INQUIRE: INteractive Querying for User-aware Informative REasoning

Tesca Fitzgerald[*,1,2] **Pallavi Koppol**[2] **Patrick Callaghan**[2] **Russell Q. Wong**[2]
Reid Simmons[2] **Oliver Kroemer**[2] **Henny Admoni**[2]
[1] Yale University [2] Carnegie Mellon University
tesca.fitzgerald@yale.edu,
{pkoppol, pcallagh, rqwong, rsimmons, okroemer, hadmoni}@andrew.cmu.edu

Abstract: Research on Interactive Robot Learning has yielded several modalities for querying a human for training data, including demonstrations, preferences, and corrections. While prior work in this space has focused on optimizing the robot's queries within each interaction type, there has been little work on optimizing over the selection of the interaction type itself. We present INQUIRE, the first algorithm to implement and optimize over a generalized representation of information gain across multiple interaction types. Our evaluations show that INQUIRE can dynamically optimize its interaction type (and respective optimal query) based on its current learning status and the robot's state in the world, resulting in more robust performance across tasks in comparison to state-of-the-art baseline methods. Additionally, INQUIRE allows for customizable cost metrics to bias its selection of interaction types, enabling this algorithm to be tailored to a robot's particular deployment domain and formulate cost-aware, informative queries.

Keywords: Active Learning, Learning from Demonstration, Human-Robot Interaction

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 Interactive Robot Learning has yielded many effective methods for obtaining this 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 [1, 2]. In a *preference* query, the teacher selects one trajectory from a set of candidates proposed by the robot [3, 4]. In response to a single trajectory proposed by the robot, the teacher can provide a *correction* [5, 6] or simply a *binary reward* [7].

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 [8]. 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, 10, 11]), 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) [12].

Our work 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 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.

2 Related Works

Learning from Demonstration involves interpreting human feedback as training data using methods such as imitation learning [1] or inverse reinforcement learning [2]. However, there are many other forms that human feedback can take, including preferences [3, 4], labels [13], and corrections [6, 5]. These approaches optimize queries *within* a single interaction type, typically by maximizing volume removal [14] or information gain resulting from the teacher's response to the query [3]. Other approaches such as min-max regret optimization have also been explored [15]. Prior work has also investigated the use of fixed strategies for selecting interaction types; for example, requesting a fixed number of demonstrations before requesting preferences for the remaining queries [9, 16]. [11] 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 [17], and using both demonstrations and rankings [18]. Furthermore, [19] presents a software library for combining different preference feedback types and demonstrations.

Several attempts have been made to impose a unifying and consistent framework across multiple interaction types. [20] describes interactions in terms of the explicit and implicit information they convey; [8] surveys how different interaction types affect a teacher's ability to provide informative feedback. Our work similarly contributes a perspective on the unifying and differentiating features of interaction types: we propose a generalized framework for computing information gain across multiple interaction types. We focus on four interaction types corresponding to the archetypes (*Showing, Categorizing, Sorting*, and *Evaluating*) identified in [12], and empirically show the effects of dynamically selecting interaction types in robot learning.

3 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 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 [8], 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 $C(q) = \{t_0, t_1\}$.

Table 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	$t \in T$	$\{0, 1\}$	$c = 0 \implies c^+ : T \setminus q \quad c^- : q$ $c = 1 \implies c^+ : q \quad c^- : T \setminus q$

```
Algorithm 1 INQUIRE - Overview
Input: Set of query states $S$
Parameters: $K$ (# of queries), $\mathcal{I}$ (interaction types)
Output: Weight vector $\omega^*$

1: $\mathbf{F} \leftarrow \{\}$
2: $\mathbf{\Omega} \leftarrow M$ random initial weight vectors
3: for $K$ iterations do
4:     $s \leftarrow$ next query state in $S$
5:     $q_i^* \leftarrow$ generate_query($s, \mathcal{I}, \mathbf{\Omega}$)  (Alg. 2)

6:     $\mathbf{F} \leftarrow \mathbf{F} \cup \{$query_teacher($q_i^*$)$\}$
7:     $\mathbf{\Omega} \leftarrow$ update_weights($\mathbf{F}$)
8: $\omega^* \leftarrow$ mean($\mathbf{\Omega}$)
9: return $\omega^*$
```

```
Algorithm 2 INQUIRE - Generate Query
Input: $s$ (state), $\mathcal{I}$ (interaction types), $\mathbf{\Omega}$ (weight samples)
Output: Query $q^*$
1: $\mathbf{T} \leftarrow$ uniformly_sample_trajectories($s$)
2: Compute $\mathbf{E} : \{\mathbf{E}_{t,t',\omega}, \forall t, t' \in \mathbf{T}, \omega \in \mathbf{\Omega}\}$    (Eq. 4)
3: for each interaction type $i \in \mathcal{I}$ do
4:     $\mathbf{Q} \leftarrow Q_i(s)$    (See Table 1)
5:     $\mathbf{C} \leftarrow \{C_i(q), \forall q \in \mathbf{Q}\}$    (See Table 1)
6:     Compute info gain matrix $\mathbf{G^{(i)}}$ from $\mathbf{E}$    (Eq. 9)
7:     $q \leftarrow \arg\max_{q'} \sum_{c \in \mathbf{C}_{q'}, \omega \in \mathbf{\Omega}} \mathbf{G}^{(i)}_{q',c,\omega}$
8:     $g \leftarrow \frac{1}{\log(\lambda_i)} \sum_{c \in \mathbf{C}_q, \omega \in \mathbf{\Omega}} \mathbf{G}^{(i)}_{q,c,\omega}$
9:     if information gain $g > g^*$ then
10:        $g^* \leftarrow g$
11:        $q^* \leftarrow q$    {Store query with highest info. gain}
12: return $q^*$
```

