Q-Learning

Hung-yi Lee

Outline

Introduction of Q-Learning

Tips of Q-Learning

Q-Learning for Continuous Actions

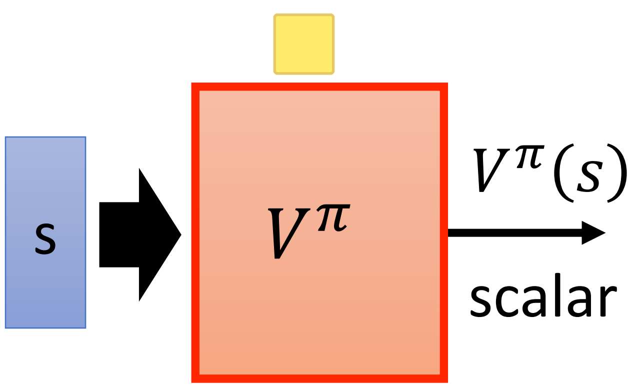
Q-Learning 是一個 Value-Based 的方法
主要是在 Train 一個 Critic

Critic

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

功能

- 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

$$V^{以前的阿光}(大馬步飛) = \text{bad}$$

$$V^{變強的阿光}(大馬步飛) = \text{good}$$

當 Actor 改變時，就算是相同的 State，
State Value Function
Critic 也會給出不同的
Output

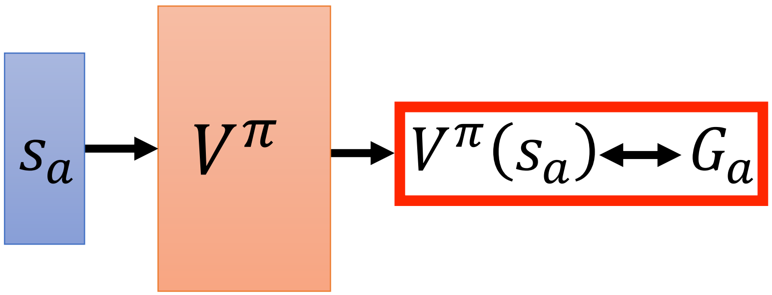
如何去訓練 State Value Function (Critic)

