BOSCH

TEXAS
The University of Texas at Austin

Massachusetts Institute of Technology

The Perils of Trial-and-Error Reward Design: Misdesign Through Overfitting and Invalid Task Specifications


Serena Booth


Brad Knox


Julie Shah


Scott Niekum


Peter Stone


Alessandro Allievi

AAAI 2023

Imagine you want to design a new environment for using or benchmarking RL.

Imagine you want to design a new environment for using or benchmarking RL.

How do you approach this?

A trial-and-error process

Step 1: Design a candidate MDP $< S, A, T, \gamma, D_0, r >$

A trial-and-error process

Step 1: Design a candidate MDP $< S, A, T, \gamma, D_0, r >$

Step 2: Pick an RL algorithm for testing

A trial-and-error process

Step 1: Design a candidate MDP $< S, A, T, \gamma, D_0, r >$

Step 2: Pick an RL algorithm for testing

Step 3: Learn a policy $\pi(a|s)$

A trial-and-error process

Step 1: Design a candidate MDP $< S, A, T, \gamma, D_0, r >$

Step 2: Pick an RL algorithm for testing

Step 3: Learn a policy $\pi(a|s)$

Step 4: If the policy isn't right, update the MDP
(especially the reward function) and repeat

This trial-and-error process is **common**.

We surveyed 24 expert RL practitioners; 92% used trial-and-error to design their most recent reward function.

“The reward signal is your way of communicating to the agent what you want achieved, not *how* you want it achieved”

- Sutton & Barto

For a Dyna-Q+ agent, Sutton & Barto replace the reward function r with $r + \kappa\sqrt{\tau}$.

(This additional term encourages exploration.)

Why does reward design practice matter?

Why does reward design practice matter?

Why be concerned about trial-and-error?

A Known Concern: Unsafe Shaping

$$R'(s,a,s') = R(s,a,s') + F(s,a,s')$$

$$F(s,a,s') = \gamma\Phi(s') - \Phi(s)$$

$$\Phi : S \to \mathbb{R}$$

Potential-based shaping is known to be safe*, meaning optimal policies are unchanged.

* *Under some assumptions*

Ng, Harada, and Stuart, 1999

A Known Concern: Unsafe Shaping

$$R'(s,a,s') = R(s,a,s') + F(s,a,s')$$

$$F(s,a,s') = \gamma\Phi(s') - \Phi(s)$$

$$\Phi : S \to \mathbb{R}$$

Potential-based shaping is known to be safe*, meaning optimal policies are unchanged.

But trial-and-error reward shaping is typically **not** potential-based.

* *Under some assumptions*

Ng, Harada, and Stuart, 1999

A Known Concern: Misspecification

Reward functions are often wrong and/or underspecified.

Amodei et al., 2016; Knox et al., 2021

A Known Concern: Misspecification

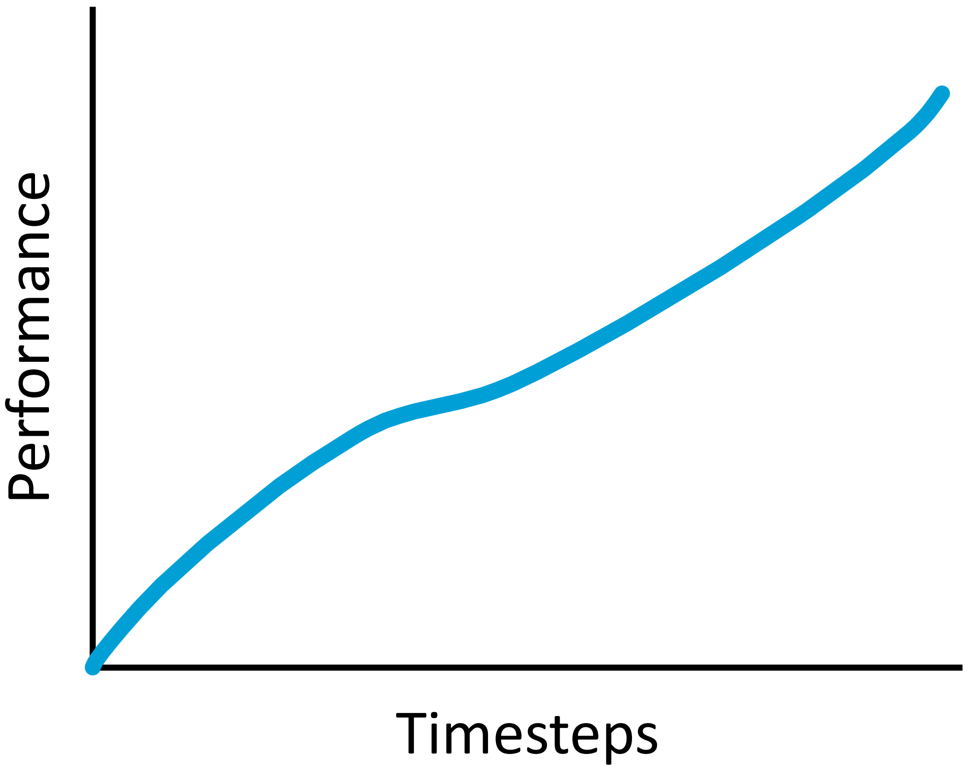
Reward functions are often wrong and/or underspecified.

Does trial-and-error reward design make this problem worse?

