Exploring Diverse Interaction Types for Human-in-the-Loop Robot Learning

Patrick Callaghan

CMU-RI-TR-23-81

November 21, 2023

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Henny Admoni, *Co-Chair*
Oliver Kroemer, *Co-Chair*
Reid Simmons
Gokul Swamy

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

For my parents.

Abstract

Teaching sessions between humans and robots will need to be maximally informative for optimal robot learning and to ease the human's teaching burden. However, the bulk of prior work considers one or two modalities through which a human can convey information to a robot—namely, kinesthetic demonstrations and preference queries. Moreover, people will teach robots to perform tasks according to their own, individual preferences; as such, robots need to represent the task in a way that can handle this heterogeneity. This thesis addresses both needs. First, we investigated how an agent can maximize its information gain by actively selecting queries from a diverse set of interaction types (including demonstrations, corrections, preference queries, and binary critiques). Second, we explored three reward function structures that could be used to model a human teacher's preferences for how an agent should perform a task. Our evaluations showed that 1.) actively selecting from among a diverse set of interaction types yields faster, more robust learning, and 2.) an agent typically learns best when its reward function structure matches its teacher's.

Acknowledgments

Without the guidance of Henny Admoni and Oliver Kroemer, I'd be in a much different place today. Thank you both so much for your unfailing kindness, patience, and interest in me as a person and as a researcher. Advisors have many roles, and I routinely feel the efforts you make to fill them all. That effort is what makes all the difference.

Thanks too to my committee members Reid Simmons and Gokul Swamy. Gokul, your wise advice on how to approach my research will stay with me during and beyond my PhD; I'm so grateful for our discussion and look forward to the discussions we'll have in the future. Reid, we've only worked together for a short while, but I already feel excited about what's to come. Thank you for your guidance so far and your belief in me.

The intellectual discussions and the fun experiences are what made my time as an MSR student as enjoyable as it ended up being, and those words—"intellectual" and "fun"—are apt descriptors for the the people I can so happily call my labmates. Thank you IAM Lab (Alex, Erin, Jacky, Kevin, Mark, Mohit, Sarvesh, Saumya, Shivam, Tab, Xinyu, Yunus, and Zilin), and HARP Lab (Abhijat, Ada, Ben, Maggie, Michelle, Mike, Pallavi, Pranay, Reuben, Stephanie, Suresh, Tesca, and Zulekha) for being the regular sources of wisdom, kindness, and laughter I imagine most hope to have in life. Additionally, imposter syndrome has had its way with me, but so many of you helped me quiet that voice and believe I am capable as a researcher. Thank you for conversations which still help me keep things in perspective.

I would be nowhere without the remarkable collection of people I can so fortunately call my family. Mom, Dad, Katie, Dan, Thomas, Luke, Maggie, and Aria—I am so enormously grateful that I never need doubt the love you have for me; it is a gift I do not take for granted. I love you all.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

Chapter 1

Introduction

As we envision robots that adapt to novel tasks and environments after deployment, it is important to consider *how* they can efficiently obtain training data to address this novelty. Research in Human Interactive Robot Learning (HIRL) has yielded many effective methods for obtaining this training data via interaction between a robot and a human teacher. For instance, in a *demonstration*, the teacher provides a trajectory that the robot should follow starting from an initial state [26, 41]. In a *preference query*, the teacher selects one trajectory from a set of candidates proposed by the robot [10, 51]. In response to a single trajectory proposed by the robot, the teacher can provide a *correction* [1, 21] or simply a *binary reward* [14]. These interaction types differ according to how the robot queries the teacher, how the teacher is constrained in providing feedback, how the robot should interpret the teacher's feedback as training data, and the physical and cognitive load imposed on the teacher [18].

Prior work in Active Learning has investigated how to formulate informative queries by maximizing the expected information gain resulting from the teacher's feedback. However, barring a few exceptions ([9, 12, 43]), prior work typically assumes that the robot uses a single interaction type for all queries. We expect that the *optimal* interaction type depends on the robot's task knowledge (which changes over time), the robot's query state (i.e., the state from which it queries the teacher), and domain-specific considerations (e.g., the time or effort it takes a teacher to respond to queries) [30]. Additionally, we suspect that the way a learning agent models its teacher's preferences can influence how it learns tasks when taught through

actively-selected interaction types and can even affect the active selection process itself.

The first work of this thesis is motivated by this question: How can a robot optimize both the *type* and *content* of its queries to a human teacher based on the information it needs at any given moment? We introduce INQUIRE: a robot learning system that performs this optimization by representing multiple interaction types in a single unified framework, enabling the robot to directly estimate and compare the expected information gain of its queries across multiple interaction types. We evaluate INQUIRE against two state-of-the-art interactive learning methods that use a single or fixed pattern of interaction types. We analyzed the effect of domain on INQUIRE's performance and selection of interaction types over time by simulating four domains with unique reward-learning problems. We found that INQUIRE learned reward functions that were more accurate and resulted in better task performance than either baseline, with particular strength in accommodating low-information query states (i.e., repeated states in which the robot has already received feedback). Furthermore, we demonstrate how INQUIRE can incorporate cost metrics (representing physical or cognitive load on the teacher), optimizing queries over both the informativeness and ease of the teacher's responses.

In the second work of this thesis, we explore how the choice of reward function *representation* affects the agent's ability to learn through multiple interaction types. In particular, we explore three variants of a standard linear combination reward representation with varied representational power and hence varied complexity. A Comparative reward representation weighs the relative importances of task features. Such a representation enables intuitive interpretation and is commonplace in the literature [2, 6, 41, 45]. In contrast, our Comparative+Specific representation provides superior representational power—that is, an ability to specify the desired parameters of a reward function—at the cost of learning both the comparative weights and specific, desired parameter values instead of only learning weights. Finally, a Grid representation is an attempt to capture the "middle ground" between the two aforementioned representations by assigning comparative weights to a collection of predefined candidate parameter values. As in our aforementioned work, we consider corrections, critiques, demonstrations, and preference queries. We design the learner using a particle-based belief model that is amenable to the different reward repre-

sentations and active query selection from among multiple interaction types. We also investigate how different reward representations affect an agent's ability to learn when querying with a single interaction type in comparison to when interaction types are actively selected by the agent. Through teaching sessions with a simulated human oracle, we found that (1.) reward representation affects the frequency with which an agent actively selects each interaction types, and (2.) an agent learns best when its reward representation matches that of its teacher's.

The two works presented herein aim to contribute to the grand goal of enabling robots to efficiently and conveniently learn tasks and behaviors directly from human teachers and in accordance with those teachers' preferences. Much work has yet to be done to achieve such a vision, but hopefully this thesis can be recalled and built upon in future efforts to understand how robots should learn from, and behave around, long-term human partners.

Chapter 2

Related Work

2.1 Reward Representations in HIRL

Prior work explores a plethora of reward representations for modeling humans and their preferences. As is standard in many inverse reinforcement learning contexts, much of that work assumes a reward function that is a linear combination of weights and features [3, 4, 48]. In [11], the authors attempt to learn a user's reward function through preference queries and by modeling the function with a Gaussian Process, and the work from [40] models reward for a task using a neural network amenable to inputs from multiple interaction types.

The work in [32] models human ranking tendencies via the Boltzmann rational model, but they model a prior over human policies instead of reward functions (as we do in this thesis). Moreover, the authors' model is a deep network (which includes a transformer as a discriminator).

While the aforementioned works each explore reward representations implicitly, they do not do so in an explicit manner as we do in our second project. At the same time, our work does not explore the possible reward representations discussed herein; future work which makes such comparisons would be worthwhile and informative for deducing which reward representations are best for HIRL.

2.2 Approximating Human Models

Accurately modeling—and *learning how* to accurately model—the humans with which robots interact remain crucial problems in HRI researchers still strive to solve. The method presented in [46] learns skills from a human demonstrator while also learning a model describing how that particular human is sub-optimal when providing those demonstrations. Like us, the aim of [34] is to learn a human model, but we seek a model of human preferences over tasks while they sought a model of what a human learner thought of their AI teacher. Similarly, the agent in [53] learns a model of strategies its human partners might employ when undertaking collaborative tasks. Investigations into the functional underpinnings of learning from multiple interaction types and how they might vary from person to person also have been made [23]. Our underlying representation of the human teacher is rather simple, but as opposed to the aforementioned work, our work explores what might be an ideal way to model teachers' task representations.

2.3 Active Learning and Learning from People

Learning from People typically refers to how robots interpret human-provided feedback as training data. This interpretation arises in the forms of imitation learning [26] or inverse reinforcement learning from demonstrations [41]. However, there are many other forms that human feedback can take, including preferences [10, 51], labels [20], and corrections [1, 21]. These approaches optimize queries *within* a single interaction type, sometimes by maximizing volume removal [45], information gain derived from the teacher's response to the query [10], or by min-max regret optimization [50]. Prior work also investigates the use of fixed strategies for selecting interaction types; for example, requesting a fixed number of demonstrations before requesting preferences for the remaining queries [27, 43]. As we do, [9] incorporates more interaction types (demonstrations, labels, and feature queries) and contributes both rule-based and decision-theoretic strategies for query selection. Other methods for learning from multiple feedback types include combining preferences with ordinal labels [35] and using both demonstrations and rankings [8]. Furthermore, [13] presents a software library for combining different preference feedback types and demonstrations.

Additionally, prior work investigates active selection of queries for a robot learner to pose their human teachers [10, 45]. In particular, the authors of [5] incorporate an active learning variant and learning from multiple interaction types to acquire knowledge of spatial understanding for use in manipulation tasks. Other work strives to unify the different interaction types that a robot could expect from a human [19, 28]. Insofar as reward function learning is concerned, progress has been made by learning from physical interactions between humans and robots—via corrections in particular [1, 38, 39]. The methods in [15, 16, 29] learn from demonstrations of trajectories that are sub-optimal. Like our work, the authors consider more than one interaction type, yet they consider only two: demonstrations and "negative" demonstrations (i.e., demonstrations of what not to do). Other work exhibits an ability to learn and execute new manipulation skills after only a single kinesthetic demonstration [49]. Finally, much prior work pertains to active learning of manipulation skills through interaction with the environment [31, 37].

Chapter 3

Actively Selecting Queries From a Diverse Set of Interaction Types

Research in Human Interactive Robot Learning (HIRL) has yielded many effective methods for obtaining training data via interaction between a robot and a human teacher. In a *demonstration*, the teacher provides the trajectory that the robot should take starting from a particular state [26, 41]. In a *preference* query, the teacher selects one trajectory from a set of candidates proposed by the robot [10, 51]. In response to a single trajectory proposed by the robot, the teacher can provide a *correction* [1, 21] or simply a *binary reward* [14].

These interaction types differ according to how the robot queries the teacher, how the teacher is constrained in providing feedback, how the robot should interpret the teacher's feedback as training data, and the physical and cognitive load imposed on the teacher [18].

Prior work in Active Learning has investigated how to formulate informative queries by maximizing the expected information gain resulting from the teacher's feedback. However, barring a few exceptions ([9, 12, 43]), prior work typically assumes that the robot uses a single interaction type for all queries. We expect that the *optimal* interaction type depends on the robot's task knowledge (which changes over time), the robot's query state (i.e., the state from which it queries the teacher), and domain-specific considerations (e.g., the time or effort it takes a teacher to respond to queries) [30].

This project is motivated by this question: How can a robot optimize both the *type* and *content* of its queries to a human teacher based on the information it needs at any given moment? We introduce INQUIRE: a robot learning system that performs this optimization by representing multiple interaction types in a single unified framework, enabling the robot to directly estimate and compare the expected information gain of its queries across multiple interaction types [1]. We evaluated INQUIRE against two state-of-the-art interactive learning methods that use a single or fixed pattern of interaction types. We analyzed the effect of domain on INQUIRE's performance and selection of interaction types over time by simulating four domains with unique reward-learning problems. We found that INQUIRE learned reward functions that were more accurate and resulted in better task performance than either baseline, with particular strength in accommodating low-information query states (i.e., repeated states in which the robot has already received feedback). Furthermore, we demonstrate how INQUIRE can incorporate cost metrics (representing physical or cognitive load on the teacher), optimizing queries over both the informativeness and ease of the teacher's responses.

3.1 Approach

We define a **query** as a set of possible choices presented to the teacher, and **feedback** as the teacher's selected choice in response to a query. Our goal is to enable a robot to (1) efficiently query a teacher using multiple interaction types, and (2) learn from feedback obtained via these interactions. We ground this goal in the problem of learning a distribution $\mathcal{W}$ over feature weight vectors $\omega \in \mathcal{W}$, each resulting in a linear reward function $r(t) = \phi(t) \cdot \omega$, where $\phi(t)$ is the feature vector of a trajectory t. Thus, our goal translates into (1) selecting queries and interaction types that minimize uncertainty over $\mathcal{W}$, and (2) updating $\mathcal{W}$ over feedback from multiple interaction types.

We present INQUIRE (Alg. 1), an algorithm comprised of three key steps for each query: (1) selecting the optimal interaction type i and corresponding query q_i^* that maximizes the information gain over the weight distribution $\mathcal{W}$ (approximated as the

[1] This work was conducted in collaboration [22].

sample set Ω), (2) recording the teacher's response to that query (i.e., feedback) in a feedback set $\mathbf{F}$, and (3) updating the weight distribution $\mathcal{W}$ such that it maximizes the likelihood of all feedback in $\mathbf{F}$. To generalize across multiple interaction types, we must contend with the differing formulations of *query* and *feedback* corresponding to each type. We follow the framing presented in [18], where each interaction type consists of a *query space* $Q(s)$ (the set of possible queries from state s) and a *choice space* $C(q)$ (the set of possible teacher feedback, i.e., the choices available to the teacher in response to a query $q \in Q(s)$). We assume the robot must query from whatever initial state s it is placed in, and cannot optimize the state s itself.

For a **demonstration**, let $\mathcal{T}(s)$ represent the set of all possible trajectories originating from the initial state s. The robot (implicitly) enables the teacher to demonstrate any trajectory in this set, and thus its query space is $Q(s) = \{\mathcal{T}(s)\}$ (i.e., a single query consisting of the entire trajectory space). The teacher's choice space is $C = \mathcal{T}(s)$ (any trajectory within that space). For a **preference**, the robot queries the teacher with two trajectories $q = \{t_0, t_1 \mid t_0, t_1 \in \mathcal{T}(s)\}$ who then chooses either t_0 or t_1. The query space is $Q(s) = \mathcal{T}(s)$ x $\mathcal{T}(s)$ and the teacher's choice space is $C(q) = \{t_0, t_1\}$. For a **correction**, the robot executes one trajectory $q \in \mathcal{T}(s)$ which the teacher then modifies to a preferable behavior. The agent's query space is $Q(s) = \mathcal{T}(s)$ and the teacher's choice space is $C(q) = \mathcal{T}(s)$. For **binary reward**, the robot executes a single trajectory $q \in \mathcal{T}(s)$, and the teacher indicates a positive or negative reward. The agent's query space is $Q(s) = \mathcal{T}(s)$ and the teacher's choice space is $C(q) = \{0, 1\}$.

The *implication* of the teacher's choice $c \in C(q)$ is a set of accepted trajectories c^+ and set of rejected trajectories c^-, which we define in Table 3.1 and use later to calculate information gain. Since the set of all possible trajectories originating from s (represented by $\mathcal{T}(s)$) is potentially infinite, we approximate it as the set T containing N trajectory samples originating from the state s and consisting of randomly selected actions.

3.1.1 Query Optimization

When optimizing the agent's query, our goal is to greedily select one that maximizes the agent's expected information gain over $\mathcal{W}$ after receiving any feedback from

Algorithm 1 INQUIRE - Overview

Input: Set of query states S
Parameters: K (# of queries), $\mathcal{I}$ (interaction types)
Output: Weight vector ω^*

$\mathbf{F} \leftarrow \{\}$
$\mathbf{\Omega} \leftarrow M$ random initial weight vectors
for K iterations **do**
  $s \leftarrow$ next query state in S
  $q_i^* \leftarrow$ generate_query$(s, \mathcal{I}, \mathbf{\Omega})$ **(Alg. 2)**
  $\mathbf{F} \leftarrow \mathbf{F} \cup \{\text{query\_teacher}(q_i^*)\}$
  $\mathbf{\Omega} \leftarrow$ update_weights$(\mathbf{F})$
end for
$\omega^* \leftarrow$ mean$(\mathbf{\Omega})$
return ω^*

Algorithm 2 INQUIRE - Generate Query

Input: s (state), $\mathcal{I}$ (interaction types), $\mathbf{\Omega}$ (weight samples)
Output: Query q^*

$\mathbf{T} \leftarrow$ uniformly_sample_trajectories(s)
Compute $\mathbf{E} : \{\mathbf{E}_{t,t',\omega}, \forall t, t' \in \mathbf{T}, \omega \in \mathbf{\Omega}\}$ **(Eq. 3.4)**
for each interaction type $i \in \mathcal{I}$ **do**
  $\mathbf{Q} \leftarrow Q_i(s)$ **(See Table 1)**
  $\mathbf{C} \leftarrow \{C_i(q), \forall q \in \mathbf{Q}\}$ **(See Table 1)**
  Compute info gain matrix $\mathbf{G^{(i)}}$ from $\mathbf{E}$ **(Eq. 3.9)**
  $q \leftarrow \text{argmax}_{q'} \sum_{c \in \mathbf{C}_{q'}, \omega \in \mathbf{\Omega}} \mathbf{G}^{(\mathrm{i})}_{q',c,\omega}$
  $g \leftarrow \frac{1}{\log(\lambda_i)} \sum_{c \in \mathbf{C}_q, \omega \in \mathbf{\Omega}} \mathbf{G}^{(\mathrm{i})}_{q,c,\omega}$
  if information gain $g > g^*$ **then**
    $g^* \leftarrow g$
    $q^* \leftarrow q$ ▷ store query with highest info. gain
  end if
end for
return q^*

Table 3.1: Each interaction involves separate query spaces, choice spaces, and choice implications.

