Augmenting Reinforcement Learning with Human Feedback

W. Bradley Knox BRADKNOX@CS.UTEXAS.EDU
University of Texas at Austin, Department of Computer Science

Peter Stone PSTONE@CS.UTEXAS.EDU
University of Texas at Austin, Department of Computer Science

Abstract

As computational agents are increasingly used beyond research labs, their success will depend on their ability to learn new skills and adapt to their dynamic, complex environments. If human users — without programming skills — can transfer their task knowledge to agents, learning can accelerate dramatically, reducing costly trials. The TAMER framework guides the design of agents whose behavior can be shaped through signals of approval and disapproval, a natural form of human feedback. More recently, TAMER+RL was introduced to enable human feedback to augment a traditional reinforcement learning (RL) agent that learns from a Markov decision process's (MDP) reward signal. Using a reimplementation of TAMER and TAMER+RL, we address limitations of prior work, contributing in two critical directions. First, the four successful techniques for combining a human reinforcement with RL from prior TAMER+RL work are tested on a second task, and these techniques' sensitivities to parameter changes are analyzed. Together, these examinations yield more general and prescriptive conclusions to guide others who wish to incorporate human knowledge into an RL algorithm. Second, TAMER+RL has thus far been limited to a *sequential* setting, in which training occurs before learning from MDP reward. We modify the sequential algorithms to learn *simultaneously* from both sources, enabling the human feedback to come at any time during the reinforcement learning process. To enable simultaneous learning, we introduce a new technique that appropriately determines the magnitude of the human model's influence on the RL algorithm throughout time and state-action space.

Appearing in *Proceedings of the ICML Workshop on New Developments in Imitation Learning*, Bellevue, WA, USA, 2011.

1. Introduction

Computational agents may soon be prevalent in society, and many of their end users will want these agents to learn to perform new tasks. For many of these tasks, the human user will already have significant task knowledge. Consequently, we seek to enable non-technical users to transfer their knowledge to the agent, reducing the cost of learning without hurting the agent's final, asymptotic performance.

In this vein, the TAMER framework guides the design of agents that learn by shaping — using signals of approval and disapproval to teach an agent a desired behavior (Knox and Stone, 2009). As originally formulated, TAMER was limited to learn exclusively from the human feedback. More recently, TAMER+RL was introduced with the goal of enabling the human feedback to augment a traditional reinforcement learning (RL) agent that learns from an MDP reward signal (Knox and Stone, 2010). However, TAMER+RL has previously only been tested on a single domain, and it has been limited to the case where the learning from human feedback happens only prior to RL: sequential TAMER+RL. Using a reimplementation of TAMER and TAMER+RL, we address these limitations by improving upon prior work in two crucial directions.

First, in Section 3, we continue with the sequential TAMER+RL approach, testing the four TAMER+RL techniques that were previously found to be successful. We test on two tasks — one identical to the single prior TAMER+RL task and a new task. We also provide a novel examination of each technique's performance at a range of combination parameter values to determine the ease of setting each parameter effectively, a critical aspect of using TAMER+RL algorithms in practice that has been previously sidestepped. Together, these analyses yield stronger, more prescriptive conclusions than were possible from prior work. Two similar combination techniques, for the first time, clearly stand out as the most effective, and we consistently observe that manipulating action selection is more effective than altering the RL update.

Second, in Section 4 we move from the sequential set-

ting of first learning only from the human and then learning from MDP reward to learning from both simultaneously. The principal benefit of simultaneous learning is its flexibility; it gives a trainer the important ability to step in as desired to alter the course of reinforcement learning while it is in progress. We demonstrate the success of the two best-performing techniques from the sequential experiments, action biasing and control sharing, in this simultaneous setting. To meet demands introduced by the simultaneous setting, we use a novel method to moderate the influence of the model of human reinforcement on the RL algorithm. Our method increases influence in areas of the state-action space that have recently received training and slowly decreases influence in the absence of training, leaving the original MDP reward and base RL agent to learn autonomously in the limit. Without this improvement, the sequential techniques would be too brittle for simultaneous learning.

2. Preliminaries

In this section, we briefly introduce reinforcement learning and the TAMER Framework.

