Combining Manual Feedback with Subsequent MDP Reward Signals for Reinforcement Learning

W. Bradley Knox
University of Texas at Austin
Department of Computer Science
bradknox@cs.utexas.edu

Peter Stone
University of Texas at Austin
Department of Computer Science
pstone@cs.utexas.edu

ABSTRACT

As learning agents move from research labs to the real world, it is increasingly important that human users, including those without programming skills, be able to teach agents desired behaviors. Recently, the TAMER framework was introduced for designing agents that can be interactively shaped by human trainers who give only positive and negative feedback signals. Past work on TAMER showed that shaping can greatly reduce the sample complexity required to learn a good policy, can enable lay users to teach agents the behaviors they desire, and can allow agents to learn within a Markov Decision Process (MDP) in the absence of a coded reward function. However, TAMER does not allow this human training to be combined with autonomous learning based on such a coded reward function. This paper leverages the fast learning exhibited within the TAMER framework to hasten a reinforcement learning (RL) algorithm's climb up the learning curve, effectively demonstrating that human reinforcement and MDP reward can be used in conjunction with one another by an autonomous agent. We tested eight plausible TAMER+RL methods for combining a previously learned human reinforcement function, $\hat{H}$, with MDP reward in a reinforcement learning algorithm. This paper identifies which of these methods are most effective and analyzes their strengths and weaknesses. Results from these TAMER+RL algorithms indicate better final performance and better cumulative performance than either a TAMER agent or an RL agent alone.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Human Factors

Keywords

shaping, reinforcement learning, human-agent interaction, human teachers

Cite as: Combining Manual Feedback with Subsequent MDP Reward Signals for Reinforcement Learning, W. Bradley Knox and Peter Stone, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

1. INTRODUCTION

As learning agents move from research labs to the real world, it is increasingly important that (1) human users, including those without programming skills, be able to teach agents desired behaviors and that (2) training samples made costly by poor performance (e.g. involving property damage, financial loss, or human injury) be reduced or eliminated. If methods are developed which allow humans to transfer their knowledge via forms of natural communication, both of these critical needs can be addressed.

Recently, the TAMER framework[1] was introduced for designing agents that can be interactively shaped by human trainers who give only positive and negative feedback signals [8, 9, 10]. TAMER creates a model, $\hat{H}$, of the human trainer's hypothetical internal reinforcement function, $H : S \times A \rightarrow \mathbb{R}$, and directly exploits its model to choose actions. Past work on TAMER showed that shaping can greatly reduce the sample complexity required to learn a good policy, can enable lay users to teach agents the behaviors they desire, and can allow agents to learn within a Markov Decision Process (MDP) in the absence of a coded reward function. In addition, the work also suggested that human reinforcement – informationally rich yet flawed – and MDP reward – poor yet flawless – are complementary signals that could be used together when a reward function is available.

TAMER, however, does not allow human training to be combined with autonomous learning based on such a coded reward function. This paper examines how to best combine shaping (via the TAMER framework) with reinforcement learning. Specifically, we focus on the scenario in which a human trainer has already trained a TAMER agent, and the learned human reinforcement function, $\hat{H}$, is available to guide a reinforcement learning agent.

In this paper, we test eight plausible methods for combining $\hat{H}$ with MDP reward in a reinforcement learning algorithm, SARSA(λ) [14]. We deem a TAMER+RL technique successful if, after a large, predetermined number of runs, either the cumulative MDP reward received by the learning agent or its final performance level is greater than it would be by using the RL algorithm alone or by greedily choosing $argmax_a[\hat{H}(s, a)]$, as a TAMER agent would.

By these criteria for success, several methods achieved positive results. We discuss which of these methods are most effective and analyze why some worked and others did not help. Additionally, we identify general patterns among TAMER+RL algorithms from the results.

[1] TAMER is short for Teaching Agents Manually via Evaluative Reinforcement.

The rest of this paper begins with an introduction to TAMER and reinforcement learning in Section 2. We then introduce the candidate TAMER+RL techniques for combining a model of human reinforcement with reinforcement learning in Section 3. We subsequently describe the experimental setup, report our results, and discuss them in Sections 4, 5, and 6, respectively. We situate this research within related work in Section 7 and then conclude.

2. TAMER AND REINFORCEMENT LEARNING

Here we introduce reinforcement learning and the TAMER Framework. We especially focus on the nature of MDP reward versus that of human reinforcement.

2.1 Markov Decision Processes and Reinforcement Learning

We assume that the task environment is a Markov Decision Process 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 and an action, of transitioning to another state on the next time step. 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, and s_{t+1}.

Reinforcement learning algorithms (see Sutton and Barto [14]), seek to learn policies ($\pi : S \rightarrow A$) for an MDP that maximize return from each state-action pair, where $return = \sum_{t=0}^{t} E[\gamma^t R(s_t, a_t)]$. Within reinforcement learning, there are two general approaches to this problem. Policy-search algorithms fix the values of some set of parameters, observe the mean return received for the fixed policy over some number of episodes, and then use a rule to determine what parameter values to try next. The other reinforcement learning approach models the expected return, or value, of a state or state-action pair when following a certain policy. When an action is expected to get the highest return given the current state, that action is typically taken (depending on the exact algorithm), possibly changing the previous policy.

