Active preference-based Gaussian process regression for reward learning and optimization

Erdem Biyik1,2,3, Nicolas Huynh4,5, Mykel J. Kochenderfer6 and Dorsa Sadigh1,7

Abstract
Designing reward functions is a difficult task in AI and robotics. The complex task of directly specifying all the desirable behaviors a robot needs to optimize often proves challenging for humans. A popular solution is to learn reward functions using expert demonstrations. This approach, however, is fraught with many challenges. Some methods require heavily structured models, for example, reward functions that are linear in some predefined set of features, while others adopt less structured reward functions that may necessitate tremendous amounts of data. Moreover, it is difficult for humans to provide demonstrations on robots with high degrees of freedom, or even quantifying reward values for given trajectories. To address these challenges, we present a preference-based learning approach, where human feedback is in the form of comparisons between trajectories. We do not assume highly constrained structures on the reward function. Instead, we employ a Gaussian process to model the reward function and propose a mathematical formulation to actively fit the model using only human preferences. Our approach enables us to tackle both inflexibility and data-inefficiency problems within a preference-based learning framework. We further analyze our algorithm in comparison to several baselines on reward optimization, where the goal is to find the optimal robot trajectory in a data-efficient way instead of learning the reward function for every possible trajectory. Our results in three different simulation experiments and a user study show our approach can efficiently learn expressive reward functions for robotic tasks, and outperform the baselines in both reward learning and reward optimization.

Keywords
Reward learning, active learning, inverse reinforcement learning, preference-based learning, human-robot interaction, Gaussian processes

Received 11 January 2023; Revised 26 July 2023; Accepted 23 September 2023

1. Introduction
Planning for robots that can act in a diverse set of environments based on human preferences can be quite challenging. It is usually impractical for human designers to directly program the desired behaviors across an exhaustive range of possible scenarios. Therefore, roboticists often integrate machine learning into their design process to deduce human preferences. One approach is to learn a robot policy directly from expert demonstrations (Ho and Ermon, 2016; Ross et al., 2013; Song et al., 2018; Stadie et al., 2017). However, in many settings, we are interested in learning a reward function that represents how a robot should act or interact within the world (Biyik, 2022).

Reward functions serve as powerful tools for prescribing desirable robot behaviors, for example, how to act safely, or what styles or goals the robot needs to follow. Unfortunately, specifying reward functions is not easy for human designers (Clark and Amodei, 2016; Christiano et al., 2017;
functions and optimizing them directly. Previous works have demonstrated that employing a series of pairwise comparisons between trajectories is an effective way to learn reward functions (Akrour et al., 2012; Biyik, 2022; Brown and Niekum, 2019; Cakmak et al., 2011; Ibarz et al., 2018; Lee et al., 2021; Lepird et al., 2015; Sadigh et al., 2017; Tucker et al., 2020b). For example, as shown in Figure 1, the robot can demonstrate the blue and green trajectories, $\xi_A$ and $\xi_B$, and query the human designer for their comparison between the two. Preference-based reward learning then utilizes a series of pairwise comparisons to accurately estimate a reward function.

However, preference-based learning techniques are typically not data-efficient, because each pairwise comparison provides at most 1 bit of information: whether $\xi_A$ is preferred over $\xi_B$ or vice versa. Therefore, active learning is frequently incorporated into this framework to identify the most informative or diverse sequence of pairwise comparison queries for efficiently converging to the underlying reward function (Akrour et al., 2012; Basu et al., 2019; Biyik et al., 2019b; Biyik, Lazar, et al., 2019; Biyik, Wang, et al., 2019; Biyik and Sadigh, 2018; Katz et al., 2019; Palan et al., 2019; Racca et al., 2019; Sadigh et al., 2017; Wilde et al., 2019, 2020, 2021).

Unfortunately, most of the existing active reward learning methods are reliant on a strong assumption about the structure of the reward function: the reward function is a linear combination of a set of hand-coded features. While this assumption is typically needed for active learning to scale, it is highly restrictive as linear reward functions often lack the required expressiveness to encode human preferences. For example, a linear reward function would necessitate several features for the human teacher to be able to specify every reward configuration of targets in the task demonstrated in Figure 1, that is, how targets compare to each other. The features to this task could be, for example, distances to each and every target. On the other hand, if the reward model was nonlinear, one could capture all possible configurations with only two features: speed for how far the ball will be thrown, and angle, representing the direction to shoot. While neural networks or kernel functions can offer this flexibility (Christiano et al., 2017; Lee et al., 2021; Katz et al., 2021), these techniques significantly increase the number of parameters required, which prohibits (or renders useless the advantage of) active learning algorithms.

Our key insight is to model the reward function using a Gaussian process (GP) (Rasmussen and Williams, 2005). GPs are nonparametric models that can capture nonlinearities, enabling us to actively and efficiently learn highly expressive reward functions.

In this work, we introduce a mathematical framework for actively fitting a GP using only pairwise comparisons between two trajectories, which we call preference data. Leveraging GPs, as opposed to linear models with manually designed features, enhances the expressiveness of reward functions by incorporating nontrivial nonlinearities.

Figure 1. The user is trying to teach the robot how to play a variant of mini-golf, where the reward differs among eight targets. In preference-based learning, instead of trying to design a reward function by hand or controlling the robot to provide demonstrations, the user simply compares two demonstrated trajectories on the robot. Here, $\xi_A$ and $\xi_B$ demonstrate two trajectories that correspond to hitting the ball towards the blue or green targets.

Besides, our active query generation method allows us to scale the advantages of active learning.

Building upon this framework, which we previously introduced in our conference paper (Biyik et al., 2020), we also consider the common use case of reward learning where the ultimate goal is to identify the best trajectory for the robot in a fixed environment (Wilde et al., 2020). In such settings, it is sufficient to find the trajectory that maximizes the reward function instead of learning the function for all possible trajectories. We call this setting reward optimization (as opposed to reward learning). We propose and empirically analyze a variant of our reward learning algorithm for reward optimization.

We make three main contributions in this work:\1

- We introduce a data-efficient mathematical framework for actively regressing a GP with preference data, enabling the learning of expressive reward functions from humans.
- We demonstrate the efficacy of our framework through three different simulated environments and user studies on a manipulator robot playing a variant of mini-golf based on different human preferences. Our results show our approach can be used for reward learning in complex settings and is more data-efficient than other baselines.
- We propose a variant of our reward learning algorithm for reward optimization, and empirically compare it
against several baselines in simulation. Our results show this variant gets closest to identifying the best trajectory with very limited data, while the original algorithm is significantly slower in terms of computation time and may perform better only after many queries.

2. Related work

In this section, we start with reviewing the prior work focused on learning reward functions from demonstrations, or other sources of data, as well as the existing works for preference-based reward optimization. Finally, we discuss related works in Gaussian process regression and how they relate to and differ from our work.

2.1. Learning reward functions from demonstrations

Learning reward functions via collected expert demonstrations has been extensively studied. This is commonly referred to as inverse reinforcement learning (IRL), where human demonstrations are assumed to approximately optimize a reward function which encodes their preferences (Abbeel and Ng, 2004, 2005; Ng and Russell, 2000; Ramachandran and Amir, 2007; Ziebart et al., 2008). The robot can then use the learned reward function to optimize its actions in the broad range of environments.

Despite promising outcomes of IRL in various domains, robots, especially manipulators with high degrees of freedom, are often too difficult to manually operate (Akgun et al., 2012; Dragan and Srinivasa, 2012; Javdani et al., 2015; Khurshid and Kuchenbecker, 2015; Losey et al., 2020). Moreover, recent studies in autonomous driving, where the robot does not have a high degree of freedom, reveal that people generally do not favor an autonomous vehicle to mimic their own demonstrations. Instead, they prefer a different behavior that tends to be more timid (Basu et al., 2017). These findings suggest that one needs to go beyond human demonstrations to effectively capture the preferred reward function.

In contrast to depending on hard-to-collect human demonstrations, our framework learns reward functions through preference queries, eliminating the need for experts capable of operating the system in the desired manner.

