Q-Learning

Hung-yi Lee

Outline

Introduction of Q-Learning

Tips of Q-Learning

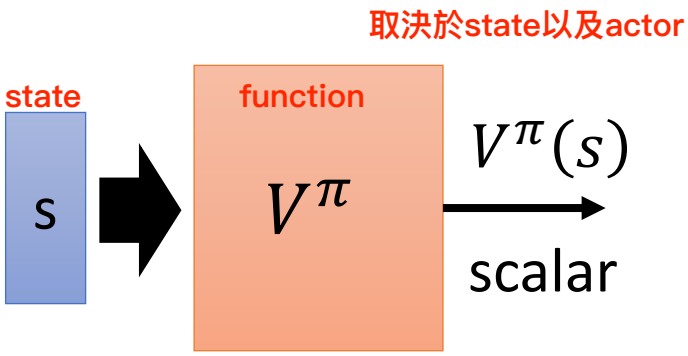
Q-Learning for Continuous Actions

Critic

critic只是來評價這個actor好不好

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



output從現在開始到結束所得到可能的reward

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

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

Critic

state

V以前的阿光(大馬步飛) = bad

actor

V變強的阿光(大馬步飛) = good

左為：critic

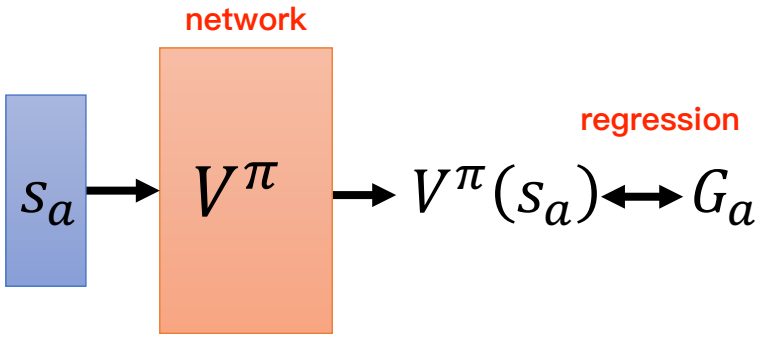
How to estimate $V^{\pi}(s)$

每次都要算cumulative reward，每次要估算就要玩完整場遊戲

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

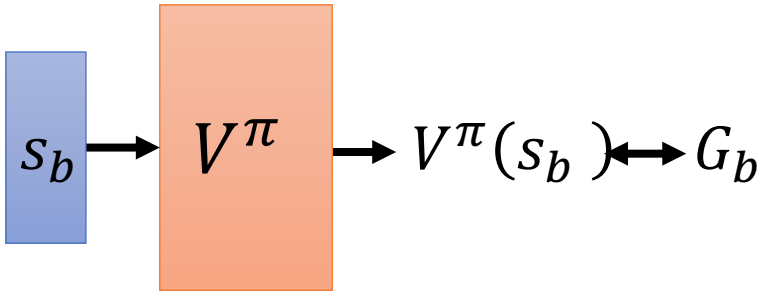
After seeing s_a,

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



After seeing s_b,

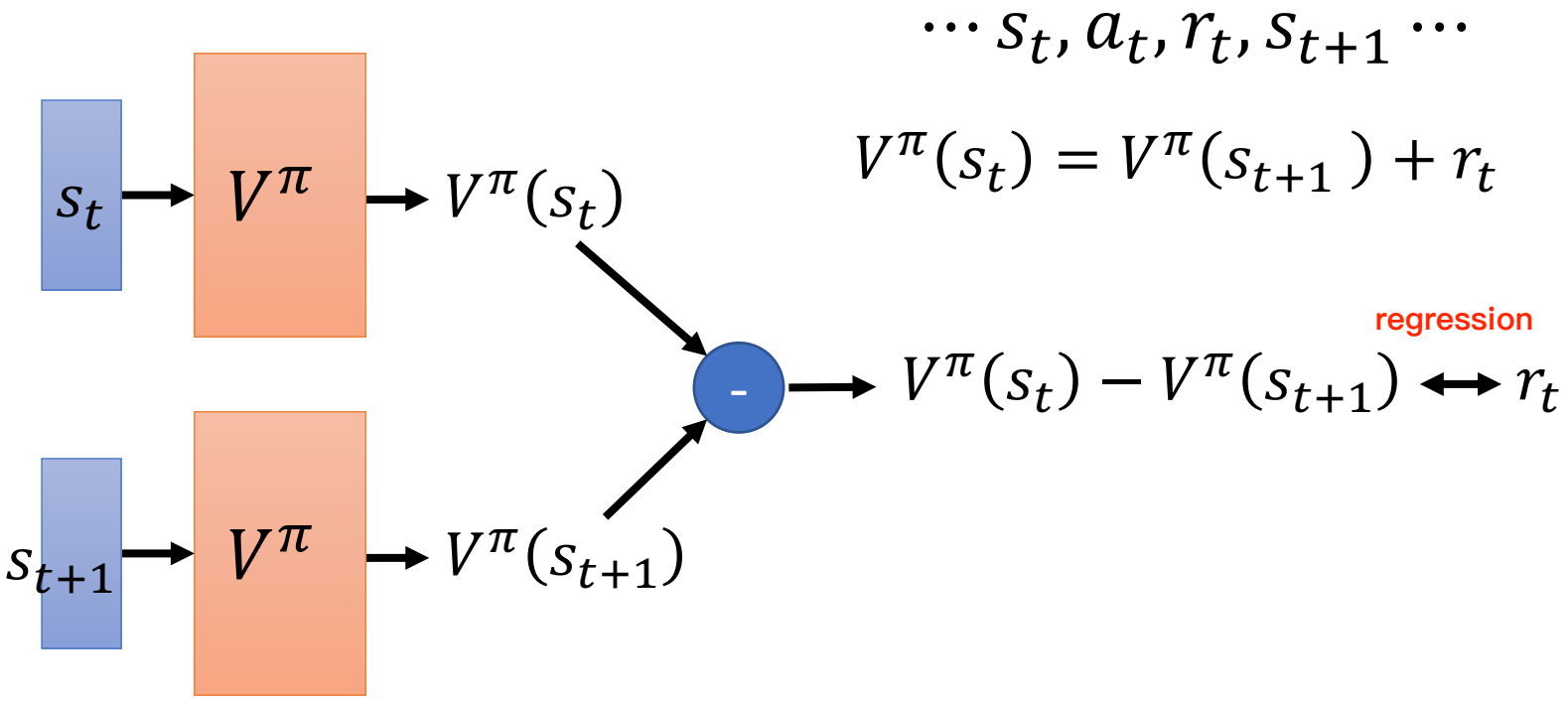
Until the end of the episode, the cumulated reward is G_b



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

不需要把遊戲玩到底就可以update model

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

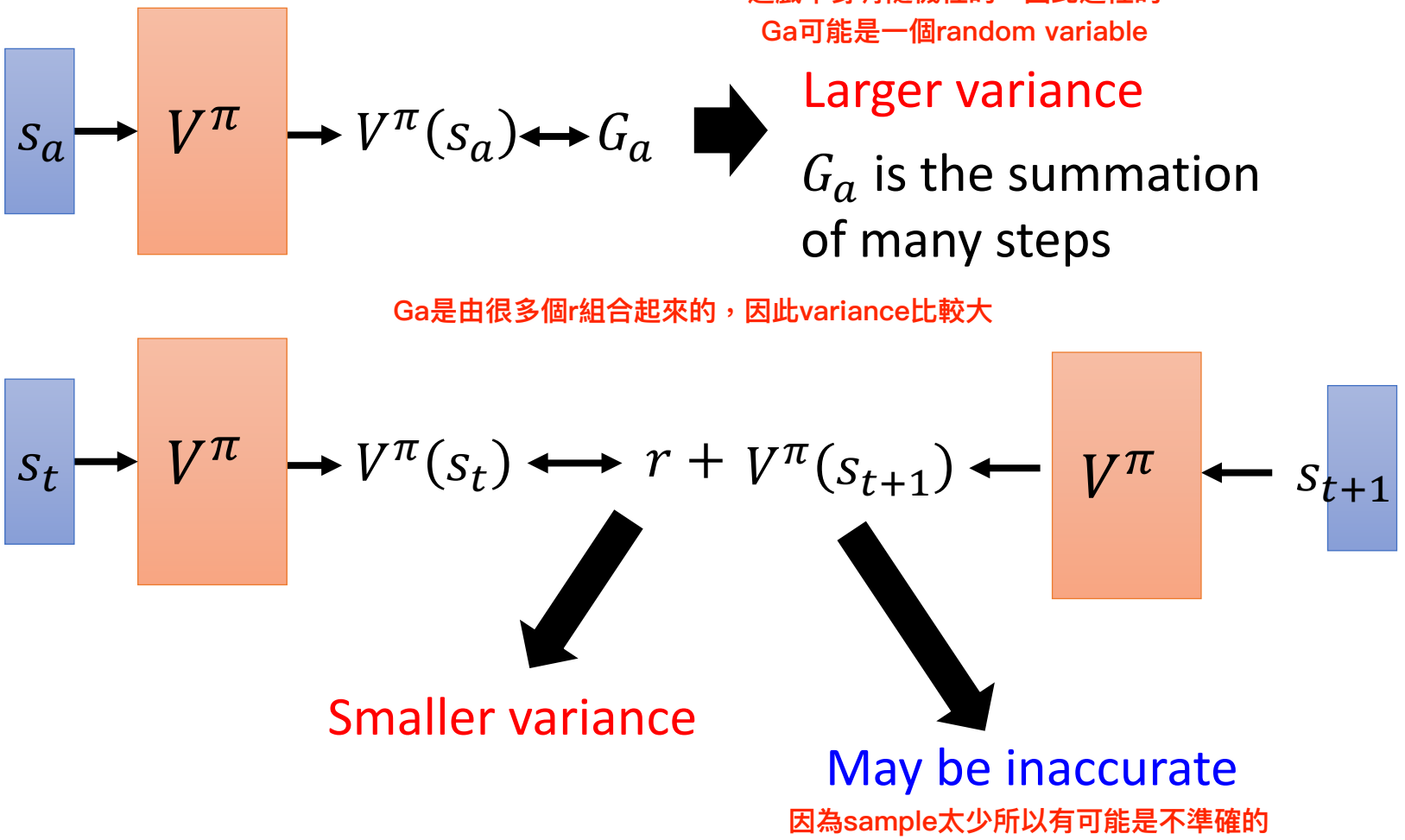


Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.

