

Policy Gradients

CS 294-112: Deep Reinforcement Learning

Sergey Levine

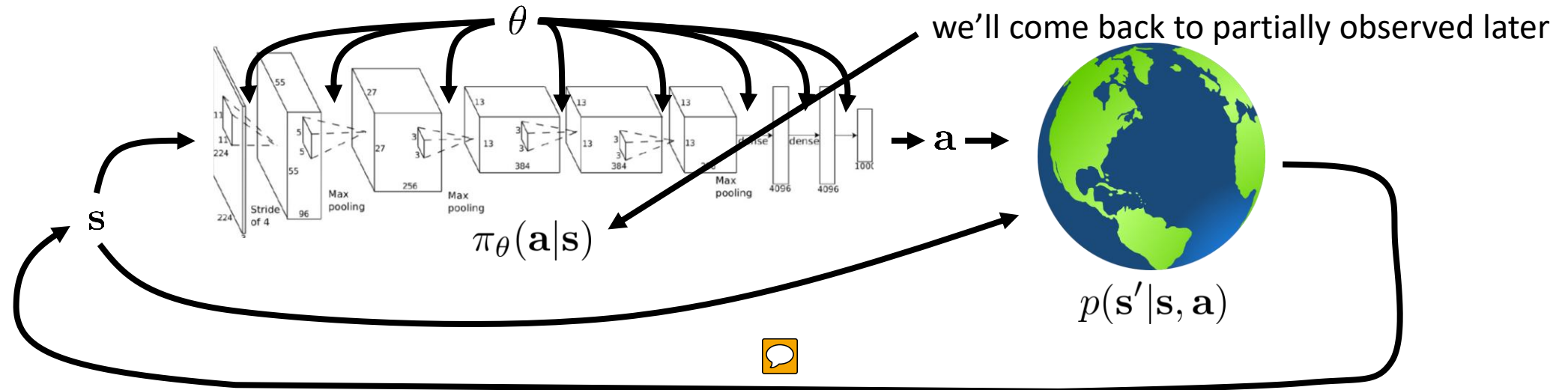
Class Notes

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

Today's Lecture

1. The policy gradient algorithm
2. What does the policy gradient do?
3. Basic variance reduction: causality
4. Basic variance reduction: baselines
5. Policy gradient examples
 - Goals:
 - Understand policy gradient reinforcement learning
 - Understand practical considerations for policy gradients

The goal of reinforcement learning



$$\underbrace{p_\theta(s_1, a_1, \dots, s_T, a_T)}_{\pi_\theta(\tau)} = p(s_1) \prod_{t=1}^T \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t)$$

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

The goal of reinforcement learning

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

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

infinite horizon case

$$\theta^* = \arg \max_{\theta} \sum_{t=1}^T E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]$$

finite horizon case

Evaluating the objective

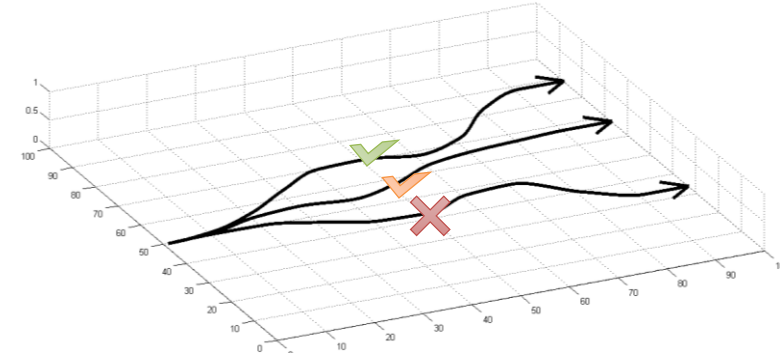
$$\theta^* = \arg \max_{\theta} \underbrace{E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]}_{J(\theta)}$$

expectation of the sum of rewards

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

sum over samples from π_{θ}

different trajectories of the agent in the environment



Direct policy differentiation

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

a convenient identity

use definition of the logarithm to rewrite equation

$$\underbrace{\pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau)} = \pi_{\theta}(\tau) \frac{\nabla_{\theta} \pi_{\theta}(\tau)}{\pi_{\theta}(\tau)} = \underbrace{\nabla_{\theta} \pi_{\theta}(\tau)}$$

$$J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} [\underbrace{r(\tau)}_{\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t)}]$$

objective is the sum of future rewards
IDEA: use gradient of the reward

$$\nabla_{\theta} J(\theta) = \int \underbrace{\nabla_{\theta} \pi_{\theta}(\tau)} r(\tau) d\tau = \int \pi_{\theta}(\tau) \underbrace{\nabla_{\theta} \log \pi_{\theta}(\tau)} r(\tau) d\tau = E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)]$$

likelihood policy gradient

Direct policy differentiation

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

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

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

log of both sides

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

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

$$\nabla_{\theta} \left[\overbrace{\log p(\mathbf{s}_1) + \sum_{t=1}^T \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) + \log p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}^{\text{= 0}} \right]$$

streiche Summenstücke die nicht von theta abhängt

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

1st sum of the gradients of the log probabilities of all the actions of the trajectory
 2nd sum: sum of the rewards of the trajectory
 -> nothing here requires the transition probabilities
 -> we just need to sample from the dynamical system without having knowledge over the system