2.2. Learning reward functions from other sources of data

In addition to demonstrations, language (Sharma et al., 2022; Sontakke et al., 2023), and physical corrections (Bajcsy et al., 2017, 2018), where the robot attempts to learn the reward function through physical human intervention, learning from rankings (Brown and Nieckum, 2019; Myers et al., 2021) is another popular approach. A particular case of this is when the rankings are only pairwise comparisons, which we call preference queries. Previous works have explored the use of preference queries for reward learning. Lepird et al. (2015); Sadigh et al. (2017) proposed acquisition functions to actively generate the queries. Additional studies broadened this approach to batch-active methods (Biyik and Sadigh, 2018; Biyik et al., 2019c), best-of-many choice queries instead of pairwise comparisons (Biyik et al., 2019a), general Markov Decision Process (MDP) settings (Katz et al., 2019), online interactive settings (Myers et al., 2023), and settings that combine expert demonstrations or other forms of human feedback with preference queries (Biyik et al., 2021; Jeon et al., 2020). The reward function assumed by these prior works is often linear with respect to some hand-coded features, thereby limiting the model flexibility and necessitating meticulous feature design (Biyik et al., 2022).

Basu et al. (2019) experimented with modeling a multi-modal non-stationary reward function, although the reward remained linear in each mode. Further, their work primarily focused only on bi-modal rewards, leaving scalability to more modes a potential problem. Christiano et al. (2017); Lee et al. (2021); Katz et al. (2021) employed reward functions that are modeled using neural networks. However, due to the large parameter spaces of neural networks, they typically need tens of thousands of preference queries to be able to learn the reward functions, which pose practicality issues when working with humans in real-world settings.

In this work, we do not make the linearity assumption and instead employ a GP to model the reward, allowing the modeling of complex reward functions without necessitating large parameter spaces. Our results show this approach significantly improves the expressive power of the learned reward function while remaining data-efficient.

2.3. Preference-based reward optimization

A few works concentrated on the reward optimization setting where the objective is to find the optimal trajectory rather than learning the reward function everywhere. Wilde et al. (2020) developed an algorithm that aims at minimizing some regret measure to find the optimal trajectory with as few preference queries as possible. Indeed, their method identified the optimal trajectory with fewer preference queries than the state-of-the-art active preference-based reward learning method, indicating reward optimization can be solved with greater data-efficiency than reward learning. However, they also assume a linear reward function, and their query generation method relies on the presence of an efficient planner which outputs the optimal trajectory given any reward function. In many applications, this planner demands reinforcement learning training, which is not efficient enough to run multiple times to optimize a single query. Contrarily, our reward learning method and its variant for reward optimization do not require such a planner.

In another line of work, Tucker et al. (2020b, 2020a) employed a Thompson sampling technique to find the optimal trajectory (or the optimal exoskeleton gait in their case) as quickly as possible. While this technique is very fast in terms of computation time, our empirical results demonstrate that our methods are more data-efficient.
More recently, reinforcement learning from human feedback (RLHF), or preference-based reinforcement learning (Casper et al., 2023; Ouyang et al., 2022; Wirth et al., 2017), gained popularity where human feedback is often in the form of pairwise preference comparisons. The goal in RLHF is to learn the optimal policy by making on-policy preference queries to the user, where queries consist of trajectories that approximately optimize the learned reward. Researchers adopted various forms of reward functions in RLHF, including nonlinear forms such as neural networks (Christiano et al., 2017) and GPs (Kupcsik et al., 2018). While this setting is similar to our reward optimization problem, they require either training a policy after every preference query, or making a very large number of queries (as in (Christiano et al., 2017)), or a policy function that is parameterized by only a small number of parameters (as in (Kupcsik et al., 2018)). Instead, we are interested in the off-policy version of the problem without the assumption of small policy parameter space. Furthermore, we also study the reward learning problem.

2.4. Gaussian process regression

On the machine learning front, González et al. (2017) and Chu and Ghahramani (2005) proposed methods for preference-based Bayesian optimization and GP regression, respectively, but they were not active. The approach by preference-based Bayesian optimization and GP regression, Chu and Ghahramani (2005) proposed methods for the reward function that is learned up to that iteration. While the latter focused on robotics tasks, they were not preference-based. Thus, it might not be feasible in many applications as humans often struggle to assign actual reward values.

In this work, we propose an active query generation method for preference-based GP regression for reward learning and optimization. This technique is data-efficient and does not require humans to assign reward values to trajectories for fitting the GP.

3. Problem formulation

We model the environment the robot is going to operate in as a finite-horizon MDP. We use \( s_t \in \mathcal{S} \) to denote the state and \( a_t \in \mathcal{A} \) for the action (control inputs) at time \( t \). A trajectory \( \xi \in \Xi \) within this MDP is a sequence that consists of the state-action pairs: \( \xi = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T) \), where \( T \) is the finite time horizon. We assume a reward function over trajectories, \( R : \Xi \rightarrow \mathbb{R} \), encoding the user’s preferences about how they want the robot to operate.

We assume the reward function \( R \) can be formulated as \( R(\xi) = f(\Psi(\xi)) \), where \( \Psi : \Xi \rightarrow \mathbb{R}^d \) defines a set of trajectory features. These features are often hand-designed, for example, average speed and minimum distance to the closest car in a driving trajectory. However, we emphasize that this formulation of \( R \) enables a more general form of functions that does not require strong assumptions—such as linearity in the features, which corresponds to constraining \( f \) to be an affine function—which is commonly used when learning reward functions. We use a GP to model \( f \), which allows us to avoid strong assumptions about the form of \( f \).

Our goal in reward learning is to learn this more general form of reward function using preference data in the form of pairwise comparisons. In reward optimization, on the other hand, the objective is to identify the trajectory that maximizes the reward function, again using preference data.

The robot will demonstrate a query \( Q \) consisting of two trajectories, \( \xi_A \) and \( \xi_B \) as shown in Figure 1 with blue and green curves, to the user, and will ask which trajectory they prefer. The user will respond to this query based on their preferences. The user’s response provides useful information about the underlying preference reward function \( R \). Of course, we cannot assume human responses are perfect for every query. Consequently, we model the noise in their responses using the commonly adopted probit model (Kupcsik et al., 2018), which assumes humans make a binary decision between the two trajectories noisily based on the cumulative distribution function (cdf) of the difference between the two reward values:

\[
P(q = \xi_A | Q = \{\xi_A, \xi_B\}) = P(R(\xi_A) - R(\xi_B) > \nu),
\]

where \( q \in Q \) denotes the user’s choice, and \( \nu \sim \mathcal{N}(0, \sigma^2) \) for some standard deviation \( \sigma \sqrt{2} \). Therefore, equivalently:
We have: linearities and uncertainties without introducing parameters.

Having defined the problem setting, we are now ready to present our method for learning data-efficient and expressive reward functions using GPs, as well as its variant for reward optimization.

4. Methods

In this section, we first give some background information about Gaussian processes. We then introduce preference-based GP regression, where we show how to update a GP with the results of pairwise comparisons. Next, we present our approach to active preference query generation for reward learning, where we discuss how to find the most informative query that accelerates the learning. Finally, we discuss how this approach can be modified for reward optimization where we do not need to learn the reward function for all possible trajectories. To simplify the notation, we replace \( \Psi(\xi) \) with \( \Psi \), with superscripts and subscripts when needed.

4.1. Gaussian processes

We start by introducing the necessary background on GPs for our work. We refer the readers to Rasmussen and Williams (2005) for other uses of GPs in machine learning.

