The International Journal of Robotics Research
XX(X):1–17
©The Author(s) 2023
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Active Preference-Based Gaussian Process Regression for Reward Learning and Optimization

Erdem Bıyık[1,2,3], Nicolas Huynh[4,5], Mykel J. Kochenderfer[6], and Dorsa Sadigh[1,7]

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, e.g. 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

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, e.g. 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; Ng et al. 1999; Christiano et al. 2017). Our goal in this work is to develop a *data-efficient* method capable of learning *expressive* reward 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 (Cakmak et al. 2011; Ibarz et al. 2018; Brown and Niekum 2019; Tucker et al. 2020b; Sadigh et al. 2017; Akrour et al. 2012; Lepird et al. 2015; Lee et al. 2021; Biyik 2022). For example, as shown in Figure 1, the robot can demonstrate the blue and green trajectories, ξ_A and ξ_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 ξ_A is preferred over ξ_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 (Sadigh et al. 2017; Biyik and Sadigh 2018; Biyik et al. 2019c,b; Basu et al. 2019; Palan et al. 2019; Katz et al.

[1]Department of Electrical Engineering, Stanford University
[2]Center for Human-Compatible Artificial Intelligence, UC Berkeley
[3]Thomas Lord Department of Computer Science, University of Southern California
[4]Department of Applied Mathematics, École Polytechnique
[5]Department of Computer Science and Technology, University of Cambridge
[6]Department of Aeronautics and Astronautics, Stanford University
[7]Department of Computer Science, Stanford University

Corresponding author:
Erdem Bıyık
Email: biyik@usc.edu

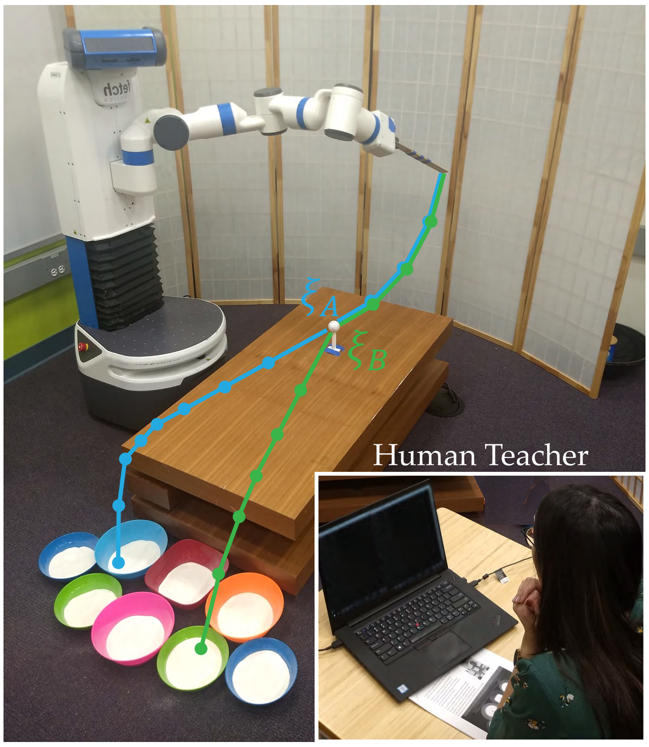

Figure 1. The user is trying to teach the robot how to play a variant of mini-golf, where the reward differs among eight targets. In preference-based learning, instead of trying to design a reward function by hand or controlling the robot to provide demonstrations, the user simply compares two demonstrated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

2019; Biyik et al. 2019a; Racca et al. 2019; Wilde et al. 2019; Akrour et al. 2012; Wilde et al. 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, i.e., 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 (Lee et al. 2021; Christiano et al. 2017; 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 non-parametric 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. 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[*]:

- 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.

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.

Learning reward functions from demonstrations. Learning reward functions via collected expert demonstrations has been extensively studied. This is commonly referred to

[*]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.

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 (Losey et al. 2020; Akgun et al. 2012; Dragan and Srinivasa 2012; Javdani et al. 2015; Khurshid and Kuchenbecker 2015). 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.

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 Niekum 2019; Myers et al. 2022) 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 (Jeon et al. 2020; Biyik et al. 2021). The reward function assumed by these priors 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 poses 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.

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,a) 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 (Wirth et al. 2017; Ouyang et al. 2022; Casper et al. 2023), 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.

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 González et al. (2017) required regression of a GP over $2d$-dimensions to model a d-dimensional function, which causes a computational burden. Houlsby et al. (2012) presented an active method for preference-based GP regression. However, similar to González et al. (2017), the regression was done over a $2d$-dimensional space where the learned model predicts the outcome of a comparison rather than outputting a reward value.

Jensen and Nielsen (2011) showed how to update a GP with preference data, but the active query generation was not of interest. Similarly, Kupcsik et al. (2018) developed and used a preference-based GP regression method for optimizing robot-to-human object handover. This work differs from the others by using on-policy trajectories for preference queries: instead of using a fixed dataset of queries, they generated them using a policy that optimizes for the reward function that is learned up to that iteration. Differently from our work, which is off-policy but generates queries actively from a pool, this work did not focus on pool-based active querying, i.e., they did not optimize an acquisition function to select a query from a set of candidate queries for data-efficiency. And importantly, they only considered the reward optimization problem, but not the reward learning problem.

Kapoor et al. (2007) formulated an active learning strategy for classification with GPs. This is a special case of our problem, as the labels in classification are consistent, i.e., the labels assigned to the samples in the dataset, even if they are incorrect, remain unchanged during training. In our case, the user's response to the same preference query may differ based on their noise model. Houlsby et al. (2011) and Daniel et al. (2015) proposed active GP regression methods for classification and reward learning, respectively. 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.

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: $\xi = (s_0, a_0, s_1, a_1, \dots, s_T, a_T)$, where T is the finite time horizon. We assume a reward function over trajectories, $R : \Xi \to \mathbb{R}$, 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 $\Psi : \Xi \to \mathbb{R}^d$ defines a set of trajectory features. These features are often hand-designed, e.g. 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, ξ_A and ξ_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:

$$P(q = \xi_A \mid Q = \{\xi_A, \xi_B\}) = P\left(R(\xi_A) - R(\xi_B) > v\right),$$

where $q \in Q$ denotes the user's choice, and $v \sim \mathcal{N}(0, 2\sigma^2)$ for some standard deviation $\sigma\sqrt{2}$. Therefore, equivalently:

$$P(q = \xi_A \mid Q = \{\xi_A, \xi_B\}) = \Phi\left(\frac{R(\xi_A) - R(\xi_B)}{\sqrt{2}\sigma}\right), \quad (1)$$

where Φ is the cumulative distribution function of the standard normal.

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.

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 Ψ, with superscripts and subscripts when needed.

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 $\mathbf{\Psi} = (\Psi_i)_{i=1}^N$, 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 nonlinearities and uncertainties without introducing

[†]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.