Amodei et al., 2016; Knox et al., 2021

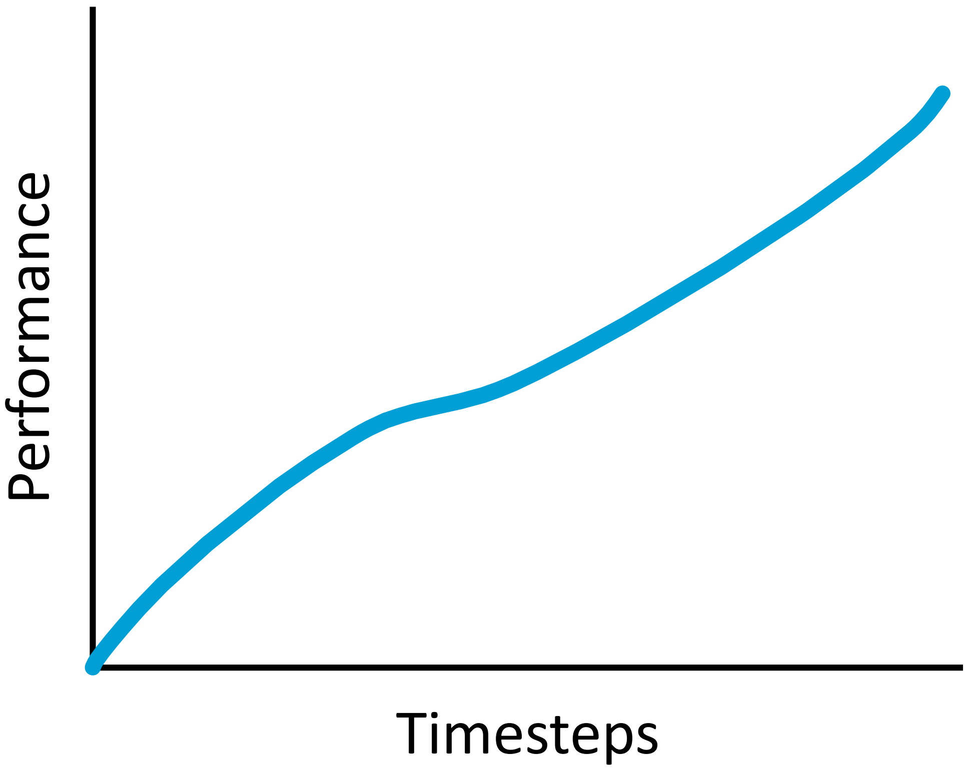
A New Concern: Overfitting

A New Concern: Overfitting

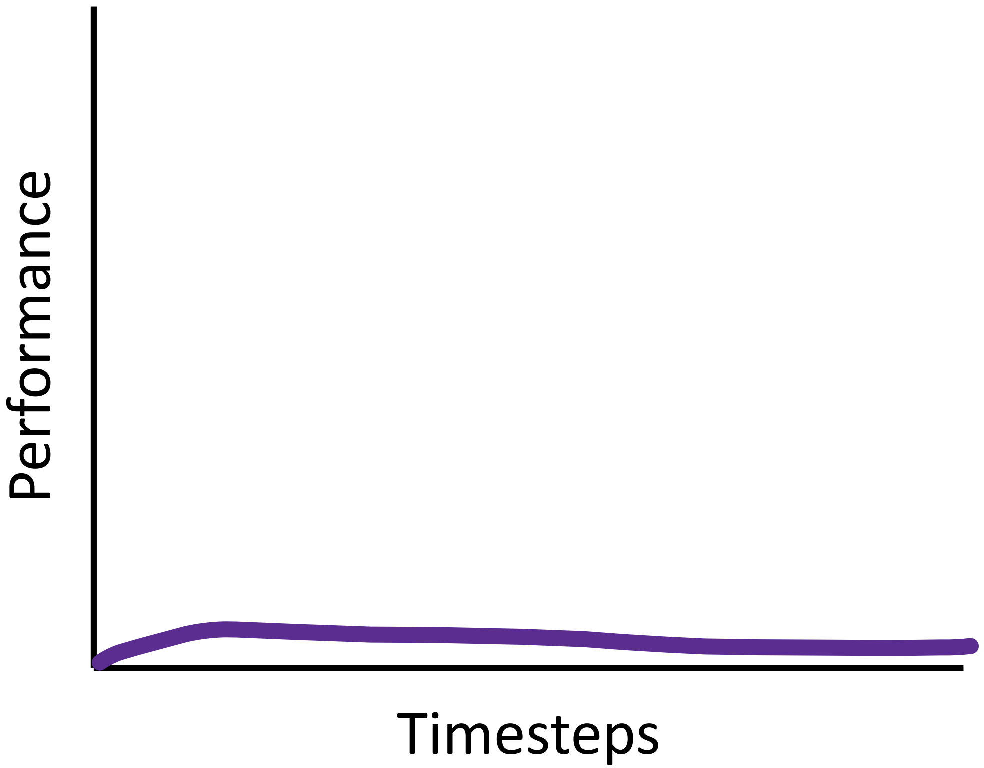


Test Algorithm

A New Concern: Overfitting



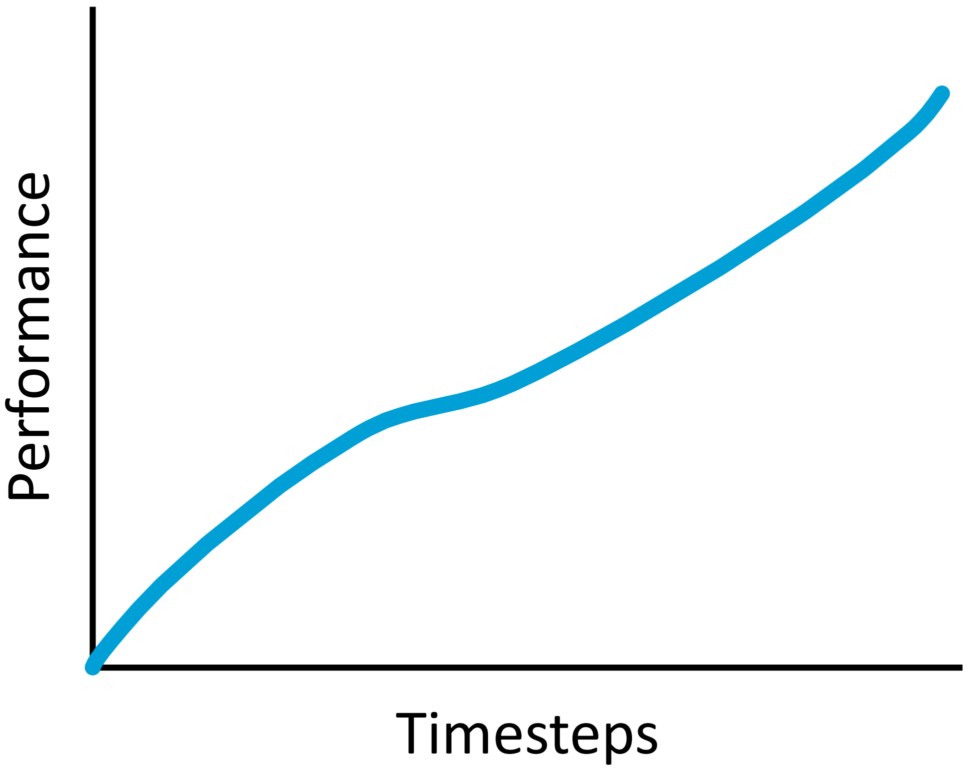
Test Algorithm



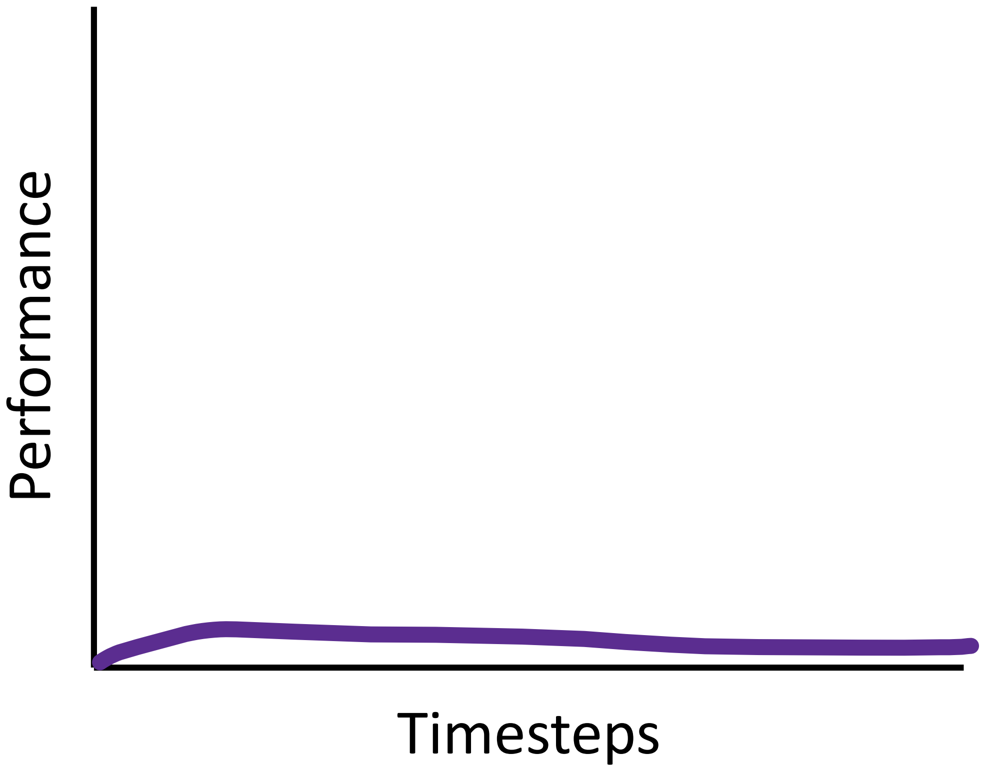
Other Algorithms

A New Concern: Overfitting

Can reward functions be *overfit* to learning algorithms and hyperparameters?



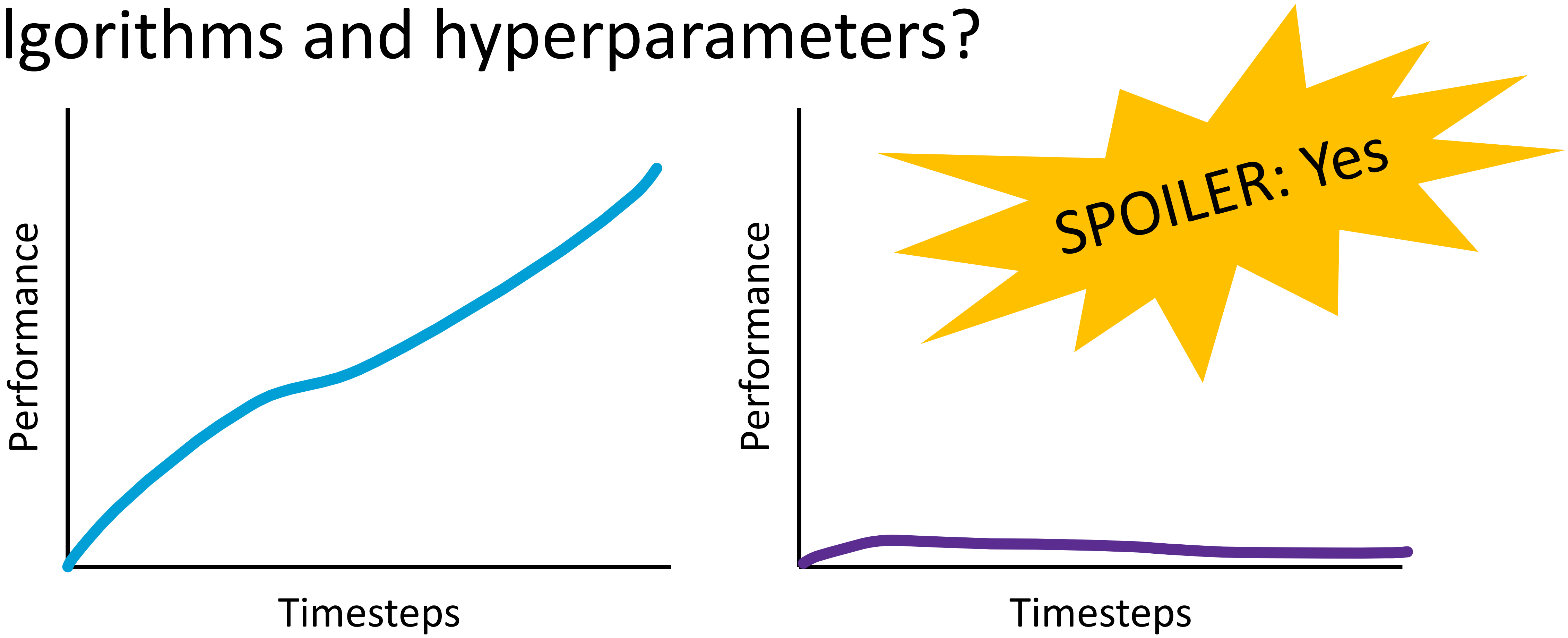
Test Algorithm



Other Algorithms

A New Concern: Overfitting

Can reward functions be *overfit* to learning algorithms and hyperparameters?



Test Algorithm   Other Algorithms

We study the implications of trial-and-error reward design.

We study the implications of
trial-and-error reward design.

We do so with both **computational studies** and **controlled observation user studies**.

On the Shoulders of Giants…

On the Shoulders of Giants…

Optimizing Rewards for Learning

Singh 2009, *Where Do Rewards Come From?*
Faust 2019, *Evolving Rewards to Automate Reinforcement Learning*

On the Shoulders of Giants…

Optimizing Rewards for Learning

Singh 2009, *Where Do Rewards Come From?*
Faust 2019, *Evolving Rewards to Automate Reinforcement Learning*

Reward Misdesign

Amodei 2016, *Concrete Problems in AI Safety*
Knox 2021, *Reward (Mis)Design for Autonomous Driving*

On the Shoulders of Giants…

Optimizing Rewards for Learning

Singh 2009, *Where Do Rewards Come From?*
Faust 2019, *Evolving Rewards to Automate Reinforcement Learning*

Reward Misdesign

Amodei 2016, *Concrete Problems in AI Safety*
Knox 2021, *Reward (Mis)Design for Autonomous Driving*

RL Reproducibility

Henderson 2018, *Deep Reinforcement Learning that Matters*
Engstrom 2019, *Implementation Matters in Deep RL*

On the Shoulders of Giants…

Optimizing Rewards for Learning

Singh 2009, *Where Do Rewards Come From?*
Faust 2019, *Evolving Rewards to Automate Reinforcement Learning*

Reward Misdesign

Amodei 2016, *Concrete Problems in AI Safety*
Knox 2021, *Reward (Mis)Design for Autonomous Driving*

