



Introduction to Embodied AI

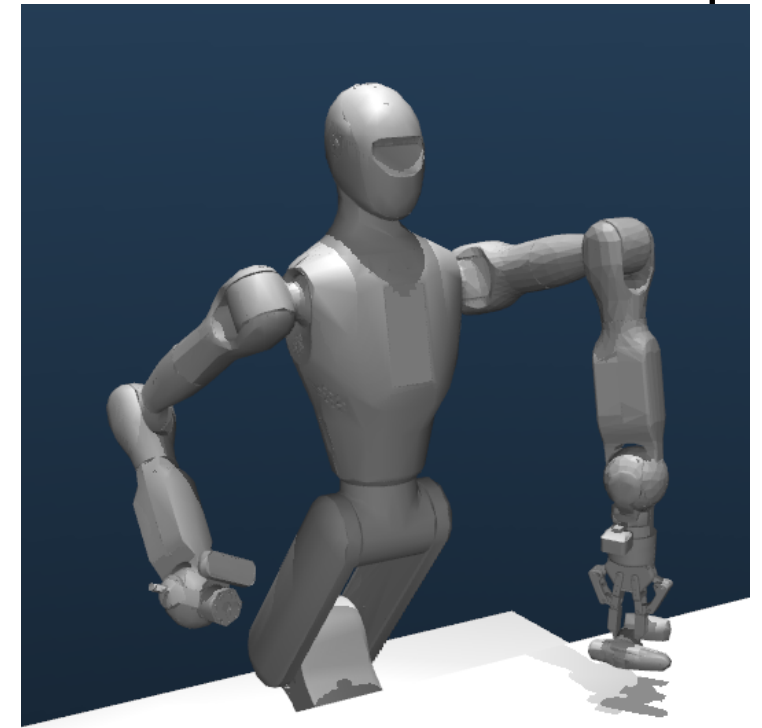
Lecture 10 – Policy III

Prof. He Wang



Logistics: Assignment 2

- Object Pose Estimation from Point Cloud Observations (15 points towards final score)
- Each student needs to do this part independently, experiments will be done in groups.
- Release date: 4/14 (Monday) night
- Due date: 4/30 Wednesday 11:59PM
- If 1 day (0 - 24 hours) past the deadline, 15% off
- If 2 day (24 - 48 hours) past the deadline, 30% off
- Zero credit if more than 2 days.



Logistics: Plan After Midterm

Lecture	04/28/2025 15:10 Monday	Robotic Task I	Legged Robot Locomotion
Assignment	04/30/2025 Wednesday	Assignment #3 released	[Assignment #3]
Due	04/30/2025 23:59 Wednesday	Assignment #2 due	
Lecture	05/12/2025 15:10 Monday	Robotic Task II	Robotic Gripper Manipulation and Dexterous Hand Manipulation
Due	05/14/2025 23:59 Wednesday	Assignment #3 due	
Lecture	05/19/2025 15:10 Monday	Multimodal LLM in EAI I	Pretrained Large Models (CLIP) and GPT-4v/4o and other Multimodal LLMs
Lecture	05/26/2025 15:10 Monday	Multimodal LLM in EAI II	Modular General Manipulation System and End-to-end Vision-Language-Action (VLA) Models
Experiment	06/02/2025 15:10 Monday	Real World Experiments	Real world experiments (at PKU-Galbot Joint Lab)

Critics as Baselines

Policy Gradient

Starting from the policy gradient in REINFORCE, to reduce variance, we introduce

1. reward-to-go (causality)
2. replace reward-to-go by Q (the expectation value of reward-to-go given s_t, a_t)
3. Subtract a state-based baseline: $V(s_t)$

To obtain the following:

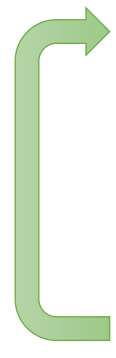
$$\nabla_{\theta} J(\theta) = \mathbb{E}_{(s_t, a_t) \sim d^{\pi}} \left[\nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t) (Q(s_t, a_t) - V(s_t)) \right]$$

Note that there is no approximation yet and this policy gradient is on-policy.

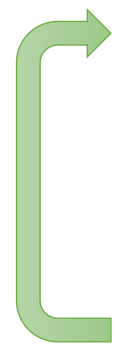
Actor-critic algorithms (with discount)

Actor-critic algorithms (with discount)

batch actor-critic algorithm:

- 
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
 2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
 3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
 2. update $\hat{V}_\phi^\pi$ using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$
 3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$
 4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Critics as state-dependent baselines

Actor-critic:
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+1}) - \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t}) \right)$$

+ lower variance (due to critic)

- not unbiased (if the critic is not perfect)

Policy gradient:
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) - b \right)$$

+ no bias

- higher variance (because single-sample estimate)

can we use $\hat{V}_{\phi}^{\pi}$ and still keep the estimator unbiased?

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right) - \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t}) \right)$$

+ no bias

+ lower variance (baseline is closer to rewards)

Control variates: action-dependent baselines

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$$

$$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} [Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$$

$$\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - V_\phi^\pi(\mathbf{s}_t)$$

+ no bias

- higher variance (because single-sample estimate)

$$\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - Q_\phi^\pi(\mathbf{s}_t, \mathbf{a}_t)$$

+ goes to zero in expectation if critic is correct!

- not correct

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\hat{Q}_{i,t} - Q_\phi^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right) + \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_{i,t})} [Q_\phi^\pi(\mathbf{s}_{i,t}, \mathbf{a}_t)]$$

use a critic *without* the bias (still unbiased),
provided second term can be evaluated

Gu et al. 2016 (Q-Prop)

Eligibility traces & n-step returns

$$\hat{A}_C^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{t+1}) - \hat{V}_\phi^\pi(\mathbf{s}_t)$$

+ lower variance

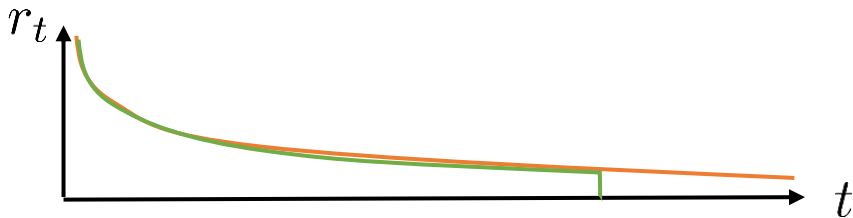
- higher bias if value is wrong (it always is)

$$\hat{A}_{MC}^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}_\phi^\pi(\mathbf{s}_t)$$

+ no bias

- higher variance (because single-sample estimate)

Can we combine these two, to control bias/variance tradeoff?