A full survey of RL algorithms is beyond the scope of this paper. In this paper, we focus on using value-function-based methods, such as SARSA(λ) [14], to augment the TAMER-based learning that can be done directly from a human's reinforcement signal.

2.1.1 The MDP Reward Signal

The reward signal within an MDP is often characterized as *sparse* and *delayed*. To illustrate, consider a Markov Decision Process that describes a chess game. Within a typical formulation, the reward function would yield zero reward for any state-action pair that does not terminate the game. A state-action pair that transitions to a win might receive a $+1$ reward, a loss would receive a -1 reward, and a stalemate would receive a 0 reward. This reward signal is sparse because discriminating reward is rarely received. It is delayed because any non-terminating state-action pair, regardless of its quality, receives 0 reward, and so the agent must wait until the end of the game to receive any information from the environment that helps it determine the quality of that state-action pair. From this *impoverished* informational signal, many reinforcement learning agents learn estimates of return to combat the signal's sparse and delayed character.

However, MDP reward can be described as *flawless* by definition; along with the transition function, it determines optimal behavior – the set of optimal polices that, for each state, choose the action with the highest possible return.

2.2 The TAMER Framework for Interactive Shaping

TAMER circumvents the sparse and delayed nature of the MDP reward signal by replacing it with a human reward signal. Though conceptually not flawless, it can be exploited to more efficiently learn good, if not optimal, behavior [10].

The TAMER Framework is an approach to the Shaping Problem [10], which is: given a human trainer observing an agent's behavior and delivering evaluative reinforcement signals, how should the agent be designed to make it leverage the human reinforcement signals to learn good behavior? Put formally:

The Shaping Problem Within a sequential decision-making task, an agent receives a sequence of state descriptions ($s_1, s_2, ...$ where $s_i \epsilon S$) and action opportunities (choosing $a_i \epsilon A$ at each s_i). From a human trainer who observes the agent and understands a predefined performance metric, the agent also receives occasional positive and negative scalar reinforcement signals ($h_1, h_2, ...$) that are correlated with the trainer's assessment of recent state-action pairs. How can an agent learn the best possible task policy ($\pi : S \rightarrow A$), as measured by the performance metric, given the information contained in the input?

The Shaping Problem is restricted to domains with a predefined performance metric to allow experimental evaluation. However, shaping will also be helpful when no metric is defined, as would likely be the case with an end-user training a service robot.

2.2.1 The Human Reinforcement Signal

To motivate TAMER's approach to shaping, we first consider what information is contained in the human reinforcement signal and how it differs from an MDP reward signal. When a human trainer is observing an agent's behavior, he has a model of the long-term effect of that behavior. Consequently, a human reinforcement signal is *rich*, containing information about whether the targeted behavior is good or bad in the long term. Further, human reinforcement is delayed only by two things: how long it takes the trainer to evaluate the targeted behavior and how long it takes to manually deliver the evaluation within a reinforcement signal. Considering that the trainer has a long-term model of any behavior's effects and can deliver reinforcement with a consistently small delay, we come to the motivating insight behind TAMER. Unlike MDP reward, human reinforcement is not sparse[2] – each reinforcement fully discriminates between approved and disapproved behavior – and it is only trivially delayed.

We note, though, that human reinforcement is, in general, fundamentally *flawed*. Humans make mistakes, often have low standards, get bored, and have many other imperfections, so their evaluations will likewise be imperfect.

[2]However, human reinforcement can be sparsely delivered.

2.2.2 The TAMER Approach

Following the insight above, TAMER dismisses the credit assignment problem inherent in reinforcement learning. It instead assumes human reinforcement to be fully informative about the quality of an action given the current state. TAMER uses established supervised learning techniques to model a hypothetical human reinforcement function, $H : S \times A \rightarrow \mathbb{R}$, treating the scalar human reinforcement value as a label for a state-action sample.[3] The TAMER framework is agnostic to the specific model and supervised learner used, leaving such decisions to the agent's designer. However, we conjecture that the models should generalize well to unseen state-action pairs and weight recent training samples more highly, as the human's internal reinforcement function is thought to change as the training session progresses.

To choose actions within some state s, a TAMER agent directly exploits the learned model $\hat{H}$ and its predictions of expected reinforcement. When acting greedily, a TAMER agent chooses actions by $a = argmax_a[\hat{H}(s, a)]$.

2.2.3 Previous Results and Conclusions

We previously [10] implemented TAMER agents for two contrasting domains: Tetris and Mountain Car (see Figure 1 for a plot of their Mountain Car results). We compared the learning curves of the TAMER agents with various autonomous agents (i.e., reinforcement learning agents). In short, we found that the shaped agents strongly outperformed autonomous agents in early training sessions, quickly exhibiting qualitatively good behavior. As the number of training episodes increases, however, many of the autonomous agents surpass the performance of TAMER agents.

This paper aims to combine TAMER's strong early learning with the often superior long-term learning of autonomous agents.

2.3 TAMER+RL

In Sections 2.1.1 and 2.2.1, we concluded that MDP reward is informationally poor yet flawless and human reinforcement is rich in information yet flawed. This observation fits the aforementioned experimental results well. With a richer feedback signal, TAMER agents were able to learn much more quickly. But the signal was flawed and TAMER agents, in some cases, plateaued at lower performance levels than autonomous learning agents.

