



Computer Engineering Department

Reinforcement Learning: Policy Gradients

Mohammad Hossein Rohban, Ph.D.

Spring 2024

Courtesy: Most of slides are adopted from CS 285 Berkeley.

Overview of Value-Based Methods

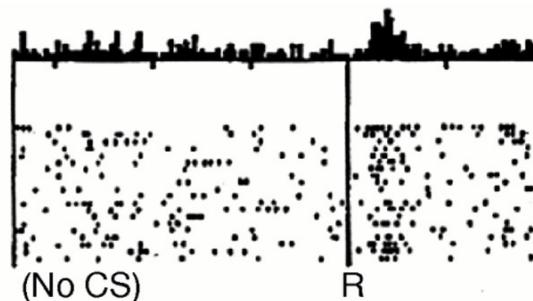
- Monte Carlo method
- Temporal difference learning
- N-step TD
- TD (λ) (forward and backward view)
- On-policy and off-policy versions of TD
- Function approximation



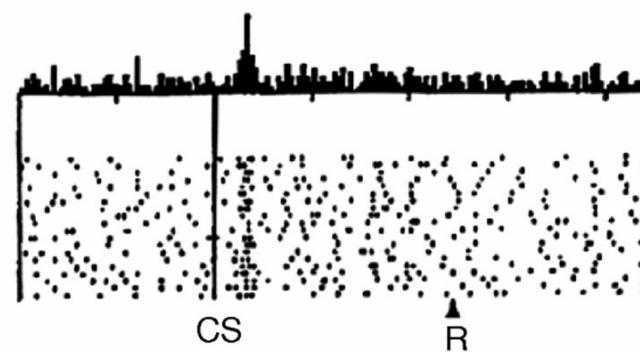
policy
Improvement
 ϵ -greedy

Reward Prediction Error in the Brain

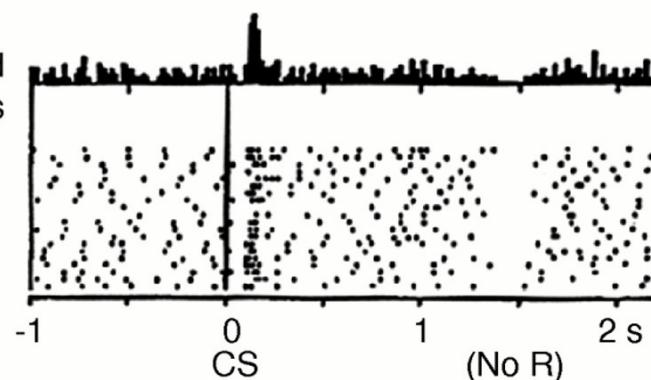
No prediction
Reward occurs



Reward predicted
Reward occurs

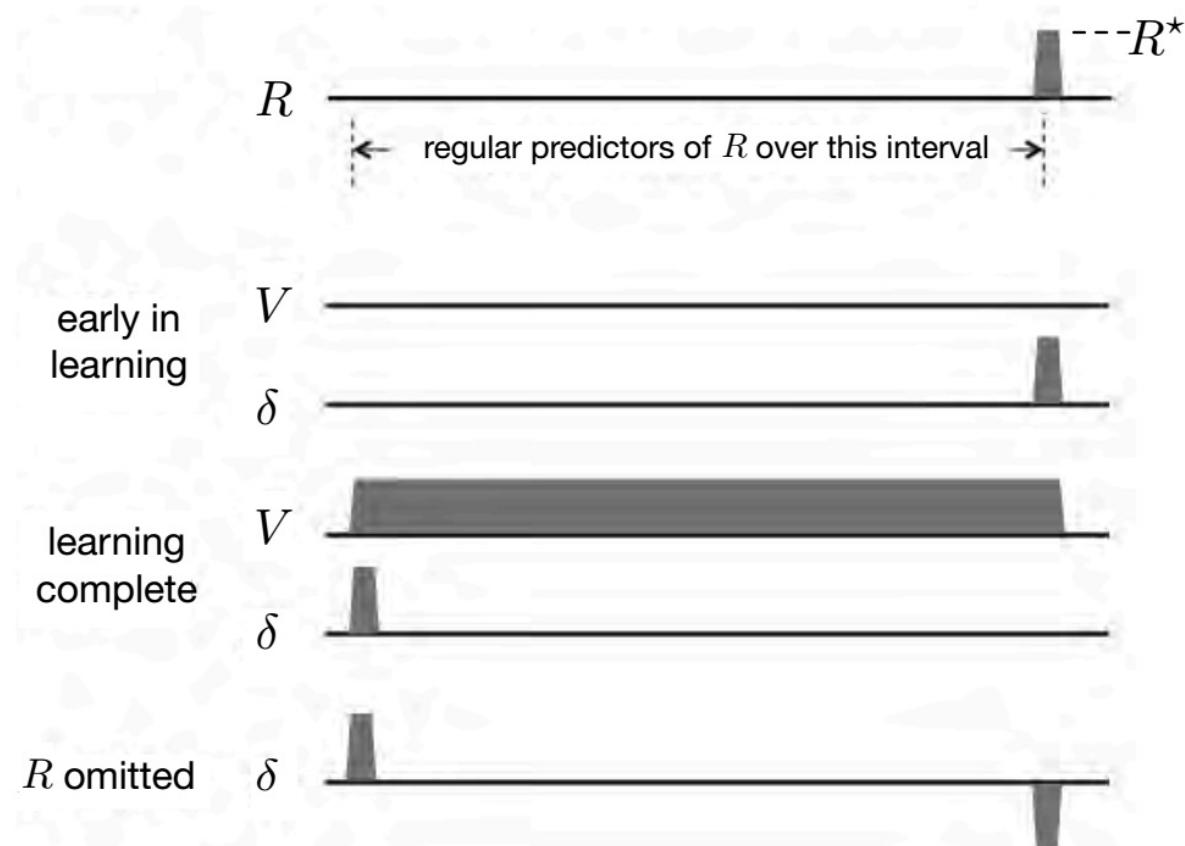


Reward predicted
No reward occurs



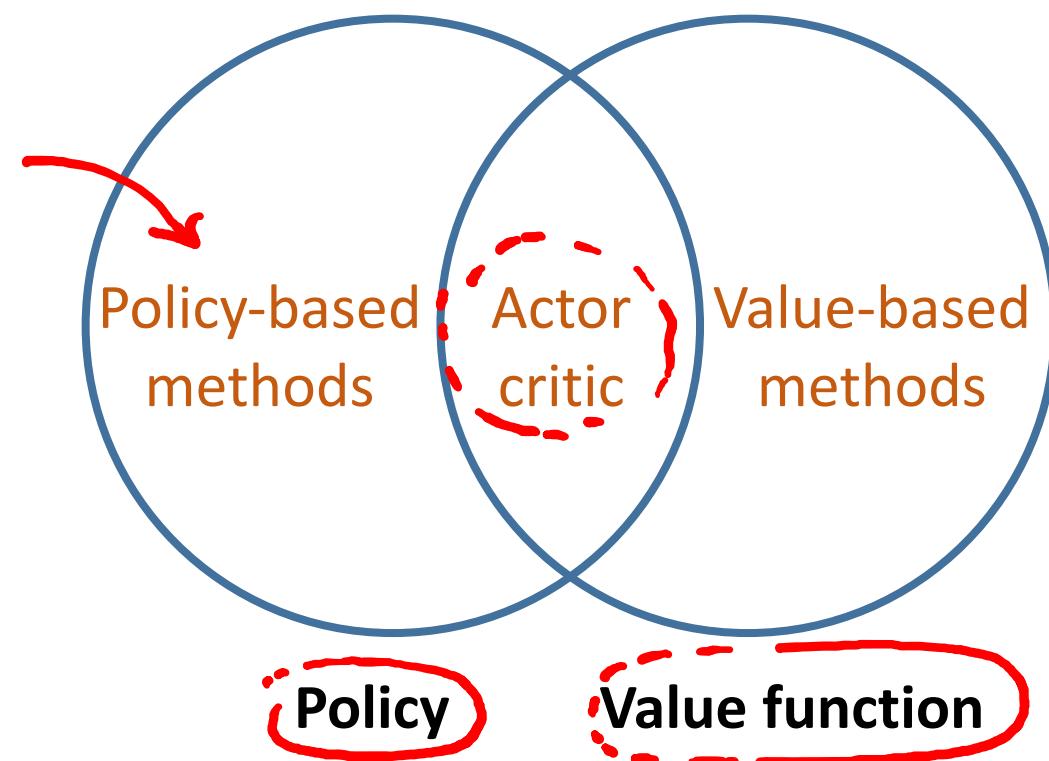
[Schultz, Dayan, and Montague. 1997]

TD Error/Dopamine Correspondence

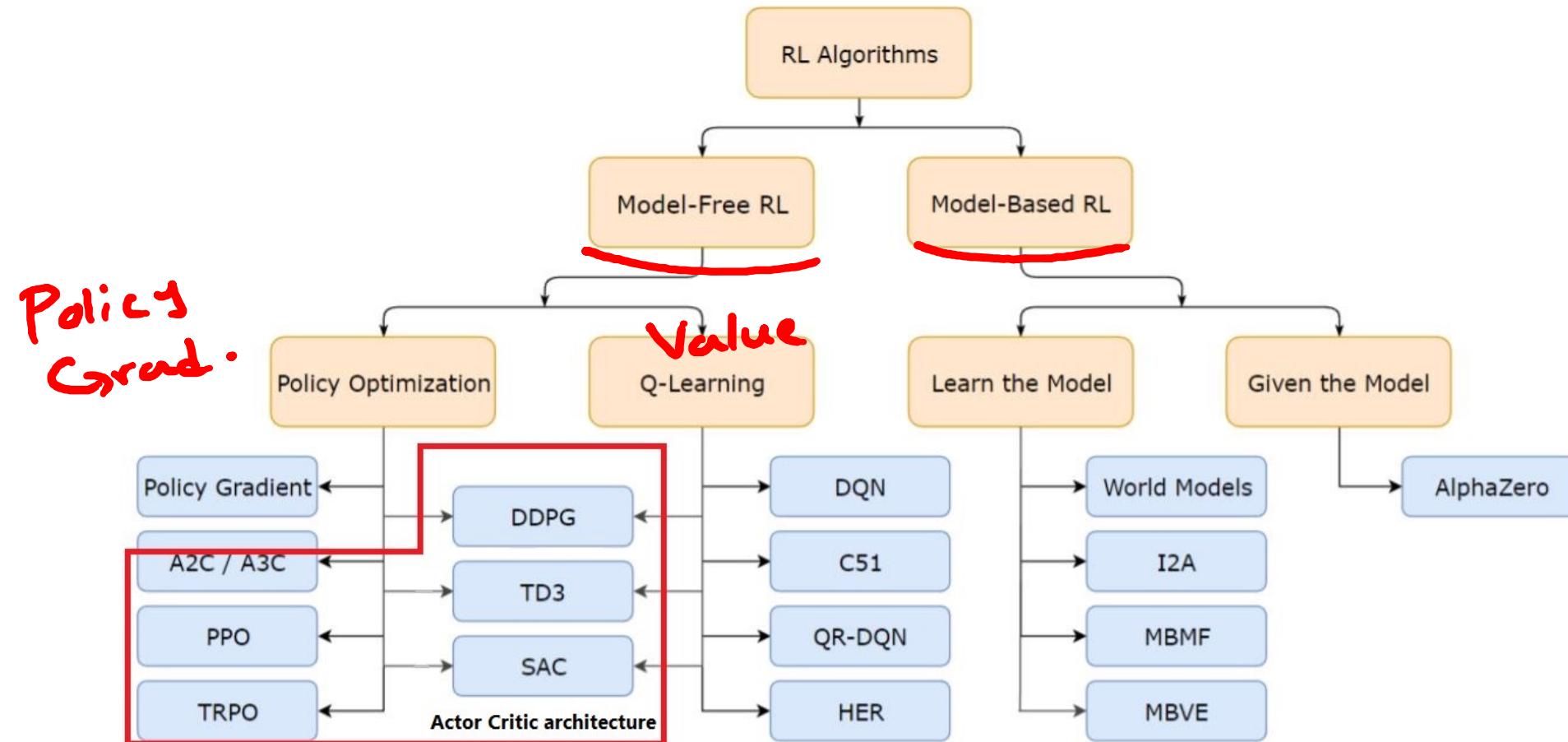


Model-Free RL

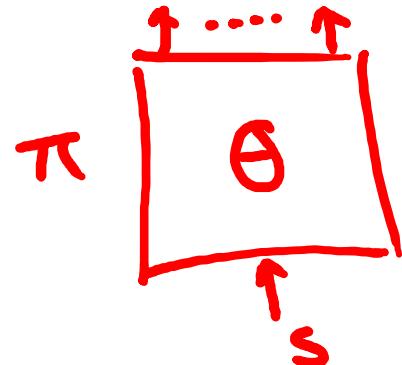
- Value-based methods
 - Learnt value function
 - Implicit policy
- Policy-based methods
 - No value function
 - Learnt policy
- Actor-critic methods
 - Learnt value function
 - Learnt policy