	Query Space $Q_i(s)$	**Query** $q \in Q_i(s)$	**Choice Space** $C_i(q)$	**Choice Implication** $c \in C_i(q) \implies (c^+, c^-)$
Demo.	$\{T\}$	T	T	$c^+ : t \in T \quad c^- : T \setminus t$
Pref.	T x T	$\{t_0, t_1\}, t_0, t_1 \in T$	$\{t_0, t_1\}$	$c^+ : t \in q \quad c^- : q \setminus c^+$
Corr.	T	$t \in T$	T	$c^+ : t' \in T \quad c^- : q$
Binary	T	$t \in T$	$\{0, 1\}$	$c = 0 \implies c^+ : T \setminus q \quad c^- : q$ $c = 1 \implies c^+ : q \quad c^- : T \setminus q$

the choice set (summarized in Alg. 2). Selecting a query involves optimizing over information gain (IG) as follows:

$$q_i^* = \underset{q \in Q_i(s)}{\operatorname{argmax}} \mathbb{E}_{c|C_i(q)} \left[\mathrm{IG}(\mathcal{W} \mid c) \right] \tag{3.1}$$

$$= \underset{q \in Q_i(s)}{\operatorname{argmax}} \sum_{c \in C_i(q)} \sum_{w \in \Omega} \left[P(c|w) \cdot \log \frac{M \cdot P(c|w)}{\sum_{w' \in \Omega} P(c|w')} \right] \tag{3.2}$$

where Ω contains M samples of the distribution $\mathcal{W}$. The expansion from Eq. 3.1 to 3.2 follows the derivation presented in [10]; see Appendix 6.1.1 for intermediate steps. We adopt the commonly-used Boltzmann-rational equation to define $P(c|\omega)$:

$$P(c|\omega) = \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \tag{3.3}$$

where $\phi(t)$ returns the feature trace of the trajectory t; that is, the sum over the feature vectors of all states visited in trajectory t.[2] Note that Eq. 3 reduces to Bayesian Inverse Reinforcement Learning [44] for each $t \in c^+$. β is a parameter representing the expected optimality of the teacher's feedback with respect to ω. We assign a value of $\beta = 20$ across all interaction types (selected through empirical evaluation).

To minimize the computational complexity of solving for Eq. 3.2, we reformulate it as a series of operations over a $|Q|$ x $|C|$ x $|\Omega|$ probability tensor $\mathbf{P}$, where $\mathbf{P}_{q,c,\omega}$

[2] See Appendix 6.2.1 for each domain's definition of ϕ.

represents the probability (according to weight sample $\omega \in \Omega$) that the teacher will select choice c in response to query q. To construct $\mathbf{P}$, let $\mathbf{E}$ be a N x N x M (i.e., $|T|$ x $|T|$ x $|\Omega|$) tensor representing exponentiated rewards:

$$\mathbf{E}_{t,t',\omega} = e^{\beta \cdot \phi(t') \cdot \omega} \qquad \Longrightarrow \qquad \left[\mathbf{E} + \mathbf{E}^{\mathbf{T}}\right]_{t,t',\omega} = e^{\beta \cdot \phi(t') \cdot \omega} + e^{\beta \cdot \phi(t) \cdot \omega} \tag{3.4}$$

All tensor transposes are performed over the first two axes. With $\mathbf{E}$ in hand, we next define the probability tensors of each interaction type as follows:

$$\mathbf{P}^{(\text{demo})}_{q,c,\omega} = \left[\mathbf{E}_0 \oslash \sum_{t \in T} \mathbf{E}^{\mathbf{T}}{}_t\right]_{c,\omega} \qquad \text{(since } |Q| = 1 \text{ for demonstrations)} \tag{3.5}$$

$$\mathbf{P}^{(\text{pref})}_{q,c,\omega} = \left[\left(\mathbf{E} \oslash \left(\mathbf{E} + \mathbf{E}^{\mathbf{T}}\right)\right)^{\mathbf{T}}, \mathbf{E} \oslash \left(\mathbf{E} + \mathbf{E}^{\mathbf{T}}\right)\right]_{c,q_0,q_1,\omega} \qquad (c \in \{0,1\} \text{ for prefs.}) \tag{3.6}$$

$$\mathbf{P}^{(\text{corr})}_{q,c,\omega} = \left[\mathbf{E} \oslash \left(\mathbf{E} + \mathbf{E}^{\mathbf{T}}\right)\right]_{q,c,\omega} \tag{3.7}$$

$$\mathbf{P}^{(\text{bnry})}_{q,c,\omega} = \left[1 - \left(\mathbf{E}_0 \oslash \alpha \sum_{t \in T} \mathbf{E}^{\mathbf{T}}{}_t\right), \mathbf{E}_0 \oslash \alpha \sum_{t \in T} \mathbf{E}^{\mathbf{T}}{}_t\right]_{c,q,\omega} \qquad (c \in \{0,1\} \text{ for binary rewards}) \tag{3.8}$$

where $\oslash$ represents an element-wise division of two matrices (i.e., $(\mathbf{A} \oslash \mathbf{B})_{ij} = \mathbf{A}_{ij} / \mathbf{B}_{ij}$) and α is a normalization factor such that $\sum_c \mathbf{P}^{(\text{bnry})}_{q,c,\omega} = 1$. For derivations, see Appendix 6.1.3. **The main effect of this formulation is that it enables tractable optimization over multiple interaction types** by sharing a common representation $\mathbf{E}$. To solve for the optimal query q_i^* using interaction type i, we use $\mathbf{P}^{(i)}$ to construct a $|Q|$ x $|C|$ x $|\Omega|$ information gain tensor $\mathbf{G}^{(i)}$:

$$\mathbf{G}^{(i)}_{q,c,\omega} = \mathbf{P}^{(i)}_{q,c,\omega} \cdot \log\left(\frac{M \cdot \mathbf{P}^{(i)}_{q,c,\omega}}{\sum_{\omega' \in \Omega} \mathbf{P}^{(i)}_{q,c,\omega'}}\right) \qquad q_i^* = \underset{q}{\text{argmax}} \sum_{c,\omega} \mathbf{G}^{(\text{i})}_{q,c,\omega} \tag{3.9}$$

We then solve for the optimal interaction type itself. To optimize over both informativeness and *interaction cost*, λ_i may be set according to domain-specific cost factors (e.g., the time or mental load involved in answering a query) for each interaction type.[3] To perform an *unweighted* optimization and maximize solely over

[3]In our evaluations, we assign a cost of 20 to each demonstration, 15 to each correction, 10 to each preference, and 5 to each binary query.

the informativeness of each query, let λ_i be a constant value over all interaction types $i \in \mathcal{I}$.

$$i^* = \underset{i \in \mathcal{I}}{\operatorname{argmax}} \frac{1}{\log(\lambda_i)} \sum_{c,\omega} \mathbf{G}^{(\mathrm{i})}_{q_i^*,c,\omega} \tag{3.10}$$

We summarize this process in Alg. 2.

3.1.2 Update Weights from Feedback

After presenting the optimal query to the teacher, the agent receives feedback and appends it to a feedback set $\mathbf{F}$—a cumulative set that contains all feedback received by the agent thus far. We then update the weight estimate such that it maximizes the likelihood of all feedback in $\mathbf{F}$:

$$\omega^* = \underset{\omega}{\operatorname{argmax}} \prod_{c \in \mathbf{F}} P(c|\omega) = \underset{\omega}{\operatorname{argmax}} \prod_{c \in \mathbf{F}} \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \tag{3.11}$$

We calculate the gradient over ω by differentiating over its log-likelihood given $\mathbf{F}$:

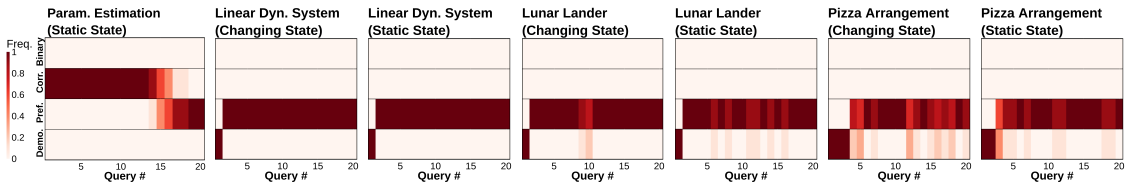
$$\frac{\partial \ell(\omega)}{\partial \omega_j} = \sum_{c \in \mathbf{F}} \left[\frac{\sum_{t \in c^+} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}} - \frac{\sum_{t \in c^+ \cup c^-} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \right] \tag{3.12}$$

$$= \sum_{c \in \mathbf{F}} \left[\beta \cdot \phi_j(c_0^+) - \frac{\sum_{t \in c^+ \cup c^-} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \right] \quad (\text{iff } |c^+| = 1) \tag{3.13}$$

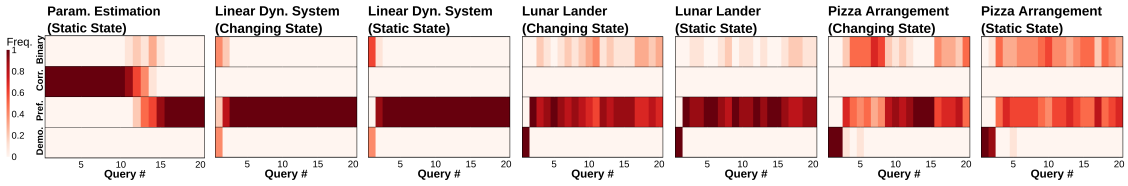
See Appendix 6.1.4 for the full derivation. After receiving feedback from each query and updating $\mathbf{F}$, we approximate Ω by randomly initializing and then performing gradient ascent on each weight sample $\omega \in \Omega$.

3.2 Results

We simulate four types of learning problems in robotics using an oracle teacher to obtain controlled evaluations. The oracle teacher, similar to INQUIRE, requires its own set of trajectory samples T'. It then selects a response to a query via one of three mechanisms: returning the highest-reward trajectory from its choice space (demonstrations/preferences), rejection sampling of trajectories followed by selection of the trajectory with the highest reward-to-distance ratio from the queried trajectory



(a) Selected interaction types *without* cost-weighting



(b) Selected interaction types *with* cost-weighting

Figure 3.1: Heatmaps illustrating how INQUIRE selects different interaction types as it learns more over time. These selections differ when deriving unweighted (top) or cost-weighted (bottom) information gain estimations. In the cost-weighted setting (bottom), INQUIRE selects more low-cost binary queries than it does in the unweighted setting (top).

(corrections), and returning whether a query meets or exceeds a reward threshold (binary reward). Implementation details can be found in Appendix 6.2.2.

The **Parameter Estimation** domain involves directly estimating a randomly-initialized, ground truth weight vector ω^* containing 8 parameters. The **Linear Dynamical System** domain, inspired by [10], simulates a controls problem and involves learning 8 parameters. The **Lunar Lander** domain [7] simulates a controls problem involving 4 parameters. The **Pizza Topping Placement** domain simulates a preference-learning problem involving 4 parameters. Each domain (except for Parameter Estimation) has a *static*-state and *changing*-state condition indicating whether the robot must formulate all queries from the same query state or not, respectively. For the full evaluation procedure and oracle implementation details for each domain see Appendix 6.2.

3.2.1 INQUIRE Query Selection

We first analyze how INQUIRE selects queries. Figure 3.1 reflects the changes in interaction types selected by INQUIRE over time. Figure 3.1a first reports these interaction selections in an unweighted query optimization setting, where all interaction types are assumed to be equally costly. In the parameter optimization

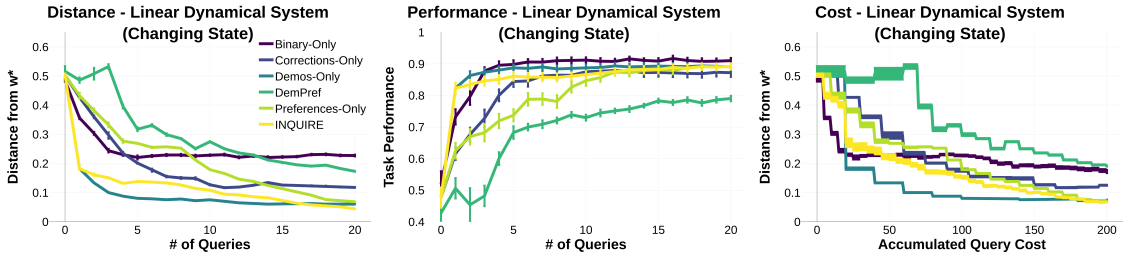
domain, INQUIRE requests corrections in the first 14-18 queries and then requests preferences as the remaining queries. Demonstrations were not enabled in this domain. In all other domains, INQUIRE requests a demonstration as its first query, then immediately switches to requesting preferences for the remaining queries (occasionally alternating between preferences and demonstrations in the Lunar Lander domain).

After assigning different cost values to each interaction type, INQUIRE chooses more diverse interaction types in order to maximize its information-to-cost ratio. As shown in Figure 3.1b, this typically results in INQUIRE posing more binary queries due to their relatively low cost. This pivot toward binary queries may occur at the start (as seen in the linear dynamical system), middle (as seen in the parameter estimation domain), or interspersed throughout the learning process (as seen in the lunar lander domain).

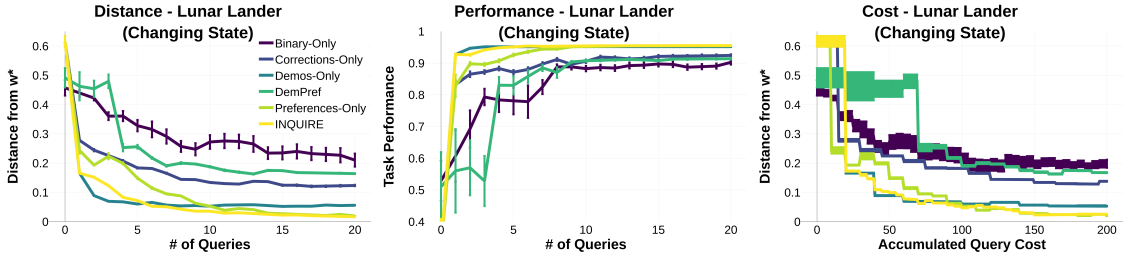
3.2.2 Learning Performance

We now analyze the effect of INQUIRE's interaction type selections on its learning performance and compare against two types of baselines. The first, DemPref [43], learns from 3 demonstrations and then learns from preference queries by using a volume removal objective function. As our second baseline, we compare INQUIRE against agents that use only one form of interaction: demonstrations, preferences, corrections, or binary reward. Note that the preference-only agent is formulated according to [10] and thus represents this baseline method.

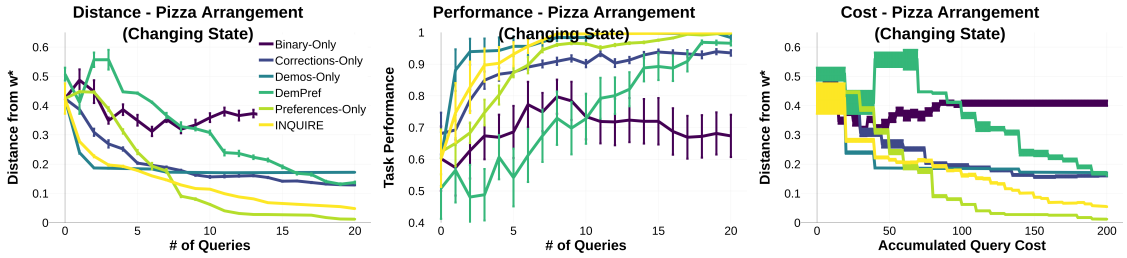
We first consider the changing-state formulation of each domain, where the robot is presented with a new state for each query. Since the Parameter Estimation domain does not contain states, we exclude it from this first set of results. Figure 3.2 illustrates this learning performance in the Linear Dynamical System and Lunar Lander domains according to three key metrics. **Distance** measures the angular distance between the ground truth feature weights (ω^*) and the algorithm's estimated feature weight $\tilde{\omega}$ after each query. **Performance** measures the task reward achieved using a trajectory optimized according to $\tilde{\omega}$ (the algorithm's estimated feature weight after each query). Performance is scaled between 0-1, with 0 and 1 representing the worst and best possible task rewards according to ω^*, respectively. Note that INQUIRE's distance and performance metrics are achieved in the unweighted condition. **Cost-vs-Distance**



(a) Linear Dynamical System



(b) Lunar Lander Task



(c) Pizza Arrangement Task

Figure 3.2: Metrics for the *changing state* condition in which the robot's initial state changes with each query. Error bars/regions represent variance across multiple evaluation runs with randomized query states and initial weights. Cost metrics are cut off after 20 queries for the *binary-only* method in (c) due to extensive computation times.

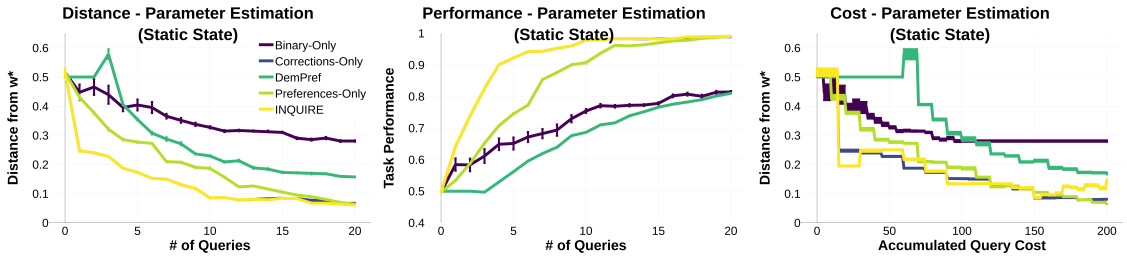
measures the relationship between the cumulative cost of each query and the resulting distance between $\tilde{\omega}$ and ω^* after each query. INQUIRE's metrics in this graph are achieved in the cost-weighted condition.