The TAMER framework does not use MDP reward. This characteristic can be a strength. It allows lay users to fully determine the goal behavior without defining and programming a reward function. However, it can also be a weakness. When the goal behavior is previously agreed upon and a reward function is available, a TAMER agent is ignoring valuable information in the reward signal – information which complements that found in the human reinforcement signal.

In this paper, we ask how one could use the knowledge of a previously trained TAMER agent to aid the learning of a reinforcement learning agent. From previous results, we expect that the gains in performance will be most pronounced in early learning.

[3]In domains with frequent time steps (an approximate rule for "frequent" is more than one time step per second), the reinforcement is weighted and then used as a label for multiple samples, each with one state-action pair from a small temporal window of recent time steps. In previous work [10], we describe this credit assignment method, including how each sample's weight is calculated.

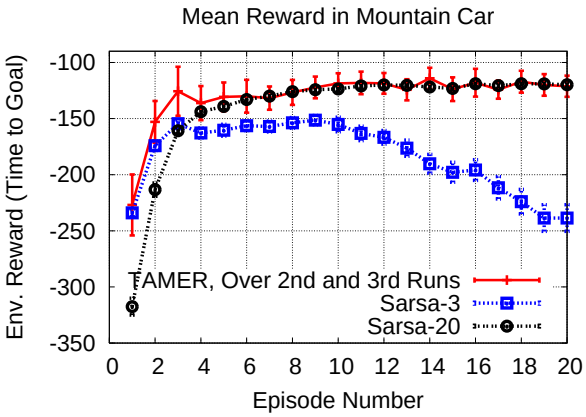


Figure 1: Error bars show a 95% confidence interval (assuming a normal distribution). The mean MDP reward (-1 per time step) received for the Mountain Car task by TAMER agents trained by 19 subjects over two training runs and by autonomous agents using SARSA(λ), using parameters tuned for best cumulative reward after 3 and 20 episodes.

3. CANDIDATE TECHNIQUES FOR COMBINING TAMER AND RL

Here we list and briefly describe the eight techniques for combining TAMER and RL. We assume that the RL algorithm learns a Q-function, where $Q(s, a)$ is an estimate of return from (s,a) under the current policy. Thus our experimental analysis will not apply to policy search algorithms.

One goal of this paper is to create tools that allow human knowledge to be added to reinforcement learning algorithms. For these tools to be practical, they should be generally and easily applicable to any RL algorithm that learns a Q-function. With that in mind, we placed the following restrictions on our methods for combining TAMER with RL:

1. The combination techniques should be independent of the model representations used for $\hat{H}$ and Q.
2. The influence of $\hat{H}$ needs to recede with time or with repeated visits to the same or similar states to keep the set of optimal policies unchanged.
3. We restrict the RL parameters to those which were found (through parameter tuning, described later) to yield the best cumulative performance over 500 episodes. We do not tune parameters for each technique, but rather keep them constant across all techniques.

We list below the eight TAMER+RL methods we tested for combining TAMER with reinforcement learning. Symbols with a prime character ($'$) signify that they are replacing the already existing analogous symbols in either the MDP definition or in the agent's feature vector or Q function.

1. $R'(s, a) = R(s, a) + (weight * \hat{H}(s, a))$. Here, the MDP reward is replaced with the sum of itself and the weighted prediction of human reinforcement. The initial weight is an algorithmic parameter and is decayed by a constant factor at some recurring point. In our experiments, the weight is decayed by a factor of 0.98 at the end of each episode.

2. $\overrightarrow{f'} = \overrightarrow{f}.append(\hat{H}(s,a))$. This technique assumes that the Q-function is learned over a feature vector that is drawn from some state-action pair. To the original feature vector, it adds $\hat{H}(s,a)$ as one additional feature, aiming to embed the knowledge contained in $\hat{H}(s,a)$ in the feature vector such that the RL algorithm can use it as much or as little as is needed. For linear models over the features, to bias initial action towards choosing as the TAMER agent would in early learning, this technique has an input parameter that sets the initial weights corresponding to the added feature (one weight per action).[4]

3. *Initially train* $Q(s,a)$ *to approximate* $(constant * \hat{H}(s,a))$. Treat $\hat{H}$ as the Q function. For our experiments, we randomly created 100,000 state-action pair samples for the Q function to train on.

4. $Q'(s,a) = Q(s,a) + constant * \hat{H}(s,a)$. This method adds a weighted prediction of human reinforcement to Q(s,a) for any occasion that it is used, including for action selection and for updating Q. The unchanging constant is an input parameter.

5. $A' = A \cup argmax_a[\hat{H}(s,a)]$. To the set of possible actions, add the action that the greedy TAMER agent would choose. An input parameter sets the Q value of the new action uniformly across states.

6. $a = argmax_a[Q(s,a) + weight * \hat{H}(s,a)]$. The weighted prediction of human reinforcement is added to Q(s,a) only during action selection. This technique differs from the fifth technique by not affecting updates to the Q function. The weight is an input parameter, and it is annealed periodically by some factor. That factor was 0.98 in our experiments and annealing occurred at the end of each episode.

