535514: Reinforcement Learning

Lecture 13 — Deterministic Policy Gradient

Ping-Chun Hsieh

April 8, 2024

Announcements

- Team Implementation project:
  - Please finalize your topic by 4/10 (Wednesday)
  - Please fill out the Google form: https://forms.gle/S2bi787XzLfUE9kb7
  - You could still edit the Google form if you submitted it earlier
  - Paper list: https://hackmd.io/@pinghsieh/r1uBfGel0
- Theory project:
  - Please select a paper by 4/17 (Wednesday)
  - Please fill out the Google form: https://forms.gle/KDNr8Hw1G3FJLgqeA
  - Paper list: https://hackmd.io/@pinghsieh/HJhuxeexA

Announcements

- HW2 has been posted on E3
  - Due on 4/19 (next Friday), 9pm Taiwan time

- Today's lecture: 10:10am-**11:00am**

- To make up for the lecture:
  - 10min extension on 4/15 (Mon.), 4/22 (Mon.), 4/29 (Mon.), 5/6 (Mon.), 5/13 (Mon.)

A Seminar Today!

NYCU CS SEMINAR

Inverse Constraint Learning and Risk Averse Reinforcement Learning for Safe AI

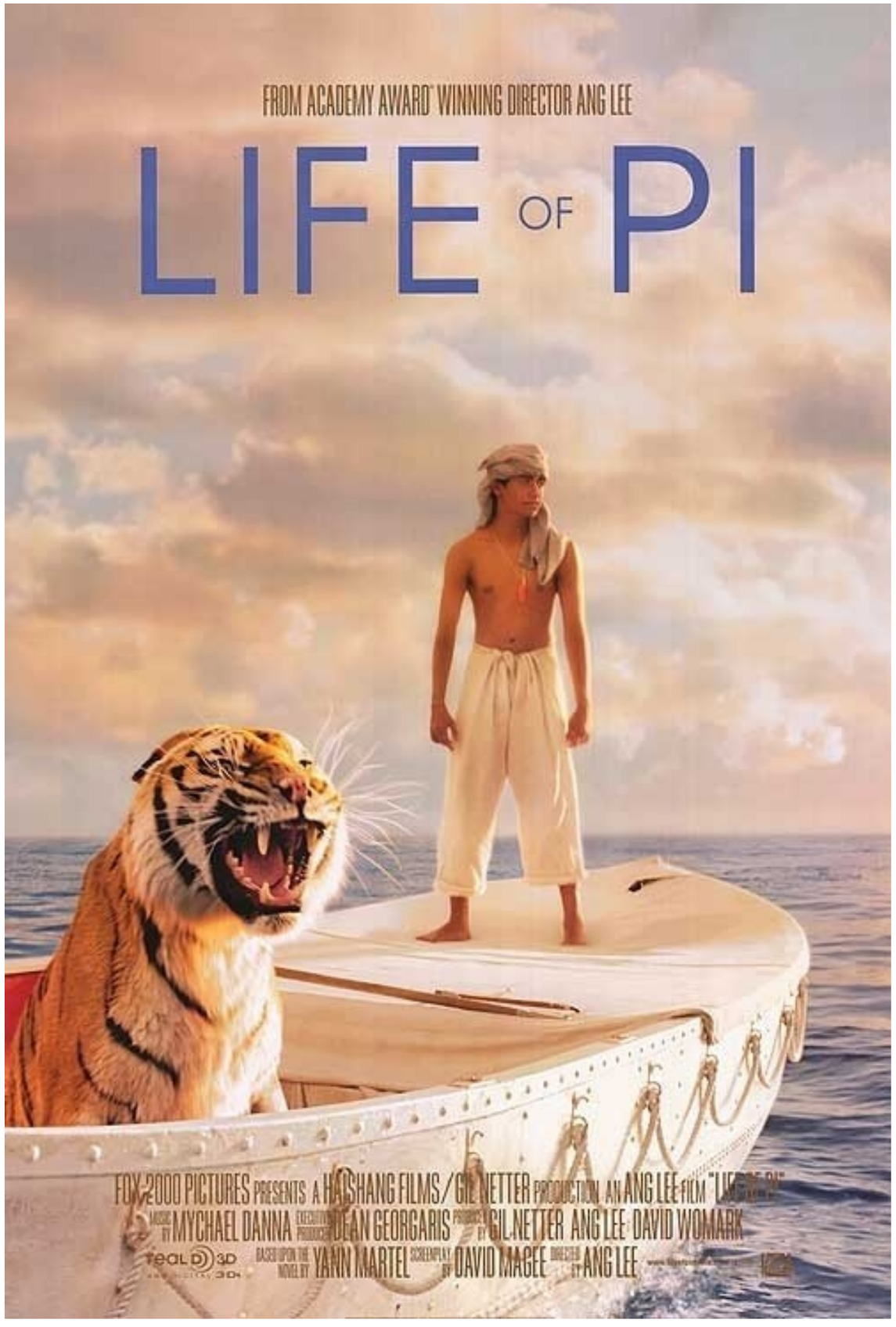
Time: 2024/04/08 (Monday) 13:30-15:00
Location: EC329 (工程三館329室)
Speaker: Prof. Pascal Poupart
(University of Waterloo)

【講題大綱】

In many applications of reinforcement learning (RL) and control, policies need to satisfy constraints to ensure feasibility, safety or thresholds about key performance indicators. However, some constraints may be difficult to specify. For instance, in autonomous driving, it is relatively easy to specify a reward function to reach a destination, but implicit constraints followed by expert human drivers to ensure a safe, smooth and comfortable ride are much more difficult to specify. I will present some techniques to learn soft constraints from expert trajectories in autonomous driving and robotics. I will also present an alternative to variance based on Gini deviation for risk-averse reinforcement learning.

Deterministic Policy Gradient

Silver et al., “Deterministic Policy Gradient Algorithms”, ICML 2014



- Which version of the story do you believe?

Stochastic PG vs Deterministic PG

- So far, we have focused on "stochastic" policy gradient

- Question: How to extend policy gradient framework to parametrized deterministic policies?

- Question: What's the benefit of learning deterministic policies?

Stochastic PG: $$\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a\,|\,s)\right]$$

integration over both state and action spaces

$\Rightarrow$ *require more samples to estimate the gradient*

Why is Deterministic PG Challenging?

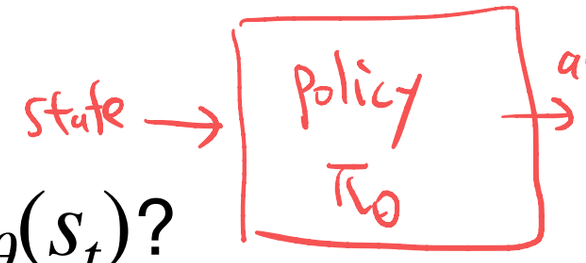
- Recall: Stochastic policy gradient (P1), $a_t \sim \pi_\theta(\cdot \mid s_t)$

$$\nabla_\theta V^{\pi_\theta}(\mu) = \sum_\tau R(\tau) \nabla_\theta P_\mu^{\pi_\theta}(\tau)$$

$$= \sum_\tau R(\tau) \left(P_\mu^{\pi_\theta}(\tau) \cdot \nabla_\theta \log(\mu(s_0)\pi_\theta(a_0 \mid s_0) P(s_1 \mid s_0, a_0)\pi_\theta(a_1 \mid s_1)\cdots) \right)$$

$$= \sum_\tau R(\tau) \left(P_\mu^{\pi_\theta}(\tau) \cdot \sum_{t=0}^{\infty} \nabla_\theta \log \pi(a_t \mid s_t) \right)$$

$\nabla_\theta P_\mu^{\pi_\theta}(\tau) = P_\mu^{\pi_\theta}(\tau) \cdot \nabla_\theta \log P_\mu^{\pi_\theta}(\tau)$



- Question: How about deterministic policies $a_t = \pi_\theta(s_t)$?

$$\nabla_\theta V^{\pi_\theta}(\mu) = \sum_\tau R(\tau) \nabla_\theta P_\mu^{\pi_\theta}(\tau)$$

$$= \sum_\tau R(\tau) \left(P_\mu^{\pi_\theta}(\tau) \cdot \nabla_\theta \log(\mu(s_0) P(s_1 \mid s_0, \pi_\theta(s_0)) P(s_2 \mid s_1, \pi_\theta(s_1))\cdots) \right)$$

$$= \sum_\tau R(\tau) \left(P_\mu^{\pi_\theta}(\tau) \cdot \sum_{t=0}^{\infty} \nabla_\theta \log P\left(s_{t+1} \mid s_t, \pi_\theta(s_t)\right) \right)$$

How to calculate this gradient?

