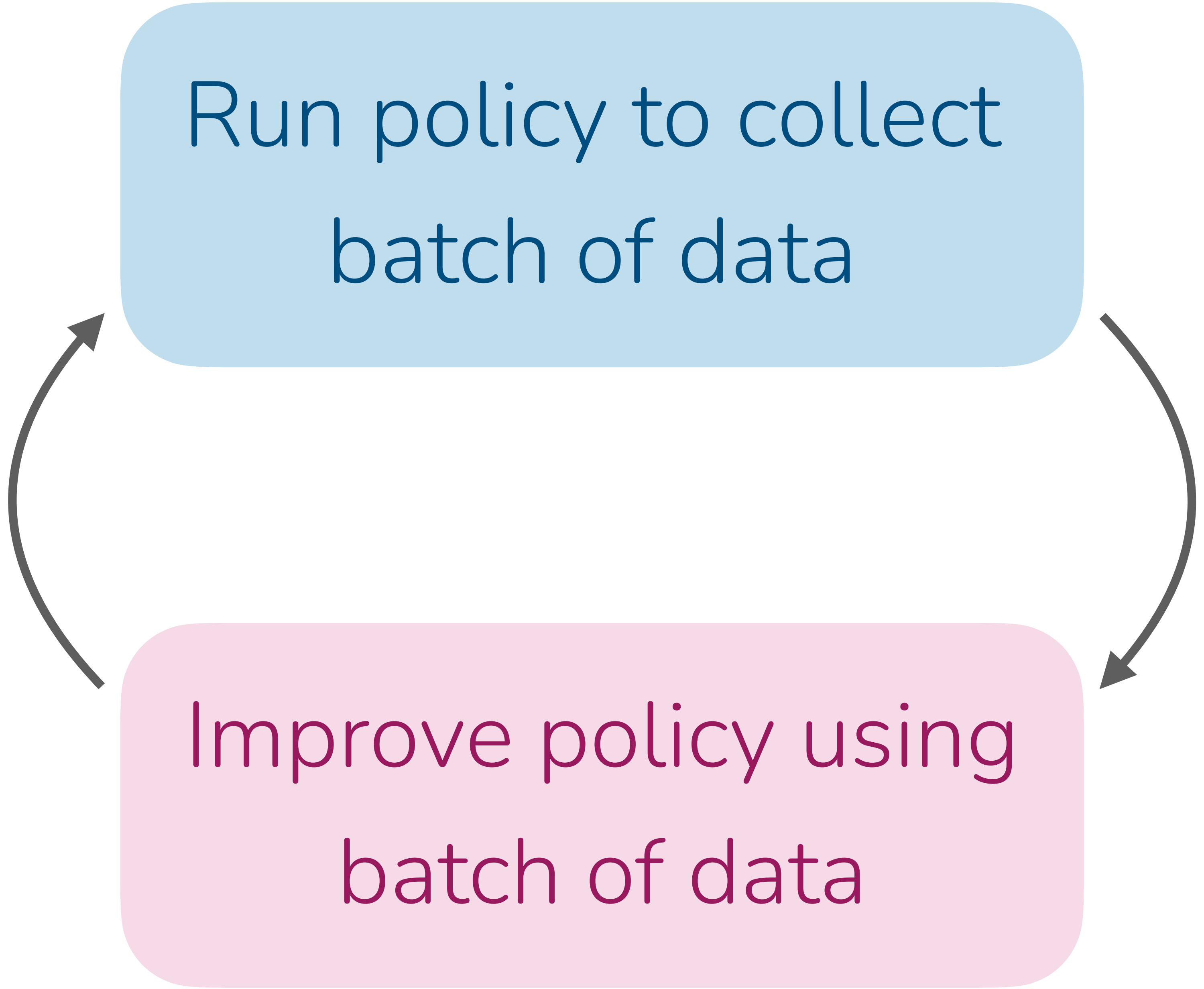
Actor Critic Methods

CS 224R

Course reminders

- Start forming **final project groups** (survey due next Wednesday)
- **Homework 1** due next Friday

Recap of Last Time



Online reinforcement learning with policy gradients

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

Do more of the above average stuff, less of the below average stuff.

Recap of Last Time

Vanilla policy gradients is fully **on-policy**.

—> only one gradient step per batch of data

attribute of online RL algorithms

Definitions.

on-policy: update uses only data from current policy

off-policy: update can reuse data from other, past policies

Importance weights for *off-policy* policy gradient:

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

Note: Not a silver bullet. Only suitable when policies are very similar.

The plan for today

Actor critic methods

1. Improving policy gradients
2. How to estimate the value of a policy
    a. Sample & directly supervise
    b. Use your own estimate
    c. Something in-between?
3. Off-policy actor-critic
    a. Importance weights & constraining step size
    b. Full off-policy version with replay buffers

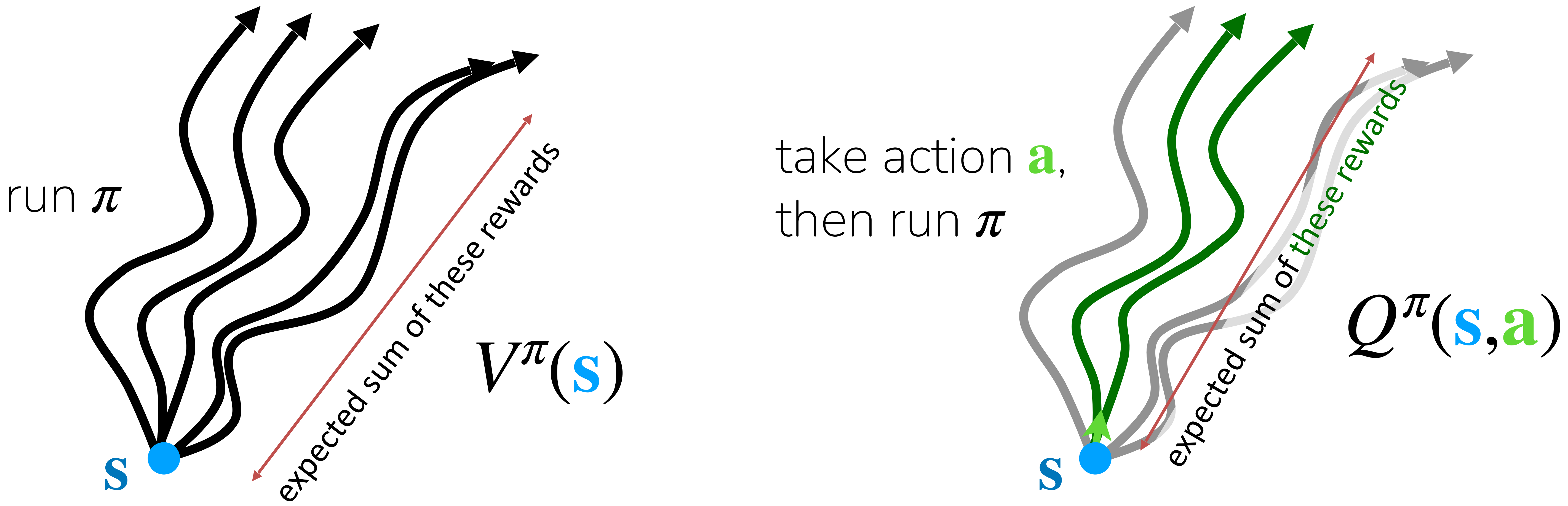
Key learning goals:

- How to estimate how good a state and action is for a policy
- How to use those estimates to form a more efficient RL algorithm

Revisiting Some Useful Objects

value function $V^{\pi}(\mathbf{s})$ - future expected rewards starting at $\mathbf{s}$ and following π

Q-function $Q^{\pi}(\mathbf{s}, \mathbf{a})$ - future expected rewards starting at $\mathbf{s}$, taking $\mathbf{a}$, then following π

