After Appendix A, the appendix is organized according to the major sections and subsections of the main content. We consider Appendix C to be its most important addition to 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 the desirability of decisions 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 $\tilde{Q}^*_{\hat{r}}$ and $\tilde{V}^*_{\hat{r}}$ 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) and creating a reward feature vector ϕ for non-linear reward functions (discussed in Appendix F.1). Additionally, although learning with P_{regret} is more sample efficient in our experiments, it is computationally slower than learning with P_{Σ_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 [14] [9] [25], 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 and 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 and Russell [19]. This challenge has not stopped IRL from being an impactful problem, 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. [26] 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, under "Instantiating Algorithm 1 for reward functions that may be non-linear").

B Preference models for learning reward functions

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.

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

B.1 Logistic-linear preference model

In Appendix E.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 Eqn. 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

$$P_{log-lin}(\sigma_1 \succ \sigma_2 | \tilde{r}) = logistic([\vec{w} \cdot \langle V^*_{\tilde{r}}(s_{\sigma_1,0}) - V^*_{\tilde{r}}(s_{\sigma_2,0}), \Sigma_{\sigma_1} - \Sigma_{\sigma_2}, V^*_{\tilde{r}}(s_{\sigma_1,|\sigma_1|}) - V^*_{\tilde{r}}(s_{\sigma_2,|\sigma_2|}) \rangle]). \quad (8)$$

This logistic-linear preference model is a generalization of P_{Σ_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 $P_{log-lin}(\cdot|\tilde{r}) = P_{regret_d}(\cdot|\tilde{r})$.

B.2 Adding a constant probability of uniformly distributed preference

Appendix E.2 also considers adaptations of P_{Σ_r}, P_{regret_d}, and $P_{log-lin}$ that add a constant probability of uniformly distributed preference, as was done by Christiano et al. [9]. 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_{Σ_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.3 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.4 Policy learning from regret-based preferences without learning a reward function

When preferences are based upon regret, an entirely different method of policy learning appears reasonable. This method is motivated by regret itself differentiating which of two segments is more desirable, including differentiating an optimal segment from a suboptimal segment.

When learning a reward function is not desired, one can instead learn a segment scoring function that estimates the regret of a segment directly. Such a regret estimator can be learned by following the same algorithm for reward learning with the *partial return* preference model that is defined in Section 2.2 and used throughout this paper, with two changes. First, each preference sample's segment ordering needs to be reversed before training. Second, the learned "reward function" is instead interpreted as a regret function, outputting a regret estimate for any segment. (Without reversing the order, the learned function would be *negated* regret.) If we assume the regret estimator perfectly models regret, any action that minimizes expected regret is an optimal action. Therefore, the policy is defined by choosing such a regret-minimizing action.

Though theoretical analysis of this approach is beyond the scope of this paper, we suspect that it would be possible to show that the above algorithm will result in an optimal policy for noiseless or stochastic preferences, given the infinite and exhaustive dataset described in Definition 3.1. In Appendix F.2.3, we provide limited empirical evidence supporting this approach.

Even when learning a reward function, this approach could be used to help create reward features for reward functions that are non-linear or have unknown reward features (as discussed in Appendix F.1) and to identify a policy for which to learn successor features.

B.5 Relationship to inverse reinforcement learning

The inputs to inverse reinforcement learning (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 connection can be made. If one assumes that a demonstrated trajectory segment is noiselessly optimal—as in the foundational IRL paper on apprenticeship learning [27])—then such a demonstration is equivalent to expressing preference or indifference for the demonstrated segment over all other segments (or, equivalently, that 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. [28]) 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, as appears necessary for a perfect measure of regret while learning a reward function [29, Algorithm 1].

C Theoretical comparisons

For convenience, Theorems 3.1 and 3.2 from Section 3 are reprinted below. Consider reviewing the definitions of optimal segments and suboptimal segments in Section 2.1 and Definition 3.1 before proceeding.

