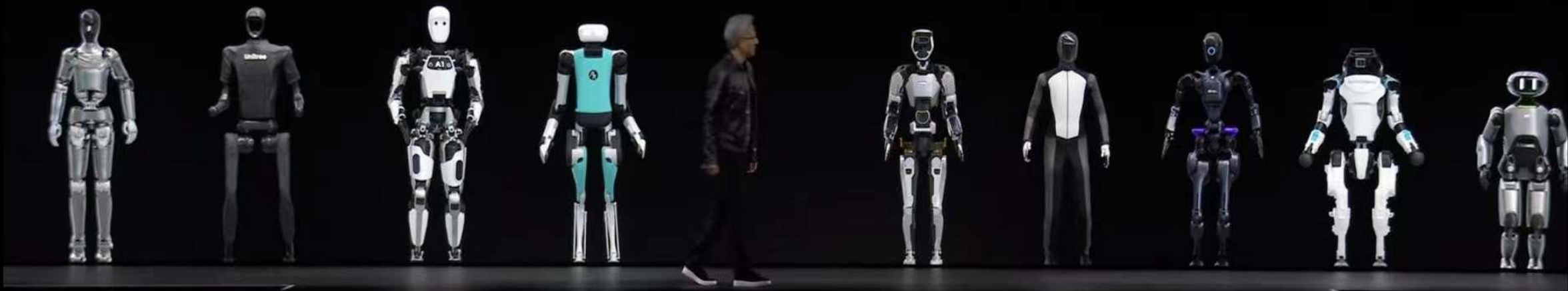




Deep Reinforcement Learning

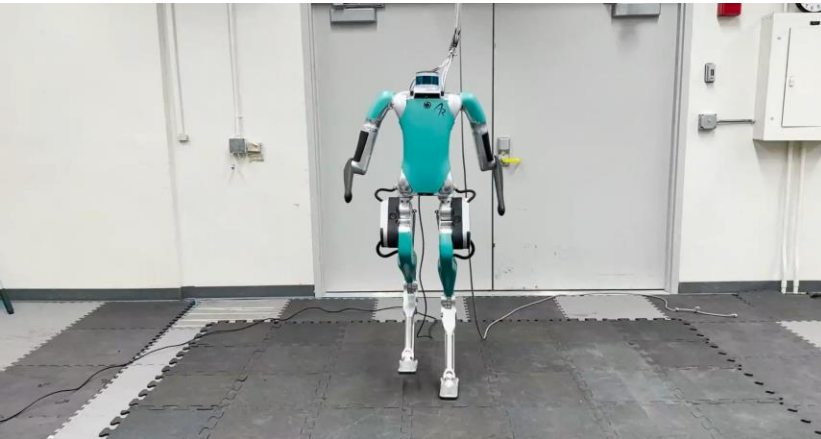
Lecture 4: Deep Q Learning & Policy Gradient

Huazhe Xu
Tsinghua University



AI This Week

2023: Learning
Humanoid Locomotion
with Transformers,
Radosovovic et al., 2023



2024:
NVIDIA GTC
Figure Demo



2025:
Stanford with Galaxea R1





BEHAVIOR ROBOT SUITE

“I think there is a world market for maybe five computers.”

Thomas Watson, president of IBM, 1943



In Lec4

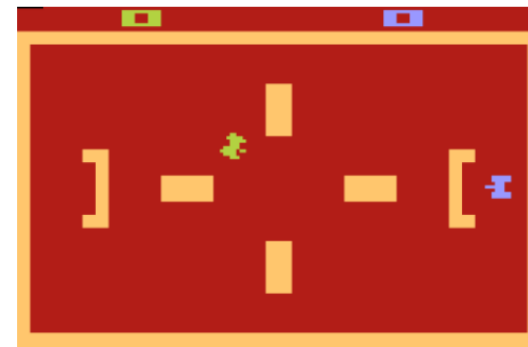
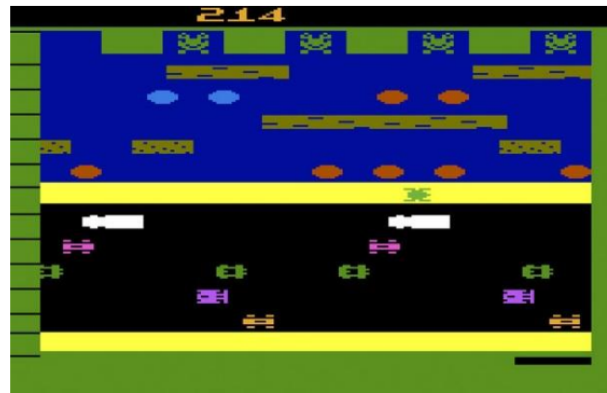
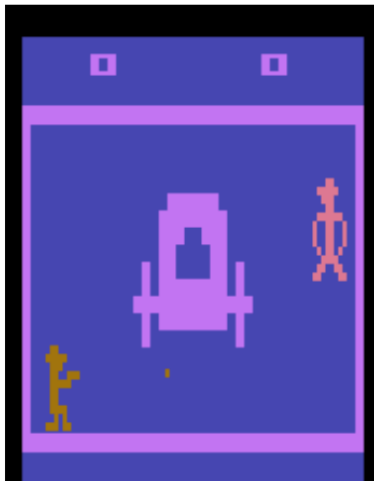
1

