

# Actor-Critic Algorithms

CS 294-112: Deep Reinforcement Learning

Sergey Levine

# Class Notes

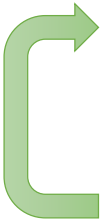
1. Homework 1 due today (11:59 pm)!
  - Don't be late!
2. Homework 2 is out today
  - Start early!
3. Remember to start forming final project groups

# Today's Lecture

1. Improving the policy gradient with a critic
  2. The policy evaluation problem
  3. Discount factors
  4. The actor-critic algorithm
- Goals:
    - Understand how policy evaluation fits into policy gradients
    - Understand how actor-critic algorithms work

# Recap: policy gradients

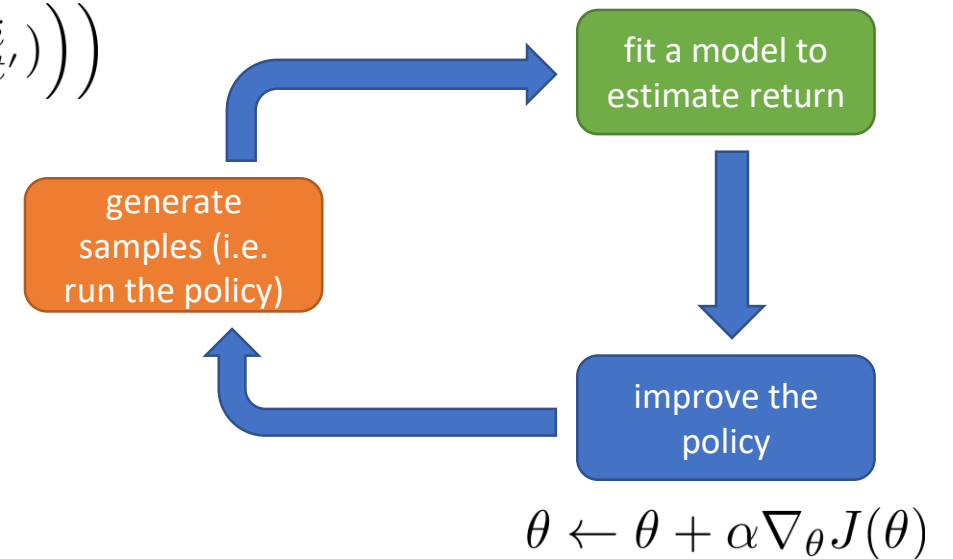
REINFORCE algorithm:

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

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

“reward to go”

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



# Improving the policy gradient

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

“reward to go”

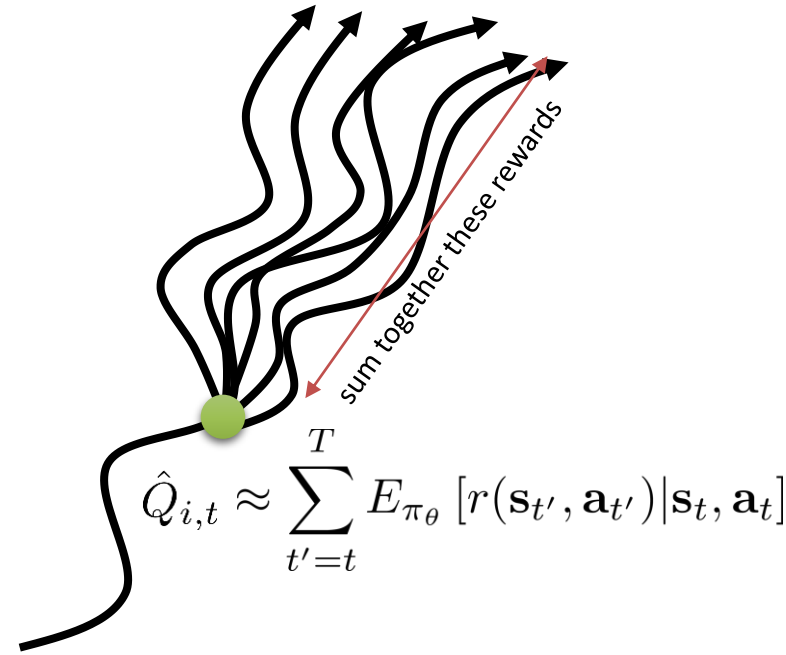
$$\hat{Q}_{i,t}$$

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

can we get 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

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



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

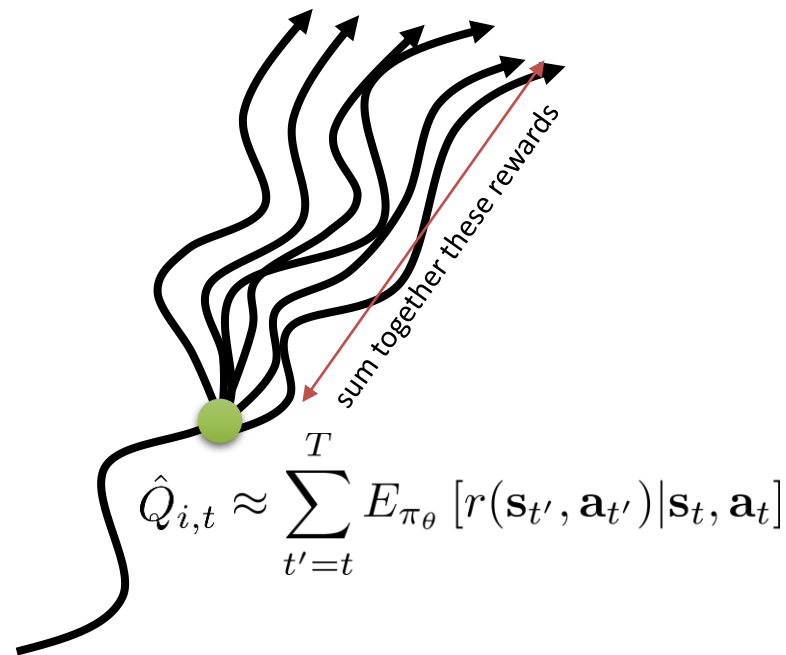
# What about the baseline?