Overview of Modern RL Methods



The Goal of Reinforcement Learning



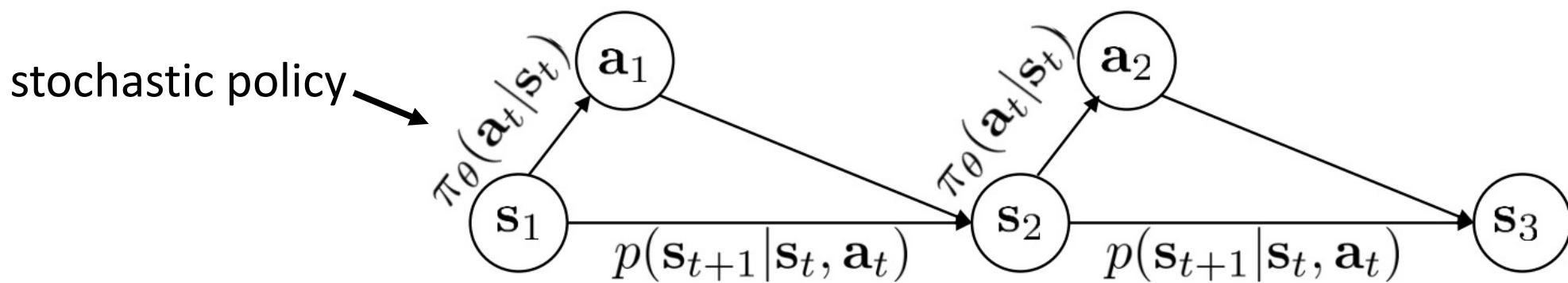
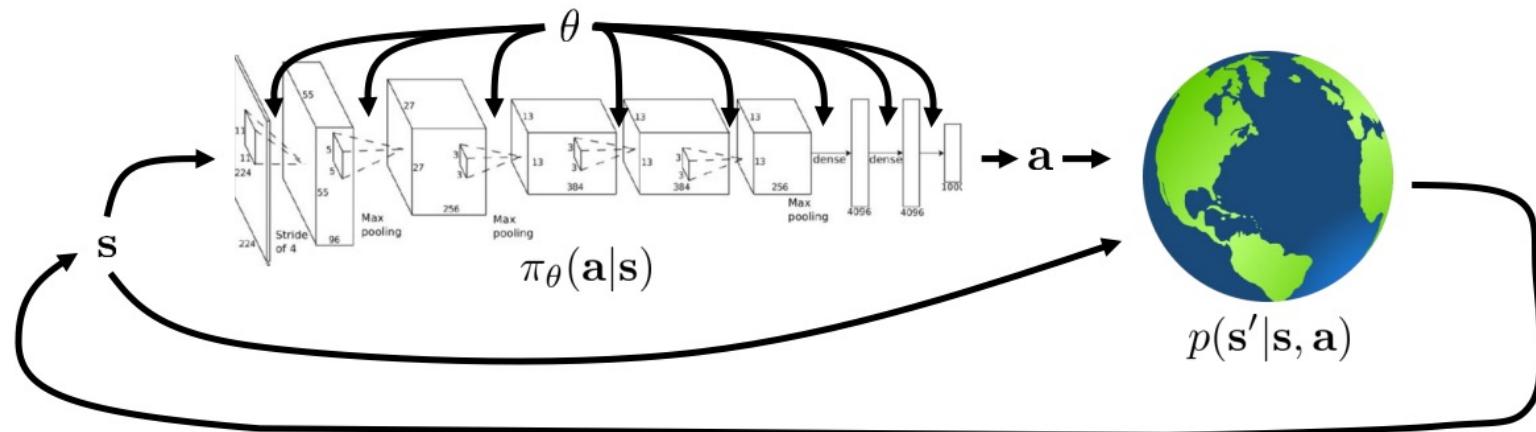
$$J(\theta) = E \left[\sum_t r(s_t, a_t) \right]$$

return
↑
 π_θ

$s_1 \sim p(s_1)$ R.V.
max θ $J(\theta)$
 $a_t \sim \pi_\theta(\cdot | s_t)$
 $s_{t+1} \sim p(\cdot | s_t, a_t)$

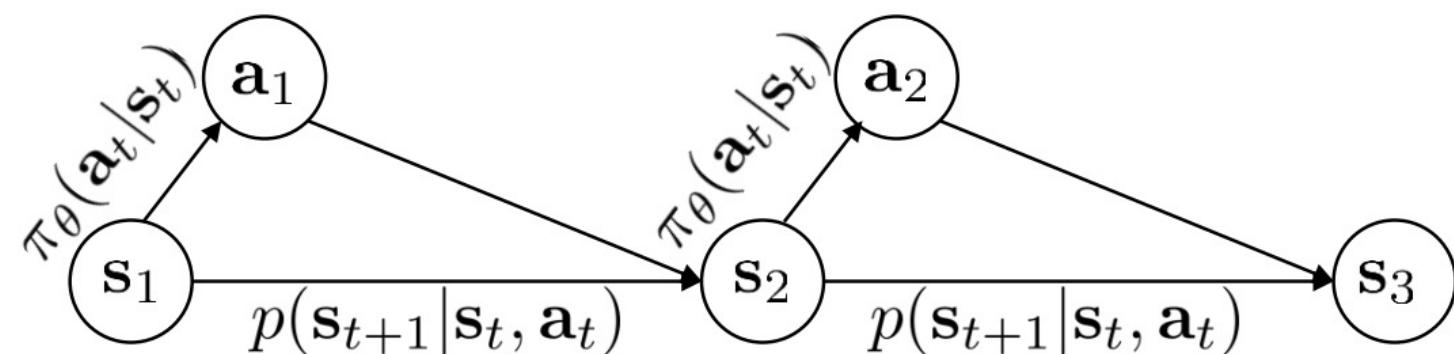
Only have access to samples

Trajectory Probability



Trajectory Probability

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



Reinforcement Learning Objective

- Finite horizon:

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

- Infinite horizon:

$$\begin{aligned} J(\theta) &= E_{(\mathbf{s}, \mathbf{a}) \sim p_\theta(\mathbf{s}, \mathbf{a})} [r(\mathbf{s}, \mathbf{a})] \\ &= \int_{\mathcal{S}} \underline{\mu(s)} \int_{\mathcal{A}} \pi_\theta(a|s) r(s, a) \end{aligned}$$

stationary distribution

The Goal of Reinforcement Learning

- The RL objective

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

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

- The goal of policy gradient:

$$\theta^\star = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Evaluating the Objective

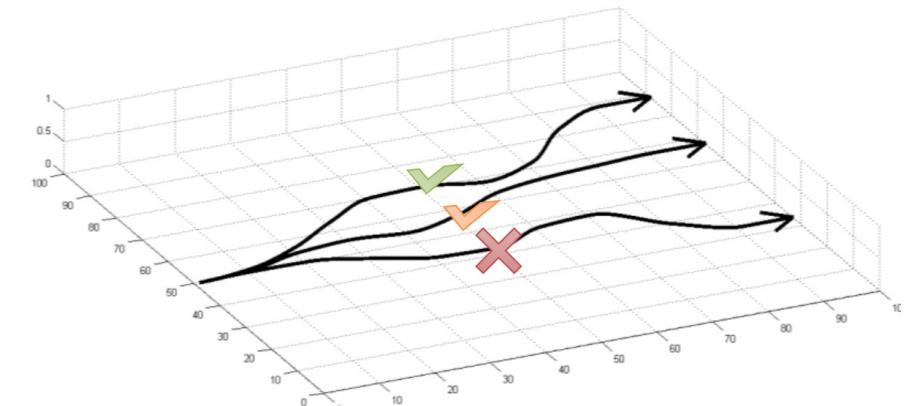
$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$J(\theta)$

Monte Carlo estimation for objective function:

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

sum over samples from π_{θ}



Direct Policy Differentiation

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

$\underbrace{\qquad\qquad\qquad}_{J(\theta)}$

$J(\theta)$

$$J(\theta) = E_{\tau \sim p_{\theta}(\tau)} [\overbrace{r(\tau)}^{\text{episode}}] = \int p_{\theta}(\tau) r(\tau) d\tau$$

$\sum_{t=1}^T r(s_t, a_t)$

$\overbrace{\qquad\qquad\qquad}^{\text{episode}}$

$$\nabla_{\theta} J(\theta) = \int \overbrace{\nabla_{\theta} p_{\theta}(\tau) r(\tau)}_{\theta} d\tau \stackrel{?}{=} \int \overbrace{p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) r(\tau)}_{\theta} d\tau = E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

$P(\cdot | \theta)$

$$\tau = (s_0, a_0, r_0, s_1, a_1, \dots)$$

a convenient identity

$$P_{\theta}(\tau) = \frac{p_{\theta}(s_0) P(s_1 | s_0, a_0)}{p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau)} = \frac{p_{\theta}(\tau) \nabla_{\theta} p_{\theta}(\tau)}{p_{\theta}(\tau)} = \nabla_{\theta} p_{\theta}(\tau)$$

$$\frac{P(s_1 | s_0, a_0)}{P(a_1 | s_1)} \frac{P(s_2 | s_1, a_1)}{\dots}$$

$$\pi_{\theta}(a_1 | s_1) \dots$$

Direct Policy Differentiation

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

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

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

log of both sides

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

$$\nabla_{\theta} \left[\cancel{\log p(\mathbf{s}_1)} + \sum_{t=1}^T \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) + \cancel{\log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)} \right]$$

$$\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

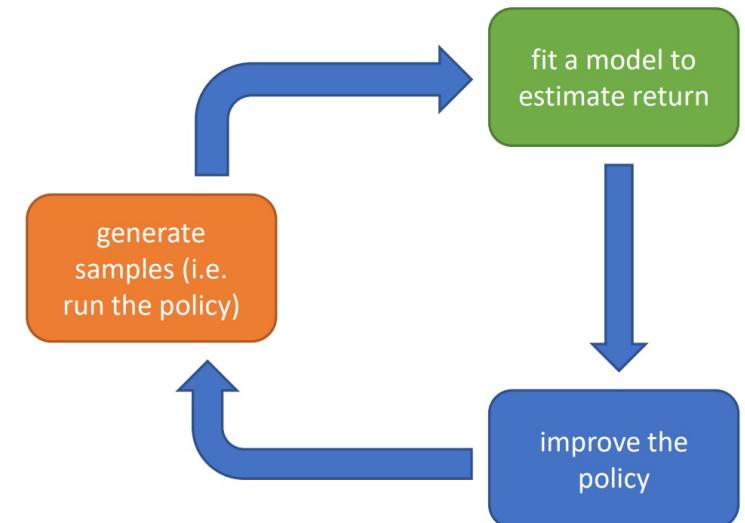
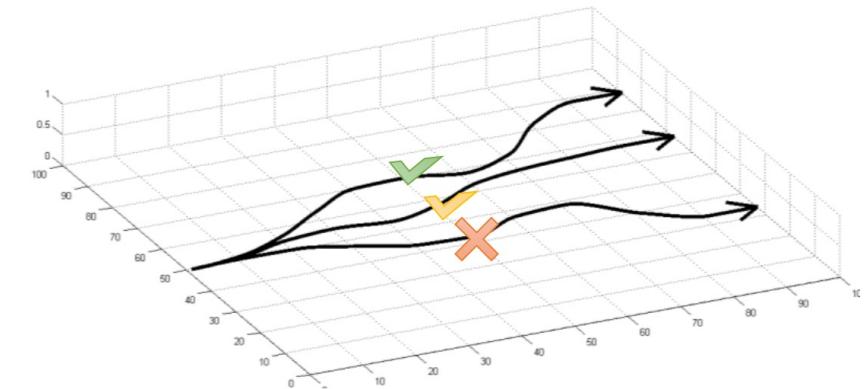
Evaluating the Policy Gradient

recall: $J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

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

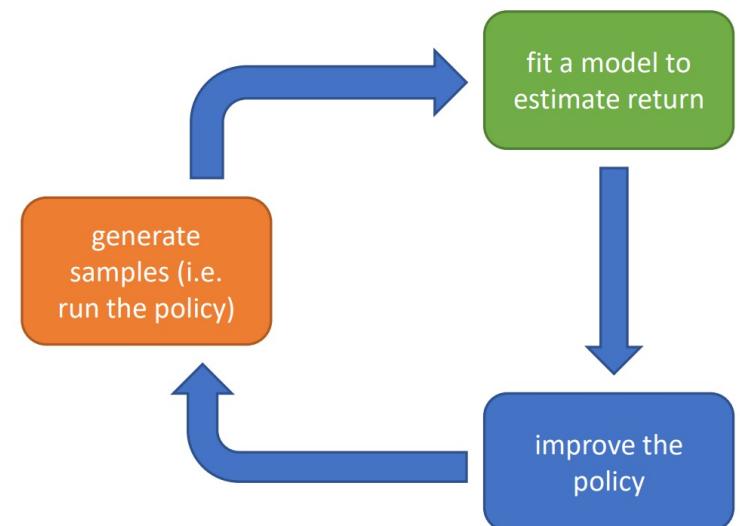
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$



