Inverse Reinforcement Learning

CS 294-112: Deep Reinforcement Learning

Sergey Levine

Today's Lecture

1. So far: manually design reward function to define a task
2. What if we want to *learn* the reward function from observing an expert, and then use reinforcement learning?
3. Apply approximate optimality model from last week, but now learn the reward!

- Goals:
  - Understand the inverse reinforcement learning problem definition
  - Understand how probabilistic models of behavior can be used to derive inverse reinforcement learning algorithms
  - Understand a few practical inverse reinforcement learning algorithms we can use

Where does the reward function come from?

Computer Games

reward

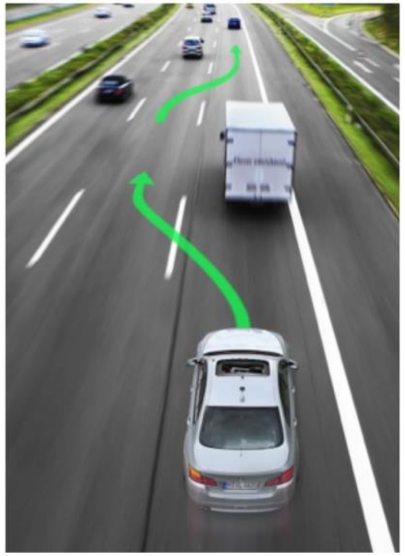
Mnih et al. '15

Real World Scenarios

robotics

dialog

autonomous driving



what is the reward?

often use a proxy

frequently easier to provide expert data

Inverse reinforcement learning: infer reward function from roll-outs of expert policy

Why should we learn the reward?

Alternative: directly mimic the expert (behavior cloning)
- simply “ape” the expert’s motions/actions
- doesn’t necessarily capture the *salient* parts of the behavior
- what if the expert has different capabilities?

Can we reason about *what* the expert is trying to achieve instead?

slides adapted from C. Finn

Inverse Optimal Control / Inverse Reinforcement Learning:

infer reward function from demonstrations

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

given:
- state & action space
- samples from π^*
- dynamics model (sometimes)

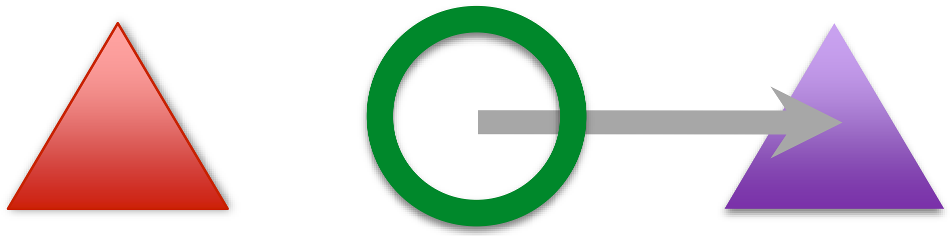
goal:
- recover reward function
- then use reward to get policy

Challenges

underdefined problem

difficult to evaluate a learned reward

demonstrations may not be precisely optimal



slides adapted from C. Finn

A bit more formally

"forward" reinforcement learning

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

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

learn $\pi^\star(\mathbf{a}|\mathbf{s})$

inverse reinforcement learning

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

samples $\{\tau_i\}$ sampled from $\pi^\star(\tau)$

learn $r_\psi(\mathbf{s}, \mathbf{a})$

reward parameters

...and then use it to learn $\pi^\star(\mathbf{a}|\mathbf{s})$

---

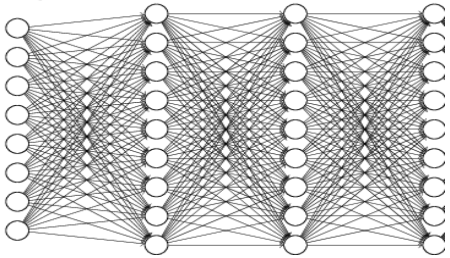
linear reward function:

$r_\psi(\mathbf{s}, \mathbf{a}) = \sum_i \psi_i f_i(\mathbf{s}, \mathbf{a}) = \psi^T \mathbf{f}(\mathbf{s}, \mathbf{a})$

neural net reward function:

$\mathbf{s}$

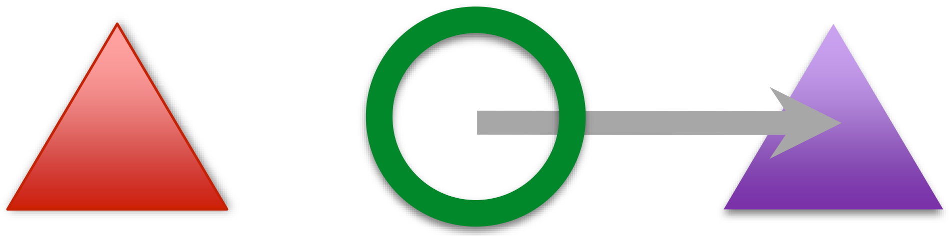
$\mathbf{a}$



$r_\psi(\mathbf{s}, \mathbf{a})$

parameters ψ

Feature matching IRL



linear reward function:

$$r_\psi(\mathbf{s},\mathbf{a}) = \sum_i \psi_i f_i(\mathbf{s},\mathbf{a}) = \psi^T \mathbf{f}(\mathbf{s},\mathbf{a})$$

if features $\mathbf{f}$ are important, what if we match their expectations?

let π^{r_ψ} be the optimal policy for r_ψ

pick ψ such that $E_{\pi^{r_\psi}}[\mathbf{f}(\mathbf{s},\mathbf{a})] = E_{\pi^\star}[\mathbf{f}(\mathbf{s},\mathbf{a})]$

still ambiguous!

