A comparison of contextual bandit approaches to human-in-the-loop robot task completion with infrequent feedback

Matt McNeill  
*Fordham University, mmcneill6@fordham.edu*

Damian Lyons  
*Fordham University, dlyons@fordham.edu*

Follow this and additional works at: [https://research.library.fordham.edu/frcv_facultypubs](https://research.library.fordham.edu/frcv_facultypubs)

Part of the Robotics Commons

**Recommended Citation**  

This Conference Proceeding is brought to you for free and open access by the Robotics and Computer Vision Laboratory at Fordham Research Commons. It has been accepted for inclusion in Faculty Publications by an authorized administrator of Fordham Research Commons. For more information, please contact considine@fordham.edu, bkilee@fordham.edu.
A comparison of contextual bandit approaches to human-in-the-loop robot task completion with infrequent feedback

Matthew McNeill  
Dept. of Computer and Information Science  
Fordham University  
New York, NY  
mmcneill6@fordham.edu

Damian Lyons  
Dept. of Computer and Information Science  
Fordham University  
New York, NY  
dlyons@fordham.edu

Abstract—Artificially intelligent assistive agents are playing an increased role in our work and homes. In contrast with currently predominant conversational agents, whose intelligence derives from dialogue trees and external modules, an autonomous domestic or workplace robot must carry out more complex reasoning. Such a robot must make good decisions as soon as possible, learn from experience, respond to feedback, and rely on feedback only as much as necessary. In this research, we narrow the focus of a hypothetical robot assistant to a room-tidying task in a simulated domestic environment. Given an item, the robot chooses where to put it among many destinations, then optionally receives feedback from a human operator. We frame the problem as a contextual bandit, a reinforcement learning approach frequently used in Web recommendation systems. We evaluate $\epsilon$-greedy and LinUCB action selection methods under a variety of infrequent feedback scenarios, with several methods for managing the lack of feedback. Our empirical results show that, while early-episode performance and overall accuracy of $\epsilon$-greedy action selection can be improved through learning from no-response feedback and careful management of remembered training episodes, a baseline LinUCB approach outperforms $\epsilon$-greedy action selection in early-episode performance, overall accuracy, and simplicity.

I. INTRODUCTION

Artificially intelligent assistive agents are playing an increased role in our work and homes. In contrast to currently predominant conversational agents, whose intelligence derives from dialogue trees and external functionality modules, an autonomous domestic or workplace robot must carry out more complex reasoning. Not all robotic applications have the luxury of a fixed, fully understood environment such as factories and warehouses. A factory robot may bring parts to pre-programmed destinations via a track in the floor, while a warehouse robot may recognize products and their destinations through barcodes and shipping databases. In contrast, construction or military robots could be brought to new locations about which they know very little, yet be expected to carry out tasks accurately and safely with minimal supervision. In taking on the role of an assistant or task runner, a robot acts like a human assistant or intern. Even the most inexperienced intern is expected to possess some knowledge about their work, rapidly learn to perform their job, and operate with minimal supervision. We judge our robot assistant with similar criteria: it must make good decisions as soon as possible, learn from experience, respond to feedback when given, and only rely on feedback as much as necessary to perform its duties.

In this research, we narrow the focus of a hypothetical robot assistant to a simple domestic environment. Consider a domestic robot tasked with putting a book away: it may reason that a bookshelf is a likely destination for a book, then navigate to a bookshelf in its environment. This seemingly simple scenario is fraught with complications. How does the robot make the connection between book and bookshelf in a way that can compel it to action? What if the robot chooses a table or desk instead, which despite being a reasonable inference is not what its operator wanted? What if the operator leaves inconsistent feedback?

To resolve these questions, we propose a framework based on a contextual bandit, a form of reinforcement learning commonly used in online recommender systems. In Section II, we review the state of robot human-in-the-loop reinforcement learning and mapping techniques relevant to this research. In Section III, we describe our reinforcement learning framework in detail. In Section IV, we conduct experiments to learn destinations for a sequence of items in accordance with a simulated human operator’s preferences. We establish a baseline $\epsilon$-greedy case when the human operator always leaves feedback, achieving approximately 75% accuracy, with correct decisions overtaking incorrect decisions after approximately 522 episodes. We show that selectively learning from no-response episodes improves performance such that infrequent feedback outperforms sessions when feedback is left every episode. Despite these improvements, we show that applying the alternate action selection method LinUCB outperforms $\epsilon$-greedy. Finally, in Section V, we summarize our results and discuss avenues of future research.

