Q-Learning

Hung-yi Lee

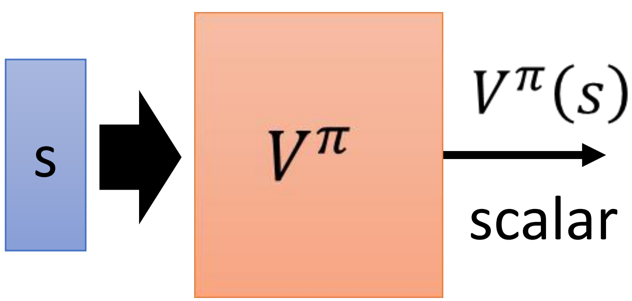
Outline

- Introduction of Q-Learning
- Tips of Q-Learning
- Q-Learning for Continuous Actions

Critic

The output values of a critic depend on the actor evaluated.

- A critic does not directly determine the action.
- Given an actor π, it evaluates how good the actor is
- State value function $V^{\pi}(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after visiting state s



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

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

Critic

$$V^{\text{以前的阿光}}(\text{大馬步飛}) = \text{bad}$$
$$V^{\text{變強的阿光}}(\text{大馬步飛}) = \text{good}$$

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

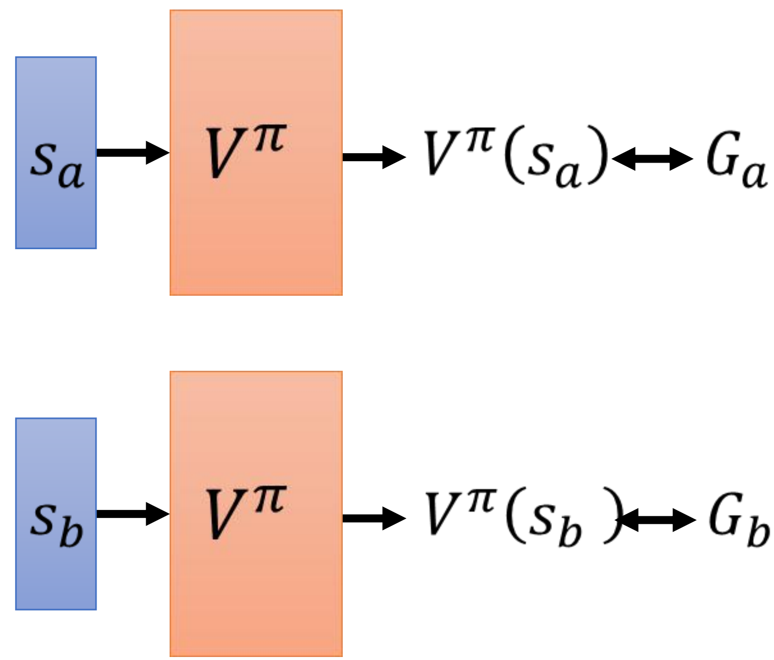
- **Monte-Carlo (MC) based approach**
  - The critic watches π playing the game

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

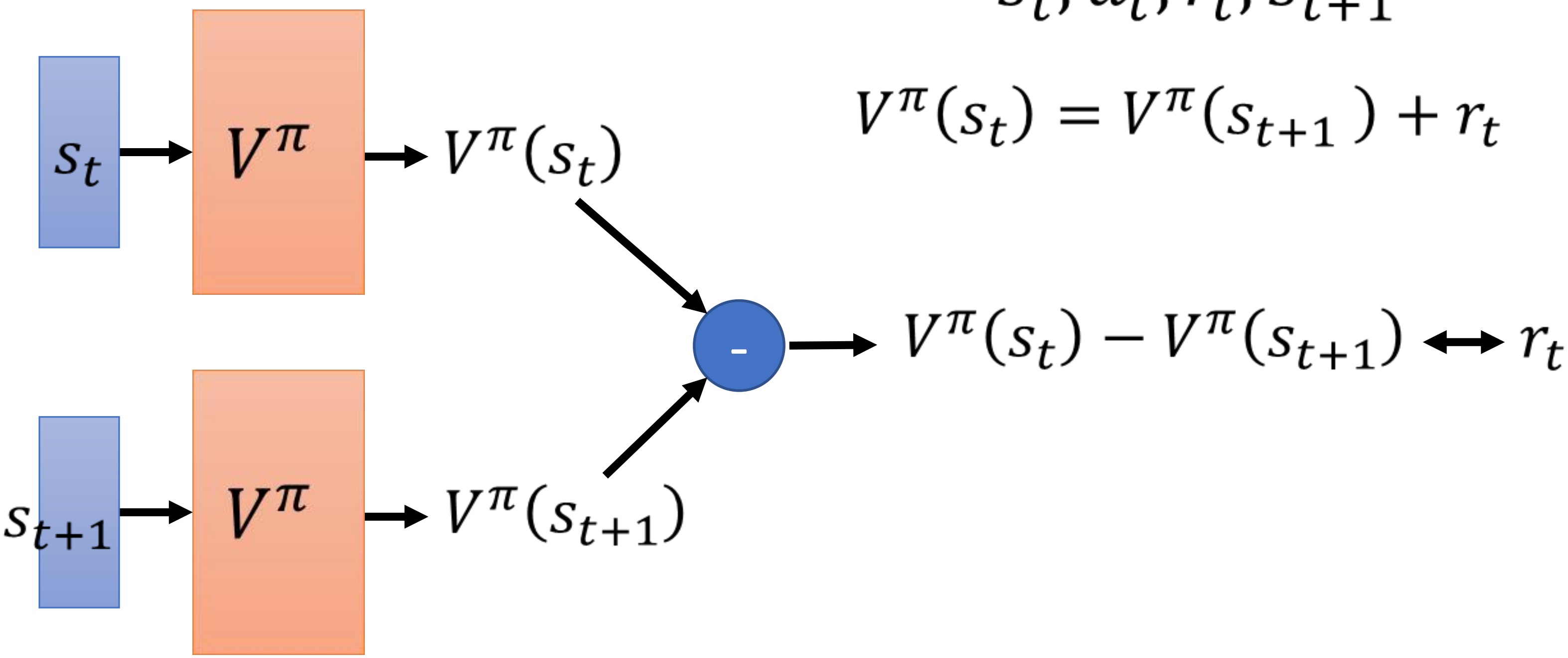


How to estimate $V^{\pi}(s)$

- **Temporal-difference (TD) approach**

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

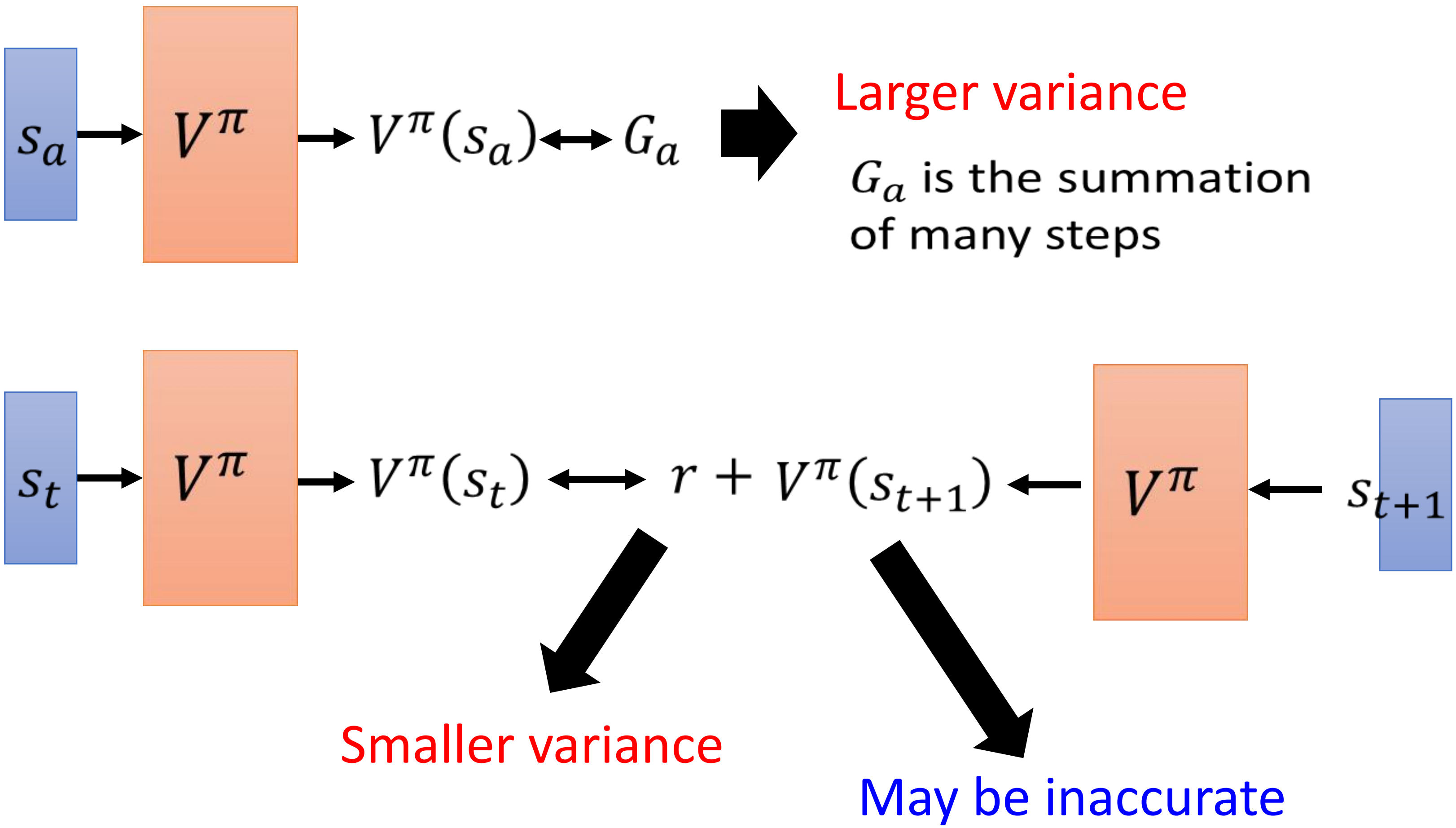
$$V^{\pi}(s_t) = V^{\pi}(s_{t+1}) + r_t$$



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

$$Var[kX] = k^2 Var[X]$$

MC v.s. TD



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: $V^\pi(s_a) = 0$

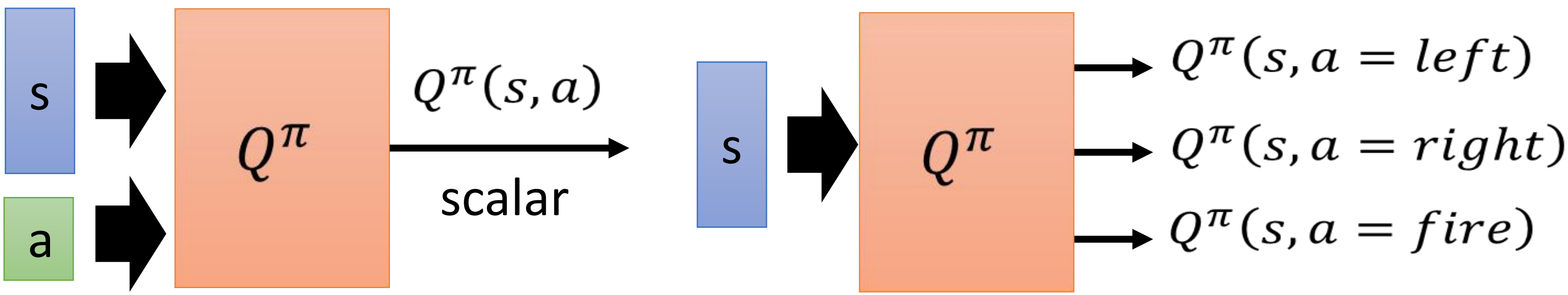
Temporal-difference:

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

(The actions are ignored here.)

