This lecture will be recorded!!!

Welcome to

DS595/CS525
Reinforcement Learning

Prof. Yanhua Li

Time: 6:00pm –8:50pm R
Zoom Lecture
Fall 2020

Quiz 5 Today

- 30 minutes on Policy Gradient (PG)

Project 3 leader board

Leaderboard for Breakout-DQN Update Date: 11/11/2020 16:00

Top	Date	Name	Score
1	11/10/2020	Akshay Sadanandan	222
2	11/5/2020	Apiwat Ditthapron	194.5
3	10/24/2020	Daniel Jeswin Nallathambi	169.04
4	11/11/2020	Songlin Hou	77.22
5	11/10/2020	Jinyang Wang	75.14
6	11/11/2020	Oscar Garcia Fernandez	60.36
7	11/11/2020	Ran Jing	51.2

Project 3 deadline is extended for 1 week, to 11/19/2020

Based on all your feedback, see below, we extend project 3 deadline for 1 week to 11/19/2020.

no extension (still 11/12/2020)	5 respondents	15 %	
3 days of extension to Sunday 11/15/2020	8 respondents	24 %	
1 week of extension to (Thursday 11/19/2020)	19 respondents	58 %	
No Answer	1 respondents	3 %	

Project 4 is available
Starts 10/29 Thursday

- https://github.com/yingxue-zhang/DS595CS525-RL-Projects/tree/master/Project4
- *Important Dates:*
- Project Proposal: Thursday 11/12/2020
- Progressive report: Thursday 11/26/2020
- Final Project:
  - Tuesday 12/8/2020 team project report is due
  - Thursday 12/10/2020 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**
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

I don't have candies for you today, but algorithms☺

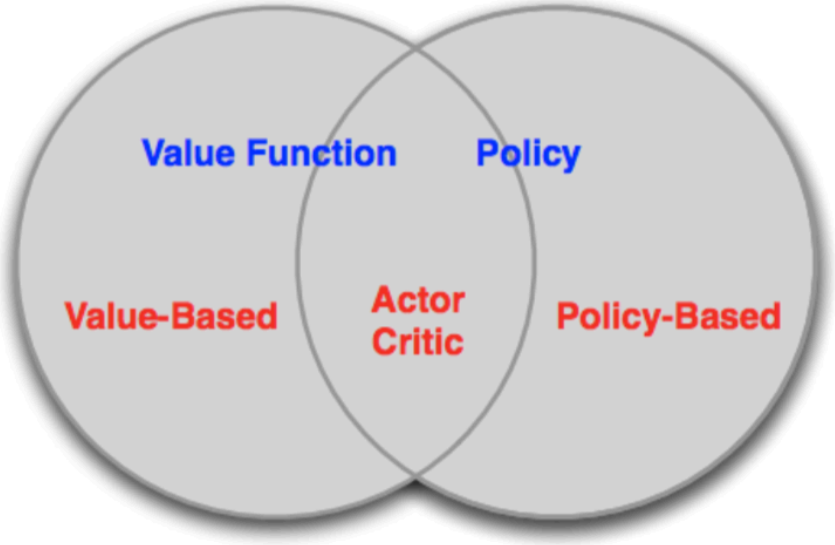
Value-Based and Policy-Based RL

Model-Free RL:

Explicit: Value function and/or policy function

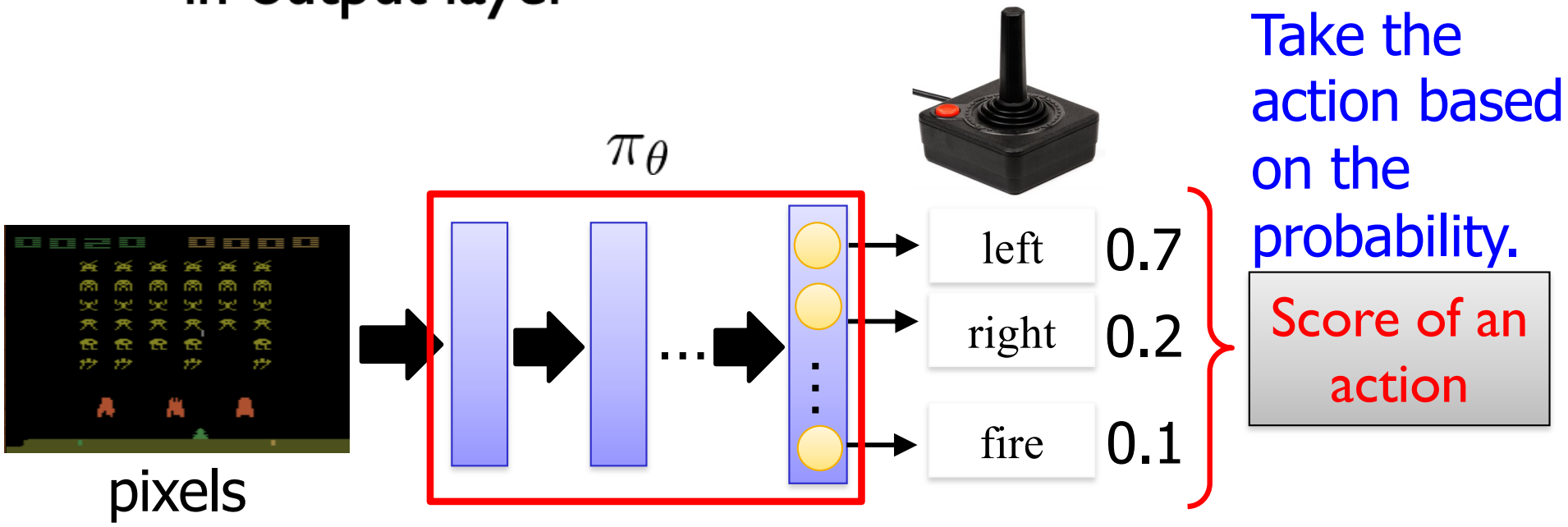
No model

- Value Based
  - Learnt Value Function
  - Implicit policy (e.g. ϵ-greedy)
- Policy Based
  - No Value Function
  - Learnt Policy
- Actor-Critic
  - Learnt Value Function
  - Learnt Policy



Policy of Actor

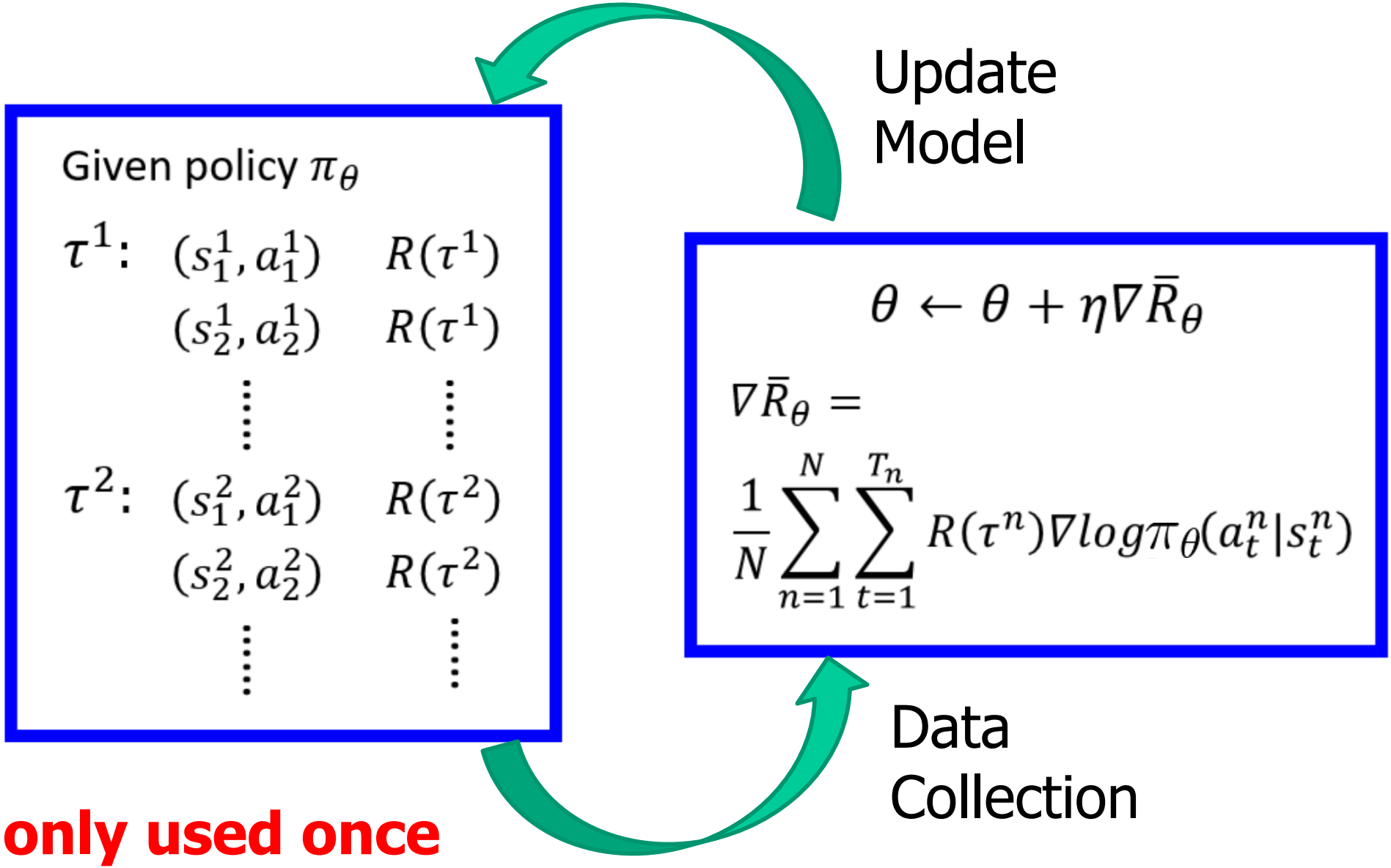
- Policy π is a network with parameter $\theta \longrightarrow \pi_\theta$
  - Input: the observation of machine represented as a vector or a matrix
  - Output: each action corresponds to a neuron in output layer



Quick Review

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

Basic Policy Gradient Algorithm



only used once

Quick Review

From basic PG algorithm to…

- Issues with the basic PG algorithm
  - TIP 1. Inaccurate update when non-negative rewards
    - Add baselines at states
  - TIP 2. Large variance
    - Assign suitable credits
    - REINFORCE and Vanilla Policy Gradient
  - TIP 3. Slow, due to the un-reusable data collection process
    - Use importance sample to reuse data when training: PPO, TRPO, PPO2

Monte-Carlo Policy Gradient (REINFORCE) Quick Review

TIP #2: Assign Suitable Credit by using returns

- Leverages likelihood ratio / score function and temporal structure

$$\Delta\theta_t = \eta \nabla_\theta \log \pi_\theta(s_t, a_t) G_t \tag{7}$$

REINFORCE:

Initialize policy parameters θ arbitrarily

for each episode $\{s_1, a_1, r_2, \cdots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

  for $t = 1$ to $T - 1$ **do**

    $\theta \leftarrow \theta + \eta \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$

  endfor

endfor

return θ

"Vanilla" Policy Gradient Algorithm

Quick Review

Using both TIP #1 & #2

The simplest way to implement it is using average return of a state s_t: $b(s_t) \approx \mathbb{E}[r_t + r_{t+1} + \cdots + r_{T-1}]$

Initialize policy parameter θ, baseline b

for iteration=$1, 2, \cdots$ **do**

Collect a set of trajectories by executing the current policy π_θ

At each timestep in each trajectory, compute

the *return* $G_t^n = \sum_{t'=t}^{T-1} r_{t'}$, and

the *advantage estimate* $\hat{A}_t = G_t^n - b(s_t)$.

Re-fit the baseline, by minimizing $||b(s_t) - R_t||^2$,

summed over all trajectories and timesteps.

