After Appendix A, the appendix is organized according to the major sections and subsections of the main content.

A Limitations and ethics

A.1 Limitations

Some limitations of the regret preference model are discussed in the paragraph "Regret as a model for human preference" in Section 2.2, including assumptions that a person giving preferences can distinguish between optimal and suboptimal segments, that they follow a Boltzmann distribution (i.e., a Luce Shepard choice rule), and that they base their preferences on decision quality even when transition stochasticity results in segment pairs for which the worse decision has a better outcome.

Our proposed algorithm (Section 6.1) has a few additional limitations. Generating candidate successor features for the approximations $\hat{Q}^*_{\hat{r}}$ and $V^*_{\hat{r}}$ may be difficult in complex domains. Specifically, challenges include choosing the set of policies or reward functions for which to compute successor features (line 3 of Algorithm 1, discussed in Appendix F.1.1) and creating a reward feature vector ϕ for non-linear reward functions (discussed in Appendix F.1.2). Additionally, although learning with P_{regret} is more sample efficient in our experiments, it is computationally slower than learning with $P_{\Sigma r}$ because of the additional need to compute successor features and the use of the softmax function to approximate $Q^*_{\hat{r}}$ and $V^*_{\hat{r}}$. Nonetheless, we may accept the tradeoff of an increase in computational time that reduces the number of human samples needed or that improves the reward function's alignment with human stakeholders' interests. Lastly, the loss during optimization with P_{regret} was unstable, which we addressed by taking the minimum loss over all epochs during training. Therefore, for more complex reward feature vectors (ϕ) than our 6-element vector for the delivery task, extra care might be needed to avoid overfitting $\hat{r}$, for example by withholding some preference data to serve as a test set.

We also generally assume that the RL algorithm and reward learning algorithm use the same discount factor as in the MDP$\backslash r$ specification. One weakness of contemporary deep RL is that RL algorithms require artificially lower discount factors than the true discount factor of the task. The interaction of this discounting with preference models is considered in Appendix F.2. Our expectation though is that this weakness of deep RL algorithms is likely a temporary one, and so we focused our analysis on simple tasks in which we do not need to artificially lower the RL algorithm's discount factor. However, further investigation of the interaction between preference models and discount factors would aid near-term application of P_{regret} to deep RL domains.

This work also does not consider which segment pairs should be presented for labeling with preferences used for reward learning. However, other research has addressed this problem through active learning (Lee et al., 2021a; Christiano et al., 2017; Akrour et al., 2011), and it may be possible to simply swap our Algorithm 1 into these active learning methods, combining the improved sample efficiency of P_{regret} with that of these active learning methods.

Regarding the human side of the problem of reward learning from preferences, further research could provide several improvements. First, we are confident that humans can be influenced by their training and by the preference elicitation interface, which is a particularly rich direction for follow-up study. We also do not consider how to handle learning reward functions from multiple human stakeholders who have different preferences, a topic we revisit in Appendix A.2. Lastly, we expect humans to deviate from any simple model, including P_{regret}, and a fine-grained characterization of how humans generate preferences could produce preference models that further improve the alignment of the reward functions that are ultimately learned from human preferences.

A.2 Ethical statement

This work is meant to address ethical issues that arise when autonomous systems are deployed without properly aligning their objectives with those of human stakeholders. It is merely a step in that direction, and overly trusting in our methods—even though they improve on previous methods for alignment—could result in harm caused by poorly aligned autonomous systems.

When considering the objectives for such systems, a critical ethical question is *which* human stakeholders' interests the objectives should be aligned with and how multiple stakeholders' interests should be combined into a single objective for an autonomous system. We do not address these important questions, instead making the convenient-but-flawed assumption that many different humans' preferences can simply be combined. In particular, care should be taken that vulnerable and marginalized communities are adequately represented in any technique or deployment to learn a reward function from human preferences in high-impact settings. The stakes are high: for example, a reward function that is only aligned with a corporation's financial interests could lead to exploitation of such communities or more broadly to exploitation of or harm to users.

In this specific work, our filter for which Mechanical Turk Workers could join our study is described in Appendix D. We did not gather demographic information and therefore we cannot assess how representative our subjects are of any specific population.

A.3 On the challenge of using regret preference models in practice

We have provided evidence—theoretically and with experimentation—that the regret preference model is more effective when precisely measured or effectively approximated. The challenge of efficiently creating such approximations presents one clear path for future research. We believe this challenge does not justify staying within the local maximum of the partial return preference model.

Like the regret preference model, inverse reinforcement learning (IRL) was founded on an algorithm that requires solving an MDP in an inner loop of learning a reward function. For example, see the seminal work on IRL by Ng & Russell (2000). IRL has been an impactful problem despite this challenge, and handling this inner-loop computational demand is the focus of much IRL research.

Future work on the application of the regret preference model can face the challenge of scaling to more complex problems. Given that IRL has made tremendous progress in this direction and Brown et al. (2020) have scaled an algorithm with similar needs to those of Algorithm 1, we are optimistic that the methods to scale can be developed, likely with light adaptation from existing methods (e.g., in Brown et al. or in Appendix F.1.1 and F.1.2).

B Preference models for learning reward functions

Here we extend the content of Section 2, focusing on preference models and learning algorithms that use them. This corresponding section of the appendix provides a simple derivation of the logistic form of these preference models, discusses extensions of the regret preference model, sketches an alternative way to learn a policy with it, and discusses the relationship of inverse reinforcement learning to learning reward functions with a regret preference model.

B.1 Derivation of the logistic expression of the Boltzmann distribution

For the reader's convenience, below we derive the logistic expression of a function that is based on two subtracted values from the Boltzmann distribution (i.e., softmax) representation that is more common in past work. These values are specifically the same function f applied to each segment, which is a general expression of both of the preference models considered here.

$$\begin{aligned} P(\sigma_1 \succ \sigma_2) &= \frac{\exp\left[f(\sigma_1)\right]}{\exp\left[f(\sigma_1)\right] + \exp\left[f(\sigma_2)\right]} \\ &= \frac{1}{1 + \frac{\exp\left[f(\sigma_2)\right]}{\exp\left[f(\sigma_1)\right]}} \\ &= \frac{1}{1 + \exp\left[f(\sigma_2) - f(\sigma_1)\right]} \\ &= \mathit{logistic}(f(\sigma_1) - f(\sigma_2)). \end{aligned} \tag{7}$$

B.2 Learning reward functions from preferences, with discounting

For the equations from the paper's body that assume that there is no temporal discounting (i.e., $\gamma=1$), we share in this section versions that do not make this assumption. If $\gamma=1$, then the equations below simplify to those in the body of the paper. To allow for fully myopic discounting with $\gamma=0$, we define $0^0=1$.

Recall that $\tilde{r}$ indicates an arbitrary reward function, which may not be the ground-truth reward function, r, and $\hat{r}$ refers to a learned reward function. Similarly, $\tilde{\gamma}$ refers to an arbitrary exponential discount factor, which may not be the ground-truth discount factor, γ, and $\hat{\gamma}$ refers to the discount factor during learning, which could be inferred or hand-coded. Also, the notation of $V^*_{\tilde{r}}$ and $Q^*_{\tilde{r}}$ are expanded in this subsection to denote the discounting in their expected return: $V^*_{(\tilde{r},\tilde{\gamma})}$ and $Q^*_{(\tilde{r},\tilde{\gamma})}$, respectively.

In most of this article, the discount factor used during reward function inference is hard-coded as $\hat{\gamma}=1$. However, in the theory of Section 3, we assume γ is *not* known to reach more general conclusions. In this subsection, for generality we likewise assume that γ is not known, using $\tilde{\gamma}$ generally and using $\hat{\gamma}$ in notation we consider specific to reward function inference.

The discounted versions of the preference models below rely on a cross entropy loss function that is identical to Equation 1 except for the inclusion of discounting notation:

$$loss(\hat{r},\hat{\gamma},D_{\succ}) = -\sum_{(\sigma_1,\sigma_2,\mu)\in D_{\succ}} \mu_1 \log P(\sigma_1 \succ \sigma_2|\hat{r},\hat{\gamma}) + \mu_2 \log P(\sigma_1 \prec \sigma_2|\hat{r},\hat{\gamma}) \tag{8}$$

Partial return With discounting, the **partial return** of a segment σ is $\sum_{t=0}^{|\sigma|-1}\gamma^t\tilde{r}_{\sigma,t}$. This notation differs from that in Section 2.1 in that the subscript of the reward symbol $\tilde{r}_{\sigma,t}$ is now expanded to include which segment it comes from.

The preference model based on partial return *with exponential discounting* is expressed below, generalizing Equation 2.

$$P_{\Sigma r}(\sigma_1 \succ \sigma_2|\tilde{r},\tilde{\gamma}) = logistic\Big(\sum_{t=0}^{|\sigma_1|-1}\tilde{\gamma}^t\tilde{r}_{\sigma_1,t} - \sum_{t=0}^{|\sigma_2|-1}\tilde{\gamma}^t\tilde{r}_{\sigma_2,t}\Big). \tag{9}$$

Regret With discounting, for a transition (s_t,a_t,s_{t+1}) in a segment containing only deterministic transitions, $regret_{\mathrm{d}}(\sigma_t|\tilde{r},\tilde{\gamma}) \triangleq V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_t) - [\tilde{r}_t + \tilde{\gamma}V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_{t+1})]$.

For a full deterministic segment, $regret_d(\cdot|\tilde{r},\tilde{\gamma})$ with exponential discounting is defined as follows, generalizing Equation 3.

$$\begin{aligned} regret_d(\sigma|\tilde{r},\tilde{\gamma}) &\triangleq \sum_{t=0}^{|\sigma|-1}\tilde{\gamma}^t regret_d(\sigma_t|\tilde{r},\tilde{\gamma}) \\ &= V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_0) - \Big(\sum_{t=0}^{|\sigma|-1}\tilde{\gamma}^t\tilde{r}_{\sigma,t} + \tilde{\gamma}^{|\sigma|}V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_{|\sigma|})\Big), \end{aligned} \tag{10}$$

Like Equation 3, this discounted form of deterministic regret also measures how much the segment reduces expected return from the start state value, $V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_0)$.