state-action marginal under π^{r_ψ}

unknown optimal policy
approximate using expert samples

maximum margin principle:

$$\max_{\psi,m} m \qquad \text{such that } \psi^T E_{\pi^\star}[\mathbf{f}(\mathbf{s},\mathbf{a})] \geq \underbrace{\max_{\pi\in\Pi} \psi^T E_\pi[\mathbf{f}(\mathbf{s},\mathbf{a})] + m}$$

need to somehow "weight" by similarity between $\pi^\star$ and π

Feature matching IRL & maximum margin

remember the "SVM trick":

$$\max_{\psi,m} m \qquad \text{such that } \psi^T E_{\pi^\star}[\mathbf{f}(\mathbf{s},\mathbf{a})] \geq \max_{\pi\in\Pi} \psi^T E_\pi[\mathbf{f}(\mathbf{s},\mathbf{a})] + m$$

$$\Updownarrow$$

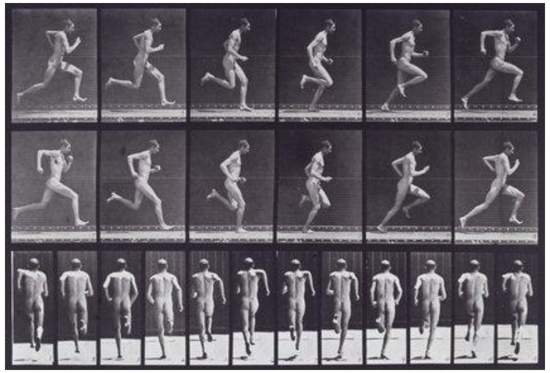
$$\min_{\psi} \frac{1}{2}\|\psi\|^2 \qquad \text{such that } \psi^T E_{\pi^\star}[\mathbf{f}(\mathbf{s},\mathbf{a})] \geq \max_{\pi\in\Pi} \psi^T E_\pi[\mathbf{f}(\mathbf{s},\mathbf{a})] + D(\pi,\pi^\star)$$

e.g., difference in feature expectations!

Issues:
- Maximizing the margin is a bit arbitrary
- No clear model of expert suboptimality (can add slack variables…)
- Messy constrained optimization problem – not great for deep learning!

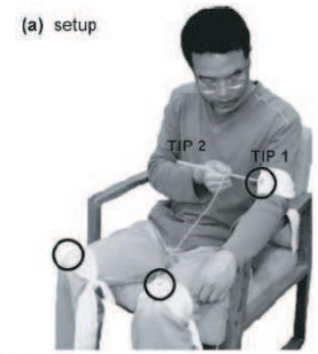
Further reading:
- Abbeel & Ng: Apprenticeship learning via inverse reinforcement learning
- Ratliff et al: Maximum margin planning

Optimal Control as a Model of Human Behavior



Muybridge (c. 1870)

Mombaur et al. ‘09

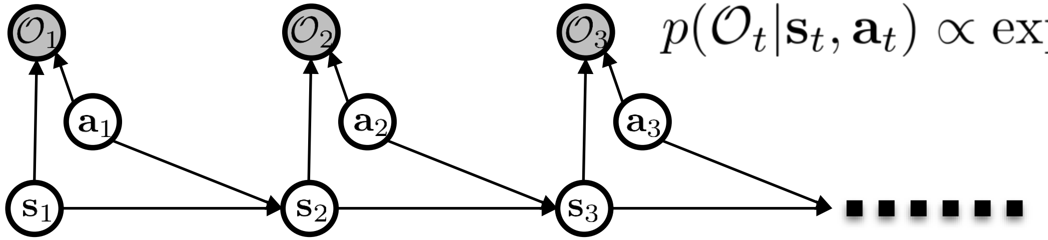


Li & Todorov ‘06

Ziebart ‘08

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

$p(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}|\mathcal{O}_{1:T})$



$p(\mathbf{s}_1)$

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

A probabilistic graphical model of decision making

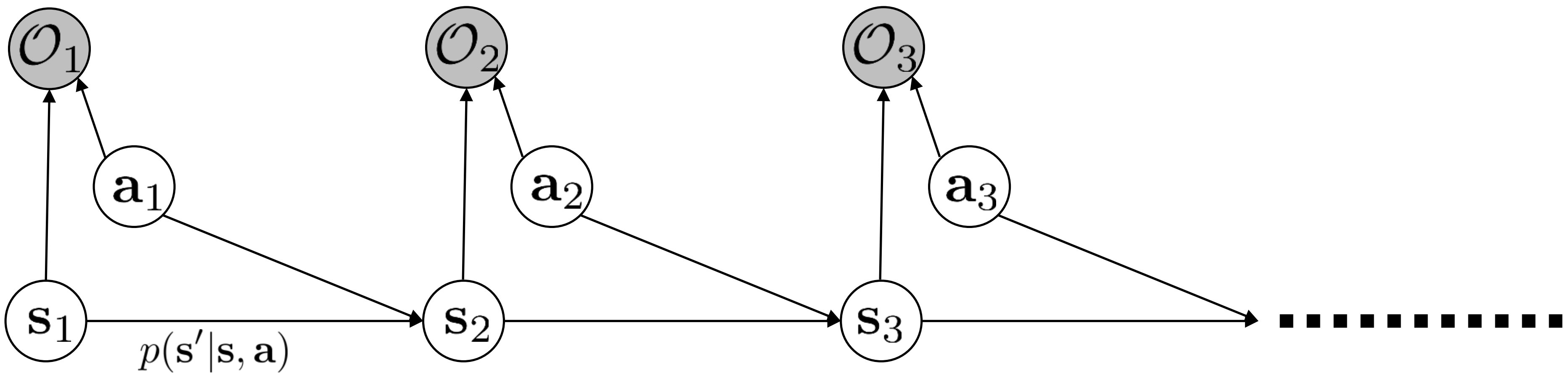
$$p(\underbrace{\mathbf{s}_{1:T}, \mathbf{a}_{1:T}}_{\tau}) = ??$$

no assumption of optimal behavior!