Update the policy, using a policy gradient estimate $\nabla \bar{R}_\theta$

which is a sum of terms $\nabla_\theta \log \pi_\theta(a_t|s_t, \theta)\hat{A}_t$.

(Plug $\nabla \bar{R}_\theta$ to SGD or ADAM)

endfor

$$\nabla \bar{R}_\theta \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} (G_t^n - b(s_t)) \nabla_\theta log \pi_\theta(a_t^n | s_t^n)$$

TIP #3: Importance Sampling + Constraints

- TIP 3. Slow, due to the un-reusable data collection process
  - Relook at
    - Basic PG,
    - REINFORCE PG
    - Vanilla PG

From on-policy to off-policy

Using the experience more than once

On-policy v.s. Off-policy

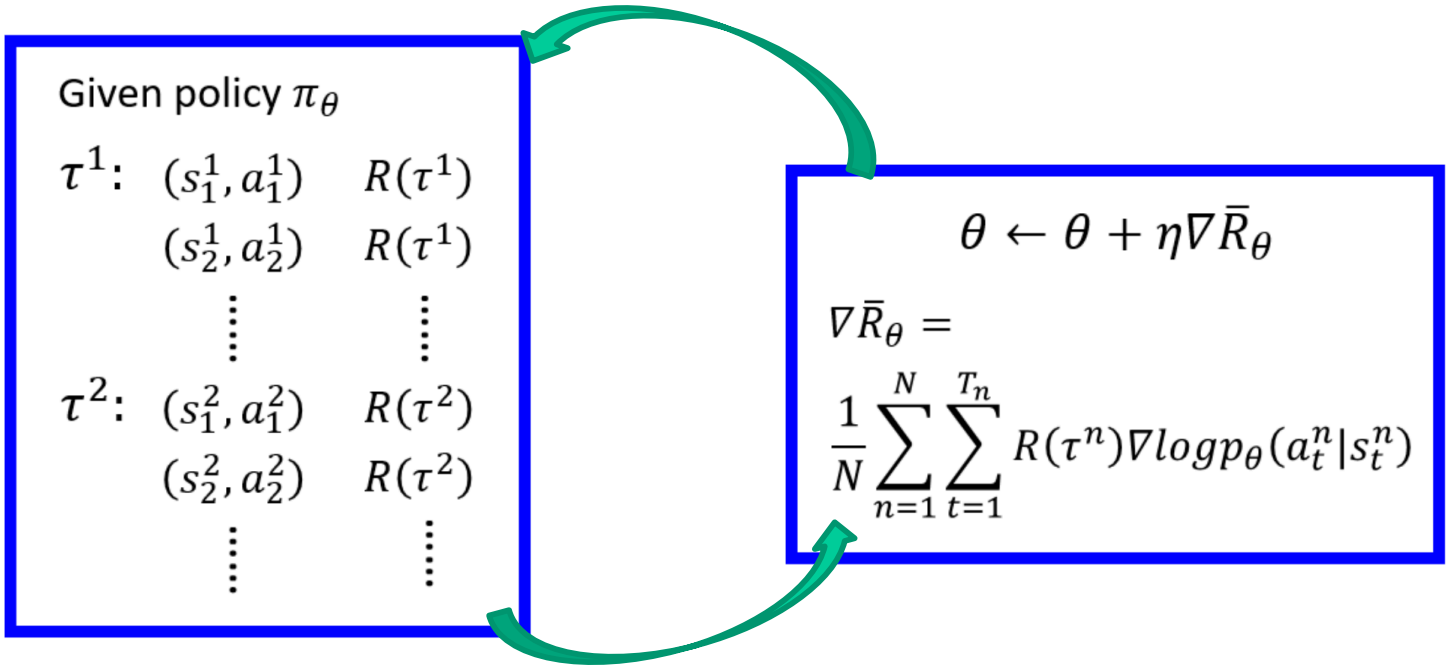
?

- On-policy: The agent learned and the agent interacting with the environment is the same.
- Off-policy: The agent learned and the agent interacting with the environment is different.

On-policy → Off-policy

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.



Hope to use the data to update θ multiple times before collecting new data.

On-policy → Off-policy

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)] = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla log p_\theta(a_t^n|s_t^n)$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

Importance Sampling

$$E_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i)$$

x^i is sampled from $p(x)$

We only have x^i sampled from $q(x)$

On-policy → Off-policy

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)] = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla log p_\theta(a_t^n|s_t^n)$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

Importance Sampling

$$E_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i)$$

x^i is sampled from $p(x)$

We only have x^i sampled from $q(x)$

$$= \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

Importance weight

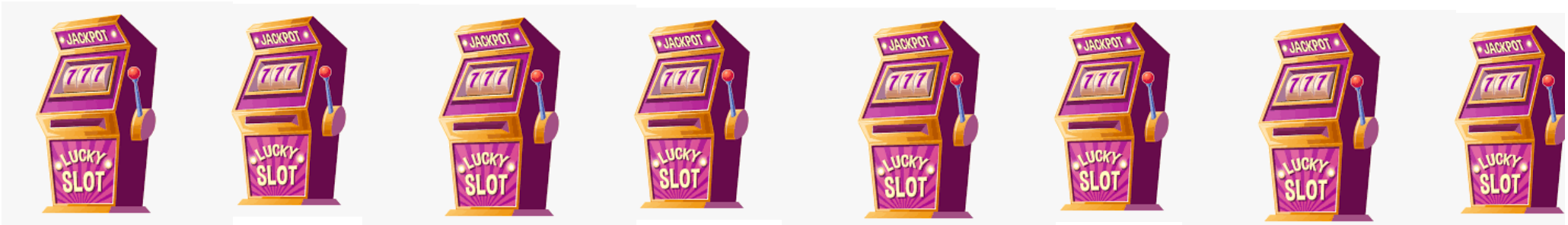
Importance Sampling

x^i is sampled from $p(x)$

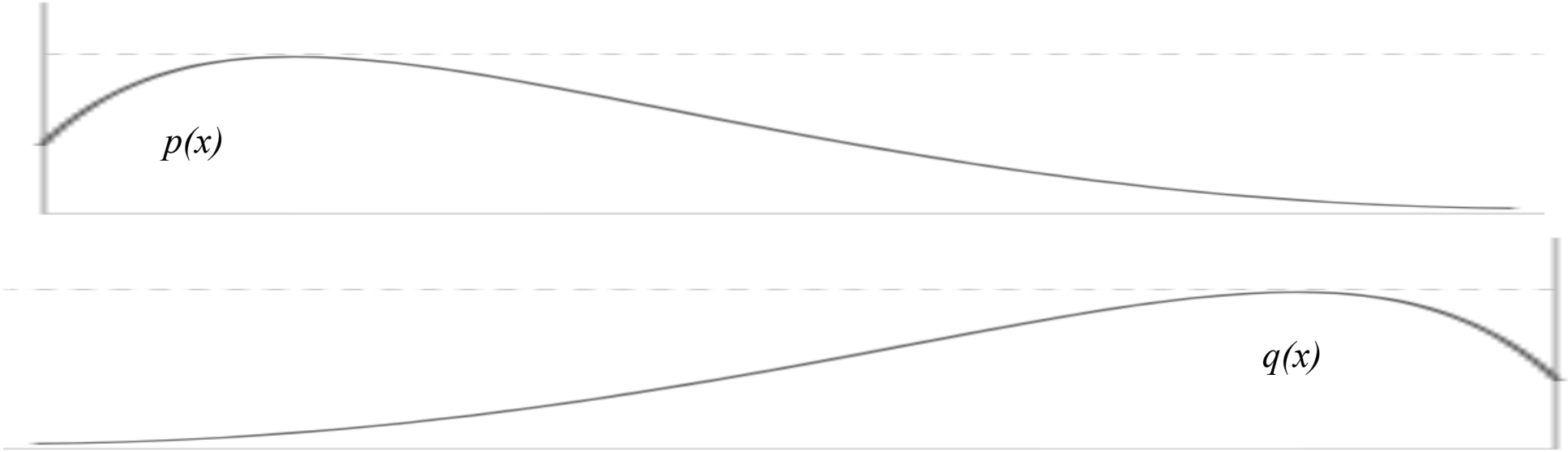
$$E_{x\sim p}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^i)$$

We only have x^i sampled from $q(x)$

$$= \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$



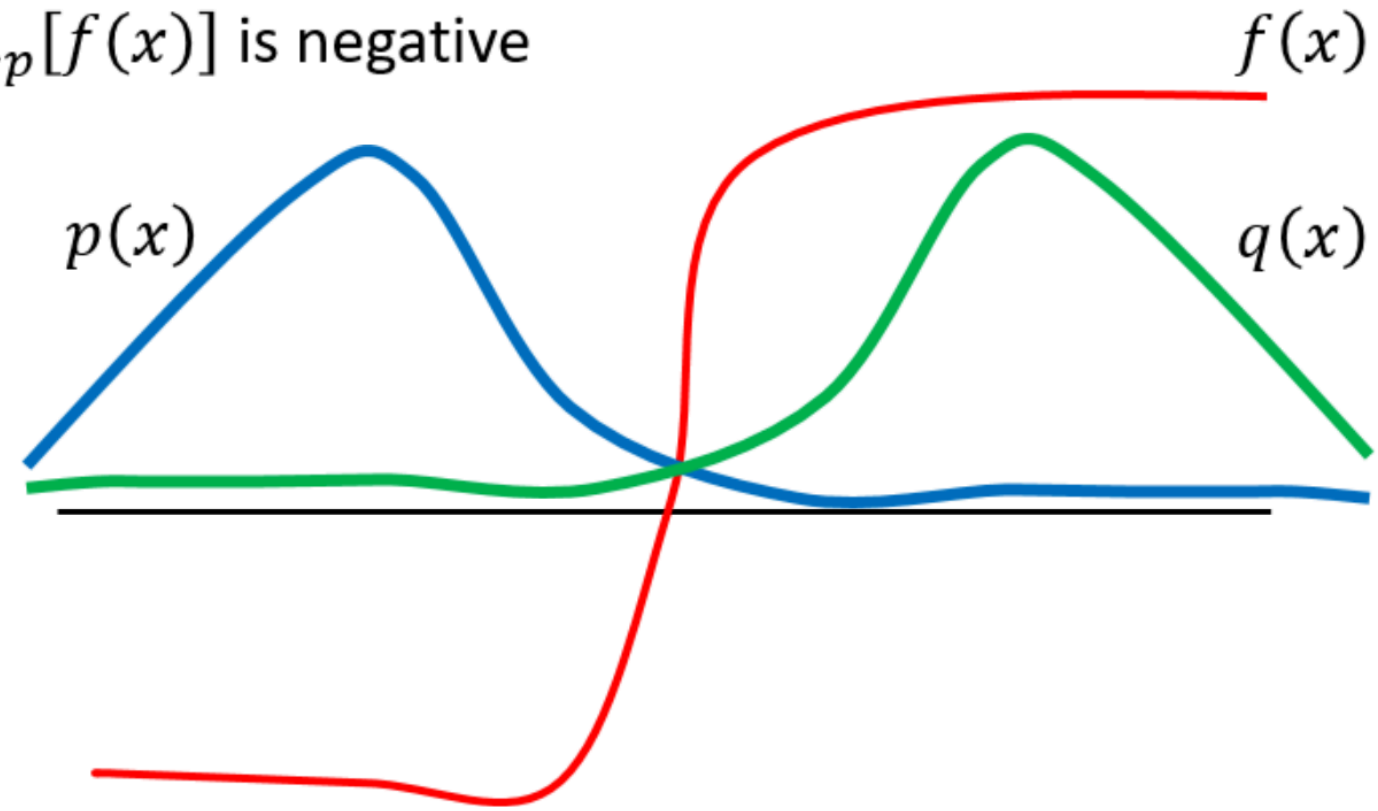
Low reward rates ----------→ high reward rates



