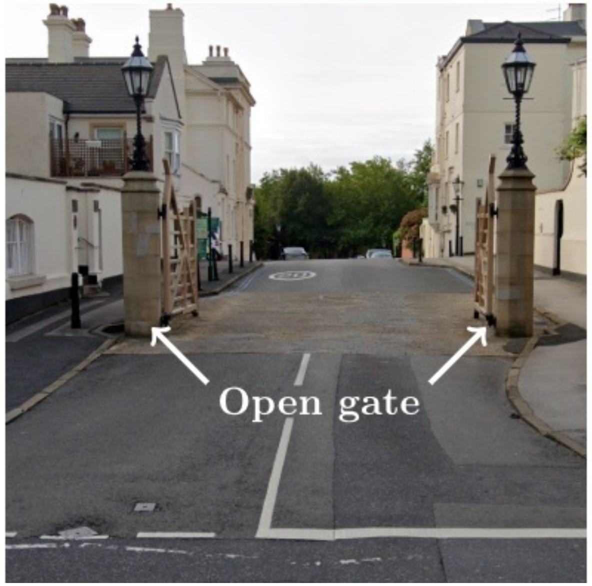
Reinforcement Learning
Autumn 2024

Abhishek Gupta

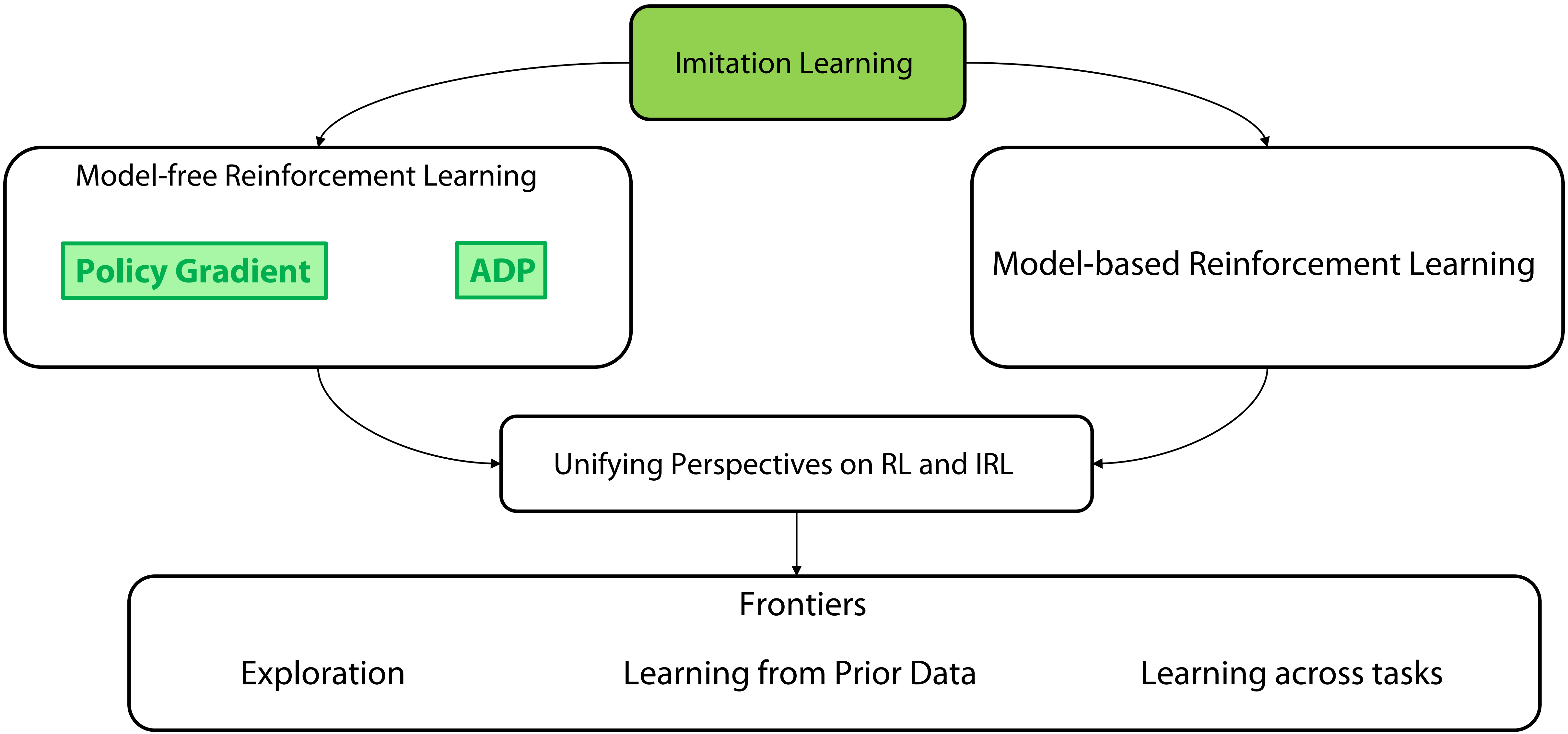
TA: Jacob Berg

Preferred route



Longer route

Class Structure



Previous Lecture Outline

The Anatomy of Model-Based Reinforcement Learning

↓

Model based RL v0 → random shooting + MPC

↓

Model based RL v1 → MPPI + MPC

↓

Model based RL v2 → uncertainty based models

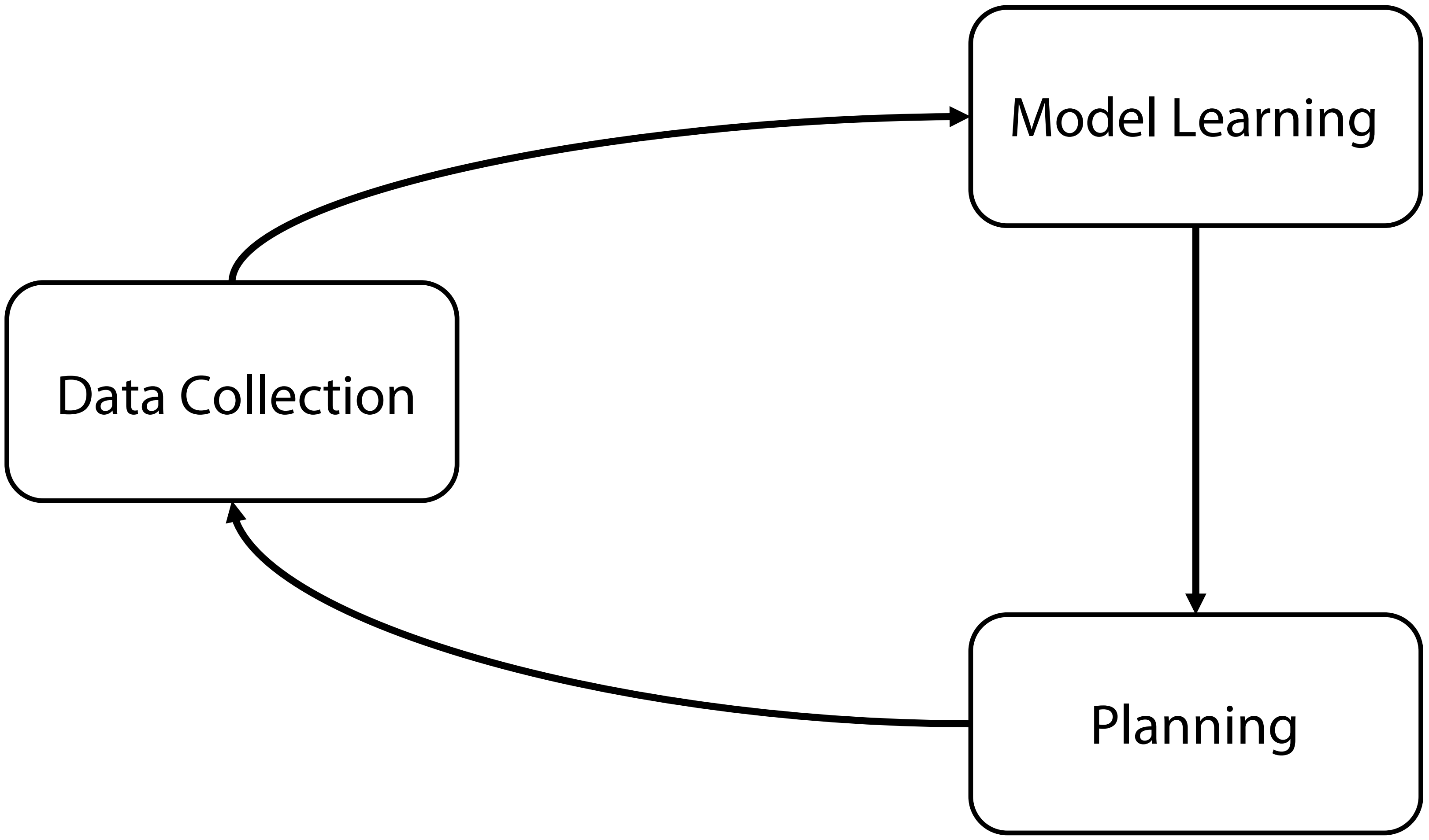
↓

Model based RL v3 → policy optimization with models

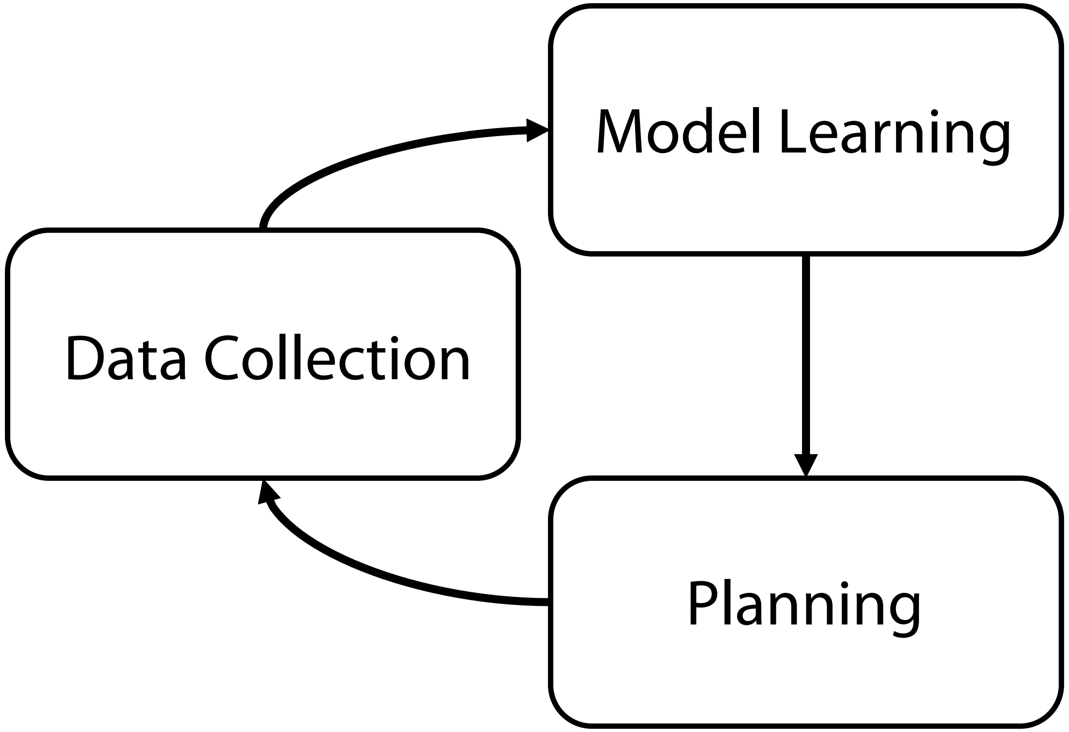
↓

Model based RL v4 → latent space models with images

Model Based RL – A template



Model Based RL – Naïve Algorithm (v0)



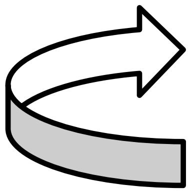
Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_{\theta}(s'|s,a)\right]$$

Data collected from planner

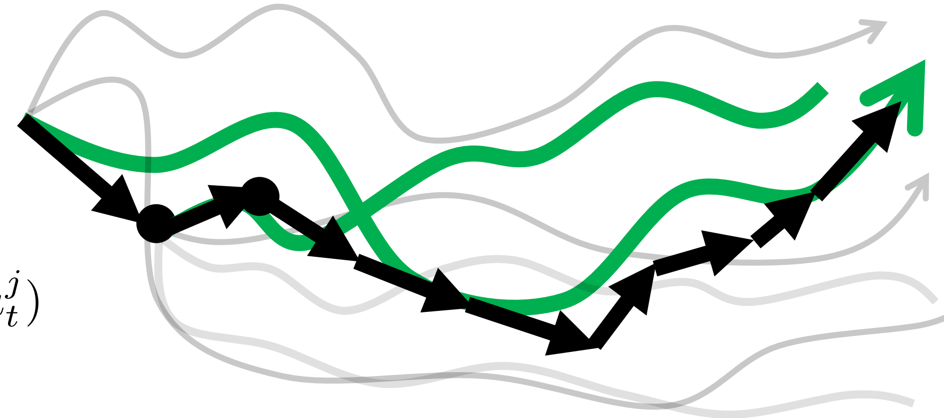
Planning with Shooting + MPC

Better than open loop planning because of feedback

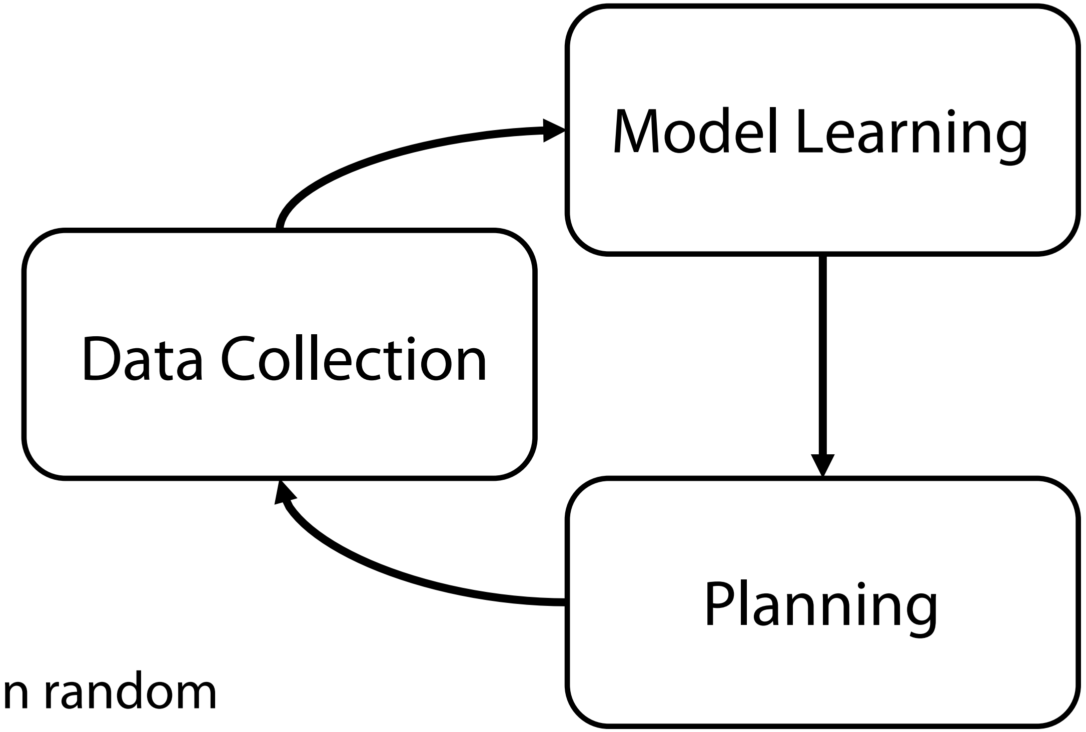


$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_{\theta}(.|\hat{s}_t^j, a_t^j)$$



Model Based RL – Better Sampling Methods (v1)

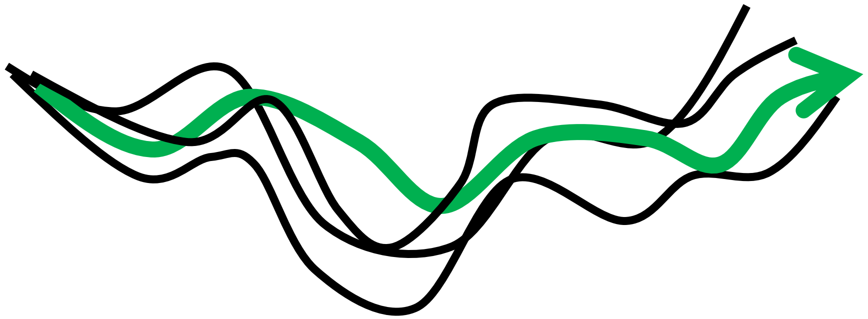


Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

Data collected from planner

Planning with MPPI + MPC



Better than random shooting + MPC, since lower variance!

Aside: Can derive this update trying to bring sampling distribution close to optimal distribution

$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$

$$p(a) \leftarrow p(a) \frac{\exp(\sum_t r(s_t, a_t))}{Z}$$

What is uncertainty?