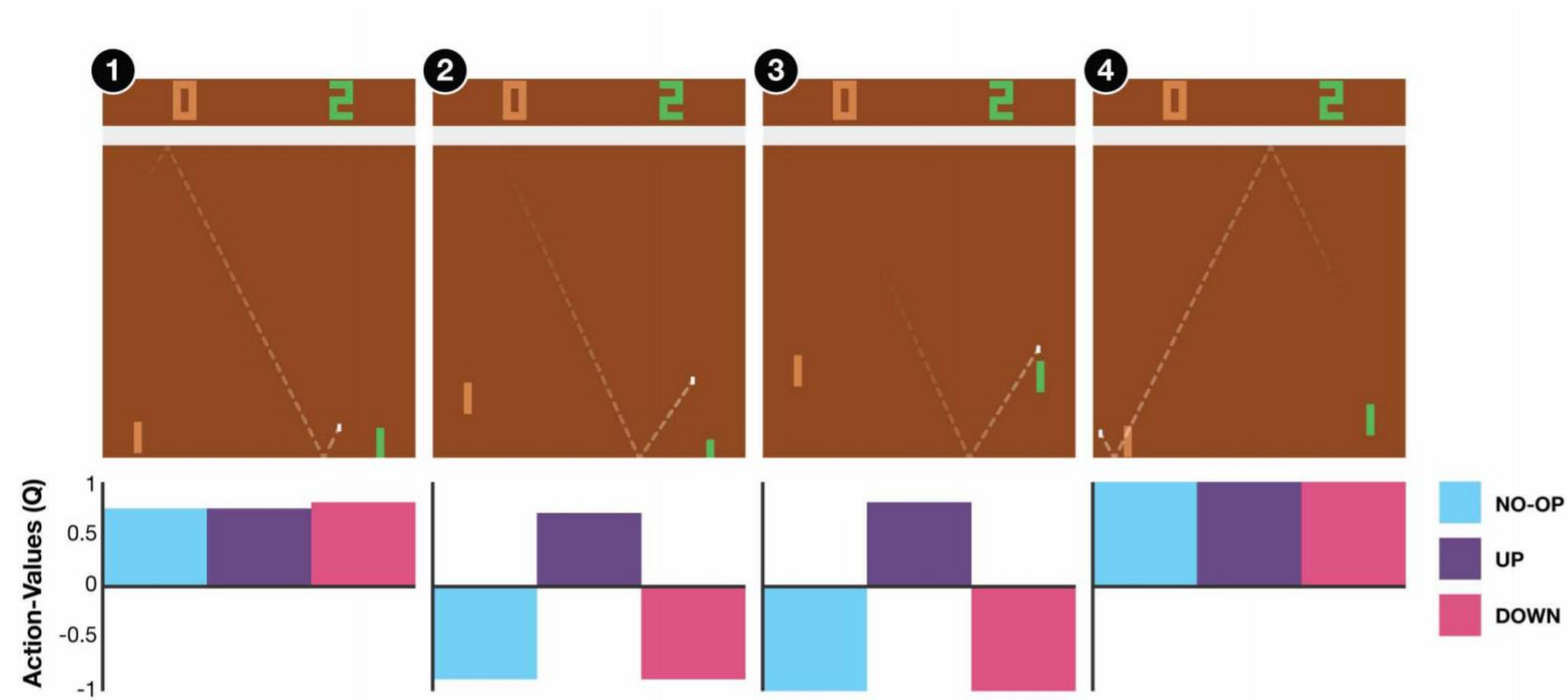
Another Critic

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



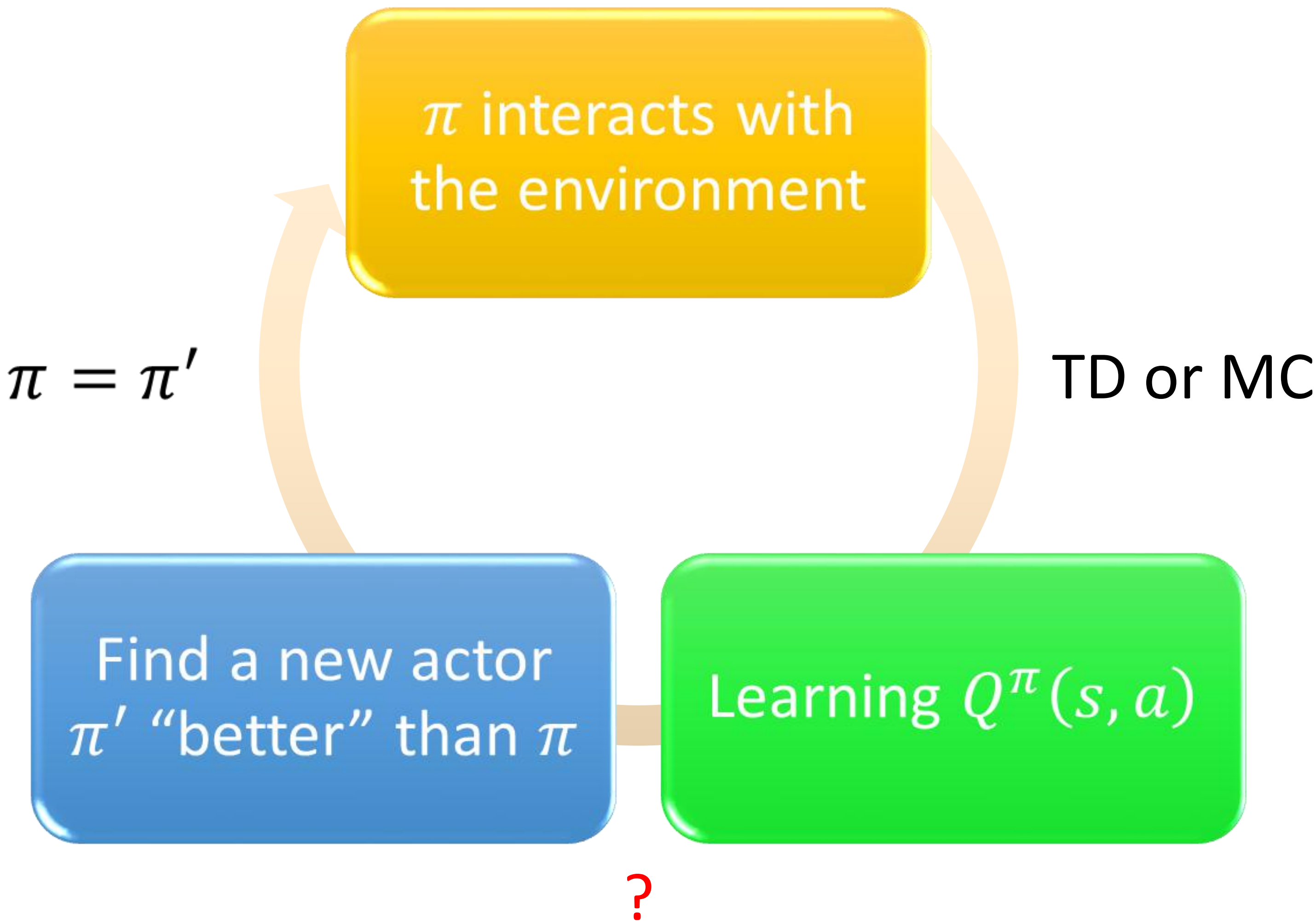
for discrete action only

State-action value function

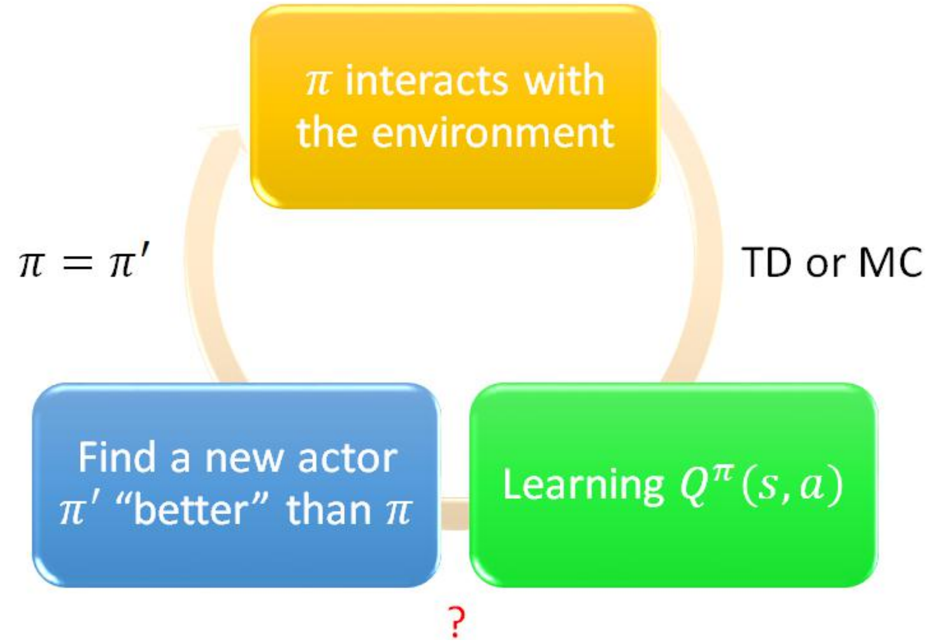


https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf

Another Way to use Critic: Q-Learning



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 (solve it later)

Q-Learning

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

$$V^{\pi'}(s) \geq V^{\pi}(s), \text{ for all state s}$$

$$V^{\pi}(s) = Q^{\pi}\big(s,\pi(s)\big)$$

$$\leq \max_a Q^{\pi}(s,a) \ = Q^{\pi}\big(s,\pi'(s)\big)$$

$$V^{\pi}(s) \leq Q^{\pi}\big(s,\pi'(s)\big)$$

$$= E[r_{t+1} + V^{\pi}(s_{t+1})|s_t = s, a_t = \pi'(s_t)]$$

$$\leq E[r_{t+1} + Q^{\pi}\big(s_{t+1},\pi'(s_{t+1})\big)|s_t = s, a_t = \pi'(s_t)]$$