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

TD比較常用

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? (MC) 3/4? (TD)

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

Sa影響到Sb

Temporal-difference:

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

3/4 3/4 0

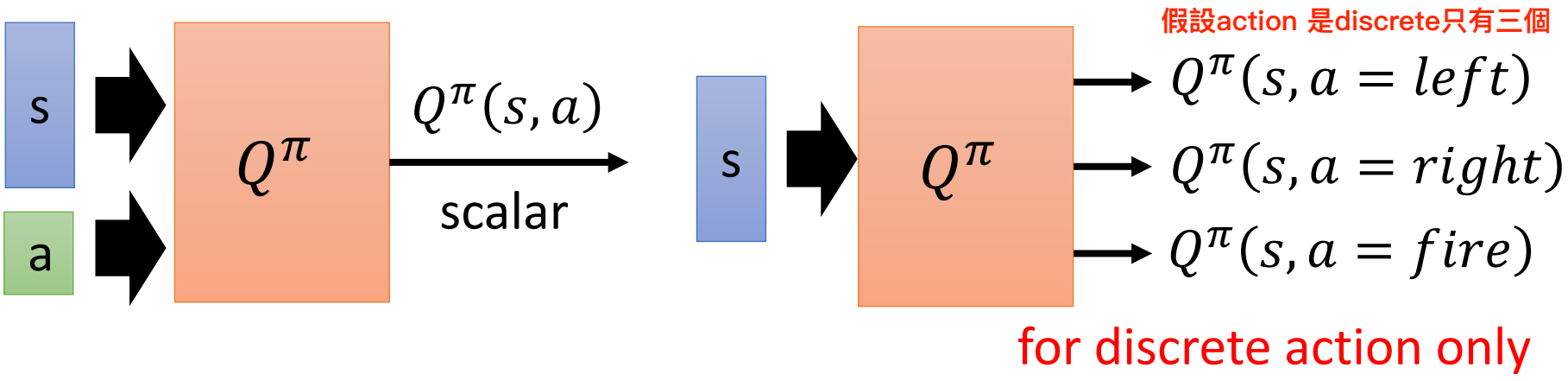
(The actions are ignored here.)

Sa沒有影響到Sb，r=0的trajectory只是巧合

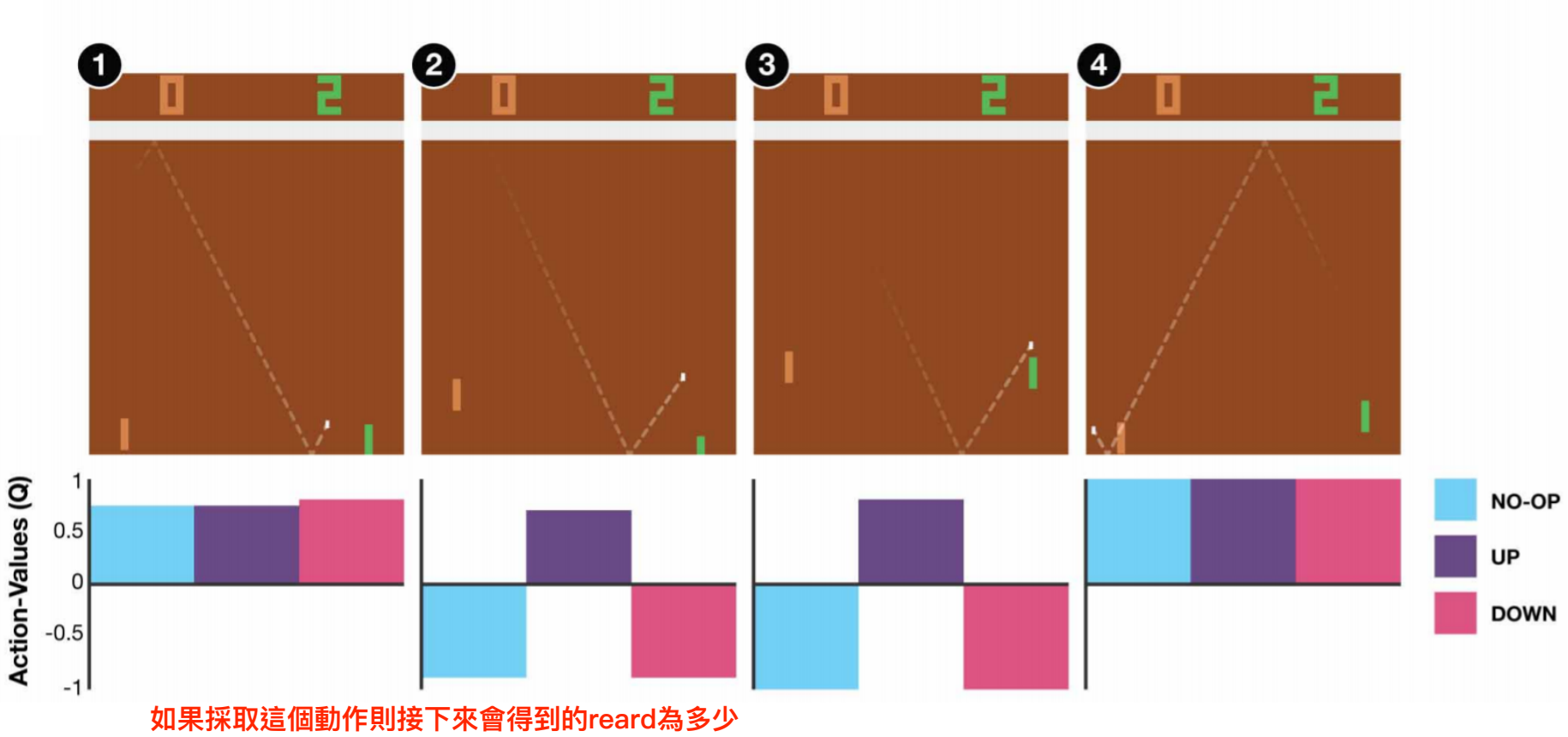
Another Critic

Q：假設在state s 強制採取action a，接下來都用actor pi繼續玩下去

- 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



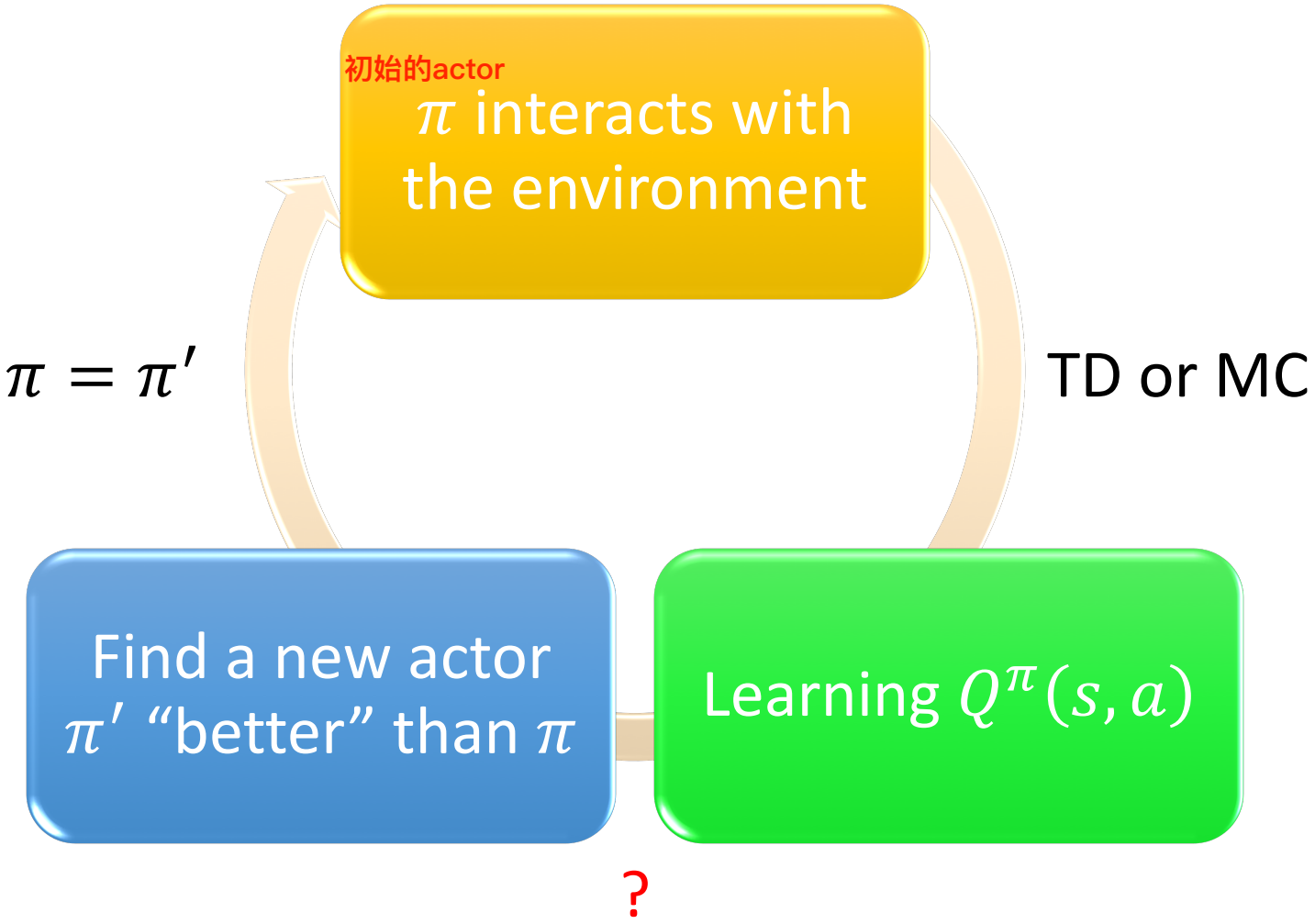
State-action value function



如果採取這個動作則接下來會得到的reard為多少

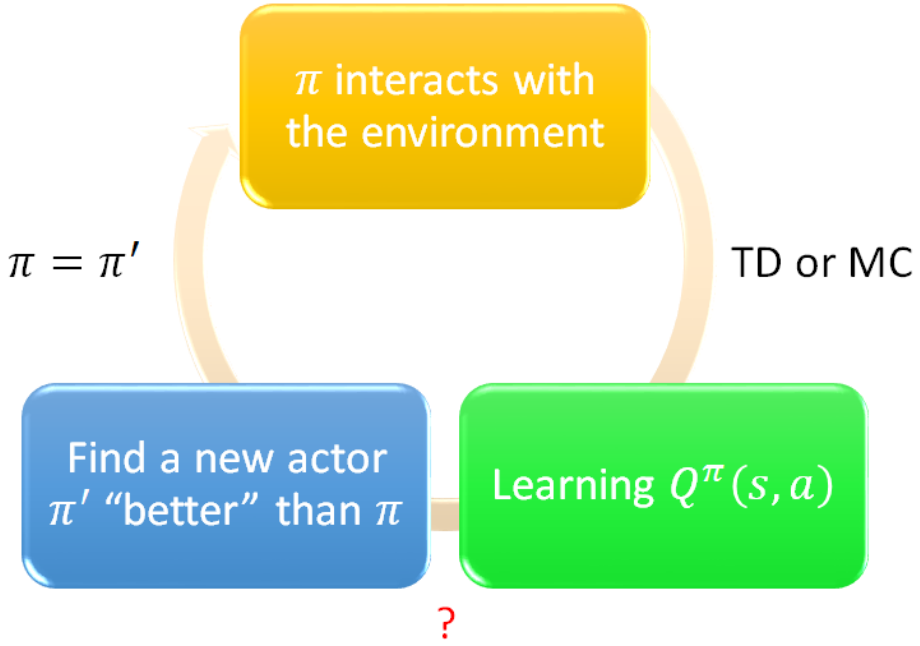
https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf