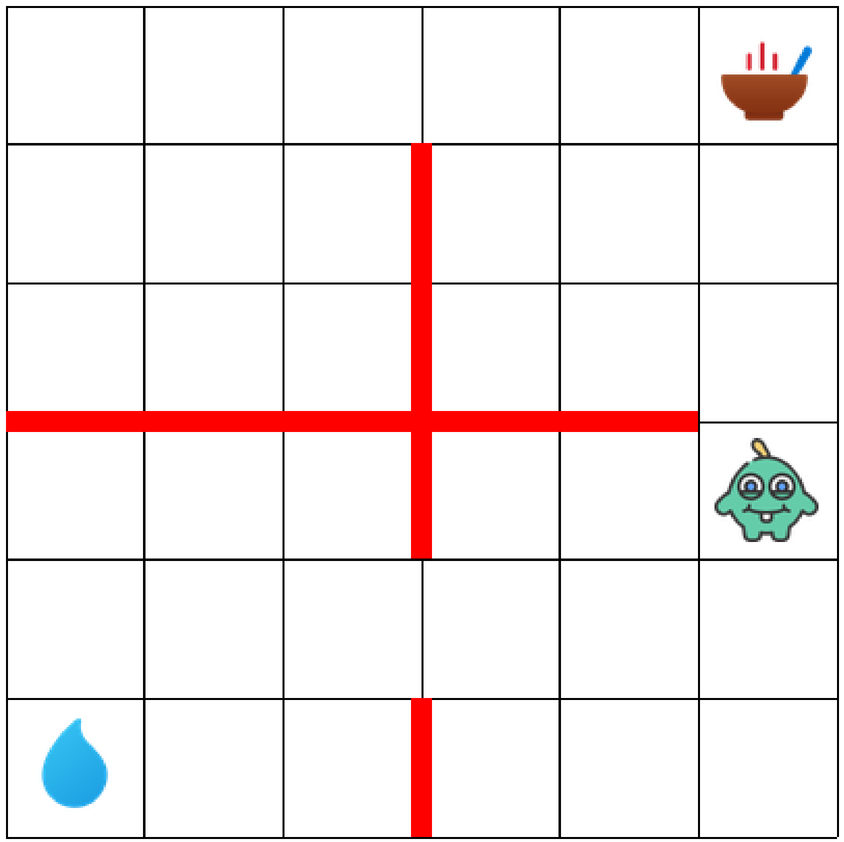
RL Reproducibility

Henderson 2018, *Deep Reinforcement Learning that Matters*
Engstrom 2019*, Implementation Matters in Deep RL*

Reward Function Inference

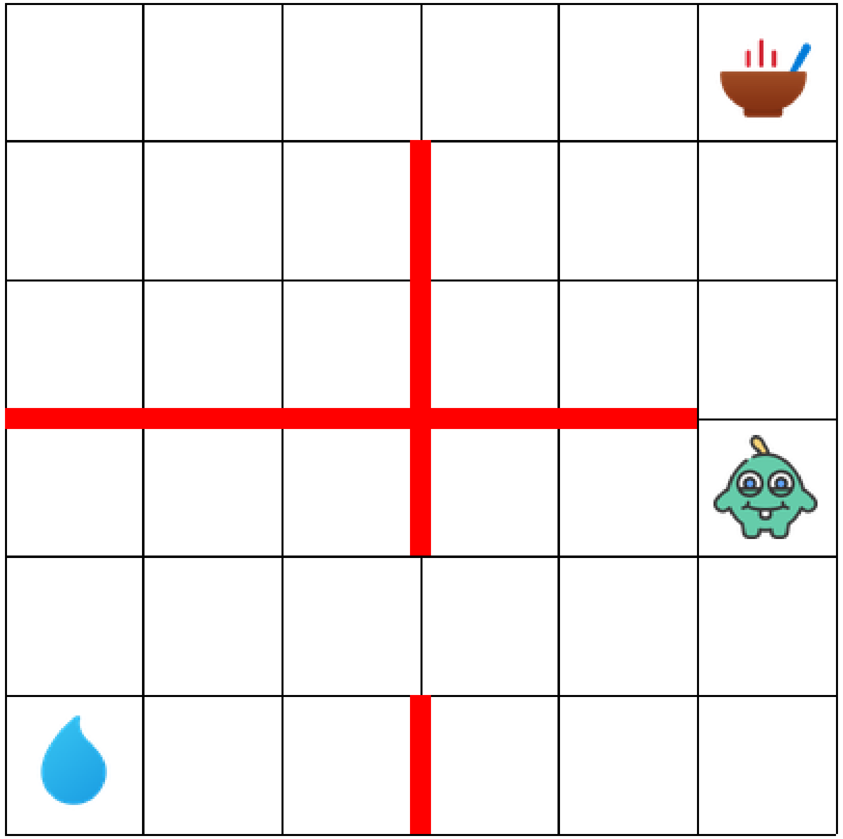
Hadfield-Menell 2016, *Inverse Reward Design*
He 2021*, Assisted Robust Reward Design*

Hungry Thirsty Domain



Singh et al., 2009, Where Do Rewards Come From?

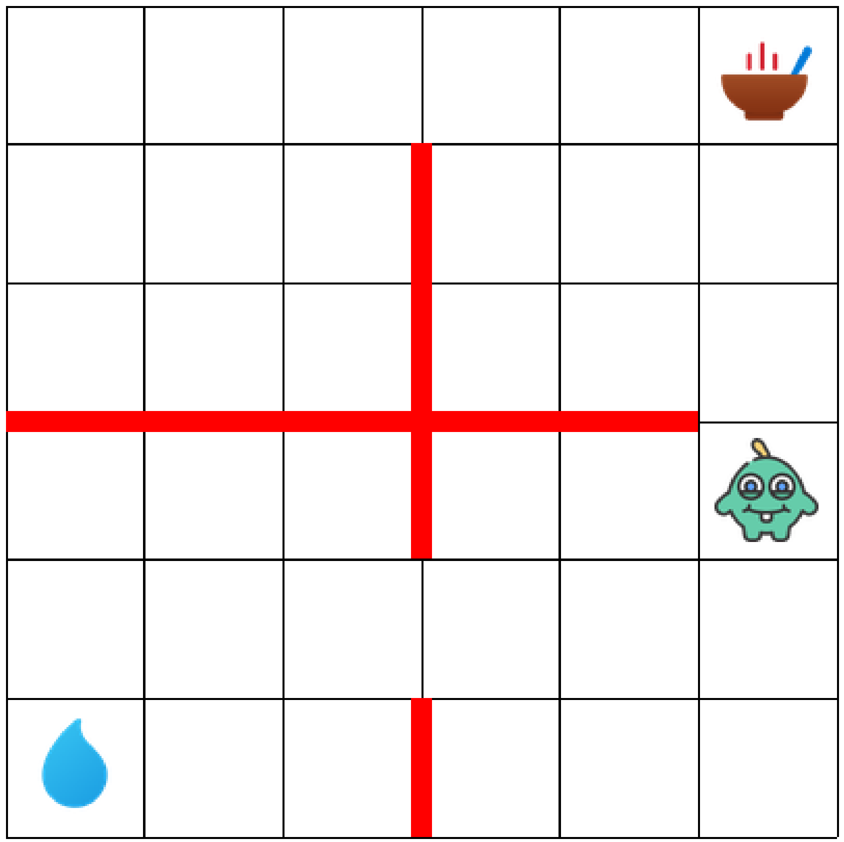
Hungry Thirsty Domain



Food in one random corner; water in another.

Singh et al., 2009, Where Do Rewards Come From?

Hungry Thirsty Domain

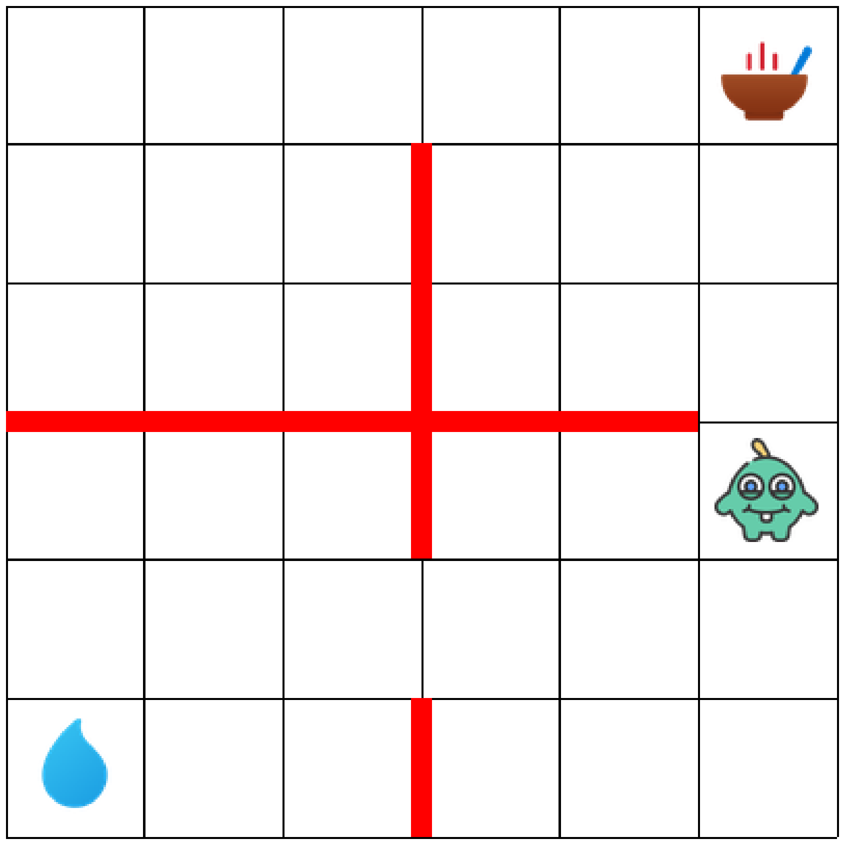


Food in one random corner; water in another.

The goal is to eat as much as possible, but the agent can only eat if not thirsty.

Singh et al., 2009, Where Do Rewards Come From?

Hungry Thirsty Domain



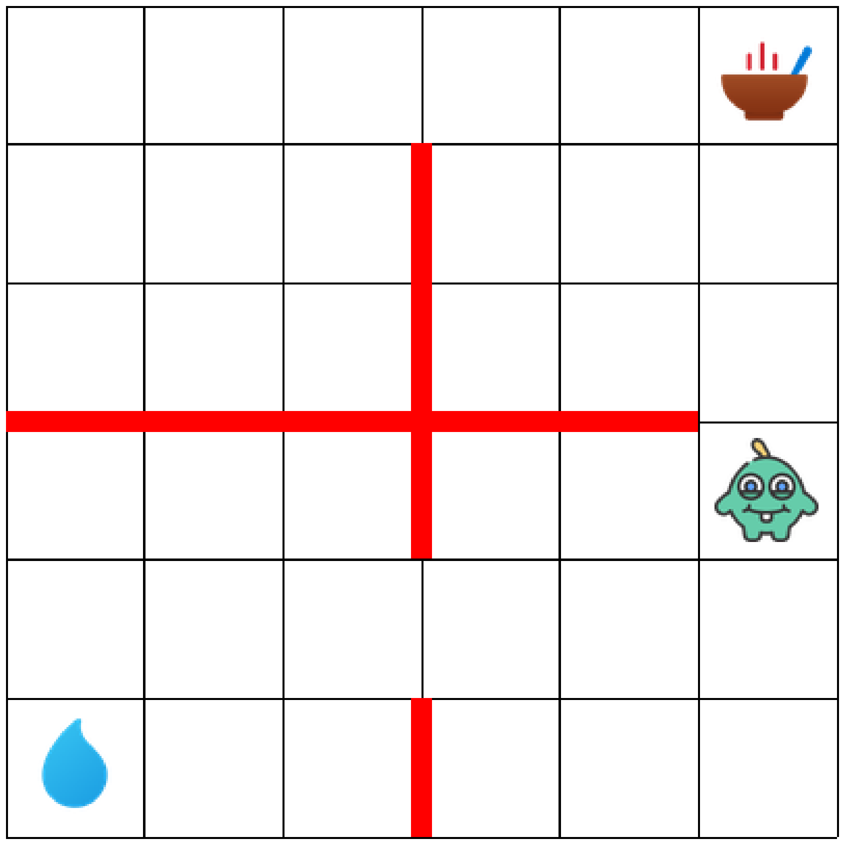
Food in one random corner; water in another.

