This Lecture will be recorded!!!

Welcome to

DS595
Reinforcement Learning

Prof. Yanhua Li

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

No Quiz Today

Office hour time change in next 2 weeks (Prof. Li)

- Prof Li's office hour

- From Tue 10:00am-11:00am;
- To Wed 11:00am-12pm
- with the same Zoom Link as course lecture
- for Week 11 Wed 3/30
- and Week 12 Wed 4/6.

- From Week 13, we resume it to Tue 10-11AM
- Other time slots by appointments

Class arrangement

https://users.wpi.edu/~yli15/courses/DS595Spring22/Schedule.html
Quiz 5 on Week #12 4/6/2022 (the Wed after next week).
30 minutes on Policy Gradient (PG)

Project 3 reminder:
Due 4/6 Next Wed
10 bonus points and a leader board

- https://users.wpi.edu/~yli15/courses/DS595Spring22/Assignments.html

- https://github.com/yingxue-zhang/DS595-RL-Projects/tree/master/Project3

Project 4 proposal due today

Group confirmation.

Project 4 is available
Starts 3/23 this Wed

- https://github.com/yingxue-zhang/DS595-RL-Projects/tree/master/Project4
- *Important Dates:*
- **Timeline:**
  Week 11 (3/30 W), Proposal Due. (Upload it to Canvas Discussion board)
  Week 13 (4/13 W), Progressive report due (Upload it to Canvas discussion board)
  Week 15 (4/25 M), Project report due. (Upload it to Canvas discussion board)
  Week 15 (4/27 W), Project poster session. (On Zoom)

Last Lecture

- Advanced DQN methods
  - Double-DQN
  - Dueling DQN
  - Prioritized DQN
  - Multi-step
  - Noisy net
  - Distributional Q-learning
  - Rainbow
  - Continuous actions
- Self-Introduction
- Imitation Learning / Inverse Reinforcement Learning
  - Introduction
  - Behavioral Cloning
  - Inverse reinforcement learning
    - Model-Based, Linear Reward Functions (this time)

This 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

Problems with many RL scenarios

- Reinforcement Learning:
  - Learning policies guided by (often sparse) rewards (e.g. win the game or not)
  - Pros: simple, cheap form of supervision / exploration
  - Cons: High sample complexity

Problems with many RL scenarios

- Where is it successful?
  - In simulation where data is cheap and parallelization is easy

- Not when:
  - Execution of actions is slow
  - Very expensive or not tolerable to fail
  - Want to be safe

Learning from Demonstrations (LfD)

- Expert provides a set of **demonstration trajectories**: sequences of states and actions
- Imitation learning is useful when is easier for the expert to demonstrate the desired behavior rather than:
  - come up with a reward that would generate such behavior,
  - coding up the desired policy directly

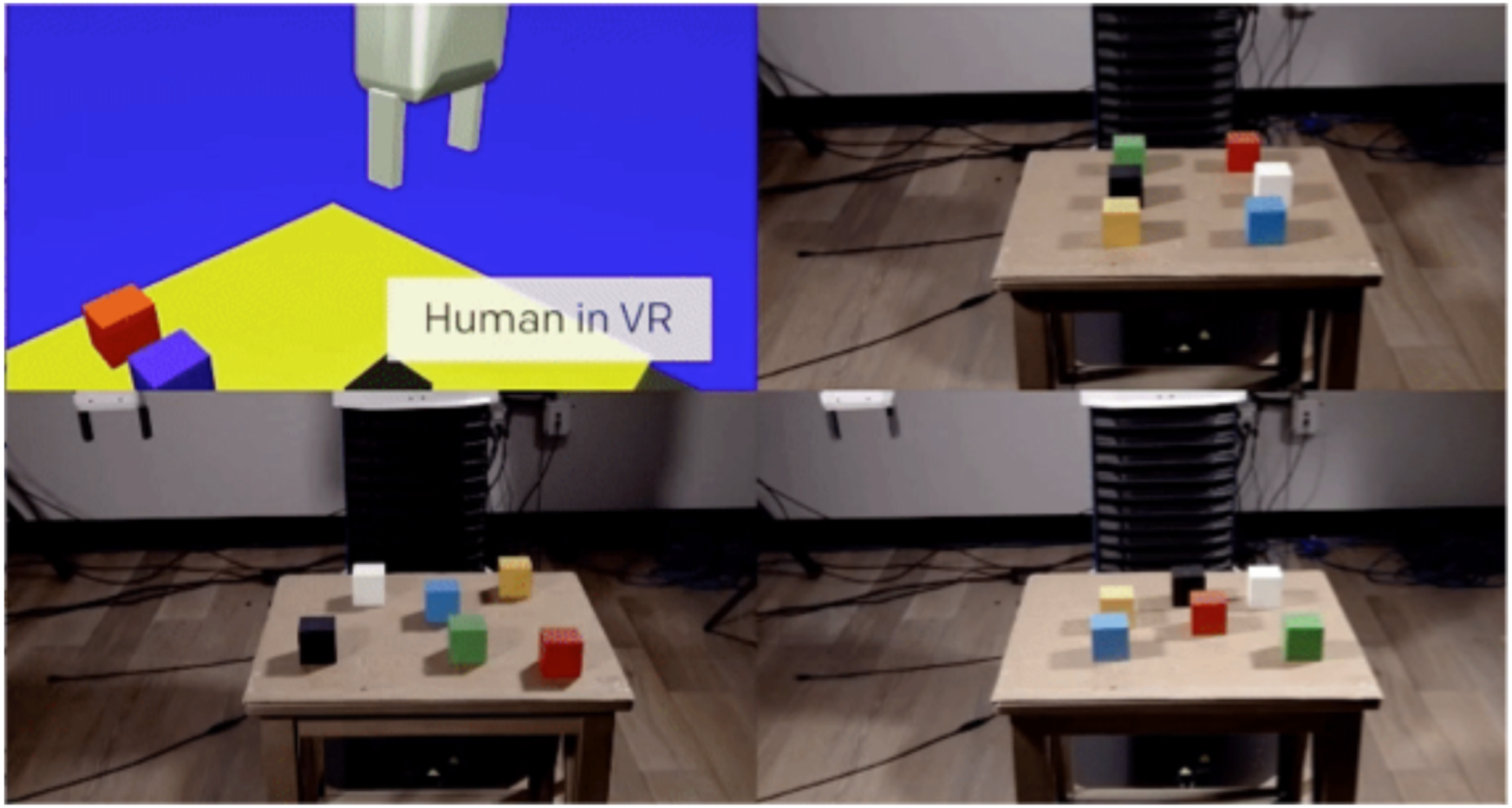
- Learning two things from imitation learning:
  - Policy
  - Reward function (why?)

Learning from Demonstrations (LfD)

- Expert provides a set of **demonstration trajectories**: sequences of states and actions
- Imitation learning is useful when is easier for the expert to demonstrate the desired behavior rather than:
  - come up with a reward that would generate such behavior,
  - coding up the desired policy directly

- Learning two things from imitation learning:
  - Policy
  - Reward function (why?)
    - Understand/reason how demonstrator makes decisions
    - Predict future behaviors
    - Good reward function for training RL agents

One Shot Imitation Learning



"ab," "cde," "fg," and "hij," where the blocks are ordered from top to bottom within each group. https://www.youtube.com/watch?v=Bc_kZ-OQh24

1. **Hard to define a reward function;**
2. **Hard to explore from a random policy.**

A Deep Learning Approach for Generalized Speech Animation
Sarah Taylor, Taehwan Kim, Yisong Yue et al., SIGGRAPH 2017

https://www.youtube.com/watch?v=9zL7qejW9fE

https://www.youtube.com/watch?v=9zL7qejW9fE

1. **Hard to define a reward function;**
2. **Hard to explore from a random policy.**

Problem Setup

Model Based for Now

- Input:
  - State space, action space
  - Transition model $P(s' \mid s, a)$
  - No reward function R
  - Set of one or more teacher's demonstrations $(s_0, a_0, s_1, s_0, \ldots)$ (actions drawn from teacher's policy π^*)
- Behavioral Cloning:
  - Can we directly learn the teacher's policy using supervised learning?
- Inverse RL:
  - Can we recover R?

We will discuss model-free (i.e., unknown P) in future lectures.

This Lecture

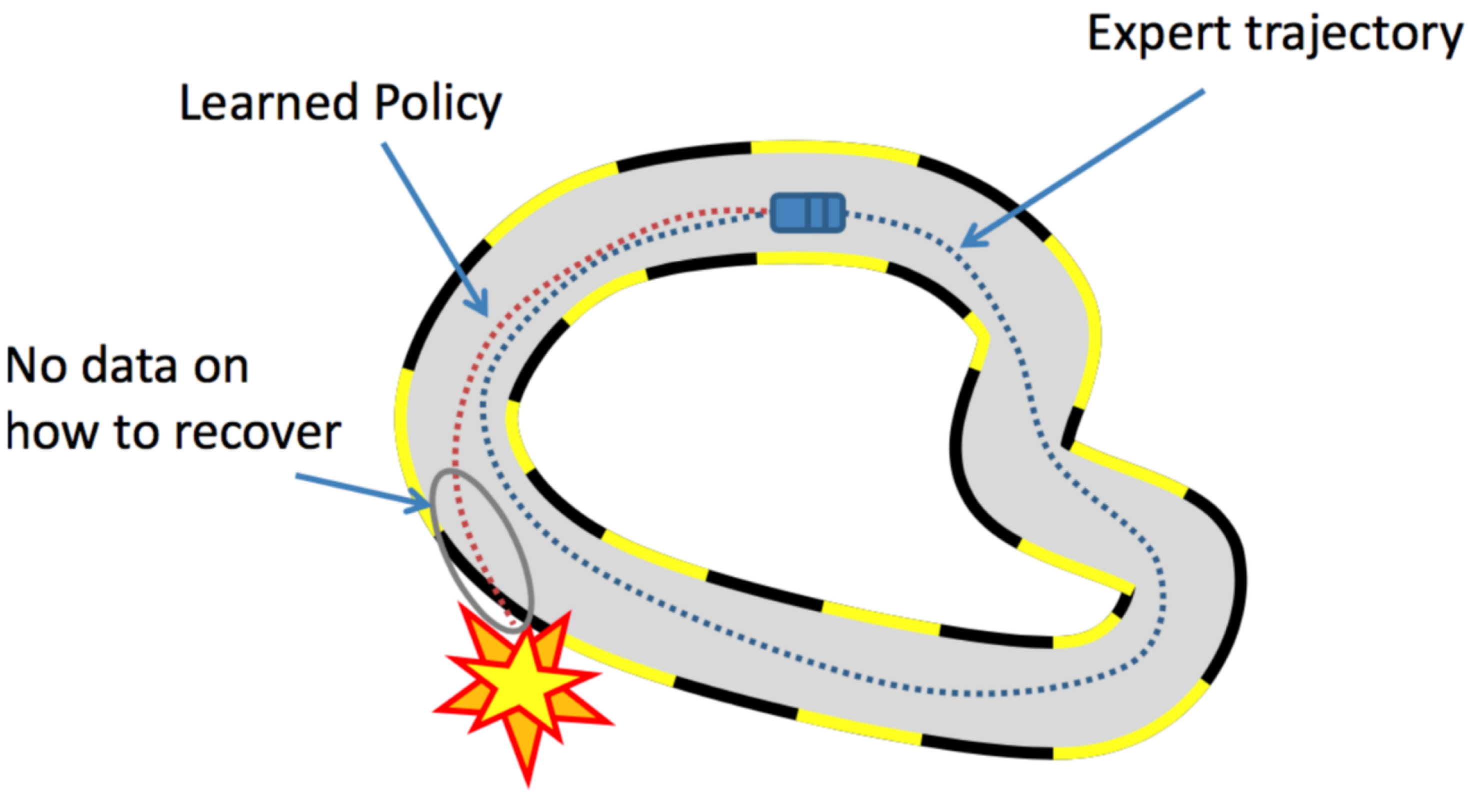
- Imitation Learning / Inverse Reinforcement Learning
  - Introduction
  - Behavioral Cloning (Learning expert policy)
  - Inverse reinforcement learning
    - Model-Based, Linear Reward Functions (this time)
- Policy Gradient
  - Intro and Stochastic Policy
  - Basic Policy Gradient Algorithm
  - Vanilla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

Behavioral Cloning

- Formulate problem as a standard machine learning problem:
  - Fix a policy class (e.g. neural network, decision tree, etc.)
  - Estimate a policy from training examples $(s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots$

Problem with the BC approach?