2.1. Reinforcement Learning

We assume that the task environment is a Markov decision process (MDP) specified by the tuple (S, A, T, γ, D, R). S and A are respectively the sets of possible states and actions. T is a transition function, $T : S \times A \times S \rightarrow \mathbb{R}$, which gives the probability, given a state s_t and an action a_t, of transitioning to state s_{t+1}. γ, the discount factor, exponentially decreases the value of a future reward. D is the distribution of start states. R is a reward function, $R : S \times A \times S \rightarrow \mathbb{R}$, where the reward is a function of s_t, a_t, and s_{t+1}. We will also consider reward that is a function of only s_t and a_t.

Reinforcement learning algorithms (see Sutton and Barto (1998)), seek to learn policies ($\pi : S \rightarrow A$) for an MDP that maximize return from each state-action pair, where $return = \sum_{t=0}^{T} E[\gamma^t R(s_t, a_t, s_{t+1})]$. In this paper, we focus on using a value-function-based RL method, namely SARSA(λ) (Sutton and Barto, 1998), augmented by the TAMER-based learning that can be done directly from a human's reinforcement signal. Though more sophisticated RL methods exist, we use SARSA(λ) for its popularity and representativeness, and because we are not concerned with finding the best overall algorithm for our experimental tasks but rather with determining how various methods for including a human model change the base RL algorithm's performance.

2.2. The TAMER Framework for Interactive Shaping

The TAMER Framework, introduced by Knox and Stone (2009) is an approach to the problem of how an agent should learn from numerically mapped reinforcement signals. Specifically, these feedback signals are delivered by an observing human trainer as the agent attempts to perform a task.[1] TAMER is motivated by two insights about human reinforcement. First, reinforcement is trivially delayed, slowed only by the time it takes the trainer to assess behavior and deliver feedback. Second, the trainer observes the agent's behavior with a model of that behavior's long-term effects, so the reinforcement is assumed to be fully informative about the quality of recent behavior. Human reinforcement is more similar to an action value (sometimes called a Q-value), albeit a noisy and trivially delayed one, than MDP reward. Consequently, TAMER assumes human reinforcement to be fully informative about the quality of an action given the current state, and it models a hypothetical human reinforcement function, $H : S \times A \rightarrow \mathbb{R}$, as $\hat{H}$ in real time by regression. In the simplest form of credit assignment, each reinforcement creates a label for the last state-action pair.[2] The output of the resultant $\hat{H}$ function — changing as the agent gains experience — determines the relative quality of potential actions, so that the exploitative action is $a = argmax_a[\hat{H}(s, a)]$.

3. Sequential TAMER+RL

Noting that TAMER agents typically learn faster than agents learning from MDP reward but to a lower performance plateau, Knox and Stone combined TAMER and SARSA(λ) (2010). Their aim was to complement TAMER's fast learning with RL's ability to often learn better policies in the long run. These conjoined TAMER+RL algorithms address a scenario in which a human trains an agent, leaving a model $\hat{H}$ of reinforcement, and then $\hat{H}$ is used to influence the base RL algorithm somehow. We call this scenario and the algorithms that address it *sequential* TAMER+RL. For all TAMER+RL approaches, only MDP reward is considered to specify optimal behavior. $\hat{H}$ provides guidance but not an objective. In this section, we reproduce and then extend prior investigations of sequential TAMER+RL, yielding more prescriptive and general conclusions than prior work allowed.

3.1. Combination techniques

Knox and Stone tested eight TAMER+RL techniques that each use $\hat{H}$ to affect the RL algorithm in a different way. Four were largely effective when compared to the SARSA(λ)-only and TAMER-only agents[3] on both mean re-

[1] In our experiments, the trainer has a button for positive reinforcement and one for negative. Multiple button presses are roughly interpreted as more intense feedback.

[2] The trivial delay is dealt with using a credit assignment technique described in Knox and Stone (2009).

[3] A TAMER-only agent simply uses $\hat{H}$ to choose actions, ignoring MDP reward. In sequential TAMER+RL, $\hat{H}$ is constant, and thus so is the agent's policy.

ward over a run and performance at the end of the run. We focus on those four techniques, which can be used on any RL algorithm that uses an action-value function. Below, we list them with names we have created. In our notation, a prime (e.g., Q') after a function means the function replaces its non-prime counterpart in the base RL algorithm.

