Int J Soc Robot (2012) 4:319–330
DOI 10.1007/s12369-012-0158-7

A Practical Comparison of Three Robot Learning from Demonstration Algorithm

Halit Bener Suay · Russell Toris · Sonia Chernova

Accepted: 13 June 2012 / Published online: 28 June 2012

Abstract Research on robot Learning from Demonstration has seen significant growth in recent years, but the field has had only limited evaluation of existing algorithms with respect to algorithm usability by naïve users. In this article we present findings from a user study in which we asked non-expert users to use and evaluate three different robot Learning from Demonstration algorithms. The three algorithms selected—Behavior Networks, Interactive Reinforcement Learning, and Confidence Based Autonomy—utilize distinctly different policy learning and demonstration approaches, enabling us to examine a broad spectrum of the field. Participants in the study were asked to teach a simple task to a small humanoid robot in a real world domain. they controlled the robot directly (teleoperation and guidance) instead of providing retroactive feedback for past actions (reward and correction). We present our quantitative findings about: (a) the correlation between the number of user-agent interactions and the performance of the agent and (b) the correlation between agent's final performance and its perceived accuracy by the participant. Comparatively, the strongest correlation was found in CBA data. We also discuss the possible reasons of our qualitative results. Additionally, we identify common trends and misconceptions that arise when non-experts are asked to use these algorithms, with the aim of informing future Learning from Demonstration approaches. Our results show that users achieved better performance in teaching the task using the CBA algorithm, whereas the Interactive Reinforcement Learning algorithm modeled user behavior most accurately.

Keywords Learning from demonstration · Human-robot interaction · Robotics

H.B. Suay (✉) · R. Toris · S. Chernova
Worcester Polytechnic Institute, 100 Institute Dr., Worcester, MA 01609, USA
e-mail: benersuay@wpi.edu

R. Toris
e-mail: rctoris@wpi.edu

S. Chernova
e-mail: soniac@wpi.edu

1 Introduction

Algorithms for robot *Learning from Demonstration (LfD)* seek to enable human users to expand the capabilities of robots through interactive teaching instead of explicit programming. Specifically, LfD methods enable new robot behaviors to be learned based on demonstrations of the desired actions performed by the teacher. This research field is inspired by naturally occurring learning paradigms in humans and other animals, and the broad research goal is to develop an intuitive method of programming that is accessible to untrained, naïve, users.

Dozens of variants of learning from demonstration algorithms have been proposed in the research literature to date, and two recent surveys cover the breadth of this field [4, 7]. Several user studies have been performed, studying different aspects of human-robot interaction [16, 27], and different aspects of algorithms that authors developed themselves [2, 24]. Despite the existence of such a diverse body of work, research in this area lacks a comparative user study utilizing a common domain which focuses on the performance and preferences of naïve users. In previous studies, policy performance is typically the focus of the evaluation [2, 24], and often learning time is compared against non-interactive methods, such as Reinforcement Learning [12, 23, 25]. However, we are not aware of any user studies that have examined the usability of LfD methods on a common test domain with naïve test subjects.

We note that this fact is not due to the lack of academic rigor in LfD research, but instead highlights some of the challenges of this research problem. The use of different robotic platforms, interfaces, and demonstration techniques leads to different representations and characteristics of demonstration data that makes comparisons difficult across applications. The use of standard data sets, which is common in other research fields, is not possible due to the human interaction component and the diversity of interaction techniques. As a result, comparison requires full reimplementation of algorithms a highly time consuming and challenging effort due to the absence of existing open source solutions.

The lack of comparison leaves many critical questions unanswered. Without the comparison of existing methods, future researchers have no guidance with respect to what learning algorithms and interaction techniques are most effective, and what underlying algorithmic assumptions accurately reflect the real-world use case. These problems are further heightened by the fact that many LfD techniques have only been evaluated through use of expert roboticists, typically the authors themselves [13, 14, 17, 22, 23]. As a result, the correctness of the underlying assumptions with respect to the type and quality of demonstrations, and the ease of use of interaction methods have not been evaluated. For example, human teachers often make noisy demonstrations to robots [13]. Since as computer scientists and roboticists we know the inhibiting effects such noise can have on learning algorithms, we may take great care in providing reasonably clear training data. This certainly cannot be expected of a naïve teacher.

Another dimension of the LfD problem is the perceived intelligence of the robot by its teacher. A human teacher's intention can generally be described as reprogramming the robot without coding, ultimately teaching it a new task. Throughout the training process, users might make broad assumptions about the robot. For example, a user might feel that the robot can see and understand everything the teacher can see. Or, a user might think that the robot can hear and comprehend any set of verbal commands given. Furthermore, a user might feel that if no immediate feedback is given by the robot, that robot must not have learned or understood the teacher.

While assumptions like the ones described above may seem insignificant, the resulting mismatch in expectations is likely to be detrimental to the success and performance of a LfD algorithm when used by the general public. If naïve teachers expect certain types of interactions and responses from the robot, then we as engineers and scientists must address these issues in our own work. It is only then that the success of LfD can progress in the population of non-expert users.

To further investigate the issues raised above, we present a user study of three established LfD algorithms, each representing a different variant of policy learning:

- Interactive Reinforcement Learning (Int-RL) [28]
- Behavior Networks (BNets) [20, 21]
- Confidence Based Autonomy (CBA) [13]

The central goals of our evaluation are to examine three well-established LfD algorithms and to (1) identify and analyze mismatches between core algorithmic assumptions and naïve user behavior, (2) examine user preferences for three distinct robot teaching interaction styles, and (3) compare algorithm usability and performance in a common domain.

To reach our goals, we applied all three algorithms to a single task and ran a user study with 31 participants. Each participant trained a robot using each of the three algorithms. In this work we report the qualitative and quantitative results we obtained. We examine not only the test subjects' performance with these methods, but also take a closer look at the preferences of the user and types of demonstrations they provide to each algorithm.

The remainder of the article is organized as follows. In Sect. 2, we discuss the field of learning from demonstration, and briefly describe the algorithms we implemented in our user study. In Sect. 3 we point out the reasons for selecting the three algorithms above. In Sect. 4 we explain our experimental setup and task domain. We then present our findings in Sects. 5 and 6. Finally we discuss the implications of our findings in Sect. 7.