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

parameters. We have:

$$P(\mathbf{f} \mid \boldsymbol{\mu}, \mathbf{K}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{f}-\boldsymbol{\mu})^\top \mathbf{K}^{-1}(\mathbf{f}-\boldsymbol{\mu})\right)}{(2\pi)^{N/2}|\mathbf{K}|^{1/2}}, \qquad (2)$$

where $\mathbf{f} = (f(\Psi_i))_{i=1}^N$, $\boldsymbol{\mu}$ and $\mathbf{K}$ are the mean vector and the covariance matrix of the GP distribution for the N items in the dataset. Here, $\mathbf{f}$ 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 θ:

$$k(\Psi_i, \Psi_j) = \exp\left(-\theta\|\Psi_i - \Psi_j\|_2^2\right) - \bar{k}(\Psi_i, \Psi_j),$$
$$\bar{k}(\Psi_i, \Psi_j) = \exp\left(-\theta\|\Psi_i - \bar{\Psi}\|_2^2 - \theta\|\Psi_j - \bar{\Psi}\|_2^2\right),$$

where $\bar{\Psi} \in \mathbb{R}^d$ is an arbitrary point for which we assume $f(\bar{\Psi}) = 0$. This is important because the query responses only depend on the relative difference between the two reward function values at the trajectories, i.e., $f(\Psi) + c$ for any $c \in \mathbb{R}$ would have the same likelihood for a dataset as $f(\Psi)$. By setting $f(\bar{\Psi}) = 0$ for some arbitrary $\bar{\Psi} \in \mathbb{R}^d$, we dissolve this ambiguity. It does not introduce an assumption because, for any function f' and for any point $\bar{\Psi}$, one can define $f(\Psi) = f'(\Psi) - f'(\bar{\Psi})$ without loss of generality—both f' and f 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.

Preference-based GP Regression

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

$$\begin{aligned} P(f \mid \boldsymbol{\mu}, \mathbf{K}, \Psi, q) &\propto P(q \mid f, \boldsymbol{\mu}, \mathbf{K}, \Psi) P(f \mid \boldsymbol{\mu}, \mathbf{K}, \Psi) \\ &= P(q \mid f, \Psi) P(f \mid \boldsymbol{\mu}, \mathbf{K}). \qquad (3) \end{aligned}$$

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, i.e., posterior mean and covariance.

Finding the posterior mean. We use the mode of the posterior as an approximation to the posterior mean:

$$\arg\max_{\mathbf{f}} \left(\log\left(p(\mathbf{q} \mid \boldsymbol{\Psi}, \mathbf{f})\right) + \log\left(P(\mathbf{f} \mid \boldsymbol{\Psi})\right)\right) \qquad (4)$$

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

$$\begin{aligned} \log\left(P(\mathbf{q} \mid \boldsymbol{\Psi}, \mathbf{f})\right) &= \sum_{i=1}^N \log P(q_i \mid \Psi_i, \mathbf{f}) \\ &= \sum_{i=1}^N \log \Phi\left(\frac{f(\Psi_i^{(1)}) - f(\Psi_i^{(2)})}{\sqrt{2}\sigma}\right) \end{aligned}$$

due to Equation (1). Adopting a zero-mean prior for f, we can write the second term of the optimization (4) as:

$$\log\left(P(\mathbf{f} \mid \boldsymbol{\Psi})\right) = -\frac{1}{2}\log|\mathbf{K}| - N\log 2\pi - \frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}$$

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

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

$$W_{ij} = -\frac{\partial^2 \log\left(P(\mathbf{q} \mid \boldsymbol{\Psi}, \mathbf{f})\right)}{\partial f^{(i)} \partial f^{(j)}}.$$

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.

Inference. Given two points $\Psi_*^{(1)}, \Psi_*^{(2)} \in \mathbb{R}^d$, we want to be able to compute the expected mean rewards $\boldsymbol{\mu}_*$ and also the covariance matrix between those two points $\mathbf{K}_*$, both of which will be useful for active query generation. These are given by:

$$\mu_* = \mathbb{E}\left[\mathbf{f}_* \mid \boldsymbol{\Psi}, \mathbf{q}, \Psi_*^{(1)}, \Psi_*^{(2)}\right] = k_*^\top \mathbf{K}^{-1}\mathbf{f}, \qquad (5)$$

[§] 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 those information. Thus, our method is easily generalizable to the cases where both preference and reward evaluation data are available.

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

$$\mathbf{K}_* = \boldsymbol{K_0} - k_* \left(\boldsymbol{I}_{2N} + \mathbf{WK}\right)^{-1} \mathbf{W} k_*^\top, \tag{6}$$

where $K_{0ij} = k\left(\Psi_*^{(i)}, \Psi_*^{(j)}\right)$ is a 2×2 matrix, and $\boldsymbol{I}_{2N}$ 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.

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:*

$$\Psi_*^{(1)}, \Psi_*^{(2)} = \underset{\Psi^{(1)},\Psi^{(2)}}{\arg\max}\, I(f; q \mid \Psi, \boldsymbol{\Psi}, \mathbf{q}),$$

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

$$\underset{\Psi^{(1)},\Psi^{(2)}}{\arg\max} \left(H(q \mid \Psi, \boldsymbol{\Psi}, \mathbf{q}) - \mathbb{E}_{f\sim P(f\mid\boldsymbol{\Psi},\mathbf{q})}\left[H(q \mid \Psi, f)\right]\right), \tag{7}$$

where H is the information entropy.

This optimization can be interpreted as follows: On one hand, maximizing the first entropy term $H(q \mid \Psi, \boldsymbol{\Psi}, \mathbf{q})$ 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 $H(q \mid \Psi, f)$ 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 $f(\Psi^{(i)})$, which is obtained using Equation (5), with $\mu^{(i)}$, the objective function can be written as

$$h\left(\Phi\left(\frac{\mu^{(1)} - \mu^{(2)}}{\sqrt{2\sigma^2 + g(\Psi^{(1)}, \Psi^{(2)})}}\right)\right) - m\left(\Psi\right) \tag{8}$$

where

$$\begin{aligned} g(\Psi^{(1)}, \Psi^{(2)}) =& \mathrm{Var}\left(f(\Psi^{(1)})\right) + \mathrm{Var}\left(f(\Psi^{(2)})\right) \\ &- 2\,\mathrm{Cov}\left(f(\Psi^{(1)}), f(\Psi^{(2)})\right), \end{aligned}$$

whose terms can be computed using Equation (6); h is the binary entropy function, i.e.,

$$h(p) = -p\log_2(p) - (1-p)\log_2(1-p),$$

and

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

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 $\Psi = \{\Psi^{(A)}, \Psi^{(A)}\}$ does not maximize our acquisition function given in (8), and is in fact a global minimizer.*

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

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

where $\mathrm{Var}\left(f(\Psi^{(A)})\right) = \mathrm{Cov}\left(f(\Psi^{(A)}), f(\Psi^{(A)})\right)$, and so $g(\Psi^{(A)}, \Psi^{(A)}) = 0$, and

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

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.

