

Advanced Policy Gradients

Deep Reinforcement Learning

Mohammad Hossein Rohban, Ph.D.

Courtesy: Some slides are adopted from CS 285 Berkeley, and CS 234 Stanford, and Pieter Abbeel's compact series on RL.

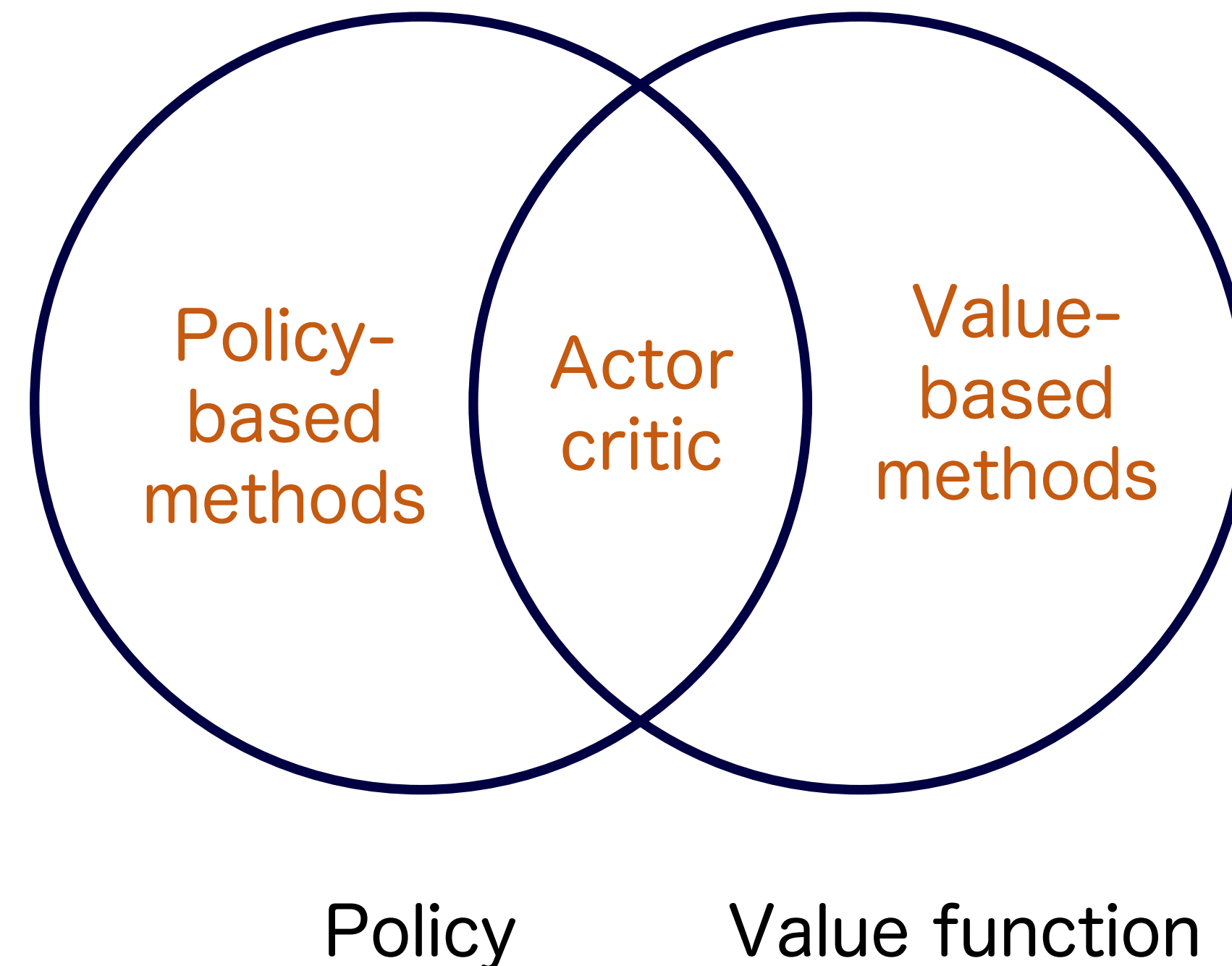


RIML Lab, CE Department, Sharif
University of Technology

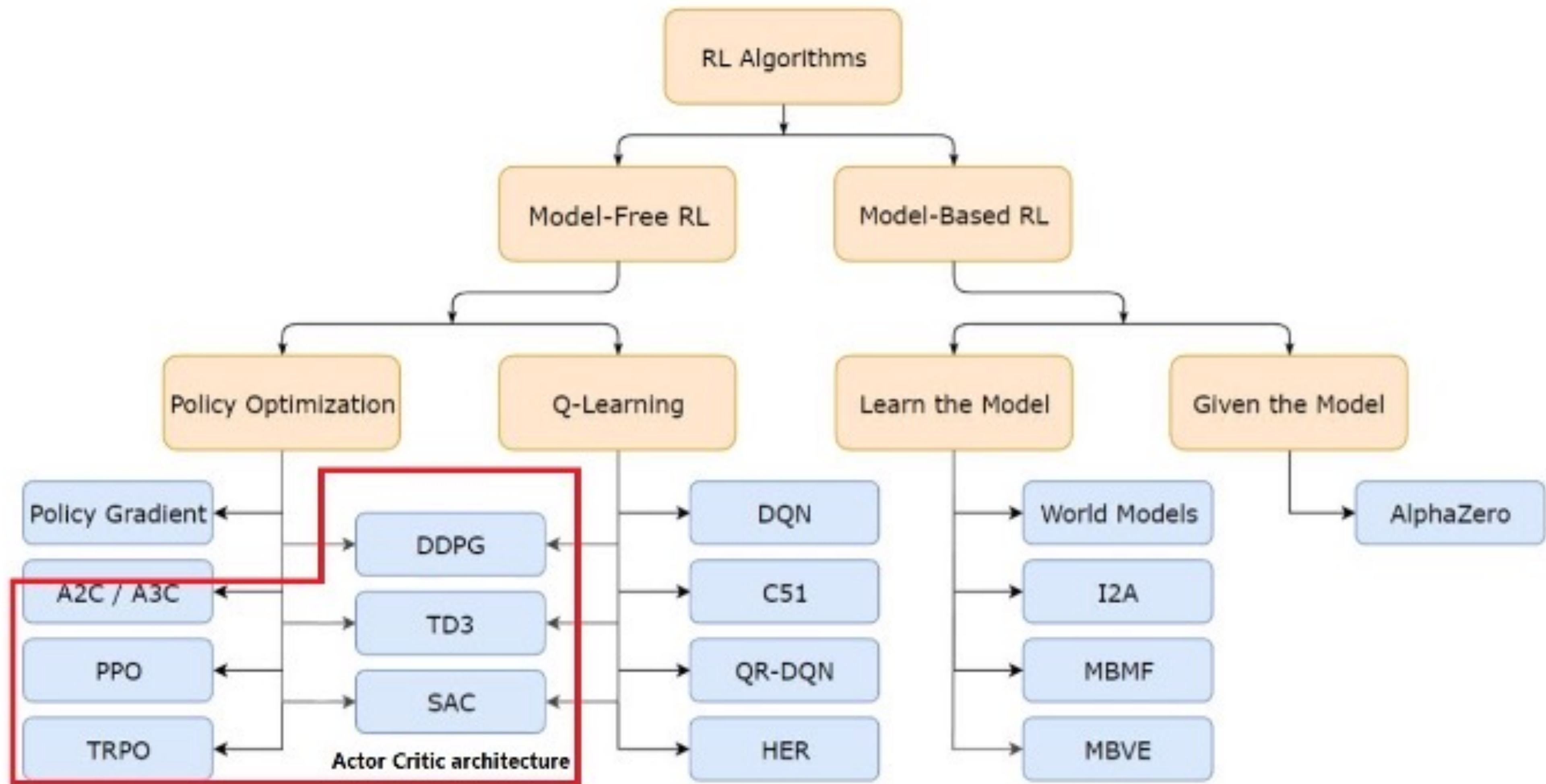
Spring 2026

Model-Free RL

- ◆ Value-based methods
 - ☑ Learnt value function
 - ☑ Implicit policy
- ◆ Policy-based methods
 - ☑ No value function
 - ☑ Learnt policy
- ◆ Actor-critic methods
 - ☑ Learnt value function
 - ☑ Learnt policy



Overview of Modern RL Methods



Policy Gradient Intuition

$$\rightarrow J(\theta) = \mathbb{E}_{a_t \sim \pi_\theta(\cdot|s_t)} \left[\sum_{t=1}^T r(s_t, a_t) \right]$$

$\max J(\theta)$

log trick

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

$$\nabla_\theta J$$

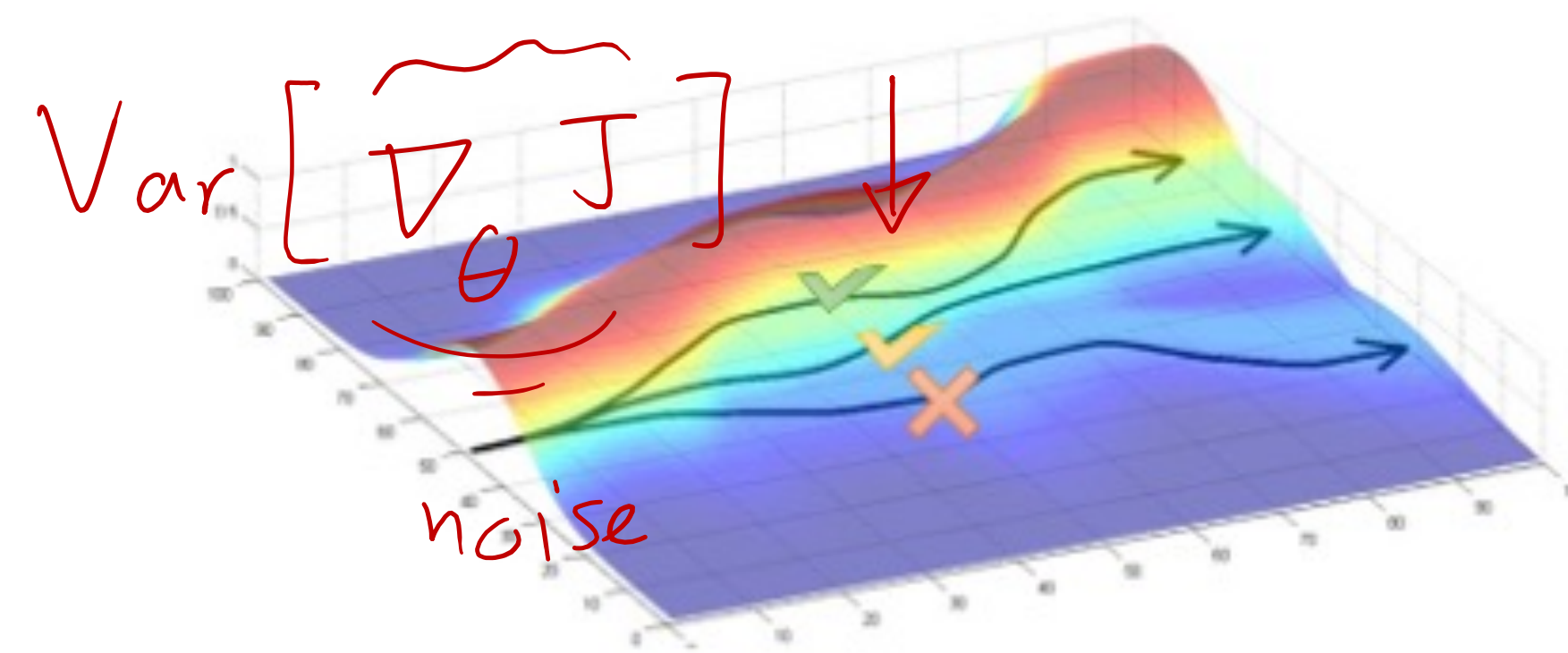
N trajectory $\sim \pi_\theta$

return

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \underbrace{\nabla_\theta \log \pi_\theta(\tau_i)}_{\sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})} r(\tau_i)$$

maximum likelihood: $\nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_\theta \log \pi_\theta(\tau_i)$

- ♦ Good stuff is made more likely
- ♦ Bad stuff is made less likely
- ♦ Simply formalizes the notion of “trial and error”!

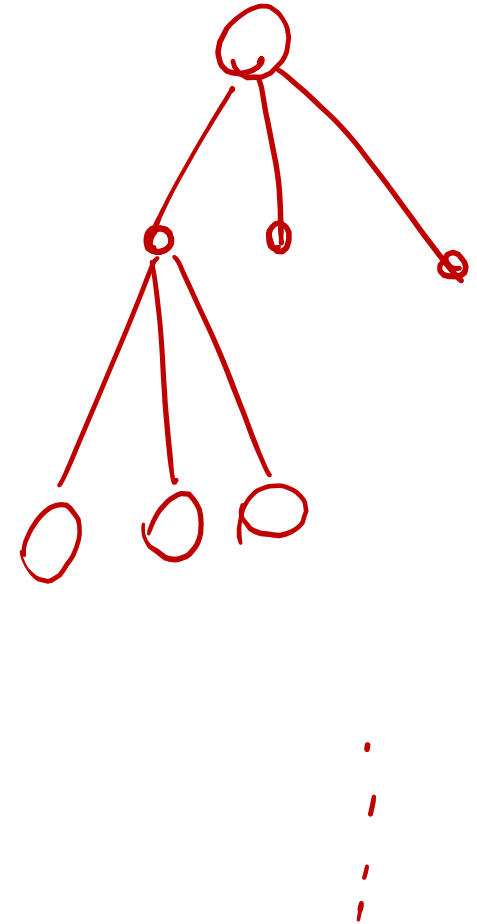


Bias and Variance of Policy Gradient

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(\tau_i) \underline{r(\tau_i)}$$

Stable

The main source of high variance



✦ Unbiased estimation:

$$E \left[\frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log \pi_{\theta}(\tau_i) r(\tau_i) \right] = \nabla_{\theta} J(\theta)$$

☑ But suffers from **high variance**!

Reducing Variance

- ◆ Everything in the gradient whose **expected is zero** could be removed, without affecting the optimization, but could lead to **lower gradient variance!**

- ◆ Causality trick

- ◆ Discount factor

- ◆ Baseline

- ◆ Actor-critic

→ *Var. reduction*

- ◆ Optimization techniques:

- ☒ Natural gradient

- ☒ Trust region

$$\frac{1}{T} \sum_{t=1}^T \nabla \log \pi(a_{i,t} | s_{i,t})$$

$$\sum_{t'=1}^T r(s_{i,t'}, a_{i,t'})$$

unbiased

$\sum_{t=1}^T \nabla \log \pi(a_{i,t} | s_{i,t})$

Sum together these rewards

sum together these rewards

not these!

$\sum_{t'=t}^T r(s_{i,t'}, a_{i,t'})$

Reducing Variance: Discount Factor

2

$\gamma = 0.9$ $\rightarrow t=10$

start @ 0

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

Not the same!

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

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

Reducing Variance: Baselines ✓

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \nabla_{\theta} \log p_{\theta}(\tau) [r(\tau) - b]$$

Const(τ)

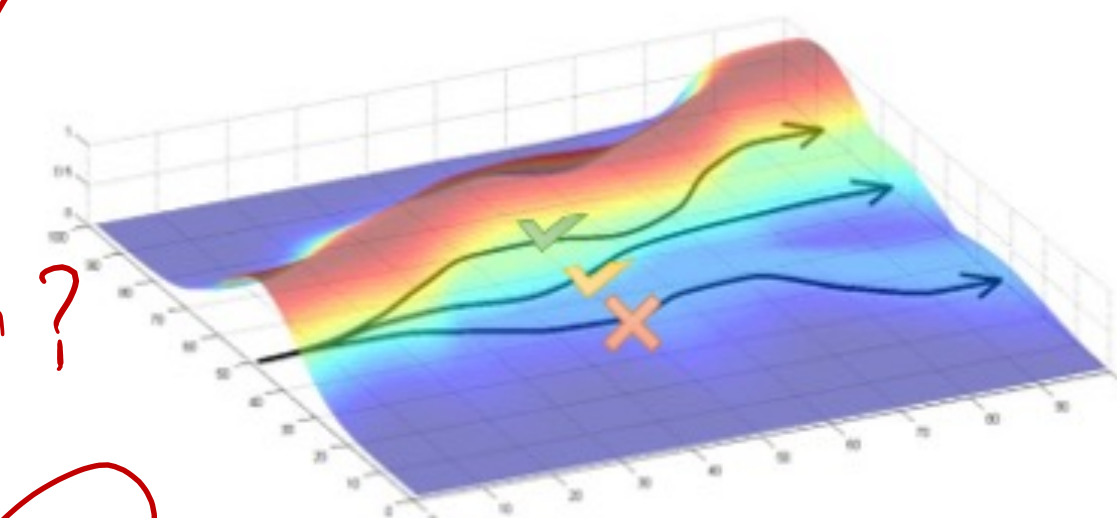
A convenient identity
 $p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) = \nabla_{\theta} p_{\theta}(\tau)$

$N \rightarrow \infty$

$$b = \frac{1}{N} \sum_{i=1}^N r(\tau_i)$$

$$\nabla \log \pi_{\theta} r(\tau) - b \nabla \log \pi_{\theta}$$

is it zero ✓
 in expectation?



$$\rightarrow E[\nabla_{\theta} \log p_{\theta}(\tau) b] = \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) b d\tau = \int \nabla_{\theta} p_{\theta}(\tau) b d\tau = b \nabla_{\theta} \int p_{\theta}(\tau) d\tau = b \nabla_{\theta} 1 = 0$$

$\hookrightarrow p_{\theta}(\tau) = p(s_1) \pi_{\theta}(a_1 | s_1) p(s_2 | s_1, a_1) \dots$

Subtracting a baseline is unbiased expectation!