- **Reward shaping**: $R'(s,a) = R(s,a) + (\beta * \hat{H}(s,a))$
- **Q augmentation**: $Q'(s,a) = Q(s,a) + (\beta * \hat{H}(s,a))$
- **Action biasing**: $Q'(s,a) = Q(s,a) + (\beta * \hat{H}(s,a))$ *only during action selection*
- **Control sharing**: $P(a{=}argmax_a[\hat{H}(s,a)]) = min(\beta, 1)$. *Otherwise use base RL agent's action selection mechanism.*

These four techniques are numbered 1, 4, 6, and 7 in Knox and Stone (2010). We altered action biasing to generalize it, but the ϵ-greedy policies we use in our experiments are not affected. In the descriptions above, β is a predefined combination parameter. In our sequential TAMER+RL experiments, β is annealed by a predefined factor after each episode for all techniques other than Q augmentation.

We now briefly discuss these techniques and situate them within related work. In the RL literature, reward shaping adds the output of a shaping function to the original MDP reward, creating a new reward to learn from instead (Dorigo and Colombetti, 1994; Mataric, 1994). As we confirm in the coming paragraph on Q augmentation, our reward shaping technique is not the only way to do reward shaping, but it is the most direct use of $\hat{H}$ for reward shaping.

If $\hat{H}$ is considered a heuristic function, action biasing is the same action selection method used in Bianchi et al.'s Heuristically Accelerated Q-Learning (HAQL) algorithm (Bianchi *et al.*, 2004). Control sharing is equivalent to Fernández and Veloso's π-*reuse exploration strategy* (2006). Note that both control sharing and action biasing only affect action selection and can be interpreted as directly guiding exploration toward human-favored state-action pairs.

Q augmentation is action biasing with additional use of $\hat{H}$ during the Q-function's update. Wiewiora et al.'s related *look-ahead advice* (2003) uses a discounted change in the output of a state-action potential function, $\gamma\phi(s_{t+1}, a_{t+1}) - \phi(s_t, a_t)$, for reward shaping and to augment action values during action selection. Interestingly, *look-ahead advice* is equivalent to Q augmentation when $\hat{H}$ is used for ϕ, the state and action space are finite, and the policy is invariant to adding a constant to all action values in the current state (e.g., ϵ-greedy and soft-max).

3.2. Sequential learning experiments

We now describe our sequential TAMER+RL experiments. We first validate our reimplementation of TAMER and TAMER+RL by reproducing Knox and Stone's results on the single task they tested. We then evaluate the algorithms' effectiveness on a different task. Additionally, we analyze our results at a range of combination parameter values (β values) to identify challenges to setting β's value without prior testing.

Following past work on TAMER and TAMER+RL, we implemented the corresponding algorithms as exactly as we could, excepting some changes to the credit assignment technique in Knox and Stone (2009).[4] Using the original $\hat{H}$ representation (linear model of RBF features), task settings, SARSA(λ) parameters, and training records from Knox and Stone (2010),[5] we repeat their experiments on the Mountain Car task,[6] using all four combination techniques found to be successful in their experiments and a range of β combination parameters. We then test these TAMER+RL techniques on a second task, Cart Pole, using an $\hat{H}$ model trained by an author. We again use SARSA(λ), choosing parameters that perform well but sacrifice some performance for episode-to-episode stability and the ability to evaluate policies that might otherwise balance the pole for too long to finish a run. In Mountain Car, the goal is to quickly move the car up a hill to the goal. The agent receives -1 reward for all transitions to non-absorbing states. In Cart Pole, the goal is to move a cart so that an attached, upright pole maintains balance as long as possible. The agent receives +1 reward for all transitions that keep the pole within a specified range of vertical. The $\hat{H}$ for Cart Pole was learned by k-Nearest Neighbor. For both tasks, we use Gaussian RBF features for SARSA(λ) and initialize Q pessimistically, as was found effective in Knox and Stone (2010). In these and later experiments, $\hat{H}$ outputs are typically in the range [-2, 2].

We evaluate each combination technique on four criteria; full success requires outperforming the corresponding $\hat{H}$'s TAMER-only policy and SARSA(λ)-only both in end-run performance and cumulative reward (or mean reward across full runs, equivalently).

[4] For space considerations, we we will not fully describe these changes. Briefly, for each reinforcement signal received, Knox and Stone create a learning sample for every time step within a window of recent experience, resulting in many samples per reinforcement in fast domains. We instead create one sample per time step, using all crediting reinforcements to create one label.

