Welcome to

DS595/CS525
Reinforcement Learning

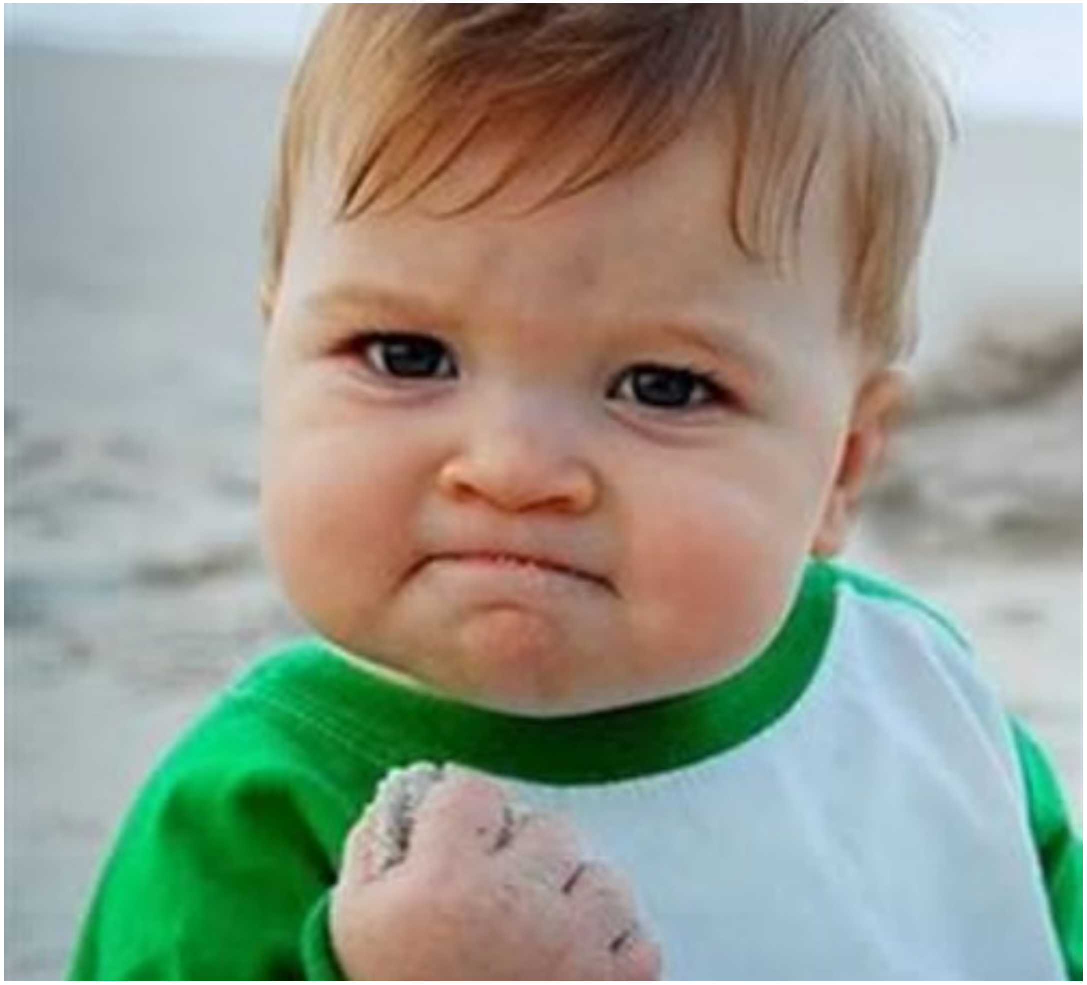
Prof. Yanhua Li

Time: 6:00pm –8:50pm R
Location: FL PH Lower
Fall 2019

Quiz 5 Today

- 20 minutes on Policy Gradient (PG)

No Quiz Next Week



Class arrangement

https://users.wpi.edu/~yli15/courses/DS595CS525Fall19/Schedule.html

-12. Week 12 (11/7 R): (Prof Li is on a travel, and invited PhD student speakers will give research work presentations)

Topic: RL and IRL Applications: Research work presentations from PhD students from Prof Li's group, by Menghai Pan and Xin Zhang.
Work #1. [SDM'19] **Menghai Pan**, Yanhua Li, Xun Zhou, Zhenming Liu, Rui Song, Hui Lu, Jun Luo, Dissecting the Learning Curve of Taxi Drivers: A Data-Driven Approach. SIAM International Conference on Data Mining, (SDM'19 Best Applied Data Science Paper Award!) (Paper PDF)
Work #2. [ICDM'19] **Xin Zhang**, Yanhua Li, Xun Zhou, Jun Luo, Unveiling Taxi Drivers' Strategies via cGAIL -- Conditional Generative Adversarial Imitation Learning, IEEE International Conference on Data Mining (Paper PDF).
Work #3. A work under double-blind review by **Xin Zhang**.

Project 3 is available
Due 10/17 Thursday
10 bonus points and a leader board

- https://users.wpi.edu/~yli15/courses/DS595CS525Fall19/Assignments.html

- https://github.com/huiminren/DS595CS525-RL-HW/tree/master/project3

Leader board (as of 5:30PM today)

Leaderboard for Breakout-DQN Update Date: 10/31/2019 17:30

Top	Date	Name	Score	Note
1	10/31/2019	**Mohamed Mahdi Alouane**	**329.46**	Double DQN with 1e-6 learning rate trained for 100K episodes
2	10/22/2019	**Prathyush SP**	**142.77**	Conv Network and Priority Buffer trained for 50k episodes
	10/18/2019	Prathyush SP	81.07	Simple DQN with Conv Based Architecture for 60k episodes
3	10/28/2019	**Sapan Agrawal**	**91.34**	Architecture described in the DQN paper for 120k episodes
4	10/26/2019	Vamshi Krishna Uppununthala	79.5	Dueling DQN for 50k episodes
5	10/24/2019	Shreesha Narasimha Murthy	56.79	Simple DQN with MSE for 40k episodes
6	10/20/2019	Sinan Morcel	53.26	Plain DQN with TA's parameters

Project 4 is available
Starts 10/17 Thursday
Due 12/12 Thursday mid-night

- https://github.com/huiminren/DS595CS525-RL-HW/tree/master/project4
- Important Dates
- Project Proposal: Thursday Today
- Project Progress: Thursday 11/14/2019
- Final Project: Thursday 12/12/2019

Project 4 Team Assignment

10 teams

Team assignment can be updated by this weekend.

Proposal is due today

Some sampled cool ideas from you.

1. Real world robot planning
2. Mojuco environment agent training
3. Multi-agent RL
4. Sparse reward problem, etc.

Last Lecture

- Imitation Learning / Inverse Reinforcement Learning
  - Introduction
  - Behavioral Cloning
  - Inverse reinforcement learning
    - Model-Based, Linear Reward Functions (this time)
- Policy Gradient
  - Intro and Stochastic Policy
  - Basic Policy Gradient Algorithm
  - Vanilla Policy Gradient
  - PPO, TRPO, PPO2

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

	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

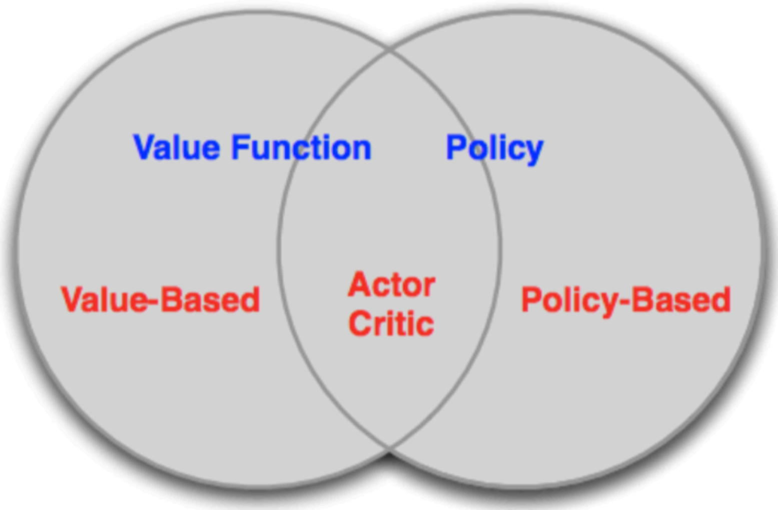
- Policy Gradient (Review Quickly)
  - Intro and Stochastic Policy
  - Basic Policy Gradient Algorithm
  - REINFORCE and Vanilla Policy Gradient
  - PPO, TRPO, PPO2
- Actor-Critic methods
  - A2C
  - A3C
  - Pathwise Derivative Policy Gradient