Average reward is not the best baseline, but it's pretty good!

Analyzing Variance

$$\tau_1 \dots \tau_N \sim \pi_\theta \longrightarrow r(\tau_1) \dots r(\tau_N) \longrightarrow \text{Avg.} \longrightarrow b$$

$$\text{Var}[x] = E[x^2] - E^2[x]$$

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

$$\text{Var} = E_{\tau \sim p_\theta(\tau)} \left[\underbrace{(\nabla_\theta \log p_\theta(\tau) (r(\tau) - b))^2}_{g(\tau)} \right] - \underbrace{E_{\tau \sim p_\theta(\tau)}^2 [\nabla_\theta \log p_\theta(\tau) (r(\tau) - b)]}_{\text{this bit is just } E_{\tau \sim p_\theta(\tau)} [\nabla_\theta \log p_\theta(\tau) r(\tau)] \text{ baselines are unbiased in expectation}}$$

$$\begin{aligned} \frac{d\text{Var}}{db} &= \frac{d}{db} E[g(\tau)^2 (r(\tau) - b)^2] = \frac{d}{db} (E[g(\tau)^2 r(\tau)^2] - 2E[g(\tau)^2 r(\tau) b] + b^2 E[g(\tau)^2]) \\ &= -2E[g(\tau)^2 r(\tau)] + 2bE[g(\tau)^2] = 0 \end{aligned}$$

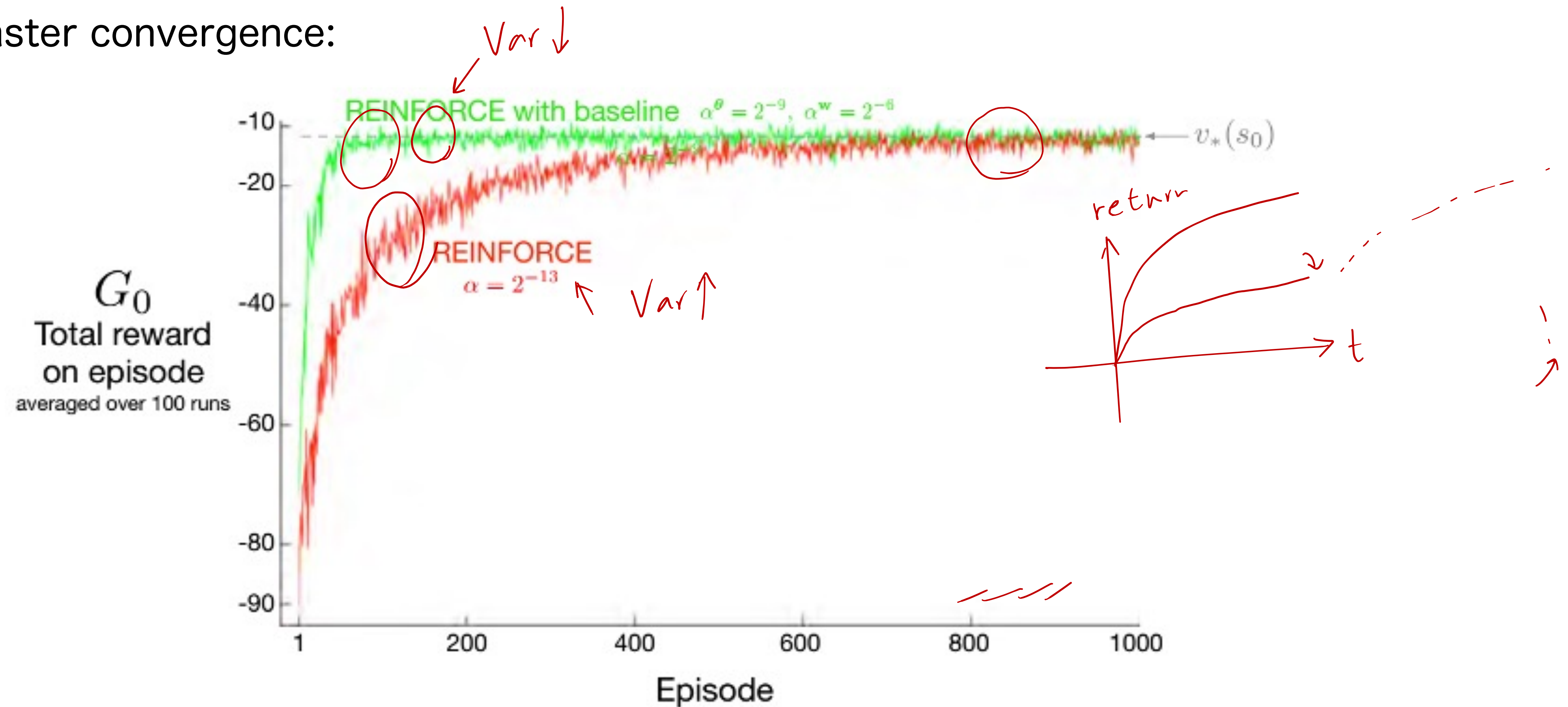
$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]}$$

weighted avg. of $r(\tau)$
This is just expected reward, but weighted by gradient magnitudes!

$$\frac{1}{N} \sum_{i=1}^N r(\tau_i) \approx E(r(\tau))$$

Reducing Variance: Baselines

Faster convergence:



Reducing Variance: Review

- ◆ Exploiting causality ✓
 - ☑ Future doesn't affect the past
- ◆ Discount factor ✓
 - ☑ Two different version
- ◆ Baselines ✓
 - ☑ Analyzing variance for deriving optimal baselines
- ◆ Now: Introducing actor-critic methods!

Policy Gradients so Far

*PPO
SAC*

*on-policy
off-policy*

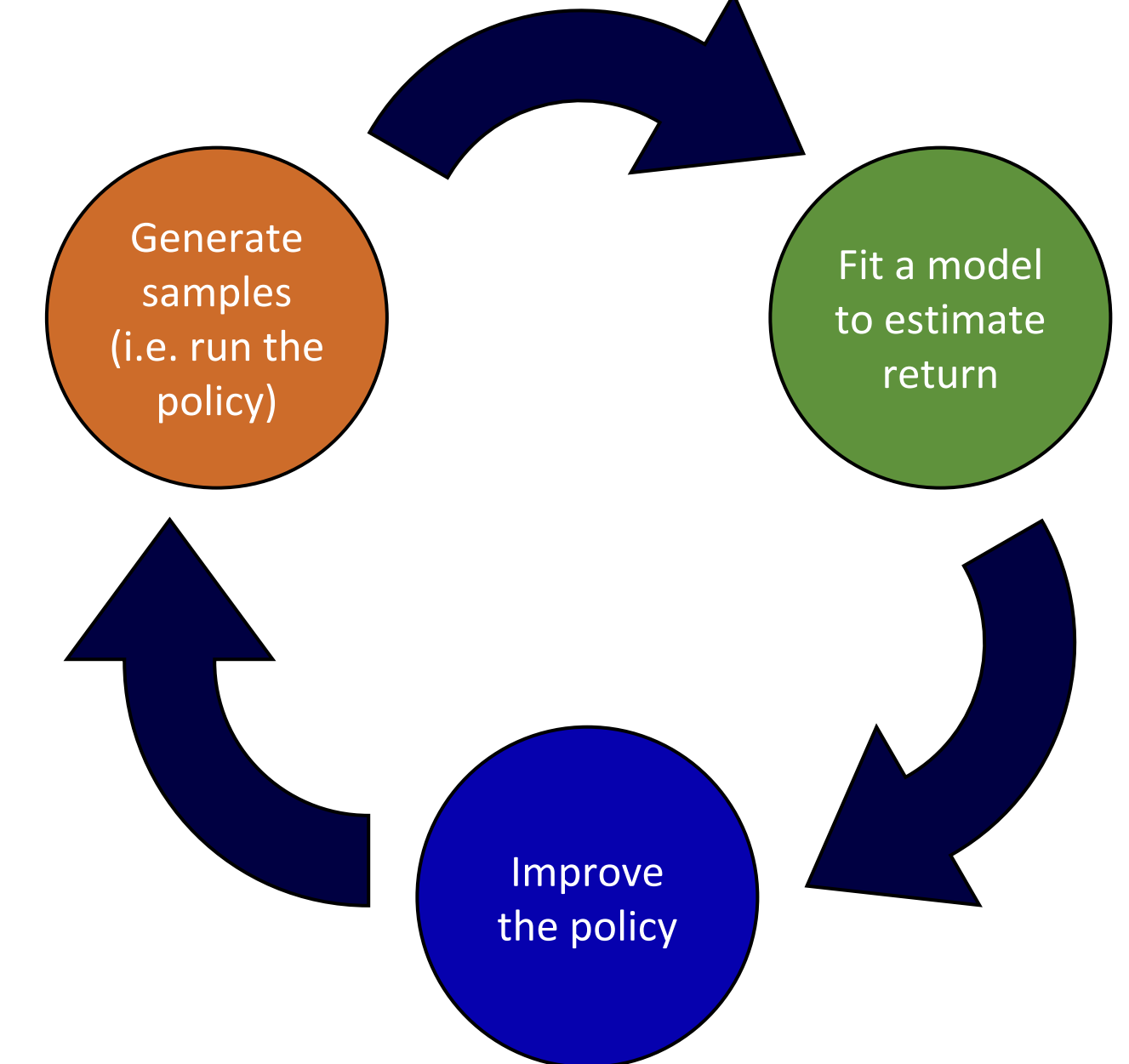
Algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)

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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$$\hat{Q}^\pi(\mathbf{x}_t, \mathbf{u}_t) = \sum_{t'=t}^T r(\mathbf{x}_{t'}, \mathbf{u}_{t'})$$



$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

TD or Approx. Critic

Actor

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \underbrace{\hat{Q}_{i,t}^\pi}_{\text{reward to go}}$$

$\sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$

Q_ϕ

Actor-Critic

Improving Estimation of Reward to Go

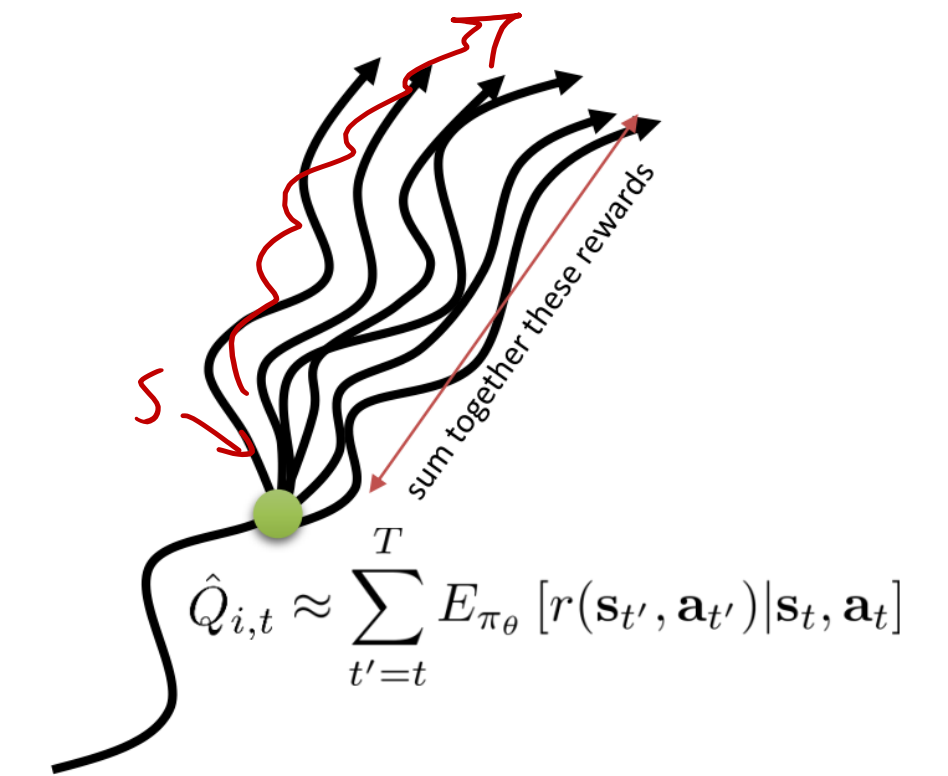
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \underbrace{\left(\sum_{t'=1}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)}_{\text{reward to go}}$$

$\hat{Q}_{i,t}$: estimate of expected reward if we take action $a_{i,t}$ in state $s_{i,t}$

✓ How to make a better estimate?

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_{\theta}} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t] : \text{true expected reward-to-go}$$

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



Improving Estimation of Reward to Go (cont.)

✓ Further improvement: Adding a baseline!

$$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t] : \text{true expected reward-to-go}$$

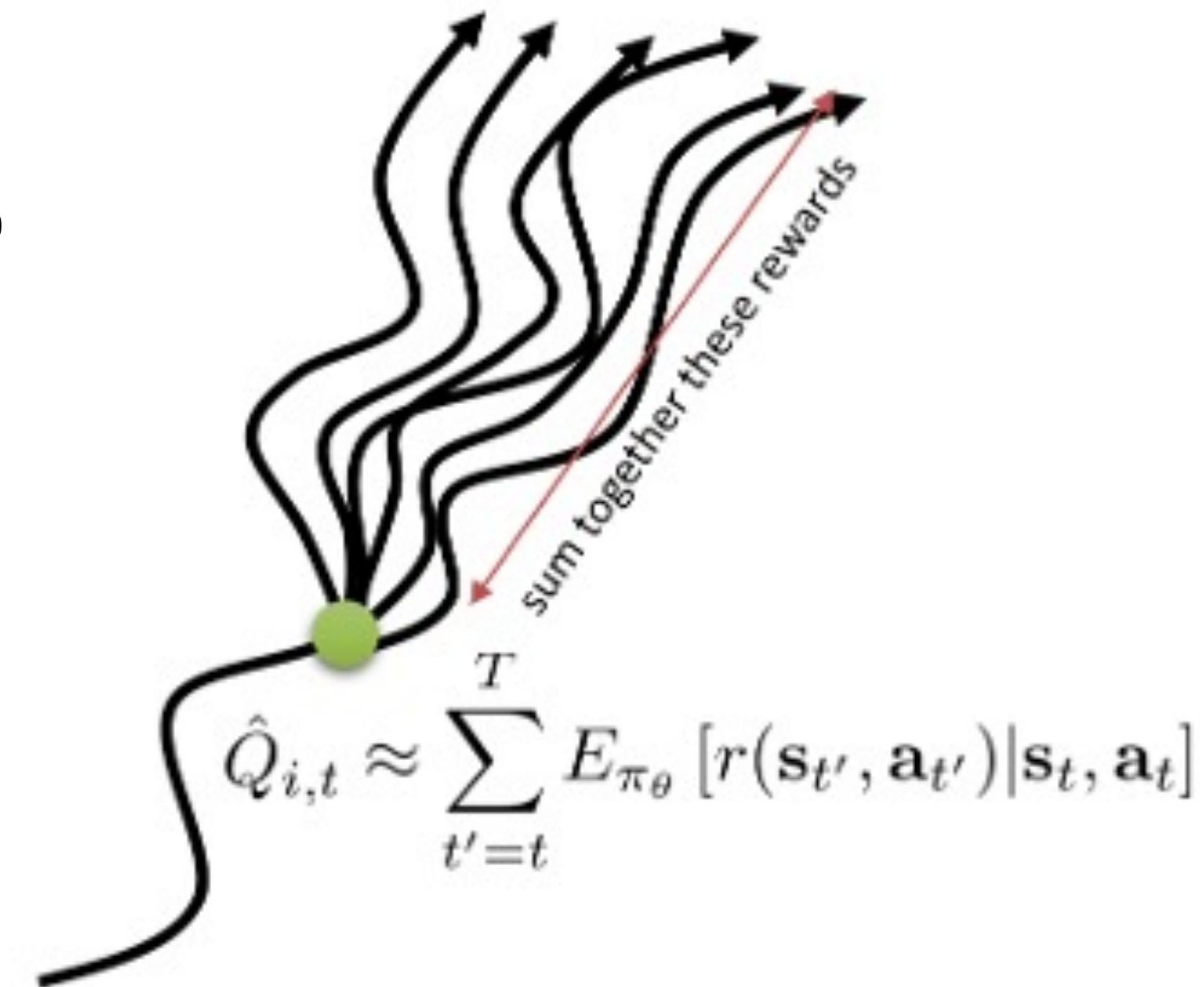
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) (Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) - \cancel{b_t})$$

$V(s_t)$

Is it similar to a known concept?

$$b_t = \frac{1}{N} \sum_i Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

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



Advantage Value

$$\nabla_{\theta} J = \frac{1}{N} \sum_{i=1}^N \sum_t \nabla_{\theta} \log \pi(a_{t,i} | s_{t,i}) [\hat{Q}(s_{t,i}, a_{t,i}) - \hat{V}(s_{t,i})]$$

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_{\theta}}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]: \text{total reward from taking } \mathbf{a}_t \text{ in } \mathbf{s}_t$$

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

$$A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi}(\mathbf{s}_t): \text{how much better } \mathbf{a}_t \text{ is}$$

Advantage
of $a_{t,i}$

=

opportunity

→ for policy
Improvement

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) A^{\pi}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