Deep Q Learning

2

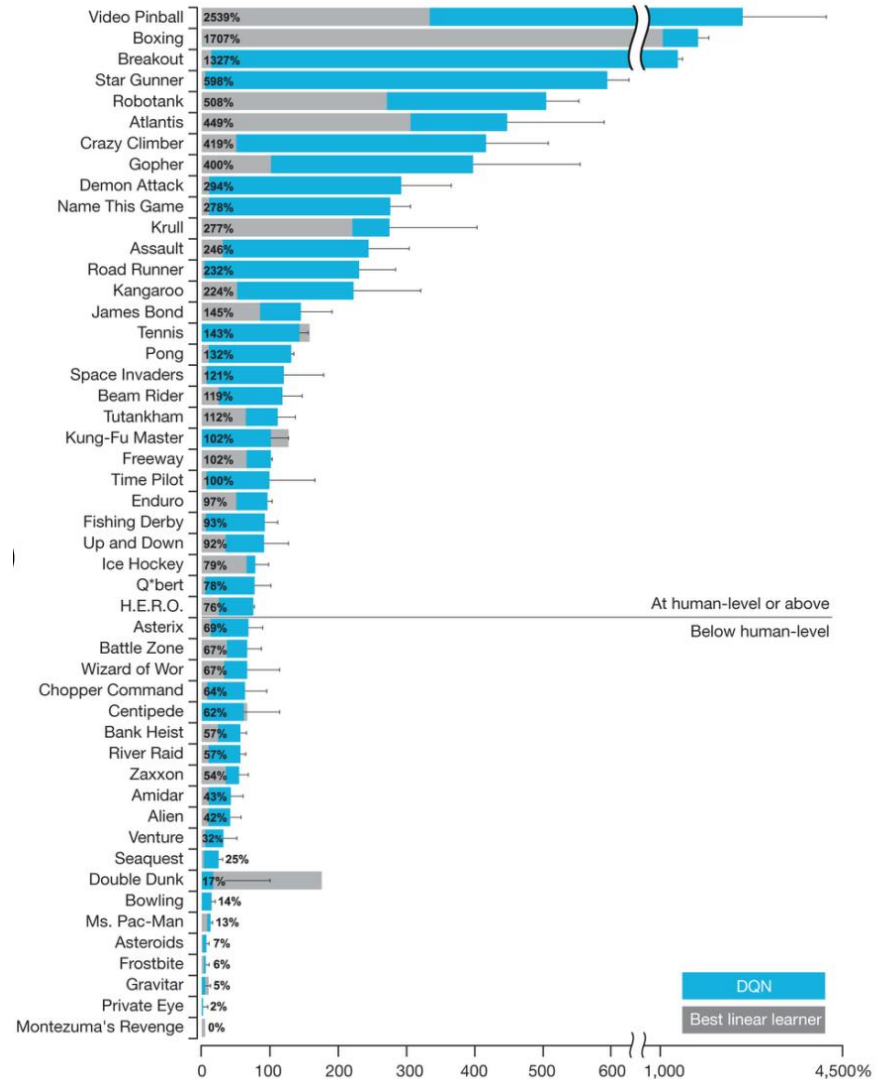
Policy Gradient

If you were Vlad Mnih, what would you do to improve?



Human-level control through deep reinforcement learning
Nature 2015.

Human-level performance



Let recap here... (If you understand this slide, you are on the right track!)

- We've learned value iteration style stuff...
- But sometimes, we don't have transition functions and reward functions.
- We decide to use experience to estimate value: MC, TD
- But it seems it still requires transition function if we need the policy: maybe we should use TD to get Q value
- Even you have Q, you are still passive. You can never know the part you don't know. You need to explore.
- ϵ -greedy for policy to explore
- But you find it is hard to calculate Q values in real games. You decide to use neural networks or other methods to approximate values.
- You add a few tricks such as experience replay, target network, reward clipping. Done!

Idea 1: Overestimation in Q Learning

$$E(\max(X1, X2, \dots)) \geq \max(E(X1), E(X2), \dots)$$

$$\mathcal{L}_i(w_i) = \left(r + \gamma \max_{a'} Q(s', a'; w_i) - Q(s, a; w_i) \right)^2$$

- For each update, you take max first and then average them with multiple samples.

Sol 1: Double DQN

- Dealing with maximization bias of Q-Learning
- Current Q-network w is used to select actions
- Older Q-network w^- is used to evaluate actions

$$I = \left(r + \gamma Q \left(s', \underset{a'}{\operatorname{argmax}} Q(s', a', \mathbf{w}), \mathbf{w}^- \right) - Q(s, a, \mathbf{w}) \right)^2$$

Idea 2: You wasted some computation on the samples that are not informative.

- Samples saved in the replay buffer are used randomly.
- It is not efficient.

