Interactive Learning from Policy-Dependent Human Feedback

James MacGlashan*, Mark K Ho†, Robert Loftin‡, Bei Peng§, David Roberts‡, Matthew E. Taylor§, Michael L. Littman†
*Cogitai, †Brown University, ‡North Carolina State University, §Washington State University

***Abstract*—For agents and robots to become more useful, they must be able to quickly learn from non-technical users. This paper investigates the problem of interactively learning behaviors communicated by a human teacher using positive and negative feedback. Much previous work on this problem has made the assumption that people provide feedback for decisions that is dependent on the behavior they are teaching and is independent from the learner's current policy. We present empirical results that show this assumption to be false—whether human trainers give a positive or negative feedback for a decision is influenced by the learner's current policy. We argue that policy-dependent feedback, in addition to being commonplace, enables useful training strategies from which agents should benefit. Based on this insight, we introduce *Convergent Actor-Critic by Humans* (COACH), an algorithm for learning from policy-dependent feedback that converges to a local optimum. Finally, we demonstrate that COACH can successfully learn multiple behaviors on a physical robot, even with noisy image features.**

I. Introduction

Programming robots is very difficult, in part because the real world is inherently rich and—to some degree—unpredictable. In addition, our expectations for physical agents are quite high and often difficult to articulate. Nevertheless, for robots to have a significant impact on the lives of individuals, even non-programmers need to be able to specify and customize behavior. Because of these complexities, relying on end-users to provide instructions to robots programmatically seems destined to fail.

Reinforcement learning (RL) from human trainer feedback provides a compelling alternative to programming because agents can learn complex behavior from very simple positive and negative signals. Furthermore, real-world animal training is an existence proof that people can train complex behavior using these simple signals. Indeed, animals have been successfully trained to guide the blind, locate mines in the ocean, detect cancer or explosives, and even solve complex, multi-stage puzzles.

However, traditional reinforcement-learning algorithms have yielded limited success when the reward signal is provided by humans and have largely failed to benefit from the sophisticated training strategies that expert animal trainers use with animals. This failure has led to the development of new reinforcement-learning algorithms that are designed to learn from human-generated rewards and investigations into how people give interactive feedback [1, 2, 3, 4, 5, 6]. In general, many of these human-centered learning algorithms are built on the insight that people tend to give feedback that reflects the policy the agent should be following, rather than as a numeric value that is meant to be maximized by the agent. While this insight seems accurate, existing approaches assume models of feedback that are independent of the policy the agent is currently following. We present empirical results that demonstrate that this assumption is incorrect and further argue that policy-dependent feedback enables effective training strategies, such as differential feedback and policy shaping, from which we would like a learning agent to benefit. Following this result, we present *Convergent Actor-Critic by Humans* (COACH), an algorithm for learning from policy-dependent human feedback that is capable of benefiting from these strategies. COACH is based on the insight that the TD-error used by actor-critic algorithms is an unbiased estimate of the *advantage function*, which is a policy-dependent value roughly corresponding to how much better or worse an action is compared to the current policy and which captures these previously mentioned training strategies. To validate that COACH scales to complex problems, we train five different behaviors on a TurtleBot robot that makes decisions every 33ms from noisy image features that are not visible to the trainer.

II. Background

For modeling the underlying decision-making problem of an agent being taught by a human, we adopt the Markov Decision Process (MDP) formalism. An MDP is a 5-tuple: $\langle S, A, T, R, \gamma \rangle$, where S is the set of possible states of the environment; A is the set of actions available to the agent; $T(s'|s,a)$ is the transition function, which defines the probability of the environment transitioning to state s' when the agent takes action a in environment state s; $R(s,a,s')$ is the reward function specifying the numeric reward the agent receives for taking action a in state s and transitioning to state s'; and $\gamma \in [0,1]$ is a discount factor specifying how much immediate rewards are preferred to more distant rewards.

