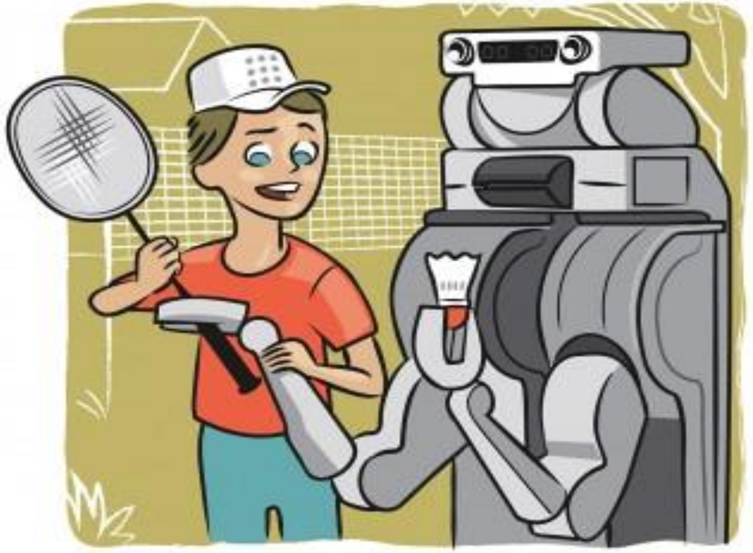
Inverse RL and Reward Learning from Preferences



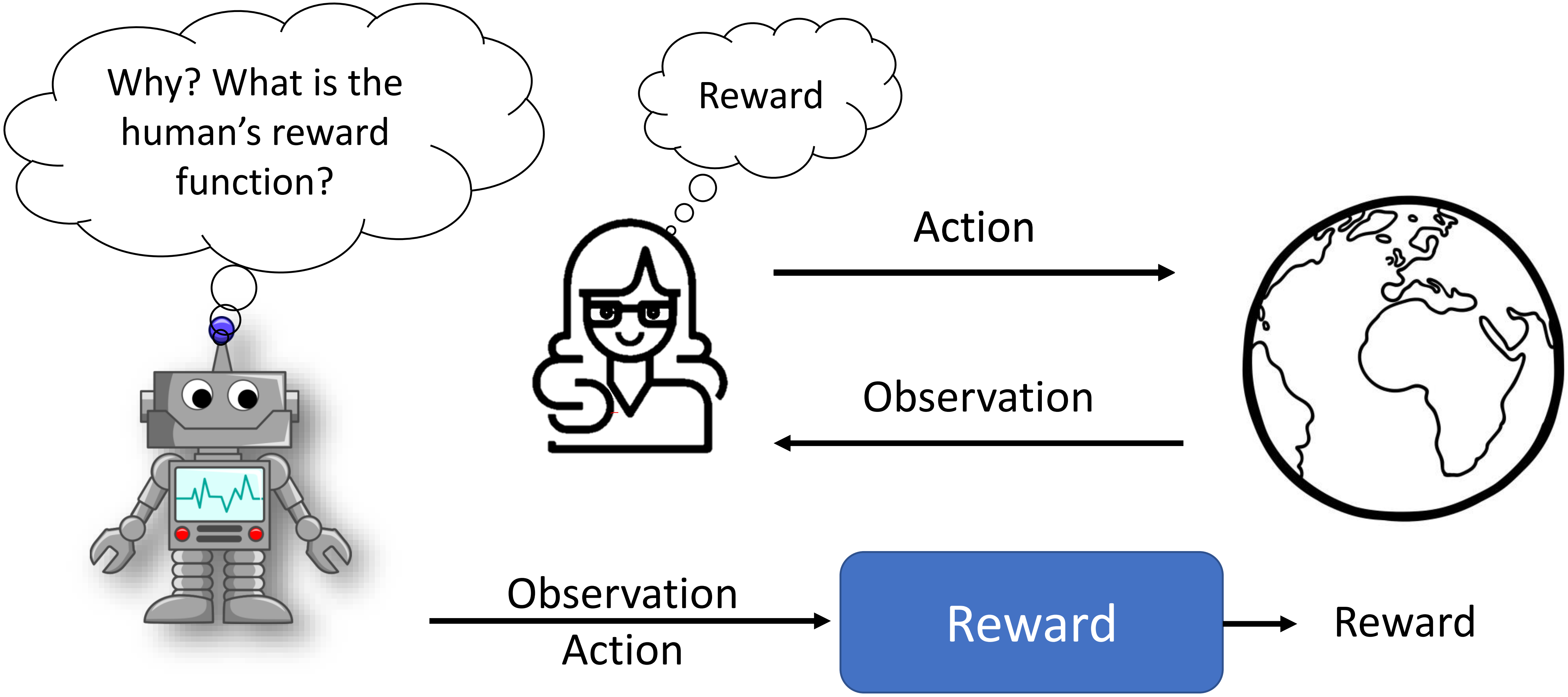
Instructor: Daniel Brown

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

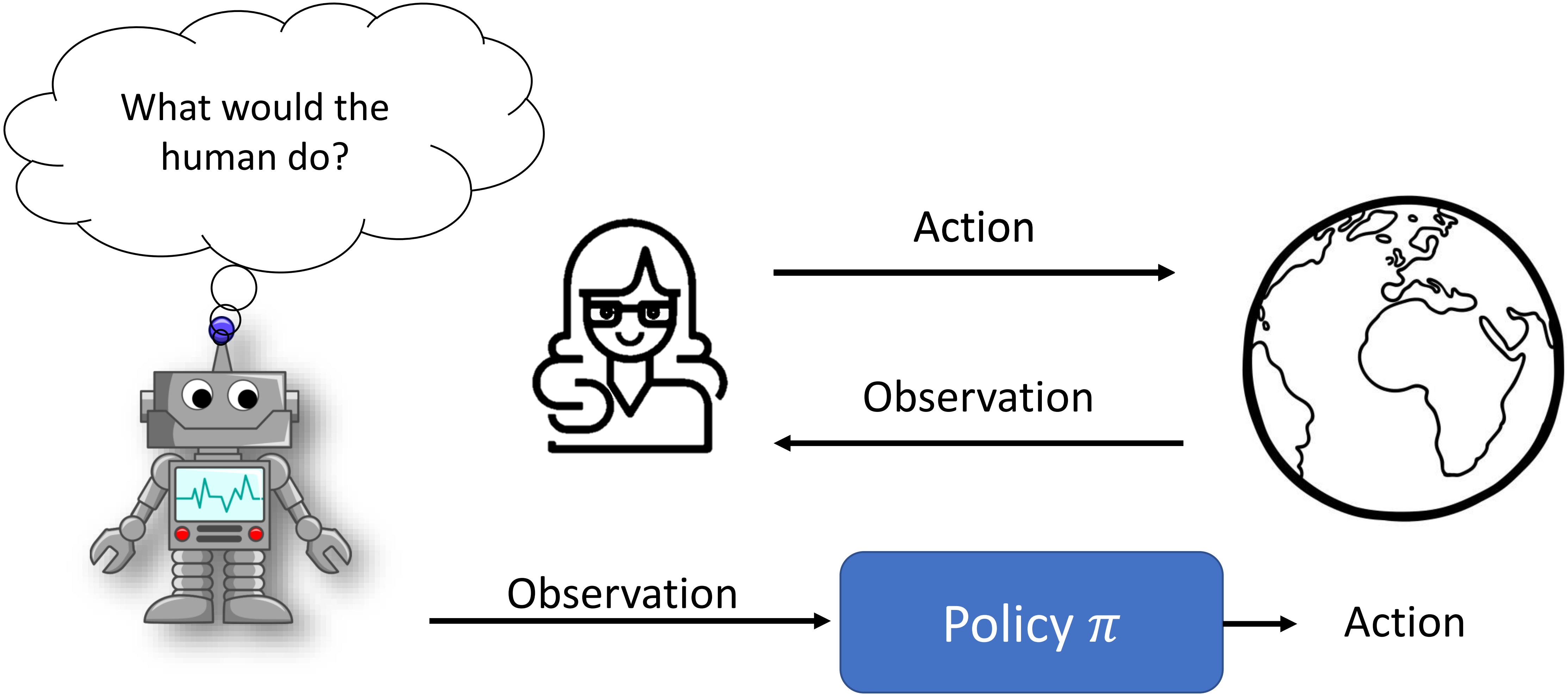
Course feedback is open

- Extra credit if class response rate is 70% or higher
  - Sliding scale if we reach 70%:
    - Extra credit points = response_rate_percentage / 10

Reward Learning
(Inverse Reinforcement Learning)



Why not just imitate behavior? (Behavioral Cloning)



Credit: Simone Giertz

Human Intent Inference

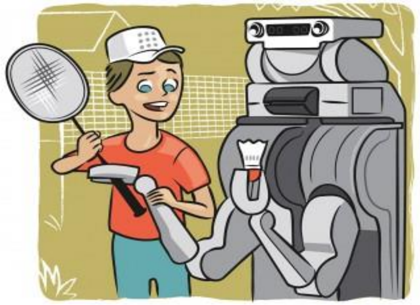
Inverse Reinforcement Learning

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

MDP/R

Imitation Learning

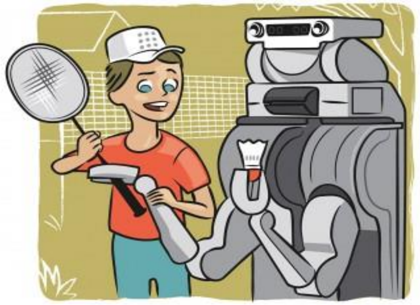
Behavioral Cloning



$\Rightarrow \pi$

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

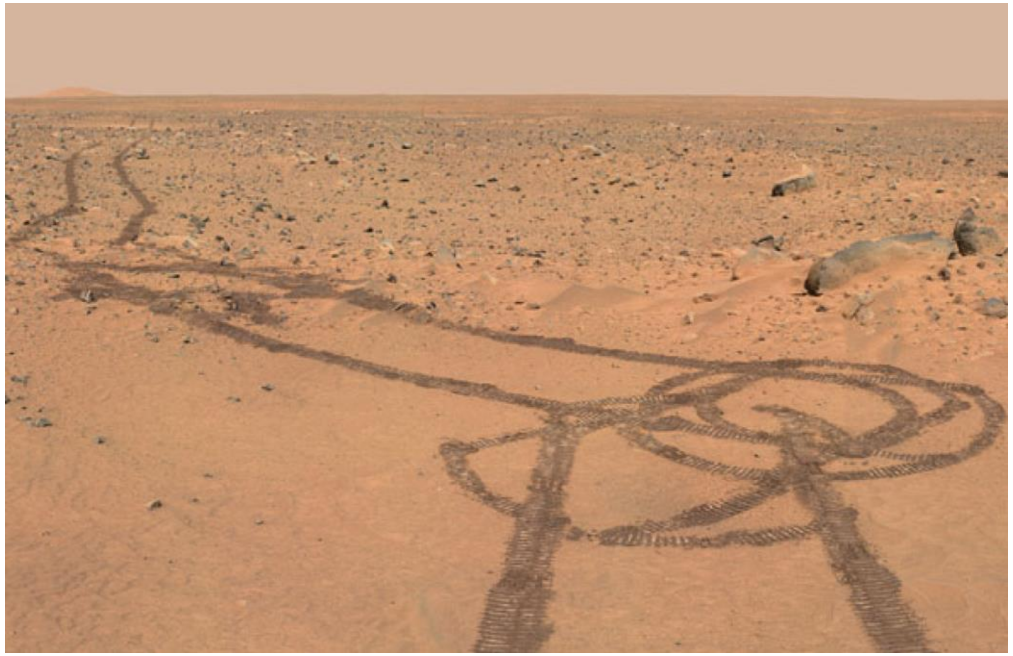
Inverse Reinforcement Learning



$\Rightarrow R \xrightarrow{RL} \pi$

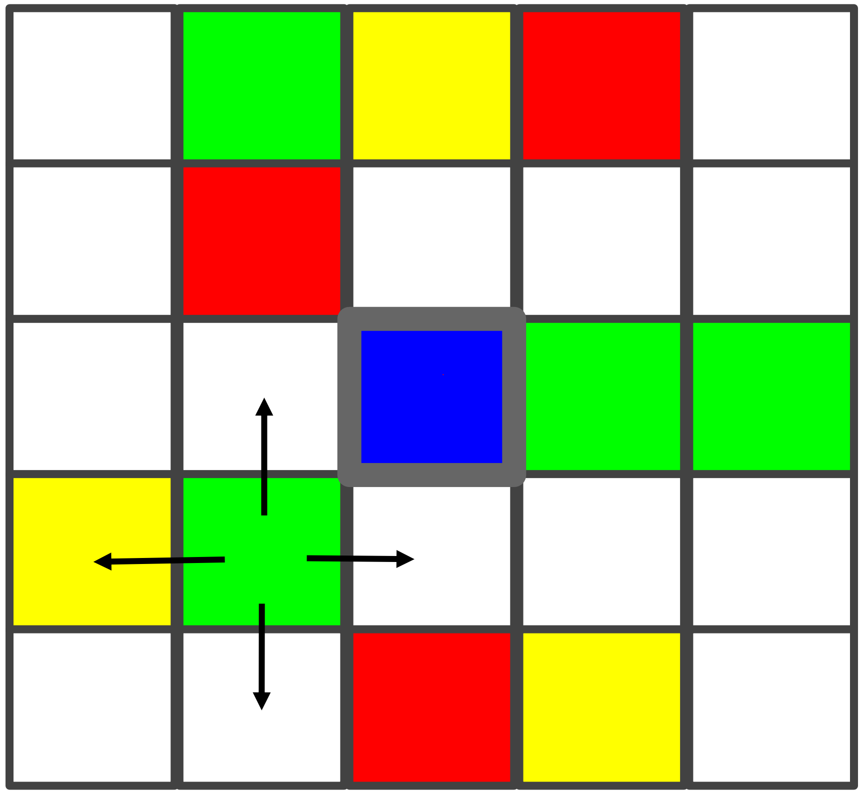
- Answers the "Why?" question
- Explain the demonstrator's behavior
- Learn a reward function capturing the demonstrator's intent
- Can require lots of data and compute
- Better generalization. Can recover from arbitrary states