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

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

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



Learning the optimality variable

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

reward parameters

given:

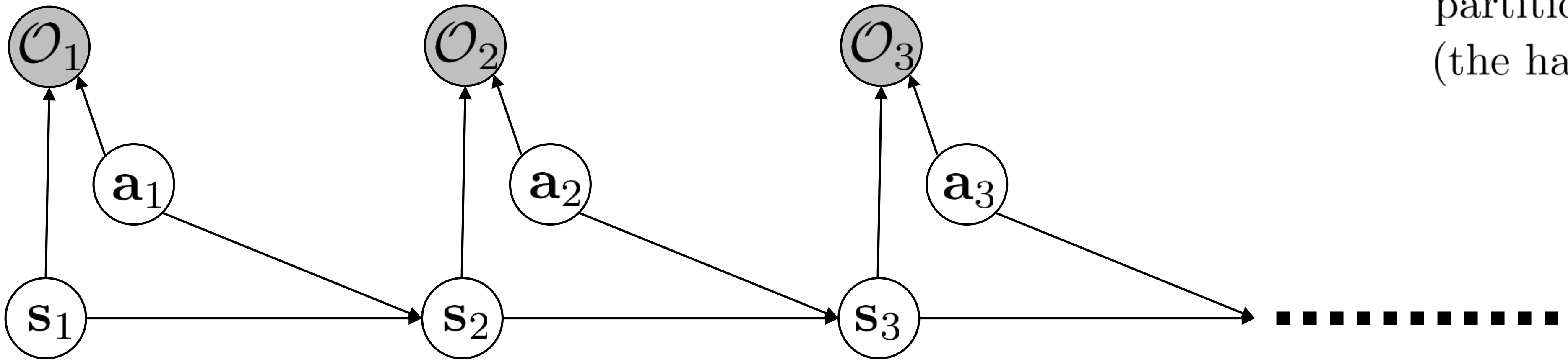
samples $\{\tau_i\}$ sampled from $\pi^\star(\tau)$

$$p(\tau|\mathcal{O}_{1:T},\psi) \propto \cancel{p(\tau)} \exp\left(\sum_t r_\psi(\mathbf{s}_t,\mathbf{a}_t)\right)$$

can ignore (independent of ψ)

maximum likelihood learning: $$\max_\psi \frac{1}{N}\sum_{i=1}^{N} \log p(\tau_i|\mathcal{O}_{1:T},\psi) = \max_\psi \frac{1}{N}\sum_{i=1}^{N} r_\psi(\tau_i) - \log Z$$

partition function
(the hard part)



The IRL partition function

$$\max_{\psi} \frac{1}{N} \sum_{i=1}^{N} r_\psi(\tau_i) - \log Z \qquad Z = \int p(\tau) \exp(r_\psi(\tau)) d\tau$$

$$\nabla_\psi \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \underbrace{\frac{1}{Z} \int p(\tau) \exp(r_\psi(\tau))}_{p(\tau|\mathcal{O}_{1:T}, \psi)} \nabla_\psi r_\psi(\tau) d\tau$$

$$\nabla_\psi \mathcal{L} = E_{\tau \sim \pi^\star(\tau)}[\nabla_\psi r_\psi(\tau_i)] - E_{\tau \sim p(\tau|\mathcal{O}_{1:T}, \psi)}[\nabla_\psi r_\psi(\tau)]$$

estimate with expert samples

soft optimal policy under current reward

Estimating the expectation

$$\nabla_\psi \mathcal{L} = E_{\tau \sim \pi^\star(\tau)}[\nabla_\psi r_\psi(\tau_i)] - \underbrace{E_{\tau \sim p(\tau|\mathcal{O}_{1:T},\psi)}[\nabla_\psi r_\psi(\tau)]}$$

$$E_{\tau \sim p(\tau|\mathcal{O}_{1:T},\psi)}\left[\nabla_\psi \sum_{t=1}^{T} r_\psi(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$= \sum_{t=1}^{T} E_{\underbrace{(\mathbf{s}_t,\mathbf{a}_t) \sim p(\mathbf{s}_t,\mathbf{a}_t|\mathcal{O}_{1:T},\psi)}}[\nabla_\psi r_\psi(\mathbf{s}_t, \mathbf{a}_t)]$$

$$p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{1:T}, \psi)p(\mathbf{s}_t|\mathcal{O}_{1:T}, \psi)$$

where have we seen this before?

$$= \frac{\beta(\mathbf{s}_t, \mathbf{a}_t)}{\beta(\mathbf{s}_t)} \qquad \propto \alpha(\mathbf{s}_t)\beta(\mathbf{s}_t)$$

$$p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{1:T}, \psi)p(\mathbf{s}_t|\mathcal{O}_{1:T}, \psi) \propto \beta(\mathbf{s}_t, \mathbf{a}_t)\alpha(\mathbf{s}_t)$$

backward message forward message

Estimating the expectation

$$\nabla_\psi \mathcal{L} = E_{\tau \sim \pi^\star(\tau)}[\nabla_\psi r_\psi(\tau_i)] - \underbrace{E_{\tau \sim p(\tau | \mathcal{O}_{1:T}, \psi)}[\nabla_\psi r_\psi(\tau)]}$$

let $\mu_t(\mathbf{s}_t, \mathbf{a}_t) \propto \beta(\mathbf{s}_t, \mathbf{a}_t)\alpha(\mathbf{s}_t)$

$$\sum_{t=1}^{T} \int \int \mu_t(\mathbf{s}_t, \mathbf{a}_t) \nabla_\psi r_\psi(\mathbf{s}_t, \mathbf{a}_t) d\mathbf{s}_t d\mathbf{a}_t$$

$$= \sum_{t=1}^{T} \vec{\mu}_t^T \nabla_\psi \vec{r}_\psi$$

state-action visitation probability for each $(\mathbf{s}_t, \mathbf{a}_t)$

The MaxEnt IRL algorithm

1. Given ψ, compute backward message $\beta(\mathbf{s}_t, \mathbf{a}_t)$ (see previous lecture)
2. Given ψ, compute forward message $\alpha(\mathbf{s}_t)$ (see previous lecture)
3. Compute $\mu_t(\mathbf{s}_t, \mathbf{a}_t) \propto \beta(\mathbf{s}_t, \mathbf{a}_t)\alpha(\mathbf{s}_t)$
4. Evaluate $\nabla_\psi \mathcal{L} = \frac{1}{N}\sum_{i=1}^N \sum_{t=1}^T \nabla_\psi r_\psi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) - \sum_{t=1}^T \int\int \mu_t(\mathbf{s}_t, \mathbf{a}_t)\nabla_\psi r_\psi(\mathbf{s}_t, \mathbf{a}_t) d\mathbf{s}_t d\mathbf{a}_t$
5. $\psi \leftarrow \psi + \eta \nabla_\psi \mathcal{L}$