Problem: Compounding Errors



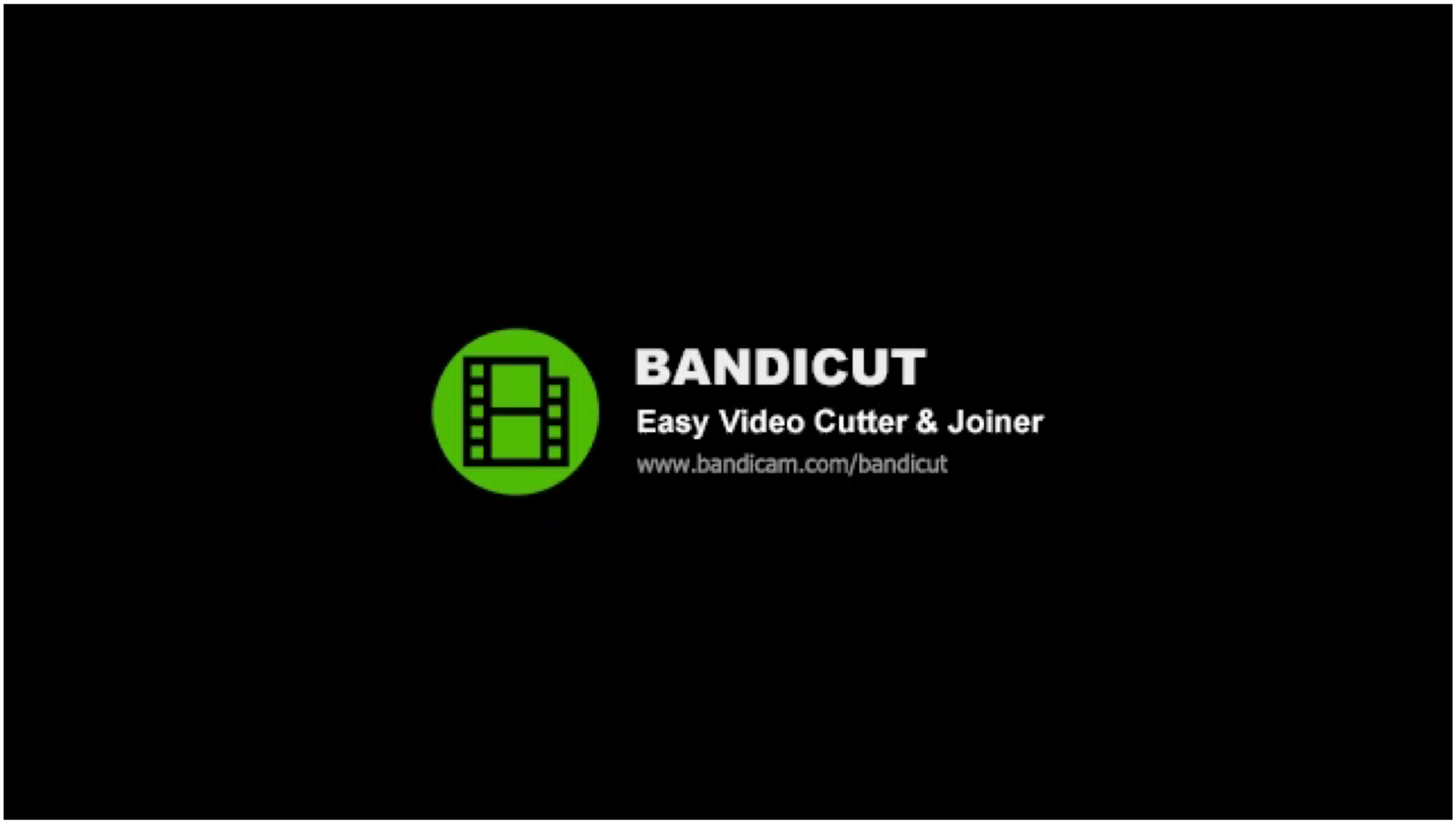
Data distribution mismatch!

In supervised learning, $(x, y) \sim D$ during train **and** test. In MDPs:

- Train: $s_t \sim D_{\pi^*}$
- Test: $s_t \sim D_{\pi_\theta}$

Behavior Cloning

The agent will copy every behavior, even irrelevant actions.



https://www.youtube.com/watch?v=j2FSB3bseek

This 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

Linear Feature Reward Inverse RL

- Recall linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \to \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi]$$

Linear Feature Reward Inverse RL

- Recall linear value function approximation
- Similarly, here consider when reward is linear over features
  - $R(s) = \boldsymbol{w}^T x(s)$ where $w \in \mathbb{R}^n, x : S \to \mathbb{R}^n$
- Goal: identify the weight vector $\boldsymbol{w}$ given a set of demonstrations
- The resulting value function for a policy π can be expressed as

$$V^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi] \quad = \mathbb{E}[\textstyle\sum_{t=0}^{\infty} \gamma^t \boldsymbol{w}^T x(s_t) \mid \pi]$$

$$= w^T \mathbb{E}[\textstyle\sum_{t=0}^{\infty} \gamma^t x(s_t) \mid \pi]$$

$$= \boldsymbol{w}^T \mu(\pi)$$

where $\mu(\pi)(s)$ is defined as the discounted weighted frequency of state features under policy π.

Inverse Reinforcement Learning

To find the reward function R used by the expert:

- Note
  $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi^*] = V^* \geq V^\pi = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) \mid \pi] \quad \forall \pi,$
- Therefore if the expert's demonstrations are from the optimal policy, to identify $\boldsymbol{w}$ it is sufficient to find w^* such that

$$w^{*T} \mu(\pi^*) \geq w^{*T} \mu(\pi), \forall \pi \neq \pi^*$$

Inverse reinforcement learning

- Goal: Learn a policy function and a reward function that are as good as the demonstration expert
- Linear reward function assumption: $R(s) = w^T x(s)$

  - Initialize $\pi=\pi_0$, stopping criteria $\varepsilon=10^{-3}$ (for example)
  - For i=1,2,… …
    - Find a reward function that the expert maximally outperforms previous policies: (Any quadratic programming solver)

      $$\arg\max_{\mathrm{w}}(\mathrm{w}^\top \mu(\pi^*) - \mathrm{w}^\top \mu(\pi))\text{, s.t., } \|\mathrm{w}\|_2 \leq 1$$

    - Find the optimal π with the current w (dynamic programming)
    - Exit if $\mathrm{w}^\top \mu(\pi^*) - \mathrm{w}^\top \mu(\pi) \leq \epsilon/2$

Suppose it is model-based, i.e., environment dynamics is known.

More on Imitation Learning

- Slides: https://drive.google.com/file/d/12QdNmMII-bGISWnm8pmD_TawuRN7xagX/view
- Video: https://www.youtube.com/watch?v=WjFdD7PDGw0

Imitation Learning

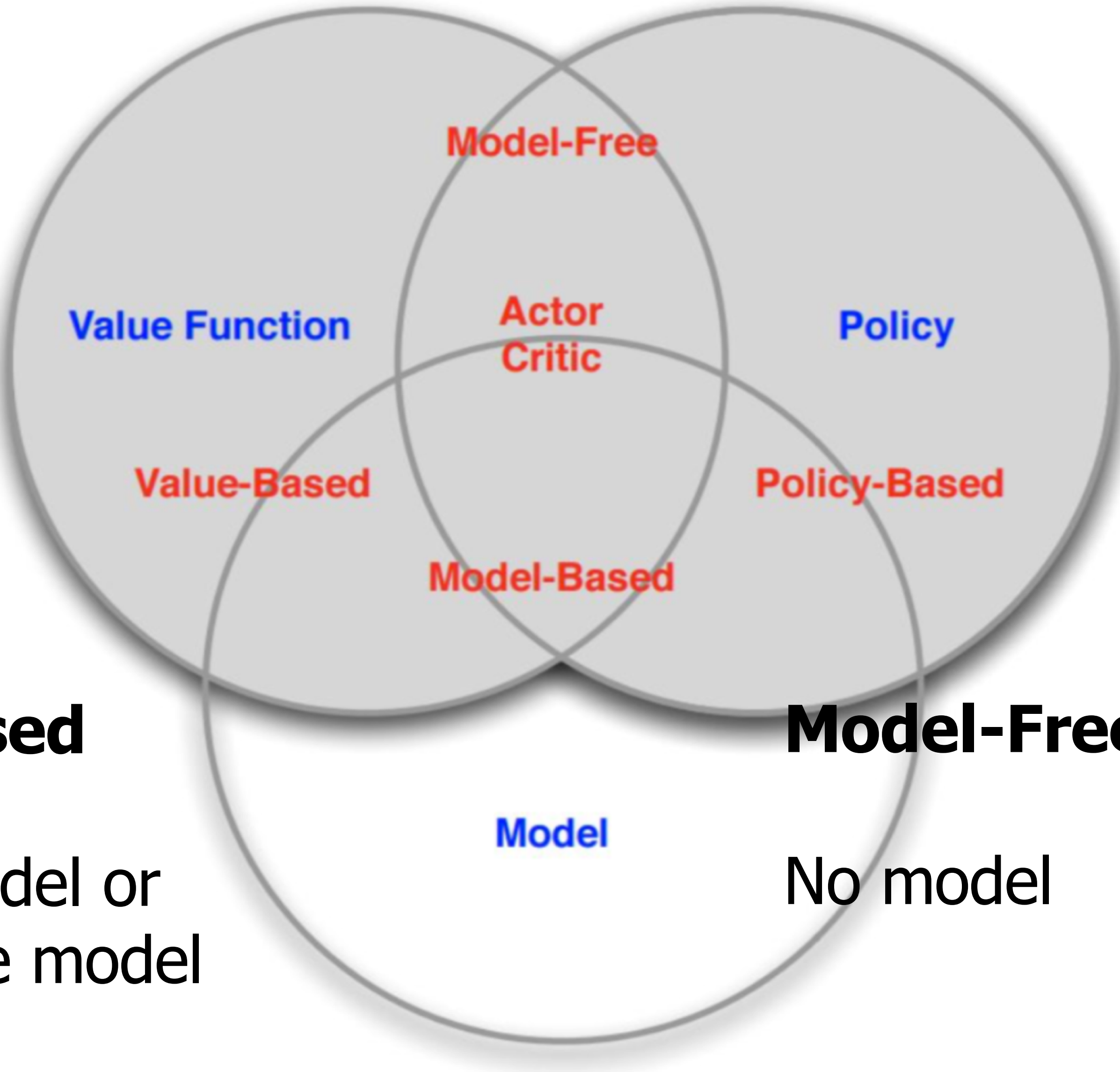
ICML 2018 Tutorial
(Slides Available Online)