Evaluating the policy gradient

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

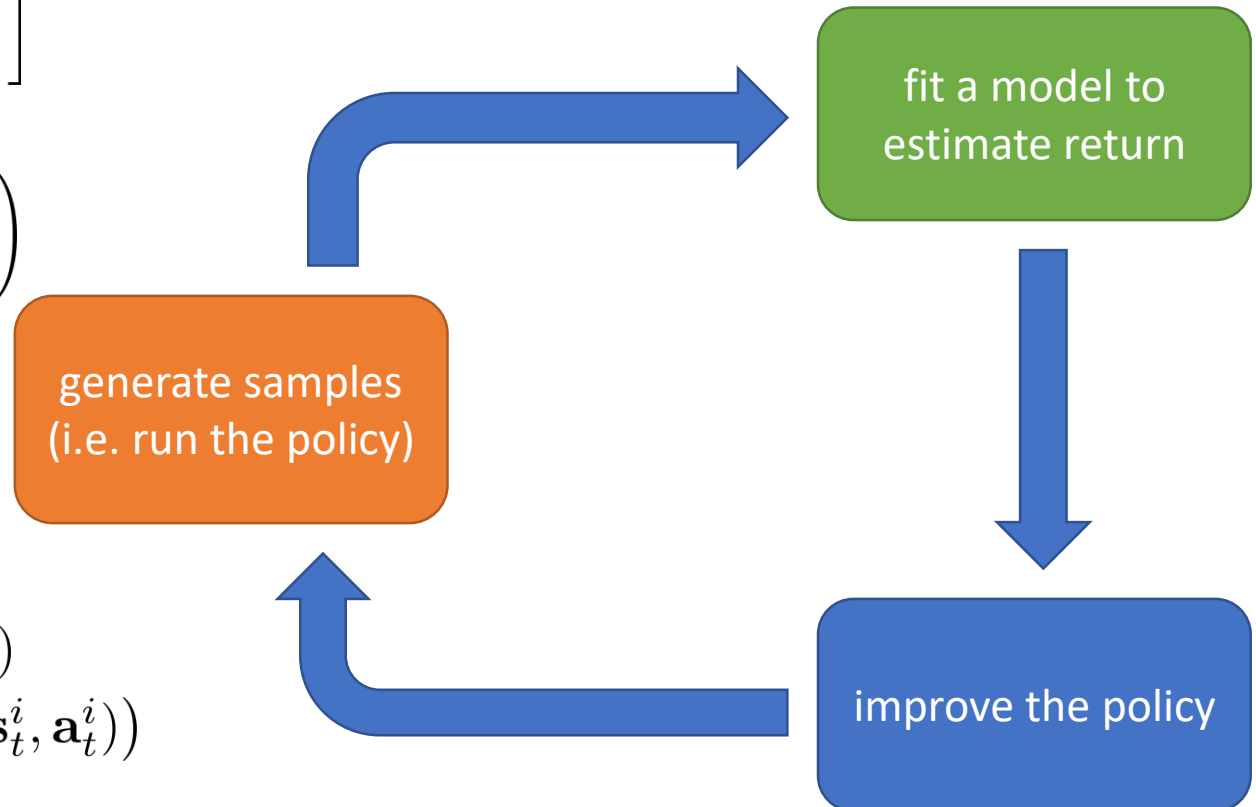
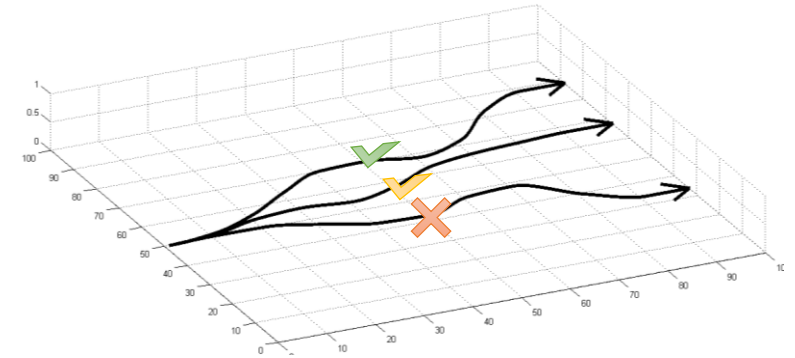
$$\nabla_{\theta} J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

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

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

REINFORCE algorithm:

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



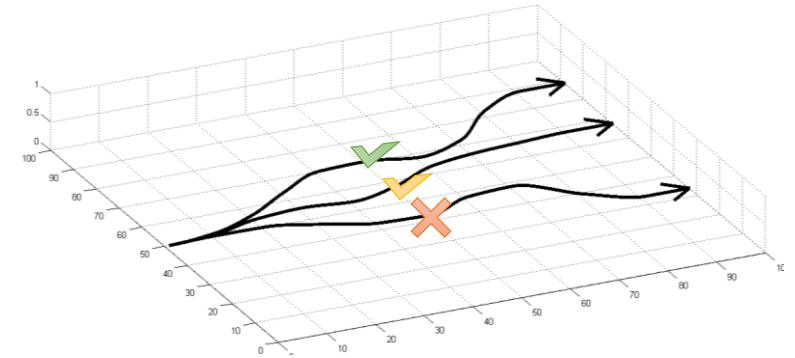
Evaluating the policy gradient

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

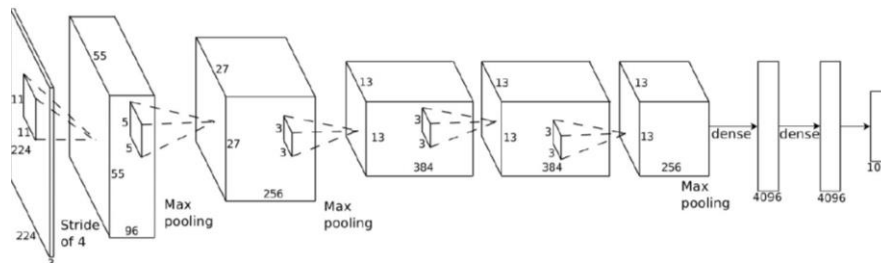
$$\nabla_{\theta} J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

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

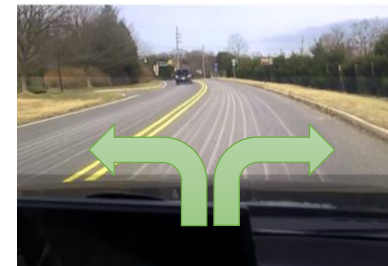
what is this?



