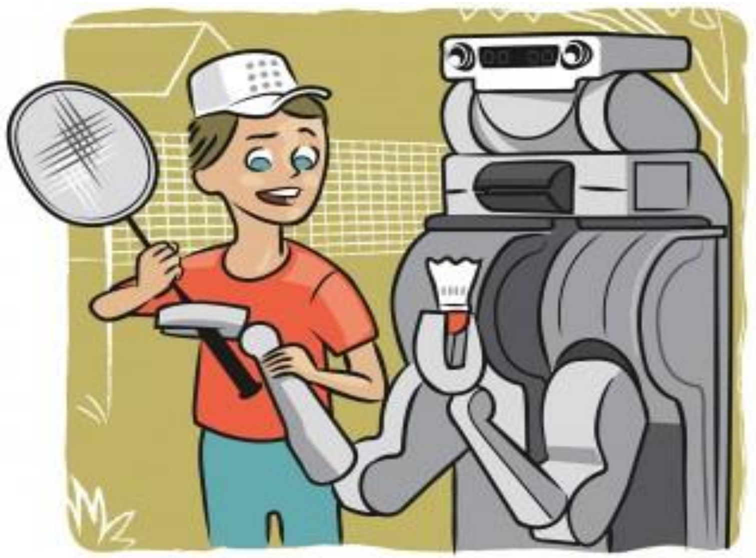
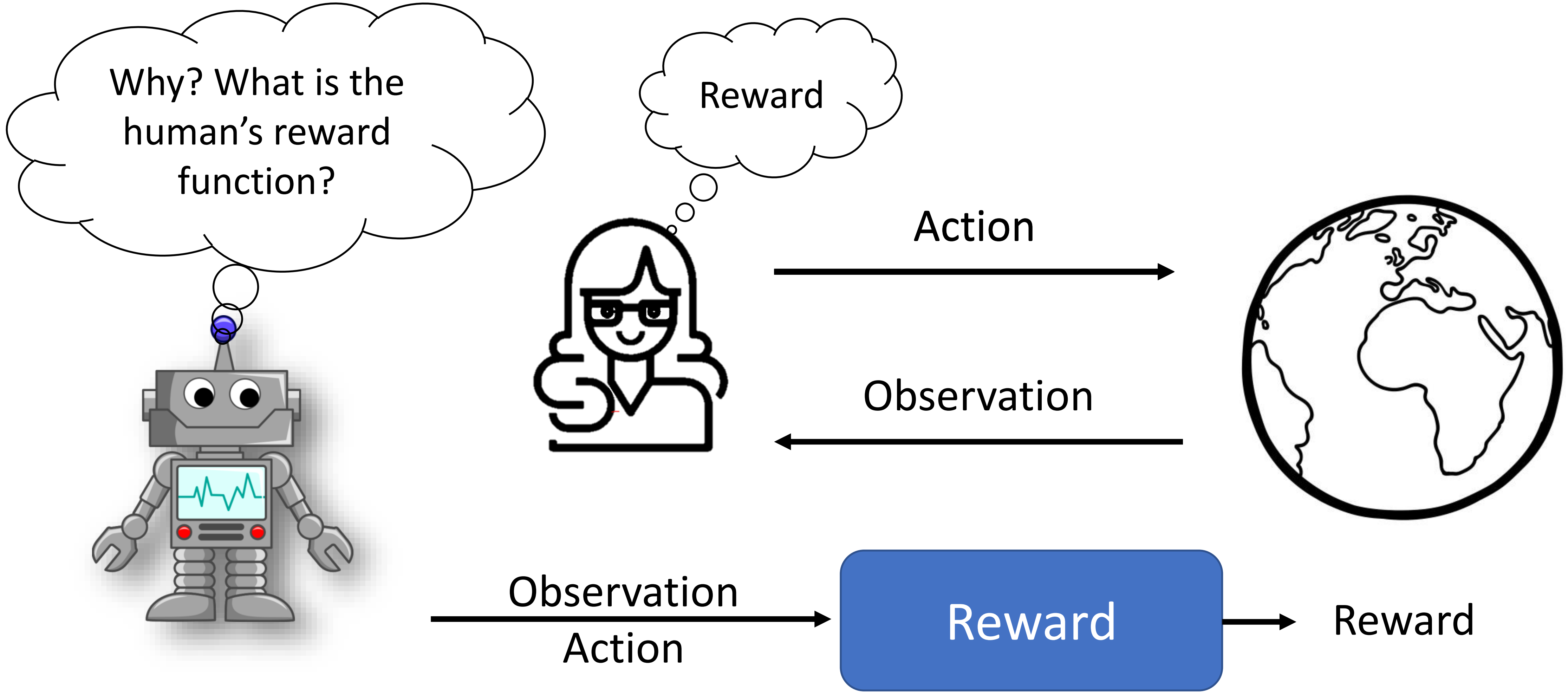
Inverse RL and Reward Learning



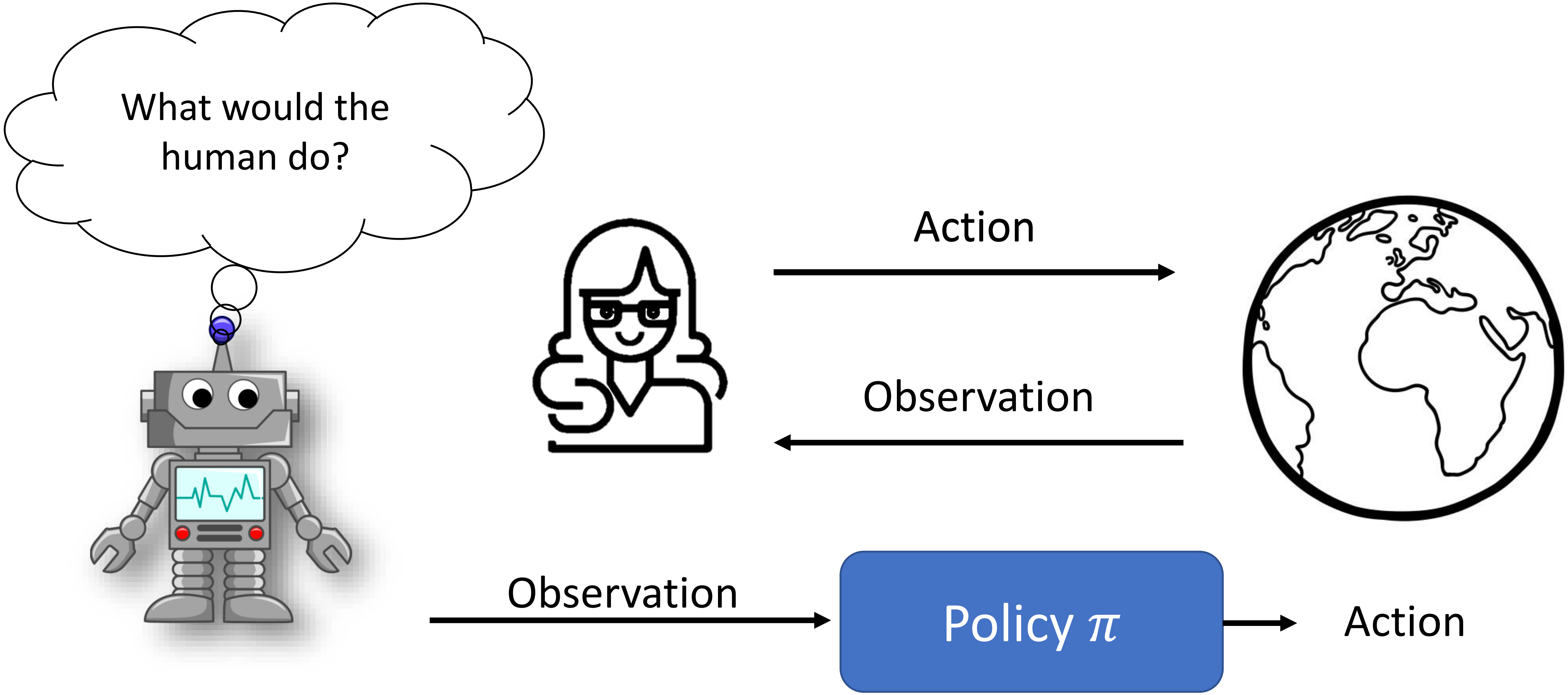
Instructor: Daniel Brown

[Some slides adapted from Sergey Levine (CS 285) and Alina Vereshchaka (CSE4/510)]

Reward Learning
(Inverse Reinforcement Learning)



Why not just imitate behavior? (Behavioral Cloning)



Credit: Simone Giertz

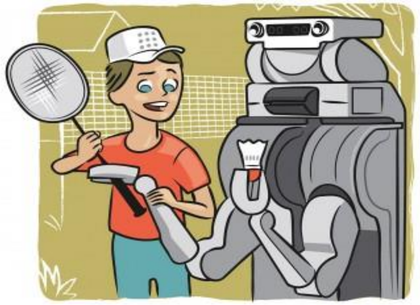
Human Intent Inference

Inverse Reinforcement Learning

- Given
  - MDP without a reward function
  - Demonstrations from an optimal policy π^*
- Recover the reward function $\boldsymbol{R}$ that makes π^* optimal

Imitation Learning

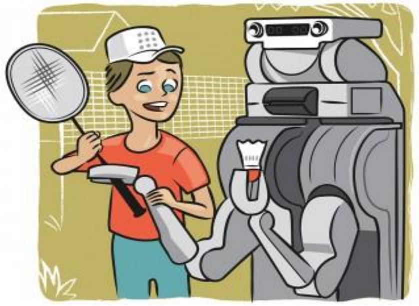
Behavioral Cloning



$\Rightarrow \pi$

- Answers the "How?" question
- Mimic the demonstrator
- Learn mapping from states to actions
- Computationally efficient
- Compounding errors

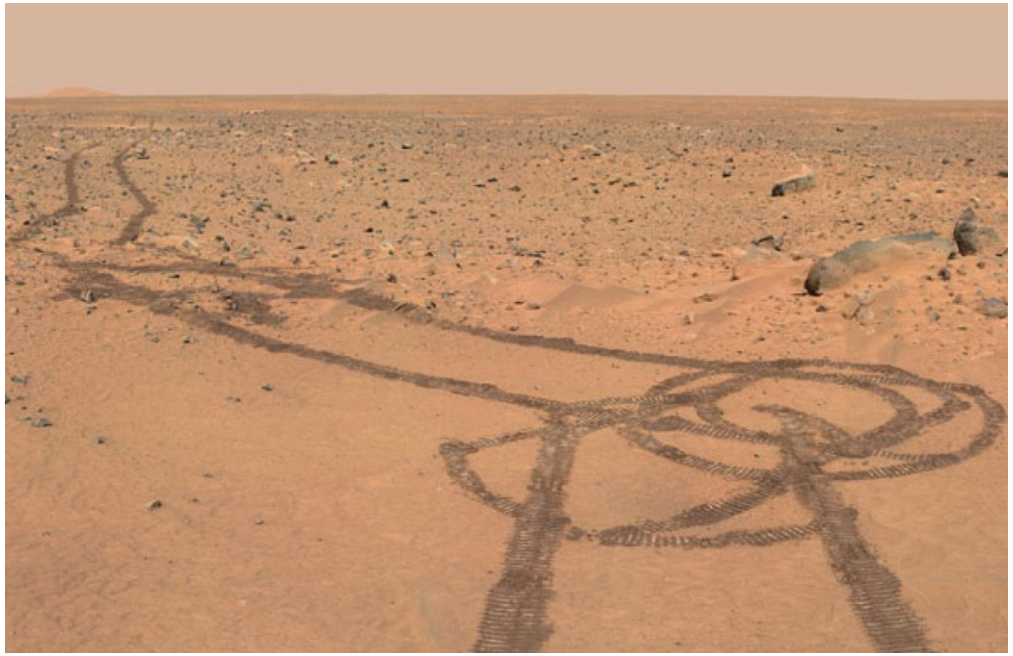
Inverse Reinforcement Learning



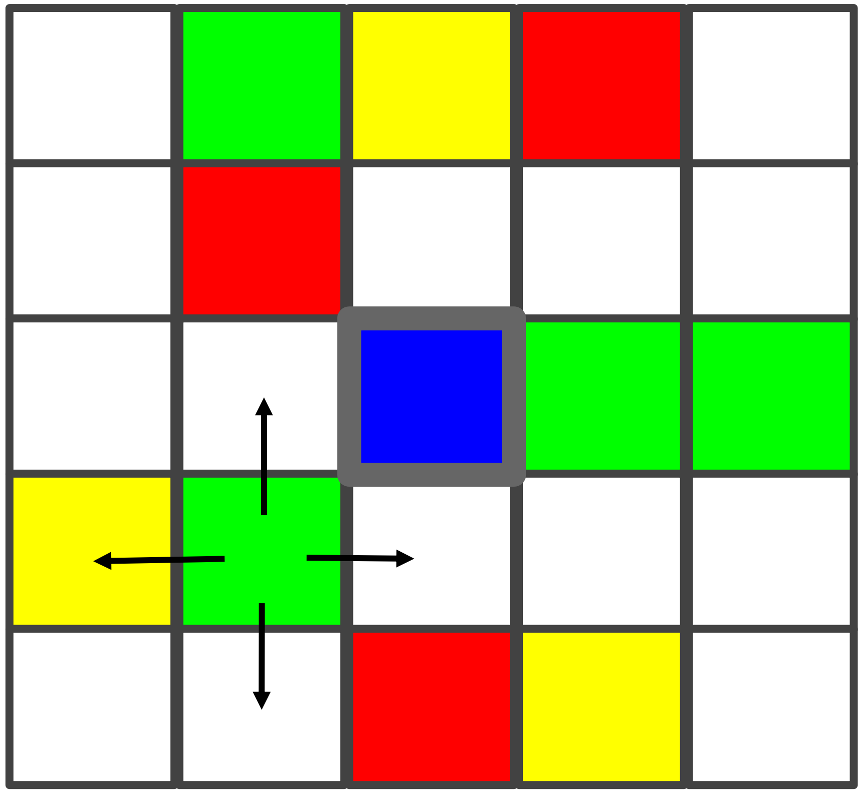
$\Rightarrow R \Rightarrow \pi$

- Answers the "Why?" question
- Explain the demonstrator's behavior
- Learn a reward function capturing the demonstrator's intent
- Can require lots of data and compute
- Better generalization. Can recover from arbitrary states

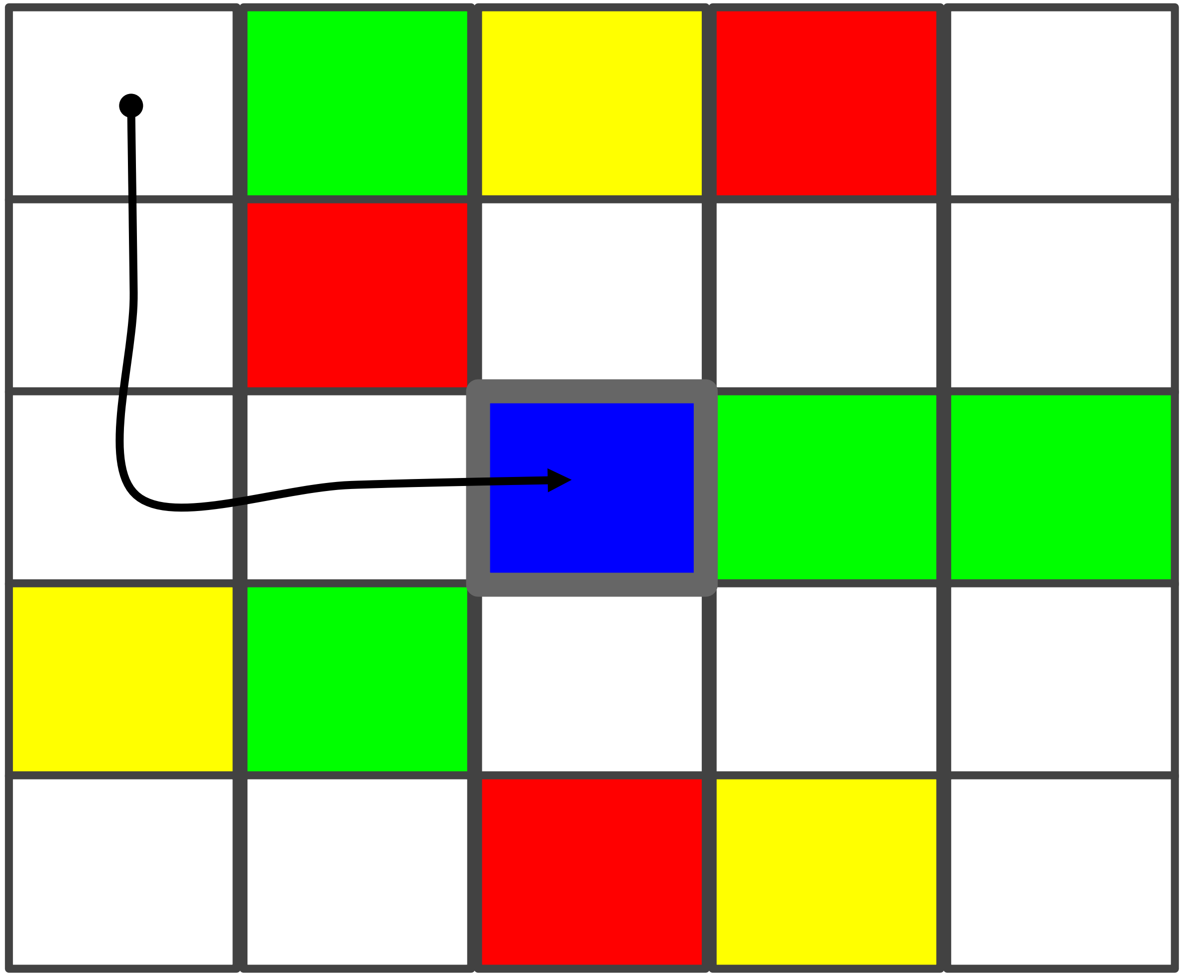
IRL Example: Teaching a robot to navigate through demonstrations

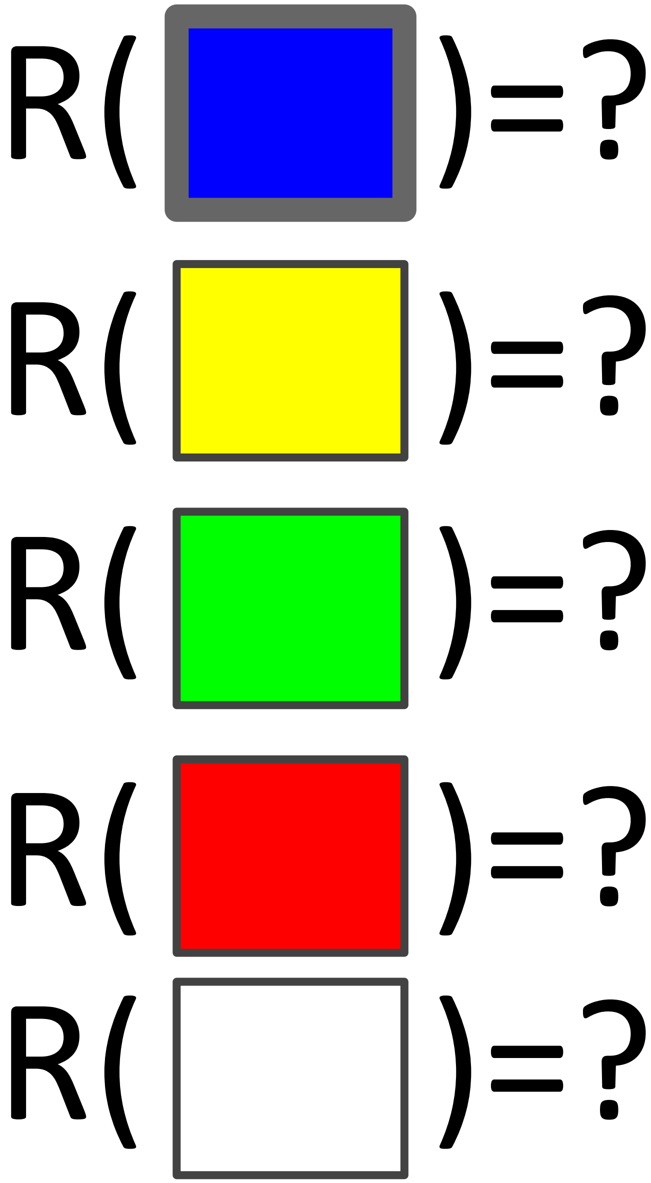


Toy version

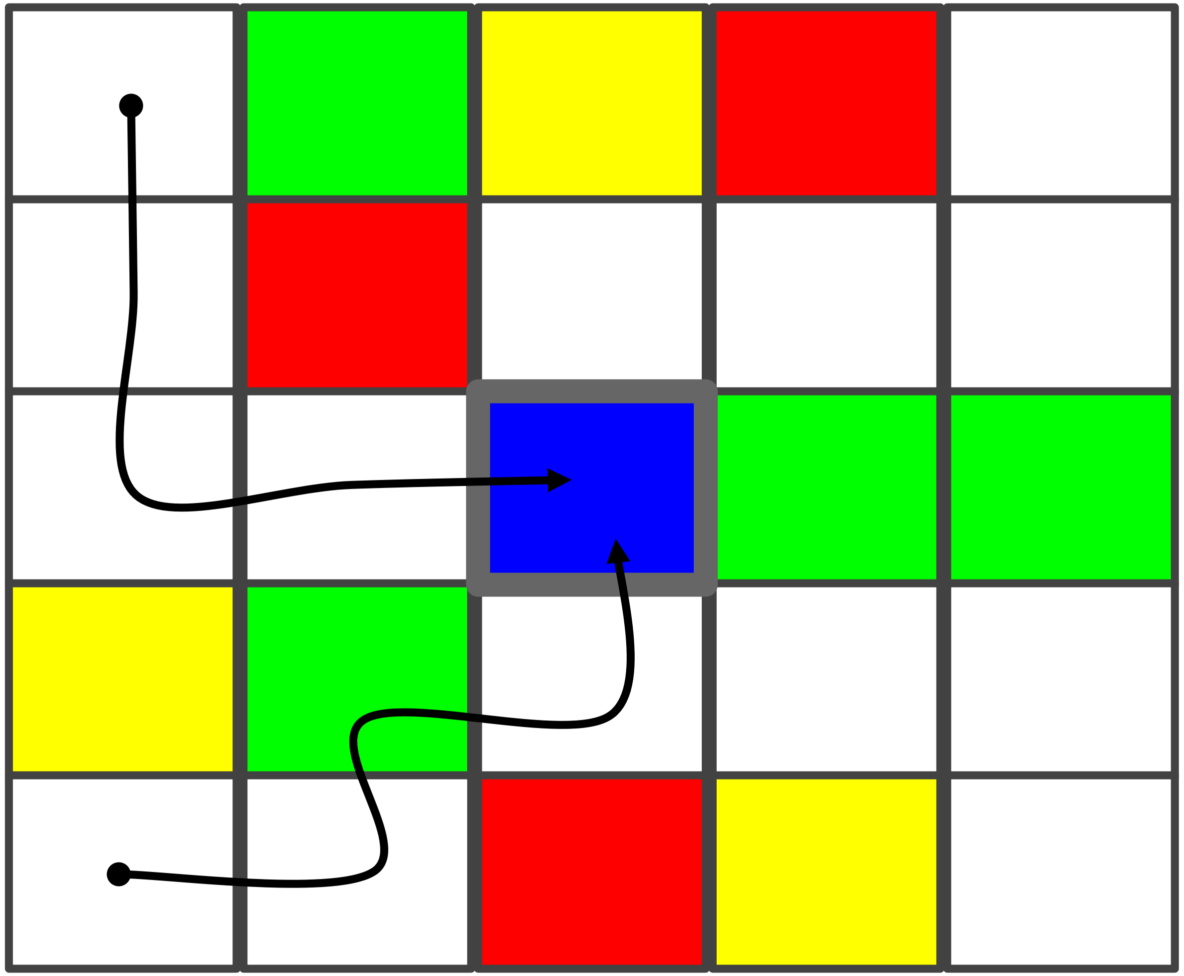


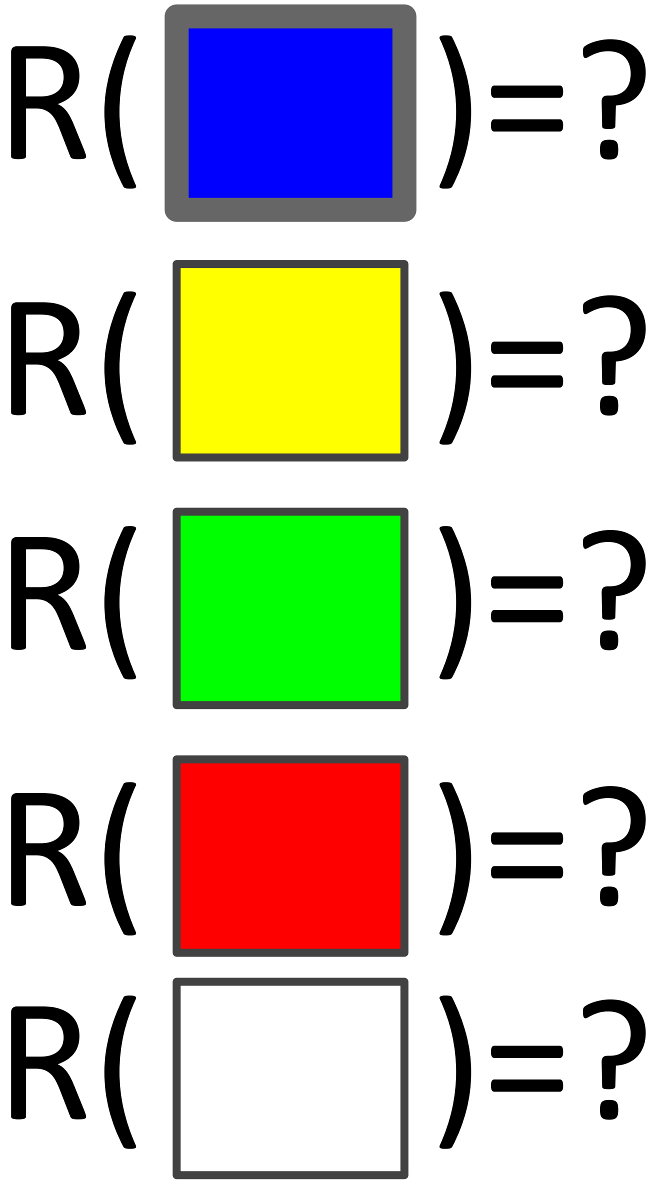
What is the reward?