Figure 3.3 presents the same three metrics for the *static-state* condition in which all 20 queries must be selected from the same initial state. Finally, we quantify these graphs by reporting the area-under-the-curve (AUC) metrics for the distance, performance, and cost curves across all tasks. These metrics are available in Appendix 6.3. The AUC metrics indicate that, compared to the baseline methods, INQUIRE results in the best average learning performance (measured both by the distance and performance plots in Figures 3.2-3.3) across all domains and dominates learning performance in the static-state domains. INQUIRE also results in the best average distance-to-cost ratio across all domains.

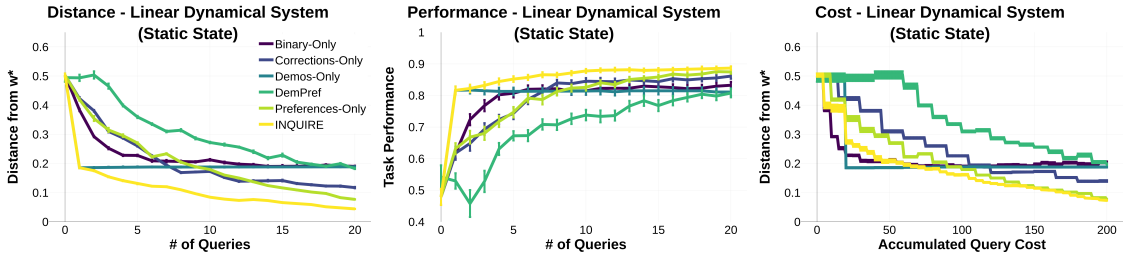
3.3 Discussion

The results show the importance of dynamically selecting interaction types according to the robot's current state. For example, demonstrations can be highly informative when provided in novel states, but when the robot may only query a teacher from a single state, multiple demonstrations are likely to be very similar (if not identical). As a result, receiving multiple demonstrations in a static query state is uninformative. We see the benefits of dynamically selecting interaction types in Figure 3.3, where INQUIRE outperforms all single-interaction methods by optimizing both query type and content to maximize the informativeness of the query feedback.

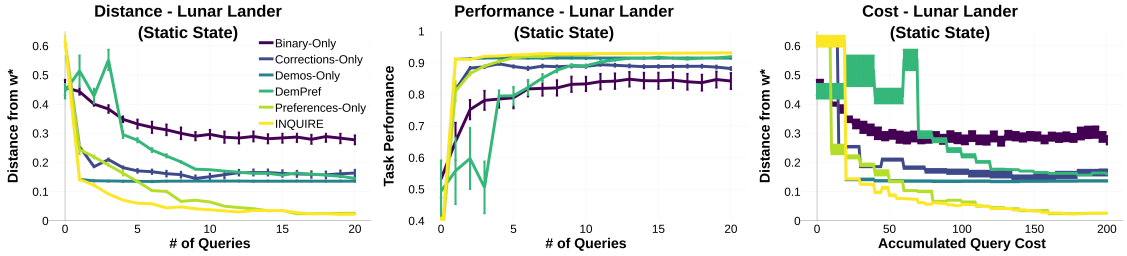
INQUIRE selects the interaction type that, after receiving feedback, minimizes the entropy over its distribution of weight estimates $\mathcal{W}$. This distribution thus serves as a representation of the robot's current model of the task reward. Figure 3.1a illustrates how INQUIRE changes the query type as it learns over time (represented by # of queries). This is particularly evident in the Parameter Estimation task, where the algorithm originally requests corrections until it has refined its model of the task reward to a point where preferences become more informative (after 14-18 queries). Overall, dynamically adapting to the robot's model of the task reward results in better performance than adopting a fixed strategy for selecting interaction types (i.e., DemPref, which always requests demonstrations before selecting preferences).



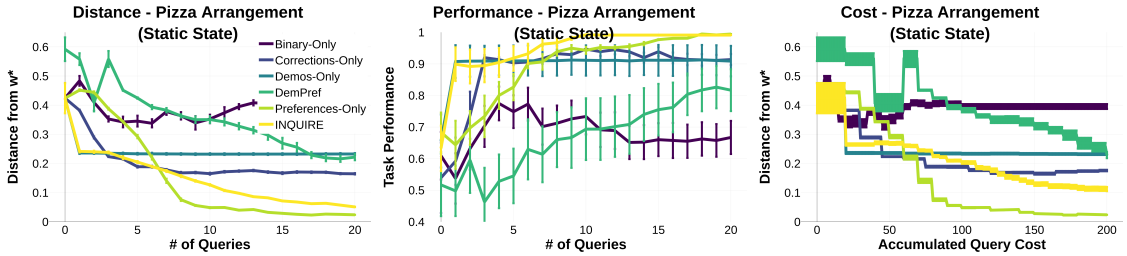
(a) Parameter Estimation Task



(b) Linear Dynamical System



(c) Lunar Lander Task



(d) Pizza Arrangement Task

Figure 3.3: Metrics for the *static state* condition in which the robot is presented with the same state for all 20 queries. Error bars/regions represent variance across multiple evaluation runs with randomized query states and initial weights. Cost metrics are cut off after 20 queries for the *binary-only* method in (a) and (d) due to extensive computation times.

An added benefit of INQUIRE is that it can incorporate a cost metric to identify cost-aware, informative queries. The AUC metrics for the cost graphs indicates that INQUIRE selects queries that, on average, minimize the cost-to-distance ratio across all domains. We expect that this cost metric is domain-specific, and can represent a number of human factors that the algorithm should take into account (e.g., the effort involved for a human to respond to each query type [18]). The cost metric used in our study thus serves as an example of how INQUIRE can factor in interaction costs.

3.4 Limitations

Our evaluation is performed using feedback from an optimal oracle. Real human feedback, however, is likely to be at least somewhat sub-optimal, and its severity likely depends on the interaction type. For example, a non-optimal demonstration may be one that is sufficient but not ideal for completing the task. In contrast, binary rewards offer only two feedback choices to the user, and thus a non-optimal binary reward may indicate the opposite information from what the user intended to convey. These examples illustrate how non-optimal feedback may need to be handled differently depending on the interaction type, and thus, should affect INQUIRE's estimation of information gain. Future work should investigate setting separate values of β (see Eq. 3) for each interaction type, with the goal of reflecting interaction-specific expectations for sub-optimal feedback.

Furthermore, INQUIRE does not yet have the ability to select the state in which it queries the teacher. Prior work in Active Learning has shown that state selection can improve the informativeness of resulting demonstrations [33], and we expect that optimizing over the query state in addition to query type and content would improve the performance of INQUIRE.

3.5 Conclusion

We introduced INQUIRE, an algorithm enabling a robot to dynamically optimize its queries and interaction types according to its task knowledge and its state within the environment. We showed that using information gain to select not just optimal

queries, but optimal interaction *types*, results in consistently high performance across multiple tasks and state configurations. Future work will include formal user studies to investigate our method's efficacy with people of varied skillsets and comfort with robots; incorporation of novel interaction types and other communication modalities; and alternative representations of the reward function and feature spaces. Moreover, we are excited at the possible extensions others might present by using our open-source framework[4] for evaluating and comparing active-learning agents across multiple environments and simulated teachers.

[4] https://github.com/HARPLab/inquire

Chapter 4

Exploring Reward Functions for Human Interactive Robot Learning

Personal robots should be able to learn the preferences of individual humans in ways that are both efficient and convenient to the person. In the context of Human-Interactive Robot Learning (HIRL), a human's preferences can be captured as a latent reward function, and to determine how to act in accordance with a human's preferences, a robot must learn a model that approximates this reward function. While there are various approaches to approximating such a reward function—for example, through large amounts of data or pre-engineered cost functions—we're interested in scenarios in which the model can be learned online through a series of actively selected queries (e.g., requests for critiques, corrections, or demonstrations of tasks) posed by an agent to a human teacher [17, 42]. Our work described in Chapter 3 explored how a robot can select from multiple interaction types to maximize information gain for improving the model. In contrast, the work described in this chapter explores ways in which model *representation* might affect an agent's ability to learn human preferences when taught through a diverse selection of interaction types.

We explore how the choice of reward function representation affects reward learning. In particular, we explore Comparative, Comparative+Specific, and Grid reward representations with the presumption that certain tradeoffs are made when choosing to learn using each. Each representation is a variant of a linear combination

of parameter weights and features. Because we only learn the parameter weights, the Comparative variant enables intuitive interpretation and computational simplicity. In contrast, the Comparative+Specific representation provides superior representational power—that is, an ability to specify the parameters of a reward function—at the expense of learning both the parameter weights and the specific parameter values. Finally, a Grid representation is an attempt to capture the "middle ground" between the two aforementioned representations by assigning weights to a predefined set of specific feature values.

Additionally, we are interested in the effects of reward representation upon an agent's ability to learn from *multiple interaction types*. More specifically—and as in the work described in Chapter 3—we consider corrections, critiques, demonstrations, and preference queries. By incorporating different interaction types, learners can query teachers for information at different levels of granularity. Moreover, multiple interaction types give people the opportunity to teach learners in different ways that might be more intuitive or preferable. As such, we design the learner using a particle-based belief model that is amenable to the different reward representations and active query selection from among multiple interaction types. We also investigate how reward representation affects an agent's ability to learn when querying with single interaction types in comparison to interaction types actively selected by the agent.

We conduct our investigation in two task domains which differ in how task reward is derived. In the first task, reward only derives from the final, discrete state of the environment. In the second task, reward is accumulated over a sequence of time-dependent state-action pairs that comprise a continuous trajectory. We investigate if a learner's reward representation has different effects depending on these two task structures.

Through experiments in simulation, we find that an agent learns best when its reward representation matches that of its teacher. Moreover, we find that an agent's reward representation directly affects the agent's query selection when choosing from among multiple interaction types; namely, agents who learn a Comparative+Specific reward function tend to select a larger diversity of interaction types. Finally, we investigate if our findings from simulation are pertinent to real-world interactions between people and agents through an IRB-approved, in-person user study and

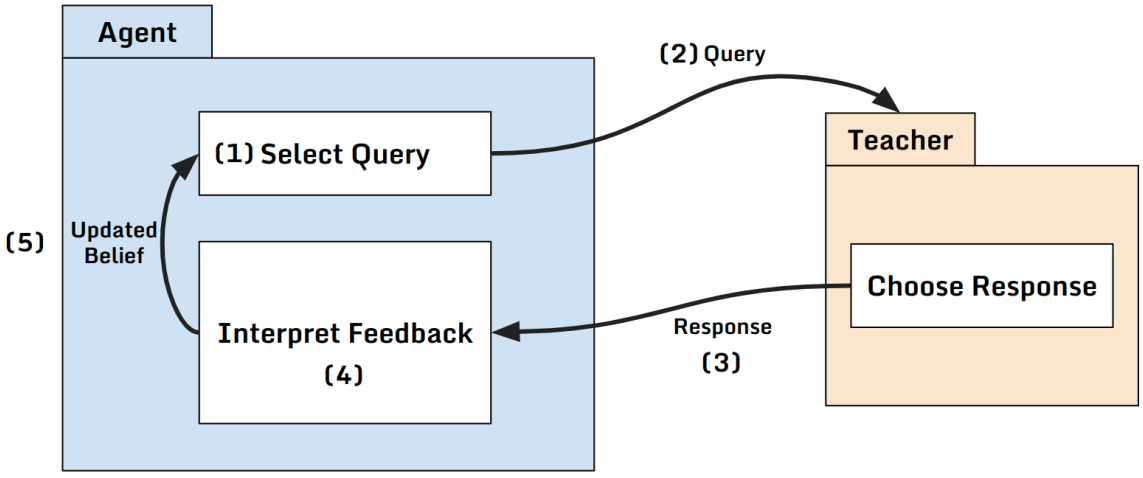


Figure 4.1: **Teaching Session Pipeline:** (1) Select a query by estimating which interaction type provides the largest information gain given the current model of the teacher. (2) Pose query to human/simulated teacher. (3) Receive response. (4) Assign probability mass to particle models and update belief by resampling particles in proportion to probability mass. (5) Use updated belief to repeat the querying process.

find that agent learning performance and active selection of interaction types when learning from human teachers largely resembles our findings from simulated teaching sessions.

In summation, this work contributes:

1. an investigation of three different reward representations in two different tasks with simulated human models,
2. comparisons of these models' learning performances across different interaction types, and
3. an in-person user study to validate the results from simulation.

4.1 Problem Statement and Assumptions

We aim to estimate an individual's reward function through feedback provided to a learning agent during a single teaching session. To this end, we assume there are four interaction types with which a robot can query a human: demonstrations, corrections, binary critiques, and preference queries [30]. Let the interaction between human and robot consist of a sequence of *interaction instances* in which each consists of a *query*

	Query Space $q \in Q_i(s)$	**Choice Space** $\boldsymbol{c}^* \in C_i(q)$	**Boltzmann Representation** $P(\boldsymbol{c}^*; q)$
Demo.	$\{\Xi\}$	$\boldsymbol{\xi} \in \Xi$	$\frac{\exp(\beta \cdot R(\boldsymbol{c}^*))}{\int_C \exp(\beta \cdot R(\boldsymbol{c}))d\boldsymbol{c}}$
Pref.	$\Xi \times \Xi$	$\boldsymbol{\xi} \in \{\boldsymbol{\xi}_0, \boldsymbol{\xi}_1\}$	$\frac{\exp(\beta \cdot R(\boldsymbol{c}^*))}{\sum_C \exp(\beta \cdot R(\boldsymbol{c}))}$
Corr.	Ξ	$\boldsymbol{\xi}' \in \Xi$	$\frac{\exp(\beta \cdot R(\boldsymbol{c}^*))}{\int_C \exp(\beta \cdot R(\boldsymbol{c}))d\boldsymbol{c}}$
Crit.	Ξ	$\boldsymbol{c}^* \in \{-1, +1\}$	$\frac{\exp(\boldsymbol{c}^* \cdot \beta \cdot R(\boldsymbol{\xi}))}{\sum_C \exp(\boldsymbol{c} \cdot \beta \cdot R(\boldsymbol{\xi}))}$

Table 4.1: Each interaction type i has a separate query space Q from which the learner generates [q]ueries, choice spaces C from which the teacher [c]hooses responses, and Boltzmann likelihood representations. Here, $\boldsymbol{c}^*$ denotes the teacher's choice.

and *response*. While the ideal teaching interaction would enable reciprocal queries prompted by both human and robot, we assume the robot leads the interaction and always queries the human.

More formally, let each interaction instance consist of:

- $\boldsymbol{q}_t \in \boldsymbol{Q}$, the query posed by the robot at time t (selected from query types $\boldsymbol{Q}$), which itself consists of
- $\boldsymbol{\xi}$, a trajectory of time-dependent states, and
- $\boldsymbol{c}^*$, the teacher's chosen response during query $\boldsymbol{q}_t$.

As in prior work, we assume the learned reward function is a linear combination of weights and features:

$$R(\boldsymbol{\xi}) = \boldsymbol{\omega} \cdot \boldsymbol{\Phi}(\boldsymbol{\xi}) = \sum_{i=1}^{d} \omega_i \times \phi_i(\boldsymbol{\xi}) \tag{4.1}$$

where $\boldsymbol{\Phi}$ is a vector-valued feature-function comprising basis functions $\phi_i : \boldsymbol{\xi} \to \mathbb{R}$; $\boldsymbol{\omega}$ is a vector of real-valued, non-negative weights; and ω_i corresponds to feature $\phi_i(\boldsymbol{\xi})$. We assume that all task-relevant features are captured by this reward function.

4.2 Reward Function Representations

Because particles can encode a reward function of arbitrary form, we use a particle filter belief model to explore the effects of reward function representation on learning performance. We examine three variants of the feature-function $\boldsymbol{\Phi}$ associated with the oft-used linear combination reward function $R(\boldsymbol{\xi}) = \boldsymbol{\omega} \cdot \boldsymbol{\Phi}(\boldsymbol{\xi})$ defined in Equation 4.1. Intuitively, these variants tradeoff between simplicity and specificity.

Empirically, we observed learning performance improved when reward weights $\boldsymbol{\omega}$ were constrained to be within $(0, 1)$. This constraint affected how we implemented the Comparative variant.

4.2.1 Comparative Variant

For tasks defined through the comparative preference representation, the learning agent only seeks to learn the weights $\boldsymbol{\omega}$ over features of the state-space. In this formulation, the features are simply the state-space parameterization of $\boldsymbol{\xi}$; that is, $\boldsymbol{\Phi} : \boldsymbol{\xi} \rightarrow \boldsymbol{\phi}$. Since $\boldsymbol{\omega}$ is constrained to be within $(0, 1)$, the learned reward function cannot down-weight features (as can be done when $\boldsymbol{\omega}$ resides within $(-1, +1)$). To address this shortcoming, the Comparative featurization includes both standard and "negated" versions of each feature. For example, a car's feature space includes velocity and *negated* velocity. In this way, if the model should encode low velocity, it can assign a large positive weight to "negated velocity" and a small positive weight to "positive velocity."

4.2.2 Comparative + Specific Variant

Tasks defined with a Comparative + Specific representation have two learnable sets: the weights $\boldsymbol{\omega}$ and the desired state-space feature values. We use absolute exponential features to model a reward function of the form:

$$R_k(\boldsymbol{\xi}) = \sum_{i=1}^{d} \hat{\omega}_i^{(k)} \phi_i(\boldsymbol{\xi}) = \sum_{i=1}^{d} \hat{\omega}_i^{(k)} \exp\big[- |\hat{\phi}_i^{(k)} - \phi_i^{(\xi)}| \big]. \tag{4.2}$$

Here, $\hat{\phi}_i^{(k)}$ is the k-th particle's approximation to the teacher's i-th optimal feature

value from the set $\boldsymbol{\phi}$ derived from $\boldsymbol{\xi}$. Since the agent can model specific feature values (i.e., $\hat{\boldsymbol{\phi}}$) along with specific feature weights (i.e., $\hat{\boldsymbol{\omega}}$) using this model, it is possible that it could be able to capture a more precise representation of a human's preferences.