Sol 2: Prioritized Experience Replay

- Weight experience according to “surprise” (or error)
- Store experience in priority queue according to DQN error

$$\left| r + \gamma \max_{a'} Q(s', a', \mathbf{w}^-) - Q(s, a, w) \right|$$

- Stochastic Prioritization

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

Idea & Sol 3: The architecture of DQN can be improved.

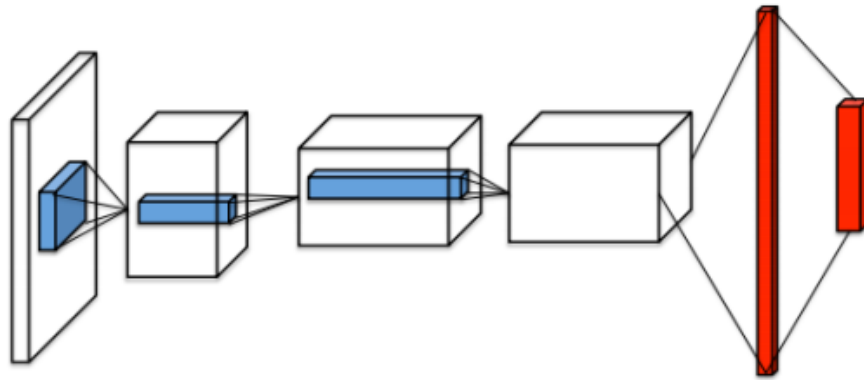
- Split Q-network into two channels
- Action-independent value function $V(s; w)$
- Action-dependent advantage function $A(s, a; w)$

$$Q(s, a; \mathbf{w}) = V(s, \mathbf{w}) + A(s, a; \mathbf{w})$$

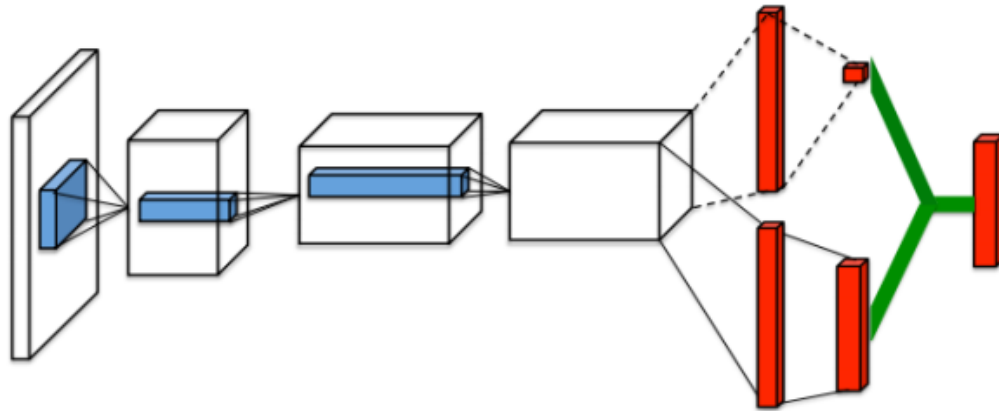
- Advantage function is defined as:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

Dueling Networks



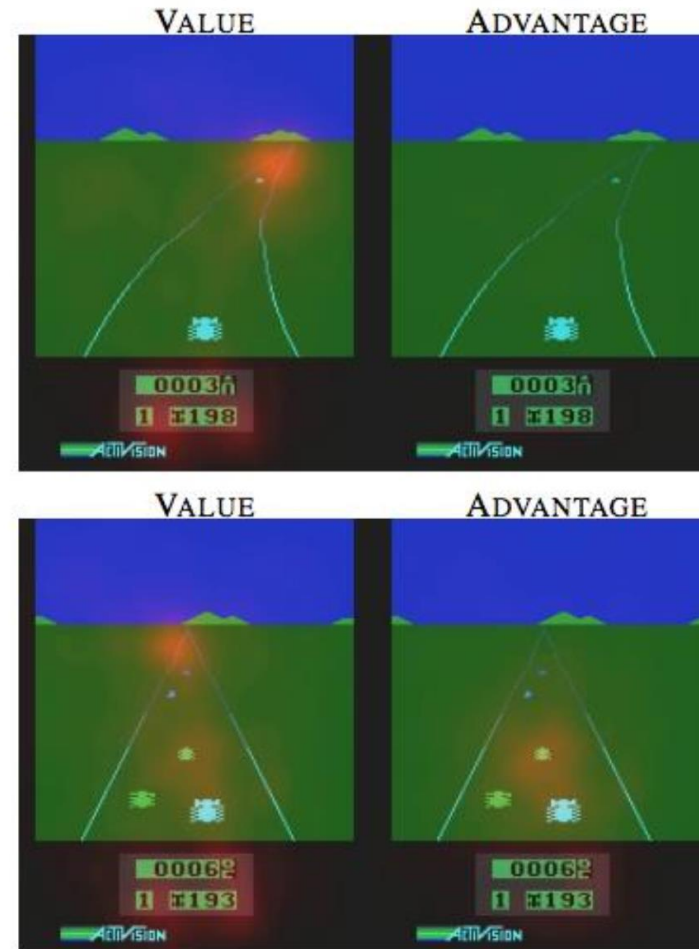
DQN



Dueling Networks

Dueling Network

- The value stream learns to pay attention to the road
- The advantage stream: pay attention only when there are cars immediately in front, so as to avoid collisions.