To create the general expression of discounted regret that accounts for potential stochastic transitions, we note that, with discounting, the effect on expected return of transition stochasticity from a transition (s_t,a_t,s_{t+1}) is $[\tilde{r}_t + \tilde{\gamma}V^*_{(\tilde{r},\tilde{\gamma})}(s_{t+1})] - Q^*_{(\tilde{r},\tilde{\gamma})}(s_t,a_t)$ and add this expression once per transition to get $regret(\sigma|\tilde{r},\tilde{\gamma})$, removing the subscript d that refers to determinism. The discounting does not change the simplified expressions in Equation 4, the regret for a single transition:

$$\begin{aligned} regret(\sigma_t|\tilde{r},\tilde{\gamma}) &= [V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_t) - [\tilde{r}_t + \tilde{\gamma}V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_{t+1})]] + [[\tilde{r}_t + \tilde{\gamma}V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_{t+1})] - Q^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_t,a^\sigma_t)] \\ &= V^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_t) - Q^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_t,a^\sigma_t) \\ &= -A^*_{(\tilde{r},\tilde{\gamma})}(s^\sigma_t,a^\sigma_t). \end{aligned} \tag{11}$$

With both discounting and accounting for potential stochastic transitions, regret for a full segment is

$$\begin{aligned} regret(\sigma|\tilde{r},\tilde{\gamma}) &= \sum_{t=0}^{|\sigma|-1} \tilde{\gamma}^t regret(\sigma_t|\tilde{r},\tilde{\gamma}) \\ &= \sum_{t=0}^{|\sigma|-1} \tilde{\gamma}^t \left[V^*_{(\tilde{r},\tilde{\gamma})}(s_t^\sigma) - Q^*_{(\tilde{r},\tilde{\gamma})}(s_t^\sigma, a_t^\sigma) \right] \\ &= \sum_{t=0}^{|\sigma|-1} -\tilde{\gamma}^t A^*_{(\tilde{r},\tilde{\gamma})}(s_t^\sigma, a_t^\sigma). \end{aligned} \tag{12}$$

The expression of regret above is the most general in this paper and can be used in Equation 5 identically as can the undiscounted version in Equation 4.

Equation 6, the approximation $\tilde{P}_{regret}$ of the regret preference model derived in Section 6.1, is expressed with discounting below.

$$\tilde{P}_{regret}(\sigma_1 \succ \sigma_2|\hat{r},\hat{\gamma}) = logistic\left(\textstyle\sum_{t=0}^{|\sigma_2|\text{-}1} \hat{\gamma}^t \left[\tilde{V}^*_{(\hat{r},\hat{\gamma})}(s_t^{\sigma_2}) - \tilde{Q}^*_{(\hat{r},\hat{\gamma})}(s_t^{\sigma_2}, a_t^{\sigma_2}) \right] - \sum_{t=0}^{|\sigma_1|\text{-}1} \hat{\gamma}^t \left[\tilde{V}^*_{(\hat{r},\hat{\gamma})}(s_t^{\sigma_1}) - \tilde{Q}^*_{(\hat{r},\hat{\gamma})}(s_t^{\sigma_1}, a_t^{\sigma_1}) \right] \right) \tag{13}$$

Note that the successor features used in Section 6.1 to determine these approximations, $\tilde{V}^*_{(\hat{r},\hat{\gamma})}$ and $\tilde{Q}^*_{(\hat{r},\hat{\gamma})}$, already include discounting.

As with the undiscounted versions of the above equations, if two segments have deterministic transitions, end in terminal states, and have the same starting state, this regret model reduces to the partial return model: $P_{regret}(\cdot|\tilde{r},\tilde{\gamma}) = P_{\Sigma r}(\cdot|\tilde{r},\tilde{\gamma})$.

If hard-coding $\hat{\gamma}$, when to set $\hat{\gamma} < 1$ during reward function inference In reinforcement learning, both γ and r together determine the set of optimal policies. Changing either γ or r while holding the other constant will often change the set of optimal policies.

For both preference models, we suspect that learning would benefit from using the same discounting during reward inference as the human used while evaluating segments to provide preferences (i.e., setting $\hat{\gamma} = \gamma$. And this same $\hat{\gamma}$ would be used for learning a policy from the learned reward function. On the other hand, when $\hat{\gamma}$ is hand-coded and $\hat{\gamma} \neq \gamma$, the reward inference algorithm will regardless attempt to find an $\hat{r}$ that explains those preferences; however, a set of optimal policies is determined by a reward function *with the discount*, and the set of optimal polices created by the human's reward function and discounting may not be determinable under a different discounting.

Not only is a specific human rater's γ unobservable, but psychology and economics researchers have firmly established that humans do not typically follow exponential discounting (Frederick et al., 2002), which should evoke skepticism for hard-coding $\hat{\gamma} < 1$ during reward function inference. One exception is humans who have been trained to apply exponential discounting, such as in certain financial settings. The best model for how humans discount future rewards and punishments is not settled, but one popular model is hyperbolic discounting. Some exploration of RL with hyperbolic discounting exists, including approximating hyperbolically discounted value function using a mixture of exponentially discounted value functions (Kurth-Nelson & Redish, 2009; Redish & Kurth-Nelson, 2010). However, it has not found clear usage beyond as an auxiliary task to aid representation learning (Fedus et al., 2019). The interpretation of human preferences over segments appears to us to be a strong candidate for using these methods to approximate hyperbolic discounting.

This research topic currently lacks a rigorous treatment of discounting when learning reward functions from human preferences and such an investigation is beyond our scope, and so we leave our guidance above as speculative.

B.3 Logistic-linear preference model

In Appendices E.2, F.2.5, and F.3.2 we also consider preference models that arise by making the noiseless preference model a linear function over the 3 components of P_{regret_d}. Building upon Equation 7 above, we set

$f(\sigma) = \vec{w} \cdot \langle V^*_{\tilde{r}}(s^\sigma_0), \Sigma_\sigma, V^*_{\tilde{r}}(s^\sigma_{|\sigma|}) \rangle$. This preference model, $P_{log-lin}$, can be expressed after algebraic manipulation as

$$P_{log-lin}(\sigma_1 \succ \sigma_2 | \tilde{r}) = logistic\Big(\vec{w} \cdot \langle V^*_{\tilde{r}}(s^{\sigma_1}_0) - V^*_{\tilde{r}}(s^{\sigma_2}_0),\ \Sigma_{\sigma_1}\tilde{r} - \Sigma_{\sigma_2}\tilde{r},\ V^*_{\tilde{r}}(s^{\sigma_1}_{|\sigma_1|}) - V^*_{\tilde{r}}(s^{\sigma_2}_{|\sigma_2|}) \rangle\Big). \tag{14}$$

This logistic-linear preference model is a generalization of $P_{\Sigma r}$ and also of P_{regret_d}, the regret preference model for deterministic transitions. Specifically, if $\vec{w} = \langle 0,1,0 \rangle$, then $P_{log-lin}(\cdot|\tilde{r}) = P_{\Sigma r}(\cdot|\tilde{r})$. And if $\vec{w} = \langle -1,1,1 \rangle$, then $P_{log-lin}(\cdot|\tilde{r}) = P_{regret_d}(\cdot|\tilde{r})$. More generally, for some constant c, $\vec{w} = \langle 0,c,0 \rangle$ and $\vec{w} = \langle -c,c,c \rangle$ recreate $P_{\Sigma r}$ and P_{regret_d} respectively but with different reward function scaling, which is the same as allowing a different temperature in the Boltzmann distribution that determines preference probabilities. In Appendix E.2, we fit $\vec{w}$ to maximize the likelihood of the human preference dataset under $P_{log-lin}(\cdot|r)$, using the ground-truth r, and compare the learned weights to those of $P_{\Sigma r}$ and P_{regret_d}.

B.4 Adding a constant probability of uniformly distributed preference

Appendix E.2 also considers adaptations of $P_{\Sigma r}$, P_{regret_d}, and $P_{log-lin}$ that add a constant probability of uniformly distributed preference, as was done by Christiano et al. (2017). The body of the paper does not consider these adaptations.

We create this adaptation, which we will call P' here, from another preference model P by $P'(\sigma_1 \succ \sigma_2) = [(1 - logistic(c)) * P(\sigma_1 \succ \sigma_2)] + [logistic(c)/2]$, where c is a constant that in practice we fit to data and $logistic(c)$ is the constant probability of uniformly random preference. The $logistic(c)$ allows any constant c to result in a the constant probability of uniformly distributed preference to be in (0,1). The term $logistic(c)/2$ gives half of the constant probability to σ_1 and half to σ_2. The term $[1 - logistic(c)]$ scales the $P(\sigma_1 \succ \sigma_2)$ probability—which could be $P_{\Sigma r}$, P_{regret_d}, or $P_{log-lin}$—to a proportion of the remaining probability. The only difference in this adaptation and Christiano et al.'s 0.1 probability of uniformly distributed preference is that we learn the value of c from training data (in a k-fold cross-validation setting), as we see in Appendix E, whereas Christiano et al. do not share how 0.1 was chosen.

B.5 Expected return preference model

In Appendix F.3, we test reward learning on a third preference model. This expected return preference model is derived by making $f(\sigma) = -(\Sigma_\sigma \tilde{r} + V^*_{\tilde{r}}(s^\sigma_{|\sigma|}))$, in Equation 7. This segment statistic $f(\sigma)$ can be considered be in between deterministic regret (Equation 3) and partial return, differing from each by one term.

