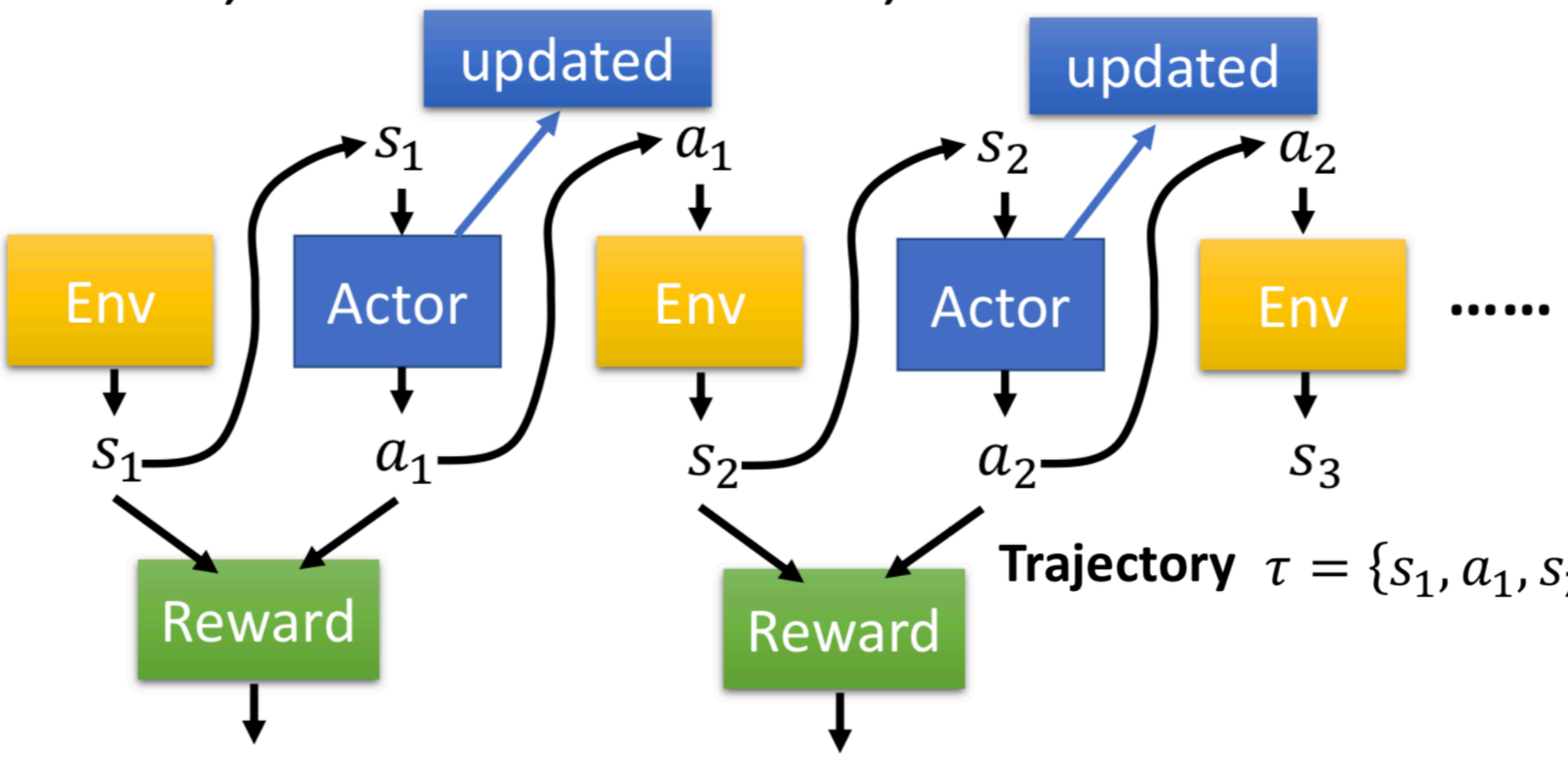
pixels



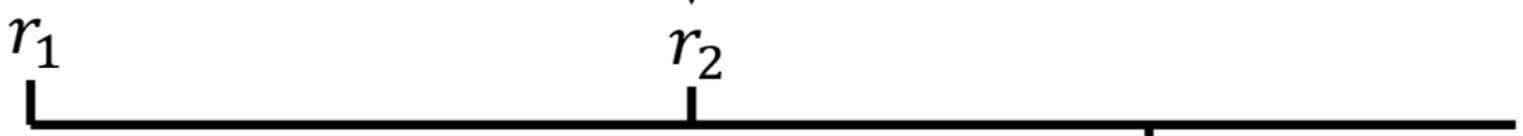


Score of an action

Actor, Environment, Reward



Trajectory $\tau = \{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\}$



Expected Reward

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau) = E_{\tau \sim p_\theta(\tau)}[R(\tau)] \qquad R(\tau) = \sum_{t=1}^{T} r_t$$

$$p_\theta(\tau)$$

$$= p(s_1) p_\theta(a_1|s_1) p(s_2|s_1, a_1) p_\theta(a_2|s_2) p(s_3|s_2, a_2) \cdots$$

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

Policy Gradient

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

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

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

It can even be a black box.

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

$$\nabla f(x) = f(x) \nabla \log f(x)$$

Trick

$$= \sum_\tau R(\tau) p_\theta(\tau) \nabla \log p_\theta(\tau)$$

$$= E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla \log p_\theta(\tau)] \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla \log p_\theta(\tau^n)$$

