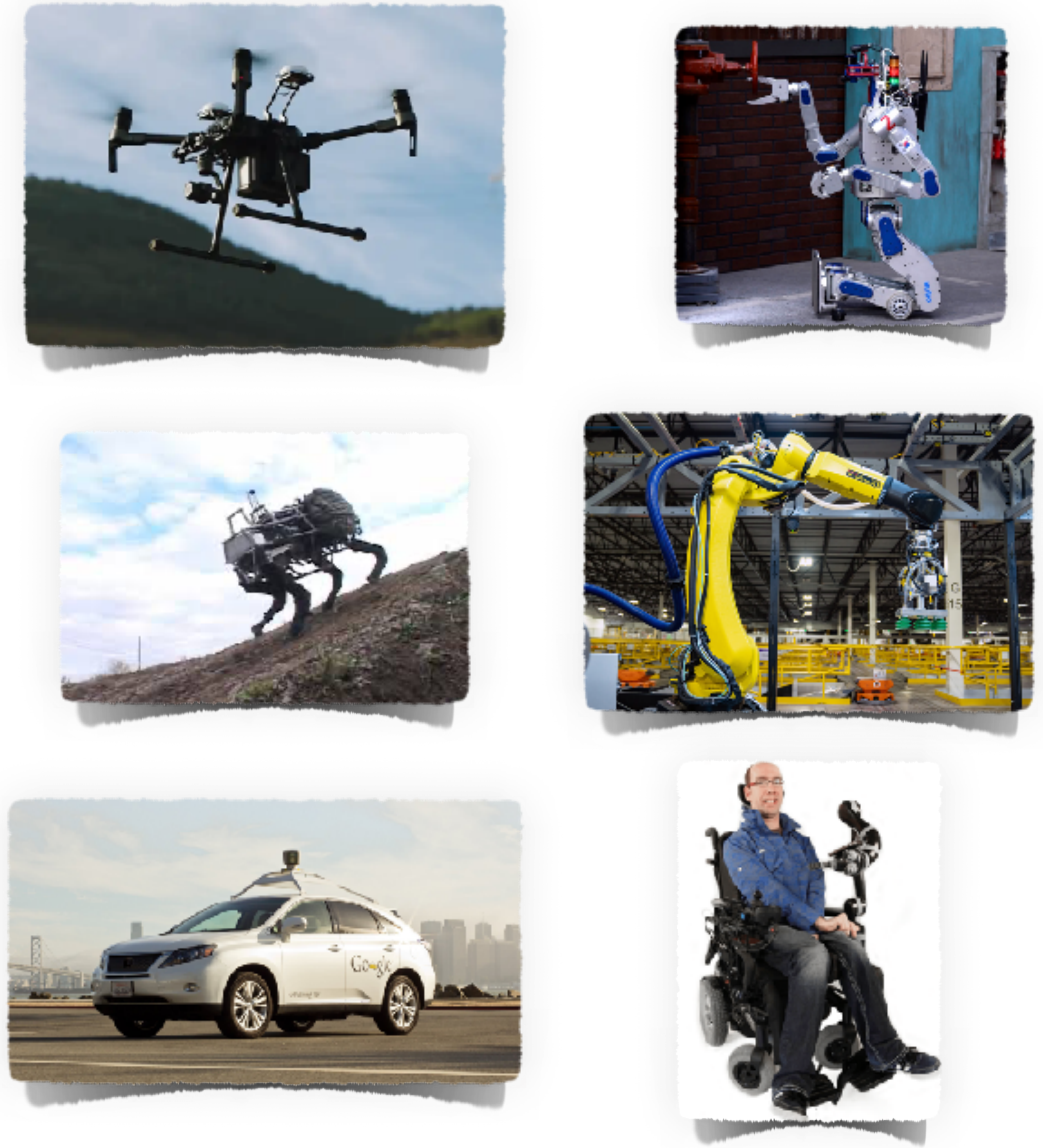
Learning for Robot Decision Making: The Big Picture

Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

How should robots learn to make good decisions?

WHY ask this question?



Take *any*
robot application

Formulate as a Markov Decision Problem (MDP)

Solve MDPs using an all-purpose toolkit

(Imitation/Reinforcement learning, Model based/free)

Deploy learners in real-world

(Safety, distribution shift, value alignment)

HOW can we answer this question?

Solve

How do you want to represent your policy?
Model-based? Model-free?
Learning: Data? Loss?

Formulate

What is the MDP?
Discrete/Stochastic/Time?
What is known/unknown?

Application

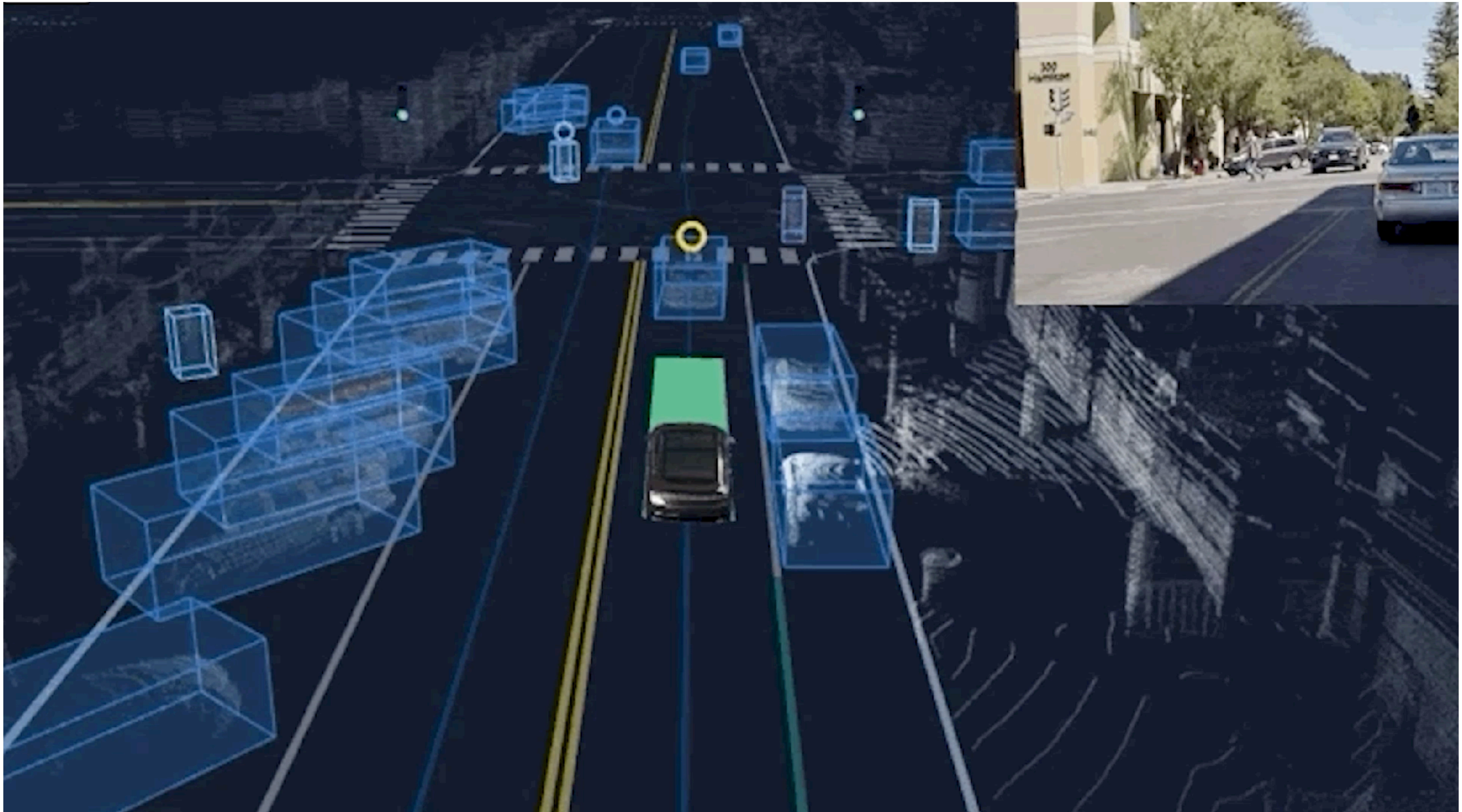
What is the robot?
What is the task?
What are the metrics?
What is good enough?

Backward

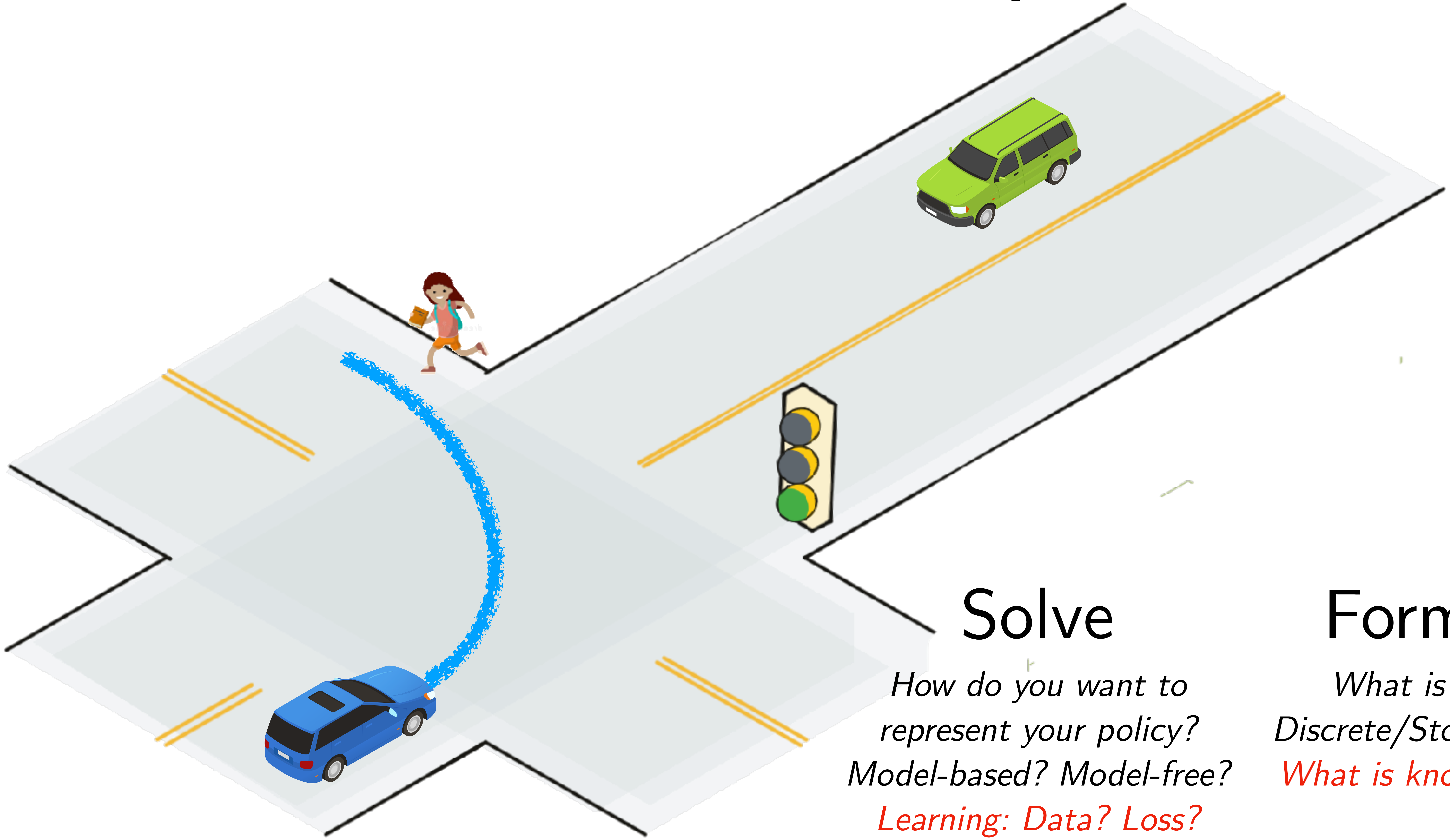
Forward

Activity!

Let's solve the Unprotected Left Turn



Let's solve the Unprotected Left Turn



Solve

How do you want to represent your policy?
Model-based? Model-free?
Learning: Data? Loss?

Formulate

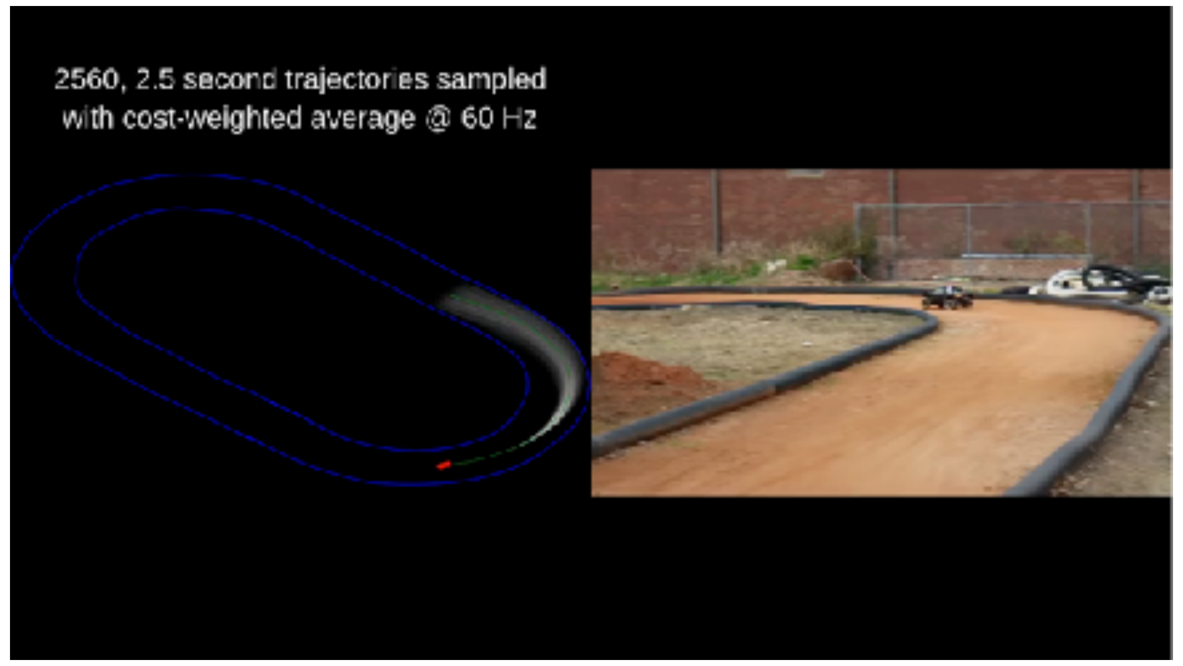
What is the MDP?
Discrete/Stochastic/Time?
What is known/unknown?

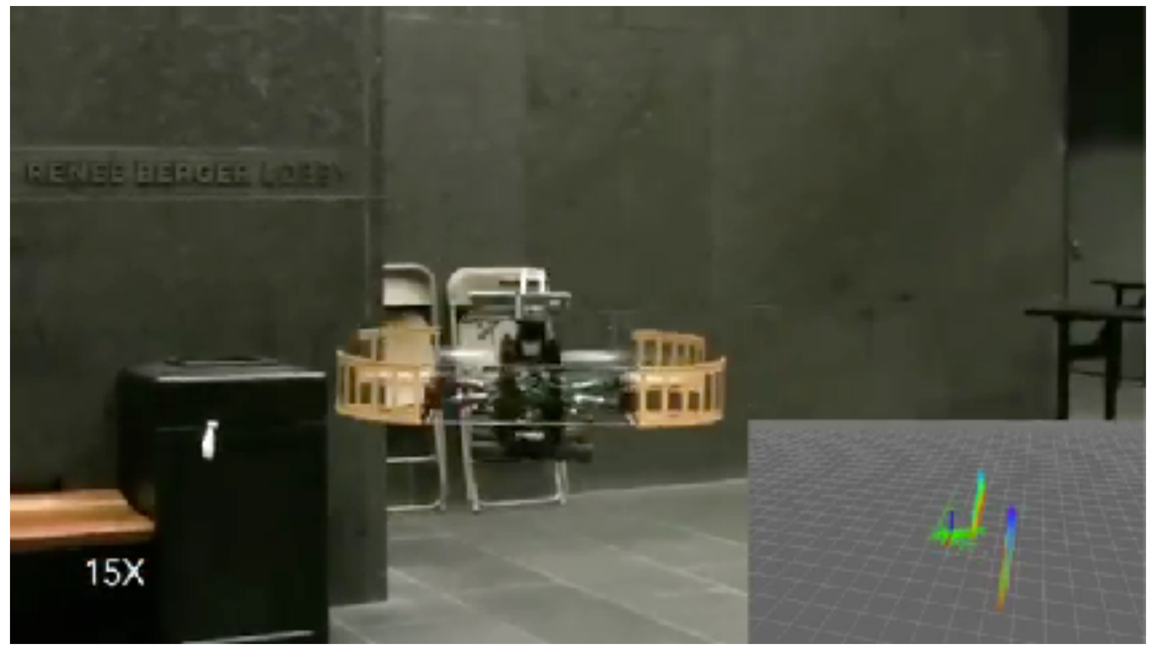
Application

What is the robot?
What is the task?
What are the metrics?
What is good enough?

