Auton Agent Multi-Agent Syst
DOI 10.1007/s10458-015-9283-7

Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning

Robert Loftin · Bei Peng · James MacGlashan · Michael L. Littman · Matthew E. Taylor · Jeff Huang · David L. Roberts

Abstract For real-world applications, virtual agents must be able to learn new behaviors from non-technical users. Positive and negative feedback are an intuitive way to train new behaviors, and existing work has presented algorithms for learning from such feedback. That work, however, treats feedback as numeric reward to be maximized, and assumes that all trainers provide feedback in the same way. In this work, we show that users can provide feedback in many different ways, which we describe as "training strategies." Specifically, users may not always give explicit feedback in response to an action, and may be more likely to provide explicit reward than explicit punishment, or vice versa, such that the lack of feedback itself conveys information about the behavior. We present a probabilistic model of trainer feedback that describes how a trainer chooses to provide explicit reward and/or explicit punishment and, based on this model, develop two novel learning algorithms (SABL and

R. Loftin (✉) · D. L. Roberts
Department of Computer Science, North Carolina State University, 890 Oval Drive, Campus Box 8206, Raleigh, NC 27695-8206, USA
e-mail: rtloftin@ncsu.edu

D. L. Roberts
e-mail: robertsd@csc.ncsu.edu

B. Peng · M. E. Taylor
School of Electrical Engineering and Computer Science, Washington State University, P. O. Box 642752, Pullman, WA 99164-2752, USA
e-mail: beipeng.peng@gmail.com

M. E. Taylor
e-mail: taylorm@eecs.wsu.edu

J. MacGlashan · M. L. Littman · J. Huang
Department of Computer Science, Brown University, 115 Waterman Street, Providence, RI 02912, USA
e-mail: jmacglashan@gmail.com

M. L. Littman
e-mail: mlittman@cs.brown.edu

J. Huang
e-mail: jeff_huang@brown.edu

Published online: 13 February 2015

I-SABL) which take trainer strategy into account, and can therefore learn from cases where no feedback is provided. Through online user studies we demonstrate that these algorithms can learn with less feedback than algorithms based on a numerical interpretation of feedback. Furthermore, we conduct an empirical analysis of the training strategies employed by users, and of factors that can affect their choice of strategy.

Keywords Learning from feedback · Reinforcement learning · Bayesian inference · Interactive learning · Machine learning · Human–computer interaction

1 Introduction

Within the field of artificial intelligence there exists a significant body of work on the problem of designing agents to learn behaviors from human trainers [5,14,27], and specifically on the problem of learning from trainer-provided feedback [8,17]. In many cases [12,17] feedback is treated as being representative of some numeric reward or value function associated with the underlying task. Under such an interpretation, the agent seeks to maximize the expected value of the feedback it receives for its actions.

In this work, however, we argue that trainer feedback is a more complicated form of discrete communication between the trainer and the learner. Simply treating feedback as a numeric reward signal (i.e., reward has a positive value, punishment has a negative value, and the goal is to maximize the average return), will in many cases lose information about the target behavior present in the trainer's feedback. There are of course many possible approaches to training via positive and negative feedback, which we will describe as *training strategies* throughout this work. The trainer's choice of strategy may depend on the nature of the training task, on the nature of the learning agent, and on the trainer's own background. The trainer may even change strategies in response to the agent's behavior.

In this work, we are specifically interested in how the trainer's strategy affects the use of the lack of feedback as a form of *implicit* feedback, and how such feedback should be interpreted. As a motivational example of this phenomenon, consider a common approach to dog training, where trainers will provide a large amount of explicit reward in the form of treats and conditioned rewards (i.e., *clicker training*), but very little explicit punishment. When such an approach is taken to providing feedback, the lack of explicit reward (i.e., withholding a treat from a dog) can itself be interpreted as a form of punishment, indicating that the dog's previous actions were incorrect. If, however, the reverse strategy was followed, and the trainer only provided explicit punishment, then the lack of feedback would indicate that the dog's actions were in fact correct. This work will focus on the rates at which trainers give explicit and implicit feedback for correct and incorrect actions. If the learning agent knows that implicit feedback is more likely for correct or for incorrect actions, it can use that information to make inferences about the correctness of an action, even when no explicit feedback has been given for that action.

Our work in this area has resulted in two main contributions with respect to understanding trainer strategies that we present in this paper:

1. We characterize, based on empirical data collected from real users, the types of strategies followed in practice by human trainers when teaching virtual agents, and look at potential factors that could affect those users' choices of strategy.
2. We present a probabilistic model which captures certain aspects of trainer strategy, and use that model to derive two algorithms, SABL and I-SABL, which explicitly consider

trainer strategy, and can therefore learn about the target behavior even from cases where no explicit feedback is given.

We demonstrate that these algorithms can be effective both with human and simulated trainers. Experiments show that agents using these algorithms can not only benefit from knowing the trainer's strategy, but can infer that strategy online during training.

2 Background and related work

This work is part of a growing literature on the problem of designing algorithms which can learn behaviors from human feedback. Our work is also motivated by work in psychology on how animals and humans learn from positive and negative feedback, specifically, the concept of behaviorism [23]. Based on the insights gained from that work, we develop an approach to learning from feedback which does not interpret feedback as numeric reward as most existing work does, but instead as a form of discrete communication from the trainer. Here we will discuss the existing work in machine learning as well as provide some background on the psychological underpinnings of our work.

2.1 Machine learning from human feedback

There exists a large body of work on the problem of learning from human trainers, and specifically on learning from trainer feedback. Some approaches [27] have treated human feedback as a form of guidance for an agent trying to solve a reinforcement learning (RL) [25] problem. In that work, human feedback did not change the numeric reward from the underlying RL problem, or the optimal policy, but improved exploration and accelerated learning. Their results show humans give reward in anticipation of good actions, instead of rewarding or punishing the agent's recent actions.