We include this preference model because judging by expected return is intuitively appealing in that it considers the partial return along the segment and the end state value of the segment, and we found it plausible that human preference providers might tend to ignore start state value, as this preference model does. However, reward learning with the regret model outperforms or matches that by this expected return preference model, as we show in Appendix F.3.

B.6 Relationship to inverse reinforcement learning

Like learning reward functions from pairwise preferences, inverse reinforcement learning (IRL) also involves learning a reward function. However, the inputs to IRL and learning reward functions from pairwise preferences are different: IRL requires demonstrations, not preferences over segment pairs. However, because a a regret-based preference model always prefers optimal segments over suboptimal segments, at least one further connection can be made. If one assumes that a demonstrated trajectory segment is noiselessly optimal—as in the foundational IRL paper on apprenticeship learning (Abbeel & Ng, 2004)—then such a demonstration is equivalent to expressing preference or indifference for the demonstrated segment over all other segments. In other words, no other segment is preferred over the demonstrated segment. However, IRL has its own identifiability issues in noiseless settings (e.g., see Kim et al. (2021)) that, viewed from the lens of preferences, come in part from the "indifference" part of the above statement: since there can be multiple optimal actions from a single state, it is not generally correct to assume that a demonstration of one such action shows a preference over all others,

and therefore it remains unclear in IRL what other actions are optimal. Note that since partial-return-based preferences can prefer suboptimal segments over optimal segments, the common assumption in IRL that demonstrations are optimal does not map as cleanly to partial-return-based preferences.

The regret preference model also relates to IRL in that the most basic version of IRL requires solving an MDP in the inner loop (see Algorithm 1 in the survey of IRL by Arora & Doshi (2021)), as appears necessary for a perfect measure of regret while learning a reward function. The progress that IRL has made addressing this challenge gives us optimism that it is similarly addressable for complex tasks in for our proposed algorithm. We discuss potential solutions in Appendix F.1.1 and F.1.2.

C Theoretical comparisons

The relevance of noiseless preference generators Because we model preferences as stochastic in Section 2, one might reasonably wonder how the above theoretical analysis of noiseless preference generators are relevant. We offer four arguments below.

First, having structured noise provides information that can help both preference models, but these proofs show that there are cases where the signal behind the noise—either regret or partial return—is not sufficient in the partial return case to identify an equivalent reward function. So, in a rough sense, regret more effectively uses both the signal and the noise, which might explain its superior sample efficiency in our experiments across both human labels and synthetic labels. Relatedly, the noiseless setting can help us understand each preference model's sample efficiency in a low-noise setting.

Second, noiseless preferences are also feasible, even if they are rare. Therefore, understanding what can be learned from them is worthwhile. Theorem 3.2 shows that there are MDPs in which there is *no* class of preference models—stochastic or deterministic—that can identify an equivalent reward function from partial-return-based preferences if the preference generator noiselessly prefers according to partial return. Specifically, we show that the mapping from two reward functions with different sets of optimal policies to partial-return based preferences is a many-to-one-mapping, and therefore the information simply does not exist to invert that mapping and identify a reward function with the same set of optimal policies. In contrast, Theorem 3.1 shows that preferences generated noiselessly (and in certain stochastic settings) by regret do not have this issue.

Third, noise is often motivated as modeling human error. Having an algorithm rely on noise—structured in a very specific, Boltzmann-rational way—is an undesirable crutch. Skalse et al. (2022) justify including noiseless preferences in their examinations of identifiability with a similar argument: "these invariances rely heavily on the precise structure of the decision noise revealing cardinal information in the infinite-data limit".

Beyond the work of Skalse et al. (2022), there is broader precedent for considering noiseless human input for theory or derivations. For instance, the foundational IRL research on apprenticeship learning (Abbeel & Ng, 2004) treats demonstrations as noiselessly optimal. Recent work by Kim et al. (2021) focuses on reward identifiability with noiseless, optimal demonstrations.

D Additional information for creating a human-labeled preference dataset

D.1 The preference elicitation interface and study overview

Here we share miscellaneous details about the preference elicitation interface from which we collected human subjects' preferences. This description builds on Section 4.2.

In selecting preferences, subjects had four options. They could prefer either trajectory (left or right), or they could express their preference to be the same or indistinguishable. To provide these preferences, subjects could either click on each of the buttons labeled "LEFT", "RIGHT", "SAME", or "CAN'T TELL" (shown in Figure 7) or by using the arrow keys to select amongst these choices.

For the interface, all icons used to visualize the task were obtained from icons8.com under their Paid Universal Multimedia Licensing Agreement.

We paid all subjects $5 per experiment (i.e., for each a Mechanical Turk HIT), which was chosen using the median time subjects took during a pilot study and then calculating the payment to result in $15 USD per hour. This hourly rate of $15 was chosen because it is commonly recommended as an improved US federal minimum wage. The human subject experiments cost $2,145 USD in total.

An experimental error resulted in the IRB-approved consent form not being presented to human subjects after Mechanical Turk Workers accepted our study. We reported this error to our IRB and received their approval to use the data.

Table 2: The task comprehension survey, designed to test participant's comprehension of the domain for the purpose of filtering data. Each full credit answer earned 1 point; each partial credit answer earned 0.5 points. We discarded the data of participants who scored less than 4.5 points overall.

Question	Full credit answer	Partial credit answer	Other answer choices
What is the goal of this world? (Check all that apply.)	• To maximize profit	• To get to a specific location. • To maximize profit Partial credit was given if both answers were selected.	• To drive as far as possible to explore the world. • To collect as many coins as possible. • To collect as many sheep as possible. • To drive sheep to a specific location.
What happens when you run into a house? (Check all that apply.)	• You pay a gas penalty. • You can't run into a house; the world doesn't let you move into it. Full credit was given if both answers were selected.	• You pay a gas penalty. • You can't run into a house; the world doesn't let you move into it. Partial credit was given if only one answer was selected.	• The episode ends. • You get stuck. • To collect as many sheep as possible.
What happens when you run into a sheep? (Check all that apply.)	• The episode ends. • You are penalized for running into a sheep. Full credit was given if both answers were selected.	• The episode ends. • You are penalized for running into a sheep. Partial credit was given if only one answer was selected.	• You are rewarded for collecting a sheep.
What happens when you run into a roadblock? (Check all that apply.)	• You pay a penalty.		• The episode ends. • You get stuck. • You can't run into a roadblock; the world doesn't let you move into it.
Is running into a roadblock ever a good choice in any town?	• Yes, in certain circumstances.		• No.
What happens when you go into the brick area? (Check all that apply.)	• You pay extra for gas.		• The episode ends. • You get stuck in the brick area. • You can't go into the brick area; the world doesn't let you move into it.
Is entering the brick area ever a good choice?	• Yes, in certain circumstances		• No

D.2 Filtering subject data

To join our study via Amazon Mechancial Turk, potential subjects had to meet the following criteria. They had to be located in the United States, have an approval rating of at least 99%, and have completed at least 100

other MTurk HITs. We selected these criteria to improve the probability of collecting data from subjects who would attentively engage with our study and who would understand our training protocol.

We assessed each subject's understanding of the delivery domain and filtered out those who did not comprehend the task, as described below. Specifically, subjects completed a task-comprehension survey, through which we assigned them a task-comprehension score. The questions and answer choices are shown in Table 2. Each fully correct answer was worth 1 point and each partially correct answer was worth 0.5 points. Task-comprehension scores were bounded between 0 and 7. We removed the data from subjects who scored below a threshold of 4.5. The threshold of 4.5 was chosen based on visual analysis of a histogram of scores, attempting to balance high standards for comprehension with retaining sufficient data for analysis.

In addition to filtering based off the task comprehension survey, we also removed a subject's data if they ever preferred colliding the vehicle into a sheep over not doing so. Since such collisions are highly undesirable in this task, we interpreted this preference as evidence of either poor task understanding or inattentiveness.

In total, we collected data from 143 subjects. Data from 58 of these subjects were removed based on subjects' responses to the survey. From what remained, data from another 35 subjects were removed for making the aforementioned sheep-collision preference errors. After this filtering, data from 50 subjects remained. This filtered data consists of 1812 preferences over 1245 unique segment pairs and is used in this article's experiments.

Regarding potential risks to subjects, this data collection had limited or no risk. No offensive content was shown to subjects while they completed the HIT. Mechanical Turk collected Worker IDs, which were used only to link preference data with the results from the task-comprehension survey for filtering data (see Appendix D.2) and then were deleted from our data. No other potentially personally identifiable information was collected.

D.3 The two stages of data collection

We collected the human preference dataset in two stages, as mentioned in Section 4.2. Here we provide more detail on each stage. These stages differed largely by their goals for data collection and, following those goals, how we chose which segment pairs were presented to subjects for their preference.

First stage Figure 12 illustrates the coordinates that segment pairs were sampled from in the first stage of data collection, varying by state value differences and by differences in partial returns over the segments. We sought a range of points that would allow a characterization of human preferences that is well distributed across different parts of the plot. To better differentiate the consequences of each preference model, we intentionally chose a large number of points in the gray area of Figure 8, where the regret and partial return preference models would disagree (i.e., each giving a different segment a preference probability greater than 0.5).