Backward

We have worked through many applications in this class …

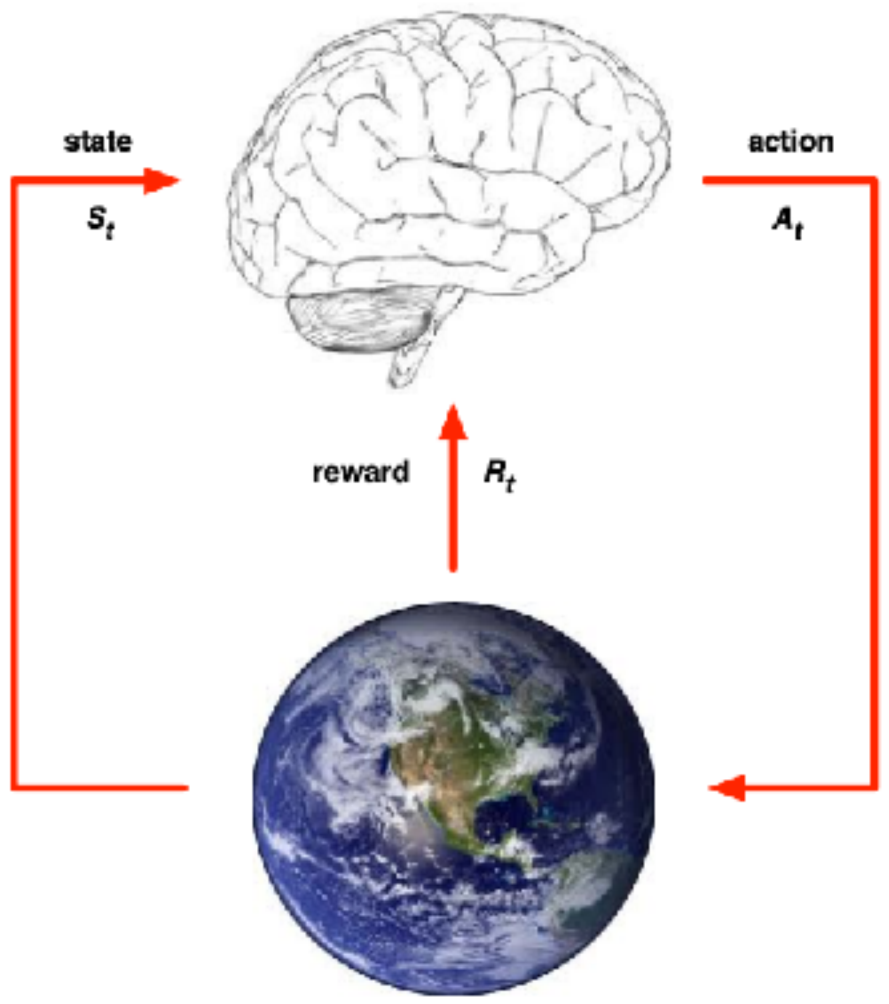




Model-Based OR Model Free?

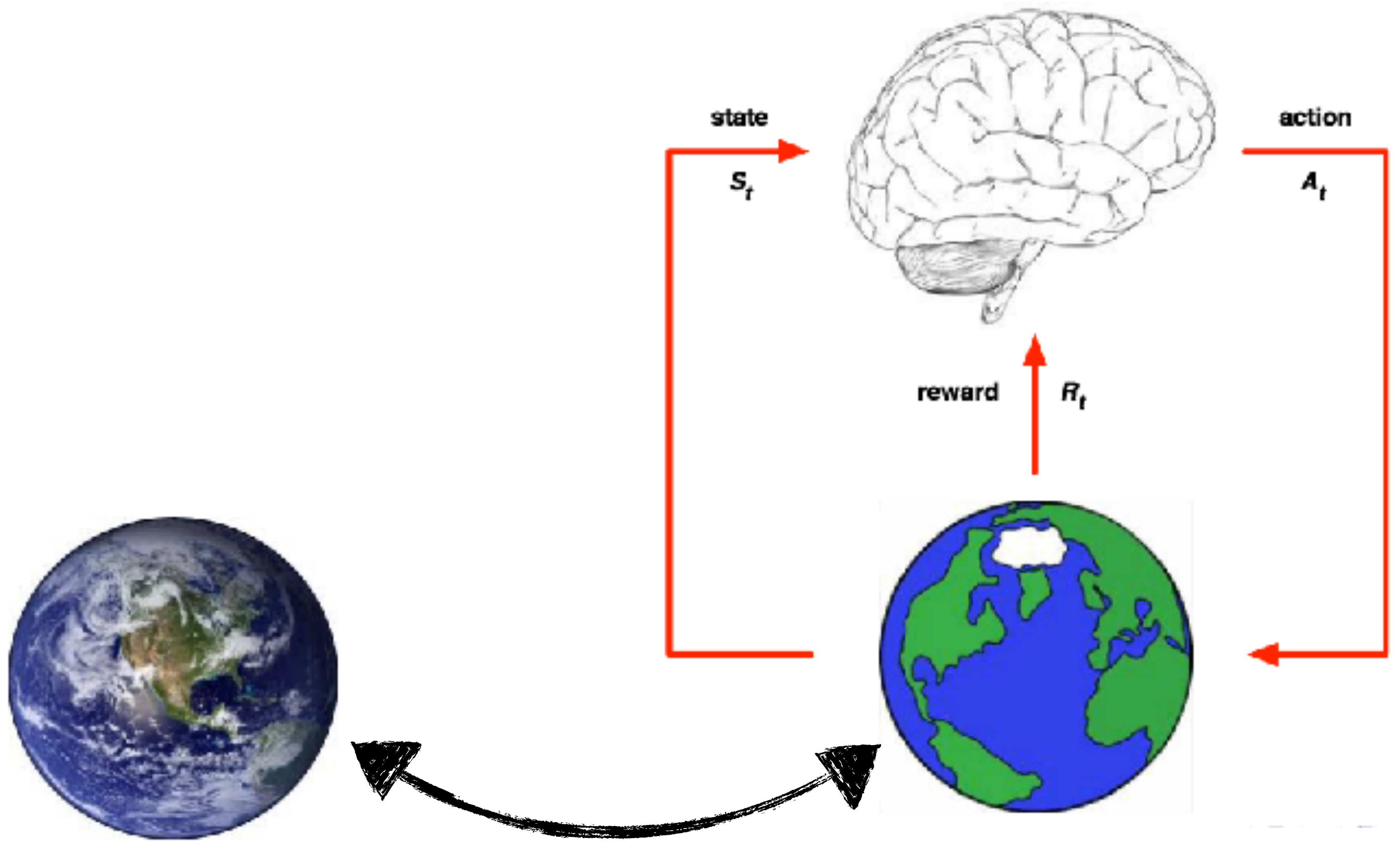
Model Free

Directly learn
π or $Q(s,a)$



Model Based

Learn a model
$P(s'|s,a)$, plan with
model to find π



Model-Based OR Model Free?

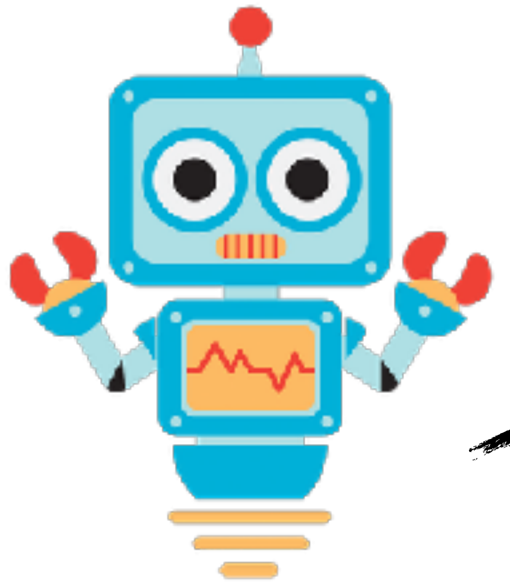
Model Free	Model Based
There exists a good enough reactive policy	You need to reason about many likely options
State space is too big to search exhaustively	Small state space / compressible state space

Model-Based ~~OR~~ *AND* Model Free

Model Based

Model Free

$$s' = \hat{M}(s, a)$$



Use model to plan

$$Q_M^*(s_T, a_T)$$

Use model-free terminal value function

HOW can we answer this question?

Solve

How do you want to represent your policy?
Model-based? Model-free?
Learning: Data? Loss?

Formulate

What is the MDP?
Discrete/Stochastic/Time?
What is known/unknown?

Application

What is the robot?
What is the task?
What are the metrics?
What is good enough?

Backward

Forward

HOW can we answer this question?

Solve

How do you want to represent your policy?
Model-based? Model-free?
Learning: Data? Loss?

Formulate

What is the MDP?
Discrete/Stochastic/Time?
What is known/unknown?

Application

What is the robot?
What is the task?
What are the metrics?
What is good enough?

Backward

Forward

Goal

Minimize value difference between robot and the human

PDL

Apply Bellman, express perf. difference as sum of Q value differences on states the robot visits

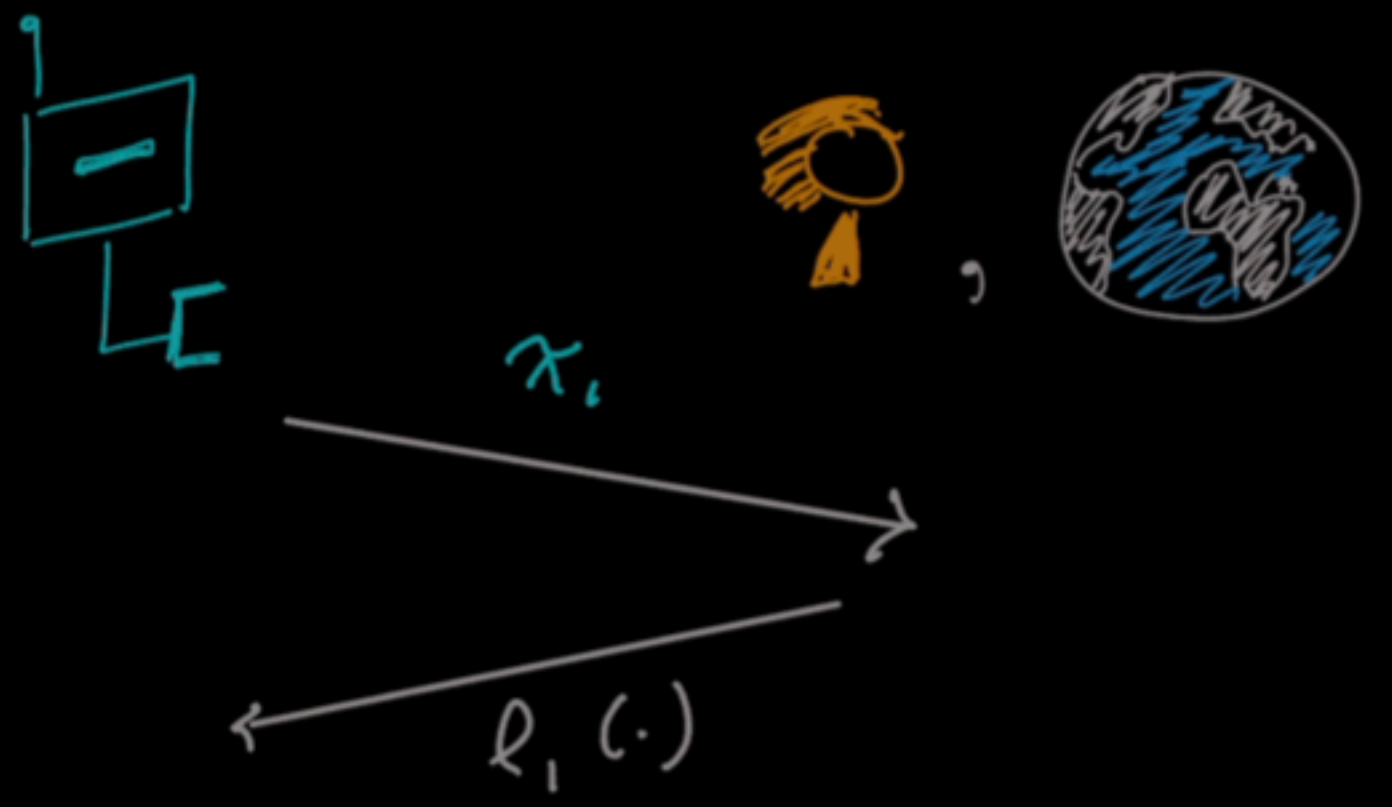
A Game

Q values are not known, must be estimated from feedback from human or the world.
Create a game between robot policy and Q value estimate, solve via no-regret online learning

5 Levels

of

Robot Learning



$$\min_{\substack{\pi \\ \text{ROBOT}}} \max_{\substack{Q^* \\ \text{ACTION VALUE}}} \sum_{i=1}^{N} Q^*(s, \pi(s)) - Q^*(s, \pi^*(s))$$

FIND A POLICY $\pi : S \to A$ SUCH THAT THE

VALUE DIFFERENCE

$$ \underbrace{J(\pi)}_{\text{ROBOT}} - \underbrace{J(\pi^*)}_{\text{HUMAN}} $$

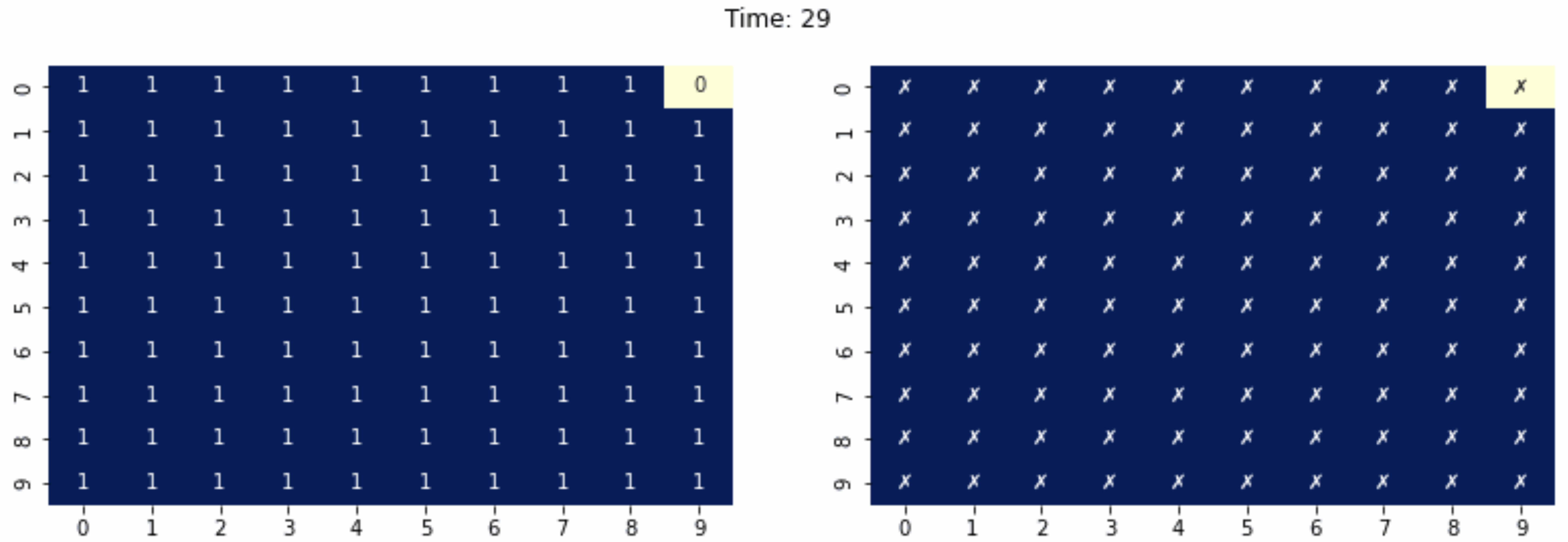
FOR ANY VALUE $J(\cdot)$ THE HUMAN CARES ABOUT

$$

$$V^*(s_t) = \min_{a_t}\left[\underset{??}{C}(s_t, a_t) + \mathbb{E}_{s_{t+1}\sim \underset{??}{P}(\cdot \mid s_t, a_t)} V^*(s_{t+1}) \right]$$