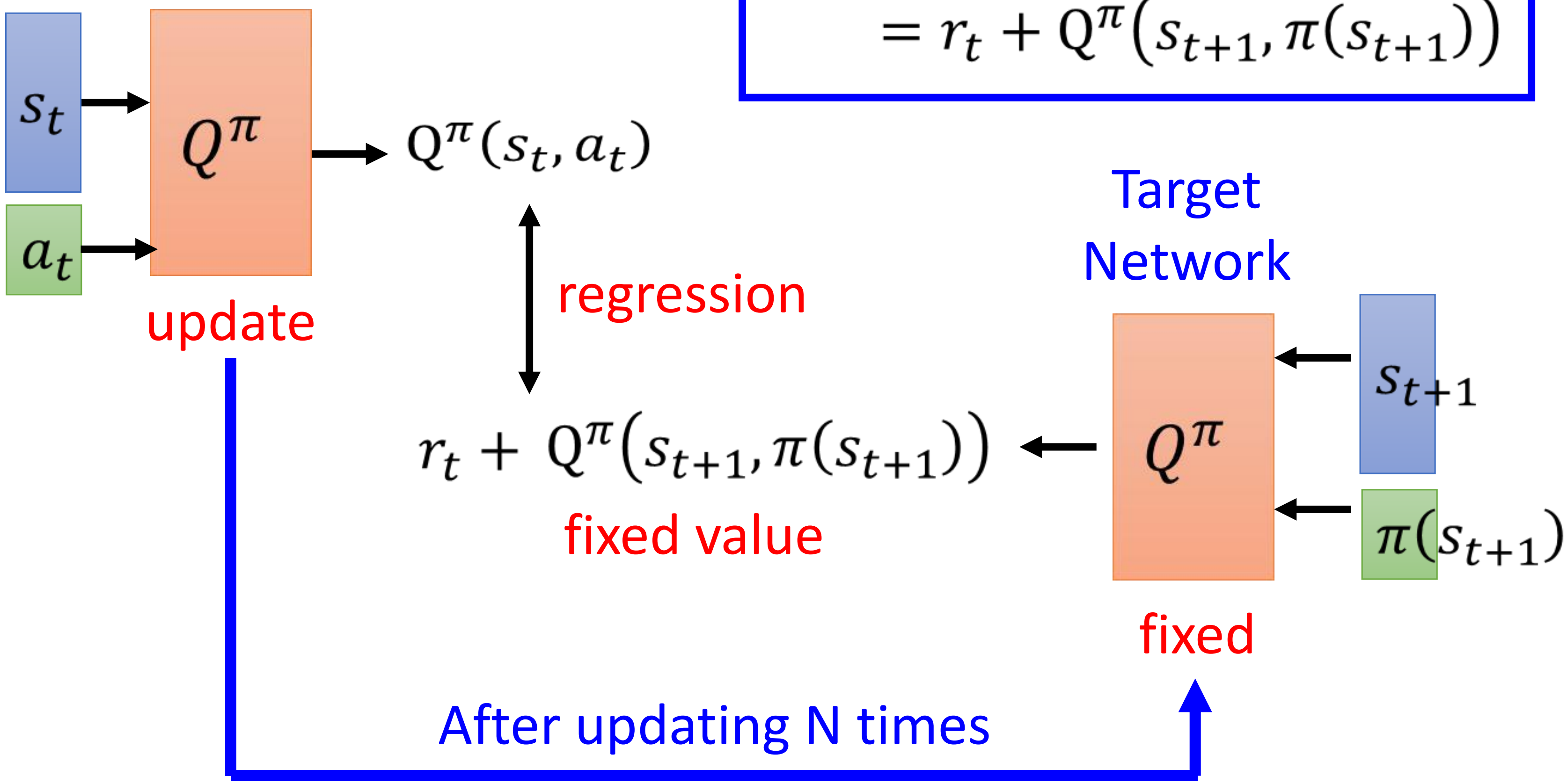
$$= E[r_{t+1} + r_{t+2} + V^{\pi}(s_{t+2}\ )|\ldots]$$

$$\leq E[r_{t+1} + r_{t+2} + Q^{\pi}\big(s_{t+2},\pi'(s_{t+2})\big)|\ldots]\ \ldots \ \leq V^{\pi'}(s)$$

Target Network

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

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



Exploration

$s \rightarrow a_1$ $\quad Q(s,a) = 0$ Never explore

$s \rightarrow a_2$ $\quad Q(s,a) = \boxed{1}$ Always sampled

$s \rightarrow a_3$ $\quad Q(s,a) = 0$ 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.

<u>Epsilon Greedy</u>

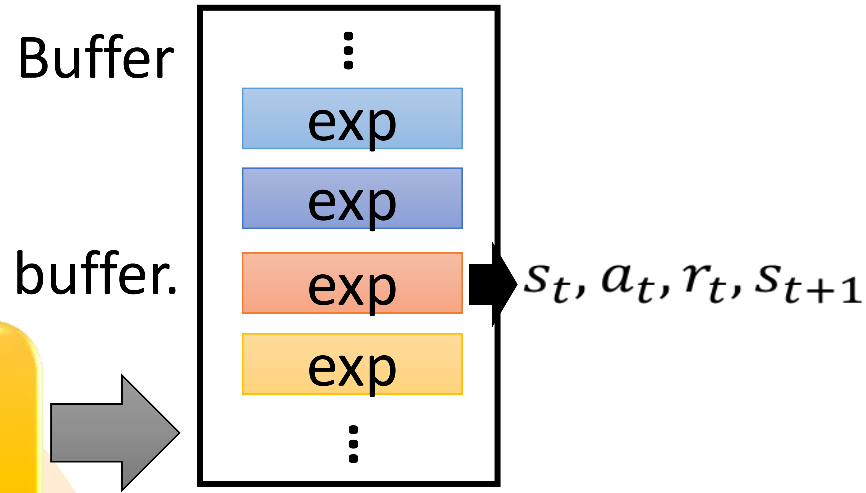
ε would decay during learning

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

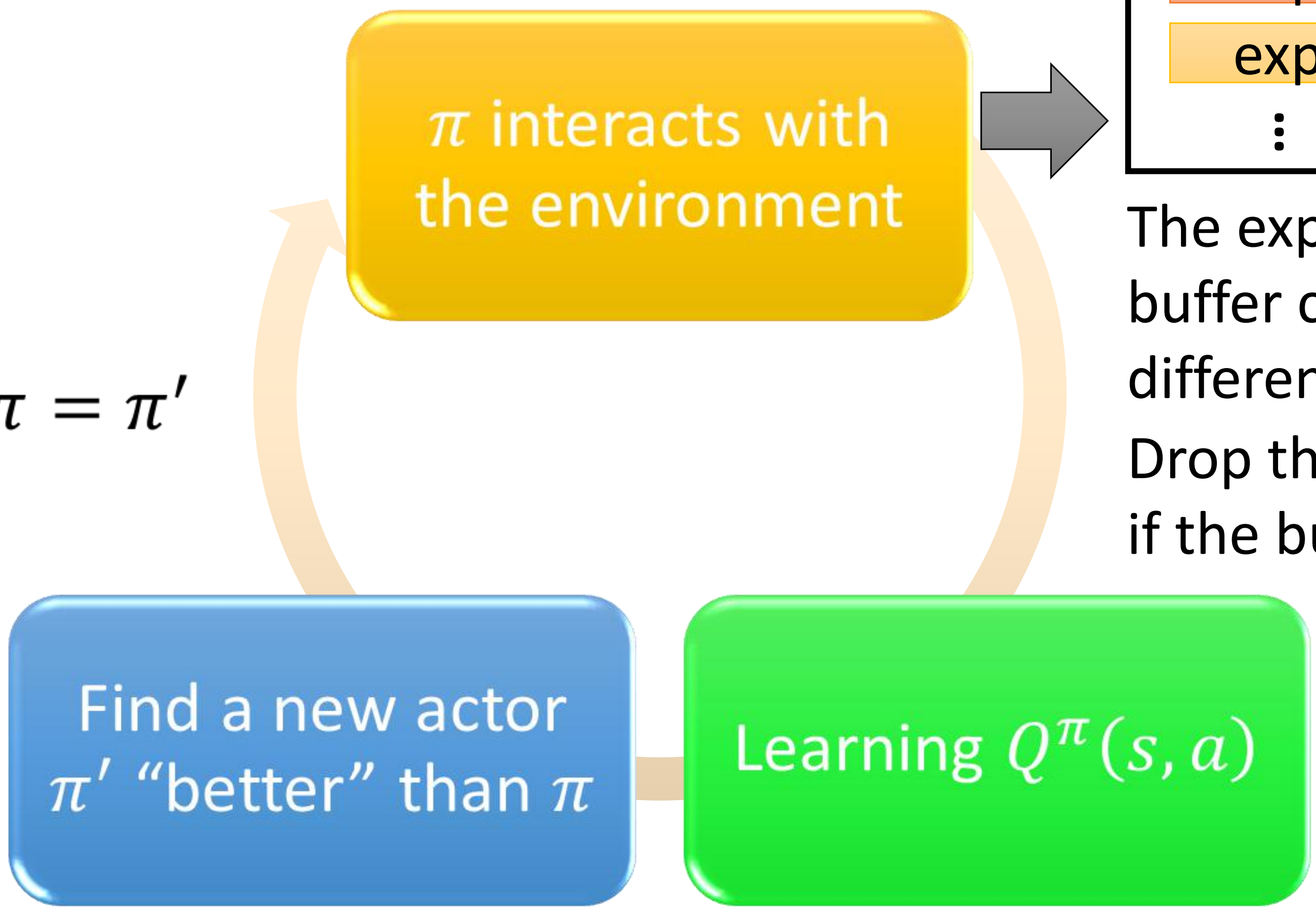
<u>Boltzmann Exploration</u>

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

Replay Buffer



Put the experience into buffer.

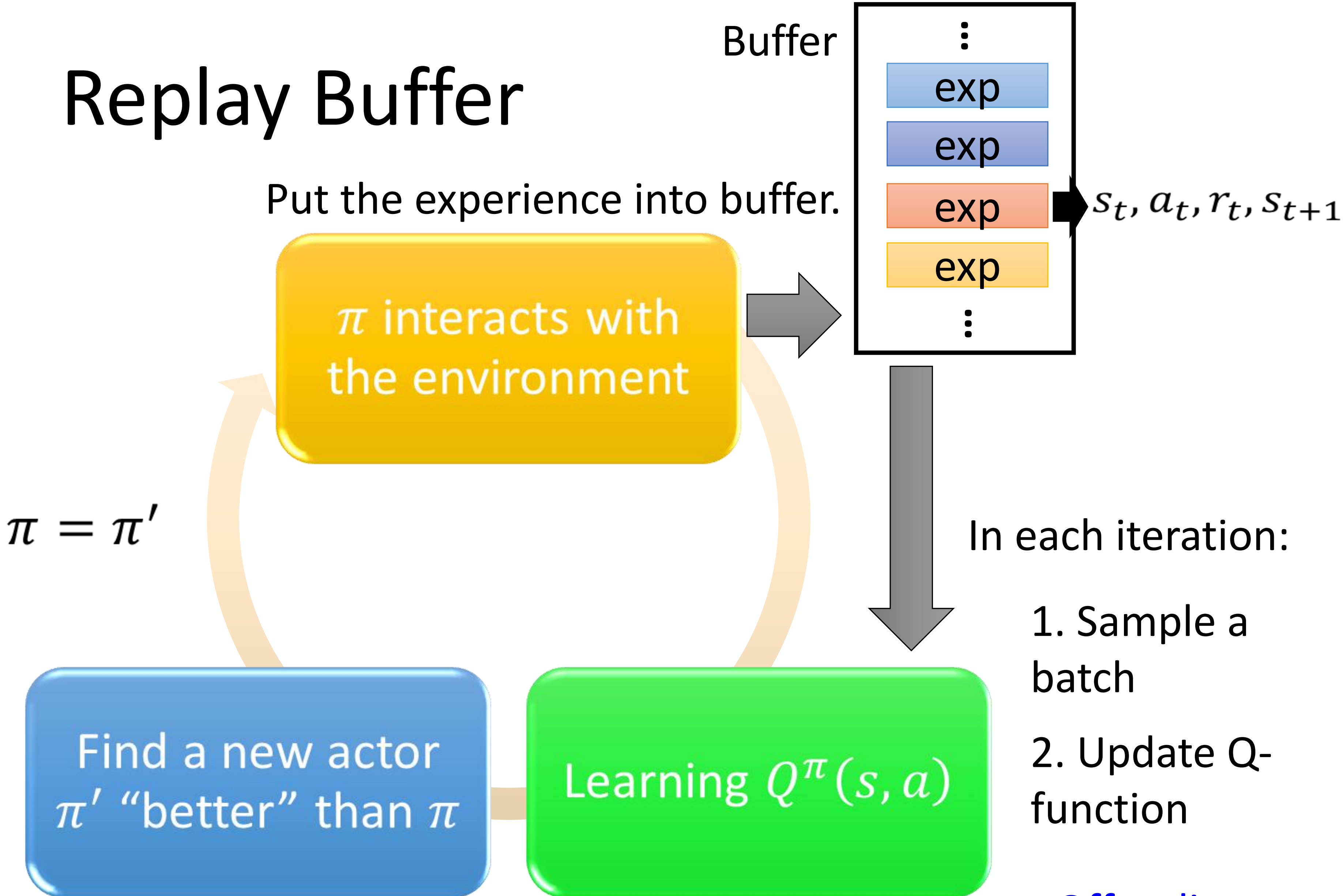


$\pi = \pi'$

The experience in the buffer comes from different policies.

Drop the old experience if the buffer is full.