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, i.e., we are trying to learn $R(\xi) = f(\Psi(\xi))$ for every possible trajectory ξ. In some cases, the ultimate goal of reward learning is to find the best trajectory, i.e., $\arg\max_\xi R(\xi)$. 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:

$$\arg\max_{\Psi^{(1)},\Psi^{(2)}} I(\xi^*; q \mid \Psi, \mathbf{\Psi}, \mathbf{q}),$$

at every iteration of querying, where ξ^* is the trajectory that maximizes the reward function. However, this is not computationally tractable. To see why, we equivalently write:

$$\arg\max_{\Psi^{(1)},\Psi^{(2)}} \left(H(q \mid \Psi, \mathbf{\Psi}, \mathbf{q}) - \mathbb{E}_{\xi^* \sim P(\xi^* \mid \mathbf{\Psi}, \mathbf{q})}\left[H(q \mid \Psi, \xi^*)\right]\right).$$

While the first entropy term is the same as in Equation 7, the second term requires computing $H(q \mid \Psi, \xi^*)$, i.e., the entropy of the user's response to the query conditioned only on the query itself and the best trajectory being ξ^*. Computing this is possible through rejection sampling: we could sample reward functions from $P(f \mid \mathbf{\Psi}, \mathbf{q})$ and only accept those whose maximizer is ξ^*. This introduces not only potential numerical instabilities but also a huge computational burden. Besides, we will have to do this rejection sampling for every ξ^* sample from $P(\xi^* \mid \mathbf{\Psi}, \mathbf{q})$ due to the expectation in the second term.

Therefore, while mathematically attractive, maximizing the information gain about the reward maximizer trajectory ξ^* is not viable in practice. Instead, we identify why our active query synthesis method for reward learning is not readily suitable for reward optimization, and modify it accordingly to adapt to reward optimization.

The problem with our query synthesis approach for reward learning is the fact that it is trying to learn the reward function over the entire landscape of trajectories. Hence, it will often query the user with two trajectories that it knows to be suboptimal with high probability. This is because it needs to learn the reward values of those trajectories, too, even though they are suboptimal. Instead, we want the active query synthesis for reward optimization to focus only on the regions of the trajectory space where the reward function may attain its globally maximum value. To this end, we propose a simple modification to our previous approach and solve:

$$\begin{aligned} &\arg\max_{\Psi^{(1)},\Psi^{(2)}} && I(f; q \mid \Psi, \mathbf{\Psi}, \mathbf{q}) \\ &\text{subject to} && \Psi^{(2)} = \arg\max_{\Psi'} \mathbb{E}_{f \sim P(f \mid \mathbf{\Psi}, \mathbf{q})}[f(\Psi')], \end{aligned}$$

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 $P(f \mid \mathbf{\Psi}, \mathbf{q})$. In this way, the algorithm will avoid synthesizing queries that include highly suboptimal trajectories because the first entropy term of the mutual information, i.e., $H(q \mid \Psi, \mathbf{\Psi}, \mathbf{q})$, 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.

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.

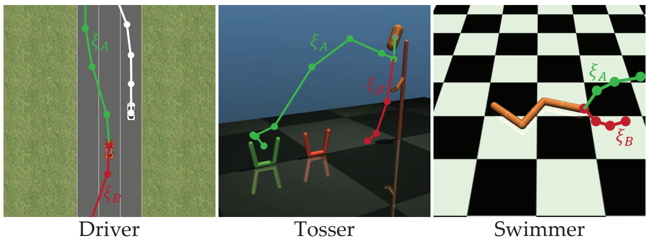

Figure 2. Sample trajectories are shown for the three simulation environments. In *Driver*, another car is cutting in front of the ego vehicle. In *Tosser*, the robot must hit the dropping capsule such that it will fall into one of the baskets. In *Swimmer*, a three-link and two-rotor swimmer robot must move over a two dimensional plane.

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 further 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 Ψ 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 nonlinearities. 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 Ψ. 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.

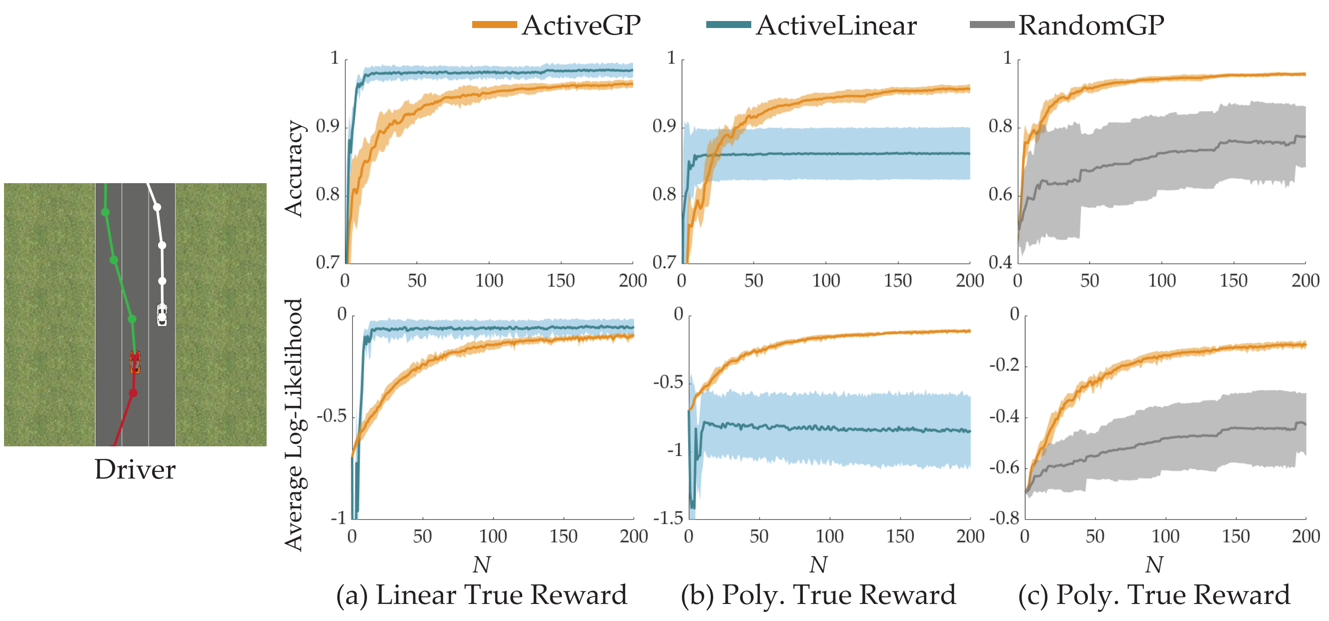

Figure 3. Accuracies and average log-likelihoods for test set queries are shown for the *Driver* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two 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.

With 10 features, the *Swimmer* environment enables us to investigate our methods in higher-dimensional problems.

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.

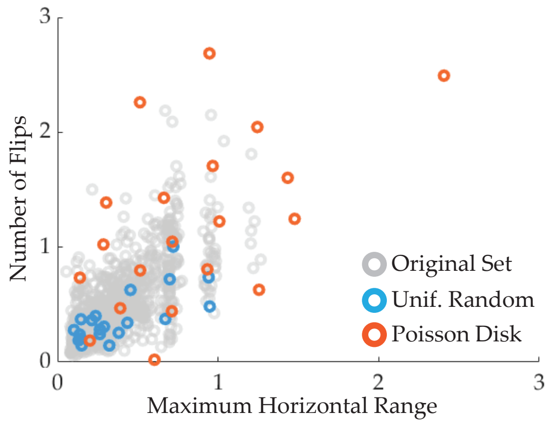

Figure 4. Features of 1000 *Tosser* trajectories are visualized in two-dimensional plane (gray). Poisson disk sampling allows us to obtain a diverse set of 20 samples (orange), whereas sampling uniformly at random yields mostly uninteresting trajectories (blue).

