Interactive Learning from Policy-Dependent Human Feedback

James MacGlashan [1] Mark K Ho [2] Robert Loftin [3] Bei Peng [4] Guan Wang [2] David L. Roberts [3]
Matthew E. Taylor [4] Michael L. Littman [2]

Abstract

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. 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.

1. 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.

Despite success when learning from environmental reward, traditional reinforcement-learning algorithms have yielded limited success when the reward signal is provided by humans. This failure underscores the importance that algorithms for learning from humans are based on appropriate models of human-feedback. Indeed, much human-centered RL work has investigated and employed different models of human-feedback (Knox & Stone, 2009b; Thomaz & Breazeal, 2006; 2007; 2008; Griffith et al., 2013; Loftin et al., 2015). Many of these algorithms leverage the observation that people tend to give feedback that is best interpreted as guidance on the policy the agent should be following, rather than as a numeric value to be maximized by the agent. However, these 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 demonstrate cases in which policy-independent learning algorithms suffer from this assumption. Following this result, we present *Convergent Actor-Critic by Humans* (COACH), an algorithm for learning from policy-dependent human feedback. COACH is based on the insight that the *advantage function* (a value roughly corresponding to how much better or worse an action is compared to the current policy) provides a better model of human feedback, capturing human-feedback properties like diminishing returns, rewarding improvement, and giving 0-valued feedback a semantic meaning that combats forgetting. We compare COACH to other approaches in a simple domain with simulated feedback. Then, to validate that COACH scales to complex problems, we train five different behaviors on a TurtleBot robot.

2. 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

[*]Equal contribution [1]Cogitai [2]Brown University [3]North Carolina State University [4]Washington State University. Correspondence to: James MacGlashan <james@cogitai.com>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

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 an agent's 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. (2009) provide 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.

3. 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 (Pilarski et al., 2011; Isbell et al., 2001), 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 lead to unintended behaviors (Knox, 2012; Ho et al., 2015).

The issues with interpreting feedback as reward have led to the insight that human feedback is better interpreted as commentary on the agent's behavior; for example, positive feedback roughly corresponds to "that was good" and negative feedback roughly corresponds to "that was bad." In the next section, we review existing HCRL approaches that build on this insight.

4. Related Work

A number of existing approaches to HCRL and RL that includes human feedback has been explored in the past. The most similar to ours, and a primary inspiration for this work, is the TAMER framework (Knox, 2012). In TAMER, trainers provide interactive numeric feedback as the learner takes actions. The learner attempts to estimate a target reward function by interpreting trainer feedback as exemplars of this 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 can also be thought of as exemplars of Q^*. Later work also investigated non-myopically maximizing the learned reward function with a planning algorithm (Knox & Stone, 2013), but this approach requires a model of the environment and special treatment of termination conditions.

Two other closely related approaches are SABL (Loftin et al., 2015) and Policy Shaping (Griffith et al., 2013). Both of these approaches treat feedback as discrete probabilistic evidence of the trainer's target parameterized policy. SABL's probabilistic model additionally includes (learnable) parameters for describing how often a trainer is expected to give explicit positive or negative feedback.

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 (Pilarski et al., 2011) and learning how to interact in an online chat room from multiple users' feedback (Isbell et al., 2001). Some complications with how people give feedback have been reported, however.

Some research has also examined combining human feedback with more traditional environmental rewards (Knox & Stone, 2010; Tenorio-Gonzalez et al., 2010; Clouse & Utgoff, 1992; Maclin et al., 2005). 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 our approach to HCRL could also straightforwardly incorporate learning from environmental reward as well, but we leave this investigation for future work.

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. (2009). We see these approaches as complementary to HCRL because it is not always possible, or convenient, to provide demonstrations. LfD approaches that learn a parameterized policy could also operate with COACH, allowing the agent to have their policy seeded by demonstrations, and then fine tuned with interactive feedback.

Note that the policy-dependent feedback we study here is viewed as essential in behavior analysis reinforcement schedules (Miltenberger, 2011). Trainers are taught to provide diminishing returns (gradual decreases in positive feedback for good actions as the agent adopts those actions), differential feedback (varied magnitude of feedbacks depending on the degree of improvement or deterioration in behavior), and policy shaping (positive feedback for suboptimal actions that improve behavior and then negative feedback after the improvement has been made), all of which are policy dependent.

