MANAGING LEARNING INTERACTIONS FOR COLLABORATIVE ROBOT LEARNING

A Dissertation Proposal
Presented to
The Academic Faculty

By

Kalesha Bullard

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing
College of Computing

Georgia Institute of Technology

December 2019

MANAGING LEARNING INTERACTIONS FOR COLLABORATIVE ROBOT LEARNING

Approved by:

Date Approved: May 17, 2019

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Robotic assistants should be able to actively engage their human partner(s) to generalize knowledge about relevant tasks within their shared environment. Yet a key challenge is not all human partners will be proficient at teaching; furthermore, humans should not be held accountable for tracking a robot's knowledge over time in a dynamically changing environment, across multiple tasks. Thus, it is important to enable these interactive robots to characterize their own uncertainty and equip them with an information gathering policy for asking the appropriate questions of their human partners to resolve that uncertainty. In this way, the robot shares the responsibility in guiding its own learning process and is a collaborator in the learning. Additionally, given the robot requires some tutelage from its partner, awareness of constraints on the teacher's time and cognitive resources available for devoting to the interaction could help the agent to use the time allotted more wisely.

This thesis examines the problem of enabling a robotic agent to leverage structured interaction with a human partner for acquiring concepts relevant to a task it must later perform. To equip the agent with the desired concept knowledge, we first explore the paradigm of Learning from Demonstration for the acquisition of (1) training instances as examples of task-relevant concepts and (2) informative features for appropriately representing and discriminating between task-relevant concepts. Given empirical evidence that a human partner can be helpful to the agent in solving the concept learning problem, we subsequently investigate the design of algorithms that enable the robot learner to autonomously manage interaction with its human partner, using a questioning policy to actively gather both instance and feature information. This thesis seeks to investigate the following hypothesis: **In the context of robot learning from human demonstrations in changeable and resource-constrained environments, enabling the robot to actively elicit multiple types of information through questions, and to reason about *what* question to ask and *when*, leads to improved learning performance.**

CHAPTER 1
INTRODUCTION

1.1 Motivation

With the growth of technology, robots are becoming more ubiquitous in society, and an important contributing factor for that is their widespread applicability in domains such as manufacturing, services, healthcare, education, defense, and space. Service robots in particular are intended to assist people in their daily lives [1] on a variety of tasks, over extended periods of time. However, we are not able to equip these robots with all of the task intelligence they will need by hard coding relevant knowledge a priori; the robots will need the ability to dynamically adapt in their environments. The goal then is to deploy robots who are able to continually acquire information that will expand their task intelligence, in dynamically changing environments.

Additionally, service robots are expected to perform tasks in spaces generally inhabited by humans and oftentimes in direct collaboration with people. As these robots are regularly solicited for aid, it becomes increasingly important for them to have the capacity to learn task knowledge in the context of the environment they are situated. For this, training data specific to the environment is necessary. And though real world environments may have an abundance of data, there is also an abundance of noise in the data as perceived through robot sensors and typically a scarcity of labeled data. Thus given a learning task, robotic agents needs a way of learning sufficient models within an abundance of noisy unlabeled data, that may dynamically change over time. For this, they can leverage interaction with their human partners.

The robotics paradigm of Learning from Demonstration (LfD) enables a robotic agent to learn a new task from examples provided by a human teacher [2, 3]. In LfD however, the model learned depends on the ability of the teacher to provide maximally informative input to the learner. Two important considerations persist however: (1) we want to leverage domain knowledge of any user, yet not all human partners will be proficient at teaching, and (2) over extended durations in a dynamically changing environment, it becomes difficult for even a proficient teacher to track the state of the learner's knowledge and accordingly decipher when to provide new information and what information to provide to help the learner refine its knowledge. The desire to minimize the cognitive burden of teaching for the human partner motivates the employment of agent-driven learning to ground task-relevant concepts. We seek to enable the robot learner to characterize its own uncertainty and autonomously solicit the information it needs from the teacher, to resolve that uncertainty. Thus, the robot shares the responsibility in guiding its own learning process and is a collaborator in the learning.

Within robotics, student-driven robot learning has primarily been encompassed by active learning. Active Learning (AL) is a subfield of machine learning which enables an agent to select unlabeled training examples about which it is most uncertain and query an oracle (or expert) for the correct labels [4, 3]. Prior literature in AL provides evidence for employing active learning with robots. Cakmak found that active learning achieved a higher execution success rate and greater generality than passive learning with novice teachers [5]. Generality was measured as skill coverage of possible goal configurations. Skills taught by an expert teacher achieved similar generality and a comparable success rate. Chernova shows in her work on active learning for a simulated driving domain that compared against a baseline of teacher guided (passive) learning with an expert teacher, a robot active learner is able to infer the teacher's driving policy with substantially fewer demonstrations and a smaller number of collisions [6]. Another advantage of using the robot active learner was that the learning session terminated when the robot learned the teacher's policy; using the teacher guided algorithm, it was difficult for teachers to select a termination point. Thus, prior findings suggest that learning is improved when the learner actively engages and guides the learning process, both with novice and expert teachers.

Accordingly, there is a growing body of AL literature within the field of robotics. In the task learning space, related literature has focused on learning an optimal policy towards the end of imitating a human demonstrator's behavior [6, 7], syntheis of queries for experimentation [8], symbol grounding [9, 10], and inferring task constraints [11]. This previous work has all focused on making *one* specific type of AL query towards generalization along that dimension of the task (*e.g.* taking the best action in a state or understanding action ordering constraints to more effectively plan). Cakmak and Thomaz introduce *three different* types of embodied queries and characterize the value of each in the context of learning lower-level skills [12], however this work does not directly compare the query types or perform action selection given the diverse pool of queries.

A brief review of prior literature in AL for robots suggests the problem is interesting and relevant to the robotics community. Nonetheless, it is important to situate why AL for LfD domains is also a difficult problem. Active learning can be thought of as the problem of *decision-making* for the goal of learning. From a decision-making persepective, robotics is a particularly challenging domain because robotic agents inherently exist in partially observable, stochastic, dynamic, and continuous environments. The environment dynamics may also be unknown to the agent, and it may exist amongst other goal-directed agents, suggesting the environment of a robot may also be unknown and multi-agent. Thus, attempting to optimally select queries to guide one's own learning in a partially observable, stochastic, dynamic, continuous, multiagent, and unknown environment represents an extremely difficult and complex decision-making problem for an artificial agent. The problem remains complex and challenging even if we consider only a subset of these environmental properties. For this thesis, we scope the broad and complex decision-making problem of active robot learning to consider the problem of concept learning in partially observable, stochastic, and "changeable" environments. By *changeable*, we mean loosely inspired by the dynamic nature of real world environments in which robots exist, whereby we simulate change over time in the environment, representative of the type of state change a robot is likely to encounter. Additionally, in LfD settings, where collaborative learning with a human partner is the goal, there are likely to be contraints imposed on the learning interaction, based upon limitations in time and cognitive resources the human has to devote to the learning interaction. Similar to some prior work in Interactive Robot Learning, this thesis explores the problem domain of concept learning for a robotic agent [10, 9, 13] and employs the broad spectrum of query types explored in AL literature [12] for acquiring concepts. However, it contributes novel exploration into the problems of interactive learning for *task-situated* concept learning within changeable and resource-constrained environments, and *arbitration between* diverse types of learning queries within a unifying principled framework.

1.2 Thesis Overview

When a robotic agent is given a recipe for a task, it must first perceptually ground each entity and concept within the recipe (*e.g.* items, locations) in order to later perform the task. Assuming no prior knowledge, this is particularly challenging when situated in new or dynamic environments, where the robot has limited representative training data. Given the agent operates in human settings however, it should be able to actively engage its human partner(s) in order to generalize knowledge about relevant tasks within their shared environment. Yet, not all human partners will be proficient at teaching; furthermore, humans should not be held accountable for tracking a robot's knowledge over time in a dynamically changing environment, across multiple tasks. Thus, it is important to enable these interactive robots to characterize their own uncertainty and equip them with an information gathering policy for asking the appropriate questions of their human partners to resolve that uncertainty. Additionally, given that a robot requires some tutelage from a human partner, awareness of constraints on the teacher's time or cognitive resources available for devoting to the interaction could help the agent to use the time allotted more wisely.

This thesis examines the problem of enabling a robotic agent to leverage structured interaction with a human partner for acquiring concepts relevant to a task it must later perform. To equip the agent with the desired concept knowledge, we first explore the paradigm of LfD for the acquisition of (1) training instances as examples of task-relevant concepts and (2) informative features for appropriately representing and discriminating between task-relevant concepts. Given empirical evidence that a human partner can be useful for solving the concept grounding problem, we subsequently investigate the design of algorithms that enable the robot learner to autonomously manage interaction with its human partner, using a questioning policy to actively gather both instance and feature information. This thesis seeks to investigate the following hypothesis: **In the context of robot learning from human demonstrations in changeable and resource-constrained environments, enabling the robot to actively elicit multiple types of information through questions, and to reason about *what* question to ask and *when*, leads to improved learning performance.**

1.2.1 Contributions of Thesis Work

Much of LfD literature for high-level tasks has focused on policy learning, learning a mapping between input states and the optimal output actions [3]. However within this work, we focus on less explored aspects of the task learning problem: (a) acquiring training instance data to solve the problem of task-situated *symbol grounding*, (b) maximizing sample efficiency for symbol grounding in sparse data environments by using humans to help solve the *feature selection* problem, (c) employing the use of *active learning* to ground relevant concepts, by extracting both informative features and representative instances from human teachers, and (d) expanding the active learning problem to *jointly optimize* for both learning objectives and environmental constraints. Toward that end, this thesis work has contributed the following:

- *Learning from Demonstration for Task-Situated Symbol Grounding:* Symbol Grounding is the problem of mapping symbolic representations to constructs in the physical world [14]. In particular, we consider the problem of grounding a high-level task, described by an abstract task recipe. The given recipe enumerates *objects* and *semantic locations* relevant for execution of the task, but not grounded in the physical environment where the robot is situated. For example, in order to serve pasta, the robot must get a “bowl” from the “cupboard”. But it does not know what a “bowl” for storing pasta looks like, where to find the “cupboard” in its environment, or where in the “cupboard” “bowls” are stored. This thesis is the first to explore the use of LfD for efficiently providing the robot with the perceptual groundings it needs for execution of the task recipe in any situated environment. Our findings suggest that these symbol groundings can be learned efficiently through demonstration and are necessary for each newly situated environment.

- *Symbol Grounding using Human-Driven Feature Selection:* Feature Selection (FS) is the problem of determining a subset of features for use in building all of the classifiers needed for the task [3]. We explore how to enable a robot learner to request feature information from a human teacher for use in constructing grounded concept models. Since the chosen model representation directly affects the learner’s ability to discriminate between task-relevant classes and thus later perform the task, the goal is for the learner to actively solicit information about which features it should use to represent and reason about the concepts. This thesis contributes the exploration of five different strategies for acquiring feature information from a human teacher and extracts insights about the most effective approaches.

- *Active Symbol Grounding through Arbitration of Diverse Learning Queries:* Traditional active learning enables a learner to request labels for unlabeled instances from an oracle, whether using existing instances or creating new instances (label requests). Prior active learning literature has also examined the request for new instances to be generated given a particular class (demonstration requests) and examined the use of feature oracles to label the relevance of individual features (feature relevance requests). However, the thesis contributes the first active learning policy which arbitrates between all of these types of queries within one unifying framework and autonomously acquires both feature and instance input from the human teacher. Another contribution of this work is that relaxes the typical AL assumption of a static unlabeled pool of data and considers tasks situated in dynamic environments.

- *Active Symbol Grounding within Time and Resource Constrained Environments:* The previous contribution was focused on the use of AL to acquire diverse types of input from a human teacher and thus more efficiently solve the symbol grounding problem. It assumed a dynamic environment where the learner could not exceed a given query budget. However, the learner did not explicitly reason about the resource constraints being imposed on it and thus was not able to adapt its strategy under different environmentally constrained conditions. This work is the first to contribute an AL policy that explicitly attempts to tradeoff the agent's learning objectives with the time and resource constraints being imposed on the agent. Our approach uses inverse reinforcement learning to infer the weights employed by a human expert questioner, with respect to a given set of decision features. A primary advantage of the approach explored is that it *expands* the capabilities of the active learner, subsuming the previous set of decision features and adding an awareness of resource consumption.

1.2.2 Outline of Dissertation Document

The dissertation is organized as follows. Technical background and related work for the research questions examined in this thesis are discussed in Chapters 2 and 3, respectively. Initial work on task-situated symbol grounding, focused on acquiring training instances from a human teacher, is detailed in Chapter 4. Work on exploring symbol grounding through human-driven feature selection is detailed in Chapter 5. The initial exploration on enabling an active learning policy for symbol grounding, through the use of diverse learning queries, is described in Chapter 6. Work building upon this to enable active learning in time and resourced constrained environments is detailed in Chapter 7. And finally, we conclude and discuss some interesting open questions in Chapter 8.

CHAPTER 2
BACKGROUND

In this thesis, the artificial agent (hereafter, called the agent) is tasked with the goal of perceptually grounding all concepts relevant to the task it must later perform. This work is motivated by the scenario of a social robot colocated in an environment with at least one human partner. As input, the agent is given a list of abstract concepts that are meaningful for the task. Symbol (or concept) grounding is the problem of mapping symbolic representations to constructs in the physical world [14]. Thus to solve the symbol grounding problem, the agent must build classifiers for all given task-relevant concepts, whereby physical instances in the agent's environment serve as training data, and the concepts serve as labels. We frame this problem as one of multi-class classification, with each instance assigned a single label, and the only labels considered are those of the task-relevant concepts. As a note, this document uses the terms symbol and concept interchangeably. This thesis explores two general approaches for enabling an agent to interactively solve this problem of grounded concept acquisition, through passive observation of human-provided input and through actively querying the human partner.

2.1 Problem Statement

Given a task, an accompanying set of task-relevant concepts Y, a finite duration of time or number of turns T to learn groundings for all $y \in Y$, and a task dataset D from which to sample training instances, the agent must learn a classifier or function approximator for each $y \in Y$. At each time step $t \leq T$, the agent perceives a scene of unlabeled instances X. The superset of features F used to represent each instance are derived from the robotic agent's sensors. Our overarching approach for solving this problem is to leverage interaction for acquiring task groundings, using the paradigm of Learning from Demonstration.

2.2 Learning from Demonstration

The field of Learning from Demonstration (LfD) examines how to enable an artificial agent to autonomously learn skills or tasks from demonstration data [2, 3]. It is a form of supervised learning, whereby the data is derived from a goal-directed external source. In the delineation of LfD by [3], the field can be partitioned into at least two broad categories of learning problems: (1) low-level skill learning and (2) high-level task learning. The problem of skill learning focuses on learning motion trajectories or primitive actions, presumably to be later used in the context of a task. The problem of task learning focuses on learning complex behaviors which utilize and combine primitive actions, assuming a library of primitive action controllers given a priori. It also subsumes other aspect of the task learning problem, like feature selection and object affordances, both presumably in service of a task.

In our setting, the agent is not seeking to learn a task policy, or mapping from states to actions, but instead concept classifiers that are necessary for performing the task. We also assume a library of primitive actions is given. The agent's goal is to acquire the necessary task-relevant concept knowledge from its human partner. Thus this problem of interactive concept grounding can still be broadly classified under the broader umbrella of task LfD.

In the initial iteration of solving the task-situated symbol grounding problem, the task is provided in the form of an abstract task plan, whereby the concepts to be learned correspond to parameters of each action in the plan. Later, this document also uses the term "task recipe" to denote one instance of a task plan. In this version of the problem, the teacher is responsible for providing demonstrations of the task, from which the agent extracts a set of labeled training instances L. A subset of relevant features $F^{'} \subseteq F$ for representing the concepts is assumed to be given a priori. In the next iteration of the problem, the assumption of relevant features *given* is relaxed. Yet, highly sample efficient learning is still critical, given the sparsity of data available to the agent and no reasonable assumption of prior concept knowledge. Since the agent is allowed to leverage interaction for learning however, this leads to exploration of both computational and human-driven feature selection for additionally inferring an optimal feature subset F^* as part of the learning problem.

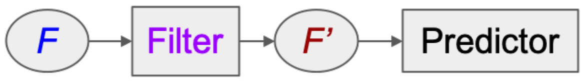
2.3 Feature Selection

Computational feature selection addresses the problem of automatically inferring a subset of informative features to be used as the underlying representation for all instances. It is an important aspect of any learning problem because the number of training instances required to sufficiently learn a model increases exponentially with the number features used to represent each instance. The choice of features also impacts how robust a model is to noise in the data. This suggests a key component of sample efficient learning is the selection of features.

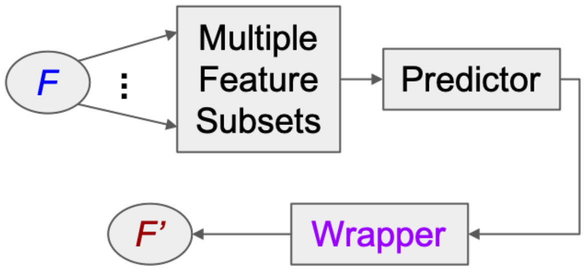
There are three classes of computational approaches for automatic feature selection that have been explored in the literature: *filters*, *wrappers*, and *embedded methods* [15]. Filters take the training data as input and examine the *relevance* of each individual feature with respect to the class label, as a preprocessing step. They typically output a ranking of all features according to degree of informativeness. The class of filtering algorithms is visually depicted by Figure 2.1a, where *predictor* represents a supervised learning algorithm, assumed to be decided and given.

One very common **filtering** algorithm ranks features based upon mutual information with the class label and selects all features whose informativeness exceeds some threshold, $I(Y|f) > \tau$. In this thesis, $\tau = 0$ so that all features providing *any* information are deemed relevant and are included in the selected subset. Equation 5.1 shows the information gain algorithm for computing *relevance* of feature $f \in F$; the equation computes a reduction in uncertainty for predicting the class Y given evidence of feature f. Thus $\forall f \in F$,

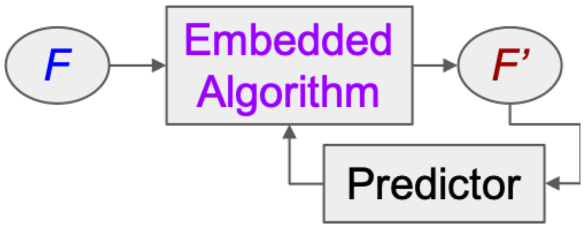
$$I(Y|f) = \sum_{y \in Y} H(y) - H(y|f) \tag{2.1}$$



(a) Filter



(b) Wrapper



(c) Embedded Method

Figure 2.1: High-level system diagrams for the three general classes of Feature Selection Algorithms: (a) filters, which take place as a preprocessing step for the predictor, (b) wrappers, which evaluate all possible feature subsets using the predictor and return the optimal subset based upon defined metrics, and (c) embedded algorithms, which incrementally evaluate a given feature subset using the predictor and iteratively refine the feature subset.

where $H(y)$ represents the entropy for class variable y.

Wrappers and embedded methods both conduct a search through the space of feature subsets, evaluating the *usefulness* of each subset, with respect to a given predictor. Wrappers typically conduct an exhaustive search and assess usefulness of all feature subsets based upon predefined performance metrics (*e.g.* learning accuracy, model complexity). At the end of the process, wrappers output an optimal feature subset, as illustrated by Figure 2.1b. Embedded algorithms, by contrast, proceed more efficiently by conducting a best first search through the feature subset space and incrementally refine the current feature subset. They typically either begin with no features and incrementally add (*forward selection*) or begin with all features and incrementally prune (*backward elimination*). Embedded algorithms terminate the incremental refinement process, given some predefined stopping criteria and output a locally optimal feature subset, as depicted in 2.1c.

In the first two iterations of solving this grounded concept acquisition problem, the goal was to understand if the agent could solely leverage interaction with a human partner to efficiently ground all concepts in its environment. The second half of the thesis focuses on enabling the agent to guide its own learning process for grounding given task-relevant concepts. the overarching approach taken is through active learning.

2.4 Active Learning

Active Learning (AL) is a subfield of Machine Learning whereby the agent queries an oracle (or teacher) for labels on selected unlabeled instances, according to predefined utility metrics [16, 17]. It differs from traditional supervised or ”passive” learning in that instead of the teacher (or environment) selecting training data and the learner passively observing, the learner queries instances that reduce its confusion.

AL algorithms typically assume the appropriate hypothesis class and representation for the problem have been selected a priori. There are three general scenarios these algorithms have been designed for: query synthesis, stream-based sampling, and pool-based sampling [16]. Query synthesis algorithms synthesize new instances de novo, given a definition of the input space, but not necessarily the data-generating distribution [8]. In the stream-based selective sampling scenario, unlabeled instances are sampled from the underlying distribution one at a time. For each sample presented, the learner determines whether to query or discard the instance. Though there are problem domains where the former two have been used, pool-based sampling has been used for the majority of AL applications and is the scenario most appropriate for the work discussed in this thesis. In pool-based sampling, the learner has access to a large pool of unlabeled data and a substantially smaller set of labeled data. This document denotes these as X and L respectively. Typically in AL literature, the unlabeled pool X is assumed to be static, and queries are selected in a greedy fashion, according to a utility metric. Once an optimal instance to query $x^* \in X$ is selected and a label is provided, $L \leftarrow L \cup \{x^*\}$ and $X \leftarrow X - \{x^*\}$. For the problem domain considered in this work however and for robotics more broadly, it is more appropriate to consider the unlabeled set (presumably deriving from instances in the robot's environment) as dynamically changing, since real world environments are dynamic. Additionally, instances correspond to objects in our domain, and removing instance queries from the unlabeled pool would require the robotic agent to track all objects it has requested information about, even as the environment changes over time, in order to later identify an object as one that has been queried. As we do not wish to make the restricting assumption that objects must be tracked over time, it is not feasible to have the agent remove each instance query made from the unlabeled pool. Thus our problem domain requires an AL algorithm able to handle a dynamically changing unlabeled set X and able to make decisions that enable learning progress, in spite of being memoryless.

Within the body of literature on pool-based AL, there are several classes of algorithms that have been investigated: uncertainty sampling approaches, committee-based approaches, decision-theoretic approaches, and cluster-based approaches. The first two families have also been combined with density weighting to tradeoff uncertainty and diversity in sample selection [17]. Cluster-based approaches are most advantageous and appropriate where there is a *very* large pool of unlabeled data and one seeks to greatly reduce the number of candidate queries by first leveraging structure in the data. As data scarcity is a key challenge in LfD domains, this class of algorithms is not as well suited for our problem domain. Thus, we will focus our discussion on the first three families.

Uncertainty sampling approaches assume a single hypothesis and utilize the posterior probability distribution over labels in Y given unlabeled instance x, $p_\theta(Y|x)$, in order to detect outliers or instances closest to a decision boundary. A commonly used metric for uncertainty sampling is *prediction entropy*, shown in Equation 2.2.

$$H_\theta(Y|x) = -\sum_{y \in Y} p_\theta(y|x) \log p_\theta(y|x) \tag{2.2}$$

Another metric employed in prior literature is based upon *prediction margin*.

$$M_\theta(Y|x) = p_\theta(\hat{y}_1|x) - p_\theta(\hat{y}_2|x) \tag{2.3}$$

where $\hat{y}_1$ and $\hat{y}_2$ represent the classifier's first and second most likely prediction. Though less commonly used, prediction margin may be more desirable when the goal is to better discriminate among classes. This thesis employs both uncertainty metrics.

Committee based approaches to AL consider a committee or ensemble of hypotheses attempting to explain the same data. Hypotheses, which are computational models, may be different instantiations derived from the same model class, *e.g.* using different parameter values or feature representations. They may also come from different model classes altogether. The primary requirement involves the ability to generate an ensemble of hypotheses and a utility metric for measuring committee disagreement or uncertainty. Committee-based approaches often involve computing a consensus probability for each label $y \in Y$ by averaging over predictions from the entire committee. Some utility metrics used have been: entropy given the distribution of consensus probabilities and average KL divergence when comparing distance of each committee member from the consensus. One limitation of both uncertainty-sampling and committee-based approaches however is they may be prone to querying outliers, as these instances may serve as a continual source of uncertainty for the classifiers, due to their inherent distance in the instance space from more prototypical examples. One way this has been addressed in the literature is by using density weighting techniques to bias query selection toward instances that more representative of the unlabeled pool. Another promising way of addressing the limitations of the prior two classes of approaches discussed is to use AL algorithms that directly optimize for expected reduction in classification error.

Decision-theoretic (DT) approaches to AL simulate all possible outcomes of each candidate query action and optimize for reduction of future expected error, either implicitly or explicitly. Equation 2.4 shows a decision-theoretic measure for explicitly computing expected error reduction given a query about instance x.

$$ER_\theta(x) = \sum_{y \in Y} p_\theta(y|x) \left[\sum_{x' \in X} 1 - p_{\theta^+}(\hat{y}|x') \right] \tag{2.4}$$

Here x represents the instance queried, y the hypothetical label provided, $x' \in X$ an instance in the unlabeled pool after retraining on the simulated new training sample $X \cup \langle x, y \rangle$, θ^+ the retrained model, and $\hat{y}$ the new model's most likely prediction for x.

DT approaches must retrain all classifiers for every possible $\langle query, response \rangle$ combination, in order to compute an expectation over all possible outcomes and select an optimal query. Since they directly optimize for achieving a desired outcome, they have generally been found to perform better than all other families of approaches for AL, with the tradeoff being that they are significantly more computationally expensive. Another advantage of decision theory is that it provides a general framework for making decisions, given different types of actions. Thus, because of its efficacy in solving learning problems and its ability to handle a diverse action space, DT is used as the basis for AL in this thesis. As a note, understanding how to make this approach more scalable and sufficiently fast to run in real time is an important challenge and left as a topic for future work.

As a first exploration into active learning for grounding task-relevant concepts, we contributed a decision-theoretic framework able to reason about the acquisition of both instance and feature information from a human teacher. However, as the problem changes to investigate AL within constrained environments, more representative of human settings, the AL framework in this thesis is expanded to more appropriately capture both the learning problem being solved and the *context* of the learning environment. This is achieved by expanding the set of decision criteria considered in the agent's objective function. However, given the decision feature space expands and performing policy rollouts (active learning episodes) can be computationally intensive and time consuming, it is important to find an efficient and scalable way to determine an optimal assignment of weights for this more diverse set of decision features. For learning an optimal set of weights, we use imitation learning.

2.5 Inverse Reinforcement Learning

Imitation Learning and LfD are often referred to synonymously in the literature and while there is very large overlap, there are also differences in terminology and some of the problems covered in each space. The primary goal in Imitation Learning is that of mimicking an expert's behavior, towards executing a given skill or task in a desired way. In Imitation Learning literature, it is generally assumed that (a) demonstrations are provided by a single expert, (b) features appropriate for characterizing the desired behavior will be selected or extracted, and (c) representations for the policy and learning algorithm will be selected a priori based upon the problem domain [18].

Imitation Learning can be broadly partitioned into two families of algorithms: behavioral cloning (BC) and inverse reinforcement learning (IRL). BC algorithms employ supervised learning to estimate a policy directly mapping states to actions. IRL algorithms, by contrast, use optimization techniques to recover the reward (objective) function from demonstrations of a policy. From the objective function, an optimal policy can be derived, using planning or reinforcement learning. Though IRL approaches only indirectly produce a policy, they are often the most succinct description of behavior for planning-oriented or deliberative tasks [18], and are thus the family of approaches used in this thesis, for imitating an expert's questioning strategy.

2.5.1 Markov Decision Process

The underlying framework typically used for IRL is Markov decision processes, a decision-theoretic framework used for sequential decision-making problems. Formally, a Markov decision process (MDP) is defined as a tuple $\{S, A, P, R, \gamma\}$ where S is the set of states, A the set of actions allowable in each state, P the transition or dynamics model which encompasses a set of state transition probabilities $P(s'|a,s)$, R the reward function, (also called the utility or objective function), and γ a discount factor [19, 20]. MDPs assume the Markov property, which says the state at time $t+1$ only depends on the state and action taken at time t. Thus $T = P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, ..., s_0, a_0) = P(s_{t+1}|s_t, a_t)$. We additionally note that though the reward function for MDPs is typically denoted by R in Reinforcement Learning literature, in order to keep consistency with literature in Decision Theory, introduced in an earlier section, we will reference the reward or utility function of the learner as U, from here forward.

The solution to an MDP is a policy $\pi(s)$. An optimal policy is obtained by maximizing expected long-term reward. Deterministic optimal policies map each state to a single optimal action a^*, whereas stochastic optimal policies map each state to a distribution over all $a \in A$. At each time step t of an episode, the agent is in state s_t, takes an action a_t, lands in state s_{t+1} as a result, and receives a reward (utility) u_t. The objective function U is defined by the mapping, $U : S \times A \rightarrow \mathbb{R}$. Given a finite time horizon T, a full learning episode τ (also called a trajectory or rollout) consists of a sequence of $\langle state, action, reward \rangle$ tuples such that $\tau = \{s_0, a_0, u_0, s_1, a_1, u_1, ..., s_T, a_T, u_T\}$.

