

UC Berkeley · CSW182 | [Deep Learning]

Designing, Visualizing and Understanding Deep Neural Networks (2021)

CSW182 (2021) · 课程资料包 @ShowMeAI



视频
中英双语字幕



课件
一键打包下载



笔记
官方笔记翻译



代码
作业项目解析



视频 · B 站 [扫码或点击链接]
<https://www.bilibili.com/video/BV1Ff4y1n7ar>



课件 & 代码 · 博客 [扫码或点击链接]
<http://blog.showmeai.tech/berkeley-csw182>

Berkeley
Q-Learning
计算机视觉

循环神经网络
风格迁移
机器学习基础

可视化
模仿学习
生成模型

梯度策略
元学习
卷积网络

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2：扫码点击**底部菜单栏**
称为 **AI 内容创作者**？回复 [添砖加瓦]

Reinforcement Learning

Designing, Visualizing and Understanding Deep Neural Networks

CS W182/282A

Instructor: Sergey Levine
UC Berkeley



From *prediction* to *control*



- i.i.d. distributed data (each datapoint is independent)
- ground truth supervision
- objective is to predict the right label

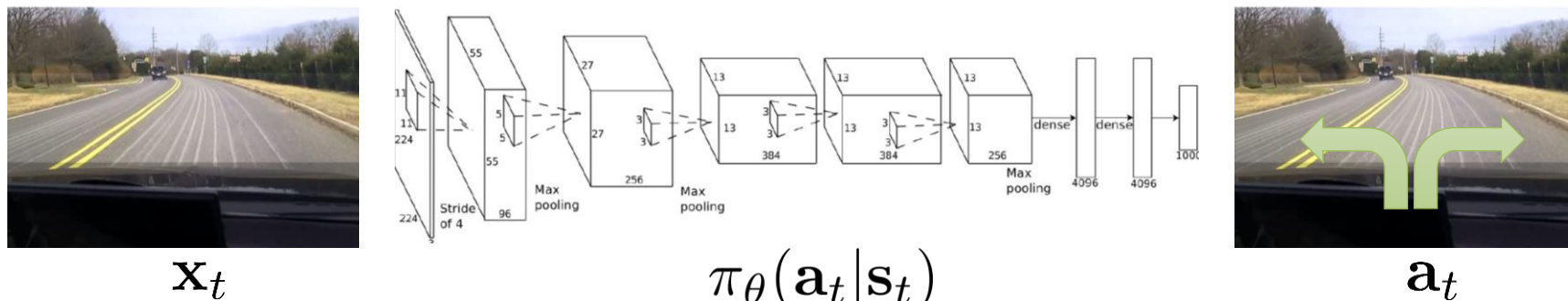
These are not **just** issues for control: in many cases, real-world deployment of ML has these same **feedback** issues

Example: decisions made by a traffic prediction system might affect the route that people take, which changes traffic



- each decision can change future inputs (not independent)
- supervision may be high-level (e.g., a goal)
- objective is to accomplish the task

How do we specify what we *want*?



which action is better or worse?

note that we switched to states here,
we'll come back to observations later

$r(\mathbf{s}, \mathbf{a})$: reward function

tells us which states and actions are better

We want to maximize
reward **over all time**, not
just greedily!



high reward



low reward

Definitions

Markov decision process

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$$

\mathcal{S} – state space

states $s \in \mathcal{S}$ (discrete or continuous)

\mathcal{A} – action space

actions $a \in \mathcal{A}$ (discrete or continuous)

r – reward function

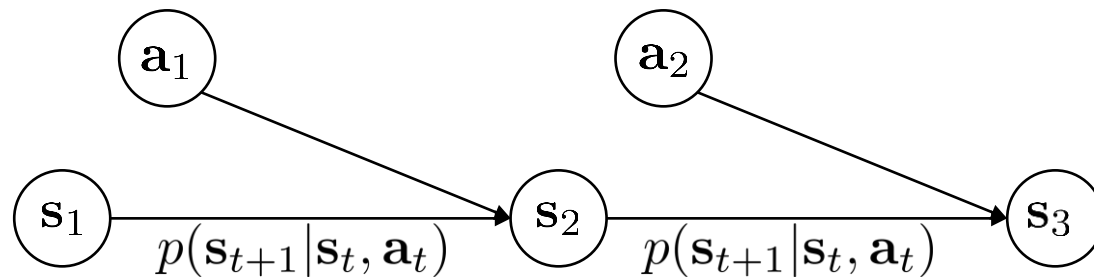
$$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$



Andrey Markov



Richard Bellman



transition **operator** or transition **probability**
or **dynamics** (all synonyms)

Definitions

Markov decision process

\mathcal{S} – state space

\mathcal{A} – action space

r – reward function

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, r\}$$

states $s \in \mathcal{S}$ (discrete or continuous)

actions $a \in \mathcal{A}$ (discrete or continuous)