Replay Buffer



In each iteration:

1. Sample a batch
2. Update Q-function

Off-policy

Typical Q-Learning Algorithm

- Initialize Q-function Q, target Q-function $\hat{Q} = Q$
- In each episode
  - For each time step t
    - Given state s_t, take action a_t based on Q (epsilon greedy)
    - Obtain reward r_t, and reach new state s_{t+1}
    - Store (s_t, a_t, r_t, s_{t+1}) into buffer
    - Sample (s_i, a_i, r_i, s_{i+1}) from buffer (usually a batch)
    - Target $y = r_i + \max_a \hat{Q}(s_{i+1}, a)$
    - Update the parameters of Q to make $Q(s_i, a_i)$ close to y (regression)
    - Every C steps reset $\hat{Q} = Q$

Outline

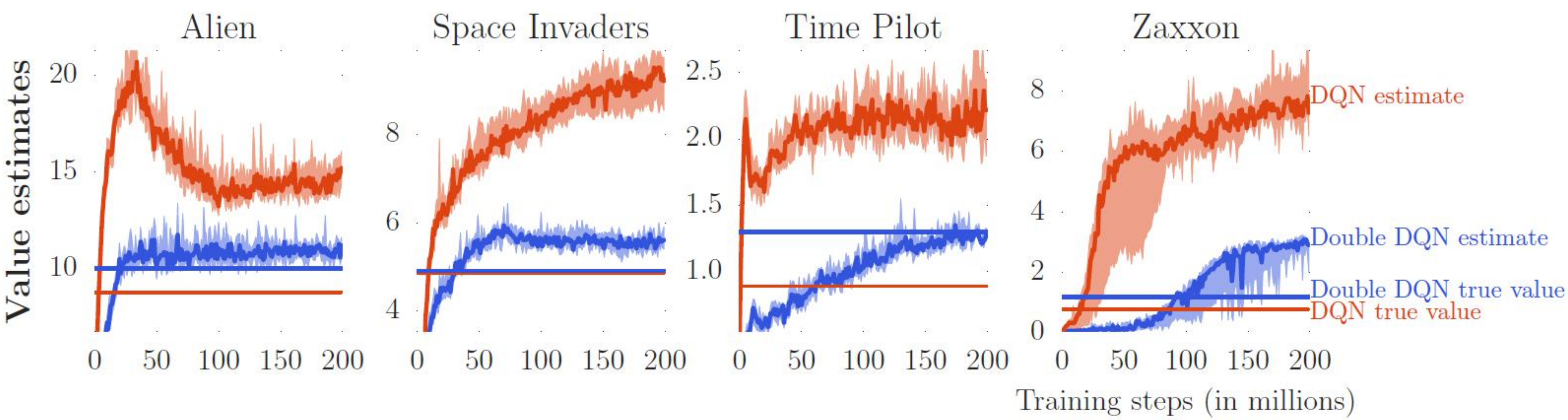
Introduction of Q-Learning

Tips of Q-Learning

Q-Learning for Continuous Actions

Double DQN

- Q value is usually over-estimated



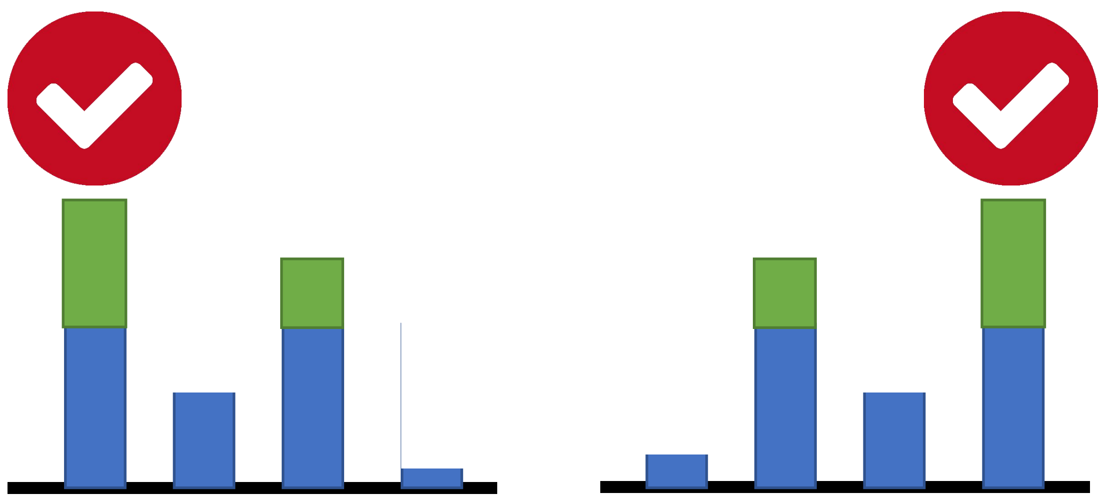
Double DQN

- 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

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



Double DQN

- 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

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

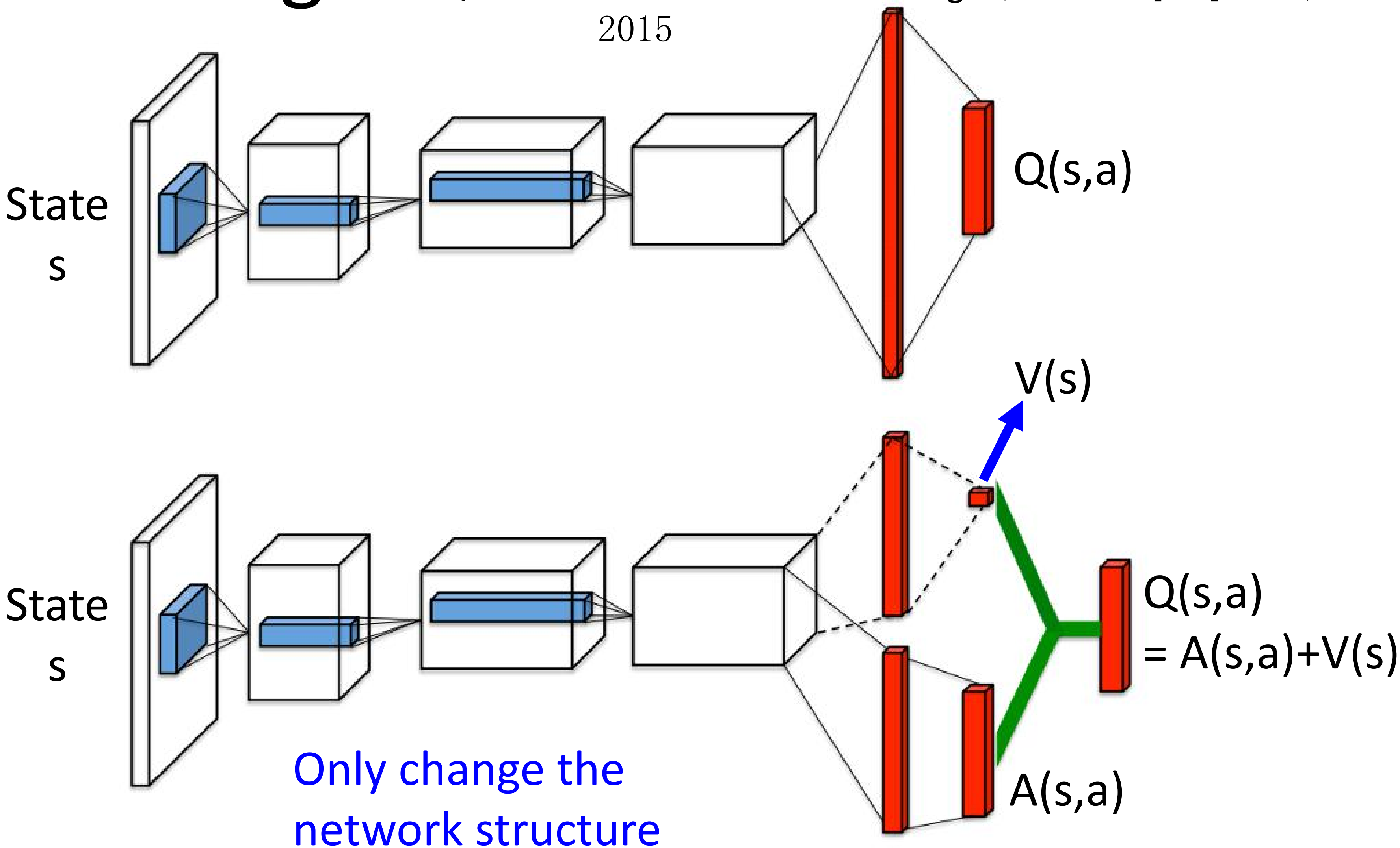
If Q over-estimate a, so it is selected. Q' would give it proper value.

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

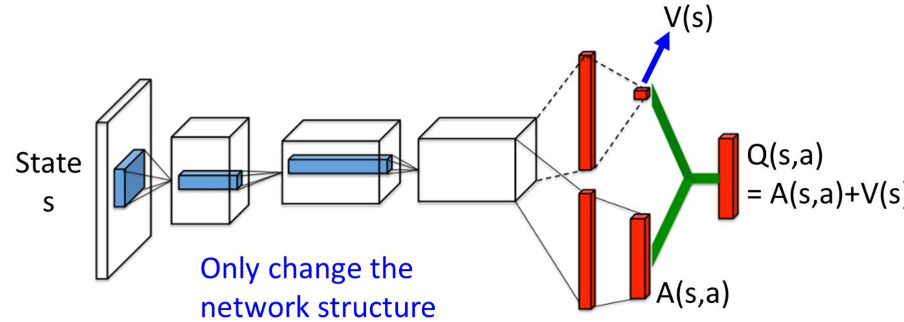
Dueling DQN

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas, “Dueling Network Architectures for Deep Reinforcement Learning”, arXiv preprint, 2015



Only change the network structure

Dueling DQN



state

Q(s,a)