Yisong Yue	Hoang M. Le
	
yyue@caltech.edu	hmle@caltech.edu
@YisongYue	@HoangMinhLe
yisongyue.com	hoangle.info

This 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
  - REINFORCE and Vanilla Policy Gradient
  - PPO, TRPO, PPO2

Types of RL agents/algorithms



Model-based

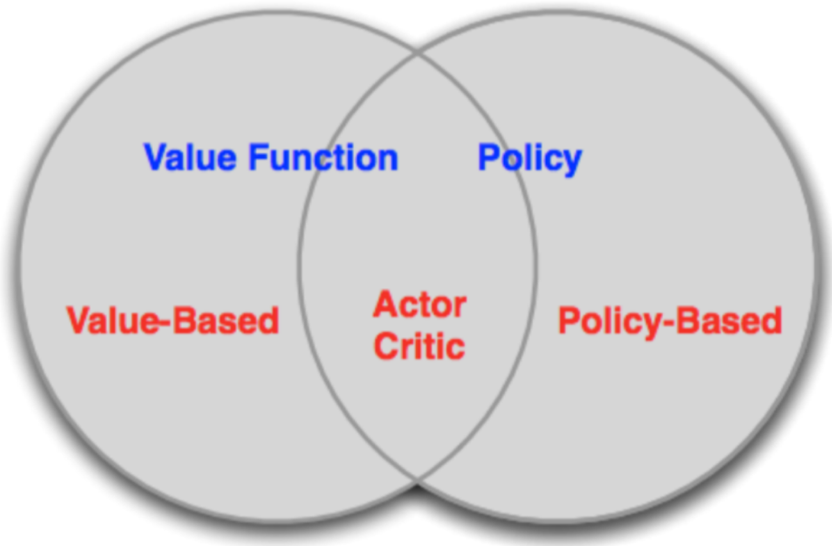
Explicit: Model or learning the model first.

Model-Free:

No model

Value-Based and Policy-Based RL

- 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



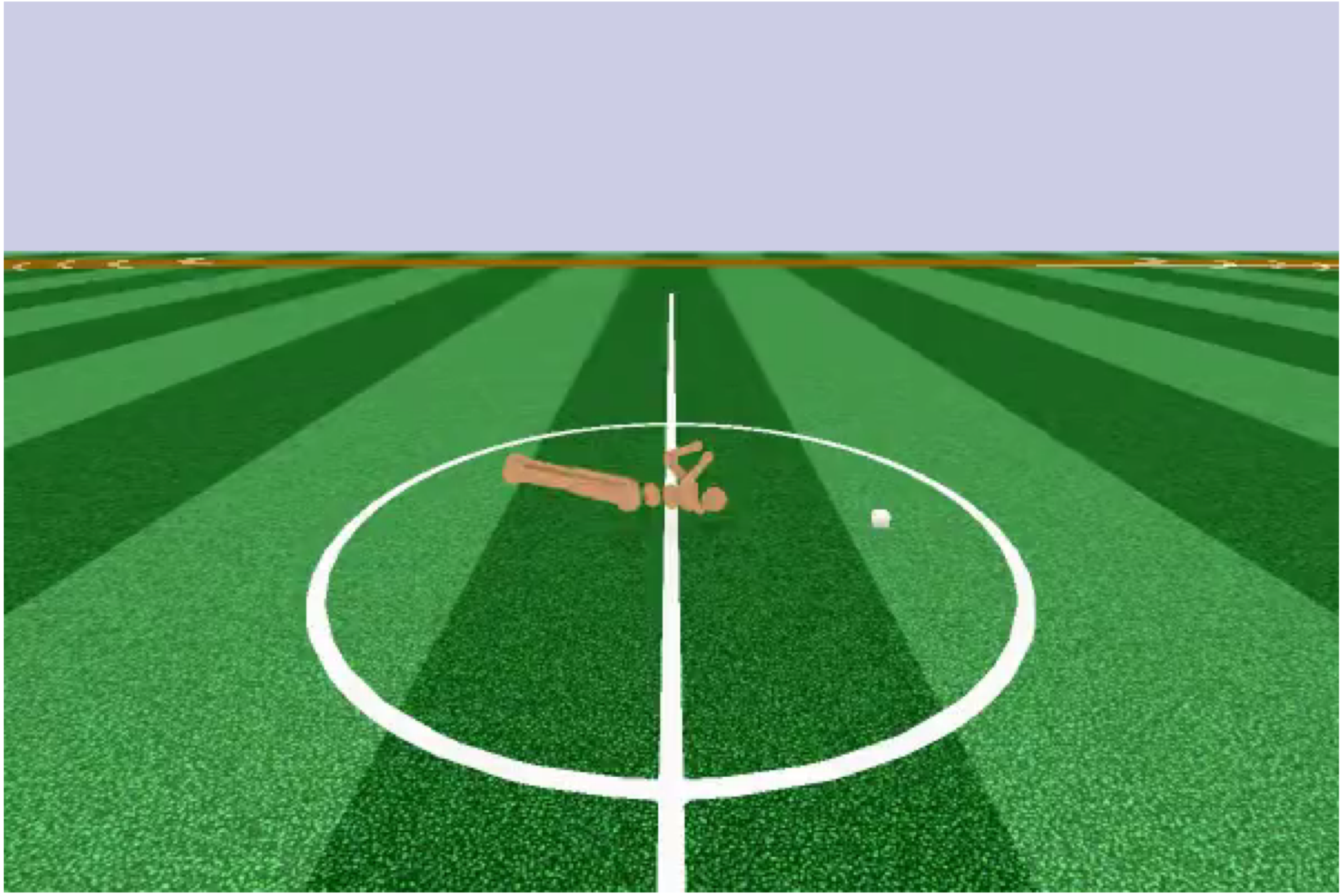
DeepMind

https://youtu.be/gn4nRCC9TwQ

OpenAI

https://blog.openai.com/openai-baselines-ppo/

PPO (Proximal Policy Optimizaiton) default reinforcement learning algorithm at OpenAI



Advantages of Policy-Based RL

Advantages of Policy-Based RL

Advantages:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

Disadvantages:

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

Advantages of Policy-Based RL

Advantages:

- Better convergence properties
- Effective in high-dimensional or continuous action spaces
- Can learn stochastic policies

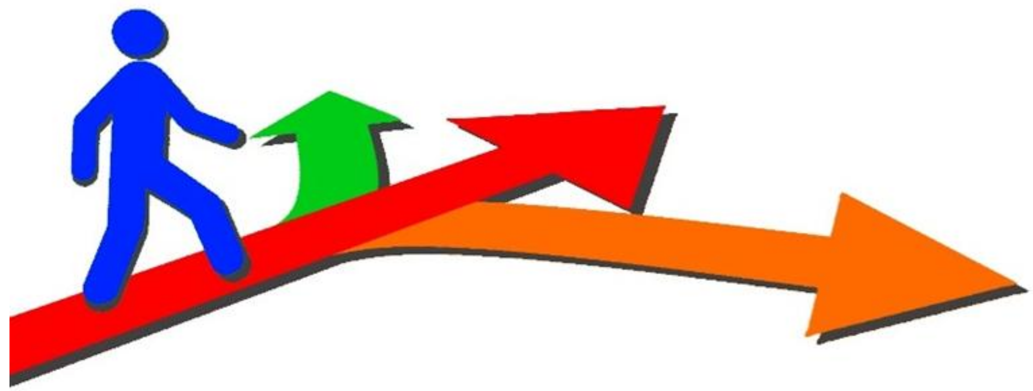
Disadvantages:

- Typically converge to a local rather than global optimum
- Evaluating a policy is typically inefficient and high variance

Stochastic Policy Example #1: Modeling Human Decisions

- Human make decisions under bounded rationality.

Trading Stocks



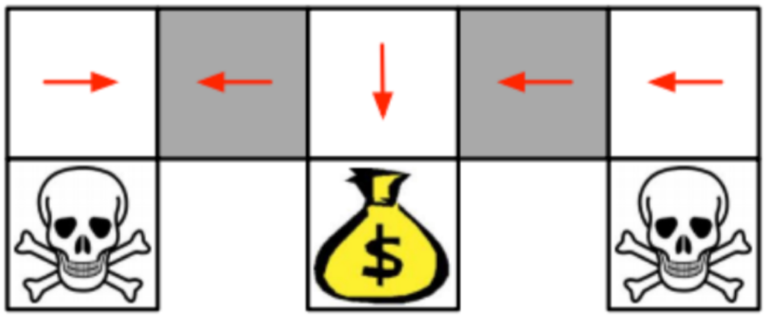
Route choices

Stochastic Policy Example #2: Rock-Paper-Scissors

- Two-player game of rock-paper-scissors
  - Scissors beats paper
  - Rock beats scissors
  - Paper beats rock
- Consider policies for iterated rock-paper-scissors
  - A deterministic policy is easily exploited
  - A uniform random policy is optimal (i.e. Nash equilibrium)

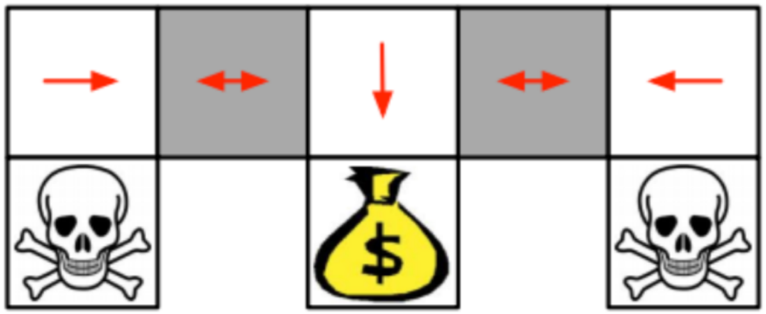
Stochastic Policy Example #3: Aliased Grid world

Gray state={Wall to N and S}



- Under aliasing, an optimal deterministic policy will either
  - move W in both grey states (shown by red arrows)
  - move E in both grey states
- Either way, it can get stuck and never reach the money
- Value-based RL learns a near-deterministic policy
  - e.g. greedy or ϵ-greedy
- So it will traverse the corridor for a long time

Stochastic Policy Example #3: Aliased Grid world



- An optimal stochastic policy will randomly move E or W in grey states

$$\pi_\theta(\text{wall to N and S, move E}) = 0.5$$

$$\pi_\theta(\text{wall to N and S, move W}) = 0.5$$

- It will reach the goal state in a few steps with high probability
- Policy-based RL can learn the optimal stochastic policy

Policy Objective Functions

- Goal: given a policy $\pi_\theta(s, a)$ with parameters θ, find best θ
- But how do we measure the quality for a policy π_θ?

- DQN: Deep Q-Learning

$$\nabla_w J(w) = \nabla_w \mathbb{E}_\pi[(Q(s,a) - \hat{Q}(s,a;w)^2]$$

+ Reply buffer
+ Fixed target Q

1: Initialize $\mathbf{w} = \mathbf{0}$, $k = 1$
2: **loop**
3: Sample tuple (s_k, a_k, r_k, s_{k+1}) given π
4: Update weights:

$$\Delta w = -\alpha(r_k + \gamma \max_{a_{k+1}} \hat{Q}(s_{k+1}, a_{k+1}; w) - \hat{Q}(s_k, a_k; w))\nabla_w \hat{Q}(s_k, a_k; w)$$

$$w = w - \Delta w$$

$$\pi(s_k) = \arg\max_{a_k} \hat{Q}(s_k, a_k), \text{with prob } 1 - \epsilon, \text{else random.}$$

5: $k = k + 1$
6: **end loop**

Policy Objective Functions

- Goal: given a policy $\pi_\theta(s, a)$ with parameters θ, find best θ
- But how do we measure the quality for a policy π_θ?