We now describe our segment-pair sampling process more specifically. We first we constructed all unique segments of length 3 and then exhaustively paired them, resulting in nearly 30 million segment pairs. Each segment pair's partial returns, start-state values, end-state values place the segment pair on a coordinate in Figure 8, and segment pairs that are not on any of the dots in Figure 8 were discarded. For the segment pairs at each coordinate, we further divided them into 5 bins: non-terminal segments with the same start state and different end states, non-terminal segments with different start states and different end states, terminal segments with the same start state and same end state, terminal segments with a different start states and the same end state, and bin of segment pairs that fit in none of the other bins. Segment pairs in the 5th bin were discarded. From each of the 4 bins corresponding to each point in Figure 8, we randomly sampled 20 segment pairs. If the bin did not have at least 20 segment pairs, all segment pairs in the bin were "sampled". All sampled segment pairs from all bins for all points in Figure 8 made up the pool of segment pairs used with Mechanical Turk. For each subject, 50 segment pairs were randomly sampled from this pool. We gathered data until we had roughly

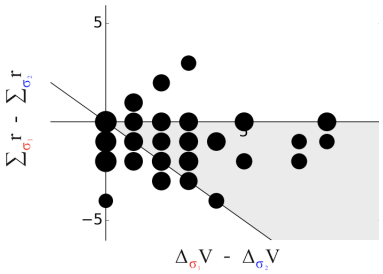

Figure 12: Coordinates from which segment pairs were sampled from during the first stage of data collection. The x-axis is state value differences between the two segments and the y-axis is partial return differences between the two segments. The areas of the circles are proportional to the number of samples at that point, and the proportionality is consistent across this plot and the 3 subplots of Figure 13.

20 labeled segment pairs per bin. After filtering subject data, this first stage contributed 1437 segment pairs out of the 1812 pairs used in our reward learning experiments in Section 6.3 and Appendix F.3.

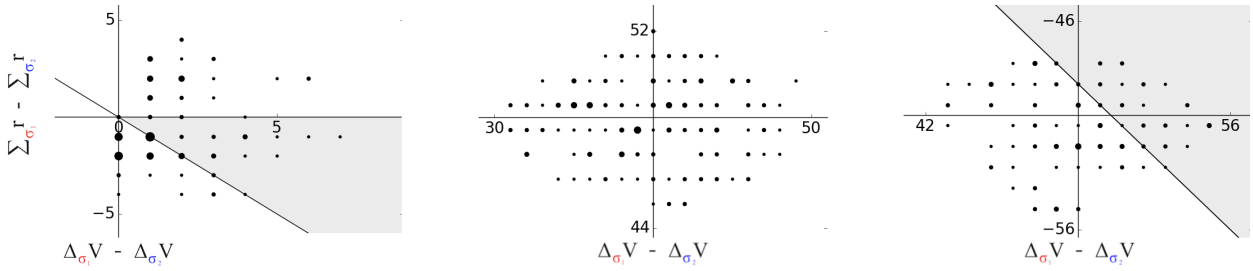

Figure 13: Coordinates from which segment pairs were sampled from during the second stage of data collection. The points are in 3 distant clusters, so they are presented in 3 separate subplots for readability. The areas of the circles are proportional to the number of samples at that point, and the proportionality is consistent across these 3 subplots and Figure 12.

Second stage When we conducted the reward-learning evaluation in Section 6 with only the data from the first stage, $P_{\Sigma r}$ performed very poorly, always performing worse than uniformly random. This performance difference is shown in Appendix F.3. In contrast P_{regret} performed well, always achieving near-optimal performance. To better assess $P_{\Sigma r}$, we investigated its results with synthetic preferences in detail and speculated that two types of additional segment pairs would aid its performance. The first of these two types include one segment that is terminal and one that is non-terminal, which we expected to help differentiate the reward for reaching terminal states from that of reaching non-terminal ones.

The second of these two types are two segments that each terminate at different t values. For example, one segment terminates on its end state, $s^{\sigma}_{|\sigma|}$, and another terminates after its first transition, at s^{σ}_{1}. These early-terminating segments can be viewed either as shorter segments or as segments of the same length as the other segments ($|\sigma|=3$), where they reach absorbing state from which no future reward can be received. We speculated that this second type of segment pairs would help learn the negative reward component for each move (i.e., the gas cost). Specifically, in the first stage's data, both segments in a pair always have the same number of non-terminal transitions, seemingly preventing preferences from providing information about whether an extra transition (from non-absorbing state) generally resulted in positive or negative reward. These segment pairs were included in all results unless otherwise stated. Note that this second type addresses the identifiability issue of the partial return model related to a constant shift in the reward function and discussed in Section 3.2.2 and Appendix F.2.2. The speculation described above was a conceptual predecessor of our understanding of this identifiability issue. In particular, any change to this gas cost—which is given at every time step—is equivalent to a constant shift in the reward function.

We now describe our segment-pair sampling process for the second stage more specifically. For the first additional type of segment pair, where one segment is terminal and one is not, we randomly pair terminal and non-terminal segments from the first-stage pool of segment pairs drawn from to present to subjects. In this pairing, each segment is only used once, and pairing stops when one of all terminal segments or all non-terminal segments have been paired. The corresponding coordinates for these pairs are shown in the two right most plots of Figure 13. For the second additional type of segment pair, we utilize all terminal segments from the pool of segment pairs shown to subjects in the first stage. For each of these terminal segments, we construct two additional segments: one that shifts the segment earlier, removing the first state and action and adds a dummy transition within absorbing state at the end, and another that shifts the segment two timesteps earlier and adds two such dummy transitions at the end. These two newly constructed segments are then each paired with the original segment, producing two new pairs for each terminal segment in the data set. The corresponding coordinates for these segment pairs are shown in the left most plot of Figure 13.

All of both types of additional segments pairs are then characterized by the coordinates shown in Figure 13. Then, as with the first stage, we randomly sampled 20 segment pairs from each coordinate to make the experimental pool for the second round of Mechanical Turk data collection. If 20 segment pairs were not available at a coordinate, we used all segment pairs for that coordinate. As in the first stage, 50 segment pairs were randomly

sampled from this pool to be presented to each subject during preference elicitation. After filtering subject data, this first stage contributed 311 segment pairs out of the 1812 pairs used in our reward learning experiments in Section 6.3 and Appendix F.3.

D.4 The study design pattern

This work follows an experimental design pattern that is often used for studying methods that take human input for evaluating the desirability of behaviors or outcomes. This pattern is illustrated for the specific case of learning reward functions from preferences in Figure 9. In this pattern, human subjects are taught to understand a specific task metric and/or are incentivized to align their desires with this metric. The human subjects then provide input to some algorithm that has no knowledge of the performance metric, and this algorithm or learned model is evaluated on how well its output performs with respect to the hidden metric. For another example, see Cui et al. (2020).

E Descriptive results

E.1 Derivation of $regret_{\mathbf{d}}(\sigma_2|\tilde{r}) - regret_{\mathbf{d}}(\sigma_1|\tilde{r}) = (\Delta_{\sigma_1} V_{\tilde{r}} - \Delta_{\sigma_2} V_{\tilde{r}}) + (\Sigma_{\sigma_1}\tilde{r} - \Sigma_{\sigma_2}\tilde{r})$

The derivation below supports our assertion in the first paragraph of Section 5.1.

$$
\begin{aligned}
&regret_{\mathrm{d}}(\sigma_2|\tilde{r}) - regret_{\mathrm{d}}(\sigma_1|\tilde{r}) \\
&= \Big([V^*_{\tilde{r}}(s_0^{\sigma_2}) - (\Sigma_{\sigma_2}\tilde{r} + V^*_{\tilde{r}}(s_{|\sigma_2|}^{\sigma_2})] - [V^*_{\tilde{r}}(s_0^{\sigma_1}) - (\Sigma_{\sigma_1}\tilde{r} + V^*_{\tilde{r}}(s_{|\sigma_1|}^{\sigma_1})] \Big) \\
&= \Big([V^*_{\tilde{r}}(s_0^{\sigma_2}) - V^*_{\tilde{r}}(s_{|\sigma_2|}^{\sigma_2})] - [V^*_{\tilde{r}}(s_0^{\sigma_1}) - V^*_{\tilde{r}}(s_{|\sigma_1|}^{\sigma_1})] \Big) - \Big(\Sigma_{\sigma_2}\tilde{r} - \Sigma_{\sigma_1}\tilde{r} \Big) \\
&= \Big([V^*_{\tilde{r}}(s_{|\sigma_1|}^{\sigma_1}) - V^*_{\tilde{r}}(s_0^{\sigma_1})] - [V^*_{\tilde{r}}(s_{|\sigma_2|}^{\sigma_2}) - V^*_{\tilde{r}}(s_0^{\sigma_2})] \Big) + \Big(\Sigma_{\sigma_1}\tilde{r} - \Sigma_{\sigma_2}\tilde{r} \Big) \\
&= (\Delta_{\sigma_1} V_{\tilde{r}} - \Delta_{\sigma_2} V_{\tilde{r}}) + (\Sigma_{\sigma_1}\tilde{r} - \Sigma_{\sigma_2}\tilde{r})
\end{aligned}
\tag{15}
$$

E.2 Losses of an expanded set of preference models on the human preferences dataset

Table 3 shows an expansion of Table 1, including models introduced in Appendix B. The logistic linear preference model, $P_{log-lin}$, provides a lower bound, given that it can express either P_{regret} or $P_{\Sigma r}$ and that we do not not observe any overfitting of its 3 parameters.

Table 3: Expanding on Table 1, mean cross-entropy test loss over 10-fold cross validation (n=1812) from predicting human preferences. Lower is better.

Preference model	**Loss (n=1,812)**
$P(\cdot) = 0.5$ (uninformed)	0.693
$P_{\Sigma r}$ (partial return)	0.620
P_{regret} (regret)	0.573
$P_{log\text{-}lin}$ (logistic linear)	0.548
$P_{\Sigma r}$ with prob of uniform response	0.620
P_{regret} with prob of uniform response	0.573
$P_{log\text{-}lin}$ with prob of uniform response	0.548

Including the constant probability of a uniformly random response, as in Christiano et al. (2017), also increases the expressivity of the model. The final three results in Table 3 show the best test loss achieved across different training runs that differ by initializing $logistic(c)$ of this model to be 0.01, 0.1, or 0.5. Surprisingly, no benefit is observed from including a constant probability of a uniformly random response. Because this augmentation of our models appears to have no effect on the likelihood, we do not include it in further analysis.