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

$$q_i^* = \arg\max_{q \in Q_i(s)} \mathbb{E}_{c|C_i(q)} \left[\text{IG}(\mathcal{W} \mid c)\right] \tag{1}$$

$$= \arg\max_{q \in Q_i(s)} \sum_{c \in C_i(q)} \sum_{w \in \Omega} \left[P(c|w) \cdot \log \frac{M \cdot P(c|w)}{\sum_{w' \in \Omega} P(c|w')} \right] \tag{2}$$

where Ω contains M samples of the distribution $\mathcal{W}$. The expansion from Eq. 1 to 2 follows the derivation presented in [3]; see Appendix A.1 for intermediate steps. We adopt the commonly-used Boltzmann-rational equation to define $P(c|\omega)$:

$$P(c|\omega) = \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \tag{3}$$

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.[1] Note that Eq. 3 reduces to Bayesian Inverse Reinforcement Learning [21] 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. 2, we reformulate it as a series of operations over a $|Q|$ x $|C|$ x $|\Omega|$ probability tensor $\mathbf{P}$, where $\mathbf{P}_{q,c,\omega}$ 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:

$$\mathbf{E}_{t,t',\omega} = e^{\beta \cdot \phi(t') \cdot \omega} \qquad \Longrightarrow \qquad \left[\mathbf{E} + \mathbf{E^T}\right]_{t,t',\omega} = e^{\beta \cdot \phi(t') \cdot \omega} + e^{\beta \cdot \phi(t) \cdot \omega} \tag{4}$$

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:

$$\mathbf{P}^{(\text{demo})}_{q,c,\omega} = \left[\mathbf{E}_0 \oslash \sum_{t \in T} \mathbf{E^T}_t\right]_{c,\omega} \qquad \text{(since } |Q| = 1 \text{ for demonstrations)} \tag{5}$$

$$\mathbf{P}^{(\text{pref})}_{q,c,\omega} = \left[\left(\mathbf{E} \oslash (\mathbf{E} + \mathbf{E^T})\right)^{\mathbf{T}}, \mathbf{E} \oslash (\mathbf{E} + \mathbf{E^T})\right]_{c,q_0,q_1,\omega} \qquad (c \in \{0,1\} \text{ for prefs.}) \tag{6}$$

$$\mathbf{P}^{(\text{corr})}_{q,c,\omega} = \left[\mathbf{E} \oslash (\mathbf{E} + \mathbf{E^T})\right]_{q,c,\omega} \tag{7}$$

$$\mathbf{P}^{(\text{bnry})}_{q,c,\omega} = \left[1 - \left(\mathbf{E}_0 \oslash \alpha \sum_{t \in T} \mathbf{E^T}_t\right), \mathbf{E}_0 \oslash \alpha \sum_{t \in T} \mathbf{E^T}_t\right]_{c,q,\omega} \qquad (c \in \{0,1\} \text{ for binary rewards}) \tag{8}$$

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 A.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 $\mathbf{G}^{(i)}$:

$$\mathbf{G}^{(i)}_{q,c,\omega} = \mathbf{P}^{(i)}_{q,c,\omega} \cdot \log\left(\frac{M \cdot \mathbf{P}^{(i)}_{q,c,\omega}}{\sum_{\omega' \in \Omega} \mathbf{P}^{(i)}_{q,c,\omega'}}\right) \qquad q_i^* = \arg\max_q \sum_{c,\omega} \mathbf{G}^{(\text{i})}_{q,c,\omega} \tag{9}$$

We then solve for the optimal interaction type itself. To perform a *cost-weighted* optimization, with the aim of optimizing over both interaction cost and informativeness, 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.[2] To perform an *unweighted* optimization and maximize solely over the informativeness of each query, let λ_i be a constant value over all interaction types $i \in \mathcal{I}$.

$$i^* = \arg\max_{i \in \mathcal{I}} \frac{1}{\log(\lambda_i)} \sum_{c,\omega} \mathbf{G}^{(\text{i})}_{q_i^*,c,\omega} \tag{10}$$

We summarize this process in Alg. 2.

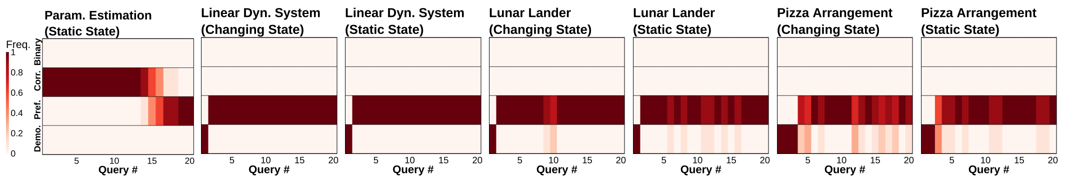
3.2 Update Weights from Feedback

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

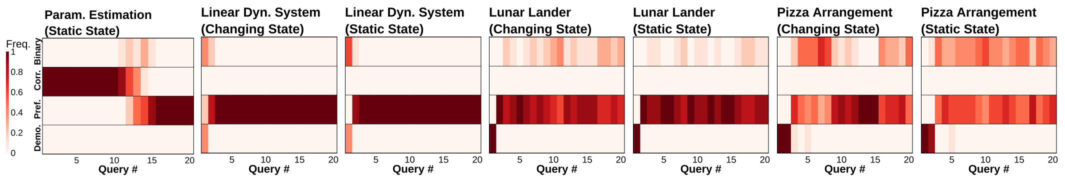
$$\omega^* = \arg\max_\omega \prod_{c \in \mathbf{F}} P(c|\omega) = \arg\max_\omega \prod_{c \in \mathbf{F}} \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \tag{11}$$

[1]See Appendix B.1 for each domain's definition of ϕ.

[2]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.



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



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

Figure 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).

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

$$\frac{\partial \ell(\omega)}{\partial \omega_j} = \sum_{c \in \mathbf{F}} \left[\frac{\sum_{t \in c^+} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}} - \frac{\sum_{t \in c^+ \cup c^-} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \right] \tag{12}$$

$$= \sum_{c \in \mathbf{F}} \left[\beta \cdot \phi_j(c_0^+) - \frac{\sum_{t \in c^+ \cup c^-} \beta \cdot \phi_j(t) \cdot e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} \right] \qquad \text{(iff } |c^+| = 1) \tag{13}$$

See Appendix A.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 $\omega \in \Omega$.

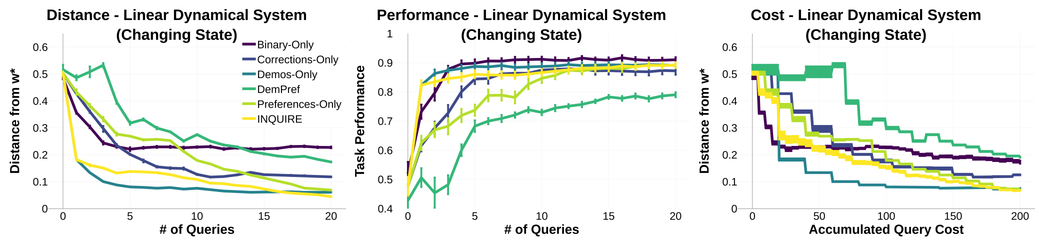
4 Results

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 (corrections), and returning whether a query meets or exceeds a reward threshold (binary reward). Implementation details can be found in Appendix B.2.