Since MDPs solve a sequential decision making problem, policies seek to maximize expected *long term* utility using Equation 2.5:

$$U_t = \sum_{k=0}^{T} \gamma^k u_t + k + 1 \tag{2.5}$$

where T can be finite or $T = \infty$, and the discount factor $\gamma \in [0,1]$ characterizes the value of future rewards. As $\gamma \rightarrow 0$, the agent is considered to be myopic and is focused primarily on immediate rewards, whereas $\gamma \rightarrow 1$ indicates the agent weights future rewards more strongly and can become farsighted as a result [19, 20].

2.5.2 Problem Statement

The problem of IRL generally seeks to learn the agent's objective function U. It is often assumed that U is a function of decision features $\phi : S \rightarrow [0,1]^k$ where k is number of decision criteria or individual objectives for which the agent is optimizing [18]. Thus, the problem of IRL takes as input a set of expert trajectories T, sequences of decision feature values ϕ_τ corresponding to each trajectory $\tau \in$ T. An IRL algorithm may also optionally take a dynamics model P. It produces an estimate for $U(\phi)$ as output.

In much of prior literature, U has been assumed to be a linear combination of the decision features, as shown in Equation 2.6. Thus, the objective function estimate would take the form of a learned weight vector w^*.

$$U(s,a) = w_1 * \phi_1(\tau) + w_2 * \phi_2(\tau) + ... + w_k * \phi_k(\tau) \tag{2.6}$$

Generally, the IRL problem is considered to be ill-posed in that many reward functions can produce the same policy. To address this, many IRL algorithms approach the problem as one of constrained optimization whereby additional objectives are added as constraints, with the goal of yielding a unique solution [18]. This thesis uses the *feature matching* IRL problem formulation, whereby the key constraint being optimized for is that of matching the learner's expected decision feature counts to the expert's empirical decision feature counts. Given a trajectory τ, the decision feature counts $\mathbf{f}_\tau$ are defined as $\sum_{s_t \in \tau} \mathbf{f}_{s_t}$, the sum of state feature values observed along the path or trajectory [21].

Table 2.1 summarizes all notation introduced in this chapter. Though each chapter that follows is self-contained and will define relevant notation, to the extent possible, the notation below remains consistent throughout the thesis document.

Table 2.1: Table of Notation

D	dataset or data sample
Y	set of concepts or labels
L	set of training (labeled) instances
F	superset of features, derived from robot sensors
X	set of scene (unlabeled) instances
O	set of candidate objects (specific type of scene instance)
E	set of learning environments, whereby agent can be situated
S	set of agent internal states or world states
A	set of communicative actions (includes *no-query* action)
U	utility or objective function of active learner
k	number of decision criteria considered by learner
T	time horizon or number of turns in learning episode
Σ	set of querying strategies (active learners)
T_E	set of expert questioning episode demonstrations
T_L	set of learner questioning episode samples

CHAPTER 3
RELATED WORK

3.1 Question-Asking for Robot Task Learning

This thesis work utilizes the robotics paradigm of learning from demonstration (LfD), within the space of learning high-level tasks. In high-level task learning, it is assumed that the robot is given access to a library of primitive action controllers. Its goal then is to acquire knowledge useful for applying or combining these actions to perform a more complex behavior or task [3]. As motivated in the introduction, we are interested in a *self-driven* robot learner, one who leverages the expertise of the human teacher in order to acquire the information it needs, by asking questions. Within the robotics literature, this is primarily addressed by Active Learning (AL).

3.1.1 Active Robot Learning

Given a pool of unlabeled instances, the problem of Active Learning seeks to query an oracle to acquire the label of an instance about which it possesses uncertainty [16, 17]. In the task learning context, AL can enable a robot to both resolve unintended ambiguities during the learning process and explore unseen parts of the state space, in order to create a more *generalized* task representation. Related work on active learning for robots has focused on learning lower-level skill controllers [22, 23, 24, 23, 25, 26, 27], an optimal policy towards the end of imitating a human demonstrator's behavior [6, 7, 22, 12, 23, 28], grounding of task symbols or descriptions [9, 10, 29, 30], and inferring sequencing constraints on actions in a task [11, 31]. There has also been related work on strategies for introspective and extrospective detection and communication of the learner's knowledge gaps [32]. All of this previous work however has focused on asking *one* specific type of query towards generalization along that dimension of the task.

Cakmak and Thomaz introduced three different types of *embodied queries* and characterize the value of each in the context of learning lower-level skills [12]. The work describes three types of queries: *label*, *demonstration*, and *feature* queries. Label, or membership, queries request the correct label from the oracle for an unlabeled training instance (*e.g.* perform a new trajectory sampled from the skill model and ask if it is a positive example of the skill). Demo queries create a new scenario and request a demonstration of a positive example from the oracle (*e.g.* generate a new hand pose and requesting a demonstration of the given skill from that state). Feature queries can be decomposed into subtypes. Feature Relevance queries ask whether a particular feature is important or relevant for a skill (*e.g.* whether orientation of hand matters for executing a given skill). Feature Value and Feature Invariance queries inquire about specific allowable values for features (*e.g.* ask if a *particular* hand orientation has to be fixed when executing the skill). But the work by Cakmak does not include an algorithm for arbitrating between different types of learning queries, nor does it consider constraints imposed on the learner.

This thesis contributes a unifying active learning framework for querying diverse types of information from a human teacher in more realistic environments, where the learner is expected to: (1) track the state of its own knowledge, (2) determine *when* it needs help, (3) request *what* task relevant knowledge it requires, and (4) consider *constraints* being imposed on it (namely, from its teacher or learning environment). The types of information considered for querying are instances (demonstrations), labels, and feature subsets.

3.1.2 Computational Curiosity

Additionally, there is a body of work that focuses on enabling an artificial agent with intrinsic motivation to explore and examining how best to explore the learner's state space for interesting examples. It is largely encompassed by the subfield of computational curiosity [33, 34, 35, 36]. From a psychological perspective, curiosity implies an intrinsic drive for exploratory behaviors, such as learning, exploration, and investigation [37]. Computational models of curiosity can be partitioned into categories according to variables used for evaluation of stimuli. A taxonomy of existing computational models synthesized by [33] identifies six categories for evaluation of stimuli: novelty, surprise, uncertainty, conflict, change, and complexity. Though this body of literature seeks to solve a different problem, it is complimentary to AL, and we leverage the literature on computational curiosity for ideas regarding metrics an AI agent may use for exploration and selection of unlabeled instances to query.

3.2 Interactive Symbol Grounding

Recently there has been an increased focus on having robots take advantage of cloud computing resources and repositories of robot tasks. Tenorth and Beetz present KnowRob, a comprehensive knowledge processing infrastructure for embedding robots with all knowledge needed to interpret vague task descriptions and perform a task [38]. KnowRob is connected to RoboEarth, which has an entire cloud-based database of objects, environments, and action recipes, usable by robotic platforms [39]. Other works explore the use of recipes in task performance as well. Misra *et.al.* introduce Tell Me Dave, a framework that focuses on understanding potentially ambiguous or vague natural language instructions and converting them to a set of execution instructions to be carried out by the robot [40]. Bollini *et.al.* present a robotic chef which collects recipes online, parses them into a sequence of execution instructions, and executes the task [41]. In this prior work however, the robot can only execute the task in environments for which it has existing primitive action controllers. Within the context of these high level task descriptions which reference abstract symbols (*e.g.* cup, plate), if the robot has a way to perceptually ground the symbols, this provides it with necessary knowledge to enable task execution in *any* situated environment. This mapping from abstract symbols to constructs in the physical world is called symbol grounding.

The symbol grounding problem has been extensively studied in the literature. Within robotics, there has been only a sample of work that has addressed the symbol grounding problem. Some works have taken as input relatively unstructured natural language input and used context from the environment in order to ground it. [42] explores the acquisition of language from multi-modal channels of sensory data (namely sequential images and natural speech from video); in contrast, our work inversely maps task-relevant language to sensory data in the environment. In [40], free-form natural language phrases are mapped into sequences of mobile manipulation instructions, to be executed by a robot. This work is complimentary to our work in that it outputs a sequence of manipulation instructions for a task, *i.e.* a task recipe which could be used as input for perceptual grounding in the environment.

Other work has explored the use of LfD to learn mappings from concepts (language) to percepts in the environment, but for the purpose of inferring the goal of a high-level task [43, 44]. Recently, [10] examined the use of active learning for symbol learning and grounding, whereby the robot physically generates informative geometric configurations of objects and acquires a symbol label (*e.g. inside(X,Y)*); this is later coupled with a learned transition model to generate task plans. Overall, these works focus on learning (1) task goals or (2) relational symbols for task planning.

For a comprehensive discussion of the extensive body of literature on LfD, the reader is directed to [2, 3]. Essentially however, the most related works have primarily focused on learning either an optimal task policy (state-to-action mapping) [45, 46, 47], inferring the objective(s) of the task (generalizing the final goal state) [43, 44, 48], or generalizing a task plan (admissible action sequences from initial to goal state) [49, 50, 51, 52, 10].

In contrast, the task-situated symbol grounding problem we aim to solve takes as input one instance of a feasible task plan in the form of a task recipe and learns a situated perceptual grounding for each *parameter* enumerated in the recipe. That is, the learning problem is to create a mapping from concepts in the task recipe to percepts in the environment, in order to enable *execution* of the given task plan in the newly situated environment. Our approach then is complementary to any LfD approach that yields a task plan consisting of parameterized actions or any LfD approach that infers task objectives as this can be used to monitor execution.

Additionally, there have been a number of works focused on learning hierarchical representations of space and groundings for general classes of objects [14, 53, 54, 55, 56]. The goal of this component of the thesis work is not to learn a generalized hierarchy of all physical space or object types within an environment; it is to enable a robot to quickly acquire the information necessary to execute a given task in a new environment, even once the environment has been perturbed. The advantage of our approach is that it allows the learner to use a small number of perceptual features while still characterizing the relevant target concepts, and it enables the human teacher to specify which objects in the environment are most appropriate for performing the given task.

3.3 Interactive Feature Selection

Feature Selection (FS) aims toward the broader goal of dimensional reduction in order to reduce sample complexity for learning a new task. It is a process of selecting a subset of features to be used in task model construction (*i.e.* for building all classifiers needed for the task). Since the number of training instances needed generally increases exponentially with the number of features used to represent a target concept, pruning uninformative features has the potential to greatly speed up the learning process and enable classifiers to more effectively discriminate because of increased robustness to noise in the data. In LfD, this is especially important because robots are typically only provided a small number of examples from the human teacher and thus efficiency in learning is critical.

For the broader applications of robotics and intelligent agents, FS has been previously explored in the literature for several problem domains: mobile robot navigation [57, 58, 59, 60]; simulated autonomous car driving [61, 58, 62]; emotion state classification for a nursing robot [63]; robot soccer and multi-robot domains [64]; gas identification [65] and fire hazard classification [66] for search and rescue; and grasp classification [67, 68]. Nonetheless, these prior works have looked at enabling a robot to automatically select features using computational algorithms with no human in the loop.

In terms of incorporating a human in the loop, there is a body of literature that examines the discovery of discriminative attributes for image classification, generated based upon what is intuitive to human annotators [69, 70, 71, 71]. This work also aims toward the broader goal of improved classification accuracy through dimensionality reduction. However, it does so by asking annotators to provide intuitive attribute labels (*e.g.* “one-handed”, “has spots”) which are essentially higher level descriptors that can be later recognized and used to effectively differentiate between classes in new images. It is analogous to *feature extraction*, or the creation of new features. In robotics however, we aim to solve a slightly different problem. The robot’s sensors will always provide a superset of low-level candidate features, only a subset of which it needs to direct its attention to for learning/performing a given task. Thus, the goal is to enable the robot to decipher which *subset* of features given to it are most relevant for the task at hand.

Towards this end, we also use a human in the loop, and there has been some prior work within the robotics community that looks at requesting feature information from a human teacher. Work by Rosenthal et. al. [72] recommends specific aspects that should be included when asking a question, in order to provide transparency to the human partner, and one of the selection criteria recommended is *feature selection*. In this work, FS is suggested as a follow-up question whereby the robot simply asks *why*, when its prediction is incorrect. It is assumed that robot can parse the open ended response of the human and may elicit high level features from the human partner (e.g. “the correct shape is a cube because it has six sides - all squares”). Embodied feature queries were introduced by [12] for a robot learning a skill from demonstration by a human teacher. The work focuses on enabling a robot to generate three types of queries using its embodiment where the query types each aim to reduce uncertainty with respect to learning a low-level skill controller. In contrast, our focus in this component of the thesis work is to examine the efficacy of different natural language question types towards acquiring the same information: a useful set of features to sufficiently characterize the given task.

The most closely aligned work has explored the use of automatic feature selection for learning a task policy from human demonstrations [73]. The *abstraction by demonstration* algorithm enables an agent to infer relevant features for the human policy based upon demonstrations given. Though this work is similar in that it seeks to learn informative feature subsets through human provided examples, our work on interactive FS differs in two ways: (1) we seek to determine the most effective approach for eliciting feature information *from* human teachers and (2) we compare the efficacy of human-driven FS approaches to computational FS approaches.

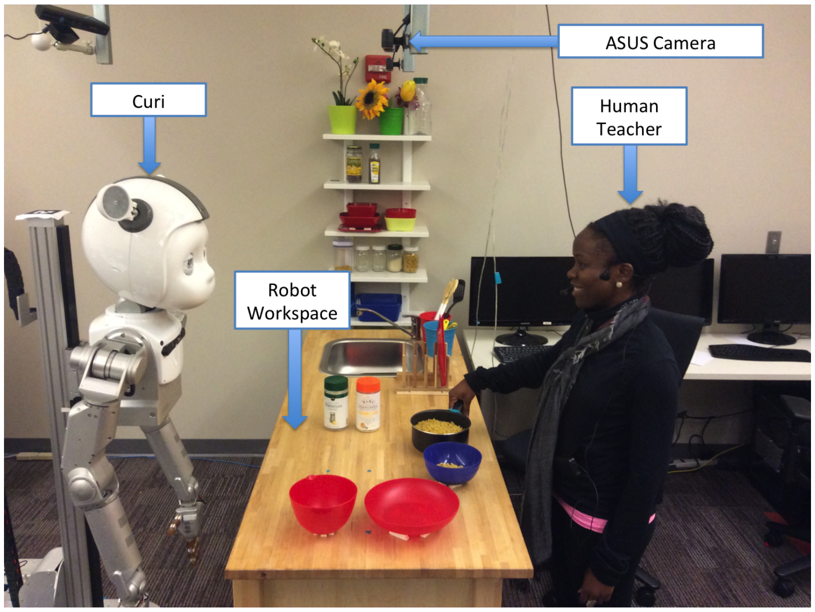
CHAPTER 4
LEARNING FROM DEMONSTRATION FOR TASK-SITUATED CONCEPT GROUNDING

4.1 Introduction

One of the learning problems we seek to address through question arbitration is concerned with task-situated *symbol grounding*. This component of the thesis work explores symbol grounding in the passive learning case, using demonstrations provided by a human teacher. Our findings provide insights about the efficiency and necessity of perceptually grounding task-relevant objects and semantic locations in the situated environment.

The motivation for the work lies in an increased focus on robots taking advantage of cloud computing repositories of high-level task descriptions (recipes) [38, 39, 40, 41]. A task recipe is one instantiation of a task plan and thus provides a sequence of instructions for execution of a task in the situated environment. In prior work however, the robot can only execute the task in environments for which it has existing primitive action controllers. Our approach addresses this problem by breaking down each task recipe into a sequence of parameterized actions; we then learn the perceptual groundings for all unique parameters from a small number of demonstrations of the task recipe in a new environment.

In this work, we use LfD to solve a *task-situated* symbol grounding problem. Symbol grounding is the problem of mapping symbolic representations to constructs in the physical world [14]. In our case, we assume this grounding is *situated* in the context of a particular task. Each task recipe has discrete set of abstract concepts concerning the objects and locations, and our system learns a classifier for each of these concepts, enabling the robot to now perform the abstract task recipe in its specific environment. We validate this approach by showing that a robot can take two abstract kitchen task recipes, and perceptually



(a) Curi's Environmental Setup

Figure 4.1: Environmental setup of Curi's workspace. Includes overhead Asus RGBD camera for perceptual input. Situates the interaction that takes place when Curi recieves a task demonstration.

ground both differently in three different kitchen environments, given only a small number of task demonstrations. Our results provide evidence that it is both feasible and necessary to ground the parameter models in each new environment where execution must take place. We present the following contributions:

- An approach for solving this situated symbol grounding problem given no information about the environment a priori, through the use of an LfD framework which requires only a small number of demonstrations

- An evaluation on image sets from three different environments for two separate tasks, which shows that we are able to successfully ground task-specific *objects* and *semantic locations* in a situated environment

4.2 Approach

Our approach uses human demonstrations to learn perceptual groundings in a specific environment, for each *parameter* referenced within the task recipe. We assume each task recipe is provided to the robot a priori and remains fixed. Our goal is to show these groundings are both (1) capable of being learned efficiently, from demonstrations, and (2) necessary to learn for each newly situated environment.

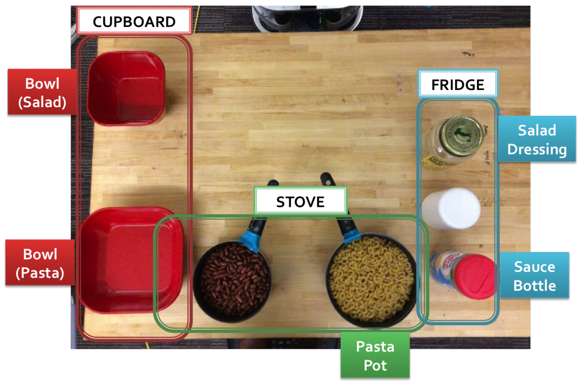
As our running example, we situate a robot within a kitchen setting, learning how to help prepare dinner. We experiment with two task recipes: *serving pasta* and *serving salad*, provided in Tables 4.1 and 4.2 respectively. Each task is to be executed in three different kitchen environments, each with different sets of dishes, pots, etc. and different relative locations for the stove, refrigerator, etc. (Figure 4.2).

Most goal-driven actions employed in such tasks manipulate some type of object and in the process, move or reposition the object in some way. Therefore, we believe there are two types of recurring *parameters* which must be well characterized for successful performance of a task: *objects* and *semantic locations*.

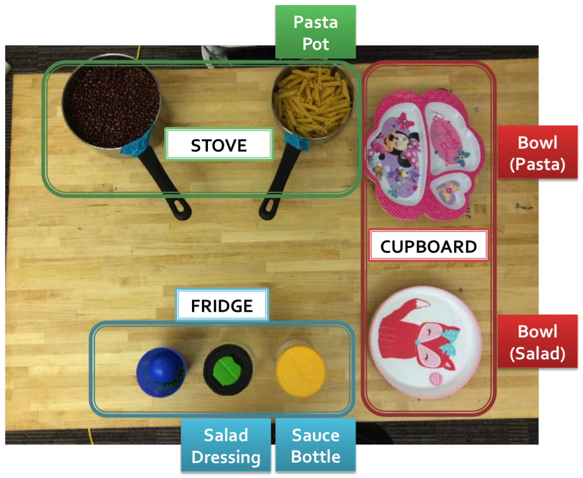
In high-level task learning, we assume the robot is given a library of primitive actions controllers a priori for task execution. In this work, we also assume semantic locations are physical constructs in the environment which remain consistent within a given environment. Given this, the robot must learn which objects housed within the environment should be used, and which physical locations in the environment correspond to the semantic locations specified in the recipe, in order to execute the task.

4.2.1 Representing Tasks and Environments

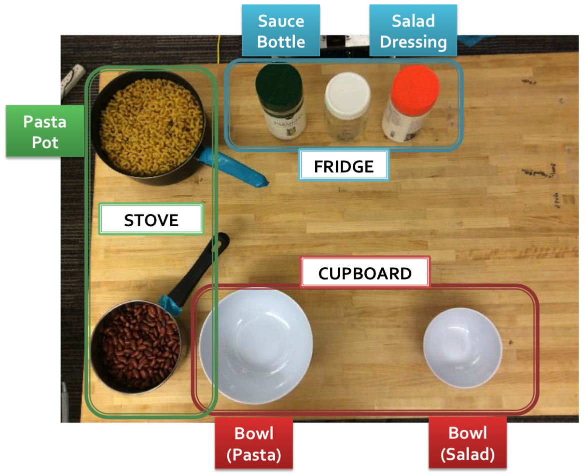
We define a high-level task, $t \in T$, as a sequence of primitive parameterized actions, $a_1, a_2, \ldots, a_n \in A$, where A is the set of actions in the task recipe. Each primitive action $a_i \in A$ contains a set of parameters, P_{a_i}, which define the dimension(s) along which the action can vary. These parameters correspond to *semantic location* parameters or *object* parameters, for any action. For example, the *Serve Pasta* task (seen in Table 4.1) is composed of a sequence of pick and place actions, parameterized as: *pick-place <target-object, start-location, end-location>*. Hence, a task has a set of parameters, such that



(a) Environment 1



(b) Environment 2



(c) Environment 3

Figure 4.2: Illustrates the different *objects* used and the configuration of the *semantic locations* on the workspace, for each kitchen environment.

Table 4.1: SERVE PASTA TASK RECIPE

Action ID	Parameterized Action
1	pick-place <pasta-pot, stove, counter>
2	pick-place <bowl, cupboard, counter>
3	pick-place <pasta-sauce, fridge, counter>

Table 4.2: SERVE SALAD TASK RECIPE

Action ID	Parameterized Action
1	pick-place <bowl, cupboard, counter>
2	pick-place <salad-dressing, fridge, counter>

$\forall a_i \in A,\ P_t = \bigcup P_{a_i}$, representing the collection of all *unique* object and semantic location parameters within the recipe. In this work, we highlight tasks composed only of *pick-place* actions, but the proposed approach can be extended to any high-level tasks with actions parameterized by objects and locations.

Each *object* parameter model, $p_t^o \in P_t$, which characterizes the action's *target object*, is a classifier selecting which candidate object in the workspace is the best match. It is modeled as a mixture of multivariate Gaussian distributions, where the dimensions of the Gaussian represent the perceptual features used to characterize an object. In the context of our running example, the set of *object* perceptual features f_o is composed of color dimensions (red, green, blue) and bounding box size dimensions (length, width, height). Hence, $f_o = (r, g, b, l, w, h)$. Each *semantic location* parameter model, $p_t^l \in P_t$, is a classifier representing where an object should either originate or be placed. It characterizes the object's *starting* or *target location*. It is also modeled as a mixture of multivariate Gaussian distributions, where the dimensions of the Gaussian here represent dimensions of physical space. In our running example, the set of *location* perceptual features is characterized by $f_l = (x, y, z)$. Extraction of all perceptual features is explained in Section 4.3.1.

Each environment $e \in E$ is defined as a collection of *objects* and a specific configuration of *semantic locations* within the physical space. We distinguish environments in two ways: (1) changing where the semantic locations are positioned in the robot's workspace (*e.g.* the cupboard is located in different places in different kitchens), and (2) changing the set of used objects (*e.g.* different homes use different sets of dishes). With each task grounded in a new environment, the robot acquires new knowledge about objects and semantic locations specifically found within that environment and task-specific context for how those objects mays be utilized (e.g. learning which of the bowls stored in the cupboard are appropriate to use for serving pasta). The ground truth objects and configuration of semantic locations defined for each environment are shown in Figure 4.2.

The learning problem then is to build task-specific classifiers for *each* unique *object* and *semantic location* parameter in the task recipe, enabling the robot to generalize different ways of executing the task in the situated environment.

4.2.2 Learning from Demonstration

Demonstrations of the task are collected by observing a human performing the task in the workspace. For each demonstration given, the robot is positioned on one side of the workspace, where the robot and the teacher are positioned directly across from each other, as shown in Fig. 4.1. For each $t \in T$, $t \longleftarrow \{a_1, a_2, \ldots, a_n\}$. The teacher provides verbal cues to denote the start and end of each action in the task; world states, $s \in S$, are then captured as snapshots of the robot's workspace *before* and *after* each action is demonstrated by the teacher. This yields $|S| = 2 * |A|$, where S is the set of world states for the task.

For each pair of world states corresponding to before and after action a_i was performed by the human teacher, s_i^{pre} and s_i^{post} respectively, the system *observes changes* in the world state resulting from the performance of action a_i. The effects of taking action are assumed to correspond to a *visible* world state change, such that $||s_i^{post} - s_i^{pre}|| >> 0$. This results from a change in the *positioning* or *state* of an object; in our running example, a bowl may move from the cupboard to the counter or go from empty to full. Over the course of an entire task demonstration, the system learns the sequence of world state changes corresponding to performing the task.

4.2.3 Building the Parameter Models

We use perceptual data from world states s_i^{pre} and s_i^{post} $\forall a_i \in A$ to learn P_t, the set of parameter models relevant to all actions within the task. When the robot subsequently executes task t in the environment, the parameter models $p_t^o \in P_t$ enable it to infer which objects to select and $p_t^l \in P_t$ guide it as to where in the environment to direct its attention when acting on the selected objects.

The learner receives as input the set S. Each $s \in S$ contains a set of object clusters (like shown in Figure 4.5), with each cluster encoded as a feature vector or point f in feature space F. The distance metric in Equation 4.1 compares each object's state before and after an action has been demonstrated. This enables tracking of the objects being manipulated and the resulting world state changes.

$$dist(o_{pre}, o_{post}) = ||f^{post}(o) - f^{pre}(o)|| \tag{4.1}$$

$$o^* = \arg\max_{o \in O} \{dist(o_{pre}, o_{post})\} \tag{4.2}$$

Each cluster is considered a candidate object o. In Equation 4.1, each $o \in O$, the set of all candidate objects in the robot's purview, is represented by a feature vector $f \in F$. $f^{pre}(o)$ represents the feature values for an object *before* an action was performed by the teacher and $f^{post}(o)$ represents the object feature values, *after* the action. In Equation 4.2, o^* represents the candidate object whose feature values changed the most by demonstration of the action. The object(s) evaluated as having undergone the greatest change is/are selected by the learner and used to update the *object* and *semantic location* parameter models associated with that action. For a *pick-place* action, the *starting location* and *target loca-*

tion parameters are updated with location values corresponding to where o^* originated and where it is placed. Let $f_l(o)$ represent the location feature values of a candidate object; then the *starting location* and *target location* models are updated with $f_l^{pre}(o^*)$ and $f_l^{post}(o^*)$, respectively. The *target object* parameter model is updated with values from both $f_o^{pre}(o^*)$ and $f_o^{post}(o^*)$.

For each $p \in P_t$, with each new task demonstration provided, a new set of m hypotheses is generated, where m is the number of training demonstrations given thus far. Hypotheses are all represented as Gaussian Mixture Models where k is the number of mixture components and with $k = [1, m] \in Z$. Hence the hypotheses range from 1 cluster with m data points in it, to m clusters with one data point each. We use a covariance prior to initialize the covariance matrix for each hypothesis, consistent with the amount of sensor noise observed for objects in the robot's workspace. Bayesian Information Criterion is used to select the best hypothesis.

4.2.4 Task Execution

After receiving training task demonstrations in the environment, the learner has task-specific classifiers for each unique *object* and *semantic location* used in the task. The learner then prepares to execute the task in a previously unseen world state. In this work, we have not yet run experiments on the physical robot. So the learner executes the task sequentially by selecting a candidate object $o \in O$, for each $a \in A$, and verbalizing how to act upon that object (*e.g.* where to place the object).

For each action type with position constraints on its start state (actions which have a *starting location* parameter), the problem of deciding which object to select is a hierarchical classification problem. The learner first identifies which region of the environment to direct its attention; this corresponds to the region that falls within a specified Mahalanobis distance of the action's *starting location* model. Then it only searches candidate objects within the identified region. We empirically determined a Mahalanobis distance threshold,

$\tau = 3.0$, yielding good performance. For semantic location parameters, this was used to evaluate model membership of a selected query point (location of a candidate object) as being positioned within the *starting location* specified for the action being executed.

Candidate objects are then evaluated for selection of the best match to the *target object* parameter model, p_t^o, according to Equation 4.3.

$$\arg\max_{o \in O} \left[p(o | N(\mu_{p_t^o}, \Sigma_{p_t^o})) \right] \tag{4.3}$$

where $N(\mu_{p_t^o}, \Sigma_{p_t^o})$ represents the learned multi-modal Gaussian distribution for the target object. The best matching candidate object $o*$ must also meet the additional criteria that it is a *likely* match to p_t^o. Likely matches are all $o \in O$ that fall within τ of $N(\mu_{p_t^o}, \Sigma_{p_t^o})$.