Issue of Importance Sampling

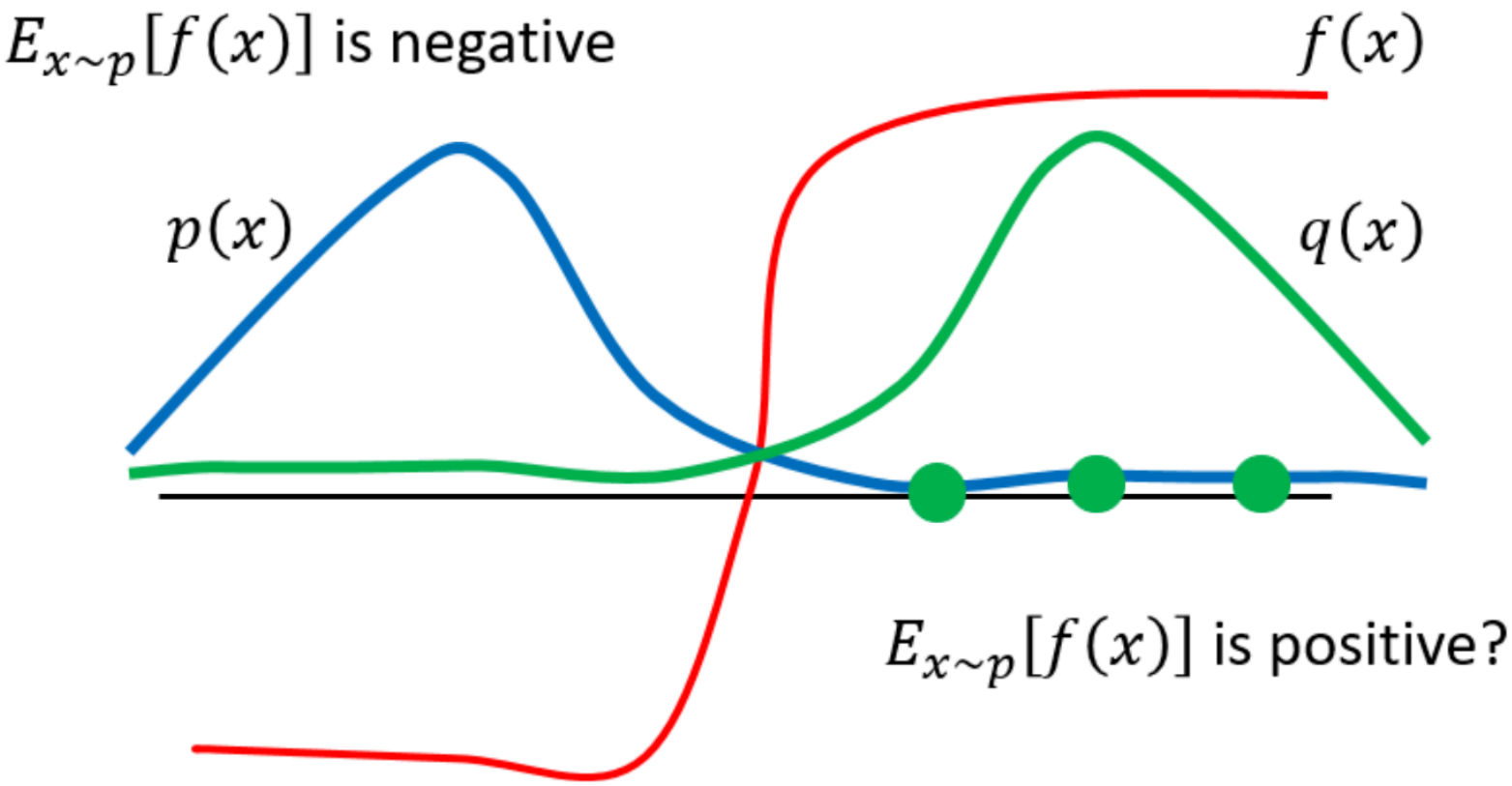
$$E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

$E_{x\sim p}[f(x)]$ is negative



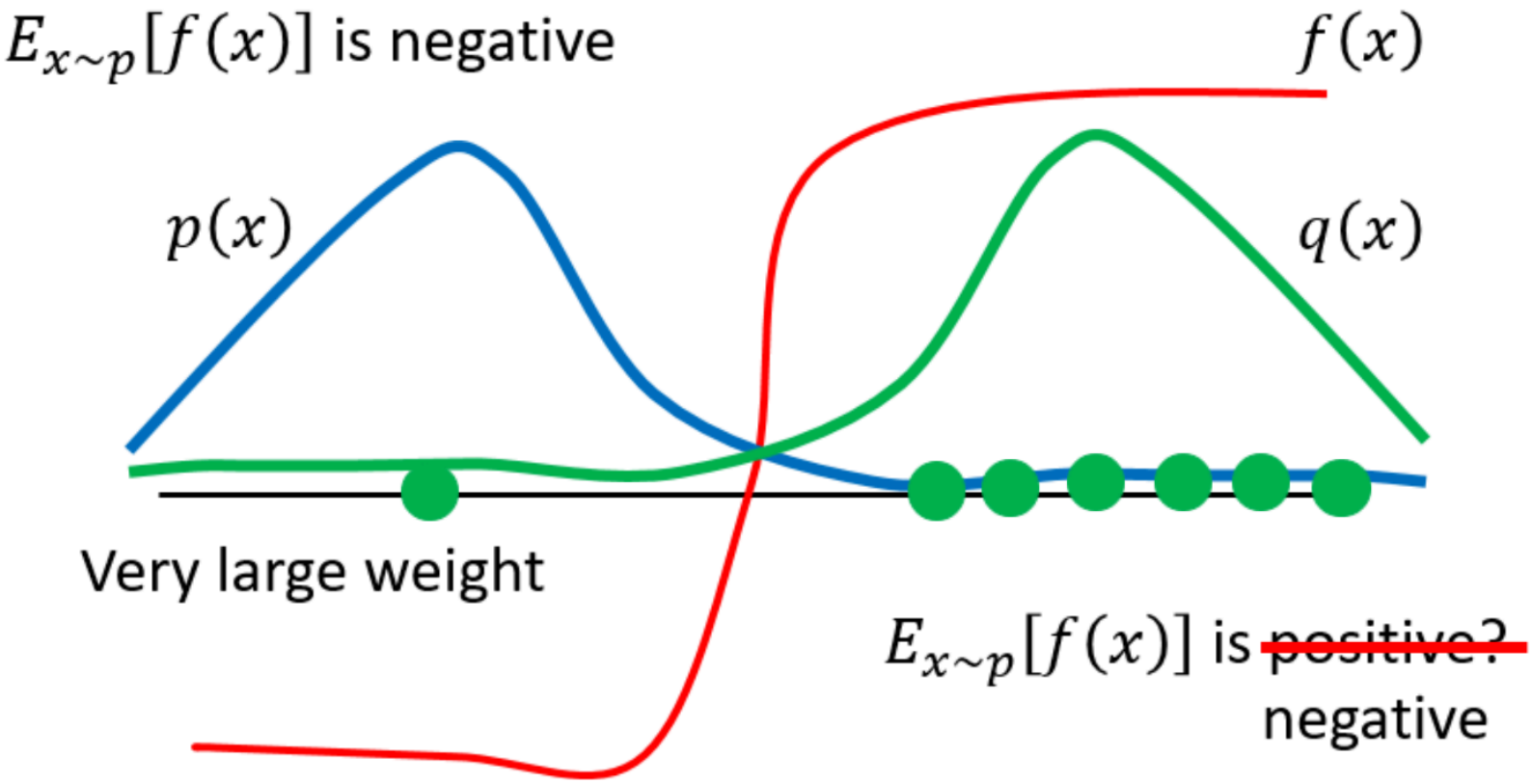
Issue of Importance Sampling

$$E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$



Issue of Importance Sampling

$$E_{x\sim p}[f(x)] = E_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$



On-policy → Off-policy

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

- Use π_θ to collect data. When θ is updated, we have to sample training data again.
- Goal: Using the sample from $\pi_{\theta'}$ to train θ. θ' is fixed, so we can re-use the sample data.

$$\nabla \bar{R}_\theta = E_{\tau \sim p_{\theta'}(\tau)}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)} R(\tau)\nabla log p_\theta(\tau)\right]$$

Basic PG

- Sample the data from θ'.
- Use the data to train θ many times.

Importance Sampling

$$E_{x \sim p}[f(x)] = E_{x \sim q}[f(x)\frac{p(x)}{q(x)}]$$

On-policy → Off-policy

Gradient for update

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$= E_{(s_t,a_t)\sim\pi_\theta}[A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{P_\theta(s_t,a_t)}{P_{\theta'}(s_t,a_t)}A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

On-policy → Off-policy

Gradient for update

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$= E_{(s_t,a_t)\sim\pi_\theta}[A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

$A^{\theta'}(s_t,a_t)$ This term is from sampled data.

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{P_\theta(s_t,a_t)}{P_{\theta'}(s_t,a_t)}A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

On-policy → Off-policy

Gradient for update

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$= E_{(s_t,a_t)\sim\pi_\theta}[A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

$A^{\theta'}(s_t,a_t)$ This term is from sampled data.

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{P_\theta(s_t,a_t)}{P_{\theta'}(s_t,a_t)} A^{\theta'}(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}\frac{p_\theta(s_t)}{p_{\theta'}(s_t)} A^{\theta'}(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

On-policy → Off-policy

Gradient for update

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$= E_{(s_t,a_t)\sim\pi_\theta}[A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

$A^{\theta'}(s_t,a_t)$

This term is from sampled data.

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{P_\theta(s_t,a_t)}{P_{\theta'}(s_t,a_t)} A^\theta(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

$$= E_{(s_t,a_t)\sim\pi_{\theta'}}[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} \frac{p_\theta(s_t)}{p_{\theta'}(s_t)} A^{\theta'}(s_t,a_t)\nabla log\pi_\theta(a_t^n|s_t^n)]$$

Why? $J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} A^{\theta'}(s_t,a_t)\right]$ When to stop?

Add Constraints

RL — The Math behind TRPO & PPO
https://medium.com/@jonathan_hui/rl-the-math-behind-trpo-ppo-d12f6c745f33

TRPO paper:
https://arxiv.org/pdf/1502.05477.pdf

PPO paper:
https://arxiv.org/pdf/1707.06347.pdf

PPO / TRPO

Proximal Policy Optimization (PPO)

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$J_{PPO}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\theta, \theta')$$

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t)\right]$$

PPO / TRPO

θ cannot be very different from θ'

Constraint on behavior not parameters

Proximal Policy Optimization (PPO) (2017)

$$\nabla f(x) = f(x)\nabla log f(x)$$

$$J_{PPO}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta KL(\theta, \theta')$$

$$J^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t)\right]$$

TRPO (Trust Region Policy Optimization) (2015)

$$J_{TRPO}^{\theta'}(\theta) = E_{(s_t,a_t)\sim\pi_{\theta'}}\left[\frac{p_\theta(a_t|s_t)}{p_{\theta'}(a_t|s_t)} A^{\theta'}(s_t, a_t)\right]$$

$$KL(\theta, \theta') < \delta$$

PPO algorithm

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t,a_t)$$

- Initial policy parameters θ^0
- In each iteration
  - Using θ^k to interact with the environment to collect $\{s_t, a_t\}$ and compute advantage $A^{\theta^k}(s_t,a_t)$
  - Find θ optimizing $J_{PPO}(\theta)$

$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

Update parameters several times

- If $KL(\theta,\theta^k) > KL_{max}$, increase β
- If $KL(\theta,\theta^k) < KL_{min}$, decrease β

Adaptive KL Penalty

PPO algorithm

$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

$$J^{\theta^k}(\theta) \approx \sum_{(s_t, a_t)} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t, a_t)} min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t), \right.$$

$$\left. clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon \right) A^{\theta^k}(s_t, a_t) \right)$$