Another Way to use Critic: Q-Learning



只要擁有一個policy，可以找出Q function，找出Q function後一定可以找到新的一個policy

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

對所有可能的state s 而言，pi的value function一定會小於pi'

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

假設已經learn出pi的Q function，則可以透過Q計算哪個action對這個state s可以有最高的 value

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

如果a是continuous的話，在解argmax的時候會有問題

但如果是discrete的話則沒有問題

Q-Learning

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

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

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

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

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

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

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

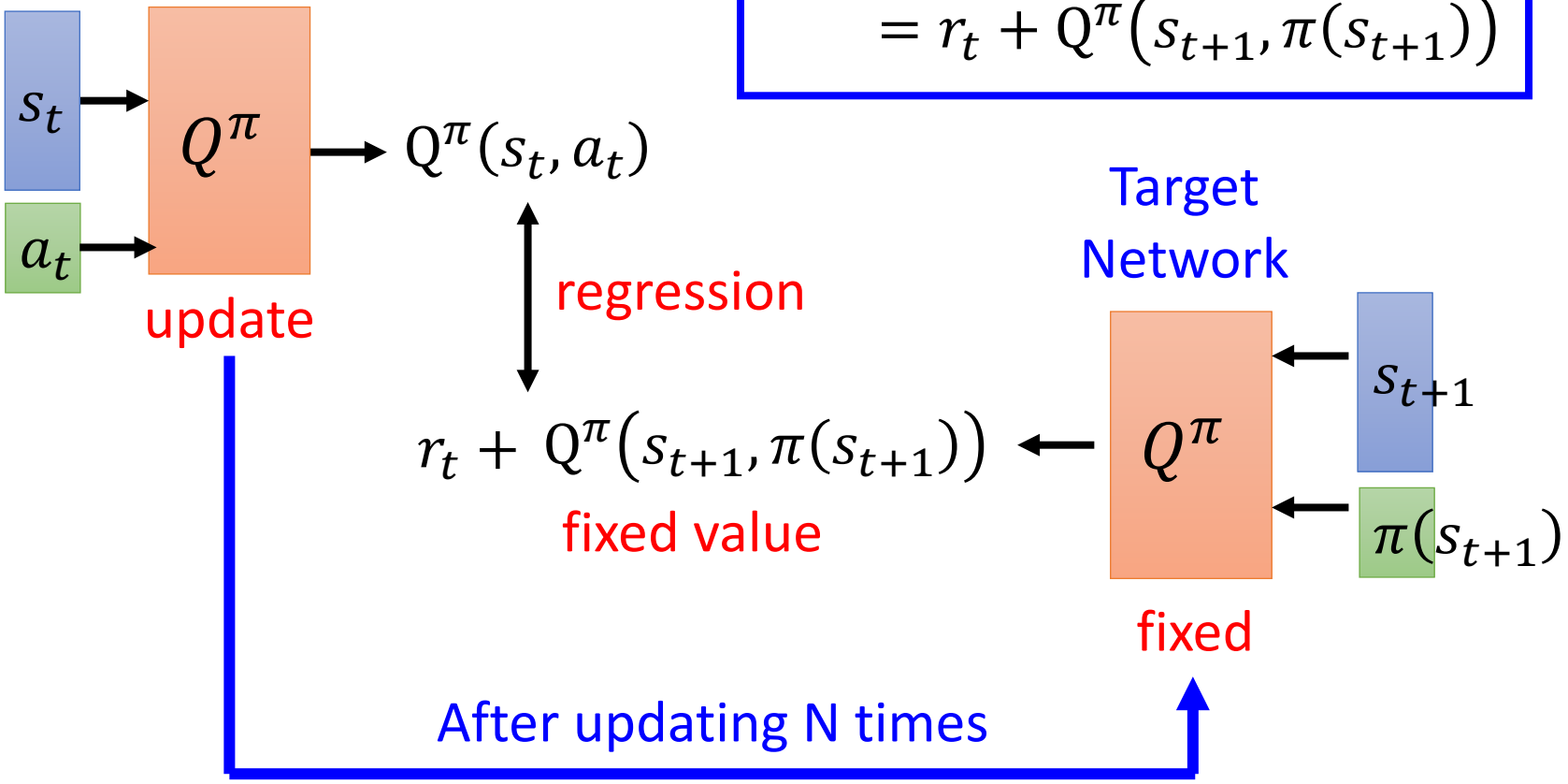
$$= E[r_{t+1} + r_{t+2} + V^{\pi}(s_{t+2})|\dots]$$

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

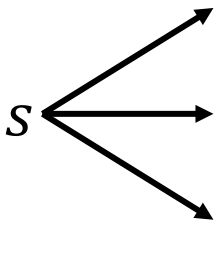
Target Network

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

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



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.

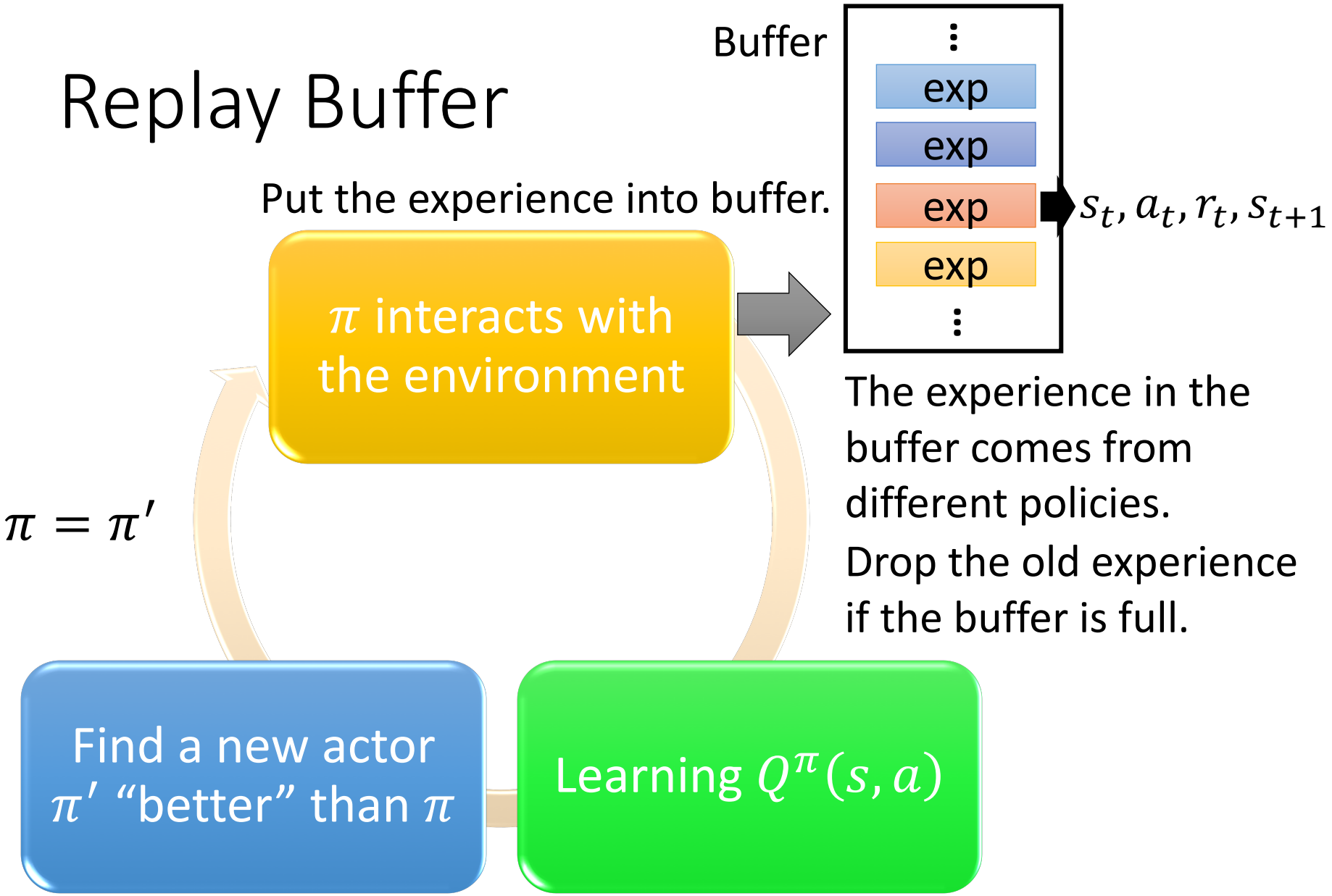
Epsilon Greedy ε would decay during learning