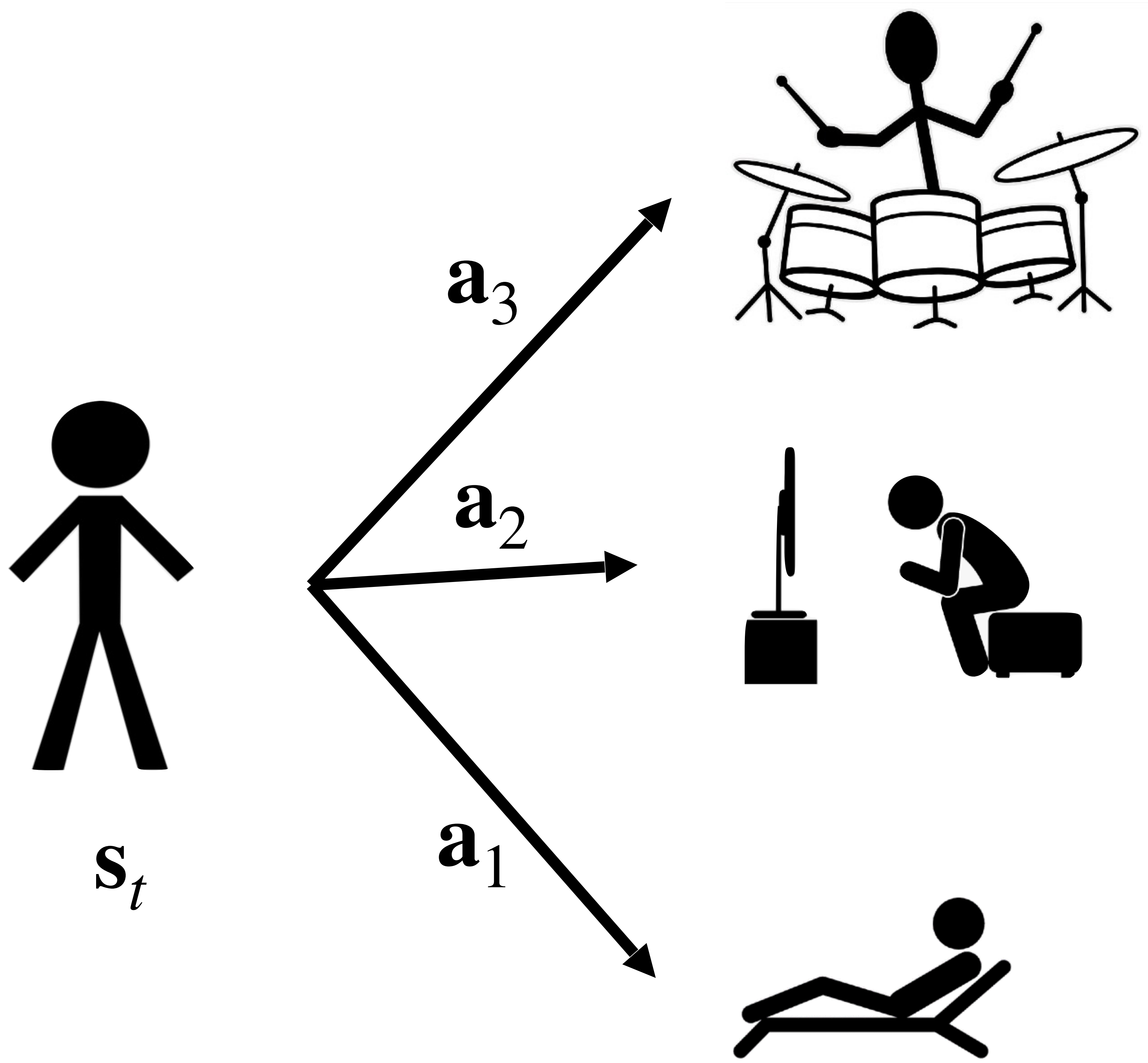


Useful relation: $V^{\pi}(\mathbf{s}) = \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}\left[Q^{\pi}(\mathbf{s}, \mathbf{a})\right]$

advantage $A^{\pi}(\mathbf{s}, \mathbf{a})$ - how much better it is to take $\mathbf{a}$ than to follow policy π at state $\mathbf{s}$

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

Let’s look at an example



Reward = 1 if I can play it in a month, 0 otherwise

Question:

For each action, what are these?

Value function: $V^{\pi}(\mathbf{s}_t) = \ ?$

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \ ?$

Advantage function: $A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \ ?$

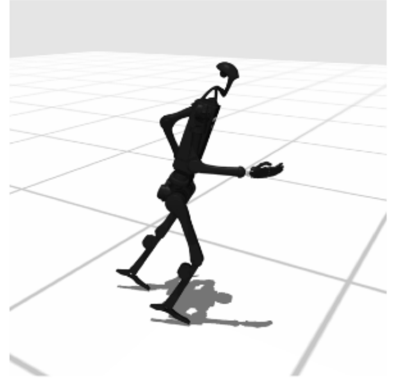
Current policy: $\pi(\mathbf{a}_1 \,|\, \mathbf{s}) = 1 \ \forall \mathbf{s}$

What is dissatisfying about policy gradients?

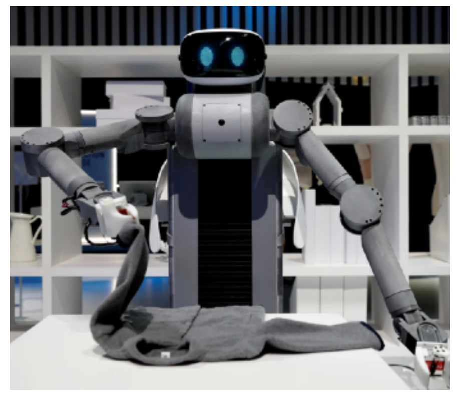
Run policy to collect batch of data

Improve policy using batch of data

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



τ^4: one small step forward then falls backwards



τ^2: folds only the sleeves
τ^3: flattens the jacket but does not fold it
τ^4: folds the jacket

-> pushes down likelihood of step forward

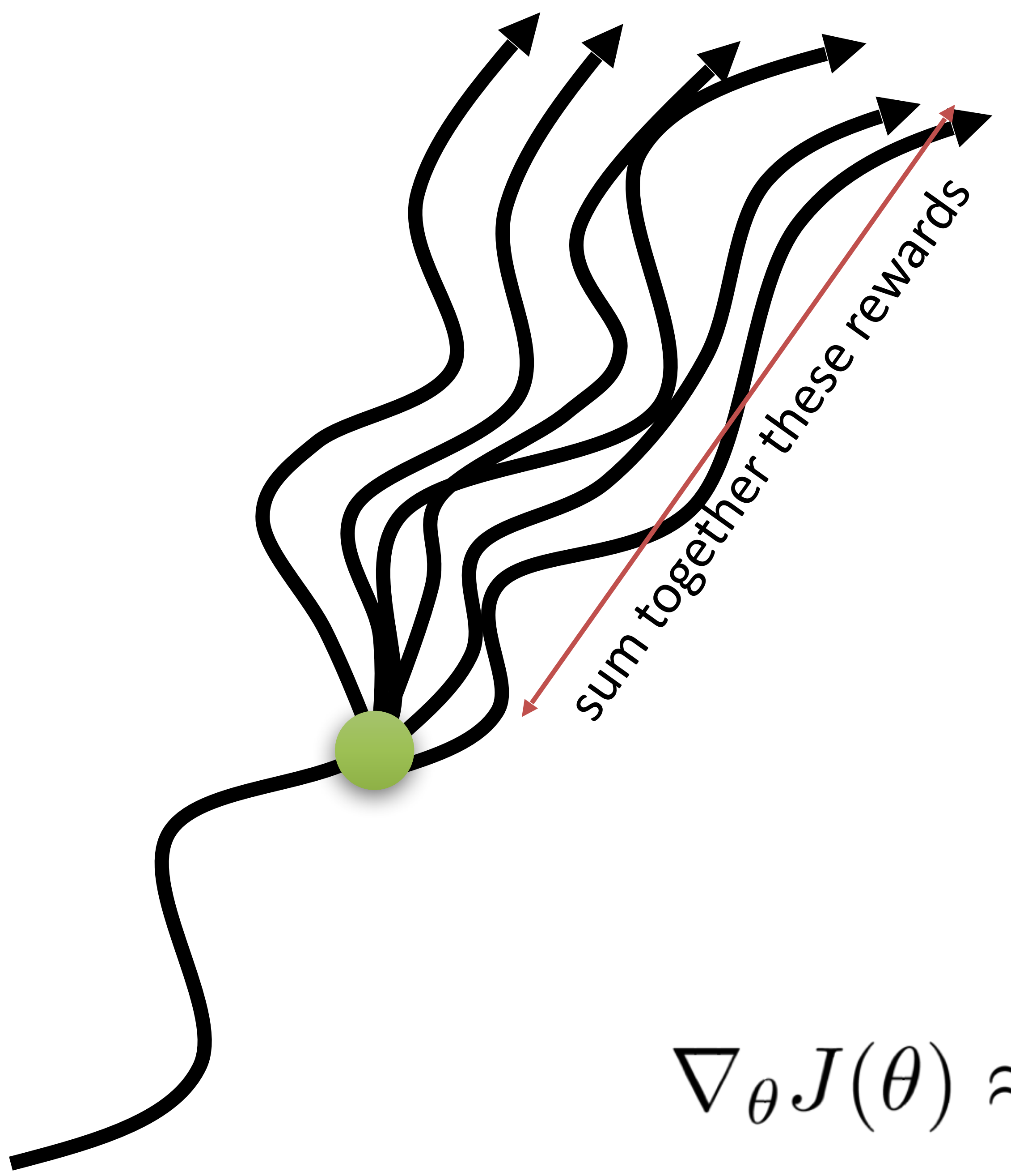
-> with sparse rewards, don't utilize τ^2 or τ^3

Policy gradients doesn't make efficient use of data! 💡 Can we learn what is good & bad?

Improving policy gradients

Estimating reward to go

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



estimate of future rewards if we take $\mathbf{a}_{i,t}$ in state $\mathbf{s}_{i,t}$

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

Can we get a better estimate?

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

true expected rewards to go

This would be way better!

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

Improving policy gradients

What about baselines?