7. $P(a = argmax_a[\hat{H}(s,a)]) = p$. *Otherwise original RL agent's action selection mechanism is used.* This method effectively either lets the RL agent choose its action normally or has the TAMER agent choose while the RL agent observes and updates as if it were making the choice. The action is chosen by $\hat{H}$ with probability p, where p is annealed periodically. In our experiments, $p = 1$ at start and is annealed by a factor of 0.98 at the end of each episode.

8. $R'(s_t,a) = R(s,a) + constant * (\phi(s_t) - \phi(s_{t-1}), where\ \phi(s) = max_a H(s,a)$. This technique converts $\hat{H}$ into a potential function ϕ to determine supplemental reward.

Note that the first technique treats $\hat{H}(s,a)$ as MDP reward or a component of it, and the eighth technique similarly uses $\hat{H}$ to change the reward signal, but only uses state-specific (and not action-specific) information from $\hat{H}$. The second technique seeks to add the information contained in $\hat{H}$. The third and fourth techniques interpret human reinforcement as estimates of expected return from (s,a). The fifth technique simply gives an extra action and biases the learner towards choosing that action early on. The last two combination techniques either take actions directly from $\hat{H}$ or push action selection towards exploiting $\hat{H}$ without affecting the updates to Q. This approach can be interpreted as using $\hat{H}$ to perform demonstrations (in a sense) for the SARSA(λ) algorithm, creating something akin to imitation learning [1].

[4]This reliance on a linear function is our only violation of the three restrictions above (violating the first restriction). If the input parameter is zero, it does not violate the restriction.

4. EXPERIMENTS

To test the eight techniques for combining TAMER and reinforcement learning, we use SARSA(λ) as our reinforcement learning algorithm and test it within the Mountain Car domain. Mountain Car[5] involves a car, starting at a random location near the the bottom of two adjacent hills, trying to get up one hill to a destination point on top. To gain enough momentum to climb the hill, the car must go back and forth on the hills. Mountain car has a continuous, two-dimensional state space; its variables are position and velocity. Three actions are available: $+c$ acceleration, $-c$ acceleration, and no acceleration. At each time step (other than the final step at the goal), the MDP reward is -1. To approximate $Q(s,a)$, the SARSA(λ) algorithm maintains a linear model over features generated by three (one for each action) two-dimensional grids of Gaussian radial basis functions. At each time step, the linear function approximator updates via gradient descent.

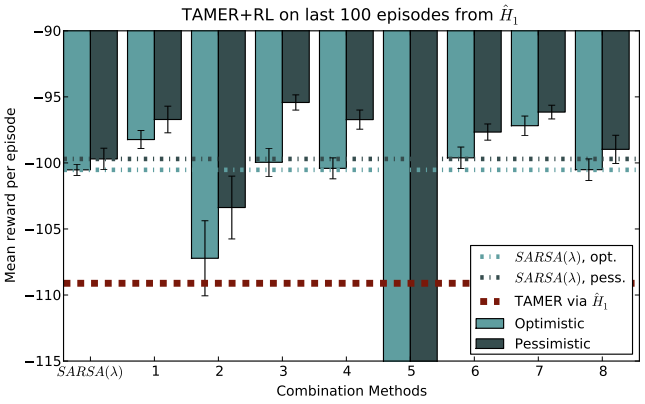
We chose Mountain Car because it is currently the only domain shown to have positive TAMER results and also works well with RL algorithms that incrementally learn value functions. SARSA(λ) with Gaussian RBFs is known to perform well in Mountain Car, though more effective algorithms do exist. However, our goal for this paper is not to study the interaction between TAMER and different RL algorithms but rather to establish that they can be combined effectively and to study the different ways for doing transfer. We have no reason to expect qualitatively different results with other value-function based RL algorithms, but we will investigate possible differences in future work.

We used two sets of parameters for SARSA(λ), each of which had six tuned parameters used by SARSA(λ) or for generating state-action features. We will refer to the two parameter sets as the optimistic set and the pessimistic set. For one set, Q-values were optimistically initialized at 0. For the other set, Q-values were pessimistically initialized, starting at approximately -120 (we had observed that a good policy can reach the goal in roughly 120 seconds or less from any state). Each set was tuned[6] to receive the highest amount of cumulative reward over 500 learning episodes (i.e., to take the least total time getting to the goal 500 times).

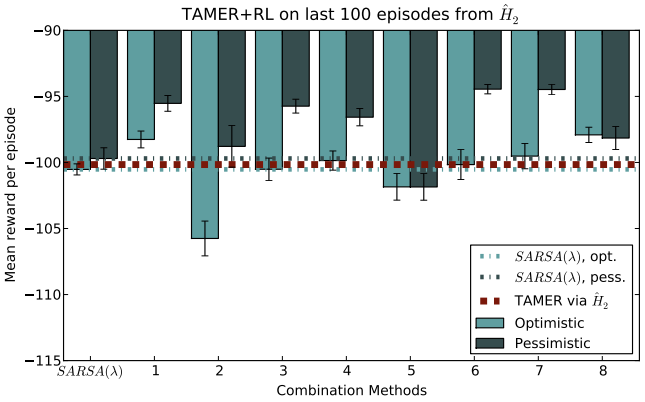
Two training sessions by different trainers provided two static, previously learned human reinforcement functions $\hat{H}$, each outputting within the range [-1.0, 1.0]. One session was chosen because it yielded a function, which we will call

[5]We use an adapted version of Mountain Car from the RL-Library [15].