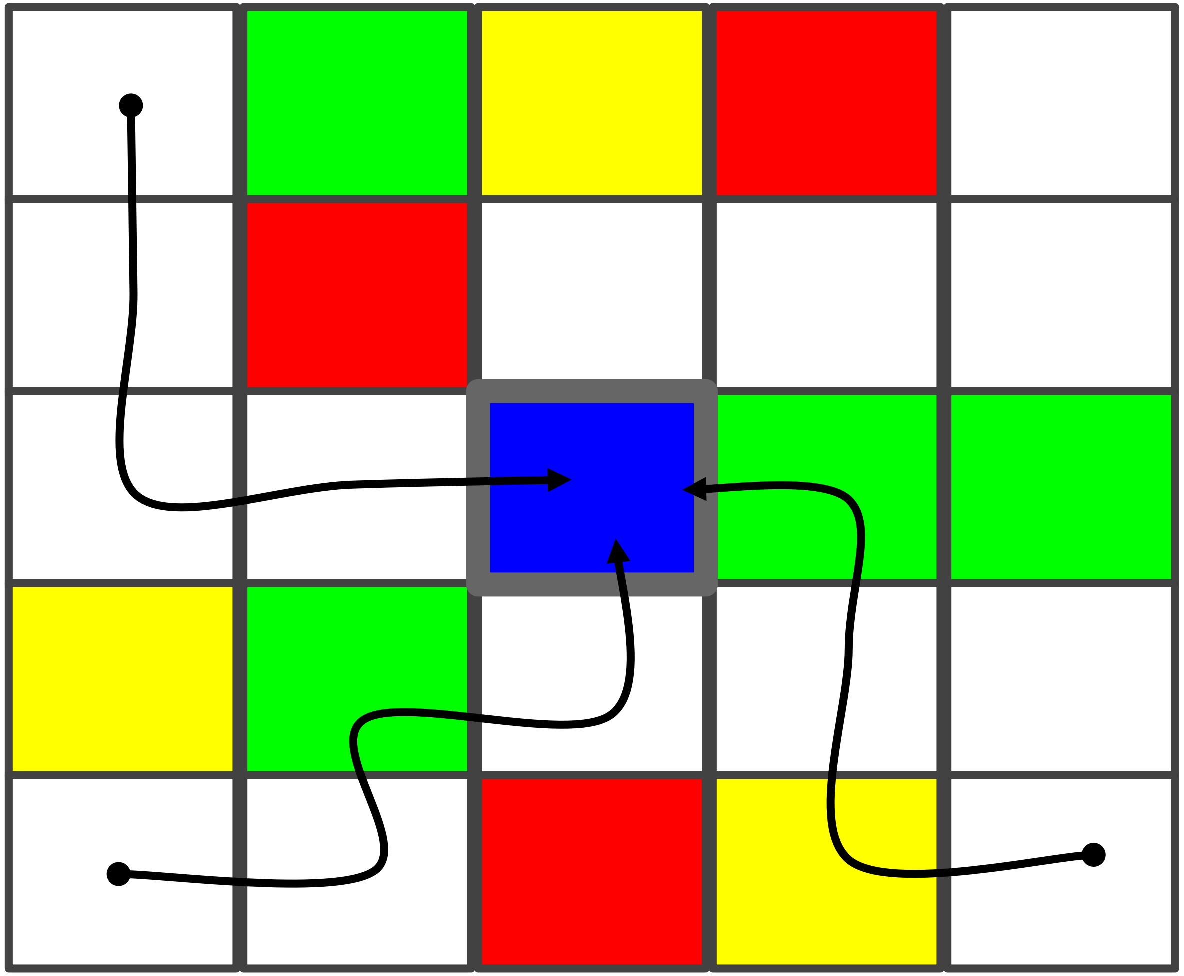


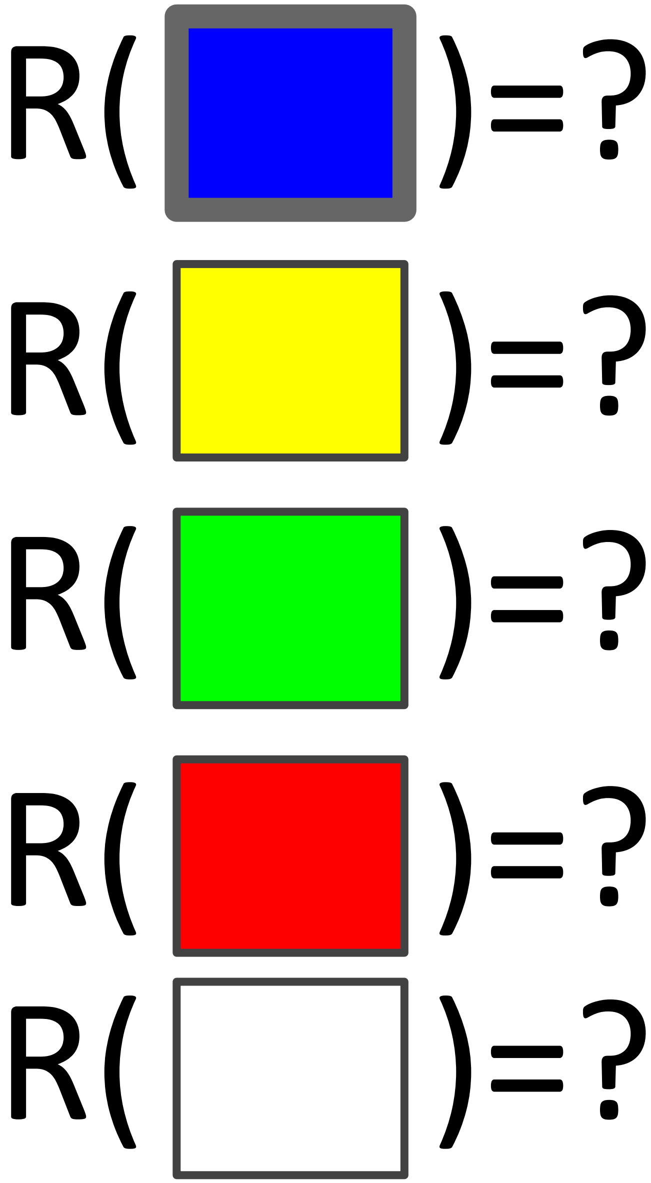
What is the reward?



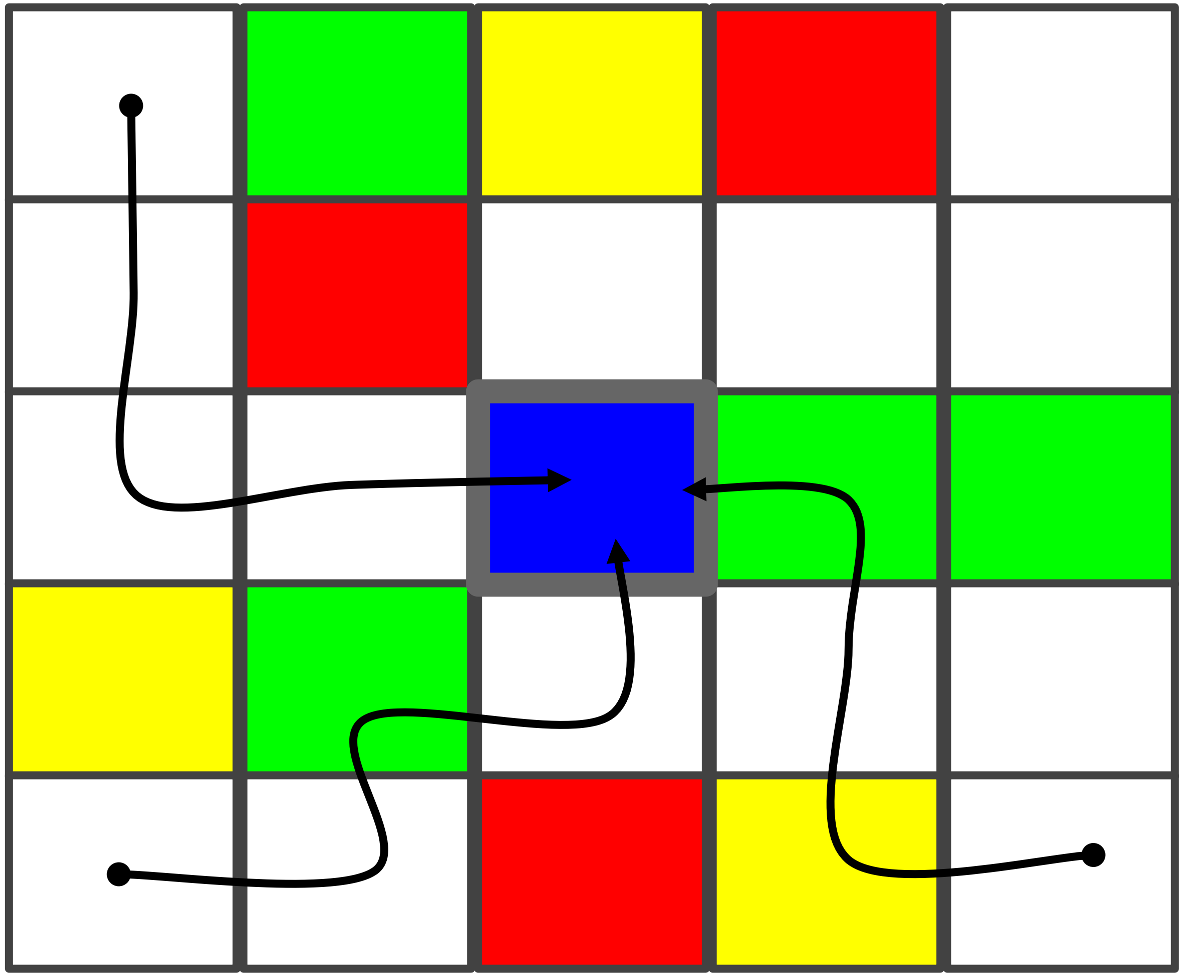


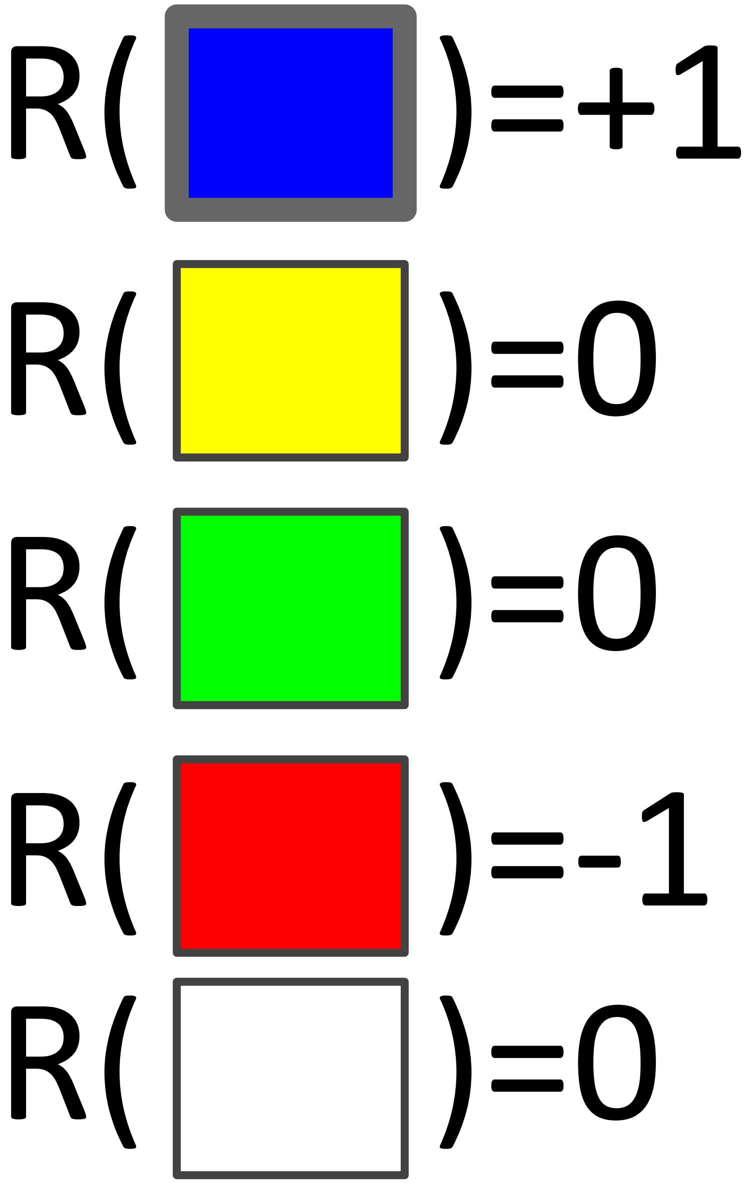
What is the reward?



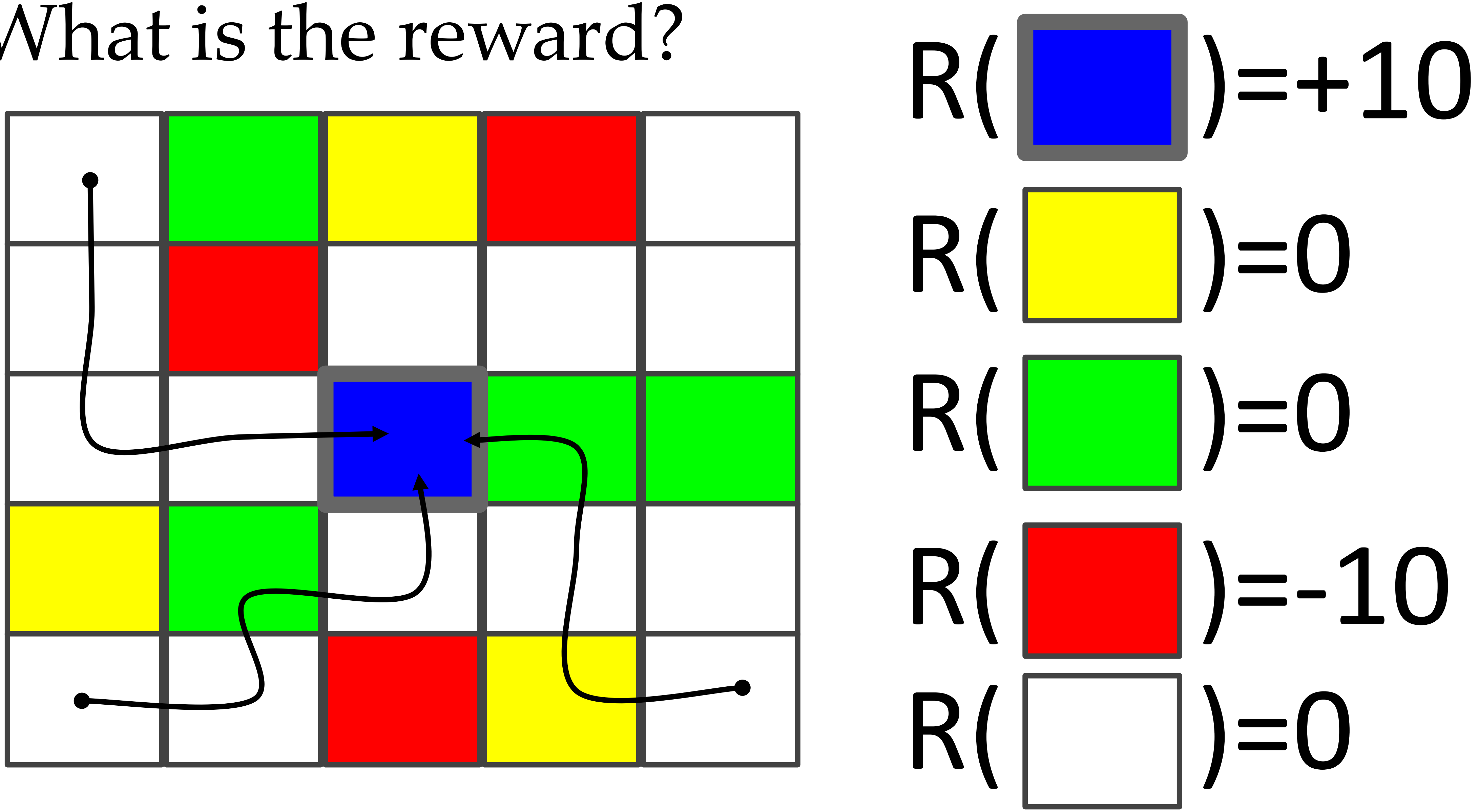


What is the reward?

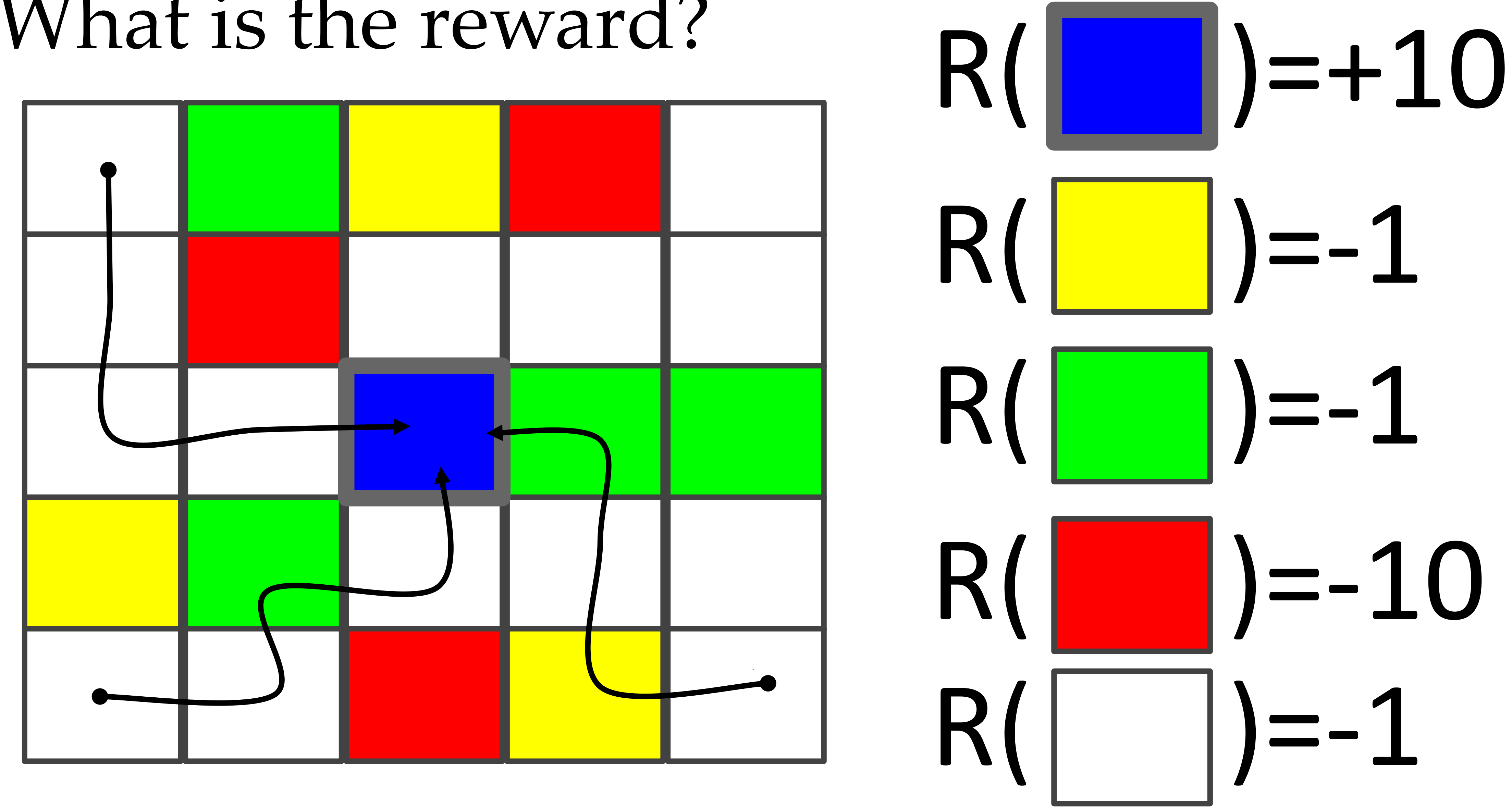




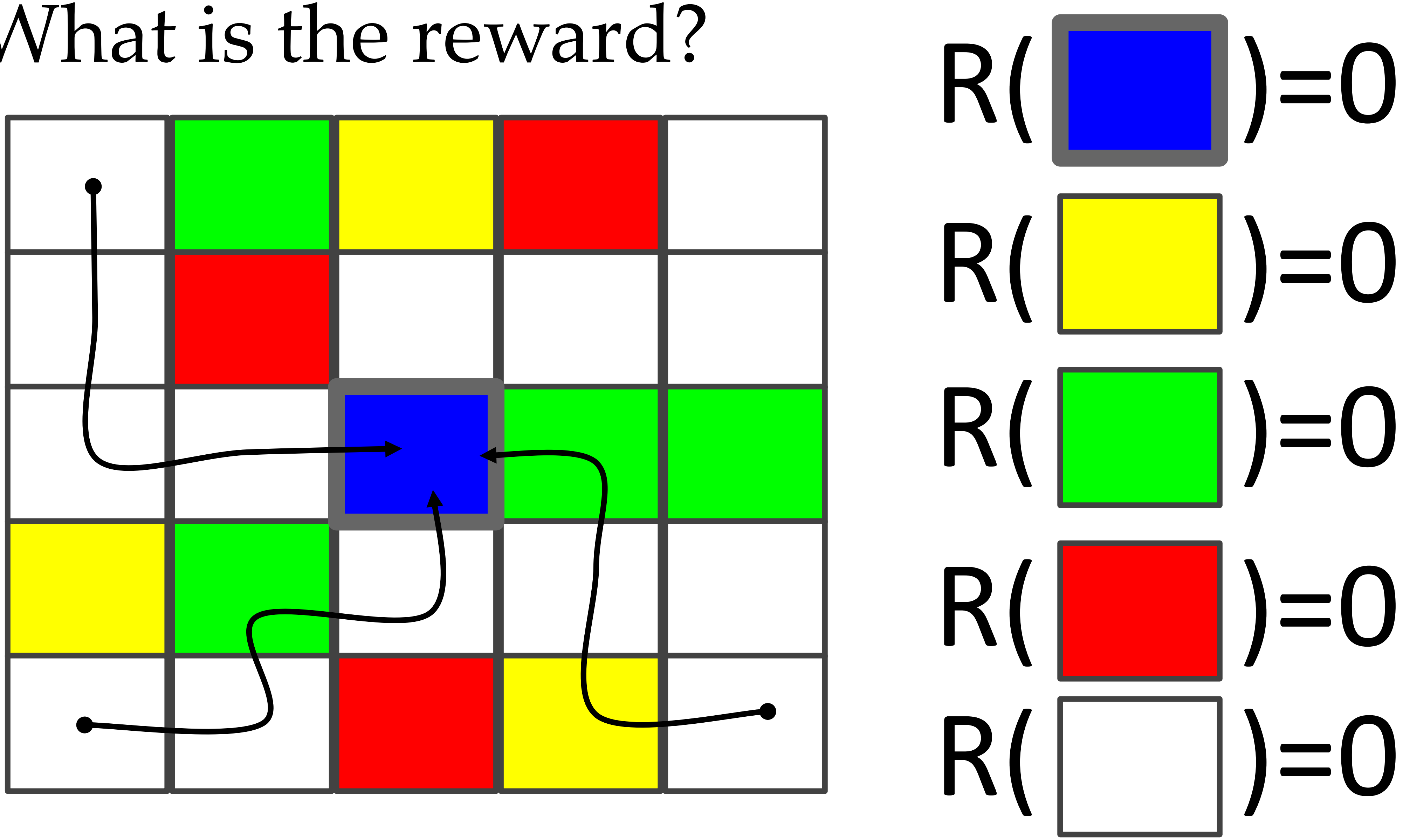
What is the reward?