  - Maximize the value function: (focus on this case first)

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

  - It can be solved by gradient ascent, since we are maximizing the objective.

This 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
  - REINFORCE and Vanilla Policy Gradient
  - PPO, TRPO, PPO2

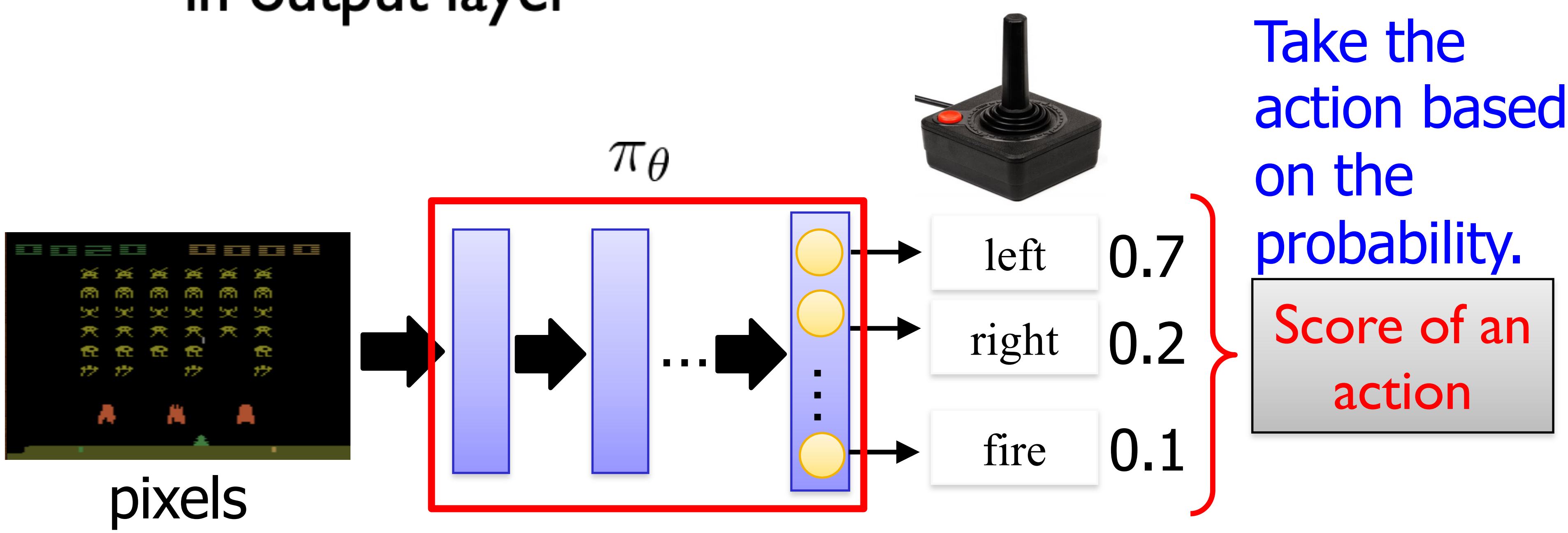
Basic Components

You cannot control

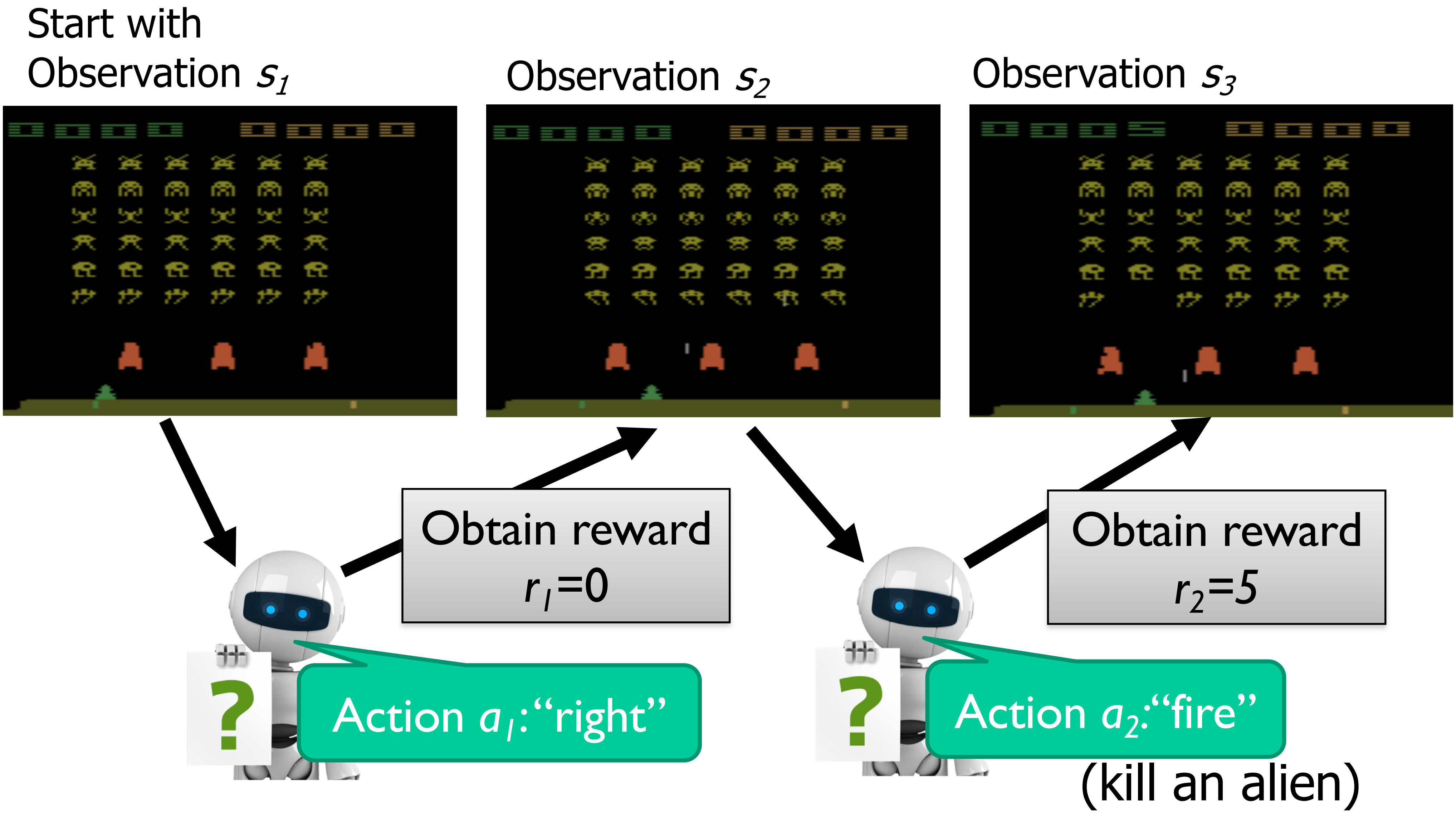
	Actor	Env	Reward Function
Video Game			Get 20 scores when killing a monster
Go	AlphaGo Google DeepMind		The rule of GO

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



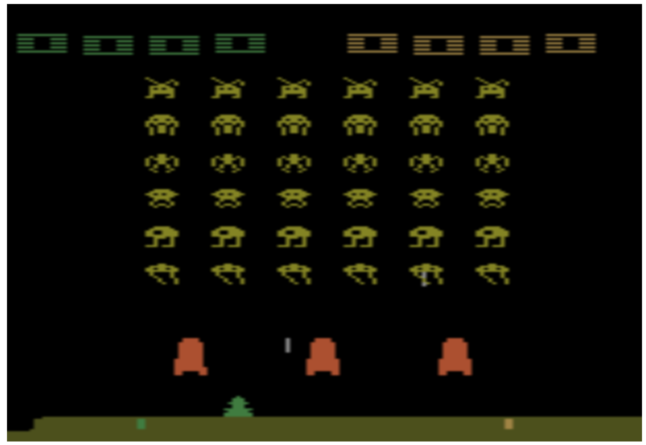
Example: Playing Video Game



Example: Playing Video Game

Start with observation s_1

Observation s_2



Observation s_3

After many turns

Game Over
(spaceship destroyed)

Obtain reward r_T

Action a_T

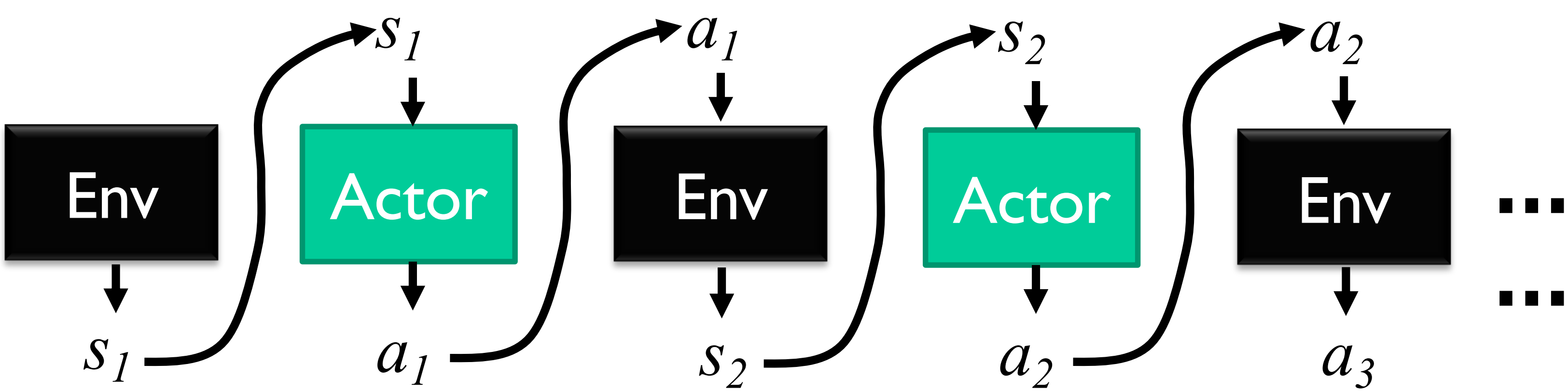
This is an ***episode***.

Total reward:

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

We want the total reward be maximized.