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

$$V^\pi(\mathbf{s}_t)$$

$$b = \text{average reward} = \frac{1}{N} \sum_i Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

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

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

With baseline:

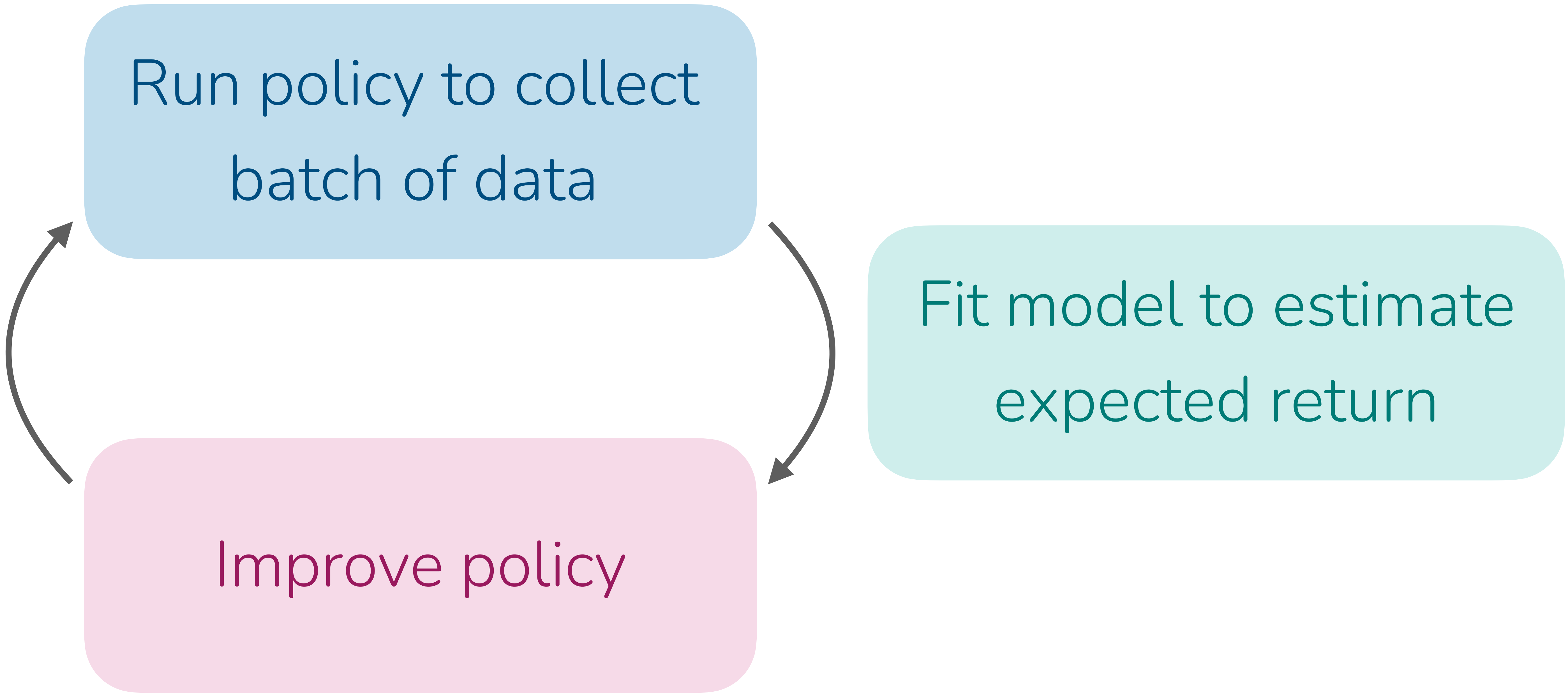
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Better estimates of A lead to less noisy gradients!

Online RL Outline

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

First: Initialize the policy (randomly, with imitation learning, with heuristics)



Online RL Outline

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

First: Initialize the policy (randomly, with imitation learning, with heuristics)

Run policy to collect batch of data

Fit model to estimate expected return

Estimate V^π, Q^π, or A^π

Improve policy

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

The plan for today

Actor critic methods

1. Improving policy gradients
2. **How to estimate the value of a policy**
    a. Sample & directly supervise
    b. Use your own estimate
    c. Something in-between?
3. Off-policy actor-critic
    a. Importance weights & constraining step size
    b. Full off-policy version with replay buffers

Key learning goals:

- How to estimate how good a state and action is for a policy
- How to use those estimates to form a better RL algorithm

Estimating expected return

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

Should we fit V^π, Q^π, or A^π?

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

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

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

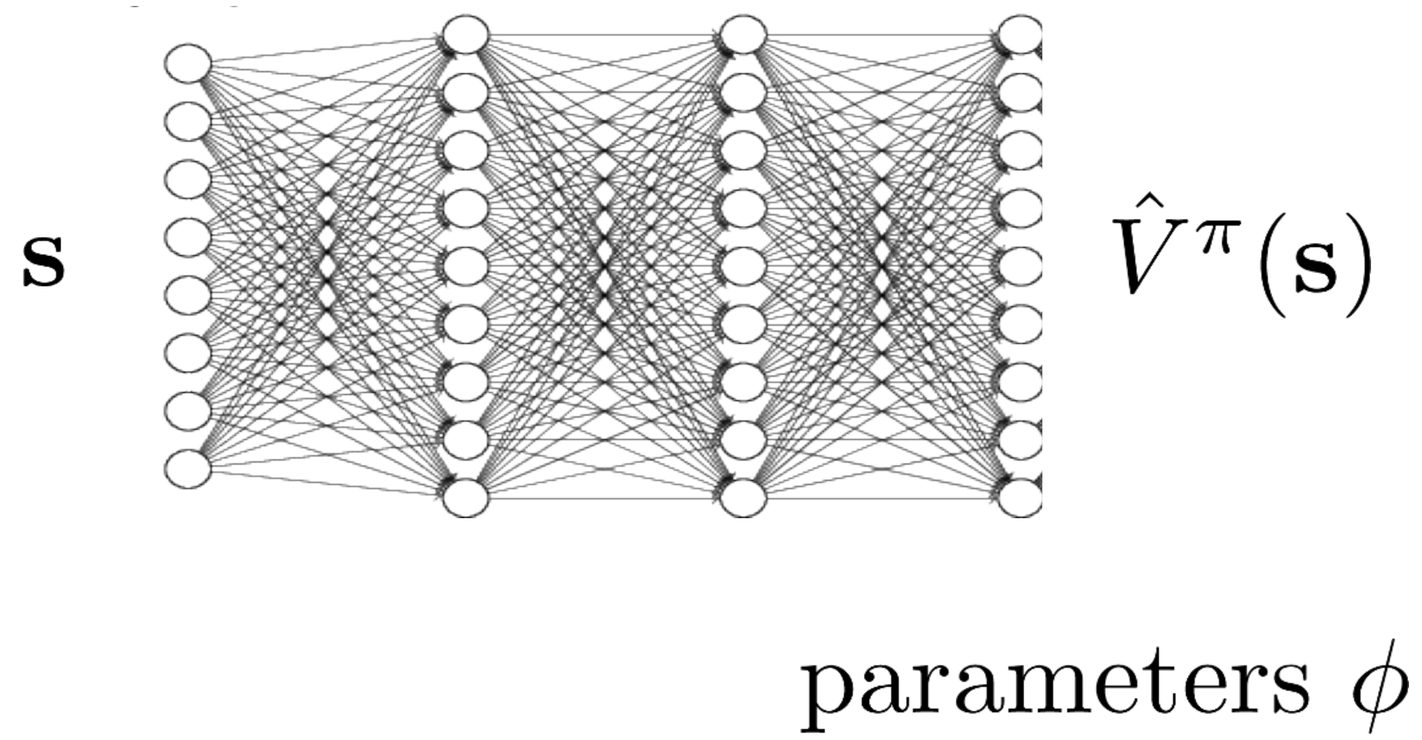
$$= r(\mathbf{s}_t,\mathbf{a}_t) + E_{\mathbf{s}_{t+1}\sim p(\cdot\,|\mathbf{s}_t,\mathbf{a}_t)}[V^\pi(\mathbf{s}_{t+1})]$$

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

(use the sampled $\mathbf{s}_{t+1}$)

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

Let's just fit V^π!