Why MaxEnt?

in the case where $r_\psi(\mathbf{s}_t, \mathbf{a}_t) = \psi^T \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t)$, we can show that it optimizes

$$\max_\psi \mathcal{H}(\pi^{r_\psi}) \text{ such that } E_{\pi^{r_\psi}}[\mathbf{f}] = E_{\pi^\star}[\mathbf{f}]$$

optimal max-ent policy under r^ψ

unknown expert policy estimated with samples

as random as possible while matching features

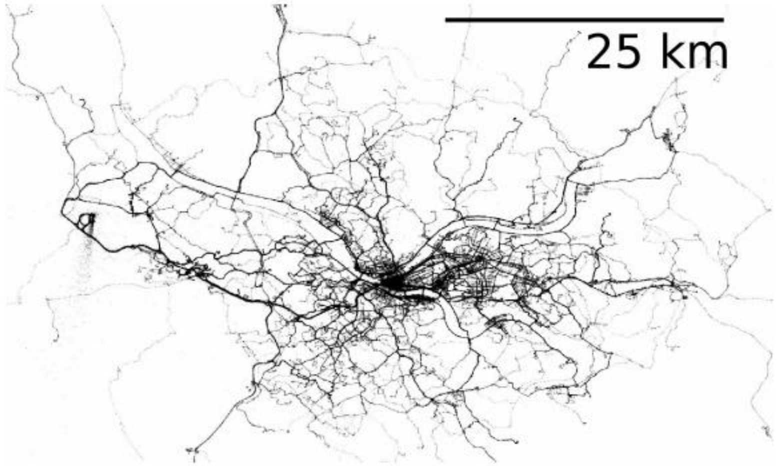
Ziebart et al. 2008: Maximum Entropy Inverse Reinforcement Learning

Case Study: MaxEnt IRL for road navigation

MaxEnt IRL with hand-designed features for learning to navigate in urban environments based on taxi cab GPS data.

Maximum Entropy Inverse Reinforcement Learning

Brian D. Ziebart, Andrew Maas, J.Andrew Bagnell, and **Anind K. Dey**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bziebart@cs.cmu.edu, amaas@andrew.cmu.edu, dbagnell@ri.cmu.edu, anind@cs.cmu.edu



Feature	Value
Highway	3.3 miles
Major Streets	2.0 miles
Local Streets	0.3 miles
Above 55mph	4.0 miles
35-54mph	1.1 miles
25-34 mph	0.5 miles
Below 24mph	0 miles
3+ Lanes	0.5 miles
2 Lanes	3.3 miles
1 Lane	1.8 miles

Feature	Value
Hard left turn	1
Soft left turn	3
Soft right turn	5
Hard right turn	0
No turn	25
U-turn	0

Break

What about larger RL problems?

- MaxEnt IRL: probabilistic framework for learning reward functions
- Computing gradient requires enumerating state-action visitations for all states and actions
  - Only really viable for small, discrete state and action spaces
  - Amounts to a dynamic programming algorithm (exact forward-backward inference)
- For deep IRL, we want two things:
  - Large and continuous state and action spaces
  - Effective learning under unknown dynamics

Unknown dynamics & large state/action spaces

Assume we don't know the dynamics, but we can sample, like in standard RL

recall:

$$\nabla_\psi \mathcal{L} = E_{\tau \sim \pi^\star(\tau)}[\nabla_\psi r_\psi(\tau_i)] - E_{\tau \sim p(\tau|\mathcal{O}_{1:T},\psi)}[\nabla_\psi r_\psi(\tau)]$$

estimate with expert samples   soft optimal policy under current reward

idea: learn $p(\mathbf{a}_t|\mathbf{s}_t, \mathcal{O}_{1:T}, \psi)$ using any max-ent RL algorithm
then run this policy to sample $\{\tau_j\}$

$$J(\theta) = \sum_t E_{\pi(\mathbf{s}_t,\mathbf{a}_t)}[r_\psi(\mathbf{s}_t,\mathbf{a}_t)] + E_{\pi(\mathbf{s}_t)}[\mathcal{H}(\pi(\mathbf{a}|\mathbf{s}_t))]$$

$$\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \frac{1}{M}\sum_{j=1}^{M} \nabla_\psi r_\psi(\tau_j)$$

sum over expert samples   sum over policy samples

More efficient sample-based updates

$$\nabla_\psi \mathcal{L} \approx \underbrace{\frac{1}{N} \sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i)}_{\text{sum over expert samples}} - \underbrace{\frac{1}{M} \sum_{j=1}^{M} \nabla_\psi r_\psi(\tau_j)}_{\text{sum over policy samples}}$$

improve (a little) ~~learn~~ $p(\mathbf{a}_t | \mathbf{s}_t, \mathcal{O}_{1:T}, \psi)$ using any max-ent RL algorithm

then run this policy to sample $\{\tau_j\}$

looks expensive! what if we use "lazy" policy optimization?

problem: estimator is now biased! wrong distribution!

solution: use importance sampling

$$\nabla_\psi \mathcal{L} \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j} \sum_{j=1}^{M} w_j \nabla_\psi r_\psi(\tau_j) \qquad w_j = \frac{\exp(r_\psi(\tau_j))}{\pi(\tau_j)}$$

Importance sampling

$$\nabla_\psi \mathcal{L} \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j}\sum_{j=1}^{M} w_j \nabla_\psi r_\psi(\tau_j)$$

$$w_j = \frac{\exp(r_\psi(\tau_j))}{\pi(\tau_j)}$$

$$\frac{\cancel{p(\mathbf{s}_1)}\prod_t \cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}\exp(r_\psi(\mathbf{s}_t,\mathbf{a}_t))}{\cancel{p(\mathbf{s}_1)}\prod_t \cancel{p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}\pi(\mathbf{a}_t|\mathbf{s}_t)}$$

$$= \frac{\exp(\sum_t r_\psi(\mathbf{s}_t,\mathbf{a}_t))}{\prod_t \pi(\mathbf{a}_t|\mathbf{s}_t)}$$

which sampling distribution $\pi(\tau)$ is best?

optimal IS distribution $q(x)$ for $E_{p(x)}[f(x)]$ is $q(x) \propto |f(x)|p(x)$

in our case, optimal π is therefore $\pi(\tau) \propto \exp(r_\psi(\tau))$

max-ent optimal policy for r_ψ

each policy update w.r.t. r_ψ brings us closer to the optimal distribution!