[5] The models we create — $\hat{H}_1$ and $\hat{H}_2$ — from the original training trajectories perform a bit better than those from Knox and Stone's experiments, which points to small implementation differences.

[6] Tasks are adapted from RL-Library (Tanner and White, 2009).

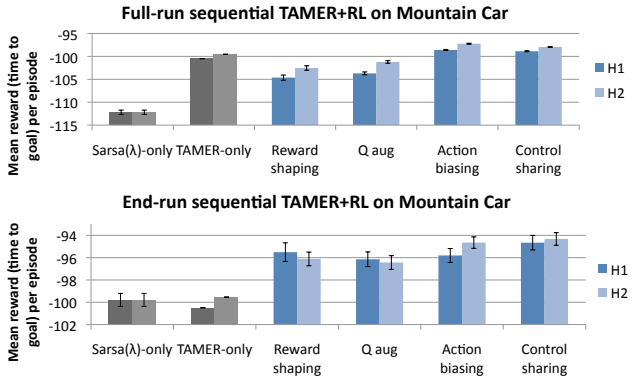

Figure 1. Comparison of TAMER+RL techniques with SARSA(λ) and the TAMER-only policy on Mountain Car over 40 or more runs of 500 episodes. $\hat{H}_1$ and $\hat{H}_2$ are models from two different human trainers. The top chart considers reward over the entire run, and the bottom chart evaluates reward over the final 10 episodes. Error bars show standard error.

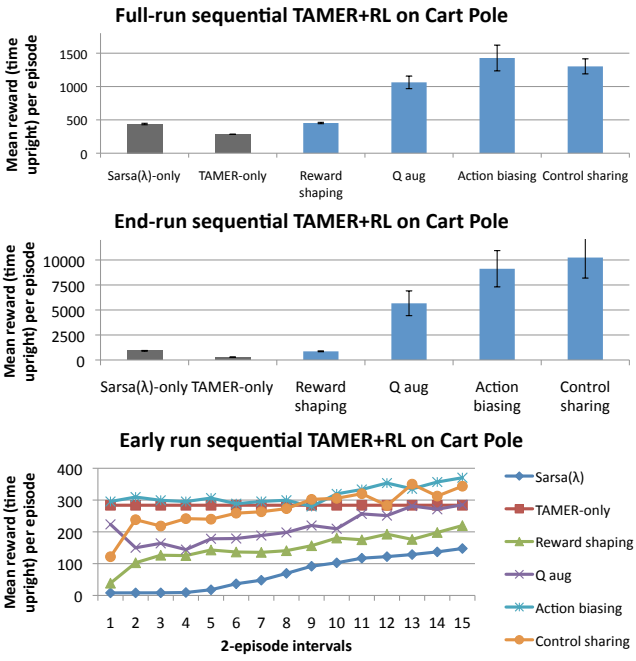

Figure 2. The same TAMER+RL comparisons as in Figure 1, but on Cart Pole over runs of 150 episodes. A single $\hat{H}$ was used. End-run performance is the mean reward during the last 5 episodes.

3.3. Sequential learning results and discussion

Figures 1 and 2 show the results of our experiments for sequential TAMER+RL. For now, we only show results for the β combination parameters that accrue the highest cumulative reward for their corresponding technique. Figure 2 additionally shows learning curves for the first 30 episodes of the Cart Pole run. (Our early-run results for Mountain Car are similar to those shown by Knox and Stone (2010)).

Qualitatively, our Mountain Car results agree with previous work. Action biasing and control sharing succeed on all four criteria and significantly outperform other techniques in cumulative reward. Reward shaping and Q augmentation also improve over SARSA(λ)-only by both metrics and over the TAMER-only policies in end-run reward.

On Cart Pole, action biasing and control sharing again succeed fully. This time, Q augmentation also meets the criteria for success, though it performs significantly worse than action biasing and control sharing. Most interestingly, reward shaping, at its best tested parameter, does not significantly alter SARSA(λ)'s performance on either metric.