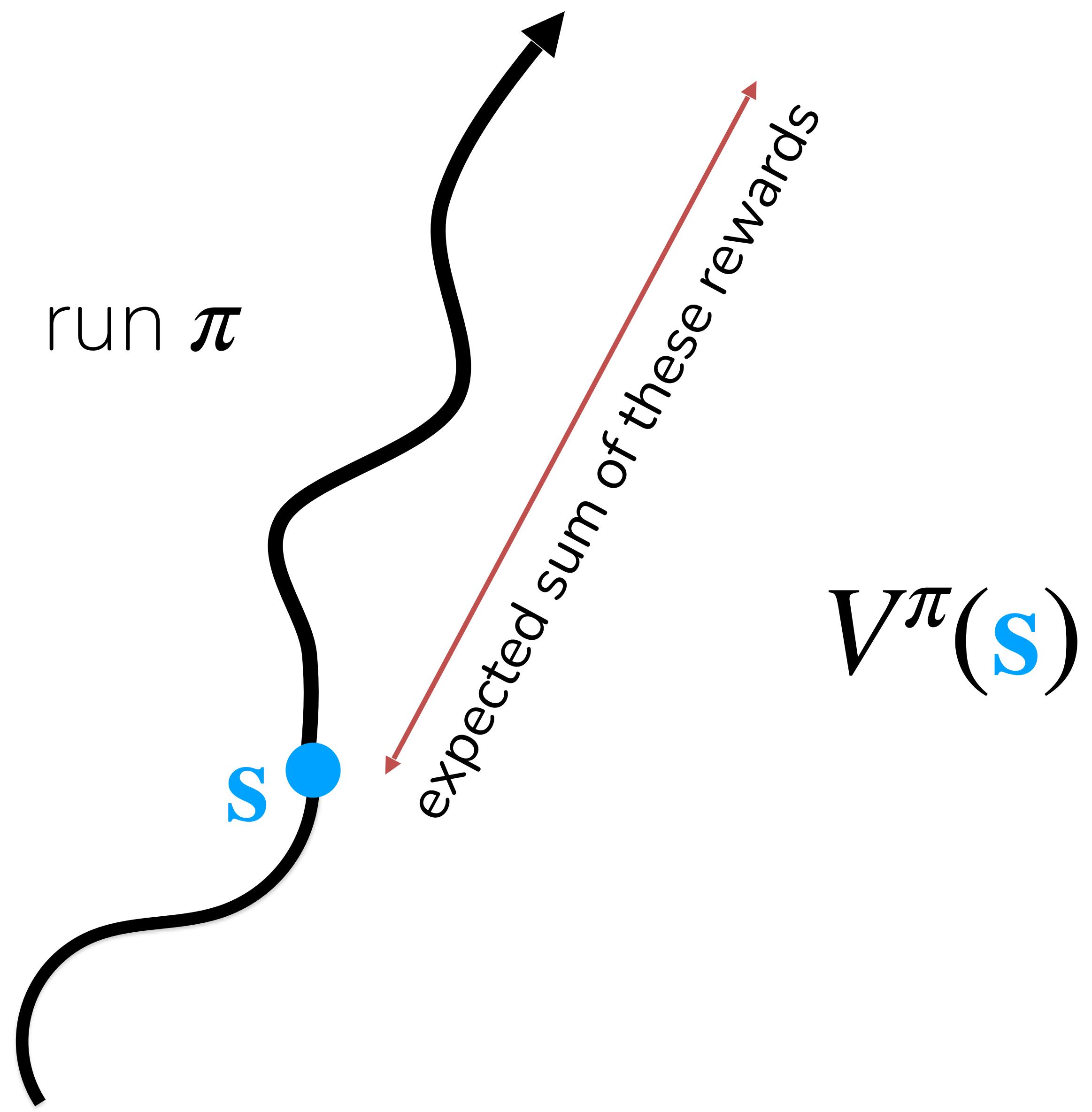


Estimating V^π

Version 1: Monte Carlo estimation

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

<- original single sample estimate



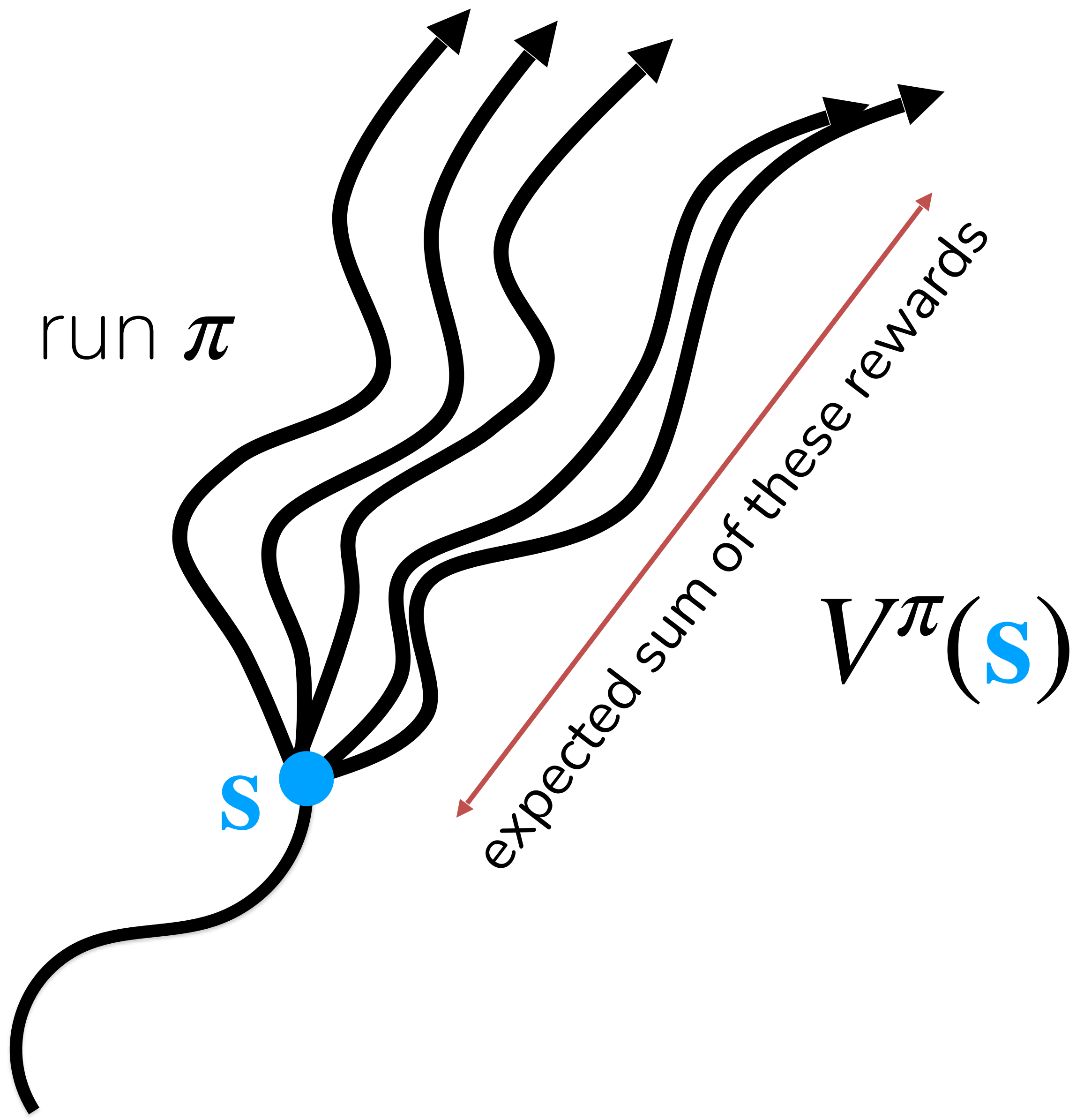
Estimating V^π

Version 1: Monte Carlo estimation

$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^{T} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$ <- original single sample estimate

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

(but can't reset the world)



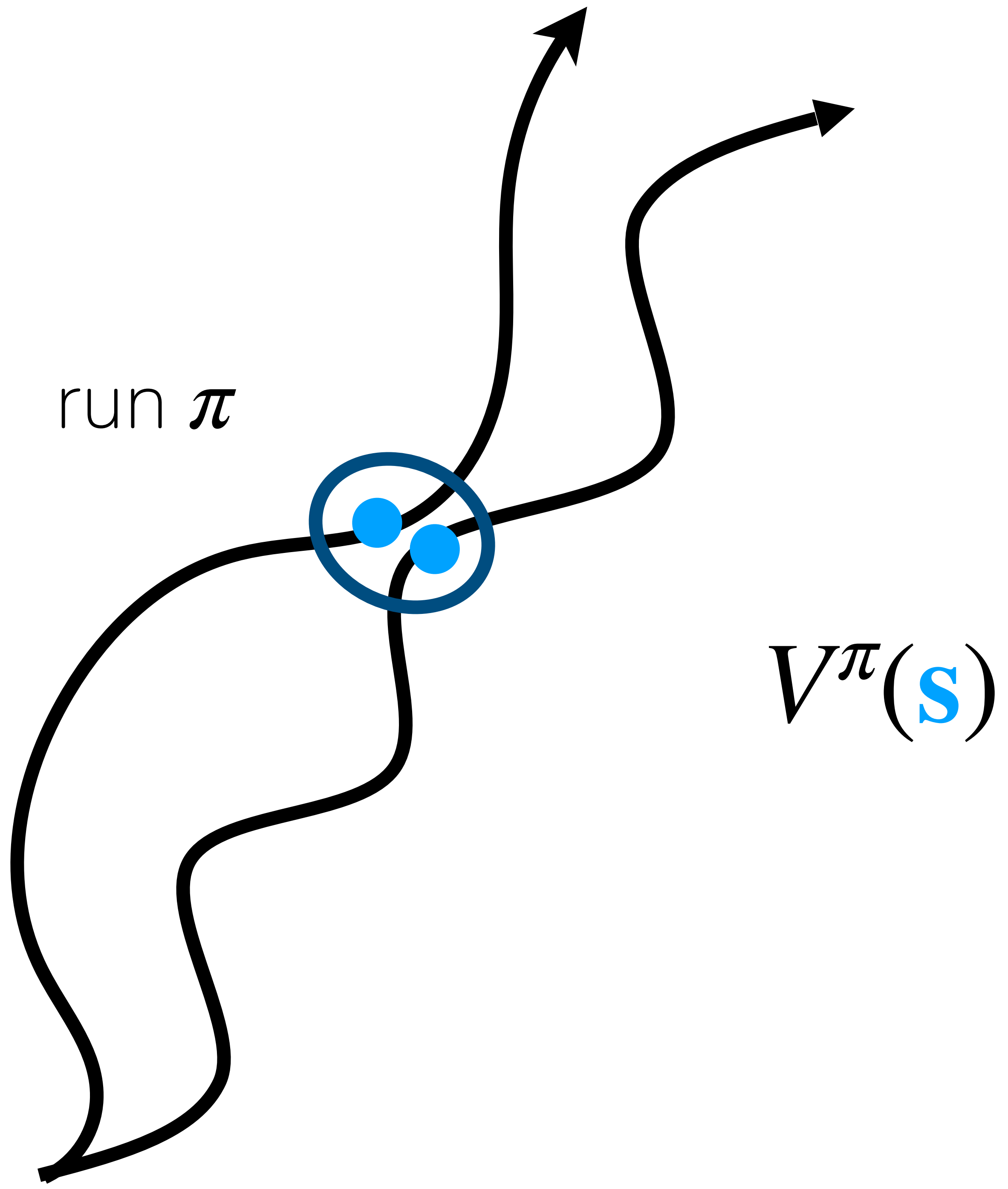
Estimating V^π

Version 1: Monte Carlo estimation

$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^{T} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$ <- original single sample estimate