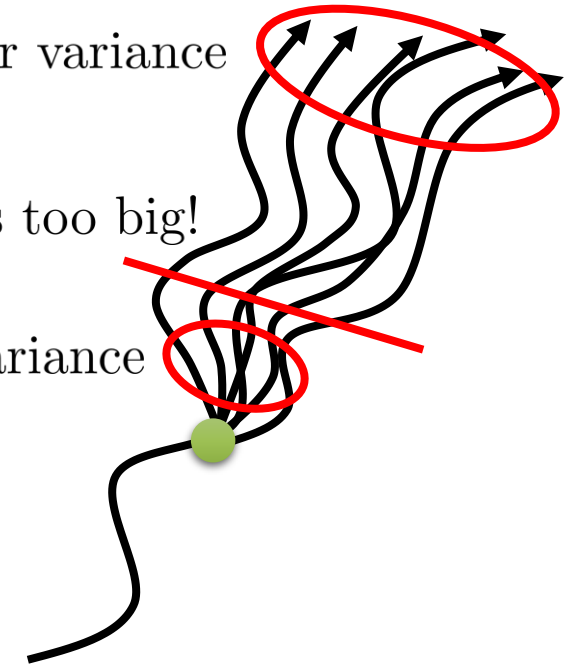
$$\hat{A}_n^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}_\phi^\pi(\mathbf{s}_t) + \gamma^n \hat{V}_\phi^\pi(\mathbf{s}_{t+n})$$

choosing $n > 1$ often works better!

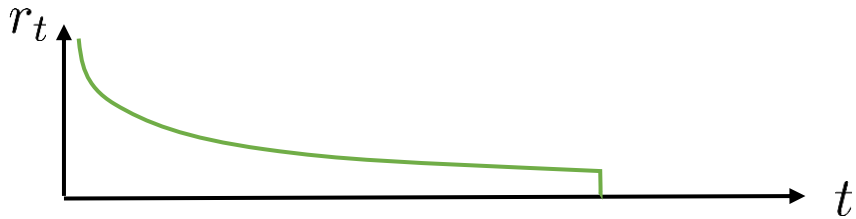
bigger variance

cut here before variance gets too big!

smaller variance



Generalized advantage estimation



Do we have to choose just one n?

Cut everywhere all at once!

$$\hat{A}_n^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}_\phi^\pi(\mathbf{s}_t) + \gamma^n \hat{V}_\phi^\pi(\mathbf{s}_{t+n})$$

$$\hat{A}_{\text{GAE}}^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{n=1}^{\infty} w_n \hat{A}_n^\pi(\mathbf{s}_t, \mathbf{a}_t)$$

weighted combination of n-step returns

How to weight?

Mostly prefer cutting earlier (less variance)

$w_n \propto \lambda^{n-1}$ exponential falloff

$$\hat{A}_{\text{GAE}}^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma((1 - \lambda)\hat{V}_\phi^\pi(\mathbf{s}_{t+1}) + \lambda(r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + \gamma((1 - \lambda)\hat{V}_\phi^\pi(\mathbf{s}_{t+2}) + \lambda r(\mathbf{s}_{t+2}, \mathbf{a}_{t+2}) + \dots))$$

$$\hat{A}_{\text{GAE}}^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{\infty} (\gamma\lambda)^{t'-t} \delta_{t'} \quad \delta_{t'} = r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) + \gamma\hat{V}_\phi^\pi(\mathbf{s}_{t'+1}) - \hat{V}_\phi^\pi(\mathbf{s}_{t'})$$

← similar effect as discount!

option 1:
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$$

remember this?

discount = variance reduction!

Recap: discount factors for policy gradients

$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$$

with critic:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\overbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1}) - \hat{V}_\phi^\pi(\mathbf{s}_{i,t})}^{\hat{A}^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} \right)$$

what about (Monte Carlo) policy gradients?

option 1: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$

option 2: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T \gamma^{t-1} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$