Actor, Environment, Reward



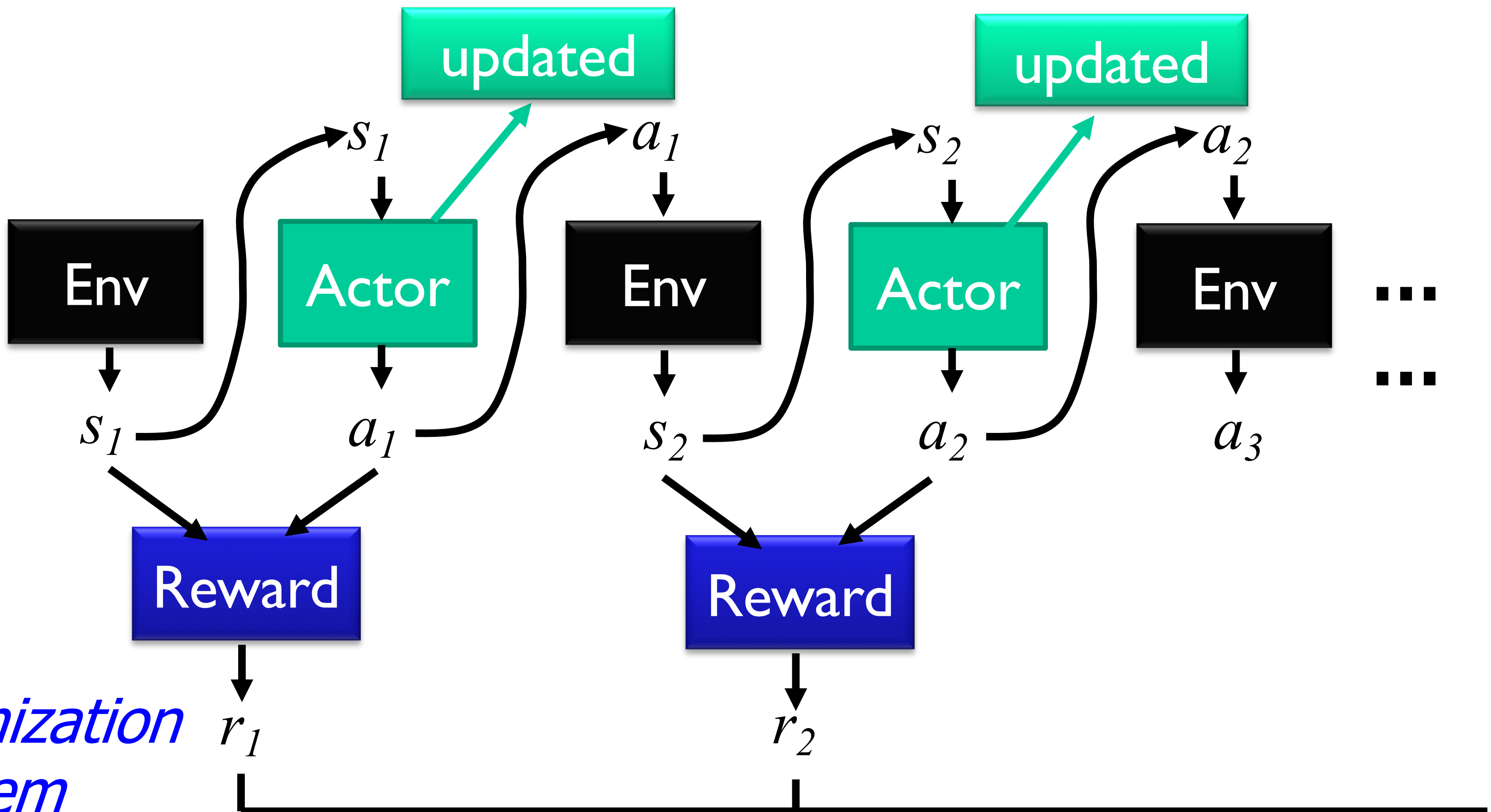
Trajectory $\tau=\{s_1, a_1, s_2, a_2, s_3, a_3, \ldots, s_T, a_T\}$

$$p_\theta(\tau)$$

$$= p(s_1)\pi_\theta(a_1|s_1)p(s_2|s_1,a_1)\pi_\theta(a_2|s_2)p(s_3|s_2,a_2)\cdots$$

$$= p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

Actor, Environment, Reward



Optimization Problem

Expected Reward (Objective to maximize)

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

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

Policy Gradient

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau) \qquad \nabla \bar{R}_\theta = ?$$

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla p_\theta(\tau) \quad = \sum_\tau R(\tau) p_\theta(\tau) \frac{\nabla p_\theta(\tau)}{p_\theta(\tau)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

Policy Gradient

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau) \qquad \nabla \bar{R}_\theta = ?$$

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla p_\theta(\tau) \quad = \sum_\tau R(\tau) p_\theta(\tau) \frac{\nabla p_\theta(\tau)}{p_\theta(\tau)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

$$= \sum_\tau R(\tau) p_\theta(\tau) \nabla log p_\theta(\tau)$$

$$\boxed{\nabla f(x) = f(x) \nabla log f(x)}$$

Policy Gradient

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau) \qquad \nabla \bar{R}_\theta = ?$$

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla p_\theta(\tau) \quad = \sum_\tau R(\tau) p_\theta(\tau) \frac{\nabla p_\theta(\tau)}{p_\theta(\tau)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

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

$$= \boxed{\sum_\tau} R(\tau) \boxed{p_\theta(\tau)} \nabla \log p_\theta(\tau)$$

$$= E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla \log p_\theta(\tau)] \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla \log p_\theta(\tau^n)$$

Policy Gradient

$$\bar{R}_\theta = \sum_\tau R(\tau) p_\theta(\tau) \qquad \nabla \bar{R}_\theta = ?$$

$$\nabla \bar{R}_\theta = \sum_\tau R(\tau) \nabla p_\theta(\tau) = \sum_\tau R(\tau) p_\theta(\tau) \frac{\nabla p_\theta(\tau)}{p_\theta(\tau)}$$

$R(\tau)$ do not have to be differentiable

It can even be a black box.

$$= \boxed{\sum_\tau} R(\tau) \boxed{p_\theta(\tau)} \nabla log p_\theta(\tau)$$

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

$$= E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla log p_\theta(\tau)] \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla log p_\theta(\tau^n)$$

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

$$= E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla log p_\theta(\tau)] \approx \frac{1}{N} \sum_{n=1}^{N} R(\tau^n) \nabla log p_\theta(\tau^n)$$

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

See Backup Slide #1 for the derivation.

Policy Gradient Algorithm

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

$$\nabla \bar{R}_\theta =$$

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

Policy Gradient Algorithm

Given policy π_θ

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

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

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

Basic Policy Gradient Algorithm

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

Update Model

Given policy π_θ

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

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

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

Data Collection

only used once

Implementation

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

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

$$\boxed{s_t^n \quad a_t^n \quad R(\tau^n)}$$

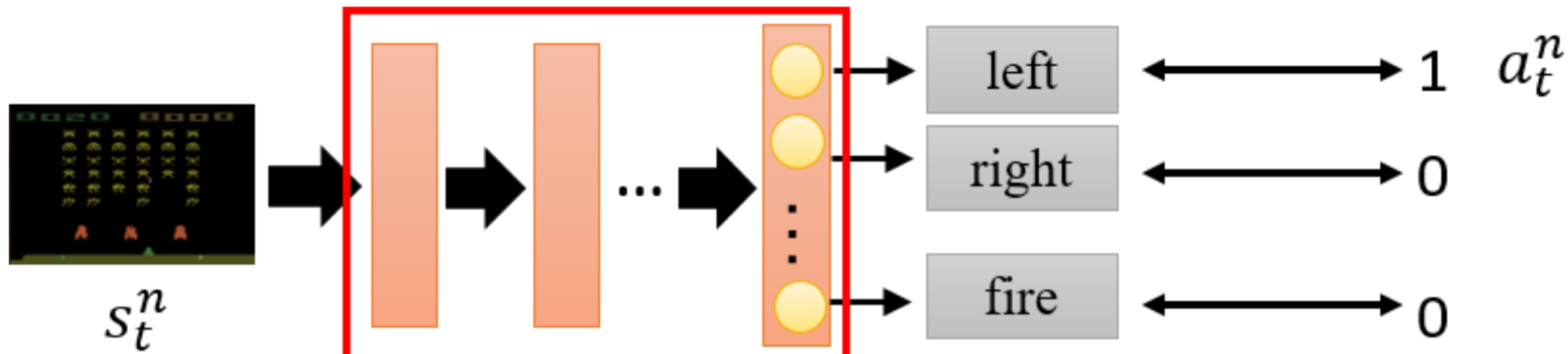
$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

Implementation

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

$s_t^n \quad a_t^n \quad R(\tau^n)$

Consider as classification problem



?

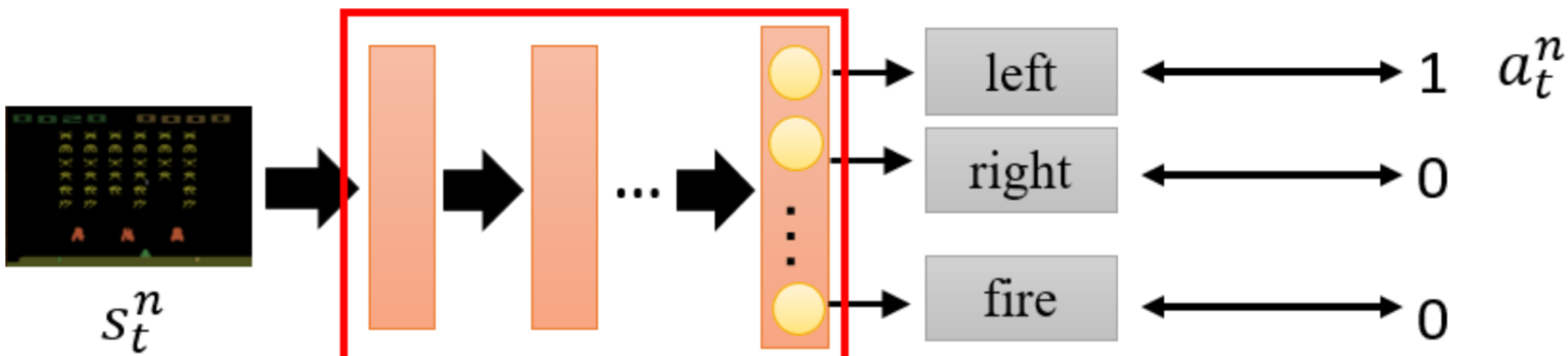
$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

Implementation

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

$s_t^n \quad a_t^n \quad R(\tau^n)$

Consider as classification problem



DNN for classification:

$$\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} log \pi_\theta(a_t^n | s_t^n) \xrightarrow{\text{TF, pyTorch ...}} \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \nabla log \pi_\theta(a_t^n | s_t^n)$$

DNN for policy gradient RL:

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

From basic PG algorithm to…?

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

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

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

Data Collection

From basic PG algorithm to…？

- Issues with the basic PG algorithm

From basic PG algorithm to…

- Issues with the basic PG algorithm
  - 1. Inaccurate update when non-negative rewards
  - 2. Large variance
  - 3. Slow, due to the un-reusable data collection process

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

This 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
  - REINFORCE and Vanilla Policy Gradient RL
  - PPO, TRPO, PPO2