Theorem 3.1 (P_{regret} is identifiable)**.** *Let P_{regret} be any function such that if $regret(\sigma_1|\tilde{r}) < regret(\sigma_2|\tilde{r})$, $P_{regret}(\sigma_1 \succ \sigma_2|\tilde{r}) > 0.5$, and if $regret(\sigma_1|\tilde{r}) = regret(\sigma_2|\tilde{r})$, $P_{regret}(\sigma_1 \succ \sigma_2|\tilde{r}) = 0.5$. P_{regret} is identifiable.*

Proof Make all assumptions in Definition 3.1. Since $\hat{r}$ minimizes cross-entropy loss, $P_{regret}(\cdot|r) = P_{regret}(\cdot|\hat{r})$ for all possible segment pairs. Also, by Equation 4 $regret(\sigma|\tilde{r}) = 0$ if and only if σ is optimal with respect to $\tilde{r}$. And $regret(\sigma|\tilde{r}) > 0$ if and only if σ is suboptimal with $\tilde{r}$.

With respect to some $\tilde{r}$, let σ^* be any optimal segment and $\sigma^{\neg *}$ be any suboptimal segment. $regret(\sigma^*|\tilde{r}) < regret(\sigma^{\neg *}|\tilde{r})$. $P_{regret}(\sigma^* \succ \sigma^{\neg *}|\tilde{r}) > 0.5$, which we refer to as being *preferred by* $P_{regret}(\cdot|\tilde{r})$. $P_{regret}(\cdot|\tilde{r})$ induces a total ordering over segments, defined by $regret(\sigma_1|\tilde{r}) < regret(\sigma_2|\tilde{r}) \Longleftrightarrow P_{regret}(\sigma_1 \succ \sigma_2|\tilde{r}) > 0.5 \Longleftrightarrow \sigma_1 > \sigma_2$ and $regret(\sigma_1|\tilde{r}) = regret(\sigma_2|\tilde{r}) \Longleftrightarrow P_{regret}(\sigma_1 \succ \sigma_2|\tilde{r}) = 0.5 \Longleftrightarrow \sigma_1 = \sigma_2$. Because regret has a minimum (0), there must be a set of segments which are ranked highest under this ordering, denoted $\Sigma^*_{\tilde{r}}$. These segments in $\Sigma^*_{\tilde{r}}$ are exactly those that achieve the minimum regret (0) and so are optimal with respect to $\tilde{r}$.

Since the dataset ($D_{\succ}$) contains all segments by assumption, $\Sigma^*_{\tilde{r}}$ contains all optimal segments with respect to $\tilde{r}$. If a state-action pair (s,a) is in an optimal segment, then by the definition of an optimal segment $Q^*_{\tilde{r}}(s,a) = V^*_{\tilde{r}}(s)$. The set of optimal policies $\Pi^*_{\tilde{r}}$ for $\tilde{r}$ is all π such that, for all (s,a), if $\pi(s,a) > 0$, then $Q^*_{\tilde{r}}(s,a) = V^*_{\tilde{r}}(s)$. In short, $\Sigma^*_{\tilde{r}}$ determines the set of every state-action pair (s,a) such that $Q^*_{\tilde{r}}(s,a) = V^*_{\tilde{r}}(s)$, and that set determines $\Pi^*_{\tilde{r}}$. Therefore $\Sigma^*_{\tilde{r}}$ determines $\Pi^*_{\tilde{r}}$, and we will refer to this determination as the function g.

We now focus on the reward function used to generate preferences, r, and on the learned reward function, $\hat{r}$. Since $P_{regret}(\cdot|r) = P_{regret}(\cdot|\hat{r})$, r and $\hat{r}$ induce the same total ordering over segments, and so $\Sigma^*_r = \Sigma^*_{\hat{r}}$. Therefore $g(\Sigma^*_r) = g(\Sigma^*_{\hat{r}})$. Since $g(\Sigma^*_r) = \Pi^*_r$ and $g(\Sigma^*_{\hat{r}}) = \Pi^*_{\hat{r}}$, $\Pi^*_r = \Pi^*_{\hat{r}}$. □