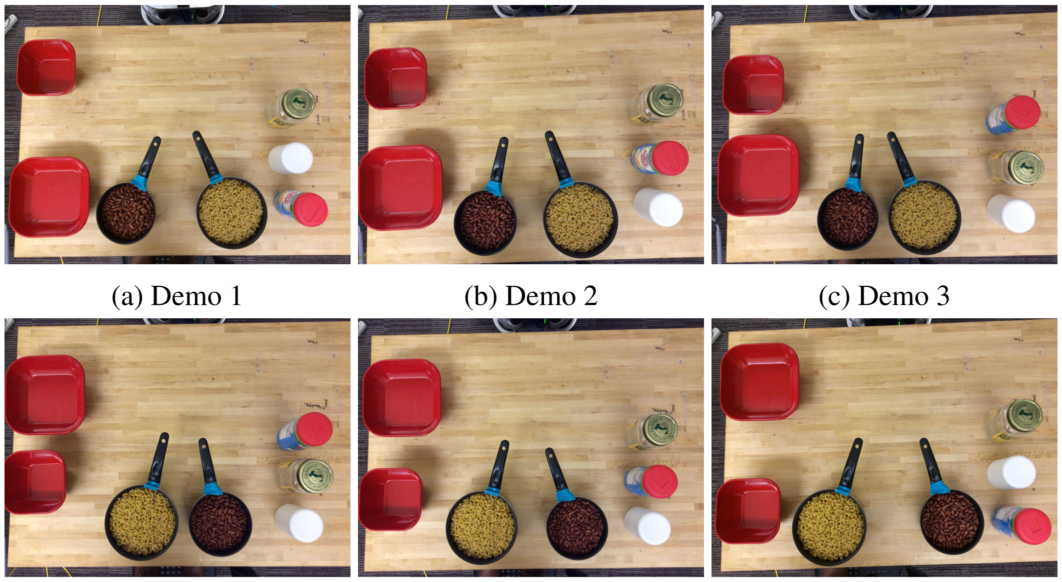
4.3 Evaluation

Our primary assertion in this paper is that in order for a robot to successfully perform a task, for which it has been *given* a recipe, it must ground the task in the environment where task execution will take place. Accordingly we have two hypotheses that we are testing: (1) This grounding can be learned quickly in a given environment using LfD, and (2) this grounding is necessary for every new environment.

In our scenario, this corresponds to the robot learning where the *stove*, *cupboard*, *refrigerator*, and *counter* are located and what a *pasta-pot*, *bowl*, *sauce-bottle*, and *salad-dressing* look like specifically for executing the tasks in each environment. Additionally, we expect the learner to concurrently learn models for each of these parameters, from only a small number of task demonstrations.

4.3.1 Environmental Setup

The environmental setup can be seen in Figure 4.1. Tasks are demonstrated on a countertop workspace. An overhead ASUS Xtion RGBD camera is pointed to the workspace. The


(a) Demo 1 (b) Demo 2 (c) Demo 3

(d) Demo 4 (e) Demo 5 (f) Demo 6

Figure 4.3: Overhead view of all initial states for the set of training demonstrations under one lighting condition, given in Environment 1 of the *Serve Pasta* task.

robot, Curi, is a mobile upper-torso humanoid with 7-DOF arms, 4-DOF hands, and a socially expressive head. In this work, the robot does not execute actions but serves to situate the teacher-learner interaction.

The RGBD camera is used to track and extract information about objects. We segment the objects above the countertop using the method presented in [74], which identifies planes based on connected-component labeling (CCL) of the normals of the point cloud data and finds spatial clusters based on CCL of distances above the selected plane. After this segmentation, we fit a rotated bounding box to the object[1]. Figures 4.5a, 4.5b, and 4.5c illustrate the output of the perceptual pipeline for world states from each environment, corresponding to images shown in Figure 4.2.

To characterize candidate objects in the workspace, we define the feature space F as a 3-tuple of $\langle color, size, location \rangle$. Color is composed of the *(r)ed*, *(g)reen* and *(b)lue* channels of the object segment in the image; size is characterized by *(w)idth*, *(l)ength* and *(h)eight)* of the fitted bounding box; and location is composed of the coordinates of the centroid

[1]We assume the rotation is only with respect to the countertop normal

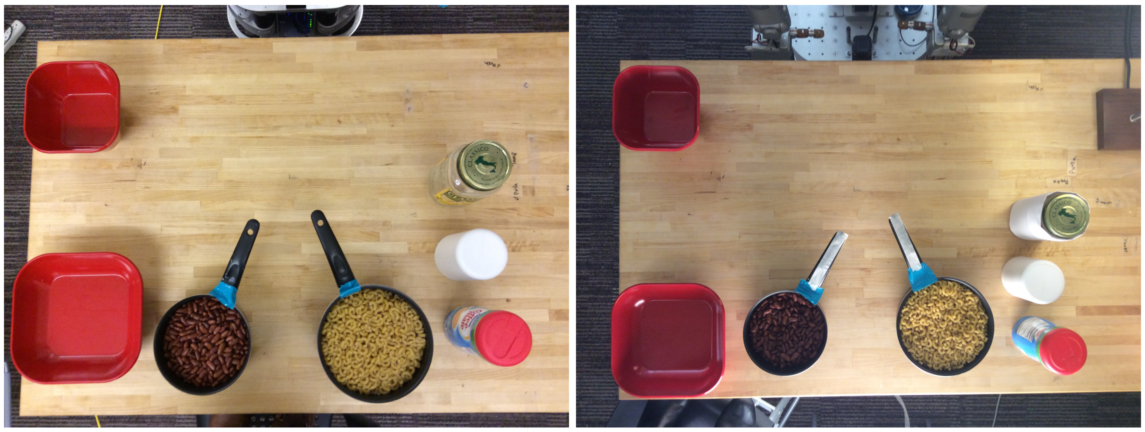
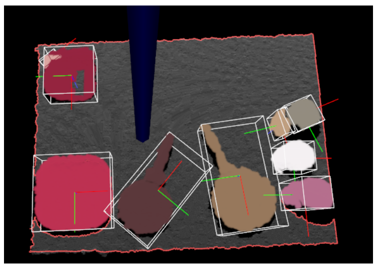

(a) Demo 1 (b) Demo 7

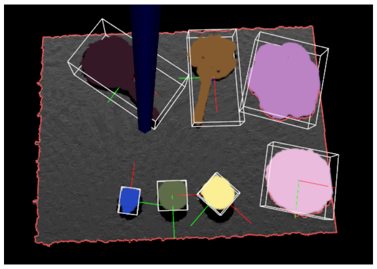
Figure 4.4: Illustration of different lighting conditions for corresponding task demonstrations in *Serve Pasta* task in Environment 1.

of the object's bounding box, with respect to the robot. So for each cluster, we extract a feature set $f = \{r, g, b, w, l, h, x, y, x\} \in F$, given as input to the learner.

4.3.2 Task Environments

To test our hypotheses, we conducted a set of experiments in different environments. We define three distinct environments in which to perform the task. One of the authors provided demonstrations of both tasks, in each of the three environments. For each environment, a specific set of objects and configuration of semantic locations was selected to be the ground truth. The workspace was systematically perturbed to yield a small change in edit distance between any two adjacent task demonstrations; nonetheless, the demonstrations had significant variance over the course of the entire set of training demonstrations in a given environment. Figure 4.3 shows the initial state for each of the first six *serve pasta* task demonstrations given in Environment 1. Perturbations within one environment represent variance in how the same task may be taught in one setting. Importantly, semantic locations remain consistent over the course of all demonstrations within the environment (*e.g.* the cupboard always remains on the right of the robot's workspace in Environment 1), and objects must be placed in the correct semantic location (*e.g.* bowls must always originate


(a) Environment 1


(b) Environment 2

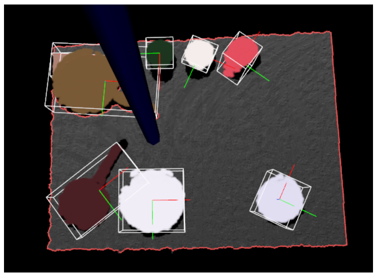

(c) Environment 3

Figure 4.5: Illustrates the perceptual clusters for *objects* used in each of the kitchen environments.

in the cupboard), but may vary as to where and how, within that location, they are placed.

The second and third environments are representative of the robot performing the task in a different setting altogether. We select distinctly different sets of objects for fulfilling the same purposes, as would be expected in three different homes. We also change where the semantic locations are positioned on the robot's workspace. Figure 4.2 shows the setup for demonstration 1 of environment 1 on the left, environment 2 in the center, and environment 3 on the right. The demonstrations given in Environments 2 and 3 were perturbed in a similar systematic way as illustrated for Environment 1. Additionally, for the *serve pasta* task, we varied the lighting conditions such that the positioning of objects in demonstrations 7 through 12 correspond to those in demonstrations 1 through 6 but under different lighting conditions, as shown for demos 1 and 7 in Figure 4.4.

4.3.3 Data Collection

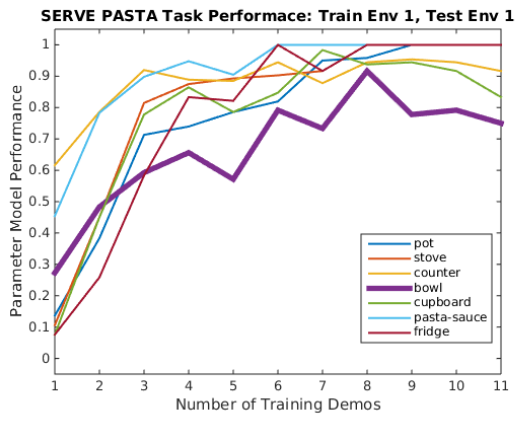
Task demonstrations were performed in the robot's workspace, using the setup detailed in Subsection 4.3.1. Each task demonstration consisted of a sequence of every action in the task recipe. Each action demonstration changed the location of one object, the *target object* of the action. It is important to note that in our experiments, the same set of candidate objects remained on the robot's workspace throughout an entire task demonstration. Therefore the change, corresponding to one action, was always a change in *positioning* for one of the existing objects, not a change in the number of objects in the task.

4.3.4 Evaluation Metrics

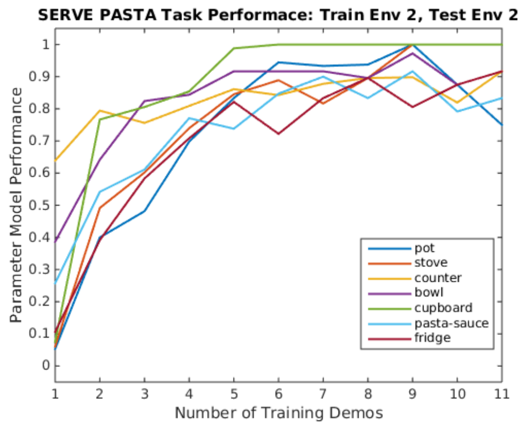
In order to assess the robot's performance in task execution, we evaluate each of the object and semantic location models separately. For a *semantic location* parameter, the goal is to assess how well the model characterizes the semantic location where the *target object* is expected to reside. Since the robot uses the learned location model as a guide for directing its attention to where it should be searching for the object of interest, the *target object* must be contained within the set of objects positioned inside of the location, in order for the *location* parameter model to perform successfully. For an *object* parameter, the goal is to assess how well the model characterizes the target object needed to achieve the goal of the action. This model is used for object selection. In order to assess the *object* parameter independently of its semantic location, the model considers all objects in its purview and selects the object that has the highest likelihood of being the *target object*. The object selected must meet the additional criteria that it is within a specified Mahalonobis distance of the target object model; this ensures that the best matching candidate object is also a *likely* match to the target object. If the candidate object selected is both a likely match and has the ground truth label associated with the target object model, the *object* parameter model performs successfully.

4.4 Results

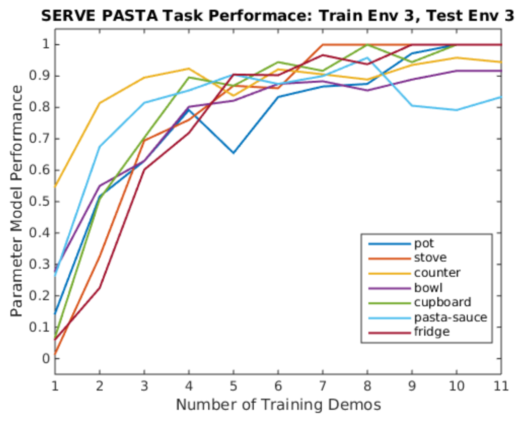
Below we detail the two different analyses performed on the experimental results: (1) performance of environment-specific task models, within the environment trained *and* (2) generalization of learned task models, across different environments. We expect to observe the set of task models trained and tested within one environment to outperform the other sets of model tests; furthermore, we expect that learned models will *only* perform well when they have been trained in the situated environment. This means that we are not expecting transfer of learned task models to a previously unseen environment since environments may



(a) Environment 1



(b) Environment 2



(c) Environment 3

Figure 4.6: Within-Environment Cross Validation. Performance for each parameter model for the *Serve Pasta* task, in each environment, tested in the *same* environment. Based on upon number of training demonstrations. *Best viewed in color.*

vary so vastly; nonetheless we aim to show that LfD enables the robot to learn the task in the situated environment at relatively little cost to the human teacher. The learning metric we use is task performance, as described in detail in the previous section.

4.4.1 Learning Model Parameters for a Specific Environment

In our first analysis, we perform within-environment cross-validation on the *serve pasta* task, for each environment $e \in E$. For the serve pasta task, a total of 12 task demonstrations were provided under two different lighting conditions, as explained in subsection 4.3.2. Let n represent the total number of task demonstrations in the dataset for one environment (so here $n = 12$) and let m be the number of *training* task demonstrations used to learn all parameter models needed for task t. Each task model t_i consists of the set of parameter models P_t, uniquely identified in the task recipe. In order to train each $p_{ti} \in P_t$, we uniformly randomly sample m task demonstrations, where $m < n$. Then $\forall p_{ti} \in P_t$, p_{ti} is tested on the remaining $n-m$ unseen demonstrations. To generate the learning curve as a function of m: $\forall m$, we train and test 12 task models.

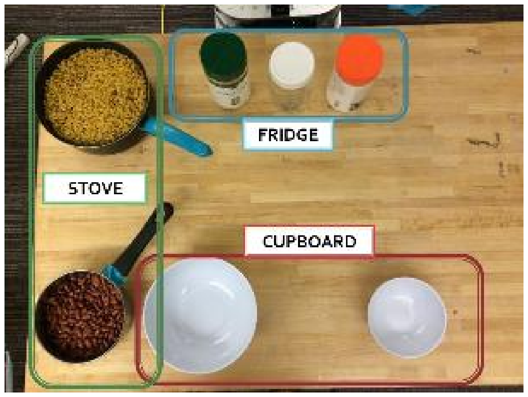
Figure 4.6 shows average learning performance as a function of the number of training demonstrations given, in each environment. Each colored line represents the performance of an individual task parameter grounded in the environment. There are a total of seven task parameters in the *Serve Pasta* task: (1) three *object* parameters – *pasta pot*, *bowl* (larger bowl for pasta), and *sauce bottle*; and (2) four *semantic location* parameters – *stove*, *cupboard*, *refrigerator*, and *counter* space. The learning curves are not smooth since we only trained and tested a small sample of the total number of possible task models that could be learned for each value of m.

Nonetheless, the learning curves exhibit a clear and consistent trend for both *objects* and *semantic locations* grounded in all three environments. Most of the learning curves begin to reach their peak performance after approximately 5-6 task demonstrations have been provided. Some classification error does occur however. For some of the curves, even as they reach the maximum number of training demonstrations that could have been provided from the dataset, performance is not optimal. The highlighted parameter model for the *bowl* in Environment 1 (purple) is an example of this. When $m = 11$, representing the case of leave-one-out cross validation, the pasta bowl classifier fails to identify the bowl used for pasta in three test cases out of 12 attempts. In all of the cases, it selects the correct object (the large red bowl) as being the closest match to the learned distribution; however in all three cases, the object selected as being the closest match slightly exceeds the Mahalonobis distance threshold of 3.0 (all had distances of less than 4.0, two were even less than 3.5). This means that although it was able to select the correct candidate object as being the closest match to the target object, amongst all candidate objects in the robot's purview, it did not have enough confidence that the closest matching candidate object was also a *likely* match. Therefore in all three cases the final output was that *no match* was found. In two of the cases, it is because the lighting conditions cause the red bowl to appear too bright to be a likely match. In the third case, the height of segmented bounding box for the bowl was too large, so the selected object appears to be too tall to be likely match. This example highlights key challenges faced grounding the different parameters even within one task: some learning problems are inherently harder than others, and the effects of sensor noise are difficult to completely overcome.

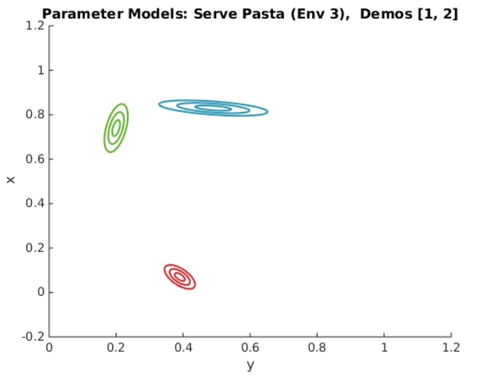
Overall however, the learning curves for the parameter models are aligned with what we would expect. All of the models were able to perform quite well with only a small number of training demonstrations; hence supporting our hypothesis that the grounding of all parameters for a task executed in a new environment can be learned quickly.

To that point, in some instances, the robot is able to learn a model that performs well with only two task demonstrations. We hypothesize that this is correlated with how varied the training demonstrations given are. This can be quantified as the maximum environment edit distance between any pair of task demonstrations used to train the task model. Environment edit distance of two task demonstrations d_a and d_b is measured using Equation 4.4, where O_a and O_b represent the set of candidate objects observed in d_i and d_j respectively and $dist(o_i, o_j)$ is defined in Equation 4.1.

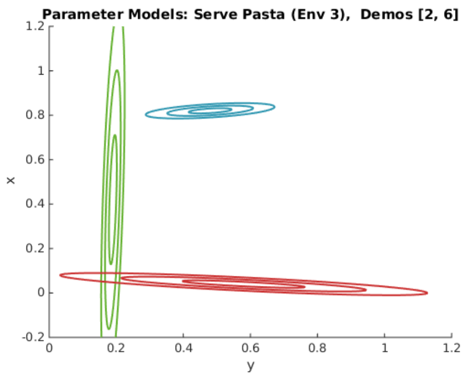
$$dist(d_a, d_b) = \sum_{o_i \in O_a} \sum_{o_j \in O_b} dist(o_i, o_j) \tag{4.4}$$



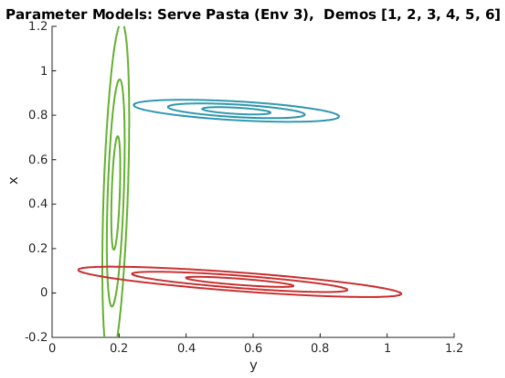
(a) Example World State



(b) Demos [1,2]



(c) Demos [2,6]



(d) Demos [1,2,3,4,5,6]

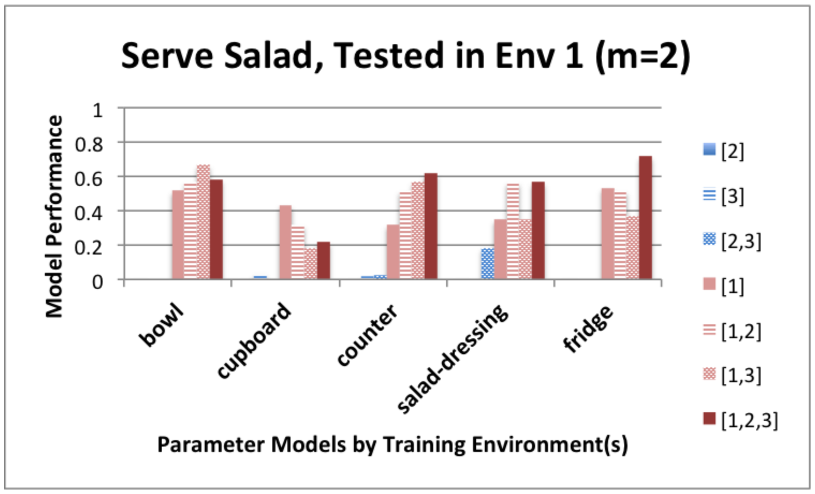
Figure 4.7: Location Models learned for stove, cupboard, and refrigerator. Trained with Demos listed. *Best viewed in color.*

In the sets of demonstrations, since the *location* of objects was the only feature we could systematically vary with each demonstration, we examine the location parameter models in Environment 3, shown in Figure 4.7a, more closely. Figures 4.7b and 4.7c show two sets of location models trained using only two task demonstrations. The demonstration numbers shown below each graph are the set of training demonstrations used to build the models shown and correlate with the images shown in Figure 4.3. The *semantic location* model for the stove is learned from the positioning of the pasta pot, the *semantic location* model for the cupboard is learned from the positioning of the bowl used for pasta (large white bowl in Env 3), and the *semantic location* model for the refrigerator is learned from the positioning of the sauce bottle (dark green bottle). Therefore, looking at the images, we can see that task demonstrations 1 and 2 have a relatively small edit distance, whereas task demonstrations 2 and 6 have a comparatively larger edit distance. Interestingly, the models in 4.7c are able to characterize quite well the semantic locations, shown in the image above the models in 4.7a. The variance is a little larger for both the stove and cupboard since the models have only seen examples of the pot and bowl placed near the edges of counter, and the fridge model variance is noticeably smaller since it has only seen two of the three general areas where the pasta sauce is typically placed. However, even with that, the location models for the stove and cupboard trained using demos 2 and 6 capture essentially the same allowable variance as do the corresponding models in 4.7d, trained using all six task demonstrations from that lighting condition. The same could be observed for the fridge model had we trained using demos 3 and 6, for example. So although the learning curves illustrated that on average after approximately 5-6 demonstrations, the robot will have acquired a perceptual grounding for each of the parameters in the task recipe; here we see that if the teacher is able select informative examples, training using LfD is even less costly, requiring less than half of the number of demonstrations required on average.

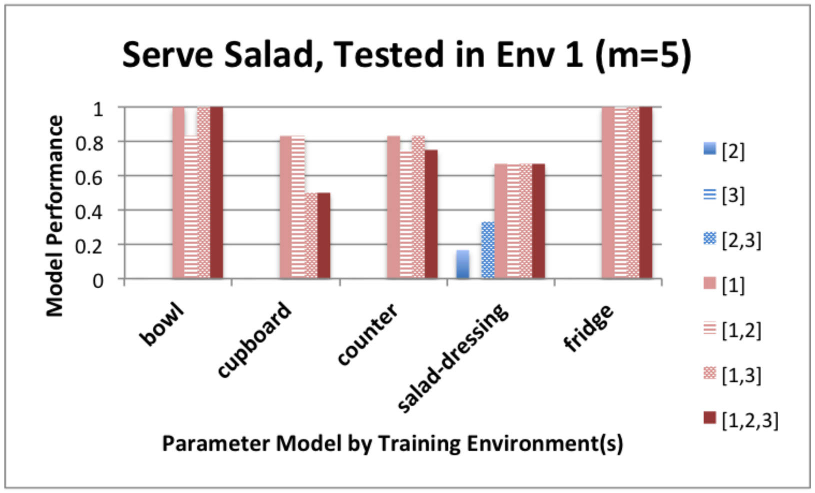
4.4.2 Transfer of Learned Models Between Environments

The next analysis examines the extent to which these models transfer between environments. Since one of our hypotheses is that the task grounding is necessary for every new environment, it was important to assess whether learning the task in one environment was sufficient for successful performance of the task in a new environment. We also examine the performance of task models trained across multiple environments, referred to as *aggregate* task models. Some aggregate models include demonstrations from the environment where the robot needs to execute the task (*i.e.* the test environment) whereas others do not.

Whether looking at the performance of task models trained in a single environment or aggregate models trained in multiple environments though, the goal here is to assess whether it is necessary for the learned task model to include demonstrations from the environment where the robot is currently situated and needs to execute the task.



(a) Parameter models learned with 2 training demos



(b) Parameter models learned with 5 training demos

Figure 4.8: Comparison of Learned Models Trained in different subsets of all kitchen environments. Performance for each parameter model for the *Serve Salad* task. All Tested in Environment 1.

In this analysis, we used between-environment cross-validation on the *serve salad* task, for each environment $e \in E$. For the serve salad task, a total of 6 task demonstrations were provided, using only one of the lighting conditions. For the task models trained in a single environment, we train task models (sets of parameter models) for every combination of $m = [1,5]$ task demonstrations. For the aggregate models, we provide every combination of m demonstrations, in *each* training environment; therefore all aggregate models are composed in total of $m * |E_t|$ training demonstrations, where E_t is the set of environments used for training a task model t. Here $n = 6$ and again each task model is tested on new world states taken from the remaining $n - m$ demonstrations. Though we tested each task model trained (from individual environments as well as aggregated across multiple environments) in every $e \in E$, in this analysis, we focus only on a case study where task models are tested in Environment 1, using either two or five training demonstrations. Figure 4.8 shows the average performance of each set of parameter models, all of which are to be used for execution in Environment 1 where the robot is presumably situated. These bar graphs compare the performance of all models that were trained in the execution environment *against* those models that exclude the environment where execution is to occur.

Specifically, the blue bars all represent models *not* trained in the situated environment, whereas the red bars all represent models that include demonstrations from the situated environment. Each pattern within a color denotes a different combination of environments where training of the task occurred. These graphs clearly demonstrate that whether trained with a very small number of task demonstrations or a larger number, *all* parameter models trained in the environment where execution is to occur significantly outperform *any* of the models learned in a subset of environments that does not include the situated environment. Presumably because the variance between environments can be substantial and thereby makes it difficult to transfer learned parameter models across environments. This finding supports our hypothesis that it is necessary to ground the task in each new environment where the robot is situated and operating.

4.5 Discussion

Given that a robot may have no information about its situated environment a priori, the goal of this work was to show that LfD is a powerful framework for enabling the robot to quickly acquire the environment-specific perceptual groundings needed to execute a task. This may be counter-intuitive, given that the primary goal of most machine learning applications is generalization; however, we claim that it is a practical approach for enabling a robot to become operational in a new environment quickly, at little cost to the human. An advantage of the proposed approach is that it does not require general classifiers for each relevant object class or a complete semantic map of the situated environment; it is able to use simple perceptual features and still perform robustly since it leverages the expert knowledge of the human teacher and the contexts of the environment and task.

As future work, we plan to conduct a user study with naive users on the physical robot, in order to assess: (1) how intuitive it is for people to teach these groundings, (2) how many demonstrations are required from naive users for the learner to build a model that accurately characterizes relevant semantic locations and the task objects, and (3) how successfully a robot is able to execute the task recipe in the situated environment given the learned groundings. We are also interested in enabling the robot to be an active learner in the interaction. We have seen in our analysis that providing informative examples reduces the amount of time that the teacher needs to spend providing demonstrations and still achieves the goal of generalizing how to execute the task in the situated environment; this can also be accomplished by a robot that actively requests informative examples from the teacher, with the goal of better generalizing its own mental model.

4.6 Conclusion

We have presented a framework that enables a robot to efficiently learn to ground higher-level tasks in new environments, given the task recipe a priori. Since personal robots work in direct collaboration with people they are able to leverage the expertise of their human partner, and acquire demonstrations that provide them with information about two essential types of parameters necessary to successfully perform a task in a real-world setting: *objects* and *semantic locations*. Our approach seeks to solve this *situated* grounding problem; it learns perceptual groundings for each unique parameter in the given task recipe, in order to generalize how to perform the task in the situated environment. We collected demonstrations of two tasks, in three different environments, representing three different kitchens, and were able to show (1) that this grounding can be learned quickly in a situated environment and (2) that this grounding is necessary for every new environment.

CHAPTER 5
CONCEPT GROUNDING USING HUMAN-DRIVEN FEATURE SELECTION

5.1 Introduction

The other learning problem we seek to address through question arbitration is concerned with task-relevant *feature selection*. This component of the thesis work explores different strategies for extracting feature information from a human teacher, for the purpose of determining which features are most appropriate for representing the task. Our findings provide insights about how a robot should request feature information in order to help it solve the feature selection problem for a task it is learning interactively.

This work is motivated by the fact that research on robot learning from demonstration (LfD) focuses on the development of robots capable of learning a wide range of tasks from a small number of human demonstrations. Much of the work in this field is particularly aimed at the development of general-purpose robots capable of performing multiple tasks, such as a household robot able to put away groceries as well as cook a meal, or a service robot able to execute multiple maintenance procedures. Most research in this area assumes that a set of features representing the state of the robot and the surrounding environment are available to the robot, and that these features are then applied to learning new actions (e.g., open cabinet) or new task models (e.g., make coffee) [3].

The state features available to the robot define the variables on which the learning computation depends. However, little prior work considers feature selection in the context of deploying a general-purpose robot able to learn new tasks. Given a new set of demonstrations, which features should the robot use to learn? The feature selection problem is substantial because a general-purpose robot may have the ability to track dozens, or even hundreds, of potential features in its environment, only a handful of which are likely to be relevant for any given task, and incorporating too many unnecessary features leads to poor learning performance. Computational feature selection techniques [75, 76, 15], which rely on identifying statistical patterns in data, may not have sufficient evidence given the small number of training examples encountered in LfD. In fact, in prior work, all LfD papers we surveyed used hand-coded state features, with the exception of [73], in which computational feature selection techniques are applied to identify relevant features based on human demonstrations in the games Frogger and Pong.

In this work, our goal is to explore interactive feature selection in which a robot can identify relevant features with the aid of a human user. Humans familiar with a target domain typically have the ability to characterize which features are important in decision making, at least at an abstract level. We explore whether non-expert users are able to identify which features are most informative for discriminating between classes of objects needed for a given task, how best to elicit the feature information from the user, and how computational feature selection compares to human-driven feature selection given varying amounts of data. Specifically, we explore two research questions. First, is a domain expert able to identify a subset of features that will enable the robot to classify unseen examples as accurately as using computational feature selection? Second, does the way in which information is elicited from the user impact the quality of resulting feature selection?