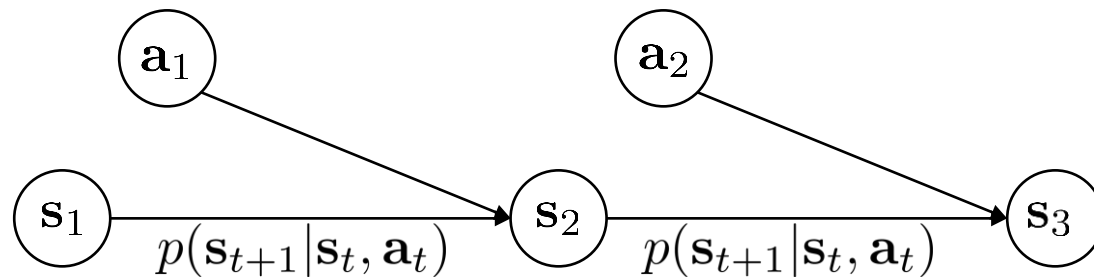
$$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$



Andrey Markov



Richard Bellman



transition **operator** or transition **probability**
or **dynamics** (all synonyms)

also referred to as $\mathcal{T}(s_{t+1} | s_t, a_t)$

Definitions

partially observed Markov decision process

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{E}, r\}$$

\mathcal{S} – state space

states $s \in \mathcal{S}$ (discrete or continuous)

\mathcal{A} – action space

actions $a \in \mathcal{A}$ (discrete or continuous)

\mathcal{O} – observation space

observations $o \in \mathcal{O}$ (discrete or continuous)

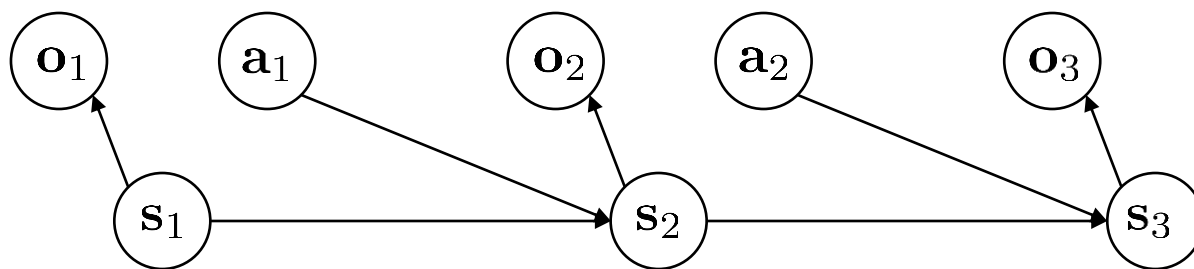
\mathcal{T} – transition operator (like before)

\mathcal{E} – emission probability $p(o_t|s_t)$

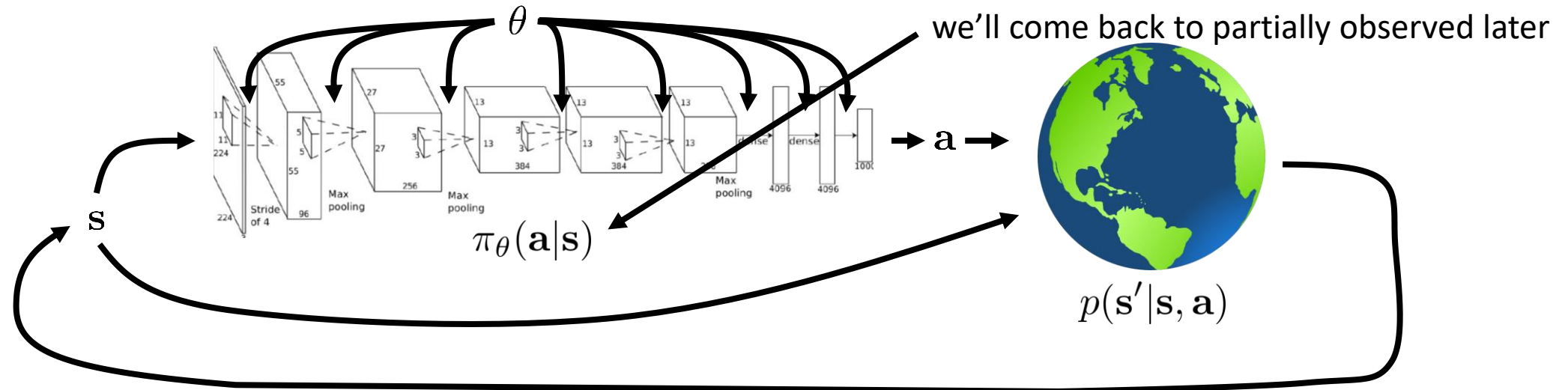
For now, we'll stick to regular
(fully observed) MDPs, but you
should know that this exists