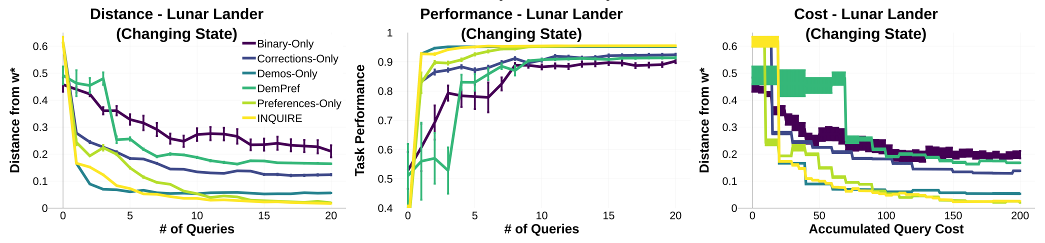
The **Parameter Estimation** domain involves directly estimating a randomly-initialized, ground truth weight vector ω^* containing 8 parameters. The **Linear Dynamical System** domain, inspired by [3], simulates a controls problem and involves learning 8 parameters. The **Lunar Lander** domain [22] simulates a controls problem involving 4 parameters. The **Pizza Arrangement** 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 B.

4.1 INQUIRE Query Selection

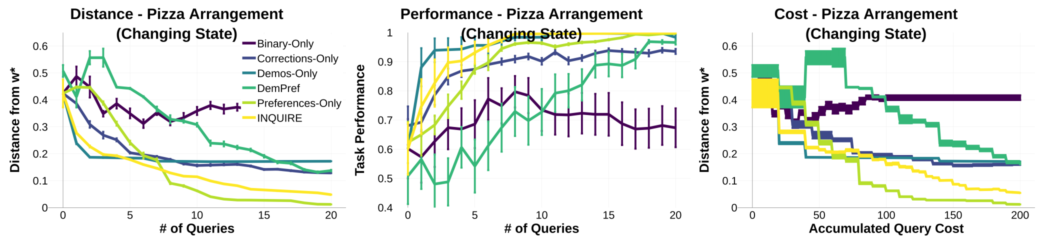
We first analyze how INQUIRE selects queries. Figure 1 reflects the changes in interaction types selected by INQUIRE over time. Figure 1a first reports these interaction selections in an unweighted query optimization setting, where all interaction types are assumed to be equally costly. In the parameter 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



(a) Linear Dynamical System



(b) Lunar Lander Task



(c) Pizza Arrangement Task

Figure 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.

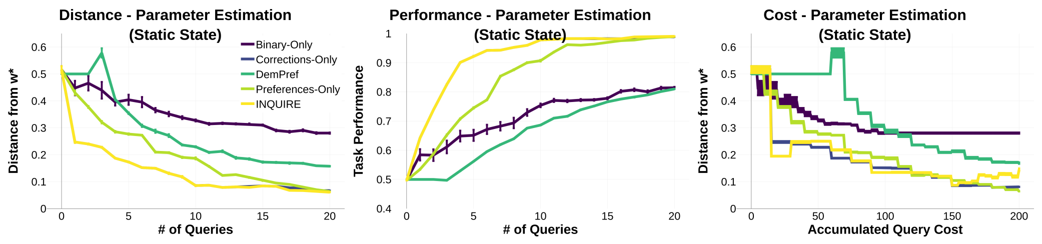
all other domains, INQUIRE requests a demonstration as its first query, then immediately switches to requesting preferences for the remaining queries (occasionally alternating between preferences and demonstrations in the Lunar Lander domain).

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 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).

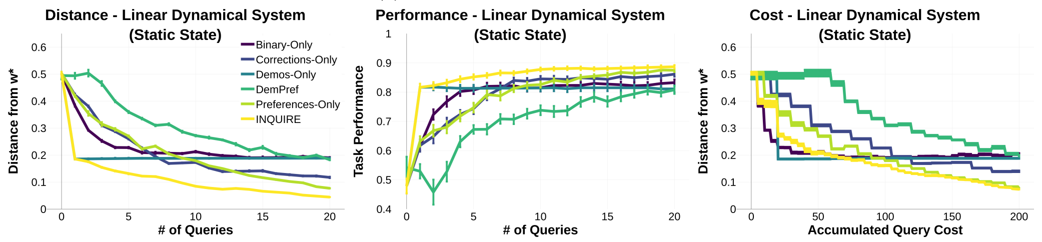
4.2 Learning Performance

We now analyze the effect of INQUIRE's interaction type selections on its learning performance and compare to two types of baselines. The first, DemPref [9], 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 [3] and thus represents this baseline method.

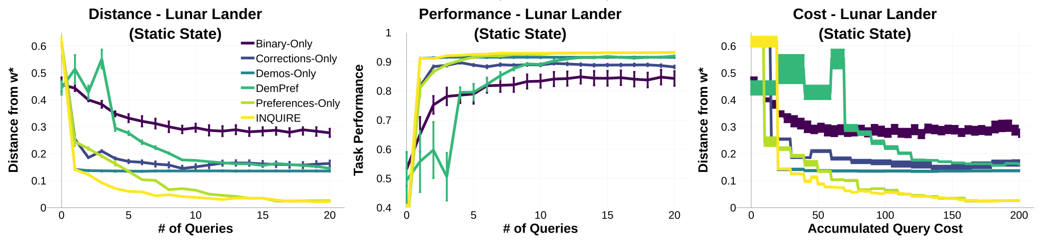
We first consider the changing-state formulation of each domain, where the robot is presented with a new state for each query. Since the Parameter Estimation domain does not contain states, we exclude it from this first set of results. Figure 2 illustrates this learning performance in the Linear Dynamical 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



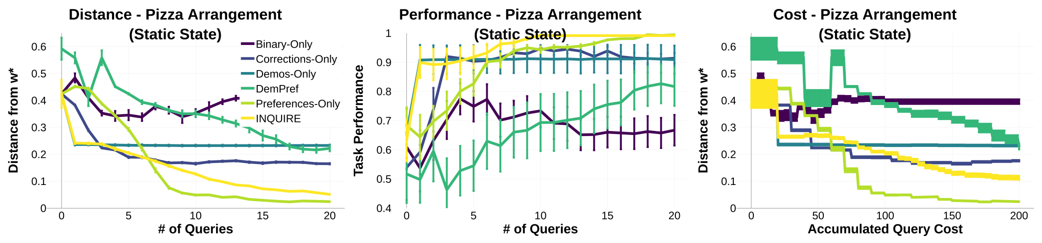
(a) Parameter Estimation Task



(b) Linear Dynamical System



(c) Lunar Lander Task



(d) Pizza Arrangement Task

Figure 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.