action				
	3	~~3~~ **4**	3	1
	1	~~-1~~ **0**	6	1
	2	~~-2~~ **-1**	3	1

=

V(s) Average of column

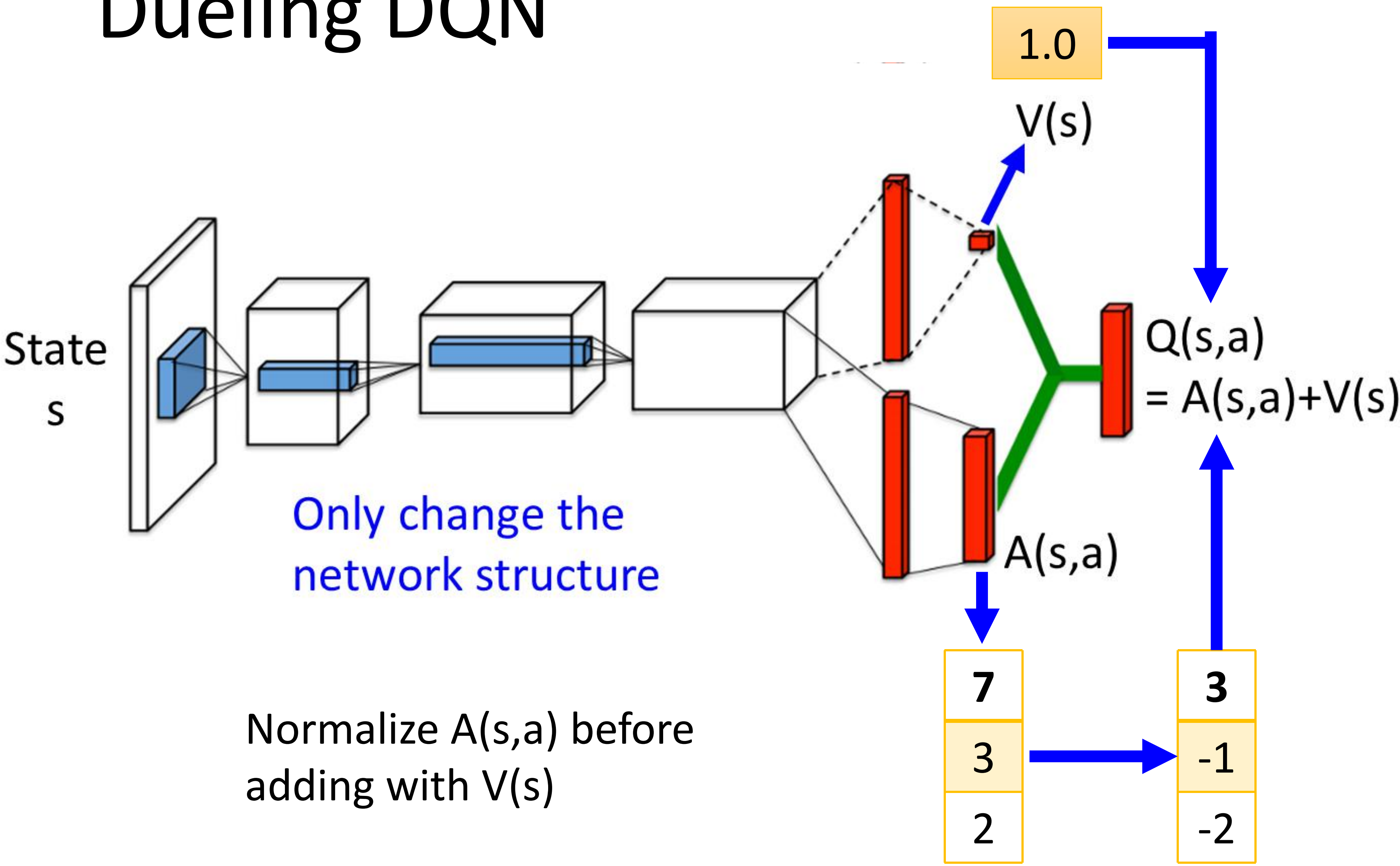
2	~~0~~ **1**	4	1

+

A(s,a) sum of column = 0

1	3	-1	0
-1	-1	2	0
0	-2	-1	0

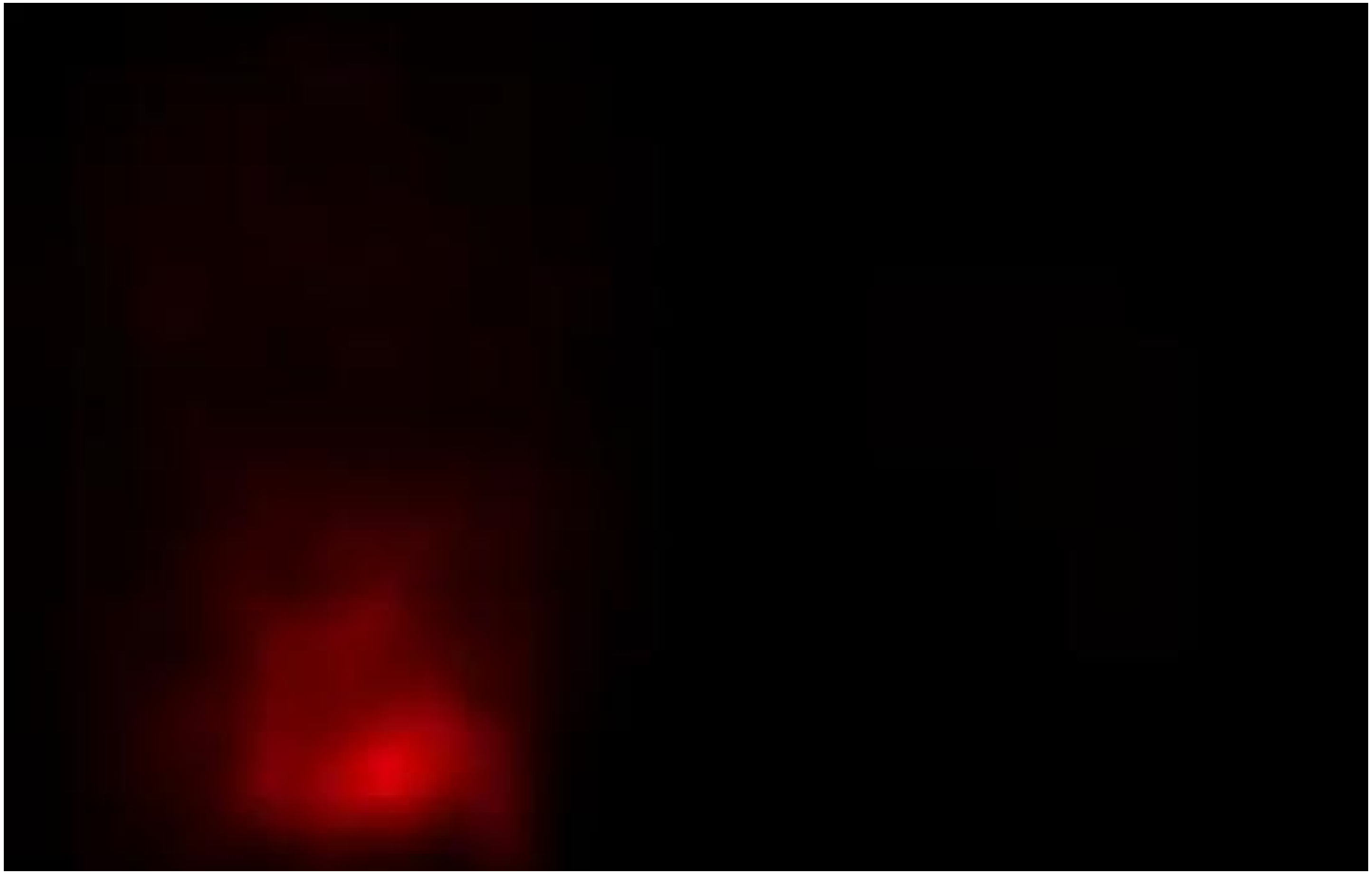
Dueling DQN



Only change the network structure

Normalize A(s,a) before adding with V(s)

Dueling DQN - Visualization



(from the link of the original paper)

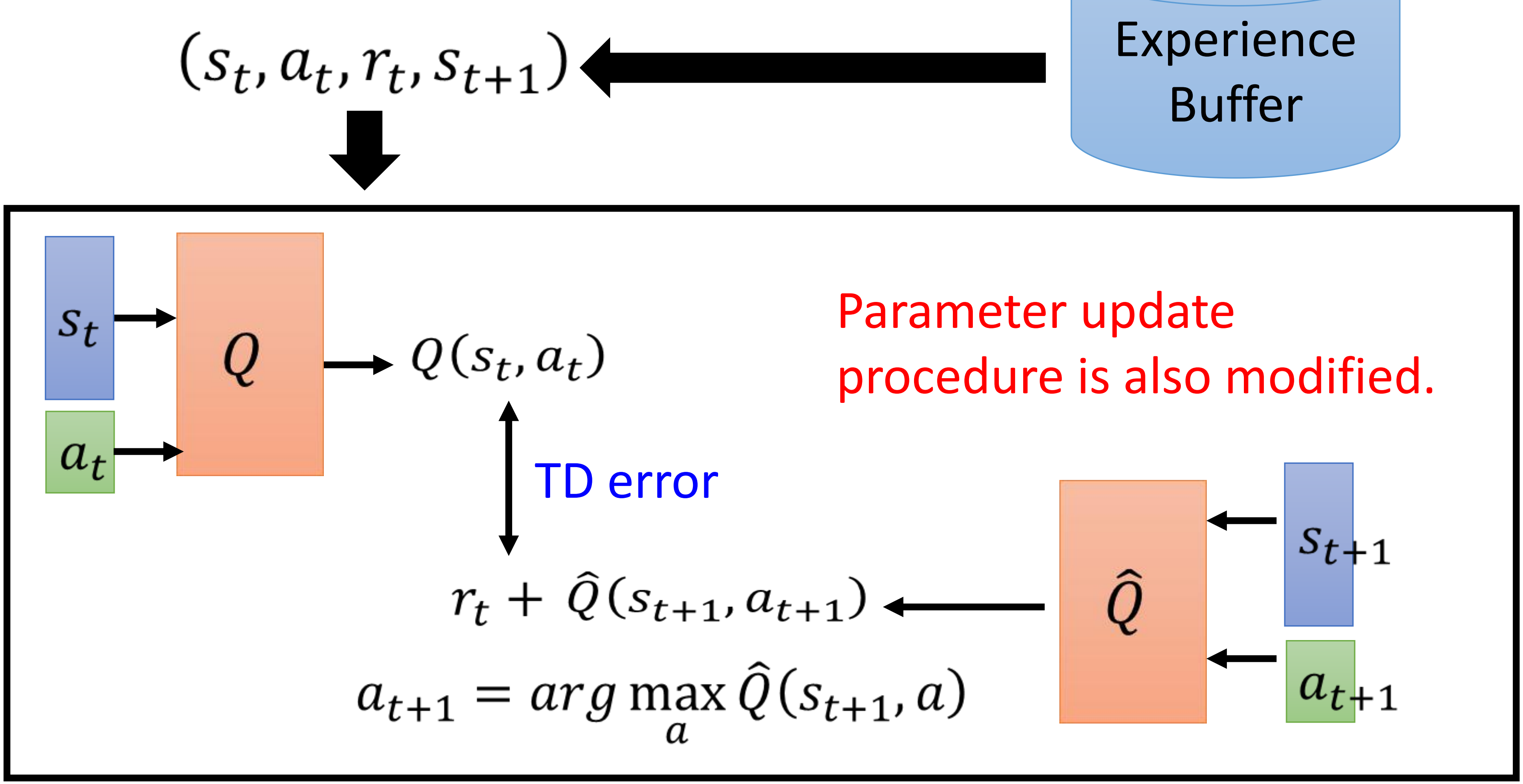
Dueling DQN - Visualization

(from the link of the original paper)

https://arxiv.org/abs/1511.05952?context=cs

Prioritized Reply

The data with larger TD error in previous training has higher probability to be sampled.



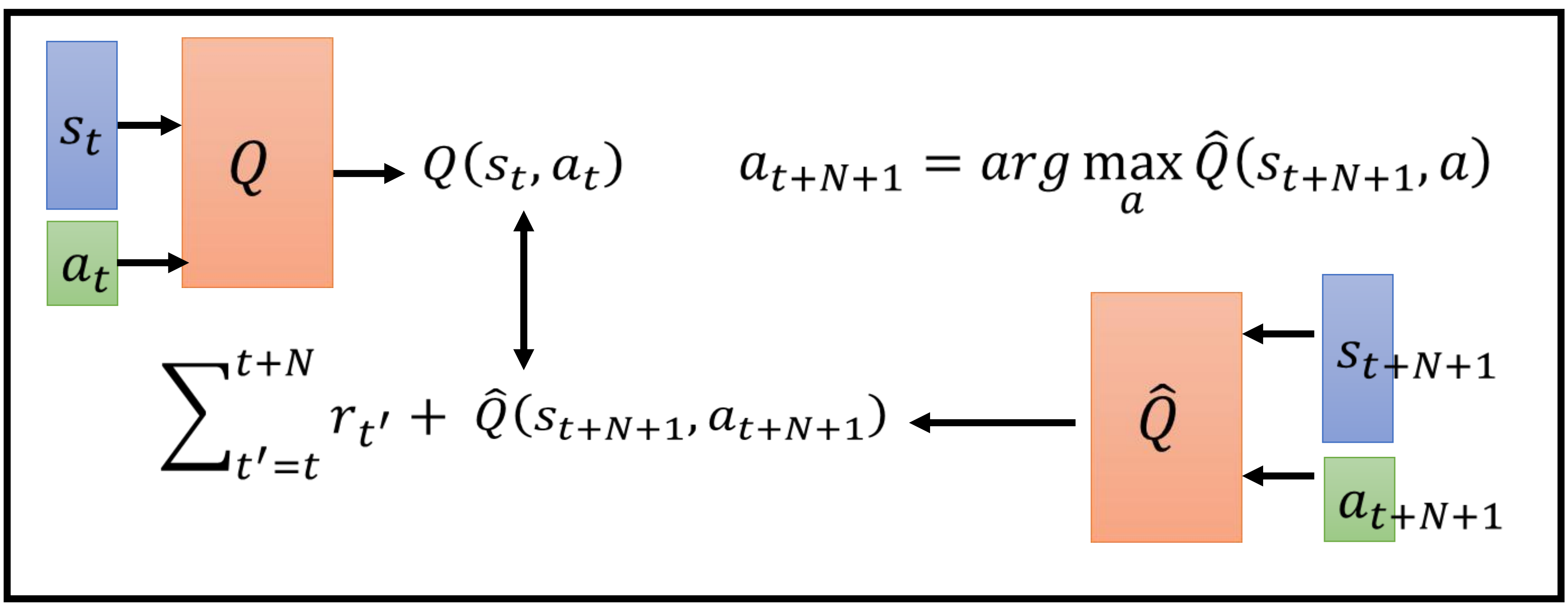
Multi-step

Balance between MC and TD

$$(s_t, a_t, r_t, \cdots, s_{t+N}, a_{t+N}, r_{t+N}, s_{t+N+1})$$

$$\cancel{(s_t, a_t, r_t, s_{t+1})}$$

Experience Buffer



Noisy Net

https://arxiv.org/abs/1706.01905
https://arxiv.org/abs/1706.10295

- Noise on Action (Epsilon Greedy)

