Newtonian Action Advice: Integrating Human Verbal Instruction with Reinforcement Learning

Socially Interactive Agents Track

Samantha Krening
Georgia Institute of Technology
Atlanta, GA
skrening@gatech.edu

Karen M. Feigh
Georgia Institute of Technology
Atlanta, GA
karen.feigh@gatech.edu

ABSTRACT

A goal of Interactive Machine Learning is to enable people without specialized training to teach agents how to perform tasks. Many of the existing algorithms that learn from human instructions are evaluated using simulated feedback and focus on how quickly the agent learns. While this is valuable information, it ignores important aspects of the human-agent interaction such as frustration. To correct this, we propose a method for the design and verification of interactive algorithms that includes a human-subject study that measures the human's experience working with the agent. In this paper, we present Newtonian Action Advice, a method of incorporating human verbal action advice with Reinforcement Learning in a way that improves the human-agent interaction. In addition to simulations, we validated the Newtonian Action Advice algorithm by conducting a human-subject experiment. The results show that Newtonian Action Advice can perform better than Policy Shaping, a state-of-the-art IML algorithm, both in terms of RL metrics like cumulative reward and human factors metrics like frustration.

CCS CONCEPTS

• **Human-centered computing → Interaction design**; *Human computer interaction (HCI)*; *Natural language interfaces*; • **Computing methodologies → Reinforcement learning**;

KEYWORDS

Interactive Machine Learning; Learning from Human Teachers; Reinforcement Learning; Natural Language Interface; Human-Subject Experiment; Verification

ACM Reference Format:
Samantha Krening and Karen M. Feigh. 2019. Newtonian Action Advice: Integrating Human Verbal Instruction with Reinforcement Learning. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 8 pages.

1 INTRODUCTION

A challenge of Interactive Machine Learning (IML) is to design algorithms that learn from human feedback rather than simulated input, i.e. oracles. Several factors must be accounted for to elicit human feedback from non-experts, including: 1) how people intuitively teach and 2) what aspects of interaction foster a positive human experience. Creating a positive human interaction is necessary because people will not use an algorithm that creates a poor experience. Unfortunately, many existing IML algorithms are evaluated solely using oracles, assuming oracle and human input are equivalent [2, 4, 19, 23]. Oracle evaluations result in valuable information about the theoretical efficiency of an algorithm, but ignore important aspects of the human-agent interaction such as how humans react to the agent. For example, an oracle will never get frustrated with the agent. We suggest that validating interaction algorithms with oracles and analyzing traditional Reinforcement Learning metrics such as cumulative reward is only the first step. The next validation step should be measuring peoples' experiences using human factors metrics, such as frustration. This work demonstrates a template for IML algorithm design and verification in which researchers: 1) design for a positive human experience, 2) test the algorithm with oracles, and 3) verify the algorithm with a human-subject experiment measuring human factors.

This paper introduces an algorithm, Newtonian Action Advice, which incorporates a human's verbal action advice with RL ("Move right. Go down."). The algorithm leverages a simple physics model to provide an agent that acts in a way people expect and find non-frustrating. We validated our algorithm by first constructing oracles to simulate human feedback to compare Newtonian Action Advice (NAA) with Policy Shaping and Bayesian Q-learning. In addition to the simulations, we conducted a human-subject experiment in which participants trained both NAA and Policy Shaping agents, and then reported on the experience of working with both agents. For equivalent human input, Newtonian Action Advice performs better than Policy Shaping, both in terms of RL metrics like cumulative reward and human factors metrics like frustration.

2 BACKGROUND

2.1 Reinforcement Learning (RL)

RL is a form of machine learning influenced by behavioral psychology in which an agent learns what actions to take by receiving rewards or punishments from its environment [21, 24]. The probability people will repeat an action in a given circumstance is increased or decreased if they receive positive or negative reinforcement.

Most RL algorithms are modeled as Markov Decision Processes (MDPs), which learn policies by mapping states to actions such that the agent's expected reward is maximized. An MDP is a tuple (S, A, T, R, γ) that describes S, the states of the domain; A, the actions the agent can take; T, the transition dynamics describing the probability that a new state will be reached given the current state

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

and action; R, the reward earned by the agent; and γ, a discount factor in which $0 \leq \gamma \leq 1$.

Bayesian Q-Learning (BQL) is an MDP-based RL algorithm in which the utility of state-action pairs are represented as probabilistic point estimates of the expected long term discounted reward [3]. BQL was used as the underlying RL algorithm for both the Policy Shaping and NAA interaction methods in this work.

2.2 Learning from Critique and Advice

Two methods of human instruction that can be used to train IML algorithms are critique and advice. Critique is positive or negative feedback provided in response to an agent's previous actions; e.g. a person saying "good" or "bad". Advice is when a person tells the agent what action to take next; e.g. "move left."