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

On-policy → Off-policy

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

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

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

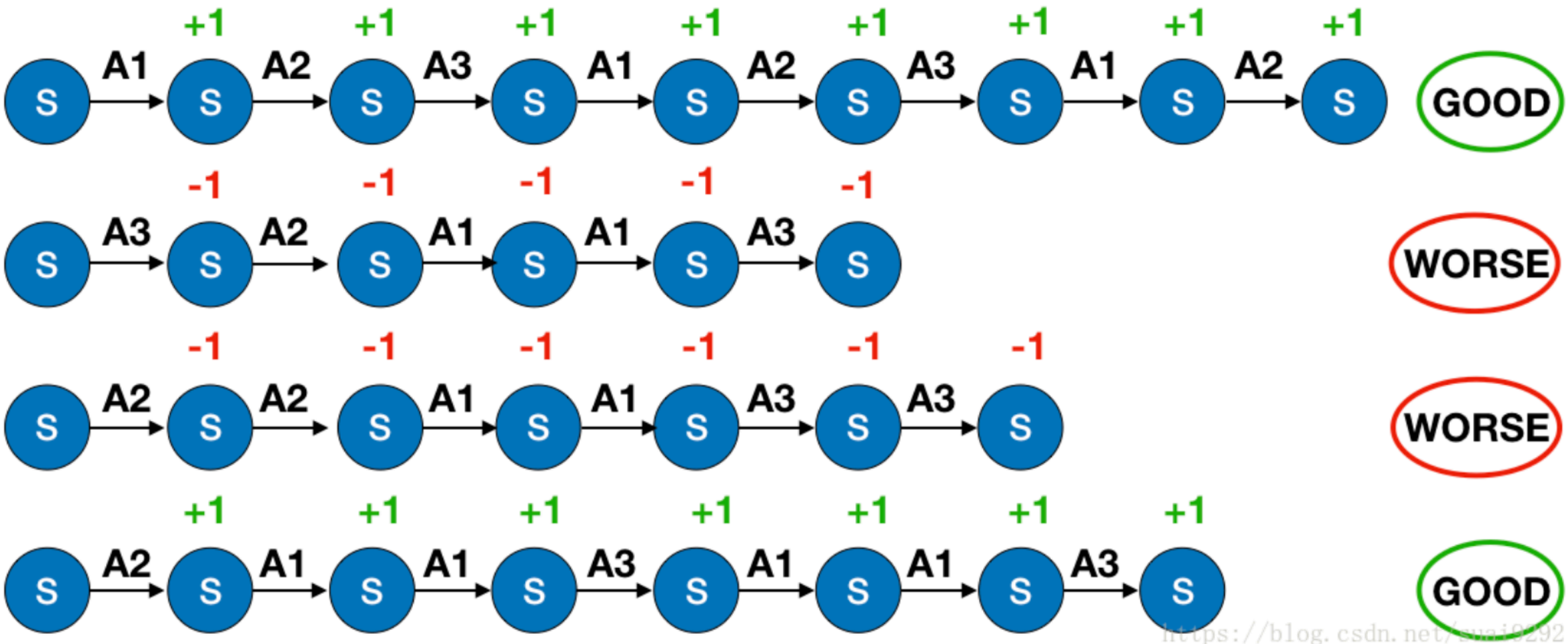
- Sample the data from θ'.
- Use the data to train θ many times.

Importance Sampling

$$E_{x \sim p}[f(x)] = E_{x \sim q}[f(x)\frac{p(x)}{q(x)}]$$

http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS18.html

Why Q-Learning?

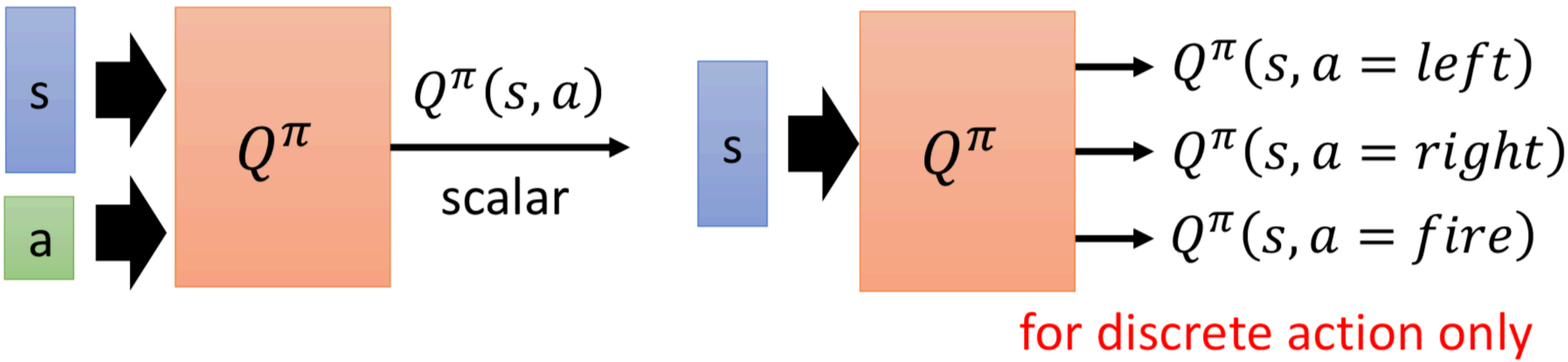


- **Suppose we win 100 games and lose 900 games altogether**
- **we'll think for the winning 100 games that the actions taken in each of these games are good for each current game state, letting all these actions update our decision network positively.**
- **And for the 900 games we lost, we thought that the actions taken in each of those games were bad, and that all of them were negative updates to our network**

https://blog.csdn.net/suai9292/article/details/79910525

Q Value Function

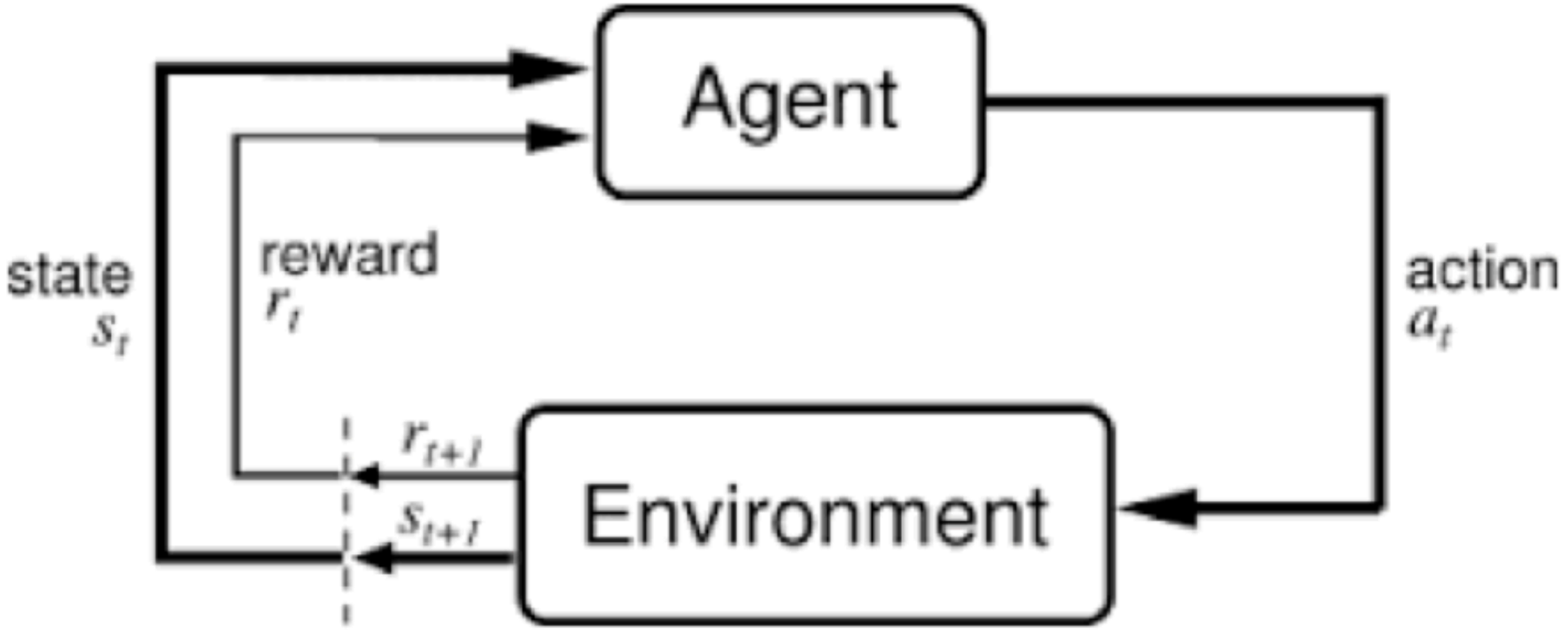
- State-action value function $Q^{\pi}(s, a)$
  - When using actor π, the *cumulated* reward expects to be obtained after taking a at state s