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

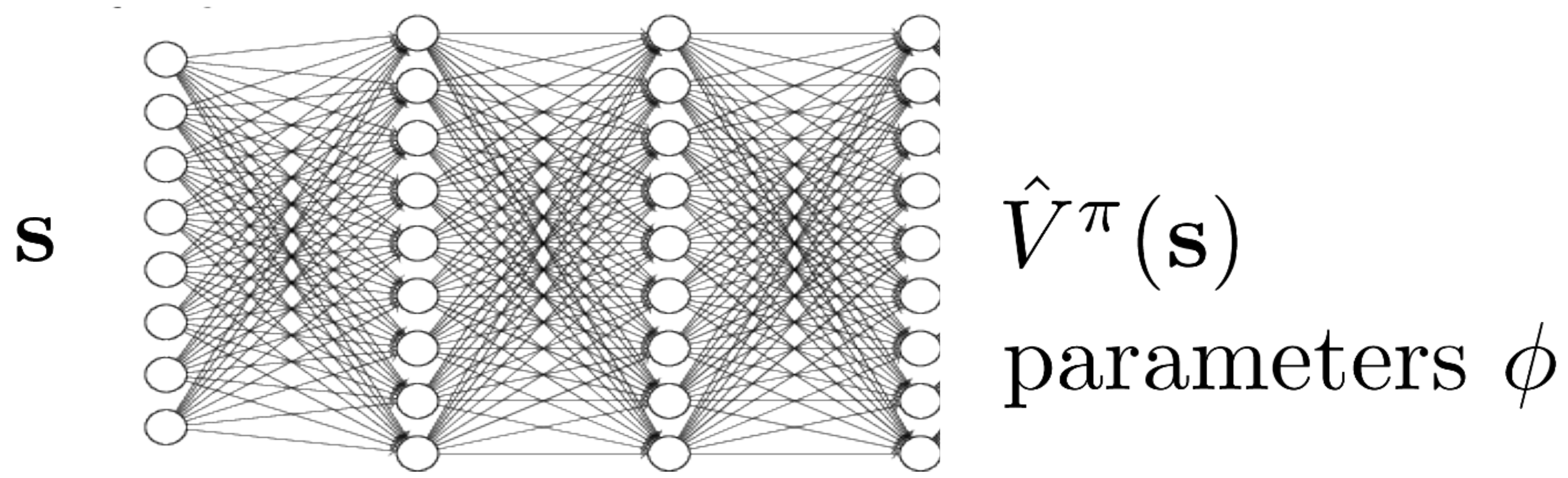
(but can't reset the world)



Step 1: Aggregate dataset of single sample estimates:

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

Step 2: Supervised learning to fit estimated value function

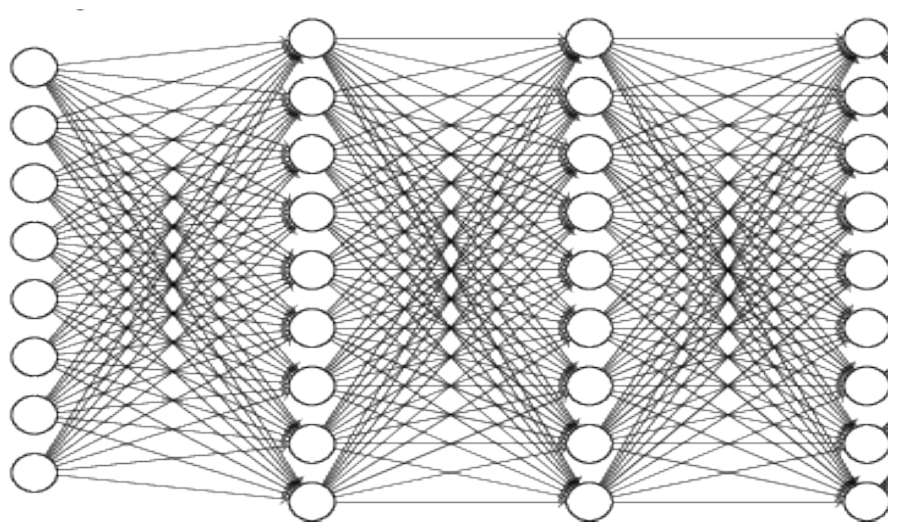


$$\mathcal{L}(\phi) = \frac{1}{2}\sum_i \left\| \hat{V}^\pi_\phi(\mathbf{s}_i) - y_i \right\|^2$$

Estimating V^π

Version 2: Bootstrapping

$\mathbf{s}$



$\hat{V}^\pi(\mathbf{s})$
parameters ϕ

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

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

$$\approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + V^\pi(\mathbf{s}_{i,t+1}) \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \underbrace{\hat{V}^\pi_\phi(\mathbf{s}_{i,t+1})}_{\text{directly use previous fitted value function!}}$$

Step 1: Aggregate dataset of "bootstrapped" estimates:

training data: $\left\{\left(\mathbf{s}_{i,t}, \underbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}^\pi_\phi(\mathbf{s}_{i,t+1})}_{y_{i,t}}\right)\right\}$ <- *update labels every gradient update!*

Step 2: Supervised learning to fit estimated value function

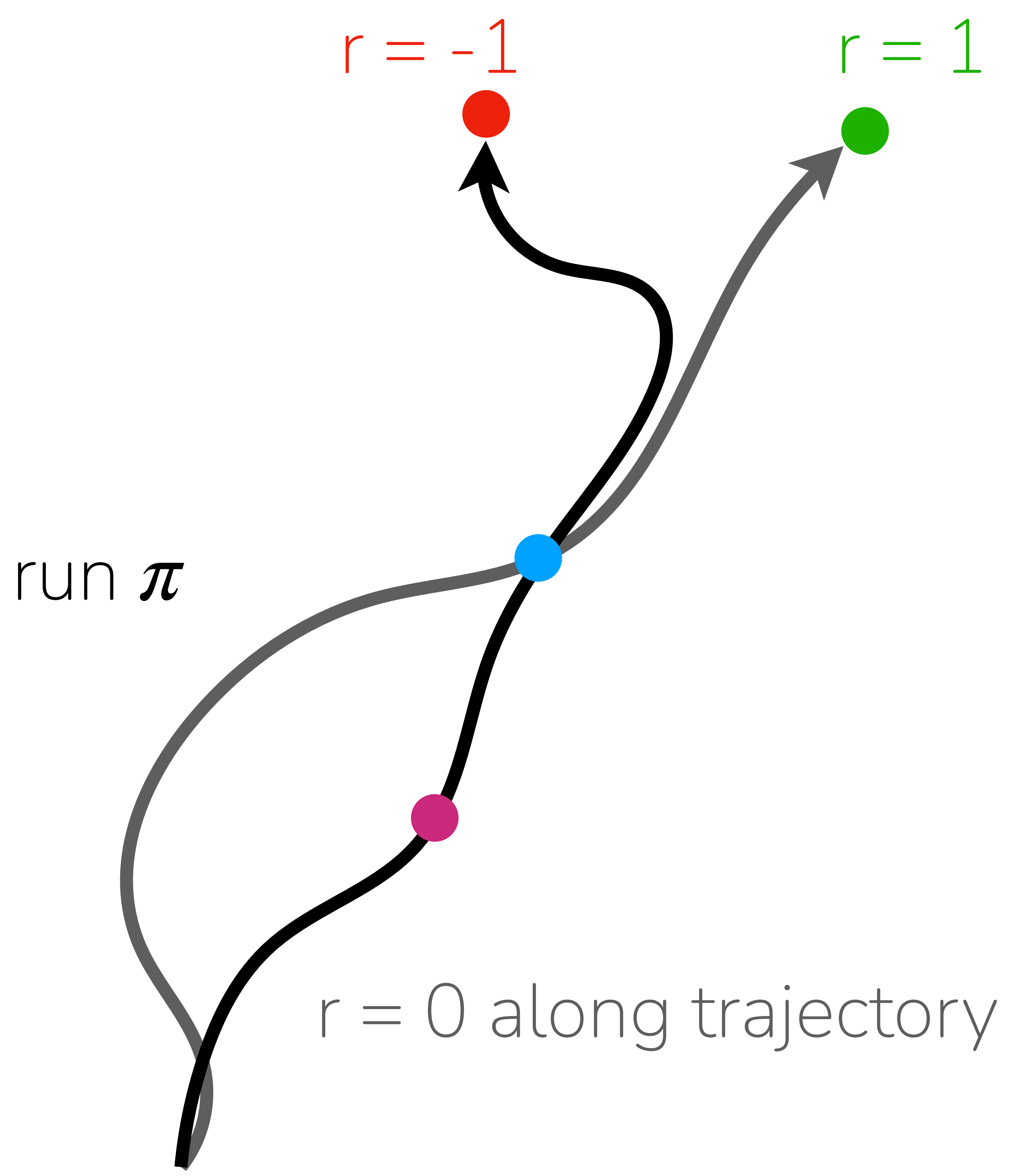
$$\mathcal{L}(\phi) = \frac{1}{2}\sum_i \left\|\hat{V}^\pi_\phi(\mathbf{s}_i) - y_i\right\|^2$$

Also referred to as a form of **temporal difference learning**

Estimating V^π: Monte Carlo vs. Bootstrap

Monte Carlo	$y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$	*supervise with roll-out's summed rewards*
Bootstrapped	$y_{i,t} = r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})$	*supervise using reward and current value estimate*

Let's look at an example:



Think-pair-share:

$\hat{V}^\pi_{MC}(\mathbf{s}) \approx ?$ $\hat{V}^\pi_{MC}(\mathbf{s}) \approx ?$

$\hat{V}^\pi_{TD}(\mathbf{s}) \approx ?$ $\hat{V}^\pi_{TD}(\mathbf{s}) \approx ?$