Critique was initially used directly as a reward signal [6], but it was later shown [8, 27] that it is more efficient to use critique as policy information. Policy Shaping is an interaction algorithm that enables a human teacher's critique to be incorporated into a BQL agent as policy information [4] and was used in this work.

Various forms of advice have been developed in other work, including linking one condition to each action [14], linking a condition to rewards [13], and connecting objects to actions [11]. Several connect conditions to higher-level actions that are defined by the researcher instead of primitive actions [7, 12, 14]. Argall et al. [1] creates policies using demonstrations and advice. Meriçli et al. [16] parses language into a graphical representation and finally to primitive actions. Maclin et al. [14] has the person provide a relative preference of actions. Sivamurugan and Ravindran [20] explored learning multiple interpretations of instructions. Tellex et al. [25] represents natural language commands as probabilistic graphical models.

Most methods are permanently influenced by the advice. Kuhlmann et al. [12] can adjust for bad advice by learning biased function approximation values that negate the advice. Maclin et al. [14] uses a penalty for not following the advice that decreases with experience. The Newtonian Action Advice developed in this paper differs because the advice can be overwritten by new, contradictory advice in the future. This makes NAA more forgiving to an imperfect user compared to other algorithms including Policy Shaping.

Many researchers incorporate advice using IF-THEN rules and formal command languages [12, 14]; if the state meets a condition, then the learner takes the advice into account. Formal command languages and IF-THEN rules require advice that is state specific and contains numbers. Similar to this work, the advice in Argall et al. [1] does not require people to give specific numbers for continuous state variables, but uses a set of predefined advice operators.

Most IML studies rely on oracle simulations rather than human-subject experiments. There are some studies that use human subjects such as Cederborg et al. [2], which investigated how to interpret silence while learning from critique with Policy Shaping. However, studies like this elicit human instruction in a non-interactive manner and use the instruction offline to train agents. Such studies demonstrate valuable algorithmic information, yet they analyze objective ML metrics while ignoring the human experience. One of our goals is to show the necessity of interactive human-subject experiments that measure human factors.

2.3 Natural Language Processing (NLP)

The two NLP tools used in this work were Automatic Speech Recognition (ASR) and sentiment analysis. ASR software transcribes the human teacher's verbal instructions to written text. The human-subject experiment in this work used Sphinx ASR software [5].

Sentiment analysis is an NLP tool used to classify movie, book, and product reviews into positive and negative [18]. Sentiment analysis has not been widely applied to action selection. One method we previously developed for using sentiment analysis is to classify natural language advice into advice of 'what to do' and warnings of 'what not to do' [11]. Many approaches to learning from language instruction require people to provide instructions using specific words, often in a specific order or format [16]. Thomason et al. [26] worked to get around limitations like keyword search by creating an agent that learns semantic meaning from the human. In this work we created a method of using sentiment analysis to filter verbal critique into positive and negative, which furthers the goal of allowing people to provide verbal instructions without being limited to a specific dictionary of words.

This work uses Stanford's deep learning sentiment analysis software [15], which uses Recursive Neural Tensor Networks and the Stanford Sentiment Treebank [22]. The Stanford Sentiment Treebank is a set of labeled data corpus of fully-labeled parse trees trained on the dataset of movie reviews from rottentomatoes.com [17].

Algorithm 1 Newtonian Action Advice algorithm

```
1:  procedure NAA
2:      for each time step do
3:          Listen for human advice
4:          if human advice given then
5:              newAdvice(state, advice)
6:          action = actionSelection()
7:          Take action, get reward
8:          Update BQL policy with reward
9:  procedure newAdvice(state, advice)
10:     adviceJustGiven ← True
11:     adviceDict[state] = advice
12:     currentAdvice = advice
13: procedure actionSelection(state)
14:     if adviceJustGiven == True then
15:         chosenAction ← currentAdvice
16:         timesNewAdviceFollowed+ = 1
17:         if state ∉ adviceDict then
18:             adviceDict[state] = chosenAction
19:         if timesNewAdviceFollowed >= S_des then
20:             adviceJustGiven = False
21:             timesNewAdviceFollowed = 0
22:     else if state ∈ adviceDict then
23:         chosenAction = adviceDict[state]
24:     else
25:         chosenAction = BQLactionSelect(state)
26:     return chosenAction
```

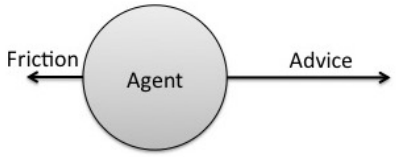


Figure 1: Simple Force Model. Actions are an external force acting on the agent, and 'friction' determines the amount of time the action will be followed after the advice is given.