r – reward function

$$r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$



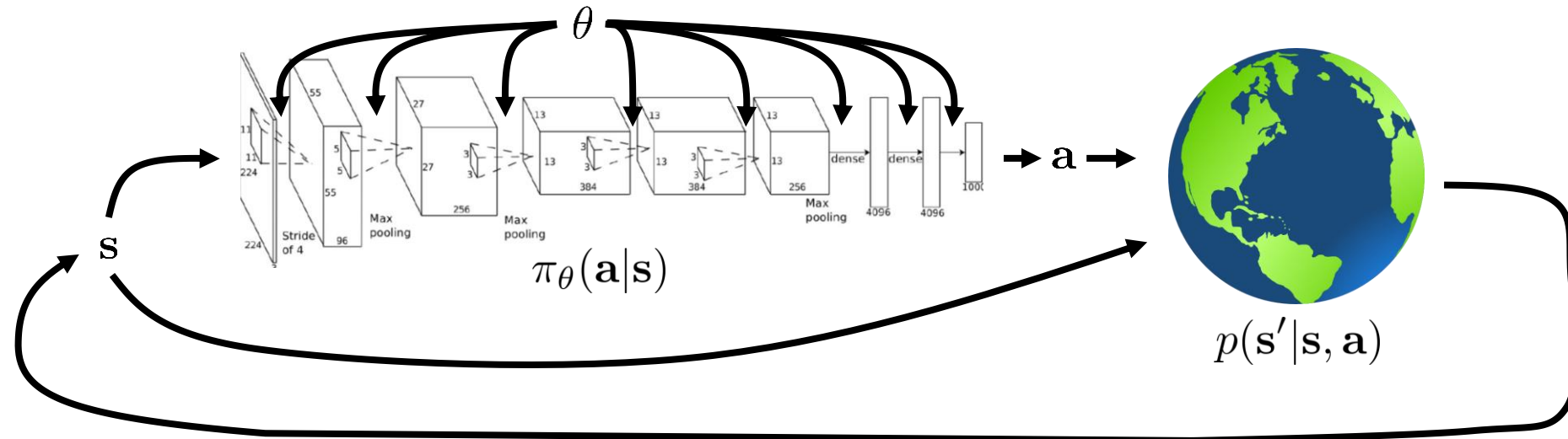
The goal of reinforcement learning



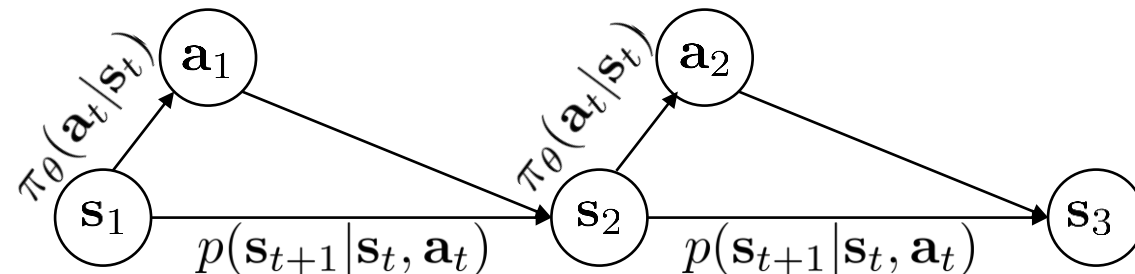
$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{a}_T)}_{p_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^T \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

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

The goal of reinforcement learning

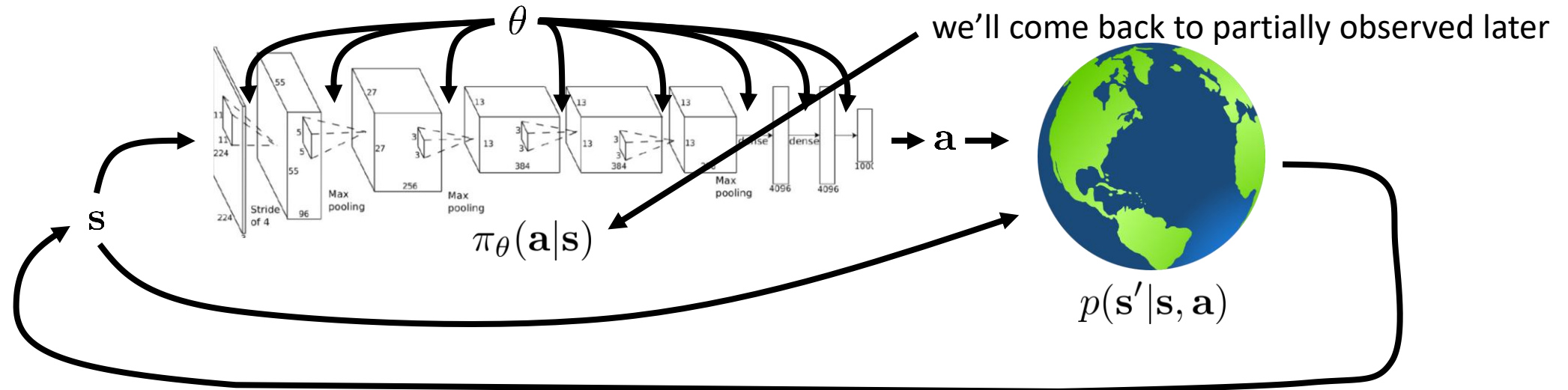


$$\underbrace{p_\theta(s_1, a_1, \dots, s_T, a_T)}_{p_\theta(\tau)} = p(s_1) \prod_{t=1}^T \underbrace{\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)}_{\text{Markov chain on } (s, a)}$$