<u>Alleatoric Uncertainty</u>

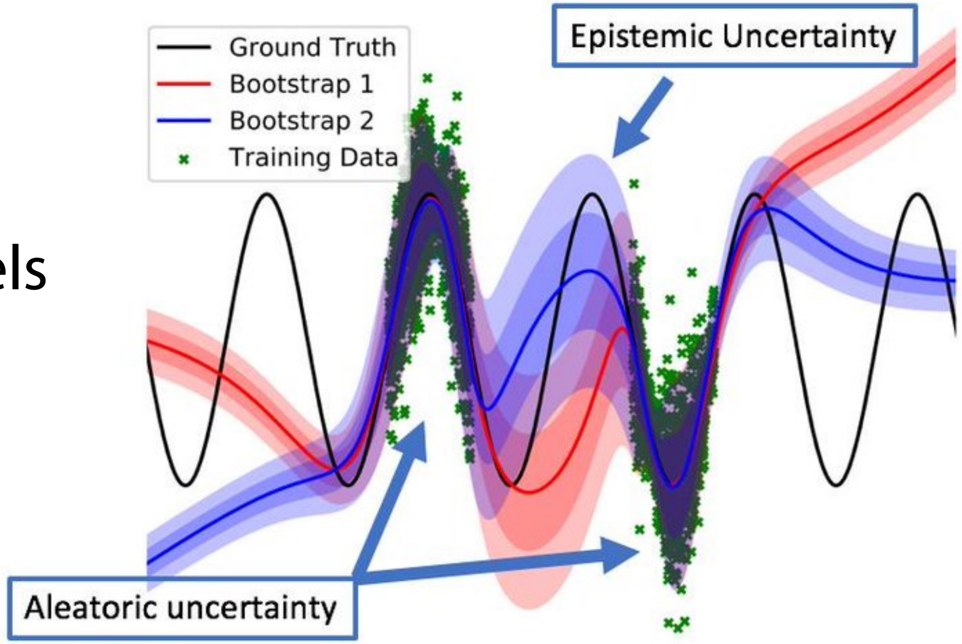
(environment stochasticity)

Easier, can use stochastic models

<u>Epistemic Uncertainty</u>

(Lack of data)

More challenging, need to compute posterior



Let's largely focus on epistemic uncertainty

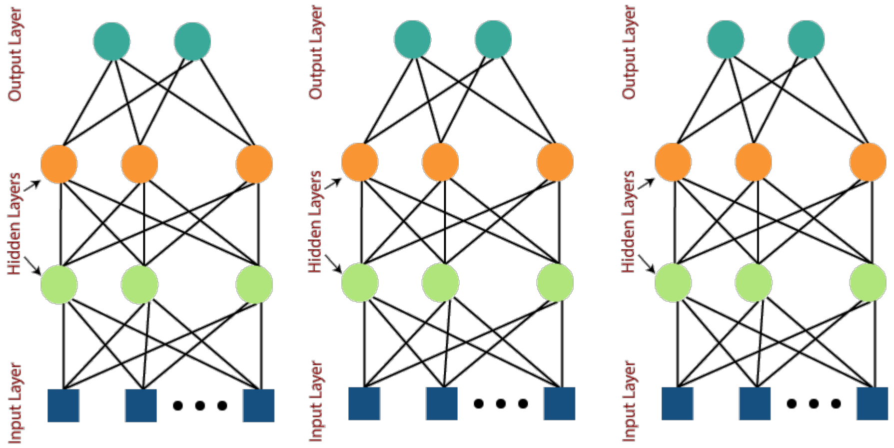
How might we measure uncertainty?

$$p(\theta|\mathcal{D})$$

Difficult to estimate directly!

Learn an ensemble of models

1. Bayesian neural networks
2. Ensemble methods
3. …



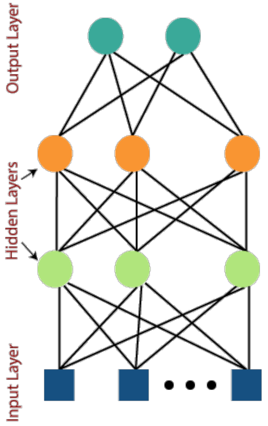
Low data regime → high ensemble variance

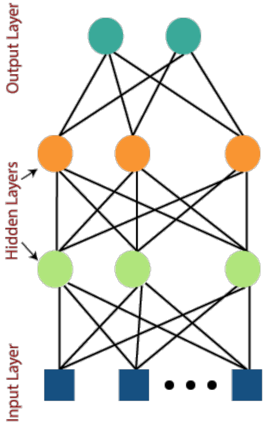
Approximate posterior

Easier and more expressive than BNNs!

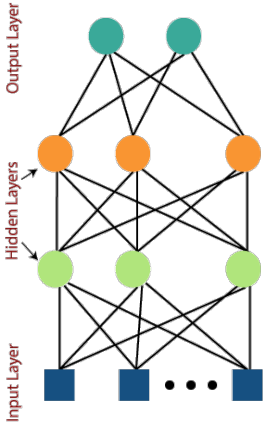
Model Based RL – Learning Ensembles of Dynamics Models

Learn ensembles of dynamics models with MLE rather than a single model





...



$$\max_{\theta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\log \hat{p}_{\theta}(s'|s, a) \right] \quad \max_{\theta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\log \hat{p}_{\theta}(s'|s, a) \right] \quad \max_{\theta} \mathbb{E}_{(s,a,s') \sim \mathcal{D}} \left[\log \hat{p}_{\theta}(s'|s, a) \right]$$

Learn ensembles by either subsampling the data or having different initializations

Lecture Outline

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

Inverse RL Problem Formulation

↓

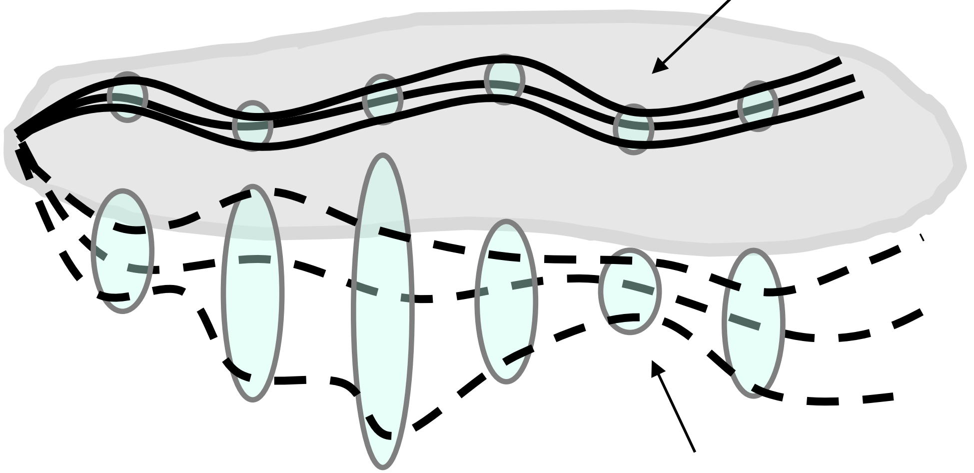
IRLv1 – max margin planning

↓

IRLv2 – max entropy IRL

Model Based RL – Integrating Uncertainty into MBRL (v2)

Take expected value under the uncertain dynamics



Expected value over ensemble

$$\arg \max_{(a_0^j, a_1^j, \ldots, a_T^j)_{j=1}^N} \sum_{i=1}^{K} \sum_{t=0}^{T} r((\hat{s}_t^j)^i, a_t^j)$$

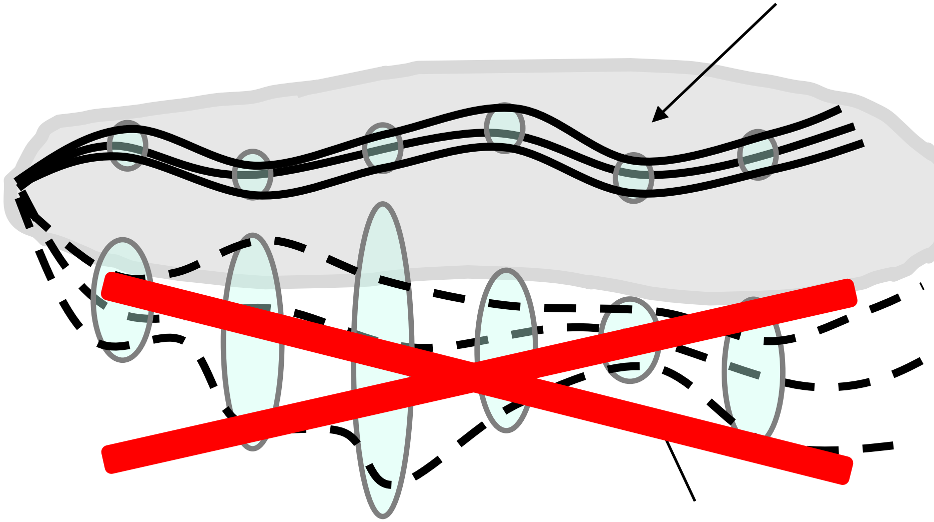
$$(\hat{s}_{t+1}^j)^i \sim \hat{p}_{\theta_i}(.|(\hat{s}_t^j)^i, a_t^j)$$

Can also swap which ensemble element is propagated at every step or just pick randomly amongst them

Avoids overly OOD settings since the expected reward is affected by uncertainty

Model Based RL – Integrating Uncertainty into MBRL (v2)

Take **pessimistic** value under the uncertain dynamics



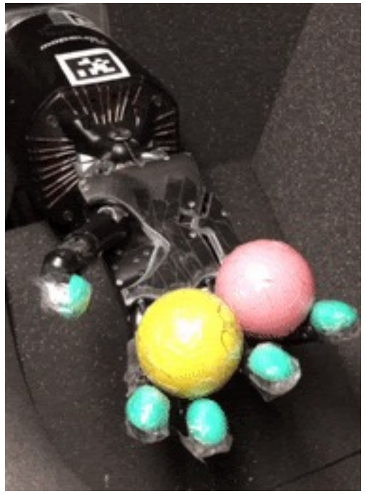
Penalize ensemble variance

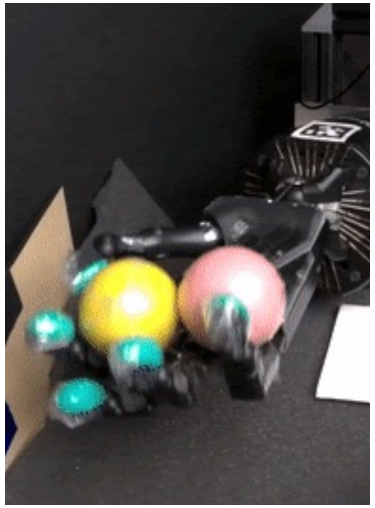
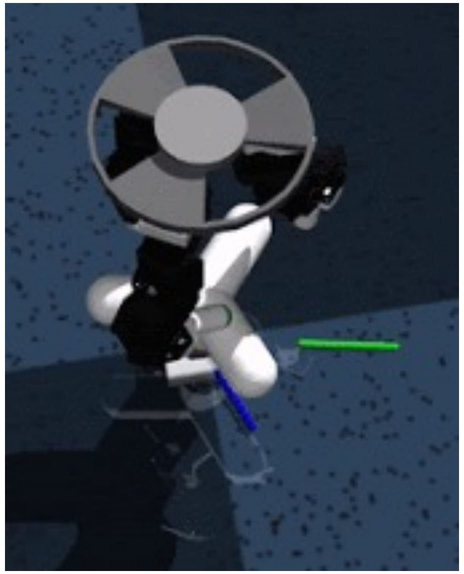
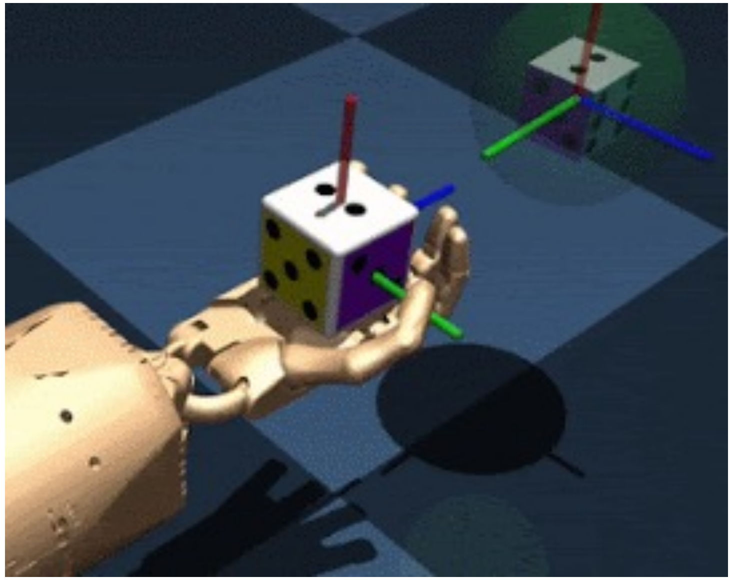
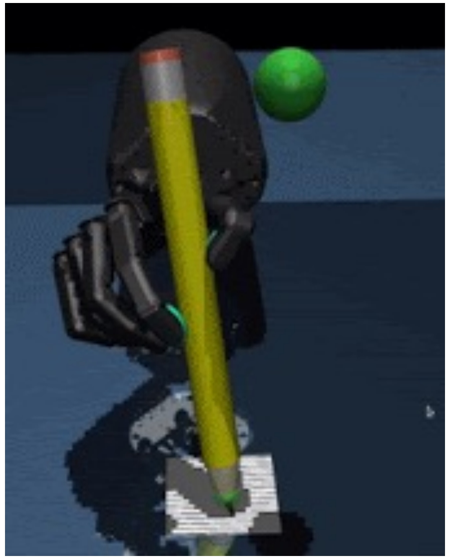
$$\arg \max_{(a_0^j, a_1^j, \ldots, a_T^j)_{j=1}^N} \sum_{i=1}^K \sum_{t=0}^T r((\hat{s}_t^j)^i, a_t^j) - \lambda \mathrm{Var}((\hat{s}_t^j)^i)$$

$$(\hat{s}_{t+1}^j)^i \sim \hat{p}_{\theta_i}(.|(\hat{s}_t^j)^i, a_t^j)$$

Avoids overly OOD settings since these states are explicitly penalized

Does this work?



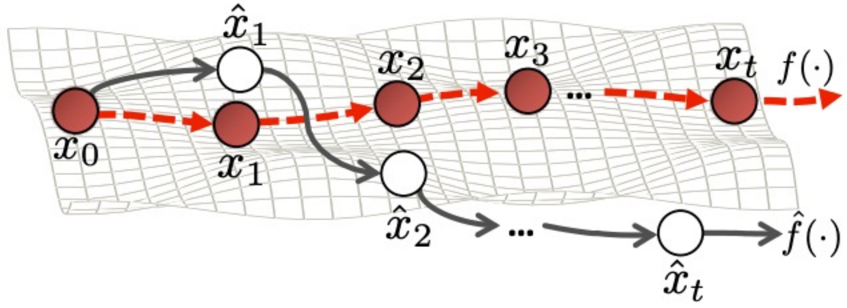






How might we deal with compounding error?

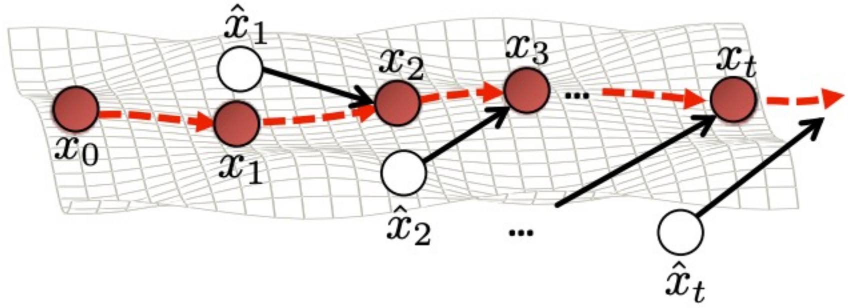
Idea 3: Cast this as an imitation learning problem

Reuse ideas from DAgger!

Compounding error



Synthetically generative corrective labels



Can help to correct model predictions with "feedback"

Can run into issues if the synthetic labels conflict with true data

Lecture Outline

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

Inverse RL Problem Formulation

↓

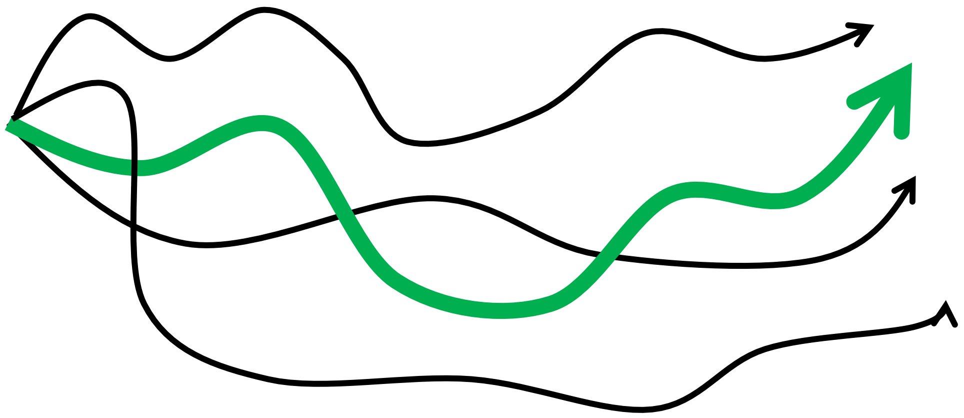
IRLv1 – max margin planning

↓

IRLv2 – max entropy IRL

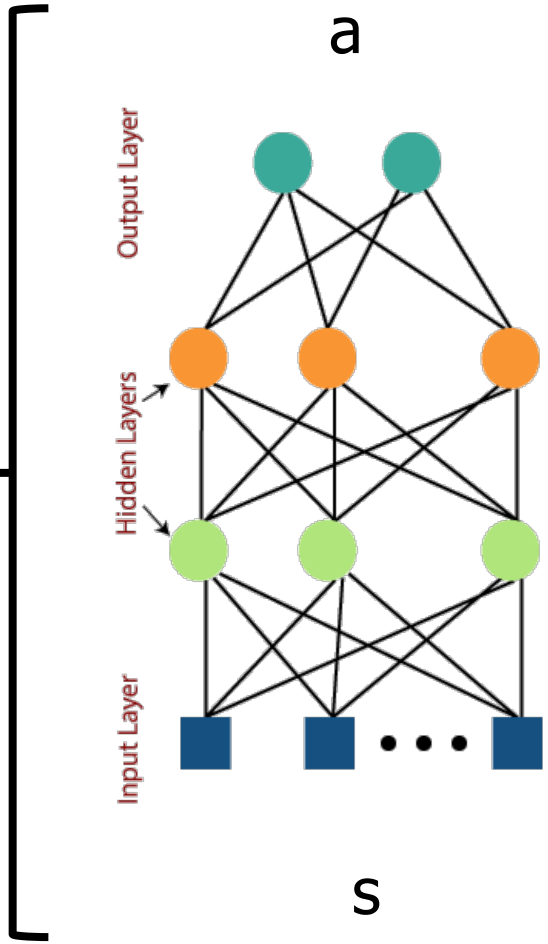
What might be the issue?