By choosing the best β parameter value for each technique, prior TAMER+RL experiments sidestep the issue of using an effective value without first testing a range of values. With experiments in two tasks, we can begin to address this problem by examining each technique's sensitivity to β parameter changes and whether certain ranges of β are effective across different tasks. In Figure 3, we show the mean performance of each combination technique as β varies. Examining the charts, we consider several criteria:

- performance at worst β value,
- range of beneficial β values,
- and existence of β values that are effective across tasks.

Evaluating the techniques on these three criteria creates a consistent story that fits with our analysis of the techniques at their best β parameter values (in Figures 1 and 2). The two methods that only affect action selection — action biasing and control sharing — emerge as the most effective techniques without a clear leader between them, and they are followed by Q augmentation and then shaping rewards.

From an RL perspective, the weakness of reward shaping may be counterintuitive. When researchers discuss combining human reinforcement with RL in the literature, reward shaping is predominantly suggested (Thomaz and Breazeal, 2006; Isbell *et al.*, 2006), possibly because human "reward" is seen as an analog to MDP reward that should be used similarly. However, though reward shaping is generally cast as a guide for exploration, it only affects exploration indirectly through precariously tampering with the reward signal. Action biasing and control sharing affect exploration directly, without manipulating reward. Thus, they achieve the stated goal of reward shaping while leaving the agent to learn accurate values from its experience. Following this line of thought, Q augmentation is identical to action biasing during action selection, boosting each action's Q-value by the weighted prediction of human reinforcement. In addition to this direct guidance on exploration, Q augmentation also changes the Q-value during the SARSA(λ) update's calculation of temporal difference error. As discussed in Section 3.1, Q augmentation is nearly

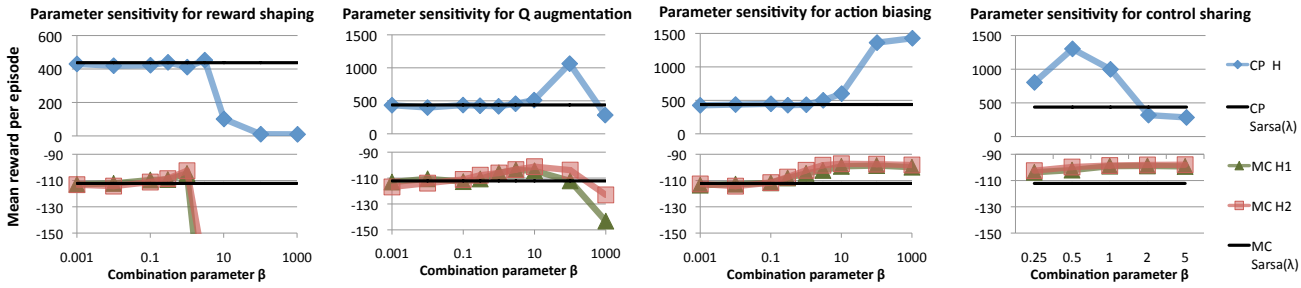

Figure 3. Performance of each technique with each tested $\hat{H}$ over a range of β parameters on two tasks: Cart Pole (CP) and Mountain Car (MC). Note changes in y-axis scaling.

equivalent to a form of reward shaping called look-ahead advice (Wiewiora *et al.*, 2003). In short, we observe that the more a technique directly affects action selection, the better it does, and the more it affects the update to the Q function for each transition experience, the worse it does. Q augmentation does both and performs between the techniques that do only one.

Taken together, these experiments validate Knox and Stone's conclusions and yield new, firmer conclusions about the relative effectiveness of each technique, endorsing action biasing and control sharing over the two other previously successful techniques. And more generally, these results endorse manipulating action selection and leaving the action-value model's update unmolested.

4. Simultaneous TAMER+RL

To this point, similarly to all prior work on TAMER, we have assumed that the human training was finished prior to any reinforcement learning. This "sequential" learning is sometimes appropriate; for instance, when a difficult-to-simulate reward function is tied to potentially costly learning trials and the agent can train in simulation without significant cost. However, in other scenarios this assumption can be limiting. In this section, we investigate how to modify sequential TAMER+RL algorithms to allow a trainer to step in as desired to alter the course of reinforcement learning while it is in progress. We call this scenario and the algorithms that address it "simultaneous" TAMER+RL. Specifically, the agent should learn simultaneously from two feedback modalities — human reinforcement and MDP reward — as one fully integrated system. As in the sequential TAMER+RL approaches, we examine techniques that use only $\hat{H}$ from TAMER in the RL algorithm, otherwise leaving the two algorithms as separate modules.