II. LITERATURE REVIEW

A key component of this work involves formulating a human-in-the-loop reinforcement learning system as a contextual bandit, though we are not the first to consider this
approach in a robotics context. Eppner [5] uses a contextual bandit in a robot arm to determine the optimal grasping strategy for a variety of items. Mendoza [8] places a contextual bandit in a robot soccer setting, where a group of robots select a free kick plan based on conditions on the field. Finally, Abel [1] uses a contextual bandit to find the optimal orientation of a solar collector given a variety of environmental factors. Unlike this research, these works learn from programmatic reward signals generated from environmental factors instead of human feedback.

Since humans are unable to leave feedback with the precision and consistency of programmatic reward signals, the problem of unreliable feedback is studied in both a robotics and contextual bandit setting. Because contextual bandits enjoy widespread use in Web content recommendation systems and advertisement selection [18], research in this space grapples with far worse feedback situations than this work. While we only consider the absence of feedback, Web site visitors often leave infrequent, unreliable, and implicit feedback. For example, clicks are a common method of generating reward signals, but are considered implicit feedback because they may not reflect the user’s true evaluation of system performance. Qi [12] considers the user’s decision-making process for whether to click on a recommended link as a latent variable and attempts to model it. While a lack of feedback is often translated to a negative reward signal, Zhang [20] attempts to determine whether sparse feedback is due to a dislike of recommended content or simply a lack of interaction. In doing so, they learn from positive and true negative feedback while ignoring true non-responses.

Outside of contextual bandits, human-in-the-loop reinforcement learning in robotics also grapples with unreliable human feedback. For example, Munzer [9] describes a method for learning human preferences during the collaborative assembly of a toolbox. The robot learns the human’s preferred approach to assembling the toolbox, allowing it to perform supportive tasks. Notably, they found that the human can neglect to correct wrong actions as much as 40% of the time without impacting the robot’s ability to learn human preferences. Arakawa [2] describes a system for receiving real-time unreliable rewards from a human operator in a Deep-Q reinforcement learning architecture. Like this work, they consider infrequent feedback, testing with feedback left 80% and 20% of the time. Additionally, they study incorrect and delayed feedback. Their method was demonstrated to be resilient against all but the worst feedback situations in both a simulated maze environment (where a goal state must be reached) and a taxi environment (where a passenger must be picked up and taken to a goal state). Schiatti [15] takes human feedback a step further, attempting to interpret EEG readings from human volunteers. During a training step, when volunteers observed simulated robots making poor decisions, their brain activity was collected and used to train a classifier. During testing sessions, the classifier was used on live brainwave activity to update a reward function. Even with relatively poor classification accuracy (60%), rewards from brainwave activity outperformed baseline reward signals in classic Q-learning.

In addition to a core reinforcement learning formulation with human feedback, this work relies on the ability of a robot not only to map, but to understand its surroundings. Specifically, our simulations assume that the robot has already performed some exploration and understands not only the shape of its environment and the location of obstacles, but also what objects exist in the environment and where they are. While robot localization and discovery, known as Simultaneous Localization and Mapping (SLAM), is generally considered a solved problem at a theoretical and conceptual level [4], the product of this process is a simple map containing only location and obstacle information. Enhancing this map with semantic information which can be incorporated into decision-making processes is an area of ongoing research.

A semantic map is a qualitative description of a robot’s surroundings. Depending on the nature of semantic features, they can be used in augmenting navigation and task planning while providing a common ground in human-robot interaction [6]. Posada [11] equips a robot with an omnidirectional overhead camera and trains a classifier to recognize area types such as “corridor”, “doorway” and “room.” These labels are applied while constructing maps with traditional SLAM methods. Zhao [21] integrates several different types of semantic labels, including detected object geometry, appearance, categories, and room classification into a dense 3D semantic map overlaid on a 3D SLAM map. Zheng [19] attempts to simultaneously label and localize objects within a scene, using particle filtering to maintain temporal consistency of the detected objects and their locations. Borkowski [3] describes a hybrid approach, combining detected objects from onboard scanners with structural and navigational information collected from existing architecture information such as Computer-Aided Design data and building description files.