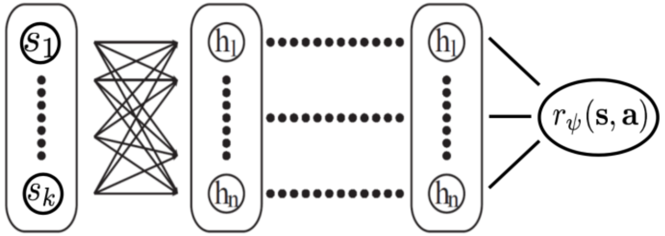
guided cost learning algorithm

(Finn et al. ICML '16)

initial policy π

human demonstrations

generate policy samples from π



Update reward using samples & demos

update π w.r.t. reward

policy π

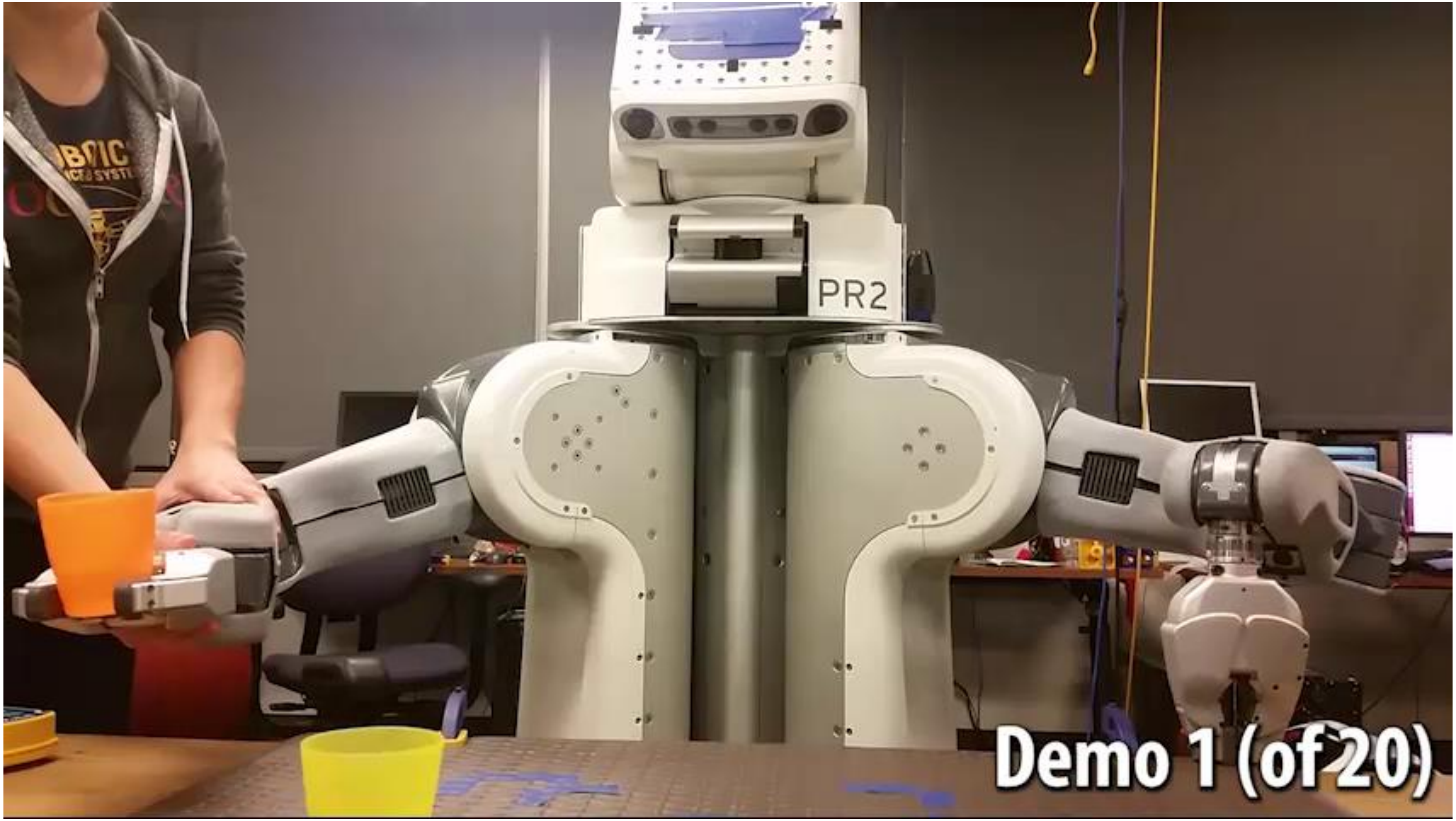
reward r

$$\nabla_\psi \mathcal{L} \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j} \sum_{j=1}^{M} w_j \nabla_\psi r_\psi(\tau_j)$$