REINFORCE Algorithm

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



Understanding Policy Gradient

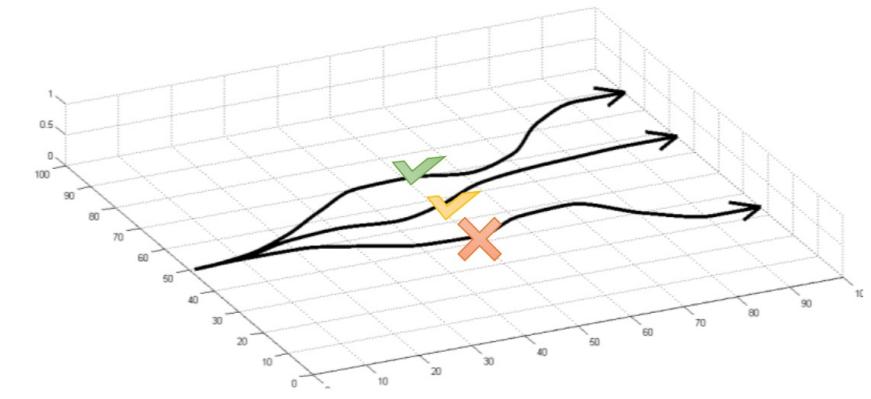
$$\text{recall: } J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

$$\nabla_\theta J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

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

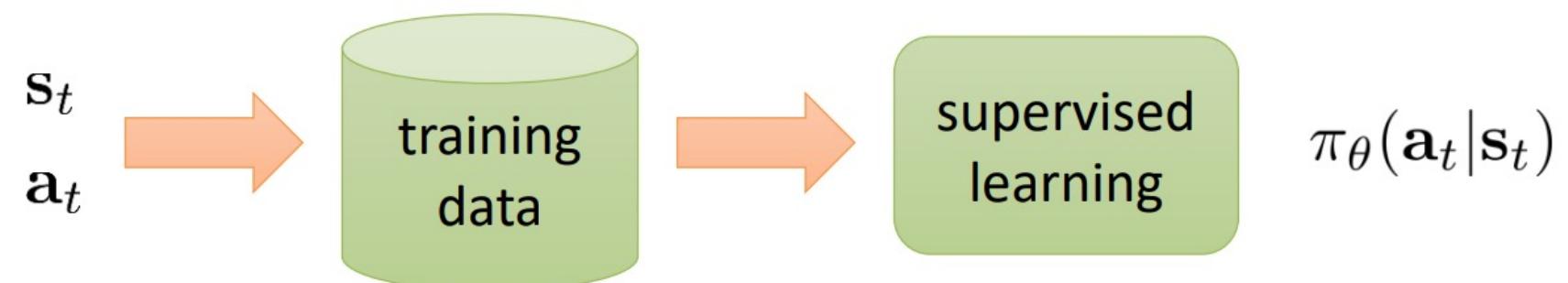


What is this?



Supervised Learning: Behavioral Cloning

Directly learns a policy by **using supervised learning** on observation-action pairs **from expert demonstrations**.



maximum likelihood:
$$\nabla_\theta J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t}) \right)$$

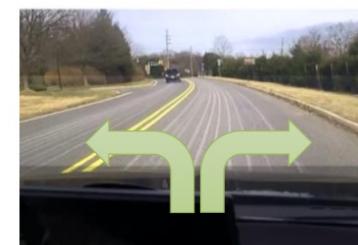
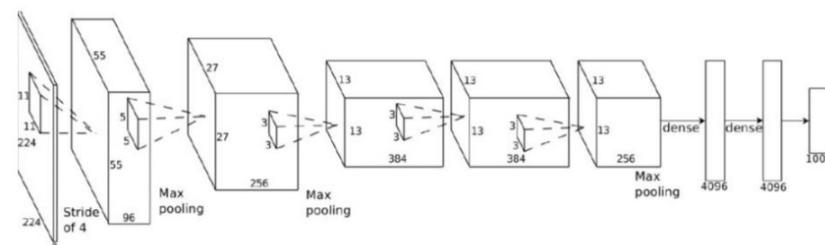
Policy Gradient Vs Maximum Likelihood

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

$$\text{maximum likelihood: } \nabla_{\theta} J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right)$$



\mathbf{s}_t



\mathbf{a}_t



\mathbf{s}_t
 \mathbf{a}_t

training
data

supervised
learning

$\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)$

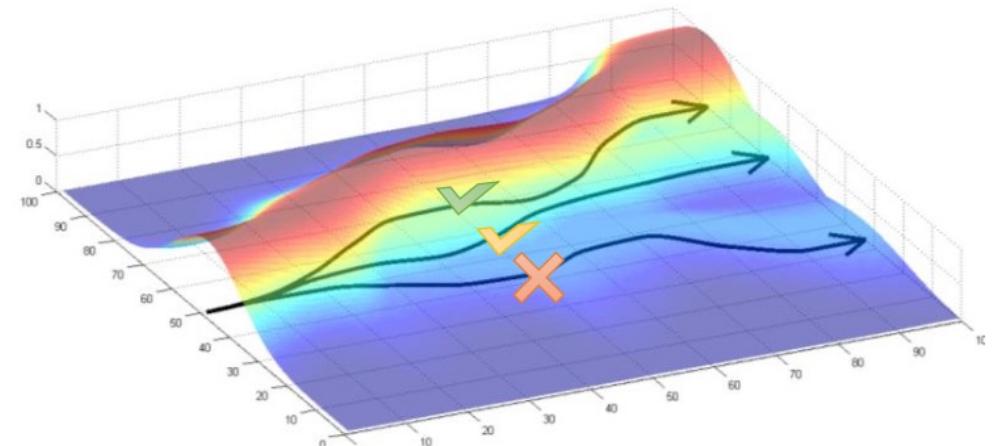
Policy Gradient Intuition

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

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

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”!



Continuous Action Space

Gaussian Policy:

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

example: $\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = \mathcal{N}(f_{\text{neural network}}(\mathbf{s}_t); \Sigma)$

$$\log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = -\frac{1}{2} \|f(\mathbf{s}_t) - \mathbf{a}_t\|_{\Sigma}^2 + \text{const}$$

$$\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = -\frac{1}{2} \Sigma^{-1} (f(\mathbf{s}_t) - \mathbf{a}_t) \frac{df}{d\theta}$$

Bias and Variance of Policy Gradient

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

The main source
of high variance

- Unbiased estimation:

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

- But suffers from **high variance!**

Reducing Variance

- Causality trick
- Discount factor
- Baseline
- Actor-critic
- Optimization techniques:
 - Natural gradient
 - Trust region

Reducing Variance: Causality

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

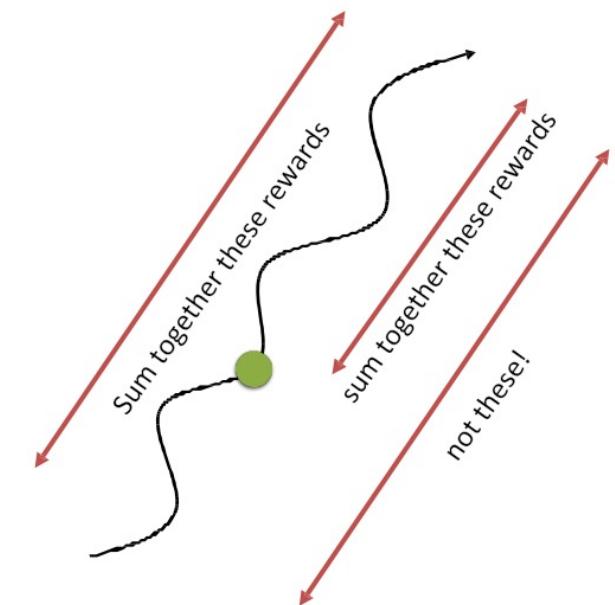
$\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})$
 $\left[\sum_{t'=1}^{t-1} r_{t'} \right]$

Causality: policy at time t' cannot affect reward at time t when $t < t'$

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

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

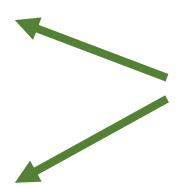
“reward to go” $\hat{Q}_{i,t}$



Reducing Variance: Discount Factor

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

 **Not the same**

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

Reducing Variance: Baselines

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

$$b = \frac{1}{N} \sum_{i=1}^N r(\tau) \quad = \frac{1}{N} \sum \nabla_{\theta} \log p_{\theta}(\tau) r(\tau)$$

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

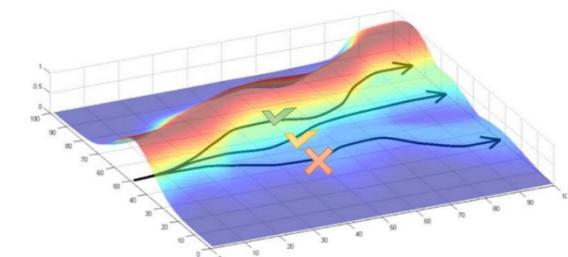
$\stackrel{=0}{\text{in expectation}}$

subtracting a baseline is *unbiased* in expectation!

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

a convenient identity

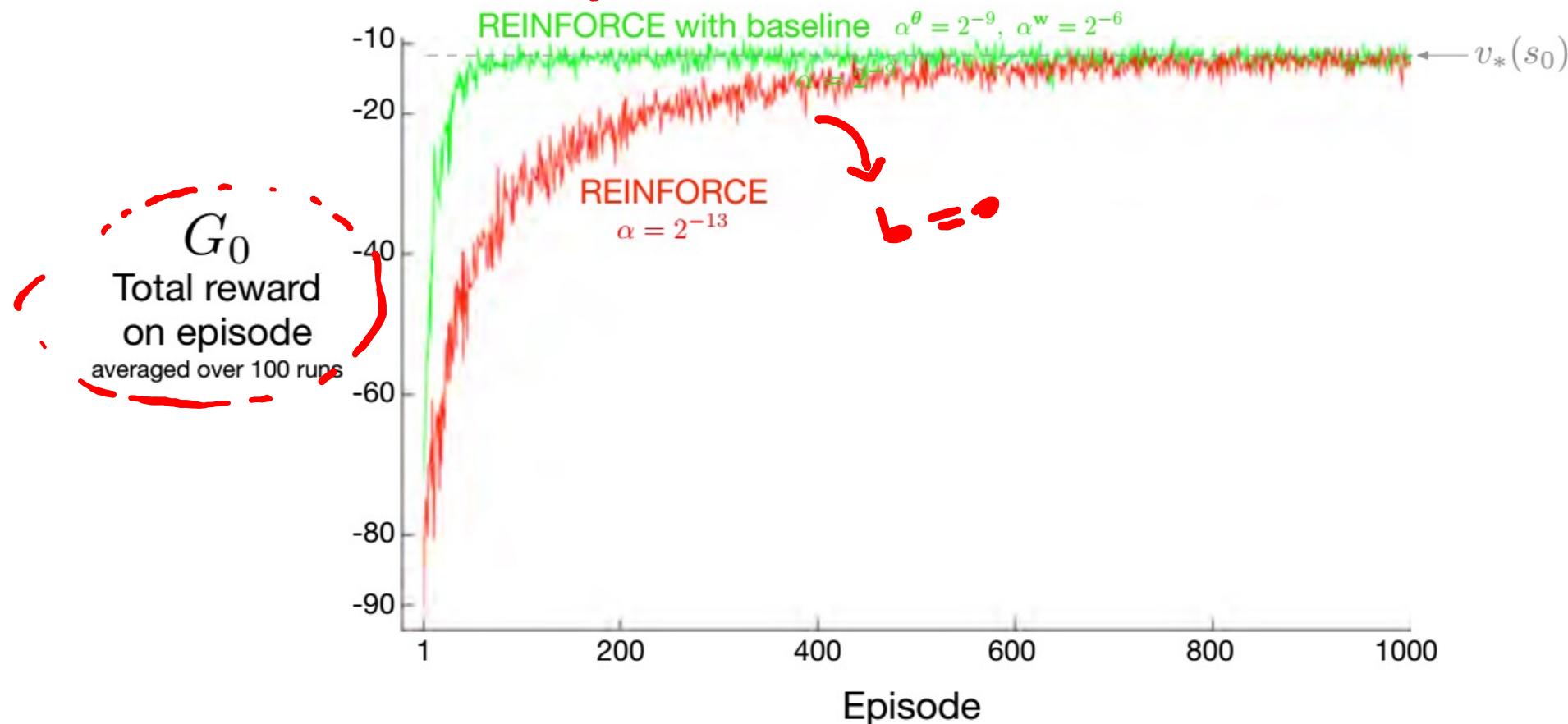
$$p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) = \nabla_{\theta} p_{\theta}(\tau)$$



Reducing Variance: Baselines

Faster convergence:

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



Analyzing Variance

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

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

$$\text{Var} = E_{\tau \sim p_{\theta}(\tau)} [(\underbrace{\nabla_{\theta} \log p_{\theta}(\tau)}_{g(\tau)} (r(\tau) - b))^2] - E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) (r(\tau) - b)]^2$$

this bit is just $E_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$
(baselines are unbiased in expectation)

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

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

This is just expected reward, but weighted by gradient magnitudes!