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

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

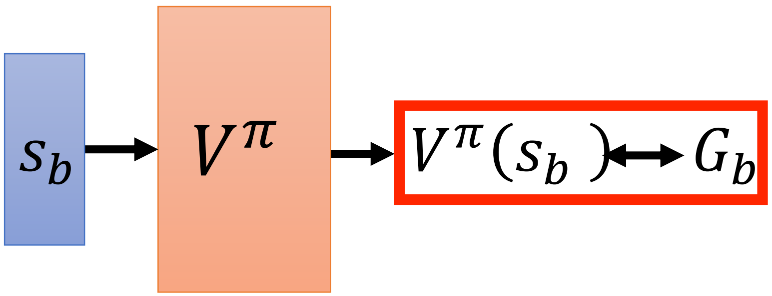
After seeing s_a,

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

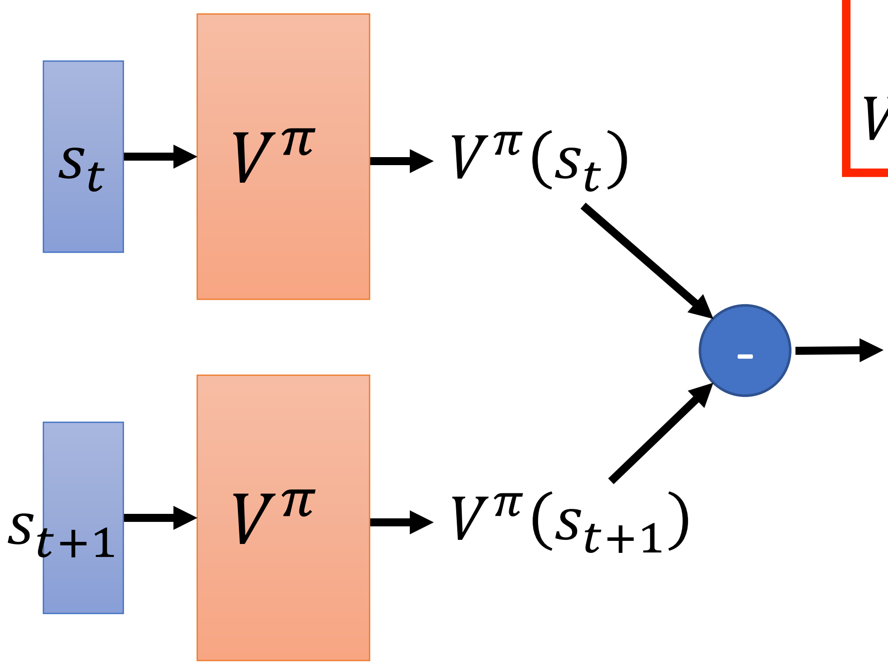


How to estimate $V^\pi(s)$

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

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

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