\mathbf{s}_t



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



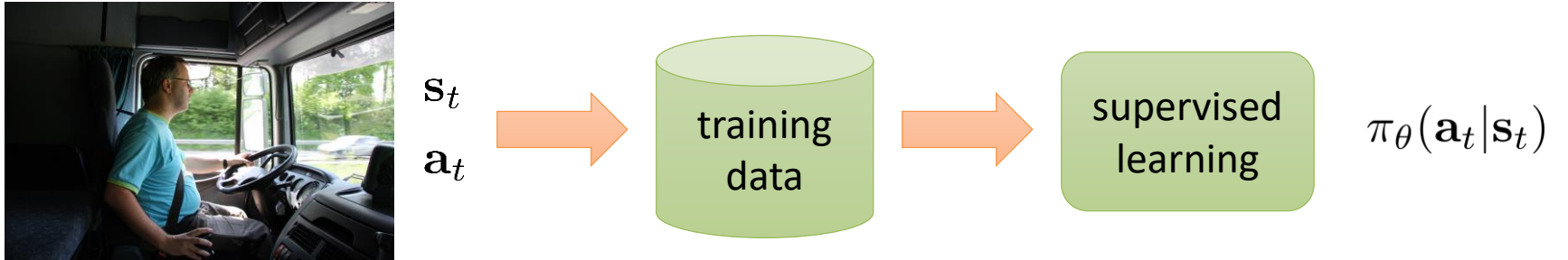
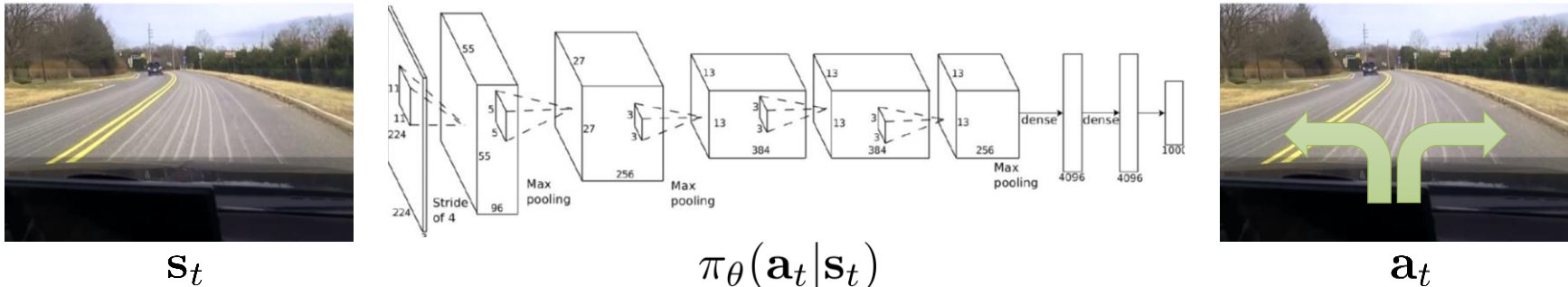
\mathbf{a}_t

Comparison to maximum likelihood

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

rewards can be understood as weights

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



Example: Gaussian policies

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

NN puts out the mean of a gaussian distribution

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

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

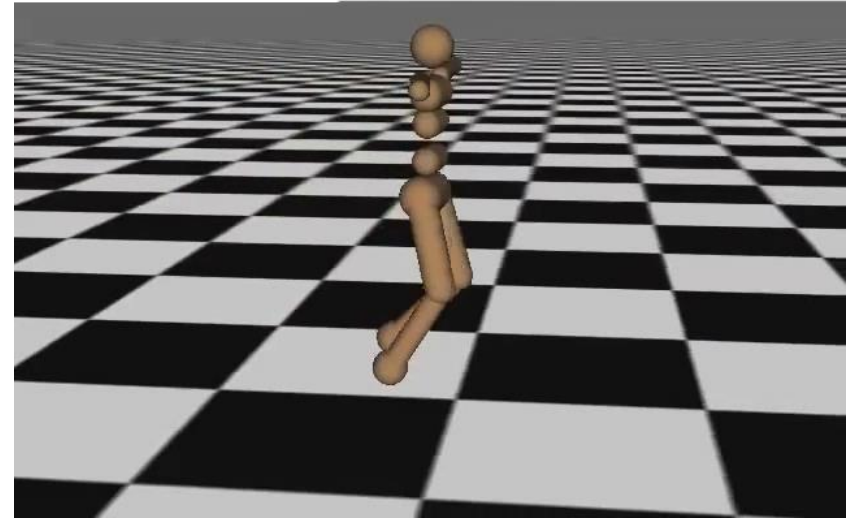
$$\nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = -\frac{1}{2} \Sigma^{-1} (f(\mathbf{s}_t) - \mathbf{a}_t) \frac{df}{d\theta}$$

just backpropagate $-\frac{1}{2} \Sigma^{-1} (f(\mathbf{s}_t) - \mathbf{a}_t) (\sum_t r(\mathbf{s}_t, \mathbf{a}_t))$

REINFORCE algorithm:

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

Iteration 2000



What did we just do?

