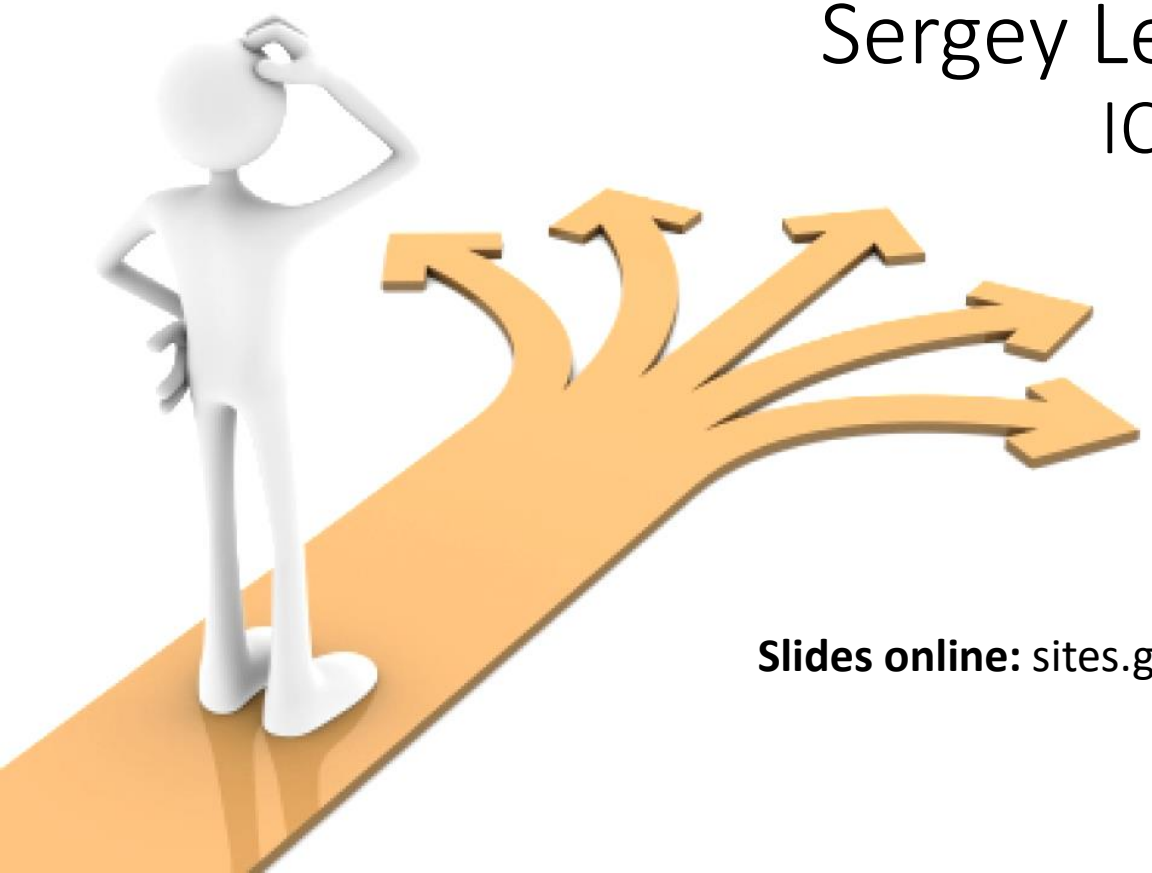
Deep Reinforcement Learning, Decision Making, and Control

Sergey Levine and Chelsea Finn

ICML 2017 Tutorial

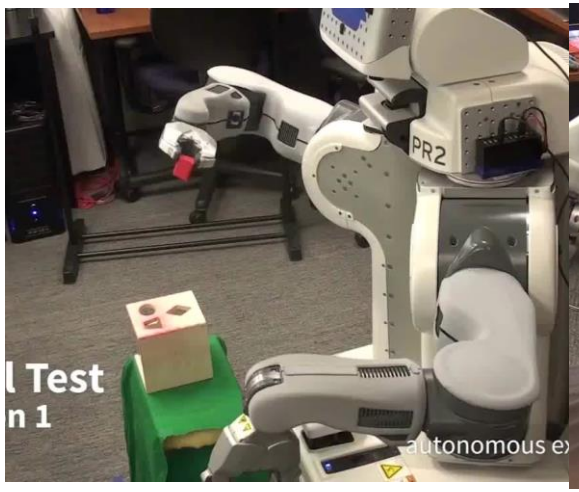


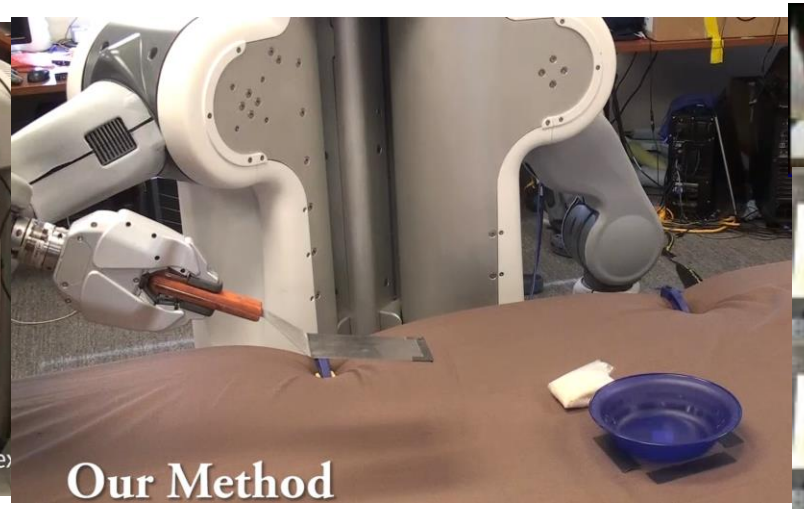
Slides online: sites.google.com/view/icml17deeprl

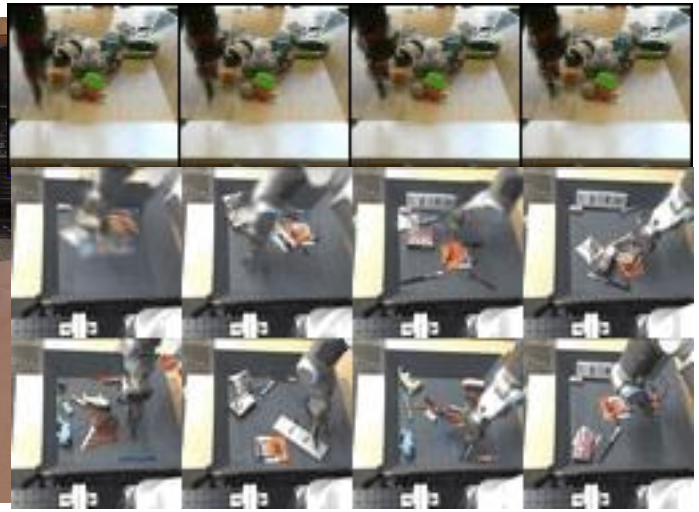
Introductions

autonomously learning complex sensorimotor skills

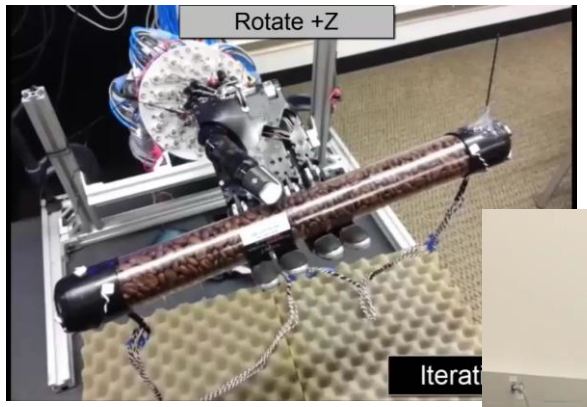
Chelsea Finn
PhD student

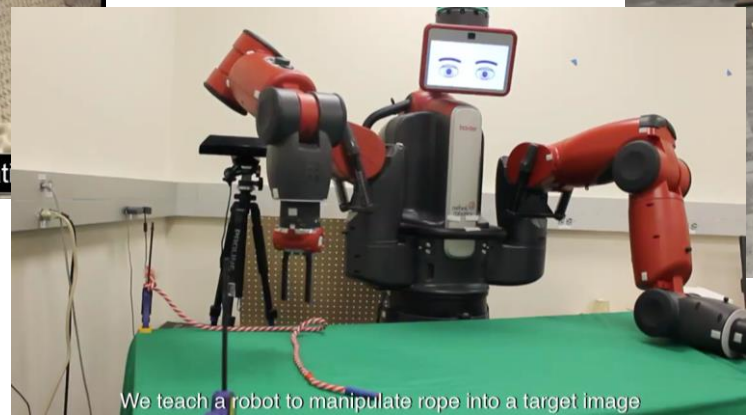






Sergey Levine
assistant professor





Deep Learning for Sequential Decision Making

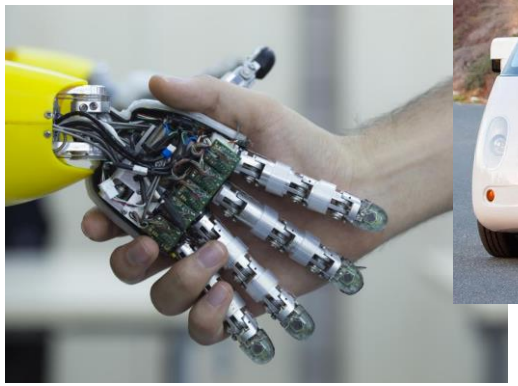
When do you <u>not</u> need sequential decision making?

When your system is making single isolated decision, e.g. classification, regression

When that decision does not affect future decisions

Common Applications

autonomous driving



robotics

language & dialogue
(structured prediction)

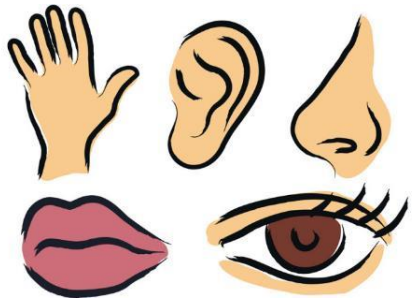
business operations

finance

+ a key aspect of intelligence

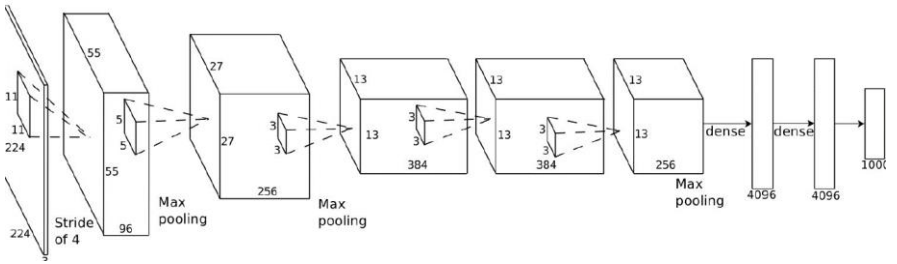
Why deep reinforcement learning?

- Interpret rich sensory inputs
  - Deep learning is really good at this



- Choose complex actions
  - Reinforcement learning is really good at this

- Use deep networks to represent mapping from sensors to actions
- Subsumes estimation, control, etc.
- Requires scaling to huge functions



Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

fundamental topics

-- BREAK –

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

more advanced topics

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

fundamental topics

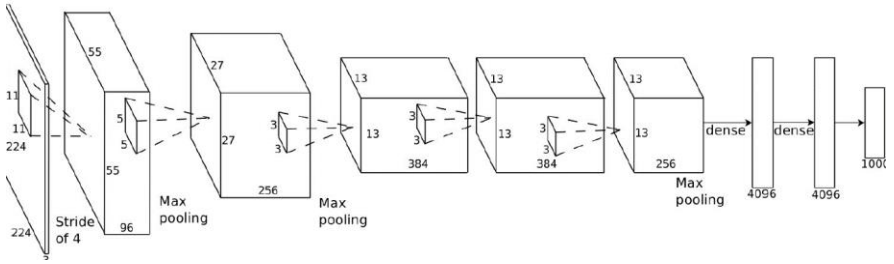
-- BREAK –

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

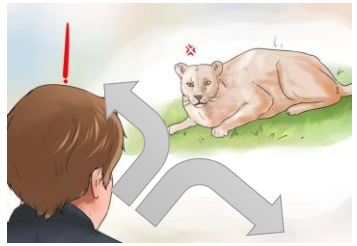
more advanced topics

Terminology & notation

$\mathbf{o}$



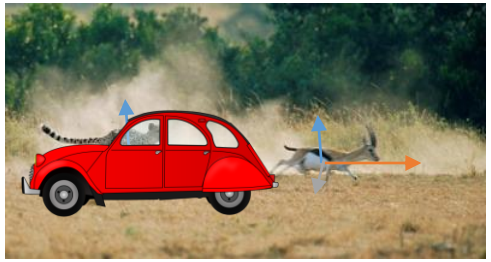
$\pi_\theta(\mathbf{a}|\mathbf{o})$



$\mathbf{a}$

$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

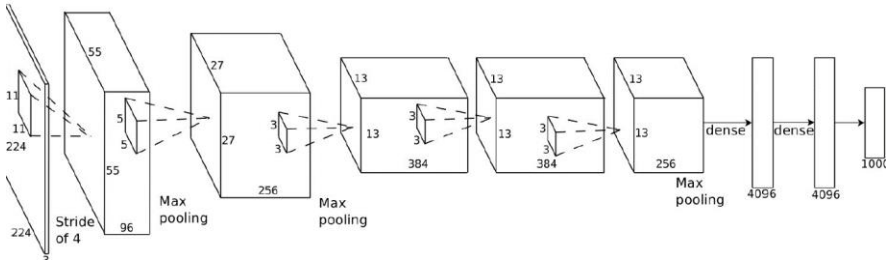
$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)

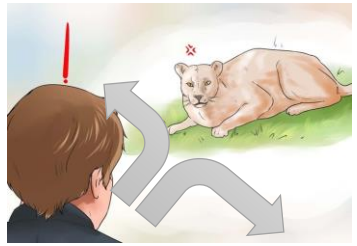


$\mathbf{o}_t$ – observation

$\mathbf{s}_t$ – state

Terminology & notation

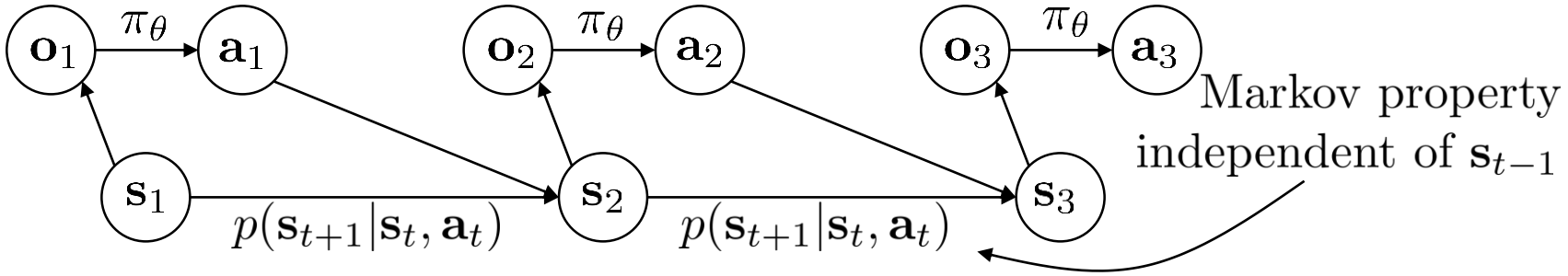




$\mathbf{o}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ $\mathbf{a}_t$

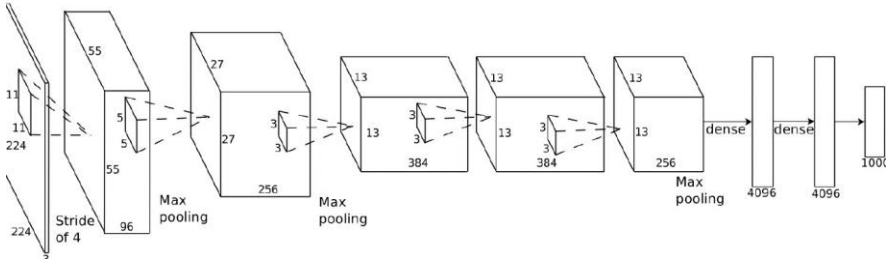
$\mathbf{s}_t$ – state
$\mathbf{o}_t$ – observation
$\mathbf{a}_t$ – action

$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$ – policy
$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ – policy (fully observed)



Imitation Learning

$\mathbf{o}_t$

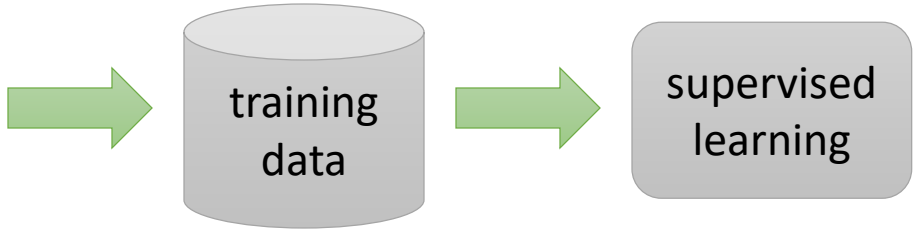


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

$\mathbf{a}_t$

$\mathbf{o}_t$

$\mathbf{a}_t$

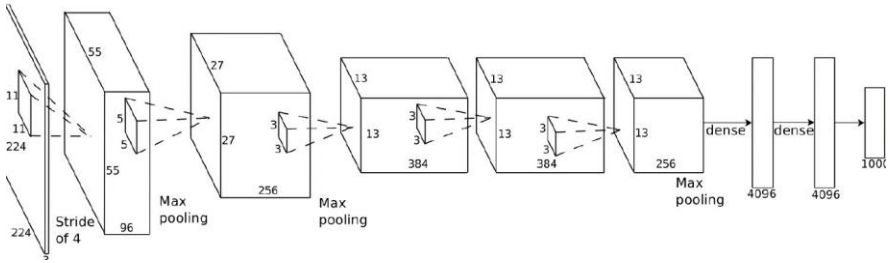


$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

Images: Bojarski et al. ‘16, NVIDIA

Reward functions

$\mathbf{o}_t$



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

$\mathbf{a}_t$

which action is better or worse?

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

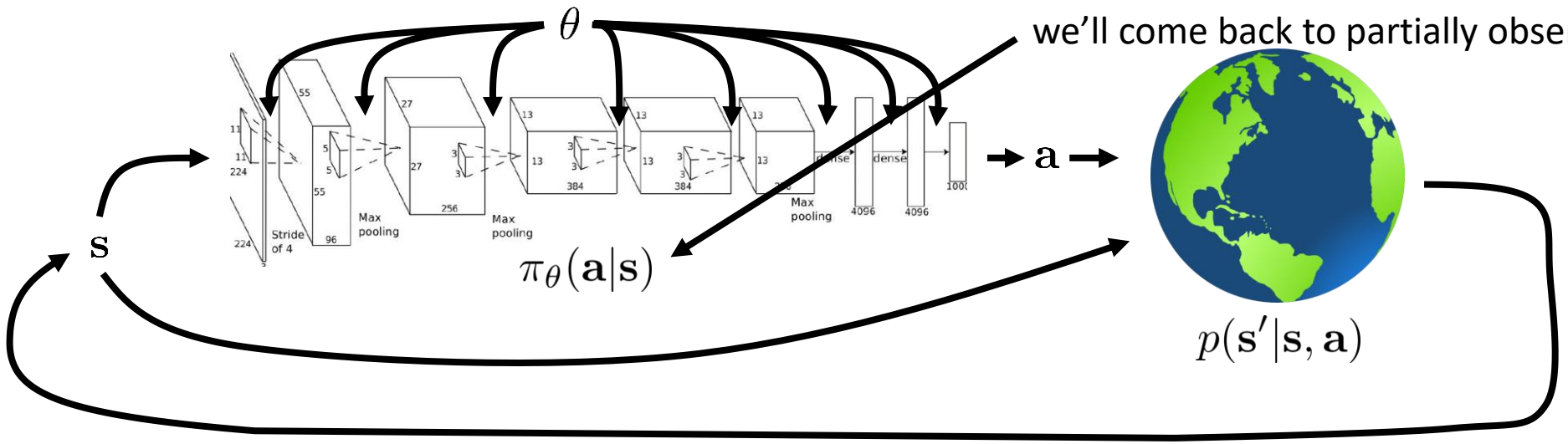
tells us which states and actions are better

$\mathbf{s}$, $\mathbf{a}$, $r(\mathbf{s}, \mathbf{a})$, and $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ define Markov decision process

high reward

low reward

The goal of reinforcement learning



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

$p_\theta(\mathbf{s}_t, \mathbf{a}_t)$ state-action marginal

$p_\theta(\mathbf{s}, \mathbf{a})$ stationary distribution

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

infinite horizon case

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

finite horizon case

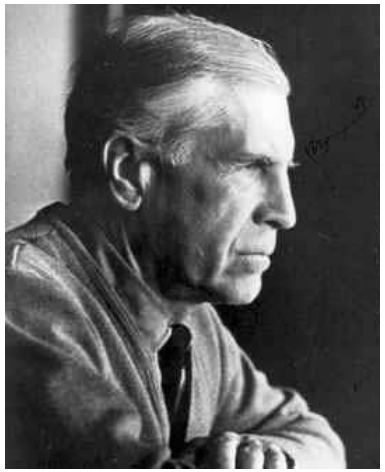
Aside: notation

$\mathbf{s}_t$ – state
$\mathbf{a}_t$ – action
$r(\mathbf{s}, \mathbf{a})$ – reward function

$\mathbf{x}_t$ – state
$\mathbf{u}_t$ – action управление
$c(\mathbf{x}, \mathbf{u})$ – cost function

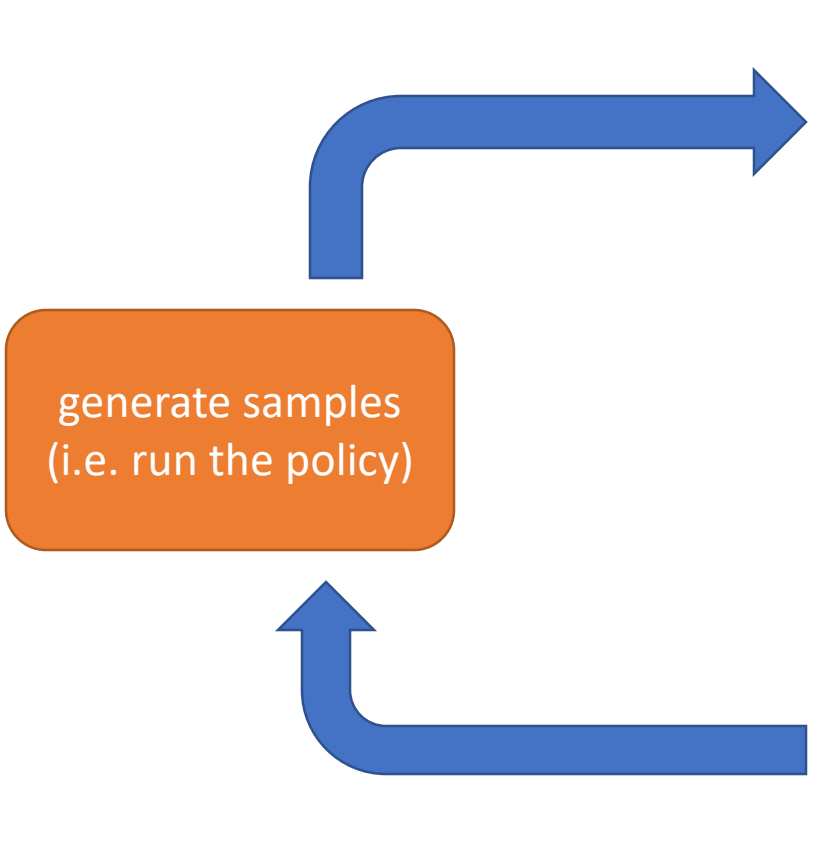
$$r(\mathbf{s}, \mathbf{a}) = -c(\mathbf{s}, \mathbf{a})$$