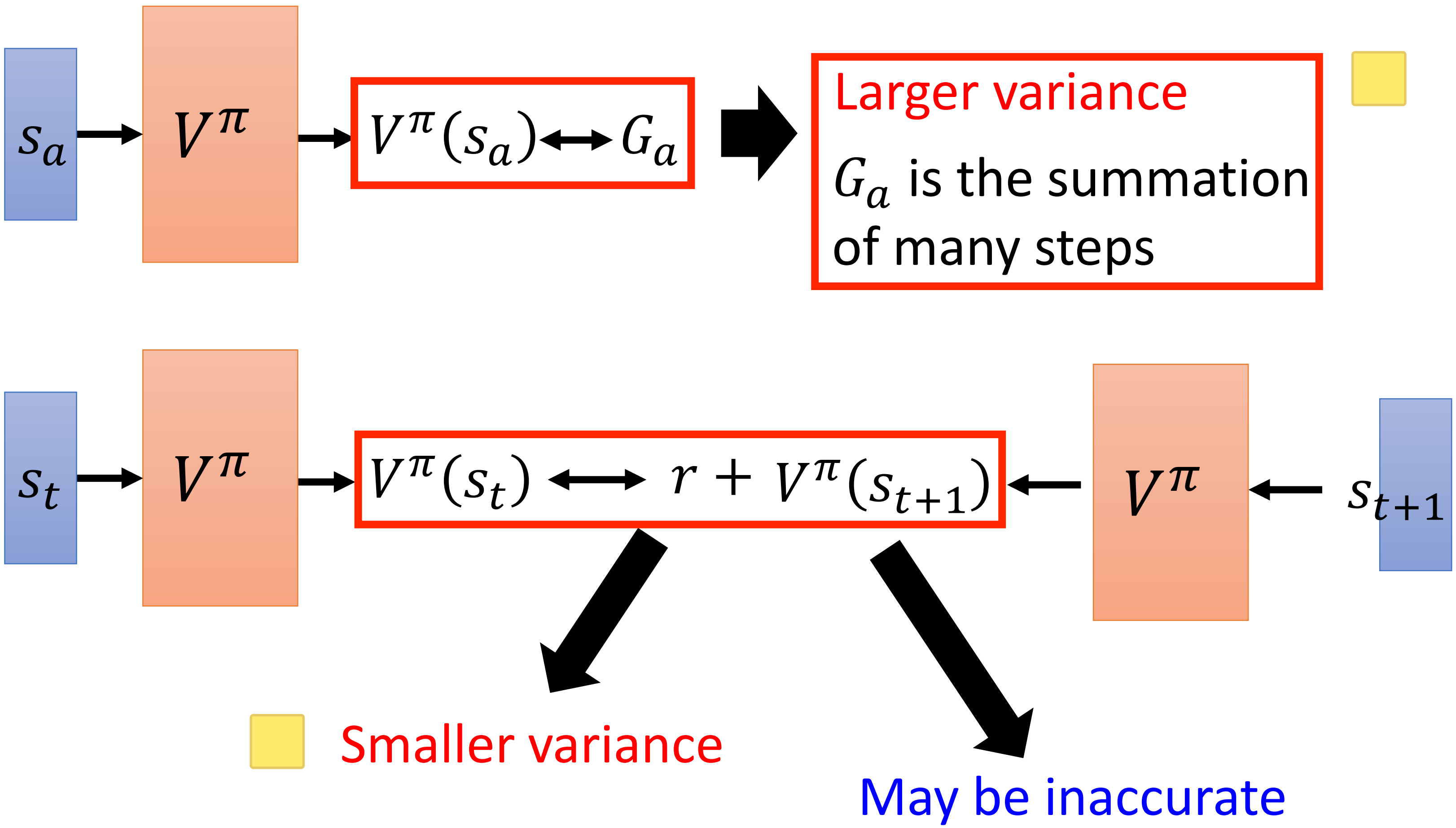


$$V^\pi(s_t) - V^\pi(s_{t+1}) \leftrightarrow r_t$$

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

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

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

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

Temporal-difference:

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

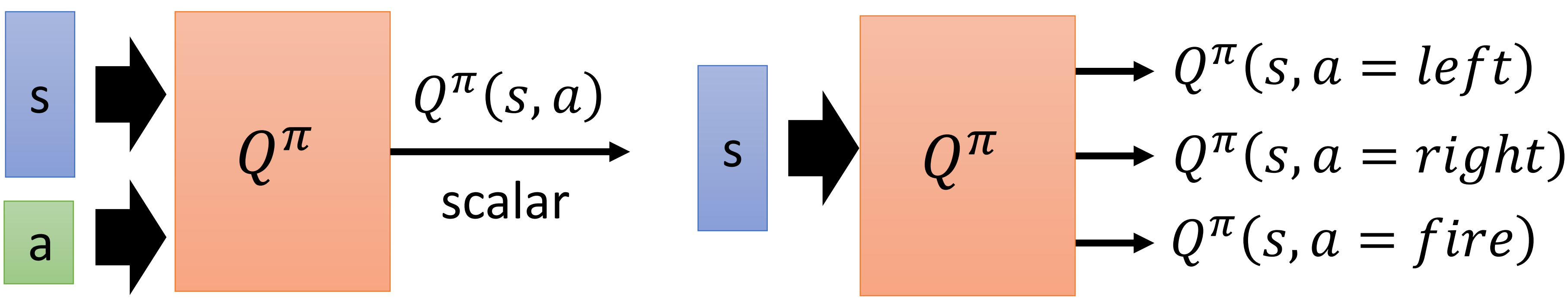
3/4 3/4 0

(The actions are ignored here.)