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

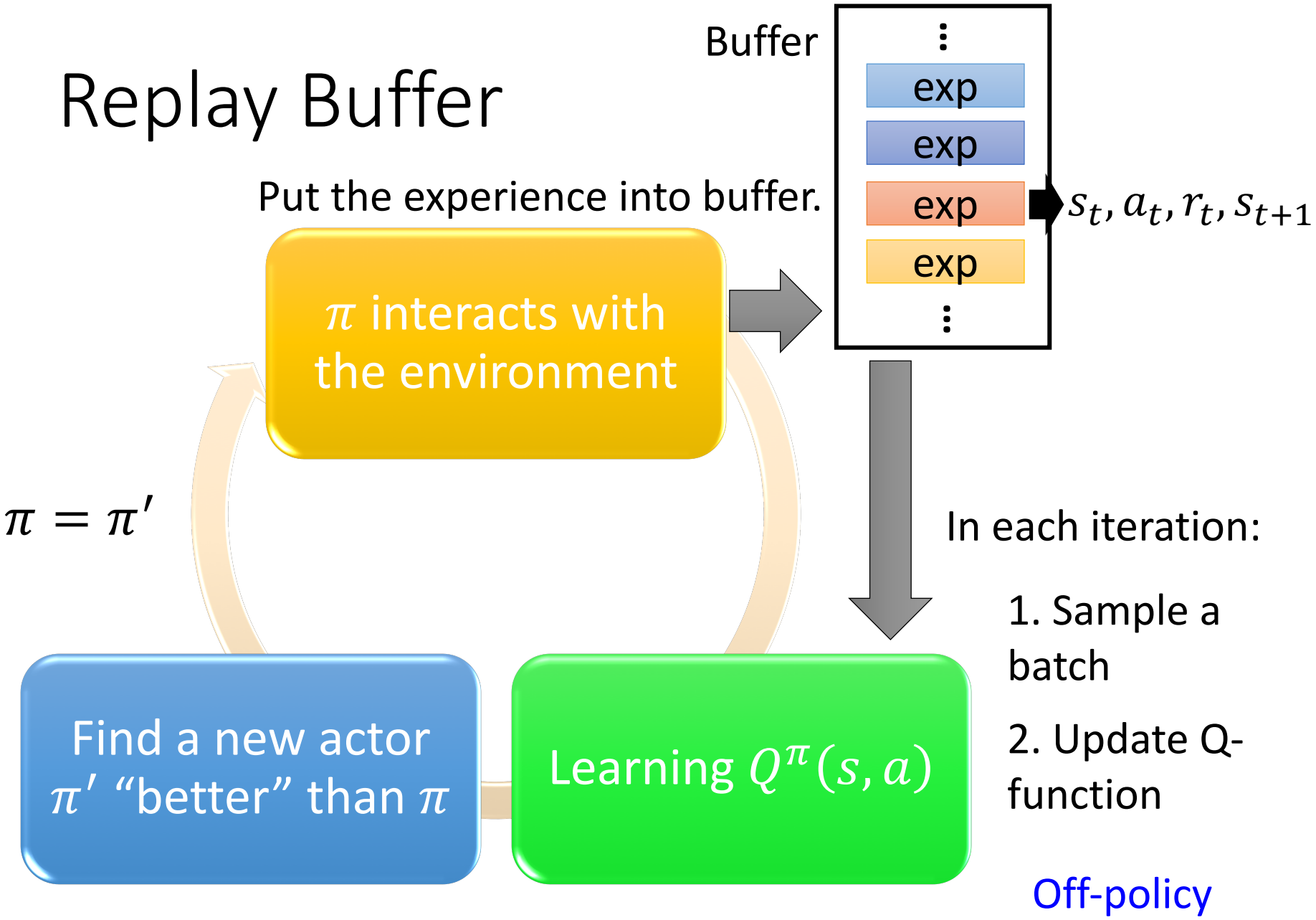
Boltzmann Exploration

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

Replay Buffer



Replay Buffer



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 butter
    - Sample (s_i, a_i, r_i, s_{i+1}) from butter (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

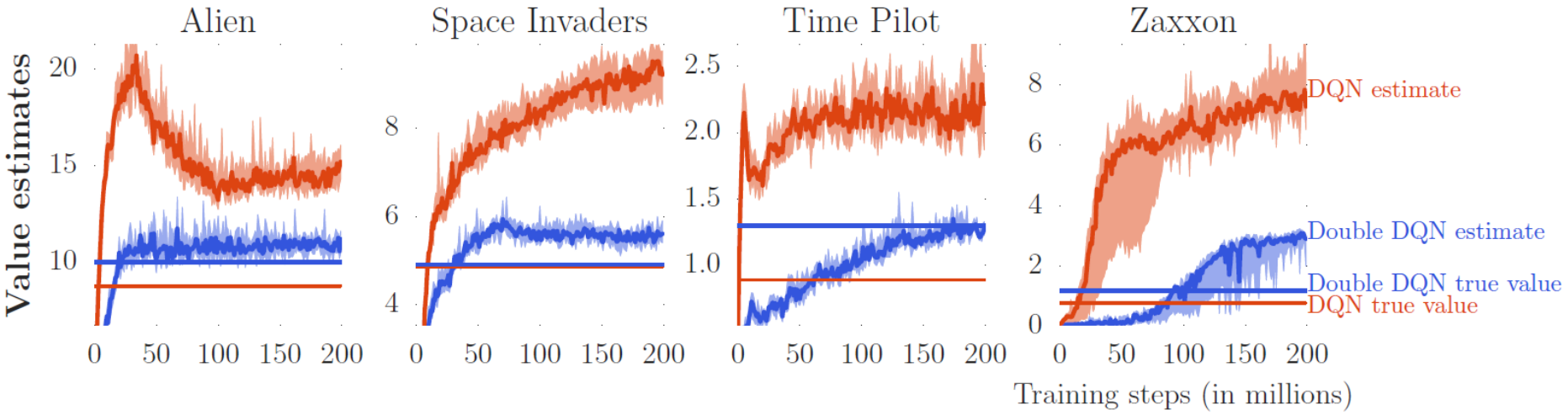
Introduction of Q-Learning

Tips of Q-Learning

Q-Learning for Continuous Actions

Double DQN

- Q value is usually over-estimated



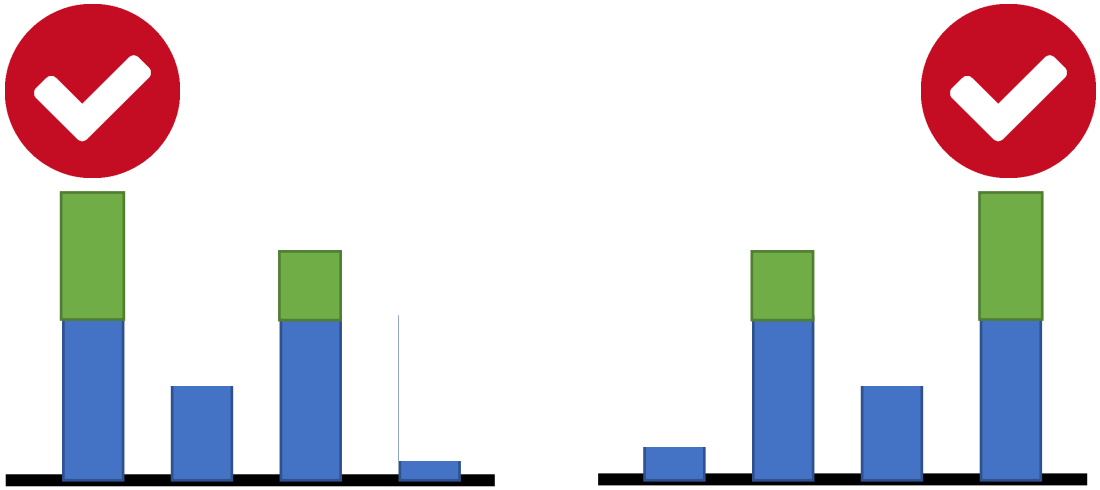
Double DQN

- Q value is usually over estimate

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

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

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

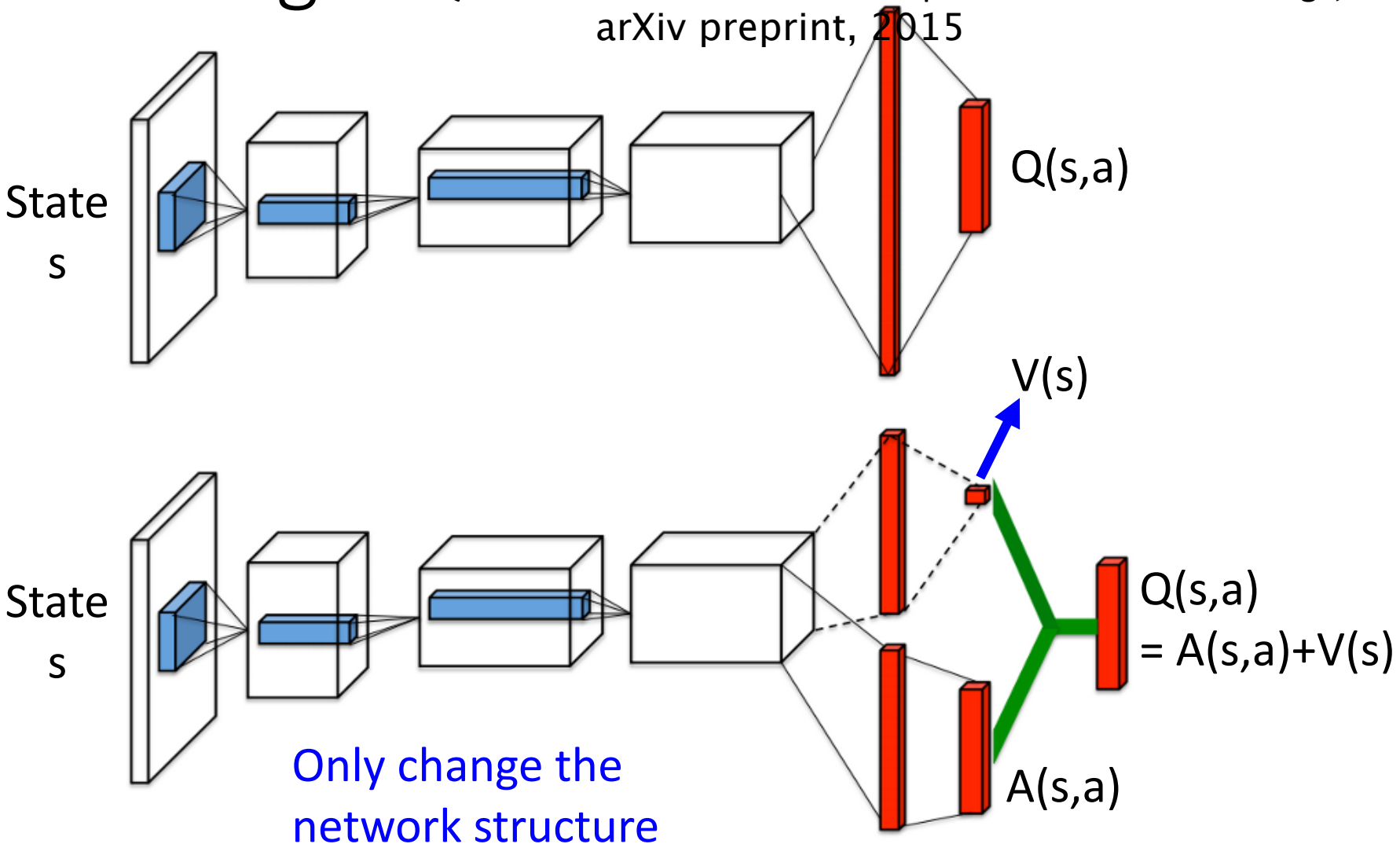
If Q over-estimate a, so it is selected. Q' would give it proper value.