The goal is to eat as much as possible, but the agent can only eat if not thirsty.

If the agent drinks, it becomes not thirsty. If the agent doesn't drink, it becomes thirsty with 10% probability.

Singh et al., 2009, Where Do Rewards Come From?

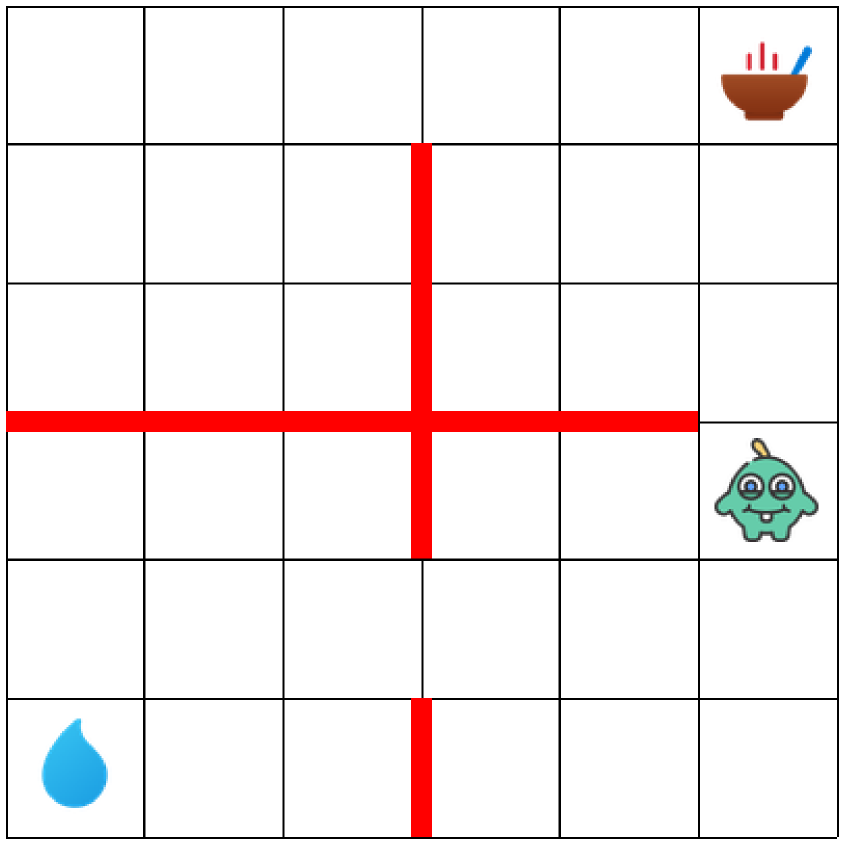
Hungry Thirsty Reward Functions



State consists of x-y coordinates, hunger status (H), and thirst status (T).

Singh et al., 2009, Where Do Rewards Come From?

Hungry Thirsty Reward Functions



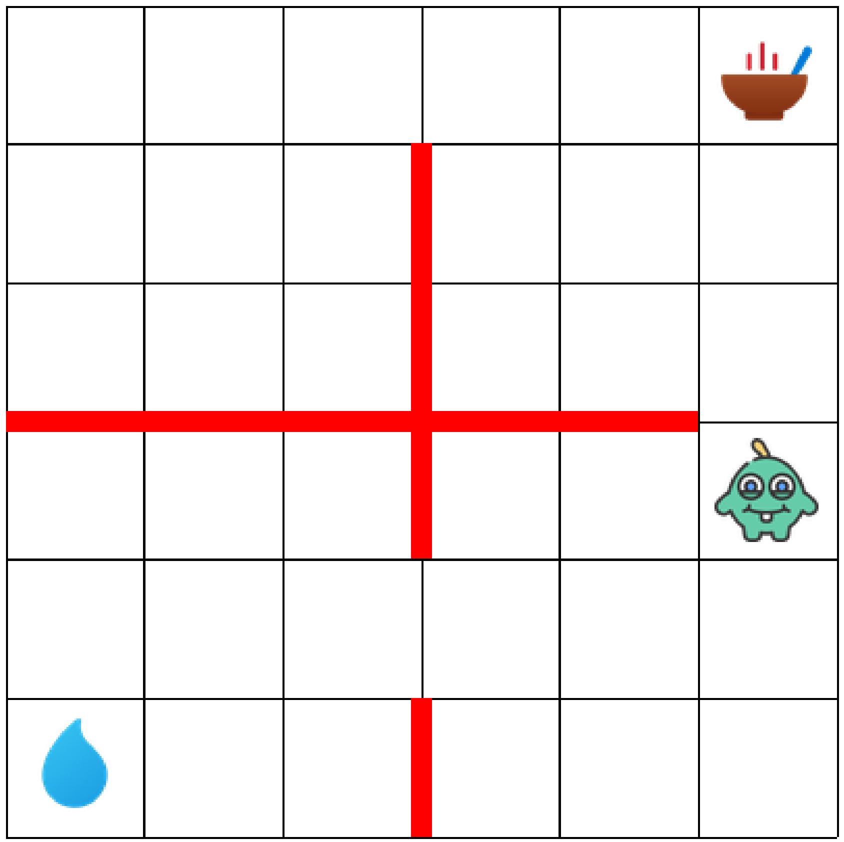
State consists of x-y coordinates, hunger status (H), and thirst status (T).

Unshaped reward function (sparse):

$$r(\neg \mathtt{H} \wedge \neg \mathtt{T}) = 1 \qquad r(\mathtt{H} \wedge \neg \mathtt{T}) = 0$$
$$r(\neg \mathtt{H} \wedge \mathtt{T}) = 1 \qquad r(\mathtt{H} \wedge \mathtt{T}) = 0$$

Singh et al., 2009, Where Do Rewards Come From?

Hungry Thirsty Reward Functions



State consists of x-y coordinates, hunger status (H), and thirst status (T).

Unshaped reward function (sparse):

$$r(\neg \mathtt{H} \land \neg \mathtt{T}) = 1 \qquad r(\mathtt{H} \land \neg \mathtt{T}) = 0$$
$$r(\neg \mathtt{H} \land \mathtt{T}) = 1 \qquad r(\mathtt{H} \land \mathtt{T}) = 0$$

Unsafely shaped reward function:

$$r(\neg \mathtt{H} \land \neg \mathtt{T}) = 0.5 \qquad r(\mathtt{H} \land \neg \mathtt{T}) = -0.01$$
$$r(\neg \mathtt{H} \land \mathtt{T}) = 1 \qquad r(\mathtt{H} \land \mathtt{T}) = -0.05$$

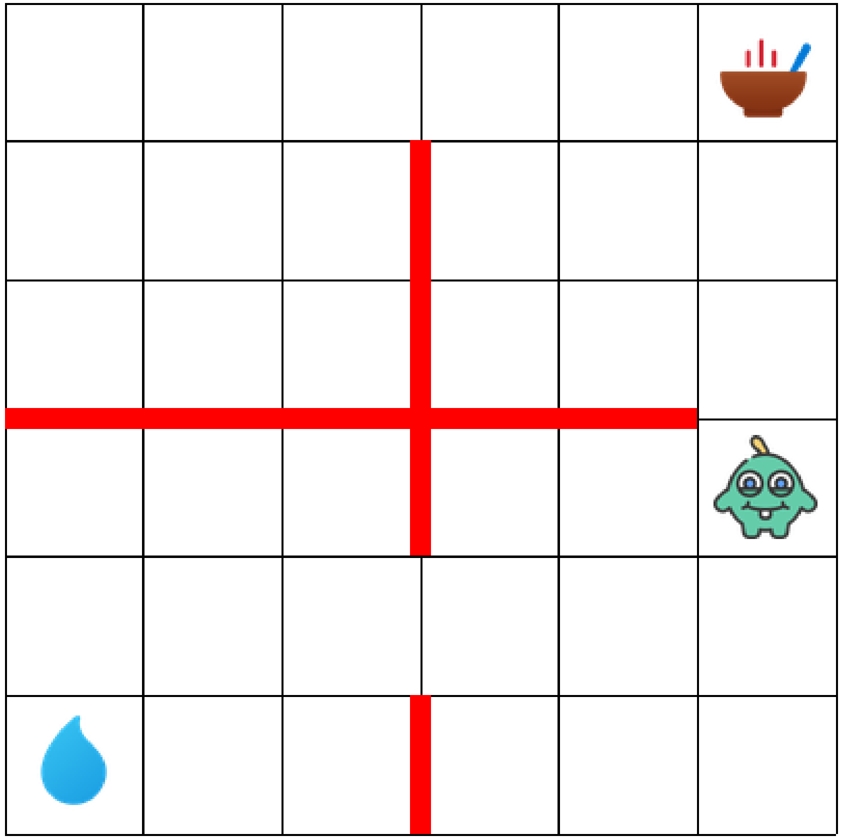
Singh et al., 2009, Where Do Rewards Come From?

Preliminaries

Define the *true* task performance metric:

$$\tau = (s_0, a_0, s_1, a_1, \cdots)$$
$$M : \tau \to \mathbb{R}$$

Hungry Thirsty True Performance Metric



True performance metric is the number of timesteps not hungry:

$$M(\tau) = \sum_{s \in \tau} \mathbb{1}(\neg \mathrm{H} \in s)$$

Preliminaries

Let $\mathcal{D}$ be a distribution of learning contexts consisting of algorithms, hyperparameters, and/or environments.

Consider a sample of $\mathcal{D}$: $D_1 \sim \mathcal{D}$.

Preliminaries

Let $\mathcal{D}$ be a distribution of learning contexts consisting of algorithms, hyperparameters, and/or environments.

Consider a sample of $\mathcal{D}$: $D_1 \sim \mathcal{D}$.

We define a reward function r_1 to be overfit with respect to the sample D_1 if there exists another reward function r_2 such that:

Preliminaries

Let $\mathcal{D}$ be a distribution of learning contexts consisting of algorithms, hyperparameters, and/or environments.

