Live vs Retrospective TAMER Training

Hasmik Grigoryan, Ryan Fletcher

May 10, 2024

Abstract

TAMER[1] is an algorithm for training a machine learning agent. A human trainer observes the agent's behavior and provides live feedback about the agent's actions. That feedback is used to improve the agent's behavior using online training. In this paper, we propose and evaluate a variation on TAMER which uses retrospective feedback and offline training, leveraging certain advantages that that strategy provides to potentially improve the performance of the resulting agents. We ran an experiment on live TAMER and retrospective TAMER in the same few environments, with nine different human trainers, and found indications that retrospective TAMER might be worth further study.

1 Introduction

1.1 Background

TAMER, introduced in 2008, is an algorithm for training a machine learning agent to perform a specific task. Unlike many prior algorithms for similar purposes, TAMER agents use neural networks to predict not actions or state transitions, but rather the human trainer's short-term reward function at a given state-action pair. That prediction is then used to infer which action to take at each timestep (conventionally through argmax of network predictions over actions).

The feedback used to train the agent is binary; positive and negative. So TAMER can effectively be thought of as "clicker training". In original TAMER, feedback is considered to be provided by the trainer immediately after a given timstep. When the environment runs slowly enough, the feedback is taken to apply to the immediately-preceding prediction. Otherwise, an adjustment must be made in order to assign credit for prompting the feedback signal to some number of the most recent predictions. The original paper used a gamma function to scale feedback over a range of predictions.

1.2 Retrospective TAMER

Proposal:

Our proposed variation of TAMER is very similar to the original. It uses a neural network the same way, the actual action prediction also uses basic argmax, and rapid-timestep feedback credit is assigned according to a gamma function (we formed our own). The functional differences lie in how feedback is provided and how training is handled.

Rather than watching the agent take actions live, the human trainer first watches the agent's entire episodic trajectory from start to finish without providing any feedback. Then the trainer watches that same trajectory again[1] and provides feedback signals to correspond with timesteps wherever they feel it is appropriate. After the feedback stage, the network is trained with all the feedback signals batched together, and then the next episode starts.

Motivation:

A potential flaw in the TAMER algorithm is that it expects the human trainer to provide a certain of feedback, pure of intention. The feedback is supposed to ignore long-term rewards and focus only on immediate "reinforcement" for a given action. However, it's unlikely that human trainers would be so disciplined. We expect that human trainers tend to have at least some focus on future gains, even when instructed otherwise. Therefore, the "clicker training" strategy may be more effective with actual human feedback (rather than programmatic automated feedback) when future utility of potentially unintuitive actions is de-obfuscated by knowledge of the complete trajectory.

2 Methodology

Our code can be found here.

[1] Discussed in Limitations section

2.1 Development

We created an experimental setup that would allow both live and retrospective TAMER to be used to train agents on each of three games: Mountain Car, Tic-Tac-Toe, and Snake. All three games required environment wrappers for gym compatibility and source code modification for experimental functionality.

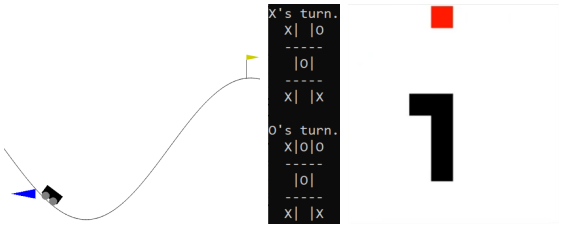

Figure 1: A Screenshot of Each Game

Mountain Car:
The typical Mountain Car game. At each timestep, the agent is given the car's current x-position and velocity, and provides acceleration direction. We added a blue arrow to indicate the car's most recent action, for the sake of making is easier for the human trainers to provide feedback based on the car's action rather than its velocity. Neural network inputs were the current position and velocity plus the logits for the proposed action (left, nothing, right). The 'nothing' action prediction was always ignored, for the sake of training brevity. A pre-trained moderately-performant model was used to start with in each mode, to make it easy to see any performance improvements or deteriorations.

Tic-Tac-Toe:
Tic-Tac-Toe on a 3×3 board. The opponent is an untrained, unchanging random agent. The player agent always moves second. Neural network inputs were logits corresponding to the value (empty, X, O) in each tile plus the logits for the proposed action (one tile of nine). Action predictions for invalid moves were always ignored, for the sake fo training brevity.

Snake:
The Snake game on a 10×10 board. The snake starts at length 1. Rather than dying on walls, the snake loops around to the opposite wall. Neural network inputs were from [2]:

(0/1) – Danger straight, danger right, danger left, facing west, facing east, facing north, facing south, food westwards, food eastwards, food northwards, food southwards, plus the logits for the proposed action (left, straight, right).