PPO algorithm

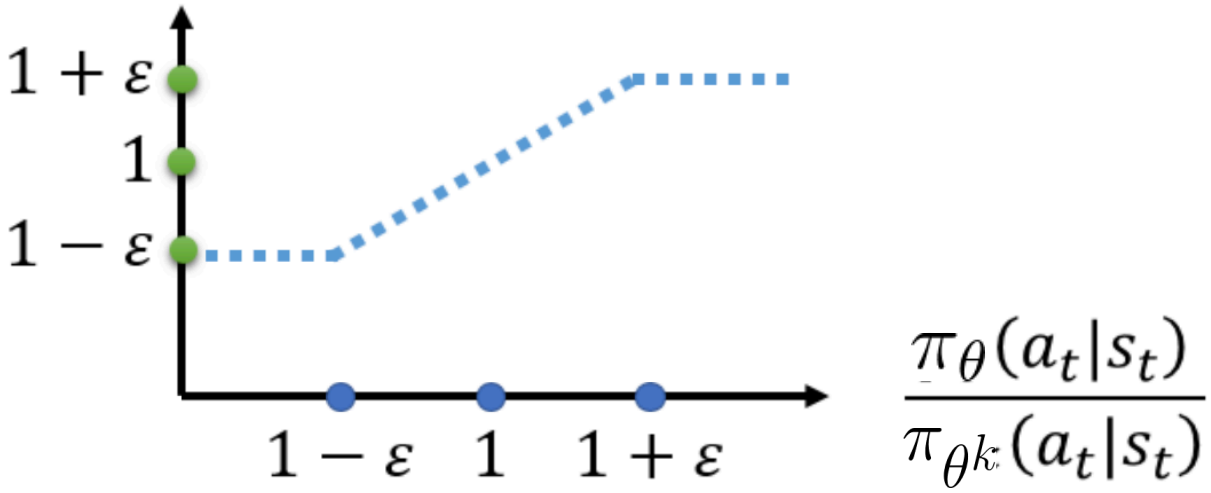
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

$$J^{\theta^k}(\theta) \approx \sum_{(s_t, a_t)} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t, a_t)}$$

$$clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t, a_t)$$



PPO algorithm

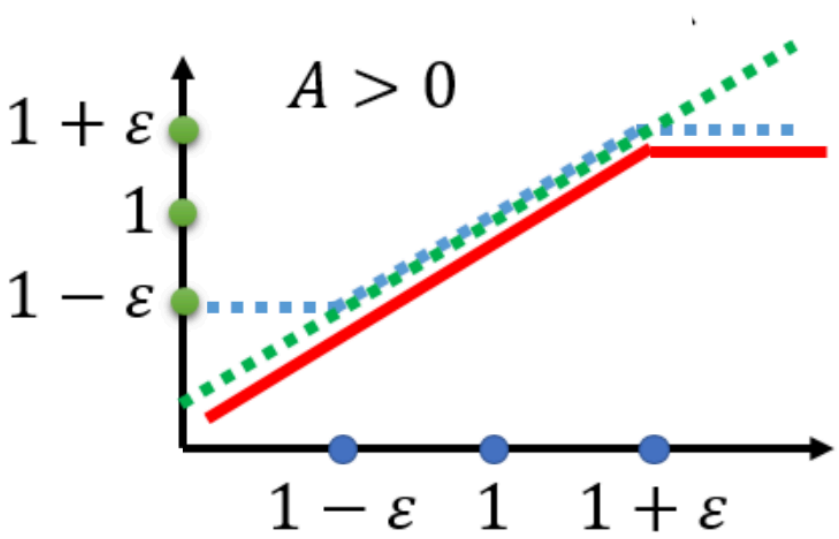
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

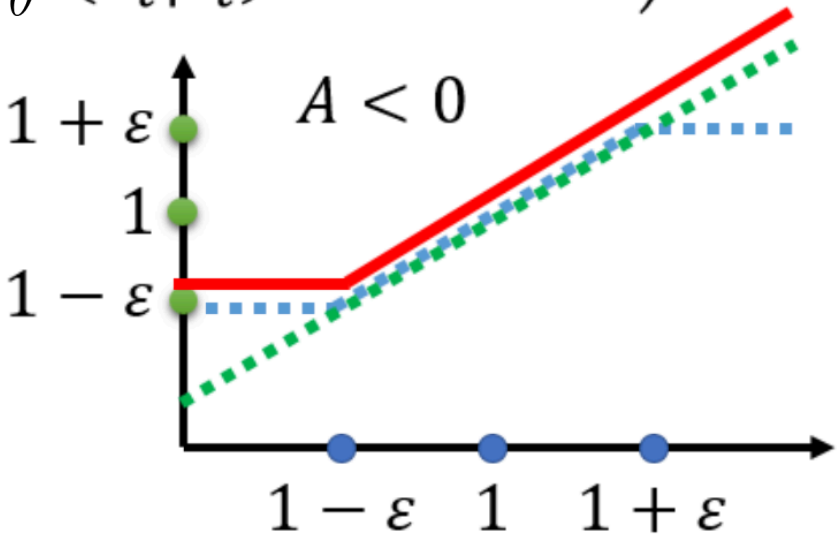
$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t)$$

PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t),\right.$$

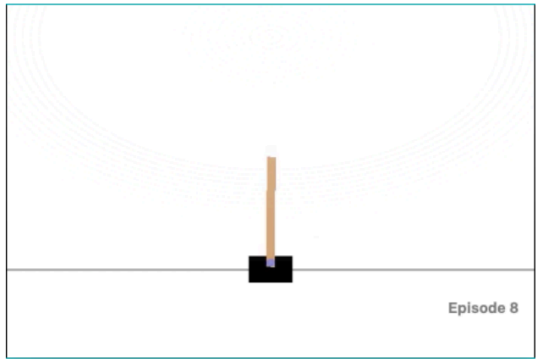
$$\left. clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right) A^{\theta^k}(s_t, a_t)\right)$$



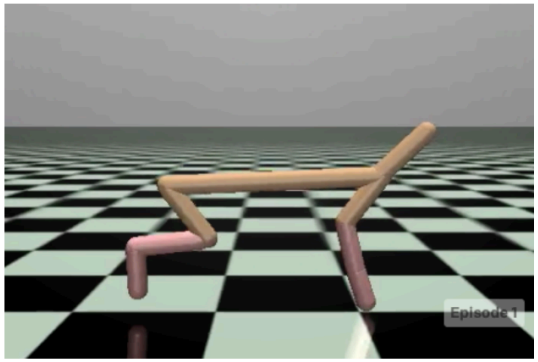


Experimental Results (with MuJoCo Tasks)