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!

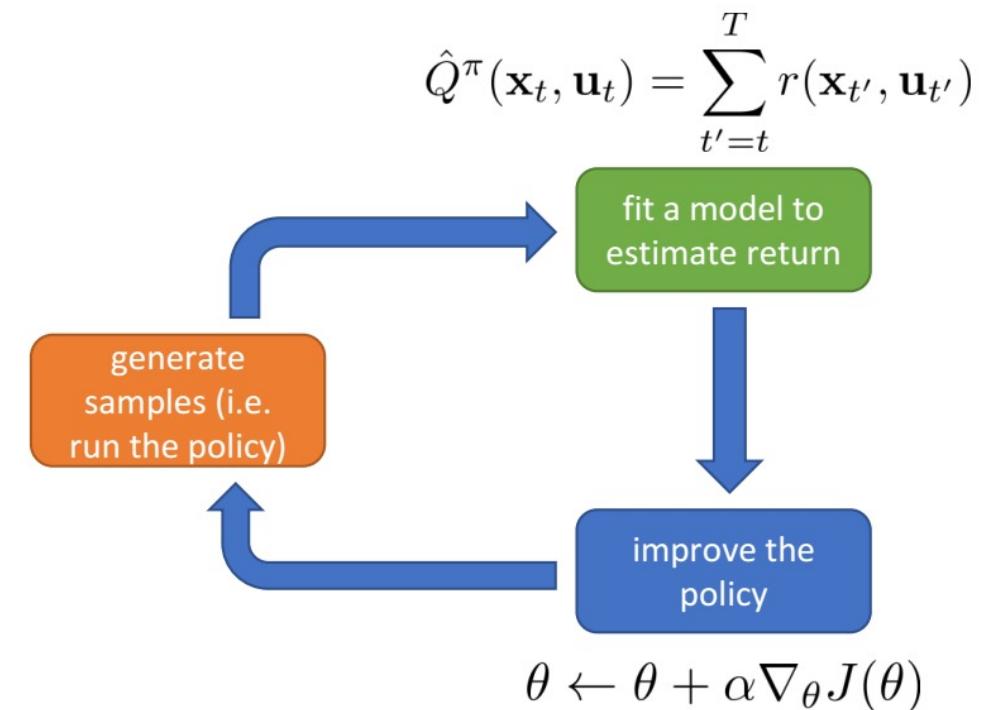
Policy Gradients so Far

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \right) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

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

“reward to go”



Improving Estimation of Reward to Go

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

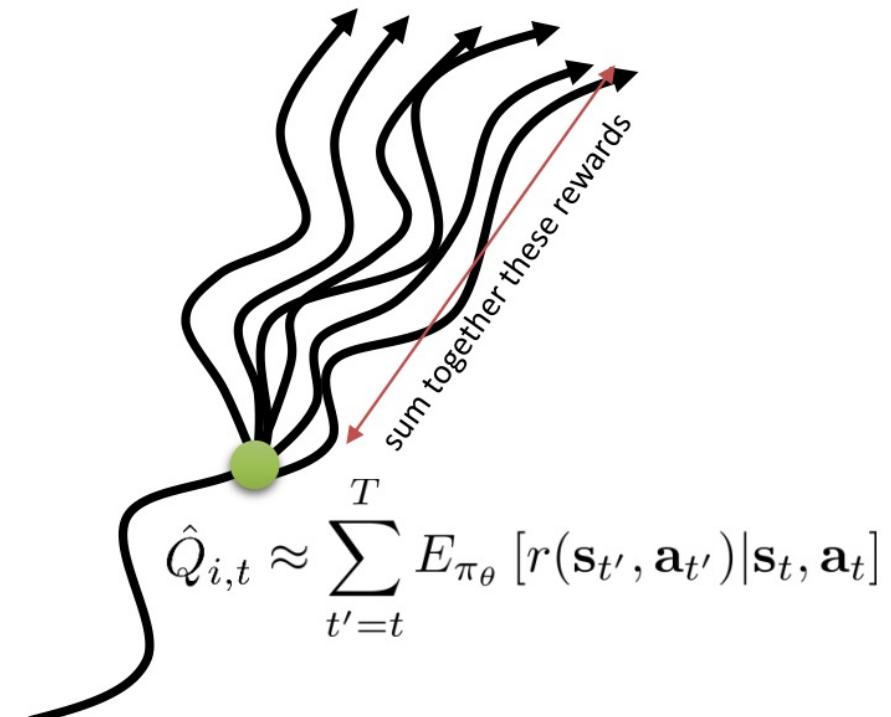
$\hat{Q}_{i,t}$: estimate of expected reward if we take action $\mathbf{a}_{i,t}$ in state $\mathbf{s}_{i,t}$

How to make a better estimate?

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

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

much lower variance!



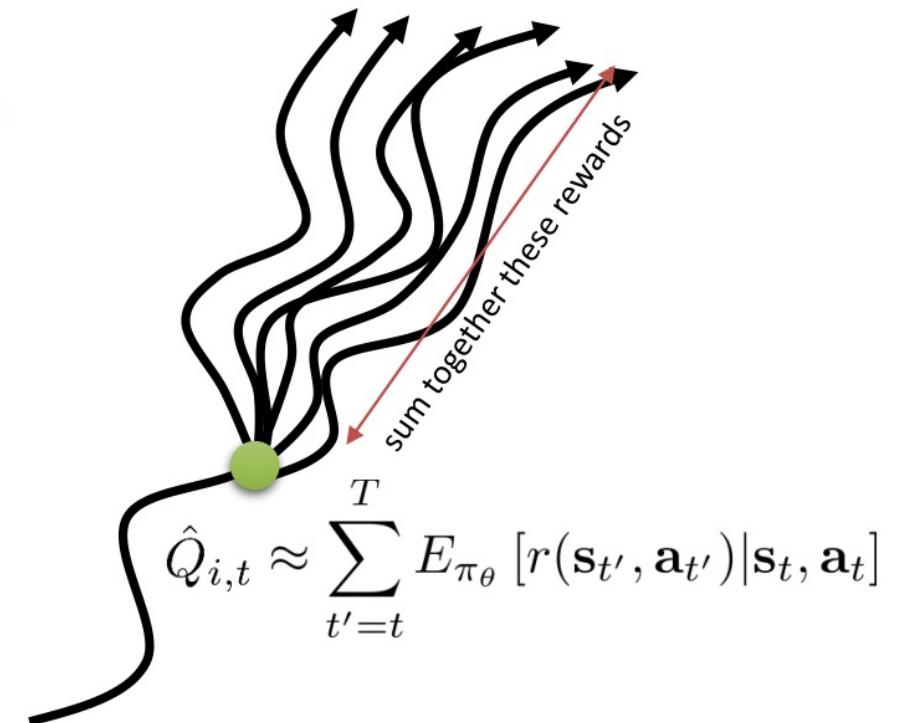
Improving Estimation of Reward to Go

Further improvement: Adding a baseline!

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

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

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



Improving Estimation of Reward to Go

Further improvement: Adding a baseline!

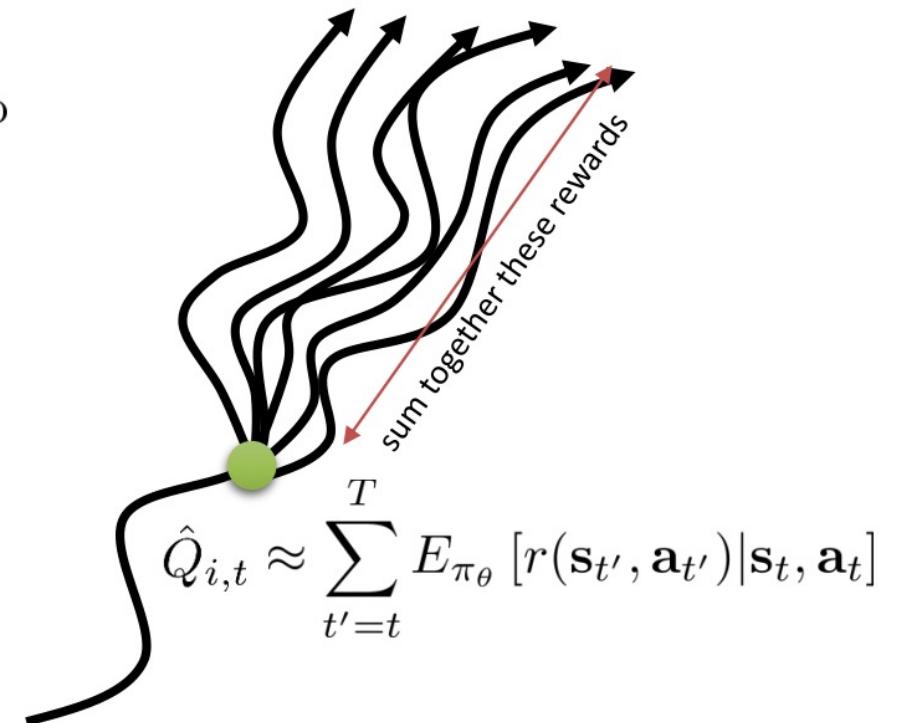
$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$: true *expected* reward-to-go

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

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



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



Advantage Value

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

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

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

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



the better this estimate, the lower the variance

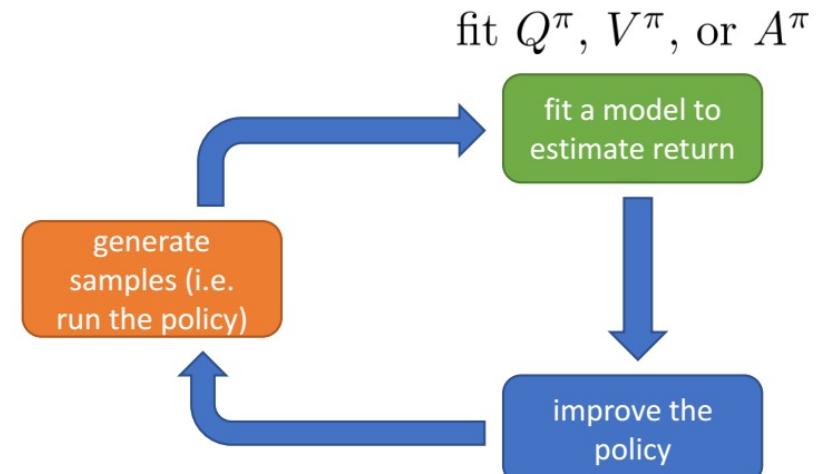
Advantage Value Approximation

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

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

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

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



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

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

$$= r(\mathbf{s}_t, \mathbf{a}_t) + \sum_{t'=t+1}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$$