5. Policy-dependent Feedback

A common assumption of existing HCRL algorithms is that feedback depends only on the quality of an agent's action selection. An alternative hypothesis is that feedback also depends on the agent's current policy. That is, an action selection may be more greatly rewarded or punished depending on how often the agent would typically be inclined to select it. For example, more greatly rewarding the agent for improving its performance than maintaining the status quo. We call the former model of feedback *policy-independent* and the latter *policy-dependent*. If people are more naturally inclined toward one model of feedback, algorithms

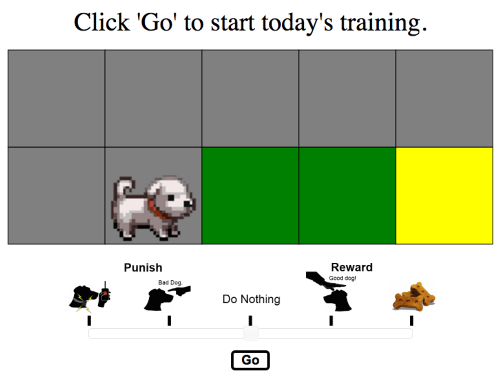


Figure 1. The training interface shown to AMT users.

based on the wrong assumption may result in unexpected responses to feedback. Consequently, we were interested in investigating which model better fits human feedback.

Despite existing HCRL algorithms assuming policy-independent feedback, evidence of policy-dependent feedback can be found in prior works with these algorithms. For example, it was often observed that trainers taper their feedback over the course of learning (Ho et al., 2015; Knox et al., 2012; Isbell et al., 2001). Although diminishing feedback is a property that is explained by people's feedback being policy-dependent—as the learner's performance improves, trainer feedback is decreased—an alternative explanation is simply trainer fatigue. To further make the case for human feedback being policy dependent, 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.

5.1. 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 at the goal. The dog's behavior was pre-programmed in such a way that the first step of the final episode would reveal if feedback was policy dependent. Each user was placed into one of three different conditions: improving, steady, or degrading. For all three conditions, the dog's behavior in the final episode was "alright," regardless of any prior feedback. The conditions differed in terms of the behavior users observed in the first two episodes. In the first two episodes, users observed bad behavior in the improving condition (improving to alright); alright behavior in the steady condition; and good behavior in the degrading condition. If feedback is policy-dependent, we would expect more positive feedback in the final episode for the improving condition, but not for policy-independent feedback since it was the same final behavior for all conditions.

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 improving 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 steady 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 degrading 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 improving condition differed from steady and degrading conditions ($p < 0.01$ for both, Bonferroni-corrected, Mann-Whitney Pairwise test).

6. 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 its interpretation as trainer feedback. Then,

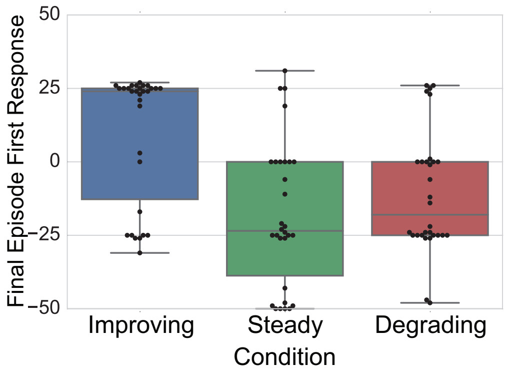


Figure 2. The feedback distribution for first step of the final episode for each condition. Feedback tended to be positive for improving behavior, but negative otherwise.

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.

6.1. The Advantage Function and Feedback

The advantage function (Baird, 1995) A^π is defined as

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

Roughly speaking, the advantage function describes how much better or worse an action selection is compared to the agent's performance under policy π. The function is closely related to the update used in policy iteration (Puterman, 1994): defining $\pi'(s) = \operatorname{argmax}_a A^\pi(s,a)$ is guaranteed to produce an improvement over π whenever π is suboptimal. It can also be used in policy gradient methods to gradually improve the performance of a policy, as described later.

It is worth nothing that feedback produced by the advantage function is consistent with that recommended in behavior analysis. It trivially results in differential feedback since it is defined as the magnitude of improvement of an action over its current policy. It induces diminishing returns because, as π improves opportunities to improve on it decrease. Indeed, once π is optimal, all advantage-function-based feedback is zero or negative. Finally, advantage function feedback induces policy shaping in that whether feedback is positive or negative for an action depends on whether it is a net improvement over the current behavior.

6.2. Convergence and Update Rule

Given a performance metric ρ, Sutton et al. (1999) 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:

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

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 $f_t = Q^\pi(s_t,a_t) - V^\pi(s_t) = A^\pi(s_t,a_t)$ 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.