The better this estimate, the lower the variance

Advantage Value Approximation

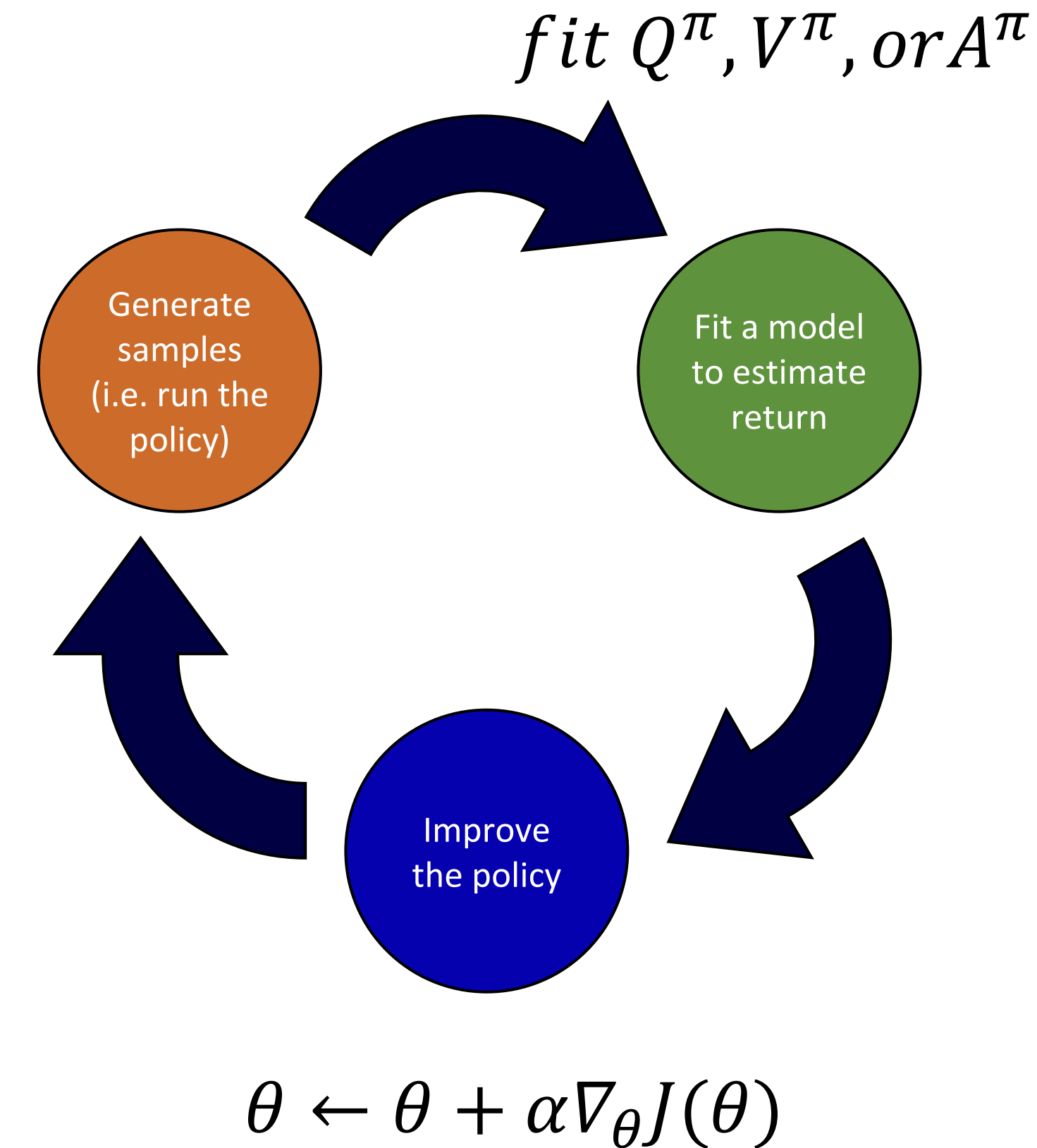
$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$$

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

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

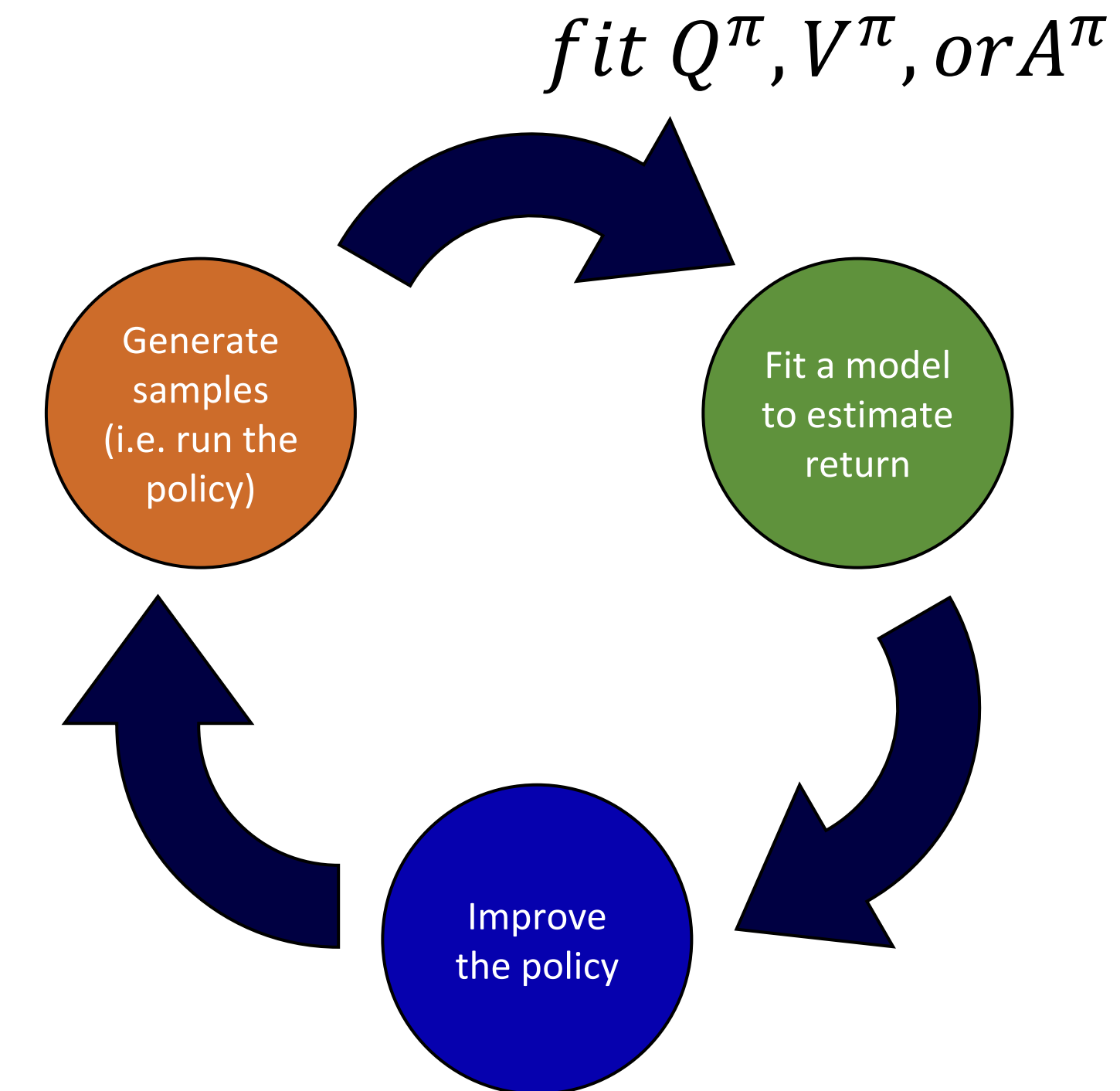
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

$$\begin{aligned} Q^\pi(\mathbf{s}_t, \mathbf{a}_t) &= \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t] \\ &= r(\mathbf{s}_t, \mathbf{a}_t) + \underbrace{\sum_{t'=t+1}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]}_{\approx V^\pi(\mathbf{s}_{t+1})} \end{aligned}$$

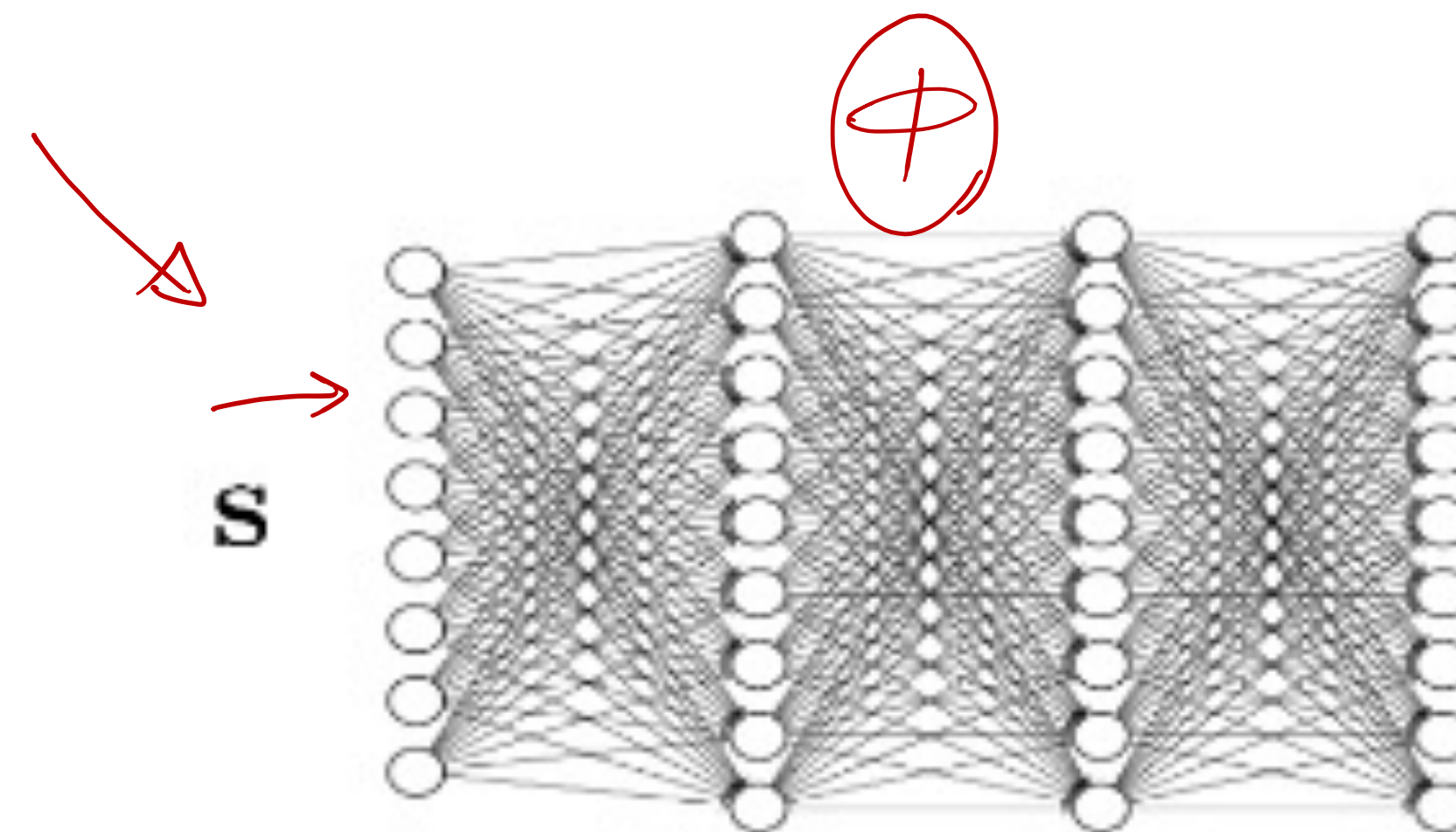


Advantage Value Approximation (cont.)

$$\begin{aligned} \spadesuit Q^\pi(\mathbf{s}_t, \mathbf{a}_t) &\approx \underbrace{r(\mathbf{s}_t, \mathbf{a}_t)} + \underbrace{V^\pi(\mathbf{s}_{t+1})} \\ \spadesuit A^\pi(\mathbf{s}_t, \mathbf{a}_t) &\approx \underbrace{r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1})}_{Q^\pi(\mathbf{s}_t, \mathbf{a}_t)} - V^\pi(\mathbf{s}_t) \end{aligned}$$



◆ Let's just fit $V^\pi(s)$!



$\hat{V}^\pi(\mathbf{s})$ $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
parameters ϕ

Policy Evaluation

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

$$J(\theta) = E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)}[V^\pi(\mathbf{s}_1)]$$

How can perform policy evaluation?

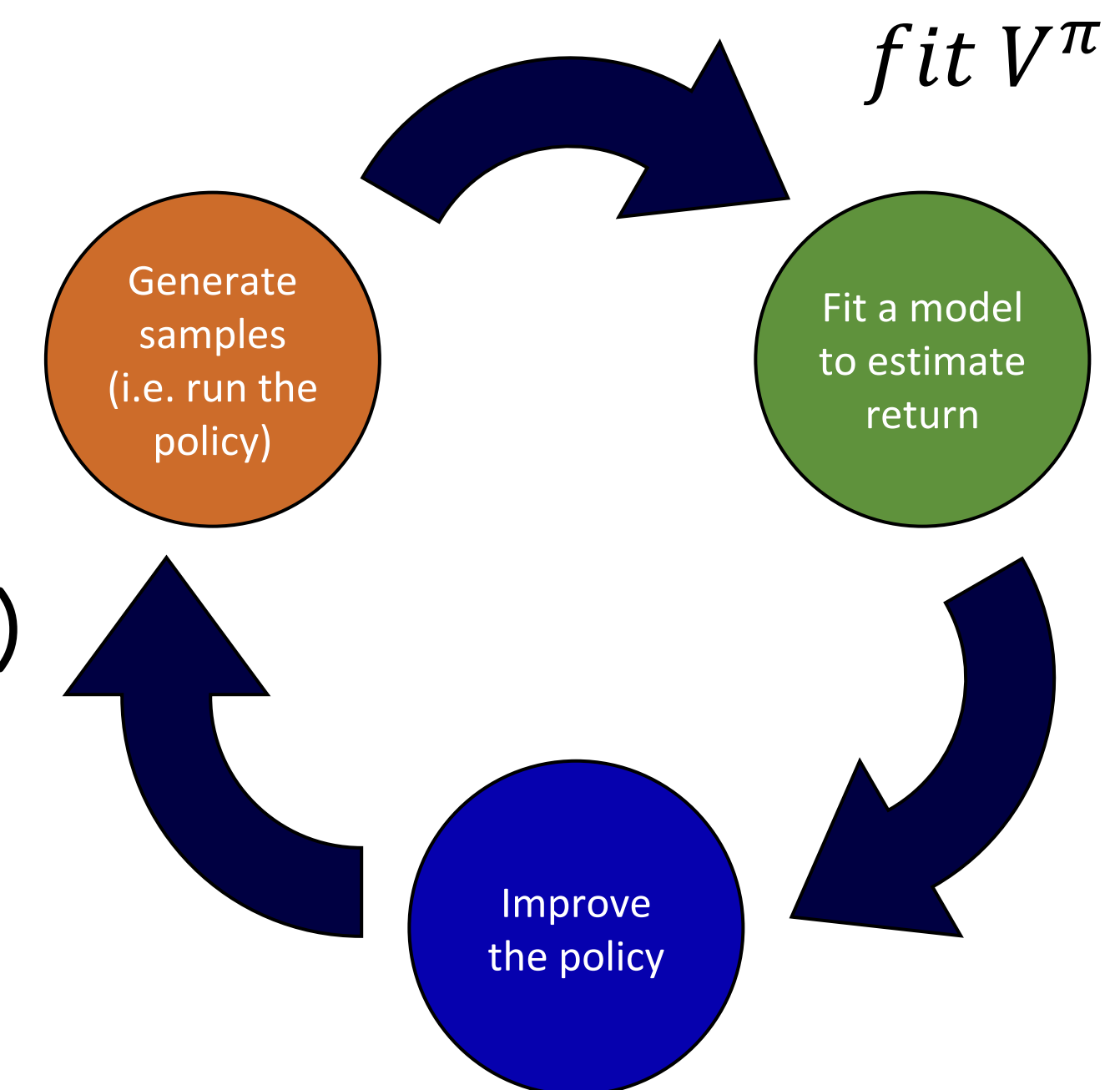
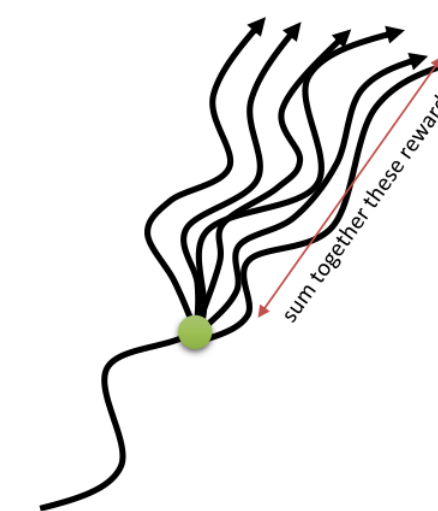
Monte Carlo policy evaluation (this is what policy gradient does)

$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

$$V^\pi(\mathbf{s}_t) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

Monte-Carlo Rollout

(requires us to reset the simulator)



$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Policy Evaluation (cont.)