A *stochastic policy* π for an MDP is a per-state action probability distribution that defines a mode of behavior; $\pi : S \times A \rightarrow [0,1]$, where $\sum_{a \in A} \pi(s,a) = 1, \forall s \in S$. In the MDP setting, the goal is to find the optimal policy π^*, which maximizes the expected future discounted reward when the agent selects actions in each state according to π^*; $\pi^* = \text{argmax}_\pi E[\sum_{t=0}^{\infty} \gamma^t r_t | \pi]$, where r_t is the reward received at time t. Two important concepts in MDPs are the value function (V^π) and action–value function (Q^π). The value function defines the expected future discounted reward from each state

when following some policy and the action–value function defines the expected future discounted reward when an agent takes some action in some state and then follows some policy π thereafter. These equations can be recursively defined via the Bellman equation: $V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a)$ and $Q^\pi(s,a) = \sum_{s'} T(s'|s,a)\left[R(s,a,s') + \gamma V^\pi(s')\right]$. For shorthand, the value functions for the optimal policies are usually denoted V^* and Q^*.

In reinforcement learning (RL), an agent interacts with an environment modeled as an MDP, but does not have direct access to the transition function or reward function and instead must learn a policy from environment observations. A common class of RL algorithms are *actor-critic* algorithms. Bhatnagar et al. [7] provides a general template for these algorithms. Actor-critic algorithms are named for the two main components of the algorithms: The actor is a parameterized policy that dictates how the agent selects actions; the critic estimates the value function for the actor and provides critiques at each time step that are used to update the policy parameters. Typically, the critique is the temporal difference (TD) error: $\delta_t = r_t + \gamma V(s_t) - V(s_{t-1})$, which describes how much better or worse a transition went than expected.

III. Human-centered Reinforcement Learning

In this work, a *human-centered reinforcement-learning* (HCRL) problem is a learning problem in which an agent is situated in an environment described by an MDP but in which rewards are generated by a human trainer instead of from a stationary MDP reward function that the agent is meant to maximize. The trainer has a target policy π^* they are trying to teach the agent. The trainer communicates this policy by giving numeric feedback as the agent acts in the environment. The goal of the agent is to learn the target policy π^* from the feedback.

To define a learning algorithm for this problem, we first characterize how human trainers typically use numeric feedback to teach target policies. If feedback is stationary and intended to be maximized, it can be treated as a reward function and standard RL algorithms used. Although this approach has had some success [8, 9], there are complications that limit its applicability. In particular, a trainer must take care that the feedback they give contains no unanticipated exploits, constraining the feedback strategies they can use. Indeed, prior research has shown that interpreting human feedback like a reward function often induces *positive reward cycles* that leads to unintended behaviors [10, 11].

The issues with interpreting feedback as reward have led to the insight that human feedback is better interpreted as a comment on the agent's behavior; for example, positive feedback roughly corresponds to "that was good" and negative feedback roughly corresponds to "that was bad." Existing HCRL work adopting this perspective includes TAMER [1], SABL [6], and Policy Shaping [5], discussed in more detail in the Related Work section. We note, though, that all assume that human feedback is independent of the agent's current policy.

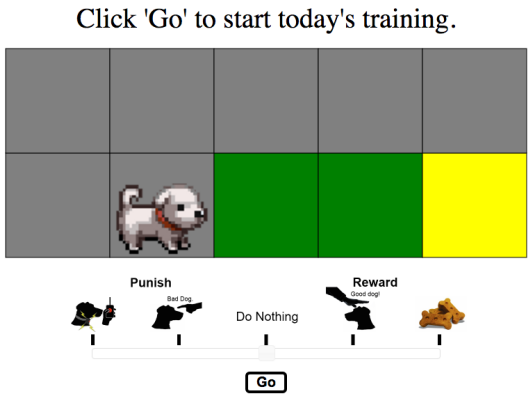

Fig. 1. The training interface shown to AMT users.

We provide empirical results that show this assumption to be incorrect.

IV. Policy-dependent Feedback

Evidence that human feedback is influenced by the agent's current policy can be seen in previous work. For example, it was observed that trainers taper their feedback over the course of learning [11, 12, 9]. One explanation for decreasing feedback rates is policy-dependent feedback, but trainer fatigue is another. We provide a stronger result showing that trainers—for the same state–action pair—choose positive or negative feedback depending on their perception of the learner's behavior. This finding serves as a warning for algorithms that rely on an assumption policy-independent feedback.