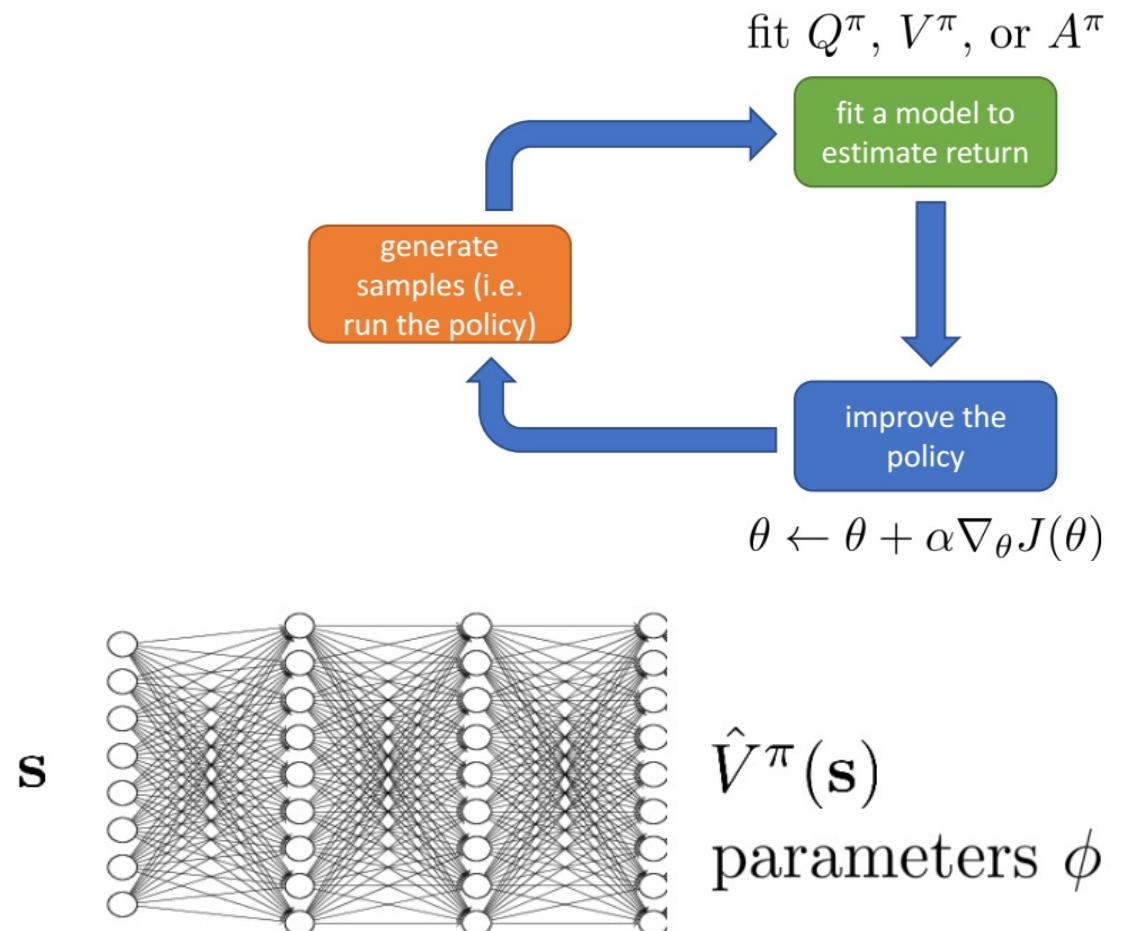
$$\approx V^\pi(\mathbf{s}_{t+1})$$

Advantage Value Approximation

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1})$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$$

let's just fit $V^\pi(\mathbf{s})$!



Policy Evaluation

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

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

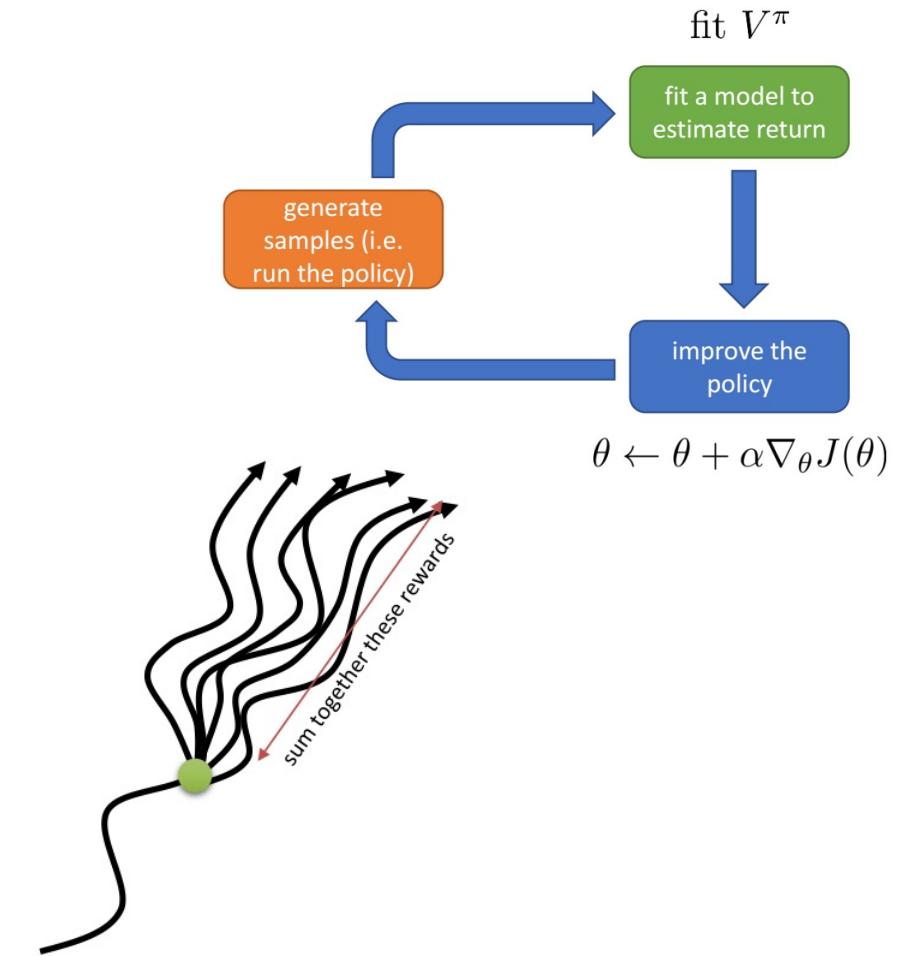
how can we perform policy evaluation?

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

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

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

(requires us to reset the simulator)



Policy Evaluation

Monte Carlo estimation with function approximator:

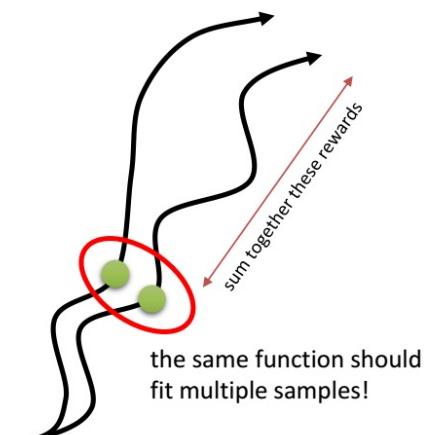
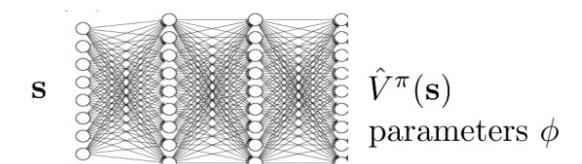
$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

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

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



Policy Evaluation

How to make a better estimate?

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

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

directly use previous fitted value function!

Policy Evaluation

Bootstrap Estimation with Function Approximator

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

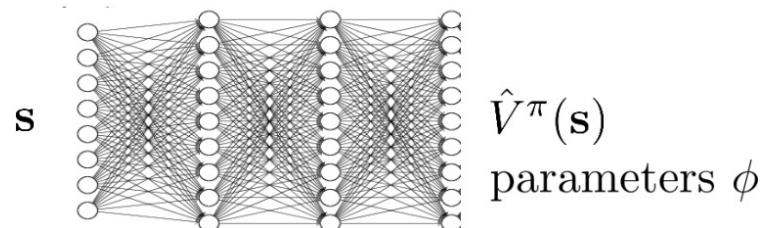
training data: $\left\{ \underbrace{\left(\mathbf{s}_{i,t}, r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1}) \right)}_{y_{i,t}} \right\}$

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

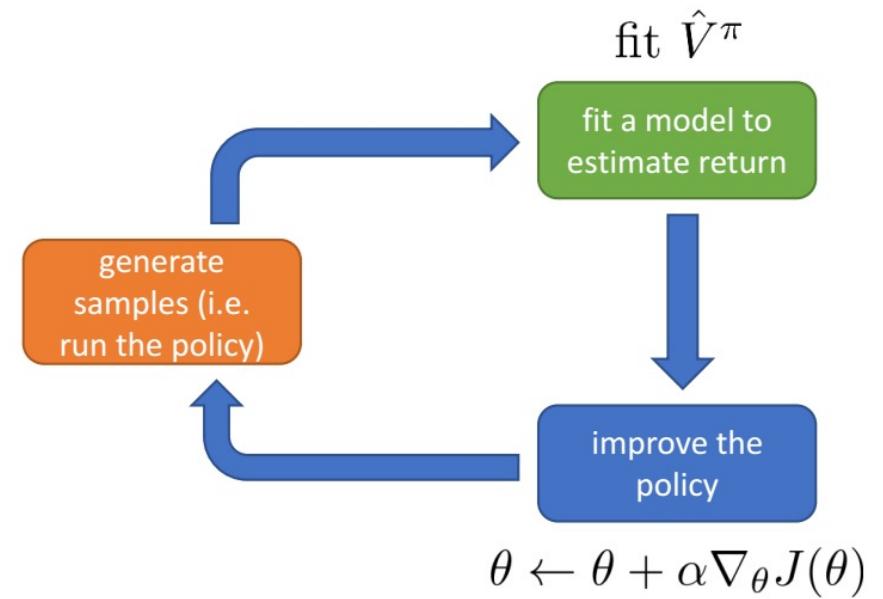
Batch Actor-Critic Algorithm

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 $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



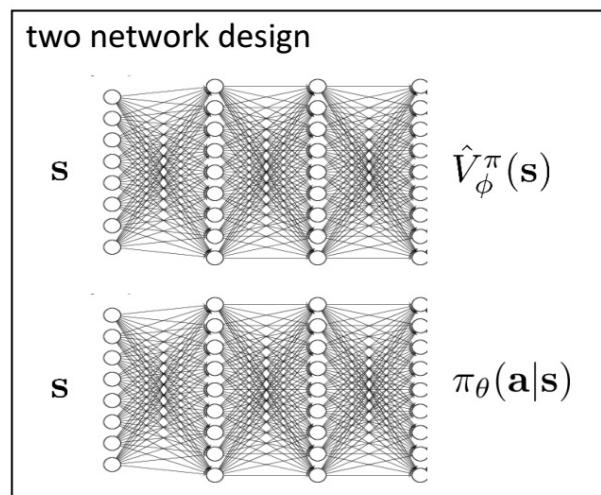
$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^T E_{\pi_\theta} [r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t]$$



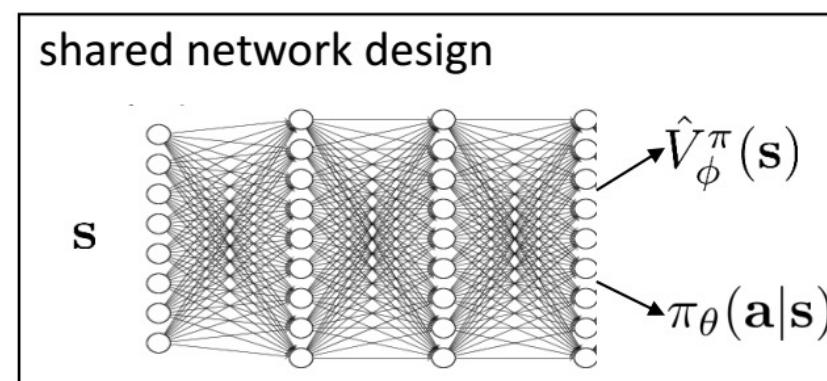
Actor-Critic Algorithm: Architecture Design

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 $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
- 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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



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



Policy Evaluation in Infinite Horizon Settings

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

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

what if T (episode length) is ∞ ?