2 Brief Description of Algorithms

Many variants of learning from demonstration algorithms have been proposed in the research literature to date [4, 7]. In one recent survey of the field [4], the authors present a categorization of existing algorithms in order to aid comparative assessments between applications. Algorithms are categorized by:

- *Demonstration Technique*: teleoperation [13, 14, 23], shadowing [19, 20], sensors on the teacher [3, 10, 15] or external observation [6, 11]; the categorization is dependent on who performs the task during demonstration (the human or robot) and what information about the demonstrator's actions is available.
- *Policy Derivation Method*: mapping functions (e.g., classification [13, 18, 30] and regression [14] techniques), system model (e.g., reinforcement learning [1, 5]) and planning [29] approaches.

The three algorithms we have chosen represent the three policy derivation (data use) categories shown in Fig. 1, and utilize different demonstration techniques. Specifically:

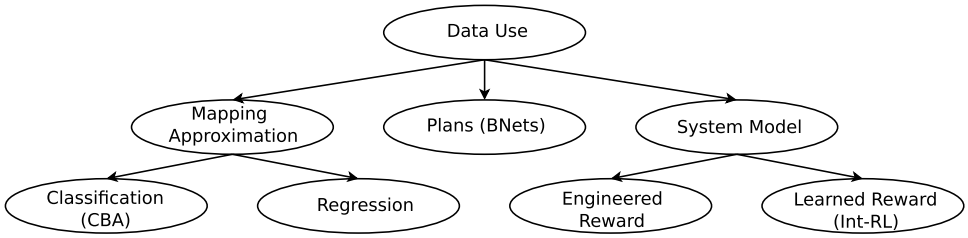


Fig. 1 Categorization of approaches to learning a policy from demonstration data. Figure adapted from [4]

- *Interactive RL*: A system model policy learning technique in which the reward signal is provided by the human user at runtime. The teacher interacts with the robot by providing reward and guidance input, in our case through an on-screen interface. Interactive Reinforcement Learning is designed in light of successive user studies and works based on positive or negative feedback for the preceding action and guidance for the following action of an agent. With feedback, the teacher can tell the agent whether the preceding action was the right or the wrong thing to do. With guidance, the teacher has control over narrowing down the action choices of the agent (i.e., limits the action space when choosing an action); however, the user cannot give a specific action command for the agent to execute.
- *Behavior Networks*: A planning-based policy learning technique. In our implementation the teacher interacts with the robot through kinesthetic teaching [10], a type of *experienced demonstration* [22] in which the user physically guides the robot through the task. Behavior Networks is a planning algorithm which is meant to be used in conjunction with teleoperation. A behavior can be any low level control or high level action. How a behavior gets activated depends on the preconditions defined for that specific behavior. The definition of each behavior depends on several factors such as the task, environmental conditions, implementation preferences, hardware limitations etc. We are interested in seeing how this flexibility (or loose definitions) will reflect on the test subjects' experience.
- *Confidence-Based Autonomy*: A mapping approximation technique based on a classification algorithm. The teacher interacts with the robot by selecting the next action or correcting a past action choice through an on-screen interface. Confidence Based Autonomy is an active learning method that is based on asking for help from a human teacher when needed. It is not only different from previous algorithms in the sense that the agent initiates the interaction with the teacher, but it also lists all available action choices to the user, allowing for robot teleoperation.

In our work we do not have a baseline control group. Instead of changing a specific variable and observing how user preference changes [9], we are interested in obtaining broader results. For example, instead of changing only the frequency of asking for help using a single learning technique, we give our test subjects completely different techniques, which are not variations of each other.

We note that the learning performance of Interactive Reinforcement Learning and Behavior Networks were recently compared on the same task in [24]. However, this work focused entirely on the computational component and was evaluated only with a single expert roboticist as the user, providing no insight into how well these techniques will perform in the general use case.

Finally, we refer readers to the recent survey papers, [4, 7], for additional information about the broader research area.

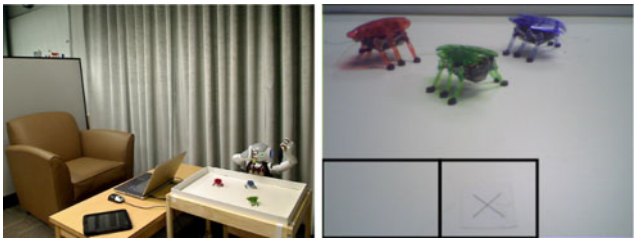
3 Selection of Algorithms

When selecting the algorithms mentioned above, we were motivated by several factors: the difference in task representation (e.g. state-action mapping versus sequential action planning), the difference in the learning process (e.g. optional feedback versus active learning) and general acceptance in the LfD community (each of the chosen algorithms has over 100 citations in the literature).

We believe that it is reasonable and fair to compare these methods on a common ground; the methods selected have elements that can possibly be attractive for different reasons. They can be thought as complementary elements or contradictory elements depending on the perspective. In this work, we have chosen to use these specific algorithms in order to study the implications, side effects, and benefits of the aspects mentioned above with naïve human teachers.

4 Experimental Setup

To evaluate the three algorithms, participants were recruited to teach a robotic agent to accomplish a simple task using all three methods. We recruited 31 participants from the WPI area, 21 male and 10 female, with ages ranging from 18 to 35. Six of the participants self-identified as an expert roboticist, while 14 were novice and 11 had no robotics background.



(a) (b)

Fig. 2 (**a**) Experimental setup, (**b**) view of the table from the Nao's on-board camera. Sweep zone: bottom-left portion of the figure. Pick--up zone: bottom-center portion of the figure. Any other portion of the image is Wait zone

Since training time can be quite lengthy in certain domains [24], we aimed to find a domain that was simple enough to see a progress in learning in a relatively short amount of time, yet dynamic and random enough to not be trivial. With this in mind, we designed the DBug domain for the Aldebaran Nao humanoid robot, which required the humanoid robot to catch and remove small HEXBUG robotic toys from a table.