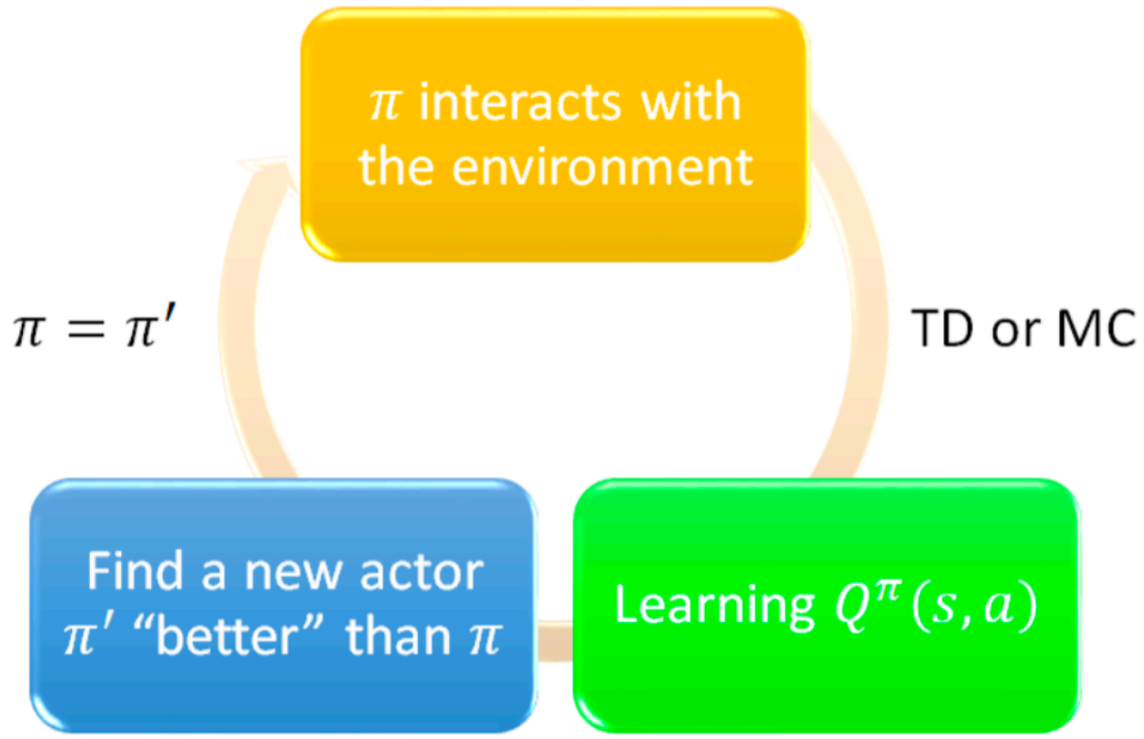
Q - learning

- Agent
- Action
- State
- Reward
- Policy(State & Action → Reward)

Notice: Also based on agent! depends!



