Retro-TAMER: Retroactive Feedback Assignment within the TAMER Framework

Andrew Wang, Gian Marco Visani

1 Introduction

TAMER (Training an Agent Manually via Evaluative Reinforcement [2]) is a framework that enables a human trainer to interactively shape an agent's policy via reinforcement signals. The human observes the agent as it is interacting with the environment, and, at any given time, they can choose to give either positive feedback or negative feedback, based on whether they want to encourage or discourage certain actions that the agent has performed. The role of the TAMER agent then is to create a predictive model of the human reward it is given. The main goal of TAMER is to reduce the sample complexity for learning a "good" policy, and doing so via a simple and intuitive shaping mechanism that allows lay users to give useful feedback regardless of technical background. We believe the most important contributions of TAMER are the following: its accessibility, as mentioned above, and its ability to push the agent in the right direction at the early stages of training, thus reducing the cost of the initial learning trials, which would otherwise be performing random actions.

We believe the TAMER framework is very promising, and for this very reason we would like to contribute to its development by addressing what we believe are potential areas of improvement. First, we believe that it is too hard for a human to *reliably* evaluate the performance of an agent *online*. In particular, due to both reaction time and analytic ability, it is difficult for a human to quickly evaluate the performance of an agent in real-time of the environment to give feedback, as required by the current TAMER framework. Individually this might lead to contradictory feedback given to the agent. This lack of reliability, or noisiness, of feedback might be what is limiting the peak performance of the current TAMER agents [2].

Second, we believe that the way credit is currently assigned in the TAMER framework is only a general attempt to cover a range of ability of human trainers, and is therefore not very precise. Traditional TAMER agents assign feedback to a set range of past actions for each given piece of feedback, and are unable to adapt to individual trainers. As a result, we expect there to be inconsistent credit assignment across the state-action space due to assignment of credit to past actions that may not have been intended by the human trainer. The imprecise assignment of credit then is also a potential limiting factor with respect to TAMER agent performance.

In this work, we attempt to address these two issues by introducing Retro-TAMER, an extension of TAMER that allows the human trainer to assign feedback to the agent retroactively, i.e. after an episode. The human is allowed to review TAMER's interaction with the environment within any given time frame in a recording, and assign feedback at specific times at their own pace. This way, the human can better review the behavior of the agent and also better estimate the impact of actions without time pressure. As a result, the feedback given is much more reliable and significantly less noisy. Furthermore, with Retro-TAMER the delay in human feedback is not an issue anymore. Feedback can be assigned very precisely to individual actions even within continuous environments, without the need for fine-tuned implementations of credit assignment.
something on findings and something on shortcomings?

2 Background Related Work

Traditional TAMER has a variety of shortcomings, which many others have attempted to address in the past. Li et al. note the unavoidable reliance of TAMER on human trainers and the tendency of trainers to be both unmotivated and lacking in attention spans, detracting from one of the goals of TAMER in terms of creating a information-rich training domain. Their solution similarly deals with impacting the quality of human feedback by changing the user interface, although in a much more additive way by providing visuals demonstrating the agent's uncertainty and past performance in an attempt incentivize trainers to give higher quality feedback at a higher frequency, and were able to show significant improvement in agent performance when trainers were provided with information on past performance [4]. In part, our motivation for continuing to look for improvements on top of TAMER is a following paper by Knox and Stone [3]. In this paper, learning performed by a TAMER agent is transferred to an RL agent that learns in an MDP framework with environmental reward. Knox and Stone demonstrate that this TAMER + RL method outperform SARSA(λ) in both cumulative reward and peak performance. Indeed, as demonstrated in the original paper on TAMER [2], TAMER is able to provide a significant jumpstart on learning in the early episodes of reward-sparse tasks. For example, in Tetris, an agent trained with TAMER is able to clear 65.89 lines on average by the third episode, whereas traditional RL agents based on Policy Iteration do not clear any lines at least until the 100^{th} episode [2]. This work truly shows the potential of TAMER in aiding the learning of reward-sparse tasks, and thus it motivates us to further improve over TAMER. In fact, this work gives us confidence that having TAMER result in a higher peak performance would potentially allow for better end results for TAMER + RL, to be explored in future research.
Although there does not appear to be existing literature regarding improvements on the precision of human feedback and credit assignment specifically within the TAMER framework, Griffith et al. do address the inherent incon-

sistency of feedback given by human trainers in policy shaping; their approach involves estimating the level of consistency of human trainers, and rely on their ADVISE method to converge to the optimal policy without the need for perfectly consistent feedback [1]. Our approach, rather than relying on eventual and depending on relatively accurate estimates of individuals' consistency, will focus on reducing the inconsistency in the human feedback provided to improve model performance.