Idea 4: Use n-step return!

- Sometimes, TD(0) does not yield reward in one step.

$$R_t^{(n)} = \sum_{k=0}^{n-1} \gamma_t^{(k)} R_{t+k+1}$$

$$I = \left(R_t^{(n)} + \gamma_t^{(n)} \max_{a'} Q(S_{t+n}, a', \mathbf{w}) - Q(s, a, \mathbf{w}) \right)^2$$

Rainbow: Combining Improvements in Deep Reinforcement Learning

Matteo Hessel
DeepMind

Joseph Modayil
DeepMind

Hado van Hasselt
DeepMind

Tom Schaul
DeepMind

Georg Ostrovski
DeepMind

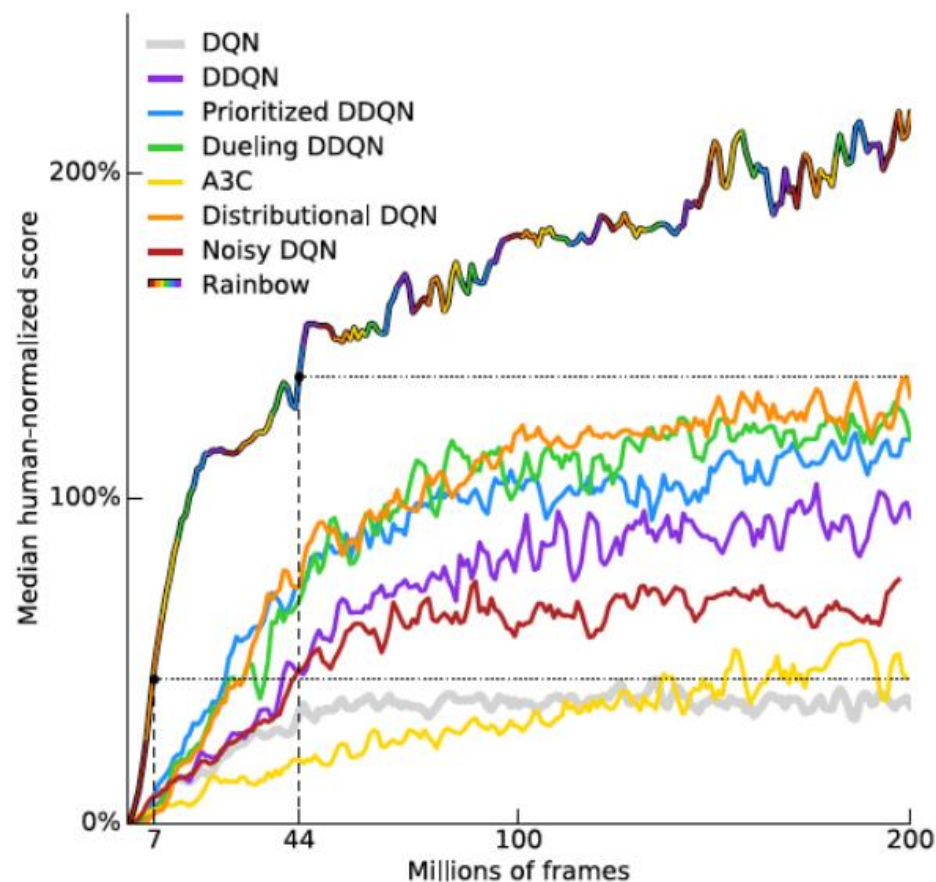
Will Dabney
DeepMind

Dan Horgan
DeepMind

Bilal Piot
DeepMind

Mohammad Azar
DeepMind

David Silver
DeepMind





In Lec4

1

Deep Q Learning

2

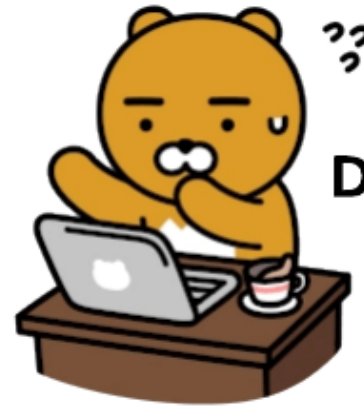
Policy Gradient

Policy Gradient

Policy Gradients



Go Right



Deep Q-Learning

Please wait, I am still
calculating Q value, only
41891 actions left...