Review: 前面介紹的 Critic 是 State Value Function

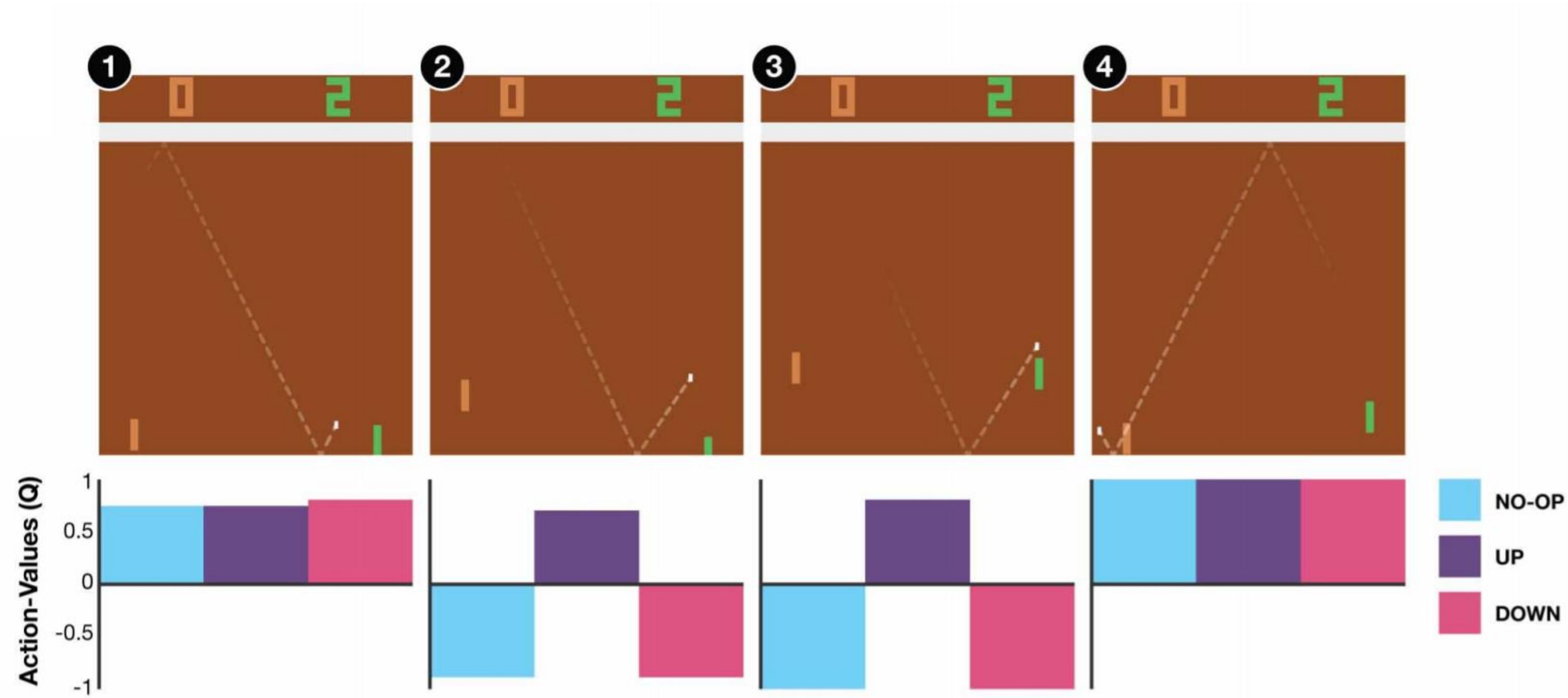
Another Critic

- 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

利用一個 Critic 搭配一個 Actor 也可以來做 Reinforcement Learning (Q-Learning)

Another Way to use Critic: Q-Learning

π interacts with the environment

TD or MC

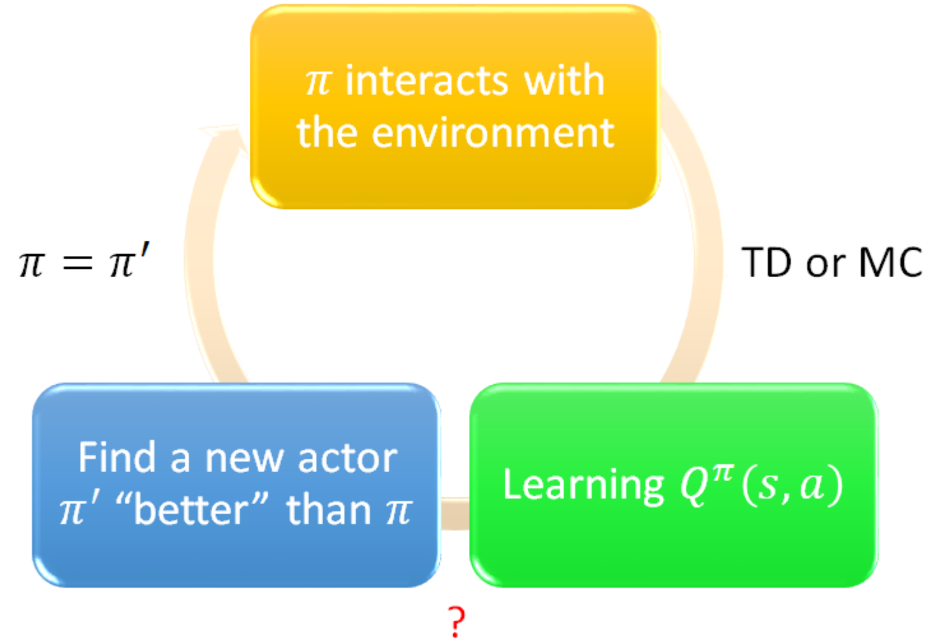
Learning $Q^{\pi}(s,a)$

?