4.1 The Study Setup

The experimental setup, depicted in Fig. 2a, consisted of the Nao's tabletop setup, a laptop used by the participant to interact with the robot, and a webcam used for recording the session.

The tabletop setup consisted of the Nao seated at a small table ($25'' \times 19''$). A red, blue, and green HEXBUG toy were placed on the table and allowed to move around. For each test, bugs were randomly put on the table. The test subject had the option of turning the HEXBUGs on and off, or moving them around at will. A short wall was added to the perimeter of the table to prevent the bugs from walking over the edge. By default HEXBUGs move forward until either of their antenna touches to an obstacle (i.e. our short wall at the perimeter of the table, another HEXBUG, the test subject's hand, or our humanoid robot's manipulator). When the toy bumps into an obstacle, it moves backwards while rotating for a short period of time. In this manner, the bugs continuously move around the table unless turned off by the user.

When learning, the Nao's camera was directed straight at the center of the table. The field of view covered the majority of the table with small portions on the left and right being out of sight. The test subjects were not told how much of the workspace was exactly covered by the camera. Using its camera, the Nao was able to continuously track the x and y position of all bugs in its field of view (as seen in Fig. 2b). We used OpenCV [8] for all image processing.

In order to assist the Nao in picking up the bug toys, a magnet was placed into its right hand. A pickup zone was marked with an $\times$ on the table in front of the Nao where it could reach down and attempt to attach a bug to the magnet (shown in Fig. 2b). Once a bug was attached to the magnet, the Nao raised its arm and asked the user to remove it manually. Additionally, a small foam brush was placed into the robot's left hand to help sweep bugs toward the pickup location. Both robot actions followed a precoded motion trajectory.

Participants used a laptop to start and stop the learning process and perform most of the interactions with the robot. A tablet computer was also provided, presenting written and pictorial instructions about how to use each training method. The task described was kept consistent across all three teaching methods.

4.2 The DBug Domain

The domain representation, $D = (S, A)$, was defined by a finite set of states S and actions A. Each state $s = (x, y) \in S$ was defined by the x and y coordinates of the HEXBUG in picture's coordinate system, considering the top left corner to be $(0, 0)$. Since there were more than one HEXBUG on the table most of the time, bug with the smallest Euclidean distance to the pickup location was used as the current state. If no bugs were detected by the robot, the state was taken as $(0, 0)$.

Using a 320 by 240 pixel image from the Nao, a total of 76,800 uniquely identifiable states were possible in the domain. The domain space was considered to be large enough such that the solution was not trivial yet small enough that a user could make reasonable progress teaching the agent within a short period of time. In the case of Int-RL, we subsampled the state space to 100 states. This composed of a 10×10 grid (a rectangle of 32 by 24 pixels per (x, y) state) overlayed over the image. This was to increase the probability of visiting the same state more than once during the training so that the agent can make progress in learning during the relatively short period of time of the study.

The action space A is defined by three actions: picking up a bug at the center pickup location using the magnet in its right hand, sweeping inwards towards the center pickup location using the foam brush in its left hand, and waiting for 1 second (i.e., doing nothing). All actions were preprogrammed. The pickup action took $\sim$9 seconds and the sweep action took $\sim$5 seconds from start to finish. The wait action was an important action to include as it was useful in teaching the robot when it was not appropriate to either sweep or pickup (e.g., when no bugs were present around the pickup location).

Sampled across the complete state space, approximately 12.5 % of the states fall into the sweep zone, 10.4 % fall into the pick-up zone, and 77.1 % fall into the wait zone (zones are shown in Fig. 2b).

4.3 Study Protocol

Each participant was allotted one hour to perform the study. The study followed the following process:

- Introduction
- Training Session 1 (10 minutes)
- Survey Questions for Algorithm 1
- Training Session 2 (10 minutes)
- Survey Questions for Algorithm 2
- Training Session 3 (10 minutes)
- Survey Questions for Algorithm 3
- General Survey

Upon arrival, the participant was given a broad overview of the task that was being trained as well as the capabilities of the Nao. It was noted that the Nao would be able to track the bugs with its own vision system and each of the three preprogrammed actions were also explained.

Following the introduction, the participant used each algorithm in turn to train the agent. Before beginning a trial, the participant was given verbal and written instructions about how the algorithm worked and allowed to ask any questions. Following each training session the participant was asked to fill out a brief questionnaire relating to the method that they had just used. The order of the algorithms was varied with participants, subdivided into three subgroups, based on the order in which they used the algorithms:

- Int-RL/BNets/CBA (10 participants),
- BNets/CBA/Int-RL (10 participants), and
- CBA/Int-RL/BNets (11 participants).

The training session for each algorithm was limited to 10 minutes. Participants were allowed to end training early, although none chose to do so. Once all three methods and questionnaires had been completed, a final general questionnaire was given which asked questions related to user preferences among the three methods.

5 Qualitative Results

In this section we present the results of the qualitative analysis of the user study and our general observations.

5.1 Algorithmic Assumptions

We begin by examining the assumptions that the algorithms make and how naïve user behavior differed from those assumptions.

The Interactive Reinforcement algorithm makes the following assumptions:

1. A positive feedback will follow most correct actions,
2. A negative feedback will follow most incorrect actions.
3. Guidance will be given in order to narrow down the choice of the action it precedes.

This approach was very straightforward for the participants and we feel that all of the assumptions listed above held when users taught the task. We believe that the design process of the Int-RL algorithm has a big influence on that finding. Int-RL was originally designed alongside user studies with an online agent [26, 28], and, with this study, we can confirm that with a physical robot, the users interacted in a similar way compared to the online agent.

The Behavior Network algorithm makes the following assumptions:

1. The teacher will make a sequence of demonstrations for each task, beginning each demonstration with START, performing one or more actions through kinesthetic teaching, and ending the demonstration with DONE.
2. If more than one demonstration sequence is performed for a given task (i.e. multiple START to DONE sequences), the demonstrations will result in complementary dependencies in the model, with no contradictory actions dependent on the same set of preconditions.

