Learning Optimal Advantage from Preferences and Mistaking it for Reward

W. Bradley Knox[1][2] Stephane Hatgis-Kessell[1] Sigurdur Orn Adalgeirsson[2] Serena Booth[3] Anca Dragan[4] Peter Stone[1][5] Scott Niekum[6]

First two authors contributed equally
[1]The University of Texas at Austin [2]Google Research [3]MIT CSAIL
[4]UC Berkeley [5]Sony AI [6]University of Massachusetts Amherst.

PAPER

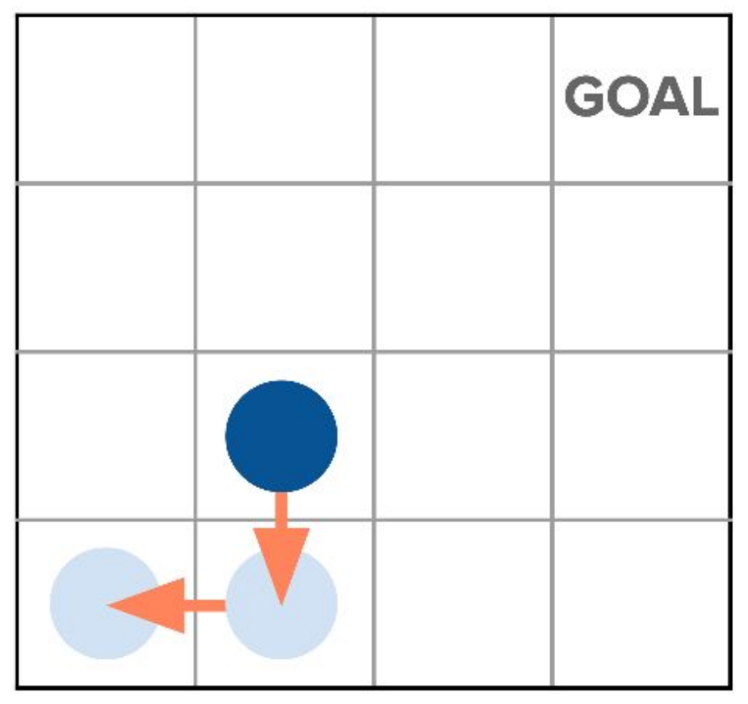
Most RLHF algorithms assume an underexamined *partial return* model of human preference. We previously found that another model based on regret better describes human preferences.

What what are the consequences of this mistaken assumption?

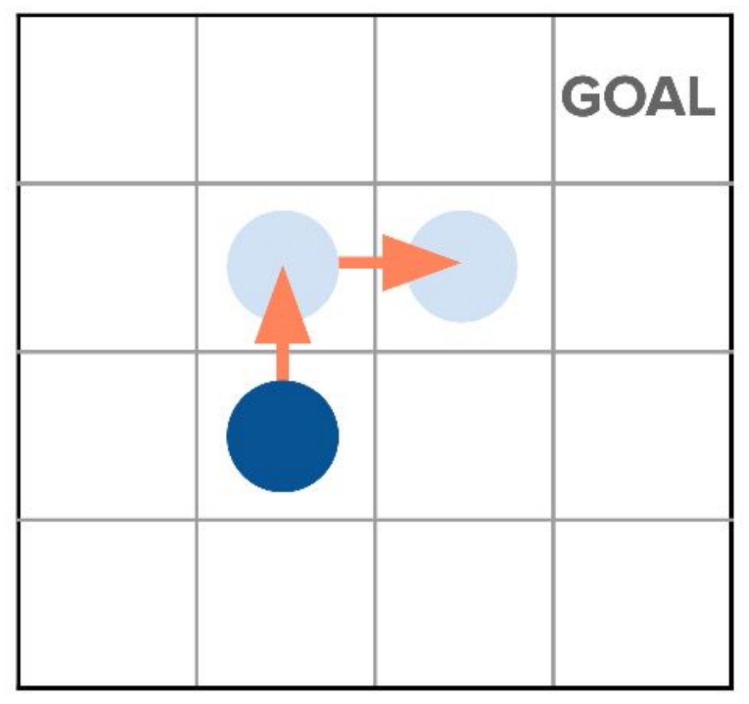
Which fits your preferences?

