Framing Reinforcement Learning from Human Reward: Reward Positivity, Temporal Discounting, Episodicity, and Performance

W. Bradley Knox[*]

Massachusetts Institute of Technology
Media Lab

Peter Stone

University of Texas at Austin
Department of Computer Science

Abstract

Several studies have demonstrated that reward from a human trainer can be a powerful feedback signal for control-learning algorithms. However, the space of algorithms for learning from such human reward has hitherto not been explored systematically. Using model-based reinforcement learning from human reward, this article investigates the problem of learning from human reward through six experiments, focusing on the relationships between *reward positivity*, which is how generally positive a trainer's reward values are; *temporal discounting*, the extent to which future reward is discounted in value; *episodicity*, whether task learning occurs in discrete learning episodes instead of one continuing session; and *task performance*, the agent's performance on the task the trainer intends to teach. This investigation is motivated by the observation that an agent can pursue different learning objectives, leading to different resulting behaviors. We search for learning objectives that lead the agent to behave as the trainer intends.

We identify and empirically support a "positive circuits" problem with low discounting (i.e., high discount factors) for episodic, goal-based tasks that arises from an observed bias among humans towards giving positive reward, resulting in an endorsement of myopic learning for such domains. We then show that converting simple episodic tasks to be non-episodic (i.e., continuing) reduces and in some cases resolves issues present in episodic tasks with generally positive reward and—relatedly—enables highly successful learning with non-myopic valuation in multiple user studies. The primary learning algorithm introduced in this article, which we call "VI-TAMER", is *the first algorithm to successfully learn*

[*]Corresponding author.
Email addresses: `bradknox@mit.edu` (W. Bradley Knox), `pstone@cs.utexas.edu` (Peter Stone)

non-myopically from reward generated by a human trainer; we also empirically show that such non-myopic valuation facilitates higher-level understanding of the task. Anticipating the complexity of real-world problems, we perform further studies—one with a failure state added—that compare (1) learning when states are updated asynchronously with local bias—i.e., states quickly reachable from the agent's current state are updated more often than other states—to (2) learning with the fully synchronous sweeps across each state in the VI-TAMER algorithm. With these locally biased updates, we find that *the general positivity of human reward creates problems even for continuing tasks*, revealing a distinct research challenge for future work.

Keywords: reinforcement learning, modeling user behavior, end-user programming, human-agent interaction, interactive machine learning, human teachers

1. Introduction

The constructs of reward and punishment form the foundation of psychological models that have provided powerful insights into the behavior of humans and other animals. Reward and punishment are frequently received in a social context from another agent. In recent years, this form of communication and its machine-learning analog—reinforcement learning—have been adapted to permit *teaching* of artificial agents by their human users (Isbell et al., 2006; Thomaz and Breazeal, 2008; Knox and Stone, 2009; Tenorio-Gonzalez et al., 2010; Suay and Chernova, 2011; Pilarski et al., 2011). In this reward-based form of teaching—which we call interactive shaping (Knox, 2012)—a human user observes an agent's behavior while generating reward instances through varying interfaces (e.g., keyboard, mouse, or verbal feedback); each instance is received by the learning agent as a time-stamped numeric value and used to inform future behavioral choices. Here the trainer considers his or her reward to encompass colloquial concepts like "reward" and "punishment", "approval" and "disapproval", or something similar.[1] We refer to these signals as *human reward*. Teaching by human reward has been employed successfully in numerous task domains, including common test-beds for reinforcement learning like grid worlds, mountain car, and pole balancing (Knox, 2012); tasks for simulated and physical robots (Suay and Chernova, 2011; León et al., 2011; Pilarski et al., 2011; Knox, 2012); and various other domains such as Tetris (Knox and Stone, 2009), a simulated cooking task (Thomaz and Breazeal, 2008), and a text-based online social environment (Isbell et al., 2006).

Interactive shaping enables people—without programming skills or complicated instruction—(1) to specify desired behavior and (2) to share task knowl-

[1]The term "punishment" is used here only in psychological and colloquial contexts. In artificial intelligence, the term "reward" includes both positively and negatively valued feedback.

edge when correct behavior is already indirectly specified (e.g., by a pre-coded reward function). Further, in contrast to the complementary approach of learning from demonstration (Argall et al., 2009; Nicolescu and Mataric, 2003; Grollman and Jenkins, 2007; Nikolaidis and Shah, 2013), learning from human reward employs a simple task-independent interface, exhibits learned behavior *during* teaching, and, we speculate, requires less task expertise and places less cognitive load on the trainer. For an extensive exploration of related work on learning sequential tasks from human teachers, we refer the reader to Chapter 2.5 of Knox (2012).

The long-term goal of this research is to improve upon existing algorithms that learn from human reward, progressing towards algorithms that quickly learn complex behaviors that conform to the trainer's desires. In pursuit of this goal, this article presents the first comparative analysis of methods for learning *exclusively* from human reward. We motivate the experimental approach herein by first noting that interactive shaping can be divided into two steps.

1. Define a learning objective with respect to human reward.
2. Improve on this objective with experience.

The best combination of learning objective and algorithm—as they interact with the reward signals that human trainers actually generate—by definition leads to the best task performance, as judged by the trainer giving this feedback. Thus, the human determines the *task objective*, an objective unknown to the agent—unlike its learning objective—that the human-agent team will ultimately be evaluated upon. (For experimental purposes in this article, however, we instruct trainers to follow specific task objectives.) For example, a trainer might decide to teach a robot to navigate to her by positively rewarding every action that aligns with the behavioral goal of navigating to her and withholding reward for other actions. The task objective, subjectively defined here, is to navigate to the trainer; a potential learning objective is to find a behavioral policy that maximizes the expectation of the sum of future human reward. As we point out in Section 7.1.2, the interaction between the example learning objective and reward strategy will typically not lead the agent to perform well with respect to the task objective in certain settings. But other learning objectives (and other settings) may indeed lead to the desired task performance. We illustrate the relationship between the learning objective and the task objective in Figure 1.

Critically, we emphasize that this learning scenario has two levels of performance: (1) how well an agent maximizes its learning objective, and (2) how well the combination of agent algorithm and learning objective perform according to the task objective. An agent that maximizes its learning objective may perform poorly on the task objective; indeed, this article provides many such examples.

This article primarily attempts to answer the following question: *What learning objective should be chosen to maximize task performance when learning from human reward?* Other questions that receive considerable focus include: *Why do different learning objectives lead to different task performances? When the algorithm cannot quickly maximize the learning objective, how is task performance affected and what implications do these effects on task performance have*

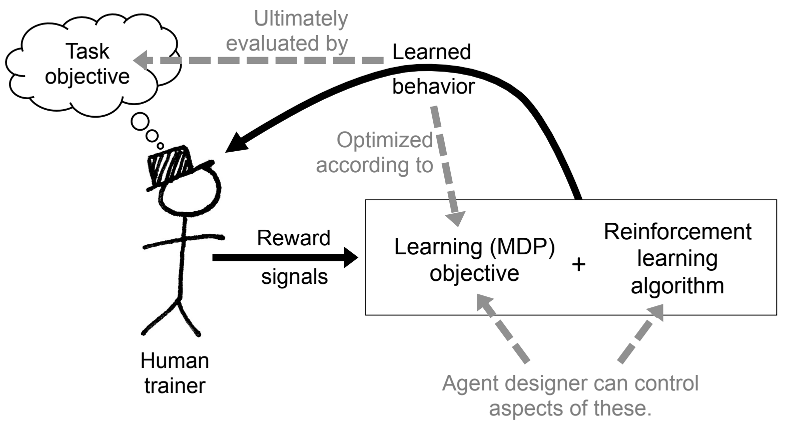


Figure 1. An illustration of the interactive shaping scenario with commentary in gray about the task and learning objectives.

for algorithm design?

To address these questions, our experiments systematically explore how to complete the previously enumerated two steps for learning from human reward. The following three dimensions are considered:

- in defining the learning objective, the choice of *temporal discounting* rate—i.e. how the agent values long-term reward in comparison to near-term reward, where an agent that only values near-term reward is called *myopic*;
- in defining the learning objective, whether the task is considered *episodic* or *continuing* (defined in Section 2);
- and how well the agent optimizes the learning objective while being trained by a human.

Throughout our experiments, we focus on how these dimensions interact to affect task performance. We also emphasize how the dimensions affect reward positivity—whether a trainer gives more positive reward instances than negative ones, and by what ratio—and how reward positivity might act as an intermediate factor in the causal connection between these dimensions and task performance.

This empirical exploration consists of *four new user studies and two experiments on previously collected data*. Though a full list of this article's contributions is given in the Conclusion (Section 11), we highlight the following contributions, all in the context of learning from human reward.

- This article provides empirical support for and justification of the myopic approach used across all previous projects.
- We report the first successful instance of non-myopic learning from human reward and evidence that non-myopic approaches—with further research—will enhance the effectiveness of teaching by human reward.

- Our results provide evidence for the incompatibility of non-myopic learning and episodic tasks for a large class of domains, and conversely provide an endorsement of framing tasks as continuing when learning from human reward.

More generally, this article stands as a case study on adapting a type of machine learning to include human interaction. In this vein, the article serves as an example of how explicit study of the human element can yield insight and performance gains over what would be achieved through naive application of machine learning—without explicit human study.

This article is organized as follows. Section 2 provides background, describing reinforcement learning and the TAMER framework, which will be used by our agents to create predictive models of human reward. In Section 3, we discuss in depth the topic of temporally discounting human reward, including a discussion of past work in the context of discounting. Section 4 presents VI-TAMER, the novel algorithm used in most of our experiments. Section 5 previews our four categories of experimental investigation, each given its own section (7–10), providing a summary of each that can be used as a reference for the reader's convenience. Section 6 describes two baseline versions of a task, an agent, and an experiment, one for each task domain employed in our experiments. Sections 7–10 detail our six experiments, organized by the four investigative categories delineated in Section 5. Section 11 concludes by summarizing the conclusions of this article as contributions and providing recommendations for applying reinforcement learning to human-generated reward.

2. Background and definitions

This section briefly describes reinforcement learning in Section 2.1, focusing on terms that will be important to this article's investigation. It then describes the TAMER framework in Section 2.2. As we explain therein, each of the learning algorithms in this article employs TAMER to create predictive models of human reward as training progresses. These learning algorithms also use the most recent model as a reward function for reinforcement learning.

2.1. Reinforcement learning

As described in Section 1, this article examines the effect of various objectives for learning from human reward on task performance. In particular, we focus on reward-based objectives used in reinforcement learning (RL) for Markov Decision Processes (MDPs) (Sutton and Barto, 1998). MDPs are denoted as $\{S, A, T, R, \gamma, D\}$. Here, S and A are the sets of possible states and actions; T is a function describing the probability of transitioning from one state to another given a specific action, such that $T(s_t, a_t, s_{t+1}) = P(s_{t+1}|s_t, a_t)$; R is a reward function, $R : S \times A \rightarrow \Re$, with a state and an action as inputs; $\gamma \in [0, 1]$ is a discount factor, controlling how much expected future reward is valued; and D is the distribution of start states for each learning episode. RL algorithms seek to learn policies ($\pi : S \rightarrow A$, deterministic in this article) for an MDP that

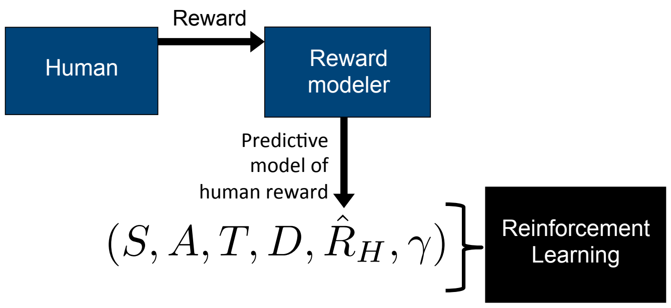

Reward
Human
Reward modeler
Predictive model of human reward
$(S, A, T, D, \hat{R}_H, \gamma)$
Reinforcement Learning

Figure 2. An illustration of the agent's learning algorithm, where inputs *to* the human are not represented. Human reward instances label samples that train the predictive reward model $\hat{R}_H$, which is used as the reward function in an MDP. A reinforcement learning algorithm plans in and/or learns from the resultant MDP, depending on whether the algorithm has knowledge of the transition function and start state distribution.

improve their discounted sum of reward over time—i.e., their *return*—from each state, where return is expressed as $Q^\pi(s, \pi(s))$ and defined in terms of reward as $Q^\pi(s,a) = \sum_{t=0}^{\infty} E_\pi[\gamma^t R(s_t, a_t)]$ (with $0^0 = 1$). The function Q is a called an action-value function. The function V, defined such that $V^\pi(s) = Q^\pi(s, \pi(s))$, is referred to as a state-value function. Both Q and V are collectively called *value functions*. We refer to return-maximizing policies as *MDP optimal*; on the other hand, policies that maximize the task objective are called *task optimal*.

For the experiments described in this article, a Markovian model of human reward, $\hat{R}_H$, is learned from human reward instances. This model completes an MDP specification for the agent to plan in, $\{S, A, T, \hat{R}_H, \gamma, D\}$ (Figure 2). Thus, the output of $\hat{R}_H(s,a)$ for an agent's current state s and action a is the reward directly experienced by the learning agent. In this research, we seek to find reward-based objectives with the following property: algorithms that perform well on the reward-based objective *with reward functions modeled on human trainers' reward* also perform well on the task objective. If we were concerned only with task-optimal behavior (and not with generally improving task performance, reaching optimality or not), this goal could be restated as finding reward-based objectives (and compatible algorithms) such that MDP-optimal behavior is also task optimal.

An important aspect of reward-based objectives is temporal discounting, which is controlled by the γ parameter of MDPs according to the expression of return above. When $\gamma = 0$ the objective is fully *myopic*. A fully myopic agent only values the reward from its immediate state and action; expected future reward is not valued. At the other extreme, an agent with a $\gamma = 1$ objective values near-term reward equally as reward infinitely far in the future. When $\gamma \in (0,1)$, future reward is discounted at an exponential rate, making near-term reward more valuable than long-term reward. Higher γ values result in a lower discount rate.

An additional dimension of MDPs is *episodicity*: whether the task is episodic or continuing. In an episodic task, the agent can reach one or more episode-terminating states, which are called *"absorbing states"* in the RL literature (Sutton and Barto, 1998). Upon reaching an absorbing state, the learning episode ends, a new episode starts with state chosen independently of the reached absorbing state, and the agent experiences reward that is not attributable to behavior during the previous episode. Absorbing states often either represent success or failure at the task, constituting *goal states* or *failure states*; we call tasks with goal states "goal-based". In contrast to an episodic task, a continuing task is ongoing, wherein all reward is attributable to all past behavior, if discounting permits.

