This lecture will be recorded!!!

Welcome to

DS595
Reinforcement Learning

Prof. Yanhua Li

Time: 6:00pm –8:50pm W
Zoom Lecture
Spring 2022

Quiz 5 Today

- 30 minutes on Policy Gradient (PG)

Project 4

- https://github.com/yingxue-zhang/DS595-RL-Projects/tree/master/Project4
- *Important Dates:*
- Progressive report: Wed. April 13, 2022 (23:59)
- Final Project:
  - Mon April 25, 2022 team project report is due
  - Wed. April 27, 2022 Virtual Poster Session

	Reinforcement Learning	Inverse Reinforcement Learning
Single Agent	**Tabular representation of reward** *Model-based control* *Model-free control* *(MC, SARSA, Q-Learning)*	**Linear reward function learning** Imitation learning Apprenticeship learning Inverse reinforcement learning MaxEnt IRL MaxCausalEnt IRL MaxRelEnt IRL
	Function representation of reward *1. Linear value function approx* *(MC, SARSA, Q-Learning)* *2. Value function approximation* *(Deep Q-Learning, Double DQN, prioritized DQN, Dueling DQN)* *3. Policy function approximation* *(Policy gradient*, PPO, TRPO) 4. Actor-Critic methods (A2C, A3C, Pathwise Derivative PG)	**Non-linear reward function learning** Generative adversarial imitation learning (GAIL) Adversarial inverse reinforcement learning (AIRL)
	Review of Deep Learning *As bases for non-linear function approximation (used in 2-4).*	**Review of Generative Adversarial nets** As bases for non-linear IRL
Multiple Agents	**Multi-Agent Reinforcement Learning** Multi-agent Actor-Critic etc.	**Multi-Agent Inverse Reinforcement Learning** MA-GAIL MA-AIRL AMA-GAIL

Applications

This Lecture

- Policy Gradient
  - Intro and Stochastic Policy
  - Basic Policy Gradient Algorithm
  - REINFORCE and Vanilla Policy Gradient
  - PPO, TRPO, PPO2
- Actor-Critic methods
  - A2C
  - A3C
  - Pathwise Derivative Policy Gradient

This Lecture

- Policy Gradient (Review Quickly)
  - Intro and Stochastic Policy
  - Basic Policy Gradient Algorithm
  - REINFORCE and Vanilla Policy Gradient
  - PPO, TRPO, PPO2
- Actor-Critic methods
  - A2C
  - A3C
  - Pathwise Derivative Policy Gradient

This Lecture

- Policy Gradient
  - Intro and Stochastic Policy
  - Basic Policy Gradient Algorithm
  - REINFORCE and Vanilla Policy Gradient
  - PPO, TRPO, PPO2
- Actor-Critic methods
  - A2C
  - A3C
  - Pathwise Derivative Policy Gradient
- Generative Adversarial Networks (GAN)
- Deep Inverse Reinforcement Learning

Review – Policy Gradient

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underbrace{\sum_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^{n}}_{G_t^n\text{ : obtained via interaction}} - \overset{\text{baseline}}{\underline{b}}\right)\nabla log\pi_\theta(a_t^n|s_t^n)$$

Review – Policy Gradient

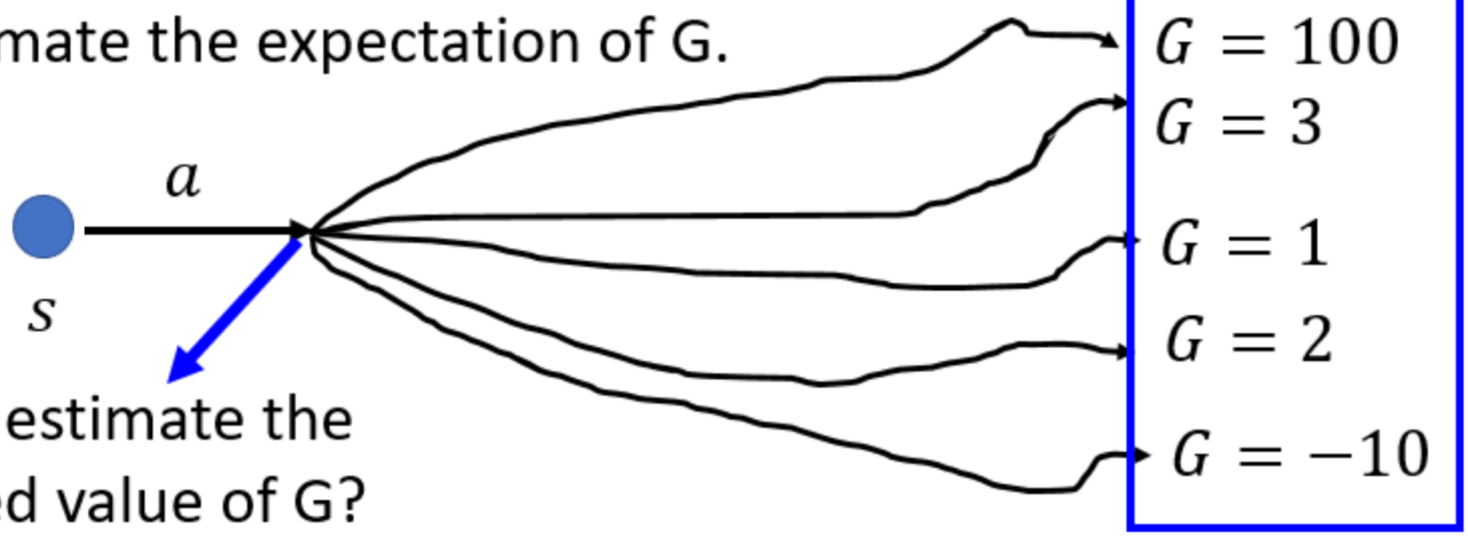
baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underline{\sum_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^n} - \underline{b}\right)\nabla log\pi_\theta(a_t^n|s_t^n)$$