A. Empirical Results

We had Amazon Mechanical Turk (AMT) participants teach an agent in a simple sequential task, illustrated in Figure 1. Participants were instructed to train a virtual dog to walk to the yellow goal location in a grid world as fast as possible but without going through the green cells. They were additionally told that, as a result of prior training, their dog was already either "bad", "alright", or "good" at the task and were shown examples of each behavior before training. In all cases, the dog would start in the location shown in Figure 1. "Bad" dogs walked straight through the green cells to the yellow cell. "Alright" dogs first moved left, then up, and then to the goal, avoiding green but not taking the shortest route. "Good" dogs took the shortest path to yellow without going through green.

During training, participants saw the dog take an action from one tile to another and then gave feedback after every action using a continuous labeled slider as shown. The slider always started in the middle of the scale on each trial, and several points were labeled with different levels of reward (praise and treats) and punishment (scolding and a mild electric shock). Participants went through a brief tutorial using this interface. Responses were coded as a numeric value from -50 to 50, with "Do Nothing" as the zero-point.

During the training phase, participants trained a dog for three *episodes* that all started in the same position and ended

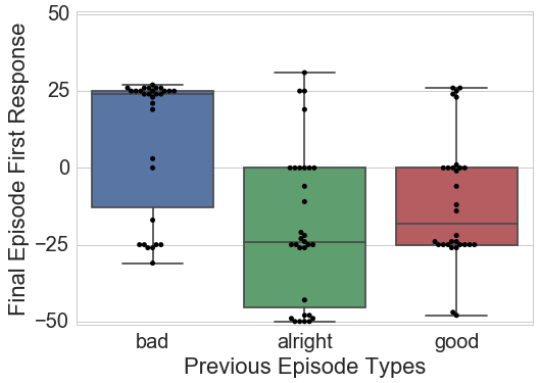

Fig. 2. Box plots with median, interquartile range, and minimum/maximum values of human responses to first step of the final episode for each condition. Feedback tended to be positive when the prior behavior was *bad* and negative otherwise.

at the goal. The dog's behavior was pre-programmed for all episodes in such a way that the first step of the final episode would reveal if feedback was policy-dependent. The dog always performed the same behavior in the first two episodes, and then performed the "alright" behavior in the third episode. Each user was placed into one of three different conditions that determined how the dog would behave in the first two episodes: either "bad," "alright," or "good." If feedback is policy dependent, we expect more positive feedback in the "bad" condition than in the "alright" or "good" condition; "alright" behavior is an improvement over the previous "bad" behavior whereas it is either no improvement or a deterioration compared to "alright" or "good" behavior.

Figure 2 shows boxplots and individual responses for the first step of the final episode under each of the three conditions. These results indicate that the sign of feedback is sensitive to the learner's policy, as predicted. The mean and median feedback under the "bad" condition is slightly positive (Mean = 9.8, Median = 24, S.D. = 22.2; planned Wilcoxon one-sided signed-rank test: $Z = 1.71, p < 0.05$), whereas it is negative for the "alright" condition (Mean = -18.3, Median = -23.5, S.D. = 24.6; planned Wilcoxon two-sided signed-rank test: $Z = -3.15, p < 0.01$) and "good" condition (Mean = -10.8, Median = -18.0, S.D. = 20.7; planned Wilcoxon one-sided signed-rank test: $Z = -2.33, p < 0.05$). There was a main effect across the three conditions ($p < 0.01$, Kruskal-Wallace Test), and pairwise comparisons indicated that only the "bad" condition differed from "alright" and "good" conditions ($p < 0.01$ for both, Bonferroni-corrected, Mann-Whitney Pairwise test).

B. Training Strategies

Beyond the fact that our previous evidence suggests that people give policy-dependent feedback, we argue that policy-dependent feedback affords desirable training strategies. Specifically, we consider three different feedback schemes that can be viewed as operationalizations of well-studied behavior analysis reinforcement schedules [13]: *Diminishing Returns*: gradually decrease positive feedback for good actions as the agent adopts those actions. *Differential Feedback*: vary the magnitude of feedbacks w.r.t. the degree of improvement or deterioration in behavior. *Policy Shaping*: provide positive feedback for suboptimal actions that improve behavior and then negative feedback after the improvement has been made.