4.2.3 Grid Variant

The Grid reward function combines traits from the two aforementioned preference representations. As is the case for the Comparative preferences, the learning agent aims to learn only the set of weights $\boldsymbol{\omega}$ when operating with a Grid model. Additionally, this representation's featurization has the same structure as the Comparative+Specific reward function, but in the Grid representation, each of a trajectory's feature values are compared to a predefined set of possible values that each feature could take. In this work, we define those values by creating a "grid" with the minimum, middle, and maximum values each ϕ_i could encode. The representation is thus:

$$
\begin{aligned}
R_k(\boldsymbol{\xi}) &= \sum_{i=1}^{d} \hat{\omega}_i^{(k)} \phi_i(\boldsymbol{\xi}) \\
R_k(\boldsymbol{\xi}) &= \sum_{i=1}^{d} \hat{\omega}_{i,\text{min}}^{(k)} \exp\big[-|\phi_i^{(\text{min})} - \phi_i^{(\xi)}|\big] \\
&\quad + \hat{\omega}_{i,\text{mid}}^{(k)} \exp\big[-|\phi_i^{(\text{mid})} - \phi_i^{(\xi)}|\big] \\
&\quad + \hat{\omega}_{i,\text{max}}^{(k)} \exp\big[-|\phi_i^{(\text{max})} - \phi_i^{(\xi)}|\big].
\end{aligned}
\tag{4.3}
$$

Intuitively, the weights $\hat{\boldsymbol{\omega}}^{(k)}$ now can favor the minimum, middle, or maximum values of each state-space parameter. Moreover, comparable weightings on two of the features (e.g., equal weight placed on the minimum and middle feature values) implicitly discretizes the space even further; such a weighting favors values between the "grid-points."

4.3 Active Query Selection and Learner Belief Updates

To generate the queries posed by the agent, we follow the work discussed in Chapter 3 and choose the interaction type expected to maximize the information gain yielded by the agent's next query [10, 22]. While additional factors can be used to inform query selection (e.g., the effort needed to provide each feedback type), our work only accounts for expected information gain.

If we call $\boldsymbol{\theta}$ the set of learnable parameters (e.g., reward function weights $\boldsymbol{\omega}$, or both $\boldsymbol{\omega}$ and $\boldsymbol{\phi}$), to compute the expected information gain for each interaction type, we find the KL-divergence between the agent's prior belief $P(\boldsymbol{\theta})$ and the expected belief if that interaction type was used to query the human $P(\boldsymbol{\theta}; \boldsymbol{q}_t)$. The optimal query at time t can be represented as:

$$\begin{aligned} \boldsymbol{q}_t^* &= \underset{\boldsymbol{q}_t \in \boldsymbol{Q}}{\operatorname{argmax}} \, \mathbb{E}\Big[IG(\boldsymbol{q}_t)\Big] \\ \boldsymbol{q}_t^* &= \underset{\boldsymbol{q}_t \in \boldsymbol{Q}}{\operatorname{argmax}} \Big[D_{KL}\left(\mathbb{E}_{\boldsymbol{q}_t}\big[P(\boldsymbol{\theta}; \boldsymbol{q}_t)\big] || P(\boldsymbol{\theta})\right)\Big]. \end{aligned} \tag{4.4}$$

Intuitively, Equation 4.4 seeks the query (among all interaction types) which is most likely to maximize the agent's information gain. When found, we then use that interaction type to query the human. As in prior work and the work described in Chapter 3, we base each representation off of the Boltzmann-rational distribution which is principally featured by the parameter β and which makes this representation an approximation of human rationality or confidence. Table 4.1 enumerates each interaction type's Boltzmann representation as we've implemented them in this work. Note that we empirically selected $\beta = 20$.

Upon posing the query and receiving the teacher's response, the agent updates its belief over particle models in a Bayesian fashion. More specifically, if $P(\boldsymbol{\theta})_{1:t-1}$ is the agent's prior belief over reward function parameters $\boldsymbol{\theta}$ after $t-1$ interaction instances, the agent's updated belief after interaction instance t is:

$$P(\boldsymbol{\theta}_{1:t}) = P(\boldsymbol{\theta}_{1:t-t}; \boldsymbol{c}_t^*) \cdot P(\boldsymbol{\theta}_{1:t-1}) \tag{4.5}$$

Algorithm 3 The main teaching session loop

```
agent: Learns reward function
teacher: Responds to agent queries
Q ← [Correction, Critique, Demonstration, Preference]
for t in interaction_instance_count do
    Q*_t = agent.select_query(Q)
end for
```

The general query-then-update procedure is outlined in Algorithm 3. Note that `interaction_instance_count` is the number of interaction instances that take place in a single teaching session. We preset this value to be 20 in our simulation evaluations and 10 in our user study.

4.4 Simulation Experiments

We first evaluate the learning performance of the different reward function representations across simulated teaching sessions. In this scenario, a learning agent is taught by an optimal human oracle in the two task domains. To investigate each reward representation's ability to handle tasks of varied complexity, we evaluate each task when defined with three and six state-space features. The task features are enumerated in Table 4.2.

For each task simulation, a single teaching session consisted of a learning agent which queried a simulated human oracle over 20 sequential interaction instances. Results were computed as an average over 10 sessions.

4.4.1 Evaluation Domains

Pizza Topping Placement Task The goal of the Pizza Topping Placement task is to teach an agent how to place toppings on a pizza according to the teacher's preference. Each interaction instance begins with a pizza void of toppings. Then, either agent or oracle places toppings one at a time until the reward it receives from the updated pizza configuration begins to decrease. Table 4.2 enumerates the task's feature space, and Section 4.4.2 provides details on how each interaction type is carried out.

Lunar Lander	Pizza Topping Placement
Distance from Landing Pad	X-Centroid (all toppings)
Lander Angle	Y-Centroid (all toppings)
Feet on Ground	Surface Coverage
Velocity	Min. Distance Between Toppings
Final Distance from Landing Pad	X-Variance (all toppings)
Angular Velocity	Y-Variance (all toppings)

Table 4.2: **Features** for each task domain. Experiments with lower task complexity use the first three features only.

Lunar Lander Task The second task is to teach an agent how to land OpenAI's Lunar Lander [7]. The episode begins with the lander somewhere in the sky, and the goal is to navigate the lander to the landing pad such that it lands in an upright pose. See Table 4.2 for the Lunar Lander feature space.

4.4.2 Simulated Human Oracle

We design our human oracle as a greedily-optimal agent which, when presented with any query $\boldsymbol{q}_t$, maximizes the immediate reward it can receive from that query. During evaluations, the oracle is instantiated with one of the reward function representations enumerated in Section 4.2 depending on the trial at hand.

When asked to provide a **demonstration** in the Pizza Topping Placement task, the oracle generates an optimal example by running a stochastic hill climbing trajectory optimization guided by its reward function. In the Lunar Lander domain, the oracle optimizes a trajectory demonstration using the cross-entropy method [52].

When asked for a **correction** to a pizza, the oracle takes the demonstrated topping arrangement and performs a stochastic hill climbing optimization constrained by the toppings already on the pizza. That is, the oracle can only *rearrange* pizza toppings; it cannot remove or add toppings to the state space.

When providing a correction in the Lunar Lander domain, the oracle uses dynamic time warping to compare its optimal trajectory to that which was demonstrated [24]. The oracle begins its comparison from the start-state and accumulates the difference in trajectories at each time-step. When that difference grows above some threshold,

it saves that point and then identifies the point between it and the start-state in which the difference between the trajectories changes most drastically. This point is now the "correction point," and the oracle then optimizes a trajectory from that correction point to the goal state using the cross-entropy method.

The oracle responds to **preference queries** by selecting the option from a pair of trajectories which yielded the highest reward. Responses to **critiques** are similar; the oracle has a predefined critique threshold defined by the midpoint between the greatest and least reward it could receive from the task. If the reward earned by the agent's demonstration is above this threshold, the oracle positively rewards the agent; otherwise, it provides negative reward. Moreover, we choose to model "relative" critiques; that is, after the oracle observes agent demonstrations (provided during corrections and critiques), the reward derived from those demonstrations becomes the updated "minimum" reward it expects to earn during the teaching session. As such, the critique threshold increases (decreases) when the agent's performance improves (degrades).

4.4.3 Variables and Metrics

We compare the learning performances of Comparative, Specific+Comparative, and Grid reward function representations. Because we're interested in scenarios in which robots learn from multiple interaction types, we first examine the learning performance when each agent is taught by each oracle using a single interaction type. We then examine learning performance when the agent actively selects its queries from among all four interaction types to see how the learning performance of certain representations might be affected. To measure performance, we compute reward as the similarity between the oracle's ideal trajectory $\boldsymbol{\xi}^*$ and a demonstration provided by the learner after each belief update.

4.4.4 Results

Results in Table 4.3 illustrate how reward representation can influence learning performance in Pizza Topping Placement. Note that because the Pizza Topping Placement results present more variability than Lunar Lander, we analyze them more

		Corrs.		Crits.		Demos.		Prefs.		Active	
Oracle	Agent	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Comp.	Comp.	**0.432**	0.336	0.186	0.276	**0.999**	0.001	**0.665**	0.144	**0.999**	0.002
	Grid	0.419	0.238	**0.399**	0.149	0.584	0.348	0.56	0.343	0.63	0.366
	Comp.+Spec.	0.271	0.137	0.387	0.159	0.997	0.004	0.613	0.182	0.96	0.087
Grid	Comp.	0.179	0.154	0.149	0.257	0.876	0.24	0.38	0.067	0.762	0.27
	Grid	**0.355**	0.281	**0.285**	0.363	**0.999**	0.001	**0.695**	0.195	**0.999**	0.001
	Comp.+Spec.	0.117	0.128	0.067	0.068	0.9	0.133	0.089	0.072	0.705	0.403
Comp.+Spec.	Comp.	0.487	0.183	0.376	0.122	0.961	0.035	0.756	0.069	**0.952**	0.06
	Grid	0.683	0.162	**0.775**	0.125	0.85	0.158	0.834	0.117	0.803	0.152
	Comp.+Spec.	**0.72**	0.153	0.591	0.217	**0.998**	0.001	**0.839**	0.085	0.926	0.126

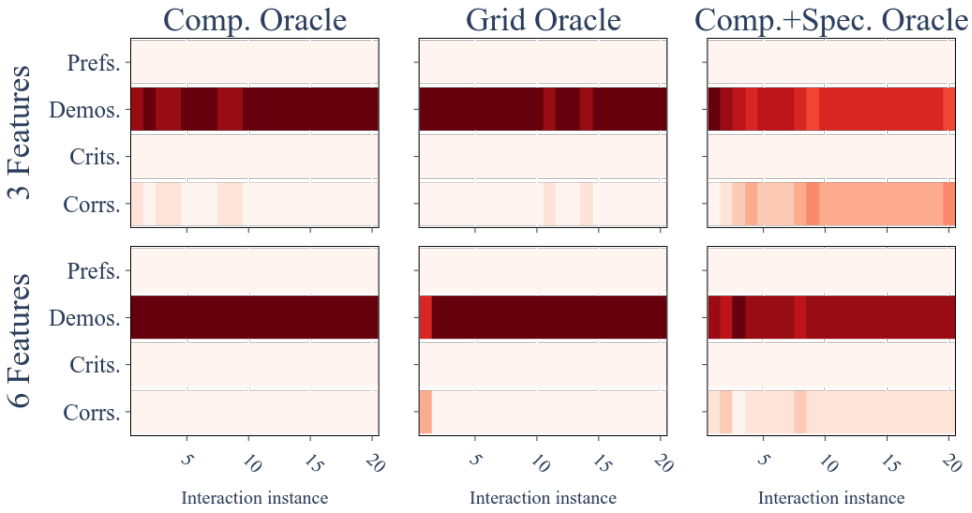
Table 4.3: Mean reward earned for single and actively selected interaction types at the end of the Pizza Topping Placement simulations with six state-space features. Bold numbers indicate the agent which learned best from a particular oracle and interaction type. Typically, the agent learns best when its reward representation matches its teacher's. Moreover, no matter the reward representation, querying with demonstrations alone yields highest final reward.

closely. As can be seen, learning performance is typically best when the agent's representation matches that of its teacher.

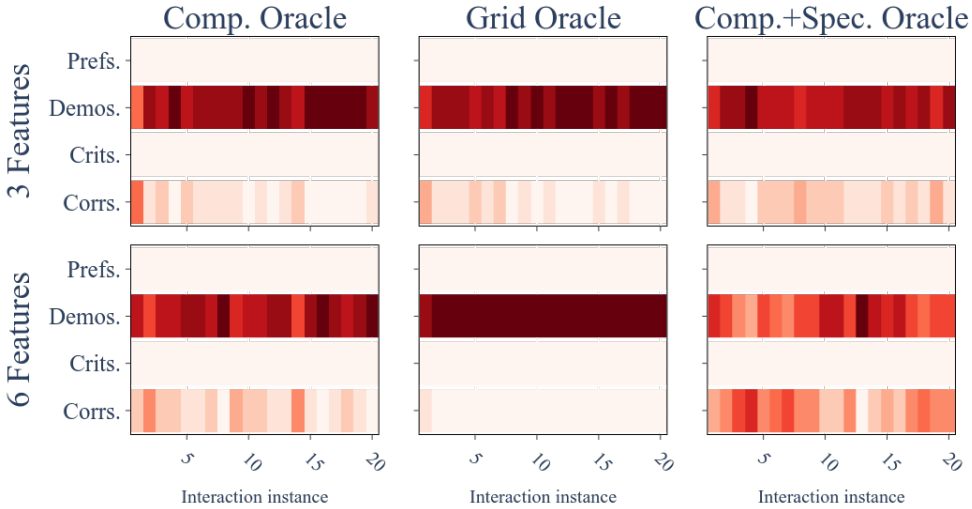
Table 4.3 also depicts the effect that different interaction types have upon learning performance. In most combinations of agent and teacher reward representations, demonstrations yield the highest final reward when only one interaction type is used to query the teacher. However, when interaction types are actively selected by the agent, the Comparative agent outperforms the Comparative+Specific agent when learning from both Comparative and Comparative+Specific oracles.

We also find that the agent's model representation affects its active selection of interaction types with which it prompts the teacher. Figure 4.2 illustrates the distribution of interaction query types for Pizza Topping Placement[1]. When the agent had a Grid representation, it typically chose to query using demonstrations and sometimes corrections (Figure 4.2b). When the agent had a Comparative representation, it mostly chose demonstration queries, though there were some

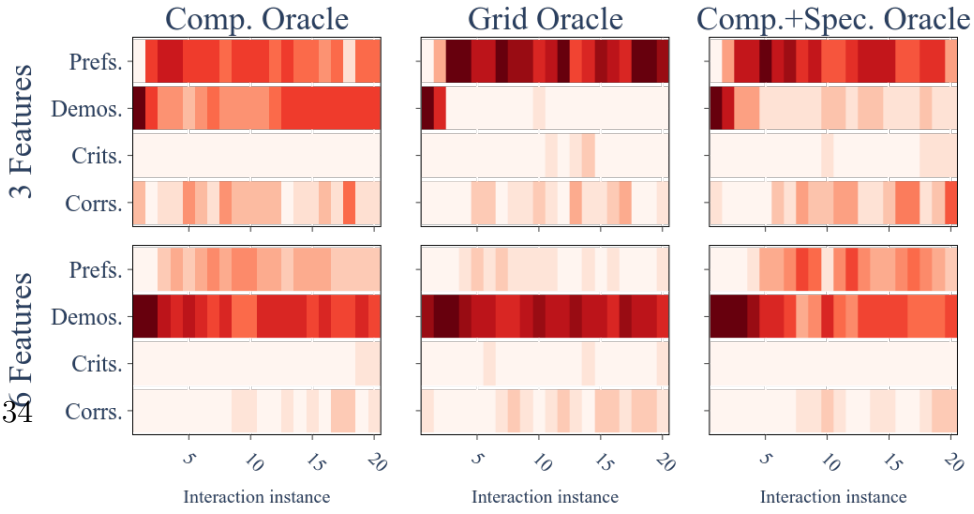
[1]While we did collect results from Lunar Lander, demonstrations were almost always selected no matter the model representation. We suspect the cause to be the massive trajectory space from which demonstrations are generated online; when a cache is used instead, interaction type selection is more diverse (yet performance degrades notably due to the relatively small space of trajectories).



(a) Interactions chosen by agent with Comparative representation.

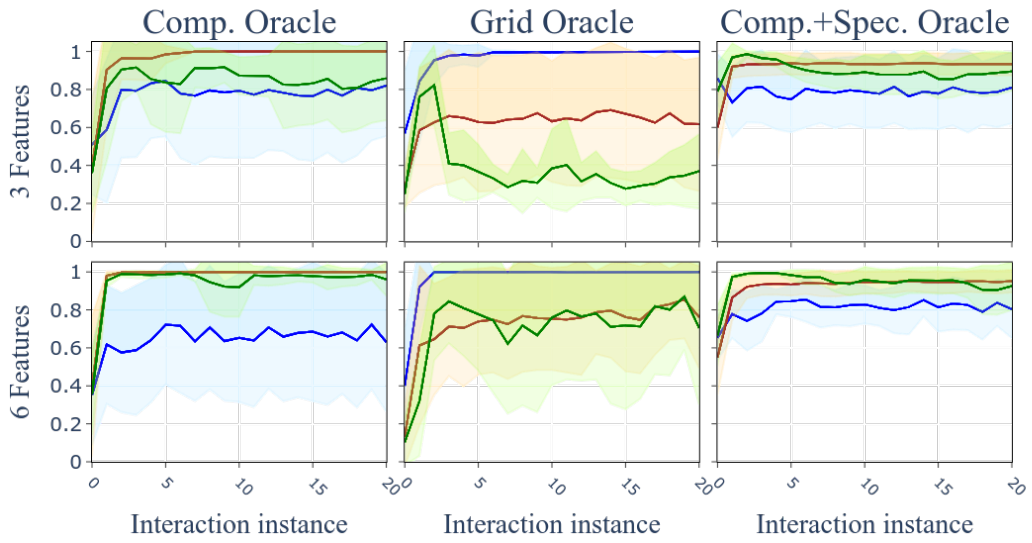


(b) Interactions chosen by agent with Grid representation.