Huge number of samples needed to reduce variance



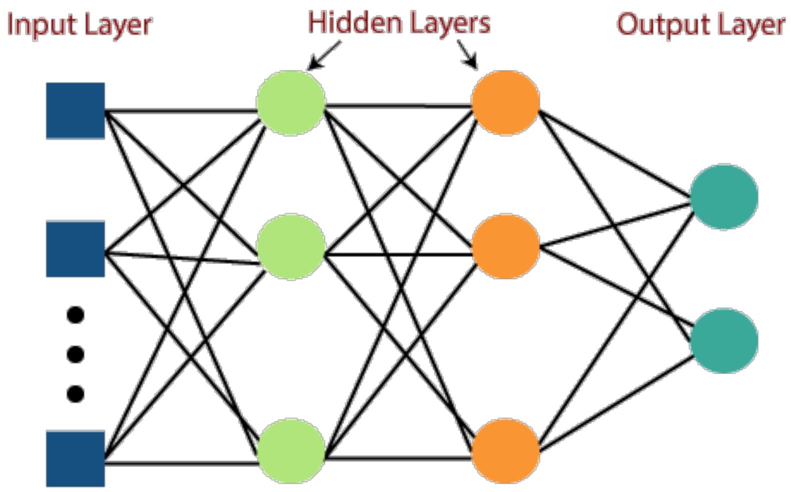
Extremely slow, hard to run in real time

Amortize planning into a policy

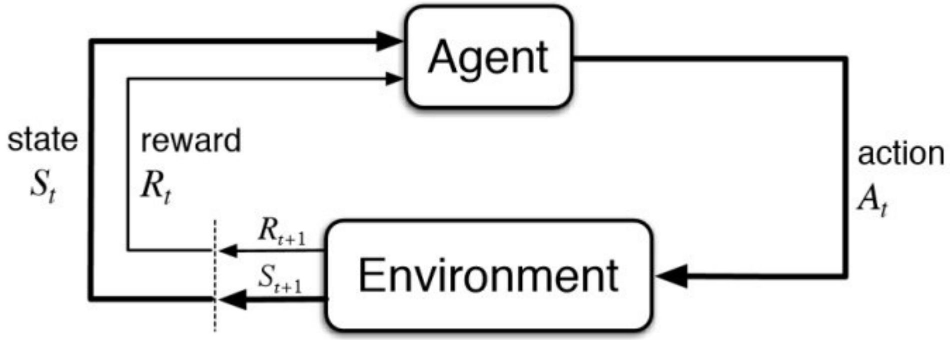


Speeding Up Model-Based Planning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_{\theta}(s'|s,a)\right]$$

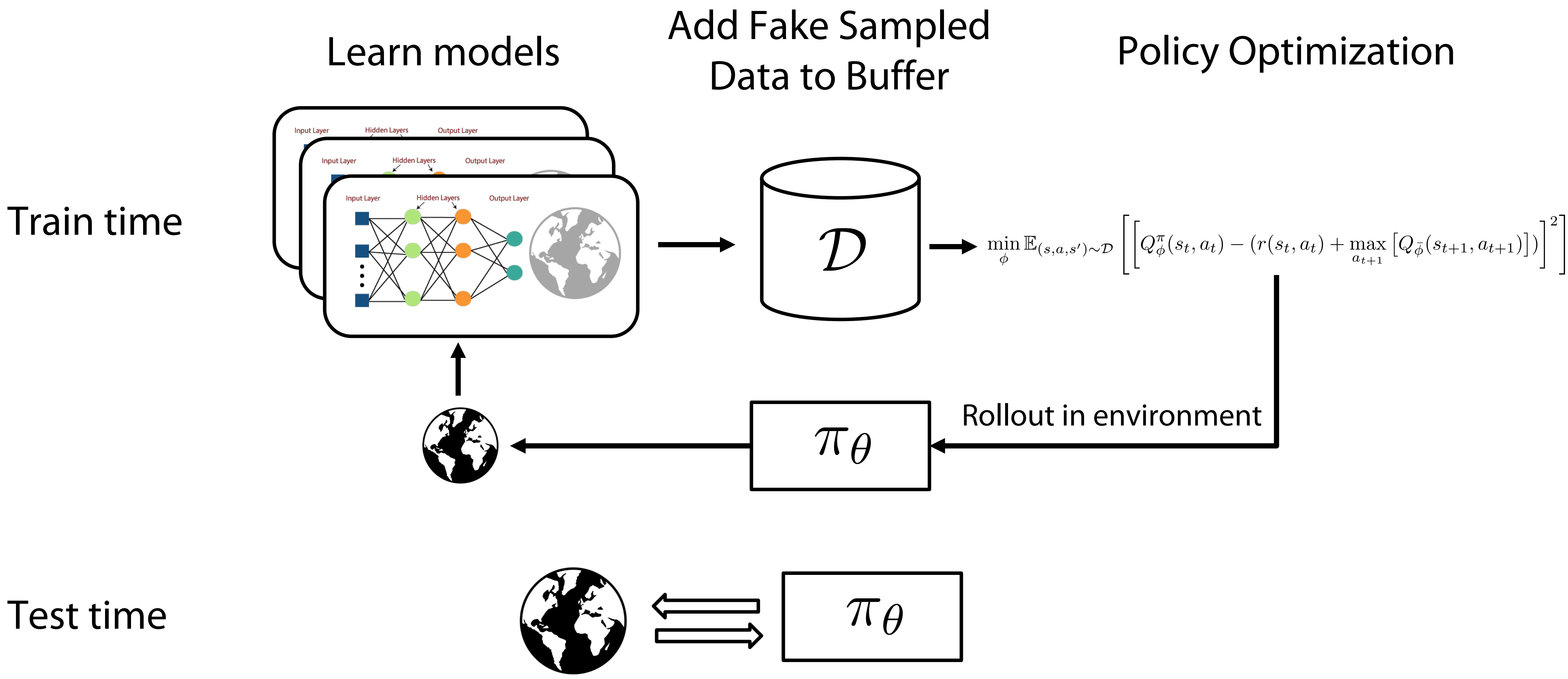


Use model(s) to generate data for policy optimization

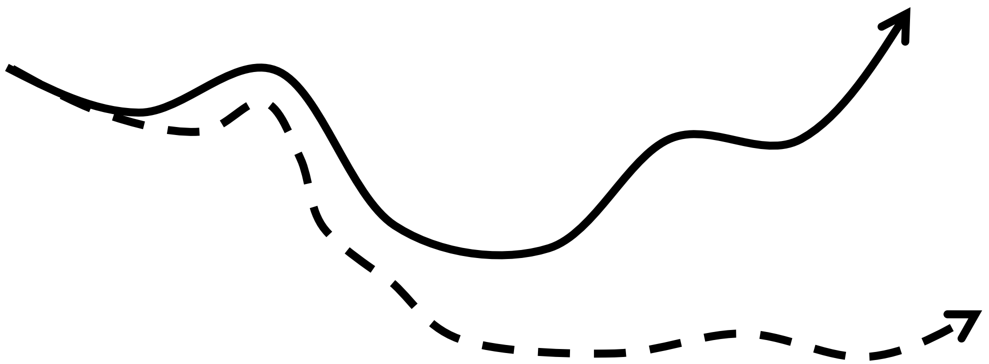


Can use PG or off-policy!

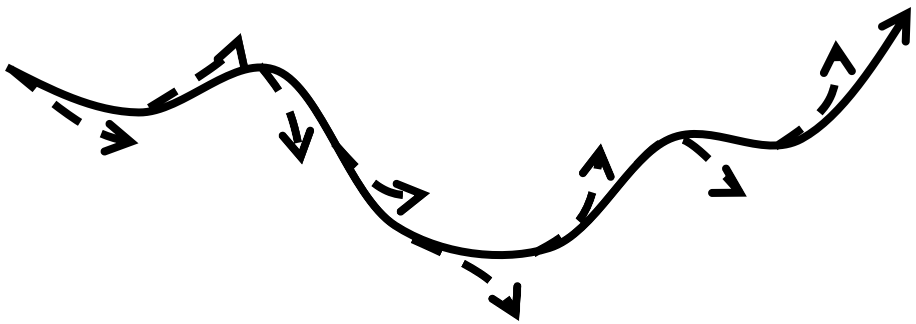
Generating Data for Policy Optimization



What matters in generating data from models?



Long horizon rollouts can deviate

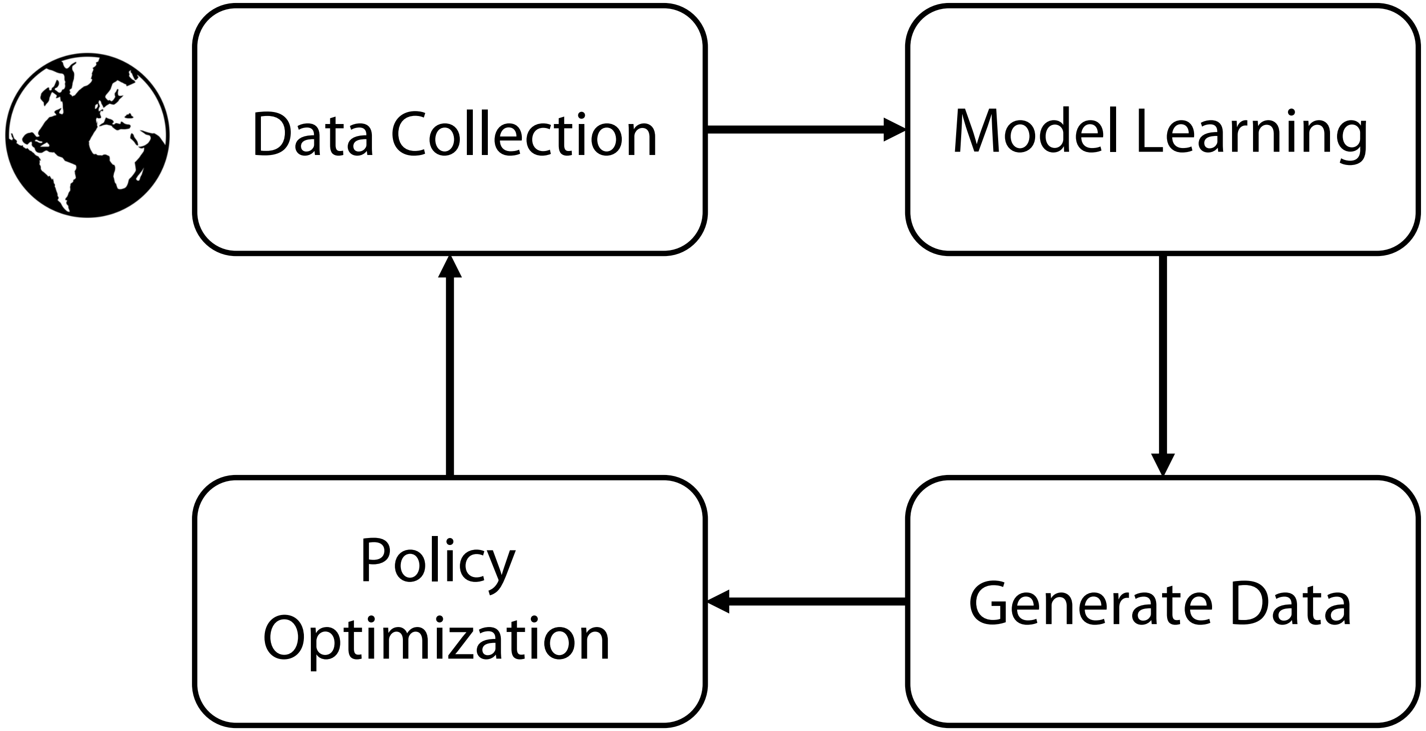


Short horizon rollouts deviate far less

Balance between off-policy coverage and compounding error

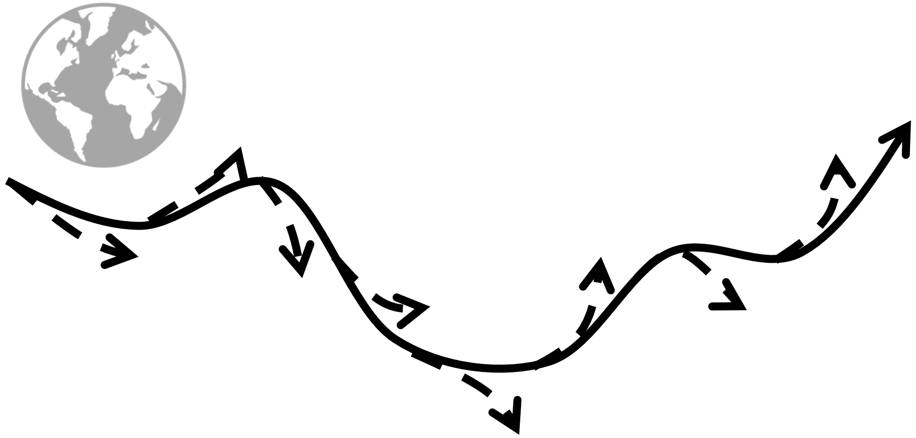
More at https://arxiv.org/abs/1906.08253

Model Based RL – Using Models for Policy Optimization (v3)



Maximum likelihood supervised Learning

$$\max_\theta \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$



$$\min_\phi \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^\pi_\phi(s_t,a_t) - \left(r(s_t,a_t) + \max_{a_{t+1}}\left[Q_{\bar{\phi}}(s_{t+1},a_{t+1})\right]\right)\right]^2\right]$$

More expensive/harder at training time, faster at test time

Does this work?

Lecture Outline

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

Inverse RL Problem Formulation

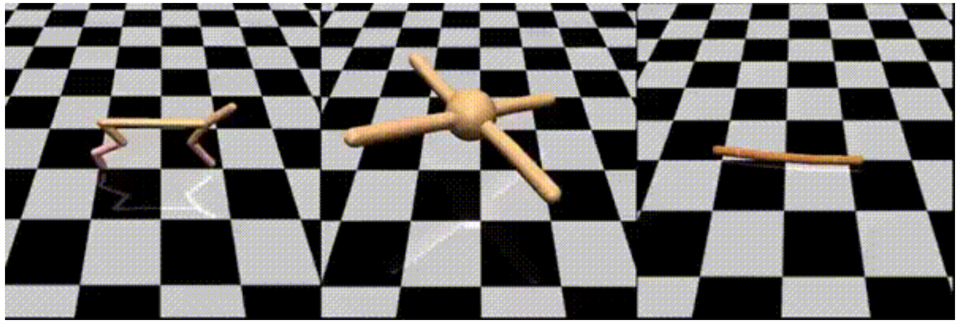
↓

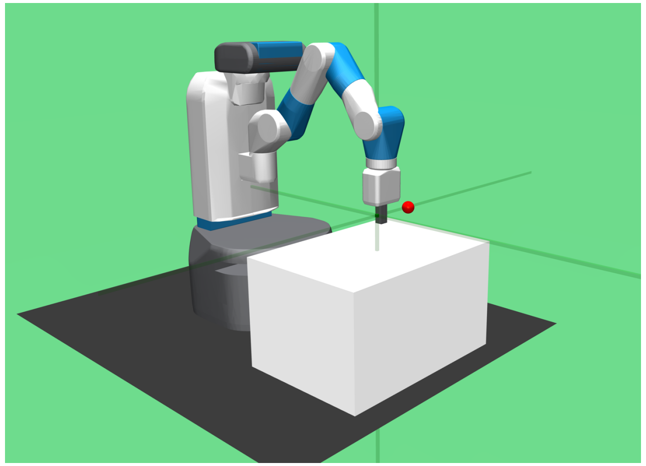
IRLv1 – max margin planning

↓

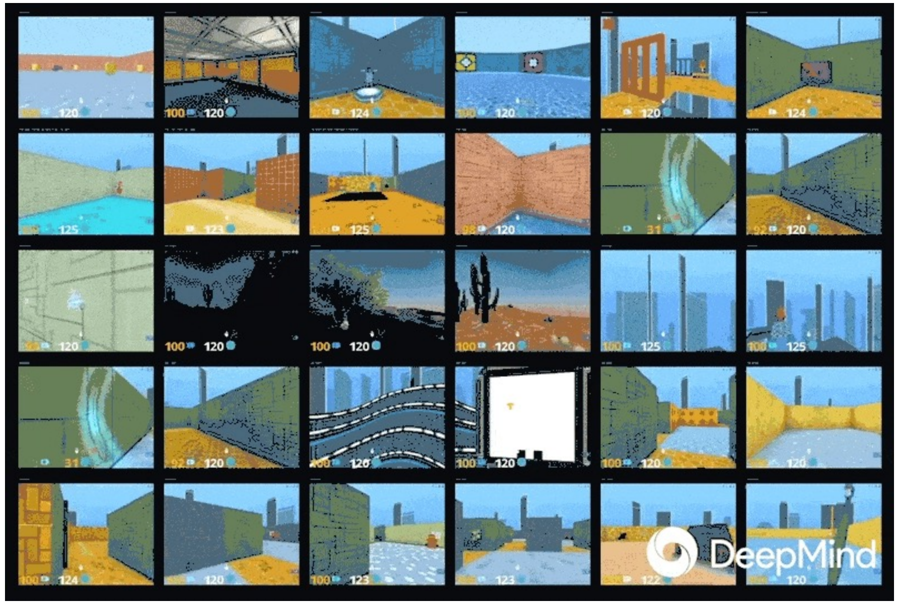
IRLv2 – max entropy IRL

What about images?