Policy-Based Reinforcement Learning

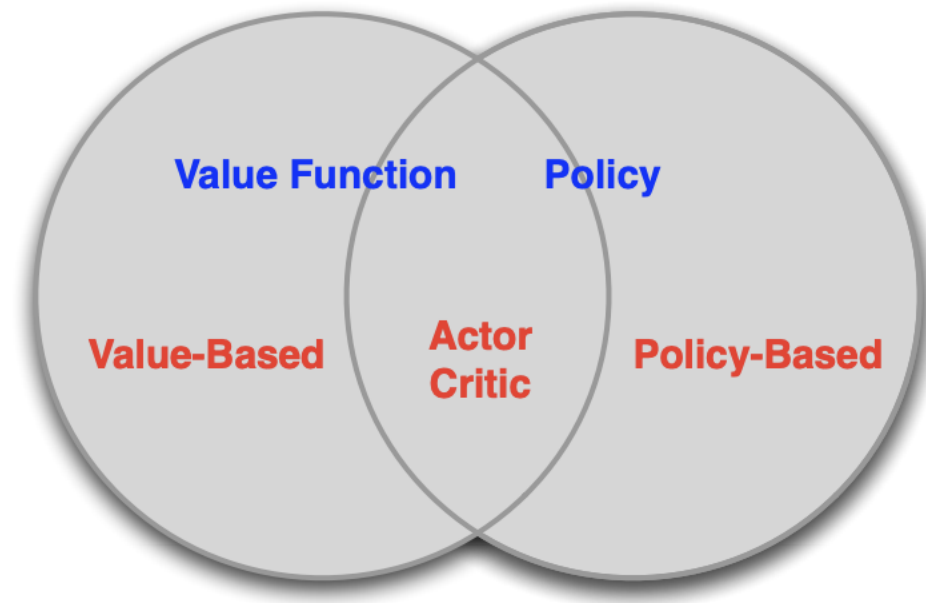
- In this lecture, we will directly parameterize the policy

$$\pi_{\theta}(s, a) = \mathbb{P}[a \mid s, \theta]$$

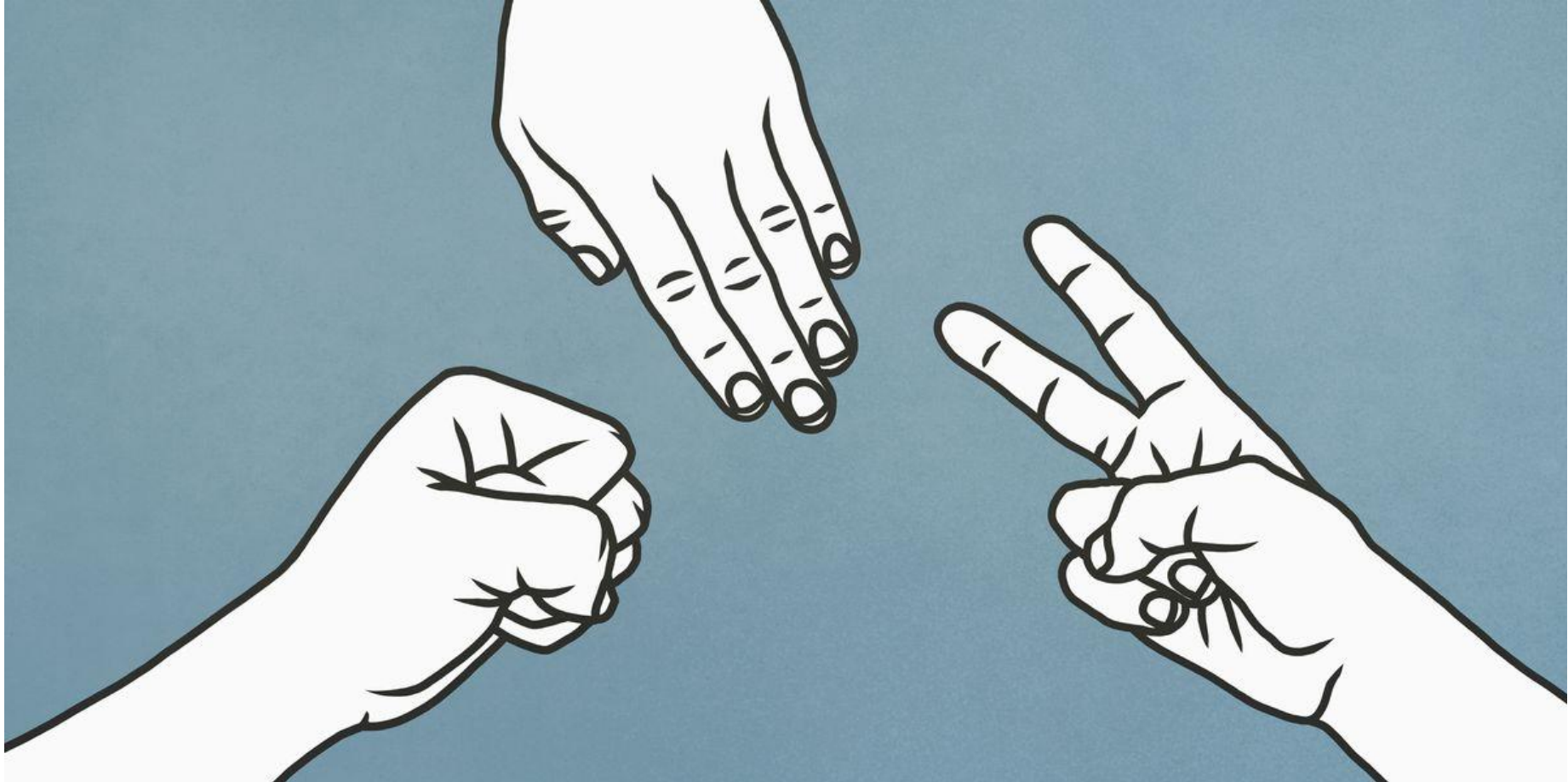
- We will focus again on model-free reinforcement learning

Value-Based vs. Policy-Based RL

- Value-Based
 - Learn Value Function
 - Implicit policy (using Q and ϵ –greedy)
- Policy-Based
 - No Value Function
 - Learn Policy
- Actor-Critic
 - Learn Value Function
 - Learn Policy



Let us play a game!