Suppose we are given a dataset \( \Psi = (\Psi_i)_{i=1}^N \), where \( \Psi_i \in \mathbb{R}^d \). We employ a probabilistic point of view for \( f \) by modeling it using a GP, which enables us to model non-linearities and uncertainties without introducing parameters. We have:

\[
P(f|\mu, K) = \frac{\exp \left( -\frac{1}{2} (f - \mu)\trans K^{-1} (f - \mu) \right)}{(2\pi)^{N/2}|K|^{1/2}},
\]

where \( f = (f(\Psi_i))_{i=1}^N \), \( \mu \) and \( K \) are the mean vector and the covariance matrix of the GP distribution for the \( N \) items in the dataset. Here, \( f \) follows a multivariate distribution. The covariance matrix depends on the kernel. In this work, we use a variant of the radial basis function (RBF) kernel with hyperparameter \( \theta \):

\[
k(\Psi, \Psi_i) = \exp \left( -\theta \|\Psi - \Psi_i\|^2 \right) - \tilde{k}(\Psi, \Psi_i),
\]

\[
\tilde{k}(\Psi, \Psi_i) = \exp \left( -\theta \|\Psi - \tilde{\Psi}\|^2 \right) - \theta \|\Psi_i - \tilde{\Psi}\|^2,
\]

where \( \tilde{\Psi} \in \mathbb{R}^d \) is an arbitrary point for which we assume \( f(\tilde{\Psi}) = 0 \). This is important because the query responses only depend on the relative difference between the two reward function values at the trajectories, that is, \( f(\Psi) + c \) for any \( c \in \mathbb{R} \) would have the same likelihood for a dataset as \( f(\Psi) \). By setting \( f(\Psi) = 0 \) for some arbitrary \( \Psi \in \mathbb{R}^d \), we dissolve this ambiguity. It does not introduce an assumption because, for any function \( f \) and for any point \( \Psi \), one can define \( f(\Psi) = f(\Psi) - f(\Psi) \) without loss of generality—both \( f \) and \( f(\Psi) \) will encode the same preferences. Finally, this variant of the RBF kernel is still positive semi-definite, because it is equivalent to the covariance kernel of a GP which is initialized with an initial data point and a standard RBF kernel prior.

4.2. Preference-based GP regression

In preference-based learning, we have a dataset \( \Psi = ((\Psi_i^{(1)}, \Psi_i^{(2)}))_{i=1}^N \), consisting of pairs of trajectories \( \Psi_i^{(1)}, \Psi_i^{(2)} \in \mathbb{R}^d \), and user responses \( q = (q_i)_{i=1}^N \), where \( q_i \in \{0, 1\} \) indicates whether the user preferred \( \Psi_i^{(1)} \) or \( \Psi_i^{(2)} \). Accordingly, \( K \) is now a \( 2N \times 2N \) matrix, whose rows and columns correspond to \( \Psi_i, \forall i \in \{1, ..., N\}, \forall j \in \{1, 2\} \). Similarly, \( \mu \) is a \( 2N \)-vector. Using a Bayesian approach to update the GP with new preference data \( (\Psi, q) \) gives:

\[
P(f|\mu, K, q) \propto P(q|f, \mu, K, \Psi)P(f|\mu, K, \Psi) = P(q|f, \Psi)P(f|\mu, K).
\]

Here, the first term is just the probabilistic human response model given in equation (1), and the second term is equation (2). However, this posterior does not follow a GP distribution similar to equation (2) and becomes analytically intractable (Jensen and Nielsen, 2011).

Prior works have shown it is possible to perform some approximation such that the posterior is another GP (Jensen and Nielsen, 2011; Rasmussen and Williams, 2005). The most common approximation techniques are:

- Laplace approximation, where the idea is to model the posterior as a GP such that the mode of the likelihood is treated as the posterior mean, and a second-order Taylor approximation around the maximum of the log-likelihood gives the posterior covariance. This technique is computationally very fast.
- Expectation propagation (EP), where the idea is to approximate each factor of the product by a Gaussian. EP is an iterative method that processes each factor iteratively to update the distribution to minimize its KL-divergence with the true posterior. While it is more accurate than Laplace approximation, it is slower in practice (Nickisch and Rasmussen, 2008).

In this paper, we use the former for its computational efficiency. Hence, we now show how to compute the quantities for Laplace approximation, that is, posterior mean and covariance.

4.2.1. Finding the posterior mean. We use the mode of the posterior as an approximation to the posterior mean:
\[
\arg\max_{\Psi}(\log(p(q|\Psi,f)) + \log(P(f|\Psi)))
\tag{4}
\]

Because the preference data are conditionally independent, the first term can be written as:

\[
\log(P(q|\Psi,f)) = \sum_{i=1}^{N} \log P(q_i|\Psi_i,f)
\]

\[
= \sum_{i=1}^{N} \log \Phi\left(\frac{f(\Psi_i^{(1)}) - f(\Psi_i^{(2)})}{\sqrt{2\sigma}}\right)
\]

Due to equation (1) and adopting a zero-mean prior for \(f\), we can write the second term of the optimization (4) as:

\[
\log(P(f|\Psi)) = -\frac{1}{2}\log|K| - N\log2\pi - \frac{1}{2}f^T K^{-1} f
\]

with these two expressions, we can now optimize the log-likelihood and thus find the mode of it to approximate the posterior mean.

### 4.2.2. Finding the posterior covariance matrix.

Following Rasmussen and Williams (2005), and omitting the derivation for brevity, the posterior covariance matrix is \((K^{-1} + W)^{-1}\) where \(W\) is the negative Hessian of the log-likelihood:

\[
W_{ij} = -\frac{\partial^2 \log(P(q|\Psi,f))}{\partial f^{(i)} \partial f^{(j)}}.
\]

Now, we know how to approximate the posterior mean and the posterior covariance for the Laplace approximation of equation (3). This allows us to model and update the reward with preference data using a GP.

We also want to perform inference from this approximated GP. Inference is not only useful for active query generation, but it also enables us to compute the reward expectations and variances given a trajectory.

#### 4.2.2.1. Inference.

Given two points \(\Psi_1^*, \Psi_2^* \in \mathbb{R}^d\), we want to be able to compute the expected mean rewards \(\mu\) and also the covariance matrix between those two points \(K_*\), both of which will be useful for active query generation. These are given by:

\[
\mu_* = \mathbb{E}[f_*|\Psi,q,\Psi_1^*,\Psi_2^*] = k_*^T K_*^{-1} f,
\]

\(k_{*,ij} = k(\Psi_i^*, \Psi_j)\) is a \(2 \times 2N\) matrix, and

\[
K_* = K_0 - k_* (I_{2N} + WK)^{-1} W k_*^T,
\]

where \(K_{0ij} = k(\Psi_i^*, \Psi_j)\) is a \(2 \times 2\) matrix, and \(I_{2N}\) is the \(2N \times 2N\) identity matrix.

Equipped with all these tools which enable us to approximate the posterior distribution with a GP and perform inference over it, we are now ready to present our contributions on the active query generation for reward learning and optimization.

### 4.3. Active query synthesis for reward learning.

While we now know how to learn the reward function \(f\) using only pairwise comparisons, this can require a tremendous amount of data, because each query will give at most 1 bit of information. Furthermore, we can expect a decreasing trend in the information gain as we learn the reward function. Therefore, it is important to select the queries for the human such that each query gives as much information as possible.

For linear reward models, previous work has shown that this can be done by maximizing the mutual information, which also makes the queries easy for the user (Biyik et al., 2019b). Extending this formulation to the reward functions modeled with a GP is not trivial, because one needs to sample from the GP many times for each trajectory, whereas a linear reward form allows the reward prediction after sampling the linear weight terms only once.

Hence, for the active query generation, our goal is to perform information gain maximization with GPs.

#### Problem 1.

Formally, we want to solve the following problem:

\[
\psi_{1*}, \psi_{2*} = \arg \max_{\psi_{1*}, \psi_{2*}} I(f; q|\Psi, \Psi, q),
\]

where \(I\) is the joint information and \(q\) is the response to the query \(\Psi = (\psi_{1*}, \psi_{2*})\). This optimization is equivalent to

\[
\arg \max_{\psi_{1*}, \psi_{2*}} \{H(q|\Psi, \Psi, q) - \mathbb{E}_{q \sim P(q|\Psi, \Psi)}[H(q|\Psi, f)]\},
\]

where \(H\) is the information entropy.

This optimization can be interpreted as follows: On one hand, maximizing the first entropy term \(H(q|\Psi, \Psi, q)\) encourages fast convergence by maximizing the uncertainty of the outcome of every query for the learned GP model. On the other hand, minimizing the second entropy term \(H(q|\Psi, f)\) encourages the ease of responding to the queries by the user, meaning the user should be certain about their choices.