State based domains



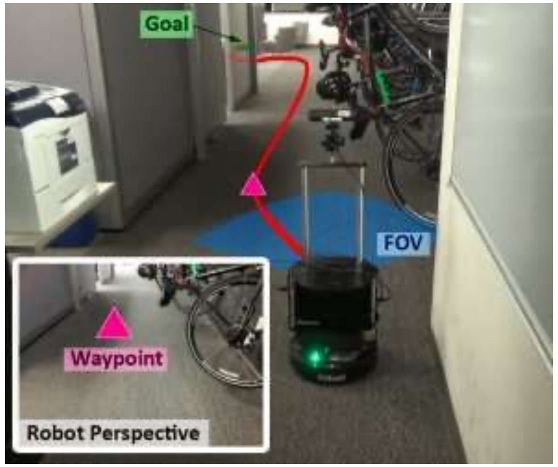
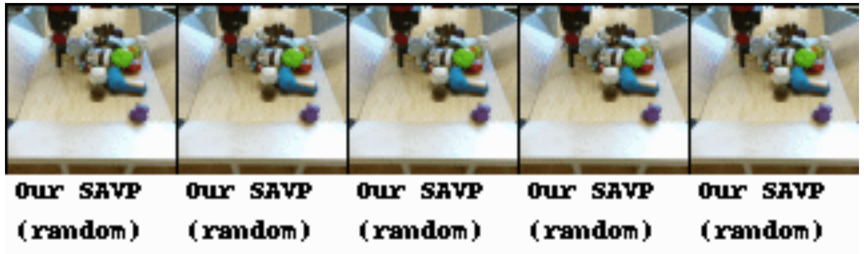


Image based domains

Why is learning from images hard?

Generative modeling is videos, challenging to model multimodal correlated predictions

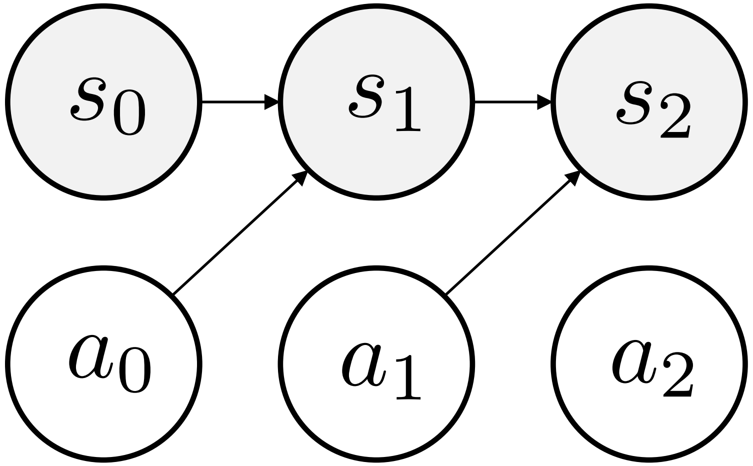


Partially observable!

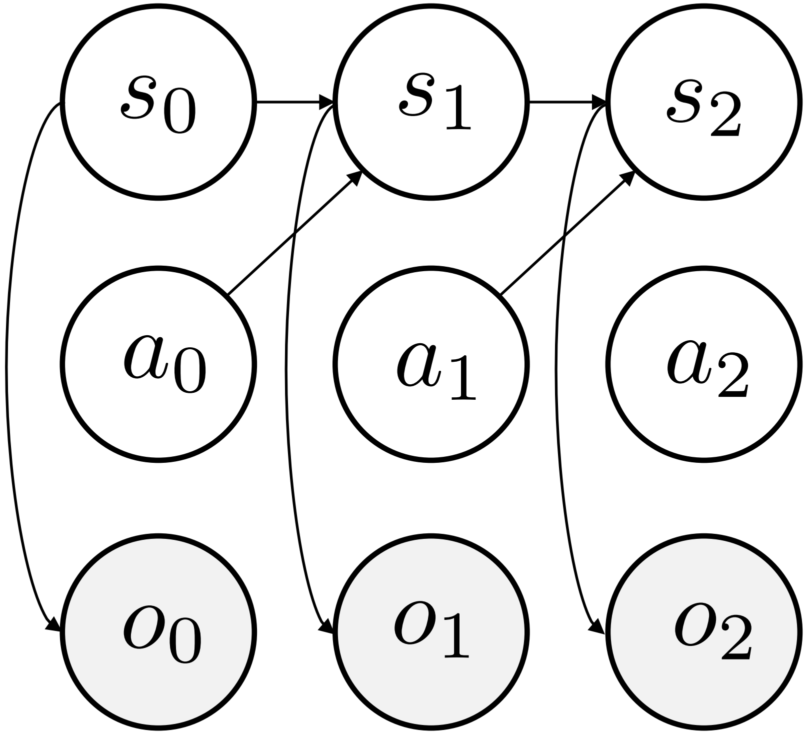
Long horizon predictions in video space can be challenging!

Model Based RL – Latent Space Models for Image Based RL (v4)

Fully observed – Markovian case



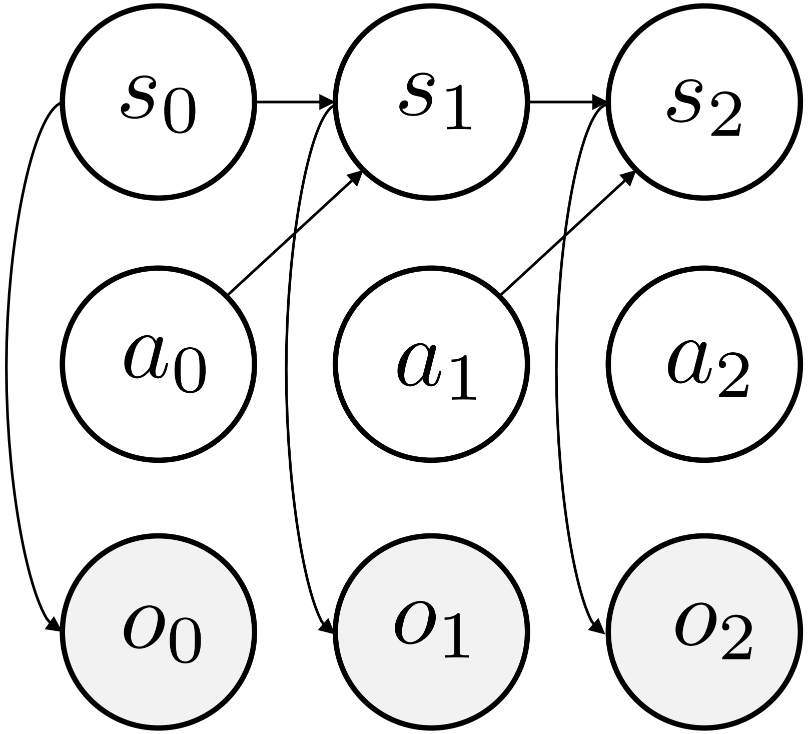
Partially observed – Non-Markovian case



If we can infer latent state and learn dynamics, then we can plan in a much smaller space

How do we infer latent state and learn dynamics in this space?

How do we **train** latent space models?



Learn latent encoder to infer latent state from observations $q_\phi(s_t|o_{1:t})$

Learn action conditioned latent transition model $p_\eta(s_{t+1}|s_t, a_t)$

$$\log p_\eta(q_\phi(s_{t+1}|o_{1:t+1})|q_\phi(s_t|o_{1:t}), a_t)$$

Learn latent decoder to reconstruct observations $p_\psi(o_t|s_t)$

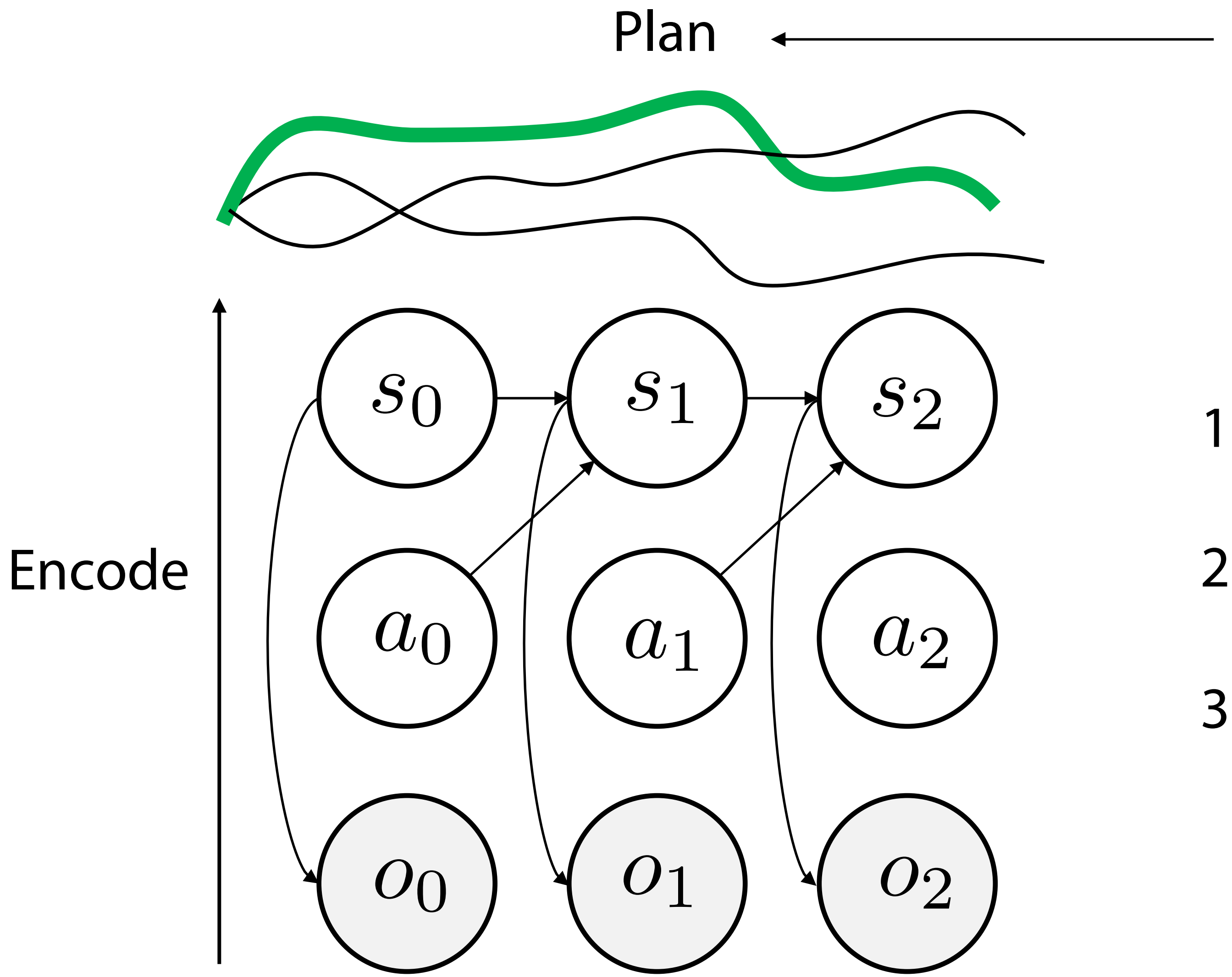
$$\log p_\psi(o_t|s_t)$$

Learn reward predictor from latent state $p_\zeta(r_t|s_t)$

$$\log p_\zeta(r_t|q_\phi(s_t|o_{1:t}))$$

Can derive the whole thing from first principles using variational inference!

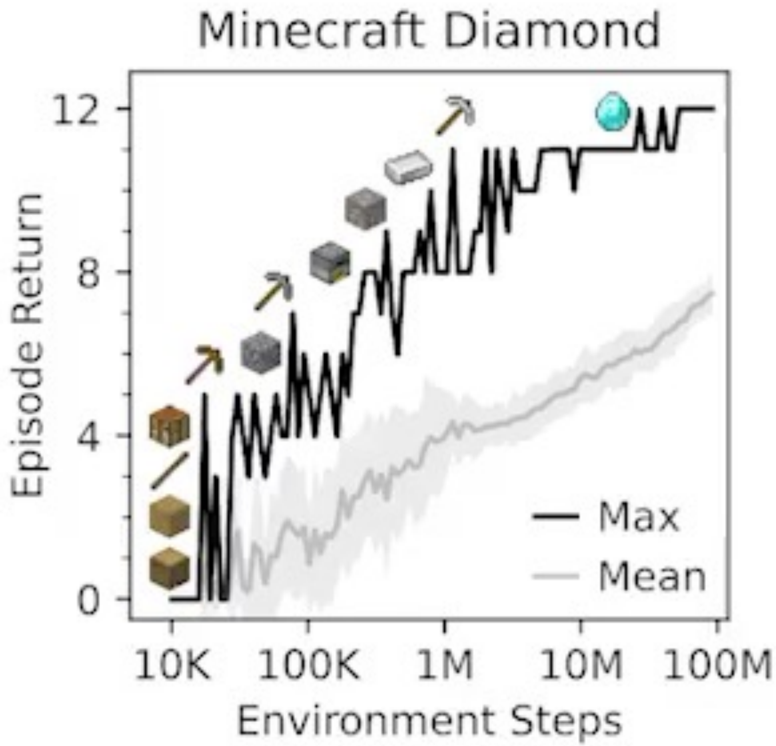
How do we **use** latent space models?



Apply any of the methods from this lecture, just in latent space!

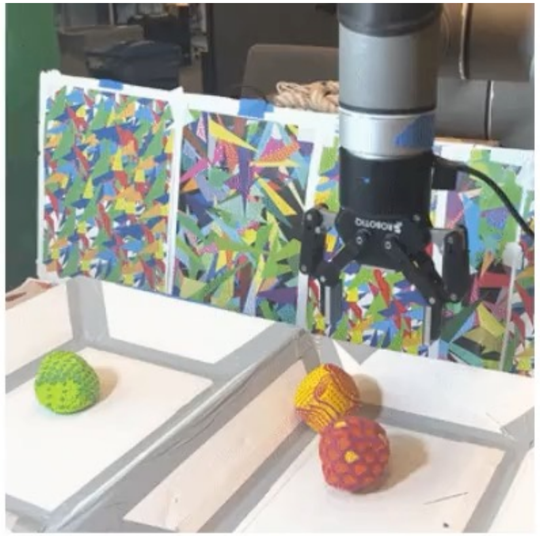
1. Avoids predicting image frames at planning time
2. Scales much better than image prediction
3. Allows for longer horizon predictions

Does this work?

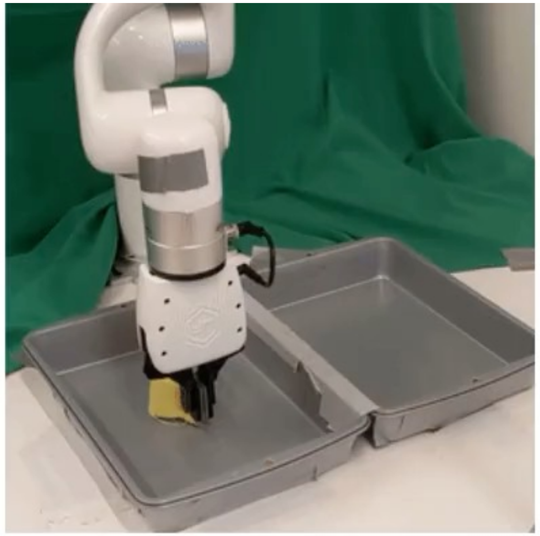


Does this work?

A1 Quadruped Walking



UR5 Multi-Object Visual Pick Place



XArm Visual Pick and Place

Sphero Ollie Visual Navigation

Training from images in < 1 hour!

Lecture Outline

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

Inverse RL Problem Formulation

↓

IRLv1 – max margin planning

↓

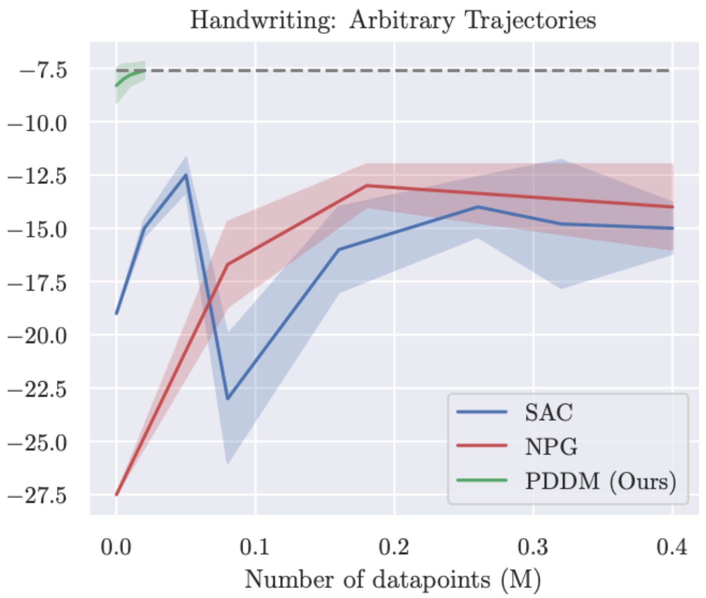
IRLv2 – max entropy IRL

Why should you care?