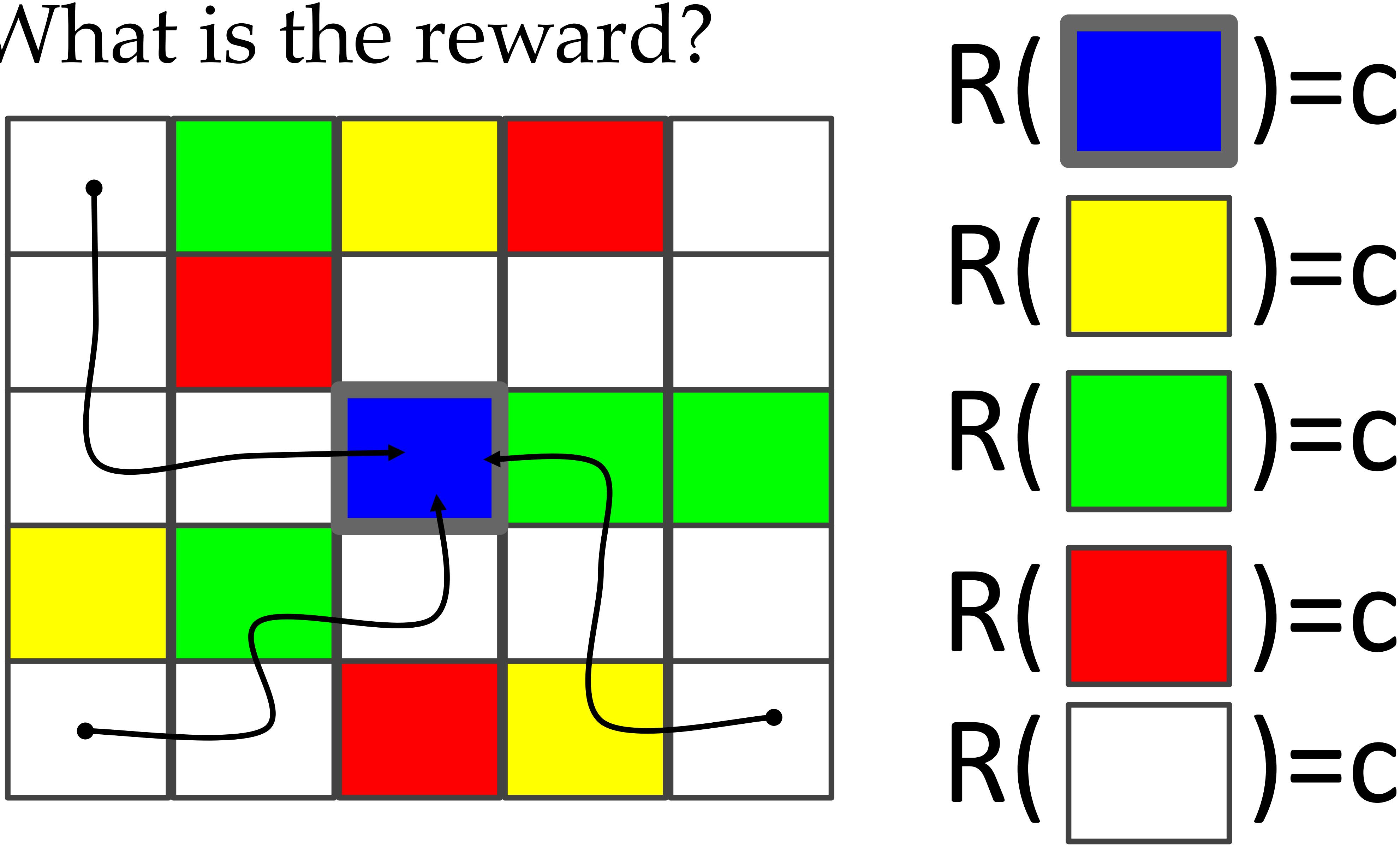
What is the reward?



What is the reward?



What is the reward?



Inverse Reinforcement Learning Formalism

- Given
  - MDP without a reward function
  - Demonstrations from an optimal policy π^*
- Recover a reward function $\boldsymbol{R}$ that makes π^* optimal

- Ill-Posed Problem
  - Infinite number of reward functions that can make π^* optimal
    - Trivial all zero reward
    - Constant reward
    - $aR + c$ (positive scaling a>0, and affine shifts)

Simpler problem: What if you know the policy?

How would you do this more generally?

Basic IRL Algorithm

- Start with demonstrations, D
- Guess initial reward function R_0
- $\hat{R} = R_0$
- Loop:
  - Solve for optimal policy $\pi^*_{\hat{R}}$
  - Compare D and $\pi^*_{\hat{R}}$
  - Update $\hat{R}$ to try and make D and $\pi^*_{\hat{R}}$ more similar

Feature count matching

- Assume the reward function is a linear combination of features:

$$R(s) = \mathbf{w}^T \phi(s)$$

- Value function becomes linear combination of (discounted) feature expectations:

$$V_R^\pi \quad = \quad \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \right]$$

Abbeel and Ng, "Apprenticeship learning via inverse reinforcement learning." ICML, 2004.

Feature count matching

- Assume the reward function is a linear combination of features:

$$R(s) = \mathbf{w}^T \phi(s)$$

- Value function becomes linear combination of (discounted) feature expectations:

$$V_R^\pi \quad = \quad \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathbf{w}^t \phi(s_t) \right]$$

Abbeel and Ng, "Apprenticeship learning via inverse reinforcement learning." ICML, 2004.

Feature count matching

- Assume the reward function is a linear combination of features:

$$R(s) = \mathbf{w}^T \phi(s)$$

- Value function becomes linear combination of (discounted) feature expectations:

$$V_R^\pi \quad = \quad \mathbf{w}^T \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \right] \quad = w^T \mu_\pi$$

Abbeel and Ng, "Apprenticeship learning via inverse reinforcement learning." ICML, 2004.

Inverse reinforcement learning: feature matching
(Abbeel and Ng 2004, Syed and Schapire 2007)

$|x^\top y| \leq \|x\|_1 \|y\|_\infty$

- If $||\boldsymbol{w}||_1 \leq 1$, then

$$
\begin{aligned}
V_R^{\pi^*} - V_R^{\pi_{\text{robot}}} &= \mathbf{w}^T(\mu_{\pi^*} - \mu_{\pi_{\text{robot}}}) \\
&\leq \|\mu_{\pi^*} - \mu_{\pi_{\text{robot}}}\|_\infty
\end{aligned}
$$

- If feature expectations match, then expected returns are identical.
- Idea: Can we update the reward guess $\hat{R}$ so the feature counts get closer?

Abbeel and Ng, "Apprenticeship learning via inverse reinforcement learning." ICML, 2004.

Problem: Many different policies can lead to same expected feature counts

Maximum Entropy IRL (Ziebart et al. 2008)

$$P(\tau) = \frac{e^{R_w(\tau)}}{Z}$$

- Collect M demonstrations $D = \{\tau_1, \ldots, \tau_M\}$
- Initialize reward weights ***w***

$$R(s) = \mathbf{w}^T \phi(s)$$

- **Loop**
  - Solve for (soft) optimal policy $\pi(a|s)$ via Value Iteration
  - Solve for expected feature counts of $\pi(a|\mathrm{s})$
  - Compute weight update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha(\mu_D - \mu_\pi)$

Soft Value Iteration

$$\pi_\Theta\left(A_t|S_t\right) = e^{Q^{\text{soft}}_{\pi_\Theta}(A_t,S_t) - V^{\text{soft}}_{\pi_\Theta}(S_t)}$$

$$V^{\text{soft}}_{\pi_\Theta}\left(S_t\right) = \log \sum_{A_t \in \mathcal{A}} e^{Q^{\text{soft}}_{\pi_\Theta}(A_t,S_t)}$$

Soft Maximum

Policy is a softmax policy.

Softmax is a Soft Maximum

- Assume b $>$ a
- $\log\left(e^{a} + e^{b}\right) =$

- If a = b
- $\log\left(e^{a} + e^{b}\right) =$

- In general $\max\{x_1, x_2, \ldots, x_n\} \leq \log \sum_i \exp(x_i) \leq \max\{x_1, \ldots, x_n\} + \log n$

Soft Value Iteration

$$\pi_\Theta(A_t|S_t) = e^{Q^{\mathrm{soft}}_{\pi_\Theta}(A_t,S_t) - V^{\mathrm{soft}}_{\pi_\Theta}(S_t)}$$

Soft Maximum

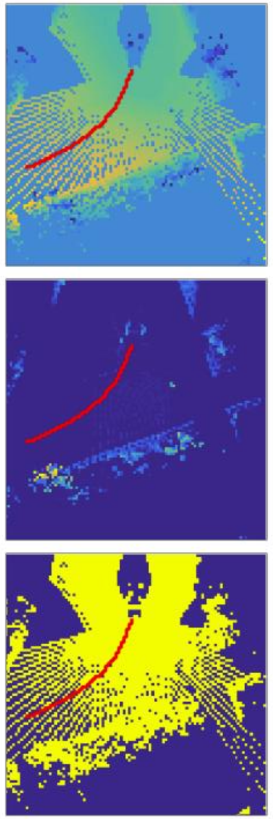
$$V^{\mathrm{soft}}_{\pi_\Theta}(S_t) = \log \sum_{A_t \in \mathcal{A}} e^{Q^{\mathrm{soft}}_{\pi_\Theta}(A_t,S_t)}$$

$$Q^{\mathrm{soft}}_{\pi_\Theta}(A_t,S_t) = R_\Theta(S_t,A_t) + \sum_{S' \in \mathcal{S}} P_T(S'|A_t,S_t)\, V^{\mathrm{soft}}_{\pi_\Theta}(S')$$

- Initialize values
- Repeat until convergence:
  - Solve for Q
  - Solve for V

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

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



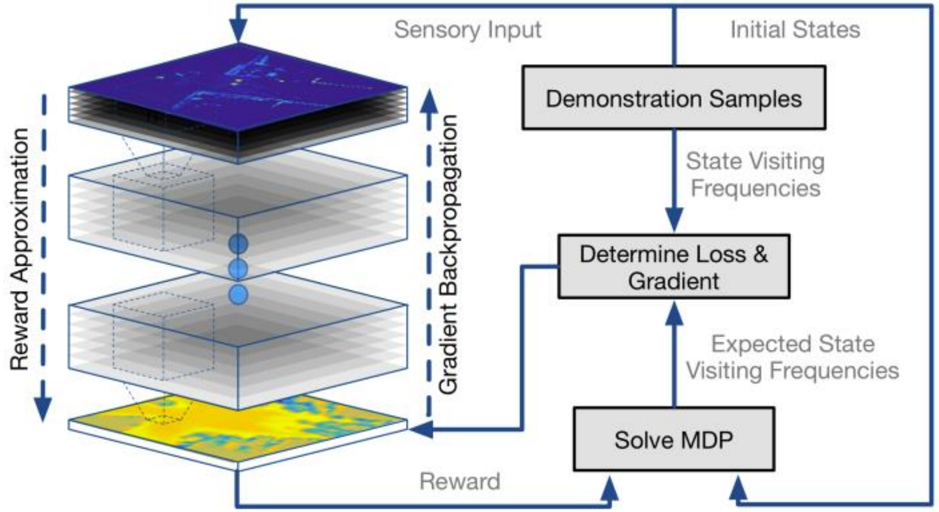


Fig. 1: Schema for training neural networks in the Maximum Entropy paradigm for IRL.

Another way to look at MaxEnt IRL

$$P(\tau) = \frac{e^{R_w(\tau)}}{Z} \qquad Z = \int e^{R_w(\tau)} d\tau$$

- Maximum Likelihood Estimation
- Find reward function that maximizes the log likelihood of the demonstration trajectories:

$$\max_\theta \frac{1}{N} \sum_{\tau \in D} R_w(\tau) - \log Z$$

How to avoid fully solving MDP

$$\max_{\theta} \frac{1}{N} \sum_{\tau \in D} R_w(\tau) - \log Z \qquad Z = \int e^{R_w(\tau)} d\tau$$

- Estimate Z with a finite set of trajectories Z_τ.
- Loop:
  - Update parameters w so demonstrations have higher reward than trajectories in Z_τ.
  - Update Z_τ

How to make this more tractable

$$P(\tau) = \frac{e^{R_w(\tau)}}{Z}$$

Relative Entropy Inverse Reinforcement Learning

Abdeslam Boularias, Jens Kober, Jan Peters
Max-Planck Institute for Intelligent Systems
72076 Tübingen, Germany
{abdeslam.boularias,jens.kober,jan.peters}@tuebingen.mpg.de

Uniform sampling to approximate Z.

Learning Objective Functions for Manipulation

Mrinal Kalakrishnan*, Peter Pastor*, Ludovic Righetti*†, and Stefan Schaal*†
kalakris@usc.edu, pastorsa@usc.edu, ludovic.righetti@a3.epfl.ch, sschaal@usc.edu
*CLMC Lab, University of Southern California, Los Angeles CA 90089
†Max Planck Institute for Intelligent Systems, Tübingen, Germany 72076

Noisy perturbations of demonstrations to approximate Z

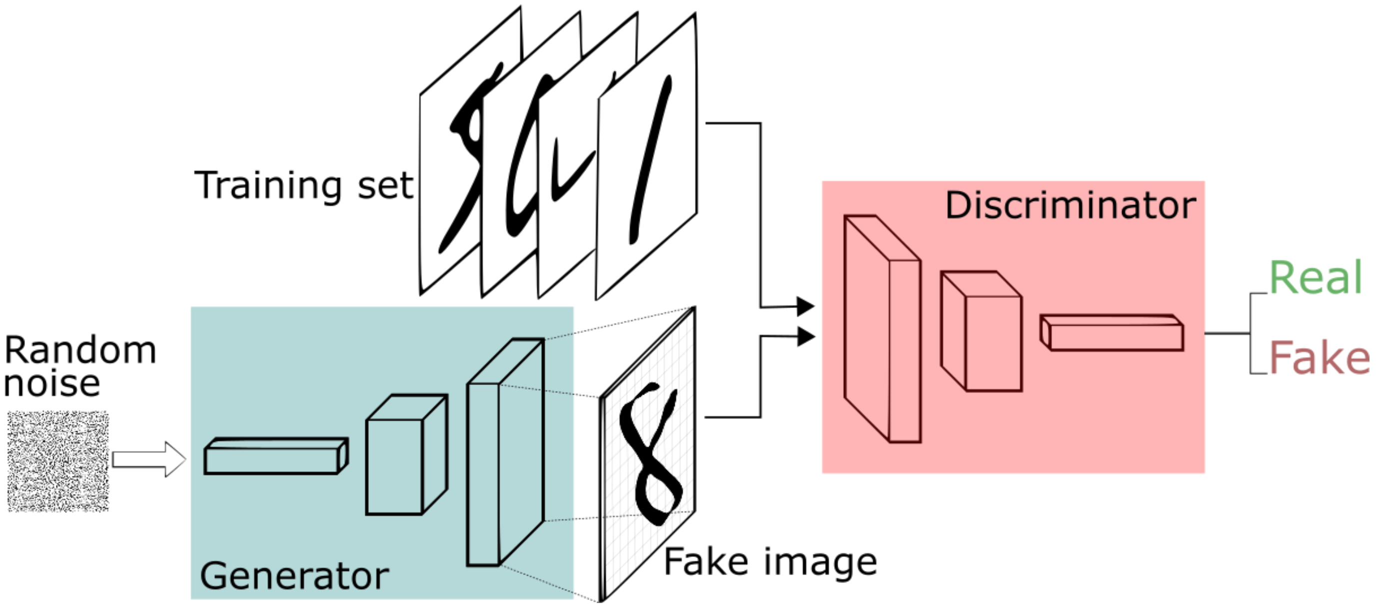
Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization

Chelsea Finn CBFINN@EECS.BERKELEY.EDU
Sergey Levine SVLEVINE@EECS.BERKELEY.EDU
Pieter Abbeel PABBEEL@EECS.BERKELEY.EDU
University of California, Berkeley, Berkeley, CA 94709 USA

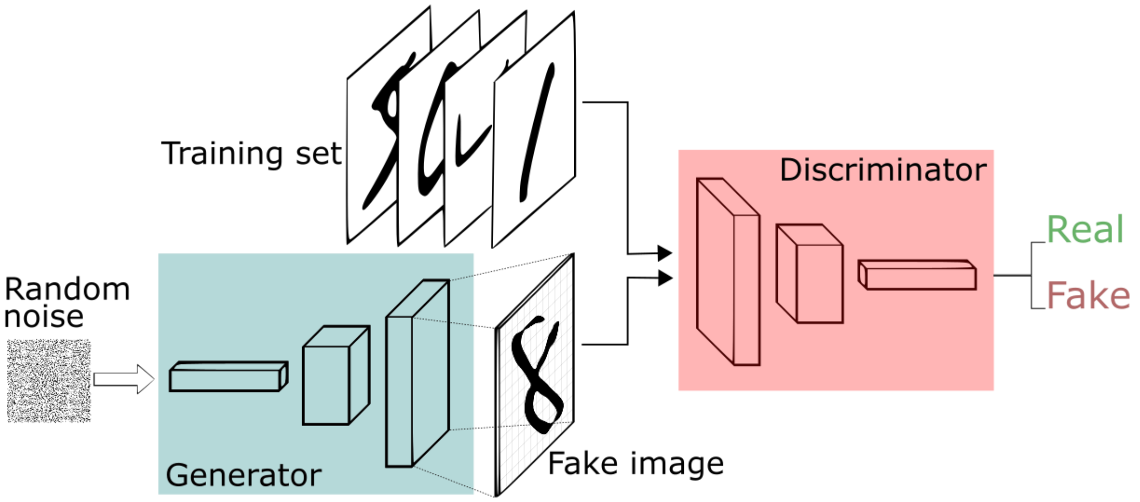
Use current policy to approximate Z. Alternate between a few steps of reward updates and a few steps of policy updates.

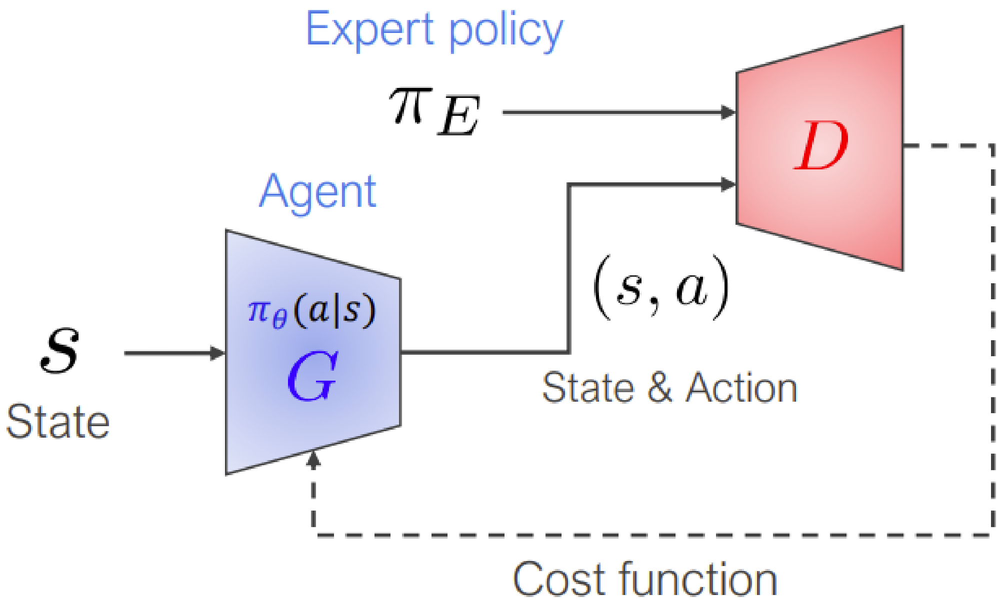
Finn et al. “Guided Cost Learning.” 2016

GANs (Generative Adversarial Networks)



GAIL (Generative Adversarial Imitation Learning)





Ho and Ermon, 2016

What if we don't want just a single reward estimate?

- Can we get a samples from the full Bayesian posterior?

$$P(R|D) \propto P(D|R)P(R)$$

Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

- Assume demonstrator is Boltzmann rational
  - Demonstrator follows a softmax policy with inverse temperature c

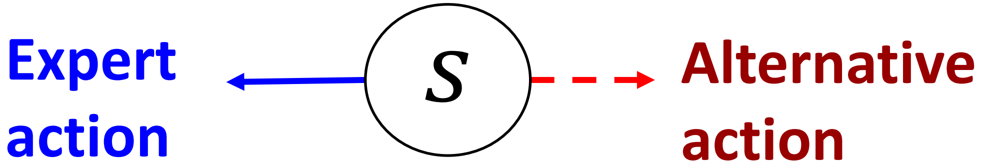
$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta \boxed{Q^*(s,a,R)}}}{\sum_{b\in A} e^{\beta Q^*(s,b,R)}}$$

$Q^*(s, a, R) =$ How much reward will I expect to see if I take action a in state s and act optimally thereafter.

Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

- Assume demonstrator is Boltzmann rational
  - Demonstrator follows a softmax policy with inverse temperature β

$$P(D|R) = \prod_{(s,a)\in D} \boxed{\frac{e^{\beta Q^*(s,a,R)}}{\sum_{b\in A} e^{\beta Q^*(s,b,R)}}}$$



$$P((s,\leftarrow)|R) = \frac{e^{Q^*(s,\leftarrow,R)}}{e^{Q^*(s,\leftarrow,R)} + e^{Q^*(s,\dashrightarrow,R)}}$$

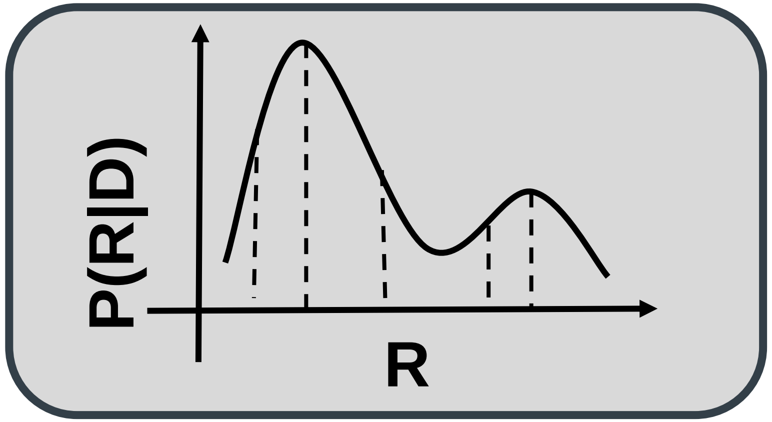
Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

- Assume demonstrator is Boltzmann rational
  - Demonstrator follows a softmax policy with inverse temperature β

$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*(s,a,R)}}{\sum_{b\in A} e^{cQ^*(s,b,R)}}$$

  - Perform Bayesian inference (MCMC) to sample from posterior distribution

$$P(R|D) \propto P(D|R)P(R)$$



Applications of Bayesian IRL

- Active Learning
- Uncertainty Estimation
- Demonstration Sufficiency

Autonomous Assessment of Demonstration Sufficiency via Bayesian Inverse Reinforcement Learning

