CS 690: Human-Centric Machine Learning

Prof. Scott Niekum

Reward Specification

Reward misspecification + hacking

Formalizing reward hacking

Instead, we ask whether there is *any* way in which improving a policy according to the proxy could make the policy worse according to the true reward; this is equivalent to asking if there exists a pair of policies π_1, π_2 where the proxy prefers π_1, but the true reward function prefers π_2. When this is the case, we refer to this pair of true reward function and proxy reward function as **hackable**.

Skalse, Joar, et al. "Defining and characterizing reward gaming." Advances in Neural Information Processing Systems 35 (2022): 9460-9471.

Sparse and dense rewards

0	0	0	100
0	0	0	0
0	0	0	0
0	0	0	0

Sparse

70	80	90	100
70	80	90	90
70	80	80	80
70	70	70	70

Dense

Potential-based reward shaping

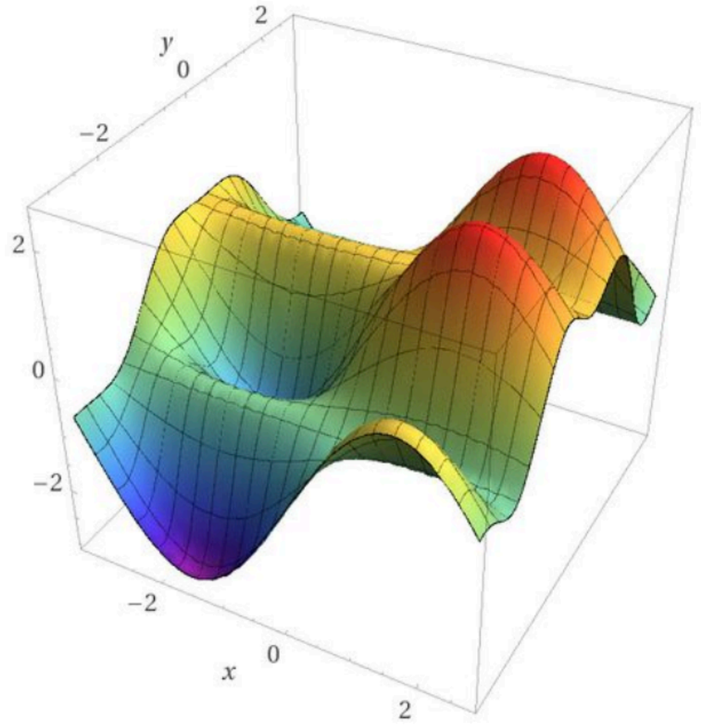
Shaped reward function:

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

Where:

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

Guaranteed not to change optimal policy!



Ng, Andrew Y., Daishi Harada, and Stuart Russell. "Policy invariance under reward transformations: Theory and application to reward shaping." ICML. Vol. 99. 1999.

…is equivalent to value function initialization

- Value function initialization can be “learned away”
- …often too early in learning
- Thus, sometimes not very useful in practice

Simple, sparse rewards can be hacked too

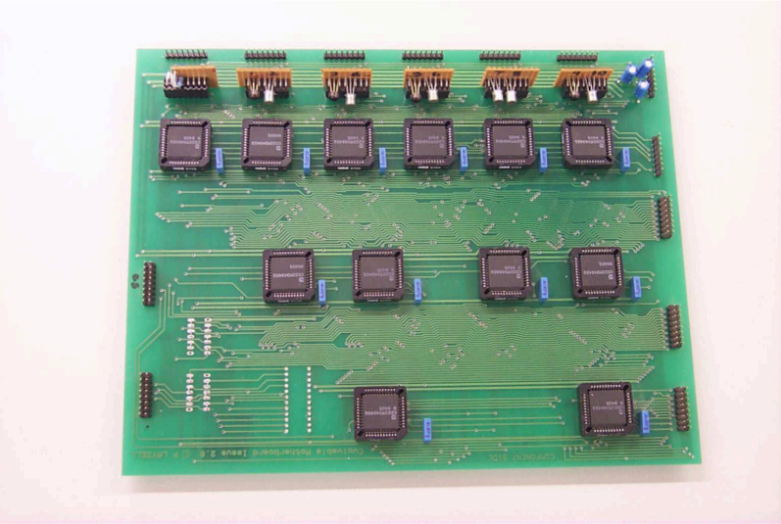


Figure 1: the evolvable motherboard (EM)

- Used an “evolvable motherboard” to perform evolutionary search over circuits
- Attempted to evolve an oscillator
- Learned a radio instead that listened to (and repeated) oscillations from a nearby PC!

