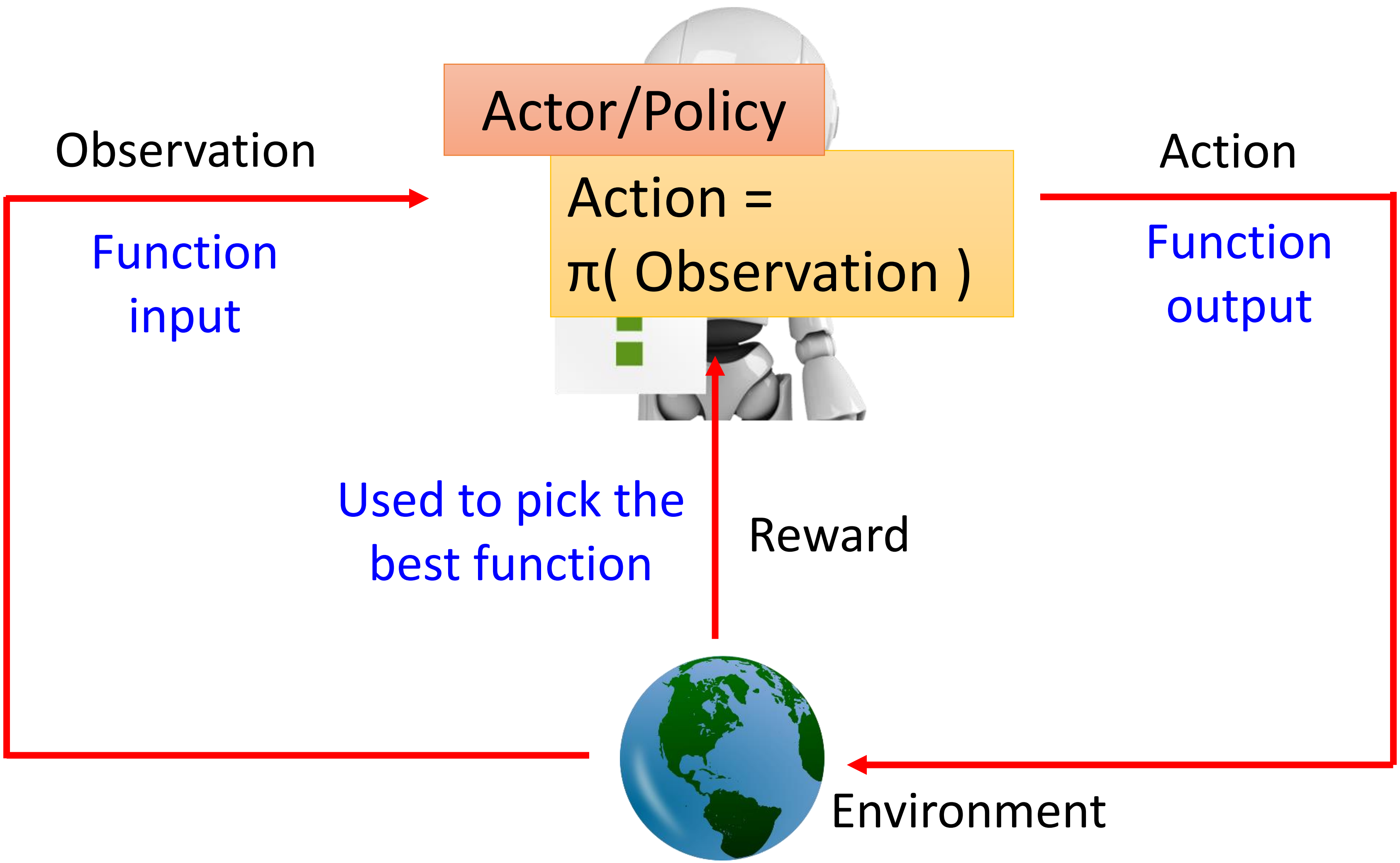
Introduction of Reinforcement Learning

Machine Learning ≈Looking for a Func



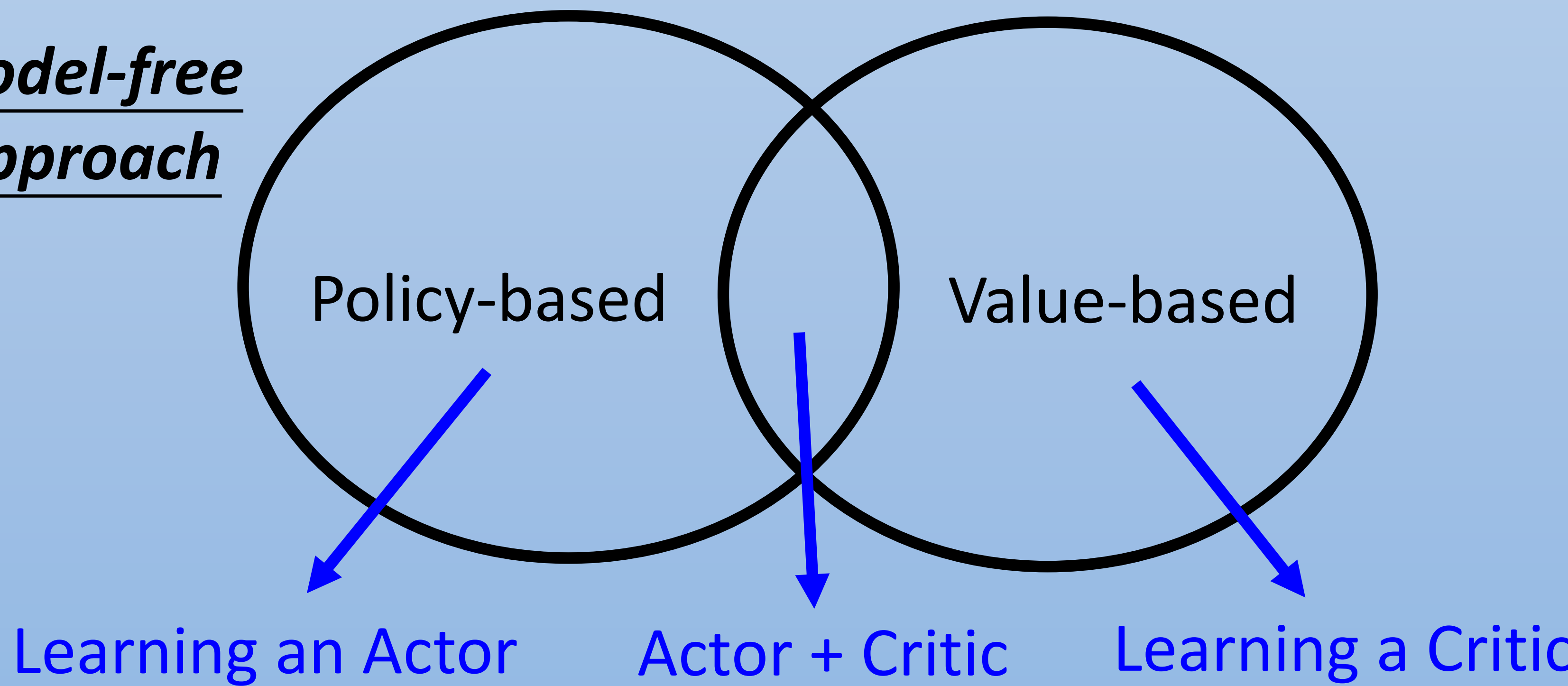
Properties of Reinforcement Learning

- **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



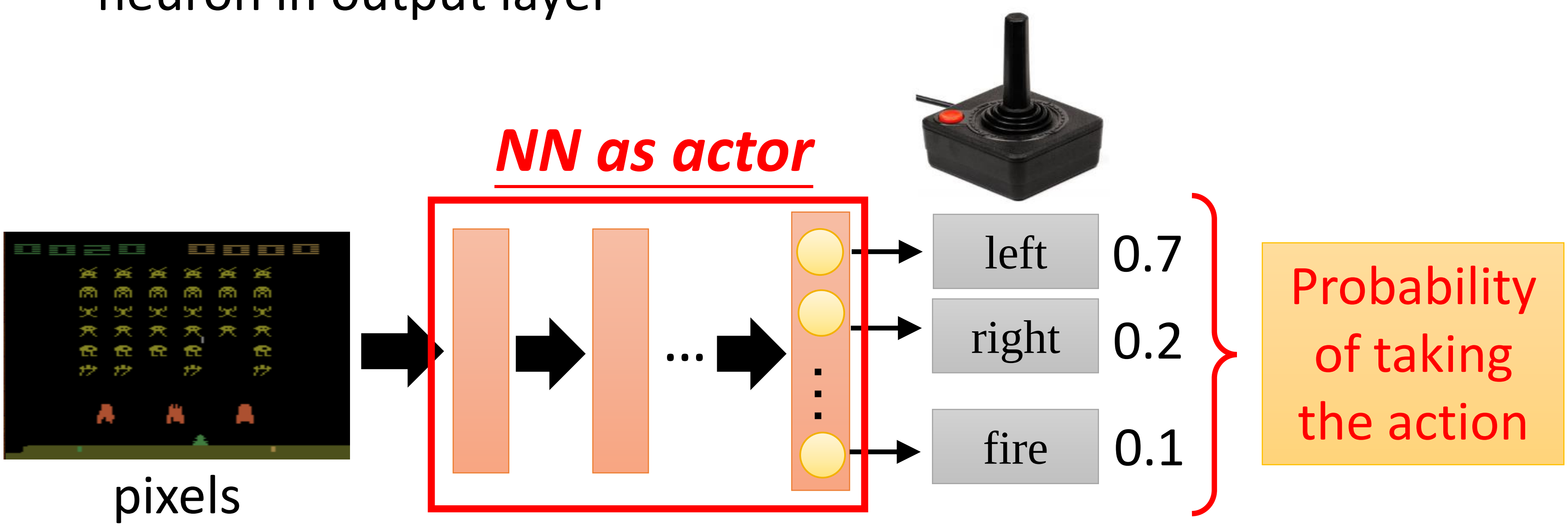
Model-based Approach

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

- Given an actor $\pi_\theta(s)$ with network parameter θ
- Use the actor $\pi_\theta(s)$ to play the video game

Total reward: $R_\theta = \sum_{t=1}^{T} r_t$,

Even with the same actor

R_θ is different each time

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

$\bar{R}_\theta$ evaluates the goodness of an actor $\pi_\theta(s)$

- 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: $P(\tau|\theta)$

$$\bar{R}_\theta = \sum_{\tau} R(\tau) P(\tau|\theta) \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n)$$

Sum over all possible trajectory

Use π_θ to play the game N times, obtain $\{\tau^1, \tau^2, \cdots, \tau^N\}$

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

Given actor parameter θ

$$
\begin{array}{lll}
\tau^1: & (s_1^1, a_1^1) & R(\tau^1) \\
 & (s_2^1, a_2^1) & R(\tau^1) \\
 & \vdots & \vdots \\
\tau^2: & (s_1^2, a_1^2) & R(\tau^2) \\
 & (s_2^2, a_2^2) & R(\tau^2) \\
 & \vdots & \vdots
\end{array}
$$