Which shows better behavior?

1)

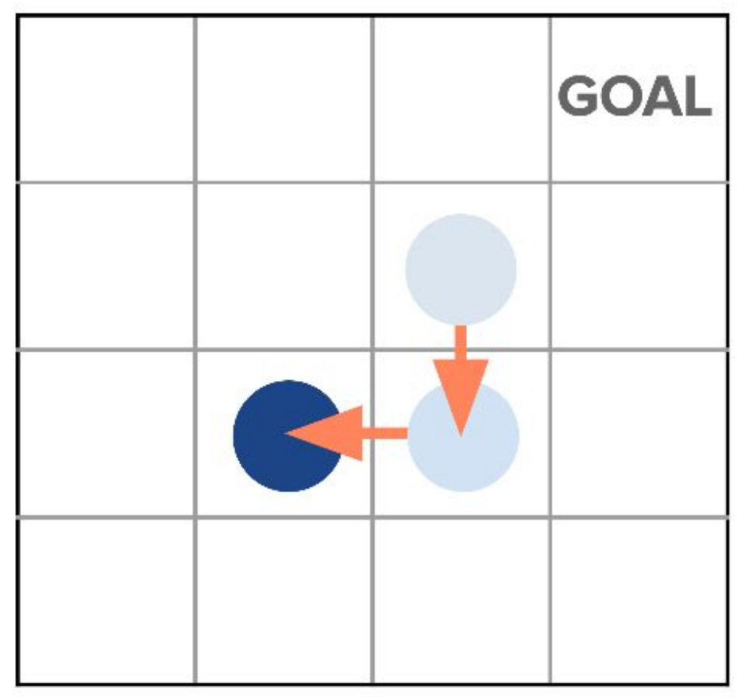


Equal partial return
Higher regret

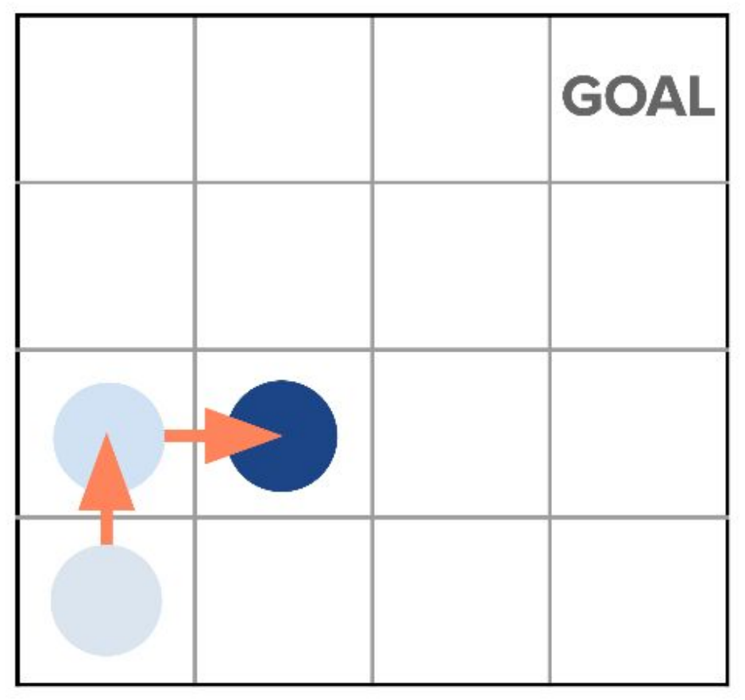


Equal partial return
Lower regret

2)