COBOT [11] was an online chat agent with the ability to learn from human users using RL techniques. It learned how to promote and make useful discussion in a chat room, combining explicit and implicit feedback from multiple human users. The TAMER algorithm [17] has been shown to be effective for learning from human feedback in a number of task domains common in the RL research community. This algorithm is modeled after standard RL methods which learn a value function from human-delivered numeric rewards. At each time step the algorithm updates its estimate of the reward function for a state-action pair using *cumulative* reward.

Similar to this work, other studies [16] have examined how users want to provide feedback, finding that: (1) there is little difference in a trainer's feedback whether they think that the agent can learn or that they are critiquing a fixed performance; and (2) humans can reduce the amount of feedback they give over time, and having the learner make mistakes can increase the rate of feedback. Our work differs because we focus on leveraging how humans naturally provide feedback when teaching, not how to manipulate that feedback.

Of existing work however, Policy Shaping [8] is most similar to the algorithms presented in this paper. In that work, and in ours, trainer feedback was interpreted as a discrete communication that depended probabilistically on the trainer's target policy, rather than the traditional approach of treating feedback as numeric reward. Both our work and Policy Shaping use a model of the feedback distribution to estimate a posterior distribution over the trainer's policy. In contrast to that work, ours focuses on handling different trainer feedback strategies, whereas Policy Shaping assumes actions which do not receive explicit trainer feedback are uninformative as to the trainer's policy (though still informative about the underlying MDP).

The algorithms presented in this work, however, use knowledge of the trainer's strategy to extract policy information from actions without explicit feedback. Further, our algorithms can infer this strategy from experience, and so can adapt to a particular trainer's strategy.

Other forms of feedback besides simple punishment and reward have also been explored, including feedback strategies employed by film directors, golf instructors, and 911 operators [9]. These experts gave rich feedback and direction in the form of explaining consequences, querying learner understanding, using assistive aids, etc. Other work has considered how users might assist learning algorithms by selecting a sequence of data in a classification task [14].

In addition to the work on learning from feedback, there is a growing body of work that examines how humans can teach agents by providing demonstrations of a desired behavior [3]. Learning from demonstration has been applied effectively to robot control problems, such as robot navigation [6]. Other work has leaned motion control policies that can mimic motions demonstrated by human trainer's [4]. In all of these cases, similar to learning from feedback, much of the challenge for the learning agent comes from the limited, sometimes incomplete information provided by the trainer.

Interestingly, some work has been done comparing the effectiveness of learning from demonstration against that of learning from feedback [18]. That work, however, suggested that the relative performance of the two approaches was task dependent. In addition, we note that in many cases it may not be possible for the trainers to actually demonstrate the desired behavior. Unlike most work in learning from demonstration, where the intended meaning of a trainer's demonstration is clear, in our work the meaning of the trainer's feedback can initially be ambiguous, and errors in feedback, unlike erroneous demonstrations, must be corrected as part of the learning process.

One approach to learning from demonstration is the use of algorithms for inverse reinforcement learning [1,5], where the learner attempts to identify the reward function of a Markov decision process that is consistent with a user's demonstrated actions, and identify a full policy that is optimal under that reward function. A number of different algorithms for inverse reinforcement learning (IRL) have been proposed, including maximum entropy IRL [28], which searches for a reward function that leads to a similar distribution over state trajectories as is observed in the training data. Another, similar approach is Bayesian IRL [22], which is of particular interest because it draws samples of the trainer's reward function from a posterior distribution over reward functions that is conditioned on observations of the trainer's policy. As our work assumes that the trainer's feedback depends probabilistically on the their desired policy, the same approach allows for sampling of reward functions from a distribution conditioned on feedback instead of observed actions.

We suggest that it may be possible to combine the algorithms described in this work with existing techniques for learning from demonstration, to allow an agent to learn from both feedback and demonstration simultaneously. In Sect. 8 we describe an algorithm for *maximum likelihood* inverse reinforcement learning, and show that it can be combined with our framework for learning from feedback, allowing our framework to be applied to sequential domains. We suggest that maximum likelihood IRL could also be used for learning from demonstration, making it possible to compute maximum likelihood estimates of trainer reward functions and policies given data including both feedback and demonstrations. In this work, however, we do not implement such an algorithm.

Existing work has shown that feedback can be combined with user demonstrations, for example, by using feedback to weight the value of different user demonstrations used to estimate the correct policy [2]. Other work has also shown that feedback can be combined with reward from some underlying Markov decision process, or some predefined shaping

reward [12,13]. It should be noted that in both of these examples, feedback was not given interactively, during the performance of a behavior, but was given as a critique to portions of an agent's performance that could be selected by the user after the agent had finished performing the behavior. Our work focuses on dealing with feedback given in real time, where the distinction between cases where the user is actively teaching the agent, and where the trainer is passively observing the agent's behavior is not always clear.

2.2 Behaviorism

The notion that trainers may follow different strategies while teaching is motivated by work on *behaviorism* and techniques for animal training using punishment and reward. Behaviorism, a field of psychology, considers how animals and humans learn from positive and negative feedback. Skinner introduced operant conditioning, a concept of providing feedback to modify the frequency of voluntary behaviors [23]. There are a number of ways in which punishment and reward can be combined to teach a behavior. These so-called *operant conditioning paradigms* can be grouped into four categories [24]: positive reward (R+), negative reward (R−), positive punishment (P+), and negative punishment (P−). Here, reward refers to any stimulus that would increase the frequency of an associated behavior, while punishment would be a stimulus that decreases the frequency of a behavior. Positive refers to adding a stimulus and negative refers to removing a stimulus. An example of R+ would be the act of giving a dog a treat (reward by adding a desirable stimulus). An example of P− would be the removal of a prized toy (punishment by removing a desirable stimulus). Thus, both positive and negative reward encourage an associated behavior, while both positive and negative punishment discourage an associated behavior.