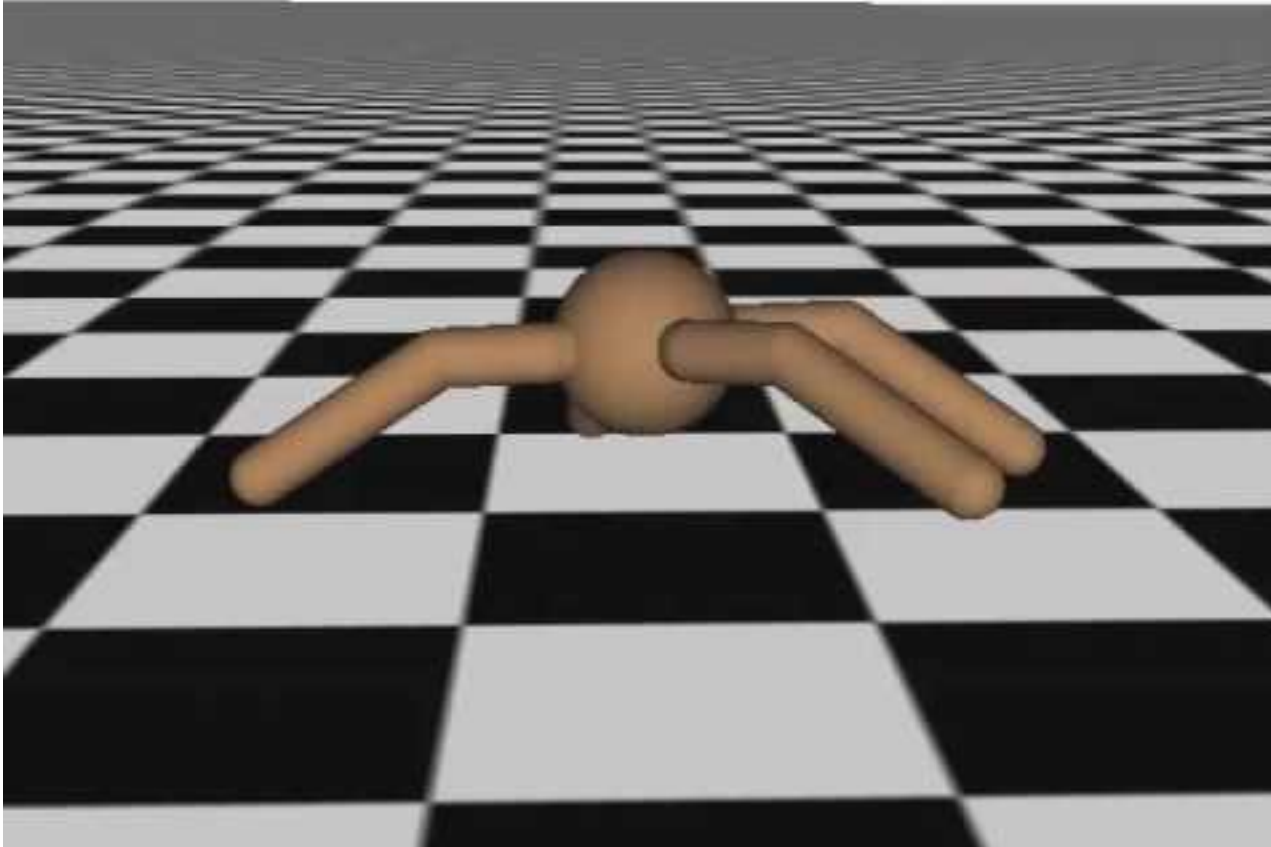
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\sum_{t'=t}^T \gamma^{t'-t} \textcircled{1} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$$

(later steps matter less) $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \gamma^{t-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left(\sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$

not the same!

Actor-critic examples

Iteration 80



Demo: learning to run and stand up

<https://www.youtube.com/watch?v=SHLuf2ZBQSw>

- High dimensional continuous control with generalized advantage estimation (Schulman, Moritz, L., Jordan, Abbeel '16)
- Batch-mode actor-critic
- Blends Monte Carlo and function approximator estimators (GAE)

Observations = joint angles, joint velocities, and some cartesian positions

Actions = joint torques

Similar to the MuJoCo Humanoid in OpenAI Gym

Actor-critic examples



Demo: labyrinth

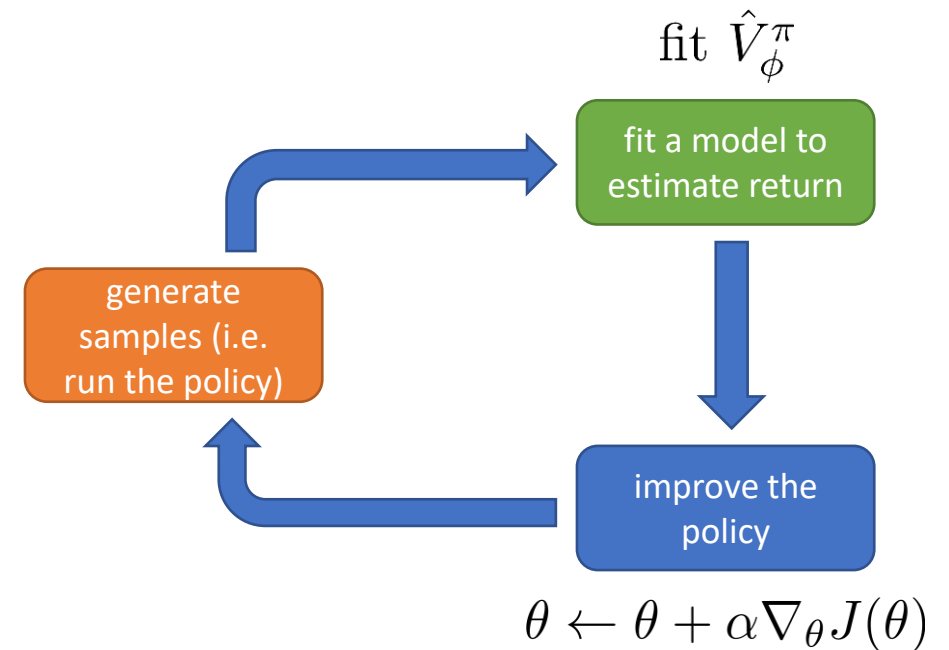
<https://www.youtube.com/watch?v=nMR5mjCFZCw>

- Asynchronous methods for deep reinforcement learning (Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu '16)
- Online actor-critic, parallelized batch
- N-step returns with $N = 4$
- Single network for actor and critic

An agent collecting rewards in previously unseen mazes using only raw pixels as input. The agent was trained using the Asynchronous Advantage Actor-Critic (A3C) algorithm and was only rewarded for picking up apples and orange portals during training.

