Actor-Critic

Hung-yi Lee

Asynchronous Advantage Actor-Critic (A3C)

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, “Asynchronous Methods for Deep Reinforcement Learning”, ICML, 2016

Review – Policy Gradient

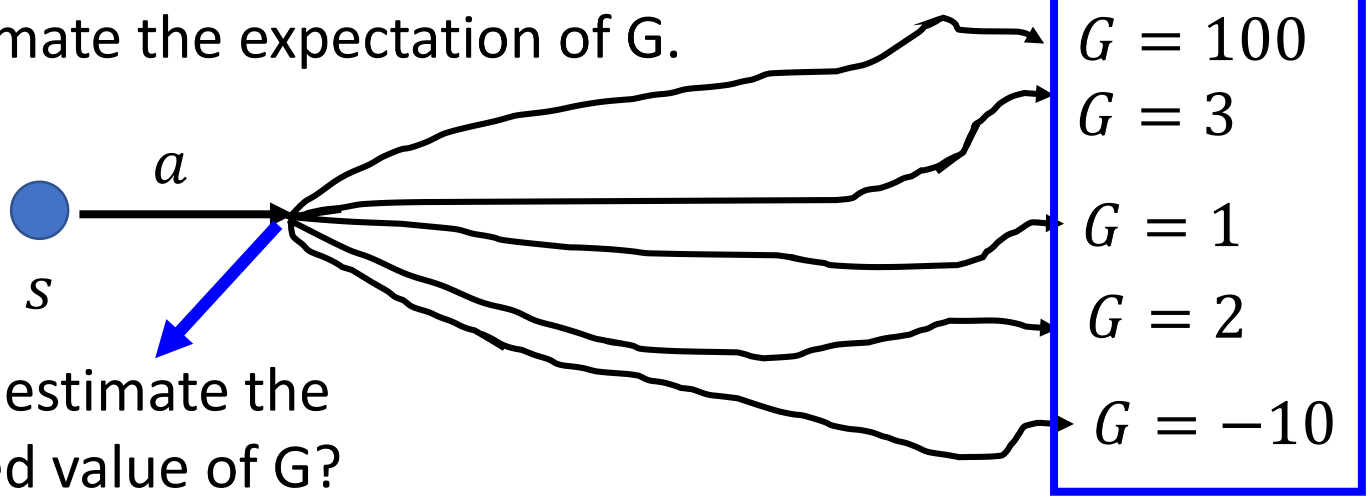
baseline

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

G_t^n : obtained via interaction

Very unstable

With sufficient samples, approximate the expectation of G.

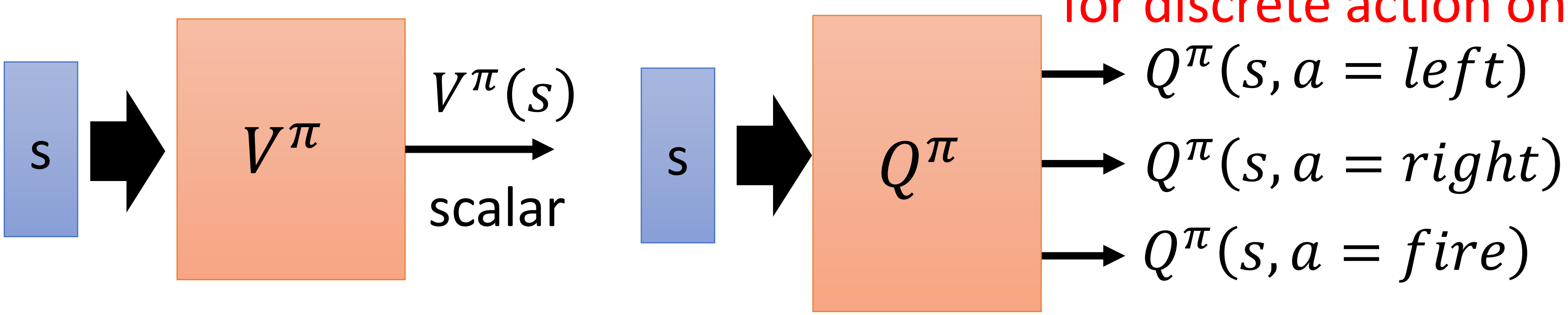


Can we estimate the expected value of G?