[6]Parameters were tuned via a hill climbing algorithm that selected one of the features, tested out many different values for that feature, and then repeated. Tuning was stopped when there was no noticeable improvement over a few iterations. The six tuned parameters are the internal discount factor, the step size for updates, λ for eligibility traces, ϵ for ϵ-greedy action selection, the variance of the Gaussian RBFs, and the number of RBF features.

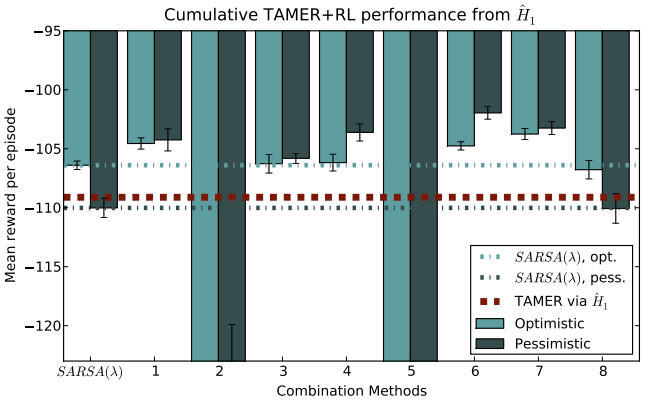


(a)

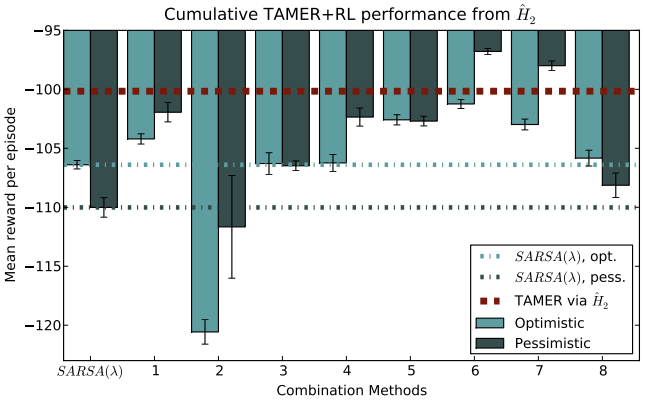


(b)

Figure 2: The mean reward received per episode under all eight techniques over the last 100 episodes of 30 or more runs of 500 episodes. Each bar graph describes experiments using different trained $\hat{H}$ functions. The two $\hat{H}$ functions display the effect of two different levels training quality. The performance of SARSA(λ) under optimistic and pessimistic initialization and the mean reward received by the static TAMER policy exploiting $\hat{H}$ are given for comparison. Error bars show 95% confidence intervals, assuming that mean reward over a run is normally distributed.



(a)



(b)

Figure 3: These bar graphs are identical to those in Figure 2, except that the statistics are calculated over all 500 episodes, allowing us to assess cumulative agent performance under each combination method. (Also note the difference in the range of the y-axes.)

$\hat{H}_1$, that performed near the middle — in terms of total MDP reward — of the functions created by 19 participants, making it representative of a typical trained TAMER agent. The other was chosen because it yielded the best function (called $\hat{H}_2$) of the 19.

For each technique that requires an input parameter, we tested four to six different parameters, all powers of ten. In the following section, we report the results using the best parameter for each technique, as determined by mean cumulative reward.

5. RESULTS

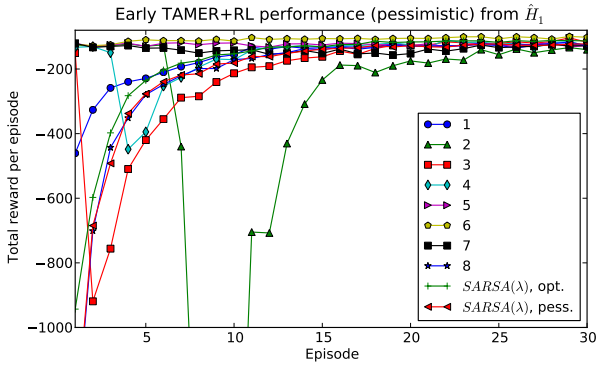
Success of a TAMER+RL combination technique can be defined in two different ways. One is to achieve a higher final performance than either SARSA(λ) alone or by exploiting the static, previously learned $\hat{H}$ directly (like a TAMER agent). The second definition of success is to receive more reward over 500 episodes than either SARSA(λ) or TAMER alone. Since SARSA(λ) alone performed best with optimistic initialization, we will compare the combination methods to optimistic SARSA(λ) exclusively.