Bird, J. and Layzell, P. (2002). The evolved radio and its implications for modelling the evolution of novel sensors. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600), volume 2, pages 1836–1841. IEEE.

How bad is an incomplete specification?

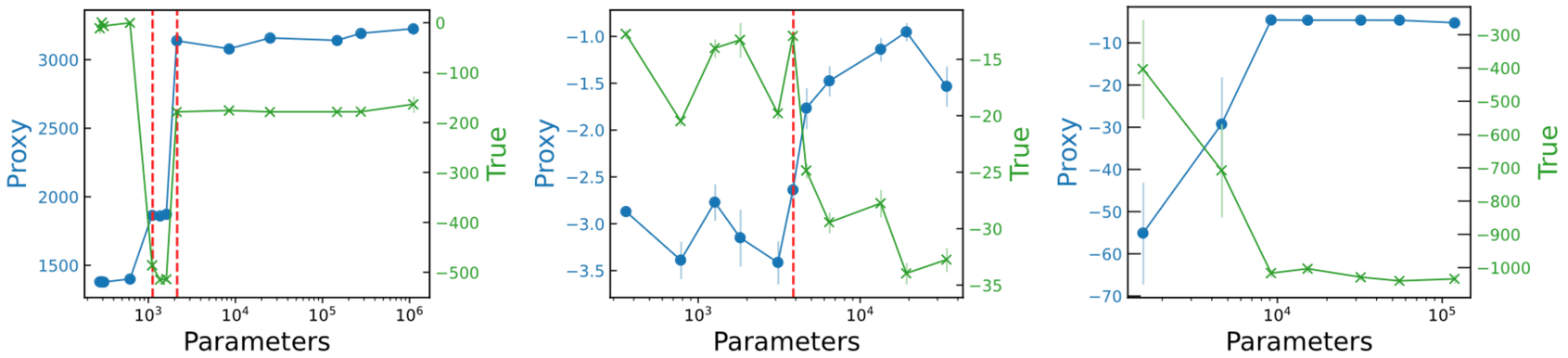
“We assume that the reward function given to the agent only has support on $J < L$ attributes”

“Our main result identifies conditions such that any misalignment is costly: starting from any initial state, optimizing any fixed incomplete proxy eventually leads the principal to be arbitrarily worse off”

“In the worst-case scenario, the robot moves in a way that subtracts an arbitrarily large amount from one of the unmentioned attributes, while gaining infinitesimally in one of the proxy attribute”

Zhuang, Simon, and Dylan Hadfield-Menell. "Consequences of misaligned AI." Advances in Neural Information Processing Systems 33 (2020): 15763-15773.

Reward hacking is sensitive to model size



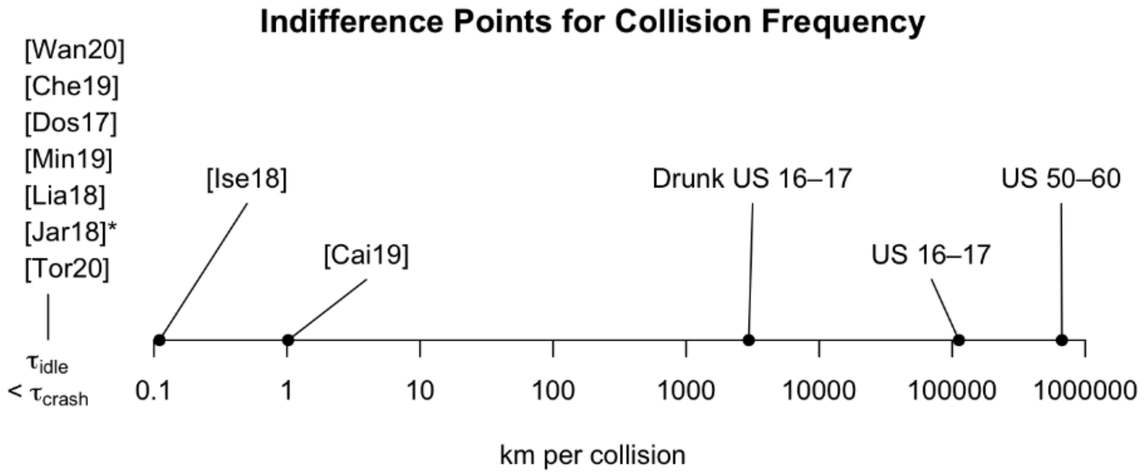
(a) Traffic - Ontological (b) COVID - Ontological (c) Glucose - Ontological

Figure 2: Increasing the RL policy's model size decreases true reward on three selected environments. The red line indicates a phase transition.