As we describe more specifically later in this section, we explore the impact on task performance of the agent's reward-based objective three dimensions: (1) the discount factor, (2) whether a task is episodic or continuing, and (3) whether the agent acts approximately MDP-optimally or is less effective in maximizing its return. Additionally we investigate these dimensions both for tasks that end only at a goal state and for a task that can end either at a goal state or a failure state.

2.2. TAMER framework for learning from human reward

In our experiments, a predictive model of human reward, $\hat{R}_H$, is learned and provides reward for the agent within an MDP specified as $\{S, A, T, \hat{R}_H, \gamma, D\}$ (Figure 2), creating what could be considered model-based RL algorithms. All agents learn $\hat{R}_H$ through the TAMER framework (Knox and Stone, 2009). See Knox (2012) for the current description of TAMER, which we describe briefly below.

The TAMER framework is a fully myopic (i.e., $\gamma = 0$), high-level algorithm for learning from human reward. TAMER consists of three modules: (1) credit assignment to create labels from delayed reward signals for training samples, (2) supervised learning from those samples to model human reward, and (3) myopic action selection using the human reward model. These modules are each described in the following paragraphs. *All of the novel learning agents in this article perform their own action selection; thus they only use the first two of* TAMER*'s modules. However, when these algorithms act myopically by $\gamma = 0$, they are effectively* TAMER *algorithms.*

To model a hypothetical human reward function, $R_H : S \times A \to \mathbb{R}$, TAMER uses established regression algorithms; we call the model $\hat{R}_H$. Labels for state-action samples are constructed from real-valued human reward. The TAMER framework does not commit to any specific model or supervised learner algorithm, 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 reward function is thought to change as the training session progresses. The algorithms employed in our experiments create such recency weighting by learning $\hat{R}_H$ through incremental gradient descent, as we detail in Section 6.

Human feedback is necessarily delayed from the event it targets by our nonzero reaction times. During the credit assignment step, this delay is addressed by spreading each human reward signal among multiple recent state-action pairs, contributing to the label of each. Each sample's weight is calculated from an estimated probability density function for the delay in reward delivery.

To choose actions within some state s, a TAMER agent directly exploits the learned model $\hat{R}_H$ and its predictions of expected reward. When acting greedily, a TAMER agent chooses the action $a = argmax_a[\hat{R}_H(s,a)]$. This is equivalent to performing reinforcement learning myopically with $\gamma = 0$. However, TAMER is used in this article's algorithms as a module for modeling reward; it does not perform action selection in any experiments conducted for this article.

In the problem this article focuses on, the human trainer's feedback is the *only* source of feedback or evaluation that the agent receives. However, TAMER and other methods for learning from human reward can be useful even when other evaluative information is available, as has been shown previously (Thomaz and Breazeal, 2008; Knox and Stone, 2010; Sridharan, 2011; Knox and Stone, 2012). TAMER has additionally been extended to learn in continuous action spaces through an actor-critic algorithm (Vien and Ertel, 2012) and to provide additional information to the trainer—either action confidence or summaries of past performance—creating changes in the quantity of reward instances given and in learned performance (Li et al., 2013).

3. Temporal discounting of human reward

The most salient distinction between all past algorithms for learning from human-generated reward and conventional reinforcement learning algorithms—beyond the obvious addition of an interface and a human—is how the algorithms' learning objectives discount the value of future reward, a choice ultimately made by the agent designer. We devote this section to the rich topic of temporal discounting (defined in Section 2.1) to describe discounting in past work and then discuss the expressivity of a reward function at different rates of discounting, building towards the discounting-based insights we derive from our experiments.

3.1. A myopic trend in past work on learning tasks from human reward

Interestingly, all previous algorithms have discounted more severely than is typical for MDPs. For context, when reward in an MDP comes from a preprogrammed reward function, γ values for discounting are rarely lower than 0.9. Yet for algorithms that learn from reward delivered by an observing human trainer, discount rates trend considerably lower. For episodic tasks, researchers have discounted by $\gamma = 0.75$ (Thomaz and Breazeal, 2008) and $\gamma = 0.9$ (Tenorio-Gonzalez et al., 2010). In continuing domains, $\gamma = 0.7$ (Isbell et al., 2006), $\gamma = 0.75$ (Suay and Chernova, 2011), $\gamma = 0.9$ (León et al., 2011), and $\gamma =$

0.99 (Pilarski et al., 2011) have been used.[2] The $\gamma = 0.99$ work is a non-obvious example of high discounting; with time steps of 5 ms, reward one second ahead is discounted by a factor of approximately 0.134. At the extreme of this trend, the TAMER framework discounts by $\gamma = 0$, learning a model of human reward that is (because of this discounting) also an action-value function. This pattern of myopic maximization of human reward has hitherto not been identified. An additional contribution of this article is to provide sufficient justification for this myopic trend.

In many of these studies, including those on TAMER, learning from human reward is shown to improve in some respect over learning only from MDP reward. Sometimes the championed learning algorithm uses both human and MDP reward and sometimes also a form of action suggestions (Thomaz and Breazeal, 2008; Tenorio-Gonzalez et al., 2010). In most of the others, learning from human reward is shown to be effective in a task where specifying an MDP reward function would be infeasible in the motivating use case (Isbell et al., 2006; Pilarski et al., 2011) (i.e., training a user-specific policy when the user cannot program).

3.2. Consequences of discounting

The two extremes of discounting have different advantages, briefly described in this section.

For $\gamma = 1$, the agent acts to maximize the undiscounted sum of future reward. With this discounting, the reward function could encode a trainer's desired policy, the trainer's idea of high-level task information such as the task goal, or some mixture of the two. Expression of high-level task information permits simpler reward functions. For example, in many goal-based tasks, one such simple reward function would output 0 for transitions that reach a goal state and -1 otherwise. These simpler reward functions with higher-level information could reduce the need for training, allow the agent to find behaviors that are more effective than those known by the trainer, and make the agent's learned behavior robust to environment changes that render ineffective a previously effective policy but leave the purpose of the task unchanged (e.g., when the MDP-optimal path to a goal becomes blocked, but the goal remains unchanged). Given a model of system dynamics (i.e., a transition model) and a planning algorithm, these advantages become even more pronounced.

For $\gamma = 0$, the agent acts myopically to maximize immediate reward. This objective is simpler algorithmically, since the reward function is equivalent to the value function with a discount factor of zero. Thus, setting $\gamma = 0$ effectively reduces reinforcement learning of a value function to supervised learning of a reward function. With supervised learning, the agent can build upon a larger body of past research than exists for reinforcement learning, including tools

[2]The discount factors for three publications were learned through personal correspondence with authors Isbell (Isbell et al., 2006) and Morales (Tenorio-Gonzalez et al., 2010; León et al., 2011).

for model selection (Guyon and Elisseeff, 2003; Arlot et al., 2010). Choosing representations and features for value function approximation is an active area of research (Parr et al., 2008; Mahadevan, 2009; Taylor and Parr, 2009; Boots and Gordon, 2010), but it is comparatively less mature. In general, maximizing a myopic objective is profoundly easier than maximizing a non-myopic objective. A disadvantage of this discounting, on the other hand, is that the reward model can encode a policy but not more general goals of the task. Thus the trainer is forced to micromanage the agent's behavior, placing relatively more burden on the human side of the human-agent system.

Our ambition in this work is to create a natural interface for which people generate reward on their own. Accordingly, we observe that *algorithm designers should choose a discounting level that is compatible with human reward rather than assuming the human trainers will fit their reward to whatever discounting is chosen.* Granted, there appears to be some flexibility in the choice of algorithm: trainers can be instructed before they teach, and humans appear to adapt to the interface and learning algorithm with which they interact. But it may nonetheless be the case that certain intuitively appealing algorithms are incompatible with some or all human training, even after instruction and practice. The rest of this article explores such a possibility.

4. The VI-TAMER Algorithm

VI-TAMER is a novel approach to learning from human reward that makes a powerful experimental tool. It is also the first algorithm to successfully learn non-myopically from human reward (in the continuing-task experiment of Section 8). VI-TAMER consists of two established algorithms running in parallel: (1) TAMER (described in Section 2.2) learns predictive models of human reward from the agent's experienced state-action pairs and human reward instances, and (2) value iteration (see Sutton and Barto (1998)) updates much more often than experienced steps occur, quickly adapting its value function to the most recent reward function changes from TAMER. In VI-TAMER, value iteration is employed as an anytime algorithm, where the current value function is used at any point the agent is asked to choose an action. As is typical for value iteration, the agent chooses actions according to its learned value function and a one-step lookahead for each potential action, which determines state-action values (again, see Sutton and Barto (1998) for details on this process and action selection from state-action values).

Algorithm 1 provides pseudocode for VI-TAMER. The algorithm consists of four parallel threads. The main agent thread initializes variables, starts the other threads, and then selects actions. The value iteration thread updates the value function with the most recent $\hat{R}_H$. The human interface thread listens to the interface to the trainer and distributes incoming reward instances amongst recently experienced state-action pairs. Lastly, the TAMER reward-modeling thread incorporates into $\hat{R}_H$ experience samples that are no longer eligible for a share of incoming reward. For legibility, we do not describe credit assignment in detail. A full description of TAMER's credit assignment is given by Knox

Algorithm 1: The VI-TAMER algorithm

```
Global variables: $V,\ \hat{R}_H,\ E_{hist}$
Global constants: $\{S, A, T, D, \gamma\}$ (an MDP without a reward function)

Main agent thread (initialization and then action selection)
 1: $E_{hist} \leftarrow \{\}$                                   // initialize experience set
 2: Initialize human reward model $\hat{R}_H$ and state-value function V arbitrarily
 3: Start value iteration, human interface, and TAMER reward-modeling
    threads
 4: repeat (for each step)
 5:   $s \leftarrow getState()$
 6:   $a \leftarrow argmax_a \left[\hat{R}_H(s,a) + \gamma \sum^{s' \in S} [T(s,a,s')V(s')]\right]$
 7:   $\hat{h} \leftarrow 0$                        // new experience starts with a label of 0
 8:   $E_{hist} \leftarrow E_{hist} \cup \{(s,a,\hat{h})\}$          // add experience sample to memory
 9:   $takeAction(a)$
10:   wait for next time step

Value iteration thread
 1: repeat (possibly at regular intervals)
 2:   for all $s \in S$ do
 3:     $V(s) \leftarrow max_a \left[\hat{R}_H(s,a) + \gamma \sum^{s' \in S} [T(s,a,s')V(s')]\right]$   // Bellman update

Human interface thread (divides reward amongst recent experiences)
 1: repeat
 2:   if new reward is received from trainer with value $h$ then
 3:     for all $(s,a,\hat{h}) \in E_{hist}$ do                 // add reward shares to labels
 4:       $\hat{h} \leftarrow \hat{h} + (h \times$ proportion of credit for reward from credit assignment$)$

TAMER reward-modeling thread
 1: repeat
 2:   for all $(s,a,\hat{h}) \in E_{hist}$ $t$hat are ineligible for future reward credit do
 3:     $updateModel(\hat{R}_H,\ (s,a,\hat{h}))$  // add completed sample to reward model
 4:     $E_{hist} \leftarrow E_{hist} \setminus (s,a,\hat{h})$       // remove completed sample from memory
```

(2012). We do note though that for any instance of human reward, the sum of the proportions of credit (line 4 of the human interface thread) across all past experiences is 1.

Because the agent learns from a frequently changing reward function, behaving optimally with respect to the current reward function is difficult. For the simple task that we run VI-TAMER on (described in Section 6.2.1), value iteration creates approximately MDP-optimal behavior with small lag in responding to changes to the reward function, a lag of a few time steps or less. Thus, *we can be confident that observed differences between experimental conditions can be attributed to the reward-based objective, not to deficiencies in maximizing that objective.*

Besides VI-TAMER, this article introduces two algorithms that, to our knowledge, are interesting mainly for the experimental role they play here and are described in Appendix B. We do not recommend either algorithm for general use. One is the limited-lookahead Sarsa(λ) algorithm described in Section 6.1. The other, aVI-TAMER, is described in Section 9.1.

5. Preview of experiments and results

Using adaptations of two different baselines for tasks, experimental designs, and agent algorithms, this article includes **six sets of empirical analyses within four investigative categories**. As described in the Introduction, these analyses together seek to find general patterns in the relationship between choice of temporal discounting (i.e., γ) and episodicity in defining the agent's MDP (and thus its learning objective); how close to MDP-optimal the agent can act during training; and task performance. Throughout these experiments, we also examine the relationship of these factors to the degree of reward positivity—the balance of positively valued reward to negatively valued reward given by the human trainer—which in some cases sheds light on how task performance is affected. From our analysis of these experiments, we form a recommendation—stated directly in the Conclusion—for how to frame the problem of learning from human reward. Specifically, the four investigative categories examine the effect on task performance of:

1. in the agent's reward-based objective, varying temporal discounting (γ) in goal-based, episodic tasks (Section 7);
2. in the agent's reward-based objective, varying temporal discounting (γ) in goal-based, continuing tasks (Section 8);
3. when the agent acts non-myopically ($\gamma = 0.99$) in a goal-based task, varying both episodicity and the agent's distance from MDP-optimality (Section 9);
4. and when the agent acts non-myopically ($\gamma = 0.99$) in a task that has both goal and failure absorbing states, varying both episodicity and the agent's distance from MDP-optimality (Section 10).

Each of these four categories contains a user study in a grid world domain (described in Section 6.2.1), and the first and third categories additionally contain analysis from the mountain car domain (described in Section 6.1). Figure 3 indicates the different experimental settings between these categories. We now provide a short summary of each investigative category and its corresponding results.

In our **episodic-task experiments**, we investigate our conjecture (proposed in Section 7.1) that in episodic tasks that are goal-based, a previously

Investigative category	Section	*Temporal discount rate*	*Episodicity*	*MDP optimal*	Terminal state(s)
Episodic task experiments	7	varies	episodic	approximately	goal only
Continuing-task experiment	8	varies	continuing	approximately	goal only
Local-bias experiments	9	$\gamma = 0.99$	varies	varies	goal only
Failure-state experiment	10	$\gamma = 0.99$	varies	varies	goal and failure

Figure 3. A description of the four investigative categories and their corresponding experimental settings. Column headers in italics denote aspects of the learning system that are dependent on the agent algorithm. We include episodicity as dependent—despite it technically being specified by the transition function—because the agent can adapt its experience in an episodic domain to appear continuing, as some of the agents do in Sections 8–10.

observed human bias towards reward positivity will create "positive circuits" (defined in Section 7.1.2) that cause the agent to avoid the goal that it is being taught to reach. In mountain car and grid-world tasks, we observe that task performance peaks at low γ values, plummeting as γ increases to 1, endorsing a myopic approach. In the more rigorously examined grid-world task, we note (1) that at high γs, high-performing agents receive the *most negative* reward, and (2) that 86.6% of trainers do indeed create positive circuits. Both of these observations give credence to our conjecture that reward positivity interacts harmfully with low discounting in episodic, goal-based tasks.

