Inverse Reinforcement Learning

CS 294-112: Deep Reinforcement Learning

Sergey Levine

Class Notes

1. Project proposal due today at 11:59 pm!
2. Homework 4 due next week
3. After that, just the project left!

Today’s Lecture

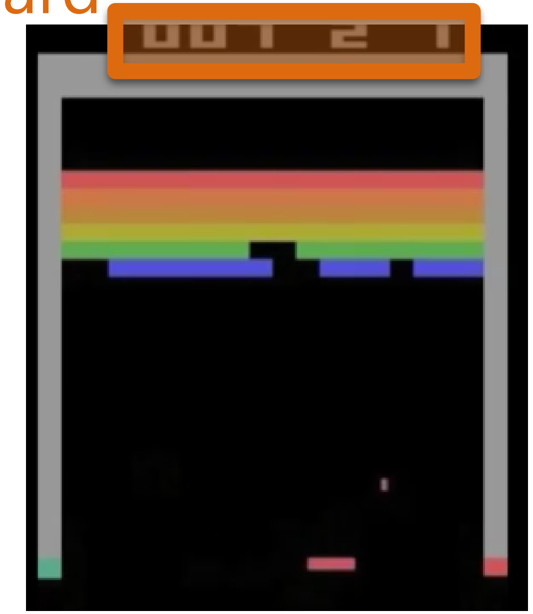
1. So far: manually design reward function to define a task
2. What if we want to *learn* the reward function from observing an expert, and then use reinforcement learning?
3. Apply approximate optimality model from last week, but now learn the reward!

- 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



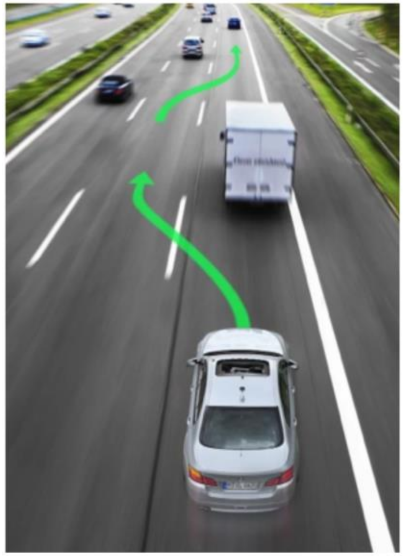
Mnih et al. '15

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

slides adapted from C. Finn

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)

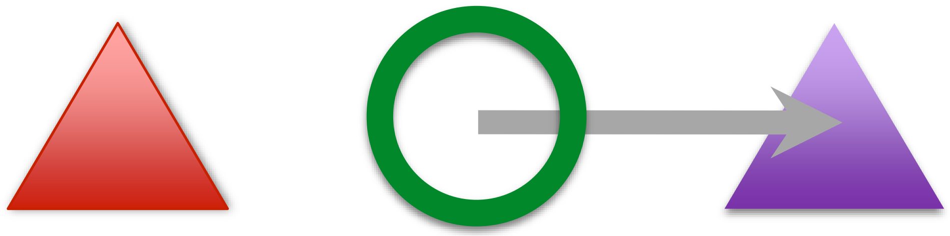
goal:
- recover reward function
- then use reward to get policy

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

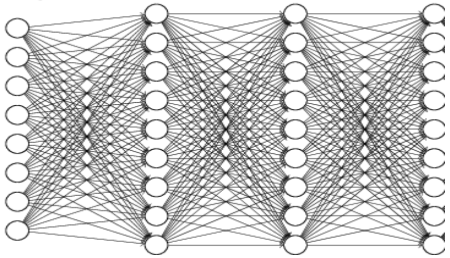
...and then use it to learn $\pi^\star(\mathbf{a}|\mathbf{s})$

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:

$\mathbf{s}$
$\mathbf{a}$

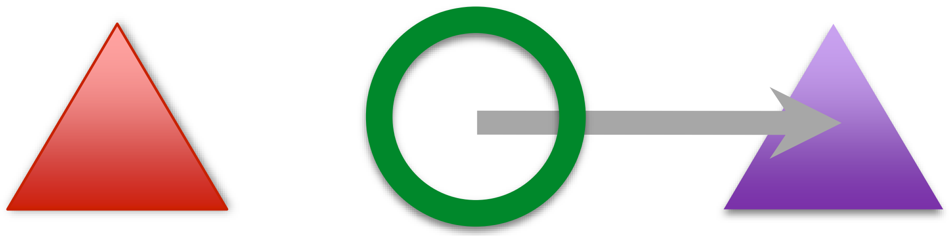


$r_\psi(\mathbf{s}, \mathbf{a})$
parameters ψ

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

state-action marginal under π^{r_ψ}

unknown optimal policy
approximate using expert samples

still ambiguous!

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":

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

$$\Updownarrow$$

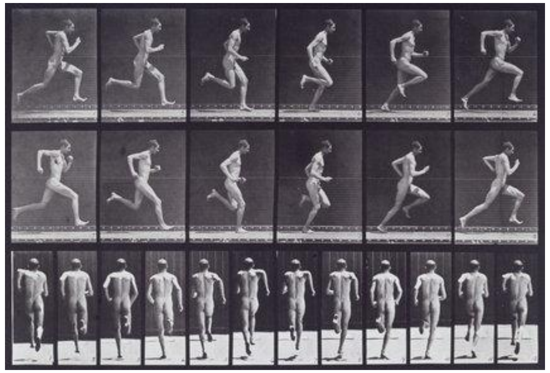
$$\min_{\psi} \frac{1}{2}\|\psi\|^2 \qquad \text{such that } \psi^T E_{\pi^\star}[\mathbf{f}(\mathbf{s},\mathbf{a})] \geq \max_{\pi \in \Pi} \psi^T E_\pi[\mathbf{f}(\mathbf{s},\mathbf{a})] + D(\pi, \pi^\star)$$

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!