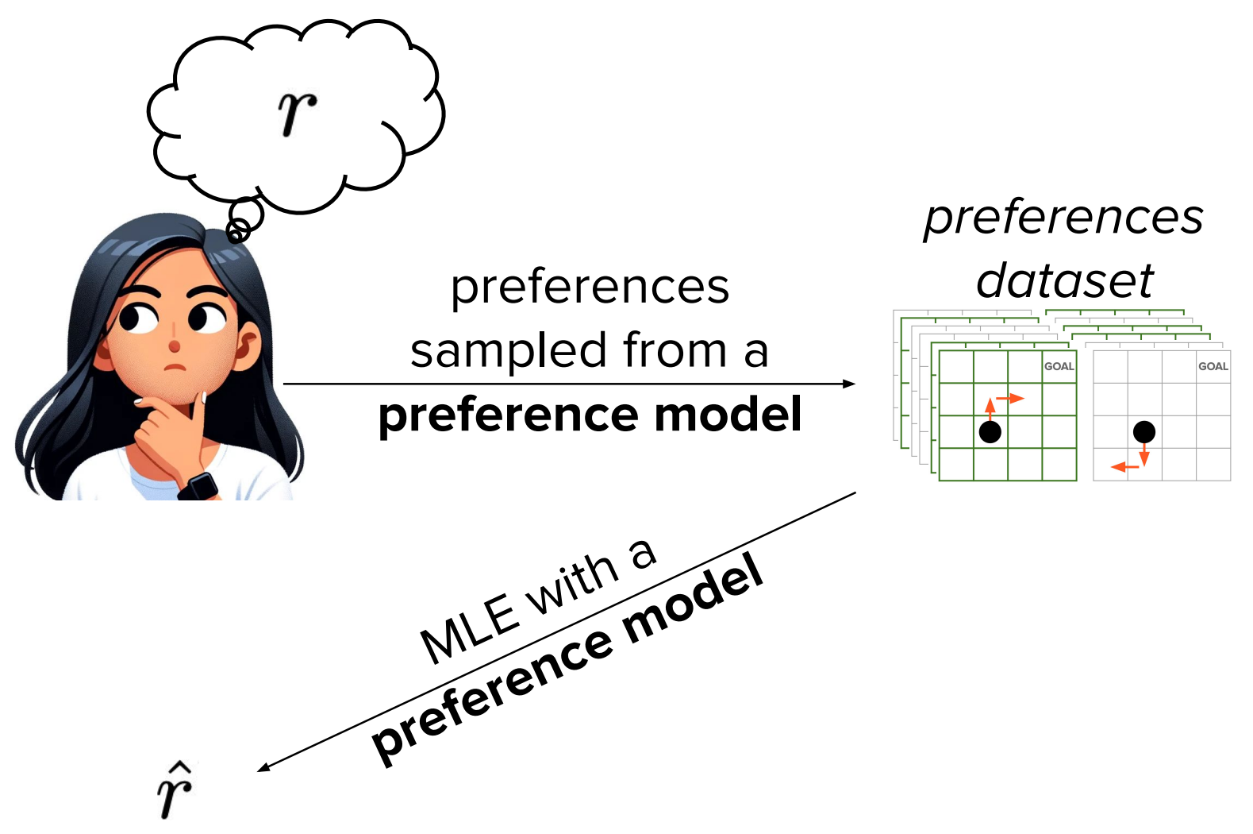
Equal partial return
Higher regret



Equal partial return
Lower regret

Background

Typical RLHF algorithm's view of the world


r
preferences sampled from a **preference model**
preferences dataset
GOAL
GOAL
MLE with a **preference model**
$\hat{r}$

The preference model

Common model: **Partial return**

$$P(\sigma_1 \succ \sigma_2) = \text{logistic}\Big(\sum_{(s,a)\in\sigma_1} r(s,a) - \sum_{(s,a)\in\sigma_2} r(s,a)\Big)$$