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

Value-Based and Policy-Based RL

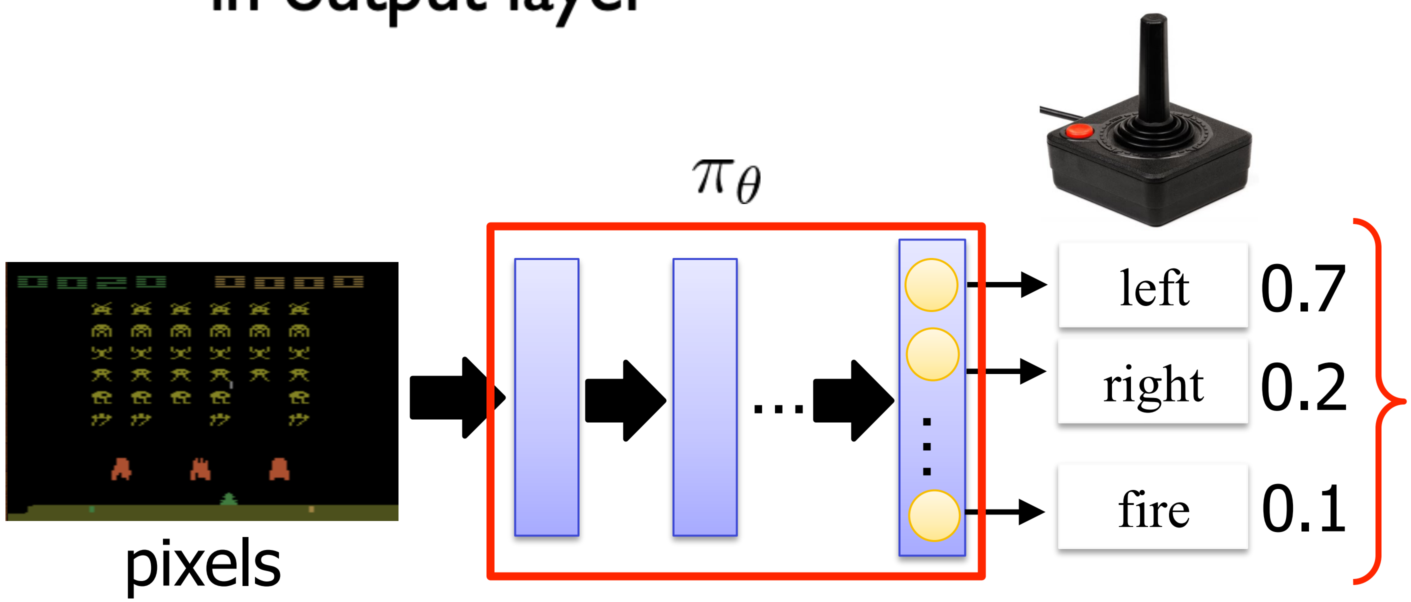
Model-Free RL:
Explicit: Value function and/or policy function
No model

- 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

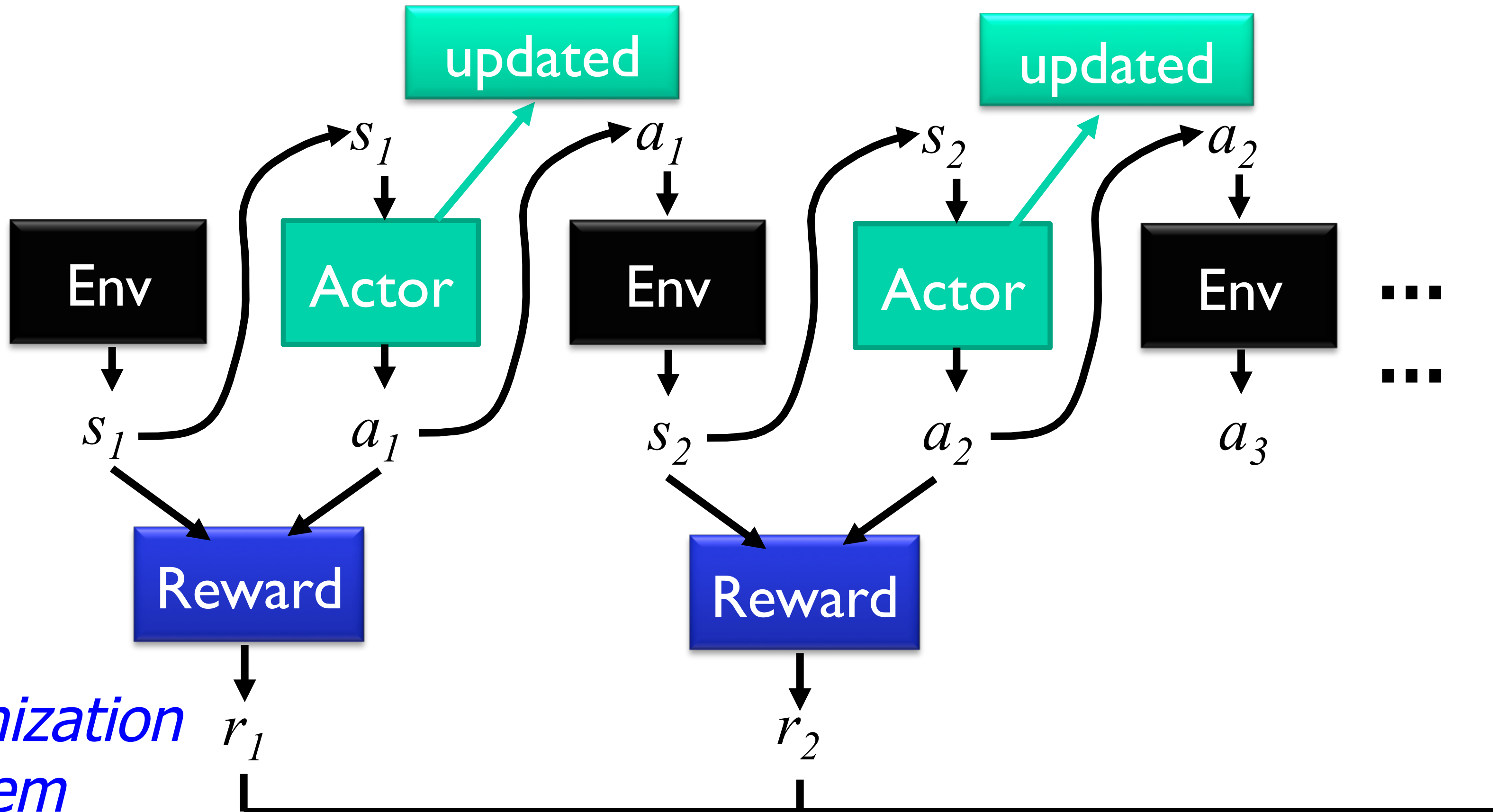
- Policy π is a network with parameter $\theta \longrightarrow \pi_\theta$
  - Input: the observation of machine represented as a vector or a matrix
  - Output: each action corresponds to a neuron in output layer



Take the action based on the probability.

Score of an action

Actor, Environment, Reward



Optimization Problem

Expected Reward (Objective to maximize`

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

$$R(\tau) = \sum_{t=1}^{T} r_t$$

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

Basic Policy Gradient Algorithm

Given policy π_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

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

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

Data Collection

only used once

Unbiased estimator

From basic PG algorithm to···

- Issues with the basic PG algorithm
  - TIP 1. Inaccurate update when non-negative rewards
    - Add baseline:
  - 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 sampling to reuse data when training: PPO, TRPO, PPO2

Monte-Carlo Policy Gradient (REINFORCE)

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

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* $R_t = \sum_{t'=t}^{T-1} r_{t'}$, and
  the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
  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 $\hat{g}$,
  which is a sum of terms $\nabla_\theta \log \pi(a_t|s_t, \theta)\hat{A}_t$.
  (Plug $\hat{g}$ into SGD or ADAM)
endfor

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

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

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.

Given policy π_θ

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

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

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

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

On-policy → Off-policy

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

- 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.

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

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

- 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.

Importance Sampling

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

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

Issue of Importance Sampling

?

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

$$Var_{x\sim p}[f(x)] \quad Var_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

$VAR[X]$
$= E[X^2] - (E[X])^2$

Issue of Importance Sampling

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

$$\mathrm{V}ar_{x \sim p}[f(x)] \quad \mathrm{V}ar_{x \sim q}[f(x)\frac{p(x)}{q(x)}]$$

$$VAR[X] = E[X^2] - (E[X])^2$$

$$\mathrm{V}ar_{x \sim p}[f(x)] = E_{x \sim p}[f(x)^2] - \left(E_{x \sim p}[f(x)]\right)^2$$

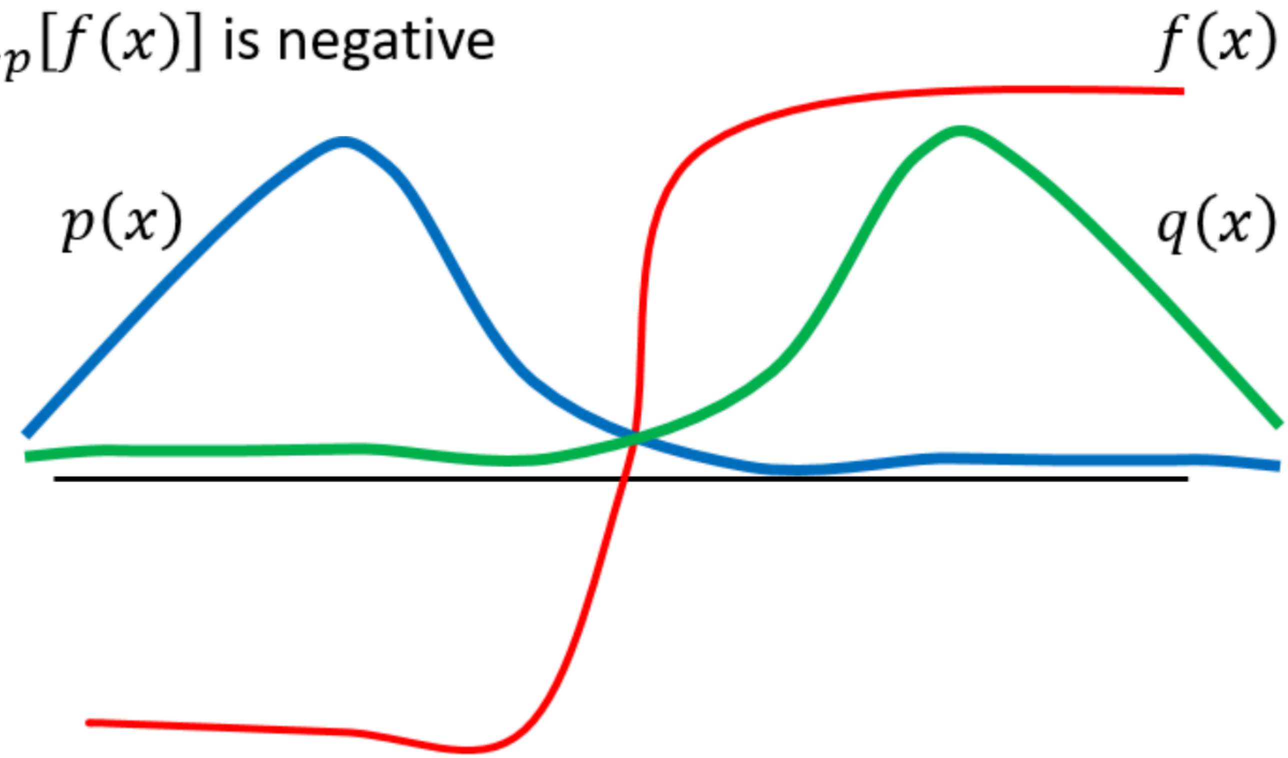
$$\mathrm{V}ar_{x \sim q}[f(x)\frac{p(x)}{q(x)}] = E_{x \sim q}\left[\left(f(x)\frac{p(x)}{q(x)}\right)^2\right] - \left(E_{x \sim q}\left[f(x)\frac{p(x)}{q(x)}\right]\right)^2$$

$$= E_{x \sim p}\left[f(x)^2 \boxed{\frac{p(x)}{q(x)}}\right] - \left(E_{x \sim p}[f(x)]\right)^2$$