Dog trainers have learned that using only positive reward (R+) to encourage desired behaviors results in fewer unintended side effects for dogs than when positive punishment (P+) is used to reduce undesired behavior [10]. We hypothesize that, in many cases, users will tend to apply this concept when training virtual agents (even if they don't realize they are doing it). We will show how in situations where users do have a bias towards R+/P− operant conditioning paradigms, learning algorithms that take these strategies into account have a significant advantage when learning from human trainers.

3 Motivations: behaviorism and trainer strategies

The goal of this work is to characterize the different strategies followed by human trainers when teaching virtual agents, and to build learning algorithms that take those strategies into account. As part of this work, we develop a probabilistic model of how feedback is provided under different strategies, and use this model both to classify strategies seen in practice, and to build probabilistic inference algorithms to learn behaviors from such feedback.

3.1 Trainer strategies

In this work, we use an idealized model of the training process, in which the learning agent takes a single action, and then *may* receive positive or negative feedback from the trainer. We hypothesize that different trainers can differ in how they provide feedback, even when teaching the same behavior. For example, when the learner takes a correct action, one trainer might provide an explicit positive feedback while another might provide no response at all.

We classify a trainer's strategy by the cases in which they give explicit feedback. Under a *balanced feedback* strategy a trainer typically gives explicit reward for correct actions

Table 1 Breakdown of strategies observed in the online user studies

Strategy	Number of training sessions exhibiting strategy
Balanced feedback	93
Reward-focused	125
Punishment-focused	6
Inactive	3

and explicit punishment for incorrect ones. A *reward-focused* strategy typically provides an explicit reward for correct actions and no response for incorrect actions, while a *punishment-focused* strategy typically provides no response for correct actions and explicit punishment for incorrect ones. An *inactive* strategy rarely gives explicit feedback of any type (making it impractical). Under a reward-focused strategy, the lack of feedback can be interpreted as an *implicit* negative feedback, while under a punishment-focused strategy, it can be interpreted as implicitly positive. ***To a strategy-aware learner, the lack of feedback can be as informative as explicit feedback.***

These strategies roughly correspond to the operant conditioning paradigms described in the behaviorism literature. A balanced feedback strategy would correspond to a R+/P+ paradigm, where both explicit punishment and explicit reward are used. A reward-focused strategy would roughly correspond to a R+/P− paradigm, while a punishment-focused strategy would correspond to a R−/P+ paradigm. An inactive strategy would correspond to a R−/P− paradigm.

We conducted three online users studies as part of this work, in which each participant went through one or more training sessions where they attempted to teach a virtual agent to perform a simple behavior. Table 1 shows the number of training sessions, from the first two of these studies, in which each of these four types of strategies was used. A user was classified as balanced if she gave explicit feedback for correct and incorrect actions more than half of the time, while inactive means she gave explicit feedback less than half the time in both cases. Reward-focused means correct actions received explicit feedback more than half the time and incorrect actions received it less than half the time; punishment-focused is the opposite case. Note that all four types were employed, but that a large percentage of users followed a reward-focused strategy. We provide this sample of results here to help emphasize the point that human trainers do follow a variety of feedback strategies. We will include a more detailed discussion of strategies in Sect. 6.

3.2 Probabilistic model of trainer strategy

One of the main contributions of this work is a formal, probabilistic model of trainer feedback. We will use this model both to characterize the strategies followed by users in the studies we conduct, and more significantly, to build learning algorithms which use probabilistic inference to identify target behaviors, while taking into account the trainer's strategy.

This probabilistic model of human feedback encapsulates differences in trainers' categorical feedback strategies. We model the learning problem as a set of discrete observations of the environment and a set of discrete actions that can be taken. The behavior being trained is represented as a *policy*, that is, a mapping from observations to actions, which in this work we will denote with λ.

Under our model, training is divided in to discrete *episodes*, in which the agent observes the state of the world, takes an action and may or may not receive some feedback from the

trainer. Our model assumes that the trainer first determines if the action taken was consistent with some target policy λ^* for the current observation, with some probability of error ϵ. The trainer then decides whether to give explicit feedback or simply do nothing. If the trainer interprets the learner's action as correct, then she will give an explicit reward with probability $1-\mu^+$, and if she interprets the action as incorrect, will give explicit punishment with probability $1-\mu^-$.[1] Therefore, when accounting for error in the trainer's interpretation, if the learner takes a correct action it will receive explicit reward with probability $(1-\epsilon)(1-\mu^+)$, explicit punishment with probability $\epsilon(1-\mu^-)$, and will receive no feedback with probability $(1-\epsilon)\mu^+ + \epsilon\mu^-$.