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

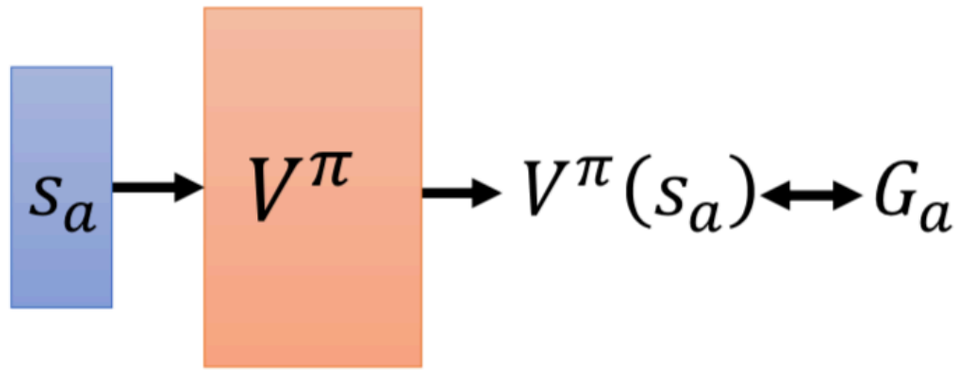
$$\pi'(s) = arg\max_{a} Q^{\pi}(s,a)$$

➢ π' does not have extra parameters. It depends on Q

- **Monte-Carlo (MC) based approach**
  - The critic watches π playing the game

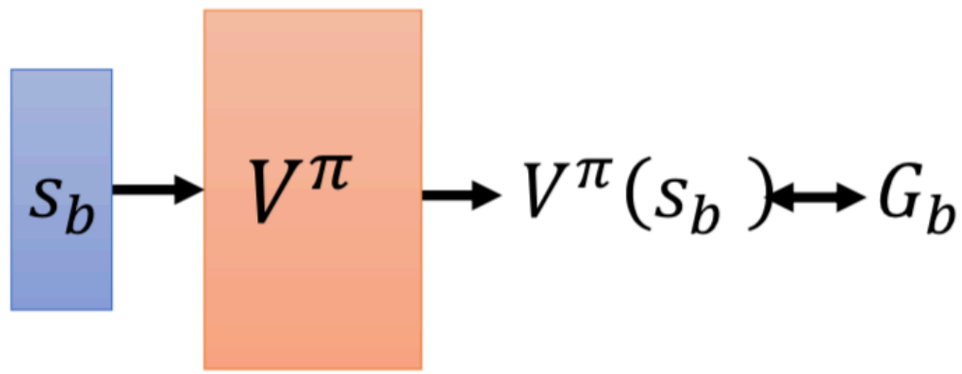
After seeing s_a,

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



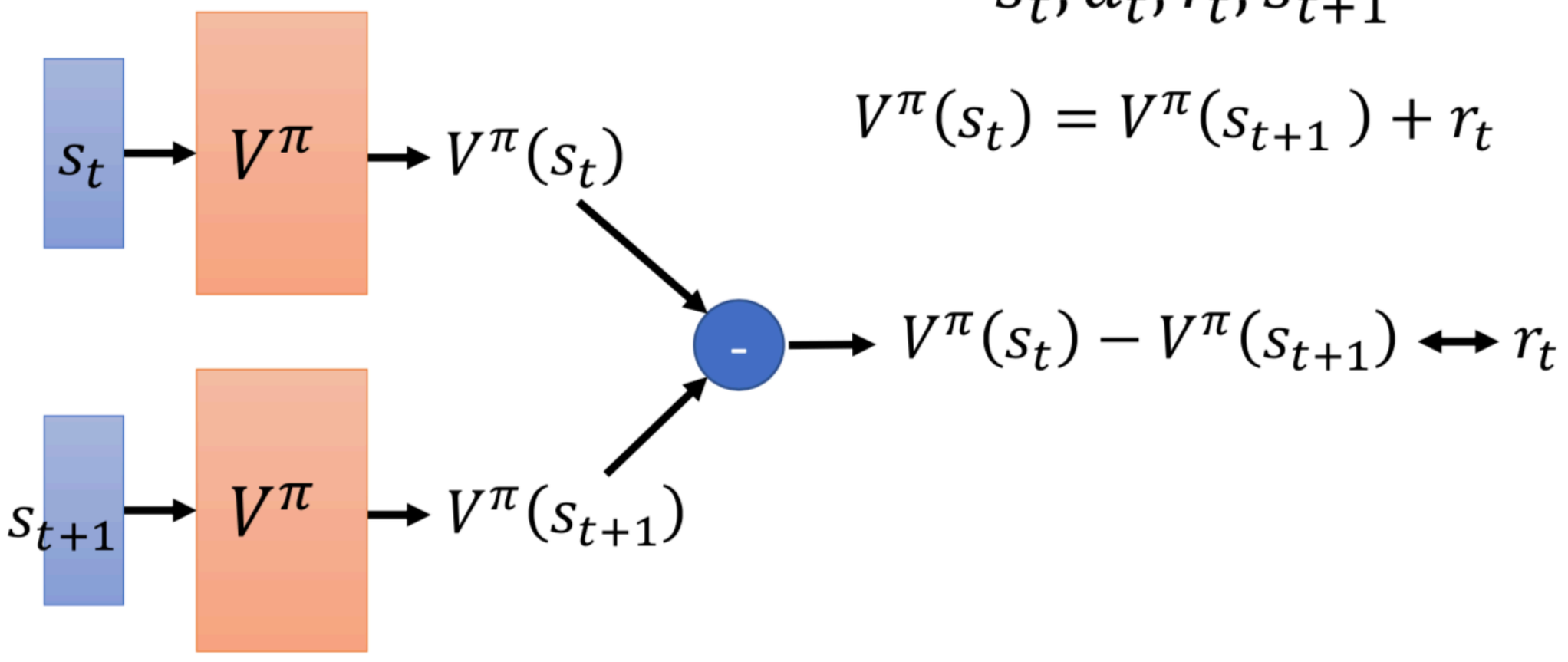
After seeing s_b,

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



http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS18.html

- **Temporal-difference (TD) 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.

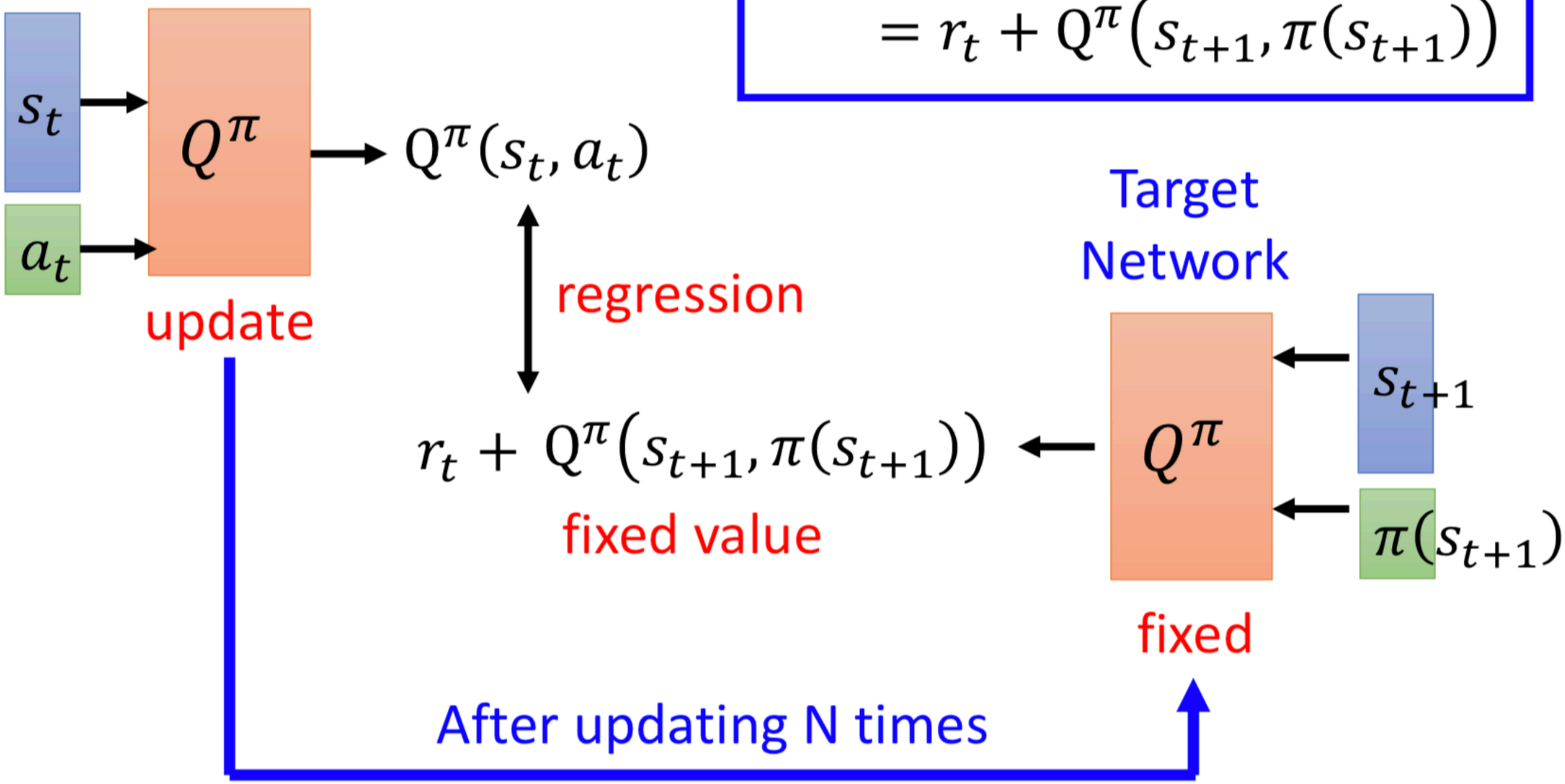
http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS18.html

Rewards can change during training?
focal loss-dynamic loss?