3 NEWTONIAN ACTION ADVICE

Newtonian Action Advice is an interaction algorithm we designed to enable an RL agent to learn from human action advice. The theory is a metaphor of Newtonian dynamics: objects in motion stay in motion unless acted on by an external force. In the NAA model, a piece of action advice provided by the human acts as an external force on the agent. Once a person provides advice (ex: "Go right"), the agent will immediately move in the direction of the external force, superseding the RL agent's normal action selection. The model contains natural friction that 'slows down' the agent's need to follow the human's advice; this ensures that after some amount of time, the agent will resume the RL algorithm's action selection. Imagine a kid on roller skates has been gently pushed in one direction - the push is advice and friction will eventually bring the kid to a stop so she's no longer following the advice. In the NAA algorithm, the friction brings the advice to a stop and lets the kid (agent) choose its own action after the advice has 'run out'. The advice does not necessarily need to specify directional motion.

Newtonian Action Advice was designed to behave in a manner that is intuitive for the human teacher. The force model allows each piece of advice to be generalized through time. If a person says, "go right," the NAA agent will keep moving right until the 'friction' causes the agent to resume normal exploration. The simplicity of the force model is a feature to improve the human experience; people deal with Newtonian mechanics in their everyday life. For example, a ball when thrown will rise to a certain height and fall back to the ground. The motion is predictable, and one does not need physics training to recognize and expect the motion. We expect that loosely mimicking this will help create a user-friendly experience.

If advice was followed in a state, the agent will follow the same advice if the state is seen in the future. This means that a person will only have to provide advice once for a given situation. Also, only the latest advice is saved for a state, so people can correct mistakes or change the policy in real time.

BQL was used as NAA's underlying RL algorithm. This choice was primarily made so the NAA algorithm could be more directly compared to Policy Shaping. The structure of the NAA algorithm is such that the BQL algorithm could be exchanged for a different RL algorithm.

At each time step in *NAA* in algorithm 1, the agent listens for advice. If advice is given, the agent updates its advice dictionary. The agent then chooses and takes an action, receives a reward, and updates the BQL policy. The *New Advice* procedure adds the new (state, advice) pair to the agent's dictionary and sets a parameter that will tell the *Action Selection* procedure to follow the new advice.

The *Action Selection* procedure checks to see whether advice has recently been given and should still be followed. If the advice is being generalized through time (due to low friction) and the new (state, advice) pair has not been added to the dictionary, it will be added. If this state is revisited in the future, the recent advice given for a previous state will be applied as if it had been given for this state, too. The friction timer is then updated. If the timer indicates that the advice has been followed enough, parameters will be reset so the agent will return to the BQL's action selection for the next time step. If advice has previously been given for this state, the agent must choose between the human advice and the BQL suggestion. For this work, we always choose the human's advice. If a researcher wants to encourage more exploration, a different method can be chosen (e.g. an algorithm similar to $\epsilon - greedy$ applied to human vs. agent action selection instead of exploration vs. exploitation). However, we have found that following advice in a probabilistic manner increases frustration since the agent seems to disregard advice [10]. In the case that no advice has been given for or generalized to the current state, the action is chosen from the BQL's method.

3.1 Combining Supervised and Reinforcement Learning to allow for personalization by end-users

When designing an interactive machine learning algorithm, one first must ask: what is the goal of IML? Is the goal to use human instruction to decrease training time? Or is the goal to enable people to teach an agent to perform a task *in the way the human intends*? If the goal of IML is to use human instruction to decrease the amount of time it takes to train the RL agent, then we do not need to care about the nature of the learned policy. IML can use powerful RL algorithms that are capable of learning from their environment, and decrease training time by using human input.

However, if the goal is to get the agent to perform a task that a non-expert specifies *in the way the human wants the task done*, then we have a problem. The policy a RL agent learns is very sensitive to the reward function. In this work, as well as most IML research, the reward function is provided by the researcher. Given a reward function, an RL algorithm will learn a policy to maximize the reward, but the policy learned may be very alien to a human mind. The agent will complete the task efficiently, but not in a way that makes sense to a human. Given a reward function and human input, the RL algorithm may initially learn a policy that conforms to the human's instructions, but eventually learn a policy that solely maximizes cumulative reward. This satisfies a goal of decreasing training time since the human will have shown the RL agent high-earning states earlier than it would have explored, but the policy may still be baffling to a non-expert. The human teacher may think their instructions were disregarded in the long-term, creating feelings of frustration and powerlessness when they cannot directly control the agent's policies.

NAA combats this problem by combining supervised and reinforcement learning to allow for personalization by end-users. A human provides information that is used to create policies that are static to the agent (supervised), but can be overwritten by the human. The agent uses RL to determine the best course of action

for parts of the state space in which human advice is not given. Not only does this enable the agent to use human input to decrease training time, it also empowers the human to customize how the agent performs the task. It is possible the learned policy will be near-optimal instead of optimal from an objective analysis of the cumulative reward; but the agent's performance will be in greater accordance with the human teacher's instructions, which will increase the human's satisfaction with the agent's performance.

3.2 Choosing the friction parameter

When calculating the algorithm's 'friction' parameter, it is beneficial to think of the parameter as the number of time steps each piece of advice should persist for, S_{des}. Increasing S_{des} causes each piece of advice to be followed for a longer time, which causes a metaphorically lower friction in the model.