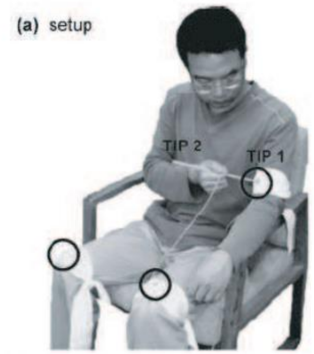
Further reading:
- Abbeel & Ng: Apprenticeship learning via inverse reinforcement learning
- Ratliff et al: Maximum margin planning

Optimal Control as a Model of Human Behavior



Muybridge (c. 1870)

Mombaur et al. ‘09



Li & Todorov ‘06

Ziebart ‘08

$$\mathbf{a}_1, \ldots, \mathbf{a}_T = \arg \max_{\mathbf{a}_1, \ldots, \mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)$$

$$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$$

optimize this to explain the data

$$\pi = \arg \max_{\pi} E_{\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t), \mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)}[r(\mathbf{s}_t, \mathbf{a}_t)]$$

$$\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$$

A probabilistic graphical model of decision making

$$\cancel{\mathbf{a}_1,\ldots,\mathbf{a}_T = \arg\max_{\mathbf{a}_1,\ldots,\mathbf{a}_T} \sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t)}$$

$$\cancel{\mathbf{s}_{t+1} = f(\mathbf{s}_t,\mathbf{a}_t)}$$

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

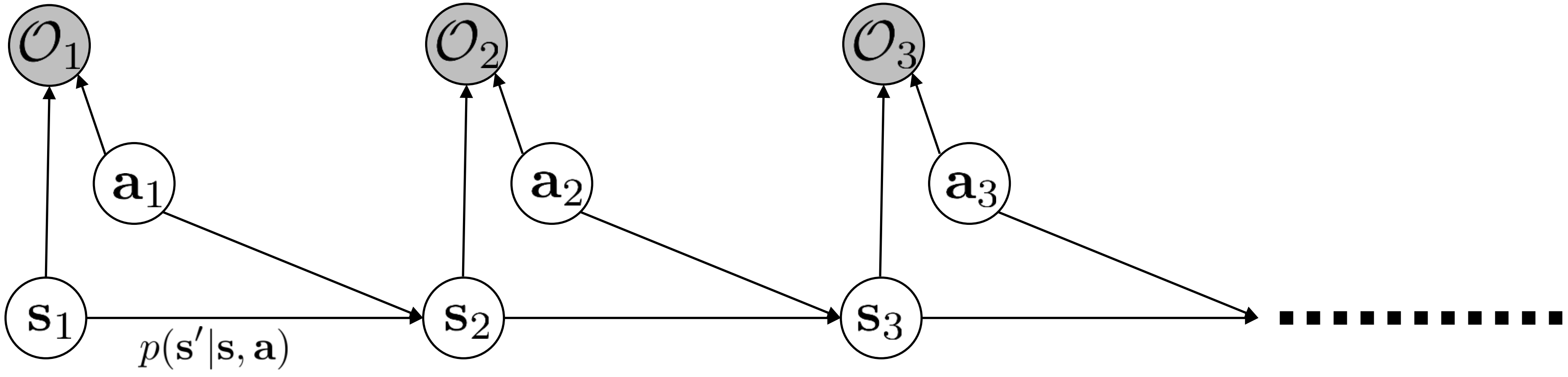
no assumption of optimal behavior!

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

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

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

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



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:

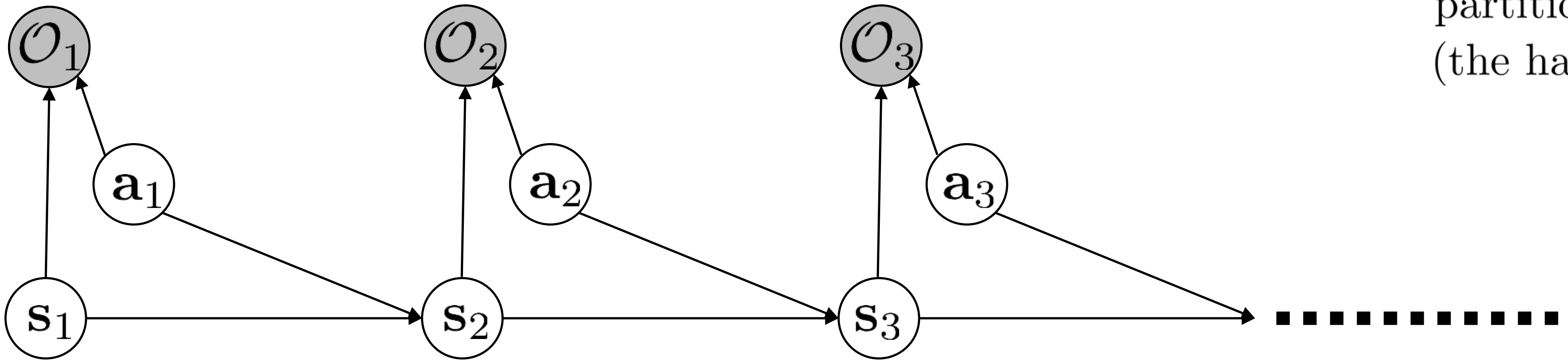
samples $\{\tau_i\}$ sampled from $\pi^\star(\tau)$

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

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

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

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

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

estimate with expert samples

soft optimal policy under current reward

Estimating the expectation

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

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

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

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

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

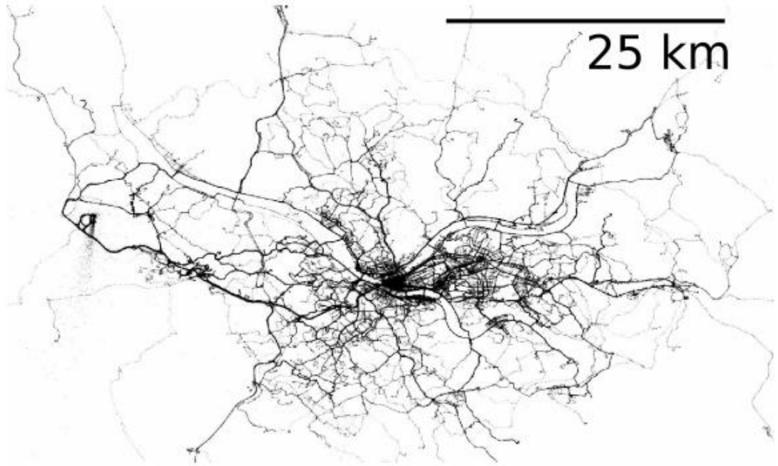
Ziebart et al. 2008: Maximum Entropy Inverse Reinforcement Learning

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

Case Study: MaxEnt Deep IRL

MaxEnt IRL with known dynamics (tabular setting), neural net cost

Maximum Entropy Deep Inverse Reinforcement Learning

Markus Wulfmeier MARKUS@ROBOTS.OX.AC.UK
Peter Ondrúška ONDRUSKA@ROBOTS.OX.AC.UK
Ingmar Posner INGMAR@ROBOTS.OX.AC.UK
Mobile Robotics Group, Department of Engineering Science, University of Oxford

NIPS Deep RL workshop 2015

Watch This: Scalable Cost-Function Learning for Path Planning in Urban Environments