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

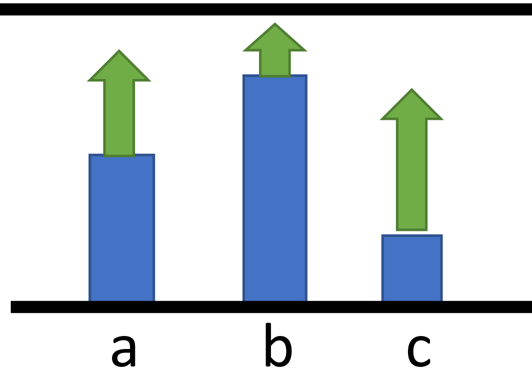
Add a Baseline

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

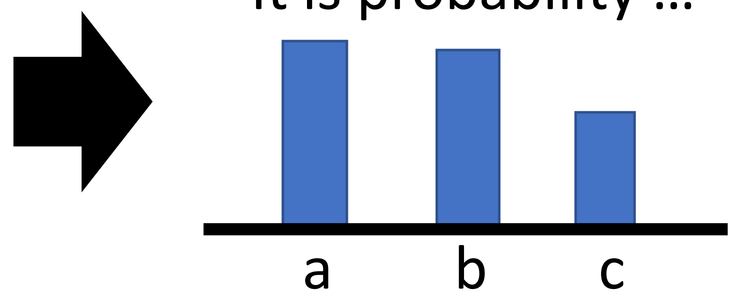
$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \bar{R}_{\theta^{old}}$$

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

Ideal case

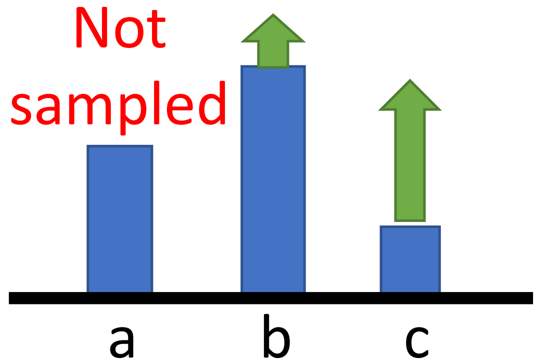


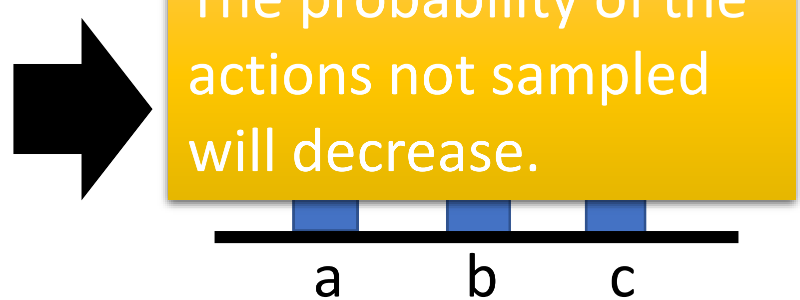
It is probability ...



Sampling ......

Not sampled





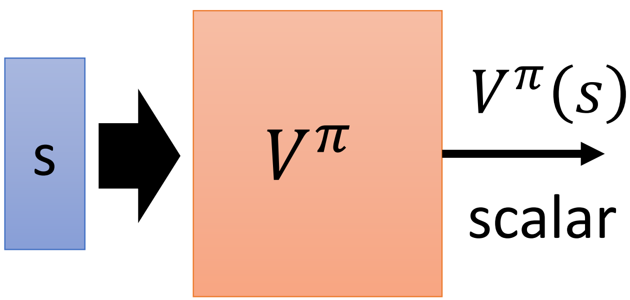
The probability of the actions not sampled will decrease.