6.3. 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 (Knox & Stone, 2009b). 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* (Barto et al., 1983) 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 for each update. This trace choice may be handled implicitly with the feedback value selection or explicitly.

Algorithm 1 Real-time COACH

```
Require: policy $\pi_{\boldsymbol{\theta}_0}$, trace set $\boldsymbol{\lambda}$, delay $d$, learning rate $\alpha$
  Initialize traces $\boldsymbol{e}_\lambda \leftarrow \boldsymbol{0}\ \forall\lambda \in \boldsymbol{\lambda}$
  observe initial state $s_0$
  for $t = 0$ to $\infty$ 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 $\lambda$
    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
```

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 (Knox, 2012). 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.

7. Comparison of Update Rules

To understand the behavior of COACH under different types of trainer feedback strategies, we carried out a controlled comparison in a simple grid world. The domain is essentially an expanded version of the dog domain used in our human-subject experiment. It is a 8×5 grid in which the agent starts in $0,0$ and must get to $7,0$, which yields $+5$ reward. However, from $1,0$ to $6,0$ are cells the agent needs to avoid, which yield -1 reward.

7.1. Learning Algorithms and Feedback Strategies

Three types of learning algorithms were tested. Each maintains an internal data structure, which it updates with feedback of the form $\langle s,a,f,s'\rangle$, where s is a state, a is an action taken in that state, f is the feedback received from the trainer, and s' is the resulting next state. The algorithm also must produce an action for each state encountered.

The first algorithm, Q learning (Watkins & Dayan, 1992), represents a standard value-function-based RL algorithm designed for reward maximization under delayed feedback. It maintains a data structure $Q(s,a)$, initially 0. Its update rule has the form:

$$\Delta Q(s,a) = \alpha[f + \gamma \max_{a'} Q(s',a') - Q(s,a)]. \quad (3)$$

Actions are chosen using the rule: $\mathrm{argmax}_a\, Q(s,a)$, where ties are broken randomly. We tested a handful of parameters and used the best values: discount factor $\gamma = 0.99$ and learning rate $\alpha = 0.2$.

In TAMER (Knox & Stone, 2009a), a trainer provides interactive numeric feedback that is interpreted as an exemplar of the reward function for the demonstrated state–action pair as the learner takes actions. We assumed that each feedback applies to the last action, and thus used a simplified version of the algorithm that did not attempt to spread updates over multiple transitions. TAMER maintains a data structure $R_H(s,a)$ for the predicted reward in each state, initially 0. It is updated by: $\Delta R_H(s,a) = \alpha f$. We used $\alpha = 0.2$. Actions are chosen via an ϵ-greedy rule on $R_H(s,a)$ with $\epsilon = 0.2$.

Lastly, we examined COACH, which is also designed to work well with human-generated feedback. We used a softmax policy with a single $\lambda = 0$ trace. The parameters were a matrix of values $\theta(s,a)$, initially zero. The stochastic policy defined by these parameters was

$$\pi(s,a) = e^{\beta\theta(s,a)} / \sum_a e^{\beta\theta(s,a)},$$

with $\beta = 1$. Parameters were updated via

$$\Delta\boldsymbol{\theta} = \alpha \nabla_{\boldsymbol{\theta}} \pi(s,a) \frac{f}{\pi(s,a)}, \quad (4)$$

where α is a learning rate. We used $\alpha = 0.05$.

In effect, each of these learning rules makes an assumption about the kind of feedback it expects trainers to use. We wanted to see how they would behave with feedback strategies that matched these assumptions and those that did not. The first feedback strategy we studied is the classical task-based reward function ("task") where the feedback is sparse: $+5$ reward when the agent reaches the goal state, -1 for avoidance cells, and 0 for all other transitions. Q-learning is known to converge to optimal behavior with this type of feedback. The second strategy provides policy-independent feedback for each state–action pair ("action"): $+5$ when the agent reaches termination, $+1$ reward when the selected action matches an optimal policy, -1 for reaching an avoidance cell, and 0 otherwise. This type of feedback serves TAMER well. The third strategy ("improvement") used feedback defined by the advantage function of the learner's current policy π, $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, where the value functions are defined based on the task rewards. This type of feedback is very well suited to COACH.

7.2. Results

Each combination of algorithm and feedback strategy was run 99 times with the median value of the number of steps needed to reach the goal reported. Episodes were ended after $1,000$ steps if the goal was not reached.