We then repeat one of the episodic-task experiments in a continuing version of the same task, in what we call the **continuing-task experiment**. We make the following four observations. First, we find for discount factors $\gamma < 1$ that task performance of MDP-optimal policies during training is generally high and independent of discounting, a pattern that is quite different from that seen in the episodic setting. Second, several strong correlations observed in the episodic-task grid-world experiment disappear, including the relationships between γ and both reward positivity and task performance. Third, in this investigation, the $\gamma = 0.99$ condition with the VI-TAMER algorithm is the *first known instance of successful non-myopic learning from human-generated reward* (i.e., with a high γ with relatively long time steps). Fourth, in two additional tests using the training data from this continuing-task experiment, we find evidence for the theoretically based conjecture that low discount rates (high γs) facilitate the communication of higher-level task information—e.g., the location of the goal rather than the exact sequence of steps to reach it. Such higher-level information enables learning that is more robust to environmental changes, better guides action in unexperienced states, and has the potential to lead the agent to learn policies that surpass those known to the trainer.

In our third investigation—the **local-bias experiments**—we examine task performance under low discount rates *when the agent cannot generally act approximately MDP-optimally.* These experiments are motivated in part by an anticipation of converting VI-TAMER to more complex tasks for which MDP-optimal behavior is intractable to find. Specifically, we focus on agents that update the values of state-action pairs with a *local bias*, meaning that pairs reachable within a few steps of the agent's current state receive more updates than those further away. Such locally biased updates are a common characteristic among RL algorithms, including Sarsa(λ), Q-learning, and Monte Carlo tree search algorithms. We find that this local bias decreases performance in the continuing setting, apparently because of further problems created by the positivity of human reward. In both experiments, we observe that locally biased agents also achieve higher task performance in continuing tasks than in episodic tasks, though only one of the two differences is statistically significant.

Our fourth investigation—the **failure-state experiment**—boosts the generality of conclusions from the previous experiments by adding a failure state to the goal-based task and then repeating the $\gamma = 0.99$ conditions from these previous experiments. The results from this failure-state experiment follow patterns observed previously, though sometimes with less statistical significance (likely because of smaller effect sizes).

6. Baselines for the task, agent, and experiment for each domain

Here we describe baseline versions of the task, the agent, and the experiment used in each task domain. *Except where otherwise specified*, the corresponding baseline set—for mountain car or grid world—is used in each of this article's experiments. For readability, we keep these descriptions at a high level. We provide relatively less information here on the mountain-car task, since the related user study was conducted for and described in previous work (Knox and Stone, 2009; Knox, 2012). Participant instructions for the grid-world task can be found in the appendix.

In all experiments a model of human reward, $\hat{R}_H$, is learned through the TAMER framework (Knox and Stone, 2009), and the output of this model provides reward for the agent within an MDP specified as $\{S, A, T, \hat{R}_H, \gamma, D\}$. Figure 2 illustrates this scenario. During training, human reward signals form labels for learning samples that have state-action pairs as features; a regression algorithm continuously updates $\hat{R}_H$ with new training samples. Additionally, all $\hat{R}_H$ models and all value functions are initialized to 0.

We employ episodic and continuing versions of both tasks, which are conventionally episodic. In the episodic version of the tasks, the goal state is absorbing. In the continuing version, upon transitioning into the goal, the agent instead experiences a transition to the start state. Consequently, in the continuing version, reward received in one "episode" can be attributed to state-action pairs in the previous "episode" (and farther in the past). Though reaching the goal in the continuing version does not mark the end of an episode, *we continue to use the word "episode" to refer to the periods of time that are divided by the attainment of the goal.* Another valid perspective for the reader is to assume the task is fundamentally episodic and that the continuing version is simply tricking the agent to make it experience the task as continuing.

During training for all experiments, human reward was communicated via two keys on the keyboard that map to 1 and -1. This mapping to 1 and -1, though not infallible, is an intuitive choice that is similar to that of related works that explain their exact mappings (Thomaz and Breazeal, 2008; Tenorio-Gonzalez et al., 2010; Pilarski et al., 2011; Suay and Chernova, 2011). Additionally, this interface allows richer feedback than it superficially appears to for two reasons. First, reward signals are asynchronous to actions, so the rate of reward signaling determines intensity. Second, to account for delays in giving feedback, the causal attribution of each reward is distributed across multiple recent time steps by TAMER's credit assignment module (Knox, 2012), further adding variety to the label values of samples for training $\hat{R}_H$.

Participants were told that the rate of feedback determined its strength and that the agent could accommodate small delays in feedback. They were not told the underlying numeric value of their reward, information that we felt would make the experiments diverge from our envisioned use cases. (Pointers to the full trainer instructions for each type of experiment are given below.)

6.1. *Off-discounting experiments, using pre-trained $\hat{R}_H$s in mountain car*

These analyses in mountain car use 19 fixed $\hat{R}_H$s learned from the training logs created from a past experiment using TAMER (Knox and Stone, 2009), taken from the third run of 19 trainers of the mountain car task. In mountain car, a simulated car must accelerate back and forth across two hills to reach the top of one. Each of these 19 fixed $\hat{R}_H$s provide reward for an RL algorithm at various discount factors. We call these experiments *off-discounting*[3] because the human reward data was gathered under $\gamma = 0$ discounting, which usually differs from the discounting used during evaluation, when the agent learns a value function from the learned reward model. We discuss at the end of Section 7.2.1 possible training bias caused by the mismatched training and testing of the off-discounting experiments. The experiments discussed in Section 6.2 do not involve this discounting mismatch and thus are called "*on-discounting*".

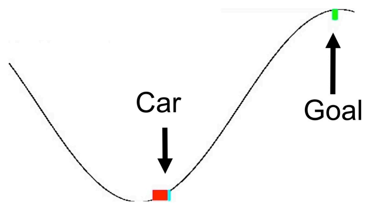


Figure 4. A screenshot of the mountain car task. The action is shown by location of the light blue vertical bar on the left (accelerating left), center (no acceleration), and right (accelerating right, as in this screenshot) of the car.

The $\hat{R}_H$s—the trainer models—are learned with the same linear representation over Gaussian radial-basis-function features that was used during the live training session, updating by incremental gradient descent. Each $\hat{R}_H$ trains on

[3]We chose the terminology "on-discounting" and "off-discounting" to loosely parallel the distinction between on-policy and off-policy reinforcement learning. "Off-policy" means learning about one policy through experience gathered under another policy; "off-discounting" means learning about return under one discount rate from experience gathered while training with another discount rate.

the first 20 episodes of its corresponding training log. To account for a small step size during gradient descent (0.001), each $\hat{R}_H$ is trained from 100 epochs on the trainer log. Credit assignment is performed by the delay-weighted, aggregate reward method described by Knox (2012), updating only when reward was received, as in the reward-only condition described in Chapter 3.4.3 therein.

Whereas VI-TAMER is the most prevalent algorithm in our experiments, the mountain-car experiments—which come first in our results, in Section 7.2.1—instead employ a limited-lookahead Sarsa(λ) algorithm that exhaustively searches a transition tree up to 3 steps ahead. Pseudocode for the algorithm is provided as Algorithm 2 in Appendix B. This algorithm estimates return for each possible immediate action by taking the highest-return path on that action's branch, where a path's return is calculated as the sum of discounted reward along the path and the discounted, learned return at the leaf state of the path. Action selection is similar to ϵ-greedy: there is a probability ϵ at each step that the agent will choose a uniformly random action, and otherwise the action is that with the highest estimated return. Lastly, the depth of the agent's exhaustive tree search is chosen from a Uniform(0,3) distribution at each step to provide a wider range of experiences. The agent updates its value function only on experienced transitions. The Sarsa(λ) parameters are below, following Sutton and Barto's notation (1998). The action-value function Q is represented by a linear model over Gaussian radial-basis-function features. For each action, the means of 1600 radial basis functions are located on a 40×40 evenly spaced grid over the state space, where the outermost means in each dimension lie on the extremes of the dimension. Additionally, an activation feature of 0.1 is added for each action, creating a total of 4803 state-action features. When an action is input to Q, the features for all other actions are zero. The width σ^2 of the Gaussian radial basis functions is 0.08, following Sutton and Barto's definition of a radial basis function's "width" and where the unit is the distance in normalized state space between adjacent Gaussian means. All weights of Q are initialized to 0. The Sarsa(λ) algorithm uses ϵ-greedy action selection, starting with $\epsilon = 0.1$ and annealing ϵ after each episode by a factor of 0.998. Eligibility traces were created as replacing traces with $\lambda = 0.84$. The step size $\alpha = 0.01$. For these experiments, the agents learn from $\hat{R}_H$ for 4000 episodes, and episodes are terminated (with an update of 0 reward) if the goal is not attained after 400 time steps, limiting the agent's maximum return to a finite value.

6.2. On-discounting experiments in a grid world

The grid-world task is used in four of our six experiments and is shown in Figure 5. Unlike with the mountain-car task, these experiments are on-discounting, involving an agent being trained with the same discount factor by which it calculates return when learning a value function (and thus a policy). In other words, the agent's behavior *during training* reflects its discount factor, removing the bias inherent in the mountain-car task.

6.2.1. The grid-world task

The grid-world task contains 30 states. At each step, the agent acts once by moving up, down, left, or right, and attempted movement through a wall results in no movement during the step. Task performance metrics are based on the time steps taken to reach the goal. The agent always starts a learning episode in the state labeled "**S**" in Figure 5. The shortest path from the start state requires 19 actions. Each time step lasts approximately 800 ms.

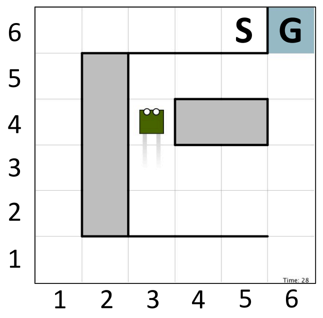


Figure 5. The baseline grid world task. To display the agent's actions and state transitions to the trainer, (1) wheel tracks are drawn from the agent's previously occupied cell, and (2) the simulated robot's eyes point in the direction of the last action. The start and goal cells are labeled 'S' and 'G' respectively.

6.2.2. The grid-world agent

For the grid-world experiments, the baseline agent algorithm is VI-TAMER. An update sweep occurs over all of the states every 20 ms, creating approximately 40 sweeps per step. We restricted update sweeps to these regular intervals to decrease the impact of varying the computational power available to the algorithm, which certainly occurred when our participants interacted with VI-TAMER through their web browsers (see Section 6.2.3). Updates within each sweep are "in place"; i.e., each update to a state value estimate overwrites the previous estimate, allowing updates to affect other updates within the same sweep. As mentioned previously, the agent's value function and reward model are each initialized to output zero at the start of training (with implications discussed in Section 9.1.1).

The TAMER module in VI-TAMER for this task represents $\hat{R}_H$ as a linear model of Gaussian radial basis functions and updates the model by incremental gradient descent. One radial basis function is centered on each cell of the grid world, effectively creating a pseudo-tabular representation that generalizes slightly between nearby cells. Each radial basis function has a width $\sigma^2 = 0.05$, where 1 is the distance to the nearest adjacent center of a radial basis function, and the linear model has an additional bias feature of constant value 0.1. $\hat{R}_H$ is updated with new feedback by incremental gradient descent with a step size of 0.2. In accordance with the most recent version of TAMER (Knox, 2012), we used aggregate reward for credit assignment with a probability distribution over feedback delay of Uniform(-0.4 seconds, -0.15 seconds), with negative values because the algorithm looks backwards in time from the feedback signal to potentially targeted events. Updates occurred at every step regardless of whether reward was provided.

6.2.3. The grid-world experiments

All grid-world experiments were conducted through subjects' web browsers via Amazon Mechanical Turk. Subjects were randomly assigned to an experimental condition. They were prepared with video instructions and a period of controlling the agent followed by a practice training session. During these instructions, subjects are told to give "reward and punishment" to the green "Kermitbot" to "teach the robot to find the water as fast as possible." Trainers were left to determine their own reward strategies, possibly including rewarding every time the agent acts as they would have acted or rewarding highly when the agent reaches the goal.

In the first grid-world experiment (in Section 7.2.2), the official training session stopped after the agent reached the goal 5 times (i.e., 5 episodes) or after 300 steps, whichever came first (unless otherwise specified). In the three other grid-world experiments, trainers were stopped after the first of 10 episodes or 450 time steps.

We *filter* the training samples by removing any sample from analysis that fulfills any of the following conditions: the sample was created from the second or later time that a specific worker acted as a subject, the log file is incomplete, the user mistyped his or her condition-specifying user ID such that the condition was incorrectly specified, or the user gave less than 2 instances of feedback per 100 time steps, which we consider to be non-compliance with our experimental instructions.

Because we repeat many of the same statistical tests throughout this article, we introduce our more common tests here:

- A Fisher's exact test comparing outcomes of reaching the goal all N times or not by condition, where N is specified as a threshold
- A Mann Whitney U test where the dependent variable is the number of episodes completed before training was stopped
- A Mann Whitney U test where the dependent variable is how many steps occurred before the agent reached the goal for *the first time*
- A Spearman correlation test of the ratios of positive to negative reward and success within a specific condition

7. Episodic-task experiments

We now describe our first investigation, focusing on the effect of discounting rates when learning from human reward in a goal-based, episodic task. We first introduce the positive circuits problem and from this problem propose a hypothesis. Then we describe the two episodic-task experiments, discussing their results.

7.1. The positive circuits problem of learning from human reward with high γs

We describe our intuition in two parts for why treating human reward identically to conventional MDP reward in episodic, goal-based tasks—i.e., using γ at or near 1—will often cause minimal task performance, a situation we call the positive circuits problem. In the discussion below, we assume MDP-optimal policies when referring to expected return.

7.1.1. Humans tend to give more positive than negative reward

Thomaz and Breazeal conducted experiments in which humans train agents in an episodic, goal-based task with $\gamma = 0.75$ (Thomaz and Breazeal, 2008). Focusing on the first quarter of the training session—when the agent's task performance is generally worst—they found that 16 out of 18 of their subjects gave more instances of positive reward than of negative reward. Over all subjects and the entire training session, 69.8% of reward instances were positive.

We also examined the balance of positive and negative reward from two previous experiments. One source of data was 27 subjects who taught TAMER agents to play Tetris in the control condition of the "critique experiment" described by Knox et al. (2012). The other source was 19 subjects who taught TAMER agents to perform the mountain car task (Knox and Stone, 2009) as defined in Sutton and Barto (1998). Comparing the sums of each trainer's positive reward values and negative reward values, we find that 45 of the 46 trainers gave more positive reward than negative over their training session. The one exception, a mountain car trainer, gave an equal amount of positive and negative reward. The Tetris agents of eight trainers could not clear even 10 lines a game, in many cases averaging less than a line cleared per game. Yet these trainers still gave more positive reward than negative reward, despite especially poor task performance.

Based on past experiments, human trainers appear to generally give more positive reward than negative reward, often with remarkable consistency.