Tu (Alina) Trinh
University of California, Berkeley

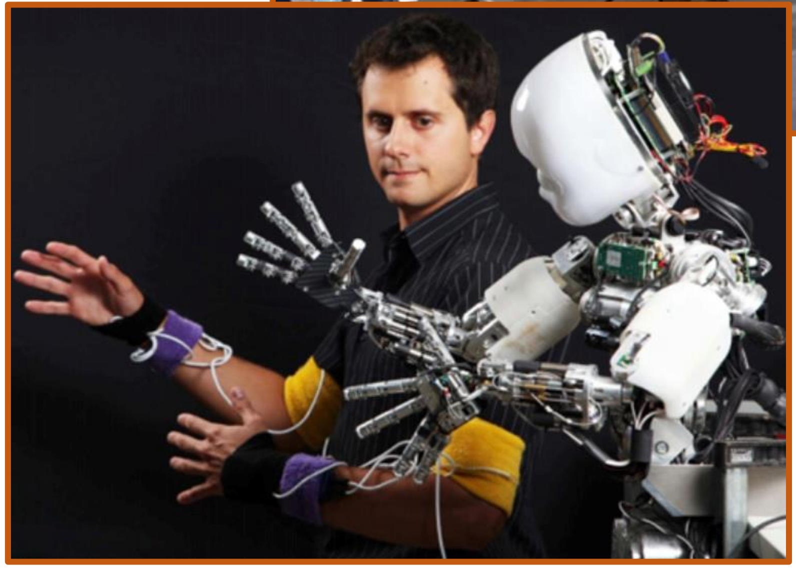
Haoyu Chen
University of Utah

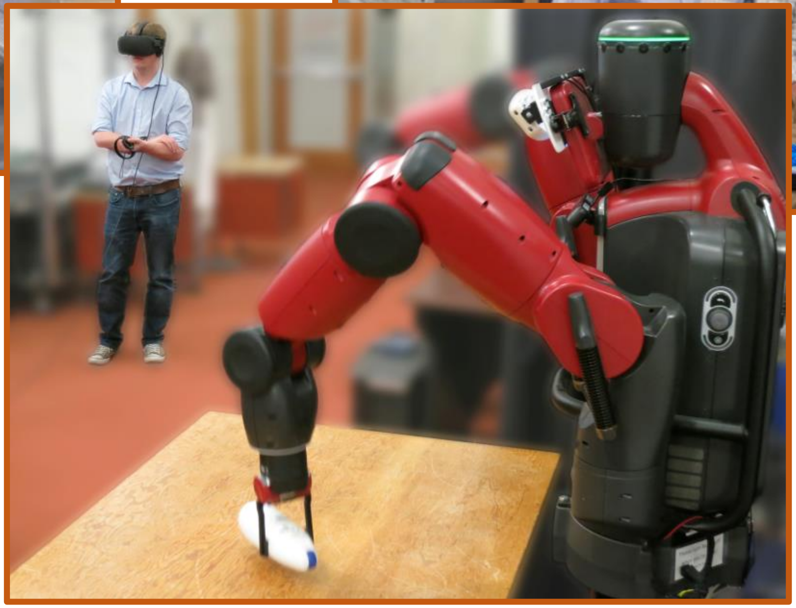
Daniel S. Brown
University of Utah


igus

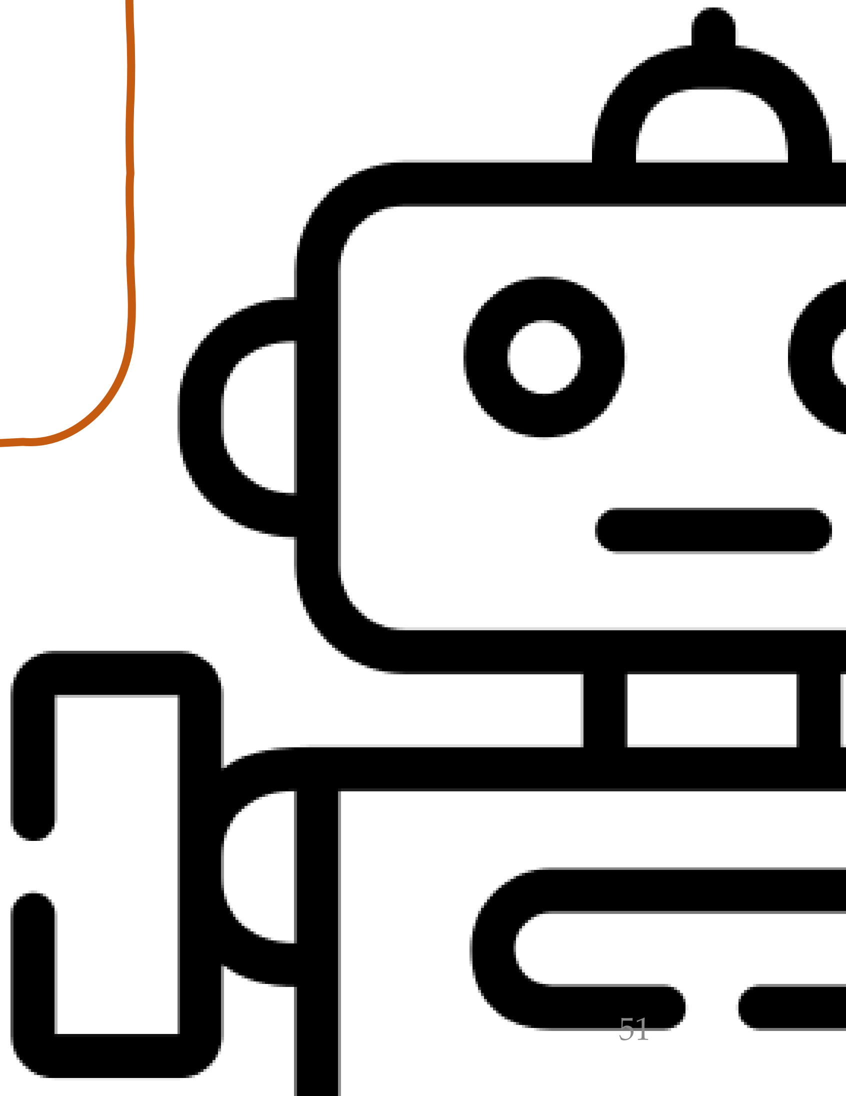

RIBA II

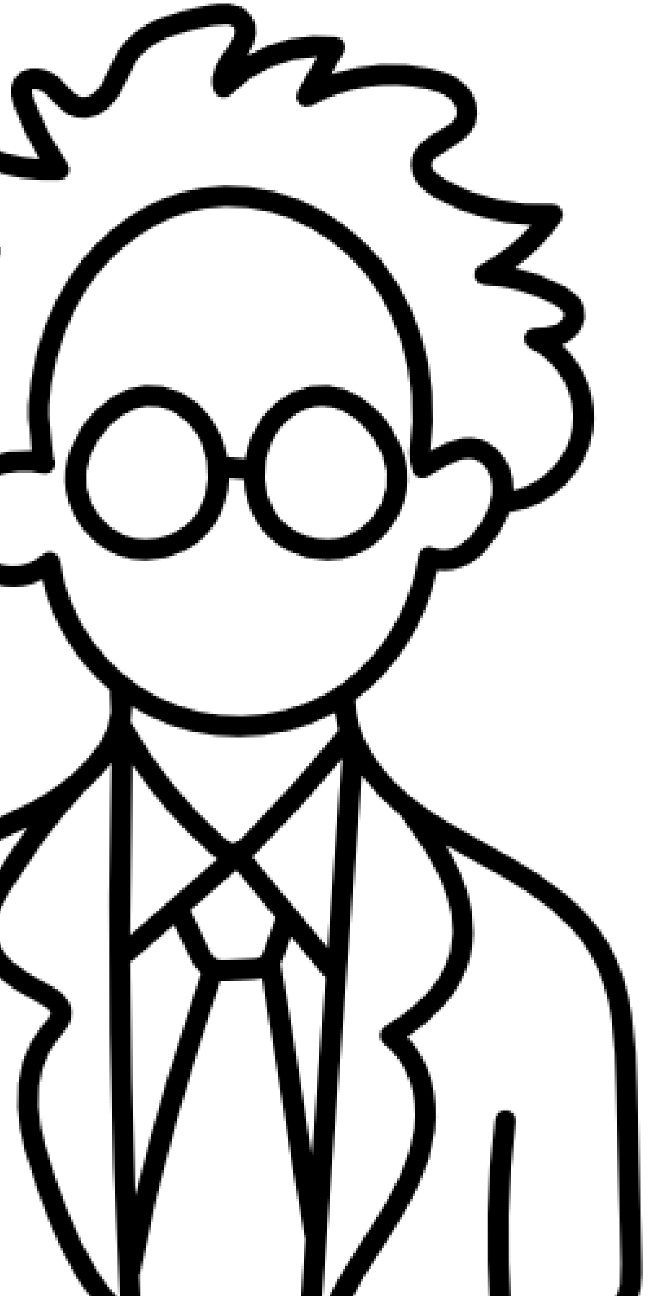
Learning From Demonstration (**LfD**)



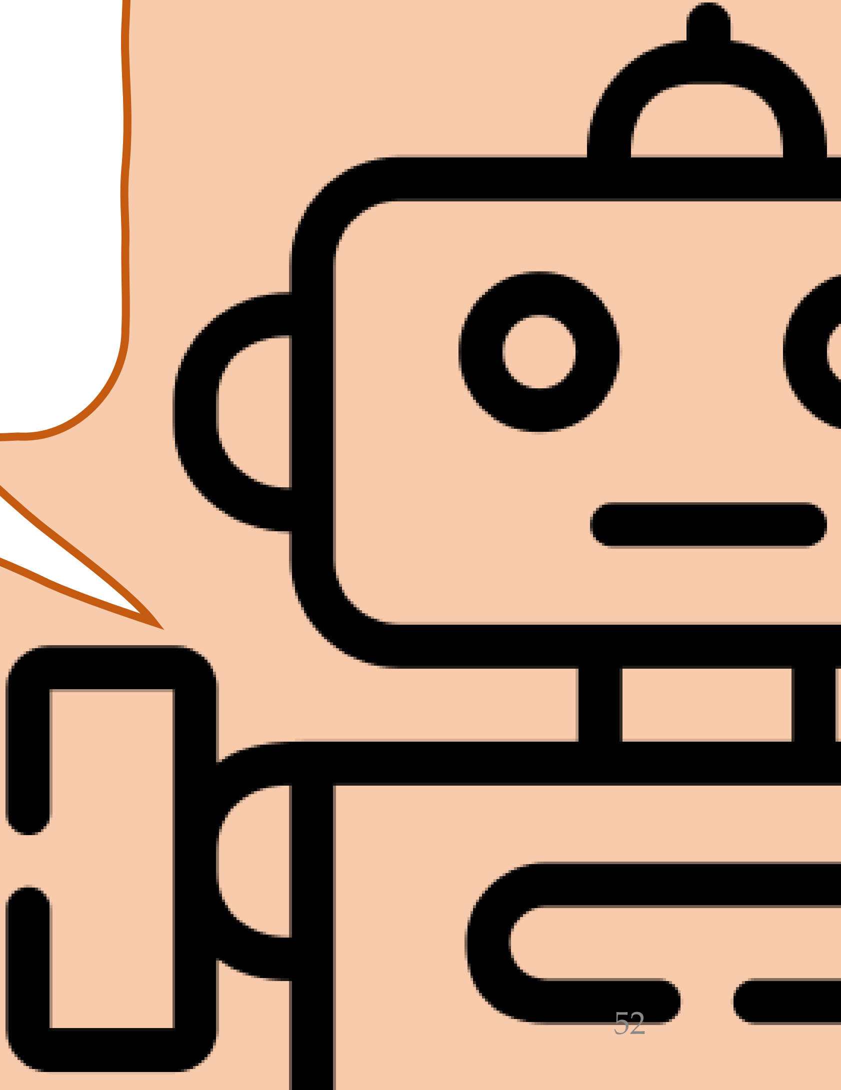


- Have I provided enough demonstrations?
- Are my demonstrations informative enough?
- Should I just supervise the robot?





- Have I received enough demonstrations?
- Are these demonstrations informative enough?

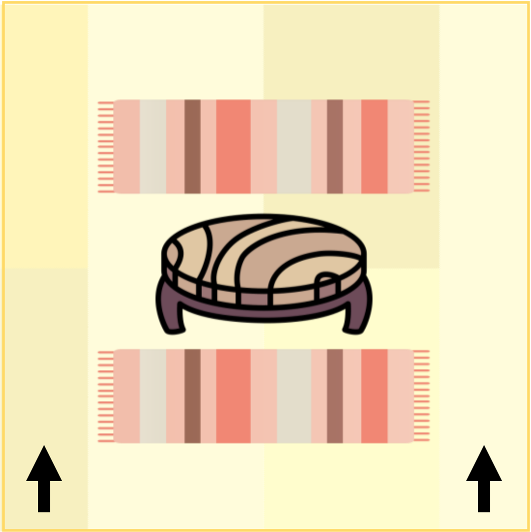


Determining Demonstration Sufficiency

Demonstration Insufficiency

Determining Demonstration Sufficiency

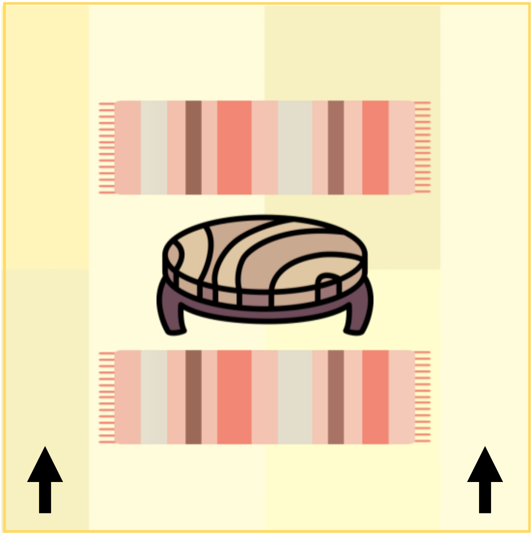
Demonstration Insufficiency



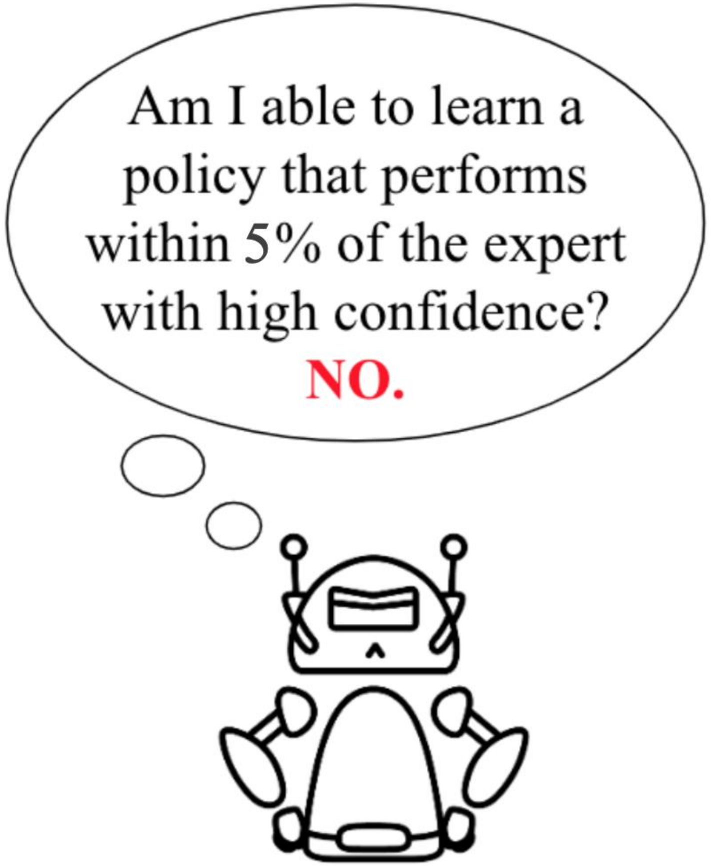
Uninformative Demos

Determining Demonstration Sufficiency

Demonstration Insufficiency

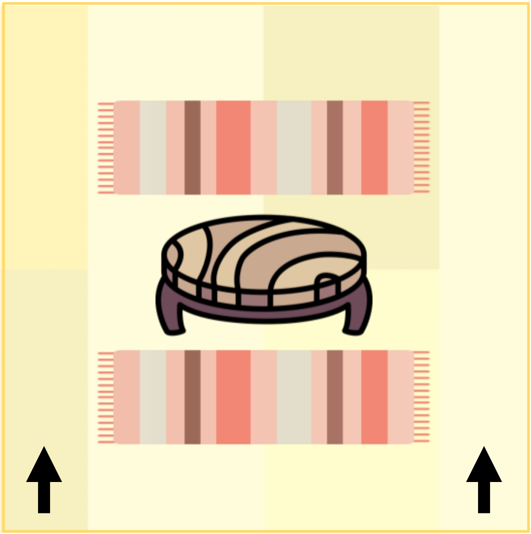


Uninformative Demos

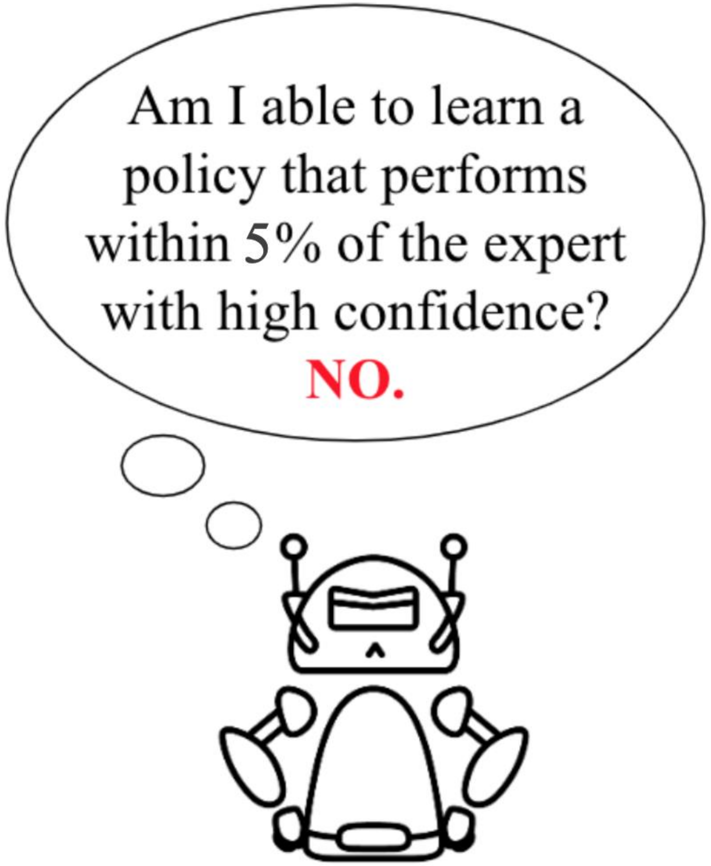


Determining Demonstration Sufficiency

Demonstration Insufficiency



Uninformative Demos



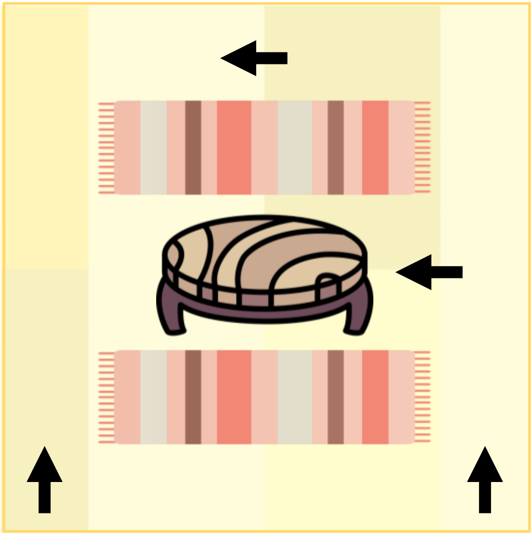
More demos needed

Determining Demonstration Sufficiency

Demonstration Sufficiency

Determining Demonstration Sufficiency

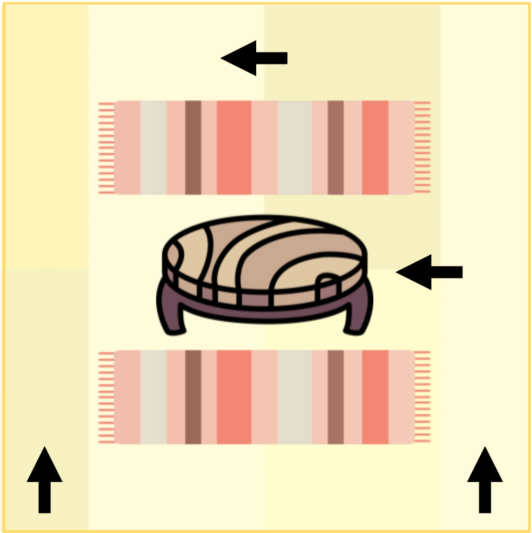
Demonstration Sufficiency



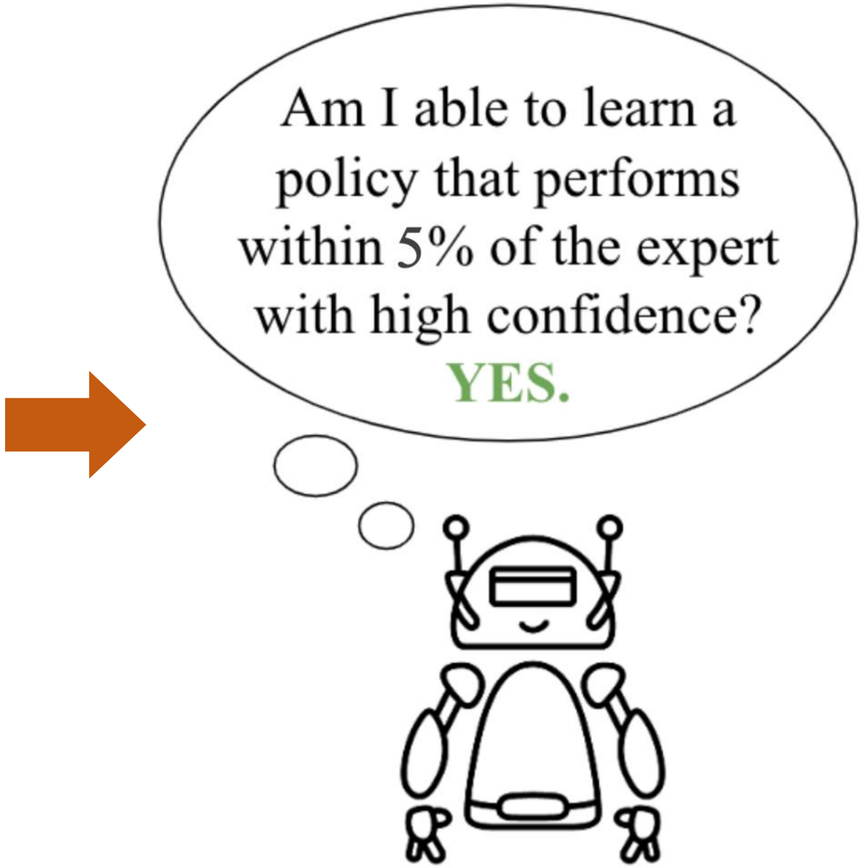
Demos

Determining Demonstration Sufficiency

Demonstration Sufficiency

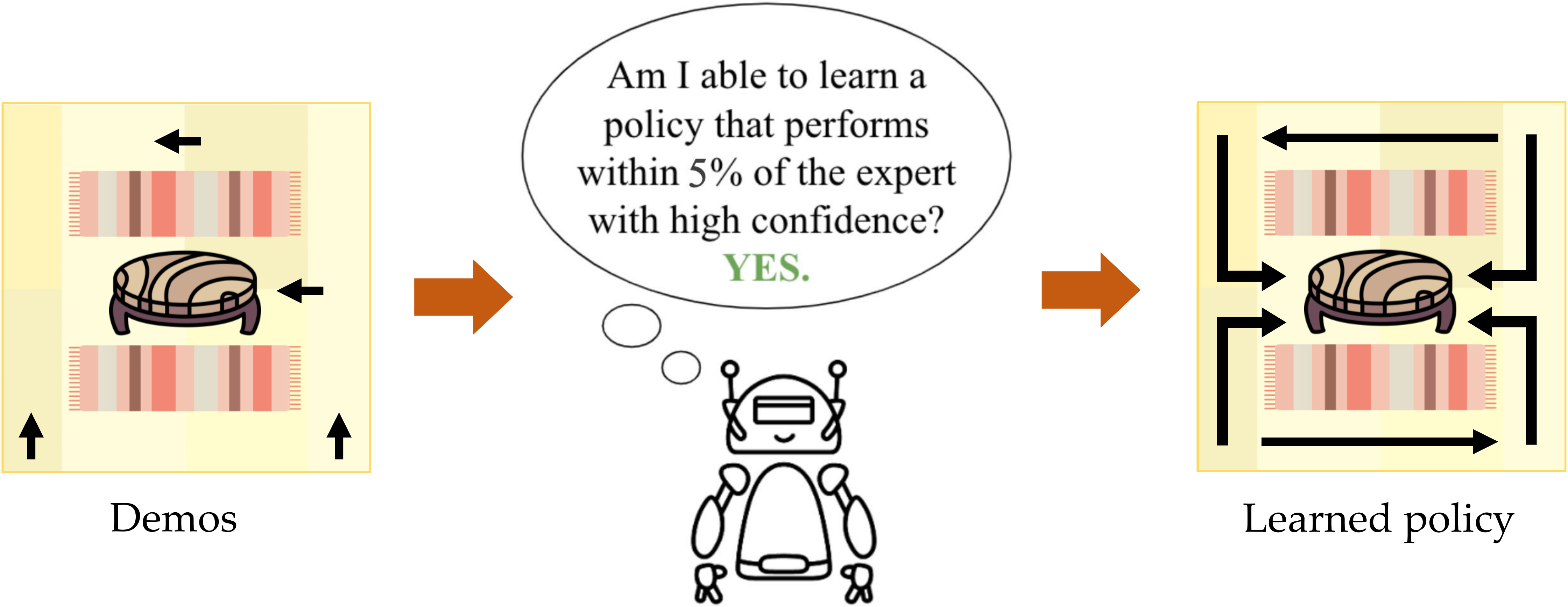


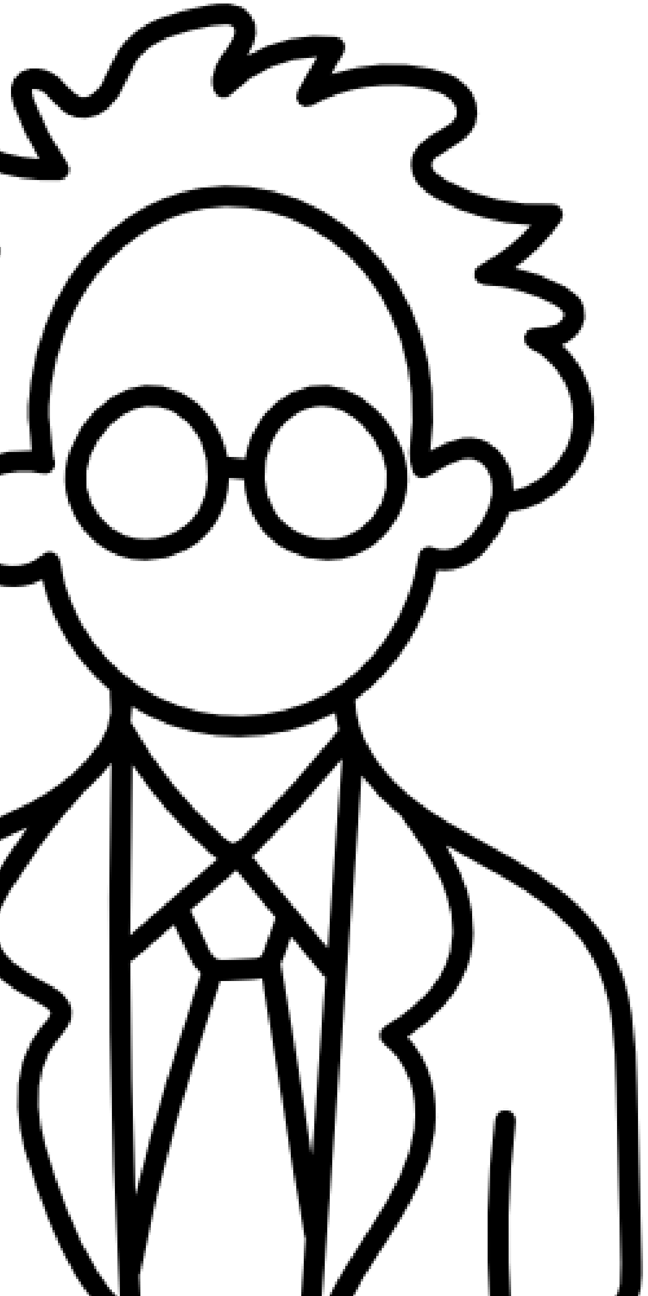
Demos



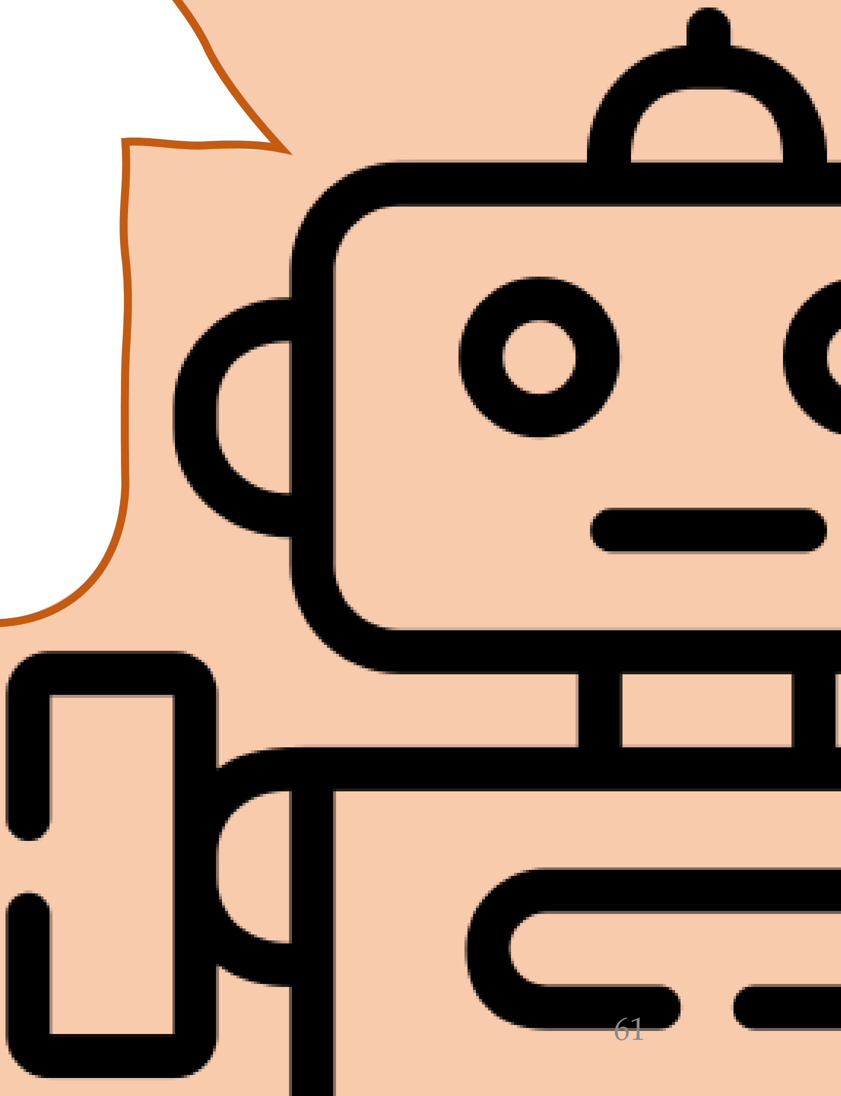
Determining Demonstration Sufficiency

Demonstration Sufficiency





- Have I received enough demonstrations?
- Are these demonstrations informative enough?
- **What is the reward function?**
- **How do I measure policy "goodness"?**