In our implementation of BNets, for the agent to understand that there was a HEXBUG to pick-up, we used a "found a bug in pick-up zone" behavior. Likewise, for understanding that there was a bug in sweep zone we used a "found a bug in sweep zone" behavior. The precondition for these behaviors to be activated was to detect a HEXBUG in the pick-up or sweep zone. Ideally these behaviors should have been the precondition for a second behavior, either "pick-up the bug" or "sweep". As stated in assumptions (1) and (2), the algorithm expects the teacher to demonstrate the task in the correct sequence. For instance, after a demonstration of "found a bug" (either by putting a HEXBUG in front of the camera manually, or by waiting for a HEXBUG to arrive randomly), a demonstration of "pick-up the bug" (by lowering the right arm of the robot and lifting it back up) would create a correct behavior network for picking up a HEXBUG upon detection.

We observed that the dynamic nature of the environment created a particular challenge for non-expert teachers using BNets. The pre-conditions for "found a bug in pick-up zone" or "found a bug in sweep zone" behaviors were often met unintentionally if the user did not control the location of the HEXBUGs (e.g., by turning them off and displacing them manually). This violated the first assumption. It was hard for most users to realize which behavior was activated when, because there was no visualization for the mental model of the agent.

Users were given the option of deleting the whole network and re-teaching the task, or keeping the current knowledge and adding new demonstrations on top of the existing

network. Although users tried to erase and re-teach the task multiple times, none of them actually succeeded to teach the sweep action in the correct states. On the other hand an expert roboticist who knew about the necessary order of preconditions, could teach the task with only two demonstrations (one for pick-up and one of sweep) by controlling the location of HEXBUGs manually. Almost every study participant had an unintended behavior dependency. An example of an unintended dependency is executing a pickup action after every sweep, even if there is no bug in the pickup zone. Similar unintended dependencies were taught by every user, in many occasions repeatedly. The user intention conformed to assumption (2). In general the users tried to demonstrate behaviors for the ultimate goal of sweeping and picking-up HEXBUGs as opposed to series of some random and conflicting series actions.

Confidence Based Autonomy with GMMs, makes the following assumptions:

1. The user will provide training data with multiple demonstrations for each action, effectively sampling the state space.
2. The user performs a correction for an incorrectly selected action during the time interval between the beginning of the incorrect action, and the beginning of the next action.

In our domain, each action had one area of the state space where the agent should have learned to execute it. With an ideal set of demonstrations the agent would have 3–4 Gaussian models in total (e.g., one model per action centered in that specific zone). We observed that the demonstrations given by the participants were very scattered and often resulted in multiple Gaussians for each action. Instead of being grouped around one center per action, Gaussian models were scattered in the state space. This decreased the efficiency of decision making and accuracy of the model. Samples were often given either too sparse to be in the same Gaussian model, or different models were overlayed due to different action labels given in very similar states. Furthermore, three users violated assumption (1) by not providing enough training data for the classifier to confidently make any predictions (minimum of 3 demonstrations of each action). Several users also violated assumption (2) by not providing corrections within the required timeframe. The delay between the transition time of the real world (e.g., $time = t$) and the state that the users wanted to give correction for (e.g., $time = k < t$) led to a mismatch between states and labels, corrupting some of the training data.

5.2 Usability

For Int-RL, in terms of usability, we observed that 10 minutes was not sufficient to enable inexperienced users to train the task well (task performance is further evaluated in Sect. 6.1). However, we note that the authors of this paper, who are familiar with the inner workings of the algorithms, were able to teach the task using all three methods within this time frame. This highlights the difference between experts and inexperienced users. Many users could not see an improvement in learning until the end of the experiment, which led some to stop rewarding or to vary their reward strategy to try and achieve a better response.

In BNets, we received positive comments about the ease of kinesthetic teaching; however, users often tried to teach the task by making new, yet incorrect, demonstrations to the agent. Even when they erased the learned network and restarted teaching, instead of adding behaviors one by one (such as teaching picking-up the HEXBUG in one demonstration, then teaching sweeping the HEXBUG in a second demonstration) they moved both arms at the same time thus creating simultaneous actions. Since the environment was dynamic, it caused learned networks to be overly complicated.

Many users were able to achieve relatively high performance using the CBA algorithm. Due to the dynamic nature of the DBug domain, the environment did not pause when the robot requested help, as in previous applications of CBA [13]. The robot was in a constant cycle of "perception–decision–execution" and the decision process (with added small wait commands to give people a time to do corrections) was perceived as lag by our participants. This is a practical issue. The study of inexperienced users across dynamic domains and ones in which the state does not change while the robot asks for help on teaching could be an interesting future work.

6 Quantitative Results

To evaluate user performance at teaching the task, we examine the final policy of each algorithm at the end of the 10 minute training session and evaluate its performance in selecting the correct action for each state. For each method, we check if the agent executes the correct action at a given zone (shown in Fig. 2b). The correct action label for each state was pre-determined by the region in which that state was located.

Figure 3 compares the percentage of correctly taught state-action pairs for each algorithm. On average, our subjects taught the pick-up action (the left three plots) and the sweep action (the middle three plots) with highest accuracy using CBA. The second most accurate result was achieved with Int-RL for these two actions. However subjects taught the wait action with higher accuracy using BNets. It is worth noting that the wait action was implicitly taught in BNets, that is, if the teacher does not demonstrate any action by moving the arms of the robot (for instance when there is no

bug in pick-up zone), then the agent learns to not-to-act, i.e. to wait. Based on the lower accuracy rate of the other two actions, we think that this implicit definition is the reason why people had more success with BNets for this specific action.

In the rest of this section we evaluate algorithm-specific performance statistics. We are specifically interested in two data patterns: a) the correlation between the number of user-agent interactions and the accuracy of the final policy, and b) the correlation between the accuracy of agent's final policy and the user's satisfaction with the agent's performance (i.e. perceived accuracy). We use the Pearson correlation coefficient to report our findings. For each algorithm we also give a visual example of one successful and one unsuccessful final policy (or plan) in order to highlight the difference.

