Interactive Shaping of a Tetris Agent Using the TAMER Framework

W. Bradley Knox and Peter Stone
University of Texas at Austin
{bradknox, pstone}@cs.utexas.edu

Introduction

As computational learning agents continue to improve their ability to learn sequential decision-making tasks, a central but largely unfulfilled goal is to deploy these agents in real-world domains in which they interact with humans and make decisions that affect our lives. People will want such interactive agents to be able to perform tasks for which the agent's original developers could not prepare it. Thus it will be imperative to develop agents that can learn from natural methods of communication. The teaching technique of shaping is one such method. In this context, we define *shaping* as training an agent through signals of positive and negative reinforcement.[1] In a shaping scenario, a human trainer observes an agent and reinforces its behavior through push-buttons, spoken word ("yes" or "no"), facial expression, or any other signal that can be converted to a scalar signal of approval or disapproval. We treat shaping as a specific mode of knowledge transfer, distinct from (and probably complementary to) other natural methods of communication, including programming by demonstration and advice-giving. The key challenge before us is to create agents that can be shaped effectively. Our problem definition is as follows:

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?

[1] We use the term "shaping" as it is used in animal learning literature (in which it was initially developed by B.F. Skinner). There, shaping is defined as training by reinforcing successively improving approximations of the target behavior (Bouton 2007). In reinforcement learning literature, it is sometimes used as in animal learning, but more often "shaping" is restricted to methods that combine the shaping reinforcement signal and the reward signal of the environment into a single signal (Ng *et al.* 1999).

Though our goal is to create agents that can be shaped to perform any task, we restrict the Shaping problem to tasks with predefined performance metrics to allow evaluation of the quality of shaping algorithms.

Expected benefits of learning from human reinforcement[2] include the following:

1. Compared to learning from the environmental reward of a Markov Decision Process, shaping can decrease sample complexity for learning a "good" policy, consuming less resources in real-world domains.
2. An agent can learn in the absence of a coded evaluation function (e.g., an environmental reward function).
3. The simple mode of communication allows lay users to teach agents the policies which they prefer, even changing the desired policy if they choose.
4. Shaped agents can learn in more complex domains than autonomous learning allows.

Previous results, described later, support the first three of these benefits. These results have appeared in three published papers (Knox and Stone 2009; 2008; Knox *et al.* 2009). This technical report describes our framework for agents that can be interactively shaped, discusses previously

[2] In this abstract, we distinguish between human reinforcement and environmental reward within a Markov Decision Process (MDP). To avoid confusion, human feedback is always called "reinforcement."

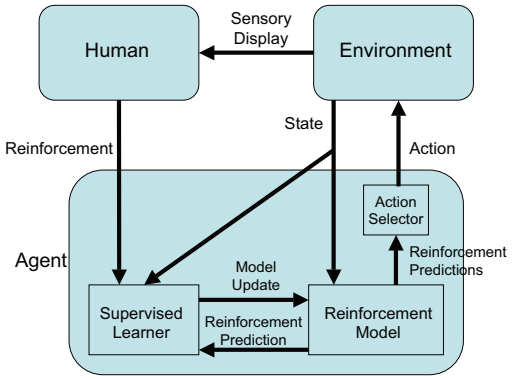


Figure 1: Framework for Training an Agent Manually via Evaluative Reinforcement (TAMER).

published experimental results, and explains our demonstration of an interactively trainable Tetris TAMER agent at the 2009 IJCAI Robotics Exhibition.[3]

The TAMER Framework

In our previous work on shaping, we introduced a framework called Training an Agent Manually via Evaluative Reinforcement (TAMER). The TAMER framework, shown in Figure 1, is an approach to the Shaping Problem that makes use of established supervised learning techniques to model a human's reinforcement function and bases its action selection on the learned model. If acting greedily, a TAMER agent chooses actions that are projected to receive the most reinforcement.

At the highest level of description, the TAMER framework allows a human to train an agent to perform a task via positive and negative reinforcement signals. From the agent's point of view, its goal is to model the human's reinforcement while exploiting that model to earn as much immediate reinforcement as possible. The human trainer's goal is, therefore, to provide reinforcement that leads the agent to perform the task well.

The TAMER framework is designed for Markov Decision Processes that have the reward function R unspecified (MDP\R). A TAMER agent seeks to learn the human trainer's reinforcement function $H : S \times A \rightarrow \mathbb{R}$. Presented with a state *s*, the agent consults its learned model $\hat{H}$ and, if choosing greedily, takes the action *a* that maximizes $\hat{H}(s, a)$. Since the agent seeks only to maximize human reinforcement, the optimal policy is defined solely by the trainer, who could choose to train the agent to perform any behavior that its model can represent. Therefore, when the agent's performance is evaluated using an objective metric, its performance will be limited by the information provided by the teacher.