Consider a sample of $\mathcal{D}$: $D_1 \sim \mathcal{D}$.

We define a reward function r_1 to be overfit with respect to the sample D_1 if there exists another reward function r_2 such that:

$$\mathbb{E}_{\tau \sim \pi_{r_1, D_1}}[M(\tau)] \ > \ \mathbb{E}_{\tau \sim \pi_{r_2, D_1}}[M(\tau)]$$

Preliminaries

Let $\mathcal{D}$ be a distribution of learning contexts consisting of algorithms, hyperparameters, and/or environments.

Consider a sample of $\mathcal{D}$: $D_1 \sim \mathcal{D}$.

We define a reward function r_1 to be overfit with respect to the sample D_1 if there exists another reward function r_2 such that:

$$\mathbb{E}_{\tau \sim \pi_{r_1, D_1}}[M(\tau)] > \mathbb{E}_{\tau \sim \pi_{r_2, D_1}}[M(\tau)]$$
$$\mathbb{E}_{\tau \sim \pi_{r_1, \mathcal{D}}}[M(\tau)] < \mathbb{E}_{\tau \sim \pi_{r_2, \mathcal{D}}}[M(\tau)]$$

A Practical Test for Overfitting

Consider a sample of $\mathcal{D}$: $D_1 \sim \mathcal{D}$.

Consider a **second** distribution sample: $D_2 \sim \mathcal{D}$.

A Practical Test for Overfitting

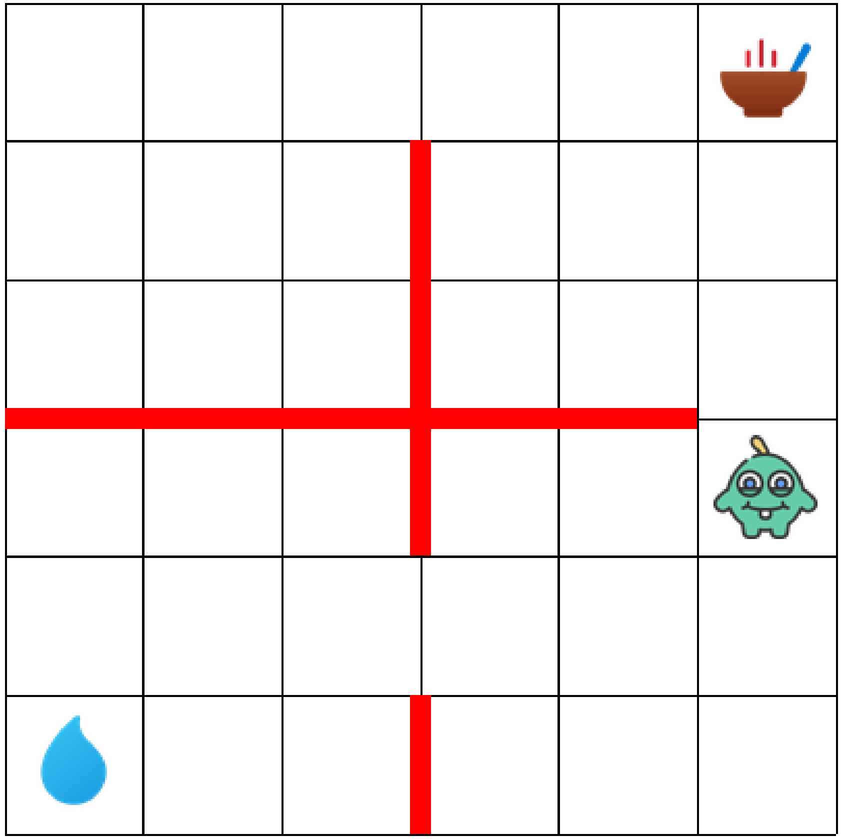
Consider a sample of $\mathcal{D}$: $D_1 \sim \mathcal{D}$.

Consider a **second** distribution sample: $D_2 \sim \mathcal{D}$.

We define a reward function r_1 to be overfit with respect to the sample D_1 if there exists another reward function r_2 such that:

$$\mathbb{E}_{\tau \sim \pi_{r_1, D_1}}[M(\tau)] \ > \ \mathbb{E}_{\tau \sim \pi_{r_2, D_1}}[M(\tau)]$$
$$\mathbb{E}_{\tau \sim \pi_{r_1, D_2}}[M(\tau)] \ < \ \mathbb{E}_{\tau \sim \pi_{r_2, D_2}}[M(\tau)]$$

Computational Experiments: Setup



Tested reward functions consist of:

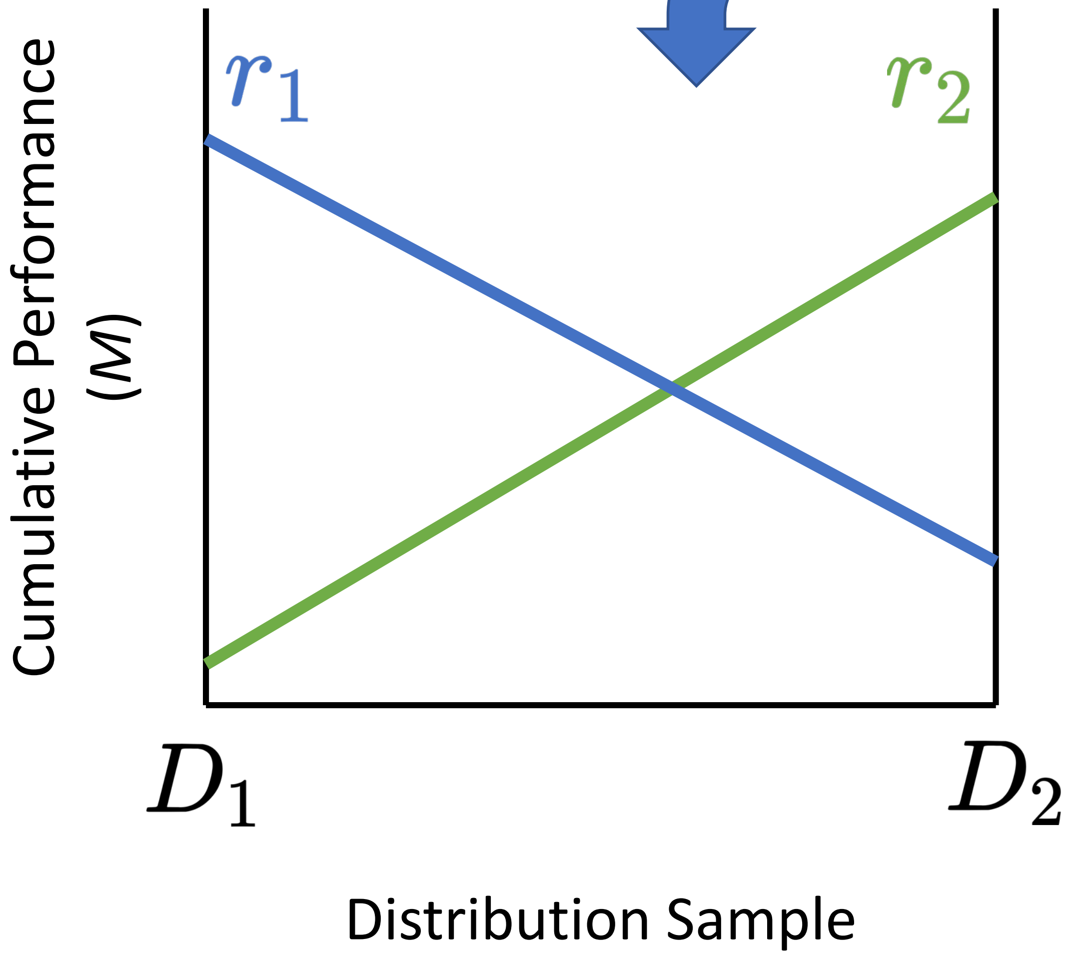
$$r(\neg \mathtt{H} \wedge \neg \mathtt{T}) = a \qquad r(\mathtt{H} \wedge \neg \mathtt{T}) = c$$
$$r(\neg \mathtt{H} \wedge \mathtt{T}) = b \qquad r(\mathtt{H} \wedge \mathtt{T}) = d$$

Where a, b, c, d ∈ [-1,1].

We test 5,196 different reward functions of this form.

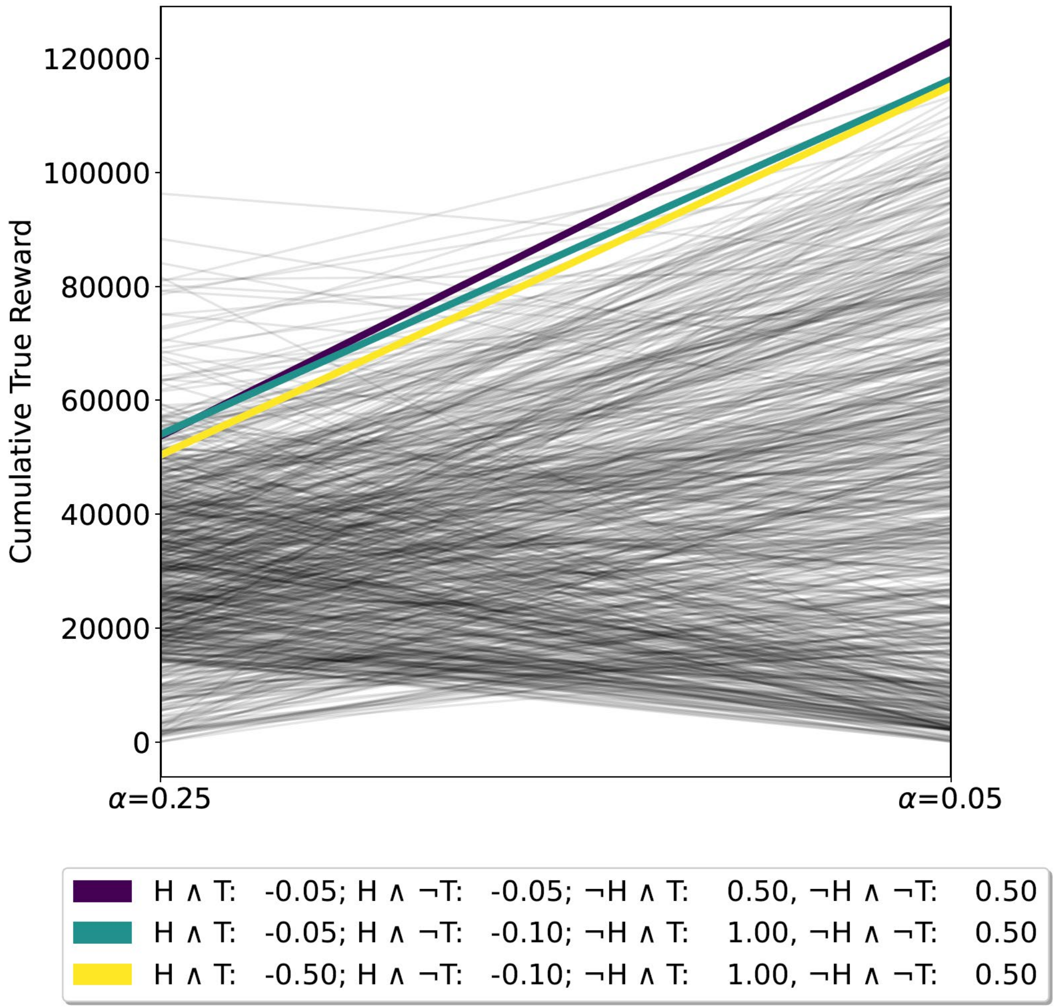
H1: Reward functions that achieve the best performance in one learning context can be suboptimal in another.

