Convergent Actor Critic by Humans

James MacGlashan[1], Michael L. Littman[1], David L. Roberts[2], Robert Loftin[2], Bei Peng[3], and Matthew E. Taylor[3]

Abstract— **Programming robot behavior can be painstaking: for a layperson, this path is unavailable without investing significant effort in building up proficiency in coding. In contrast, nearly half of American households have a pet dog and at least some exposure to animal training, suggesting an alternative path for customizing robot behavior. Unfortunately, most existing reinforcement-learning (RL) algorithms are not well suited to learning from human-delivered reinforcement. This paper introduces a framework for incorporating human-delivered rewards into RL algorithms and preliminary results demonstrating feasibility.**

I. Introduction

Programming robots is very difficult, in part because the real world is inherently rich and—to some degree—unpredictable. Nevertheless, for robots to have a significant impact on the lives of individuals, even non-programmers need to be able to specify and customize their behavior.

Reinforcement learning (RL), in which the human trainer provides the 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 such simple signals.

However, traditional RL algorithms have yielded limited success when the reward signal is provided by humans and has 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 RL algorithms that are designed to learn from human-generated rewards and investigation into how people give interactive feedback [6]. In general, many of these human-centered learning algorithms are built on the insight that humans tend to give feedback as a comment on the policy the agent is following, rather than as a numeric value that is meant to be maximized by the agent. While this insight seems accurate, existing approaches use interpretations of feedback that fail to account for, or explain, the different ways in which people give feedback.

We advocate that the *advantage function* is a better interpretation of human feedback that can be straightforwardly incorporated into actor-critic RL algorithms. The advantage function roughly corresponds to how much better taking some action in a state is than following the current policy in that state, and motivates feedback strategies often incorporated by people, but not captured by existing human-centered RL algorithms. For example:

1) Initially give positive feedback for an action and then withdraw the feedback for the same action with time.
2) Provide feedback of different magnitudes with respect to the quality of improvement in behavior.
3) Give positive feedback for suboptimal, but sufficient, actions, and then raise the quality standard necessary for positive feedback over time.

The first strategy reduces the burden on the trainer while the other two allow for explicit behavior shaping.

This paper argues that learning from human reward is a natural fit for actor-critic algorithms, presents one such algorithm, and provides proof of concept results, showing the feasibility of this approach.

II. Background

We formulate the problem using a Markov Decision Process (MDP). 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 $s' \in S$ when the agent takes action $a \in A$ in environment state $s \in S$; $R(s,a,s')$ 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.

The goal is to find an optimal policy $\pi^\star$ that maximizes the expected future discounted reward. The state value function is the expected future discounted reward from each state when following some policy π: $V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi]$. Similarly, the state-action value function is $Q^\pi(s,a) = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi]$.

Bhatnagar et al. [1] provides a general template for *actor-critic* algorithms, shown in Algorithm 1. 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 after each action selection (used to update the policy parameters). The critique is often the temporal difference (TD) error: $\delta_t = r_t + \gamma V(s_t) - V(s_{t-1})$, which describes how much better or worse things went than expected. The main difference between different actor-critic algorithms is how the critic's value function and actor's policy parameters are updated. When states (or actions) are continuous, actor-critic algo-

This work was supported in part by NSF grants IIS1348109, IIS1319412, and IIS1643614. [1] Brown University Computer Science Department, [2] North Carolina State University Department of Computer Science, [3] Washington State University School of Electrical Engineering and Computer Science

Algorithm 1 Actor-Critic Template

```
Input: parameterized policy $\pi_{\theta_0}$, value function $V_0$
  observe initial state $s_0$
  for $t = 0$ to $\infty$ do
      select and execute action $a \sim \pi_{\theta_t}(s_t, \cdot)$
      observe next state $s_{t+1}$ and reward $r_{t+1}$
      compute TD error $\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)$
      update $V_t$ to $V_{t+1}$                    ▷ Algorithm Specific
      update $\theta_t$ to $\theta_{t+1}$ using $\delta_t$          ▷ Algorithm Specific
```

rithms typically employ function approximation for the value function and policy to generalize experience.