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** 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 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 C. 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 2-3) 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.

5 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, 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 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).

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 [8]). The cost metric used in our study thus serves as an example of how INQUIRE can factor in interaction costs.

6 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 [23], and we expect that optimizing over the query state in addition to query type and content would improve the performance of INQUIRE.

7 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[3] for evaluating and comparing active-learning agents across multiple environments and simulated teachers.

[3] `https://github.com/HARPLab/inquire`

Acknowledgments

The authors would like to thank the members of HARP Lab, IAM Lab, and the anonymous reviewers for their valuable feedback. This work is supported in part by the Office of Naval Research (N00014-18-1-2503).

References

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), apr 2017. ISSN 0360-0300. doi:10.1145/3054912. URL https://doi.org/10.1145/3054912.

[2] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *ICML*, volume 1, page 2, 2000.

[3] E. Biyik, M. Palan, N. C. Landolfi, D. P. Losey, and D. Sadigh. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning (CoRL)*, pages 1177–1190, 2020.

[4] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz. A survey of preference-based reinforcement learning methods. *The Journal of Machine Learning Research*, 18(1):4945–4990, 2017.

[5] T. Fitzgerald, E. Short, A. Goel, and A. Thomaz. Human-guided trajectory adaptation for tool transfer. In *Intl. Conf. on Autonomous Agents and MultiAgent Systems*, pages 1350–1358, 2019.

[6] A. Bajcsy, D. Losey, M. O'Malley, and A. Dragan. Learning robot objectives from physical human interaction. *Proceedings of Machine Learning Research*, 78:217–226, 2017.

[7] C. Celemin and J. Ruiz-del Solar. An interactive framework for learning continuous actions policies based on corrective feedback. *Journal of Intelligent & Robotic Sys.*, 95(1):77–97, 2019.

[8] Y. Cui, P. Koppol, H. Admoni, S. Niekum, R. Simmons, A. Steinfeld, and T. Fitzgerald. Understanding the relationship between interactions and outcomes in human-in-the-loop machine learning. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[9] M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh. Learning reward functions by integrating human demonstrations and preferences. *RSS*, 2019.

[10] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

[11] K. Bullard, A. L. Thomaz, and S. Chernova. Towards intelligent arbitration of diverse active learning queries. In *IROS*, pages 6049–6056, 2018.

[12] P. Koppol, H. Admoni, and R. G. Simmons. Interaction considerations in learning from humans. In *IJCAI*, pages 283–291, 2021.

[13] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters. Active reward learning. In *Robotics: Science and Systems*, 2014.

[14] D. Sadigh, A. Dragan, S. Sastry, and S. A. Seshia. Active preference-based learning of reward functions. In *RSS*, 2017.

[15] N. Wilde, D. Kulić, and S. L. Smith. Active preference learning using maximum regret. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10952–10959. IEEE, 2020.

[16] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. Reward learning from human preferences and demonstrations in atari. *NeurIPS*, 31:8011–8023, 2018.

[17] K. Li, M. Tucker, E. Bıyık, E. Novoseller, J. W. Burdick, Y. Sui, D. Sadigh, Y. Yue, and A. 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. IEEE, 2021.

[18] D. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[19] E. Bıyık, A. Talati, and D. Sadigh. Aprel: A library for active preference-based reward learning algorithms. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 613–617. IEEE, 2022.

[20] H. J. Jeon, S. Milli, and A. D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning. In *NeurIPS*, 2020.

[21] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

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

[23] M. S. Lee, H. Admoni, and R. Simmons. Machine teaching for human inverse reinforcement learning. *Frontiers in Robotics and AI*, 8:188, 2021.

[24] T. Giorgino. Computing and visualizing dynamic time warping alignments in r: the dtw package. *Journal of statistical Software*, 31:1–24, 2009.

A Approach Details

A.1 Information Gain Derivation

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

$$
\begin{aligned}
&\mathrm{IG}(X, y) && (14)\\
&= H(X) - H(X|y) && (15)\\
&= -\mathbb{E}_{x|X} \log P(x) + \mathbb{E}_{x|X,y} \log P(x|y) && (16)\\
&= \sum_{x \in X} P(x|y) \cdot \log P(x|y) - \sum_{x \in X} P(x) \cdot \log P(x) && (17)
\end{aligned}
$$

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:

$$\max_{q \in Q} \mathbb{E}_{c|C_i(q)} \left[\mathrm{IG}(\mathcal{W}, c)\right] \tag{18}$$

We now solve for the expected information gain according to Eqs. 14- 17 and following the derivation presented in [3]:

$$
\begin{aligned}
&\mathbb{E}_{c|C_i(q)} \left[\mathrm{IG}(\mathcal{W}, c)\right] && (19)\\
&= H(\mathcal{W}) - \mathbb{E}_{c|C_i(q)} \left[H(\mathcal{W}|c)\right] && (20)\\
&= -\mathbb{E}_{\mathcal{W}} \left[\log P(\mathcal{W})\right] + \mathbb{E}_{\mathcal{W}, c|C_i(q)} \left[\log P(\mathcal{W}|c)\right] && (21)\\
&= \mathbb{E}_{\mathcal{W}, c|C_i(q)} \left[\log P(\mathcal{W}|c) - \log P(\mathcal{W})\right] \qquad \text{(see proof in Sec. A.1.1)} && (22)\\
&= \mathbb{E}_{\mathcal{W}, c|C_i(q)} \left[\log \frac{P(\mathcal{W}|c)}{P(\mathcal{W})}\right] && (23)\\
&= \mathbb{E}_{\mathcal{W}, c|C_i(q)} \left[\log \frac{P(c|\mathcal{W})}{P(c)}\right] \qquad \text{(by Bayes' rule)} && (24)\\
&= \sum_{c \in C_i(q)} \left[P(c) \sum_{w \in \mathcal{W}} \left[P(w|c) \cdot \log \frac{P(c|w)}{P(c)}\right]\right] && (25)\\
&= \sum_{c \in C_i(q)} \left[P(c) \sum_{w \in \mathcal{W}} \left[\frac{P(w)P(c|w)}{P(c)} \cdot \log \frac{P(c|w)}{P(c)}\right]\right] && (26)\\
&= \sum_{c \in C_i(q)} \sum_{w \in \mathcal{W}} \left[P(w) \cdot P(c|w) \cdot \log \frac{P(c|w)}{P(c)}\right] && (27)\\
&= \sum_{c \in C_i(q)} \sum_{w \in \mathcal{W}} \left[P(w) \cdot P(c|w) \cdot \log \frac{P(c|w)}{\sum_{w' \in \mathcal{W}} P(w') \cdot P(c|w')}\right] && (28)\\
&\approx \frac{1}{M} \sum_{c \in C_i(q)} \sum_{w \in \Omega} \left[P(c|w) \cdot \log \frac{M \cdot P(c|w)}{\sum_{w' \in \Omega} P(c|w')}\right] && (29)
\end{aligned}
$$

