# Fast Customization Using Human-Feedback in Assistive Robotics

James Staley

## 1 Introduction

Robots and Assistive Agents that can learn from caregivers or users are much more likely to be used. An agent will leave the factory with a standardized set of abilities, but standard implementations will not handle every users needs. An agent that can alter its performance based on live feedback will be more useful and much more likely to gain acceptance into peoples lives.

Taking this one step further, an agent should be able to adjust its behavior in real-time – fast enough that the user can tell the robot is reacting to their feedback. Unfortunately, user-feedback tends to be sparse and inconsistent. Therefore, it may be helpful to spread the reward from user feedback throughout the tabular value function during planning steps to speed up learning.

In this paper we will adjust the acceleration profile of an assistive arm's predetermined feeding trajectory to align with a user's preferences as expressed through their positive or negative feedback.

## 2 Background Related Work

In reinforcement learning, an agent repeatedly explores a state-action space, building up a policy as it encounters reward. A trained agent uses its learned policy to take the action that maximized its expected future reward given the current state. The reward signal is a defining aspect of reinforcement learning. Typically the reward is given by the environment, but recent work has focused on incorporating human-feedback into training.

Researchers have implemented several different methods for incorporating human-feedback. The learning system may treat human-feedback as an additional source of environmental reward in a process known as reward shaping [3]. In this case when the user gives positive or negative feedback it is simply weighted and summed with the environment reward for the credited transition. This is easy to implement and straightforward, but it doesn't treat human-feedback as anything special. Human-feedback can be considered to contain high-level information about the task, and so there may be some benefit in distinguishing it from environmental reward.

TAMER [2] ignores any environmental reward and uses human-feedback to determine a value function by updating either a table entry (in tabular learning) or a weights vector (in function approximation). This method leads to rapid early learning that can be expanded upon through more human-feedback or other reinforcement learning techniques.

Policy Shaping [1] retains environmental reward, but keeps a separate policy based on received feedback and combines the two when deciding which actions to take. The learned feedback policy must make an assumption about the consistency and frequency of the user's feedback, but otherwise requires fewer parameters than other reinforcement algorithms with comparable performance.

Directly relevant to our use case, Pilarski et al. [4] train an agent to map electrical muscle signals to prosthetic arm motion using human-feedback via an Actor-Critic system. The Actor-Critic system uses an actor, which selects actions and shapes the policy, and a critic, which estimates the value function used to update the weight vector. They use tile coding to approximate $S$ as a binary feature vector $x(s)$.

All of the above methods use simple binary feedback where the user can approve of, disapprove of, or make no comment on the behavior of the user.

# 3 Methodology

We plan to approach this problem using the simplest representation possible, and then scaling up complexity as is required. Initially we'll perform tabular Q-learning (eqn. 1) where our reward will be provided through human-feedback by a TAMER-like algorithm. We aim to learn the user's preferred acceleration profile for a standardized feeding trajectory. We do this in part to further simplify the task and in part to isolate learning a user's preferences from learning how to perform the task at hand.

$$Q(S,A) \leftarrow Q(S,A) + \alpha[Reward + \gamma \max Q(S',A') - Q(S,A)] \tag{1}$$

One of our primary goals is to learn over a large state-action space quickly enough to show improvement in one session with a single user. We'll use Dyna-Q [5] to speed up the effects of feedback on behavior. Dyna-Q keeps a model of our state-action transitions and simulates $n$ actions for each action our agent takes in the real world. This propagates the feedback we get throughout our table and enables a user to see the results of their feedback more quickly. Using Dyna-Q may help us clarify whether user-feedback *should* be propagated throughout a table in the way reward is.

## 3.1 State Representation

The state of the arm is represented by the angular positions, velocities, and accelerations of all n-joints.

$S = \theta_n V_n A_n$

Position, velocity, and acceleration will each be discretized to be represented in a tabular form. For each of $n$ joints we will discretize state values with resolutions, $r_\theta$, $r_v$, and $r_a$. Our table will be of dimensions $n * r_\theta * r_v * r_a$, which works out to hundreds of thousands of states[1].

Fortunately, a smaller subset of this space will actually be visited by the arms path, which will move between predetermined start and goal positions.

### 3.2 Action Representation

We represent our action as an n-dimensional vector holding the change in acceleration of each of n joints. Acceleration changes by either $+increment$, $-increment$, or $no-change$.

### 3.3 Time Representation

We use a more course representation of time for learning than ROS does for controlling the arm. This helps us avoid rapid changes in joint accelerations which could damage the arm and would be confusing to a user. Longer time samples also simplifies the problem of crediting feedback to actions, and allows steadier motions to occur which are likely easier for a user to critique.

### 3.4 Feedback Representation

Initially a user may give a simple *thumbs-up* or *thumbs-down* as is used in comparable research to indicate feedback. If the model works as intended we will expand the feedback to a continuous scale $f \in [-1, 1]$, which we hope could lead to more interesting reserach questions and results.

## 4 Evaluation

We evaluated our trained agents based on two criteria: (1) Does the agent learn from a user's feedback quickly enough to exhibit preferred behavior assuming the user has a static preference[2], and (2) Does the user recognize that the agent has altered its behavior in line with their wishes.

Our first metric was tested using oracles. Oracles have perfect information about the state of the arm and clearly defined consistent preferences for the acceleration profiles for the arms. The oracles gave feedback that guided the agent based on how closely the agents performance matched the oracle's preference. The oracles were configured to give feedback at different rates and consistency to test the robustness of the learning algorithm, and to determine whether it operated quickly enough to be used with human subjects.

[1] Assuming course resolutions for velocity and acceleration (tens of values).

[2] i.e. A stationary policy underlies the user's preference.

Assuming our first metric indicates successful learning, our second metric will be tested with human subjects as part of a complementary Socially Assistive Robotics study.

# 5 Timeline and Individual Responsibilities

This project was undertaken by one person[3], and he is responsible for all deadlines.

- *Week 1* - Build simplest simulation environment. Single joint arm, Tabular Dyna-Q with fixed rewards. A simple 2D visualization is necessary to debug arm-behavior bugs.
- *Week 2* - Oracle implementation and feedback. Scaled up arm to 2-3 joints.
- *Week 3* - 7-DOF Arm with Oracle Feedback. Tweak feedback distribution to speed up learning.
- *Week 4* - Non-binary feedback. Integrate with full robot-arm.
- *Week 5* - Overflow and Catch-Up. Possible user testing.
- *Week 6* - Write Report.

# References


[1] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy Shaping: Integrating Human Feedback with Reinforcement Learning. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2625–2633. Curran Associates, Inc., 2013.

[2] W. Bradley Knox and P. Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pages 292–297, August 2008.

[3] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[4] P. M. Pilarski, M. R. Dawson, T. Degris, F. Fahimi, J. P. Carey, and R. S. Sutton. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–7, June 2011.


[3] In spite of the frequent use of "we".

[5] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.