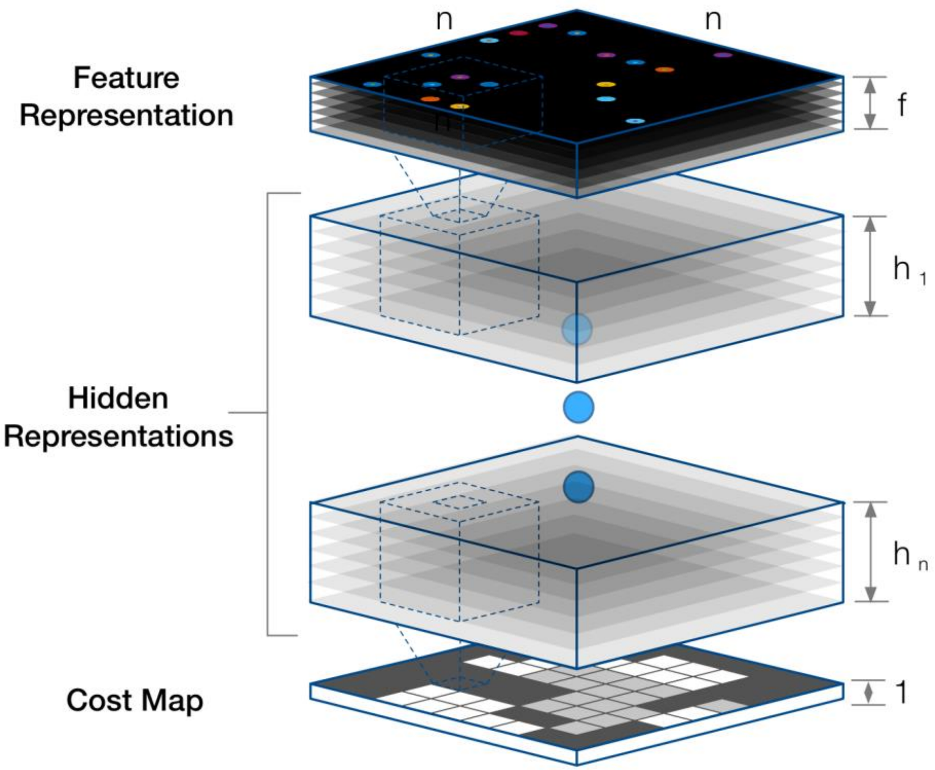
Markus Wulfmeier[1], Dominic Zeng Wang[1] and Ingmar Posner[1]

IROS 2016

slides adapted from C. Finn

Case Study: MaxEnt Deep IRL

MaxEnt IRL with known dynamics (tabular setting), neural net cost



Algorithm 1 Maximum Entropy Deep IRL

Input: $\mu_D^a, f, S, A, T, \gamma$

Output: optimal weights θ^*

$\theta^1 = \text{initialise\_weights}()$

Iterative model refinement

for n = 1 : N **do**

  $r^n = \text{nn\_forward}(f, \theta^n)$

  Solution of MDP with current reward

  $\pi^n = \text{approx\_value\_iteration}(r^n, S, A, T, \gamma)$

  $\mathbb{E}[\mu^n] = \text{propagate\_policy}(\pi^n, S, A, T)$

  Determine Maximum Entropy loss and gradients

  $\mathcal{L}_D^n = \log(\pi^n) \times \mu_D^a$

  $\frac{\partial \mathcal{L}_D^n}{\partial r^n} = \mu_D - \mathbb{E}[\mu^n]$

  Compute network gradients

  $\frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n} = \text{nn\_backprop}(f, \theta^n, \frac{\partial \mathcal{L}_D^n}{\partial r^n})$

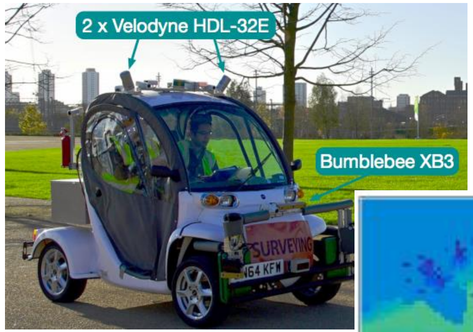
  $\theta^{n+1} = \text{update\_weights}(\theta^n, \frac{\partial \mathcal{L}_D^n}{\partial \theta_D^n})$

end for

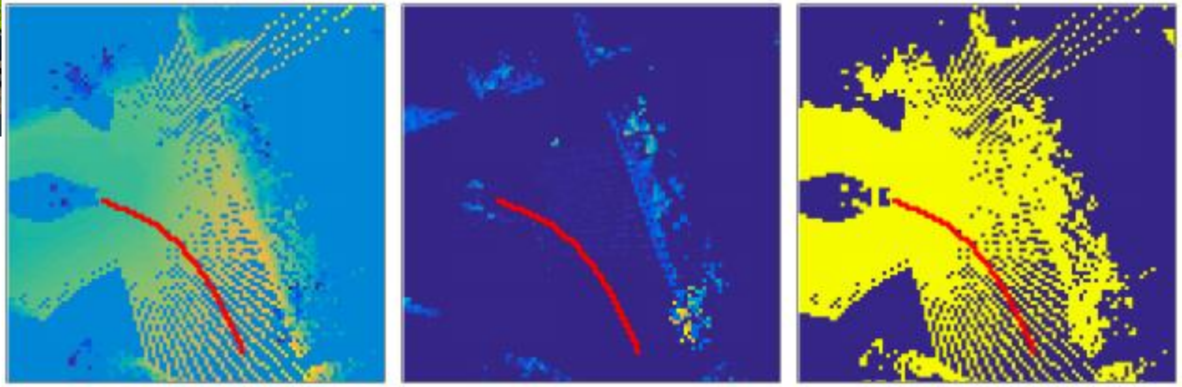
Need to iteratively solve MDP for every weight update

Case Study: MaxEnt Deep IRL

MaxEnt IRL with known dynamics (tabular setting), neural net cost



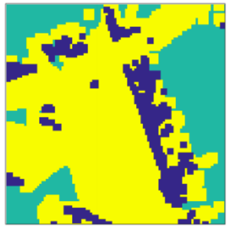
demonstrations



mean height | height variance | cell visibility

120km of demonstration data

manually designed cost:



test-set trajectory prediction:

Prediction metrics	Standard FCN	Pooling FCN	MS FCN	Manual CF
NLL	69.35	69.73	65.39	78.13
MHD	0.221	0.230	0.200	0.284

MHD: modified Hausdorff distance

slides adapted from C. Finn

Case Study: MaxEnt Deep IRL

MaxEnt IRL with known dynamics (tabular setting), neural net cost

Strengths

- scales to neural net costs

Limitations

- still need to repeatedly solve the MDP
- assumes known dynamics

slides adapted from C. Finn

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 $\{\tau_j\}$

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

sum over expert samples

sum over policy samples

More efficient sample-based updates

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

sum over expert samples   sum over policy samples

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

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

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

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

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!