Theorem 3.2 (Noiseless P_{Σ_r} is not identifiable)**.** *Let P_{Σ_r} be any function such that if $\Sigma_{\sigma_1}\tilde{r} > \Sigma_{\sigma_2}\tilde{r}$, $P_{\Sigma_r}(\sigma_1 \succ \sigma_2|\tilde{r}) = 1$, and if $\Sigma_{\sigma_1}\tilde{r} = \Sigma_{\sigma_2}\tilde{r}$, $P_{\Sigma_r}(\sigma_1 \succ \sigma_2|\tilde{r}) = 0.5$. There exists an MDP in which P_{Σ_r} is not identifiable.*

Below we present two proofs of Theorem 3.2. Each are proofs by counterexample. Though only one proof is needed, we present two because each counterexample demonstrates a qualitatively different category of how the partial return preference model can fail to identify the set of optimal policies.

Proof based on stochastic transitions: Assume the following class of MDPs, illustrated in Figure 7. The agent always begins at start state s_0. From s_0, action a_{safe} always transitions to s_{safe}, getting a reward of 0. From s_0, action a_{risk} transitions to s_{win} with probability 0.5, getting a reward of r_{win}, and transitions to s_{lose} with with probability 0.5, getting a reward of -10. In all MDPs in this class, $r_{win} > 0$. All 3 possible resulting states (s_{safe}, s_{win}, and s_{lose}) are absorbing states, from which all further reward is 0.

If $r_{win} \geq 10$, a_{risk} is optimal in s_0. If $r_{win} \leq 10$, a_{safe} is optimal in s_0. Three single-transition segments exist: $(s_0, a_{safe}, s_{safe})$, (s_0, a_{risk}, s_{win}), and $(s_0, a_{risk}, s_{lose})$. By noiseless P_{Σ_r}, $(s_0, a_{risk}, s_{win}) \succ (s_0, a_{safe}, s_{safe}) \succ (s_0, a_{risk}, s_{lose})$, regardless of the value of r_{win}. In other words, P_{Σ_r} is insensitive the what the optimal action is from s_0 in this class of MDPs.

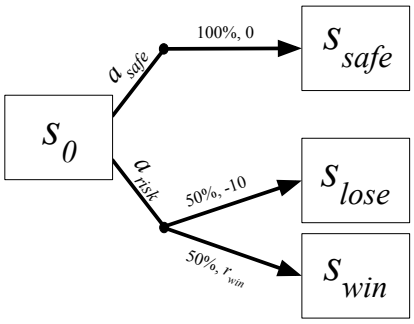

Figure 7: A class of MDPs in which, if $r_{win} > 0$, the partial return preference model fails the test for identifiability.

Now assume MDP M, where r_{win} = 11. In linear form, the weight vector for the reward function r_M can be expressed as $w_{r_{M_1}} = < -10, 0, 11 >$. Let $\hat{r}_M$ have $w_{\hat{r}_M} = < -10, 0, 9 >$. Both r_M and $\hat{r}_M$ have the same preferences as above, meaning that $\hat{r}_M$ minimizes loss on an infinite preferences dataset $D_\succ$ created by P_{Σ_r}, yet it has a different optimal policy. Therefore, noiseless P_{Σ_r} is not identifiable. □

In contrast, note that by noiseless P_{regret}, the preferences are different than those above for P_{Σ_r}. If $r_{win} > 10$, then $(s_0, a_{risk}, s_{win}) \sim (s_0, a_{risk}, s_{lose}) \succ (s_0, a_{safe}, s_{safe})$, If $r_{win} < 10$, then $(s_0, a_{safe}, s_{safe}) \succ (s_0, a_{risk}, s_{win}) \sim (s_0, a_{risk}, s_{lose})$. Intuitively, this difference comes from P_{regret} always giving higher preference probability to optimal actions, even if they result in bad outcomes. Another perspective can be found from the utility theory of Von Neumann and Morgenstern [30]. Specifically, P_{Σ_r} gives preferences over outcomes, which in the terms of utility theory can only learn an ordinal utility function. Ordinal utility functions are merely consistent with the preference ordering over outcomes and do not generally capture preferences over actions when their outcomes are stochastically determined. The deterministic regret preference model, P_{regret_d}, also has this weakness in tasks with stochastic transitions. On the other hand, P_{regret} forms preferences over so-called lotteries—the distribution over possible outcomes—and can therefore learn a cardinal utility function, which can explain preferences over risky action.