It is important to have a stochastic policy in some games!



And in the real world, many things are continuous!



Policy Objective Functions

- Goal: given policy $\pi_\theta(s, a)$ with parameters θ , find best θ
- But how do we measure the quality of a policy π_θ ?
- We can use average reward per time-step:

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) \mathcal{R}_s^a$$

- Where $d^{\pi_\theta}(s)$ is stationary distribution of Markov chain for π_θ
- Or you may simply use V^π or return from some state s

Policy Gradient

- Let $J(\theta)$ be any policy objective function
- Policy gradient algorithms search for a local maximum in $J(\theta)$ by ascending the gradient of the policy, w.r.t. parameters θ

$$\Delta\theta = \alpha \nabla_{\theta} J(\theta)$$

- Where $\nabla_{\theta} J(\theta)$ is the policy gradient

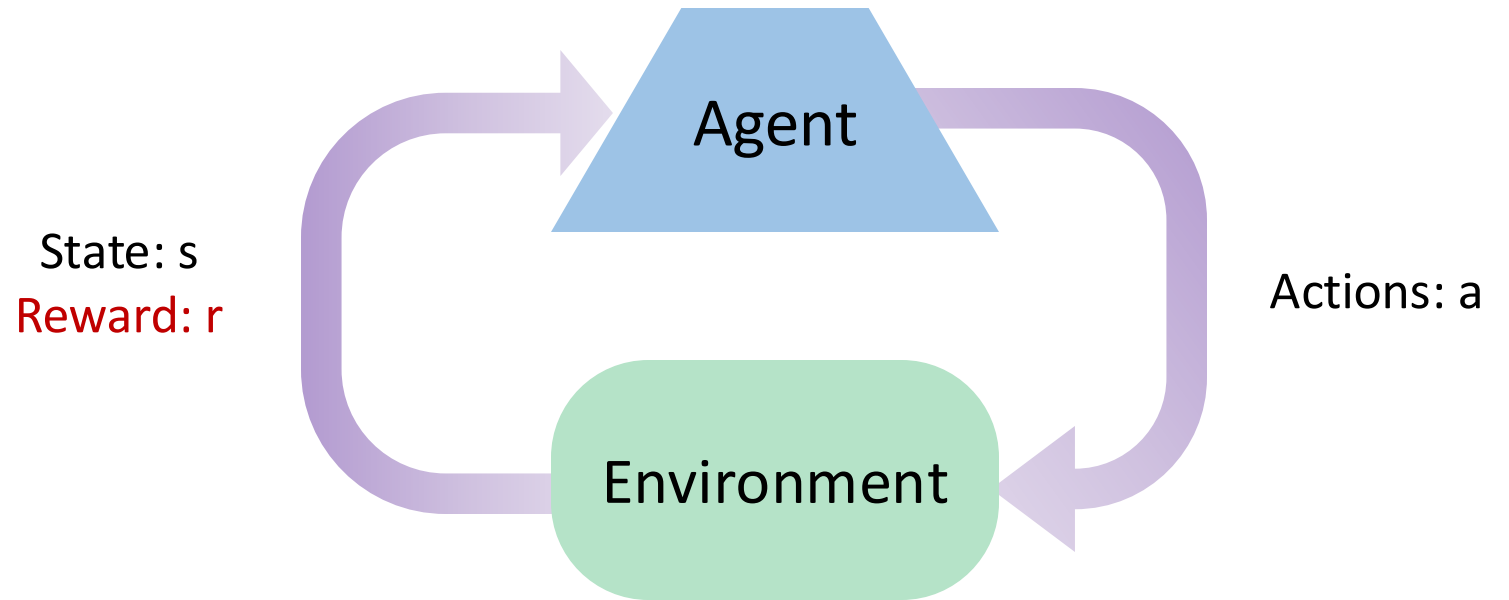
$$\nabla_{\theta} J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$

- α is stepsize

But look at the objective we want to do
gradient ascent!

$$\nabla_{\theta} \mathbb{E}_{\tau} [R(\tau)]$$

Where τ is my trajectory



Two ways

- Make the world differentiable
- Deal with the scalar reward and find another way out to calculate the gradient
- The first way is not a joke. You can use differentiable simulators.

arXiv > cs > arXiv:2202.00817

Computer Science > Machine Learning

[Submitted on 2 Feb 2022 (v1), last revised 22 Aug 2022 (this version, v2)]

Do Differentiable Simulators Give Better Policy Gradients?