IRL Example: Teaching a robot to navigate through demonstrations

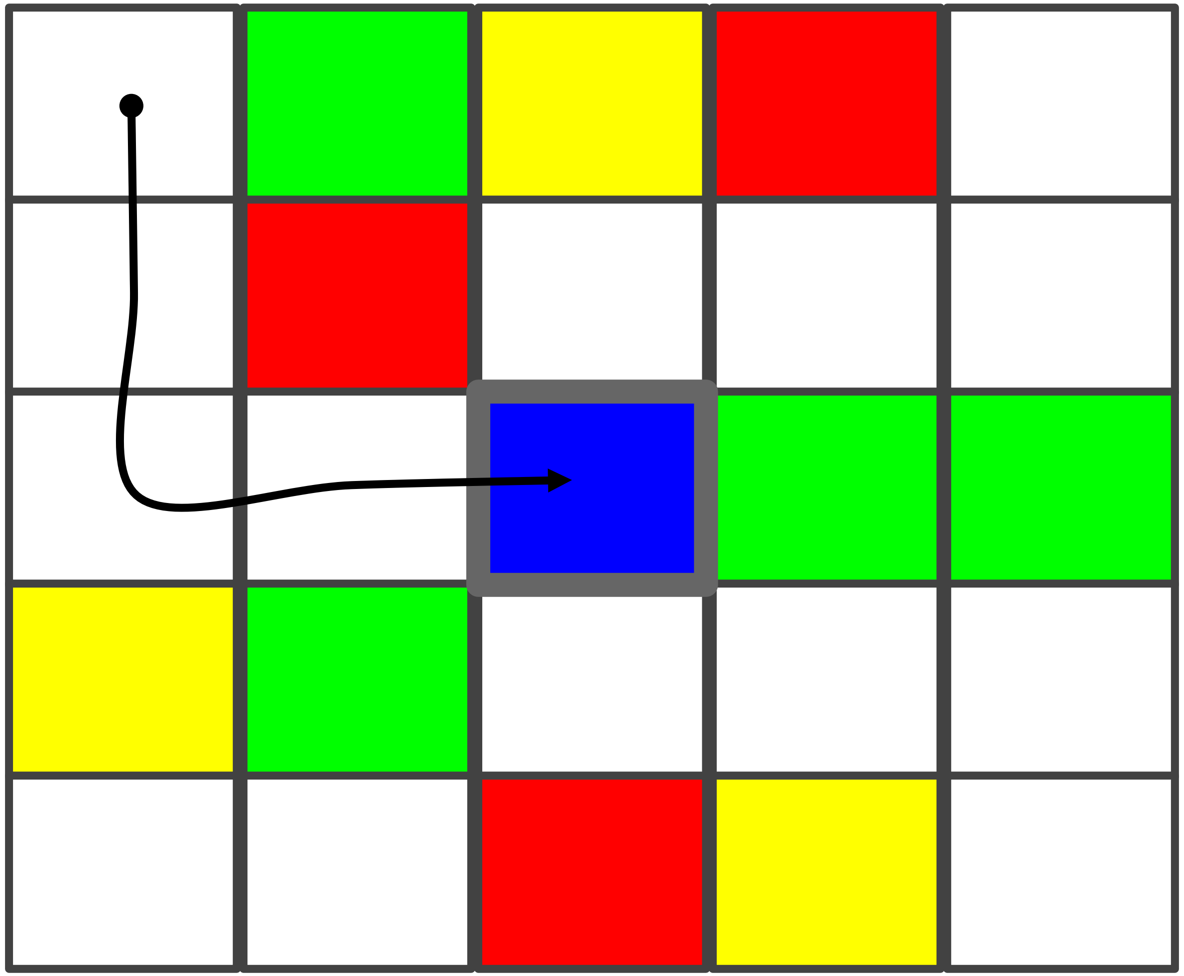


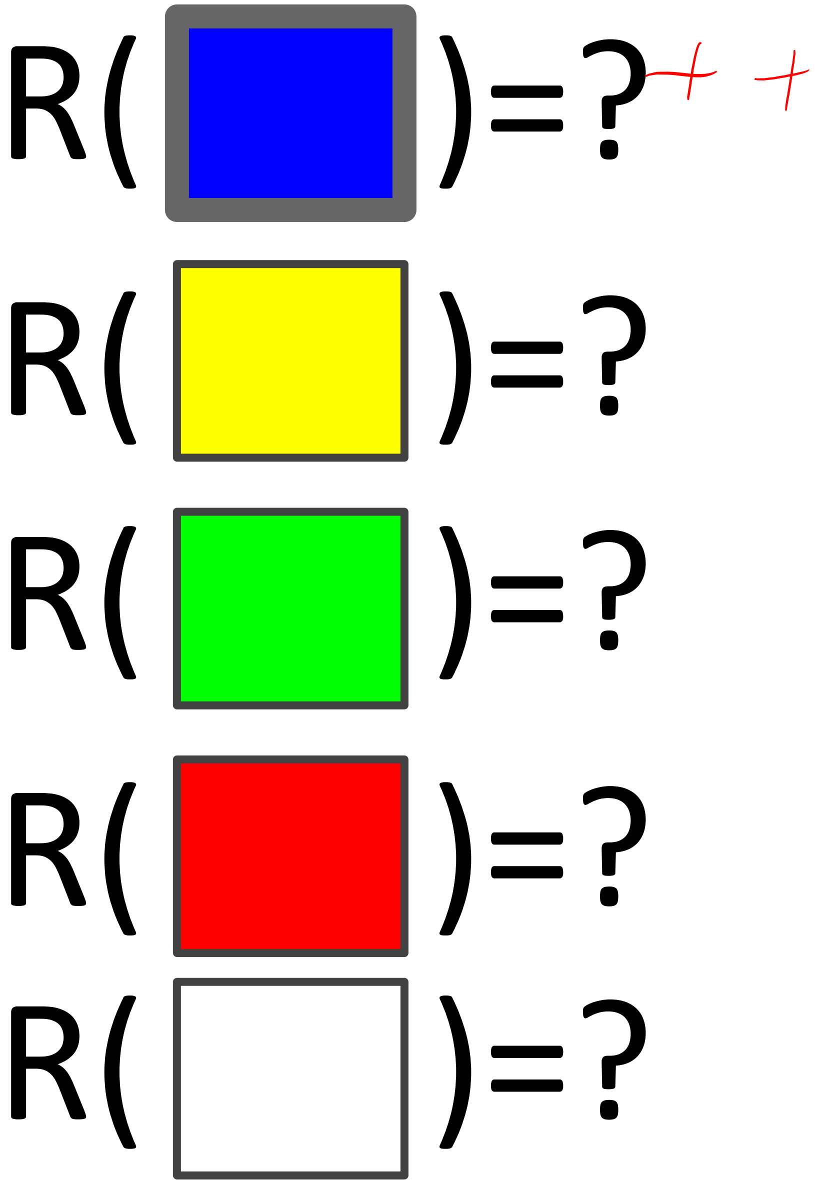
Toy version

T - deterministic

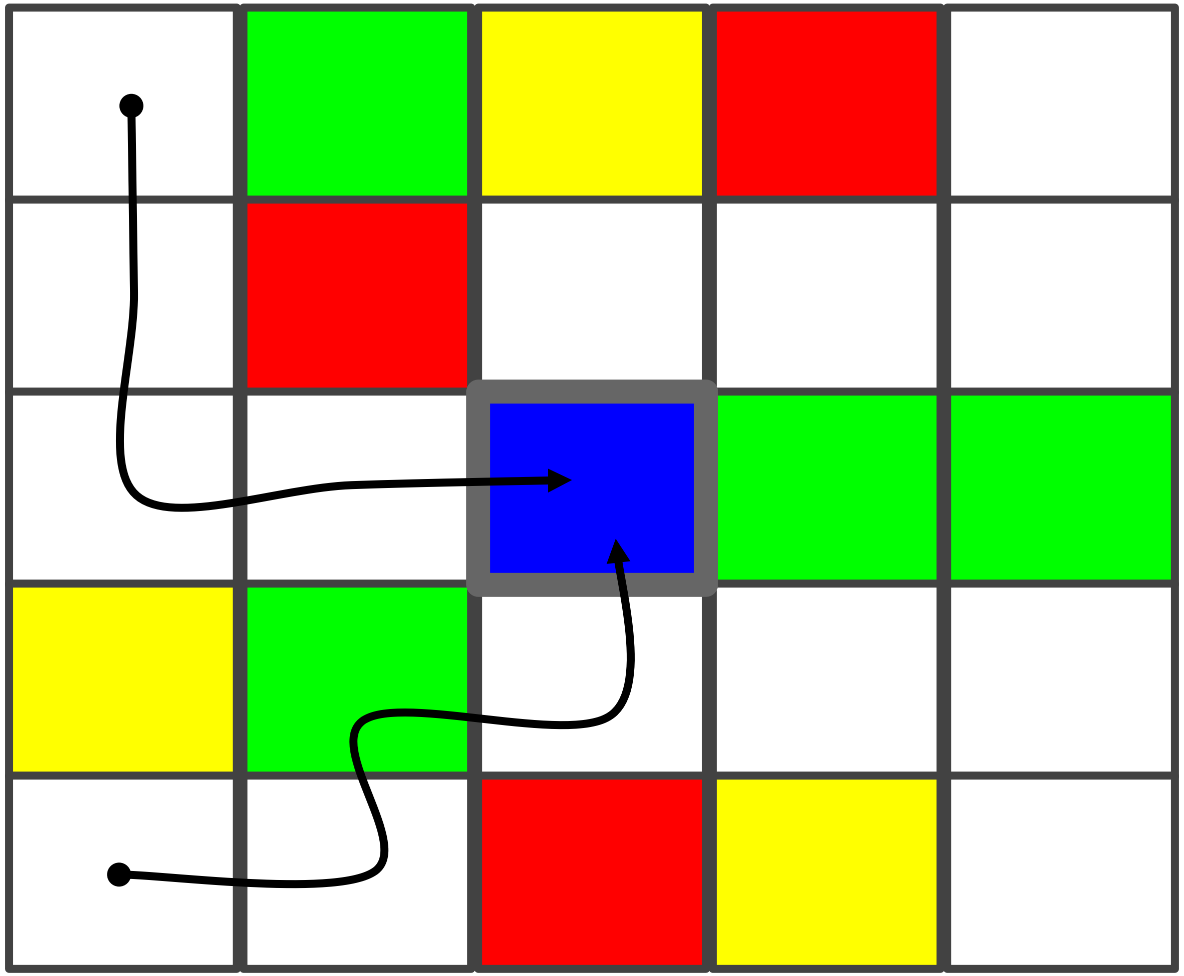


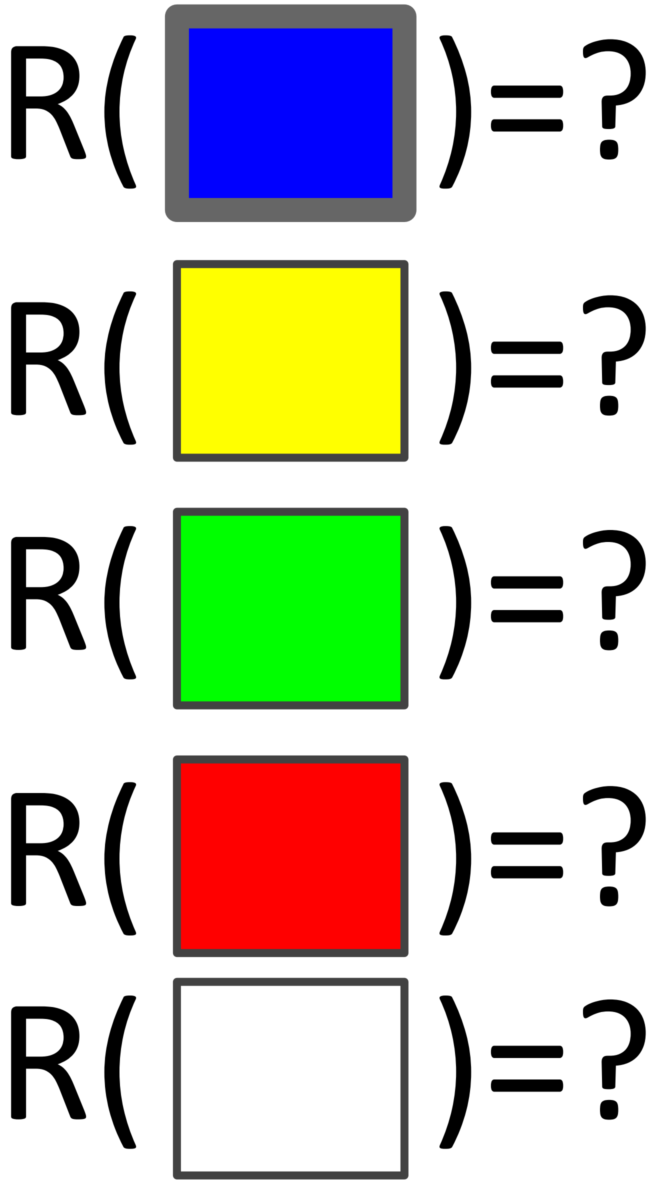
What is the reward?





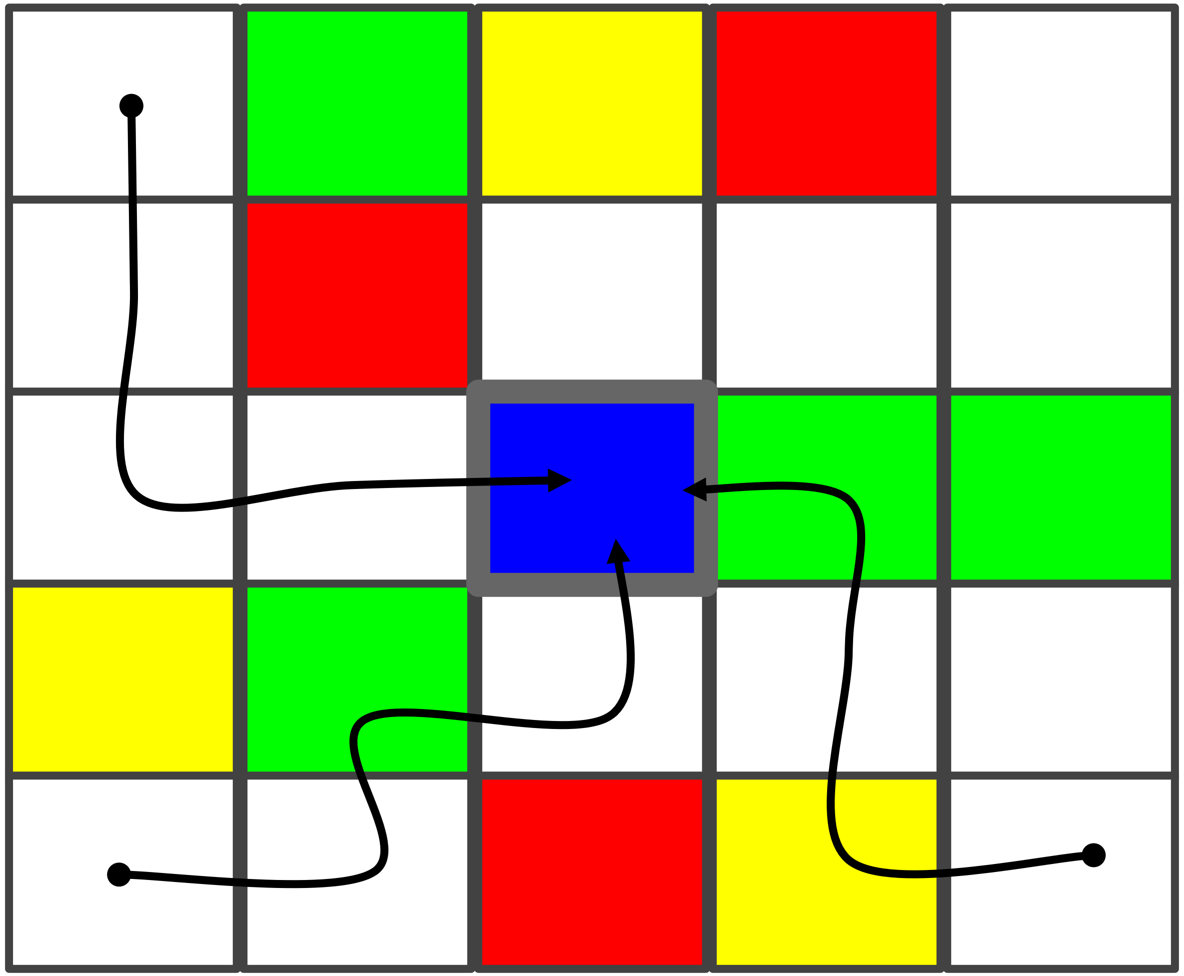
What is the reward?

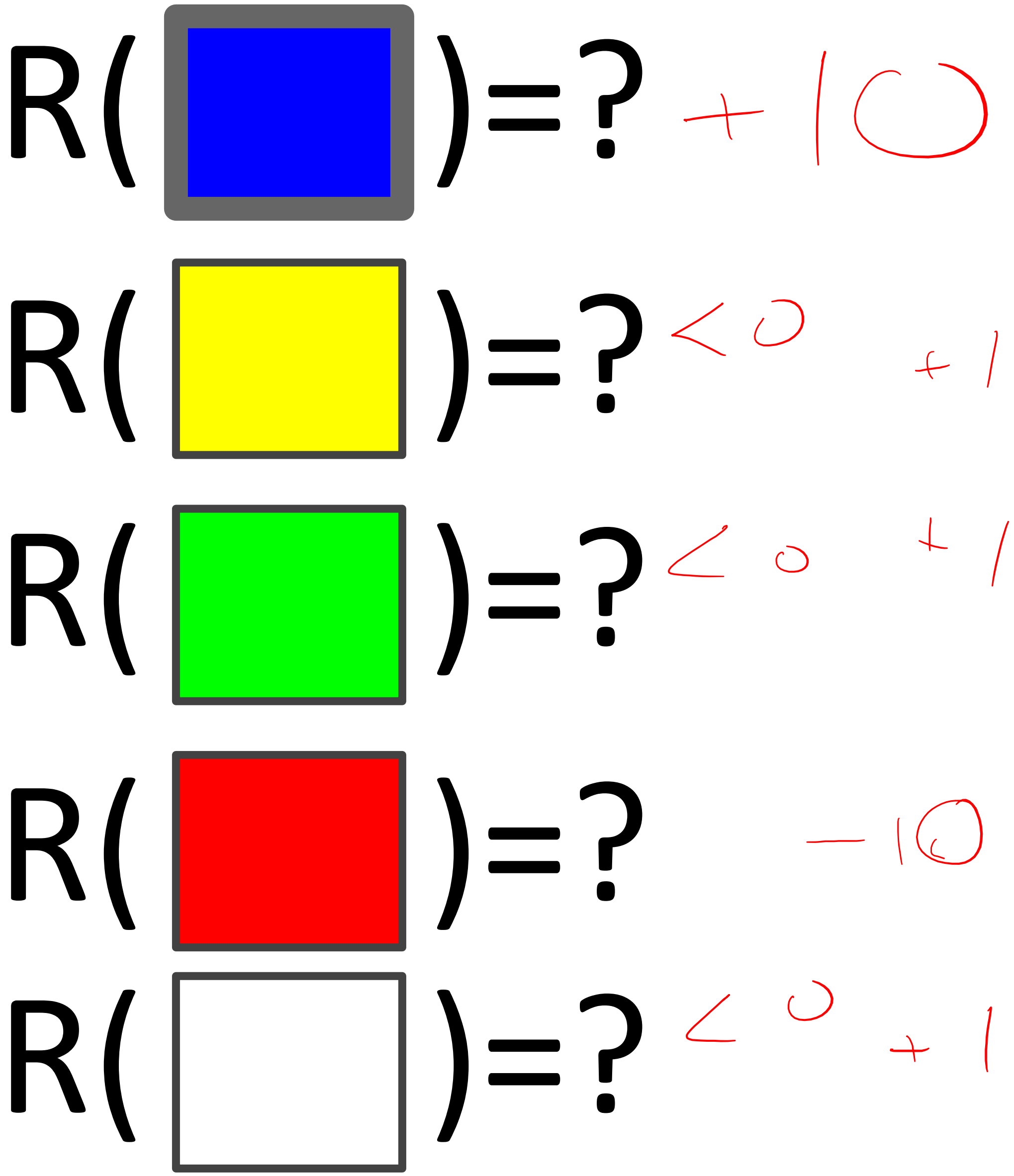




$\gamma = 1$

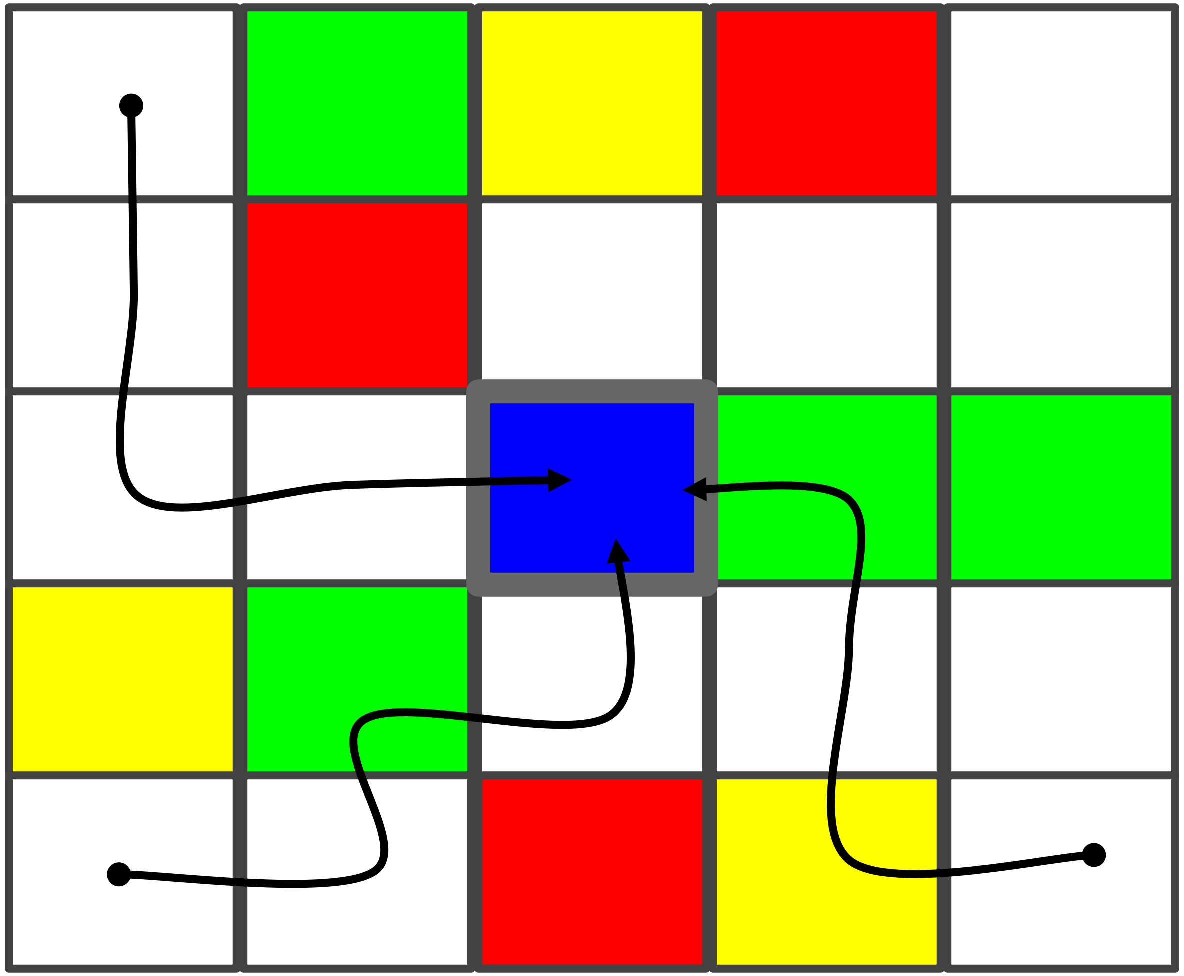
What is the reward?