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

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

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

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



# State & state-action value functions

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

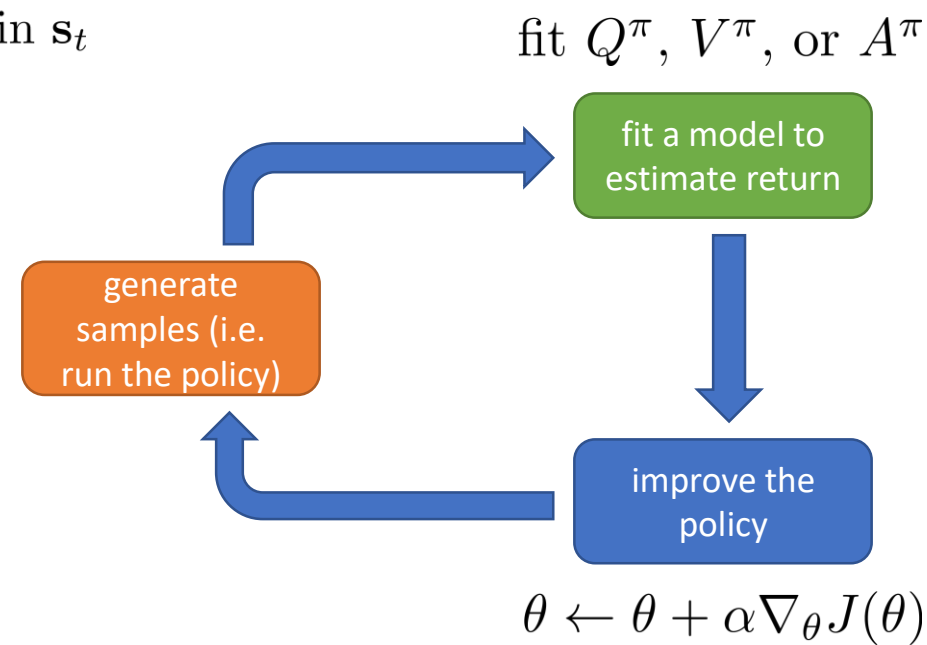
$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$ : how much better  $\mathbf{a}_t$  is

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

the better this estimate, the lower the variance

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

unbiased, but high variance single-sample estimate



# Value function fitting

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

what is the sum total of all the future rewards expected if we take action  $a$  at time  $t$  and continue to take actions in the future until time  $T$  in accordance with policy  $\pi$ ?

$$V^\pi(s_t) = E_{a_t \sim \pi_\theta(a_t | s_t)} [Q^\pi(s_t, a_t)]$$

what is the sum of rewards we can expect under policy  $\pi$  from state  $s_t$  until time  $T$ , if we were to sum individual rewards from each action sampled from a fixed policy  $\pi$ ?  
What is the sum total of future rewards that can be expected if start in state  $t$  and continue until the end?

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

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

fit *what* to *what*?

what is the advantage of taking action  $a$  at time  $t$  in state  $s$  over the baseline, the baseline being the average action in state  $s$  at time  $t$  until time  $T$ . Note that the  $V^\pi(s_t)$  must be smaller (worse) than  $Q^\pi(s_t, a_t)$  if  $a_t$  is the correct action that leads to maximum reward. Therefore, for the correct action in state  $s_t$  the advantage should be positive, for incorrect action - negative.

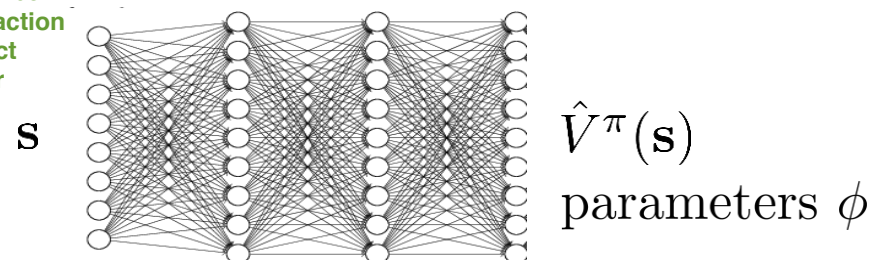
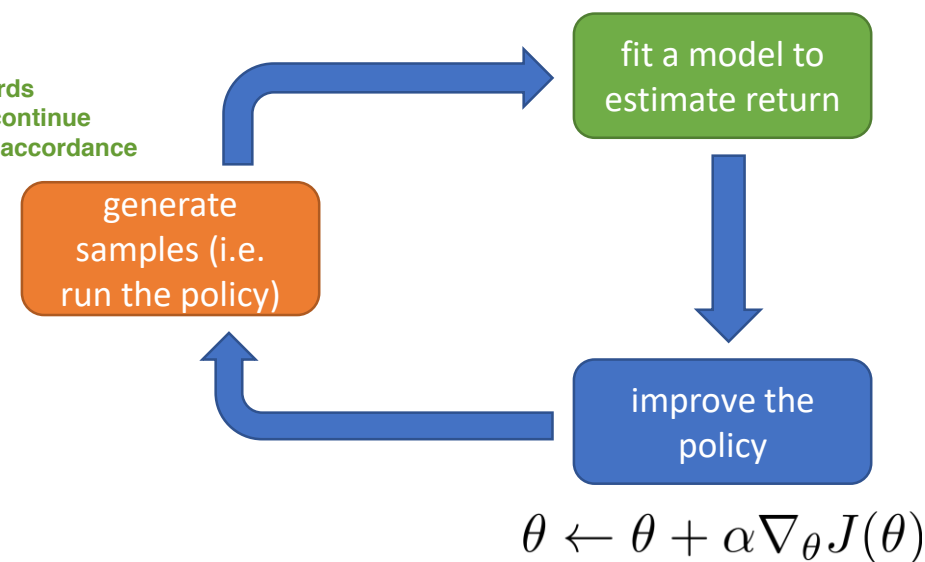
$Q^\pi, V^\pi, A^\pi$ ?

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \underbrace{E_{s_{t+1} \sim p(s_{t+1} | s_t, a_t)} [V^\pi(s_{t+1})]}$$

$$A^\pi(s_t, a_t) \approx r(s_t, a_t) + V^\pi(s_{t+1}) - V^\pi(s_t)$$

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

fit  $Q^\pi, V^\pi$ , or  $A^\pi$



What is  $Q^\pi(s_t)$  in essence? It is the sum of reward at the current time step (reward being the function of state and action both at the current step,  $r_{t+1} = r(s_t, a_t)$ ) and the EXPECTED value at all the remaining time steps.