Richard Bellman



Lev Pontryagin

The anatomy of a reinforcement learning algorithm



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)

fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)

estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

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

$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

fundamental topics

-- BREAK –

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

more advanced topics

Direct policy differentiation

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

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

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

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

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

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

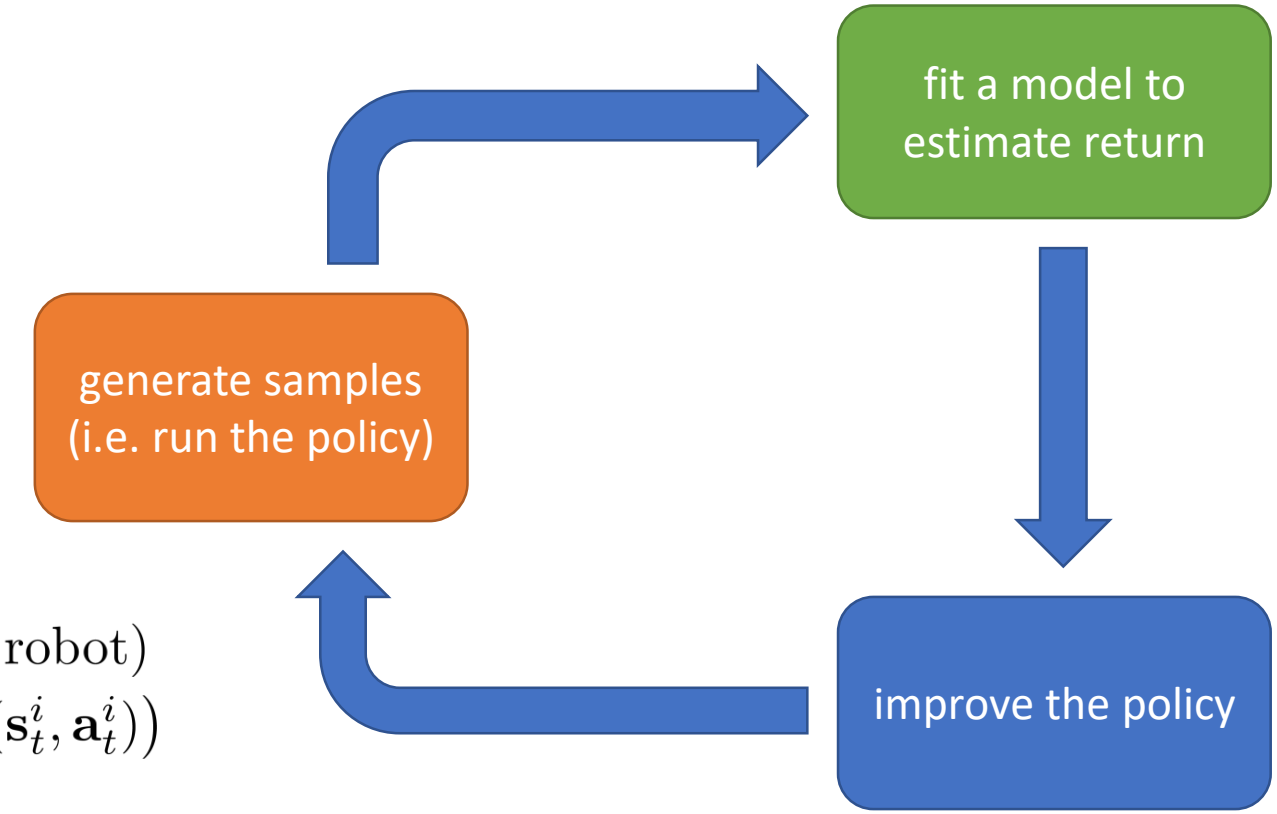
$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T}\nabla_\theta\log\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T}r(\mathbf{s}_t,\mathbf{a}_t)\right)\right]$$

Evaluating the policy gradient

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

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

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



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

Evaluating the policy gradient

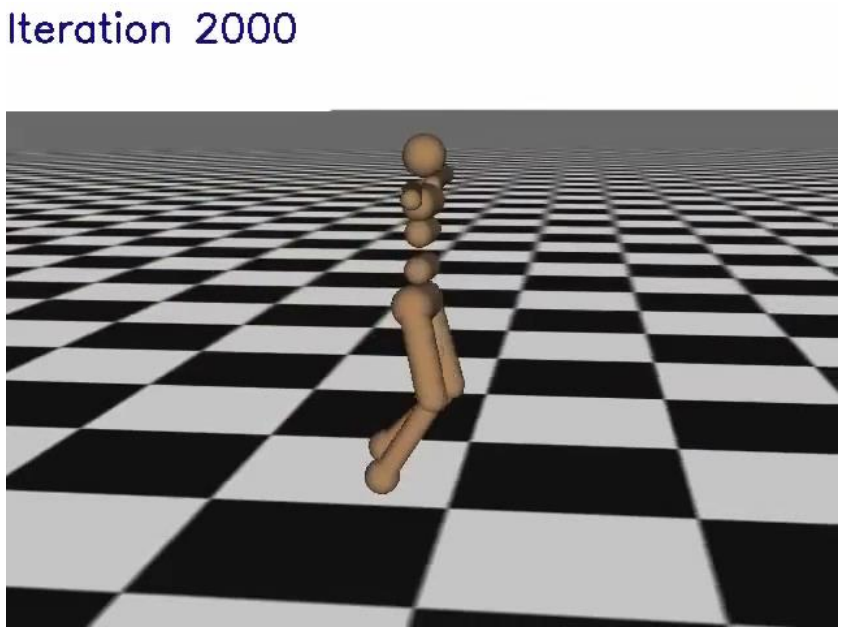
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

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

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

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

how? just backpropagate $-\frac{1}{2}\Sigma^{-1}(f(\mathbf{s}_t) - \mathbf{a}_t)$



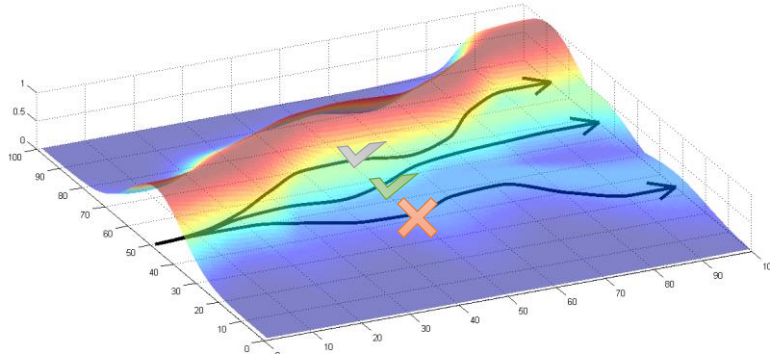
REINFORCE algorithm:

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

What did we just do?

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

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



good stuff is made more likely

bad stuff is made less likely

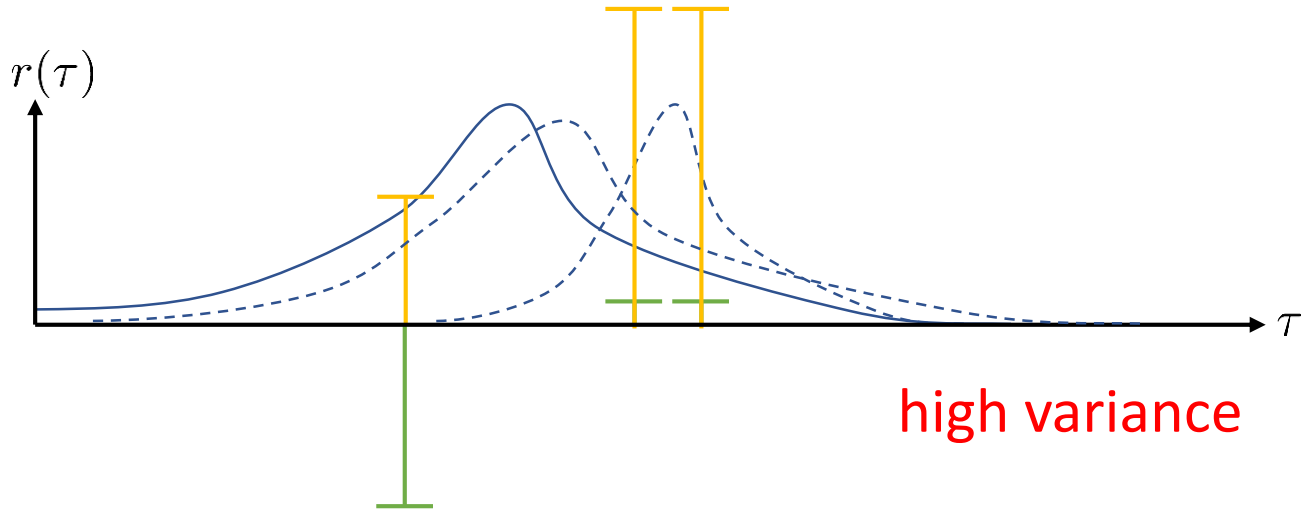
simply formalizes the notion of “trial and error”!

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

What is wrong with the policy gradient?

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



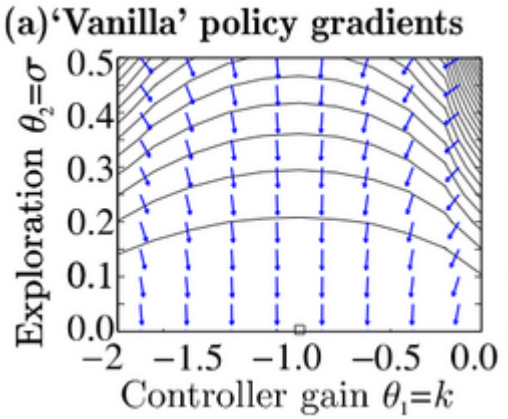
high variance

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

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

slow convergence

hard to choose learning rate



(image from Peters & Schaal 2008)

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 $t < t'$

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

"reward to go"

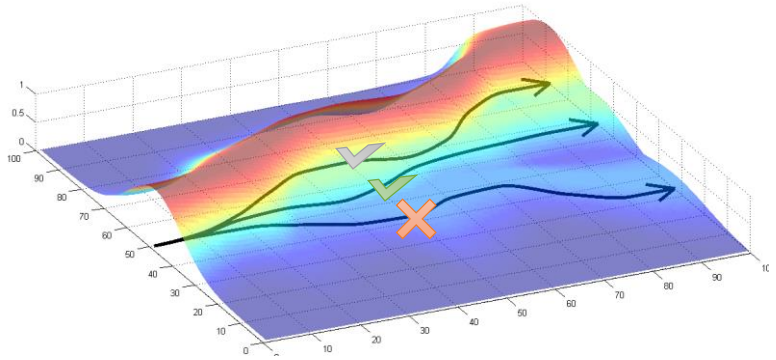
$$\hat{Q}_{i,t}$$

Baselines

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

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

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!

Control variates

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

what if b is *not* a constant?

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

often can be computed analytically

example: quadratic baseline, Gaussian policy

see also: Gu et al. 2016 (Q-Prop)

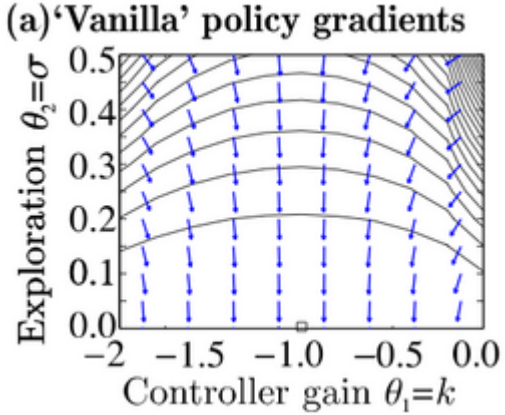
Covariant/natural policy gradient

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \qquad \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

some parameters change probabilities a lot more than others!

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

controls how far we go



can we *rescale* the gradient so this doesn't happen?

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

parameterization-independent divergence measure

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

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

Fisher-information matrix

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

can estimate with samples

Covariant/natural policy gradient

$$D_{\mathrm{KL}}(\pi_{\theta'} \| \theta_\pi) \approx (\theta' - \theta)^T \mathbf{F} (\theta' - \theta) \qquad \mathbf{F} = E_{\pi_\theta}[\log \pi_\theta(\mathbf{a}|\mathbf{s}) \log \pi_\theta(\mathbf{a}|\mathbf{s})^T]$$

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

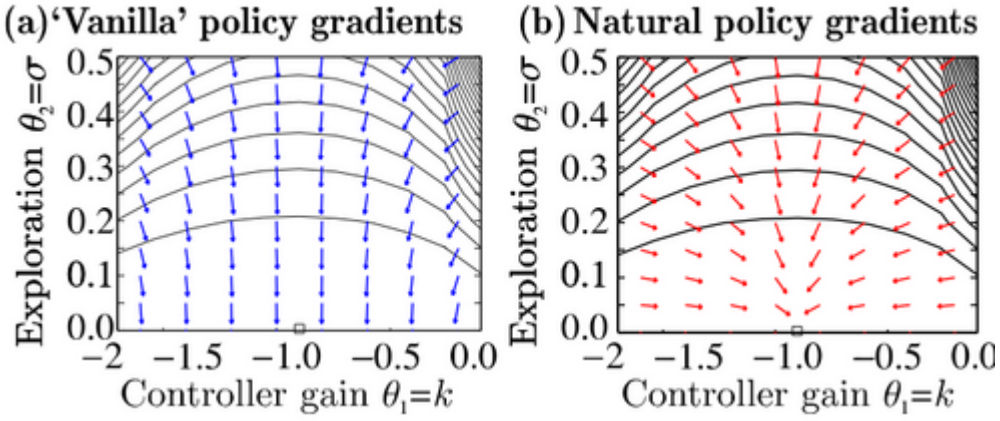
$$\theta \leftarrow \theta + \alpha \mathbf{F}^{-1} \nabla_\theta J(\theta)$$

natural gradient: pick α

trust region policy optimization: pick ϵ

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

conjugate gradient works well for ths



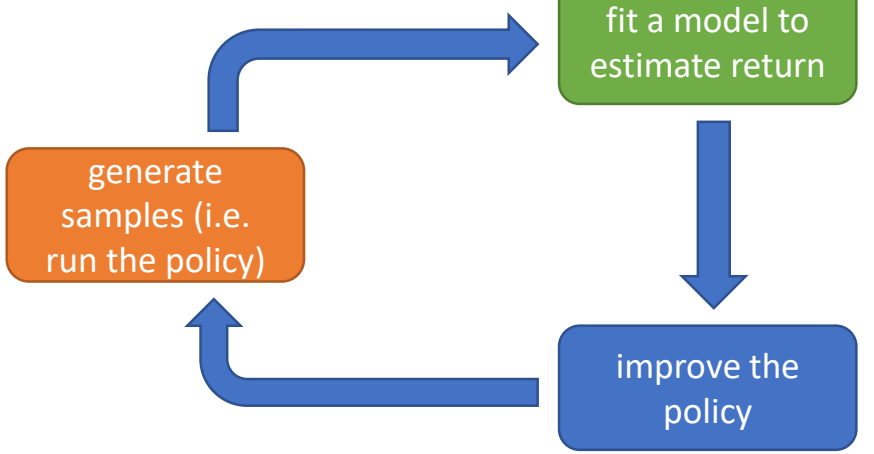
(figure from Peters & Schaal 2008)

see Schulman, L., Moritz, Jordan, Abbeel (2015) Trust region policy optimization

Review

- Policy gradient
  - Directly differentiate expected reward
  - Formalizes trial-and-error learning
- Reducing variance
  - Use causality
  - Use a baseline
- Improve convergence rate
  - Natural/covariant gradient
  - Automatic step adjustment

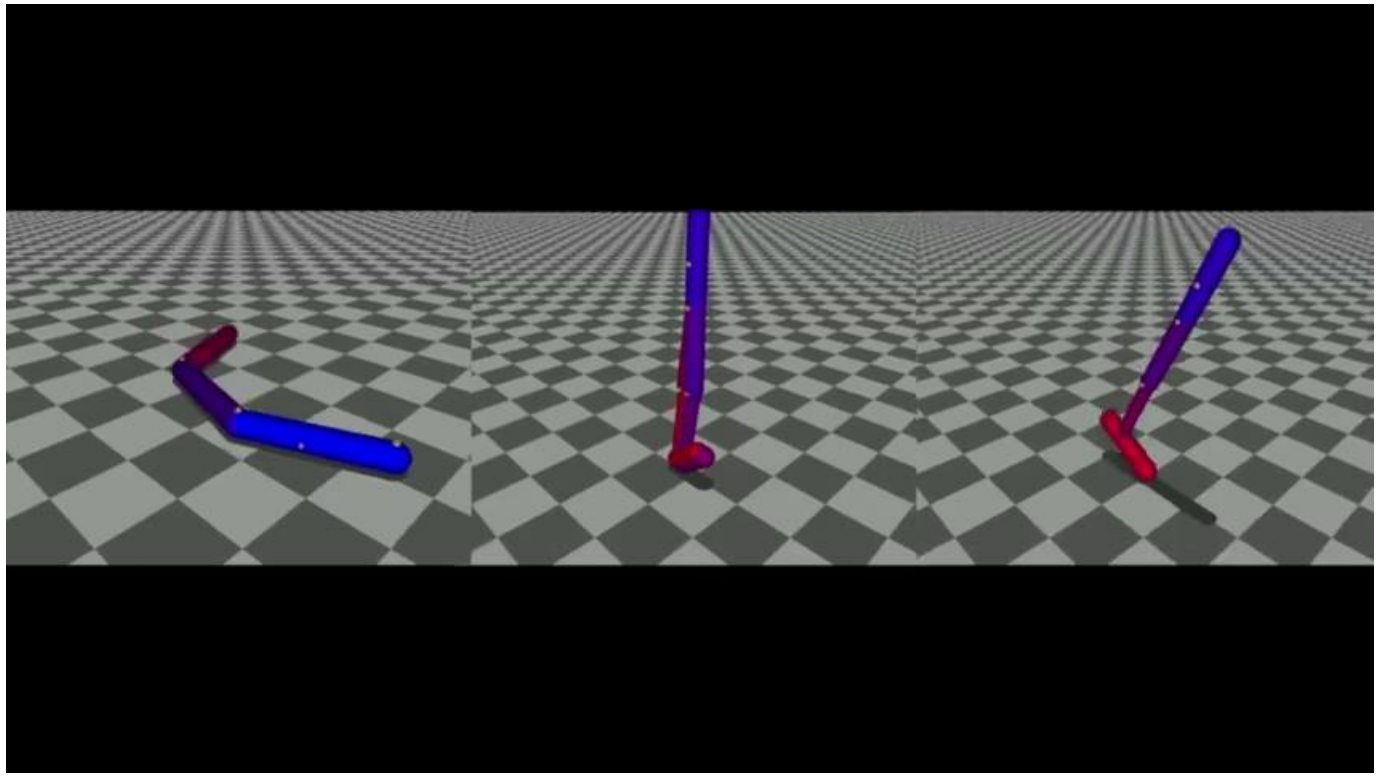
$$\hat{Q}^{\pi}(\mathbf{x}_t, \mathbf{u}_t) = \sum_{t'=t}^{T} r(\mathbf{x}_{t'}, \mathbf{u}_{t'})$$



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

Policy gradient example: TRPO

- Natural gradient
- Automatic step adjustment
- Discrete and continuous actions
- Easy to use
- Code available (see Duan et al. ‘16)



Schulman, L., Moritz, Jordan, Abbeel. ‘15

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
  - L. & 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

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

fundamental topics

-- BREAK –

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

more advanced topics

Improving the policy evaluation

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

"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

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

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

Definitions: state & state-action value functions

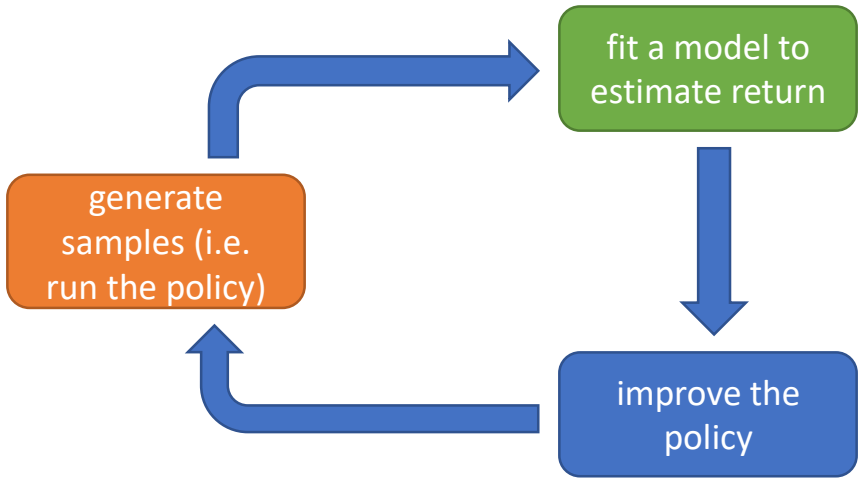
$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

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

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

fit A^π!



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

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

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

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

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

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

fit *what* to *what*?

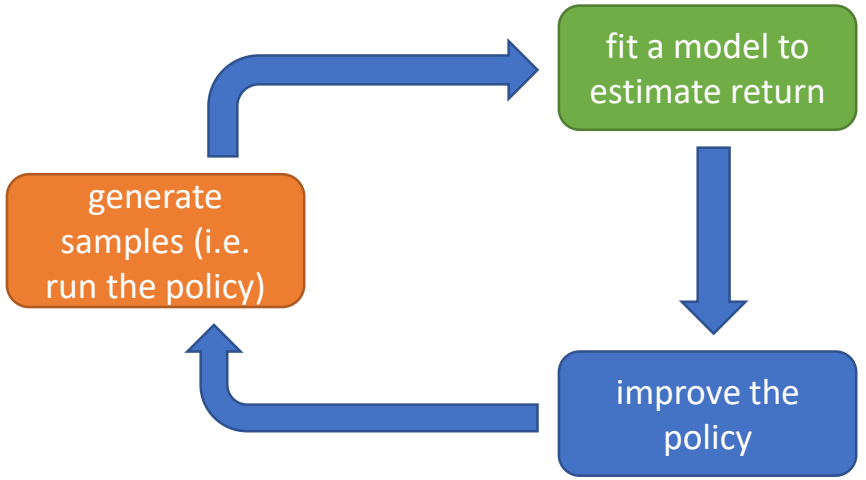
Q^π, V^π, A^π?

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

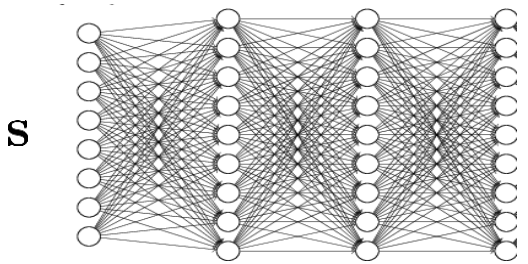
$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$$

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

fit A^π!