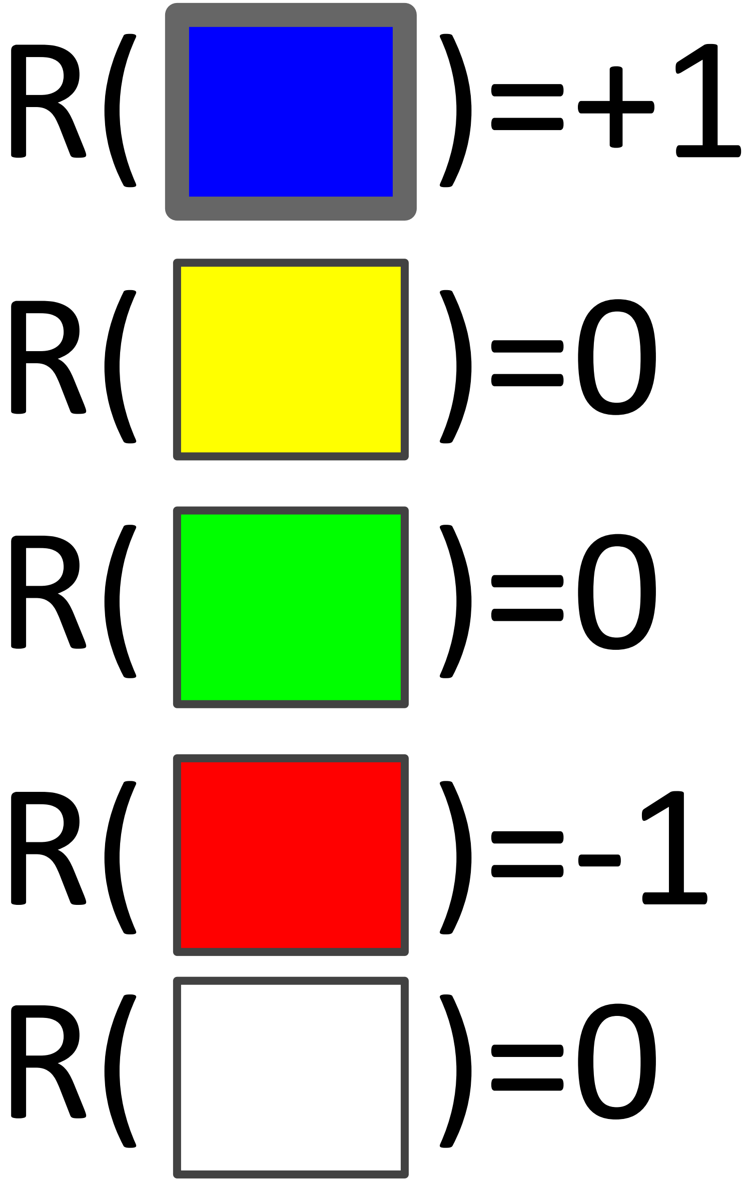




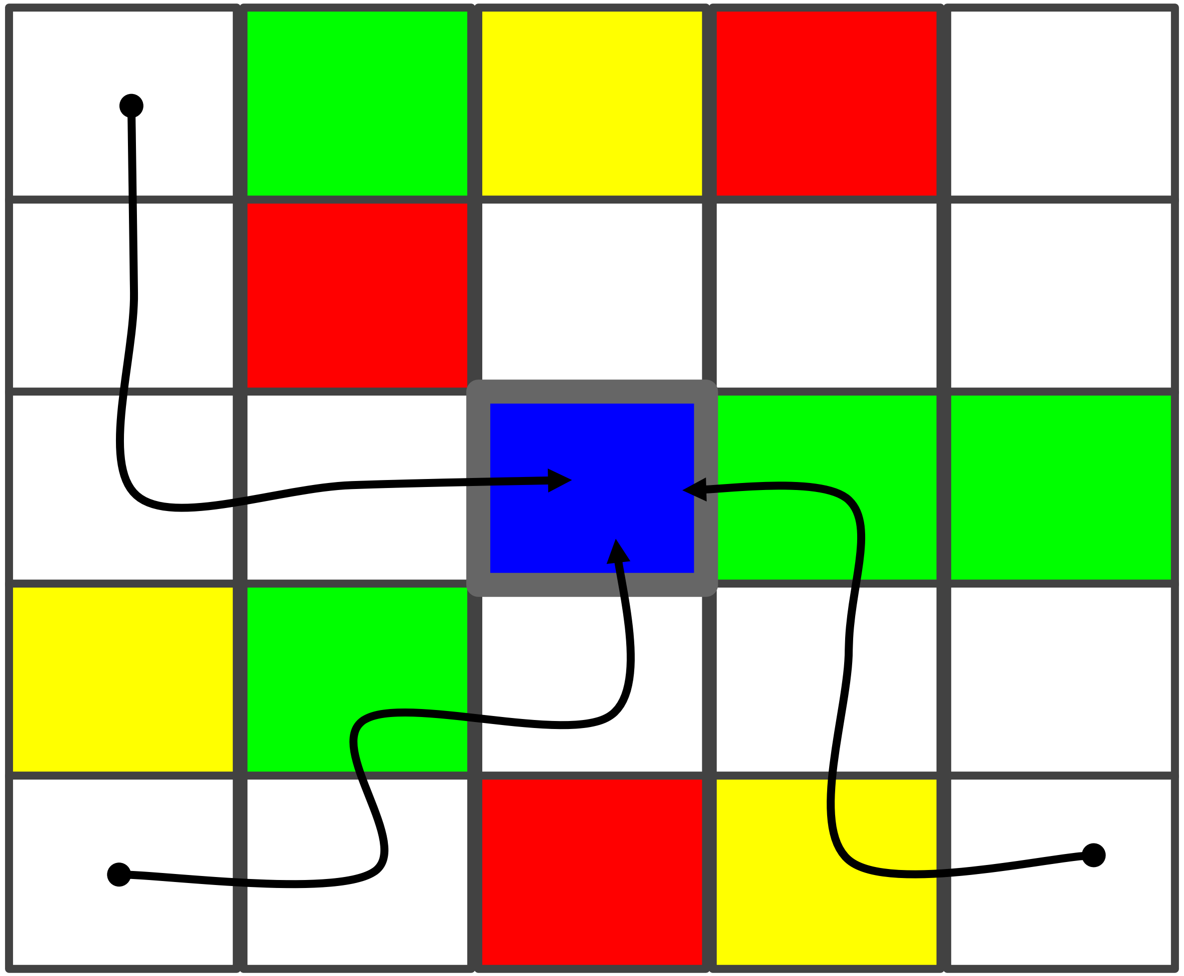
$\gamma < 1$

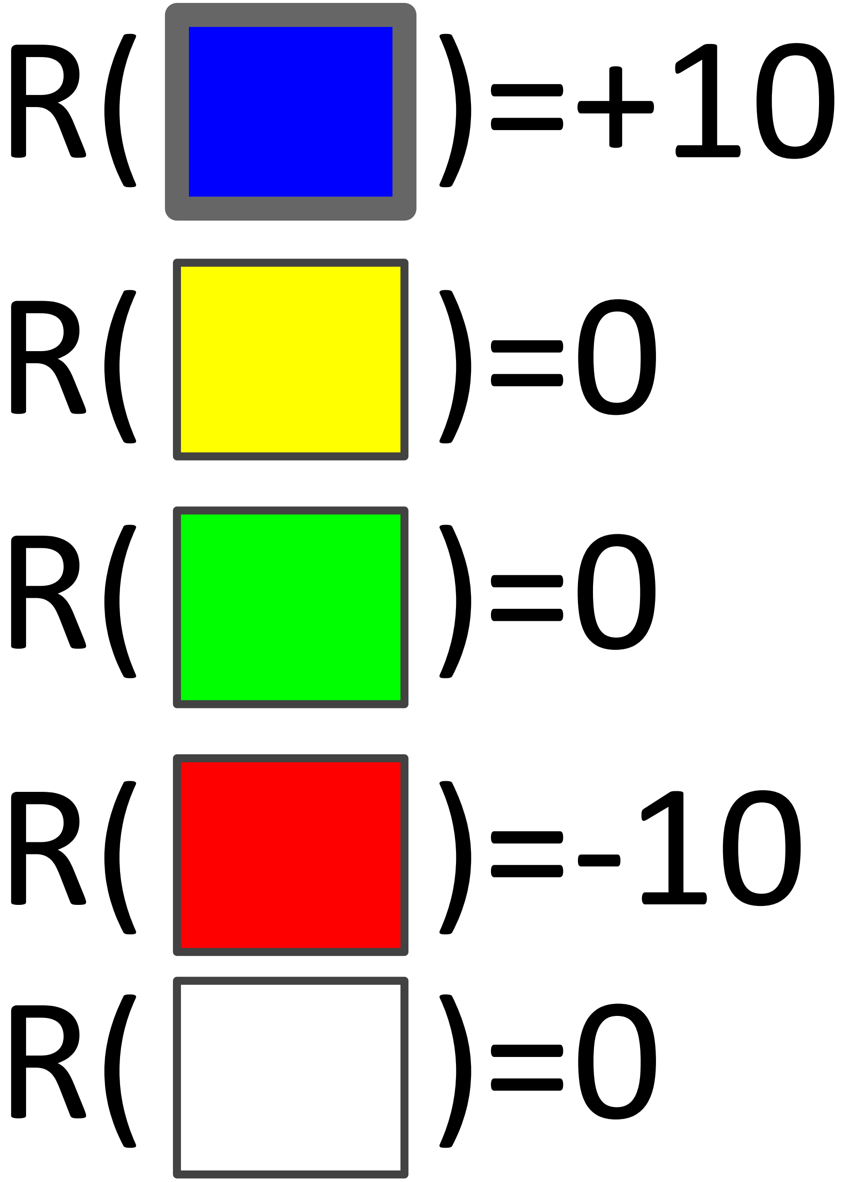
What is the reward?



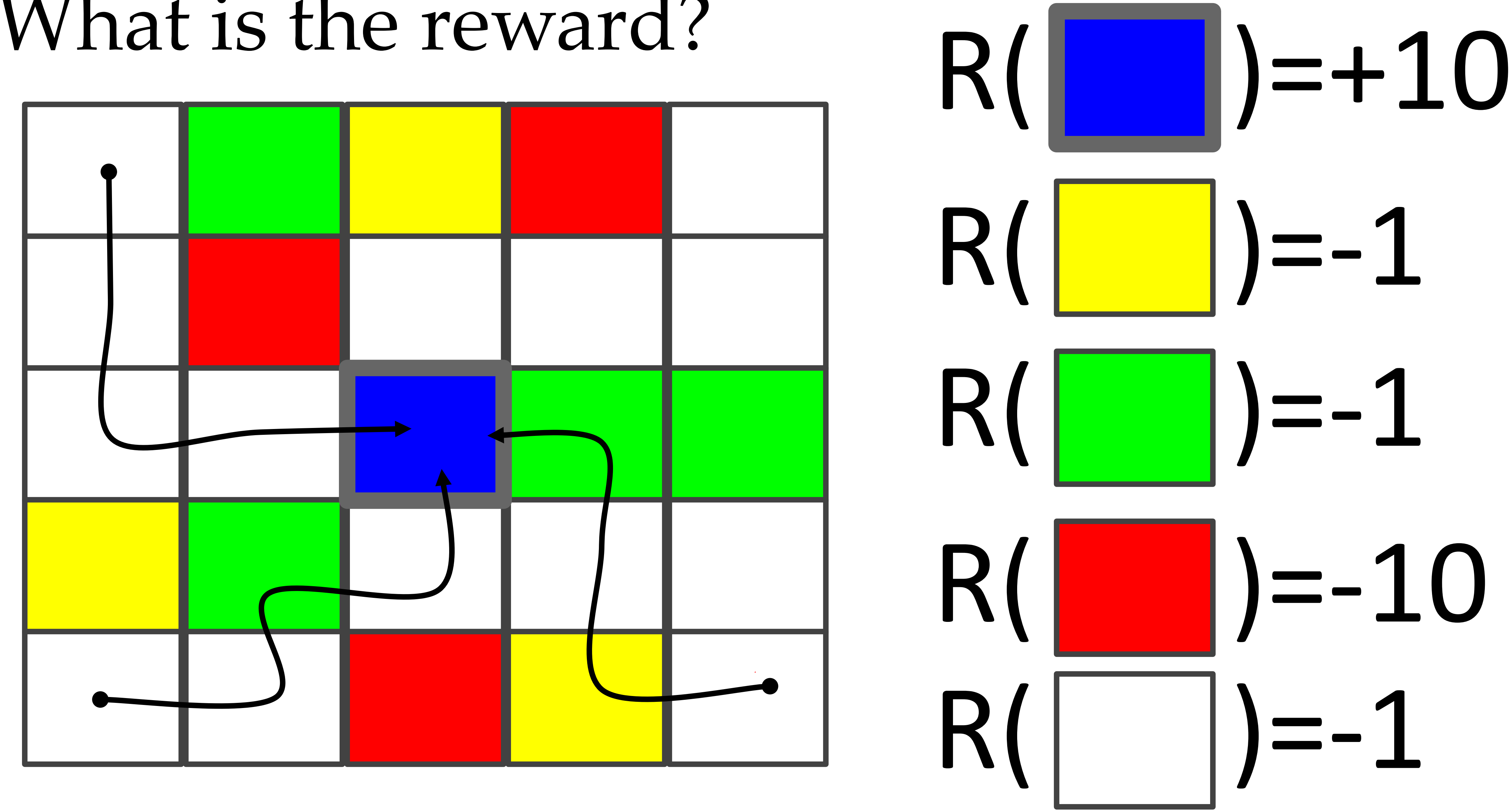


What is the reward?

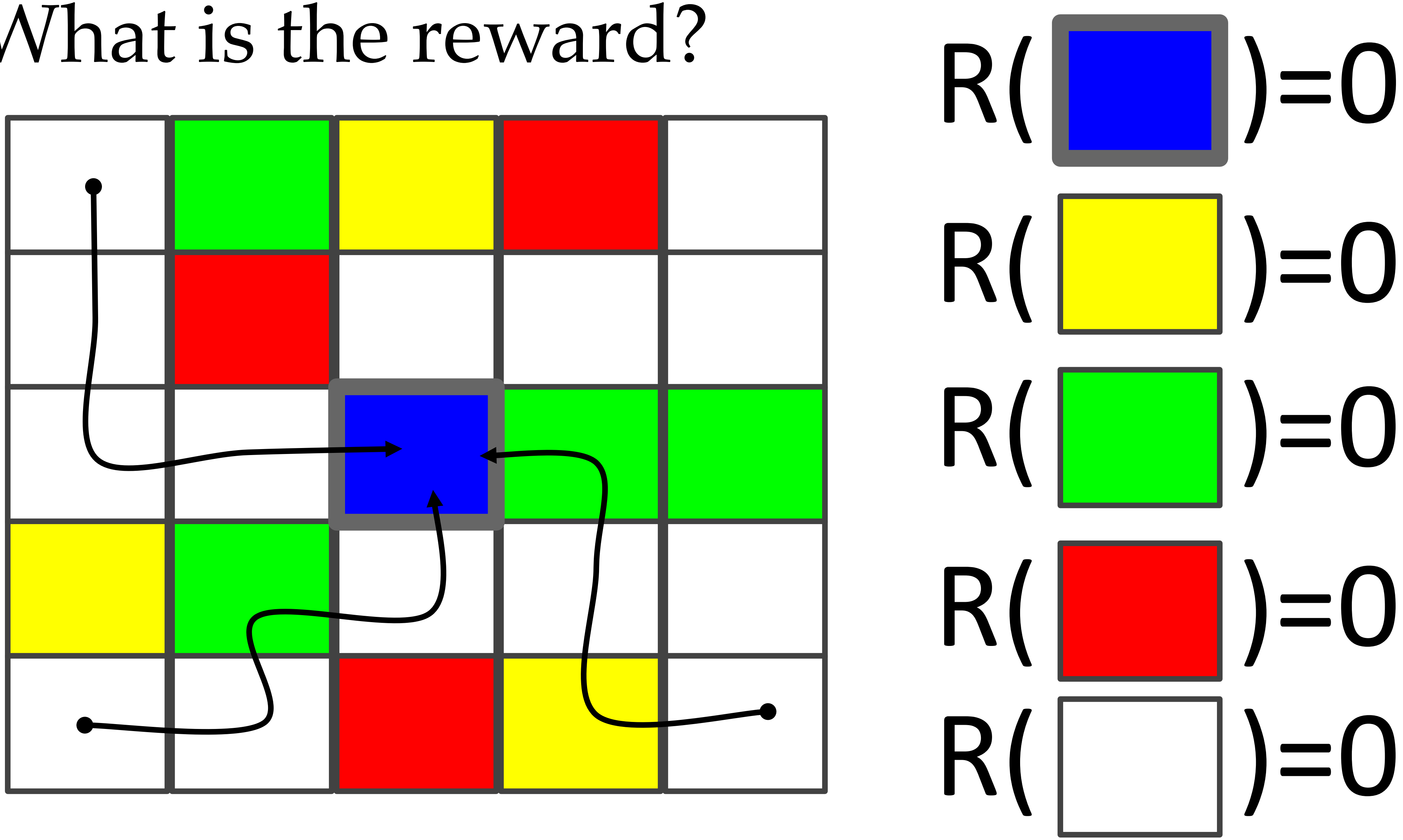




What is the reward?