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

A.1.1 Proof of Eq. 22

$$
\begin{align}
& -\mathbb{E}_{\mathcal{W}}\left[\log P(\mathcal{W})\right] + \mathbb{E}_{\mathcal{W},c|C_i(q)}\left[\log P(\mathcal{W}|c)\right] \tag{30}\\
& = \mathbb{E}_{\mathcal{W},c|C_i(q)}\left[\log P(\mathcal{W}|c)\right] - \mathbb{E}_{\mathcal{W}}\left[\log P(\mathcal{W})\right] \tag{31}\\
& = \left[\sum_{w\in\mathcal{W}} P(w) \sum_{c\in C_i(q)} P(c|w)\cdot\log P(w|c)\right] - \left[\sum_{w\in\mathcal{W}} P(w)\cdot\log P(w)\right] \tag{32}\\
& = \sum_{w\in\mathcal{W}} P(w)\cdot\left[\left(\sum_{c\in C_i(q)} P(c|w)\cdot\log P(w|c)\right) - \log P(w)\right] \tag{33}\\
& = \sum_{w\in\mathcal{W}} P(w)\cdot\left[\sum_{c\in C_i(q)} P(c|w)\cdot\log P(w|c) - \sum_{c\in C_i(q)} P(c|w)\cdot\log P(w)\right] \tag{34}\\
& = \sum_{w\in\mathcal{W}} P(w)\cdot\sum_{c\in C_i(q)} P(c|w)\cdot\left[\log P(w|c) - \log P(w)\right] \tag{35}\\
& = \mathbb{E}_{\mathcal{W},c|C_i(q)}\left[\log P(\mathcal{W}|c) - \log P(\mathcal{W})\right] \tag{36}
\end{align}
$$

A.2 KL Divergence Formulation

We now show that we can alternatively derive Eq. 29 from the standard KL divergence equation:

$$
\text{KL}(P||Q) = \sum_{x\in X}\left[P(x)\cdot\log\frac{P(x)}{Q(x)}\right] \tag{37}
$$

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

$$
\max_{q\in Q} \mathbb{E}_{c|C_i(q)}\left[\text{KL}(P(\mathcal{W}|c)||P(\mathcal{W}))\right] \tag{38}
$$

We now solve for the optimal query:

$$
\begin{align}
& \mathbb{E}_{c|C_i(q)}\left[\text{KL}(P(\mathcal{W}|c)||P(\mathcal{W}))\right] \tag{39}\\
& = \mathbb{E}_{c|C_i(q)}\left[\sum_{w\in\mathcal{W}}\left[P(w|c)\cdot\log\frac{P(w|c)}{P(w)}\right]\right] \tag{40}\\
& = \mathbb{E}_{c|C_i(q)}\left[\sum_{w\in\mathcal{W}}\left[P(w|c)\cdot\log\frac{P(c|w)}{P(c)}\right]\right] \tag{41}\\
& = \sum_{c\in C_i(q)}\left[P(c)\sum_{w\in\mathcal{W}}\left[P(w|c)\cdot\log\frac{P(c|w)}{P(c)}\right]\right] \tag{42}
\end{align}
$$

Which is equivalent to Eq. 25, and thus results in Eq. 29.

A.3 Probability Tensor Derivations

See Table 1 for all definitions of q, c, c^+, c^- for each interaction type. In the demonstration case, we define $\mathbf{P}$ as follows:

$$\begin{aligned}
\mathbf{P}^{(\text{demo})}_{q,c,\omega} &= \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} && (43)\\
&= \frac{e^{\beta \cdot \phi(c_0^+) \cdot \omega}}{\sum_{t \in T} e^{\beta \cdot \phi(t) \cdot \omega}} \qquad \text{(since } |c^+| = 1 \text{ and } c^+ \cup c^- = T \text{ for demonstrations)} && (44)\\
&= \frac{\mathbf{E^T}_{0,c_0^+,\omega}}{\sum_{t \in T} \mathbf{E^T}_{0,t,\omega}} && (45)\\
&= \left[\mathbf{E^T}_0 \oslash \sum_{t \in T} \mathbf{E}_t\right]_{c,\omega} \qquad \text{(since there is a 1-1 correlation between } c \text{ and } c^+ \text{ in demos)} && (46)
\end{aligned}$$

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:

$$\begin{aligned}
\mathbf{P}^{(\text{pref})}_{q,c,\omega} &= \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} && (47)\\
&= \frac{e^{\beta \cdot \phi(c_0^+) \cdot \omega}}{e^{\beta \cdot \phi(q_0) \cdot \omega} + e^{\beta \cdot \phi(q_1) \cdot \omega}} \qquad \text{(since } |c^+| = 1 \text{ and } c^- = q \setminus c^+ \text{ in preferences)} && (48)\\
&\text{Since } c_0 \implies c^+ = \{q_0\} \text{ and } c_1 \implies c^+ = \{q_1\}\text{:} && (49)\\
&= \left[\frac{e^{\beta \cdot \phi(q_0) \cdot \omega}}{e^{\beta \cdot \phi(q_0) \cdot \omega} + e^{\beta \cdot \phi(q_1) \cdot \omega}}, \frac{e^{\beta \cdot \phi(q_1) \cdot \omega}}{e^{\beta \cdot \phi(q_0) \cdot \omega} + e^{\beta \cdot \phi(q_1) \cdot \omega}}\right]_c \qquad \text{(where } c \in \{0,1\}\text{)} && (50)\\
&= \left[\frac{\mathbf{E^T}_{q_0,q_1,\omega}}{[\mathbf{E} + \mathbf{E^T}]_{q_0,q_1,\omega}}, \frac{\mathbf{E}_{q_0,q_1,\omega}}{[\mathbf{E} + \mathbf{E^T}]_{q_0,q_1,\omega}}\right]_c && (51)\\
&= \left[\left(\mathbf{E} \oslash (\mathbf{E} + \mathbf{E^T})\right)^{\mathbf{T}}, \mathbf{E} \oslash (\mathbf{E} + \mathbf{E^T})\right]_{c,q_0,q_1,\omega} && (52)
\end{aligned}$$