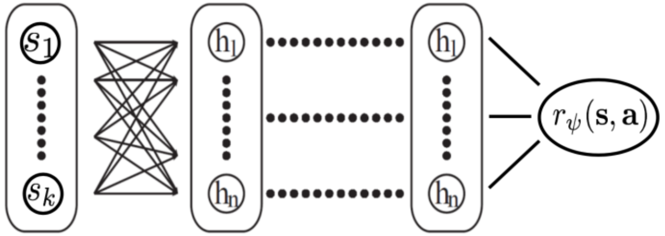
guided cost learning algorithm

(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

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

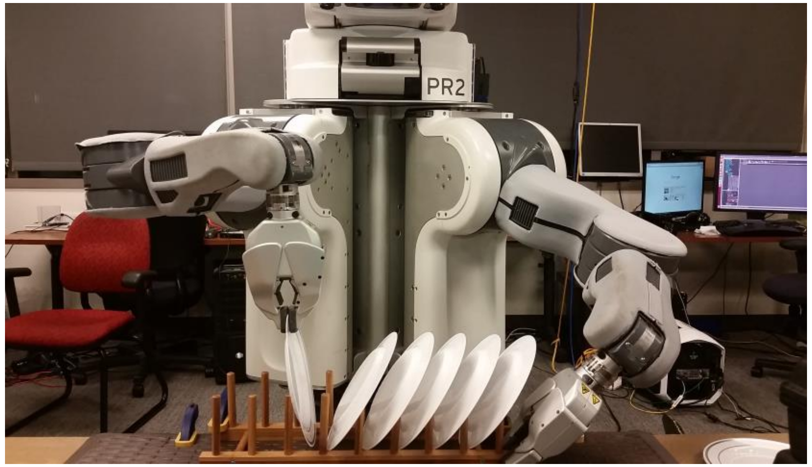
$$w_j = \frac{\exp(r_\psi(\tau_j))}{\pi(\tau_j)}$$

slides adapted from C. Finn

Guided Cost Learning Experiments

Real-world Tasks

dish placement



state includes goal plate pose

pouring almonds

state includes unsupervised visual features [Finn et al. ’16]

Comparison

Relative Entropy IRL
(Boularias et al. ’11)

slides adapted from C. Finn

Comparisons

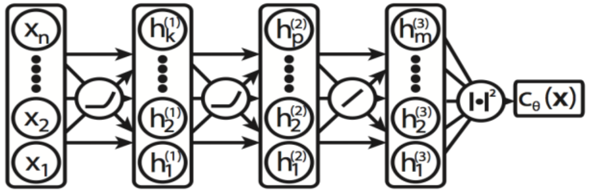
Path Integral IRL
(Kalakrishnan et al. '13)

Relative Entropy IRL
(Boularias et al. '11)