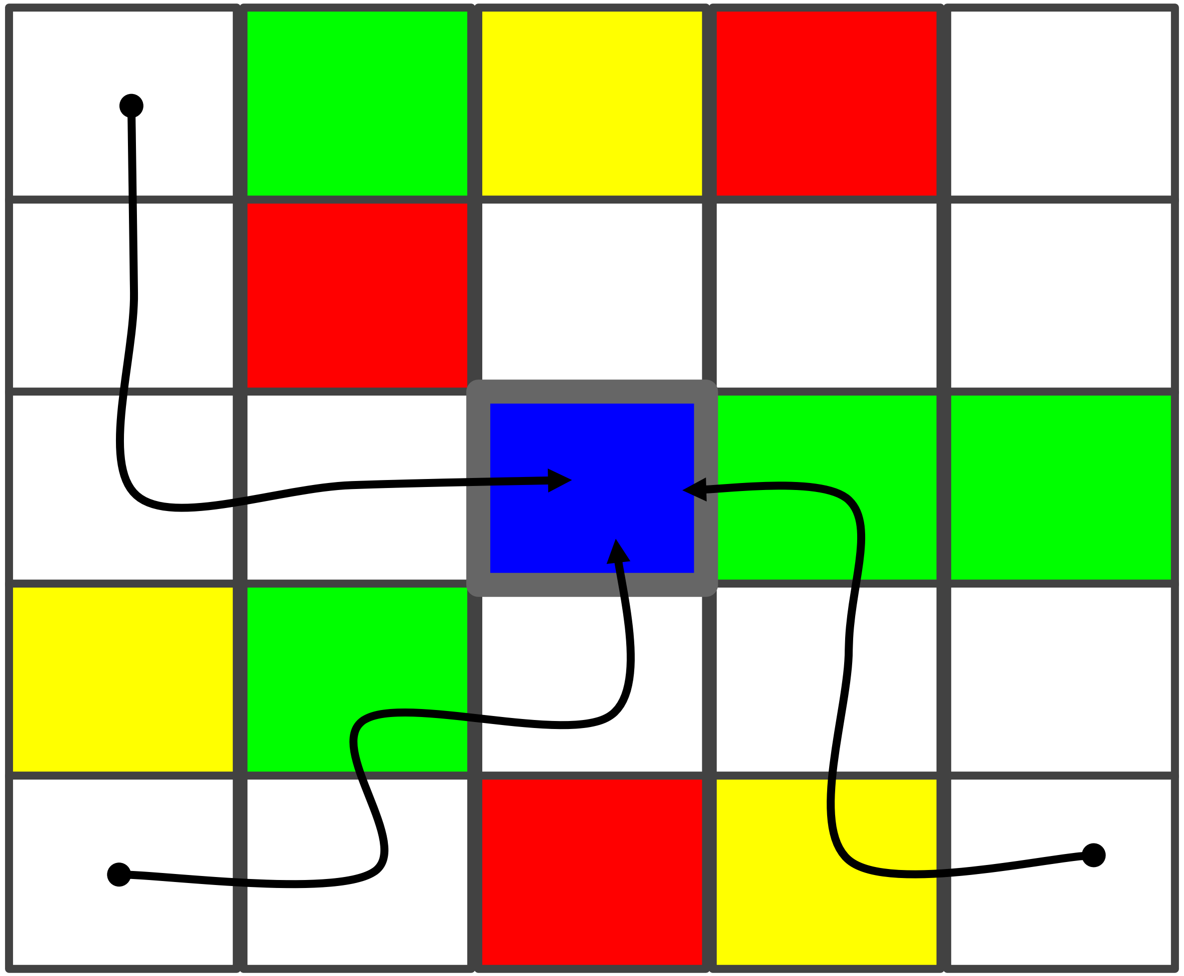


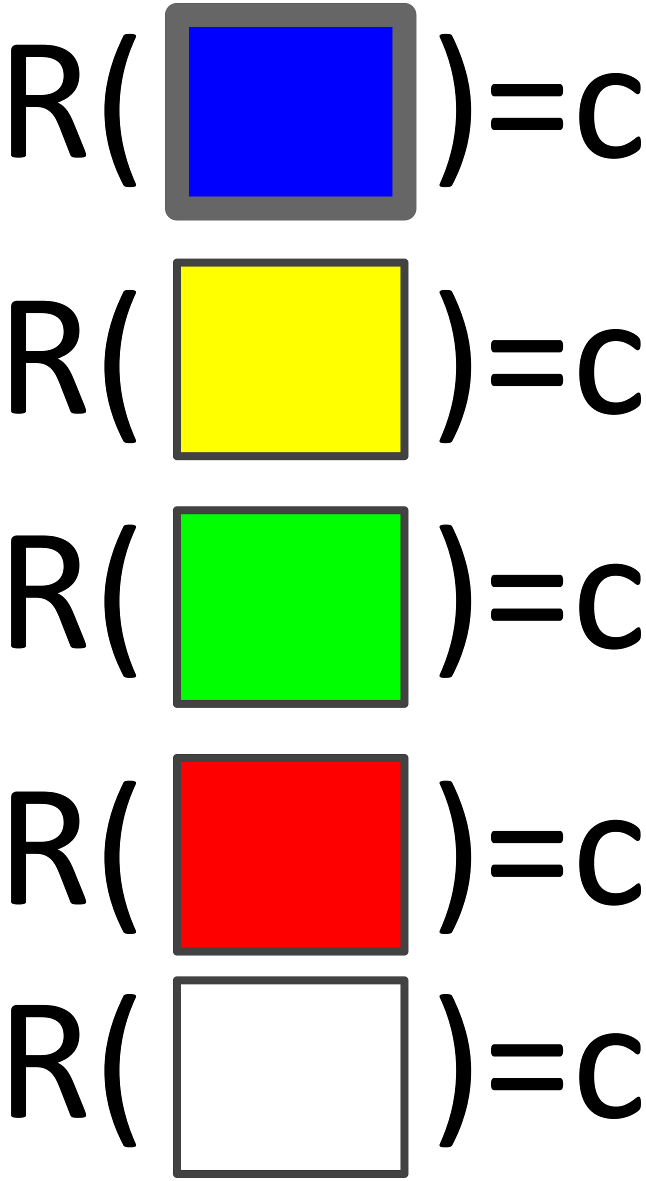
What is the reward?



$c < 0$

What is the reward?





Inverse Reinforcement Learning Formalism

- Given
  - MDP without a reward function
  - Demonstrations from an optimal policy π^*
- Recover ~~the~~ 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)

How would you do this?

- maximize likelihood of D given $\hat{R}$

Basic IRL Algorithm

T, S, A, γ
MDP/R

input

- Start with demonstrations, D
- Guess initial reward function R_0
- $\hat{R} = R_0$
- Loop:

computationally hard

  - 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

Flashback: Approximate Q-Learning

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \ldots + w_n f_n(s,a)$$

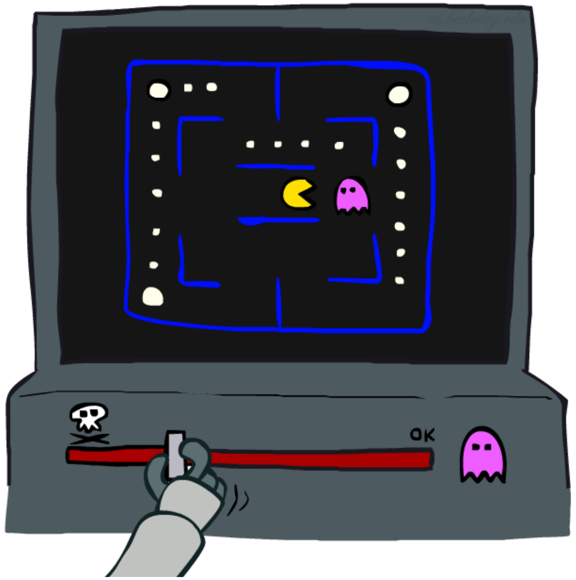
- Q-learning with linear Q-functions:

$$\text{transition } = (s, a, r, s')$$

$$\text{difference} = \left[r + \gamma \max_{a'} Q(s', a')\right] - Q(s,a)$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[\text{difference}]$$ Exact Q's

$$w_i \leftarrow w_i + \alpha\,[\text{difference}]\, f_i(s,a)$$ Approximate Q's



- Intuitive interpretation:
  - Adjust weights of active features
  - E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state's features

- Formal justification: online least squares

Feature count matching

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

$$R(s) = \mathbf{w}^T \phi(s) = w_1 \phi_1(s) + w_2 \phi_2(s) \ldots$$

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

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

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

K

- 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:

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

$\in \mathbb{R}^K$

disc
white
red
yellow
blue

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

Inverse reinforcement learning: feature matching

$\|x\|_\infty = \max_i |x_i|$

(Abbeel and Ng 2004, Syed and Schapire 2007)

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

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

difference in features

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

- 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 $\boldsymbol{w}$

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

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

step size

$\mu_D[i] > \mu_\pi[i]$

Soft Value Iteration

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

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

Soft Maximum

Policy is a softmax policy.

$$e^{Q(A,S) - \log \sum_b e^{Q(b,S)}}$$

$$e^{Q(A,S)} e^{-\log \sum_b e^{Q(b,S)}}$$

$$= \frac{e^{Q(A,S)}}{\sum_b e^{Q(b,S)}}$$

Soft Maximum

$\log(e^{-b} \cdot e^{b}) = \log(e^0) = 0$

- Let a > b
- $\log(e^a + e^b) =$ $\log(e^a + e^b) + \log e^{-b} + \log e^{b}$

$\log((e^a + e^b)e^{-b}) + b$

$= \log(e^{a-b} + 1) + b$

- If a = b
- $\log(e^a + e^b) =$ $\log(2e^a)$

$= \log(2) + \log e^a = \log(2) + a$

- In general $\max\{x_1, x_2, \dots, x_n\} \leq \log \sum_i \cancel{x_i}\, e^{x_i} \leq \max\{x_1, \dots, x_n\} + \log n$

Soft Value Iteration

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

Soft Maximum

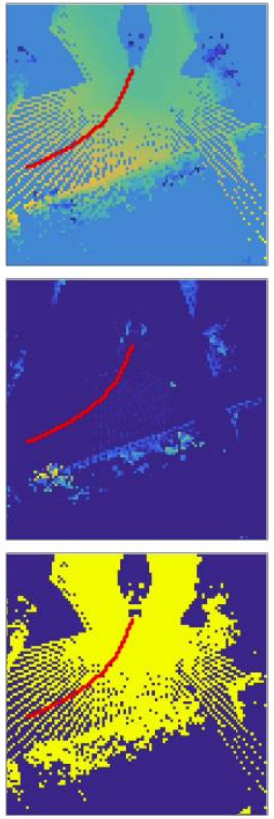
$$V^{\text{soft}}_{\pi_\Theta}(S_t) = \log \sum_{A_t \in \mathcal{A}} e^{Q^{\text{soft}}_{\pi_\Theta}(A_t,S_t)}$$

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

- Initialize value of terminal states to 0 and other values to $-\infty$
- Repeat:
  - Solve for Q
  - Sove 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]



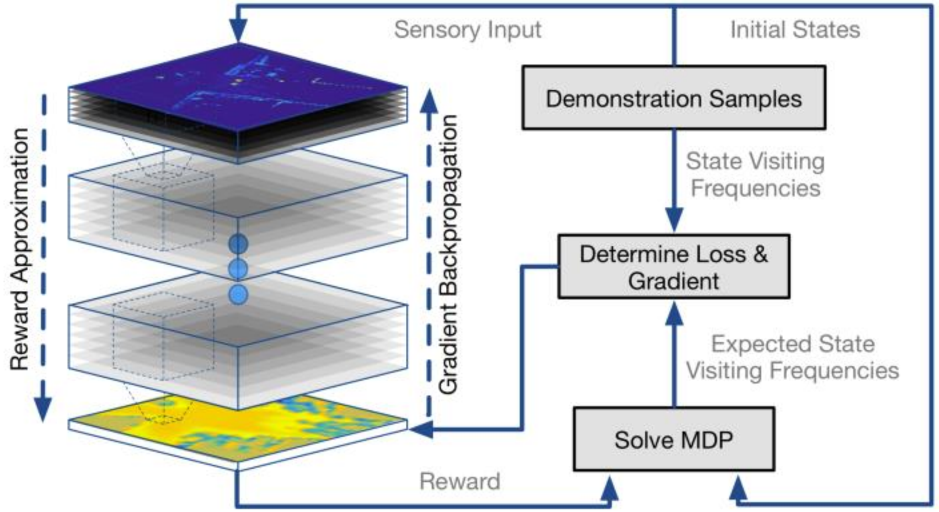


Fig. 1: Schema for training neural networks in the Maximum Entropy paradigm for IRL.

Another way to look at MaxEnt IRL

$\int_\tau e^{R_\theta(\tau)} d\tau$

$$P(\tau) = \frac{e^{R_\theta(\tau)}}{Z} \quad \propto$$

$\max \log P(\tau)$

$$Z = \int e^{R_\theta(\tau)} d\tau$$

$\max \log \frac{e^{R_\theta(\tau)}}{Z}$

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

$\max \log e^{R_\theta(\tau)} - \log Z$

$|D| = N$

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

How to avoid fully solving MDP

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

- Estimate Z with a finite set of trajectories Z_{τ}.
- Loop:
  - Update parameters θ so demonstrations have higher reward than trajectories in Z_{τ}.
  - Update Z_{τ}

How to make this more tractable

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

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

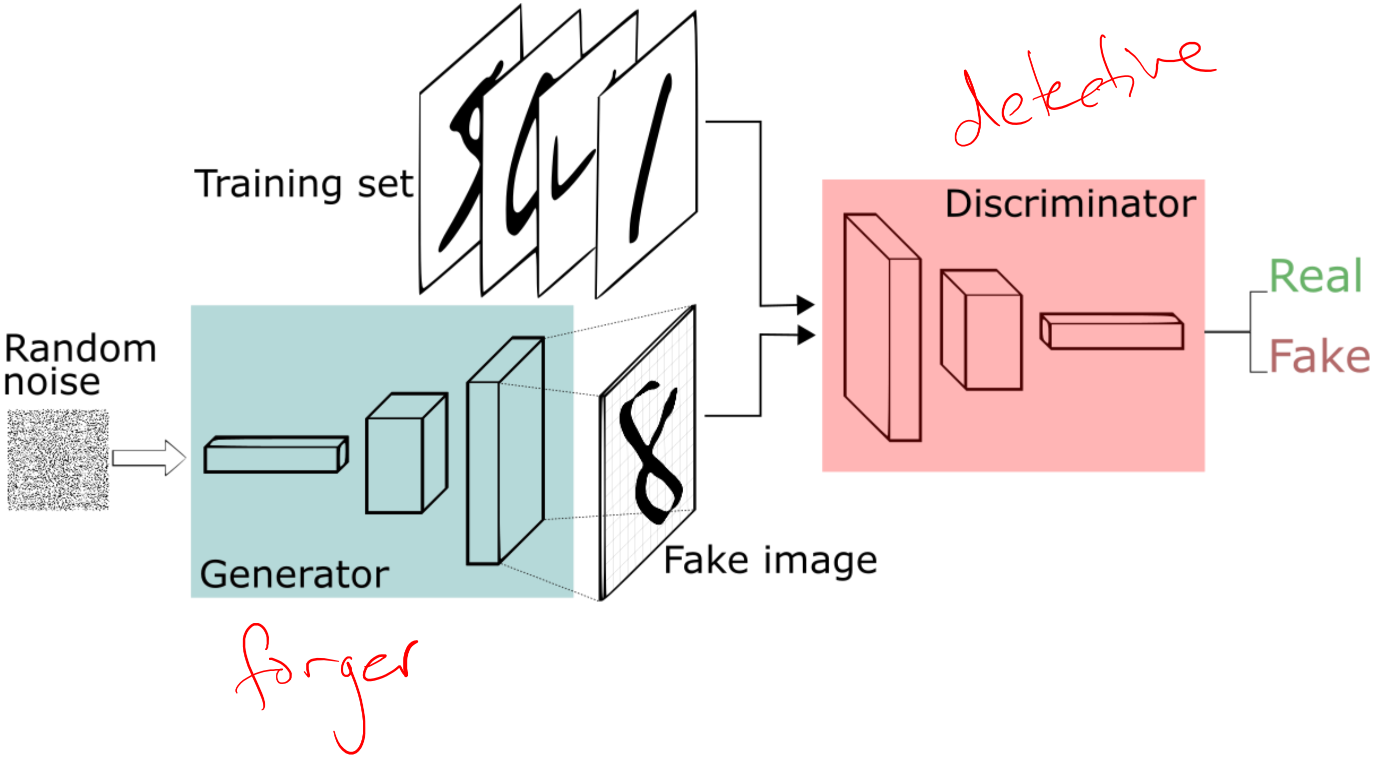
Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization

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

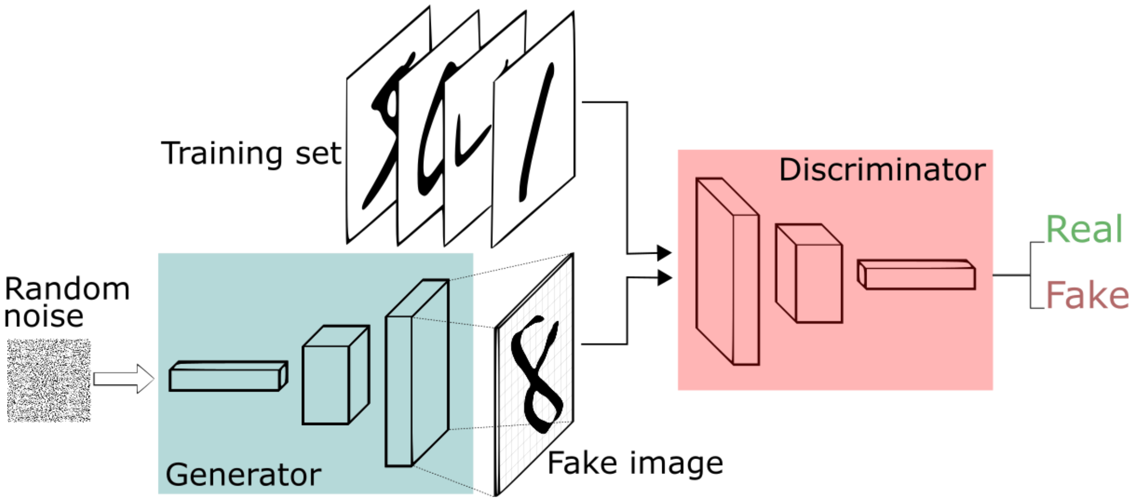
Use current policy to approximate Z.
Alternate between a few steps of reward updates and a few steps of policy updates.