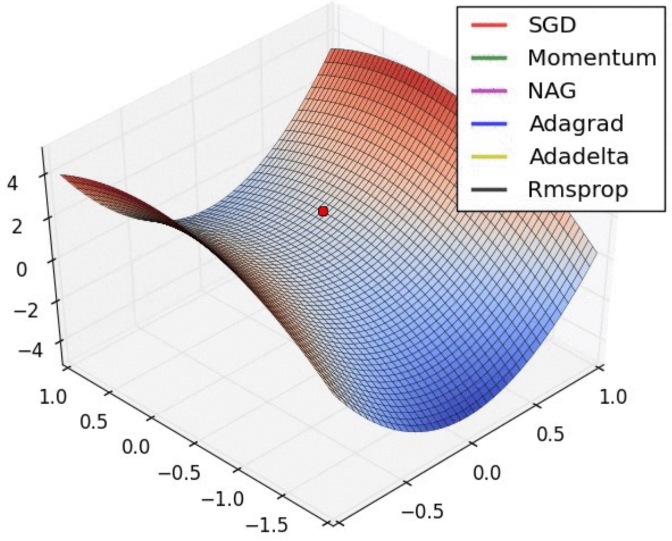
Model based RL **<u>may be</u>** a much more practical path to real world robotics

Transfer/Adaptive

Efficiency



Simplicity



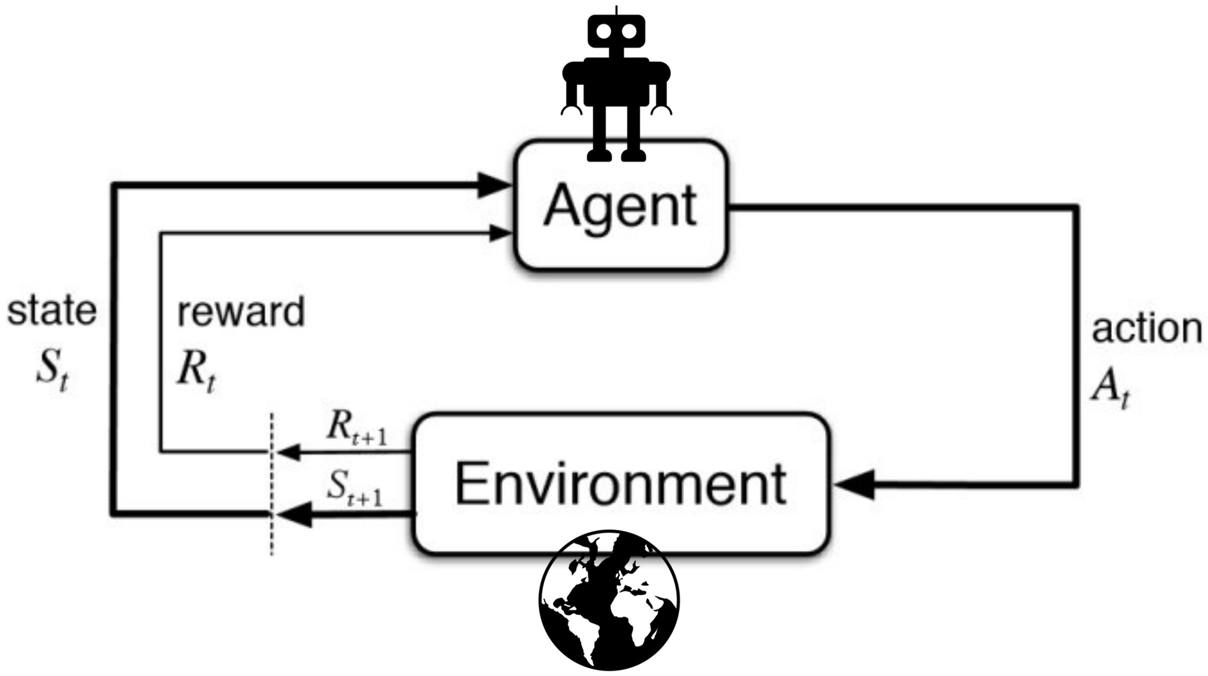
← Likely to be the most future proof one!

Are models really that different than Q-functions?

	Models	Q-functions
Similar	1. Off-policy 2. Models the future Very different than PG methods → on-policy, models current given future	
Different	1. 1-step modeling 2. Models states 3. Can evaluate arbitrary policies 4. Parametric storage of training data	1. Cumulative modeling 2. Models returns 3. Can evaluate only policy π 4. Non-parametric storage of data

Ok let’s switch gears to inverse reinforcement learning

Let's revisit the premise of reinforcement learning



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

or

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

We studied a bunch of different algorithms to solve this

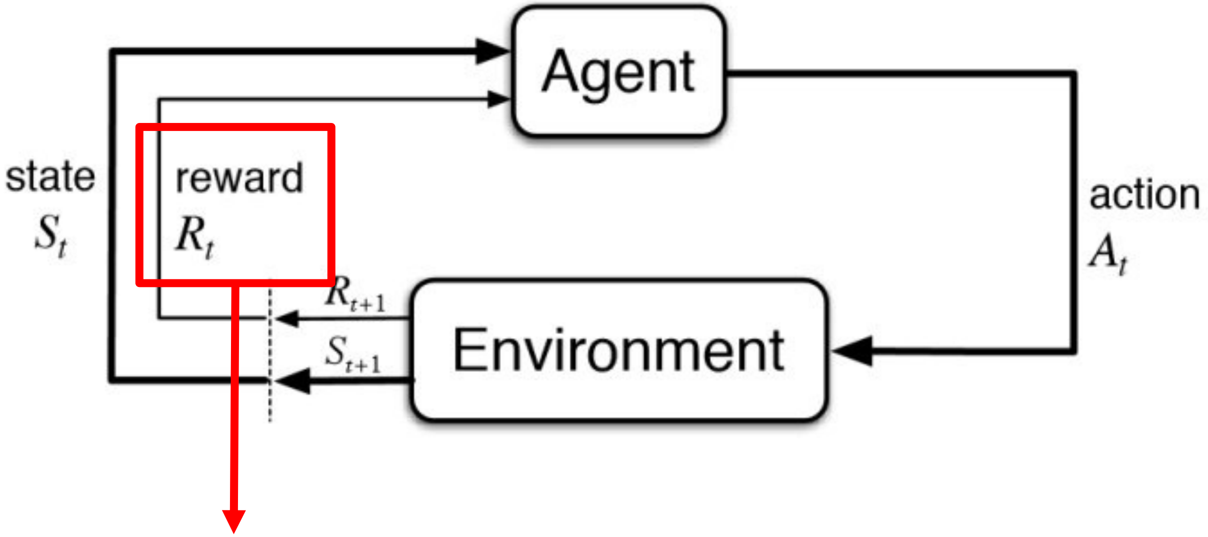
Model-based RL

Policy gradients

Actor-critic

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

Reinforcement Learning requires Task Specification

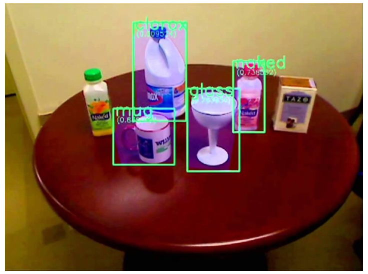


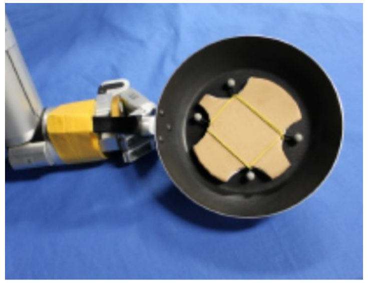
Does not magically appear in most settings

Has to be manually specified

→ can we do better?

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.

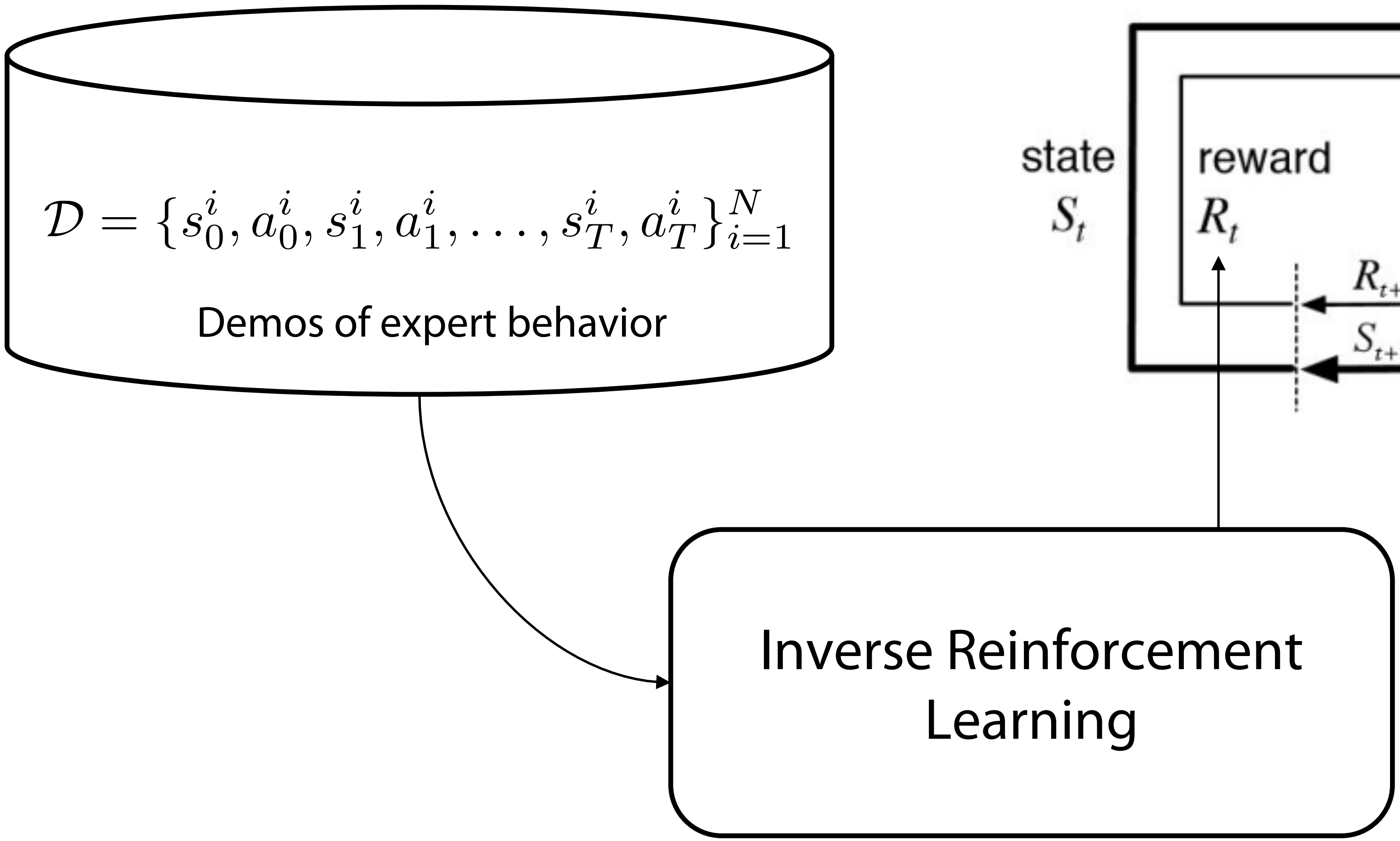
† See item O.2.

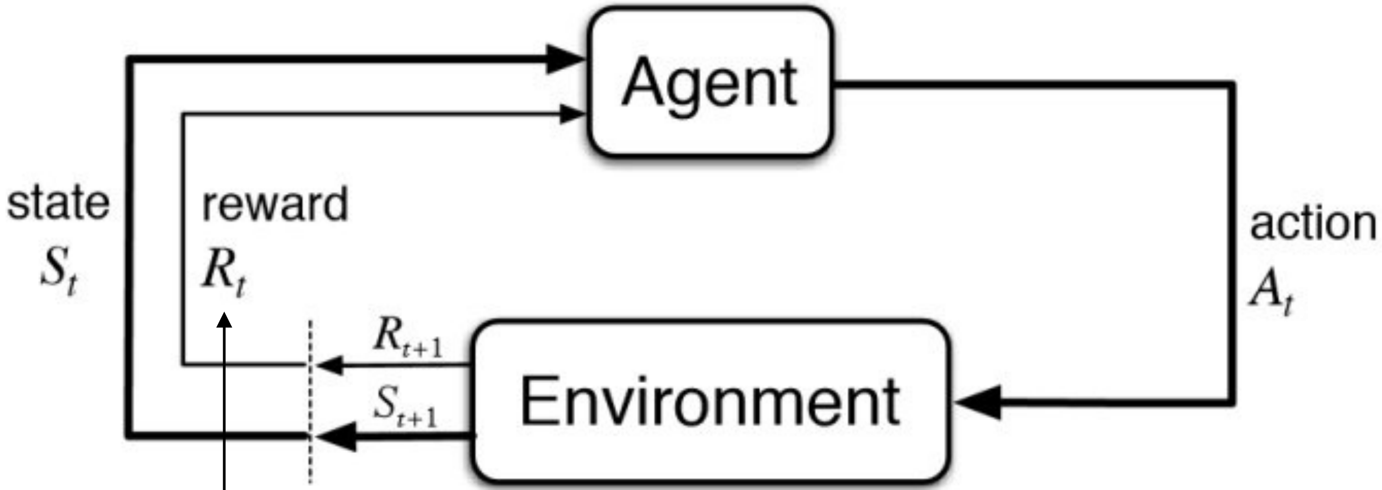
‡ Hero's health is quartically interpolated between 0 (dead) and 1 (full health); health at fraction x of full health is worth $\left(x + 1 - (1-x)^4\right)/2$. This function was not tuned; it was set once and then untouched for the duration of the project.

Table 6: Shaped Reward Weights

Learning from Demonstrations

Avoid manual reward specification by learning from demos of optimal behavior





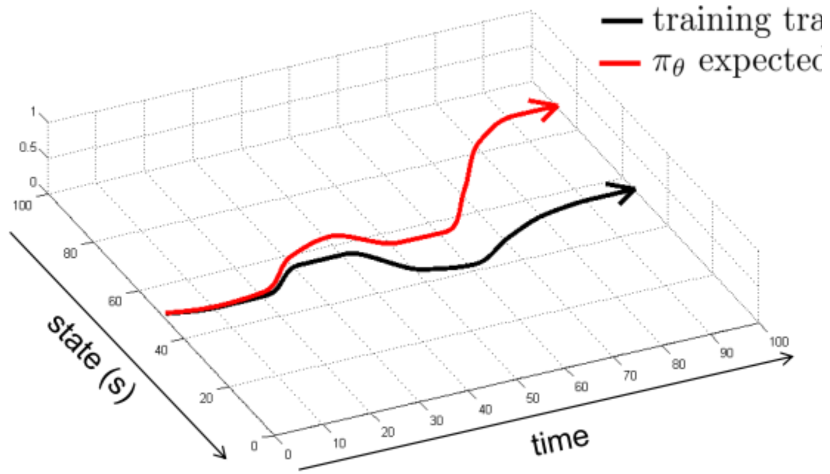
Infer rewards from demonstrations of optimal behavior

→ Optimal behavior optimizes some reward via RL, so the RL process must be "inverted" to find the reward

But haven't we already learned from demonstrations?

Imitation learning via Behavior Cloning (L2)

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



Main difference between BC and IRL:

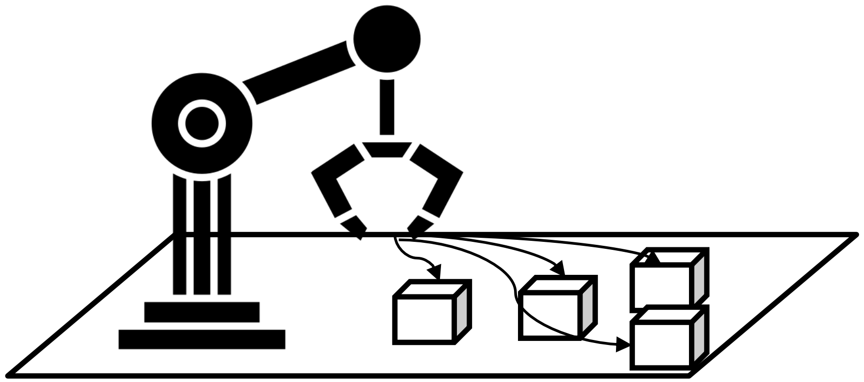
1. BC learns policies, IRL learns rewards
2. BC assumes no environment access, IRL typically assumes either known model or sampling access

Why does this matter?

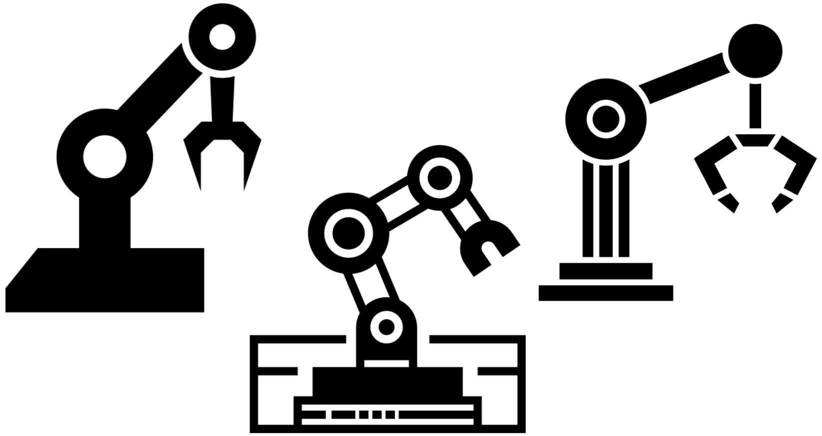
Zooming out – why do we care about imitation?