Wall of math time (policy gradients)

The goal of reinforcement learning



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

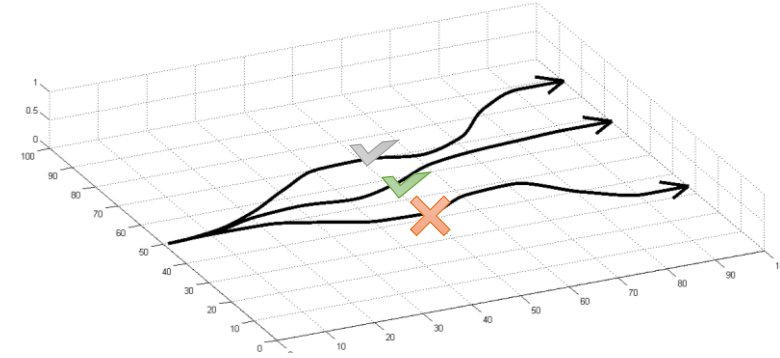
$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Evaluating the objective

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

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

sum over samples from π_{θ}



Direct policy differentiation

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

a convenient identity

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

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

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

Direct policy differentiation

$$\begin{aligned}
 \theta^* &= \arg \max_{\theta} J(\theta) \\
 J(\theta) &= E_{\tau \sim \pi_{\theta}(\tau)}[r(\tau)] \\
 \nabla_{\theta} J(\theta) &= E_{\tau \sim \pi_{\theta}(\tau)}[\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)]
 \end{aligned}$$

log of both sides

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

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

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

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

Evaluating the policy gradient

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

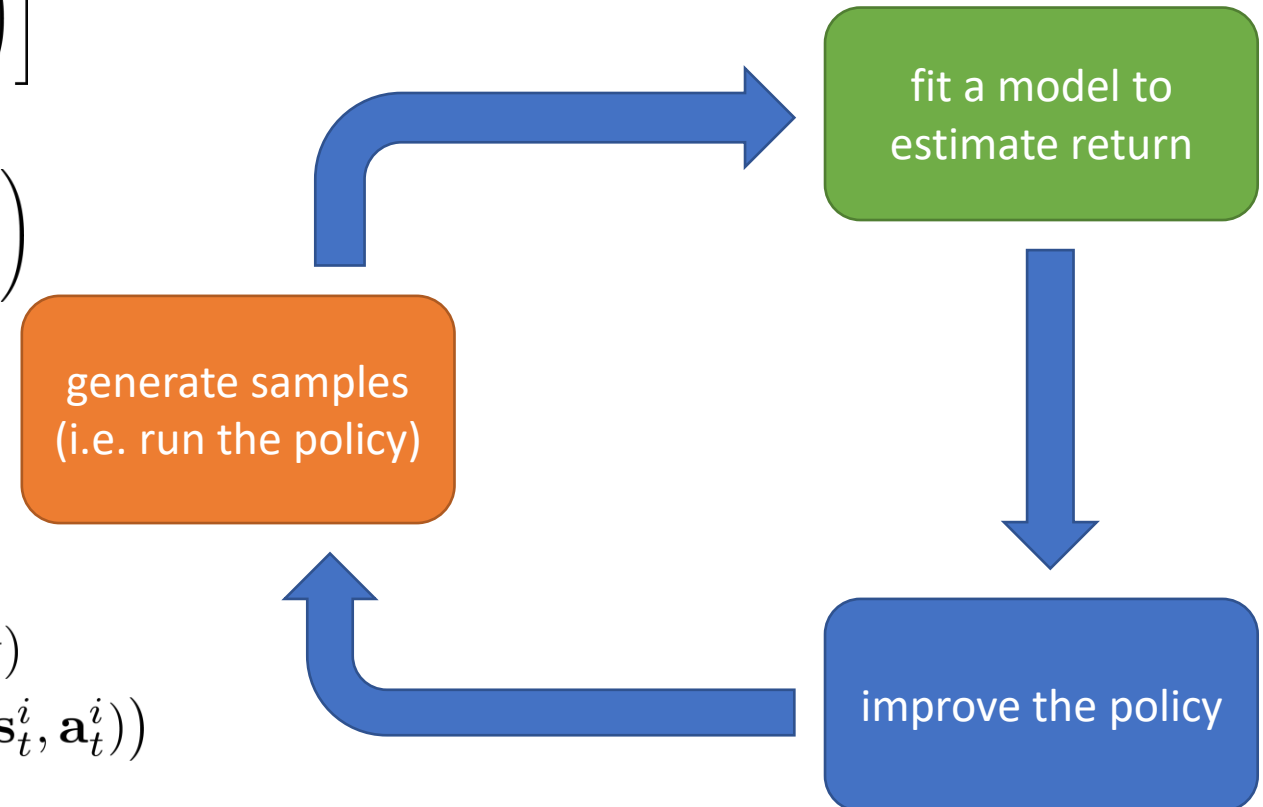
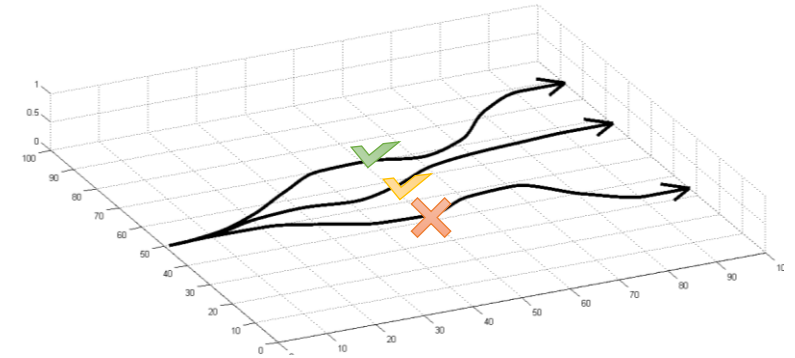
$$\nabla_{\theta} J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