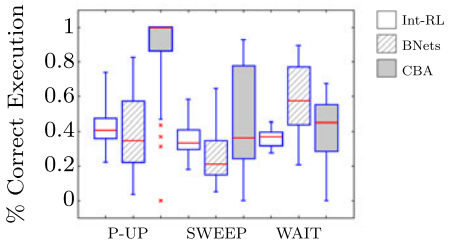
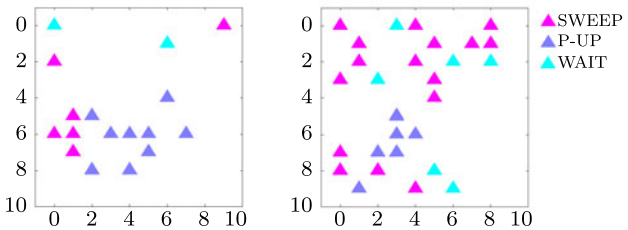


Fig. 3 Percentage of correctly executed actions in all three of the methods. The *red bars* in each box show median values. The whiskers cover 3 interquartile range (IQR) and the *red dots* show the data points that lie between 3 and 5 IQR (Color figure online)



(a) An Example of Successful Policy

(b) An Example of Unsuccessful Policy

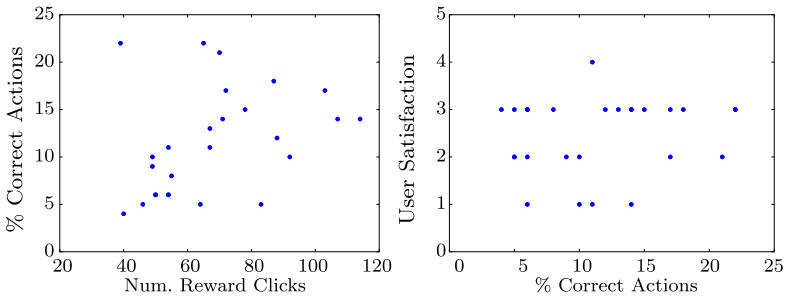
Fig. 4 Example Int-RL Policies

6.1 Interactive Reinforcement Learning Results

For our implementation of Interactive Reinforcement Learning, we used Q-learning with ε-greedy action selection. We used the following parameters for Q-learning: 0.5 for initial Q-values, with a learning rate $\alpha = 0.3$ and discount factor $\gamma = 0.75$. Figure 4 shows a successful and unsuccessful sample Int-RL policy. Each triangle shows a sub-sampled state that was not "tied" (where, more than one action had the same Q-value). Although every (x, y) pair symbolizes a subsampled state, to increase the readability of the plots, we left "tied" states empty. Ties can be caused by that state being unexplored or more than one action being equally rewarded by the teacher, i.e. noisy demonstrations. For example in Fig. 4a, approximately 20 % of the states were rewarded enough to assign a higher Q-value to one action than the others. For a given state, the action that has the highest Q-value (shown in Fig. 4) is executed during exploitation of the policy. If the state is equal to one of the empty states, then the action is selected randomly.

Figure 5a shows how the frequency of demonstrations is correlated with the agent's final accuracy. The scatter plot presents the number of inputs given by the user (positive or negative rewards) and the resulting policy accuracy. The correlation coefficient is found as $R = 0.35$ for this plot. This shows a very weak positive correlation between the two variables. That is, the increase in the number of rewards given is approximately 35 % responsible of the increase in final accuracy of the agent. Figure 5b shows the scatter plot for agent's final policy accuracy versus the users' self-reported satisfaction with the agent's performance in response to the question "How well did the robot learn the task?" (1-Not at all, 4-Very well). The correlation coefficient for this plot is $R = 0.16011$, which shows almost no correlation between the participants' perceived accuracy and the real performance of the agent. We believe that one reason for this finding can be the overall poor performance of the final policies. When we look at the y-axis, we can see that the maximum accuracy shows that approximately 22 % of the state space was labeled correctly.

Fig. 5 Int-RL User Statistics



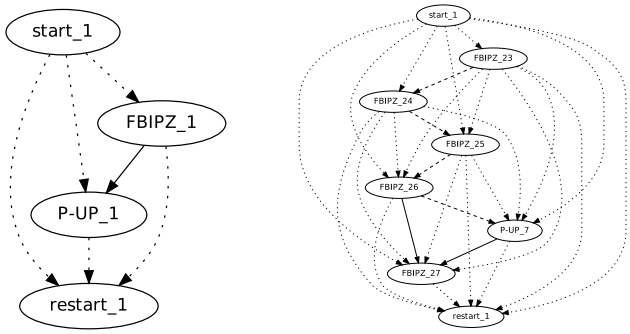
(a) Effect of user interaction on agent's performance

(b) User satisfaction vs. performance

6.2 Behavior Networks Results

Two of the networks taught by our participants are shown in Fig. 6. Each behavior is shown in an ellipse. Start and restart are pseudo-behaviors that signal the beginning and the end of the network. Dotted lines indicate that the behaviors above must happen before those below. Dashed lines are enabling preconditions, which indicate behaviors that enable the activation of others. Solid lines show permanent preconditions, that is, the behavior above should remain active while behaviors below are also active.

Figure 6a shows a simple and correct Behavior Network. When executed, this network would execute the pick-up action in presence of a bug in the pick-up zone, which is the correct behavior. Figure 6b shows a network that waits until it finds a HEXBUG in pick-up zone for 4 cycles (FBIPZ_23,24,25,26), then picks up the HEXBUG (P-UP_7), it also requires to find another HEXBUG in order to restart executing the network (FBIPZ_27 is a precondition of restart_1). This network is inefficient for the target task, and although it still executes the pick-up action after finding several HEXBUGs, the unnecessary interdependencies between behaviors reduce the efficiency of this policy.



(a) A simple and correct Behavior Network.

(b) A relatively more complicated Behavior Network produced by a participant.