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

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

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

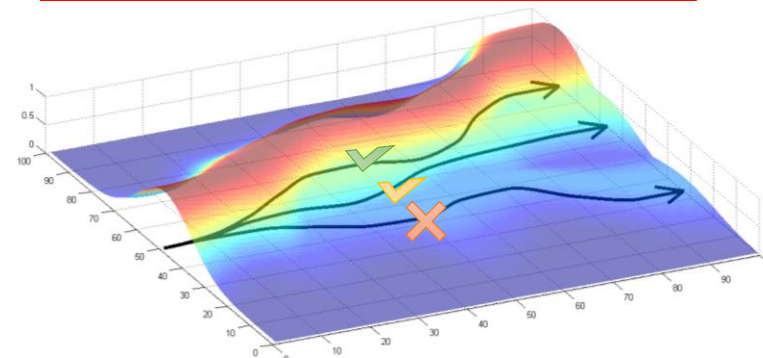
bad stuff is made less likely

simply formalizes the notion of “trial and error”!

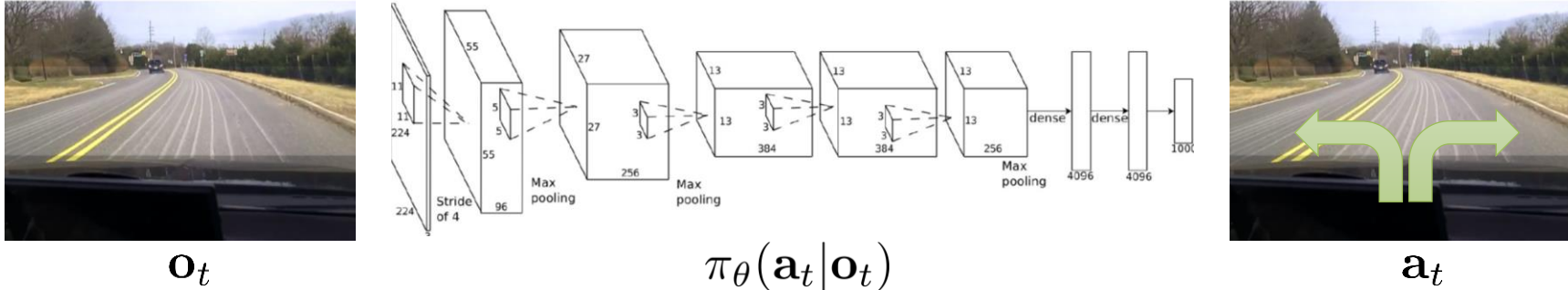
REINFORCE algorithm:

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

increase probability of the good trajectories
decrease probability of bad trajectories



Partial observability



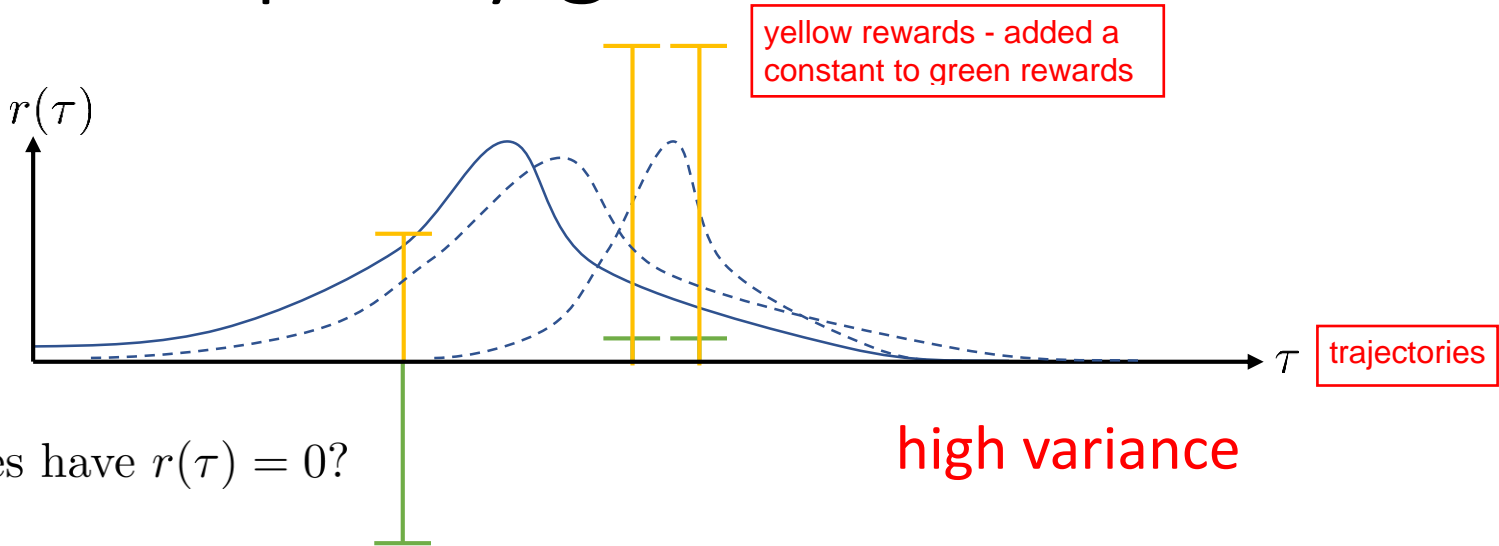
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{o}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{o}_{i,t}, \mathbf{a}_{i,t}) \right)$$

Markov property is not actually used!