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

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

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



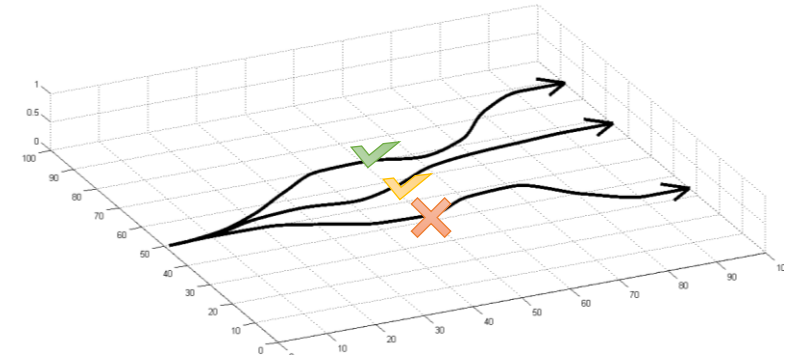
Evaluating the policy gradient

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

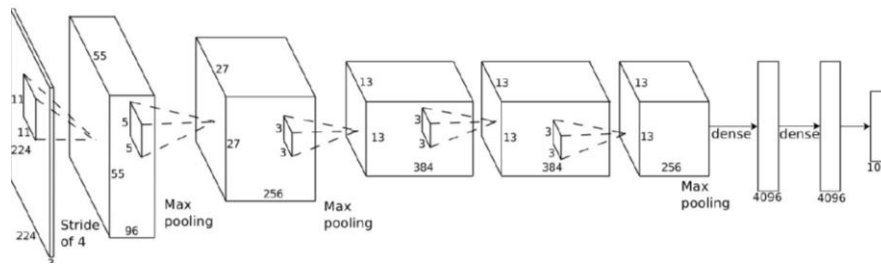
$$\nabla_{\theta} J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

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

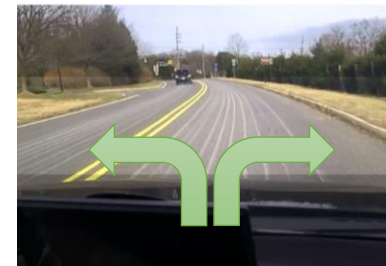
what is this?