A pre-trained moderately-performant model was used to start with in each mode, to make it easy to see any performance improvements or deteriorations. **Experiment:**

For feedback, participants were instructed to "press the 'c' key when the AI makes a suboptimal move and press the 'v' key when the AI makes an optimal move". This wording was chosen to focus the trainers on short-term action-based gains like the original TAMER paper intended without revealing any under-the-hood algorithmic details that may have prompted the participants to try to game the system. Some of the experimental participants were out of the Amherst area, so instructions were compiled to have them download and install the software required to run the experiment (Python, Conda, some packages, and our source code). The exact experimental procedure is:

1. Participant is greeted with a message that tells them exactly what to expect and what they need to do.
2. Participant is given an opportunity to play a couple runs of Mountain Car (in full control) to familiarize themselves with the game's dynamics. (We found this necessary in order to have any chance of successful training from those unfamiliar with the game.)
3. Participant does live or retrospective portion of experiment (chosen at random).
4. Participant does other portion of experiment.

Live portion:

1. Participant greeted with instructions about how the live feedback procedure works (feedback is given whenever appropriate and as often as appropriate, and is incorporated immediately by agent).
2. Participant instructed on how to give feedback for Mountain Car.
3. Participant performs two practice episodes giving feedback to Mountain Car, then three actual training episodes.
4. Repeat 2-3 for Tic-Tac-Toe, then for Snake.

Retrospective portion:

1. Participant greeted with instructions about how the retrospective feedback procedure works (episode trajectory is displayed without feedback, then same trajectory is displayed again, during which feedback is given whenever appropriate and as often as appropriate, and is incorporated by agent after the end of the episode).
2. Participant instructed on how to give feedback for Mountain Car.

3. Participant performs a practice episode pair for Mountain Car, then three actual training episode pairs.
4. Repeat 2-3 for Tic-Tac-Toe, then for Snake.

2.2 Data Collection

During all feedback provision stages of the experiment, our software tracked which feedback signals were sent and at what timesteps of the episode. We also saved a checkpoint of the active neural network after each episode (after each batch training event in the case of the retrospective mode). This information was all saved to .json files associated with specific participants (the participants were allowed to give themselves aliases after they finished experimenting, for the sake of anonymity).

2.3 Data Analysis

To analyze the performance of the trained networks, each checkpoint network was used for 100 runs of the game. We logged the average performance and the standard deviation per game per checkpoint. The following performance metrics were used.

- Mountain Car : Percent progress towards the flag. (i.e. scaled maximum x-axis value)
- Snake : Number of apples eaten
- Tic Tac Toe: -1 for a game lost, 0 for a tie, and 1 for a win

In addition to performance, we looked at the frequency and timing of feedback provided. The logged jsons were compiled into an easy to analyze tabular format, the csv of which can be found on our github.

3 Results

We were most interested in the difference in performance and feedback between live and retrospective versions of the games.

3.1 Performance

We found that performance often suffered from feedback alltogether. We hypothesize that this is due to the fact that many of our participants had very little to no experience with ML or RL. As such they may have lacked the intuition of how to train the networks. Nevertheless, we can look at the difference between live and retrospective approaches to training.

In Figure 1 we can see that retrospective feedback results in slightly better models than live feedback. For Mountain Car checkpoints 1 and 3 retrospective networks outperform live networks by a margin large enough that it may reach statistical significance. For Snake the average difference is very small, which may be a result of overall badly performing models. And finally for Tic-Tac-Toe while the margin is not very large we see that retrospective feedback actually improves performance over more training rounds while live does not.

In Figure 2 we can look at performance difference between live and retrospective networks per user, without averaging. Here any markers that are below 0 line imply that the corresponding user at the given checkpoint trained a better network using Retrospective feedback than Live. For Mountain Car we can see that the majority of markers are below the 0 line. For snake the results are fairly evenly distributed below and above 0 line. This was expected given the lack of significant difference in the average performance plot. For Tic-Tac-Toe the results are drastic and retrospective performs better for almost all users.

3.2 Frequency

We saw that retrospective feedback performs slightly better on average. We are interested in finding the root cause of that. Do people provide more accurate or more frequent feedback when using retrospective feedback environments? To answer this question we first look at the average frequency of feedback for live and retrospective feedback modes. We define feedback frequency as the number of feedbacks (key presses) divided by number of frames in the game. Because we require feedback for every action in Tic-Tac-Toe, this metric only applies to Mountain Car and Snake.

In Figure 3 above we can see that feedback frequency increases slightly over checkpoints. More importantly, participants provide slightly more frequent feedback in retrospective feedback environments than live. We expected this difference to be more drastic, but it's noteworthy nevertheless. The more frequent could be the reason behind slightly better performing models. However, given that participants were not often successful in training networks overall, more frequent feedback would not necessarily result in better models. So we take a look at the type of feedback users provided to find more clues.