5.1 End performance

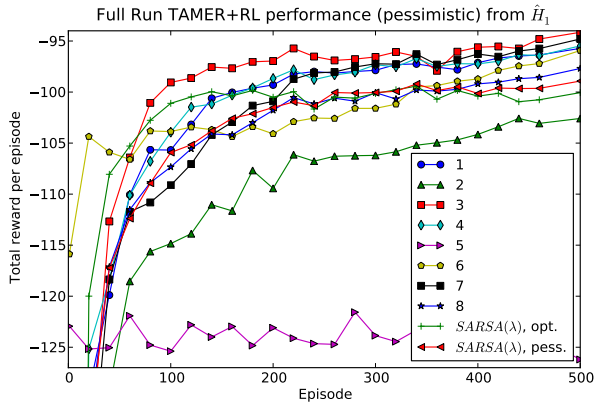
To evaluate the end performance level achieved under each combination, we look at the mean reward received over the last 100 episodes, shown in Figure 2. (Looking at many episodes removes some of the effects of the stochastic initial start state for each episode.) Compared to SARSA(λ), the combination techniques are quite effective at improving final performance. With pessimistic initialization, all of the techniques improve performance except Methods 2 and 5, which are the worst two methods with respect to cumulative reward as well. Also, Method 8 only marginally improves performance with one $\hat{H}$. However, under optimistic initialization, the combination techniques' final performance improves on that of SARSA(λ)'s less often and to a lesser extent than the corresponding pessimistic techniques. Under optimistic initialization, only Method 1 consistently has better final performance than SARSA(λ). Consequently, unless we explicitly state which initialization method we are discussing, the reader should assume that we speak of the pessimistic set, as it yields greater positive results.

5.2 Cumulative reward

From Figure 3, we can evaluate the second condition for success – the mean reward received across all 500 episodes



(a)



(b)

Figure 4: Learning curves using $\hat{H}_1$ and pessimistic initialization. The first plot focuses on the early learning trials. The plot that shows the full run has been smoothed for readability.

during all runs. For the most part, each combination technique performs similarly with both $\hat{H}_1$ and $\hat{H}_2$. Nearly every technique performs better with pessimistic initialization than with optimistic initialization, which we discuss in Section 6.2. The best technique for both $\hat{H}$s is pessimistic Method 6, $a = argmax_a[Q(s,a) + weight * \hat{H}(s,a)]$, followed by the similar Method 7, $P(a = argmax_a[\hat{H}(s,a)]) = p$. Methods 1, $r' = r + (weight * \hat{H}(s,a))$; 6; and 7 are the only techniques which perform better under both initialization types than optimistic SARSA(λ). Method 4, where $Q'(s,a) = Q(s,a) + constant * \hat{H}(s,a)$, also outperforms optimistic SARSA(λ) across both $\hat{H}$s.

Method 3, training $Q(s,a)$ to approximate $(constant * \hat{H}(s,a))$, improves SARSA($\lambda$) a small amount under pessimistic initialization, but does not consistently outperform optimistic SARSA(λ). Method 8 exhibits small improvement under $\hat{H}_2$, but overall is the least influential method. Method 5 is the only one that differs greatly between $\hat{H}$s, showing significant improvement with $\hat{H}_2$ and much worse performance with $\hat{H}_1$. Method 2, adding $\hat{H}(s,a)$ as an additional state-action feature, consistently performs worse than SARSA(λ) alone.

Figure 4 shows learning curves for each technique with $\hat{H}_1$ and pessimistic initialization. The two plots display performance early in the run and across the entire run.

6. DISCUSSION

In this section we discuss which combination methods effectively transferred knowledge from $\hat{H}$ to SARSA(λ). We identify patterns among the results, give explanations for the failures of certain methods, and the impact of using optimistic versus pessimistic initializations.

6.1 Comparing the combination techniques

The relative performance of the TAMER+RL techniques is qualitatively similar across the two $\hat{H}$s. For the "typical" human reinforcement function, $\hat{H}_1$, several of the pessimistic combination techniques improve both cumulative reward and final performance compared to running SARSA(λ) alone or merely acting as a static TAMER agent, satisfying all of our standards for success (Section 4). The same techniques also perform well for the "best" available human reinforcement function, $\hat{H}_2$, outperforming SARSA(λ) for both success criteria and achieving better final performance than the static TAMER agent. However, only Methods 6 and 7 achieve better cumulative reward than the high performance TAMER agent.

To determine whether the SARSA(λ) agents would, if given more time, reach the final performance level of the successful methods, we ran 10 runs of SARSA(λ) over 1000 episodes. The optimistic and pessimistic initializations respectively average -102.809 and -98.842 reward per episode during the last 100 episodes (starting after the 900th). Almost all of the pessimistic combination methods that outperform SARSA(λ) in final performance in the 500 episodes experiments also outperform SARSA(λ) when given 1000 episodes. Therefore combining SARSA(λ) with TAMER actually achieves performance levels that are otherwise unreachable in twice the training time (and possibly ever).

From the results, we notice two patterns:

- *Initially manipulating the model of Q correlates with poor performance.* Three of the worst four combination methods — 2, 3, and 5 — each involve an initial manipulation of the learning model. Method 3 changes the initial parameters of the model. Methods 2 and 5, the two worst performing methods, each change the input space of the model. If this pattern is a factor in performance, a plausible explanation is that the tuned parameters are specific to the model under which they were tuned.