Overfitting in Parallel Coordinate Plots



Intersections indicate overfitting.

H1: Reward functions that achieve the best performance in one learning context can be suboptimal in another.

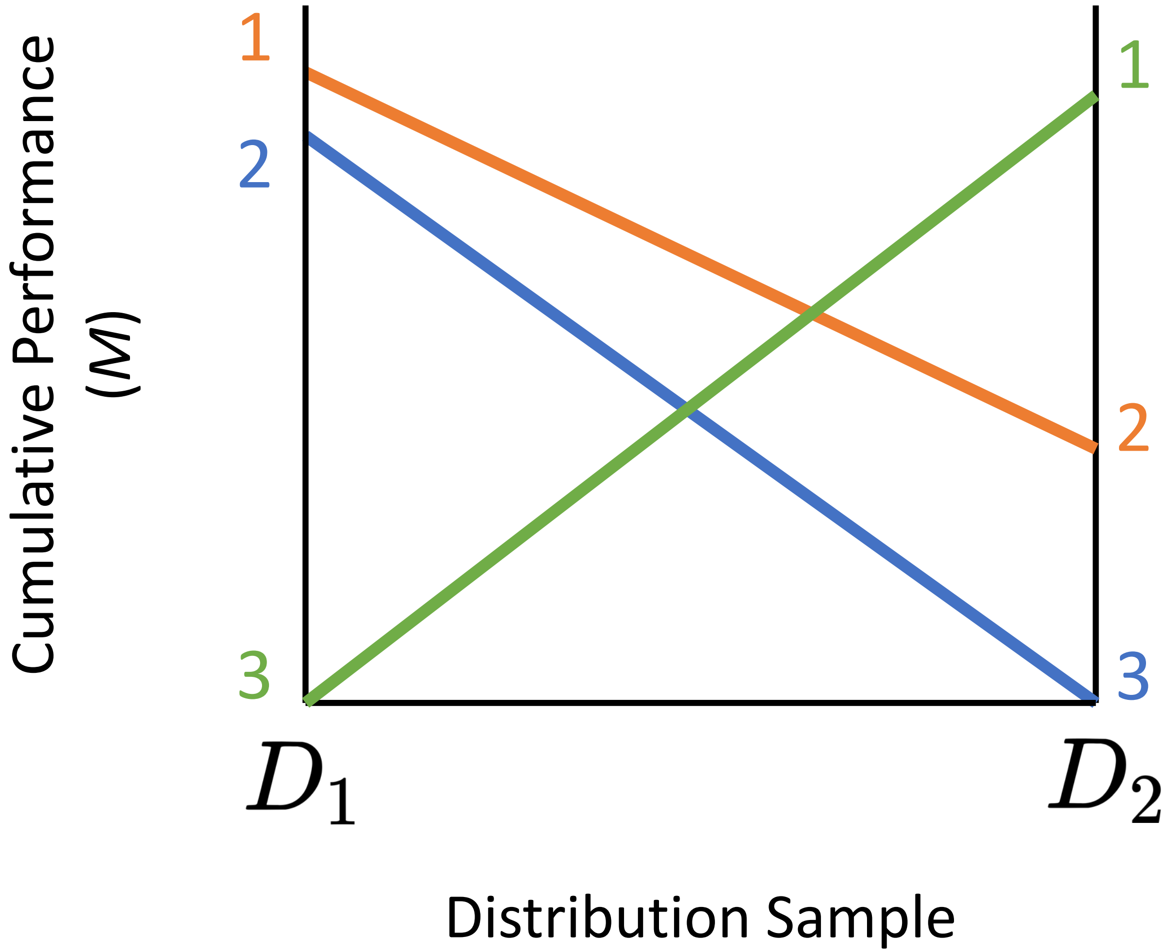


For all experiments, we find the best performing reward functions differ across learning contexts.

This is evidence of overfitting.

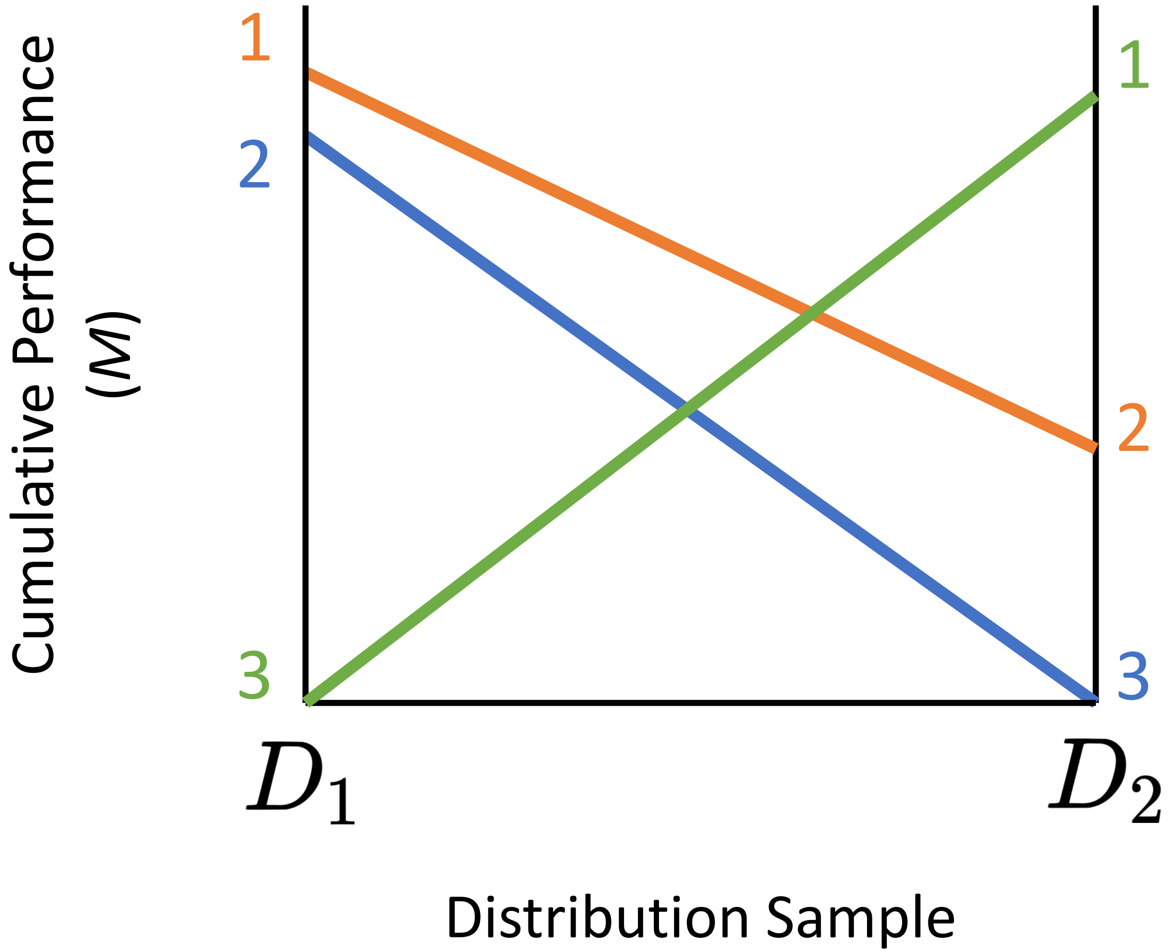
H2: The cumulative performances achieved with different reward functions are *uncorrelated* across different learning contexts.

H2: The cumulative performances achieved with different reward functions are uncorrelated across different learning contexts.



We rank all reward functions for each experiment setting (D_1 & D_2).

H2: The cumulative performances achieved with different reward functions are *uncorrelated* across different learning contexts.



We rank all reward functions for each experiment setting (D_1 & D_2).

We compare the ordering of these rankings using Kendall's tau.

H2: The cumulative performances achieved with different reward functions are uncorrelated across different learning contexts.

$\mathcal{D}_1$	$\mathcal{D}_2$	τ_b
$\gamma = 0.99$	$\gamma = 0.8$	0.07
$\gamma = 0.99$	$\gamma = 0.5$	0.04
$\gamma = 0.8$	$\gamma = 0.5$	0.12
$\alpha = 0.25$	$\alpha = 0.05$	0.11

We rank all reward functions for each experiment setting (D_1 & D_2).

We compare the ordering of these rankings using Kendall's tau.

We find that these rankings are **uncorrelated** ($|\tau_b| < 0.1$)

H2: The cumulative performances achieved with different reward functions are uncorrelated across different learning contexts.

$\mathcal{D}_1$	$\mathcal{D}_2$	τ_b
$\gamma = 0.99$	$\gamma = 0.8$	0.07
$\gamma = 0.99$	$\gamma = 0.5$	0.04
$\gamma = 0.8$	$\gamma = 0.5$	0.12
$\alpha = 0.25$	$\alpha = 0.05$	0.11

We rank all reward functions for each experiment setting (D_1 & D_2).

We compare the ordering of these rankings using Kendall's tau.