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.
- 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.

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).

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, e.g. 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 *Tosser*. 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 4 for the *Tosser* 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 of the true responses under the reward functions that are learned with N actively chosen queries (up to $N = 200$). While existing methods that adaptively select the number of queries considering the cost of making each query (Biyik et al. 2019b) is applicable to our ACTIVEGP method, we leave its evaluation to future work.

For data-efficiency in reward learning analysis, we again used the true human responses to the queries in the diverse test set (only from the polynomial reward functions) to calculate the accuracy and the log-likelihood under the learned reward functions.

For data-efficiency in reward optimization, we calculated regret: the true reward difference between the actual best trajectory and the predicted best trajectory. A data-efficient optimization algorithm should be able to decrease regret within as few queries as possible.

Expressiveness. Figures 3(a,b), 5(a,b) and 6(a,b) show the results of expressiveness simulations (with 5 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 3(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 3(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.

Data-Efficiency in Reward Learning. We then evaluated how our active query generation helps with data-efficiency in reward learning. Figures 3(c), 5(c), and 6(c) compare ACTIVEGP and RANDOMGP for the simulation environments (with 5 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.

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 ACTIVEGP, RANDOMGP, THOMPSONGP, and SEMIACTIVEGP. It can be seen that RANDOMGP is significantly worse not only for reward learning, i.e., in terms of accuracy and log-likelihood metrics, but also for reward optimization, i.e., in terms of the regret metric. In *Tosser*, the remaining three methods performed comparably. This is arguably because the *Tosser* environment has only 2 features, and so

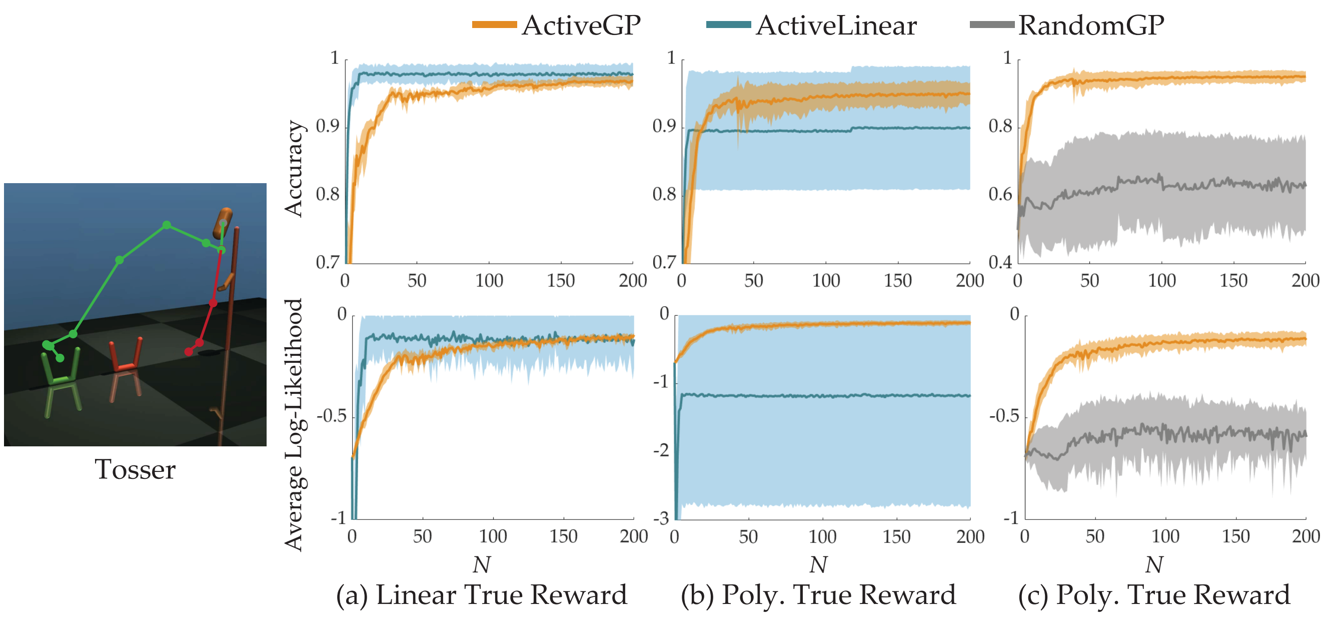

Figure 5. Accuracies and average log-likelihoods for test set queries are shown for the *Tosser* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two 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.

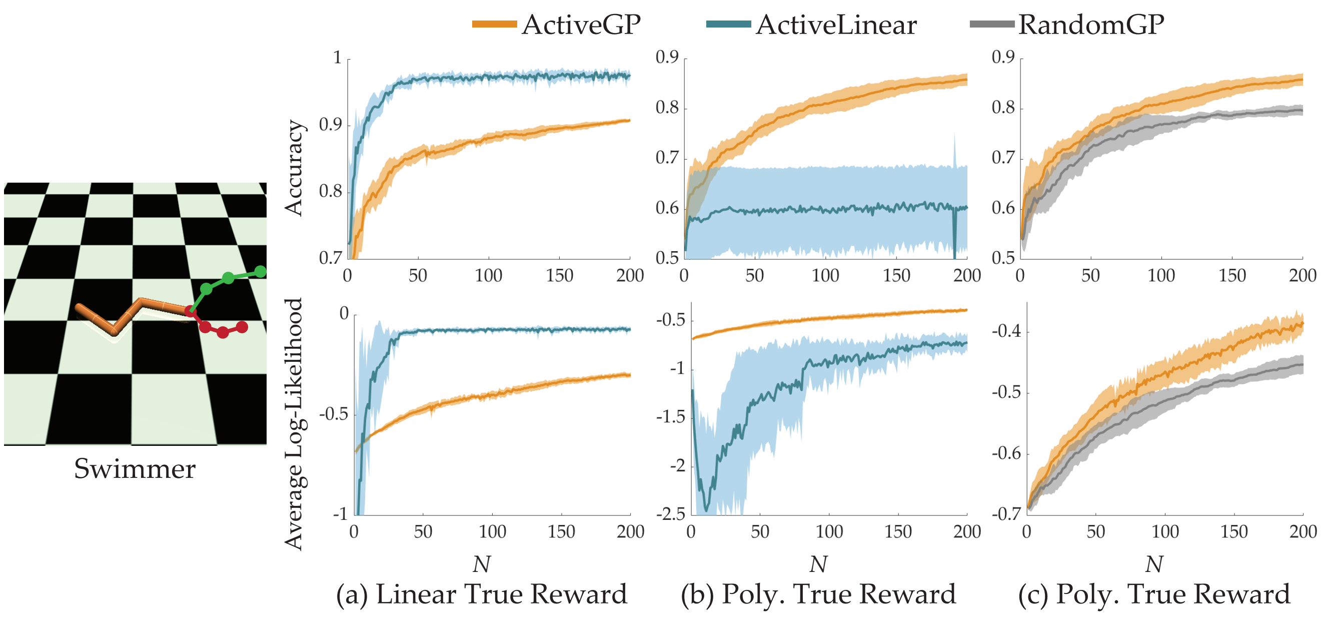

Figure 6. Accuracies and average log-likelihoods for test set queries are shown for the *Swimmer* environment (mean±std over 5 runs). **(a)** Expressiveness results when the true underlying reward function is linear. **(b)** Expressiveness results when the true underlying reward function is a degree-of-two 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.

it is relatively easy to find the trajectory that maximizes the reward.