The principle challenge for autonomously learning agents (i.e., those that receive feedback in the form of environmental reward rather than human reinforcement) is to assign credit from environmental reward to the entire history of past state-action pairs. A key insight of the TAMER framework is that the difficult problem of credit assignment inherent in reinforcement learning is no longer present with an attentive human trainer. The trainer can evaluate an action or short sequence of actions, considering the long-term effects of each, and deliver positive or negative feedback within a small temporal window after the behavior. Assigning credit within that window presents a challenge in itself, which we address by defininining a probability density function $f(x)$ over the time window and distributing "credit" among the actions as follows (for example, see Figure 2). If t and t' are times of consecutive time steps, credit for the time step at t is $h \times \int_{t'}^{t} f(x)dx$. Assuming that credit is properly assigned within the temporal window, we assert that a trainer can directly label behavior. Therefore, modeling the trainer's reinforcement function *H* is a supervised learning problem.

[3]Much of this abstract overlaps with the first author's thesis statement for the AAAI/SIGART Doctoral Consortium and the most recent paper on TAMER (Knox and Stone 2009).

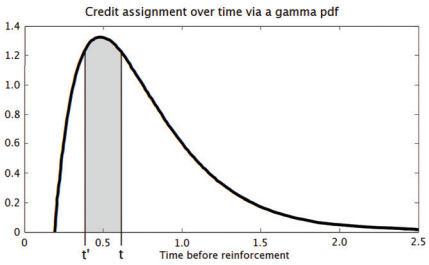


Figure 2: Probability density function $f(x)$ for a gamma(2.0, 0.28) distribution. Reinforcement signal h is received at time 0. If t and t' are times of consecutive time steps, credit for the time step at t is $\int_{t'}^{t} f(x)dx$. Note that time moves backwards as one moves right along the x-axis.

Experimental Results

We developed TAMER algorithms for two contrasting task domains – Tetris and Mountain Car. Full explanations of the implemented algorithms can be found in previous work (Knox and Stone 2009). Tetris has a complex state-action space and low time step frequency, and Mountain Car is simpler but occurs at a high frequency (seven actions per second).

For the experiments, human trainers observed the agents in simulation on a computer screen. Positive and negative reinforcement was given via two keys on the keyboard. The trainers were read instructions and were not told anything about the agent's features or learning algorithm. Nine humans trained Tetris agents. Nineteen trained Mountain Car agents. For each task, at least a fourth of the trainers did not know how to program a computer.

Tetris

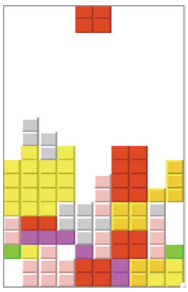


Figure 3: A screenshot of RL-Library Tetris.

Tetris is notoriously difficult for temporal difference learning methods that model a function such as a value or action-value function. In our own work, we were only able to get Sarsa(λ) (Sutton and Barto 1998) to clear approximately 30 lines per game with a very small step size α and after hundreds of games. Bertsekas and Tsitiklis report that they were unable to get optimistic TD(λ) to make "substantial progress". RRL-KBR (Ramon and Driessens 2004) does somewhat better, getting to 50 lines per game after 120 games or so.[4] The only successful approach that learns a value function (of those found by the authors) is λ-policy iteration (Bertsekas and Tsitsiklis 1996), which reaches several thousand lines after approximately 50 games.

However, λ-policy iteration differs in two important ways from the rest of these value-based approaches. First, it begins with hand-coded weights that already achieve about 30

[4]We should note that Ramon et. al. rejected a form of their algorithm that reached about 42 lines cleared on the third game. They deemed it unsatisfactory because it unlearned by the fifth game and never improved again, eventually performing worse than randomly. Ramon et al.'s agent is the only one we found that approaches the performance of our system after 3 games.

Table 1: Results of various Tetris agents.

Method	Mean Lines Cleared		Games
	at Game 3	at Peak	for Peak
TAMER	**65.89**	**65.89**	**3**
RRL-KBR (2004)	5	50	120
Policy Iteration (1996)	~ 0 (no learning until game 100)	3183	1500
Genetic Algorithm (2004)	~ 0 (no learning until game 500)	586,103	3000
CE+RL (2006)	~ 0 (no learning until game 100)	348,895	5000

lines per game. Getting to that level of play is nontrivial, and some learning algorithms, such as Sarsa(λ), fail to even reach it when starting with all weights initialized to zero. Second, λ-policy iteration gathers many state transitions from 5 games and performs a least-squares batch update. All other successful learning methods likewise perform batch updates after observing a policy for many games (as many as 500 in the best-performing algorithms), likely because of the high stochasticity of Tetris. Additionally, of those that have the necessary data available, all previous algorithms that model a function are unstable after reaching peak performance, unlearning substantially until they clear less than half as many lines as their peak.