In the corrections case:

$$\begin{aligned}
\mathbf{P}^{(\text{corr})}_{q,c,\omega} &= \frac{\sum_{t \in c^+} e^{\beta \cdot \phi(t) \cdot \omega}}{\sum_{t \in c^+ \cup c^-} e^{\beta \cdot \phi(t) \cdot \omega}} && (53)\\
&= \frac{e^{\beta \cdot \phi(c_0^+) \cdot \omega}}{e^{\beta \cdot \phi(c_0^+) \cdot \omega} + e^{\beta \cdot \phi(c_0^-) \cdot \omega}} \qquad \text{(since } |c^+| = 1 \text{ and } |c^-| = 1\text{)} && (54)\\
&= \frac{\mathbf{E^T}_{q,c,\omega}}{[\mathbf{E} + \mathbf{E^T}]_{q,c,\omega}} \qquad \text{(due to 1-1 correlation between } q \text{ and } c^- \text{ and between } c \text{ and } c^+ \text{ in corrections)} && (55)\\
&= \left[\mathbf{E^T} \oslash (\mathbf{E} + \mathbf{E^T})\right]_{q,c,\omega} && (56)
\end{aligned}$$

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:

$$
\begin{aligned}
\mathbf{P}^{\text{(bnry)}}_{q,c,\omega} &= \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} && (57)\\
&\text{Since } c_0 \implies c^+ = T\setminus q, \quad c^- = q && (58)\\
&\text{and } c_1 \implies c^+ = q, \quad c^- = T\setminus q: && (59)\\
&= \frac{1}{\alpha}\left[\frac{1}{|T\setminus q|}\cdot\frac{\sum_{t\in T\setminus q} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in T} e^{\beta\cdot\phi(t)\cdot\omega}}, \frac{e^{\beta\cdot\phi(q)\cdot\omega}}{\sum_{t\in T} e^{\beta\cdot\phi(t)\cdot\omega}}\right]_c \qquad \text{(since } c\in\{0,1\} \text{ in binary rewards)} && (60)\\
&= \left[\frac{1-\mathbf{P}^{\text{(demo)}}_{0,q,\omega}}{\alpha\left(|T|-1\right)}, \frac{\mathbf{P}^{\text{(demo)}}_{0,q,\omega}}{\alpha}\right]_c \qquad \text{(where } \alpha \text{ is a normalization factor s.t. } \sum_c \mathbf{P}^{\text{(bnry)}}_{q,c,\omega} = 1) && (61)\\
&= \left[1-\frac{\mathbf{P}^{\text{(demo)}}_{0,q,\omega}}{\alpha}, \frac{\mathbf{P}^{\text{(demo)}}_{0,q,\omega}}{\alpha}\right]_c \qquad \text{(since } \sum_c \mathbf{P}^{\text{(bnry)}}_{q,c,\omega} = 1) && (62)\\
&= \left[1-\left(\mathbf{E^T}_0 \oslash \alpha\sum_{t\in T}\mathbf{E}_t\right), \mathbf{E^T}_0 \oslash \alpha\sum_{t\in T}\mathbf{E}_t\right]_{c,q,\omega} && (63)
\end{aligned}
$$

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

A.4 Gradient Derivation

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

$$
\begin{aligned}
\omega^* &= \arg\max_\omega \prod_{c\in\mathbf{F}} P(c|\omega) && (64)\\
&= \arg\max_\omega \prod_{c\in\mathbf{F}} \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} && (65)
\end{aligned}
$$

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

$$
\begin{aligned}
\ell(\omega) &= \log\prod_{c\in\mathbf{F}} \frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}} && (66)\\
&= \sum_{c\in\mathbf{F}}\left[\log\frac{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}}\right] && (67)\\
&= \sum_{c\in\mathbf{F}}\left[\log\left(\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}\right) - \log\left(\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}\right)\right] && (68)
\end{aligned}
$$

$$
\frac{\partial\ell(\omega)}{\partial\omega_j} = \sum_{c\in\mathbf{F}}\left[\frac{\sum_{t\in c^+}\beta\cdot\phi_j(t)\cdot e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+} e^{\beta\cdot\phi(t)\cdot\omega}} - \frac{\sum_{t\in c^+\cup c^-}\beta\cdot\phi_j(t)\cdot e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}}\right] \qquad (69)
$$

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

$$
\frac{\partial\ell(\omega)}{\partial\omega_j} = \sum_{c\in\mathbf{F}}\left[\beta\cdot\phi_j(c^+_0) - \frac{\sum_{t\in c^+\cup c^-}\beta\cdot\phi_j(t)\cdot e^{\beta\cdot\phi(t)\cdot\omega}}{\sum_{t\in c^+\cup c^-} e^{\beta\cdot\phi(t)\cdot\omega}}\right] \qquad (70)
$$

A.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}.

B Evaluation Details

B.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 ($r(\omega) = \omega \cdot \omega^* = \cos(\theta)$).