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

for discrete action only



Estimated by TD or MC

Actor-Critic

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

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

baseline

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

G_t^n : obtained via interaction

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

Advantage Actor-Critic

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

Estimate two networks? We can only estimate one.

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

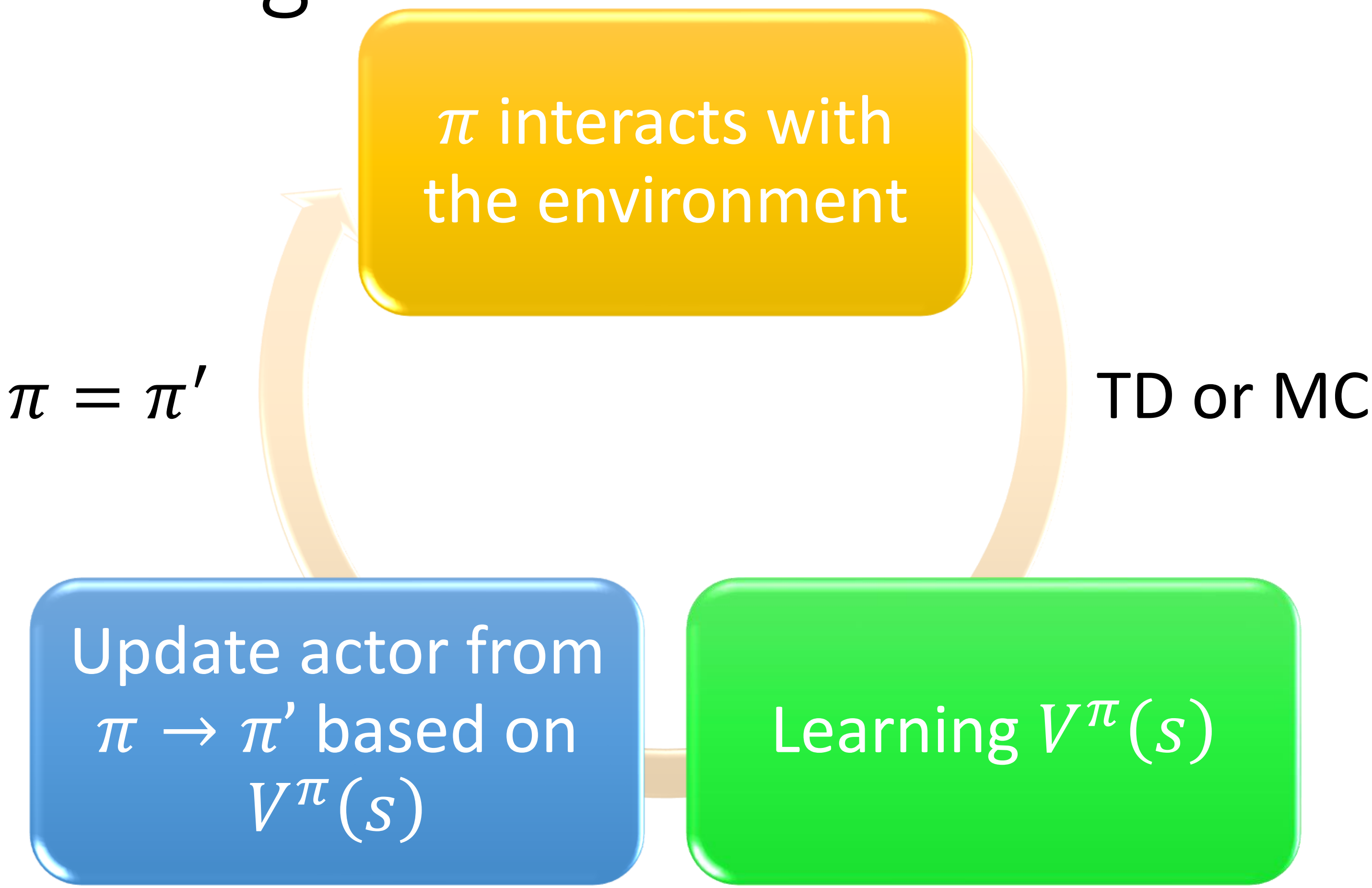
Only estimate state value

A little bit variance

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

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

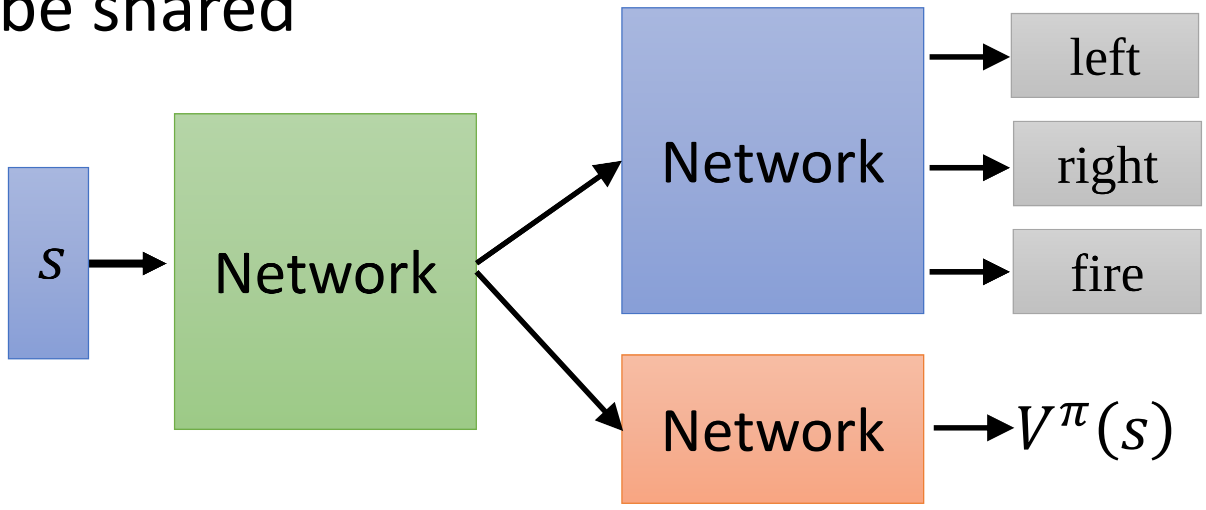
Advantage Actor-Critic



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

Advantage Actor-Critic

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



  - Use output entropy as regularization for $\pi(s)$
    - Larger entropy is preferred → exploration