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

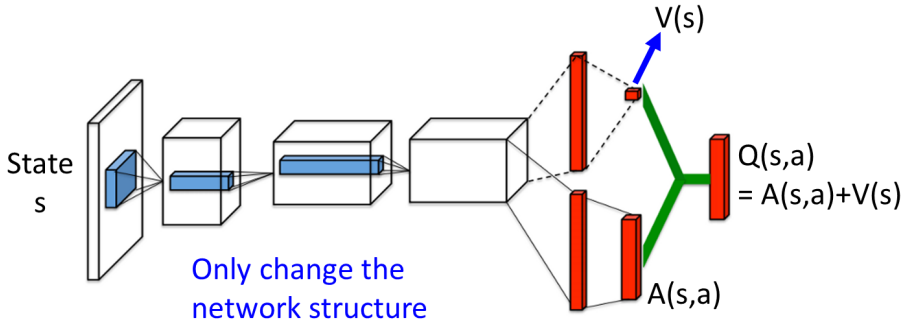
Hado V. Hasselt, "Double Q-learning", NIPS 2010
Hado van Hasselt, Arthur Guez, David Silver, "Deep Reinforcement Learning with Double Q-learning", AAAI 2016

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



Dueling DQN



Q(s,a) = V(s) + A(s,a)

action

state

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

=

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

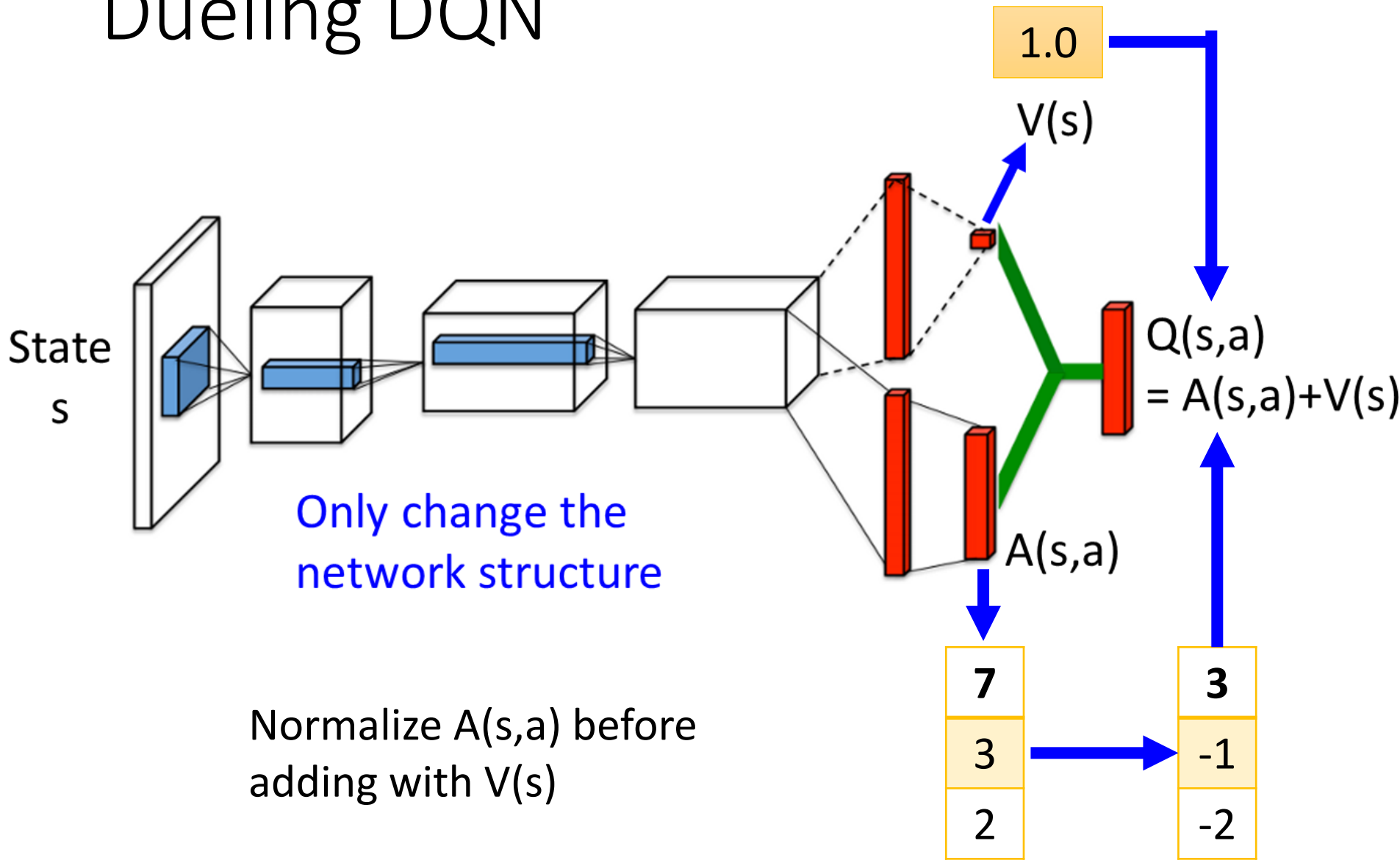
+

Average of column

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

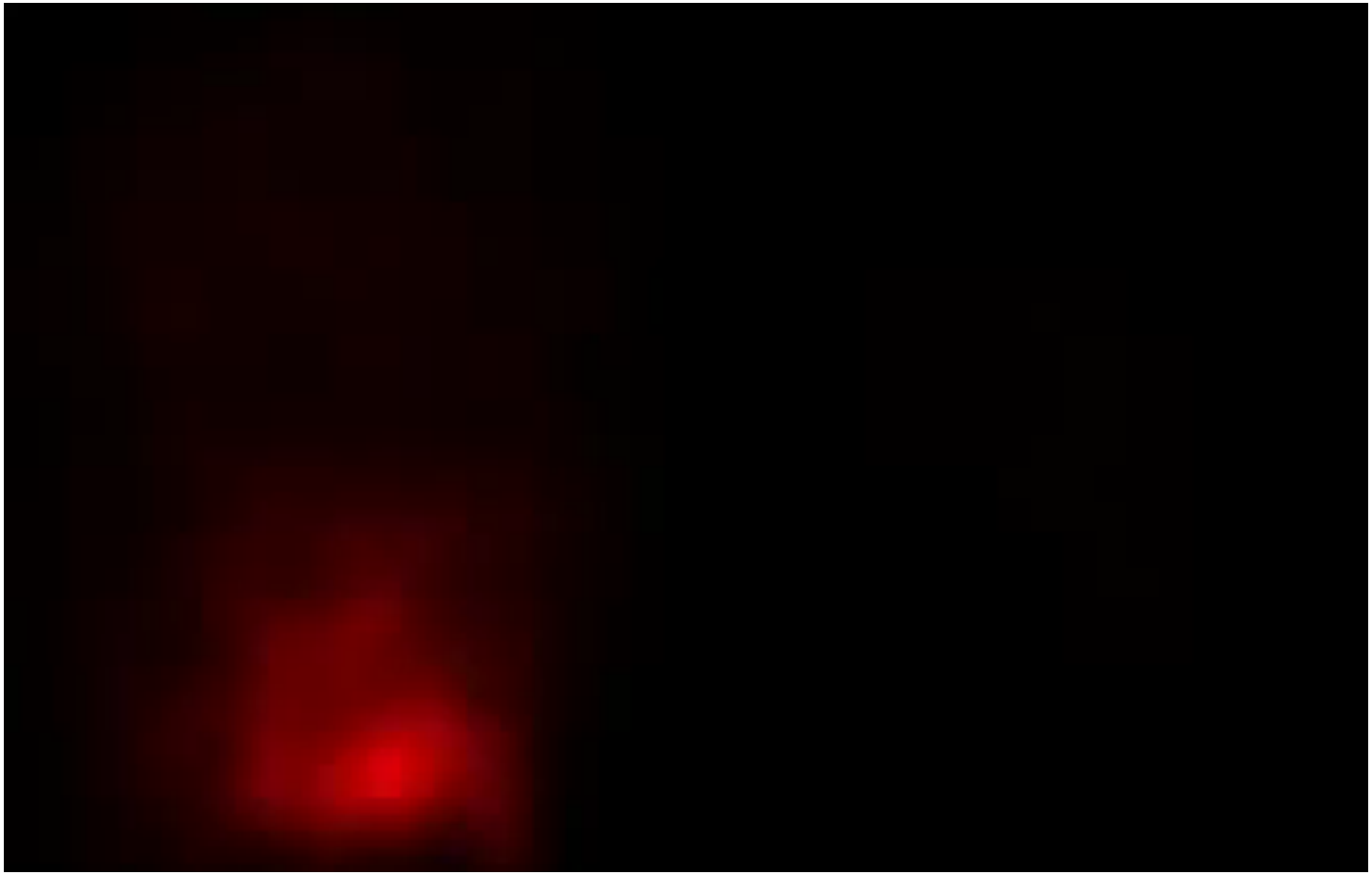
sum of column = 0

Dueling DQN



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

Dueling DQN - Visualization



(from the link of the original paper)

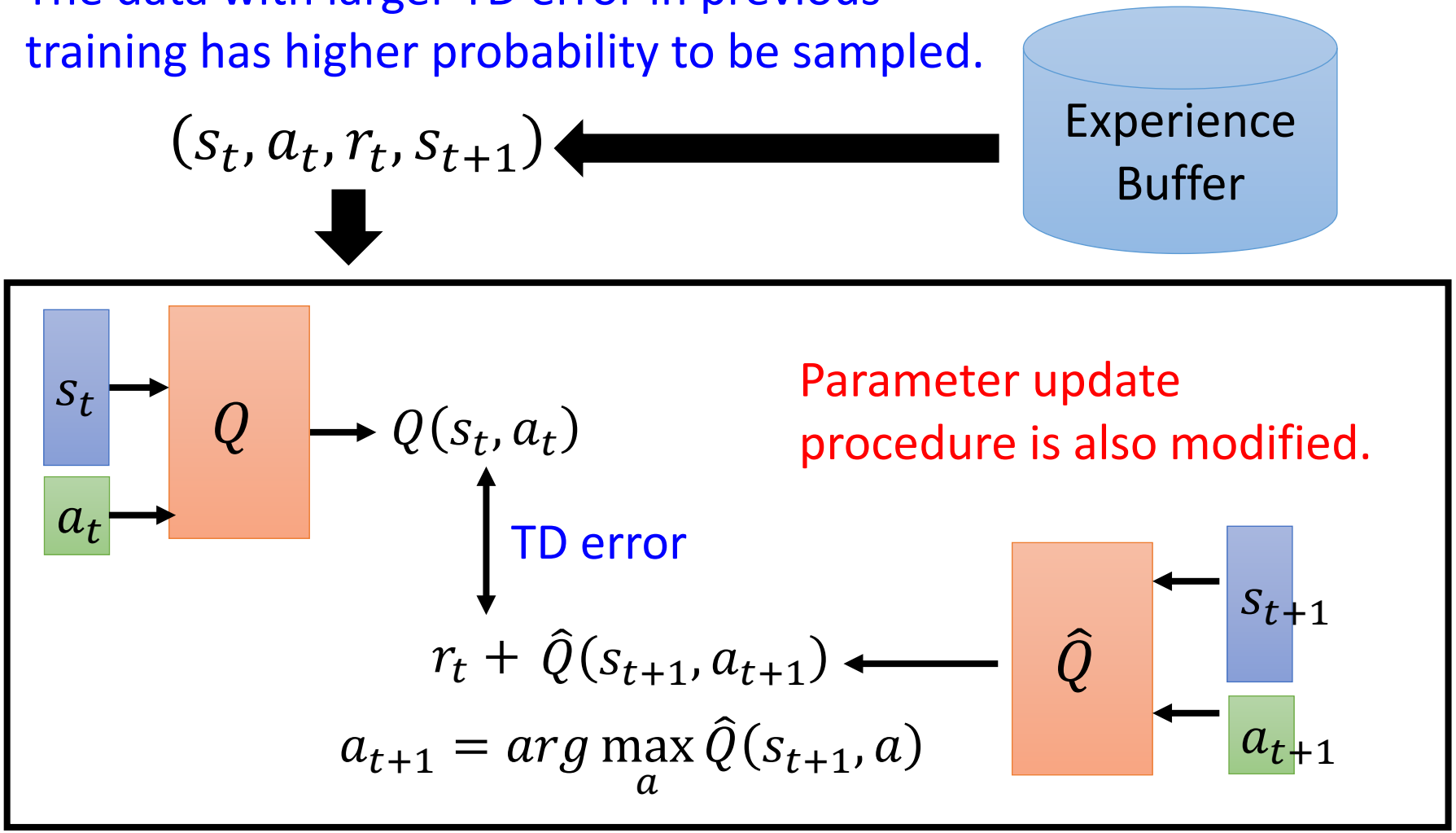
Dueling DQN - Visualization

(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.