Challenge: It seems we need to know the model P to find the gradient

Despite this, the deterministic PG indeed exists and enjoys a simple form

Deterministic Policy Gradient (DPG)

- Consider continuous actions and deterministic policy: $a = \pi_\theta(s)$
- Assumptions: $\nabla_a Q(s,a), \nabla_a P(s'|s,a), \nabla_\theta \pi_\theta(s), \nabla_a R(s,a)$ exist
- **Deterministic Policy Gradient Theorem**:

$$\nabla_\theta V_\mu^{\pi_\theta} = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\left[\nabla_\theta \pi_\theta(s)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\right]$$

$V_\mu^{\pi_\theta} = \mathbb{E}_{s\sim\mu}[V^{\pi_\theta}(s)]$

Suppose $a \in \mathbb{R}^1$: $Q^{\pi_\theta}(s,a)$, $\pi_\theta(s)$, a

State visitation distribution

Tells us how to change the θ

Determines the "action direction"

- Comparison: Stochastic policy gradient

(P3) Q-value and discounted state visitation:

$$\nabla_\theta V_\mu^{\pi_\theta} = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right]$$

Connecting Stochastic PG and Deterministic PG

Deterministic PG: $\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\left[\nabla_\theta \pi_\theta(s)\, \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\right]$

Stochastic PG: $\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi_\theta}(s,a)\, \nabla_\theta \log \pi_\theta(a\,|\,s)\right]$

"perturbed" or "noisy" version of $\pi_\theta(s)$

- Question: Any connection between the two PGs?
- Consider a stochastic policy: $\nu_\sigma(\pi_\theta, a)$ with $\nu_0(\pi_\theta(s), a) \equiv \pi_\theta(s)$
- **Deterministic PG is the Limiting Case of Stochastic PG**:

  Under mild technical conditions on ν_σ (differentiability, bounded gradient, translation invariance), we have

$$\lim_{\sigma\downarrow 0} \nabla_\theta V^{\nu_\sigma(\pi_\theta)}(\mu) = \nabla_\theta V^{\pi_\theta}(\mu)$$

Silver et al., "Deterministic Policy Gradient Algorithms", ICML 2014

Let’s apply deterministic PG to design algorithms

(On-Policy) Deterministic Actor-Critic Algorithm

Deterministic PG: $$\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\left[\nabla_\theta \pi_\theta(s)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\right]$$

- Deterministic Actor-Critic (with Value Function Approximation):
  - Critic: estimate $Q_w \approx Q^{\pi_\theta}$ by TD(0) bootstrapping
  - Actor: updates policy parameters θ by deterministic policy gradient

Step 1: Initialize θ_0, w_0 and step sizes α_θ, α_w

Step 2: Sample a trajectory $\tau = (s_0, a_0, r_1, \cdots) \sim P_\mu^{\pi_\theta}$

For each step of the current trajectory $t = 0,1,2,\cdots$

$$\Delta w_k \leftarrow \Delta w_k + \alpha_w\left(r_t + \gamma Q_{w_k}(s_{t+1}, a_{t+1}) - Q_{w_k}(s_t, a_t)\right)\nabla_w Q_w(s_t, a_t)|_{w=w_k}$$

$$\Delta\theta_k \leftarrow \Delta\theta_k + \alpha_\theta\gamma^t\left(\nabla_\theta\pi_\theta(s_t)\nabla_a Q_w(s_t, a)|_{a=\pi_\theta(s_t)}\right)$$

$$= \nabla_\theta Q_w(s_t, \pi_\theta(s_t))|_{\theta=\theta_k}$$

$$\theta_{k+1} \leftarrow \theta_k + \Delta\theta_k, w_{k+1} \leftarrow w_k + \Delta w_k$$

A Quick Remark on DPG Expression

- In Deterministic Actor-Critic:

$$\Delta\theta_k \leftarrow \Delta\theta_k + \alpha_\theta \gamma^t \Big(\underbrace{\nabla_\theta \pi_\theta(s_t) \nabla_a Q_w(s_t, a)|_{a=\pi_\theta(s_t)}}_{= \nabla_\theta Q_w(s_t, \pi_\theta(s_t))|_{\theta=\theta_k}} \Big)$$

- In the original DPG expression:

$$\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \Big[\underbrace{\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s)}}_{= \nabla_\theta Q^{\pi_\theta}(s, \pi_\theta(s))|_{\theta=\theta_k} \text{ ??}} \Big]$$

- Question: Any issue with deterministic policies?
  - Get stuck at local max (as $\nabla_a Q^{\pi_\theta}(s,a)$ can be zero).
  - Exploration issue

- Question: Is it possible to learn π but act under another policy β?

π: deterministic

β: stochastic exploratory

- Question: Any issue with deterministic policies?

Insufficient exploration

- Question: Is it possible to learn π but act under another policy β?

Off-policy learning!

Off-Policy Learning with Deterministic Policy Gradients

On-Policy vs Off-Policy

- On-policy:

  Learned policy = Policy used to interact with the environment

- Off-policy:

  Learned policy ≠ Policy used to interact with the environment

  Called “behavior policy”

Kazami Hayato
(learning agent)

Asurada
(policy for interaction)

Off-Policy Learning

Off-policy learning

1. Learn a target policy $\pi_\theta(a \mid s)$ and compute $V^{\pi_\theta}(s)$ or $Q^{\pi_\theta}(s, a)$
2. In the meantime, follow a behavior policy $\beta(a \mid s)$

$$\{s_0, a_0, r_1, s_1, a_1, r_2, \cdots s_{T-1}, a_{T-1}, r_T\} \sim \beta$$

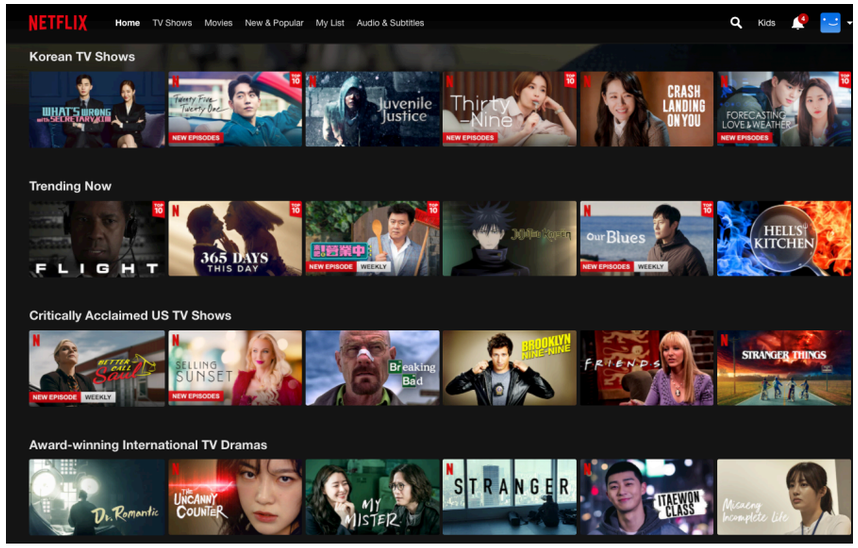
- Why is off-policy learning useful?
  1. Learn from observing humans or other agents
  2. Reuse experience generated from old policies $\pi_1, \pi_2 \cdots, \pi_{k-1}$
  3. Learn about optimal policy while following an exploratory policy
  4. Learn about multiple policies while following one policy

(Slide Credit: David Silver)

Off-Policy Learning is Essential in Many “Real-World” Problems

- Recommender Systems

➡ Deploy a safe policy for collecting user data without losing user’s interest



➡ Learn a better policy from these data

- Robot Control

➡ Deploy a safe policy for collecting robot data without hurting the machine

➡ Learn a good policy from these data

What are the behavior policies in the above applications?

Deterministic PG in Off-Policy Learning

(On-policy) DPG: $\nabla_\theta V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\mu^{\pi_\theta}}\left[\nabla_\theta \pi_\theta(s)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\right]$

- Question: Does the deterministic PG remain the same in off-policy learning?