♦ Monte Carlo estimation with function approximator:

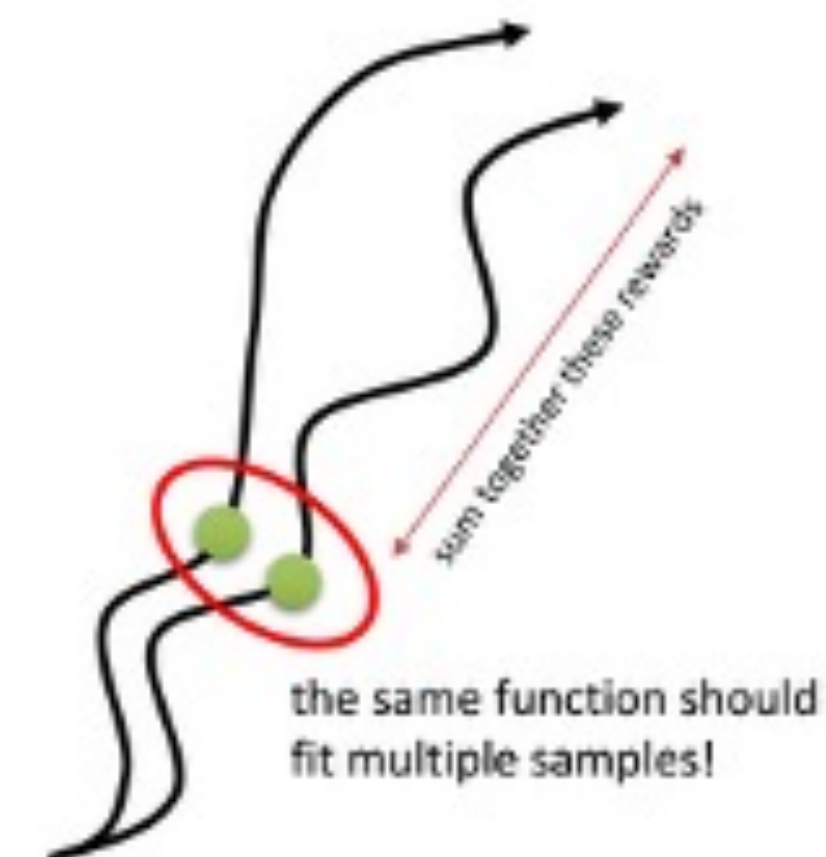
$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

not as good as this: $V^\pi(\mathbf{s}_t) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$

but still pretty good!

training data: $\left\{ \left(\underset{=}{\mathbf{s}_{i,t}}, \underbrace{\sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})}_{y_{i,t}} \right) \right\}$ Var target ↑ VS.

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



Policy Evaluation (cont.)

◆ How to make a better estimate?

ideal target:

$$\underbrace{y_{i,t}} = \sum_{t'=t}^T E_{\pi_{\theta}}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{i,t}] \approx \underbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} + \underbrace{V^{\pi}(\mathbf{s}_{i,t+1})}_{\text{directly use previous fitted value function!}} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \underbrace{\hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+1})}$$

Monte Carlo target:

$$y_{i,t} = \sum_{t'=t}^T r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$$

✓ S.

Policy Evaluation (cont.)

◆ Bootstrap Estimation with Function Approximator

ideal target:

$$y_{i,t} = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{i,t}] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$$

training data: $\left\{ \left(\mathbf{s}_{i,t}, \underbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})}_{y_{i,t}} \right) \right\}$

supervised regression:

$$\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}$$

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

$b = V(s_t) \rightarrow \text{depends on } \tau$

$\tau \rightarrow \theta$

π

$$\hat{A}(s, a) = r(s, a) + \hat{V}_\phi(s') - \hat{V}_\phi(s)$$

$$\underline{\theta} \leftarrow \nabla_{\underline{\theta}} J = \frac{1}{N} \sum_i \sum_t (\nabla \log \pi) \hat{A}(s_{t,i}, a_{t,i})$$

Batch Actor-Critic Algorithm

batch actor-critic algorithm:

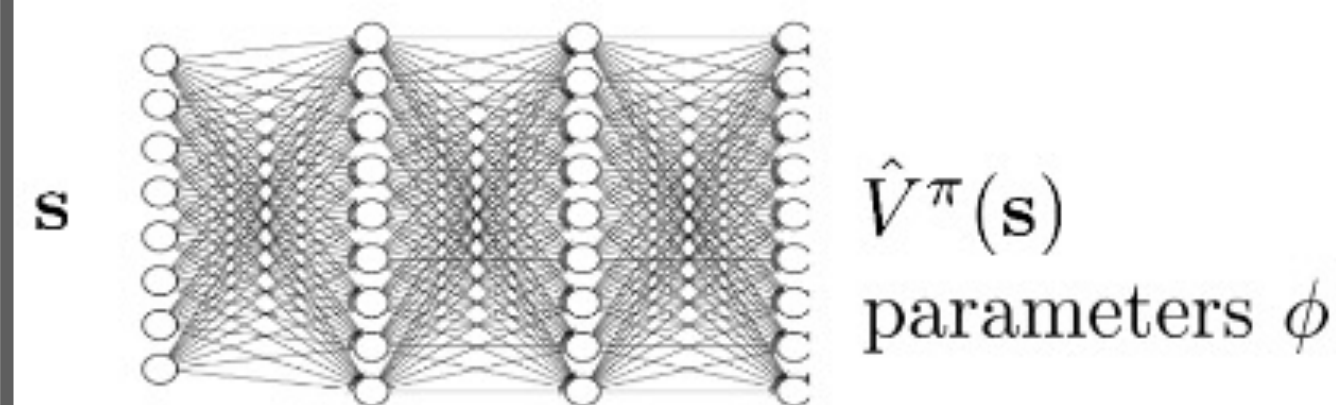
1. sample $\{s_i, a_i\}$ from $\pi_\theta(a|s)$ (run it on the robot)

2. fit $\hat{V}_\phi^\pi(s)$ to sampled reward sums *Monte-Carlo Target*

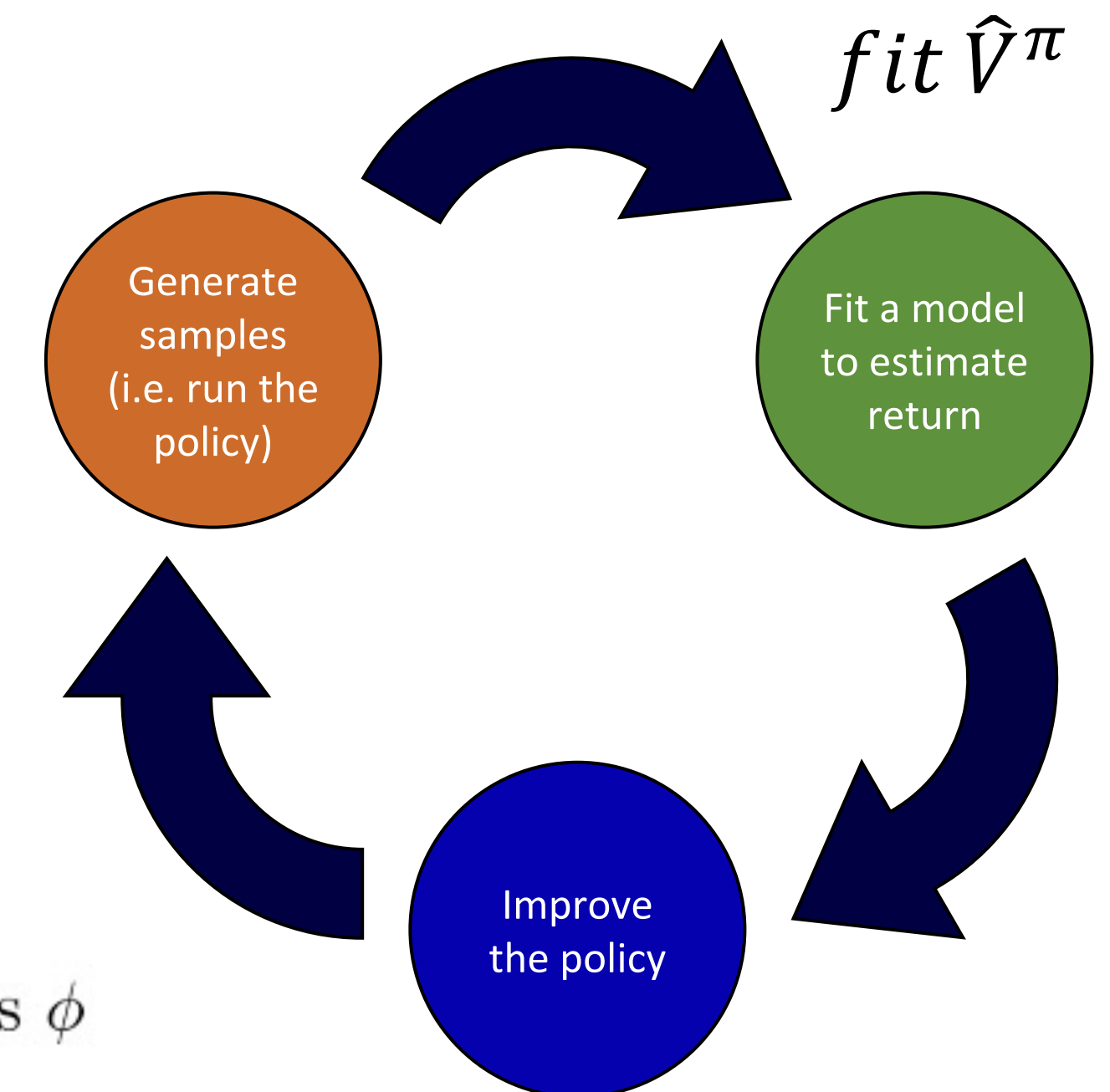
3. Evaluate *adv.*

4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(a_i|s_i) \hat{A}^\pi(s_i, a_i)$

5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



$$V^\pi(s_t) = \sum_{t'=t}^T E_{\pi_\theta}[r(s_{t'}, a_{t'}) | s_t]$$



$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Actor-Critic Algorithm: Architecture Design

batch actor-critic algorithm:

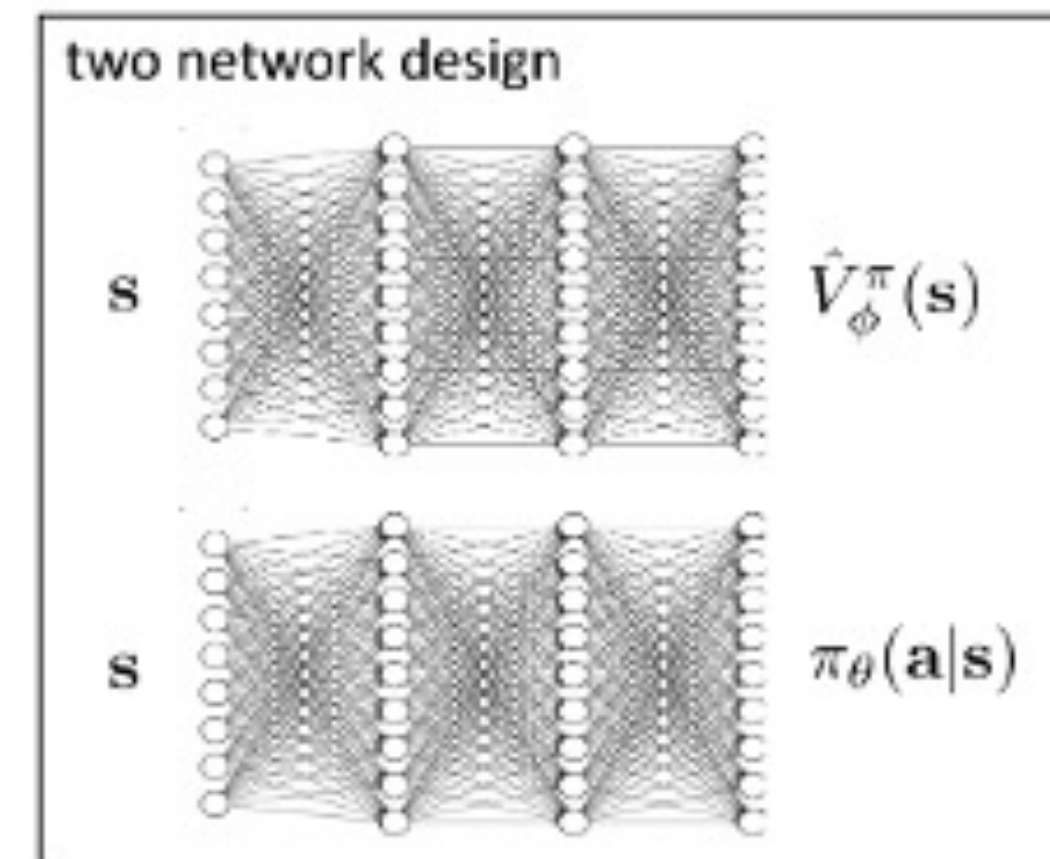
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)

2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums

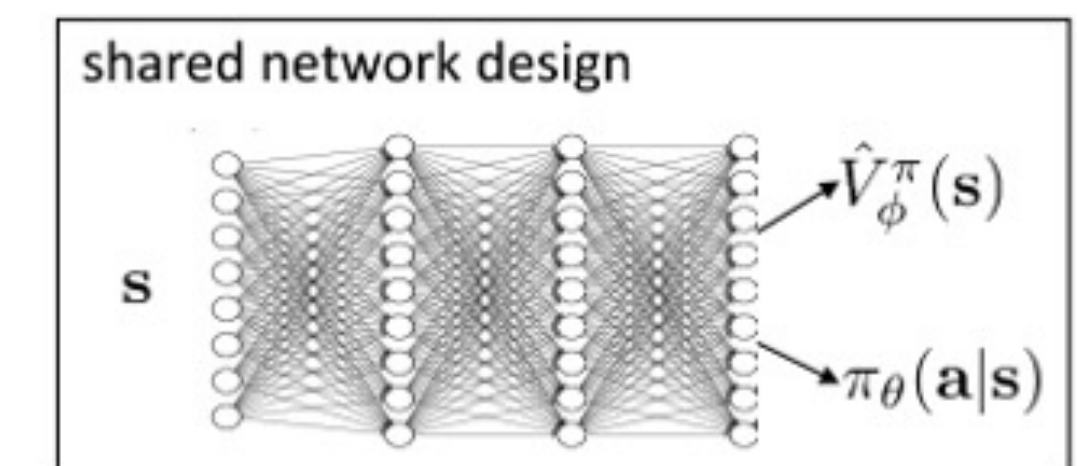
3. Evaluate

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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



+simple & stable
- no shared features between actor and critic



Actor-Critic Algorithm: Architecture Design

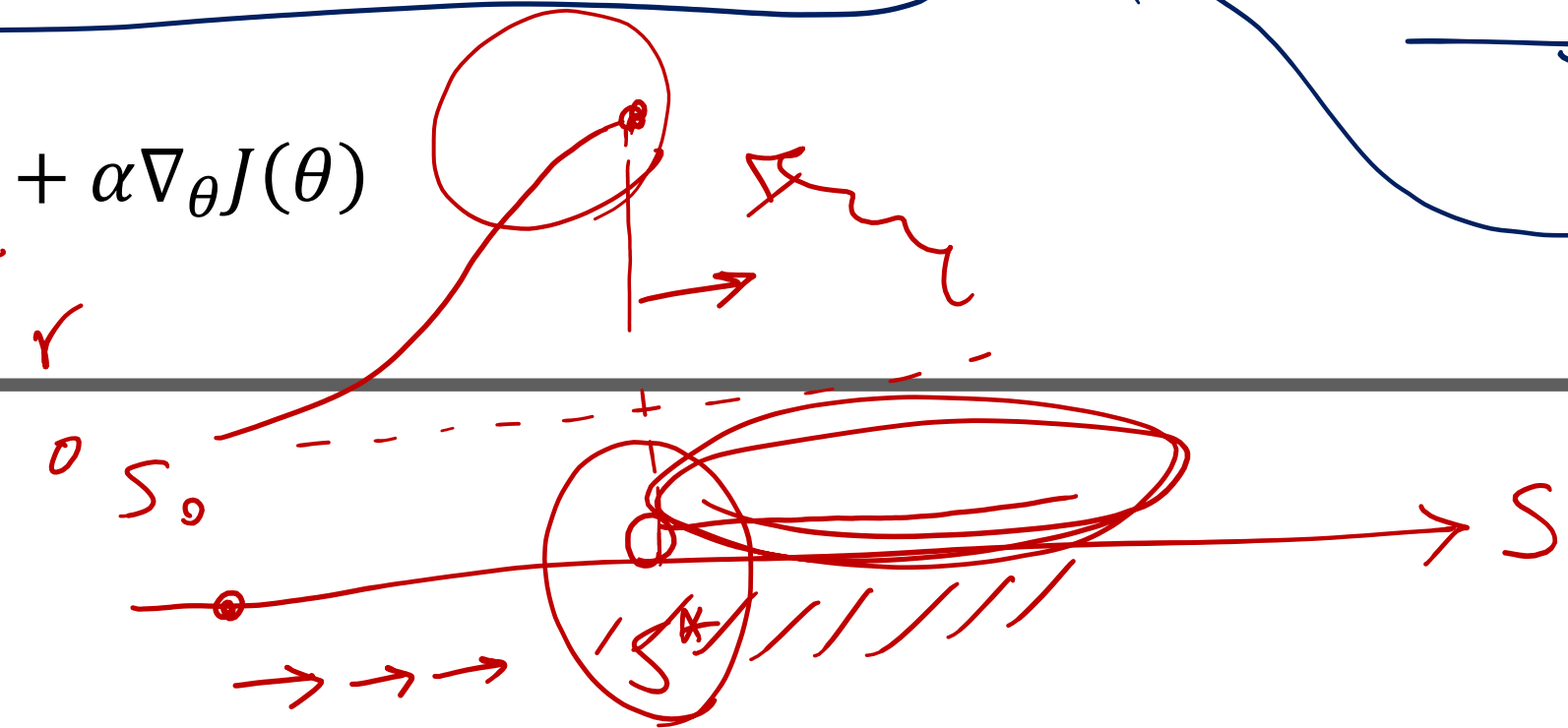
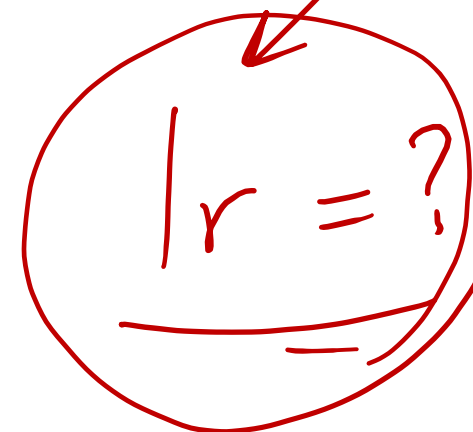
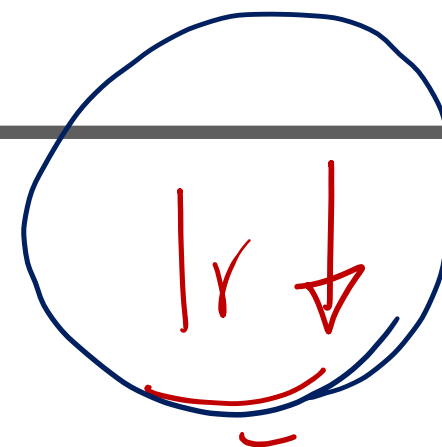
batch actor-critic algorithm:

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

online actor-critic algorithm:

- k times
- $m \cdot k$
1. take action $\underline{a} \sim \pi_\theta(\underline{a}|s)$, get (s, a, s', r)
 2. update $\hat{V}_\phi^\pi$ using target $\underline{r} + \gamma \hat{V}_\phi^\pi(s')$
 3. Evaluate adv.
 4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\underline{a}|s) \hat{A}^\pi(s, a)$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
- reusable?
- $O(m \cdot k)$
- Computational Cost?