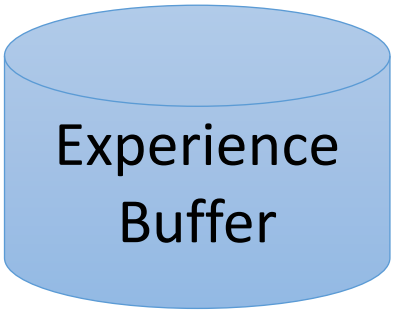


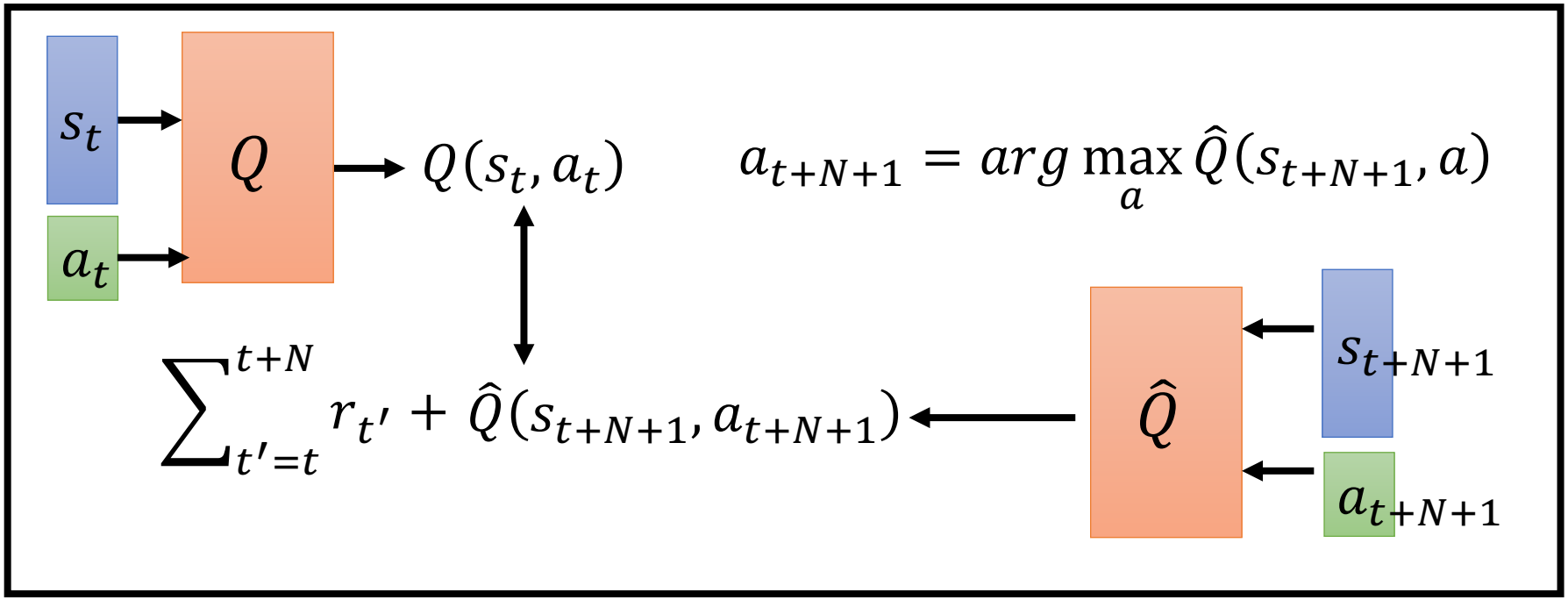
Multi-step

Balance between MC and TD

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

~~(s_t, a_t, r_t, s_{t+1})~~





Noisy Net

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

- Noise on Action (Epsilon Greedy)

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

- Noise on Parameters

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

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

$Q(s,a)$ → $\tilde{Q}(s,a)$

Add noise

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

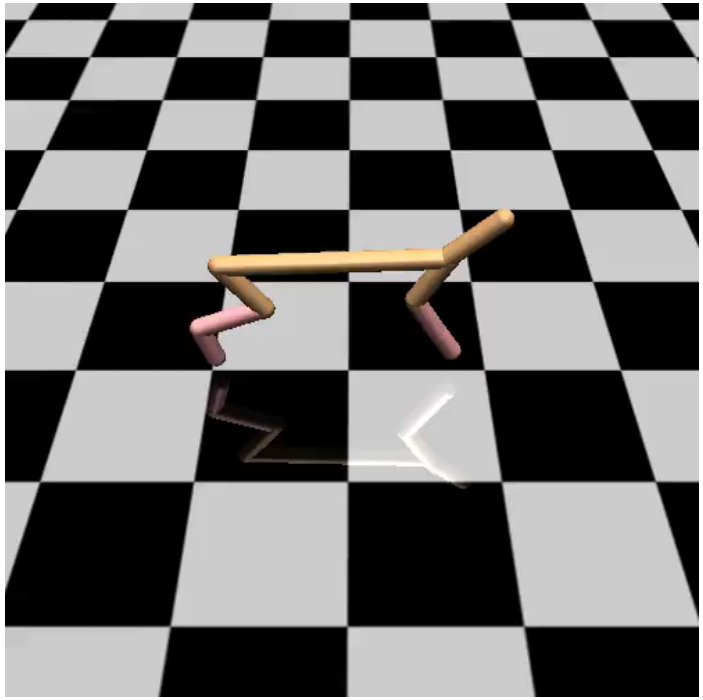
隨機亂試

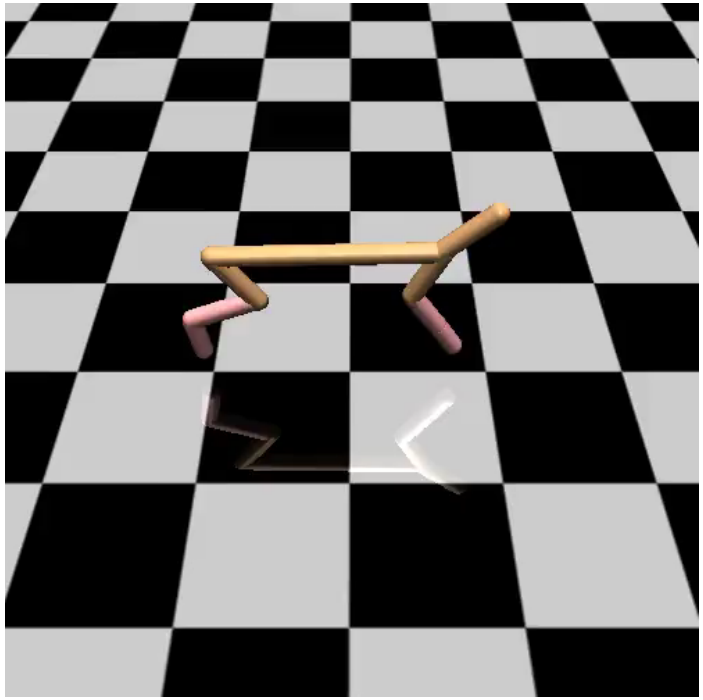
- Noise on Parameters
  - Given the same (similar) state, the agent takes the same action.
    - ⟶ State-dependent Exploration
  - Explore in a *consistent* way

有系統地試

Demo

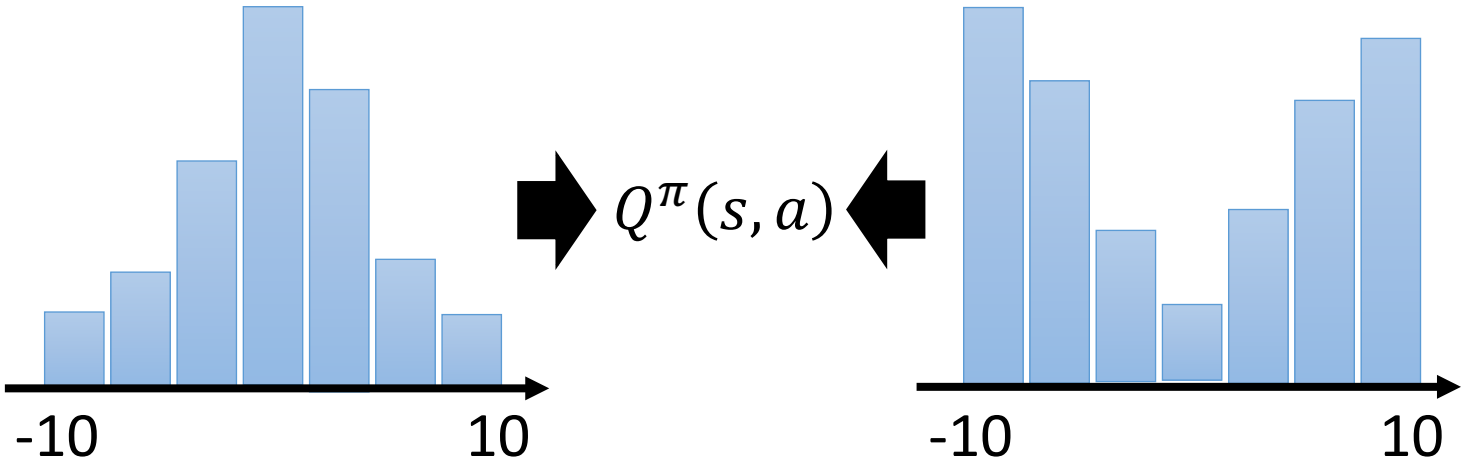
https://blog.openai.com/better-exploration-with-parameter-noise/