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

- Noise on Parameters

Inject noise into the parameters of Q-function **at the beginning of each episode**

$$a = \arg\max_a \tilde{Q}(s,a)$$

$$Q(s,a) \xrightarrow{\text{Add noise}} \tilde{Q}(s,a)$$

The noise would **NOT** change in an episode.

Noisy Net

- Noise on Action
  - Given the same state, the agent may takes different actions.
  - No real policy works in this way

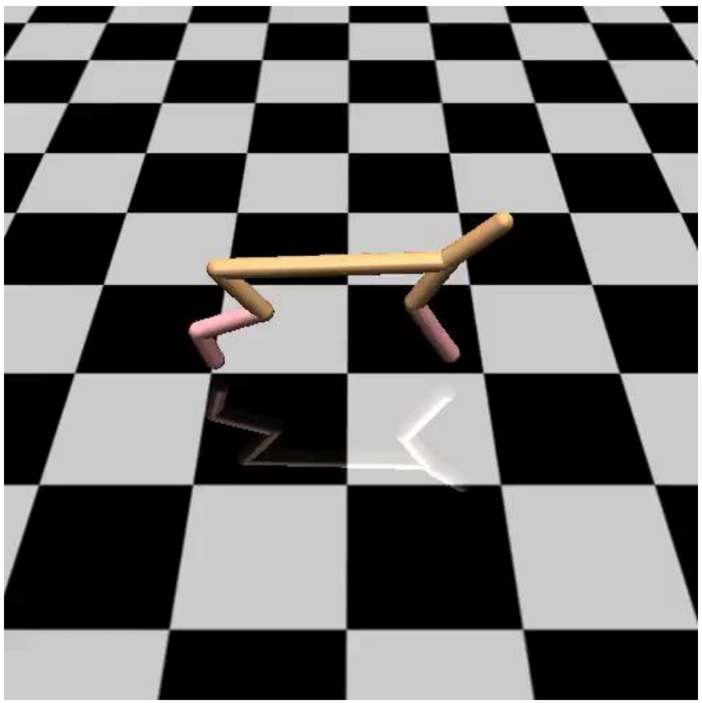
隨機亂試

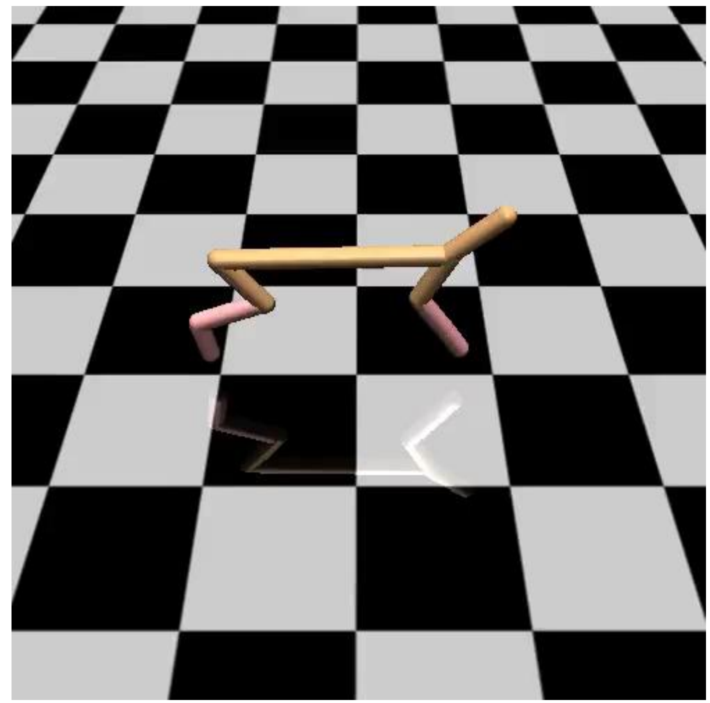
- Noise on Parameters
  - Given the same (similar) state, the agent takes the same action.
    - $\longrightarrow$ State-dependent Exploration
  - Explore in a *consistent* way

有系統地試

Demo

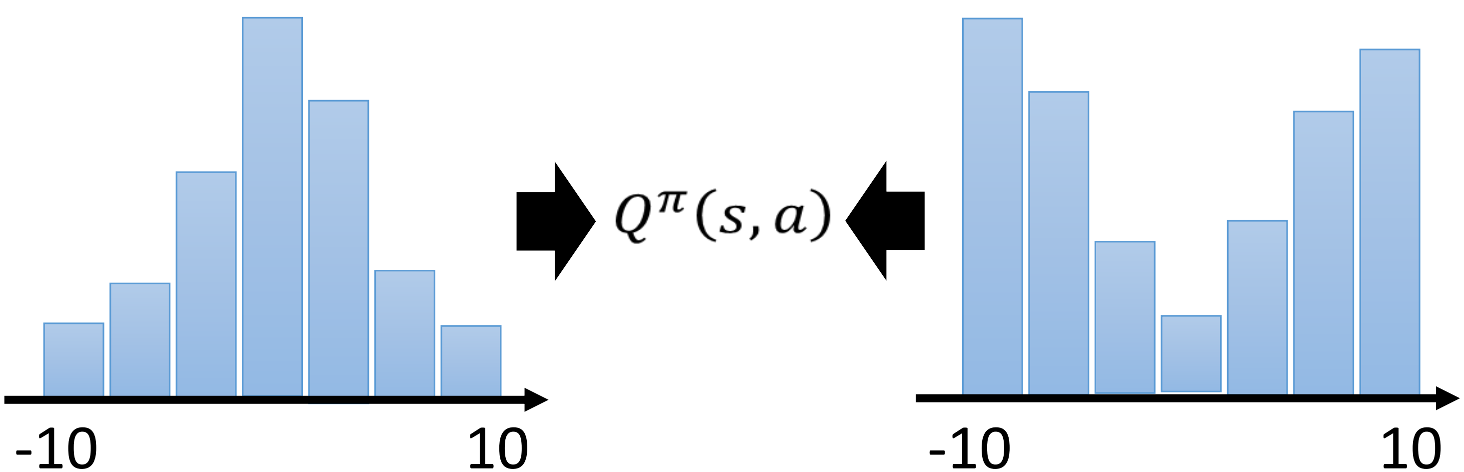
https://blog.openai.com/better-exploration-with-parameter-noise/





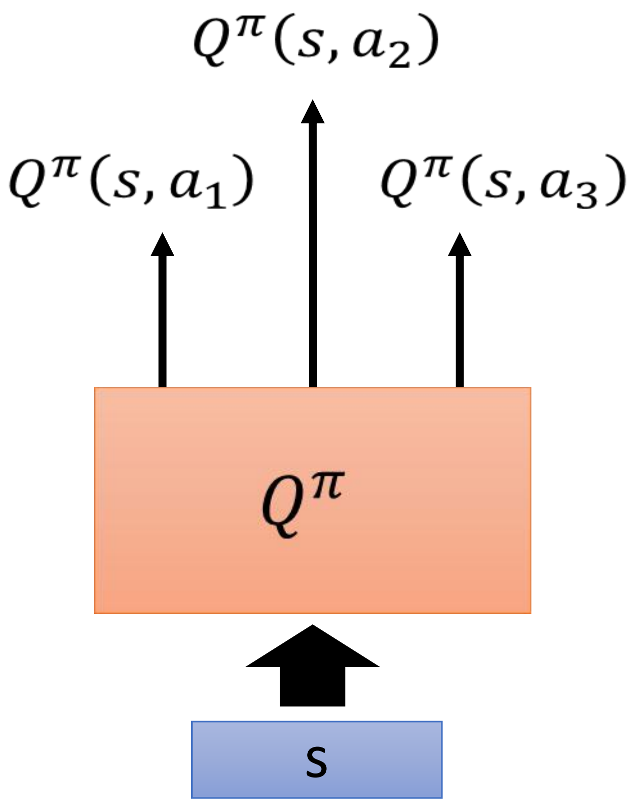
Distributional Q-function

- 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

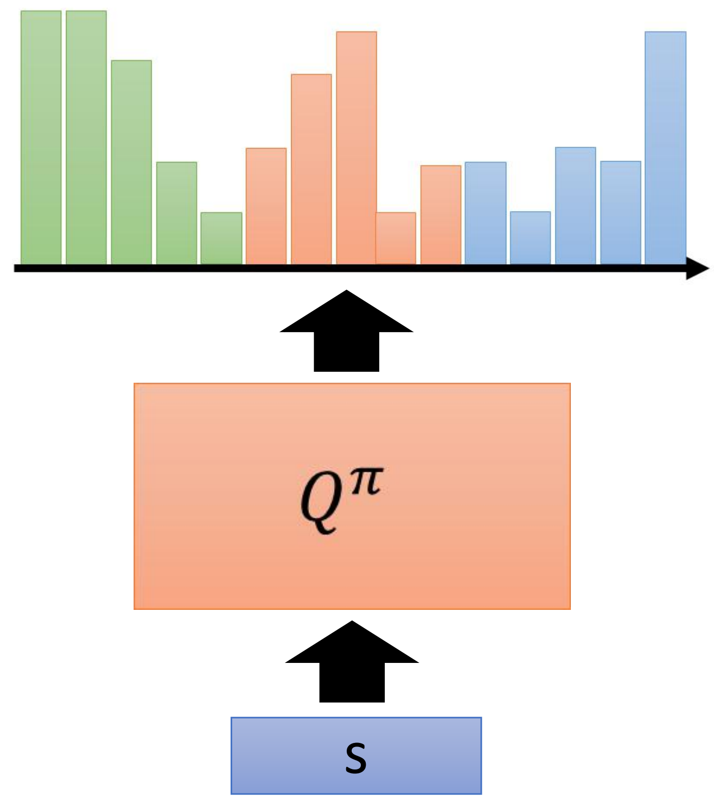


Different distributions can have the same values.