Slow



Proximal Policy Optimization

2017

$$\pi_{\theta} \rightarrow \pi_{\theta'}$$

- ♦ We don't want the new policy to change a lot in an iteration. Why?

$$D_{KL}(\pi_{\theta'}(\cdot|s) || \pi_{\theta}(\cdot|s)) \leq \epsilon$$

enforce

- ♦ What is the effect of the constraint?

- ♦ Recall KL-Divergence:

$$D_{KL}(P || q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

$$\Rightarrow D_{KL}(\pi_{\theta'}(\cdot|s) || \pi_{\theta}(\cdot|s)) = \sum_a \pi_{\theta'}(a|s) \log \frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}$$

- ♦ We are effectively constraining the ratio

$$\frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}$$

Proximal Policy Optimization

Let's design a simpler objective that directly constrains

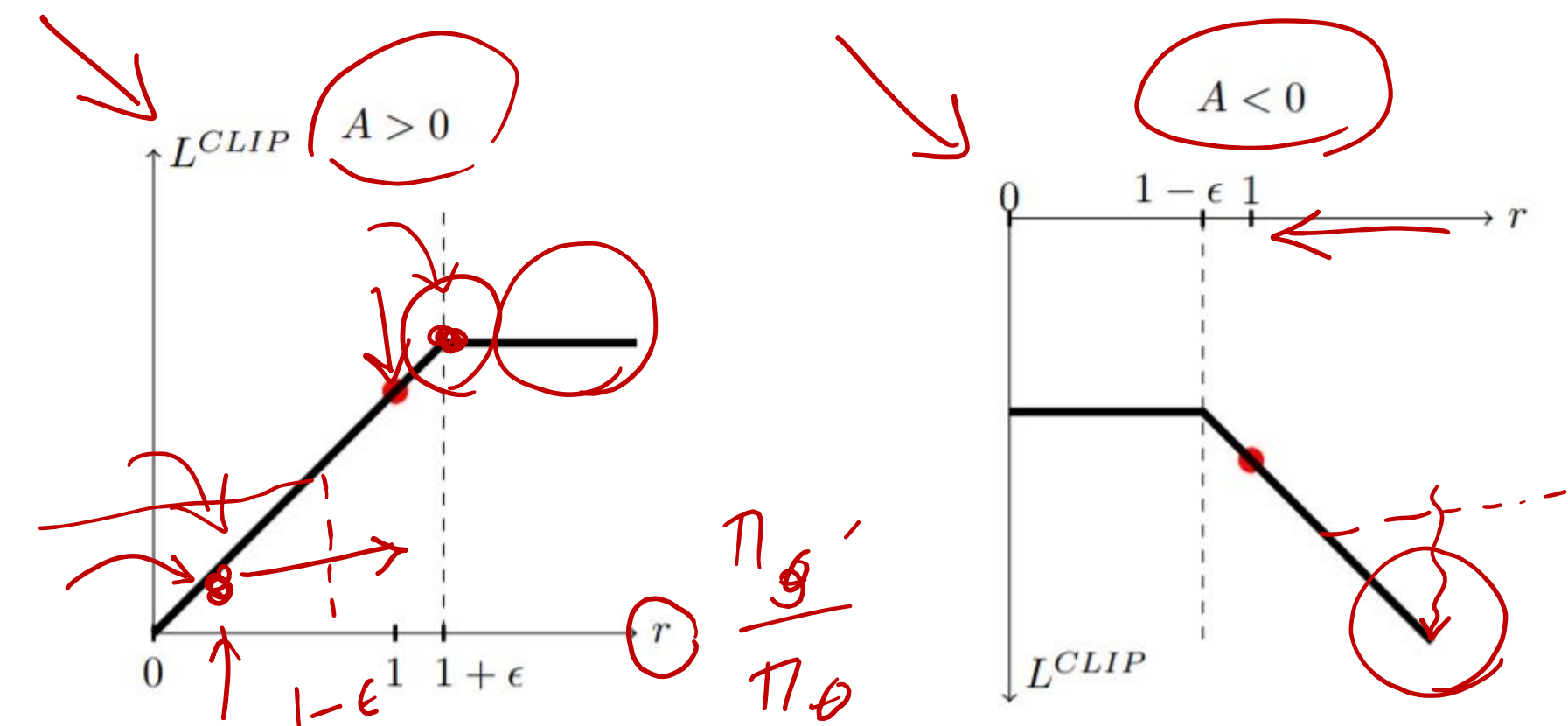
$$obj = \text{clip}\left(\frac{\pi_{\theta'}}{\pi_{\theta}}, 1-\epsilon, 1+\epsilon\right) \cdot A^{\pi_{\theta}}(s, a)$$

$\max_{\pi_{\theta}} \mathbb{E} \{ A^{\pi_{\theta}}(s, a) \}$
 importance Wei.
 $\mathbb{E}_{\pi_{\theta}^{old}} \left\{ \frac{\pi_{\theta}}{\pi_{\theta}^{old}} A^{\pi_{\theta}^{old}} \right\}$
 importance sampling

could be easily implemented with auto-diff packages

$\nabla_{\theta'} obj$
 $\arg\max_{\theta'} \mathbb{E}_{\{s \sim \mu_{\theta}, a \sim \pi_{\theta}\}} \min \left\{ \begin{array}{l} \frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)} A^{\pi_{\theta}}(s, a), \\ \text{clip}\left(\frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta}}(s, a) \end{array} \right.$
 unclipped
 ϵ lr

$\diamond$ Where $\text{clip}(x, 1-\epsilon, 1+\epsilon) = \begin{cases} 1-\epsilon & \text{if } x < 1-\epsilon \\ x & \text{if } 1-\epsilon \leq x \leq 1+\epsilon \\ 1+\epsilon & \text{if } x > 1+\epsilon \end{cases}$



Proximal Policy Optimization

Algorithm 1 PPO, Actor-Critic Style

```
→ for iteration=1,2,... do
  → for actor=1,2,...,N do
    Run policy  $\pi_{\theta_{\text{old}}}$  in environment for  $T$  timesteps
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T \rightarrow \bar{V}^{\pi_{\theta_{\text{old}}}}$ 
  end for
  Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
   $\theta_{\text{old}} \leftarrow \theta$ 
end for
```

SL
Data is Static

RL
Data depends on policy

min (clipped, unclipped)

Proximal Policy Optimization (cont.)

batch actor-critic algorithm:

1. Input: initial policy parameters θ_0 , initial value function parameters ϕ_0

2. for $k = 0, 1, 2, \dots$ do

3. Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.

4. Compute rewards-to-go $\hat{R}_t$.

5. Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k} .

6. Update the policy by maximizing the PPO-Clip objective (several epochs and mini-batches):

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

7. Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2,$$

typically via some gradient descent algorithm.