- *Gently pushing the behavior of the learning agent toward what the* TAMER *agent would do and removing the influence of $\hat{H}$ slowly and smoothly correlates with good performance.* Three of the four methods that outperformed SARSA(λ) in both cumulative reward and final performance – Methods 1, 6, and 7 – exert influence on the agent through a weight that decays exponentially. The worst four methods do not use a decaying weight. Further, the two methods that create a form of demonstration (affecting action choice and nothing else), differ in the subtlety of their influence. Method 6 increases the Q-value during action selection by the weight times the predicted human reinforcement. Contrastingly, Method 7 is all-or-nothing in its influence. Either it chooses the action via $\hat{H}$ or frees the SARSA(λ) algorithm choose based on the current Q-function. The gentler Method 6 achieves significantly better cumulative performance (though it performs roughly equivalently during the last 100

episodes).

Aside from the possible influence of the above patterns, there is another plausible reason for the consistent failure of Method 2 (adding $\hat{H}(s,a)$ as an extra state-action feature) across $\hat{H}$s, initialization types, and success criteria (i.e., final performance or cumulative reward). Temporal difference learning algorithms such as SARSA(λ) often learn poorly when using function approximators over features that generalize any learning across large areas of the state-action space [3]. Within the linear model we created over state-action features, changing the weight for the $\hat{H}$ feature during a single update affects *every* state-action pair.

The impotence of Method 8, which uses a potential function over states, $\phi(s)$ to augment reward, suggests that the human reinforcement signal more or less only carries information about the relative desirability of actions given a state, rather than information about the relative value of states. It is possible, though, that the human trainers adapted their reinforcement to the TAMER system, which would not use such state information.

Taking into account both final performance and cumulative reward, Method 6 appears to be most effective. Under both $\hat{H}$s, it yields the best cumulative reward and essentially ties for the best final performance.

6.2 Optimistic versus Pessimistic Initialization

Despite SARSA(λ) performing best with optimistic initialization, once it is combined with these eight methods, it almost uniformly performs best with pessimistic initialization. Upon examining the step-by-step changes in Q-values during the first few episodes for several optimistically initialized methods, we noticed that any method which biases early behavior towards a certain action (given the state) is actually making the other actions have relatively higher Q-values. This effect occurs because when Q-values are optimistically initialized, they can only go down. So by biasing early behavior, we are making the desired behavior look less desirable (from a Q-value perspective). Further, the only way that the agent can learn that the undesired state-action pairs are not as good as their optimistic initial values is by choosing those actions (which are all those actions that the TAMER agent would not make!) and learning that they are overvalued.

Observing that the techniques were not performing especially well under optimistic initialization, we instead tried pessimistic initialization. We set our model weights so that the initial Q-values were all approximately -120. Note that such an initialization is only pessimistic with regards to a good policy. It is actually optimistic for a low-performing policy. With such an initialization, biasing early behavior increases the Q-value for taken actions within a generally good policy and decreases their Q-values within a generally bad policy. We believe that this initialization effect explains why the combination methods improve learning more with pessimistic initialization.

7. RELATED WORK

In this section, interactive shaping and transferring from a learned model of human reinforcement are situated within previous work.

7.1 Interactively Learning from Human Reinforcement (Shaping)

Within the context of human-teachable agents, a human trainer shapes an agent by reinforcing successively improving approximations of the target behavior. When the trainer can only give positive reinforcements, this method is sometimes called *clicker training*, which comes from a form of animal training in which an audible clicking device is previously associated with reinforcement and then used as a reinforcement signal itself to train the animal.

Previous work on clicker training has involved teaching tricks to entertainment agents. Kaplan et al. [7] and Blumberg et al. [2] implement clicker training on robotic and simulated dogs, respectively. Blumberg et al.'s system is especially interesting, allowing the dog to learn multi-action sequences and associate them with verbal cues. Though significant in that they are novel techniques of teaching pose sequences to their respective platforms, neither is evaluated using an explicit performance metric, and it remains unclear if and how these methods can be generalized to other, possibly more complex MDP settings.

Thomaz & Breazeal [19] 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. This approach is similar to our Method 1, $R'(s,a) = R(s,a) + (weight * \hat{H}(s,a))$, differing in that Thomaz & 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).

In another example of mixing human reinforcement with on-going reinforcement learning, Isbell et al. [6] enable a social software agent, Cobot, to learn to model human preferences in LambdaMOO. The agent "uses reinforcement learning to proactively take action in this complex social environment, and adapts his behavior based on multiple sources of human [reinforcement]." Like Thomaz and Breazeal's agent, Cobot does not explicitly learn to model the human reinforcement function, but rather uses the human reinforcement as a reward signal in a standard RL framework.

The TAMER framework is distinct from previous work on human-delivered reinforcement in that it is designed both for a human-agent team and to work in complex domains through function approximation, generalizing to states unseen. It also uniquely forms a model of the human trainer's intentional reinforcement, which it exploits for action selection in the presence or absence of the human trainer.

7.2 Transfer Learning for RL

Transfer learning for reinforcement learning typically focuses on how to use information learned in a source task to improve learning in a different target task. Our type of transfer differs: the task stays constant and we transfer from one type of task knowledge (an $\hat{H}$ function) to a different type (a Q function).