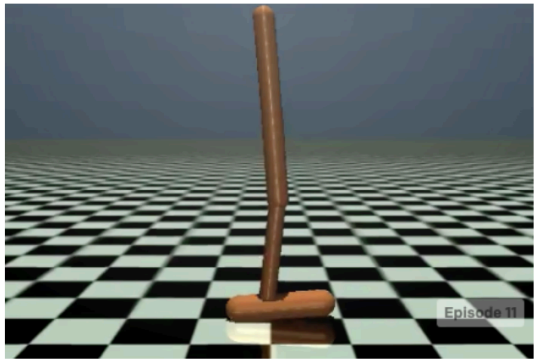
https://arxiv.org/abs/1707.06347



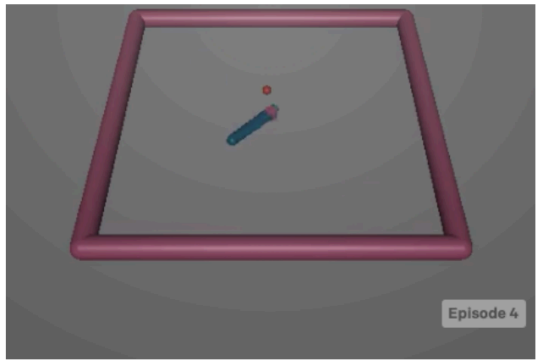
(a) CartPole-v0



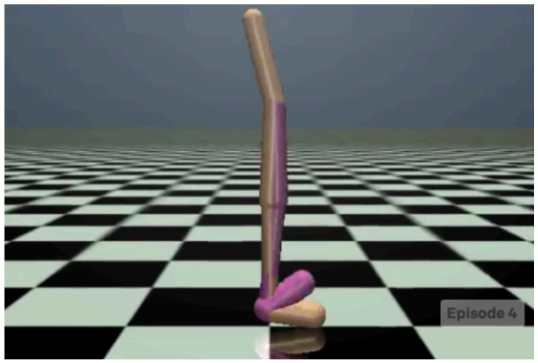
(b) HalfCheetah-v2



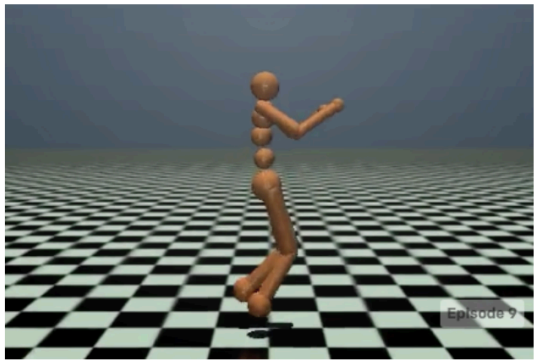
(c) Hopper-v2



(d) Reacher-v2



(e) Walker-v2



(f) Humanoid-v2

https://arxiv.org/abs/1707.06347

Experimental Results

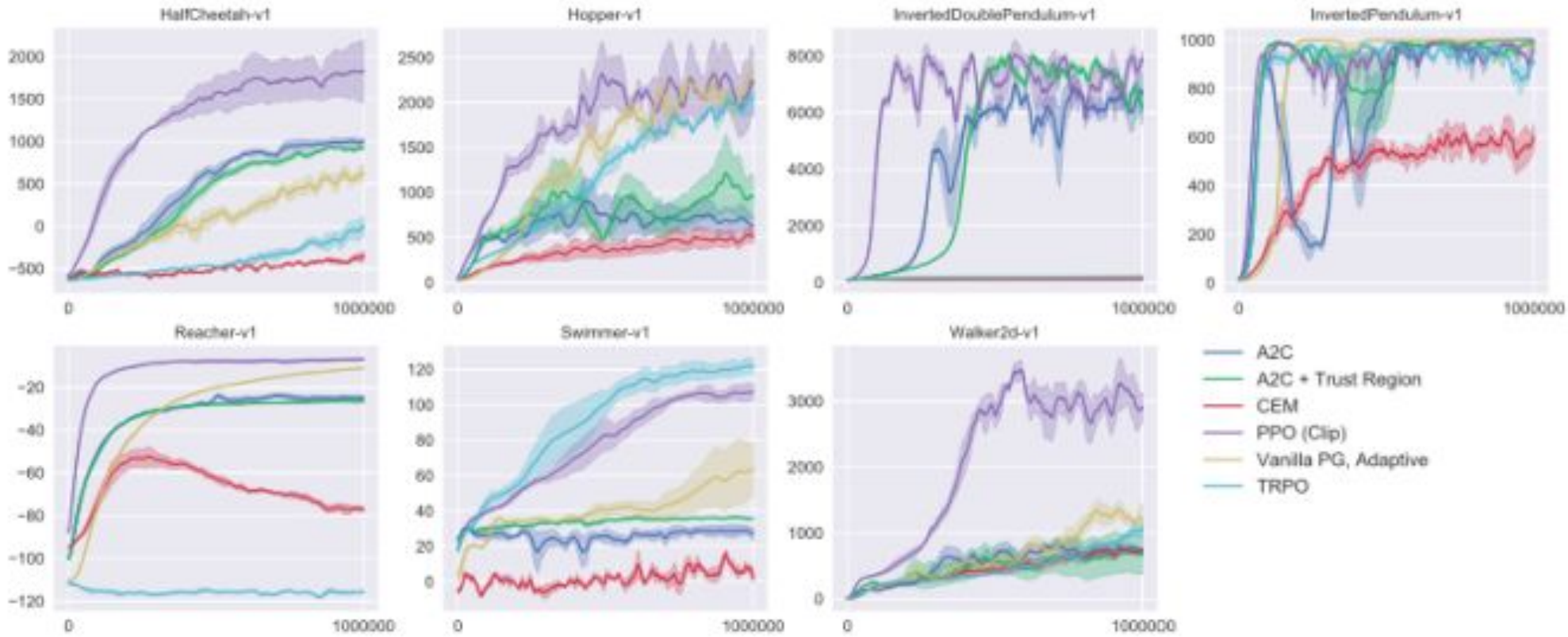


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

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

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

baseline

G_t^n : obtained via interaction

Very unstable

Review – Policy Gradient

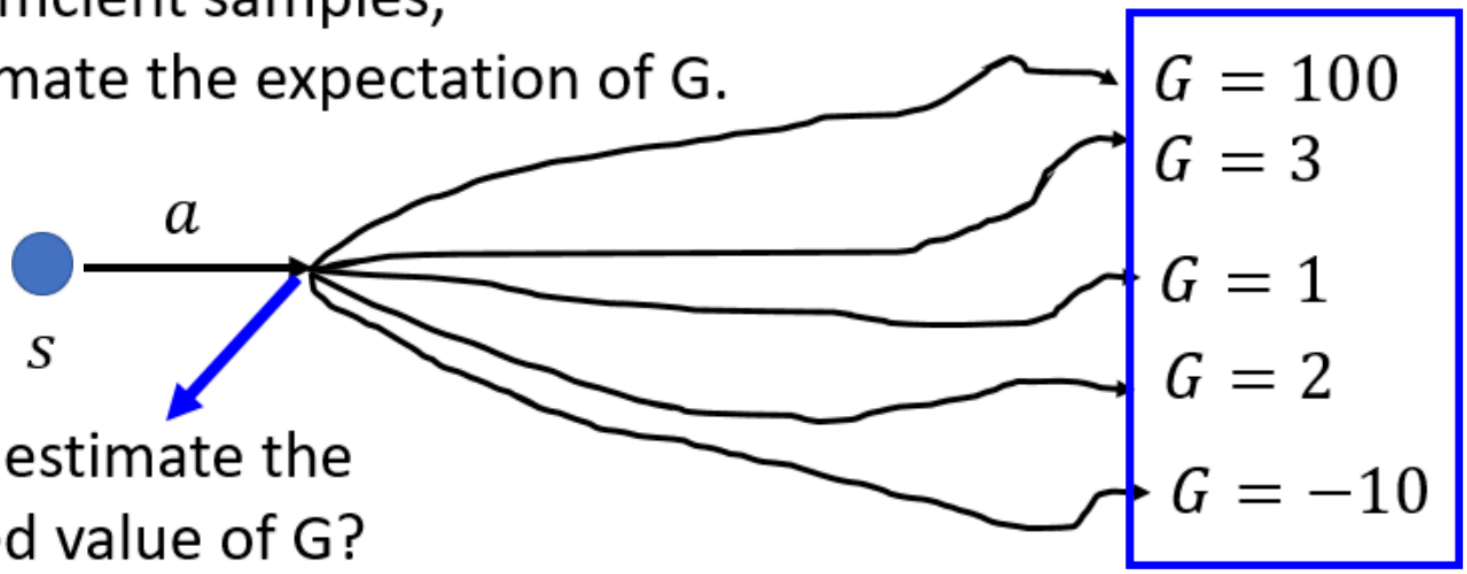
baseline

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

G_t^n : obtained via interaction

Very unstable

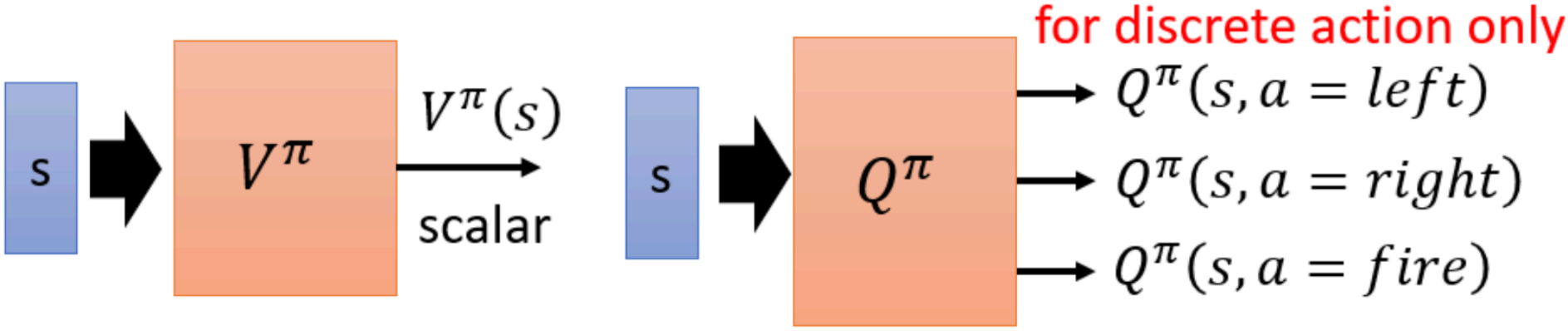
With sufficient samples,
approximate the expectation of G.



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

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

baseline

G_t^n : obtained via interaction

Actor-Critic

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

baseline

G_t^n : obtained via interaction

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

Actor-Critic

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

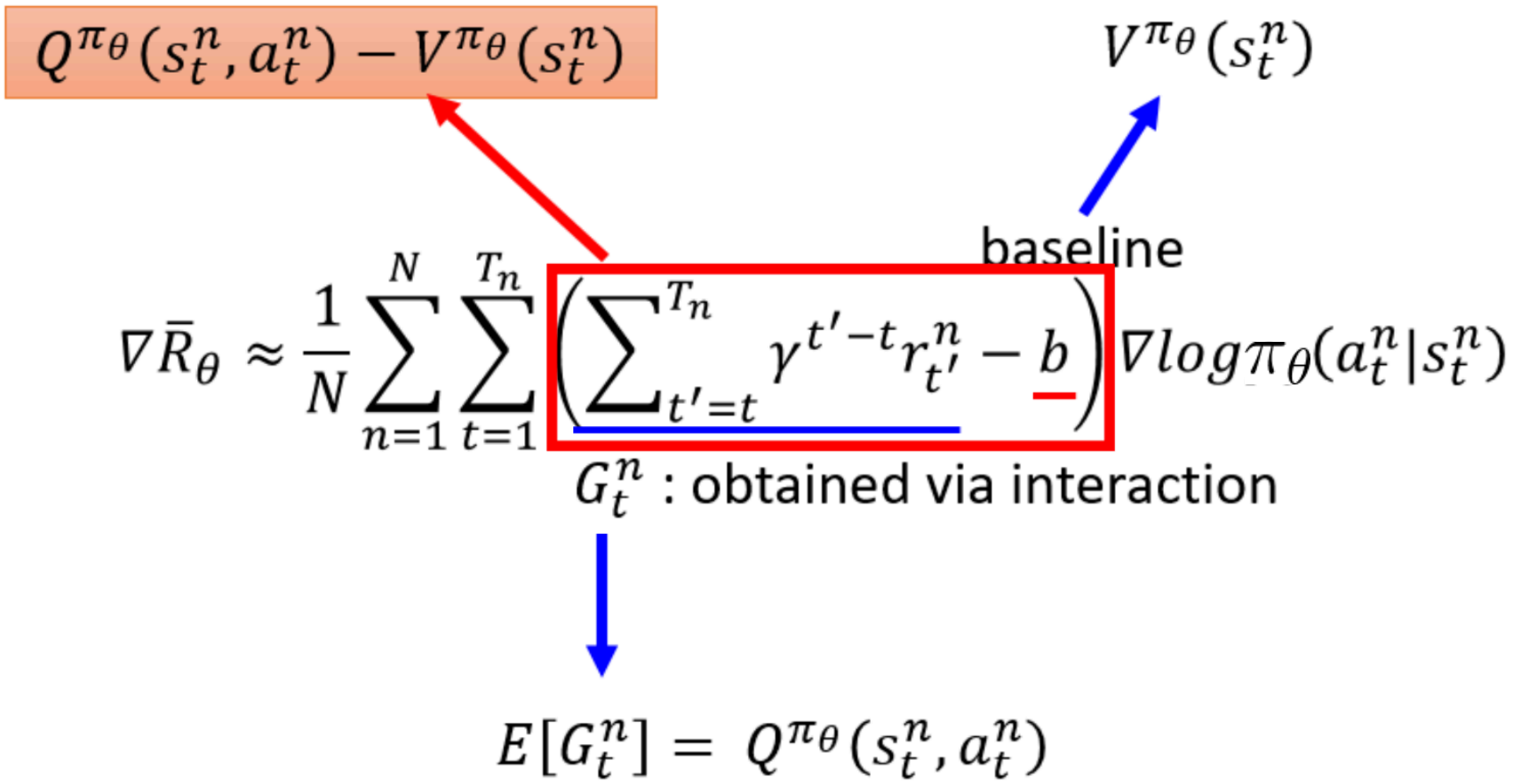
baseline

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

G_t^n : obtained via interaction

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

Actor-Critic



Advantage Actor-Critic

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

Estimate two networks? We can only estimate one.

Advantage Actor-Critic

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

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

Estimate two networks? We can only estimate one.