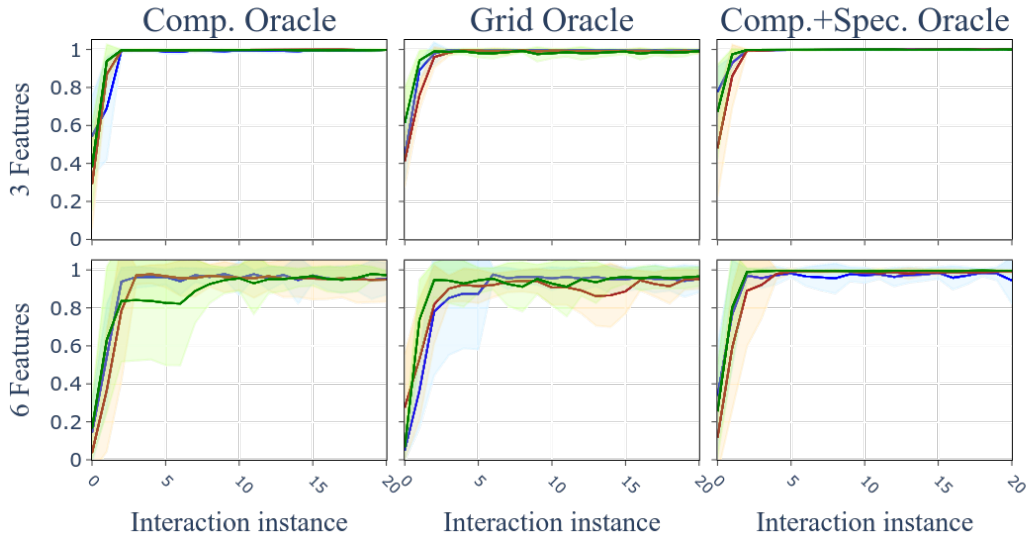


(c) Interactions chosen by agent with Comparative+Specific representation.

Figure 4.2: Learning agent's chose interaction types when actively selected during Pizza Topping Arrangement. Values accumulated across 10 teaching sessions.



(a) **Pizza Topping Placement.**



(b) **Lunar Lander.**

Figure 4.3: The reward earned by each agent when taught by a simulated oracle with a particular reward representation. Brown lines the Comparative learner, blue lines represent the Grid learner, and green lines the Comparative+Specific learner.

correction queries with a Comparative+Specific teacher (Figure 4.2a). When the agent had a Comparative+Specific representation, it typically selected demonstrations and then switched to preferences—a behavior reminiscent of DemPref (Figure 4.2c) [43].

Finally, we find that reward representation affects an agent's ability to learn from the various oracles over time regardless of task complexity. Figure 4.3 depicts the results from the Pizza Topping Placement and Lunar Lander task simulations. Because we're primarily interested in active learning scenarios with multiple interaction types, we discuss the simulations in which the agent actively selected its queries.

In Pizza Topping Placement (Figure 4.3a), the Comparative agent exhibits an ability to learn well from a Comparative+Specific teacher; just the same, a Comparative+Specific agent learns well from a Comparative teacher, save for the 3-feature task. When it comes to the Grid representation, the Grid agent learns well from a Grid teacher, but it struggles to learn from both the Comparative and Comparative+Specific teachers. Comparative and Comparative+Specific agents both struggle to learn from a Grid teacher.

In Lunar Lander with three features (Figure 4.3b), we see strong learning performance (i.e., convergence in less than five interaction instances and small error) no matter the representation used by agent or teacher. When the task dimensionality increases to six features, we start to see differences in performance—most notably in the variance and rate of convergence. On average, each agent appears to demonstrate strong learning performance with all three teachers. Interestingly, the Comparative+Specific agent struggles early on when taught by a Comparative teacher but eventually converges to about the same performance as the Comparative agent; the inverse is true when the Comparative agent is taught by the Comparative+Specific teacher. Surprisingly, the Grid agent demonstrates the same struggles early on when taught by the Grid teacher.

4.4.5 Discussion of Simulation Results

While there are differences in each agent's learning performance across different teacher-learner combinations, the Lunar Lander results showcase that these differences do not derive solely from the different model representations. Lunar Lander proves to be a task domain in which all agents demonstrate strong learning performance

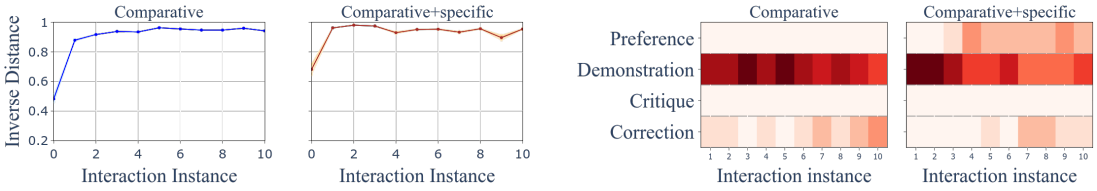
when taught by any teacher; clearly, model representation has small effect on learning this task. More nuance is introduced in Pizza Topping Placement where we see the Grid learner struggle to learn from Comparative and Comparative+Specific teachers and we see the Comparative+Specific learner demonstrate comparable (but slightly inferior) final performance as compared to the Comparative learner when taught by either Comparative or Comparative+Specific teachers.

The finding that learning from demonstrations only yields superior learning performance is consistent with the understanding that demonstrations are the most informative of the studied interaction types. Even so, this result shouldn't lead us to design demonstration-only interactions; other human factors (e.g., human-required effort, frustration, etc.) need to be accounted for when designing algorithms that efficiently learn from people but in ways that are convenient to the human teacher.

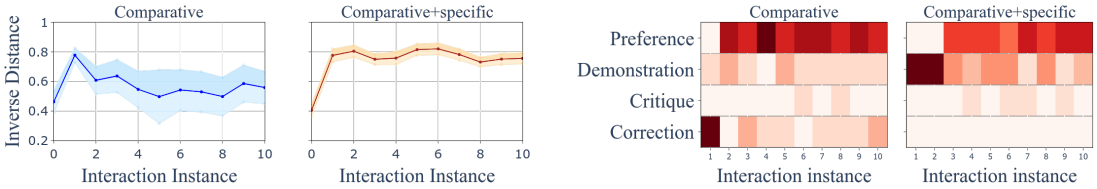
While our primary performance metric is the distance between the teacher's ideal and the agent's demonstration at the end of a teaching session, it is interesting to consider the Comparative+Specific agent's quick convergence to the optimum when taught by a matching oracle. If quick teaching sessions are the goal, perhaps Comparative+Specific is the model of choice given the quickness with which it reaches optimal performance in both simulated tasks.

The results from Table 4.3 and Figure 4.3 suggest that if the teacher's reward model is known, the agent should model the task in a comparable manner. Teaching with critiques only is an exception to this conclusion—the Comparative model performs best for all teachers—although its markedly poor performance makes it undesirable when teaching with a single interaction type. If the teacher's reward model is *not* known, as is the case with most human-agent interactions, a Comparative agent generally matches or outperforms the Comparative+Specific agent, but the difference is small.

As can be seen in Figure 4.2, model representation can affect an agent's choices when actively selecting its queries. If the goal is to have an agent which learns from a variety of interaction types, then it would seem a Comparative+Specific representation could be well suited.



(a) Pizza Topping Placement.



(b) Lunar Lander.

Figure 4.4: **User Study Results.** Similarity metrics and the counts of interaction types selected by each agent in the two task domains when conducted with human participants. In Pizza Topping Placement, we observe strong, consistent learning performances of both Comparative and Comparative+Specific learners. In Lunar Lander, the Comparative+Specific agent demonstrates superior learning performance and consistency; in fact, the Comparative agent's performance appears to degrade as teaching sessions proceed. While the Comparative+Specific agent generally expects demonstrations and then preferences to yield the most information when querying participants, the Comparative agent's expectations appear more domain-dependent.

4.5 User Study

To test if our results from simulation could be leveraged in real-world human-robot interactions, we evaluate the performance of learning agents in an in-person user study with human teachers. Because of resource limitations inherent to user studies, we choose to assess an active learning agent using Comparative and Comparative+Specific representations and omit the Grid representation due to its inferior performance on the Pizza Topping Placement task simulation.

Study Details. We recruited eleven in-person participants (5 men, 6 women; mean age: 23.9, SD: 4.9) and performed the study at Carnegie Mellon University. Participants were recruited through the university's pool of research participants. We performed a within-subjects study (i.e., each participant experienced both experi-

mental conditions) and counterbalanced the order in which participants experienced the conditions. Each participant taught both tasks to a simulated agent over 10 interaction instances, the types of which were actively selected by the agent from the four interaction types described earlier. In the Lunar Lander task, corrections and demonstrations were provided via keyboard inputs while preference and critique feedback was provided via a basic terminal user interface. In the Pizza Topping Placement task, mouse inputs were used to provide each type of feedback.

Running the Lunar Lander simulation to optimize trajectories takes a significant amount of time—on the order of minutes for a single interaction instance. This computation rendered online trajectory generation infeasible for our study. As such, we approximated the space of trajectories using a cache of 2,200 trajectories.

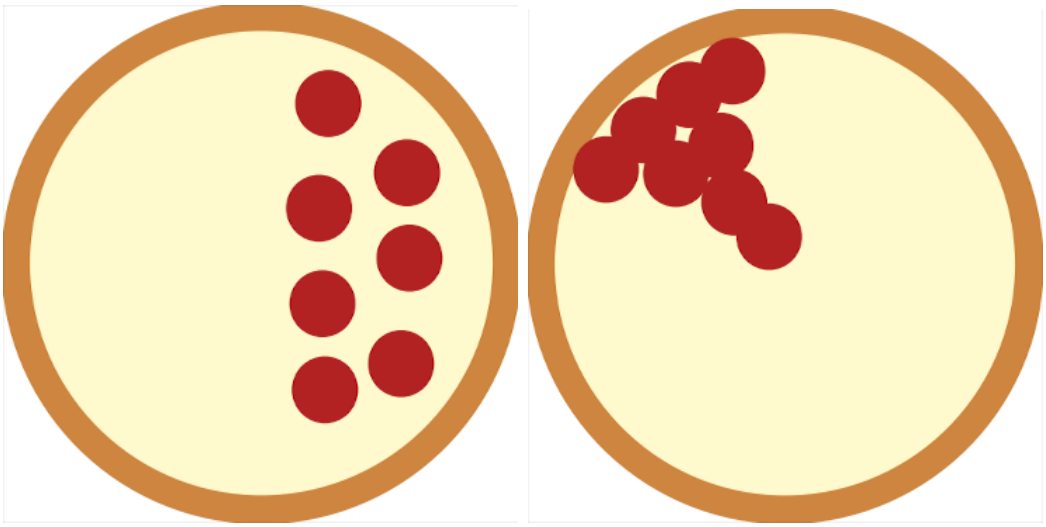


Figure 4.5: Examples of goal topping placements given to participants during Pizza Topping Placement user study.

The task goal in Lunar Lander was apparent: Land the spaceship on the landing pad and in an upright pose. Less obvious was the task goal for Pizza Topping Placement. As such, and to avoid erratic goal pizzas, we provided each participant with a unique topping arrangement to teach the agent (examples shown in Figure 4.5). Topping arrangements were chosen arbitrarily albeit with the objectives of being learnable by the agent (i.e., goals could be captured by the feature space) and apparent to the participant that the agent was (or was not) learning.

To evaluate how the Comparative and Comparative+Specific reward representa-

tions each affect learning performance, we asked each user to provide a demonstration of their ideal trajectory before the teaching session began and then compared its features with the agent's attempted trajectory at each time step. To ensure fair comparison, we computed all similarity metrics in the Comparative+Specific feature space.

4.5.1 Results

Figure 4.4 depicts the performance of each agent averaged across all participants. Both learners demonstrated an ability to learn participants' reward functions in the Pizza Topping Placement task. In Lunar Lander, however, the Comparative+Specific agent performed better than the Comparative agent (Figure 4.4b). While the Comparative agent demonstrated an ability to quickly improve its model, the agent's performance decreased as interactions proceeded. Note that the Comparative agent primarily chose corrections as the first interaction type and then switched to preference queries during Lunar Lander. Given that the Comparative+Specific agent's performance tends to stagnate in its own right after switching to preference queries, perhaps the participants' responses to the agents' preference queries were difficult data points from which the agent could learn no matter the representation.

Figure 4.4 also depicts the frequency with which the agent selected each interaction type. As was the case in our simulation experiments (see Figure 4.2), the Comparative agent primarily selects demonstrations over the course of Pizza Topping Placement and the Comparative+Specific agent selects demonstrations early but then favors preferences. This selection of interaction types for both agents in Pizza Topping Placement aligns with our finding in simulation that the agent's reward representation influences the interaction types with which it will query a teacher. Even so, the learner primarily selected demonstrations during Lunar Lander simulations yet display a greater diversity of selection in the study. We attribute the tendency to pick a diverse selection of interaction types to the pre-computed trajectory cache we employ in our study rather than the online trajectory generation we employed in simulated tasks. Indeed, when we rerun simulations using the trajectory cache, we observe that interaction type selection approximately mirrors the patterns observed during the study. This outcome suggests that both domain complexity and model representation

affect an agent's interaction type selections.

We also gathered qualitative metrics from our participants. At the end of each teaching session, participants were shown the evolution of the agent's belief via a sequence of demonstrations in which each demonstration t corresponded to the agent's belief after receiving a query response at time t. After viewing this sequence, participants reported how much they agreed with the statement: "The agent appeared to learn the goal over time." We collected responses on a 7-point Likert scale and report them in Figure 4.6.

Participants perceived comparable learning performance between the two agents; in both cases, 64% "Somewhat Agree" that the agent learned. Participants tended to observe learning more readily after teaching a Comparative agent in Pizza Topping Placement; 73% of participants at least "Somewhat Agree" that the Comparative agent learned compared to the 55% of participants who felt the same about the Comparative+Specific agent. While consistent with our results from simulation, this result conflicts with our quantitative evaluations that show the Comparative+Specific agent achieves superior learning performance.

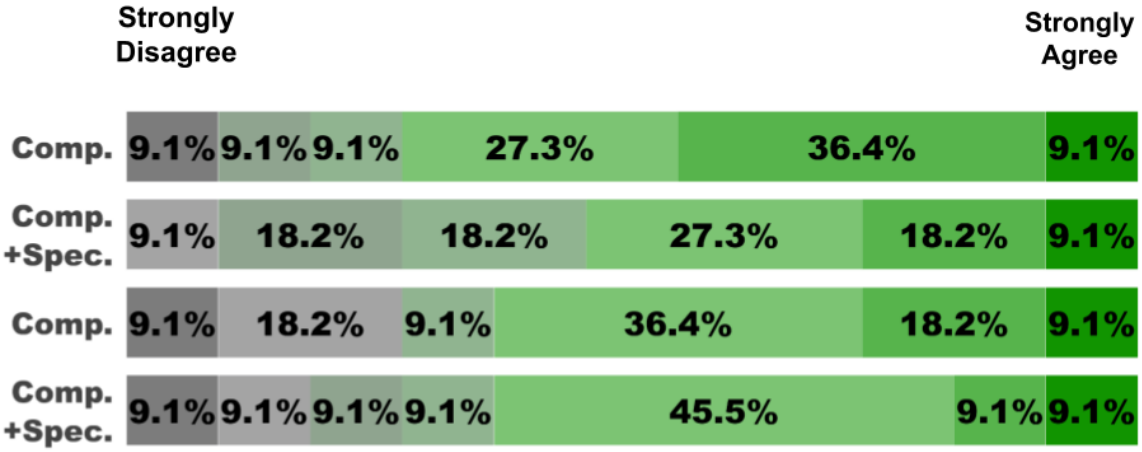


Figure 4.6: Participant responses on a 7-point Likert scale for the Pizza Topping Placement (top two bars) and Lunar Lander (bottom two bars) tasks.

4.5.2 Discussion of User Study Results

We conducted our user study to see if the results corroborate those from simulation. While our quantitative results suggest both agents learn the Pizza Topping Placement

task to a comparable degree (as our teaching simulations also show), the Comparative+Specific agent is the superior learner in Lunar Lander, a result which conflicts with those from simulation. Even so, learning performance is notably inferior for both agents when compared to their performance in simulation, likely due to our use of the pre-computed trajectory cache[2].

Interestingly, the agents' interaction type selections when interacting with people follow the patterns observed in Pizza Topping Placement. While interaction type selection is not the same across Lunar Lander experiments (teaching simulations primarily saw agents select demonstrations), when we rerun simulations with the trajectory cache, the interaction type selections follow the same pattern as that from the study.

Finally, the qualitative results suggest people *may* more readily perceive a Comparative agent's learning. That perception, however, is hardly stronger than that perceived of the Comparative+Specific agent. As such, it would be presumptive to draw conclusions from this result. Indeed, this inconclusive result (which also conflicts with the quantitative superiority of the Comparative+Specific agent) corroborates the difficult task of designing methods which reliably capture human preferences. Furthermore, it is evidence that we should be wary of relying too heavily upon our objective evaluations when deeming certain approaches "good" or "bad" from simulation or user study alone.

4.6 Conclusion

In this work, we take a step towards robots learning human rewards by exploring the ways a robot learner should internally represent their human teacher's preferences over the way a robot executes tasks. Our experiments in simulation indicate that robot learners being taught tasks would best be suited to match their reward function representation with that of their teacher. Our user study corroborates these findings and those that indicate a learner's reward representation influences its selection of interaction types when actively querying its teacher. Future research directions could include further investigation into interaction types and their representations,

[2]When we rerun simulations with the cache, we see similar performance degradation.

incorporating more informed prior information about the human into the robot's querying process, interactions that include back-and-forth querying between learner and teacher, and how to use different interaction types to learn long-term, transferable human preferences.

Chapter 5

Conclusion

This thesis set out to address two needs: 1.) the need for robots to learn from multiple interaction types, and 2.) the need for robots to have effective, efficient internal representations of their human teachers. To address the first, we introduced INQUIRE, an algorithm enabling a robot to dynamically optimize its queries and interaction types according to its task knowledge and its state within the environment. We showed that using information gain to select not just optimal queries, but optimal interaction *types*, results in consistently high performance across multiple tasks and state configurations.