Pan, Alexander, Kush Bhatia, and Jacob Steinhardt. "The effects of reward misspecification: Mapping and mitigating misaligned models." *arXiv preprint arXiv:2201.03544* (2022).

How often does misspecification really happen?

	Sanity check failures	Brief explanation
1	Unsafe reward shaping	If reward includes guidance on behavior that deviates from only measuring desired outcomes, reward shaping exists.
2	Mismatch in people's and reward function's preference orderings	If there is human consensus that one trajectory is better than another, the reward function should agree.
3	Undesired risk tolerance via indifference points	Assess a reward function's risk tolerance via indifference points and compare to a human-derived acceptable risk tolerance.
4	Learnable loophole(s)	If learned policies show a pattern of undesirable behavior, consider whether it is explicitly encouraged by reward.
5	Missing attribute(s)	If desired outcomes are not part of reward function, it is indifferent to them.
6	Redundant attribute(s)	Two or more reward function attributes include measurements of the same outcome.
7	Trial-and-error reward design	Tuning the reward function to improve RL agents' performances has unexamined consequences.
8	Incomplete description of problem specification	Missing descriptions of reward function, termination conditions, discount factor, or time step duration may indicate insufficient consideration of the problem specification.



Knox, W. B.; Allievi, A.; Banzhaf, H.; Schmitt, F.; and Stone, P. 2023. Reward (mis) design for autonomous driving. Artificial Intelligence, 316: 103829.

Overfitting: “Correct” specifications aren’t enough?

- How do people really create and use reward functions?
  - Often iteratively while observing behaviors that they generate
  - Can overfit to particular algorithms and hyperparameters
  - Won’t generalize to new algorithms / parameter settings

- What if state distribution changes from training to deployment?
  - Policy may have overfit to a feature that was perfectly correlated with good performance in training distribution, but not test distribution
  - May cause it to do dangerous things in test distribution when correlation is not longer perfect

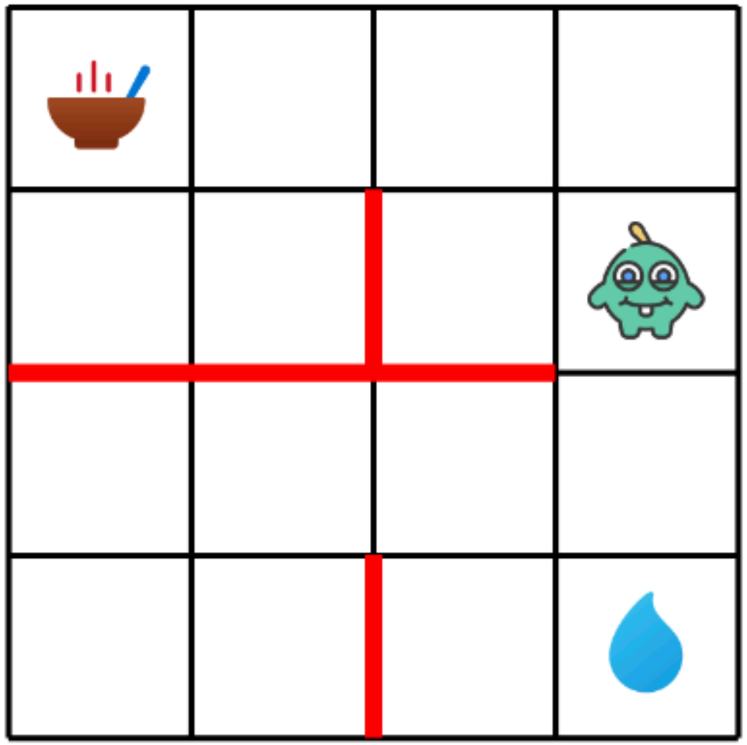
Reward function overfitting

Reward Function Overfitting Let $M : \tau \rightarrow \mathbb{R}$ be the true task performance metric. For example, this metric might encode whether the agent reached a goal state or not. Let a **learning context** be a tuple of an RL algorithm, hyperparameter values, and an MDP$\backslash r$; given a reward function, a learning context can be used to train a policy. We claim a reward function r_1 is overfit with respect to one or more learning contexts, $D_1 \sim \mathcal{D}$, if there exists an alternative reward function r_2 such that the task performance metric is optimized over D_1 but not over the larger distribution, $\mathcal{D}$:

$$\begin{aligned} \mathop{\mathbb{E}}_{\tau \sim \pi_{r_1, D_1}} [M(\tau)] &> \mathop{\mathbb{E}}_{\tau \sim \pi_{r_2, D_1}} [M(\tau)] \\ \mathop{\mathbb{E}}_{\tau \sim \pi_{r_1, \mathcal{D}}} [M(\tau)] &< \mathop{\mathbb{E}}_{\tau \sim \pi_{r_2, \mathcal{D}}} [M(\tau)] \end{aligned} \tag{1}$$

Booth, Serena, et al. "The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. No. 5. 2023.

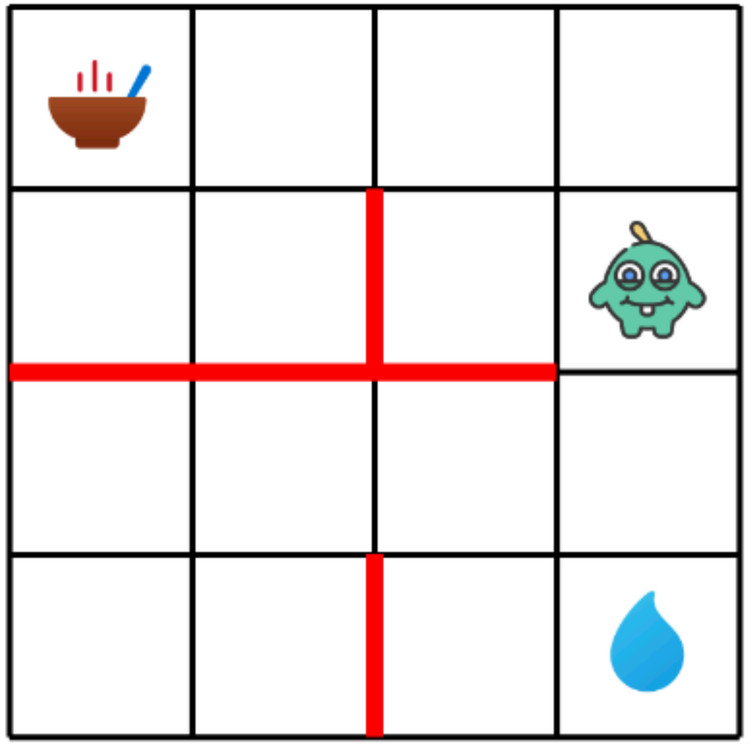
Reward function overfitting



Hungry-Thirsty domain

Reward function overfitting

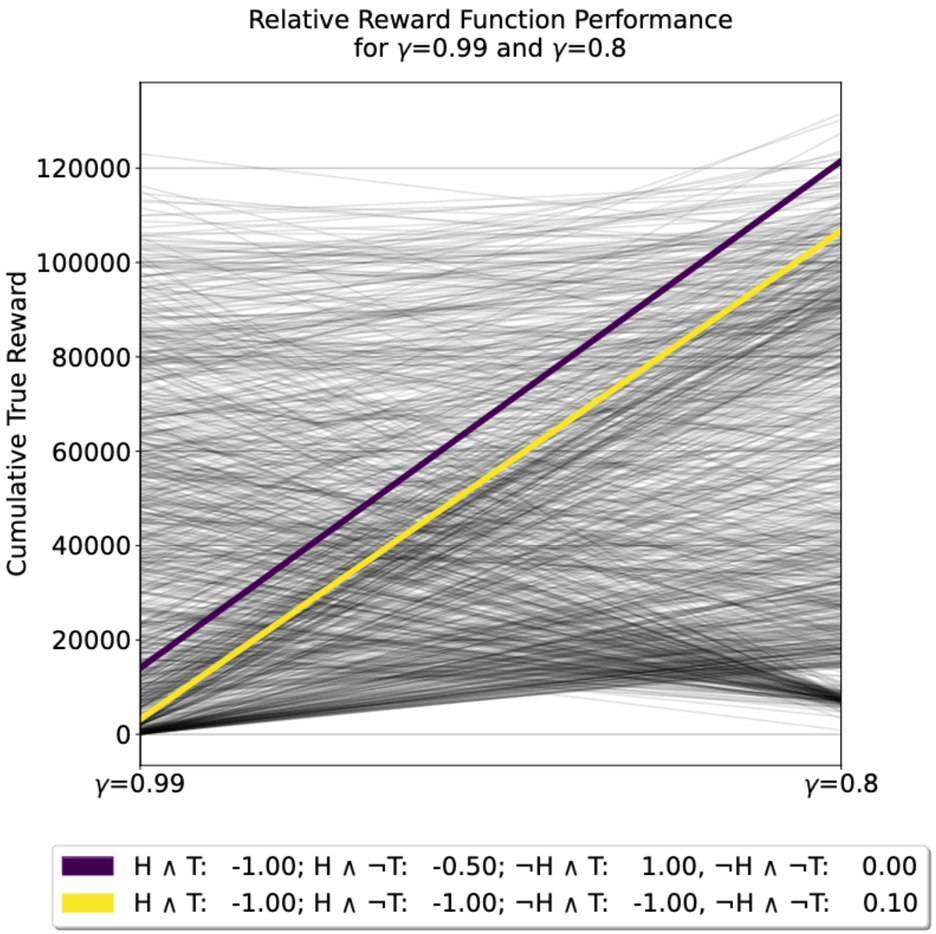
Hungry-Thirsty domain



For our experiments, all reward functions take the form:

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

Is reward function performance affected by learning context?



# of Reward Fns	D_1	D_2	τ_b	p-value
5196	$\gamma = 0.99$	$\gamma = 0.8$	0.07	< 0.01
	$\gamma = 0.99$	$\gamma = 0.5$	0.04	0.07
	$\gamma = 0.8$	$\gamma = 0.5$	0.12	< 0.01
	$\alpha = 0.25$	$\alpha = 0.05$	0.11	< 0.01
107	PPO	A2C	0.25	0.01
	PPO	DDQN	-0.04	0.62
	PPO	QLearn	0.13	0.08
	A2C	QLearn	-0.08	0.29
	A2C	DDQN	-0.01	0.87
	DDQN	QLearn	-0.06	0.41

Reward function overfitting: human study

- Experts often design invalid reward functions
  - 83% of time successfully design valid RF for “close” configuration of food and water