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



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

parameters ϕ

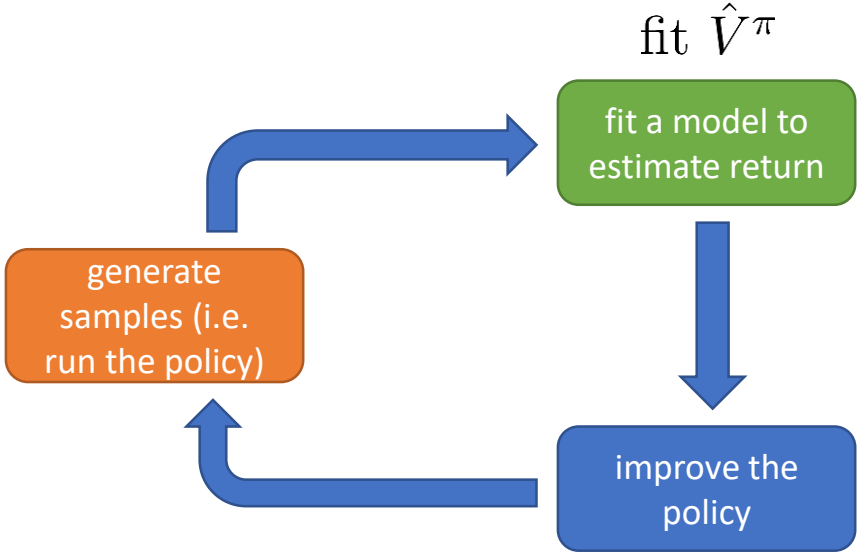
$$V^\pi(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t\right]$$

fit to samples of this expectation

Value function fitting

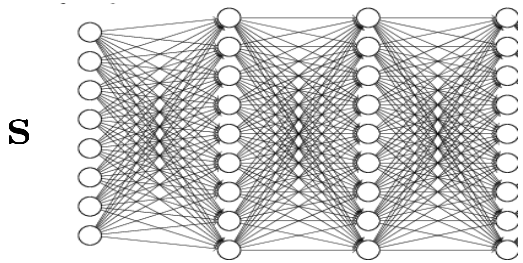
batch actor-critic algorithm:

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



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

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



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

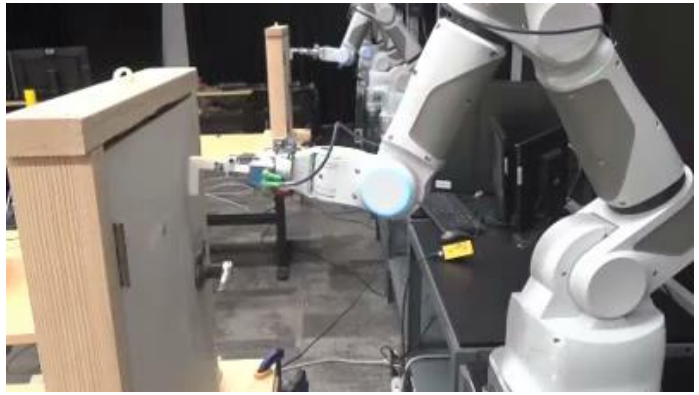
Aside: discount factors

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

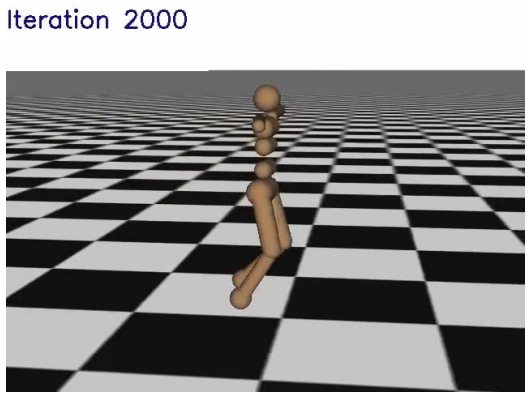
$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}^{\pi}_{\phi}(\mathbf{s}_i) - y_i \right\|^2$$

what if T (episode length) is ∞?

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



episodic tasks



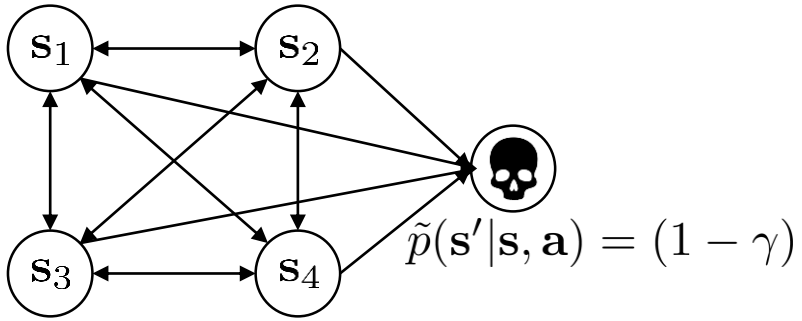
continuous/cyclical tasks

simple trick: better to get rewards sooner than later

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

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

γ changes the MDP:



Online actor-critic

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 $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
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 $\hat{V}^\pi_\phi$ using target $r + \gamma \hat{V}^\pi_\phi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}^\pi_\phi(\mathbf{s}') - \hat{V}^\pi_\phi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Details and last-minute gift ideas

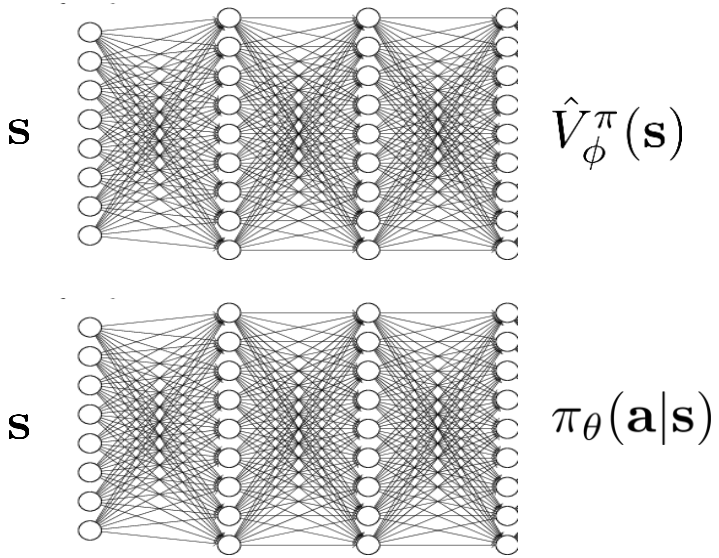
online actor-critic algorithm:

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

works best with a batch (e.g., parallel workers)

we can design better estimators
(for both batch and online)
See Schulman, Moritz, L. Jordan, Abbeel '16:
Generalized advantage estimation

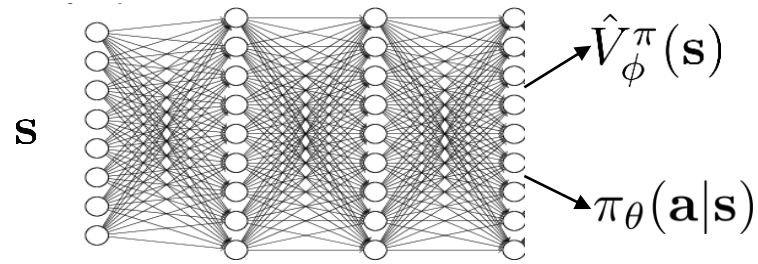
two network design



+ simple & stable

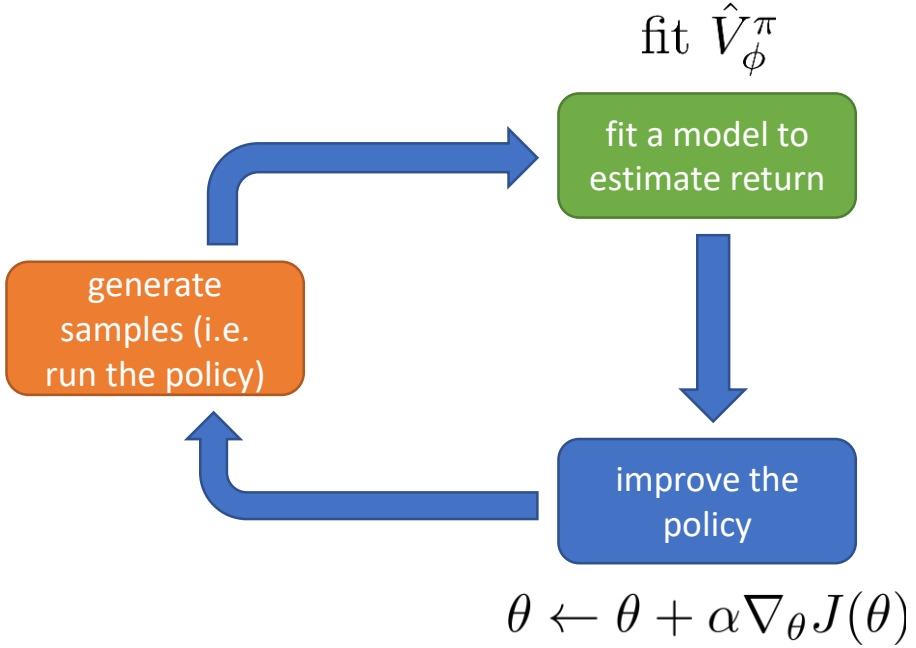
- no shared features between actor & critic

shared network design

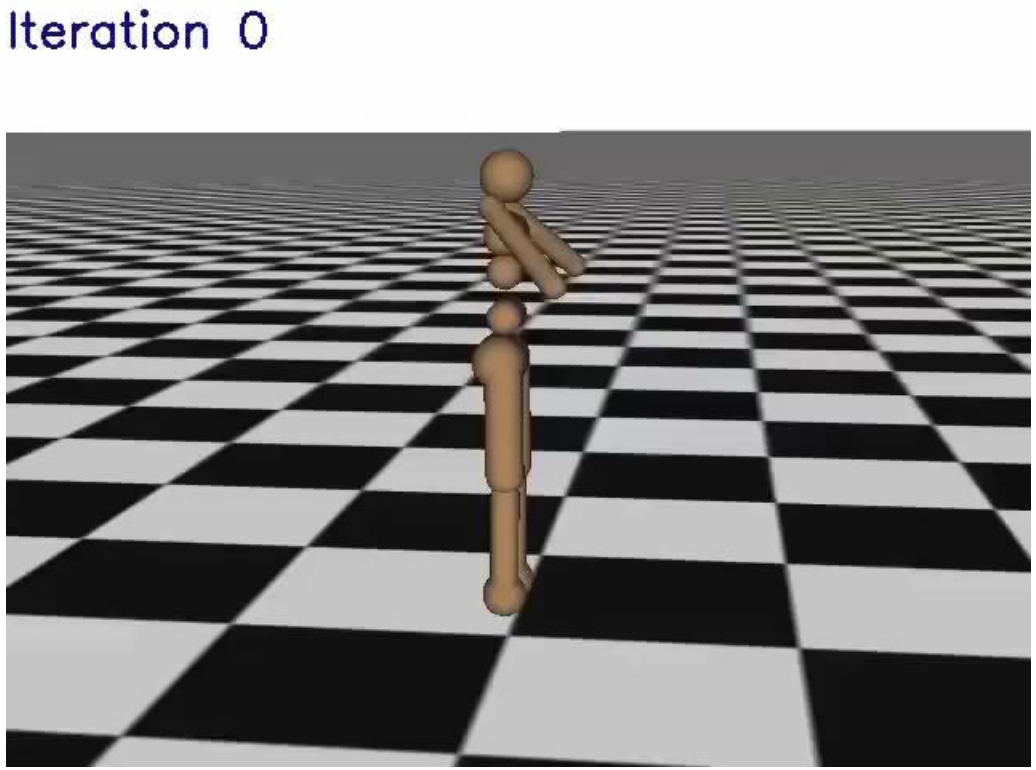


Review

- Actor-critic algorithms:
  - Two neural networks (or one net with two heads)
  - Actor: predict action from state
  - Critic: evaluate state (can also evaluate state and action)
- Connection to policy gradient:
  - Just policy gradient with function approximator for advantage
  - Multiple ways to estimate advantage
- Can be batch or online



Actor-critic examples



- TRPO-GAE (Schulman, Moritz, L., Jordan, Abbeel ‘16)
- Batch-mode actor-critic
- Blends Monte Carlo and function approximator estimators

- A3C (Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, Kavukcuoglu)
- Online actor-critic, parallelized batch
- 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

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

fundamental topics

-- BREAK –

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

more advanced topics

Can we omit policy gradient completely?

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

$\arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t)$: best action from $\mathbf{s}_t$, if we then follow π

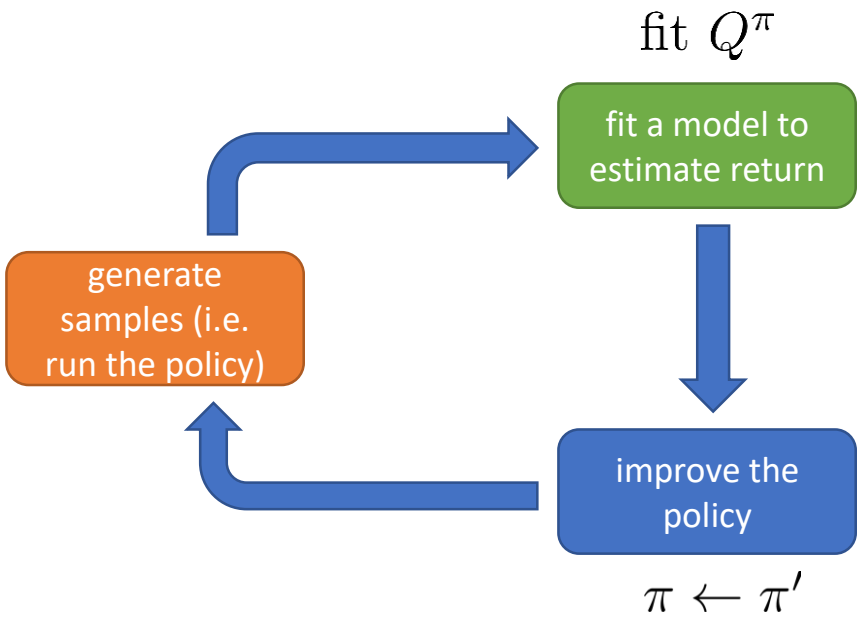
at *least* as good as any $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$

regardless of what $\pi(\mathbf{a}_t|\mathbf{s}_t)$ is!

forget policies, let's just do this!

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

as good as π
(probably better)



Dynamic programming

recall: $Q^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}[V^\pi(\mathbf{s}')]$

enough to compute this, if we know the dynamics

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

what is π? $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$

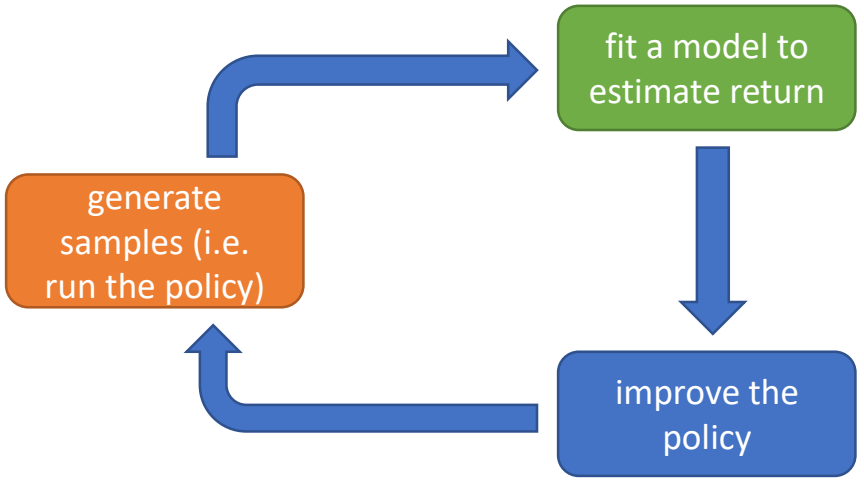
value iteration algorithm:

1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')]$
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

problem: need to know $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

problem: need to represent $V(\mathbf{s})$

$Q^\pi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}[V^\pi(\mathbf{s}')]$



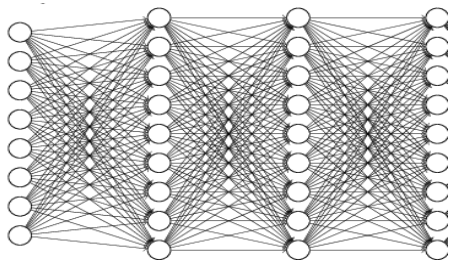
$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$

Bellman error minimization

how do we represent $V(\mathbf{s})$?

big table, one entry for each discrete $\mathbf{s}$

neural net function $V : \mathcal{S} \rightarrow \mathbb{R}$

$\mathbf{s}$  $V(\mathbf{s})$

parameters ϕ

$$\mathbf{s} = 0 : V(\mathbf{s}) = 0.2$$
$$\mathbf{s} = 1 : V(\mathbf{s}) = 0.3$$
$$\mathbf{s} = 2 : V(\mathbf{s}) = 0.5$$

curse of dimensionality

$$|\mathcal{S}| = (255^3)^{200 \times 200}$$

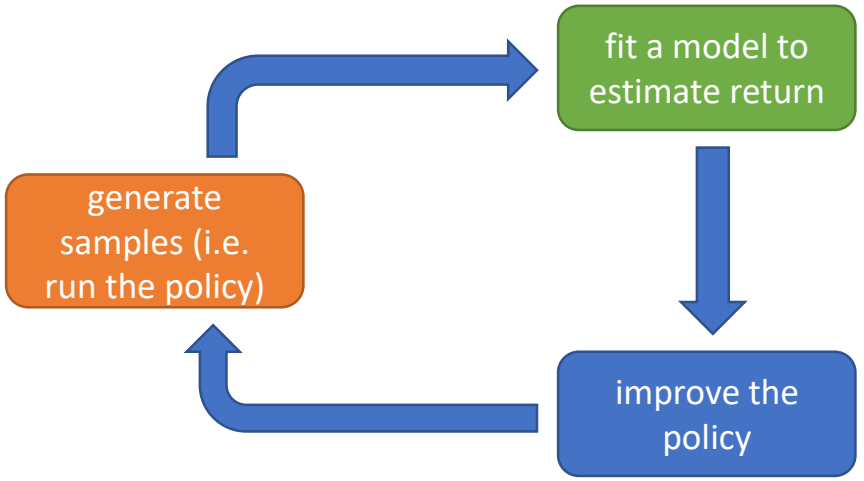
(more than atoms in the universe)

$$Q^\pi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}[V^\pi(\mathbf{s}')]$$

$$\mathcal{L}(\phi) = \frac{1}{2} \left\| V_\phi(\mathbf{s}) - \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a}) \right\|^2$$

fitted value iteration algorithm:

1. set $\mathbf{y}_i \leftarrow \max_{\mathbf{a}_i} (r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)])$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \left\| V_\phi(\mathbf{s}_i) - \mathbf{y}_i \right\|^2$



$$V^\pi(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q^\pi(\mathbf{s}, \mathbf{a})$$

What if we don't know the transition dynamics?

fitted value iteration algorithm:

1. set $\mathbf{y}_i \leftarrow \max_{\mathbf{a}_i}(r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)])$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|V_\phi(\mathbf{s}_i) - \mathbf{y}_i\|^2$

need to know outcomes for different actions!

What if we learn Q instead? $Q^\pi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}[V^\pi(\mathbf{s}')]$

fitted Q iteration algorithm:

1. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma E[V_\phi(\mathbf{s}'_i)]$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2}\sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

approxiate $E[V(\mathbf{s}'_i)] \approx \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$

doesn't require simulation of actions!

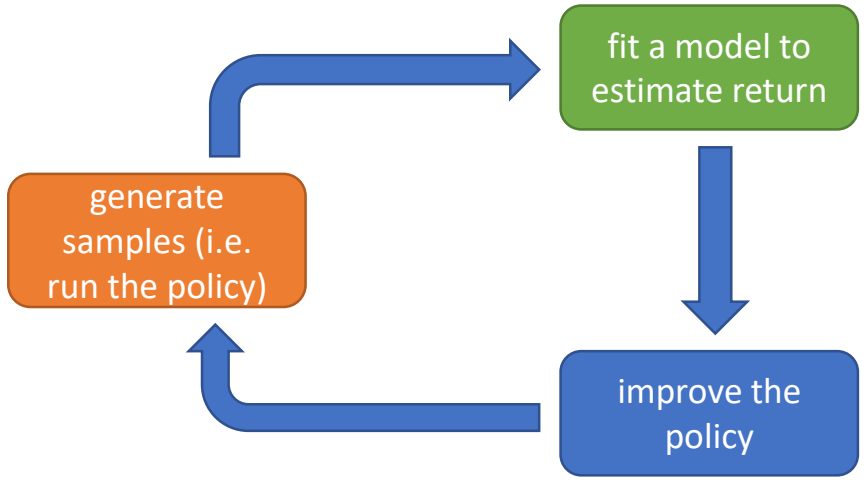
+ works even for off-policy samples (unlike actor-critic)

+ only one network, no high-variance policy gradient

- no convergence guarantees for non-linear function approximation (more on this later)

Online Q-learning

$$Q_\phi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')$$



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

fitted Q iteration algorithm:

1. set $\mathbf{y}_i \leftarrow r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$
2. set $\phi \leftarrow \arg\min_\phi \frac{1}{2} \sum_i \|Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{y}_i\|^2$

off policy, so many choices here!

online Q iteration algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

$$\pi'(\mathbf{a}_t|\mathbf{a}_t) = \begin{cases} 1 - \epsilon \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \\ \epsilon/(|\mathcal{A}| - 1) \text{ otherwise} \end{cases}$$

“epsilon-greedy”

Q-learning in practice

online Q iteration algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$
2. $\phi \leftarrow \phi - \alpha \frac{dQ_\phi}{d\phi}(\mathbf{s}_i, \mathbf{a}_i)(Q_\phi(\mathbf{s}_i, \mathbf{a}_i) - [r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}'_i, \mathbf{a}'_i)])$

no gradient through target value

- sequential states are strongly correlated
- target value is always changing
- Q-learning is *not* gradient descent!

a better Q-learning algorithm:

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)$, add it to $\mathcal{R}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j\}$ from $\mathcal{R}$ uniformly
3. compute $y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mathbf{a}'_j)$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps, or Polyak average $\phi' \leftarrow \tau\phi' + (1-\tau)\phi$

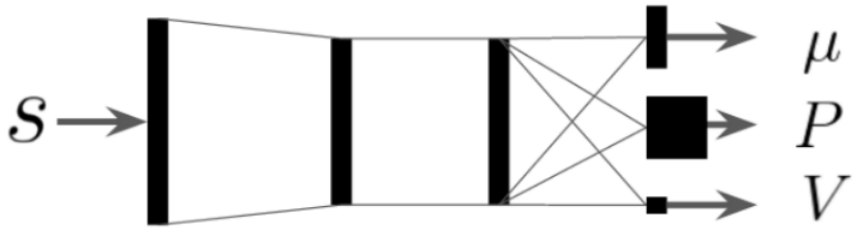
- still not guaranteed to converge!