# Policy evaluation

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

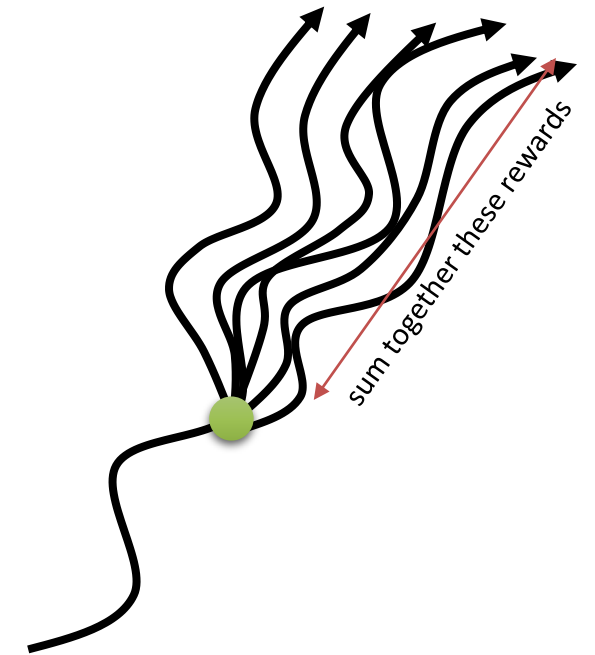
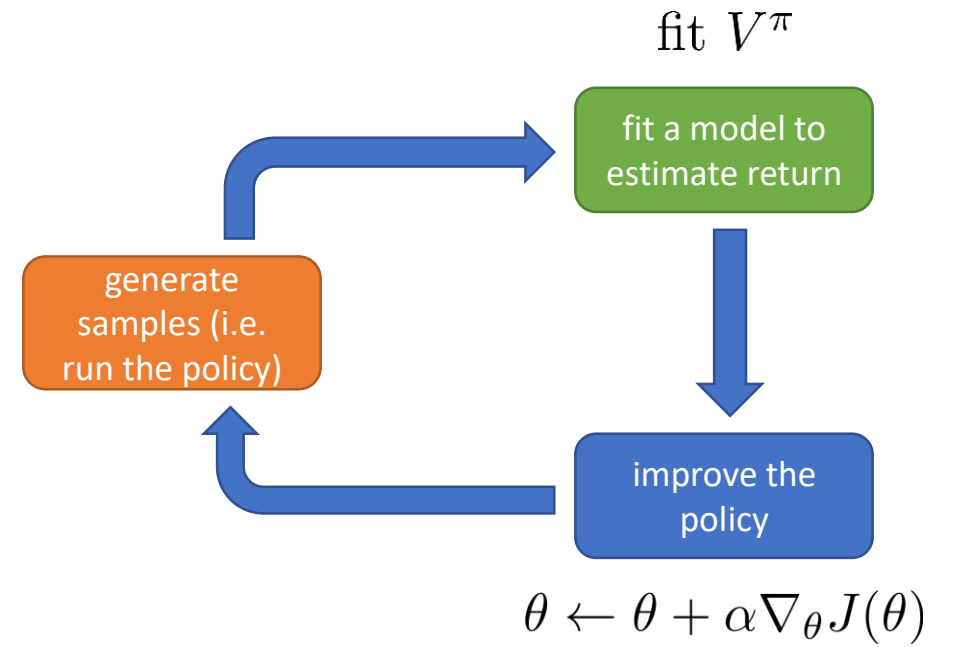
$$J(\theta) = E_{\mathbf{s}_1 \sim p(\mathbf{s}_1)} [V^\pi(\mathbf{s}_1)]$$

how can we perform policy evaluation?

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

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

$$V^\pi(\mathbf{s}_t) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) \quad (\text{requires us to reset the simulator})$$



# Monte Carlo evaluation with function approximation

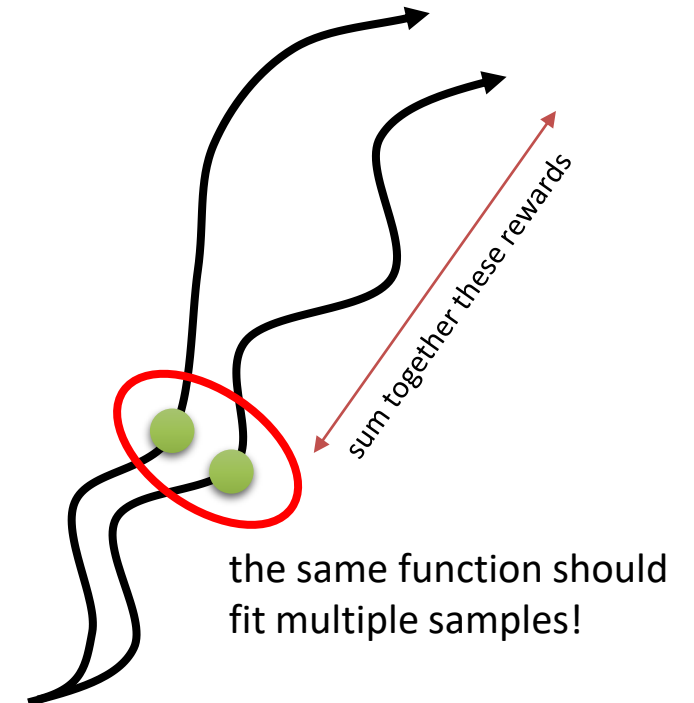
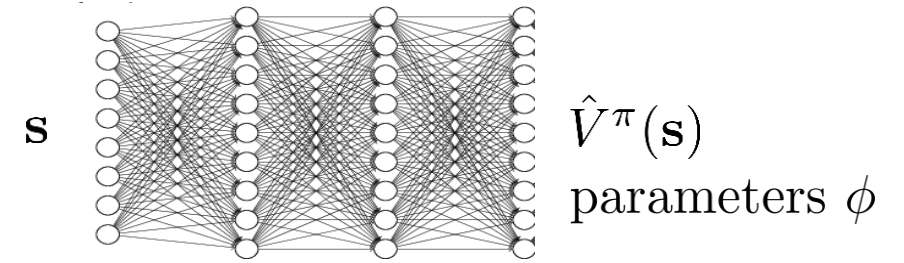
$$V^\pi(\mathbf{s}_t) \approx \sum_{t'=t}^T r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