In our experiments, human trainers practiced for two runs. Data from the third run is reported. Of the algorithms that model an actual function, our gradient descent, linear TAMER is the only one that clearly does not unlearn in Tetris.[5] Of those that also perform incremental updates, TAMER learns the fastest and to the highest final performance, reaching 65.89 lines per game by the third game (Table 1 and Figure 4).[6]

Policy search algorithms, which do not model a value or reinforcement function, attain the best final performance, reaching hundreds of thousands of lines per game (Bohm *et al.* 2004; Szita and Lorincz 2006), though they require many games to get there and do not learn within the first 500 games.

Within our analysis, TAMER agents learn much more quickly than all previously reported agents and reach a final performance that is higher than all other incrementally updating algorithms.

Mountain Car

The Tetris results demonstrate TAMER's effectiveness in a domain with relatively infrequent actions. Conversely, our experiments in Mountain Car test its performance in a domain with frequent actions. The autonomous algorithm used

[5]This statement is supported by training by one of the authors but not necessarily by the subjects, since the trainer subjects usually stopped training after the TAMER agent reached satisfactory performance.

[6]Most trainers stopped giving feedback by the end of the fifth game, stating that they did not think they could train the agent to play any better. Therefore most agents are operating with a static policy by the sixth game. Score variations come from the stochasticity inherit in Tetris, including the highest scoring game of all trainers (809 lines cleared), which noticeably brings the average score of game 9 above that of the other games.

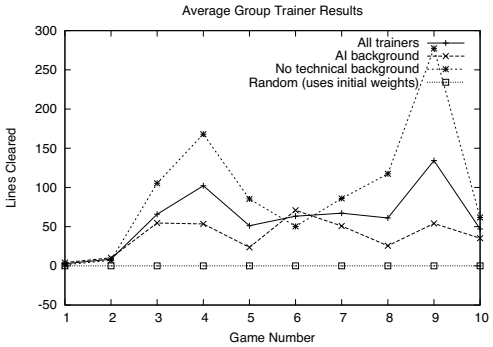
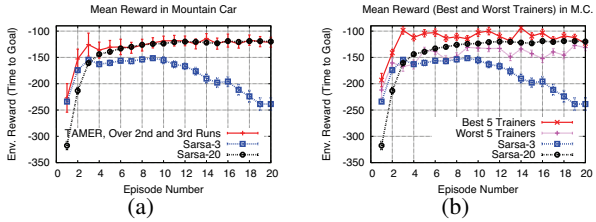


Figure 4: The mean number of lines cleared per game by experimental group.



(a) (b)

Figure 5: Error bars show a 95% confidence interval (with a Gaussian assumption). (a) The mean environmental reward (-1 per time step) received for the Mountain Car task for the second and third agents (runs) shaped by each trainer under TAMER and for autonomous agents using Sarsa(λ), using parameters tuned for best cumulative reward after 3 and 20 episodes. (b) The average amount of environmental reward received by agents shaped by the best five and worst five trainers, as determined over all three runs.

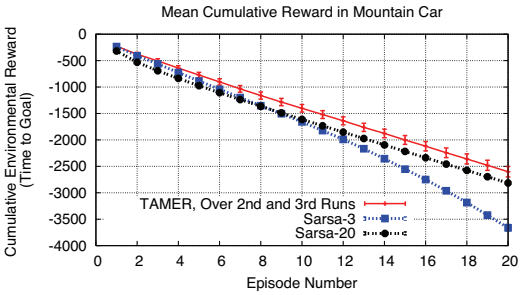


Figure 6: Mean cumulative environmental reward received for the Mountain Car task.

for comparison was Sarsa(λ) (Sutton and Barto 1998) with the same function approximator (a linear model over Gaussian RBF features, using gradient descent updates), an algorithm that is known to perform well on Mountain Car. Two Sarsa(λ) agents were used: one tuned for total cumulative environmental reward across all previous episodes (Figure 6) after 3 episodes and one tuned for after 20 episodes (which we will refer to as Sarsa-3 and Sarsa-20, respectively). We tuned via a hill-climbing algorithm that varied one parameter (α, λ, or ϵ in the standard notation of Sarsa (Sutton and Barto 1998)) at a time, testing the agent's performance under each value for that parameter, taking the best performing value, and then repeating (for fifty or more iterations). The specific number of episodes (3 and 20) were chosen to exhibit different emphases on the trade-off between learning quickly and reaching the best asymptotic per-

formance.