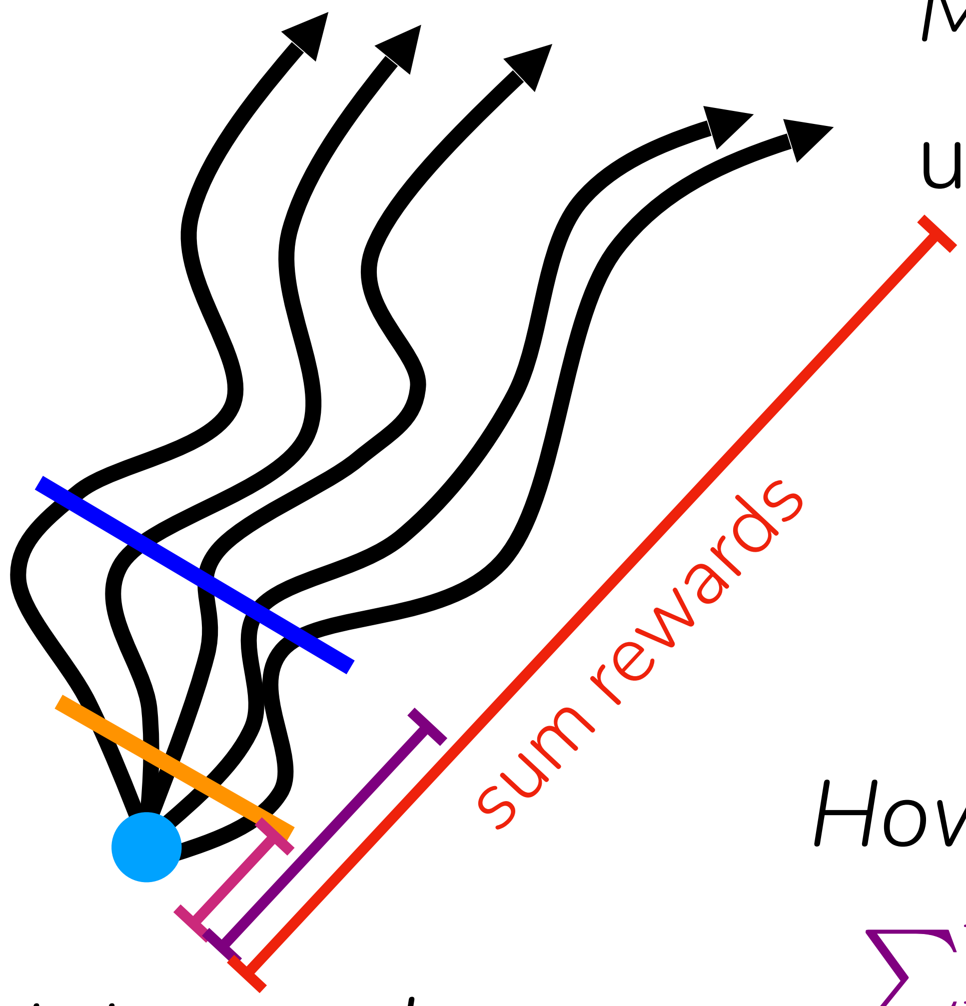
Is there middle ground?
Can we balance bias and variance?

Estimating V^{π}: Monte Carlo vs. Bootstrap

Monte Carlo	$y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$	*supervise with roll-out's summed rewards*
Bootstrapped	$y_{i,t} = r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+1})$	*supervise using reward and current value estimate*
N-step returns	$y_{i,t} = \sum_{t'=t}^{t+n-1} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) + \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+n})$	

Monte Carlo:

using $\sum_{t}^{T} r$



Bootstrapped:

using r+V

How about

$\sum_{t}^{t+n-1} r + V_{t+n}$

-> less variance than MC

-> lower bias than 1-step bootstrap

$n > 1$, $n < T$ *often works the best in practice!*

Aside: discount factors

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

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

what if T (episode length) is ∞?

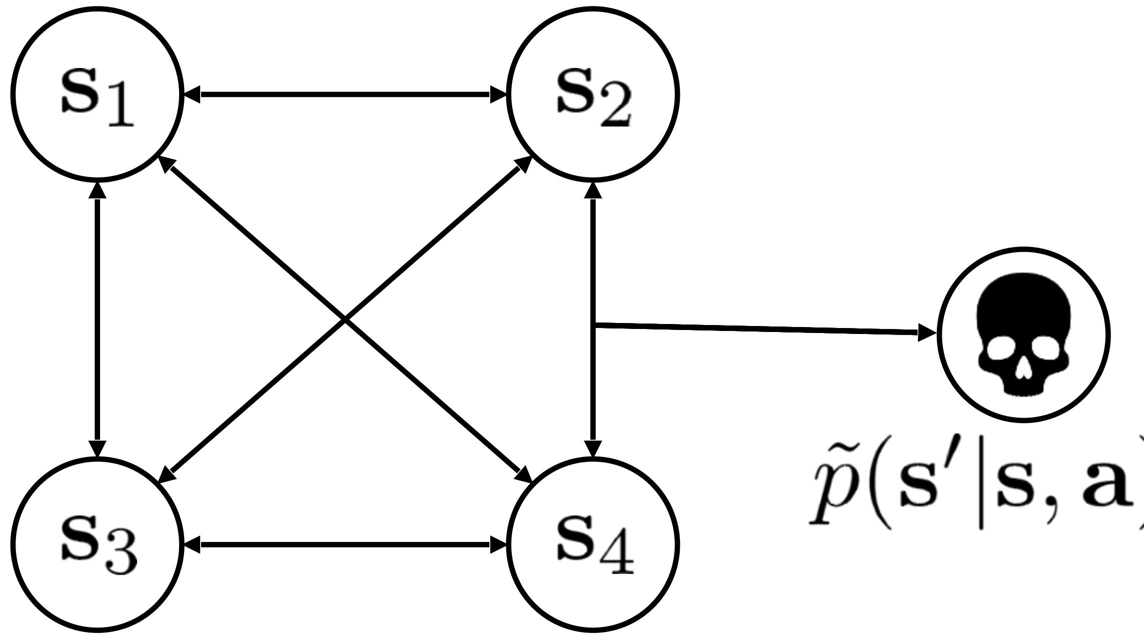
$\hat{V}_\phi^\pi$ can get infinitely large in many cases

simple trick: better to get rewards sooner than later

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

discount factor $\gamma \in [0, 1]$ (0.99 works well)

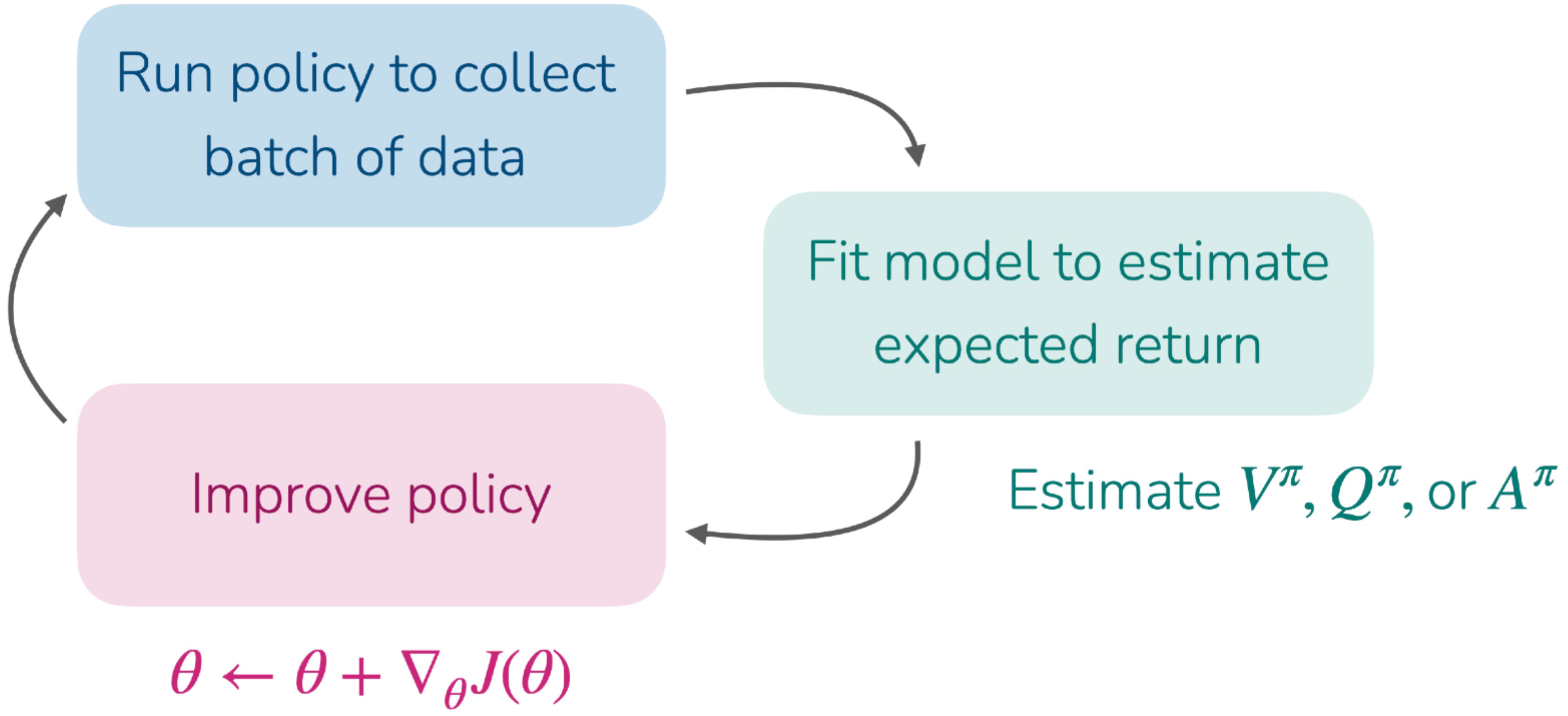
γ changes the MDP:



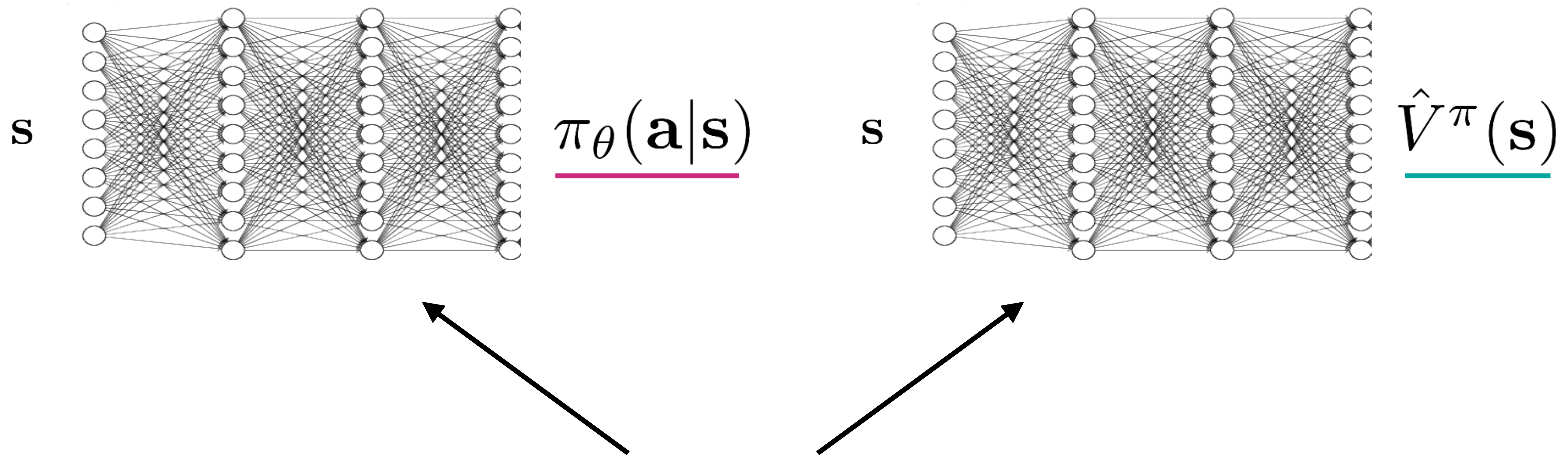
$\tilde{p}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = (1 - \gamma)$

$\tilde{p}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) = \gamma p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

A Full Algorithm Walkthrough



1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}^{\pi_\theta}_\phi$ to summed rewards in data
3. Evaluate $\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) = r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \gamma \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t+1,i}) - \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t,i}) \quad \forall t, i$
4. Evaluate $\nabla_\theta J(\theta) \approx \sum_{t,i} \nabla_\theta \log \pi_\theta(\mathbf{a}_{t,i} \mid \mathbf{s}_{t,i}) \, \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



$$y_{i,t} = \sum_{t'=t}^{t+n-1} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) + \gamma^n \hat{V}^\pi_\phi(\mathbf{s}_{i,t+n})$$

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

“actor-critic algorithm”

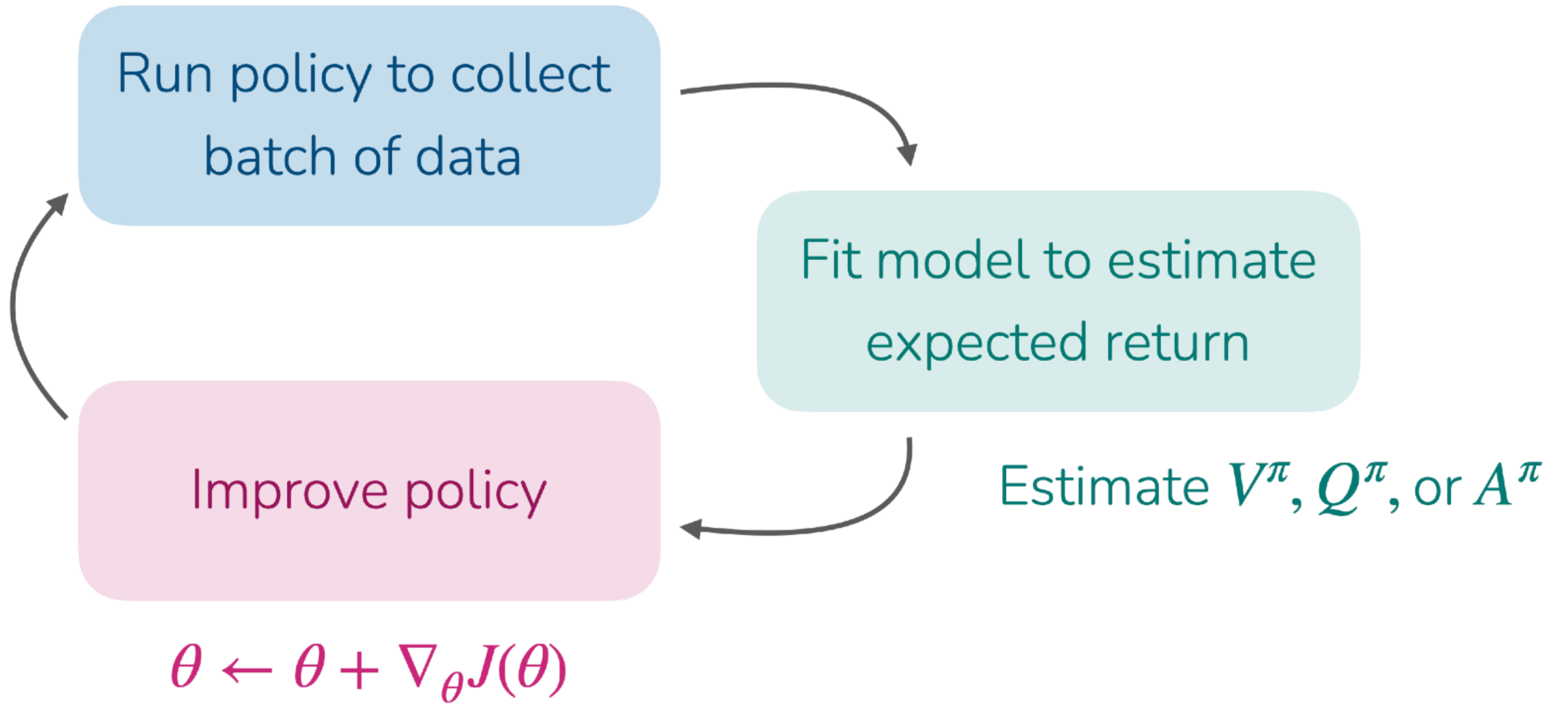
Review so far

Algorithms

Policy gradients: observe what is good vs. bad, then do more of good stuff

Actor-critic: learn to estimate what is good vs. bad, then do more of the good stuff

i.e. get a better policy gradient by using neural network to estimate value function



How to estimate value: "policy evaluation"

Supervised learning directly on observed sum of future rewards

Supervised learning on current reward + value estimate of next state

Hybrid: Supervised learning on sum of next n rewards + value of state after that

What about off-policy versions?

The plan for today

Actor critic methods

1. Improving policy gradients
2. How to estimate the value of a policy ("policy evaluation")
   a. Sample & directly supervise ("Monte Carlo estimation")
   b. Use your own estimate ("bootstrapping", "temporal difference learning")
   c. Something in-between? ("N-step returns")
3. **Off-policy actor-critic**
   a. Importance weights & constraining step size
   b. Full off-policy version with replay buffers

Key learning goals:

- How to estimate how good a state and action is for a policy
- How to use those estimates to form a better RL algorithm

Off-Policy Actor-Critic Methods

Version 1: Multiple Gradient Steps

1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}_\phi^{\pi_\theta}$ to summed rewards in data
3. Evaluate $\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) = r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \gamma \hat{V}_\phi^{\pi_\theta}(\mathbf{s}_{t+1,i}) - \hat{V}_\phi^{\pi_\theta}(\mathbf{s}_{t,i}) \quad \forall t, i$
4. Evaluate $\nabla_\theta J(\theta) \approx \sum_{t,i} \nabla_\theta \log \pi_\theta(\mathbf{a}_{t,i} \mid \mathbf{s}_{t,i}) \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$ <- use importance weights here
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Off-Policy Actor-Critic Methods

Version 1: Multiple Gradient Steps

1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}^{\pi_\theta}_\phi$ to summed rewards in data
3. Evaluate $\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) = r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \gamma \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t+1,i}) - \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t,i}) \quad \forall t, i$
4. Evaluate $\nabla_{\theta'} J(\theta') \approx \sum_{t,i} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{t,i}|\mathbf{s}_{t,i}) \ \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$ <- use importance weights here
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Policy will increase probability on actions with high advantages

Advantages based on old policy.

What can go wrong if you take too many gradient steps?

Off-Policy Actor-Critic Methods

Version 1: Multiple Gradient Steps

1. Sample batch of data $\{(\mathbf{s}_{1,i}, \mathbf{a}_{1,i}, \ldots, \mathbf{s}_{T,i}, \mathbf{a}_{T,i})\}$ from π_θ
2. Fit $\hat{V}^{\pi_\theta}_\phi$ to summed rewards in data
3. Evaluate $\hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) = r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i}) + \gamma \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t+1,i}) - \hat{V}^{\pi_\theta}_\phi(\mathbf{s}_{t,i}) \quad \forall t, i$
4. Evaluate $\nabla_{\theta'} J(\theta') \approx \sum_{t,i} \frac{\pi_{\theta'}(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}{\pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{t,i}|\mathbf{s}_{t,i}) \; \hat{A}^{\pi_\theta}(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$ <- use importance weights here
5. Update $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Policy will increase probability on actions with high advantages

Advantages are based on old policy, will get out-dated

What can go wrong if you take too many gradient steps?

💡Idea 1: Use KL constraint on policy.

We will see this in LLM preference optimization

$$\mathbb{E}_{\mathbf{s} \sim \pi_\theta} \left[D_{KL}(\pi_{\theta'}(\cdot \mid \mathbf{s}) \| \pi_\theta(\cdot \mid \mathbf{s})) \right] \leq \delta$$

💡Idea 2: Can we bound the importance weights?

Doesn't directly constrain policy, but removes incentives

-> Key idea behind proximal policy optimization (PPO)

Off-Policy Actor-Critic Methods

So far:
- use one batch of policy data for one gradient step (fully on-policy)
- use one batch of policy data for multiple gradient steps (starting to be off-policy)

Can we be even more off-policy?