To address the second need, we explored the ways a robot learner should internally represent their human teacher's preferences over the way a robot executes tasks. Our experiments in simulation indicate that robot learners being taught tasks would best be suited to match their reward function representation with that of their teacher. Our user study corroborates these findings and those that indicate a learner's reward representation influences its selection of interaction types when actively querying its teacher.

Future work which builds off of the two projects presented in this thesis include formal user studies to investigate INQUIRE's efficacy with people of varied skillsets and comfort with robots, and alternative representations of the reward function and feature spaces. Additionally, future research directions could include further investigation into interaction types and their representations, incorporating more informed prior information about the human into the robot's querying process, interactions that

include back-and-forth querying between learner and teacher, and how to use different interaction types to learn long-term, transferable human preferences.

Chapter 6

Appendix

6.1 INQUIRE: Approach Details

6.1.1 Information Gain Derivation

Information gain can be computed by calculating the change in entropy in a distribution X after receiving the datapoint y:

$$\begin{aligned} &\mathrm{IG}(X, y) && (6.1)\\ &= H(X) - H(X|y) && (6.2)\\ &= -\mathbb{E}_{x|X}\log P(x) + \mathbb{E}_{x|X,y}\log P(x|y) && (6.3)\\ &= \sum_{x \in X} P(x|y) \cdot \log P(x|y) - \sum_{x \in X} P(x) \cdot \log P(x) && (6.4) \end{aligned}$$

We adapt this generic formulation to use our definitions of interaction types, query space, and choice space, and aim to solve for the optimal query:

$$\max_{q \in Q} \mathbb{E}_{c|C_i(q)} \left[\mathrm{IG}(\mathcal{W}, c)\right] \tag{6.5}$$

We now solve for the expected information gain according to Eqs. 6.1- 6.4 and

following the derivation presented in [10]:

$$\mathbb{E}_{c|C_i(q)}\left[\mathrm{IG}(\mathcal{W}, c)\right] \tag{6.6}$$
$$= H(\mathcal{W}) - \mathbb{E}_{c|C_i(q)}\left[H(\mathcal{W}|c)\right] \tag{6.7}$$
$$= -\mathbb{E}_{\mathcal{W}}\left[\log P(\mathcal{W})\right] + \mathbb{E}_{\mathcal{W},c|C_i(q)}\left[\log P(\mathcal{W}|c)\right] \tag{6.8}$$
$$= \mathbb{E}_{\mathcal{W},c|C_i(q)}\left[\log P(\mathcal{W}|c) - \log P(\mathcal{W})\right] \quad \text{(see proof in Sec. 6.1.1)} \tag{6.9}$$
$$= \mathbb{E}_{\mathcal{W},c|C_i(q)}\left[\log \frac{P(\mathcal{W}|c)}{P(\mathcal{W})}\right] \tag{6.10}$$
$$= \mathbb{E}_{\mathcal{W},c|C_i(q)}\left[\log \frac{P(c|\mathcal{W})}{P(c)}\right] \quad \text{(by Bayes' rule)} \tag{6.11}$$
$$= \sum_{c\in C_i(q)}\left[P(c)\sum_{w\in\mathcal{W}}\left[P(w|c)\cdot\log\frac{P(c|w)}{P(c)}\right]\right] \tag{6.12}$$
$$= \sum_{c\in C_i(q)}\left[P(c)\sum_{w\in\mathcal{W}}\left[\frac{P(w)P(c|w)}{P(c)}\cdot\log\frac{P(c|w)}{P(c)}\right]\right] \tag{6.13}$$
$$= \sum_{c\in C_i(q)}\sum_{w\in\mathcal{W}}\left[P(w)\cdot P(c|w)\cdot\log\frac{P(c|w)}{P(c)}\right] \tag{6.14}$$
$$= \sum_{c\in C_i(q)}\sum_{w\in\mathcal{W}}\left[P(w)\cdot P(c|w)\cdot\log\frac{P(c|w)}{\sum_{w'\in\mathcal{W}}P(w')\cdot P(c|w')}\right] \tag{6.15}$$
$$\approx \frac{1}{M}\sum_{c\in C_i(q)}\sum_{w\in\Omega}\left[P(c|w)\cdot\log\frac{M\cdot P(c|w)}{\sum_{w'\in\Omega}P(c|w')}\right] \tag{6.16}$$

Where Ω contains M samples of the distribution $\mathcal{W}$.

Proof of Eq. 6.9

$$
\begin{aligned}
&- \mathbb{E}_{\mathcal{W}} \left[\log P(\mathcal{W}) \right] + \mathbb{E}_{\mathcal{W}, c|C_i(q)} \left[\log P(\mathcal{W}|c) \right] && (6.17)\\
&= \mathbb{E}_{\mathcal{W}, c|C_i(q)} \left[\log P(\mathcal{W}|c) \right] - \mathbb{E}_{\mathcal{W}} \left[\log P(\mathcal{W}) \right] && (6.18)\\
&= \left[\sum_{w \in \mathcal{W}} P(w) \sum_{c \in C_i(q)} P(c|w) \cdot \log P(w|c) \right] - \left[\sum_{w \in \mathcal{W}} P(w) \cdot \log P(w) \right] && (6.19)\\
&= \sum_{w \in \mathcal{W}} P(w) \cdot \left[\left(\sum_{c \in C_i(q)} P(c|w) \cdot \log P(w|c) \right) - \log P(w) \right] && (6.20)\\
&= \sum_{w \in \mathcal{W}} P(w) \cdot \left[\sum_{c \in C_i(q)} P(c|w) \cdot \log P(w|c) - \sum_{c \in C_i(q)} P(c|w) \cdot \log P(w) \right] && (6.21)\\
&= \sum_{w \in \mathcal{W}} P(w) \cdot \sum_{c \in C_i(q)} P(c|w) \cdot \left[\log P(w|c) - \log P(w) \right] && (6.22)\\
&= \mathbb{E}_{\mathcal{W}, c|C_i(q)} \left[\log P(\mathcal{W}|c) - \log P(\mathcal{W}) \right] && (6.23)
\end{aligned}
$$

6.1.2 KL Divergence Formulation

We now show that we can alternatively derive Eq. 6.16 from the standard KL divergence equation:

$$
\mathrm{KL}(P||Q) = \sum_{x \in X} \left[P(x) \cdot \log \frac{P(x)}{Q(x)} \right] \tag{6.24}
$$

Where P and Q represent the data distribution before and after receiving feedback, respectively. We convert this formulation to our terminology as follows:

$$
\max_{q \in Q} \mathbb{E}_{c|C_i(q)} \left[\mathrm{KL}(P(\mathcal{W}|c)||P(\mathcal{W})) \right] \tag{6.25}
$$

We now solve for the optimal query:

$$\mathbb{E}_{c|C_i(q)}\left[\mathrm{KL}(P(\mathcal{W}|c)||P(\mathcal{W}))\right] \tag{6.26}$$

$$= \mathbb{E}_{c|C_i(q)}\left[\sum_{w\in\mathcal{W}}\left[P(w|c)\cdot\log\frac{P(w|c)}{P(w)}\right]\right] \tag{6.27}$$

$$= \mathbb{E}_{c|C_i(q)}\left[\sum_{w\in\mathcal{W}}\left[P(w|c)\cdot\log\frac{P(c|w)}{P(c)}\right]\right] \tag{6.28}$$

$$= \sum_{c\in C_i(q)}\left[P(c)\sum_{w\in\mathcal{W}}\left[P(w|c)\cdot\log\frac{P(c|w)}{P(c)}\right]\right] \tag{6.29}$$

Which is equivalent to Eq. 6.12, and thus results in Eq. 6.16.

6.1.3 Probability Tensor Derivations

See Table 3.1 for all definitions of q, c, c^+, c^- for each interaction type. In the demonstration case, we define $\mathbf{P}$ as follows:

$$\mathbf{P}^{(\text{demo})}_{q,c,\omega} = \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} \tag{6.30}$$

$$= \frac{e^{\beta\cdot\phi(c_0^+)\cdot\omega}}{\sum_{t\in T} e^{\beta\cdot\phi(t)\cdot\omega}} \qquad (\text{since } |c^+|=1 \text{ and } c^+\cup c^- = T \text{ for demonstrations}) \tag{6.31}$$

$$= \frac{\mathbf{E^T}_{0,c_0^+,\omega}}{\sum_{t\in T}\mathbf{E^T}_{0,t,\omega}} \tag{6.32}$$

$$= \left[\mathbf{E^T}_0 \oslash \sum_{t\in T}\mathbf{E}_t\right]_{c,\omega} \qquad (\text{since there is a 1-1 correlation between } c \text{ and } c^+ \text{ in demos}) \tag{6.33}$$

where $\oslash$ represents an element-wise division of two matrices (i.e., $(\mathbf{A}\oslash\mathbf{B})_{ij} = \mathbf{A}_{ij}/\mathbf{B}_{ij}$).

In the preference case:

$$\mathbf{P}^{(\text{pref})}_{q,c,\omega} = \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} \tag{6.34}$$

$$= \frac{e^{\beta\cdot\phi(c_0^+)\cdot\omega}}{e^{\beta\cdot\phi(q_0)\cdot\omega} + e^{\beta\cdot\phi(q_1)\cdot\omega}} \qquad (\text{since } |c^+| = 1 \text{ and } c^- = q \setminus c^+ \text{ in preferences}) \tag{6.35}$$

$$\text{Since } c_0 \implies c^+ = \{q_0\} \text{ and } c_1 \implies c^+ = \{q_1\}: \tag{6.36}$$

$$= \left[\frac{e^{\beta\cdot\phi(q_0)\cdot\omega}}{e^{\beta\cdot\phi(q_0)\cdot\omega} + e^{\beta\cdot\phi(q_1)\cdot\omega}}, \frac{e^{\beta\cdot\phi(q_1)\cdot\omega}}{e^{\beta\cdot\phi(q_0)\cdot\omega} + e^{\beta\cdot\phi(q_1)\cdot\omega}}\right]_c \qquad (\text{where } c \in \{0,1\}) \tag{6.37}$$

$$= \left[\frac{\mathbf{E^T}_{q_0,q_1,\omega}}{[\mathbf{E}+\mathbf{E^T}]_{q_0,q_1,\omega}}, \frac{\mathbf{E}_{q_0,q_1,\omega}}{[\mathbf{E}+\mathbf{E^T}]_{q_0,q_1,\omega}}\right]_c \tag{6.38}$$

$$= \left[\left(\mathbf{E} \oslash (\mathbf{E}+\mathbf{E^T})\right)^{\mathbf{T}}, \mathbf{E} \oslash (\mathbf{E}+\mathbf{E^T})\right]_{c,q_0,q_1,\omega} \tag{6.39}$$

In the corrections case:

$$\mathbf{P}^{(\text{corr})}_{q,c,\omega} = \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} \tag{6.40}$$

$$= \frac{e^{\beta\cdot\phi(c_0^+)\cdot\omega}}{e^{\beta\cdot\phi(c_0^+)\cdot\omega} + e^{\beta\cdot\phi(c_0^-)\cdot\omega}} \qquad (\text{since } |c^+| = 1 \text{ and } |c^-| = 1) \tag{6.41}$$

$$= \frac{\mathbf{E^T}_{q,c,\omega}}{[\mathbf{E}+\mathbf{E^T}]_{q,c,\omega}} \quad (\text{due to 1-1 correlation between } q \text{ and } c^- \text{ and between } c \text{ and } c^+ \text{ in corrections}) \tag{6.42}$$

$$= \left[\mathbf{E^T} \oslash (\mathbf{E}+\mathbf{E^T})\right]_{q,c,\omega} \tag{6.43}$$

In the binary reward case, we compare the likelihood of the teacher demonstrating

q to the average likelihood of demonstrating any other trajectory in T:

$$\mathbf{P}^{(\text{bnry})}_{q,c,\omega} = \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} \tag{6.44}$$

$$\text{Since } c_0 \implies c^+ = T\setminus q, \quad c^- = q \tag{6.45}$$

$$\text{and } c_1 \implies c^+ = q, \quad c^- = T\setminus q\text{:} \tag{6.46}$$

$$= \frac{1}{\alpha}\left[\frac{1}{|T\setminus q|}\cdot\frac{\sum_{t\in T\setminus q} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in T} e^{\beta\cdot\phi(t)\cdot\omega}}, \frac{e^{\beta\cdot\phi(q)\cdot\omega}}{\sum_{t\in T} e^{\beta\cdot\phi(t)\cdot\omega}}\right]_c \qquad (\text{since } c\in\{0,1\} \text{ in binary rewards}) \tag{6.47}$$

$$= \left[\frac{1-\mathbf{P}^{(\text{demo})}_{0,q,\omega}}{\alpha\left(|T|-1\right)}, \frac{\mathbf{P}^{(\text{demo})}_{0,q,\omega}}{\alpha}\right]_c \quad (\text{where } \alpha \text{ is a normalization factor s.t. } \sum_c \mathbf{P}^{(\text{bnry})}_{q,c,\omega} = 1) \tag{6.48}$$

$$= \left[1-\frac{\mathbf{P}^{(\text{demo})}_{0,q,\omega}}{\alpha}, \frac{\mathbf{P}^{(\text{demo})}_{0,q,\omega}}{\alpha}\right]_c \qquad (\text{since } \sum_c \mathbf{P}^{(\text{bnry})}_{q,c,\omega} = 1) \tag{6.49}$$

$$= \left[1-\left(\mathbf{E^T}_0 \oslash \alpha \sum_{t\in T}\mathbf{E}_t\right), \mathbf{E^T}_0 \oslash \alpha \sum_{t\in T}\mathbf{E}_t\right]_{c,q,\omega} \tag{6.50}$$

where $\alpha = \frac{1-\mathbf{P}^{(\text{demo})}_{0,q,\omega}}{|T|-1} + \mathbf{P}^{(\text{demo})}_{0,q,\omega}$

6.1.4 Gradient Derivation

Our goal is to update the weight estimate such that it maximizes the likelihood of all feedback in $\mathbf{F}$:

$$\omega^* = \underset{\omega}{\text{argmax}} \prod_{c\in\mathbf{F}} P(c|\omega) \tag{6.51}$$

$$= \underset{\omega}{\text{argmax}} \prod_{c\in\mathbf{F}} \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} \tag{6.52}$$

We calculate the gradient over ω by differentiating over its log-likelihood given $\mathbf{F}$:

$$\ell(\omega) = \log \prod_{c \in \mathbf{F}} \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \tag{6.53}$$

$$= \sum_{c \in \mathbf{F}} \left[\log \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \right] \tag{6.54}$$

$$= \sum_{c \in \mathbf{F}} \left[\log \left(\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega} \right) - \log \left(\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega} \right) \right] \tag{6.55}$$

$$\frac{\partial \ell(\omega)}{\partial \omega_j} = \sum_{c \in \mathbf{F}} \left[\frac{\sum_{t \in c^+} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}} - \frac{\sum_{t \in c^+ \cup c^-} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \right] \tag{6.56}$$

Note that when c^+ contains a single trajectory (i.e., in all interaction types except for binary reward), this gradient simplifies to:

$$\frac{\partial \ell(\omega)}{\partial \omega_j} = \sum_{c \in \mathbf{F}} \left[\beta \cdot \phi_j(c_0^+) - \frac{\sum_{t \in c^+ \cup c^-} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \right] \tag{6.57}$$

6.1.5 Training Parameters

We enforce $\forall \omega \in \mathcal{W},\ ||\omega|| = 1$. We set a high convergence threshold (10^{-3}) when updating each weight sample in order to maintain sparsity within Ω (which becomes less sparse as $\mathbf{F}$ grows with more queries), and then fully converge (convergence threshold of 10^{-6}) for reporting the distance between ω^* and the weight estimate $\tilde{\omega}$ after each query. During gradient descent, we use a step size of 5x10^{-4} for all tasks except for the Pizza domain, where we use a step size of 10^{-4}.

6.2 INQUIRE: Evaluation Details

6.2.1 Domain Implementations

Domain #1: Parameter Estimation This task involves directly estimating a randomly-initialized, ground truth weight vector ω^* containing 8 parameters. This formulation represents a generic learning problem relevant to many robotics tasks,

such as learning the relative importance between task outcomes according to a user's preference. There is no "state" in this domain, and each "trajectory" consists of a single sample of the weight vector. As a result, we do not enable demonstration queries in this domain since the resulting feedback would be akin to directly providing ω^* to the algorithm. The feature representation ϕ of a sample returns the sample itself. Since $||\omega|| = ||\omega^*|| = 1$, the reward of any sampled weight vector directly reflects the cosine similarity between it and the ground truth vector ($r(\omega) = \omega \cdot \omega^* = \cos(\theta)$).

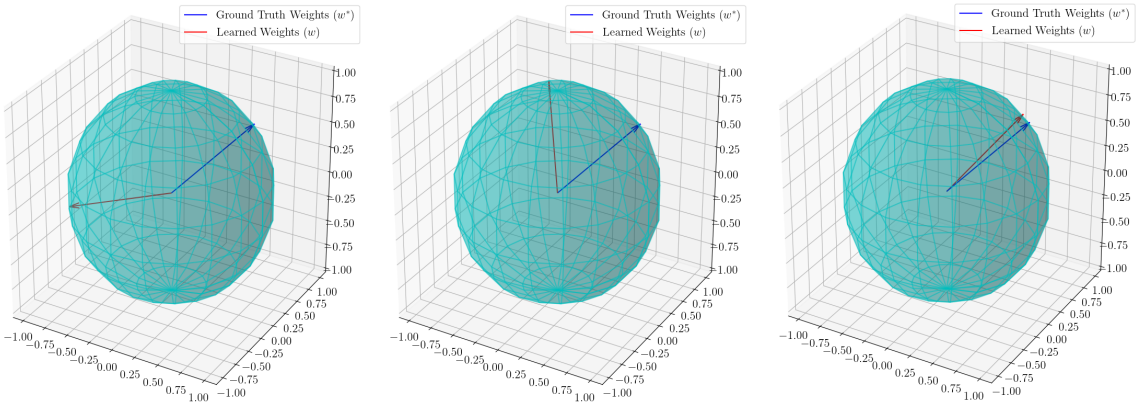


Figure 6.1: In the Parameter Estimation domain, the robot is tasked with estimating a high-dimensional ground truth weight vector w^* with its own set of learned weights w. To visualize this concept, a simpler case is illustrated above in three dimensions. All weight vectors (ground truth and learned weights) are unit vectors, and therefore lie on a unit sphere. Over time, the robot updates w by interacting with a teacher to gain a better estimate of w^*.