Since TAMER empirically compares most favorably against RL algorithms in early learning (Knox and Stone, 2009), we expect the greatest gains to come from training near the beginning of learning. However, training at any suboptimal point along the learning curve should benefit the agent, and we hope to do little harm if the agent is already performing optimally and the trainer's feedback cannot help.

Some desirable characteristics for simultaneous learning are:

1. *behavioral consistency*: The agent's behavior should not be erratic, making it difficult to give feedback to specific actions.
2. *responsiveness to the trainer*: The agent should quickly and obviously demonstrate that it is learning from human reinforcement to maintain interactivity.
3. *trainer can give feedback to the RL-only policy*: If a trainer comes in midway through learning, the trainer should be able to *capture* the good aspects of what has already been learned and criticize the negative aspects.
4. *trainer's influence is applied appropriately*: $\hat{H}$'s influence on the RL algorithm's learning and/or action selection should be larger in more recently trained areas of the state-action space and smaller in areas trained less recently.

Simultaneous learning — and its inclusion of RL-based action selection during training — presents new challenges for maintaining behavioral consistency. For instance, control sharing abruptly shifts between two policies, which can create erratic behavior with many different actions (both good and bad) in a small time period, increasing the difficulty of giving clear feedback. Also note that the second and third characteristics are in opposition. Fully responding to the trainer's reinforcement requires abandoning the policy learned by MDP reward. Our module for determining human influence, described in the following section, strikes a balance by ramping up the influence of $\hat{H}$ with increased reinforcement, keeping the RL policy early on.

4.1. Determining the immediate influence of $\hat{H}$

Simultaneous learning allows human trainers to insert themselves at any point of the learning process. Consequently, $\hat{H}$'s influence should increase in areas of the state-action space with recent reinforcement — but not in areas

that have not been targeted with feedback — and decrease in the absence of reinforcement, leaving the set of optimal policies unchanged in the limit. Thus, we must do more than annealing a combination parameter, as is done in sequential learning.

We determine $\hat{H}$'s influence through a novel adaptation of the eligibility traces often used in reinforcement learning (Sutton and Barto, 1998). We will refer to it as the *eligibility module*. The general idea of this eligibility module is that we maintain an eligibility trace for each state-action feature[7], normalized between 0 and 1, that represents the recency of training while that feature was active (i.e., nonzero). Then, the eligibility traces and a time step's feature vector together calculate a measure of the recency of training in similar feature vectors. That measure, multiplied by a constant scaling parameter c_s, is used as the β term introduced in Section 3.1. The implementation follows.

Let $\overrightarrow{e}$ be the vector of traces and $\overrightarrow{f_n}$ be the feature vector normalized such that each element of $\overrightarrow{f_n}$ exists within the range $[0, 1]$. The eligibility module is designed to make β a function of $\overrightarrow{e}$, $\overrightarrow{f_n}$, and c_s with range $[0, c_s]$. A guiding design constraint is that when $\overrightarrow{e} = \overrightarrow{1}$ (i.e., each element of $\overrightarrow{e}$ is the maximum allowed), the normalized dot product of $\overrightarrow{e}$ and any $\overrightarrow{f_n}$, denoted $n(\overrightarrow{e} \cdot \overrightarrow{f_n})$, should equal 1 (since it weights the influence of $\hat{H}$). To achieve this, we make $n(\overrightarrow{e} \cdot \overrightarrow{f_n}) = \overrightarrow{e} \cdot (\overrightarrow{f_n} \,/\, \| \overrightarrow{f_n} \|_1) = (\overrightarrow{e} \cdot \overrightarrow{f_n}) \,/\, (\| \overrightarrow{f_n} \|_1) = \beta \,/\, c_s$. Thus, at any time step with normalized features $\overrightarrow{f_n}$, the influence of $\hat{H}$ is calculated as $\beta = c_s(\overrightarrow{e} \cdot \overrightarrow{f_n})/(\| \overrightarrow{f_n} \|_1)$. This formula has a desirable mathematical characteristic; for a given $\overrightarrow{e}$, β is higher when relatively large feature values correspond to large trace values — indicating the current state-action pair is similar to the recently trained state-action pairs — and β is smaller when large feature values correspond to small trace values.