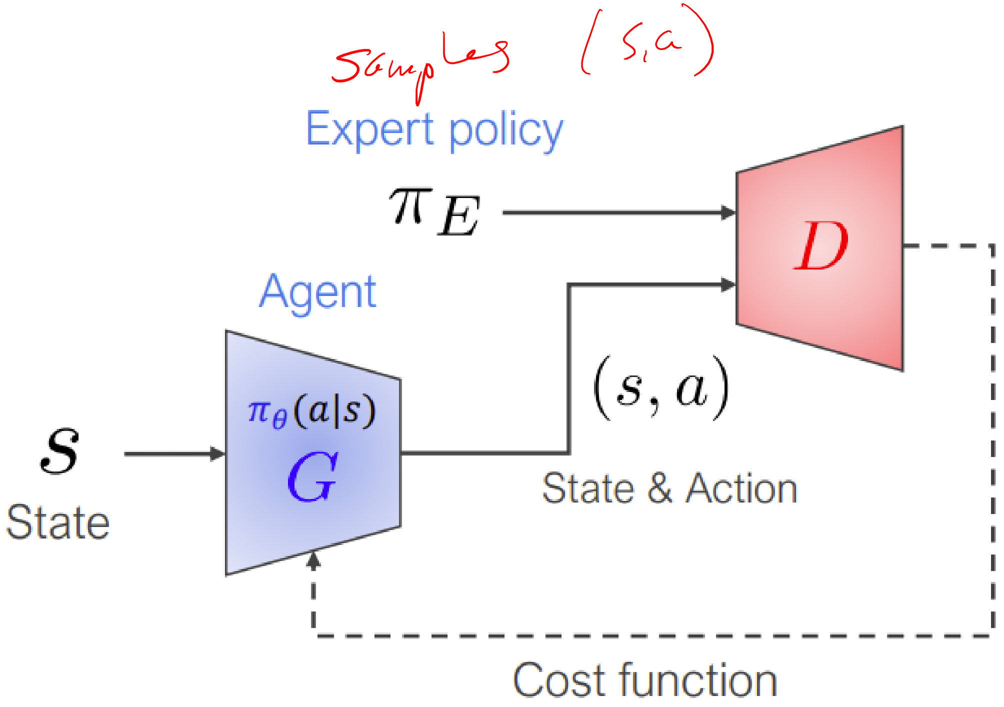
Finn et al. “Guided Cost Learning.” 2016

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?

likelihood

prior

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

Markov Chain Monte Carlo (MCMC)

Markov chain:

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \dashrightarrow$$

$$P(X_1) \qquad P(X_t|X_{t-1})$$

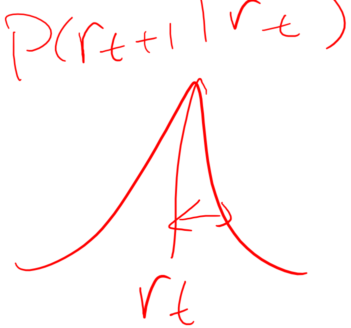
Stationary Distribution:

$$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x)P_\infty(x)$$

MCMC is a sampling approach for Bayesian inference where we construct a Markov chain such that the stationary distribution is the posterior distribution we care about.

$r_0 \to r_1 \to r_1 \to \ldots$

MCMC (Metropolis Hastings Algorithm)



- We want to sample from P(R|D)
- Start with random sample r_0
- Loop
  - Sample $r' \sim q(R_{t+1}|r_t)$ proposal
  - With probability $\min\left\{1, \underbrace{\frac{P(r'|D)}{P(r_t|D)}}_{\text{posterior ratio}}\right\}$ set $\cancel{x_{t+1}} = \cancel{x'}$ $r_{t+1} = r'$
  - Else set $r_{t+1} = r_t$

Assume q is symmetric. For example, a Gaussian distribution with mean x_t and standard deviation σ

Accept!

Reject!

Normalizing constant cancels in the ratio!

$$\frac{P(r'|D)}{P(r|D)} = \frac{\frac{P(D|r')P(r')}{\cancel{P(D)}}}{\frac{P(D|r)P(r)}{\cancel{P(D)}}} = \frac{P(D|r')P(r')}{P(D|r)P(r)}$$

Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

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

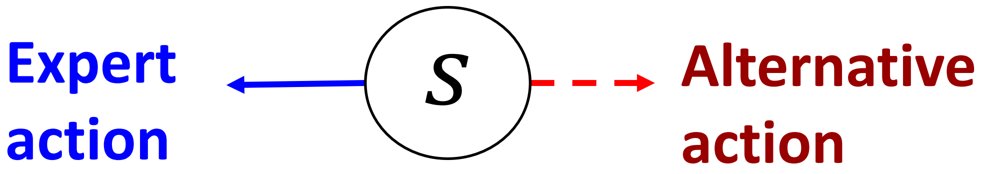
$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{c\boxed{Q^*(s,a,R)}}}{\sum_{b\in A} e^{cQ^*(s,b,R)}}$$

$Q^*(s, a, R) =$ 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 Boltzman rational
  - Demonstrator follows a softmax policy with inverse temperature c

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



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

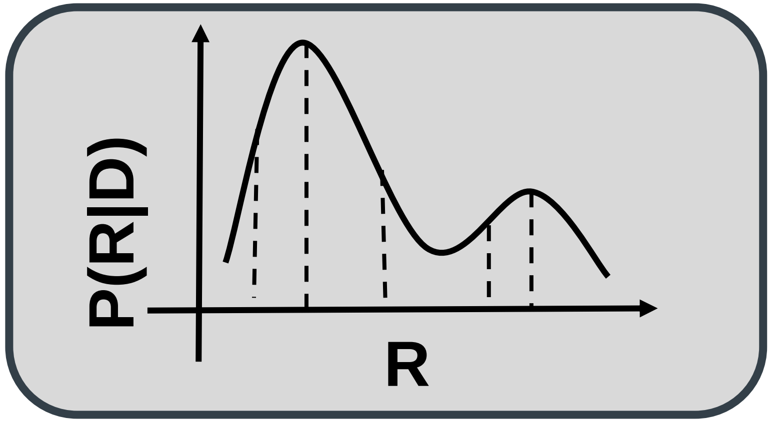
Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

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

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

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

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



Applications of Bayesian IRL

$P(R \mid D)$

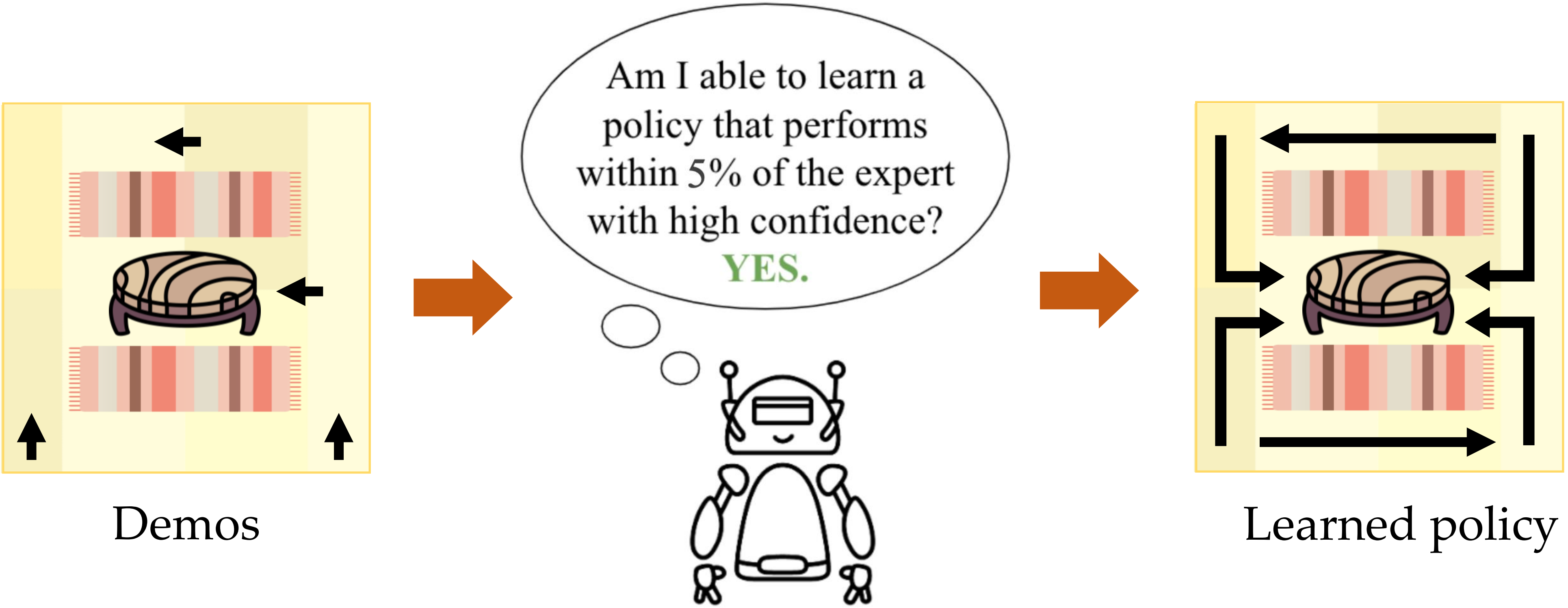
- Active Learning — the algorithm picks its training data
- Uncertainty Estimation
- Demonstration Sufficiency

BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

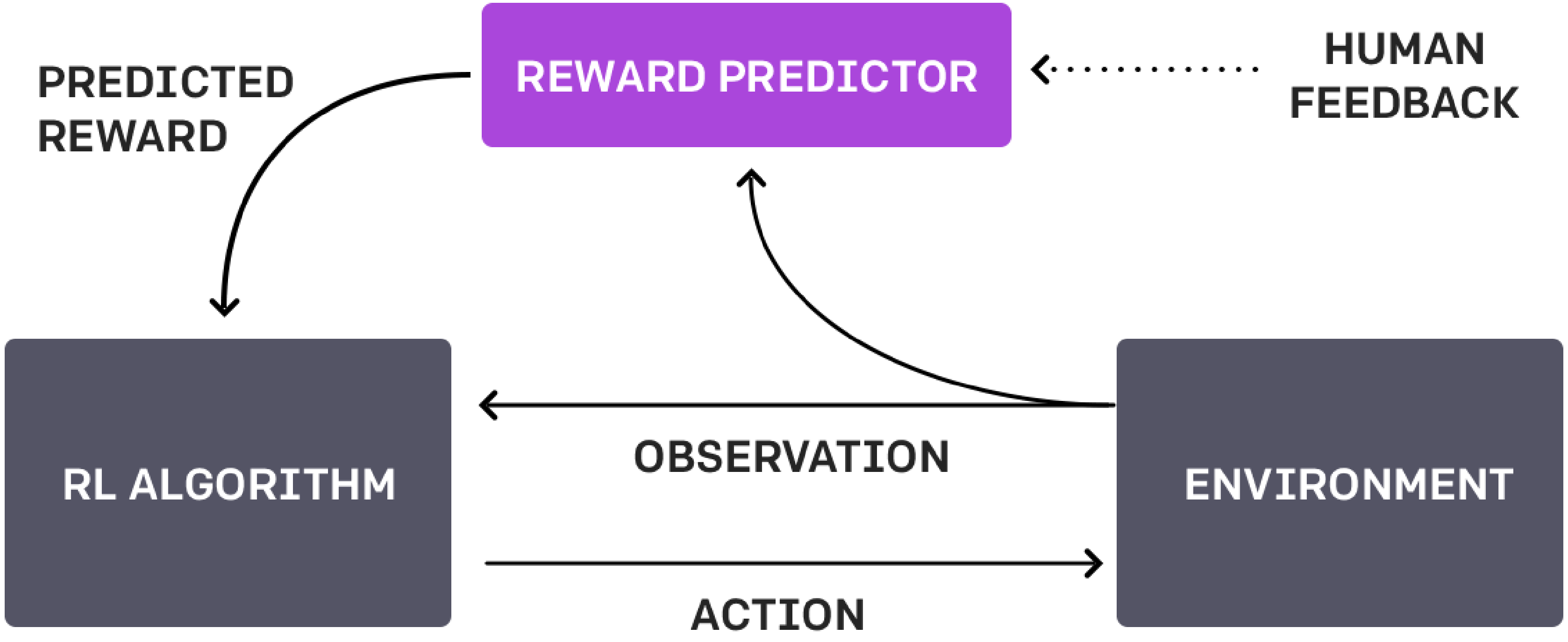
Center for Human-Compatible Artificial Intelligence

ROBOTICS CENTER
THE UNIVERSITY OF UTAH

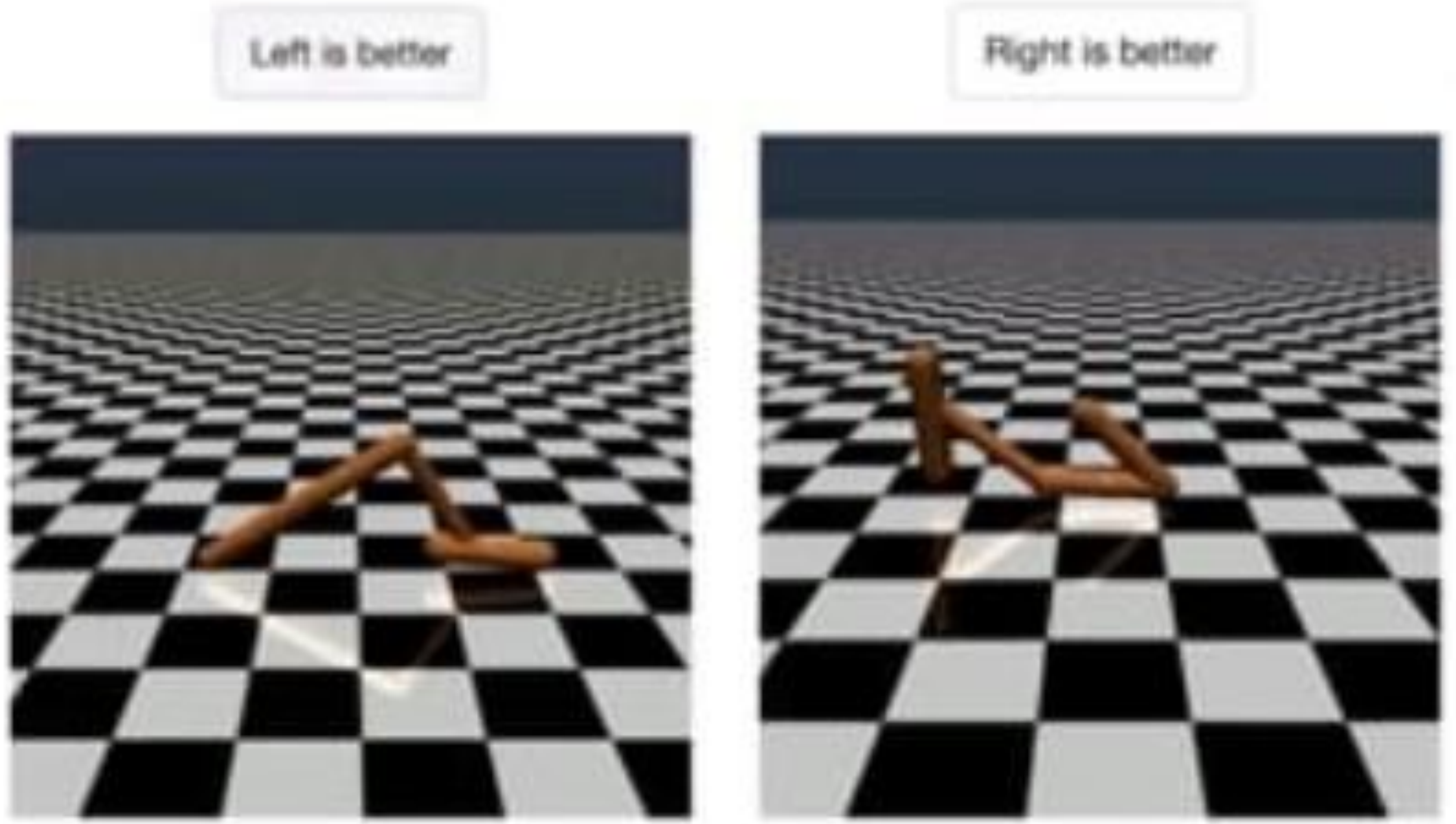
Autonomous Assessment of Demonstration Sufficiency via Bayesian Inverse Reinforcement Learning



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?

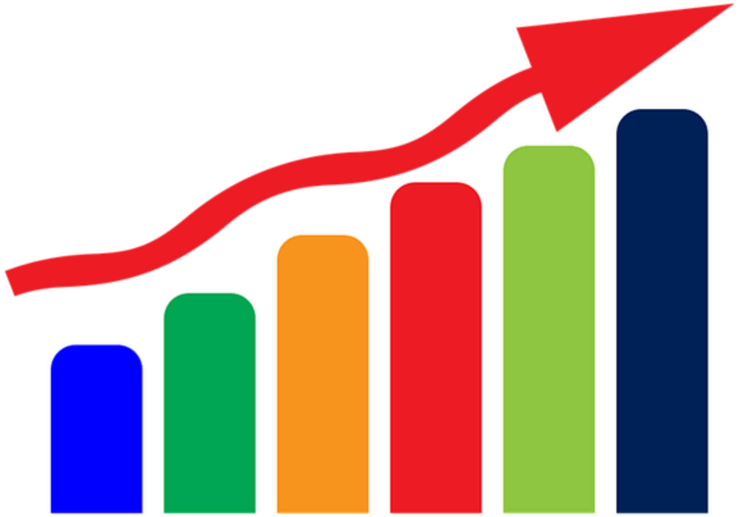
Inverse Reinforcement Learning

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

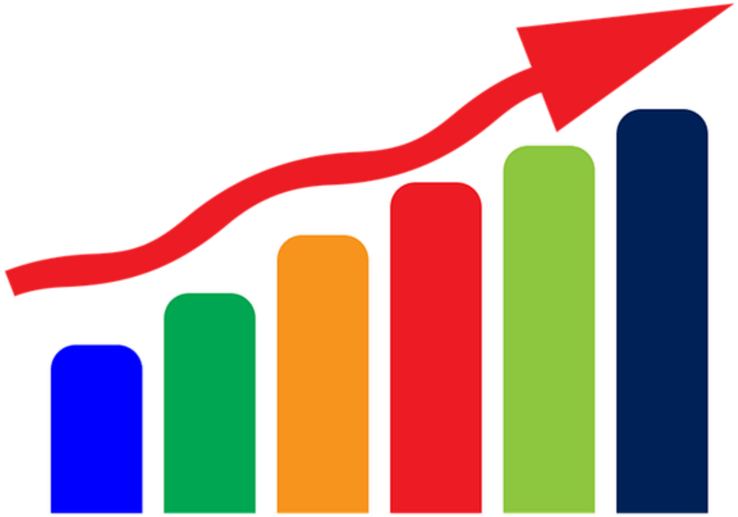
Prior approaches …

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 …

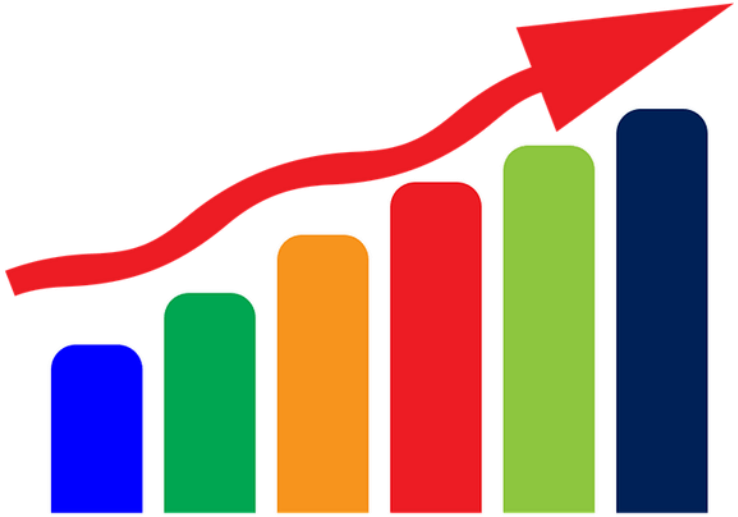
1. ~~Typically couldn't do much better than the demonstrator.~~

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

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

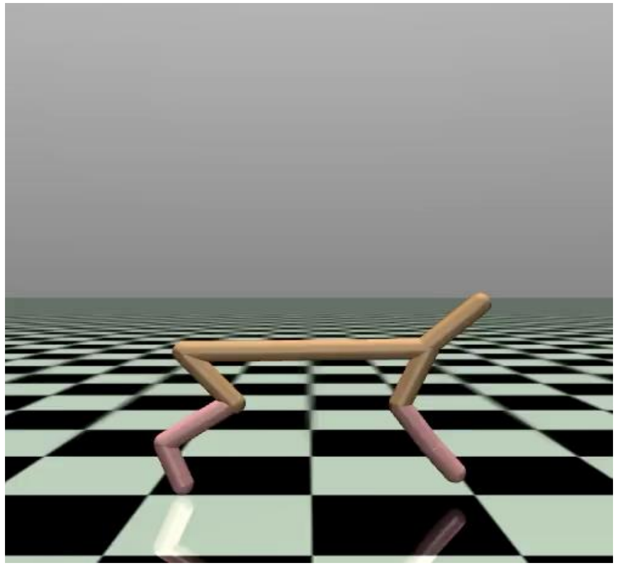
   Reward learning becomes a supervised learning problem.