\hat{V}_ϕ^π can get infinitely large in many cases

simple trick: better to get rewards sooner than later

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

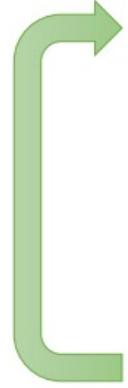
$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}_\phi^\pi(\mathbf{s}_i) - y_i \right\|^2$$

$$\hat{A}^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

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

Actor-Critic Algorithm: Infinite Horizon

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 $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \boxed{\gamma} \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
 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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Critics as 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}_{ϕ}^{π} 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)

Eligibility Traces and N-step Returns

$$\hat{A}_C^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}_\phi^\pi(s_{t+1}) - \hat{V}_\phi^\pi(s_t)$$

$$\hat{A}_{MC}^\pi(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}_\phi^\pi(s_t)$$

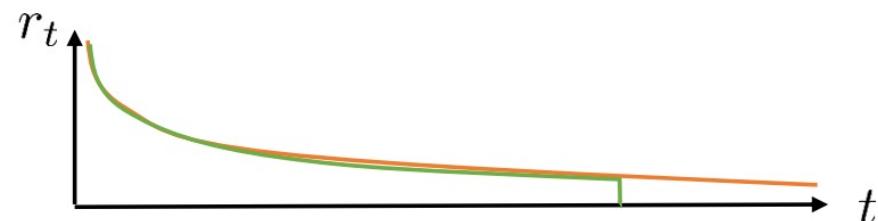
+ lower variance

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

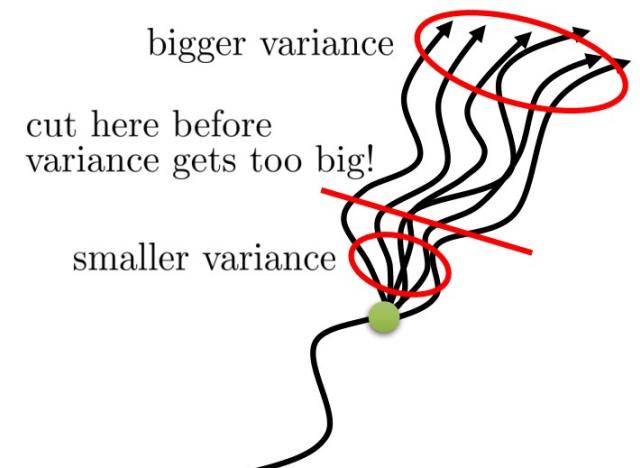
+ no bias

- higher variance (because single-sample estimate)

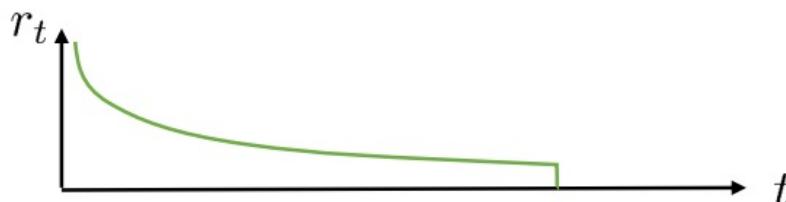
Can we combine these two, to control bias/variance tradeoff?



$$\hat{A}_n^\pi(s_t, a_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(s_{t'}, a_{t'}) - \hat{V}_\phi^\pi(s_t) + \gamma^n \hat{V}_\phi^\pi(s_{t+n})$$



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

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

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

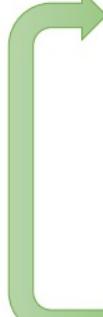
$$\delta_{t'} = r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_{t'+1}) - \hat{V}_\phi^\pi(\mathbf{s}_{t'})$$

Actor-Critic Algorithm: Batch vs. Online

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:

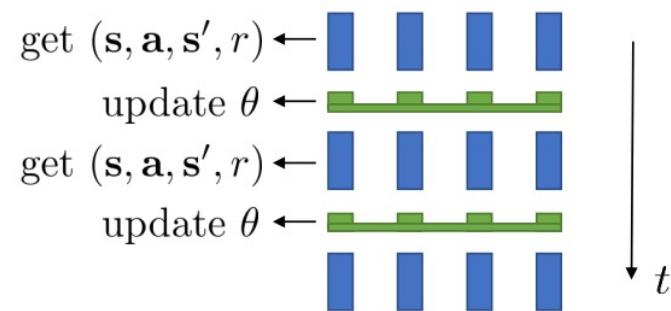
- 
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
 2. update \hat{V}_ϕ^π using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$
 3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}')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

Online Actor-Critic in Practice: A2C and A3C

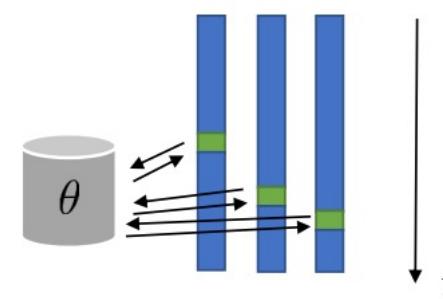
online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
 2. update \hat{V}_ϕ^π using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}') \leftarrow$
 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. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a}) \leftarrow$
 5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
- works best with a batch (e.g., parallel workers)

synchronized parallel actor-critic



asynchronous parallel actor-critic



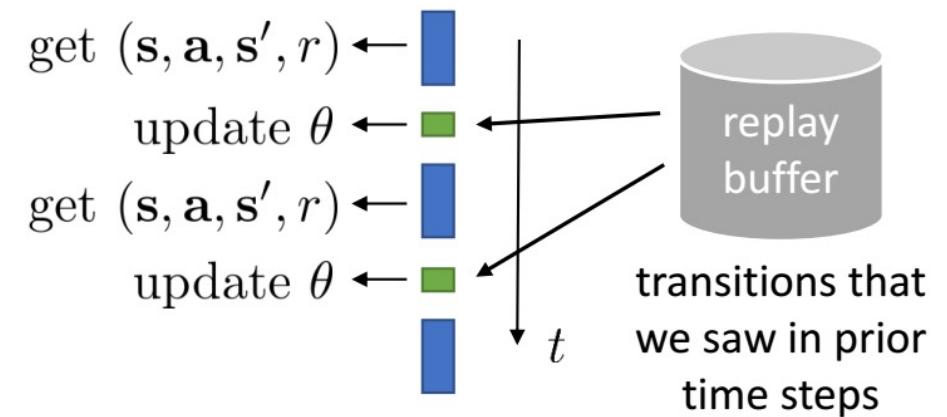
From On-Policy to Off-Policy Actor-Critic

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}_ϕ^π 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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Can we remove the on-policy assumption?

off-policy actor-critic



From On-Policy to Off-Policy Actor-Critic

off-policy 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}_ϕ^π using targets $y_i = r_i + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i)$ for each \mathbf{s}_i
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma V_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



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

not the right target value

not the action π_θ would have taken!

This algorithm is broken!

Can you spot the problems?

Off-Policy Actor-Critic: Fixing the Value Function

off-policy 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}_ϕ^π 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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

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

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

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

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

not the right target value

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

not from replay buffer \mathcal{R} !

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

Off-Policy Actor-Critic: Fixing the Policy Update

off-policy 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}_ϕ^π 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 !

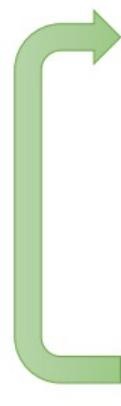
sample $\mathbf{a}_i^\pi \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i) \longrightarrow \nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\underline{\mathbf{a}}_i^\pi|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \underline{\mathbf{a}}_i^\pi)$

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

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

Off-Policy Actor-Critic

off-policy 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}_ϕ^π 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

Deterministic Policy Gradient

Deterministic policy gradient or Q-learning with continuous action?

What is the problem of Q-learning with **continuous action**?

$$\pi(\mathbf{a}_t | \mathbf{s}_t) = \begin{cases} 1 & \text{if } \mathbf{a}_t = \arg \max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{target value } y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$$

How do we perform the max operation?!

Deterministic Policy Gradient

learn an approximate maximizer

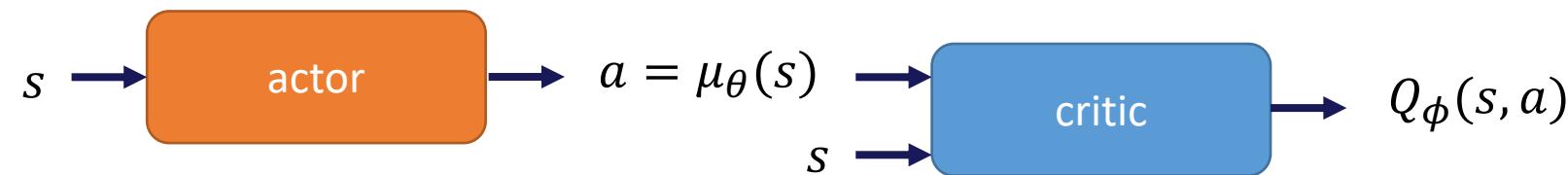
$$\max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a}) = Q_\phi(\mathbf{s}, \arg \max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a}))$$

idea: train another network $\mu_\theta(\mathbf{s})$ such that $\mu_\theta(\mathbf{s}) \approx \arg \max_{\mathbf{a}} Q_\phi(\mathbf{s}, \mathbf{a})$

how? just solve $\theta \leftarrow \arg \max_\theta Q_\phi(\mathbf{s}, \mu_\theta(\mathbf{s}))$

$$\frac{dQ_\phi}{d\theta} = \frac{d\mathbf{a}}{d\theta} \frac{dQ_\phi}{d\mathbf{a}}$$

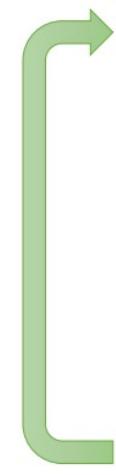
$$\text{new target } y_j = r_j + \gamma Q_{\phi'}(\mathbf{s}'_j, \mu_\theta(\mathbf{s}'_j)) \approx r_j + \gamma Q_{\phi'}(\mathbf{s}'_j, \arg \max_{\mathbf{a}'} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j))$$



Deterministic Policy Gradient

learn an approximate maximizer

DDPG:

- 
1. take some action \mathbf{a}_i and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to \mathcal{B}
 2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from \mathcal{B} uniformly
 3. compute $y_j = r_j + \gamma Q_{\phi'}(\mathbf{s}'_j, \mu_{\theta'}(\mathbf{s}'_j))$ using *target nets* $Q_{\phi'}$ and $\mu_{\theta'}$
 4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
 5. $\theta \leftarrow \theta + \beta \sum_j \frac{d\mu}{d\theta}(\mathbf{s}_j) \frac{dQ_\phi}{d\mathbf{a}}(\mathbf{s}_j, \mu(\mathbf{s}_j))$
 6. update ϕ' and θ'
- off-policy updates!**

Deterministic Policy Gradient

off-policy policy gradient:

$$\begin{aligned}\nabla_{\theta} J_{\beta}(\pi_{\theta}) &\approx \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^{\beta}(s) \nabla_{\theta} \pi_{\theta}(a|s) Q^{\pi}(s, a) da ds \\ &= \mathbb{E}_{s \sim \rho^{\beta}, a \sim \beta} \left[\frac{\pi_{\theta}(a|s)}{\underline{\beta_{\theta}(a|s)}} \nabla_{\theta} \log \pi_{\theta}(a|s) Q^{\pi}(s, a) \right]\end{aligned}$$

off-policy deterministic policy gradient:

$$\begin{aligned}\nabla_{\theta} J_{\beta}(\mu_{\theta}) &\approx \int_{\mathcal{S}} \rho^{\beta}(s) \nabla_{\theta} \mu_{\theta}(a|s) Q^{\mu}(s, a) ds \\ &= \mathbb{E}_{s \sim \rho^{\beta}} \left[\nabla_{\theta} \mu_{\theta}(s) \left. \nabla_a Q^{\mu}(s, a) \right|_{a=\mu_{\theta}(s)} \right]\end{aligned}$$

Actor-Critic: Review

- Actor-critic algorithms
 - Actor: policy
 - Critic: value function
- Policy evaluation: fitting value function
- Infinite horizon: discount factor
- Actor-critic design
- Critics as baselines
- Generalized advantage estimation
- Bias–variance tradeoff
- Batch mode or online mode
- On-policy or off-policy
- DPG and DDPG

Off-Policy Policy Gradients

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

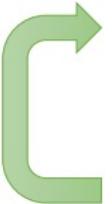
$$J(\theta) = E_{\tau \sim p_\theta(\tau)}[r(\tau)]$$

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

This is the problem!

- 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) ← Can't just skip this!
 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. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

Policy Gradients Objective with 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)

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

importance weight

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

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

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 Importance Sampling

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

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

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' ?

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

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

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

Off-Policy Policy Gradient

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

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

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

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

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

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

future actions don't affect current weight

ignore this part!

Off-Policy Policy Gradient: First Order Approximation

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

Approximation:

$$\begin{aligned} \nabla_{\theta'} J(\theta') &\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\pi_{\theta'}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})}{\pi_{\theta}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t} & (\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \sim \pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t) \\ &= \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \cancel{\frac{\pi_{\theta'}(\mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{s}_{i,t})}} \frac{\pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})}{\pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t} \end{aligned}$$

ignore this part!