  Nope! (state visitation distribution shall change)

- Consider a new objective for off-policy learning: (Why reasonable?)

$$J_\beta(\pi_\theta) := \sum_s d_\mu^\beta(s) V^{\pi_\theta}(s)$$

- "Off-policy" deterministic PG: Let's use $\nabla_\theta J_\beta(\pi_\theta)$!

Off-Policy Deterministic PG

- **Off-Policy Deterministic Policy Gradient**:

$$\nabla_\theta J_\beta(\pi_\theta) \approx \mathbb{E}_{s\sim d_\mu^\beta}\Big[\nabla_\theta \pi_\theta(s)\, \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\Big]$$

- Question: Is the above easy to operate with?

- Derivation:

$$\begin{aligned}
\nabla_\theta J_\beta(\pi_\theta) &= \nabla_\theta\Big(\sum_s d_\mu^\beta(s) V^{\pi_\theta}(s)\Big) \\
&= \nabla_\theta\Big(\sum_s d_\mu^\beta(s) Q^{\pi_\theta}(s,\pi_\theta(s))\Big) \\
&\approx \sum_s d_\mu^\beta(s)\Big(\nabla_\theta \pi_\theta(s)\, \nabla_a Q^{\pi_\theta}(s,a)|_{a=\mu_\theta(s)}\Big)
\end{aligned}$$

The DPG paper (Silver, ICML 2014) dropped a term $\nabla_\theta Q^{\pi_\theta}(s,a)$

Why is $\nabla_\theta Q^{\pi_\theta}(s,a)$ Difficult to Evaluate?

- Recall from the expression of deterministic PG:

$$\nabla_\theta V^{\pi_\theta}(s) = \frac{1}{1-\gamma_\infty}\mathbb{E}_{s'\sim d_\mu^{\pi_\theta}}\Big[\nabla_\theta \pi_\theta(s')\,\nabla_a Q^{\pi_\theta}(s',a)|_{a=\pi_\theta(s')}\Big]$$

$$= \sum_{s'}\sum_{t=0}^{\infty}\gamma^t P(s\to s',t,\pi_\theta)\,\nabla_\theta\pi_\theta(s')\,\nabla_a Q^{\pi_\theta}(s',a)\Big|_{a=\pi_\theta(s')}$$

Accordingly, we have

$$\nabla_\theta Q^{\pi_\theta}(s,a) = \nabla_\theta\Big(R(s,a)+\gamma\sum_{s'}P(s'|s,a)V^{\pi_\theta}(s')\Big)$$

$$= \gamma\sum_{s'}P(s'|s,a)\,\nabla_\theta V^{\pi_\theta}(s')$$

$$= \gamma\sum_{s'}P(s'|s,a)\sum_{s''}\sum_{t=0}^{\infty}\gamma^t P(s'\to s'',t,\pi_\theta)\,\nabla_\theta\pi_\theta(s'')\,\nabla_a Q^{\pi_\theta}(s'',a)\Big|_{a=\pi_\theta(s'')}$$

hard to evaluate in off-policy learning (why?)

Off-Policy Deterministic Actor-Critic (OPDAC) Algorithm

- Off-Policy Deterministic Actor-Critic (OPDAC):
  - Critic: estimate $Q_w \approx Q^{\pi_\theta}$ by TD bootstrapping
  - Actor: updates policy parameters θ by off-policy deterministic PG

Step 1: Initialize θ_0, w_0 and step sizes α_θ, α_w

Step 2: Sample a trajectory $\tau = (s_0, a_0, r_1, \cdots) \sim P_\mu^\beta$

For each step of the current trajectory $t = 0,1,2,\cdots$

$$\Delta w_k \leftarrow \Delta w_k + \alpha_w \left(r_t + \gamma Q_{w_k}(s_{t+1}, \pi_\theta(s_{t+1})) - Q_{w_k}(s_t, a_t) \right) \nabla_w Q_w(s_t, a_t)|_{w=w_k}$$

$$\Delta \theta_k \leftarrow \Delta \theta_k + \alpha_\theta \gamma^t \left(\nabla_\theta \pi_\theta(s_t) \nabla_a Q_{w_k}(s_t, a)|_{a=\pi_\theta(s_t)} \right)$$

$$\theta_{k+1} \leftarrow \theta_k + \Delta\theta_k, w_{k+1} \leftarrow w_k + \Delta w_k \qquad = \nabla_\theta Q_{w_k}(s_t, \pi_\theta(s_t))|_{\theta=\theta_k}$$

- Question: Can you identify differences between OPDAC and DAC?

Deep Deterministic Policy Gradient (DDPG)

(= OPDAC with Deep Neural Nets)

Lillicrap et al., “Continuous control with deep reinforcement learning”, ICLR 2016

What is DDPG?

- DDPG: Combine OPDAC with NN nonlinear VFA
  - Off-policy: Exploration
  - Nonlinear VFA: Convergence issue

- To tackle the above issues, DDPG applies several techniques:

(T1) Experience replay (for data-efficient off-policy learning)

(T2) Ornstein-Uhlenbeck process for exploration (optional)

(T3) Target networks

(T1) Experience Replay

- Main idea:

  1. Store the previous experiences (s, a, s', r) into a buffer

  2. Sample a mini-batch from the buffer at each step

(similar to mini-batch SGD in supervised learning)

- Purposes:

1. Better estimate of DPG: Break correlations between successive steps in a trajectory (“more stable learning”, as stated in many papers)

2. Better data efficiency: Fewer interactions with environment needed for convergence

(T2) Ornstein-Uhlenbeck Process for Exploration

- Issue with Gaussian noise exploration $a_t = \pi_\theta(s_t) + N(0,\sigma^2)$?

- Ornstein-Uhlenbeck (OU) process: Similar to Gaussian policies, but with temporal correlation

Brownian motion

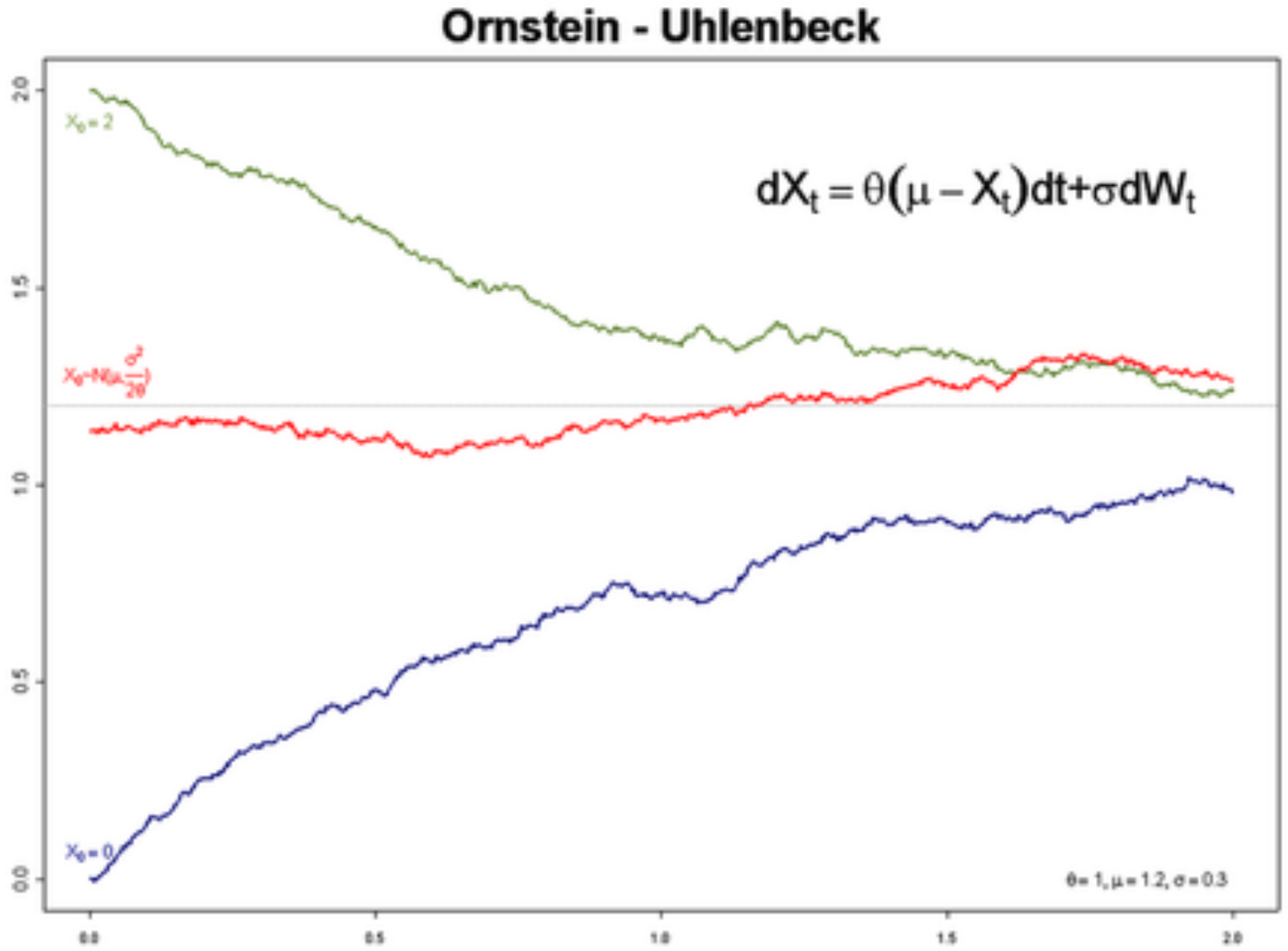
$$dx_t = \theta(\mu - x_t)dt + \sigma \cdot dW_t$$