Let $(S_{des}, S_{min}, S_{max})$ be the (desired, minimum, maximum) number of steps advice should persist for with units of steps. Let $(\Delta t_{des}, \Delta t_{min}, \Delta t_{max})$ be the (desired, minimum, maximum) time between given advice in seconds. U (steps/second) is the domain update rate. Equations 1-3 show how to calculate the minimum, maximum, and desired values of the friction parameter. Two items should be noted for S_{min}. First, the value of Δt_{min} has a lower bound based on human limitations. It is nonsensical to provide advice that only lasts for a fraction of a second; if Δt_{min} is too small, it may occur that the agent follows the advice for such a short amount of time that it is not perceivable by the human. We suggest $\Delta t_{min} \geq 0.5(seconds)$. Second, the advice must last for at least one time step, so $S_{min} \geq 1(step)$. Depending on the domain, task, and nature of the actions, we suggest a starting value of Δt_{des} to be between 2-8 seconds.

$$S_{des} = \Delta t_{des} * U \tag{1}$$

$$S_{min} = \Delta t_{min} * U \tag{2}$$

$$S_{max} = \Delta t_{max} * U \tag{3}$$

Equation 4 shows the bounds on the friction parameter. The value of S_{min} has the potential to run into a hard boundary based on human limitations, while S_{max} is a flexible boundary based on desired behavior.

$$(1step) \leq S_{min} \leq S_{des} \leq S_{max} \tag{4}$$

Instead of calculating the friction parameter using the desired time between when advice is given, the average duration of each action can be used. Let $\Delta a(actions)$ be the desired number of actions between when advice is given. spa_{avg} is the average number of steps it takes to complete an action in units of (steps/action). tpa_{avg} (seconds/action) is the average time it takes to complete an action. If the human teacher is instructing the agent to take primitive actions, then $spa_{avg} = 1(step)$. If the human is providing instructions for higher order actions, then $spa_{avg} \geq 1(step)$. An action must last for the duration of at least one time step, so $\Delta a \geq 1$. Equations 5 and 6 show expressions for S_{des} depending on whether the researcher has easier access to spa_{avg} or tpa_{avg}.

$$S_{des} = \Delta a * spa_{avg} \tag{5}$$

$$S_{des} = \Delta a * tpa_{avg} * U \tag{6}$$

If primitive actions are being used, it is likely that the ideal S_{des} parameter will be fairly large because the human teacher will want a given piece of advice to be followed for several consecutive time steps. If higher order actions are being used, a smaller S_{des} may be beneficial because it will already take the agent several time steps to carry out the higher order action.

4 EXPERIMENTAL METHOD

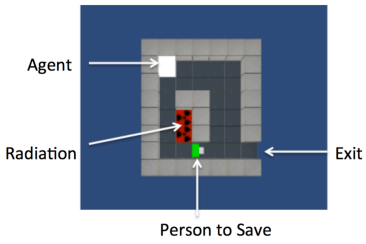


Figure 2: Radiation World Initial Condition

We validated our NAA algorithm first with oracles to test the theoretical performance of the algorithm, and then with a human-subject experiment to compare the human teacher's experience with another IML algorithm, Policy Shaping.

Both the simulations and human-subject experiment used the same task domain. Oracles and human participants were required to teach agents to rescue a person in Radiation World, a game developed in the unity minecraft environment (Figure 2). In the experimental scenario, there has been a radiation leak and a person is injured and immobile. The agent must find the person and take him to the exit while avoiding the radiation. In the Radiation World map, the light gray represents walls and dark gray is open space where the agent can move.

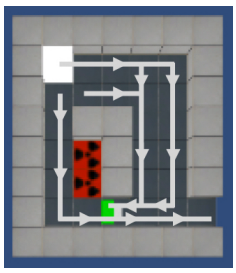
4.1 Constructing Oracles

We first tested the Newtonian Action Advice algorithm with simulations that used a constructed oracle to simulate human feedback. Each oracle was instantiated with a probability, p_{advice}, that determined how often to check for advice from the oracle. We provided the advice for the oracles to test several cases, including maximum friction, two cases of minimal advice, and decreased friction.

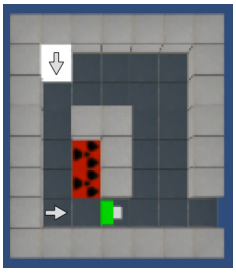
The same oracle method and advice dictionary were used to test the Policy Shaping agent. The advice dictionary was converted to critique for the Policy Shaping agent in the following manner: if the agent took the advised action for the state, the critique was positive; otherwise, the critique was negative.

