

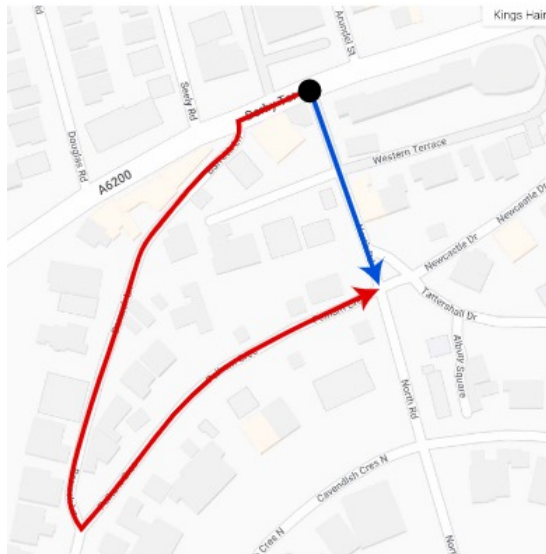


Reinforcement Learning

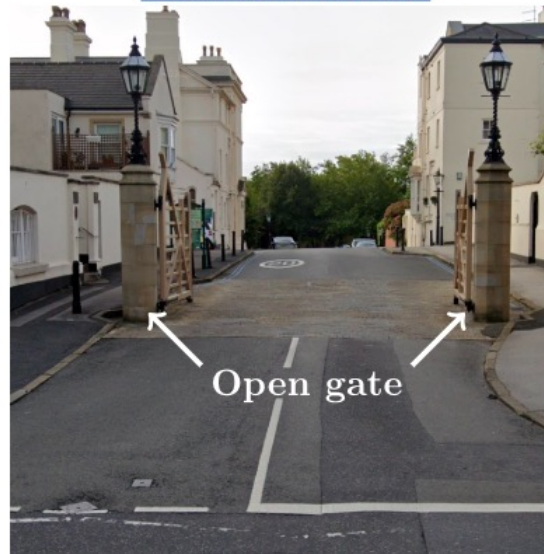
Spring 2026

Abhishek Gupta

TA: Mateo Guaman Castro



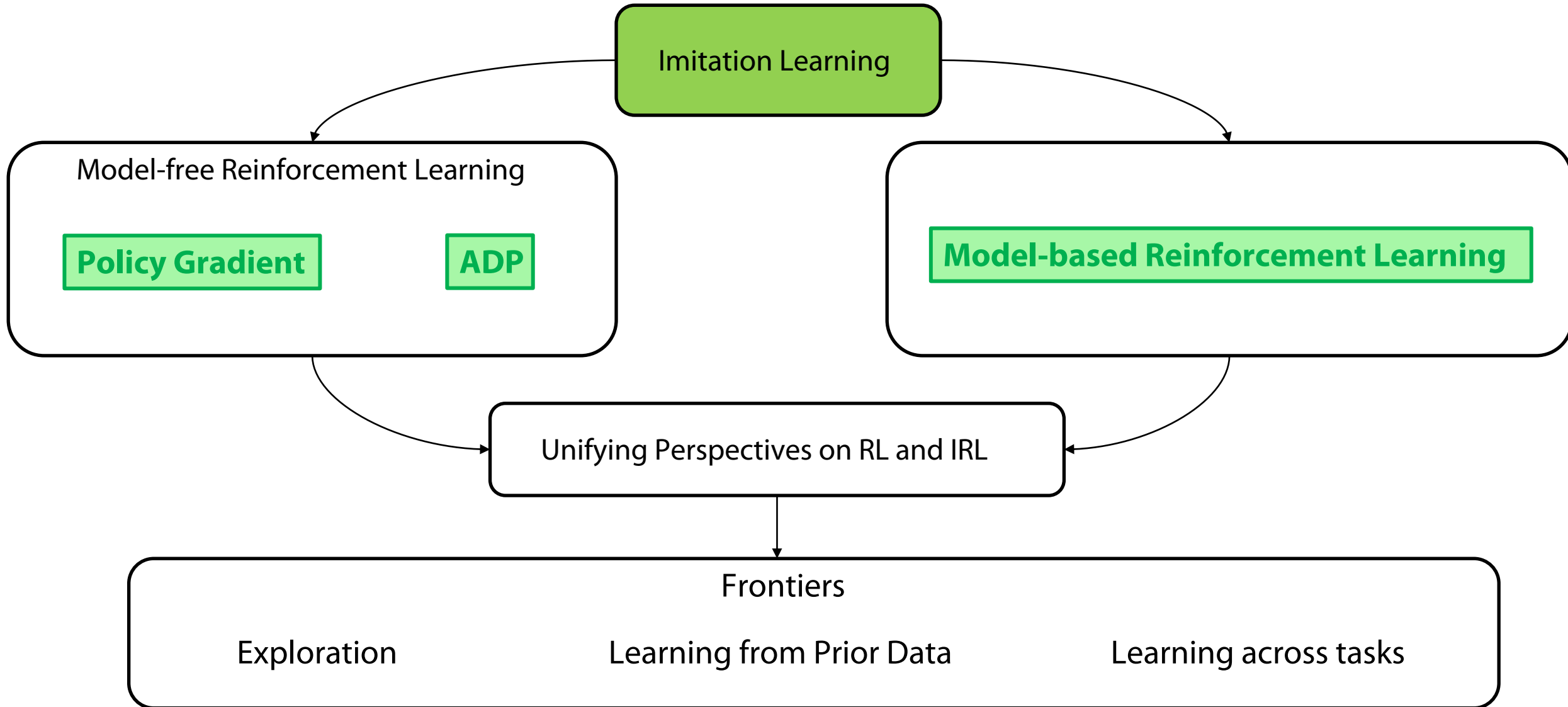
Preferred route



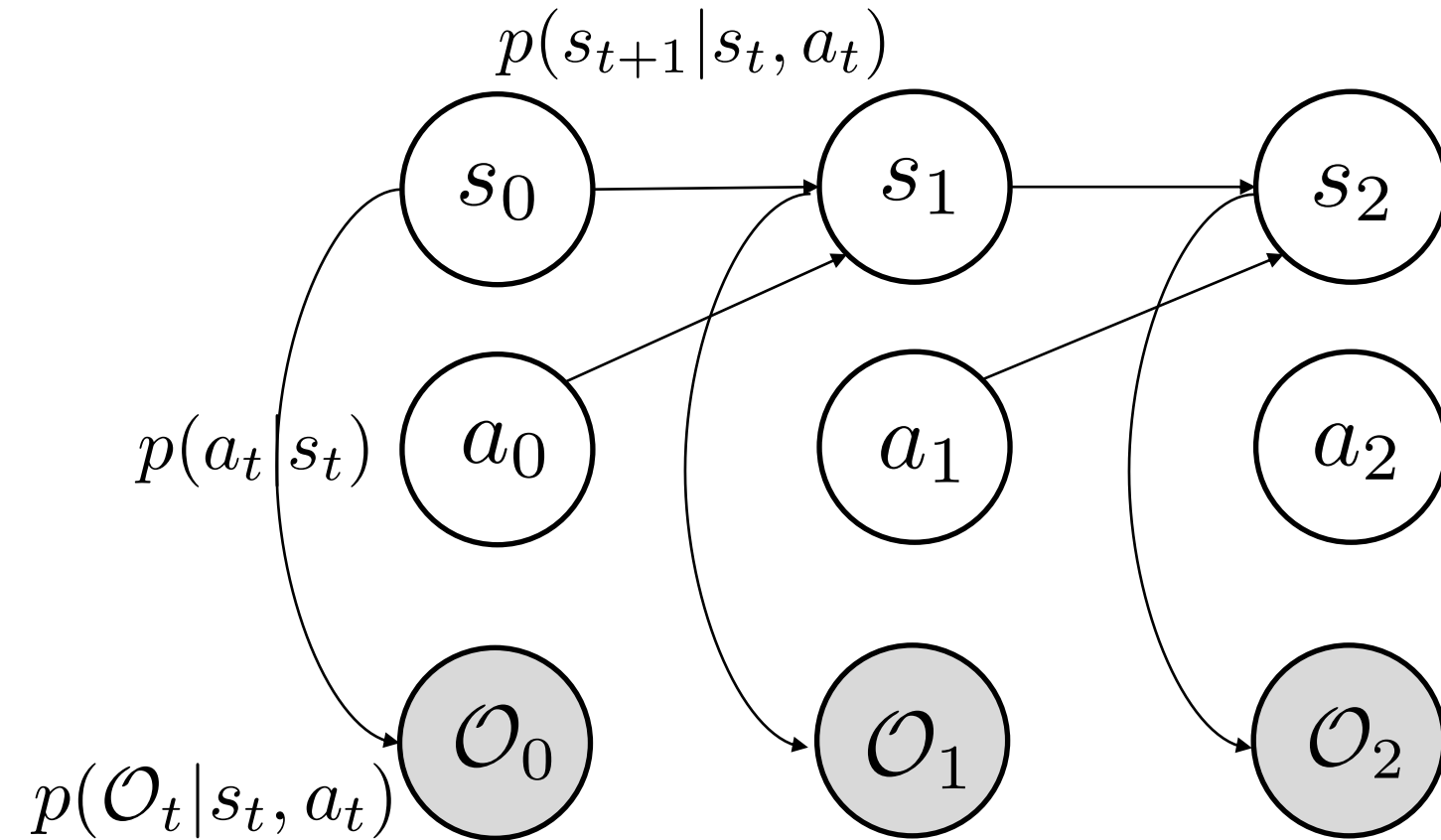
Longer route



Class Structure



Using Probabilistic Graphical Models for Decision Making



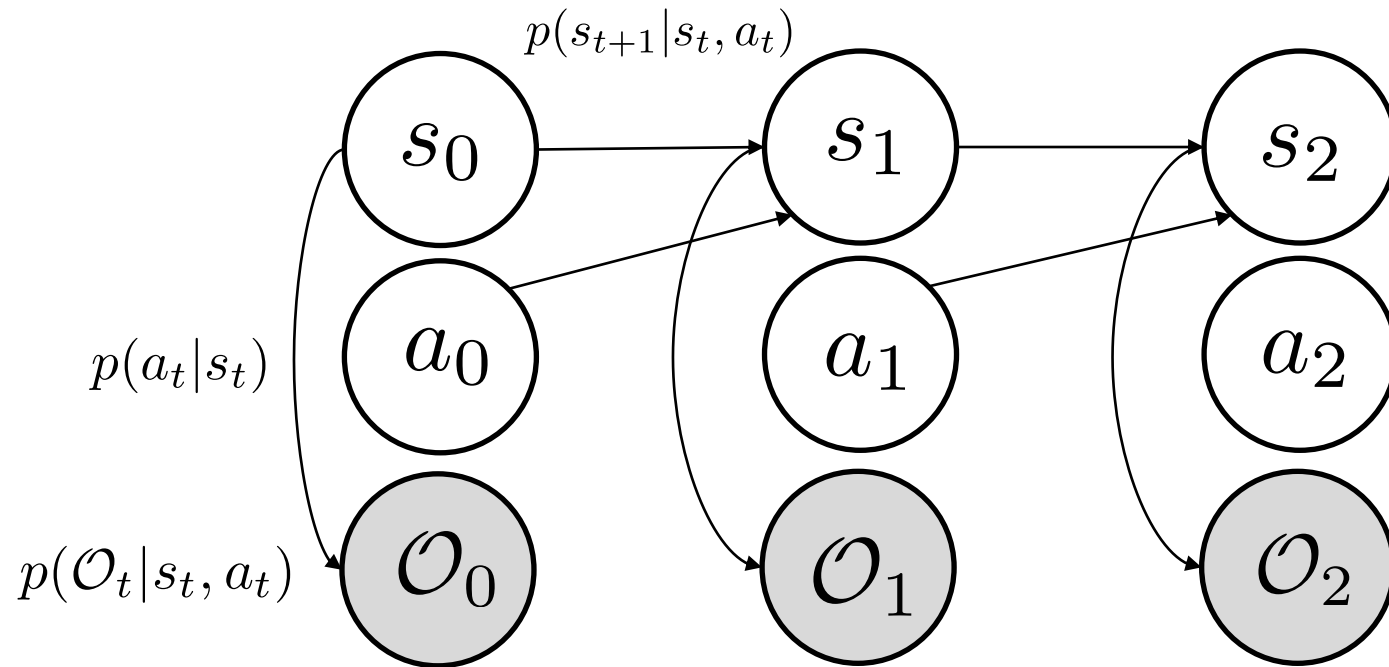
$$p(O_t|s_t, a_t) = \exp(r(s_t, a_t))$$

Rewards must be negative
(subtract max reward WLOG)

Introduce binary “optimality” variables – optimal if $O=1$, suboptimal if $O=0$

Agents are observed to be **optimal**

Ok big whoop, what do we do this?



$$p(O_t|s_t, a_t) = \exp(r(s_t, a_t))$$

$$p(\tau|\mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

Use case 1:

Derive soft RL algorithms

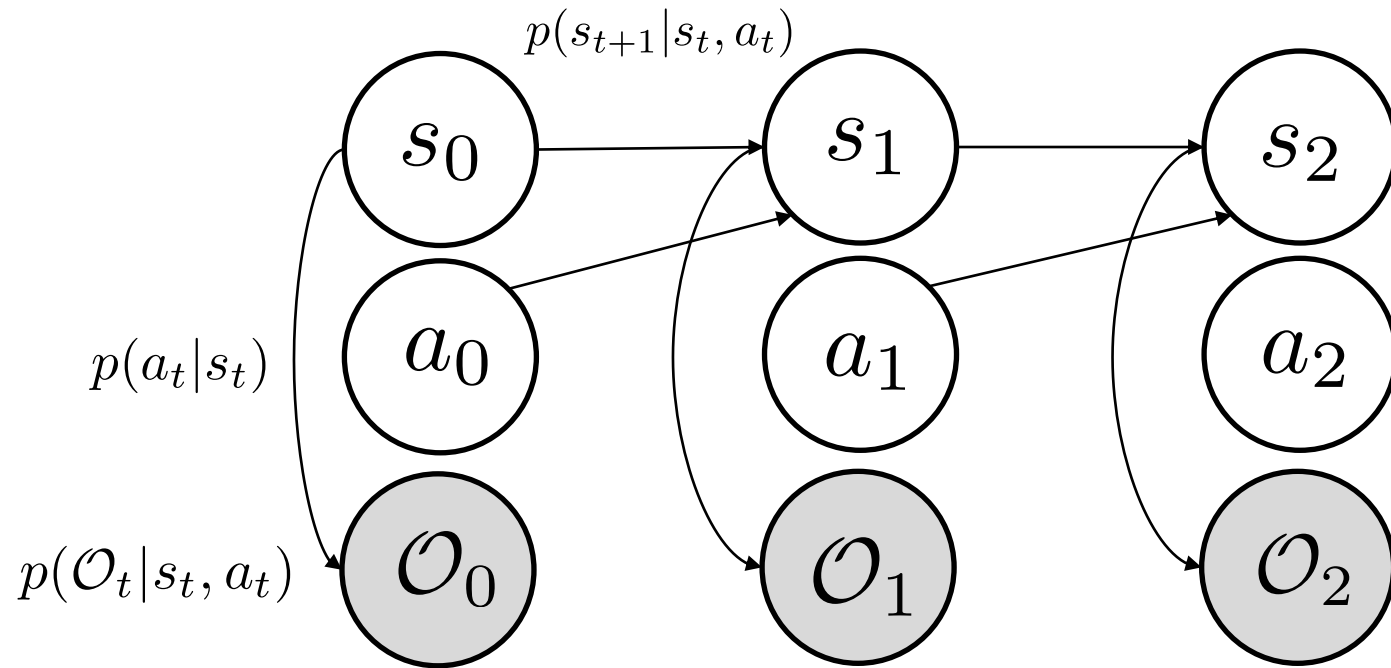
Use case 2:

Derive soft inverse RL algorithms

Use case 3:

Great algorithms for transfer

So what are we doing inference over?



$$p(O_t|s_t, a_t) = \exp(r(s_t, a_t))$$

$$p(\tau|\mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

Use case 1:

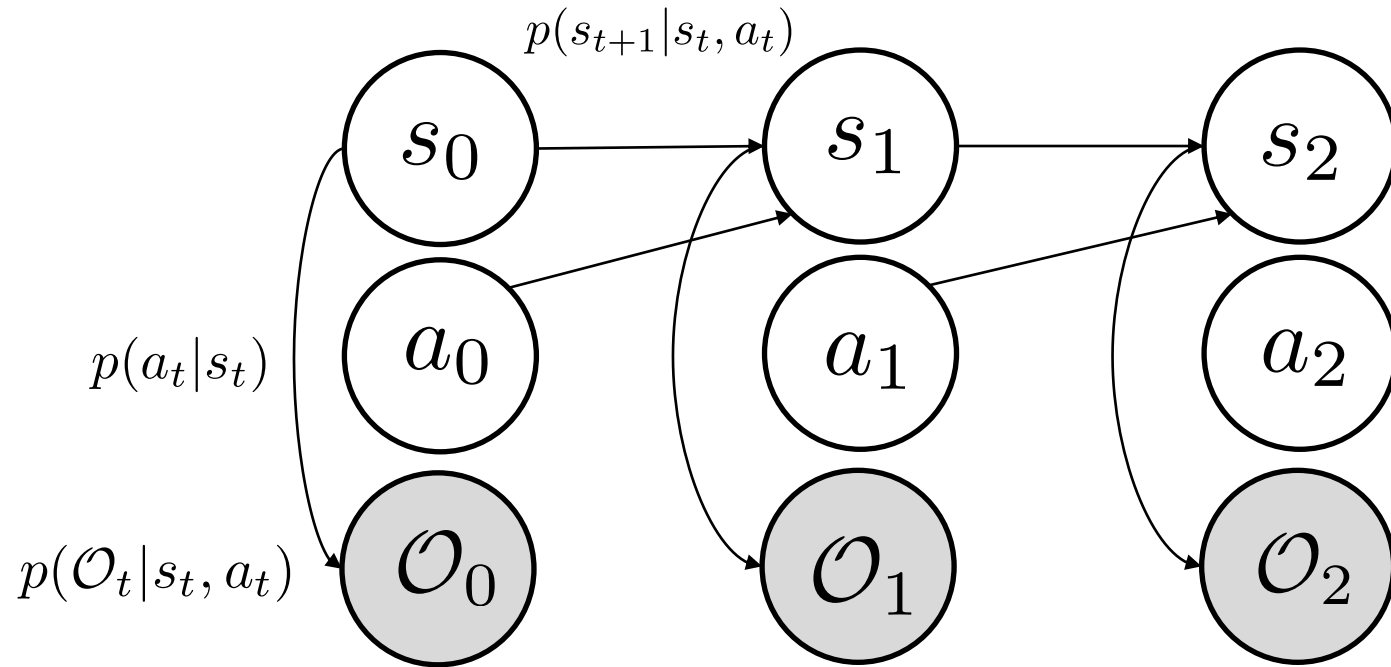
Derive soft RL algorithms

Insight: Computing optimal policy \rightarrow posterior inference

$$p(a_t|s_t, \mathcal{O}_{t:T} = 1)$$

“Given that you are acting optimally, what is the likelihood of a particular action at a state”

Why isn't this trivial?



$$p(\mathcal{O}_t | s_t, a_t) = \exp(r(s_t, a_t))$$

$$p(\tau | \mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

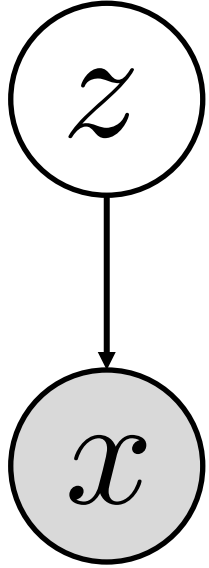
Optimal Policy \rightarrow Posterior Inference

$$p(a_t | s_t, \mathcal{O}_{t:T} = 1) = \frac{p(a_t, \mathcal{O}_{t:T} = 1 | s_t)}{p(\mathcal{O}_{t:T} = 1 | s_t)} = \frac{\int \int \cdots \int p(a_{t:T}, \mathcal{O}_{t:T} = 1, s_{t:T}) ds_{t+1:T} da_{t+1:T}}{\int \int \cdots \int p(a_{t:T}, \mathcal{O}_{t:T} = 1, s_{t:T}) ds_{t+1:T} da_{t:T}}$$

“Given that you are acting optimally, what is the likelihood of a particular action at a state”

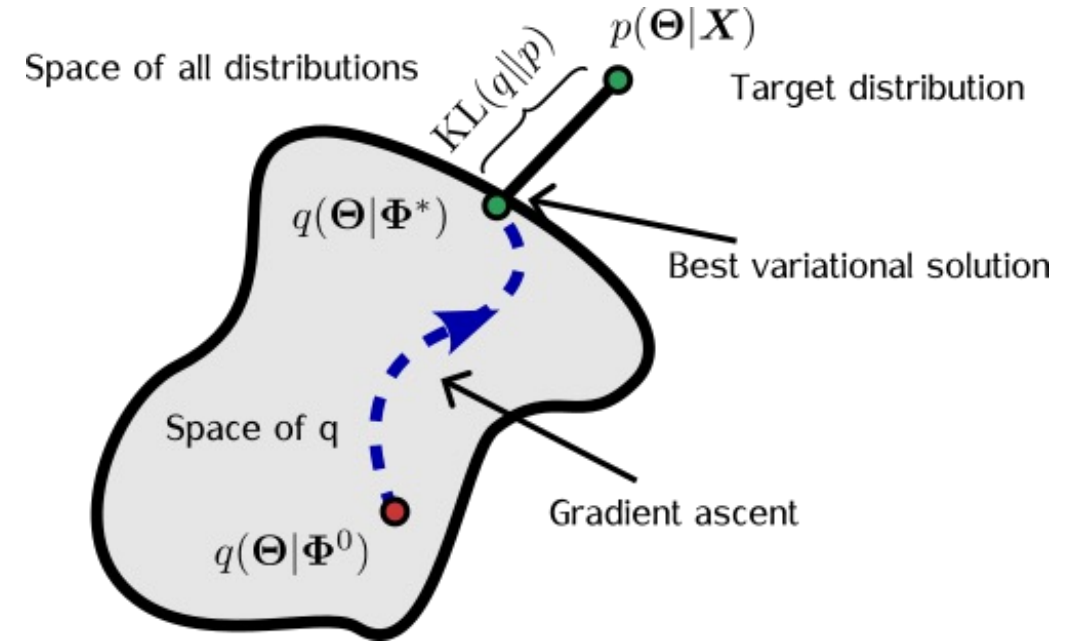
Difficult/intractable to compute
 \rightarrow Most RL algorithms are approximations to this

What is the key idea behind variational inference?