Diminishing returns is a useful strategy because it decreases the burden of how actively a trainer must supply feedback and removes the need for explicit training and execution phases. Differential feedback is useful because it can serve to highlight the most important behaviors in the state space and communicate a kind of urgency for learning them. Finally, policy shaping concerns feedback that signals an improvement relative to the current baseline—as in the AMT study above. It is useful for providing a direction for the learner to follow at all times, even when the space of policies is continuous or otherwise impractical to search.

V. Convergent Actor-Critic by Humans

In this section, we introduce *Convergent Actor-Critic by Humans* (COACH), an actor-critic-based algorithm capable of learning from policy-dependent feedback. COACH is based on the insight that the *advantage function* is a good model of human feedback and that actor-critic algorithms update a policy using the critic's TD error, which is an unbiased estimate of the advantage function. Consequently, an agent's policy can be directly modified by human feedback without a critic component. We first define the advantage function and describe how it relates to the three previously mentioned training strategies. Then, we present the general update rule for COACH and its convergence. Finally, we present *Real-time COACH*, which includes mechanisms for providing variable magnitude feedback and learning in problems with a high-frequency decision cycle.

A. The Advantage Function and Training Strategies

The advantage function [14] A^π is defined as

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{1}$$

Roughly speaking, the advantage function describes how much better or worse an action selection is compared to the agent's performance following policy π. We now show that feedback assigned according to the advantage function follows the patterns of all three training strategies. Showing that the advantage function captures the differential feedback strategy is trivial, because the advantage function is defined by how much better taking an action over its current policy is expected to be.

To show that the advantage function induces diminishing returns, consider what happens when the learner improves its behavior by shifting to a higher scoring action a in some state s. As its probability of selecting a goes to 1, $A^\pi(s,a) = Q^\pi(s,a) - Q^\pi(s,a) = 0$, because the value function $V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a)$. Since the expected value is a smooth linear combination of the Q-values, as the

agent adopts action a, $A^\pi(s,a) \to 0$, resulting in gradually decreasing feedback.

To show that the advantage function induces policy shaping, let us first assume w.l.o.g. that there are three possible actions, where action a_3 is exclusively optimal, and action a_2 is better than action a_1. That is, $Q^*(s,a_3) > Q^*(s,a_2) > Q^*(s,a_1)$. For illustrative purposes, let us also assume that the agent has learned the optimal policy in all states except state s, wherein the agent with near-certain probability selects the worst action a_1 ($\pi(s,a_1) \approx 1$) and that all actions lead to some other state. In this scenario, $A^\pi(s,a) = Q^*(s,a) - \sum_{a'} \pi(s,a')Q^*(s,a') \approx Q^*(s,a) - Q^*(s,a_1)$. Since $Q^*(s,a_2) > Q^*(s,a_1)$, it follows that $A^\pi(s,a_2)$ is positive. Consequently, in this condition, suboptimal action a_2 would receive positive feedback.

Now consider the case when the agent with near-certainty selects optimal action a_3. Under this condition, $A^\pi(s,a) \approx Q^*(s,a) - Q^*(s,a_3)$ and since $Q^*(s,a_3) > Q^*(s,a_2)$, $A^\pi(s,a_2)$ is negative. Once again, since V^π is a smooth linear function of Q^π, as the agent adopts optimal action a_3, $A^\pi(s,a_2)$ becomes negative. Therefore, suboptimal actions can be rewarded and then punished as the agent improves at the task, producing policy shaping.