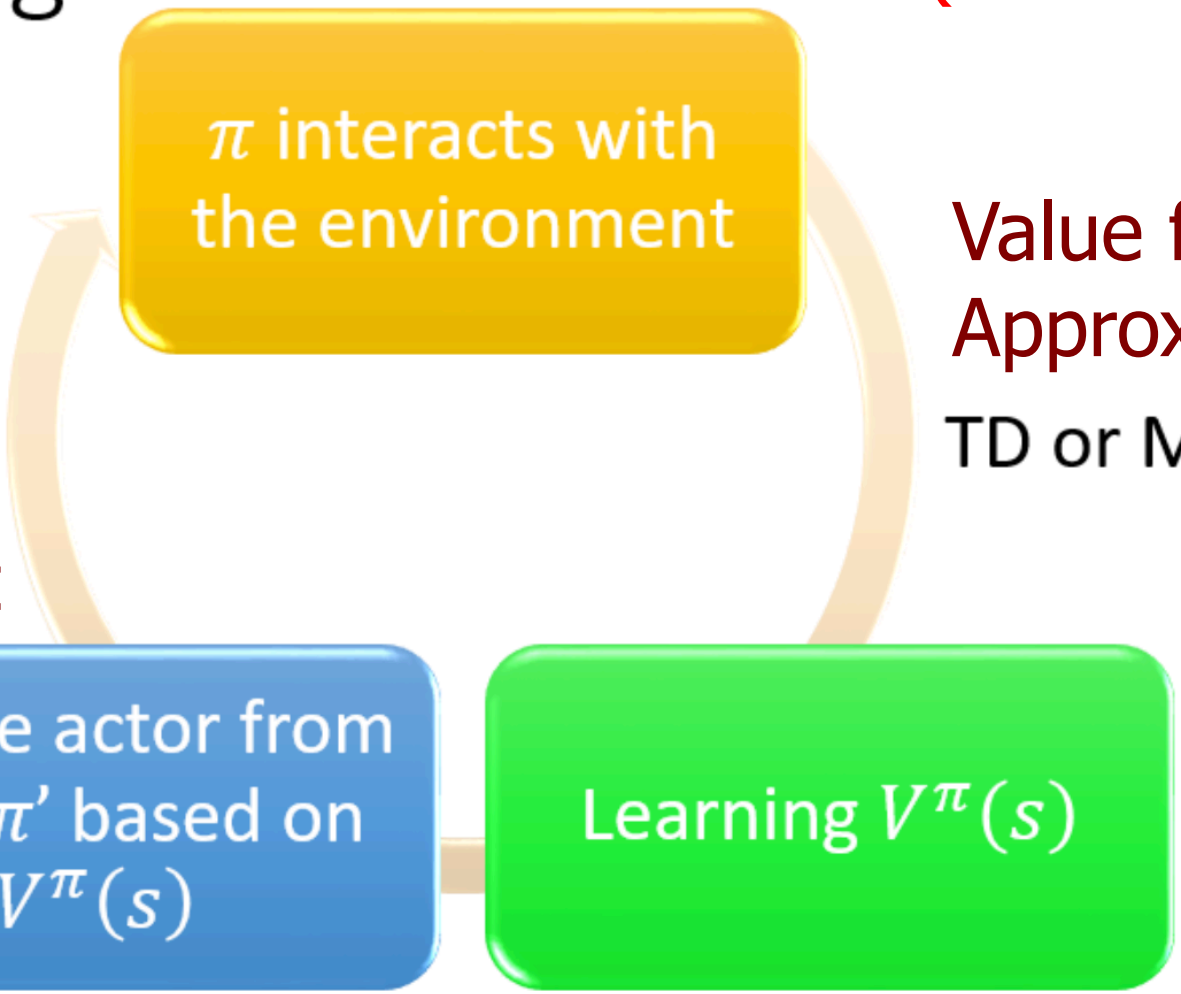
Only estimate state value

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

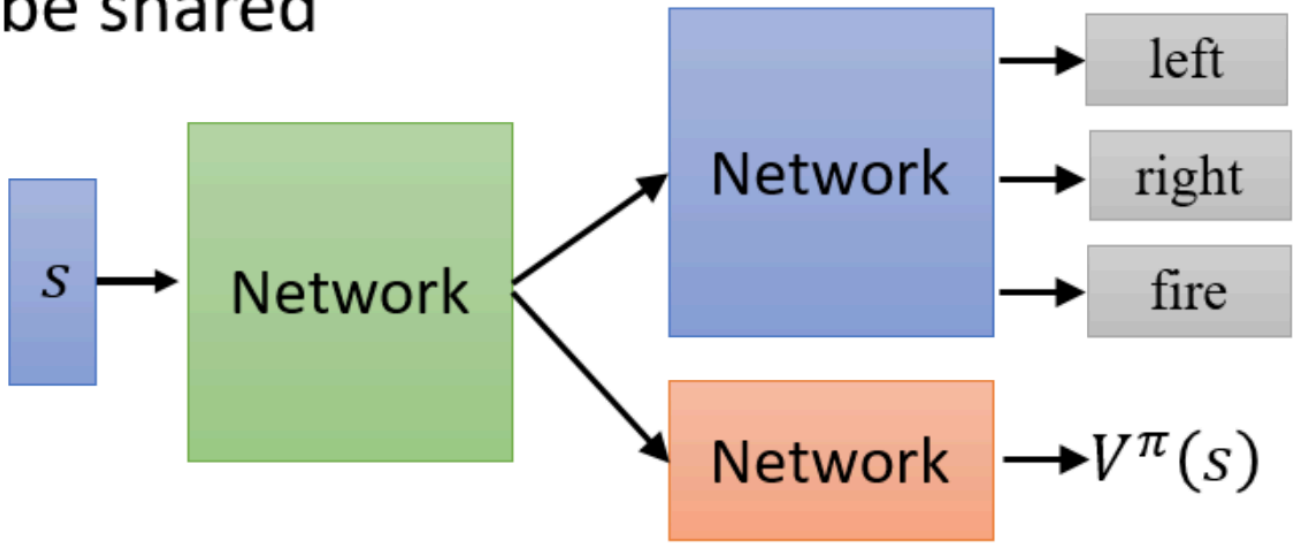
$\pi = \pi'$

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

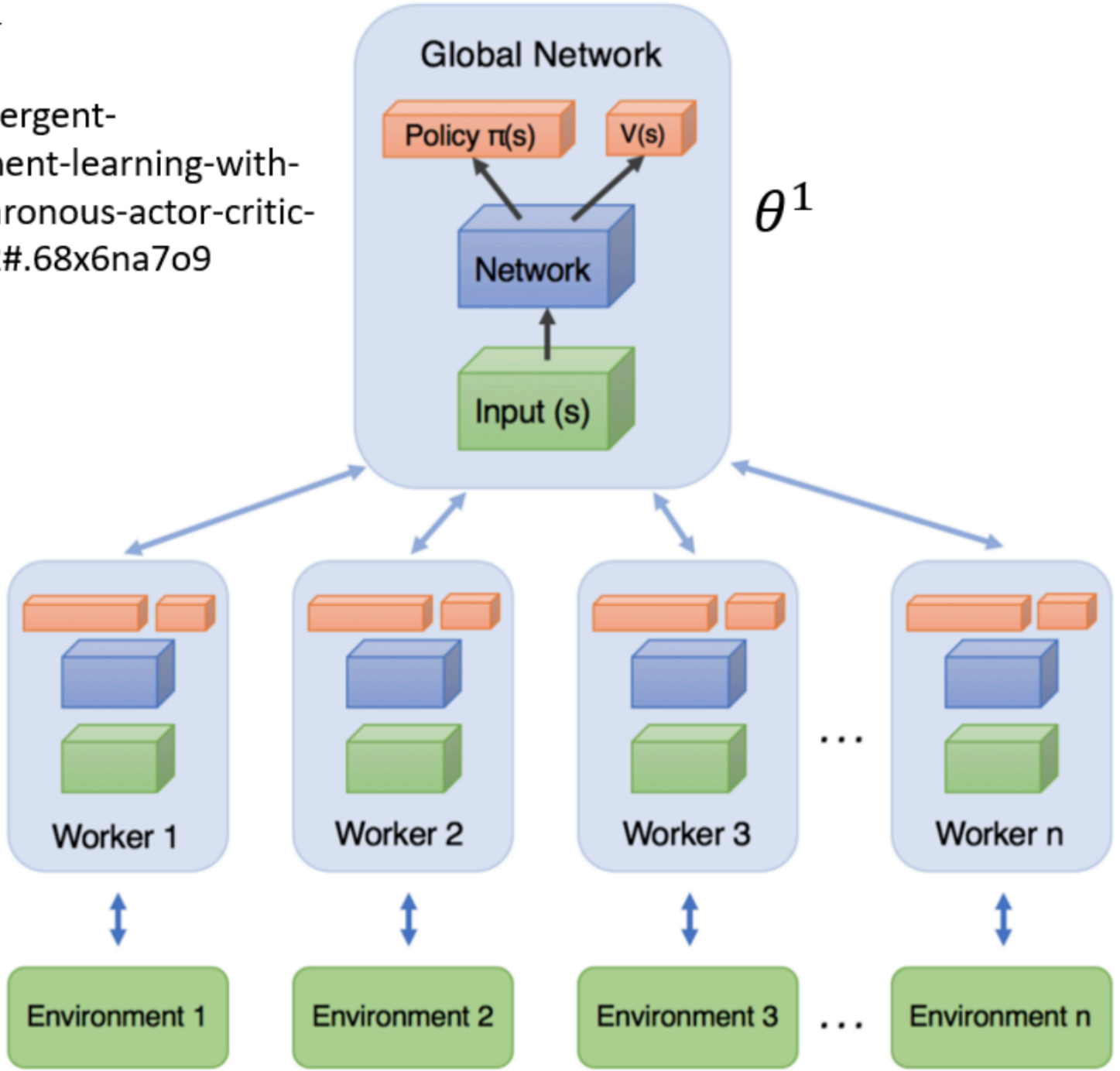
- Tips
  - The parameters of actor $\pi(s)$ and critic $V^{\pi}(s)$ can be shared



Asynchronous Advantage Actor-Critic (A3C)

Asynchronous

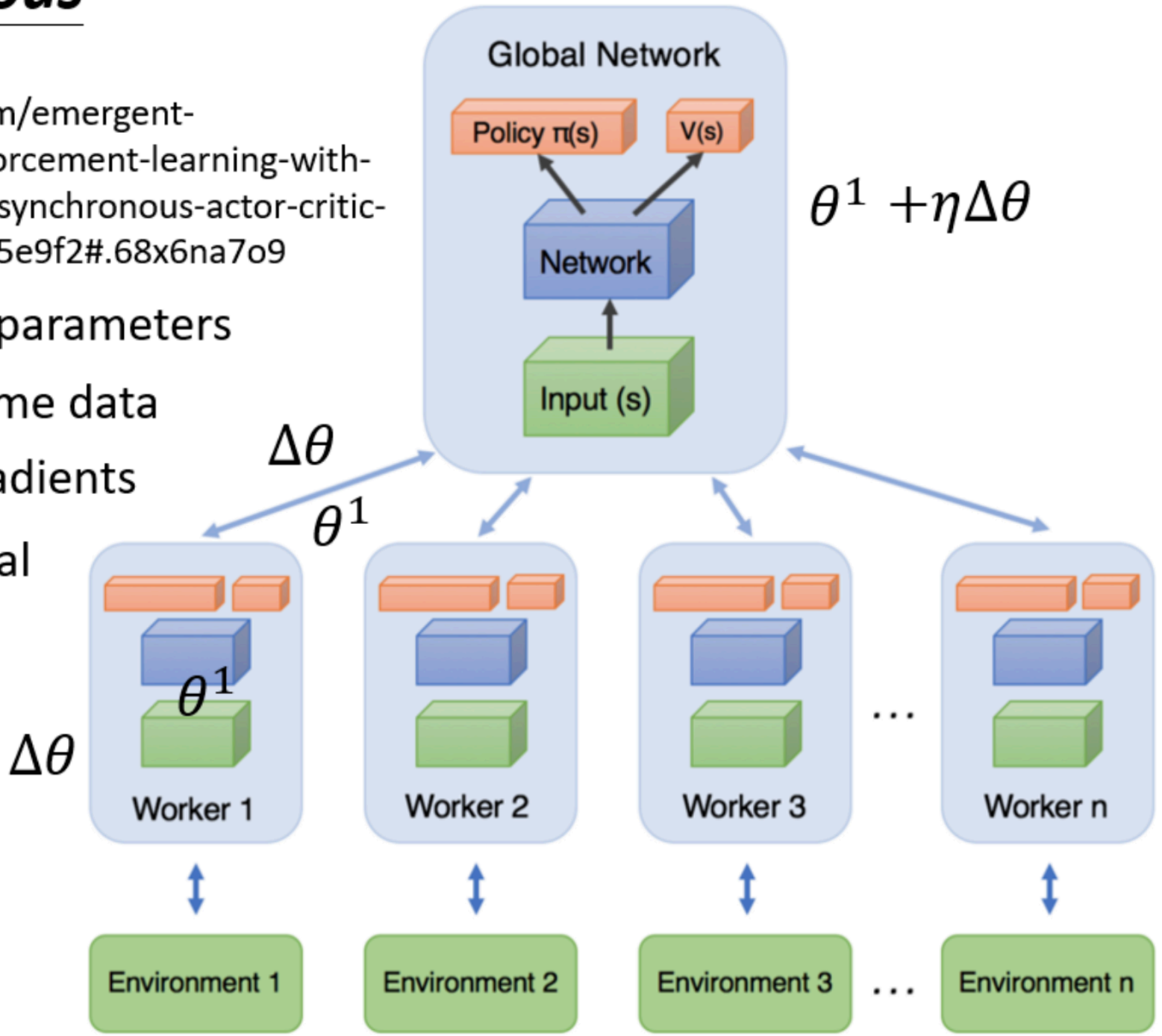
Source of image:
https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9