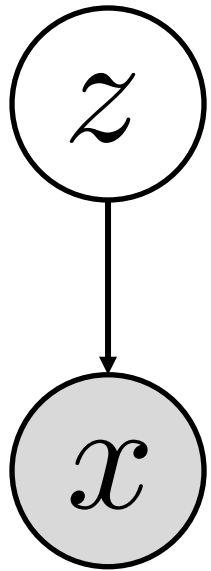
$$p(z|x) = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz}$$

Intractable!



Approximate challenging posterior with closest possible "tractable" posterior

Let's derive the Evidence Lower Bound



$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Intractable!

$$\phi^* \leftarrow \arg \min_{\phi} D_{KL}(q_{\phi}(z|x) || p(z|x))$$

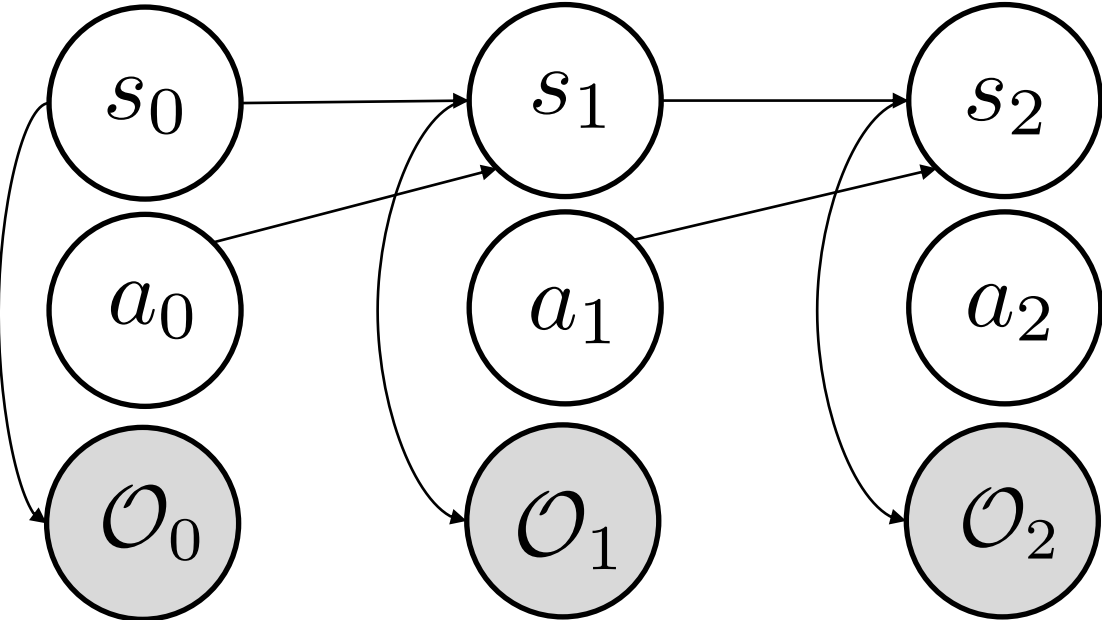
Unknown
Known

$$D_{KL}(q_{\phi}(z|x) || p(z|x)) = D_{KL}(q(z|x) || p(z)) - \mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] + \log p(x)$$

View 1: Find best posterior

View 2: Maximize marginal likelihood

Lets revisit our original inference problem in control



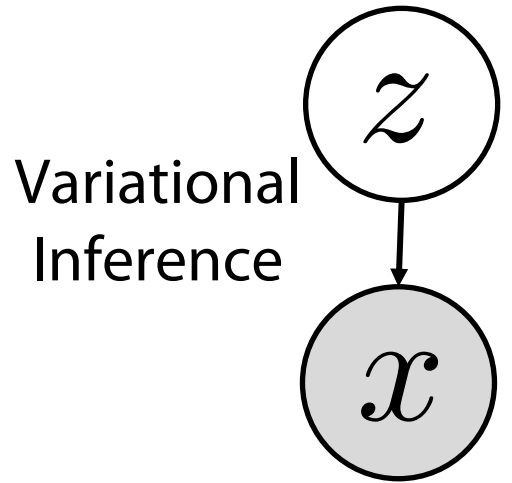
Optimal Policy \rightarrow Posterior Inference

Approximate $p(a_t | s_t, \mathcal{O}_{t:T} = 1)$ by $q(a_t | s_t, \mathcal{O}_{t:T} = 1)$

$$\max_q \mathbb{E}_{x \sim p(x)} \left[\mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) \right]$$

x
 \updownarrow
 $(\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_T)$

z
 \updownarrow
 $(s_0, a_0, s_1, a_1, \dots, s_T, a_T)$



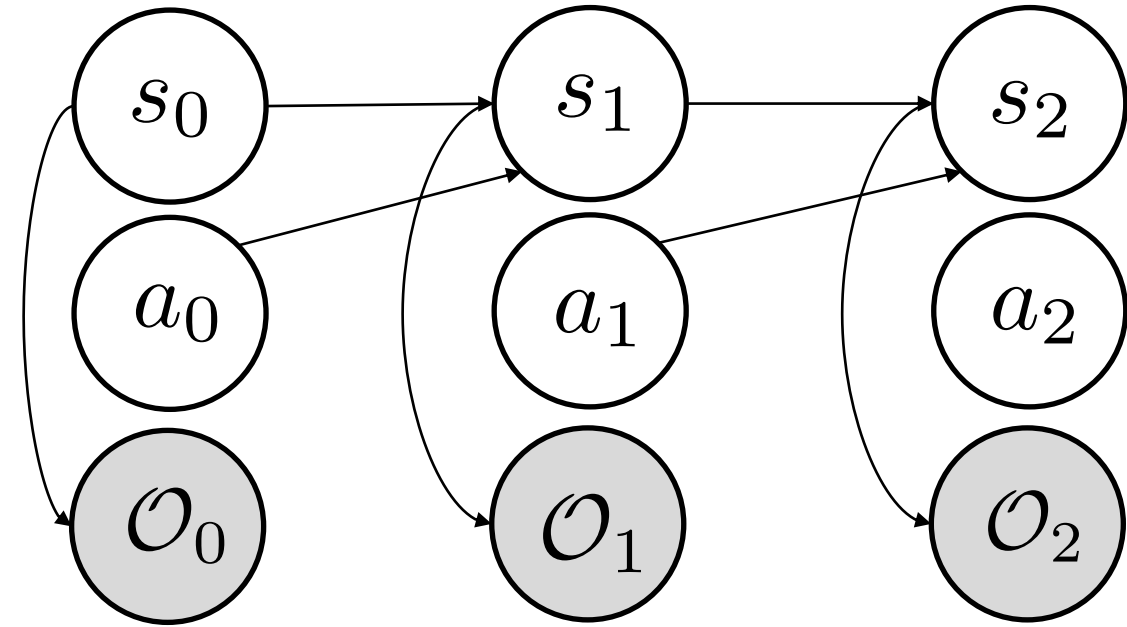
So what do we need to compute this?

\rightarrow Choice for $q(z|x) -- q(s_0, a_0, \dots, s_T, a_T | \mathcal{O}_0, \dots, \mathcal{O}_T)$

Key desiderata:

- 1) Can sample,
- 2) Compute KL Divergence

Choice of Variational Family for Approximate Inference



$$\max_q \mathbb{E}_{x \sim p(x)} \left[\mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) \right]$$

x z
 \updownarrow \updownarrow
 $(\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_T)$ $(s_0, a_0, s_1, a_1, \dots, s_T, a_T)$

So what do we need to compute this?

→ Choice for $q(s_0, a_0, \dots, s_T, a_T | \mathcal{O}_0, \dots, \mathcal{O}_T)$

Use temporal structure of the **true** dynamics in q

$$q(s_0, a_0, \dots, s_T, a_T | \mathcal{O}_0, \dots, \mathcal{O}_T) = p(s_0) p(s_1 | s_0, a_0) q(a_0 | s_0) p(s_2 | s_1, a_1) q(a_1 | s_1) \dots q(a_T | s_T)$$

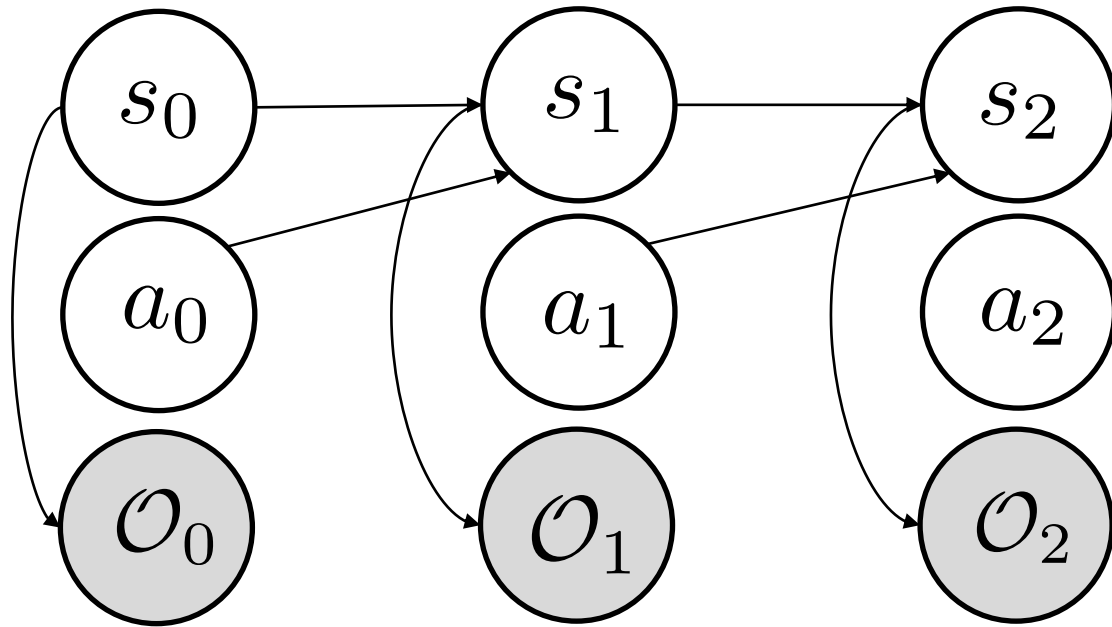
Does it satisfy:

- 1) Can sample,
- 2) Compute KL Divergence

$$= p(s_0) \prod_{t=0}^T p(s_{t+1} | s_t, a_t) q(a_t | s_t)$$

True dynamics and initial state
Approximate policy for $p(a_t | s_t, \mathcal{O}_{t:T} = 1)$

Ok so what did we show?



Find approximate posterior $q(z|x)$ by optimizing the ELBO

$$\max_q \mathbb{E}_{x \sim p(x)} \left[\mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) \right]$$

x z
 \updownarrow \updownarrow
 $(\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_T)$ $(s_0, a_0, s_1, a_1, \dots, s_T, a_T)$

$$q(s_0, a_0, \dots, s_T, a_T | \mathcal{O}_0, \dots, \mathcal{O}_T) = p(s_0) \prod_{t=0}^T p(s_{t+1} | s_t, a_t) q(a_t | s_t)$$

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t \log p(\mathcal{O}_t | s_t, a_t) - \log q(a_t | s_t) \right] = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot | s_t)) \right]$$

Maximize ELBO with SGD = policy gradient!

Lecture outline

Recap



Control as Inference to Derive Q-learning

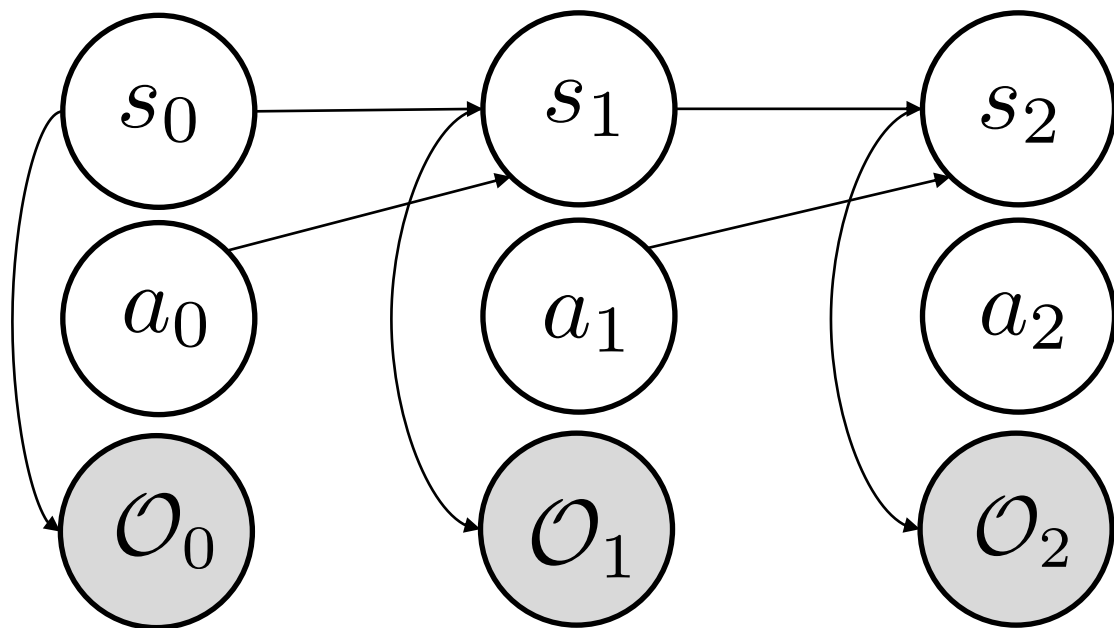


Control as Inference to Derive Model-Based RL



Inverse RL

Can we derive (soft) Q-learning from the ELBO?



Find approximate posterior $q(z|x)$ by optimizing the ELBO

$$\max_q \mathbb{E}_{x \sim p(x)} \left[\mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) \right]$$

x
 \updownarrow
 $(\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_T)$

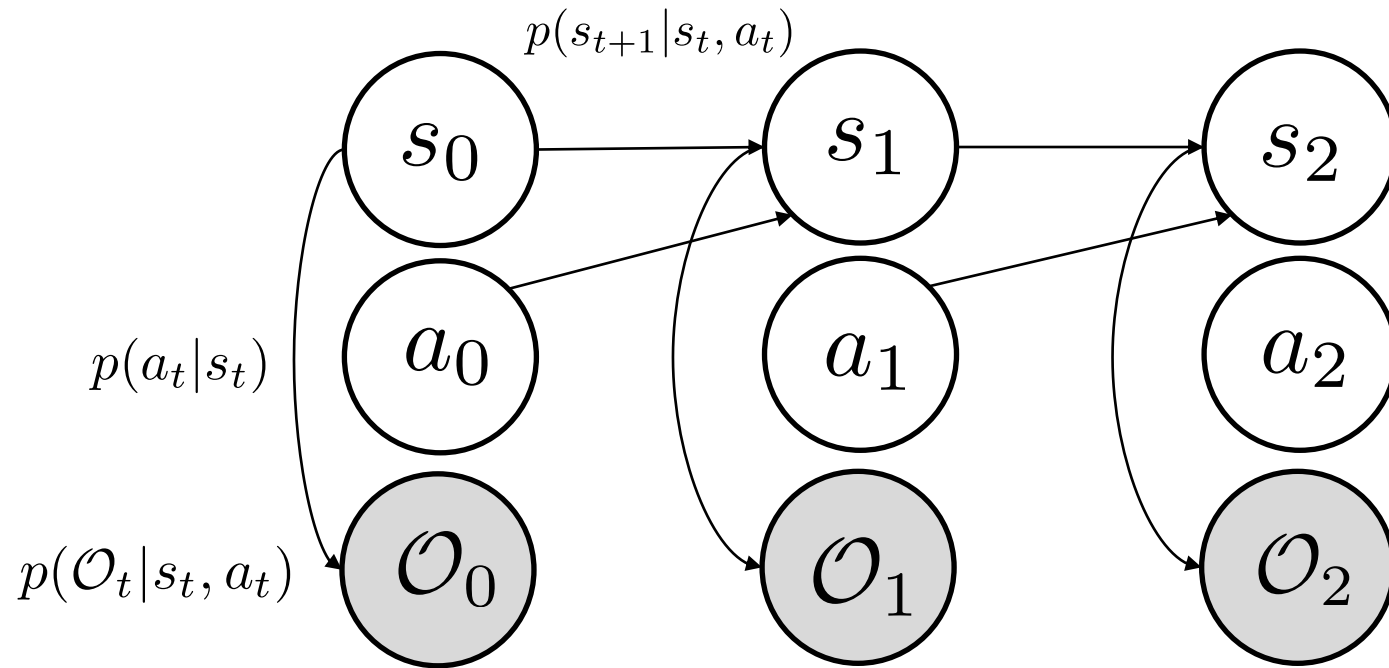
z
 \updownarrow
 $(s_0, a_0, s_1, a_1, \dots, s_T, a_T)$

$$q(s_0, a_0, \dots, s_T, a_T | \mathcal{O}_0, \dots, \mathcal{O}_T) = p(s_0) \prod_{t=0}^T p(s_{t+1} | s_t, a_t) q(a_t | s_t)$$

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t \log p(\mathcal{O}_t | s_t, a_t) - \log q(a_t | s_t) \right] = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot | s_t)) \right]$$

Maximize ELBO with DP = Soft Q learning!

So what are we doing inference over?



$$p(\mathcal{O}_t|s_t, a_t) = \exp(r(s_t, a_t))$$

$$p(\tau|\mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

Use case 1:

Derive soft RL algorithms

Analogues for optimal Q and V

$$V(s_t) = \log p(\mathcal{O}_{t:T} = 1|s_t)$$

$$Q(s_t, a_t) = \log p(\mathcal{O}_{t:T} = 1|s_t, a_t)$$

“Likelihood of being optimal in the future at some state, action”

Let's optimize the last step of the ELBO

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot | s_t)) \right]$$

Consider the last time step

$$\begin{aligned} \max \mathbb{E}_{s_T \sim q(s_T)} & \left[\mathbb{E}_{a_T \sim q(a_T | s_T)} [r(s_T, a_T) - \log q(a_T | s_T)] \right] \\ & \text{(log-exp)} \\ = \mathbb{E}_{s_T \sim q(s_T)} & \left[\mathbb{E}_{a_T \sim q(a_T | s_T)} [\log \exp(r(s_T, a_T)) - \log q(a_T | s_T)] \right] \\ = \mathbb{E}_{s_T \sim q(s_T)} & \left[\mathbb{E}_{a_T \sim q(a_T | s_T)} \left[\log \exp(r(s_T, a_T)) - \log \int \exp(r(s_T, a_T)) da_T - \log q(a_T | s_T) \right] + \log \int \exp(r(s_T, a_T)) da_T \right] \\ & \text{(Add subtract to normalize)} \\ = \mathbb{E}_{s_T \sim q(s_T)} & \left[\mathbb{E}_{a_T \sim q(a_T | s_T)} \left[\log \frac{\exp(r(s_T, a_T))}{\int \exp(r(s_T, a_T)) da_T} - \log q(a_T | s_T) \right] \right] \\ = \mathbb{E}_{s_T \sim q(s_T)} & \left[\mathbb{E}_{a_T \sim q(a_T | s_T)} \left[-\log \frac{q(a_T | s_T)}{\frac{\exp(r(s_T, a_T))}{\int \exp(r(s_T, a_T)) da_T}} \right] \right] = \mathbb{E}_{s_T \sim q(s_T)} \left[-D_{KL}(q(a_T | s_T) \parallel \frac{\exp(r(s_T, a_T))}{\int \exp(r(s_T, a_T)) da_T}) \right] \\ & \text{(Definition of KL divergence)} \end{aligned}$$

Let's optimize the last step of the ELBO

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot | s_t)) \right]$$

Consider the last time step

$$\max \mathbb{E}_{s_T \sim q(s_T)} \left[\mathbb{E}_{a_T \sim q(a_T | s_T)} [r(s_T, a_T) - \log q(a_T | s_T)] \right]$$

$$= \mathbb{E}_{s_T \sim q(s_T)} \left[-D_{KL}(q(a_T | s_T) \parallel \frac{\exp(r(s_T, a_T))}{\int \exp(r(s_T, a_T)) da_T}) \right]$$

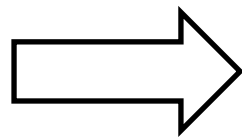
KL-divergence $D(p, q)$ is always non negative and is minimized when $p = q$

$$q(a_T | s_T) = \frac{\exp(r(s_T, a_T))}{\int \exp(r(s_T, a_T)) da_T}$$

Ok let's simplify

$$Q(s_T, a_T) = r(s_T, a_T)$$

$$V(s_T) = \log \int \exp(r(s_T, a_T)) da_T$$



$$q(a_T | s_T) = \exp(Q(s_T, a_T) - V(s_T))$$

Optimal policy is proportional to exponential advantage
(soft-max)