7.1.2. Consequences of positive reward bias for learning with large discount factors

In many goal-based tasks with discrete state, there exist repeatedly executable trajectories of state-action pairs that start and end in the same state. Additionally, in many goal-based tasks with continuous state variables, there are likewise subsets of states an agent can repeatedly visit, though the exact trajectory may differ each time. We call such repetitious behavior *behavioral circuits*. Such circuits exist for many MDPs, including any deterministically transitioning MDP with at least one recurrent state and any MDP that contains at least one state in which an agent can remain by taking some action. A simple example is an agent walking in circles in a navigational task. For such tasks, given the predominance of positive reward, it is likely that at least one such circuit will provoke positive net reward over each traversal of the circuit. Assuming that the goal-based task is episodic (i.e., a goal state is an absorbing state that ends a learning episode, a large class of problems), the MDP's discount factor γ is conventionally 1. Given that $\gamma = 1$, the expectation of return from states along a net-positive reward circuit will consequently be infinity, since the return is the sum of infinitely repeated positive reward. *Therefore, if a circuit exists with net-positive reward, an MDP-optimal policy for $\gamma = 1$ will never reach the goal*; an absorbing state ends accrual of reward, making the return of a goal-reaching state-action pair finite, regardless of how large the reward is for reaching the goal. Thus, we call this issue the positive circuits problem. The general problem of positive circuits in RL has been discussed previously in the reward-shaping

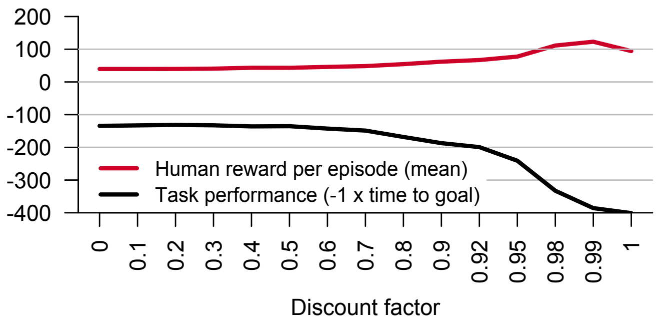


Figure 6. Aggregate results for the off-discounting analysis in an episodic setting, showing mean task performance and mean sum of $\hat{R}_H$-generated reward per episode for mountain car, over the final 500 episodes of 4000 episodes of learning.

literature (Randløv and Alstrøm, 1998; Ng et al., 1999) but to our knowledge has not been connected to human-generated reward or episodicity.

The positive circuits problem is most clearly explained for the case when $\gamma = 1$, but circuits with net-positive reward can also be problematic at high γ values that are less than 1. For instance, if $\gamma = 0.99$ and some circuit exists that has an average reward of 0.5 per transition, the expected return from at least one state in this circuit will be approximately 50 or higher (because $\sum_{t=0}^{\infty}(0.99^t \times 0.5) = 50$). Though finite, such high expectations of return may, despite the trainer's best efforts, be larger than the expectation of return for any path from the state to the goal.

Trainer adaptation may be insufficient to avoid such a goal-averse result; delivering reward such that there are zero repeatable circuits of positive net reward may be severely unnatural for a trainer. Consequently, we hypothesize that RL algorithms using low γs will generally perform better on the trainer's internal task performance metric on goal-based, episodic tasks.

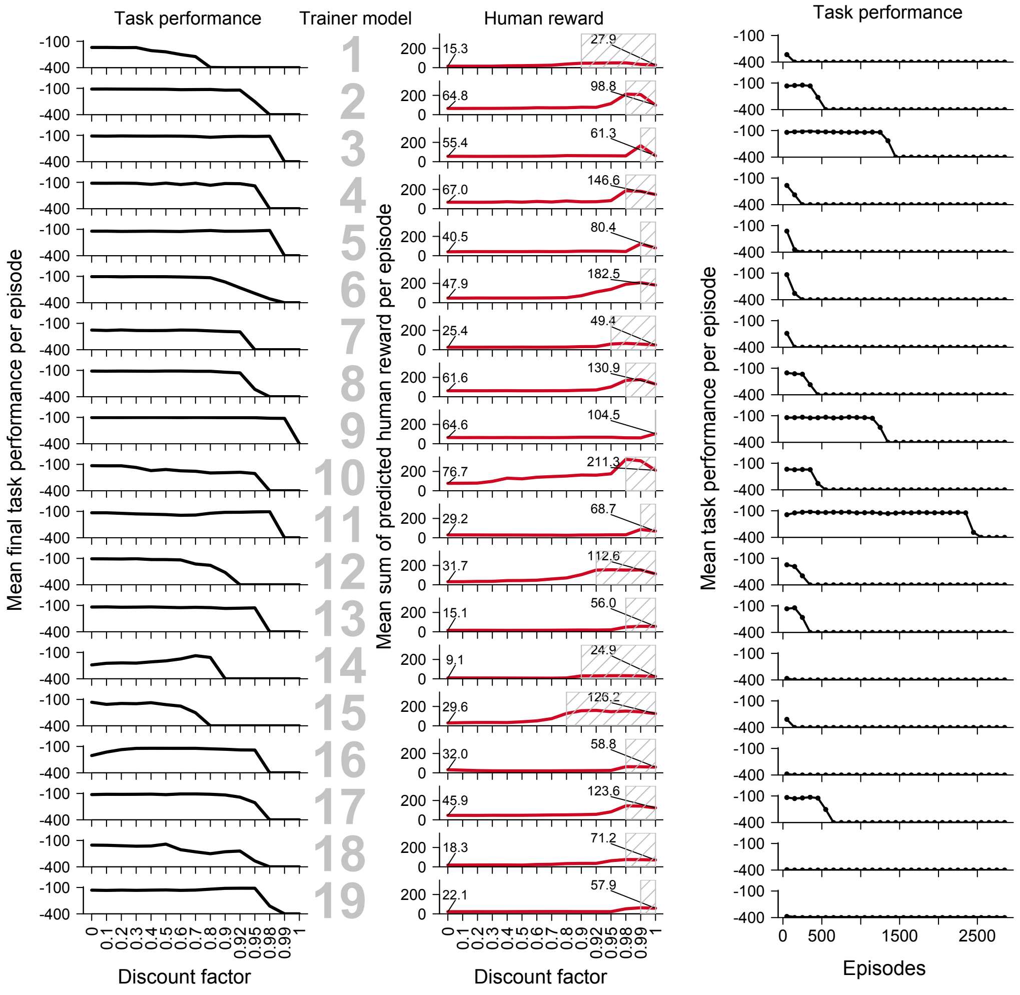
7.2. Empirical analysis: discounting in episodic tasks

In this section, we present two empirical analyses of the impact of different discount factors when learning goal-based, episodic tasks from human reward. Recall that, as discussed in Section 2, maximizing the discounted sum of human reward is not equivalent to maximizing task performance. In fact, it is precisely the relationship between these two types of maximizations that we are investigating.

For both tasks used below, the conventional MDP specifications (i.e., with hard-coded reward functions) have $\gamma = 1$. Thus, at $\gamma = 1$ $\hat{R}_H$ is being used as if it were interchangeable with a conventional MDP reward function.

7.2.1. Off-discounting experiment (episodic)

We first describe the results of the episodic-task version of the experiment specified in Section 6.1, using 19 fixed $\hat{R}_H$s that were pre-trained during a



(a) (b)

Figure 7. (a) Non-aggregate version of the off-discounting results in Figure 6, showing learned task performance over 500 episodes after 3500 episodes of learning (left) and mean total reward per episode over these final 500 episodes (right) for each of the 19 trainer models. Gray shading in the "Human reward" column indicates the inclusive range of γ values at which the agent's task performance is minimal (-400 per episode). (b) Learning curves at $\gamma = 1$, showing mean task performance over 100 episode intervals for each trainer model.

previous user study (Knox and Stone, 2009). Recall that during training for this previous study, agents acted to maximize reward under $\gamma = 0$ discounting; we address the consequences of this experiment's off-discounting character at the end of this subsection. Mean task performance and mean total reward per episode for each tested discount factor across all 19 $\hat{R}_H$ models are displayed in Figure 6. Additionally, Figure 7 displays the same data for each $\hat{R}_H$ model separately to show the consistency of qualitative results between various human models and to permit examination of the relationship of reward accrual and performance for individual reward models over different discount factors. We consider final performance to be over the last 500 episodes of learning (of the total 4000 episodes described in Section 6.1).

Importantly, at final performance all 19 trainer models led to the worst possible task performance with $\gamma = 1$. With $\gamma = 0.99$, 18 models led to minimal task performance. We visually examined agents learning at $\gamma = 1$ from five of the trainer models; each agent exhibited a circuitous behavior.

Additionally, the mean sum of predicted human reward per episode increases as performance decreases, as can be seen in the plots of the final task performance with each trainer model (Figure 7). For all 19 trainer models, the mean human reward accrued per episode is higher at discount factor of 1 than 0. Further, for almost every trainer, at every γ value that leads to worst-possible task performance (i.e., values shaded gray in the "Human reward" column of Figure 7), the corresponding mean total reward per episode is higher than at all γ values that lead to better-than-minimal performance. The three exceptions (trainer models 2, 3, and 6) break this general observation by small margins, 15% or less. A more subjective, visual analysis of Figure 7 reveals that, for $\gamma < 1$, the change in performance between two contiguously plotted discount factors is roughly proportional to the change in mean reward accrued. This subjectively assessed pattern suggests that increased reward accrual from a change in the discount factor is tightly associated with a decrease in performance.

Three general patterns emerge. We have noted the first: task performance decreases as the discount factor increases. Second, agent algorithms also accrue higher amounts of predicted human reward as the discount factor increases. Third, as the discount factor varies within $\gamma \in [0, 1)$, decreases in performance are closely tied to increases in reward accrued. From these three patterns, we observe that the best task performance is not aligned with behavior that accrues the most predicted human reward. Additionally, these results provide initial evidence of the presence of positive circuits: compared with agents optimizing for low γ values, agents optimizing for high γ values learn policies that accrue more human reward but do not reach the goal, and a sampling of the learned policies for $\gamma = 0$ showed consistently circuitous behavior.

Figure 7b shows learning curves at 100-episode intervals for a single run at $\gamma = 1$ for each trainer model. Good initial performance lasts for a varying amount of time but then degrades to worst-possible performance quickly. In the plots, this degradation occurs during the intervals with intermediate performance.

In general, the choice of RL algorithm will impact performance, so one might ask whether the algorithm used here is actually learning an MDP-optimal policy for its corresponding human reward model and discount factor. At $\gamma = 0$ and $\gamma = 1$, the answer appears to be "yes." At $\gamma = 0$, the agent optimizes return at tree search depths greater than 0. When the search depth is zero, it uses the learned value for Q(s,a), which is roughly equivalent to $\hat{R}_H(s, a)$ after many learning samples at or near (s,a). At $\gamma = 1$, if the RL algorithm learns an infinitely repeatable sequence of actions with positive net reward, then the disastrous policy that repeats that sequence is necessarily within the set of MDP-optimal policies. As mentioned above, we visually checked the behavior of five human models' corresponding algorithms while they exhibited the worst possible performance, and each agent repeatedly visited roughly the same circuit of states until the episode limit was reached. During this circuitous behavior, the maximum Q values at all observed states were positive. Therefore, the results for $\gamma = 0$ and $\gamma = 1$ can be considered correct — independent of the RL algorithm used — with confidence. However, for $0 < \gamma < 1$, another RL algorithm might learn a policy with a higher mean total reward per episode than in these results.

There is one important caveat to the conclusions we draw from this off-discounting analysis. Training occurred with TAMER algorithms, effectively at $\gamma = 0$. We strongly suspect that trainers adjust to the algorithm with which they interact; if the agent is maximizing immediate reward, we expect that its trainer will likely give more reward for immediately previous behavior. Only a *on-discounting* analysis—as we perform in the following experiment—will address whether this caveat of trainer adjustment has affected our conclusions.

We note that off-discounting analysis has one considerable advantage that on-discounting analysis lacks, which makes these two analyses complementary. Specifically, the off-discounting analysis permits us to examine what occurs with a fixed $\hat{R}_H$ at many different γ values. In particular, the tight relationship between increases in reward accrued and decreases in performance as the discount factor changes—as a close comparison of the left and right columns of Figure 7(a) reveals—is more evident than when comparing reward functions trained separately at different discount factors (as in the upcoming Figure 9). Our observation of this relationship from these off-discounting results precipitated the framing of this article's research and the experiments described hereafter. In addition to this strength of off-discounting analysis, the off-discounting results add supporting evidence for the conclusions within this section, though the on-discounting results provide stronger evidence.

7.2.2. On-discounting experiment (episodic)

In the episodic-task analysis described here in Section 7.2.2—as in the off-discounting analysis in the previous Section 7.2.1—the human reward model $\hat{R}_H$ is learned by TAMER and provides predictions that are interpreted as reward by an RL algorithm. But unlike the previous analysis, $\hat{R}_H$ is learned while performing reinforcement learning, and the RL algorithm—not TAMER—selects actions while learning $\hat{R}_H$. Thus this experiment is on-discounting, and the human trainer will be adapting to the same algorithm, with the same γ, that is being tested. The agent, task, and experiment conform to the baseline specified in Section 6.2.

Subjects from Amazon Mechanical Turk were randomly assigned to train an algorithm using one of five different discount factors: 0, 0.7, 0.9, 0.99, and 1. 10 subjects were recruited per condition; for these five conditions, the respective number of subjects after filtering (Section 6.2.3) was 10, 8, 10, 7, and 7. As explained in Section 6.2, the actual training session stopped after the agent reached the goal 5 times or after 300 steps, whichever came first.

Results Figure 8 shows the success rate of trained agents by condition, dividing them among those that never reach the goal, reach the goal 1–4 times, and reach the goal the maximum 5 times. These results exhibit a clear pattern of task performance worsening as the discount factor increases. This pattern is supported by significance testing. We conducted Fisher's exact tests of whether the agent reached the goal 5 times, by condition: between $\gamma = 0$ and $\gamma = 1$, $p = 0.0006$ (extremely significant); between $\gamma = 0$ and $\gamma = 0.9$, $p = 0.0325$ (significant); and between $\gamma = 0$ and $\gamma = 0.7$, $p = 0.4444$ (not significant).

To evaluate reward positivity, the basis for the positive circuits problem (Section 7.1), we examine the ratio of cumulative positive reward to cumulative negative reward given by trainers in each condition, shown in Figure 9. Success appears highly related to this ratio; in Figure 9, we are able to draw a dividing line at each condition between all agents that never reach the goal and all other, more successful agents. Additionally, the ratio of this division between success and failure monotonically decreases as the discount factor increases, which supports our conjecture that the positivity of human reward becomes more problematic as the discount factor increases (Section 7.1). Without recognition of the positive circuits problem (Section 7.1), this pattern of lower-performing agents getting *more* reward would be quite counter-intuitive. Further, negative Spearman's correlations between discount factor and these ratios are extremely significant both for all trainers and for only trainers whose agents reached the goal once or more ($p <= 0.0005$), but the correlation when considering only goal-reaching trainers is stronger (correlation coefficient $\rho = -0.7594$, compared to $\rho = -0.543$ for all trainers). We conjecture that γ affects ratios by both filtering out trainers that give too much positive reward in conditions of higher γs and by pressuring trainers to adjust their ratio in response to the agent. In surveys given after training, at least one trainer, from the $\gamma = 0.9$ group, spoke of his attempts to adapt to the agent: "When [the reward] key is stroked there is not much response in the robot. Only [the punishment] key stroke worked."