We defer the full derivation of (7) to the appendix, but here we give an easy-to-implement formulation of the optimization. Denoting the posterior mean of \(f(\psi_{1*})\), which is obtained using equation (5), with \(\mu(\psi_{1*})\), the objective function can be written as

\[
h\left(\mu(\psi_{1*}) - \mu(\psi_{2*})\right)
= \frac{1}{2\sigma^2 + g(\psi_{1*}, \psi_{2*})}
\]

\[
-m(\Psi)
\]

where

\[
g(\psi_{1*}, \psi_{2*}) = \text{Var}(f(\psi_{1*})) + \text{Var}(f(\psi_{2*}))
- 2 \text{Cov}(f(\psi_{1*}), f(\psi_{2*}))
\]

whose terms can be computed using equation (6); \(h\) is the binary entropy function, that is,

\[
h(p) = -p \log_2(p) - (1-p) \log_2(1-p),
\]
and

$$m(\Psi) = \frac{\sqrt{\pi \ln(2)\sigma^2 \exp \left(-\frac{(\mu(1) - \mu(2))^2}{\pi \ln(2)\sigma^2 + 2g(\Psi(1), \Psi(2))}\right)}}{\sqrt{\pi \ln(2)\sigma^2 + 2g(\Psi(1), \Psi(2))}}$$

Synthesizing queries that maximize this objective will give very informative data points for preference-based GP regression for reward learning and improve data-efficiency.

Previously, Biyik et al. (2019b) have shown for the linear reward models that using an information gain formulation accelerates the learning whereas volume removal-based methods (such as Sadigh et al., 2017) rely on local optima and can produce trivial queries that compare the exact same trajectory and so gives no information. In the following, we show our formulation also does not suffer from this trivial query problem.

**Remark 1.** The trivial query \(\Psi = \{\Psi^{(d)}, \Psi^{(d)}\}\) does not maximize our acquisition function given in (8), and is in fact a global minimizer.

**Proof.** For the query \(\Psi = \{\Psi^{(d)}, \Psi^{(d)}\}\), we rewrite (8) as

$$h\left(\Phi \left(\frac{\mu(1) - \mu(2)}{2\sigma^2 + g(\Psi^{(d)}, \Psi^{(d)})}\right)\right) - m(\Psi) = 1 - m(\Psi)$$

where \(\text{Var}(f(\Psi^{(d)})) = \text{Cov}(f(\Psi^{(d)}), f(\Psi^{(d)}))\), and so \(g(\Psi^{(d)}, \Psi^{(d)}) = 0\), and

$$m(\Psi) = \frac{\sqrt{\pi \ln(2)\sigma^2 \exp \left(-\frac{(\mu(1) - \mu(2))^2}{\pi \ln(2)\sigma^2 + 2g(\Psi^{(d)}, \Psi^{(d)})}\right)}}{\sqrt{\pi \ln(2)\sigma^2 + 2g(\Psi^{(d)}, \Psi^{(d)})}} = 1$$

which makes the objective value 0. Since the information gain must be nonnegative, this completes the proof that the trivial query is a global minimizer of the objective.

### 4.4. Active query synthesis for reward optimization

In the previous section, we described how one can generate queries to maximize the information gain from query responses. However, note that it is the information gain about the entire reward function, that is, we are trying to learn \(R(\zeta) = f(\Psi(\zeta))\) for every possible trajectory \(\zeta\). In some cases, the ultimate goal of reward learning is to find the best trajectory, that is, \(\arg \max R(\zeta)\). For example, if we are trying to find the most comfortable gait for an exoskeleton user (Tucker et al., 2020b), we do not need to know how two suboptimal gaits compare to each other. Or if we are trying to train a race car to autonomously drive in a fixed parkour, it is sufficient to find the best trajectory.

Mathematically, one could attempt to solve:

$$\arg \max_{\Psi^{(d)}, \Psi^{(d)}} \{I(f; q | \Psi, \Psi, q)$$

subject to \(\Psi^{(2)} = \arg \max_{\Psi} \mathbb{E}_{f \sim \mathcal{P}(f|\Psi, q)} [I(f(\Psi^*)]$$

where we constrain one of the trajectories in the query to be the trajectory that maximizes the mean reward function with respect to the posterior distribution \(P(f|\Psi, q)\). In this way, the algorithm will avoid synthesizing queries that include highly suboptimal trajectories because the first entropy term of the mutual information, that is, \(H(q|\Psi, \Psi, q)\), is very low for the queries where the GP model can already predict the user’s response \(q\).
In the next section, we will validate this approach, as well as our original approach for reward learning, against various baselines in simulation experiments.

5. Simulation experiments

In this section, we present our experiments in three simulation domains to demonstrate how (i) GP rewards improve expressiveness over linear reward functions, and (ii) active query generation improves data-efficiency over random querying, both in reward learning and reward optimization.

5.1. Simulation environments

To validate our framework on robotics tasks, we used three simulation environments: a 2D Driver simulation (Sadigh et al., 2016), a MuJoCo (Todorov et al., 2012) environment to simulate a Tosser robot that tries to throw a capsule-shaped object into a basket (Biyik and Sadigh, 2018), and another MuJoCo environment implemented in OpenAI Gym Brockman et al. (2016) to simulate a Swimmer robot that tries to move as far as possible on a 2D plane. We show images from these environments with sample trajectories in Figure 2. For example in Driver, the user is asked whether they would move forward or backward in the given scenario. In Swimmer, the user is asked whether to move down, or up but a little further. While the users would have a common response to this query, some questions may differ among the users. For instance, in Tosser, the query asks the user whether to throw the ball into the green basket or to drop it instead. Depending on the users’ preferences about the green basket, different users may have different responses.

In the first two environments, we use the following simple features for the function $\Psi$ similar to Biyik and Sadigh (2018):

- **Driver**: Distance to the other car, speed, heading angle, and distance to the closest lane center.
- **Tosser**: The maximum horizontal range, and the number of capsule flips.

In contrast to what the previous work reported, here we do not need to fine-tune the feature parameters to learn the reward functions because GPs can effectively capture nonlinearity. To further demonstrate this, we do not hand-design any features in Swimmer. Instead, we directly use the state and action information, averaged over the time steps, as the features function $\Psi$. Specifically, these correspond to:

- **Swimmer**: Angle of the front tip, angle of the first rotor, angle of the second rotor, velocity of the tip along the $x$-axis, velocity of the tip along the $y$-axis, angular velocity of front tip, angular velocity of first rotor, angular velocity of second rotor, the torque applied on the first rotor, and the torque applied on the second rotor.

5.1.1. Simulated human model. We simulated human responses with an underlying true reward function $f$ with some Gaussian noise, in accordance with equation (1). We modeled $f$ as either a degree-of-two polynomial, or a linear function, or a function that is randomly drawn from the GP prior. In the former two cases, we selected the parameters of $f$ as i.i.d. random samples from the standard normal distribution.

5.2. Baselines

For our analyses, we compared five methods:

- **RANDOMGP**: The reward is modeled using a Gaussian process. The two distinct trajectories selected in each training query are sampled from a training dataset uniformly at random.
- **ACTIVELINEAR**: The reward is modeled as a linear combination of features, and the active query generation method of Biyik et al. (2019b) selects the most informative comparison queries at every step of training.
- **ACTIVEGP**: The reward is modeled as a Gaussian process. We will use our active query generation method to generate the most informative comparison queries to efficiently learn the reward function.

Figure 2. Sample trajectories are shown for the three simulation environments. In Driver, another car is cutting in front of the ego vehicle. In Tosser, the robot must hit the dropping capsule such that it will fall into one of the baskets. In Swimmer, a three-link and two-rotor swimmer robot must move over a two dimensional plane.
• SEMIACTIVEGP: The reward is modeled as a Gaussian process. We will use the variant of our active query generation method, which we modified for reward optimization. We call this method SEMIACTIVEGP since one of the two trajectories in each query is constrained to be the maximizer trajectory of the posterior mean.
• THOMPSONGP: The reward is modeled as a Gaussian process. We will generate the comparison queries such that each query consists of: (1) the trajectory that was preferred in the previous query (an arbitrary trajectory for the first query), and (2) the trajectory that maximizes a reward function sample from the posterior given the data provided thus far. This baseline is from Tucker et al. (2020b).

We generated a training dataset of trajectories with uniformly randomly selected actions. At every iteration of ACTIVEGP and ACTIVELINEAR, we computed the expected information gain of each possible query from this dataset to select the most informative query. Similarly, SEMIACTIVEGP performs a search over this dataset to select the trajectory that will be queried along with the current posterior mean. This approach decreases the computation time compared to solving a continuous optimization over all possible trajectories as it was done by Sadigh et al. (2017); Palan et al. (2019). For fairness, RANDOMGP and THOMPSONGP also used this dataset to select their queries.