The parameters $\mu^+ \in [0, 1]$ and $\mu^- \in [0, 1]$ represent the trainer's preference for giving neutral feedback for correct and incorrect actions, respectively, and encode the trainer's feedback strategy. For example, $\mu^+ = 0.1$, $\mu^- = 0.1$ correspond to a balanced feedback strategy where nearly every action receives explicit feedback, while $\mu^+ = 0.1$, $\mu^- = 0.9$ correspond to a reward-focused strategy, where only actions interpreted as correct tend to receive explicit feedback. Putting these elements together, for time step t (each time step corresponds to an episode with the agent observing the world, choosing an action and receiving feedback), we have a distribution over the feedback f_t conditioned on the observation o_t, action a_t, and the trainer's target policy λ^*,

$$p\left(f_t = f^+ | o_t, a_t, \lambda^*\right) = \begin{cases} (1-\epsilon)(1-\mu^+), & \lambda^*(o_t) = a_t \\ \epsilon(1-\mu^+), & \lambda^*(o_t) \neq a_t, \end{cases} \tag{1}$$

$$p\left(f_t = f^- | o_t, a_t, \lambda^*\right) = \begin{cases} \epsilon(1-\mu^-), & \lambda^*(o_t) = a_t \\ (1-\epsilon)(1-\mu^-), & \lambda^*(o_t) \neq a_t, \end{cases} \tag{2}$$

$$p\left(f_t = f^0 | o_t, a_t, \lambda^*\right) = \begin{cases} (1-\epsilon)\mu^+ + \epsilon\mu^-, & \lambda^*(o_t) = a_t \\ \epsilon\mu^+ + (1-\epsilon)\mu^-, & \lambda^*(o_t) \neq a_t. \end{cases} \tag{3}$$

where f^+ is an explicit positive feedback, f^- is an explicit negative feedback, and f^0 represents a lack of feedback.

What is important to note about this model is that, depending on the strategy (and the corresponding μ^+ and μ^- parameters) used, the lack of feedback may be more probable for correct actions than incorrect actions, or vice versa. Therefore, the correct inference to make from a lack of feedback depends on the training strategy being used. This model formalizes the idea that learning depends on the training strategy being employed.

3.3 Numeric reward versus discrete feedback

We can compare the discrete, probabilistic interpretation of trainer feedback used in this work against the numerical reward interpretation used in much of the existing literature. Under the numeric interpretation, each action receives a continuously valued reward signal, and the agent attempts to find the action which maximizes the average reward received. We argue (and our experimental results support) that trainer feedback can be interpreted more effectively as a form of discrete communication between the trainer and the learner, allowing the agent to learn the desired behavior in less time and with less effort on the part of the trainer.

There are some interpretations of trainer feedback that are more easily represented under a numerical interpretation, and that are not modeled by our probabilistic interpretation.

[1] Note that for the μ parameters, $+$ and $-$ distinguish reward and punishment, and not explicit/implicit feedback as in the R+/P+ notation taken from the behaviorism literature.

Specifically, our model does not consider the magnitude of the feedback provided for an action, either the magnitude of an individual feedback signal, or the number of feedback signals given in response to a single action. Under a numerical interpretation, an individual feedback signal can be assigned different numerical values (e.g., the verbal response "good" being given a smaller value than "Great!"). Under such an interpretation, a single action can also receive multiple feedback signals, with the total value of these signals being assigned to an action. Therefore, it is possible under a numerical interpretation of feedback for two actions to each receive positive feedback, but one action to have a greater estimated value than the other, and so be assumed to be preferable to the other.

Under our model, however, we do not directly consider the relative utility of one action versus another. One action is not considered to be more "correct" than another, but is instead considered more likely to be the correct action. It would be possible to interpret the magnitude of feedback under our model, with the magnitude of feedback representing the *certainty* on the part of the trainer that the action was correct. That is to say, that the trainer is less likely to erroneously give a feedback signal of large magnitude than one of small magnitude. Therefore, when comparing two actions that have each received one feedback signal, the action that had received the larger magnitude signal would be considered more likely to be correct. Similarly, when more than one feedback signal is given in response to an action, we would assume that if the action were not correct then the trainer would have to have repeatedly given erroneous feedback signals to get such a response. As making multiple incorrect feedback signals is far less likely than making a single incorrect feedback, this would mean that an action receiving multiple positive feedback signals is more likely to be correct than one that has received only a single positive feedback.

It is not clear, however, that the magnitude of users feedback actually reflects the relative utility of the action for which that feedback is given. Similarly, the number of individual feedback signals may not directly relate to the users preference for one action over another. There may be many possible interpretations for feedback of differing magnitude and frequency, some of which may not convey much information about the correctness of an action. For example, differences in the number or magnitude of feedback signals may be the result of frustration on the part of the trainer, or some global measure of the agent's performance.

In the user studies presented in this work, we explicitly choose to consider neither the magnitude of a feedback signal (we only allow for one level of positive and negative feedback), nor the number of feedback signals given for a single action (our learning algorithms only consider the final feedback given in response to an action). This interpretation gives trainers the opportunity to correct a mistaken feedback immediately after giving it. Additionally, mistakes early in the training process are easier to overcome, as the trainer does not need to provide a large amount of feedback to outweigh the previous, incorrect feedback. As this work will demonstrate, learning algorithms which treat feedback as something other than a numeric reward signal, and which explicitly consider multiple possible interpretations of feedback, can be much better suited to learning from human trainers.

4 Strategy-aware Bayesian learning

In this work, we develop algorithms for learning from feedback that account for differences in trainer strategy. Specifically, we take advantage of the fact that, under reward-focused and punishment-focused training strategies, the lack of any feedback can convey as much

Algorithm 1 The SABL algorithm. The feedback distribution $p(f_t|o_t, a_t, \lambda^*(o_t) = a')$ is described by Eqs. 1, 2 and 3. $takeAction(a_t)$ does not return until the episode finishes.

$\forall o \in O, a \in A\colon P[o, a] \leftarrow \frac{1}{|A|}$
$t \leftarrow 0$
while user has not terminated learning **do**
  $o_t \leftarrow observeWorld()$
  $a_t \leftarrow \text{argmax}_{a' \in A} P[o_t, a']$
  $takeAction(a_t)$
  $f_t \leftarrow receiveFeedback()$
  for all $a' \in A$ **do**
    $P[o_t, a'] \leftarrow p(f_t|o_t, a_t, \lambda^*(o_t) = a')P[o_t, a']$
  end for
  $P[o_t, \cdots] \leftarrow normalize(P[o_t, \cdots])$
  $t \leftarrow t + 1$
end while

information about the target behavior as explicit feedback. We will demonstrate experimentally that this approach allows agents to learn behaviors in less time, and with fewer feedbacks, when compared to approaches that ignore trainer strategy.