Can use policy gradient in partially observed MDPs without modification

What is wrong with the policy gradient?

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



even worse: what if the two “good” samples have $r(\tau) = 0$?

1st problem - for a finite amount of samples - the amount of samples has a high impact on high variance

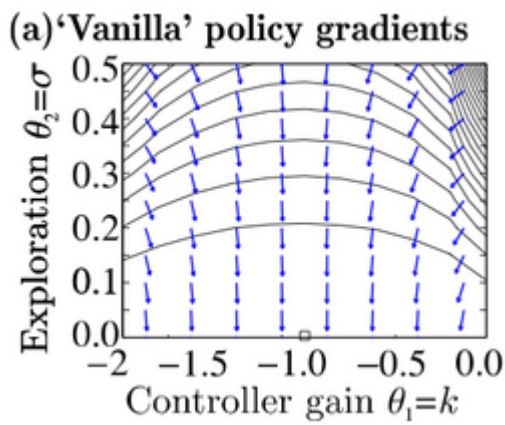
$$\log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) = -\frac{1}{2\sigma^2} (k\mathbf{s}_t - \mathbf{a}_t)^2 + \text{const} \quad \theta = (k, \sigma)$$

$$r(\mathbf{s}_t, \mathbf{a}_t) = -\mathbf{s}_t^2 - \mathbf{a}_t^2$$

linear state transition with k

reward = move state to 0
and use small actions
LQR - linear quadratic regulator

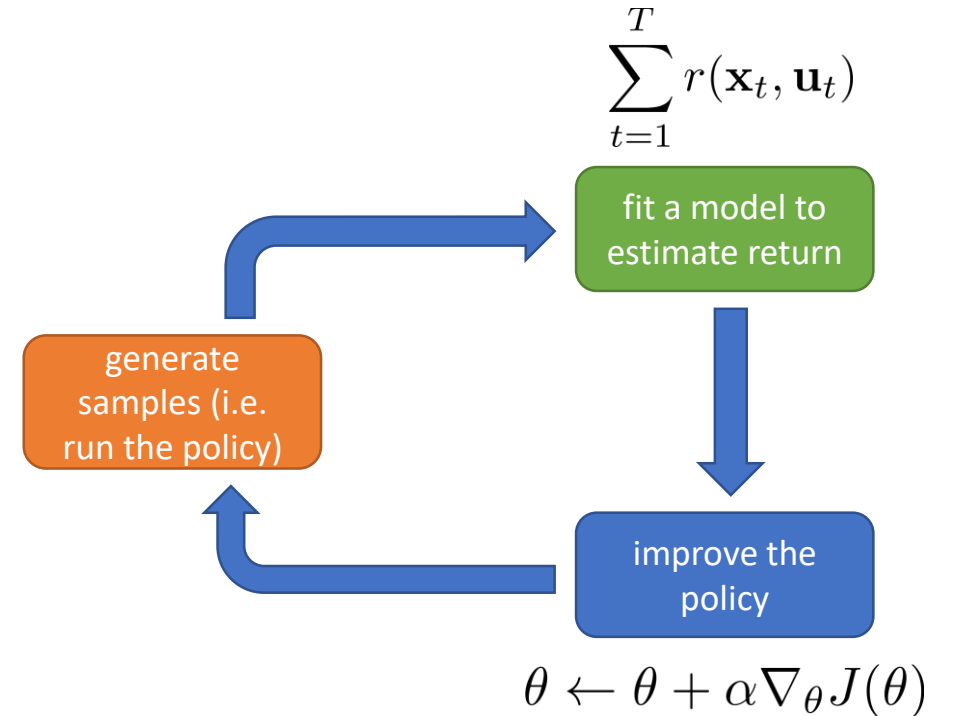
slow convergence
hard to choose learning rate



(image from Peters & Schaal 2008)

Review

- Evaluating the RL objective
 - Generate samples
- Evaluating the policy gradient
 - Log-gradient trick identity - trick
 - Generate samples
- Understanding the policy gradient
 - Formalization of trial-and-error
- Partial observability
 - Works just fine
- What is wrong with policy gradient?



Break

Reducing variance

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

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

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

1 turns into t
for the subscript in the sum
Meaning rewards from t to the end of the episode

“reward to go”

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

Baselines

a convenient identity

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

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

subtract the average reward for the policy gradient
in order to make it more reliable - less variance

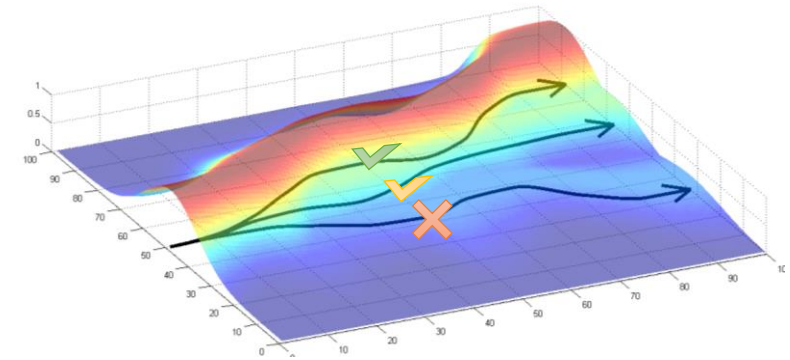
$$b = \frac{1}{N} \sum_{i=1}^N r(\tau) \quad \text{but... are we *allowed* to do that??}$$

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

π is probability distribution
integral of probability distribution = 1

subtracting a baseline is *unbiased* in expectation!