5.3. Evaluation

We conduct several analyses:

1. We compare GP reward with linear reward in terms of expressiveness (ACTIVEGP vs ACTIVELINEAR).
2. Next, we compare active query generation for reward learning with random querying baseline in terms of data-efficiency (ACTIVEGP vs RANDOMGP).
3. Finally, we compare all query generation methods that employ a GP reward for optimization in terms of data-efficiency, but also include evaluations for reward learning for completeness (ACTIVEGP vs SEMIACTIVEGP vs THOMPSONGP vs RANDOMGP).

5.3.1. Test set generation. For the first two analyses on expressiveness and data-efficiency, we also generated test sets of trajectories from the same distribution as the training set. However, it would not be fair to use the test set as is. Obtained with uniformly random action sequences, the majority of the training set consists of uninteresting trajectories, for example, the ego agent moves slightly forward and backward (similar to a random walk) in Driver and in Swimmer, or the robot does not hit the capsule in Tossor. Using the test set without further modification would mean we give more importance to these uninteresting behaviors as they form the majority in the datasets. Obviously, this is not the case. We want to learn the reward function everywhere in the dynamically feasible region with equal importance. Hence, we adopted Poisson disk sampling (Bridson, 2007) to get a diverse set of trajectories from the test set. Poisson disk sampling makes sure the difference between trajectories is above some threshold by rejecting the samples that violate this constraint. In this work, we used $L_2$ distance between the feature vectors to quantify these differences. A small example set of samples is compared to uniformly random samples in Figure 3 for the Tossor environment.

After obtaining the diverse test set, we stored the true (noiseless) response of the simulated user for each possible query in this set. For the analysis of expressiveness, we computed the accuracy and the log-likelihood under the reward model, one can consider an approach to combine the learned reward functions. However, as shown in Figures 4(a) and 5(a), even in that case, the GP model captures nonlinearities better than the linear model and provides better learning (Figures 4(b), 5(b) and 6(b)). When the true reward function is linear in features, a linear model naturally learns faster. However, as shown in Figures 4(a) and 5(a), even in that case, the GP model can achieve linear model’s performance. In the Swimmer environment, on the other hand, ACTIVEGP seems to be performing worse than ACTIVELINEAR, although still improving after 200 queries. This is only due to the higher-dimensional feature function. To further improve the reward model, one can consider an approach to combine the linear and GP models by keeping a belief distribution over whether the true reward is linear or not, and actively querying the user according to this belief. We leave this extension as future work.

5.3.2. Expressiveness. Figures 4(a), 4(b), 5(a), 5(b) and 6(a), 6(b) show the results of expressiveness simulations (with five random seeds). When the true reward is polynomial, the linear model results in very high variance in both accuracy and likelihood, because its performance relies on how good a linear function can explain the true nonlinear reward. In this case, the GP model captures nonlinearities better than the linear model and provides better learning (Figures 4(b), 5(b) and 6(b)). When the true reward function is linear in features, a linear model naturally learns faster. However, as shown in Figures 4(a) and 5(a), even in that case, the GP model can achieve linear model’s performance. In the Swimmer environment, on the other hand, ACTIVEGP seems to be performing worse than ACTIVELINEAR, although still improving after 200 queries. This is only due to the higher-dimensional feature function. To further improve the reward model, one can consider an approach to combine the linear and GP models by keeping a belief distribution over whether the true reward is linear or not, and actively querying the user according to this belief. We leave this extension as future work.

5.3.3. Data-efficiency in reward learning. We then evaluated how our active query generation helps with data-efficiency in reward learning. Figure 4(c), 5(c), and 6(c) compare ACTIVEGP and RANDOMGP for the simulation environments (with five random seeds). It can be seen that active querying significantly accelerates learning over random
querying. It should be noted that the number of samples taken via Poisson disk sampling matters: While choosing a very small number will increase the variance in the results, choosing a very large number will make random querying seem like it performs comparable to (or even better than) the active querying as the test set will mostly consist of uninteresting trajectories, which are also abundant in the training set, as we stated earlier.

5.3.4. Data-efficiency in reward optimization. Finally, we compared the query generation methods in terms of their data-efficiency in reward optimization. For this, we randomly sampled a reward function from the GP prior and used it as the true reward function. Our baselines that model the reward as a GP then attempted to learn this true reward function by querying the simulated user up to \( N = 200 \) times. We repeated this experiment in the three simulation environments, with 50 random seeds each. We used more random seeds compared to previous simulation experiments because the regret metric does not take an average over several data points.

Figures 7, 8, and 9 show the results that compare \textsc{ActiveGP}, \textsc{RandomGP}, \textsc{ThompsonGP}, and \textsc{SemiaactiveGP}. It can be seen that \textsc{RandomGP} is significantly worse not only for reward learning, that is, in terms of accuracy and log-likelihood metrics, but also for reward optimization, that is, in terms of the regret metric. In \textsc{Tosser}, the remaining three methods performed comparably. This is arguably because the \textsc{Tosser} environment has only two features, and so it is relatively easy to find the trajectory that maximizes the reward.

On the other hand, we observe some significant differences in the \textsc{Driver} environment. First, \textsc{SemiaactiveGP} led to the best initial reduction in regret—even better than its unconstrained version \textsc{ActiveGP}. The reason for this is easy to see: \textsc{ActiveGP} asks queries that maximize the information about the entire reward function, so it spends some queries to learn about the regions of the reward function that it already knows to be suboptimal. On the other hand, \textsc{SemiaactiveGP} constrains one of the trajectories in the query to be the best trajectory with respect to the posterior, thereby forcing the

![Figure 3](image1). Features of 1000 \textsc{Tosser} trajectories are visualized in two-dimensional plane (gray). Poisson disk sampling allows us to obtain a diverse set of 20 samples (orange), whereas sampling uniformly at random yields mostly uninteresting trajectories (blue).

![Figure 4](image2). Accuracies and average log-likelihoods for test set queries are shown for the \textsc{Driver} environment (mean ± std over five runs). (a) Expressiveness results when the true underlying reward function is linear. (b) Expressiveness results when the true underlying reward function is a degree-of-two polynomial. (c) Data-efficiency results that compare \textsc{ActiveGP} with \textsc{RandomGP}. Accuracies and average log-likelihoods for test set queries are shown (mean±std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood.
search to focus more on the regions of the reward function that may include the optimal trajectory. The differences are even more amplified in the Swimmer environment as it has more features per trajectory. In this environment, SEMIACTIVEGP again leads to the lowest regret initially.

Besides, SEMIACTIVEGP generates queries much faster than ACTIVEGP as it needs to optimize only one trajectory (see Table 1). It should also be noted, however, that SEMIACTIVEGP could not decrease the regret as much as ACTIVEGP in the long term in the Driver environment. This is because

---

**Figure 5.** Accuracies and average log-likelihoods for test set queries are shown for the Tosser environment (mean±std over five runs). (a) Expressiveness results when the true underlying reward function is linear. (b) Expressiveness results when the true underlying reward function is a degree-of-two polynomial. (c) Data-efficiency results that compare ACTIVEGP with RANDOMGP. Accuracies and average log-likelihoods for test set queries are shown (mean ± std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood.

**Figure 6.** Accuracies and average log-likelihoods for test set queries are shown for the Swimmer environment (mean ± std over five runs). (a) Expressiveness results when the true underlying reward function is linear. (b) Expressiveness results when the true underlying reward function is a degree-of-two polynomial. (c) Data-efficiency results that compare ACTIVEGP with RANDOMGP. Accuracies and average log-likelihoods for test set queries are shown (mean ± std). Active query generation improves data-efficiency over random querying in both tasks. This can be seen through both accuracy and log-likelihood.
ACTIVEGP explores the entire landscape of trajectories whereas SEMIACTIVEGP may lack exploration on the regions that it believes to be suboptimal but in fact include the optimal trajectory even though this might a rare case. On the other hand in Swimmer, SEMIACTIVEGP remains as the lowest regret method for the entire 200 queries. While it is possible that ACTIVEGP will again outperform when the data are abundant, experimenting with $N > 200$ is computationally too expensive, as we will discuss in the Conclusion section.