B. Convergence and Update Rule

Given a performance metric ρ, Sutton et al. [15] derive a policy gradient algorithm of the form: $\Delta\boldsymbol{\theta} = \alpha\nabla_{\boldsymbol{\theta}}\rho$. Here, $\boldsymbol{\theta}$ represents the parameters that control the agent's behavior and α is a learning rate. Under the assumption that ρ is the discounted expected reward from a fixed start state distribution, they show that

$$\nabla_{\boldsymbol{\theta}}\rho = \sum_s d^\pi(s) \sum_a \nabla_{\boldsymbol{\theta}}\pi(s,a)Q^\pi(s,a),$$

where $d^\pi(s)$ is the component of the (discounted) stationary distribution at s. A benefit of this form of the gradient is that, given that states are visited according to $d^\pi(s)$ and actions are taken according to $\pi(s,a)$, the update at time t can be made as:

$$\Delta\boldsymbol{\theta}_t = \alpha_t \nabla_{\boldsymbol{\theta}}\pi(s_t,a_t)\frac{f_{t+1}}{\pi(s_t,a_t)}, \tag{2}$$

where $E[f_{t+1}] = Q^\pi(s_t,a_t) - v(s)$ for any action-independent function $v(s)$.

In the context of the present paper, f_{t+1} represents the feedback provided by the trainer. It follows trivially that if the trainer chooses the policy-dependent feedback $f_t = Q^\pi(s_t,a_t)$, we obtain a convergent learning algorithm that (locally) maximizes discounted expected reward. In addition, feedback of the form $f_t = Q^\pi(s_t,a_t) - V^\pi(s_t) = A^\pi(s,a)$ also results in convergence. Note that for the trainer to provide feedback in the form of Q^π or A^π, they would need to "peer inside" the learner and observe its policy. In practice, the trainer estimates π by observing the agent's actions.

Algorithm 1 Real-time COACH

Input: policy $\pi_{\boldsymbol{\theta}_0}$, trace set $\boldsymbol{\lambda}$, delay d, learning rate α
  Initialize traces $\boldsymbol{e}_\lambda \leftarrow \mathbf{0}\ \forall\lambda \in \boldsymbol{\lambda}$
  observe initial state s_0
  for $t = 0$ to ∞ **do**
    select and execute action $a_t \sim \pi_{\boldsymbol{\theta}_t}(s_t,\cdot)$
    observe next state s_{t+1}, sum feedback f_{t+1}, and λ
    for $\lambda' \in \boldsymbol{\lambda}$ **do**
      $\boldsymbol{e}_{\lambda'} \leftarrow \lambda'\boldsymbol{e}_{\lambda'} + \frac{1}{\pi_{\boldsymbol{\theta}_t}(s_{t-d},a_{t-d})}\nabla_{\boldsymbol{\theta}_t}\pi_{\boldsymbol{\theta}_t}(s_{t-d},a_{t-d})$
    end for
    $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha f_{t+1}\boldsymbol{e}_\lambda$
  end for

C. Real-time COACH

There are challenges in implementing Equation 2 for real-time use in practice. Specifically, the interface for providing variable magnitude feedback needs to be addressed, and the question of how to handle sparseness and the timing of feedback needs to be answered. Here, we introduce *Real-time COACH*, shown in Algorithm 1, to address these issues.

For providing variable magnitude reward, we use reward aggregation [1]. In reward aggregation, a trainer selects from a discrete set of feedback values and further raises or lowers the numeric value by giving multiple feedbacks in succession that are summed together.

While sparse feedback is not especially problematic (because no feedback results in no change in policy), it may slow down learning unless the trainer is provided with a mechanism to allow feedback to affect a history of actions. We use *eligibility traces* [16] to help apply feedback to the relevant transitions. An eligibility trace is a vector that keeps track of the policy gradient and decays exponentially with a parameter λ. Policy parameters are then updated in the direction of the trace, allowing feedback to affect earlier decisions. However, a trainer may not always want to influence a long history of actions. Consequently, real-time COACH maintains multiple eligibility traces with different temporal decay rates and the trainer chooses which eligibility trace to use. This trace choice may be handled implicitly with the feedback value selection or explicitly.

Due to reaction time, human feedback is typically delayed by about 0.2 to 0.8 seconds from the event to which they meant to give feedback [10]. To handle this delay, feedback in Real-time COACH is associated with events from d steps ago to cover the gap. Eligibility traces further smooth the feedback to older events.