On the other hand, we observe some significant differences in the *Driver* environment. First, SEMIACTIVEGP led to the best initial reduction in regret – even better than its unconstrained version ACTIVEGP. The reason for this is easy to see: 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, SEMIACTIVEGP constrains one of the trajectories in the query to be the best trajectory with respect to the posterior, thereby forcing the 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

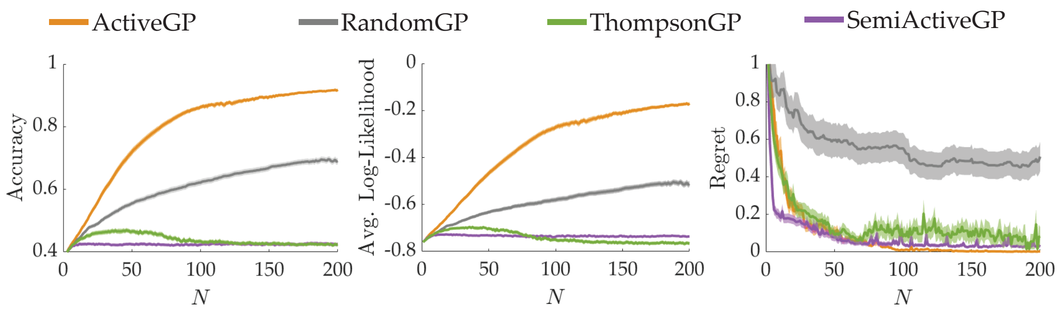

Figure 7. Accuracies, average log-likelihoods for test set queries, and regret values are shown for the *Driver* environment (mean±std over 50 runs). SEMIACTIVEGP and THOMPSONGP are bad at reward learning but competitive at reward optimization. SEMIACTIVEGP leads to the lowest regret values with very limited data, but ACTIVEGP achieves the lowest regret when the enough user responses are actively collected. Interestingly, THOMPSONGP performs worse than both ACTIVEGP and SEMIACTIVEGP with enough data.

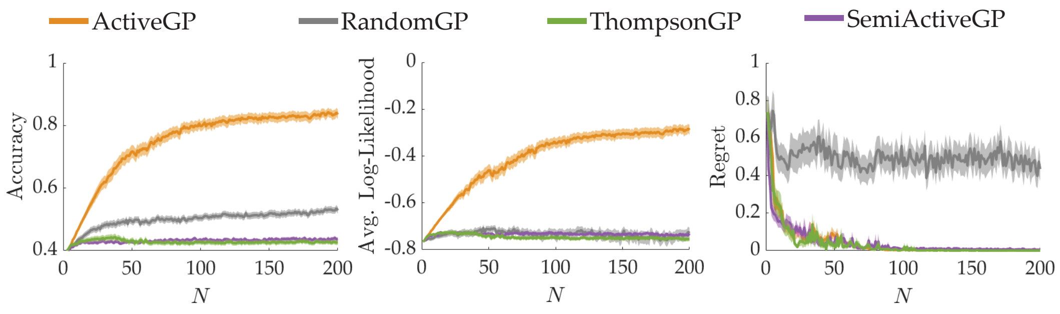

Figure 8. Accuracies, average log-likelihoods for test set queries, and regret values are shown for the *Tosser* environment (mean±std over 50 runs). SEMIACTIVEGP and THOMPSONGP are bad at reward learning but competitive at reward optimization, where all methods other than RANDOMGP performs comparably.

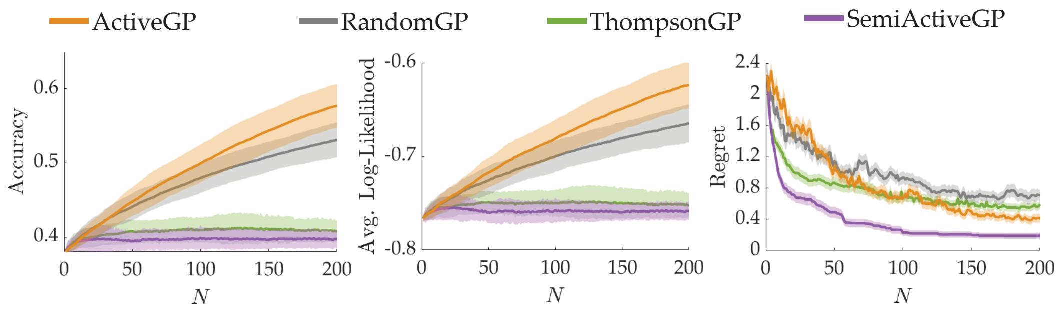

Figure 9. Accuracies, average log-likelihoods for test set queries, and regret values are shown for the *Swimmer* environment (mean±std in the first two plots and mean±s.e. in the third plot over 50 runs). SEMIACTIVEGP and THOMPSONGP are bad at reward learning. However, SEMIACTIVEGP clearly outperforms all others in reward optimization where THOMPSONGP also starts as a competitive method but incurs higher regret as the number of queries increases.

(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 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.

Table 1. Query Generation Times (seconds)

	Driver	**Tosser**	**Swimmer**
ActiveGP	83.4 ± 35.7	83.9 ± 35.3	81.7 ± 35.1
RandomGP	$(6.9 \pm 1.7) \times 10^{-5}$	$(6.9 \pm 1.9) \times 10^{-5}$	$(6.4 \pm 1.6) \times 10^{-5}$
ThompsonGP	$(5.0 \pm 1.0) \times 10^{-3}$	$(4.6 \pm 0.8) \times 10^{-3}$	$(0.7 \pm 2.0) \times 10^{-2}$
SemiActiveGP	0.12 ± 0.03	0.12 ± 0.03	0.12 ± 0.03

Reported query generation times are mean±std over 200 queries, and are based on our approach that uses a pre-computed set of 1000 trajectories.