Let's optimize the step before

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot | s_t)) \right]$$

Consider the second last time step

$$\arg \max_q \mathbb{E}_{s_{T-1} \sim q(s_{T-1})} \left[\mathbb{E}_{a_{T-1} \sim q(a_{T-1} | s_{T-1})} \left[r(s_{T-1}, a_{T-1}) - \log q(a_{T-1} | s_{T-1}) + \mathbb{E}_{\substack{s_T \sim p(s_T | s_{T-1}, a_{T-1}) \\ a_T \sim q(a_T | s_T)}} [r(s_T, a_T) - \log q(a_T | s_T)] \right] \right]$$

Exactly what we computed in the last step

$$\arg \max_q \mathbb{E}_{s_{T-1} \sim q(s_{T-1})} \left[\mathbb{E}_{a_{T-1} \sim q(a_{T-1} | s_{T-1})} \left[r(s_{T-1}, a_{T-1}) - \log q(a_{T-1} | s_{T-1}) + \mathbb{E}_{s_T \sim p(s_T | s_{T-1}, a_{T-1})} [V(s_T)] \right] \right]$$

From the last slide

Let us call this $Q(s_{T-1}, a_{T-1})$

$$Q(s_{T-1}, a_{T-1}) = r(s_{T-1}, a_{T-1}) + \mathbb{E}_{s_T \sim p(s_T | s_{T-1}, a_{T-1})} [V(s_T)] \quad \text{(Looks like Bellman!)}$$

$$\arg \max_q \mathbb{E}_{s_{T-1} \sim q(s_{T-1})} \left[\mathbb{E}_{a_{T-1} \sim q(a_{T-1} | s_{T-1})} \left[Q(s_{T-1}, a_{T-1}) - \log q(a_{T-1} | s_{T-1}) \right] \right]$$

Looks a lot like the previous time-step

Let's optimize the step before

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot | s_t)) \right]$$

Consider the second last time step

$$\arg \max_q \mathbb{E}_{s_{T-1} \sim q(s_{T-1})} \left[\mathbb{E}_{a_{T-1} \sim q(a_{T-1} | s_{T-1})} \left[Q(s_{T-1}, a_{T-1}) - \log q(a_{T-1} | s_{T-1}) \right] \right]$$

$$Q(s_{T-1}, a_{T-1}) = r(s_{T-1}, a_{T-1}) + \mathbb{E}_{s_T \sim p(s_T | s_{T-1}, a_{T-1})} [V(s_T)]$$

Referring back to the last time step math and pattern matching

$$V(s_{T-1}) = \log \int \exp(Q(s_{T-1}, a_{T-1})) da_{T-1}$$

$$Q(s_{T-1}, a_{T-1}) = r(s_{T-1}, a_{T-1}) + \mathbb{E}_{s_T \sim p(s_T | s_{T-1}, a_{T-1})} [V(s_T)]$$

$$q(a_{T-1} | s_{T-1}) = \exp(Q(s_{T-1}, a_{T-1}) - V(s_{T-1}))$$

Optimal policy is proportional to exponential advantage
(soft-max)

Let's make it recursive

This suggests a recursive dynamic programming algorithm!

$$Q(s_T, a_T) = r(s_T, a_T)$$

$$V(s_T) = \log \int \exp(r(s_T, a_T)) da_T$$

For $t = T-1$ to 1:

$$Q_t(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} [V_{t+1}(s_{t+1})] \quad \text{(Bellman update)}$$

$$V_t(s_t) = \log \int \exp(Q_t(s_t, a_t)) da_t \quad \text{(Soft-max)}$$

$$q(a_t|s_t) = \exp(Q_t(s_t, a_t) - V_t(s_t)) \quad \text{(Soft-max)}$$

Very similar to the “soft” (entropy) Q-learning procedure from earlier lectures!

What does this suggest as an algorithm?

Optimize a "soft" Bellman equation

$$Q(s_t, a_t) \leftarrow r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s} [V(s_{t+1})]$$

$$Q_{\text{soft}}(s_t, a_t) \leftarrow r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s} [V_{\text{soft}}(s_{t+1})]$$

$$V(s_t) \leftarrow \max_a Q(s_t, a)$$

$$V_{\text{soft}}(s_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp \left(\frac{1}{\alpha} Q_{\text{soft}}(s_t, a') \right) da'$$

$$\pi(a|s_t) \leftarrow \arg \max_a Q(s_t, a)$$

$$\pi_{\text{soft}}(a|s_t) = \exp \left(\frac{1}{\alpha} (Q_{\text{soft}}(s_t, a) - V_{\text{soft}}(s_t)) \right)$$

Go from max to "softmax" (imagine if α goes to 0, it becomes a max)

Prevents premature collapse of exploration while smoothing out optimization landscape!

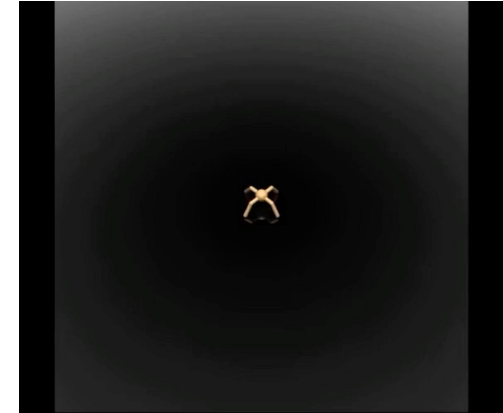
Why should we ever do soft-Q learning?

Optimization benefits

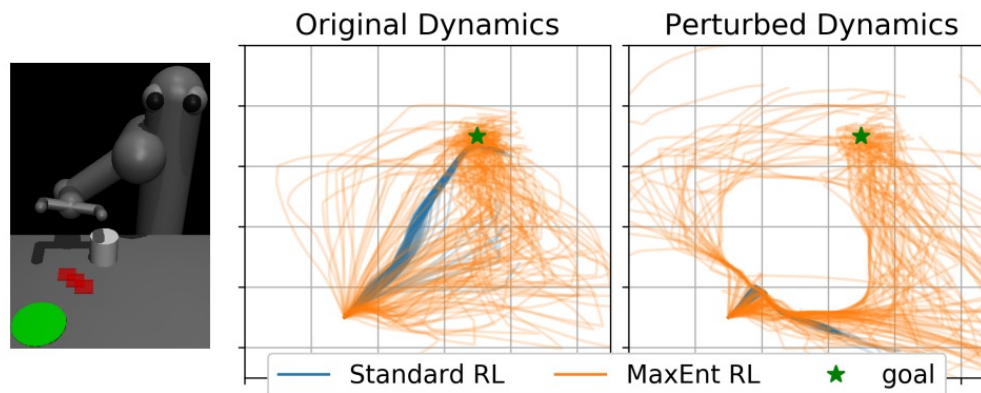
Corollary 5.1. (Iteration complexity with log barrier regularization) Let $\beta_\lambda := \frac{8\gamma}{(1-\gamma)^3} + \frac{2\lambda}{|\mathcal{S}|}$. Starting from any initial $\theta^{(0)}$, consider the updates (13) with $\lambda = \frac{\epsilon(1-\gamma)}{2\left\|\frac{d\rho^*}{\mu}\right\|_\infty}$ and $\eta = 1/\beta_\lambda$. Then for all starting state distributions ρ , we have

$$\min_{t < T} \{V^*(\rho) - V^{(t)}(\rho)\} \leq \epsilon \quad \text{whenever} \quad T \geq \frac{320|\mathcal{S}|^2|\mathcal{A}|^2}{(1-\gamma)^6 \epsilon^2} \left\|\frac{d\rho^*}{\mu}\right\|_\infty^2.$$

Transfer



Deals better with misspecification



Ok so what did we show?

Find approximate posterior $q(z|x)$ by optimizing the ELBO using dynamic programming

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t)}} \left[\sum_t \log p(\mathcal{O}_t | s_t, a_t) - \log q(a_t | s_t) \right] = \max_q \mathbb{E}_{x \sim p(x)} \left[\mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) \right] = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t|s_t) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot|s_t)) \right]$$

Can derive a “soft” dynamic programming Q-learning update

For $t = T-1$ to 1:

$$Q_t(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)} [V_{t+1}(s_{t+1})] \quad (\text{Bellman update})$$

$$V_t(s_t) = \log \int \exp(Q(s_t, a_t)) da_t \quad (\text{Soft-max})$$

$$q(a_t | s_t) = \exp(Q_t(s_t, a_t) - V_t(s_t)) \quad (\text{Soft-max})$$

Lecture outline

Recap



Control as Inference to Derive Q-learning

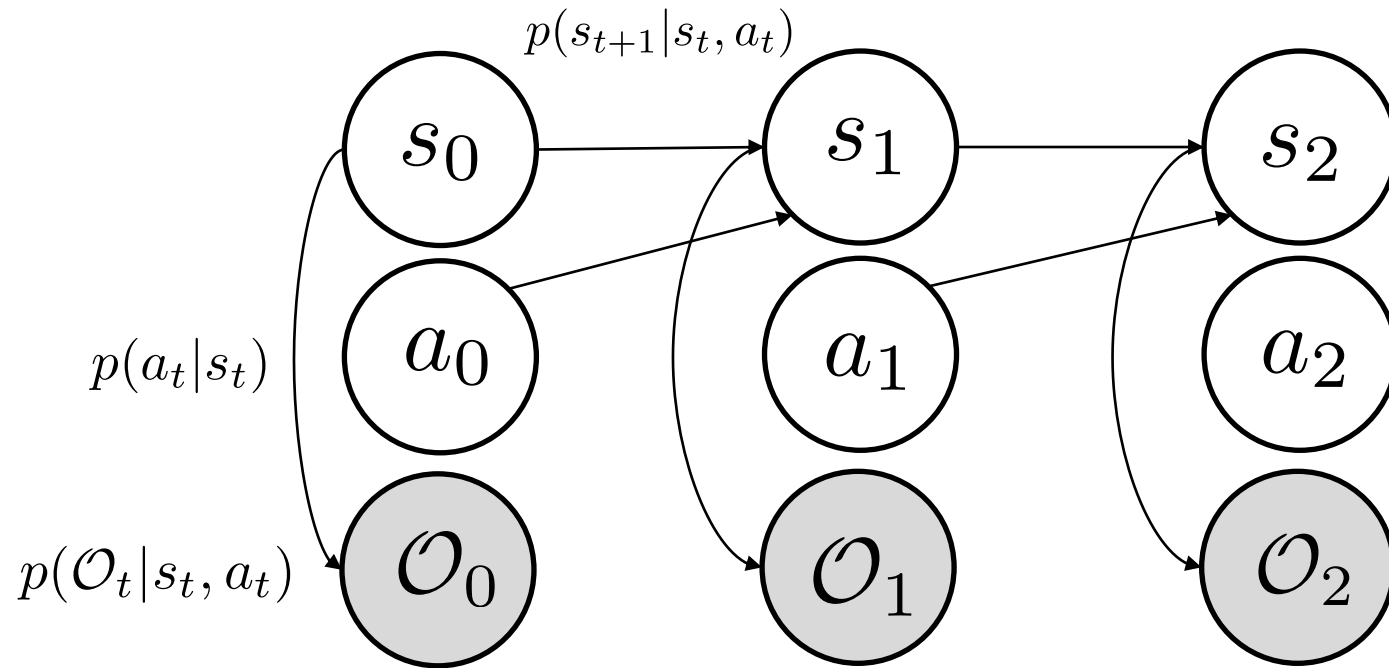


Control as Inference to Derive Model-Based RL



Inverse RL

Let's back up from VI to max likelihood



$$p(\mathcal{O}_t|s_t, a_t) = \exp(r(s_t, a_t))$$

$$p(\tau|\mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

Let us assume we get a bunch of data of (s, a, s', r) from the true system p

We will try to learn a surrogate model \hat{p} to approximate p , use it for posterior inference

Model Learning via Maximum Likelihood

$$\min_{\hat{p}} D_{KL}(p(s_0, \dots, s_T, a_0, \dots, a_T, \mathcal{O}_0, \dots, \mathcal{O}_T) || \hat{p}(s_0, \dots, s_T, a_0, \dots, a_T, \mathcal{O}_0, \dots, \mathcal{O}_T))$$

↓ Definition of KLD

$$\max_{\hat{p}} \mathbb{E}_{p(s_0, \dots, s_T, a_0, \dots, a_T, \mathcal{O}_0, \dots, \mathcal{O}_T)} [\log \hat{p}(s_0, \dots, s_T, a_0, \dots, a_T, \mathcal{O}_0, \dots, \mathcal{O}_T)]$$

↓ Expansion of joint

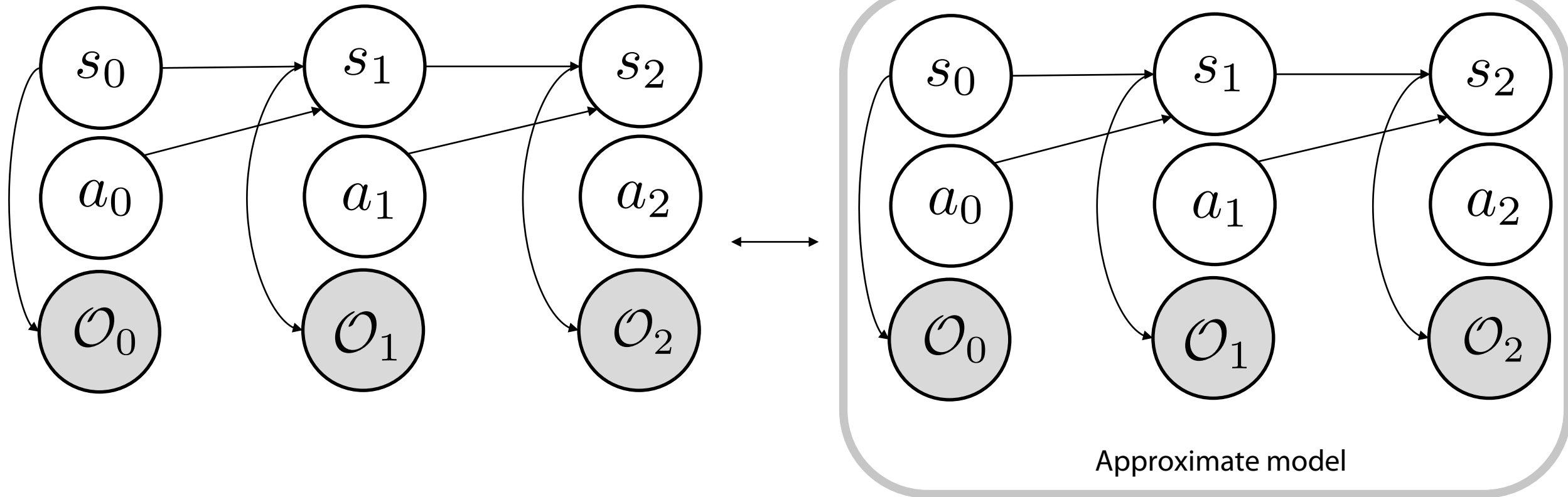
$$\max_{\hat{p}} \mathbb{E}_{p(s_0, \dots, s_T, a_0, \dots, a_T, \mathcal{O}_0, \dots, \mathcal{O}_T)} \left[\log \hat{p}(s_0) + \sum_t [\log \hat{p}(s_{t+1} | s_t, a_t) + \log \hat{p}(\mathcal{O}_t | s_t, a_t)] \right]$$

Model learning

Reward learning

Fitting \hat{p} amounts to supervised learning on dynamics and rewards

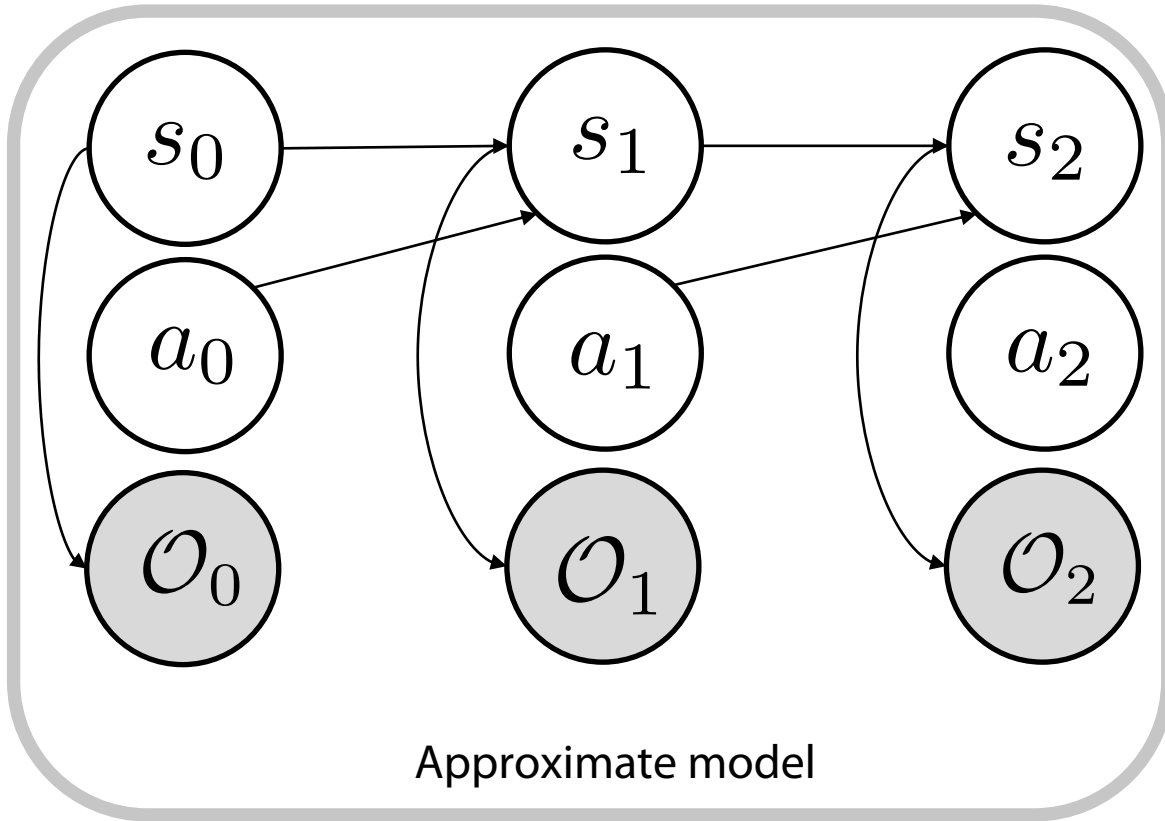
Model Learning via Maximum Likelihood



Fitting \hat{p} amounts to supervised learning on dynamics and rewards

How do we actually use this approximate model to obtain optimal actions?

Policy Extraction via Posterior Inference



Key idea: pretend that approximate model \hat{p} is the true model



$$\hat{p}(a_t | s_t, \mathcal{O}_{t:T} = 1)$$

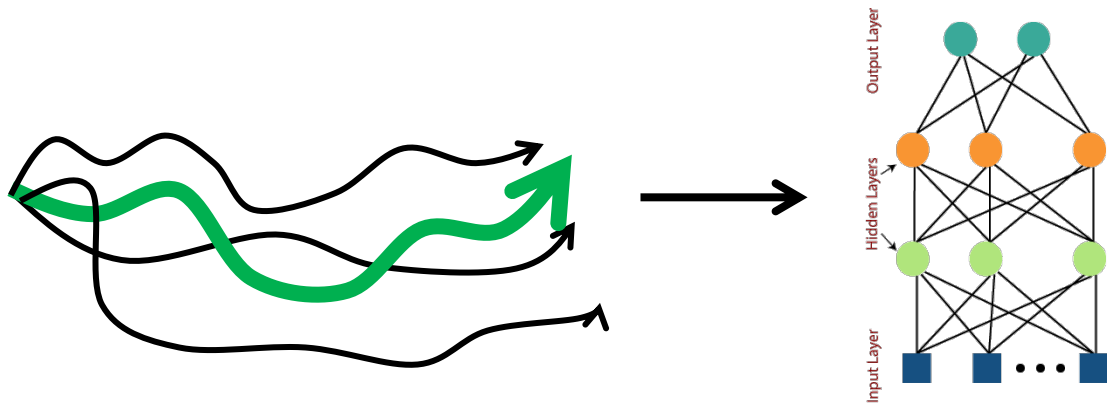
Just like in MFRL \rightarrow
perform posterior inference

Certainty equivalence \longleftarrow But pretend that the model were true

Ok so how we do perform this inference?