Target Network

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

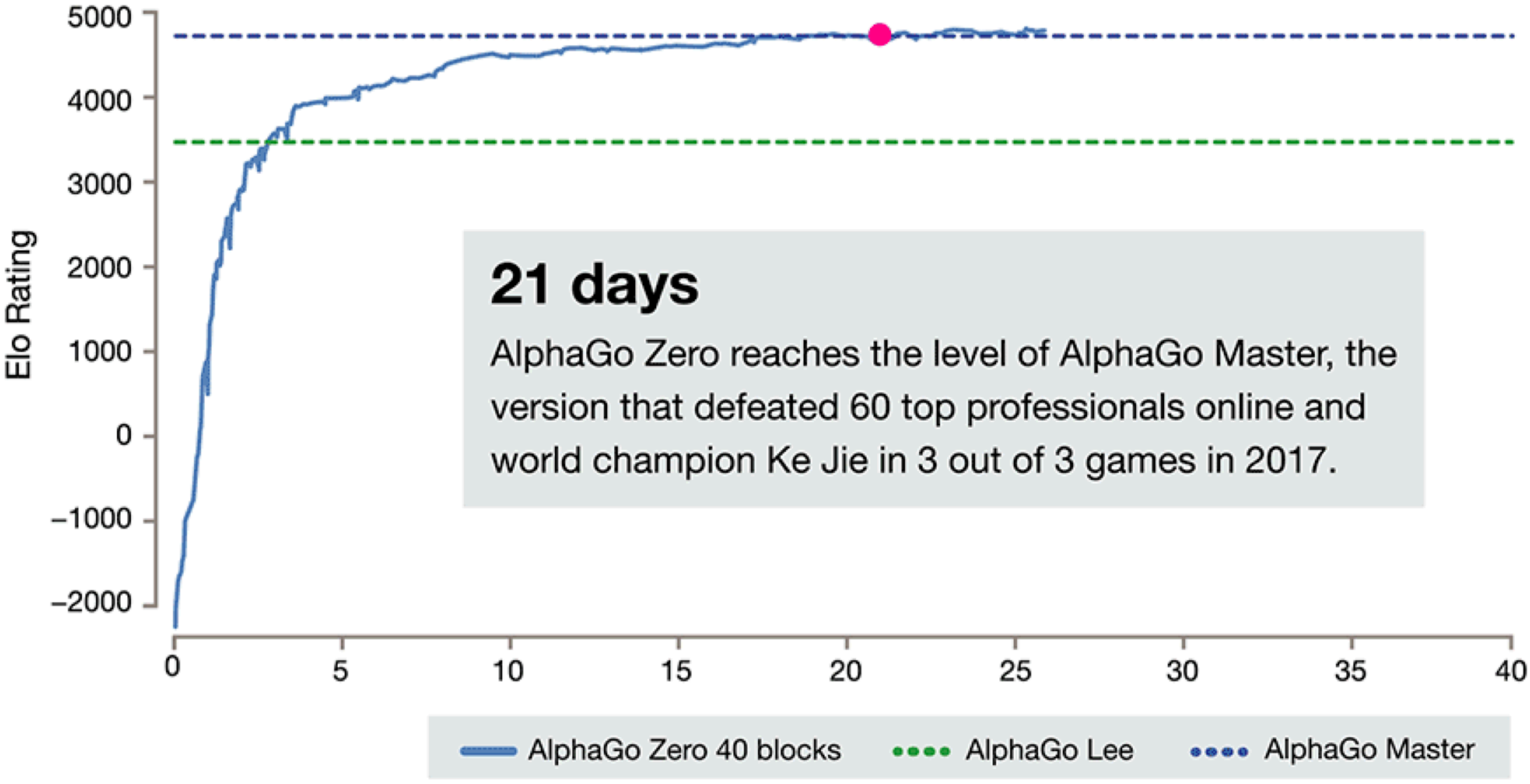
$$Q^{\pi}(s_t, a_t) = r_t + Q^{\pi}\left(s_{t+1}, \pi(s_{t+1})\right)$$