4.2 Human-Subject Experiment

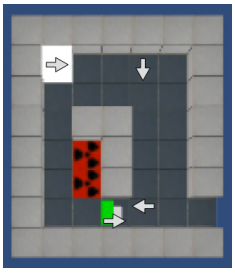
We conducted a repeated measures human-subject experiment in which we investigated the effect of two different interaction methods, NAA and Policy Shaping, on the human's experience of teaching the agent. Both agents learned from verbal instruction, which was transcribed to text using ASR software. After language processing, the human instructions were sent to an interaction algorithm (Policy Shaping or NAA).



a: Extensive advice given to simulation to avoid radiation.

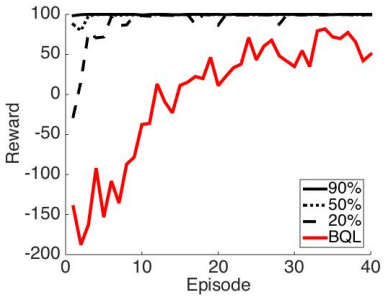


b: Minimal advice given to simulation to complete task with minimal steps.

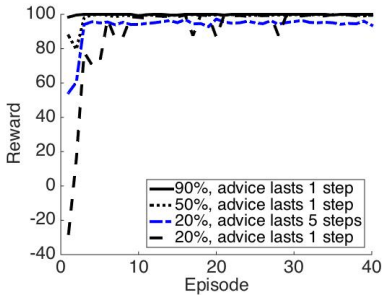


c: Minimal advice given to simulation to avoid radiation.

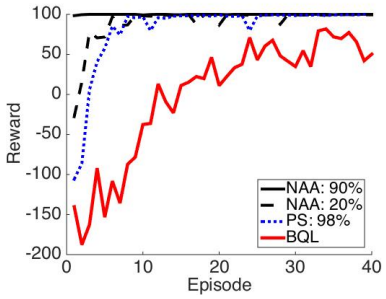
Figure 3: Advice oracles were instantiated with.



a: The amount of advice provided impacts reward. (advice given 0, 20, 50, and 90% of the time)



b: Friction parameter impacts reward. (advice given 20, 50, 90% of the time)



c: Policy Shaping with critique given 98% of the time performs worse than NAA with 20% advice.

Figure 4: Cumulative reward comparisons

The Policy Shaping agent learned by incorporating a human teacher's positive and negative critique. People were instructed to provide critique in response to the agent's actions. We used sentiment analysis as a filter to enable people to provide verbal critique without restricting their vocabulary. For example, a participant could give varied critique such as, "That's great," or "That is a bad idea."

The NAA agent learned from a human teacher's action advice. Participants were instructed to tell the agent to move in a desired direction. For example, if participants wanted the agent to move right, they should say, "right." The only four words the participants used while training the NAA agent were, "up," "down," "left," and "right." These four directions were grounded to the agent's actions.

The experiment collected data from 24 participants, with an age range of 18-62 years old. The experiment randomly split participants into two groups. The first group trained the Policy Shaping agent first, and the second group trained the NAA agent first. Participants were told to stop training when either the agent was performing as intended or the participant wanted to stop. After participants finished training an agent, they filled out a questionnaire concerning the experience. In the questionnaires, the participants scored frustration, perceived performance, transparency, immediacy, and perceived intelligence. For example, immediately after training an agent, the participants were asked to score the intelligence of the agent on a continuous scale from [0:10]. A value of 0 indicated that the agent was not intelligent, while 10 meant very intelligent. The same scale of [0:10] was used for additional human factors metrics including performance, frustration, transparency, and immediacy. Values of 0 corresponded to poor performance, low frustration, non-transparent use of feedback, and a slower response time. Values of 10 meant excellent performance, high frustration, clear use of feedback, and an immediate response time, respectively. A statistical analysis determined if there was a significant difference in the human experience between the two interaction methods.

5 RESULTS AND DISCUSSION

5.1 Oracle Results

5.1.1 Test 1: No generalization through time (extreme friction). We simulated how the percentage of time advice is followed impacts performance. In this simulation, the NAA agent did not generalize a given piece of advice to other states immediately after the advice was given, meaning that one piece of advice counted for only one time step (maximum friction with $S = 1(step)$). The oracle was built with advice given for every square in the grid (Figure 3a).

Incorporating human instruction by using the Newtonian Action Advice algorithm allows the RL agent to achieve a higher level of performance in many fewer episodes than without human input. As advice is given for a greater number of individual times steps (increasing from 20% to 90%), the agent accumulates more reward

and completes each episode with fewer actions (Figure 4a). The case with no human input is shown as BQL on the figures.

5.1.2 Test 2: Minimal Advice - shortest path. The minimal advice to take the shortest path (which takes the agent next to the radiation) is comprised of only two pieces of action advice equivalent to a human saying, "First move *down*. Then go *right*." The minimal advice used to create the oracle in this case is represented in Figure 3b. Given only two pieces of advice, the NAA agent was able to complete the episode in 10 steps achieving a reward of 102.0 every single episode. The NAA agent was set to follow each piece of advice for $S = 5(steps)$ before returning to the BQL baseline action selection.