Review

- Actor-critic algorithms:
 - Actor: the policy
 - Critic: value function
 - Reduce variance of policy gradient
- Policy evaluation
 - Fitting value function to policy
- Discount factors
 - Carpe diem Mr. Robot 
 - ...but also a variance reduction trick
- Actor-critic algorithm design
 - One network (with two heads) or two networks
 - Batch-mode, or online (+ parallel)
- State-dependent baselines
 - Another way to use the critic
 - Can combine: n-step returns or GAE



Actor-critic suggested readings

- Classic papers
 - Sutton, McAllester, Singh, Mansour (1999). Policy gradient methods for reinforcement learning with function approximation: actor-critic algorithms with value function approximation
- Deep reinforcement learning actor-critic papers
 - Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu (2016). Asynchronous methods for deep reinforcement learning: A3C -- parallel online actor-critic
 - Schulman, Moritz, L., Jordan, Abbeel (2016). High-dimensional continuous control using generalized advantage estimation: batch-mode actor-critic with blended Monte Carlo and function approximator returns
 - Gu, Lillicrap, Ghahramani, Turner, L. (2017). Q-Prop: sample-efficient policy-gradient with an off-policy critic: policy gradient with Q-function control variate

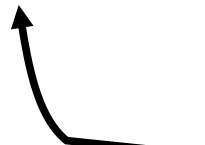
Off-Policy RL

Policy gradient is on-policy

$$\theta^* = \arg \max_{\theta} J(\theta)$$

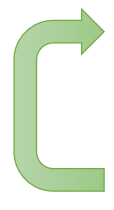
$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$$

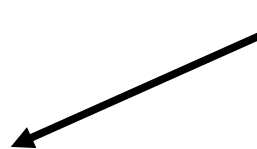
$$\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

 this is trouble...

- Neural networks change only a little bit with each gradient step
- On-policy learning can be extremely inefficient!

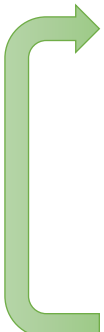
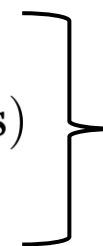
REINFORCE algorithm:

- 
1. sample $\{\tau^i\}$ from $\pi_{\theta}(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
 2. $\nabla_{\theta} J(\theta) \approx \sum_i (\sum_t \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^i|\mathbf{s}_t^i)) (\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i))$
 3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

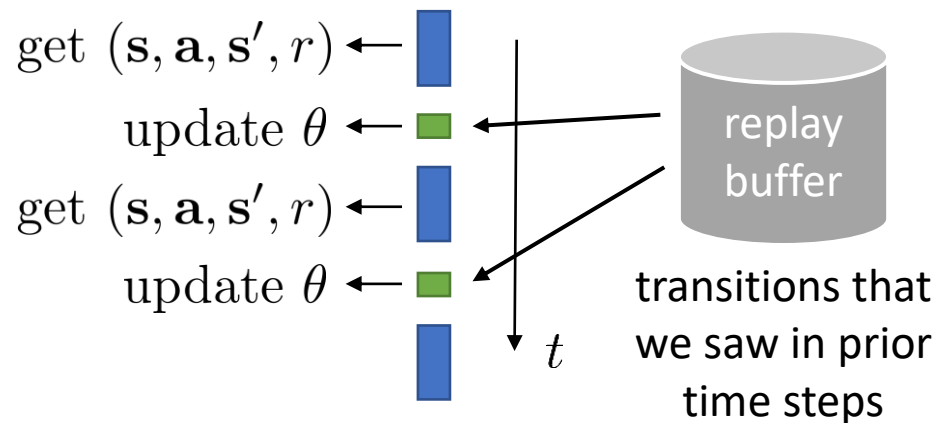
 can't just skip this!

Can we remove the on-policy assumption in AC?

online actor-critic algorithm:

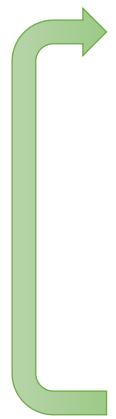
- 
1. take action $\mathbf{a} \sim \pi_{\theta}(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
 2. update $\hat{V}_{\phi}^{\pi}$ using target $r + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}')$
 3. evaluate $\hat{A}^{\pi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}') - \hat{V}_{\phi}^{\pi}(\mathbf{s})$
 4. $\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}|\mathbf{s}) \hat{A}^{\pi}(\mathbf{s}, \mathbf{a})$
 5. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$
- 
- form a **batch** by using old previously seen transitions

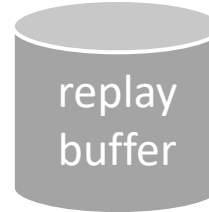
off-policy actor-critic



What if We use Off-Policy Data in AC

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
 2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
 3. update $\hat{V}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
 4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
- not the action π_θ would have taken!
- not the right target value



$$\mathcal{L}(\phi) = \frac{1}{N} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$$

batch size

This algorithm is broken!

Can you spot the problems?

Fixing the value function

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{V}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i) - V_\phi^\pi(\mathbf{s}_i)$
5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

not the right target value

not the action π_θ would have taken!

where does this come from?

3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
 $= r_i + \gamma \hat{Q}_\phi^\pi(\mathbf{s}'_i, \mathbf{a}'_i)$

not from replay buffer $\mathcal{R}$!

$$\mathbf{a}'_i \sim \pi_\theta(\mathbf{a}'_i|\mathbf{s}'_i)$$

$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t] = E_{\mathbf{a} \sim \pi(\mathbf{a}_t|\mathbf{s}_t)} [Q(\mathbf{s}_t, \mathbf{a}_t)]$$

~~$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t]$$~~

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t]$$

“total reward we get if we take $\mathbf{a}_t$ in $\mathbf{s}_t$...
... and then follow the policy π ”

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_i \left\| \hat{Q}_\phi^\pi(\mathbf{s}_i, \mathbf{a}_i) - y_i \right\|^2$$

Fixing the policy update

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
 2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
 3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{Q}_\phi^\pi(\mathbf{s}'_i, \mathbf{a}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
 4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = Q(\mathbf{s}_i, \mathbf{a}_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 5. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
 6. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

not the action π_θ would have taken!

use the same trick, but this time for $\mathbf{a}_i$ rather than $\mathbf{a}'_i$!

sample $\mathbf{a}_i^\pi \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$$

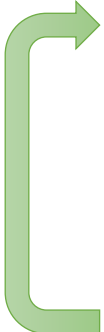
↑
not from replay buffer $\mathcal{R}$!

higher variance, but convenient
why is higher variance OK here?

in practice: $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$

What else is left?

online actor-critic algorithm:

- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
 2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
 3. update $\hat{Q}_\phi^\pi$ using targets $y_i = r_i + \gamma \hat{Q}_\phi^\pi(\mathbf{s}'_i, \mathbf{a}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
 4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi | \mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}_i^\pi)$ where $\mathbf{a}_i^\pi \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Is there any remaining problem?