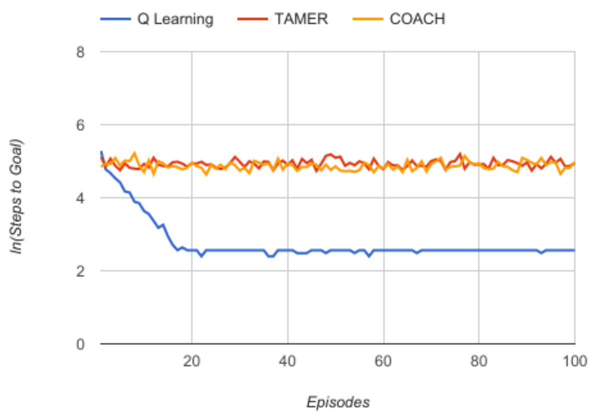
Figure 3(a) shows the steps needed to reach the goal for the three algorithms trained with task feedback. The figure shows that TAMER can fail to learn in this setting. COACH also performs poorly with $\lambda = 0$, which prevents feedback from influencing earlier decisions. We did a subsequent experiment (not shown) with $\lambda = 0.9$ and found that COACH converged to reasonable behavior, although not as quickly as Q learning. This result helps justify using traces to combat the challenges of delayed feedback.

Figure 3(b) shows results with action feedback. This time, Q learning fails to perform well, a consequence of this feedback strategy inducing positive behavior cycles as it tries to avoid ending the trial, the same kind of problem that HCRL algorithms have been designed to avoid. Both TAMER and COACH perform well with this feedback strategy. TAMER performs slightly better than COACH, as this is precisely the kind of feedback TAMER was designed to handle.

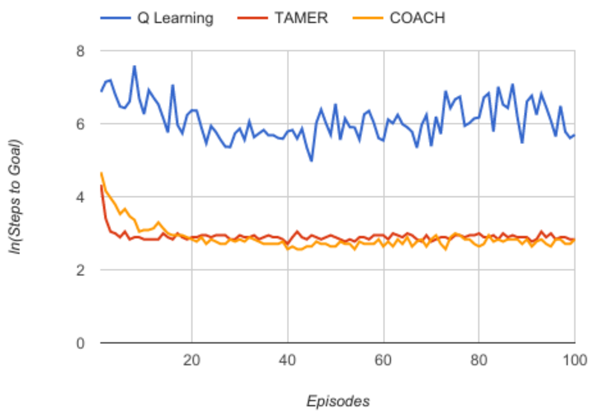
Figure 3(c) shows the results of the three algorithms with improvement feedback, which is generated via the advantage function defined on the learner's current policy. These results tells a different story. Here, COACH performs the best. Q-learning largely flounders for most of the time, but with enough training sometimes start to converge. (Although, 14% of the time, Q learning fails to do well even after 100 training episodes). TAMER, on the other hand, performs very badly at first. While the median score in the plot shows TAMER suddenly performing more comparably to COACH after about 10 episodes, 29% of our training trials completely failed to improve and timed-out across all 100 episodes.

8. Robotics Case Study

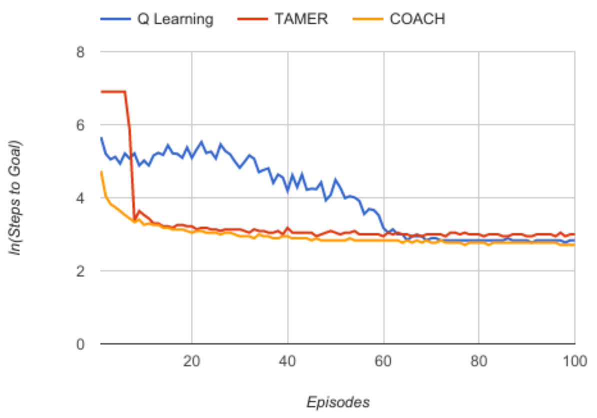
In this section, we present qualitative results on Real-time COACH applied to a TurtleBot robot. The goal of this study was to test that COACH can scale to a complex domain involving multiple challenges, including training an agent that operates on a fast decision cycle (33ms), noisy non-Markov observations 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 (Knox et al., 2013).



(a) Task feedback



(b) Action feedback



(c) Improvement feedback

Figure 3. Steps to goal for Q learning (blue), TAMER (red), and COACH (yellow) in Cliff world under different feedback strategies. The y-axis is on a logarithmic scale.