Second, THOMPSONGP performed worse than ACTIVEGP and SEMIACTIVEGP in Driver and eventually in Swimmer, even though it generates each query significantly faster (see Table 1). This might be surprising because THOMPSONGP is based on Thompson sampling, an algorithm that is known to minimize...
cumulative regret over queries in the non-comparative setting, where the user provides direct reward estimations instead of preferences between trajectories. One possible explanation for this result is that THOMPSONGP’s querying is likely to generate queries that consist of trajectories that are close to each other in terms of their reward values. This causes the (simulated) user’s feedback to be more noisy and less informative. On the contrary, both ACTIVEGP and SEMIACTIVEGP try to minimize the user uncertainty as we discussed earlier, thereby potentially leading to better reward optimization results, as well.

6. User studies

6.1. Experiment setup

We also compare our method ACTIVEGP with ACTIVELINEAR and RANDOMGP on a user study with a Fetch mobile manipulator robot (Wise et al., 2016). In this study, the human subjects teach the Fetch robot how to play a variant of mini-golf where the robot can achieve different scores by hitting the ball to different targets (see Figures 1 and 10 for the setup). However, these scores are only known to the human. In fact, the robot does not even know the locations of the targets, and it tries to learn the reward as a function of its control inputs. Fixing some of the joints, we let the robot vary only its shot speed and angle, which are also the features of the reward function.

This experiment setting is interesting because a linear reward function can only encode whether the robot must hit the ball to the right or to the left, or whether it must hit with high or low speed. It cannot particularly encourage (or discourage) hitting with a modest angle and/or speed. Therefore, as we show in Figure 10, the targets that are around the middle region cannot be the maximizers of a linear reward function. Crosses show where the ball hits the ground.

In addition to the accuracy, we assessed whether the robot could successfully learn how to perform a good shot. For this, after the subjects responded to 50 queries, the robot demonstrated three more trajectories each of which correspond to the optimal trajectory of one method, the trajectory that maximizes the learned reward function. Again, the order of these trajectories was shuffled. After watching each demonstration, the subjects assigned a score to the shot from a 9-point rating scale (1-very bad and 9-very good).

6.2. Subjects and procedure

We recruited 10 users (6 males, 4 females) with an age range from 19 to 28. Each user first assigned their distinct scores (from 2 to 9) to the eight targets. The robot then queried them with 50 pairwise comparison questions: 15 for ACTIVEGP, 15 for ACTIVELINEAR, 15 for RANDOMGP, and five queries generated uniformly at random to create a test set. We shuffled the order of queries to avoid any bias. We used the reward models, each of which is learned with 15 queries, to predict the user responses in the test set. The prediction score on the test set provides an accuracy metric.

6.3. Results and discussion

We provide a video that gives an overview of user studies and their results at https://youtu.be/SLSO21Bj9Mw. Figure 11(a) shows the prediction accuracy values on the test sets collected from the subjects (averaged over the subjects). By modeling the reward using a GP and querying the users with the most informative questions, ACTIVEGP achieves significantly higher prediction accuracy (0.74 ± 0.04, mean ± se) compared to both ACTIVELINEAR (0.62 ± 0.07) and RANDOMGP (0.62 ± 0.06) with p < .05.
(Wilcoxon signed rank test). The results from this user study are aligned with our simulation user studies.

In reward learning, it is crucial to validate whether the learned reward function can encode the desired behavior or not. Figure 11(b) shows the user ratings to the trajectories that the robot showed after learning the user preferences via three different methods. ACTIVEGP obtains significantly higher scores (6.9 ± 0.6) than both ACTIVELINEAR (3.4 ± 0.7) and RANDOMGP (5.1 ± 0.7) with \( p < .05 \). While ACTIVELINEAR occasionally achieves high scores when the users’ preferred target is near the edge, it generally fails to produce the desired behavior due to its low expressive power.

7. Conclusion

7.1. Summary

We developed an active preference-based GP regression technique for reward learning and optimization. Our work tackles the lack of expressiveness of reward functions, data-inefficiency, and the incapability to demonstrate or quantify trajectories. Our results in simulations and user studies suggest our method is more successful in expressiveness and data-efficiency in reward learning compared to the baselines. Similarly, we achieve high data-efficiency in reward optimization with a simple modification in our query generation procedure.

7.2. Limitations and future work

We developed our methods only for pairwise comparisons. While extending them to learning from rankings is not mathematically very complicated, its data-efficiency compared to pairwise comparisons needs thorough analysis. Similarly, one could easily incorporate options to denote user uncertainty, which was shown to ease the process for humans (Biyik et al., 2019b; Wilde et al., 2021). GP regression becomes computationally heavy when the domain is high-dimensional (when \( d \) is large). This is a limitation of our work due to the use of GPs and can be alleviated through efficient rank-one GP update approximations (Tucker et al., 2020a). Another computational challenge due to GPs is about the number of preference data samples: when it is too large, inverting the kernel matrix becomes too costly. In such cases, one could consider incorporating some of the data in to the GP prior, for example, to the mean function, or using scalable GPs that are beyond the scope of this work (Hensman et al., 2013; Liu et al., 2020). Although our methods ease the feature design, there still needs to be a design stage—it is often unrealistic to hope \( \Psi(\xi) = \xi \) will work due to the high dimensionality of \( \Xi \). Further research is warranted to simultaneously learn both the reward function \( f \) and the feature function \( \Psi \). Finally, in reward optimization, Thompson sampling based approaches still remain attractive due to their time-efficiency even though they performed worse in terms of data-efficiency. Future work can investigate whether regularizing the Thompson sampling to avoid querying trajectories with similar reward values improves its data-efficiency.

Acknowledgments

We thank Farid Soroush for the early discussions on active preference-based GP regression; Sydney M. Katz, Amir Maleki, and Juan Carlos Aragon for the discussions on alternative ways to ease feature design.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the NSF (National Science Foundation) grants #1941722 and #2218760.

ORCID iD

Erdem Biyik  🐦 https://orcid.org/0000-0002-9516-3130

Notes

1. Parts of this work have been published in Robotics: Science and Systems (RSS) conference (Biyik et al., 2020). This paper adds
the reward optimization variant of the algorithm, and new results on simulation experiments.

2. Due to computational reasons, we assume \( d \) is small. Compared to previous methods that assume \( R \) to be linear in features, this is a very mild assumption.

3. The Python code for active query generation is publicly available at 
https://github.com/Stanford-ILIAD/active-preference-based-gpr.

4. In cases where some human evaluations of the reward functions are available in the form of trajectory-reward pairs, one could add more of these terms to include the information. Thus, our method is easily generalizable to the cases where both preference and reward evaluation data are available.

References

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on machine learning, Banff, AB, July 2004, pp. 1–8.

Abbeel P and Ng AY (2005) Exploration and apprenticeship learning in reinforcement learning. In: Proceedings of the 22nd international conference on machine learning, Bonn, Germany, 7–11 August 2005, pp. 1–8.

Akgun B, Cakmak M, Jiang K, et al. (2012) Keyframe-based learning from demonstration. International Journal of Social Robotics 4(4): 343–355.

Akrour R, Schoenauer M and Sebag M (2012) April: active preference-based reinforcement learning. In: Joint European conference on machine learning and knowledge discovery in databases, Bristol, UK, September 2012, pp. 24116–28131.

Bajcsy A, Losey DP, O’Malley MK, et al. (2017) Learning robot objectives from physical human interaction. Proceedings of Machine Learning Research 78: 217–226.

Bajcsy A, Losey DP, O’Malley MK, et al. (2018) Learning from physical human corrections, one feature at a time. In: ACM/IEEE international conference on Human-Robot Interaction (HRI), Chicago, IL, 05–08 March 2018, pp. 141–149.

Basu C, Yang Q, Hungerman D, et al. (2017) Do you want your autonomous car to drive like you? In: ACM/IEEE international conference on Human-Robot Interaction (HRI), Vienna, Austria, 06–09 March 2017, pp. 417–425.

Basu C, Biyik E, He Z, et al. (2019) Active learning of reward dynamics from hierarchical queries. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, November 2019.

Biyik E (2022) Learning preferences for interactive autonomy. PhD Thesis, Electrical Engineering Department, Stanford University, Stanford, CA.

Biyik E, Palan M, Landolfe NC, et al. (2019b) Asking easy questions: a user-friendly approach to active reward learning. In: Conference on Robot Learning (CoRL), Osaka, Japan, October 2019.