\mathbf{s}_t



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

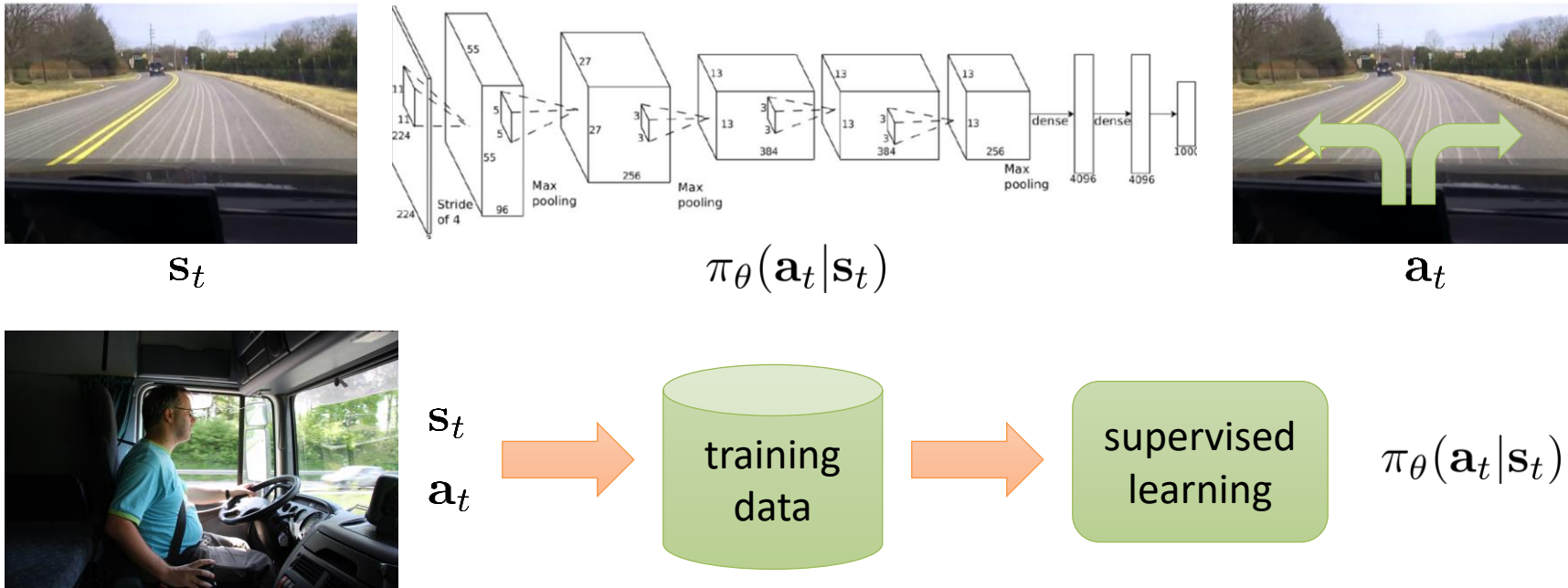


\mathbf{a}_t

Comparison to maximum likelihood

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

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



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

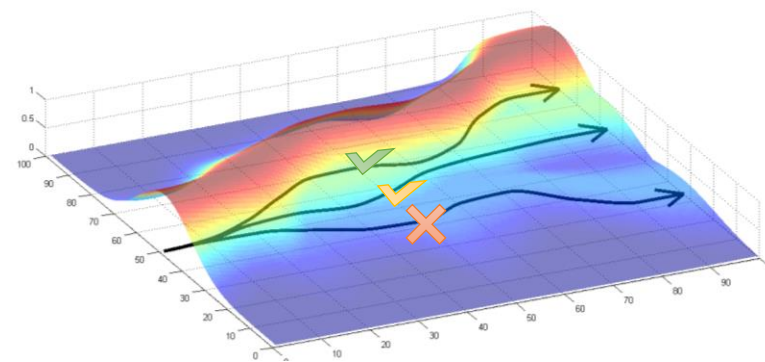
good stuff is made more likely

bad stuff is made less likely

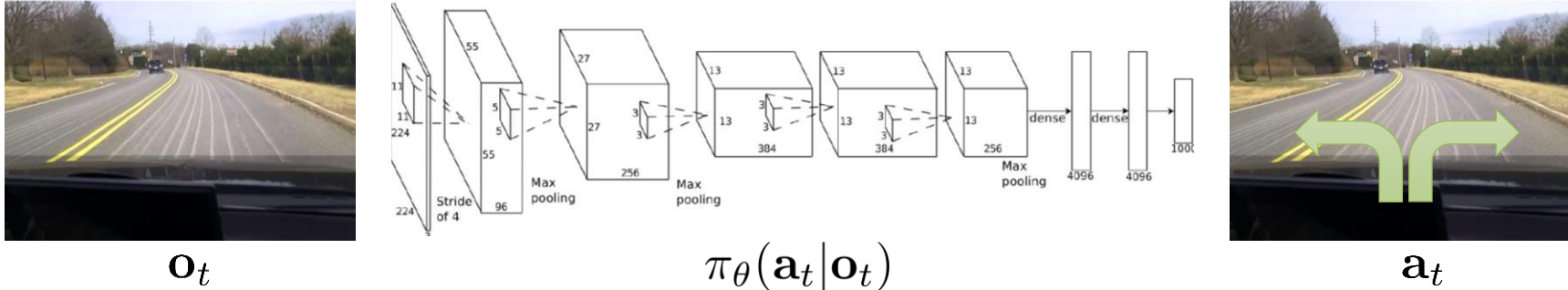
simply formalizes the notion of “trial and error”!

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



Partial observability



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

Markov property is not actually used!