Using accumulating traces capped at 1, the trace is updated with $\overrightarrow{f_n}$ during training: $e_i := min(1, e_i + (f_{n,i} * a))$, where e_i and $f_{n,i}$ are the i^{th} elements of $\overrightarrow{e}$ and $\overrightarrow{f_n}$, respectively, and a is a constant factor that moderates the speed of accumulation. During time steps without training, $\overrightarrow{e} := decayFactor * \overrightarrow{e}$.

4.2. Simultaneous learning experiments

Our experiments test the effectiveness of simultaneous TAMER+RL when training starts either at the beginning of learning or after some learning has occurred. We again use Mountain Car and Cart Pole, and we focus on the two best-performing combination techniques, action biasing and control sharing. For the eligibility module, the scaling parameter c_s for Mountain Car and Cart Pole is respectively 100 and 200 for action biasing and 2 and 1 for control sharing. These values were chosen to be on the upper end of each method's effective β values in Figure 3. The accumulation factor a for eligibility is 0.2. Training in Mountain Car occurs either for 16 episodes, starting at episode 1, or for 12 episodes after 20 episodes of SARSA(λ)-only learning. In Cart Pole, training at start occurs for 12 episodes, and training after 25 episodes of SARSA(λ)-only learning lasts 8 episodes. The start times are chosen to represent the beginning of learning and also a point at which the SARSA(λ) agent has learned a policy that is much improved but still quite flawed.[8] The number of episodes corresponds to an informal assessment of how many episodes are needed to satisfactorily train the agent; training at later start times progresses more quickly. The trainer has a button that starts and stops training during the designated training episodes, letting the human observe without the agent updating $\hat{H}$ or the eligibility module.

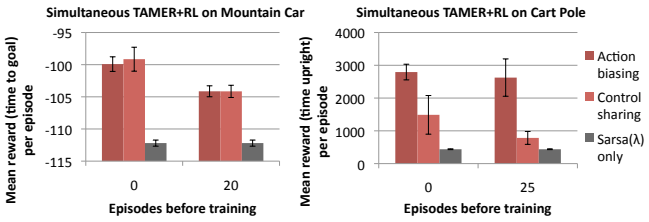

Figure 4. Simultaneous TAMER+RL results. Mean reward is calculated over runs of 500 episodes in Mountain Car and 150 episodes in Cart Pole. Standard error is shown.

An added experimental challenge is that the training is inextricably bound to one specific run, whereas sequential experiments can reuse the same training session for any number of parameters and combination techniques, limiting the depth of analysis that can be done for a set number of trainer-hours. Mountain Car and Cart Pole training sessions typically took around 8 minutes and 15 minutes each, respectively. Consequently, each experimental condition was limited to 3 runs of training for a total of 12 runs on each task.

4.3. Simultaneous learning results and discussion

The results of our simultaneous TAMER+RL experiments are shown in Figure 4. Though the sample size is too small to show statistical significance, there is a clear pattern of both action biasing and control sharing outperforming SARSA(λ). The condition that is closest to SARSA(λ) in

[7]The feature vector is extracted from the current state-action pair. We advise using features that generalize across state space (e.g., Gaussian RBFs). The state-action features need not match those of either $\hat{H}$ or Q.

[8]Note that sequential TAMER+RL differs from simultaneous TAMER+RL where training occurs at the start because the sequential algorithm begins with a pre-trained $\hat{H}$. The training episodes are not counted in sequential TAMER+RL experiments.

terms of standard error, control sharing on Cart Pole where training begins after 25 episodes, still receives almost twice the reward of SARSA(λ). We also observe that training at the beginning of learning is more effective than training after some autonomous learning, as we expected. Seeing this, one might ask whether the n episodes of RL-only learning before training is helping or whether the prior learning should be abandoned to start from scratch. We can test this. Starting from scratch after n episodes is the same as simply training from the start and stopping n episodes early. So if we ignore the first n episodes of the later-training group and the last n episodes of the training-at-start group, the comparison of the groups' mean reward addresses this question. Of four such comparisons (2 techniques x 2 tasks), the later-training group outperforms three times and is roughly equal once, suggesting that the prior learning does indeed help. These results serve as proof of concept for the effectiveness of simultaneous TAMER+RL with our eligibility module.