Reward is predominately positive (a ratio greater than 1) for 66.7% of trainers in this experiment, which supports the conjecture that human reward generally has a positive bias. At $\gamma = 0$, all trainers are predominately positive, fitting results from TAMER experiments (Section 7.1.1). At $\gamma = 0.7$, the closest condition to that of the $\gamma = 0.75$ used in Thomaz and Breazeal, 5 of 8 trainers are predominately positive, 2 are predominately negative, and 1 trainer's ratio is exactly 1, roughly fitting Thomaz and Breazeal's observations (Thomaz and Breazeal, 2008). However, for $\gamma \geq 0.99$ most trainers are predominately negative, contrasting with what has been reported in past work, none of which

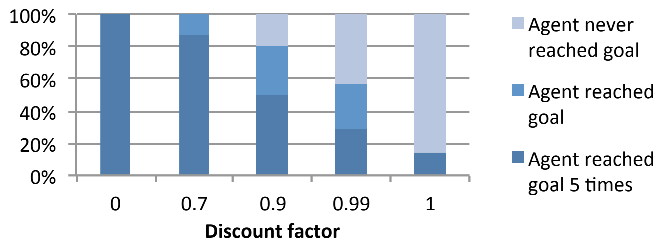


Figure 8. Success rates for the episodic grid-world experiment by discount factor.

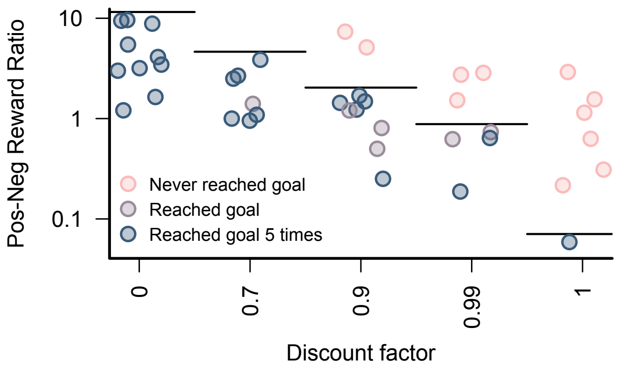


Figure 9. Ratio of cumulative positive reward to cumulative negative reward given by each trainer, divided by discount factor condition and task performance. Jitter has been added along the x-axis for readability. For each condition, a horizontal line was placed above the mark for the highest ratio at which a subject trained the agent to reach the goal at least once.

includes such non-myopic learning.

Since this task has discrete states, we can precisely test for positive circuits by narrowing the definition of a behavioral circuit from Section 7.1.2 to be infinitely repeatable trajectories that start and end on the *same* state. Thus, any behavioral circuit with net-positive reward is a positive circuit. After training, *there was at least one positive circuit in 35 of the 42 MDPs created from trainers' reward models.* In other words, 83.3% of the trained agents would exhibit the positive circuits problem if acting to maximize return with $\gamma = 1$. Half of the predominately negative trainers created positive circuits. Those without positive circuits all had positive-to-negative reward ratios below 0.63 and generally were from higher γ conditions: one from 0.7 and two each from 0.9, 0.99, and 1.

One strategy that solves the problem of positive cycles is to make all instances of reward negative. We reject this strategy, since it would require a priori knowledge that all absorbing states are goals and effectively remove the trainer's power to decide which absorbing states are desirable. We empirically focus on goal-based tasks in this article (though Section 10 includes a task that also contains a failure state), but we only consider strategies for learning from human reward that free the trainer to designate any absorbing state as a goal or a failure. Another possible strategy is to make the reward from absorbing state positive instead of zero, such that the agent receives this value for infinite steps in the future (though only in theory, since the calculation of return for constant reward values is simple and can be done without actually experiencing that reward). This strategy is arguably more biologically plausible than giving a reward of zero, since finishing an activity typically frees a biological agent to accrue further reward on some other activity. We do not directly investigate the effects of giving positive reward from absorbing state, but the strategy in the following section—making the task appear continuing—is equivalent to setting the absorbing-state reward such that its resulting return is equal to the agent's discounted expectation of return from the distribution of starting states, assuming the value function is accurate for the policy it assesses.

Taken together, the two experiments in this section provide evidence that *agents should act myopically (i.e., use small discount factors) when learning goal-based, episodic tasks from human reward.* A refinement of this recommendation is given in the Conclusion.

8. Continuing-task experiment

In the on-discounting experiment described in this section, we investigate the impact of the discount rate when a task is continuing. For episodic tasks, we argued in Section 7.1 that a positive reward bias among human trainers combined with high discount factors can lead to infinite behavioral circuits—created by what we call "positive circuits"—and consequently minimal task performance. In Section 7.2, we found that myopic discounting (low γs) in episodic settings avoids this problem. However, positive circuits may only cause severe problems in episodic tasks, since an agent reaching an absorbing goal state is effectively penalized; it is exiting a world rich with positive reward. We examine another strategy to remove this penalty: formulate the task as continuing, making the goal a gateway to more opportunity to accrue reward. Unlike for episodic MDPs, the optimal policy of a continuing MDP is unaffected by adding a constant value to the reward from all non-absorbing state; thus, the positivity of human reward may not present the same problem for MDP-optimal policies in continuing tasks.

8.1. Experiment and analysis of results

This experiment uses the baseline agent, experimental design, and continuing task in a grid-world (Section 6.2), repeating our experiment from Section 7.2.2 almost exactly, only changing the task to be continuing as described in Section 6.2.1. For consistency with the previous experiment, we analyze data from the first 5 episodes that occur before the 301st time step and also retain the $\gamma = 1$ condition, even though such discounting is generally inappropriate for

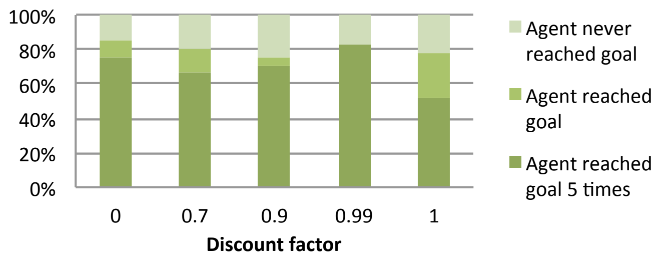


Figure 10. Success rates for the continuing-task experiment by discount factor.

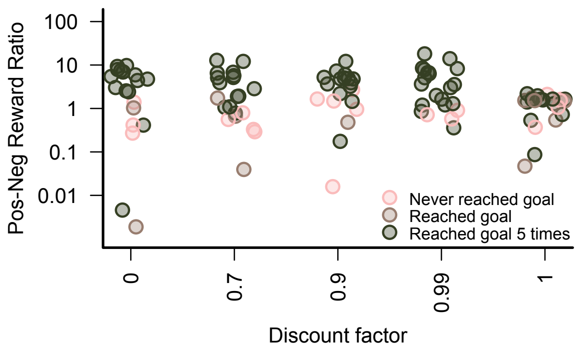
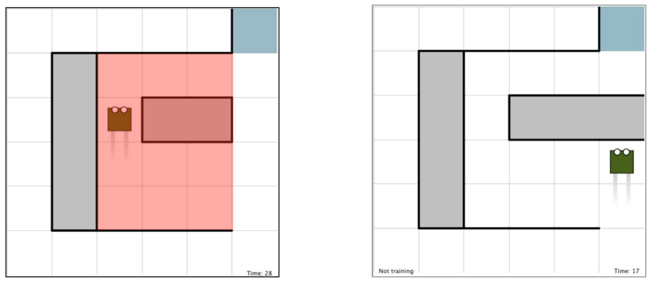


Figure 11. For the continuing-task experiment, ratio of cumulative positive reward to cumulative negative reward given by each trainer (with x-axis jitter).

continuing tasks. 25 subjects were run per condition, and one subject in the $\gamma = 0.9$ condition was replaced by another subject for not following instructions (a practice followed only in this experiment). After filtering (Section 6.2.3), for γs of 0, 0.7, 0.9, 0.99, and 1, there were respectively 20, 21, 20, 23, and 23 subjects. Figures 10 and 11 show results (presented analogously to Figures 8 and 9).

In comparison to our prior episodic-task experiment in the grid world (Section 7.2.2), the results in the continuing-task version are markedly different. As shown in Figure 10, the task success rate at $\gamma = 1$ is lower than at other conditions, which we expect given that this discounting is generally avoided for continuing tasks to make rewards in the finite future meaningful. The other discount factors create similar task performance, with $\gamma = 0.99$ achieving the highest mean rate of full success. In Fisher's exact tests of whether the agent reached the goal 5 times, there is a marginally significant difference between $\gamma = 0.99$ and $\gamma = 1$ conditions ($p = 0.0574$); no other pairwise comparisons between conditions are significantly different.[4]

[4]Note that episodicity cannot affect a $\gamma = 0$ agent, making this condition identical to the $\gamma = 0$ condition of our prior experiment. The difference in success rate at $\gamma = 0$ in the two



(a) (b)

Figure 12. Illustrations of the two tests of the benefits of non-myopic learning, testing agent performance after training. (a) Starting from (highlighted) states off the optimal path. (b) Blocking the optimal path.

Patterns exhibited by the ratios of cumulative positive reward to cumulative negative reward among trainers (as shown in Figure 11) also differ from the episodic experiment. Specifically, there is no significant correlation between the positive-to-negative reward ratios of fully successful trainers and discount factor when $\gamma = 1$ is excluded (Spearman's coefficient $\rho = -0.0564$, $p = 0.6628$), though the correlation is significant with $\gamma = 1$ included ($\rho = -0.3233$, $p = 0.0050$). Further, the relationship between reward positivity and task performance is closer to the intuitive expectation that high-performance agents receive more positively-biased reward: Spearman's correlations between ratios and success categories (no, partial, and full success) are significantly correlated in the three $\gamma \leq 0.9$ conditions (Spearman's coefficient $\rho > 0.595$, $p < 0.006$ for all $\gamma \leq 0.9$) and $\rho > 0$ as well for the two other conditions, though the correlation is insignificant.

*For this task and this approximately MDP-optimal RL algorithm (*VI-TAMER*), converting the task to continuing does indeed appear to remove the adverse impact of reward positivity at high discount factors, overcoming the positive circuits problem.* However, based only on the roughly equivalent task performance for all $\gamma < 1$ conditions, the choice of which discounting to use is unclear. In the next subsection, we investigate whether higher-level task information was communicated by the trainer, making learning more robust to changes to the environment or more general at certain γ values.

8.2. Benefits of non-myopic learning

At non-myopic discount rates (i.e., γs near 1), reward can communicate a desired policy, the goals of the task, or a mix of the two, as discussed in Section 3.2. To illustrate, we note that an example reward function for $\gamma < 1$ that provides only goal-related information is to give +1 reward at the goal

grid world experiments is likely because of either randomness—their difference is insignificant by a Fisher's exact test of reaching the goal 5 times ($p = 0.2890$)—or this experiment, run at a different time, sampled from a lower-performing population.

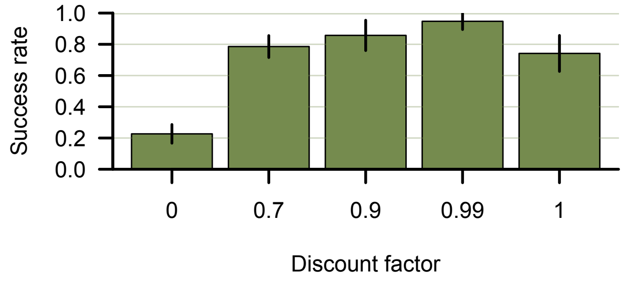


Figure 13. Mean rate of successfully trained agents reaching the goal in less than 100 time steps *from the 10 states off of the optimal path*. **Standard error bars are calculated using a single agent's success rate as one sample.**

and 0 otherwise. Alternatively, for a continuing task with $\gamma < 1$, a reward function that gives +1 reward for transitions that follow a desired policy and -1 otherwise only communicates policy-related information; no higher-level task information is communicated. Note that this latter reward function describes a plausible heuristic for human training, one that is supported by participants' descriptions of their own training strategies. Therefore, we cannot assume that human feedback contains higher-level task information.

We now investigate whether the trained agents do learn more than a policy, using the full training data from this experiment (up to 10 episodes or 450 time steps). Since $\gamma = 1$ is generally inappropriate for continuing tasks, we expect $\gamma = 0.99$ to yield the best results. We restrict our analysis to those agents successfully trained to get to the goal 5 times in less than 300 steps. Thus, we effectively ask, *given that an agent learns a good (and usually optimal) policy from human reward, what else does the agent learn?*

We first test the learned policy from 10 states that are not along the optimal path from the start state, which are highlighted in Figure 12a. These states may have never been experienced by the agent during training, in which case $\hat{R}_H$ is built without any samples from the state. Simple policy generalization from nearby, optimal-path states would help in only some of these 10 states, so the ability of the agent to get to the goal reflects whether the agent has learned some information about the task goals. Agents that had learned their policies at higher γs were more often able to get to the goal in less than 100 time steps (Figure 13). 18 of 19 successfully trained agents in the $\gamma = 0.99$ condition reached the goal from every tested state. We note though that different discount factors might lead to different levels of experience in these tested states, providing a confounding factor.

In the second test, an obstacle is placed in the state two cells below the goal (Figure 12b), blocking the optimal path, and we then determine whether the agent can still reach the goal in less than 100 time steps. Thus, we test the effects of changing the task-optimal policy but keeping constant the task goal: get to the goal state as quickly as possible. For the two state-action pairs that previously transitioned into the newly blocked state, the agent's reward function is modified to output 0 to reflect the agent's lack of knowledge about how the trainer would reward these transitions. Note that if there was no generalization while learning $\hat{R}_H$, this modification would be identical to learning $\hat{R}_H$ with next state incorporated as a third input (i.e., $\hat{H} : S \times A \times S \to \mathbb{R}$, instead of $\hat{H} : S \times A \to \mathbb{R}$), still initializing all outputs to 0. In the $\gamma = 0.99$ condition, 9 of 19 successfully trained agents reached the goal. One agent from each of the $\gamma = 0.9$ and $\gamma = 1$ conditions also reached the goal (of 14 and 12 total, respectively); no agents with $\gamma < 0.9$ did.