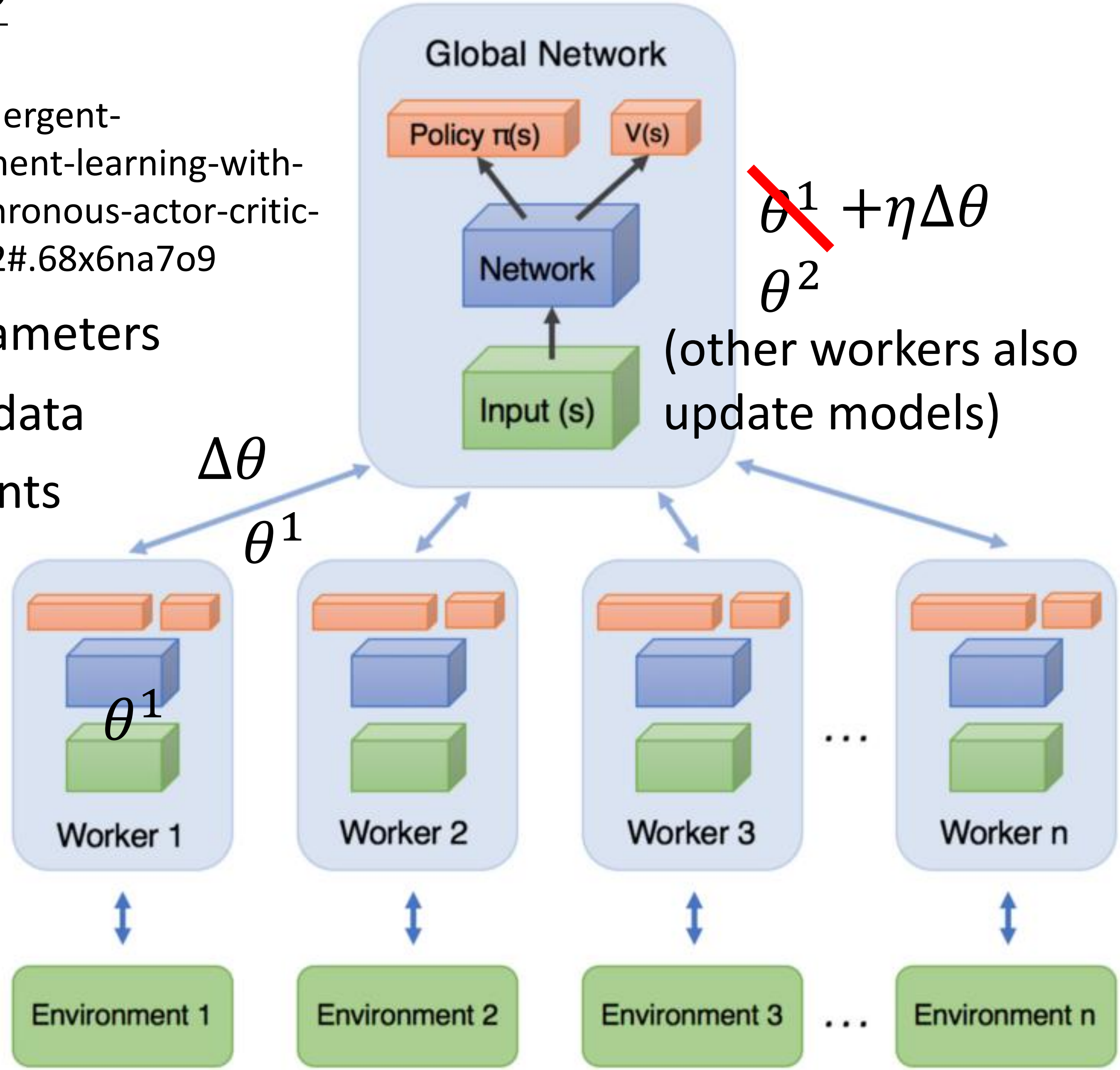
Asynchronous Advantage Actor-Critic (A3C)