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



# Can we do better?

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

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

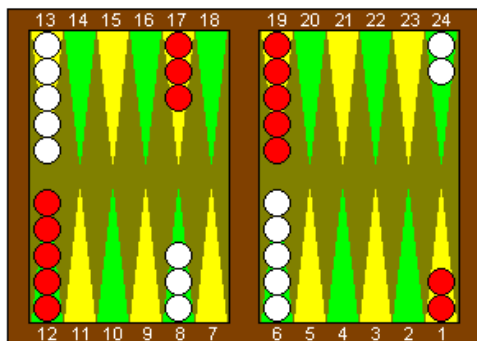
training data:  $\left\{ \left( \mathbf{s}_{i,t}, \underbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}_\phi^\pi(\mathbf{s}_{i,t+1})}_{y_{i,t}} \right) \right\}$

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

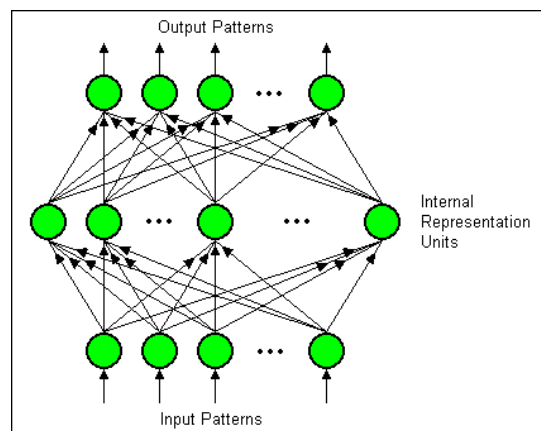
sometimes referred to as a “bootstrapped” estimate

# Policy evaluation examples

TD-Gammon, Gerald Tesauro 1992

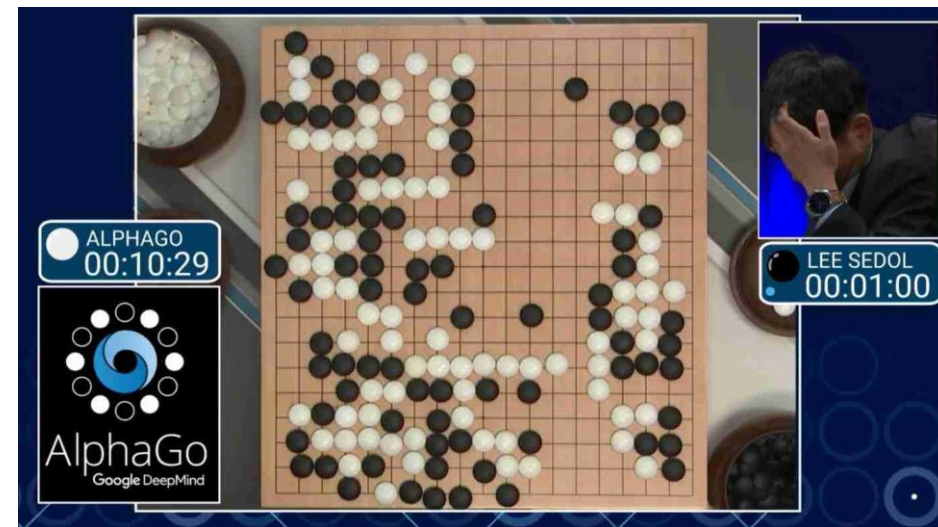


**Figure 2.** An illustration of the normal opening position in backgammon. TD-Gammon has sparked a near-universal conversion in the way experts play certain opening rolls. For example, with an opening roll of 4-1, most players have now switched from the traditional move of 13-9, 6-5, to TD-Gammon's preference, 13-9, 24-23. TD-Gammon's analysis is given in Table 2.



**Figure 1.** An illustration of the multilayer perceptron architecture used in TD-Gammon's neural network. This architecture is also used in the popular backpropagation learning procedure. Figure reproduced from [9].

AlphaGo, Silver et al. 2016



reward: game outcome

value function  $\hat{V}_{\phi}^{\pi}(\mathbf{s}_t)$ :

expected outcome given board state

reward: game outcome

value function  $\hat{V}_{\phi}^{\pi}(\mathbf{s}_t)$ :

expected outcome given board state

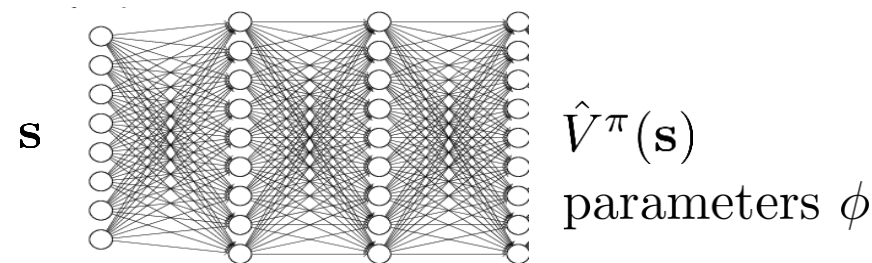
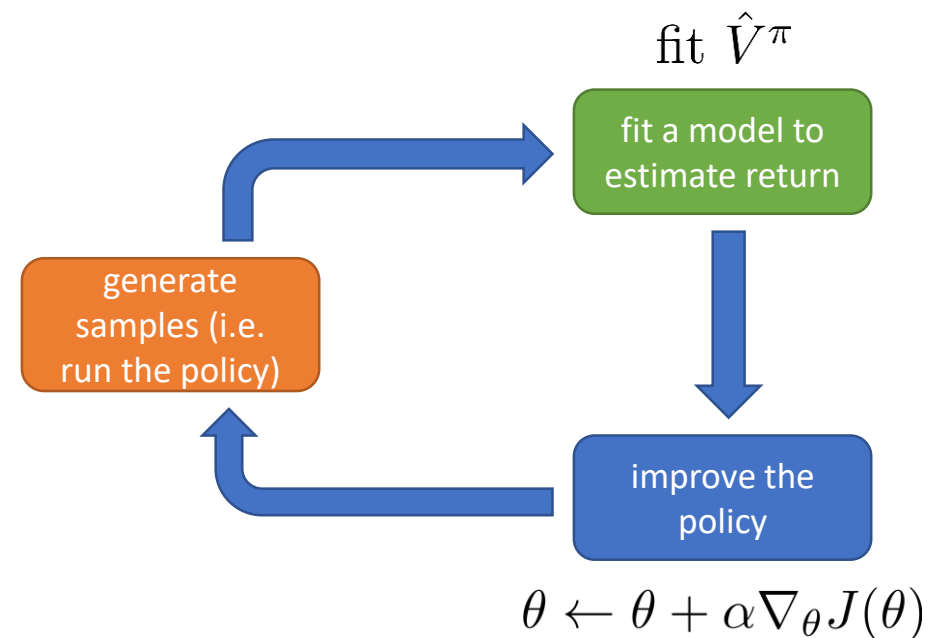
# An actor-critic algorithm