3 Theoretical Framework

3.1 Environment

We used Open-AI gym's Mountain Car environment for our experiments. In Mountain Car, a simulated car must get to the top of a hill. The car begins between two steep hills and must go back and forth to gain enough momentum to reach the goal. We chose Mountain Car primarily for two reasons:

1. It has a dense state-action history, and thus suffers from the problem of delayed and especially noisy feedback, which is handled with delayed credit assignment in normal TAMER [2].

2. It is a relatively simple task, which does not take long to perform, and thus it makes performing multiple experiments much more feasible. However, the fact that agent must first move away from the goal state to reach it makes it complex enough to be worth exploring. In fact, it was explored in the original TAMER paper by Knox and Stone.

In this environment, each state is represented by a pair of numbers: the position and the velocity along the x-axis. At each state, the agent can perform one of three actions: accelerate left, accelerate right, or do not accelerate at all.

3.2 Original TAMER

We tried to follow the original implementation from [2] as closely as possible. However, we found the description of the implementation to be not exhaustive enough. At any timestep, the human trainer is allowed to give either positive or negative feedback to the agent, using the right arrow key for the former and the left arrow key for the latter. This feedback is encoded in h; $h = 1$ when the feedback is positive, $h = -1$ when it is negative, and $h = 0$ otherwise. Whenever the human feedback is not zero, it is learned by a linear model trained over non-linear features. The features combine a state and an action as follows: the next state is computed given the action, and the two states are fed into a quadratic kernel to create a 2x2 matrix of features, which are then flattened and used as features for the linear model. For the linear model, we used an SGDRegressor, which allows for iterative training using stochastic gradient descent. At every timestep, on a state s, each of the three actions a is simulated to get the next

state $s^{'}$ given that action. Then, three sets of features $f(s,a)$ are computed using the current state and the three possible next states, and they are passed to the linear model to predict the human feedback for each state-action pair: $\hat{H}(s,a)$. The best action is chosen greedily as $a_{best} = max_a(\hat{H}(s,a))$. Note that this implementation implies a fully greedy action selection ($\varepsilon = 0$) without any regard for future feedback , as in the original implementation [2]. [1]

Delayed credit assignment is implemented exactly as in [2]: human feedback is uniformly distributed across all state-action pairs occurring between 0.2 and 0.6 seconds in the past, from the time human feedback was registered. We refer the reader to [2] for a justification over which this window of time was chosen.

As in [2], the human trainer is informed of the action the agent is choosing at each timestep (in which direction the car is accelerating). This is necessary in this task since it is not obvious from observing the behavior of the agent in the environment alone. An example of the training interface is shown in Figure 1. Note that, since we could not directly change the environment (as explained in the next section), we had to display the actions, as well as the notifications of feedback instances, onto the terminal.

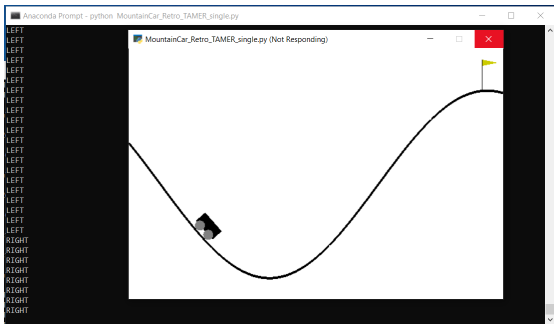

Figure 1: Example of the user interface for training a TAMER agent.

[1] The original implementation used an RBF kernel. However, when we tried it, it usually took only **one** feedback instance correctly placed to result in quasi-optimal behavior by the agent. We do not know why this is the case, but it is fascinating to us. Thus, we opted for a simpler, but still non-linear kernel, in order to allow Retro-TAMER some space of improvement for comparison between models.

Algorithm 1: Greedy TAMER Algorithm

```
Data: Input: stepSize
actions = [left, still, right];
history = [];
ReinfModel.init(stepSize);
while true do
    a ← argmax_a(ReinfModel.predict(getFeatures(s , a)));
    s' ← env.step(a);
    currTime ← time();
    history.update(s, a, s', currTime);
    h ← getHumanReinfSincePreviousTimeStep();
    if h ≠ 0 then
        toAssign ← assignCredit(history, 0.2, 0.6, currTime);
        for (s, a, cred) in toAssign do
            ReinfModel.update(getFeatures(s, a), cred);
            h ← 0;
        end
    else
    end
    s ← s';
end
```

3.3 Retro-TAMER

3.3.1 Design choices and high-level implementation

At a high level, Retro-TAMER is implemented as follows: after an episode assigning feedback with normal TAMER, the human trainer can assign feedback retroactively by replaying the episode, moving forward step by step with the top arrow key. Ideally, we would have liked:

1. to allow for scrolling backwards in time during replay;
2. to allow for the use of retroactive feedback alone.