The idea is from 李思叡

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

$\Delta\theta$

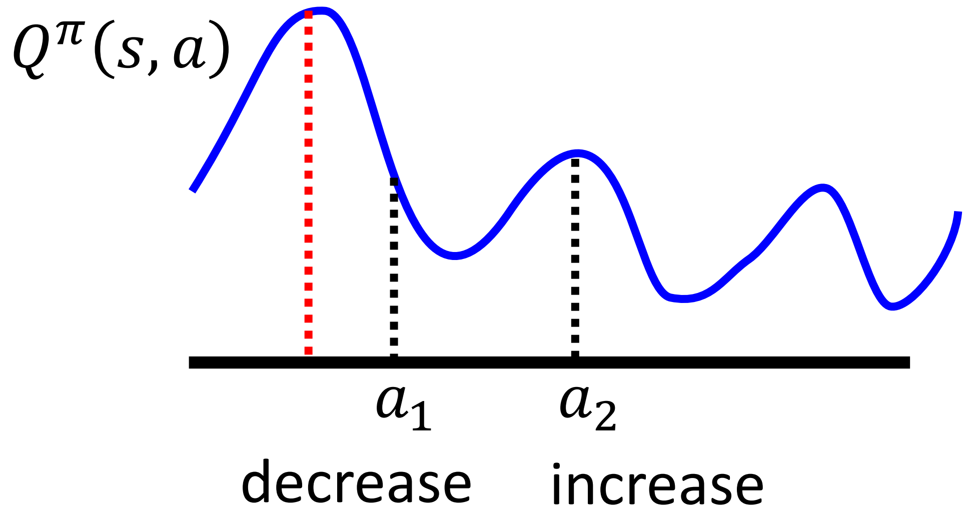
Pathwise Derivative Policy Gradient

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

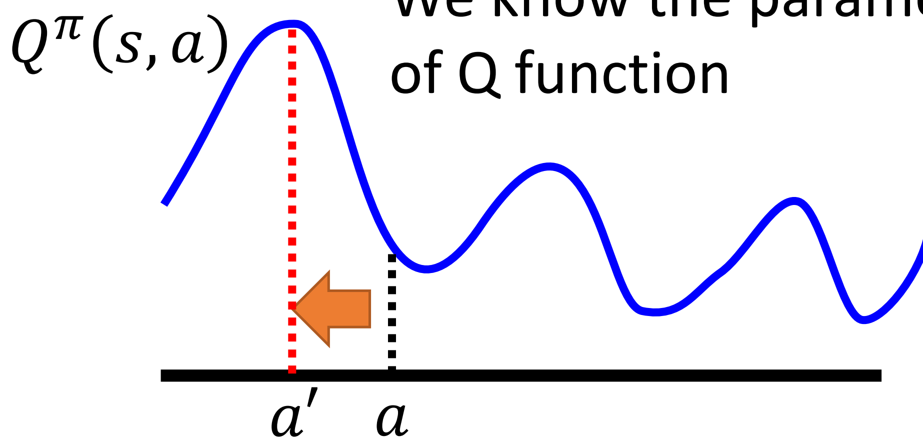
Original Actor-critic



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



Actor

Critic

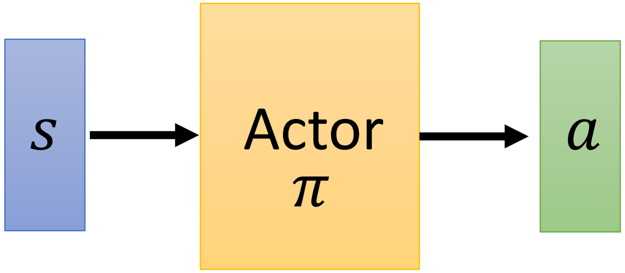
Pathwise derivative policy gradient

Original Actor-critic

http://www.cartomad.com/comic/109000081104011.html

Action a is a *continuous vector*

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



Actor as the solver of this optimization problem

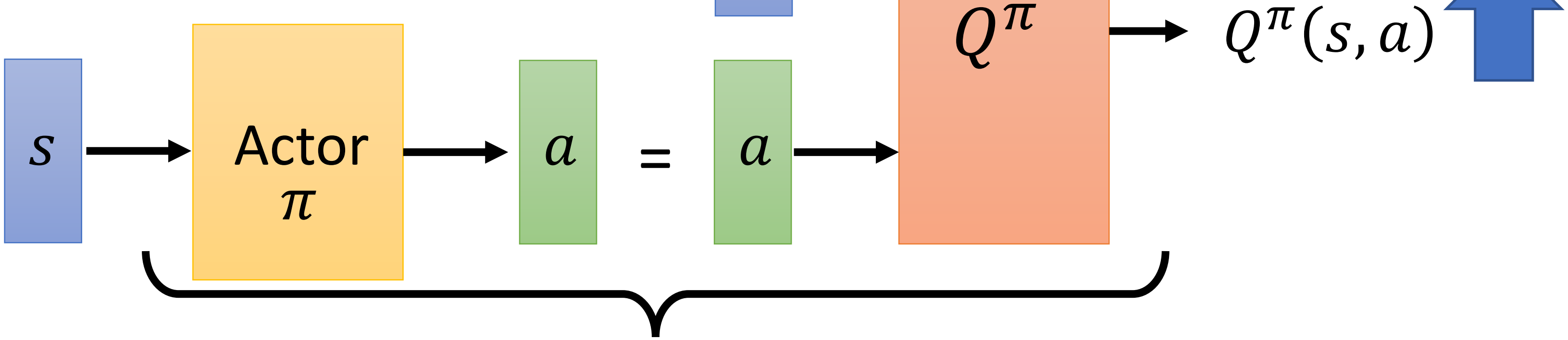
Pathwise Derivative Policy Gradient