The TAMER agents were shaped for three runs of twenty episodes by each trainer. We consider the first run a practice run for the trainer and present the combined data from second and third runs. Results are shown in Figures 5 and 6. Figure 5(a) shows that the TAMER agents, on average, consistently outperformed the Sarsa-3 agent and outperformed the Sarsa-20 for the first five episodes, after which Sarsa-20 agents showed comparable performance. Under the guidance of the best trainers (Figure 5(b)), TAMER agents consistently outperform any Sarsa agent, and, under the worst trainers, they perform somewhat worse than the Sarsa-20 agent. Most importantly, TAMER agents, on average, also outperformed each Sarsa agent in mean cumulative environmental reward through the length of a run (Figure 6). Since each time step incurred a -1 environmental reward, Figure 6 is also a measure of sample complexity. The four trainers who did not have a computer science background achieved performance as good or marginally better than the fifteen who did. Overall, the results, though less dramatic than those for Tetris, support our claim that TAMER can reduce sample complexity over autonomous algorithms.

The Tetris Demonstration

During the IJCAI Robotics Exhibition, visitors came by and trained Tetris TAMER agents. The agents differed in one significant way from the agents previously used for our experiments. An experimental agent formed a model of human reinforcement, $\hat{H}$, using 21 features over the state and action. These state-action features are the difference of the following *state* features (taken from Bertsekas and Tsitsiklis (1996) from the next state and the current state (i.e. the change in state features resulting from the action):

- the 10 column heights,
- the absolute values of 9 differences in heights of consecutive columns,
- the height of the tallest column, and
- the number of holes on the board, where a hole is defined as an empty cell with a filled cell somewhere directly above it.

However, we added features (taken from Bohm et al. (2004)) since performing our experiments, and consequently, a Tetris TAMER agent that was trained during IJCAI also had the following state features for a total of 46 features:

- the maximum well depth, where a well is a column of empty space surrounded on both sides by filled cells,
- the sum of the depths of all wells, and
- the 23 squares of the above state features.

With the added features, the agent's ability to learn different strategies for different playing situations improves. From anecdotal evidence, a good trainer can get the Tetris agent to average five to ten times as many lines per game as a similar trainer could with the original 21 features. Of those at IJCAI who trained agents, more than half were able (from our subjective assessment) to get the Tetris agent playing well on their first try. (In the experiments, we had trainers do two practice runs before the real training session.)

The training interface was also improved for the demonstration. Reinforcement was administered by a handheld presentation remote, which is less cumbersome than bending over a keyboard.

Conclusion

The TAMER framework, which allows human trainers to shape agents via positive and negative reinforcement, provides an easy-to-implement technique that:

1. works in the absence of an environmental reward function,
2. reduces sample complexity, and
3. is accessible to people who lack knowledge of computer science.

Our experimental data suggests that TAMER agents outperform autonomous learning agents in the short-term, arriving at a "good" policy after very few learning trials. Observations of trainers at the 2009 IJCAI Robotics Exhibition further support this hypothesis. The experiments also suggests that well-tuned autonomous agents are better at maximizing final, peak performance after many more trials.

Given this difference in strengths, we aim to explore how best to use both human reinforcement, H, and, when available, environmental reward, R, relying on the former more heavily for early learning and on the latter for fine-tuning to achieve better results than either method can achieve in isolation. We also plan to test TAMER in additional domains to characterize when an agent designer should and should not use shaping.

References

D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

N. Bohm, G. Kokai, and S. Mandl. Evolving a heuristic function for the game of Tetris. *Proc. Lernen, Wissensentdeckung und Adaptivitat LWA*, 2004.

M.E. Bouton. *Learning and Behavior: A Contemporary Synthesis*. Sinauer Associates, 2007.

W. Bradley Knox and Peter Stone. Tamer: Training an agent manually via evaluative reinforcement. In *IEEE 7th International Conference on Development and Learning*, August 2008.

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*, September 2009.

W. Bradley Knox, Ian Fasel, and Peter Stone. Design principles for creating human-shapable agents. In *AAAI Spring 2009 Symposium on Agents that Learn from Human Teachers*, March 2009.

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

J. Ramon and K. Driessens. On the numeric stability of gaussian processes regression for relational reinforcement learning. *ICML-2004 Workshop on Relational Reinforcement Learning*, pages 10–14, 2004.

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

I. Szita and A. Lorincz. Learning Tetris Using the Noisy Cross-Entropy Method. *Neural Computation*, 18(12), 2006.