III. Human-centered Reinforcement Learning

We define a *human-centered reinforcement learning* (HCRL) problem as a learning problem in which an agent is in an MDP but rewards do not come from the environment. Instead, a human trainer has a target policy $\pi^\star$ she is trying to teach the agent. The trainer communicates this policy by providing positive and negative feedbacks. The goal of the agent is to learn the target policy $\pi^\star$ from these feedbacks.

If the human trainer gives stationary rewards that the agent is meant to maximize, standard RL algorithms could be used. Although there has been some success with this approach [13] there are complications. In particular, a trainer must take care that the feedback strategy she uses contains no unanticipated exploits, which in turn limits the space of feedback strategies she can use. Indeed, prior research has shown that when people give feedback to artificial agents, they often unintentionally induce *positive cycles* if their feedback is interpreted as reward to be maximized [3], [4]. A positive cycle is a feedback pattern that is maximized by a looping behavior that endlessly accrues a net positive return.

Instead, we take the stance that rather than considering reward as fixed, we treat it as commenting on the agent's behavior. Positive feedback roughly corresponds to "yes, that was good" and a negative feedback roughly corresponds to "no, that was bad."

The advantage function A^π is defined as $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. Roughly speaking, the advantage function describes how much better or worse an action selection is compared to the current expected behavior.

We next consider one example training strategy that a human trainer may leverage, as motivated in the Introduction. When a *diminishing returns* strategy is employed by a trainer, positive feedback is reduced as the agent begins to consistently take the correct action in that context. It is formalized in MDP terms as follows. Without loss of generality, consider state s with exclusively optimal action a, and an initial policy π_0 that does not consistently select action a in state s ($\pi_0(s,a) \ll 1$) and for which switching to action a would constitute a policy improvement. The diminishing returns strategy entails that the initial feedback for taking action a in state s will be positive, but goes to zero as $\pi_t(s,a) \to 1$. This strategy is primarily useful to a trainer as a means to limit how actively they have to respond to every behavior, allowing them to focus their attention to contexts in which the agent has not yet learned the correct behavior. It also serves as a means to indicate that the agent should not change anything in that context.

To show that assigning feedback according to the advantage function reproduces the diminishing returns strategy, first note that if switching to action a constitutes a policy improvement to π_0, it follows that $Q^{\pi_0}(s,a) > V^{\pi_0}(s,a)$; therefore, $A^{\pi_0}(s,a) > 0$, thereby yielding an initially positive feedback for taking action a in state s. Moreover, as $\pi(s,a)$ goes to one, $A^\pi(s,a)$ goes to zero, because $V^\pi(s)$ is defined as a linear combination of the Q-values, where the weights are the probability of each action being taken.

This is one example of how a reasonable human training strategy can be incorporated into the actor-critic framework, motivating our choice of framing. Showing that other training strategies, including those in the Introduction, can fit into the actor-critic framework is left for future work.

IV. Convergent Actor Critic by Humans

If the advantage function is a useful model for human feedback, an appropriate learning algorithm is one that can learn from the advantage function. Fortunately, actor-critic algorithms are already primed for learning from the advantage function. Recall that actor-critic algorithms update the agent's policy parameters with the TD error. The TD error, however, is simply an unbiased estimate of the advantage function [1]. Therefore, if human feedback accurately approximates the advantage function, then learning can be performed with an actor-critic algorithm that sets the δ_t term directly to the human feedback value.