III. Method

In this work, we examine the reasoning a general-purpose autonomous robot might carry out to accomplish tasks given by a human operator. Depending on the setting, a task could be anything, including delivering a message, restocking a store shelf, or repairing a broken item. However, for this research, we set aside the complexities of completing an open-ended set of tasks. Instead, we explore the initial stage of reasoning a general-purpose robot might perform when given a task: determining where to perform it. The best place to accomplish a task depends not only on the nature of the task and the properties of associated items, but also on the preferences of
the human operator. To further simplify the problem, we strip the robot down to a putting-away or room-tidying robot, whose only job is to accept an item given by a human operator and find an appropriate place to put it.

A session with the room-tidying robot takes place in a messy room filled with many items which must be put away. A human operator knows where each item should go, but the robot does not. An episode is depicted in Figure 1, and consists of the three basic steps in reinforcement learning problems: at time $t$, an agent (the room-tidying robot) receives a state (an item), chooses an action (a destination), then receives a reward (the human operator indicating whether it chose correctly). To learn the operator’s preferences, we employ a form of reinforcement learning commonly used in online recommender systems called a contextual bandit. Contextual bandits are an extension of stateless $k$-armed bandits, which attempt to learn which out of $k$ possible actions yields the highest reward to maximize cumulative rewards. Contextual bandits incorporate additional state information, expanding action rewards such that the bandit must learn which out of $k$ possible actions yields the highest reward for a particular state [17].

### A. State and action space

Our room-tidying robot operates in a simulated domestic environment containing common household objects, such as the one-bedroom apartment map shown in Figure 2. Each label in the map represents some object the robot has found in an earlier exploration step or through a priori knowledge. To clean up, we give the robot a sequence of household items we wish to put away, such as books, clothing, and electronics. Each object in the map represents a possible destination where an item can be brought. Formally, the environment contains a set of items $I = \{i_1, i_2, \ldots, i_n\}$, while the state space contains a set of items $S = \{s_1, s_2, \ldots, s_j\}$. Each $f \in F$ is a physical object or architectural feature like the kind seen in Figure 2. While $F$ could be an exhaustive set of every possible feature in the environment, including minutiae such as walls, flooring, and doorways, for practical reasons we restrict $F$ to only include environment features potentially involved with task completion. Additionally, we do not include obstacles in $F$ because we consider navigation to be solvable through traditional methods of path planning.

To transform detected physical objects into a real-valued state and action space, we introduce functions $F_S(f) = (e_F^s, \theta_F^s)$ and $F_z(i) = (e_F^z, \theta_F^z)$, which map objects $f \in F$ and $i \in I$ to some unique labels $e_F^s$ and $e_F^z$, and to parameters $\theta_F^s$ and $\theta_F^z$. Though not considered in this work, $\theta_F^s$ consists of location data associated with room feature $f$ as determined through a combination of SLAM and object detection with localization, using tools readily available in robot development libraries such as ROS [13], image recognition methods like YOLO [14], or other techniques explored in Section II. Any future work performing real-world experiments within this framework should determine $\theta_F^s$ for all $f \in F$ and exploit it during navigation. In contrast, we imagine $\theta_F^z$ to be open-ended, consisting of any interesting properties of item $i \in I$ which can be detected through sensors, visual recognition, external databases or even verbal commands given by the human operator. We note that properties need not be uniform across all items. Indeed, we expect a variety of properties to be collected for each type of item, with an example shown in Figure 3. In this example, our hypothetical room-tidying robot can weigh both “shirt” and “book” item on a built-in scale, but can only determine genre for the book (presumably through accessing a UPC database or through some scan of the cover) and cleanliness for the shirt (through some hypothetical laundry cleanliness classifier). Regardless of the method, we wish to incorporate many properties in $\theta_i$ so that different types of items can be better distinguished during training.