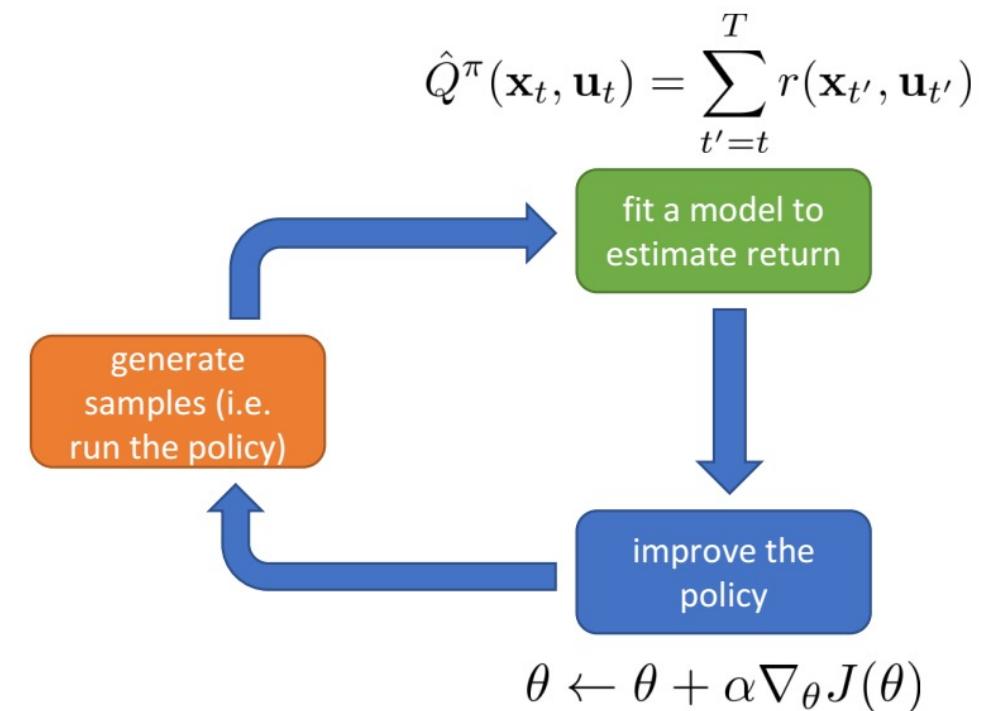
Recap: Policy Gradients

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \left(\sum_{t'=t}^T r(\mathbf{s}_{t'}^i, \mathbf{a}_{t'}^i) \right) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

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

“reward to go”

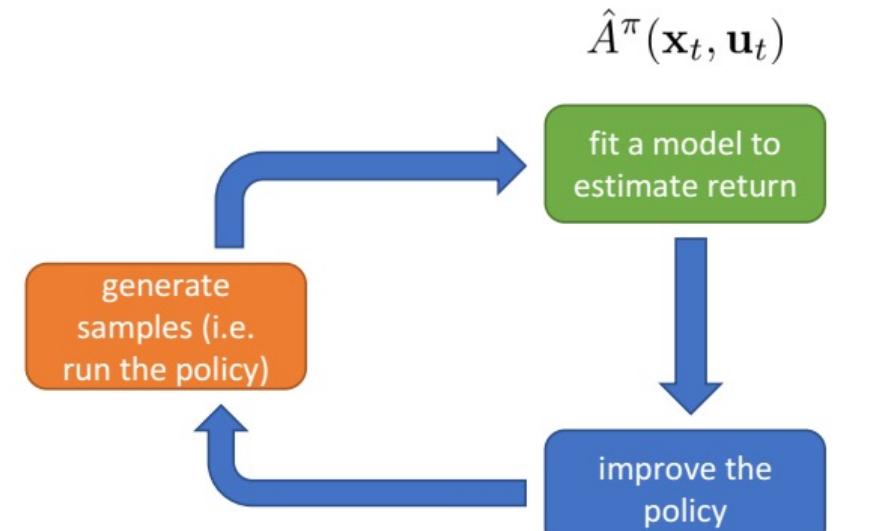


Policy Gradient as Policy Iteration

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

main steps of policy gradient algorithm:

- 
1. Estimate $\hat{A}^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ for current policy π
 2. Use $\hat{A}^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ to get *improved* policy π'



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

Familiar to policy iteration algorithm:

- 
1. evaluate $A^{\pi}(\mathbf{s}, \mathbf{a})$
 2. set $\pi \leftarrow \pi'$

Policy Gradient as Policy Iteration

$$J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\sum_t \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

claim: $J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right]$

could be interpret as policy improvement!

Policy Gradient as Policy Iteration

claim: $J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right]$

proof:
$$\begin{aligned} J(\theta') - J(\theta) &= J(\theta') - E_{\mathbf{s}_0 \sim p(\mathbf{s}_0)} [V^{\pi_\theta}(\mathbf{s}_0)] \\ &= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)} [V^{\pi_\theta}(\mathbf{s}_0)] \\ &= J(\theta') - E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(\mathbf{s}_t) \right] \\ &= J(\theta') + E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t)) \right] \\ &= E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] + E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t)) \right] \\ &= E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t (r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V^{\pi_\theta}(\mathbf{s}_{t+1}) - V^{\pi_\theta}(\mathbf{s}_t)) \right] \\ &= E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \end{aligned}$$

Policy Gradient as Policy Iteration

$$J(\theta') - J(\theta) = E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

↑
expectation under $\pi_{\theta'}$ ↑
advantage under π_θ

$$\begin{aligned} E_{\tau \sim p_{\theta'}(\tau)} \left[\sum_t \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] &= \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)} \left[\gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \\ &= \sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \left[\underbrace{\frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)}}_{} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \end{aligned}$$

↑
is it OK to use $p_\theta(\mathbf{s}_t)$ instead?

Policy Gradient as Policy Iteration

Can we ignore distribution mismatch?

$$\sum_t E_{\mathbf{s}_t \sim p_{\theta'}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \stackrel{?}{\approx} \underbrace{\sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]}_{\bar{A}(\theta')}$$

why do we want this to be true?

$$J(\theta') - J(\theta) \approx \bar{A}(\theta') \Rightarrow \theta' \leftarrow \arg \max_{\theta'} \bar{A}(\theta')$$

2. Use $\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)$ to get *improved* policy π'

is it true? and when?

$p_\theta(\mathbf{s}_t)$ is *close* to $p_{\theta'}(\mathbf{s}_t)$ when π_θ is *close* to $\pi_{\theta'}$

Some Useful Preliminaries: Taylor Series

Approximation of a differentiable function around a given point with sum of terms of the function's derivatives:

$$f(x) \approx f(x_0) + (x - x_0)^T \nabla f(x_0) + \frac{1}{2} (x - x_0)^T H(x - x_0) + \dots$$

Some Useful Preliminaries: Constrained Optimization

- equality constraints: method of Lagrange multipliers

$$\begin{aligned} & \text{optimize } f(x) \\ & \text{subject to: } g(x) = 0 \end{aligned} \quad \longrightarrow \quad \mathcal{L}(x, \lambda) = f(x) + \lambda g(x)$$

- Inequality constraints: KKT

$$\begin{aligned} & \text{optimize } f(x) \\ & \text{subject to:} \\ & \quad g_i(x) \leq 0, \\ & \quad h_j(x) = 0 \end{aligned} \quad \longrightarrow \quad \begin{aligned} & \mathcal{L}(x, \mu, \lambda) = f(x) + \mu^T g(x) + \lambda^T h(x) \\ & \text{subject to:} \\ & \quad \mu_i \geq 0, \\ & \quad \mu^T g(x) = 0 \end{aligned}$$

Some Useful Preliminaries: KL-Divergence

A Common distance measure for distributions:

$$D_{KL}(p||q) = \int_x p(x)\log\frac{p(x)}{q(x)}dx$$

Other useful distance measures:

- Total variation distance
- Wasserstein distance
- Jensen–Shannon divergence
- ...

Some Useful Preliminaries: Fisher Information

likelihood function: $p_\theta(x)$

score function: $\nabla_\theta \log p_\theta(x)$

Fisher information: measuring the amount of information that a random variable (x) carries about likelihood parameters (θ):

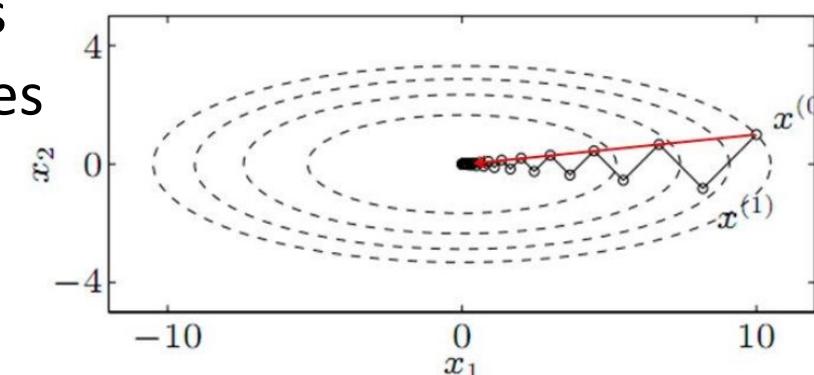
$$\begin{aligned}[I(\theta)]_{i,j} &= E_{x \sim p_\theta(x)} \left[\left(\frac{\partial}{\partial \theta_i} \log p_\theta(x) \right) \left(\frac{\partial}{\partial \theta_j} \log p_\theta(x) \right) \right] \\ &= -E_{x \sim p_\theta(x)} \left[\frac{\partial^2}{\partial \theta_i \partial \theta_j} \log p_\theta(x) \right]\end{aligned}$$

variance (covariance)
of score function

curvature of score function

Importance of step size in RL

- Supervised learning: Step too far \rightarrow next updates will fix it
- Reinforcement learning: Policy is determining data collection!
 - Step too far \rightarrow bad policy
 - Next batch: collected under bad policy
 - May not be able to recover from a bad choice, collapse in performance!
- Learning rate tuning is hard
 - Poor conditioning could be more dangerous in RL settings
 - More sophisticated optimizers can reduce numerical issues
 - Need for advanced learning rate adjustment methods!



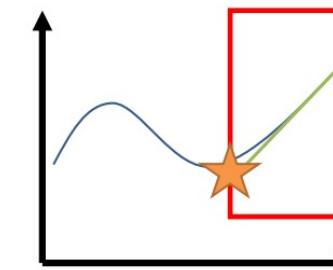
Natural Policy Gradient

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

Could be shown as a constraint problem:

$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta) \text{ s.t. } \underline{\|\theta' - \theta\|^2 \leq \epsilon}$$

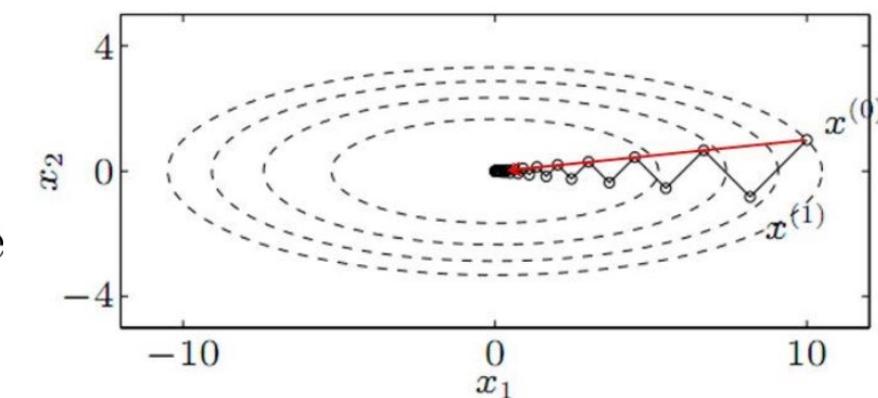
controls how far we go



Can we rescale the gradients to reduce the problem with poor conditioning?

$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta) \text{ s.t. } \underline{D(\pi_{\theta'}, \pi_{\theta}) \leq \epsilon}$$

parameterization-independent divergence measure



Natural Policy Gradient

$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta) \text{ s.t. } D(\pi_{\theta'}, \pi_{\theta}) \leq \epsilon$$

parameterization-independent divergence measure

usually KL-divergence: $D_{\text{KL}}(\pi_{\theta'} \| \pi_{\theta}) = E_{\pi_{\theta'}} [\log \pi_{\theta} - \log \pi_{\theta'}]$

Taylor expansion:

$$D_{\text{KL}}(\pi_{\theta'} \| \pi_{\theta}) \approx (\theta' - \theta)^T \underline{\mathbf{F}} (\theta' - \theta)$$

Fisher-information matrix

$$\mathbf{F} = E_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}|\mathbf{s}) \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}|\mathbf{s})^T]$$



can estimate with samples

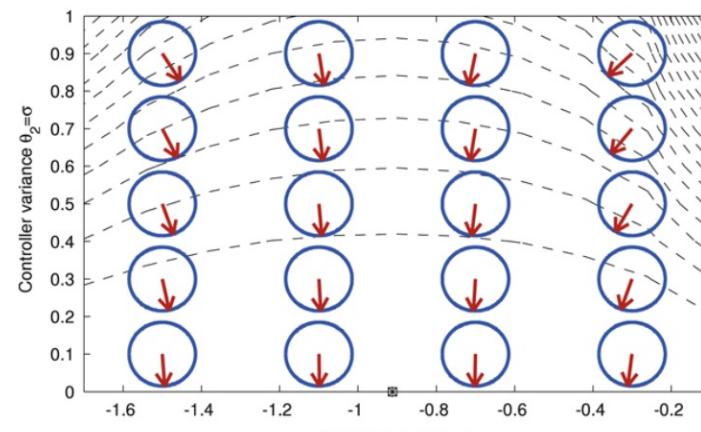
Natural Policy Gradient

$$D_{\text{KL}}(\pi_{\theta'} \parallel \pi_{\theta}) \approx (\theta' - \theta)^T \mathbf{F}(\theta' - \theta)$$

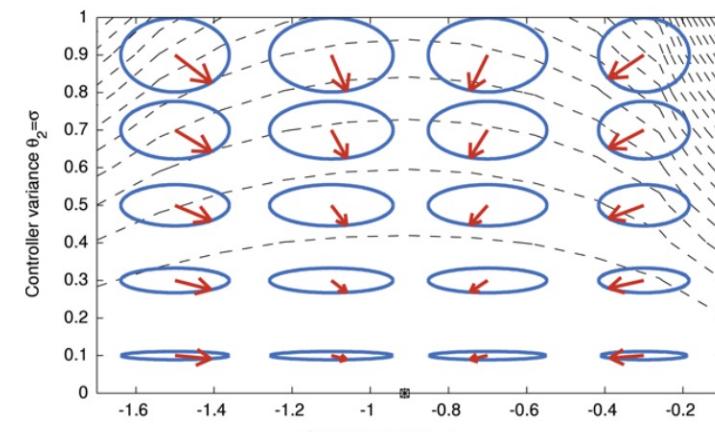
$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta) \text{ s.t. } D(\pi_{\theta'}, \pi_{\theta}) \leq \epsilon$$

$$\theta \leftarrow \theta + \alpha \mathbf{F}^{-1} \nabla_{\theta} J(\theta)$$

natural gradient: pick α



(a) Vanilla policy gradient.