Some Key Issues for DQN

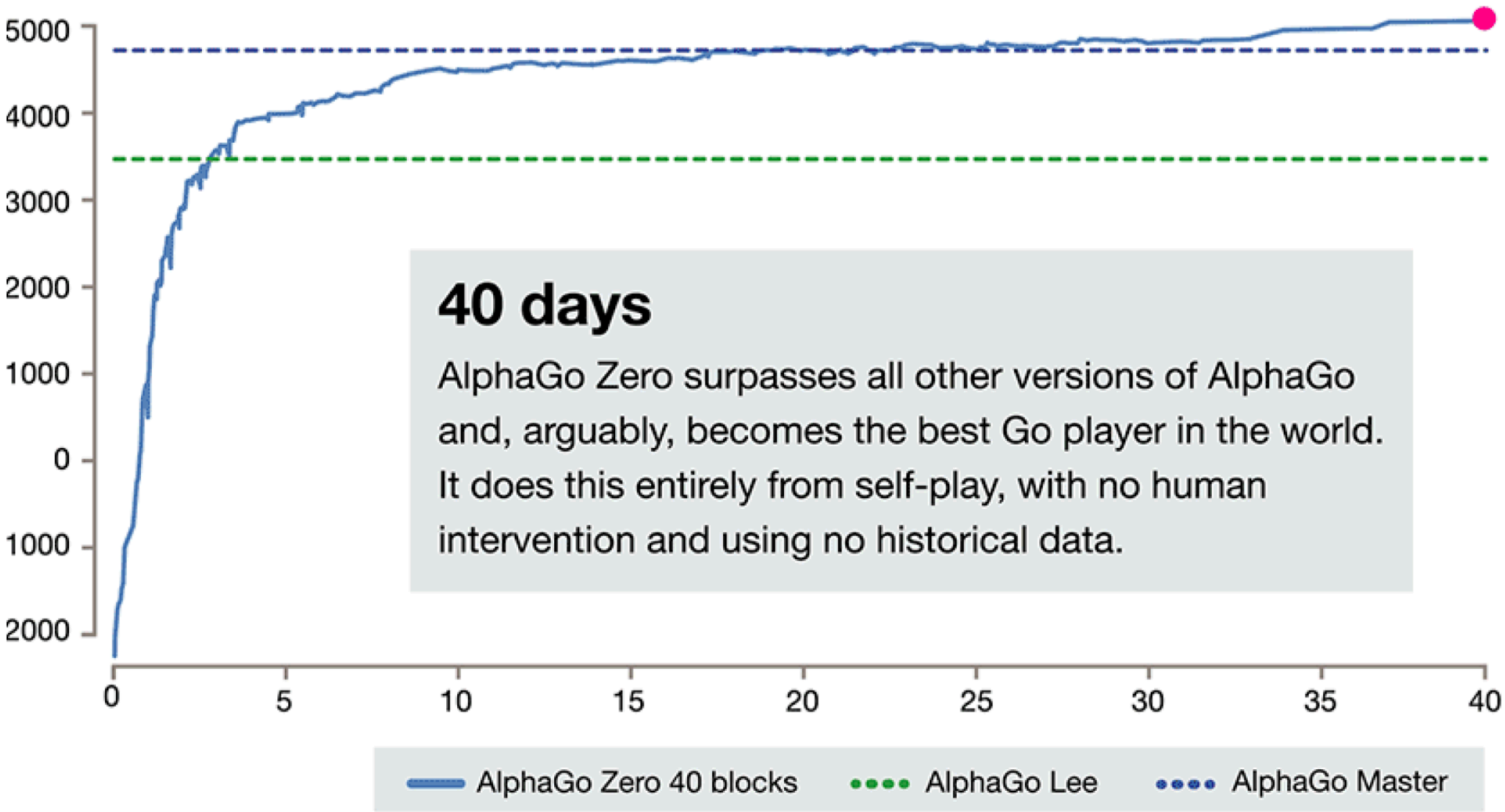
Time cost

The basic flaw of pure reinforcement learning – start from zero

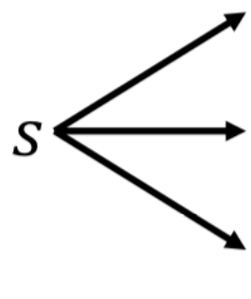


Time cost

The basic flaw of pure reinforcement learning – start from zero



Exploration



$Q(s,a) = 0$ Never explore

$Q(s,a) = 1$ Always sampled

$Q(s,a) = 0$ Never explore

- The policy is based on Q-function

$$a = arg\max_a Q(s,a)$$

This is not a good way for data collection.

Epsilon Greedy ε would decay during learning

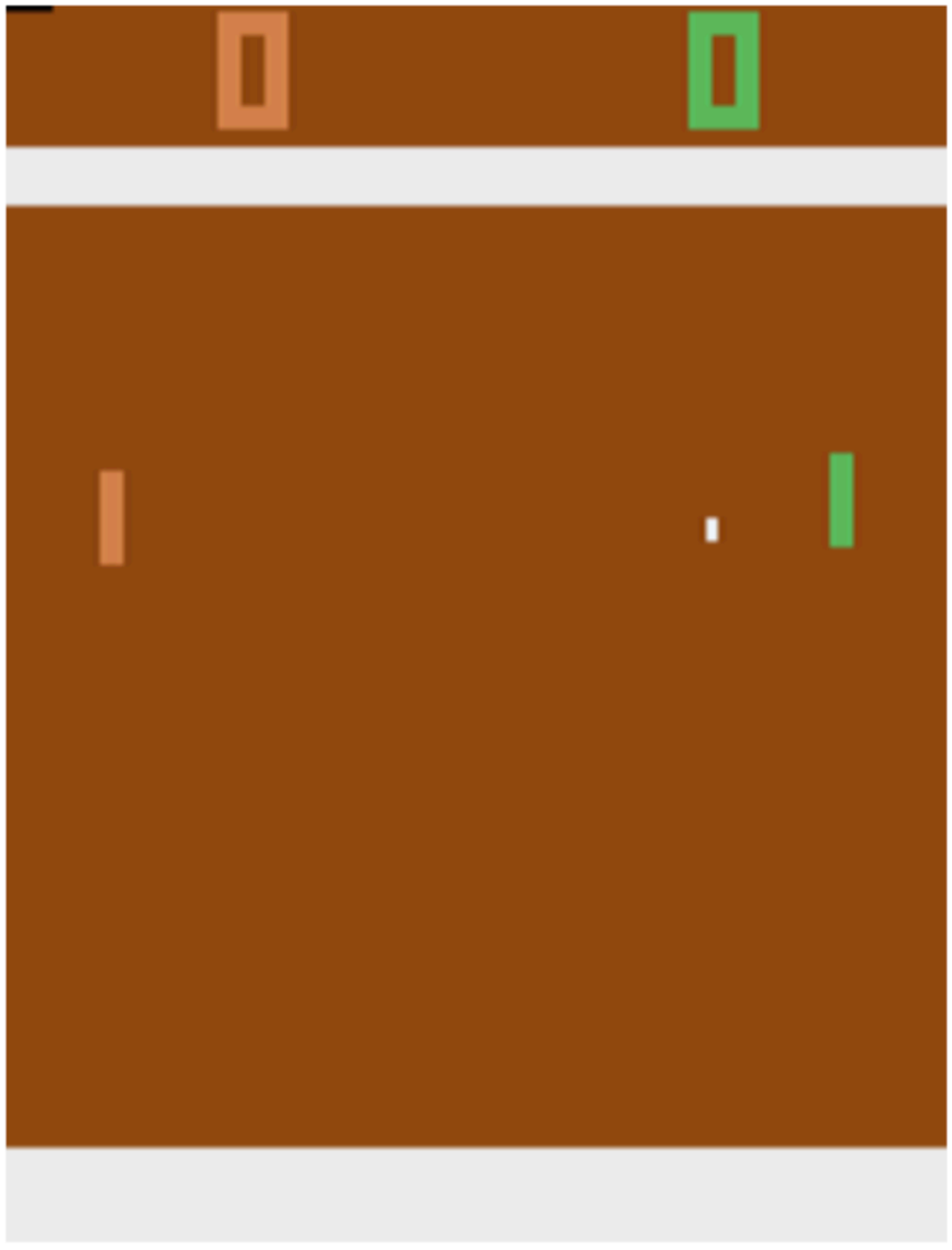
$$a = \begin{cases} arg\max_a Q(s,a), & with\ probability\ 1-\varepsilon \\ random, & otherwise \end{cases}$$