Mnih et al. ‘13

Q-learning with continuous actions

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

Option 1: use function class that is easy to optimize



NAF Architecture.

$$Q(\boldsymbol{x}, \boldsymbol{u}|\theta^Q) = A(\boldsymbol{x}, \boldsymbol{u}|\theta^A) + V(\boldsymbol{x}|\theta^V)$$

$$A(\boldsymbol{x}, \boldsymbol{u}|\theta^A) = -\frac{1}{2}(\boldsymbol{u} - \boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))^T \boldsymbol{P}(\boldsymbol{x}|\theta^P)(\boldsymbol{u} - \boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu))$$

Gu, Lillicrap, Sutskever, L., ICML 2016

Option 2: learn an approximate maximizer

DDPG (Lillicrap et al., ICLR 2016)

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

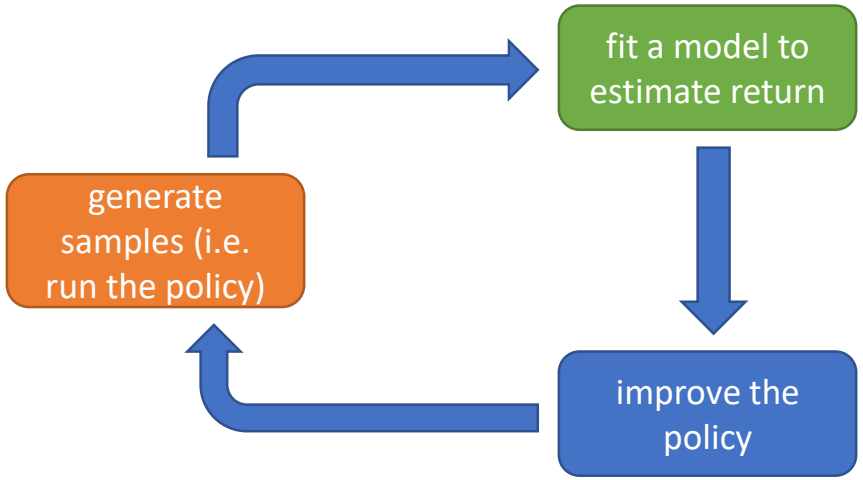
$y_j = r_j + \gamma \max_{\mathbf{a}'_j} Q_{\phi'}(\mathbf{s}'_j, \mu_\theta(\mathbf{s}'_j))$

“deterministic” actor-critic
(really approximate Q-learning)

Review

- Value-based methods
  - Don’t learn a policy explicitly
  - Just learn value or Q-function
- If we have Q-function, we have a policy
- Batch mode or online
- Not guaranteed to converge
- Replay buffers, target networks
- Approximations for continuous actions

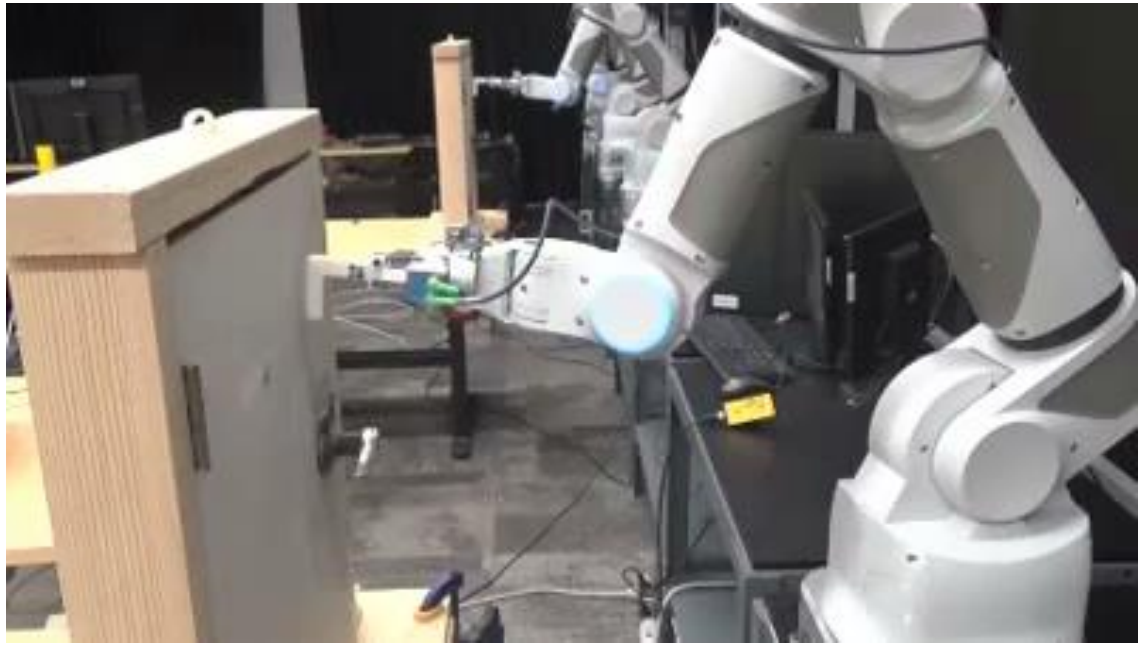
$$Q_\phi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')$$



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

Q-learning examples

- DQN, Mnih et al. ‘13
- Q-learning with convolutional networks
- Uses replay buffer and target network



- Gu*, Holly*, Lillicrap, L. ‘17
- Uses NAF for real-world robotic control
- Learns door opening using 2 robots in 2.5 hours

Q-learning suggested readings

- Classic papers
  - Watkins. (1989). Learning from delayed rewards: introduces Q-learning
  - Riedmiller. (2005). Neural fitted Q-iteration: batch-mode Q-learning with neural networks
- Deep reinforcement learning Q-learning papers
  - Lange, Riedmiller. (2010). Deep auto-encoder neural networks in reinforcement learning: early image-based Q-learning method using autoencoders to construct embeddings
  - Mnih et al. (2013). Human-level control through deep reinforcement learning: Q-learning with convolutional networks for playing Atari.
  - Van Hasselt, Guez, Silver. (2015). Deep reinforcement learning with double Q-learning: a very effective trick to improve performance of deep Q-learning.
  - Lillicrap et al. (2016). Continuous control with deep reinforcement learning: continuous Q-learning with actor network for approximate maximization.
  - Gu, Lillicrap, Stuskever, L. (2016). Continuous deep Q-learning with model-based acceleration: continuous Q-learning with action-quadratic value functions.
  - Wang, Schaul, Hessel, van Hasselt, Lanctot, de Freitas (2016). Dueling network architectures for deep reinforcement learning: separates value and advantage estimation in Q-function.

gradient-free methods
(e.g. NES, CMA, etc.)

10x

fully online methods
(e.g. A3C)

10x

policy gradient methods
(e.g. TRPO)

10x

replay buffer value estimation methods
(Q-learning, DDPG, NAF, etc.)

10x

model-based deep RL
(e.g. guided policy search)

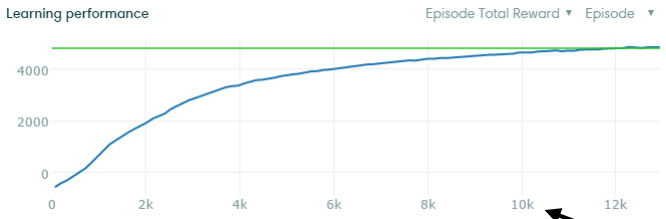
10x

model-based “shallow” RL
(e.g. PILCO)

Evolution Strategies as a Scalable Alternative to Reinforcement Learning

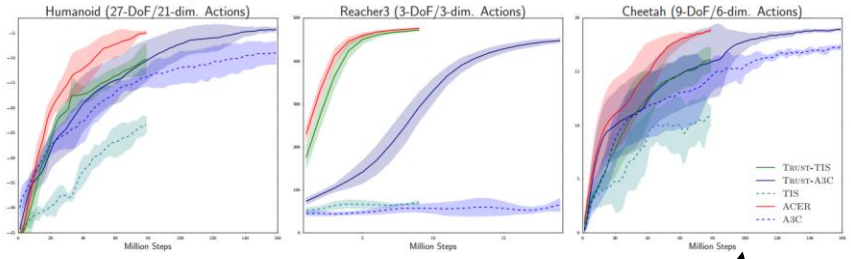
Tim Salimans [1] Jonathan Ho [1] Xi Chen [1] Ilya Sutskever [1]

half-cheetah (slightly different version)



TRPO+GAE (Schulman et al. ‘16)

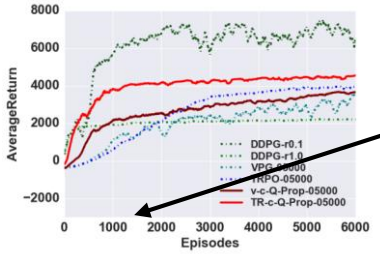
10,000,000 steps
(10,000 episodes)
(~ 1.5 days real time)



Wang et al. ‘17

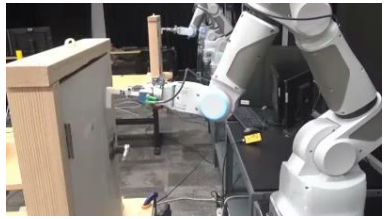
100,000,000 steps
(100,000 episodes)
(~ 15 days real time)

half-cheetah

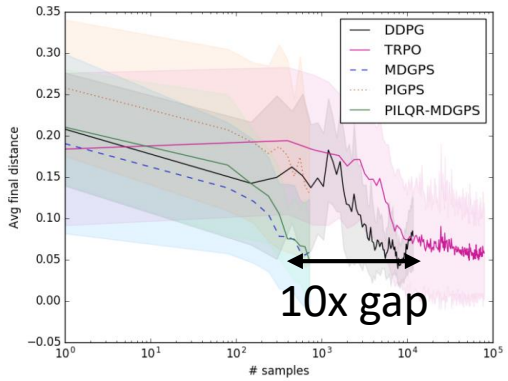


Gu et al. ‘16

1,000,000 steps
(1,000 episodes)
(~ 3 hours real time)



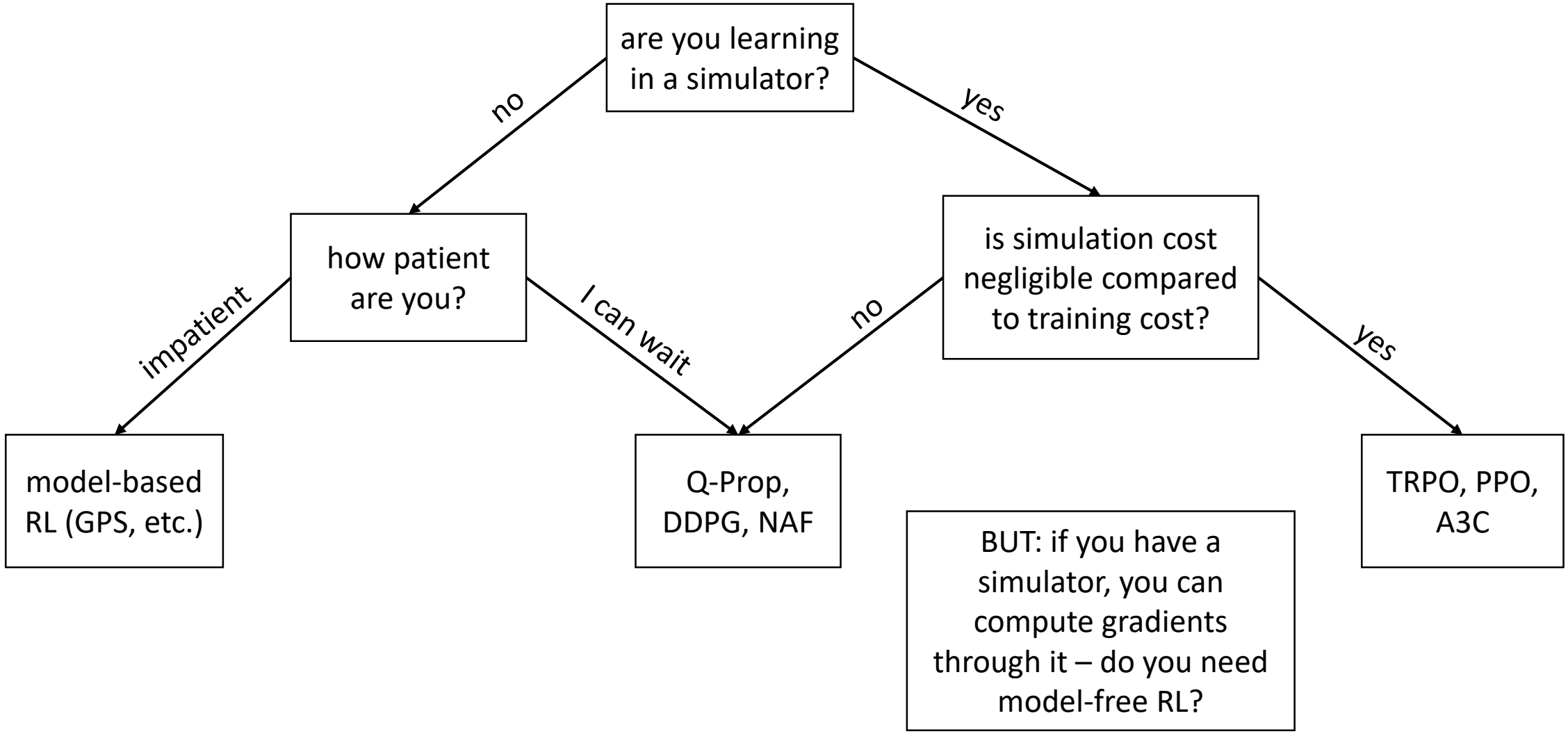
	cart-pole	cart-double-pole	unicycle
state space	$\mathbb{R}^4$	$\mathbb{R}^6$	$\mathbb{R}^{12}$
# trials	≤ 10	20–30	≈ 20
experience	$\approx 20\,\mathrm{s}$	$\approx 60\,\mathrm{s}$–$90\,\mathrm{s}$	$\approx 20\,\mathrm{s}$–$30\,\mathrm{s}$
parameter space	$\mathbb{R}^{305}$	$\mathbb{R}^{1816}$	$\mathbb{R}^{28}$



Chebotar et al. ’17 (note log scale)

about 20 minutes of experience on a real robot

Which RL algorithm to use?



Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

fundamental topics

-- BREAK –

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

more advanced topics

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

fundamental topics

-- BREAK –

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

more advanced topics

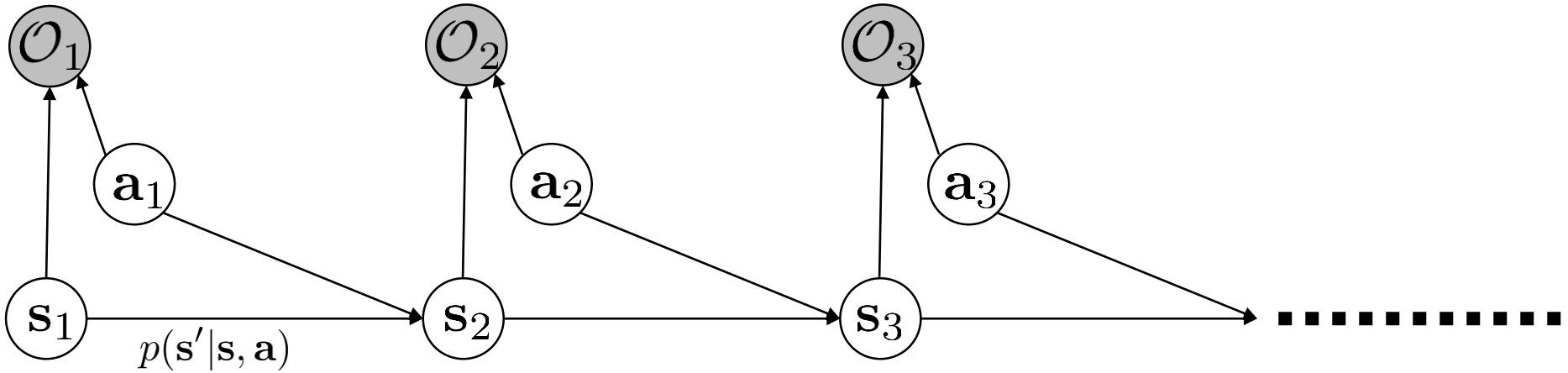
Is RL inference in a graphical model?

$$p(\underbrace{\mathbf{s}_{1:T}, \mathbf{a}_{1:T}}_{\tau}) = ??$$

$p(\tau|\mathcal{O}_{1:T})$ $\qquad$ $p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) \propto \exp(r(\mathbf{s}_t, \mathbf{a}_t))$

$$p(\tau|\mathcal{O}_{1:T}) = \frac{p(\tau, \mathcal{O}_{1:T})}{p(\mathcal{O}_{1:T})}$$

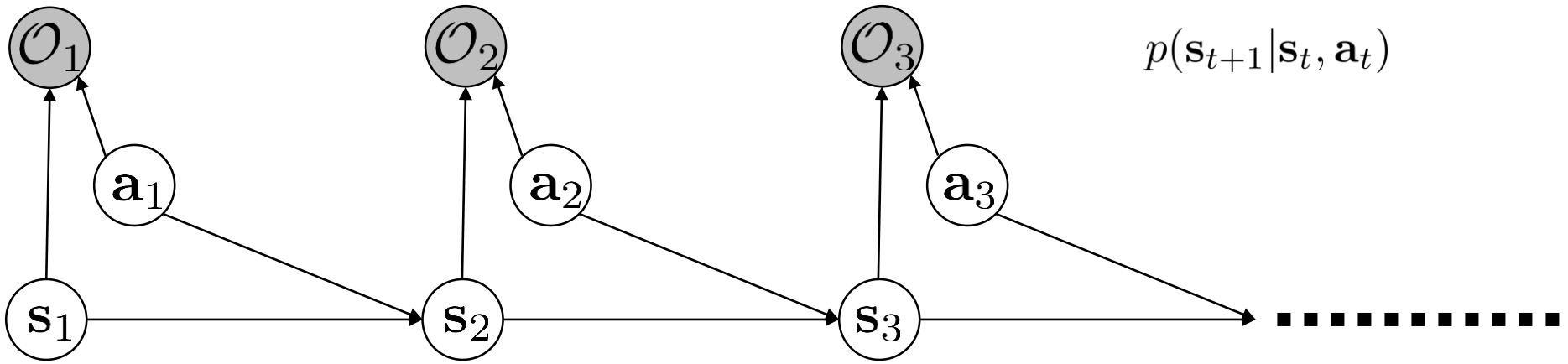
$$\propto p(\tau) \prod_t \exp(r(\mathbf{s}_t, \mathbf{a}_t)) = p(\tau) \exp\left(\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right)$$



Inference = planning

$$p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) \propto \exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$



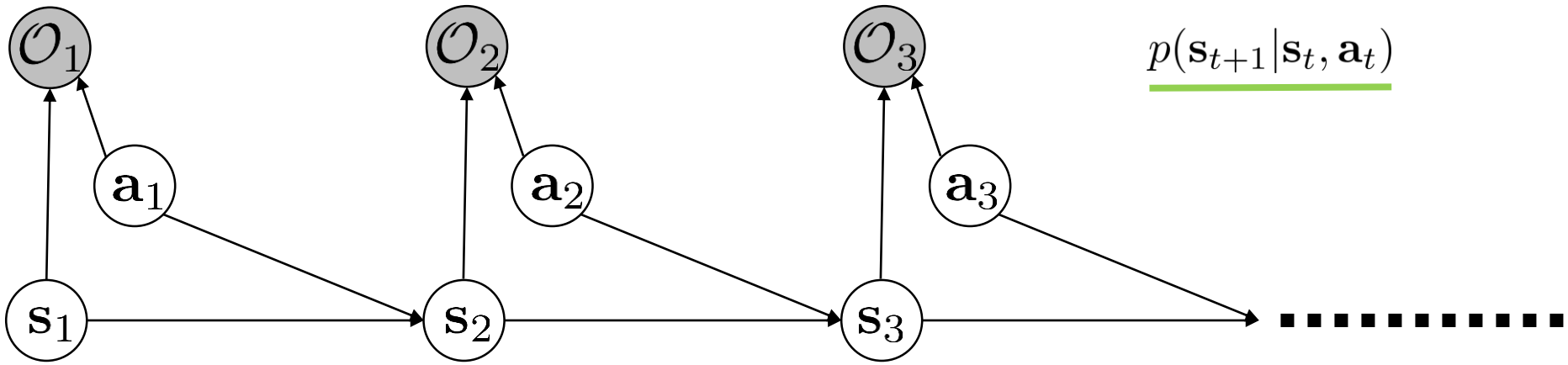
how to do inference?

1. compute backward messages $\beta_t(\mathbf{s}_t, \mathbf{a}_t) = p(\mathcal{O}_{t:T}|\mathbf{s}_t, \mathbf{a}_t)$
2. compute policy $p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{1:T})$
3. compute forward messages $\alpha_t(\mathbf{s}_t) = p(\mathbf{s}_t|\mathcal{O}_{1:t-1})$

Backward messages

$$p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) \propto \exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$



$$\begin{aligned}\beta_t(\mathbf{s}_t, \mathbf{a}_t) &= p(\mathcal{O}_{t:T}|\mathbf{s}_t, \mathbf{a}_t) \\ &= \int p(\mathcal{O}_{t:T}, \mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) d\mathbf{s}_{t+1} \\ &= \int p(\mathcal{O}_{t+1:T}|\mathbf{s}_{t+1}) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) d\mathbf{s}_{t+1}\end{aligned}$$

$$p(\mathcal{O}_{t+1:T}|\mathbf{s}_{t+1}) = \int \underbrace{p(\mathcal{O}_{t+1:T}|\mathbf{s}_{t+1}, \mathbf{a}_{t+1})}_{\beta_t(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})} p(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}) d\mathbf{a}_{t+1}$$

which actions are likely *a priori*
(assume uniform for now)

for $t = T-1$ to 1:

$$\beta_t(\mathbf{s}_t, \mathbf{a}_t) = p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}[\beta_{t+1}(\mathbf{s}_{t+1})]$$

$$\beta_t(\mathbf{s}_t) = E_{\mathbf{a}_t \sim p(\mathbf{a}_t|\mathbf{s}_t)}[\beta_t(\mathbf{s}_t, \mathbf{a}_t)]$$

A closer look at the backward pass

value iteration algorithm:

1. set $Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma E[V(\mathbf{s}')]$
2. set $V(\mathbf{s}) \leftarrow \max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

for $t = T - 1$ to 1:

$$\beta_t(\mathbf{s}_t, \mathbf{a}_t) = p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)}[\beta_{t+1}(\mathbf{s}_{t+1})]$$

$$\beta_t(\mathbf{s}_t) = E_{\mathbf{a}_t \sim p(\mathbf{a}_t|\mathbf{s}_t)}[\beta_t(\mathbf{s}_t, \mathbf{a}_t)]$$

"optimistic" transition
(not a good idea!)

let $V_t(\mathbf{s}_t) = \log \beta_t(\mathbf{s}_t)$

let $Q_t(\mathbf{s}_t, \mathbf{a}_t) = \log \beta_t(\mathbf{s}_t, \mathbf{a}_t)$

$$V_t(\mathbf{s}_t) = \log \int \exp(Q_t(\mathbf{s}_t, \mathbf{a}_t)) d\mathbf{a}_t$$

$$Q_t(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \log E[\exp(V_{t+1}(\mathbf{s}_{t+1}))]$$

deterministic transition: $Q_t(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + V_{t+1}(\mathbf{s}_{t+1})$

a better stochastic model: $Q_t(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + E[V_{t+1}(\mathbf{s}_{t+1})]$

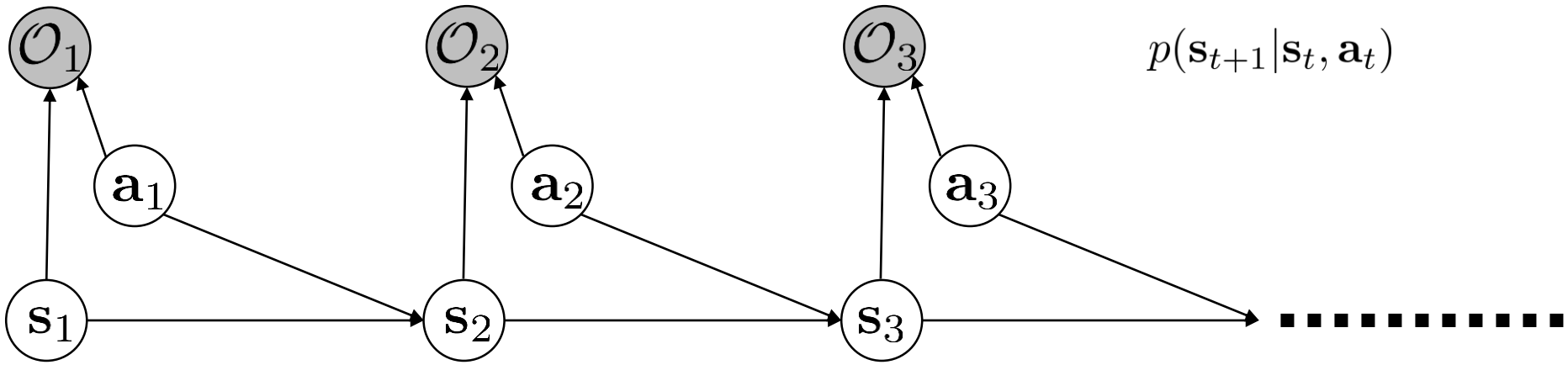
$V_t(\mathbf{s}_t) \rightarrow \max_{\mathbf{a}_t} Q_t(\mathbf{s}_t, \mathbf{a}_t)$ as $Q_t(\mathbf{s}_t, \mathbf{a}_t)$ gets bigger!