Can we reuse data from previous batches, i.e. all of the past trial-and-error data?

Key ideas:
- maintain a buffer of all past data   *"replay buffer"*
- adjust equations to remove on-policy assumptions

Off-Policy Actor-Critic Methods

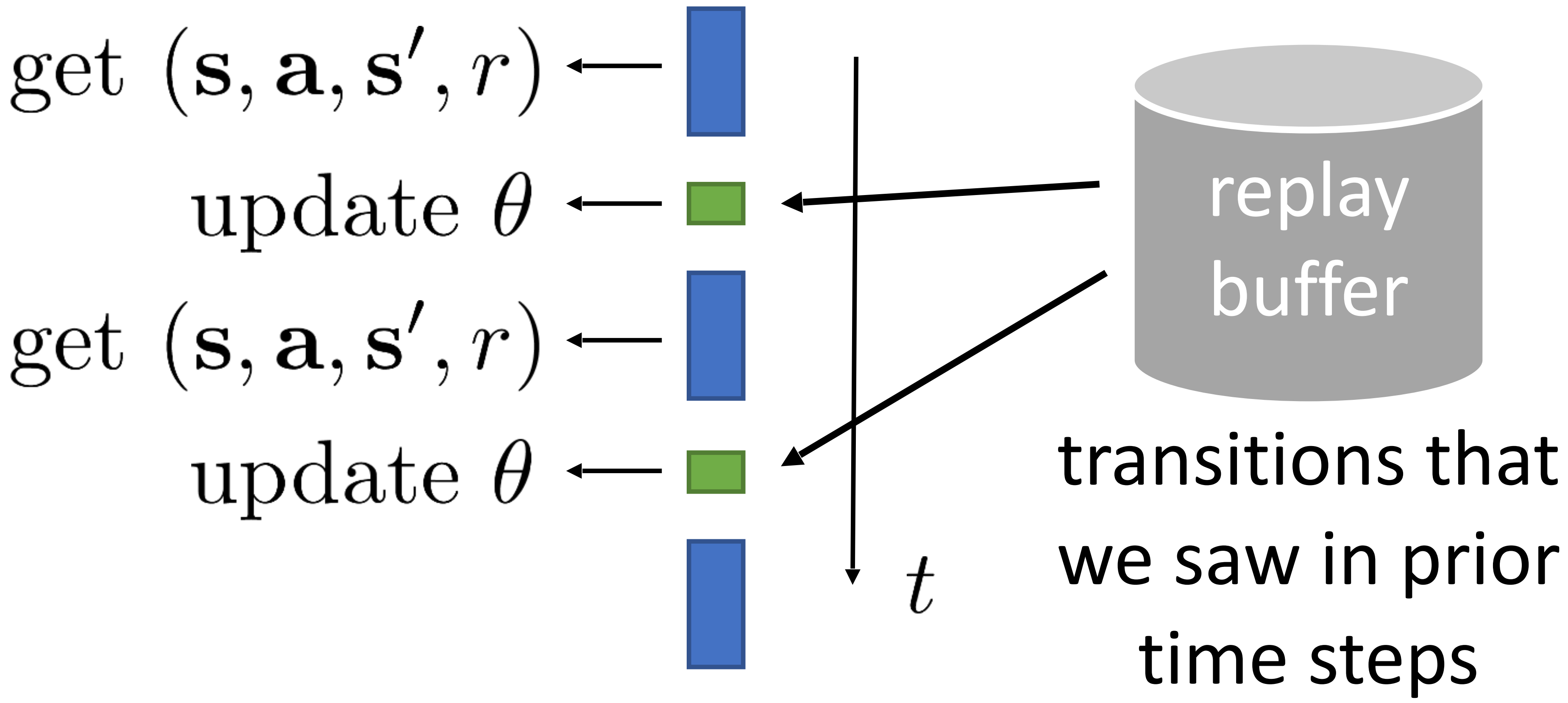
Version 2: Replay buffers

actor-critic algorithm:

1. collect experience $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ } Add this to replay buffer
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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

} Do this on minibatch sampled from **all previous data**

off-policy actor-critic



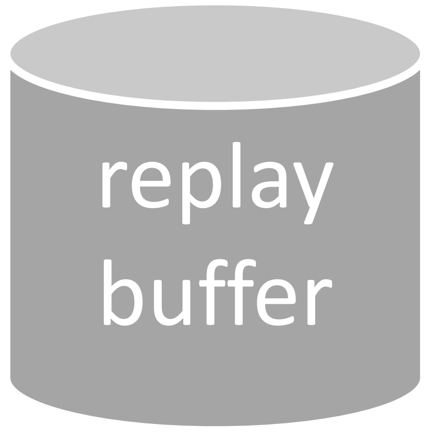
Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Replay buffers

online actor-critic algorithm:

1. collect experience $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (& add to replay buffer)
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{V}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i)$ for each $\mathbf{s}_i$
4. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
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. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



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

mini batch size

not the right target value

not the action π_θ would have taken!

The algorithm is currently broken 😢

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Fixing the value function

online actor-critic algorithm:

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

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

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

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

not the right target value

where does this come from?

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

$$= r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$$

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

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

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

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

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Fixing the policy update

online actor-critic algorithm:

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

not the action π_θ would have taken!

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

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

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

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

higher variance, but convenient

why is higher variance OK here?

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

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Anything else?

online actor-critic algorithm:

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

Any remaining problems?

$\mathbf{s}_i$ 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

Slide adapted from Sergey Levine

Off-Policy Actor-Critic Methods

Version 2: Some implementation details

online actor-critic algorithm:

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

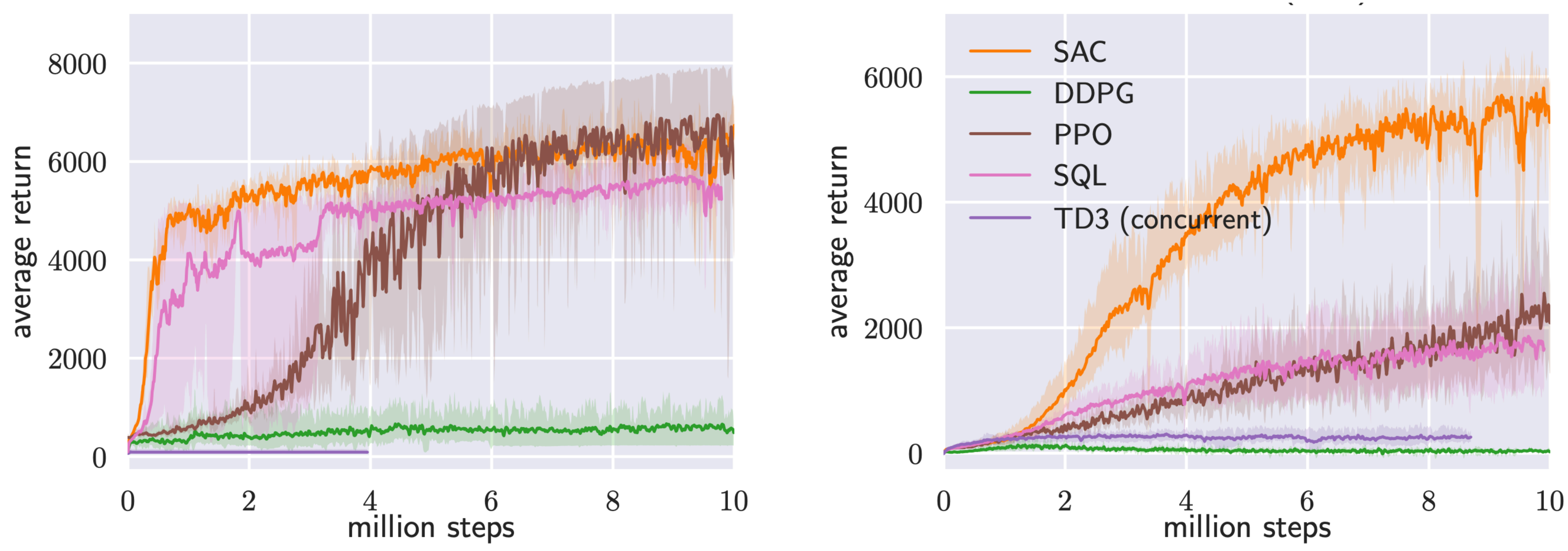
also fancier ways to fit Q-functions (more on this in next two lectures)

can also use **reparameterization trick** to better estimate the gradient (for Gaussian policy)

Example practical algorithm:

Haarnoja, Zhou, Abbeel, Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 2018.

More off-policy vs. less off-policy actor critic?



(e) Humanoid-v1

(f) Humanoid (rllab)

<- more off-policy
(i.e. with replay buffer)

<- less off-policy
(i.e. no replay buffer)

+ Off-policy with replay buffer (e.g. soft actor-critic) can be far more data efficient

- They can also generally be harder to tune hyperparameters, less stable (than e.g. PPO)

The plan for today

Actor critic methods

1. Improving policy gradients
2. How to estimate the value of a policy (“**policy evaluation**”)
   a. Sample & directly supervise (“**Monte Carlo estimation**”)
   b. Use your own estimate (“**bootstrapping**”, “**temporal difference learning**”)
   c. Something in-between? (“**N-step returns**”)
3. Off-policy actor-critic
   a. Importance weights & constraining step size
   b. Full off-policy version with replay buffers

Key learning goals:

- How to estimate how good a state and action is for a policy
- How to use those estimates to form a better RL algorithm

Next Week

Q-learning (last big online RL method)

+

Practical implementation of online RL algorithms

Course reminders

- Form **final project groups & ideas** (survey due next Wednesday)
- **Homework 1** due next Friday