G_t^n : obtained via interaction

Very unstable

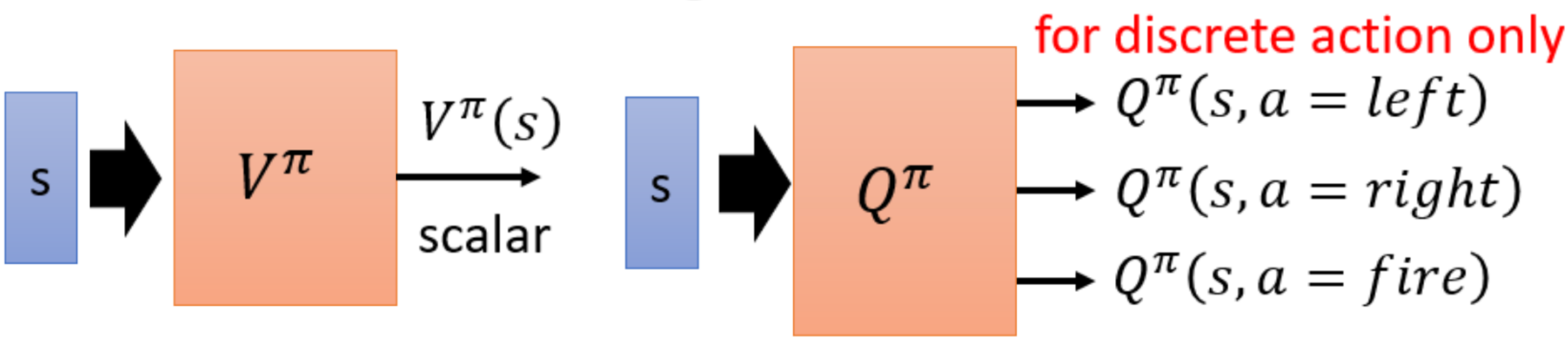
With sufficient samples,
approximate the expectation of G.



Can we estimate the
expected value of G?

Review – Q-Learning

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



Estimated by TD or MC

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underbrace{\sum_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^n}_{G_t^n\text{ : obtained via interaction}} - \overset{\text{baseline}}{\underline{b}}\right)\nabla log\pi_\theta(a_t^n|s_t^n)$$

Actor-Critic

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\underbrace{\sum_{t'=t}^{T_n}\gamma^{t'-t} r_{t'}^n}_{G_t^n \text{ : obtained via interaction}} - \overset{\text{baseline}}{\underline{b}}\right)\nabla log\pi_\theta(a_t^n|s_t^n)$$

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Actor-Critic

$V^{\pi_\theta}(s_t^n)$

baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^n - b\right)\nabla log\pi_\theta(a_t^n|s_t^n)$$

G_t^n : obtained via interaction

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Actor-Critic

$$\boxed{Q^{\pi_\theta}(s_t^n, a_t^n) - V^{\pi_\theta}(s_t^n)}$$

$$V^{\pi_\theta}(s_t^n)$$

baseline

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \boxed{\left(\underbrace{\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n} - \underline{b} \right)} \nabla log \pi_\theta(a_t^n | s_t^n)$$

G_t^n : obtained via interaction

$$E[G_t^n] = Q^{\pi_\theta}(s_t^n, a_t^n)$$

Advantage Actor-Critic

$$\boxed{Q^{\pi}(s_t^n, a_t^n) - V^{\pi}(s_t^n)}$$

Estimate two networks? We can only estimate one.

Advantage Actor-Critic

$$Q^{\pi}(s_t^n, a_t^n) - V^{\pi}(s_t^n)$$

Estimate two networks? We can only estimate one.

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

Only estimate state value

A little bit variance

$$Q^{\pi}(s_t^n, a_t^n) = E[r_t^n + V^{\pi}(s_{t+1}^n)]$$

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

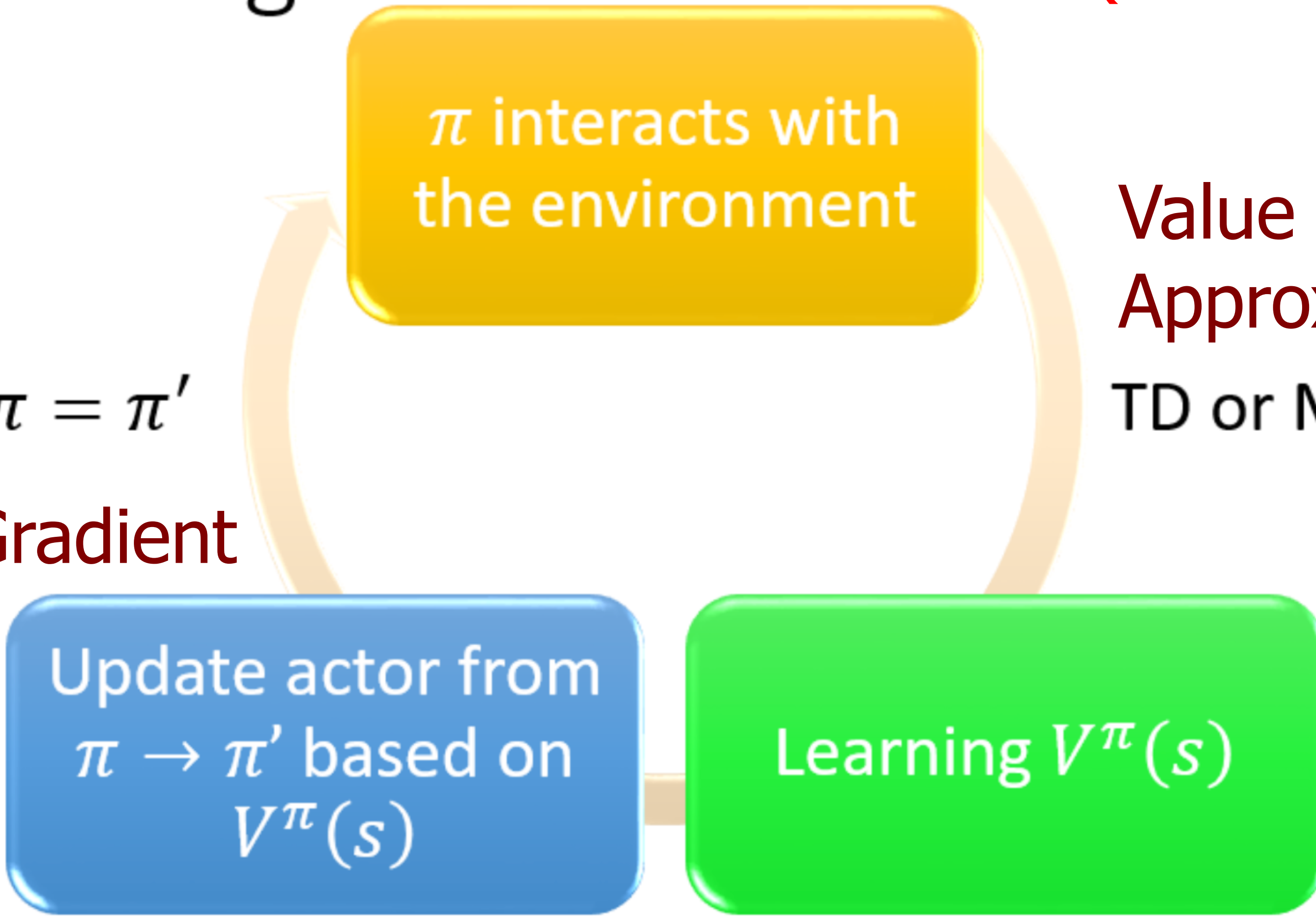
Advantage Actor-Critic (A2C algorithm)