Ziebart et al. '10 "Modeling Interaction via the Principle of Maximum Causal Entropy"

Policy computation

$$p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) \propto \exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$



2. compute policy $p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{1:T})$

$$\beta_t(\mathbf{s}_t, \mathbf{a}_t) = p(\mathcal{O}_{t:T}|\mathbf{s}_t, \mathbf{a}_t)$$

$$\beta_t(\mathbf{s}_t) = p(\mathcal{O}_{t:T}|\mathbf{s}_t)$$

$$\begin{aligned} p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{1:T}) &= \pi(\mathbf{a}_t|\mathbf{s}_t) \\ &= p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{t:T}) \\ &= \frac{p(\mathbf{a}_t, \mathbf{s}_t|\mathcal{O}_{t:T})}{p(\mathbf{s}_t|\mathcal{O}_{t:T})} \\ &= \frac{p(\mathcal{O}_{t:T}|\mathbf{a}_t, \mathbf{s}_t)p(\mathbf{a}_t, \mathbf{s}_t)/\cancel{p(\mathcal{O}_{t:T})}}{p(\mathcal{O}_{t:T}|\mathbf{s}_t)p(\mathbf{s}_t)/\cancel{p(\mathcal{O}_{t:T})}} \\ &= \frac{p(\mathcal{O}_{t:T}|\mathbf{a}_t, \mathbf{s}_t)}{p(\mathcal{O}_{t:T}|\mathbf{s}_t)} \frac{p(\mathbf{a}_t, \mathbf{s}_t)}{p(\mathbf{s}_t)} = \frac{\beta_t(\mathbf{s}_t, \mathbf{a}_t)}{\beta_t(\mathbf{s}_t)} \cancel{p(\mathbf{a}_t|\mathbf{s}_t)} \end{aligned}$$

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \frac{\beta_t(\mathbf{s}_t, \mathbf{a}_t)}{\beta_t(\mathbf{s}_t)}$$

Policy computation with value functions

for $t = T - 1$ to 1:

$$Q_t(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + E[V_{t+1}(\mathbf{s}_{t+1})]$$

$$V_t(\mathbf{s}_t) = \log \int \exp(Q_t(\mathbf{s}_t, \mathbf{a}_t))\mathbf{a}_t$$

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \frac{\beta_t(\mathbf{s}_t, \mathbf{a}_t)}{\beta_t(\mathbf{s}_t)} \qquad V_t(\mathbf{s}_t) = \log \beta_t(\mathbf{s}_t)$$

$$Q_t(\mathbf{s}_t, \mathbf{a}_t) = \log \beta_t(\mathbf{s}_t, \mathbf{a}_t)$$

$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \exp(Q_t(\mathbf{s}_t, \mathbf{a}_t) - V_t(\mathbf{s}_t)) = \exp(A_t(\mathbf{s}_t, \mathbf{a}_t))$$

variants:

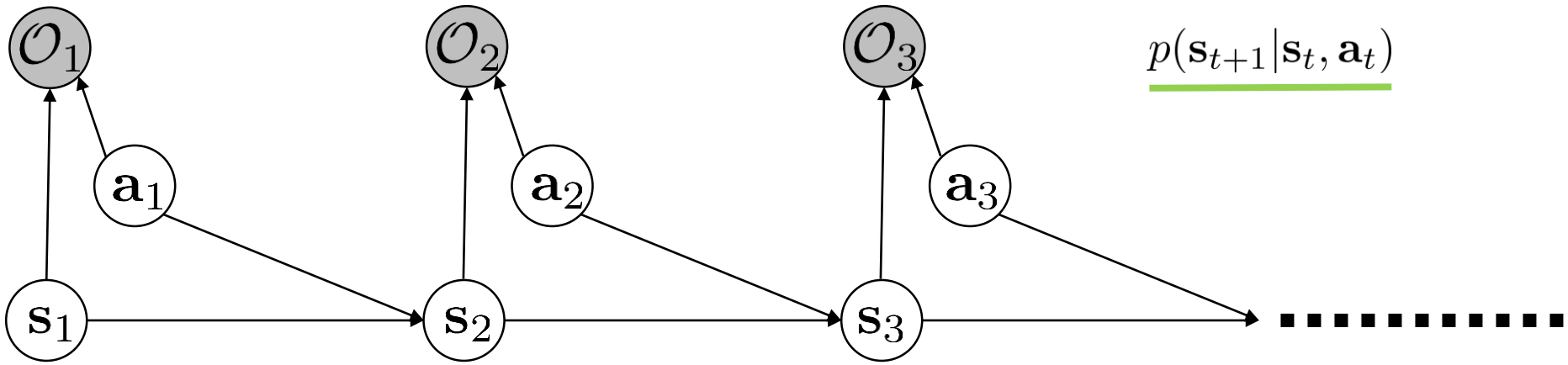
discounted SOC: $Q_t(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma E[V_{t+1}(\mathbf{s}_{t+1})]$

explicit temperature: $V_t(\mathbf{s}_t) = \alpha \log \int \exp\left(\frac{1}{\alpha} Q_t(\mathbf{s}_t, \mathbf{a}_t)\right) d\mathbf{a}_t$

Forward messages

$$p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) \propto \exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$



$$\alpha_t(\mathbf{s}_t) = p(\mathbf{s}_t|\mathcal{O}_{1:t-1})$$

$$= \int p(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) p(\mathbf{a}_{t-1}|\mathbf{s}_{t-1}, \mathcal{O}_{t-1}) \frac{p(\mathbf{s}_{t-1}|\mathcal{O}_{1:t-2})}{\alpha_{t-1}(\mathbf{s}_{t-1})} d\mathbf{s}_{t-1} d\mathbf{a}_{t-1}$$

$$\alpha_1(\mathbf{s}_1) = p(\mathbf{s}_1) \text{ (usually known)}$$

$$p(\mathbf{a}_{t-1}|\mathbf{s}_{t-1}, \mathcal{O}_{t-1}) = \frac{p(\mathcal{O}_{t-1}|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) p(\mathbf{a}_{t-1}|\mathbf{s}_{t-1})}{p(\mathcal{O}_{t-1}|\mathbf{s}_{t-1})}$$

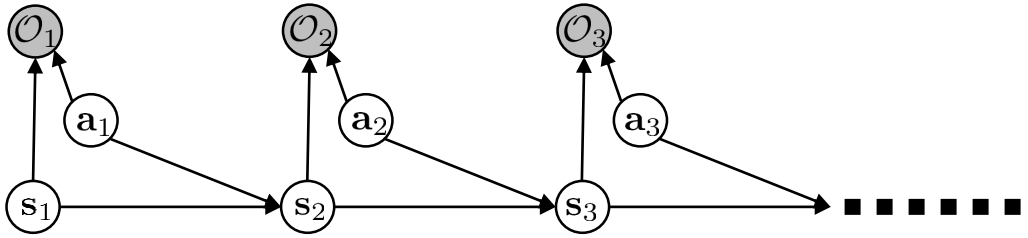
what if we want $p(\mathbf{s}_t|\mathcal{O}_{1:T})$?

$$p(\mathbf{s}_t|\mathcal{O}_{1:T}) = \frac{p(\mathbf{s}_t, \mathcal{O}_{1:T})}{p(\mathcal{O}_{1:T})} = \frac{p(\mathcal{O}_{t:T}|\mathbf{s}_t) p(\mathbf{s}_t, \mathcal{O}_{1:t-1})}{p(\mathcal{O}_{1:T})} \propto \beta_t(\mathbf{s}_t) \frac{p(\mathbf{s}_t|\mathcal{O}_{1:t-1})}{\alpha_t(\mathbf{s}_t)} p(\mathcal{O}_{1:t-1}) \propto \beta_t(\mathbf{s}_t)\alpha_t(\mathbf{s}_t)$$

$\beta_t(\mathbf{s}_t)$

Summary

1. Probabilistic graphical model for optimal control



2. Control = inference (similar to HMM, EKF, etc.)

3. Very similar to dynamic programming, value iteration, etc. (but “soft”)

Q-learning with soft optimality

standard Q-learning: $\phi \leftarrow \phi + \alpha \nabla_\phi Q_\phi(\mathbf{s}, \mathbf{a})(r(\mathbf{s}, \mathbf{a}) + \gamma V(\mathbf{s}') - Q_\phi(\mathbf{s}, \mathbf{a}))$

target value: $V(\mathbf{s}') = \max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')$

soft Q-learning: $\phi \leftarrow \phi + \alpha \nabla_\phi Q_\phi(\mathbf{s}, \mathbf{a})(r(\mathbf{s}, \mathbf{a}) + \gamma V(\mathbf{s}') - Q_\phi(\mathbf{s}, \mathbf{a}))$

target value: $V(\mathbf{s}') = \text{soft}\max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}') = \log \int \exp(Q_\phi(\mathbf{s}', \mathbf{a}'))d\mathbf{a}'$

$\pi(\mathbf{a}|\mathbf{s}) = \exp(Q_\phi(\mathbf{s}, \mathbf{a}) - V(\mathbf{s})) = \exp(A(\mathbf{s}, \mathbf{a}))$

1. take some action $\mathbf{a}_i$ and observe $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_i', r_i)$, add it to $\mathcal{R}$
2. sample mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}_j', r_j\}$ from $\mathcal{R}$ uniformly
3. compute $y_j = r_j + \gamma \text{soft}\max_{\mathbf{a}_j'} Q_{\phi'}(\mathbf{s}_j', \mathbf{a}_j')$ using *target* network $Q_{\phi'}$
4. $\phi \leftarrow \phi - \alpha \sum_j \frac{dQ_\phi}{d\phi}(\mathbf{s}_j, \mathbf{a}_j)(Q_\phi(\mathbf{s}_j, \mathbf{a}_j) - y_j)$
5. update ϕ': copy ϕ every N steps, or Polyak average $\phi' \leftarrow \tau\phi' + (1-\tau)\phi$

Policy gradient with soft optimality

$$\pi(\mathbf{a}|\mathbf{s}) = \exp(Q_\phi(\mathbf{s},\mathbf{a}) - V(\mathbf{s})) \text{ optimizes } \sum_t E_{\pi(\mathbf{s}_t,\mathbf{a}_t)}[r(\mathbf{s}_t,\mathbf{a}_t)] + E_{\pi(\mathbf{s}_t)}[\mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t))]$$

policy entropy

intuition: $\pi(\mathbf{a}|\mathbf{s}) \propto \exp(Q_\phi(\mathbf{s},\mathbf{a}))$ when π minimizes $D_{\text{KL}}(\pi(\mathbf{a}|\mathbf{s}) \| \frac{1}{Z}\exp(Q(\mathbf{s},\mathbf{a})))$

$$D_{\text{KL}}(\pi(\mathbf{a}|\mathbf{s}) \| \tfrac{1}{Z}\exp(Q(\mathbf{s},\mathbf{a}))) = E_{\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \mathcal{H}(\pi)$$

often referred to as “entropy regularized” policy gradient

combats premature entropy collapse

turns out to be closely related to soft Q-learning:
see Haarnoja et al. ‘17 and Schulman et al. ‘17

Ziebart et al. ‘10 “Modeling Interaction via the Principle of Maximum Causal Entropy”

Benefits of soft optimality

- Improve exploration and prevent entropy collapse
- Principled approach to break ties
- Can reduce to hard optimality as reward magnitude increases
- Good model for modeling human behavior (more on this later)
- Compositionality

$Q_1(\mathbf{s}, \mathbf{a})$: Q-function for reward $r_1(\mathbf{s}, \mathbf{a})$

$Q_2(\mathbf{s}, \mathbf{a})$: Q-function for reward $r_2(\mathbf{s}, \mathbf{a})$

$Q_1(\mathbf{s}, \mathbf{a}) + Q_2(\mathbf{s}, \mathbf{a})$ is *not* the Q-function for $r_1(\mathbf{s}, \mathbf{a}) + r_2(\mathbf{s}, \mathbf{a})$ under conventional (hard) optimality

soft optimality: $\pi_1(\tau)\pi_2(\tau) = \exp(\log p_1(\tau) + \log p_2(\tau))$

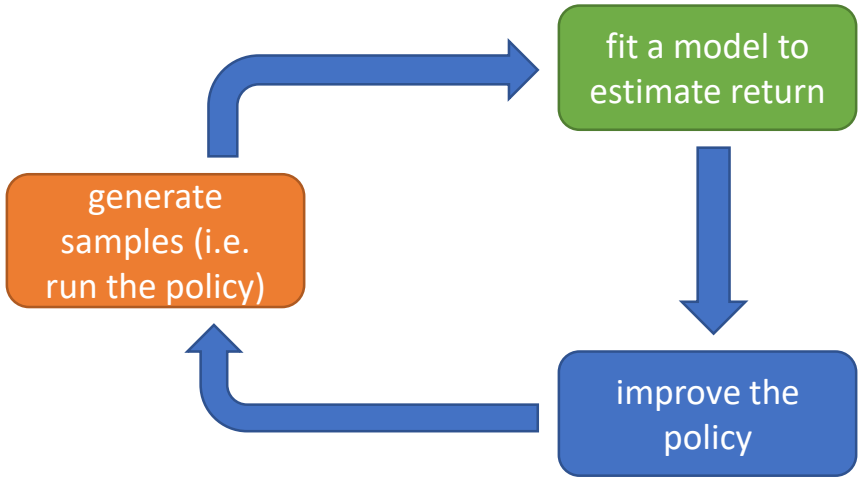
recall that $\pi_1(\mathbf{a}|\mathbf{s}) \propto \exp(Q_1(\mathbf{s}, \mathbf{a}))$

turns out that $Q_1 + Q_2$ *is* the (soft) Q-function for $r_1 + r_2$

Review

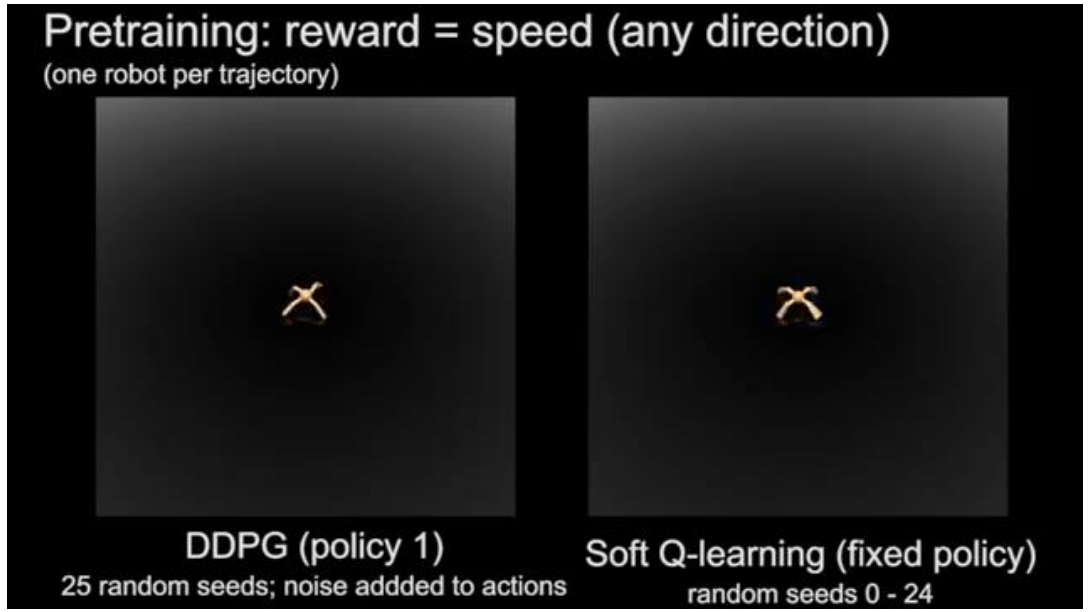
- Reinforcement learning can be viewed as inference in a graphical model
  - Value function is a backward message
  - Maximize reward and entropy (the bigger the rewards, the less entropy matters)
- Soft Q-learning
- Entropy-regularized policy gradient

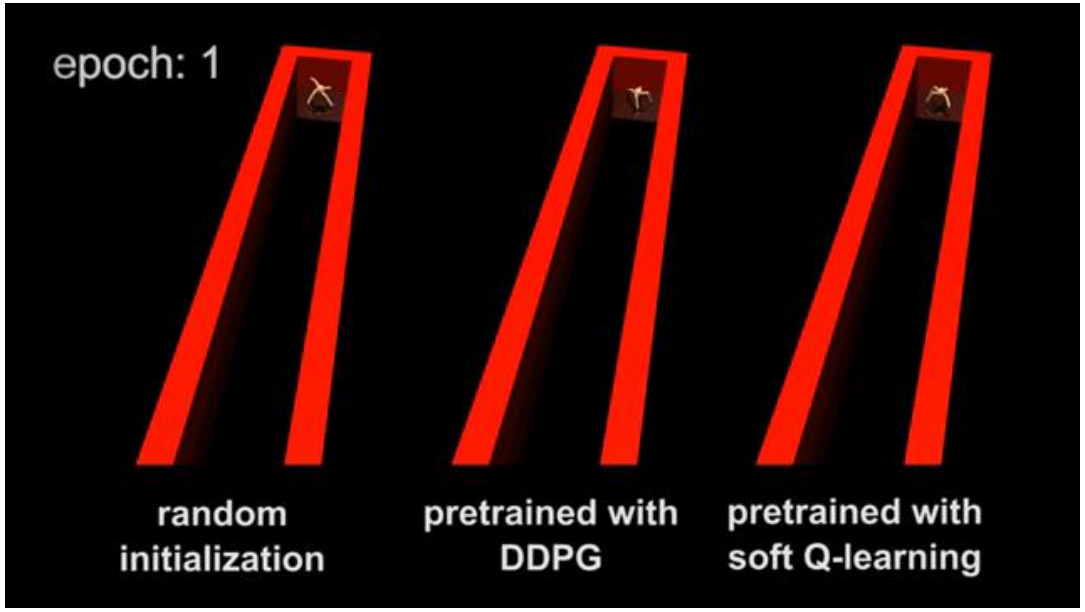
$$Q_\phi(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \gamma \text{soft}\max_{\mathbf{a}'} Q_\phi(\mathbf{s}', \mathbf{a}')$$



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

Soft optimality example





Haarnoja*, Tang*, Abbeel, L. (2017)

Soft optimality suggested readings

- Todorov. (2006). Linearly solvable Markov decision problems: one framework for reasoning about soft optimality.
- Todorov. (2008). General duality between optimal control and estimation: primer on the equivalence between inference and control.
- Kappen. (2009). Optimal control as a graphical model inference problem: frames control as an inference problem in a graphical model.
- Ziebart. (2010). Modeling interaction via the principle of maximal causal entropy: connection between soft optimality and maximum entropy modeling.
- Rawlik, Toussaint, Vijaykumar. (2013). On stochastic optimal control and reinforcement learning by approximate inference: temporal difference style algorithm with soft optimality.
- Haarnoja*, Tang*, Abbeel, L. (2017). Reinforcement learning with deep energy based models: soft Q-learning algorithm, deep RL with continuous actions and soft optimality
- Nachum, Norouzi, Xu, Schuurmans. (2017). Bridging the gap between value and policy based reinforcement learning.
- Schulman, Abbeel, Chen. (2017). Equivalence between policy gradients and soft Q-learning.

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

— BREAK —

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

less advanced topics

more advanced topics

Where does the reward come from?

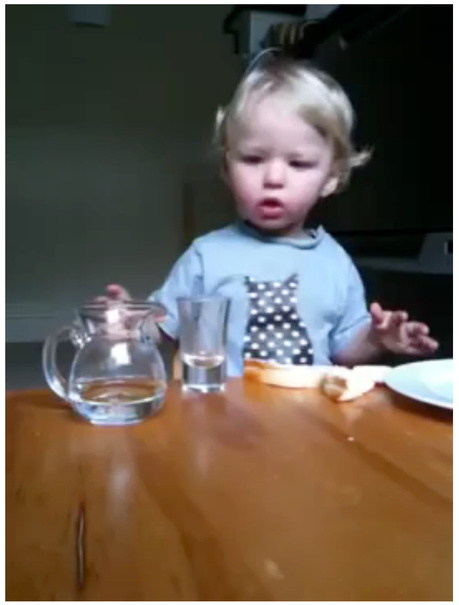
Computer Games

reward

Mnih et al. '15

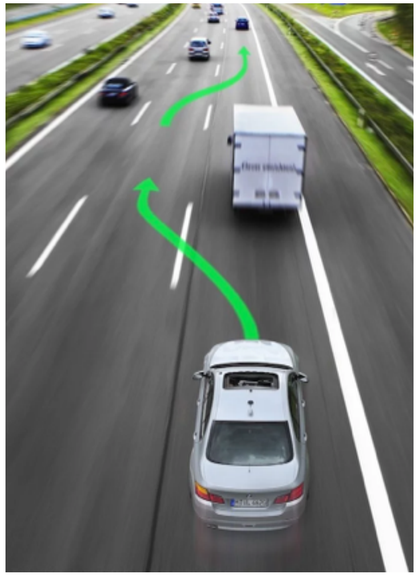
Real World Scenarios

robotics



dialogue

autonomous driving



what is the reward?
often use a proxy

often easier to provide expert data

Approach: infer reward function from roll-outs of expert policy

Inverse Optimal Control / Inverse Reinforcement Learning:

infer reward function from demonstrations

(IOC/IRL) (Kalman '64, Ng & Russell '00)

given:
- state & action space
- roll-outs from π^*
- dynamics model [sometimes]

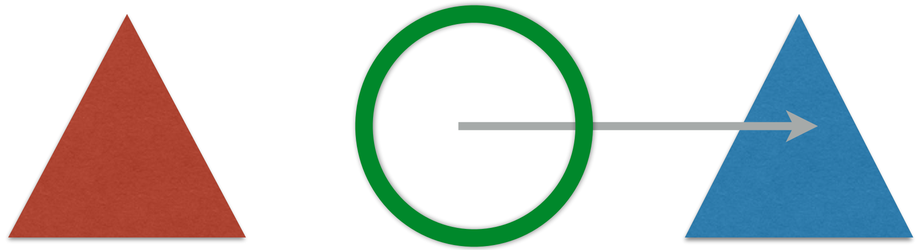
goal:
- recover reward function
- then use reward to get policy

Challenges

underdefined problem

difficult to evaluate a learned reward

demonstrations may not be precisely optimal



Maximum Entropy Inverse RL

(Ziebart et al. '08)

handle ambiguity using probabilistic model of behavior

Notation:

$$\tau = \{s_1, a_1, ..., s_t, a_t, ..., s_T\}$$

trajectory

$$R_\psi(\tau) = \sum_t r_\psi(s_t, a_t)$$

learned reward

$$\mathcal{D} : \{\tau_i\} \sim \pi^*$$

expert demonstrations

MaxEnt Formulation:

$$p(\tau) = \frac{1}{Z} \exp(R_\psi(\tau))$$

$$Z = \int \exp(R_\psi(\tau)) d\tau$$

$$\max_\psi \sum_{\tau \in \mathcal{D}} \log p_{r_\psi}(\tau)$$

(energy-based model for behavior)

Maximum Entropy Inverse RL

(Ziebart et al. '08)

handle ambiguity using probabilistic model of behavior

0. Initialize ψ, gather demonstrations $\mathcal{D}$
1. Solve for optimal policy $\pi(\mathbf{a}|\mathbf{s})$ w.r.t. reward r_ψ
2. Solve for state visitation frequencies $p(\mathbf{s}|\psi)$
3. Compute gradient $\nabla_\psi \mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{\tau_d \in \mathcal{D}} \frac{dr_\psi}{d\psi}(\tau_d) - \sum_s p(s|\psi) \frac{dr_\psi}{d\psi}(s)$
4. Update ψ with one gradient step using $\nabla_\psi \mathcal{L}$

How can we handle more complex problem settings?