To address the second research question, we developed three general categories of approaches for allowing humans to communicate feature information to the robot:

1. **Direct Communication** - the user *directly* communicates about features useful for the task, either by selecting or eliminating, from a superset of candidate features

2. **Indirect Inference** - the human teacher selects a small number of instances from each task-relevant object class and the robot learner *indirectly* infers which features are being communicated by the examples shown

3. **Combined Approach** - the human teacher *both* selects a small number of instances

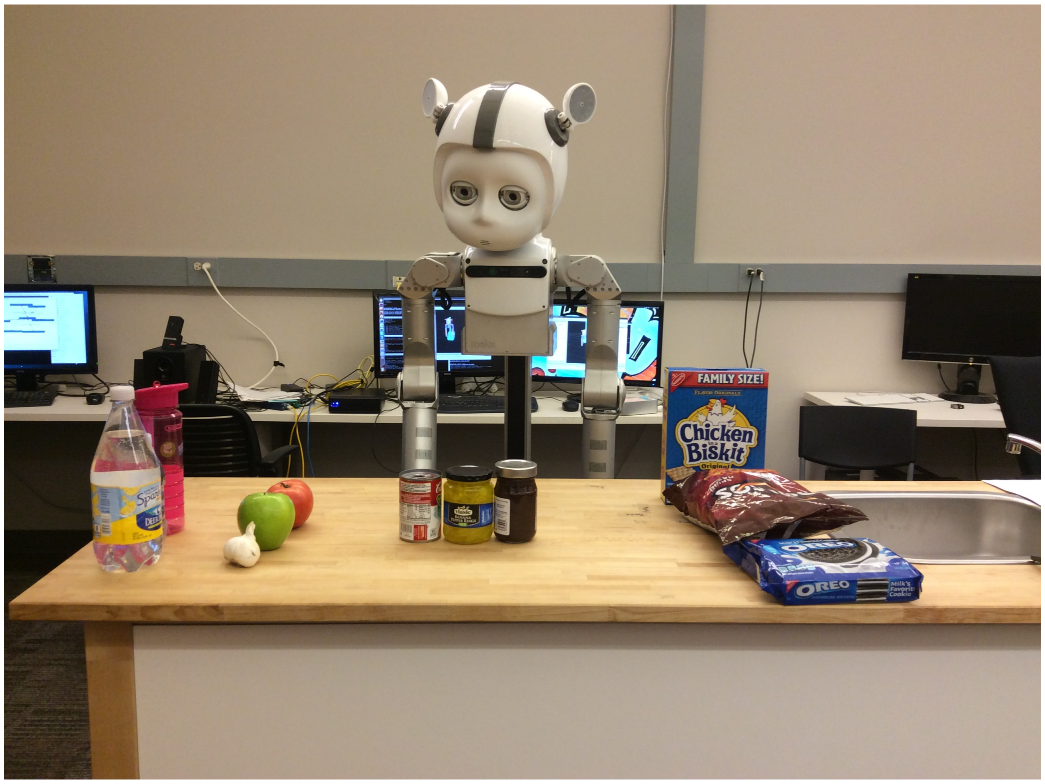


Figure 5.1: Instances selected by user study participant to teach robot about the specified classes of objects in *groceries* task.

from each object class as examples *and* subsequently chooses features being communicated from a superset of candidate features

To study both research questions we conducted a between-subjects user study with 30 participants. Participants were asked to help a robot discriminate between four classes of objects needed for a household task, by communicating about informative features using one of the interaction strategies above. Our findings show that (1) the performance of human feature selection is on-par with computational methods for domains in which they have prior knowledge, (2) *direct communication* is the most effective strategy for eliciting feature information from users when the task features are intuitive, and (3) when a relatively large amount of training data is available, asking a human teacher to first select a small number of informative instances and then *indirectly* inferring the features being communicated leads to the best performance. We also conducted a follow-on study in three additional task domains to examine how reliably users directly communicate informative features; the supplemental findings show that people are able to select useful features for a task only when the features are semantically interpretable.

5.2 Learning Task

5.2.1 Problem Statement

The problem of learning task features consists of determining a subset of features for use in building all of the classifiers needed for the task [3]. In our problem formulation, the robot is given a set of task-specific labels to be learned, Y, and the superset of all candidate features, F, associated with the observed state of the world. The robot's goal is to learn how to classify instances of all the labels Y. Feature selection then, either by using a computational approach or by asking questions of a human teacher, can help the robot determine a subset of features $F' \subset F$ that represent a single state space appropriate for all classes Y.

5.2.2 Problem Domain

As our running example, we situate a robot within a kitchen setting, learning to help sort and put away groceries, as shown in Figure 5.1. We define the *sort groceries* task as teaching the robot to distinguish between four object classes (produce, snacks, food cans & jars, and beverages). For each object instance encountered by the robot, we compute the superset of all candidate features F based on perceptual information extracted from a RGB-D image of the object, the object's relative location to the robot, the robot's joint position information at this time and audio input from the environment. Table 5.1 lists the feature categories and the number of features each decomposes into, for a total of 84 low-level features. The robot's goal is to determine which of the listed features are relevant to the sort groceries task.

We use the University of Washington RGB-D Object Dataset to obtain a standard set of object images for testing [77]. The object dataset includes over 200,000 images in total, encompassing over 300 objects organized into 51 categories (*e.g.* can), with multiple object instances per category (*e.g.* pepsi can, mountain dew can, etc.). For each object instance, the database contains several hundred images captured from different viewpoints and distances from the camera, and some objects in the dataset have been captured under more than one lighting condition. For the sort groceries task, we consider only images related to produce (fruits and vegetables), snacks (food bags, food boxes, and cereal), food cans & jars, and beverages (water bottles and jugs). In addition to using the images, we also purchased approximately 60 objects from the dataset to use in the user study.

5.3 Computational Feature Selection

In this section, we briefly discuss established computational feature selection methods and establish a baseline by examining how feature selection impacts learning performance in our domain.

5.3.1 Algorithm Overview

Feature Selection (FS) aims to eliminate irrelevant and redundant features such that $g(F) \rightarrow F'$, where g is the feature selection function. In selecting feature subsets, features are typically evaluated for *relevance* or *usefulness* [75, 76]. There are three classes of computational approaches for automatic feature selection that have been explored in the literature: *filters*, *wrappers*, and *embedded methods* [15]. Of those, we used filtering and embedded algorithms since they are the most computationally efficient. In terms of classifier representation, after validating learning performance with three different classifiers (k-nearest neighbors, support vector machines, and random forests), we observed the best performance using an SVM classifier with a radial basis function kernel. Thus, SVMs used for all learned models in this work. Below we briefly introduce the feature selection techniques employed. All algorithm implementations were obtained from the Weka Software Library

Table 5.1: High-Level Task Features *(per object instance)*

absolute location (3-dimensional) of object in environment (3)
orientation (yaw) of object on surface (1)
location of object relative to five specified interest points (15)
location of robot's base in environment (3)
pose of robot's two hands (location + orientation quaternion) relative to its body (14)
pose of robot's two hands (location + orientation quaternion) relative to counter (14)
orientation (yaw) of robot's base on ground (1)
robot hand states (open or closed) (2)
position for each joint of robot's 7-dof arms (14)
average color of object (3)
object bounding box size measurements (3)
area of object bounding box (1)
volume of object bounding box (2)
aspect ratio for object bounding box (1)
surface area to volume ratio for object bounding box (1)
compactness of object point cloud (1)
number of SIFT features (measure of visual texture) (1)
max/min/average volume of noise in environment over duration of learning interaction (3)
weight of object (1)

[78].

Filtering

Filters take as input the training data and examine the *relevance* of each feature $f \in F$ as it pertains to each class label $y \in Y$. The **filtering** algorithm **(FI)** employed ranks features based upon information gain IG and selects all features $f \in F$ such that $IG(Y|f) > \tau$ where $\tau = 0$. Thus $\forall f \in F$,

$$I(Y|f) = \sum_{y \in Y} H(y) - H(y|f) \tag{5.1}$$

where $H(y)$ represents the entropy of variable y.

Embedded Methods

Embedded methods conduct a best first search through the space of feature subsets, evaluating the *usefulness* of each subset $F' \subset F$ with respect to a given predictor. We use two embedded algorithms in this work: **embedded selection (ES)** begins with no features and incrementally adds (*forward selection*) whereas **embedded reduction (ER)** begins with all features and incrementally prunes (*backward elimination*). Both embedded algorithms employed use a greedy search strategy and evaluate subsets based upon two metrics: predictive ability of each feature and redundancy between them. **Embedded selection (ES)** begins with no features and incrementally adds (*forward selection*) whereas **embedded reduction (ER)** begins with all features and incrementally prunes (*backward elimination*). The algorithm used to evaluate $F' \in \mathscr{P}(F)$, the powerset of F, was introduced in [79] and is given by Equation 5.2.

$$M_F = \frac{m\overline{\tau_{cf}}}{\sqrt{m + m(m+1)\overline{\tau_{ff}}}} \tag{5.2}$$

where M_F represents the merit of F' containing m features, $\overline{\tau_{cf}}$ represents mean class-feature correlation, and $\overline{\tau_{ff}}$ represents mean feature-feature intercorrelation. The correlations are averaged $\forall f \in F'$.

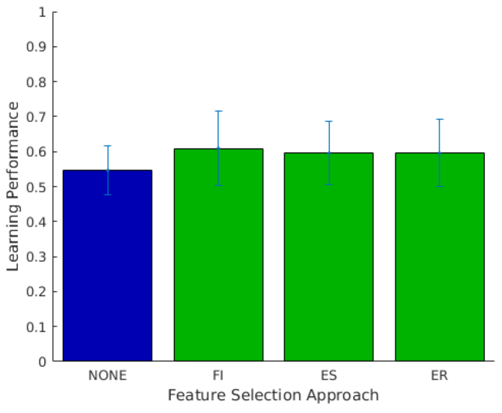
5.3.2 Evaluation

To evaluate the effect of feature selection on our chosen domain, we compare the performance of the above algorithms and an SVM classifier on the sort groceries task. For the evaluation we create a test set, P^{test}, containing 1000 task-relevant images sampled without replacement using stratified random sampling (SRS) from the object dataset. The training set, P^{train}, is similarly sampled to generate $k = 10$ disjoint training samples, $D_{i,\ldots,k}$, each consisting of n sampled images. We use the following evaluation metric to evaluate algorithm performance:

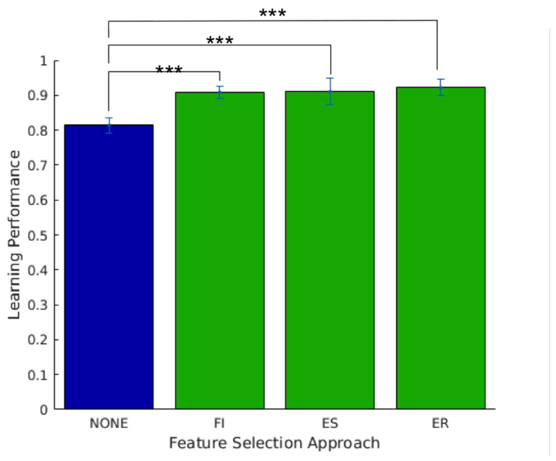
$$E[acc_{\mathbb{D}}(a)] = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{n}\sum_{x\in D_i}[1-\delta(h_i^a(x),y)] \tag{5.3}$$

where $E[acc_{\mathbb{D}}(a)]$ represents the learning accuracy using feature selection approach a with respect to distribution $\mathbb{D}$, $h_i^a(x)$ is the hypothesis of the learner using a given instance x in training set D_i, y is the ground truth label for instance x, and the quantity $\delta(h_i^a(x),y)$ is 1 if $h_i^a(x) \neq y$ and 0 otherwise.

We wanted to test the FS approaches, given a small and large amount of training data. Figure 5.2 reports the results for $n = 12$ and $n = 100$ in order to simulate each scenario. In both conditions, feature selection aids with learning performance. However the difference in expected performance between a learner using computational feature selection and a learner using no feature selection is dependent upon the amount of training data observed. When there is a relatively large amount of training data, as depicted by figure 5.2b, using computational feature selection yields statistically significantly less error than using none, no matter which approach is employed. When only a small amount of training data is available, overall classification performance is lower and there is no statistically significant difference between the expected error of learners with no feature selection and learners



(a) Train Set Size = 12



(b) Train Set Size = 100

Figure 5.2: Learning performance of computational FS algorithms for classification of objects in *Groceries* task. Test Set Size = 1000.

with computational feature selection.

Hence what we observe is that computational approaches are limited in their ability to improve learning performance when there is a small amount of data, since these approaches are data-driven. Nonetheless in LfD, it is commonly the case that a robot is provided only a small number of examples from the human teacher and needs to leverage these in order to learn the task. This motivates the need for other techniques for acquiring a subset of discriminative features where there is limited training data available.

From this point forward, we use only the ER algorithm as our baseline computational feature selector, since it narrowly outperforms other computational approaches as the amount of data increases.

5.4 Human-Driven Feature Selection

In this work, we are interested in enabling a robot to characterize the essential features of a task when there are still very few (or even no) examples to observe. Given limited data, being more selective about features helps the robot better discriminate between the object classes relevant to the task. We hypothesize that humans with task domain knowledge can help the robot by providing information about what features they believe to be most informative for the task.

We compare five approaches to determine how human teachers can best aid with the feature selection problem. We group the techniques into three categories based upon interaction style and the type of information they provide: (1) *direct* communication of features, (2) *indirect* inference of features, and (3) *combination* of indirect and direct communication of features. Table 5.2 shows the source for training instances and selection of feature subsets, using each category of FS approach.

5.4.1 Direct Communication of Features

We wanted to examine whether the way in which features are requested impacts learning performance. Therefore, we explore two *direct communication* approaches for eliciting the information: (a) **human feature selection (HFS)** and (b) **human feature reduction (HFR)**. For both approaches, the human teacher is provided a list of hierarchically arranged candidate features and has the option to choose entire (sub)categories of features to indicate that every feature in the set should be marked or alternatively only choose the individual features within the hierarchical category that are appropriate (*e.g. size features of object*: *volume*, *surface area*, *length*, *width*, and *height*). For human feature selection, the teacher's goal is to provide *only* features they believe to be most useful for the robot in determining which class an unseen object belongs to. In contrast, for human feature reduction, the teacher's goal is to specify features the robot learner should *not* pay attention to (*i.e.* features it should *ignore*) because they will *not* help it determine the class of an unseen object.

5.4.2 Indirect Inference of Features

There is one *indirect inference* approach that we explore: **human instance selection (HIS)**. With this approach, the human teacher's goal is to teach the robot how to distinguish between the four classes of objects needed for the task by providing a small number of examples of each. The examples selected for each class $y \in Y$ are specifically intended to help the robot determine which features are most useful when identifying objects belonging to class y. As a note, we only *seed* the training set with examples selected by the teacher; the rest are automatically generated using SRS.

5.4.3 Combined Approach for Conveying Features

Lastly, there are two *combined* approaches that we explore: (a) **human instance selection + human feature selection (HIS-FS)** and (b) **human instance selection + human feature reduction (HIS-FR)**. For both approaches, the teacher's goal is to use the direct communication approach *first*, then subsequently select instances. The motivation for the combined approaches is to allow the teacher to subsequently reflect and explicitly communicate to the robot learner what they were attempting to implicitly highlight through the instances selected prior.

5.4.4 Evaluation and User Study

We sought to explore two research questions in this work:

1. *Learning* - Is a domain expert able to identify a subset of features that will enable the robot to classify unseen examples as accurately as using computational feature selection?

2. *Interaction* - Does the way in which feature information is elicited from the user impact the quality of resulting feature selection?

Toward that end, we have two hypotheses that we are testing: (1) humans intuitively understand and are able to characterize informative features of a task for which they have prior knowledge and (2) humans will do better at characterizing the task *indirectly* (selecting representative instances) than *directly* (enumerating useful features). We hypothesized that people would be more adept at *indirectly* communicating features because some candidate features generated by the robot's sensors may not be as intuitive for people, and with that, we would not necessarily expect the features used by people to map directly to features generated by robot sensors.

For evaluation, we conducted a between-subjects user study with 30 participants on Georgia Tech's campus, to collect data from humans about what features they would teach to help a robot differentiate between the task-relevant object classes. There were three conditions tested (10 participants per condition): (1) feature selection, (2) feature reduction, and (3) instance selection. For the study, all task-relevant objects were grouped by class on a table, but spaced out sufficiently for participants to see and interact with individual objects. All objects purchased corresponded to instances in the object dataset and therefore could be mapped to a corresponding set of images for processing.

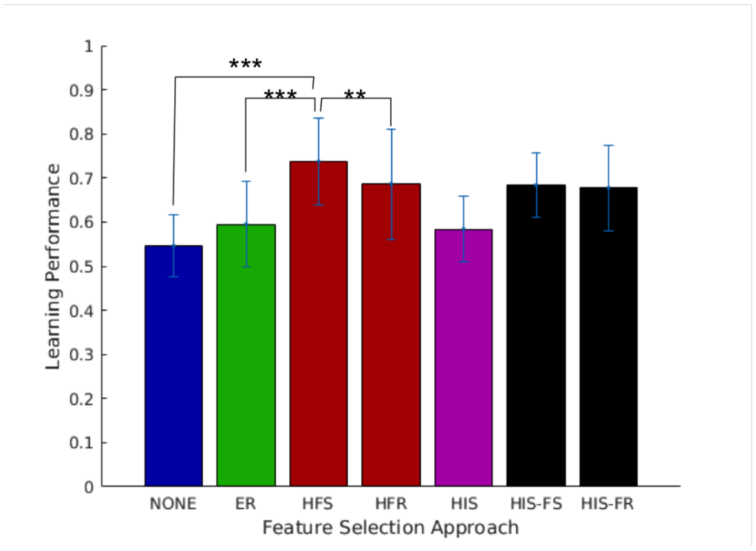
In the first two study conditions (HFS and HFR), participants are given the option to interact with the objects in any way they desire in order to help them decide which features to select or prune. They were also asked to do a brief exit survey upon completing the teaching task. For the third study condition (HIS), once three examples per class were selected by the teacher, all twelve examples are brought to the robot's workspace. Figure 5.1 shows an example of a complete demonstration for all object classes associated with the sort groceries task. Then, instead of completing an exit survey, participants from the third condition were equally subdivided into two sets. Directly following the selection of instances, one set was asked to *additionally* perform feature selection (HIS-FS); the other, feature reduction (HIS-FR). The order was not counterbalanced in this condition. We intentionally requested that each of these participants *first* communicate about features in an *indirect* way by selecting instances, *then* communicate features in a *direct* way by either (a) choosing relevant features or (b) eliminating irrelevant features.

This study provided the data needed for all five human-driven feature selection approaches discussed on the given household task. For the HIS approach, we only process the first teaching strategy used by participants in the third user study condition. As an additional note, the data for one participant from the instance selection + feature selection subgroup had to be excluded, thereby leaving data from 29 users mapped to the interaction strategies, as follows:

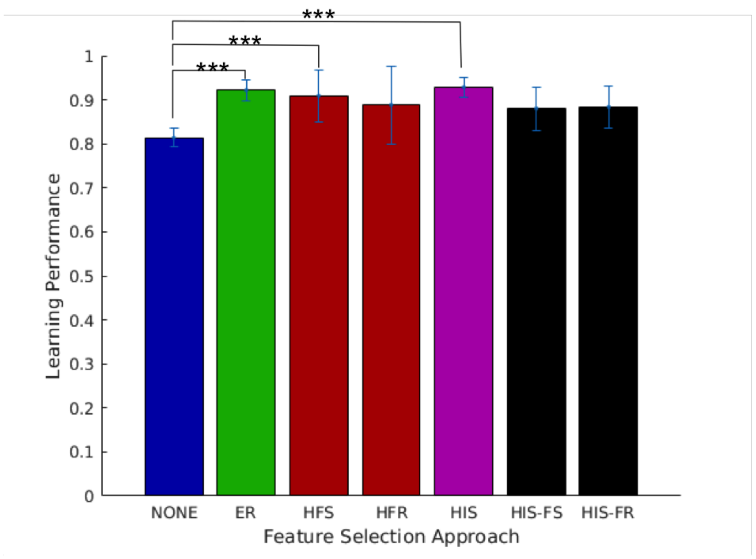
- HFS: 10 users
- HFR: 10 users
- HIS: 9 users
- HIS-FS: 4 users
- HIS-FR: 5 users

5.4.5 Learning Episode

Now that we have experimental data from users around feature subsets, we need to evaluate the extent to which the features they indicated are useful in learning the various classifiers needed for the task. We evaluate learned models as a function of the number of training instances n, in order to understand how human-driven feature selection compares to computational feature selection, given both small and large amounts of training data. Specifically, each learning episode consists of training $k|A|$ models, k different learned models for each $a \in A$, in each iteration j of the learning episode, as n increments from $n = 12$ to $n = 100$. The reason we generate k models $\forall a \in A$ is because we randomly generate k disjoint training samples during each iteration j. Thus for each j, $\forall a \in A$, we take the aggregate performance for all k learned models generated by approach a, in order to evaluate the *expected* performance and variance of a. We start the episode with $n = 3|Y| = 12$ examples since that is the number shown by the teacher, and we selected a termination point for the learning episode empirically based upon when learning performance converges. More details are included in the subsections below.



(a) Train Set Size = 12



(b) Train Set Size = 100

Figure 5.3: Learning performance of human-driven FS approaches for classification of objects relevant to *Unpack Groceries* task. The task involves four object classes where each training set has an even distribution of the classes. Test Set Size = 1000.

Generation of Training Samples

For *indirect* and *combined* approaches, we collected three human demonstrated examples for each classifier needed for the task. Thus at the beginning of a learning episode, the training set is a uniformly distributed sample $D_{i\ldots k,0}$ containing $3|Y|$ object instances, where D_i is the i^{th} training sample, $D_{i,0}$ represents the initial set for the i^{th} training sample, and

Table 5.2: Source of Training Data and Feature Sets (*per User*)
n = num instances in training sample
k = num training samples

FS Approach	*n*	*k*	*Instances*	*Features*
None	12	10	SRS	All
	100	10	SRS	
Computational FS {*ER*}	12	10	SRS	ER
	100	10	SRS	
Direct {*HFS, HFR*}	12	10	SRS	Human
	100	10	SRS	
Indirect {*HIS*}	12	1	Human	ER
	100	10	Human + SRS	
Combined {*HIS-FS, HIS-FR*}	12	1	Human	Human
	100	10	Human + SRS	

$|Y| = 4$ for the *sort groceries* task. For the *computational* and *direct* feature selection approaches, whereby *all* training instances are generated using SRS, this corresponds to $k = 10$ disjoint initial training sets $D_{i...k,0}$. For the *indirect* and *combined* approaches, where human teachers have selected the instances, there is only one initial training set, and it is composed *only* of the examples selected by the human teacher; thus $k = 1$.

After the initial set of training examples, all remaining instances are generated using SRS $\forall a \in A$. Thus for each a, in each subsequent iteration j, a new set of object instances is sampled from D^{train} such that

$$\forall y \in Y, \forall i \mid D_{i,j} \leftarrow D_{i,j} \cup \{o_y\}$$

where o_y is an object instance belonging to class y. Table 5.2 summarizes the generation of training data $\forall a \in A$.

Selection of Feature Subsets

In each iteration of a learning episode, after new instances have been added to the training sample, feature subsets must be selected by each $a \in A$, a classifier for each trained and tested, then learning performance recorded. For the *computational* and *indirect* approaches, feature subsets are also *dynamically* updated in each iteration. The ER algorithm is used to compute a new subset of useful features for both, based upon the *updated* training set. For the *indirect* and *combined* approaches, human teachers have already provided the subset of informative features; therefore the feature subset associated with each of these approaches remains *fixed* throughout the entire learning episode and is used for every training sample $D_i \ldots k$.

5.4.6 Results

Results in figures 6.4b and 5.3b reflect learning performance for each $a \in A$ for $n = 12$ and $n = 100$ instances respectively, as computed by Equation 5.4.

$$E[acc_{\mathbb{D}}(a)] = \frac{1}{|U_a|} \sum_{u \in U_a} \left[\frac{1}{k} \sum_{i=1}^{k} \frac{1}{n} \sum_{x \in D_i^u} [1 - \delta(h_i^a(x), y)] \right] \tag{5.4}$$

where D_i^u represents a training set that may have either been completely randomly generated or at least partially selected by the user $u \in U_a$, the set of users for approach a. Importantly, learning performance is now averaged across all $u \in U_a$. For the baseline of no FS and the computational approach, let $|U_a| = 1$ to denote one oracle that randomly generates examples for each training sample D_i. We used the Mann-Whitney U-test to compute pairwise statistical significance comparisons for each pair of FS approaches. However because there were so many comparisons, the bar graphs in Figure 5.3 only highlight the relationships that juxtapose the best performing human-driven approaches with baseline approaches, given small and large amounts of data.

Looking at figure 6.4b, with a small amount of training data, we observe that allowing a human teacher to provide feature information about the task yields a statistically significant increase in expected learning performance as compared to using only a computational feature selection approach. Specifically, the HFS interaction strategy appears to be the most effective way of eliciting information about useful features from the human teacher. It also dominates the second best human-driven interaction strategy, HFR. With a large amount of training data, computational feature selection (ER) and the best human-driven feature selection approaches (HIS and HFS) perform comparably; all three are statistically significantly better than a learner with *no* feature selection. Therefore while humans are not necessarily needed when there is sufficient training data available for the learner to observe, the fact that human approaches are still on par with the best computational approach suggests that the domain knowledge extracted from humans is both *useful* for the learning task and *reliable* as sample size grows.

All statistical significance relationships are shown in Tables 5.3 and 5.4. For each approach $a \in A$, $N = k|U_a|$ where k can be found in Table 5.2. $|U_a|$ for all human-driven approaches is listed at the end of Subsection 5.4.4. For each pair of approaches (cell) and given value of n, the corresponding table shows the probability p that error(approach a) $<$ error(approach b). Each row shows which FS approaches are dominated by a whereas each column shows which approaches dominate b. So *e.g.*, we observe that the baseline of no FS is dominated by every FS approach when there is a large amount of training data available.

5.4.7 Additional Task Domains

Our second hypothesis was that people would be *not* be as successful in *directly* communicating about task features; nonetheless our results failed to support this hypothesis. We believed this to be at least partially attributable to the features in the *sort groceries* task being quite intuitive for people. Thus we conducted a follow-on study to explore this further;

Table 5.3: Statistical Significance Relationships
where A = *Error(approach a)* and B = *Error(approach b)*
(p-values for $n = 12$)
$* = p < 0.05$; $** = p < 0.01$; $*** = p < 0.001$

$H: A < B$	*b:* None (N=10)	*b:* ER (N=10)	*b:* HFS (N=100)	*b:* HFR (N=100)	*b:* HIS (N=90)	*b:* HIS-FS (N=40)	*b:* HIS-FR (N=50)
a: None	–	0.82	1.0	1.0	0.84	0.99	0.98
a: ER	0.19	–	1.0	0.99	0.41	0.93	0.93
a: HFS	***	***	–	**	***	0.09	0.07
a: HFR	***	***	1.0	–	**	0.33	0.40
a: HIS	0.18	0.60	1.0	1.0	–	0.95	0.96
a: HIS-FS	*	0.09	0.91	0.67	0.07	–	0.06
a: HIS-FR	*	0.08	0.93	0.60	0.06	0.63	–

it consisted of an online survey whereby 48 participants were given three tasks: (1) playing Pacman for a reinforcement learning agent [80], (2) autonomous navigation through a crowded environment for a mobile robot [60], and (3) classification of fire, smoke, and thermal reflections for a humanoid firefighting robot [66]. For each task domain, the participant was shown an image of the domain, then asked to check off all features they believed to be most useful for a robot learning to perform the specified task.

For each domain, an empirically validated set of useful features is provided by the source referenced, thus used as our baseline for comparison. All participants were recruited from the same population of on-campus students. Table 5.5 lists the baseline features selected for each domain; for the firefighting task, all features listed are with respect to pixel intensities from thermal images of the scene. Figure 5.4 illustrates the amount of overlap between the human-selected feature subset and the baseline subset for each domain, where

Table 5.4: Statistical Significance Relationships
where A = *Error(approach a)* and B = *Error(approach b)*
(p-values for $n = 100$
$* = p < 0.05$; $** = p < 0.01$; $*** = p < 0.001$)

$H: A < B$	*b:* None (N=10)	*b:* ER (N=10)	*b:* HFS (N=100)	*b:* HFR (N=100)	*b:* HIS (N=90)	*b:* HIS-FS (N=40)	*b:* HIS-FR (N=50)
a: None	–	1.0	1.0	1.0	1.0	1.0	1.0
a: ER	***	–	0.59	0.28	0.87	*	**
a: HFS	***	0.41	–	*	0.94	***	***
a: HFR	***	0.72	0.96	–	1.0	**	**
a: HIS	***	0.13	0.06	***	–	***	***
a: HIS-FS	***	0.99	1.0	0.99	1.0	–	0.57
a: HIS-FR	***	0.99	1.0	1.0	1.0	0.43	–

a feature was included if selected by at least half of the participants.