Asynchronous

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

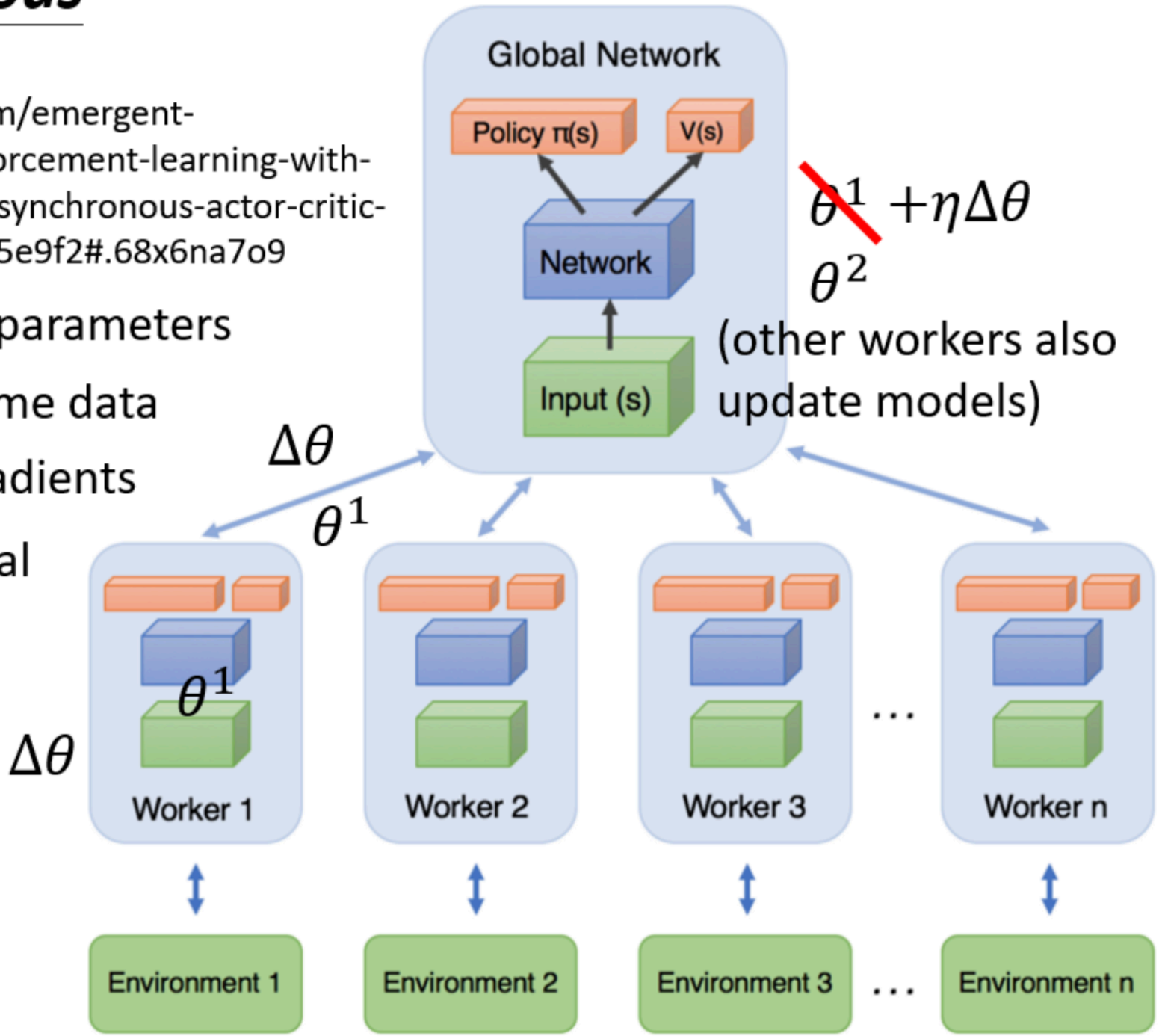
1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