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

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

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



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

# Aside: discount factors

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

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

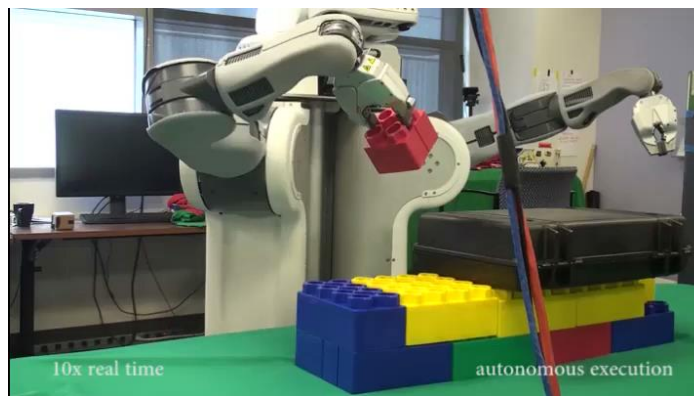
what if  $T$  (episode length) is  $\infty$ ?

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

simple trick: better to get rewards sooner than later

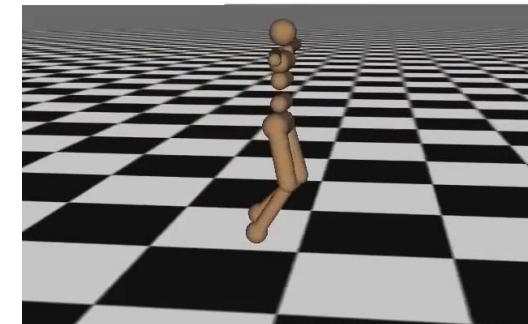
$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}_{\phi}^{\pi}(\mathbf{s}_{i,t+1})$$

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



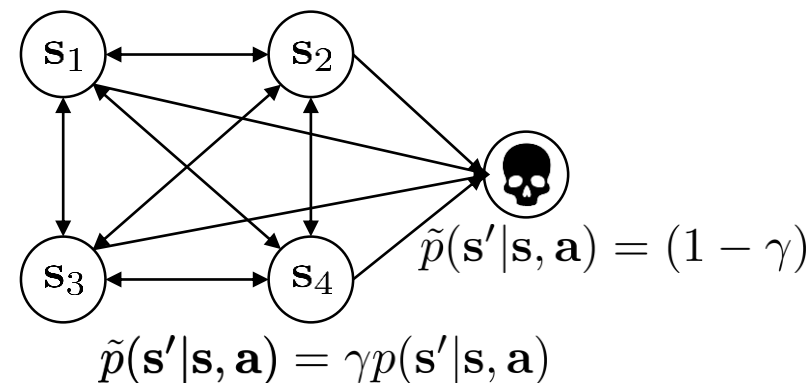
episodic tasks

Iteration 2000



continuous/cyclical tasks

$\gamma$  changes the MDP:



# Aside: discount factors for policy gradients

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

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

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

what about (Monte Carlo) policy gradients?

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

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

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

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

not the same!

# Which version is the right one?

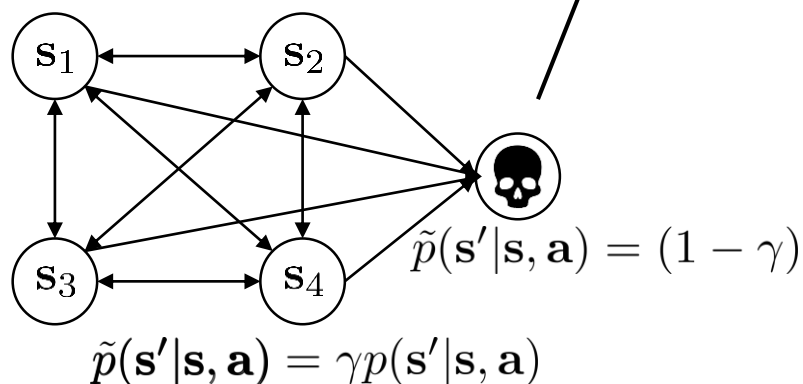
option 1: 
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left( \sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$$

option 2: 
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \gamma^{t-1} \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left( \sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$$

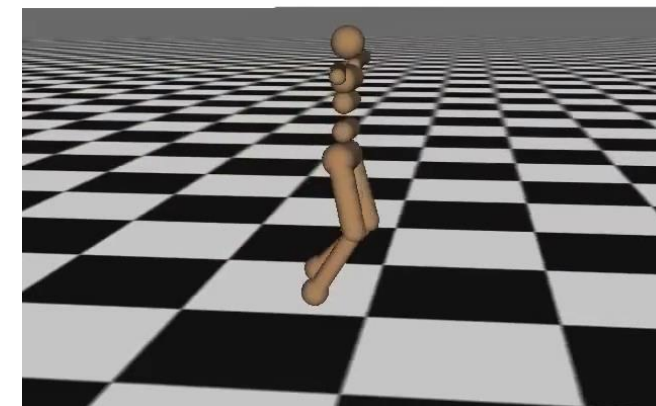
this is what we actually use...

why?

Iteration 2000



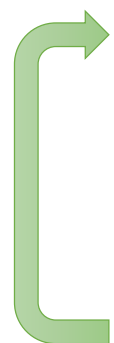
later steps don't matter if you're dead!