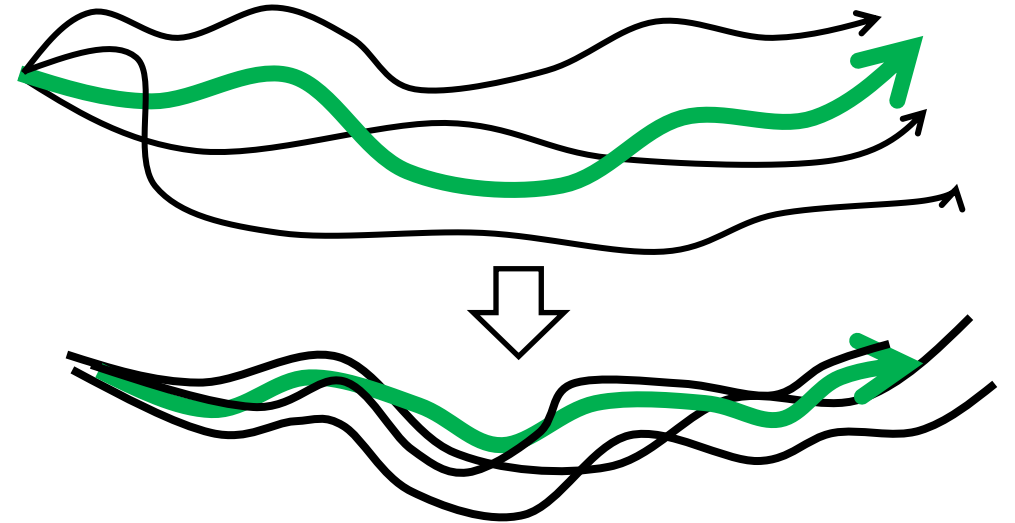
$$\hat{p}(a_t | s_t, \mathcal{O}_{t:T} = 1)$$

Idea 1: Variational inference in \hat{p}



Model-based policy optimization methods (Dyna ++)

Idea 2: Use Monte-Carlo Sampling for Inference



MPPI-style planning methods

Equivalence between posterior inference and MPPI

$$\hat{p}(a_t | s_t, \mathcal{O}_{t:T} = 1)$$

Let's expand out the nasty integrals with Bayes rule

$$= \frac{\hat{p}(a_t, s_t, \mathcal{O}_{t:T} = 1)}{\hat{p}(s_t, \mathcal{O}_{t:T} = 1)}$$

$$= \frac{\int \int \cdots \int \hat{p}(a_t, s_t, a_{t+1}, s_{t+1}, \dots, a_T, s_T, \mathcal{O}_{t:T} = 1) ds_{t+1} da_{t+1} \dots ds_T da_T}{\hat{p}(s_t, \mathcal{O}_{t:T} = 1)}$$

$$\propto \int \int \cdots \int \hat{p}(a_t, s_t, a_{t+1}, s_{t+1}, \dots, a_T, s_T, \mathcal{O}_{t:T} = 1) ds_{t+1} da_{t+1} \dots ds_T da_T$$

Equivalence between posterior inference and MPPI

$$\hat{p}(a_t | s_t, \mathcal{O}_{t:T} = 1)$$

$$\propto \int \int \cdots \int \hat{p}(a_t, s_t, a_{t+1}, s_{t+1}, \dots, a_T, s_T, \mathcal{O}_{t:T} = 1) ds_{t+1} da_{t+1} \dots ds_T da_T$$

$$\propto \int \int \cdots \int \hat{p}(s_0) \Pi_t \left[\begin{array}{ccc} \text{Dynamics} & \text{Action prior} & \text{Optimality} \\ \hat{p}(s_{t+1} | s_t, a_t) p(a_t | s_t) p(\mathcal{O}_t | s_t, a_t) \end{array} \right] ds_{t+1} da_{t+1} \dots ds_T da_T$$

$$\propto \int \int \cdots \int \hat{p}(s_0) \Pi_t \left[\hat{p}(s_{t+1} | s_t, a_t) p(a_t | s_t) \right] \exp \left[\sum_t r(s_t, a_t) \right] ds_{t+1} da_{t+1} \dots ds_T da_T$$

Substituting optimality definition $p(\mathcal{O}_t | s_t, a_t) = \exp(r(s_t, a_t))$

$$\propto \mathbb{E}_{\substack{s_0 \sim \hat{p}(s_0) \\ a_t \sim \hat{p}(a_t | s_t) \\ s_{t+1} \sim \hat{p}(s_{t+1} | s_t, a_t)}} \left[\exp \left[\sum_t r(s_t, a_t) \right] \right] \quad \text{Just using definition of expectation}$$

Equivalence between posterior inference and MPPI

$$\hat{p}(a_t | s_t, \mathcal{O}_{t:T} = 1)$$

$$\propto \mathbb{E}_{\substack{s_0 \sim \hat{p}(s_0) \\ a_t \sim \hat{p}(a_t | s_t) \\ s_{t+1} \sim \hat{p}(s_{t+1} | s_t, a_t)}} \left[\exp \left[\sum_t r(s_t, a_t) \right] \right]$$

Taking a bunch of samples through model \rightarrow

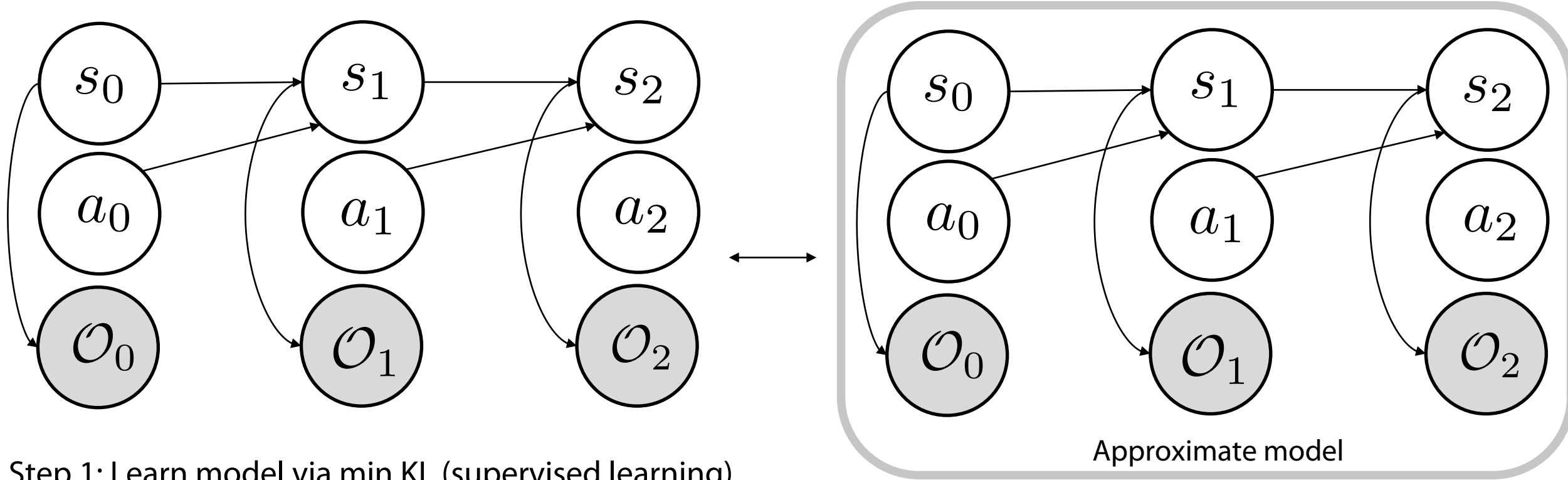
choose actions proportional to the expected sum of rewards

Can keep repeating with updated action prior

$$\propto \mathbb{E}_{\substack{s_0 \sim \hat{p}(s_0) \\ a_t \sim \hat{p}(a_t | s_t) \\ s_{t+1} \sim \hat{p}(s_{t+1} | s_t, a_t)}} \left[\exp \left[\sum_t r(s_t, a_t) \right] \right]$$

Can be thought of as a sampling-based Monte-Carlo approximation to posterior

Ok so what did we show?



Step 1: Learn model via min KL (supervised learning)

$$\max_{\hat{p}} \mathbb{E}_{p(s_0, \dots, s_T, a_0, \dots, a_T, \mathcal{O}_0, \dots, \mathcal{O}_T)} \left[\log \hat{p}(s_0) + \sum_t [\log \hat{p}(s_{t+1} | s_t, a_t) + \log \hat{p}(\mathcal{O}_t | s_t, a_t)] \right]$$

Step 2: Obtain posterior actions via Monte-Carlo approximation (approx MPPI)

$$\propto \mathbb{E}_{\substack{s_0 \sim \hat{p}(s_0) \\ a_t \sim \hat{p}(a_t | s_t) \\ s_{t+1} \sim \hat{p}(s_{t+1} | s_t, a_t)}} \left[\exp \left[\sum_t r(s_t, a_t) \right] \right]$$

Lecture outline

Recap



Control as Inference to Derive Q-learning

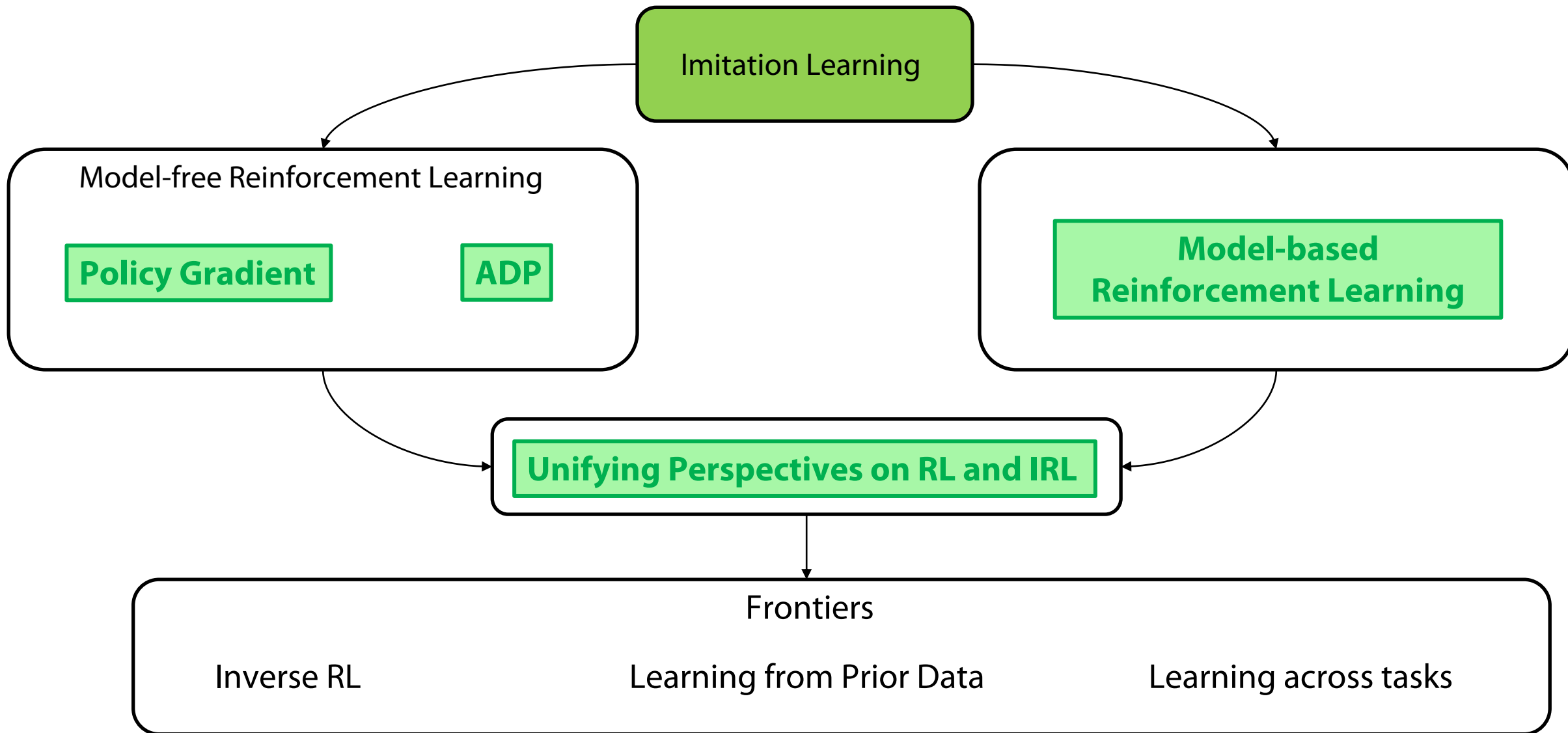


Control as Inference to Derive Model-Based RL



Inverse RL

Class Structure



Lecture Outline

Why Imitation? + Problem formulation



IRLv1 – max margin planning



IRLv2 – max entropy IRL

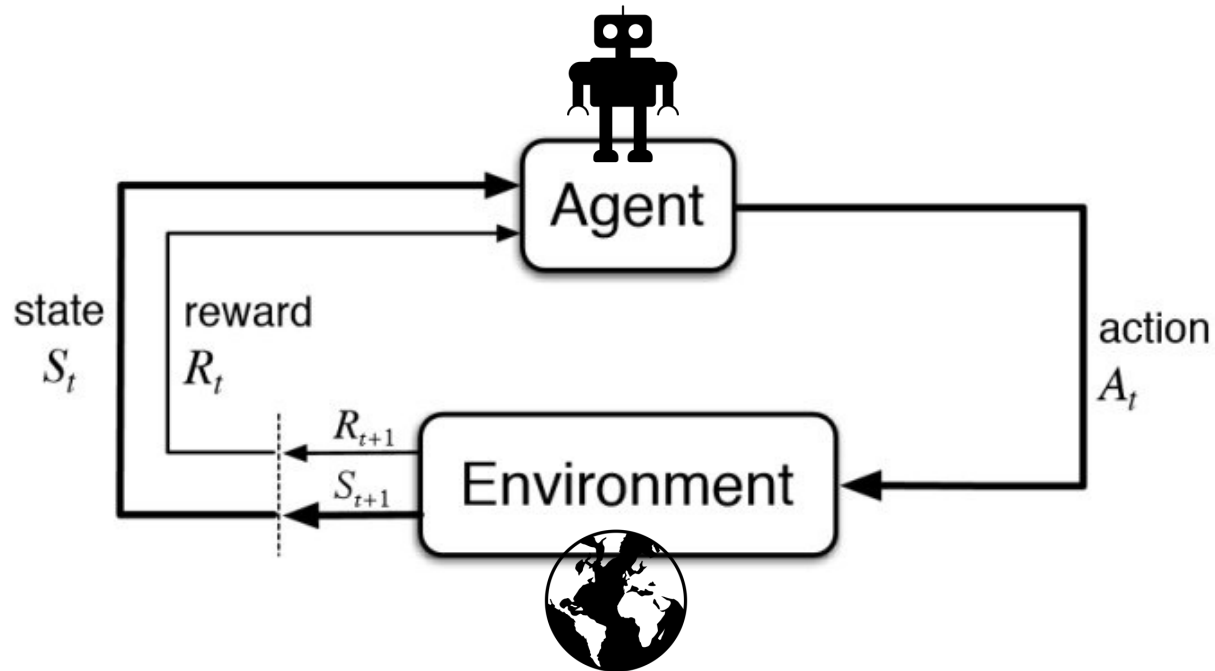


IRLv3 – partial policy optimization



IRLv4 – adversarial IRL

Let's revisit the premise of reinforcement learning



We studied a bunch of different algorithms to solve this

Model-based RL

Policy gradients

$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T r(s_t, a_t) \right]$$

or

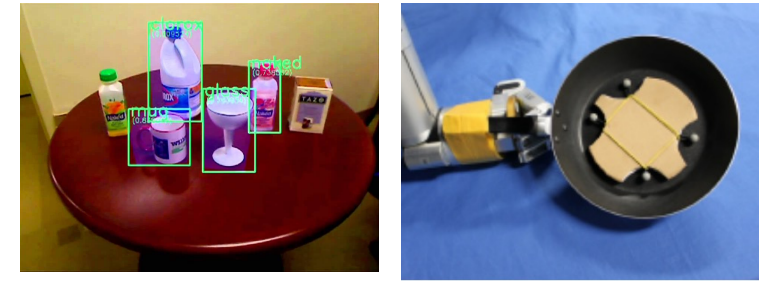
$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}(q(\cdot | s_t)) \right]$$

Actor-critic

But they all operate under the same assumption:
reward is known!

Reinforcement Learning requires Task Specification

Manual state estimation/perception



Complex reward specification

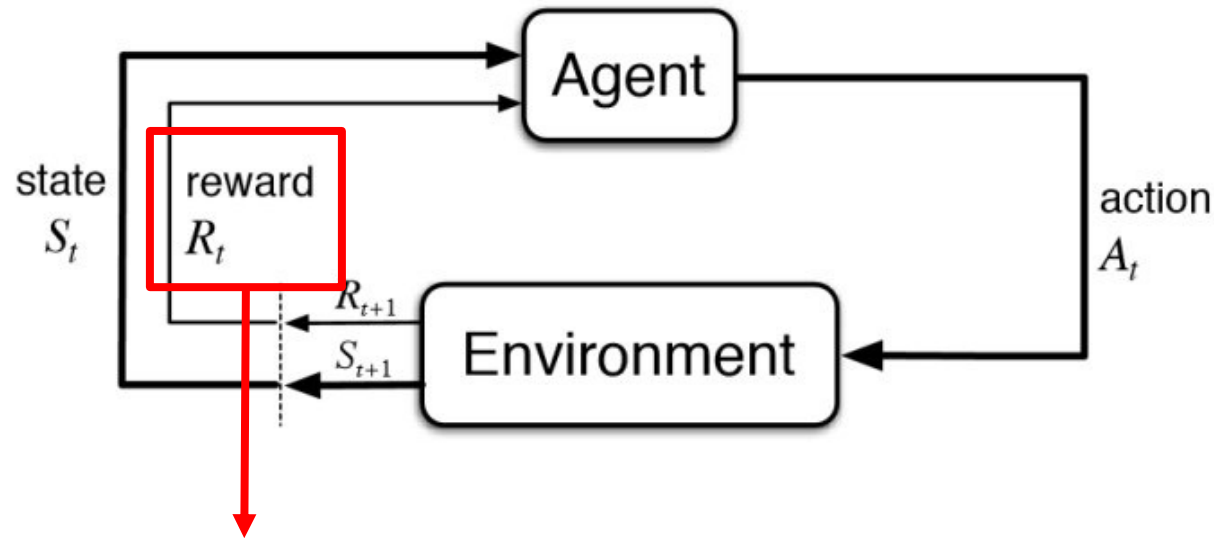
Name	Reward	Heroes	Description
Win	5	Team	
Hero Death	-1	Solo	
Courier Death	-2	Team	
XP Gained	0.002	Solo	
Gold Gained	0.006	Solo	For each unit of gold gained. Reward is not lost when the gold is spent or lost.
Gold Spent	0.0006	Solo	Per unit of gold spent on items without using courier.
Health Changed	2	Solo	Measured as a fraction of hero's max health. [‡]
Mana Changed	0.75	Solo	Measured as a fraction of hero's max mana.
Killed Hero	-0.6	Solo	For killing an enemy hero. The gold and experience reward is very high, so this reduces the total reward for killing enemies.
Last Hit	-0.16	Solo	The gold and experience reward is very high, so this reduces the total reward for last hit to ~ 0.4.
Deny	0.15	Solo	
Gained Aegis	5	Team	
Ancient HP Change	5	Team	Measured as a fraction of ancient's max health.
Megas Unlocked	4	Team	
T1 Tower*	2.25	Team	
T2 Tower*	3	Team	
T3 Tower*	4.5	Team	
T4 Tower*	2.25	Team	
Shrine*	2.25	Team	
Barracks*	6	Team	
Lane Assign [‡]	-0.15	Solo	Per second in wrong lane.

* For buildings, two-thirds of the reward is earned linearly as the building loses health, and one-third is earned as a lump sum when it dies.

[‡] See item O.2.

[‡] Hero's health is quartically interpolated between 0 (dead) and 1 (full health); health at fraction x of full health is worth $(x + 1 - (1 - x)^4) / 2$. This function was not tuned; it was set once and then untouched for the duration of the project.

Table 6: Shaped Reward Weights



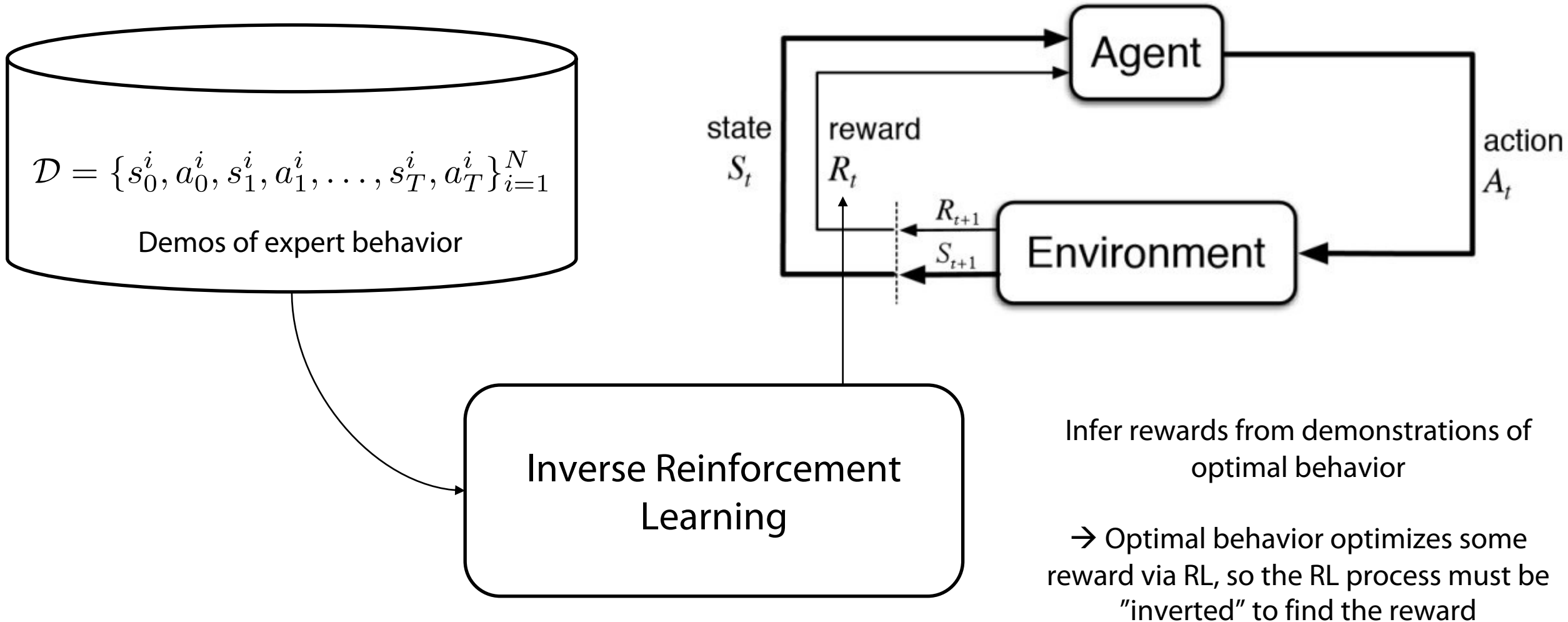
Does not magically appear in most settings

Has to be manually specified

→ can we do better?