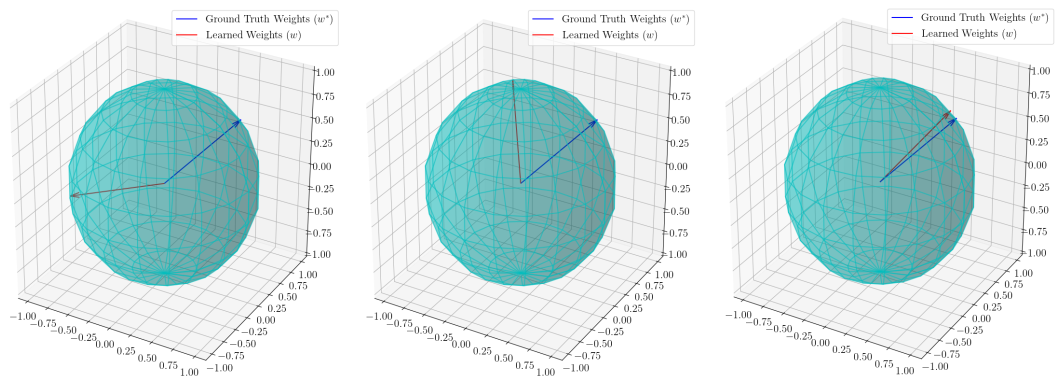


Figure 4: In the Parameter Estimation domain, the robot is tasked with estimating a high-dimensional ground truth weight vector w^* with its own set of learned weights w. To visualize this concept, a simpler case is illustrated above in three dimensions. All weight vectors (ground truth and learned weights) 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.

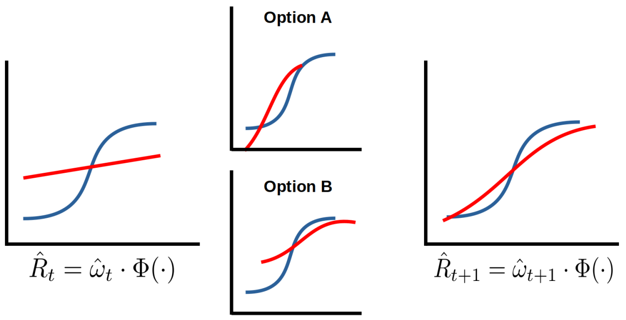


Figure 5: An example of a preference query in the Linear Dynamical System domain. At time $= t$, the learned reward function yields the red "trajectory." After posing a preference query (which consists of options A and B), the corresponding belief update yields the approximated reward function at time $= t + 1$.

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 [9], 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.

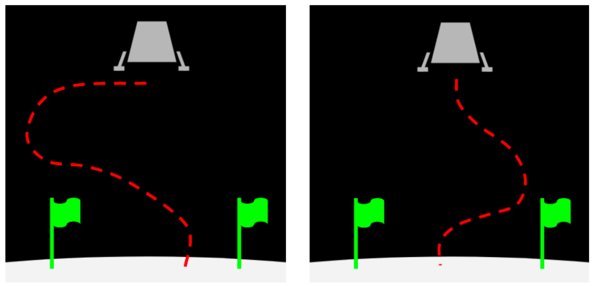


Figure 6: The Lunar Lander domain involves having the robot pilot a lunar lander to safely descend and arrive at a landing pad. The depicted preference query illustrates two different trajectories that may be taken by the lander to reach the destination.

Domain #4: Pizza Arrangement 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.

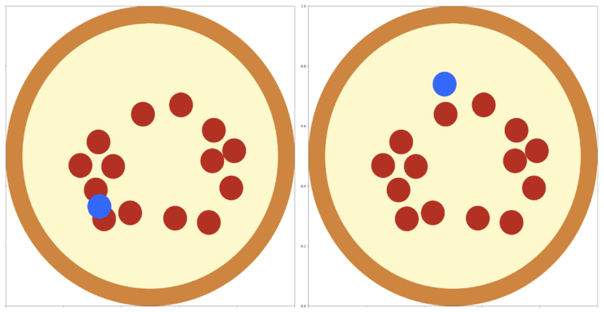


Figure 7: The task in the Pizza Arrangement domain is to learn how to place toppings according to a human's reward over topping positions. In the depicted preference query, a human's choice indicates their preference for the "next" topping's position (choices represented in blue).

B.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):

$$\text{Oracle}_{\text{demo}}(q) = \underset{t\in T'}{\arg\max}\,(\phi(t)\cdot\omega^*) \qquad \text{Oracle}_{\text{pref}}(q) = \underset{t\in C(q)}{\arg\max}\,(\phi(t)\cdot\omega^*) \tag{71}$$

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:

$$\forall t \in T', \phi(t)\cdot\omega^* \geq \phi(q)\cdot\omega^* \tag{72}$$

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

$$\text{Oracle}_{\text{corr}}(q) = \underset{t\in T'}{\arg\max}\,\frac{\Delta_r(q,t)}{\Delta_d(q,t)} \tag{73}$$

$$\Delta_r(q,t) = \min_{t'\in T'} \frac{\phi(t)\cdot\omega^* - \phi(q)\cdot\omega^*}{\phi(t')\cdot\omega^* - \phi(q)\cdot\omega^*} \qquad \Delta_d(q,t) = \min_{t'\in T'} \frac{e^{\delta(t,q)}}{e^{\delta(t',q)}} \tag{74}$$

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 Arrangement 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:

$$R = \{\omega^*\cdot\phi(t), \forall t\in T'\} \qquad \text{Oracle}_{\text{bnry}}(q) = \begin{cases} + & R(\omega^*\cdot\phi(q)) \geq \alpha \\ - & \text{otherwise}\end{cases} \tag{75}$$

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

B.3 Evaluation Procedure

Algorithm 3 Evaluation Procedure

Input: generate_query and update_weights methods according to algorithm being tested

```
1: Generate ground truth reward function $\omega^*$
2: Generate 10 test states
3: Compute optimal trajectory $t_{\text{max}}$ for each test case using $\omega^*$
4: Compute least-optimal trajectory $t_{\text{min}}$ for each test case using $\omega^*$
5: for each of 10 runs do
6:     Generate 20 query states (if testing in the static condition, repeat the same state 20 times)
7:     for each of 20 queries do
8:         $s \leftarrow$ next query state
9:         $q^* \leftarrow$ generate_query($s, \mathcal{I}, \mathbf{\Omega}$)
10:        $\mathbf{F} \leftarrow \mathbf{F}+$ query_oracle($q^*$)
11:        $\mathbf{\Omega} \leftarrow$ update_weights($\mathbf{F}$)
12:        $\tilde{\omega} \leftarrow$ mean($\Omega$)
13:        Record distance: $\frac{\text{arcccos}(\tilde{\omega}\cdot\omega^*)}{\pi}$
14:        for each of 10 test states do
15:            Compute optimal trajectory $t$ from the test state according to $\tilde{\omega}$
16:            Record performance: $\frac{\phi(t)\cdot\omega^* - \phi(t_{\text{min}})\cdot\omega^*}{\phi(t_{\text{max}})\cdot\omega^* - \phi(t_{\text{min}})\cdot\omega^*}$
```

C 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 8: AUC values for the distance plots in Figs 2-3. Darker cells indicate lower (better) values.