  - But only 47% valid when far apart! Agent gets obsessed with drinking water.

- Experts overfit to algorithms
  - 68% settled on a final RF that was worse than a previous RF for some of the algorithms
  - Hard to say exactly how “overfit” should be defined here, however

Reward function overfitting: human study

- 97% shaped rewards even though not told to explicitly
- Often shaped incorrectly — weighting based on myopic “state goodness” rather than imagining how the RF be optimized by RL.
  - “It’s best to be ¬H ∧ ¬T, so I’ll set that to the max, 1. Being ¬T is better than being ¬H. Worst is at H∧T; setting that to -1” —> in one case lead to always drinking
- Only 30% considered long-term cumulative rewards
  - “A positive reward for H ∧ ¬T is not the way to go. A combination with a negative reward for H ∧ T makes it worse, since it would rather accumulate positive rewards at the water instead of searching for food.”

Reward function overfitting: practical consequences

- When would this really matter?
- What could be done?

Goal misgeneralization

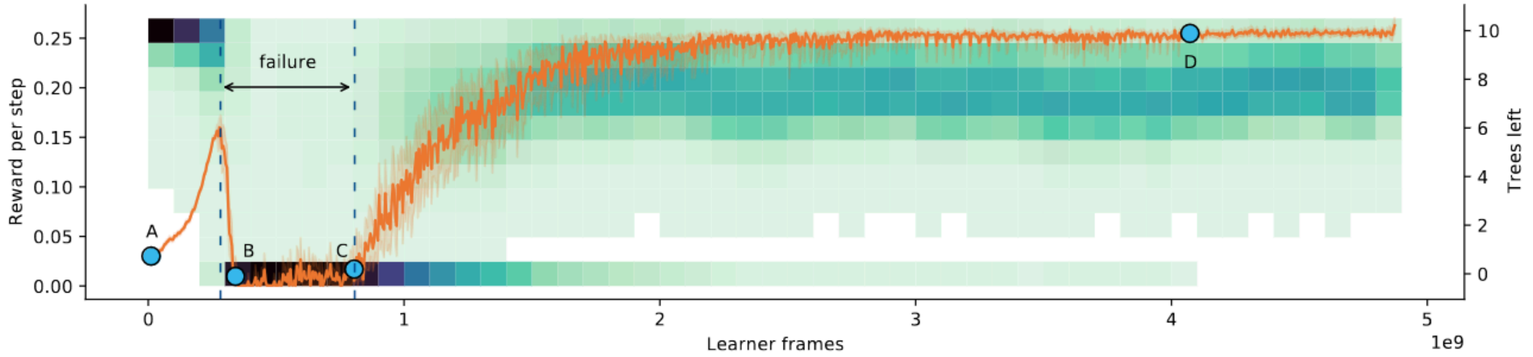
- Optimizing for a ***correlate*** of good performance that works at training time, but not test time due to ***distribution shift***
- Another view: multiple hypotheses fit the training data well, but not all generalize well to test data