How can we:

(1) handle unknown dynamics? (2) avoid solving the MDP in inner loop

$$\max_{\psi} \sum_{\tau \in \mathcal{D}} \log p_{r_\psi}(\tau)$$

$$p(\tau) = \frac{1}{Z} \exp(R_\psi(\tau))$$

$$Z = \int \exp(R_\psi(\tau)) d\tau$$

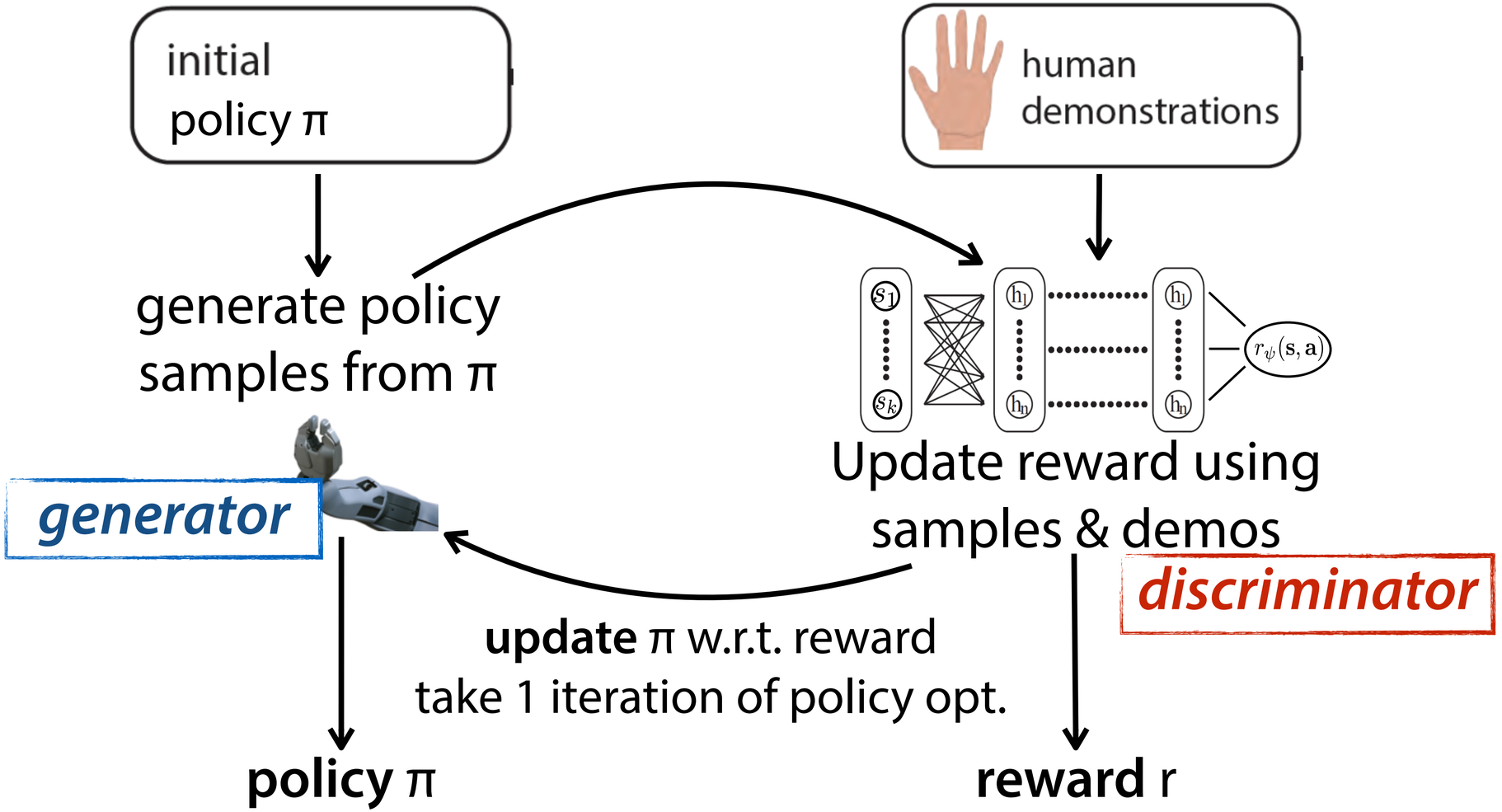
sample-based estimate Z

learn dynamics to estimate Z

[by constructing a policy]

guided cost learning & generative adversarial imitation algorithm

(Finn et al. ICML '16, Ho & Ermon NIPS '16)



update reward in inner loop of policy optimization

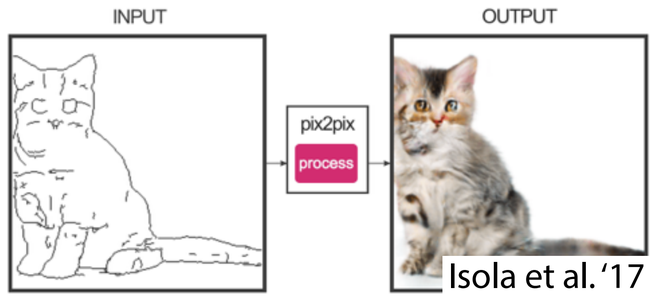
Aside: Generative Adversarial Networks

(Goodfellow et al. '14)

Similar to inverse RL, **GANs** learn an objective for generative modeling.

Zhu et al. '17

Arjovsky et al. '17



Isola et al. '17

	trajectory τ	⟷	**sample x**
Inverse RL	**policy** π~q(τ)	⟷	**generator** G GANs
	reward r	⟶	**discriminator** D

(Finn*, Christiano*, et al. '16)

Connection to Generative Adversarial Networks

(Goodfellow et al. '14)

	trajectory τ	⟷	**sample x**
Inverse RL	**policy** π~q(τ)	⟷	**generator** G GANs
	reward r	⟶	**discriminator** D

data distribution p

Reward/discriminator optimization:

GCL:

$$D^*(\tau) = \frac{p(\tau)}{p(\tau) + q(\tau)}$$

$$D_\psi(\tau) = \frac{\frac{1}{Z}\exp(R_\psi)}{\frac{1}{Z}\exp(R_\psi) + q(\tau)}$$

GAIL:

$$D_\psi(\tau) = \text{ some classifier}$$

Both:

$$\mathcal{L}_{\text{discriminator}}(\psi) = \mathbb{E}_{\tau\sim p}[-\log D_\psi(\tau)] + \mathbb{E}_{\tau\sim q}[-\log(1 - D_\psi(\tau))]$$

(Finn*, Christiano*, et al. '16)

Connection to Generative Adversarial Networks

(Goodfellow et al. '14)

	trajectory τ	⟷	**sample x**
Inverse RL	**policy** π~q(τ)	⟷	**generator** G GANs
	reward r	⟶	**discriminator** D

data distribution p

Policy/Generator optimization:

$$\mathcal{L}_{\text{generator}}(\theta) = \mathbb{E}_{\tau\sim q}[\log(1 - D_\psi(\tau)) - \log D_\psi(\tau)]$$

$$= \mathbb{E}_{\tau\sim q}[\log q(\tau) + \log Z - R_\psi(\tau)]$$

entropy-regularized RL

Unknown dynamics: train generator/policy with RL

Baram et al. ICML '17: use dynamics model to backprop through discriminator

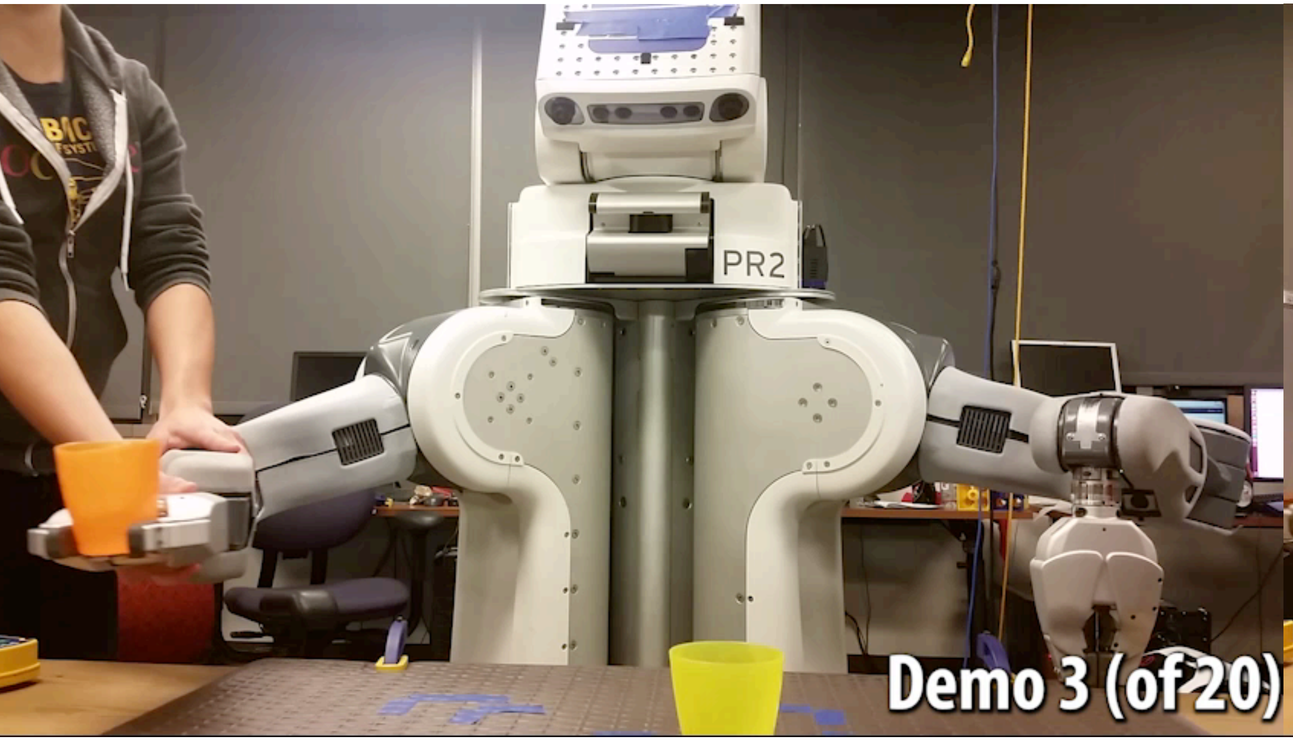
(Finn*, Christiano*, et al. '16)

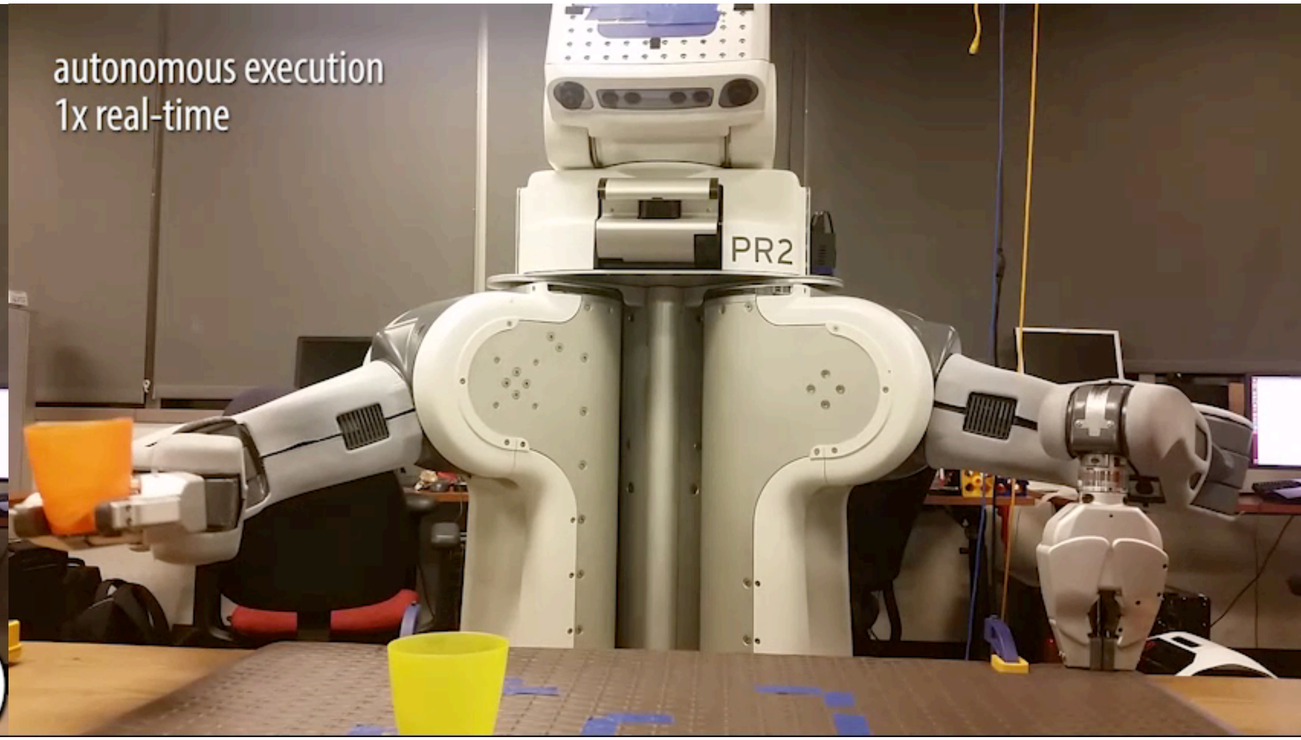
Guided Cost Learning Results

Finn et al. ICML '16

demonstrations

learned behavior



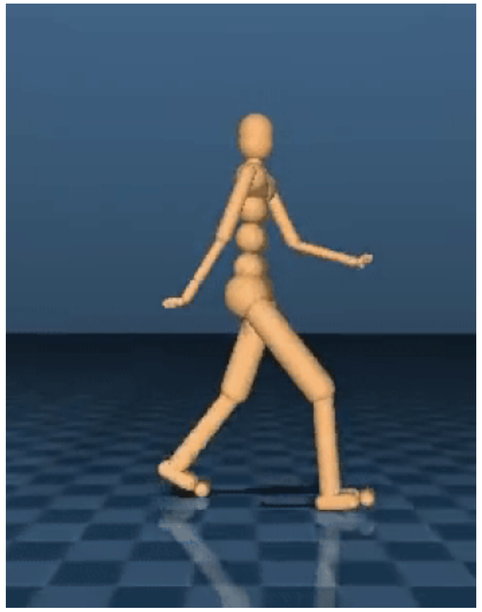


Generative Adversarial Imitation Learning Results

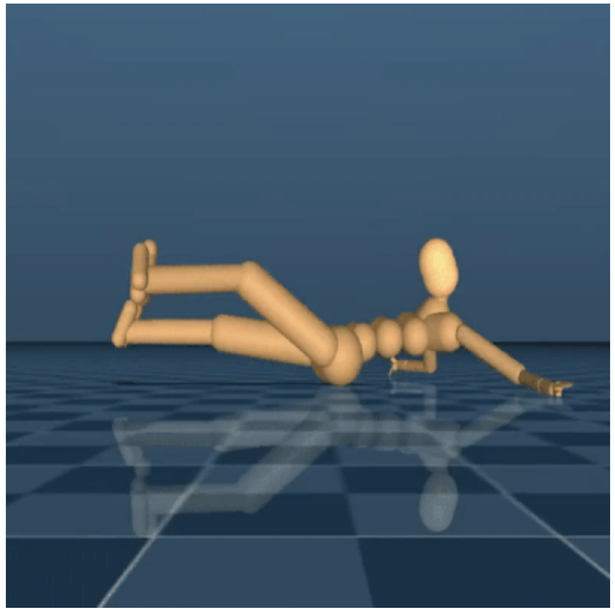
Ho & Ermon NIPS '16

learned behaviors from human motion capture

Merel et al. '17



walking



falling & getting up

Inverse RL Review

Goal: infer reward function underlying expert demonstrations

Evaluating the partition function

- initial approaches solve MDP in inner loop of IRL and/or assume known dynamics
- with unknown dynamics, estimate Z using samples

Connection to Generative Adversarial Networks

- sampling-based MaxEnt IRL is a GAN with a special form of discriminator, and uses RL to optimize generator

Suggested Reading on Inverse RL

Classic Papers

- **Abbeel & Ng ICML '04.** *Apprenticeship Learning via Inverse Reinforcement Learning*. Good introduction to inverse reinforcement learning
- **Ziebart et al. AAAI '08**. *Maximum Entropy Inverse Reinforcement Learning*. Introduction of probabilistic method for inverse reinforcement learning

Modern Papers

- **Wulfmeier et al. arXiv '16**. *Deep Maximum Entropy Inverse Reinforcement Learning*. MaxEnt IRL using deep reward functions
- **Finn et al. ICML '16**. *Guided Cost Learning*. Sampling-based method for MaxEnt IRL that handles unknown dynamics and deep reward functions
- **Ho & Ermon NIPS '16**. *Generative Adversarial Imitation Learning*. IRL method building on Abbeel & Ng '04 using generative adversarial networks

Further Reading on Inverse RL

MaxEnt-based IRL: Ziebart et al. AAAI '08, Wulfmeier et al. arXiv '16, Finn et al. ICML '16

Adversarial IRL: Ho & Ermon NIPS '16, Finn*, Christiano* et al. arXiv '16, Baram et al. ICML '17

Handling multimodality: Li et al. arXiv '17, Hausman et al. arXiv '17, Wang, Merel et al. '17

Handling domain shift: Stadie et al. ICLR '17

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

less advanced topics

— BREAK —

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

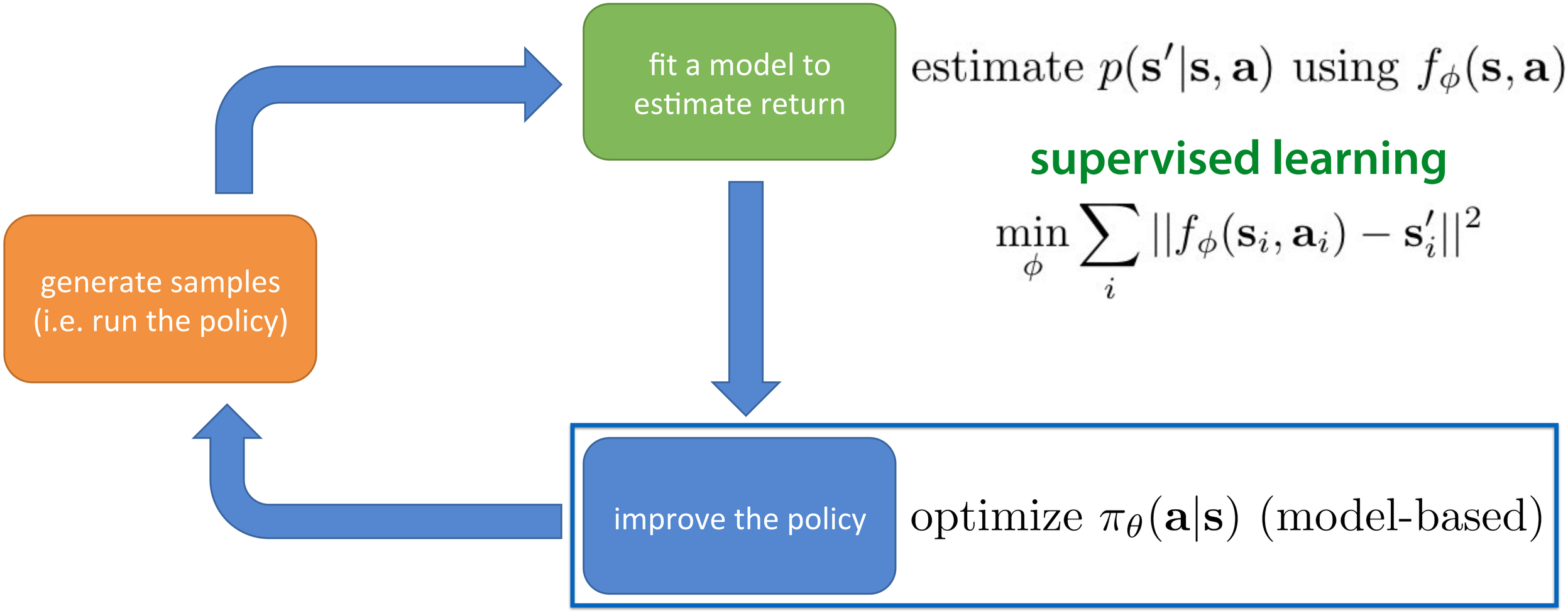
more advanced topics

Next: Model-based [Deep] RL

Everything so far: *model-free*

Main idea: learn model of environment

Why? leads to better efficiency



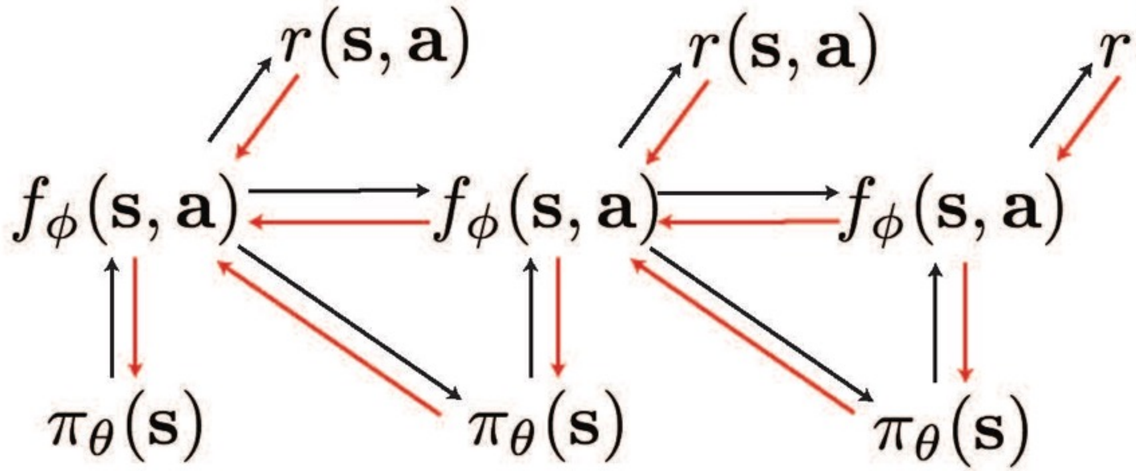
$$\text{estimate } p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \text{ using } f_\phi(\mathbf{s},\mathbf{a})$$

supervised learning

$$\min_\phi \sum_i ||f_\phi(\mathbf{s}_i,\mathbf{a}_i) - \mathbf{s}_i'||^2$$

$$\text{optimize } \pi_\theta(\mathbf{a}|\mathbf{s}) \text{ (model-based)}$$

Approach 1: Backprop through model to optimize policy



backpropagate

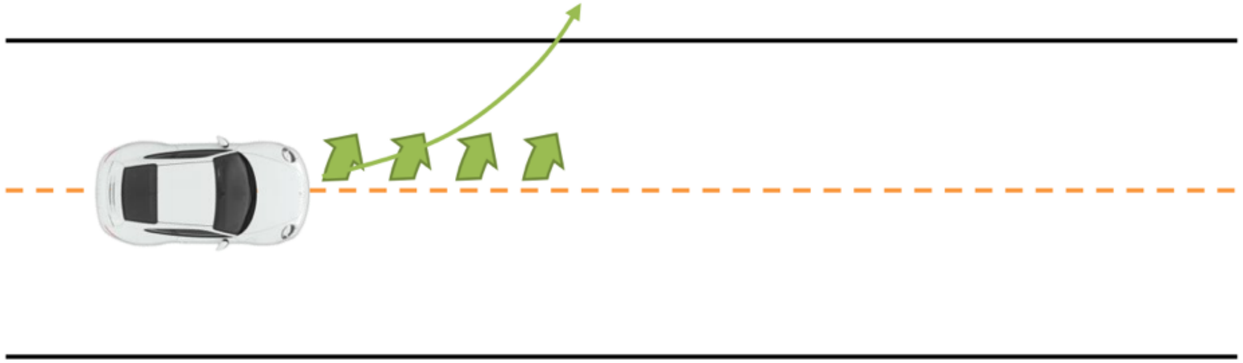
$$\max_\theta \sum_t r(\mathbf{s}_t, \mathbf{a}_t)$$

Algorithm:

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model $f_\phi(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$
3. backpropagate through $f_\phi(\mathbf{s}, \mathbf{a})$ into policy to optimize $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
4. run $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, appending visited tuples $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to $\mathcal{D}$

How can this approach fail?

Policy optimization will exploit imprecisions in model



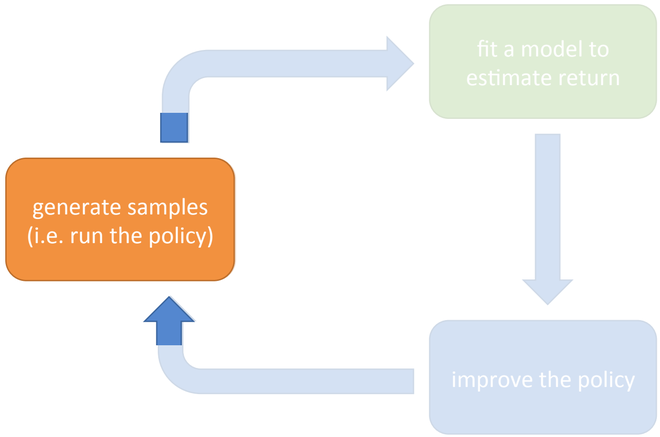
Refitting model using on-policy data will help.

But generally, learning a good global model is hard.