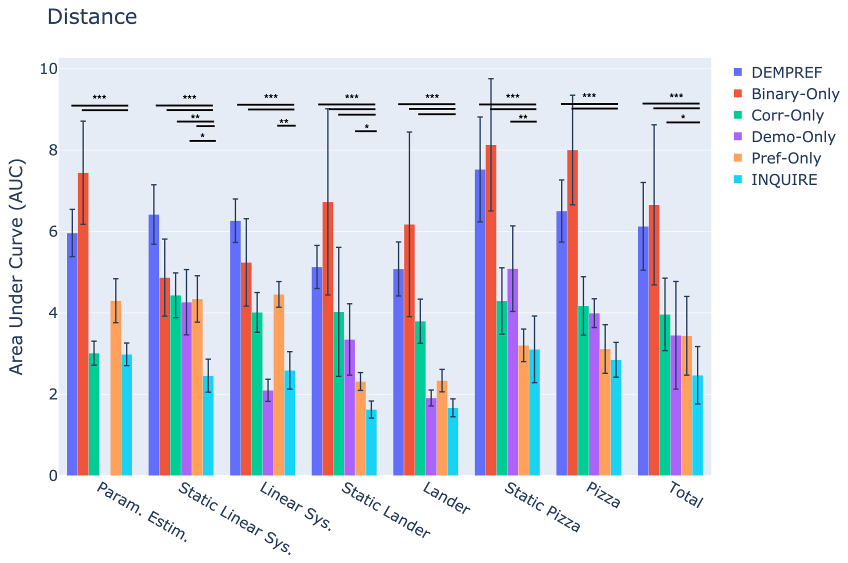


Figure 9: Visualizing Fig. 8, with statistical significance noted. (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$)

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 10: AUC values for the performance plots in Figs 2-3. Darker cells indicate higher (better) values.

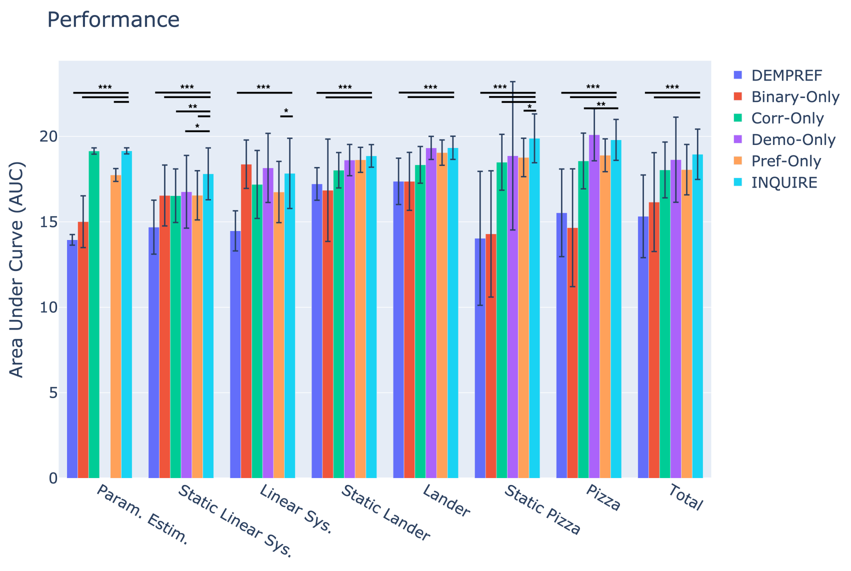
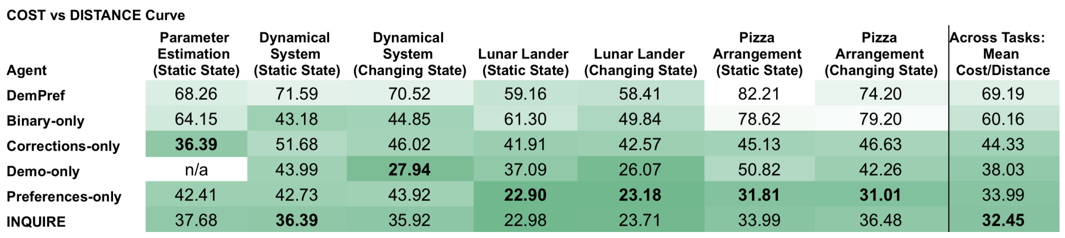


Figure 11: Visualizing Fig. 10, with statistical significance noted. (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$)



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 12: AUC values for the cost plots in Figs 2-3. Darker cells indicate lower (better) values.

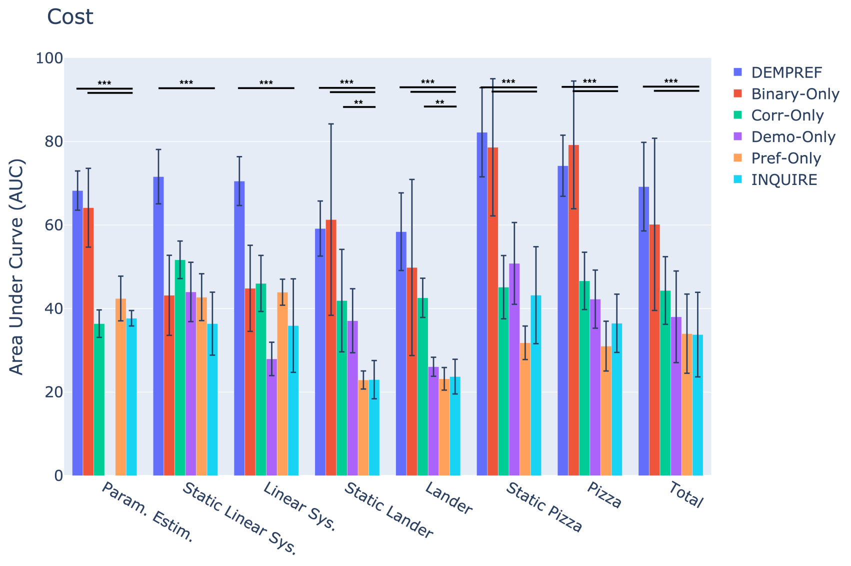


Figure 13: Visualizing Fig 12, with statistical significance noted. (*: $p < 0.05$, **: $p < 0.01$, ***: $p < 0.001$)