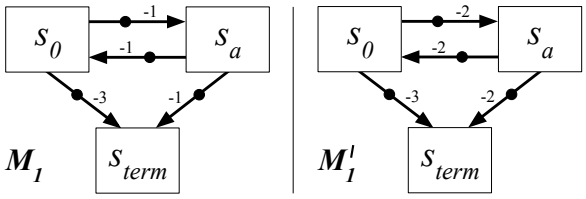

Figure 8: An MDP (M_1) where $\Pi^*_r = \Pi^*_{\hat{r}}$ is not guaranteed for the partial return preference model, failing the test for identifiability with segments of length 1. The ground truth reward function is shown to the left, and an MDP M_1' with an alternative reward function is shown to the right. Under partial return, both create the same set of preferences despite having different optimal actions from s_0.

Since the proof above is focused on stochastic settings, we show the lack of identifiability for noiseless P_{Σ_r} can be found for quite different reasons in a deterministic MDP.

Proof based on segments of fixed length: Consider the MDP M_1 in Figure 8 and assume preferences are given over segments with length 1 (i.e., containing one transition). The optimal policy for M_1 is to move rightward from s_0, whereas optimal behavior for M_1' is to move leftward from s_0. In *both* M_1 and M_1', preferences by P_{Σ_r} are as follows, omitting the action for brevity: $(s_a,s_0) \sim (s_a,s_{term}) \sim (s_0,s_a) \succ (s_0,s_{term})$. As in the previous proof, P_{Σ_r} is insensitive to certain changes in the reward function that alter the set of optimal policies. Whenever this characteristic is found, $\Pi_r^* = \Pi_{\hat{r}}^*$ is not guaranteed, failing the test for identifiability. Here specifically, the reward function for M_1' would achieve 0 cross-entropy loss on an exhaustive preference dataset created in M_1 with the noiseless preferences from the partial return preference model, despite the optimal policy in M_1' conflicting with the ground truth optimal policy.

The logic of this proof can be applied for trajectories of length 2 in the MDP M_2 shown in Figure 9. Together, M_1 and M_2 suggest a rule for constructing an MDP where $\Pi_r^* = \Pi_{\hat{r}}^*$ is not guaranteed for P_{Σ_r}, failing the identifiability test for any fixed segment length, $|\sigma|$: set the number of states to the right of s_0 to $|\sigma|$ (not counting s_{term}), set the reward r_{fail} for (s_0, s_{term}) such that $r_{fail} < 0$, and set the reward for each other transition to $c + r_{fail}/(|\sigma|+1)$, where $c > 0$. Given an MDP constructed this way, an alternative reward function that results in the same preferences under P_{Σ_r} yet has a different optimal action from s_0 can then be constructed by changing all reward other than r_{fail} to $c + r_{fail}/(|\sigma|+1)$, where c now is constrained to $c < 0$ and $c \times |\sigma| < r_{fail}$. Note that the set of preferences for each of these MDPs is the same even when including segments that reach terminal state before $|\sigma|$ transitions (which can still be considered to be of length $|\sigma|$ if the terminal state is an absorbing state from which reward is 0).