$$w_j = \frac{\exp(r_\psi(\tau_j))}{\pi(\tau_j)}$$

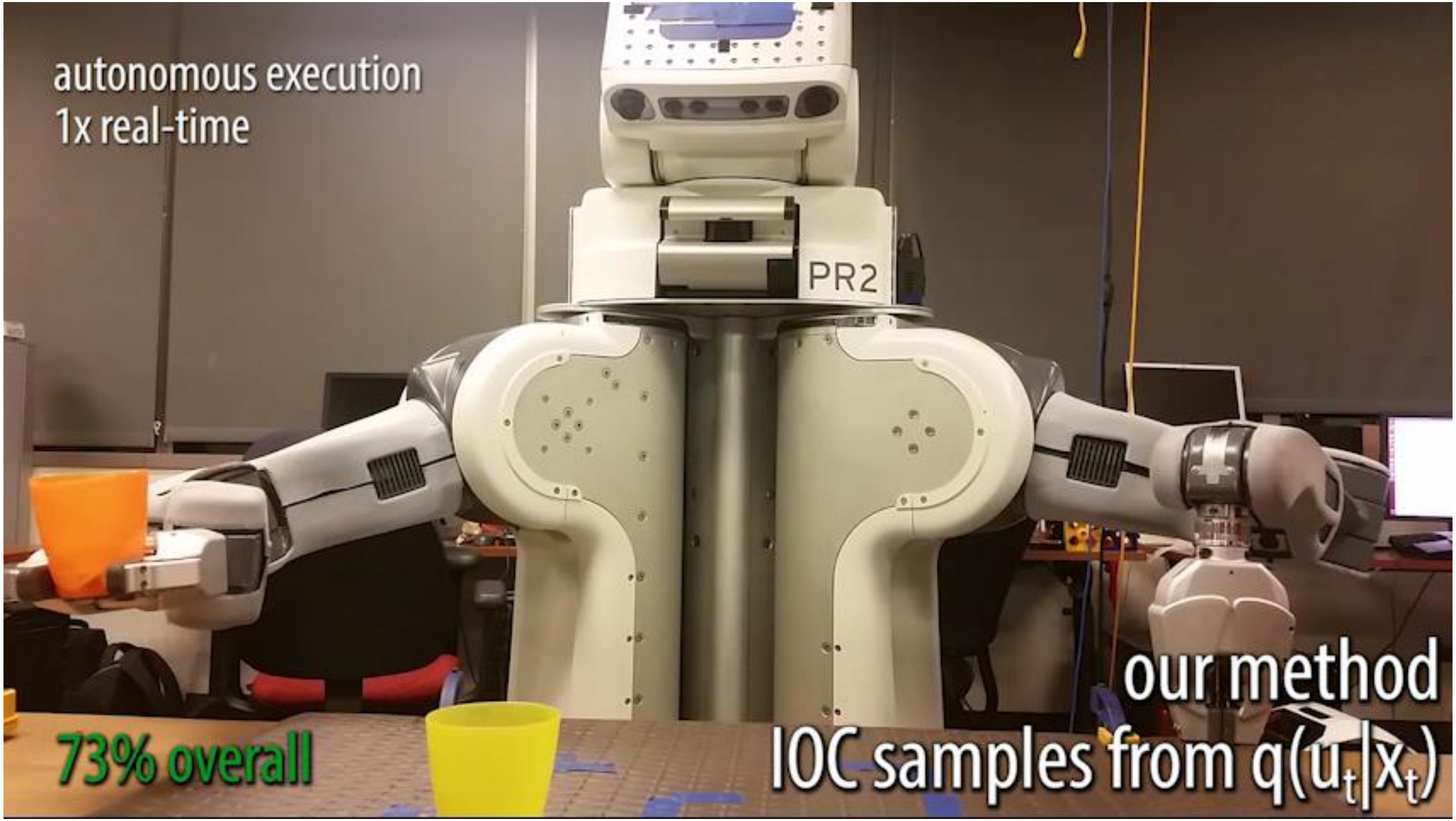
slides adapted from C. Finn

Example: learning pouring with a robot



Finn et al. Guided cost learning.

Example: learning pouring with a robot



Finn et al. Guided cost learning.

It looks a bit like a game…

initial
policy π

samples from $\pi_\theta(\tau)$

human
demonstrations

samples from $\pi^\star(\tau)$

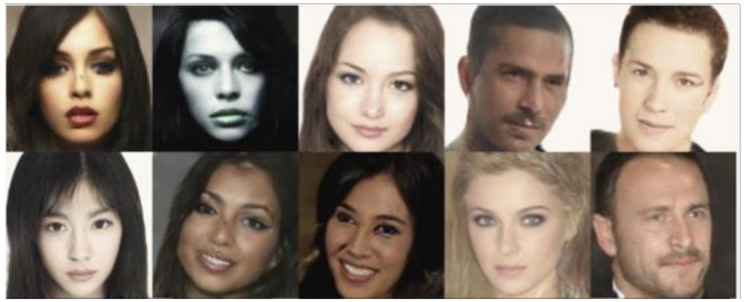
$$\nabla_\psi \mathcal{L} \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\psi r_\psi(\tau_i) - \frac{1}{\sum_j w_j} \sum_{j=1}^{M} w_j \nabla_\psi r_\psi(\tau_j)$$

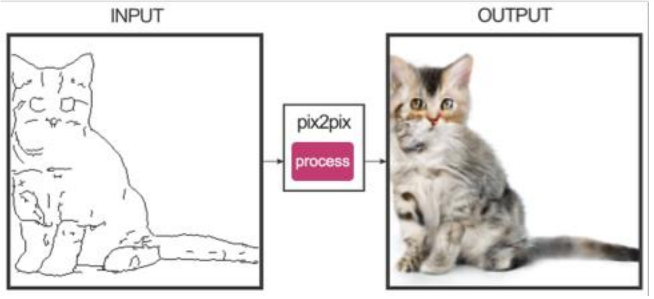
demos are made more likely, samples less likely

$$\nabla_\theta \mathcal{L} \approx \frac{1}{M} \sum_{j=1}^{M} \nabla_\theta \log \pi_\theta(\tau_j) r_\psi(\tau_j)$$

policy changed to make it *harder* to distinguish from demos

Generative Adversarial Networks





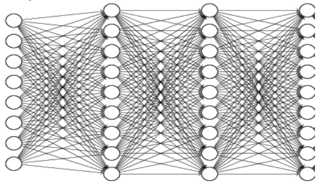
Zhu et al. '17

Arjovsky et al. '17

Isola et al. '17

"generator"