Proposed model: Regret

$$P(\sigma_1 \succ \sigma_2) = \text{logistic}\Big(\sum_{(s,a)\in\sigma_1} A_r^*(s,a) - \sum_{(s,a)\in\sigma_2} A_r^*(s,a)\Big)$$

The regret of a segment **measures how much it deviates from optimal behavior.**

Initial insight

If an RLHF algorithm learns from regret-based preferences yet assumes the partial return model, then it **approximates A_r^* and then uses it as a reward function.**

General results: using optimal advantage as reward

When A_r^* is known exactly

- [Theory] **Optimal policies are preserved.**
- [Theory] **An underspecification issue is resolved** where choice of discount factor (γ) can be impactful yet arbitrary.
- [Theory] **Reward is highly shaped**, effectively setting $\phi(s) = V_r^*(s)$ as recommended by Ng et. al. (1999).
- Since $argmax_a A_r^*$ creates an optimal policy, using A_r^* as reward **wastes computation and environment sampling**.

When A_r^* is approximated as $\widehat{A_r^*}$

- [Theory] If $max_a \widehat{A_r^*}(\cdot, a) = 0$, then using $\widehat{A_r^*}$ as reward creates a set of policies equivalent to $argmax_a \widehat{A_r^*}$.
- **Otherwise, performance can be catastrophically poor.**
  - **Adding transitions from absorbing state** to early-terminating segments ameliorates this issue.
  - **Why?** Including segments with transitions from absorbing state encourages $max_a \widehat{A_r^*}(\cdot, a) = 0$.
- **Arbitrary bias towards or against termination determines performance differences**:

Condition	π_r^* terminates	π_r^* does not terminate
Max loop partial return > 0	$greedy\ Q^*_{r_{\widehat{A_r^*}}}$	$greedy\ \widehat{A_r^*}$
Max loop partial return < 0	$greedy\ \widehat{A_r^*}$	$greedy\ Q^*_{r_{\widehat{A_r^*}}}$

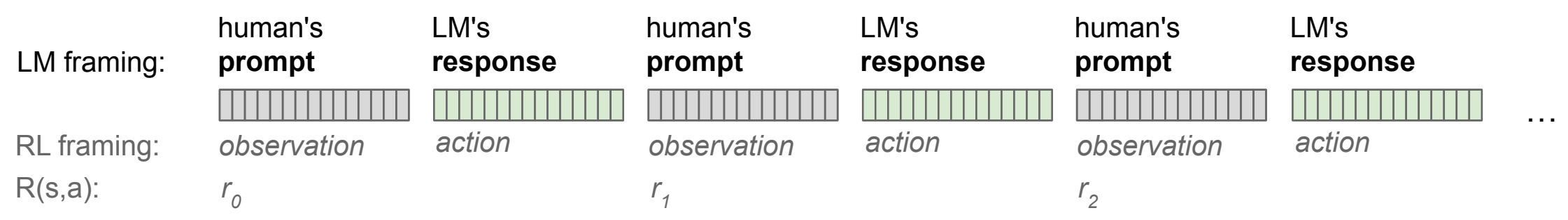
Table: Hypothesis regarding which algorithm performs as well or better than the other, given 2 conditions.

- When adding absorbing transitions, **reward is also highly shaped** with the approximation error of $\widehat{A_r^*}$.

Reframing LLM fine-tuning

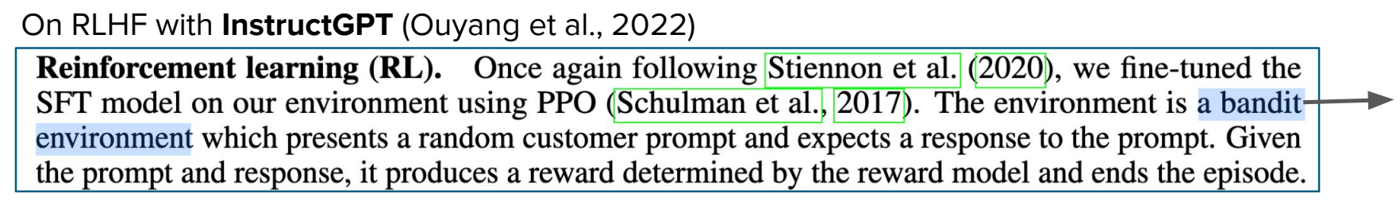
Is it possible that **annotators give regret-based preferences** *and* **engineers using fine-tuning are unknowingly applying the regret preference model?**