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



Analyzing variance

can we write down the variance?

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

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

$$\text{Var} = E_{\tau \sim \pi_{\theta}(\tau)} [(\nabla_{\theta} \log \pi_{\theta}(\tau) (r(\tau) - b))^2] - \underbrace{E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) (r(\tau) - b)]^2}_{\text{this bit is just } E_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)]}$$

(baselines are unbiased in expectation)

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

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

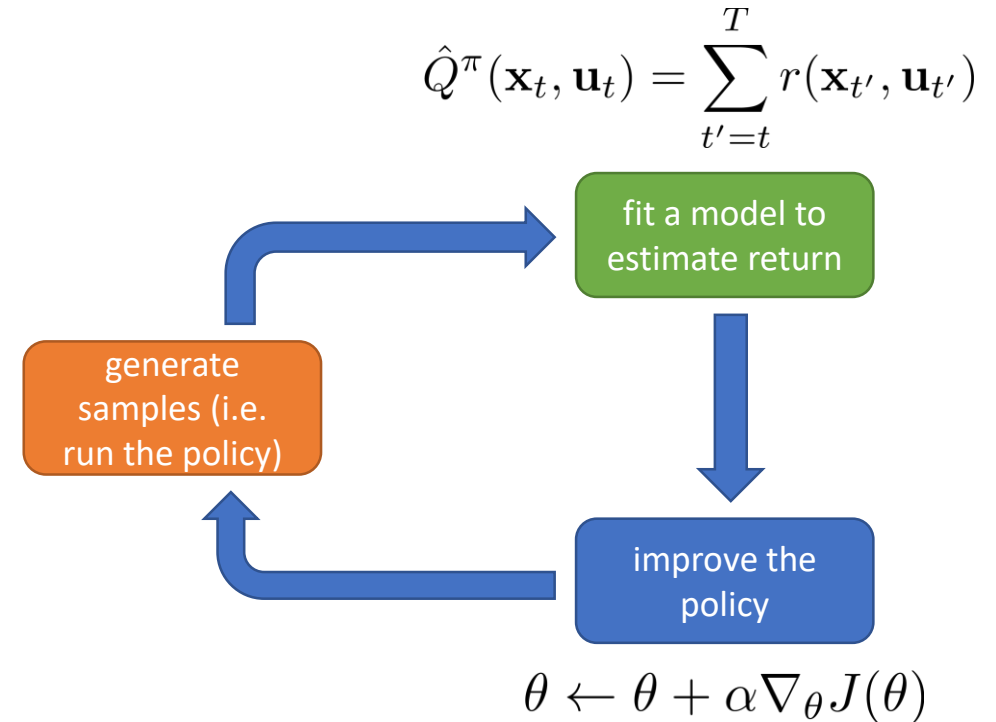
← This is just expected reward, but weighted by gradient magnitudes!

baseline step is updated after every gradient step.
either mean changes because of new distributions
or optimal baseline changes because expectation changes

optimal base line for policy gradient
in practice - not a big difference between constant baseline and optimal baseline

Review

- The high variance of policy gradient
- Exploiting causality
 - Future doesn't affect the past
- Baselines
 - Unbiased!
- Analyzing variance
 - Can derive optimal baselines



Policy gradient is on-policy

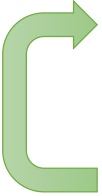
$$\theta^* = \arg \max_{\theta} J(\theta)$$

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

$$\nabla_{\theta} J(\theta) = \underbrace{E_{\tau \sim \pi_{\theta}(\tau)}[\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)]}_{\text{this is trouble...}}$$

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

REINFORCE algorithm:

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

can't just skip this!



Off-policy learning & importance sampling

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

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

what if we don't have samples from $\pi_{\theta}(\tau)$?

(we have samples from some $\bar{\pi}(\tau)$ instead) maybe old policy

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

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

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

importance sampling

$$E_{x \sim p(x)}[f(x)] = \int p(x) f(x) dx$$

$$= \int \frac{q(x)}{q(x)} p(x) f(x) dx$$

$$= \int q(x) \frac{p(x)}{q(x)} f(x) dx$$

$$= E_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right]$$

summary of
importance
sampling
- draw samples
from $q(x)$ and
distribute them
under $p(x)/q(x)$

we don't have to know anything
about dynamics in order to compute
importance weights
we just need new samples

Deriving the policy gradient with IS

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

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

a convenient identity

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

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

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

the only bit that depends on θ'

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

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

The off-policy policy gradient

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

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

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

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

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

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

future actions don't affect current weight

A first-order approximation for IS (preview)

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

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

$$\theta^* = \arg \max_{\theta} \sum_{t=1}^T \underbrace{E_{(\mathbf{s}_t, \mathbf{a}_t) \sim p_{\theta}(\mathbf{s}_t, \mathbf{a}_t)} [r(\mathbf{s}_t, \mathbf{a}_t)]}_{\text{expectation under state-action marginal}}$$


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

$$J(\theta') = \sum_{t=1}^T E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[\cancel{\frac{p_{\theta'}(\mathbf{s}_t)}{p_{\theta}(\mathbf{s}_t)}} E_{\mathbf{a}_t \sim \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} r(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \quad \text{We'll see why this is reasonable later in the course!}$$

ignore this part

Policy gradient with automatic differentiation

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

 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!