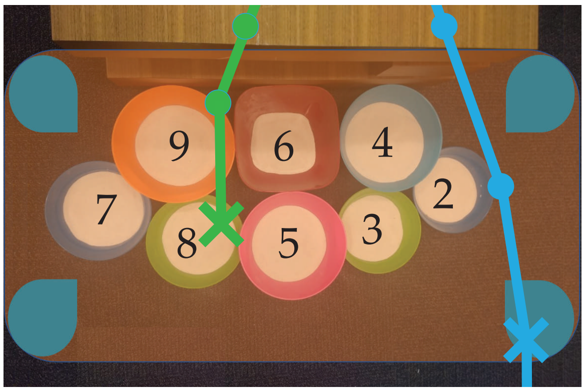

Figure 10. Top view of the eight targets in the variant of mini-golf user study. The users assign distinct scores from 2 to 9 to the targets. The figure shows an example of this ranking. While the robot is capable of hitting the ball into the entire shaded region, the maximizers of a linear reward always lie near the corners of the shaded region in blue. Therefore, while the GP reward model can query the user with better trajectories (e.g. the green trajectory), the linear model only explores the boundaries (e.g. the blue trajectory that throws the ball outside of this region). Crosses show where the ball hits the ground.

User Studies

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 Figure 1 and Figure 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.

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 5 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.

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 3 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, 9-very good).

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 $p < 0.05$ (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 3 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 $p < 0.05$. 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.

Conclusion

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

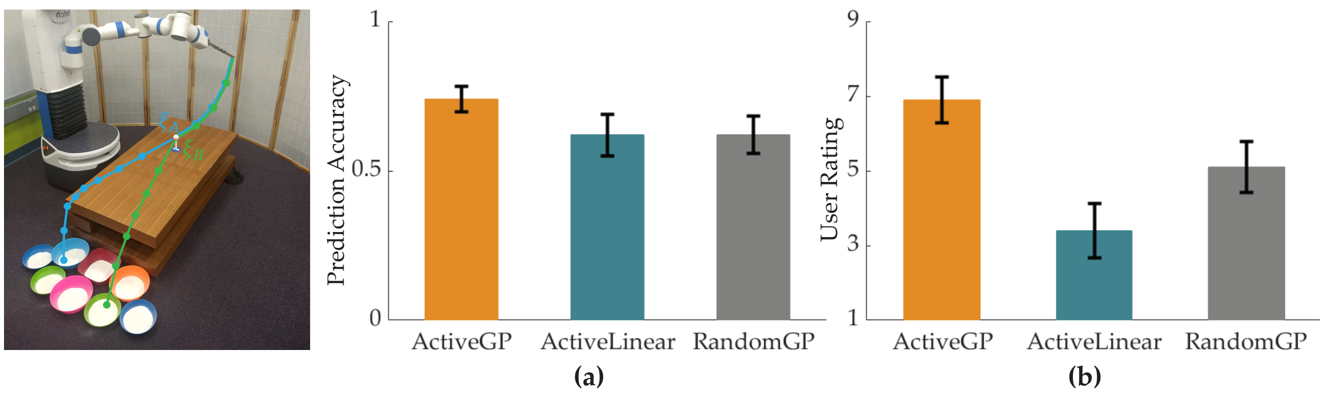

Figure 11. (a) Prediction accuracy results (mean±se). Each trained with 15 queries, ACTIVEGP achieves significantly higher prediction accuracy than both ACTIVELINEAR and RANDOMGP ($p < 0.05$). **(b)** User ratings on the final robot performance (mean±se). ACTIVEGP accomplishes the task significantly better than both ACTIVELINEAR and RANDOMGP ($p < 0.05$).

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.

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, e.g., to the mean function, or using scalable GPs that are beyond the scope of this work (Liu et al. 2020; Hensman et al. 2013). 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 Ξ. Further research is warranted to simultaneously learn both the reward function f and the feature function Ψ. 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. We acknowledge funding by NSF grants #1941722 and #2218760.

References

Abbeel P and Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: *International Conference on Machine Learning (ICML)*. pp. 1–8.

Abbeel P and Ng AY (2005) Exploration and apprenticeship learning in reinforcement learning. In: *International Conference on Machine Learning (ICML)*. pp. 1–8.

Akgun B, Cakmak M, Jiang K and Thomaz 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 learning-based reinforcement learning. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. pp. 116–131.

Bajcsy A, Losey DP, O'Malley MK and Dragan AD (2017) Learning robot objectives from physical human interaction. *Proceedings of Machine Learning Research* 78: 217–226.

Bajcsy A, Losey DP, O'Malley MK and Dragan AD (2018) Learning from physical human corrections, one feature at a time. In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. pp. 141–149.

Basu C, Biyik E, He Z, Singhal M and Sadigh D (2019) Active learning of reward dynamics from hierarchical queries. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Basu C, Yang Q, Hungerman D, Sinahal M and Dragan AD (2017) Do you want your autonomous car to drive like you? In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. pp. 417–425.

Biyik E (2022) *Learning Preferences For Interactive Autonomy*. PhD Thesis, Electrical Engineering Department, Stanford University.

Biyik E, Huynh N, Kochenderfer MJ and Sadigh D (2020) Active preference-based gaussian process regression for reward learning. In: *Proceedings of Robotics: Science and Systems (RSS)*. DOI:10.15607/rss.2020.xvi.041.

Biyik E, Lazar DA, Sadigh D and Pedarsani R (2019a) The green choice: Learning and influencing human decisions on shared roads. In: *IEEE Conference on Decision and Control (CDC)*.

Biyik E, Losey DP, Palan M, Landolfi NC, Shevchuk G and Sadigh D (2021) Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research (IJRR)* DOI:10.1177/02783649211041652.

Biyik E, Palan M, Landolfi NC, Losey DP and Sadigh D (2019b) Asking easy questions: A user-friendly approach to active reward learning. In: *Conference on Robot Learning (CoRL)*.

Biyik E and Sadigh D (2018) Batch active preference-based learning of reward functions. In: *Conference on Robot Learning (CoRL)*.

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)*. DOI:10.1109/hri53351.2022.9889650.

Biyik E, Wang K, Anari N and Sadigh D (2019c) Batch active learning using determinantal point processes. *arXiv preprint arXiv:1906.07975* .

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

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J and Zaremba W (2016) Openai gym. *arXiv preprint arXiv:1606.01540* .

Brown DS and Niekum S (2019) Deep bayesian reward learning from preferences. In: *NeurIPS Workshop on Safety and Robustness in Decision Making*.

Cakmak M, Srinivasa SS, Lee MK, Forlizzi J and Kiesler S (2011) Human preferences for robot-human hand-over configurations. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. pp. 1986–1993.

Casper S, Davies X, Shi C, Gilbert TK, Scheurer J, Rando J, Freedman R, Korbak T, Lindner D, Freire P, Wang T, Marks S, Segerie CR, Carroll M, Peng A, Christoffersen P, Damani M, Slocum S, Anwar U, Siththaranjan A, Nadeau M, Michaud EJ, Pfau J, Krasheninnikov D, Chen X, Langosco L, Hase P, Biyik E, Dragan A, Krueger D, Sadigh D and Hadfield-Menell D (2023) Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217* .

Christiano PF, Leike J, Brown T, Martic M, Legg S and Amodei D (2017) Deep reinforcement learning from human preferences. In: *Advances in Neural Information Processing Systems (NeurIPS)*. pp. 4299–4307.

Chu W and Ghahramani Z (2005) Preference learning with gaussian processes. In: *International Conference on Machine Learning (ICML)*. pp. 137–144.