Pre-Ranked Demonstrations

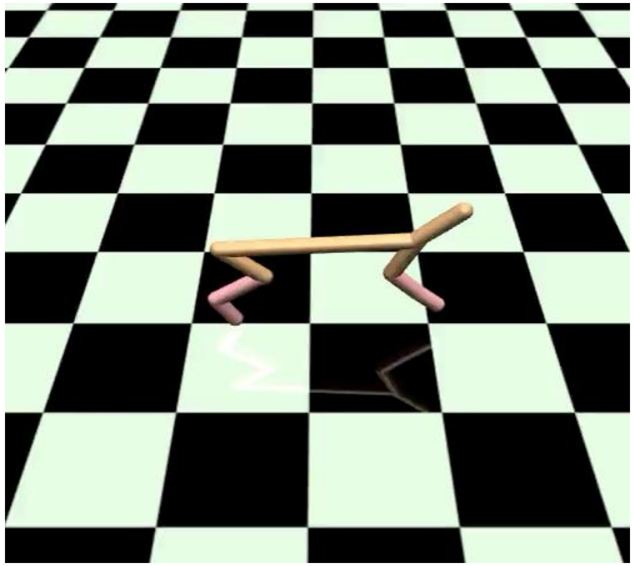


Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Trajectory-ranked Reward Extrapolation (T-REX)



$\prec \cdots \prec$

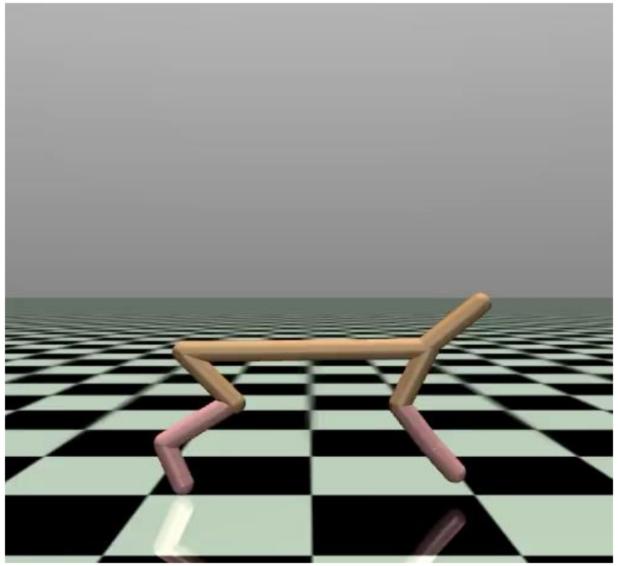


Reward Function

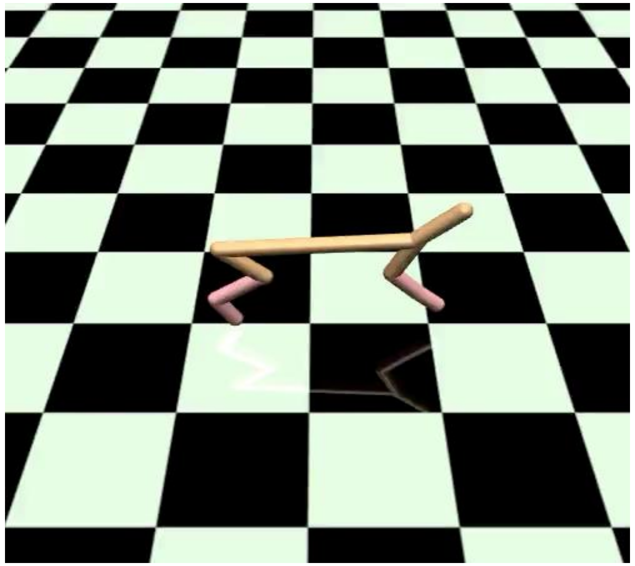
Pre-ranked demonstrations

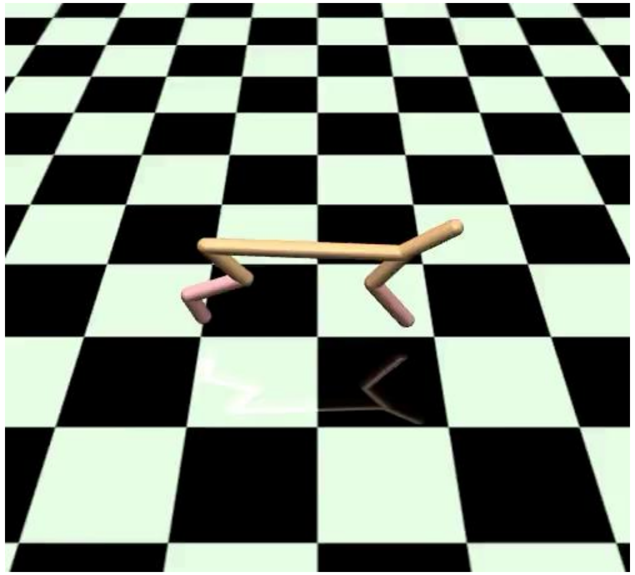
Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Trajectory-ranked Reward Extrapolation (T-REX)



$\prec \cdots \prec$





Pre-ranked demonstrations

T-REX Policy

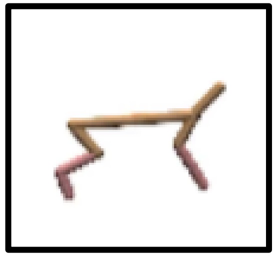
Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via IRL from Observations." ICML 2019

Reward Function

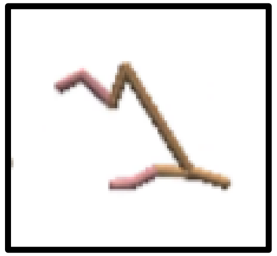
$$R_\theta : S \to \mathbb{R}$$

Examples of S:

Current Robot Joint
Angles and Velocities



$\to 0.5$



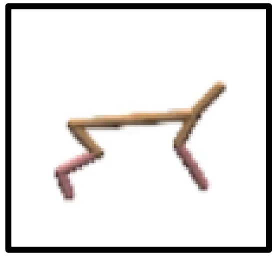
$\to -0.7$

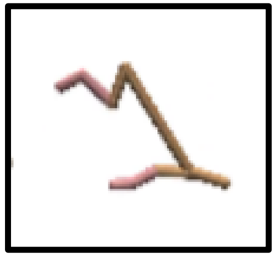
Reward Function

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

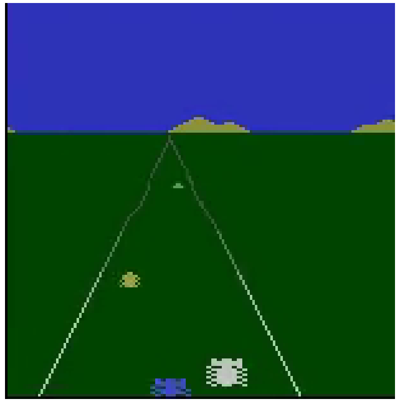
Examples of S:

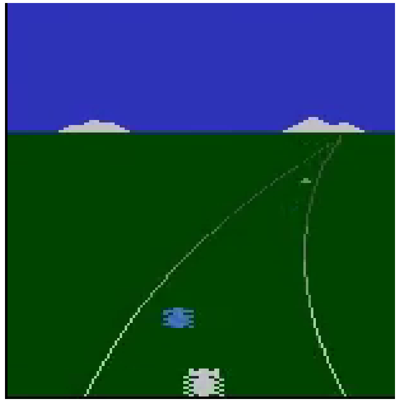
Current Robot Joint Angles and Velocities

 $\rightarrow 0.5$

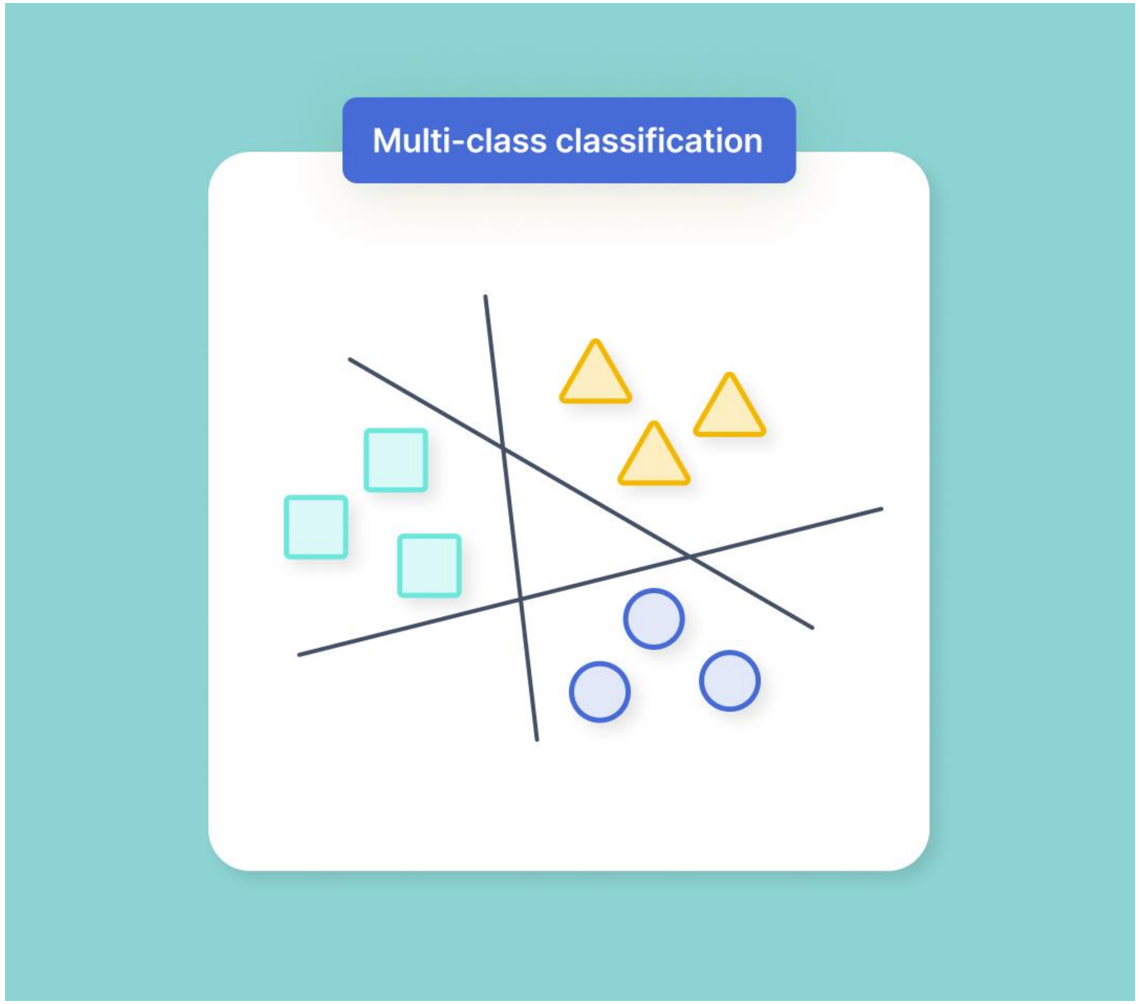
 $\rightarrow -0.7$

Short Sequence of Images

 $\rightarrow 0.9$

 $\rightarrow -1.2$

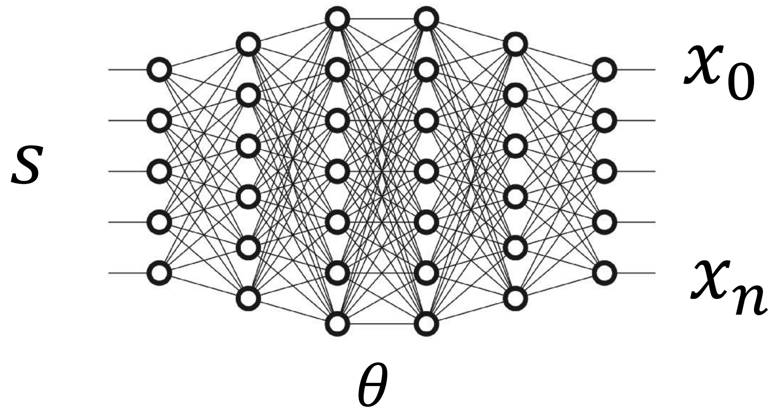
Binary Classification and the Cross Entropy Loss



https://www.v7labs.com/blog/cross-entropy-loss-guide

Flashback: How should we parameterize our policy?

- We need to be able to do two things:
  - Sample actions $a_t \sim \pi_\theta(\cdot\,|s_t)$
  - Compute log probabilities $\log \pi_\theta(a_t|s_t)$
- Categorical (classifier over discrete actions)
  - Typically, you output a value x_i for each action (class) and then the probability is given by a softmax equation



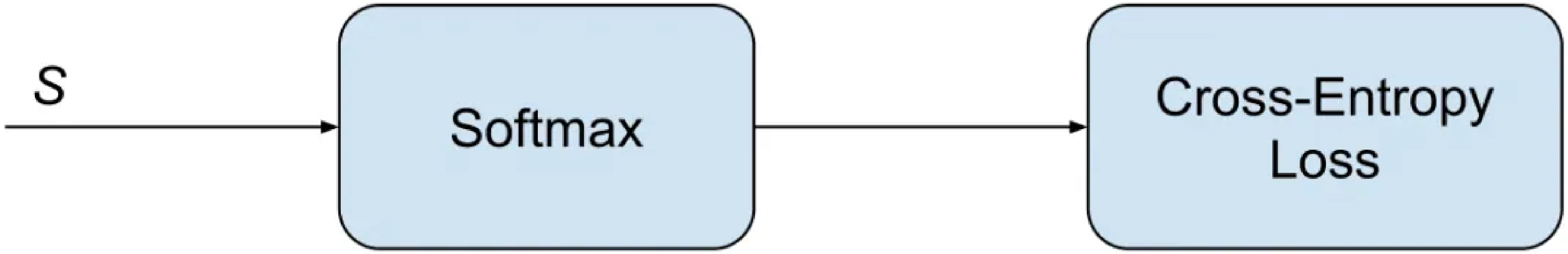
$$\pi_\theta(a_i|s) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Cross Entropy

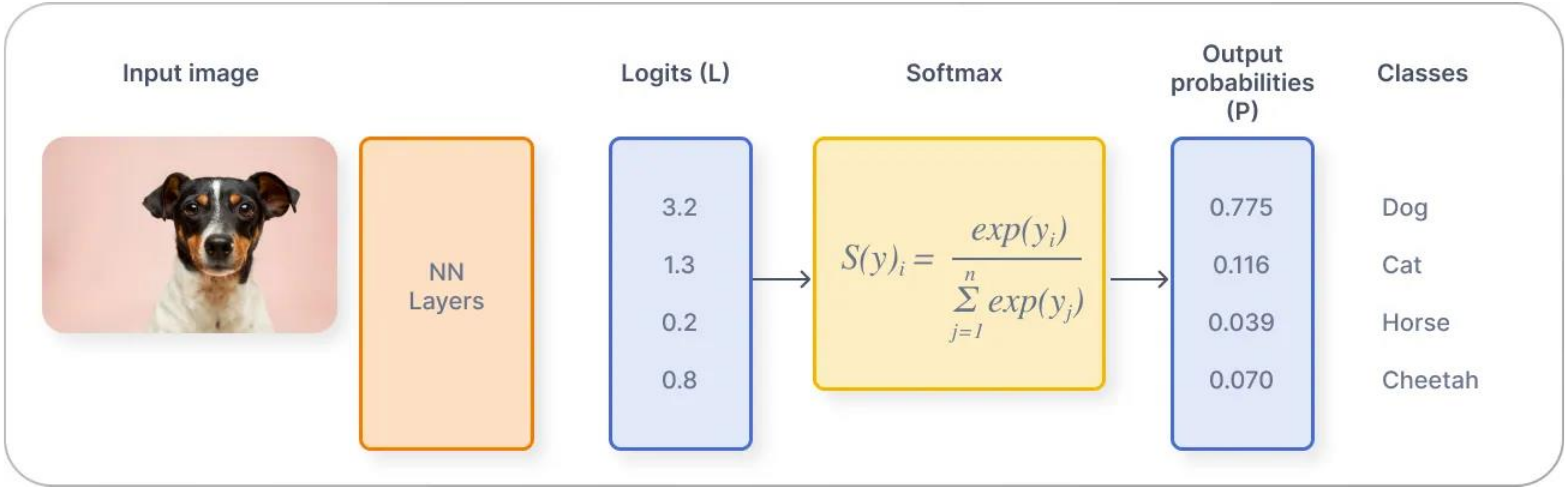
True probability distribution (one-shot)