Issue of Importance Sampling

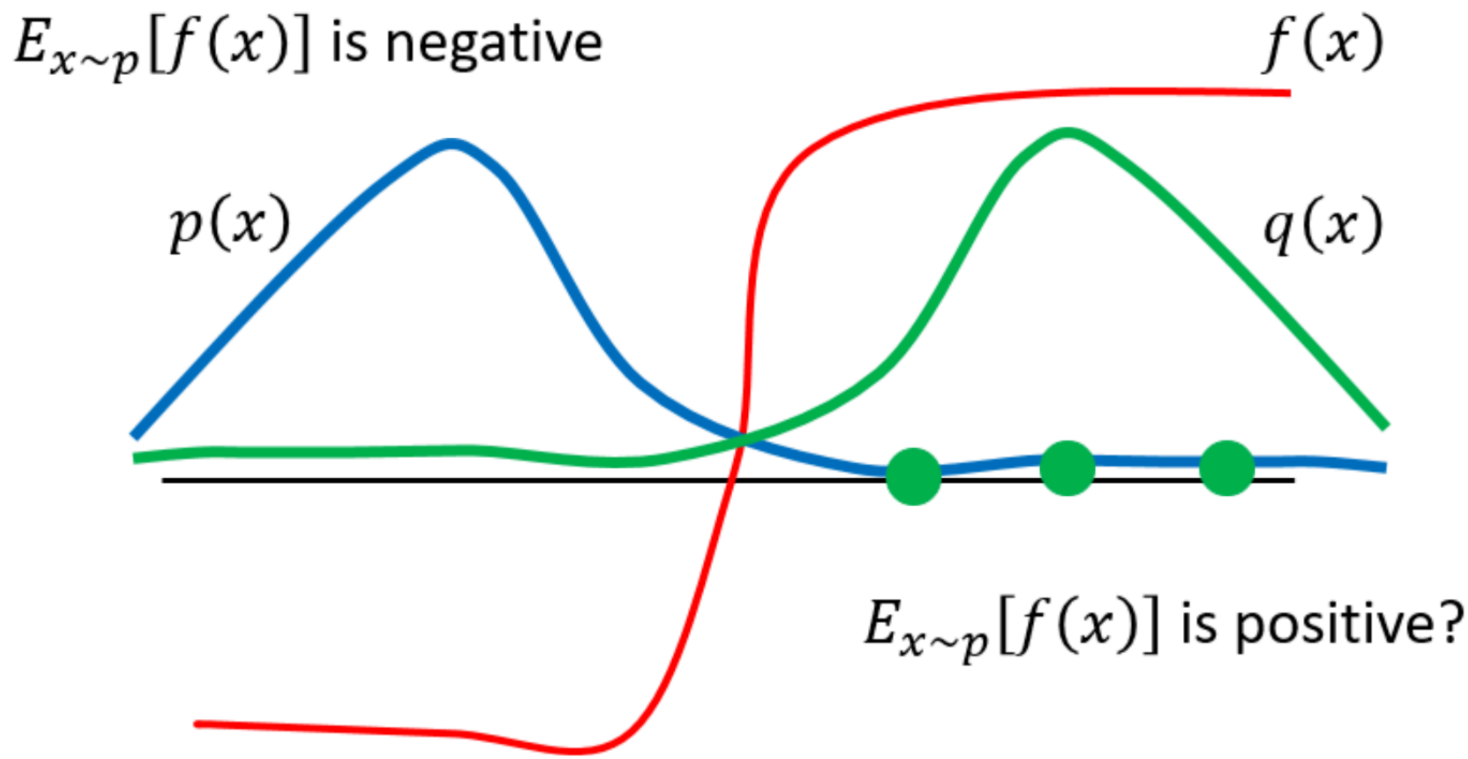
$$E_{x \sim p}[f(x)] = E_{x \sim q}[f(x)\frac{p(x)}{q(x)}]$$

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



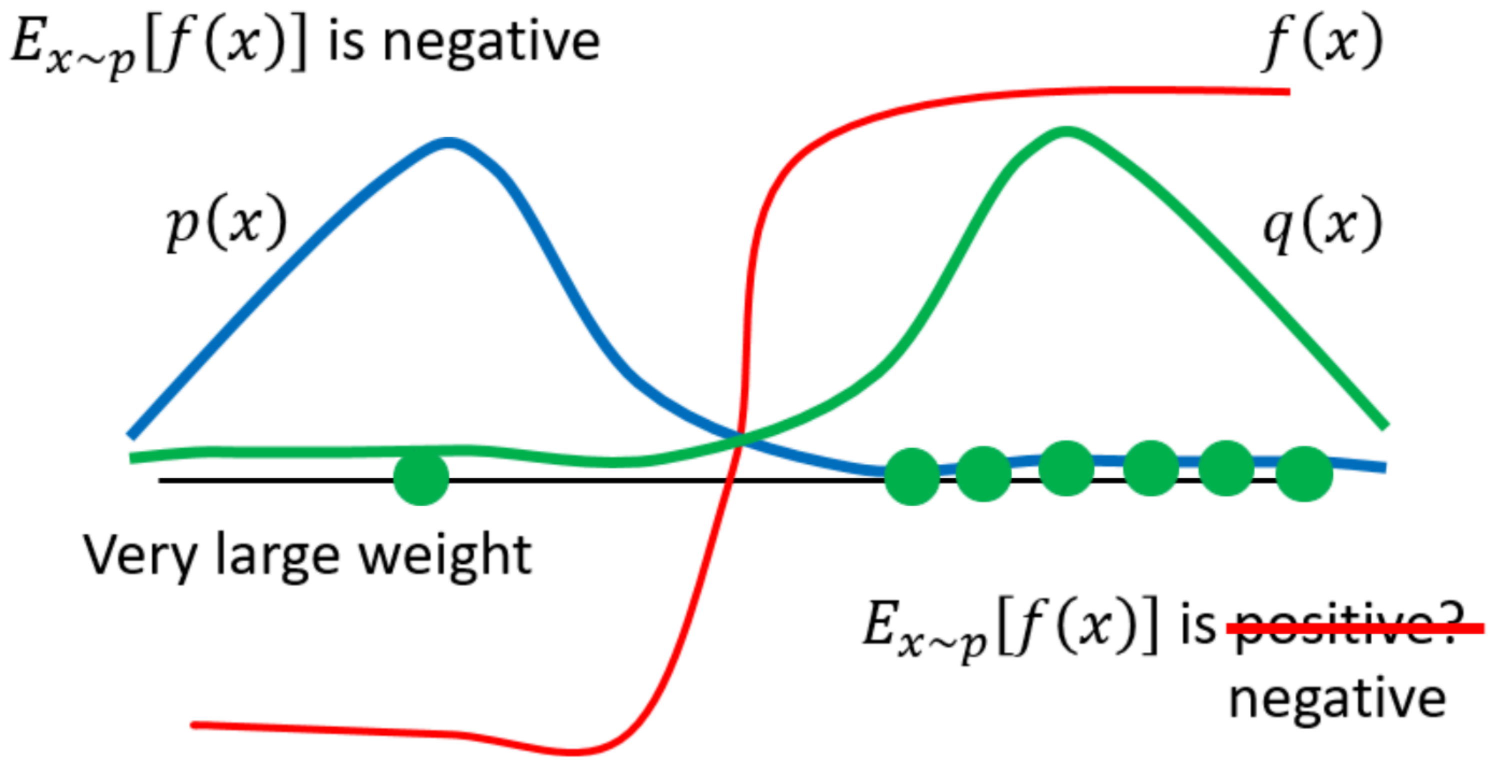
Issue of Importance Sampling

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



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

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

- 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.

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

Basic PG

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

<u>Importance Sampling</u>

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

On-policy → Off-policy

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

Gradient for update

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

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

On-policy → Off-policy

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

Gradient for update

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

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

This term is from sampled data.

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

?

On-policy → Off-policy

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

Gradient for update

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

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

This term is from sampled data.

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

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

On-policy → Off-policy

?

Gradient for update

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

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

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

This term is from sampled data.