initial
distribution q_0

generate policy
samples from q

human
demonstrations

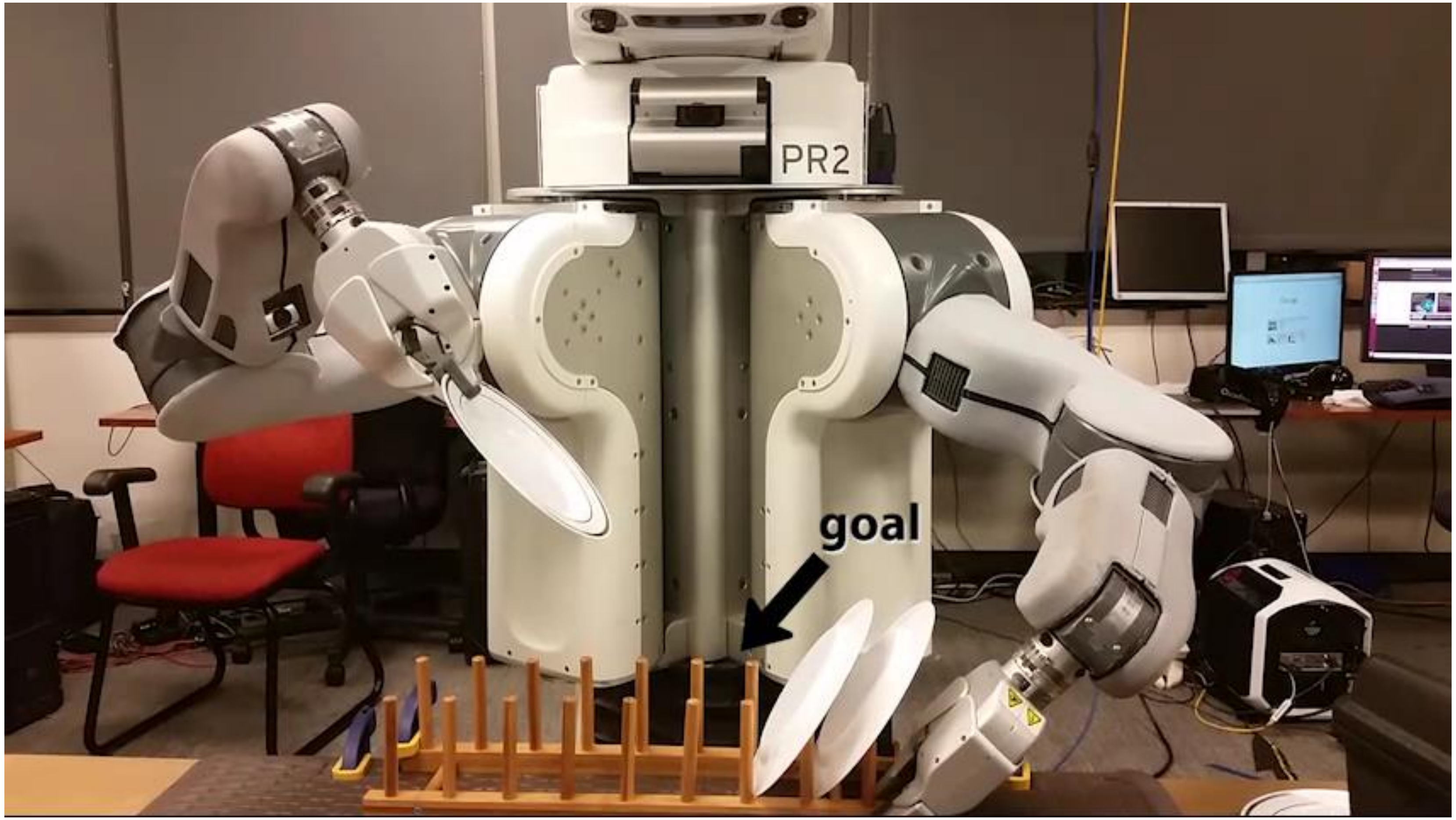


Update reward using
samples & demos

reward r

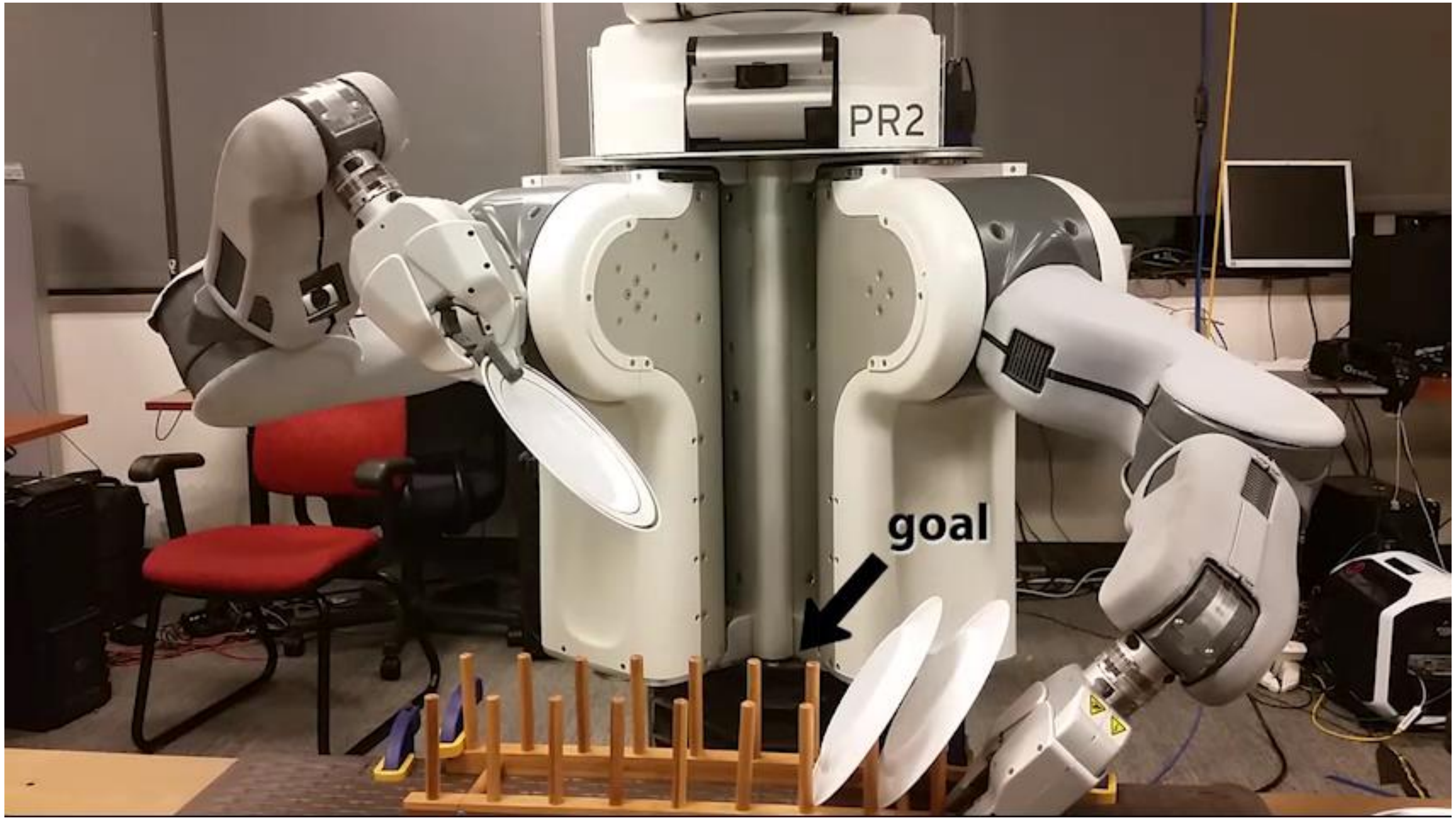
slides adapted from C. Finn

Dish placement, standard reward