The NAA model with a decreased friction parameter allows a human teacher to say "down, right," instead of, "down, down, down, down, down, right, right, right, right, right." It makes for a better and more intuitive experience to provide less instruction and not have to constantly repeat advice.

5.1.3 Test 3: Minimal Advice - avoiding radiation. The minimal advice to take a path that avoids the radiation is comprised of only four pieces of action advice equivalent to a human saying, "First move *right*. Then go *down*. Move *left* then immediately *right* after rescuing the injured person." This advice, which was used to construct the oracle for this case, can be seen in Figure 3c. Given only four pieces of advice, the NAA agent was able to complete the episode in 12 steps achieving a reward of 100.0 every single episode. The NAA agent was set to follow each piece of advice for $S = 5(steps)$ before returning to the BQL baseline action selection.

This case is an example of the optimal vs. customized discussion in Section 3.1. The learned policy was near-optimal instead of optimal from an objective analysis of the cumulative reward since the path to avoid the radiation was slightly longer, but the agent's performance was in accordance with the human teacher's advised path.

5.1.4 Test 4: Generalization through time (friction effect). We studied how the algorithm performs as the friction of the NAA model is decreased (increased S parameter). Both minimal advice tests have shown that a small amount of advice paired with a lowered friction can enable the NAA agent to perform optimally or near-optimally from the first episode. To test the friction effect, we built an oracle with the same advice given for every square as the extreme friction oracle (Figure 3a).

When advice is given 20% of the time, the agent with a lower friction $S = 5(steps)$ initially performs better than a higher friction $S = 1(step)$. However, in later episodes the agent with lower friction earns a lower cumulative reward while taking more steps to complete each episode compared to the high friction agent. In general, these results indicate that, while lowering the friction can increase initial performance, it can also cause a lower-performing policy to be learned by the agent.

But what is really going on in this case? We have seen in both minimal advice tests that minimal advice paired with a lowered friction enables the agent to perform optimally or near-optimally from the very first episode. Why would providing more advice ($p_{advice} = 20\%$) harm the agent's performance, particularly when the extreme friction case showed that increasing advice increases performance? The core issue is a limitation of the oracle. At every time step, the oracle listens for advice with a probability of 20%. This is not how a human would provide advice. The decreased performance in this case occurs when the agent spends time repeatedly banging into walls after advice has been generalized to a wall state instead of the oracle providing advice for that state. The probability $p_{advice} = 20\%$ is low enough that this behavior is not corrected for many episodes. Human teachers who observed this behavior would quickly provide an extra piece of advice to make sure the agent did not fruitlessly waste time. When people decrease advice, they tend to limit themselves to the most important pieces of advice, such as the minimal advice cases. The oracle has no way to know which advice is the most important, and so provides advice in a way that is not indicative of human behavior.

A possible solution to this problem is to build more elaborate oracles that more accurately represent human behavior. There are three main issues with this approach: 1) the use of and response to an algorithm will vary across individuals, so multiple contradictory oracles would need to be constructed, 2) an oracle's ability to provide a type of input does not mean a human is likely or able to provide that input in reality, and 3) it is very unlikely that even the most elaborate oracle could simulate the human's response to the agent, such as frustration. A more practical solution to this problem is to test algorithms with human-subject experiments.

This case shows why IML researchers should verify interaction algorithms with human-subject experiments in addition to simulations. If we had analyzed these results without understanding the limitations of the oracle, we might have discarded parameterizations using a lower friction.

5.1.5 Test 5: Comparison of Newtonian Action Advice and Policy Shaping. Figure 4c shows that, given equivalent input, Newtonian Action Advice can learn faster using less human instruction than Policy Shaping. Even when Policy Shaping used advice 98% of the time, it learned slower than the NAA agent that was using input only 20% of the time. The oracle used the same setup as the extreme friction and friction effect cases (Figure 3a).

When learning from human teachers in practice, however, the performance of each agent is entirely dependent on the instruction provided by the person. Neither agent is guaranteed to perform better than the other. If the human provides no instructions, the Policy Shaping and NAA agents perform equally since they both reduce to a BQL algorithm. In order to investigate how the performance of the agents varied with real human teachers, as well as how the human experience was impacted by interacting with each agent, we conducted a human-subject experiment.

5.2 Human Subject Results

This section will summarize the results from the human-subject experiment and provide further evidence of why it is necessary to use human-subject studies instead of relying only on oracle simulations. An in-depth analysis of the human factors reasoning about why people prefer NAA to Policy Shaping is found in the associated publications (Krening and Feigh [9, 10]).