Can use policy gradient in partially observed MDPs without modification

Making policy gradient actually work

A better estimator

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

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

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

“reward to go”

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

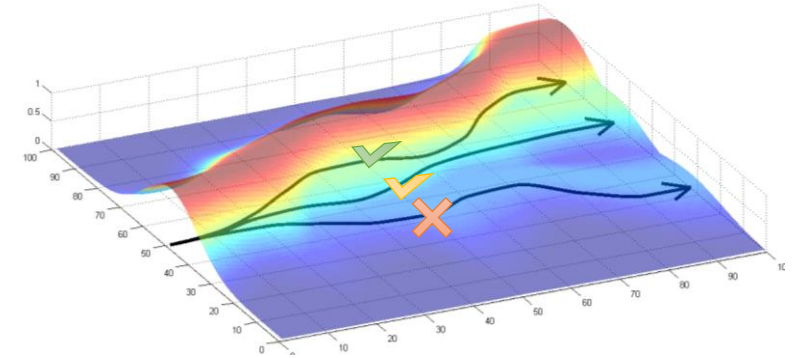
Baselines

a convenient identity

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

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

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



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

subtracting a baseline is *unbiased* in expectation!

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

Policy gradient is on-policy

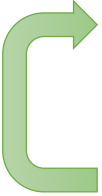
$$\theta^* = \arg \max_{\theta} J(\theta)$$

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

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

- 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)

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

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

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

importance sampling

$$\begin{aligned} E_{x \sim p(x)}[f(x)] &= \int p(x) f(x) dx \\ &= \int \frac{q(x)}{q(x)} p(x) f(x) dx \\ &= \int q(x) \frac{p(x)}{q(x)} f(x) dx \\ &= E_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right] \end{aligned}$$

Deriving the policy gradient with 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

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

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

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

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

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

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

future actions don't affect current weight

if we ignore this, we get
a policy iteration algorithm
(more on this in a later lecture)

A first-order approximation for IS (preview)

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

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

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

$(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \sim \pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t)$

off-policy policy gradient: $\nabla_{\theta'} J(\theta') \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \frac{\pi_{\theta'}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})}{\pi_{\theta}(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \hat{Q}_{i,t}$


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

This simplification is very commonly used in practice!