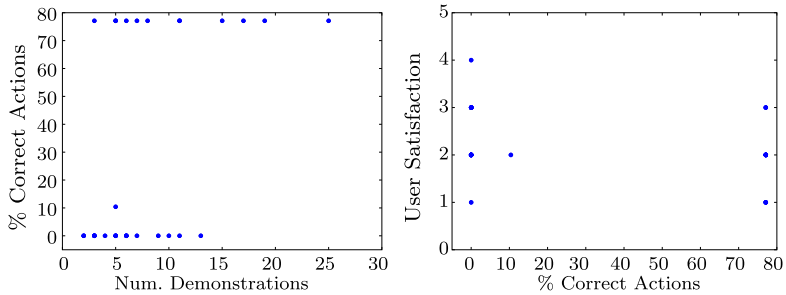
Fig. 6 Examples of the networks demonstrated by the subjects. P-UP: Pick-Up, FBIPZ: Found a bug in Pick-Up Zone. Numbers in the end of the behaviors show their unique id numbers

The Behavior Networks algorithm differs from the other two techniques in that the user demonstrations do not accumulate over time. The BNets algorithm generates a policy from the ordered set of demonstrations performed between the user-indicated START and END signals. As a result, the entire policy can be taught in a very short period of time by an expert (under a minute for the paper authors). A non-expert teacher who has little experience with the algorithm can demonstrate the task multiple times and the agent may still perform poorly. Indeed, we observed this outcome with several of our participants, who explored the behavior of the algorithm through trial and error to gain understanding of kinesthetic teaching, the robot's perception, and the conditioning behavior of the algorithm (pre- vs. post-condition). In summary, a large number of demonstrations is not required to teach a successful policy, however, we expected our participants to get better at demonstrating the task over time.

Figure 7a shows the scatter plot of the total number of demonstrations versus policy performance. We note that there are only three levels for the agent's accuracy. This behavior results from the pre- and post-conditions precoded into the algorithm. For example, once the BNets agent learns to pick-up a HEXBUG when it is located in the pick-up zone, the correct pickup behavior is automatically learned for every state in the pickup zone. This is due to the fact that the pre-condition of the pickup action generalizes across a region of states (the entire pickup area) and not a single (x, y) location.

Keeping these facts in mind, the three levels in Fig. 7a are respectively at 0 %, 10.4 % and 77.1 %, referring to accurate performance of *none of the actions*, *pick-up*, and *wait*. No participant was able to teach the *sweep* action successfully in their final policy. Importantly, we note that although the policy performance is 77.1 % for correctly teaching the wait action, this is caused by the fact that in most states, our domain required to wait. The correlation coefficient found for this plot is $R = 0.48$. This shows some (although not so strong) correlation between the participants' experience in teaching (i.e. number of demonstrations) and the accuracy of the final policy.

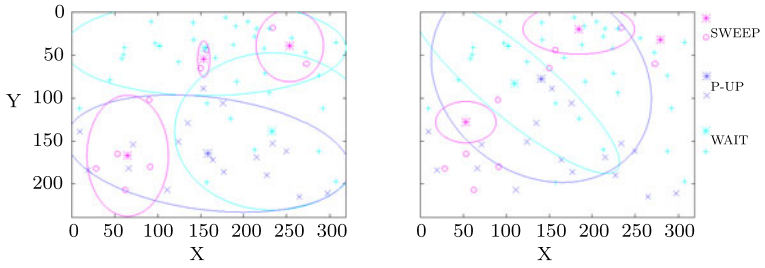
Fig. 7 BNets User Statistics



(a) Effect of user interaction on agent's performance

(b) User satisfaction vs. performance

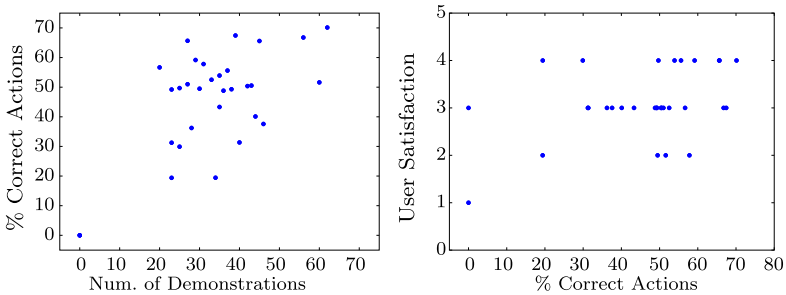
Fig. 8 Example CBA Policies



(a) An Example of Successful Policy

(b) An Example of Unsuccessful Policy

Fig. 9 CBA User Statistics



(a) Effect of user interaction on agent's performance

(b) User satisfaction vs. performance

Finally, Fig. 7b shows the scatter plot of the agent's accuracy versus the participants' self-reported satisfaction in response to the question "How well did the robot learn the task?". Although weak, the correlation coefficient $R = -0.35$ shows a negative correlation between these two variables. We believe that the correlation coefficient is negative because of the fact that users evaluated performance based on the success of picking up as many HEXBUGs as possible. Hence, *waiting* at appropriate times was not considered as successful as *picking up* or *sweeping*. Our measure of accuracy gives equal weight to all state-action pairs across the state space.

6.3 Confidence Based Autonomy Results

In order to assess the user statistics with CBA, we look at the final GMM classifier that was constructed for each user. A sample of a successful and unsuccessful CBA policy is given in Fig. 8. To help the reader compare both policies, we highlight several characteristics of the learned classifier. First, when we look at the Gaussian models for the *wait* action in Fig. 8a, we see two distinct models, covering most of the correct zones, whereas the Gaussian for the same action in Fig. 8b covers only a small part of the correct zone, and is mostly overlayed with other Gaussians. Second, in Fig. 8a, the Gaussian for *pick-up* action is centered very close to its zone, whereas in Fig. 8b, it covers most of the wait zone. Third, the biggest Gaussian model for *sweep* action is centered close to its zone in Fig. 8a whereas in Fig. 8b Gaussians are in the wait zone. Finally, two more Gaussian models for *pick-up* action can be seen in Fig. 8a, it is possible that they are the results of the violation of our assumption (2) mentioned in the previous section; when the user trains the agent with more training samples, the risk of obtaining more Gaussian models than necessary increases. Furthermore, incorrectly timed corrections introduce noise and lead to poor model accuracy.