Value-based Approach

Learning a Critic

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

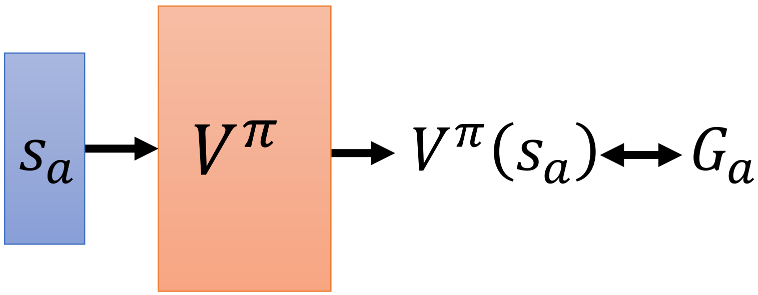
$V^{\pi}(s)$ is smaller

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

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

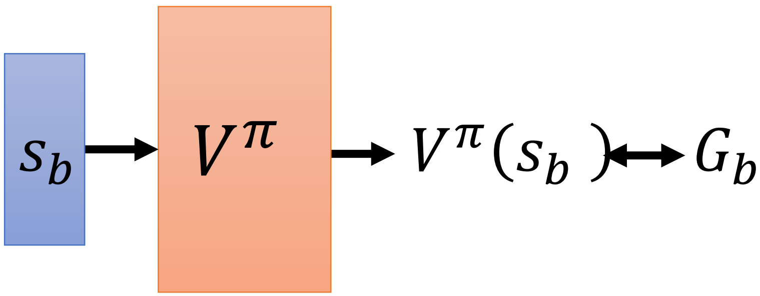
After seeing s_a,

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