Immediately after training each agent, participants were asked to score aspects of their experience training the agent, including frustration, perceived performance, transparency, immediacy, and

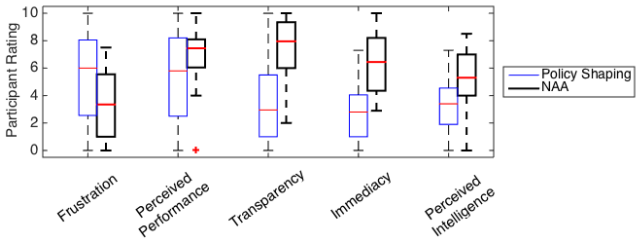


Figure 5: Comparison of the Subjective Human Factors Metrics from the Human-Subject Study.

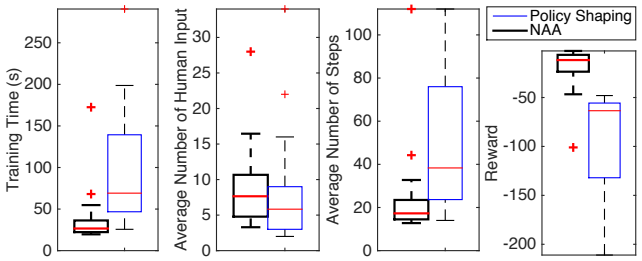


Figure 6: Comparison of the Objective Metrics from the Human-Subject Study.

perceived intelligence (Figure 5). Paired t-tests were conducted for each metric in which the null hypothesis was the pairwise difference between the two paired groups had a mean equal to zero. We found that all measured aspects of the human experience differed significantly between the two agents (Table 1). For all metrics except frustration, a higher value indicates a better experience (more intelligent, more transparent, responds immediately to instruction, and better performance). Low values indicate low frustration, less intelligent, less transparent, slow response to instruction, and worse performance.

Table 1: Results of Statistical Tests on Subjective and Objective Metrics from the Human-Subject Study

Subjective Metrics	Accept/Reject	p
Frustration	Reject	0.0046
Transparency	Reject	1.3738e-05
Perceived Intelligence	Reject	1.4192e-04
Perceived Performance	Reject	0.0350
Immediacy	Reject	2.0291e-06
Objective Metrics	**Accept/Reject**	p
Training Time (s)	Reject	7.6406e-04
Avg Reward	Reject	3.3849e-07
Avg Number Inputs	Accept	0.2627
Avg Number Steps	Reject	0.0037

In summary, compared to Policy Shaping participants found the Newtonian Action Advice agent to be:

- More intelligent
- Less frustrating
- Clearer in terms of how the agent used human input
- More immediate in terms of using human input
- Better able to complete the task as the person intended

In addition to creating a better human experience, the NAA agent also performed better than Policy Shaping in terms of objective RL metrics (Figure 6). Paired t-tests were run on training time and average reward, while Wilcoxon Signed Rank tests were performed on the number of inputs and steps (Table 1). The average training time (defined as the amount of time the person trained the agent) and number of steps the agent took to complete each episode were smaller for NAA. The average reward was higher for NAA than Policy Shaping. However, the average number of inputs provided by the human teachers was statistically equal for the two interaction algorithms.

The nature of the agent's response to the human's instruction created significantly different human experiences between the two agents. Consider that both agents used the same underlying BQL algorithm, and so would perform equally with no human input. Also, people provided a statistically equal number of instructions for both agents, so both were equally effortful for the person. And yet, the peoples' perceptions of the agents differed across all subjective metrics, including frustration, perceived intelligence, and transparency. A thorough analysis of the participants' responses in Krening and Feigh [10] shows that the main factors that influenced the participants' frustration levels were whether the agent's behavior made the person feel powerless, whether the agent's choices were transparent, the complexity of the instruction format, and whether the agent immediately acted on the person's instructions. The objective metrics like training time and reward were not factors mentioned by the participants. This tells us that the human experience cannot be improved solely by designing algorithms to optimize the objective metrics that are available in oracle testing.

Think of control systems for a minute. Control systems usually contain a controller and observer. An observer estimates values of the state space like position and velocity so the controller knows where the agent is and what it has to do to reach a goal. If a control system does not observe part of the state space, like velocity, the controller may push the system to move dangerously fast or mathematically diverge to infinity and "blow up". IML verification that only uses oracle simulations is like a control system that is not fully-observed; the human experience is "blowing up" because it is never observed by researchers, and so algorithms are not designed to correct for the human experience. To correct this, we need to observe the human experience by performing human-subject experiments that analyze human factors such as frustration. Then, we need to use the results from the human participants to influence the design of algorithms for a better human experience. If people dislike interacting with an agent, they will not continue to use that agent.

6 CONCLUSIONS

This paper presented Newtonian Action Advice, a method to integrate a human's interactive action advice (ex: "move left") with RL. The results show that Newtonian Action Advice can perform better than Policy Shaping, both in terms of RL metrics like cumulative

reward and human factors metrics like frustration. NAA can learn faster using less human instruction than Policy Shaping. NAA creates a better human experience than Policy Shaping. Compared to Policy Shaping, participants found the NAA agent to be less frustrating, clearer and more immediate in terms of how the agent used human input, better able to complete the task as the participant intended, and more intelligent.