Learning from Demonstrations

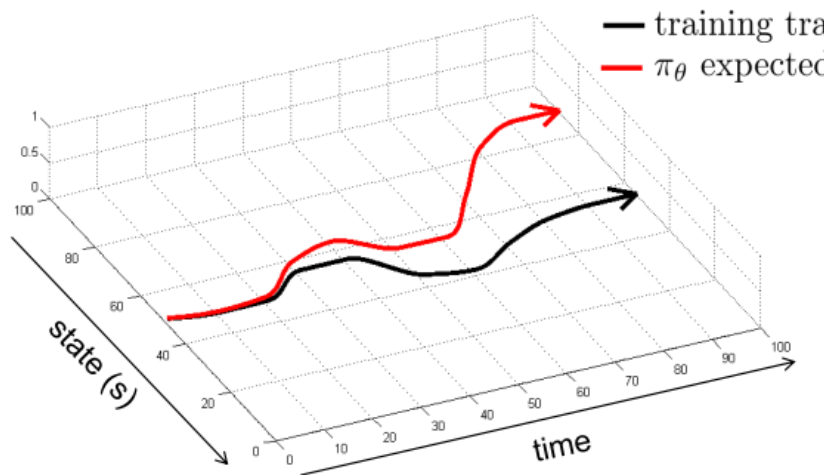
Avoid manual reward specification by learning from demos of optimal behavior



But haven't we already learned from demonstrations?

Imitation learning via Behavior Cloning (L2)

$$\arg \max_{\theta} \mathbb{E}_{(s^*, a^*) \sim \mathcal{D}} [\log \pi_{\theta}(a^* | s^*)]$$



Main difference between BC and IRL:

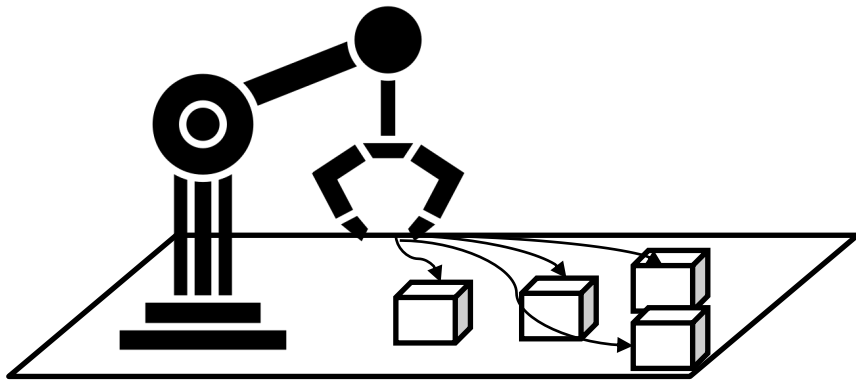
1. BC learns policies, IRL learns rewards
2. BC assumes no environment access, IRL typically assumes either known model or sampling access

Why does this matter?

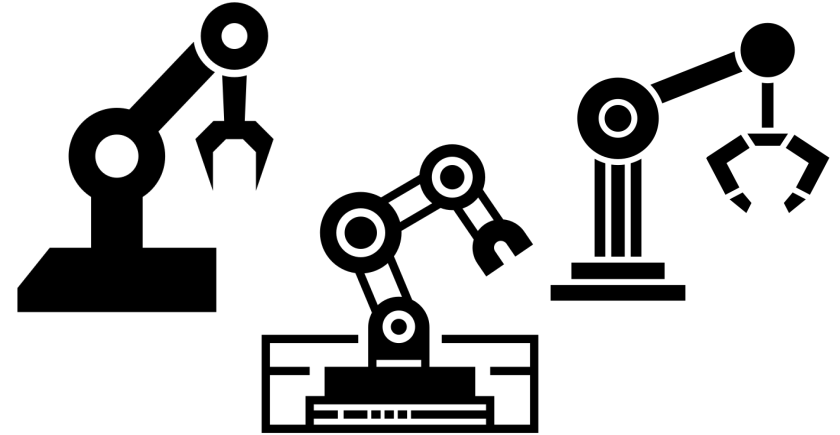
Zooming out – why do we care about imitation?

Imitation learning is all about generalization

Generalization across states



Generalization across dynamics

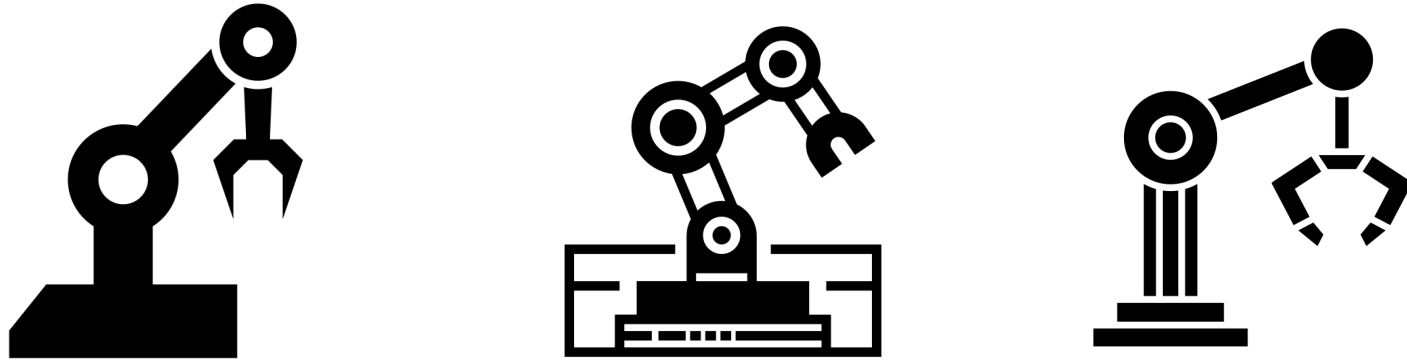


Covariate shift is just a manifestation of generalization

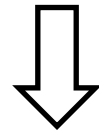
What if learning something else generalized better than policies?

Cross-Embodiment/Dynamics Transfer

Rewards may allow for cross dynamics transfer



Can all share the same reward, even with different dynamics!



Policies and Q/V functions entangle dynamics, rewards do not

Addressing Compounding Error

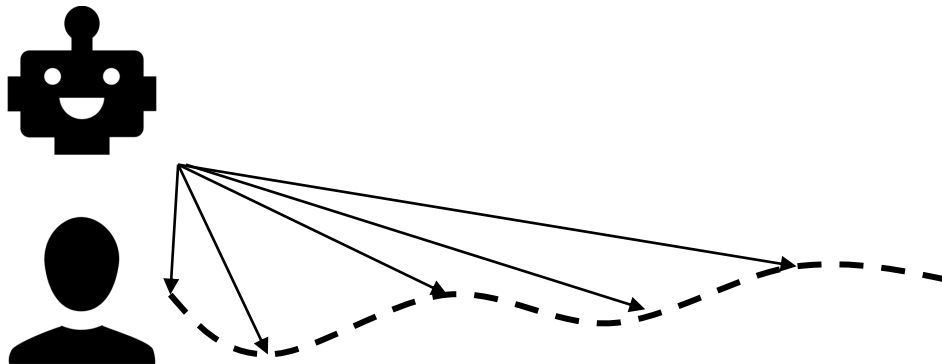
Reward can avoid covariate shift issues with forward KL

Imitation Learning via BC

$$\max_{\theta} \mathbb{E}_{(x,y) \sim \mathcal{D}} [\log \hat{p}_{\theta}(y|x)]$$

Sampling from expert

$$D_{\text{KL}}(p^* || p_{\theta})$$

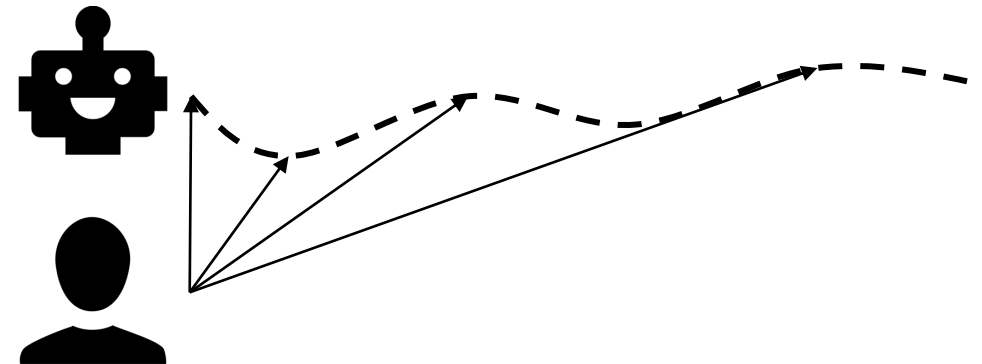


Reinforcement Learning with Inferred Reward

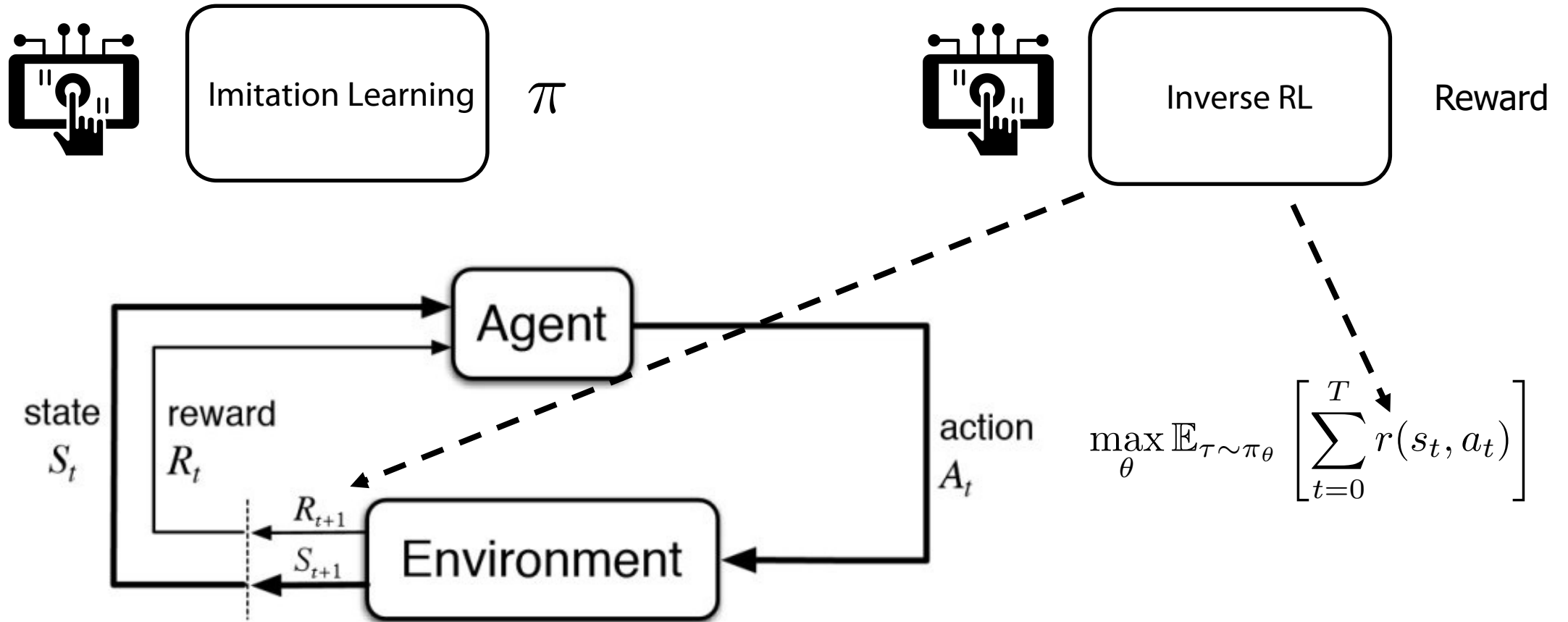
$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T r(s_t, a_t) \right]$$

Sampling from policy

What we care about $\longrightarrow D_{\text{KL}}(p_{\theta} || p^*)$



Learning Rewards from Human Data

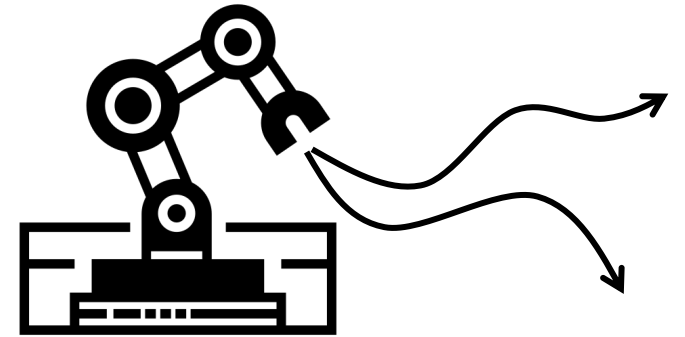
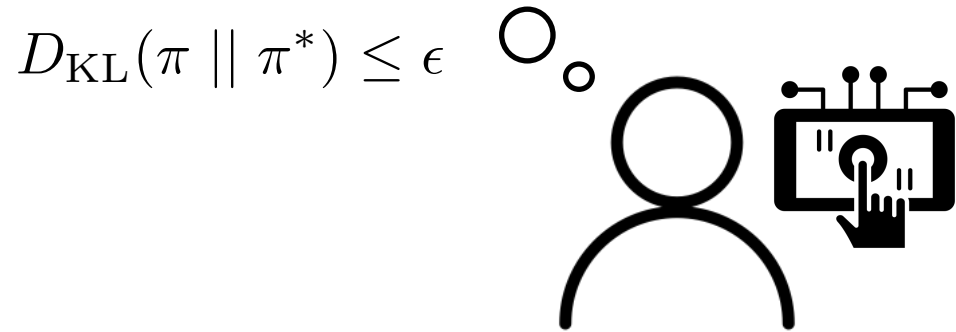


Is this even a well-defined problem?

How can we learn rewards?

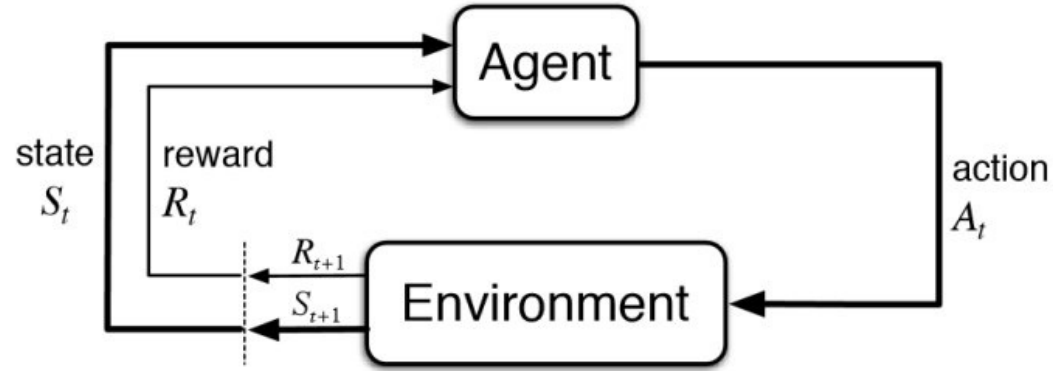
We must make more assumptions on the expert provided data

$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T r(s_t, a_t) \right]$$

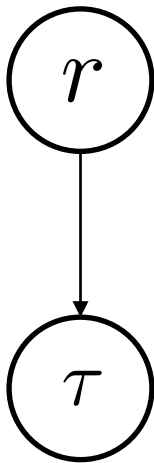


Experts are assumed to be “noisily” optimal

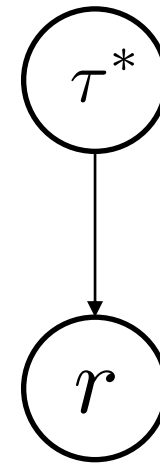
Why is this “inverse” reinforcement learning?



RL: Rewards generate trajectories



IRL: Expert trajectories generate rewards



Is this well defined?

IRL problem statement + assumptions

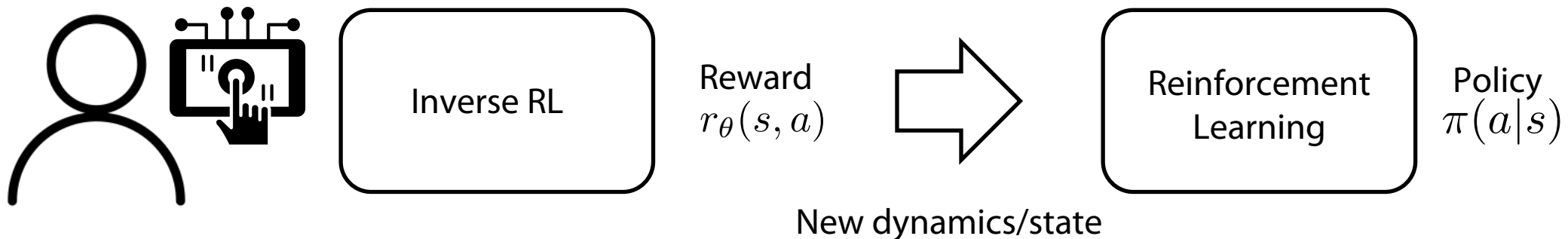
Reinforcement Learning

State: Known
Action: Known
Transition Dynamics: Unknown but can sample
Reward: **Known**
Expert policy: Unknown
Expert traces: **Unknown**

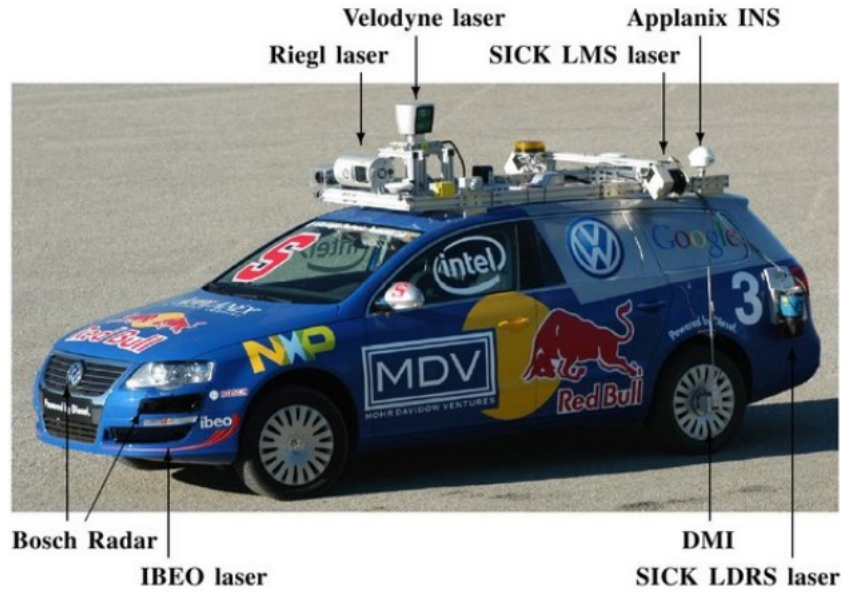
Inverse Reinforcement Learning

State: Known
Action: Known
Transition Dynamics: Unknown but can sample
Reward: **Unknown**
Expert policy: Unknown
Expert traces: **Known**

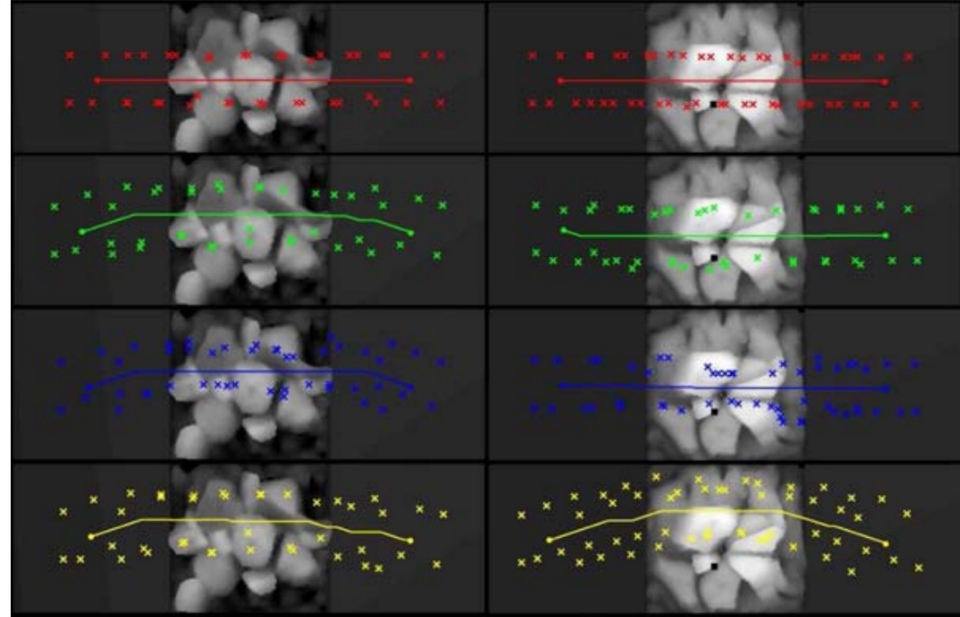
Find r that **explains** the demonstrator behavior as noisily optimal



Inverse RL Applications

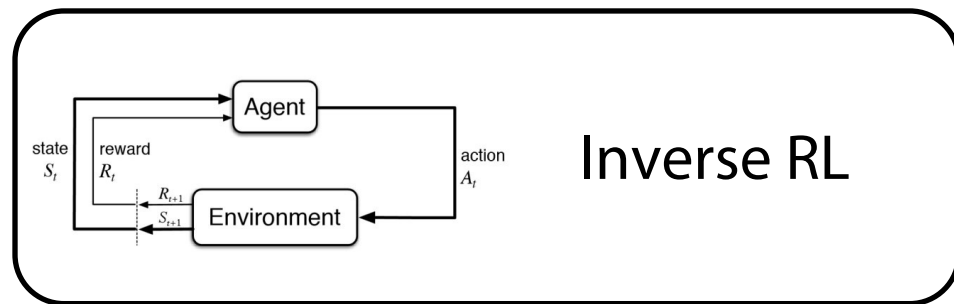
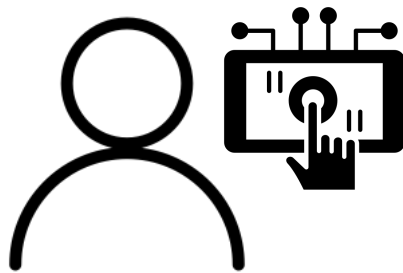


Inverse RL Applications



Why is this hard?