LEARN OPTIMAL VALUE BY INTERACTIONS W/ HUMAN + WORLD

Two Fundamental Approaches

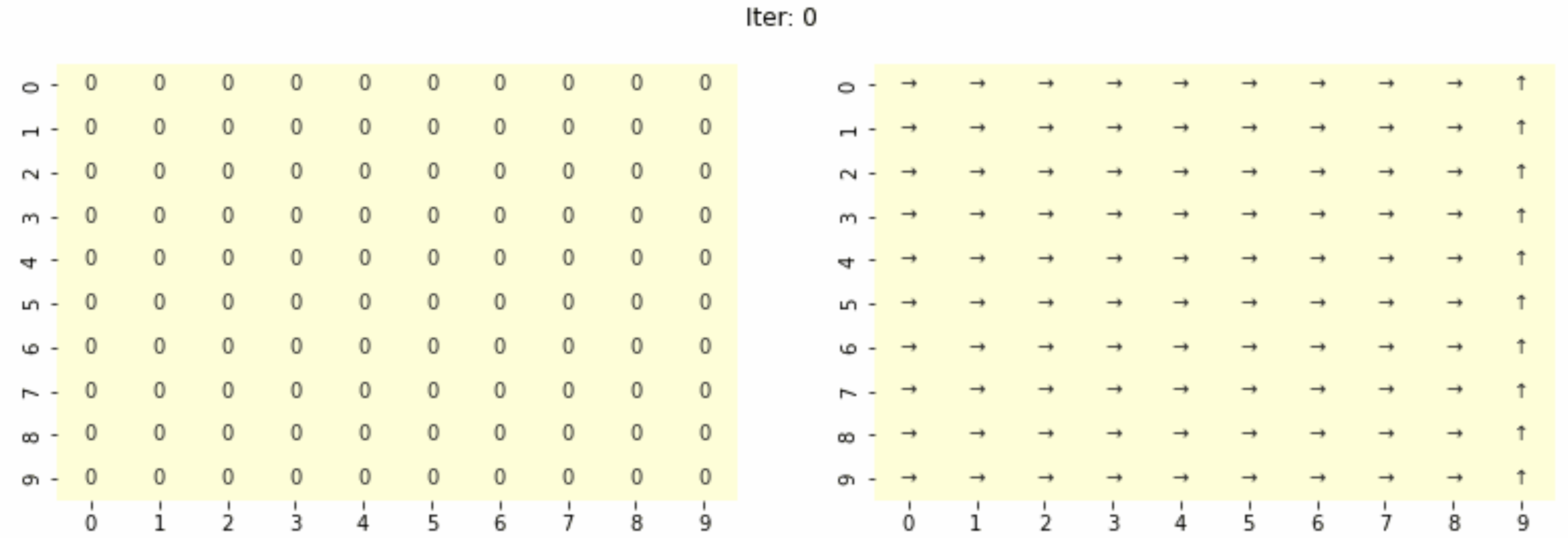


$V^*(s_t) = \min_a[c(s_t, a) + V^*(s_{t+1})]$ $\pi^*(s_t) = \arg\min_a[c(s_t), a) + V^*(s_{t+1})]$

Value Iteration

Iterate over *optimal* value

min() operator in value iteration step



$V^\pi(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s')]$ $\pi^+(s) = \arg\min c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s')]$

Policy Iteration

Evaluate value of current policy, then improve

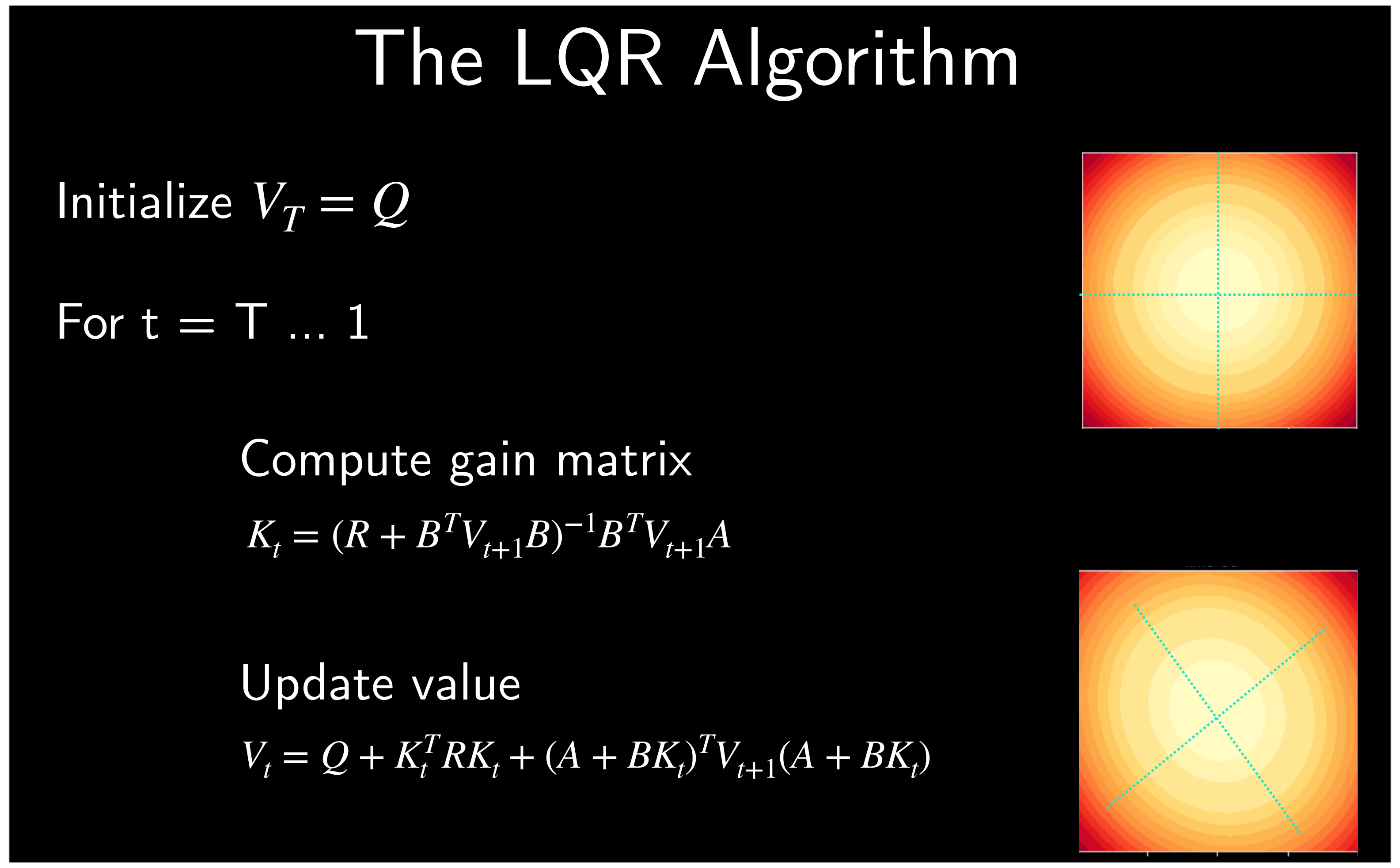
min() operator in policy improvement

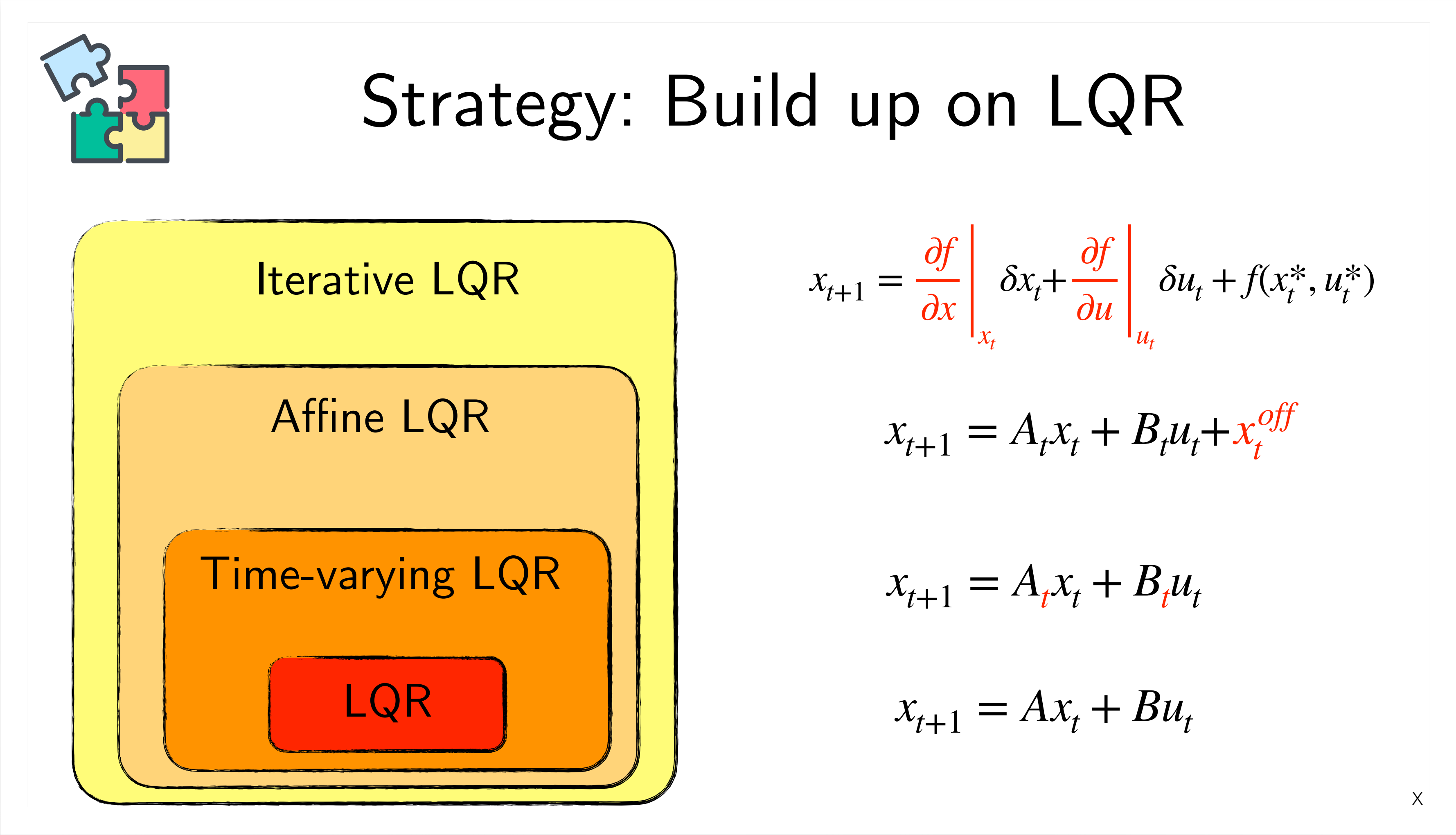
How do we scale these approaches?

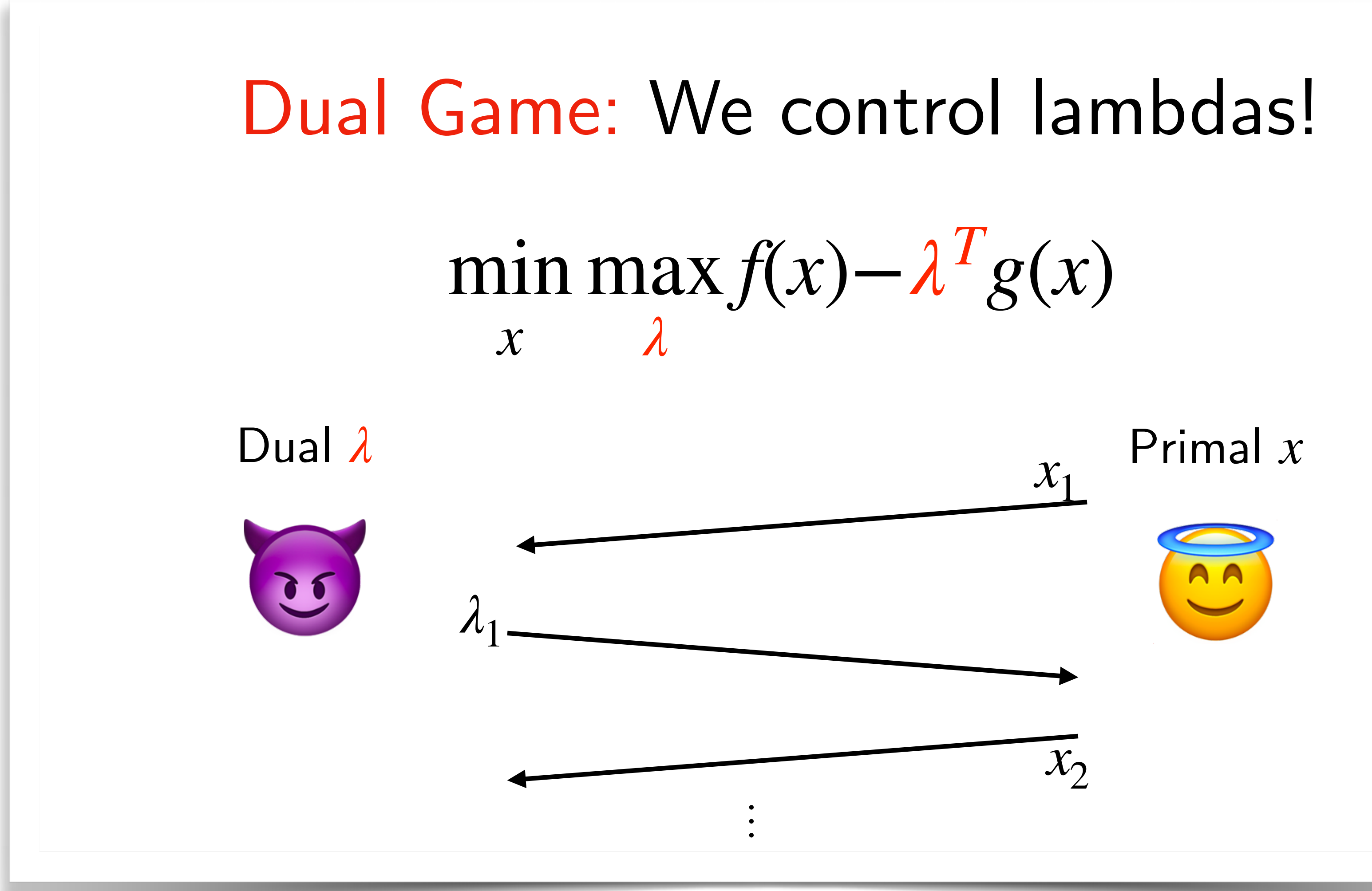
For continuous MDP (but linear)?

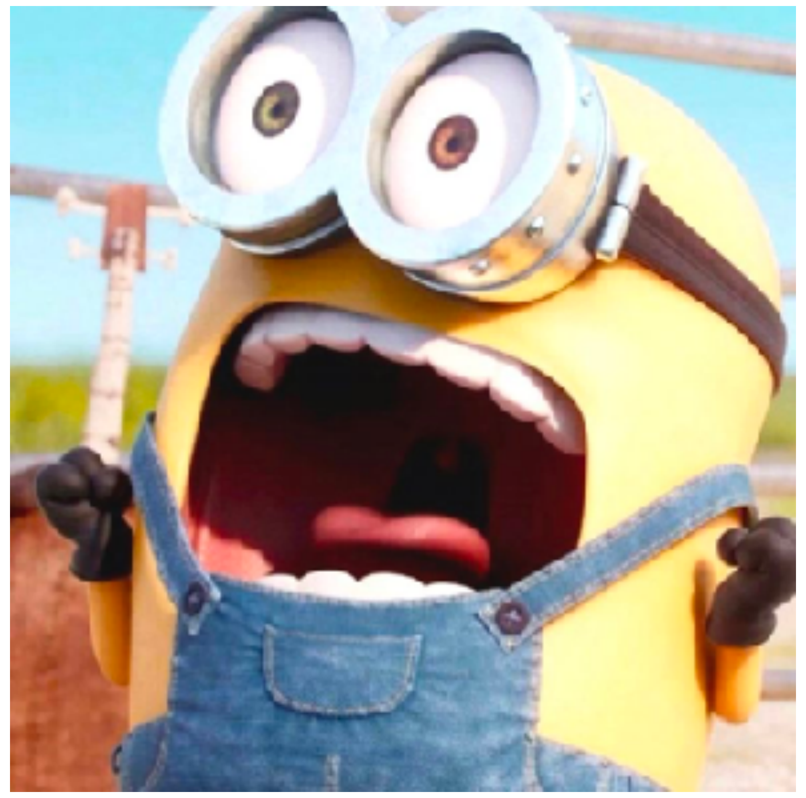
For non-linear MDP?

Handle constraints?





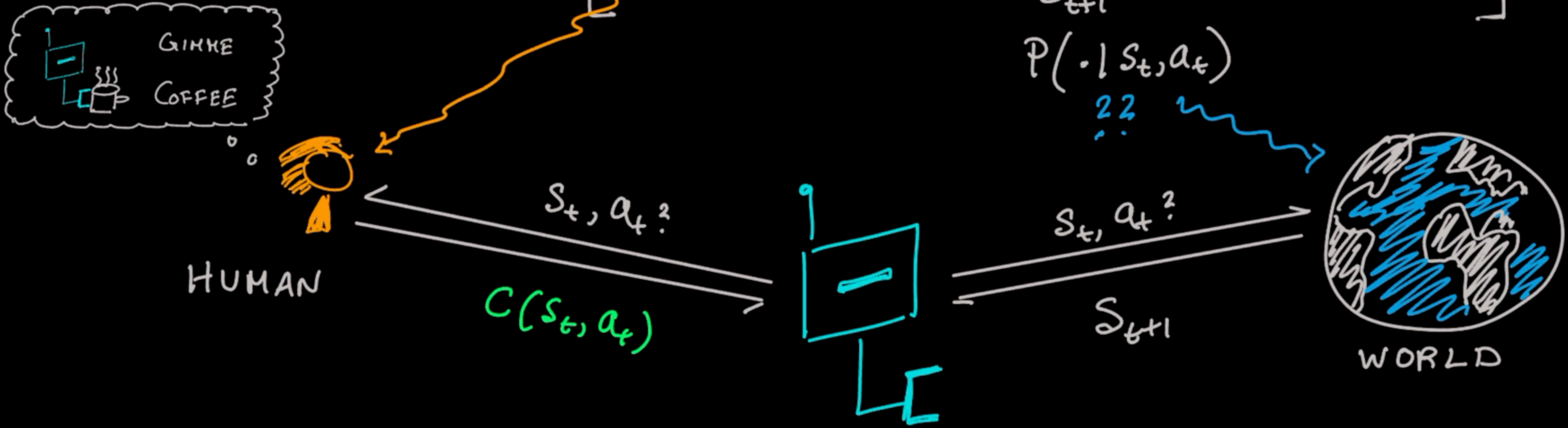




… What if your MDP is really complex?