8. end for

$$J(\theta) = \mathbb{E} \left\{ \sum_{t=0}^T v(s_t, a_t) \right\}$$

$$Q(s_1, a_1)$$

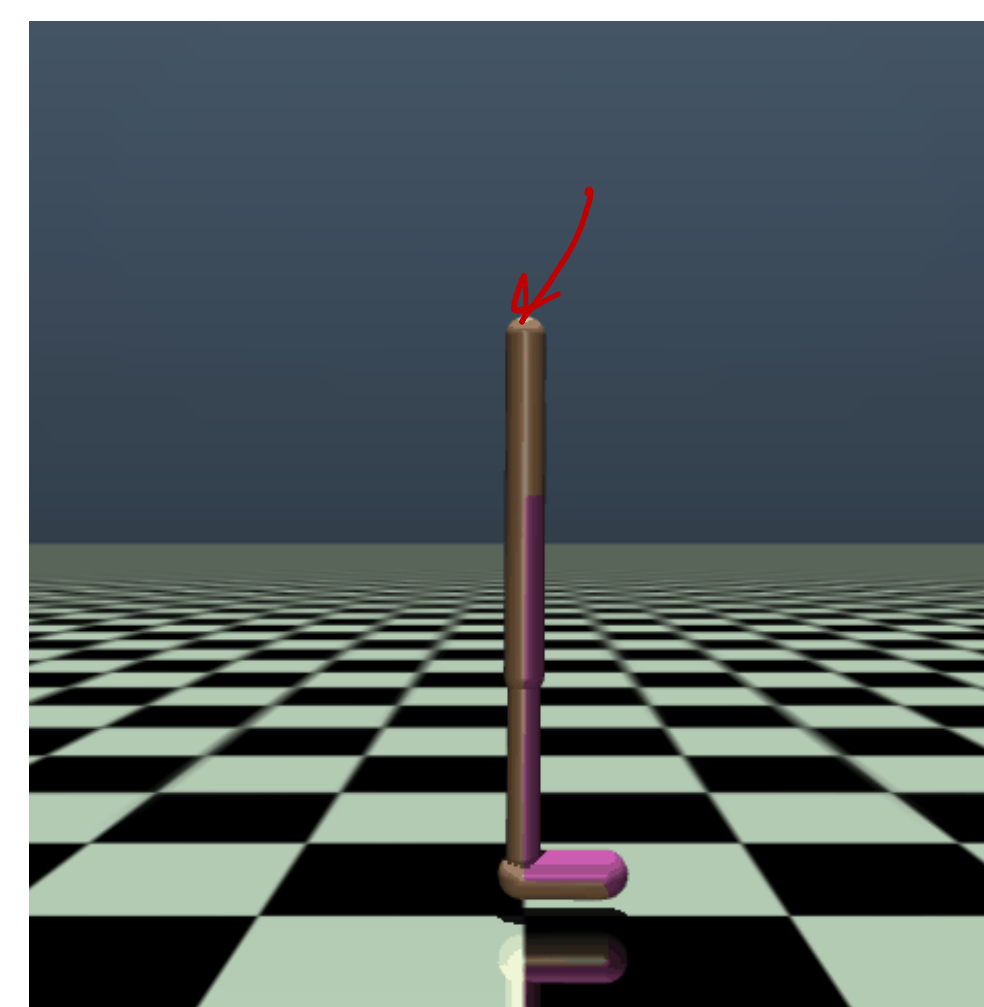
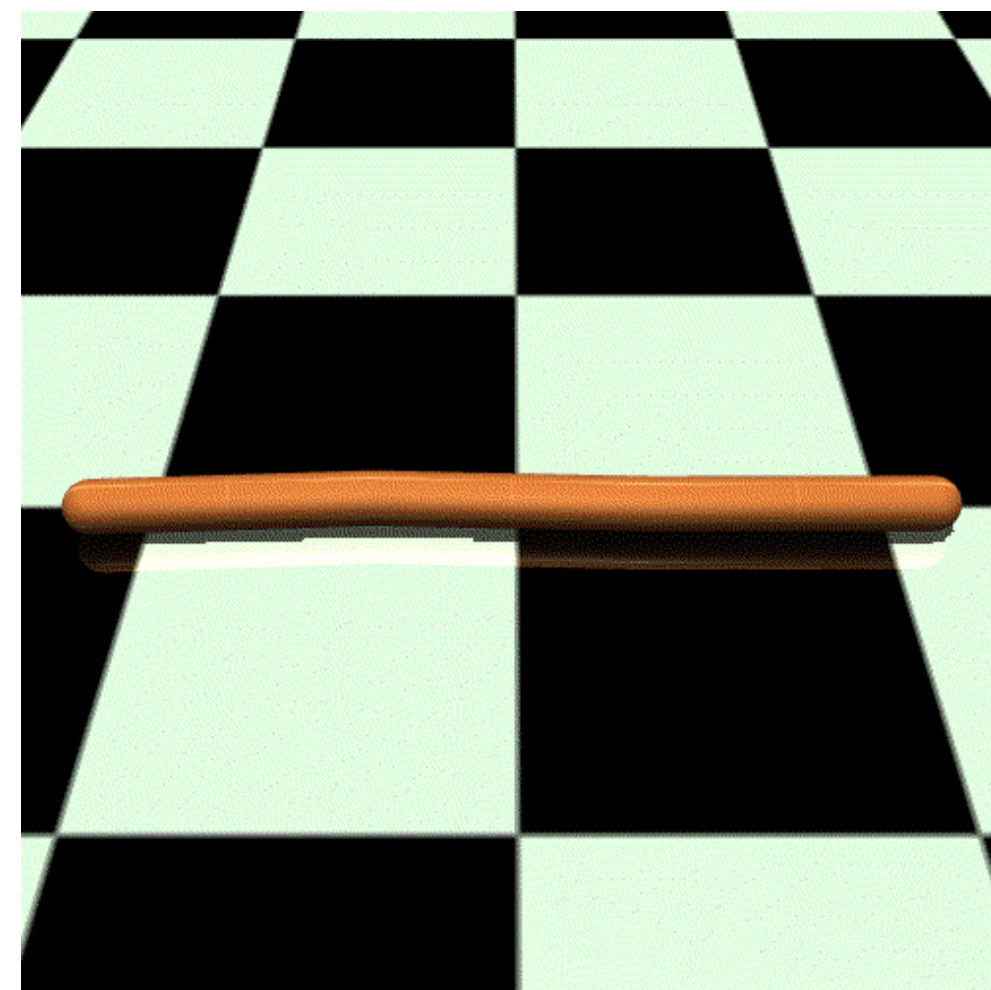
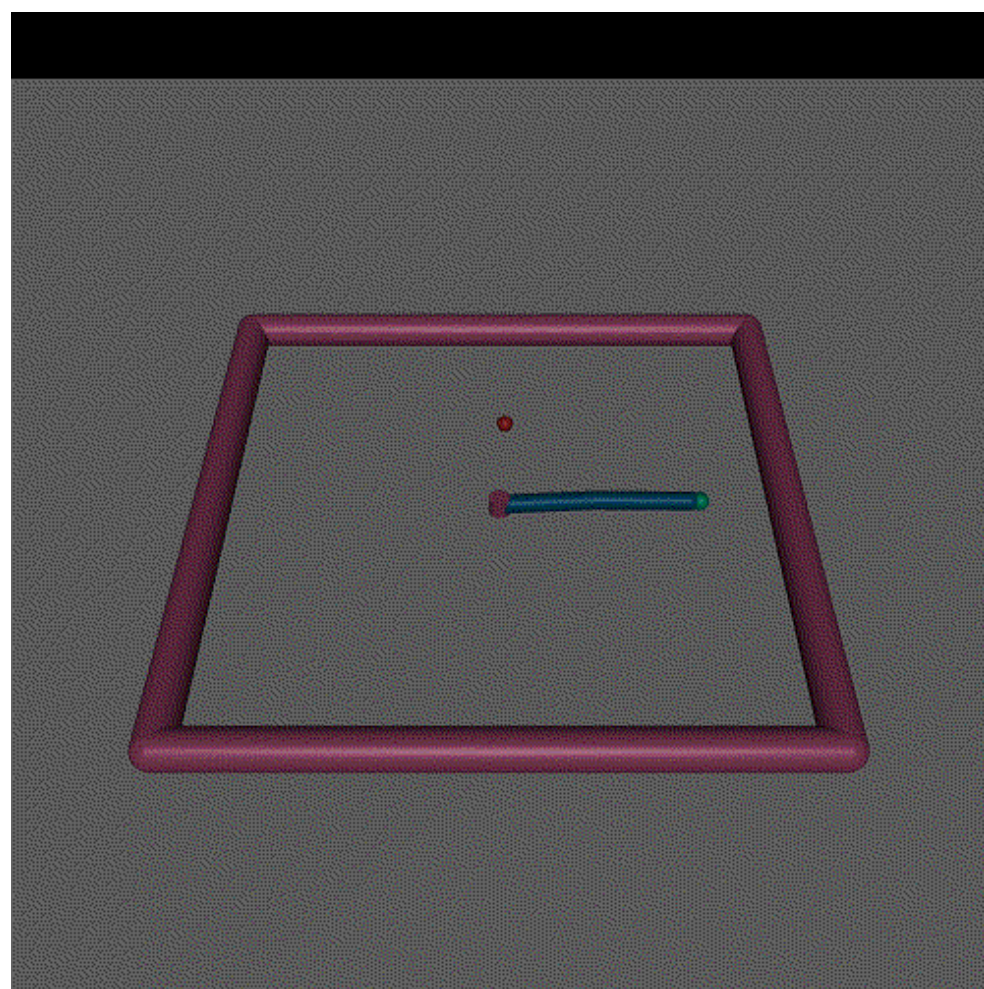
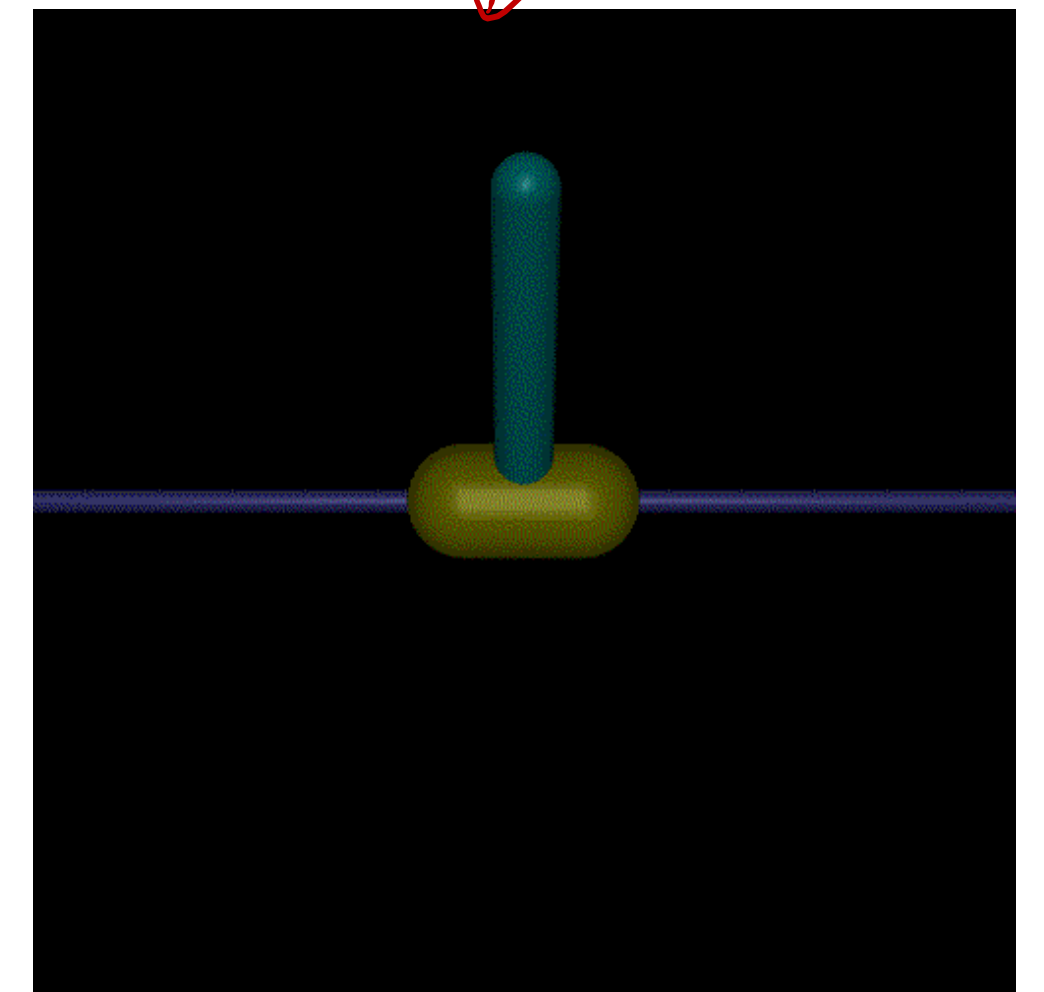
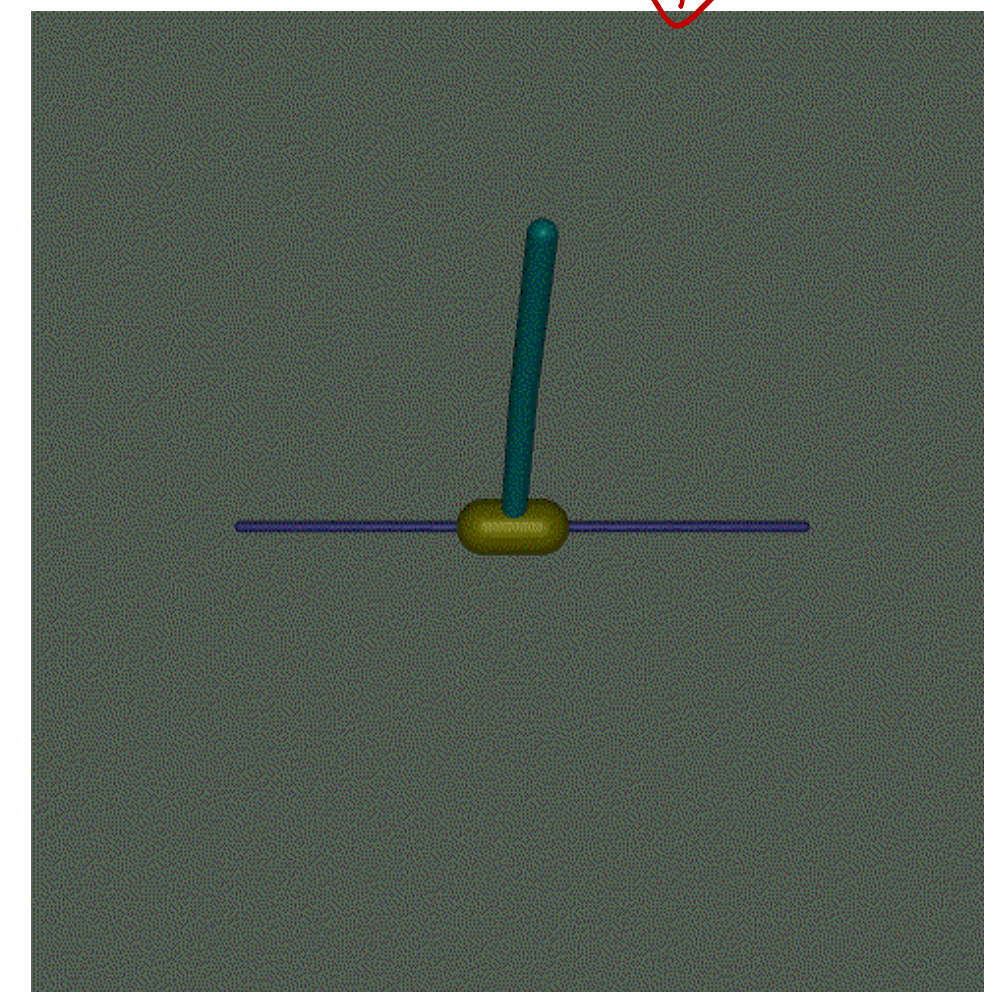
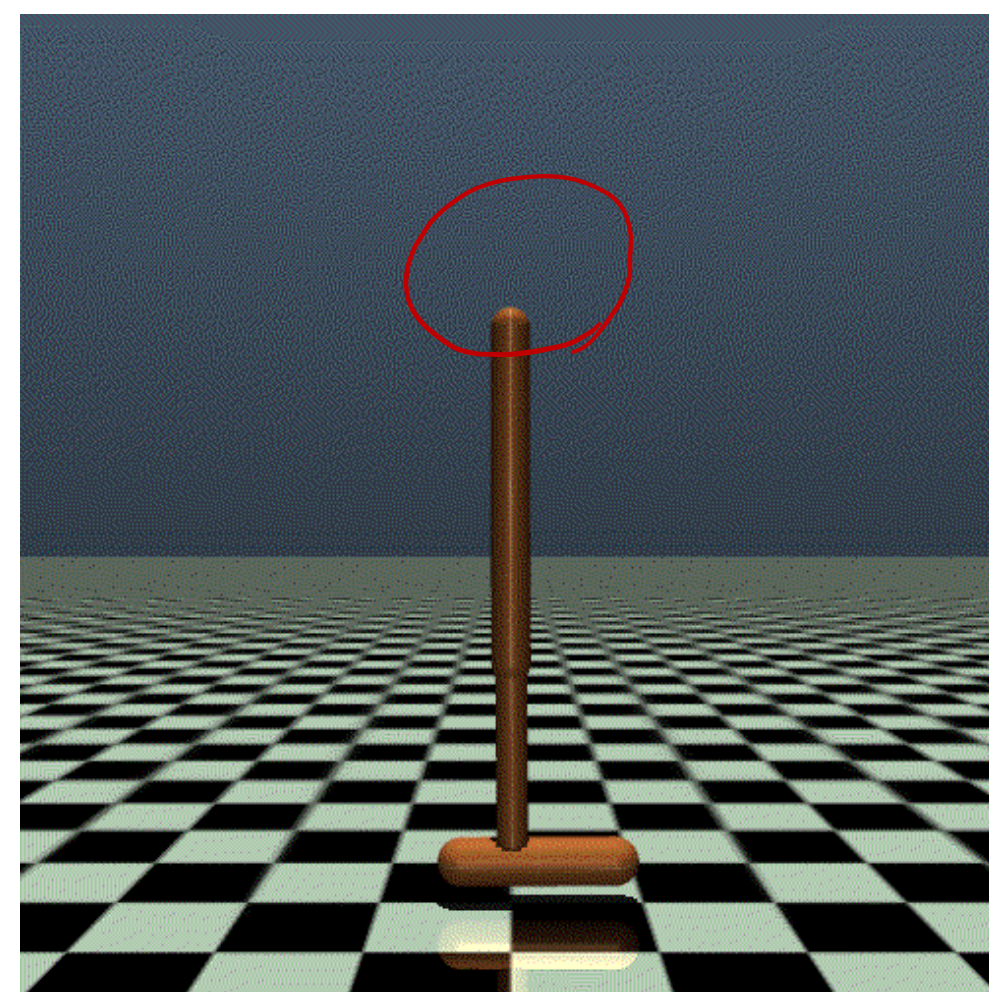
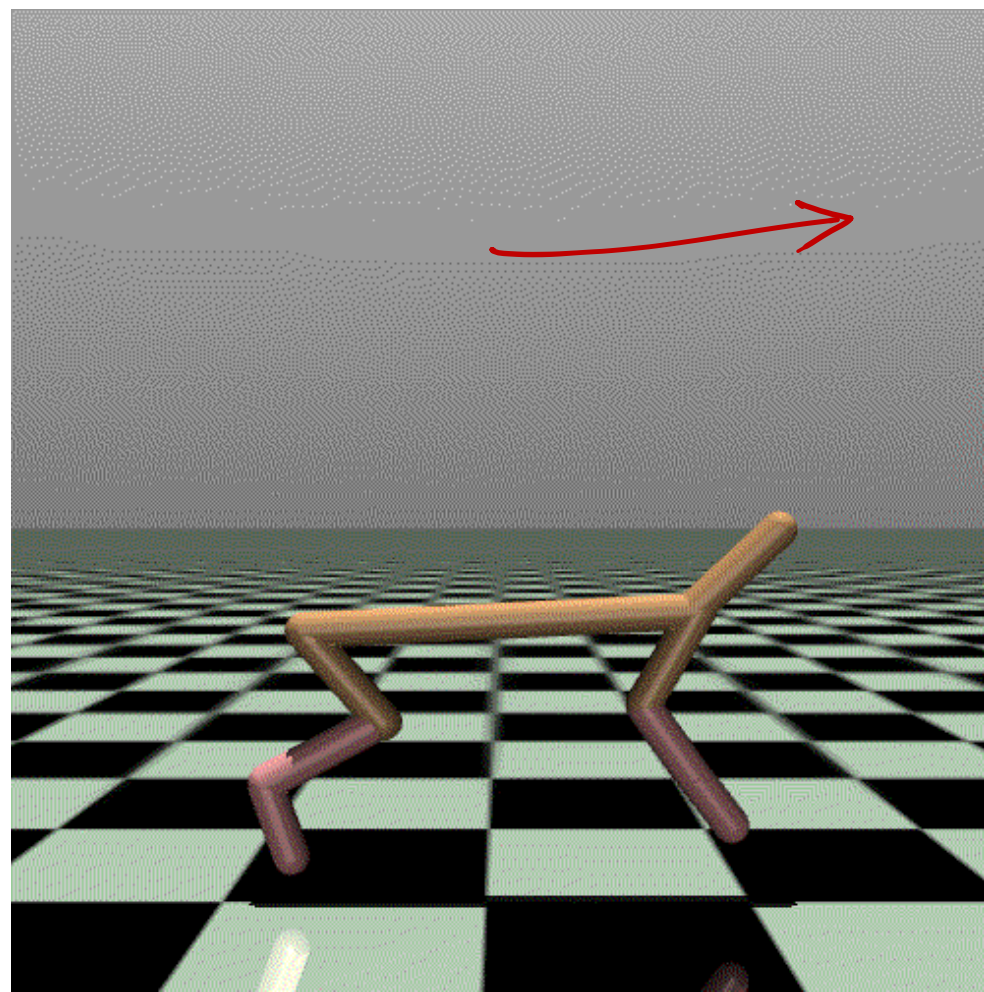
$$\{Q\}$$

$$= \mathbb{E} \{ Q - V(s_1) \}$$

$$A^{\pi_{\theta}}$$

$$= \mathbb{E}_{\pi_{\theta \text{ old}}} \left\{ \frac{\pi_{\theta}}{\pi_{\theta \text{ old}}} A^{\pi_{\theta \text{ old}}} \right\}$$

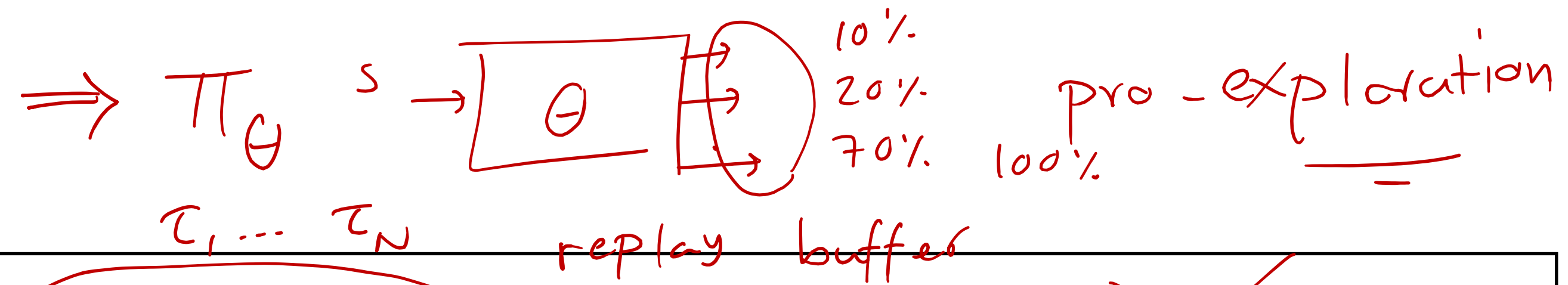
Environments



PPO Results



Soft Actor Critic



One cause for the poor sample efficiency of deep RL methods is ~~on-policy~~ learning: some of the most commonly used deep RL algorithms, such as TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017b) or A3C (Mnih et al., 2016), require new samples to be collected for each gradient step.

More importantly, the maximum entropy formulation provides a substantial improvement in exploration and robustness: as discussed by Ziebart (2010), maximum entropy policies are robust in the face of model and estimation errors, and as demonstrated by (Haarnoja et al., 2017), they improve exploration by acquiring diverse behaviors.

Soft Policies

$$J(\theta) = \mathbb{E}_{\pi} \left[\sum_{t=1}^T r(s_t, a_t) \right]$$

$$J(\pi) = \sum_{t=0}^T \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi}} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t)) \right]$$

$\alpha > 0$ would cause more spread of prob. over actions in every state.

◆ How does the Bellman equation change?

Bellman's Op.

$$\mathcal{T}^{\pi} Q(s_t, a_t) \triangleq r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} [V(s_{t+1})]$$

Where

reward $t' > t$
entropy $t' > t$

$$\mathcal{H}(\pi(\cdot | s_t)) = - \sum_a \pi(a | s) \log \pi(a | s)$$

entropy @ $t' = t$

$$V(s_t) = \mathbb{E}_{a_t \sim \pi} [Q(s_t, a_t) - \log \pi(a_t | s_t)]$$

Used to be:

$$Q(s, a) = \mathbb{E} \left(\sum_{t'=0}^{\infty} \gamma^{t'} r(s_{t'}, a_{t'}) \right)$$

$$\begin{aligned} \underline{Q}_{t+1}(s, a) &\leftarrow r(s, a) + \gamma \mathbb{E}_{s'} [V_t(s')] \\ V_t(s) &\leftarrow \mathbb{E}_{a \sim \pi} [Q(s, a)] \end{aligned}$$

Soft Policies (cont.)

Lemma 1

(Soft Policy Evaluation). Consider the soft Bellman backup operator $\mathcal{T}^\pi$ in Equation (2) and a mapping $Q^0: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{T}^\pi Q^k$.

Then the sequence Q^k will converge to the soft Q-value of π as $k \rightarrow \infty$.

Val. Iter.

Optimal Soft Policy

Conventional RL:

$$\underline{\pi}(s) = \underset{a}{\operatorname{argmax}} Q(s, a)$$

max

vs.