(b) Natural policy gradient.

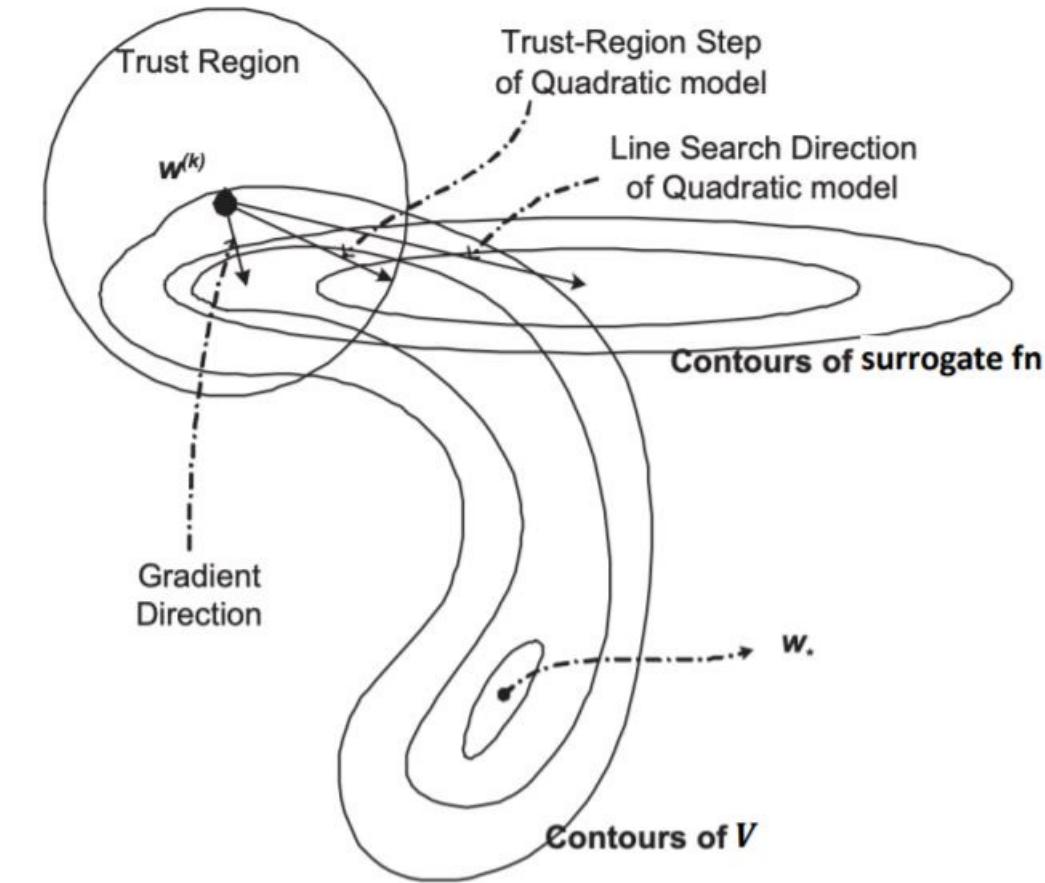
trust region policy optimization: pick ϵ

can solve for optimal α while solving $\mathbf{F}^{-1} \nabla_{\theta} J(\theta)$

(figure from Peters & Schaal 2008)

Trust Region Method

- We often optimize a surrogate objective
- Surrogate objective may be trustable only in a small region
- Limit search to small trust region



Cs885, waterloo 2022

Trust Region Policy Optimization

Recall from “Policy Gradient as Policy Iteration”:

$$\theta' \leftarrow \arg \max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

$\bar{A}(\theta')$

such that $D_{\text{KL}}(\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)) \leq \epsilon$

surrogate loss

trust region

for small enough ϵ , this is guaranteed to improve $J(\theta') - J(\theta)$

Policy Gradient with Constraints

How do we enforce the constraint?

$$\theta' \leftarrow \arg \max_{\theta'} \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

such that $D_{\text{KL}}(\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)) \leq \epsilon$

$$\mathcal{L}(\theta', \lambda) = \sum_t E_{\mathbf{s}_t \sim p_\theta(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] - \lambda(D_{\text{KL}}(\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)) - \epsilon)$$

1. Maximize $\mathcal{L}(\theta', \lambda)$ with respect to θ'
2. $\lambda \leftarrow \lambda + \alpha(D_{\text{KL}}(\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \| \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)) - \epsilon)$

Intuition: raise λ if constraint violated too much, else lower it
an instance of *dual gradient descent*

Natural Gradient Based on Trust Region

Recall:

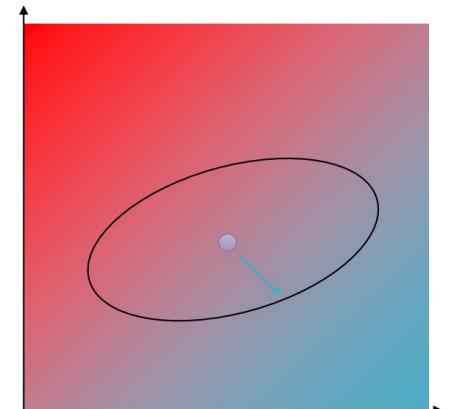
$$\theta' \leftarrow \arg \max_{\theta'} \nabla_{\theta} J(\theta)^T (\theta' - \theta)$$

such that $D_{\text{KL}}(\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \| \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)) \leq \epsilon$

$$D_{\text{KL}}(\pi_{\theta'} \| \pi_{\theta}) \approx \frac{1}{2} (\theta' - \theta)^T \mathbf{F} (\theta' - \theta)$$

$$\theta' = \theta + \alpha \mathbf{F}^{-1} \nabla_{\theta} J(\theta)$$

natural gradient



$$\alpha = \sqrt{\frac{2\epsilon}{\nabla_{\theta} J(\theta)^T \mathbf{F} \nabla_{\theta} J(\theta)}}$$

Proximal Policy Optimization

- TRPO is conceptually and computationally challenging in large part because of the constraint in the optimization.

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

- What is the effect of the constraint?
- Recall KL-Divergence:

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

We are effectively constraining the ratio $\frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}$

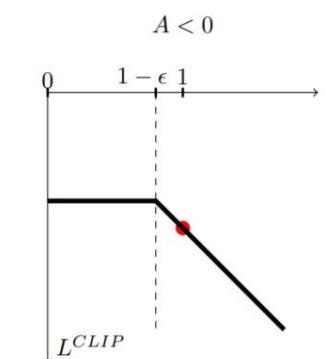
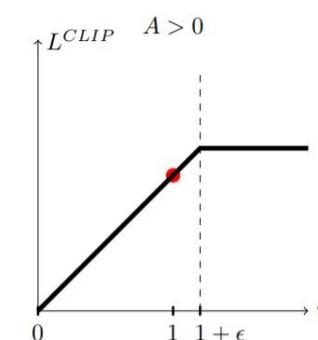
Proximal Policy Optimization

- Let's design a simpler objective that directly constrains $\frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}$

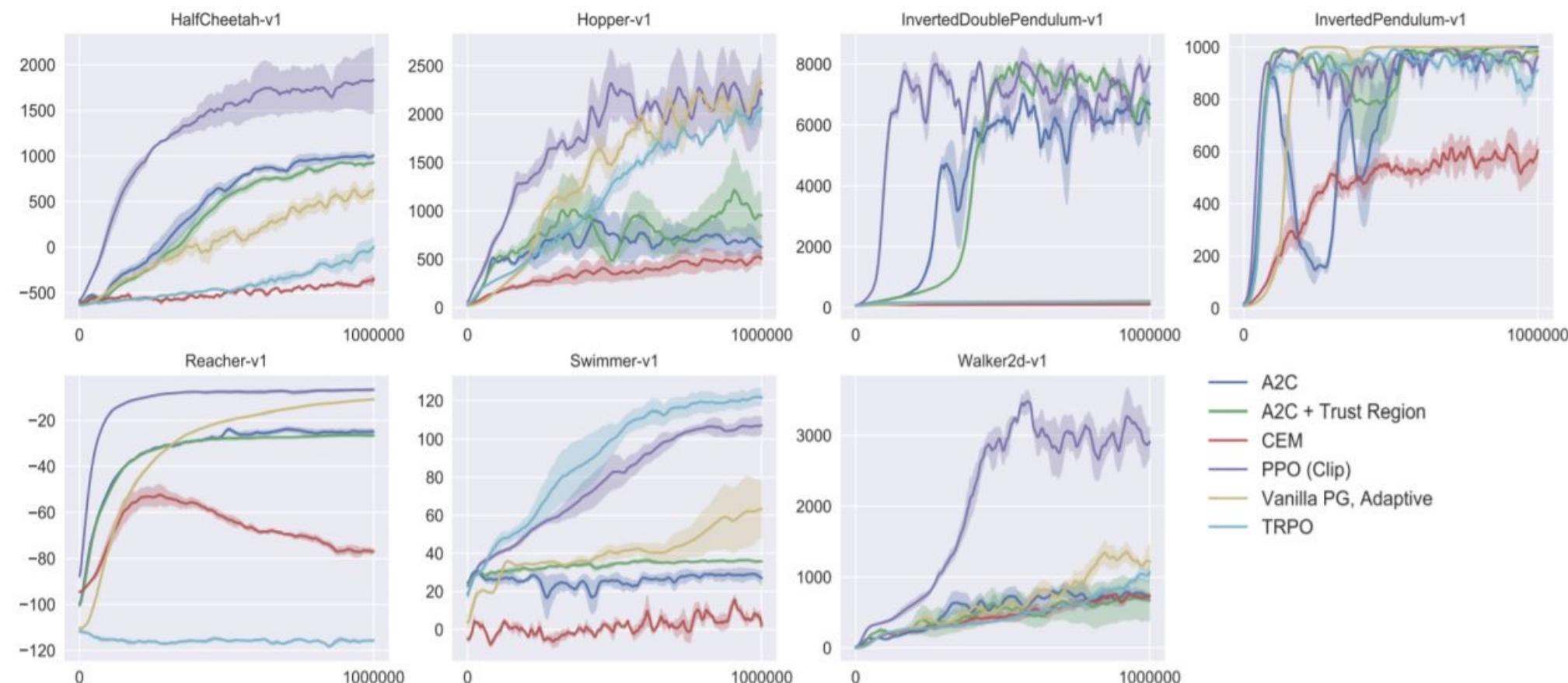
$$\operatorname{argmax}_{\theta'} E_{\{s \sim \mu_{\theta}, a \sim \pi_{\theta}\}} \min \left\{ \begin{array}{l} \frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)} A^{\pi_{\theta}}(s, a), \\ \text{clip}\left(\frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)}, 1 - \epsilon, 1 + \epsilon\right) A^{\pi_{\theta}}(s, a) \end{array} \right.$$

could be easily implemented
with auto-diff packages

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



PPO vs TRPO



Review

- Policy gradient = policy iteration
- Optimize advantage under new policy state distribution
- Using old policy state distribution optimizes a bound, if the policies are close enough
- Results in constrained optimization problem
- Gradient ascent: first order approximation to objective
- Regular gradient ascent has the wrong constraint (in parameter space): use natural gradient with D_{KL} constraint
- PPO simply uses the importance sampling ratio for regularization

Policy Gradient in Practice

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \underbrace{\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}}_{\text{pretty inefficient to compute these explicitly!}}$$

How can we compute policy gradients with automatic differentiation?

We need a graph such that its gradient is the policy gradient!

Implementing Policy Gradient

How can we compute policy gradients with automatic differentiation?

We need a graph such that its gradient is the policy gradient!

maximum likelihood:

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

Just implement “pseudo-loss” as a weighted maximum likelihood:

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

 cross entropy (discrete) or squared error (Gaussian)

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](#)

Related Papers

- 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](#)
- 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](#)