The primary insight extracted from this follow-on study is that the only task people were not able to characterize in a way closely aligned with the computationally validated baseline is the one where most of the features selected were difficult to interpret semantically.

5.5 Discussion

Our overall findings are summarized in Table 5.6. It highlights the highest performing method(s) for selecting a useful feature subset as we vary two parameters: (1) amount of training data and (2) use of a human teacher.

The *bottom row* is specifically what were interested in exploring in this work. The findings suggests that even without having yet seen any training examples, a robot learner can leverage the knowledge of a domain expert to identify a subset of features useful for

Table 5.5: Task Domain Selected Features

Pacman	**Navigation**	**Firefighting**
grid width / height	location / orientation of robot	mean
grid cell locations	speed / direction robot is traveling	variance
location of walls	location / orientation of each pedestrian	standard deviation
number of ghosts	num pedestrians per square foot, in robot neighborhood	skewness
location of Pacman / ghosts / food / capsule(s)	speed / direction each pedestrian is traveling, relative to robot	dissimilarity
amount of food remaining	movement of each pedestrian towards / away from / perpendicular to robot	entropy
whether Pacman has been eaten		correlation

constructing a more discriminative task representation. This supports our first hypothesis that humans *are* able to help solve the feature selection problem and is valuable for LfD domains where the robot is learning from a teacher but training data provided is typically limited. Additionally, we have some insights about successful and unsuccessful ways to extract this feature information, when optimizing for classification accuracy.

In considering all five of the strategies we examined for eliciting feature information

HUMAN		**DATA**	
		Small	*Large*
	No	FI	ER
	Yes	HFS	HIS, ER, HFS

Table 5.6: Experimental Findings

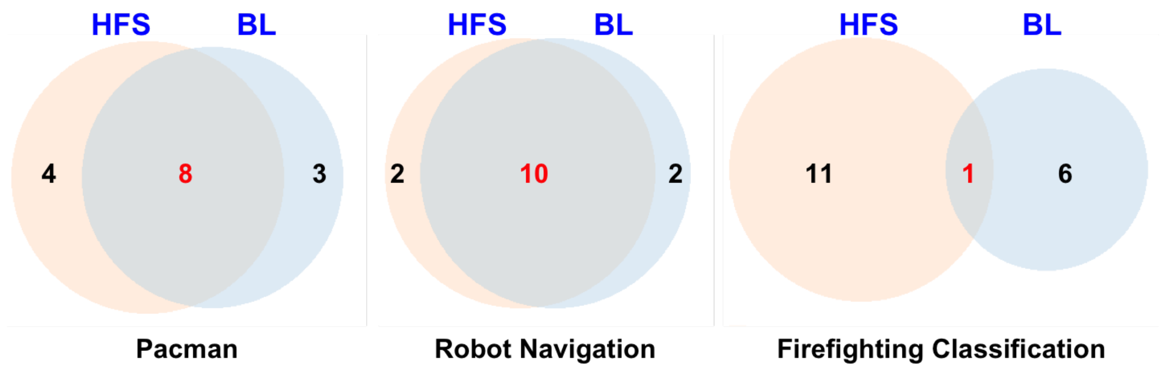


Figure 5.4: Venn Diagrams to show amount of overlap between selected feature subsets for each task domain

from humans teachers, two approaches emerged as most effective: human feature selection (HFS) and human instance selection (HIS). HFS outperformed all other approaches (both computational and human-driven) given a *small* amount of training data; both HIS and HFS were comparable to computational feature selection given a *large* amount of training data, but neither was able to significantly outperform the best computational feature selection approach (ER). So contrary to what we hypothesized, *direct communication* about features proved to be the most effective overall strategy for users. This was further validated in our follow-on study, providing the insight that users are able to select informative features *given* that the features can be understood intuitively.

Other insights gleaned from the results were that HFS always dominated HFR and that the combined approaches never yielded the best performance, as compared to other human-driven approaches. In future work, we can explore why we observe these trends.

5.6 Conclusion

Enabling robots to request the most useful features for characterizing a task is an important step toward autonomous task model construction. With only a small amount of data, computational feature selection approaches are limited in their ability to output the most useful features for discriminating between classes of objects needed for a task. Therefore, using computational feature selection as a baseline, this work explored: (1) whether a human teacher is able to characterize the most informative features of a classification task as accurately as computational approaches and (2) the best way to extract this feature information from the teacher. Our results suggest that a human teacher can *directly select* a subset of features that will be informative for discriminating between the task-relevant object classes given that the features are semantically interpretable. And in the case that the robot learner has either no or a small number of training examples, we can expect the subset selected by a human teacher to be more useful for classifying unseen task-relevant objects than that selected by a computational feature selection algorithm.

CHAPTER 6
ACTIVE CONCEPT GROUNDING THROUGH ARBITRATION OF DIVERSE LEARNING QUERIES

6.1 Introduction

The paradigm of Learning from Demonstration (LfD) enables an agent to learn a new task from examples provided by a human teacher [3]. In LfD however, the model learned depends on the ability of the teacher to provide appropriate examples to the learner. Placing the primary burden of conveying maximally informative input on the teacher presents an inherent challenge, as it is not feasible to expect every human with task domain knowledge to also understand how an agent models the task and be proficient at teaching it. Yet we want to leverage the domain knowledge of *any* user, independent of teaching skills. Therefore we seek to enable a learning agent to characterize its own uncertainty and autonomously solicit information it needs from the teacher to resolve that uncertainty, thus a collaborator in the learning.

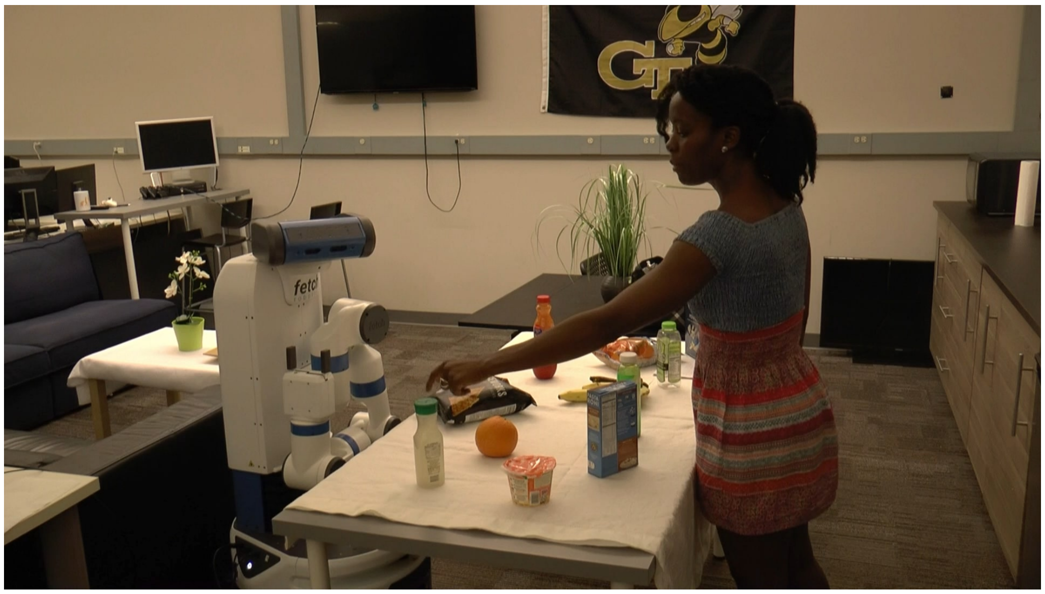


Figure 6.1: Example learning interation between robot and human partner for *lunch packing* task.

Student-driven agent learning has primarily been encompassed by the field of active learning (AL). Using AL techniques, an agent autonomously selects unlabeled training examples, based upon predetermined selection criteria, and queries an oracle for correct labels [4, 17]. In high-level task LfD, related literature has focused on learning an optimal policy for imitating a human demonstrator's behavior [6, 7], symbol grounding [9, 10], and inferring task constraints [11]. Importantly, the previous work has primarily focused on making *one* specific type of AL query towards generalization along that dimension of the task (*e.g.* taking the optimal action in a state). However, there is a wealth of information an agent can acquire from a human's domain knowledge. And proficient learners, like humans, combine information rather than simply focusing on one type of question. This work is the first to contribute algorithms for enabling an AL agent to *arbitrate* between diverse types of queries, with the goal of autonomously gathering *both* informative features *and* representative instances from the human teacher.

In this work, AL is used to solve a task-situated symbol grounding problem. Symbol grounding is the problem of mapping symbolic representations (labels, concepts) to constructs in the physical world [14]. Assuming no prior knowledge, the robotic learning agent is given a task (*e.g.* serving pasta) and with it, task-relevant concepts (*e.g.* cooking pot, pasta sauce) it must ground, in order to later perform the task in the situated environment. The agent learns to ground the concepts by actively querying its human partner.

The primary contributions of the work are (1) investigating whether enabling a learning agent with strategies for arbitrating between diverse types of AL queries improves learning performance, (2) exploring the design of rule-based (RB) and decision-theoretic (DT) arbitration strategies that enable the agent to acquire and appropriately prioritize feature and instance information useful for the given task, and (3) analyzing the tradeoffs between rule-based and decision-theoretic strategies with respect to learning performance in the agent's situated environment. We conducted an experiment comparing five query arbitration strategies, each gathering both feature and instance information by employing multiple query types, against two baseline approaches for making requests that each only obtain training instances by employing queries of one type. The evaluation was conducted on two tasks consisting of different computer vision datasets. Our findings showed that all RB and DT strategies outperformed both baselines on both tasks. We also found the DT strategy was able to consistently perform at least as well as all RB strategies but had an advantage in that it could additionally reason about *when* to make queries. Thus in the task where environmental change was both more gradual and substantial, similar to many real-world environments, the DT strategy statistically significantly outperformed all other strategies by its ability to adapt to the rate of environmental change and distribute its questions over time, thereby minimizing uninformative requests and acquiring a more representative training sample than any other strategy.

6.2 Related Work

As motivated in the introduction, we are interested in a *self-driven* learning agent who can leverage the expertise of its human partner in order to acquire the information it needs, by asking questions. Within machine learning literature, this is primarily addressed by AL. Our work is inspired by the scenario of a robot assistant able to acquire groundings necessary for later performing a task in the situated environment. For task learning, AL can enable a robot to both resolve unintended ambiguities during the learning process and explore unseen parts of the state space, in order to create a more *generalized* task representation. Related work on AL for robots has explored the learning of low-level action controllers [22, 23, 24], an optimal policy towards the end of imitating a human demonstrator's behavior [6, 7], grounding of goal state symbols [10], inferring task sequencing constraints [11], and retrieval of objects by the use of curiosity in human-robot dialog [29]. There has also been related work on strategies for introspective and extrospective detection and communication of the learner's knowledge gaps [32]. All of this previous work however has focused on asking *one* specific type of query towards generalization along that dimension of the task.

More closely aligned work includes the proposal of a framework with three types of embodied queries: *label*, *demonstration*, and *feature* queries. It characterizes the value of each in the context of learning lower-level motion trajectories [12]. Yet this work by Cakmak and Thomaz does not include arbitration between the query types, and the entire framework has not yet been applied to the domain of high-level task learning. Additional work within the robotics community looks at requesting feature information from a user. Rosenthal et. al. [72] recommend feature selection as a specific aspect which should be included when asking a question, in order to provide transparency to the human partner; however it does not include an algorithm for enabling the agent to autonomously reason about when to request feature information. Bullard et. al. [81] compare five different approaches for eliciting informative feature subsets from a human teacher and provide insights about the most effective ways to request features from the teacher. Though we can leverage insights from the findings of both, the contribution of this work is in *arbitrating* between several types of AL queries within one coherent questioning framework, such that the learning agent is able to reason about both employing multiple types of questions and acquiring diverse types of information from its human partner.

6.3 Problem Formulation and Overview

In our problem formulation, the AL agent must solve a *task-situated* symbol grounding problem, defined by [82], in which it must map abstract object symbols to perceptual input associated with physical entities in the agent's environment. Given a set of objects from a scene in the agent's purview, each object instance is represented by a feature vector $\mathbf{x} =< f_1 ... f_m >$. Instances are modeled by the superset of features F extracted from the robot's sensors (*e.g.* color, height). A set of binary classifiers, one for each symbol $y \in Y$, the set of object symbols, each take as input an instance $\mathbf{x}$ and produce a degree of confidence $p(y|\mathbf{x}) = [0,1]$ that $\mathbf{x}$ has label y. For each symbol to be learned, a binary Gaussian process classifier with a radial basis function kernel is trained. This representation was selected because of its efficacy in producing probabilistic predictions of unlabeled instances given only sparse training data.

6.3.1 Query Types

To acquire input data, the learning agent must query the human teacher. We utilize three types of candidate queries, which map to two different types of input data to be processed by the symbol grounding models: (1) instances and (2) features. Figure 7.1 illustrates what type of input data each AL query type provides.

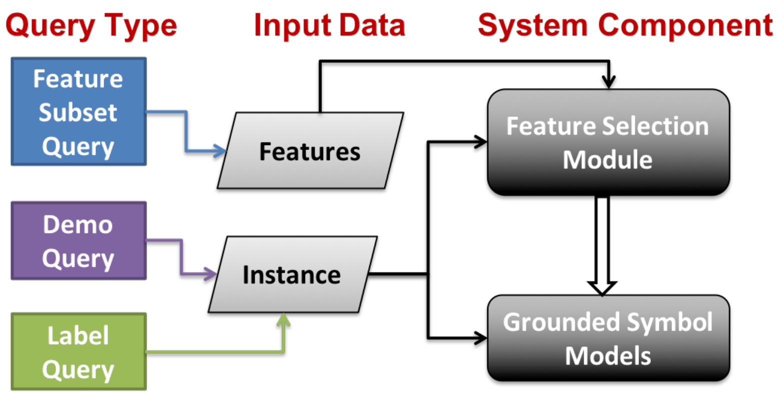


Figure 6.2: High-level system diagram mapping query types to type of input each provides and system modules processing the data.

Demonstration Queries

Demonstration (or demo) queries (DQ), analogous to active class selection [83], involve the learner requesting a new demonstration of how a concept (symbol) is embodied in the physical world. DQs provide new instances to the system with each demonstration selected by the teacher; the learner is responsible for communicating which symbol it requires a demonstration of.

Label Queries

Label queries (LQ), are synonymous to membership or instance queries extensively explored in AL literature [16, 17, 9, 10, 84]. The learner selects an unlabeled instance, based upon predefined selection criteria, and requests the correct label from the teacher. LQs provide new instances to the system as well, but the instances have been specifically targeted and selected by the learner.

Feature Subset Queries

Feature subset queries (FSQ) involve the learner requesting a subset of features useful for discriminating between the task-relevant classes; the teacher selects the features. We employ the human feature selection (HFS) approach introduced in prior work and found to be most effective in eliciting feature subsets from humans [81].

6.3.2 Learning Episode

The agent uses a single querying strategy $\hat{\sigma}$ throughout the entire learning episode. The episode begins with the teacher specifying the task and all relevant symbols to be grounded. At each turn t in the episode, the learner observes the state of the world, extracting perceptual input for all objects in the scene. The set of candidate queries consists of (a) one DQ associated with each symbol classifier, (b) one LQ for each object in the scene, and (c) one FSQ, which we constrain to a one-time query, since the answer is not expected to change over time. The learner uses $\hat{\sigma}$ to select a query to make at t, makes the query, then receives teacher feedback. If it receives a new instance, the instance is added to the training set of every symbol classifier (either as a positive or negative example). If provided with a subset of features, it updates all classifiers $y \in Y$ to only consider human selected features from that point in the episode. Lastly, the agent checks stopping criteria to determine whether to continue or end the episode.

6.4 Querying Strategies

Our goal was to assess the impact of three separate aspects on the agent's ability to learn the task-relevant concepts: (1) the ability to acquire diverse types of information, (2) the assignment of priorities to the query types, and (3) the ability to determine when to ask questions. Towards that end, we explore (1) two baseline strategies each making queries of one type, (2) random selection between queries of diverse types, and (3) two categories of *experimental* strategies for arbitration (rule-based and decision-theoretic).

6.4.1 Baseline Query Selection

Each *baseline* strategy, employs one query type and acquires only training instances from the teacher, as is typical in interactive learning. Neither has the ability to explicitly reason about when to make queries; they simply acquire data at every turn, until the learning episode concludes.

1. **BL: passive (P)** – Employing only DQs essentially reduces to the traditional LfD scenario, or passive learning, whereby the teacher continually selects examples and the learner passively observes.
2. **BL: active (A)** – Employing only LQs essentially reduces to traditional AL, whereby the agent continually selects unlabeled instances according to predefined criteria (*e.g.* uncertainty) and the teacher provides requested labels.

Passive learning is simulated using stratified random sampling from a generated task dataset. Uncertainty sampling [16, 17] is used for the active baseline, measured as entropy H of object instance $\mathbf{x}$ in the scene. It is computed by:

$$H(y|x) = -\sum_{y \in Y} P_\theta(y|\mathbf{x}) log P_\theta(y|\mathbf{x}) \tag{6.1}$$

6.4.2 Arbitration Strategies

The simplest arbitration strategy we employ is **random query selection (R)**. Given candidate queries of all types, R simply randomly selects one, at each turn. After a feature subset has been requested using HFS, it will select between only DQs and LQs. We validate this strategy to differentiate the relative benefits of multiple query types from the ordering effects of the *experimental* arbitration strategies.

Our experimental approach is inspired by Dialog Management literature, which has traditionally employed rule-based and data-driven approaches for action selection. In keeping with this, we explore two different classes of algorithms for the design of the experimental arbitration strategies.

Experimental: Rule-Based Arbitration

The prioritization of query types for RB strategies follows from machine learning literature. The number of training examples required to learn an accurate model of a concept increases exponentially with the number of features in the state space representation. Thus, machine learning systems typically employ feature extraction as a preprocessing step before training ensues, to increase sample efficiency. Additionally, in AL systems, some passive learning is often done first to obtain a small unbiased training sample for building initial models of the concepts. Then LQs are made on the remaining unlabeled instances, as selected by learner.

Based on these standard practices, we designed RB prioritization as follows: (1) request for task features using HFS, (2) initial demonstrations given by teacher, (3) label requests made by learner for refinement of initial symbol models. We investigate the following rule-based strategies:

1. **ARB: HFS + passive (HFS+P)** – Imposes constraint that features must first be selected by teacher, then passive learning ensues until termination of episode

2. **ARB: HFS + active (HFS+A)** – Imposes constraint that features must first be selected by teacher, then active learning ensues until termination of episode
3. **ARB: HFS + P + A (All/CD)** – Imposes constraint that features must first be selected by teacher, then a minimal set of demonstrations provided by teacher, then refinement of groundings is done through active learning. This strategy also tries to maintain a uniform class distribution.

Experimental: Decision-Theoretic Arbitration

RB strategies explored provide the agent with a seemingly intuitive set of heuristics for selecting a query type at each turn. However, they do not encapsulate any notion of agent goals. They do not allow the agent to directly compare queries of different types and reason about which most enables learning progress. And they do not allow the agent to reason about *whether* to even make a query. Thus we formulate a **decision-theoretic (DT)** arbitration framework which explicitly models the agent's *learning state*, allows direct comparison of diverse query actions, and encodes agent *learning goals* within a multiattribute objective function.

Let D be the set of training instances acquired by the agent, Y the set of task-relevant symbols, X the set of scene objects (unlabeled instances), and F_y be set of features used to represent symbol y. Learning state s is represented as the current estimate of the joint probability distribution between instances and symbols, where each element p_{xy} is the posterior probability that instance x is an example of symbol y. The current state s is dependent on both D and F_y for each $y \in Y$. The robotic agent's goal is to sufficiently ground and generalize its model of each $y \in Y$.

With respect to the assessment of learning progress, it is not feasible to assume the agent has access to a labeled test set that it can use to evaluate its current performance. Thus the agent needs a different way to both evaluate a candidate query action a and recognize when no query will help it to make it progress towards its learning goals. The expected utility of each candidate $a \in A$ is computed as a linear combination of two goals: (1) maximization of each symbol classifier's ability to discriminate aptly, and (2) minimization of selection bias in the training sample acquired.

1. *Average Classifier Discriminability (ACD)* – Ascertains ability of symbol classifiers to differentiate between most probable and least probable examples of their class. It employs the function:

   $$ACD(s) = \frac{1}{|Y|} \sum_{y \in Y} [p_s(y|x_{max}) - p_s(y|x_{min})]$$

   where p_s represents the probabilistic prediction for y in the current state s, and x_{max} and x_{min} represent the instances in the scene predicted to be the *most* and *least* probable examples of class y, respectively. ACD value should increase over the course of the learning episode, indicating that the symbol classifiers are improving in their ability to differentiate between examples in X.

2. *Class Distribution Uniformity (CDU)* – Assesses selection bias in the training sample, resulting from the agent collecting a sample unrepresentative of the underlying distribution. We assume a uniform distribution of symbol classes should be acquired by the agent so as not to bias it towards any task-relevant object, however the formula can be adapted for a nonuniform distribution as well. CDU employs the following function:

   $$CDU(s) = \frac{|D_{y_{min},s}|}{|D_{y_{max},s}|}$$

   where $D_{y_{max},s}$ and $D_{y_{min},s}$ are each subsets of the training sample. They represent the subsets of positive examples for y_{max} and y_{min}, the symbols most and least represented in the training sample at state s. This value is maximal when the class distribution is uniform.

Combining the above metrics, utility of s is computed as:

$$U(s) = w_1 ACD(s) + w_2 CDU(s) \tag{6.2}$$

To select an optimal query action a^*, the agent computes the *expected* utility (EU) of taking each $a \in A$, the set of candidate actions, and takes an argmax. EU is computed as

$$EU(a|D,X,Y,F) = \sum_{s'} U(s') * P(s'|a,D,X,Y,F) \tag{6.3}$$

where s' is a candidate next state resulting from taking a, dependent upon teacher feedback. Importantly, the agent's decision rule is to only take action a^* if $EU(a^*) > U(s)$. Else, the agent makes no query at turn t. Intuitively, this suggests a query should only be made if is expected to improve the state of the learning, *i.e.* engender learning progress.

To assess a DQ for symbol $\hat{y}$, the agent simulates the set of plausible responses by the teacher. Given the agent has requested a demonstration of $\hat{y}$, it assumes the teacher will draw the demonstration from X. Thus $\forall x \in X$, it simulates a resulting state s' by adding labeled instance $< x, \hat{y} >$ to D and computing the $U(s')$ according to Equation 7.2. We approximate the probability of s' occurring (*i.e.* the teacher providing x as a positive example of $\hat{y}$) as $p(x|\hat{y})$.

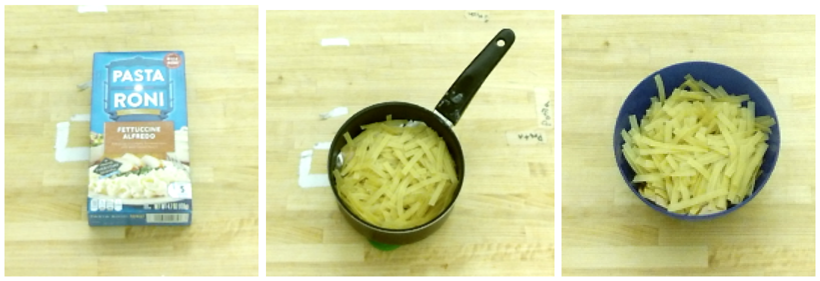
To assess a LQ for instance $\hat{x}$, the agent again simulates the set of plausible responses by the teacher. Given the agent has requested a label for $\hat{x}$, it assumes the teacher will provide it a label from Y. Thus $\forall y \in Y$, it simulates a resulting state s' by adding labeled instance $< \hat{x}, y >$ to D and computes the $U(s')$ according to Equation 7.2. We approximate the probability of state s' occurring (*i.e.* the teacher providing y as a label for $\hat{x}$) as $p(y|\hat{x})$.

To assess an FSQ, it is too computationally expensive to simulate all feature subsets the teacher could possibly provide, since there are $2^{|F|}$ candidate subsets. Thus, to substantially prune the search space, the agent can use feature subsets outputted by the computational feature selectors of each $y \in Y$. These subsets are the best approximates it currently has for informative features, and prior work has shown that given task features are intuitive, HFS is *at least* as good as computational feature selection [81]. Since we assume task features are semantically interpretable by a human, the agent can expect the feature subset it would receive to be *at least* as good as those outputted by computational methods. Thus, the agent simulates the set of plausible responses by the teacher with the computational feature subsets generated. Utility for an FSQ is computed as follows: $\forall y \in Y$, it simulates a resulting state s' by changing F_y to $F_{y,c}$ for each $y \in Y$, where $F_{y,c}$ is the computational feature subset computed for symbol y. Thus it retrains all symbol classifiers, given D, but with $F_{y,c}$ as the underlying representation. We approximate the probability of state s' occurring (*i.e.* the teacher providing feature subset $F_{y,c}$) as $1/|Y|$, a uniform distribution over the computational feature subsets generated by the symbol models.

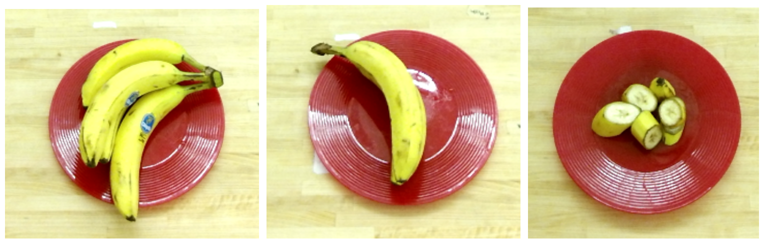
6.5 Evaluation

Given the research questions being explored, we evaluated three hypotheses: (1) arbitration strategies will outperform baseline strategies since they acquire both informative features and training instances from humans domain experts, (2) prioritizing the acquisition of feature data over instances will result in more efficient learning, and (3) DT arbitration will better adapt within dynamic environments since it additionally reasons about when to make queries.

To evaluate all strategies, we conducted an experiment with two different tasks: (1) a *pack-luncbox* task and (2) a *prepare-lunch* task. Each task uses the same four object symbols: main dish, snack, fruit, and beverage. However, the task datasets have different properties and were created from different image datasets. Each $\sigma \in \Sigma$ is evaluated using two metrics: learning accuracy (how well agent identifies unseen examples of each symbol) and sample efficiency (number of questions needed to sufficiently ground all symbols).



(a) Pasta (*state*:in box) (b) Pasta (*state*:in pot) (c) Pasta (*state*:in bowl)



(d) Banana (*state*:bunch) (e) Banana (*state*:single) (f) Banana (*state*:sliced)

Figure 6.3: Illustration of object state changes for *main dish* and *fruit* objects classes in *prepare-lunch* task.

6.5.1 Data Collection

The pack-lunchbox task assumes all groundings remain static, which means the way the object is embodied in the world does not change. The main dishes (instant noodles) are always packaged, the beverages (water and soda cans) remain bottled or canned, the fruit (apples, oranges, peaches, pears) is whole and ripe, and the snacks (food bags, *e.g.* chips) remain closed. Data for this task was collected from the University of Washington RGB-Dataset of common household objects [77]. The image dataset includes over 200,000 object images in total, encompassing over 300 objects organized into 51 categories (e.g. soda can), with multiple object instances per category (*e.g.* pepsi can, mountain dew can). For each object instance, there are several hundred images, captured from three camera viewpoints; a small subset of object instances are additionally captured under different lighting conditions. For the pack-lunchbox task, we only consider object instances relevant to the symbols being grounded. Given images of object instances from the UW dataset, we generated five disjoint training datasets for the pack-lunchbox task and one hold-out test dataset, each training dataset consisting of $n = 3200$ images and the test dataset consisting of $n = 800$ images. All datasets contain images of the same set of task-relevant object instances. For each dataset, stratified random sampling without replacement was used to generate a uniform class distribution.

In the prepare-lunch task, some objects change state, presumably as lunch is being prepared. The motivation for this is even within the same environment, groundings for a particular symbol can change over time (*e.g.* an apple transitioning from whole to sliced, or pasta going from being packaged in a box to being served in a bowl). This property of dynamically changing groundings is an important part of our problem domain and is thus explored in the second task. Towards that end, we varied the states of two objects symbols (main dish and fruit) and allowed the other two to remain static (beverages and snacks). Figure 7.2 illustrates changes that could reasonably occur as the task is being performed.

Data collection was done using a Kinect RGB-D sensor on our mobile manipulator robot platform. The sensor is mounted on the head of the robot and was angled to look down at the robot's workspace for taking color and depth images of each object. As data was being collected, the orientation of each object instance was systematically varied, and each object instance was also moved to various positions around the workspace to create different lighting angles. A total of approximately 200 images were taken, four object categories which decompose into 14 different object instances: 3 boxes of pasta, 4 brands of chips, 3 types of fruit, and 4 beverages. From the image dataset created, train and test datasets for the task were generated so as to ensure a representative sample of the object states collected in the data. Given that we aimed for a uniform distribution of classes and the same underlying distribution of object instances in each dataset, not all images were used. The training dataset for the prepare-lunch task contained $n = 80$ images and the test dataset contained $n = 40$ images. For this task, since the number of images of transformed objects taken from our robot was several orders of magnitude smaller than the UW dataset, we seeded each training dataset with the same set of images but added Gaussian noise to the extracted features; we also added Gaussian noise to simulated features for both task datasets. We again generated five training datasets for the second task. For both tasks, the training and test datasets generated are disjoint.