Biyik E and Sadigh D (2018) Batch active preference-based learning of reward functions. In: Conference on Robot Learning (CoRL), Zurich, Switzerland, October 2018, pp. 120–127.

Biyik E, Lazar DA, Sadigh D, et al. (2019a) The green choice: learning and influencing human decisions on shared roads. In: IEEE Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019.

Biyik E, Wang K, Anan N, et al. (2019c) Batch active learning using determinantal point processes. ArXiv Preprint arXiv:1906.07975.

Biyik E, Huynh N, Kochenderfer MJ, et al. (2020) Active preference-based Gaussian process regression for reward learning. In: Proceedings of Robotics: Science and Systems (RSS), Corvalis, OR, 12–16 July 2020. DOI: 10.15607/rss.2020.xvi.041.

Biyik E, Losey DP, Palan M, et al. (2021) Learning reward functions from diverse sources of human feedback: optimally integrating demonstrations and preferences. The International Journal of Robotics Research 41: 45–67. DOI: 10.1177/02783649211041652.

Biyik E, Talati A and Sadigh D (2022) Aprel: a library for active preference-based reward learning algorithms. In: 17th ACM/IEEE international conference on Human-Robot Interaction (HRI), Sapporo, Japan, 07–10 March 2022. DOI: 10.1109/hr153351.2022.9889650.

Bridson R (2007) Fast Poisson disk sampling in arbitrary dimensions. SIGGRAPH Sketches 10: 1278780–1278807.

Brockman G, Cheung V, Pettersson L, et al. (2016) Openai gym. ArXiv Preprint arXiv:1606.01540.

Brown DS and Nickum S (2019) Deep Bayesian reward learning from preferences. In: NeurIPS workshop on safety and robustness in decision making, Vancouver, BC, December 2019, pp. n/a.

Cakmak M, Srinivasa SS, Lee MK, et al. (2011) Human preferences for robot-human hand-over configurations. In: 2011 IEEE/RSJ international conference on intelligent robots and systems, San Francisco, CA, 25–30 September 2011, pp. 1986–1993.

Casper S, Davies X, Shi C, et al. (2023) Open problems and fundamental limitations of reinforcement learning from human feedback. ArXiv Preprint arXiv:2307.15217.

Christiano PF, Leike J, Brown T, et al. (2017) Deep reinforcement learning from human preferences. In: Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, December 2017, pp. 4299–4307.

Chu W and Gharahmani Z (2005) Preference learning with Gaussian processes. In: International Conference on Machine Learning (ICML), Bonn, Germany, August 2005, pp. 137–144.

Clark J and Amodei D (2016) Faulty reward functions in the wild. https://openai.com/blog/faulty-reward-functions/

Daniel C, Kroemer O, Viering M, et al. (2015) Active reward learning with a novel acquisition function. Autonomous Robots 39(3): 389–405.

Dragan AD and Srinivasa SS (2012) Formalizing assistive teleoperation. Robotics: Science and Systems. Cambridge, MA: MIT Press.

González J, Dai Z, Damianou A, et al. (2017) Preferential Bayesian optimization. In: International Conference on Machine Learning (ICML), Sydney, Australia, August 2017, pp. 1282–1291.

Hensman J, Fusi N and Lawrence ND (2013) Gaussian processes for big data. Uncertainty in Artificial Intelligence. Catalina Island, CA: Citeseer, 282.

Ho J and Ermon S (2016) Generative adversarial imitation learning. In: Advances in Neural Information Processing Systems (NeurIPS), Barcelona, Spain, December 2016, pp. 4565–4573.

Houselby N, Huszár F and Gharahmani Z, et al. (2011) Bayesian active learning for classification and preference learning. In: NIPS workshop on Bayesian optimization, experimental design and bandits: theory and applications. (accessed December 2011).
Houlsby N, Huszar F, Ghahramani Z, et al. (2012) Collaborative Gaussian processes for preference learning. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Lake Tahoe, NV, December 2012, pp. 2096–2104.

Ibarz B, Leike J, Pohlen T, et al. (2018) Reward learning from human preferences and demonstrations in atari. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Montreal, QC, December 2018, pp. 8011–8023.

Javadi S, Srinivasa SS and Bagnell JA (2015) Shared autonomy via hindsight optimization. *Robotics: Science and Systems* 2015.

Jensen BS and Nielsen JB (2011) Pairwise judgements and absolute ratings with gaussian process priors. Technical Report, Department of Applied Mathematics and Computer Science, University of Copenhagen (DTU).

Jeon HJ, Milli S and Dragan A (2020) Reward-rational (implicit) choice: a unifying formalism for reward learning. *Advances in Neural Information Processing Systems* 33: 4415–4426.

Kapoor A, Grauman K, Urtasun R, et al. (2007) Active learning with Gaussian processes for object categorization. In: International Conference on Computer Vision (ICCV), Rio de Janeiro, Brazil, October 2007, pp. 1–8.

Kazts SM, Bilhan ACL and Kochenderfer MJ (2019) Learning an urban air mobility encounter model from expert preferences. In: *Digital Avionics Systems Conference (DASC)*, San Diego, CA, September 2019, pp. 1–8.

Katz SM, Maleki A, Bryk E, et al. (2021) Preference-based learning of reward function features. *ArXiv Preprint* arXiv:2103.02727.

Khurshid RP and Kuchenbecker KJ (2015) Data-driven motion mappings improve transparency in teleoperation. *Presence: Teleoperators and Virtual Environments* 24(2): 132–154.

Kupcsik A, Hsu D and Lee WS (2018) Learning dynamic robot-to-human object handover from human feedback. *Robotics Research* 1: 161–176.

Lee K, Smith L and Abbeel P (2020) Pebble: feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In: International conference on machine learning, *Virtual*, July 2021.

Lepid JR, Owen MP and Kochenderfer MJ (2015) Bayesian preference elicitation for multiobjective engineering design optimization. *Journal of Aerospace Information Systems* 12(10): 634–645.

Liu H, Ong YS, Shen X, et al. (2020) When Gaussian process meets big data: a review of scalable gps. *IEEE Transactions on Neural Networks and Learning Systems* 31(11): 4405–4423.

Losey DP, Srinivasan K, Mandlekar A, et al. (2020) Controlling assistive robots with learned latent actions. In: International Conference on Robotics and Automation (ICRA), *Virtual*, June 2020, pp. 378–384.

Myers V, Biyik E, Anari N, et al. (2021) Learning multimodal rewards from rankings. In: Conference on Robot Learning, London, UK, November 2021, pp. 342–352. *PMLR*.

Myers V, Biyik E and Sadigh D (2023) Active reward learning from online preferences. In: International Conference on Robotics and Automation (ICRA), London, UK, May 2023.

Ng AY and Russell SJ (2000) *Algorithms for inverse reinforcement learning*. In: International Conference on Machine Learning (ICML), Stanford, CA, June 2000, 1, p. 2.

Ng AY, Harada D and Russell S (1999) Policy invariance under reward transformations: theory and application to reward shaping. *International Conference on Machine Learning (ICML)* 99: 278–287.

Nickisch H and Rasmussen CE (2008) Approximations for binary Gaussian process classification. *Journal of Machine Learning Research* 9(Oct): 2035–2078.

Ouyang L, Wu J, Jiang X, et al. (2022) Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35: 27730–27744.

Palan M, Landolfi NC, Shevchuk G, et al. (2019) Learning reward functions by integrating human demonstrations and preferences. In: *Robotics: science and systems*, Messe Freiburg, Germany, June 2019.

Racca M, Oulasvirta A and Kyriki V (2019) Teacher-aware active robot learning. In: *ACM/IEEE international conference on Human-Robot Interaction (HRI)* Daegu and South Korea, March 2019, pp. 335–343.

Ramachandran D and Amir E (2007) *Bayesian inverse reinforcement learning*. In: International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India, January 2007, 7, pp. 2586–2591.

Rasmussen CE and Williams CK (2005) *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Ross S, Melik-Barkhudarov N, Shankar KS, et al. (2013) Learning monocular reactive uav control in cluttered natural environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013, pp. 1765–1772.

Sadigh D, Sastry S, Seshia SA, et al. (2016) Planning for autonomous cars that leverage effects on human actions. In: *Robotics: science and systems*, Cambridge, MA, July 2016.