- Discrete-time approximation of OU:

$$X_{t+1} - X_t = \theta(\mu - X_t)\Delta t + \sigma \cdot \Delta W_t$$

i.i.d. normal random variables $\sim \mathcal{N}(0,\Delta_t)$

Example of OU Process



(Same OU process with 3 different initial conditions)

How about a sequence of i.i.d. Gaussian random variables?

(Figure Source: Wikipedia)

(T3) Target Networks

- Idea: Use separate *target networks* ($\bar{\pi}_\theta$ for actor, $\bar{Q}_w$ for critic) that are updated only periodically
- For DDPG, the critic update with target networks

$$\Delta w_k \leftarrow \Delta w_k + \alpha_w \Big(r_t + \gamma \bar{Q}_{w_k}(s_{t+1}, \bar{\pi}_\theta(s_{t+1})) - Q_{w_k}(s_t, a_t) \Big) \nabla_w Q_w(s_t, a_t)|_{w=w_k}$$

target update

- Similar to value iteration:

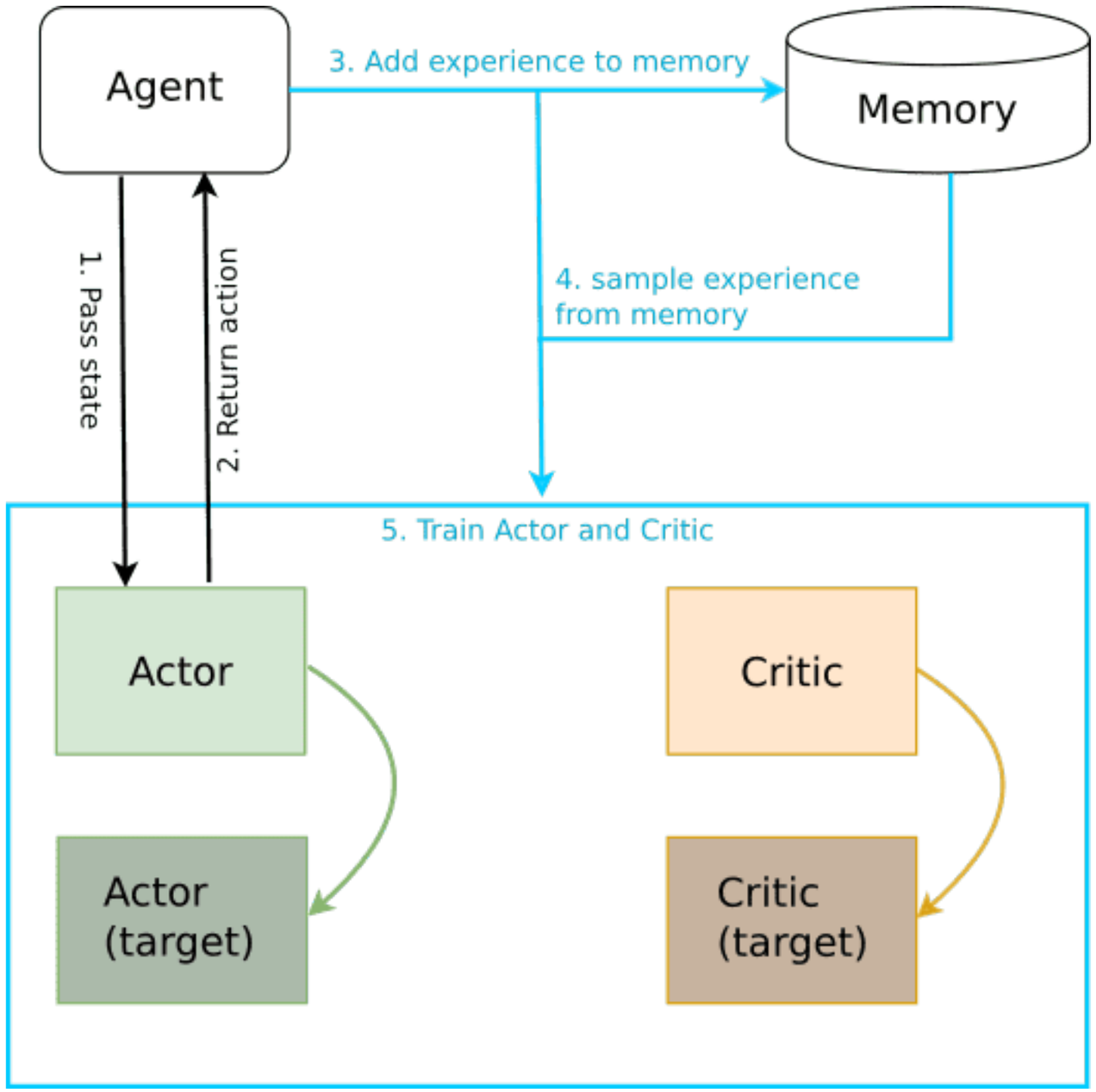
$$V(s) \leftarrow \max_a R(s,a) + \gamma \sum_{s'} P(s' \mid s, a) \bar{V}(s)$$

update target

- Purpose: Mitigate divergence

(Slide Credit: Pascal Poupart)

DDPG Architecture



(Figure Credit: https://www.katnoria.com/ddpg-reacher/)

Pseudo Code of DDPG Algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$

2 evaluation networks and 2 target networks

Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1

action drawn from a deterministic policy with exploration

  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R

experience replay

    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

Update actor and critic

This can be viewed as the gradient of Q w.r.t. θ

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

Update target networks (small τ for stability)