$\mathbf{z}$



$p_\theta(\mathbf{x}|\mathbf{z})$

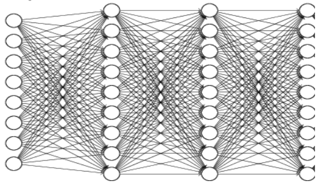
samples from $p_\theta(\mathbf{x})$

data ("demonstrations")

samples from $p^\star(\mathbf{x})$

$D(\mathbf{x}) = p_\psi(\text{real image}|\mathbf{x})$

$\mathbf{x}$



$$\theta \leftarrow \arg\max_\theta E_{\mathbf{x}\sim p_\theta} \log D_\psi(\mathbf{x})$$

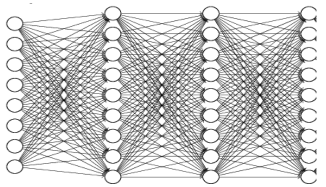
$$\psi = \arg\max_\psi \frac{1}{N} \sum_{\mathbf{x}\sim p^\star} \log D_\psi(\mathbf{x}) + \frac{1}{M} \sum_{\mathbf{x}\sim p_\theta} \log(1 - D_\psi(\mathbf{x}))$$

Goodfellow et al. '14

Inverse RL as a GAN

$$D(\mathbf{x}) = p_\psi(\text{real image}|\mathbf{x})$$

$\mathbf{x}$



which discriminator is best?

$$D^\star(\mathbf{x}) = \frac{p^\star(\mathbf{x})}{p_\theta(\mathbf{x}) + p^\star(\mathbf{x})}$$

for IRL, optimal policy approaches $\pi_\theta(\tau) \propto p(\tau)\exp(r_\psi(\tau))$

choose this parameterization for discriminator:

optimize Z w.r.t. same objective as ψ!

$$D_\psi(\tau) = \frac{p(\tau)\frac{1}{Z}\exp(r(\tau))}{p_\theta(\tau) + p(\tau)\frac{1}{Z}\exp(r(\tau))} = \frac{\cancel{p(\tau)}\frac{1}{Z}\exp(r(\tau))}{\cancel{p(\tau)}\prod_t \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \cancel{p(\tau)}\frac{1}{Z}\exp(r(\tau))} = \frac{\frac{1}{Z}\exp(r(\tau))}{\prod_t \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \frac{1}{Z}\exp(r(\tau))}$$

optimize this w.r.t. ψ

$$\psi \leftarrow \arg\max_\psi E_{\tau\sim p^\star}[\log D_\psi(\tau)] + E_{\tau\sim\pi_\theta}[\log(1 - D_\psi(\tau))]$$

we don't need importance weights anymore – they are subsumed into Z

Finn*, Christiano* et al. "A Connection Between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models."

Inverse RL as a GAN

generator/policy

$\pi_\theta(\tau)$

samples from $\pi_\theta(\tau)$

data/demonstrations

samples from $p^\star(\tau)$

$$\psi \leftarrow \arg\max_\psi E_{\tau\sim p^\star}[\log D_\psi(\tau)] + E_{\tau\sim\pi_\theta}[\log(1 - D_\psi(\tau))]$$

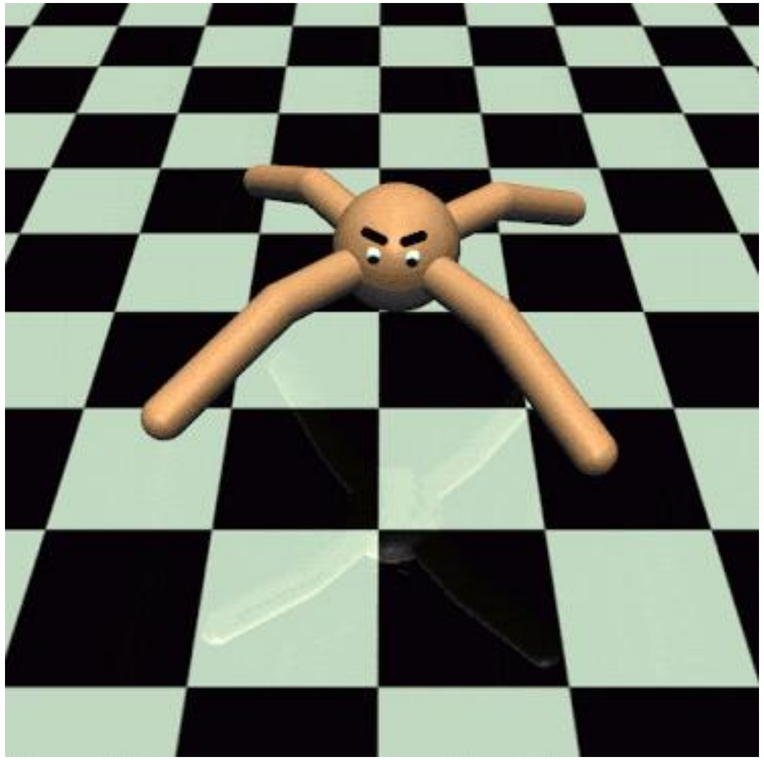
$$D_\psi(\tau) = \frac{\frac{1}{Z}\exp(r(\tau))}{\prod_t \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \frac{1}{Z}\exp(r(\tau))}$$

$$\nabla_\theta \mathcal{L} \approx \frac{1}{M}\sum_{j=1}^{M} \nabla_\theta \log \pi_\theta(\tau_j) r_\psi(\tau_j)$$

policy changed to make it *harder* to distinguish from demos

Finn*, Christiano* et al. “A Connection Between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models.”