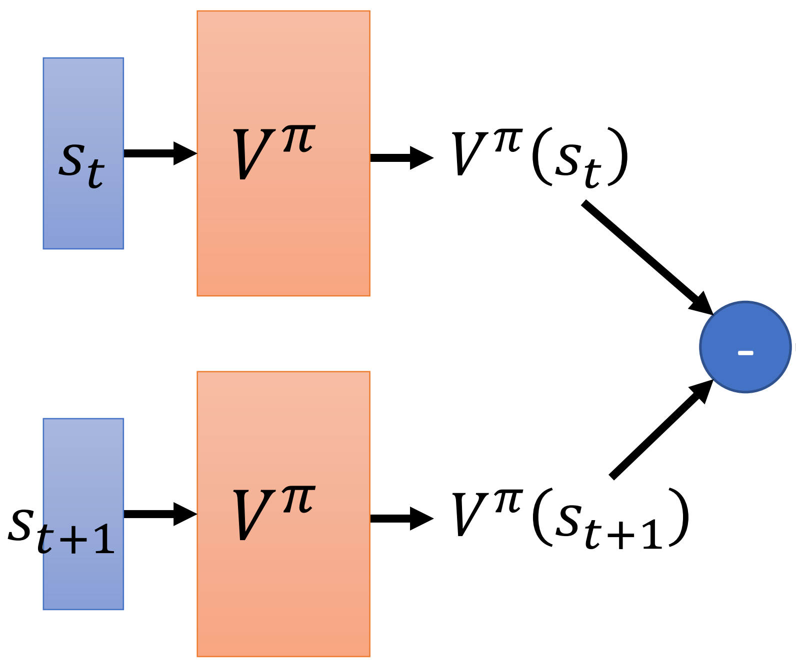


How to estimate $V^{\pi}(s)$

- Temporal-difference approach

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

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

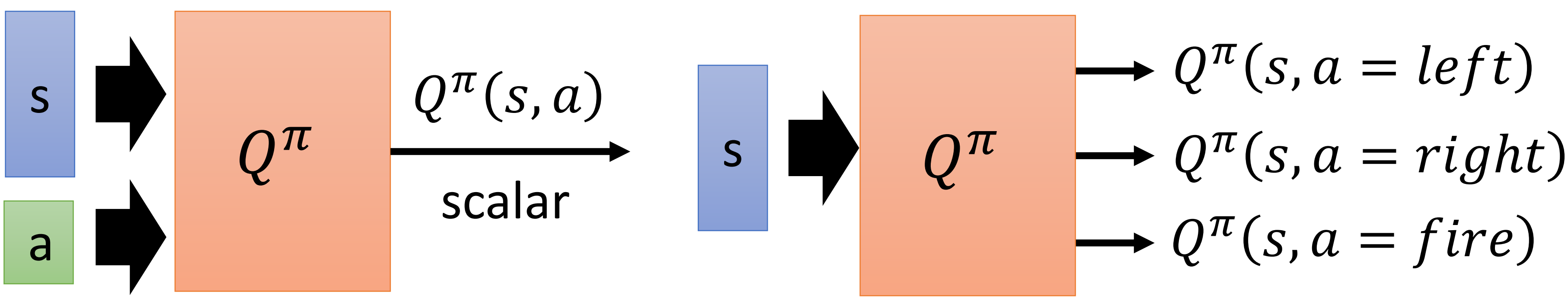


$$V^{\pi}(s_t) - V^{\pi}(s_{t+1}) \longleftrightarrow r_t$$

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