$\mathbf{s}_i$ didn't come from $p_\theta(\mathbf{s})$

nothing we can do here, just accept it

intuition: we want optimal policy on $p_\theta(\mathbf{s})$
but we get optimal policy on a *broader* distribution

Off-policy learning & importance sampling

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$$

what if we don't have samples from $p_{\theta}(\tau)$?
(we have samples from some $\bar{p}(\tau)$ instead)

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

$$p_{\theta}(\tau) = p(\mathbf{s}_1) \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\frac{p_{\theta}(\tau)}{\bar{p}(\tau)} = \frac{\cancel{p(\mathbf{s}_1)} \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}}{\cancel{p(\mathbf{s}_1)} \prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}} = \frac{\prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t)}$$

importance sampling

$$\begin{aligned} E_{x \sim p(x)}[f(x)] &= \int p(x) f(x) dx \\ &= \int \frac{q(x)}{q(x)} p(x) f(x) dx \\ &= \int q(x) \frac{p(x)}{q(x)} f(x) dx \\ &= E_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right] \end{aligned}$$

Deriving the policy gradient with IS

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$$

a convenient identity

$$p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) = \nabla_{\theta} p_{\theta}(\tau)$$

can we estimate the value of some *new* parameters θ' ?

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

the only bit that depends on θ'

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_{\theta}(\tau)} \left[\frac{\nabla_{\theta'} p_{\theta'}(\tau)}{p_{\theta}(\tau)} r(\tau) \right] = E_{\tau \sim p_{\theta}(\tau)} \left[\frac{\cancel{p_{\theta'}(\tau)}}{\cancel{p_{\theta}(\tau)}} \nabla_{\theta'} \log p_{\theta'}(\tau) r(\tau) \right]$$

now estimate locally, at $\theta = \theta'$: $\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$

The off-policy policy gradient

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

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_{\theta}(\tau)} \left[\frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right] \quad \text{when } \theta \neq \theta'$$

$$\frac{p_{\theta'}(\tau)}{p_{\theta}(\tau)} = \frac{\prod_{t=1}^T \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}$$

$$= E_{\tau \sim p_{\theta}(\tau)} \left[\left(\prod_{t=1}^T \frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} \right) \left(\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right] \text{ what about causality?}$$

$$= E_{\tau \sim p_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \underbrace{\left(\prod_{t'=1}^t \frac{\pi_{\theta'}(\mathbf{a}_{t'} | \mathbf{s}_{t'})}{\pi_{\theta}(\mathbf{a}_{t'} | \mathbf{s}_{t'})} \right)}_{\text{future actions don't affect current weight}} \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \left(\cancel{\prod_{t''=t}^{t'} \frac{\pi_{\theta'}(\mathbf{a}_{t''} | \mathbf{s}_{t''})}{\pi_{\theta}(\mathbf{a}_{t''} | \mathbf{s}_{t''})}} \right) \right]$$

future actions don't affect current weight

if we ignore this, we get
a policy iteration algorithm
(more on this in a later lecture)

A first-order approximation for IS (preview)

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \underbrace{\left(\prod_{t'=1}^t \frac{\pi_{\theta'}(\mathbf{a}_{t'} | \mathbf{s}_{t'})}{\pi_{\theta}(\mathbf{a}_{t'} | \mathbf{s}_{t'})} \right)}_{\text{exponential in } T} \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \right) \right]$$

let's write the objective a bit differently...

on-policy policy gradient: $\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$

$(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \sim \pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t)$

off-policy policy gradient: $\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\pi_{\theta'}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})}{\pi_{\theta}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$

We'll see why this is
reasonable
later in the course!

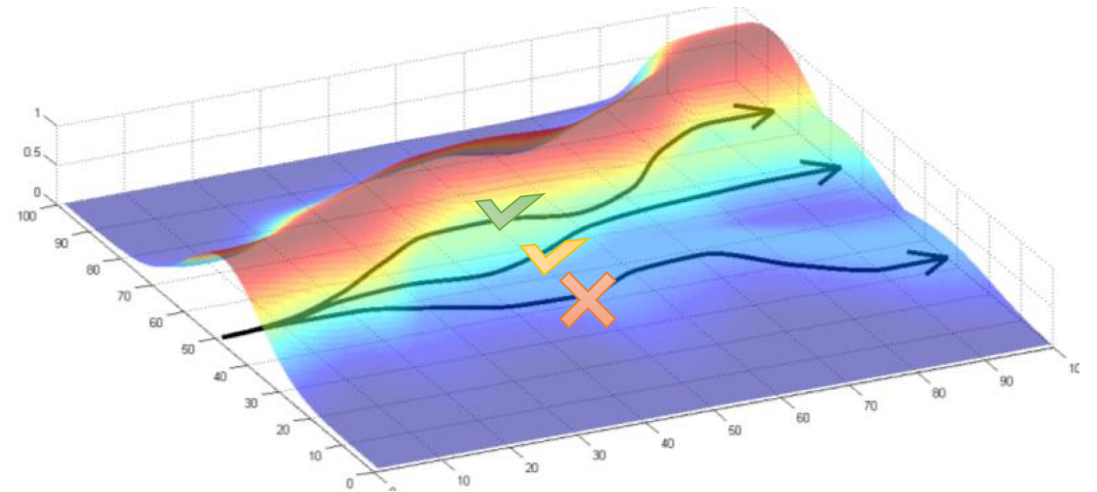
$$= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\cancel{\pi_{\theta'}(\mathbf{s}_{i,t})}}{\cancel{\pi_{\theta}(\mathbf{s}_{i,t})}} \frac{\pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$$

ignore this part

Problem of Importance Sampling

- If optimized J, it will make $\frac{\pi_{\theta}}{\pi_{old}}$ larger and larger. Distribution will significantly change.
 - distribution change too may lead policy network performance much worse
 - policy sampled bad actions, lead to worse policy.
 - training collapsed!