Large state space, stochastic, continuous actions …

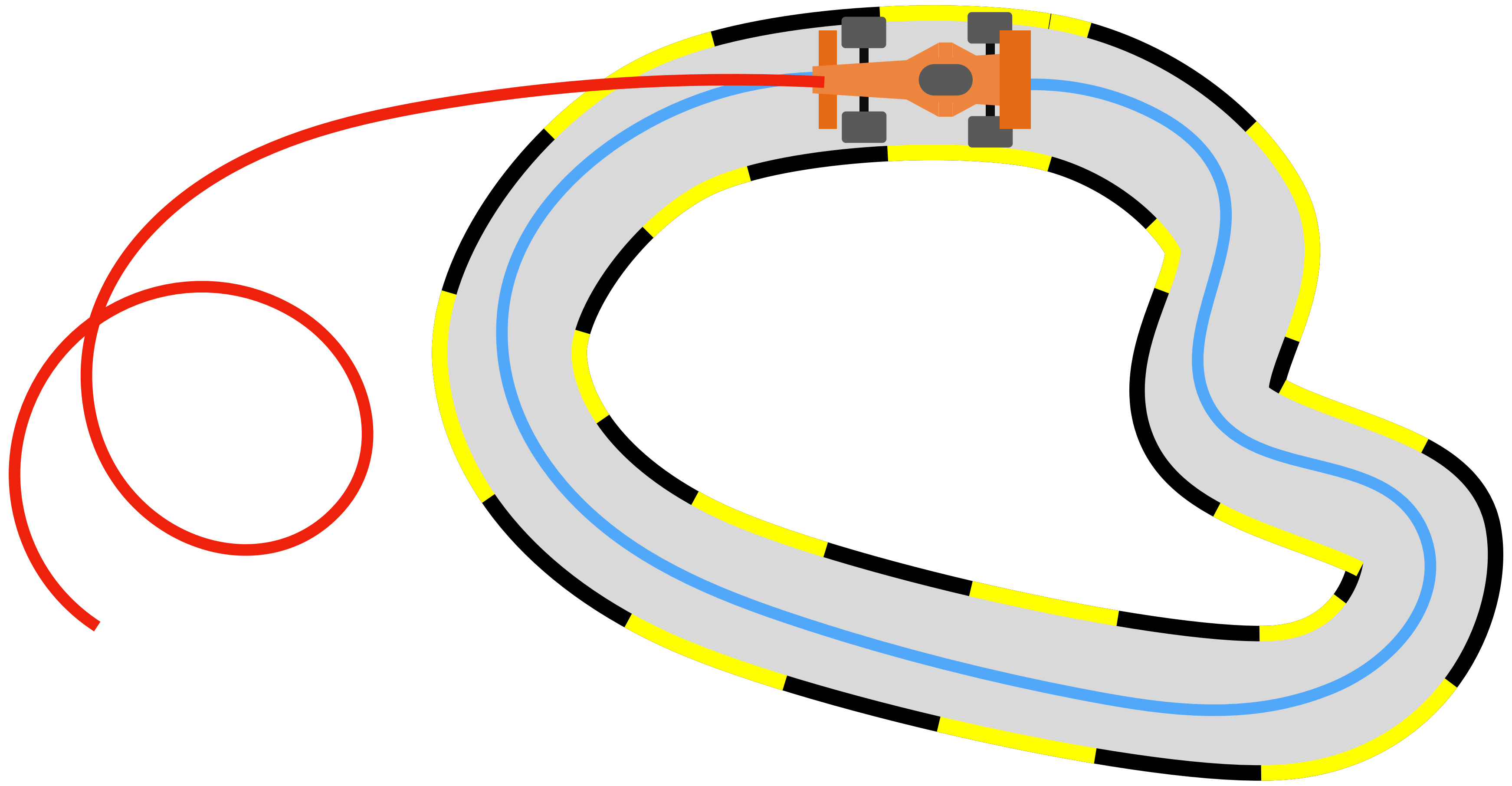
$$V^*(s_t) = \min_{a_t} \left[C(s_t, a_t) + \mathop{\mathbb{E}}_{s_{t+1} \sim P(\cdot \mid s_t, a_t)} V^*(s_{t+1}) \right]$$



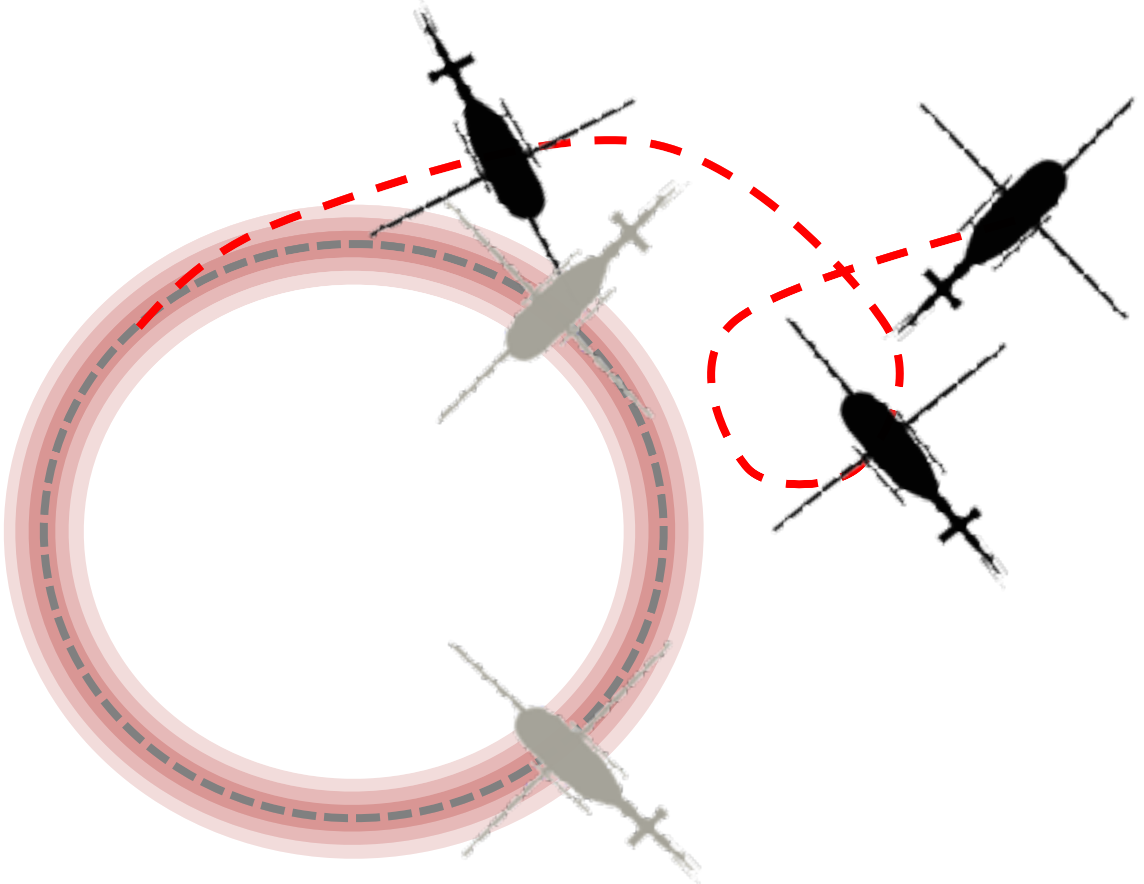
LEARN OPTIMAL VALUE BY INTERACTIONS W/ HUMAN + WORLD

Where all did we see covariate shift?

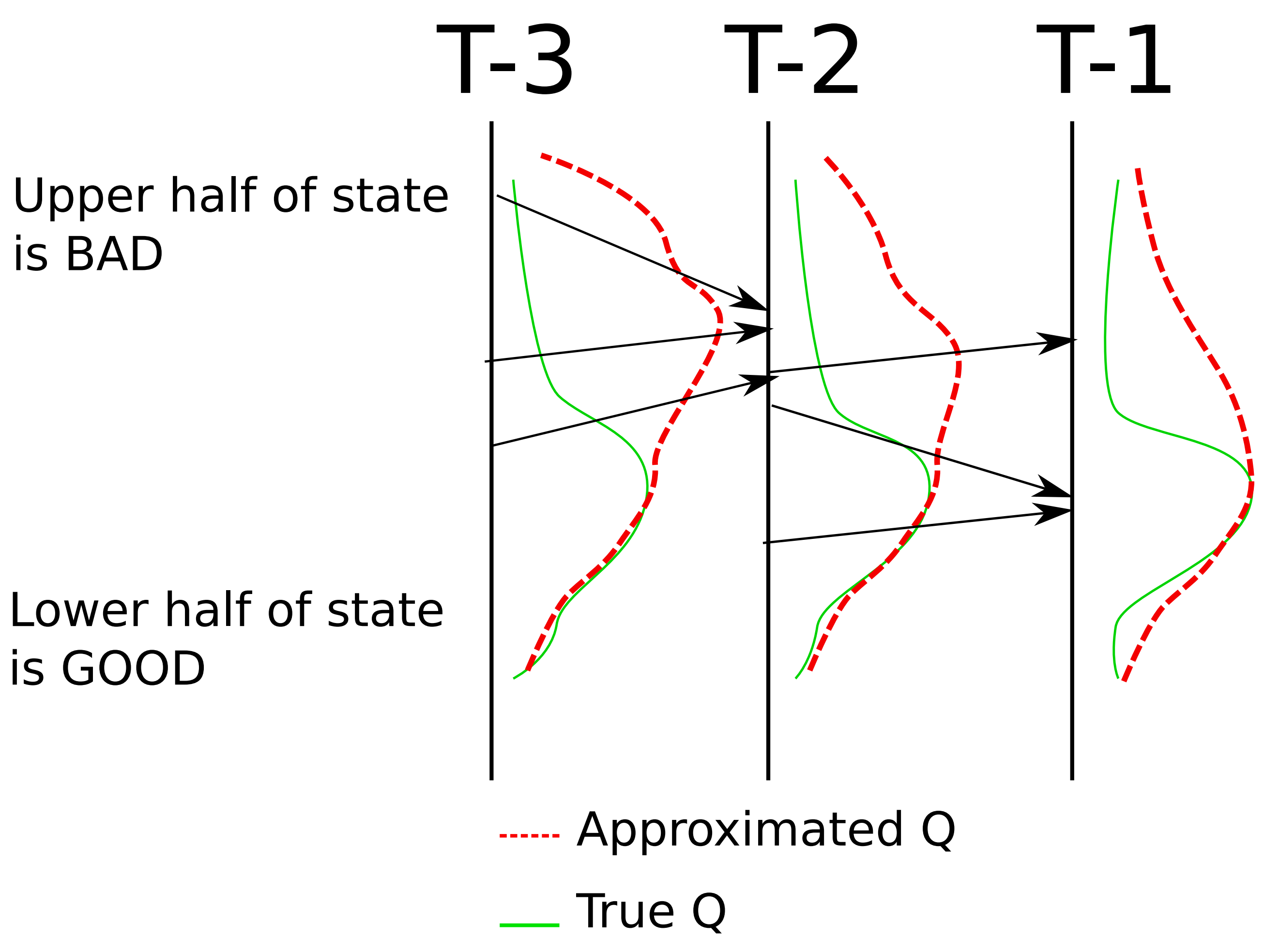
Imitation Learning?



Model Based RL?



Approximate Dynamic Programming?



Interactive Learning

Learner | Adversary

Initialize policy

π_1 [policy]

Chooses loss

$l_1(.)$ [loss]

Update policy

π_2

Chooses loss

$l_2(.)$

⋮ ⋮

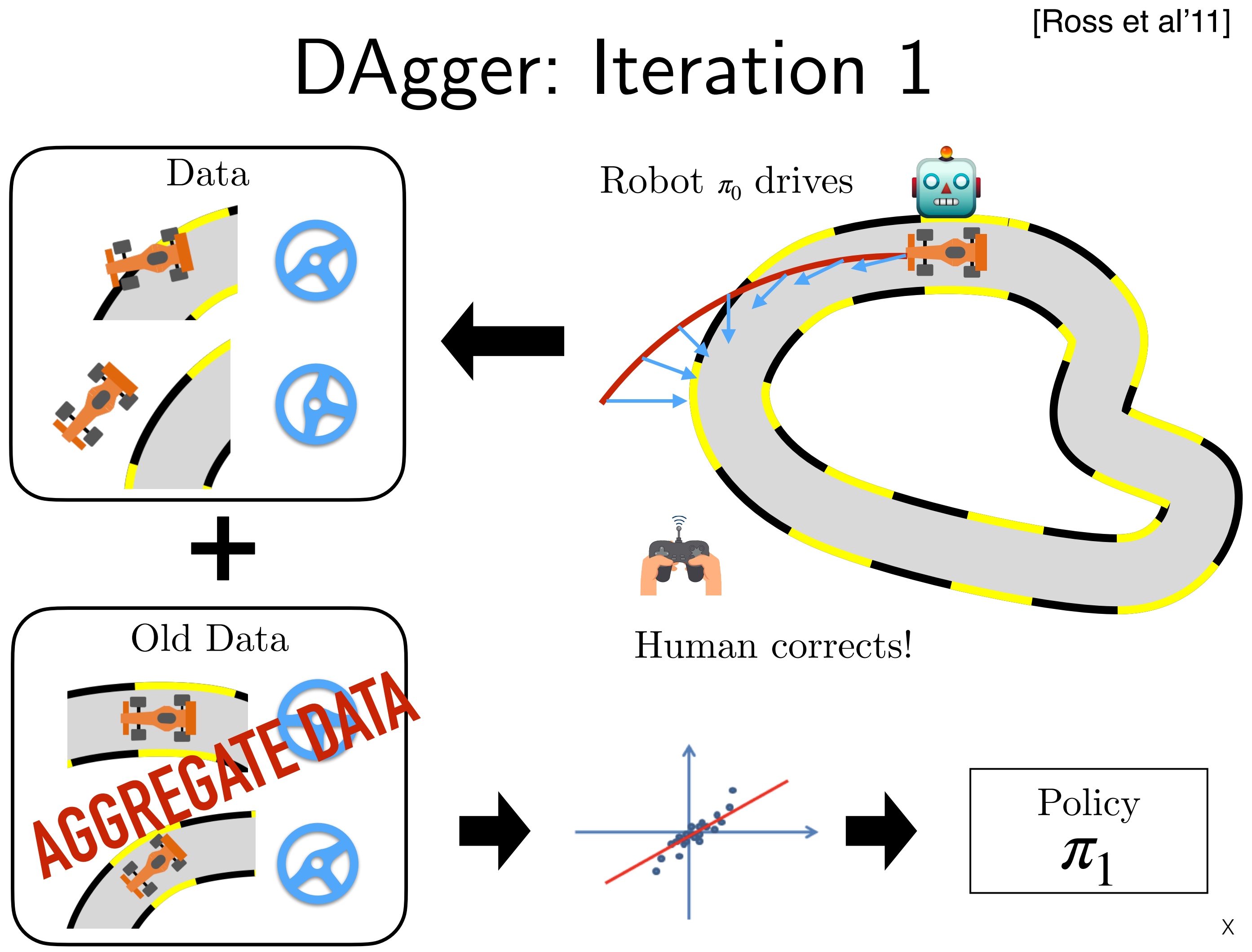
Learning is a Game!