Tip 1: Add a Baseline

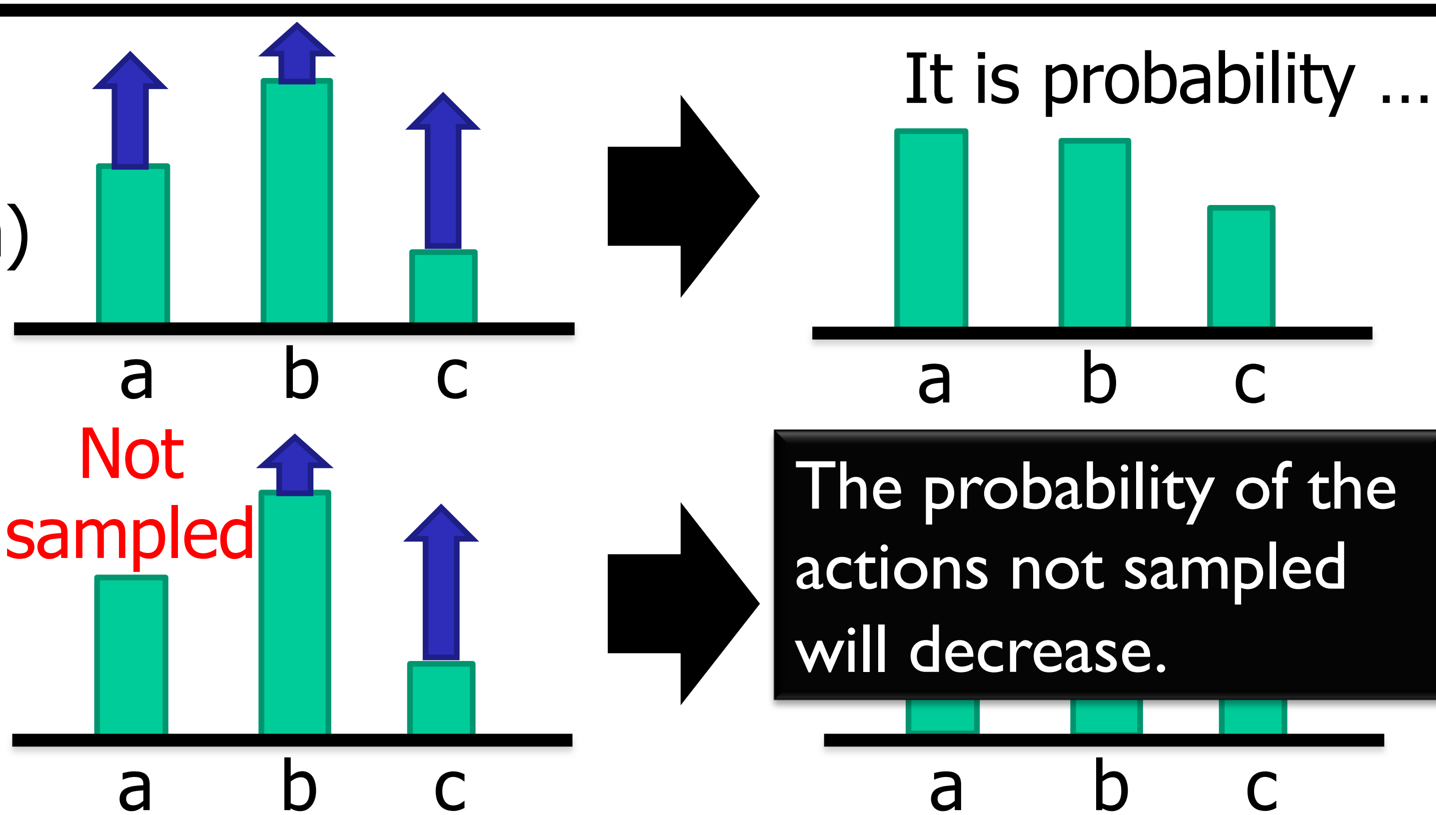
$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

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

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

Ideal Case (policy function)

With Sampling ……



Tip 2: Assign Suitable Credit

Return:	$\times 3$	$\boxed{\times -2}$	$\boxed{\times -2}$	$\times -7$	$\boxed{\times -2}$	$\boxed{\times -2}$
	(s_a, a_1)	(s_b, a_2)	(s_c, a_3)	(s_a, a_2)	(s_b, a_2)	(s_c, a_3)
Reward:	+5	+0	-2	-5	+0	-2
Total Reward:	$R = +3$			$R = -7$		

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

$$G_t = \sum_{t'=t}^{T_n} r_{t'}^n$$

Backup Slide #2 of why we can safely do this.

Tip 2: Assign Suitable Credit

Advantage Function $A^\theta(s_t, a_t)$

How good it is if we take a_t other than other actions at s_t.

Estimated by "*critic*" (later)

Can be state-dependent

$b(s_t) \approx \mathbb{E}[r_t + r_{t+1} + \cdots + r_{T-1}]$

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

$$G_t = \sum_{t'=t}^{T_n} r_{t'}^n \rightarrow \sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n$$

Add discount factor $\gamma < 1$

b(s) defined on states (average returns from a state) does not introduce bias.
(see backup slide #3 for proof).

Monte-Carlo Policy Gradient (REINFORCE)

TIP #2: Assign Suitable Credit by using returns

- Leverages likelihood ratio / score function and temporal structure

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

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**

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

  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* $G_t^n = \sum_{t'=t}^{T-1} r_{t'}$, and
  the *advantage estimate* $\hat{A}_t = G_t^n - b(s_t)$.
 (Re-fit the baseline, by minimizing $||b(s_t) - G_t^n||^2$,
  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

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

This 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
  - REINFORCE and Vanilla Policy Gradient
  - PPO, TRPO, PPO2

Quick Review

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

Basic Policy Gradient Algorithm

Given policy π_θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

$\vdots$ $\vdots$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

$\vdots$ $\vdots$

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

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

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

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**

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

  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 $||b(s_t) - G_t^n||^2$,
  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.

Given policy π_θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

$\vdots$ $\vdots$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

$\vdots$ $\vdots$

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

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

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

On-policy → Off-policy

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

- 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

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

- 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

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

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

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

$$= \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx = E_{x\sim q}[f(x)\underbrace{\frac{p(x)}{q(x)}}_{\text{Importance weight}}]$$