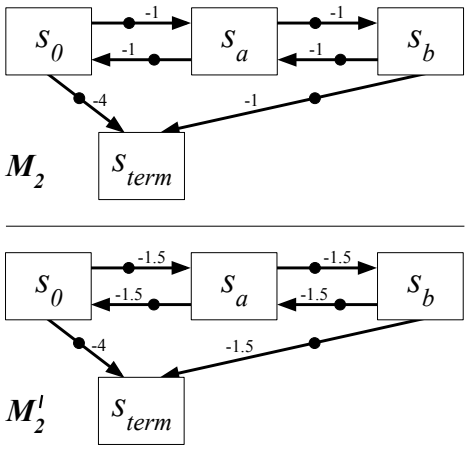

Figure 9: An MDP (M_2) where $\Pi_r^* = \Pi_{\hat{r}}^*$ is not guaranteed for the partial return preference model, failing the test for identifiability with segments of length 2. The ground truth reward function is shown in the top diagram, and an MDP M_2' with an alternative reward function is shown in the bottom diagram. Under partial return, both create the same set of preferences despite having different optimal actions from s_0.

The relevance of noiseless preference generators Because we model preferences as stochastic in Section 2, at this point 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.

Lastly, there is precedent for considering noiseless human input for theory or derivations. For instance, the foundational IRL research by Abbeel and Ng on apprenticeship learning [27] treats demonstrations as noiselessly optimal. Recent work by Kim et al. [28] focuses on reward identifiability with noiseless, optimal demonstrations.

D Additional information for creating a human-labeled preference dataset

D.1 The user interface and study overview

Here we share miscellaneous details about the user 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 3) 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.

D.2 Filtering subject data

Before someone could join our study via Amazon Mechancial Turk, they 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. 58 of these subjects were removed based on their responses to the survey, and another 35 were removed for making preference errors. After filtering by both the comprehension survey and subject error, we used the data from 50 subjects. This included 1812 preferences over 1245 unique segment pairs.

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

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 10 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 4, 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 4, and segment pairs that are not on any of the dots in Figure 4 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 4, 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 4 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 20 labeled segment pairs per bin. After filtering subject data, this first stage contributed 1501 segment pairs out of the 1812 pairs used in our reward learning experiments in Section 6.3 and Appendix F.3.

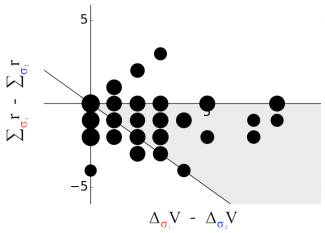

Figure 10: Coordinates from which segment pairs were sampled from during the first stage of data collection. The x-axis is state value differences between the two segments and the y-axis is partial return differences between the two segments. The areas of the circles are proportional to the number of samples at that point, and the proportionality is consistent across this plot and the 3 subplots of Figure 11.

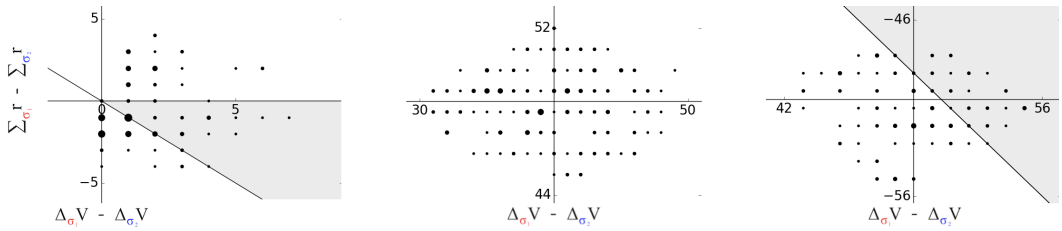

Figure 11: Coordinates from which segment pairs were sampled from during the second stage of data collection. The points are in 3 distant clusters, so they are presented in 3 separate subplots for readability. The areas of the circles are proportional to the number of samples at that point, and the proportionality is consistent across these 3 subplots and Figure 10.

Second stage When we conducted the reward-learning evaluation in Section 6 with only the data from the first stage, P_{Σ_r} performed very poorly, always performing worse than uniformly random. This performance difference is shown in Appendix **??**. In contrast P_{regret} performed well, always achieving near-optimal performance. To better assess P_{Σ_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 $s_{\sigma,1}$. 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.

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 11. 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 11.

All of both types of additional segments pairs are then characterized by the coordinates shown in Figure 11. 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. 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. [31].