Finally, we note that just as there are numerous variants of actor-critic update rules, similar variations can be used in the context of COACH.

VI. Related Work

An inspiration for our work is the TAMER framework [10]. In TAMER, trainers provide interactive numeric feedback as the learner takes actions. The feedback is interpreted as an exemplar of the reward function for the previous state–action

pair and is used to learn the reward function. When the agent makes rapid decisions, TAMER divides the feedback among the recent state–action pairs according to a probability distribution. TAMER makes decisions by myopically choosing the action with the highest reward estimate. Because the agent myopically maximizes reward, the feedback may be interpreted as exemplars of Q^*. Later work investigated non-myopically maximizing the learned reward function with a planning algorithm [17], but this approach requires a model of the environment and special treatment of termination conditions. Because TAMER expects feedback to be policy independent, it does not support the diminishing returns or policy-shaping strategies, and handles differential feedback only in so far as it uses numeric feedback. In our robotics case study, we provide an explicit example where TAMER's failure to support diminishing returns can result in unlearning.

Two other closely related approaches are SABL [6] and Policy Shaping [5] (unrelated to the policy shaping feedback *strategy* defined above). Both of these approaches treat feedback as discrete probabilistic evidence of a parametrized policy. SABL's probabilistic model additionally includes (learnable) parameters for describing how often a trainer is expected to give explicit positive or negative feedback. Both of these approach assume policy-independent feedback and do not support the three training strategies described.

There have also been some domains in which treating human feedback as reward signals to maximize has had some success, such as in shaping the control for a prosthetic arm [8] and learning how to interact in an online chat room from multiple users' feedback [9]. An interesting area of future work is to test whether performance in these domains can be improved with COACH given our insights into the nature of human feedback—work on the chat-room learning domain did report challenges due to diminishing feedback.

Some research has examined combining human feedback with more traditional environmental rewards [18, 19, 20, 21]. A challenge in this context in practice is that rewards do not naturally come from the environment and must be programmatically defined. However, it is appealing because the agent can learn in the absence of an active trainer. We believe that COACH could be straightforwardly modified to learn in this setting as well.

Finally, a related research area is *learning from demonstration* (LfD), in which a human provides examples of the desired behavior. There are a number of different approaches to solving this problem surveyed by Argall et al. [22]. We see these approaches as complementary to HCRL because it is not always possible, or convenient, to provide demonstrations. LfD approaches that learn a parametrized policy could also operate with COACH, allowing the agent to have their policy seeded by demonstrations, and then fine tuned with interactive feedback.

VII. Robotics Case Study

In this section, we present qualitative results applying Real-time COACH to a TurtleBot robot. The goal of this study was to test that COACH can scale to a complex domain involving multiple challenges, including training an agent that operates on a fast decision cycle (33ms), noisy non-Markov observations from a camera, and agent perception that is hidden from the trainer. To demonstrate the flexibility of COACH, we trained it to perform five different behaviors involving a pink ball and cylinder with an orange top using the same parameter selections. We discuss these behaviors below. We also contrast the results to training with TAMER. We chose TAMER as a comparison because, to our knowledge, it is the only HCRL algorithm with success on a similar platform [23].

The TurtleBot is a mobile base with two degrees of freedom that senses the world from a Kinect camera. We discretized the action space to five actions: forward, backward, rotate clockwise, rotate counterclockwise, and do nothing. The agent selects one of these actions every 33ms. To deliver feedback, we used a Nintendo Wii controller to give $+1$, $+4$, or -1 numeric feedback, and pause and continue training. For perception, we used only the RGB image channels from the Kinect. Because our behaviors were based around a relocatable pink ball and a fixed cylinder with an orange top, we hand constructed relevant image features to be used by the learning algorithms. These features were generated using techniques similar to those used in neural network architectures. In the future, we will investigate learning these features along with the policy. The features were constructed by first transforming the image into two color channels associated with the color of the ball and cylinder. Sum pooling to form a lower-dimensional 8×8 grid was applied to each color channel. Each sum-pooling unit was then passed through three different normalized threshold units defined by $T_i(x) = \min(\frac{x}{\phi_i}, 1)$, where ϕ_i specifies how quickly the ith threshold unit saturates. Using multiple saturation parameters differentiates the distance of objects, resulting in three "depth" scales per color channel. Finally, we passed these results through a 2×8 max-pooling layer with stride 1.