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. The features were constructed by first transforming the image into two color channels associated with the colors 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 the saturation point. 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 to 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, which 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 (Sutton & Barto, 1998). 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 $c_{min} = 1$ (Knox, 2012). (See prior work for parameter meaning.) TAMER's reward-function approximation used the same representation as COACH.

8.1. 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 because 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 responded by moving forward—the positive feedback actually caused it to unlearn the rewarded behavior. This counter-intuitive response 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. Roughly, because TAMER does not treat zero reward as special, a positive reward can be a negative influence if it is less than expected. COACH does not exhibit this problem—any positive reward for staying will strengthen the behavior.

9. 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 showed that COACH learns robustly in the face of multiple feedback strategies and finally showed that COACH can be used in the context of robotics 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, investigating how people model the current policy of the agent and how their model differs from the agent's actual policy may produce even greater gains.

Acknowledgements

We thank the anonymous reviewers for their useful suggestions and comments. This research has taken place in part at the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL's support includes NASA NNX16CD07C, NSF IIS-1149917, NSF IIS-1643614, and USDA 2014-67021-22174.

References

Argall, Brenna D, Chernova, Sonia, Veloso, Manuela, and Browning, Brett. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57 (5):469–483, 2009.

Baird, Leemon. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the twelfth international conference on machine learning*, pp. 30–37, 1995.

Barto, A.G., Sutton, R.S., and Anderson, C.W. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-13(5):834 –846, sept.-oct. 1983.

Bhatnagar, Shalabh, Sutton, Richard S, Ghavamzadeh, Mohammad, and Lee, Mark. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

Clouse, Jeffery A and Utgoff, Paul E. A teaching method for reinforcement learning. In *Proceedings of the Ninth International Conference on Machine Learning (ICML92)*, pp. 92–101, 1992.

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

Ho, Mark K, Littman, Michael L., Cushman, Fiery, and Austerweil, Joseph L. Teaching with rewards and punishments: Reinforcement or communication? In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, 2015.

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

Knox, W Bradley and Stone, Peter. Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the Fifth International Conference on Knowledge Capture*, pp. 9–16, 2009a.

Knox, W Bradley and Stone, Peter. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16. ACM, 2009b.

Knox, W Bradley and Stone, Peter. Learning non-myopically from human-generated reward. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pp. 191–202. ACM, 2013.

Knox, W Bradley, Glass, Brian D, Love, Bradley C, Maddox, W Todd, and Stone, Peter. How humans teach agents. *International Journal of Social Robotics*, 4(4): 409–421, 2012.

Knox, W Bradley, Stone, Peter, and Breazeal, Cynthia. Training a robot via human feedback: A case study. In *Social Robotics*, pp. 460–470. Springer, 2013.

Knox, W. Bradley Knox and Stone, Peter. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, May 2010.

Knox, William Bradley. *Learning from human-generated reward*. PhD thesis, University of Texas at Austin, 2012.

Loftin, Robert, Peng, Bei, MacGlashan, James, Littman, Michael L., Taylor, Matthew E., Huang, Jeff, and Roberts, David L. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30(1):30–59, 2015.

Maclin, Richard, Shavlik, Jude, Torrey, Lisa, Walker, Trevor, and Wild, Edward. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the National Conference on Artificial intelligence*, volume 20, pp. 819, 2005.

Miltenberger, Raymond G. *Behavior modification: Principles and procedures*. Cengage Learning, 2011.

Pilarski, Patrick M, Dawson, Michael R, Degris, Thomas, Fahimi, Farbod, Carey, Jason P, and Sutton, Richard S. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–7. IEEE, 2011.

Puterman, Martin L. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Sutton, Richard S, McAllester, David A, Singh, Satinder P, Mansour, Yishay, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pp. 1057–1063, 1999.

Tenorio-Gonzalez, Ana C, Morales, Eduardo F, and Villaseñor-Pineda, Luis. Dynamic reward shaping: training a robot by voice. In *Advances in Artificial Intelligence–IBERAMIA 2010*, pp. 483–492. Springer, 2010.

Thomaz, Andrea L and Breazeal, Cynthia. Robot learning via socially guided exploration. In *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pp. 82–87. IEEE, 2007.

Thomaz, Andrea L and Breazeal, Cynthia. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172:716–737, 2008.

Thomaz, Andrea Lockerd and Breazeal, Cynthia. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*, volume 6, pp. 1000–1005, 2006.

Watkins, Christopher J. C. H. and Dayan, Peter. Q-learning. *Machine Learning*, 8(3):279–292, 1992.