Follow the leader is aggressive

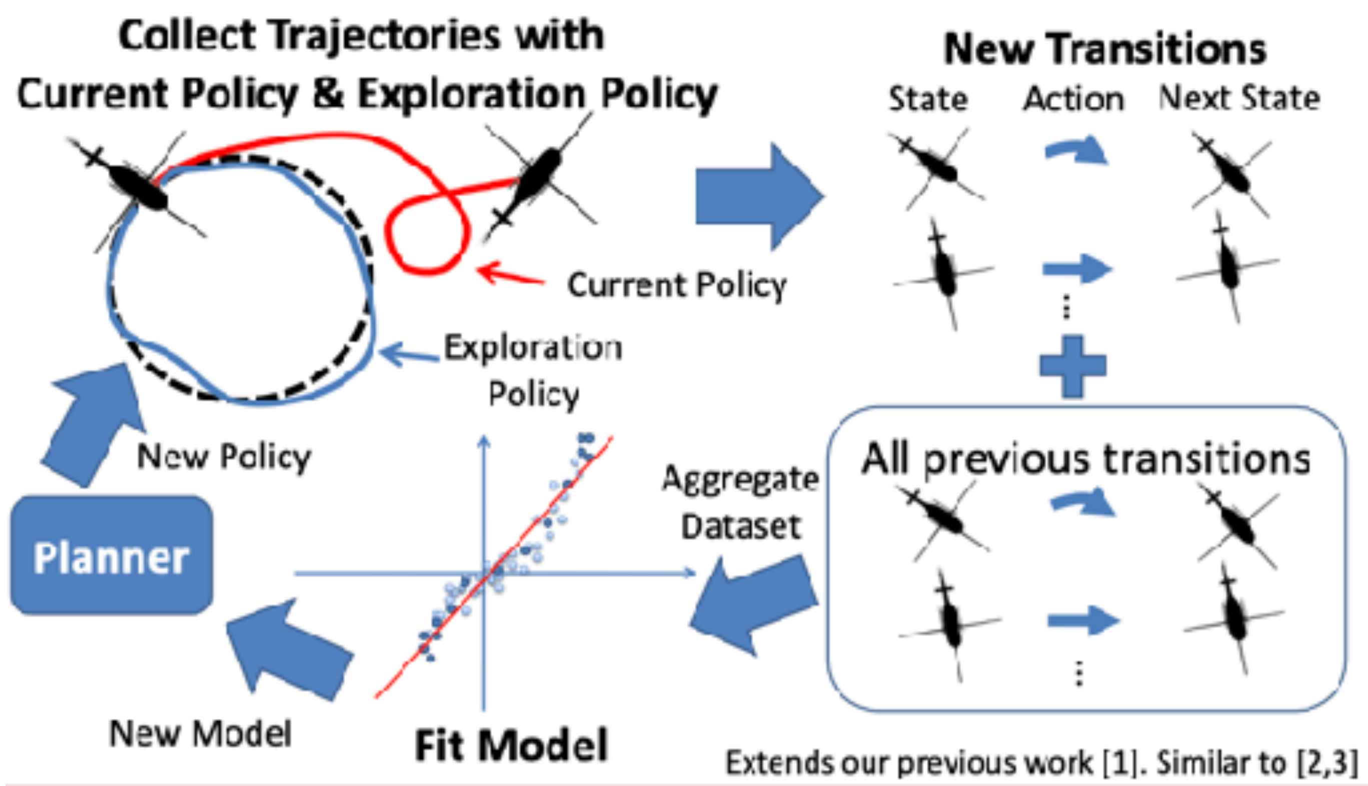
Slowly change predictions, achieve no-regret

No regret solves all!

DAgger



DAgger for SysID



Conservative policy iteration

Idea 1: Conservative Policy Iteration (CPI)

$$\pi' = (1 - \alpha)\pi + \alpha\pi_{greedy}$$

Mix in old policy and greedy policy

Can prove that performance difference is bounded by

$$V^{\pi'}(s) - V^{\pi}(s) \geq \alpha A_{greedy} - 2\alpha^2 \frac{\gamma}{1 - \gamma}$$

How much greedy policy improves based on estimate

How much distribution shift hurts!

GOAL: MINIMIZE

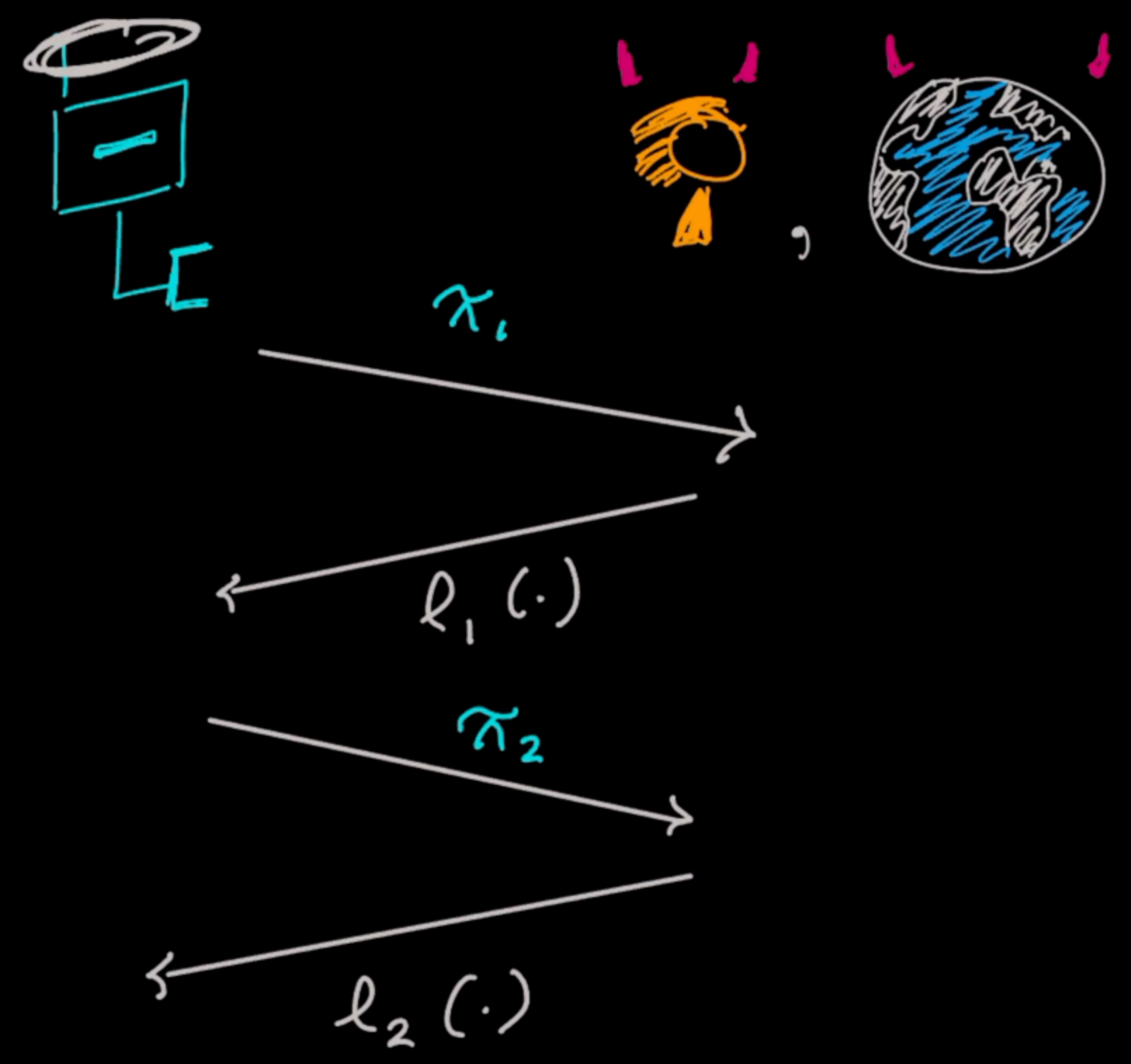
AVG. REGRET

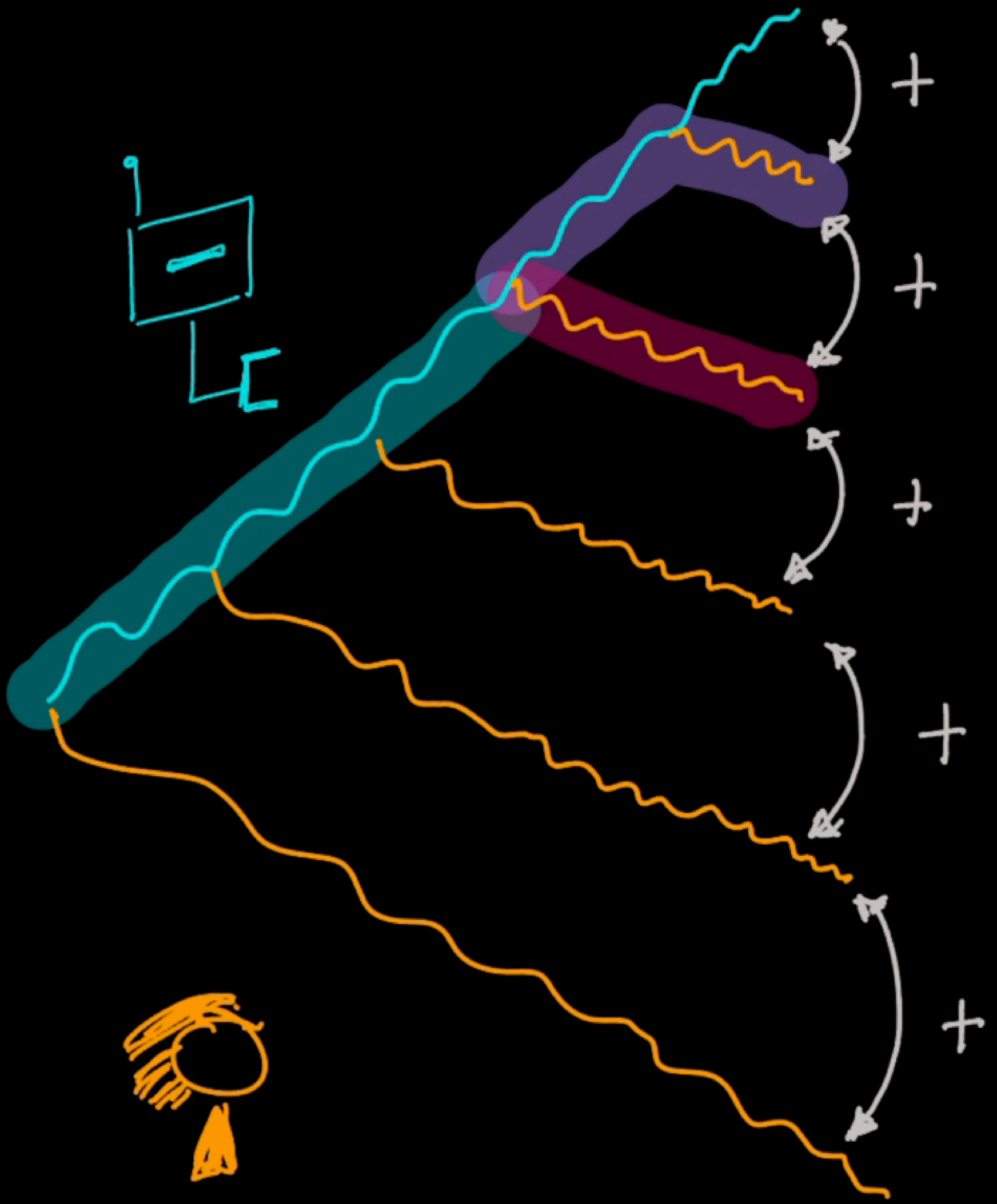
$$\frac{1}{N}\left[\sum_{i=1}^{N} \underbrace{l_i(\pi_i)}_{\text{LEARNER LOSS}} - \underbrace{\min_{\pi^*} \sum_{i=1}^{N} l_i(\pi^*)}_{\text{BEST LOSS}}\right]$$

AS $N \to \infty$, AVG REGRET $\to 0$

"DO AS WELL AS YOU COULD HAVE IF YOU HAD ALL THE DATA UPFRONT"

INTERACTIVE ONLINE LEARNING





Performance Difference Lemma

$$J(\pi) - J(\pi^*)$$

$$= \sum_{t=1}^{T} \underbrace{\mathbb{E}_{s_t \sim d_\pi}}_{\text{States the robot visits}} \underbrace{\Big[\underbrace{Q^*(s_t, \pi(s_t))}_{\text{Action value of robot}} - \underbrace{Q^*(s_t, \pi^*(s_t))}_{\text{Action value of human}} \Big]}_{\text{"Disadvantage"}}$$

Loss

$\ell(s, \pi(s))$

Reinforcement Learning: Brass Tacks

We don't know the MDP, all we see are traces (s,a,s')

Model Based:
Learn a model. Plan with the model.

Model Free:
Forget about models. Learn the policy.