Value function Approximation

TD or MC

$\pi = \pi'$

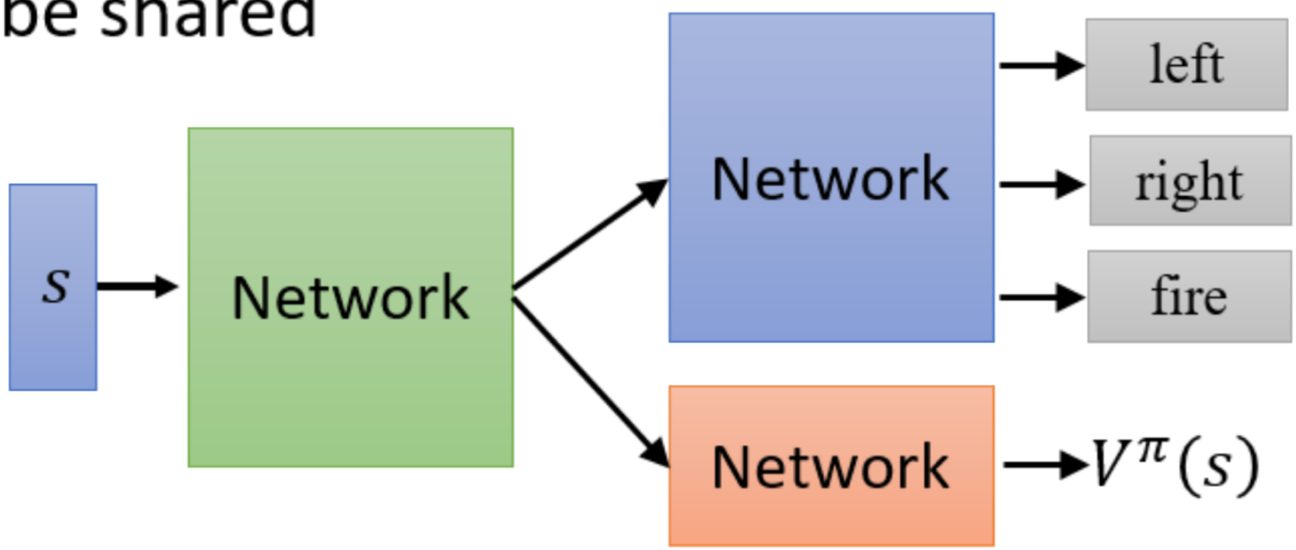
Policy Gradient



$$\nabla \bar{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(r_t^n + V^\pi(s_{t+1}^n) - V^\pi(s_t^n)\right)\nabla log\pi_\theta(a_t^n|s_t^n)$$

Advantage Actor-Critic

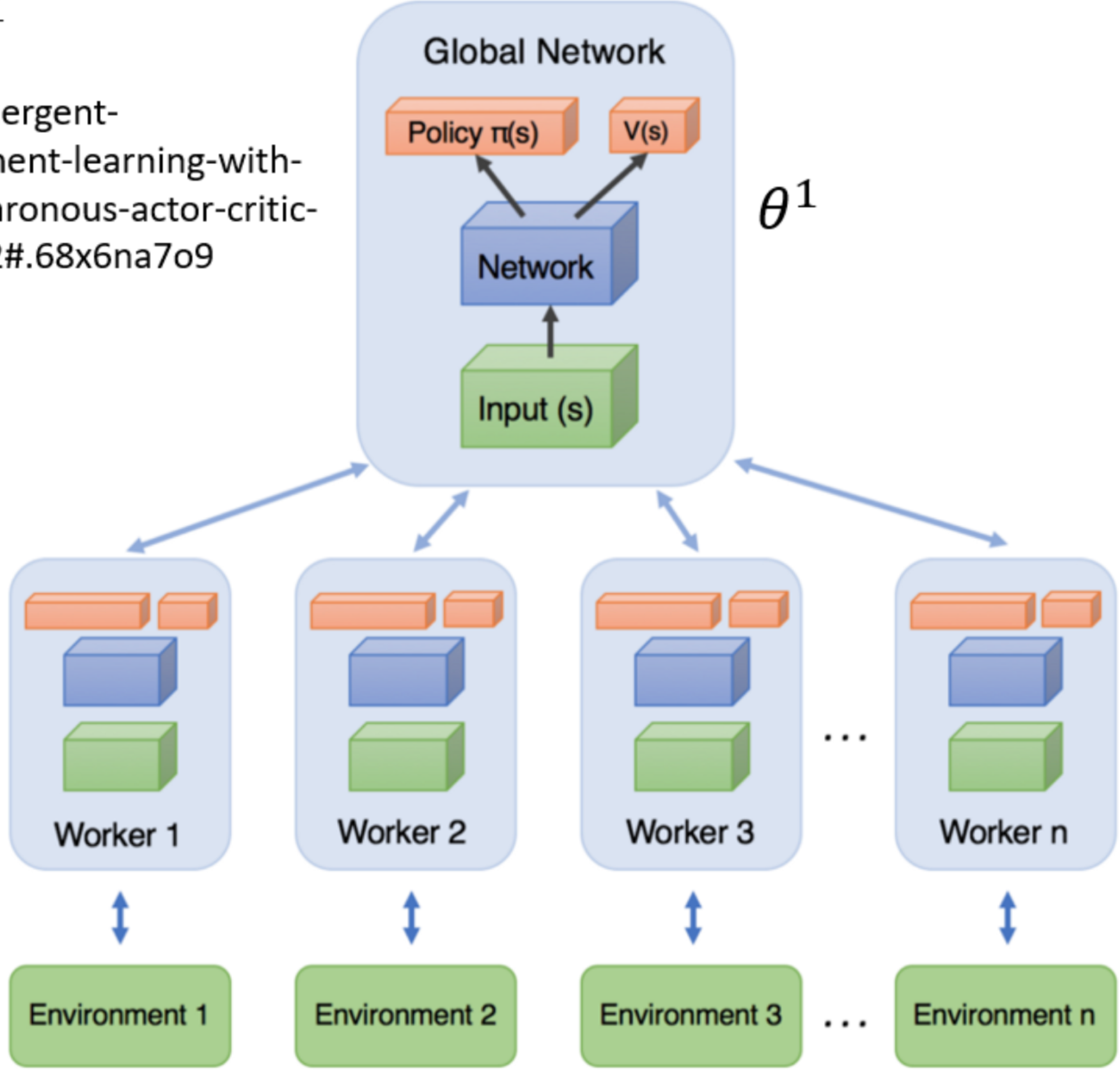
- Tips
  - The parameters of actor $\pi(s)$ and critic $V^\pi(s)$ can be shared