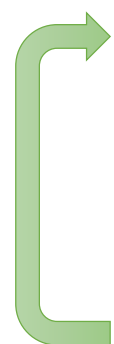


# Actor-critic algorithms (with discount)

batch actor-critic algorithm:

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

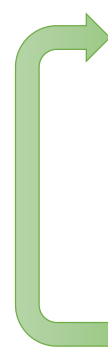
online actor-critic algorithm:

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

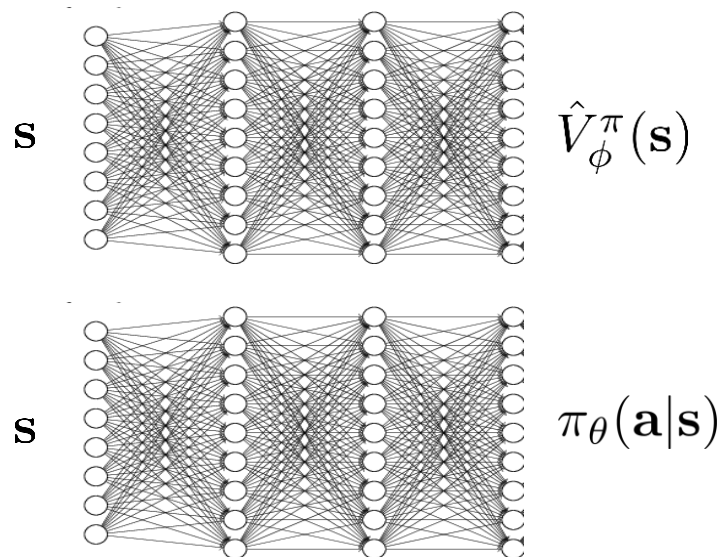
Break

# Architecture design

online actor-critic algorithm:

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

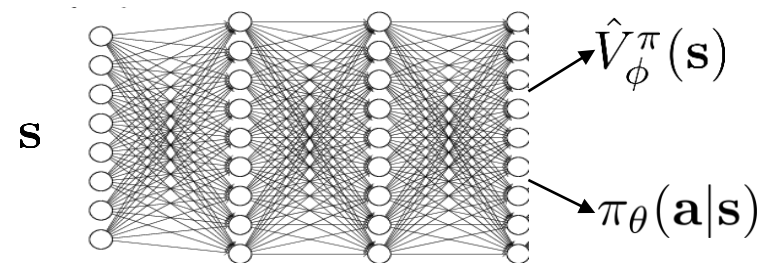
two network design



+ simple & stable

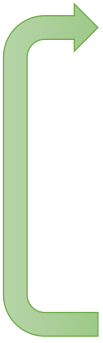
- no shared features between actor & critic

shared network design

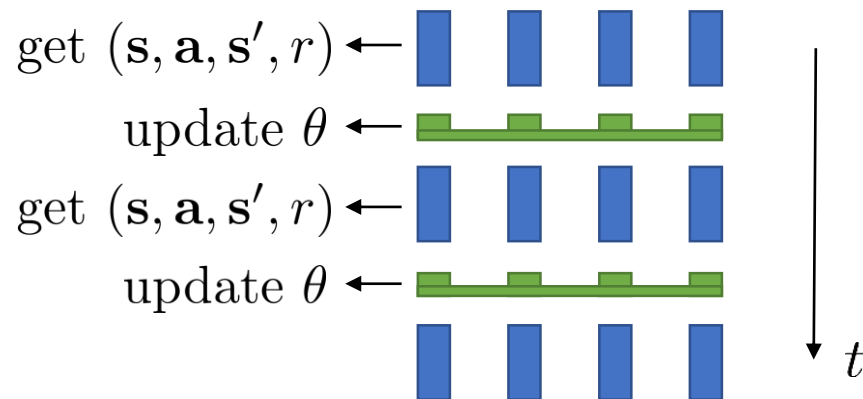


# Online actor-critic in practice

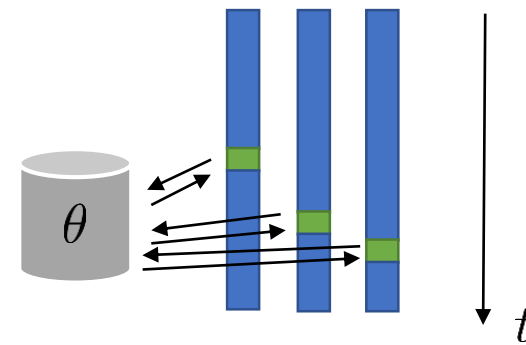
online actor-critic algorithm:

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

synchronized parallel actor-critic



asynchronous parallel actor-critic



# Critics as state-dependent baselines

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

+ lower variance (due to critic)

- not unbiased (if the critic is not perfect)

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

+ no bias

- higher variance (because single-sample estimate)

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

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

+ no bias

+ lower variance (baseline is closer to rewards)

## You'll implement this for HW2!

# Control variates: action-dependent baselines

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

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

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

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

+ no bias

- higher variance (because single-sample estimate)

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

+ goes to zero in expectation if critic is correct!

- not correct

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

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

Gu et al. 2016 (Q-Prop) – we'll talk more about variance reduction later

# Eligibility traces & n-step returns

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

+ lower variance

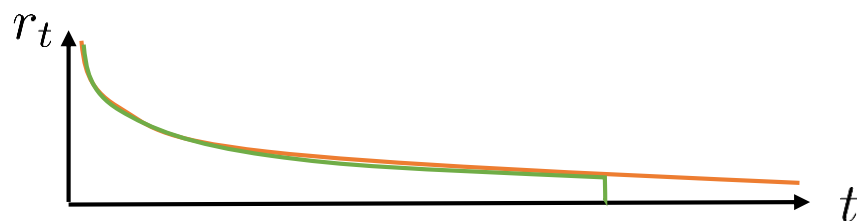
- higher bias if value is wrong (it always is)

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

+ no bias

- higher variance (because single-sample estimate)