♦ The optimal soft policy (optimizing entropy augment objective) is:

Soft RL:

$$\pi^*(a|s) = \frac{\exp Q(s, a)}{\sum_{a'} \exp Q(s, a')}$$

Softmax

$$\pi(a|s) = \frac{e^{Q^\pi(s, a)}}{\sum_{a'} e^{Q^\pi(s, a')}} \quad \uparrow$$

$$\max_{\pi} \mathbb{E}_{a \sim \pi(\cdot|s)} [Q(s, a) + \alpha (-1) \log \pi(a|s)]$$

$\sum_a \pi(a|s) = 1$

$$\mathcal{L} = \sum_a Q^\pi(s, a) \pi(a|s) - \alpha \pi(a|s) \log \pi(a|s) - \lambda (\sum \pi(a|s) - 1)$$

$$\frac{\partial \mathcal{L}}{\partial \pi(a|s)} = Q^\pi(s, a) - \alpha \log \pi(a|s) - \alpha - \lambda = 0 \leadsto \pi(a|s) = \beta e^{\frac{1}{\alpha} Q^\pi(s, a)}$$

Soft actor-critic

1. Q-Function update

update Q-Function to evaluate policy

$$Q(s, a) \leftarrow r(s, a) + \mathbb{E}_{s' \sim p_{s, a}, a' \sim \pi} [Q(s', a') - \log \pi(a' | s')]$$

This converges to Q^π

2. Update policy

Update the policy with gradient of information projection

$$\pi_{\text{new}} = \arg \min_{\pi'} D_{\text{KL}} \left(\pi'(\cdot | s) \parallel \frac{1}{Z} \exp Q^{\pi_{\text{old}}}(s, \cdot) \right)$$

In practice only take one gradient step on this objective

3. Interact with the world, collect more data

Critic Upd.

Q, V, π
critic → actor

τ^π

(s, a, s')
 π_{old}

New Q-network

targets
s' replay

$$r(s, a) + \gamma \mathbb{E}_{s', a' \sim \pi} [Q(s', a') - \log \pi(a' | s')]$$

$$r(s, a) + \gamma V(s')$$

$$\pi^* = \frac{e^{\frac{1}{2} Q(s, a)}}{Z}$$

Soft actor-critic (cont.)

Soft Actor-Critic algorithm:

Inputs: The learning rates, λ_π , λ_Q , and λ_V for functions π_θ , Q_w , and V_ψ respectively; the weighting factor τ for exponential moving average.

1. Initialize parameters θ, w, ψ , and $\bar{\psi}$.

→ 2. for each iteration do

3. (In practice, a combination of a single environment step and multiple gradient steps is found to work best).

→ 4. for each environment setup do

5. ~~$a_t \sim \pi_\theta(a_t|s_t)$~~

6. ~~$s_{t+1} \sim \rho_\pi(s_{t+1}|s_t, a_t)$~~

7. $\Rightarrow \mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$ $\rightsquigarrow$ replay buff.

8. for each gradient update step do

9. $\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi)$

10. $w \leftarrow w - \lambda_Q \nabla_w J_Q(w)$

11. $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta J_\pi(\theta)$

12. $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$

Loss functions

New Bellman

$$\rightarrow J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim \mathbb{D}} \left[\frac{1}{2} \left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi} [Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t)] \right)^2 \right]$$

Sample π_ϕ

alternative:

With

(*)

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathbb{D}} \left[\frac{1}{2} \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right]$$

target

sampled from replay

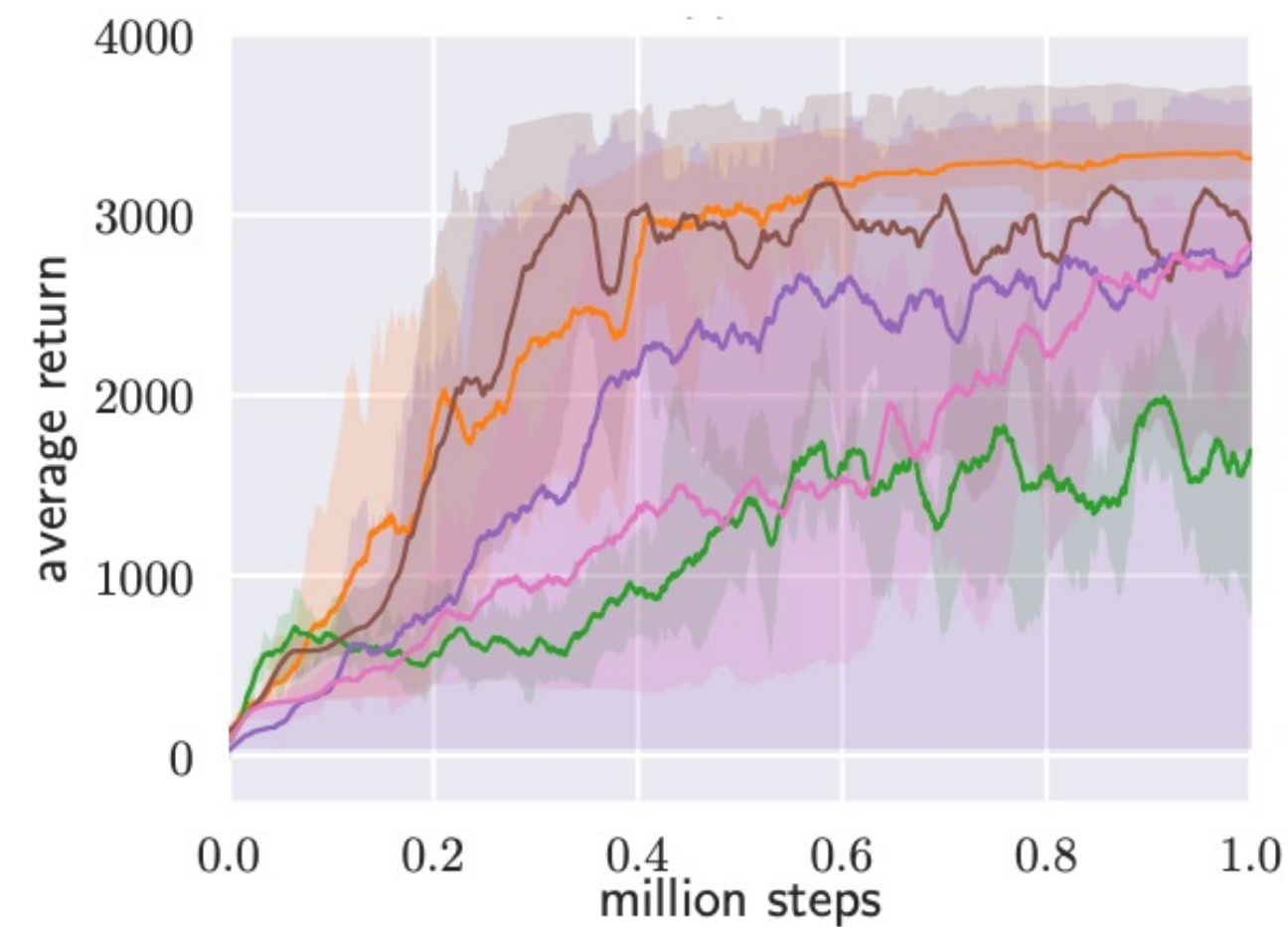
$\mathbf{s}_t, \mathbf{a}_t \sim \pi$

$$\rightarrow \hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim \mathbb{P}} [V_{\bar{\psi}}(\mathbf{s}_{t+1})]$$

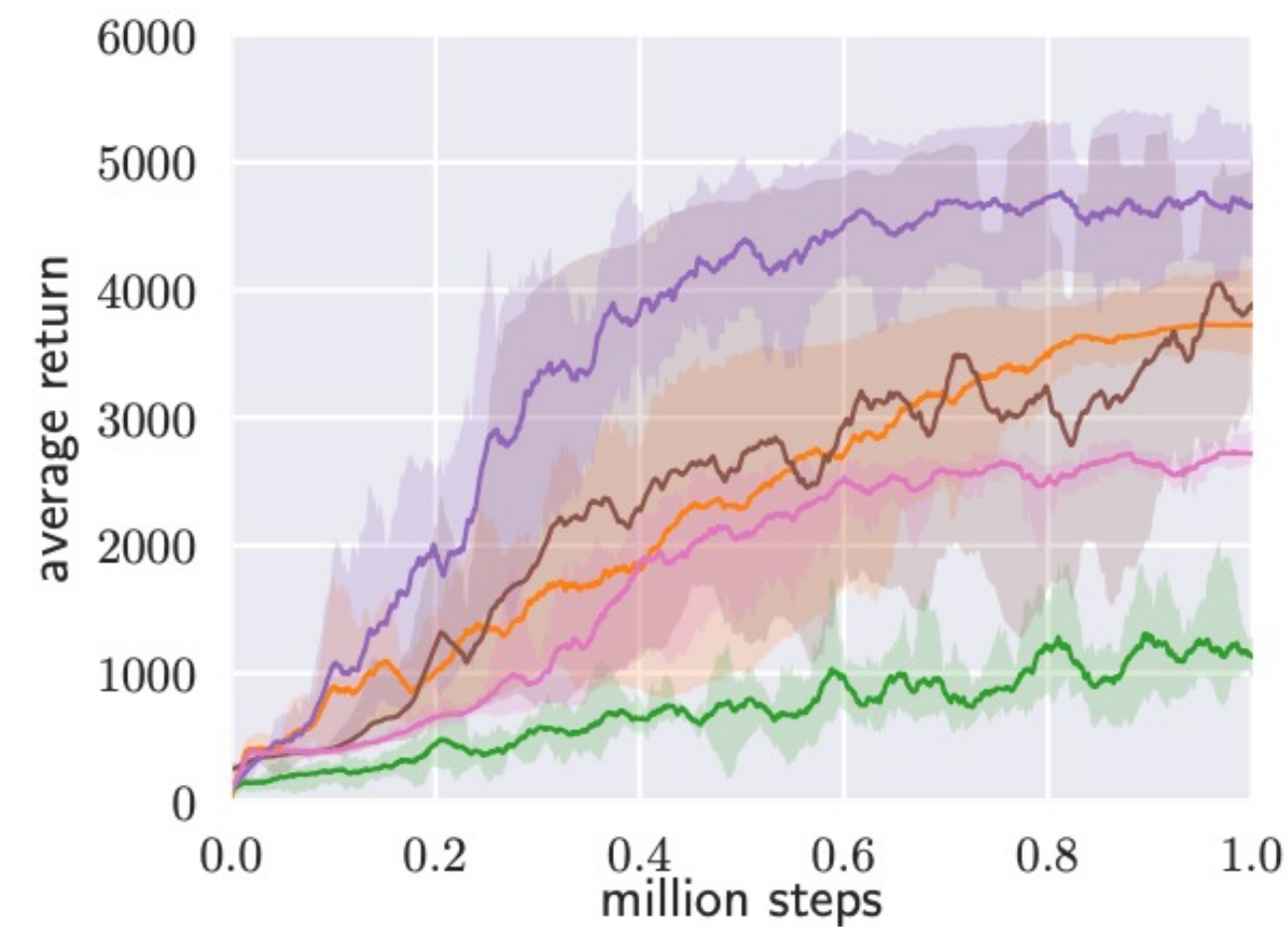
$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathbb{D}} \left[\underline{D}_{\text{KL}} \left(\pi_\phi(\cdot | \mathbf{s}_t) \left| \frac{\exp(Q_\theta(\mathbf{s}_t, \cdot))}{Z_\theta(\mathbf{s}_t)} \right. \right) \right]$$

replay buff.