Goal misgeneralization

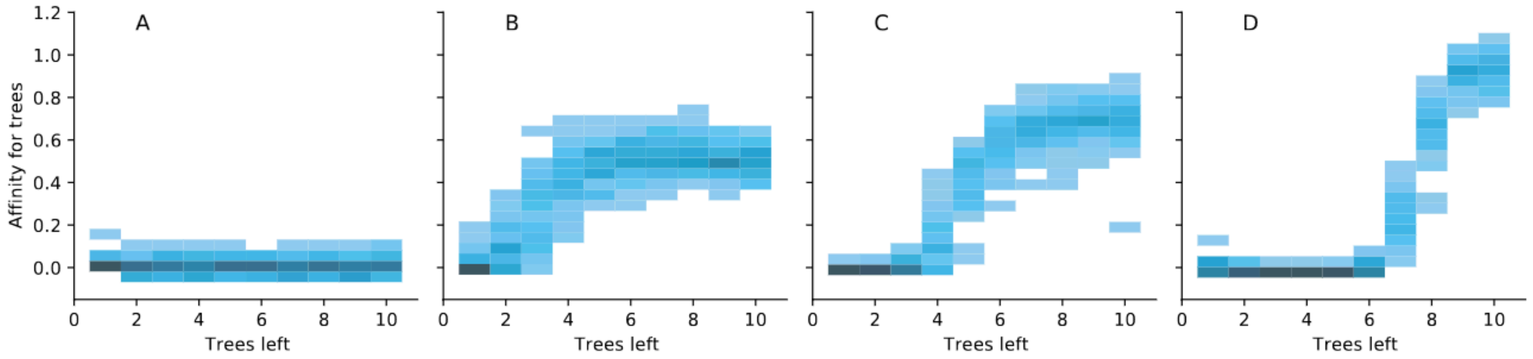
- **Monster domain**
  - Gets obsessed with picking up shields even when monsters aren’t around.
  - Domain shift = longer episodes, so that most monsters are killed by the end
  - Fixed by greater training diversity

- **Tree gridworld**
  - Domain shift caused by the agent itself! Chopping down trees lowers respawn rate
  - Takes a long time to recover from the always-chop policy

Goal misgeneralization



(a) We plot average per-step reward in orange, and show trees remaining in green. At X learner frames and Y trees, darker green signifies higher probability that around learner frame X the environment contained Y trees.



(b) We evaluate the agent's 'affinity' for trees at various points in training. For a given number of trees remaining in the environment, an affinity of 0 roughly corresponds to the random policy, and an affinity of 1 roughly corresponds to the greedy policy. See Appendix C.3 for the definition of affinity and details of evaluation.

Goal misgeneralization

- **LLM evaluation of linear expressions**
  - Learns to always ask questions because some variables are always unknown during training
  - At test time, all variables are known, but still asks unnecessary questions
- **Cultural transmission**
  - Need to reach target locations in certain order
  - Bad correlation! Training: learns to follow optimal partner; Test: continues to follow pessimal partner
  - Rewards are part of agent observation, so should be able to recognize (in principle) that it is doing the wrong thing (due to RNN policy that can remember history)

Extrapolating to catastrophic risk?

- **Superhuman hacker**
  - Desired: write new software features that can be merged via pull requests
  - Misgeneralized: Get humans to click the ‘merge’ button

- **Other scenarios?**

Mitigation

- Diverse training data
- Maintaining uncertainty
- Understanding inductive biases
- Interpretability
- Model-assisted detection