  end for
end for

A Few Hidden Issues of DDPG

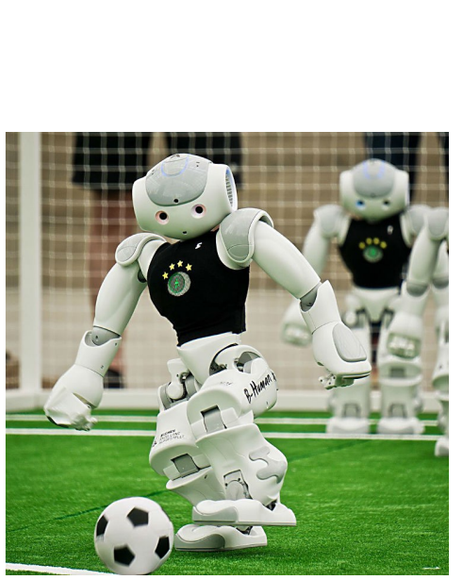
(A1) Two interpretations of DDPG

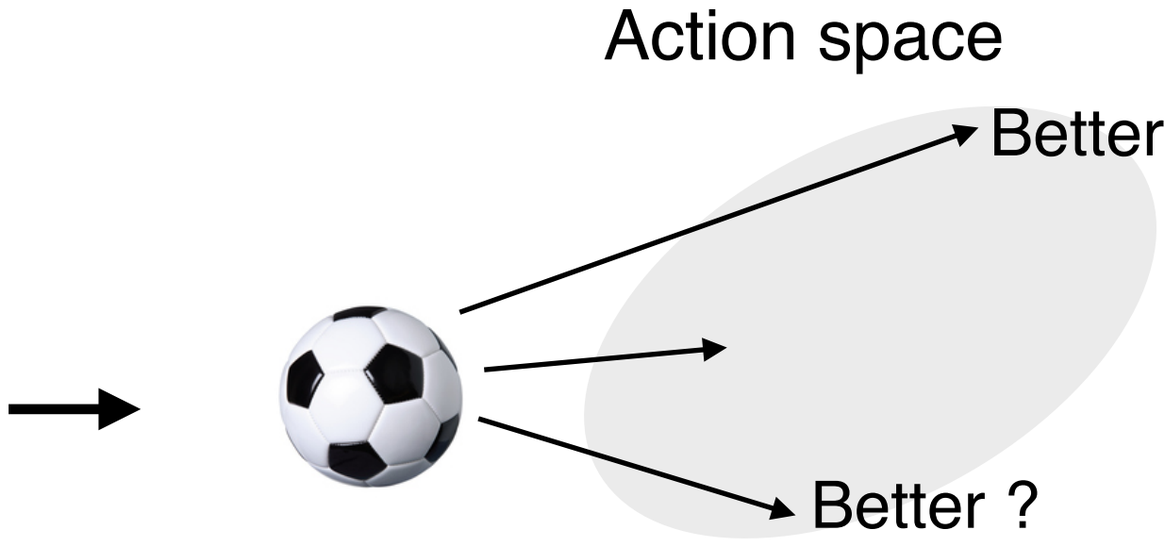
(A2) Action constraints

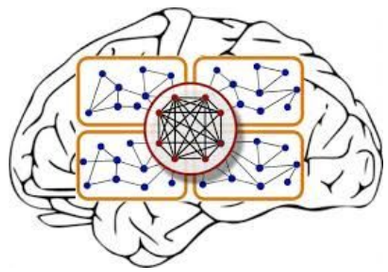
(A3) Overestimation of Q functions

A Motivating Example of “Search & Mimic”

(A Robocup example)







Current state

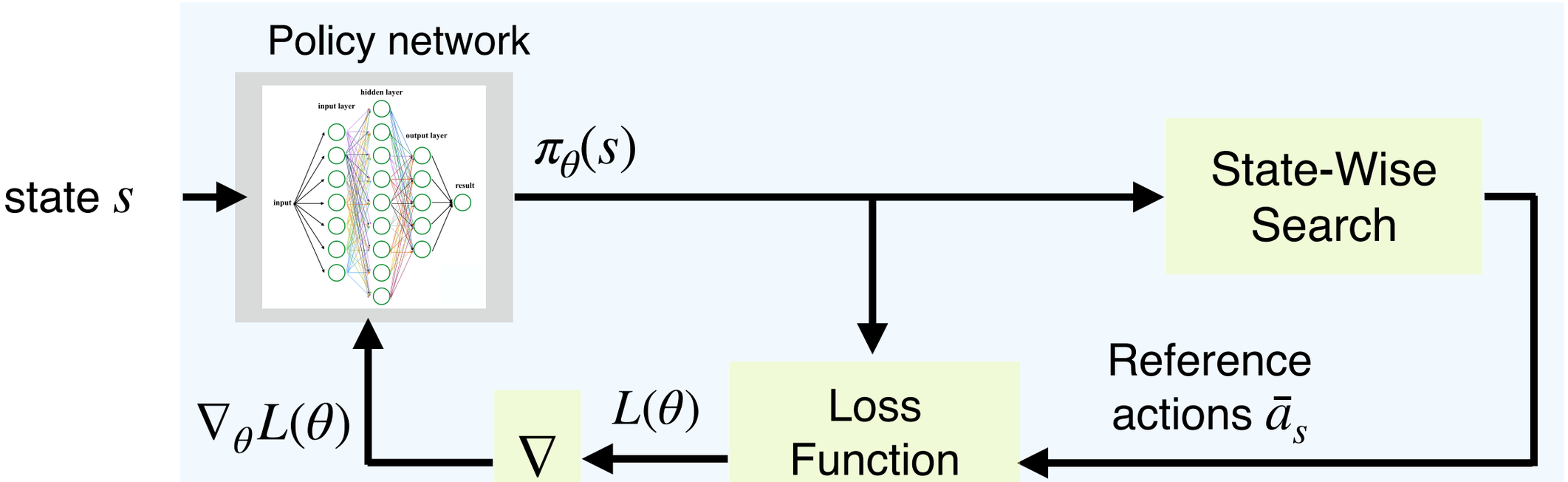
Search for a better reference action (higher Q value)

Mimic (update the policy towards the reference action)

RL = Repeatedly “search & mimic” in different scenarios

(A1) Alternative Interpretation of DDPG

Let's formally write down "Search & Mimic" approach:



$$\bar{a}_s = \pi_{\bar{\theta}}(s) + \eta \nabla_a Q_w(s,a)$$

$$L(\theta) = \frac{1}{|B|} \sum_{s \in B} \left(\pi_\theta(s) - \bar{a}_s \right)^2$$

$$\nabla_\theta L(\theta) = \frac{1}{|B|} \sum_{s \in B} \nabla_\theta \left(\pi_\theta(s) - \bar{a}_s \right)^2 = \frac{1}{|B|} \sum_{s \in B} \nabla_\theta \left(\pi_\theta(s) - \left(\pi_{\bar{\theta}}(s) + \eta \nabla_a Q_w(s,a) \right) \right)^2$$

One Surprising Fact: “Search & Mimic” and DDPG Are Equivalent!

Theorem: $\Delta\theta_{DDPG}$ & $\Delta\theta_{S\&M}$ are parallel

---

$$\Delta\theta_{DDPG} \propto \nabla_\theta J_\mu(\pi_\theta) = \frac{1}{|B|}\sum_{s\in B} \nabla_a Q_w(s,a)\,\nabla_\theta \pi_\theta(s) \qquad \text{(By DPG theorem)}$$

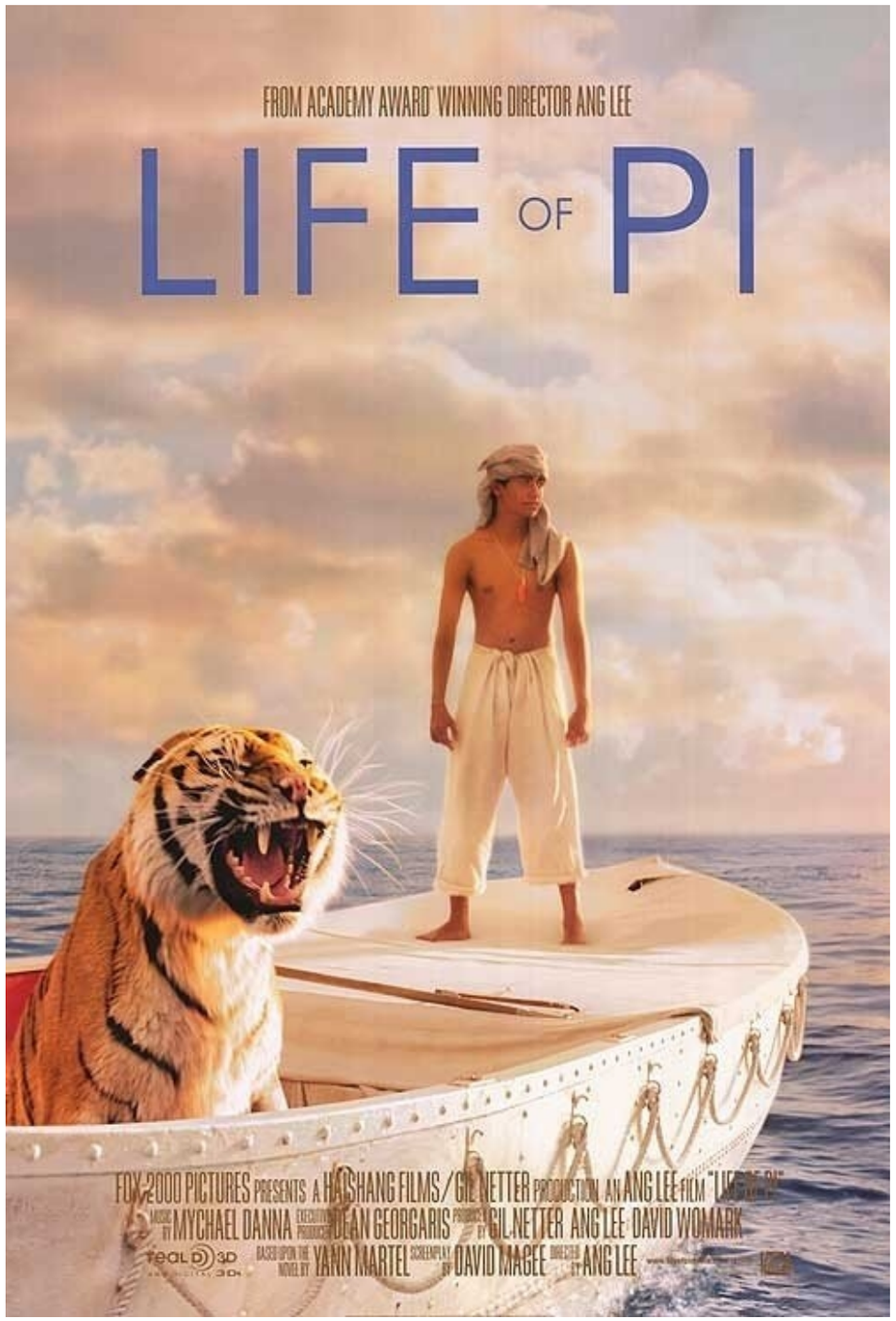
$$\Delta\theta_{S\&M} \propto \nabla_\theta L(\theta) = \frac{1}{|B|}\sum_{s\in B} \nabla_\theta \Big(\pi_\theta(s) - \bar{a}_s\Big)^2 \qquad \text{(By Search \& Mimic)}$$