With the mechanics of the state and observation space fully described, we can formally define state space $S$ and action space $A$ as the set of all label/parameter pairs derived from the mapping functions introduced above:

$$S = \bigcup_{i \in I} \{F_z(i)\} \quad A = \bigcup_{f \in F} \{F_S(f)\}$$

### B. Reinforcement learning problem formulation

When contextual bandits are deployed in recommender systems, the context frequently consists of information about a Web site visitor, while actions frequently consist of advertisements which can be shown to the user or articles which can be recommended [22]. The reward consists of signals like clickthrough rate (how frequently a user clicks through to read a recommended article) [7]. Our problem is comparatively simple: instead of users, we have items for the robot to put away; instead of articles, we have destinations for the robot to bring the item to; and instead of clickthrough rate, we have a binary approval or disapproval of the robot’s decision. The robot’s job is to determine the value $q_s(s, a)$ of a state $s \in S$
and action $a \in A$, which is equivalent to the expected reward $R_t$ at time $t$ given a state $S_t$ and action $A_t$:

$$q_*(s, a) \doteq \mathbb{E}[R_t | A_t = a, S_t = s]$$

(2)

Of course, the room-tidying robot does not know which action is best for a given state, so it must estimate based on prior experience. The value estimate for a state $S_t$ and action $A_t$ at time $t$ is denoted by $\hat{q}(S_t, A_t, w)$, where $w$ is a weight vector. With this estimate, we can greedily select the estimated highest-valued action $A^*_t$ by

$$A^*_t = \arg\max_a \hat{q}(s, a, w)$$

(3)

But how do we compute $\hat{q}$? Estimating a value can be accomplished through a variety of methods, including linear regression, complex neural network architectures, and even tabular methods if the state space is discrete. For this work, we consider a linear estimator due to its favorable convergence properties, the relative simplicity and computational efficiency over neural networks, and a continuous state space. Specifically, we adopt two linear approaches to value estimation and action selection: $\epsilon$-greedy with linear regression and LinUCB.

For $\epsilon$-greedy action selection, we treat $\hat{q}$ as a linear regression problem, and find the inner product between $w$ and the real-valued vector representation $x(s, a)$ of state $s \in S$ and possible destination $a \in A$:

$$\hat{q}(s, a, w) = w^T x(s, a) = \sum_{i=1}^d w_i x_i(s, a)$$

(4)

where $d$ is the number of weights in the weight vector $w$, $w_i$ is the $i$th element of $w$, and $x_i(s, a)$ is the $i$th element of $x(s, a)$ [17]. To choose an action, we assign a probability $\epsilon/(k - 1)$ to all actions except the highest-valued action $\arg\max_a \hat{q}(s, a, w)$, which receives a probability of $1 - \epsilon$. We choose an action $A^*_t$ from this probability distribution, receive a reward $R_t$, and update the weight vector with

$$w_{t+1} = w_t + \alpha[R_t - \hat{\epsilon}(S_t, A_t, w_t)]x(S_t, A_t)$$

(5)

where $\hat{\epsilon}(S_t, A_t, w_t)$ is the value estimate at time $t$ and $\alpha$ is the learning rate.

In contrast with $\epsilon$-greedy action selection, which chooses exploration targets in a random, undirected manner, LinUCB is an Upper Confidence Bound (UCB) approach which optimistically selects actions which seem best given prior experience [7]. Intuitively, the algorithm chooses destinations it has low confidence in, so it can explore them and improve its value estimates. As it accumulates experience, it becomes increasingly certain in its estimates and selects actions in an increasingly greedy manner. Like the simple method introduced above for $\epsilon$-greedy action selection, LinUCB is linear. However, it is different in several important aspects: first, it employs disjoint linear models, such that there is one linear model for each action $k$. Additionally, it estimates not only a mean payoff for each action, but a confidence interval for that estimate. To choose an action with LinUCB, we employ a different method of value estimation:

$$\hat{q}_{\text{LinUCB}}(s, a) = x(s)^T \hat{\theta}_a + \alpha \sqrt{x(s)^T A_a^{-1} x(s)}$$