Across the 10 folds, the mean weights learned for $P_{log-lin}(\cdot|\tilde{r})$ were $\vec{w} = \langle -0.18, 0.34, 0.32 \rangle$, where each weight applies respectively to segments' start state values, partial returns, and end state values. Scaled to have a maximum weight of 1 for easy comparison with $P_{\Sigma r}$ and P_{regret_d}, $\vec{w}_{scaled} = \langle -0.53, 1.0, 0.94 \rangle$. First, we note that these weights are closer to those that make $P_{log-lin} = P_{regret_d}$ (i.e., $\vec{w} = \langle -1,1,1 \rangle$) than to those that make $P_{log-lin} = P_{\Sigma r}$ (i.e., $\vec{w} = \langle 0,1,0 \rangle$). Also, the notable deviation from the weights of P_{regret_d} is the weight for the start state value, which has half as much impact as the regret preference model gives it. In other words, this lower weight suggests that our subjects did tend to weigh the maximum possible expected return from each segment's start state, but they did so less than they weighed the reward accrued along each segment and the maximum expected return from each segment's end state.

F Results from learning reward functions

This section provides additional implementation details for Section 6, discussion of potential improvements, and additional analyses that thematically fit in Section 6.

F.1 An algorithm to learn reward functions with $regret(\sigma|\hat{r})$

We describe below additional details of our instantiation of Algorithm 1.

Doubling the training set by reversing preference samples Because the ordering of preference pairs is arbitrary—i.e., $(\sigma_1 \prec \sigma_2) \Longleftrightarrow (\sigma_2 \succ \sigma_1)$—for all preference datasets we double the amount of data by duplicating each preference sample with the opposite ordering and the reversed preference. This provides more training data and avoids learning any segment ordering effects.

Collecting the policies from which successor feature functions are calculated For this article's instantiation of Algorithm 1, we collect successor feature functions by randomly sampling a large number of reward functions and then calculating the successor feature functions for their optimal polices. This procedure is more precisely described below.

1. Create a reward function by sampling with replacement each element of its weight vector, $\boldsymbol{w}_{\tilde{\boldsymbol{r}}}$, from $\{-50,-10,-2,-1,0,1,5,10,50\}$.
2. For this reward function, use value iteration to approximate its maximum entropy optimal policy and that policy's successor feature function.
3. If this successor feature function policy differs from all previously calculated successor feature functions, save it and go to step 1.
4. Otherwise it is a redundant policy. If less than 300 consecutive redundant policies have been found, go to step 1.

The policy collection process above is terminated after 300 consecutive redundant policies are found. Finally, we calculate the maximum entropy optimal policy for the optimal policy for the ground-truth reward function, r, and *remove the successor feature function for any policy that matches the optimal policy for* r. In other words, we remove any policies for other reward functions that were also optimal for r, making the regret-based learning problem more difficult. We ensured that the ground-truth reward function was not represented to better approximate real-world reward learning applications, in which one would be unlikely to have the optimal policy for learning a successor features function. On the specific delivery task on which we gathered human preferences, the process above resulted in 70 reward functions.

Early stopping without a validation set During training, the loss for the P_{regret} model tended to show cyclical fluctuations, reaching low loss and then spiking. To handle this volatility, we used the $\hat{r}$ that achieved the lowest loss over all epochs of training, not the final $\hat{r}$. For $P_{\Sigma r}$ and P_{regret}, we found no evidence of overfitting with our linear representation of the reward function, but with a more complex representation, such early stopping likely should be based upon the loss of the model on a validation set. A better understanding of the cyclical loss fluctuations we observe during training could further improve learning with P_{regret}.

Discounting during value iteration Despite the delivery domain being an episodic task, a low-performing policy can endlessly avoid terminal states, resulting in negative-infinity values for both its return and successor features based on the policy. To prevent such negative-infinity values, we apply a discount factor of $\gamma = 0.999$ during value iteration—which is also where successor feature functions are learned—and when assessing the mean returns of policies with respect to the ground-truth reward function, r. We chose this high discount factor to have negligible effect on the returns of high-performing policies (since relatively quick termination is required for high performance) while still allowing value iteration to converge within a reasonable time.

Hyperparameters for learning Below we describe the other specific hyperparameters used for learning a reward function with both preference models. These hyperparameters were used across all experiments. For all models, the learning rate, softmax temperature, and number of training iterations were tuned on the noiseless synthetic preference data sets such that each model achieved an accuracy of 100% on our specific delivery task and then were tuned further on stochastic synthetic preferences on our specific delivery task.

Reward learning with the partial return preference model
learning rate: 2; number of training epochs: 30,000; and optimizer: Adam (with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and eps= $1e-08$).

Reward learning with the regret preference model
learning rate: 0.5; number of training epochs: 5,000; optimizer: Adam (with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and eps=$1e-08$); and softmax temperature: 0.001.

Logistic regression with both preference models, for the likelihood analysis in Section 5.2 and Appendix E.2
learning rate: 0.5; number of training iterations: 3,000; optimizer: stochastic gradient descent; and evaluation: 10-fold cross validation.

Computer specifications and software libraries used The computer used to run experiments shown in Figures 15, 16, 17, 18, 19, 21, 22, and 23 had the following specification. Processor: 2x AMD EPYC 7763 (64 cores, 2.45 GHz); Memory: 284 GB.

The computer used to run all other experiments had the following specification. Processor: 1x Core™ i9-9980XE (18 cores, 3.00 GHz) & 1x WS X299 SAGE/10G | ASUS | MOBO; GPUs: 4x RTX 2080 Ti; Memory: 128 GB.

Pytorch 1.7.1 (Paszke et al., 2019) was used to implement all reward learning models, and statistical analyses were performed using Scikit-learn 0.23.2 (Pedregosa et al., 2011).

F.1.1 For Algorithm 1, choosing an input set of policies for learning successor feature functions

A set of policies is input to Algorithm 1 and used to create successor feature functions, which are in turn used for generalized policy improvement (GPI) to efficiently estimate optimal value functions for $\hat{r}$ during learning. Which policies to insert is an important open question for successor-feature-based methods in general, but our intuition is that the performance of GPI under successor-feature-based methods is improved with a greater diversity of successor feature functions (via a diverse set of policies) and by having some policies that perform decently (but not necessarily perfectly) on the reward functions for which V^* and Q^* outputs values are being estimated via GPI.

Recalling that an input policy can come from policy improvement with an arbitrary reward function, we offer the following ideas for how to source an input set of policies.

- Choose reward function parameters according to some random distribution (as we do).
- Create a set of reward functions that differ in a structured way, such as each reward function being a point in a grid formed in parameter space.
- Learn policies from a separate demonstration dataset, using an imitation learning algorithm.
- Bootstrap from $\hat{r}$. Specifically, during learning augment the current set of successor feature functions ($\Psi_Q^{\pi_{SF}}$ and $\Psi_V^{\pi_{SF}}$) by learning one new successor feature function via policy improvement on the current $\hat{r}$; then continue reward learning with the augmented set of successor feature functions and repeat this process as desired.

The input set of policies could come from multiple different sources, including the ideas above.

F.1.2 Instantiating Algorithm 1 for reward functions that may be non-linear

Algorithm 1 operates under the assumption that the reward function can be represented as a linear combination of reward features. These reward features are obtained through a reward-features function $\boldsymbol{\phi}$, which is given as input to the algorithm. Here we address situations when the linearity assumption does not hold.

If the state and action space are discrete, one could linearly model all possible (deterministic) reward functions by creating a feature for each (s,a,s') that is 1 for (s,a,s') and 0 otherwise. A downside of this approach is that the learned reward function cannot benefit from generalization, which has two negative consequences. First, in complex tasks, generalization would typically have decreased the training set size required to learn a reward function $\hat{r}$ with optimal policies that perform well on the ground-truth reward function, r. Second, the reward function will not generalize to different tasks that share the same *symbolic* reward function, such as always having the same reward from interacting with a particular object type.

If the reward features are unknown or the reward is known to be non-linear, another method is to create a reward features function that permits a linear *approximation* of the reward function. Several methods to derive some or all of these reward features appear promising:

- Reward features can be learned by minimizing several auxiliary losses in a self-supervised fashion, as by Brown et al. (2020). After optimizing for these various objectives using a single neural network, the activations of the penultimate layer of this network can be used as reward features. Such auxiliary tasks may include minimizing the mean squared error of the reconstruction loss for the current state from a lower-dimensional embedding and the original state, predicting how much time has passed between states by minimizing the mean squared error loss (i.e., learning a temporal difference model), predicting the action taken between two states by minimizing the cross entropy loss (i.e., learning an inverse dynamics model), predicting the next state given the current state and action by minimizing the mean squared error loss(i.e., learning a forward dynamics model), and predicting which of two segments is preferred given a provided ranking by minimizing the t-rex loss.
- Reward features could also be learned by first learning a reward function represented as a neural network *using a partial return preference model*, and then using the activations of the penultimate layer of this neural network to provide reward features.

F.2 Results from synthetic preferences

F.2.1 Learning reward functions from 100 randomly generated MDPs

Here we describe how each MDP in the set of 100 MDPs discussed in section 6.2 was generated. We also extend the analysis to illustrate how often each preference model performs better than uniformly random and give further details on our statistical tests.

Design choices The 100 MDPs are all instances of the delivery domain, but they have different reward functions. The height for each MDP is sampled from the set {5,6,10}, and the width is sampled from {3,6,10,15}. The proportion of cells that are terminal failure states is sampled from the set {0,0.1,0.3}. There is always exactly one terminal success state. The proportion of "mildly bad" cells were selected from the set {0,0.1,0.5,0.8}, and the proportion of "mildly good" cells were selected from {0,0.1,0.2}. Mildly good cells and mildly bad cells respectively correspond to cells with coins and roadblocks in our specific delivery task, but the semantic meaning of coins and roadblocks is irrelevant here. Each sampled proportion is translated to a number of cells (rounding down to an integer when needed) and then cells are randomly chosen to fill the grid with each of the above types of states until the proportions are satisfied.