Estimating the Reward

Bayesian
IRL

Estimating the Reward

$(s, a) \in \mathcal{D}$

demonstrations

$\longrightarrow$ $\boxed{\text{Bayesian IRL}}$

Estimating the Reward

$$P(R|D) \propto P(D|R)$$

$$= \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b \in A} e^{\beta Q^*_R(s,b)}}$$

$(s, a) \in \mathcal{D}$

demonstrations → Bayesian IRL

Estimating the Reward

$$P(R|D) \propto P(D|R)$$

$$= \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}}$$

$(s, a) \in \mathcal{D}$ $\rightarrow$ $\boxed{\text{Bayesian IRL}}$ $\rightarrow$ R_{robot}

demonstrations

estimated reward function

Measuring Policy Goodness

- Normalized expected value difference (**nEVD**)

$$nEVD(\pi_{\text{robot}}, R^*) = \frac{V^*_{R^*} - V^{\pi_{\text{robot}}}_{R^*}}{V^*_{R^*} - V^{\pi_{\text{rand}}}_{R^*}}$$

- Puts policy regret in interpretable percentage form

Measuring Policy Goodness

- Normalized expected value difference (**nEVD**)

$$nEVD(\pi_{\text{robot}}, R^*) = \frac{V^*_{R^*} - V^{\pi_{\text{robot}}}_{R^*}}{V^*_{R^*} - V^{\pi_{\text{rand}}}_{R^*}}$$

- Puts policy regret in interpretable percentage form
- We only have an estimate of R*, so…

Measuring Policy Goodness

$$P(R|D) \propto P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}}$$

$$(s, a) \in \mathcal{D}$$

demonstrations

$$\boxed{\text{Bayesian IRL}}$$

$$R_{\text{robot}}$$

estimated reward function

$$R_1, \ldots, R_n$$

$$\pi_{\text{robot}}$$

Measuring Policy Goodness

$$P(R|D) \propto P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}}$$

$(s,a) \in \mathcal{D}$

demonstrations

Bayesian IRL

R_{robot}

estimated reward function

$R_1, \ldots, R_n$

π_{robot}

$$\nu_\alpha\left(nEVD(\pi_{\text{robot}}, R)\right)$$

α worst-case bound on nEVD

Measuring Policy Goodness

$$P(R|D) \propto P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}}$$

$(s, a) \in \mathcal{D}$

demonstrations

Bayesian IRL

R_{robot}

estimated reward function

$R_1, \ldots, R_n$

π_{robot}

demonstration sufficiency!

$$\varepsilon > \nu_\alpha(nEVD(\pi_{\text{robot}}, R))$$

α worst-case bound on nEVD

Comparing With Theoretical Bounds

How many demonstrations is enough for a simple gridworld?

nEVD Threshold	Ours	Abbeel and Ng '04	Syed and Schapire '07
0.1	17	1,600,000	3,700,000
0.3	16	180,000	410,000
0.5	15	65,000	150,000

$\underbrace{\text{Abbeel and Ng '04} \quad \text{Syed and Schapire '07}}_{\text{based on Chernoff-Hoeffding bound}}$

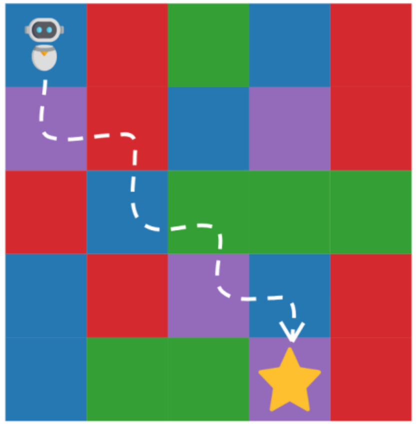
[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. ICML 2004.
[2] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. NeurIPS 2007.

Baselines and Environments

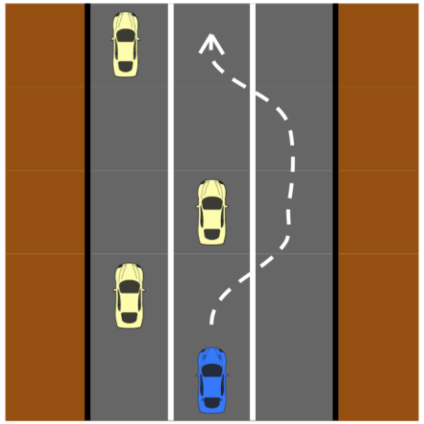
- Convergence heuristic
- Validation set heuristic

Baselines and Environments

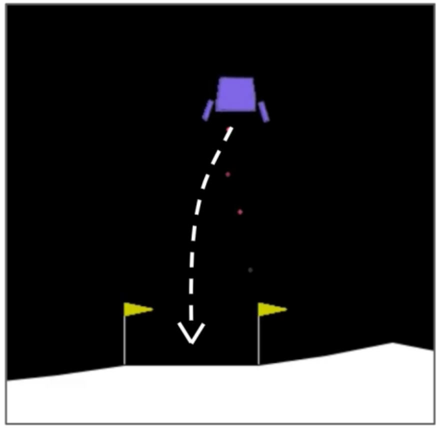
- Convergence heuristic
- Validation set heuristic



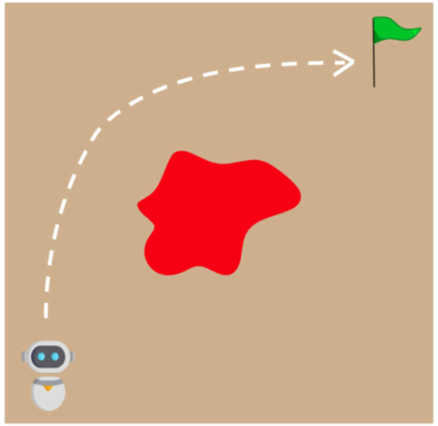
Gridworld



Driving

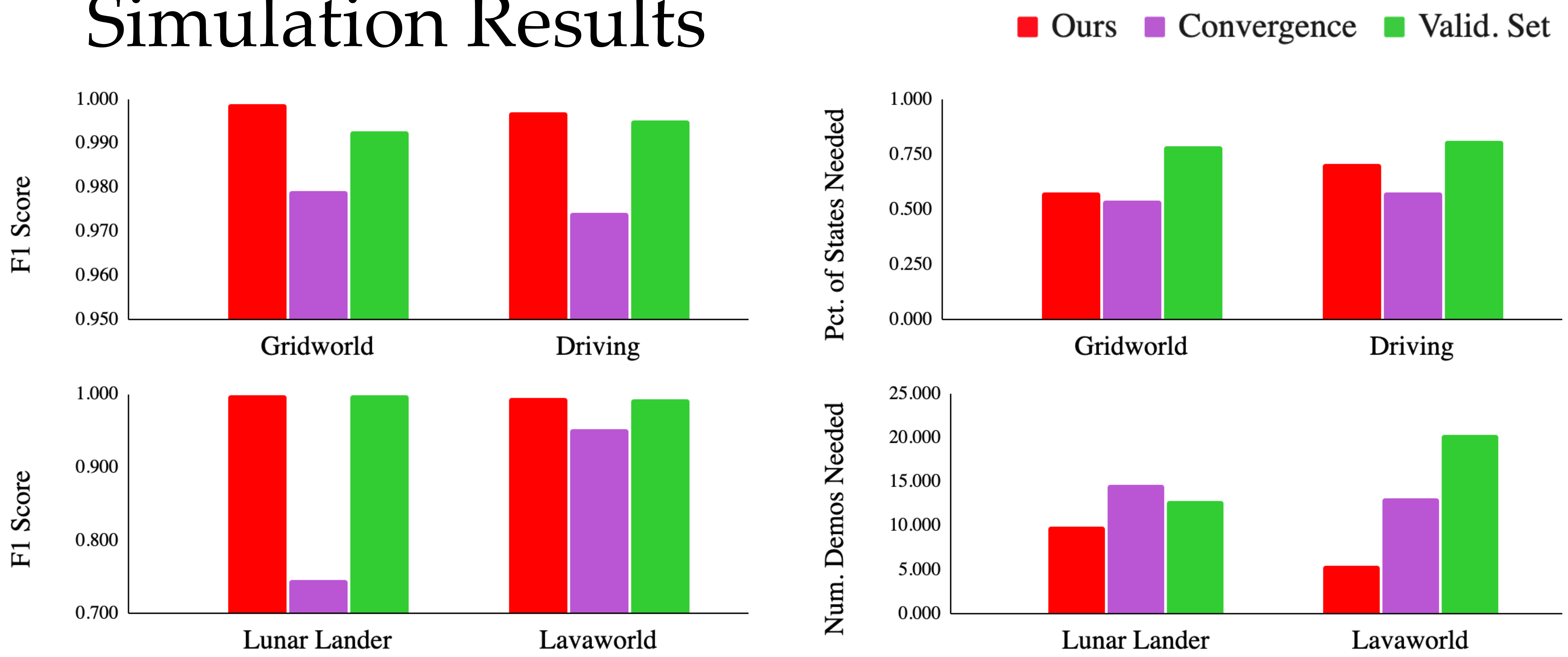


Lunar lander



Lavaworld

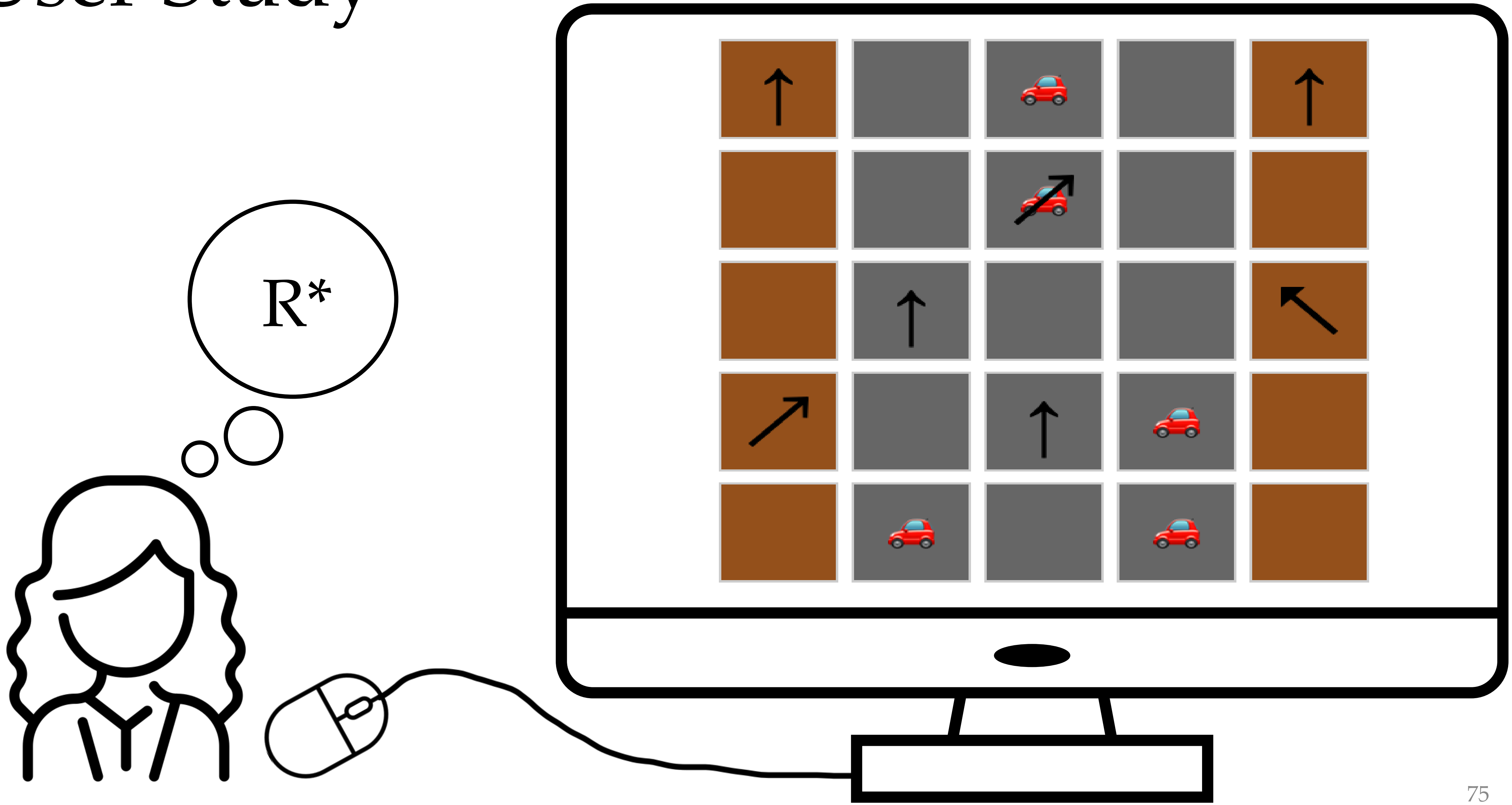
Simulation Results



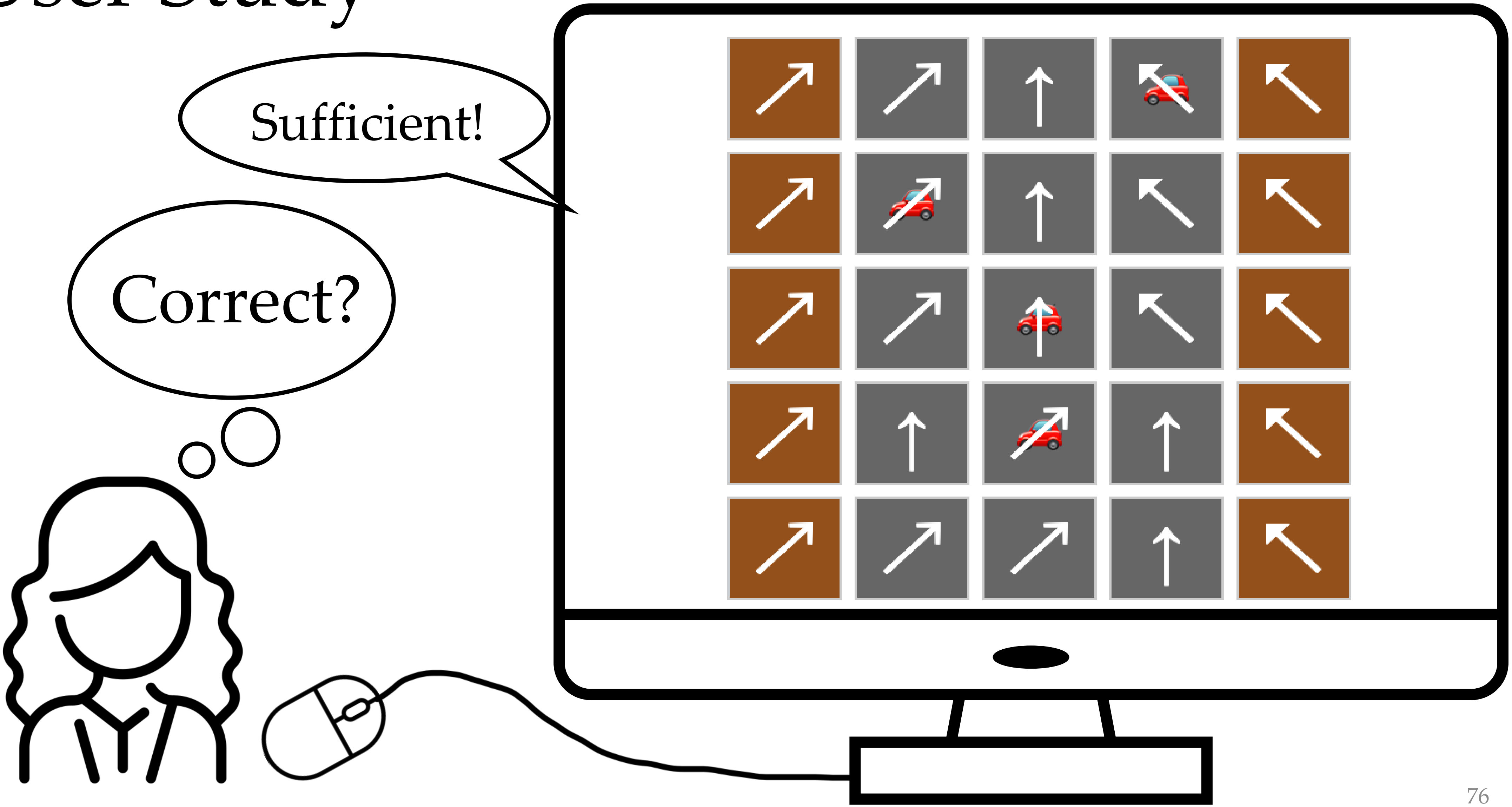
on average, 5% more accurate than baselines