$$= \frac{1}{|B|}\sum_{s\in B} \nabla_\theta \Big(\pi_\theta(s) - \big(\pi_{\bar{\theta}}(s) + \eta\,\nabla_a Q_w(s,a)\big)\Big)^2$$

$$= \frac{1}{|B|}\sum_{s\in B} \nabla_a Q_w(s,a)\,\nabla_\theta \pi_\theta(s)$$

(For more details, please refer to our UAI 2021 paper, available at https://arxiv.org/pdf/2102.11055.pdf)

Rethinking DDPG: Two Interpretations



What does DDPG actually do?

- Version 1: Maximize $V^{\pi_\theta}(\mu)$ by policy gradient

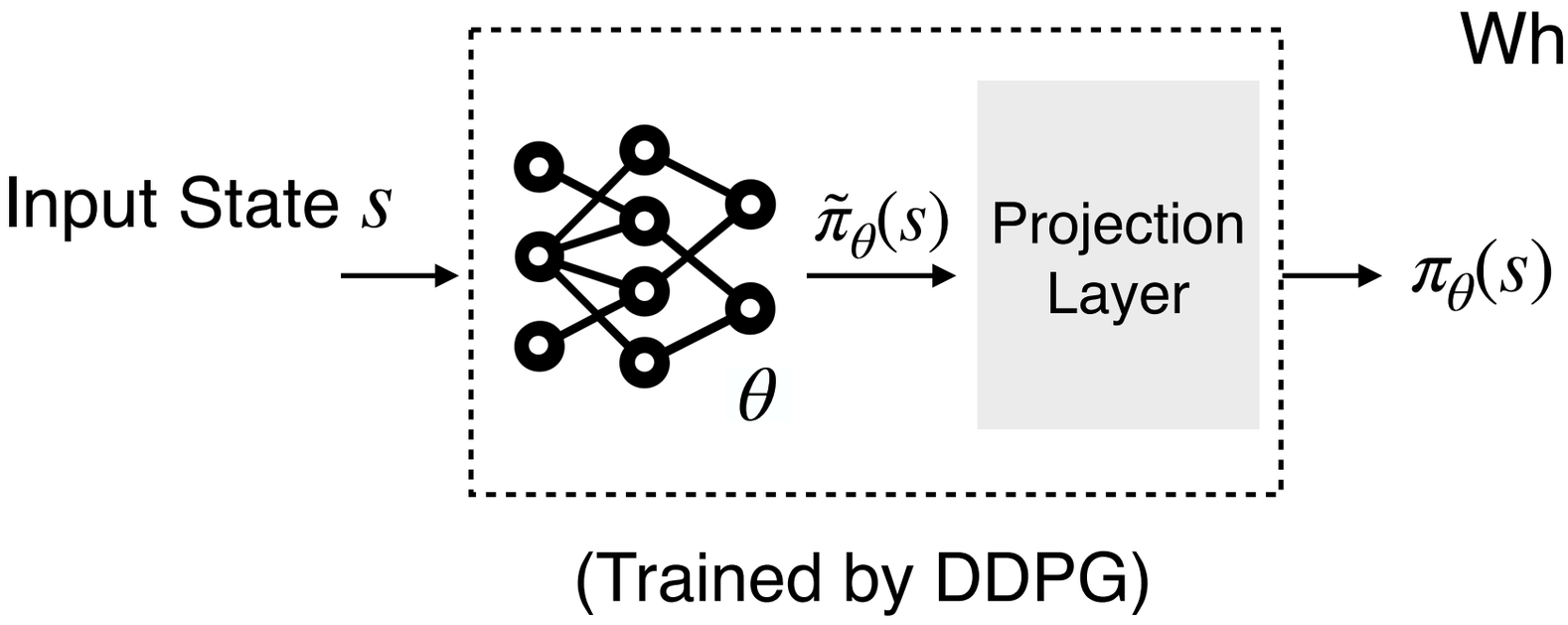
- Version 2: State-wise search in action space & mimicking

- Which version of the story better describes our learning process?

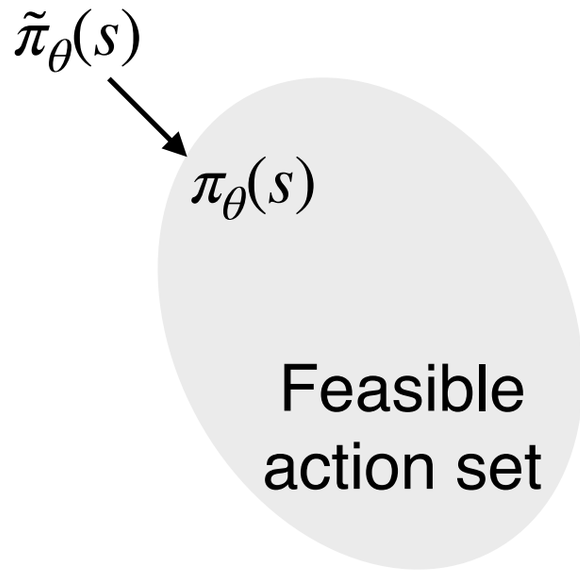
(A2) Action Constraints

Question: How to apply DDPG if there are “action constraints”?

Existing Solution for Action Constraints: Projection Layer



What projection layer does is:



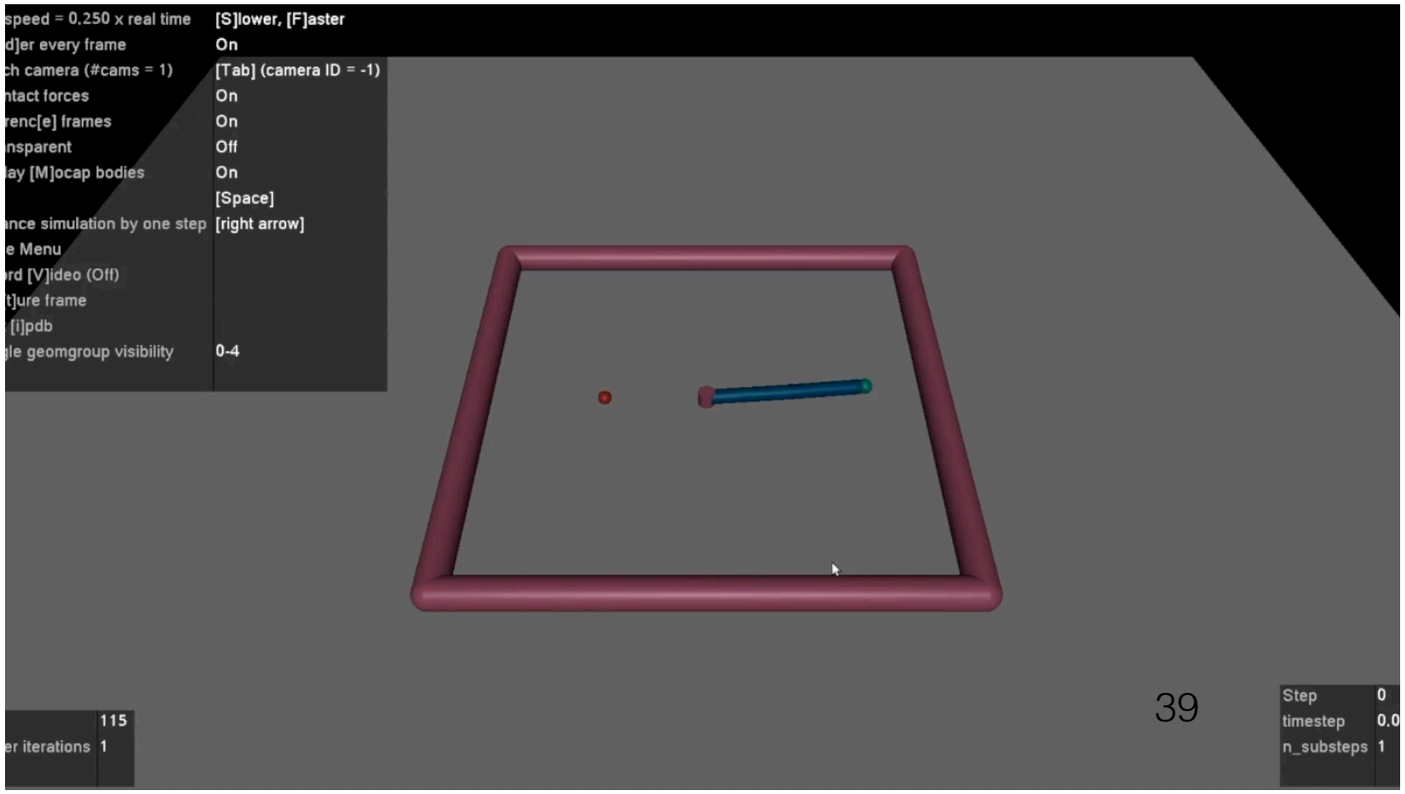
However, this projection layer is problematic!

T. Pham, et al., “Optlayer - practical constrained optimization for deep reinforcement learning in the real world,” ICRA 2018

Why is Projection Layer Problematic?
A Motivating Experiment

Reacher in MuJoCo

- Two joints with torques $u_1, u_2 \in [-1,1]$
- Constraints: $|u_1 + u_2| \leq 0.1, u_1^2 + u_2^2 \leq 0.02$
- Algorithm: DDPG+Projection Layer (trained for 500k steps)

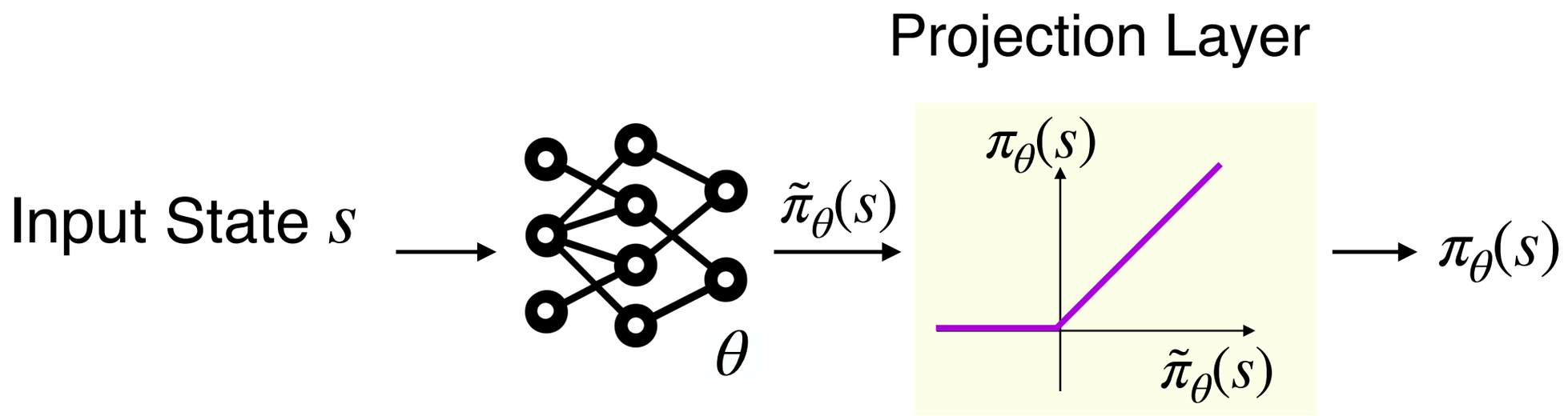


Zero-gradient issue!

What is Zero Gradient in DDPG+Projection Layer? A Motivating Example

Suppose output actions $\pi_\theta(s)$ must be nonnegative (i.e., $\pi_\theta(s) \geq 0$)

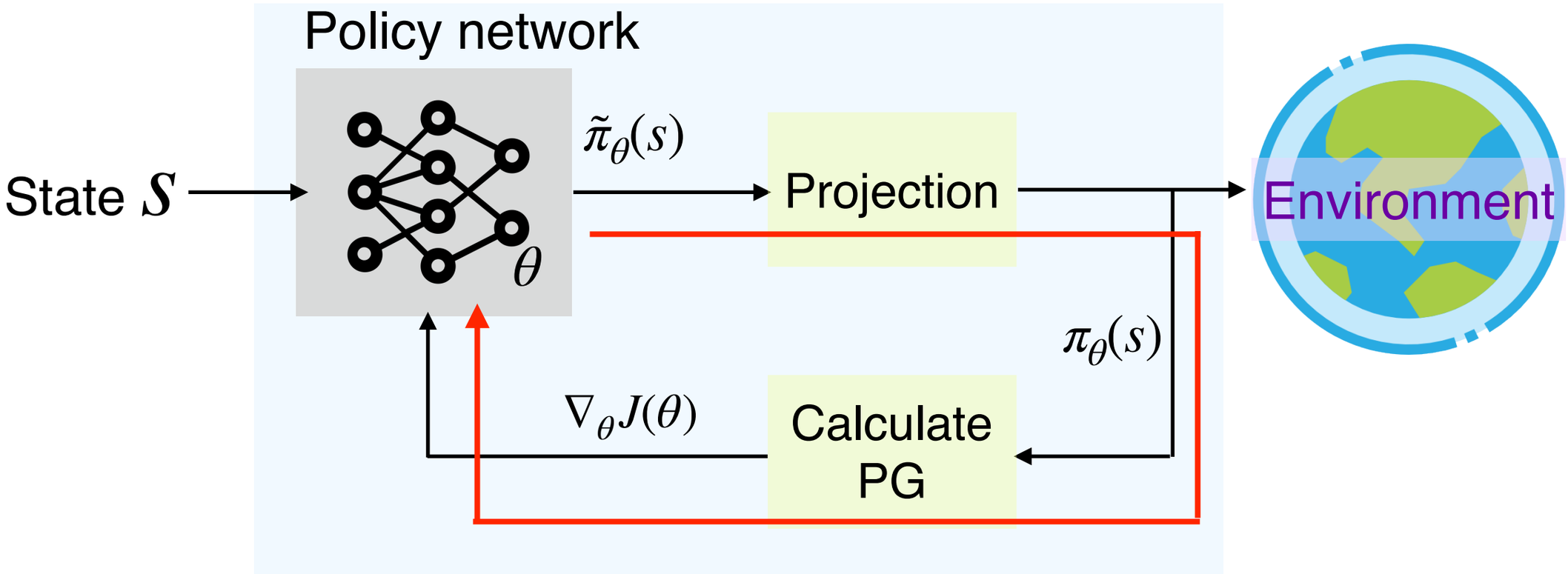
(In this case: Projection = ReLU)



When $\hat{\pi}_\theta(s) < 0$, any small perturbation on θ has no effect!