Then, the ground-truth reward component for each of the above cell types were sampled from the following sets:

- Terminal failure states: {0,1,5,10,50}
- Terminal success states: {−5,−10,−50}

- Mildly bad cells: $\{-2,-5,-10\}$

Mildly good cells always have a reward component of 1, and the component for white road surface cells is always -1. There are no cells with a higher road surface penalty (analogous to the bricks in the delivery domain).

Better than random performance Figure 14 complements the results in Figure 10, showing the percentage of MDPs in which each preference model outperforms a policy that chooses actions according to a uniformly random probability distribution. We can see that at this performance threshold, lower than that in Figure 10, the regret preference model outperforms the partial return preference model in most conditions. Even when their performance in this plot—based on outperforming uniformly random actions—is nearly identical, Figure 10 shows that the regret preference model achieves near optimal performance at a higher proportion.

Details for statistical tests We performed a Wilcoxon paired signed-rank test on the normalized average returns achieved by each model over the set of 100 randomly generated MDPs. All normalized average returns below -1 were replaced with -1, so that all such returns were in the range $[-1,1]$. This clipping was done because any normalized average return below 0 is worse than uniformly random, so the difference between a normalized return of -1 and -1000 is relatively unimportant compared to the difference between 1 and 0. Results are shown in Table 4.

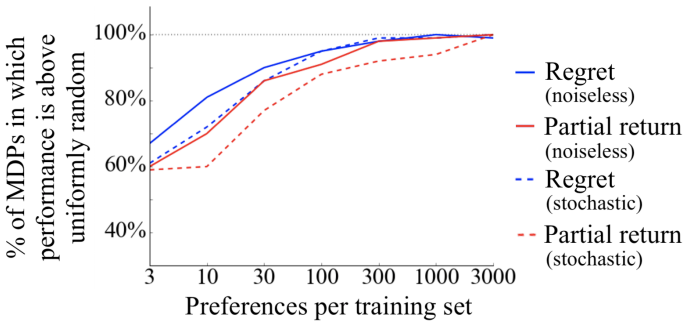

Figure 14: Comparison of performance over 100 randomly generated deterministic MDPs, showing the percentage of MDPs in which each model performed better than an agent taking actions by a uniformly random policy. This plot complements Figure 10, which shows the percentage of MDPs in which the models perform near-optimally.

Table 4: Results of the Wilcoxon paired signed-rank test on normalized average returns for each preference model.

Preference generator type	$\lvert D_{\succ}\rvert=3$	$\lvert D_{\succ}\rvert=10$	$\lvert D_{\succ}\rvert=30$	$\lvert D_{\succ}\rvert=100$	$\lvert D_{\succ}\rvert=300$	$\lvert D_{\succ}\rvert=1000$	$\lvert D_{\succ}\rvert=3000$
Noiseless (P_{regret} vs. $P_{\Sigma r}$)	w=1003, p=0.115	w=917, p=0.007	w=739, p=0.012	w=487, p=0.007	w=284, p<0.001	w=301, p=0.002	w=289, p=0.001
Stochastic (P_{regret} vs. $P_{\Sigma r}$)	w=979, p=0.541	w=1189.5, p=0.018	w=891, p=0.027	w=710, p=0.018	w=285, p<0.001	w=460, p=0.002	w=199, p<0.001

Additionally, we investigate whether P_{regret} and $P_{\Sigma r}$ learn near-optimal policies on the same MDPs within this set of 100 randomly generated MDPs. Results for this analysis are shown below.

Table 5: A table showing the count of the number of MDPs where both, either, or neither of the models achieved near optimal performance.

Model(s)	$\lvert D_{\succ}\rvert=3$	$\lvert D_{\succ}\rvert=10$	$\lvert D_{\succ}\rvert=30$	$\lvert D_{\succ}\rvert=100$	$\lvert D_{\succ}\rvert=300$	$\lvert D_{\succ}\rvert=1000$	$\lvert D_{\succ}\rvert=3000$
Both models	31	40	66	72	83	87	88
Only P_{regret}	20	26	17	18	14	8	8
Only $P_{\Sigma r}$	10	12	7	8	3	3	3
Neither	39	22	10	2	0	2	1

F.2.2 The effect of including transitions from absorbing state

In Section 3.2.2 we describe one context in which partial return is not identifiable because reward functions that differ by a constant can have different sets of optimal policies yet will have identical preference probabilities according to the partial return preference model (via Eqs. 2 and 9). This lack of identifiability arises specifically

in tasks that have the characteristic of having at least one state from which trajectories of *different* lengths are possible. Tasks that terminate upon completing a goal or reaching a failure state typically have this characteristic. Below we describe an imperfect method to remove this lack of identifiability. Then we show that the partial return preference model does much worse in our main experiments with synthetic preferences and human preferences when this method is not applied.

An imperfect method to prevent this source of unidentifiability for partial return Put simply, the approach to prevent all constant shifts of reward from resulting in the same preference probabilities is to force $\hat{r}$(s,a,s')=0 in at least one tuple of state, action, and next state, (s,a,s'), and include in the training dataset one or more segments with that tuple.

Technically, this solution addresses the source of this identifiability issue, that any constant shift in the output of the reward function will not change the likelihood of a preferences dataset (but can change the set of optimal policies). With this solution, a constant shift cannot be applied to all outputs, since at least one reward output is hard-coded to 0. And applying a constant shift to all other outputs would change the likelihood of the infinite, exhaustive preferences dataset that is assumed in identifiability theory.

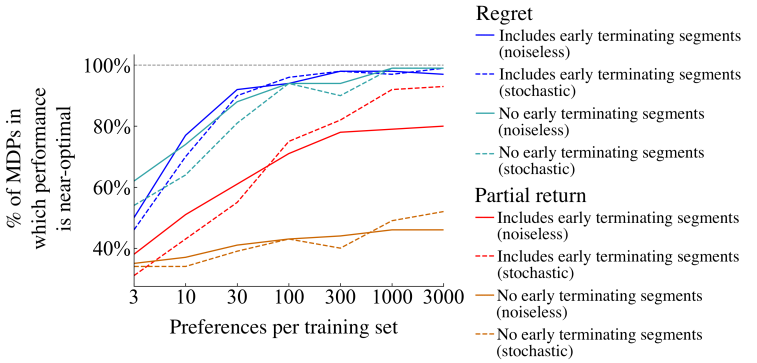

Figure 15: Performance comparison over 100 randomly generated deterministic MDPs. The results in this plot expand upon the experiments in Figure 10, adding results for datasets that do not have any segments that terminate early.

More intuitively, setting one $\hat{r}(s,a,s')=0$ for one (s,a,s') tuple anchors the reward function. To explain, reward function inference effectively learns an ordering over (s,a,s') tuples. Let us arbitrarily assume this ordering is in ascending order of preference. Then setting the reward for one such tuple to 0 forces all (s,a,s') tuples that are later in the ordering to have positive reward and all (s,a,s') tuples that are earlier in the ordering to have negative reward. (s,a,s') tuples that are equal in the ordering are assigned a reward of 0.

However, assuming that reward is 0 in any state has at least two undesirable properties. First, it requires adding human task knowledge that is beyond what the preferences dataset contains, technically changing the learning problem we are solving. Second, while it resolves some ambiguity regarding what reward function has the highest likelihood, this resolution may not actually align with the ground-truth reward function. In an attempt to reduce the impact of these undesirable properties, we only set the reward for *absorbing* state to 0. Absorbing state is the theoretical state that an agent enters upon terminating the task. All actions from the absorbing state merely transition to the absorbing state.

In practice, to include segments with transitions from absorbing state, we add *early terminating* segments, meaning that termination in an n-length segment occurs before the final transition.

The method above is not a perfect fix, however. Assuming that transitions from absorbing state have 0 reward also consequently assumes that humans consider time after termination to have 0 return. We are uncertain that humans will always provide preferences that are aligned with this assumption. For instance, if termination frees the agent to pursue other tasks with positive reward, then we might be mistaken to assume that humans are giving preferences as if there is only 0 reward after termination.

As mentioned in Section 3.2.2, past authors have acknowledged this issue with learning reward functions under the partial return preference model, apparently unaware of this solution of including transitions from absorbing states. Instead, they have used normalization (Christiano et al., 2017; Ouyang et al., 2022), tanh activations (Lee et al., 2021a), and L2 regularization (Hejna & Sadigh, 2023) that resolve their algorithms' insensitivity to constant shifts in reward. However, these papers do not discuss what assumptions regarding alignment are implicitly made by these ambiguity-resolving choices. Curiously, artificially forcing episodic tasks to be fixed horizon is common (e.g., as done by Christiano et al. (2017, p. 14) and Gleave et al. (2022)) but has not, to

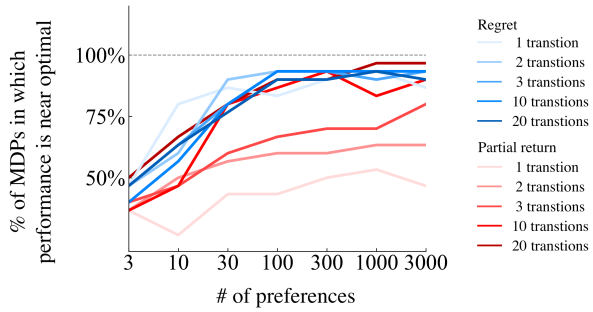

Figure 16: Using noiselessly generated preferences, comparison of performance over 100 randomly generated deterministic MDPs, showing the percentage of MDPs in which each model performed near-optimally after learning from preference datasets of different segment lengths.