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

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

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

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)

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

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

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

TRPO (Trust Region Policy Optimization) (2015)

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

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

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

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

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

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

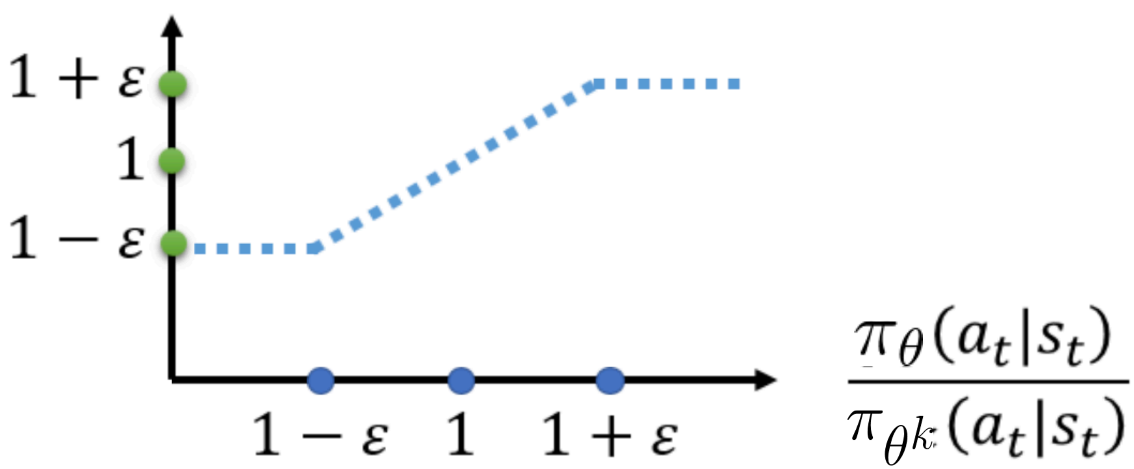
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

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

PPO2 algorithm

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

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



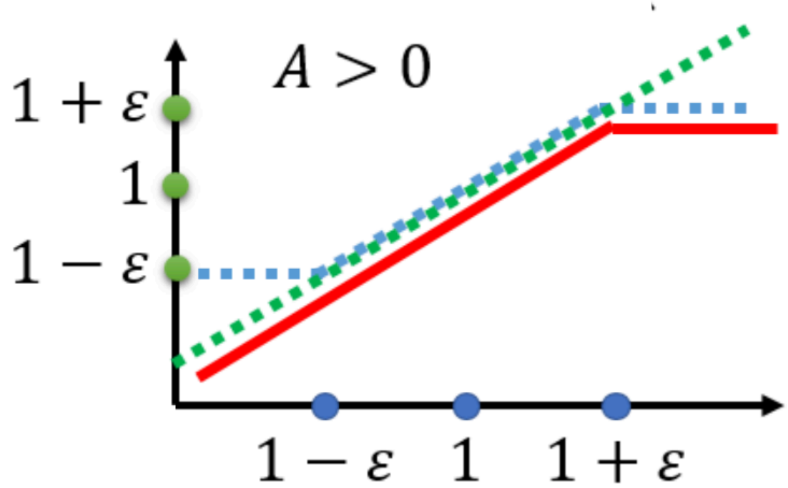
PPO algorithm

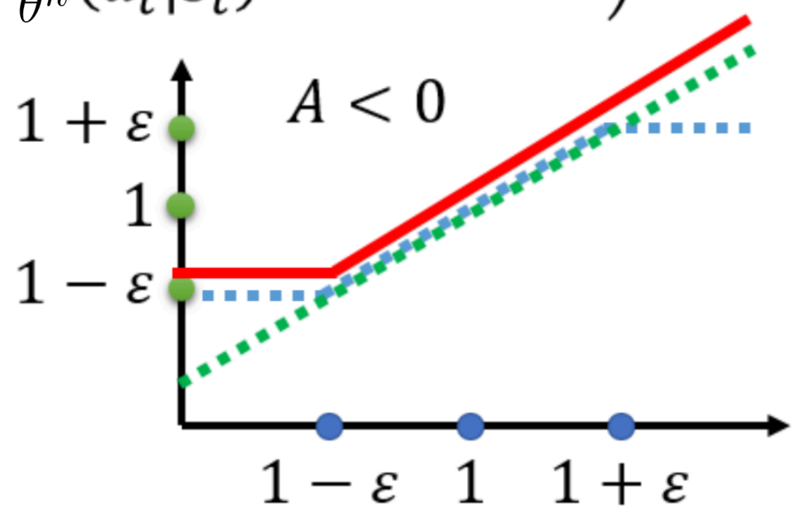
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

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

PPO2 algorithm

$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t), \right.$$
$$\left. clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon \right) A^{\theta^k}(s_t, a_t) \right)$$