This work also acts a template for the design and verification of IML algorithms that includes: 1) designing for a positive human experience, 2) testing the algorithm with oracles, and 3) verifying the algorithm with a human-subject experiment measuring human factors. In order to create algorithms people will enjoy interacting with, we must move beyond oracle simulations to test agents in human-participant studies measuring human factors.

ACKNOWLEDGMENTS

This work was funded under ONR grant number N000141410003.

REFERENCES

[1] Brenna D Argall, Brett Browning, and Manuela Veloso. 2008. Learning robot motion control with demonstration and advice-operators. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 399–404.

[2] Thomas Cederborg, Ishaan Grover, Charles L Isbell, and Andrea Lockerd Thomaz. 2015. Policy Shaping with Human Teachers.. In *IJCAI*. 3366–3372.

[3] Richard Dearden, Nir Friedman, and Stuart Russell. 1998. Bayesian Q-learning. In *AAAI/IAAI*. 761–768.

[4] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Isbell, and Andrea L Thomaz. 2013. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems*. 2625–2633.

[5] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alexander I Rudnicky. 2006. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, Vol. 1. IEEE, I–I.

[6] Charles Isbell, Christian R Shelton, Michael Kearns, Satinder Singh, and Peter Stone. 2001. A social reinforcement learning agent. In *Proceedings of the fifth international conference on Autonomous agents*. ACM, 377–384.

[7] Madhura Joshi, Rakesh Khobragade, Saurabh Sarda, Umesh Deshpande, and Swati Mohan. 2012. Object-Oriented Representation and Hierarchical Reinforcement Learning in Infinite Mario. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, Vol. 1. IEEE, 1076–1081.

[8] W Bradley Knox and Peter Stone. 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 5–12.

[9] Samantha Krening and Karen M Feigh. 2018. Characteristics that influence perceived intelligence in AI design. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 62. SAGE Publications Sage CA: Los Angeles, CA, 1637–1641.

[10] Samantha Krening and Karen M Feigh. 2018. Interaction Algorithm Effect on Human Experience with Reinforcement Learning. *ACM Transactions on Human-Robot Interaction (THRI)* 7, 2 (2018), 16.

[11] Samantha Krening, Brent Harrison, Karen M Feigh, Charles Lee Isbell, Mark Riedl, and Andrea Thomaz. 2017. Learning from Explanations using Sentiment and Advice in RL. *IEEE Transactions on Cognitive and Developmental Systems* 9, 1 (2017), 44–55.

[12] Gregory Kuhlmann, Peter Stone, Raymond Mooney, and Jude Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *The AAAI-2004 workshop on supervisory control of learning and adaptive systems*.

[13] James MacGlashan, Monica Babes-Vroman, Marie desJardins, Michael Littman, Smaranda Muresan, Shawn Squire, Stefanie Tellex, Dilip Arumugam, and Lei Yang. 2015. Grounding English Commands to Reward Functions. In *Proceedings of Robotics: Science and Systems*. Rome, Italy.

[14] Richard Maclin, Jude Shavlik, Lisa Torrey, Trevor Walker, and Edward Wild. 2005. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the National Conference on Artificial intelligence*, Vol. 20. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 819.

[15] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010

[16] Cetin Meriçli, Steven D Klee, Jack Paparian, and Manuela Veloso. 2014. An interactive approach for situated task specification through verbal instructions. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1069–1076.

[17] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 115–124.

[18] Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval* 2, 1-2 (2008), 1–135.

[19] Himanshu Sahni, Brent Harrison, Kaushik Subramanian, Thomas Cederborg, Charles Isbell, and Andrea Thomaz. 2016. Policy shaping in domains with multiple optimal policies. In *Proceedings of the 2016 International Conference on AAMAS*. International Foundation for AAMAS, 1455–1456.

[20] Manimaran Sivasamy Sivamurugan and Balaraman Ravindran. 2012. Instructing a Reinforcement Learner.. In *FLAIRS Conference*.

[21] Burrhus Frederic Skinner. 1938. The behavior of organisms: An experimental analysis. (1938).

[22] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Vol. 1631. Citeseer, 1642.

[23] Kaushik Subramanian, Charles L Isbell Jr, and Andrea L Thomaz. 2016. Exploration from demonstration for interactive reinforcement learning. In *Proceedings of the 2016 International Conference on AAMAS*. International Foundation for AAMAS, 447–456.

[24] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[25] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation.. In *AAAI*, Vol. 1. 2.

[26] Jesse Thomason, Shiqi Zhang, Raymond J Mooney, and Peter Stone. 2015. Learning to Interpret Natural Language Commands through Human-Robot Dialog.. In *IJCAI*. 1923–1929.

[27] Andrea L Thomaz and Cynthia Breazeal. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence* 172, 6-7 (2008), 716–737.