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

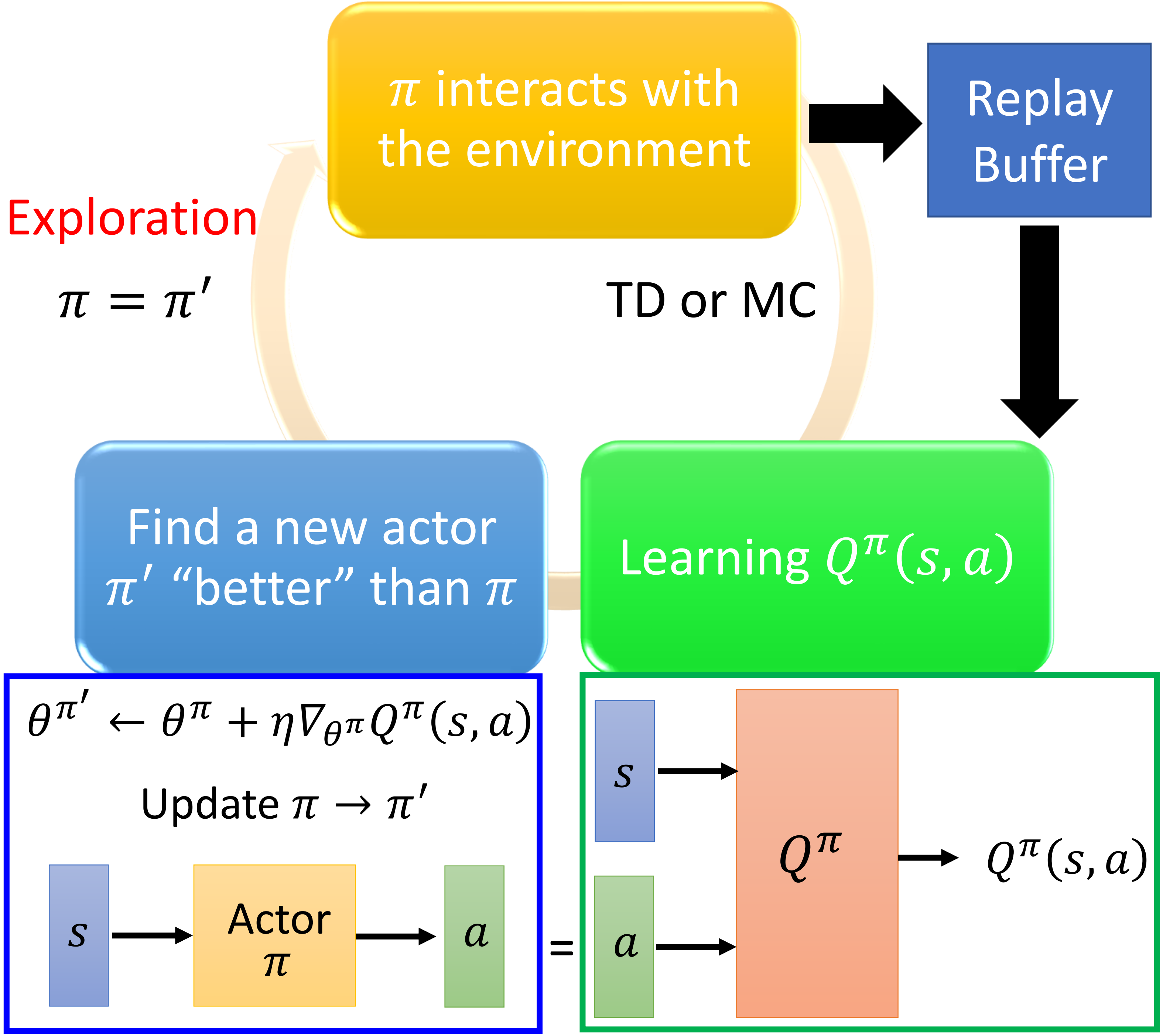
Gradient ascent:

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

Update $\pi \rightarrow \pi'$



This is a large network



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 $\hat{\pi} = \pi$
- 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 $\hat{\pi} = \pi$

Connection with GAN

Method	GANs	AC
Freezing learning	yes	yes
Label smoothing	yes	no
Historical averaging	yes	no
Minibatch discrimination	yes	no
Batch normalization	yes	yes
Target networks	n/a	yes
Replay buffers	no	yes
Entropy regularization	no	yes
Compatibility	no	yes

David Pfau, Oriol Vinyals, “Connecting Generative Adversarial Networks and Actor-Critic Methods”, arXiv preprint, 2016