on average, 25% more sample efficient than baselines

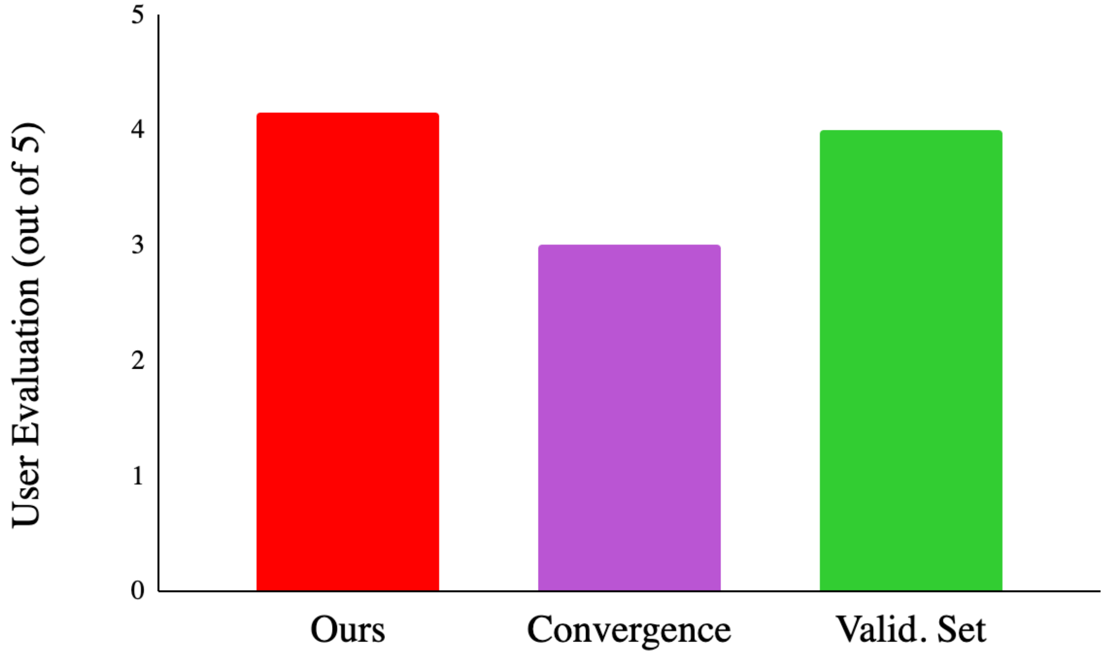
User Study

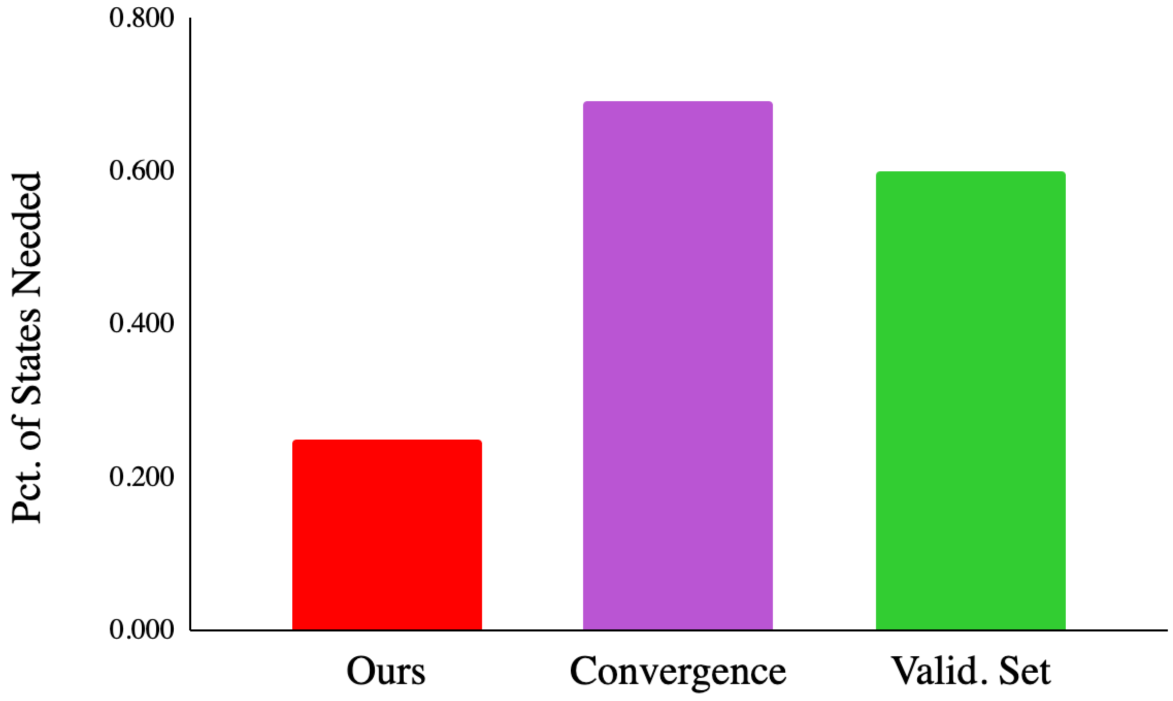


User Study

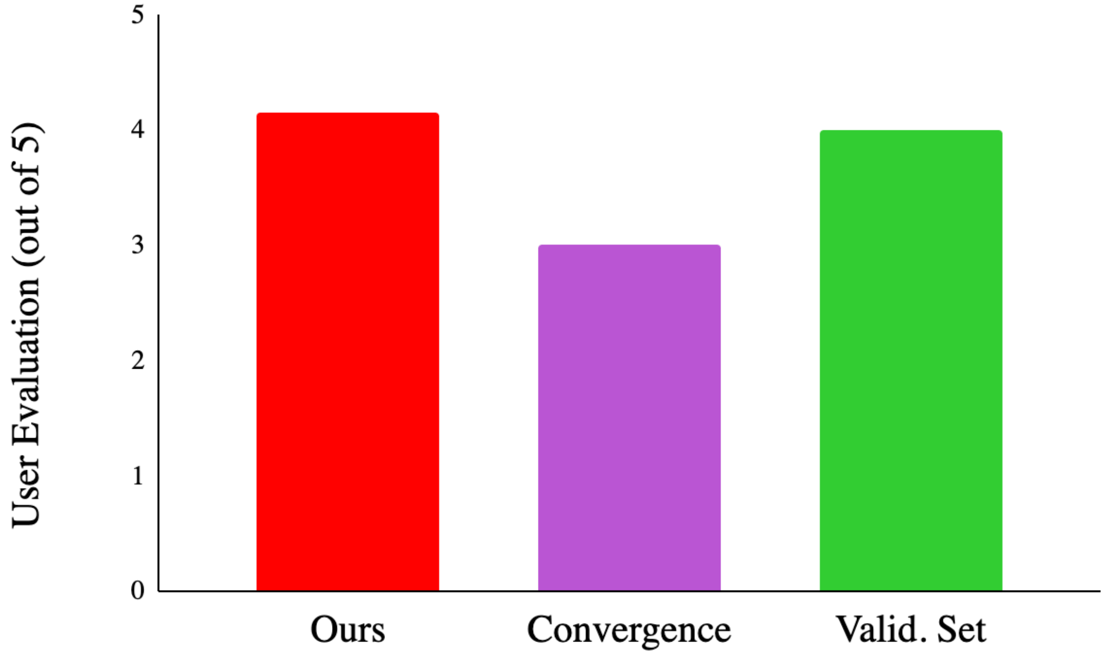


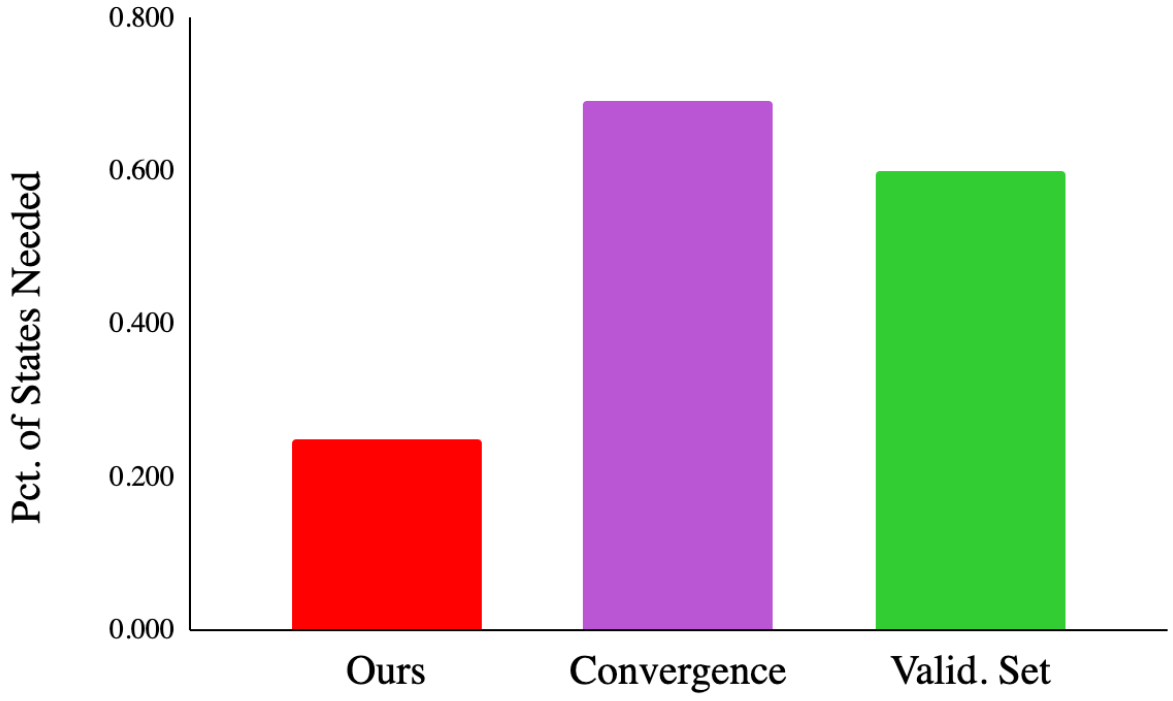
User Study Results





User Study Results





robust against suboptimal demonstrations: ~12%

Future Work

$$P(R|D) \propto P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}}$$

$$(s,a) \in \mathcal{D}$$

Bayesian IRL

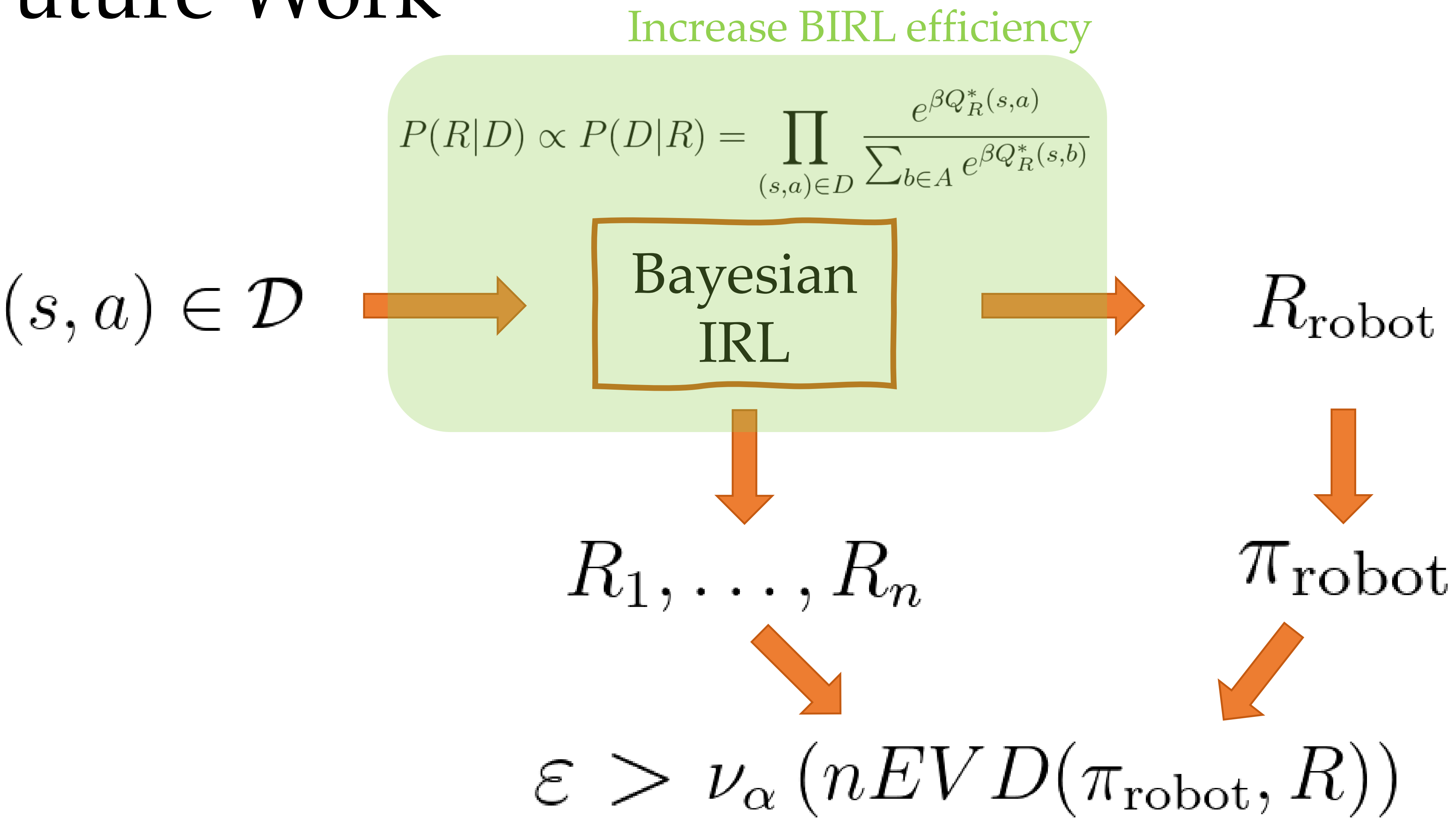
$$R_{\text{robot}}$$

$$R_1, \ldots, R_n$$

$$\pi_{\text{robot}}$$

$$\varepsilon > \nu_\alpha(nEVD(\pi_{\text{robot}}, R))$$

Future Work


Increase BIRL efficiency
$P(R|D) \propto P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}}$
$(s,a) \in \mathcal{D}$
Bayesian IRL
R_{robot}
$R_1, \ldots, R_n$
π_{robot}
$\varepsilon > \nu_\alpha(nEVD(\pi_{\text{robot}}, R))$

Future Work

Increase BIRL efficiency

$$P(R|D) \propto P(D|R) = \prod_{(s,a)\in D} \frac{e^{\beta Q^*_R(s,a)}}{\sum_{b\in A} e^{\beta Q^*_R(s,b)}}$$

$(s, a) \in \mathcal{D}$

Calibrate to demonstrator's optimality

Bayesian IRL

R_{robot}

$R_1, \ldots, R_n$

π_{robot}

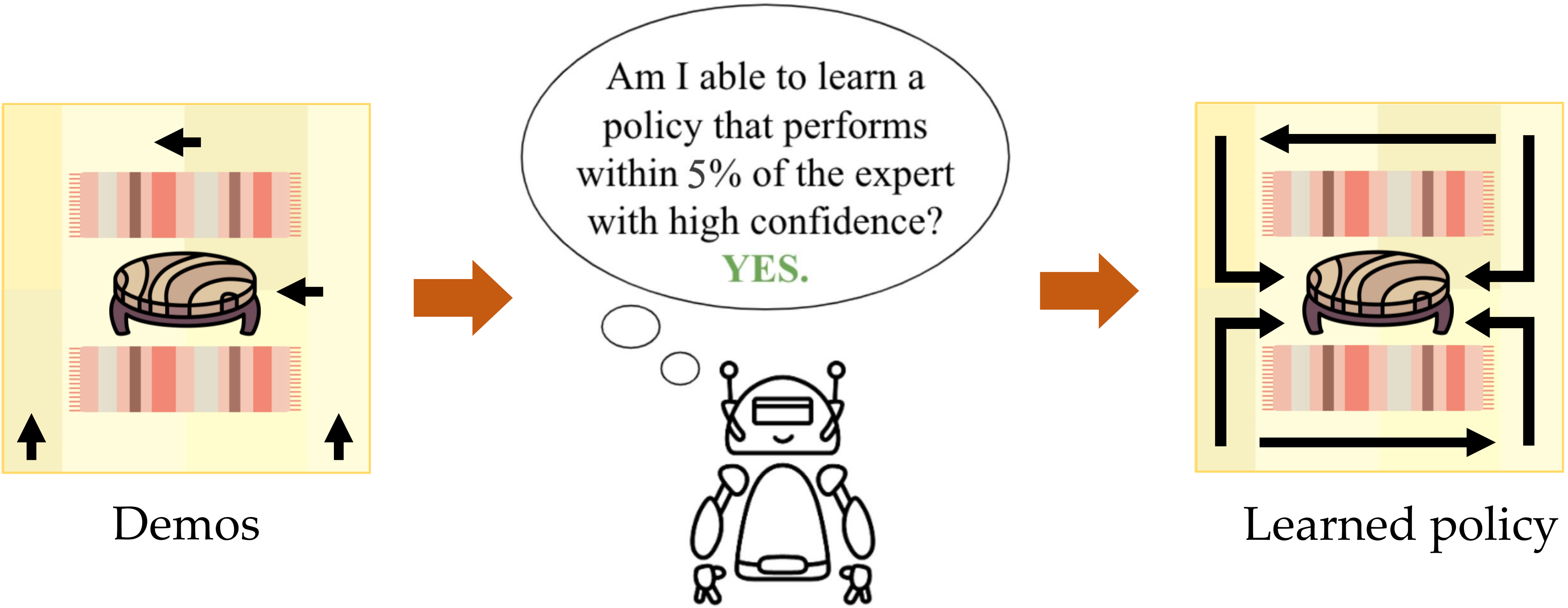
$$\varepsilon > \nu_\alpha(nEVD(\pi_{\text{robot}}, R))$$

BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Center for Human-Compatible Artificial Intelligence

ROBOTICS CENTER
THE UNIVERSITY OF UTAH

Autonomous Assessment of Demonstration Sufficiency via Bayesian Inverse Reinforcement Learning

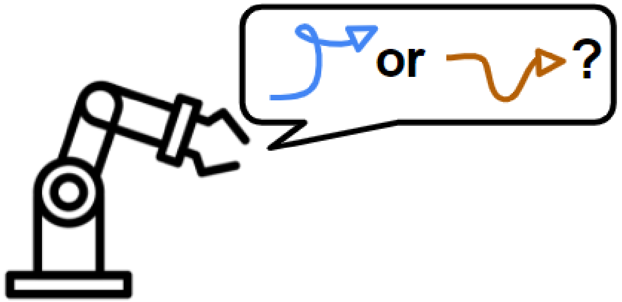


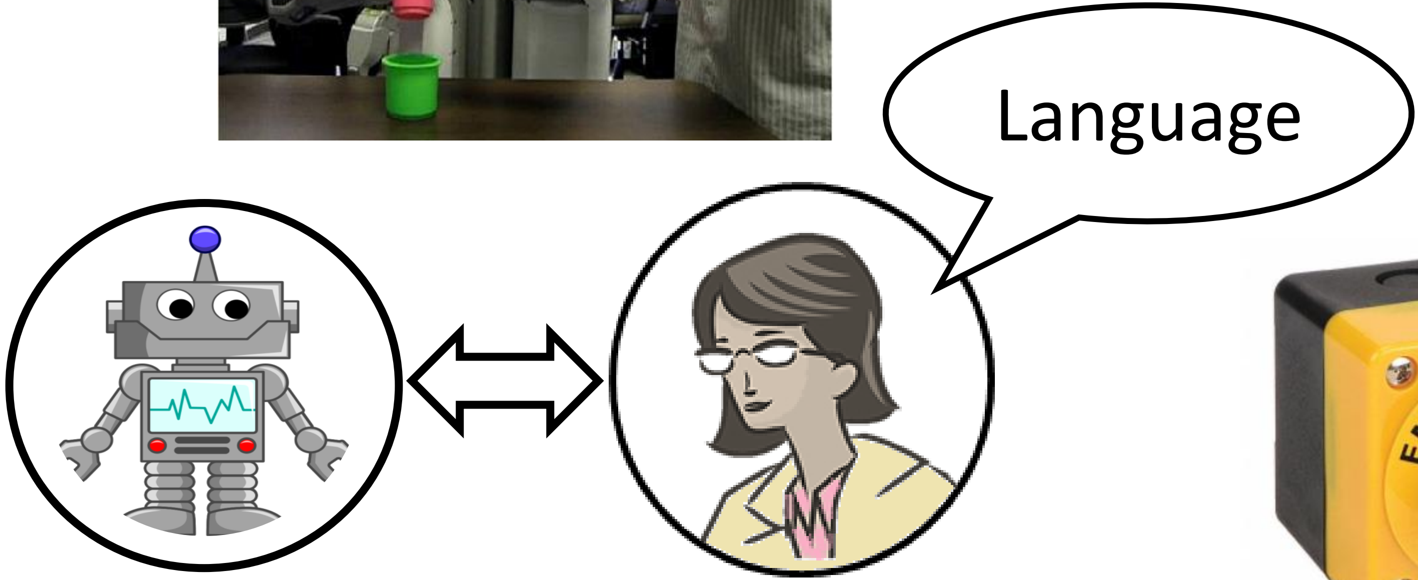
What if I want to learn rewards from more than demonstrations?

Reward-rational (implicit) choice: A unifying formalism for reward learning

Hong Jun Jeon[*1], **Smitha Milli**[*2], **Anca Dragan**[2]
hjjeon@stanford.edu, smilli@berkeley.edu, anca@berkeley.edu
[*]Equal contribution,
[1]Stanford University,
[2]University of California, Berkeley

How do we learn from diverse types of feedback of unknown quality?





Unifying Human Feedback Types

Boltzmann Rational Choice Model

$$C = \{c_1, \ldots . c_n\} \qquad r: C \to \mathbb{R}$$

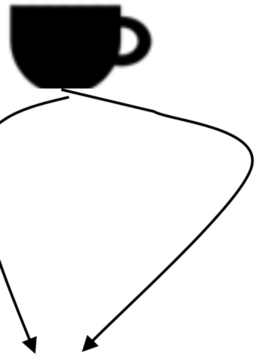
$$\mathbb{P}(c_i|r) = \frac{\exp(\boldsymbol{\beta} r(c_i))}{\sum_C \exp(\boldsymbol{\beta} r(c'))}$$

$\boldsymbol{\beta} \to \mathbf{0}$ random (non-rational) choices

$\boldsymbol{\beta} \to \infty$ deterministic (perfectly-rational) choices

Jeon et al. "Reward-rational (implicit) choice: A unifying formalism for reward learning." *NeurIPS* 2020.

Trajectory Comparisons (Pairwise Prefs)



τ_2 τ_1

$$C = \{\tau_1, \tau_2\}$$

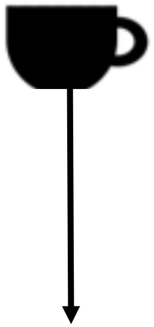
$$\mathbb{P}(\tau_i|r) = \frac{\exp(\beta r(\tau_i))}{\exp(\beta r(\tau_i)) + \exp(\beta r(\tau_i))}$$

Boltzmann Rational Choice Model

$$C = \{c_1, \dots. c_n\} \qquad r: C \to \mathbb{R}$$

$$\mathbb{P}(c_i|r) = \frac{\exp(\beta r(c_i))}{\sum_C \exp(\beta r(c'))}$$

Demonstrations



$C = \{\text{All Possible Trajectories }\}$

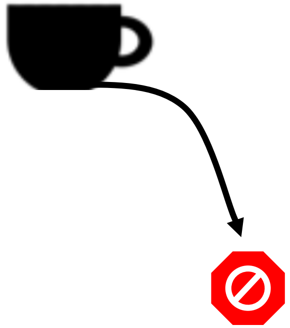
$$\mathbb{P}(\tau|r) = \frac{\exp(\beta r(\tau))}{\sum_{\mathrm{T}} \exp(\,\beta r(\tau'))}$$

Boltzmann Rational Choice Model

$C = \{c_1, \dots . \, c_n\} \qquad r: C \to \mathbb{R}$

$$\mathbb{P}(c_i|r) = \frac{\exp(\beta r(c_i))}{\sum_{C} \exp(\,\beta r(c'))}$$

E-Stops



$$C = \{\tau_{:1}, \dots, \tau_{:T}\}$$

$$\mathbb{P}(\tau_{:i} | r) = \frac{\exp(\beta r(\tau_{:i}))}{\sum_1^T \exp \beta r(\tau_{:j}))}$$

Boltzmann Rational Choice Model

$$C = \{c_1, \dots . c_n\} \qquad r: C \to \mathbb{R}$$

$$\mathbb{P}(c_i | r) = \frac{\exp(\beta r(c_i))}{\sum_C \exp(\beta r(c'))}$$

Reward Learning from Human Feedback

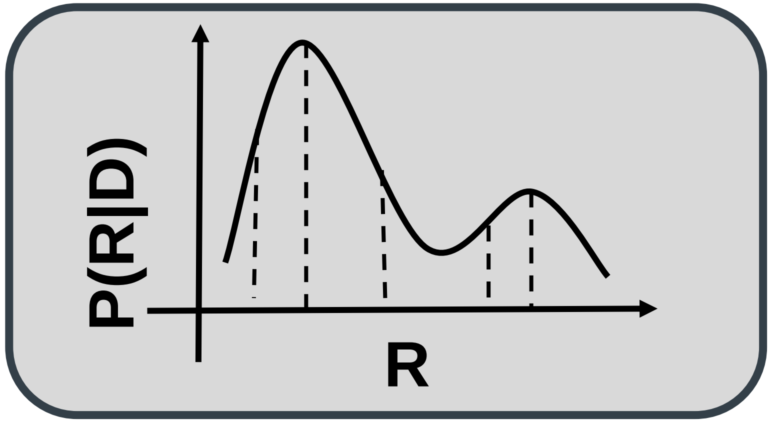
- Assume demonstrator is Boltzmann rational
  - Demonstrator follows a softmax policy with inverse temperature β

$$C = \{c_1, \ldots . c_n\} \qquad r: C \to \mathbb{R}$$

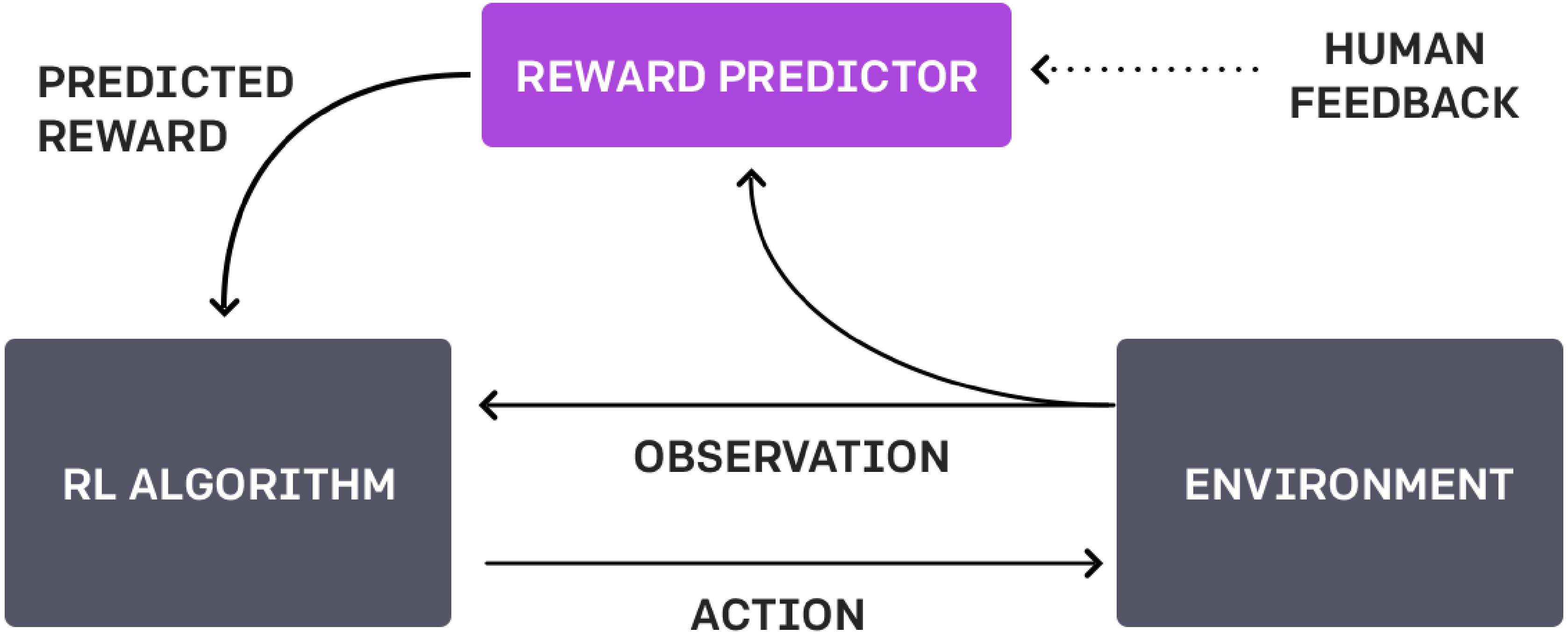
$$\mathbb{P}(c_i|r) = \frac{\exp(\boldsymbol{\beta} r(c_i))}{\sum_C \exp(\boldsymbol{\beta} r(c'))}$$

  - Perform Bayesian inference (MCMC) to sample from posterior distribution

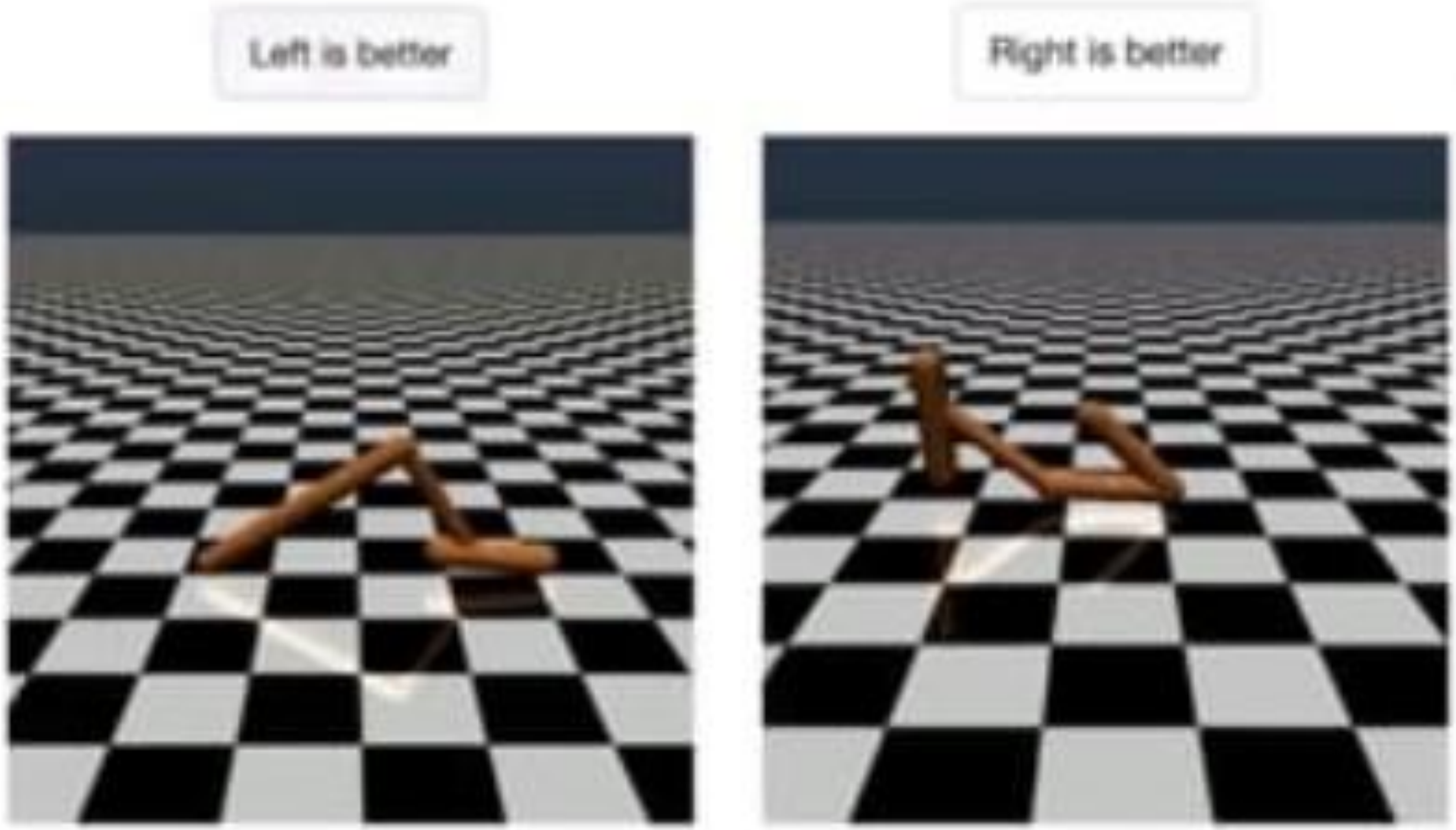
$$P(R|D) \propto P(D|R)P(R)$$



RL from Human Feedback (RLHF)



RL from Human Preferences



https://arxiv.org/abs/1706.03741

Why would you want to learn a reward from ranked examples?