As in previous methods, we calculate policy performance as the percentage of correctly labeled states. The scatter plot in Fig. 9a shows how number of demonstrations is correlated with policy performance. The correlation coefficient found for this plot is, $R = 0.64$, which implies a somewhat strong correlation between these two variables. Note that it would be inaccurate to compare the strength of this correlation to our previous number of demonstrations versus performance plots; although the policy performance is calculated using the equivalent data, the values representing the number of demonstrations have inherently different meaning due to the differences in the algorithms. As a result, we can *not* conclude that in CBA user demonstrations were more effective to improve the accuracy of the agent than in previous methods. However, we can conclude that when using CBA, user demonstrations seem to be accountable for 64 % of the improvement in policy performance.

Figure 9b shows the scatter plot of participant self-reported performance evaluation versus final policy performance. The correlation coefficient found for this plot is $R = 0.39$, which implies a weak correlation between our participants' votes and the actual performance of their agents. The question we asked was the same across all methods, hence

the variable on x-axis is comparable to previous methods. Likewise, the agent accuracy was calculated in the same way across all methods. As a result, we can compare the correlation coefficient we obtain for this plot to our previous findings. We found no correlation in Fig. 5b, a negative correlation in Fig. 7b, and a positive correlation in Fig. 9b. Based on these result, we conclude that participants were able to assess performance of their agents most accurately when using the CBA algorithm.

7 Discussion

Moving away from performance statistics, in this section we discuss several design factors for these algorithms.

7.1 Participant Feedback

The general survey included space where participants were able to write free-form feedback about the algorithms in the study. All 31 participants included feedback on their form; the following three topics were most frequently repeated:

- the need for additional feedback from the robot regarding what it was sensing,
- the need for better understanding of what the robot "is thinking", or what it understands about the task,
- the need of communicating with the robot more naturally, specifically through speech.

These three points, especially the importance of feedback, are surely not unprecedented findings. However, their importance is often underestimated in existing LfD methods, and our aim is to draw attention to these factors for future research.

We also asked our participants which tasks they would like robots to perform in the future. Figure 10 shows a simple world of the most repeated words in participants' answers.

7.2 Factors Affecting Agent Performance

Our user study has four main components for each method: the teacher, the interface, the algorithm and the task domain. The teacher and the task domain were the common components across all three experiments performed with each participant, whereas the interface and algorithms were held constant across participants. Quantifying the effect of each component on different aspects of the agent's performance requires a narrower, more focused user study. However based on our observations we can make general comments on how human and algorithmic factors contribute to each algorithm's performance.

Fig. 10 Participants' most repeated words in their answers to our question "What tasks do you want robots to perform in the future?"

Teaching the policy using the Int-RL algorithm requires more time than when using other methods. This is an algorithmic factor due to the fact that the agent selects its actions probabilistically and converges to a stable policy only after repeated exposure to feedback. The occurance of users abandoning rewards before the agent converges to a stable policy (possibly due to lack of feedback) is a human factor. People quitting early because learning is too slow might be considered to be a combination of both.

Teaching a BNets agent often requires trial and error. This is an algorithmic factor; the necessity of defining pre/post-conditions for each behavior leaves a knowledge gap between the person who defines them and the person who makes demonstrations. On the other hand, realizing the relationship between given demonstrations and resulting behaviors of the robot is a human factor. Again, people giving up on trial and error process after a few demonstrations (because robot is now behaving any differently) might be considered to be a mixed effect of both factors on agent's final performance.

Teaching a CBA agent requires demonstrations and corrections. There is requirement minimum number of demonstrations for the agent to begin classifying the training data. This is an algorithmic factor which may change based on the classifier implemented. Modeling the timing of corrections, on the other hand, is a human factor. Depending on the situation, for instance if the teacher is distracted and reacts slowly, corrections may have a negative impact on agent's final performance.

7.3 Required Task Knowledge by the Programmer

For the Int-RL and CBA algorithms, all that must be defined by the programmer is the state and action sets of the robot. For the Behavior Networks algorithm, the programmer needs to know not only the state and actions, but also the state values that will result from the execution of different actions in order to form the pre/post-conditions of behaviors. We found that coding the Behavior Networks required

a significantly higher degree of knowledge of the domain, as well as a precise measurement of the sensor values that would be observed. Without this pre-coded information, the robot's state would not match the post-conditions of any behaviors and no behaviors would be activated and learned.

8 Conclusion

We present the findings we obtained from a user study implemented in a real-world domain, in which we asked non-expert users to use and evaluate three different robot learning from demonstration algorithms. The three algorithms selected—Behavior Networks, Interactive Reinforcement Learning, and Confidence Based Autonomy—utilize distinctly different policy learning and demonstration approaches, enabling the study of a broad spectrum of the field.

We analyzed the user study data with respect to robot policy performance and user feedback. Our results show that users achieved better performance in teaching the task using the CBA algorithm. Users also positively rated the usability of the CBA algorithm, with many users showing preference for the ability to give direct commands to the robot. Many participants also positively reviewed the kinesthetic teaching interface used in the BNets algorithm. However, users were often confused by the opaqueness of the teaching experience and performance of the result. In case of the Interactive Reinforcement Learning algorithm, despite our use of a subsampled state-space to improve algorithm performance, the experiment time was not sufficient to allow participants to train the complete task. However, we found that the Int-RL algorithm most accurately modeled user behavior, whereas both CBA and BNets made algorithmic assumptions that did not hold when the algorithms were applied to naïve users.

An interesting direction for future work will be to extract the strengths of the three individual algorithms and to merge them with each other to create a new integrated method for learning from demonstration. The results of our study indicate that presenting a model of the agent's internal policy to the teacher during training may significantly improve transparency and usability of future algorithms. However, the presentation of this knowledge should be intuitive and straight-forward to the teacher, as it may cause more confusion than clarification.