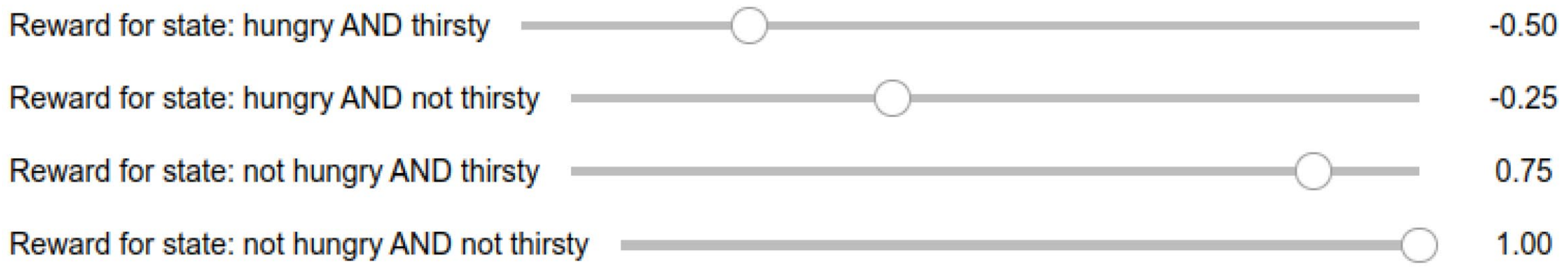
We find that these rankings are **uncorrelated** (| τ_b | < 0.1) or **slightly correlated** (| τ_b | < 0.2).

Conclusion?

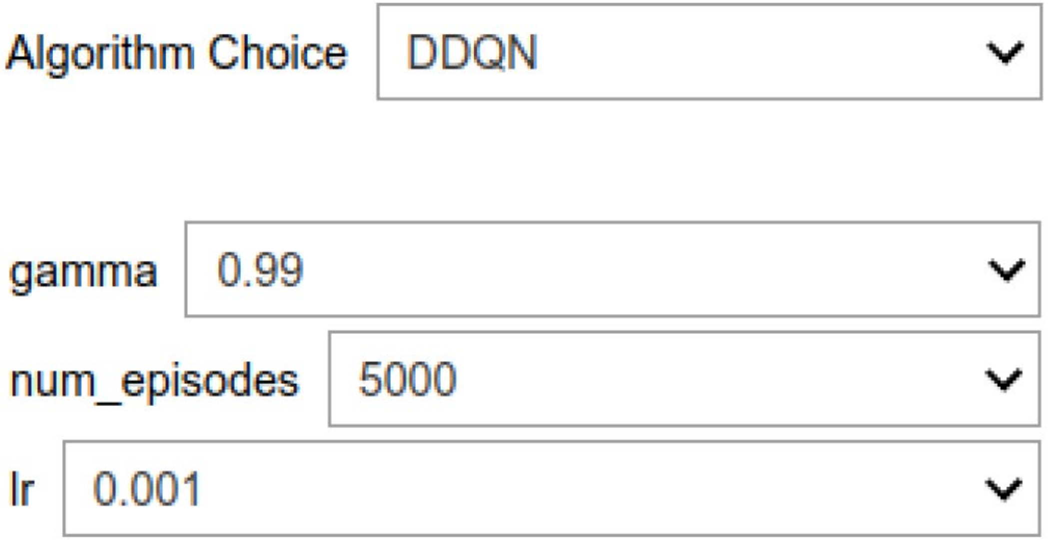
Overfitting to hyperparameters (and deep RL algorithms) is a concern.

Controlled Observation User Study (n=30)

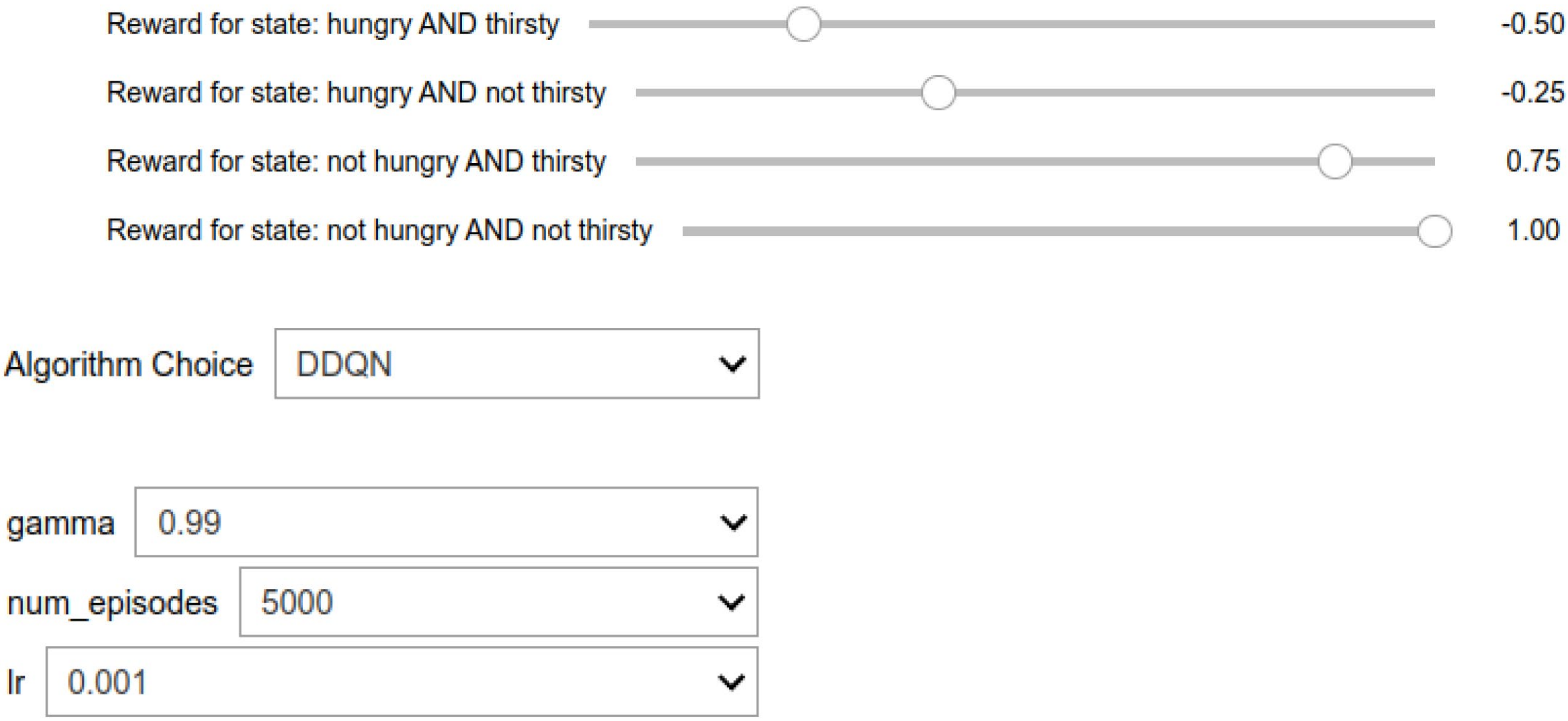
User Study Conducted in Jupyter Notebooks



User Study Conducted in Jupyter Notebooks



User Study Conducted in Jupyter Notebooks



User Study Conducted in Jupyter Notebooks

Reward for state: hungry AND thirsty -0.50

Reward for state: hungry AND not thirsty -0.25

Reward for state: not hungry AND thirsty 0.75

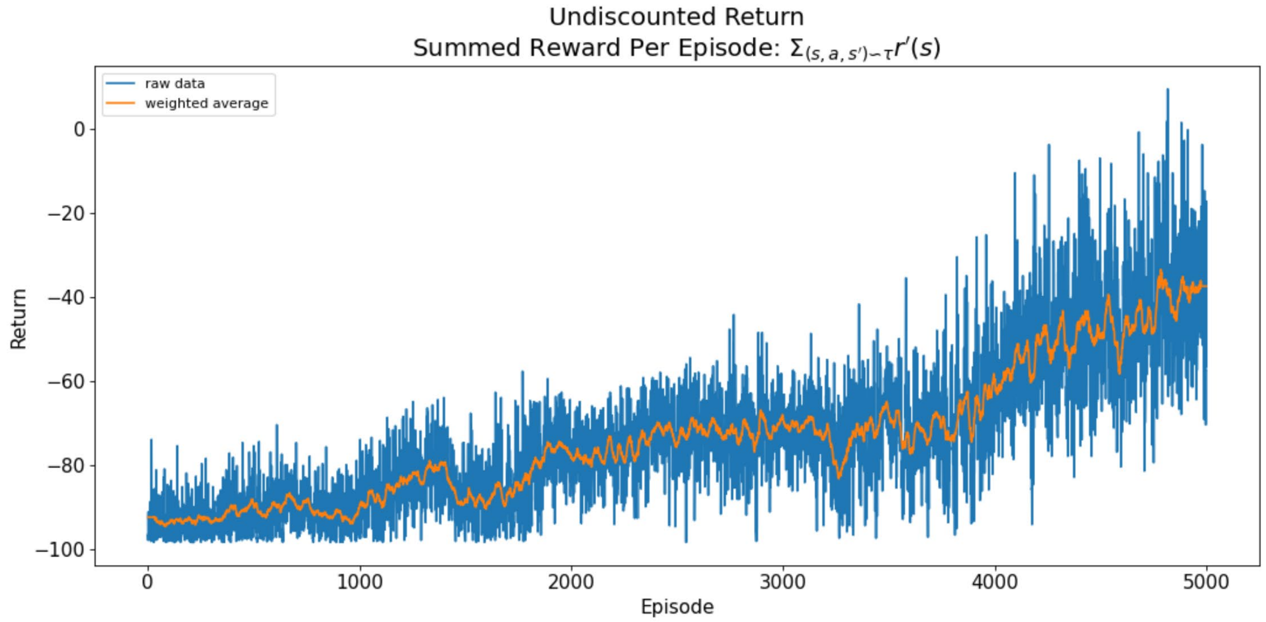
Reward for state: not hungry AND not thirsty 1.00

Algorithm Choice: DDQN

gamma: 0.99

num_episodes: 5000

lr: 0.001



Experts Overfit Reward Functions, too

User P20 first tried a reward function which achieved M=138,092 with DDQN.

They ultimately selected a different reward function, which achieved M=1,031 with DDQN.