Some approaches create inter-task mappings from a source MDP to a target one. Such mappings have been used in the target task to initialize the Q function [18], as we did in Method 3, and to use recorded (s,a,r,s') experience tuples from the source task to estimate the transition and reward functions for model-based RL [16]. Other approaches derive a policy from the source task and use that policy to guide learning in the target task. For example, Fernandez and

Veloso [5] created an agent that uses a policy from a previous task to aid learning in a task that differed only in its reward function. The agent learns whether to choose from the old policy, choose from the policy it was currently learning, or explore. This approach is also similar to our Method 5. From a policy π_s derived from a learned source task, Taylor and Stone [17] add $\pi_s(s)$ as an extra action (similar to Method 5), increase the Q-value of the $Q(s, \pi_s(s))$ (similar to Method 4), and add a state value that takes the value $\pi_s(s)$ (similar to Method 2), comparing the results of these three techniques [17].

Other forms of transfer can occur within the same task. *Shaping rewards* [4, 11] is changing the output of the reward function to learn the same task, as we did in Method 1. The difference of the shaped reward function and the original one can be seen as the output of a shaping function. With a few assumptions, Ng et al. [12] prove that such a function f, if static, must be defined as $f = \phi(s') - \phi(s)$, where $\phi : S \rightarrow \mathbb{R}$, to guarantee that shaping the reward function will not change the set of optimal policies. Method 8 is a good-faith attempt to convert $\hat{H}$to a potential function for reward shaping. Additionally, Method 1 anneals the output of its shaping function, avoiding Ng et al.'s theoretical constraint.[7]

Imitation learning, or programming by design, has also been used to improve learning within reinforcement learning [13]. Another agent provides demonstrations from its policy while the agent of concern observes and learns, a technique similar to Method 7.

The combination methods which are not mentioned in this section are novel, as is transferring from a model of human reinforcement to a reinforcement learning algorithm.

8. CONCLUSION

In this paper, we introduce, test, and analyze results from eight TAMER+RL methods for combining a model of human reinforcement $\hat{H}$, learned within the TAMER Framework, with a reinforcement learning algorithm, SARSA(λ). We obtain positive results for a number of the techniques. Several combination methods outperform SARSA(λ) alone in both cumulative reward and final performance level for two different $\hat{H}$s. These methods also outperform both $\hat{H}$s in final performance and the more typical $\hat{H}$ in cumulative reward.

One goal of this paper is to provide tools that will allow an agent designer to capture task knowledge from a human trainer – who may not know how to program – and use that knowledge to improve the performance of reinforcement learning algorithms. When using reinforcement learning, we expect that adding TAMER will be most useful for the following types of tasks:

1. Tasks which require much exploration before discriminatory reward is received. This class especially includes tasks in which the path to the goal is complex and zero reward is received until the goal is reached.

2. Tasks in which local maximums make the best solution difficult to find.

3. When the task has a noisy MDP reward signal.

[7]However, Mountain Car violates one of their assumptions, as there are Mountain Car policies that never reach the absorbing goal state.

As a part of our ongoing research agenda, we plan to follow up on the patterns identified in Section 6, testing whether they hold when the task or RL algorithm changes. Lastly, we intend to extend the results in this paper to create methods to simultaneously learn from both a human trainer via TAMER and from RL.

9. ACKNOWLEDGMENTS

The authors would like to to thank Brian Tanner for his extensive support of RL-Glue and RL-Library. 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 National Science Foundation (CNS-0615104 and IIS-0917122), ONR (N00014-09-1-0658), DARPA (FA8650-08-C-7812), and the Federal Highway Administration (DTFH61-07-H-00030). The first author is supported by an NSF Graduate Research Fellowship.

10. REFERENCES

[1] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[2] B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M. Johnson, and B. Tomlinson. Integrated learning for interactive synthetic characters. *SIGGRAPH*, 2002.

[3] J. Boyan and A. Moore. Generalization in reinforcement learning: Safely approximating the value function. *NIPS*, 1995.

[4] M. Dorigo and M. Colombetti. Robot shaping: Developing situated agents through learning. *Artificial Intelligence*, 70(2):321–370, 1994.

[5] F. Fernandez and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. *AAMAS*, 2006.

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

[7] F. Kaplan, P. Oudeyer, E. Kubinyi, and A. Miklósi. Robotic clicker training. *Robotics and Autonomous Systems*, 38(3-4), 2002.

[8] W. B. Knox, I. Fasel, and P. Stone. Design principles for creating human-shapable agents. In *AAAI Spring 2009 Symposium on Agents that Learn from Human Teachers*, March 2009.

[9] W. B. Knox and P. Stone. TAMER: Training an agent manually via evaluative reinforcement. In *IEEE 7th International Conference on Development and Learning*, August 2008.

[10] W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *The Fifth International Conference on Knowledge Capture*, September 2009.

[11] M. Mataric. Reward functions for accelerated learning. *ICML*, 1994.

[12] A. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. *ICML*, 1999.

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

[14] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[15] B. Tanner and A. White. Rl-glue: Language-independent software for reinforcement-learning experiments. *JMLR*, 10, 2009.

[16] M. E. Taylor, N. K. Jong, and P. Stone. Transferring instances for model-based reinforcement learning. *ECML PKDD*, 2008.

[17] M. E. Taylor and P. Stone. Cross-domain transfer for reinforcement learning. *ICML*, 2007.

[18] M. E. Taylor, P. Stone, and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *JMLR*, 8(1):2125–2167, 2007.

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