H.J. Terry Suh, Max Simchowitz, Kaiqing Zhang, Russ Tedrake



Li et al, DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact

How can we compute the gradient?

- You have to recall a few math tricks before we get started!

$$\nabla_{\theta} \log z = \frac{1}{z} \nabla_{\theta} z$$

$$\mathbb{E}_{x \sim p(x)} [f(x)] = \int_x p(x) f(x) dx$$

$$\frac{a}{b} = \frac{a \cdot p(x)}{b \cdot p(x)}$$

The math from the following slides are important!

- It is been a while after we learned bellman equation. Finally, there are some important formula.
- I am sorry that you have to know, understand, and memorize them.

Let's clarify the setting before taking gradients

- Let a be the action we will choose,
- s be the current state
- θ be the weights of the policy neural network
- The policy neural network will directly output probabilities $\pi(a|s; \theta)$

At each step, we can do gradient ascent to maximize expected reward. (Single action)

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{a \sim p(a|s;\theta)} [r(a)] &= \nabla_{\theta} \sum_a p(a | s; \theta) r(a) \\ &= \sum_a r(a) \nabla_{\theta} p(a | s; \theta) \\ &= \sum_a r(a) p(a | s; \theta) \frac{\nabla_{\theta} p(a | s; \theta)}{p(a | s; \theta)} \\ &= \sum_a r(a) p(a | s; \theta) \nabla_{\theta} \log p(a | s; \theta) \\ &= \mathbb{E}_{a \sim p(a|s;\theta)} [r(a) \nabla_{\theta} \log p(a | s; \theta)]\end{aligned}$$

What if we want to maximize the reward of a trajectory?

- The probability of a trajectory:

$$p(\tau \mid \theta) = \underbrace{\mu(s_0)}_{\text{initial state distribution}} \cdot \prod_{t=0}^{T-1} \underbrace{[\pi(a_t \mid s_t, \theta)]}_{\text{policy}} \cdot \underbrace{p(s_{t+1}, r_t \mid s_t, a_t)}_{\text{transition fn.}}$$

$$\nabla_{\theta} \mathbb{E}_{x \sim p(x \mid s; \theta)} [r(x)] = \mathbb{E}_{x \sim p(x \mid s; \theta)} [r(x) \nabla_{\theta} \log p(x \mid s; \theta)]$$

$$\nabla_{\theta} \mathbb{E}_{\tau} [R(\tau)] = \mathbb{E}_{\tau} \left[\underbrace{\nabla_{\theta} \log p(\tau \mid \theta)}_{\text{What is this?}} \underbrace{R(\tau)}_{\text{Reward of a trajectory}} \right]$$

Calculating the giant monster!

$$\begin{aligned}\log p(\tau \mid \theta) &= \log \mu(s_0) + \log \prod_{t=0}^{T-1} [\pi(a_t \mid s_t, \theta) \cdot p(s_{t+1}, r_t \mid s_t, a_t)] \\ &= \log \mu(s_0) + \sum_{t=0}^{T-1} \log [\pi(a_t \mid s_t, \theta) \cdot p(s_{t+1}, r_t \mid s_t, a_t)] \\ &= \cancel{\log \mu(s_0)} + \sum_{t=0}^{T-1} [\cancel{\log \pi(a_t \mid s_t, \theta)} + \cancel{\log p(s_{t+1}, r_t \mid s_t, a_t)}]\end{aligned}$$

$$\nabla_{\theta} \mathbb{E}_{\tau}[R(\tau)] = \mathbb{E}_{\tau} \left[\underbrace{\nabla_{\theta} \log p(\tau \mid \theta)}_{\text{Derived Above}} \underbrace{R(\tau)}_{\text{Reward of a trajectory}} \right]$$

Now you can take derivative of a trajectory return.

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

REINFORCE --- MC-based method

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i \mid \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Baseline with REINFORCE

- 1. We all know MC causes high _____(variance/bias).
- 2. What if $R(\tau) = 0$ but actually it is the greatest trajectory?



- We noticed a great property:

$$\mathbb{E}_{\pi_{\theta}}[\nabla_{\theta} \log \pi_{\theta}(s, a) B(s)] = 0$$

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

Baseline with REINFORCE

1. We all know MC causes high _____(variance/bias).
 - Why variance is reduced?
2. What if $R(\tau) = 0$ but actually it is the greatest trajectory?