Distributional Q-function

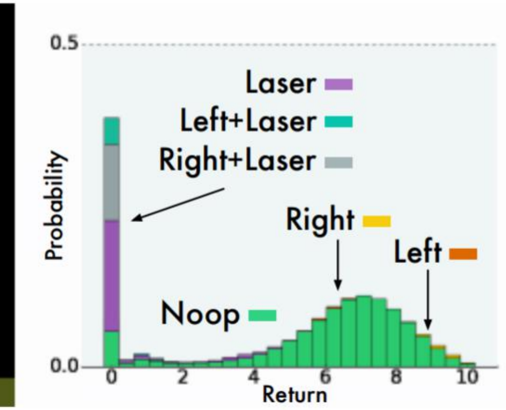


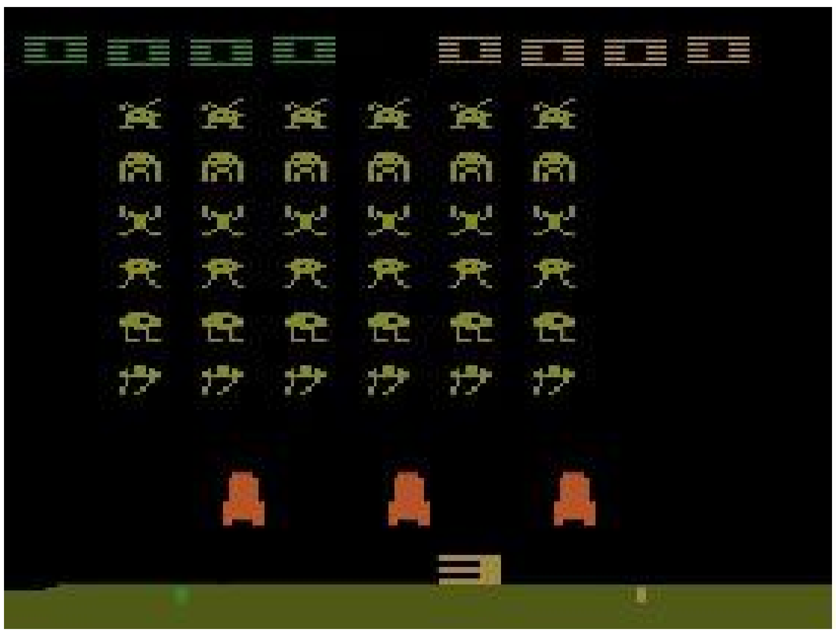
A network with 3 outputs

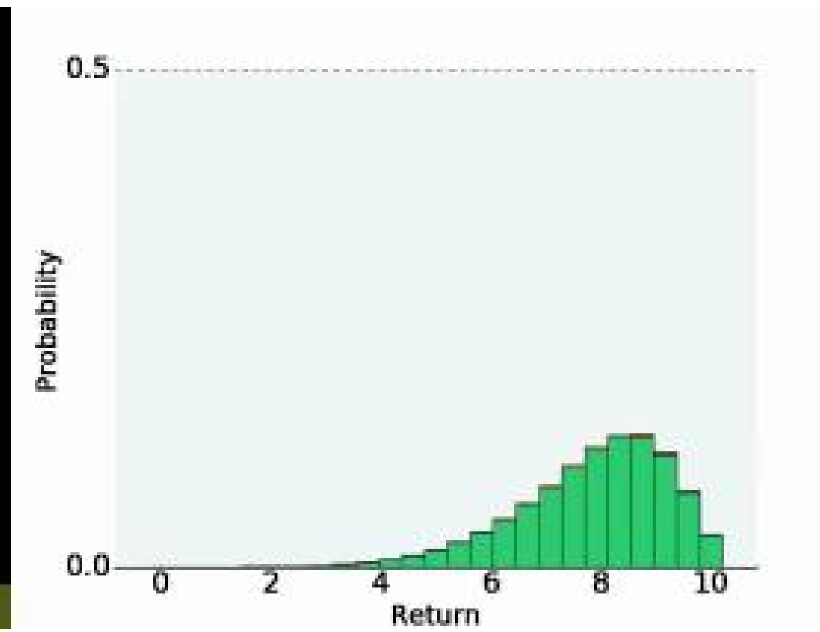


A network with 15 outputs
(each action has 5 bins)