Find r that **explains** the demonstrator behavior as noisily optimal



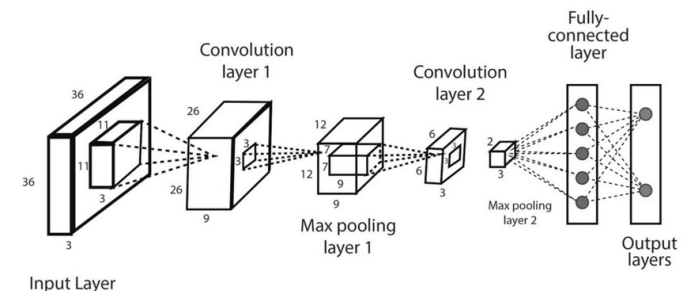
Challenging for a variety of reasons:

1. Inherently underspecified
2. R and π both unknown
3. Difficult optimization with T unknown.
4. Distributions/comparison metrics unknown

Reward Function

$$r_{\theta}(s, a)$$

Can be parameterized by arbitrary function approximator



Lecture Outline

Why Imitation? + Problem formulation



IRLv1 – max margin planning



IRLv2 – max entropy IRL

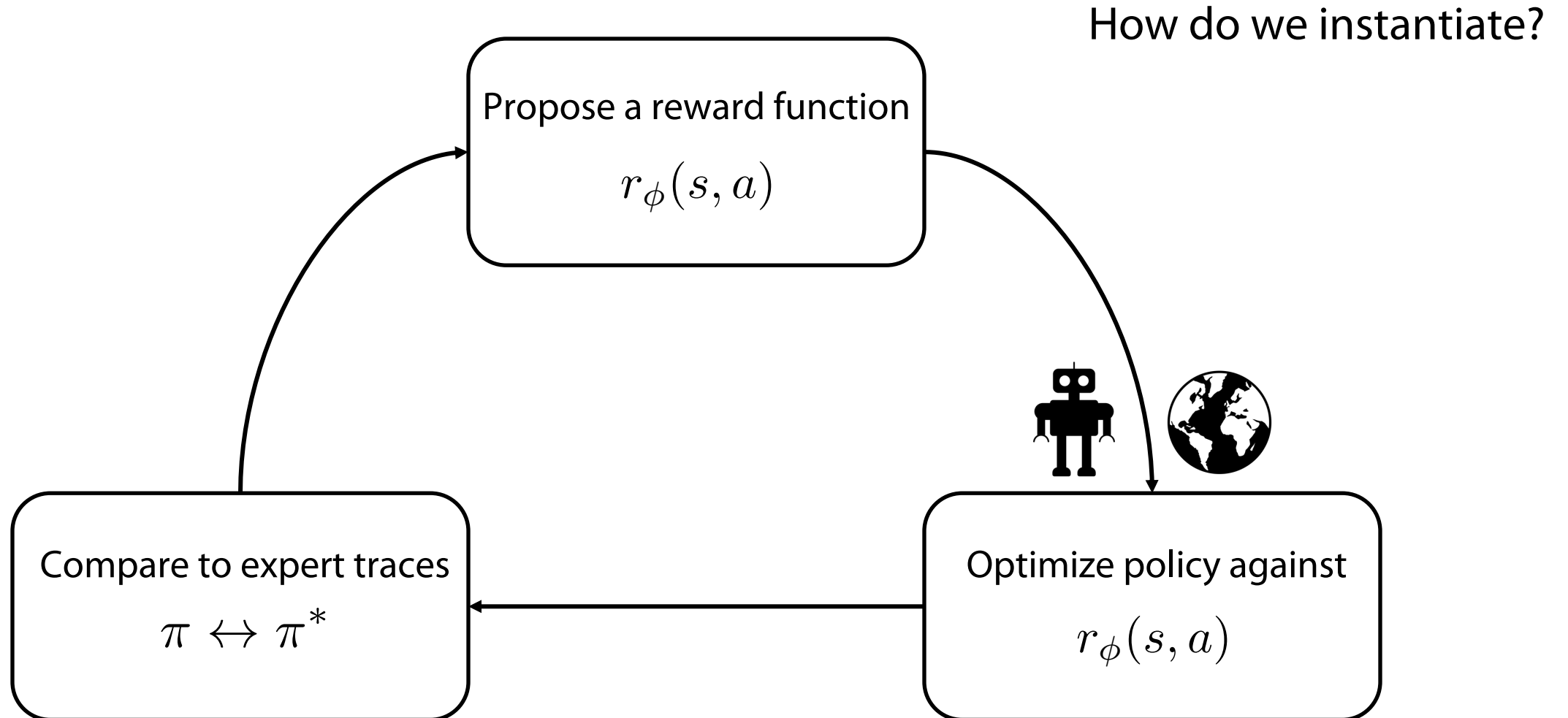


IRLv3 – partial policy optimization

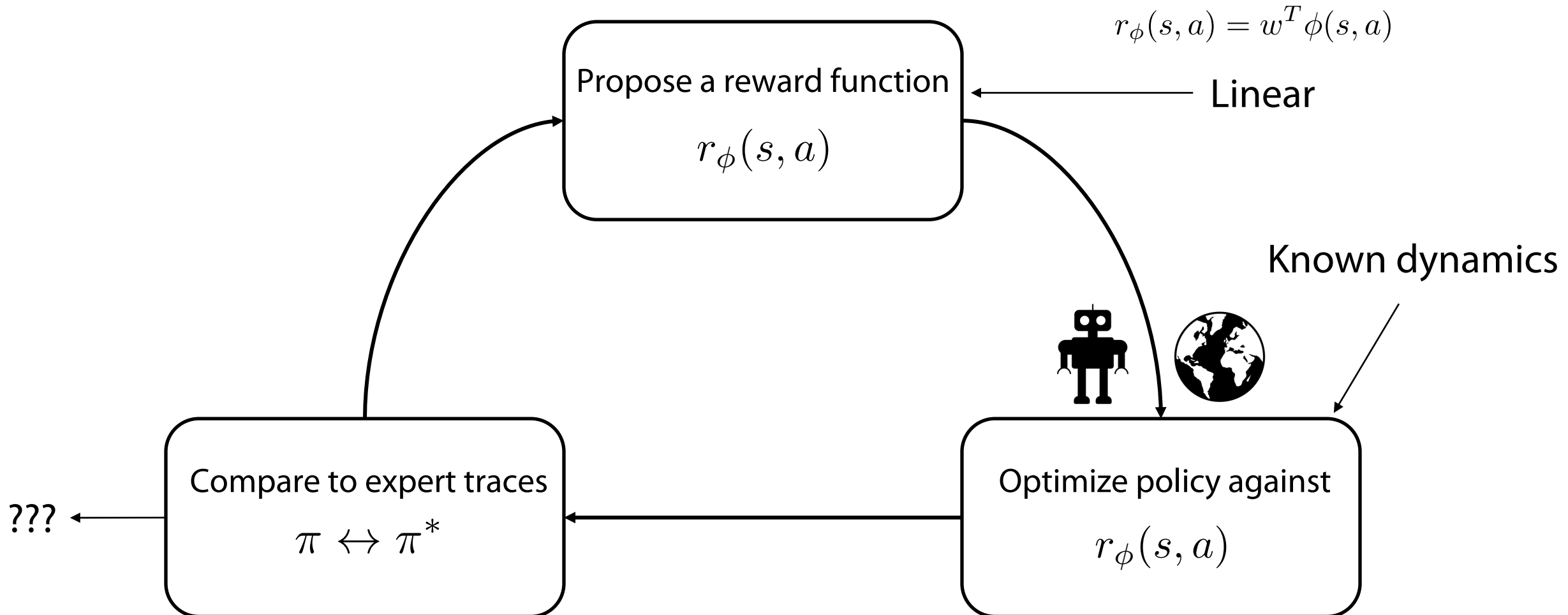


IRLv4 – adversarial IRL

A Formula for Inverse Reinforcement Learning



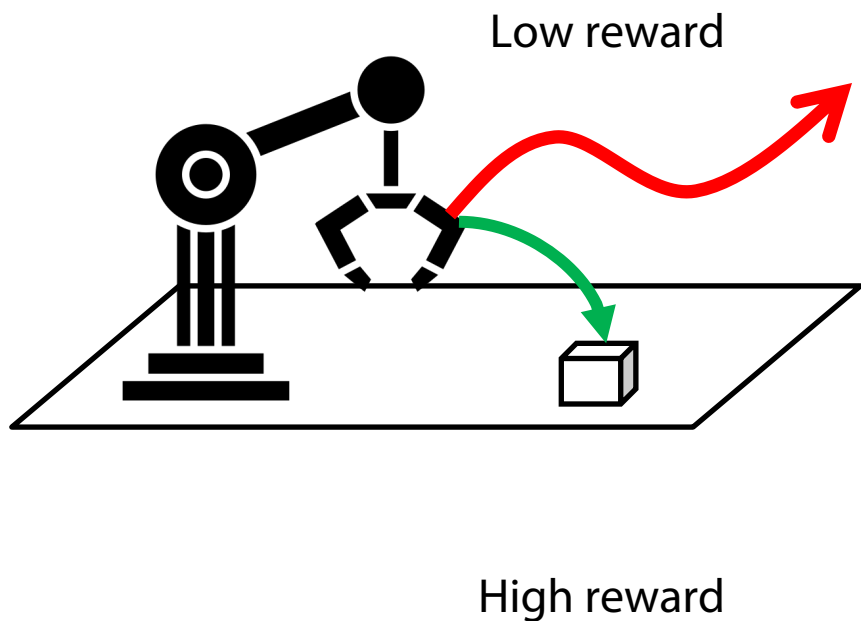
IRL v0 – Assumptions



IRL v0 – What is a good reward function?

A good reward would evaluate optimal data higher than all other data

$$V_r^{\pi^*}(s) \geq V_r^{\pi}(s) \quad \forall \pi, \forall s$$



Find w^* such that $r(s, a) = w^{*T} \phi(s, a)$

$$\mathbb{E}_{\pi^*} \left[\sum_t \gamma^t r(s_t, a_t) \right] \geq \mathbb{E}_{\pi} \left[\sum_t \gamma^t r(s_t, a_t) \right], \quad \forall \pi$$

$$\mathbb{E}_{\pi^*} \left[\sum_t \gamma^t w^{*T} \phi(s_t, a_t) \right] \geq \mathbb{E}_{\pi} \left[\sum_t \gamma^t w^{*T} \phi(s_t, a_t) \right], \quad \forall \pi$$

$$w^{*T} \mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \phi(s_t, a_t) \right] \geq w^{*T} \mathbb{E}_{\pi} \left[\sum_t \gamma^t \phi(s_t, a_t) \right], \quad \forall \pi$$

$$\mu(\pi^*, \phi)$$

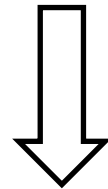
$$\mu(\pi, \phi)$$

Underdefined, $w^* = 0$ trivially satisfies!

IRL v0 – What is a good reward function?

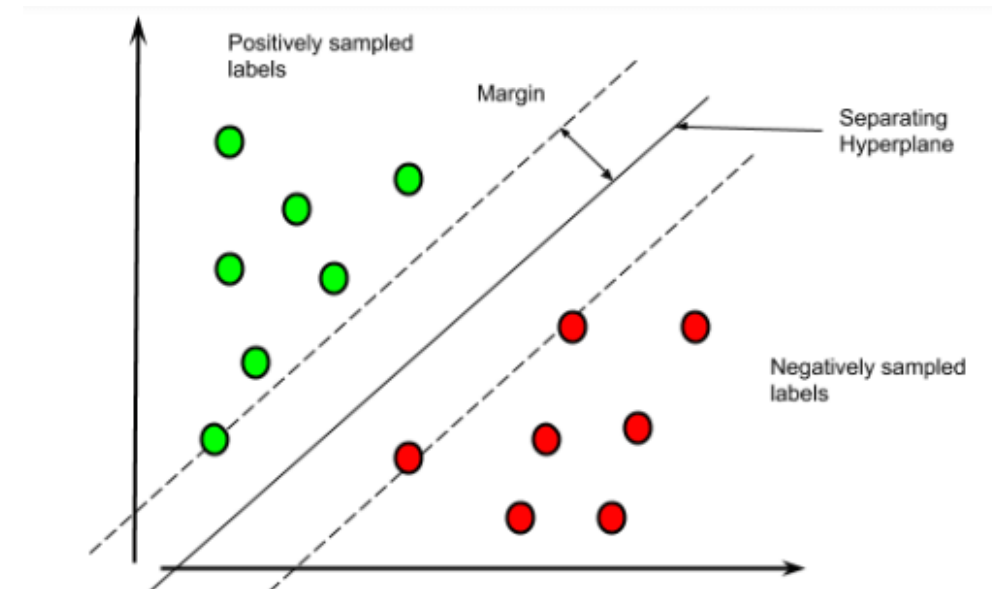
How do we tackle ambiguity?

$$w^{*T} \mathbb{E}_{\pi^*} [\phi(s, a)] \geq w^{*T} \mathbb{E}_{\pi} [\phi(s, a)] \quad \forall \pi, \forall s$$



$$\max_{w, m} m$$

$$\text{s.t. } w^T \mu^{\pi^*} \geq w^T \mu^{\pi} + m, \forall \pi \in \Pi$$



Find rewards which maximize the gap between the expert and all other policies

IRL v1 – Max Margin Feature Matching

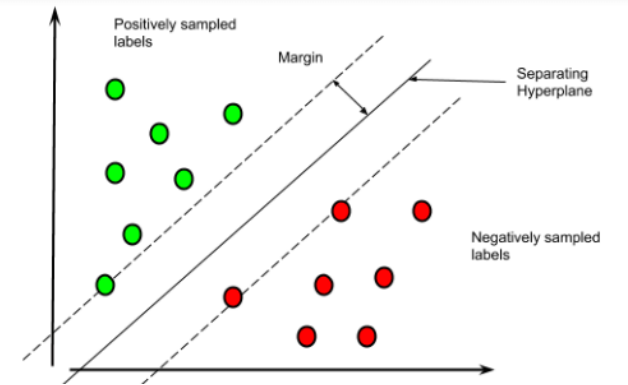
Choose w such that “margin” is maximized

$$\begin{aligned} \max m \\ \text{s.t. } w^T \mu^{\pi^*} \geq w^T \mu^{\pi} + m, \forall \pi \in \Pi \end{aligned}$$

Looks a lot like an SVM!



$$\begin{aligned} \min \|w\|_2 \\ \text{s.t. } w^T \mu^{\pi^*} \geq w^T \mu^{\pi} + 1, \forall \pi \in \Pi \end{aligned}$$



What might the issues be →

1. Uniform gap across all π, π^*
2. Noisily optimal may compromise the optimization

IRL v1 – (Fancy) Max Margin Feature Matching

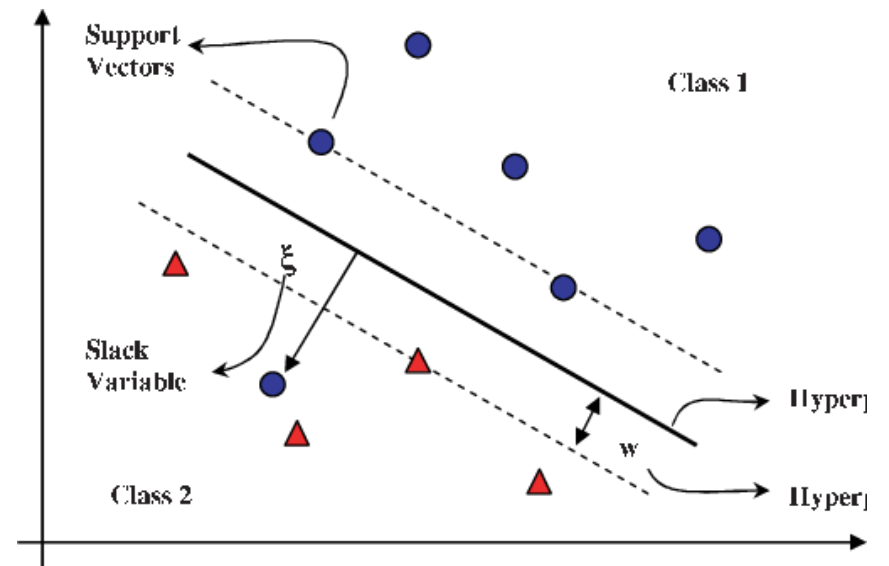
Maximum margin \rightarrow Structured Max-Margin + Slack

$$\begin{aligned} \min & \|w\|_2 \\ \text{s.t.} & w^T \mu^{\pi^*} \geq w^T \mu^{\pi} + 1, \forall \pi \in \Pi \end{aligned}$$

Bigger for more different policies

$$\begin{aligned} \min & \|w\|_2 + C\zeta \\ \text{s.t.} & w^T \mu^{\pi^*} \geq w^T \mu^{\pi} + D(\pi, \pi^*) - \zeta, \forall \pi \in \Pi \end{aligned}$$

Slack allows for noisy optimality

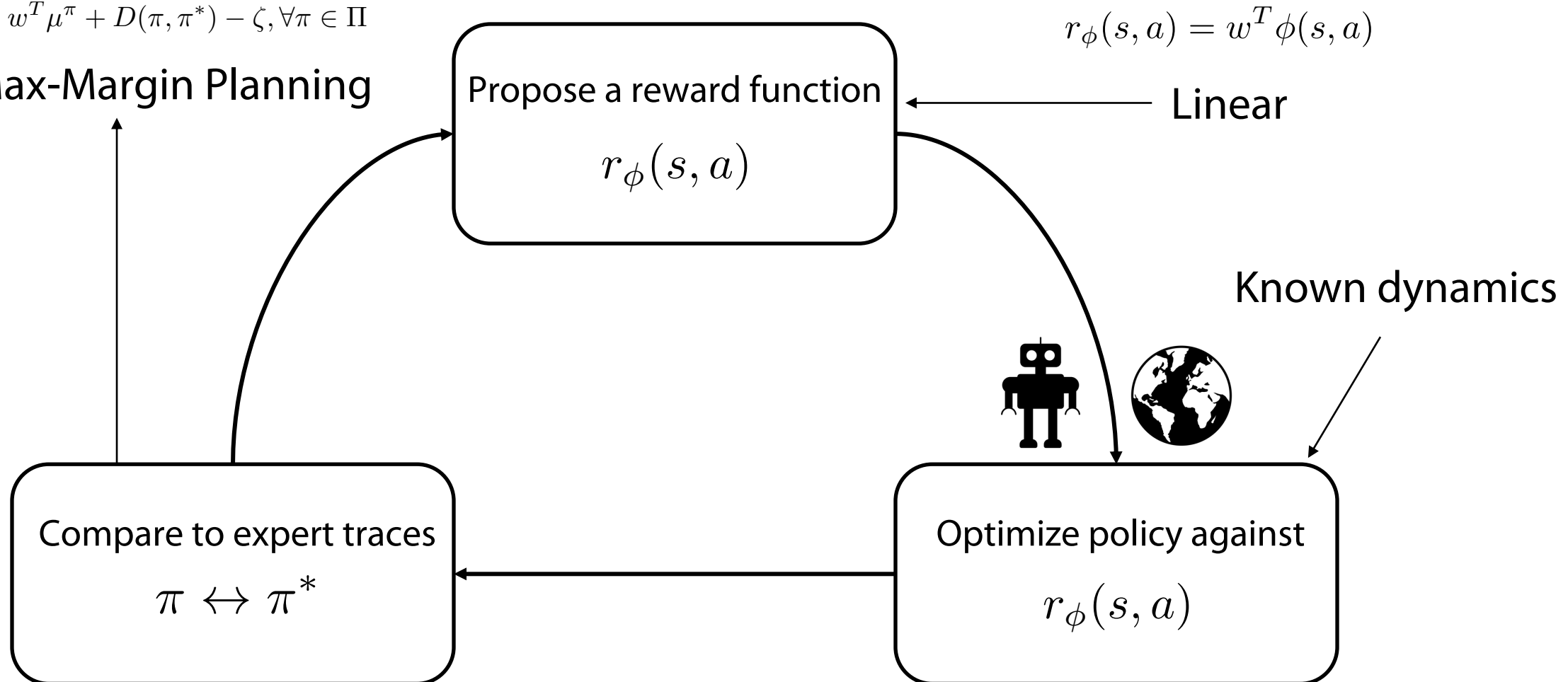


IRL v1 – Max Margin Feature Matching

$$\min \|w\|_2 + C\zeta$$

$$\text{s.t. } w^T \mu^{\pi^*} \geq w^T \mu^\pi + D(\pi, \pi^*) - \zeta, \forall \pi \in \Pi$$

Solve Max-Margin Planning



IRL v1 – Max Margin Feature Matching

1. Start with a random policy π_0

2. Find the w that optimizes

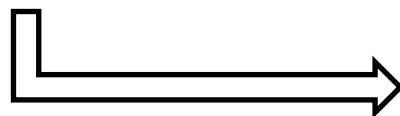
$$\min_{w, \zeta} \|w\|_2 + C\zeta$$

$$\text{s.t. } w^T \mu^{\pi^*} \geq w^T \mu^\pi + D(\pi, \pi^*) - \zeta, \forall \pi \in \{\pi_0, \pi_1, \dots, \pi_i\}$$

3. Solve for the optimal policy against $r_\phi(s, a) = w^{(i)T} \phi(s, a)$

$$\pi_{i+1} \rightarrow \text{Opt}(r_\phi(s, a), T)$$

4. Add to constraint set and repeat



Output the optimal reward function w^*

Max Margin Feature Matching in Action



Lecture Outline

Why Imitation? + Problem formulation



IRLv1 – max margin planning



IRLv2 – max entropy IRL



IRLv3 – partial policy optimization



IRLv4 – adversarial IRL

IRL v1 – Why this may not be enough?

$$\begin{aligned} \min \quad & \|w\|_2 + C\zeta \\ \text{s.t.} \quad & w^T \mu^{\pi^*} \geq w^T \mu^\pi + D(\pi, \pi^*) - \zeta, \forall \pi \in \Pi \end{aligned}$$

May not be able to deal with scenario where true margin is quite small for some policies

Not clear if this is a good way to deal with suboptimality

Constrained optimization is tough to optimize for non-linear functions

What if we had a "softer" notion of margin?