(6)

where $x(s)$ is the real-valued vector representation of state $s$, $\hat{\theta}_a$ is a weight vector for action $a$, and $\alpha$ is the exploration coefficient. $A_a$ is given by

$$A_a = D_a^T D_a + I_d$$

(7)

where $D_a$ is all accumulated context vectors $x_{t,a}$ for action $a$. $I_d$ is a $d \times d$ identity matrix, where $d$ is the number of features in feature vector $D_a$. Like $\epsilon$-greedy action selection in Equation 3, we select the highest-valued action $A^*_t$:

$$A^*_t = \arg\max_a [\hat{q}_{\text{LinUCB}}(S_t, a)]$$

(8)

To update weights $\hat{\theta}_a$, we employ ridge regression such that

$$\hat{\theta}_a = (D_a^T D_a + I_d)^{-1} D_a^T c_a$$

(9)

where $c_a$ is the vector of accumulated rewards corresponding with accumulated context vectors $D_a$.

IV. EXPERIMENTS & RESULTS

A. Dataset

To produce a dataset for each session, we draw from a pool of household items $I$ and a pool of destination furniture $F$. We select items for inclusion in the pools based on two criteria: they must be common household items, and we must have some logical justification for choosing them. For example, in choosing “shirt” for $I$, we select counterparts “dresser” and “washing machine” for $F$, but not nonsensical locations such as “fishtank” or “dishwasher.” In seeking to simulate a real session with a physical robot, we limit the number of items and destinations to an amount that might be collected for a real-world testing session. $I$ contains 12 items and $F$ contains 19 destinations, resulting in a total of 228 possible combinations. The complete set of destinations are bookcase, table, coffee table, desk, bed, sink, bathtub, toolbox, refrigerator, toy chest, piano, record player, stove, washing machine, cupboard, couch, nightstand, dresser, and closet.

To simulate human preferences, we grant each item $i \in I$ a set of $c_i$ characteristics $\theta_i = \{\theta_{i1}^1, \theta_{i2}^2, \ldots, \theta_{ic}^c\}$. Each $\theta \in \theta_i$ consists of a set of properties appropriate for the item as described in Section III-A. Properties consist of genre for media items, cleanliness for clothing-related items and weight for all items. Genre labels include {n/a, art, classic, sci-fi, textbook, coffee table book}, cleanliness includes {n/a, clean, dirty}, and weight is sampled from a distribution unique to each $\theta$. Each item/characteristic pair $(i, \theta) \in I \times \theta_i$ is assigned to a set of $d_i$ possible destinations from the destination pool $F$ such that

$$(i, \theta) \rightarrow \{f_{i1}, f_{i2}, \ldots, f_{id_i}\}$$

(10)

At the beginning of each session, we randomly select a destination for each $(i, \theta)$ pair, and consider this to be the
human’s preference. Available destinations and combinations of $\theta$ for each item is shown in Table I, where we have many variations on one item (a book), each of which can go to several possible destinations; one or two variations of several items, each of which can go to a few possible destinations; and one destination (not shown in the chart above) with which no items are associated. Of the 228 possible combinations of item/destination pairs, only 30 are utilized for any session. Once a set of item/destination pairings are established for each session, we generate a series of 2000 episodes, each with a randomly selected item/destination pair representing an item given to the room-tidying bot with a hidden destination reflective of the human operator’s preference. The bandit’s job is to determine the human operator’s preferred destination given an item.

B. Experiments

1) $\epsilon$-greedy baseline: We perform a set of experiments with an $\epsilon$-greedy action selection policy, learning an action value estimate $\hat{q}$ with the support vector machine module supplied with SKLearn [10]. We simulate the human operator leaving feedback 100%, 75%, 50%, and 25% of the time, represented by the variable $p$. If the operator does not leave feedback, we do not learn from the episode. During early experimentation, we found that a descending $\epsilon$ policy yielded good results, so we begin with $\epsilon = 0.3$ and descend at a rate of 0.001 per episode until reaching $\epsilon = 0.1$. Results are shown in Figure 4, showing mean prediction accuracy and hypothetical cumulative rewards, or cumulative rewards had the human left feedback after every episode. The hypothetical cumulative reward is marked at its lowest point, or the turning point, highlighting the specific episode where the robot begins to perform correct actions more frequently than incorrect actions. The outcome is straightforward: Because the robot ignores no-response feedback, low-feedback simulations learn with fewer episodes than they actually encounter. Simulations where feedback is given more often achieve higher prediction accuracy. Additionally, the less frequently feedback is given, the longer it takes for the robot to perform mostly correct actions. Simulations where $p = 1.0$ and $p = 0.75$ reach their turning points at episodes 522 and 581. In contrast, $p = 0.5$ does not reach this point until episode 985, and $p = 0.25$ does not reach its turning point within any of the 2000 episodes.