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

Daniel C, Kroemer O, Viering M, Metz J and Peters J (2015) Active reward learning with a novel acquisition function. *Autonomous Robots* 39(3): 389–405.

Dragan AD and Srinivasa SS (2012) Formalizing assistive teleoperation. In: *Robotics: Science and Systems*.

González J, Dai Z, Damianou A and Lawrence ND (2017) Preferential bayesian optimization. In: *International Conference on Machine Learning (ICML)*. pp. 1282–1291.

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

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

Houlsby N, Huszar F, Ghahramani Z and Hernández-Lobato JM (2012) Collaborative gaussian processes for preference learning. In: *Advances in Neural Information Processing Systems (NeurIPS)*. pp. 2096–2104.

Houlsby N, Huszár F, Ghahramani Z and Lengyel M (2011) Bayesian active learning for classification and preference learning. In: *NIPS Workshop on Bayesian optimization, experimental design and bandits: Theory and applications*.

Ibarz B, Leike J, Pohlen T, Irving G, Legg S and Amodei D (2018) Reward learning from human preferences and demonstrations in atari. In: *Advances in Neural Information Processing Systems (NeurIPS)*. pp. 8011–8023.

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

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

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 and Darrell T (2007) Active learning with gaussian processes for object categorization. In: *International Conference on Computer Vision (ICCV)*. pp. 1–8.

Katz SM, Bihan ACL and Kochenderfer MJ (2019) Learning an urban air mobility encounter model from expert preferences. In: *Digital Avionics Systems Conference (DASC)*.

Katz SM, Maleki A, Bıyık E and Kochenderfer MJ (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: Volume 1* : 161–176.

Lee K, Smith L and Abbeel P (2021) Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In: *International Conference on Machine Learning*.

Lepird 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 and Cai J (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, Garg A and Sadigh D (2020) Controlling assistive robots with learned latent actions. In: *International Conference on Robotics and Automation (ICRA)*.

Myers V, Biyik E, Anari N and Sadigh D (2022) Learning multimodal rewards from rankings. In: *Conference on Robot Learning*. PMLR, pp. 342–352.

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

Ng AY, Harada D and Russell S (1999) Policy invariance under reward transformations: Theory and application to reward shaping. In: *International Conference on Machine Learning*

(ICML), volume 99. pp. 278–287.

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

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, Almeida D, Wainwright C, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A 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 and Sadigh D (2019) Learning reward functions by integrating human demonstrations and preferences. In: *Robotics: Science and Systems*.

Racca M, Oulasvirta A and Kyrki V (2019) Teacher-aware active robot learning. In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. pp. 335–343.

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

Rasmussen CE and Williams CK (2005) *Gaussian processes for machine learning*. MIT Press.

Ross S, Melik-Barkhudarov N, Shankar KS, Wendel A, Dey D, Bagnell JA and Hebert M (2013) Learning monocular reactive uav control in cluttered natural environments. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 1765–1772.

Sadigh D, Dragan AD, Sastry SS and Seshia SA (2017) Active preference-based learning of reward functions. In: *Robotics: Science and Systems*.

Sadigh D, Sastry S, Seshia SA and Dragan AD (2016) Planning for autonomous cars that leverage effects on human actions. In: *Robotics: Science and Systems*.

Sharma P, Sundaralingam B, Blukis V, Paxton C, Hermans T, Torralba A, Andreas J and Fox D (2022) Correcting robot plans with natural language feedback. *arXiv preprint arXiv:2204.05186* .

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

Sontakke SA, Arnold S, Zhang J, Pertsch K, Biyik E, Sadigh D, Finn C and Itti L (2023) Roboclip: One demonstration is enough to learn robot policies. In: *Advances in Neural Information Processing Systems (NeurIPS)*.

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

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)*. pp. 5026–5033.

Tucker M, Cheng M, Novoseller E, Cheng R, Yue Y, Burdick JW and Ames AD (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)*. IEEE, pp. 3423–3430.

Tucker M, Novoseller E, Kann C, Sui Y, Yue Y, Burdick J and Ames AD (2020b) Preference-based learning for exoskeleton gait optimization. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Wilde N, Biyik E, Sadigh D and Smith SL (2021) Learning reward functions from scale feedback. In: *5th Annual Conference on Robot Learning*.

Wilde N, Kulić 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)*. IEEE, pp. 10952–10959.

Wirth C, Akrour R, Neumann G, Fürnkranz J 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, Diehr E and Dymesich D (2016) Fetch and freight: Standard platforms for service robot applications. In: *Workshop on Autonomous Mobile Service Robots*.

Ziebart BD, Maas AL, Bagnell JA and Dey AK (2008) Maximum entropy inverse reinforcement learning. In: *AAAI Conference on Artificial Intelligence (AAAI)*, volume 8. pp. 1433–1438.

Appendix

Active Query Generation Derivation

Let Σ 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:

$$\begin{aligned}
H(q\,|\Psi,\boldsymbol{\Psi},\boldsymbol{q}) &= h\left(\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)\mathcal{N}([f^{(1)},f^{(2)}]\mid[\mu^{(1)},\mu^{(2)}],\Sigma)df^{(2)}df^{(1)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)})^2+c'(f^{(2)}-\mu^{(2)})^2+2d(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))}df^{(2)}df^{(1)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c((f^{(1)}-\mu^{(1)})^2+\frac{2d}{c}(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))+c'(f^{(2)}-\mu^{(2)})^2)}df^{(1)}df^{(2)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2-\frac{d^2}{c}(f^{(2)}-\mu^{(2)})^2+c'(f^{(2)}-\mu^{(2)})^2)}df^{(1)}df^{(2)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int e^{-\frac{1}{2}c'(f^{(2)}-\mu^{(2)})^2}e^{\frac{1}{2}\frac{d^2}{c}(f^{(2)}-\mu^{(2)})^2}\int \Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2)}df^{(1)}df^{(2)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\int \frac{\Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2)}}{\frac{\sqrt{2\pi}}{\sqrt{c}}}\frac{\sqrt{2\pi}}{\sqrt{c}}df^{(1)}df^{(2)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\int \frac{\Phi\left(\frac{f^{(1)}-f^{(2)}}{\sqrt{2}\sigma}\right)e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)}+\frac{d}{c}(f^{(2)}-\mu^{(2)}))^2)}}{\frac{\sqrt{2\pi}}{\sqrt{c}}}df^{(1)}df^{(2)}\right)
\end{aligned}$$

Using the mathematical identity $\int_x \phi(x)N(x|\mu,\sigma^2)dx = \phi(\frac{\mu}{\sqrt{1+\sigma^2}})$, we obtain

$$\begin{aligned}
H(q\mid\Psi,\boldsymbol{\Psi},\boldsymbol{q}) &= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\Phi\left(\frac{\mu^{(1)}-\frac{d}{c}f^{(2)}+\frac{d}{c}\mu^{(2)}-f^{(2)}}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}}\right)df^{(2)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\Phi\left(\frac{\mu^{(1)}+\frac{d}{c}\mu^{(2)}-(\frac{d}{c}+1)f^{(2)}}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}}\right)df^{(2)}\right)\\
&= h\left(\frac{\sqrt{cc'-d^2}}{\sqrt{2\pi}\sqrt{c}}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}(f^{(2)}-\mu^{(2)})^2}\frac{\Phi\left(\frac{-(\frac{d}{c}+1)(f^{(2)}-\frac{\mu^{(1)}+\frac{d}{c}\mu^{(2)}}{\frac{d}{c}+1})}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}}\right)}{\frac{\sqrt{2\pi c}}{\sqrt{c'c-d^2}}}\frac{\sqrt{2\pi c}}{\sqrt{c'c-d^2}}df^{(2)}\right)
\end{aligned}$$