Generalization via inverse RL

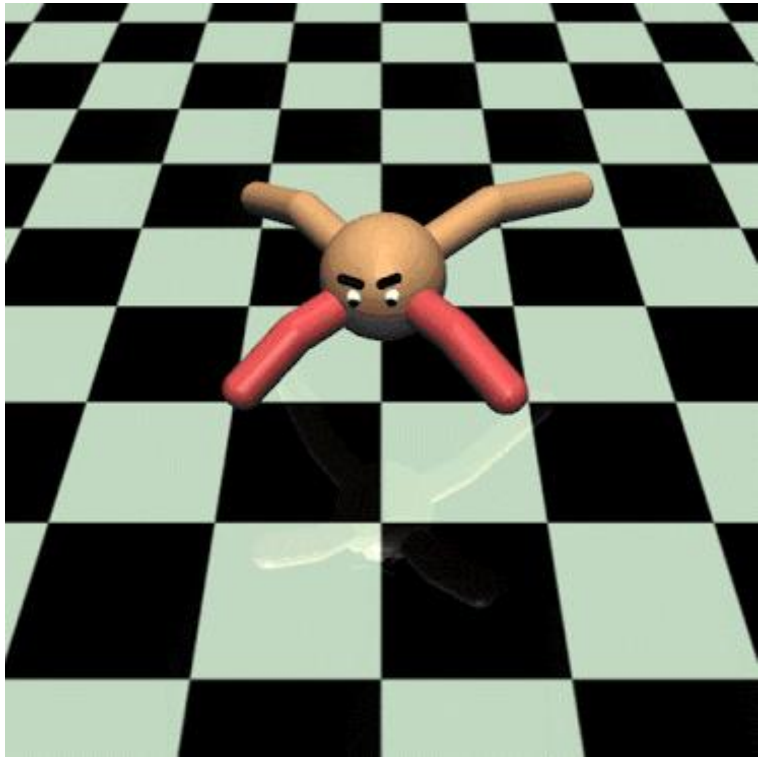


demonstration

what can we learn from the demonstration to enable better **transfer**?

need to decouple the **goal** from the **dynamics**!

policy = **reward** + **dynamics**



reproduce behavior under different conditions

Fu et al. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning

Can we just use a regular discriminator?

generator/policy

$\pi_\theta(\tau)$

samples from $\pi_\theta(\tau)$

data/demonstrations

samples from $p^\star(\tau)$

$$\psi \leftarrow \arg\max_\psi E_{\tau\sim p^\star}[\log D_\psi(\tau)] + E_{\tau\sim\pi_\theta}[\log(1 - D_\psi(\tau))]$$

$D_\psi(\tau) = \text{standard binary neural net classifier}$

$$\nabla_\theta \mathcal{L} \approx \frac{1}{M}\sum_{j=1}^{M} \nabla_\theta \log \pi_\theta(\tau_j) \log D_\psi(\tau_j)$$

policy changed to make it *harder* to distinguish from demos

Pros & cons:

\+ often simpler to set up optimization, fewer moving parts

\- discriminator knows *nothing* at convergence

\- generally cannot reoptimize the "reward"

Ho & Ermon. Generative adversarial imitation learning.

IRL as adversarial optimization

Guided Cost Learning

ICML 2016

minimized maximized

reward function

robot attempt


human demonstrations

learns distribution $p(\tau)$ such that demos have max likelihood

$p(\tau) \propto \exp(r(\tau))$ (MaxEnt model)

$$D(\tau) = \frac{\frac{1}{Z}\exp(r(\tau))}{\frac{1}{Z}\exp(r(\tau)) + \pi(\tau)}$$

actually the same thing!

Generative Adversarial Imitation Learning

Ho & Ermon, NIPS 2016

False True

classifier

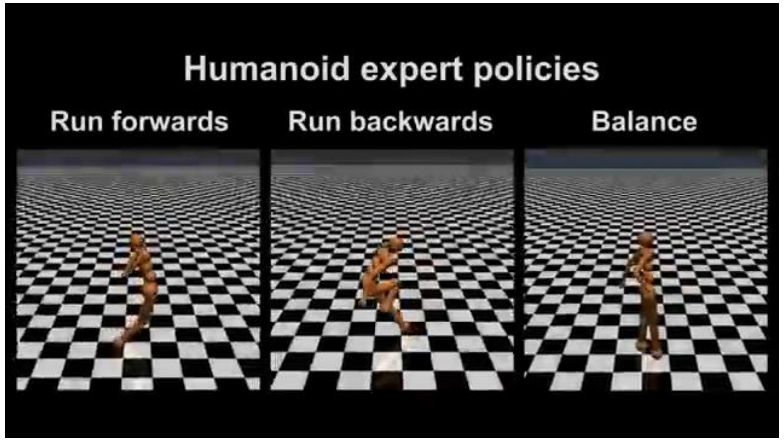
robot attempt


human demonstrations

$D(\tau) =$ probability τ is a demo

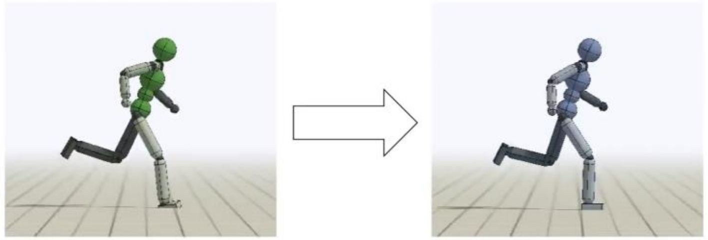
use $\log D(\tau)$ as "reward"

$D(\tau) =$ some classifier



Hausman, Chebotar, Schaal, Sukhatme, Lim

Motion Imitation



Reference Motion (Mocap)

Learned Policy (Simulation)

the goal is to train a simulated character to imitate the motion.

Peng, Kanazawa, Toyer, Abbeel, Levine

Review

- IRL: infer unknown reward from expert demonstrations
- MaxEnt IRL: infer reward by learning under the control-as-inference framework
- MaxEnt IRL with dynamic programming: simple and efficient, but requires small state space and known dynamics
- Sampling-based MaxEnt IRL: generate samples to estimate the partition function
  - Guided cost learning algorithm
  - Connection to generative adversarial networks
  - Generative adversarial imitation learning (not IRL per se, but similar)

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 to probabilistic method for inverse reinforcement learning

Modern Papers:
Finn et al. ICML '16. *Guided Cost Learning*. Sampling based method for MaxEnt IRL that handles unknown dynamics and deep reward functions
Wulfmeier et al. arXiv '16. *Deep Maximum Entropy Inverse Reinforcement Learning*. MaxEnt inverse RL using deep reward functions
Ho & Ermon NIPS '16. *Generative Adversarial Imitation Learning*. Inverse RL method using generative adversarial networks
Fu, Luo, Levine ICLR '18. Learning Robust Rewards with Adversarial Inverse Reinforcement Learning