These analyses support the conjecture that *agents taught with higher discount factors learn about the task goals themselves, making the agents generally more robust to changes in the environment and more able to act appropriately in previously unexperienced states.* That agents may learn task goals raises the tantalizing prospect that, under the right circumstances, an agent receiving reward from a human trainer could learn a policy that is far superior to that envisioned by the trainer. These benefits of non-myopic learning underscore the importance of creating algorithms for complex, real-world tasks that can learn non-myopically. As the next two sections confirm, achieving this goal is non-trivial.

9. Local-bias experiments

In considerably more complex domains than the 30-state grid-world task used in our continuing-task experiment, agents will likely be unable to perform value iteration with iterating sweeps over the entire state space; even ignoring the possibility of continuous states or actions, some tasks simply contain too many states to quickly and repeatedly perform Bellman updates on all states. In this section, we describe two experiments in which agents generally *do not* act approximately MDP-optimally.

9.1. On-discounting local bias experiment

In anticipation of scaling the high-γ, continuing-task approach found successful in the previous experiment, we implemented a version of value iteration that learns asynchronously, which we call aVI-TAMER. Pseudocode for aVI-TAMER is given in Algorithm 3 in Appendix B. Instead of updating each state once and then repeating as in VI-TAMER,[5] aVI-TAMER updates state-action pairs through the Monte Carlo tree search strategy Upper Confidence Trees (UCT). UCT-based search has been successful in tasks with especially large state spaces (Kocsis and Szepesvári, 2006), originally in games like Go (Gelly and Silver, 2008) but also in more general reinforcement learning tasks (Hester and Stone, 2011).

aVI-TAMER is mostly identical to VI-TAMER: the human reward model $\hat{R}_H$ is learned as before, using TAMER; a tabular action-value function is maintained;

[5]VI-TAMER is not synchronous in the strictest sense—where the entire sweep across state updates with the same value function—but we find this term "asynchronous" useful for distinguishing these two approaches.

and the agent acts greedily with respect to that function. Unlike VI-TAMER, aVI-TAMER's "planning" consists of repeatedly considering different possible 40-step trajectories from its current state. Transitions from these trajectories provide updates for value iteration.[6] Planning trajectories are chosen by UCT (Kocsis and Szepesvári, 2006), where the search tree is reset at the start of each time step to respect the corresponding change to the reward function $\hat{R}_H$ at that step, which generally makes past search results inaccurate. The confidence value for UCT is 1.

For the aVI-TAMER algorithm, the number of updates to each state's value differs considerably among states; in contrast, between sweep iterations in our value iteration implementation, all states have been updated an equal number of times. Instead of a balanced distribution of updates, state transitions that can quickly be reached from the current state receive many more Bellman updates than transitions from less "local" states. For complex tasks in general, this bias towards local updating appears desirable, since an effective learning agent will likely visit regions of the state space that are worth understanding more often than areas that can be ignored during learning. Additionally, this local updating bias occurs in a large fraction of common RL algorithms, including Sarsa(λ), Q-learning, and Monte Carlo tree search algorithms. We chose aVI-TAMER as a representative of this class of algorithms because it learns much more quickly than most other locally-biased algorithms. However, we recognize that there may be unforeseen benefits from the worse MDP-based performance of these other algorithms, similar to aVI-TAMER in the following section's failure-state experiment outperforming VI-TAMER in the episodic framing of the task. We note that there are alternative planning-based RL algorithms such as real-time dynamic programming (RTDP) Barto et al. (1995), which we do not explore here.

This experiment departs from the on-discounting baseline set of agent, grid-world task, and experiment specifications only by the inclusion of aVI-TAMER as the algorithm for two conditions. Since this investigation is focused on the effect of locally-biased updates on a high-γ algorithm, all three conditions calculate return with $\gamma = 0.99$. We are primarily interested in two conditions: VI-cont and aVI-cont, which respectively denote VI-TAMER and aVI-TAMER acting in a continuing version of the task. Note that this VI-cont condition is identical to the $\gamma = 0.99$ condition of the continuing-task experiment of the previous section; it is rerun here to account for the differing population that subjects will be drawn from. As a third condition called aVI-epis, we added aVI-TAMER in an episodic version to see what gains are retained by making the task continuing when updates are locally biased. The results of this experiment are shown in Figures 14 and 15. All results concern the full duration of training unless otherwise specified. 26 subjects were run per condition, resulting in the following

[6]Using experienced transitions for action-value-updating value iteration is only valid for deterministic policies and transitions, as we have here. Also, note that the update mechanism is not itself Monte Carlo.

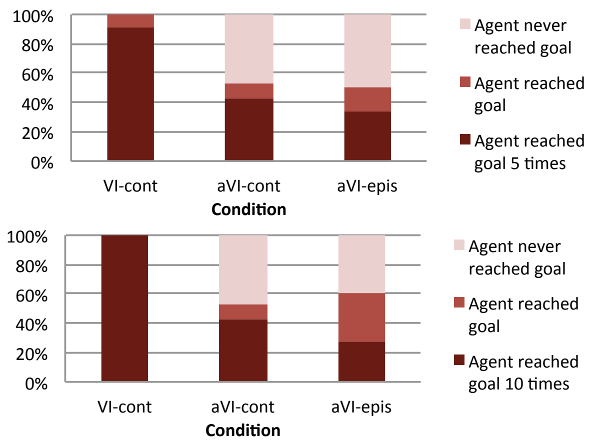


Figure 14. Success rates by condition for the local-bias experiment. Through their bias towards updating "local" states, the aVI-TAMER conditions create behavior that is farther from MDP-optimal for the current reward function than is behavior learned by VI-TAMER. The top plot shows success with the stopping points used for Figures 10 and 11 of Section 7.2.2's experimental results, the first of 5 episodes or 300 time steps. The lower plot displays success with this experiment's stopping points, the first of 10 episodes or 450 time steps.

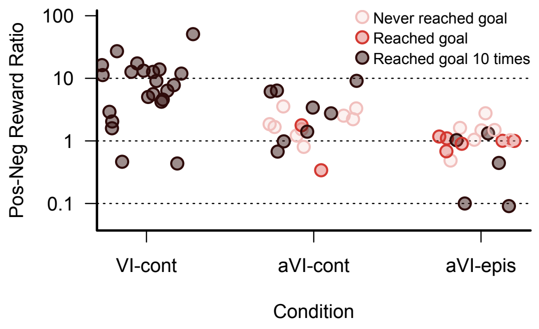


Figure 15. For the local-bias experiment, ratio of cumulative positive reward to cumulative negative reward given by each trainer (with x-axis jitter).

number of samples by condition after filtering: VI-cont, 22; aVI-cont, 19; aVI-epis, 18.

We observe that the results for the VI-cont condition are similar to that of the equivalent $\gamma = 0.99$ condition in the continuing-task experiment (shown in Figures 10 and 14). The performance is insignificantly higher in this experiment by a Fisher's exact test of whether the agent reached the goal 10 times ($p = 0.1085$). *This experiment further supports the assertion that the* VI-TAMER *algorithm successfully learns from human-generated reward at high discount fac-*

tors in the continuing grid-world task.

9.1.1. Effect of local bias in the continuing task

Comparing the two continuing conditions of this experiment—VI-cont and aVI-cont—we observe that *locally biased updates result in worse performance than* VI-TAMER*'s uniform updates* (Figure 14). This difference is highly significant by a Fisher's exact test of reaching the goal 10 times ($p = 0.0001$) and by a Mann Whitney U test of episodes finished before training was stopped ($p = 0.0016$). We also conducted a Mann Whitney U test of how many steps it took the agent to reach the goal *the first time*. This result is highly significant as well ($mean_{VI-cont} = 93.45$; $mean_{aVI-cont} = 272.11$; $median_{VI-cont} = 70$; $median_{aVI-cont} = 250$; $p < 0.0001$), indicating that the change to locally biased updates slows early learning.

VI-TAMER effectively performs 4800 Bellman updates to an action-value function per time step (40 sweeps $\times$ 30 states $\times$ 4 actions), compared to medians of 589 for the aVI-cont group and 1004 the aVI-epis group. Though aVI-TAMER's updates—dependent on the subject's computer—were less frequent, we doubt this difference is a meaningful factor in the results; a four-fold increase in aVI-TAMER's update speed would add less than one level of depth to its exhaustive search tree (which is extended by greedy rollouts to reach the full trajectory depth of 40), given that the four actions and deterministic transitions create a branching factor of four in the search tree. Likewise, a sixteen-fold increase would give aVI-TAMER more updates per time step than VI-TAMER but add less than two levels of depth to aVI-TAMER's search tree.

Other than the number of updates per step, the only difference between aVI-TAMER and VI-TAMER in the continuing conditions is which state-action values are updated. We suspect that the locally-biased character of aVI-TAMER's updates is interacting with the positivity of human reward to lower the aVI-TAMER's algorithm's performance. Specifically, local bias causes nearby states to receive more updates, and the positivity of reward—with an action-value function initialized to 0, as in all experiments of this article—makes states that are updated more often appear more desirable, consequently strengthening the local bias even further. In early learning, the aVI-TAMER agent will not learn the true values of states along its MDP-optimal path if it does not get near those states, and policies that bring the agent back to previously experienced—and thus highly valued—states will be followed. The value-function heat maps in Figure 16 support this explanation of the performance differences; states that are far from experienced states often have not been updated even once and retain their original state values of 0.

Additionally, when the agent does not visit the states near the goal, the trainer cannot give feedback for those states, preventing the agent from learning about what reward might be received along those critical states. Accurate expectations for rewards from such states might alleviate the effect of local bias by providing positive near-term reward for desirable paths. Therefore an aVI-TAMER agent ironically needs to traverse to the goal to learn the accurate reward predictions that might lead it to traverse to the goal.

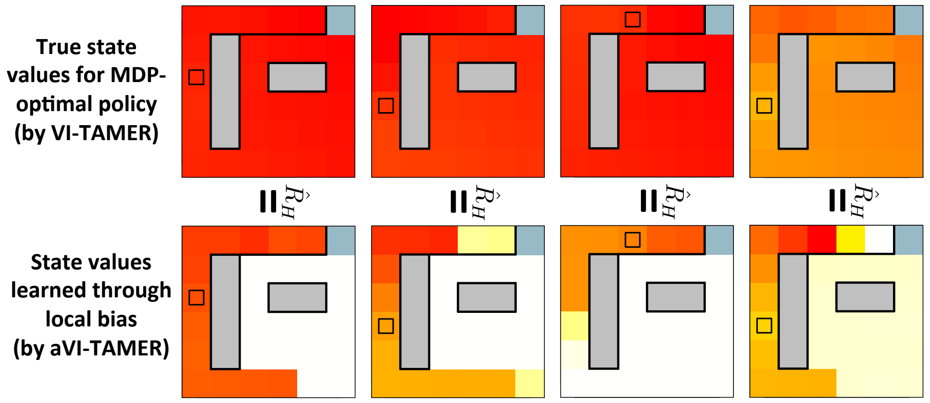


Figure 16. Heat maps showing value functions learned by VI-TAMER and aVI-TAMER, where both maps in a column were created through the same experience trajectory. Both algorithms learn from a training log from the VI-TAMER condition, containing a sequence of state-action pairs and time-stamped human reward signals. Training is stopped during the *first* step at which the reward function specifies the task-optimal path to the goal. 4 logs were chosen from the 22 of the VI-cont condition at varying time-to-first-goal values. The VI-TAMER algorithm used to create these heat maps performs 2000 update sweeps per step, increasing from the 40 per step in the experiment to approximate the MDP-optimal value function with further accuracy. The aVI-TAMER algorithm experiences the same trajectory, learning from 1000 planning trajectories per step. Each heat map shows state values after learning of $\hat{R}_H$ is stopped and the corresponding algorithm performs one further step worth of updates. The deepest red corresponds to highest value of both value functions in the column, and white squares represent the lowest values, which is 0 or less in all four columns. The location of the agent at the trajectory's stopping point is shown by an agent-sized square.

One potential solution to this problem of overly local exploration is to optimistically initialize $\hat{R}_H$ to drive exploration of the state-action space. But that approach has its own potential pitfalls. In particular, too much optimism leads to extensive exploration of the state-action space. Such a great amount of exploration during training would frustrate, exhaust, and confuse the trainer, since such exploration would include much behavior that goes against the trainer's feedback, making the agent appear unresponsive and unable to learn for a considerable period. Thorough exploration would additionally sacrifice the fast learning that is one of the chief appeals of interactive shaping. Nonetheless, some form of mild optimism—or simply realism, initializing at some estimate of average reward—may solve the problems of overly local exploration. We leave an analysis of such an approach to future work, instead focusing our remaining experiments on improving the generality of the findings we report in this and previous sections.

9.1.2. Effect of episodicity for locally biased learning

Given that the locally biased updates of aVI-TAMER worsen performance compared to the approximately MDP-optimal performance of VI-TAMER, we now ask whether the choice of episodic or continuing formulations still has an appreciable effect in the context of aVI-TAMER's locally biased updates. Comparing these two conditions by the same three statistical tests applied above to compare the VI-cont and aVI-cont conditions, none are significant (*all* $p > 0.49$) and the corresponding means, medians, and proportions for the tests are quite similar across conditions.

The ratios of positive to negative reward in Figure 15 are negatively and significantly correlated with success for aVI-epis (Spearman's coefficient $\rho = -0.50$, $p = 0.035$) and uncorrelated for aVI-cont (Spearman's coefficient $\rho = 0.19$, $p = 0.45$), following the pattern observed from the continuing-task experiment for VI-TAMER. We also look at how often an agent *relapses* into sub-optimal task performance after an episode that is completed in minimal time. As mentioned previously, uniformly raising the value of all rewards by a constant value does not affect the MDP-optimal policy for a continuing task but often will for an episodic task (assuming the value is not raised at an absorbing state). Consequently, we suspect that trainers will generally give more positive reward after their agent acts optimally (though their reward is not necessarily uniformly higher), which may cause more problems for the episodic aVI-TAMER condition. In the aVI-cont condition, of the 11 agents that finish an episode in minimal time during the first five episodes, 36.6% subsequently relapse into non-optimal behavior before reaching the 10-episode or 450-time-step endpoint. In the aVI-epis condition, 77.7% of the 9 such agents do. This difference is marginally significant in a Fisher's exact test ($p = 0.0923$).

In overall performance, aVI-TAMER-guided updates do not clearly benefit from making the task continuing. However, we do observe that success in the continuing version still appears unrelated to reward positivity. Additionally, the relapse rate is lower in the continuing task. We suspect that these strengths of the continuing version are balanced against one advantage of the episodic version, that there is a simple but counterintuitive method for teaching the agent to act task-optimal: giving only negative reward. This hypothesis informs the experiment in Section 10.