However, we found that we could not change the variables of the environment manually, and that impeded us from doing either thing. What we could do was to store the history of actions of an entire episode, and simulate the episode using the environments' public functions. By seeding the random number generator, we could initialize the car in the right position at the beginning of the simulation.
Luckily, the first limitation is not a very strong one in this environment. Indeed, mountain car is simple enough that moving backwards to re-analyze an action is generally not necessary. The second limitation was instead stronger. We could not manage to stop the episode at fixed time intervals and replay it. This limitation needs to be addressed to enable Retro-TAMER for continuous

tasks and to enable the use of Retro-TAMER alone. It would be possible to assign feedback retroactively over entire episodes, but it is less feasible in terms of human time since the human trainer needs to wait until the agent has reached the goal via random actions alone. Furthermore, in this case, the action selection of the agent cannot be completely greedy, as it instead is in the original implementation of TAMER, since the agent might get stuck in a loop of unsuccessful actions that will never lead to the goal. In normal TAMER, this is never an issue, since the agent's action selection mechanism is changed during the episode by the human trainer's feedback. We could add randomness to the action selection mechanism, but it would need to be fine-tuned and thus defeat the simplicity of the framework. For these reasons, we chose to allow the human trainer to give feedback interactively, as with normal TAMER, together with retroactive feedback.
Although it might seem unfair to compare the performance at matching episodes, since our Retro-TAMER implementation uses normal TAMER as well, we would like to point out that:

1. a change in peak performance is still relevant
2. evaluating performance at the second episode for Retro-TAMER and at the third episode for TAMER is a fair comparison
3. more feedback does not necessarily translate to better performance, as found in [4].

Therefore, we believe that the performance of Retro-TAMER as implemented can still be compared to TAMER, and the results can still be significant.

3.3.2 Versions of Retro-TAMER

We implemented two versions of Retro-TAMER:

1. **Single:** each feedback instance is assigned to an individual action: namely, the current action taken by the agent, unlike in credit assignment. This is the natural adaptation of TAMER to retroactive feedback assignment: since the dense history of state-action pairs is not an issue anymore, feedback does not need to be delayed nor spread out to multiple pairs.
2. **Multiple:** feedback is spread out uniformly across contiguous timesteps, on a window of 0.0-0.4 seconds before the feedback was given. This is analogous to credit assignment during normal TAMER, but it is not delayed (since there is no need to account for human reaction time/decision speed). In a dense state-action space where it is likely to have sequences of actions equal in value, like Mountain Car, this might be better. It might allow better training with less feedback instances given, but it might be inaccurate or introduce an element of noisiness similarly to traditional credit assignment in normal TAMER, albeit to a lesser degree.

4 Experimental Results

4.1 Experiments

We tested and evaluated three agents: TAMER, Retro-TAMER single-feedback, and Retro-TAMER multiple-feedback. We had multiple human trainers, all of whom trained all the agents. We ensured that the increase in level of skill at training the agents did not bias the data by making the trainers train the agents in different orders. In the end, however, we ended up performing the most tests. Each agent was trained until an episode occurred in which it did not require any feedback to get to the goal quickly. This was assessed by leaving the agent swing for a few seconds at the beginning of each episode, and by starting giving feedback only if it stops making progress after each swing. We define the performance at this episode to be the peak performance. We are aware of the fact that, by this episode, it is not guaranteed that the agent has learned to get to the goal from any starting position (in fact, the starting position is randomly chosen at the beginning of the episode to be somewhere in between the two hills. However, we consider this episode to be an important milestone in the learning process of the agent, and thus to be a good indicator of the learning process of the agent.
We have 18 runs for traditional TAMER, 11 runs for Retro-TAMER single-feedback, and 13 runs for Retro-TAMER multiple-feedback.

4.2 Metrics

To evaluate our models, we considered the following metrics. Average number of timesteps per episode, to see the average behavior of the agent as training progesses. Average number of episodes to get to peak performance; average timesteps at peak performance. Average timesteps at the second episode, to get a sense of the improvement of Retro-TAMER over TAMER after only one round of extra retroactive feedback given.

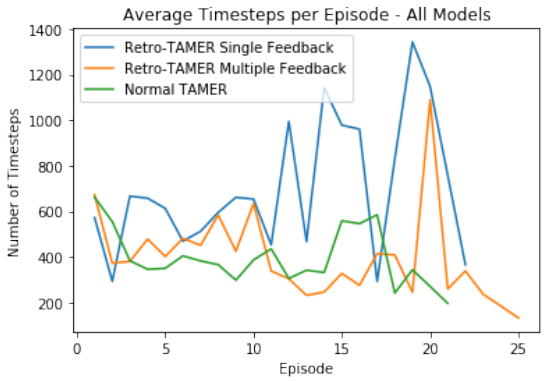

Figure 2: Timesteps per episode for all three trained TAMER models

Agent	Episodes to Peak	Tsteps at Peak	Tsteps 2nd Episode
TAMER	**6.8**	**185.7**	554.9
Retro-TAMER multiple	8.5	229.3	372.8
Retro-TAMER single	10.4	452.7	**293.6**

Table 1: Table summarizing three of the metrics of interest.