The multi-turn language problem



LM framing:	human's **prompt**	LM's **response**	human's **prompt**	LM's **response**	human's **prompt**	LM's **response**	…
RL framing:	*observation*	*action*	*observation*	*action*	*observation*	*action*	
R(s,a):	r_0		r_1		r_2		

On RLHF with **InstructGPT** (Ouyang et al., 2022)



Reinforcement learning (RL). Once again following Stiennon et al. (2020), we fine-tuned the SFT model on our environment using PPO (Schulman et al., 2017). The environment is a bandit environment which presents a random customer prompt and expects a response to the prompt. Given the prompt and response, it produces a reward determined by the reward model and ends the episode.

But the multi-turn problem is not a bandit problem!

Deriving the fine-tuning decision rule

Partial return assumes the learned function approximates *r*.

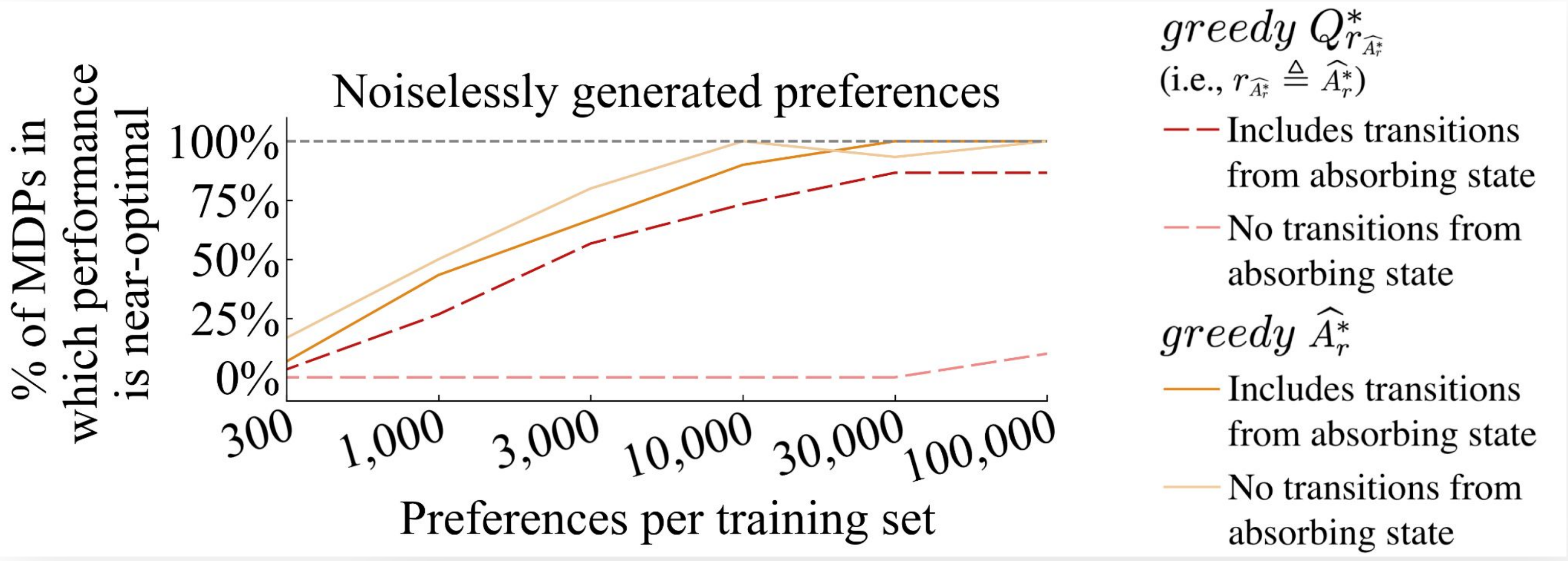
Must assume γ=0

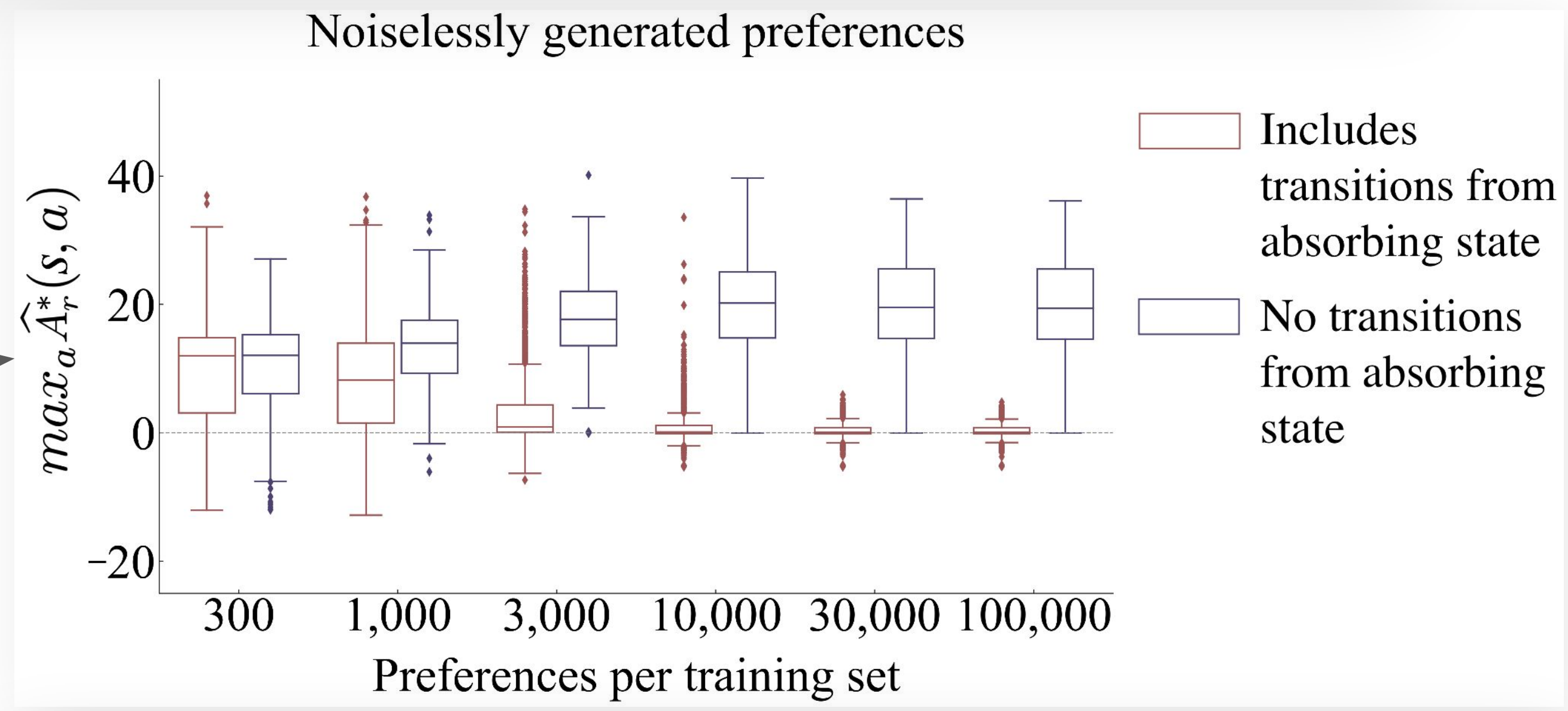
$$\begin{aligned}\pi_r^*(s) &= argmax_a\ Q_r^*(s,a)\\ &= argmax_a(r(s,a) + \gamma E_{s'}[V_r^*(s')])\\ &= argmax_a\ r(s,a) \longleftarrow \text{bandit task}\end{aligned}$$