Approach 2: Plan & replan using model

model-predictive control (MPC)

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn model $f_\phi(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$
3. backpropagate through $f_\phi(\mathbf{s}, \mathbf{a})$ to choose actions.
4. execute the first planned action, observe resulting state $\mathbf{s}'$
5. append $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ to dataset $\mathcal{D}$



REPLANNING HELPS WITH MODEL ERRORS

+ replan to correct for model errors

- compute intensive

Approach 3: Using learned local models

Levine & Abbeel NIPS '14

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ to collect $\mathcal{D} = \{(\mathbf{s}, \mathbf{a}, \mathbf{s}')_i\}$
2. learn local model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i||^2$

   e.g. using linear regression
3. update local policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ using local model f_ϕ with KL constraint.
4. run $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$, putting visited tuples $(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ in $\mathcal{D}$

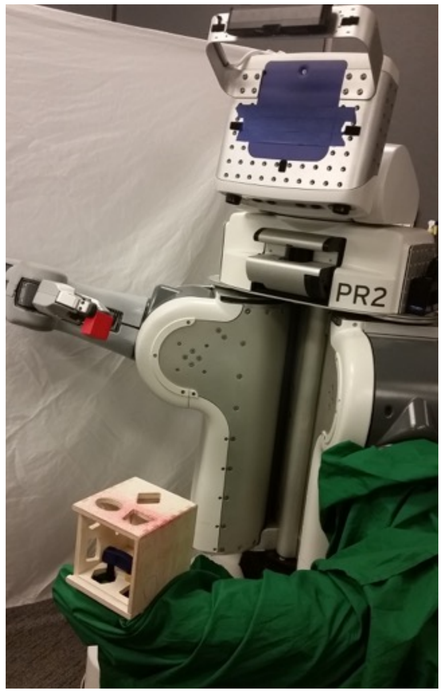
Guided policy search: supervise one global policy using multiple local policies

Case Study: Guided Policy Search

Objective: maximize reward under the policy

$$\max_{\theta} \mathbb{E}_{\pi_\theta}[r(\tau)]$$

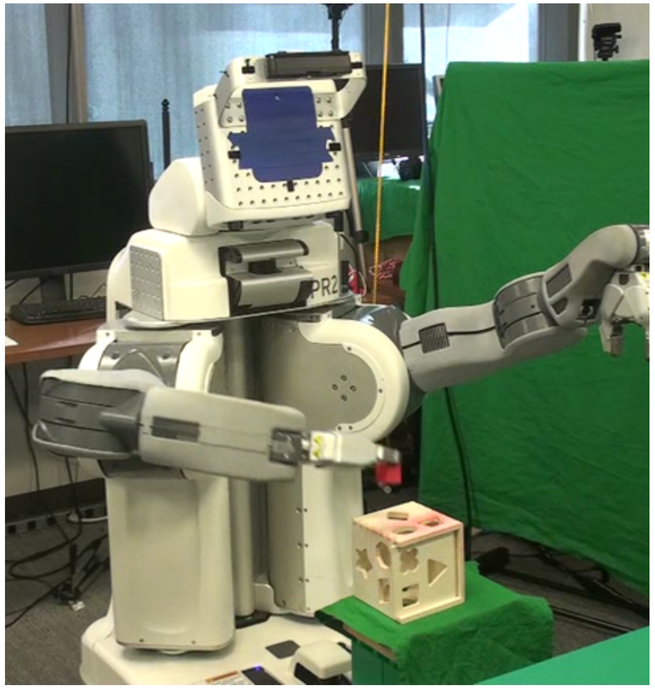
Training time



$\mathbf{s}_t \longrightarrow \mathbf{a}_t$

target pose known

Test time



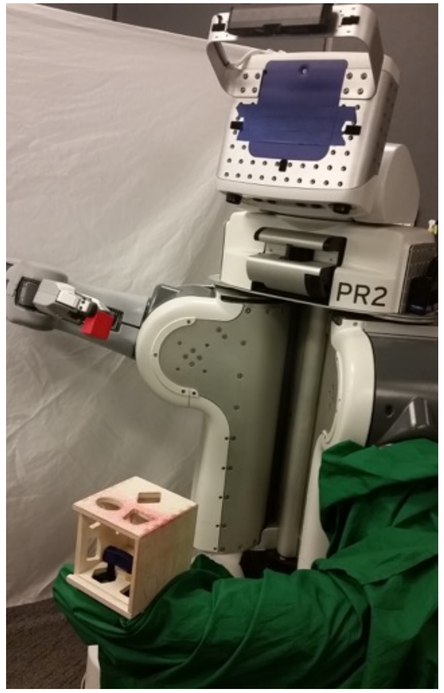
$\mathbf{o}(\mathbf{s}_t) \longrightarrow \mathbf{a}_t$

Guided Policy Search Overview

Objective: maximize reward under the policy

$$\max_{\theta} \mathbb{E}_{\pi_\theta}[r(\tau)]$$

Training time



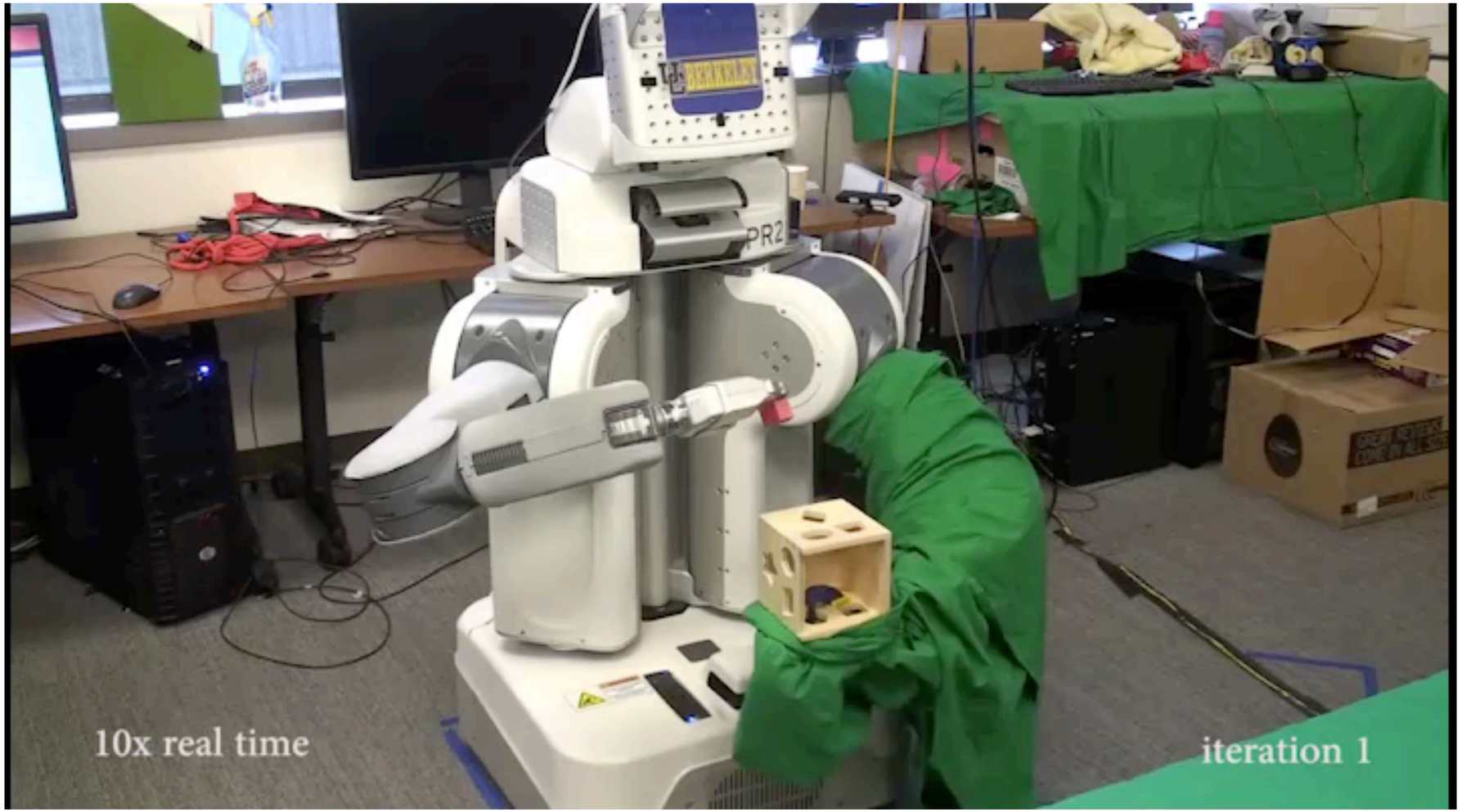
$\mathbf{s}_t \longrightarrow \mathbf{a}_t$

target pose known

- take samples for each target position
- fit local model and solve for local policy for each target position
- use supervision from local policies to train global neural network policy w/ vision

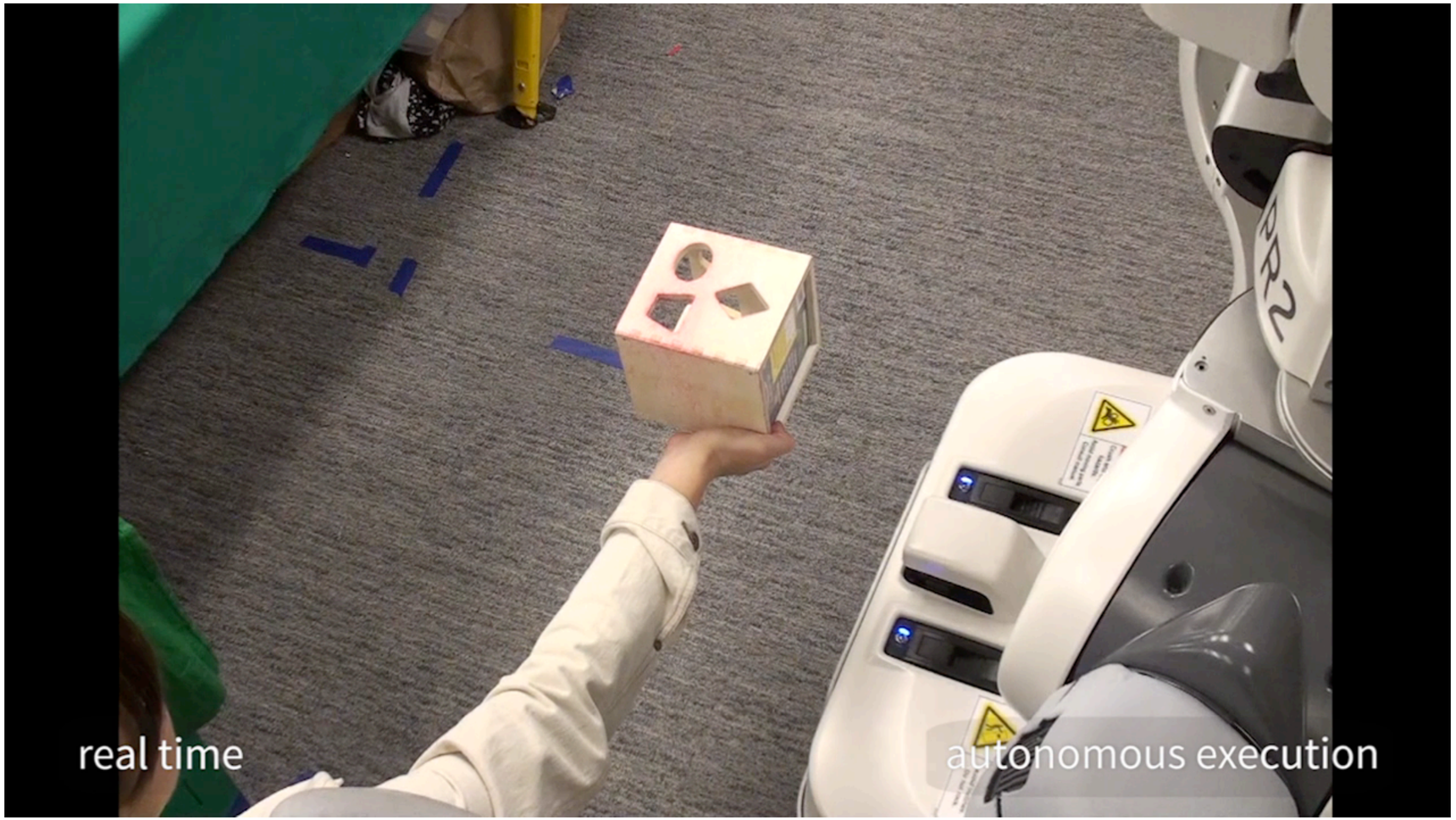
Guided Policy Search Learning

Levine*, Finn*, et al. JMLR ‘16



Guided Policy Search Results

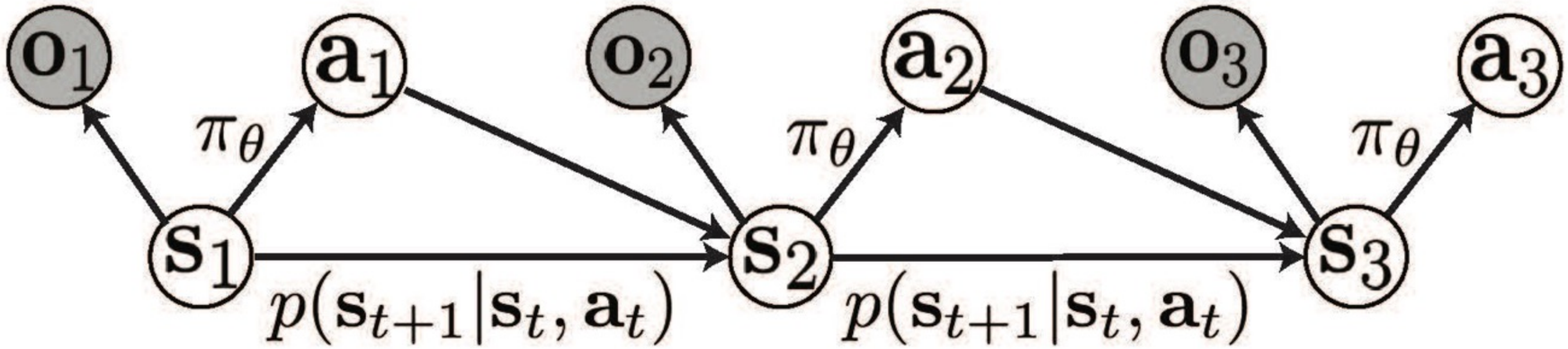
Levine*, Finn*, et al. JMLR ‘16



< 300 trials = 25 min of robot time (per task)

+ efficiently learn complex vision-based skills - requires state during training

Only access to high-dimensional observations (i.e. images)?

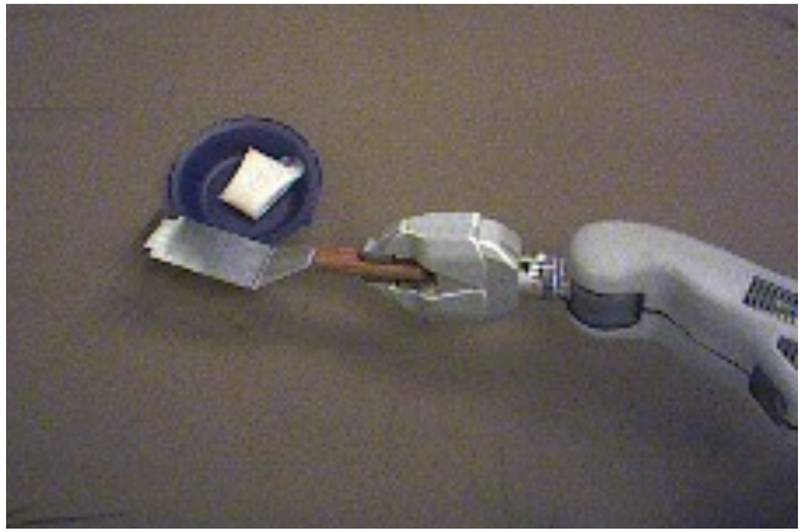


also: no reward signal with only observations

Only access to high-dimensional observations (i.e. images)?

also: no reward signal with only observations

one option: provide image of goal

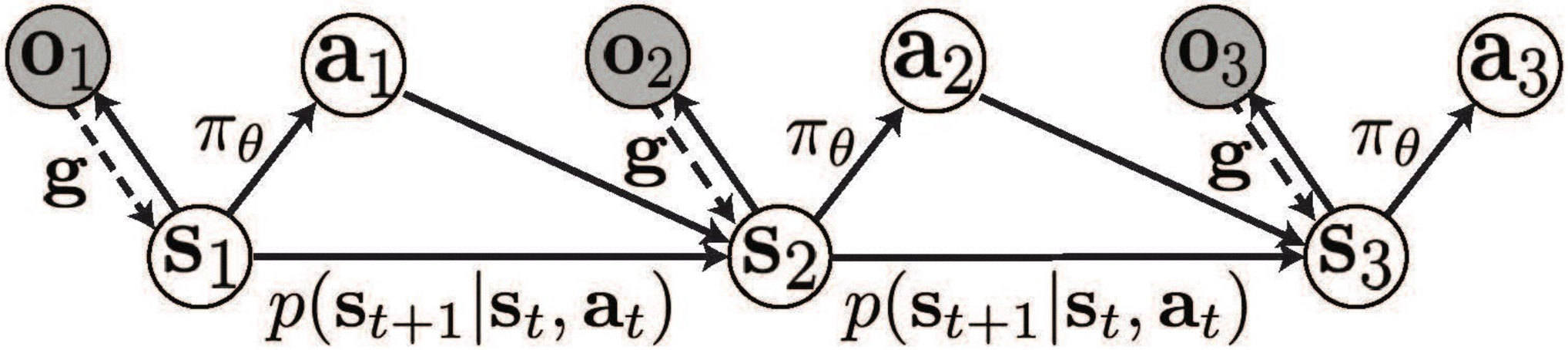


Approaches

1. Learn model in latent space
2. Learn model of observations (e.g. video)
3. Inverse models [won't cover]

Learning in Latent Space

Key idea: learn embedding $g(\mathbf{o}_t)$, then learn model in latent space



What do we want g to be?

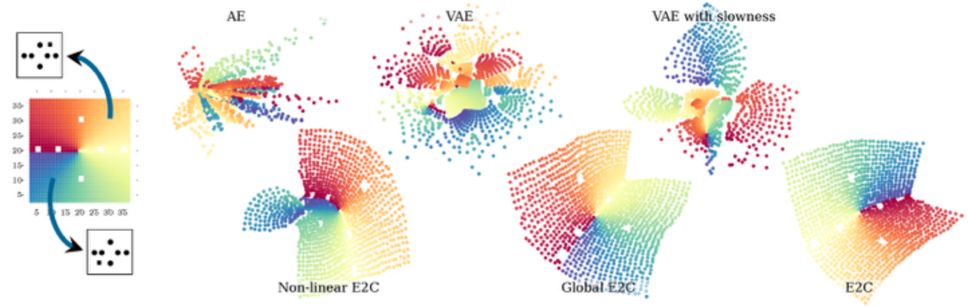
Learning in Latent Space

Key idea: learn embedding $\mathbf{s}_t = g(\mathbf{o}_t)$, then do model-based RL in latent space

Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images

Manuel Watter* **Jost Tobias Springenberg***
Joschka Boedecker
University of Freiburg, Germany
{watterm, springj, jboedeck}@cs.uni-freiburg.de

Martin Riedmiller
Google DeepMind
London, UK
riedmiller@google.com



NIPS 2015

Deep Spatial Autoencoders for Visuomotor Learning

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, Pieter Abbeel

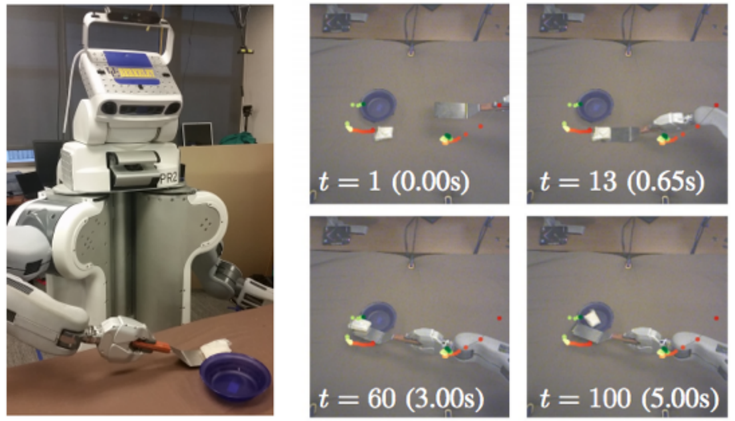
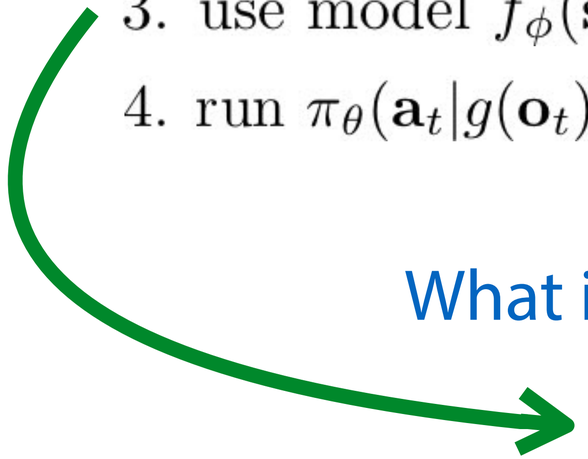


Fig. 1: PR2 learning to scoop a bag of rice into a bowl with a spatula (left) using a learned visual state representation (right).

ICRA 2016

Learning in Latent Space

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{o}_t)$ (e.g., exploratory policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$
2. learn latent embedding of observation $\mathbf{s}_t = g(\mathbf{o}_t)$ and dynamics model $\mathbf{s}' = f_\phi(\mathbf{s}, \mathbf{a})$
3. use model $f_\phi(\mathbf{s}, \mathbf{a})$ to optimize policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
4. run $\pi_\theta(\mathbf{a}_t|g(\mathbf{o}_t))$, appending visited tuples $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to $\mathcal{D}$



What is reward for optimizing policy?

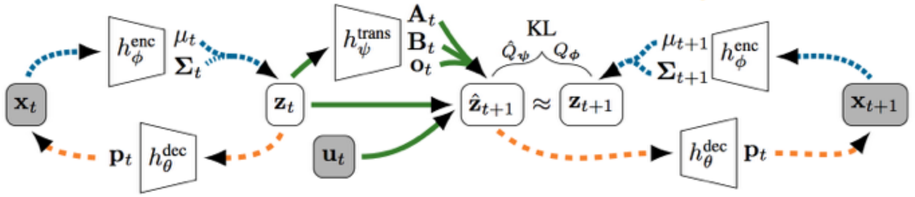
reward signal: $r(\mathbf{o}, \mathbf{a}) = r(\mathbf{a}) + ||g(\mathbf{o}) - g(\mathbf{o}_{\text{goal}})||$

Learning in Latent Space

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{s}_t)$ (e.g., exploratory policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$
2. learn latent embedding of observation $\mathbf{s}_t = g(\mathbf{o}_t)$ and dynamics model $\mathbf{s}' = f_\phi(\mathbf{s}, \mathbf{a})$
3. use model $f_\phi(\mathbf{s}, \mathbf{a})$ to optimize policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
4. run $\pi_\theta(\mathbf{a}_t|g(\mathbf{o}_t))$, appending visited tuples $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to $\mathcal{D}$

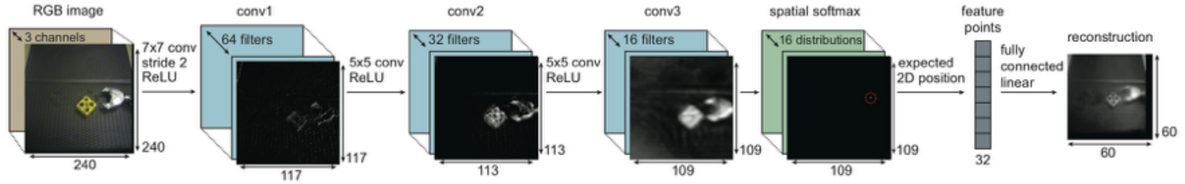
How to optimize latent embedding?