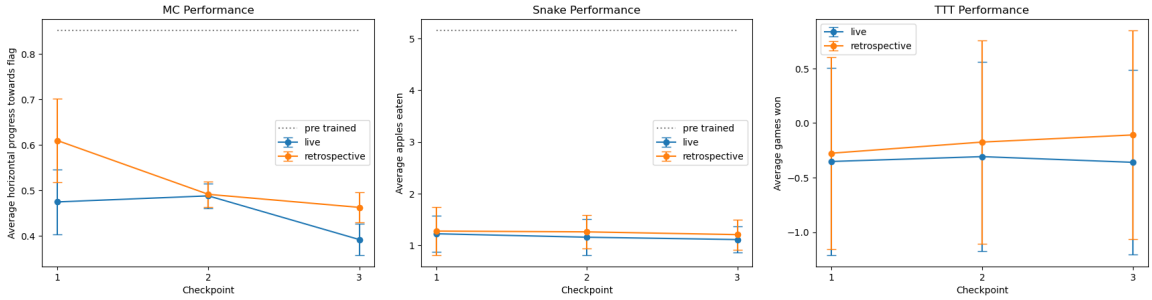

Figure 2: Live vs Retrospective Average Performance.

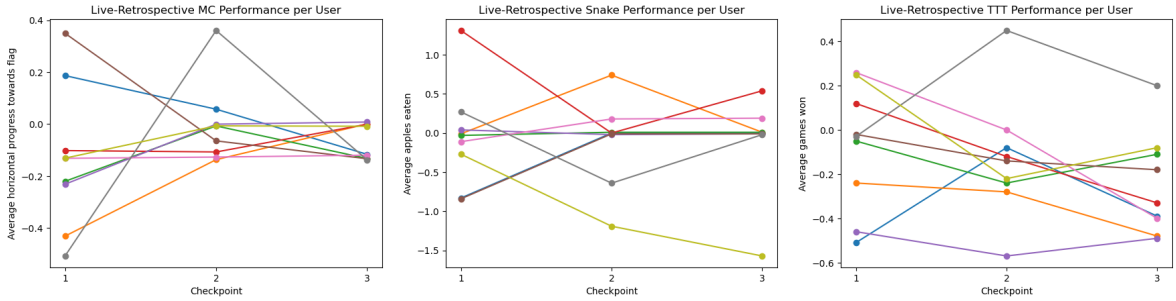

Figure 3: Live - Retrospective Performance per User.

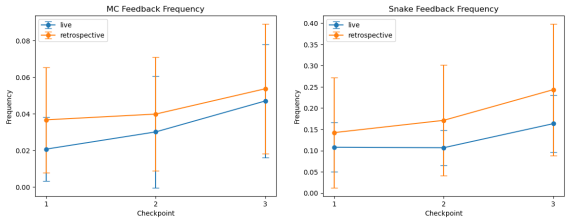

Figure 4: Live vs Retrospective Average Feedback Frequency

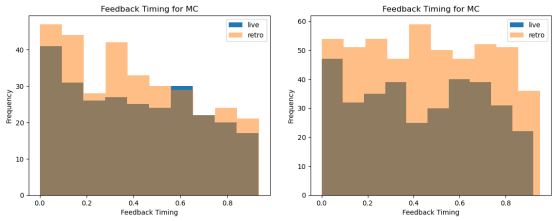

Figure 5: Live vs Retrospective Timing

3.3 Timing

To analyze timing of feedback we look at how far into the game users provided feedback. We define timing as the index of the frame for which feedback was provided divided by total number of frames in the game.

In Figure 4 we see that users provided relatively uniform timing from the beginning of the game to the end. There are no drastic differences in the timing of feedback in live vs retrospective modes.

A more interesting result shows however when we look at interaction between type of feedback (good/bad) and mode of feedback. In Figure 5 we see that for both games: Mountain Car and Snake, people were much more prone to giving bad feedback towards the beginning of the game, when they were providing live feedback. This patterns is not visible in the retrospective mode. In retrospective there's a much better balance between good and bad feedback throughout the game. This result was very interesting to us because it does not align with our intuition of how our users would perform. Moreover, TAMER can be very sensitive to skewed feedback (e.g. much more negative feedback than positive) so we suspect that the increased performance of retrospective environment came from this phenomenon.

3.4 User Experience

Finally, we discuss the difference we noticed in the experience of training using live vs retrospective modes. As this is more subjective and qualitative analysis, be base it off of our own experience coupled with watching some of our participants behavior.