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.

$$
\begin{aligned}
&regret_{\mathrm{d}}(\sigma_2|\tilde{r}) - regret_{\mathrm{d}}(\sigma_1|\tilde{r}) \\
&\quad = \Big([V^*_{\tilde{r}}(s_{\sigma_2,0}) - (\Sigma_{\sigma_2}\tilde{r} + V^*_{\tilde{r}}(s_{\sigma_2,|\sigma_2|})] - [V^*_{\tilde{r}}(s_{\sigma_1,0}) - (\Sigma_{\sigma_1}\tilde{r} + V^*_{\tilde{r}}(s_{\sigma_1,|\sigma_1|})] \Big) \\
&\quad = \Big([V^*_{\tilde{r}}(s_{\sigma_2,0}) - V^*_{\tilde{r}}(s_{\sigma_2,|\sigma_2|})] - [V^*_{\tilde{r}}(s_{\sigma_1,0}) - V^*_{\tilde{r}}(s_{\sigma_1,|\sigma_1|})] \Big) - \Big(\Sigma_{\sigma_2}\tilde{r} - \Sigma_{\sigma_1}\tilde{r} \Big) \\
&\quad = \Big([V^*_{\tilde{r}}(s_{\sigma_1,|\sigma_1|}) - V^*_{\tilde{r}}(s_{\sigma_1,0})] - [V^*_{\tilde{r}}(s_{\sigma_2,|\sigma_2|}) - V^*_{\tilde{r}}(s_{\sigma_2,0})] \Big) + \Big(\Sigma_{\sigma_1}\tilde{r} - \Sigma_{\sigma_2}\tilde{r} \Big) \\
&\quad = (\Delta_{\sigma_1} V_{\tilde{r}} - \Delta_{\sigma_2} V_{\tilde{r}}) + (\Sigma_{\sigma_1}\tilde{r} - \Sigma_{\sigma_2}\tilde{r})
\end{aligned}
\tag{9}
$$

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 provides a lower bound in most cases, given that it can express anything the other preference models can and the rarity of overfitting its 3 parameters. Therefore, the intended comparisons are either between P_{regret} and P_{Σ_r} without the constant probability of a uniformly random response or between them with it. We embolden the result with lower loss between these two preference models for each such comparison.

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.

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)	0.69
P_{Σ_r} (partial return)	0.62
P_{regret} (regret)	**0.57**
$P_{log\text{-}lin}$ (logistic linear)	0.55
P_{Σ_r} with prob of uniform response	0.63
P_{regret} with prob of uniform response	**0.59**
$P_{log\text{-}lin}$ with prob of uniform response	0.57

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

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

Because the ordering of preference pairs is arbitrary, 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.

For this specific instantiation, we compute successor feature functions by first randomly creating a large number of reward functions. Specifically, each reward function is created by sampling with replacement each element of its weight vector, $\boldsymbol{w}_{\tilde{\boldsymbol{r}}}$, from $\{-50,-10,-2,-1,0,1,5,10,50\}$. We also included the ground-truth reward function, r, at this point, resulting in 70 reward functions. For each reward function, we create its maximum entropy optimal policy through value iteration. In practice, we learned the successor feature functions as part of the value iteration process. Finally, we remove any successor feature functions for redundant policies and then also remove the successor features function for the optimal policy for r. Note that the only effect of including r in the earlier step was to allow us to 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. (However, the policy from the regret estimator described in Appendix B.4 could be used to learn a successor features function, gaining the benefit of having a successor features function for at least one policy that is likely to perform decently in the task.)

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 $\hat{r}$. A better understanding of these cyclical fluctuations could further improve learning with P_{regret}.

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.

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 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.

The computer used to run all 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; and operating system drive: 2 TB NVMe (3,500 MB/s read).

Pytorch 1.7.1 [32] was used to implement all reward learning models, and statistical analyses were performed using Scikit-learn 0.23.2 [33].