Boltzmann Exploration

$$P(a|s) = \frac{exp\big(Q(s,a)\big)}{\sum_a exp\big(Q(s,a)\big)}$$

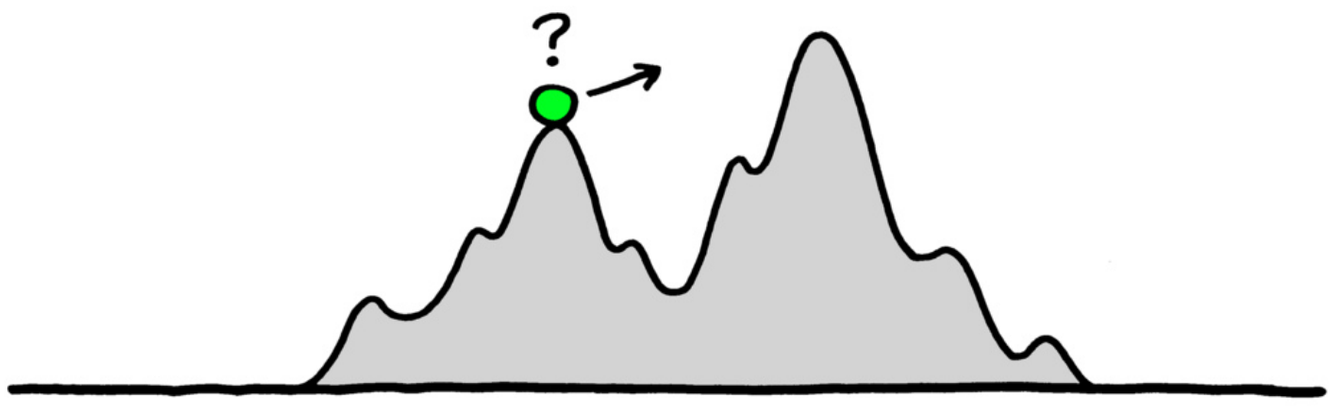
http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLDS18.html

Observation



(a) Pong

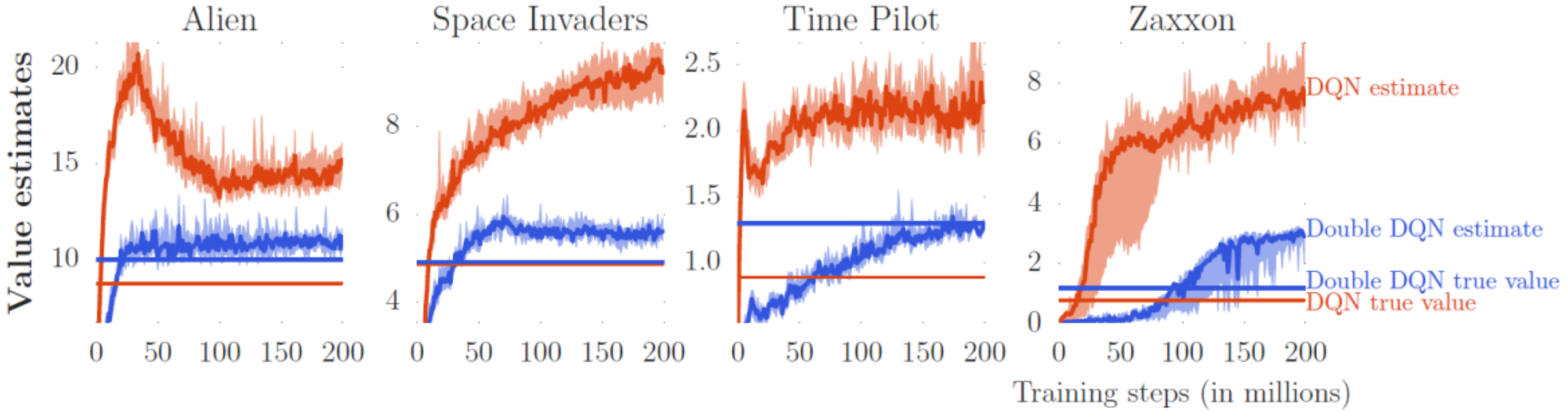
Partial In many real world tasks agent does not have the scope to observe the complete environment .



Hausknecht, Matthew, and Peter Stone. "Deep recurrent q-learning for partially observable mdps." *CoRR, abs/1507.06527* 7.1 (2015).

Over-estimated

- Q value is usually over-estimated

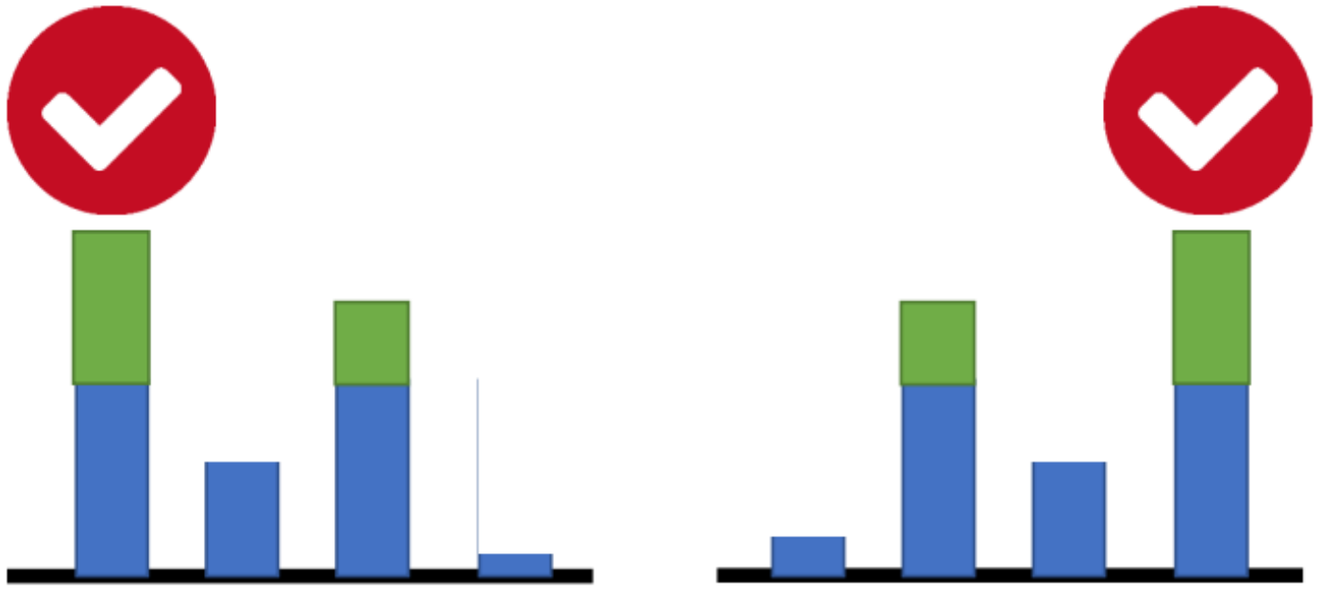


- Q value is usually over estimate

$$Q(s_t, a_t) \longleftrightarrow r_t + \max_a Q(s_{t+1}, a)$$

Tend to select the action that is over-estimated

$Q(s_{t+1}, a)$



- Q value is usually over estimate

$$Q(s_t, a_t) \longleftrightarrow r_t + \max_a Q(s_{t+1}, a)$$

- Double DQN: two functions Q and Q' Target Network

$$Q(s_t, a_t) \longleftrightarrow r_t + Q'\left(s_{t+1}, arg\max_a Q(s_{t+1}, a)\right)$$

If Q over-estimate a, so it is selected. Q' would give it proper value.