9.2. Off-discounting local bias experiment (continuing)

We now describe a second experiment that incorporates local bias such that the agents are generally not acting approximately MDP-optimally at higher γ values. This experiment is the continuing-task version of the mountain car experiment from Section 7.2.1, using the same limited-lookahead Sarsa(λ) algorithm. Here, however, the task is made continuing by replacing any transition to an absorbing state with a transition to a start state. Like aVI-TAMER, this algorithm (and Sarsa(λ) generally) is locally biased in its updates. With pretrained $\hat{R}_H$s, this experiment differs from the local-bias experiment in the grid world in that

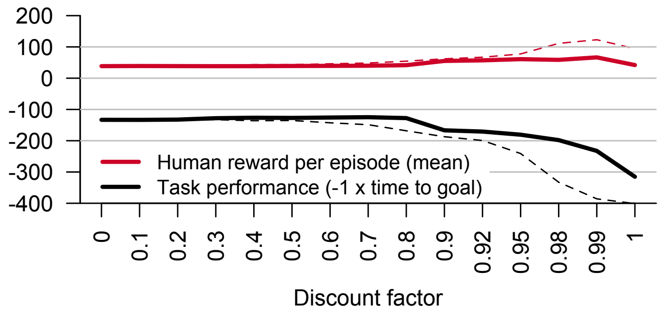


Figure 17. Aggregate results for the off-discounting model analysis with local bias from the limited-lookahead Sarsa(λ) algorithm. Results from the continuing setting are shown in bold solid lines; thin dotted lines show results from the episodic setting (and were shown previously in Figure 6). The plot shows mean task performance and mean sum of $\hat{R}_H$-generated reward per episode for mountain car, over the final 500 episodes of 4000 episodes of learning.

1. as mentioned in Section 7.2.1, the $\hat{R}_H$s likely reflect trainer adaptation to the $\gamma = 0$ discount factor under which these human reward models were trained;
2. the samples that are used to learn the $\hat{R}_H$s come from multiple full trajectories to the goal (though not always efficient trajectories), unlike in the grid-world experiment where the agent must be trained on-discounting to reach the goal before such full trajectories are available for modeling the reward function; and
3. the agent explores through an action selection policy similar to ϵ-greedy.

Experimental results for both the continuing and episodic versions of the task are shown in Figure 17. We observe that performance is higher in the continuing task than in the episodic task at all $\gamma \geq 0.3$, by a range of 4.7 time steps per episode at $\gamma = 0.3$ to 153.3 time steps per episode at $\gamma = 0.99$. By Wilcoxon signed-rank tests—non-parametric tests for paired data—at each γ value, this improvement in performance is significant by $p \leq 0.05$ at all $\gamma \geq 0.3$. Additionally, we observe that the highest mean performance across all γs in both episodic and continuing tasks is at $\gamma = 0.7$, outperforming the myopic $\gamma = 0$ agent by 8.4 time steps per episode; however, this difference—calculated with the continuing-task $\gamma = 0$ performance, which is slightly higher than the episodic version—is not significant by a Wilcoxon signed-rank test ($p = 0.246$). This improvement over full myopia occurs despite the $\hat{R}_H$s having been trained at $\gamma = 0$ (i.e., the results are on-discounting or $\gamma = 0$ but off-discounting otherwise). In this second local-bias experiment, which differs from the first in the key ways listed above, we see *a performance advantage conferred by making the task continuing, even with local bias.* However, performance in the continuing task is still considerably lower at non-myopic discounting than at more myopic values; performance drops monotonically for $\gamma \geq 0.7$, with especially poor performance at $\gamma > 0.9$. Thus a non-myopic agent that learns successfully from human reward *with local bias* is achieved in neither task.

10. Failure-state experiment

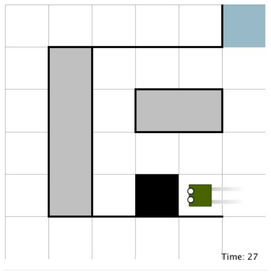


Figure 18. The grid-world task with a failure state added.

In this experiment, the task is further manipulated to have a failure state that is closer to the start state than the goal state, as shown in Figure 18. In the episodic version of this task, both the failure and goal states are absorbing states; in the continuing version, transitions to either are experienced by the agent as transitions to the start state. When this modified task is episodic, giving only negative reward is likely to create MDP-optimal policies that fail by repeatedly looping through the failure state. In designing this task, we hope to represent the large class of tasks for which failure can be achieved more quickly than the goal (e.g., driving to a destination without crashing). In this task, we predicted that the continuing version would outperform the episodic version by a greater margin. This experiment tests both this prediction and the generality of the results from the other experiments, which thus far have been exclusively demonstrated in a goal-only task.

We use the same algorithms and discount rate ($\gamma = 0.99$) as in the previous experiment, adding as a fourth condition the VI-TAMER algorithm with the episodic version of the task, making this a full 2x2 experiment. We refer to the new condition as VI-epis. Except for an additional instruction to make the agent avoid the failure state—a "black pit that takes [the agent] back to the start"—the experiment was conducted identically as the baseline experiment until the agent reaches the goal a 10th time. If a trainer reaches that point, instead of stopping the experiment we allow the user to continue training until all 450 time steps have passed. We made this adjustment to add resolution among the most successful trainers (e.g., between trainers who would get the agent to the goal 11 times or 18 times). 20 subjects were run per condition; after filtering the data (as explained in Section 6.2.3), the number of subjects were as follows: 15 for VI-cont, 16 for VI-epis, 14 for aVI-cont, and 14 for aVI-epis.

Results from this experiment are described by Figures 19 and 20 and the table of statistical test results in Figure 21. Generally speaking, we observe the same performance patterns as in the experiments with the goal-only task, though these patterns are less pronounced. VI-cont performs best in all comparisons, significantly so in 5 of 6. For both VI-TAMER and aVI-TAMER algorithms, performance is better for the continuing version of the task, except for aVI-TAMER's mean rank of time to goal from the Mann Whitney U test of the time steps before the agent first reached the goal, where aVI-epis is insignificantly better than aVI-cont. Interestingly, the new comparison between VI-epis and aVI-epis reveals that the agent performs better in the algorithm with locally biased updates, aVI-epis. We suspect that this result arises because the sub-optimality

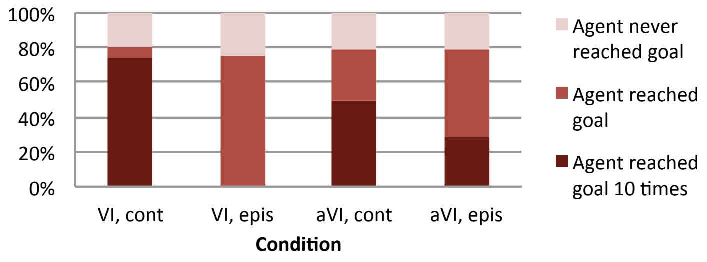


Figure 19. Success rates by condition for the failure-state experiment, which investigates the effects of locally-biased updates (the avi-tamer conditions) and episodicity when there is both a goal state and a failure state.

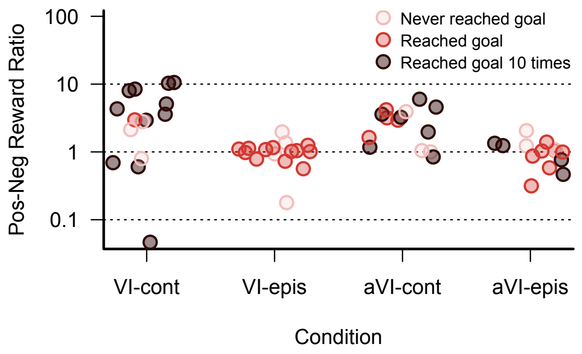


Figure 20. For the same experiment, the ratio of cumulative positive reward to cumulative negative reward given by each trainer (with x-axis jitter).

of aVI-TAMER makes it less likely to find existing positive circuits, which could prevent the agent with a $\gamma = 0.99$ reward-based objective from going to the goal. In other words, failing to achieve an undesirable objective can be better than achieving it. This result raises the surprising prospect that the combination of an agent objective that poorly aligns MDP-optimal and task-optimal behavior combined with an agent that poorly maximizes that objective might produce better results than can be achieved by any considerably sub-optimal agent attempting to maximize a perfectly aligned objective.

We also examine reward positivity (Figure 20). For both VI-TAMER and aVI-TAMER, reward was more positive in the continuing version of the task, which fits observations from the two experiments on the goal-only task. In VI-epis, there is a marginally significant negative correlation between success and reward positivity by a Spearman's correlation test ($\rho = -0.4266$, $p = 0.0994$); this correlation is insignificant for the other 3 conditions. Overall, we see that episodicity has a smaller effect on reward positivity in this failure-state task than in the goal-only task. We suspect that this observation is connected to a previously described effect of adding the failure state: that the simple strategy of always giving negative reward, without regard for the state or action, no

	VI-epis			**aVI-cont**			**aVI-epis**		
Test	Fisher Success	MWU Eps. Finished	MWU Time to Goal	Fisher Success	MWU Eps. Finished	MWU Time to Goal	Fisher Success	MWU Eps. Finished	MWU Time to Goal
VI-cont	**p<0.0001, VI-cont**	**p=0.0022, VI-cont**	**p=0.0128, VI-cont**	p=0.2635, VI-cont	**p=0.0574, VI-cont**	**p=0.0257, VI-cont**			
VI-epis							**p=0.0365, aVI-epis**	**p=0.0615, aVI-epis**	p=0.5222, aVI-epis
aVI-cont							p=0.4401, aVI-cont	p=0.5823, aVI-cont	p=0.4593, aVI-epis

Figure 21. Statistical tests for comparisons of interest. Test shorthand is as follows: "Fisher Success" is a Fisher's exact test of whether the agent reached the goal 10 times or more, "MWU Eps. Finished" is a Mann Whitney U test of the number of episodes completed before training was stopped at 450 time steps, and "MWU Time to Goal" is a Mann Whitney U test of how many steps occurred before the agent reached the goal for *the first time*. Each cell contains the p-value for the corresponding test as well as the name of the condition that performed better on the metric, which could be highest proportion of success, highest mean ranking of episodes finished, or lowest mean ranking of time to first goal. (The Mann-Whitney U test converts samples to rankings.) Cells with p-values below 0.1 are emboldened.

longer creates MDP-optimal behavior that is also task-optimal.

From analyzing these results, we believe that adding the failure state affected the ease of training in both positive and negative ways. As an alternate absorbing state to the goal, the failure state generally forces trainers to give more discriminating reward (e.g., the arbitrarily all-negative strategy for the episodic version becomes unsuccessful). In comparison to the goal-only task, on the other hand, the aVI-TAMER algorithm performed better overall in both the continuing and episodic versions of the failure-task; this increase might be due in part to the failure state being used as an intermediate "goal" that the learner makes updates for, goes to, and then gets experience and reward for those states near it, which then help the agent go to the real goal. Because of these multiple factors that likely affect performance (as well as randomness and different subject populations at different experiment times), we hesitate to draw strong conclusions from comparisons *across* experiments. However, we can say that this experiment does not reveal an increased performance difference between the aVI-epis and aVI-cont conditions, as we had predicted.

Nonetheless, the results from this experiment give additional empirical support for the generality of the patterns that have been identified previously in this article, showing that these patterns appear when the task contains both desirable and undesirable absorbing state. Most important among these patterns are that (1) performance is better for a continuing formulation of the task—especially when the agent acts approximately MDP-optimally as with the

VI-TAMER algorithm—and that (2) the choice of the best algorithm for complex tasks, where MDP-optimal behavior is generally intractable, is a challenging direction for future work. In such future work, we suspect that learning tabular value functions for large, discrete state spaces will provide a more easily analyzed platform than would continuous state spaces or learning value functions via function approximation, both of which might eventually be critical for scaling learning from human feedback.

11. Conclusion

Any solution to the problem of interactive shaping—i.e., learning sequential tasks from human-generated reward—requires the definition of a reward-based objective and an agent algorithm. This article examines the relationship between reward discounting rates, episodicity, reward positivity, acting approximately MDP-optimally or not, and the ultimate objective, task performance. These relationships are examined in six experiments across two domains. The table in Figure 22 summarizes our findings.

We note the following contributions in this article:

1. identifying and ultimately giving sufficient justification for the myopic trend in past work on learning from human reward;
2. linking human reward positivity to positive circuits and empirically establishing positive circuits' prevalence;
3. adding structure to the body of past work on learning from human reward by framing and comparatively analyzing the impact on task performance of temporal discounting, whether the task is experienced as episodic or continuing, and other factors;
4. empirically finding that for approximately MDP-optimal agents, converting the otherwise episodic grid-world task to a continuing task (a) enables successful training at non-myopic discount rates, (b) removes negative correlations between reward positivity and discount factor values, and (c) removes negative correlations between reward positivity and task performance within non-myopic conditions;
5. achieving the first known instance of consistently successful training of a non-myopic agent by live, human-generated reward signals;
6. demonstrating that successfully trained agents with non-myopic objectives learn higher-level task information, making them more robust to changes in their environments and better able to act from states in which they lack experience;
7. and showing that when the agent's MDP-based performance is worsened—as it must be for complex tasks—by the common practice of locally biased learning, task performance worsens significantly in continuing tasks.

These contributions lead to the following concrete recommendation for determining a learning objective for human-generated reward signals. *Given the possibility that a human trainer will want to teach a goal-based task, an agent*

	Approximately MDP-optimal (VI-TAMER)	Locally biased learning (UCT-driven aVI-TAMER)
Episodic	• Effective with **few** trainers • Success **negatively correlated** with reward positivity in both tasks	• Effective with **some** trainers • Success **negatively correlated** with reward positivity in goal-only task
Continuing	• Effective with **almost all** trainers • Success **independent** of reward positivity in both tasks	• Effective with **some** trainers • Success **independent** of reward positivity in both tasks

Figure 22. Qualitative summary of this article's experimental conclusions on non-myopic, on-discounting learning (in the grid-world task). Note that VI-TAMER is *approximately* optimal because it does not iterate until convergence between each update to the human reward model and the subsequent action selection.

should either learn myopically or experience the task as continuing—even though the task may more naturally be considered episodic. Note that this recommendation is only conditional on the *possibility* that a human trainer would want to teach a goal-based task, a condition we expect to generally be true in rich environments that permit numerous desirable tasks. Making this recommendation more specific, we suggest myopic learning for current applications—where the unsolved problems of local bias are likely unavoidable—and non-myopic learning for research aiming to improve the power and generality of algorithms for learning from human reward.

This article also presents a contribution from a broader perspective. The work herein—an in-depth case study on adapting a machine learning problem to include human interaction—clearly indicates that *machine learning algorithms generally cannot be applied with naive hopes that the human will conform to the assumptions of the algorithms. Rather, the human element must be explicitly investigated.*

This article represents substantial progress in the effort to create effective algorithms for learning from human-generated reward. We note, however, that more analysis is required to establish the generality of our observations. Changing the reward-giving interface, the mapping of reward cues (e.g. keys) to scalar values, the instructions to trainers, our algorithmic choices, and the task to be learned—though all carefully chosen to avoid overt bias—might create qualitatively different results.