Asynchronous Advantage Actor-Critic (A3C)

Asynchronous

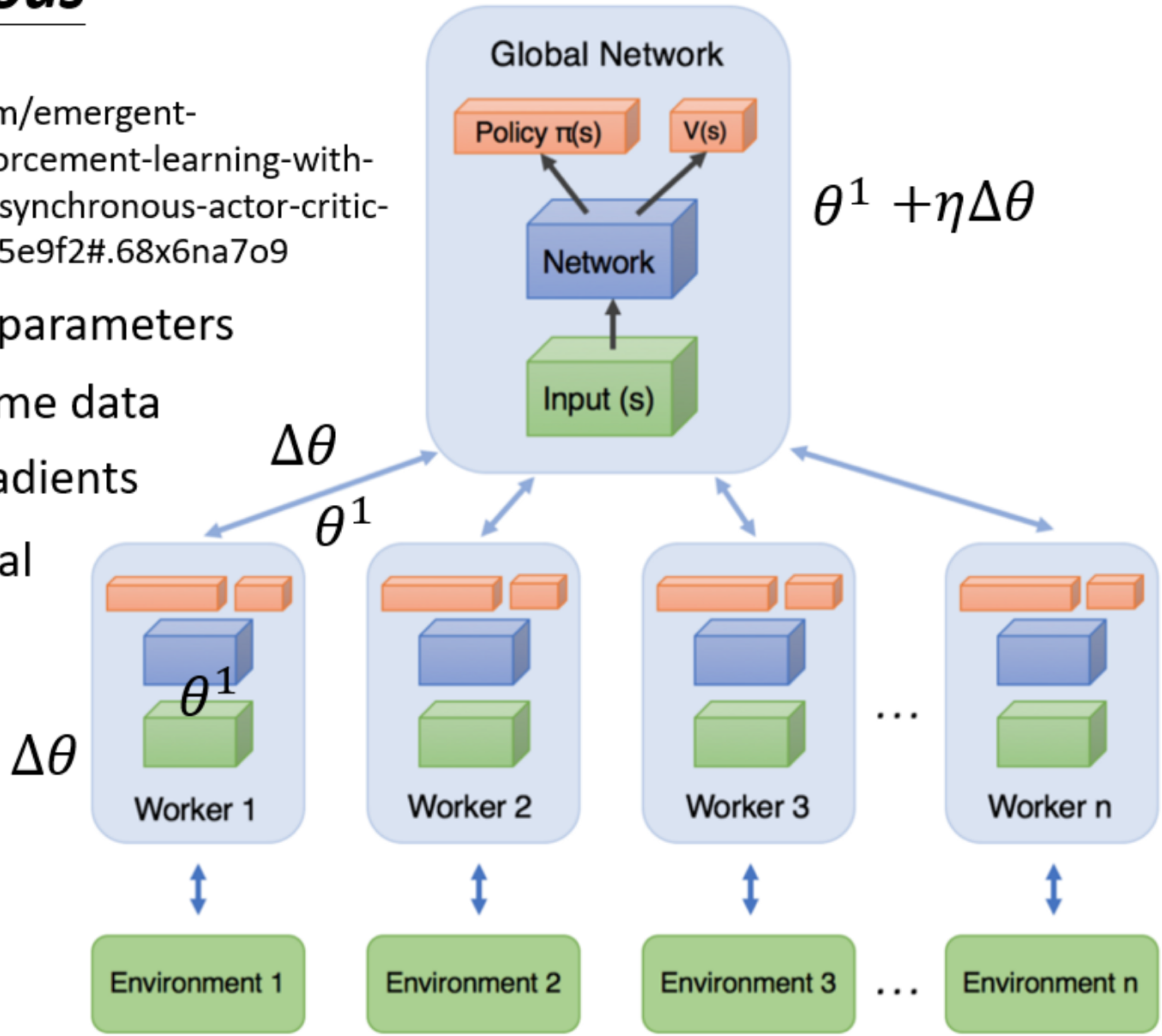
Source of image:
https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9



Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

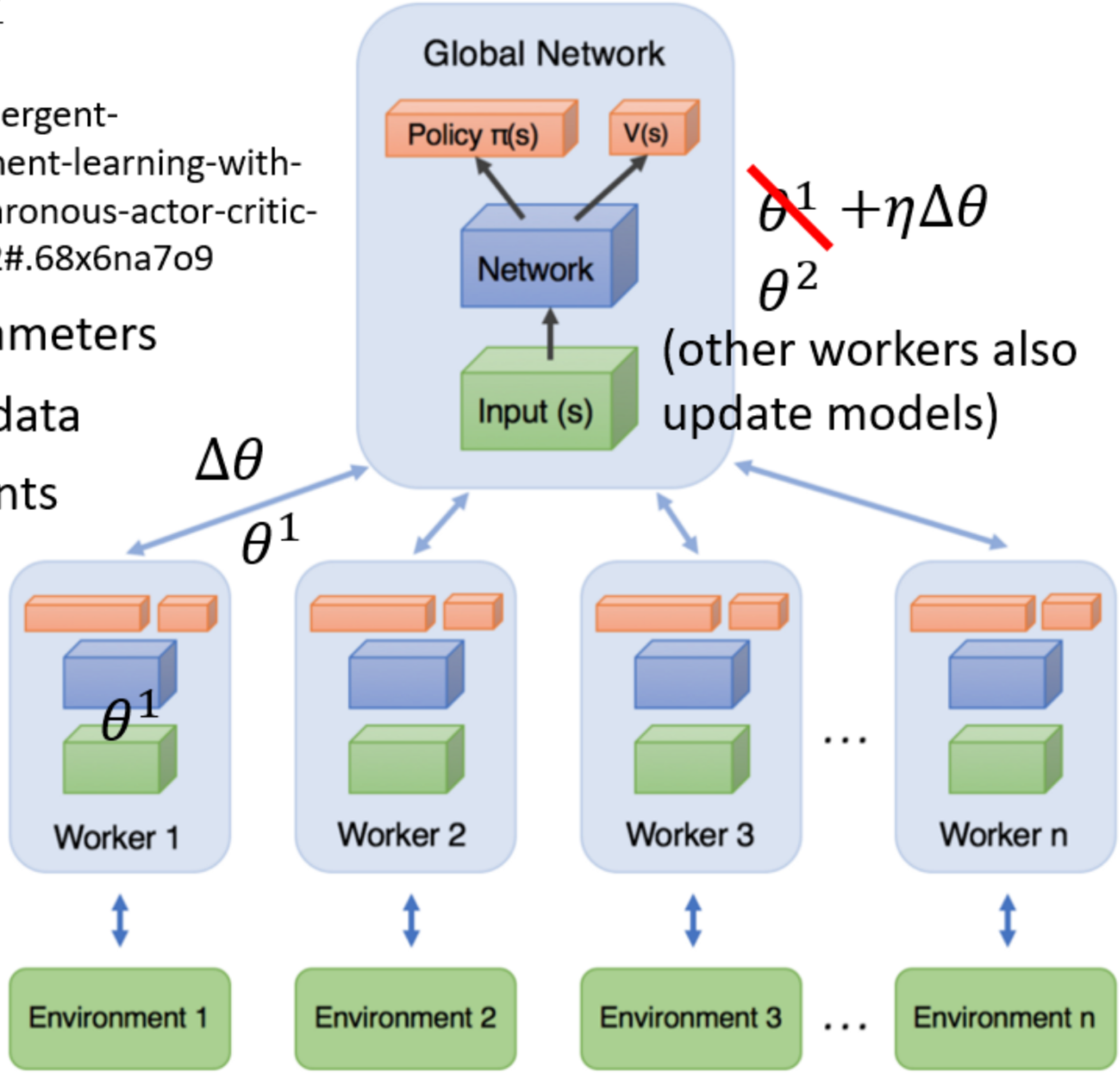
1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



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



$\cancel{\theta^1} + \eta\Delta\theta$

θ^2

(other workers also update models)

$\Delta\theta$

θ^1

θ^1

$\Delta\theta$

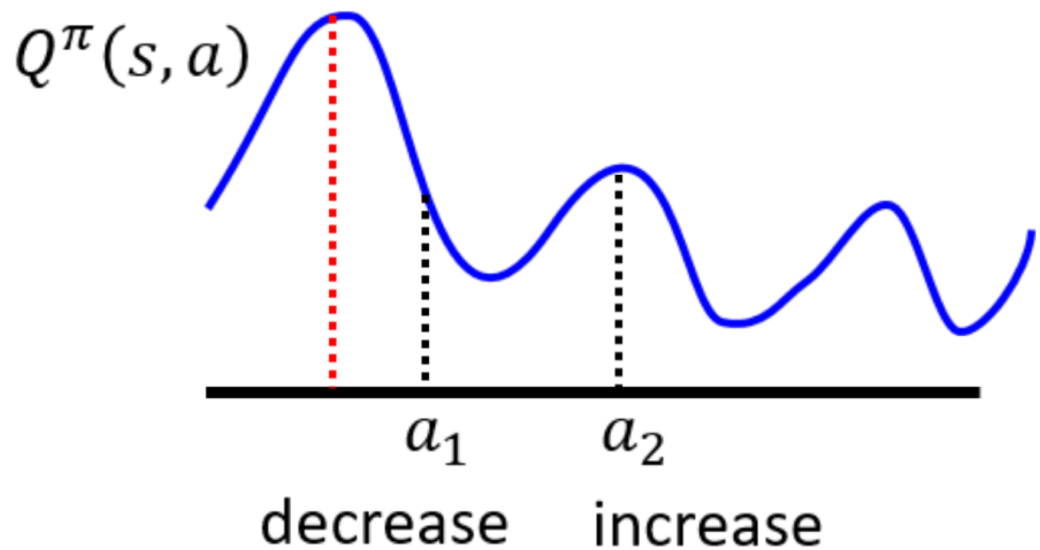
Pathwise Derivative Policy Gradient

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, Martin Riedmiller, "Deterministic Policy Gradient Algorithms", ICML, 2014

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, "CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING", ICLR, 2016