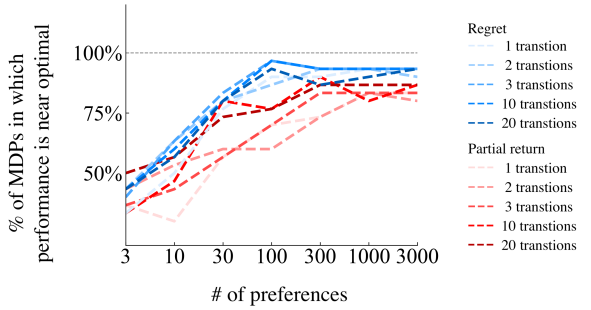

Figure 17: Using stochastically generated preferences, comparison of performance over 100 randomly generated deterministic MDPs, showing the percentage of MDPs in which each model performed near-optimally after learning from preference datasets of different segment lengths.

our knowledge, been justified as a way to address this identifiability issue. Our solution above involving early termination could be cast as a specific form of forcing a fixed horizon (infinite in this case). In one recent analysis of reward identifiability, Skalse et al. (2022) make assumptions that are identical to our solution above but do not discuss their assumptions' relationship to the partial return preference model otherwise being insensitive to constant shifts in reward when the lengths of segments in pairs are the same. We advise researchers, when forcing a fixed horizon via other methods, to explain what bias they are introducing regarding the set of optimal policies, either for generating synthetic preferences or during reward function inference.

Empirical results Figure 15 expands upon the synthetic-preferences results of Figure 10 in Section 6.2, adding results for datasets that lack segments that terminate early. For each combination of preference model and whether early terminating segments are included (i.e., for each different color in Figure 15), we used a learning rate chosen from testing on a set of 30 MDPs. Specifically, we chose the learning rate from the set $\{0.005, 0.05, 0.5, 1, 2\}$ that had the highest mean percentage of near-optimal performance on these 30 MDPs across training sets of 30, 300, and 3000 preferences. These 30 MDPs were taken from the 100 randomly generated MDPs used in Section 6.2. So, for testing with the chosen learning rates, we replaced them with 30 new MDPs generated by the same sampling procedure, determining the 100 MDPs used for these results. Also, a different random seed is used here than in Section 6.2. In these results shown in Figure 15, performance by the partial return model is worse without early terminating segments, whereas the regret model does not appear to be sensitive to their inclusion.

The effect of removing such early-terminating segments from the *human* preferences dataset is tested in Appendix F.3.4. There we similar results: only performance with the partial return model is harmed by removing segment pairs with early-terminating segments and the decrease in performance is severe.

F.2.3 Varying segment length

Here we consider empirically the effect of different fixed lengths of the segments in the preference data set. All other experiments in this article assume the segment length $|\sigma| = 3$, and this analysis is a limited investigation into whether changing $|\sigma|$ affects results. In general, we suspect that increasing segment length has a trade off: it makes credit assignment within the segment more difficult yet also gives preference information about more unique and complex combinations of events, which could reduce the probability of the preference providing new information that is not already contained in the remainder of the training dataset. Put more simply, preferences over larger segments appear to give information that is harder to use but covers more transitions.

To conduct this analysis, the following process is followed for each of 30 MDPs that were sampled as described in Appendix F.2.1. For each $n \in \{1, 2, 3, 4, 6, 10, 20\}$, we synthetically create preference datasets with segment length $|\sigma| = n$. Each segment was generated by choosing a non-terminal start state and n actions, all uniformly randomly. As in Appendix F.2.1, each preference model acts as a preference generator to label these segment

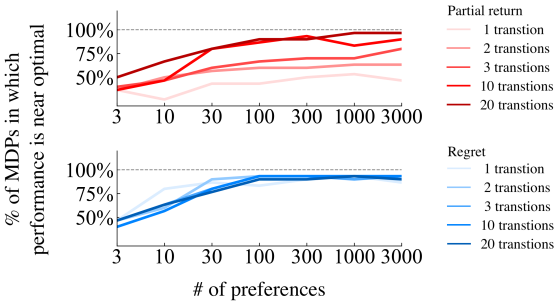

Figure 18: The results from Figure 16 (noiselessly generated preferences) divided by preference model, allowing further perspective.

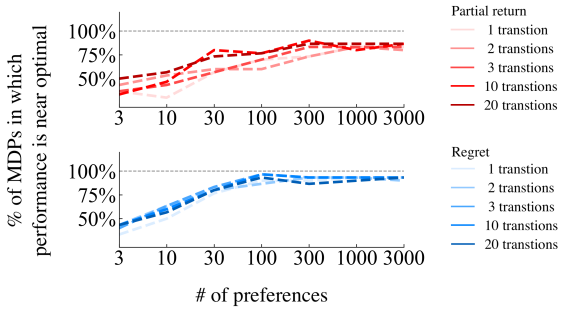

Figure 19: The results from Figure 17 (stochastically generated preferences) divided by preference model, allowing further perspective. Segment lengths 4 and 6 are added here.

pairs, resulting in datasets that differ only in their labels, and then each preference model is used for reward learning on the same dataset it labeled.

We observe the following:

- With noiselessly generated preferences (Figure 16), the performance with each preference model is similar for segments with 20 transitions, though it is sometimes slightly better with the partial return preference model. At a segment length of 10, performance with the regret preference model is better or similar, depending on the number of preferences. For all other segment lengths, performance with the regret preference model is better.
- With stochastically generated preferences (Figure 17), the performance with each preference model is generally better with the regret preference model, although sometimes the partial return preference model has marginally better performance. Additionally, the variance of performances with the partial return preference model is higher.

Additionally, we conduct Wilcoxon paired signed-rank tests for a positive effect of segment length on performance. Four tests are conducted, one per combination of preference model and whether preferences are generated noiselessly or stochastically. The steps to conduct this test are below. For each combination of preference model and each preference dataset of size $|D_{\succ}| \in \{3,10,30,100,300,1000,3000\}$, we calculate Kendall's τ correlation measures between the following two orderings:

- segment lengths in ascending order, (1,2,3,10,20), and
- segment lengths in the ascending order of the corresponding percentage of MDPs in which near-optimal performance was achieved (e.g., if performance is near-optimal in 90% of MDPs for $|\sigma| = 20$ and is near-optimal in 60% of MDPs for $|\sigma| = 10$, then $|\sigma| = 20$ would be later in the ordering than $|\sigma| = 10$).

For each of the resultant 7 τ values—one for each $|D_{\succ}|$—we create a pair for a Wilcoxon paired signed-ranked test: $(\tau,0)$. The 0 in the pair is the expected Kendall's τ value for uniformly random orderings of segment lengths. Therefore, the pair represents a comparison between the correlation between a training dataset's segment length and its performance and no correlation. Each Wilcoxon paired signed-rank test is conducted on these 7 pairs.

- With noiselessly generated preferences and the partial return preference model (Figure 18), $p = 0.016$ and the mean τ across the 7 training dataset sizes is 0.80, indicating a significant and large correlation between segment length and ordering by performance. Our visual inspection of Figure 18 shows that the effect size is large, so we conclude that in this experiment increasing segment length meaningfully improves performance.

- With stochastically generated preferences and the partial return preference model and (Figure 19), $p=0.016$ and the mean τ across the 7 training dataset sizes is 0.51, indicating a significant and moderate correlation between segment length and ordering by performance. Our visual inspection of Figure 19 shows that the effect size is smaller but still observable, so we conclude that in this experiment increasing segment length somewhat improves performance.
- With noiselessly generated preferences and the regret preference model (Figure 18), $p=0.219$ and the mean τ across the 7 training dataset sizes is 0.25, which is not a significant between segment length and ordering by performance. Likewise, our visual inspection of Figure 19 does not reveal any effect, so we conclude that in this experiment increasing segment length does not improve performance.
- With stochastically generated preferences and the regret preference model (Figure 19), $p=0.016$ and the mean τ across the 7 training dataset sizes is 0.47, indicating a significant and moderate correlation between segment length and ordering by performance. Our visual inspection of Figure 19 does not reveal this effect, indicating that the effect size is small, so we conclude that in this experiment increasing segment length improves performance with only minor effect.

F.2.4 Reward learning in stochastic MDPs

Table 6: Stochastic MDPs: Proportion of 10 MDPs in which performance was near optimal, with varied reward functions. Entering a terminal risk cell results in r_{win} and r_{lose}, each with 50% probability.

Preference generator	$r_{win}=1$ $r_{lose}=-50$	$r_{win}=1000$ $r_{lose}=-50$	$r_{win}=100$ $r_{lose}=-1$	$r_{win}=100$ $r_{lose}=-1000$
Noiseless P_{regret}	1.0	1.0	1.0	1.0
Stochastic P_{regret}	1.0	1.0	1.0	1.0
Noiseless $P_{\Sigma r}$	1.0	0.0	1.0	0.0
Stochastic $P_{\Sigma r}$	1.0	0.0	1.0	1.0

Although we theoretically consider MDPs with stochastic transitions in Section 3, we have not yet *empirically* compared $P_{\Sigma r}$ and P_{regret} in tasks with stochastic transitions, which we do below.

We randomly generated 20 MDPs, each with a 5×5 grid. Instead of terminal cells that are associated with success or failure, these MDPs have terminal cells that are either risky or safe. A single terminal *safe* cell was randomly placed, and the number of terminal *risk* cells was sampled from the set {1,2,7} and then these terminal risk cells were likewise randomly placed. No other special cells were used in this set of MDPs. To add stochastic transitions, the delivery domain was modified such that when an agent moves into a terminal risk cell there is a 50% chance of receiving a lower reward, r_{lose}, and a 50% chance of receiving a higher reward, r_{win}. All other transitions are deterministic. As in the unmodified delivery domain, moving to any non-terminal state results in a reward of -1. Moving to the terminal safe state yields a reward of +50, like the terminal success state of the unmodified delivery domain. Therefore, depending on the values of r_{win} and r_{lose}, it may be better to move into a terminal risk state than to avoid it. All segments were generated by choosing a start state and three actions, all uniformly randomly. For each MDP, the preference dataset $D_{\succ}$ contains 3000 segment pairs.