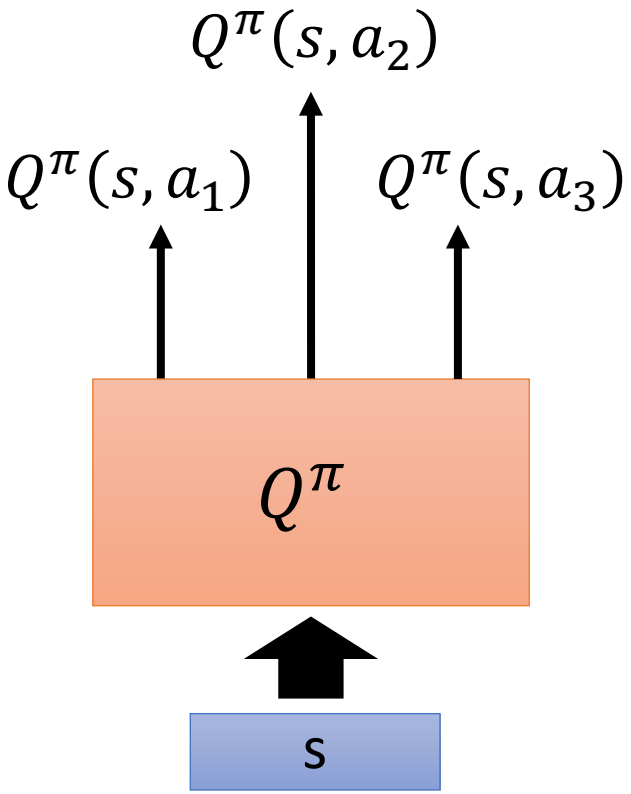
Distributional Q-function

- 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

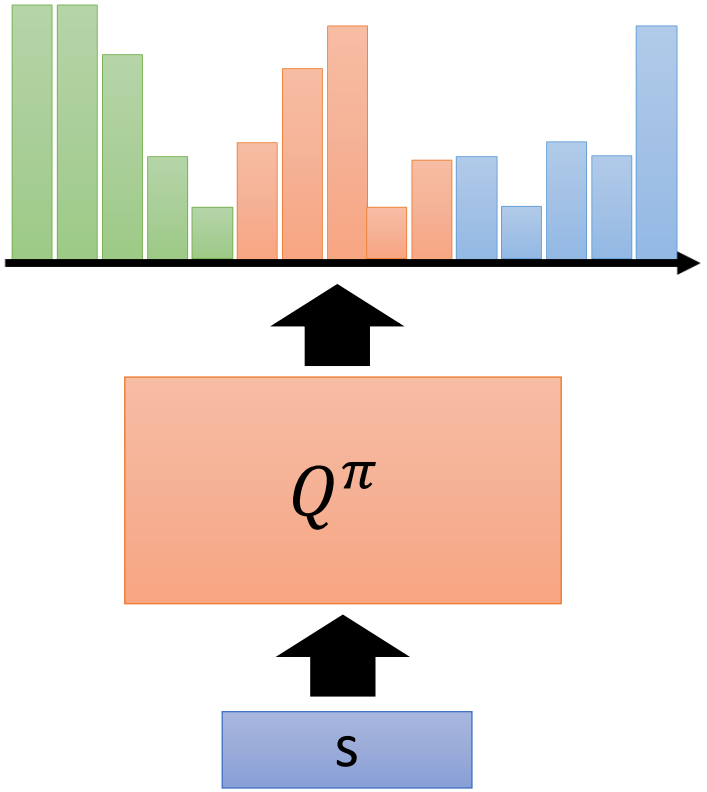


Different distributions can have the same values.