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



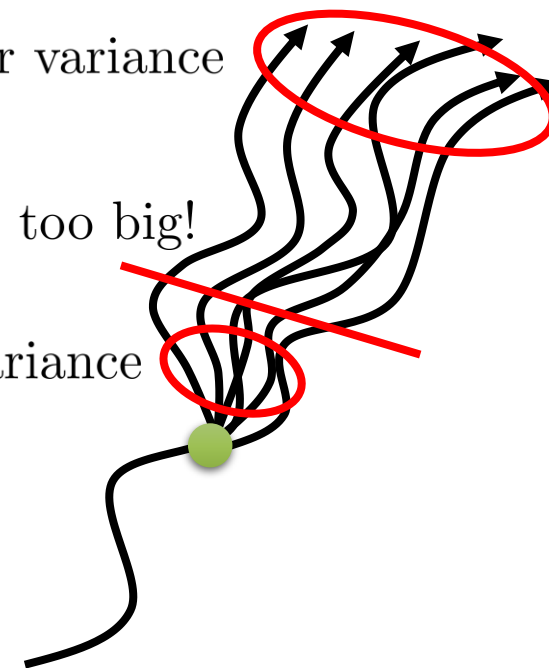
$$\hat{A}_n^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}_\phi^\pi(\mathbf{s}_t) + \gamma^n \hat{V}_\phi^\pi(\mathbf{s}_{t+n})$$

choosing  $n > 1$  often works better!

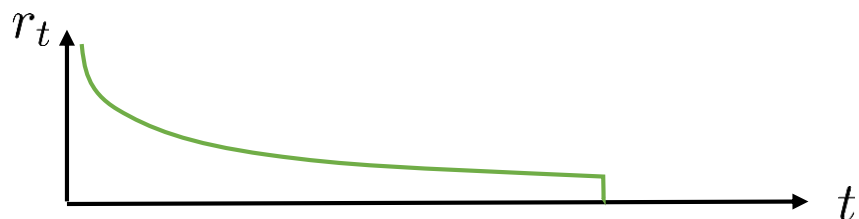
bigger variance

cut here before variance gets too big!

smaller variance



# Generalized advantage estimation



Do we have to choose just one n?

Cut everywhere all at once!

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

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

Weighted combination of n-step returns

exponential falloff

How to weight?

Mostly prefer cutting earlier (less variance)

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

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

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


similar effect as discount!

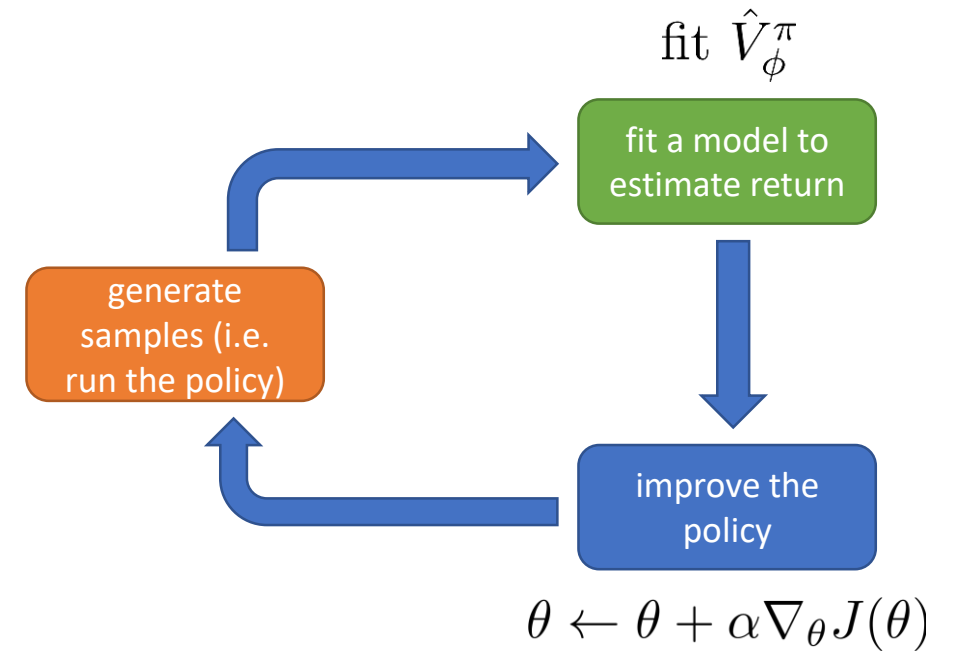
option 1:  $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \left( \sum_{t'=t}^T \gamma^{t'-t} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)$

remember this?  
discount = variance reduction!