Model Free RL: Actor Critic

Actor

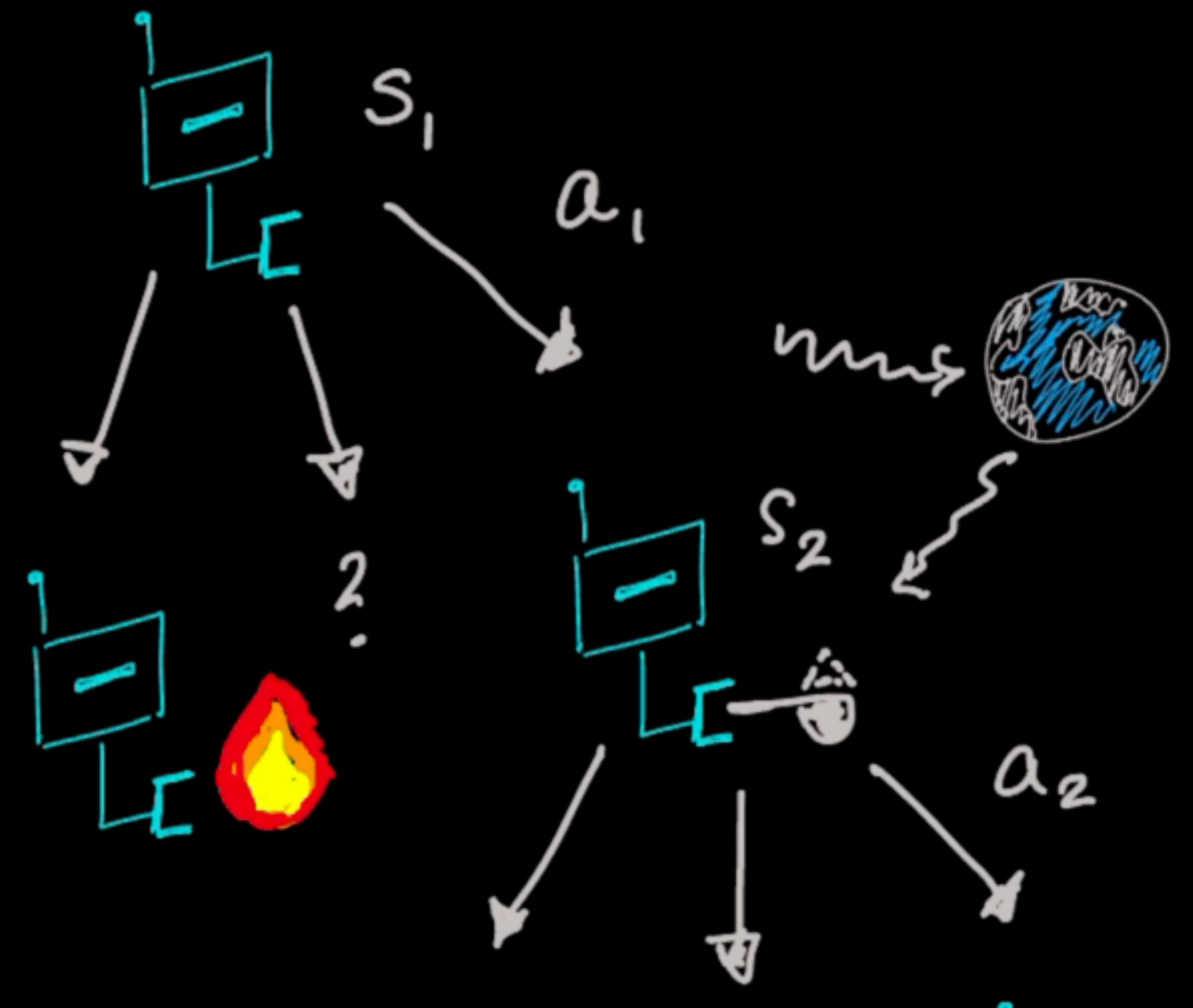
Critic

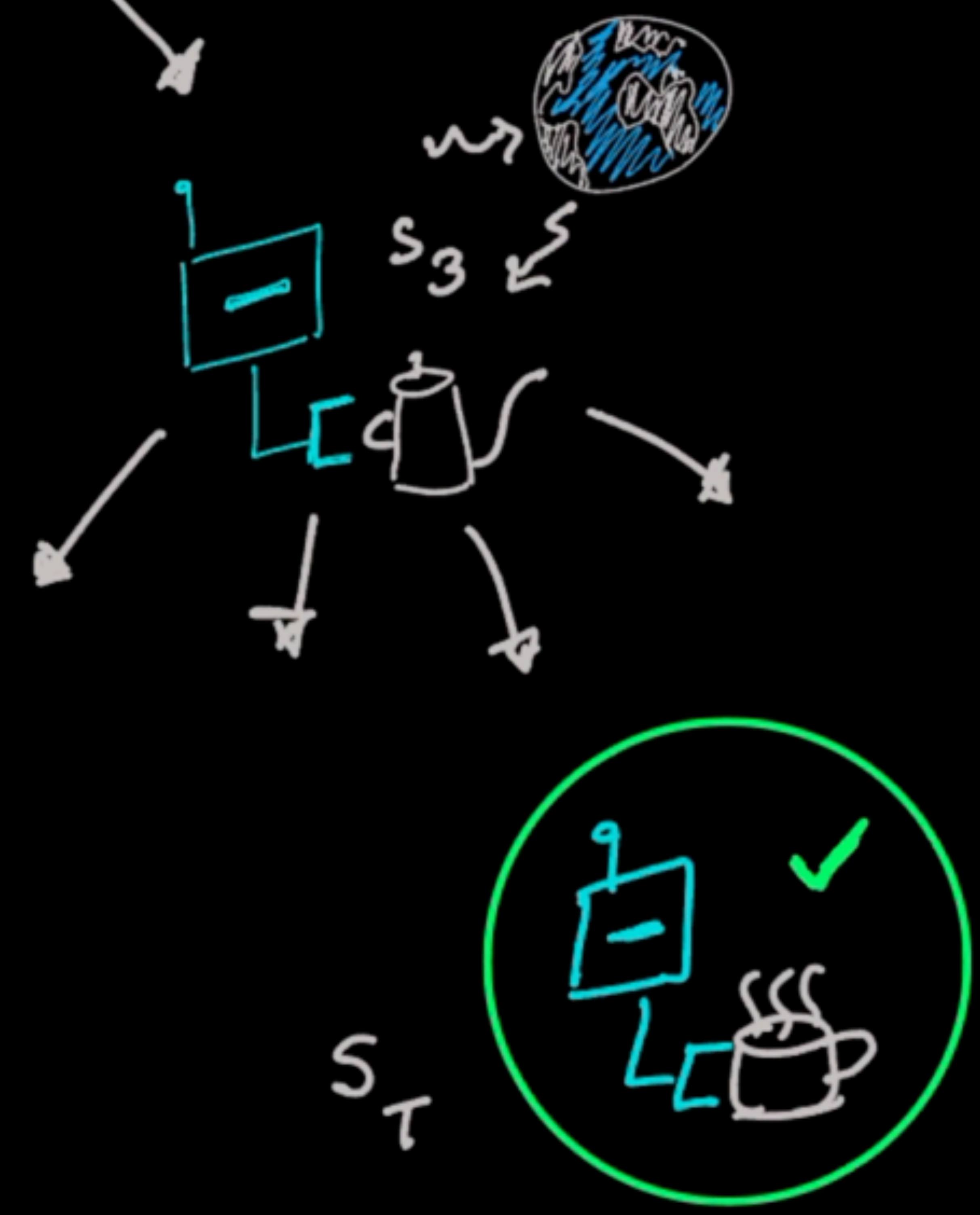
Policy improvement of π

Estimates value functions $Q^{\pi}_{\phi}/V^{\pi}_{\phi}/A^{\pi}_{\phi}$

(Natural) Policy Gradient

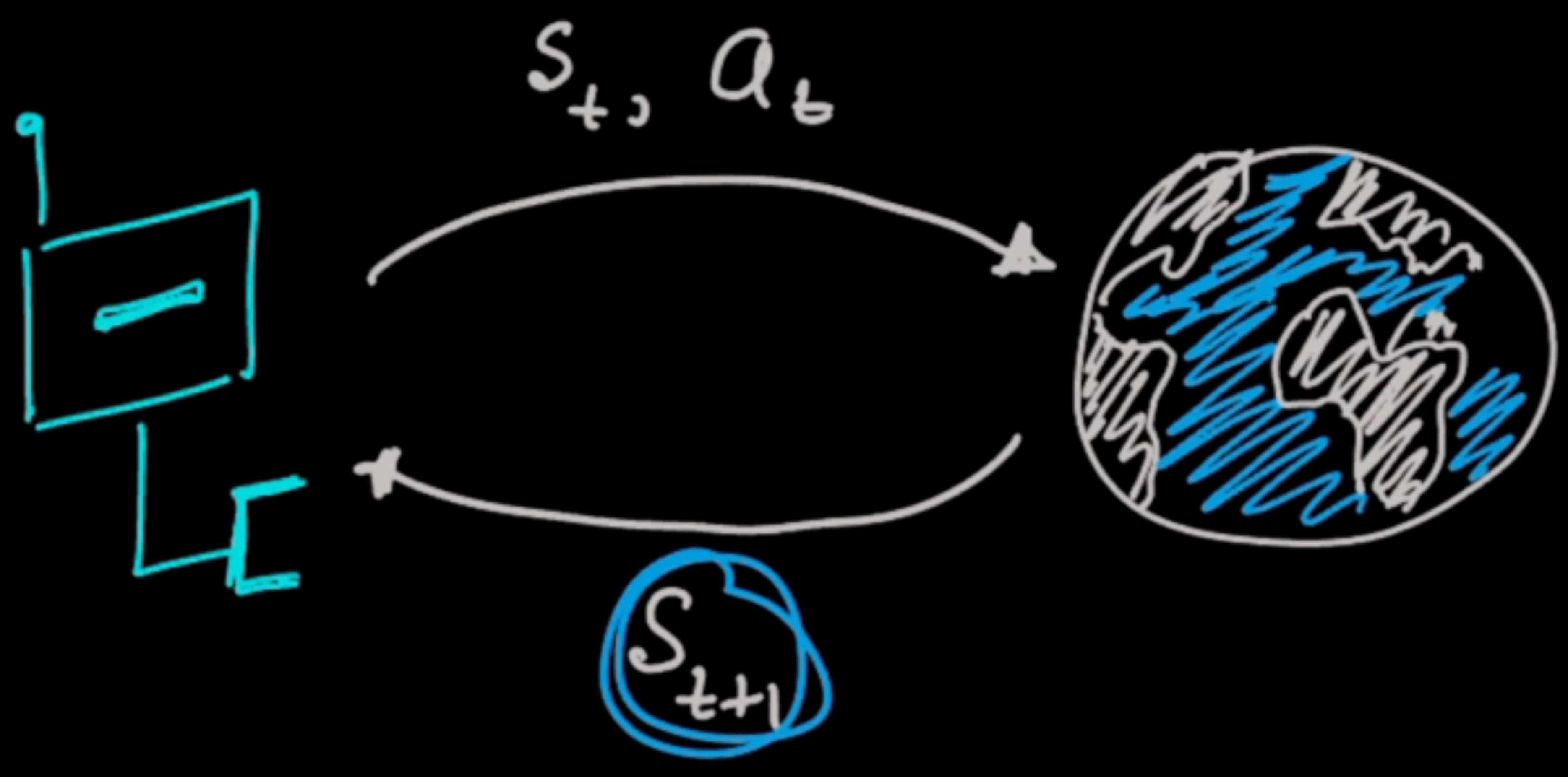
TD, MC





Explore all actions to see next state to estimate Q^*.

Reinforcement Learning



Estimate $Q^*(s_t, a_t)$ such that

$$Q^*(s_t, a_t) = c(s_t, a_t) + \underset{s_{t+1}}{E} \min_{a_{t+1}} Q^*(s_{t+1}, a_{t+1})$$

i.e. Bellman consistent

Imitation Learning: Brass Tacks

We don't know the MDP, all we see are human actions (a*)

Learn Cost:

Learn a cost that makes human look cheap, learner look expensive

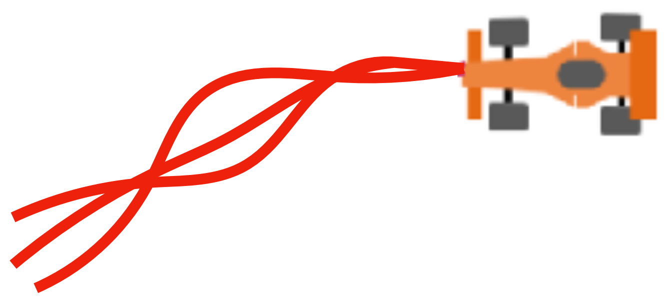
Learn Values:

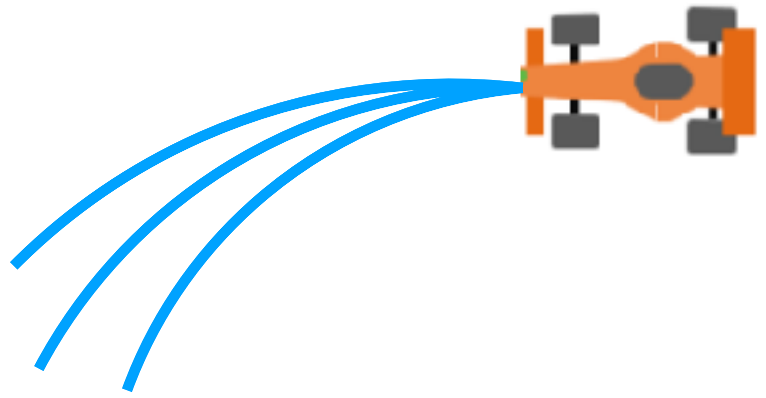
Learn Q* that makes human look cheap, learner look expensive

Inverse Optimal Control (Learn Cost)

Make human look cheap, learner look expensive

$$\max_{\phi} \min_{\theta} \; \mathbb{E}_{s_t, a_t \sim \pi_\theta}[C_\phi(s_t, a_t)] - \mathbb{E}_{s_t^*, a_t^* \sim \pi^*}[C_\phi(\xi)]$$



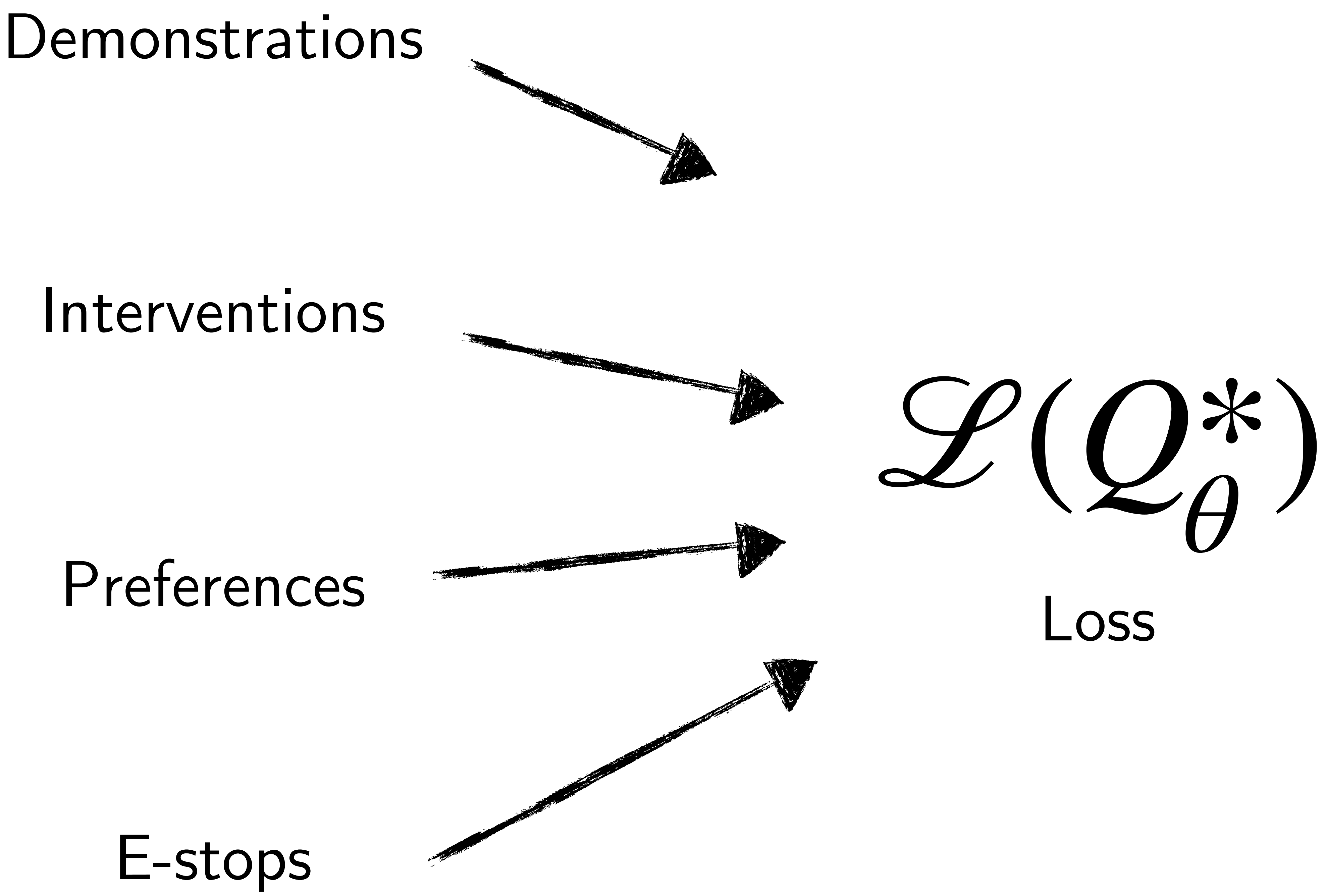


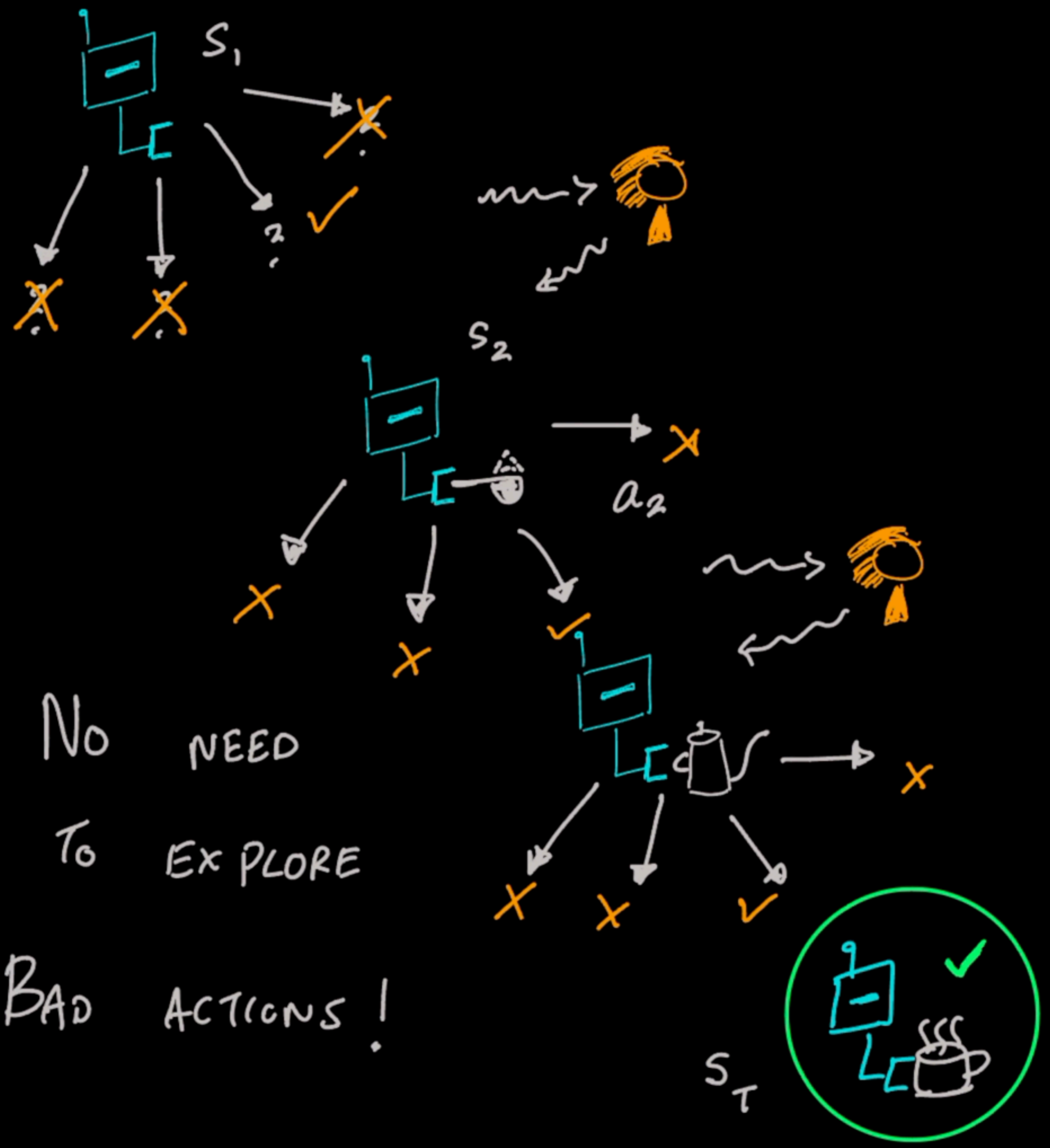
Cost of Learner

Cost of Expert

Learn Values

Estimate Q* from demonstrations, interventions, preferences, ..
and even E-stops!