We noticed a great property:

$$\mathbb{E}_{\pi_{\theta}}[\nabla_{\theta} \log \pi_{\theta}(s, a) B(s)] = 0$$

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

If you do not know the math, here is the proof:

Recall that all probability distributions are normalized:

$$\int_x P_\theta(x) = 1$$

Take the gradient of both sides of the normalization condition:

$$\nabla_\theta \int_x P_\theta(x) = \nabla_\theta 1 = 0$$

Use the log derivative trick to get:

$$\begin{aligned} 0 &= \nabla_\theta \int_x P_\theta(x) \\ &= \int_x \nabla_\theta P_\theta(x) \\ &= \int_x P_\theta(x) \nabla_\theta \log P_\theta(x) \end{aligned}$$

Another way to reduce variance

- Policy at time t' cannot affect reward at time t when $t < t'$

$$\begin{aligned}\nabla_{\theta} J(\theta) &\approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} \mid \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right) \\&= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} \mid \mathbf{s}_{i,t}) \sum_{t'=1}^T r(\mathbf{s}_{i,t'}, a_{i,t'}) \right] \\&= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} \mid \mathbf{s}_{i,t}) \hat{Q}(\mathbf{s}_{i,t}, a_{i,t}) \right] \\&= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} \mid \mathbf{s}_{i,t}) \sum_{t'=t}^T r(\mathbf{s}_{i,t'}, a_{i,t'}) \right]\end{aligned}$$

Causality!

The Role of Baselines in Policy Gradient Optimization

Jincheng Mei^{1*}

Wesley Chung²

Valentin Thomas³

Bo Dai¹

Csaba Szepesvári^{4, 5, *}

Dale Schuurmans^{1, 5}

¹Google Research, Brain Team ²Mila, McGill University ³Mila, University of Montreal

⁴DeepMind ⁵Amii, University of Alberta

{jcmei, bodai, szepi, schuurmans}@google.com {wesley.chung2, vltm.thomas}@gmail.com

- by showing that the variance of natural policy gradient estimates remains unbounded with or without a baseline, we find that variance reduction cannot explain their utility in this setting. Instead, the analysis reveals that the primary effect of the value baseline is to **reduce the aggressiveness of the updates** rather than their variance.

How to parameterize your policy?

- Of course, neural networks!
- Softmax
- Gaussian

Softmax Policy

- Weight actions using linear combination of features $\phi(s, a)^\top \theta$
- Probability of action is proportional to exponentiated weight

$$\pi_\theta(s, a) \propto e^{\phi(s, a)^\top \theta}$$

- The score function is

$$\nabla_\theta \log \pi_\theta(s, a) = \phi(s, a) - \mathbb{E}_{\pi_\theta}[\phi(s, \cdot)]$$

Gaussian Policy


- We can also use Gaussian policy.
- Mean is a linear combination of state features $\mu(s) = \phi(s)^\top \theta$
- Variance may be fixed σ^2 , or can also be parameterized
- Policy is Gaussian, $a \sim \mathcal{N}(\mu(s), \sigma^2)$
- The score function is

$$\nabla_{\theta} \log \pi_{\theta}(s, a) = \frac{(a - \mu(s))\phi(s)}{\sigma^2}$$

Can we use other people's experience with PG?

- No. Why?
- It is on-policy.
- Can we make it off-policy?
- What else is wrong with PG?

On-policy troubles

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


It has to be trajectory from “me”!

We want to use some else'experience.



Including the past of yourself

Importance Sampling

- **Importance sampling** is a **Monte Carlo method** for evaluating properties of a particular **distribution**, while only having **samples generated from a different distribution** than **the distribution of interest**. (Wikipedia)
- $f(x)$, where $x \sim p(x)$:

$$E[f(x)] = \int f(x)p(x)dx \approx \frac{1}{n} \sum_i f(x_i)$$

$$E[f(x)] = \int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx \approx \frac{1}{n} \sum_i f(x_i)\frac{p(x_i)}{q(x_i)}$$

Off-Policy PG Objective

- Use samples from another policy $\bar{p}(\tau)$

$$J(\theta) = E_{\tau \sim \bar{p}(\tau)} \left[\frac{p_{\theta}(\tau)}{\bar{p}(\tau)} r(\tau) \right]$$

$$\frac{p_{\theta}(\tau)}{\bar{p}(\tau)} = \frac{\cancel{p(s_1)} \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}}{\cancel{p(s_1)} \prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}} = \frac{\prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t)}$$

Off-Policy PG

$$\theta^* = \arg \max_{\theta} J(\theta) \quad J(\theta) = E_{\tau \sim p_{\theta}(\tau)} [r(\tau)]$$

$$\begin{aligned} \nabla_{\theta'} J(\theta') &= E_{\tau \sim p_{\theta}(\tau)} \left[\frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right] \\ &= E_{\tau \sim p_{\theta}(\tau)} \left[\left(\prod_{t=1}^T \frac{\pi_{\theta'}(\mathbf{a}_t \mid \mathbf{s}_t)}{\pi_{\theta}(\mathbf{a}_t \mid \mathbf{s}_t)} \right) \left(\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t \mid \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right] \\ &= E_{\tau \sim p_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t \mid \mathbf{s}_t) \left(\prod_{t'=1}^t \frac{\pi_{\theta'}(\mathbf{a}_{t'} \mid \mathbf{s}_{t'})}{\pi_{\theta}(\mathbf{a}_{t'} \mid \mathbf{s}_{t'})} \right) \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \right] \end{aligned}$$

Future actions do not affect current weight



In Lec4

1

Deep Q Learning

2

Policy Gradient

In next lecture, you will learn

- Off-policy PG
- Actor Critic

And that is almost all the basic stuff you need to know before we move to the advanced world.