Human feedback may be sparse if the agent decision cycle is fast. Sparseness is not problematic in itself because when the advantage function is zero, it indicates no change to the policy. However, if feedback is going to be sparse, it may be useful to allow the trainer to provide feedback about a history of actions, rather than just the last action. *Eligibility traces* are one way to accomplish that. In an actor-critic RL setting, an eligibility trace is associated with each policy parameter. It keeps track of how recently a parameter was "active" and decays exponentially with a parameter λ. Parameter updates are then performed as a product of the trace and δ_t term, allowing rewards and TD errors to affect parameters associated with much older events. Eligibility traces can be similarly used in human advantage algorithms to enable critiques about a history of actions.

Incorporating these considerations yields the *COnvergent Actor Critic by Humans* (COACH) algorithm, with state function approximation, in Algorithm 2. $\boldsymbol{e}_\lambda$ is an eligibility trace vector of equal dimensionality to the number of policy parameters and there is one for each of the different λ values provided to the trainer interface. The feedback signal represents the sum of feedback signals given in the time between the action execution and the resulting state being observed—it is observed with a trainer-provided λ value. This feedback could be associated with the action event in the past that covers the necessary delay to account for human reaction time. All traces are updated with respect to

Algorithm 2 COACH Algorithm

```
Input: policy $\pi_{\theta_0}$, trace set $\boldsymbol{\lambda}$, learning rate $\alpha$
  Initialize traces $\boldsymbol{e}_\lambda \leftarrow \mathbf{0}, \forall \lambda \in \boldsymbol{\lambda}$
  Observe initial state $s_0$
  for $t = 0$ to $\infty$ do
      Select and execute action $a_t \sim \pi_{\theta_t}(s_t, \cdot)$
      Observe next state $s_{t+1}$
      Receive feedback $f_{t+1}$ and $\lambda$ from trainer
      for $\lambda' \in \boldsymbol{\lambda}$ do
          $\boldsymbol{e}_{\lambda'} \leftarrow \boldsymbol{e}_{\lambda'} + \lambda' \nabla_{\theta_t} \pi_{\theta_t}(s_{t-d}, a_{t-d})$
      $\theta_{t+1} \leftarrow \theta_t + \alpha f_{t+1} \boldsymbol{e}_\lambda$
```

the delayed state action event with which the feedback is associated. Finally, the policy parameters are updated in the direction of the trace for the λ and feedback received.

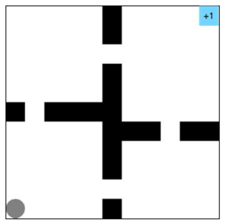
V. Experiments

This section evaluates COACH in simulated grid world environments under two conditions: first, when a human is not involved and reward comes from a stationary and sparse MDP-style reward function; and second, when trained by a human. The first condition provides insight into our algorithm's ability to scale to sparse feedback and allows us to use a standard RL algorithm (designed to learn under these assumptions) as a baseline. Results from the second condition allow us to see how our algorithm performs where discrete observations are noise free, global, and when human-trainer feedback is perfectly timed with actions. It also highlights how standard RL algorithms do not respond well to human-generated reward, even when controlling the rest of the environment to match their assumptions.

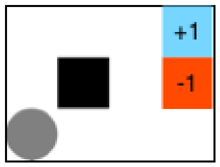
A. *MDP-style Reward Functions*

We chose two standard MDP grid worlds to compare COACH and Q-learning. The first is "four rooms" [11]: an 11×11 deterministic grid world (Figure 1a) with a single goal location yielding a reward of $+1$ and 0 everywhere else. The second, introduced by Russell and Norvig [10], is a stochastic 4×3 grid world (Figure 1b) with both a goal state, yielding a reward of $+1$, and a failure terminal state, yielding a reward of -1; all other steps yield a reward of -0.04. In both grids, reaching a goal/terminal state ends the task, starting a new episode of learning in the bottom left corner. The agent can move in the four cardinal directions. In the Russell-Norvig grid, actions cause the agent to move orthogonally with probability 0.2.