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 (solve it later)

Q-Learning

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

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

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

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

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

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

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

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

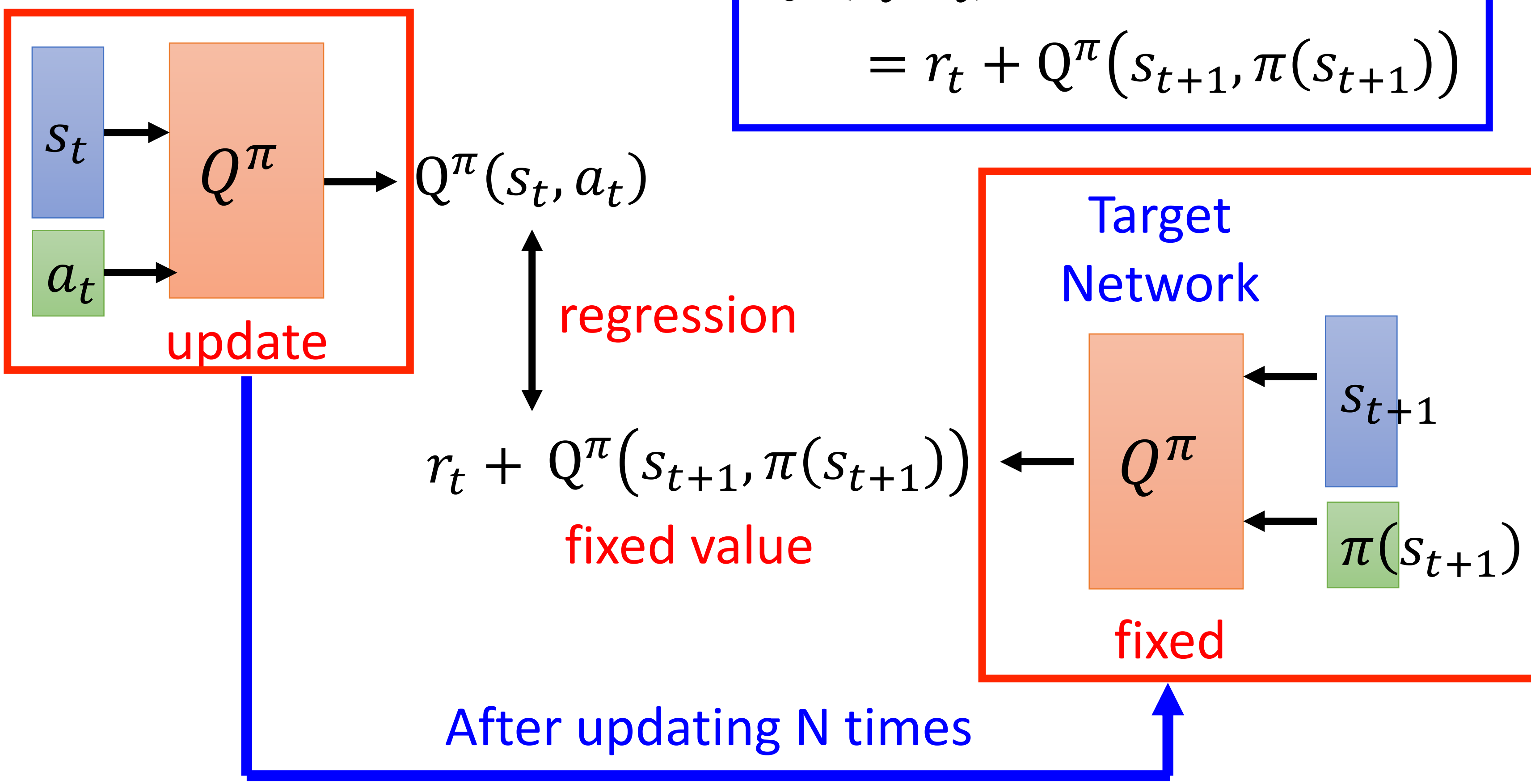
$$\leq E[r_{t+1} + r_{t+2} + Q^{\pi}\big(s_{t+2}, \pi'(s_{t+2})\big) | \ldots] \ldots \leq V^{\pi'}(s)$$