4.1 The SABL algorithm

Here we present the strategy-aware Bayesian learning (SABL) algorithm. The SABL algorithm assumes that trainer feedback is provided according to the probabilistic model presented previously. Using this model of feedback, SABL computes a maximum likelihood estimate of the trainer's target policy λ^* given the feedback that the user has provided; that is, it computes

$$\underset{\lambda}{\text{argmax}}\, p(h_{1\ldots t}|\lambda^* = \lambda),$$

where h_t is the training history of actions, observations, and feedback. If a user provides multiple feedbacks during an episode, SABL only considers the most recent, allowing a user to correct a mistaken feedback. Algorithm 1 is an outline of SABL. Note that only the current likelihood distribution is needed to compute the likelihood given a new episode, and therefore the full training history does not need to be considered when updating the policy probabilities.

SABL requires that we specify the trainer's strategy before learning, but in practice we are unlikely to know what that strategy will be, as trainers may use a variety of strategies. Specifying an incorrect strategy can severely degrade the performance of the algorithm. For example, if the trainer follows a reward-focused strategy, while the agent assumes that they follow a punishment-focused strategy, then the agent will interpret the lack of feedback as indicating that the previous action was correct, when in reality the lack of feedback means the previous action was incorrect.

Specifying that the trainer's strategy is balanced, that is, $\mu^+ = \mu^-$, will cause the agent to ignore episodes where no feedback is given, and while it will prevent the agent from harmfully misinterpreting the lack of feedback, it will also prevent it from gaining any knowledge from such episodes. In the next section we will extend SABL to allow it to infer the trainer's strategy online based on the training history.

Algorithm 2 The I-SABL algorithm. The $EMupdate(\lambda, h)$ function computes a new policy according to Eq. 4.

```
λ ← randomPolicy()
h ← ⟨⟩
t ← 0
while user has not terminated learning do
    o_t ← observeWorld()
    a_t ← λ(o_t)
    takeAction(a_t)
    f_t ← receiveFeedback()
    h ← ⟨h_0, ... h_{t−1}, (o, a, f)⟩
    λ ← randomPolicy()
    repeat
        λ′ ← λ
        λ ← EMupdate(λ, h)
    until λ == λ′
    t ← t + 1
end while
```

4.2 SABL for unknown strategies: inferring-SABL

While SABL will perform well when it knows the trainer's μ^+ and μ^- parameters, in practice the trainer's strategy will likely be unknown. If, however, the learner knows from explicit feedback the correct action for some observations, it can infer the strategy by looking at the history of feedback for those observations. For example, if more explicit feedback is given for correct actions than incorrect ones, then the strategy is likely reward-focused. Under SABL's probabilistic model we can treat the unknown μ values representing the trainer's strategy as hidden parameters, and can marginalize over possible strategies to compute the likelihood of a possible target policy λ. Inferring-SABL, or I-SABL, finds a maximum likelihood estimate of the target policy, given the training history. I-SABL attempts to find

$$\operatorname*{argmax}_{\lambda} \sum_{s \in S} p(h_{1\ldots t}, s | \lambda^* = \lambda),$$

where S is the set of possible training strategies (μ^+, μ^- values), $p(s)$ is uniform for all $s \in S$, and $h_{1\ldots t}$ is the training history up to the current time t.

In some domains it will be possible to restrict the space of possible policies such that the marginal likelihood of each policy can be explicitly computed. In the general case, however, the space of possible policies will be exponential in the number of observations, and so algorithms for approximate inference may be needed. In this work we use the Expectation Maximization [7] algorithm in such cases to compute a maximum likelihood estimate of the target policy, and treat the unknown μ^+ and μ^- parameters as continuous, hidden variables ranging from 0 to 1. The ith EM update step is then

$$\begin{aligned}\lambda_{i+1} &= \operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 p(\mu^+, \mu^- | h, \lambda_i) \ln \left[p(h, \mu^+, \mu^- | \lambda) \right] d\mu^+ d\mu^- \\ &= \operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 p(\mu^+, \mu^- | h, \lambda_i) \ln \left[p(h | \mu^+, \mu^-, \lambda) p(\mu^+, \mu^- | \lambda) \right] d\mu^+ d\mu^-,\end{aligned}$$

where λ_i is the current estimate of the policy and λ_{i+1} is the new estimate of the policy.

As the μ parameters are continuous, we integrate over their range, which is the unit square. Because we have no prior knowledge that one strategy is more likely than another, we assume that all possible combinations of μ parameters are equally probable, and independent of the desired policy (with enough data the EM algorithm should estimate a similar distribution over values regardless of our initial assumption). The probability density function over these parameters $p(\mu^+, \mu^-|\lambda) = 1$, and can therefore be divided out leaving

$$
\begin{aligned}
&= \operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 p(\mu^+, \mu^-|h, \lambda_i) \ln p(h|\mu^+, \mu^-, \lambda) d\mu^+ d\mu^- \\
&= \operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 \frac{p(h|\mu^+, \mu^-, \lambda_i) p(\mu^+, \mu^-|\lambda_i)}{p(h|\lambda_i)} \ln p(h|\mu^+, \mu^-, \lambda) d\mu^+ d\mu^- \\
&= \operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 \frac{p(h|\mu^+, \mu^-, \lambda_i)}{p(h|\lambda_n)} \ln p(h|\mu^+, \mu^-, \lambda) d\mu^+ d\mu^- \\
&= \operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 p(h|\mu^+, \mu^-, \lambda_i) \ln p(h|\mu^+, \mu^-, \lambda) d\mu^+ d\mu^-,
\end{aligned}
$$