The 10 MDPs of each condition differed from those of the other conditions by their ground-truth reward function r, with different r_{win} and r_{lose} values. As in Section 6.2, regardless of whether the stochastic or noiseless version of preference model generates the preference labels, the stochastic version of the same preference model is used for learning the reward function.

The results are shown below in Table 6, indicating that for both noiseless and stochastic preference datasets, P_{regret} is always able to achieve near-optimal performance, whereas $P_{\Sigma r}$ is not. These results expand upon and support the first proof of Theorem 3.2 in Section 3.

F.2.5 Generating preferences and learning reward functions with different preference models

Using synthetically generated preferences, here we investigate the effects of choosing the incorrect model. Specifically, either P_{regret} or $P_{\Sigma r}$ generates preference labels, and then the *other* preference model is used to learn a reward function from these preference labels. Through this mixing of preference models, we add two new

conditions to the analysis in the delivery domain in Section 6.2. The results are shown in Figure 20. We observe that, as expected, each preference model performs best when learning on preference labels it generated. Of all four combinations of preference models during generation and reward function inference, the best performing combination is doing reward inference with P_{regret} on preferences generated by P_{regret}.

Between the two mixed-model conditions, we observe that learning with the partial return preference model on preferences created via regret outperforms the reverse. These results suggest that learning reward functions with the regret preference model may be more sensitive to how much the preference generator deviates from it. However, we note that humans *do not* appear to be giving preferences according to partial return, so these results—though informative—are not worrisome.

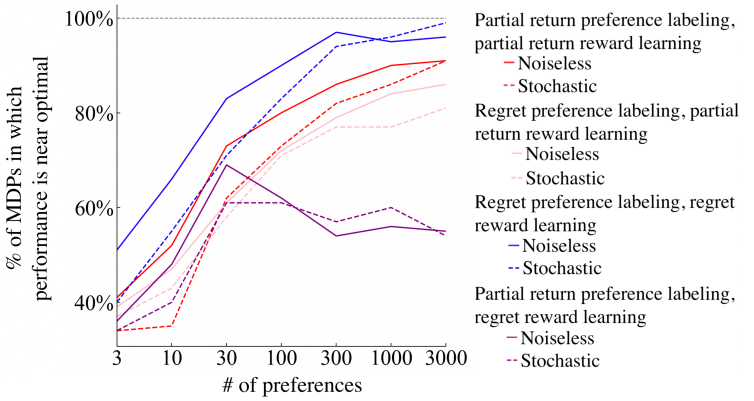

Figure 20: Performance comparison over 100 randomly generated deterministic MDPs, with synthetically generated preferences. This plot expands upon Figure 10 by including mismatches between the preference model generating preference data and the preference model used during learning of the reward function.

F.3 Results from human preferences

In this section we expand upon the results described in Section 6.3, which involve learning reward functions from the dataset of human preferences we collected.

F.3.1 Wilcoxon paired signed-rank test

For the Wilcoxon paired signed-rank test described in Section 6.3, normalized mean returns were clipped to $[-1,1]$ as in Appendix F.2.1. The result from each test is shown in Table 7. These results are surprisingly significant, given that both models reach 100% near-optimal performance for 5 and 10 partitions in Figure 11. However, in this setting, learning a reward function with the regret preference model tends to result in a higher mean score than learning one with the partial return preference model, even when both are above the threshold for near optimal performance.

Table 7: Results from Wilcoxon paired signed-rank tests.

	5 partitions	10 partitions	20 partitions	50 partitions	100 partitions
P_{regret} vs. $P_{\Sigma r}$ preference models	w=0 p=0.043	w=6 p=0.028	w=24 p=0.007	w=216 p=0.003	w=939 p=0.076

F.3.2 Expanded plots, with the expected return and logistic linear preference models included

Figure 21 show the same results as Figure 11, but with additional results from the expected return preference model introduced in Appendix B.5 and the logistic linear preference model ($P_{log-lin}$) introduced in Appendix B.3. Figure 22 shows the same results with a different performance threshold, that of performing better than uniformly random action selection, which receives a 0 on our normalized mean return metric. The regret preference model matches or outperforms *both* other preference models in all partitionings of the human data, at both thresholds (near optimal and better than random).

F.3.3 Performance on only human preferences from the first stage of data collection

As previously mentioned in Section D and Appendix D, when learning reward functions only from the data from the first stage of human data collection, the partial return model does worse. This first stage of data contains 1437 preferences, which is 79% of the full dataset. The specific performance of the partial return preference model on the full set of first-stage data (i.e., 1 partition) is a normalized mean return of -12.7, worse than a uniformly random policy. The regret preference model achieves 0.999, close to optimal performance of 1.0.

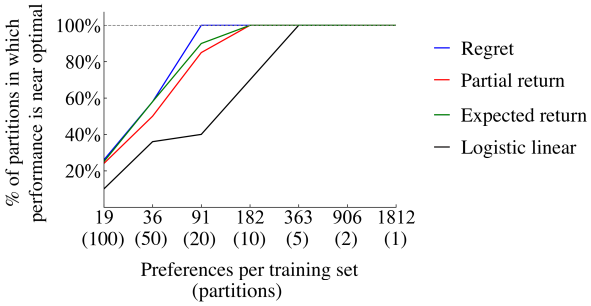

Figure 21: Performance comparison over various amounts of human preferences. Each partition has the number of preferences shown or one less. This plot is identical to Figure 11 except that results for the expected return preference model are included and a different random seed was used to partition the human preferences.

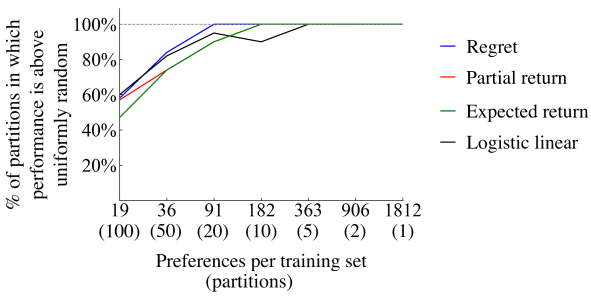

Figure 22: Performance comparison over various amounts of human preferences. Each partition has the number of preferences shown or one less. This plot tests the same learned reward functions as in Figure 21, but it thresholds on outperforming a uniformly random policy rather than on near-optimal performance.

F.3.4 Performance without early-terminating segments

Segment pairs with early-terminating segments are one type of segment pairs (of two types) that are in the second stage of data collection but not in this first stage. The identifiability issue of the partial return model related to a constant shift in the reward function—discussed in Section 3.2.2 and Appendix F.2.2—provides sufficient justification for the low performance in Appendix F.3.3 observed without inclusion of early-terminating segments.

To more directly test the effect of early-terminating segments, we repeat the analysis in Appendix F.3.3 while only removing the segment pairs from the second stage that have early-terminating segments. We get the same normalized mean return, -12.7 for the partial return preference model and 0.999 for the regret preference model. Therefore, removal of the early-terminating segments is sufficient to cause the partial return preference model to perform poorly.

F.3.5 Generalization of learned reward functions to other MDPs with the same ground-truth reward function

We also test how well these learned reward functions generalize to other instantiations of the domain. To do so, we keep the same hidden ground-truth reward function but otherwise randomly sample MDP configurations identically as done for the analysis of Sec. 6.2, which is detailed in Appendix F.2.1. Procedurally, this analysis starts identically as that in Section 6.3, learning a reward function from randomly sampled partitions of the human preference data. Each learned reward function is then tested within the 100 MDPs. For example, for two partitions of the data, each algorithm learns two reward functions and each such $\hat{r}$ is tested on 100 MDPs. To test a reward function in an MDP, the same procedure of value iteration is used as in Sections 6.2 and 6.3 to find an approximately optimal policy, the performance of which is then tested on the MDP (with the ground-truth reward function).

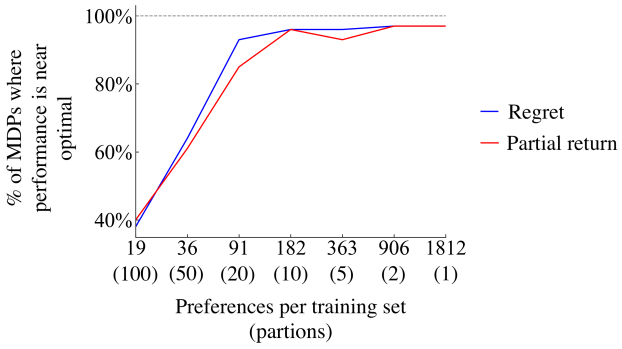

Figure 23: Performance comparison of learned reward functions' generalization to 100 MDPs with the same reward function, when learned from various amounts of human preferences. Each partition has the number of preferences shown or one less.

The results are shown in Figure 23. The two preference models perform similarly at 100, 10, 2, and 1 partition(s), and otherwise the regret preference model outperforms the partial return preference model. The most pronounced differences are at 20 and 5 partitions, where the

partial return preference model fails to reach near-optimal performance approximately twice as often as the regret preference model.

For each training set size, we conduct the same Wilcoxon paired signed-rank test as in Section 6.3 and Appendix F.3.1, except that for each of the 100 MDPs, we calculate the normalized mean return, and the *mean* of normalized mean returns across all 100 MDPs represents a single sample for the statistical test. Across all 7 training set sizes, no statistical significance is found ($p > 0.2$). Unlike most other analyses in this paper, we cannot here conclude superior performance from assuming the regret preference model.