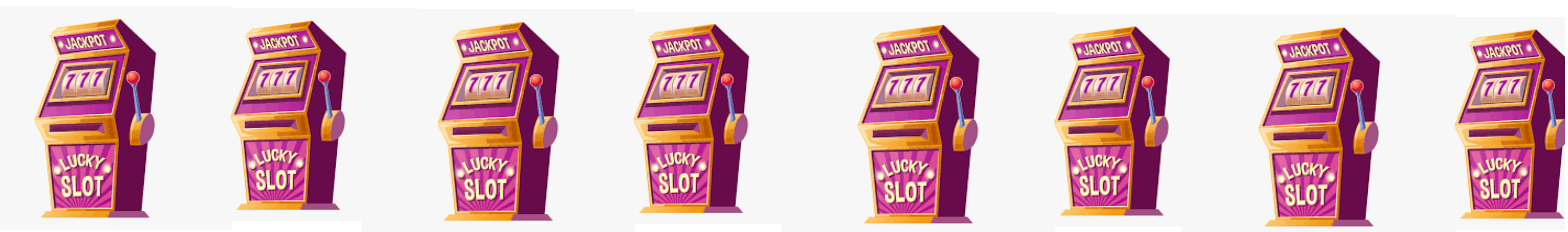
Importance Sampling

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

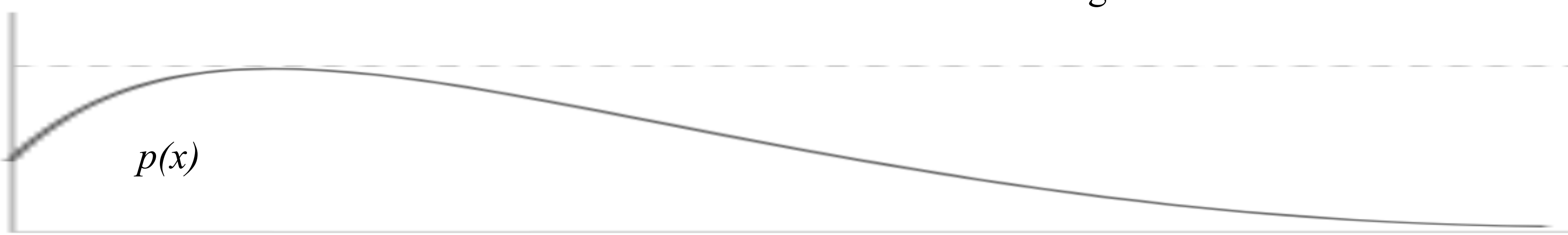
x^i is sampled from $p(x)$

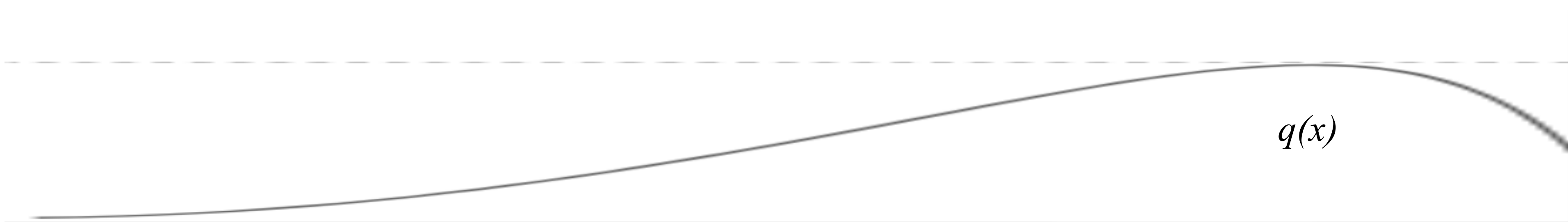
We only have x^i sampled from $q(x)$

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



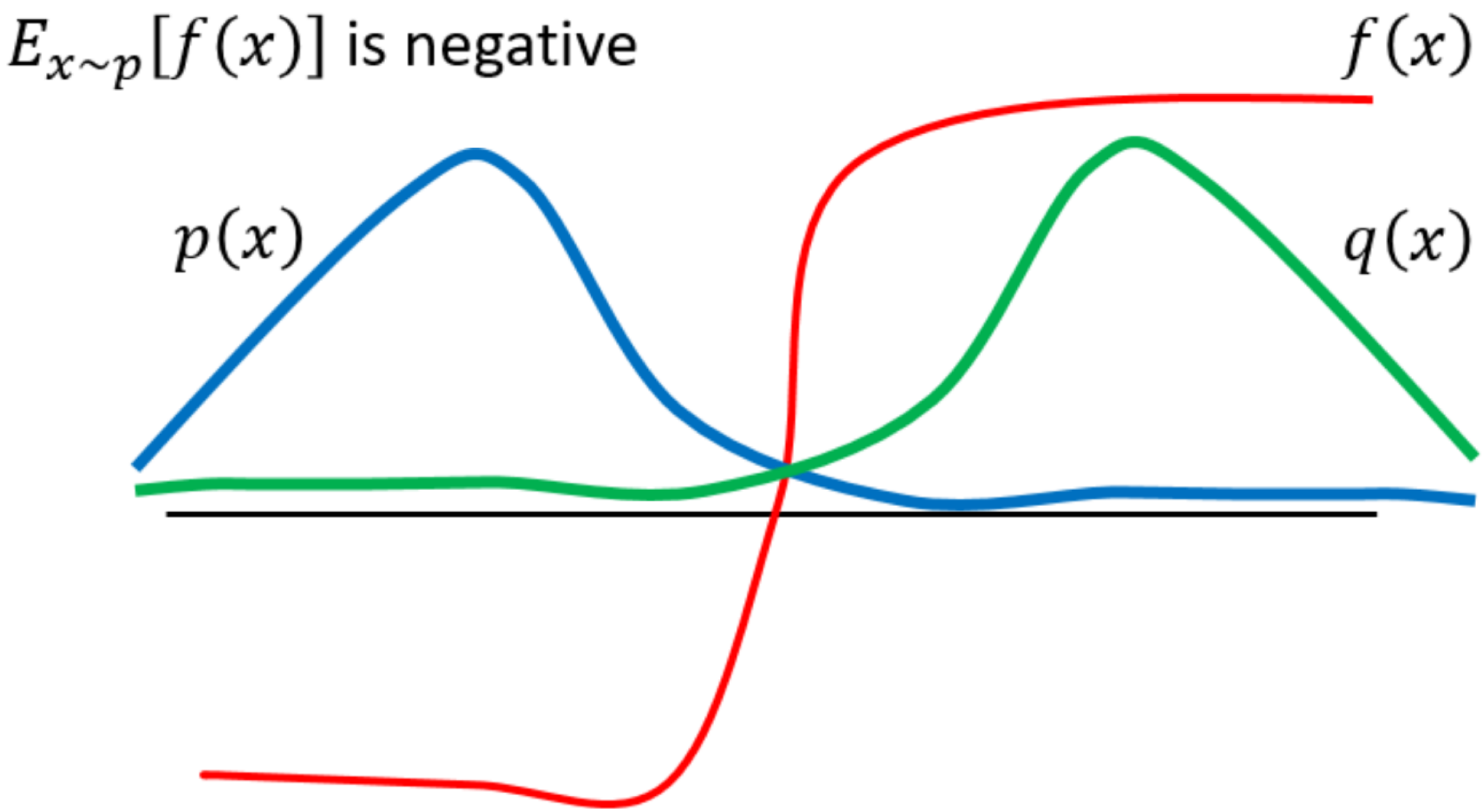
Low reward rates --→ high reward rates





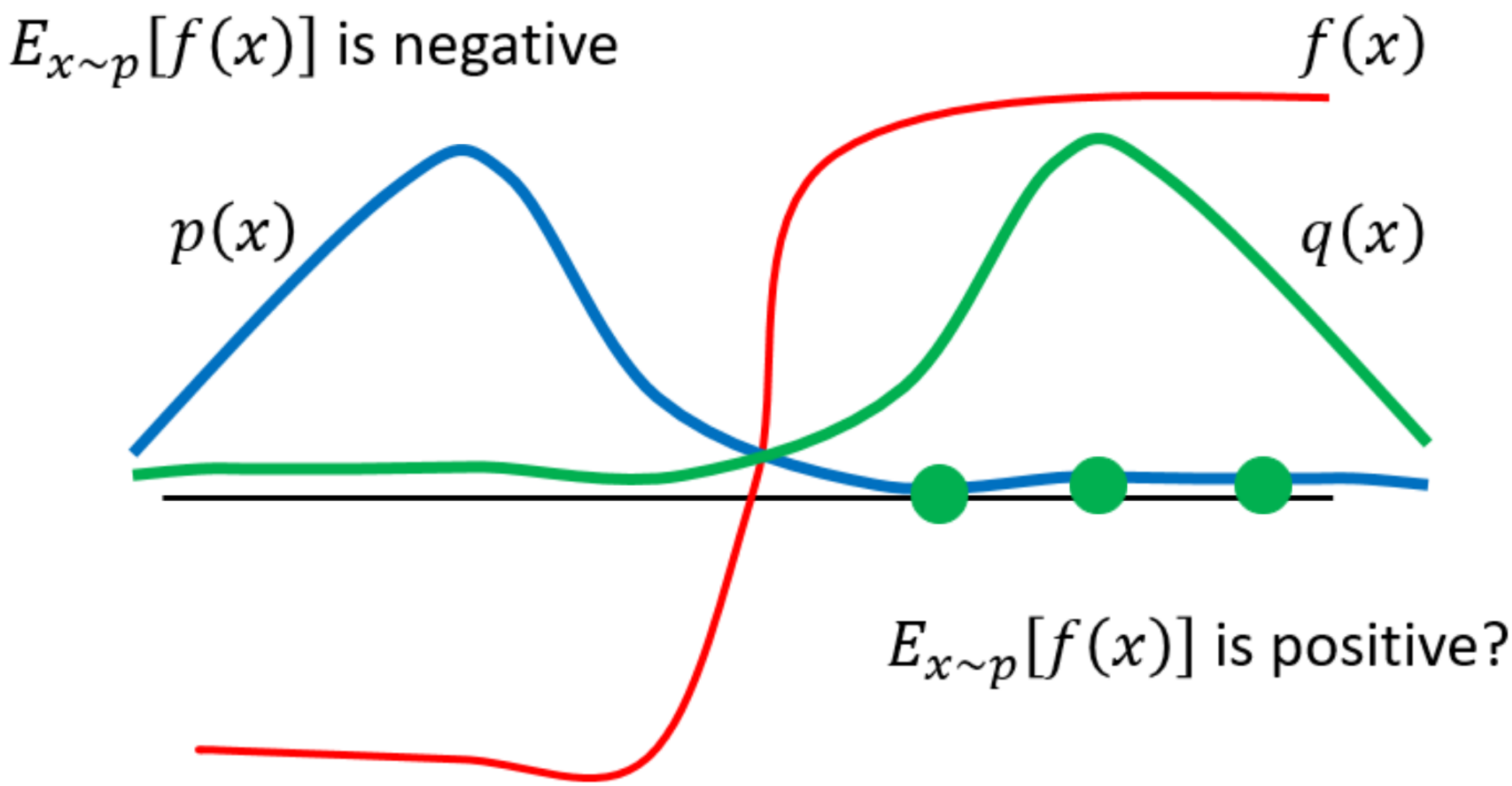
Issue of Importance Sampling

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



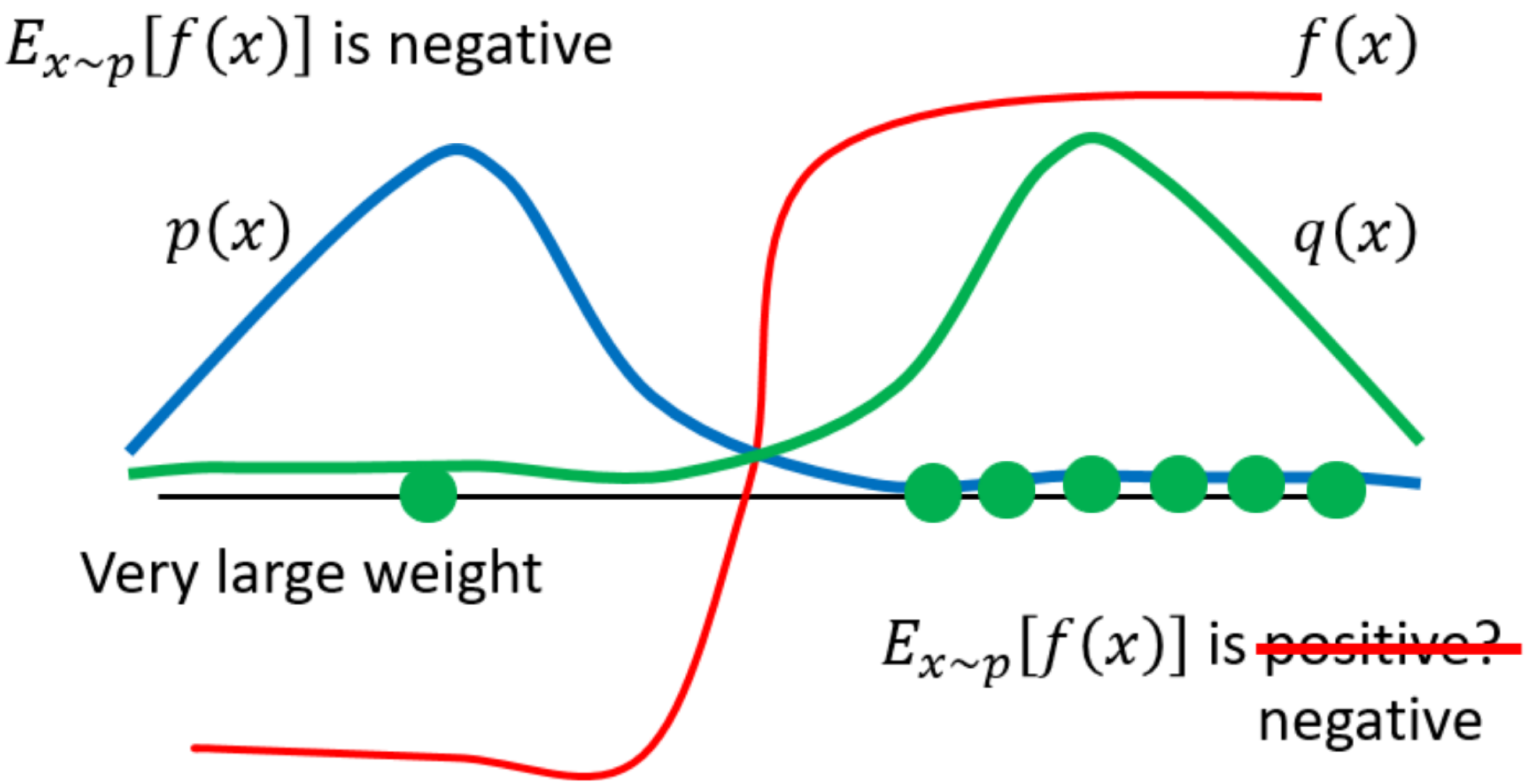
Issue of Importance Sampling

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



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

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

Gradient for update

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

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

On-policy → Off-policy

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

Gradient for update

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

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

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

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

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

This term is from sampled data.

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

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

On-policy → Off-policy

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

Gradient for update

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

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

This term is from sampled data.

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

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

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)

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

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

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

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

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

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

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

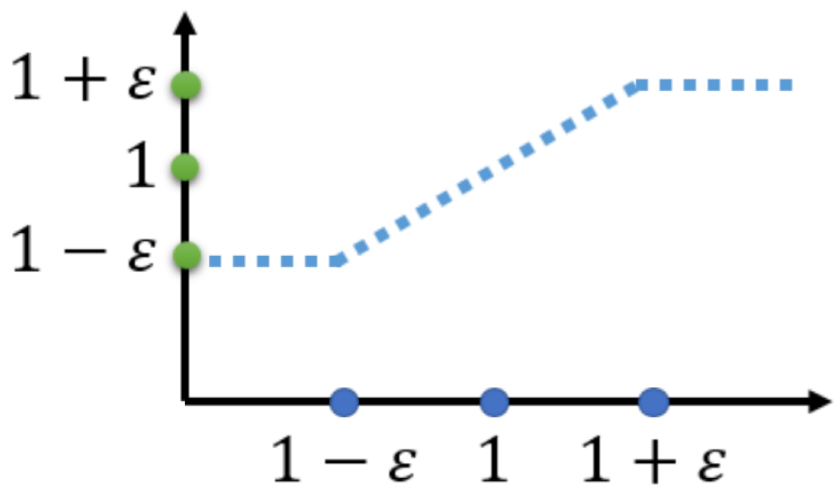
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

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

PPO2 algorithm

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

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



$$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)}$$

PPO algorithm

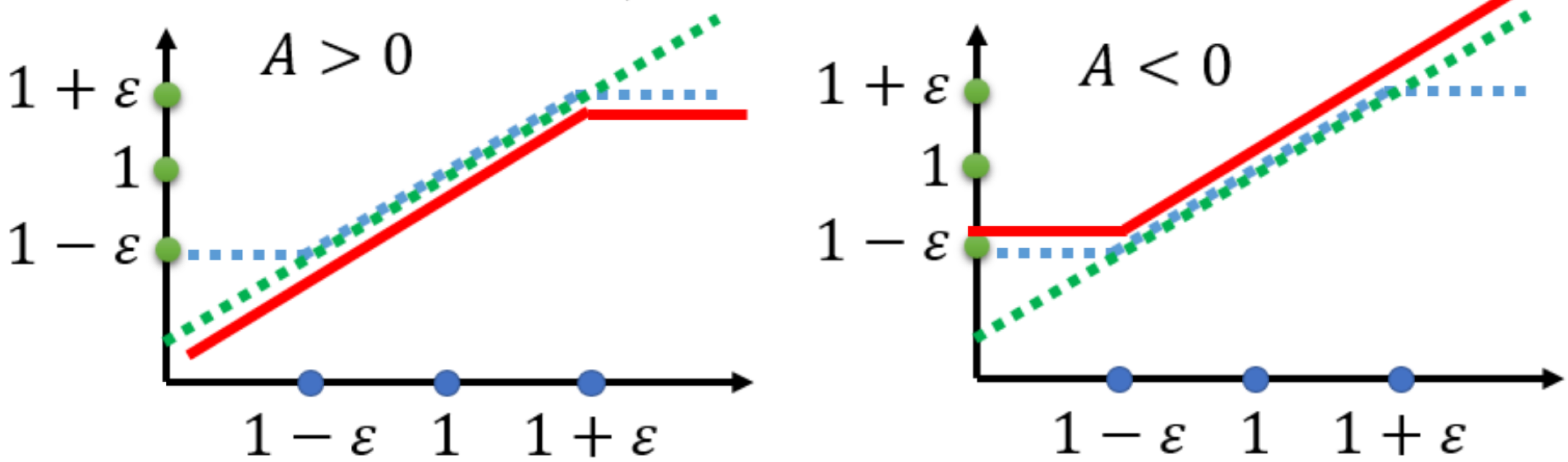
$$J_{PPO}^{\theta^k}(\theta) = J^{\theta^k}(\theta) - \beta KL(\theta, \theta^k)$$