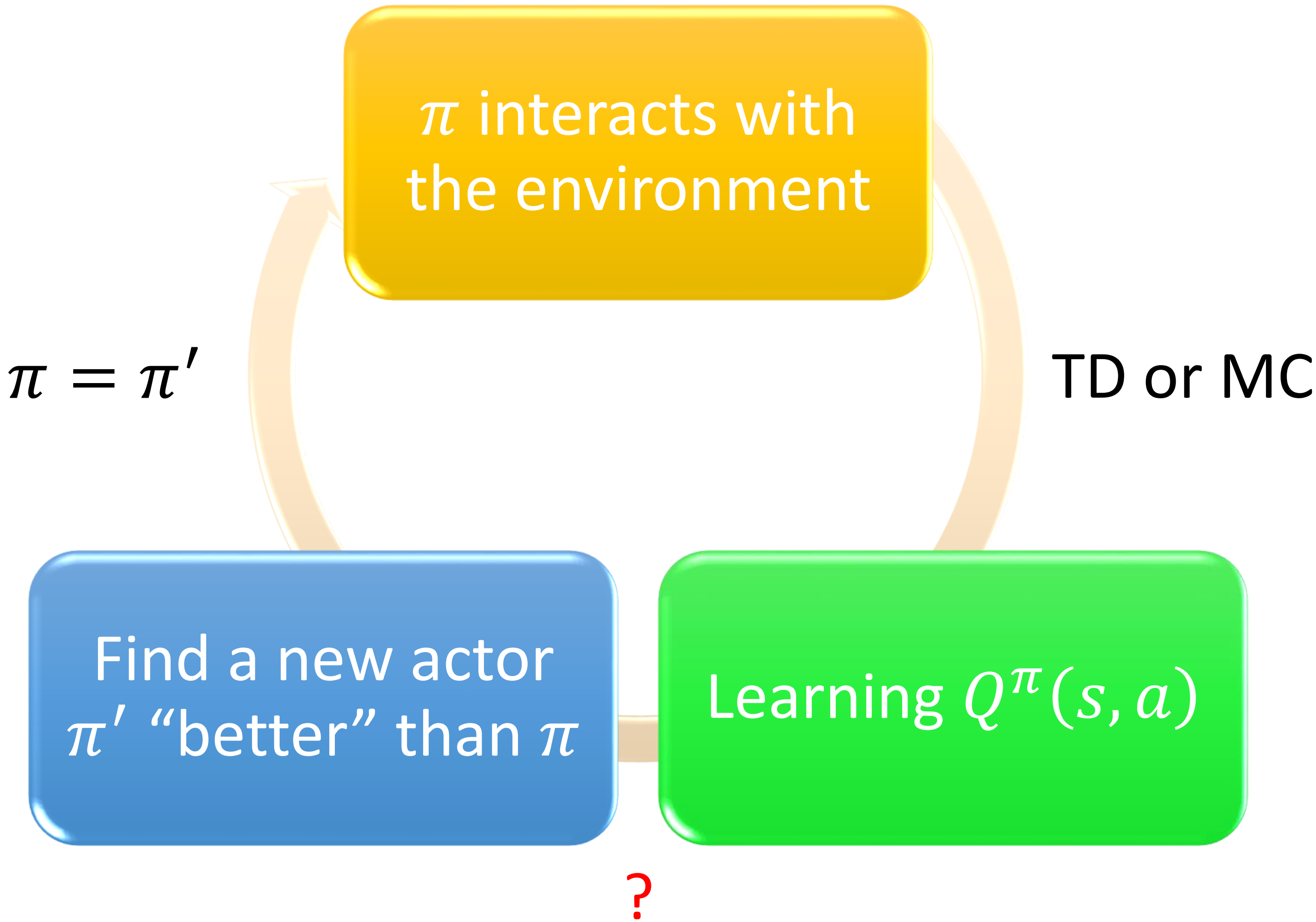
Another Critic

- 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



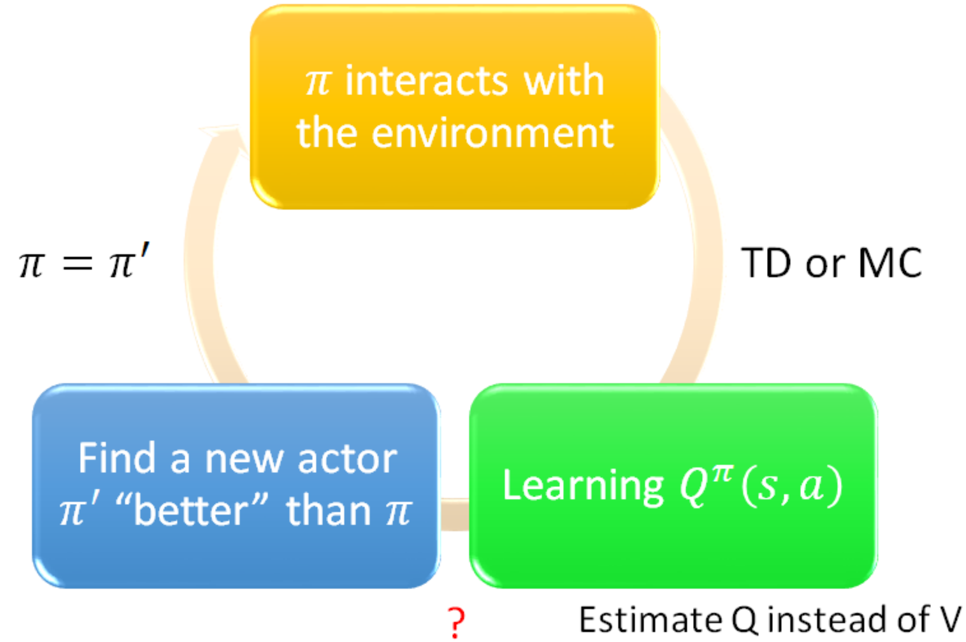
for discrete action only

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

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

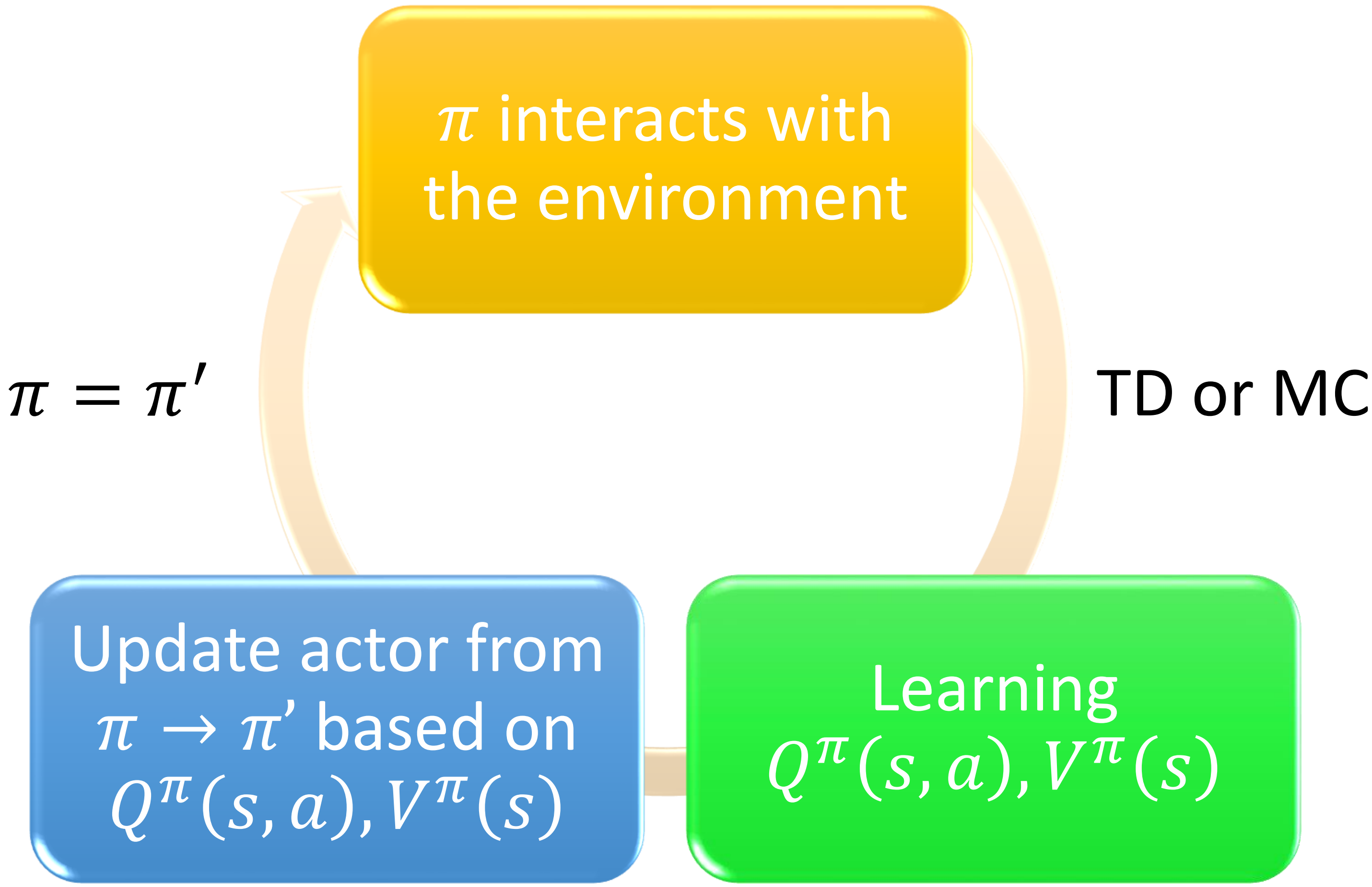
➢ π' does not have extra parameters. It depends on Q