Domain #2: Linear Dynamical System We consider a simple Linear Dynamical System representing a robot that optimizes its controls according to a learned task objective. We represent the dynamics of the robot's state $\mathbf{s}$ as $d\mathbf{s}/dt = \mathbf{A}\mathbf{s}(t) + \mathbf{B}\mathbf{u}(t)$ by using dynamics matrix $\mathbf{A}$, input matrix $\mathbf{B}$, and random controls $\mathbf{u}$. An optimal control vector is one that results in a trajectory of states maximizing $\frac{1}{|T|}\sum_{s\in T}\phi(\mathbf{s})\cdot\omega^*$. We define the feature representation $\phi(\mathbf{s})$ of a state $\mathbf{s}$ as the concatenation of the element-wise, absolute difference between the robot's pose at time t and the goal pose, and the controls $\mathbf{u}(t)$. We experiment with an 8-dimensional feature-space (4 pose elements and 4 corresponding controls).

In a demonstration query, the oracle provides a trajectory (produced by simulating a series of controls) from the initial state that maximizes the total reward. In a

preference query, the algorithm proposes two trajectories and the oracle selects the option which yields higher reward. In a corrections query, the algorithm proposes a trajectory and the oracle returns a trajectory that maximizes the reward-to-similarity ratio. In a binary reward query, the algorithm proposes a trajectory and the oracle indicates whether that trajectory results in reward that exceeds the agent's internal threshold.

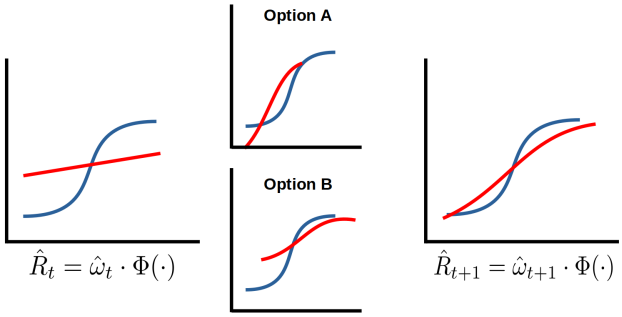


Figure 6.2: An example of a preference query in the Linear Dynamical System domain. At time $= t$, the learned reward function yields the red "trajectory." After posing a preference query (which consists of options A and B), the corresponding belief update yields the approximated reward function at time $= t + 1$.

Domain #3: Lunar Lander We define a ω^* that results in the agent efficiently moving from its start state to an upright pose on the landing pad. We use the same feature representation as in [43], consisting of four features: the lander's angle, velocity, distance from the landing pad, and final position with respect to the landing pad. We implement each query type in the same manner as in the Linear Dynamical System.

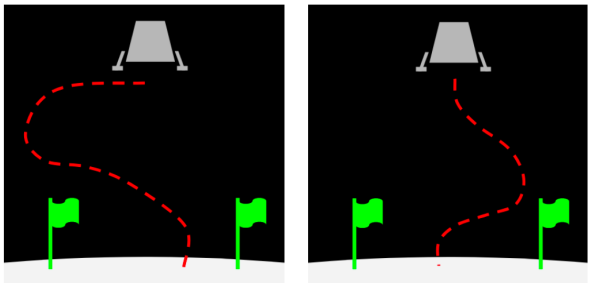


Figure 6.3: The Lunar Lander domain involves having the robot pilot a lunar lander to safely descend and arrive at a landing pad. The depicted preference query illustrates two different trajectories that may be taken by the lander to reach the destination.

Domain #4: Pizza Topping Placement We approximate a preference-learning task in which the robot learns to place toppings on only the left side of a pizza and with uniform spacing between them. We define each "trajectory" as the *next* action the robot should take from the current pizza state; thus, the trajectory is defined as the (x, y)-coordinate of the next topping to be placed. The feature representation consists of four features: the x and y position of the topping, its distance to its nearest-neighboring topping, and the difference between that distance and 4cm.

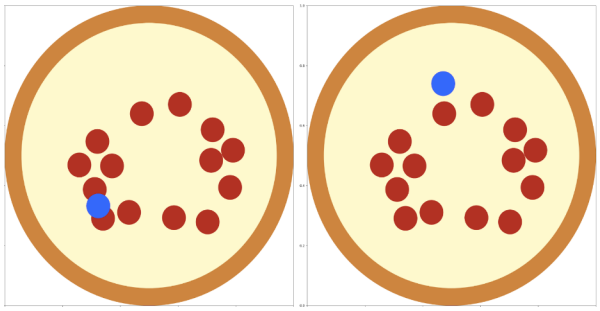


Figure 6.4: The task in the Pizza Topping Placement domain is to learn how to place toppings according to a human's reward over topping positions. In the depicted preference query, a human's choice indicates their preference for the "next" topping's position (choices represented in blue).

6.2.2 Oracle Implementation

When responding to a query, the oracle requires its own set of trajectory samples. Similar to INQUIRE, we derive this set by uniformly sampling N trajectories; however, the two sample sets are kept separate, and so we distinguish the oracle's trajectory set as T' (resampled for each query state).

Demonstration/Preferences The oracle returns the highest-reward trajectory (according to ω^*) from a uniformly-sampled trajectory set T' (for demonstrations) or from the pair of queried trajectories $C(q)$ (for preferences):

$$\text{Oracle}_{\text{demo}}(q) = \underset{t \in T'}{\text{argmax}} \left(\phi(t) \cdot \omega^*\right) \qquad \text{Oracle}_{\text{pref}}(q) = \underset{t \in C(q)}{\text{argmax}} \left(\phi(t) \cdot \omega^*\right) \tag{6.58}$$

Corrections The oracle produces T' by performing rejection sampling; it uniformly samples trajectories and accepts only those with a reward greater than or equal to

the queried trajectory q until T' contains N trajectories:

$$\forall t \in T', \phi(t) \cdot \omega^* \geq \phi(q) \cdot \omega^* \tag{6.59}$$

After producing this trajectory set, the oracle selects the trajectory with the highest ratio of reward-to-distance from the queried trajectory:

$$\text{Oracle}_{\text{corr}}(q) = \underset{t \in T'}{\text{argmax}} \frac{\Delta_r(q,t)}{\Delta_d(q,t)} \tag{6.60}$$

$$\Delta_r(q,t) = \min_{t' \in T'} \frac{\phi(t) \cdot \omega^* - \phi(q) \cdot \omega^*}{\phi(t') \cdot \omega^* - \phi(q) \cdot \omega^*} \qquad \Delta_d(q,t) = \min_{t' \in T'} \frac{e^{\delta(t,q)}}{e^{\delta(t',q)}} \tag{6.61}$$

The distance metric δ between two trajectories is domain-specific. In the Parameter Estimation domain, we define this as the angular distance between the two parameter vectors. In the Linear Dynamical System and Lunar Lander domains, we define δ as the normalized distance between the two trajectories' aligned x and y poses over time. We use the DTW-Python package [24] to align trajectories via Dynamic Time Warping and return their normalized distances. In the Pizza Topping Placement domain, we define δ as the Euclidean distance between two toppings.

Binary Reward The oracle produces T' by uniformly sampling N trajectories and produces a cumulative distribution R over ground-truth rewards for T'. It then selects a positive or negative reward indicating whether the agent's query q meets or exceeds a threshold percentile α:

$$R = \{\omega^* \cdot \phi(t), \forall t \in T'\} \qquad \text{Oracle}_{\text{bnry}}(q) = \begin{cases} + & R(\omega^* \cdot \phi(q)) \geq \alpha \\ - & \text{otherwise} \end{cases} \tag{6.62}$$

We set $\alpha = 0.75$ in our experiments.

6.2.3 Evaluation Procedure

Algorithm 4 Evaluation Procedure

Input: generate_query and update_weights methods according to algorithm being tested

Generate ground truth reward function ω^*
Generate 10 test states
Compute optimal trajectory $t_{\max}$ for each test case using ω^*
Compute least-optimal trajectory $t_{\min}$ for each test case using ω^*
for each of 10 runs **do**
    Generate 20 query states (if testing in the static condition, repeat the same state 20 times)
    for each of 20 queries **do**
        $s \leftarrow$ next query state
        $q^* \leftarrow$ generate_query$(s, \mathcal{I}, \mathbf{\Omega})$
        $\mathbf{F} \leftarrow \mathbf{F}+$ query_oracle(q^*)
        $\mathbf{\Omega} \leftarrow$ update_weights$(\mathbf{F})$
        $\tilde{\omega} \leftarrow$ mean(Ω)
        Record distance: $\frac{\arccos(\tilde{\omega} \cdot \omega^*)}{\pi}$
        for each of 10 test states **do**
            Compute optimal trajectory t from the test state according to $\tilde{\omega}$
            Record performance: $\frac{\phi(t) \cdot \omega^* - \phi(t_{\min}) \cdot \omega^*}{\phi(t_{\max}) \cdot \omega^* - \phi(t_{\min}) \cdot \omega^*}$
        end for
    end for
end for

6.3 AUC Figures

QUERIES vs DISTANCE Curve

Agent	Parameter Estimation (Static State)	Dynamical System (Static State)	Dynamical System (Changing State)	Lunar Lander (Static State)	Lunar Lander (Changing State)	Pizza Arrangement (Static State)	Pizza Arrangement (Changing State)	Across Tasks: Mean w^* Distance
DemPref	5.96	6.42	6.26	5.13	5.08	7.53	6.50	6.13
Binary-only	7.44	4.87	5.24	6.73	6.18	8.13	8.00	6.66
Corrections-only	3.01	4.43	4.01	4.02	3.80	4.29	4.17	3.96
Demo-only	n/a	4.26	**2.10**	3.35	1.91	5.09	3.99	3.45
Preferences-only	4.30	4.34	4.45	2.31	2.34	3.20	3.11	3.44
INQUIRE	**2.98**	**2.46**	2.59	**1.62**	**1.67**	**3.10**	**2.85**	**2.47**

Figure 6.5: AUC values for the distance plots in Figs 3.2-3.3. Darker cells indicate lower (better) values.

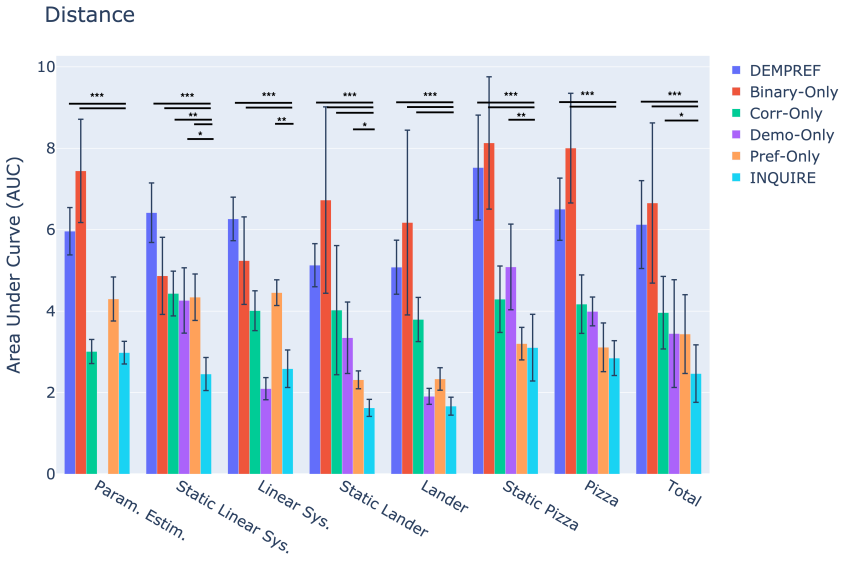


Figure 6.6: Visualizing Fig. 6.5, with statistical significance noted. (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$)

QUERIES vs PERFORMANCE Curve

Agent	Parameter Estimation (Static State)	Dynamical System (Static State)	Dynamical System (Changing State)	Lunar Lander (Static State)	Lunar Lander (Changing State)	Pizza Arrangement (Static State)	Pizza Arrangement (Changing State)	Across Tasks: Mean Performance
DemPref	13.95	14.69	14.47	17.22	17.37	14.03	15.53	15.32
Binary-only	15.01	16.54	18.37	16.85	17.37	14.29	14.65	16.15
Corrections-only	19.14	16.53	17.19	18.02	18.33	18.49	18.56	18.04
Demo-only	n/a	16.76	**18.15**	18.61	19.32	18.86	**20.10**	18.63
Preferences-only	17.74	16.55	16.74	18.63	19.05	18.77	18.89	18.05
INQUIRE	**19.15**	**17.81**	17.83	**18.86**	**19.33**	**19.88**	19.79	**18.95**

Figure 6.7: AUC values for the performance plots in Figs 3.2-3.3. Darker cells indicate higher (better) values.

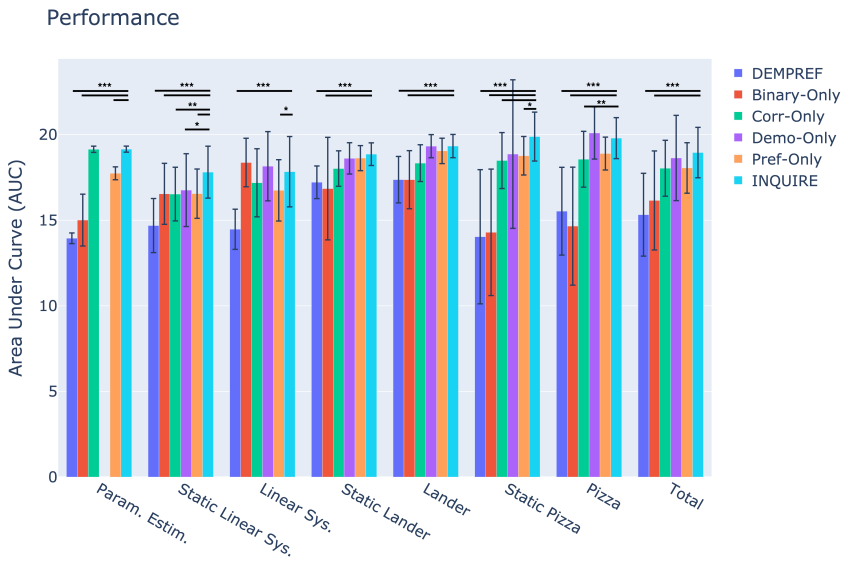


Figure 6.8: Visualizing Fig. 6.7, with statistical significance noted. (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$)

COST vs DISTANCE Curve

Agent	Parameter Estimation (Static State)	Dynamical System (Static State)	Dynamical System (Changing State)	Lunar Lander (Static State)	Lunar Lander (Changing State)	Pizza Arrangement (Static State)	Pizza Arrangement (Changing State)	Across Tasks: Mean Cost/Distance
DemPref	68.26	71.59	70.52	59.16	58.41	82.21	74.20	69.19
Binary-only	64.15	43.18	44.85	61.30	49.84	78.62	79.20	60.16
Corrections-only	**36.39**	51.68	46.02	41.91	42.57	45.13	46.63	44.33
Demo-only	n/a	43.99	**27.94**	37.09	26.07	50.82	42.26	38.03
Preferences-only	42.41	42.73	43.92	**22.90**	**23.18**	**31.81**	**31.01**	33.99
INQUIRE	37.68	**36.39**	35.92	22.98	23.71	33.99	36.48	**32.45**

Figure 6.9: AUC values for the cost plots in Figs 3.2-3.3. Darker cells indicate lower (better) values.

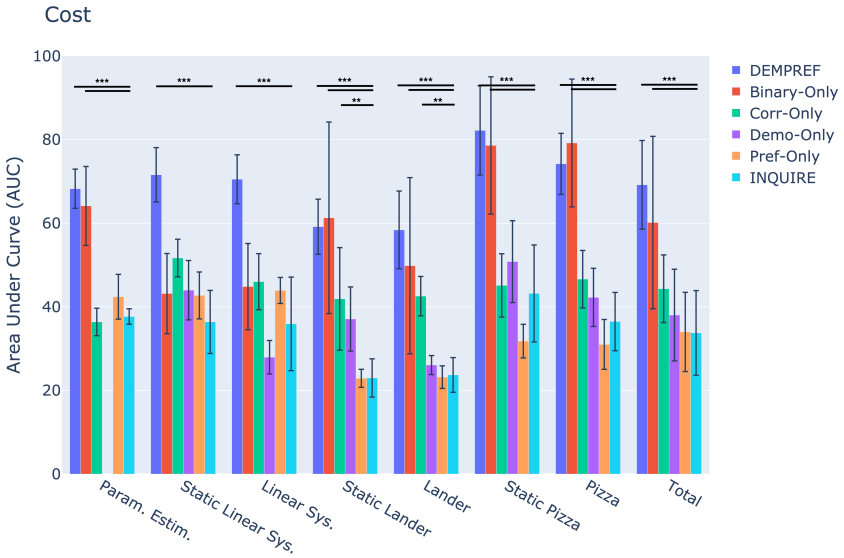


Figure 6.10: Visualizing Fig 6.9, with statistical significance noted. (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$)

6.4 Exploring Reward Functions: Filtering Reward Function Beliefs

To undertake this project's learning task, we designed the agent's belief model of the reward function as a particle filter. Particle filters are probabilistic state models which perform Bayesian inference through sequential, online updates to a belief space [47]. Commonly used in robotics for state estimation, they've also been used to approximate the motions of articulated objects and to track the poses of manipulated objects [25, 36].