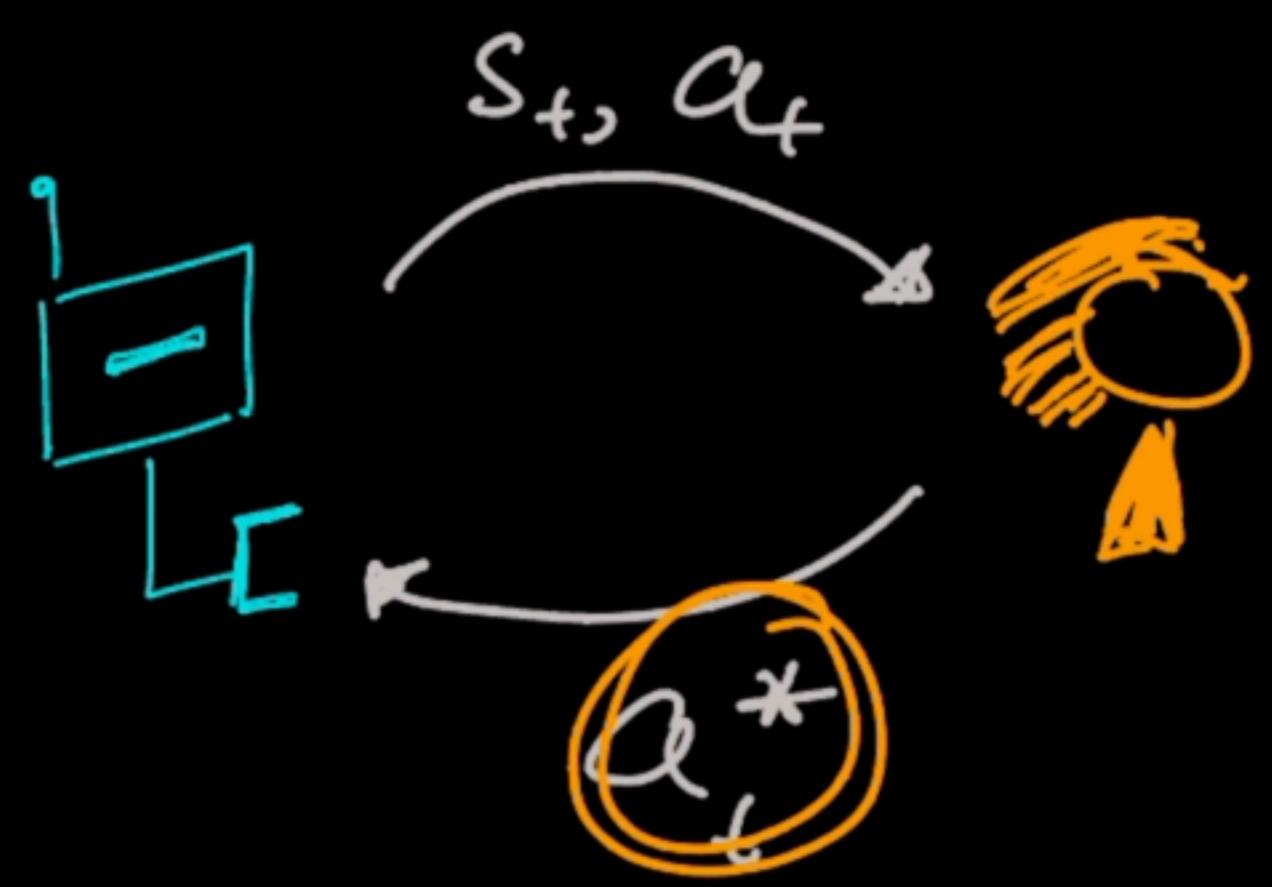




No need to explore

Bad actions!

Imitation Learning



Estimate $Q^*(s_t, a_t)$ such that

$$Q^*(s_t, a_t) = \min_{a_t'} Q^*(s_t, a_t')$$

i.e. Human is optimal

The Imitation Game

We have an interactive expert.

Apply PDL in forward direction: roll-in learner, roll-out expert

$$\min_{\pi} \quad \max_{Q^*} \sum_{t=1}^{T} \mathbb{E}_{s_t \sim \pi}[Q^*(s_t, \pi(s_t)) - Q^*(s_t, \pi^*(s_t))]$$

😇 😈

Use no-regret learning to solve the game!

$$O(\epsilon T)$$

The RL Game

We don't have interactive expert.

Apply PDL in reverse direction: roll-in expert, roll-out learner

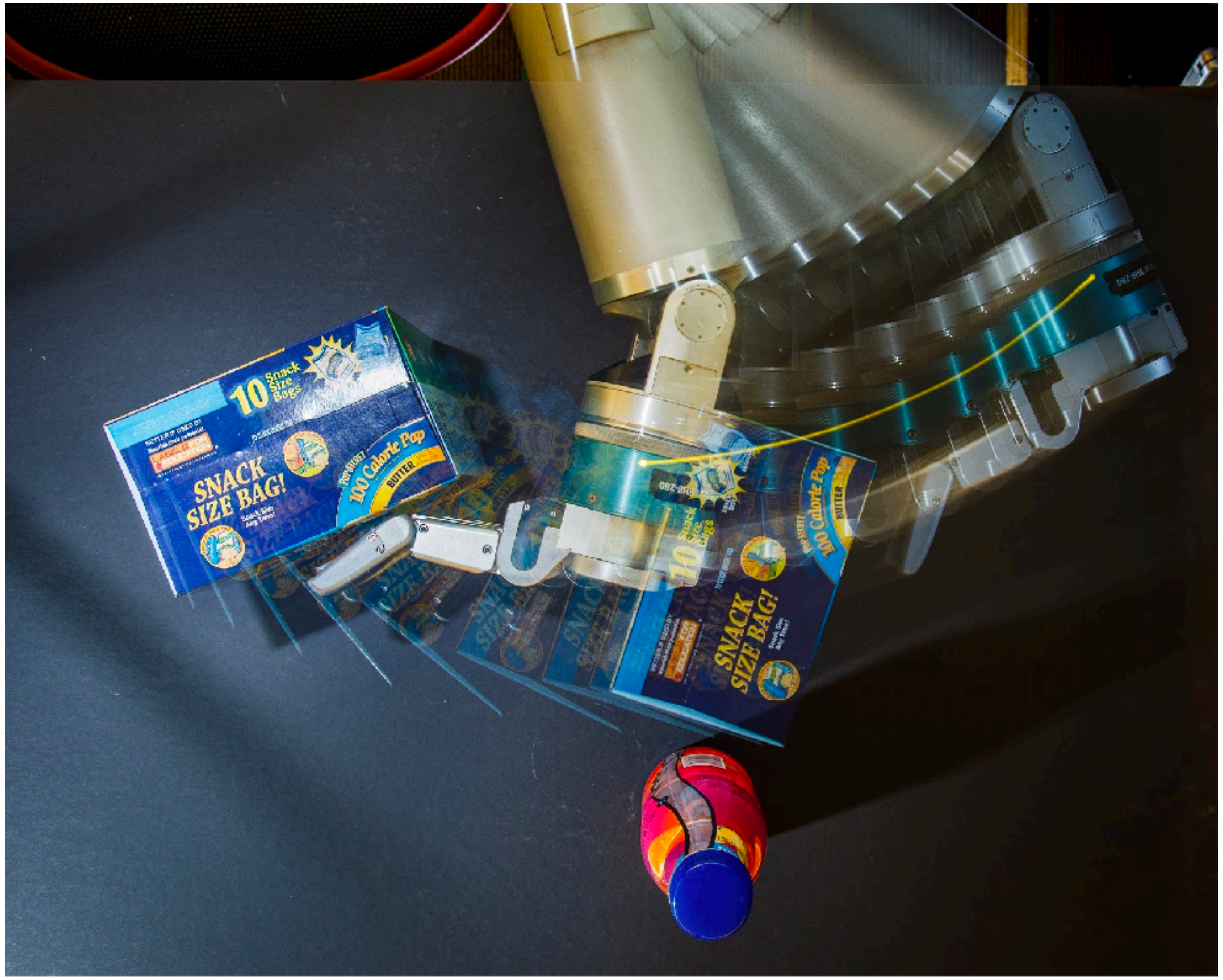
$$\min_{\pi} \quad \max_{Q^{\pi}} \sum_{t=1}^{T} \mathbb{E}_{s \sim \pi^*} Q^{\pi}(s, \pi(s)) - Q^{\pi}(s, \pi^*(s))$$

😇 😈

Use no-regret learning to solve the game! $O(\epsilon T^2)$

A grand unification of IL / RL Games?

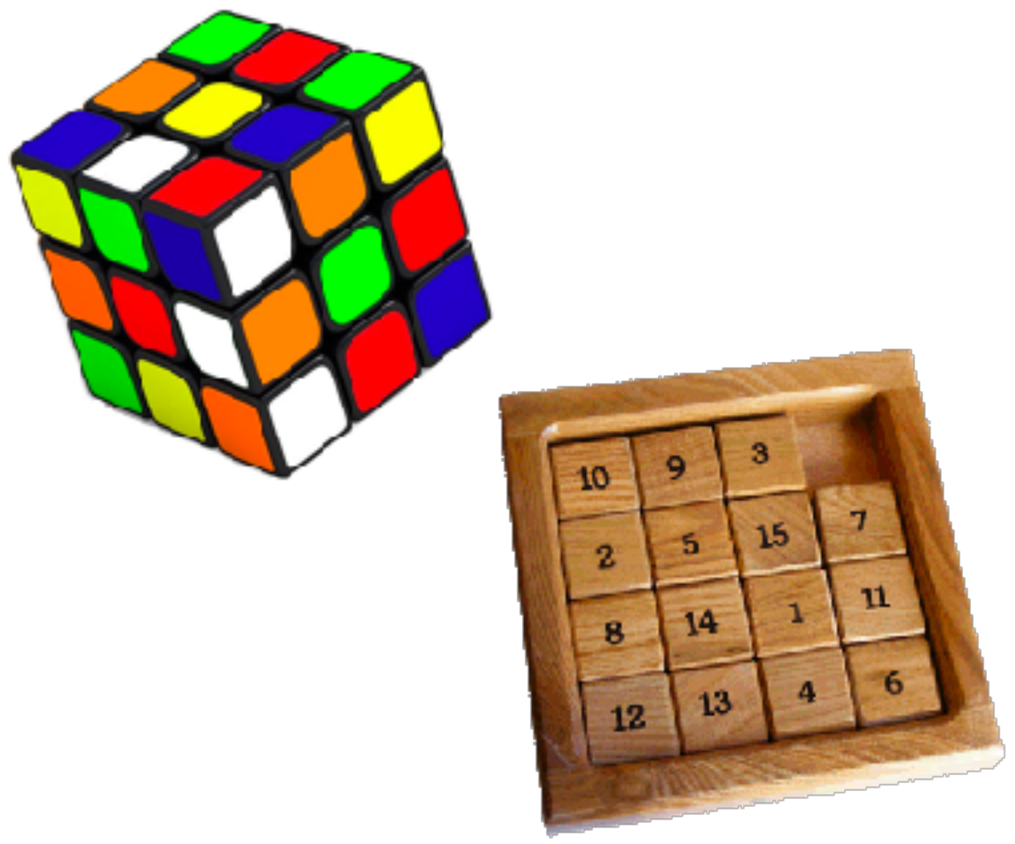
A simple question:
Can learning help us build better planners?



A prospective grad student:
“Is planning just A*?”

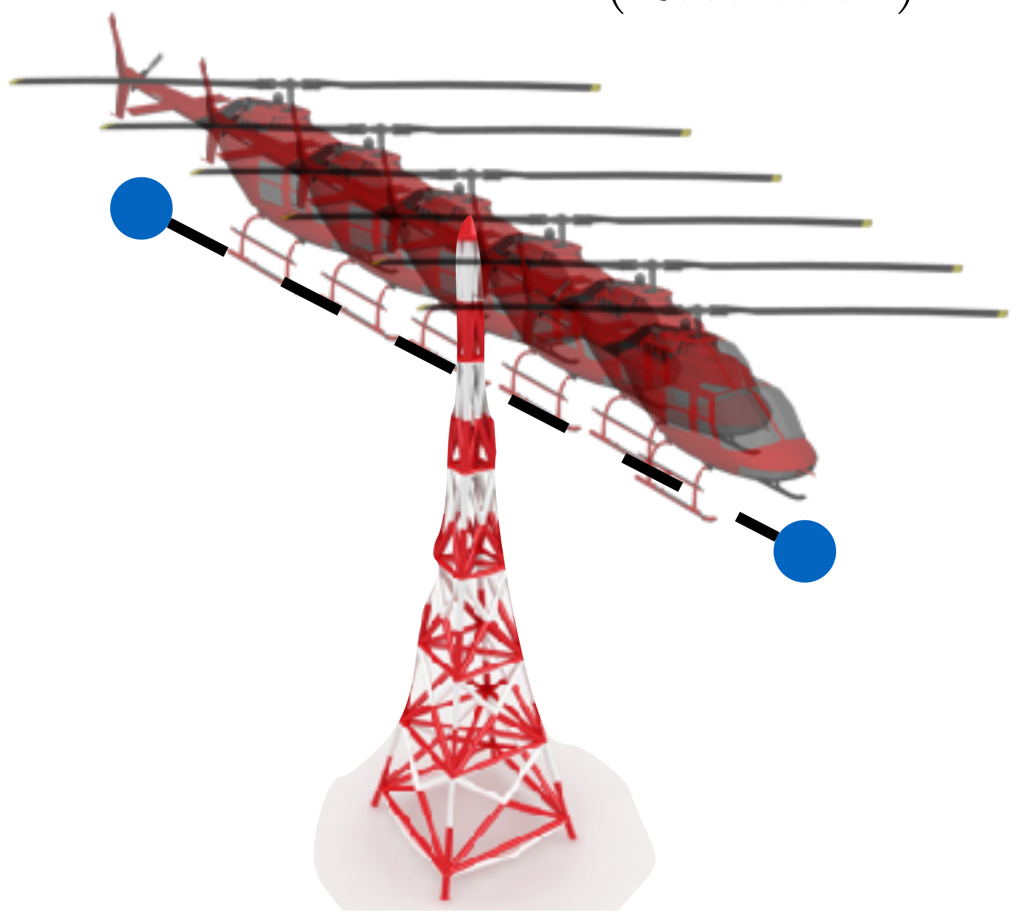
Motion Planning: Dealing with expensive collision checking

Trivial

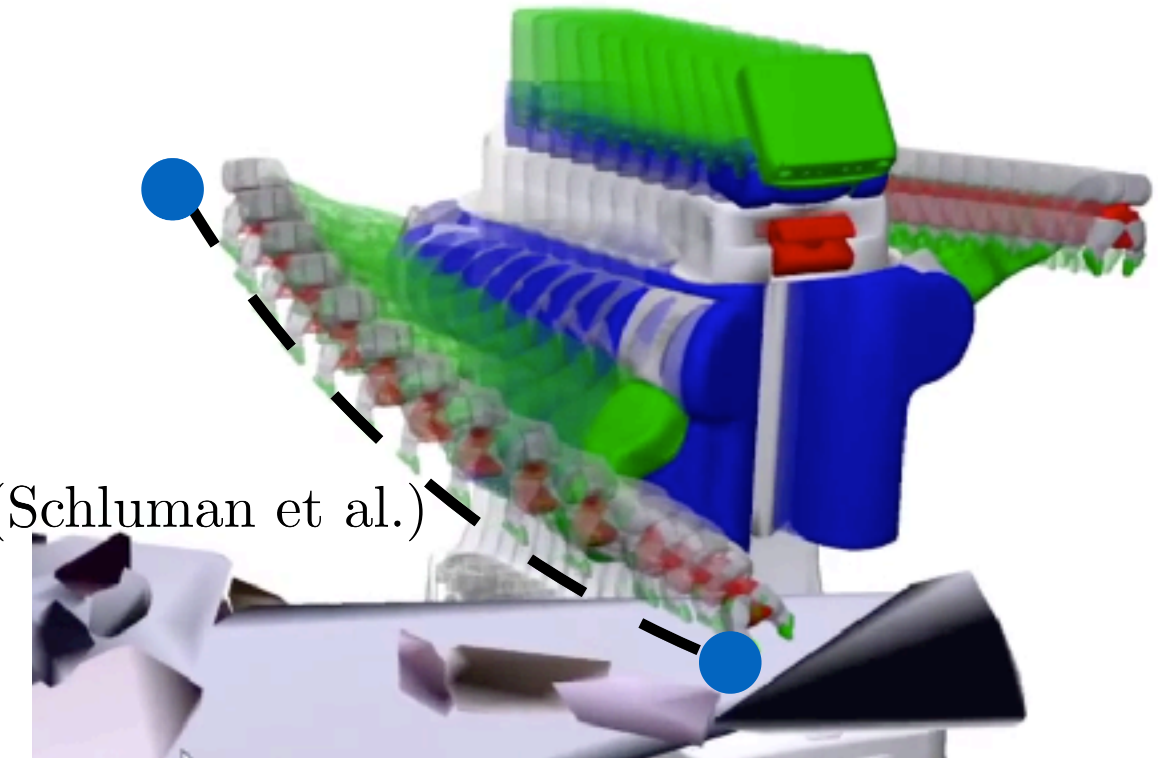


Medium

(Ross et al.)