6.5.2 Sensory Input

Since our work is intended for a robotic agent, we collected real-world vision data from a robot's camera[1], as well as simulated multi-modal feature information to represent features that would be extracted from other robot sensors[2], resulting in 90 low level features in total. Thus for each object in the scene, F is computed based upon perceptual information taken from multiple robot sensors.

6.5.3 Experimental Design

In designing experiments, the agent should be provided with perceptual input that simulates a robotics domain. Thus, we had two goals: (1) since a robot's perceptual system typically outputs one feature vector per cluster in the scene, the system is designed to randomly sample one image per scene object from the specified task dataset each time a new set of observations is generated and (2) since robots typically operate in dynamically changing environments, the system samples a new set of observations every r turns in the learning episode, where r represents the rate of environmental change. For the task where groundings remain static, environmental changes include only viewpoint and/or lighting. Groundings changing over time (Figure 7.2) means environmental changes may also include physical object state change. At each turn t, X contains only one observation (image)

[1] *object bounding box position, orientation on table, color, size dimensions, area, volume, aspect ratio, visual texture, compactness of object's point cloud, and density of point cloud contour*

[2] *object's location relative to interest points in the environment (e.g. counter top, stove, refrigerator, pantry), the object's location relative to the robot base, absolute location of robot's base in the environment, location of the robot's base with respect to the counter top, the robot's joint positions for each arm, pose of the robot's hands, robot's hand states (open vs closed), weight of the object, and max/min/average volume of noise in the environment over duration of learning episode.*

of each object in the scene. To simulate environmental change, the perceptual system generates a new set of observations. Else, it outputs the set of observations from $t-1$. Given X, the agent decides whether to query, then updates and evaluates all symbol models following feedback given. The teacher for all experiments was one of the authors.

6.6 Results

We compare learning accuracy resulting from employing each of the different questioning strategies. To test each strategy $\sigma \in \Sigma$, learning accuracy is computed using:

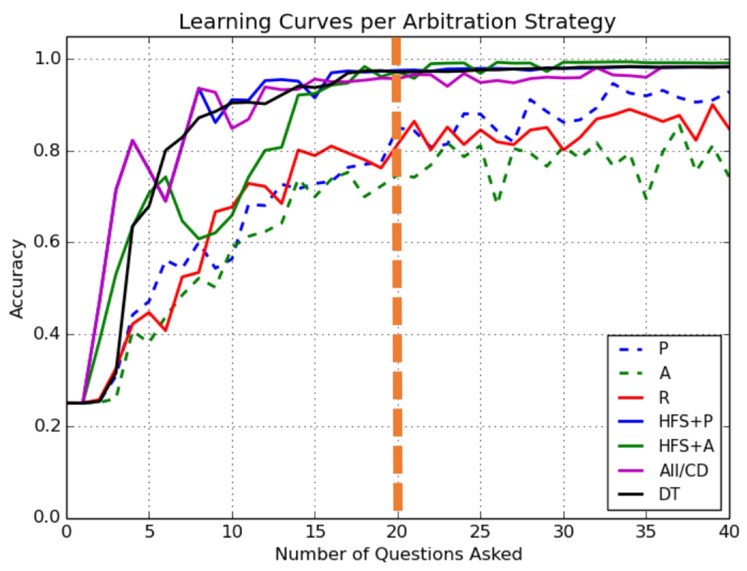
$$E[A_{\mathbb{D}}(\sigma)] \approx \frac{1}{k}\sum_{i=1}^{k}\frac{1}{n}\sum_{\mathbf{x}\in D_i}[1-\delta(h_i^{\sigma}(\mathbf{x}),y)] \tag{6.4}$$

where E is the expected value of the learning accuracy using σ on training dataset D_i with respect to distribution $\mathbb{D}$, $h_i^{\sigma}(x)$ is the hypothesis of the learner using σ trained on D_i then tested on instance $\mathbf{x}$ in the test dataset, y is the ground truth label for $\mathbf{x}$, and the quantity $\delta(h_i^{\sigma}(\mathbf{x}),y)$ is 1 if $h_i^{\sigma}(\mathbf{x}) \neq y$ and 0 otherwise. Also, n is the number of test instances in each task test dataset and k is the number of task training datasets used. For both tasks, $k=5$; $n=800$ for pack-lunchbox task and $n=40$ for prepare-lunch task.

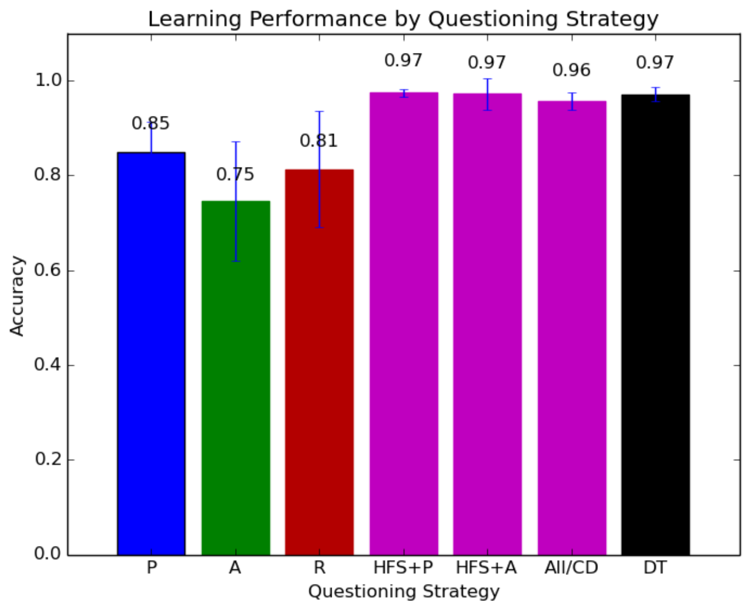
6.6.1 Learning Static Groundings

Figures 6.4 and 7.4 show learning accuracy for each $\sigma \in \Sigma$ in the pack-lunchbox and prepare-lunch tasks respectively. Each σ was given a 40 question budget in the former and 60 question budget in the latter since it is a harder learning problem. We use the Mann-Whitney U-test to compute statistical significance comparisons for each pair of strategies.

Our first hypothesis was that strategies gathering both feature and instance information will outperform baselines acquiring only instances. Our resulting learning curves support this hypothesis for all experimental strategies (*i.e.* rule-based and decision-theoretic), with respect to both learning accuracy and number of questions necessary for learning perfor-



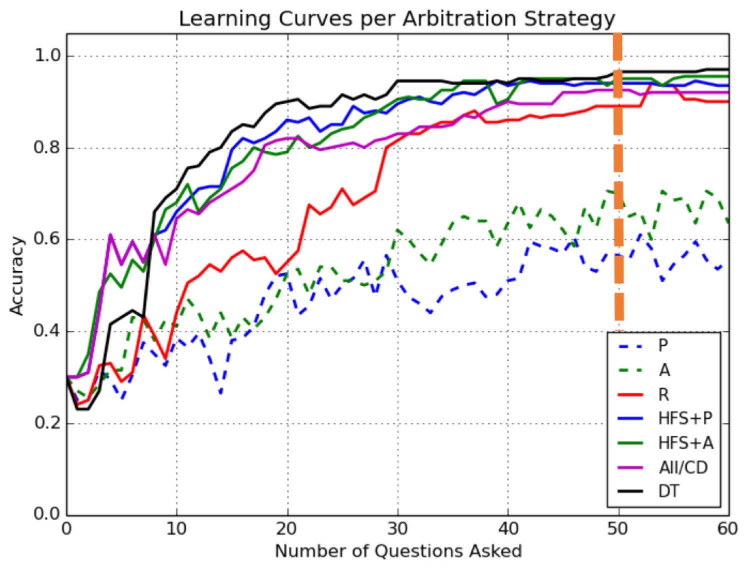
(a) Pack Lunchbox Task



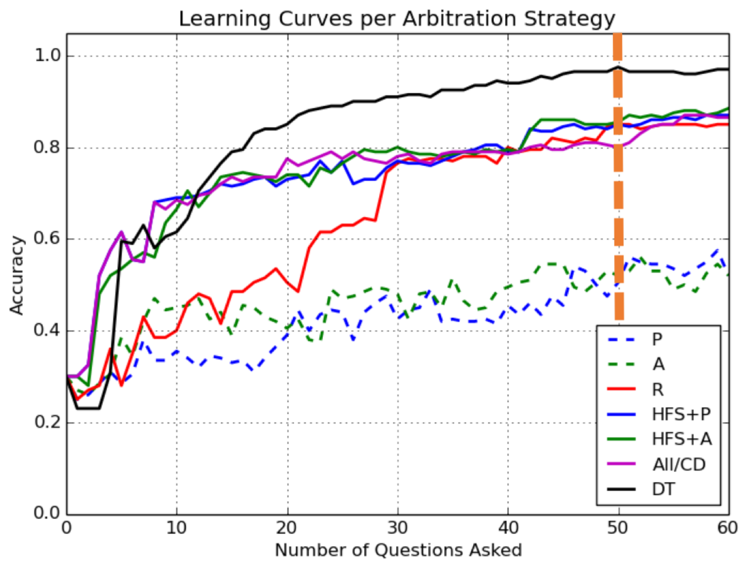
(b) Performance after 20 questions

Figure 6.4: (a) Accuracy of all strategies for pack-lunchbox task, as a function of number of questions asked. Baseline approaches use computational feature selection; experimental strategies request human-selected features. (b) Comparison of accuracy once learning has stabilized for best strategies (after 20 questions).

mance to stabilize. For the pack-lunchbox task (Figure 6.4), learning performance for all experimental strategies begins to stabilize after approximately 20 questions have been asked, whereas performance does not stabilize for the baseline strategies until approxi-



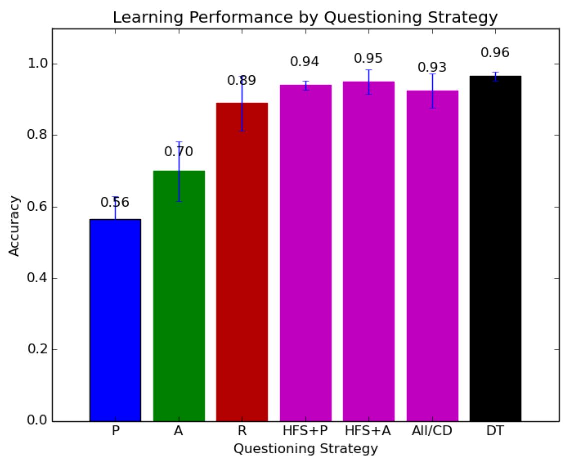
(a) Prepare Lunch Task
(*rapidly* changing environment)



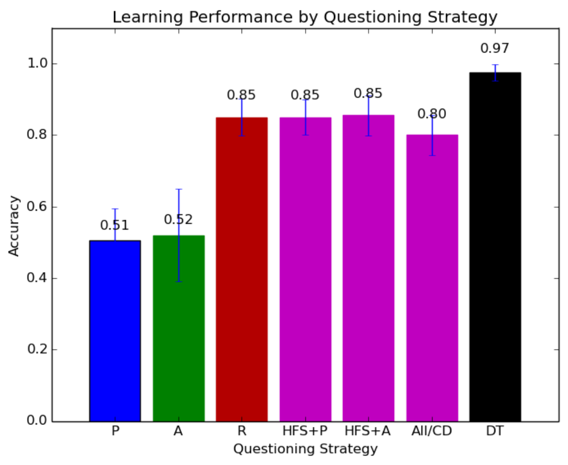
(b) Prepare Lunch Task
(*gradually* changing environment)

Figure 6.5: Accuracy of all strategies for prepare-lunch task, as a function of number of questions asked. Baseline approaches use computational feature selection; experimental strategies request human-selected features. Performance under both (a) *rapid* change (every turn) and (b) *gradual* change (every 20 turns).

mately 40 questions have been asked. And the baselines still require additional questions to reach the performance of the experimental strategies. Thus on the easier learning task, the experimental strategies are able to sufficiently learn the task-relevant concepts with less



(a) Performance after 50 questions
(*rapidly* changing environment)



(b) Performance after 50 questions
(*gradually* changing environment)

Figure 6.6: Accuracy for prepare-lunch task when learning stabilized for best questioning strategy (after 50 questions), as denoted by the vertical red bars on learning curves. Performance under (a) *rapid* change (every turn) and (b) *gradual* change (every 20 turns)

than half the number of questions. For the prepare-lunch task (Figure 7.4), all arbitration strategies significantly outperform both baseline strategies throughout the entire duration of the learning episodes tested, the random strategy outperforms the baselines for a little more than half of the episode, and the rate of increase for the baseline strategies is very gradual. Thus we do not expect learning performance to stabilize for any of the baseline learners on the more difficult learning task until well after any of the learners using an arbitration strategy conclude their episodes.

The second hypothesis being tested was prioritization of a feature subset request over instance acquisition, imposed by experimental strategies, would result in more efficient learning, as compared to a random arbitrator, which also combines all query types but with no apparent strategy. We found that on average, the random strategy seems to perform on par with the two baseline strategies incorporating only one query type, in the pack-luncbox task, which means it takes much longer to learn the concepts than experimental arbitration strategies for this task. This supports our hypothesis. However, random is able to perform comparably with the experimental arbitration strategies after approximately 30 questions on average, in the prepare-lunch task. Thus all arbitration strategies sufficiently learn the task after approximately 50 questions. This fails to support our hypothesis. To understand why, we examined the episodes more closely. We found that in episodes where the random learner requests human features, learning performance spikes and quickly becomes comparable to that of the experimental strategies thereafter. In episodes where an FSQ is not made, this essentially reduces to the case of randomly selecting between only DQs and LQs; in those cases, we observed learning performance comparable to baseline strategies for the duration of the episode. And since the pack-lunchbox task has over four times the number of objects as the prepare-lunch task (55 vs 12), and thus considers approximately four times the number of candidate queries per turn, R takes substantially longer to randomly select an FSQ in pack-lunchbox than in prepare-lunch. This explains the significant shift in the performance of R in the prepare-lunch task (Fig 7.4) but not in the pack-luncbox task (Fig 6.4a); it takes much longer to happen in the latter case. The overall implication is the acquisition of informative features has a significant impact on learning performance; thus if the teacher can provide them, a feature subset request should be prioritized.

6.6.2 Learning Groundings that Change over Time

Our final hypothesis was DT arbitration would better adapt within dynamic environments because it additionally reasons about *when* to make queries. We aimed to understand the impact of the *rate* of environmental change on efficacy of arbitration strategy employed. For this analysis, we focus on the prepare-lunch task since we found that rate of environmental change did not noticeably impact learning performance for the pack-lunchbox task, where groundings remain static. In the prepare-lunch task however, the environment *must* change for the agent to encounter all possible symbol groundings, since the groundings themselves change over time. Thus, we use prepare-lunch for exploration of dynamic groundings.

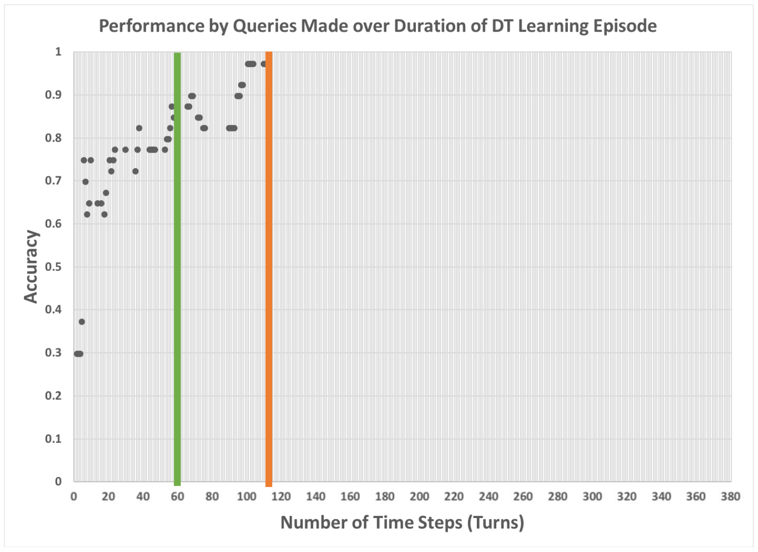
Figures 6.5a and 6.5b compare learning performance per number of questions asked in prepare-lunch, given both *rapid* and *gradual* environmental change. When rapid change is occurring (every turn), performance $\forall\sigma \in \Sigma$ stabilizes after approximately 50 questions. As shown in Figure 6.6a, all RB strategies (magenta) and the DT strategy (black) perform comparably. However, under gradually changing conditions (every 20 turns) [3], DT clearly and significantly outperforms all other strategies for most of the learning episode. Figure 6.6b highlights this by comparing performance of all $\sigma \in \Sigma$ at 50 questions, where DT begins stabilizing. Here, all arbitration strategies statistically significantly outperform both baselines. Moreover, DT statistically significantly outperforms *all* other strategies. In all cases, $p < .05$.

To better understand why DT dominates in this setting, we examine Figure 6.7, which visually depicts one learning episode for the DT strategy under both rapid and gradual environmental change. Accuracy is plotted as a function of time steps elapsed. Gray vertical bars represent change occurring in the environment. The dots indicate time steps where a query is made. The green vertical bar indicates when all *other* strategies complete their episode (after 60 time steps) since all other $\sigma \in \Sigma$ make a query at *every* time step until their questioning budget is depleted. The red vertical bar indicates when the DT agent depletes its questioning budget and completes its episode.

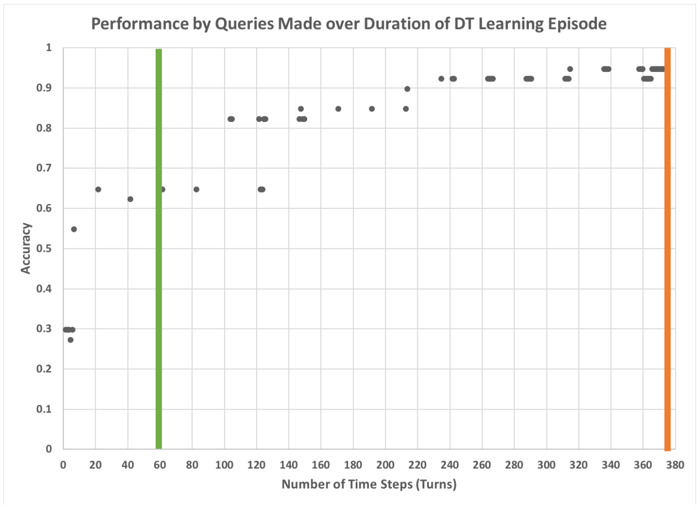
To further understand what is changing, we look at examples of world state for the *Prepare Lunch* task-relevant groundings, in figures 6.8a, 6.8b, and 6.8c, respectively. Though the instances taken as input to the system were actually sampled from the our task dataset, the following images are shown simply to visually illustrate the individual state of each of the objects in the agent's scene, in the first three environmental states. That is, the state of all of the objects corresponding to each of the first three gray vertical lines in the graphs. Importantly, as described earlier, the fruit and the main dish classes are changing in substantial ways, representing an external change of state, due to the dynamic nature of the environment in which the robotic agent exists.

Comparing the graphs, under gradual change, the DT agent makes less frequent re-

[3]Accompanying video at https://www.kaleshabullard.com/research/



(a) Rapid Environmental Change
(perturbation every turn)



(b) Gradual Environmental Change
(perturbation every 20 turns)

Figure 6.7: Learning Performance as Number of Time Steps in Learning Episode for *Decision-Theoretic* Strategy

quests and distributes its questions over 374 time steps. Whereas under rapid change, it takes only 112 time steps to complete its episode and still achieves comparable performance. Even more compelling, under both conditions, its queries are generally made soon after environmental change occurs. Thus illustrating its ability to be adaptive and responsive to environmental change and successfully acquire a representative training sample, independent of the *rate* of change. By comparison, since all other strategies make a query at every time step, they end up acquiring many redundant training examples when the environment is changing slowly. This leads to training samples that inadequately represent the diversity (variance) in each task-relevant class and classifiers that perform sub-optimally on a representative test set. In short, the DT agent allots its questioning budget more wisely, so it is largely unaffected by the slowly changing conditions. In a long-term setting, this is especially compelling because the agent can effectively reason about how to refine its models as a function of change in the environment and does not have to rely on the user to track the state of its knowledge over extended durations or decipher when and how to help the agent update its models.

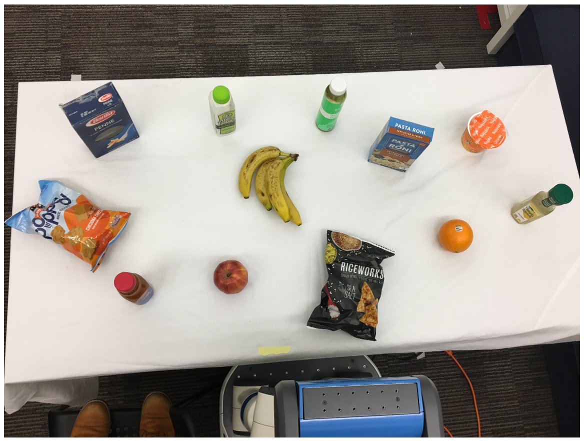
6.7 Discussion

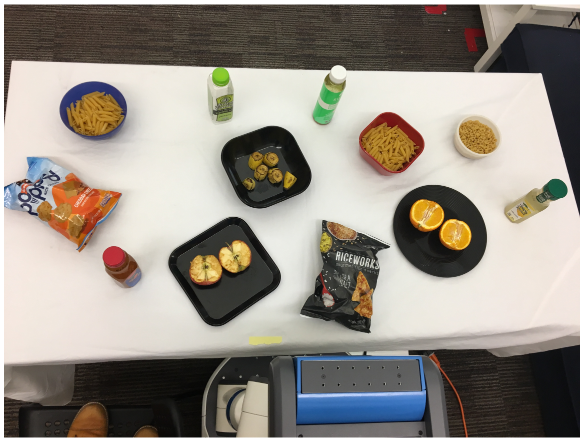
From our experimental investigations, two key insights emerge: (1) enabling the learning agent to ask questions that elicit diverse types of input (*i.e.* both informative features and instances) and appropriately prioritize the query types consistently leads to more efficient learning of the task-relevant concepts and (2) given a dynamic environment and constrained questioning budget (typical in human settings), the DT strategy is able to make the best use of the limited number of questions by deciphering both *when* to make a query and *what* query to make.

The DT strategy is not without its limitations however. DQs are costly to a human teacher because they often require more effort and time. The DT strategy used an average of 32% and 38% of its questioning budget making demo requests for the pack-lunchbox and prepare-lunch tasks respectively, in the rapidly changing environment, and an average of 52% for the prepare-lunch task in the gradually changing environment. This is substantially more demo requests than the rule-based strategies, which are fixed at 8 DQs, independent of the task, or on average 20% and 13% of the budget for the pack-lunchbox and prepare-lunch tasks respectively. In future work, we are interested in exploring interaction-related attributes in the DT strategy's objective function, so it reasons about an optimal action, considering both learning progress and social aspects of the interaction.

6.8 Conclusion

This work explored the use of rule-based and decision-theoretic strategies for arbitrating between AL queries of different types, to enable a learning agent to acquire diverse types of information (*i.e.* informative features *and* training instances) from a human teacher. We conducted experiments on two different tasks under different environmental conditions, comparing 4 experimental arbitration strategies against baselines of more traditional passive and active learning, as well as random query selection. Overall, the questioning strategies that enabled the learning agent to (1) extract diverse types of information and (2) prioritize acquiring feature information early in the learning episode, more efficiently learned to ground the task-relevant concepts. Moreover, given a dynamically changing environment and constrained questioning budget, the DT strategy was the only strategy able to acquire a representative training sample, independent of the rate of environmental change, because it reasons about both *what* query to make and *when* to query. These findings show that strategic arbitration eliciting diverse types of information from the teacher is able to consistently maximize learning performance when grounding concepts.


(a) Environmental State 1


(b) Environmental State 2

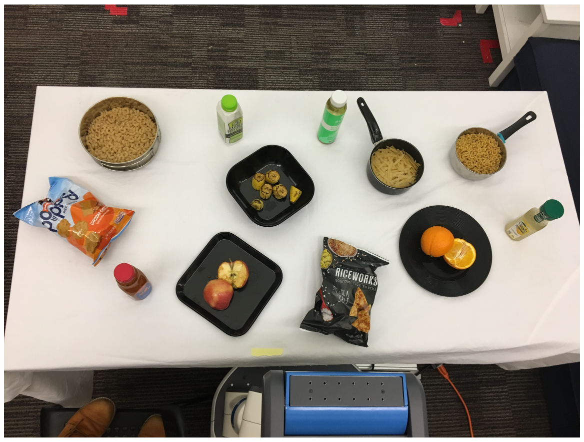

(c) Environmental State 3

Figure 6.8: Prepare Lunch Task: depicts state of objects in three environmental scenes observed by agent.

CHAPTER 7
ACTIVE CONCEPT GROUNDING WITHIN CONSTRAINED ENVIRONMENTS THROUGH IMITATION OF AN EXPERT QUESTIONER

7.1 Introduction

Active learning (AL) agents are intended to learn from an oracle, often assumed to be human, but typically not *designed* for more realistic human environments. Understanding environmental context however is especially important for robotic agents, generally assumed to be colocated in the environment with the oracle or teacher. Within the robotics community, there has been AL work aimed at understanding [85, 86, 87, 88], modeling [89, 31], and improving [9, 30] interaction with a human partner. An important aspect of the interactive learning problem, this body of work focuses on *interaction* with the teacher, but there still remains the open question of how the learner should integrate reasoning about the *environment* in which it is situated.

Specifically, external constraints imposed on the learner may have *direct* implications for solving the learning problem. For example, a teacher has only a limited time frame of availability or limited cognitive resources that can be devoted to answering the learner's questions. This information may need to influence the learner's questioning policy. However, the problem of trading off learning goals with environmental constraints is relatively unexplored within AL literature, particularly when considering dynamic environments. Yet this problem is important for learning in realistic human settings.

In this work, we investigate the question of how to enable an active learner to reason about its learning objectives within a dynamically changing environment while concurrently considering time and resource constraints provided for solving the learning problem. We use a decision-theoretic approach to active learning, whereby the individual decision

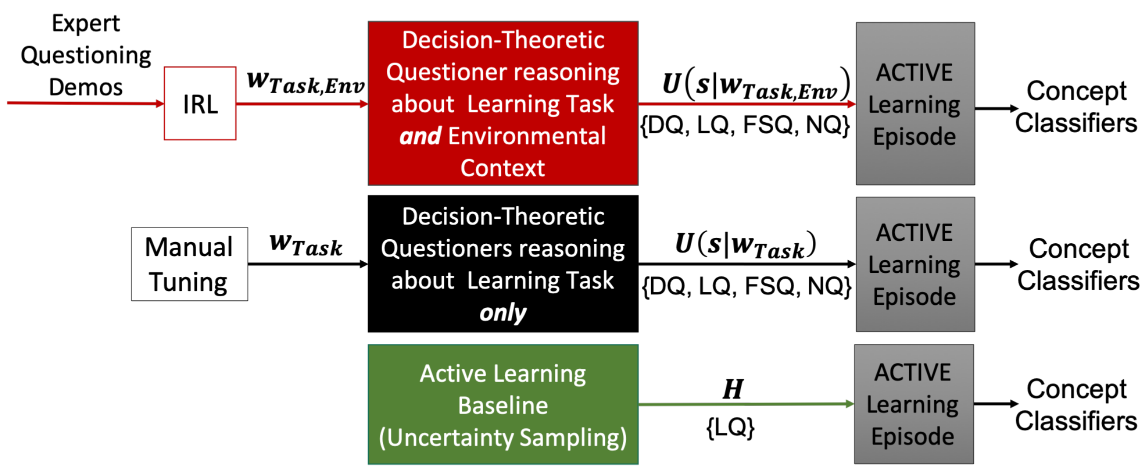


Figure 7.1: Learning system diagram, illustrating how each active learning strategy performs query selection.

criteria (or decision features) within the objective function are hand designed and include both task-centric and environment-centric features. Nonetheless, since the learning agent must consider multiple and diverse decision criteria, it becomes difficult to manually tune the individual objectives. Thus we propose imitation of an expert questioner for learning to weight the decision features. Our approach employs Inverse Reinforcement Learning (IRL) for inferring weights of the objective function from demonstrations of an expert policy.

In the experiments conducted, the agent is given a concept learning problem that it must use active learning to solve, under different environmentally constrained conditions. This work makes the following contributions:

- first AL work to reason about environmental constraints within the objective function of the learner
- first AL work to use imitation learning for mimicking the policy of an expert questioner

We evaluate efficacy using two separate task datasets and show that environmentally-aware reasoning allows our algorithm to significantly outperform an established AL baseline of uncertainty sampling and task-centric questioning strategies examined.

7.2 Related Work

Active Learning encompasses an extensive body of literature, spanning across several problem domains. We focus here on the most relevant work within the broader space, active learning for robots and embodied artificial agents. Most literature in AL for robots solves learning problems directly relevant to robotics domains: learning an expert policy to derive desirable robot behavior [6, 7, 22, 12, 23, 25, 28, 26, 27], inferring sequencing constraints on actions in a task [11, 31], and grounding task-relevant symbols or descriptions [9, 10, 29, 90, 30]. A key limitation however is current approaches *only* reason about the learning problem the agent must solve, but not time and resource constraints imposed by the environment in which the agent is situated. In other words, in prior literature, the learner typically uses an objective function to reason about its learning goals but does not *additionally* consider environmental constraints.