Imitation learning is all about generalization

Generalization across states



Generalization across dynamics

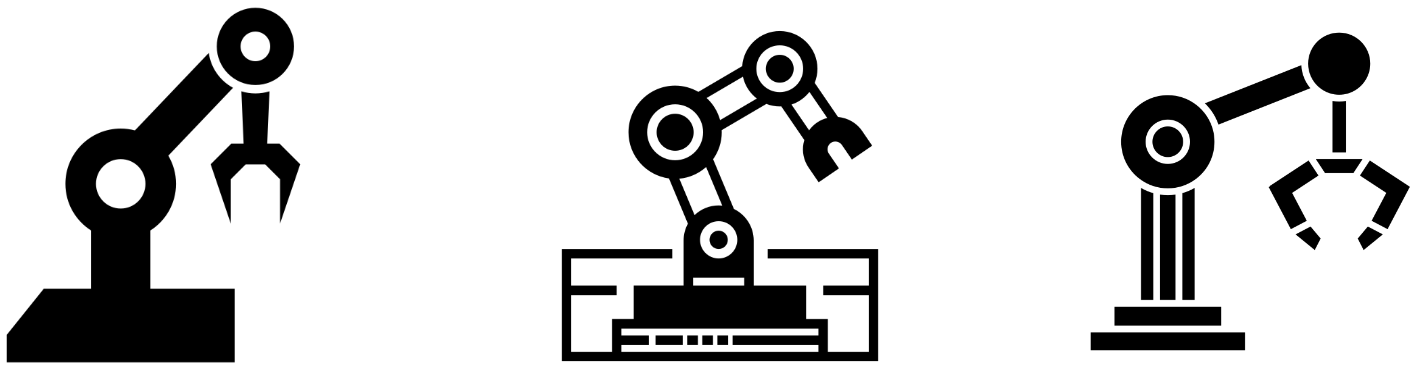


Covariate shift is just a manifestation of generalization

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

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

Sampling from expert

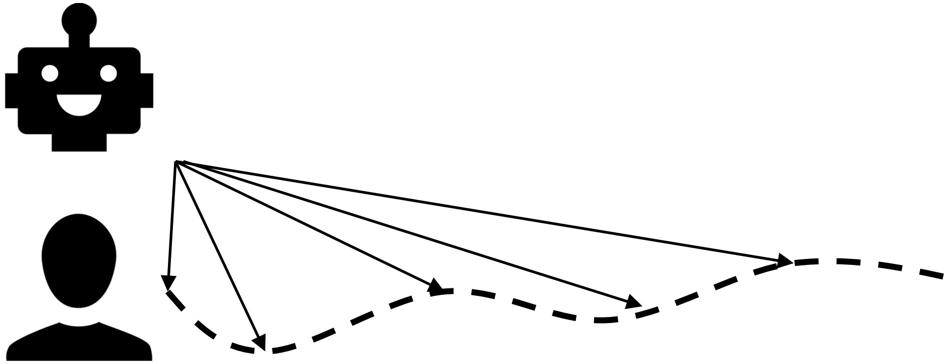
$$D_{\mathrm{KL}}(p^*||p_\theta)$$

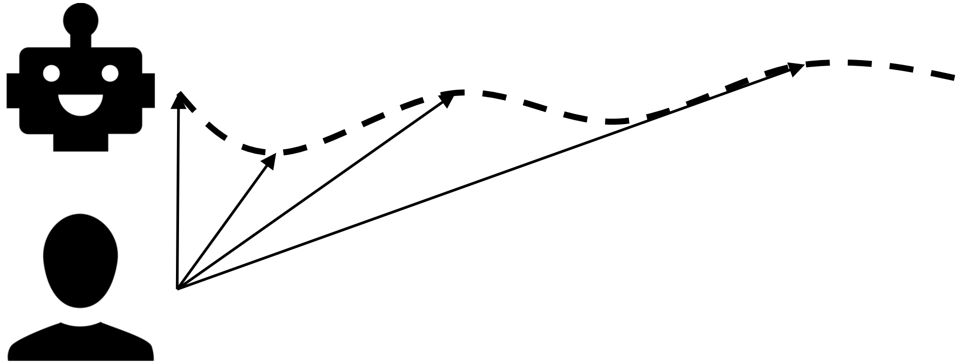
Reinforcement Learning with Inferred Reward

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

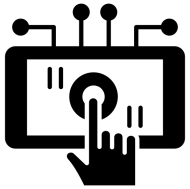
Sampling from policy

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

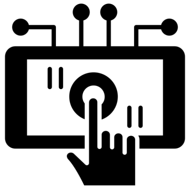




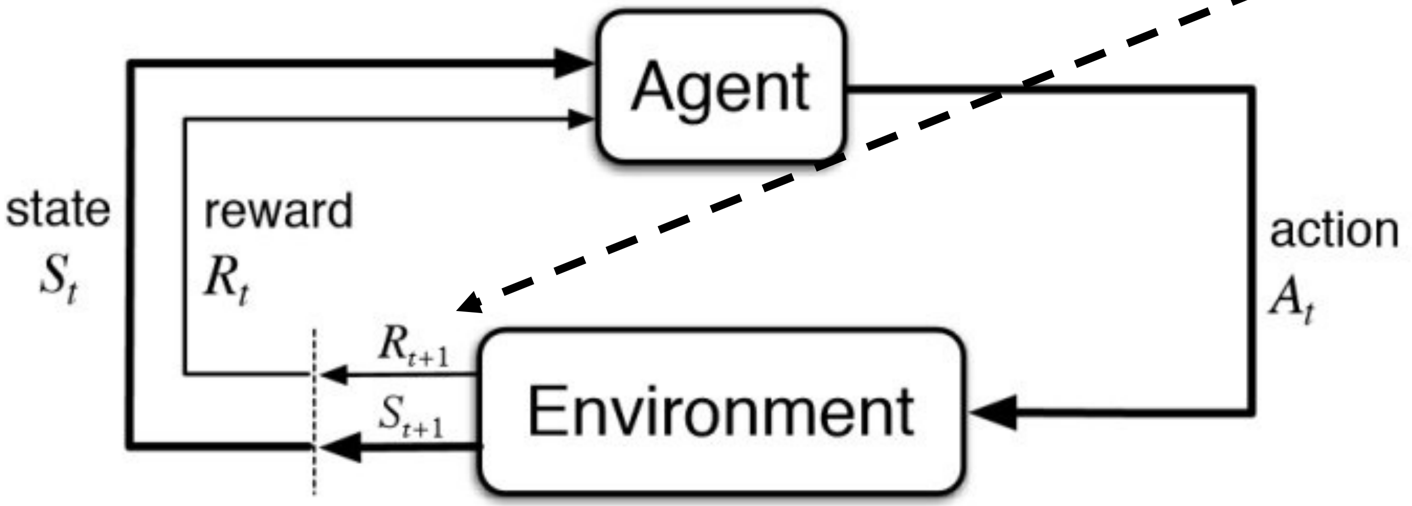
Learning Rewards from Human Data



Imitation Learning → π



Inverse RL → Reward



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

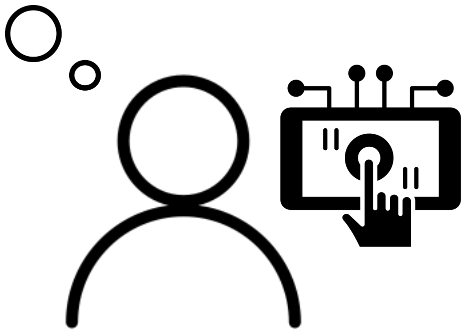
Use human provided data to infer a reward function

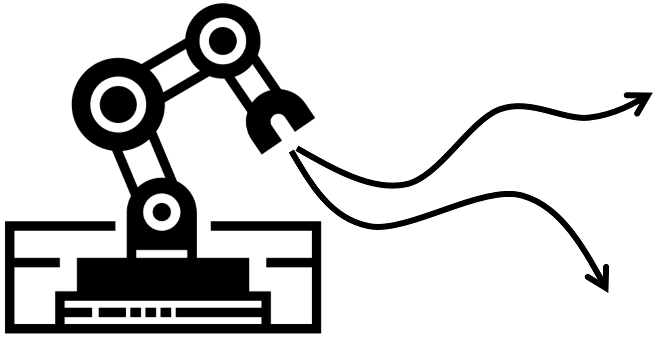
How can we learn rewards?

We must make some assumptions on the expert provided data

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

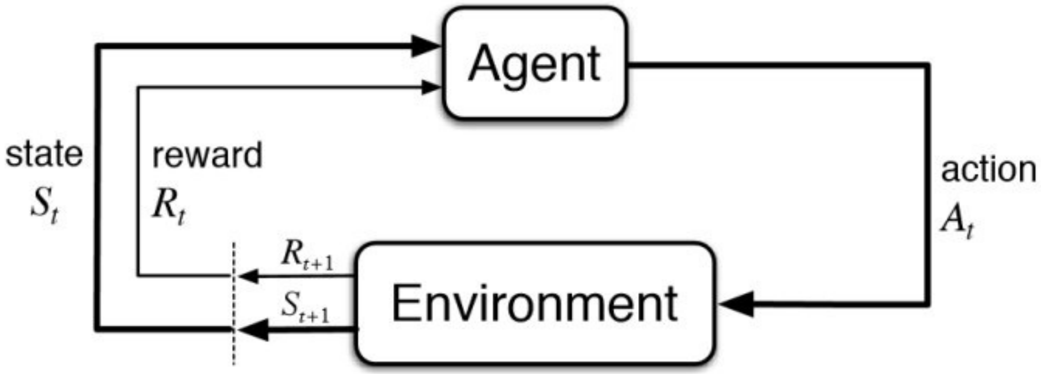
$$D_{\mathrm{KL}}(\pi \,||\, \pi^*) \leq \epsilon$$



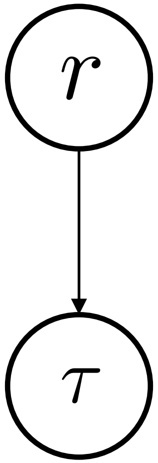


Experts are assumed to be “noisily” optimal

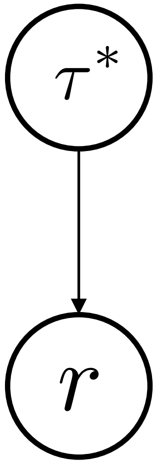
Why is this "inverse" reinforcement learning?



<u>RL: Rewards generate trajectories</u>



<u>IRL: Expert trajectories generate rewards</u>



Is this well defined?

IRL problem statement + assumptions

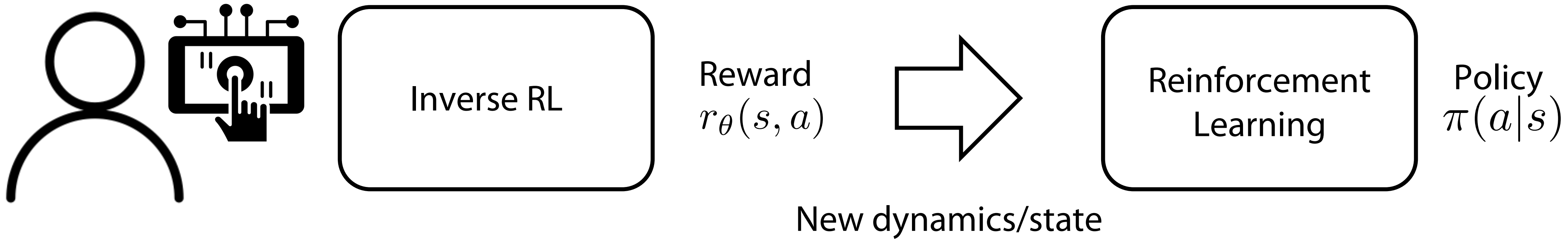
Reinforcement Learning

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

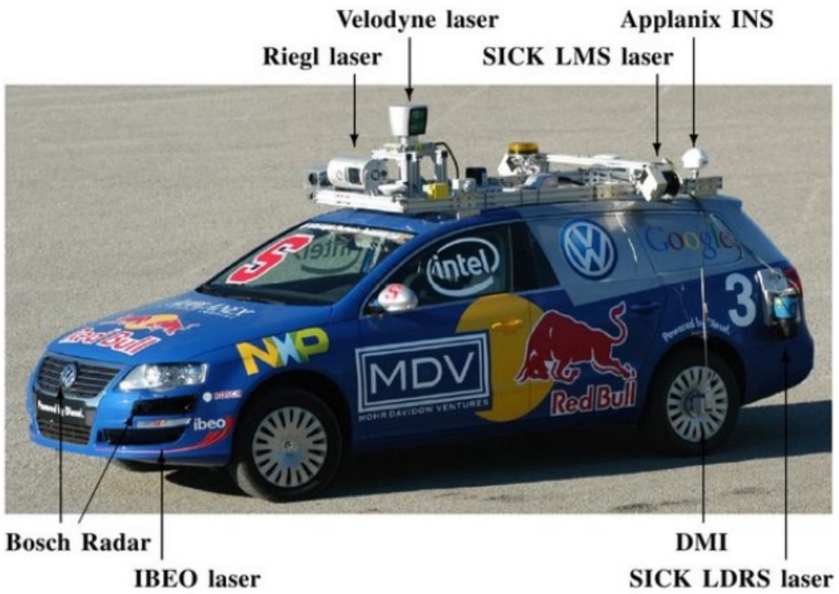
Inverse Reinforcement Learning

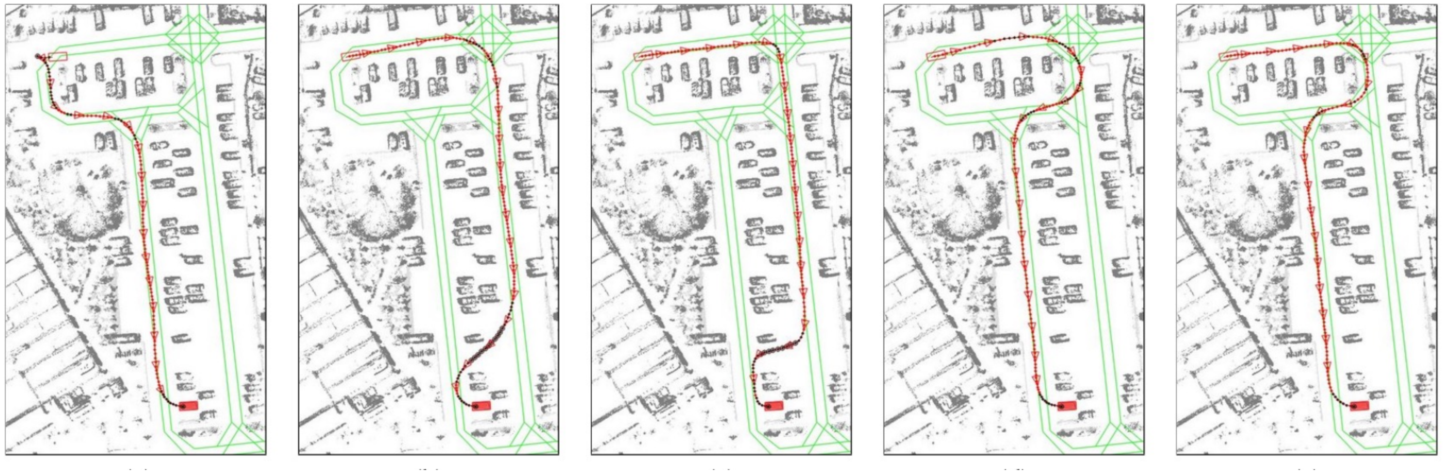
State: Known
Action: Known
Transition Dynamics: Unknown but can sample
Reward: **Unknown**
Expert policy: Unknown
Expert traces: **Known**

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

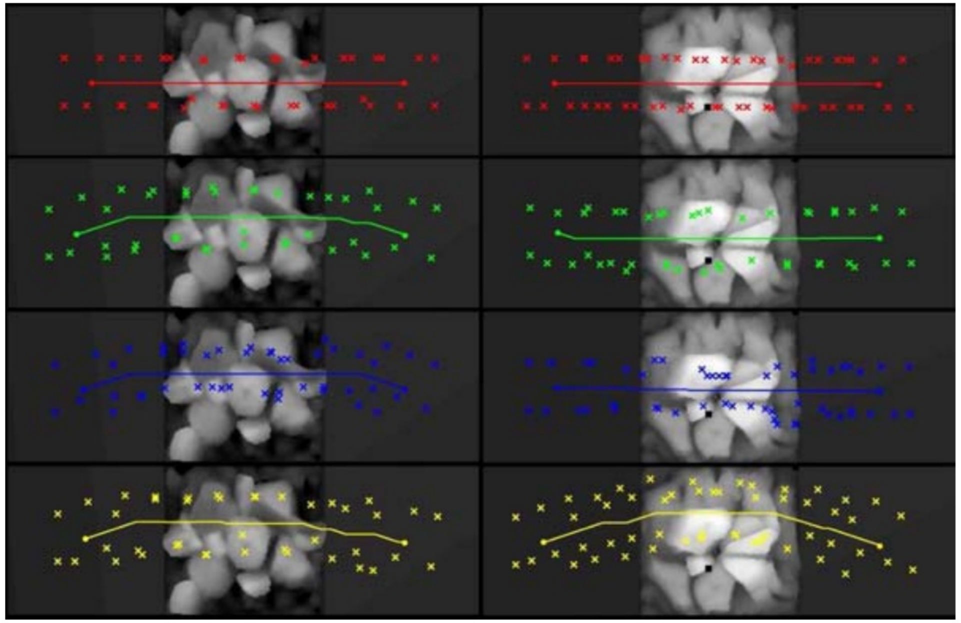


Inverse RL Applications



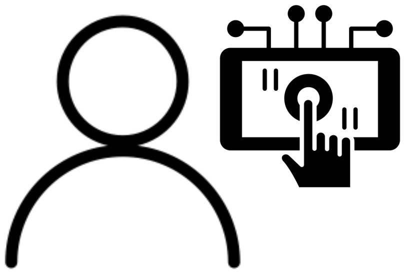


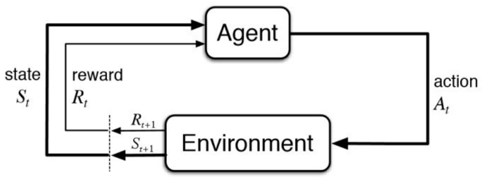
Inverse RL Applications



Why is this hard?