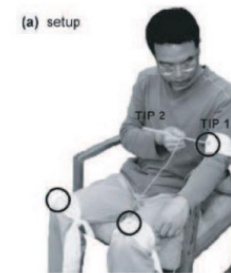
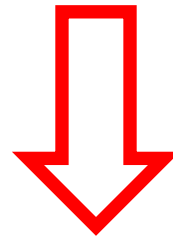
We have talked about “soft” optimality before!

We derived max-ent RL as maximum likelihood on optimality (lower bound) wrt policy

$$\max_q \mathbb{E}_{x \sim p(x)} \left[\mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) \right]$$

Control as inference

$$\mathbb{E}_{\substack{s_0 \sim p(s_0) \\ a_t \sim q(a_t | s_t) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t)}} \left[\sum_t \log p(\mathcal{O}_t | s_t, a_t) - \log q(a_t | s_t) \right]$$



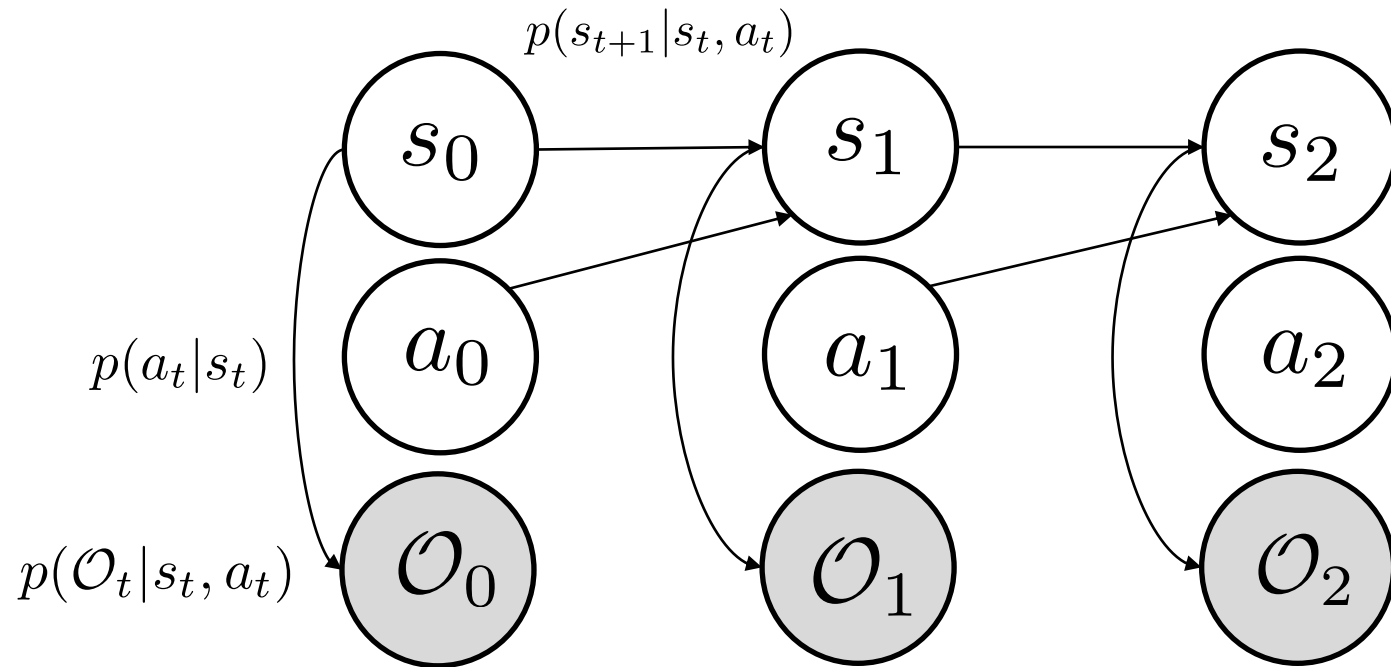
Li & Todorov '06



Ziebart '08

Can we invert this to do inverse RL with a softer notion of margin?

Let's revisit the graphical model



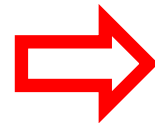
$$p(\mathcal{O}_t | s_t, a_t) = \exp(r_\phi(s_t, a_t))$$

$$p(\tau | \mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

$$p(\tau)$$

$$p(\tau | \mathcal{O}_{0:T} = 1)$$

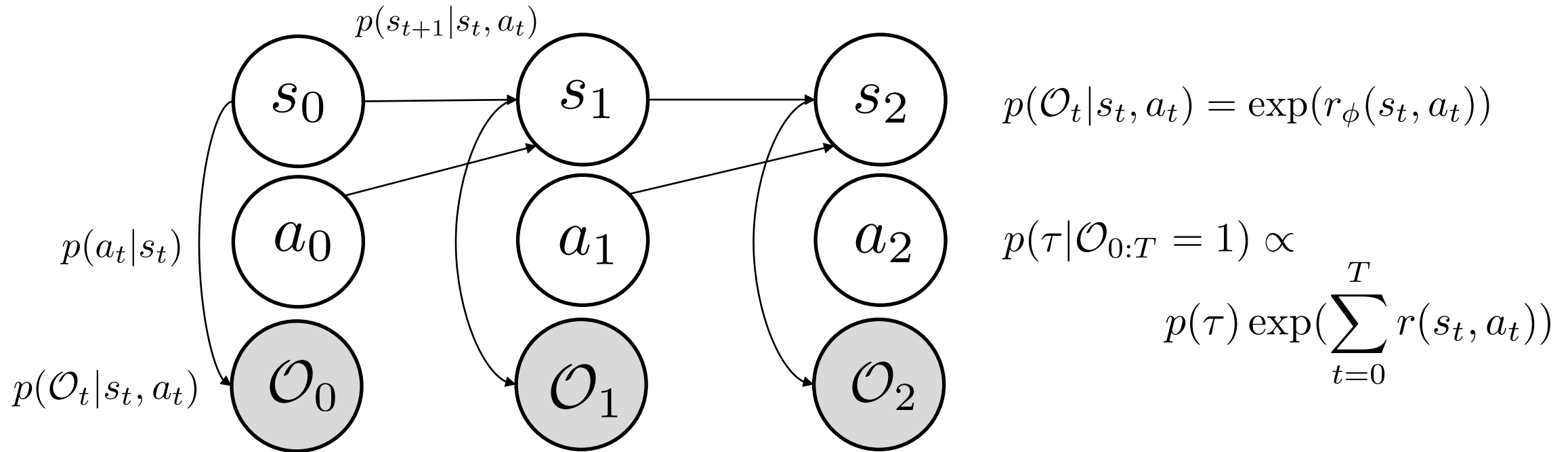
Uninformed behavior according to prior/dynamics



Soft optimal behavior conditioned on optimality

We were trying to find $p(a_t | s_t, \mathcal{O}_{t:T} = 1)$ given reward

IRLv2 – Maximum Entropy Inverse RL

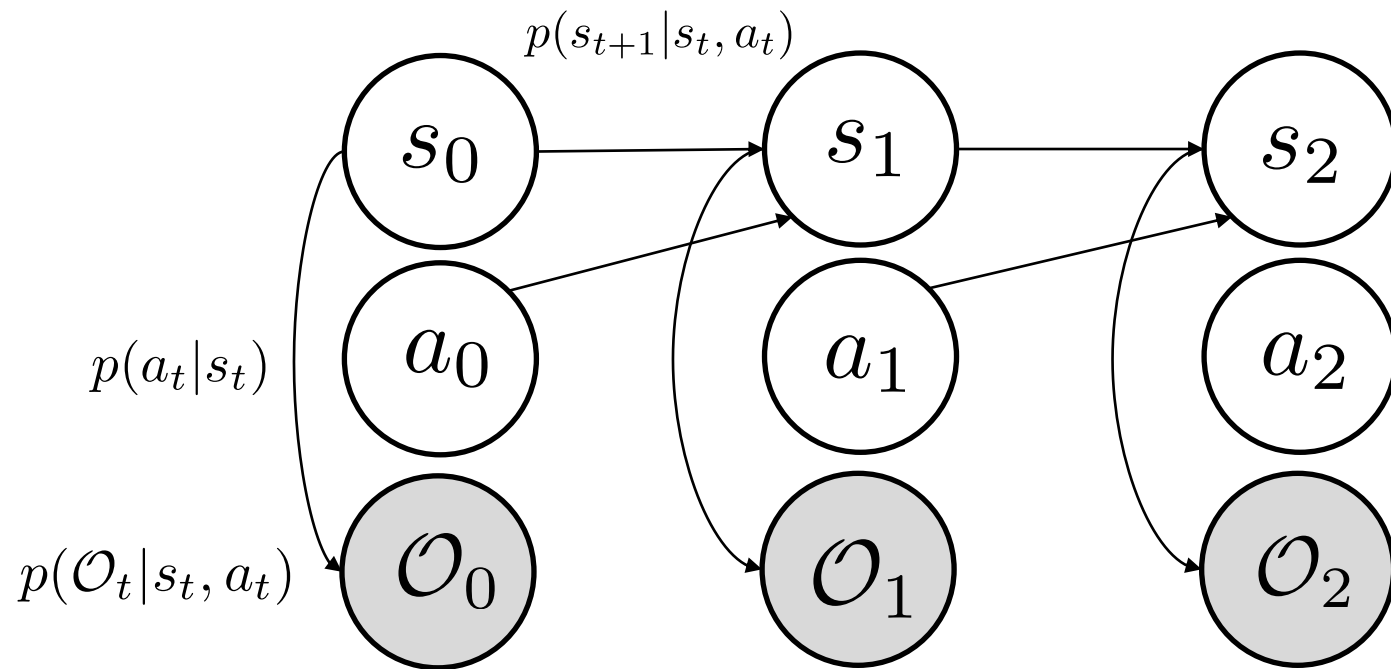


Now we are given (s, a) from optimal, we need to find the reward function that best explains the data

→ Maximum likelihood estimation!

(Find r , that maximizes the likelihood of (s, a) being produced on observed optimality)

Inverse RL in CAI graphical model



$$p(O_t|s_t, a_t) = \exp(r_\phi(s_t, a_t))$$

$$p(\tau|\mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

→ Maximum likelihood estimation!

(Find r , that maximizes the likelihood of (s, a) being produced on observed optimality)

$$\max_{\phi} \mathbb{E}_{\tau \sim \mathcal{D}^*} [\log p(\tau|\mathcal{O}_{0:T} = 1)] \quad (\text{Find optimality CPD that best explains observed data})$$

Maximum likelihood optimality estimation

$$p(\tau | \mathcal{O}_{0:T} = 1) \propto \cancel{p(\tau)} \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

Independent of reward

$$= \frac{\exp\left(\sum_{t=0}^T r(s_t, a_t)\right)}{\int \int p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right) ds_{0:T} da_{0:T}}$$

Hard to estimate – partition function (Z)



$$\max_{\phi} \mathbb{E}_{\tau \sim \mathcal{D}^*} [\log p(\tau | \mathcal{O}_{0:T} = 1)]$$

Difficult to compute analytically, but it's gradient has a nice form!

Maximum likelihood optimality estimation

$$p(\tau | \mathcal{O}_{0:T} = 1) = \frac{\exp(\sum_{t=0}^T r(s_t, a_t))}{\int \int p(\tau) \exp(\sum_{t=0}^T r(s_t, a_t)) ds_{0:T} da_{0:T}}$$

$$\begin{aligned} \max_{\phi} \mathbb{E}_{\tau \sim \mathcal{D}^*} [\log p(\tau | \mathcal{O}_{0:T} = 1)] \\ &= \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\log \left(\exp \left(\sum_{t=0}^T r_{\phi}(s_t, a_t) \right) \right) - \log Z \right] \\ &= \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T r_{\phi}(s_t, a_t) \right] - \log Z \end{aligned}$$

Easy to compute

Hard to compute

Let's take the gradient

$$\max_{\phi} \mathbb{E}_{\tau \sim \mathcal{D}^*} [\log p(\tau | \mathcal{O}_{0:T} = 1)]$$

$$\mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T r_{\phi}(s_t, a_t) \right] - \log Z$$

$$\nabla_{\phi} \mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T \nabla_{\phi} r_{\phi}(s_t, a_t) \right] - \nabla_{\phi} \log Z$$

$$\nabla_{\phi} \log Z = \frac{1}{Z} \nabla_{\phi} Z$$

$$Z = \int p(\tau) \exp(r(\tau)) d\tau$$

$$\nabla_{\phi} \mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T \nabla_{\phi} r_{\phi}(s_t, a_t) \right] - \frac{1}{Z} \int p(\tau) \exp(r_{\phi}(\tau)) \nabla_{\phi} r_{\phi}(\tau) d\tau$$

Notice this is exactly the soft optimality posterior

$$p(\tau | \mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

Let's take the gradient

$$\mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T r_\phi(s_t, a_t) \right] - \log Z$$

$$\nabla_\phi \mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T \nabla_\phi r_\phi(s_t, a_t) \right] - \frac{1}{Z} \int p(\tau) \exp(r_\phi(\tau)) \nabla_\phi r_\phi(\tau) d\tau$$

Notice this is exactly the soft optimality posterior

$$p(\tau | \mathcal{O}_{0:T} = 1) \propto p(\tau) \exp\left(\sum_{t=0}^T r(s_t, a_t)\right)$$

$$\nabla_\phi \mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T \nabla_\phi r_\phi(s_t, a_t) \right] - \mathbb{E}_{\tau \sim p(\tau | \mathcal{O}_{0:T}=1)} \left[\sum_{t=0}^T \nabla_\phi r_\phi(s_t, a_t) \right]$$

Push up gradients along experts

Push down gradients along soft optimal policy under current reward

Computable, with RL in the inner loop

IRLv2 – Maximum Entropy Inverse RL

$$\nabla_{\phi} \mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T \nabla_{\phi} r_{\phi}(s_t, a_t) \right] - \frac{1}{Z} \int p(\tau) \exp(r_{\phi}(\tau)) \nabla_{\phi} r_{\phi}(\tau) d\tau$$

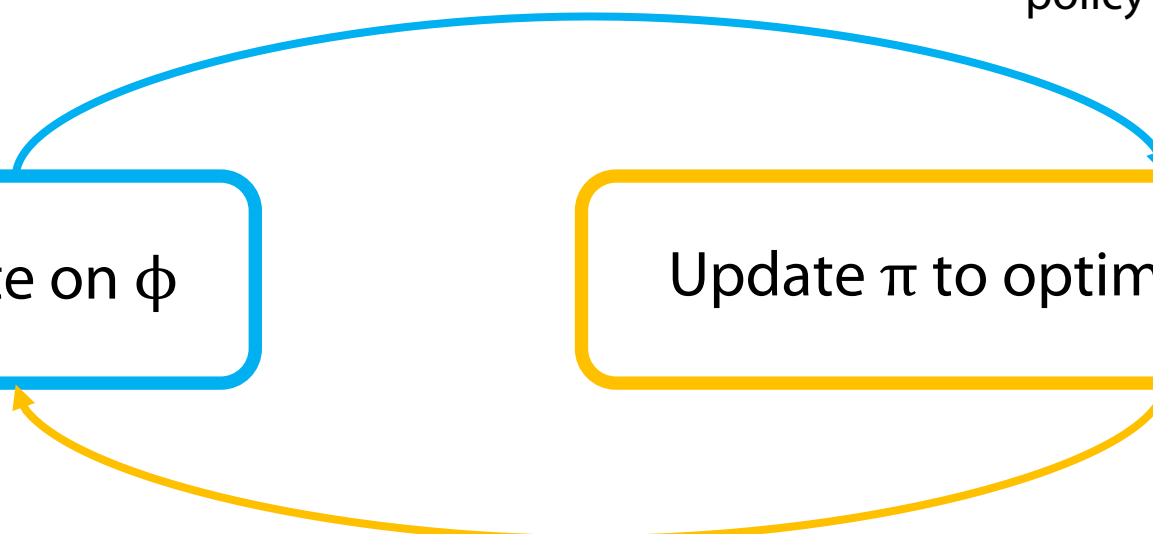
$$\nabla_{\phi} \mathcal{L}(\phi) = \mathbb{E}_{\tau \sim \mathcal{D}^*} \left[\sum_{t=0}^T \nabla_{\phi} r_{\phi}(s_t, a_t) \right] - \mathbb{E}_{\tau \sim p(\tau | \mathcal{O}_{0:T}=1)} \left[\sum_{t=0}^T \nabla_{\phi} r_{\phi}(s_t, a_t) \right]$$

Push up gradients along experts

Push down gradients along soft optimal policy under current reward

Update on ϕ

Update π to optimal using current r_{ϕ}

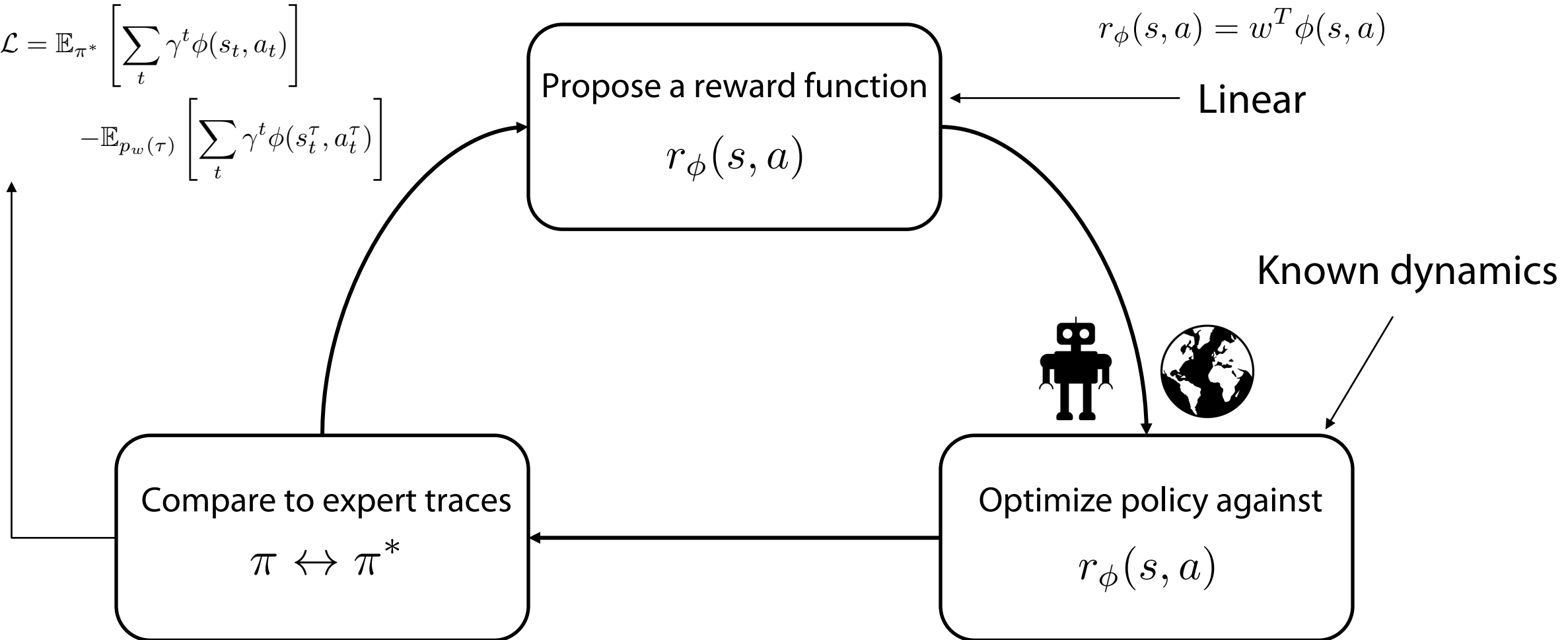


IRL v2 – Max-Ent IRL – Put it together

Maximum Entropy

$$\nabla_w \mathcal{L} = \mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \phi(s_t, a_t) \right]$$

$$- \mathbb{E}_{p_w(\tau)} \left[\sum_t \gamma^t \phi(s_t^\tau, a_t^\tau) \right]$$

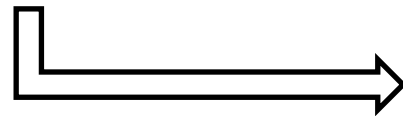


IRL v2 –Max-Entropy Inverse RL (Pseudocode)

1. Start with a random policy π_0 and weight vector w
2. Find the “soft” optimal policy under $w - p_w(\tau)$
3. Take a gradient step on w