The most relevant prior work explored autonomous arbitration between multiple types of active learning queries, acquiring both feature and instance input from the teacher, and situated in dynamic environments [90]. A primary contribution of this work was an algorithm for arbitration between diverse query types that could also adapt the frequency of its questioning to the rate at which objects changed in its environment. Nonetheless, a key limitation is the inability to reason about constraints imposed by the teacher or learning environment. Thus, while able to adapt its questioning strategy to the rate of environmental change, the learner has no mechanism for adapting its strategy to the query budget given or amount of learning time allocated.

7.3 Problem Formulation and Approach

Symbol (or concept) grounding is the problem of mapping symbolic representations to constructs in the physical world [14]. Specifically, the agent must solve a *task-situated* concept grounding problem, whereby it is given abstract task-relevant concepts to be perceptually grounded in its environment, in a way appropriate for the task. For example, when learning concepts for the *serve breakfast* task, *eggs* scrambled or sunny-side up would be a more appropriate grounding than a dozen cartoned eggs.

We formalize the problem as follows: Given a set of objects $\mathbf{X}$ from a scene in the agent's purview taken at time t, each object instance $\mathbf{x} \in \mathbf{X}$ is represented by a feature vector $x^t =< f_1^t ... f_m^t >$. We assume the agent has both exteroceptive and proprioceptive sensors for perceiving its external environment and internal state. Each object instance then is modeled by the superset of features F extracted from the agent's sensors at t (*e.g.* object height or color, position of robot base or end effector). A set of binary classifiers, one for each symbol $y \in Y$, the set of object symbols, each take as input an instance $\mathbf{x}$ and produce a degree of confidence $p(y|\mathbf{x}) = [0, 1]$ that $\mathbf{x}$ has label y. For each symbol, a Gaussian Process Classifier was trained. This representation was selected because it both probabilistically models agent uncertainty and learns well from sparse data.

In dynamic environments, groundings also change over time and concept models must be refined accordingly. This may include change in the physical object state (*e.g.* eggs going from being in a shell to scrambled) or objects being replaced within the same category (*e.g.* breakfast beverage being served one day as coffee in a mug and another as orange juice in a glass). Since it is unreasonable to expect a human partner to track the agent's knowledge over time, in a changing environment, we take an *active* concept grounding approach.

7.3.1 Active Learning for Concept Grounding

Active Learning enables a learner to query an oracle or teacher for information about which is has uncertainty. It typically assumes a query will be made at every turn and seeks to equip the learner with a utility function for selecting an *optimal* query [16]. However, real world environments often do not allow the learner unlimited queries or time for querying, and simultaneously change over time. This means it is not always the best use of time and resources to make a query at every time step, until the query budget is depleted. Thus, employing a traditional AL strategy may not maximize learning in dynamically changing, constrained environments, as shown by [90].

We present a decision-theoretic AL approach which extends prior work intended for dynamic environments [90]. In that work, the authors contributed a decision-theoretic framework for arbitrating between multiple types of AL queries, acquiring both informative features and representative training instances from a teacher. Building upon that framework, our approach contributes a model that is able to reason about *both* the agent's concept learning goals *and* external time and resource constraints imposed on the agent. Specifically, the objective function of the learner is *expanded* to include decision criteria which reason about environmental context. Equation 7.1 shows the learner's objective function used at each turn t to assess the expected utility (EU) of an action a, given the current learning state s_t.

The learning state at t includes: $\{$estimate of posterior probability distributions of $y \in Y$ for all $\mathbf{x} \in \mathbf{X}$, interaction history, query budget, and teaching time allocation$\}$. The set of candidate actions A_t consist of demonstration queries for each of the task-relevant concepts $[DQ(y)\ \forall y \in Y]$, label queries for each object in the current scene $[LQ(\mathbf{x})\ \forall \mathbf{x} \in \mathbf{X}]$, a feature subset query $[FSQ]$ to identify relevant features for discriminating between task concepts, and a no query action $[NQ]$. Thus, there are $|A_t| = |Y| + |\mathbf{X}| + 2$ candidate actions from which the agent can choose at each turn t. Additionally, each of the query types is associated with a cost, given a priori. The learner selects an optimal action a^* as

$$\begin{aligned} a^* &= \arg\max_a \ EU(a|s_t) \\ &= \arg\max_a \sum_{s_{t+1}} P(s_{t+1}|a, s_t)\ U(s_{t+1}) \end{aligned} \tag{7.1}$$

where

$$U(s) = w_1\phi_1(s) + w_2\phi_2(s) + ... + w_n\phi_n(s) \tag{7.2}$$

The set of decision features $\phi \in \Phi$ used in computing $U(s)$ comprise the representation for the agent's objective (decision) function and is primarily what distinguishes prior work from the approaches introduced in this work. $U(s)$ is represented as a function of decision features $\phi : S \rightarrow [0,1]^k$, where k is number of decision criteria or individual objectives for which the agent is optimizing.

Baseline Approaches

We employ two AL models from prior literature, as baselines for comparison: a standard uncertainty sampling approach (**U-sampling**) and a state-of-the-art decision-theoretic approach for arbitrating between diverse query actions (**DT-iros**). Uncertainty sampling algorithms are possibly the most commonly employed class of AL strategies in the literature [16, 17]. They assume a single hypothesis θ and utilize the posterior probability distribution over labels $y \in Y$ given unlabeled instance x, $p_\theta(Y|\mathbf{x})$, in order to detect outliers or instances closest to a decision boundary. Like other standard AL approaches, they query at every turn, each time requesting a label for a maximally informative instance, based upon predetermined selection criteria. A commonly used metric for uncertainty sampling is *prediction entropy*: $-\sum_{y \in Y} p_\theta(y|\mathbf{x}) \log p_\theta(y|\mathbf{x})$. We employ this as our standard AL baseline (**U-sampling**).

Decision-theoretic approaches to active learning simulate all possible outcomes of each candidate query action and optimize with respect to future *expected* utility. This work builds from prior work employing decision theory to arbitrate between diverse types of learning queries, including a supplemental no-query action [90]. The set of decision features investigated were *average classifier discriminability* and *class distribution uniformity*.

Given a set of instances in the agent's purview $\mathbf{X}$ and a task-relevant concept y, the *classifier discriminability* metric assesses the *range* of probabilities over the set of instances: $p_\theta(y|x_{max}) - p_\theta(y|x_{min})$, where x_{max} and x_{min} are the model's prediction of the most and least probable examples of class y, respectively. Range is a standardized metric of statistical dispersion; an average is taken over all $y \in Y$. Class distribution uniformity assesses selection bias in the training sample, due to an unrepresentative class distribution. It is a useful decision feature in sparse data environments, as has traditionally been the assumption in Learning from Demonstration settings, where the learner does not have sufficient evidence to confidently infer the underlying distribution of classes. This metric incentivizes the learner to minimize sample selection bias. Given this work also seeks to arbitrate between *all* action types, we employ this previously published decision-theoretic objective function, where the number of decision features $k = 2$, as our state-of-the-art baseline (**DT-iros**).

Experimental Approaches

We introduce two experimental questioning policies: a learning-centric model intended to improve the state-of-the-art (**DT-task**) and an environmentally-aware active learner (**DT-task-env**). For the learning-centric model, we propose two *additional* decision features that we believe improves the performance of the originally published DT-iros algorithm, even before consideration of environmental context: *instance variation* and *label prediction margin*, defined by Equations 7.3 and 7.4 respectively.

$$IV(s) = \frac{1}{|Y|} \sum_{y \in Y} \frac{\sigma(p(\mathbf{X}|y))}{\mathbb{E}[p(\mathbf{X}|y)]} \tag{7.3}$$

$$PM(s) = \frac{1}{|X|} \sum_{x \in X} p_{\theta_1}(y_1|\mathbf{x}) - p_{\theta_2}(y_2|\mathbf{x}) \tag{7.4}$$

Instance variation is a standardized measure of statistical dispersion. Given a class y and a set of scene instances $\mathbf{X}$, it is a measure of relative standard deviation of the class conditional distribution $p_\theta(\mathbf{X}|y)$. Intuitively, it attempts to assess each classifier's ability to recognize variation *amongst* the set of unlabelled instances.

In the context of concept learning, the class-conditional distribution $p(\mathbf{X}|y)$ can be thought of as the likelihood of each unlabelled instance $\mathbf{x} \in \mathbf{X}$ being selected as an example of class y. Given that multiple, diverse instances within a scene may serve as positive examples of a given class, it seems useful to employ decision features which approximate the learner's ability to recognize diversity amongst the set of unlabelled instances in its purview. Because of this, both *classifier discriminability* and *instance variation* are measures of statistical dispersion, but along different dimensions. Whereas, classifier discriminability is a measure of statistical dispersion over the likelihood of instances belonging to a class, instance variation quantifies the statistical dispersion over the features values of instances. The former rewards the learner for differentiating between the most prototypical and improbable examples of each class; the latter rewards the learner for recognizing greater variation between instances. Both decision features incentivize the selection of queries which increase the learner's *recognition* of the underlying *diversity* that exists within the pool of unlabelled instances.

Given an unlabelled instance x and a distribution over class labels $p(\mathbf{Y}|x)$, *label prediction margin* measures the difference between what the learner predicts to be the most probable label y_1 and second most probable label y_2. Previously employed in AL literature [16], it is a measure of uncertainty; as the margin increases, the learner is more confident about its prediction. It is computed for all scene instances, then averaged. This decision feature incentivizes *accuracy* in the class prediction for each unlabelled instance.

Thus, the first decision function proposed in this work (**DT-task**) subsumes the set of decision features considered by DT-iros, considering *four* learning-centric criteria that each optimize for different aspects of the concept learning problem.

The primary contribution of this work however is in the addition of *environmental context* into the AL agent's objective function. We *introduce* the following environmental features:

- *query budget consumption* – measures the proportion of query budget consumed at turn t, given the query history
- *remaining time usage* – measures the proportion of allocated time remaining after

turn t

- *non-query time passed* – measures the proportion of consecutive turns no query was made within a sliding time window T_w; here the size of the time window is proportional to the rate of environmental change; it is computed as $t_{NQ} = \frac{n_{NQ}}{|T_w|}$; $t_{NQ} \to 0$ when the learner has just queried and $t_{NQ} \to 1$ when the learner has *not* queried throughout the entire duration of the time window; intuitively this metric is intended to penalize the agent for being *too* passive, in a dynamically changing environment

The environmentally-aware agent's objective function (**DT-task-env**) is then composed of a linear combination of *seven* decision features, a subset of which roughly attempt to estimate progress towards learning goals (*i.e.* learning *task* centric) and the remaining features intended to incentivize wise time and resource management (*i.e. environment* centric). All decision-theoretic learners described can arbitrate between *all* communicative action types.

Given the different types of decision features being considered however, it is challenging to decipher how to trade them off (*e.g.* budget consumption versus prediction margin). One key observation is humans can often intuitively reason about decision criteria that are difficult to compare quantitatively; thus, we propose to observe the strategy of a human expert questioner, given the same learning problem, and infer how the expert trades off the given decision criteria.

7.3.2 Imitating an Expert Questioning Strategy

Imitation learning seeks to efficiently learn desired behavior by mimicking a domain expert [18]. Within imitation learning literature, Inverse Reinforcement Learning (IRL) aims to recover the expert's reward (objective) function from demonstrations of a policy [91, 21]. We employ a state of the art IRL algorithm, maximum entropy IRL [92], to infer the weights $w \in W$ of Equation 7.2, for the active learner's decision features, $\forall \phi \in \Phi$, as wielded by an expert.

The maximum entropy loss function L_{ME} maximizes entropy of distributions over paths followed by the expert, while satisfying the constraint that the learner's decision feature counts should ideally match those of the expert. The problem is formulated as follows: Given $\phi_E(\tau)\ \forall\tau \in T^{demo}$, find an optimal weight vector $\mathbf{w}$ such that

$$\mathbf{w}^* = \arg\max_{\mathbf{w}}\ -\sum_{\tau} p(\tau|\mathbf{w})\ \ln p(\tau|\mathbf{w}) \tag{7.5}$$

subject to the constraint

$$\mathbb{E}\left[\phi_E(\tau)\right] = \mathbb{E}\left[\phi_L(\tau)\right] \tag{7.6}$$

where $\phi_E(\tau)$ and $\phi_L(\tau)$ represent the feature counts of the expert and learner respectively, for a trajectory τ. In our problem domain, a trajectory is a sequence of learning states visited and communicative actions taken $\{s_1, a_1, s_2, a_2, ...s_T, a_T\}$ at each time step $t <= T$, the maximum number of iterations allowed in a learning episode.

Optimization using the maximum entropy loss $L_{ME}(\mathbf{w})$ is equivalent to maximizing the log likelihood of the expert demonstrations [92, 18]:

$$\begin{aligned}
\mathbf{w}^* &= \arg\max_{\mathbf{w}}\ L_{ME}(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}}\ \sum_{\tau} p(\tau|\mathbf{w})\ \ln \frac{1}{p(\tau|\mathbf{w})} \\
&\propto \arg\max_{\mathbf{w}}\ \sum_{\tau} p(\tau|\mathbf{w}) \\
\mathbf{w}^* &\propto \arg\max_{\mathbf{w}}\ \sum_{\tau} \ln p(\tau|\mathbf{w})
\end{aligned}$$

Using this formulation, the gradient of the IRL loss, shown in Equation 7.7, is the difference between the empirical feature counts (demonstrated by the expert) and the expected feature counts, computed from sample trajectories generated with $\mathbf{w}$.

$$\nabla_{\mathbf{w}} L_{ME} = \mathbb{E}_{\pi^{\mathbb{E}}}\left[\phi_E(\tau)\right] - \sum_{\tau} p(\tau|\mathbf{w})\phi_L(\tau) \tag{7.7}$$

We used an empirically determined maximum number of iterations as stopping criteria for the IRL algorithm. Weights for the environmentally-aware active learner's objective function were learned offline and tested for generalization in AL episodes under different environmental conditions.

7.3.3 Learning Episode

Figure 7.1 shows the high-level flow for the learning system. For each active learning episode conducted, the task-relevant concepts and questioning strategy are given as input. Within an episode, at each turn t, the agent perceives all objects in its purview, computes its estimate of the posterior probability distributions $p(y|x)\ \forall x,y$ to update learning state s_t, determines the set of candidate actions A_t, computes $EU(a|s_t)\ \forall a \in A_t$, then takes an optimal action a^*. The learning episode concludes once $t = T$.

7.4 Evaluation

This work explores an AL strategy *designed* to optimize for environmental constraints and proposes an imitation learning approach for accomplishing this. Toward this end, we test two hypotheses: (1) Reasoning *additionally* about environmental context can enable an AL agent to adapt its questioning strategy and improve its learning performance under constrained conditions, and (2) Imitation Learning can be used to infer an expert's strategy for managing time and resources allocated to solve a given learning problem, then generalized to other constrained environments. As illustrated in Figure 7.1, each of the questioning strategies conduct their own learning episodes, during which binary classifiers are trained for all task-relevant classes, based upon information gathered. We use recognition accuracy on hold-out test sets for assessing the concept models learned, given each questioning strategy.

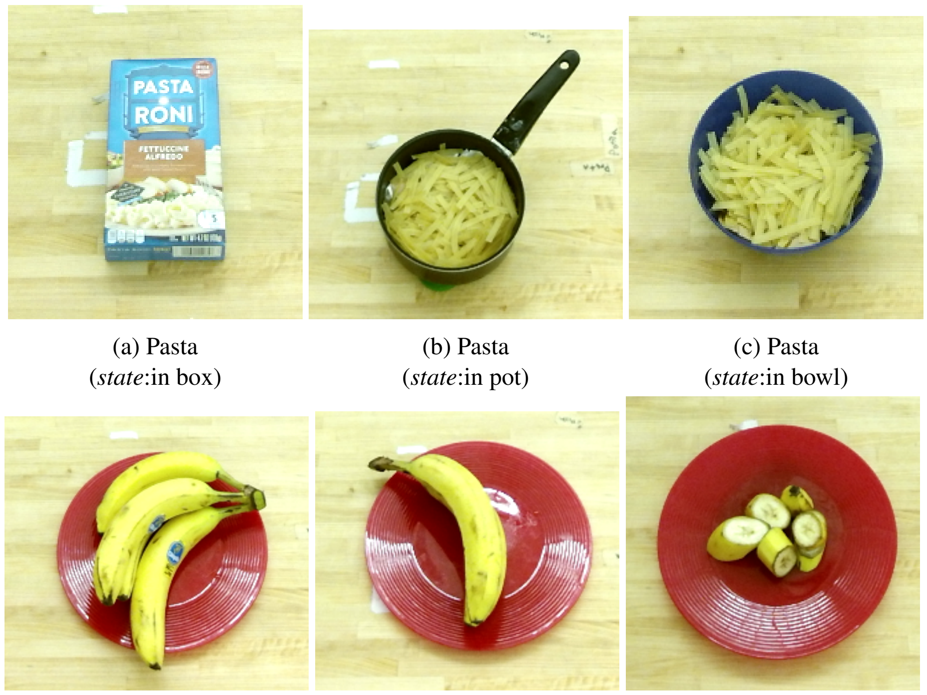
7.4.1 Experimental Design

In evaluating the AL approaches, we focus experiments on a concept grounding task in a dynamic environment, under *different* environmentally constrained conditions. We also examine performance on another task for generalization of the learned decision feature weights across tasks. Both concept grounding tasks are given the same four abstract concepts to ground (main dish, snack, fruit, and beverage), but are generated from different object RGB-D datasets and represent different properties of dynamic change.

The *prepare-lunch* task, the most difficult of the two learning problems, is our focus; it places emphasis on the same objects changing *state*, as one might expect over the course of the task (*e.g.* pasta going from being in a box in the pantry to being cooked in a pot to being served in bowl for lunch.) It was extracted from a local RGB-D object dataset focused on state-change. Figure 7.2 shows an example of the type of dynamic change the learner may expect to see in this task. In the second *pack lunchbox* task, objects do not change state, but have greater within-category diversity. For example, the *fruit* class contains apples, oranges, peaches, and pears, and the *beverage* class contains varieties of both soda and water. This task was extracted from the University of Washington RGB-D dataset of common household objects [77]. As both tasks are from prior literature, details regarding data collection can be found in [90].

Since RGB-D datasets were being used for evaluation, in order to create a learning environment that more closely approximates real-world settings, we simulated multi-modal features, representing features extracted from a robot's other sensors[1]. Gaussian noise was added to all simulated features since robot sensor data is typically noisy. We also simulated dynamic change in the environment by sampling a new set of object images at a predetermined rate, to represent the scene changing. At each turn t, O contains only one observation

[1]*object's location relative to interest points in the environment (e.g. counter top, stove, refrigerator, pantry), the object's location relative to the robot base, absolute location of robot's base in the environment, location of the robot's base with respect to the counter top, the robot's joint positions for each arm, pose of the robot's hands, robot's hand states (open vs closed), weight of the object, and max/min/average volume of noise in the environment over duration of learning episode*



(a) Pasta (*state*:in box)

(b) Pasta (*state*:in pot)

(c) Pasta (*state*:in bowl)

(d) Banana (*state*:bunch)

(e) Banana (*state*:single)

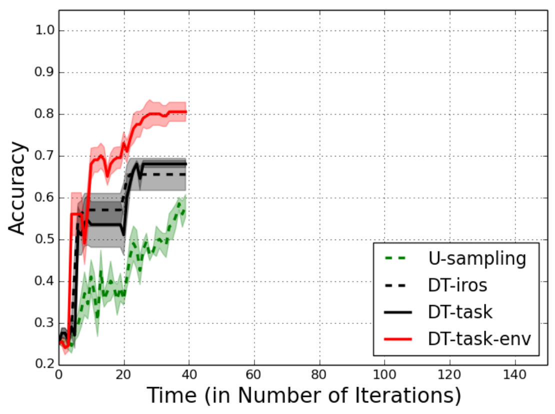
(f) Banana (*state*:sliced)

Figure 7.2: Illustration of object state changes for *main dish* and *fruit* objects classes in *prepare-lunch* task.

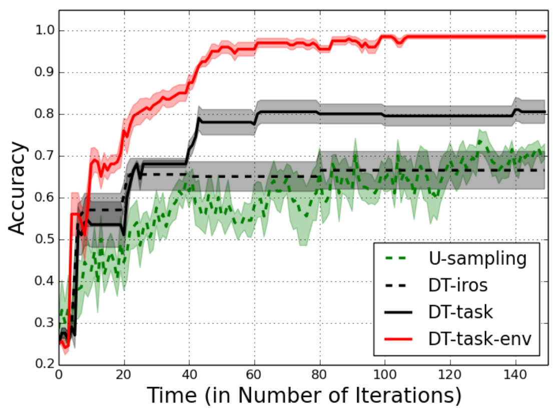
(image) of each object in the scene. To simulate environmental change, the perceptual system generates a new set of observations. Else, it outputs the set of observations from $t-1$. In the prepare-lunch task, objects can change state.

Using the complete RGB-D object datasets, we generated five smaller *task* training data samples and one disjoint hold-out test sample for each task. Since the UW dataset is several orders of magnitude larger than the local dataset created, the data sample sizes vary by task. Each of the training and test task samples are 80 images and 40 images respectively for the prepare-lunch task and 3200 images and 800 images respectively for the pack-lunchbox task.

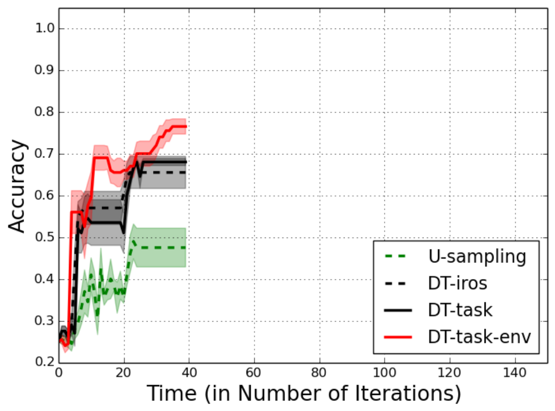
There are four AL algorithms being evaluated on each task: the uncertainty sampling baseline (**U-sampling**), two task-centric decision-theoretic learners (**DT-iros** and **DT-task**), and our experimental environmentally-aware decision-theoretic approach (**DT-task-env**), which reasons about *both* learning objectives *and* environmental constraints.



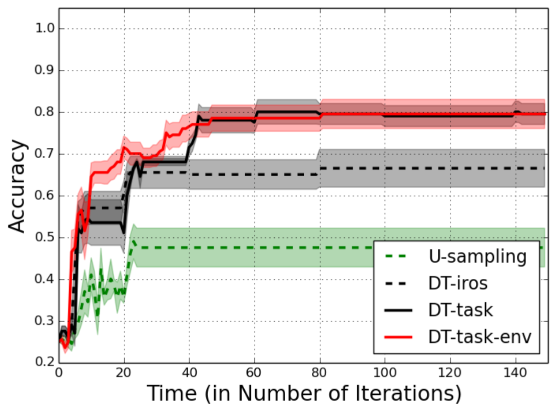
(a) CONDITION 3: Constrained *Time*



(b) CONDITION 4: Unconstrained



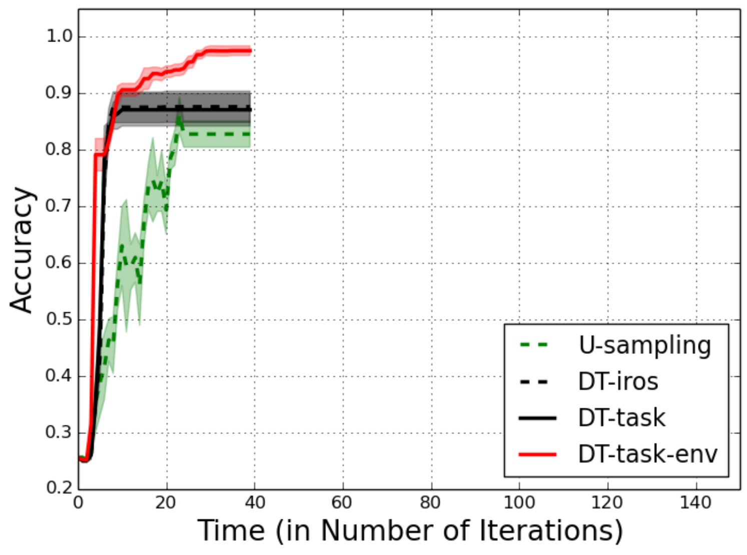
(c) COND 1: Constrained *Time* & *Budget*



(d) CONDITION 2: Constrained *Budget*

Figure 7.3: Prepare Lunch Task. Shows performance (test accuracy with standard error) for each AL strategy under different environmentally constrained conditions. Parameters of allocated time and query budget imposed on the learner vary, with: (a) *only time* constrained [budget: high (500), time: low (40)], (b) *neither* time nor query budget constrained [budget: high (500), time: high (150)], (c) *both* time and query budget constrained [budget: low (25), time: low (40)], and (d) *only* query *budget* constrained [budget: low (25), time: high (150)].

For training of DT-task-env decision feature weights, an expert questioner was given a very constrained query budget (15) and time allocation (30 turns) to ground the *prepare-lunch* task concepts. During training, the environmental scene changed every 10 turns, and major object state changes took place with each scene change; thus spreading the query budget out over the allocated time period affords the opportunity to acquire a more diverse and representative training sample. This was a key part of the strategy employed by the expert used, which was one of the authors of this paper.



(a) CONDITION 1: Constrained *Time* & *Budget*

Figure 7.4: Pack Lunchbox Task. Shows performance (test accuracy with standard error) on a separate task, under the most constrained experimental condition: *both* time *and* query budget constrained.

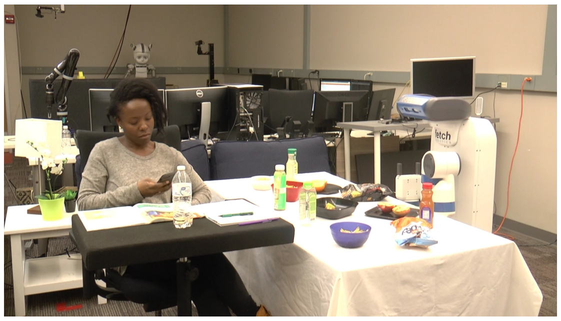
The expert provided three demonstrations of questioning sessions (learning episodes) in the training scenario. As part of the strategy demonstrated, the expert also always requested relevant features for discriminating between concepts (FSQ) early in the learning episode (within the first five turns), focused most queries on the least costly query type, and focused on quickly acquiring representative training examples for each class. During IRL training, the maximum number of iterations was set to 100, and we selected the set of weights $\mathbf{w}^*$ that performed best on the validation set. Qualitatively examining the rollouts associated with $\mathbf{w}^*$ under the training conditions, the behavior of the imitation learner was able to closely match the expert's questioning strategy. Environmental conditions were held constant across demonstrations and IRL training. Values changed for testing were time and budget allocation and frequency of environmental change.

7.4.2 Results

To test our hypotheses, we first investigated generalization to *different* environmentally constrained conditions. The goal was to vary time on one axis and resources (query bud-

(a) Wise management of time and cognitive resources, partner fully engaged with robot

(b) Wise usage of time while partner is present, though partner is multitasking


(c) Robot continues to contemplate questions, as partner finds ways to keep busy

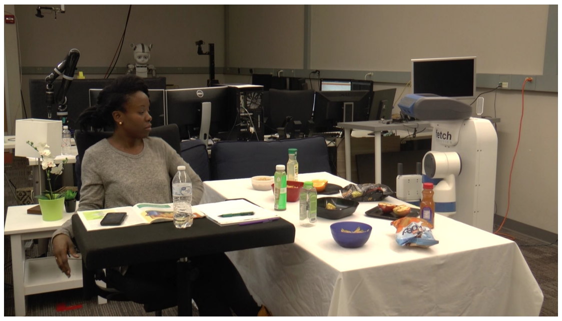

(d) Missed opportunity to engage, as partner appears increasingly bored

Figure 7.5: Juxtaposition of Human Partner Engagement, as a result of Robot's Management of Learning Session. This illustration is primarily to provide context for what the learning sessions look like and why the problem of managing the learning interaction is important. In the top images, the robot appears to be wisely managing its time with the human partner, though the human has different amounts of cognitive resources available (comparing scenario (a) with (b)). In both bottom images, by contrast, the robot appears *not* to be taking full advantage of the time allocated with the human partner. On the left, the partner keeps busy by checking her cell phone, but on the right is becoming increasingly bored as the robot seemingly continues to contemplate.

get) on the other axis. In order to have a feasible stopping point, we could not truly allow unlimited time or query budget; however, as defined here, the *constrained* versus *unconstrained* parameters denote an order of magnitude difference in allocation. All action types were assigned an a-priori cost, which is 2 for demo queries and feature subset queries, 1 for label queries, and 0 for no query. This can be assigned in any way desired, but for our purposes, was intended to map roughly with the cognitive load required by the teacher in answering a particular type of question. Figure 7.3 shows learning curves for each combination of time and resource parameters. For each task, learning curves are averaged over 5 runs, each run sampling from a different pre-generated task training data sample. Examining the subfigures: from left to right, time allocation is increased (from 40 to 150) and from bottom to top, query budget is increased (from 25 to 500). Thus fig 7.3c is the most constrained (budget 25, time 40) and most similar to the training scenario (budget 15, time 30). Overall, the left half roughly corresponds to the agent being given approximately 20 questions, assuming the most costly queries are minimized, whereas the right half corresponds to the agent being given unlimited queries during the time allocated. In testing, once a strategy has exceeded its query budget, it is no longer allowed to make queries, representing complete resource consumption. Thus for the remainder of the episode, it must select the no-query action at no cost. It should also be stated that we assume at most one query per turn.