5. Related Work

In this section, we situate our work within prior research on naturally transferring knowledge to a reinforcement learning agent. We focus on work not already mentioned in Section 3.1, or in the previous papers on TAMER (Knox and Stone, 2009; 2010).

In the only other example of an agent learning simultaneously from human reinforcement and MDP reward, Thomaz and Breazeal (2006) interfaced a human trainer with a table-based Q-learning agent in a virtual kitchen environment. Their agent seeks to maximize its discounted total reward, which for any time step is the sum of human reinforcement and environmental reward. Their approach is a form of reward shaping, differing in that Thomaz and Breazeal directly apply the human reinforcement value to the current reward (instead of modeling reinforcement and using the output of the model as supplemental reward).

Judah et al. consider a learning scenario that alternates between "practice", where actual world experience is gathered, and an offline labeling of actions as good or bad by a human critic (2010). Using an elegant probabilistic technique with a few assumptions, the human criticism is input to a loss function that lessens the expected value of candidate policies while also automatically determining the level of influence given to the criticism. From some mixed results and comments from frustrated subjects, they predicted that redesigning their system to be more interactive and to let the human train periodically — characteristics of simultaneous TAMER+RL — would improve performance.

Imitation learning, or programming by demonstration, has also been used to improve reinforcement learning, using preprogrammed policies (Price and Boutilier, 2003) or humans (Taylor *et al.*, 2011; Smart and Kaelbling, 2000) to provide demonstrations for an agent that observes and learns. These methods are similar to control sharing. An advantage, though, of reinforcement over demonstration is that reinforcement permits learning the relative values of actions, allowing techniques like action biasing to gently push the behavior of the RL agent towards the policy endorsed by $\hat{H}$, whereas pure demonstration is all or nothing — either the demonstrator or the learning agent chooses the action. Additionally, trainers can give reinforce state-action pairs visited by the agent's policy, whereas demonstrations might not ever visit areas of the state space that the imitation learning algorithm visits.

6. Conclusion

Prior work on TAMER+RL is limited by having only tested on a single domain and by simply taking the best β combination parameter from testing. Further, past TAMER+RL algorithms were designed for sequential learning and were unsuitable for simultaneously learning from the trainer and the MDP reward signal. This paper addresses these limitations, giving a clear endorsement of using $\hat{H}$ to affect action selection and, for the first time, enabling a human trainer to interactively provide feedback at any time during the learning process, a critical improvement towards the practicality and widespread applicability of the TAMER framework.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the NSF (IIS-0917122), ONR (N00014-09-1-0658), and the Federal Highway Administration (DTFH61-07-H-00030). An NSF Graduate Research Fellowship supports the first author.

References

R.A.C. Bianchi, C.H.C. Ribeiro, and A.H.R. Costa. Heuristically Accelerated Q–Learning: a new approach to speed up Reinforcement Learning. *Advances in AI – SBIA*, 2004.

M. Dorigo and M. Colombetti. Robot shaping: Developing situated agents through learning. *Artificial Intelligence*, 1994.

F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. *AAMAS*, 2006.

C.L. Isbell, M. Kearns, S. Singh, C.R. Shelton, P. Stone, and D. Kormann. Cobot in LambdaMOO: An Adaptive Social Statistics Agent. *AAMAS*, 2006.

K. Judah, S. Roy, A. Fern, and T.G. Dietterich. Reinforcement Learning Via Practice and Critique Advice. *AAAI*, 2010.

W.B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The TAMER framework. *K-CAP*, 2009.

W.B. Knox and P. Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. *AAMAS*, 2010.

M.J. Mataric. Reward functions for accelerated learning. *ICML*, 1994.

B. Price and C. Boutilier. Accelerating reinforcement learning through implicit imitation. *JAIR*, 19:569–629, 2003.

W.D. Smart and L.P. Kaelbling. Practical reinforcement learning in continuous spaces. *ICML*, 2000.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

B. Tanner and A. White. RL-Glue: Language-independent software for reinforcement-learning experiments. *JMLR*, 10, 2009.

M. Taylor, H.B. Suay, and S. Chernova. Integrating reinforcement learning with human demonstrations of varying ability. *AAMAS*, 2011.

A.L. Thomaz and C. Breazeal. Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance. *AAAI*, 2006.

E. Wiewiora, G. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. *ICML*, 2003.