with the marginal probability $p(h|\lambda_n)$ removed as a constant. Computing this quantity is still computationally intractable, as it must be optimized over all possible values of λ. If the training history h is replaced by histories h^o, for all observations $o \in O$, then the update becomes

$$
\begin{aligned}
&\operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 p(h|\mu^+, \mu^-, \lambda_i) \ln \prod_{o \in O} p(h^o|\mu^+, \mu^-, \lambda(o))) d\mu^+ d\mu^- \\
&= \operatorname*{argmax}_{\lambda \in P} \int_0^1 \int_0^1 p(h|\mu^+, \mu^-, \lambda_i) \sum_{o \in O} \ln(p(h^o|\mu^+, \mu^-, \lambda(o))) d\mu^+ d\mu^- \\
&= \operatorname*{argmax}_{\lambda \in P} \sum_{o \in O} \int_0^1 \int_0^1 p(h|\mu^+, \mu^-, \lambda_i) \ln(p(h^o|\mu^+, \mu^-, \lambda(o))) d\mu^+ d\mu^-.
\end{aligned}
$$

With this form, we can now optimize each observation-action mapping individually, for each $o \in O$. The logarithmic term can be further simplified by splitting up the history h^o into episodes with positive, $h^{o,+}$, and negative, $h^{o,-}$ feedback, as well as, $h^{o,0}$, episodes without feedback:

$$
\begin{aligned}
&\ln p(h^o|a, \mu^+, \mu^-) \\
&\quad = \ln \left[p(h^{o,+}|a, \mu^+, \mu^-) p(h^{o,-}|a, \mu^+, \mu^-) p(h^{o,0}|a, \mu^+, \mu^-) \right] \\
&\quad = \ln p(h^{o,+}|a, \mu^+, \mu^-) + \ln p(h^{o,-}|a, \mu^+, \mu^-) + \ln p(h^{o,0}|a, \mu^+, \mu^-).
\end{aligned}
$$

Let $|h^{o,+}|$ be the total number of episodes where positive feedback was received following observation o, and let $|h_a^{o,+}|$ be the number of episodes with positive feedback given for action a after observation o. Put differently, $|h_a^{o,+}|$ is the total number of episodes with observation o where the correct action was taken, and positive feedback was given, assuming that a is in fact the correct action for o. Further, let the values $|h^{o,-}|$ and $|h_{a,}^{o,-}|$ be defined analogously, but for negative feedback. We can then simplify the first term as

$$
\begin{aligned}
&\ln p(h^{o,+}|a,\mu^+,\mu^-)\\
&\quad=\ln\left[((1-\epsilon)(1-\mu^+))^{|h_a^{o,+}|}(\epsilon(1-\mu^+))^{|h^{o,+}|-|h_a^{o,+}|}\right]\\
&\quad=\ln\left[\left(\frac{(1-\epsilon)}{\epsilon}\right)^{|h_a^{o,+}|}(\epsilon(1-\mu^+))^{|h^{o,+}|}\right]\\
&\quad=|h_a^{o,+}|\ln\left(\frac{(1-\epsilon)}{\epsilon}\right)+|h^{o,+}|\ln\epsilon(1-\mu^+).
\end{aligned}
$$

We can similarly simplify the second term of the summation as

$$\ln p(h^{o,-}|a,\mu^+,\mu^-)=|h_a^{o,-}|\ln\left(\frac{(\epsilon)}{1-\epsilon}\right)+|h^{o,-}|\ln(1-\epsilon)(1-\mu^+).$$

The terms $|h^{o,+}|\ln\epsilon(1-\mu^+)$ and $|h^{o,-}|\ln(1-\epsilon)(1-\mu^+)$ can be dropped from the maximization as they do not depend on a, and the remaining terms can be pulled out of the integration, as they do not depend on the μ parameters.

The final term, which does depend on the μ parameters, simplifies to

$$
\begin{aligned}
&\ln p(h^{o,0}|a,\mu^+,\mu^-)\\
&\quad=\ln\left[((1-\epsilon)\mu^+ +\epsilon\mu^-)^{|h_a^{o,0}|}(\epsilon\mu^+ +(1-\epsilon)\mu^-)^{|h^{o,0}|-|h_a^{o,0}|}\right]\\
&\quad=|h_a^{o,0}|\ln\left[\frac{(1-\epsilon)\mu^+ +\epsilon\mu^-}{\epsilon\mu^+ +(1-\epsilon)\mu^-}\right]+|h^{o,0}|\ln(\epsilon\mu^+ +(1-\epsilon)\mu^-),
\end{aligned}
$$

where $|h^{o,0}|$ and $|h_a^{o,0}|$ represent the number of episodes where no feedback was given after observation o, and the number of episodes where no feedback was given for action a taken after observation o. Once again the second term does not depend on the correct action and so can be removed from the optimization.

Therefore, the EM update can be simplified to maximizing the following term for a policy's action separately for each observation o:

$$\lambda_{i+1}(o)=\underset{a\in A}{\operatorname{argmax}}\left[\alpha(|h_a^{o,+}|-|h_a^{o,-}|)+\beta|h_a^{o,0}|\right],\tag{4}$$

where we define values

$$
\begin{aligned}
\alpha&=\ln\left[\frac{(1-\epsilon)}{\epsilon}\right]\int_0^1\int_0^1 p(h|\mu^+,\mu^-,\lambda_i)d\mu^+d\mu^-,\ \text{and}\\
\beta&=\int_0^1\int_0^1 p(h|\mu^+,\mu^-,\lambda_i)\ln\left[\frac{(1-\epsilon)\mu^+ +\epsilon\mu^-}{\epsilon\mu^+ +(1-\epsilon)\mu^-}\right]d\mu^+d\mu^-,
\end{aligned}
$$

as simplifications of the expectation step, which can be computed once for each EM update. Algorithm 2 gives an outline of the full I-SABL learning algorithm.