We used the Mann-Whitney U-test to perform pairwise statistical comparisons of our experimental approach (DT-task-env) with each of the baseline approaches at the end of the allocated learning time, for all four experimental conditions. A one-tailed test was conducted, as the goal was to understand if the experimental environmentally-aware approach *improves* performance over the baseline task-centric approaches to AL. In conditions 1, 3, and 4, DT-task-env statistically outperforms *all* other approaches by the end of the learning time. In condition 2, it statistically outperforms all approaches except DT-task, compared to which it performs equivalently. Using the second (pack-lunchbox) task, we were able to replicate our results in another domain. We only show the most constrained condition (limited time *and* query budget) in Figure 7.4. Overall, there is a clear pattern. DT-task-env *always* performs *at least* as well as the task-centric baselines but moreover *dominates* task-centric approaches under most of the environmental conditions examined. This confirms our second hypothesis that a questioning strategy learned through imitation of an expert in one environment *can* be used to generalize to other constrained environments. Our first hypothesis however is *not* supported by findings from Condition 2.

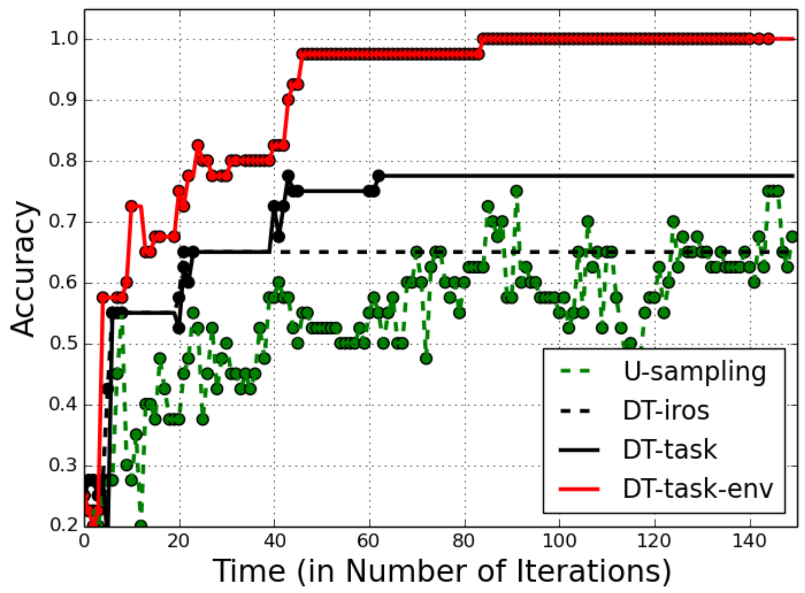
To better understand what behavior leads to these findings, we analyze learning episodes from one training data sample under two different experimental conditions in Figure 7.6.

The dots represent points where queries were made for the given strategy. We find that given a limited query budget and ample time (figure 7.6b), DT-task and DT-task-env employ very similar *conservative* strategies. Both use most of their budget closer to the beginning of the episode, as they attempt to build initial concept models, but also attempt to modulate budget consumption with rate of environmental change. However, when the agent is allowed to ask unlimited queries given the same time frame (figure 7.6a), these two strategies behave very differently. Whereas the task-centric learners employ exactly the same strategy (because they have no ability to reason about environmental constraints), DT-task-env employs a very *liberal* strategy. In fact, it makes at least an order of magnitude more queries both than DT-task and DT-iros (134 versus 22 and 9), largely accounting for its complete domination over the other strategies. Also notably, U-sampling asks a question at *every* turn until it exceeds its budget, since it is able to modulate *neither* for environmental change *nor* for external constraints. It also does not have the capability to autonomously select a feature subset query, so it must rely upon computational feature selection for solving its learning problem. The decision-theoretic approaches, by contrast, are able to reason about and request an FSQ early in the episode, making them significantly more sample efficient.

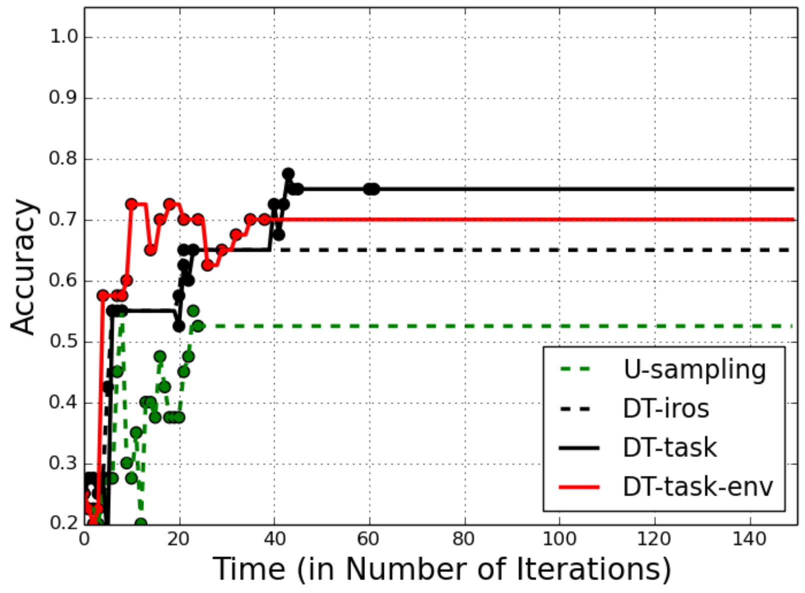
The key implication of all of the experimental findings is that the DT-task-env strategy has the ability to effectively *adapt* its questioning behavior *both* to the rate of environmental change (like DT-iros and DT-task) *and* to time and resource constraints imposed externally. In more realistic environments, this is compelling as it gives human partners the capability to specify their own time and cognitive load constraints, with the understanding that the agent can integrate this knowledge into its reasoning about the learning task.

7.5 Conclusion

This work contributed a first exploration and novel computational approach for solving the problem of active learning under externally imposed time and resource constraints. Imitation of an expert questioner's policy was used to learn to weight the diverse set of decision criteria for an environmentally-aware active learner. Experiments were conducted under various environmentally constrained conditions and on two concept learning tasks. Key findings show the experimental approach presented statistically outperformed a standard uncertainty sampling baseline and the strictly task-centric active learners under most environmental conditions; thus representing a promising alternative for active learners in more realistic human environments.



(a) CONDITION 4: Unconstrained



(b) CONDITION 2: Constrained *Budget*

Figure 7.6: Questioning Behavior of each Strategy in Prepare-Lunch Task for *one* training sample. Dots indicate when a query is made.

CHAPTER 8
CONCLUSION AND OPEN QUESTIONS

This thesis can be partitioned into two high-level units. First, we sought to understand if we could leverage interaction with a human agent to help a robotic agent solve the symbol (or concept) grounding problem and how sample efficient the interactive learning process was. Given that knowledge, we sought to enable the agent with algorithms to autonomously guide its own learning process, by reasoning about its questioning policy for actively acquiring the concept knowledge it aimed to learn.

Specifically, we set out to validate the claim: **In the context of robot learning from human demonstrations in changeable and resource-constrained environments, enabling the robot to actively elicit multiple types of information through questions, and to reason about *what* question to ask and *when*, leads to improved learning performance.**

8.1 Summary of Thesis Contributions

Toward that end, this thesis has made the following contributions:

- *Learning from Demonstration for Task-Situated Concept Grounding:* Employed LfD to perceptually ground object and semantic location symbols (concepts), associated with abstract task recipes, in robot's situated environment
- *Concept Grounding using Human-Driven Feature Selection:* Introduced human-driven feature selection for robot task learning and several approaches for eliciting feature information from human teachers
- *Active Concept Grounding through Autonomous Arbitration of Diverse Query Types:* First to explore active learning algorithms for eliciting *both* feature *and* instance in-

formation from an oracle and introduced a decision-theoretic framework for arbitrating between diverse types of information, as well as inferring when to make queries in a dynamically changing environment.

- *Active Concept Grounding within Time and Resource Constrained Environments:* Contributed the first active learning algorithm for explicitly trading off learning objectives with environmental constraints externally imposed on the learner, and first work to use imitation of an expert questioner to infer an active learning policy.

8.2 Open Questions

Yet, there are still many open questions that remain, especially if one important future goal is to deploy an intelligent and curious information gathering agent capable of reasoning about its own goals, and leveraging interaction towards accomplishing them. We categorize the open challenges discussed into six broad categories:

8.2.1 Perception

The properties of environmental change captured by the object datasets used in this thesis serve only as a proxy for partial observability encountered in the real world. Additionally, the scenes considered by the agent were designed with simple object configurations, so as to circumvent the complex scene segmentation problem, which is still an active area of computational perception research. It follows then that one open question pertains to how the queries for grounding concepts change when one considers more complex perceptual data as input. For example, considering cluttered scenes, significant occlusion in scenes, or even video feed where there is a continuous stream of dynamic change in the environment. Another question that arises is how the problem changes considering pixel feature space.

8.2.2 Behavior Generation

The active learning models contributed in this thesis are intended for use by any type of artificial agent and thus agnostic to an agent's embodiment. However, *what* an agent decides to ask cannot always be decoupled from *how* it must use its embodiment to execute the query. For example, query cost may increase if a query requires fine-grained manipulation that is difficult for the agent to maneuver or simply takes a longer time to generate than other queries. Ideally, this should be incorporated into the agent's model or decision function. One question that arises then is how to incorporate additional costs associated with time or energy expenditure of candidate behaviors that can be generated, given each query. Other questions are (1) how to decipher the most appropriate behaviors to generate, given a query, (2) how to evaluate behaviors generated, and (3) whether noise/errors in motion planning and execution should be incorporated into the active learning model and if so, how to do this in a principled and generalizable way.

8.2.3 Scale

The problem instances considered in this thesis contained a relatively small number of concepts and objects; however, the active learning framework was designed to be independent of the specific problem instantiation and applicable to the broader class of multiclass classification problems in sparse data environments. In fact, it is the case that concepts and instances could be dynamically added or removed, even within the same learning episode. Moreover, decision theory provides a principled and interpretable framework for making decisions and has the following key advantages over other families of active learning approaches: (1) ability to directly compare utitlity across a diverse set of actions and (2) ability to directly optimize for desired future outcomes, which often results in maximal learning performance. One key challenge with decision-theoretic apporaches however is how computationally intensive they are. The amount of deliberation necessary at each turn is dependent upon the number of possible resulting new states, which is dervied from considering every feasible $\langle query, response \rangle$ combination and computing an expectation. So one important question that arises is how to scale decision-theoretic approaches as the problem space grows (*i.e.* number of instances and concepts increase substantially). There is also an important open question about how to speed up the deliberation process of the decision theoretic approaches explored in this thesis to run in ”interaction-time”, particularly as the training sample size grows and retraining all classifiers at each turn, for every possible outcome, becomes too computationally expensive. By ”interaction time”, we mean seemingly natural timing given an online interaction with a human user. This largely mimics the time one would expect a human to deliberate before asking a new question.

8.2.4 Information Gathering

Within the space of questioning for information gathering, it would be interesting to move beyond queries currently considered by active learning literature and enable reasoning about a richer space of queries. For example: queries that allow new features completely unknown to the agent to be discovered, queries that synthesize new instances, queries that test affordances of objects, or queries that request explanations from the teacher. Some of the key challenges with expanding the types of queries considered by the agent are how to represent these queries, how to predict candidate responses in order to reason about possible outcomes, and how to incorporate knowledge acquired from each type of query.

Regarding the broader goal of information gathering, there is an interesting question that arises about the combination of active learning and active perception. In the scenarios considered in this thesis, the agent passively observes its environment and only actively queries a human partner when confusion arises. But it would be even more powerful if the agent could actively explore and perturb its environment in order to seek out additional examples not in its purview and actively find new candidate queries. This is especially true if it needs to maximize the time frame for which it has access to the human. This would introduce a tradeoff between exploration of its environment and exploration through querying its human partner. This also expands the set of resources it must consider, *e.g.* constraints on power consumption become more important as it actively navigates its world. Another advantage of an information gathering framework that autonomously reasons about both queries and environmental exploration is the opportunity to leverage the complimentary fields of active learning and semi-supervised learning, towards solving the learning problem. Whereas active learning provides the agent with the capability to query an oracle for labels to confusing unlabeled instances, semi-supervised learning focuses on extrapolating from what it has already learned and independently finding new training instances amongst the vast amounts of unlabeled data in its environment.

To build upon this further, if we aim to create a *curious* information gathering agent, the field of computational curiosity has investigated several metrics for enabling an agent with a curiosity-driven exploration policy. These include the metrics of novelty, surprise, uncertainty, conflict, change, and complexity. So one interesting question is how employing and combining subsets of these metrics for exploration would impact the behavior of the agent. An agent can also exhibit both intrinsic and extrinsic motivations for curiosity. Thus, there is also an inherent question about how one defines and enables an *optimal* curiosity-driven robotic agent towards continual learning in physical environments, given that it is capable of employing different faculties for information gathering and that it can optimize for many different definitions of ”curious”.

8.2.5 Communication and Interaction

The active learning contributions within this thesis can be seen as enabling the agent to reason about its communication policy, and to this end, are focused on questions of *what* to communicate and *when* to communicate, given its goals. Another very interesting question that naturally arises is: when the agent decides to make a query, *how* should it communicate its intent? Since robots have embodiment, this is not only a question of natural language generation, but also of natural non-verbal behavior generation and synchronization between multiple modalities. With respect to language generation, an interesting direction might be autonomous generation of explanations for questions that may seem redundant or would otherwise raise questions in the mind of a user, as to the source of the agent's uncertainty or confusion. Explanation generation is less about communicating intent and more about communicating internal state, which is also an important aspect of *how* to communicate in a way that helps one achieve self-goals. Transparency of internal state can be important for managing perception by and thus relationships with other agents. Thus accordingly, the question of how also significantly impacts how sociable the agent will be perceived by users and the overall usability of the system. Conducting user studies to understand these dimensions also becomes important.

8.2.6 Multi-Agent Modeling

The decision-making framework used for the learning agent is designed for reasoning in stochastic environments, and the objective function attempts to reason about uncertainty in state, potentially deriving from partial observability or dynamic change in the environment. The framework assumes however that a transition model is provided (though only an estimate that changes over time) and does not explicitly reason about other agents. These are common assumptions in active learning literature. However, there are interesting research questions that could be explored around explicitly modeling the human teacher. In Dialogue research, much of the literature builds user models from dialogue or conversation datasets. Similarly, it might be interesting to have the agent model different teaching strategies and automatically infer the strategy of the user, in order to better estimate the informativeness of open-ended queries, *e.g.* demo queries, feature subset queries.

It might also be interesting to have the agent automatically infer time and query budget constraints, based upon perceived social cues by the teacher. For example, if a person continuously checks their watch, it suggests they have little time to continue devoting to the interaction. Similarly, by tone of voice, an agent may be able to detect impatience or annoyance on the part of the user, suggesting a reduced query budget. A key challenge here becomes the recognition of relevant social cues and how to quantify those cues to predict time and budget constraints, which serve as input to the decision-making algorithm.

8.2.7 Cognition

There are interesting open questions that lie mostly in the realm of cognition for the agent. This thesis assumed the agent to be memoryless and for the active concept grounding work, scenes with relevant instances were given to the agent, so it did not have to reason about directing its own attention. It would be interesting to relax both of these assumptions and explore how the problem changes with faculties for memory retrieval and attention selection. Memory retrieval would be especially important if the agent were placed in a longer term setting, where a primary goal would be lifelong learning across multiple tasks. Attention selection is important for efficiently searching for instances of concepts in large or complex environments or if employing active perception to explore the environment.

Finally, one very fascinating part of cognition is metacognition, and the problem of lifelong learning provides a useful context for thinking about why this might be important. Given that an agent has both long-term goals and short-term goals and is continually learning over the course of its lifetime, introspection is important. This could mean explicit reasoning about its current policy's success in achieving an identified goal or even which goals are best for it to focus its attention on in the current context. For task-situated concept learning, it would be interesting if an agent could reason about it's current questioning policy with respect to its learning goals and determine how to modify its policy, if not effective in achieving the goals. One example might be by adapting the weights of its decision features online because certain features are more useful in the beginning of an episode, when the agent is building initial models, but much less so and thus slowing learning, once it has reasonable models and is only making refinements. Another example might be in the space of goal reasoning and determining that it should context-switch or focus on a different set of learning goals because it is now situated in a different part of the environment and is unlikely to see instances of the concepts it was previously focused on. Though this is likely the most abstract and ill posed of all open questions discussed, enabling a lifelong learning agent to reason about its current policy given short-term goals as well as long-term learning goals and how to be intentional about adapting those is a critical component of artificial general intelligence.

REFERENCES

[1] H Christensen, T Batzinger, K Bekris, K Bohringer, J Bordogna, G Bradski, O Brock, J Burnstein, T Fuhlbrigge, R Eastman, *et al.*, "A roadmap for us robotics: From internet to robotics," *Computing Community Consortium and Computing Research Association, Washington DC (US)*, 2009.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[3] S. Chernova and A. L. Thomaz, *Robot Learning from Human Demonstration*. Morgan and Claypool Publishers, 2014.

[4] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, pp. 55–66, 2010.

[5] M. Cakmak, "Guided teaching interactions with robots: Embodied queries and teaching heuristics," PhD thesis, Georgia Institute of Technology, 2012.

[6] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, p. 1, 2009.

[7] M. Lopes, F. Melo, and L. Montesano, "Active learning for reward estimation in inverse reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, Springer, 2009, pp. 31–46.

[8] R. D. King, K. E. Whelan, F. M. Jones, P. G. Reiser, C. H. Bryant, S. H. Muggleton, D. B. Kell, and S. G. Oliver, "Functional genomic hypothesis generation and experimentation by a robot scientist," *Nature*, vol. 427, no. 6971, p. 247, 2004.

[9] C. Chao, M. Cakmak, and A. L. Thomaz, "Transparent active learning for robots," in *ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2010, pp. 317–324.

[10] J. Kulick, M. Toussaint, T. Lang, and M. Lopes, "Active learning for teaching a robot grounded relational symbols," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, AAAI Press, 2013, pp. 1451–1457.

[11] B. Hayes and B. Scassellati, "Discovering task constraints through observation and active learning," in *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 4442–4449.

[12] M. Cakmak and A. L. Thomaz, "Designing robot learners that ask good questions," in *ACM/IEEE Int Conf on Human-Robot Interaction*, 2012, pp. 17–24.

[13] C. Chao, M. Cakmak, and A. L. Thomaz, "Towards grounding concepts for transfer in goal learning from demonstration," in *Development and Learning (ICDL), 2011 IEEE International Conference on*, IEEE, vol. 2, 2011, pp. 1–6.

[14] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 335–346, 1990.

[15] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[16] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[17] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and information systems*, pp. 1–35, 2013.

[18] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[19] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[21] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 1.

[22] O. Kroemer, R. Detry, J. Piater, and J. Peters, "Combining active learning and reactive control for robot grasping," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1105–1116, 2010.

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

[24] E Gribovskaya, F. d'Halluin, and A. Billard, "An active learning interface for bootstrapping robot's generalization abilities in learning from demonstration," in *RSS Workshop Towards Closing the Loop: Active Learning for Robotics*, 2010.

[25] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, "Active incremental learning of robot movement primitives," 2017.

[26] N. Chen, A. Klushyn, A. Paraschos, D. Benbouzid, and P. Van der Smagt, "Active learning based on data uncertainty and model sensitivity," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1547–1554.

[27] Y. Cui and S. Niekum, "Active reward learning from critiques," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6907–6914.

[28] C. Basu, M. Singhal, and A. 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*, ACM, 2018, pp. 132–140.

[29] J. Thomason, A. Padmakumar, J. Sinapov, J. Hart, P. Stone, and R. J. Mooney, "Opportunistic active learning for grounding natural language descriptions," in *Conference on Robot Learning*, 2017, pp. 67–76.

[30] M. Racca, A. Oulasvirta, and V. Kyrki, "Teacher-aware active robot learning," in *Proceedings of the 2019 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2019.

[31] M. Racca and V. Kyrki, "Active robot learning for temporal task models," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2018, pp. 123–131.

[32] M. Majnik, M. Kristan, and D. Skočaj, "Knowledge gap detection for interactive learning of categorical knowledge," 2013.

[33] Q. Wu and C. Miao, "Curiosity: From psychology to computation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, p. 18, 2013.

[34] L. Macedo and A. Cardoso, "Towards artificial forms of surprise and curiosity," in *Proceedings of the European Conference on Cognitive Science, S. Bagnara, Ed*, Citeseer, 1999, pp. 139–144.

[35] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE transactions on evolutionary computation*, vol. 11, no. 2, pp. 265–286, 2007.

[36] J. Schmidhuber, "Curious model-building control systems," in *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, IEEE, 1991, pp. 1458–1463.

[37] T. B. Kashdan, P. Rose, and F. D. Fincham, "Curiosity and exploration: Facilitating positive subjective experiences and personal growth opportunities," *Journal of personality assessment*, vol. 82, no. 3, pp. 291–305, 2004.

[38] M. Tenorth and M. Beetz, "KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System," *International Journal of Robotics Research (IJRR)*, 2013, Accepted for publication.

[39] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 3, pp. 643–651, 2013.

[40] D. K. Misra, J. Sung, K. Lee, and A. Saxena, "Tell me dave: Context-sensitive grounding of natural language to manipulation instructions," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, 2014.

[41] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus, "Interpreting and executing recipes with a cooking robot," in *Experimental Robotics*, Springer, 2013, pp. 481–495.

[42] D. K. Roy and A. P. Pentland, "Learning words from sights and sounds: A computational model," *Cognitive science*, vol. 26, no. 1, pp. 113–146, 2002.

[43] C. Chao, M. Cakmak, and A. Thomaz, "Towards grounding concepts for transfer in goal learning from demonstration," in *Proceedings of the Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*, IEEE, vol. 2, 2011, pp. 1–6.

[44] R. Cubek, W. Ertel, and G. Palm, "High-level learning from demonstration with conceptual spaces and subspace clustering," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 2592–2597.

[45] G. Hovland, P. Sikka, and B. McCarragher, "Skill acquisition from human demonstration using a hidden markov model," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, Ieee, vol. 3, 1996, pp. 2706–2711.

[46] O. Jenkins, M. Mataric, and S. Weber, "Primitive-based movement classification for humanoid imitation," in *Proceedings, First IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2000)*, 2000.

[47] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Inelligence Research*, vol. 34, no. 1, pp. 1–25, 2009.

[48] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, IEEE, 2008, pp. 367–372.

[49] M. van Lent and J. E. Laird, "Learning procedural knowledge through observation," in *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, Victoria, British Columbia, Canada: ACM Press, 2001, pp. 179–186, ISBN: 1-58113-380-4.

[50] H. Veeraraghavan and M. Veloso, "Teaching sequential tasks with repetition through demonstration (short paper)," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AMMAS '08)*, 2008.

[51] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2003, pp. 241–248.

[52] S. Ekvall and D. Kragic, "Robot learning from demonstration: A task-level planning approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 223–234, 2008.

[53] B. Kuipers, "The spatial semantic hierarchy," *Artificial Intelligence*, vol. 119, no. 1, pp. 191–233, 2000.

[54] H. Zender, O. M. Mozos, P. Jensfelt, G.-J. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 493–502, 2008.

[55] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 3515–3522.

[56] M. R. Walter, S. Hemachandra, B. Homberg, S. Tellex, and S. Teller, "Learning semantic maps from natural language descriptions," Robotics: Science and Systems, 2013.

[57] C. Diuk, L. Li, and B. R. Leffler, "The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 2009, pp. 249–256.

[58] D. Floreano, T. Kato, D. Marocco, and E. Sauser, "Coevolution of active vision and feature selection," *Biological cybernetics*, vol. 90, no. 3, pp. 218–228, 2004.

[59] S. Zhang, L. Xie, and M. D. Adams, "Entropy based feature selection scheme for real time simultaneous localization and map building," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, IEEE, 2005, pp. 1175–1180.

[60] D. Vasquez, B. Okal, and K. O. Arras, "Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, 2014, pp. 1341–1346.

[61] S. Whiteson, P. Stone, K. O. Stanley, R. Miikkulainen, and N. Kohl, "Automatic feature selection in neuroevolution," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, ACM, 2005, pp. 1225–1232.

[62] S. Loscalzo, R. Wright, and L. Yu, "Predictive feature selection for genetic policy search," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 5, pp. 754–786, 2015.

[63] M. Swangnetr and D. B. Kaber, "Emotional state classification in patient–robot interaction using wavelet analysis and statistics-based feature selection," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 1, pp. 63–75, 2013.

[64] D. L. Vail and M. M. Veloso, "Feature selection for activity recognition in multi-robot domains.," in *AAAI*, vol. 8, 2008, pp. 1415–1420.

[65] M. Trincavelli and A. Loutfi, "Feature selection for gas identification with a mobile robot," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 2852–2857.

[66] J.-H. Kim, S. Jo, and B. Y. Lattimer, "Feature selection for intelligent firefighting robot classification of fire, smoke, and thermal reflections using thermal infrared images," *Journal of Sensors*, vol. 2016, 2016.

[67] L. Y. Chang, N. S. Pollard, T. M. Mitchell, and E. P. Xing, "Feature selection for grasp recognition from optical markers," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, IEEE, 2007, pp. 2944–2950.

[68] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Using depth and appearance features for informed robot grasping of highly wrinkled clothes," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 1703–1708.

[69] J. Y. Zou, K. Chaudhuri, and A. T. Kalai, "Crowdsourcing feature discovery via adaptively chosen comparisons," *arXiv preprint arXiv:1504.00064*, 2015.

[70] D. Parikh and K. Grauman, "Interactively building a discriminative vocabulary of nameable attributes," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 1681–1688.

[71] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 1778–1785.

[72] S. Rosenthal, A. K. Dey, and M. Veloso, "How robots' questions affect the accuracy of the human responses," in *IEEE Int. Symp. on Robot and Human Interactive Communication*, IEEE, 2009, pp. 1137–1142.

[73] L. C. Cobo, P. Zang, C. L. Isbell Jr, and A. L. Thomaz, "Automatic state abstraction from demonstration," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, Citeseer, vol. 22, 2011, p. 1243.

[74] A. J. B. Trevor, S. Gedikli, R. B. Rusu, and H. I. Christensen, "Efficient organized point cloud segmentation with connected components," in *3rd Workshop on Semantic Perception, Mapping and Exploration (SPME)*, Karlsruhe, Germany, 2013.

[75] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1, pp. 245–271, 1997.

[76] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[77] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1817–1824.

[78] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[79] M. A. Hall, "Correlation-based feature subset selection for machine learning," PhD thesis, University of Waikato, Hamilton, New Zealand, 1998.

[80] U. Berkeley, *Intro to ai project 3: Reinforcement learning*, 2014 (accessed 01-March-2017).

[81] K. Bullard, S. Chernova, and A. L. Thomaz, "Human-driven feature selection for a robotic agent learning classification tasks from demonstration," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6923–6930.

[82] K. Bullard, B. Akgun, S. Chernova, and A. L. Thomaz, "Grounding action parameters from demonstration," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, IEEE, 2016, pp. 253–260.

[83] R. Lomasky, C. E. Brodley, M Aernecke, D. Walt, and M Friedl, "Active class selection," in *Machine learning: ECML 2007*, Springer, 2007, pp. 640–647.

[84] S. Chernova and M. Veloso, "Multi-thresholded approach to demonstration selection for interactive robot learning," in *ACM/IEEE Int. Conf. on Human robot interaction*, 2008, pp. 225–232.

[85] M. Cakmak, C. Chao, and A. L. Thomaz, "Designing interactions for robot active learners," *Autonomous Mental Development, IEEE Transactions on*, vol. 2, no. 2, pp. 108–118, 2010.

[86] W. B. Knox, P. Stone, and C. Breazeal, "Training a robot via human feedback: A case study," in *International Conference on Social Robotics*, Springer, 2013, pp. 460–470.

[87] V. Gonzalez-Pacheco, M. Malfaz, A Castro-Gonzalez, J. C. Castillo, F Alonso, and M. A. Salichs, "Analyzing the impact of different feature queries in active learning for social robots," *International Journal of Social Robotics*, pp. 1–14, 2018.

[88] K. Bullard, S. Chernova, and A. L. Thomaz, "Human-driven feature selection for a robot learning classification tasks from demonstration," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on. IEEE*, 2018, pp. 6923–6930.

[89] S. Rosenthal and M. Veloso, "Modeling humans as observation providers using pomdps," in *RO-MAN, 2011 IEEE*, IEEE, 2011, pp. 53–58.

[90] K. Bullard, A. L. Thomaz, and S. Chernova, "Towards intelligent arbitration of diverse active learning queries," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 6049–6056.

[91] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, vol. 1, 2000, p. 2.

[92] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.