https://arxiv.org/abs/1707.06347

Experimental Results

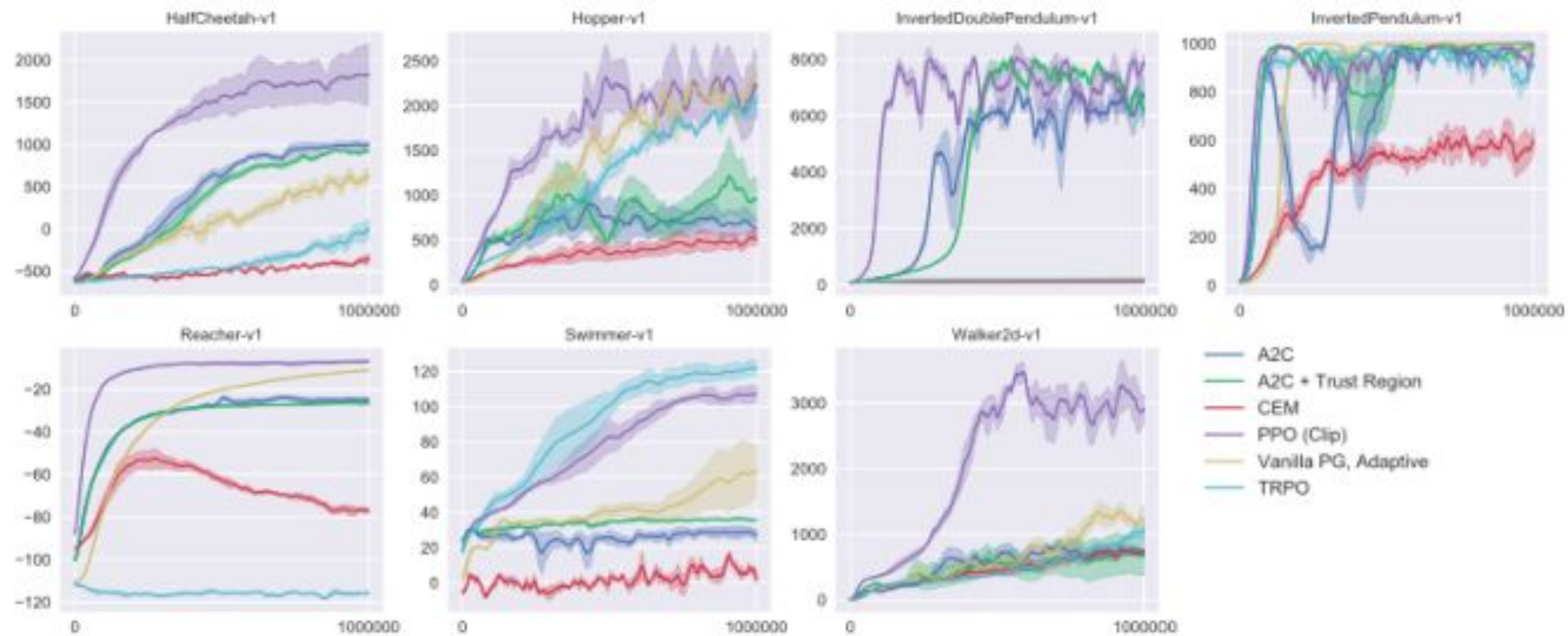


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

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

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

G_t^n : obtained via interaction

Very unstable

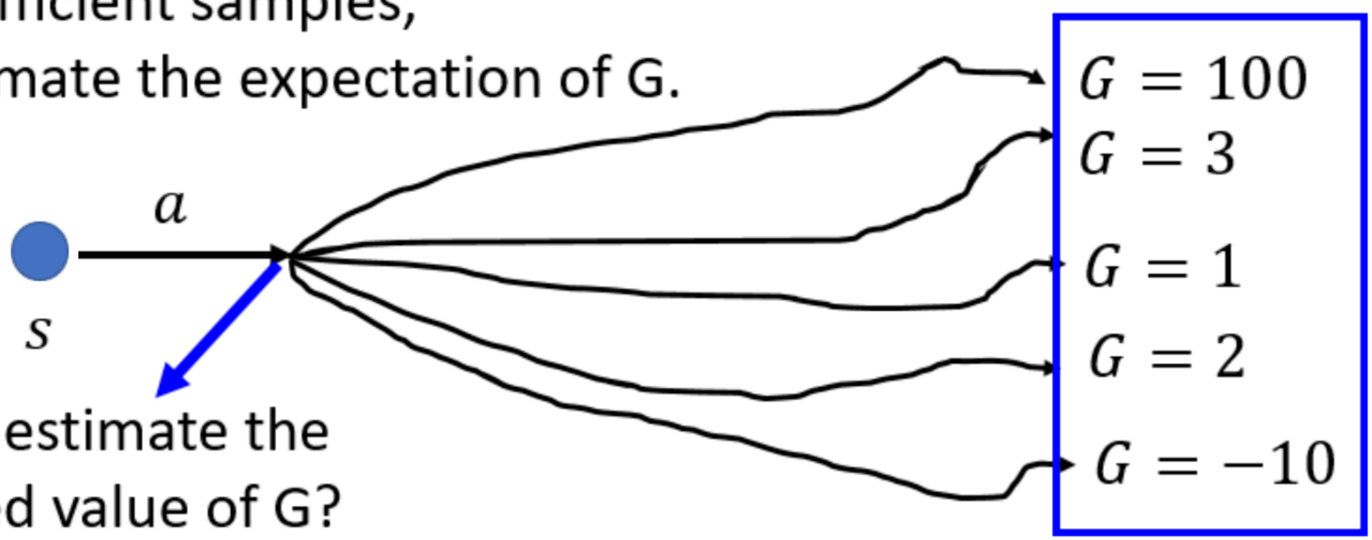
Review – Policy Gradient

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

G_t^n : obtained via interaction

Very unstable

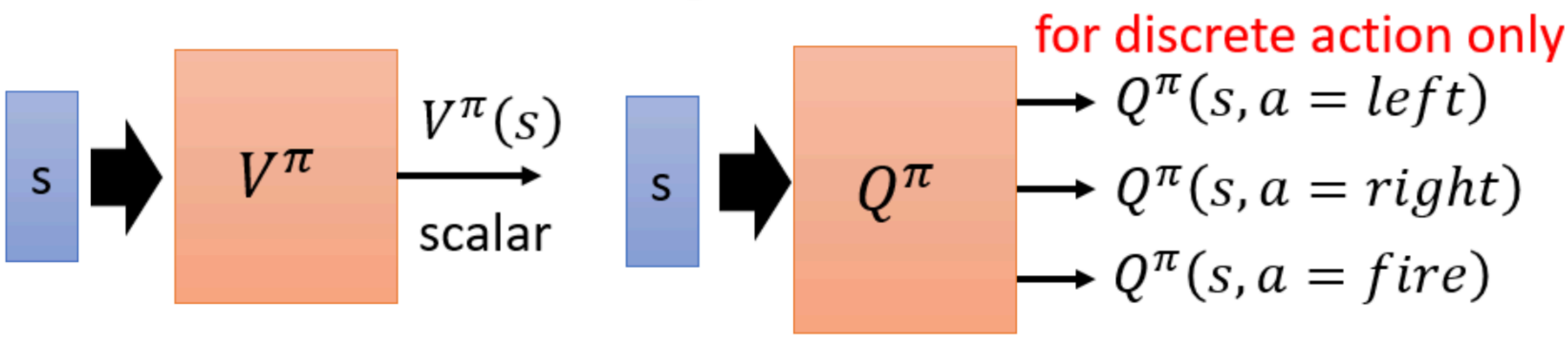
With sufficient samples, approximate the expectation of G.



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



Estimated by TD or MC

Actor-Critic

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

Actor-Critic

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

G_t^n : obtained via interaction

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

Actor-Critic

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

baseline

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

G_t^n : obtained via interaction

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

Actor-Critic

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

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

baseline

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

G_t^n : obtained via interaction

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

Advantage Actor-Critic

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

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.