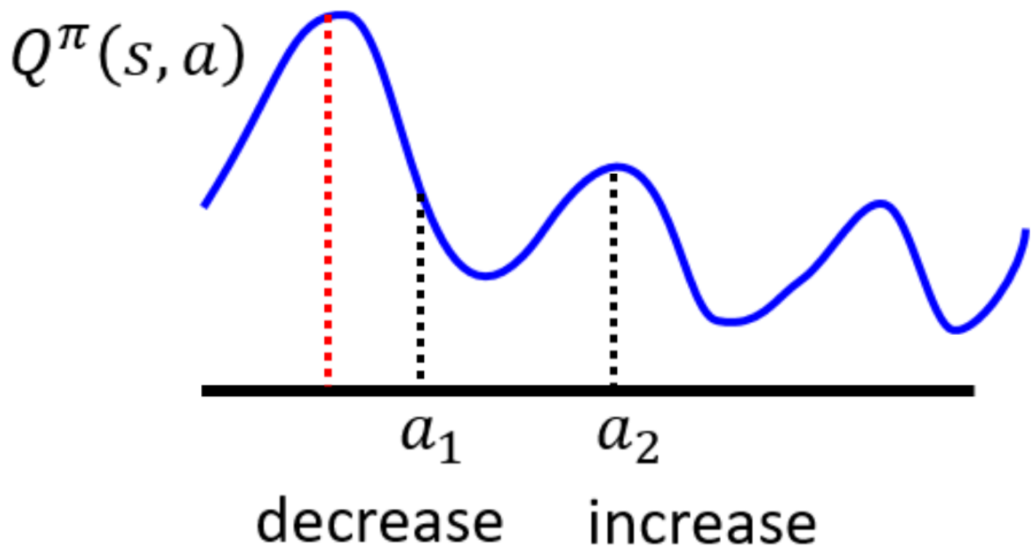
Another Way to use Critic

Original Actor-critic



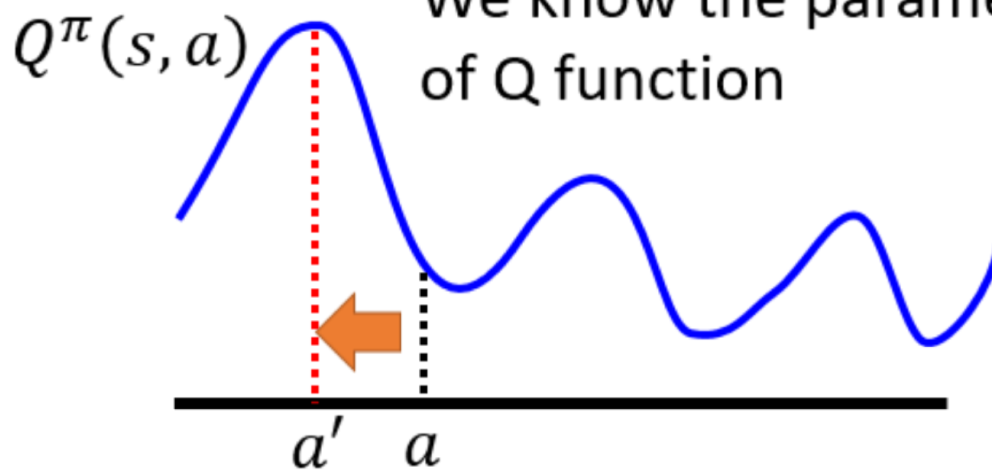
Another Way to use Critic

Original Actor-critic



Pathwise derivative policy gradient

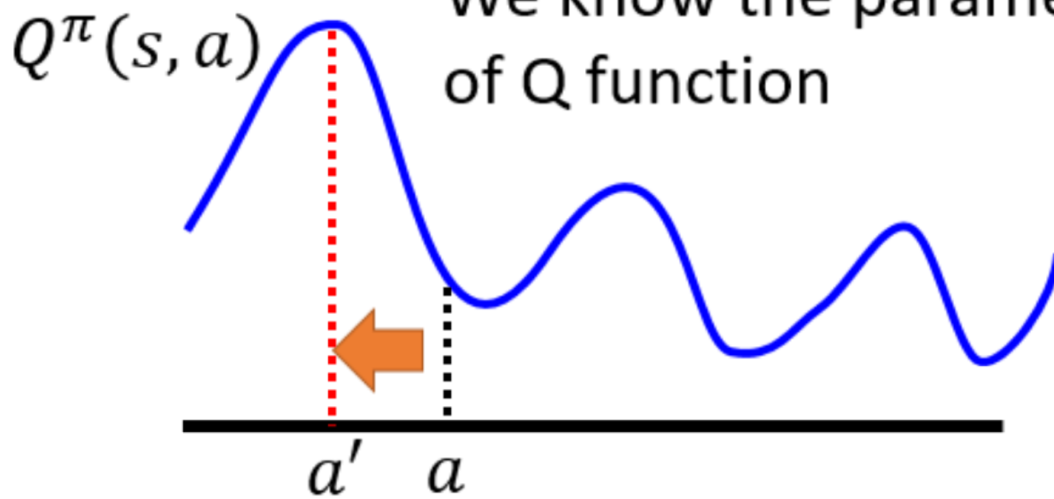
We know the parameters of Q function



From Q function we know that taking a' at state s is better than a

Pathwise derivative policy gradient

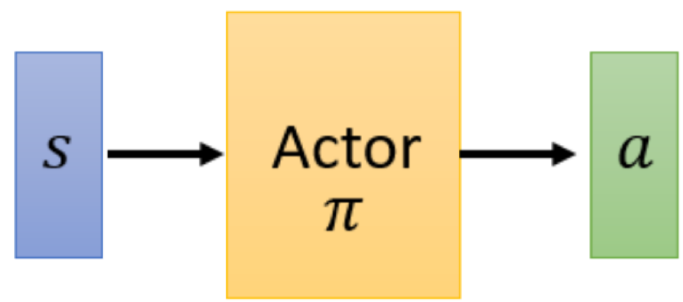
From Q function we know that taking a’ at state s is better than a