5 User studies

As part of this work we conducted two sets of online user studies (one set with volunteers recruited via email and another using Amazon Mechanical Turk) that addressed two main questions. First, we wanted to understand how users provide feedback (or choose not to provide feedback), when teaching virtual agents. Second, we wanted to evaluate the effectiveness of the SABL and I-SABL learning algorithms against algorithms based on a numerical interpretation of reward.

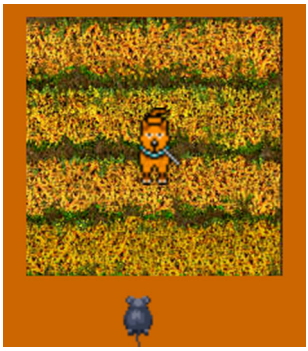

Fig. 1 A *screenshot* of the study interface. Additional buttons that begin and end training have been cropped out

In each study, participants trained a virtual agent to move towards objects as they approached from different sides of the screen. In our volunteer studies, this agent was represented by a sprite of a dog and the object to be approached was represented as a rat, which would run away when the dog moved towards it. The Mechanical Turk studies also used these images in some experiments, but in addition used other visual representations to gauge the effect of the agent's appearance on the user's behavior. The learning task used in all of the user studies could be described as a contextual bandit domain [20], where the agent can observe the state of the world, and take some action, but its actions have no effect on the probability of subsequent states of the world occurring, only on the probability of the immediate feedback.

As we are interested in how the participants' backgrounds affected their training strategies (particularly their background with dog training), we had participants fill out surveys before they began the study. In these surveys the participants were asked to indicate their age, gender, education, history with dog ownership, experience in training dogs, and with which dog-training paradigms they were familiar (if any).

Before beginning training, users were taken through a tutorial, which first animated approaching objects and then instructed the user how to reward and punish the learner. After the tutorial, the users began a series of training sessions; each session was performed with a different virtual agent that learned from scratch. The user was told that each session required that the agent be trained from scratch.

In our training task, the learning agent began at the center of the screen, and the objects arrived once every two seconds from the edges of the screen. The objects came from three points along each of the four edges, resulting in 12 possible observations. When an object appeared, the agent moved from the center towards one of the edges. If the learner moved towards the edge from which the object was coming, that object was chased away. If the learner ran to a different edge, the object entered the field in the center and disappeared. Figure 1 shows the agent and task environment with the dog and rat sprites used in most of our user studies.

To train the learner to chase the objects away, users could provide reward, punishment, or no feedback. Users signaled when training was complete by pressing a button. Data for a training session was included only if it was terminated by the user signaling it was complete.

To get a better understanding of how users chose to train the agents, after each training session participants were shown a textual input box, and were asked: "Please describe the strategy you used when training the [agent] during the previous experiment. For example, when did you provide reward/punishment or when did you decide to change the task or start over (if appropriate)? Is there anything else you want to say about training the [agent]?"

5.1 Volunteer studies

The first set of studies we conducted (the volunteer studies) focused on how training strategies differed between users for a fixed training task, and on how a user's prior training experience affected their choice of strategy. As such, the learning agent in these studies was represented as a drawing of a dog, and the approaching object as a rat. In both the first and second studies, each training session used a different learning algorithm (in random order). The two volunteer studies also evaluated the SABL and I-SABL learning algorithms developed as part of this work, comparing them against two RL-based algorithms, M_{-0} and M_{+0} (discussed in Sect. 7). Specifically, M_{-0}, M_{+0}, and the SABL algorithm, were evaluated in one study, while the SABL and I-SABL algorithms were evaluated in a second study.

Participants for the two volunteer studies (which we will refer to as volunteer study 1 and volunteer study 2, respectively) were recruited from three different sources: (1) a senior-level game design class at North Carolina State University (credit was offered for participation), (2) the North Carolina State University computer science departmental mailing list, and (3) two Internet communities focused on dog training (a Facebook group about positive-reinforcement training and a Japanese dog forum). Although the recruiting sources were the same for both volunteer studies, the distribution from each source was different since recruitment was performed at different times.

5.2 Amazon mechanical turk studies

In another set of user studies, we considered how the training task itself, and the interface provided to the trainer, would affect their choice of strategy. We were particularly interested in whether the appearance of the agent would affect strategy choice, and whether feedback from the agent to the user would affect that choice.

To efficiently recruit a large number of participants, we ran this set of studies using the Amazon Mechanical Turk system, which allowed us to present the studies to a large pool of Mechanical Turk users, and to provide each of them with a small amount of compensation for completing the study. Each study was defined within Mechanical Turk as a Human Intelligence Task, and participants were given a base compensation of $0.25, and were given a bonus if the agent reached 90 % policy accuracy. We had a total of eight separate Human Intelligence Tasks (which we will denote as AMT 1 through AMT 8) which were published to Mechanical Turk. Each task had its own set of experimental conditions, and its own set of participants, though it is possible that some users participated in more that one task. Table 2 summarizes the eight individual studies published through Mechanical Turk. It should be noted that the learning algorithm used to control the agents in these studies was chosen at random each time, and was either the balanced feedback version of the SABL, or the I-SABL algorithm.