Sadigh D, Dragan AD, Sastry SS, et al. (2017) Active preference-based learning of reward functions. In: *Robotics: science and systems*, Cambridge, MA, July 2017.

Sharma P, Sundaralingam B, Blukis V, et al. (2022) Correcting robot plans with natural language feedback. *ArXiv Preprint* arXiv:2204.05186.

Song J, Ren H, Sadigh D, et al. (2018) Multi-agent generative adversarial imitation learning. In: *Advances in Neural Information Processing Systems (NeurIPS)*, Montreal, QC, December 2018, pp. 7461–7472.

Sontakke SA, Arnold S, Zhang J, et al. (2023) Roboclip: one demonstration is enough to learn robot policies. In: *Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, December 2023.

Stadie BC, Abbeel P and Sutskever I (2017) Third-person imitation learning. In: International Conference on Learning Representations (ICLR), Toulon, France, April 2017.

Todorov E, Erez T and Tassa Y (2012) Mujoco: a physics engine for model-based control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Algarve, Portugal, October 2012, pp. 5026–5033.

Tucker M, Cheng M, Novoseller E, et al. (2020a) *Human preference-based learning for high-dimensional optimization of exoskeleton walking gaits*. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, October 2020, pp. 3423–3430.

Wilde N, Kulic D and Smith SL (2019) Bayesian active learning for collaborative task specification using equivalence regions. *IEEE Robotics and Automation Letters* 4(2): 1691–1698.
Wilde N, Kulić D and Smith SL (2020) Active preference learning using maximum regret. In: 2020 IEEE/RSJ International conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, 24 October 2020–24 January 2021, pp. 10952–10959, IEEE.
Tucker M, Novoseller E, Kann C, et al. (2020b) Preference-based learning for exoskeleton gait optimization. In: IEEE International Conference on Robotics and Automation (ICRA), Virtual, June 2020.
Wilde N, Biyik E, Sadigh D, et al. (2021) Learning reward functions from scale feedback. In: 5th annual conference on robot learning, London, UK, November 2021.
Wirth C, Akrour R, Neumann G, et al. (2017) A survey of preference-based reinforcement learning methods. Journal of Machine Learning Research 18(136): 1–46.
Wise M, Ferguson M, King D, et al. (2016) Fetch and freight: standard platforms for service robot applications. In: Workshop on autonomous mobile service robots, New York City, NY, July 2016.
Ziebart BD, Maas AL, Bagnell JA, et al. (2008) Maximum entropy inverse reinforcement learning. In: AAAI conference on Artificial Intelligence (AAAI), Chicago, IL, July 2008, 8, pp. 1433–1438.

Appendix

Active query generation derivation

Let $\Sigma$ be the posterior covariance matrix between $f(\Psi^{(1)})$ and $f(\Psi^{(2)})$. And let

$$
\Sigma^{-1} = \begin{bmatrix} c & d \\ d & c' \end{bmatrix}.
$$

Throughout the derivation, all integrals are calculated over $\mathbb{R}$, but we drop it to simplify the notation. We write the first entropy term in the optimization (7) as:

$$
H(q|\Psi, q) = h \left( \int \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) N \left( \left[ f^{(1)}, f^{(2)} \right], \left[ \mu^{(1)}, \mu^{(2)} \right], \Sigma \right) df^{(2)} df^{(1)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} \right)^2 + c' \left( f^{(2)} - \mu^{(2)} \right)^2 + 2d \left( f^{(1)} - \mu^{(1)} \right) \left( f^{(2)} - \mu^{(2)} \right) \right)} df^{(2)} df^{(1)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} \right)^2 + 2d \left( f^{(1)} - \mu^{(1)} \right) \left( f^{(2)} - \mu^{(2)} \right) \right)} df^{(2)} df^{(1)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} df^{(2)} df^{(1)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} df^{(2)} df^{(1)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$

$$
= h \left( \frac{\sqrt{cc' - dd'}}{2\pi} \int \frac{1}{\sqrt{2\sigma}} \Phi \left( \frac{f^{(1)} - f^{(2)}}{\sqrt{2}\sigma} \right) e^{-\frac{1}{2} \left( c \left( f^{(1)} - \mu^{(1)} + d \left( f^{(2)} - \mu^{(2)} \right) \right)^2 \right)} \frac{1}{\sqrt{2\pi}} df^{(1)} df^{(2)} \right)
$$
Using the mathematical identity \( \int \phi(x)N(x; \mu, \sigma^2)dx = \phi(\mu/\sqrt{1 + \sigma^2}) \), we obtain

\[
H(q|\Psi, \Psi, q) = \Phi \left( \frac{\mu^{(1)} - \mu^{(2)}}{\sqrt{2\sigma^2 + g(\Psi^{(1)}, \Psi^{(2)})}} \right)
\]

where \( g(\Psi^{(1)}, \Psi^{(2)}) = \text{Var}(f(\Psi^{(1)})) + \text{Var}(f(\Psi^{(2)})) - 2\text{Cov}(f(\Psi^{(1)}), f(\Psi^{(2)})) \)

One can then expand the expression in the denominator and use the facts that \( \text{Var}(f(\Psi^{(1)})) = c'c\sigma'' - d^2 \), \( \text{Var}(f(\Psi^{(2)})) = c'c\sigma'' - d^2 \) and \( \text{Cov}(f(\Psi^{(1)}), f(\Psi^{(2)})) = -d/c\sigma'' - d^2 \) to obtain

\[
H(q|\Psi, \Psi, q) = h \left( \Phi \left( \frac{\mu^{(1)} - \mu^{(2)}}{\sqrt{2\sigma^2 + g(\Psi^{(1)}, \Psi^{(2)})}} \right) \right)
\]

We next make the derivation for the second entropy term. To simplify the notation, we let \( \sigma'^2 = \pi \ln (2)/2 \), \( \sigma''^2 = \sigma^2 + 1/c \), and \( \sigma^2 = c\left(1 + \frac{c}{2}\right)^2/c'c - d^2 \). By performing a linearization over the logarithm of the second entropy term as in Housby et al. (2011),
By another change of variables for the outer integral with \( v = f^{(2)}/c + d/c f^{(2)} \),

\[
\mathbb{E}_{\mathcal{P}(\Psi, q) | H(q | \Psi, f)} = \frac{\sqrt{c e - d^2}}{2 \pi} \int e^{-\frac{(v - \mu^{(2)})^2}{2 \sigma^2}} e^{-\frac{|v|^2}{2 c}} \frac{1}{\sqrt{2 \pi} \sqrt{\sigma^2 + \frac{1}{c}}} dv
\]

By identifying the inner integral as a convolution of two Gaussians, we get

\[
\mathbb{E}_{\mathcal{P}(\Psi, q) | H(q | \Psi, f)} = \frac{\sqrt{c e - d^2}}{2 \pi} \int e^{-\frac{(v - \mu^{(2)})^2}{2 \sigma^2}} e^{-\frac{|v|^2}{2 c}} \frac{1}{\sqrt{2 \pi} \sqrt{\sigma^2 + \frac{1}{c}}} dv
\]
By repeating the same convolution trick for the second integral,

\[ E_{f \sim P(f|\Psi, q)}[H(q|\Psi, f)] = \frac{1}{1 + \frac{d}{c}} \sqrt{c' c - d'^2} \cdot \frac{1}{\sqrt{\pi}} \frac{1}{\sqrt{\sigma^2 + \frac{1}{c}}} \frac{2\pi \sigma_\beta \sigma''}{\sqrt{2\pi \sigma_b^2 + \sigma''^2}} \frac{1}{\frac{2}{\sigma_b^2 + \sigma''^2}} \left( \frac{(\mu^{(1)} - \mu^{(2)})^2}{\sigma_b^2 + \sigma''^2} \right) \]

Again, we express this in terms of covariance and variance expressions:

\[ E_{f \sim P(f|\Psi, q)}[H(q|\Psi, f)] = \frac{\sqrt{\ln(2)} \sigma^2 \exp \left( -\frac{(\mu_a - \mu_b)^2}{\ln(2) \sigma^2 + 2g(\Psi^{(1)}, \Psi^{(2)})} \right)}{\sqrt{\ln(2) \sigma^2 + 2g(\Psi^{(1)}, \Psi^{(2)})}} \]