$$\Downarrow$$

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

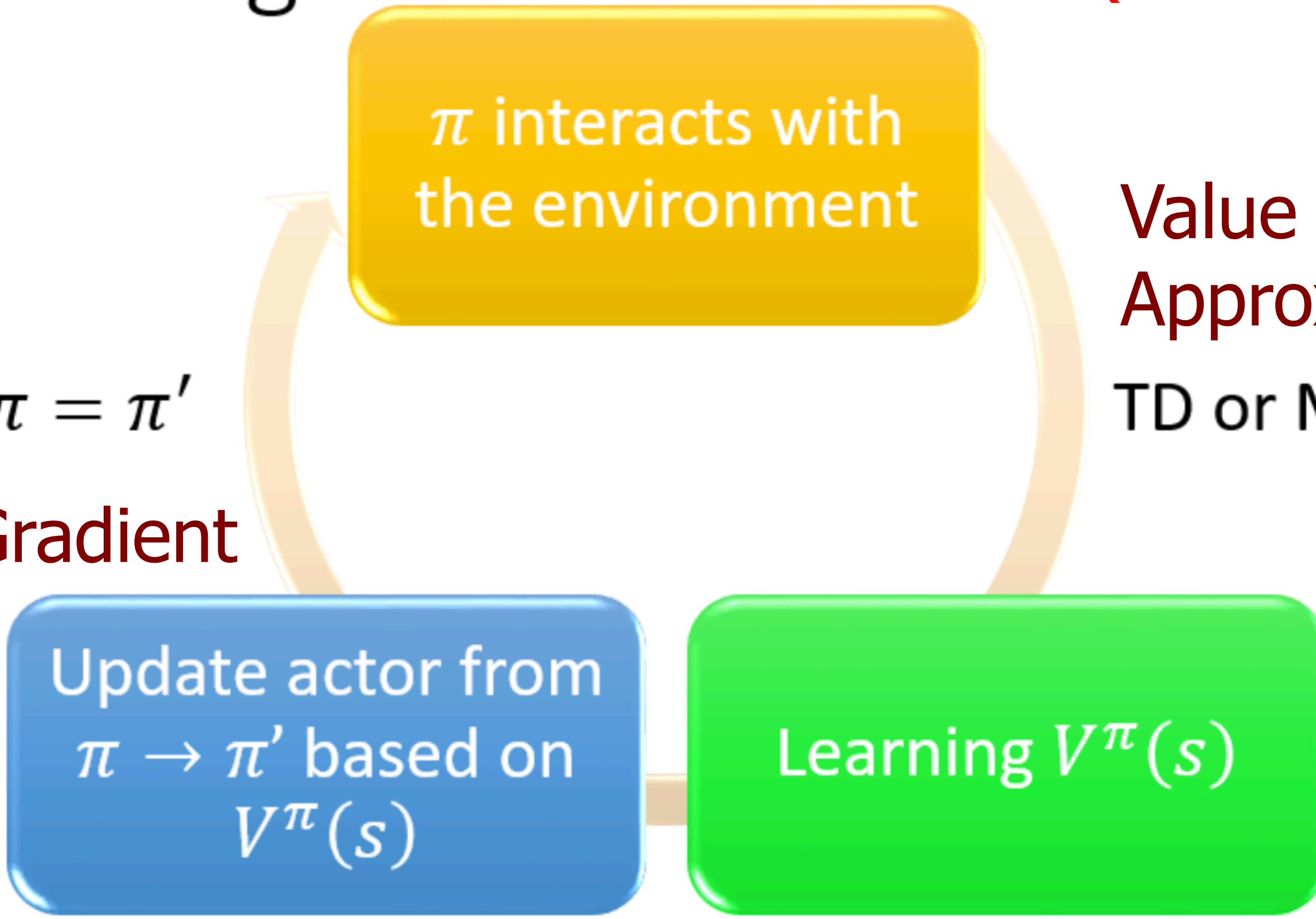
Only estimate state value

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

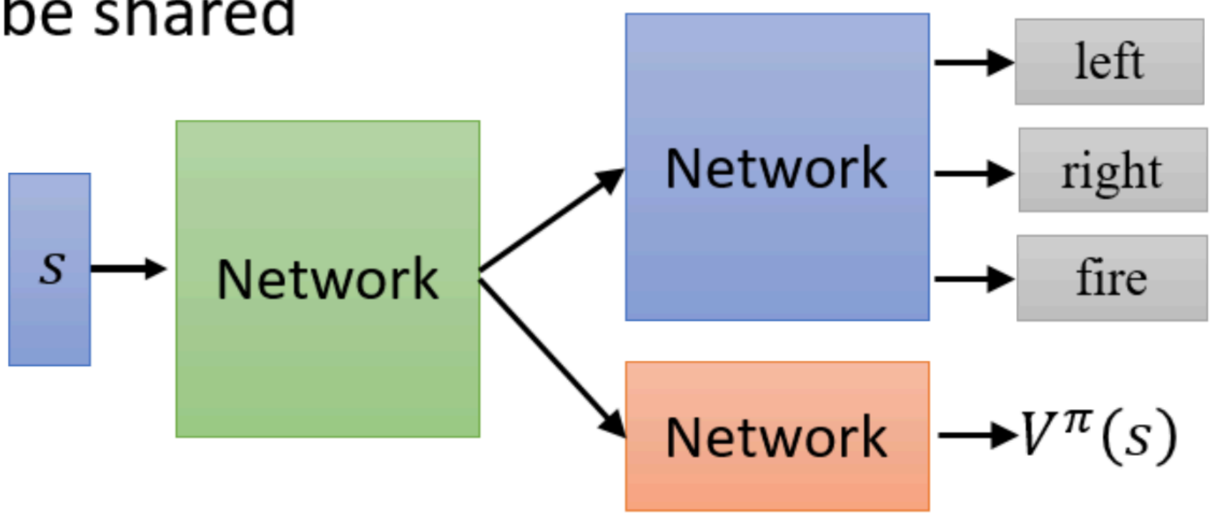
$\pi = \pi'$

Policy Gradient

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

Advantage Actor-Critic

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

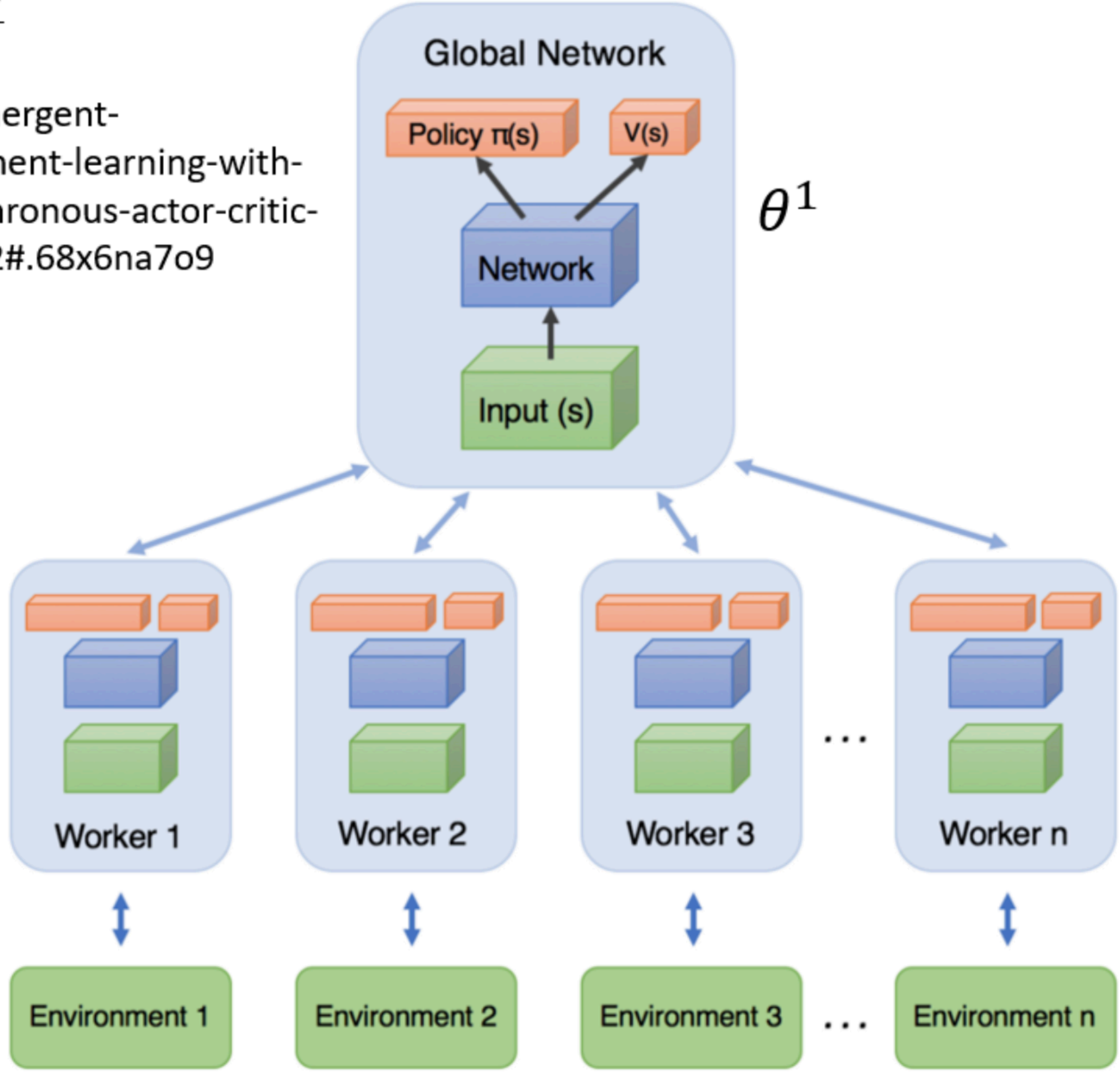


  - Use output entropy as regularization for $\pi(s)$
    - Larger entropy is preferred → exploration

Asynchronous Advantage Actor-Critic (A3C)

Asynchronous

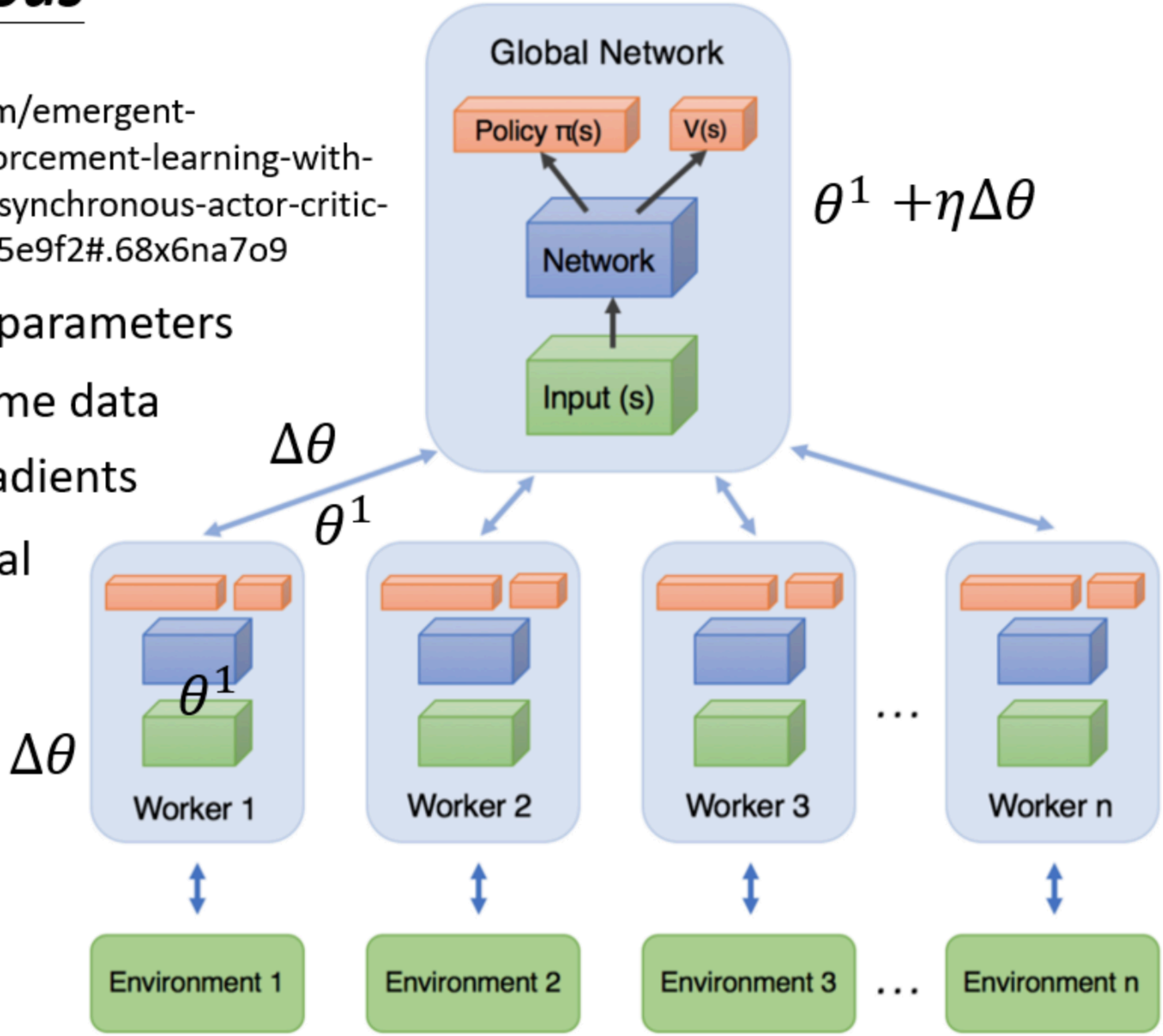
Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9



Asynchronous

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

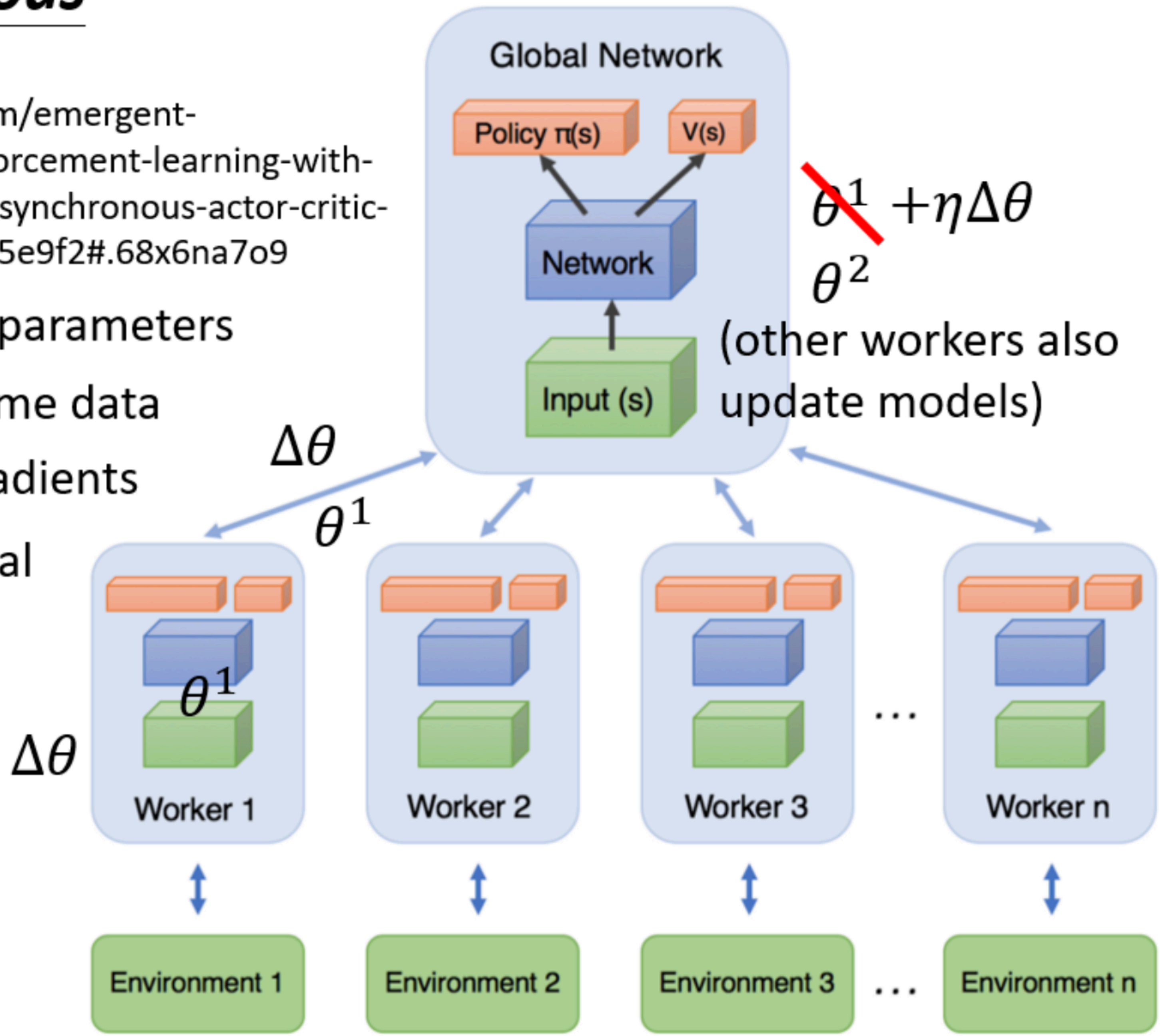
1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