2) $\epsilon$-greedy with no-response feedback: Because we ignore no-response feedback, the cause of poor performance in low-feedback situations is likely a lack of training examples. As an alternative to the baseline approach, we perform the same experiment but train on no-response feedback with a neutral reward signal of 0. While it may seem counterintuitive to reward the robot for the inaction of its handler, we see in Figure 5 that this simple change causes an improvement in prediction accuracy for lower values of $p$. $p = 0.25$ has improved to a 0.615, $p = 0.5$ has approximately matched the performance of $p = 1.0$, and $p = 0.75$ has exceeded it at 0.77. In the corresponding hypothetical cumulative reward analysis, we see an even more dramatic improvement: correct actions begin to overtake incorrect actions much sooner than before, with $p = 0.75$ surpassing $p = 1.0$ at episode 418. $p = 0.5$ has improved to a turning point of episode 658, and $p = 0.25$ now reaches its turning point within the 2000 episodes at episode 1253.

3) $\epsilon$-greedy with replacement of no-response feedback: Despite having increased performance by learning from no-response feedback, we believe there is room for further improvement. We suspect that, while no-response feedback

<table>
<thead>
<tr>
<th>Item (s)</th>
<th>Possible Destinations ($d_i$)</th>
<th>Possible $\theta$ ($c_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>Mug</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Pen</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Laptop</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Phone</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Pillow</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Sock</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Shirt</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Jeans</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Jacket</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Sheet Music</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hammer</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4. $\epsilon$-greedy baseline. Above: Accuracy of 25 sessions with 50-episode rolling average, std. deviation in gray. Below: Mean cumulative reward as though human left feedback after every episode. The lowest point is marked, where robot begins making more correct than incorrect decisions. Bars show std. deviation of lowest point position.
encourages early exploration, it becomes less useful over time. Additionally, in low-feedback situations where \( p < 0.5 \), more than half of all accumulated experience is no-response feedback. In an attempt to correct this remaining convergence problem, we note that the best performance was achieved accumulating no-response feedback when \( p = 0.75 \). With this in mind, we ran a series of experiments maintaining a history of previous episodes. We compare new episode contexts with accumulated episode contexts, replacing similar old no-feedback response episodes with new episodes. Furthermore, we limit the frequency of replacements in an attempt to maintain a ratio of 75% explicit feedback to 25% no-response feedback, regardless of the value of \( p \). At time \( t \), the number of episodes to be replaced \( n_{r,t} \) is given by

\[
n_{r,t} = \lceil \tau (n_{a,t} - n_{i,t}) \rceil
\]  

where \( n_{i,t} \) is the number of no-response feedback episodes needed to maintain the desired ratio at time \( t \), \( n_{a,t} \) is the number of no-feedback episodes in \( D_t \) at time \( t \), and \( \tau \) is some constant to limit the number of episodes removed at once. Experimentation found \( \tau = 0.05 \) to produce good results.

Historical episodes are selected for removal through random sampling, selecting \( n_{r,t} \) episodes with probabilities

\[
P = \text{softmax} \left[ \bigcup_{o \in D_{0,t}} -d(o_t, o) \right]
\]  

where \( D_{0,t} \) is the set of accumulated episodes which received no-response feedback at time \( t \), and \( d(o_t, o) \) is the Euclidean distance between current episode \( o_t \) and historical episode \( o \). Once replacement is complete, the SVM is retrained with the smaller history. In early experimentation, we attempted other similarity metrics, some which amplified the distance between item types. However, we found little difference between results with other metrics and negative Euclidean distance, so we use it here for simplicity.