We know the parameters of Q function

Action a is a *continuous vector*

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



Actor as the solver of this optimization problem

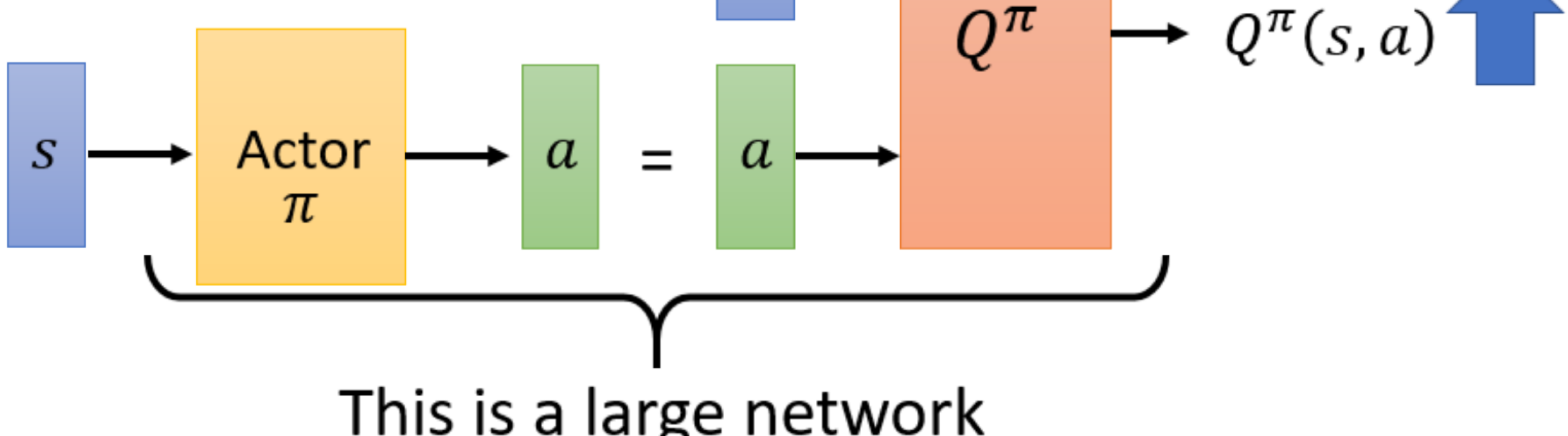
Pathwise Derivative Policy Gradient

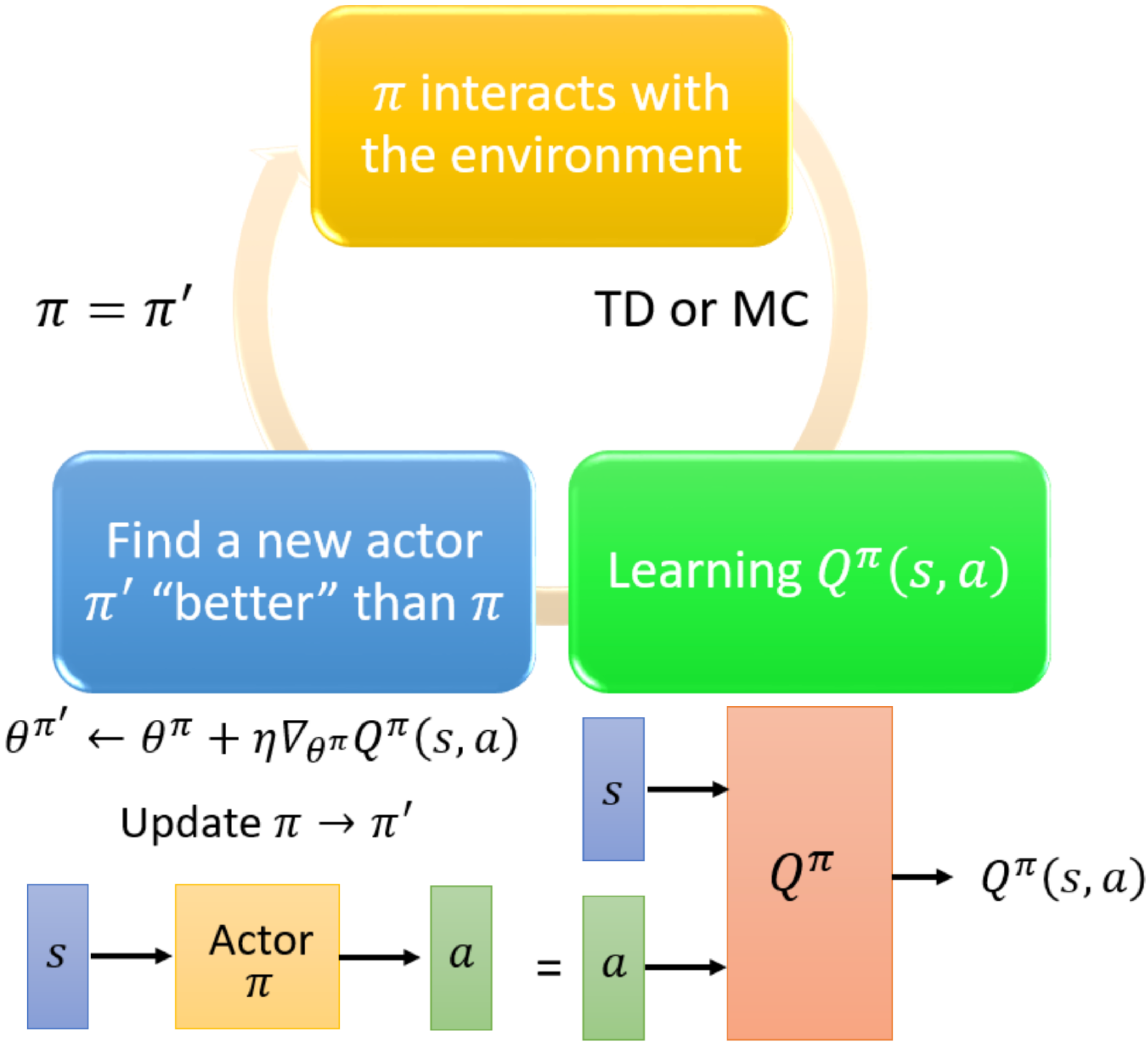
$$\pi'(s) = arg\max_a Q^\pi(s,a)$$ a is the output of an actor

Gradient ascent:

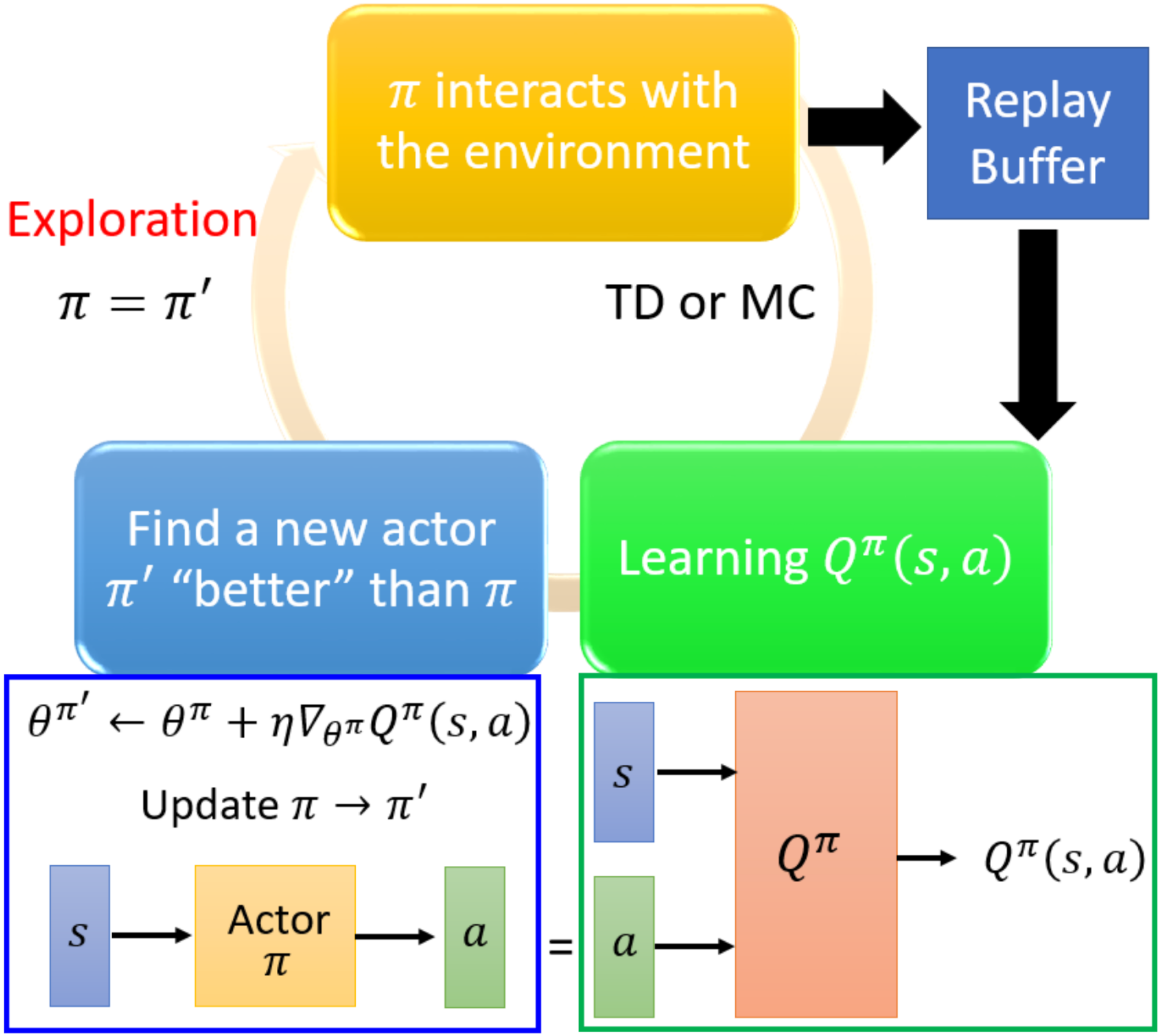
$$\theta^{\pi'} \leftarrow \theta^\pi + \eta \nabla_{\theta^\pi} Q^\pi(s,a)$$

Update $\pi \rightarrow \pi'$





π interacts with the environment
TD or MC
Learning $Q^{\pi}(s,a)$
Find a new actor π' "better" than π
$\pi = \pi'$
$\theta^{\pi'} \leftarrow \theta^{\pi} + \eta \nabla_{\theta^{\pi}} Q^{\pi}(s,a)$
Update $\pi \rightarrow \pi'$
s
Actor π
a
=
a
s
Q^{π}
$Q^{\pi}(s,a)$


π interacts with the environment
Replay Buffer
Exploration
$\pi = \pi'$
TD or MC
Find a new actor π' "better" than π
Learning $Q^\pi(s,a)$
$\theta^{\pi'} \leftarrow \theta^\pi + \eta \nabla_{\theta^\pi} Q^\pi(s,a)$
Update $\pi \rightarrow \pi'$
s
Actor π
a
=
s
a
Q^π
$Q^\pi(s,a)$

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 (exploration)
    - 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 $\hat{Q} = Q$

Q-Learning Algorithm ➡ *Pathwise Derivative Policy Gradient*

- Initialize Q-function Q, target Q-function $\hat{Q} = Q$, actor π, target actor $\hat{\pi} = \pi$

Replaced ε-greedy policy with π network.

- In each episode
  - For each time step t
    1. - Given state s_t, take action a_t based on ~~Q~~ π (exploration)
    - 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)
    2. - Target $y = r_i + \cancel{\max_a \hat{Q}(s_{i+1}, a)}\ \hat{Q}\left(s_{i+1}, \hat{\pi}(s_{i+1})\right)$
    - Update the parameters of Q to make $Q(s_i, a_i)$ close to y (regression)
    3. - Update the parameters of π to maximize $Q\left(s_i, \pi(s_i)\right)$
    - Every C steps reset $\hat{Q} = Q$
    4. - Every C steps reset $\hat{\pi} = \pi$

<table>
<tr><th></th><th>Reinforcement Learning</th><th>Inverse Reinforcement Learning</th></tr>
<tr><td rowspan="4">Single Agent</td><td>Tabular representation of reward
Model-based control
Model-free control
(MC, SARSA, Q-Learning)</td><td rowspan="2">Linear reward function learning
Imitation learning
Apprenticeship learning
Inverse reinforcement learning
MaxEnt IRL
MaxCausalEnt IRL
MaxRelEnt IRL</td></tr>
<tr><td rowspan="2">Function representation of reward
1. Linear value function approx
(MC, SARSA, Q-Learning)
2. Value function approximation
(Deep Q-Learning, Double DQN, prioritized DQN, Dueling DQN)
3. Policy function approximation
(Policy gradient, PPO, TRPO)
4. Actor-Critic methods (A2C, A3C, Pathwise Derivative PG)</td></tr>
<tr><td>Non-linear reward function learning
Generative adversarial imitation learning (GAIL)

Adversarial inverse reinforcement learning (AIRL)</td></tr>
<tr><td>Review of Deep Learning
As bases for non-linear function approximation (used in 2-4).</td><td>Review of Generative Adversarial nets</td></tr>
<tr><td>Multiple Agents</td><td>Multi-Agent Reinforcement Learning
Multi-agent Actor-Critic
etc.</td><td>Multi-Agent Inverse Reinforcement Learning
MA-GAIL
MA-AIRL
AMA-GAIL</td></tr>
</table>

Applications

Questions?