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





Inverse RL

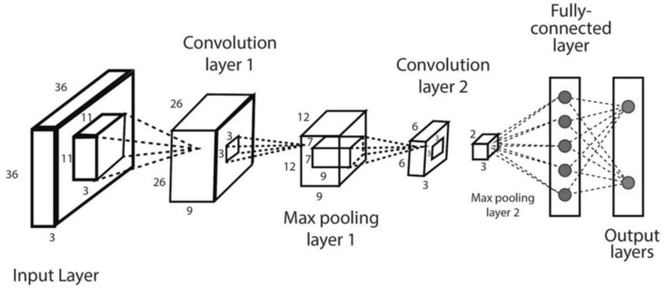
Reward Function

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

Can be parameterized by arbitrary function approximator

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



Lecture Outline

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

Inverse RL Problem Formulation

↓

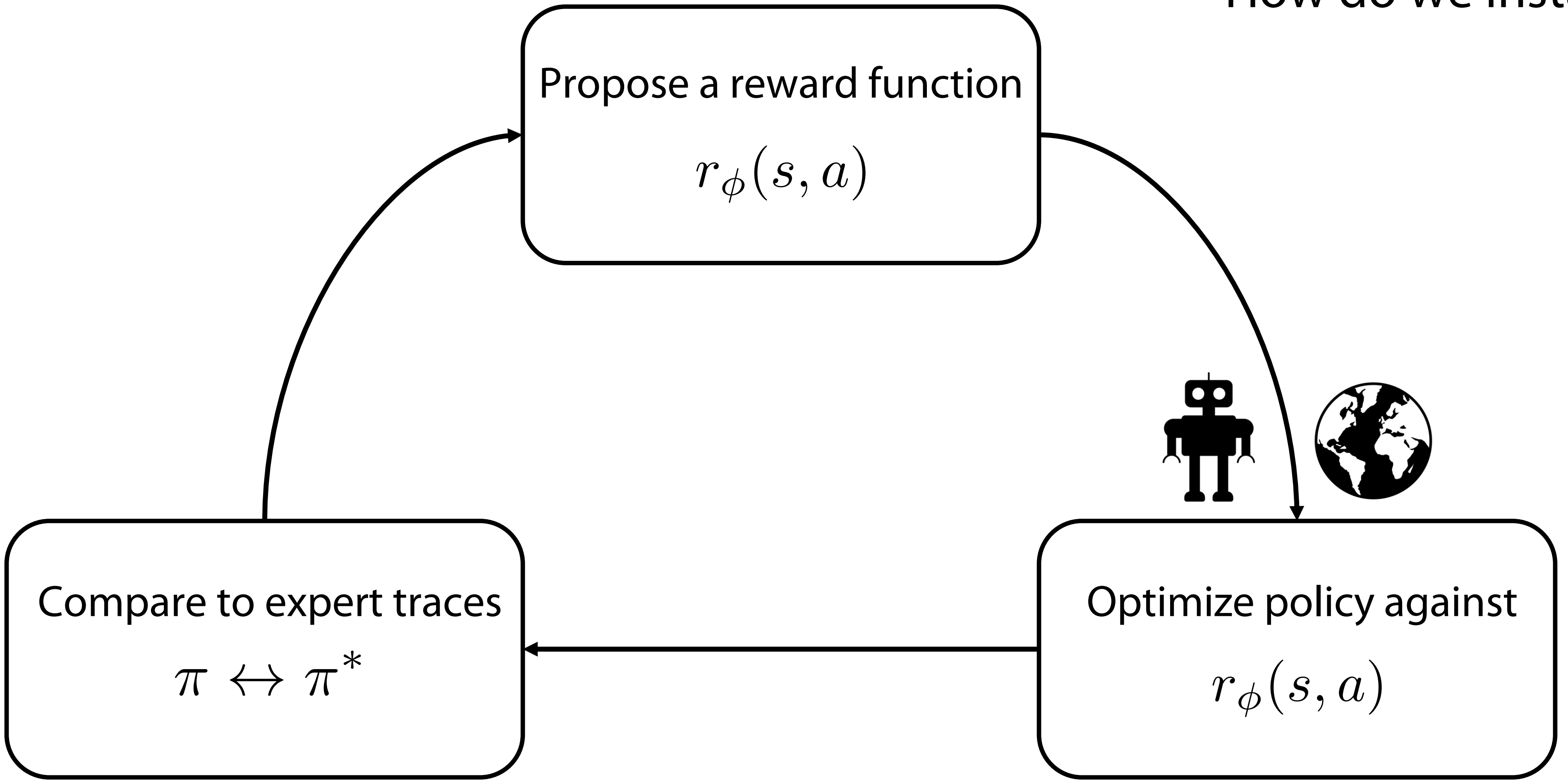
IRLv1 – max margin planning

↓

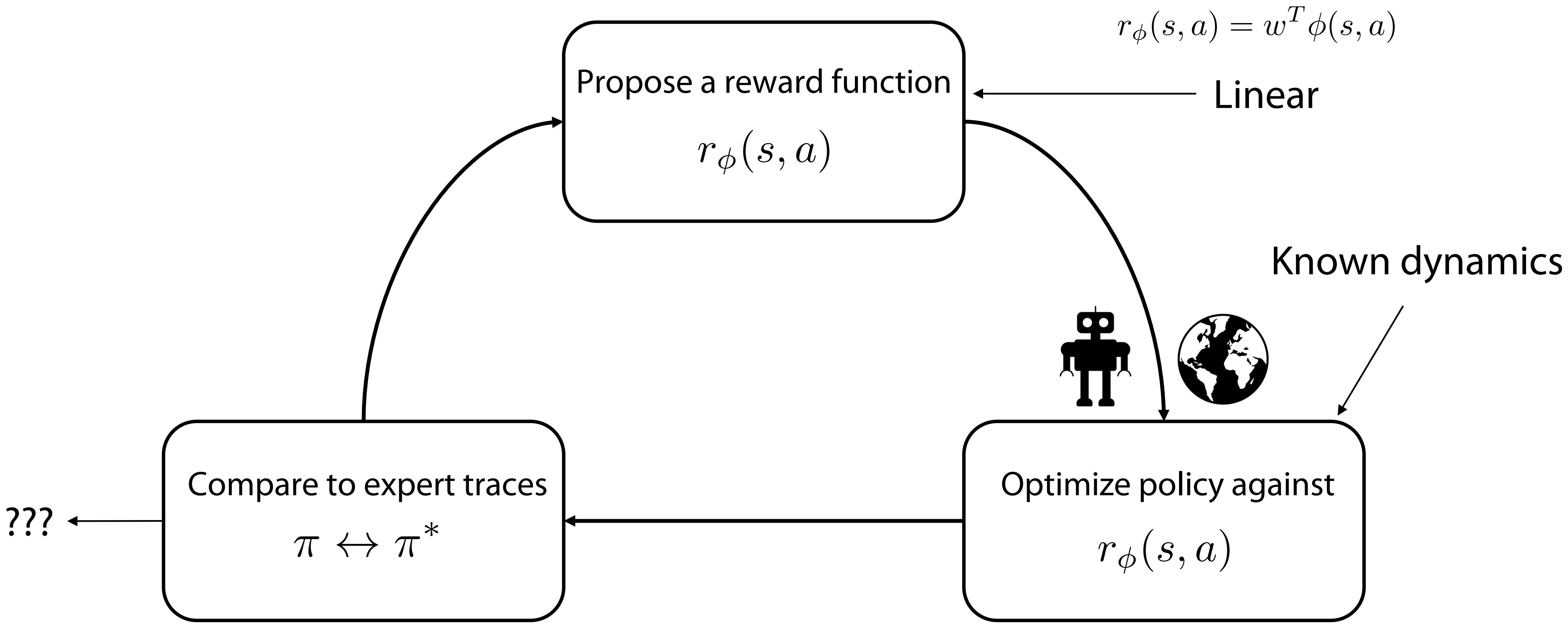
IRLv2 – max entropy IRL

A Formula for Inverse Reinforcement Learning

How do we instantiate?



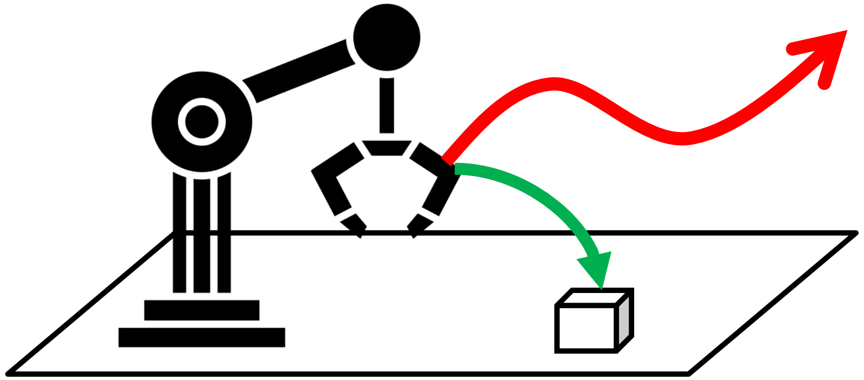
IRL v0 – Assumptions



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

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

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

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

Underdefined, w* = 0 trivially satisfies!

IRL v0 – What is a good reward function?

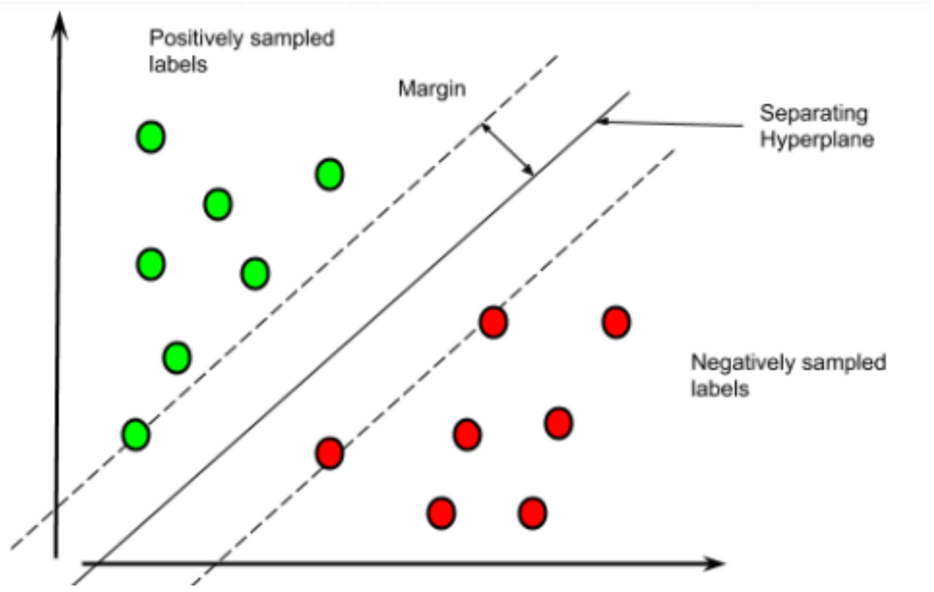
How do we tackle ambiguity?

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

$$\Downarrow$$

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

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



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

IRL v1 – Max Margin Feature Matching

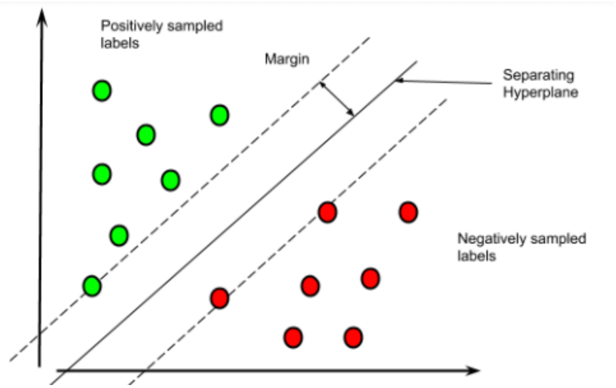
Choose w such that "margin" is maximized

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

Looks a lot like an SVM!

$$\Downarrow$$

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



What might the issues be →

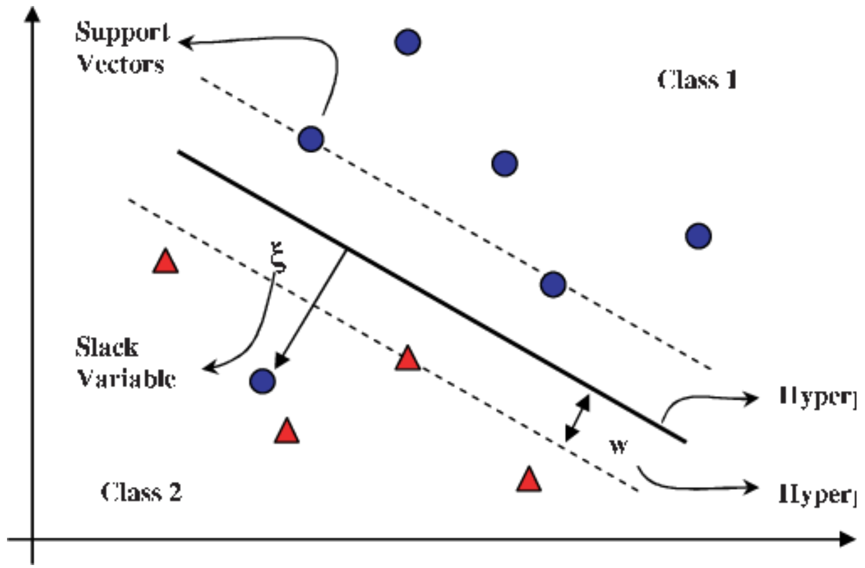
1. Uniform gap across all π, π^*
2. Noisily optimal may compromise the optimization

https://cs229.stanford.edu/lectures-spring2022/main_notes.pdf

IRL v1 – (Fancy) Max Margin Feature Matching

Maximum margin → Structured Max-Margin + Slack

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

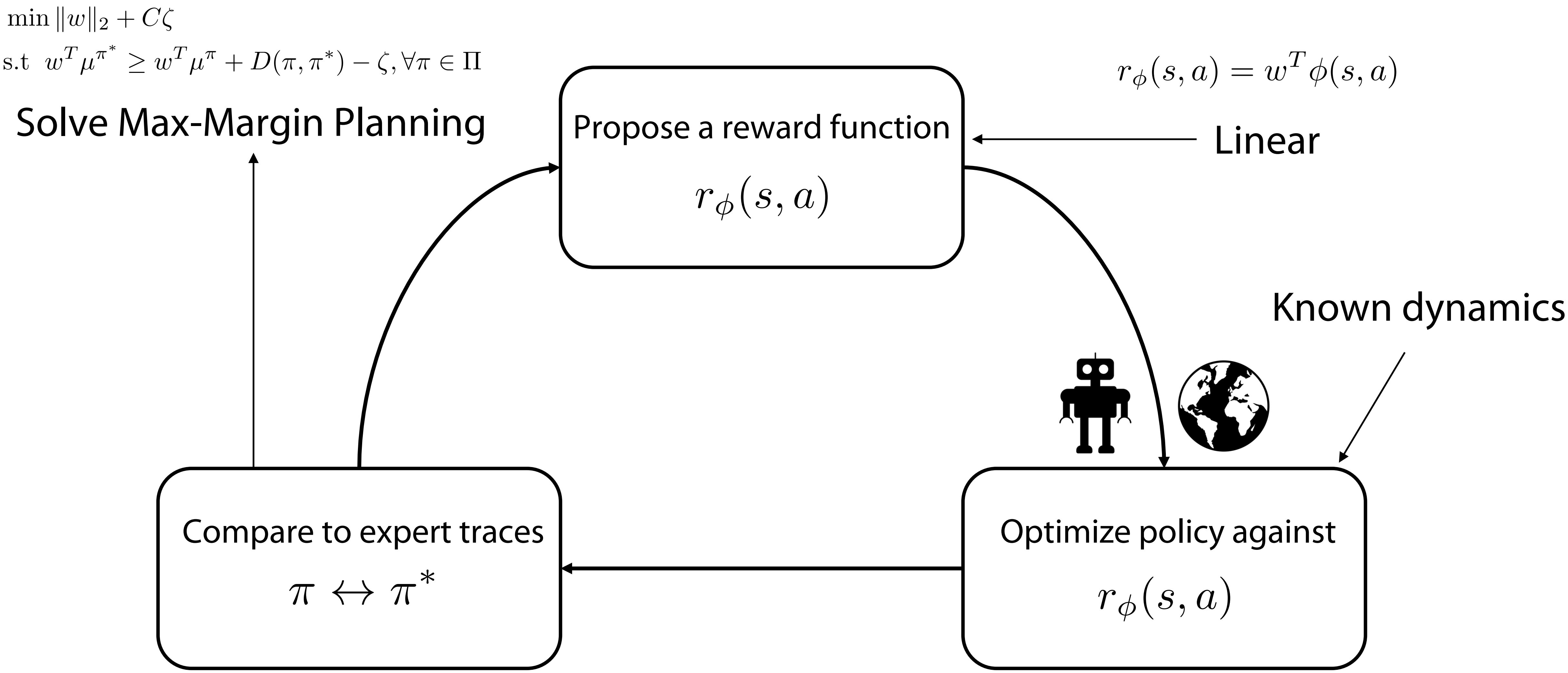


Bigger for more different policies

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

Slack allows for noisy optimality

IRL v1 – Max Margin Feature Matching



IRL v1 – Max Margin Feature Matching

1. Start with a random policy π_0
2. Find the w that optimizes

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

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

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

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

Inverse RL Problem Formulation

↓

IRLv1 – max margin planning

↓

IRLv2 – max entropy IRL

IRL v1 – Why this may not be enough?

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

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

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

Can we do better?

Aside: Feature Matching

Instead of focusing on the reward function, focus on the feature expectations

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

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

$$= \left| w^T \mu(\pi^*) - w^T \mu(\pi) \right|$$

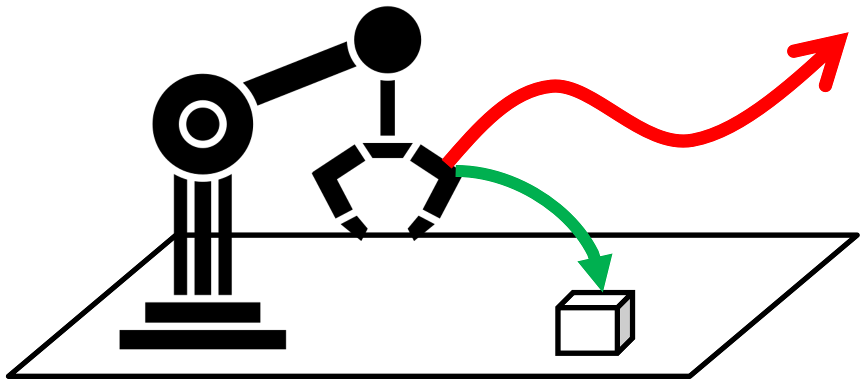
$$\leq \|w\|_2 \|\mu(\pi^*) - \mu(\pi)\|_2 \qquad \|w\|_2 < 1 \qquad \|\mu(\pi^*) - \mu(\pi)\|_2 < \epsilon$$

$\leq \epsilon$ => If average feature expectations are close, then values are close

Abbeel, Ng '04

Intuition on Feature Matching

Let's provide some intuition



Features - distance to object
end effector position
object orientation
….