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



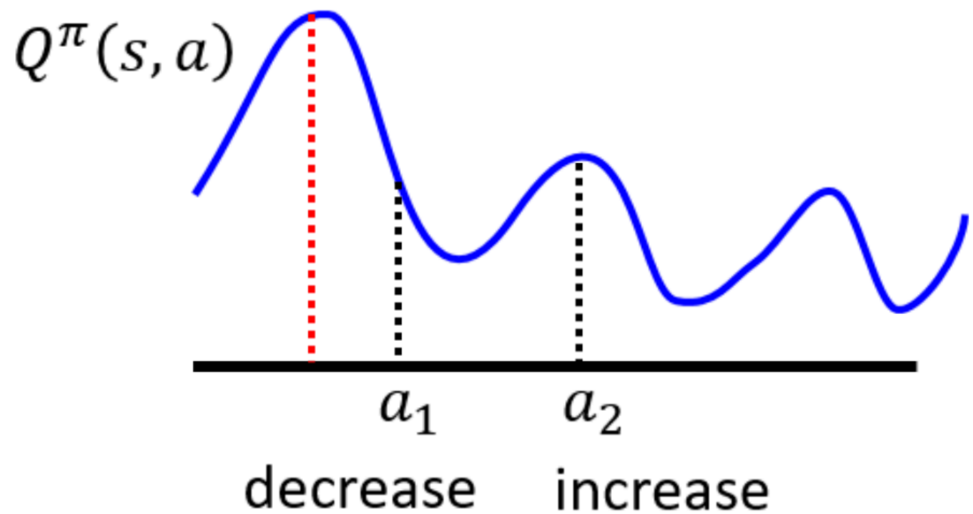
Pathwise Derivative Policy Gradient

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

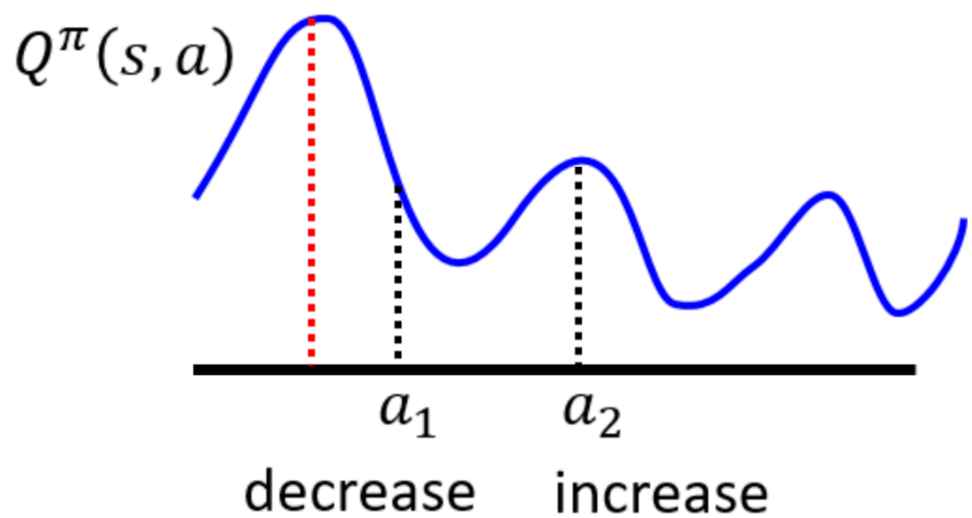
Another Way to use Critic

<u>Original Actor-critic</u>



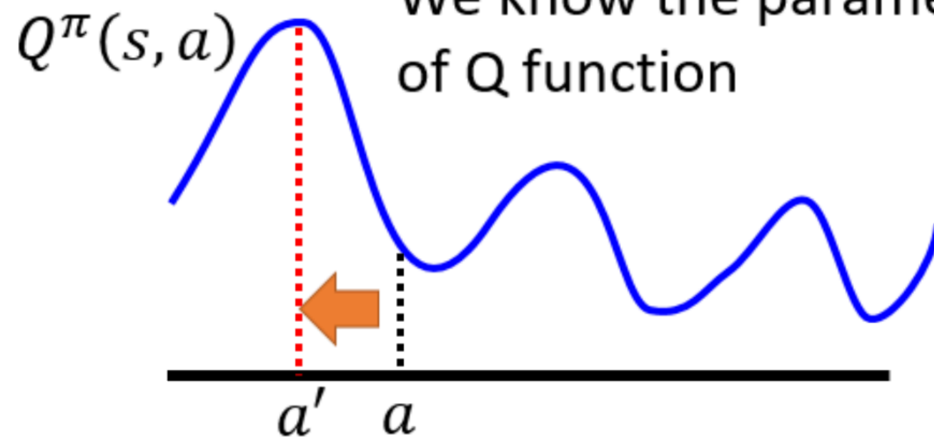
Another Way to use Critic

<u>Original Actor-critic</u>



<u>Pathwise derivative policy gradient</u>

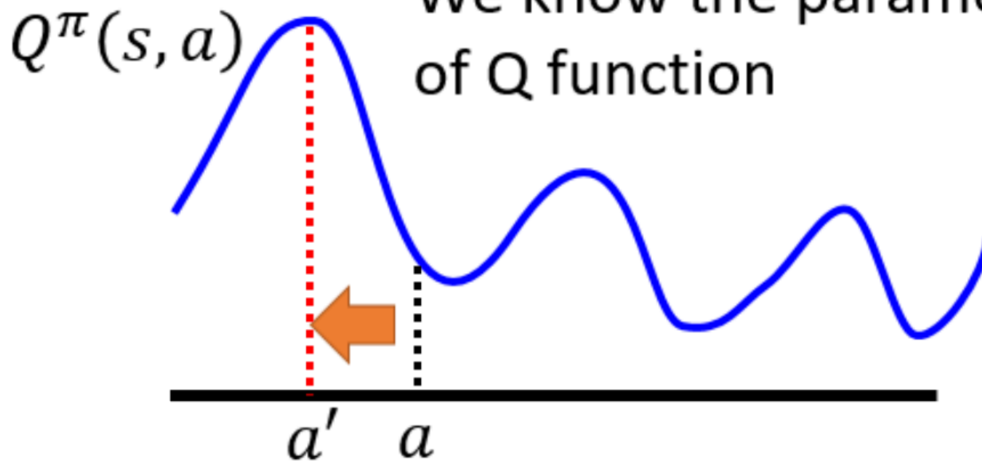
We know the parameters of Q function



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

Pathwise derivative policy gradient

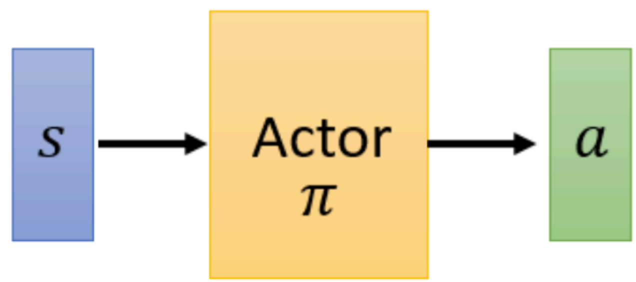
From Q function we know that taking a’ at state s is better than a