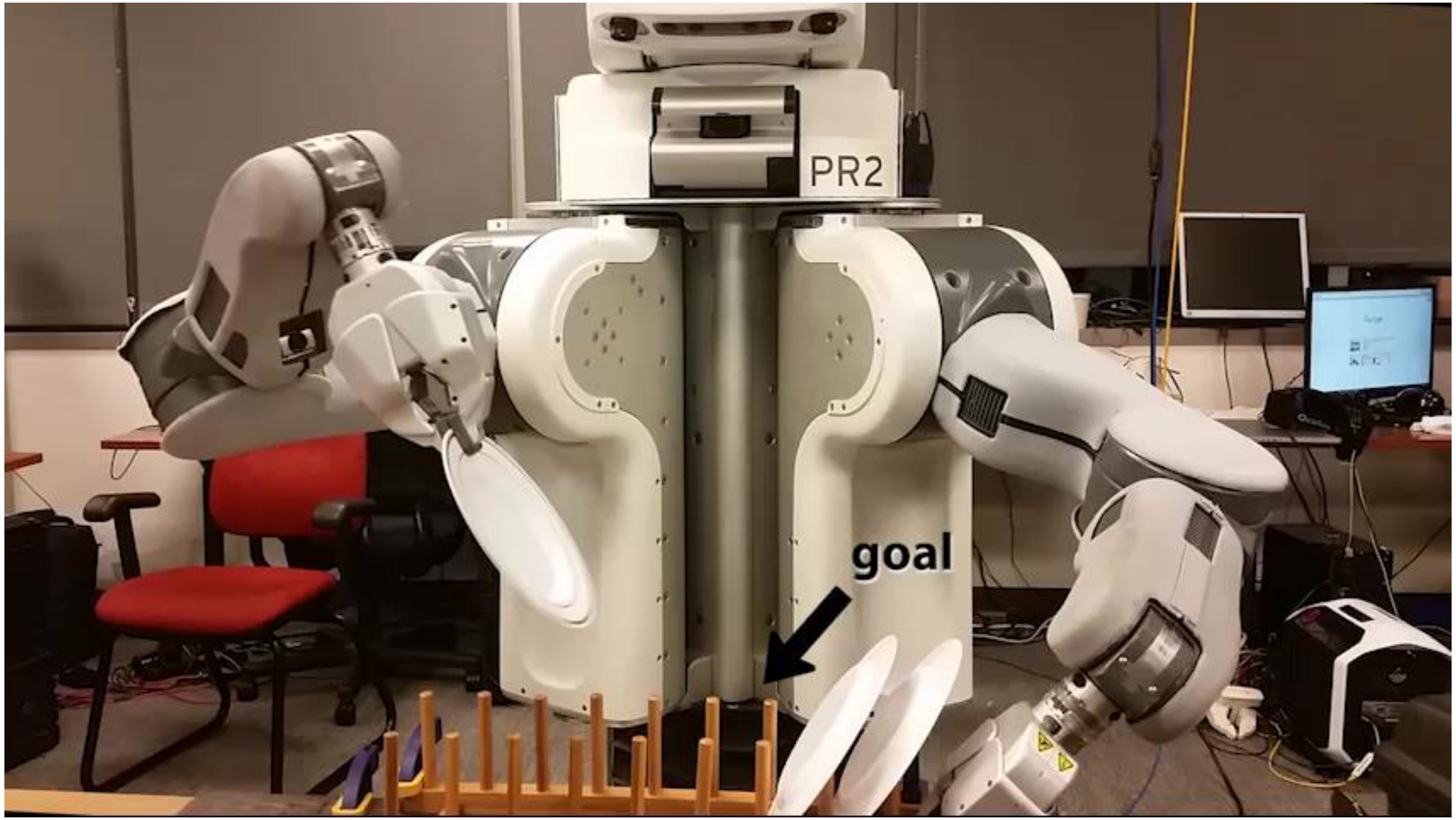
slides adapted from C. Finn

Dish placement, RelEnt IRL



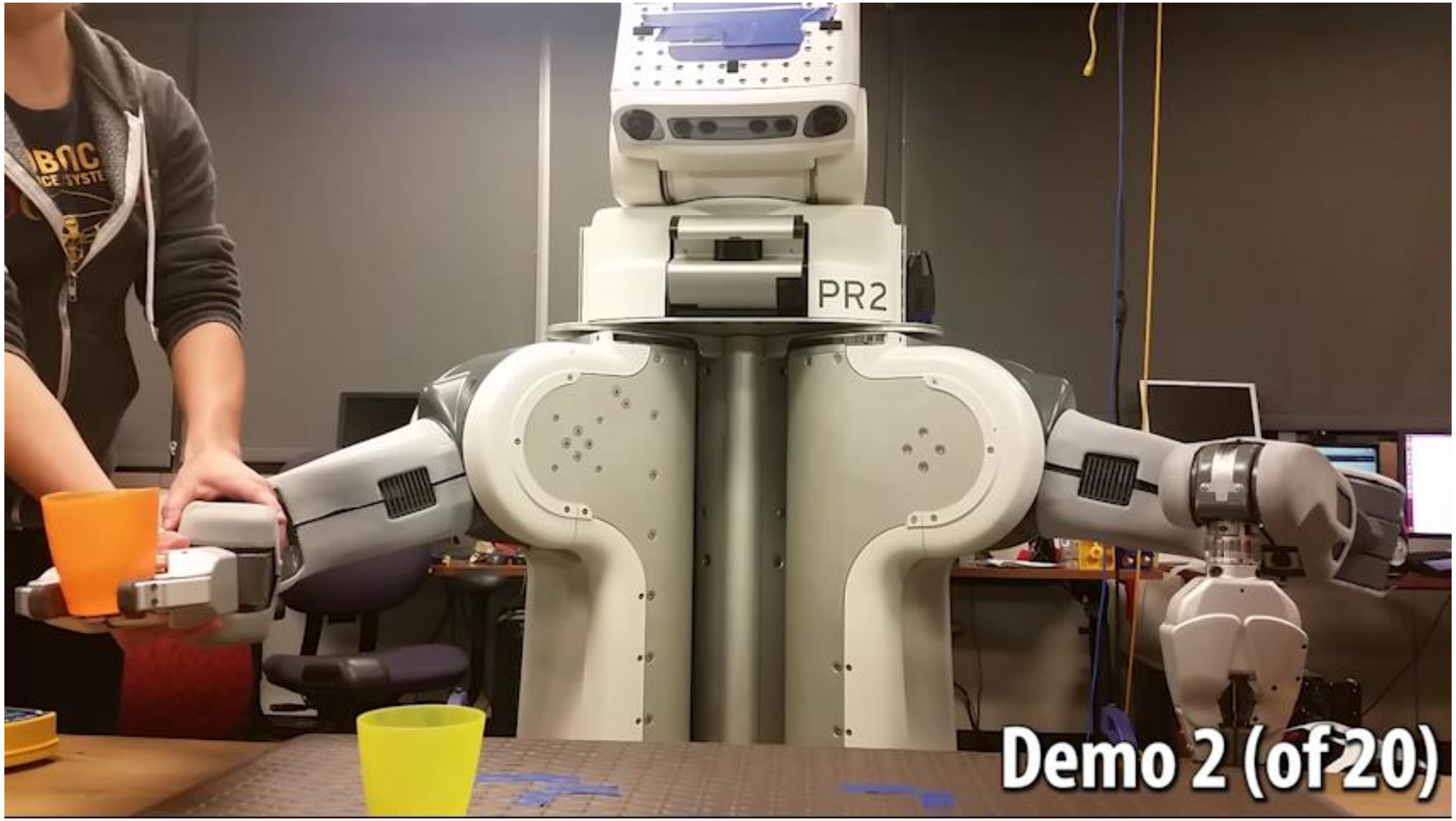
slides adapted from C. Finn

Dish placement, GCL policy



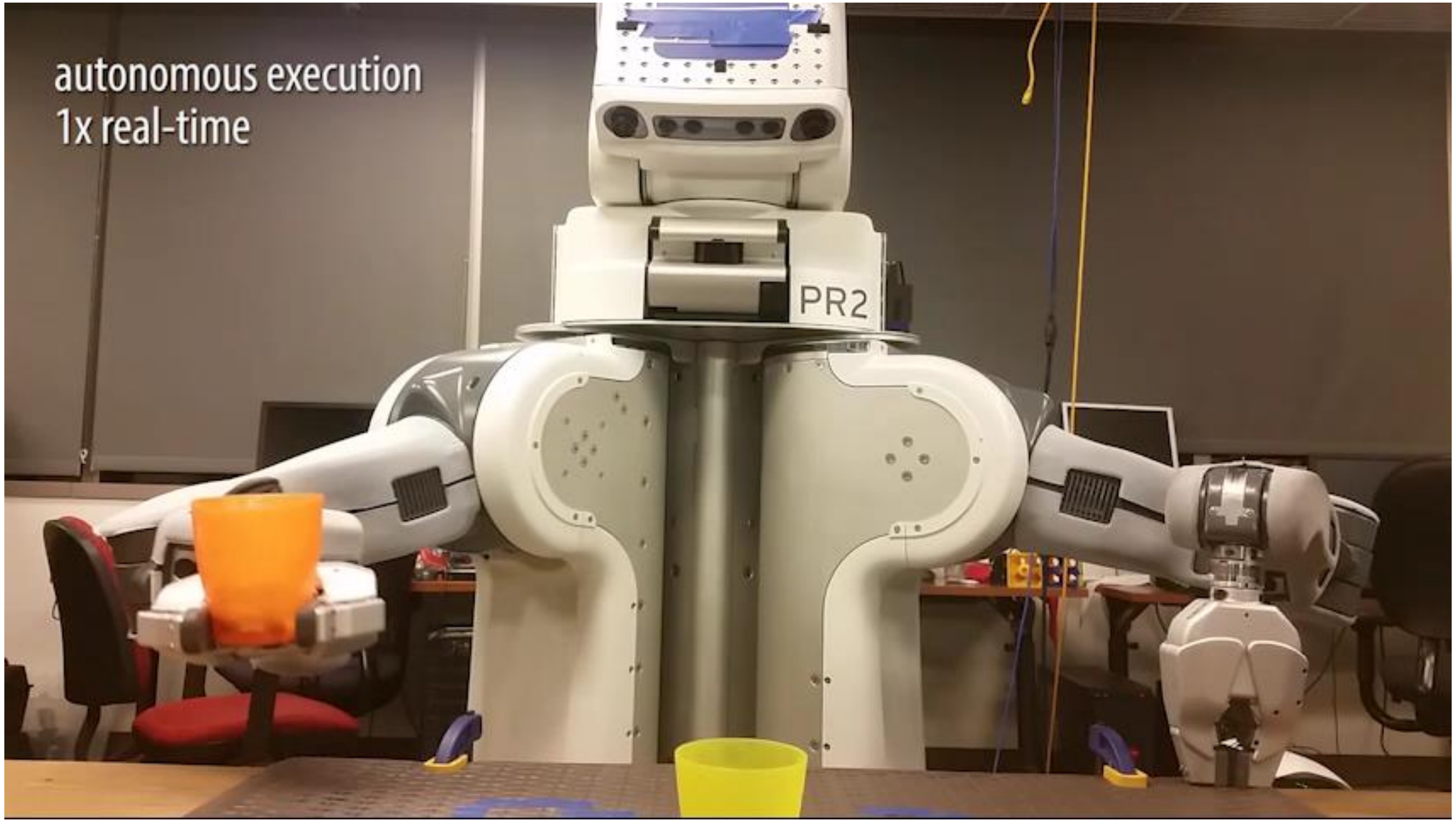
slides adapted from C. Finn

Pouring, demos



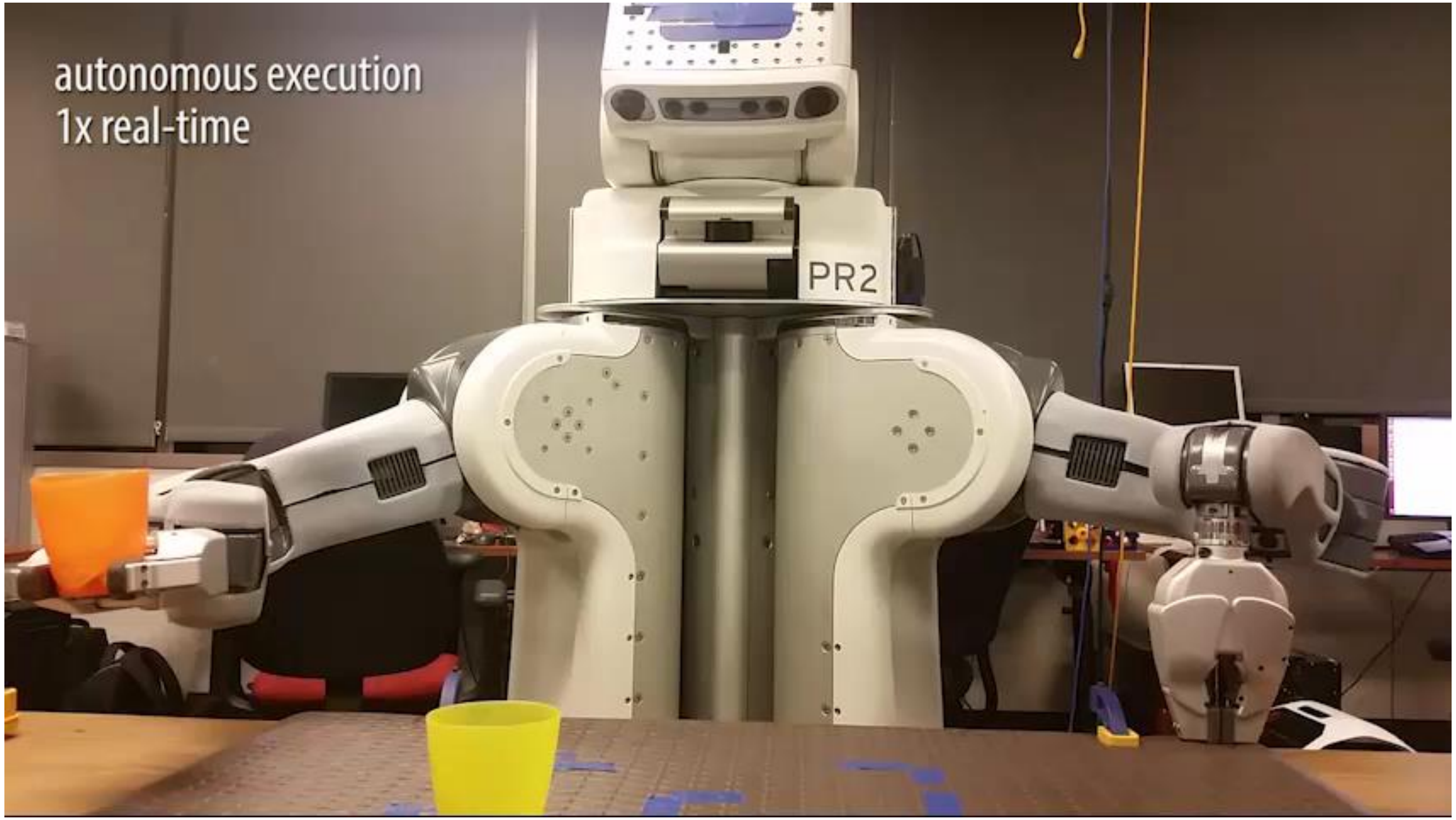