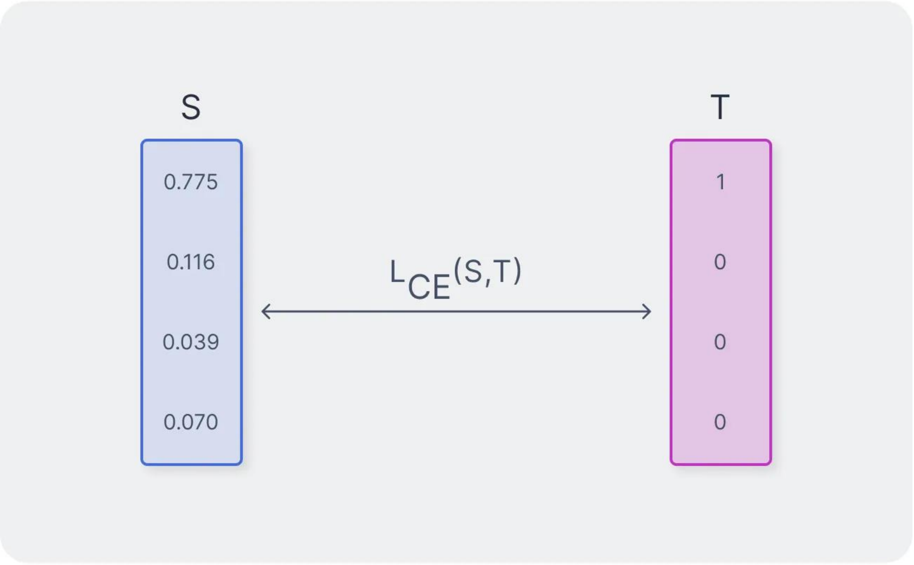
$$H(p, q) = -\sum_{x \epsilon classes} p(x) log\, q(x)$$

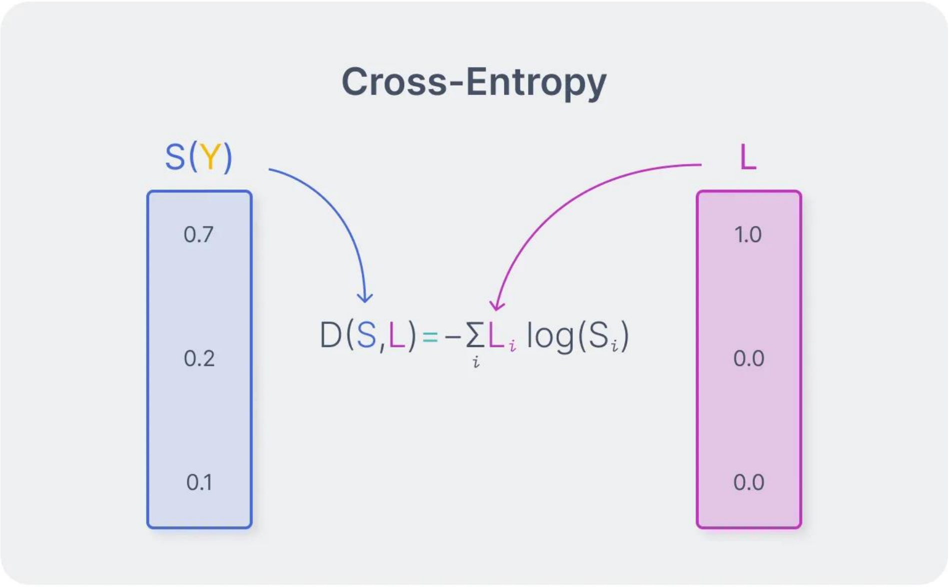
Your model's predicted probability distribution



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


Input image
NN Layers
Logits (L)
3.2
1.3
0.2
0.8
Softmax
$S(y)_i = \frac{exp(y_i)}{\sum_{j=1}^{n} exp(y_j)}$
Output probabilities (P)
0.775
0.116
0.039
0.070
Classes
Dog
Cat
Horse
Cheetah


S
0.775
0.116
0.039
0.070
$L_{CE}(S,T)$
T
1
0
0
0


Cross-Entropy
S(Y)
0.7
0.2
0.1
$D(S,L) = -\sum_i L_i \log(S_i)$
L
1.0
0.0
0.0

Trajectory-ranked Reward Extrapolation (T-REX)

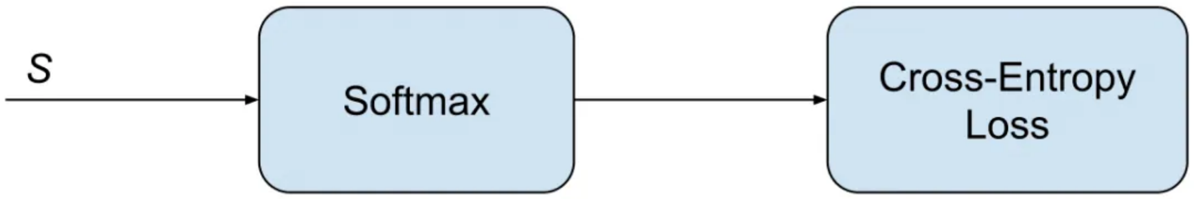
$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

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

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)

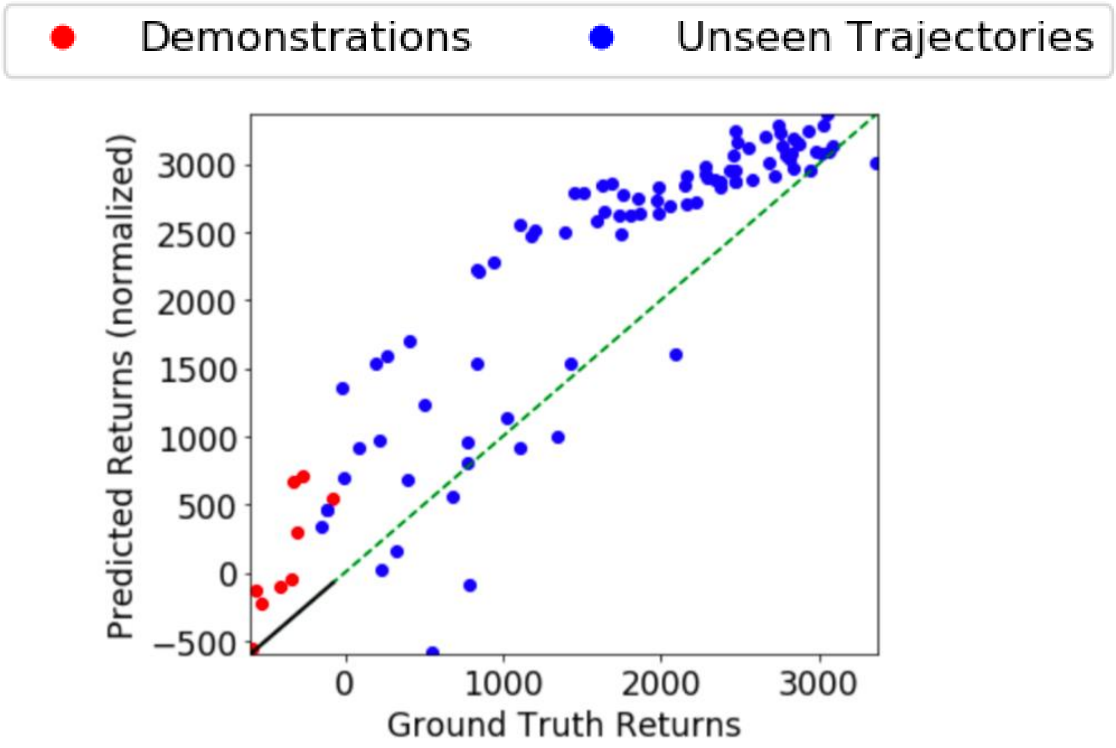
$$\boxed{\tau_1} \prec \boxed{\tau_2} \prec \cdots \prec \tau_T$$

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

Minimize cross-entropy loss

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

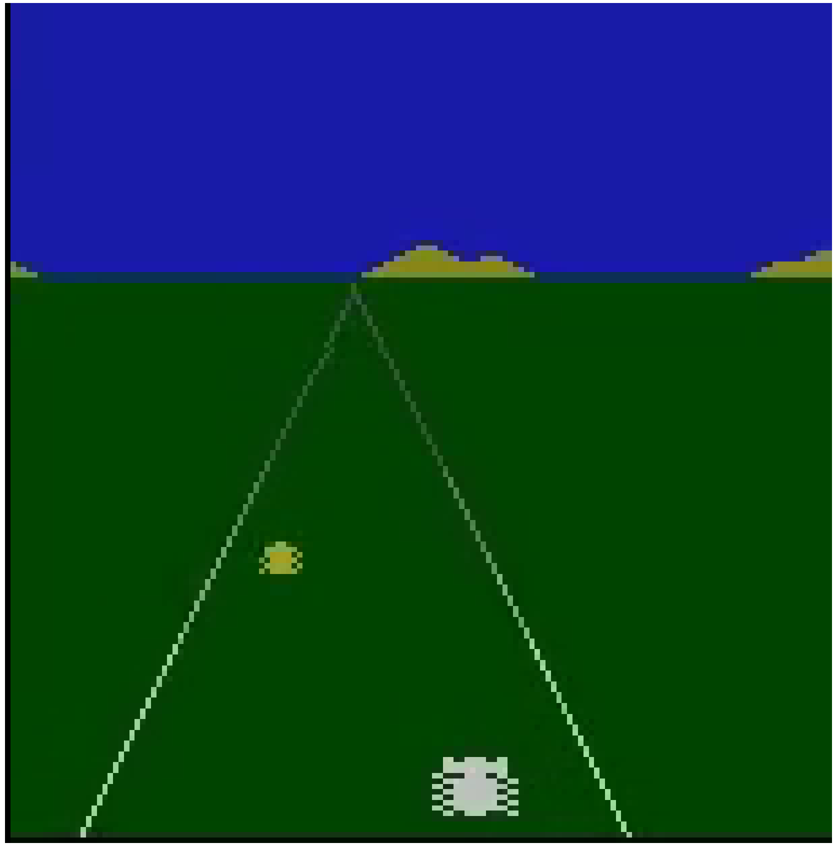
Reward Extrapolation



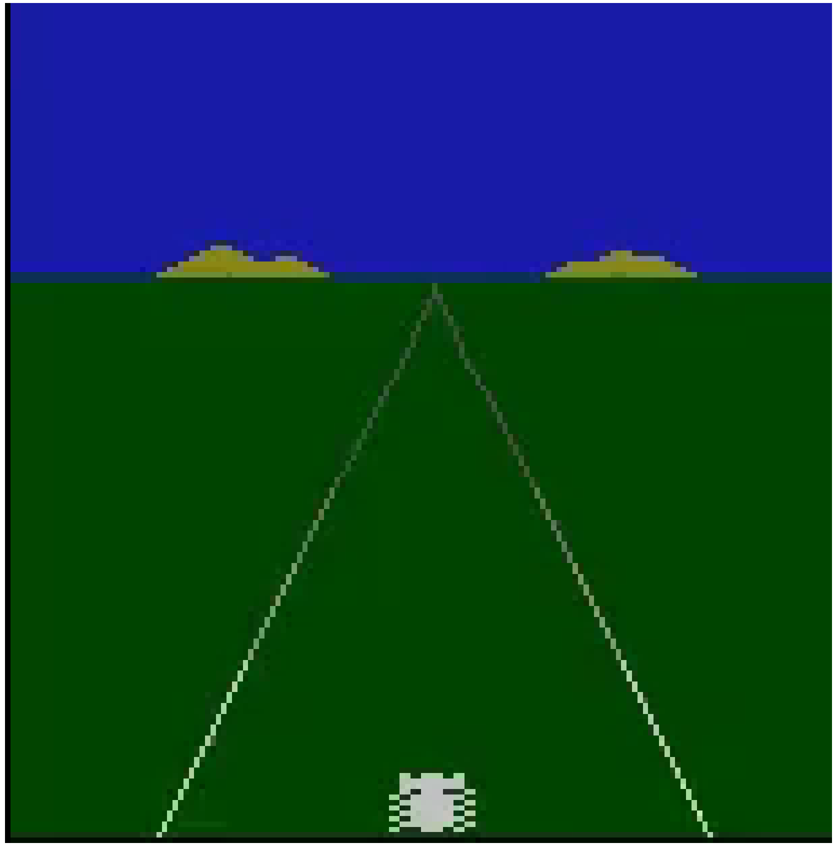
T-REX can extrapolate beyond the performance of the best demo

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

“Autonomous Driving” in Atari



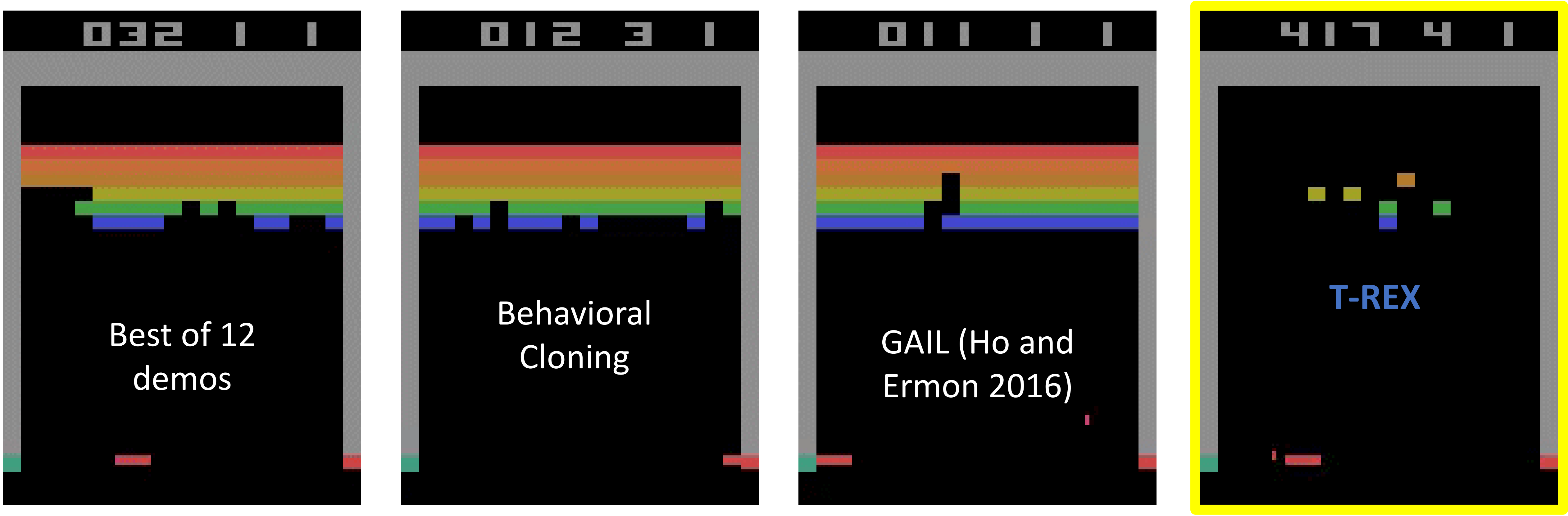
Best demo (Score = 84)



T-REX (Score = 520)

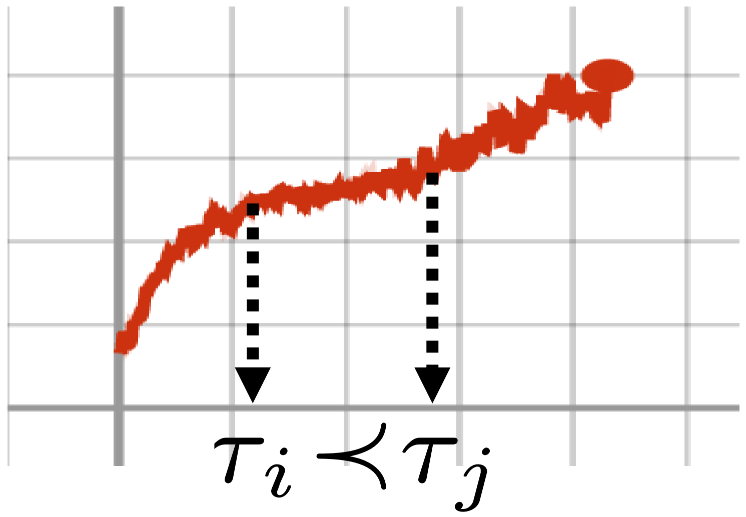
Uses only 12 ranked demonstrations

Atari Breakout

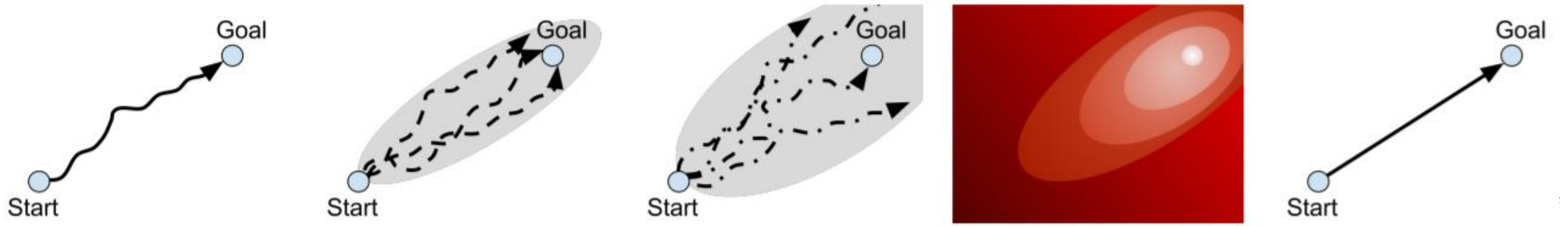


What if you don't have explicit preference labels?