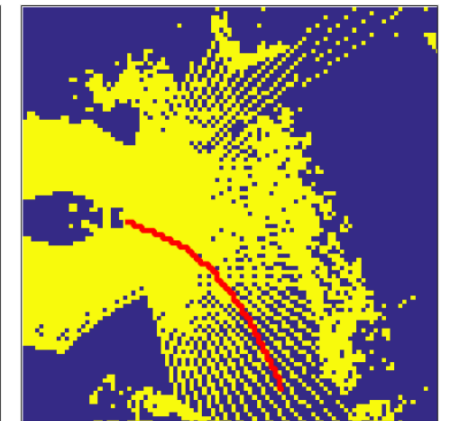
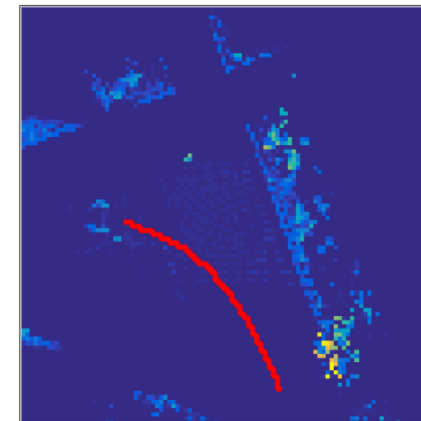
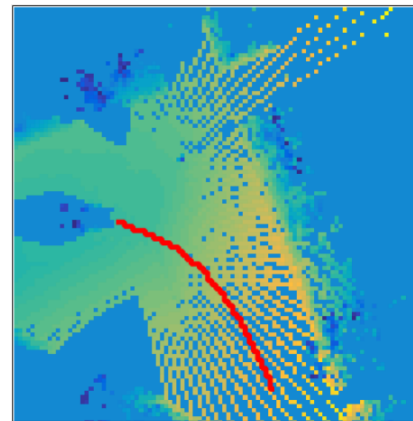
$$\nabla_w \mathcal{L} = \mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \phi(s_t, a_t) \right] - \mathbb{E}_{p_w(\tau)} \left[\sum_t \gamma^t \phi(s_t^\tau, a_t^\tau) \right]$$

4. Repeat



Output the optimal reward function w^*

Max-Ent IRL in Action



Lecture Outline

Why Imitation? + Problem formulation



IRLv1 – max margin planning



IRLv2 – max entropy IRL



IRLv3 – partial policy optimization



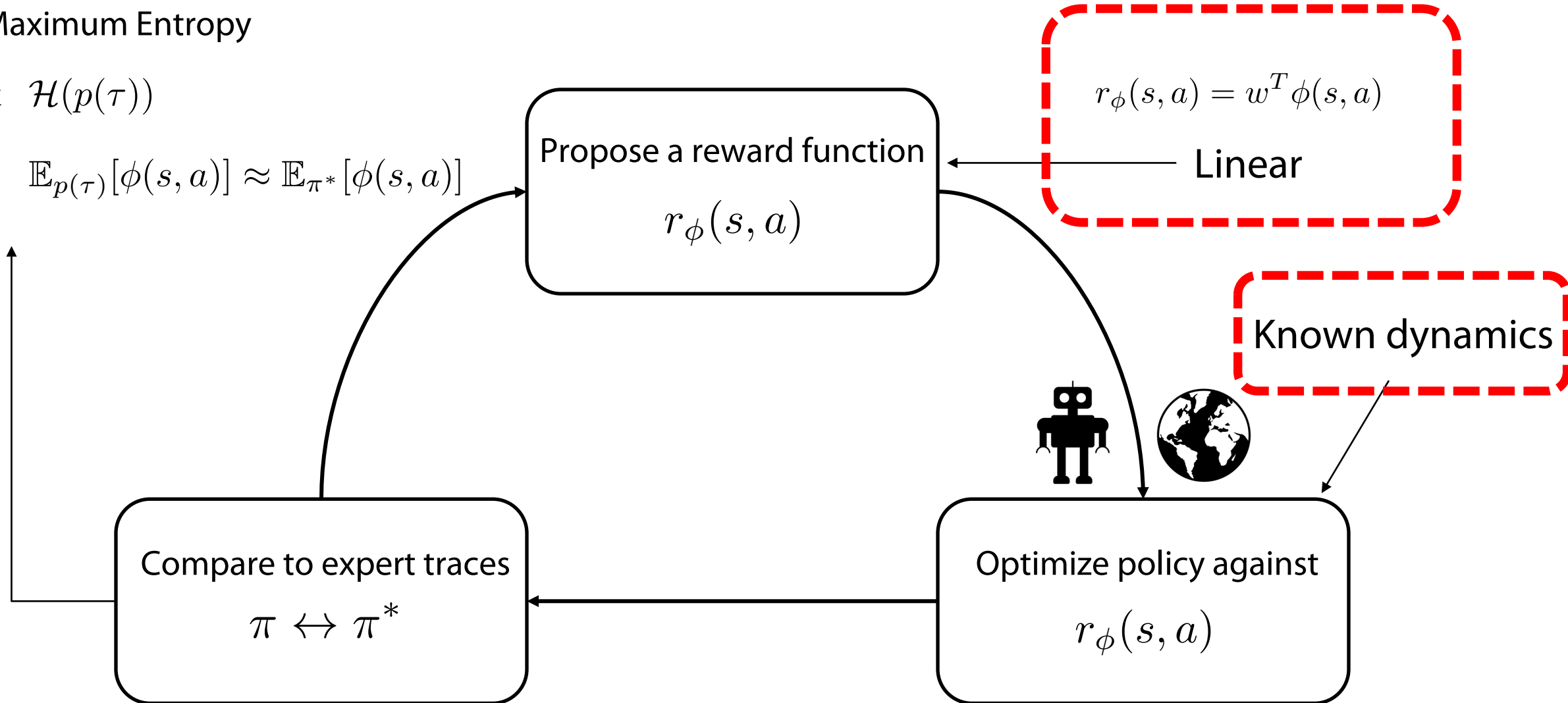
IRLv4 – adversarial IRL

Ok but no way this could work?

Maximum Entropy

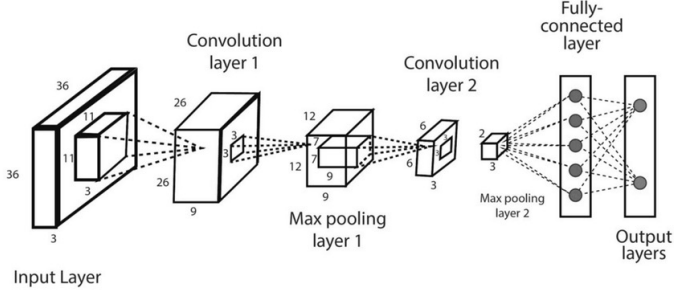
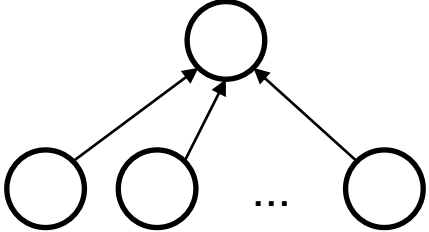
$$\max_{p(\tau)} \mathcal{H}(p(\tau))$$

$$\text{s.t. } \mathbb{E}_{p(\tau)}[\phi(s, a)] \approx \mathbb{E}_{\pi^*}[\phi(s, a)]$$



Linear Rewards → Neural Net Rewards

Max-ent IRL allows us to go from linear rewards to arbitrary neural network rewards



Linear Max-Ent IRL

$$\max_w \mathbb{E}_{\pi^*} \left[\sum_t w^T \gamma^t \phi(s_t, a_t) \right] - \log \int_{\tau} \left[\exp \left(\sum_t w^T \gamma^t \phi(s_t, a_t) \right) \right] d\tau$$



Non-Linear Max-Ent IRL

$$\max_{\theta} \mathbb{E}_{\pi^*} \left[\sum_t \gamma^t r_{\theta}(s_t, a_t) \right] - \log \int_{\tau} \left[\exp \left(\sum_t \gamma^t r_{\theta}(s_t, a_t) \right) \right] d\tau$$

Can simply replace, w with arbitrary θ and use autodiff!

Avoiding Complete Policy Optimization

Optimize policy against

$$r_\phi(s, a)$$

← Assumes dynamics are known so we can just do (fast) planning

What happens when dynamics are unknown!

$$\mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

← What if we only **improved** the policy a little bit

$$-\mathbb{E}_{p_w(\tau)} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

Biased!

Requires complete “soft” policy optimization

Avoiding Complete Policy Optimization

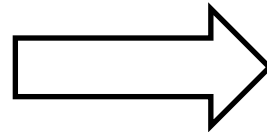
Importance sampling to the rescue!

$$\mathbb{E}_{p(x)} [f(x)] = \mathbb{E}_{q(x)} \left[\frac{p(x)}{q(x)} f(x) \right]$$

$$\mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

$$- \mathbb{E}_{p_w(\tau)} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

Importance
Sampling



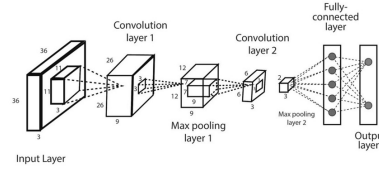
$$\mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

$$- \mathbb{E}_q \left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

$$\frac{\exp(\sum_t r_{\theta}(s_t, a_t))}{\prod_t \pi_{\theta}(a_t | s_t)}$$

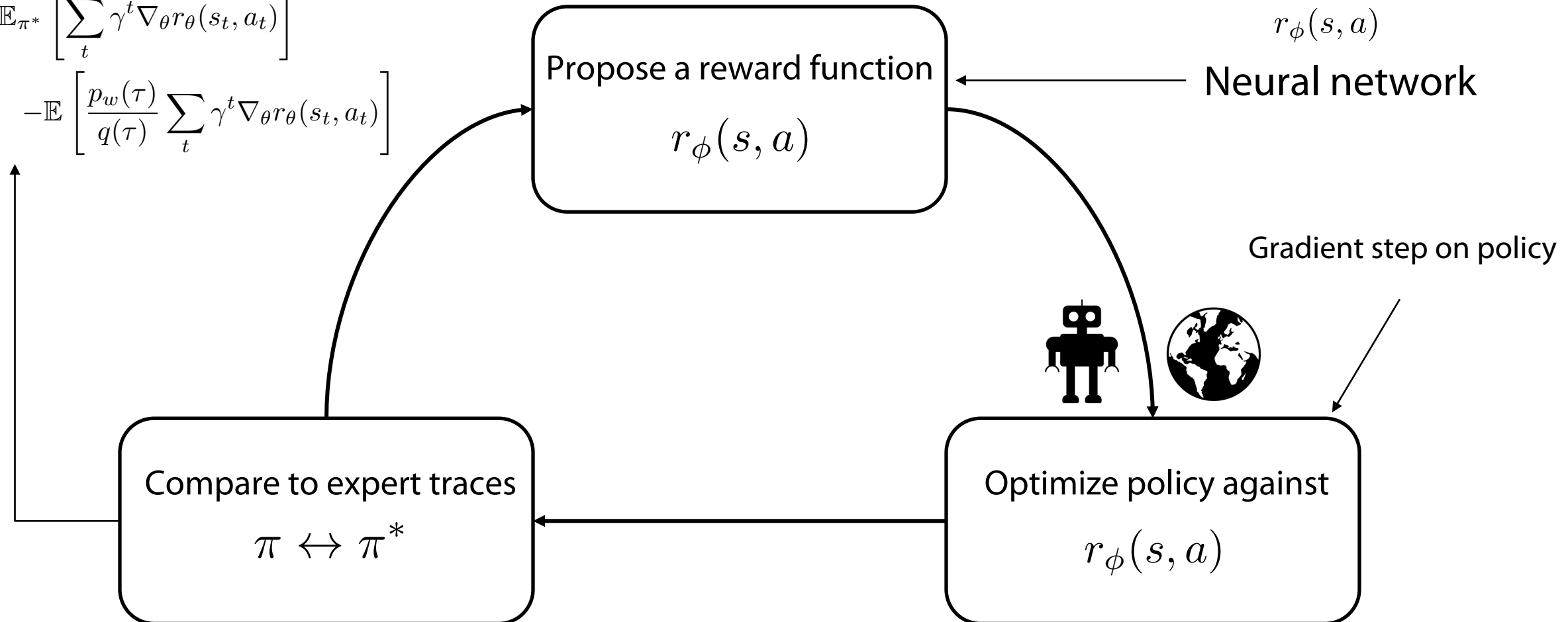
Can transfer significantly more from iteration to iteration rather than doing full nested optimization

IRLv4 – Guided Cost Learning

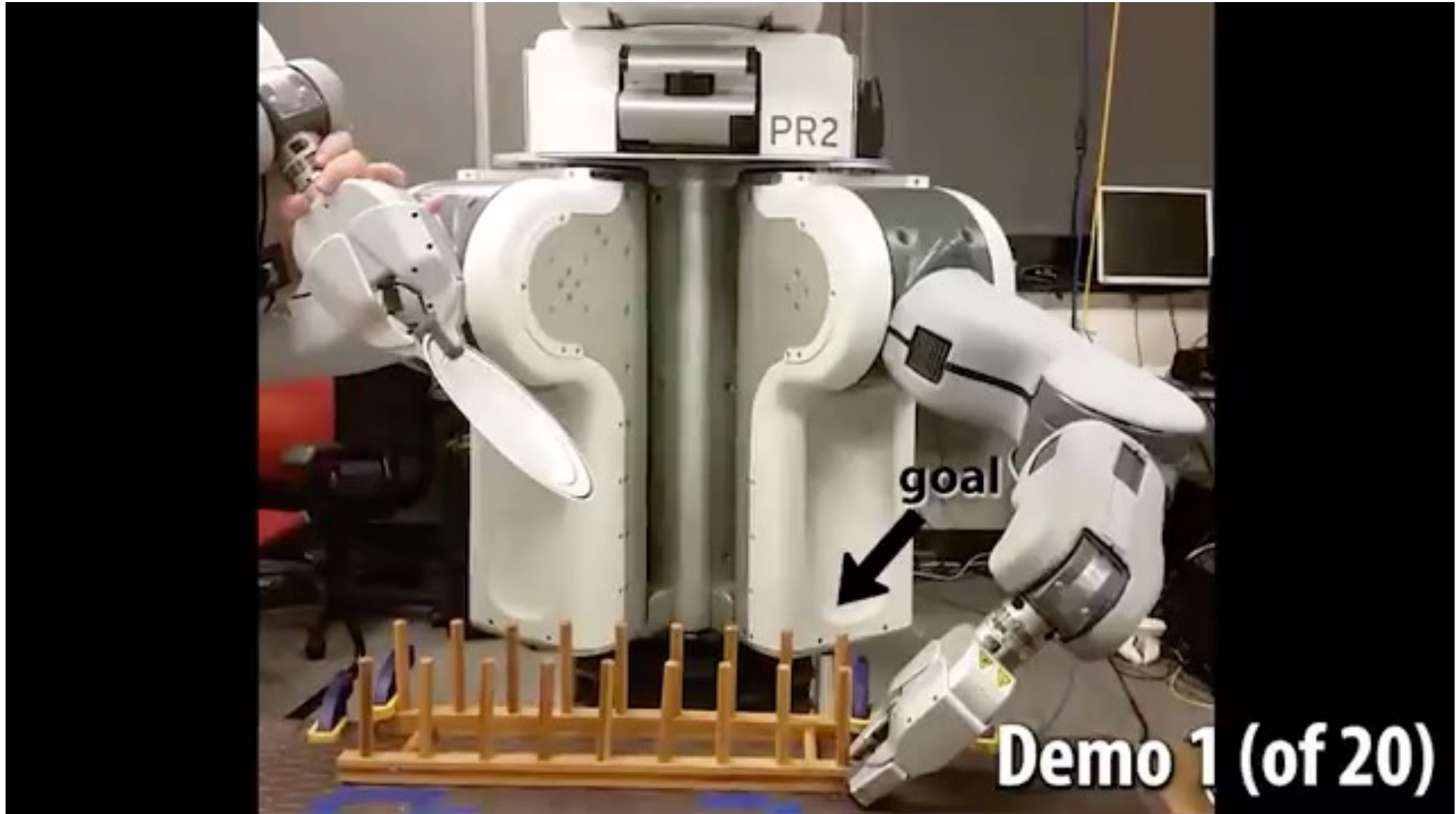


Gradient Step on Reward

$$\mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$
$$-\mathbb{E} \left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$



IRLv4 – Guided Cost Learning



Lecture Outline

Why Imitation? + Problem formulation



IRLv1 – max margin planning



IRLv2 – max entropy IRL



IRLv3 – partial policy optimization



IRLv4 – adversarial IRL

Connecting Maximum-Entropy RL to GANs

Looks like a game

1. Start with a random policy π_0 and weight vector w

2. Take a step on "soft" optimal policy under $w - p_w(\tau)$

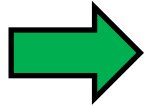
3. Take a gradient step on w

4. Repeat

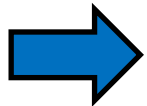
$$\nabla_{\theta} \mathcal{L} = \mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right] - \mathbb{E}_q \left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

Output the optimal reward function w^*

Generator



Discriminator



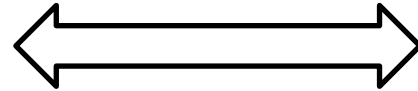
Recasting GAIL as an IRL method

For a particular parameterization of the discriminator, GAIL recovers a reward

Max-Ent Inverse RL

$$\mathbb{E}_{\pi^*} \left[\sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right] - \mathbb{E}_q \left[\frac{p_w(\tau)}{q(\tau)} \sum_t \gamma^t \nabla_{\theta} r_{\theta}(s_t, a_t) \right]$$

Push up demos, push down policy



With some massaging

GAIL

$$\mathbb{E}_{\pi^*} [D_{\psi}(\tau)] + \mathbb{E}_{\pi_{\theta}} [(1 - D_{\psi}(\tau))]$$

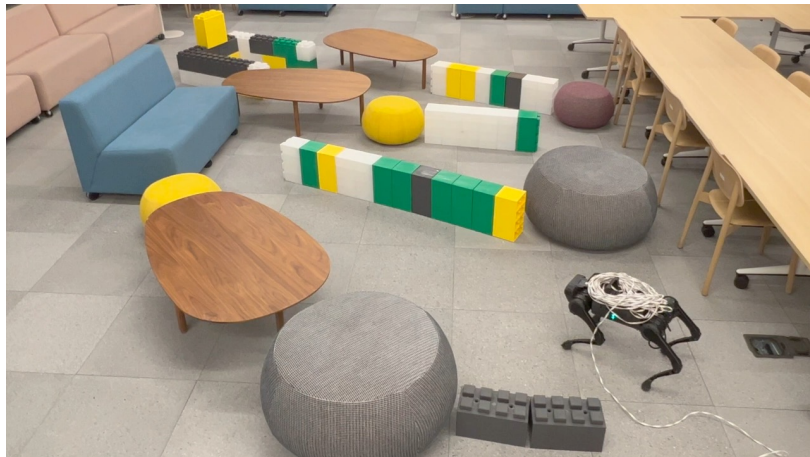
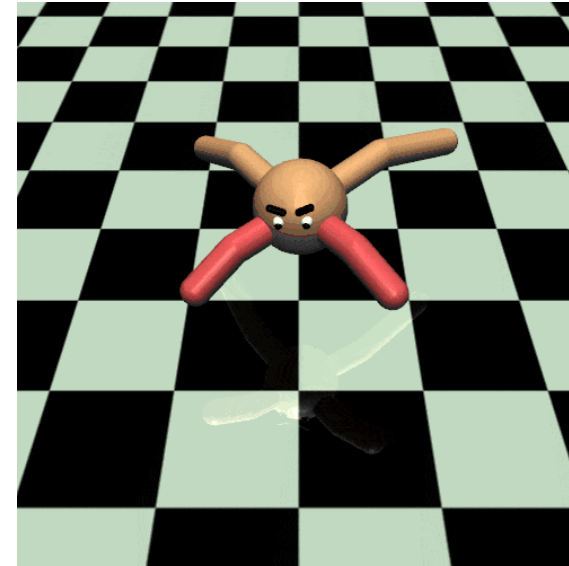
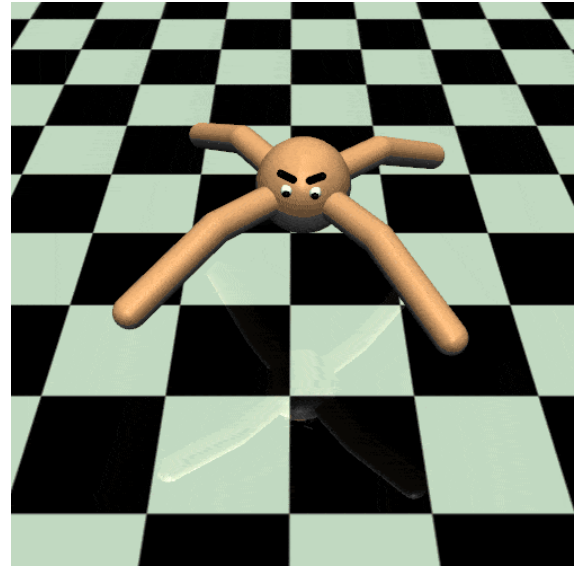
Push up real data, push down generated

$$D_{\theta}(\tau) = \frac{\frac{1}{Z} \exp(r_{\theta}(\tau))}{\frac{1}{Z} \exp(r_{\theta}(\tau)) + \prod_t \pi_{\theta}(a_t | s_t)}$$

GAIL (which is just a GAN), recovers Max-Ent IRL

In practice, often use GAIL and just log D as reward

Adversarial IRL in Action



Lecture Outline

Why Imitation? + Problem formulation



IRLv1 – max margin planning



IRLv2 – max entropy IRL



IRLv3 – partial policy optimization



IRLv4 – adversarial IRL