Instantiating Algorithm 1 for reward functions that may be non-linear Algorithm 1 assumes that the reward function can be expressed as a linear combination of reward features that are provided by a reward-features function ϕ that is input to the algorithm. Here we address situations when that assumption does not hold. If the reward features are unknown or the reward is known to be non-linear, one 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. [26]. 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.
- An additional auxiliary objective that may be promising is to use a neural network to learn a regret estimator as described in Appendix B.4 and then use the activations of its penultimate layer as reward features.
- 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 12 complements the results in Figure 5, 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 5, 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 5 shows that the regret preference model achieves near optimal performance at a higher proportion.

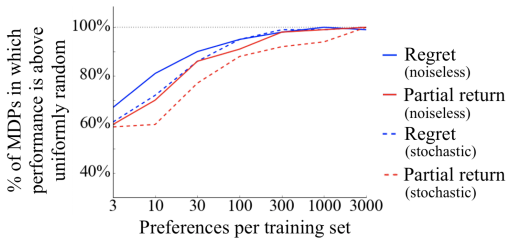

Figure 12: Comparison of performance over 100 randomly generated deterministic MDPs, showing the percentage of MDPs in which each model performed better than an agent taking actions by a uniformly random policy. This plot complements Figure 5, which shows the percentage of MDPs in which the models perform near-optimally.

Details for statistical tests We performed a Wilcoxon 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.

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

Preference generator type	$\|D_{\succ}\|=3$	$\|D_{\succ}\|=10$	$\|D_{\succ}\|=30$	$\|D_{\succ}\|=100$	$\|D_{\succ}\|=300$	$\|D_{\succ}\|=1000$	$\|D_{\succ}\|=3000$
Noiseless (P_{regret} vs. P_{Σ_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_{Σ_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_{Σ_r} learn near-optimal policies on the same MDPs within this set of 100 randomly generated MDPs. Results for this analysis are shown below.

F.2.2 Varying segment size

Here we consider the effect of increasing the fixed length of segments in the preference data set. Specifically, we learn a reward function using preference datasets that contained segments of lengths $n \in 6,9,12,15,18,21$ in the specific delivery task, where each dataset contained segments of the same length. Each preference dataset contained 3000 segment pairs. Each segment was generated by

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_{Σ_r}	10	12	7	8	3	3	3
Neither	39	22	10	2	0	2	1

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 pairs, resulting in datasets that differ only in their labels, and then the preference model is used for reward learning on the same dataset it labeled. Unlike in Appendix F.2.1, all segments with termination reach terminal state on their final transition, which we had already observed was problematic for P_{Σ_r} (and the motivation for the second stage of data collection). Therefore, these results are intended to provide comparison between values of n more so than between preference models.

Regardless of the value of n, P_{regret} always achieves the optimal mean return of 41.2. P_{Σ_r} always achieves a mean return of -500.5. This analysis provides limited evidence that segment size does not have a large effect, though further analysis is needed to make this assertion with confidence.

F.2.3 Policy learning without reward learning, using a regret estimator

Here we test the performance of policy learning from regret-based preferences *without learning a reward function*, using a regret estimator, as described in Appendix B.4.

A set of 30 random MDPs is generated as described in Appendix F.2.1, except that the possible MDP widths are instead sampled from the set $\{1,5,6,10\}$ and the heights are sampled from $\{3,5,6,10\}$. For each MDP, the preference dataset $D_{\succ}$ contains 3000 segment pairs, randomly sampled as in Appendix F.2.1. If 3000 unique segment pairs do not exist, then each possible segment pair is used once. Preference labels are given by P_{regret}.

Anticipating that a linear function over the six-element reward feature vector ϕ would be insufficient to estimate regret, we expanded ϕ to include both the six reward components and an identity function for each possible state-action pair, creating an addition to the feature vector of size $|S|\times|A|$ that contains all zeroes except a single 1 indicating the state and action that initiates a transition.

Table 6 shows results for noiseless and stochastic preferences.

Table 6: Success of learning a regret estimator with noiseless and stochastic preferences.