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



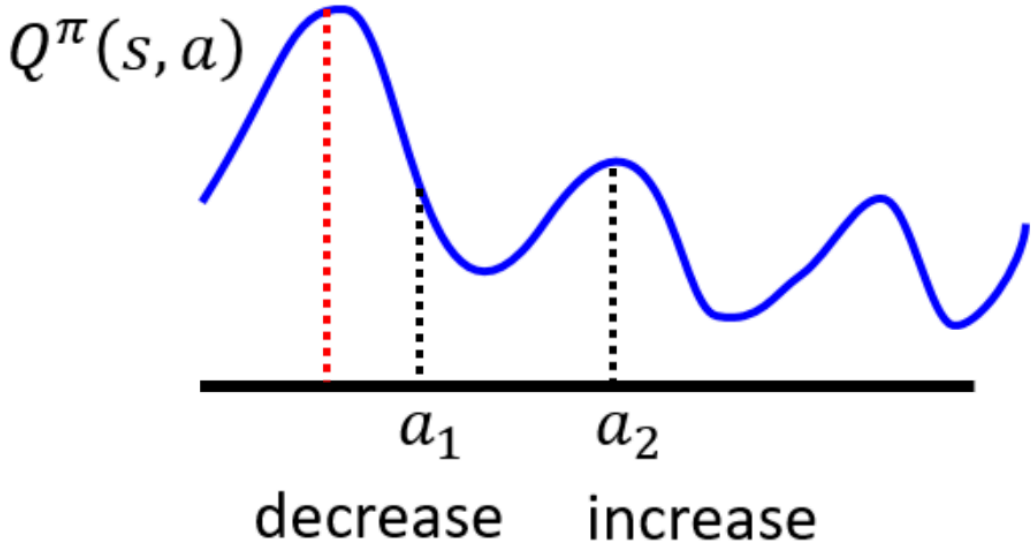
Pathwise Derivative Policy Gradient

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

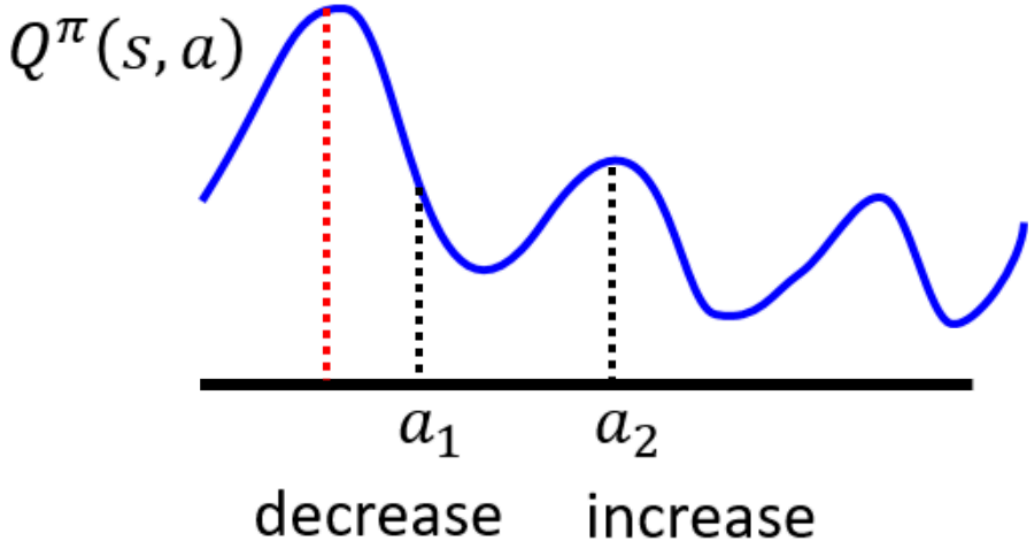
Another Way to use Critic

Original Actor-critic



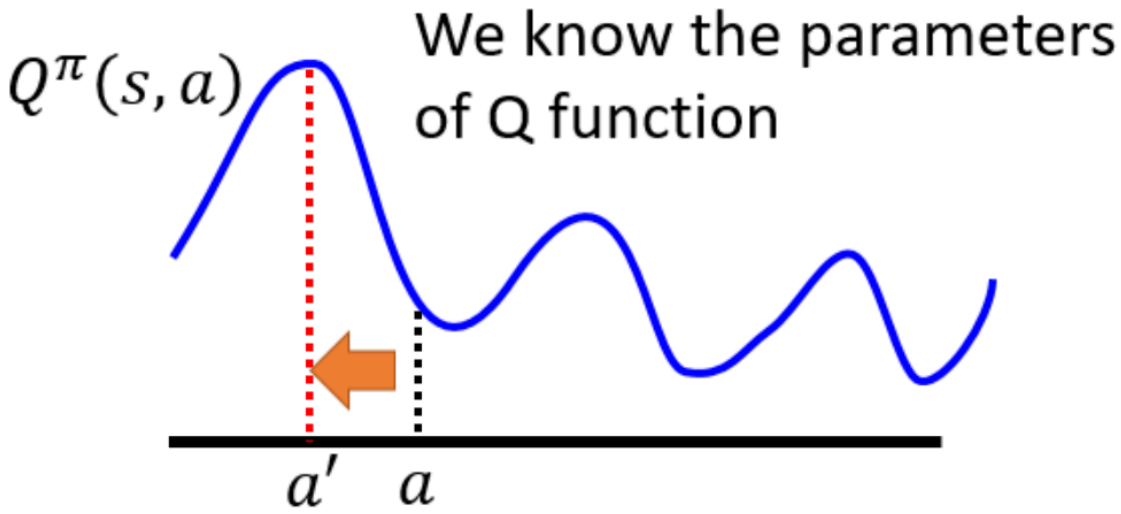
Another Way to use Critic

Original Actor-critic



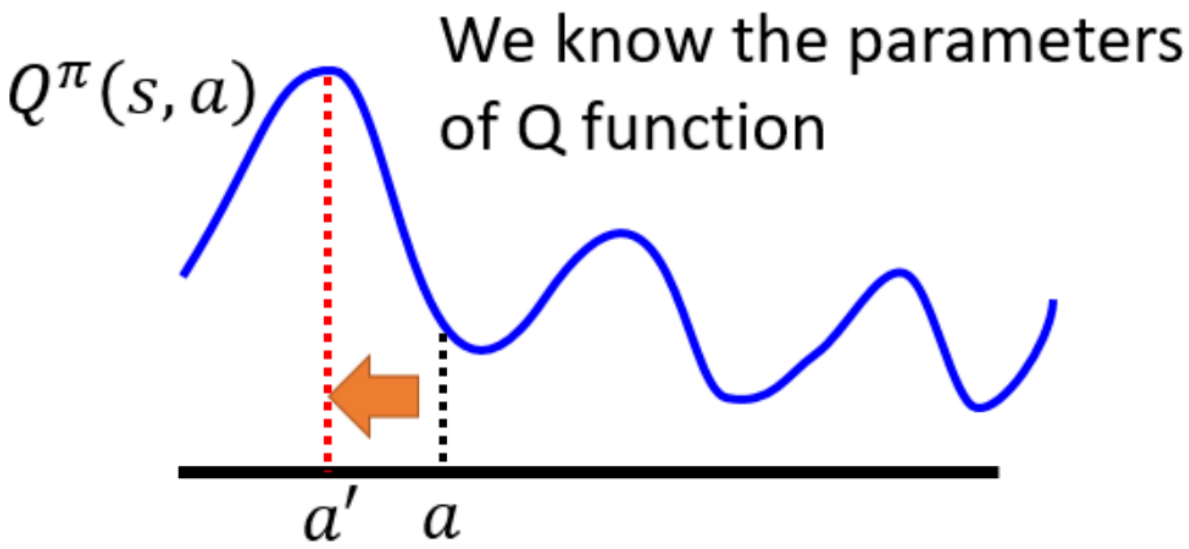
Pathwise derivative policy gradient

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



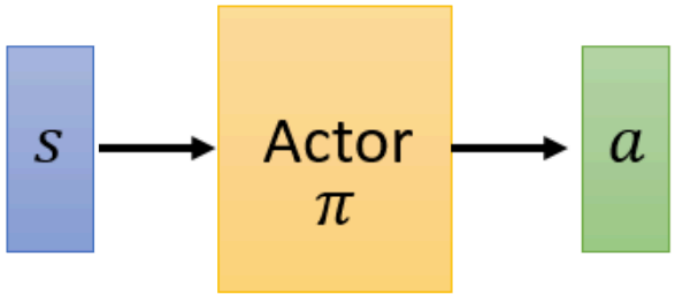
Pathwise derivative policy gradient

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



Action a is a *continuous vector*

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



Actor as the solver of this optimization problem

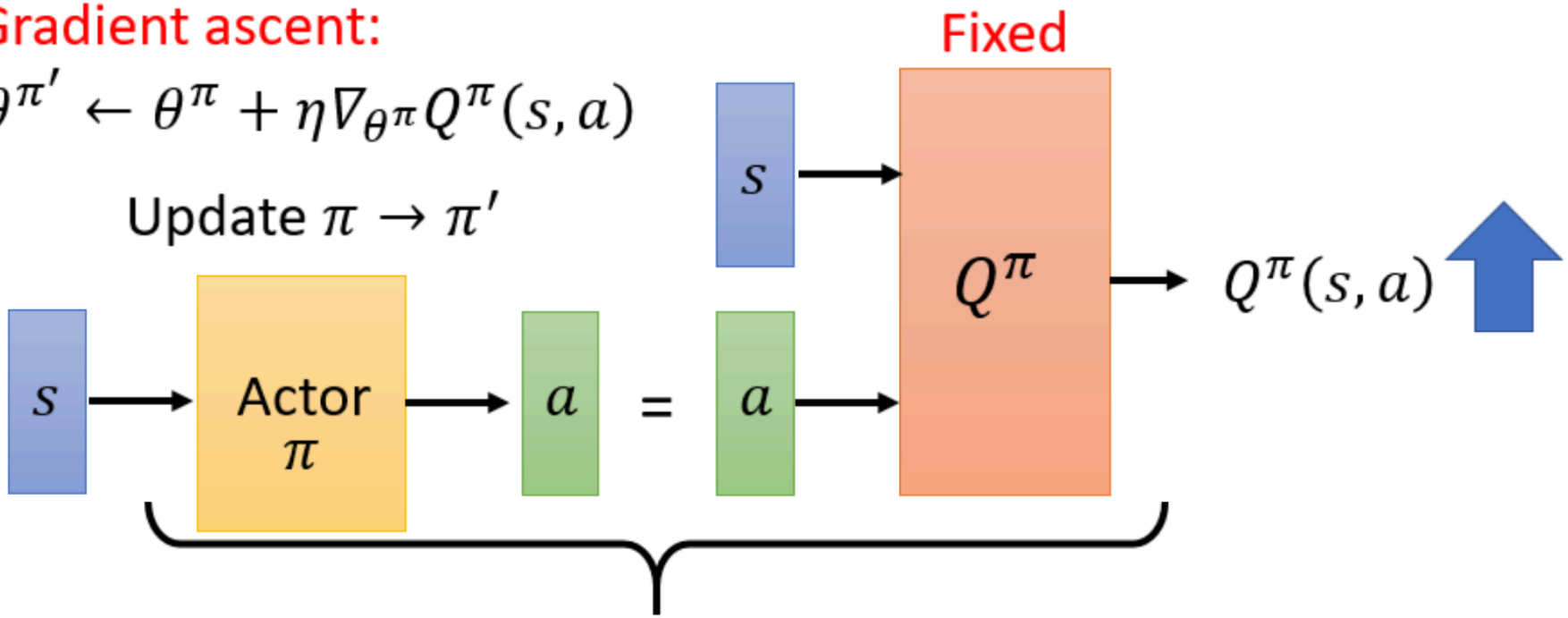
Pathwise Derivative Policy Gradient

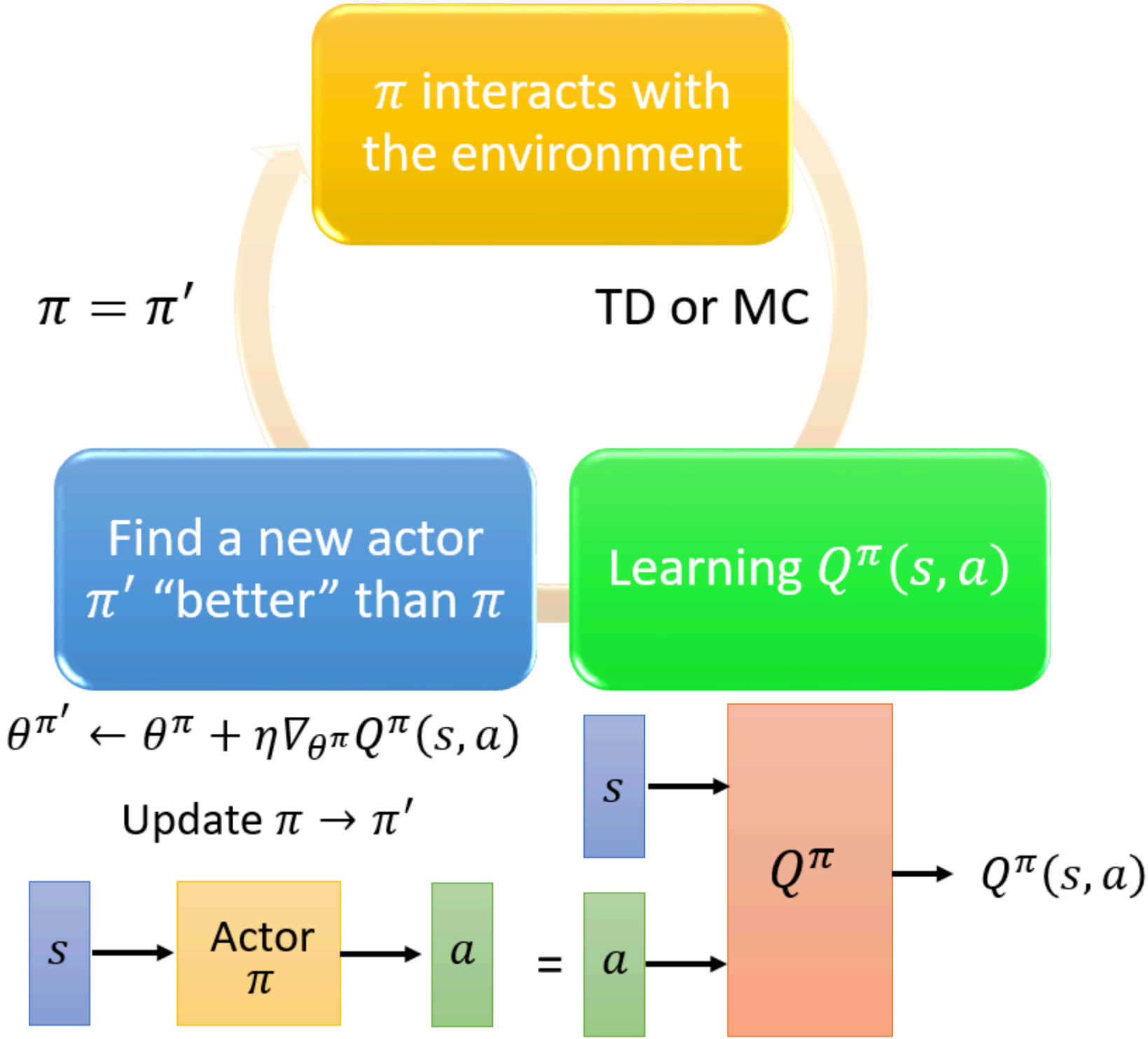
$\pi'(s) = arg\max_a Q^{\pi}(s,a)$  a is the output of an actor

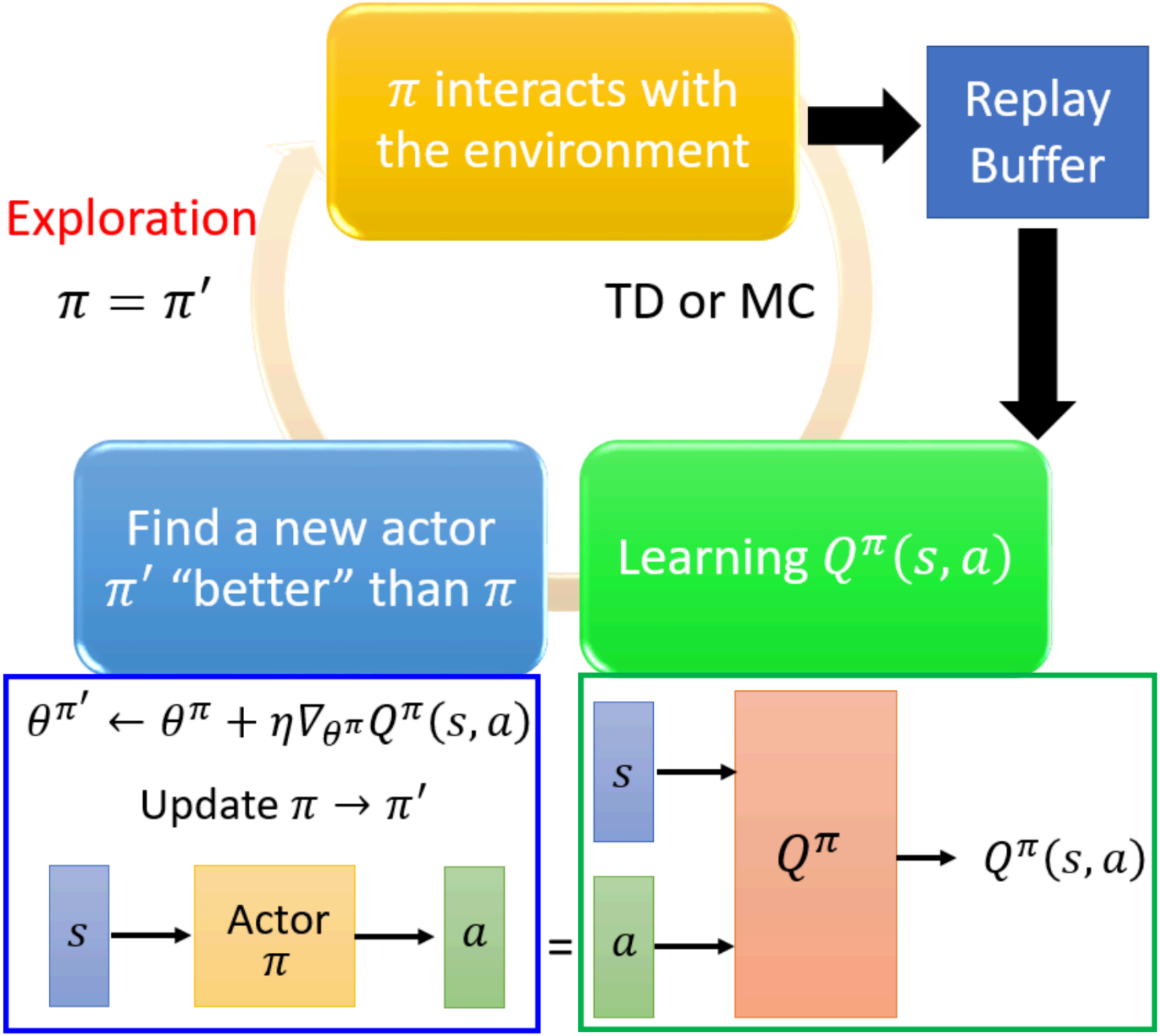
Gradient ascent:

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

Update $\pi \rightarrow \pi'$




π interacts with the environment
$\pi = \pi'$
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}
$Q^{\pi}(s, a)$


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 $y = r_i + \max_a \hat{Q}(s_{i+1}, a)$ ~~$\max_a \hat{Q}(s_{i+1}, a)$~~ $\hat{Q}\left(s_{i+1}, \hat{\pi}(s_{i+1})\right)$
   - 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 $\hat{\pi} = \pi$

The Lecture after next week

- Advanced deep reinforcement learning approaches
  - Sparse Reward Problems/Techniques
  - Generative Adversarial Networks (GANs) Review
  - Deep Inverse reinforcement learning
    - Entropy based IRL
    - GAN (Generative adversarial networks)
    - GAIL (Generative adversarial imitation learning)

	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**
Multiple Agents	**Multi-Agent Reinforcement Learning** Multi-agent Actor-Critic etc.	**Multi-Agent Inverse Reinforcement Learning** MA-GAIL MA-AIRL AMA-GAIL

Applications

Questions?