Target Network

實際上在 Implement State-Action Value Function 時，會利用兩個 Network 來實作！

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

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



Exploration

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

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

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

Epsilon Greedy

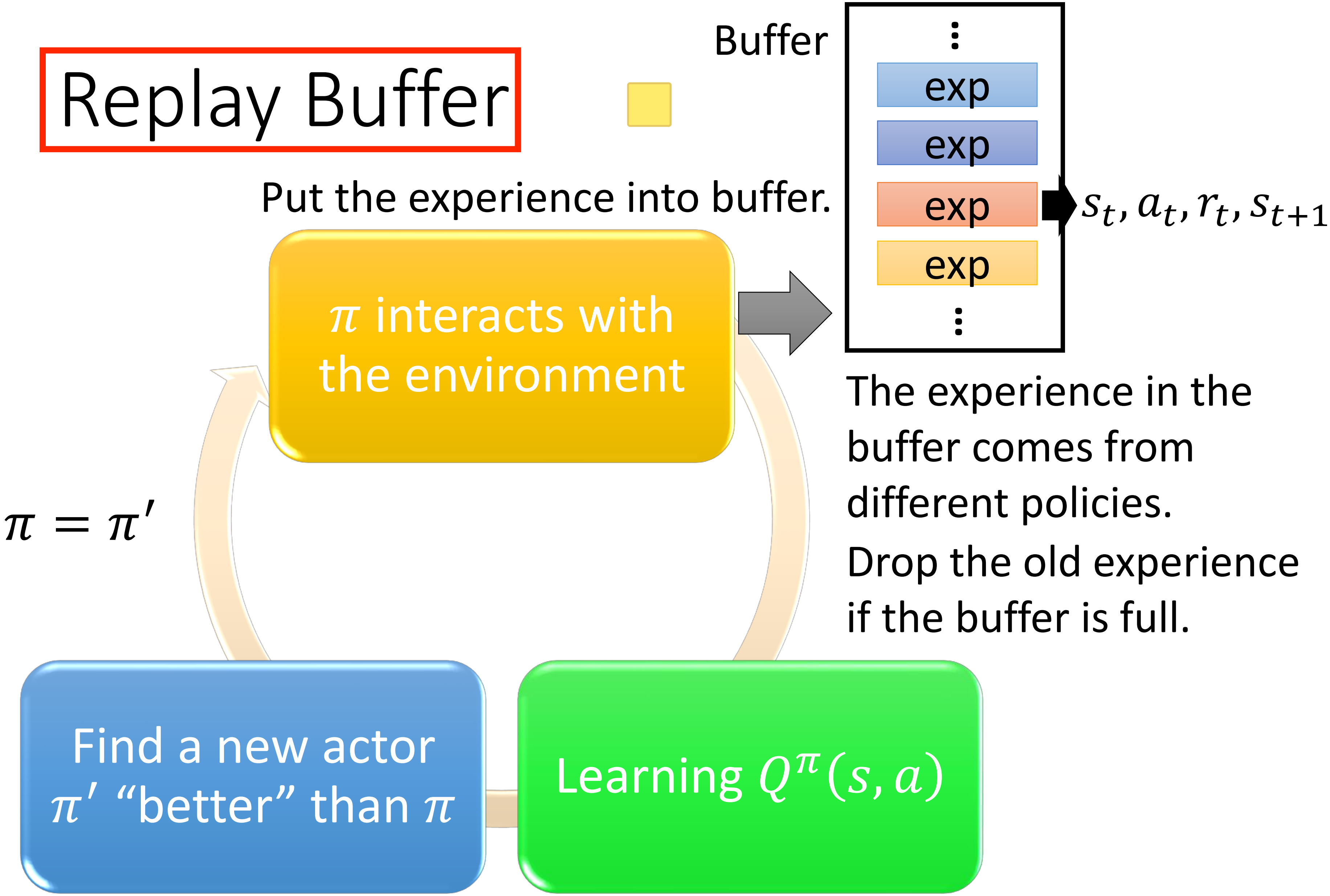
ε would decay during learning

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

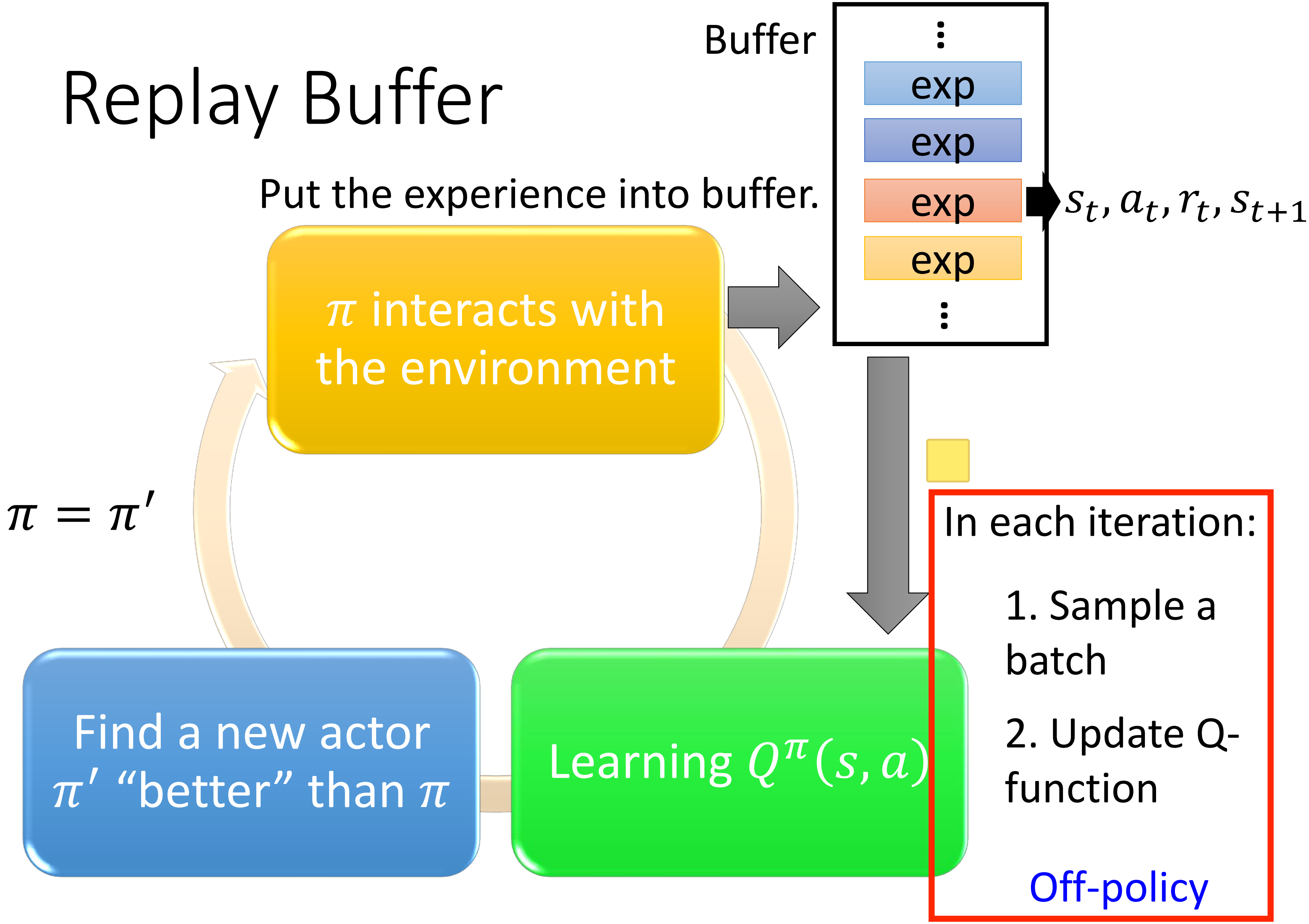
Boltzmann Exploration

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

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 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 $\hat{Q} = Q$