To formalize our particle filter representation, let:

- $\boldsymbol{p} = \hat{\boldsymbol{\theta}}$ be a d-dimensional tuple of parameters that represents a discrete, *hypothesized* reward function $\hat{R}(\cdot)$;
- $\boldsymbol{p}^{(k)} \in \mathcal{P}$ be the k-th particle in the particle set $\mathcal{P}$; and
- $m_k \in \boldsymbol{m}$ be the k-th particle's importance weight, with $m_k \in (0, 1)$ and $\sum_{i=1}^{d} m_i = 1$.

Intuitively, a particle filter's importance weights $\boldsymbol{m}$ represent a probability dis-

tribution over the filter's particle set $\mathcal{P}$. Thus, each importance weight m_k is an approximation of the probability that $\boldsymbol{p}^{(k)}$ represents the "true" reward function of the teacher. The filter's primary function is to allocate greater weight to the more likely particles by evaluating an observation and then performing a corresponding Bayesian update over its importance weights.

It's critical for the reader to grasp our formulation's distinction between *importance* weights $\boldsymbol{m}$ and *parameter* weights $\boldsymbol{\omega}$. Each particle has its own importance weight m_k that represents how strongly the robot believes that particle is the human's true reward function. Distinct from this importance weight is its set of parameter weights $\hat{\boldsymbol{\omega}}^{(k)}$, which directly contribute to the value yielded by the k-th particle's reward function $R_k(\cdot)$.

Each filtering step produces a distribution of particle importance weights $\boldsymbol{m}_t$ conditioned on the teacher's response $\boldsymbol{c}^*$ to query $\boldsymbol{q}_t$ (see Equation 4.5). In our formulation, the prior belief $P(\boldsymbol{\theta})$ is approximated by the importance weights $\boldsymbol{m}_t$, and the likelihood term depends upon the interaction type chosen by the robot for query $\boldsymbol{q}_t$. See Table 4.1 for descriptions of the likelihood representations for the different query interaction types.

The filtering process is depicted in Figure 4.1. First, the learner presents a query to the teacher through the process described in Section 4.3. After receiving a response $\boldsymbol{c}^*$ to query $\boldsymbol{q}_t$, the agent updates its belief using its prior and the likelihood term $P(\boldsymbol{c}^*|\boldsymbol{q}_t, \boldsymbol{m}_t)$ as in Equation 4.5. Finally, the filter resamples particles from the posterior $P(\boldsymbol{\theta}_t; \boldsymbol{c})$ to get the new approximation to the prior belief $\boldsymbol{m}_{t+1}$. After each such interaction instance, the learner's particle set should filter out unlikely particles and continue homing in on the teacher's reward function R^*.

Bibliography

[1] Andrea Bajcsy, Dylan P. Losey, Marcia K. O'Malley, and Anca D. Dragan. Learning from Physical Human Corrections, One Feature at a Time. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 141–149, Chicago IL USA, February 2018. ACM. ISBN 978-1-4503-4953-6. doi: 10.1145/3171221.3171267. URL https://dl.acm.org/doi/10.1145/3171221.3171267. ZSCC: 0000059. 1, 2.3, 3

[2] Chandrayee Basu, Mukesh Singhal, and Anca D. Dragan. Learning from Richer Human Guidance: Augmenting Comparison-Based Learning with Feature Queries. *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 132–140, February 2018. doi: 10.1145/3171221.3171284. URL http://arxiv.org/abs/1802.01604. ZSCC: 0000034 arXiv: 1802.01604. 1

[3] Chandrayee Basu, Mukesh Singhal, and Anca D. Dragan. Learning from Richer Human Guidance: Augmenting Comparison-Based Learning with Feature Queries. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 132–140, Chicago IL USA, February 2018. ACM. ISBN 978-1-4503-4953-6. doi: 10.1145/3171221.3171284. URL https://dl.acm.org/doi/10.1145/3171221.3171284. ZSCC: 0000034. 2.1

[4] Andreea Bobu, Dexter R. R. Scobee, Jaime F. Fisac, S. Shankar Sastry, and Anca D. Dragan. LESS is More: Rethinking Probabilistic Models of Human Behavior. In *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pages 429–437, March 2020. doi: 10.1145/3319502.3374811. URL http://arxiv.org/abs/2001.04465. arXiv:2001.04465 [cs, stat]. 2.1

[5] Andreea Bobu, Chris Paxton, Wei Yang, Balakumar Sundaralingam, Yu-Wei Chao, Maya Cakmak, and Dieter Fox. Learning Perceptual Concepts by Bootstrapping from Human Queries, July 2022. URL http://arxiv.org/abs/2111.05251. arXiv:2111.05251 [cs]. 2.3

[6] W. Bradley Knox and Peter Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In *2008 7th IEEE International Conference on*

Development and Learning, pages 292–297, August 2008. doi: 10.1109/DEVLRN.2008.4640845. ISSN: 2161-9476. 1

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, June 2016. URL http://arxiv.org/abs/1606.01540. arXiv:1606.01540 [cs]. 3.2, 4.4.1

[8] Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations. *arXiv:1904.06387 [cs, stat]*, July 2019. URL http://arxiv.org/abs/1904.06387. arXiv: 1904.06387. 2.3

[9] Kalesha Bullard, Andrea L. Thomaz, and Sonia Chernova. Towards Intelligent Arbitration of Diverse Active Learning Queries. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6049–6056, Madrid, October 2018. IEEE. ISBN 978-1-5386-8094-0. doi: 10.1109/IROS.2018.8594279. URL https://ieeexplore.ieee.org/document/8594279/. 1, 2.3, 3

[10] Erdem Bıyık, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. Asking Easy Questions: A User-Friendly Approach to Active Reward Learning. *arXiv:1910.04365 [cs]*, October 2019. URL http://arxiv.org/abs/1910.04365. ZSCC: 0000042 arXiv: 1910.04365. 1, 2.3, 3, 3.1.1, 3.2, 3.2.2, 4.3, 6.1.1

[11] Erdem Bıyık, Nicolas Huynh, Mykel J. Kochenderfer, and Dorsa Sadigh. Active Preference-Based Gaussian Process Regression for Reward Learning. *arXiv:2005.02575 [cs]*, June 2020. URL http://arxiv.org/abs/2005.02575. ZSCC: 0000025 arXiv: 2005.02575. 2.1

[12] Erdem Bıyık, Dylan P. Losey, Malayandi Palan, Nicholas C. Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning Reward Functions from Diverse Sources of Human Feedback: Optimally Integrating Demonstrations and Preferences. *arXiv:2006.14091 [cs]*, August 2021. URL http://arxiv.org/abs/2006.14091. ZSCC: NoCitationData[s1] arXiv: 2006.14091. 1, 3

[13] Erdem Bıyık, Aditi Talati, and Dorsa Sadigh. APReL: A Library for Active Preference-based Reward Learning Algorithms, January 2022. URL http://arxiv.org/abs/2108.07259. arXiv:2108.07259 [cs]. 2.3

[14] Carlos Celemin, Rodrigo Pérez-Dattari, Eugenio Chisari, Giovanni Franzese, Leandro de Souza Rosa, Ravi Prakash, Zlatan Ajanović, Marta Ferraz, Abhinav Valada, and Jens Kober. Interactive Imitation Learning in Robotics: A Survey, October 2022. URL http://arxiv.org/abs/2211.00600. arXiv:2211.00600 [cs]. 1, 3

[15] Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from Suboptimal Demonstration via Self-Supervised Reward Regression. In *Proceedings of the 2020*

Conference on Robot Learning, pages 1262–1277. PMLR, October 2021. URL https://proceedings.mlr.press/v155/chen21b.html. ISSN: 2640-3498. 2.3

[16] Letian Chen, Sravan Jayanthi, Rohan Paleja, Daniel Martin, Viacheslav Zakharov, and Matthew Gombolay. Fast Lifelong Adaptive Inverse Reinforcement Learning from Demonstrations, September 2022. URL http://arxiv.org/abs/2209.11908. arXiv:2209.11908 [cs]. 2.3

[17] Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, February 2023. URL http://arxiv.org/abs/1706.03741. arXiv:1706.03741 [cs, stat]. 4

[18] Yuchen Cui, Pallavi Koppol, Henny Admoni, Scott Niekum, Reid Simmons, Aaron Steinfeld, and Tesca Fitzgerald. ***Understanding the Relationship between Interactions and Outcomes in Human-in-the-Loop Machine Learning. page 10. ZSCC: 0000000. 1, 3, 3.1, 3.3

[19] Yuchen Cui, Pallavi Koppol, Henny Admoni, Scott Niekum, Reid Simmons, Aaron Steinfeld, and Tesca Fitzgerald. Understanding the Relationship between Interactions and Outcomes in Human-in-the-Loop Machine Learning. volume 5, pages 4382–4391, August 2021. doi: 10.24963/ijcai.2021/599. URL https://www.ijcai.org/proceedings/2021/599. ISSN: 1045-0823. 2.3

[20] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active Reward Learning. In *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation, July 2014. ISBN 978-0-9923747-0-9. doi: 10.15607/RSS.2014.X.031. URL http://www.roboticsproceedings.org/rss10/p31.pdf. 2.3

[21] Tesca Fitzgerald, Elaine Short, Ashok Goel, and Andrea Thomaz. Human-guided Trajectory Adaptation for Tool Transfer. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 1350–1358, Richland, SC, May 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-6309-9. 1, 2.3, 3

[22] Tesca Fitzgerald, Pallavi Koppol, Patrick Callaghan, Russell Quinlan Jun Hei Wong, Reid Simmons, Oliver Kroemer, and Henny Admoni. INQUIRE: INteractive Querying for User-aware Informative REasoning. November 2022. URL https://openreview.net/forum?id=3CQ3Vt0v99. 1, 4.3

[23] Gaurav R. Ghosal, Matthew Zurek, Daniel S. Brown, and Anca D. Dragan. The Effect of Modeling Human Rationality Level on Learning Rewards from Multiple Feedback Types, August 2022. URL http://arxiv.org/abs/2208.10687. arXiv:2208.10687 [cs]. 2.2

[24] Toni Giorgino. Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. *Journal of Statistical Software*, 31:1–24, August 2009. ISSN 1548-7660. doi: 10.18637/jss.v031.i07. URL https://doi.org/10.18637/

jss.v031.i07. 4.4.2, 6.2.2

[25] Karol Hausman, Scott Niekum, Sarah Osentoski, and Gaurav S. Sukhatme. Active articulation model estimation through interactive perception. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3305–3312, Seattle, WA, USA, May 2015. IEEE. ISBN 978-1-4799-6923-4. doi: 10.1109/ICRA.2015.7139655. URL http://ieeexplore.ieee.org/document/7139655/. ZSCC: 0000063. 6.4

[26] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys*, 50(2):21:1–21:35, April 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL https://dl.acm.org/doi/10.1145/3054912. 1, 2.3, 3

[27] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari, November 2018. URL http://arxiv.org/abs/1811.06521. arXiv:1811.06521 [cs, stat]. 2.3

[28] Hong Jun Jeon, Smitha Milli, and Anca Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 4415–4426. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2f10c1578a0706e06b6d7db6f0b4a6af-Abstract.html. 2.3

[29] Aleksandra Kalinowska, Ahalya Prabhakar, Kathleen Fitzsimons, and Todd Murphey. Ergodic imitation: Learning from what to do and what not to do. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3648–3654, May 2021. doi: 10.1109/ICRA48506.2021.9561746. ISSN: 2577-087X. 2.3

[30] Pallavi Koppol, Henny Admoni, and Reid Simmons. Interaction Considerations in Learning from Humans. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 283–291, Montreal, Canada, August 2021. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-9-6. doi: 10.24963/ijcai.2021/40. URL https://www.ijcai.org/proceedings/2021/40. ZSCC: 0000000. 1, 3, 4.1

[31] Oliver Kroemer and Gaurav S. Sukhatme. Learning spatial preconditions of manipulation skills using random forests. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 676–683, Cancun, Mexico, November 2016. IEEE. ISBN 978-1-5090-4718-5. doi: 10.1109/HUMANOIDS.2016.7803347. URL http://ieeexplore.ieee.org/document/7803347/. 2.3

[32] Cassidy Laidlaw and Anca Dragan. The Boltzmann Policy Distribution: Accounting for Systematic Suboptimality in Human Models. page 21, 2022. ZSCC:

NoCitationData[s0]. 2.1

[33] Michael S. Lee, Henny Admoni, and Reid Simmons. Machine Teaching for Human Inverse Reinforcement Learning. *Frontiers in Robotics and AI*, 8, 2021. ISSN 2296-9144. URL https://www.frontiersin.org/articles/10.3389/frobt.2021.693050. 3.4

[34] Michael S. Lee, Henny Admoni, and Reid Simmons. Reasoning about Counterfactuals to Improve Human Inverse Reinforcement Learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9140–9147, Kyoto, Japan, October 2022. IEEE. ISBN 978-1-66547-927-1. doi: 10.1109/IROS47612.2022.9982062. URL https://ieeexplore.ieee.org/document/9982062/. 2.2

[35] Kejun Li, Maegan Tucker, Erdem Bıyık, Ellen Novoseller, Joel W. Burdick, Yanan Sui, Dorsa Sadigh, Yisong Yue, and Aaron D. Ames. ROIAL: Region of Interest Active Learning for Characterizing Exoskeleton Gait Preference Landscapes. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3218, May 2021. doi: 10.1109/ICRA48506.2021.9560840. URL http://arxiv.org/abs/2011.04812. arXiv:2011.04812 [cs]. 2.3

[36] Jacky Liang, Ankur Handa, Karl Van Wyk, Viktor Makoviychuk, Oliver Kroemer, and Dieter Fox. In-Hand Object Pose Tracking via Contact Feedback and GPU-Accelerated Robotic Simulation, November 2020. URL http://arxiv.org/abs/2002.12160. arXiv:2002.12160 [cs]. 6.4

[37] Jacky Liang, Saumya Saxena, and Oliver Kroemer. Learning Active Task-Oriented Exploration Policies for Bridging the Sim-to-Real Gap. *arXiv:2006.01952 [cs]*, November 2020. URL http://arxiv.org/abs/2006.01952. ZSCC: 0000005 arXiv: 2006.01952. 2.3

[38] Dylan P. Losey and Marcia K. O'Malley. Learning the Correct Robot Trajectory in Real-Time from Physical Human Interactions. *ACM Transactions on Human-Robot Interaction*, 9(1):1:1–1:19, December 2019. doi: 10.1145/3354139. URL https://doi.org/10.1145/3354139. ZSCC: 0000010. 2.3

[39] Dylan P. Losey, Andrea Bajcsy, Marcia K. O'Malley, and Anca D. Dragan. Physical Interaction as Communication: Learning Robot Objectives Online from Human Corrections, July 2021. URL http://arxiv.org/abs/2107.02349. arXiv:2107.02349 [cs, eess]. 2.3

[40] Shaunak A Mehta and Dylan P Losey. Unified Learning from Demonstrations, Corrections, and Preferences during Physical Human-Robot Interaction. page 21. 2.1

[41] Andrew Y. Ng. Algorithms for Inverse Reinforcement Learning. 1, 2.3, 3

[42] Gennaro Notomista, Siddharth Mayya, Mario Selvaggio, Maria Santos, and

Cristian Secchi. A Set-Theoretic Approach to Multi-Task Execution and Prioritization. *arXiv:2003.02968 [cs, eess, math]*, March 2020. URL http://arxiv.org/abs/2003.02968. ZSCC: 0000004 arXiv: 2003.02968. 4

[43] Malayandi Palan, Nicholas C. Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning Reward Functions by Integrating Human Demonstrations and Preferences. *arXiv:1906.08928 [cs]*, June 2019. URL http://arxiv.org/abs/1906.08928. ZSCC: 0000054 arXiv: 1906.08928. 1, 2.3, 3, 3.2.2, 4.4.4, 6.2.1

[44] Deepak Ramachandran. Bayesian Inverse Reinforcement Learning. page 6. 3.1.1

[45] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. Active Preference-Based Learning of Reward Functions. In *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, July 2017. ISBN 978-0-9923747-3-0. doi: 10.15607/RSS.2017.XIII.053. URL http://www.roboticsproceedings.org/rss13/p53.pdf. ZSCC: 0000180. 1, 2.3

[46] Mariah L. Schrum, Erin Hedlund-Botti, and Matthew Gombolay. Reciprocal MIND MELD: Improving Learning From Demonstration via Personalized, Reciprocal Teaching. November 2022. URL https://openreview.net/forum?id=f_XmiyZcsjL. 2.2

[47] Maarten Speekenbrink. A tutorial on particle filters. ZSCC: 0000053. 6.4

[48] Halit Bener Suay and Tim Brys. Learning from Demonstration for Shaping through Inverse Reinforcement Learning. 2.1

[49] Eugene Valassakis, Georgios Papagiannis, Norman Di Palo, and Edward Johns. Demonstrate Once, Imitate Immediately (DOME): Learning Visual Servoing for One-Shot Imitation Learning, July 2022. URL http://arxiv.org/abs/2204.02863. arXiv:2204.02863 [cs]. 2.3

[50] Nils Wilde, Alexandru Blidaru, Stephen L. Smith, and Dana Kulić. Improving User Specifications for Robot Behavior through Active Preference Learning: Framework and Evaluation. *The International Journal of Robotics Research*, 39(6):651–667, May 2020. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364920910802. URL http://arxiv.org/abs/1907.10412. ZSCC: 0000013 arXiv: 1907.10412. 2.3

[51] Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A Survey of Preference-Based Reinforcement Learning Methods. 1, 2.3, 3

[52] Y Xiang, D. Y Sun, W Fan, and X. G Gong. Generalized simulated annealing algorithm and its application to the Thomson model. *Physics Letters A*, 233(3):216–220, August 1997. ISSN 0375-9601. doi: 10.1016/S0375-9601(97)00474-X. URL https://www.sciencedirect.com/science/article/pii/S037596019700474X. 4.4.2

[53] Michelle Zhao, Reid Simmons, and Henny Admoni. Coordination With Humans Via Strategy Matching. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9116–9123, Kyoto, Japan, October 2022. IEEE. ISBN 978-1-66547-927-1. doi: 10.1109/IROS47612.2022.9982277. URL https://ieeexplore.ieee.org/document/9982277/. 2.2