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

PPO2 algorithm

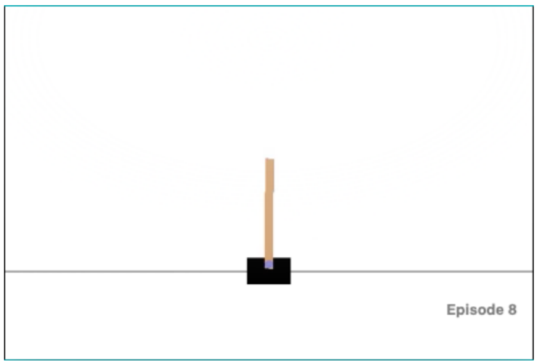
$$J_{PPO2}^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t),\right.$$

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

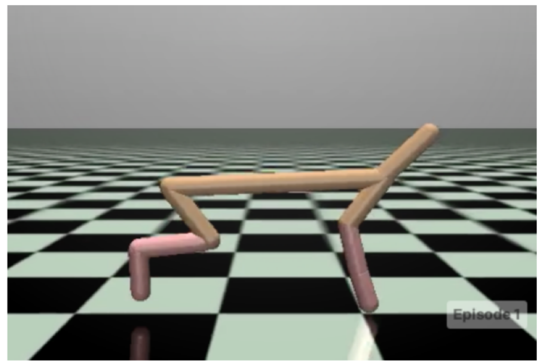


Experimental Results (with MuJoCo Tasks)

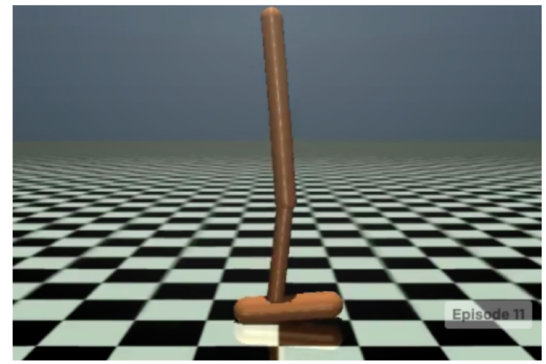
https://arxiv.org/abs/1707.06347



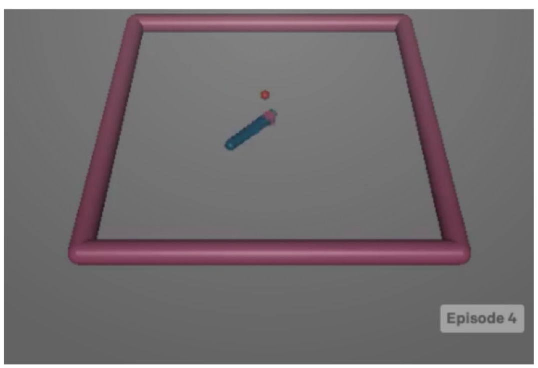
(a) CartPole-v0



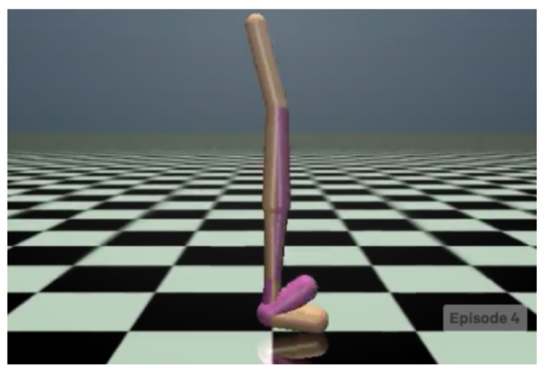
(b) HalfCheetah-v2



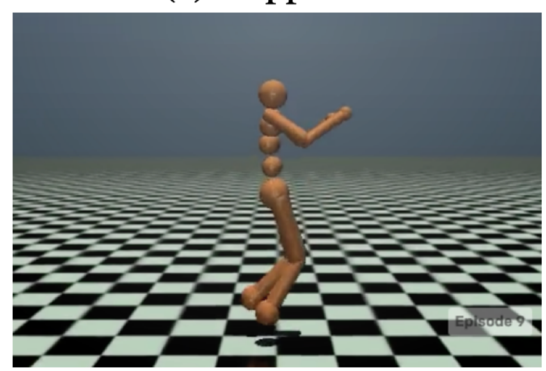
(c) Hopper-v2



(d) Reacher-v2



(e) Walker-v2



(f) Humanoid-v2

https://arxiv.org/abs/1707.06347

Experimental Results

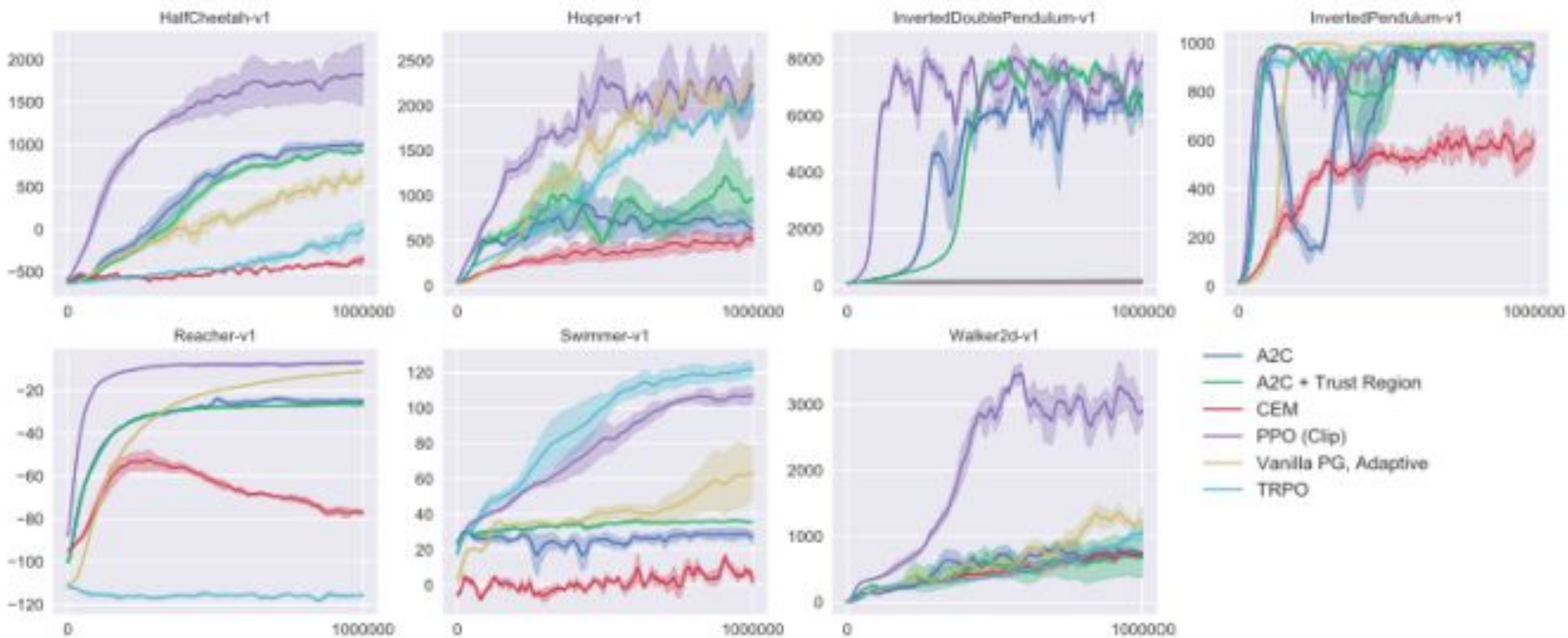


Figure 3: Comparison of several algorithms on several MuJoCo environments, training for one million timesteps.

	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)	**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?

What is next?

- Other deep reinforcement learning approaches

  - (Asynchronous) Advantage Actor Critic:
    - A2C
    - A3C

  - Deep Inverse reinforcement learning
    - Entropy based IRL
    - GAN (Generative adversarial networks)
    - GAIL (Generative adversarial imitation learning)

Backup Slide #1

$$
\begin{aligned}
\nabla_\theta \log P(\tau^{(i);\theta}) &= \nabla_\theta \log \left[\underbrace{\mu(s_0)}_{\text{Initial state distrib.}} \prod_{t=0}^{T-1} \underbrace{\pi_\theta(a_t|s_t)}_{\text{policy}} \underbrace{P(s_{t+1}|s_t, a_t)}_{\text{dynamics model}} \right] \\
&= \nabla_\theta \left[\log \mu(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t|s_t) + \log P(s_{t+1}|s_t, a_t) \right] \\
&= \sum_{t=0}^{T-1} \underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{no dynamics model required!}}
\end{aligned}
$$

Policy Gradient: Use Temporal Structure

Backup Slide #2

- Previously:

$$\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau \left[\left(\sum_{t=0}^{T-1} r_t \right) \left(\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \right) \right]$$

- We can repeat the same argument to derive the gradient estimator for a single reward term $r_{t'}$.

$$\nabla_\theta \mathbb{E}[r_{t'}] = \mathbb{E} \left[r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

- Summing this formula over t, we obtain

$$V(\theta) = \nabla_\theta \mathbb{E}[R] = \mathbb{E} \left[\sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

$$= \mathbb{E} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t, s_t) \sum_{t'=t}^{T-1} r_{t'} \right]$$

Baseline $b(s)$ Does Not Introduce Bias–Derivation

Backup Slide #3

$$
\begin{aligned}
&\mathbb{E}_\tau[\nabla_\theta \log \pi(a_t|s_t,\theta)b(s_t)] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t,\theta)b(s_t)]\right] \text{ (break up expectation)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\mathbb{E}_{s_{(t+1):T},a_{t:(T-1)}}[\nabla_\theta \log \pi(a_t|s_t,\theta)]\right] \text{ (pull baseline term out)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\mathbb{E}_{a_t}[\nabla_\theta \log \pi(a_t|s_t,\theta)]] \text{ (remove irrelevant variables)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \pi_\theta(a_t|s_t)\frac{\nabla_\theta \pi(a_t|s_t,\theta)}{\pi_\theta(a_t|s_t)}\right] \text{ (likelihood ratio)} \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\sum_a \nabla_\theta \pi(a_t|s_t,\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}\left[b(s_t)\nabla_\theta \sum_a \pi(a_t|s_t,\theta)\right] \\
&= \mathbb{E}_{s_{0:t},a_{0:(t-1)}}[b(s_t)\nabla_\theta 1] \\
&= \mathbb{E}_{s_0:t,a_0:(t-1)}[b(s_t)\cdot 0] = 0
\end{aligned}
$$