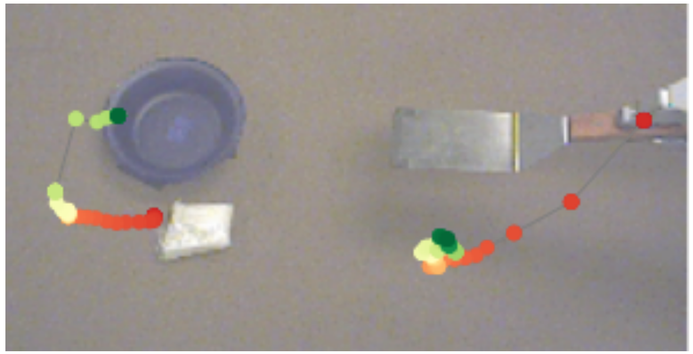
Watter et al. ‘15



learn embedding & model **jointly**

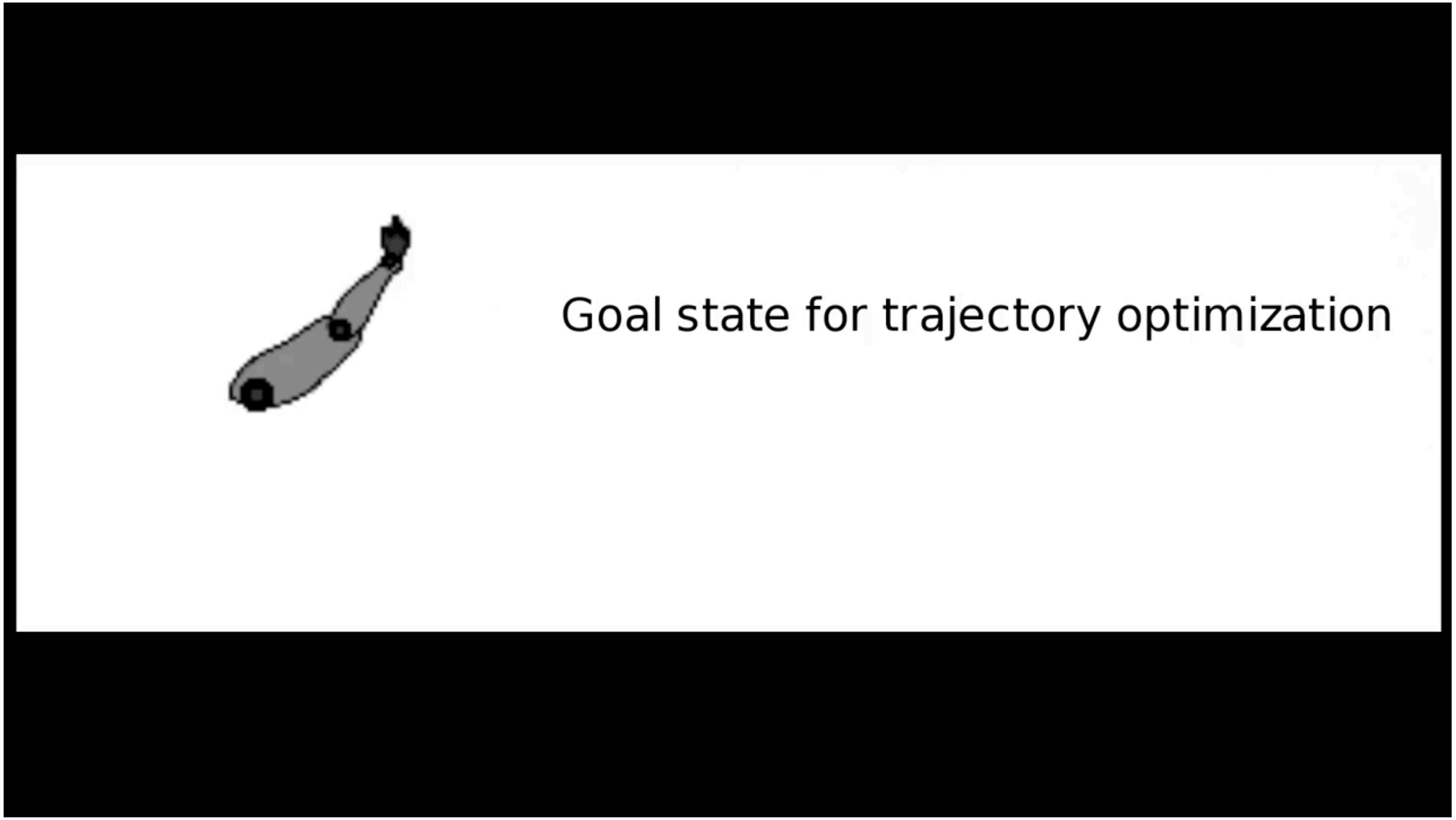
Finn et al. ‘16





embedding is **smooth and structured**

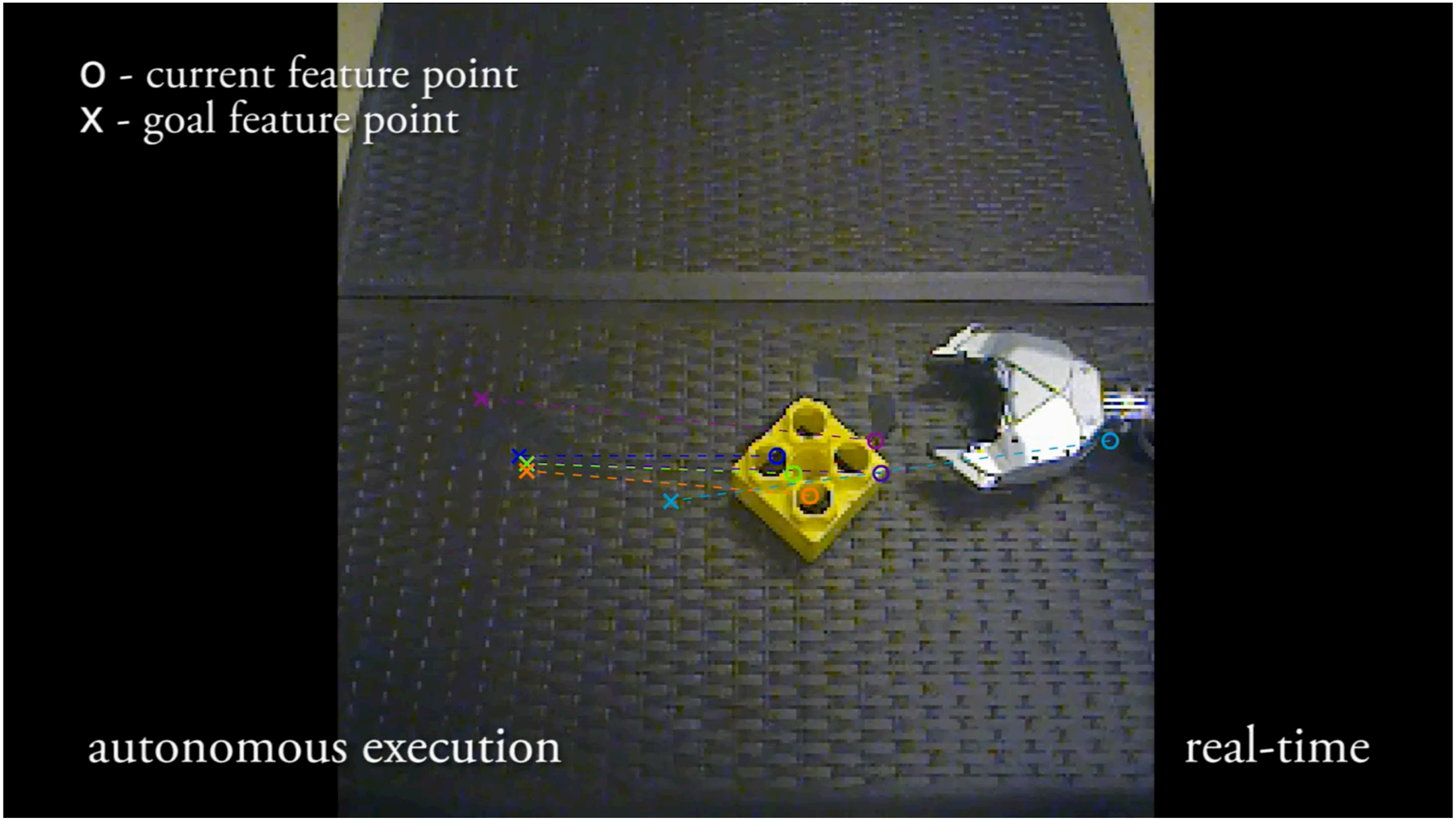
Learning in Latent Space



~300 trials = ~25 min of robot time (per task)

Watter et al. NIPS ‘15

Learning in Latent Space



125 trials = 11 min of robot time (per task)

Finn et al. ICRA ‘16

Learning in Latent Space

Pros:

+ Learn complex visual skills very efficiently

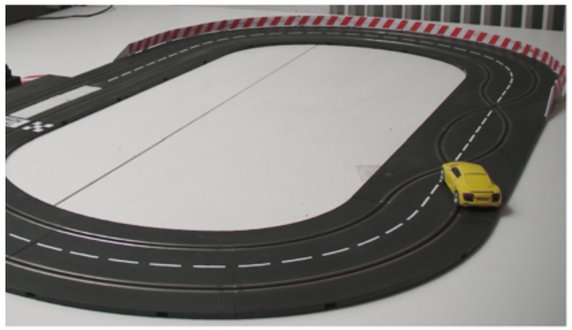
+ Structured representation enables effective learning

Cons:

- Reconstruction objectives might not recover the right representation

Aside: Low-dimensional embedding can also be useful for model-free approaches

model-free RL in latent space

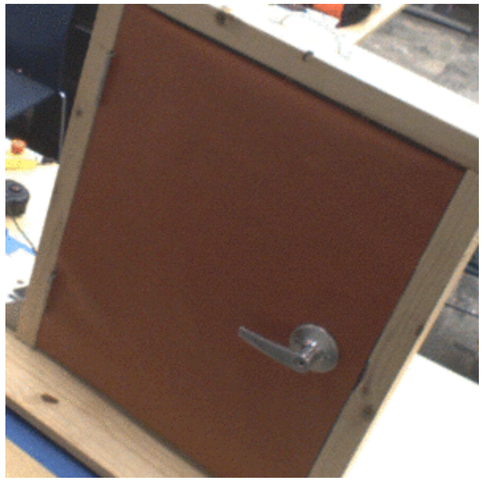


FQI in latent space
Lange et al. '12

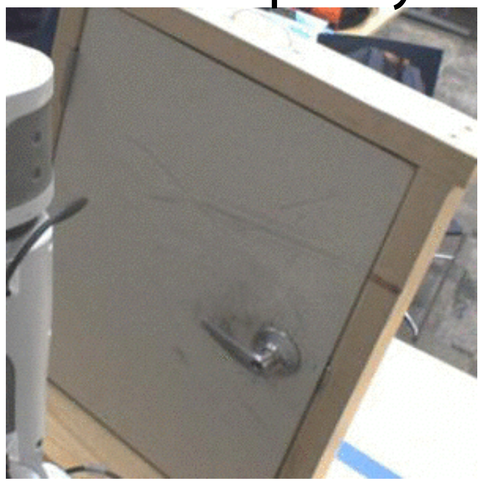
TRPO in latent space
Ghadirzadeh et al. '17

use embedding for reward function

video demonstration



learned policy



acquire reward using ImageNet features + model-free RL

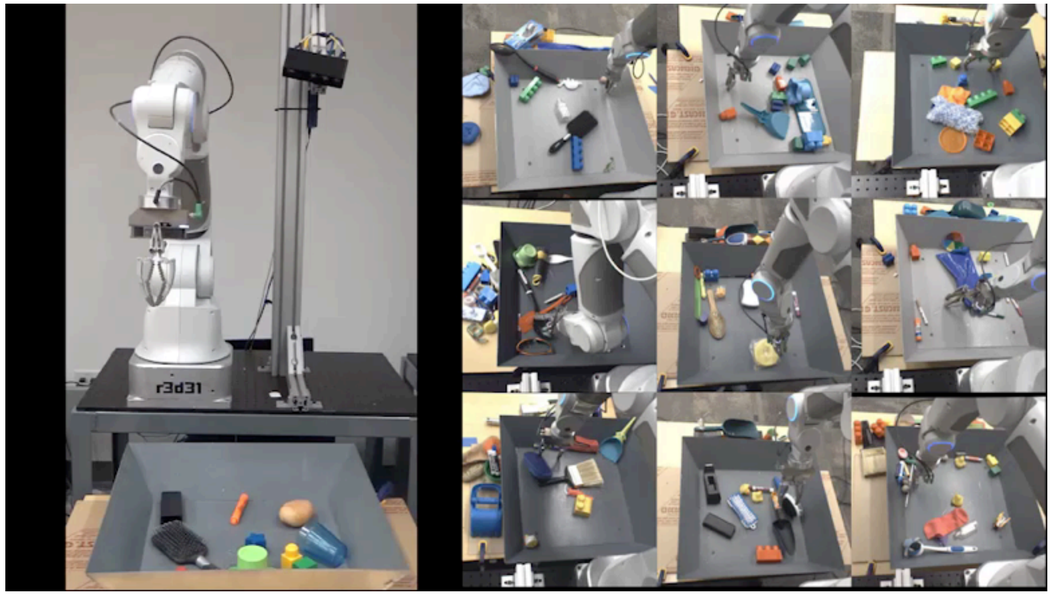
Sermanet et al. RSS '17

If you have a reward, you can predict it to form better latent space.
(Jaderberg et al. '17, Shelhamer et al. '17)

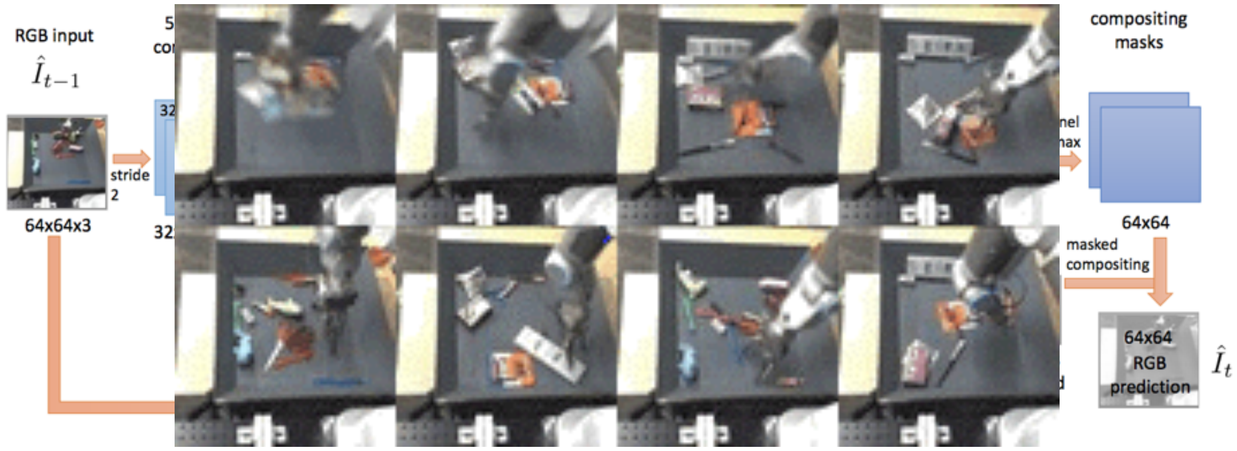
Modeling directly in observation space

Recall MPC

1. run base policy $\pi_0(\mathbf{a}_t|\mathbf{o}_t)$ (e.g., random policy) to collect $\mathcal{D} = \{(\mathbf{o}, \mathbf{a}, \mathbf{o}')_i\}$
2. learn model $f_\phi(\mathbf{o}, \mathbf{a})$ to minimize $\sum_i ||f_\phi(\mathbf{o}_i, \mathbf{a}_i) - \mathbf{o}'_i||^2$
3. backpropagate through $f_\phi(\mathbf{o}, \mathbf{a})$ to choose actions.
4. execute the first planned action, observe resulting state $\mathbf{o}'$
5. append $(\mathbf{o}, \mathbf{a}, \mathbf{o}')$ to dataset $\mathcal{D}$



action-conditioned video prediction



Finn et al. NIPS ’16, Finn & Levine ICRA ‘17

Modeling directly in observation space

Finn & Levine ICRA '17

<2 days of *unsupervised* robot time

Only human involvement: programming initial motions and providing objects to play with.

Modeling directly in observation space

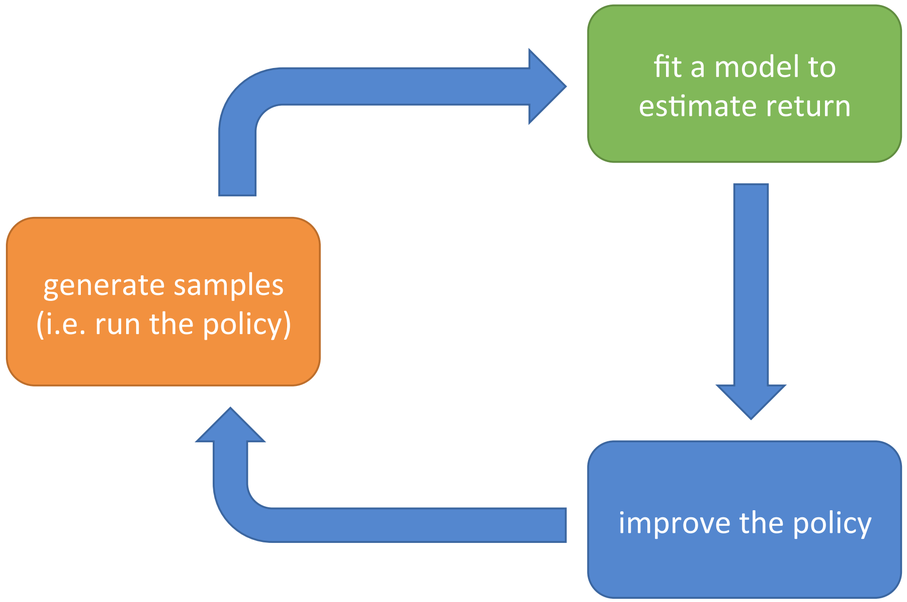
Pros:

+ Entirely self-supervised
+ Learn for a variety of tasks
+ More efficient than single-task model-free learning

Cons:

- Can’t [yet] handle as complex skills as model-free methods

Model-based RL Review



$\text{estimate } p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \text{ using } f_\phi(\mathbf{s},\mathbf{a})$

supervised learning

$\text{optimize } \pi_\theta(\mathbf{a}|\mathbf{s}) \text{ (model-based)}$

e.g., backprop through model

Correcting for model errors:
refitting model with new data, replanning with MPC, using local models

Model-based RL from raw observations:
learn latent space, typically with unsupervised learning, or
model & plan directly in observational space

Suggested Reading on Model-based RL

Tassa et al. IROS '12. *Synthesis and Stabilization of Complex Behaviors*. Good introduction to MPC with a known model

Levine*, Finn* et al. JMLR '16. *End-to-End Learning of Deep Visuomotor Policies*. Thorough paper on guided policy search for learning real robotic vision-based skills

Heess et al. NIPS '15. *Stochastic Value Gradients*. Backdrop through dynamics to assist model-free learner

Watter et al. NIPS '15. *Embed-to-Control*, Learn latent space and use model-baed RL in learned latent space to reach image of goal

Finn & Levine ICRA '17. *Deep Visual Foresight for Planning Robot Motion*. Plan using learned action-conditioned video prediction model

Further Reading on Model-based RL

Use known model: Tassa et al. IROS '12, Tan et al. TOG '14, Mordatch et al. TOG '14

Guided policy search: Levine*, Finn* et al. JMLR '16, Mordatch et al. RSS '14, NIPS '15

Backprop through model: Deisenroth et al. ICML '11, Heess et al. NIPS '15, Mishra et al. ICML '17, Degrave et al. '17, Henaff et al. '17

Inverse models: Agrawal et al. NIPS '16

MBRL in latent space: Watter et al. NIPS '15, Finn et al. ICRA '16

MPC with deep models: Lenz et al. RSS '15, Finn & Levine ICRA '17

Combining Model-Based & Model-Free:

- use roll-outs from model as experience: Sutton '90, Gu et al. ICML '16
- use model as baseline: Chebotar et al. ICML '17
- use model for exploration: Stadie et al. arXiv '15, Oh et al. NIPS '16
- model-free policy with planning capabilities: Tamar et al. NIPS '16, Pascanu et al. '17
- model-based look-ahead: Guo et al. NIPS '14, Silver et al. Nature '16

Model-Based vs. Model-Free Learning

Models:

+ Easy to collect data in a scalable way (self-supervised)
+ Possibility to transfer across tasks
+ Typically require a smaller quantity of supervised data
- Models don’t optimize for task performance
- Sometimes harder to learn than a policy
- Often need assumptions to learn complex skills (continuity, resets)

Model-Free:

+ Makes little assumptions beyond a reward function
+ Effective for learning complex policies
- Require a lot of experience (slower)
- Not transferable across tasks

Tutorial Outline

1. **RL Problem Set-up**
2. **Model-Free RL**
   a. policy gradients
   b. actor-critic algorithms
   c. value functions

less advanced topics

— BREAK —

3. **Soft optimality**
4. **Inverse RL**
5. **Model-Based RL**
6. **Frontiers & Open Challenges**

more advanced topics

Frontiers of Deep Learning for Decision Making

What is missing for these algorithms practical for real-world problems?
And, what is needed to reach to human-level decision making?

Open Problems:

How can our agents learn effectively using [much] less experience?

How do we ensure that our agents explore safely?

Where do rewards come from?

Next Slides: Initial approaches towards solving these problems

Frontiers: Reducing Sample Complexity 1

Scalar reward is an extremely sparse signal, while at the same time, humans can learn without any external rewards. How can we improve efficiency?

- better exploration and self-supervision (Osband et al. NIPS '16, Houthooft et al. NIPS '16, Pathak et al. ICML '17, Fu*, Co-Reyes* et al. '17, Tang et al. ICLR '17, Plappert et al. '17)
- options & hierarchy (Kulkarni et al. NIPS '16, Vezhnevets et al. NIPS '16, Bacon et al. AAAI '16, Heess et al. '17, Vezhnevets et al. ICML '17, Tessler et al. AAAI '17)
- leveraging stochastic policies for better exploration (Florensa et al. ICLR '17, Haarnoja et al. ICML '17)
- auxiliary objectives (Jaderberg et al. '17, Shelhamer et al. '17, Mirowski et al. ICLR '17)

Frontiers: Reducing Sample Complexity 2

Humans can't learn individual complex tasks from scratch.

Maybe our agents shouldn't either.

We ultimately want our agents to learn many tasks in many environments

- learn to learn new tasks quickly (Duan et al. '17, Wang et al. '17, Finn et al. ICML '17)
- share information across tasks in other ways (Rusu et al. NIPS '16, Andrychowicz et al. '17, Cabi et al. '17, Teh et al. '17)

Where should the tasks come from?

- automatic task/curriculum generation (Sukhbaatar et al. '17, Geng et al. '17, Matiisen et al. '17)

Frontiers: Ensuring Safety

How can we ensure that the agent explores safely?

- incorporating uncertainty (Kahn et al. ’17, Richter & Roy RSS ‘17)
- learning from off-policy data (Thomas ’17)
- human interventions (Zhang et al. AAAI ’17, Saunders et al. ‘17)
- introducing probabilistic constraints (Achiam et al. ICML ’17)
- train in simulation & transfer (Sadeghi & Levine RSS ’17, Tobin et al. ’17, You et al. ’17, James et al. ‘17)

Frontiers: Acquiring Rewards

How are ways in which we can get rewards from humans?

- learn from human pairwise preferences (Christiano et al. '17)
- cooperatively determine reward (Hadfield-Menell et al. NIPS '16)

Should we give the robot other forms of goals?

- inverse reinforcement learning (see previous slides for references)
- infer reward from video of human (Sermanet et al. RSS '17, Gupta et al. '17)
- provide goal image (Watter et al. NIPS '15, Finn et al. ICRA '16, Agrawal et al. NIPS '16)
- specify where objects should move in image (Finn & Levine '17)
- language instructions (Oh et al. '17, Andreas et al. ICML '17)

Closing Note

Most deep learning systems that are deployed today use supervised learning.

Thank you!

Slides online: sites.google.com/view/icml17deeprl

Further Resources:

Berkeley Deep RL course: rll.berkeley.edu/deeprlcourse

CMU Deep RL course: katefvision.github.io

Stanford RL course: web.stanford.edu/class/cs234/index.html

David Silver's RL course: www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html