Using the same identity again,

$$H(q\mid\Psi,\boldsymbol{\Psi},\boldsymbol{q}) = h\left(\Phi\left(\frac{\mu^{(1)}-\mu^{(2)}}{\sqrt{2}\sigma\sqrt{1+\frac{1}{2c\sigma^2}}\sqrt{1+\frac{c}{2\sigma^2+\frac{1}{c}}\frac{(1+\frac{d}{c})^2}{c'c-d^2}}}\right)\right)$$

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

$$H(q\mid\Psi,\boldsymbol{\Psi},\boldsymbol{q}) = h\left(\Phi\left(\frac{\mu^{(1)}-\mu^{(2)}}{\sqrt{2\sigma^2+g(\Psi^{(1)},\Psi^{(2)})}}\right)\right).$$

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

We next make the derivation for the second entropy term. To simplify the notation, we let $\sigma'^2 = \frac{\pi ln(2)}{2}$, $\sigma''^2 = \sigma'^2 + \frac{1}{c}$, and $\sigma_b^2 = \frac{c(1+\frac{d}{c})^2}{c'c-d^2}$. By performing a linearization over the logarithm of the second entropy term as in Houlsby et al. (2011),

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}[H(q\,|\Psi,f)] &\approx \frac{\sqrt{c'c-d^2}}{2\pi}\int\int e^{-\frac{(f^{(1)}-f^{(2)})^2}{\pi ln(2)}} e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)})^2+c'(f^{(2)}-\mu^{(2)})^2+2d(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))}df^{(1)}df^{(2)}\\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'(f^{(2)}-\mu^{(2)})^2}\int e^{-\frac{(f^{(1)}-f^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}(c(f^{(1)}-\mu^{(1)})^2+2d(f^{(1)}-\mu^{(1)})(f^{(2)}-\mu^{(2)}))}df^{(1)}df^{(2)}\\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'f^{(2)2}}\int e^{-\frac{(f^{(1)}+\mu^{(1)}-f^{(2)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cf^{(1)2}-df^{(1)}f^{(2)}}df^{(1)}df^{(2)}\\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}c'f^{(2)2}}\int e^{-\frac{(f^{(1)}+\mu^{(1)}-f^{(2)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}c(f^{(1)}+\frac{d}{c}f^{(2)})^2+\frac{1}{2}\frac{d^2}{c}f^{(2)2}}df^{(1)}df^{(2)}\\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}f^{(2)2}}\int e^{-\frac{(f^{(1)}-f^{(2)}+\mu^{(1)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}c(f^{(1)}+\frac{d}{c}f^{(2)})^2}df^{(1)}df^{(2)}
\end{aligned}$$

By the change of variables for the inner integral with $u = f^{(1)} + \frac{d}{c}f^{(2)}$,

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q\mid\Psi,f)\right] &= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}f^{(2)2}}\int_u e^{-\frac{(u-\frac{d}{c}f^{(2)}+\mu^{(1)}-f^{(2)}-\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cu^2}dudf^{(2)}\\
&= \frac{\sqrt{c'c-d^2}}{2\pi}\int e^{-\frac{1}{2}\frac{c'c-d^2}{c}f^{(2)2}}\int_u e^{-\frac{((1+\frac{d}{c})f^{(2)}-u-\mu^{(1)}+\mu^{(2)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cu^2}dudf^{(2)}
\end{aligned}$$

By another change of variables for the outer integral with $v = \frac{f^{(2)}}{1+d/c}$,

$$\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q\mid\Psi,f)\right] = \frac{1}{1+\frac{d}{c}}\frac{\sqrt{c'c-d^2}}{2\pi}\int_v e^{-\frac{1}{2}\frac{c'c-d^2}{c}\frac{v^2}{(1+\frac{d}{c})^2}}\int_u e^{-\frac{(v-u+\mu^{(2)}-\mu^{(1)})^2}{2\sigma'^2}} e^{-\frac{1}{2}cu^2}dudv$$

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

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q\mid\Psi,f)\right] &= \frac{1}{1+\frac{d}{c}}\frac{\sqrt{c'c-d^2}}{2\pi}2\pi\sigma'\frac{1}{\sqrt{c}}\int_v e^{-\frac{1}{2}\frac{c'c-d^2}{c}\frac{v^2}{(1+\frac{d}{c})^2}}\frac{1}{\sqrt{2\pi}\sqrt{\sigma'^2+\frac{1}{c}}}e^{-\frac{1}{2}\frac{(v-(\mu^{(1)}-\mu^{(2)}))^2}{\sigma'^2+\frac{1}{c}}}dv\\
&= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma'\frac{1}{\sqrt{c}}\frac{1}{\sqrt{2\pi}\sqrt{\sigma'^2+\frac{1}{c}}}\int_v e^{-\frac{1}{2}\frac{v^2}{\sigma_b^2}}e^{-\frac{1}{2}\frac{(v-(\mu^{(1)}-\mu^{(2)}))^2}{\sigma''^2}}dv
\end{aligned}$$

By repeating the same convolution trick for the second integral,

$$\begin{aligned}
\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q\mid\Psi,f)\right] &= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma'\frac{1}{\sqrt{c}}\frac{1}{\sqrt{2\pi}\sqrt{\sigma'^2+\frac{1}{c}}}2\pi\sigma_b\sigma''\frac{1}{\sqrt{2\pi}\sqrt{\sigma_b^2+\sigma''^2}}e^{-\frac{1}{2}\frac{(\mu^{(1)}-\mu^{(2)})^2}{\sigma_b^2+\sigma''^2}}\\
&= \frac{1}{1+\frac{d}{c}}\sqrt{c'c-d^2}\sigma'\frac{1}{\sqrt{c}}\frac{1}{\sqrt{\sigma'^2+\frac{1}{c}}}\sigma_b\sigma''\frac{1}{\sqrt{\sigma_b^2+\sigma''^2}}e^{-\frac{1}{2}\frac{(\mu^{(1)}-\mu^{(2)})^2}{\sigma_b^2+\sigma''^2}}
\end{aligned}$$

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

$$\mathbb{E}_{f\sim P(f|\mathbf{\Psi},\boldsymbol{q})}\left[H(q\mid\Psi,f)\right] = \frac{\sqrt{\pi\ln(2)\sigma^2}\exp\left(-\frac{(\mu_a-\mu_b)^2}{\pi\ln(2)\sigma^2+2g(\Psi^{(1)},\Psi^{(2)})}\right)}{\sqrt{\pi\ln(2)\sigma^2+2g(\Psi^{(1)},\Psi^{(2)})}}$$