One advantage of live feedback was the immediate response of the agent to the feedback provided. The advantage for this effect is twofold:

- It made training more entertaining. The interactive aspect of live training: clicking a

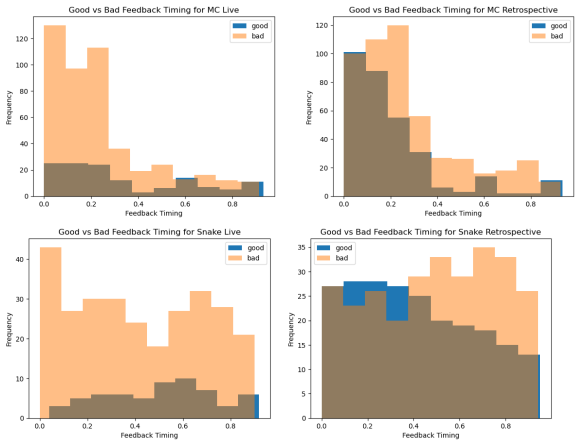

Figure 6: Feedback Mode and Type Interaction for Mountain Car and Snake

button and seeing the agent change behavior, gamified the process and made for a more pleasant training experience. Meanwhile, training using the retrospective environment requires the user to continue to pay attention and give feedback without getting any cues as to what the network is learning.

- Making the agent change behavior resulted in a larger set of states being observed in initial rounds of trainign where the model was not yet capable of playing the game. For instance, in Mountain Car when the model is not yet good it will often get stuck in the gorge. Using live feedback it can be taught to get out of it within a single episode. As such it will observe states and potential feedbacks for a large set of states. Meanwhile, during retrospective feedback the first few rounds of training will have to be focused on getting the car out of the dip before any feedback can be given about how to climb the mountain.

- Humans cannot fall prey to confirmation bias due to seeing the outcome of a game. We did not see evidence of this in our games, however it is possible that in a more complex environment humans may base their judgmenet of specific actions off of whether the game was won or not.

As a tradeoff to these two disadvantages, retrospective feedback offers a few rather significant advantages.

- At every episode, we see the true performance of the agent: how it would perform without a human in the loop. A network that is very sensitive to feedback could potentially simply do the opposite of what it was doing every time it gets bad feedback from a human. We ran into such a network when training mountain car and were able to consistently use it to win the game using live feedback. However, the network was not learning how to play the game at all, and was only able to win while live feedback is being provided. This can be a very important pitfall of TAMER environment and mislead people into believing that the agent is performing well when it is completely useless in deployment. With retrospective feedback approach this problem is completely solved.

- As we saw during our quantitative analysis, humans tend to give slightly better and more frequent feedback using retrospective feedback environment. This may be particularly important in more fast paced games or games where multiple strategies can result in a win.

4 Conclusion

Overall, our experiment showed that TAMER may have space to be improved using the retrospective feedback method for some environments. Retrospective feedback helps the humans avoid being mislead into believing that an agent is learning when it is only responding to the latest feedback. We noticed some counter-intuitive behavior around being more biased towards negative feedback in the beginning of the training episodes when training in a live environment. We believe that accounting for such discrepancies may improve TAMER. Finally, we note that TAMER overall proved to be difficult to use for training a network. One of the biggest attraction points of this method is that humans do not need to have ML experience to be able to train a good model. In our experience this was not the case. Furthermore, we believe TAMER is very sensitive to incorrect or biased feedback can only be used for very simple environments.

5 Limitations

We were limited in time and resources for this project which resulted in some limitations worth mentioning.

- We had a total of 9 participants, the majority of whom did not have much experience with machine learning. We believe a version of this experiment with significantly more participants would help us identify which of the

phenomena observed is due to lack of understanding of the training process and which is a notable pattern in the way humans train models using TAMER. Additionally, having the resources to educate our participants in how the models work, or giving them more practice runs could be helpful.

- In order to limit the experiments to under 20 minutes per participant we had to give the participants a pretrained network and allow for 3 training episodes. This could be an issue if participants wanted to train a vastly different strategy than what we used. If our participants had more time or were compensated for their efforts we could give them an untrained network and ask them to train each network for 5-10 episodes. This would result in more generalizable analysis.

- One of the important aspect of retrospective feedback is that the participants do not have to be quick on their feet. This was not necessarily the case in our experiment because they did not have the interface to rewind the game or edit their feedback. Instead the reaction time requirement was only slightly mitigated by showing them the trajectory of the run ahead of time. They still had to give feedback on time during a regular speed run of the game.

- Finally we note that our training environments are rather simple and our analysis was more focused on which feedback mode resulted in better performance where none of them were particularly good. We cannot claim that these results would generalize to scenarios where participants are able to train good models or to environments with more complex strategic games.

References

[1] 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.

[2] E. Byrd, "Qlearning: Teaching ai to play snake," 2023. https://8thlight.com/insights/qlearning-teaching-ai-to-play-snake [Accessed: May 10, 2024].