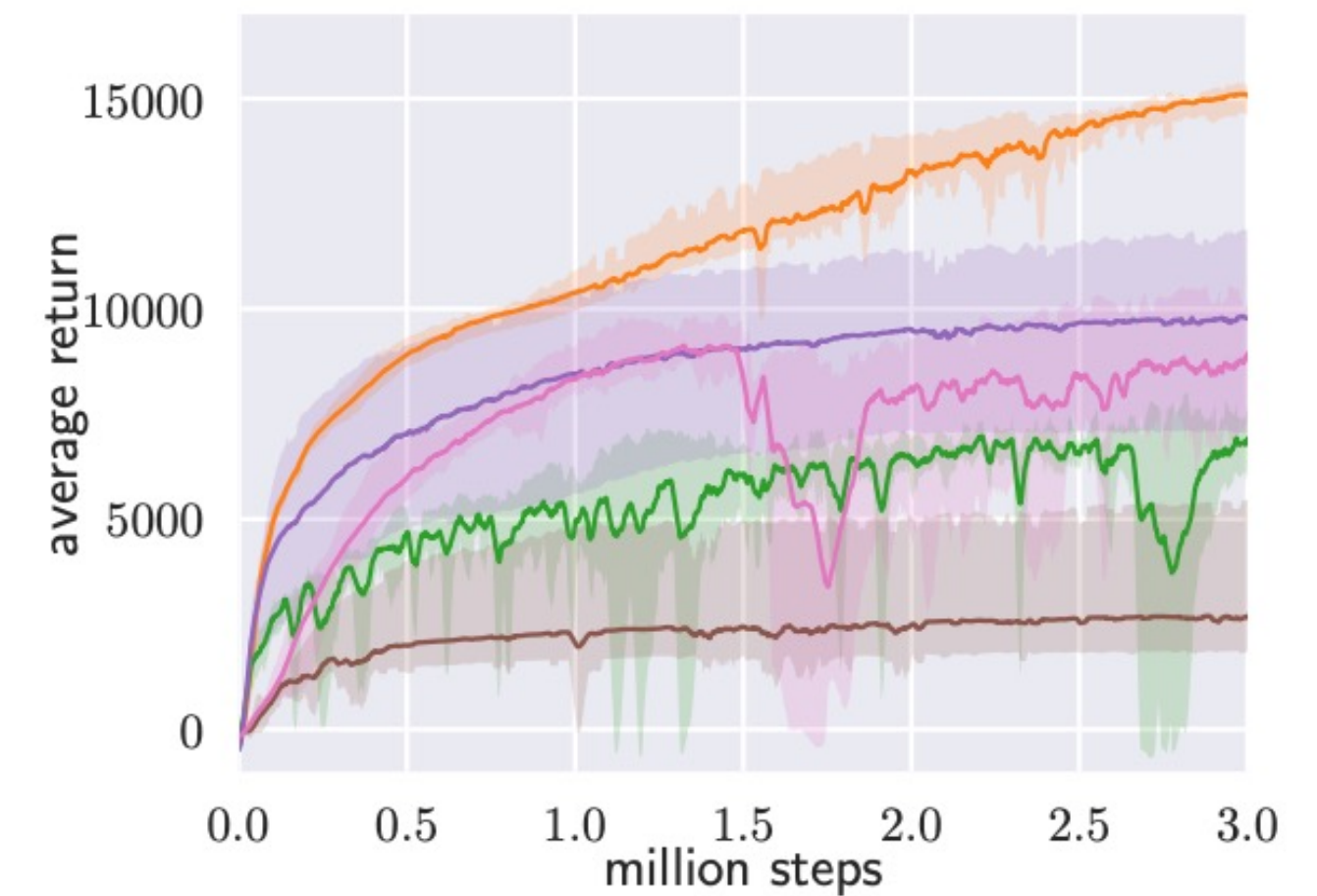
SAC Results



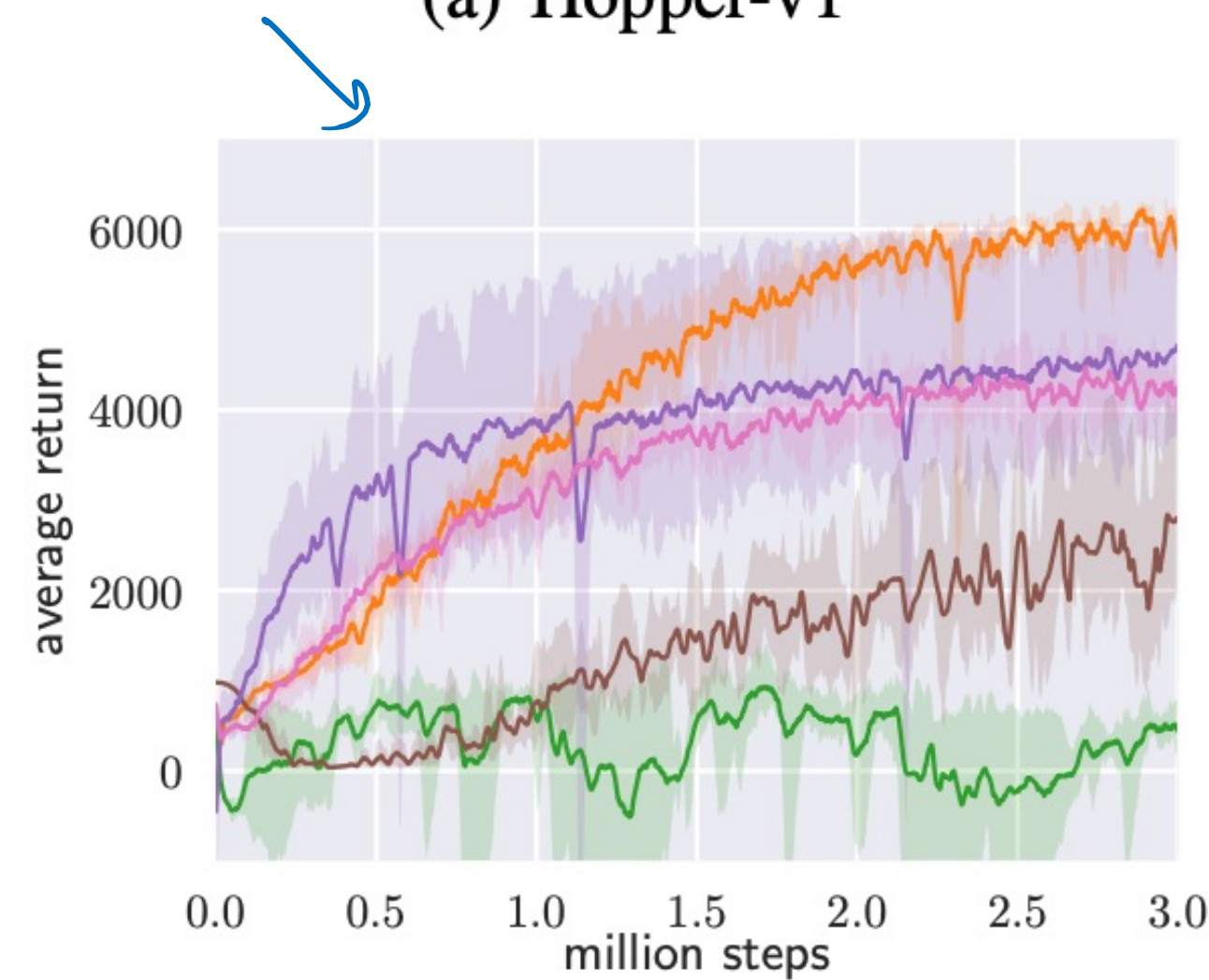
(a) Hopper-v1



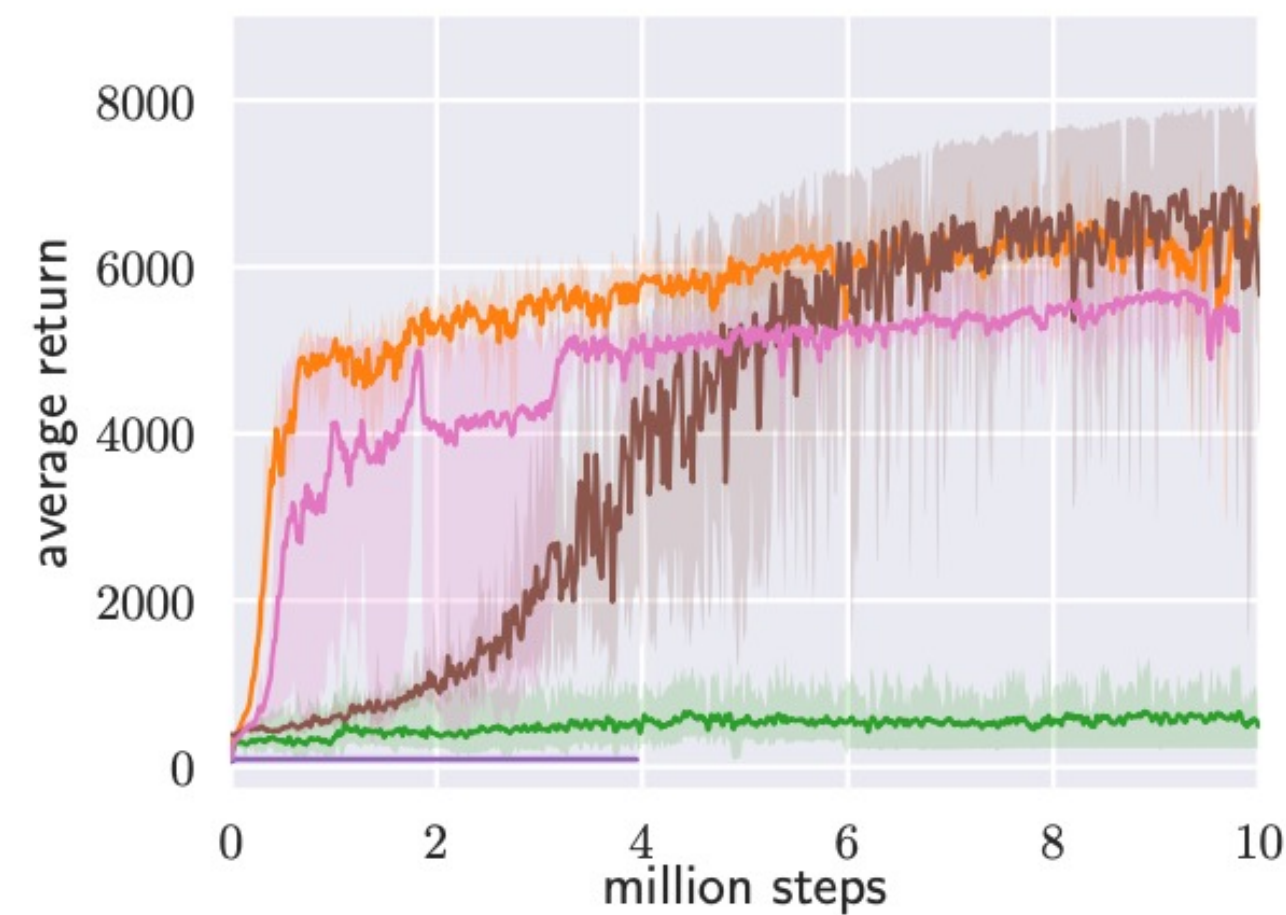
(b) Walker2d-v1



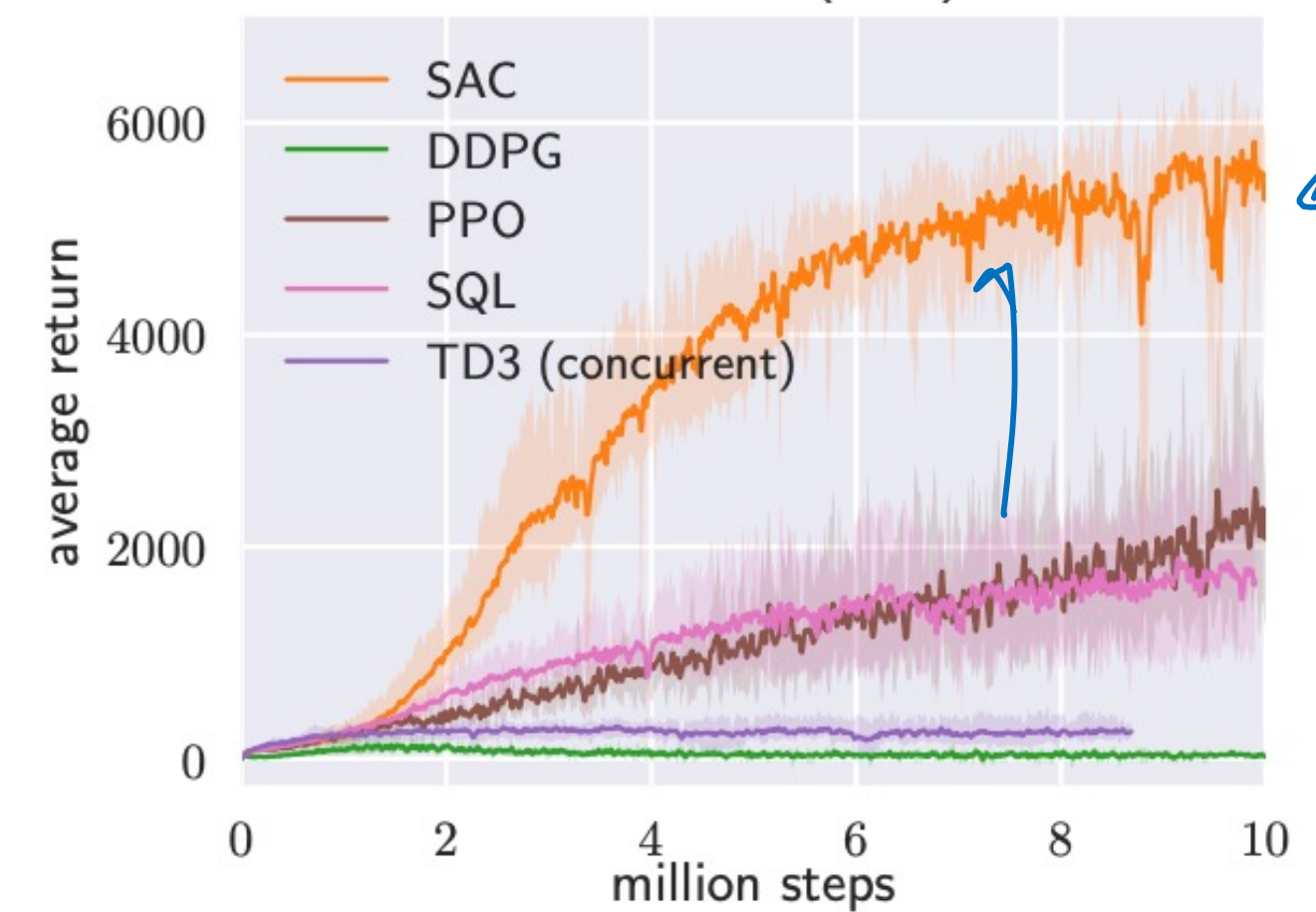
(c) HalfCheetah-v1



(d) Ant-v1

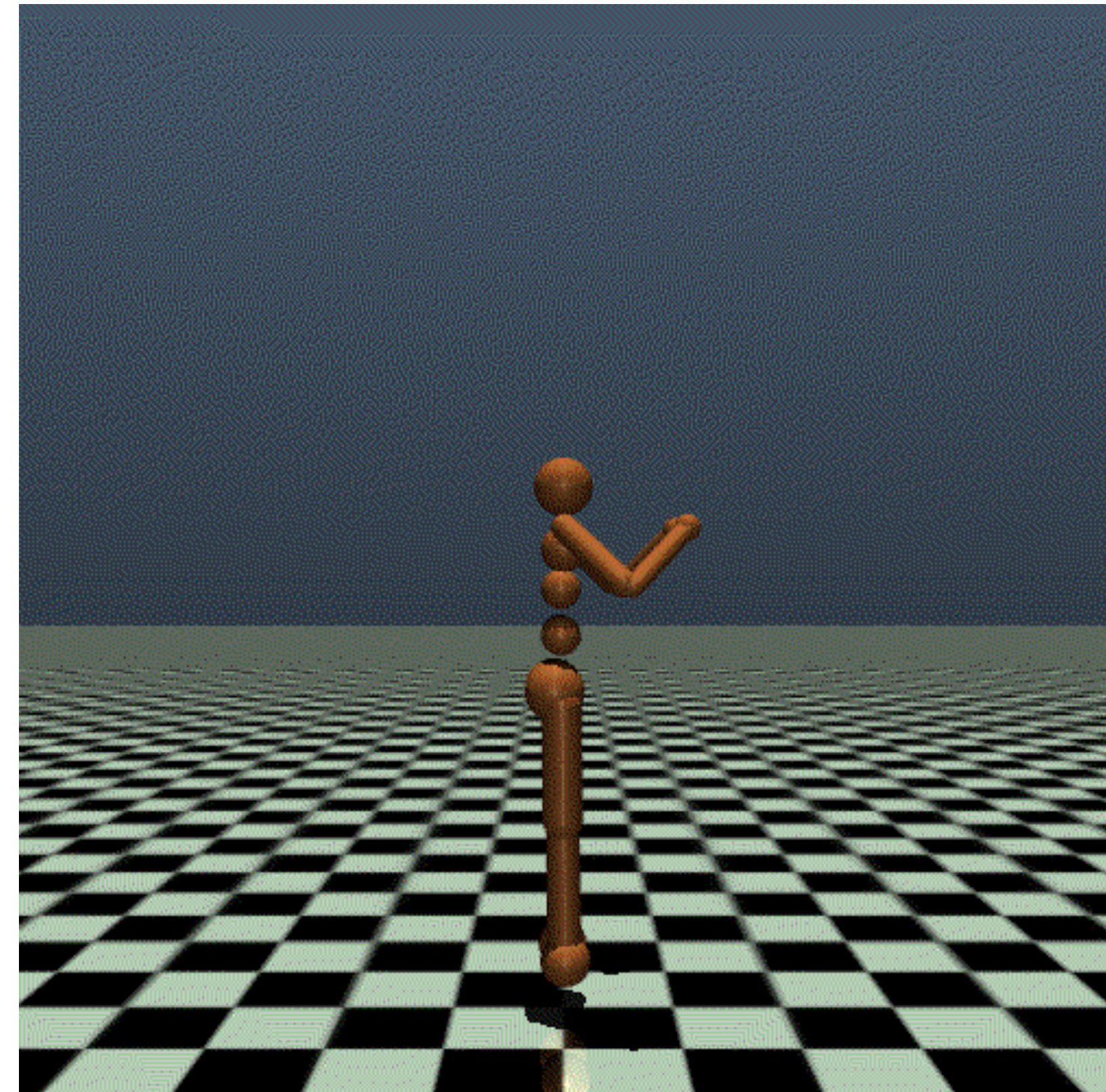
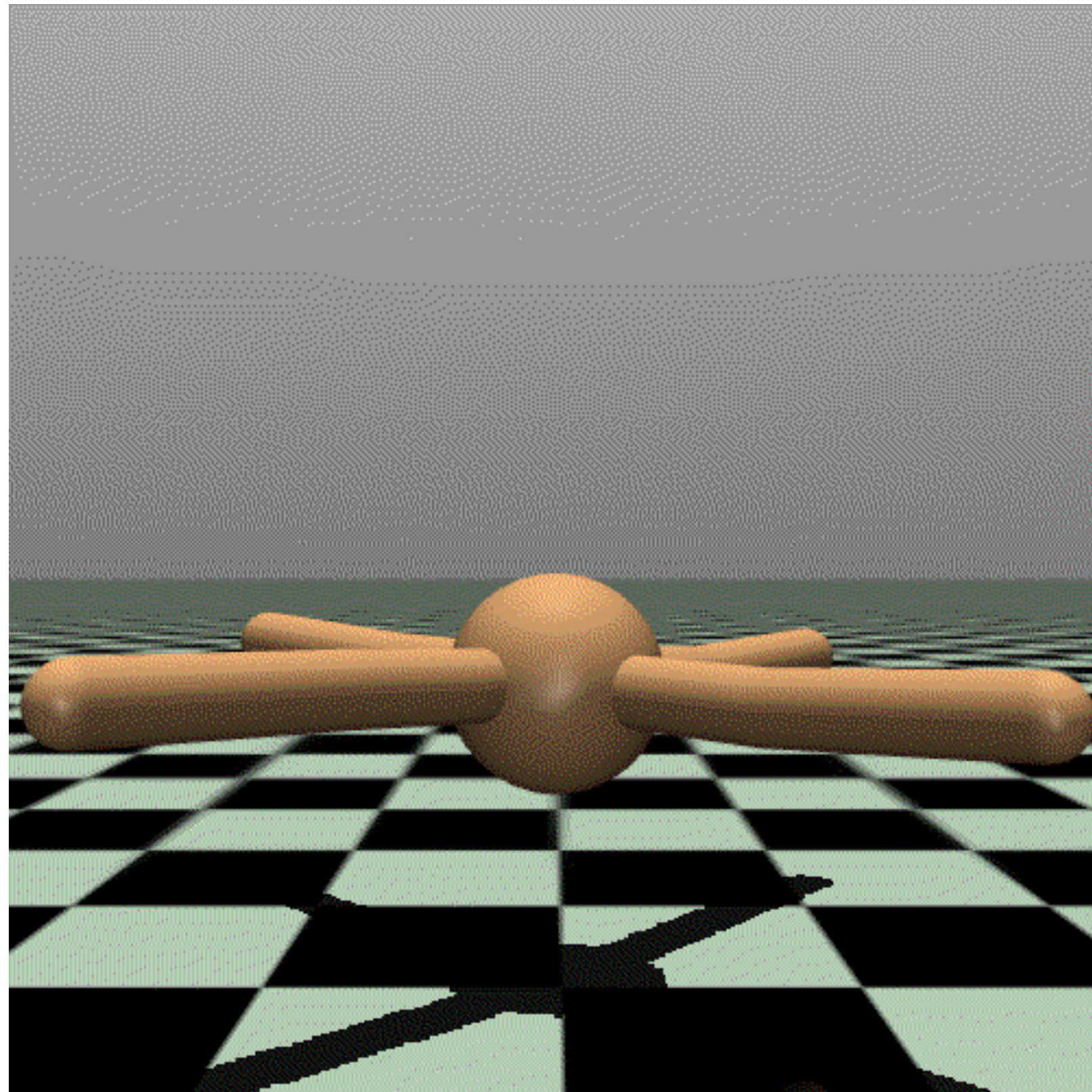


(e) Humanoid-v1



(f) Humanoid (rllab)

Environments



Related Papers

- ♦ Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning: [introduces REINFORCE algorithm](#)
- ♦ Sutton, McAllester, Singh, Mansour (1999). Policy gradient methods for reinforcement learning with function approximation: [actor-critic algorithms with value function approximation](#)
- ♦ 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: [TD\(\$\lambda\$ \) actor-critic](#)
- ♦ Degris, White, Sutton. (2012). Off-policy actor-critic: [off-policy actor-critic with importance sampling](#)
- ♦ Silver et al. (2014). Deterministic policy gradient algorithms: [DPG](#)
- ♦ Lillicrap et al. (2016). Continuous control with deep reinforcement learning: continuous Q-learning with actor network for approximate maximization: [DDPG](#)

Related Papers (cont.)

- ♦ Kakade (2001). A Natural Policy Gradient: [natural policy gradient](#)
- ♦ Schulman, L., Moritz, Jordan, Abbeel (2015). Trust region policy optimization: [TRPO](#)
- ♦ Schulman, Wolski, Dhariwal, Radford, Klimov (2017). Proximal policy optimization algorithms: [PPO](#)