The five behaviors we trained were push–pull, hide, ball following, alternate, and cylinder navigation. In push–pull, the TurtleBot is trained to navigate toward the ball when it is far, and back away from it when it is near. The hide behavior has the TurtleBot back away from the ball when it is near and turn away from it when it is far. In ball following, the TurtleBot is trained to navigate to the ball. In the alternate task, the TurtleBot is trained to go back and forth between the cylinder and ball. Finally, cylinder navigation involves the agent navigating to the cylinder. We further classify training methods for each of these behaviors as *flat*, involving the push–pull, hide, and ball following behaviors; and *compositional*, involving the alternate and cylinder navigation behaviors.

In all cases, our human trainer (one of the co-authors) used differential feedback and diminishing returns to quickly reinforce behaviors and restrict focus to the areas needing tuning. However, in alternate and cylinder navigation, they attempted more advanced compositional training methods. For alternate, the agent was first trained to navigate to the ball when it sees it, and then turn away when it is near. Then,

the same was independently done for the cylinder. After training, introducing both objects would cause the agent to move back and forth between them. For cylinder navigation, they attempted to make use of an animal-training method called *lure training* in which an animal is first conditioned to follow a lure object that is then used to guide it through more complex behaviors. In cylinder navigation, they first trained the ball to be a lure, used it to guide the TurtleBot to the cylinder, and finally gave a $+4$ reward to reinforce the behaviors it took when following the ball (turning to face the cylinder, moving toward it, and stopping upon reaching it). The agent would then navigate to the cylinder without requiring the ball to be present.

For COACH parameters, we used a softmax parameterized policy, where each action preference value was a linear function of the image features, plus $\tanh(\theta_a)$, where θ_a is a learnable parameter for action a, providing a preference in the absence of any stimulus. We used two eligibility traces with $\lambda = 0.95$ for feedback $+1$ and -1, and $\lambda = 0.9999$ for feedback $+4$. The feedback-action delay d was set to 6, which is 0.198 seconds. Additionally, we used an actor-critic parameter-update rule variant in which action preference values are directly modified (along its gradient), rather than by the gradient of the policy [24]. This variant more rapidly communicates stimulus–response preferences. For TAMER, we used typical parameter values for fast decision cycle problems: delay-weighted aggregate TAMER with uniform distribution credit assignment over 0.2 to 0.8 seconds, $\epsilon_p = 0$, and $c_{min} = 1$ [10]. (See prior work for parameter meaning.) For TAMER's reward function approximation, we used the same parameters as the action preferences in COACH.

A. Results and Discussion

COACH was able to successfully learn all five behaviors and a video showing its learning is available online at https://vid.me/3h2s. Each of these behaviors were trained in less than two minutes, including the time spent verifying that a behavior worked. Differential feedback and diminishing returns allowed only the behaviors in need of tuning to be quickly reinforced or extinguished without any explicit division between training and testing. Moreover, the agent successfully benefited from the compositional training methods, correctly combining subbehaviors for alternate, and quickly learning cylinder navigation with the lure.

TAMER only successfully learned the behaviors using the flat training methodology and failed to learn the compositionally trained behaviors. In all cases, TAMER tended to forget behavior, requiring feedback for previous decisions it learned to be resupplied after it learned a new decision. For the alternate behavior, this forgetting led to failure: after training the behavior for the cylinder, the agent forgot some of the ball-related behavior and ended up drifting off course when it was time to go to the ball. TAMER also failed to learn from lure training, which was expected since TAMER does not allow reinforcing a long history of behaviors.