➢ Not suitable for continuous action a

Deep Reinforcement Learning

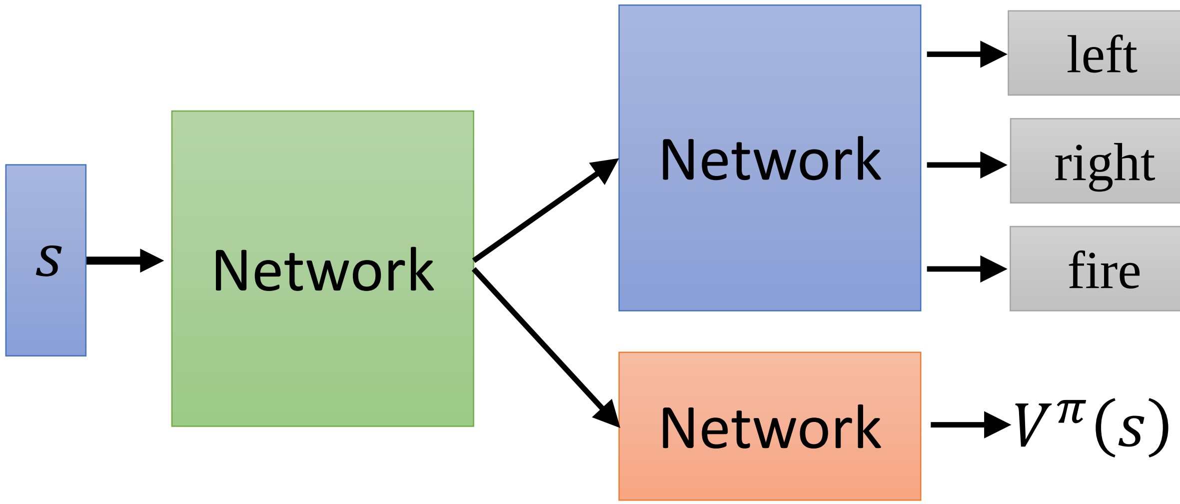
Actor-Critic

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