$$J(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \hat{A}_t \right]$$



TRPO

- TRPO: Optimize mean reward on a constraint (use importance sampling)

$$\begin{aligned} & \underset{\theta}{\text{maximize}} && \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] \\ & \text{subject to} && \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]] \leq \delta. \end{aligned}$$

- Intuition: π_{θ} shouldn't change too much in 1 step.
 - equals to change the step length in every step, avoiding training collapse
 - Theoretically, It can be proved that the mean reward function is monotonous
- TRPO uses hard constraints to control the change.
 - monotonous -> local minima
 - complicate math

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) |_{\theta_k} \hat{A}_t.$$

$$\hat{x}_k \approx \hat{H}_k^{-1} \hat{g}_k,$$

$$\theta_{k+1} = \theta_k + \alpha^j \sqrt{\frac{2\delta}{\hat{x}_k^T \hat{H}_k \hat{x}_k}} \hat{x}_k,$$

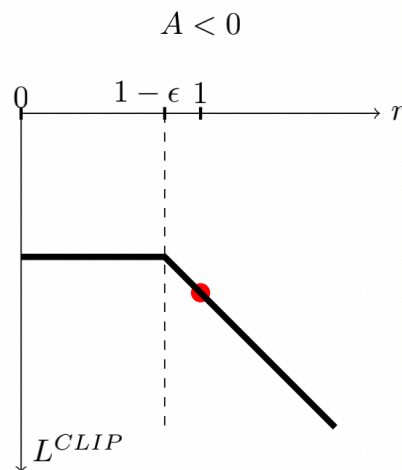
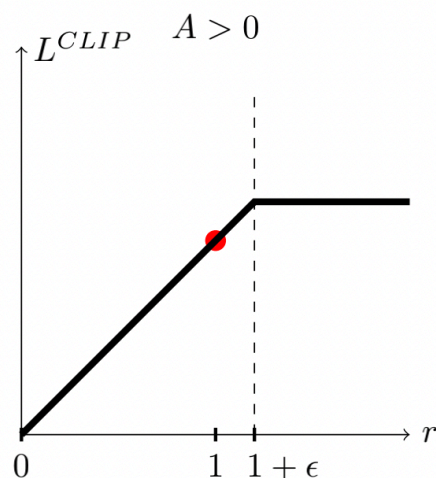
PPO

- Following TRPO's intuition, PPO provides two methods to constraint policy distribution's change using the first-order algorithm
 - Clipped Surrogate Objective
 - Adaptive KL Penalty Coefficient
- Soft constraint
 - Trade-off: reward monotonous and policy randomness
- Other tricks to make training more stable and better sample efficiency
 - Importance sampling
 - GAE

PPO - Clipped Surrogate Objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t [r_t(\theta) \hat{A}_t].$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t [\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t)]$$



PPO - Adaptive KL penalty coefficient

- KL constraint in TRPO is hard to apply
- PPO uses Adaptive KL penalty coefficient to have the same effect
 - can be alternative/addition to Clipped Surrogate Objective
- we can achieve some target value of KL divergence d_{targ}

$$L^{KL\text{PEN}}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)] \right]$$

Compute $d = \hat{\mathbb{E}}_t [\text{KL}[\pi_{\theta_{\text{old}}}(\cdot | s_t), \pi_{\theta}(\cdot | s_t)]]$

- If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$
 - If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$
- During training, Sometimes KL divergence is significantly different from d_{targ} , but can fast adjust.

PPO - Total loss and other tricks

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)],$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

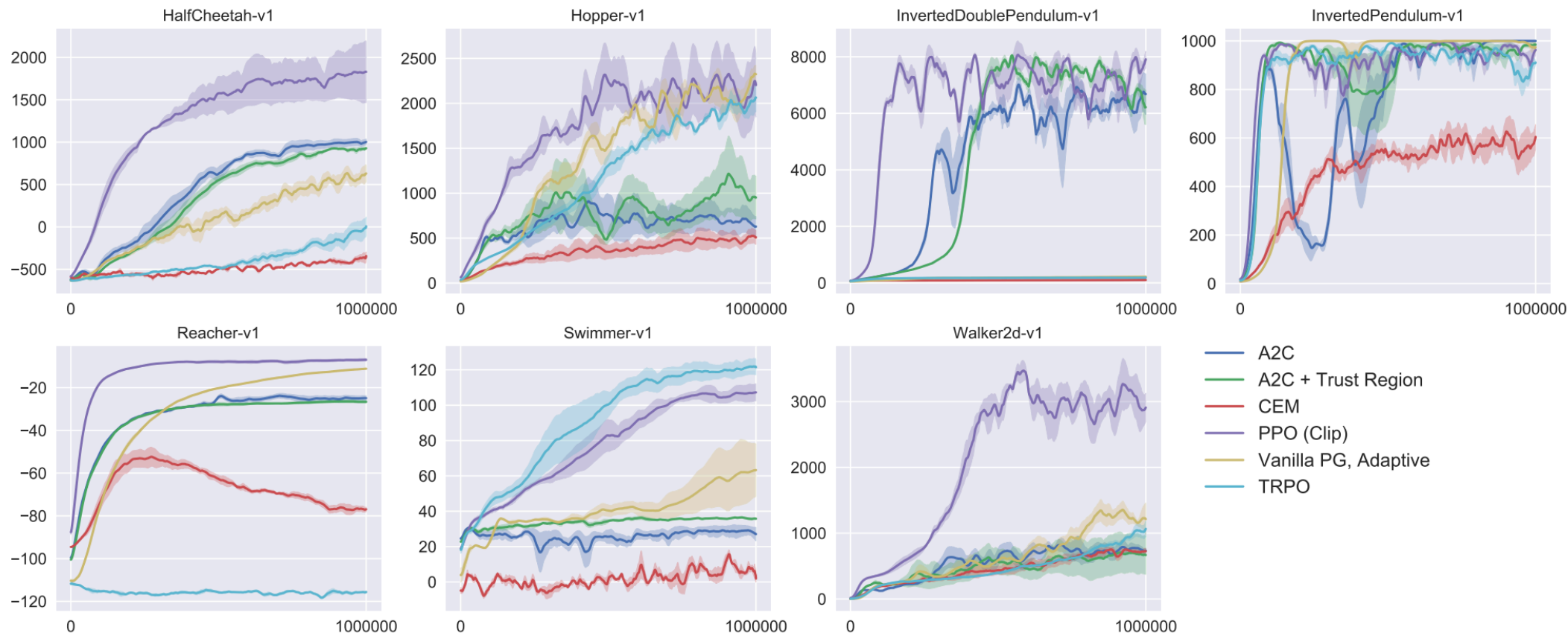
PPO - Experiments

- Exp 1, Clipping VS. Adaptive KL
 - Clipping > Adaptive KL > Fixed KL > benchmark