We believe TAMER's forgetting is a result of interpreting feedback as reward-function exemplars in which new feedback in similar contexts can change the target. To illustrate this problem, we constructed a well-defined scenario in which TAMER consistently unlearns behavior. In this scenario, the goal was for the TurtleBot to always stay whenever the ball was present, and move forward if just the cylinder was present. We first trained TAMER to stay when the ball alone was present using many rapid rewards (yielding a large aggregated signal). Next, we trained it to move forward when the cylinder alone was present. We then introduced both objects, and the TurtleBot correctly stayed. After rewarding it for staying with a single reward (weaker than the previously-used many rapid rewards), the TurtleBot moved forward. This counter-intuitive unlearning is a consequence of the small reward decreasing its reward-function target for the stay action to a point lower than the value for moving forward. COACH does not exhibit this problem—any reward for staying will strengthen the behavior.

VIII. Conclusion

In this work, we presented empirical results that show that the numeric feedback people give agents in an interactive training paradigm is influenced by the agent's current policy and argued why such policy-dependent feedback enables useful training strategies. We then introduced COACH, an algorithm that, unlike existing human-centered reinforcement-learning algorithms, converges to a local optimum when trained with policy-dependent feedback. We finally we showed that COACH scales up in the context of a robotics case study in which a TurtleBot was successfully taught multiple behaviors with advanced training methods.

There are a number of exciting future directions to extend this work. In particular, because COACH is built on the actor-critic paradigm, it should be possible to combine it straightforwardly with learning from demonstration and environmental rewards, allowing an agent to be trained in a variety of ways. Second, because people give policy-dependent feedback, greater gains may be possible by investigating how people model the current policy of the agent and how their model differs from the agent's actual policy.

References

[1] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The tamer framework," in *Proceedings of the fifth international conference on Knowledge capture*. ACM, 2009, pp. 9–16.

[2] A. L. Thomaz and C. Breazeal, "Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance," in *AAAI*, vol. 6, 2006, pp. 1000–1005.

[3] A. Thomaz and C. Breazeal, "Robot learning via socially guided exploration," in *ICDL Development on Learning*, 2007.

[4] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," 2008.

[5] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," in *Advances in Neural Information Processing Systems*, 2013, pp. 2625–2633.

[6] R. Loftin, B. Peng, J. MacGlashan, M. L. Littman, M. E. Taylor, J. Huang, and D. L. Roberts, "Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 1, pp. 30–59, 2015.

[7] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor–critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[8] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton, "Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning," in *2011 IEEE International Conference on Rehabilitation Robotics*. IEEE, 2011, pp. 1–7.

[9] C. Isbell, C. R. Shelton, M. Kearns, S. Singh, and P. Stone, "A social reinforcement learning agent," in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 377–384.

[10] W. B. Knox, "Learning from human-generated reward," Ph.D. dissertation, University of Texas at Austin, 2012.

[11] M. K. Ho, M. L. Littman, F. Cushman, and J. L. Austerweil, "Teaching with rewards and punishments: Reinforcement or communication?" in *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, 2015.

[12] W. B. Knox, B. D. Glass, B. C. Love, W. T. Maddox, and P. Stone, "How humans teach agents," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 409–421, 2012.

[13] R. G. Miltenberger, *Behavior modification: Principles and procedures*. Cengage Learning, 2011.

[14] L. Baird *et al.*, "Residual algorithms: Reinforcement learning with function approximation," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 30–37.

[15] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation." in *NIPS*, vol. 99, 1999, pp. 1057–1063.

[16] A. Barto, R. Sutton, and C. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-13, no. 5, pp. 834 –846, sept.-oct. 1983.

[17] W. B. Knox and P. Stone, "Learning non-myopically from human-generated reward," in *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 2013, pp. 191–202.

[18] B. Knox and P. Stone, "Combining manual feedback with subsequent MDP reward signals for reinforcement learning," in *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2010.

[19] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Villaseñor-Pineda, "Dynamic reward shaping: training a robot by voice," in *Advances in Artificial Intelligence–IBERAMIA 2010*. Springer, 2010, pp. 483–492.

[20] J. A. Clouse and P. E. Utgoff, "A teaching method for reinforcement learning," in *Proceedings of the Ninth International Conference on Machine Learning (ICML92)*, 1992, pp. 92–101.

[21] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild, "Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression," in *Proceedings of the National Conference on Artificial intelligence*, vol. 20, no. 2, 2005, p. 819.

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

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

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.