Demo

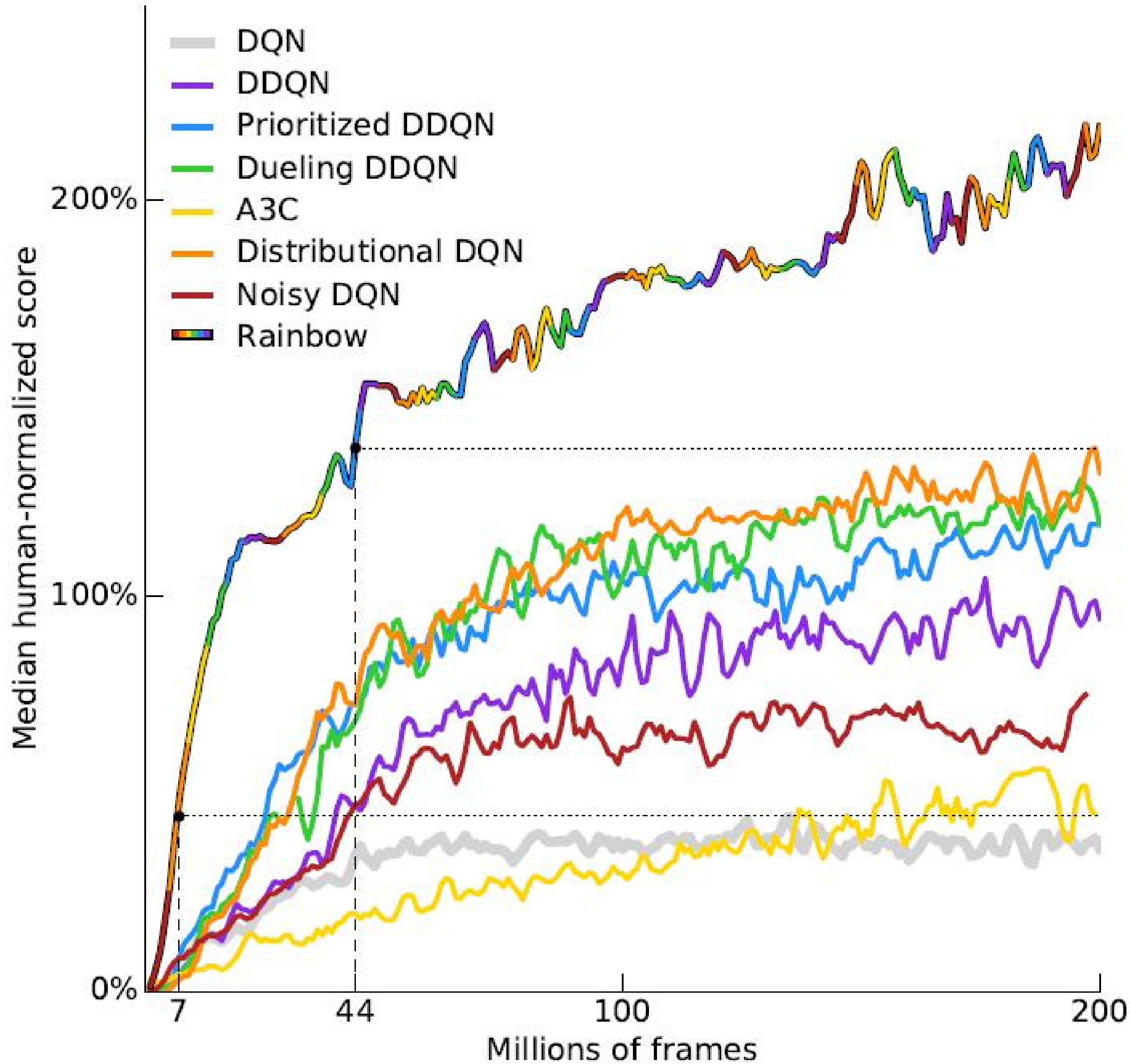






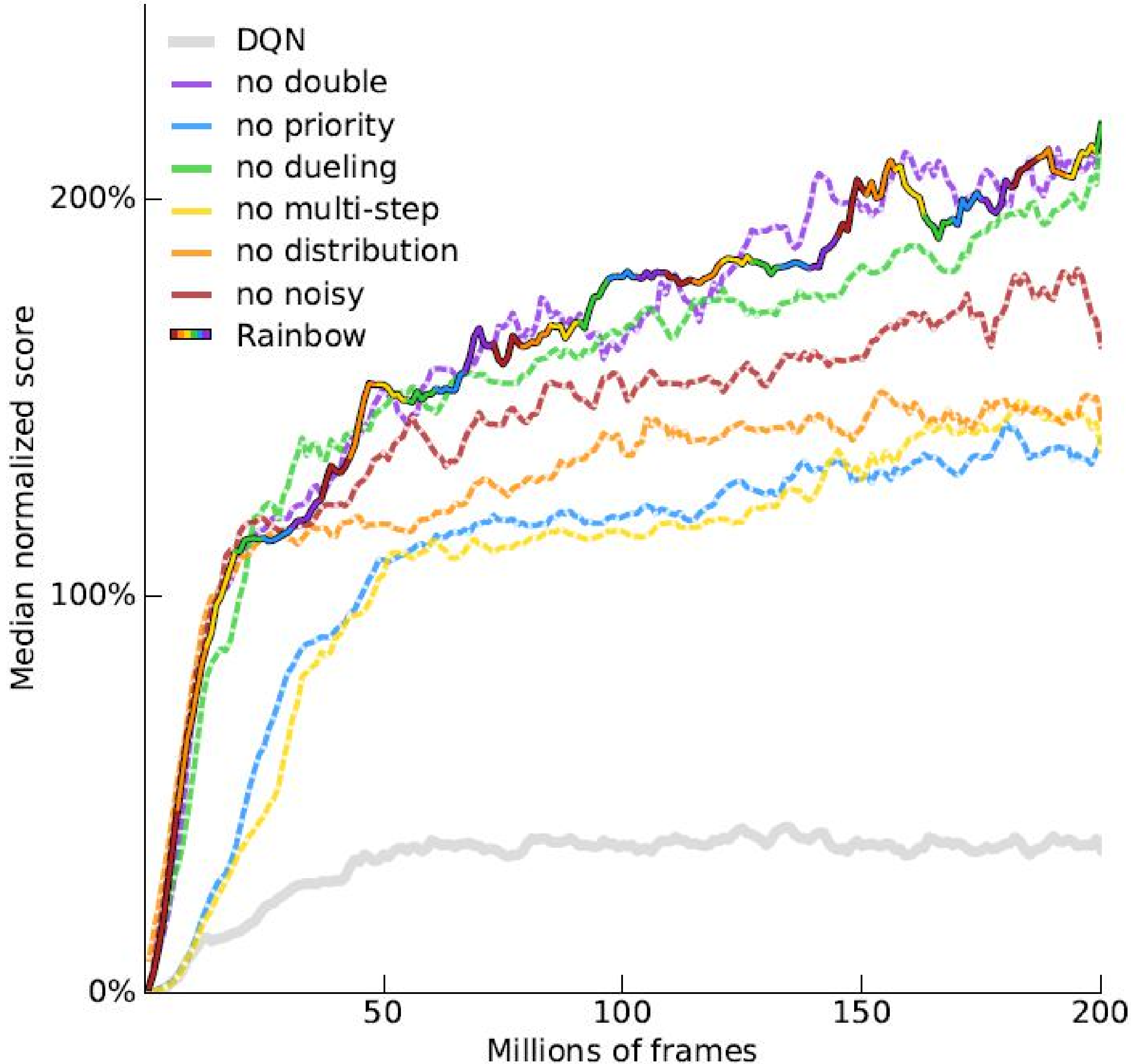
https://youtu.be/yFBwyPuO2Vg

Rainbow



https://arxiv.org/abs/1710.02298

Rainbow



https://arxiv.org/abs/1710.02298

Outline

Introduction of Q-Learning

Tips of Q-Learning

Q-Learning for Continuous Actions

Continuous Actions

- Action a is a *continuous vector*

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

Solution 1

Sample a set of actions: $\{a_1, a_2, \cdots, a_N\}$

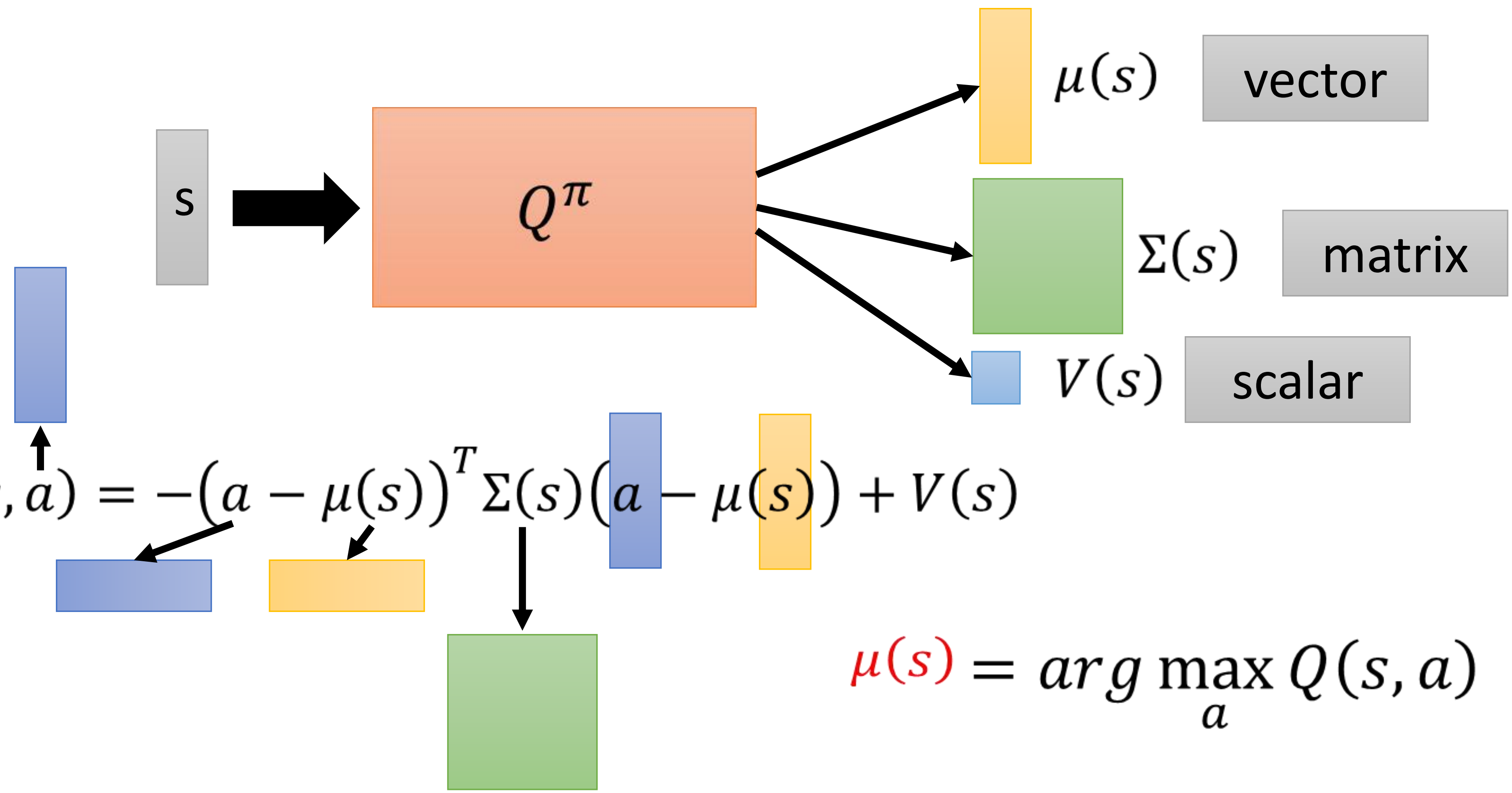
See which action can obtain the largest Q value

Solution 2

Using gradient ascent to solve the optimization problem.

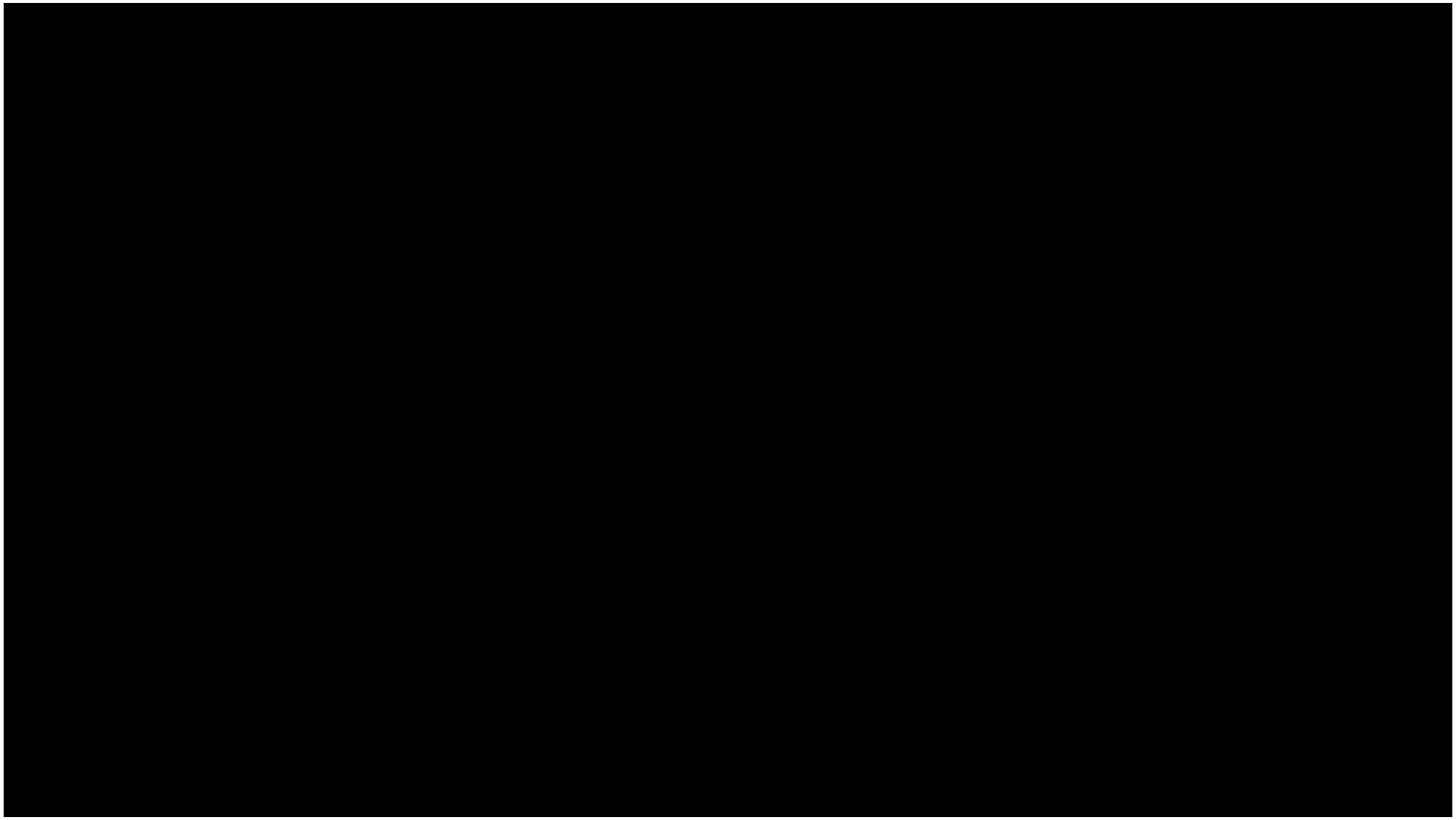
Continuous Actions

Solution 3 Design a network to make the optimization easy.



$$Q(s,a) = -\left(a - \mu(s)\right)^T \Sigma(s) \left(a - \mu(s)\right) + V(s)$$

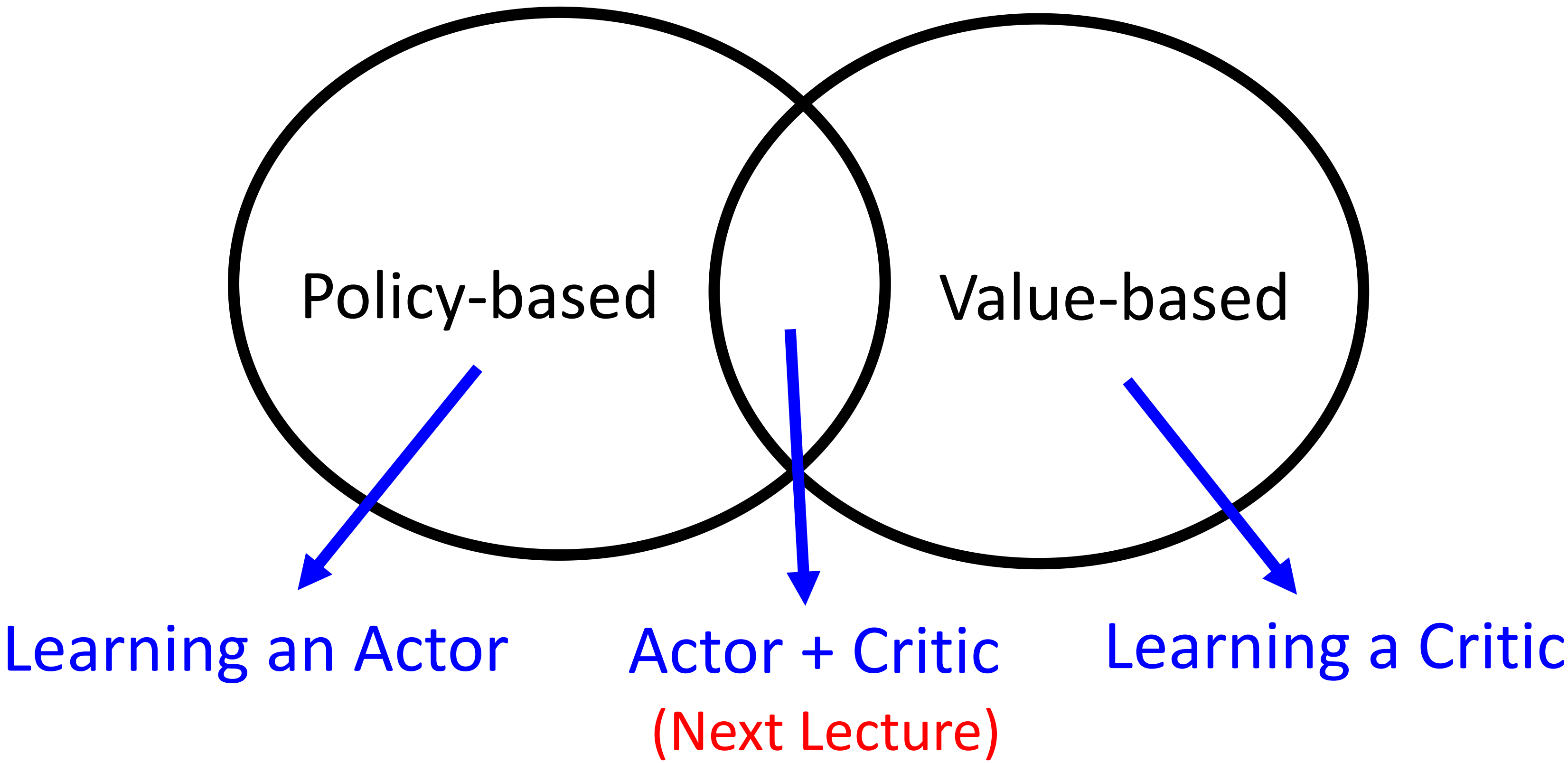
$$\mu(s) = arg \max_a Q(s,a)$$



https://www.youtube.com/watch?v=ZhsEKTo7V04

Continuous Actions

Solution 4 Don’t use Q-learning



Acknowledgement

- 感謝林雨新同學發現投影片 上的錯字