How about Q' overestimate? The action will not be selected by Q.

Hado V. Hasselt, "Double Q-learning", NIPS 2010
Hado van Hasselt, Arthur Guez, David Silver, "Deep Reinforcement Learning with Double Q-learning", AAAI 2016

Reward delay

- **In a Markov process we set R as the total reward**

$$R = r_1 + r_2 + r_3 + \ldots + r_n$$

- **The total future reward at time t can be expressed as:**

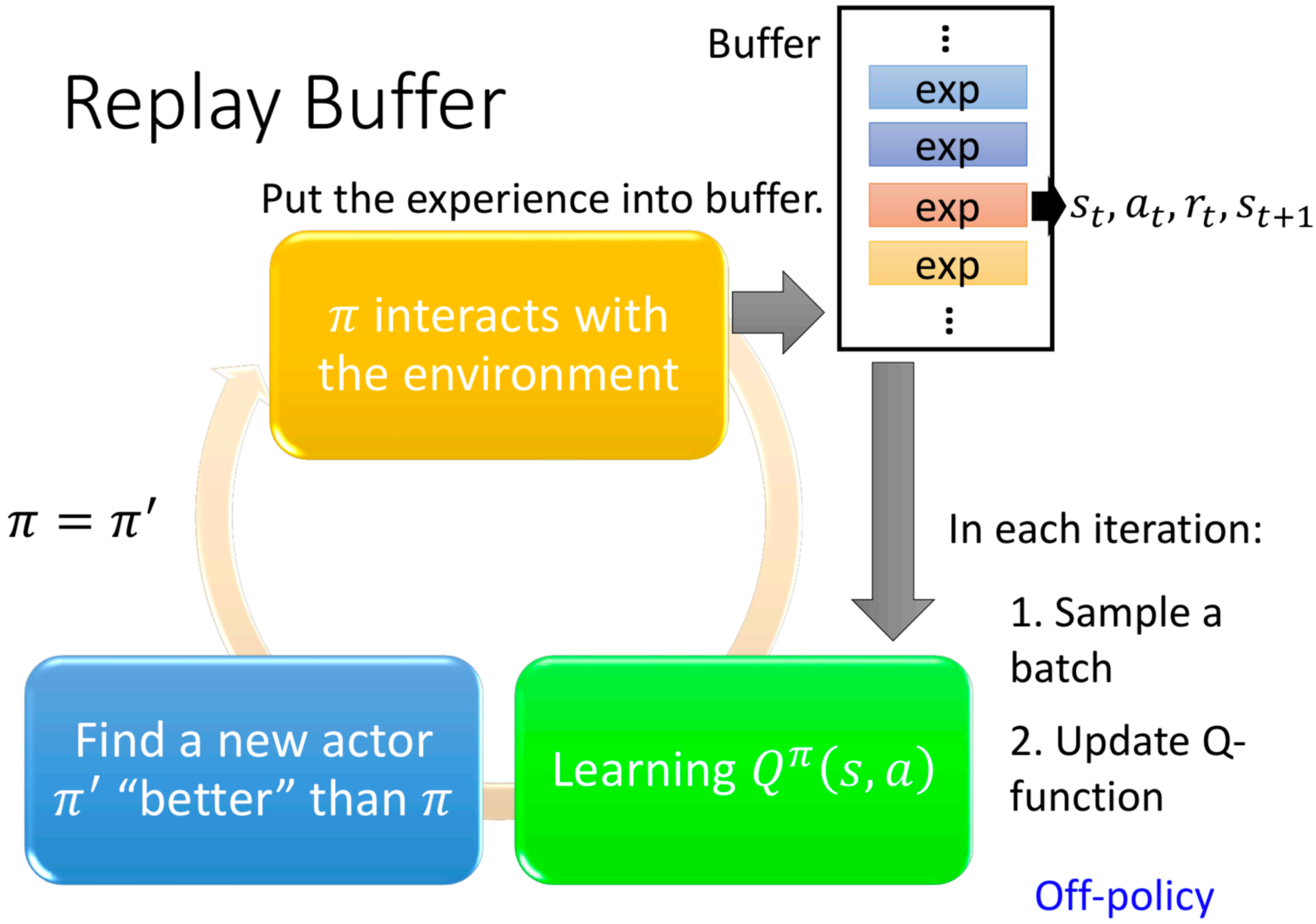
$$R_t = r_t + r_{t+1} + r_{t+2} + \ldots + r_n$$

- **These are all future rewards, now we make a discount (γ is the discount factor)**

$$R_t = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \ldots)) = r_t + \gamma R_{t+1}$$

  - **The larger the γ, the longer the long-term vision,**
  - **The smaller the γ, the more utilitarian attention to the current reward.**

Replay Buffer



Application

- **Robotics and industrial automation**
- **Play game**

Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602* (2013).

Silver, David, et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm." *arXiv preprint arXiv:1712.01815* (2017).

- **Recommend System— take KDD 2018 as an example**

Supervised Reinforcement Learning with Recurrent Neural Network for Dynamic Treatment Recommendation

Lu Wang (East China Normal University); Wei Zhang (East China Normal University); Xiaofeng He (East China Normal University); Hongyuan Zha (Georgia Institute of Technology)

Stablizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation

Shi-Yong Chen (Nanjing University); Yang Yu (Nanjing University); Qing Da (Alibaba Group); Jun Tan (Alibaba Group); Hai-Kuan Huang (Alibaba Group); Hai-Hong Tang (Alibaba Group)

Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning

Xiangyu Zhao (Michigan State University); Liang Zhang (JD.com); Zhuoye Ding (JD.com); Long Xia (Data Science Lab, JD.com); Jiliang Tang (Michigan State University); Dawei Yin (JD.com)

Recommend paper

【1】Johnson J D, Li J, Chen Z. Reinforcement Learning: An Introduction : R.S. Sutton, A.G. Barto, MIT Press, Cambridge, MA 1998, 322 pp. ISBN 0-262-19398-1[J]. Neurocomputing, 2000, 35(1):205-206. (Q-learning)

【2】Kröse B J A. Learning from delayed rewards[J]. Robotics & Autonomous Systems, 1995, 15(4):233-235. (reward delayed)

【3】Tesauro G. Practical issues in temporal difference learning[J]. Machine Learning, 1992, 8(3-4):257-277. (Temporal difference)

【4】Cohen J D, Mcclure S M, Yu A J. Should I Stay or Should I Go? How the Human Brain Manages the Trade-off between Exploitation and Exploration[J]. Philosophical Transactions Biological Sciences, 2007, 362(1481):933. (trade-off between Exploitation and Exploration)

【5】Kaelbling L P, Littman M L, Moore A W. Reinforcement Learning: A Survey[J]. J Artificial Intelligence Research, 1996, 4(1):237–285.(Survey)

Thanks and QA