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)
```

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

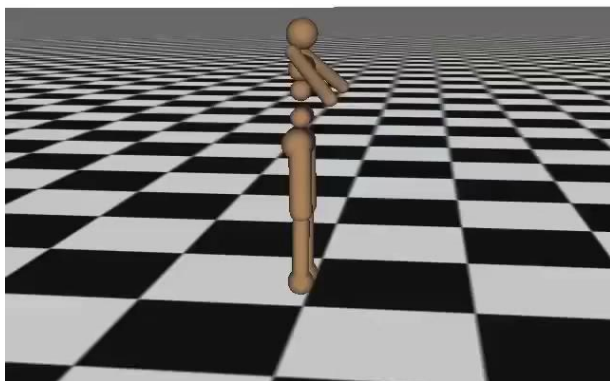
q_values

Policy gradient in practice

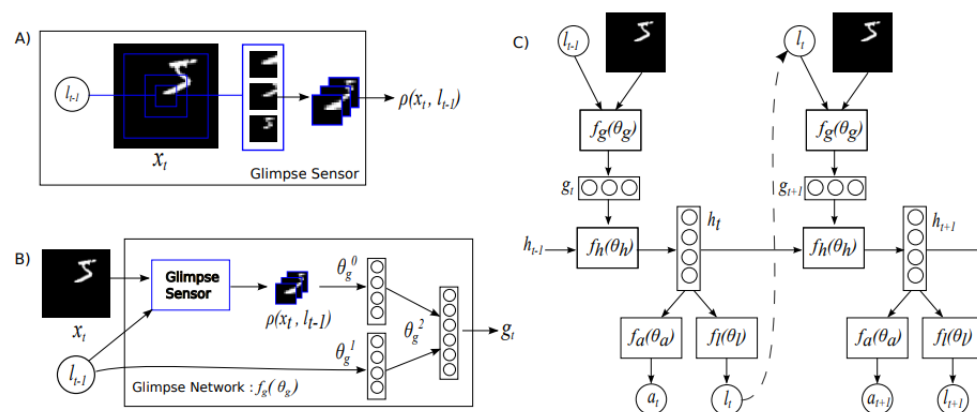
- Remember that the gradient has high variance
 - This isn't the same as supervised learning!
 - Gradients will be really noisy!
- Consider using much larger batches
- Tweaking learning rates is very hard
 - Adaptive step size rules like ADAM can be OK-ish
 - We'll learn about policy gradient-specific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

What are policy gradients used for?

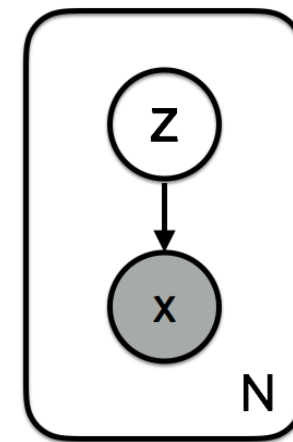
Iteration 0



learning for control



“hard” attention: network selects where to look
(Mnih et al. Recurrent Models of Visual Attention)




Discrete latent variable models
(we'll learn about these later!)

Long story short: policy gradient (REINFORCE) can be used in any setting where we have to **differentiate** through a stochastic but non-differentiable operation

Review

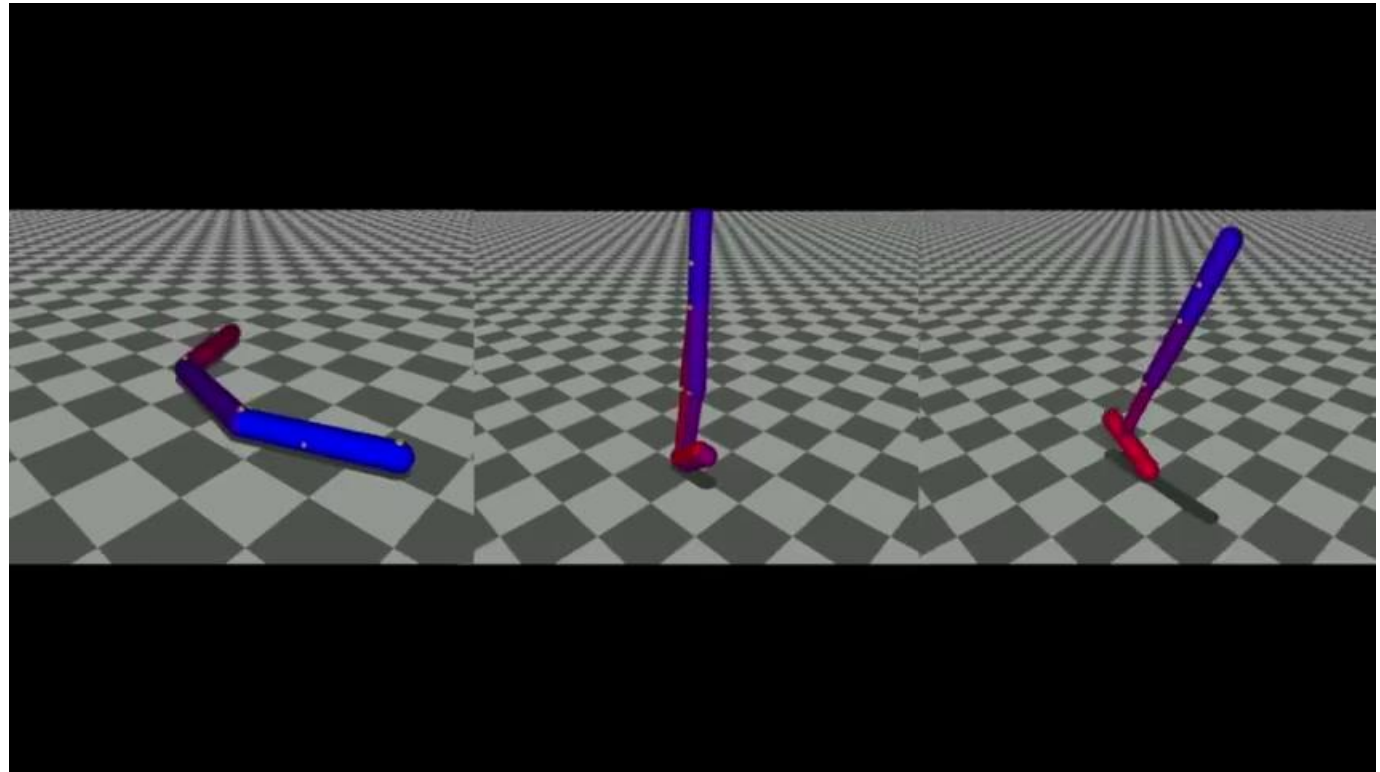
- Evaluating the RL objective
 - Generate samples
- Evaluating the policy gradient
 - Log-gradient trick
 - Generate samples
- Policy gradient is on-policy
- Can derive off-policy variant
 - Use importance sampling
 - Exponential scaling in T
 - Can ignore state portion (approximation)
- Can implement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

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

Example: trust region policy optimization

- Natural gradient with automatic step adjustment
- Discrete and continuous actions



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

UC Berkeley · CSW182 | [Deep Learning]

Designing, Visualizing and Understanding Deep Neural Networks (2021)

CSW182 (2021) · 课程资料包 @ShowMeAI



视频
中英双语字幕



课件
一键打包下载



笔记
官方笔记翻译



代码
作业项目解析



视频 · B 站 [扫码或点击链接]
<https://www.bilibili.com/video/BV1Ff4y1n7ar>



课件 & 代码 · 博客 [扫码或点击链接]
<http://blog.showmeai.tech/berkeley-csw182>

Berkeley
Q-Learning
计算机视觉

循环神经网络
风格迁移
机器学习基础

可视化
模仿学习
生成模型

梯度策略
元学习
卷积网络

Awesome AI Courses Notes Cheatsheets 是 [ShowMeAI](#) 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets · 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094



微信公众号

资料下载方式 2：扫码点击**底部菜单栏**
称为 **AI 内容创作者**？回复 [添砖加瓦]