Matching features probably means that behavior is roughly similar

From max margin to max-ent IRL

Two key ideas in maximum-entropy IRL:

1. Prefer good trajectories → Feature matching
2. Weight other trajectories equally to deal with ambiguity → Maximum entropy

<u>Notation:</u>

Trajectory distribution – $p(\tau)$

Feature expectations:

Policy $\mu(p) = \mathbb{E}_{p(\tau)}\left[\sum_t \gamma^t \phi(s_t, a_t)\right]$

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

$$\max_p \; \mathcal{H}(p(\tau)) = -\int p(\tau) \log p(\tau) d\tau$$ Max-entropy

$$\mu(p) = \mu(\pi^*)$$ Match features

$$\int p(\tau) = 1$$ Be a probability

Let's simplify

$$\max_p \ \mathcal{H}(p(\tau)) = -\int p(\tau)\log p(\tau)d\tau$$ Max-entropy

$$\mu(p) = \mu(\pi^*)$$ Match features

$$\int p(\tau) = 1$$ Be a probability

Set up the Lagrangian

$$\max_p \min_{w,\lambda} \mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

$$\min_{w,\lambda} \max_p \mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

Solve wrt p

Solve wrt w, λ

Connect the dots!

Let's simplify – solve for p

Set up the Lagrangian

$$\max_{p} \min_{w,\lambda} \mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

$$\min_{w,\lambda} \max_{p} \mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

Solve wrt p

$$\nabla_p \left[\mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1) \right] = 0$$

$$\nabla_p \left[-\int p(\tau) \log p(\tau) d\tau + w^T(\int p(\tau)\mu(\tau)d\tau - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1) \right] = 0$$

$$-\log p(\tau) - 1 + w^T\mu(\tau) - \lambda = 0$$

$$p(\tau) = \exp(-1 + w^T\mu(\tau) - \lambda)$$

Intuition: $p(\tau)$ is proportional to the exponential reward of a trajectory $w^T \mu(\tau)$

Let's simplify – solve for λ

$$\min_{w,\lambda} \max_{p} \mathcal{H}(p(\tau)) + w^T(\mu(p) - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

$$p(\tau) = \exp(-1 + w^T\mu(\tau) - \lambda)$$

$$\Downarrow$$

$$\min_{w,\lambda} - \int p(\tau)\log p(\tau)d\tau + w^T(\int p(\tau)\mu(\tau)d\tau - \mu(\pi^*)) - \lambda(\int p(\tau)d\tau - 1)$$

$$\min_{w,\lambda} - \int p(\tau)(-1 + \cancel{w^T\mu(\tau)} - \cancel{\lambda})d\tau + w^T(\int \cancel{p(\tau)\mu(\tau)d\tau} - \mu(\pi^*)) - \lambda(\int \cancel{p(\tau)d\tau} - 1)$$

$$\min_{w,\lambda} \int p(\tau)d\tau - w^T\mu(\pi^*) + \lambda$$

$$\min_{w,\lambda} \int \exp(-1 + w^T\mu(\tau) - \lambda)d\tau - w^T\mu(\pi^*) + \lambda \quad = \min_{w,\lambda} \exp(-1-\lambda) \underbrace{\int \exp(w^T\mu(\tau))d\tau}_{Z} - w^T\mu(\pi^*) + \lambda$$

$$\nabla_\lambda \left[\exp(-1-\lambda)Z - w^T\mu(\pi^*) + \lambda\right] = 0 \quad \Longrightarrow \exp(-1-\lambda) = \frac{1}{Z}$$

$$\Downarrow$$

$$\min_{w} 1 - w^T\mu(\pi^*) + \lambda$$

$$= \min_{w} \log Z - w^T\mu(\pi^*)$$

Ok – let's unpack what we have so far

$$\max_p \ \mathcal{H}(p(\tau)) = -\int p(\tau) \log p(\tau) d\tau$$ Max-entropy

$$\mu(p) = \mu(\pi^*)$$ Match features

$$\int p(\tau) = 1$$ Be a probability

Solve wrt p

$$p(\tau) = \exp(-1 + w^T \mu(\tau) - \lambda)$$

Solve wrt λ

$$Z = \int \exp(w^T \mu(\tau)) d\tau \qquad \exp(-1 - \lambda) = \frac{1}{Z}$$

Objective reduces to $\min_w \log Z - w^T \mu(\pi^*)$

⇩

Solve wrt *w* Find reward function!

Turns out this has nice intuitive properties

$$\max_p \; \mathcal{H}(p(\tau)) = -\int p(\tau) \log p(\tau) d\tau$$ Max-entropy

$$\mu(p) = \mu(\pi^*)$$ Match features

$$\int p(\tau) = 1$$ Be a probability

⇩

Objective reduces to $\min_w \log Z - w^T \mu(\pi^*)$

$$Z = \int \exp(w^T \mu(\tau)) d\tau$$

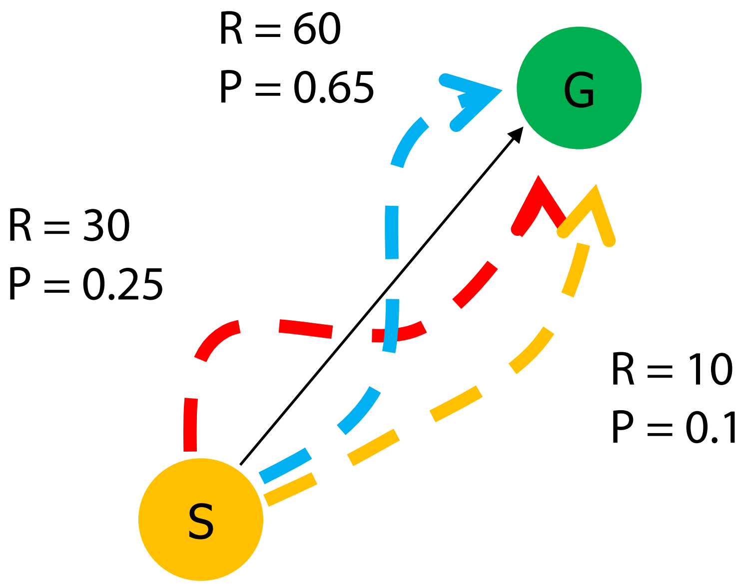
⇩

$$\max_w \log \frac{\exp(w^T \mu(\pi^*))}{\int \exp(w^T \mu(\tau)) d\tau}$$

Maximum likelihood with exponential family

$$= \max_w \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[\log \frac{\exp(w^T \mu(\tau^*))}{\int \exp(w^T \mu(\tau)) d\tau} \right]$$

Intuition: trajectories are chosen **proportional** to their reward



Turns out this has nice intuitive properties

$$\max_p \; \mathcal{H}(p(\tau)) = -\int p(\tau) \log p(\tau) d\tau$$ Max-entropy

$$\mu(p) = \mu(\pi^*)$$ Match features

$$\int p(\tau) = 1$$ Be a probability

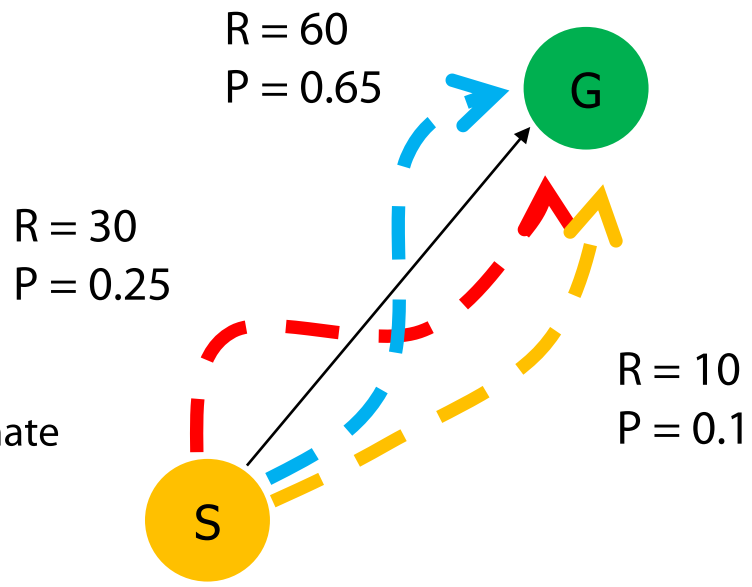
⇓

Maximum likelihood with exponential family

$$\max_w \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[\log \frac{\exp(w^T \mu(\tau^*))}{\int \exp(w^T \mu(\tau)) d\tau} \right]$$

→ Hard to estimate

Intuition: trajectories are chosen **proportional** to their reward



⇓

Let's solve with gradient descent! Has a nice tractable form

Maximum likelihood estimation of w

$$\max_w \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[\log \frac{\exp(w^T \mu(\tau^*))}{\int \exp(w^T \mu(\tau)) d\tau} \right]$$

$$J(w) = \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[w^T \mu(\tau^*) \right] - \log \int \exp(w^T \mu(\tau)) d\tau$$

Painful to estimate log integral

Gradient has a much nicer form

$$\nabla J(w) = \nabla_w \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[w^T \mu(\tau^*) \right] - \nabla_w \log \int \exp(w^T \mu(\tau)) d\tau$$

$$\nabla J(w) = \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[\nabla_w w^T \mu(\tau^*) \right] - \frac{\int \exp(w^T \mu(\tau)) \nabla_w w^T \mu(\tau) d\tau}{\int \exp(w^T \mu(\tau)) d\tau}$$

$$\nabla J(w) = \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[\nabla_w w^T \mu(\tau^*) \right] - \int p_w^*(\tau) \nabla_w w^T \mu(\tau) d\tau$$

$$\nabla J(w) = \mathbb{E}_{\tau^* \sim \mathcal{D}^e} \left[\nabla_w w^T \mu(\tau^*) \right] - \mathbb{E}_{\tau \sim p_w^*(\tau)} \left[\nabla_w w^T \mu(\tau) \right]$$

Push up on data

Push down on policy

Soft optimal policy for

$$r_w(s_t, a_t) = w^T \phi(s_t, a_t)$$

$$p_w^*(\tau) = \frac{\exp(w^T \mu(\tau))}{\int \exp(w^T \mu(\tau')) d\tau'}$$

IRLv2 – Maximum Entropy Inverse RL

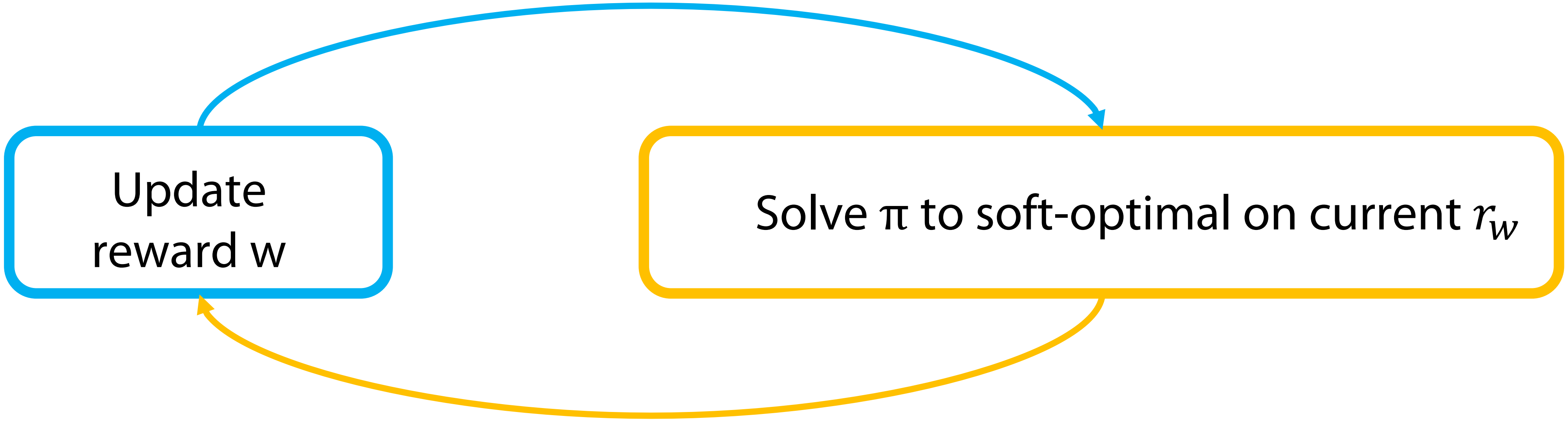
$$\nabla J(w) = \mathbb{E}_{\tau^* \sim \mathcal{D}^e}\left[\nabla_w w^T \mu(\tau^*)\right] - \mathbb{E}_{\tau \sim p_w^*(\tau)}\left[\nabla_w w^T \mu(\tau)\right]$$

Push up on data   Push down on policy

Soft optimal policy for

$$r_w(s_t, a_t) = w^T \phi(s_t, a_t)$$

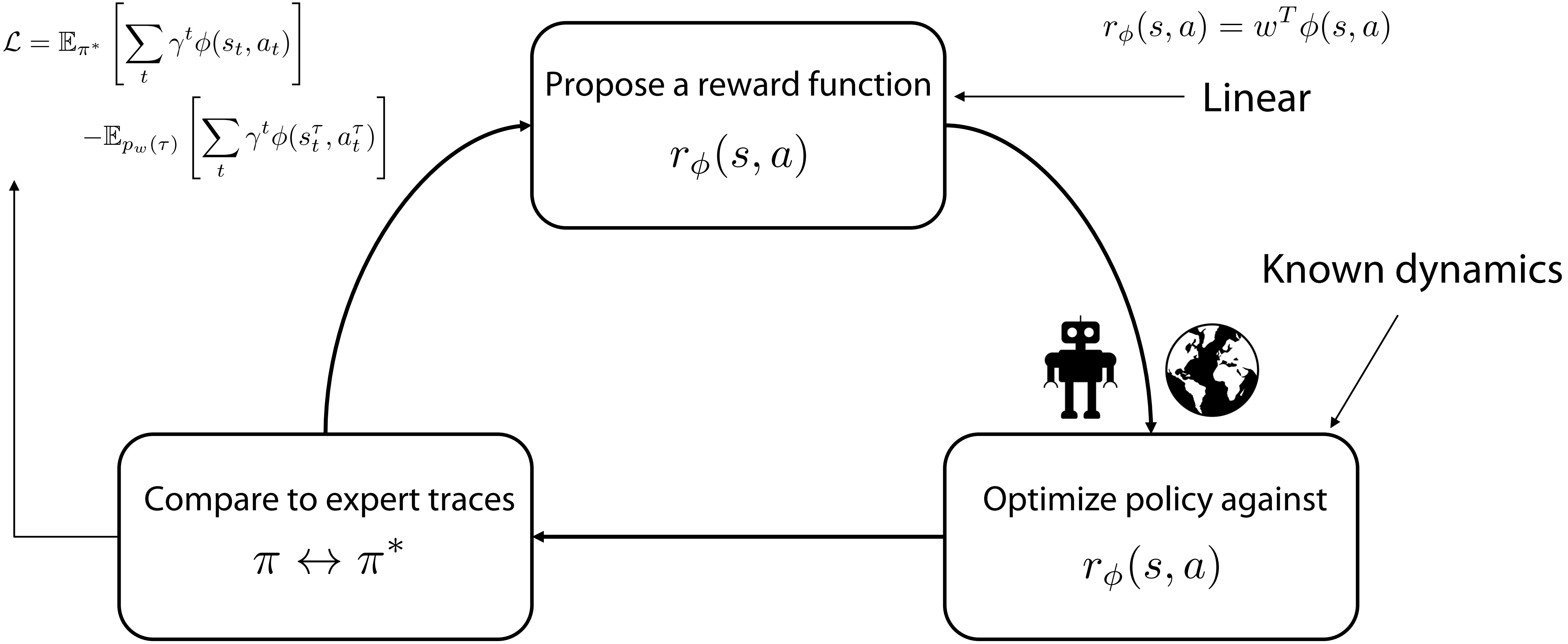
$$p_w^*(\tau) = \frac{\exp(w^T \mu(\tau))}{\int \exp(w^T \mu(\tau'))d\tau'}$$



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)

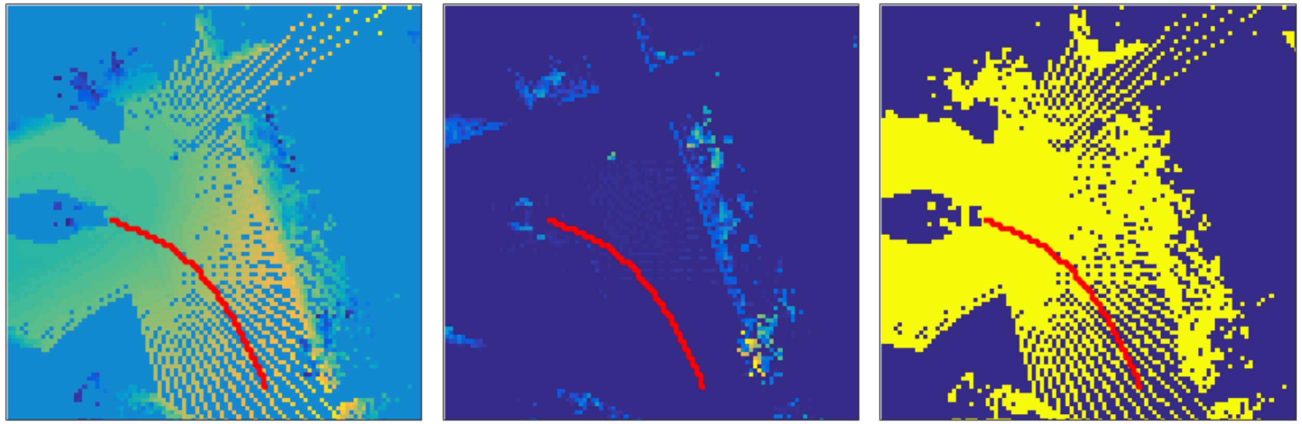
1. Start with a random policy π_0 and weight vector w
2. Find the "soft" optimal policy under w – $p_w(\tau)$
3. Take a gradient step on w

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

4. Repeat

Output the optimal reward function w*

Max-Ent IRL in Action



Lecture Outline

Model based RL v2 → uncertainty based models

↓

Model based RL v3 → policy optimization with models

↓

Model based RL v4 → latent space models with images

↓

Inverse RL Problem Formulation

↓

IRLv1 – max margin planning

↓

IRLv2 – max entropy IRL

Class Structure