# Review

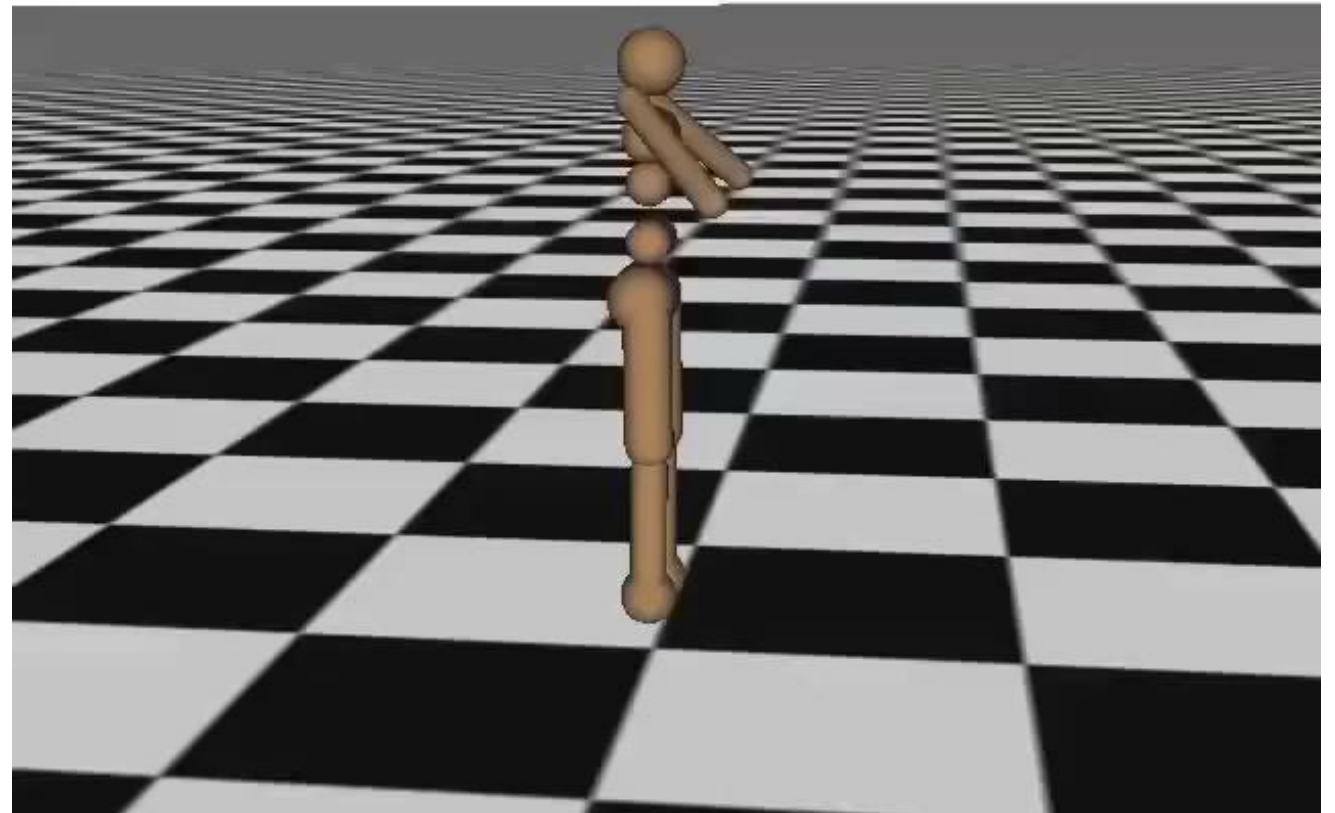
- Actor-critic algorithms:
  - Actor: the policy
  - Critic: value function
  - Reduce variance of policy gradient
- Policy evaluation
  - Fitting value function to policy
- Discount factors
  - Carpe diem Mr. Robot 
  - ...but also a variance reduction trick
- Actor-critic algorithm design
  - One network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



# Actor-critic examples

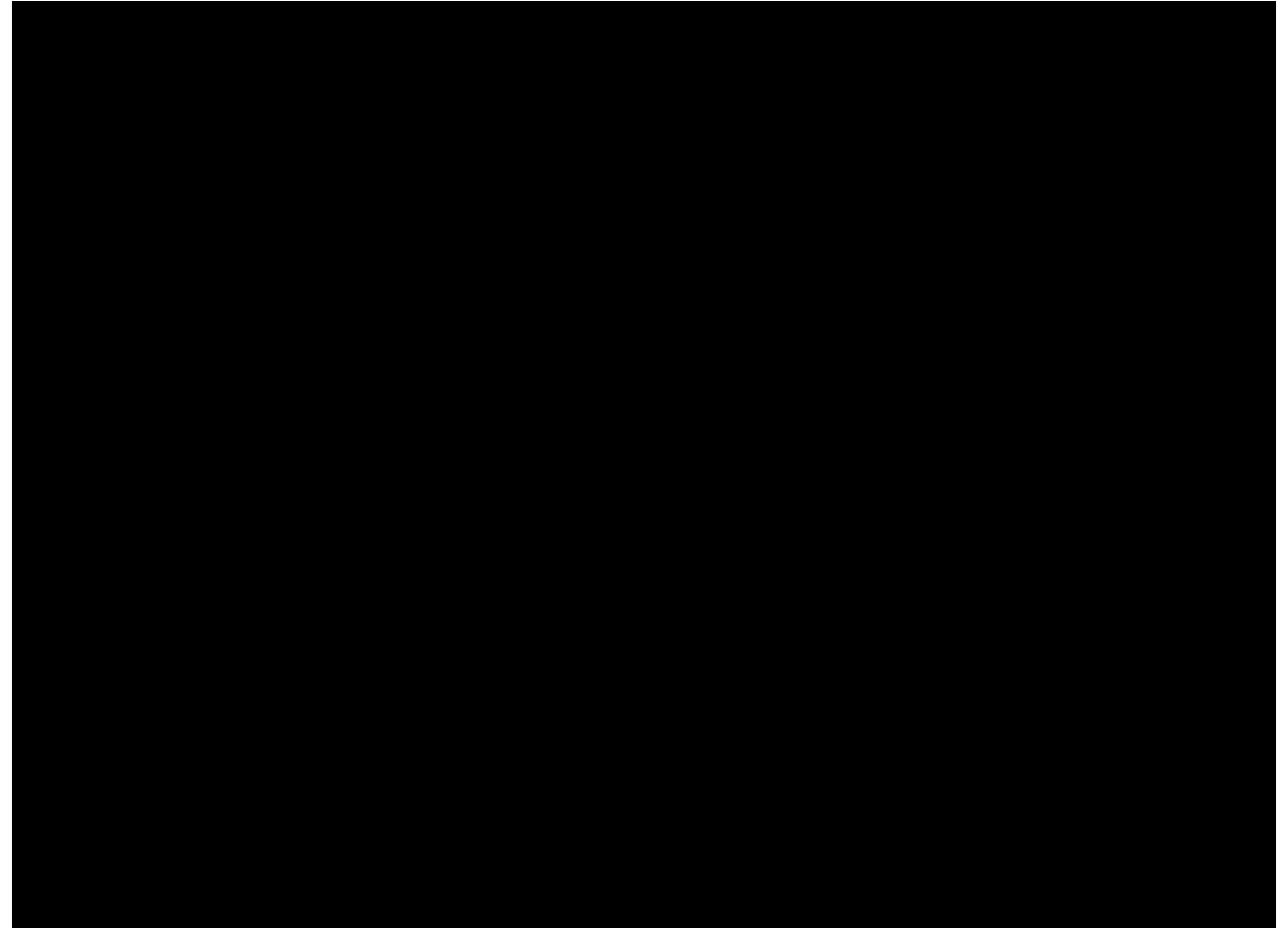
- High dimensional continuous control with generalized advantage estimation (Schulman, Moritz, L., Jordan, Abbeel '16)
- Batch-mode actor-critic
- Blends Monte Carlo and function approximator estimators (GAE)

Iteration 0



# Actor-critic examples

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



# Actor-critic suggested readings

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