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	0.82
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

PPO - Experiments

- Exp 2, PPO vs. Other on-policy RL algorithm
 - Sample efficiency
 - Final performance



Implementation Tricks

<https://zhuanlan.zhihu.com/p/512327050>

<https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/>

Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO , Logan Engstrom, ... , MIT, ICLR 2020

PPO - Update tricks

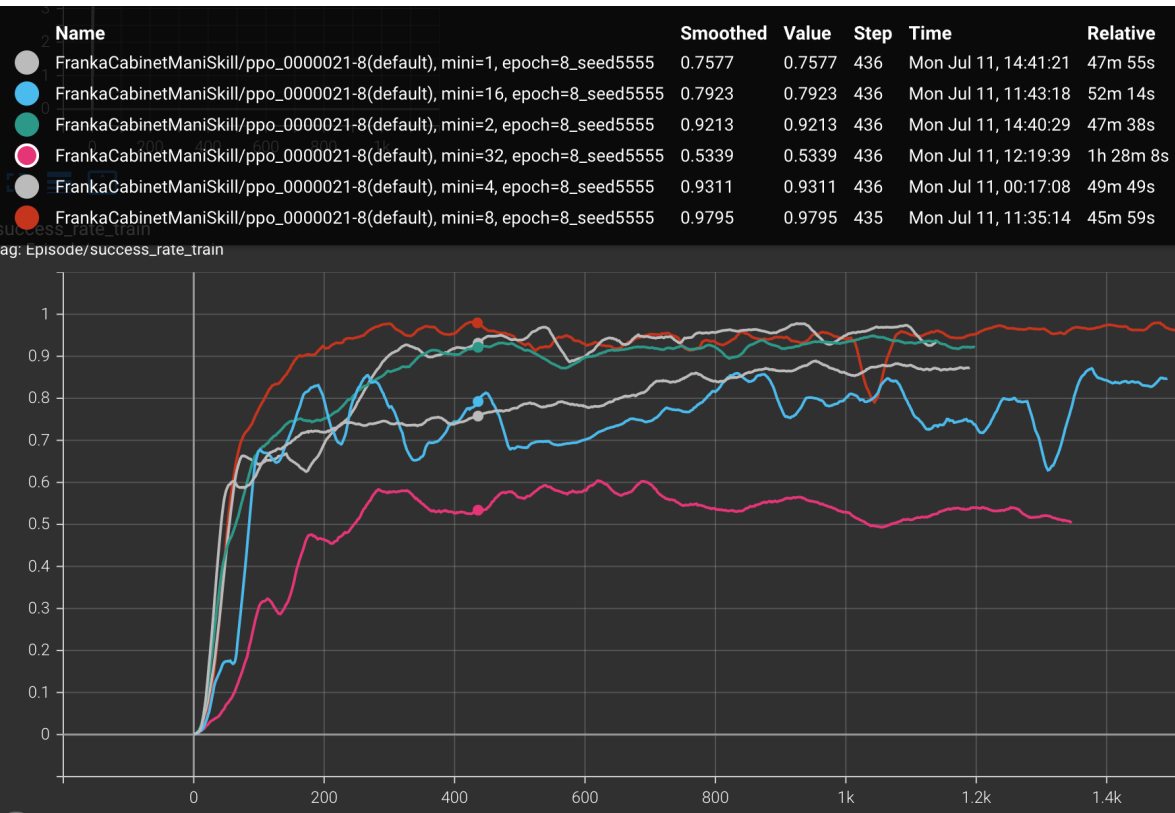
- 1, sample action-state pairs $\{s_t, a_t, s_{t+1}\} \rightarrow B$ from π_θ
2, update $V_\phi, \pi_{\theta'}$ **for K epochs**, each epoch use all of data in B
3, $\theta' \rightarrow \theta$

1. In practice, K shouldn't be too large otherwise the data error would be large.
 - In PPO's original paper, K=15 for RoboschoolHumanoidFlagrun
2. Minibatch: In each epoch, divided buffer B into several minibatches. each batch updates network once. (Like batch gradient decent)
 - Minibatch should be suitable.
 - large: stuck in local minima
 - small: too noisy gradient



PPO - Update tricks

- Our own experiment: state-based close door in 44 different objects.
 - Minibatch size ~ 5500
- RoboschoolHumanoidFlagrun: mini-batch = 4096



Hyperparameter	Value
Horizon (T)	512
Adam stepsize	*
Num. epochs	15
Minibatch size	4096
Discount (γ)	0.99
GAE parameter (λ)	0.95
Number of actors	32 (locomotion), 128 (flagrun)
Log stdev. of action distribution	LinearAnneal(-0.7, -1.6)

PPO -Normalization / scaling

- Advantage normalization: $A_t^{norm} = \frac{A_t - \mu_A}{\sqrt{\sigma_A^2 + \epsilon}}$
 - only used in training time
 - normalize in a minibatch
 - When $V_\varphi(s_t)$ isn't well estimated, it helps A_t centered 0.
- State normalization : $s_t^{norm} = \frac{s_t - \mu_{run}}{\sqrt{\sigma_{run}^2 + \epsilon}}$
 - Need to restore the running mean of each dimension of states and use it in test time.
 - Similar to BatchNorm, it helps with training.
- Reward scaling : $r_t^{scale} = \frac{r_t}{\sqrt{\sigma_{run}^2 + \epsilon}}$
 - scaled reward makes value function $V_\varphi(s_t)$ training easier