Experts Overfit Reward Functions, too

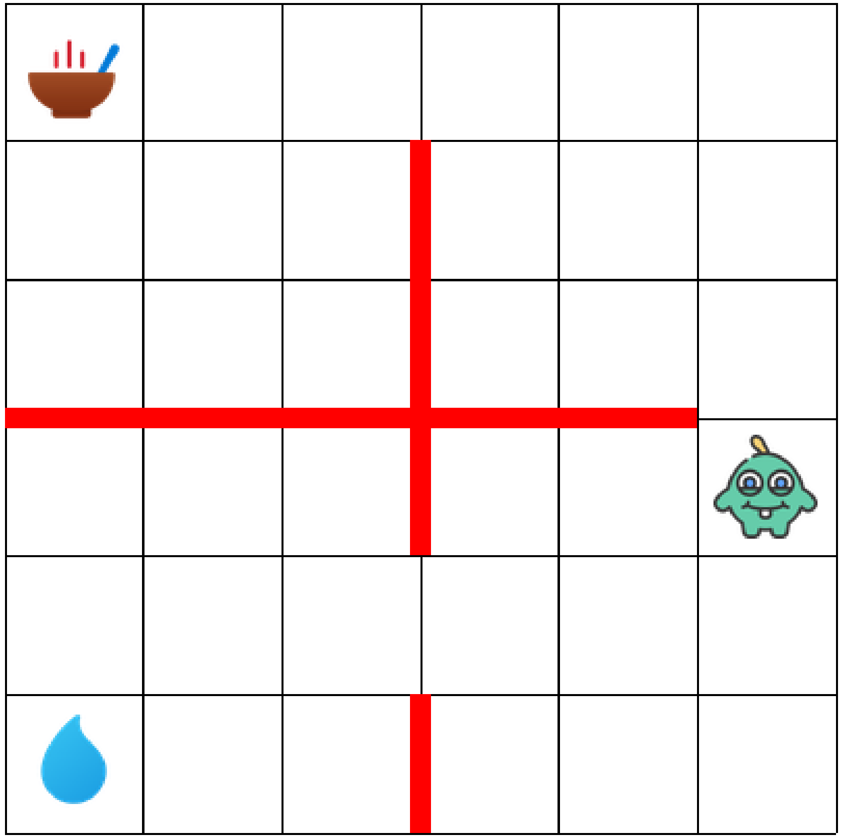
68% of users overfit reward functions

User P20 first tried a reward function which achieved M=138,092 with DDQN.

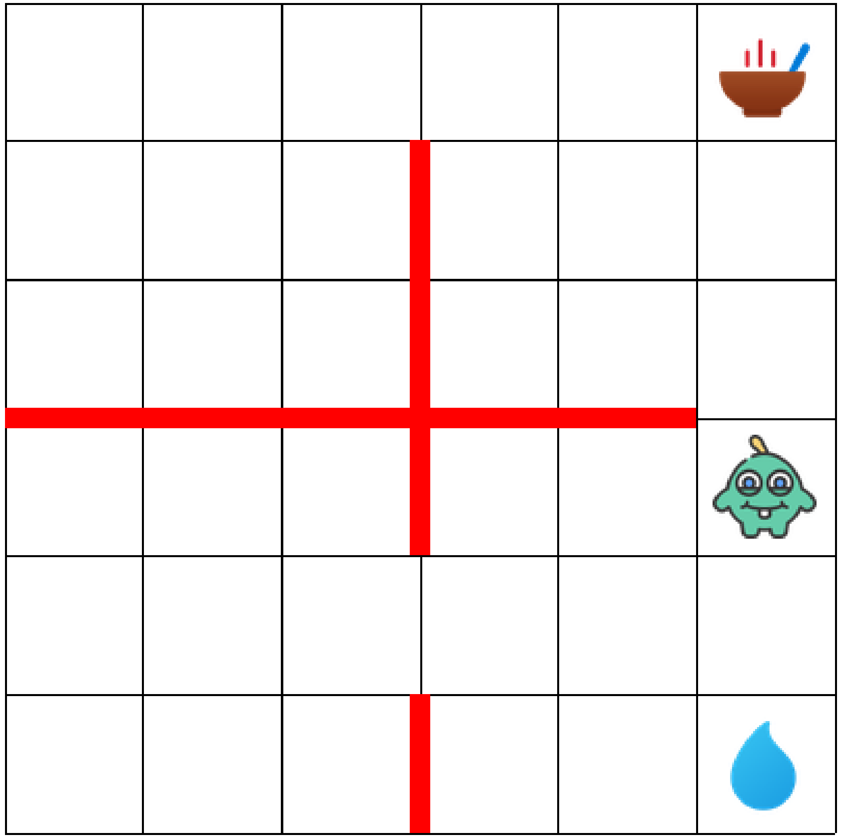
They ultimately selected a different reward function, which achieved M=1,031 with DDQN.

Experts are currently bad at writing reward functions.

Experts are currently bad at writing reward functions.

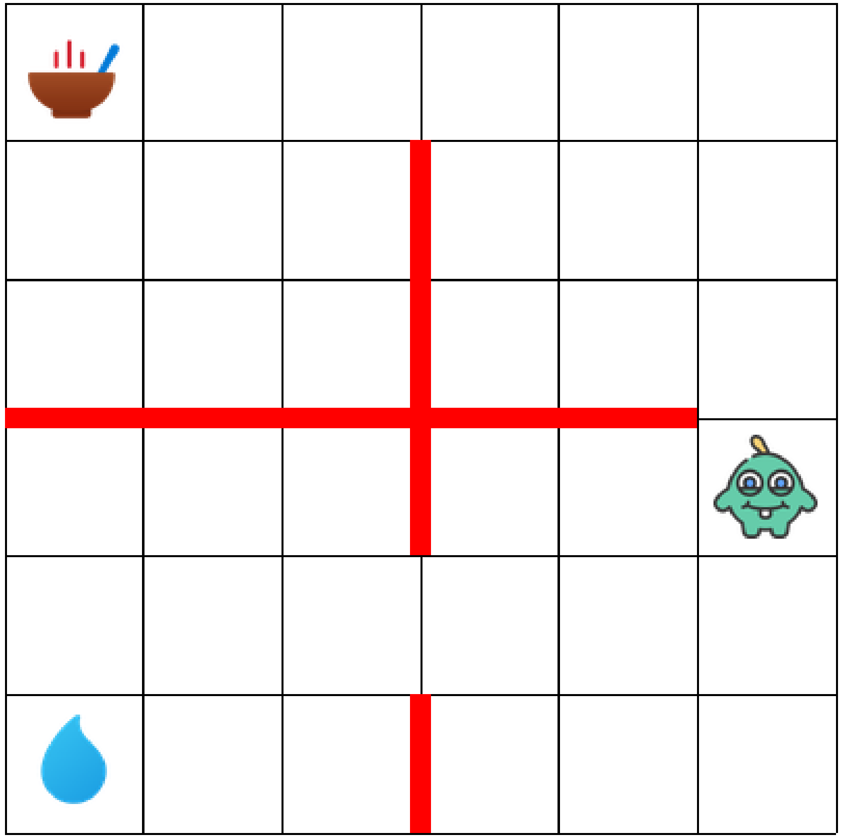


Hard configuration
(15 steps between water & food)



Easy configuration
(5 steps between water & food)

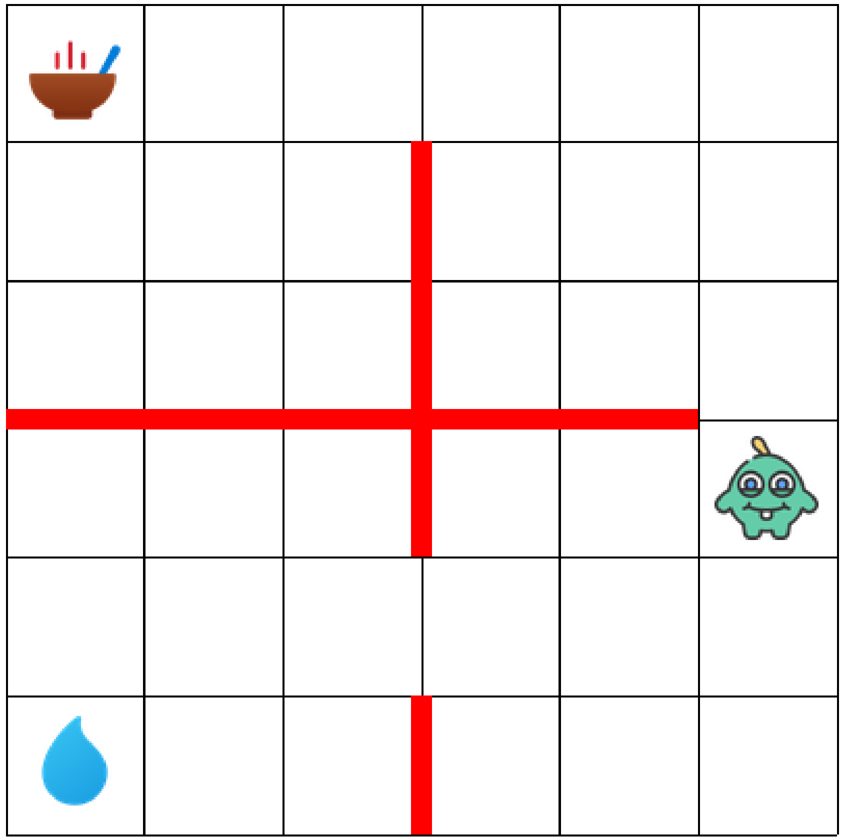
Experts are currently bad at writing reward functions.



Hard configuration
(15 steps between water & food)

53% of RL experts wrote reward functions which **failed to encode the task** in the hard case.

Experts are currently bad at writing reward functions.



Hard configuration
(15 steps between water & food)

53% of RL experts wrote reward functions which **failed to encode the task** in the hard case.

For example, **P3**'s reward function:

$$r(\neg\mathtt{H} \wedge \neg\mathtt{T}) = 1.0 \qquad r(\mathtt{H} \wedge \neg\mathtt{T}) = -0.1$$
$$r(\neg\mathtt{H} \wedge \mathtt{T}) = 1.0 \qquad r(\mathtt{H} \wedge \mathtt{T}) = -1.0$$

Most experts (83%) use
a *myopic* design strategy.

"It's best to not be hungry and thirsty, so I'll set that to the max, 1. Being not thirsty is better than being not hungry. Worst is at hungry AND thirsty; setting that to -1"

-P25

People are bad at reasoning about reward accumulation.

Takeaways

Reward functions can be overfit to learning algorithms.

Takeaways

Reward functions can be overfit to learning algorithms.

Practitioners should construct two reward functions:
one for learning and one for evaluating.

Takeaways

Reward functions can be overfit to learning algorithms.

Practitioners should construct two reward functions: one for learning and one for evaluating.

We should work to support human reward designers by aligning reward design & the RL objective.

Limitations & Future Work

Only tested with one domain!

Limitations & Future Work

Only tested with one domain!

Can alternative models of reward help?

Limitations & Future Work

Only tested with one domain!

Can alternative models of reward help?

How has overfitting affected the research record?

BOSCH
TEXAS
The University of Texas at Austin
Massachusetts Institute of Technology

The Perils of Trial-and-Error Reward Design


Serena Booth


Brad Knox


Julie Shah


Scott Niekum


Peter Stone


Alessandro Allievi

Code: github.com/serenabooth/reward-design-perils

Contact: sbooth@mit.edu

AAAI 2023