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

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)

Policy gradient with automatic differentiation

Pseudocode example (with discrete actions):

Maximum likelihood:

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
loss = tf.reduce_mean(negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

Policy gradient with automatic differentiation

Pseudocode example (with discrete actions):

Policy gradient:

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# q_values - (N*T) x 1 tensor of estimated state-action values
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
weighted_negative_likelihoods = tf.multiply(negative_likelihoods, q_values)
loss = tf.reduce_mean(weighted_negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

this is in a loop, where new samples are generated after gradient is applied to theta

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

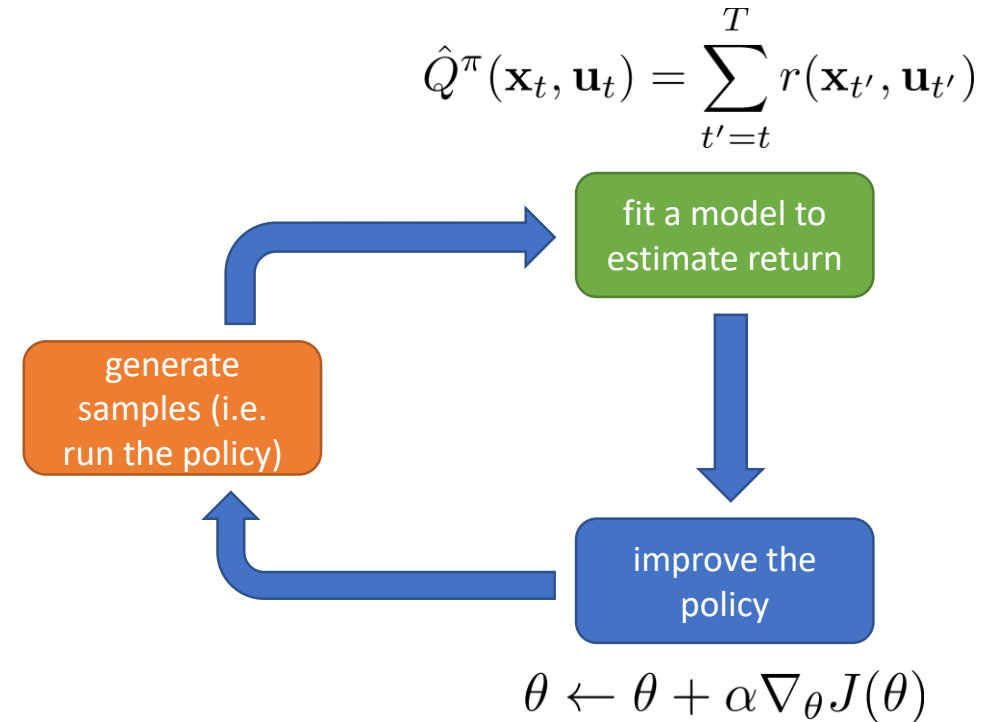
q_values

Policy gradient in practice

- Remember that the gradient has high variance
 - This isn't the same as supervised learning!
 - Gradients will be really noisy!
- Consider using much larger batches
- Tweaking learning rates is very hard
 - Adaptive step size rules like ADAM can be OK-ish
 - We'll learn about policy gradient-specific learning rate adjustment methods later!

Review

- Policy gradient is on-policy
- Can derive off-policy variant
 - Use importance sampling
 - Exponential scaling in T
 - Can ignore state portion (approximation)
- Can implement with automatic differentiation – need to know what to backpropagate
- Practical considerations: batch size, learning rates, optimizers



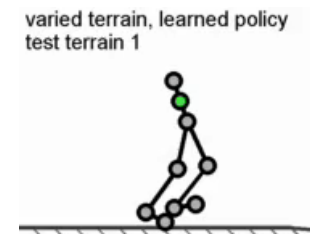
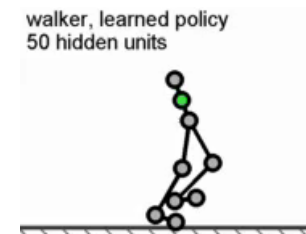
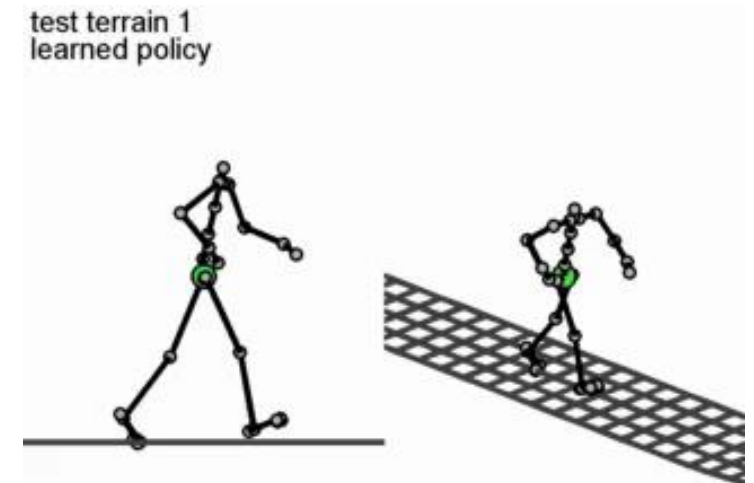
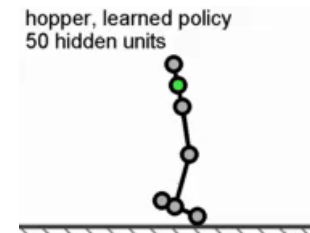
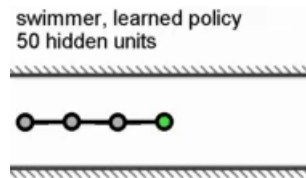
Advanced policy gradient topics