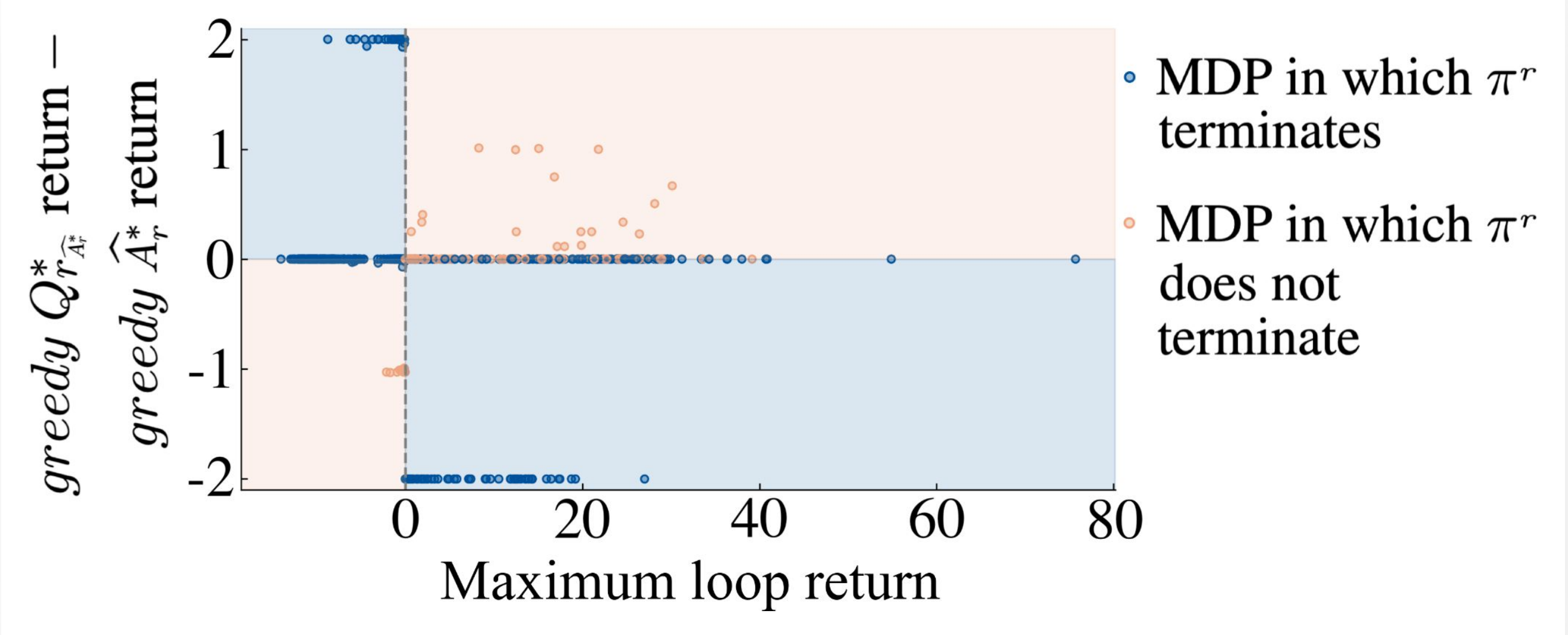
Regret
Assumes the learned function approximates A_r^*.
No γ hyperparameter.
$$\pi_r^*(s) = argmax_a\ A_r^*(s,a)$$