4.3 Results

As seen in Figure 2 and the metric table, we were not able to achieve superior average episodes to peak, not timesteps at peak in our focus models comapred to the baseline TAMER model. However, we noted in Figure 2 that the Retro-TAMER models tended to follow a pattern of having relatively high performance in early episodes, before going trough a period of episodes where training appears to "get stuck" and take many more timesteps to get through episodes. We show in the table that at the second episode, both Retro-TAMER models outperform the traditional TAMER model by a very significant amount.
Qualitatively, during training the Retro-TAMER models, we noted that in later episodes, the agents would repeatedly get stuck near the initial position: they would pick a direction to accelerate and only accelerate in that direction, resulting in the agent moving up and down one side of the slope over and over again. In these situations, it can take an excessively large amount of feedback to get the agent "unstuck"; however, once unstuck, the agent is almost immediately able to get to the goal on its own [2]. Characteristics of where the agent tended to get stuck was at low velocities and near the starting initial positions.

5 Conclusion/Future Work

Overall, we were able to implement Retro-TAMER for both single and multiple credit assignment to past actions, and make quantitative comparisons against the traditional TAMER model. In the end, it appears that while Retro-TAMER does not perform significantly better than normal TAMER on Mountain Car, our experiments show that Retro-TAMER can lead to better agent performance very early on in the training process. Our current thoughts on the unexpectedly poor performance of our Retro-TAMER implementation revolve around the nature of the retroactive feedback: during retroactive feedback assignment, the trainer is not able to see the impact of the feedback they assign, and as a result may not be able to assign as accurate or precise feedback as they can during the normal TAMER training process. They may also assign too much feedback in the unlimited time given, which muddles the distribution of data passed into the agent model.
We hope to address the limitations of our model in continued work on the project, and we believe it would be worth the time: Indeed, infinitely precise

[2] Video example of "stuck" training instance, for Retro-TAMER Multiple Feedback: https://drive.google.com/open?id=1Hp4El0RVkY4XH4vV3ElXJxnt2bAr2h6s

and careful feedback can be assigned only retroactively, and we can imagine tasks in robotics in which the human might not be able to give feedback on the fly, but only after a task has been completed. Therefore, figuring out the best practices in retroactive feedback assignment is definitely important.
We think future work can start from the limitations of our work. Having full control over the simulation would allow experiments involving Retroactive feedback assignment alone, as well as the ability to scroll backwards in time in a simulation to better inspect the agent's behavior. A major motivation for solely assigning retroactive feedback is that it would allow us to somewhat bypass the issue of the trainers not being able to see the direct impact of their feedback: currently, the retroactive feedback is only assigned at the end of each episode, but if for example the trainer was able to assign feedback retroactively every 30 timesteps or so, there would be a finer control over the feedback being assigned and a more direct and immediate understanding of the results of their feedback. As discussed earlier on, the gym environment we used for Mountain Car is rather inflexible with respect to allowing for changes of the internal environment without using their predefined functions; to accomplish this then, we would be forced to reconsider our choice of an environment to train the agent in. Furthermore, the human trainer could be allowed to undo feedback that it has assigned, if they realized they have made a mistake. This is not possible to do when feedback is assigned interactively, and it would further reduce the amount of contradictory feedback.
Then, future work should focus on finding out critical points, such as the the instances in which the car got stuck for longer periods of time, as we have highlighted in earlier sections. Tailored metrics and data analysis methods need to be considered in order to find critical points.
Finally, we believe it would be worth to try Retro-TAMER on a more complex environment, such as Tetris. In a more complex environment, critical loops are harder to get to, and the added complexity might allow for the main features of Retro-TAMER (i.e. its reduced noisiness in feedback) to have a bigger impact and show clearer separation of performance between the traditional TAMER model and our focus models. On a similar note, it would be interesting to try Retro-TAMER + RL and see if Retro-TAMER provides a stronger baseline for an RL agent than TAMER does, which would be indicative that Retro-TAMER increases the quality of the agents' learned predictive model.

References

[1] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, pages 2625–2633, 2013.

[2] W Bradley Knox and Peter Stone. Interactively shaping agents via human

reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009.

[3] W. Bradley Knox and Peter Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, AAMAS '10, pages 5–12, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[4] Guangliang Li, Hayley Hung, Shimon Whiteson, and W Bradley Knox. Using informative behavior to increase engagement in the tamer framework. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 909–916. International Foundation for Autonomous Agents and Multiagent Systems, 2013.