From this research, we believe that the greatest future contributions of learning from human reward will come from non-myopic objectives and will likely be in continuing tasks. However, we expect that naively designed agents with biases towards local updates—agents often well-suited for complex tasks—will ineffectively learn from human reward even in continuing tasks; the problems of reward positivity extend beyond episodic tasks. Identifying algorithms that learn non-myopically from human-generated reward in complex domains—where approximately MDP-optimal behavior will likely be impossible—remains a critical research question.

Acknowledgments

This work has taken place in the Personal Robots Group at the MIT Media Lab and in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030). We thank George Konidaris and Rich Sutton for fruitful discussions on discounting human reward and Tom Walsh for feedback on our manuscript.

References

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

Arlot, S., Celisse, A., et al., 2010. A survey of cross-validation procedures for model selection. Statistics surveys 4, 40–79.

Barto, A. G., Bradtke, S. J., Singh, S. P., 1995. Learning to act using real-time dynamic programming. Artificial Intelligence 72 (1), 81–138.

Boots, B., Gordon, G. J., 2010. Predictive state temporal difference learning. In: Advances in neural information processing systems. pp. 271–279.

Gelly, S., Silver, D., 2008. Achieving master level play in 9× 9 computer go. In: Proceedings of AAAI. pp. 1537–1540.

Grollman, D., Jenkins, O., Apr 2007. Dogged learning for robots. In: International Conference on Robotics and Automation (ICRA 2007). Rome, Italy, pp. 2483–2488.
URL `http://www.cs.brown.edu/~cjenkins/papers/dang_ICRA_2007.pdf`

Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. The Journal of Machine Learning Research 3, 1157–1182.

Hester, T., Stone, P., 2011. Learning and using models. In: Wiering, M., van Otterlo, M. (Eds.), Reinforcement Learning: State of the Art. Springer Verlag, Berlin, Germany.

Isbell, C., Kearns, M., Singh, S., Shelton, C., Stone, P., Kormann, D., 2006. Cobot in LambdaMOO: An Adaptive Social Statistics Agent. Proceedings of The 5th Annual International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

Knox, W., Stone, P., 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. Proceedings of The 9th Annual International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

Knox, W. B., August 2012. Learning from human-generated reward. Ph.D. thesis, Department of Computer Science, The University of Texas at Austin.

Knox, W. B., Glass, B. D., Love, B. C., Maddox, W. T., Stone, P., 2012. How humans teach agents: A new experimental perspective. International Journal of Social Robotics, Special Issue on Robot Learning from Demonstration 4 (4), 409–421.

Knox, W. B., Stone, P., September 2009. Interactively shaping agents via human reinforcement: The TAMER framework. In: The 5th International Conference on Knowledge Capture.

Knox, W. B., Stone, P., June 2012. Reinforcement learning with human and MDP reward. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

Kocsis, L., Szepesvári, C., 2006. Bandit based monte-carlo planning. Machine Learning: ECML 2006, 282–293.

León, A., Morales, E., Altamirano, L., Ruiz, J., 2011. Teaching a robot to perform task through imitation and on-line feedback. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 549–556.

Li, G., Hung, H., Whiteson, S., Knox, W. B., May 2013. Using informative behavior to increase engagement in the TAMER framework.

Mahadevan, S., 2009. Learning Representation and Control in Markov Decision Processes. Now Publishers Inc.

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

Nicolescu, M., Mataric, M., 2003. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In: AAMAS. ACM, pp. 241–248.

Nikolaidis, S., Shah, J., 2013. Human-robot cross-training: Computational formulation, modeling and evaluation of a human team training strategy. In: Proceedings of the IEEE/ACM International Conference on Human-Robot Interaction.

Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., Littman, M., 2008. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In: Proceedings of the 25th international conference on Machine learning. ACM, pp. 752–759.

Pilarski, P., Dawson, M., Degris, T., Fahimi, F., Carey, J., Sutton, R., 2011. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In: IEEE International Conference on Rehabilitation Robotics (ICORR). IEEE, pp. 1–7.

Randløv, J., Alstrøm, P., 1998. Learning to drive a bicycle using reinforcement learning and shaping. In: Proceedings of the Fifteenth International Conference on Machine Learning. Citeseer, pp. 463–471.

Sridharan, M., 2011. Augmented reinforcement learning for interaction with non-expert humans in agent domains. In: Proceedings of IEEE International Conference on Machine Learning Applications.

Suay, H., Chernova, S., 2011. Effect of human guidance and state space size on interactive reinforcement learning. In: 20th IEEE International Symposium on Robot and Human Interactive Communication (Ro-Man). pp. 1–6.

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

Taylor, G., Parr, R., 2009. Kernelized value function approximation for reinforcement learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM, pp. 1017–1024.

Tenorio-Gonzalez, A., Morales, E., Villaseñor-Pineda, L., 2010. Dynamic reward shaping: training a robot by voice. Advances in Artificial Intelligence–IBERAMIA, 483–492.

Thomaz, A., Breazeal, C., 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. Artificial Intelligence 172 (6-7), 716–737.

Vien, N. A., Ertel, W., 2012. Reinforcement learning combined with human feedback in continuous state and action spaces. In: Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on. IEEE, pp. 1–6.

Appendix A. Details for the on-discounting, grid-world experiments

These experiments were conducted through a web interface via Amazon.com's Mechanical Turk service. Each subject's task is called a "HIT" by Amazon, meaning a Human Intelligence Task. Subjects received the following initial instructions:

> Kermitbot needs your help!
>
> Kermit the Robot Frog wants to be more like a real frog, but he doesn't understand that frogs need water. On top of that, he's always getting lost in the corridors. Poor Robo-kermie. Your job is to teach him how to go to the local watering hole. The videos below give the main instructions, but first:
>
> - The final result of your Robo-kermie pupil will be tested later for how quickly it can get to the water. The trainers of the best 50% of robots will be paid 25% more.
> - We strongly suggest closing all other applications currently running on your computer. Otherwise the game might have problems that hurt your chance of being in the top half of trainers.
> - Don't refresh your browser! If something is terribly wrong, describe it in detail at the end and you may receive credit.
> - Reasons we would reject your HIT: (1) You either did not watch the videos fully or did not answer the questions. (2) The records indicate that you did not honestly try to train the robot. (We will not reject simply for poor robot performance, though. But we will be sad.)
> - You can only do this HIT once. We will only pay each worker for one completion.
>
> Start your task here. You'll go through 6 steps. When you are asked for your HIT ID, enter unique user ID exactly (all the characters that are red above). Do not put in your worker ID!!! Once you finish, you'll be given a number. Then answer the questions below and enter the number at the bottom.

The word "here" contains a hyperlink that opens a page with an introductory video. The script for the video is below, with on-screen text in brackets.

> [Teach a Kermitbot to find the water]
>
> For this Turk task, you'll be training a simulated robot to play a game. Together you'll form a team: you (the trainer) and the robot (the student). Your robot's performance in the game will be judged against that of other Turkers.

> Kermit the Robot Frog is very thirsty. He needs your help to find the water. So your goal is to teach the robot to find the water as fast as possible. [As fast as possible!]
>
> [Play the game yourself] Before you teach the robot, we'll have you do the task yourself by controlling it. Click on the box below, and Kermit to the water three times.

After watching the video, the subject controls the agent to get to the goal 3 times. The experiment will not progress if the subject does not complete this task. To the right of the applet containing the agent and its environment are the instructions, "To play the game, move Kermitbot to water with the arrow keys." Once this stage is complete, the subject clicks on a button to go to another page. Again, an instructional video is at the top; the script is below.

> Good job. Now I'll describe how you're going to train the agent.
>
> [Training basics] Here's the challenge. You'll be training the robot through reward and punishment signals. The forward slash button—which is also the question mark button—gives reward. Every time you push it, it gives a little bit of reward to the robot. You'll also see the screen flash blue when you give reward. The 'z' button gives punishment and will make the screen flash red. You can think of this as similar to training a dog or another animal through reward and punishment, but it will be somewhat different.
>
> [Pre-practice pointers] I'll give you a couple of pointers now before you practice.
>
> Number one. You can reward or punish rapidly to send a stronger signal than if you just pushed it once. [1. Rapid presses = stronger feedback] So, if I saw something I really liked, I might push reward, reward, reward, reward, reward, reward (really fast, eventually inarticulate) ... [well, not quite that much] and that would be a lot stronger than if I just pressed reward.
>
> Second pointer. The robot knows that people can only respond so quickly, so don't worry about your feedback being slightly delayed. [1. Rapid presses = stronger feedback, 2. Small delays in feedback are okay.]
>
> When you're ready to start practicing, click on the box below, and follow the instructions beside it.
>
> Try your hardest, but don't be hard on yourself. The first time is often rough, and this is just practice. Remember that, as the robot is learning from you, you are also learning, learning how to teach the robot.

As on the previous page, an applet is below the video. Through the applet, the subject practices training the agent until the agent reaches the goal twice or 200 time steps pass (at 800 milliseconds each). To the right of the applet are the following instructions:

> To train the robot:
>
> - '/' rewards recent behavior
> - 'z' punishes recent behavior
> - The arrow keys do nothing. Kermitbot is in control now.
>
> To control the game environment:
>
> - '0' pauses
> - '2' unpauses

Once practice is complete, the subject clicks a button to move to another page. At the top is another video with the following script:

> [The real test] The mandatory practice is over, and the real test begins soon. If your practice training didn't go well, don't worry; it's just practice.
>
> [Closing instructions] For the real test, you'll train a little bit longer. The approximate amount of time is at the bottom of the screen. [3 minutes]
>
> Good luck, and thank you.

Through the applet below the video, the subject trains the agent in the session that is actually used as experimental data. To the right of the applet is the same set of instructions as on the previous page (indicating push-keys for training). The training session lasts for the durations reported in the article, which differ amongst the experiments. At the end of the training session, the trainer is given a "finish code" and is told to return to the original page. On this page is a questionnaire, which we use to screen for non-compliant subjects but have not analyzed further. At the end of the questionnaire, the trainer inputs the finish code received after training, completing their portion of the experiment.

Appendix B. Algorithms

Pseudocode for two of the three novel algorithms in this article are given below. A description of limited-lookahead Sarsa(λ) (Algorithm 2) can be found in Section 6.1, and Section 9.1 describes aVI-TAMER (Algorithm 3). A description of and pseudocode for the third algorithm, VI-TAMER, is in Section 4.

Algorithm 2: Limited-lookahead Sarsa(λ)

```
$\hat{R}_H$ learned from a previous record of training a TAMER agent.
Global variables: $Q,\ e,\ \hat{R}_H,\ maxDepth$

Main agent thread
 1: Initialize action-value function Q arbitrarily
 2: $e(s,a) = 0$, for all $s \in S$, $a \in A$                    // initialize eligibility traces
 3: repeat (for each episode)
 4:   $s \leftarrow getState()$
 5:   $a \leftarrow epsilonGreedySearch(s)$                               // choose action
 6:   repeat (for each step of episode)
 7:     $takeAction(a)$
 8:     wait for next time step
 9:     $s' \leftarrow getState()$
10:     $a' \leftarrow epsilonGreedySearch(s')$
11:     $\delta \leftarrow [\hat{R}_H(s,a) + \gamma Q(s',a')] - Q(s,a)$        // temporal-difference error
12:     $e(s,a) \leftarrow e(s,a) + 1$        // increment eligibility trace for current $(s,a)$
13:     for all $s \in S$, $a \in A$ do
14:       $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
15:       $e(s,a) \leftarrow \gamma\lambda e(s,a)$                          // decay eligibility traces
16:     end for
17:     $s \leftarrow s'$
18:     $a \leftarrow a'$
19:   until $s$ is terminal

function epsilonGreedySearch(s)
 1: if $random(0,1) > \epsilon$ then
 2:   $a \leftarrow greedySearch(s)$
 3: else
 4:   $a \leftarrow$ random action from uniform distribution over $A$
 5: end if
 6: return $a$

function greedySearch(s)
 1: $depth \leftarrow getRandomInteger(0, maxDepth)$          // choose search depth
 2: return $argmax_a getLookAheadVal(s, a, depth)$

function $getLookAheadVal(s, a, depthToGo)$
 1: if $depth = 0$ then return $Q(s,a)$
 2: $s' \leftarrow sampleTransition(s,a)$
 3: return $\gamma \times [\hat{R}_H(s,a) + max_{a'} getLookAheadVal(s', a', depth - 1)]$
```

Algorithm 3: The aVI-TAMER algorithm

Human interface thread and TAMER reward-modeling thread are identical to that in VI-TAMER.
Global variables: Q, $\hat{R}_H$, E_{hist}, state-action visit counter $visits(\cdot,\cdot)$
Global constants: $\{S, A, T, D, \gamma\}$ (an MDP without a reward function), $confConst$, $rolloutDepth$

Main agent thread

```
1:  Initialize human reward model $\hat{R}_H$ and action-value function Q
2:  $E_{hist} \leftarrow \{\}$; $visits(s,a) \leftarrow 0$ for all $(s,a) \in S \times A$
3:  Start value iteration, human interface, and TAMER reward-modeling
    threads
4:  repeat (for each step)
5:      $s \leftarrow getState()$
6:      $a \leftarrow argmax_a Q(s,a)$
7:      $\hat{h} \leftarrow 0$
8:      $E_{hist} \leftarrow E_{hist} \cup \{(s,a,\hat{h})\}$          // add experience sample to memory
9:      $visits(s,a) \leftarrow 0$ for all $(s,a) \in S \times A$          // reset search tree
10:     $takeAction(a)$
11:     wait for next time step
```

Value iteration thread (asynchronous, order by UCT)

```
1:  repeat (for each step of episode, possibly at regular intervals)
2:      $s \leftarrow getState()$
3:      $depthToGo \leftarrow rolloutDepth$
4:      $rollout \leftarrow getUCTRollout(s, depthToGo)$
5:      for all $(s,a) \in rollout$ do
6:          $Q(s,a) \leftarrow \left[\hat{R}_H(s,a) + \gamma \sum^{s' \in S} [T(s,a,s') \times max_{a'} Q(s',a')]\right]$
```

function getUCTRollout(s, depthToGo)

```
1:  if $depthToGo == 0$ or $isTerminalState(s)$ return {}
2:  else if $\exists a \in A$ such that $visits(s,a) = 0$ then
3:      $a \leftarrow argmax_{a \in A \ s.t. \ visits(s,a)=0} Q(s,a)$ // greedily select untaken action
4:  else
5:      $stateVisits \leftarrow \sum^{a \in A} visits(s,a)$
6:      $confBound \leftarrow confConst \times [ln(stateVisits/visits(s,a))]^2$
7:      $a \leftarrow argmax_a Q(s,a) + confBound$
8:  end if
9:  $visits(s,a) \leftarrow visits(s,a) + 1$
10: $s' \leftarrow sampleNextState(s,a)$
11: $rollout \leftarrow \{(s,a)\} \cup getUCTRollout(s', depthToGo - 1)$
12: return $rollout$
```