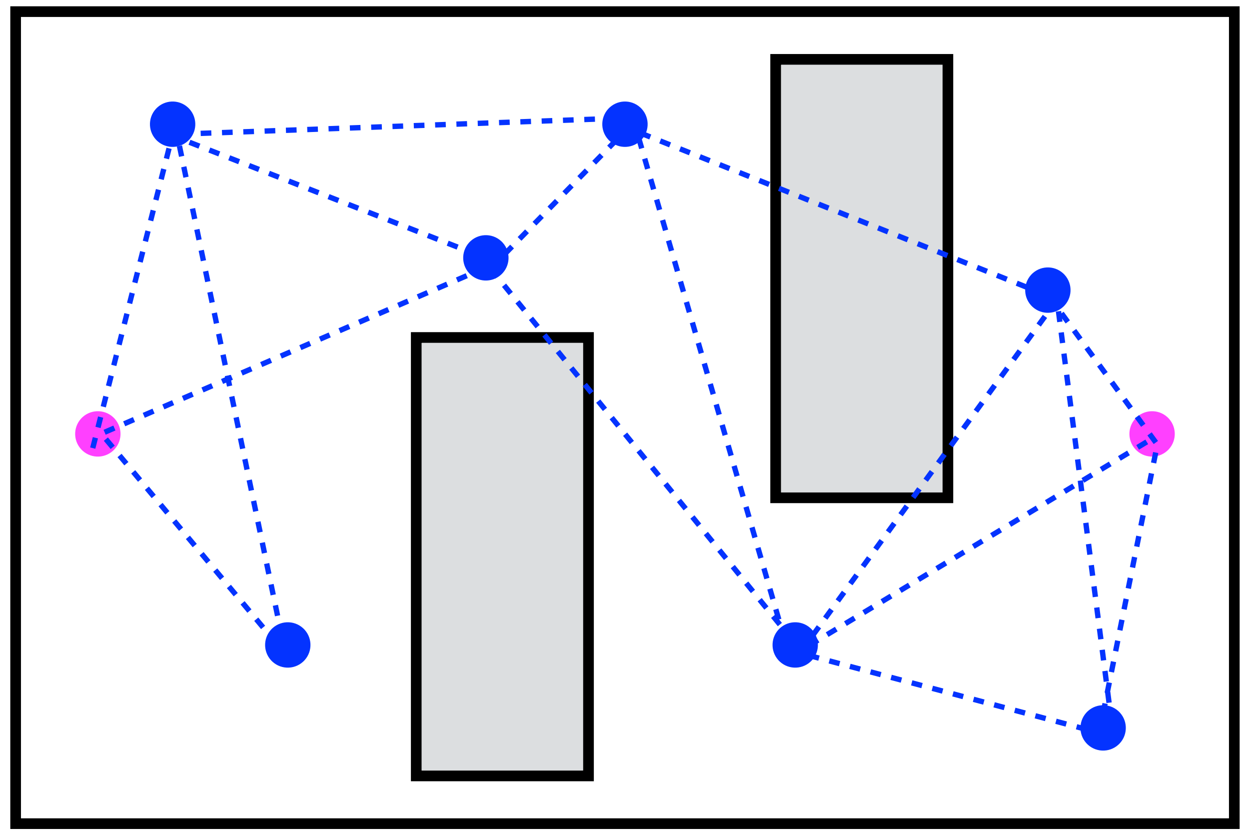
Expensive



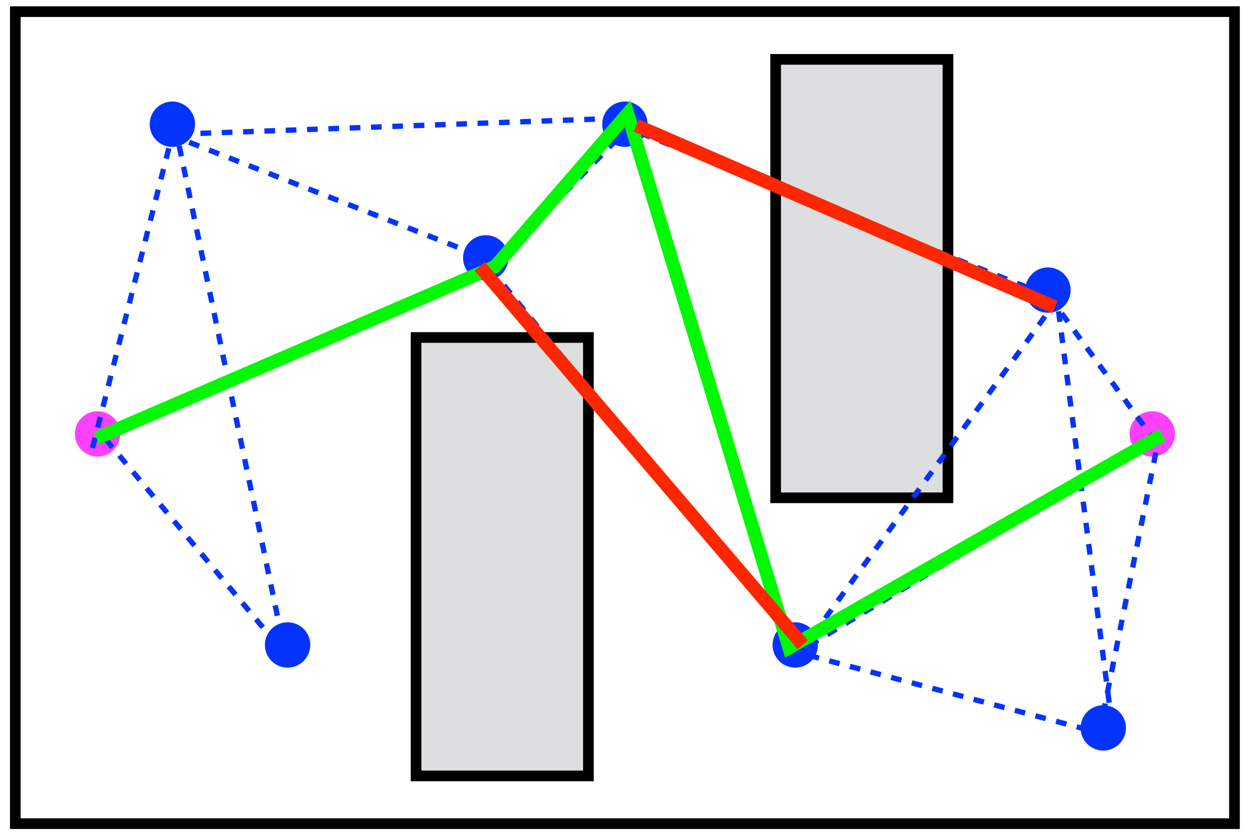
(Schluman et al.)

(LaValle'06, Bialkowski'11, Hauser'15, ...)

General framework for motion planning



Create a graph



Search the graph

Interleave

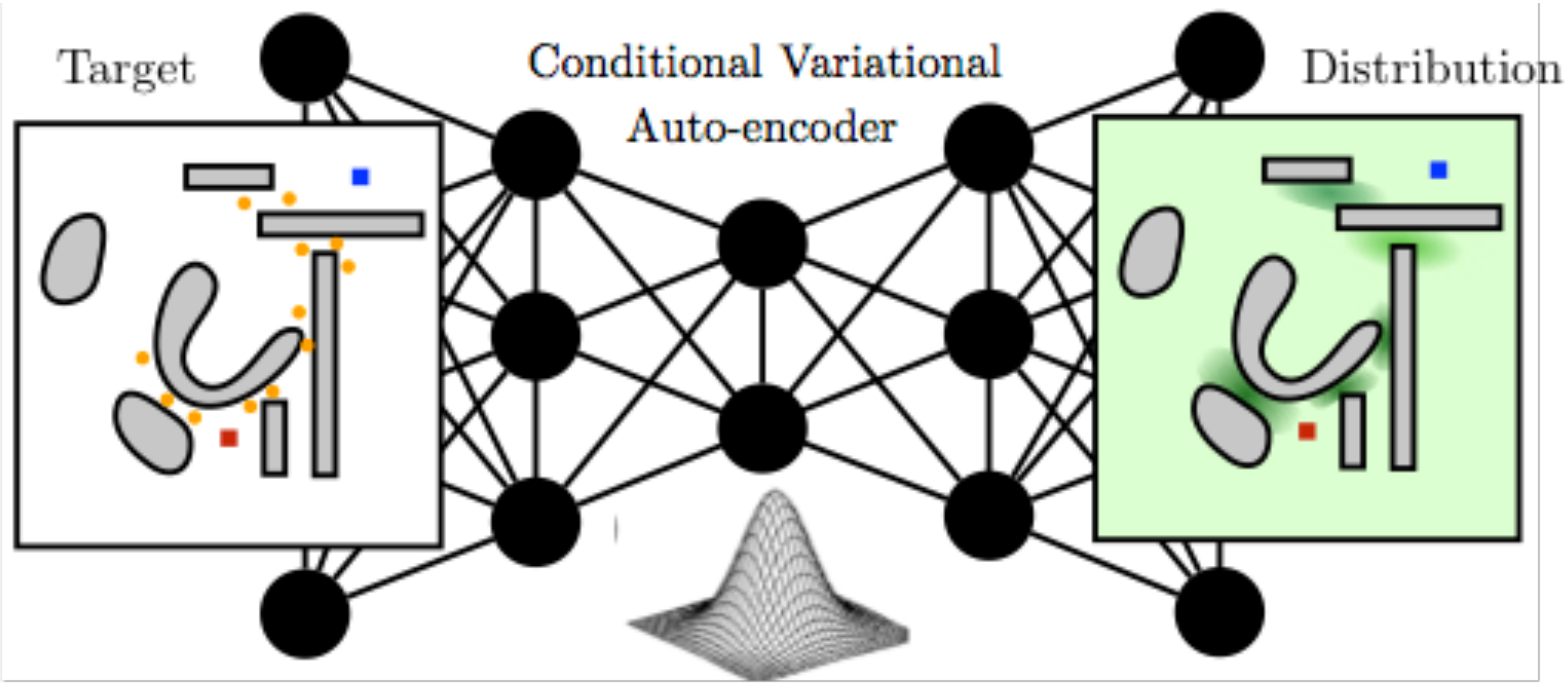
General framework for motion planning

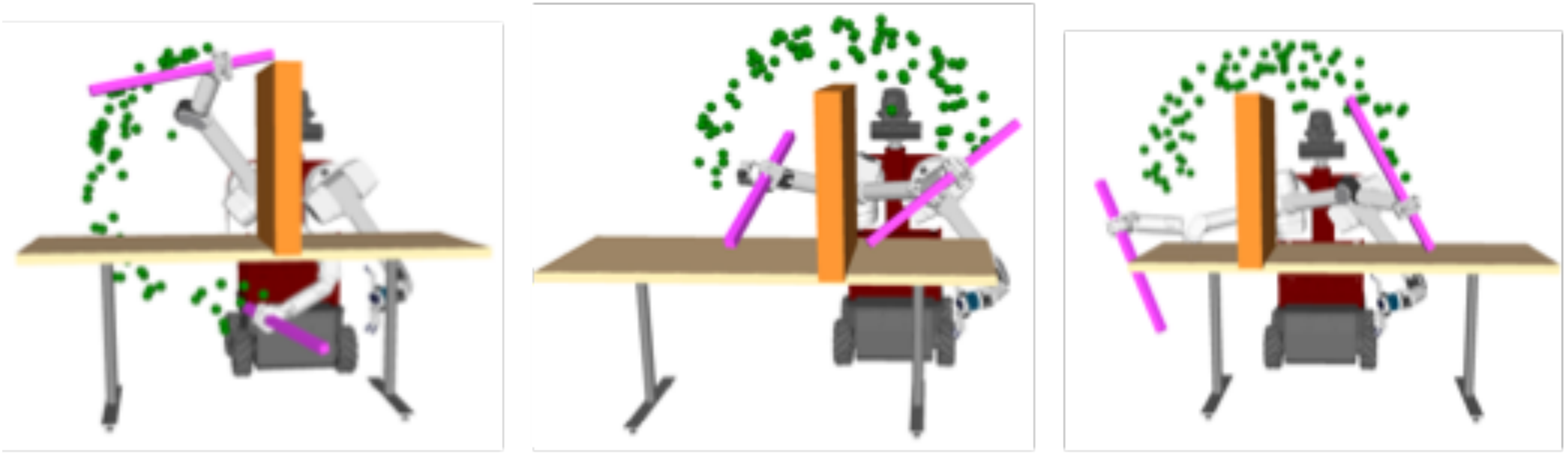
Any planning algorithm		Create graph		Search graph		Interleave
	=	e.g. fancy random sampler	×	e.g. fancy heuristic	×	e.g. fancy way of densifying
		Learn sampler!		Learn heuristic!		

LEGO: Leveraging Experience in Roadmap Generation for Sampling-Based Planning

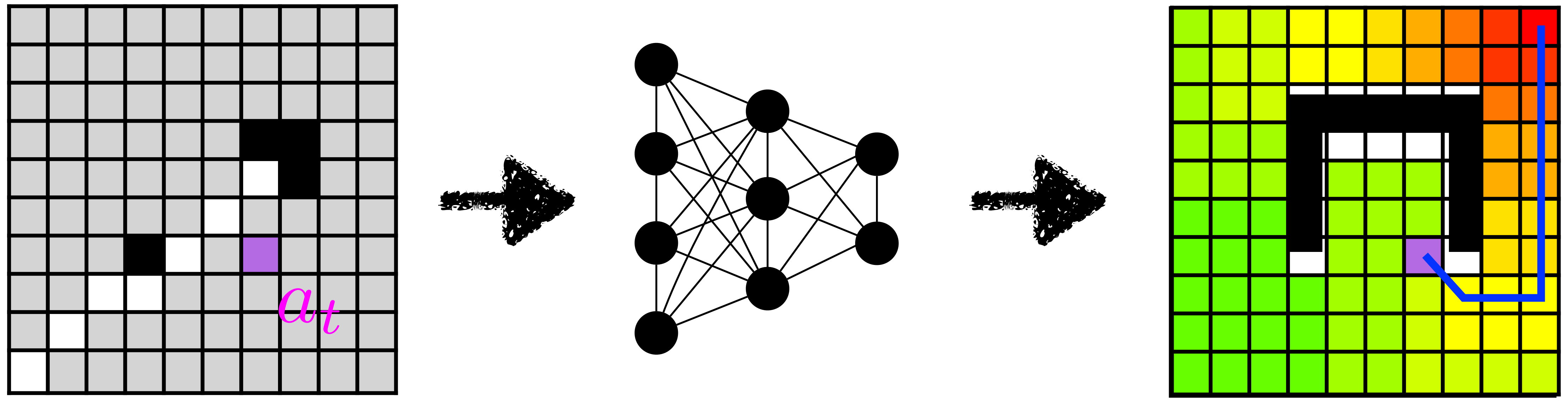
Rahul Kumar*[1], Aditya Mandalika*[2], Sanjiban Choudhury*[2] and Siddhartha S. Srinivasa*[2]

Learning a Sampler

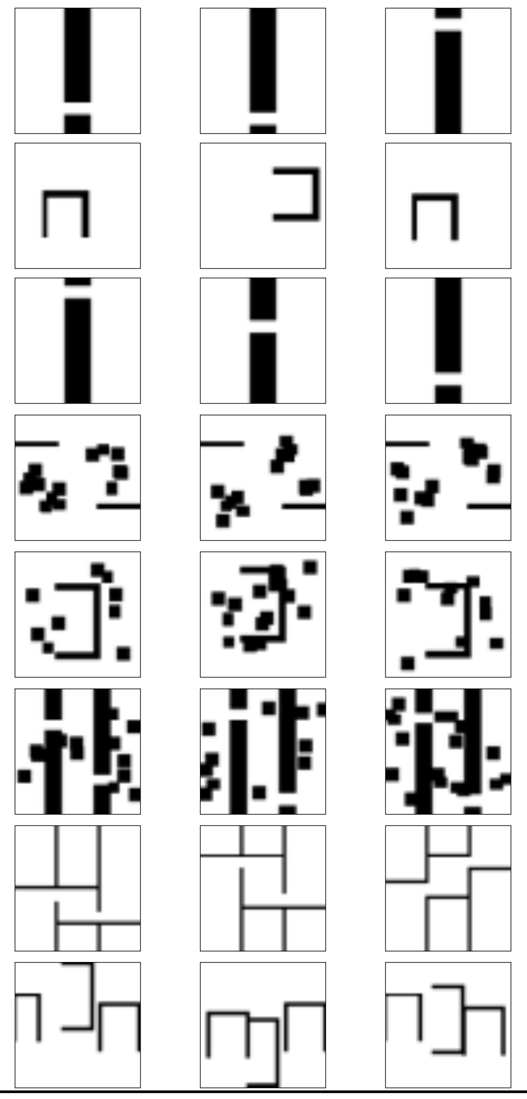




Learn a Heuristic



search tree → train heuristic → imitate true cost-to-go



	Ours	Baseline Learning			Baseline Handcrafted			
	SAIL	**SL**	**CEM**	**QL**	$h_{\mathbf{EUC}}$	$h_{\mathbf{MAN}}$	**A***	**MHA***
alternating_gap	0.039	0.432	0.042	1.000	1.000	1.000	1.000	1.000
single_gap	**0.158**	0.214	0.057	1.000	0.184	0.192	1.000	0.286
shifting_gap	0.104	0.464	1.000	1.000	0.506	0.589	1.000	0.804
forest	0.036	0.043	0.048	0.121	0.041	0.043	1.000	0.075
bugtrap_forest	0.147	0.384	0.182	1.000	0.410	0.337	1.000	0.467
gaps_forest	0.221	1.000	1.000	1.000	1.000	1.000	1.000	1.000
maze	0.103	0.238	0.479	0.399	0.185	0.171	1.000	0.279
multiple_bugtrap	0.479	0.480	1.000	0.835	0.648	0.617	1.000	0.876


Open Challenges