slides adapted from C. Finn

Pouring, RelEnt IRL



slides adapted from C. Finn

Pouring, GCL policy



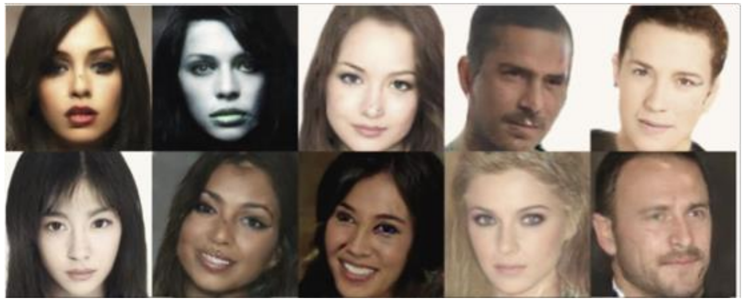
slides adapted from C. Finn

Aside: Generative Adversarial Networks

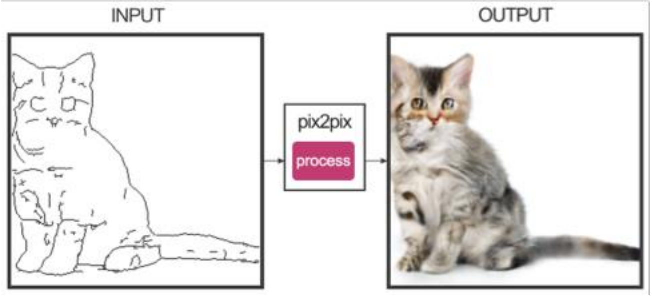
(Goodfellow et al. '14)