Preference Model	% of MDPs in which performance was better than uniformly random	% of MDPs in which performance was near optimal
Noiseless P_{regret} preference labels, learning a regret estimator	90%	86.67%
Stochastic P_{regret} preference labels, learning a regret estimator	80%	73.33%

F.2.4 Artificially lowering the discount factor

Almost all deep reinforcement learning algorithms artificially add discounting to tasks that are episodic [9]. Considering that much of the past work that used partial return preference models also involved deep RL, here we re-interpret results above to probe how such discounting affects performance of this preference model if preferences are actually given by P_{regret}. Specifically, the analysis above in Appendix F.2.3 on policy learning without reward learning is applicable to this topic. If the regret estimator is instead considered a negated reward function, then taking a minimal-regret action is equivalent to taking a maximum-value action under fully myopic discounting, $\gamma=0$.

F.2.5 Reward learning in stochastic MDPs

Although we theoretically consider MDPs with stochastic transitions in Appendix C, we have not yet *empirically* compared P_{Σ_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. The results are shown below, indicating that for both noiseless and stochastic preference datasets, P_{regret} is always able to achieve near-optimal performance, whereas P_{Σ_r} is not.

Table 7: Stochastic MDPs: $\%$ of MDPs in which performance was near optimal, with varied reward functions.

Preference Model	$r_{win}=1$ $r_{lose}=-50$	$r_{win}=10^3$ $r_{lose}=-50$	$r_{win}=100$ $r_{lose}=-1$	$r_{win}=100$ $r_{lose}=-10^3$
Noiseless P_{regret}	100%	100%	100%	100%
Stochastic P_{regret}	100%	100%	100%	100%
Noiseless P_{Σ_r}	100%	0%	100%	0%
Stochastic P_{Σ_r}	100%	0%	100%	100%

The results above expand upon and support the first proof of Theorem 3.2 in Appendix C.

F.3 Results from human preferences

In this section we provide further detail regarding the analysis in Section 6.3. For the Wilcoxon paired signed-rank test, normalized mean returns were clipped to $[-1,1]$ as in Appendix F.2.1. The result from each test is shown in Table F.3.

Table 8: Results from Wilcoxon signed-rank tests.

	5 partitions	10 partitions	20 partitions	50 partitions	100 partitions
P_{regret} vs. P_{Σ_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

Figure Figure 13 show the same results as Figure 6, but with additional results from the expected return preference model introduced in Appendix B.3. Figure 14 shows the same results with a different 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).

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. 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 -3.8, whereas the regret preference model achieves 1.0, close to optimal performance.

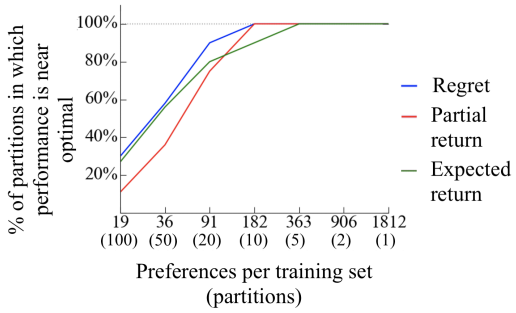

Figure 13: Performance comparison over various amounts of human preferences. Each partition has the number of preferences shown or one less. This plot is identical to Figure 6 except that results for the expected return preference model are included.

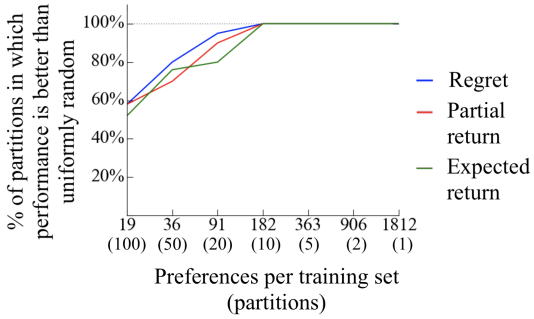

Figure 14: Performance comparison over various amounts of human preferences. Each partition has the number of preferences shown or one less. This plot focuses on outperforming a uniformly random policy, whereas Figure 6 thresholds on near-optimal performance.