- What more is there?
- Next time: introduce value functions and Q-functions
- Later in the class: natural gradient and automatic step size adjustment

Example: policy gradient with importance sampling

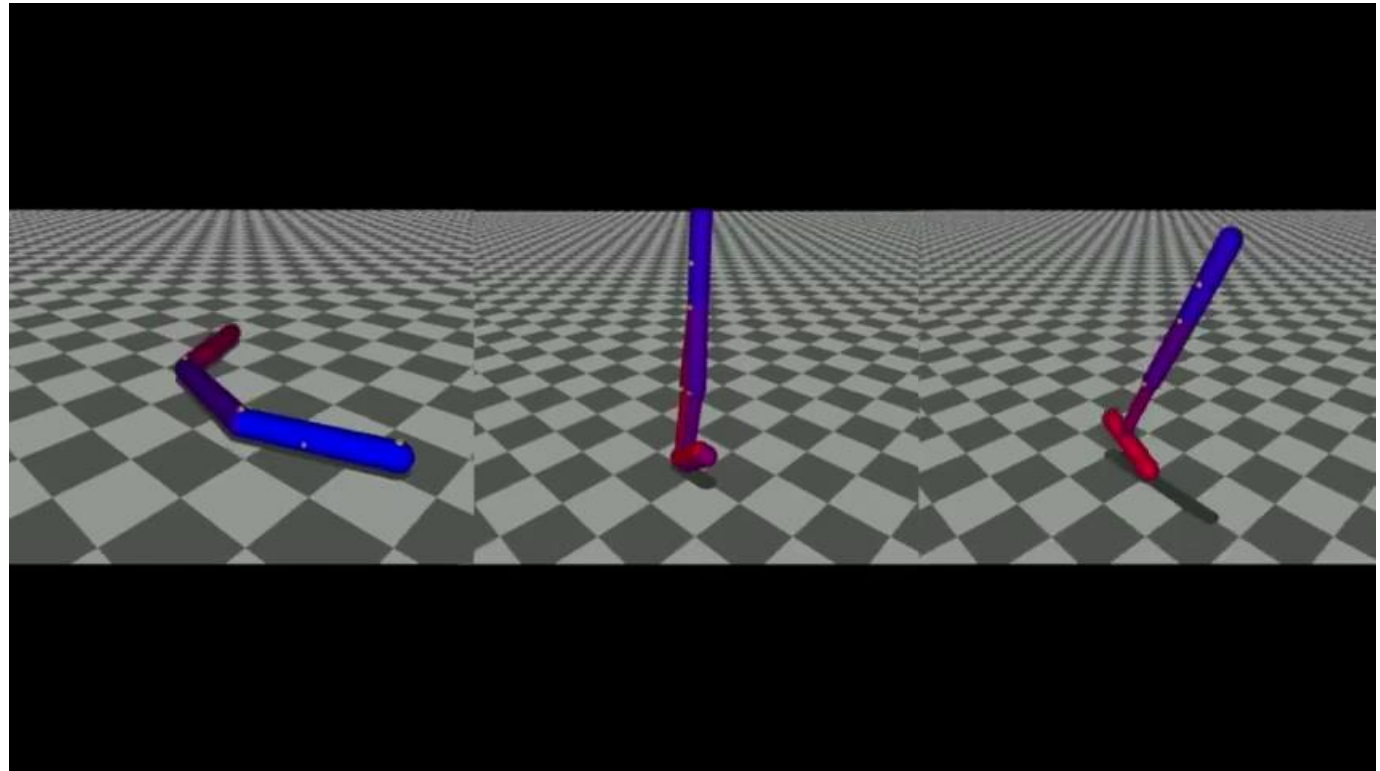
$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_{\theta}(\tau)} \left[\sum_{t=1}^T \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \left(\prod_{t'=1}^t \frac{\pi_{\theta'}(\mathbf{a}_{t'} | \mathbf{s}_{t'})}{\pi_{\theta}(\mathbf{a}_{t'} | \mathbf{s}_{t'})} \right) \left(\sum_{t'=t}^T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

- Incorporate example demonstrations using importance sampling
- Neural network policies



Example: trust region policy optimization

- Natural gradient with automatic step adjustment (we'll learn about this later)
- Discrete and continuous actions
- Code available (see Duan et al. '16)



Policy gradients suggested readings

- Classic papers
 - Williams (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning: introduces REINFORCE algorithm
 - Baxter & Bartlett (2001). Infinite-horizon policy-gradient estimation: temporally decomposed policy gradient (not the first paper on this! see actor-critic section later)
 - Peters & Schaal (2008). Reinforcement learning of motor skills with policy gradients: very accessible overview of optimal baselines and natural gradient
- Deep reinforcement learning policy gradient papers
 - Levine & Koltun (2013). Guided policy search: deep RL with importance sampled policy gradient (unrelated to later discussion of guided policy search)
 - Schulman, L., Moritz, Jordan, Abbeel (2015). Trust region policy optimization: deep RL with natural policy gradient and adaptive step size
 - Schulman, Wolski, Dhariwal, Radford, Klimov (2017). Proximal policy optimization algorithms: deep RL with importance sampled policy gradient

paper that
suggest taking
rewards from t'
= t instead of 1
(beginning of
episode)