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

Zhu et al. '17



Arjovsky et al. '17



Isola et al. '17

Inverse RL

trajectory τ ⟷ **sample x**

policy π~q(τ) ⟷ **generator G**

reward r ⟶ **discriminator D**

GANs

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

Connection to Generative Adversarial Networks

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

Reward/discriminator optimization:

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

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

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

(Finn*, Christiano*, et al. ’16)

Connection to Generative Adversarial Networks

Inverse RL			GANs
trajectory τ	⟷	**sample x**
policy π~q(τ)	⟷	**generator G**
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 learned dynamics model to backdrop through discriminator

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

guided cost learning algorithm

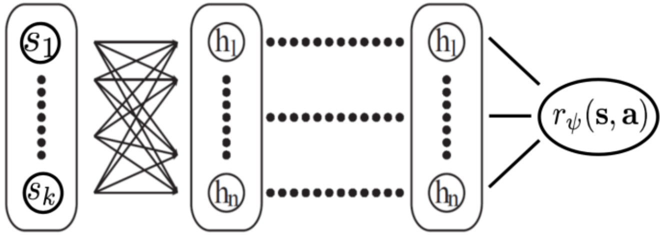
(Finn et al. ICML '16)

initial
policy π

human
demonstrations

generate policy
samples from π

generator



Update reward using
samples & demos

discriminator

update π w.r.t. reward
take 1 policy opt. step

policy π

reward r

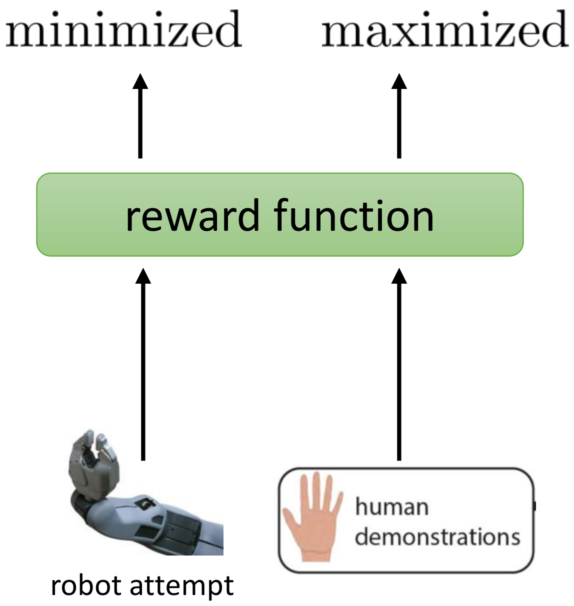
Update reward in inner loop of policy optimization

slides adapted from C. Finn

IRL as adversarial optimization

Guided Cost Learning

ICML 2016



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

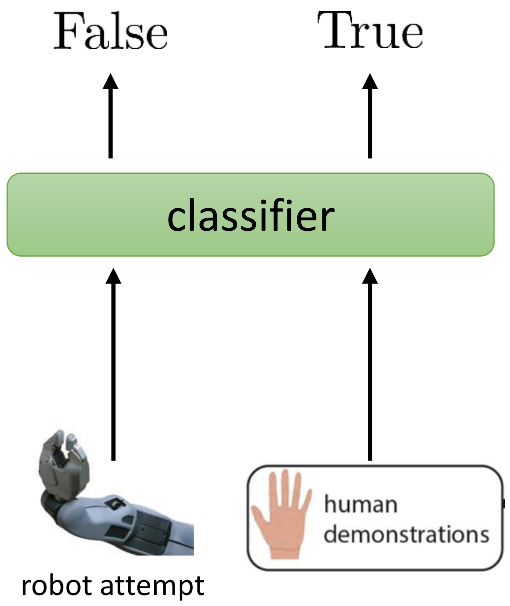
$p(\tau) \propto \exp(r(\tau))$ (MaxEnt model)

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

actually the same thing!

Generative Adversarial Imitation Learning

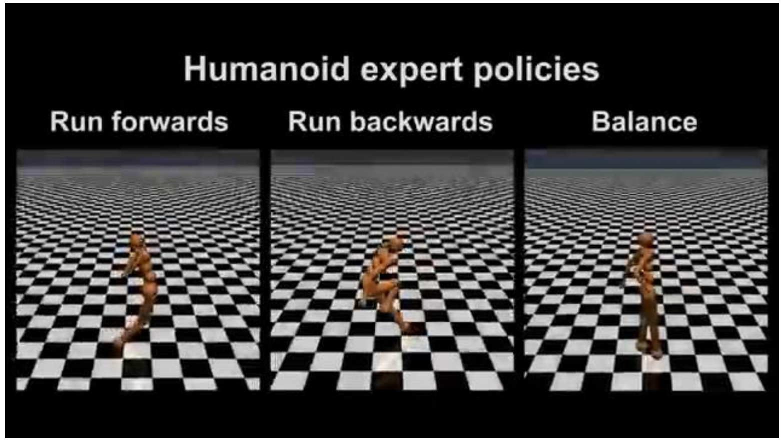
Ho & Ermon, NIPS 2016



$D(\tau) =$ probability τ is a demo

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

$D(\tau) =$ some classifier



Hausman, Chebotar, Schaal, Sukhatme, Lim

Merel, Tassa, TB, Srinivasan, Lemmon, Wang, Wayne, Heess

Generative Adversarial Imitation Learning Experiments

(Ho & Ermon NIPS '16)

learned behaviors from human motion capture
Merel et al. '17

walking

falling & getting up

IRL = generative adversarial networks = energy-based models

Guided cost learning optimizes the MaxEnt IRL objective:

$$p(\tau) \propto \exp(r(\tau)) \text{ (MaxEnt model)}$$

energy

This is exactly the same as GAN with this discriminator:

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

MaxEnt IRL involves optimizing an energy-based model

GANs are energy-based models

Finn*, Christiano*, Abbeel, L. '16

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
- Differential MaxEnt IRL: good for large, continuous spaces, but requires known dynamics and is local
- 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 very 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