Learning from a learner [ICML'19]

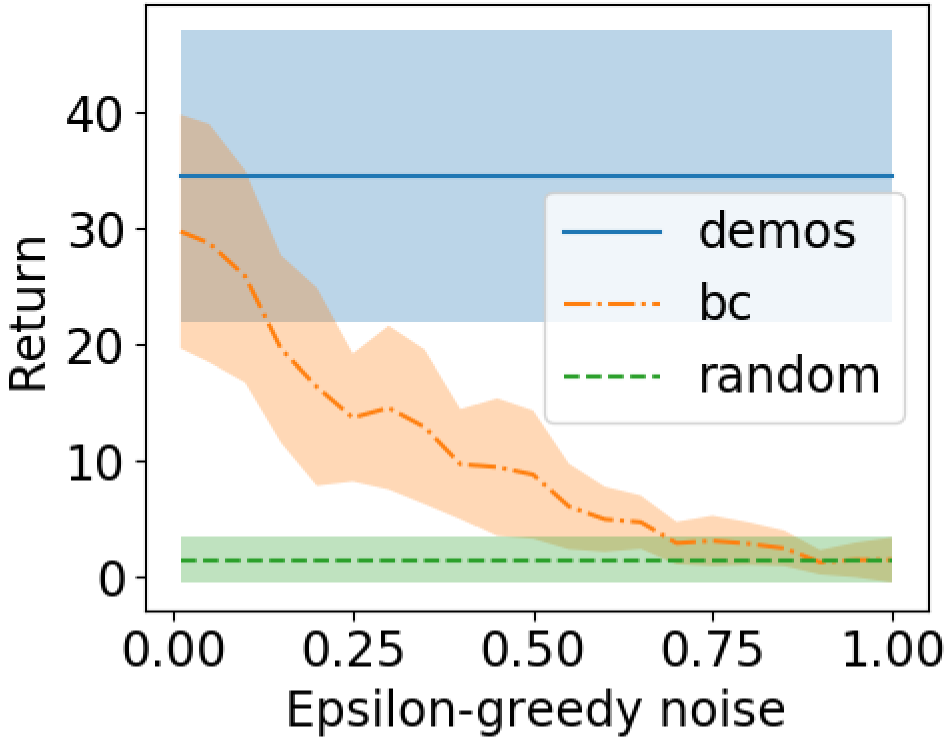


Automatic preference label generation [CoRL'20]



Automatic Rankings via Noise Injection

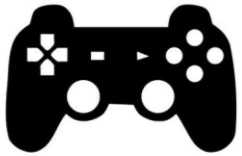
- Assumption: Demonstrator is significantly better than a purely random policy.
- Provides automatic rankings as noise increases.
- Generates a large diverse set of ranked demonstrations



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

Disturbance-based Reward Extrapolation (D-REX)

Behavioral Cloning

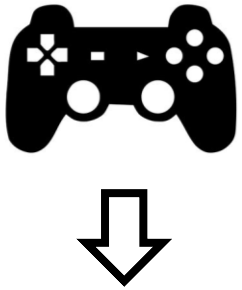


$$\pi_{\mathrm{BC}}$$

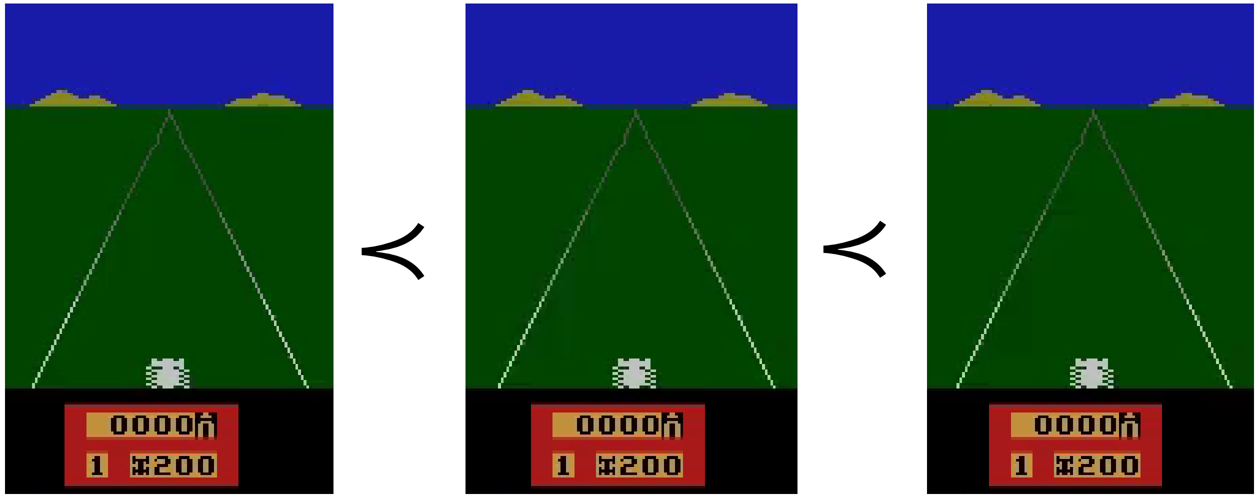
Brown et al. “Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations." CoRL 2019

Disturbance-based Reward Extrapolation (D-REX)

Behavioral Cloning → Automatic Rankings via Noise Injection



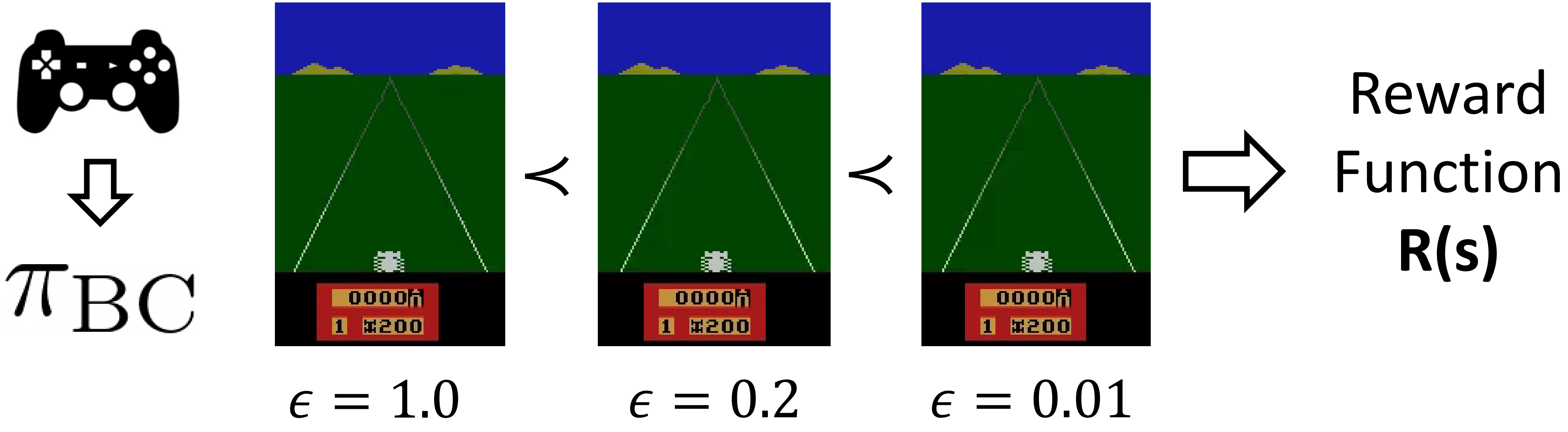
π_{BC}



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

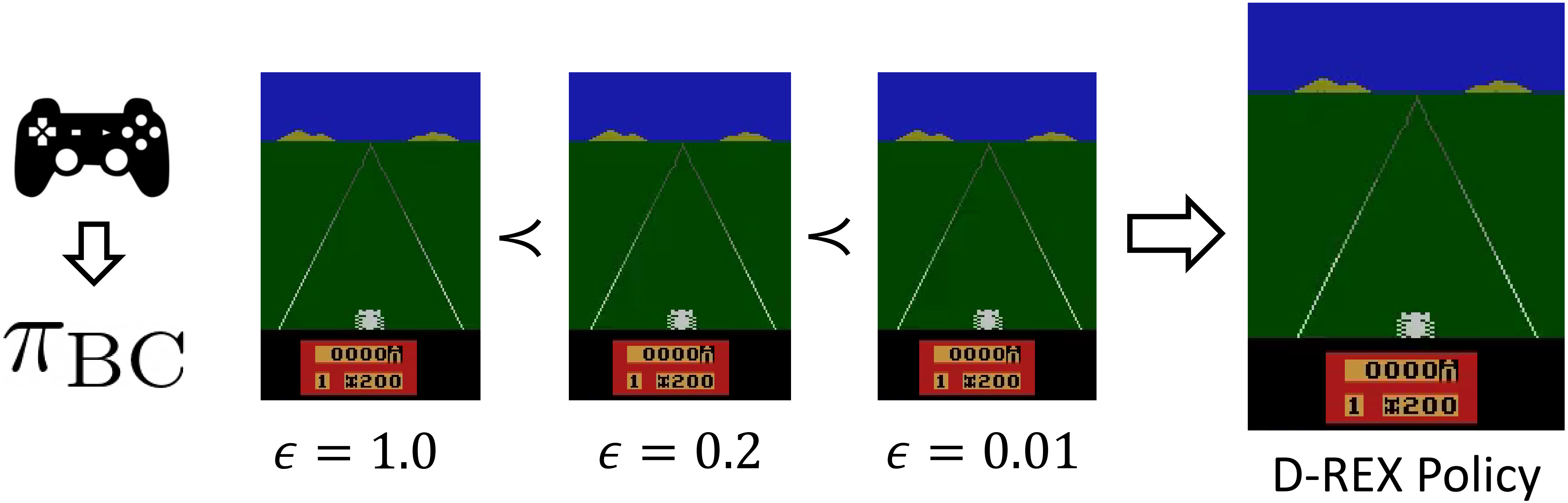
Disturbance-based Reward Extrapolation (D-REX)

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



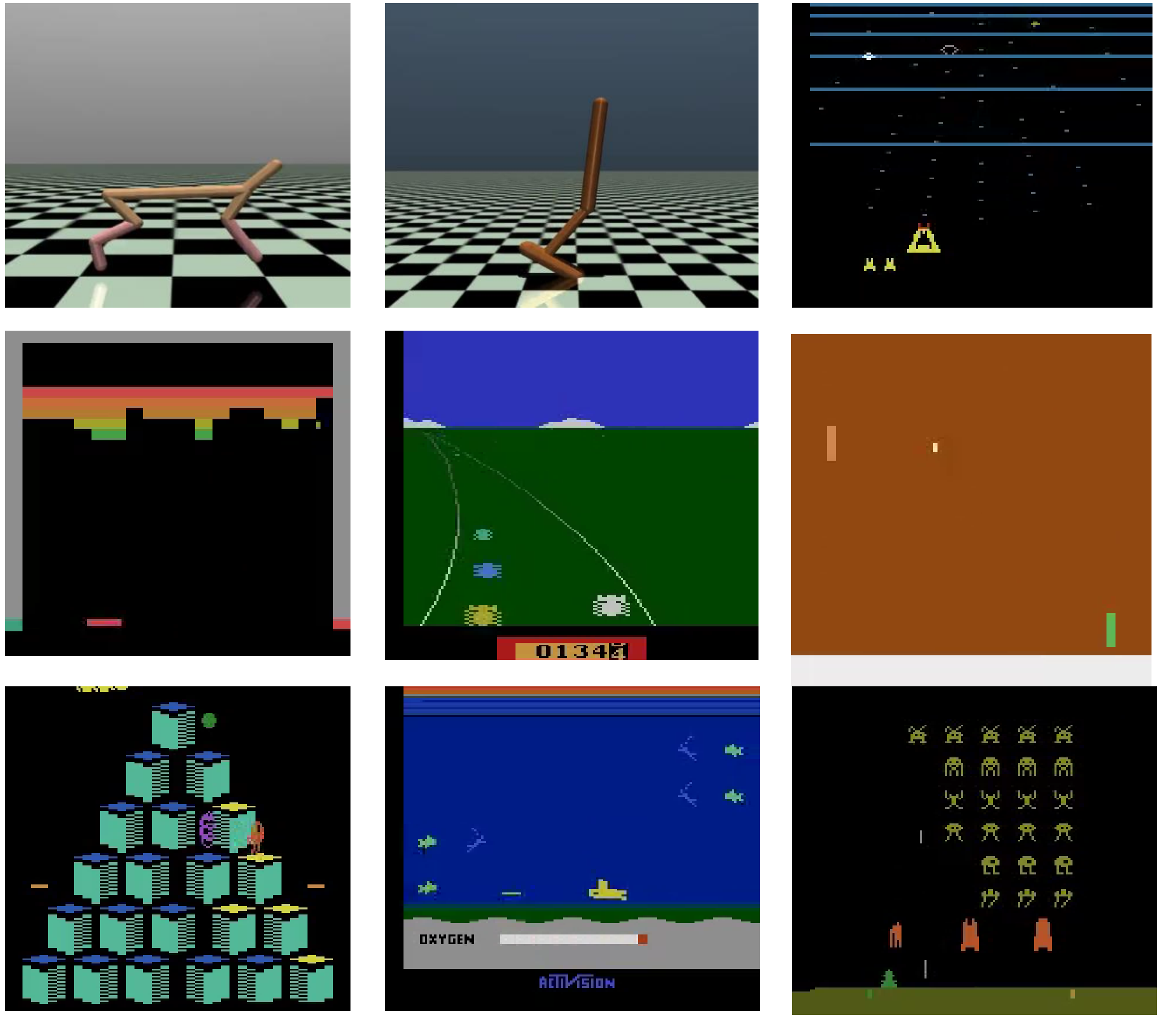
Disturbance-based Reward Extrapolation (D-REX)

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

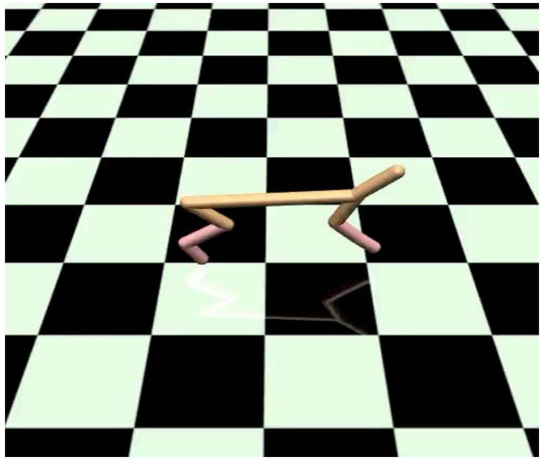


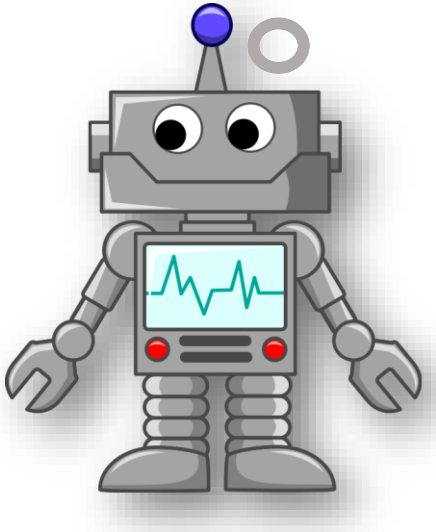
Experiments

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



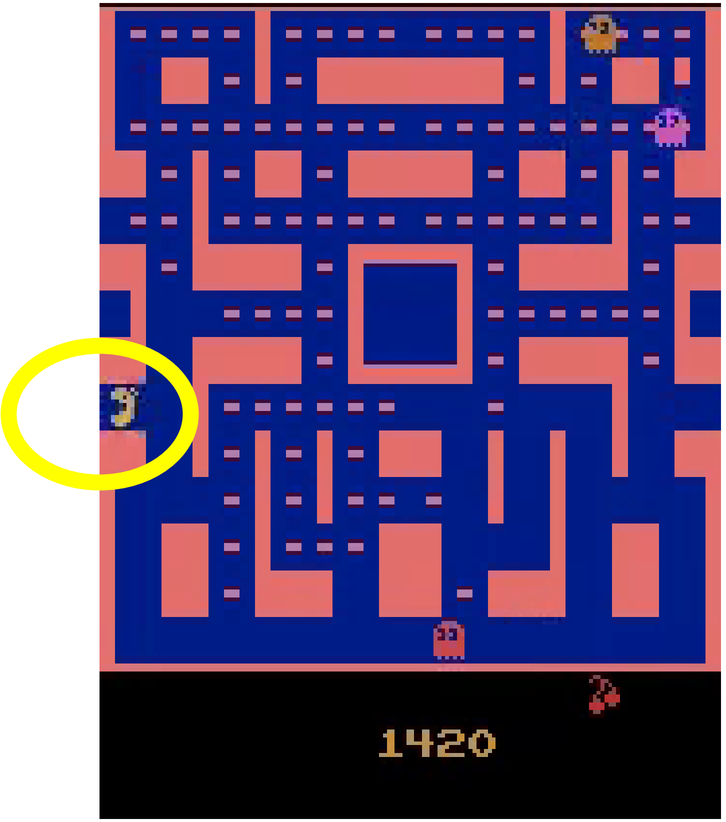


AI systems can **efficiently** infer human intent from **suboptimal demonstrations**.

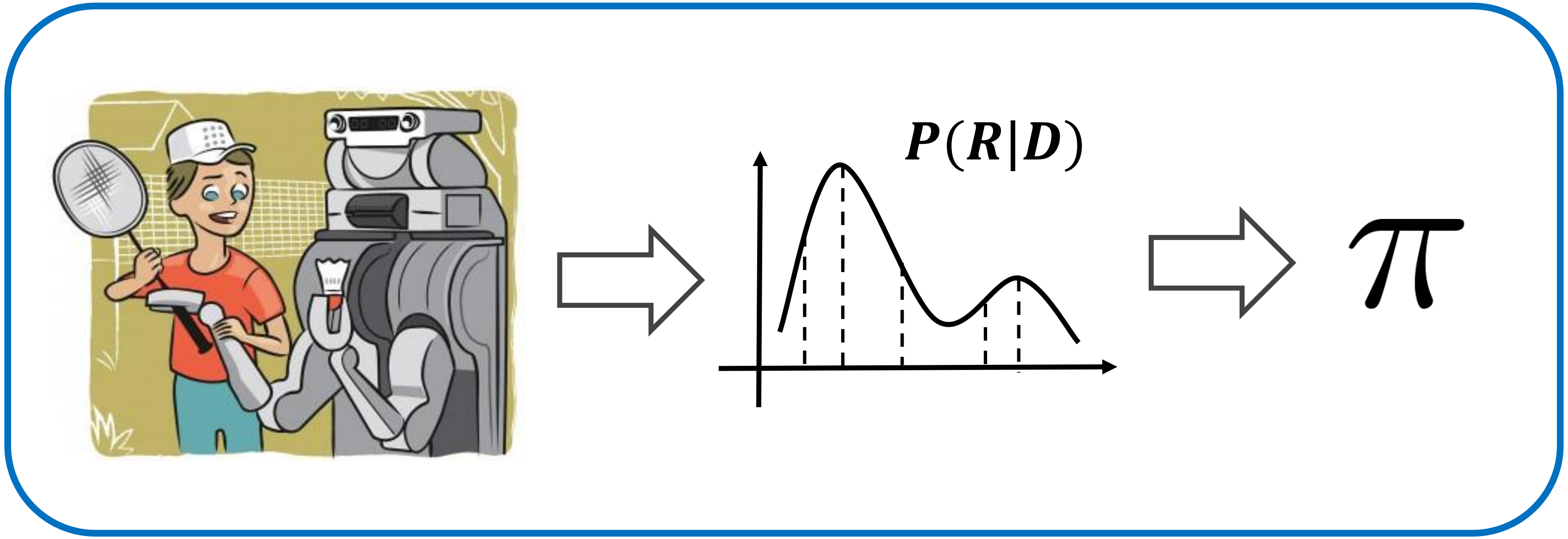
T-REX only learns a maximum likelihood estimate of the reward function.

 $\Rightarrow R \Rightarrow \pi$

Reward Hacking



- Overfit to spurious correlations
- No consideration of alternative hypotheses


$P(R|D)$
π

Next time: LLMs and ChatGPT