References

1. Abbeel P, Ng AY (2004) Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st international conference on machine learning (ICML '04), Baniff, Canada
2. Akgun B, Cakmak M, Yoo JW, Thomaz AL (2012) Trajectories and keyframes for kinesthetic teaching: a human-robot interaction perspective. In: Proceedings of the international conference on human-robot interaction (HRI)
3. Aleotti J, Caselli S (2005) Trajectory clustering and stochastic approximation for robot programming by demonstration. In: 2005 IEEE/RSJ international conference on intelligent robots and systems (IROS 2005)
4. Argall BD, Chernova S, Veloso M, Browning B (2009) A survey of robot learning from demonstration. Robot Auton Syst 57:469–483
5. Atkeson CG, Schaal S (1997) Robot learning from demonstration. In: Fisher DH Jr (ed) Machine learning: proceedings of the fourteenth international conference (ICML '97), San Francisco, CA, pp 12–20
6. Bentivegna DC, Ude A, Atkeson CG, Cheng G (2002) Humanoid robot learning and game playing using PC-based vision. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS '02), Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland, October
7. Billing E, Hellström T (2010) A formalism for learning from demonstration. Paladyn J Behav Robot 1:1–13. doi:10.2478/s13230-010-0001-5
8. Bradski G, Kaehler A (2008) Learning OpenCV. O'Reilly Media
9. Cakmak M, Chao C, Thomaz AL (2010) Designing interactions for robot active learners. IEEE Trans Auton Ment Dev 2(2):108–118
10. Calinon S, Billard A (2007) Incremental learning of gestures by imitation in a humanoid robot. In: Second annual conference on human-robot interactions (HRI '07), Arlington, VA, March
11. Chella A, Dindo H, Infantino I (2006) A cognitive framework for imitation learning. Robot Auton Syst (Special issue on the social mechanisms of robot programming by demonstration) 54(5):403–408
12. Chernova S, Veloso M (2007) Confidence-based learning from demonstration using Gaussian mixture models. In: Proceedings of the international conference on autonomous agents and multiagent systems (AMMAS '07)
13. Chernova S, Veloso M (2009) Interactive policy learning through confidence-based autonomy. J Artificial Intelligence Research 1–25
14. Grollman DH, Jenkins OC (2007) Dogged learning for robots. In: International conference on robotics and automation, Rome, Italy, April, pp 2483–2488
15. Ijspeert AJ, Nakanishi J, Schaal S (2002) Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings of the IEEE international conference on robotics and automation (ICRA '02) (Received the ICRA '02 best paper award)
16. Kim ES, Leyzberg D, Tsui KM, Scassellati B (2009) How people talk when teaching a robot. In: Proceedings of the international conference on human robot interaction (HRI '09)
17. Bradley Knox W, Stone P (2010) Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In: Proceedings of the international conference on autonomous agents and multiagent systems
18. Mataric MJ (2002) Sensory-motor primitives as a basis for learning by imitation: linking perception to action and biology to robotics. In: Dautenhahn K, Nehaniv C (eds) Imitation in animals and artifacts. MIT Press, Cambridge, pp 392–422
19. Nehmzow U, Akanyeti O, Weinrich C, Kyriacou T, Billings SA (2007) Robot programming by demonstration through system identification. In: IEEE/RSJ international conference on intelligent robots and systems (IROS 2007)
20. Nicolescu MN, Mataric MJ (2001) Learning and interacting in human-robot domains. IEEE Trans Syst Man Cybern, Part A, Syst Hum 31(5):419–430
21. Nicolescu MN, Matarić MJ (2002) A hierarchical architecture for behavior-based robots. In: Proceedings of the first international joint conference on autonomous agents and multiagent systems, Part 1 (AAMAS '02), New York, NY, USA. ACM, New York, pp 227–233

22. Nicolescu MN, Mataric MJ (2003) Natural methods for robot task learning: instructive demonstrations, generalization and practice. In: Proceedings of the second international joint conference on autonomous agents and multiagent systems (AAMAS '03), New York, NY, USA. ACM, New York, pp 241–248
23. Smart WD (2002) Making reinforcement learning work on real robots. PhD thesis, Department of Computer Science, Brown University, Providence, RI
24. Suay HB, Chernova S (2011) A comparison of two algorithms for robot learning from demonstration. In: IEEE international conference on systems, man, and cybernetics
25. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
26. Thomaz AL, Breazeal C (2007) Asymmetric interpretations of positive and negative human feedback for a social learning agent. In: Proceedings of the 16th IEEE international symposium on robot and human interactive communication (RO-MAN)
27. Thomaz AL, Breazeal C (2008) Experiments in socially guided exploration: lessons learned in building robots that learn with and without human teachers. Connect Sci 20:91–110
28. Thomaz AL, Breazeal C (2006) Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In: Proceedings of the 21st national conference on artificial intelligence (AAAI)
29. van Lent M, Laird JE (2001) Learning procedural knowledge through observation. In: Proceedings of the 1st international conference on knowledge capture (K-CAP '01), New York, NY, USA. ACM, New York, pp 179–186
30. Voyles RM, Khosla PK (2001) A multi-agent system for programming robots by human demonstration. Integr Comput-Aided Eng 8(1):59–67

Halit Bener Suay is a Ph.D. student and a Research Assistant at Worcester Polytechnic Institute (WPI), Robotics Engineering. In 2011, he joined Robot Autonomy and Interactive Learning group. He has a B.Sc. and a M.Sc. degree in Aeronautics Engineering from Istanbul Technical University and The University of Tokyo. His current research interest if focused on Learning from Demonstration for robots.

Russell Toris is a Ph.D. student in Computer Science at Worcester Polytechnic Institute (WPI) and is a member of the Robot Autonomy and Interactive Learning group. His primary research interests include learning algorithms, cloud-solutions, and crowdsourcing for personal robotics applications, human-robot interaction, and social robotics. Russell received his B.S. with Honors from Dickinson College in 2011.

Sonia Chernova is an Assistant Professor at Worcester Polytechnic Institute (WPI), where she directs the Robot Autonomy and Interactive Learning group. Her research interests lie in interactive machine learning, adjustable autonomy, crowdsourcing and human-robot interaction. Her work focuses on the development algorithms that enable robots to learn through social interaction with humans. Prof. Chernova received her Ph.D. from Carnegie Mellon University in 2009 and completed a postdoc at the MIT Media Lab before joining WPI.