The accuracy of this pruning approach is shown in Figure 6, with significant gains to \( p = 0.25 \), bringing it to an accuracy of 0.7. Additionally, we see small gains to \( p = 0.5 \), which now slightly surpasses \( p = 1.0 \) at 0.81. \( p = 0.75 \) has improved to approximately 0.79. Hypothetical cumulative rewards show corresponding gains: The turning point of \( p = 0.75 \) has remained relatively unchanged at 429, \( p = 0.5 \) has improved from 658 to 454, and \( p = 0.25 \) from 1253 to 797. These modifications show that it is possible to maintain good performance even in low-feedback situations. We believe that maintaining a limited ratio of no-response feedback to explicit responses roughly translates to “I’m not sure about this but it could be right,” and is ultimately beneficial for learning with an \( \epsilon \)-greedy strategy.

4) LinUCB: We evaluate LinUCB as an alternate value estimation and action selection approach with the same 3 methods introduced in previous experiments: ignoring no-response feedback, learning from both explicit and no-response feed-
back, and maintaining a fixed ratio of explicit feedback to no-
response feedback by throwing out irrelevant episodes. The
LinUCB exploration constant was set to $\alpha = 0.5$, since ex-
perimentation determined this to produce good results. Results
are shown in Figures 7, 8, and 9. We see that LinUCB signif-
icantly outperforms $\epsilon$-greedy action selection in overall accu-
racy and in the number of episodes before the turning point.
Unlike $\epsilon$-greedy action selection, LinUCB actually performs
worse with accumulated no-response feedback than without.
Introducing replacement like we did for $\epsilon$-greedy action selec-
tion did little to improve matters, at best bringing performance
and early-episode performance back to approximately where it
was before. That LinUCB outperforms $\epsilon$-greedy is no surprise:
the performance difference is a result of each algorithm’s
approach to exploration. $\epsilon$-greedy approaches are unguided
because they explore possible destinations uniformly. UCB
approaches are guided, choosing exploration targets based on
their uncertainty in each destination’s value estimate. While
no-response feedback helped to direct the $\epsilon$-greedy policy, it
is a distraction for LinUCB, which already gravitates towards
actions it has not explored.

V. CONCLUSIONS & FUTURE RESEARCH
In this work, we described a reinforcement learning ap-
proach to general-purpose robot task completion in a simulated
domestic environment, where a room-tidying robot learns a
human’s preferences for where different types of items should
be taken. We demonstrated how a contextual bandit approach using $\epsilon$-greedy or LinUCB action selection can successfully learn human preferences, even if the human does not leave feedback in every episode. We showed that learning from episodes where the human does not leave feedback and manipulating the history of accumulated episode data results in better performance with an $\epsilon$-greedy policy, with feedback left 75% and 50% of the time outperforming sessions where the human left feedback every time. Despite these improvements, we showed that LinUCB significantly outperforms $\epsilon$-greedy methods, reaching approximately 94% accuracy.

There are many opportunities for continued research. First, despite the performance we observed through experimentation, these results exceed what we expect from small-scale real-world tests. Even LinUCB does not converge to its maximum accuracy for hundreds of episodes, and takes over 100 episodes to begin making more correct decisions than incorrect. Consequently, future research will focus on improving early-episode accuracy. Without sufficient human feedback to reliably determine item destinations, we believe that inferring them through commonsense reasoning will dramatically improve early-episode performance. Large-scale commonsense knowledge bases like ConceptNet [16] are a promising source of a priori information, and we intend to integrate it as a component of action selection in the future.

Second, we consider a major open research area to be additional robot actions once it has chosen a destination. A true general-purpose robot must be equipped to carry out a variety of actions with different items in many locations. In this work, we limit the scope of the robot to deciding on a location where a future unspecified action will be performed. While we see the selection of this future action as a separate issue outside the scope of this work, in the absence of explicit human feedback, the reward signal could be augmented to incorporate the success or failure of this later action.

Finally, we assume that the robot has already performed some preliminary exploration and has discovered possible destinations already. We do not address the problem of a robot starting with no awareness of its surroundings. In this situation, the robot must balance the exploration of its environment with the exploitation of destinations it has already found. We hypothesize that a “go exploring” action could exist as an augmentation to already-discovered item destinations. Additionally, the robot could conduct ongoing image recognition as it performs its tasks, discovering environment features simply by traveling. Finally, pathfinding algorithms could choose deliberately indirect paths, preferring to take the robot through unexplored areas as it navigates to its chosen destination.

REFERENCES