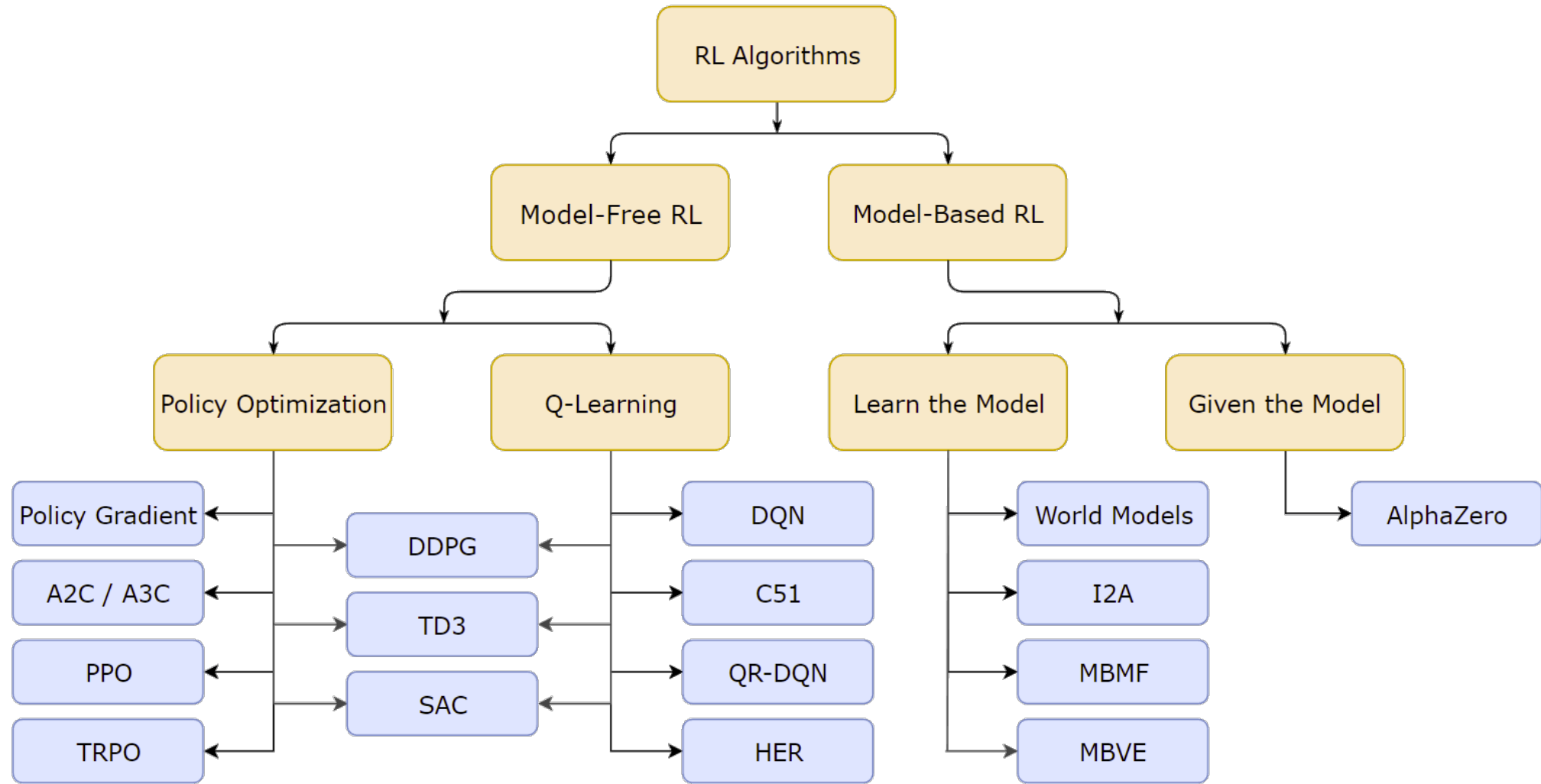
PPO - Initialization / activation

- Orthogonal Initialization
 - 1, Gaussian distribution($\mu = 0$, $\sigma = 1$) initialize W.
 - 2, SVD, $W = U^T D V$, $V \rightarrow W$. W is orthogonal matrices
 - [Engstrom, Ilyas, et al., \(2020\)](#) find orthogonal initialization to outperform the default Xavier initialization.
 - [Andrychowicz, et al. \(2021\)](#) find centering the action distribution around 0 (i.e., initialize the policy output layer weights with 0.01”) to be beneficial
- Tanh activation
 - use Tanh activation instead of ReLU (PPO original paper)
 - only 2 or 3 hidden layer MLP, gradient vanishing may not have significant damage.

Pros and Cons

- Pros:
 - better sample efficiency
 - Importance sampling, minibatch
 - more stable training and better performance
 - Clipped Surrogate Objective, Adaptive KL, other implemental tricks
 - Simpler math than TRPO
- Cons:
 - Still poor sample efficiency compared with off-policy RL algorithm
 - Need to train in simulator, has domain gap to real-world environment
 - many task-specific tricks
 - Carefully choose hyperparameters

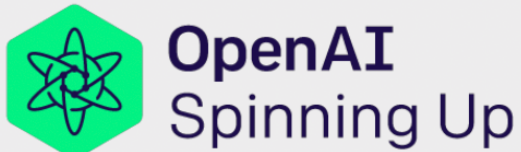
A Taxonomy of RL Algorithms



Further Reading and Practice

OpenAI Tutorial to RL

<https://spinningup.openai.com/en/latest/user/introduction.html>



ALGORITHMS DOCS

Vanilla Policy Gradient

Trust Region Policy Optimization

Proximal Policy Optimization

Deep Deterministic Policy Gradient

Twin Delayed DDPG

Soft Actor-Critic



Introduction to Embodied AI

Next week:
Lecture 11, Locomotion