For Q-learning, we tested a handful of parameters and used the best. For COACH we used a softmax policy with a single $\lambda = 0.8$ trace. Figures 2a and 2b show the reward per episode averaged over 10 trials. These results show that neither algorithm has an advantage in this scenario, even though only Q-Learning was designed explicitly for learning in MDPs with sparse rewards. These results suggest that COACH can scale to situations where humans give less frequent feedback.



(a) The four rooms grid



(b) The Russell-Norvig grid

Fig. 1: The grid worlds used for MDP learning.

B. *Human Training: Rewarded for Reaching Goal*

We next present results of a human (the first author of this paper) training an agent using COACH, SARSA(λ), and TAMER, in four rooms with and without function approximation. Here, we use SARSA(λ) instead of Q-learning because it 1) has better convergence properties when using function approximation; and 2) because SARSA(λ) with a large λ is better behaved in non-Markovian environments [7], which human feedback often entails.

During learning, the trainer watched the agent act. After each movement, the agent waited one second, thereby giving the trainer sufficient time to provide a $+1$ or -1 feedback.

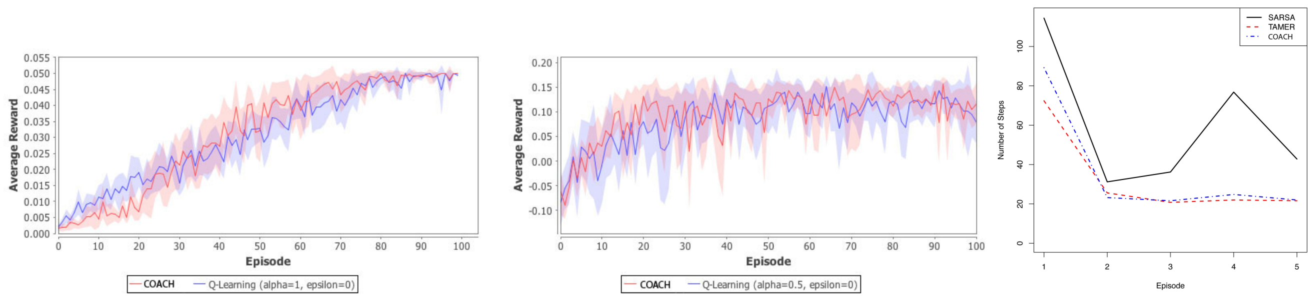
We used tabular COACH with the agent following a greedy version of its learned soft-max policy[1] and a single trace with $\lambda = 0.8$. For TAMER, we used the *Infrequent TAMER*, which assumes that each feedback can perfectly target the last action, and similarly used a tabular model of the human reward. For SARSA(λ), we used a tabular Q-function, a greedy policy, and $\lambda = 0.99$.

Figure 2c shows the number of time steps of training for the first five episodes, averaged over five trials. The trainer aggressively rewarded and punished correct and incorrect behavior in the first episode. After that, feedback was used more sparingly to fine tune behavior. COACH and TAMER both learn very fast and perform similarly on this task, with TAMER doing slightly better in the first round. TAMER's slightly better initial performance may be due to the fact that TAMER's assumption that feedback applies to only the last action better matches the initial training strategy. COACH could match this assumption by using (or including) a λ value of zero. SARSA(λ), on the other hand, is sporadic and tended to get stuck trying to optimize positive reward cycles. Furthermore, SARSA(λ) did not converge well—the lower feedback rates in later trials caused SARSA(λ) to determine that its estimate of the value function was wrong and it unlearned behavior, requiring additional feedback to correct.

C. *Human Training: Lure Training*

Next, we attempted to perform *lure training* with each of the algorithms. Lure training is a standard animal-training "design pattern" in which an animal is first conditioned to follow a lure that the trainer can easily manipulate (*e.g.*, a ball on a stick). After conditioning, the trainer uses the lure to guide the animal through complex tasks that otherwise would have taken the animal a long time to complete. After

[1]Since we expect frequent feedback, an exploration policy is unnecessary.