Distributional Q-function

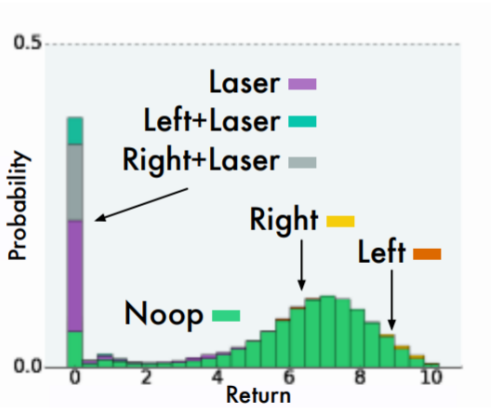


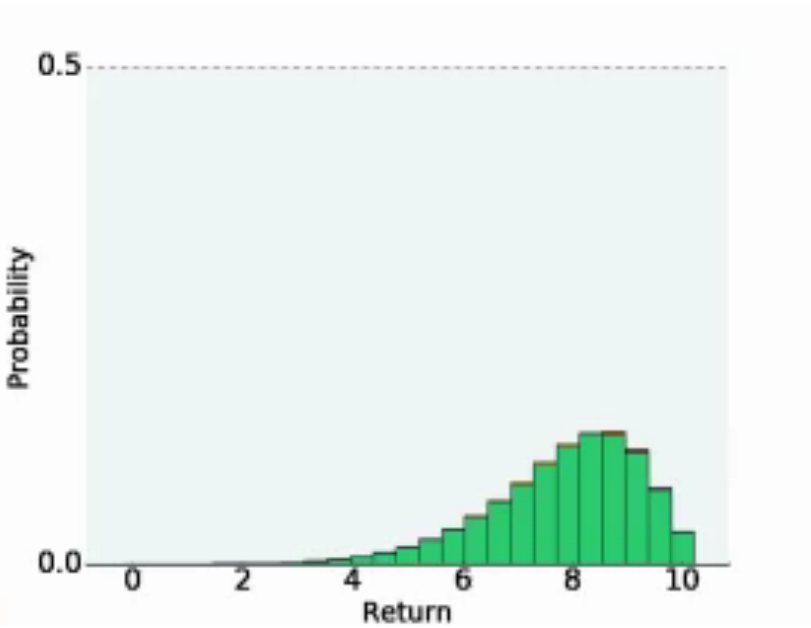
A network with 3 outputs



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

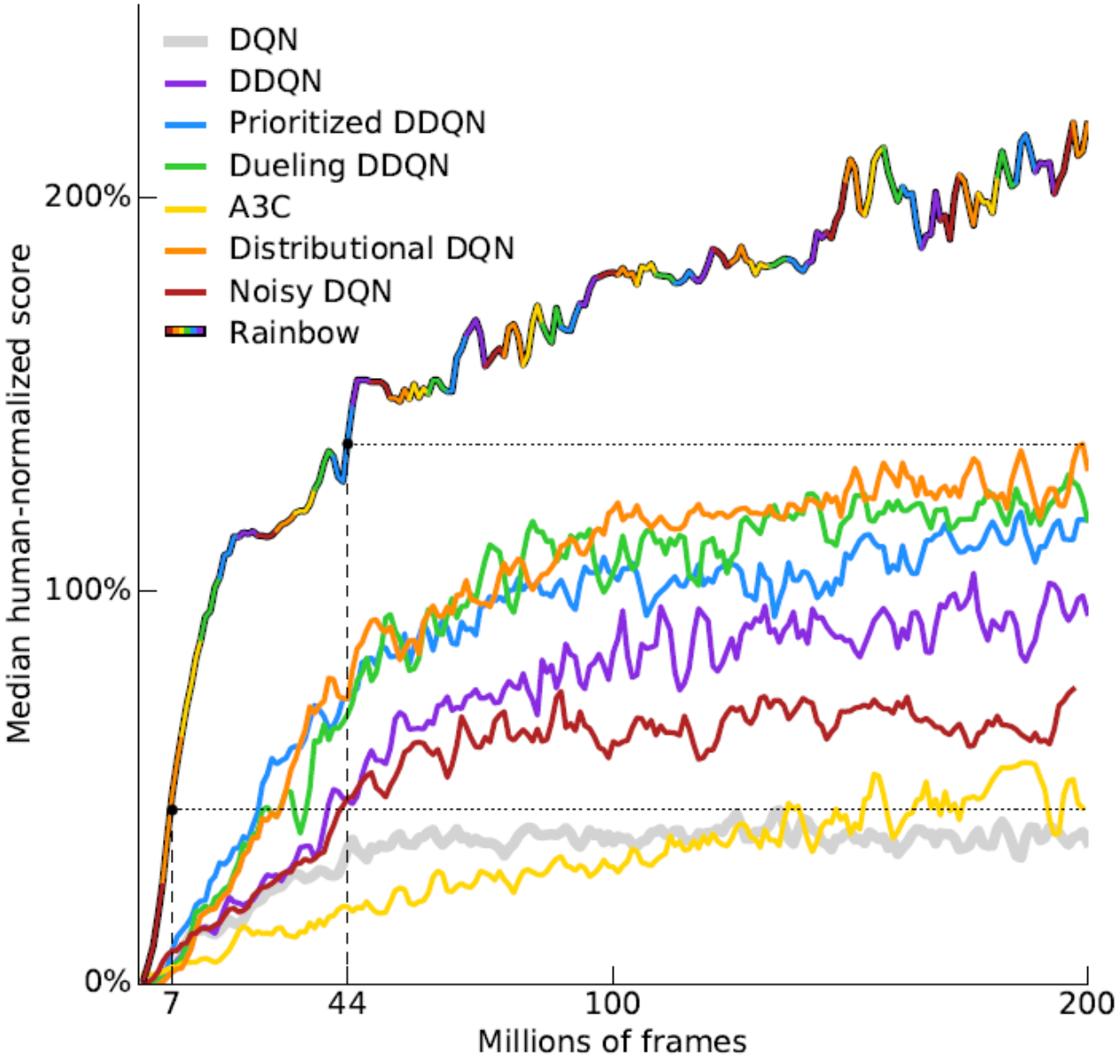
Demo





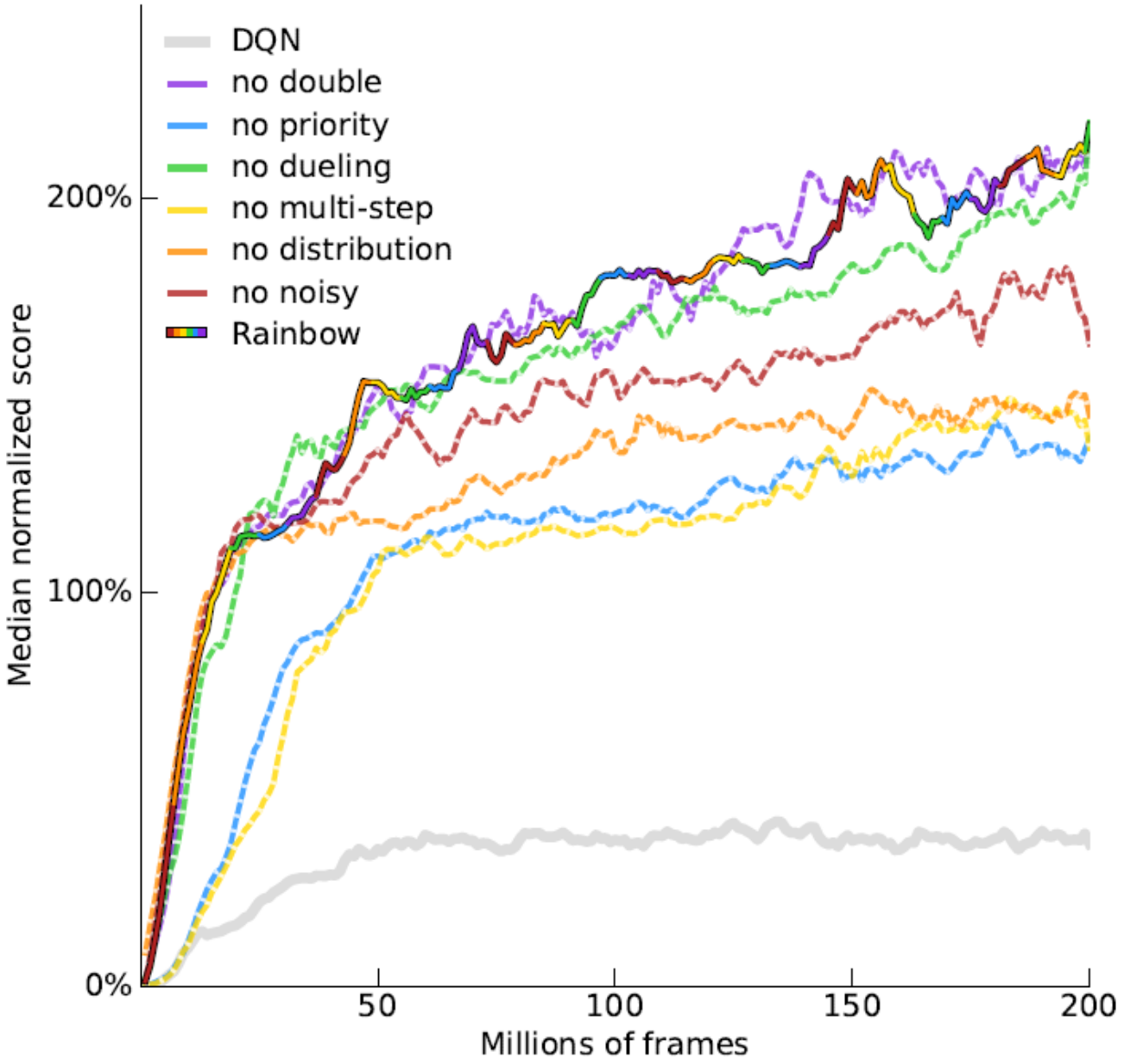
https://youtu.be/yFBwyPuO2Vg

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: $\{a_1, a_2, \cdots, a_N\}$

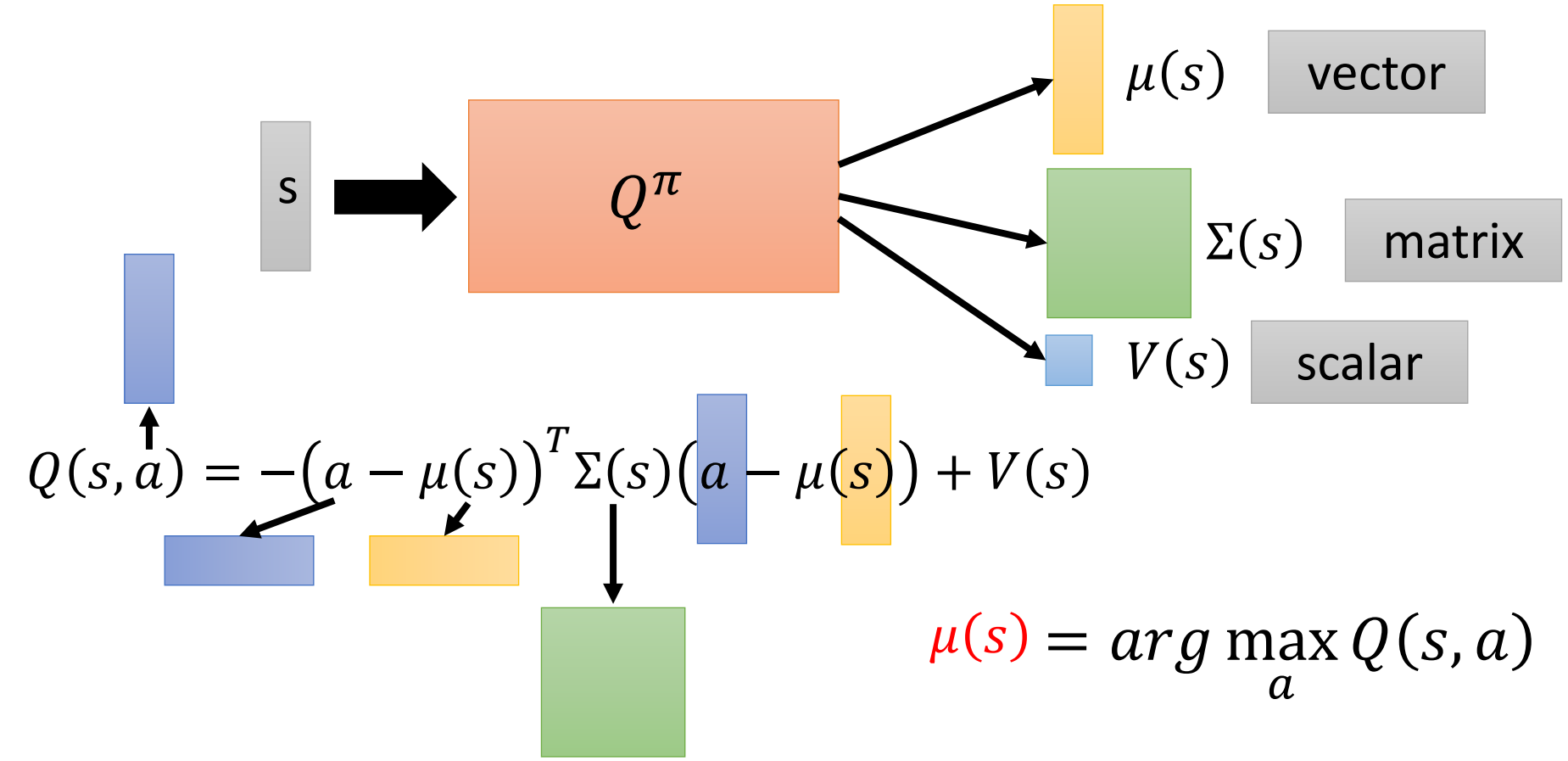
See which action can obtain the largest Q value

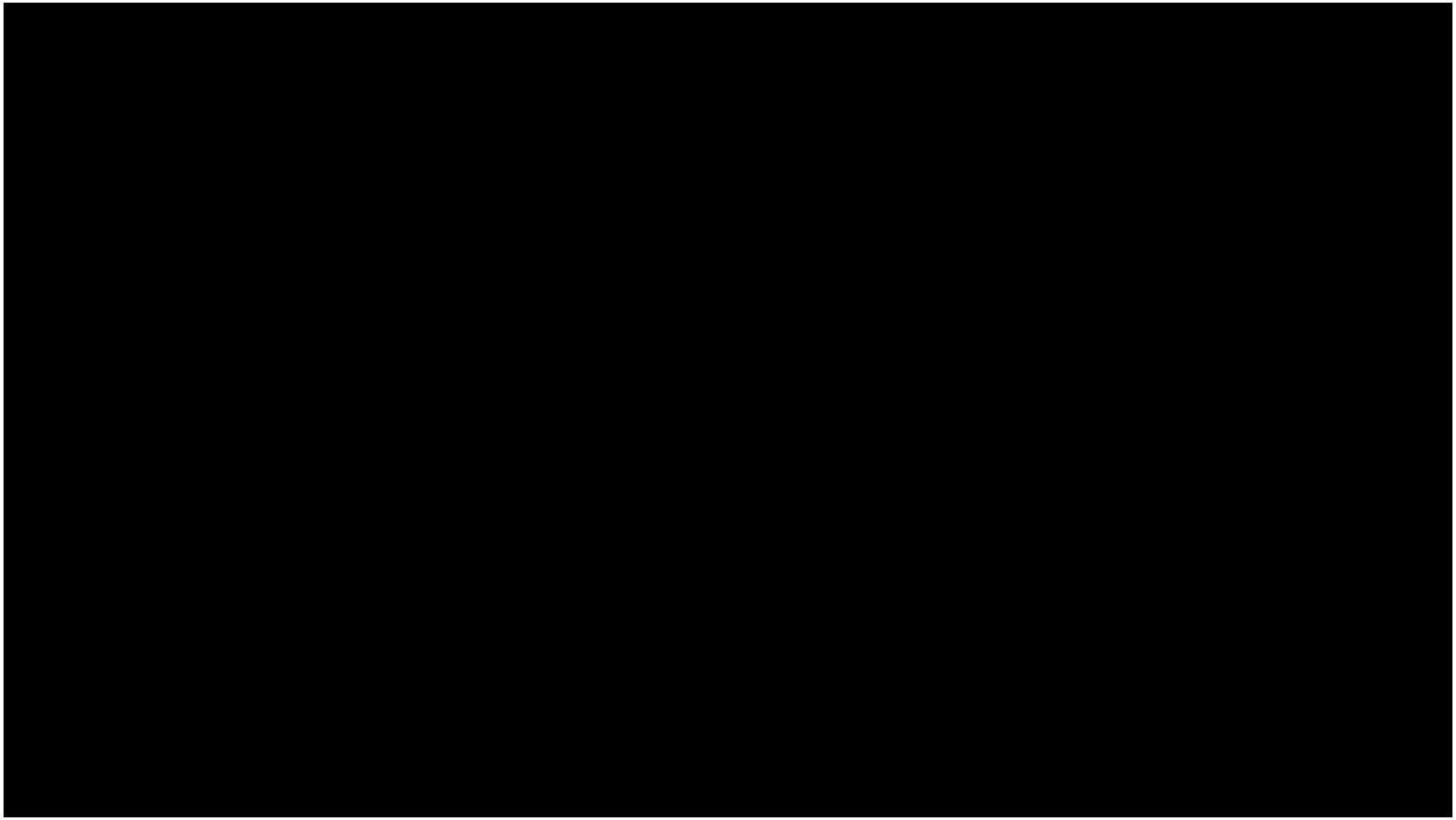
Solution 2

Using gradient ascent to solve the optimization problem.

Continuous Actions

Solution 3 Design a network to make the optimization easy.





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

Continuous Actions

Solution 4 Don't use Q-learning