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

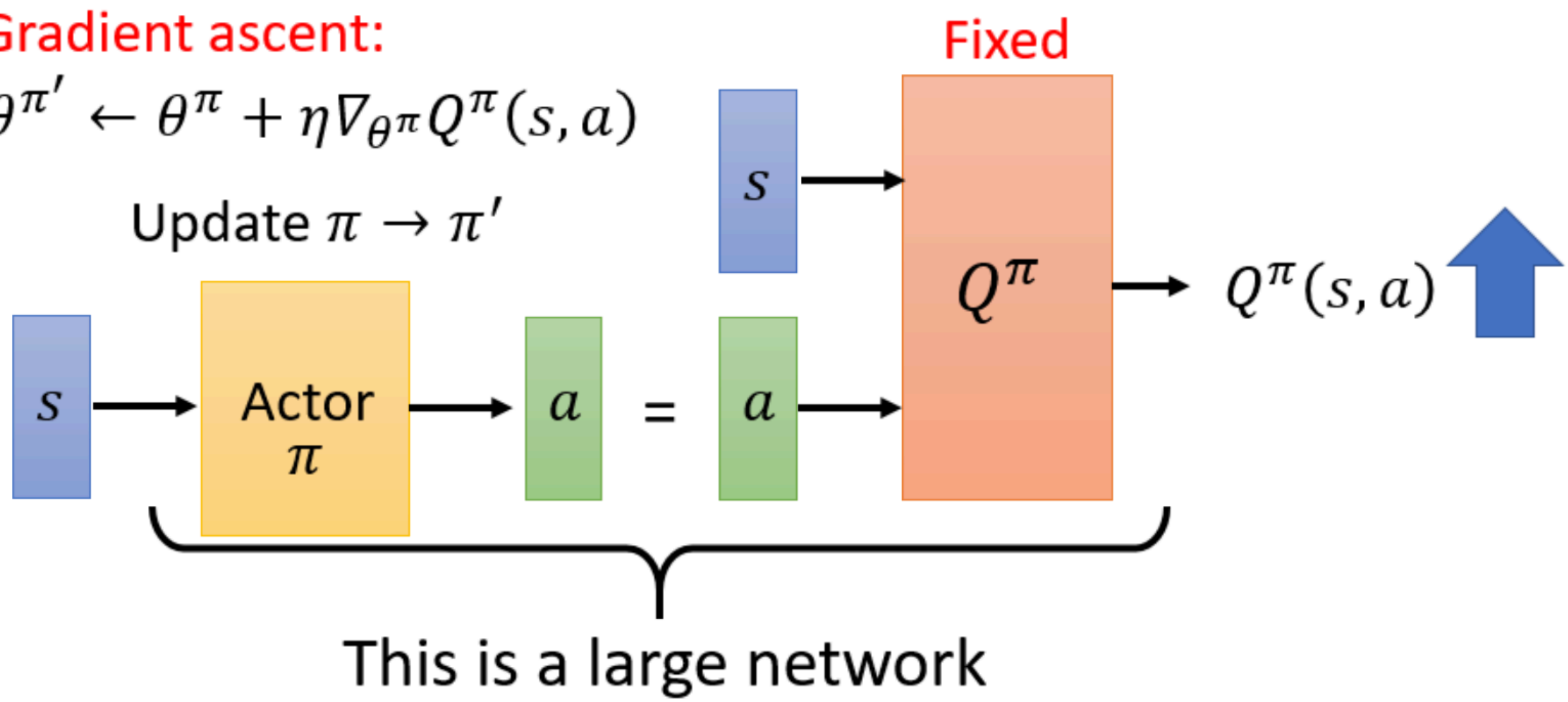
Pathwise Derivative Policy Gradient

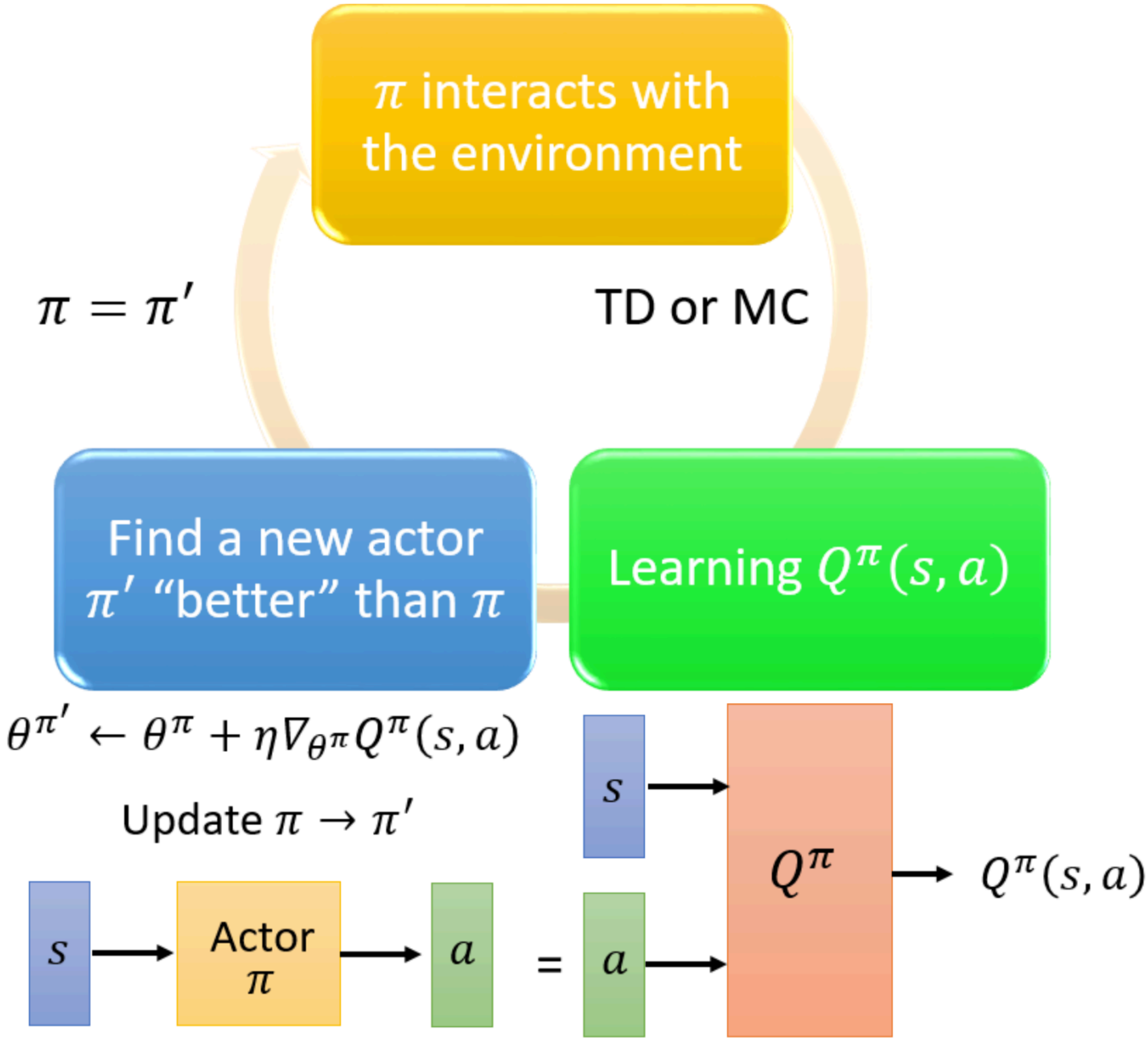
$\pi'(s) = arg \max_a Q^\pi(s,a)$ a is the output of an actor

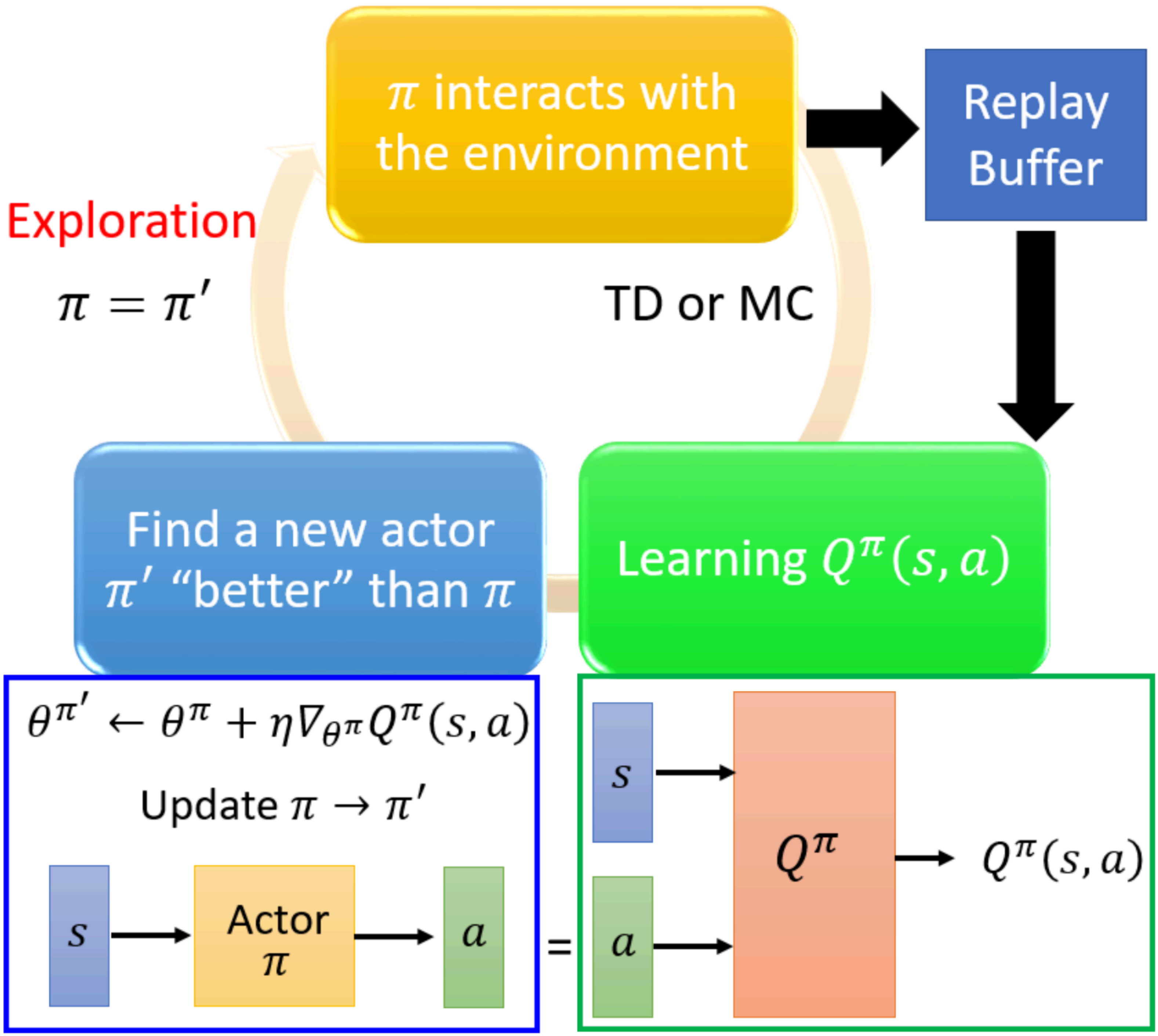
Gradient ascent:

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

Update $\pi \rightarrow \pi'$




π interacts with the environment
TD or MC
$\pi = \pi'$
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
=
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
$Q^\pi(s,a)$

<u>Q-Learning Algorithm</u>

- 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

Replaced ε-greedy policy with π network.

  - 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

Next Lecture

-12. Week 12 (11/7 R): (Prof Li is on a travel, and invited PhD student speakers will give research work presentations)

Topic: RL and IRL Applications: Research work presentations from PhD students from Prof Li's group, by Menghai Pan and Xin Zhang.
Work #1. [SDM'19] **Menghai Pan**, Yanhua Li, Xun Zhou, Zhenming Liu, Rui Song, Hui Lu, Jun Luo, Dissecting the Learning Curve of Taxi Drivers: A Data-Driven Approach. SIAM International Conference on Data Mining, (SDM'19 Best Applied Data Science Paper Award!) (Paper PDF)
Work #2. [ICDM'19] **Xin Zhang**, Yanhua Li, Xun Zhou, Jun Luo, Unveiling Taxi Drivers' Strategies via cGAIL -- Conditional Generative Adversarial Imitation Learning, IEEE International Conference on Data Mining (Paper PDF).
Work #3. A work under double-blind review by **Xin Zhang**.

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** 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

Questions?