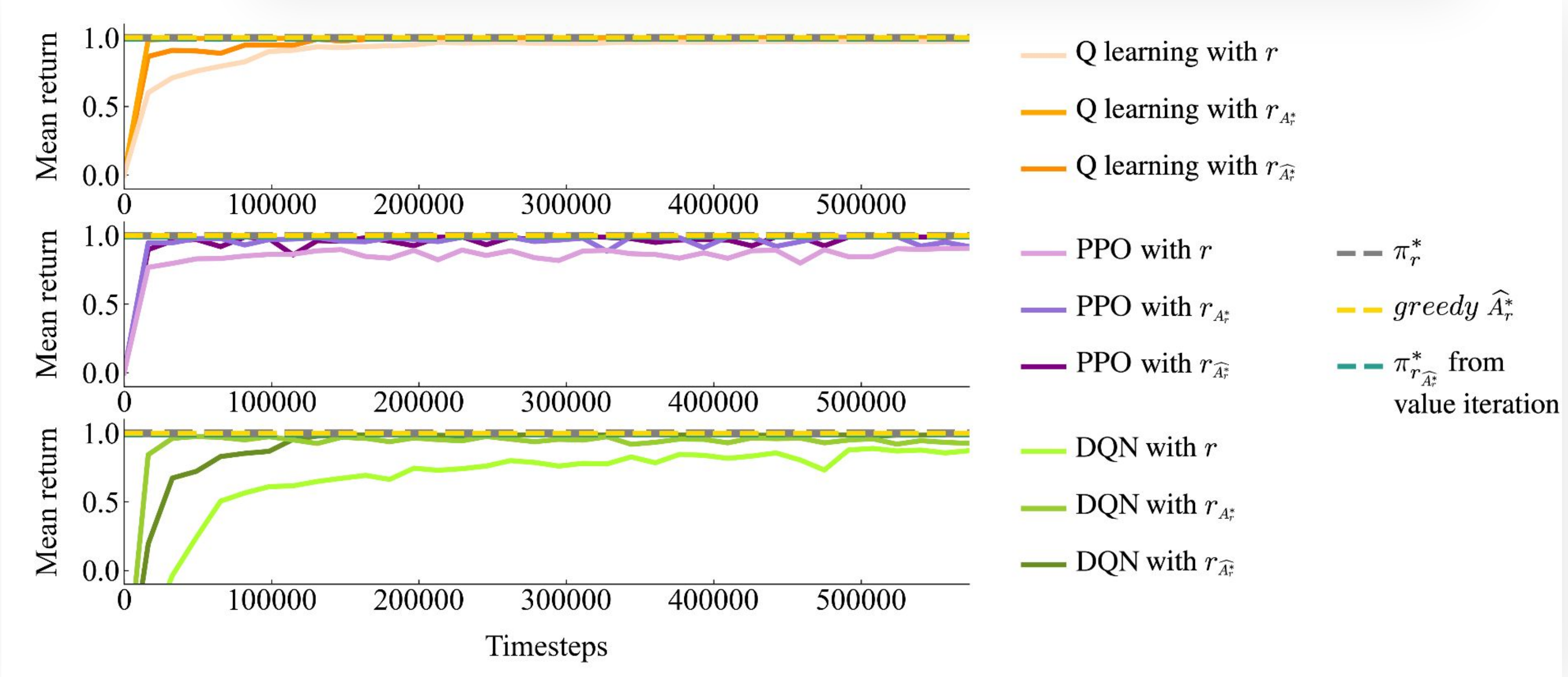
We get the same fine-tuning algorithm with a better supported preference model and without the arbitrary assumption of γ=0!

Experiments in 30+ gridworld MDPs









Conclusions

- Shaping results may explain why the partial return preference model often performs well.
- Revealed large pitfall and amelioration by including absorbing states in early-terminating segments.
- Offers a simpler reframing of the main method for fine-tuning LLMs with RLHF.