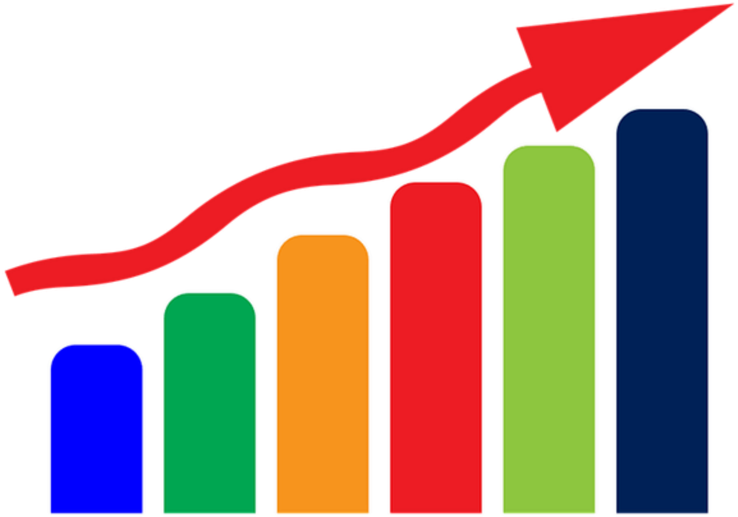
Inverse Reinforcement Learning

Prior approaches …

1. Typically couldn't do much better than the demonstrator.

2. Were hard to scale to complex problems.

Pre-Ranked Demonstrations



Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Inverse Reinforcement Learning

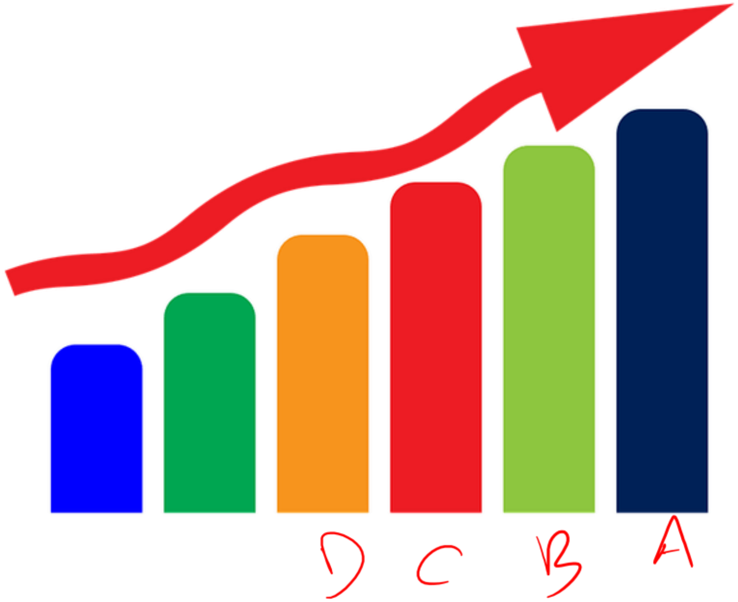
Prior approaches …

1. ~~Typically couldn't do much better than the demonstrator.~~

Find a reward function that explains the ranking, allowing for extrapolation.

2. Were hard to scale to complex problems.

Pre-Ranked Demonstrations



Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Inverse Reinforcement Learning

Prior approaches …

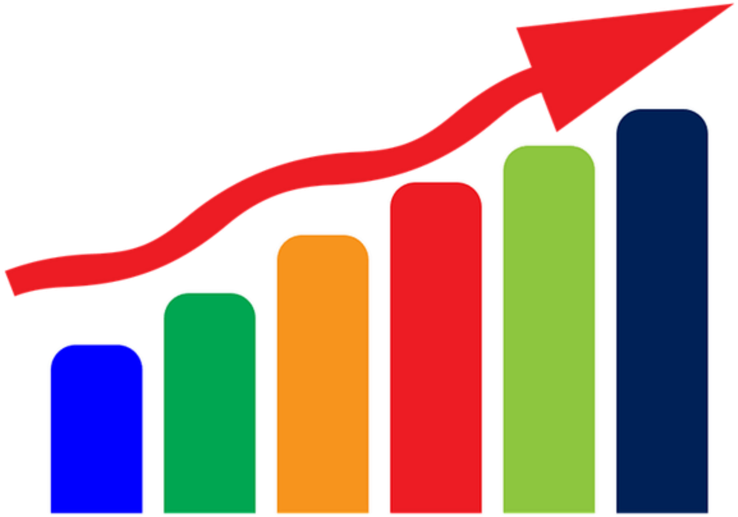
1. ~~Typically couldn't do much better than the demonstrator.~~

Find a reward function that explains the ranking, allowing for extrapolation.

2. ~~Were hard to scale to complex problems.~~

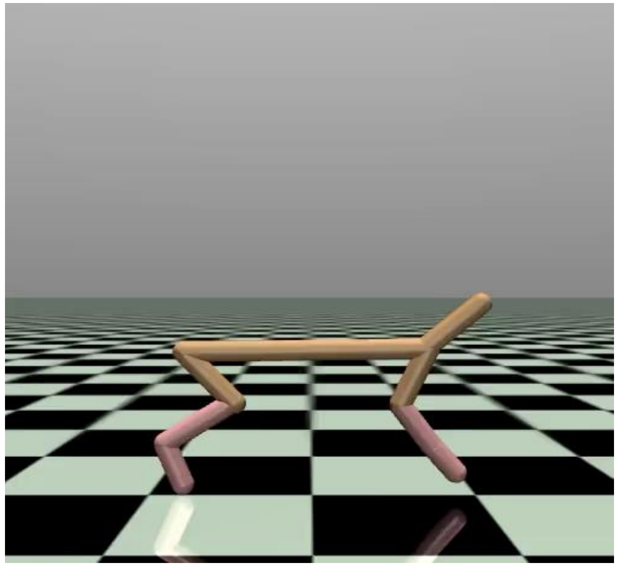
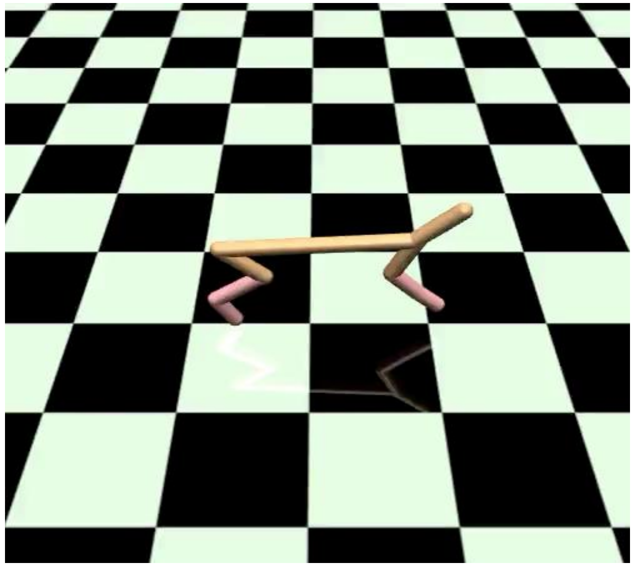
Reward learning becomes a supervised learning problem.

Pre-Ranked Demonstrations



Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Trajectory-ranked Reward Extrapolation (T-REX)

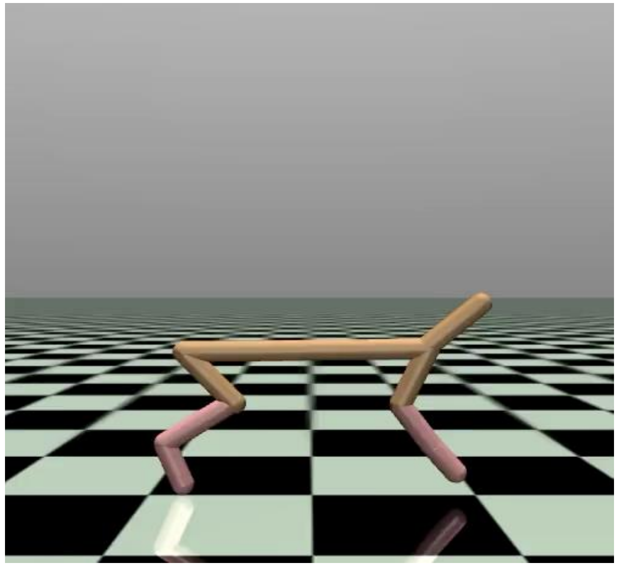
 $\prec \cdots \prec$ 

Reward Function

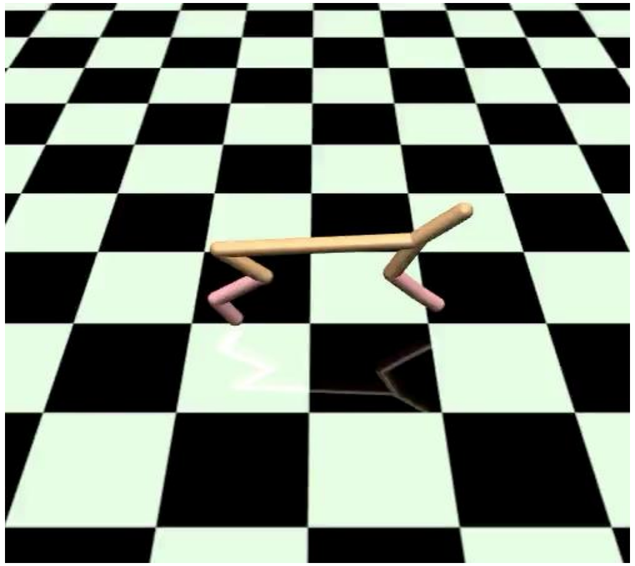
Pre-ranked demonstrations

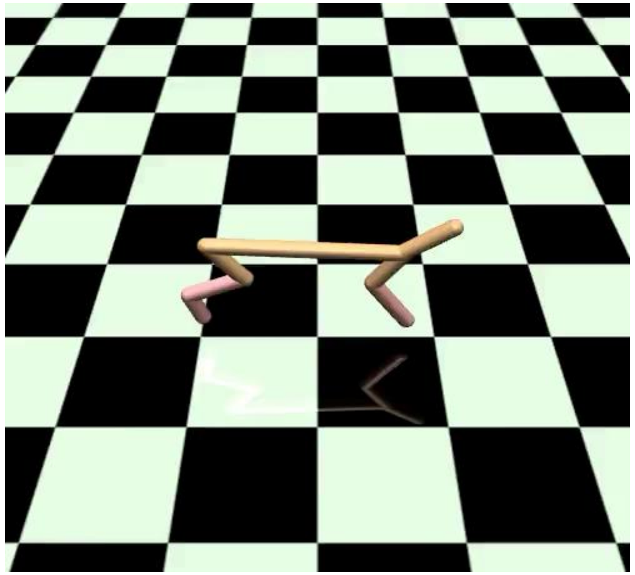
Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Trajectory-ranked Reward Extrapolation (T-REX)



$\prec \cdots \prec$





Pre-ranked demonstrations

T-REX Policy

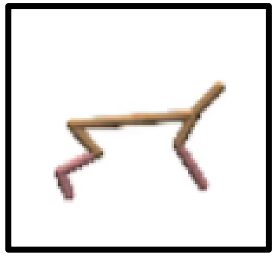
Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Reward Function

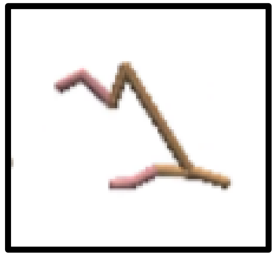
$$R_\theta : S \rightarrow \mathbb{R}$$

Examples of S:

Current Robot Joint
Angles and Velocities



$\rightarrow 0.5$



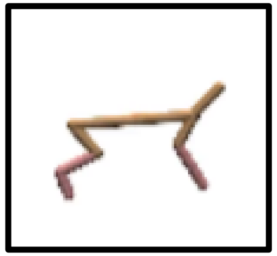
$\rightarrow -0.7$

Reward Function

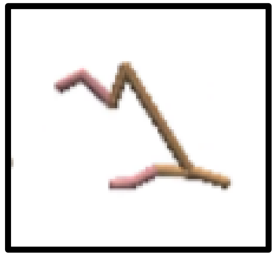
$$R_\theta : S \to \mathbb{R}$$

Examples of S:

Current Robot Joint Angles and Velocities

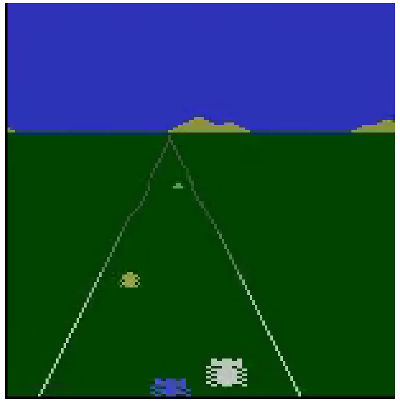


$\to 0.5$

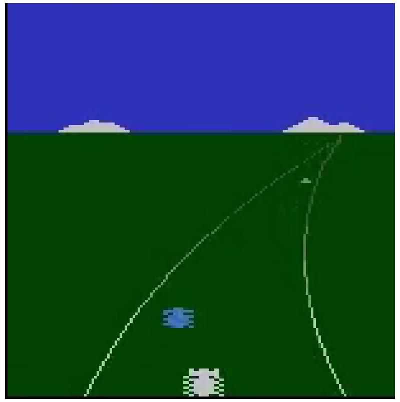


$\to -0.7$

Short Sequence of Images



$\to 0.9$



$\to -1.2$

Trajectory-ranked Reward Extrapolation (T-REX)

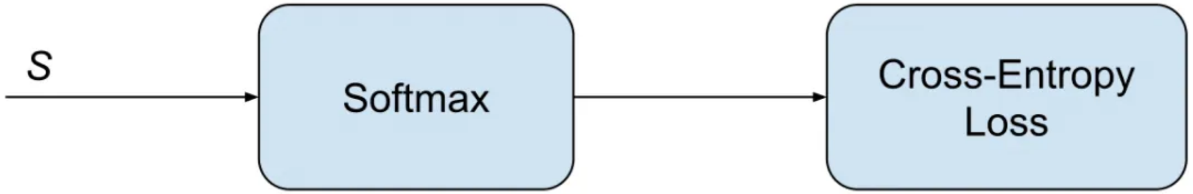
$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

$$\boxed{\sum_{s \in \tau_1} R_\theta(s)} < \boxed{\sum_{s \in \tau_2} R_\theta(s)}$$

Bradley-Terry pairwise ranking loss

$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_j} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)}$$

Trajecto… tion (T-REX…



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

$$\boxed{\sum_{s \in \tau_1} R_\theta(s)} < \boxed{\sum_{s \in \tau_2} R_\theta(s)}$$

Logits

Minimize cross-entropy loss

$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_j} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)}$$

Trajectory-ranked Reward Extrapolation (T-REX)

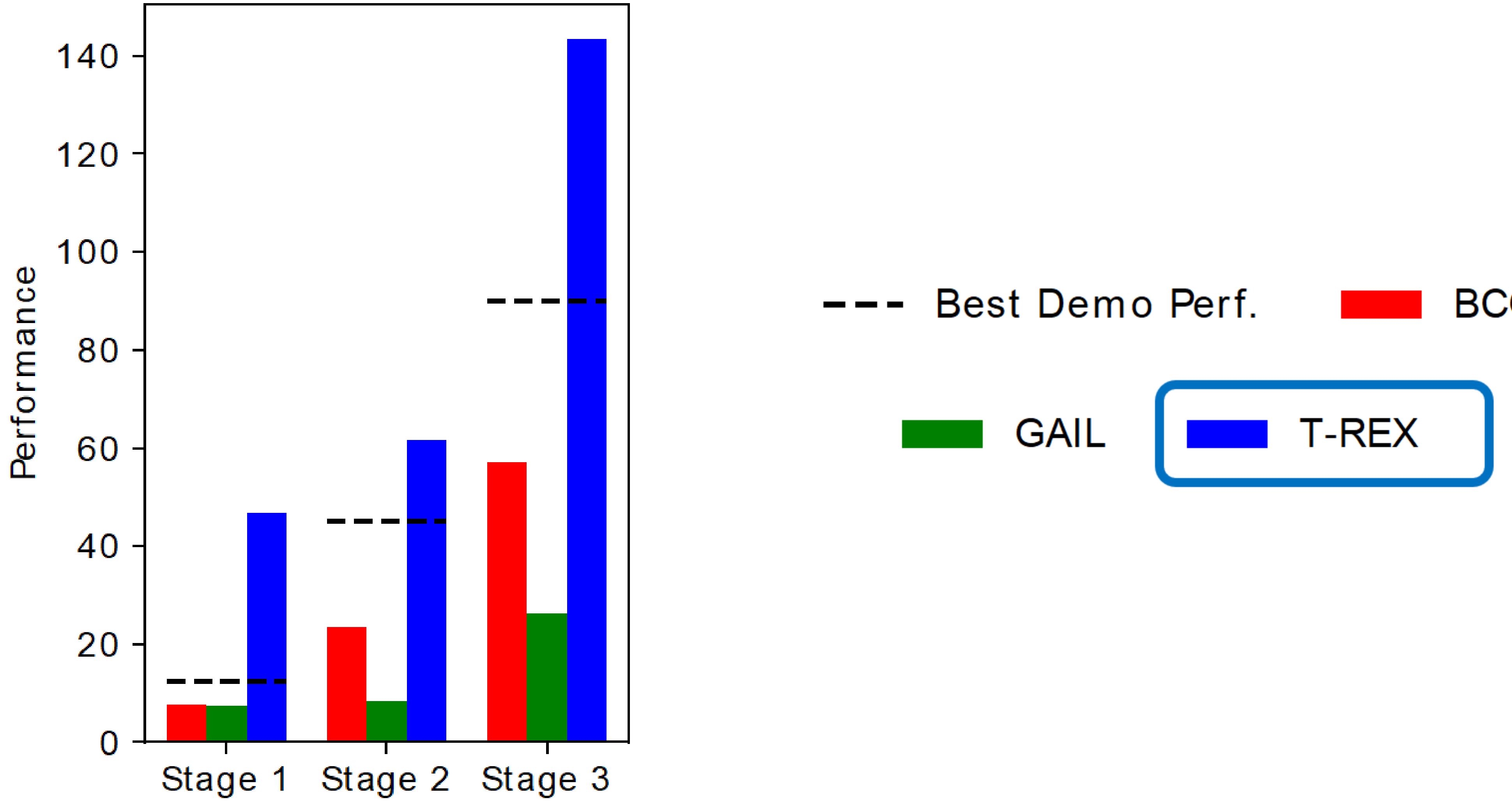
$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

Given pre-ranked demos, reward learning can be formulated as a standard supervised learning task.

Minimize cross-entropy loss

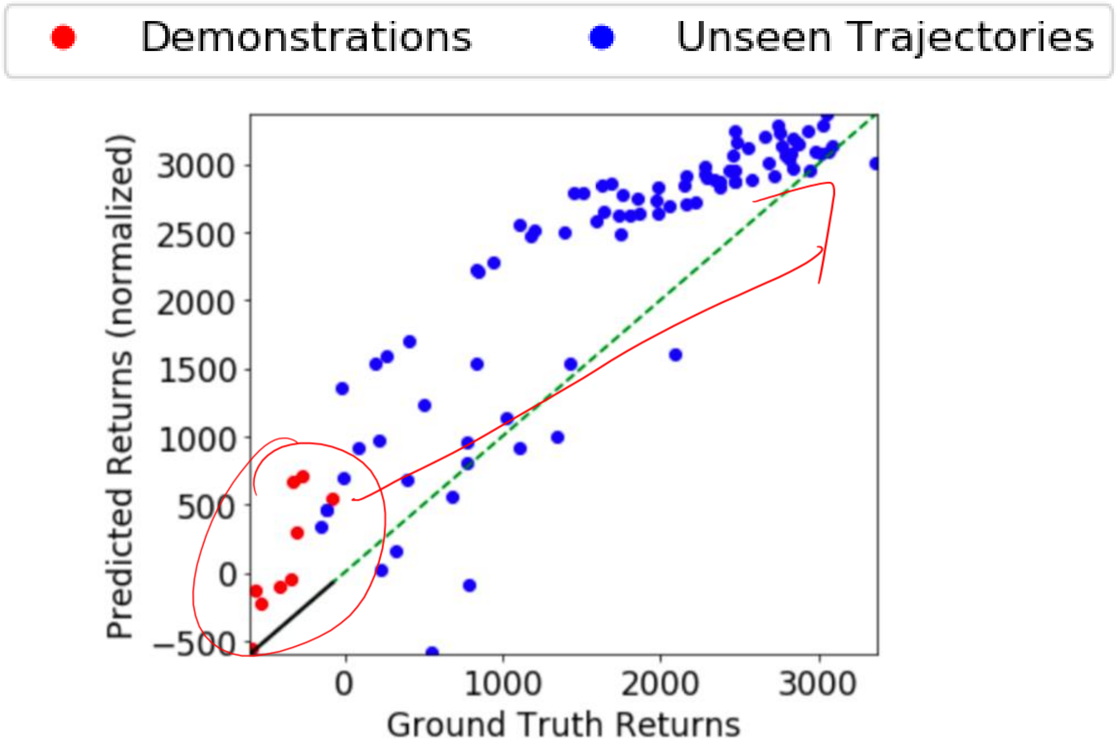
$$\mathcal{L}(\theta) = -\sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_j} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)}$$

T-REX Policy Performance



Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

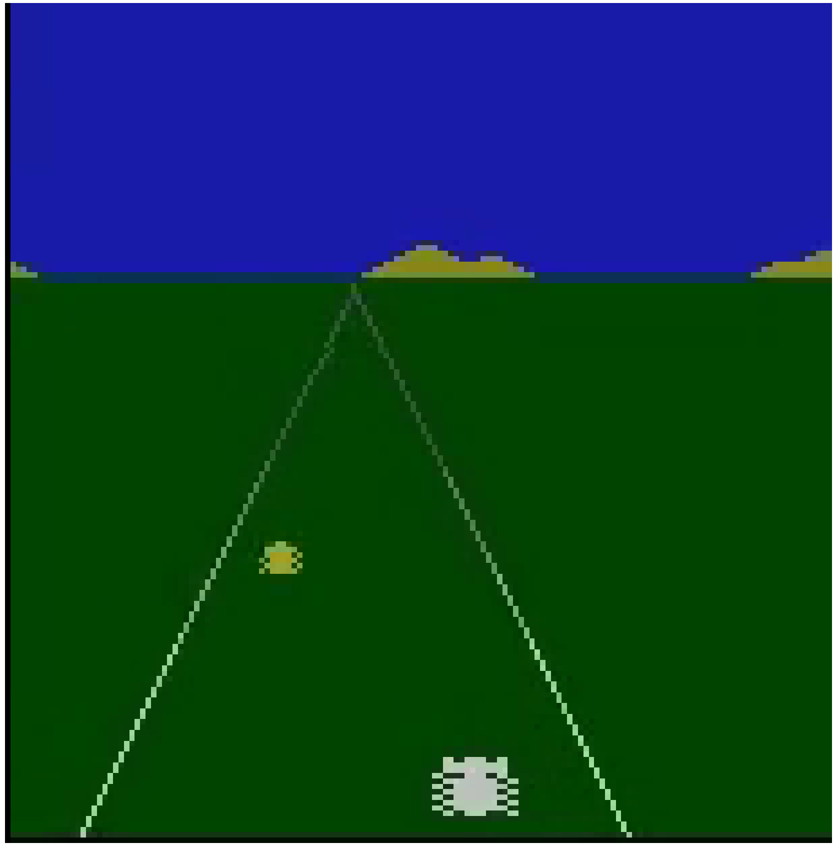
Reward Extrapolation



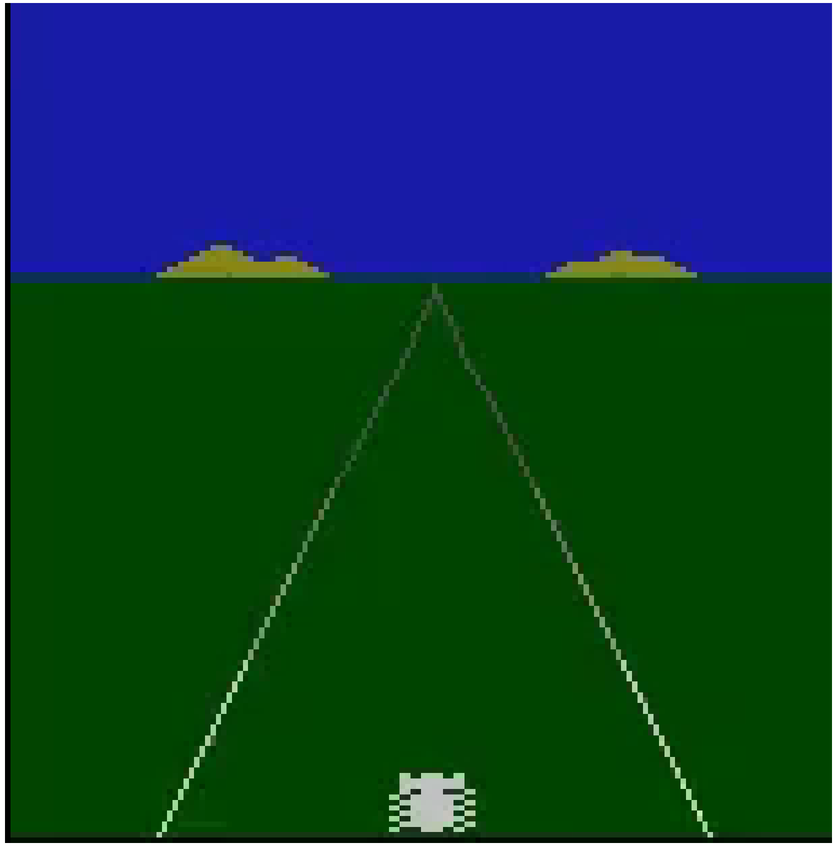
T-REX can extrapolate beyond the performance of the best demo

Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

“Autonomous Driving” in Atari



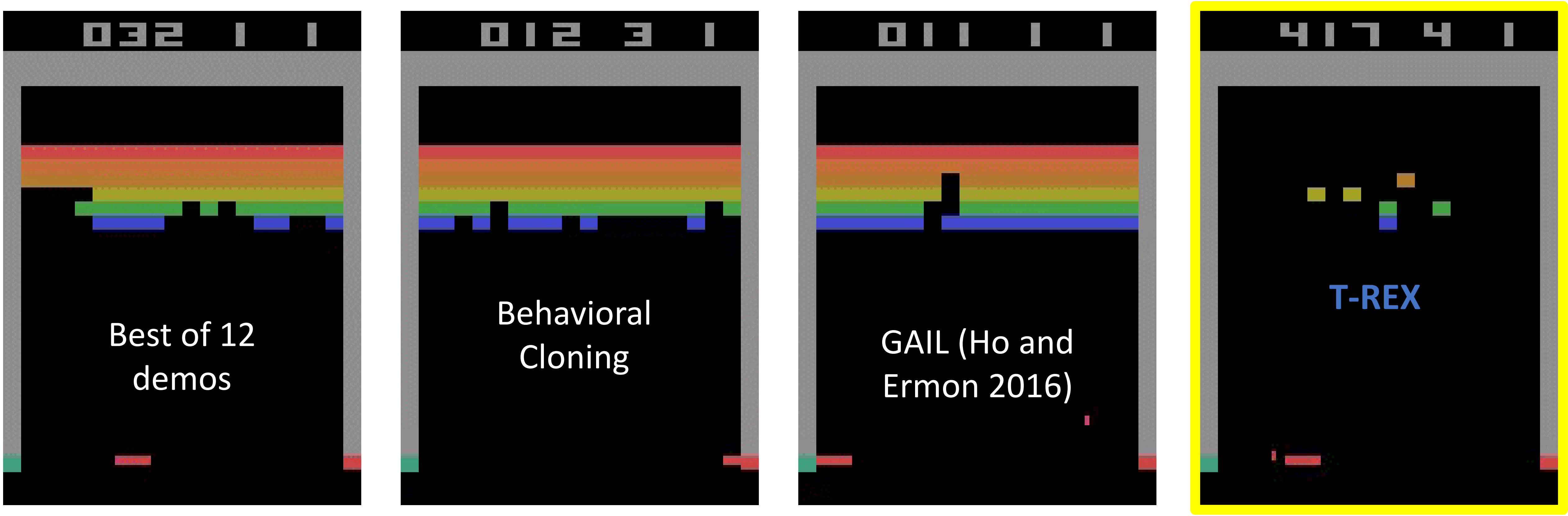
Best demo (Score = 84)



T-REX (Score = 520)

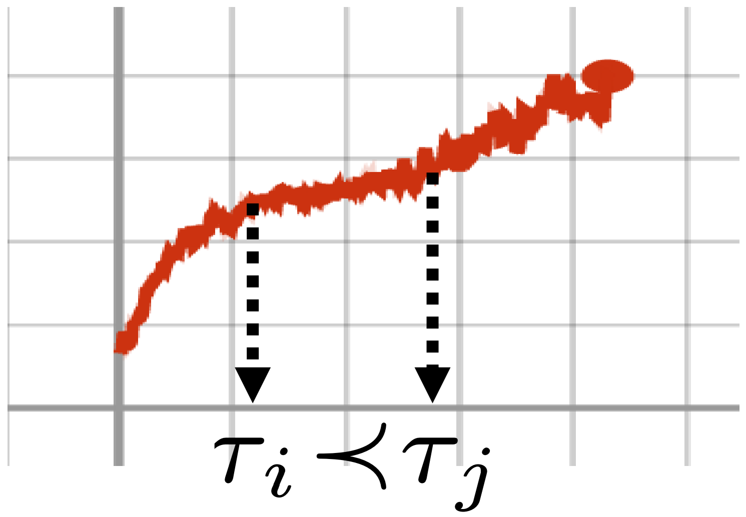
Uses only 12 ranked demonstrations

Atari Breakout

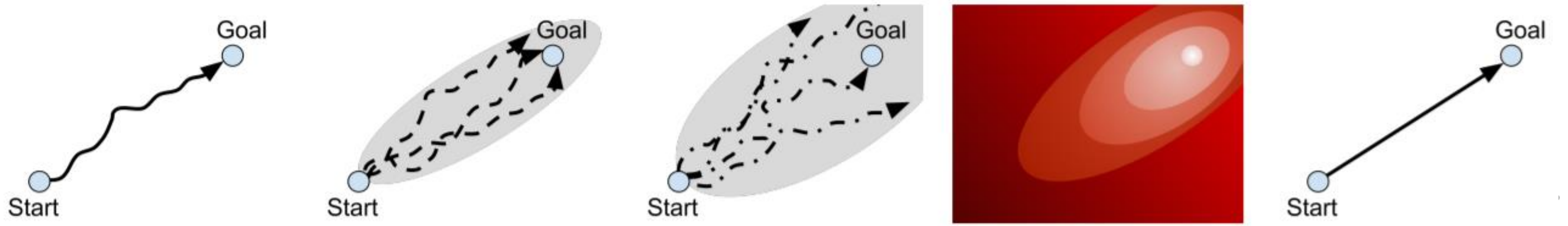


What if you don't have explicit preference labels?

Learning from a learner [ICML'19]

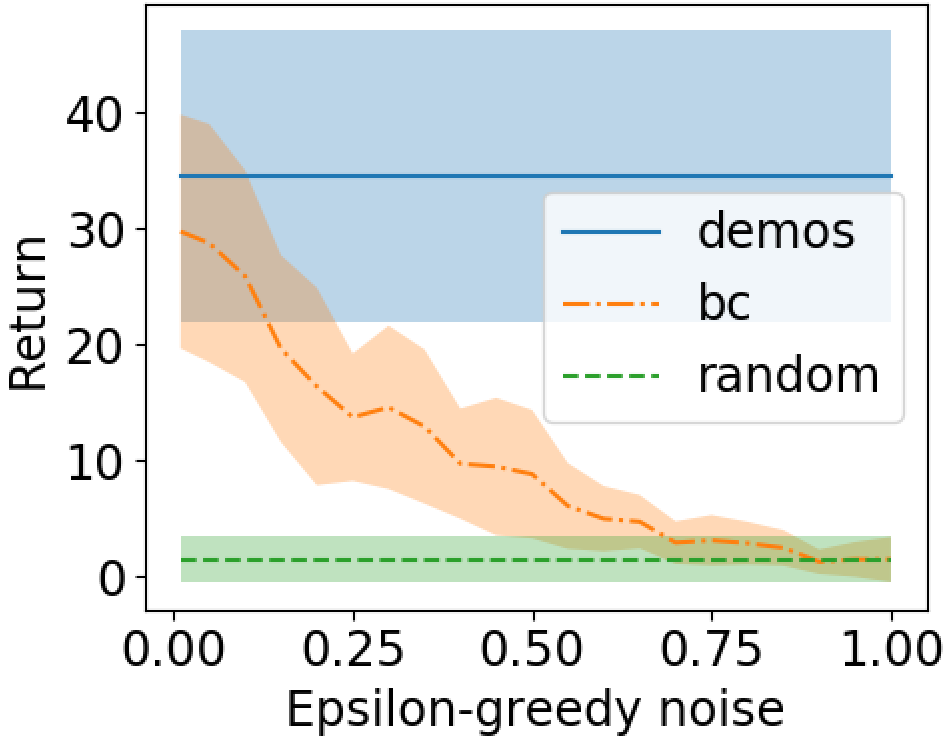


Automatic preference label generation [CoRL'20]



Automatic Rankings via Noise Injection

- Assumption: Demonstrator is significantly better than a purely random policy.
- Provides automatic rankings as noise increases.
- Generates a large diverse set of ranked demonstrations

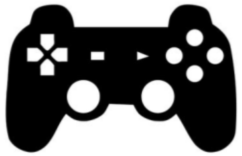


Brown et al. “Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations." CoRL 2019

Disturbance-based Reward Extrapolation (D-REX)

Behavioral Cloning

$s \to a$

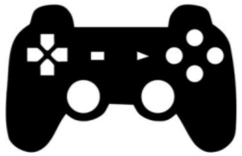


$$\pi_{\mathrm{BC}}$$

Brown et al. "Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations." CoRL 2019

Disturbance-based Reward Extrapolation (D-REX)

Behavioral Cloning → Automatic Rankings via Noise Injection



⇩

π_{BC}

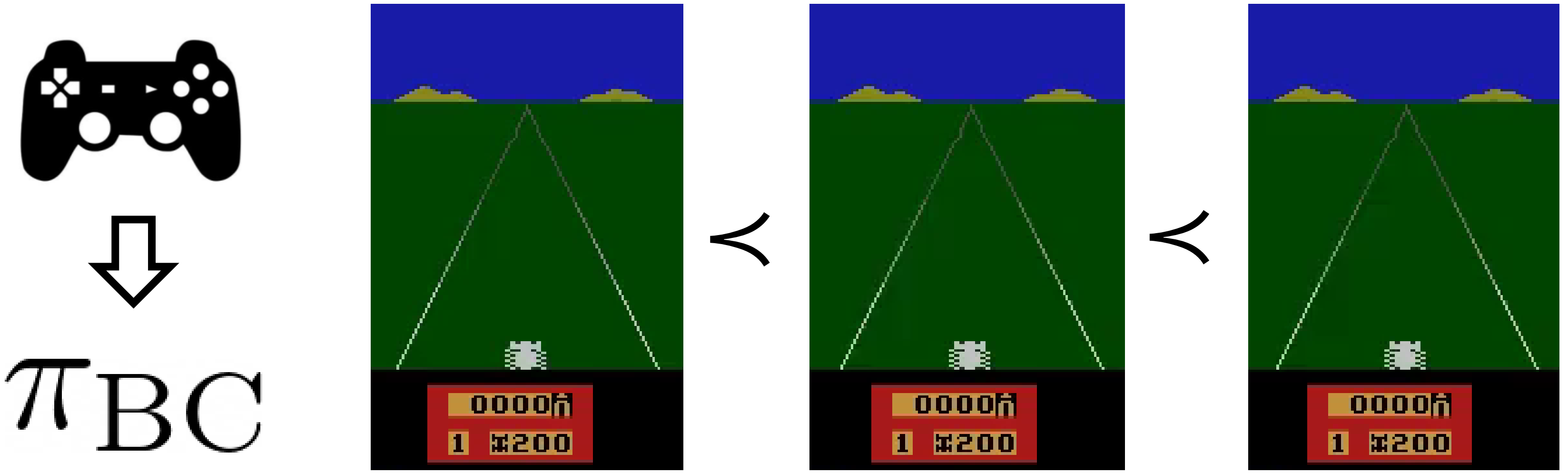
$\prec$

$\prec$

$\epsilon = 1.0$ $\epsilon = 0.2$ $\epsilon = 0.01$

Disturbance-based Reward Extrapolation (D-REX)

Behavioral Cloning → Automatic Rankings via Noise Injection → T-REX

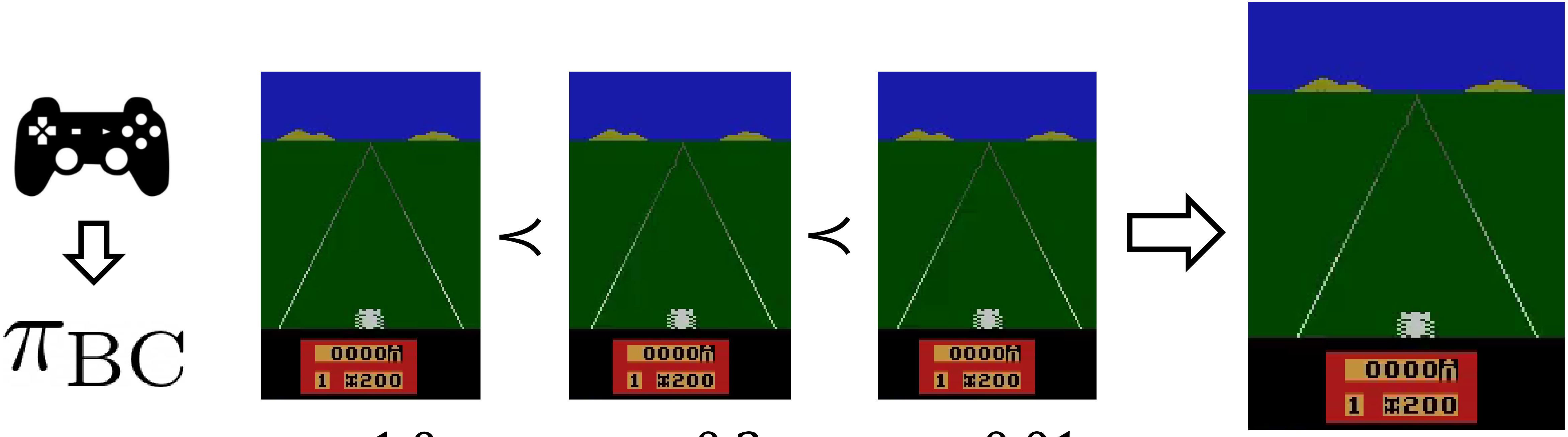


π_{BC}

$\epsilon = 1.0 \prec \epsilon = 0.2 \prec \epsilon = 0.01 \Rightarrow$ Reward Function **R(s)**

Disturbance-based Reward Extrapolation (D-REX)

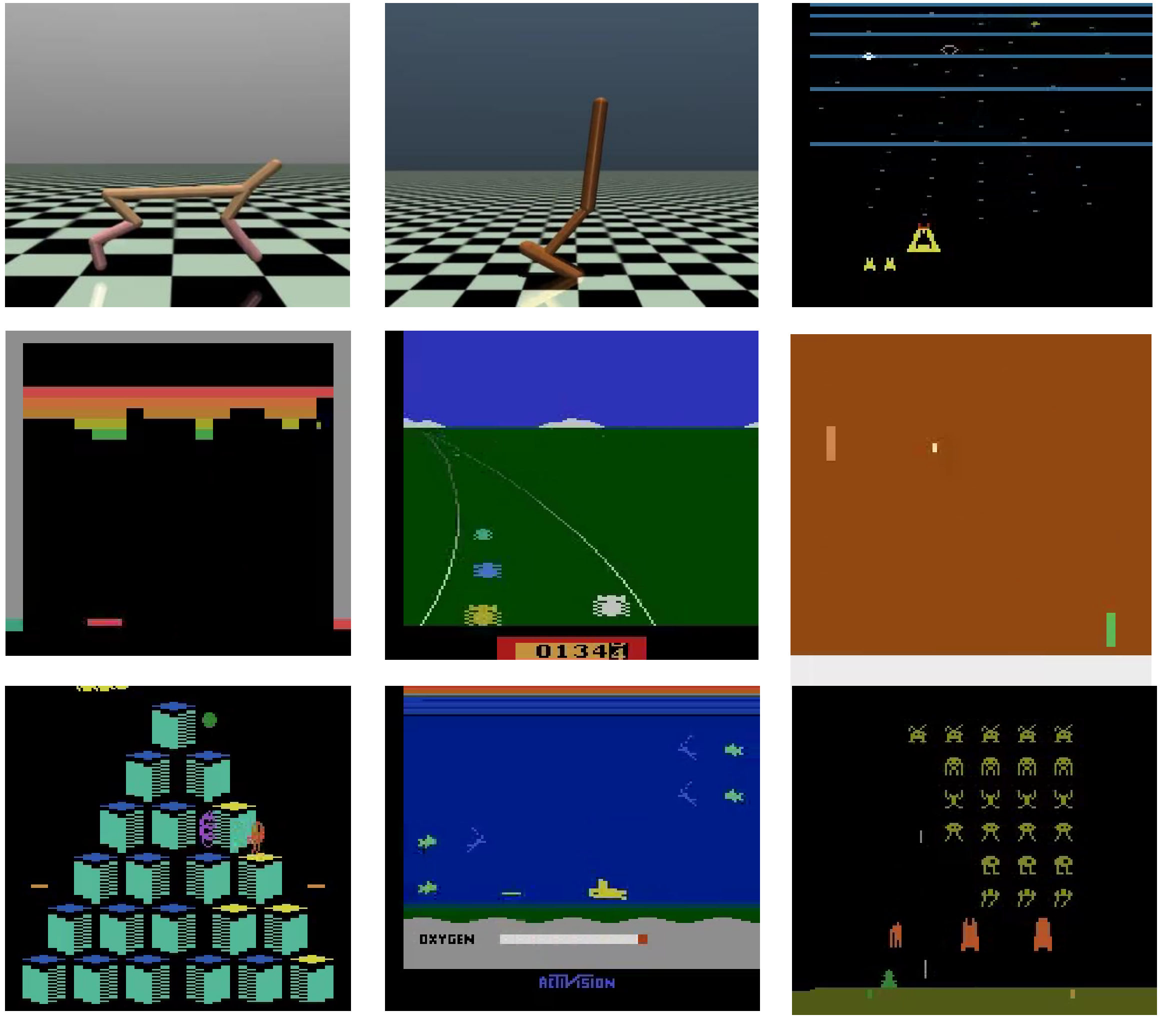
Behavioral Cloning → Automatic Rankings via Noise Injection → T-REX → Policy Optimization



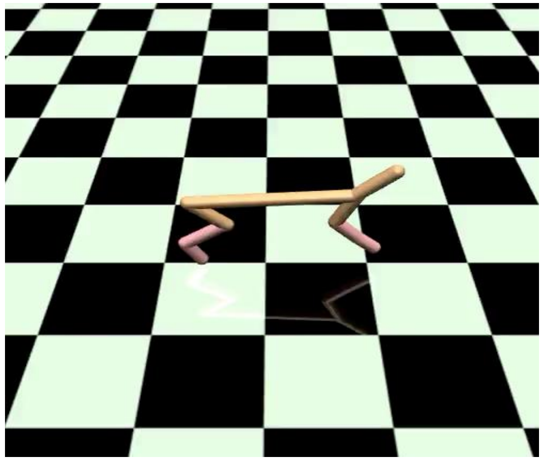
$\epsilon = 1.0$ $\prec$ $\epsilon = 0.2$ $\prec$ $\epsilon = 0.01$ $\Rightarrow$ D-REX Policy

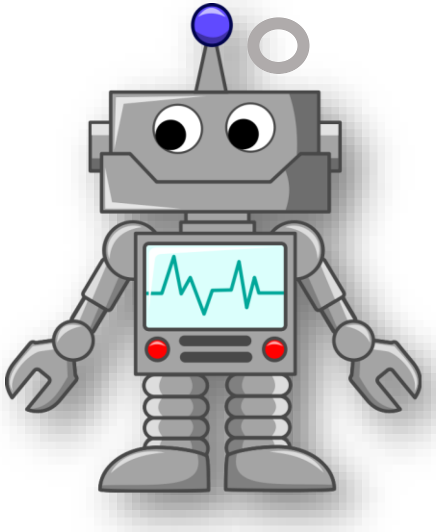
Experiments

D-REX consistently outperforms the best demonstration as well as outperforming BC and GAIL.



Brown et al. “Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations." CoRL 2019



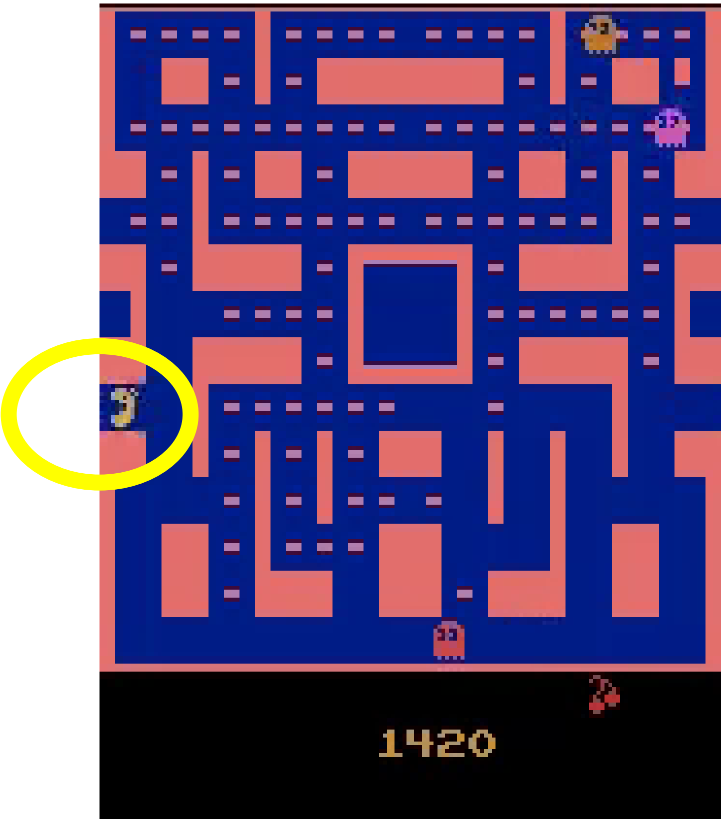


AI systems can **efficiently** infer human intent from **suboptimal demonstrations**.

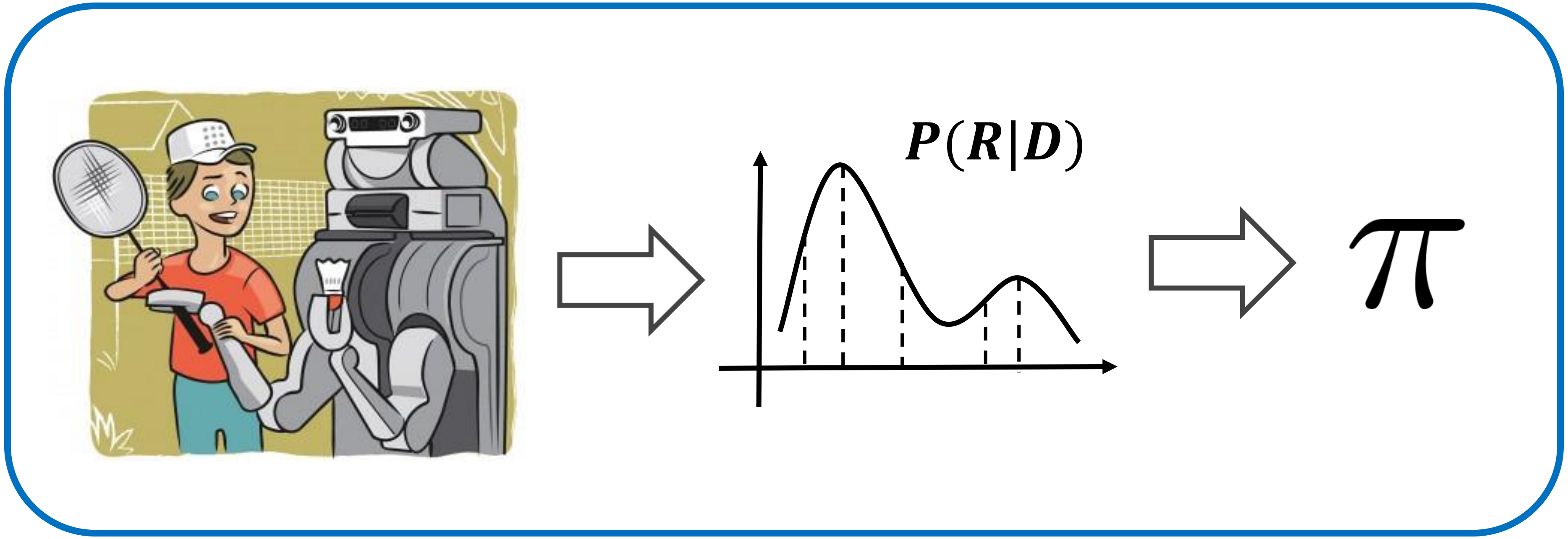
T-REX only learns a maximum likelihood estimate of the reward function.

$$\Rightarrow R \Rightarrow \pi$$

Reward Hacking



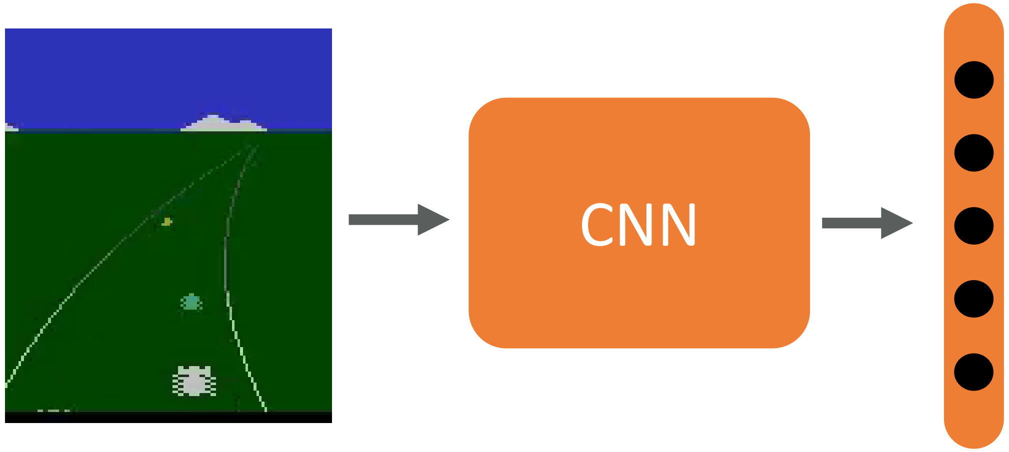
- Overfit to spurious correlations
- No consideration of alternative hypotheses



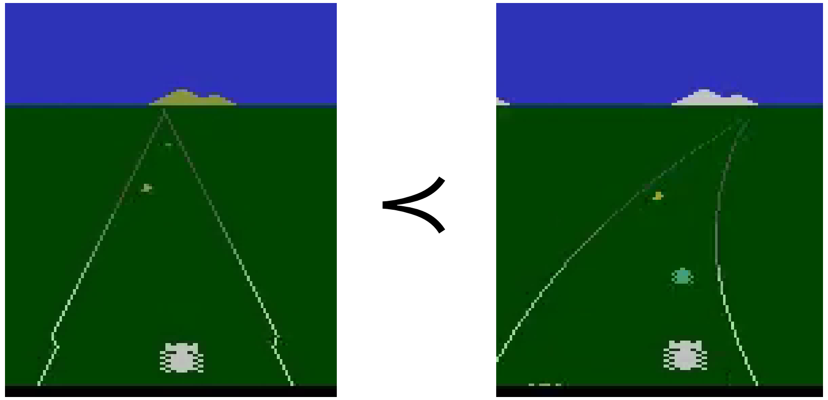
$P(R|D)$
π

Idea: Fast Bayesian Inference

Low-Dimensional Latent Space



Preference-Based Likelihood Function



Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2020.

Next time: LLMs and ChatGPT