Zero gradient! (No learning progress at all)

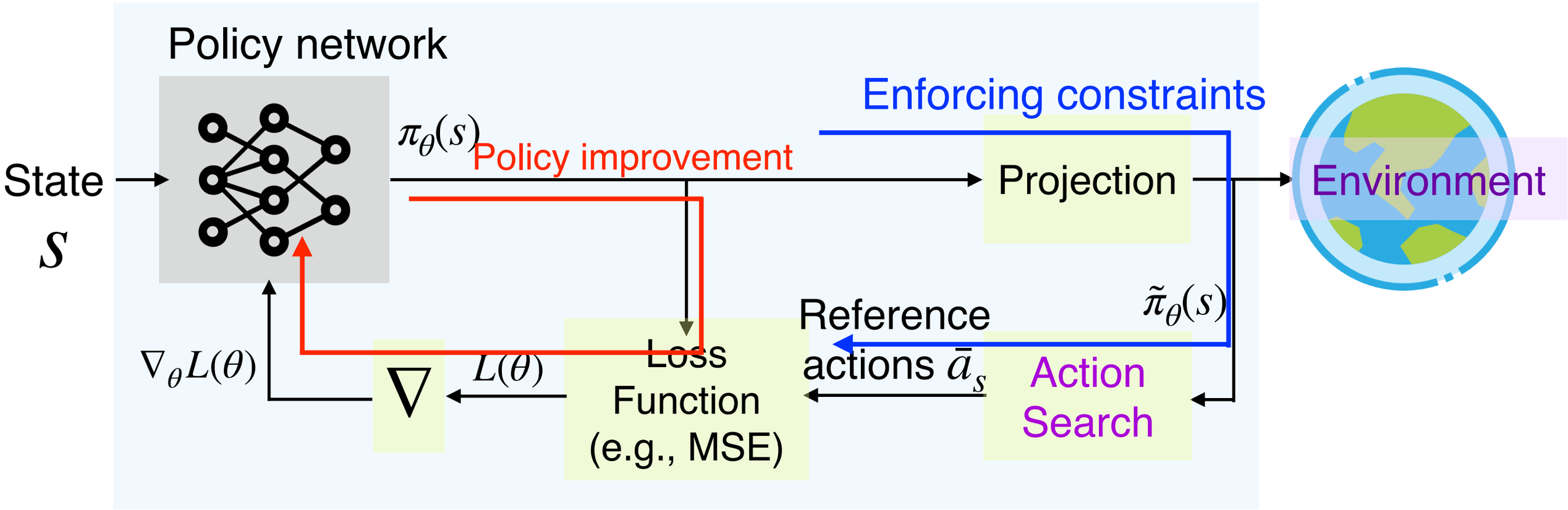
Why DDPG+Projection Layer Fails: Tight Coupling of DPG & Projection



PG and Projection are in the same training loop!

We shall decouple these two components!

“Search & Mimic” Solves Zero Gradient Issue!

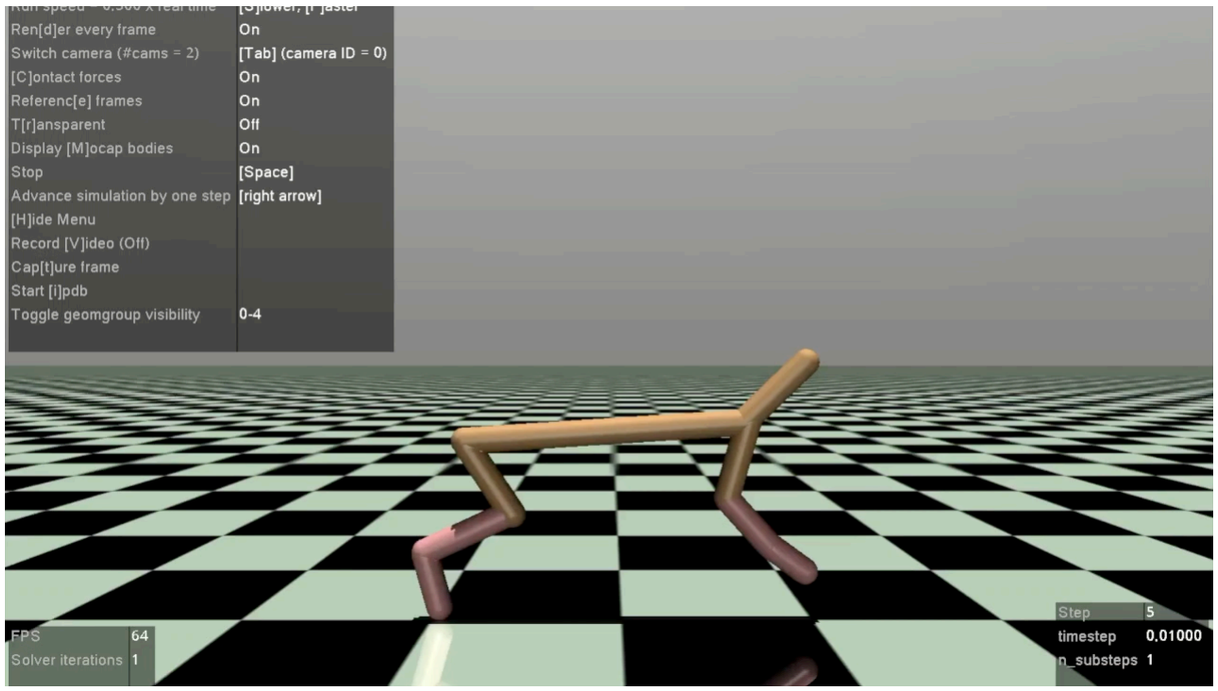


“Search & Mimic” decouples policy improvement and constraint satisfaction

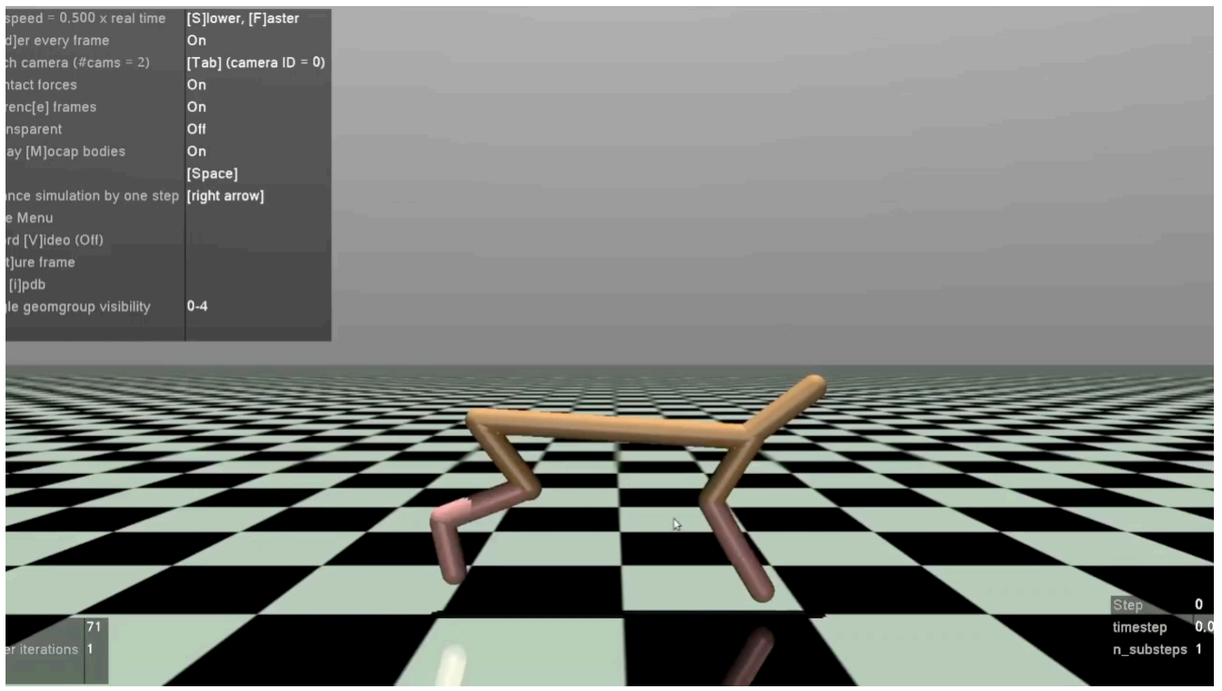
Hence, it completely avoids zero gradient!

Demo: Halfcheetah in MuJoCo

Search & Mimic



DDPG + Projection Layer



- The perspective of *"state-wise search & mimicking"* is very useful in many practical RL problems:

Escaping from Zero Gradient: Revisiting Action-Constrained Reinforcement Learning via Frank-Wolfe Policy Optimization

Jyun-Li Lin[1*] **Wei Hung**[1*] **Shang Hsuan Yang**[1*] **Ping-Chun Hsieh**[1] **Xi Liu**[2]

[1]Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
[2]Applied Machine Learning, Facebook AI, Menlo Park, CA, USA
[*]Equal Contribution

(UAI 2021, available at https://arxiv.org/pdf/2102.11055.pdf)