(a) Four rooms results (b) Russell-Norvig grid results (c) Human training times

Fig. 2: Average reward per learning episode for COACH (red) and Q-learning (blue) on the (a) four rooms, and (b) Russell-Norvig grids. Bands indicate the 95% confidence interval. (c) Average human training times in four rooms for SARSA(λ) (black), TAMER (red), and COACH (blue).

being guided through the task with the lure, and rewarded at the end, the animal is able to complete the task without the lure (or requires only limited additional reinforcement).

For lure training, we added the ability for the trainer to instantiate a lure in the grid world. Before training on the full task, the agent was conditioned to follow the lure in two separate training areas. After conditioning, the lure was used in the full grid to guide the agent to the goal—a single positive reward was given once, if the agent reached goal.

Of the three learning algorithms, only COACH successfully learned from the lure training (results not shown). After luring the agent to the goal and giving it a single reward, the agent was able to complete the task without the lure and without additional feedback, showing that strategies used by humans may be effective for training actor-critic algorithms.

D. Robot Experiments

We successfully trained a TurtleBot learning with COACH to execute five behaviors: push pull, hide, alternate, ball following, and target navigation using lure training. Details of the Turtlebot experiments are omitted for space. A seven-minute video showing this proof of concept is at `https://www.youtube.com/watch?v=XRwYcX6eJz4`.

VI. Related Work

There is significant related work that allows humans to provide reward feedback for learning agents. Most related is work by Knox *et al*. [5] on the TAMER algorithm. TAMER learns a human trainer providing positive and negative feedback, learning to greedily maximize a short-term expectation of human reward. Other related work includes Thomaz and Breazeal [13], which introduces an RL algorithm that incorporates interactive human guidance to elicit a significantly faster learning interaction; Tenorio-Gonzalez *et al*. [12], which introduces a dynamic reward shaping approach to learn from infrequent and inconsistent human feedback; and Griffith *et al*. [2], which develops a Bayesian approach using policy shaping. Our prior work [8], [9] differs in that it treats the human feedback as categorical and does not use eligibility traces or the actor-critic framework.

In general, all these works suffer if the human trainer changes her feedback strategy over time, unlike when the reward is modeled as the advantage function.

VII. Conclusion and Future Work

This work provided two main contributions. First, we advocated that human feedback is better modeled with the advantage function. Second, we introduced a new actor-critic algorithm specifically designed to learn from humans. Preliminary experiments show encouraging results.

In the future, we plan to 1) implement additional actor-critic algorithms that can learn from human-provided rewards, 2) perform human trial studies to identify how well average end-users do at training robots using such actor-critic algorithms, 3) provide proofs of correctness for this class of algorithms, and 4) augment the critic so that it can combine human-provided reward with environmental reward.

References

[1] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. Technical report, University of Alberta, 2009.

[2] S. Griffith, K. Subramanian, J. Scholz, C. Isbell, and A. L. Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *NIPS*, pages 2625–2633, 2013.

[3] 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.

[4] W. B. Knox. *Learning from human-generated reward*. PhD thesis, University of Texas at Austin, 2012.

[5] 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*, 2009.

[6] W. B. Knox, P. Stone, and C. Breazeal. Training a robot via human feedback: A case study. In *Social Robotics*, pages 460–470. 2013.

[7] J. Loch and S. P. Singh. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML*, pages 323–331, 1998.

[8] 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. *J. of Autonomous Agents and Multi-Agent Systems*, 2015.

[9] B. Peng, J. MacGlashan, R. Loftin, M. L. Littman, D. L. Roberts, and M. E. Taylor. A Need for Speed: Adapting Agent Action Speed to Improve Task Learning from Non-Expert Humans. In *AAMAS*, 2016.

[10] S. Russell and P. Norvig. *AI a modern approach*, volume 2. 2005.

[11] R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

[12] 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*, pages 483–492. 2010.

[13] A. L. Thomaz and C. Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*, volume 6, pages 1000–1005, 2006.