There were two main sets of conditions in the Mechanical Turk studies. AMT 1 through AMT 3 looked at how changing the visual representation of the agent affected the choice of strategy, the assumption being that users would be less likely to punish an agent that appeared as a dog than they would an agent that appeared as an inanimate object, or as an animal with less positive associations, such as a snake (see Fig. 2). The sprite of the approaching object was also changed to be more appropriate given the agent's sprite. The alternative sprite combinations used for these studies are shown in Table 2. AMT 1 switched randomly between the dog/rat and robot/battery sprites, while AMT 2 tested only the snake/bird combination, and AMT 3 used only the arrow/square combination. In AMT 1, 2 and 3 the participant received a bonus of $0.25 for reaching 90 % policy accuracy.

Table 2 Summary of Amazon Mechanical Turk studies, results of which are discussed in Sects. 6.2, 6.3 and 6.4

Study	Sprite	Condition	Performance bonus	No. participants
AMT 1	Dog/rat or robot/battery	Agent appearance	$0.25	162
AMT 2	Snake/bird	Agent appearance	$0.25	162
AMT 3	Arrow/square	Agent appearance	$0.25	120
AMT 4	Dog/rat	Policy accuracy	$0.25	30
AMT 5	Dog/rat	Audible response	$0.25	30
AMT 6	Dog/rat	Increased bonus	$0.75	30
AMT 7	Dog/rat	Policy accuracy	$0.75	30
AMT 8	Dog/rat	Audible response	$0.75	30

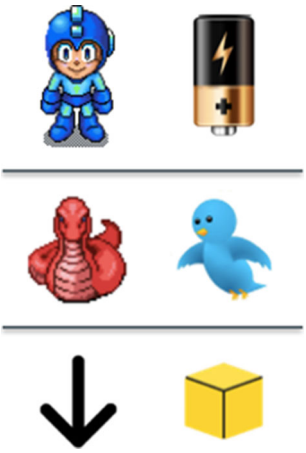

Fig. 2 Alternative sprite combinations used in the Mechanical Turk Studies, in addition to being represented as a dog, the agent could also have been a robot, a snake, or an arrow

AMT 4 through AMT 8 looked at how feedback given by either the training interface or the agent itself would affect the strategy used, and each used the dog/rat sprite combination. In AMT 4 and AMT 7, the interface showed the user the percentage of the agent's policy that was correct at that moment. AMT 5 and AMT 8 had the agent give an audible cry when it was punished.[2] AMT 4 and 5 gave participants a bonus of $0.25 for reaching 90 % accuracy, while AMT 6, 7 and 8 gave bonus of $0.75. AMT 6 gave no special feedback to the user, and was meant to evaluate the effect of increasing the bonus to $0.75.

6 Analysis of training strategies used in practice

Our main hypothesis in conducting these studies was that human trainers follow a variety of strategies when teaching behaviors using feedback. As such, we characterized the distribution of different training strategies, and the factors that influenced that distribution.

We used our probabilistic model of the training process to categorize the strategies that participants in our studies followed. As discussed previously in Sect. 3.1, we group strategies into four categories by the conditions under which they do and do not provide explicit feed-

[2] Though users were instructed to enable their computer speakers, we have no way of knowing whether the participant could actually hear the dog cry.

Table 3 Breakdown of strategies used in AMT 1, 2 and 3 when training an agent appearing as a dog, robot, snake or arrow

Agent	Balanced feedback	Reward-focused	Punishment-focused	Inactive
Dog	151	25	1	1
Robot	188	21	0	4
Snake	64	7	2	3
Arrow	43	6	1	2

back (balanced feedback, reward-focused, punishment-focused and inactive). Specifically, we estimated the μ^+ and μ^- parameters used for each training session by computing the fraction of correct and incorrect actions that did not receive explicit feedback. The strategy for a session was classified as balanced if both μ^+ and μ^- were less than $\frac{1}{2}$ (recall that low μ^+ and μ^- values correspond to frequent *explicit* feedback). If μ^+ was less than $\frac{1}{2}$ while μ^- was greater than $\frac{1}{2}$, the strategy was classified as reward-focused, while if the opposite case was true the strategy was classified as punishment-focused. The strategy was classified as inactive if both μ^+ and μ^- were greater that $\frac{1}{2}$.

We first consider the results of volunteer studies 1 and 2. In those studies we are primarily interested in the overall distribution of strategies used, as well as how the user's background influenced their choice of strategy. We only consider data from the 105 users (between the two studies) who completed at least one training session. Table 1 in Sect. 3.1 summarizes the distribution of training strategies from volunteer studies 1 and 2.

Recall that some participants for these studies were explicitly recruited due to their experience in training dogs and they trained a learner depicted with a dog sprite (Fig. 1). Overall, the dominant strategies in these studies were reward-focused (frequent rewards, few punishments) and balanced feedback (frequent rewards and punishments). The least used strategy was inactive, which is reassuring, as the use of such a strategy could indicate that users were confused about the task or interface, or were not fully engaged with the task.

We expected the balanced feedback strategy to be common, because the strategy represents providing as much information to the learner as possible. As one participant described it in the post-experiment survey, "I just punished the dog if they went to the wrong side and rewarded them when they went to the right side." We also expected to see many users using reward-focused strategies, since that is a common dog-training paradigm. One participant explained, "I tried to Reward only. Rewarded when the dog was moving or had moved toward the rat, and provided no opportunity for Reward when the dog moved away from the rat."

Table 3 summarizes the distribution of strategies used in the first three Mechanical Turk studies (AMT 1, 2 and 3). We only report data from training sessions where at least 50 % policy accuracy was achieved.[3] In this study, unlike the first two, balanced feedback strategies were much more common than reward-focused strategies. However, reward-focused strategies were still common, and still occurred much more frequently than punishment-focused or inactive strategies.

We note that a few participants changed strategies during training sessions. We divided each experiment at its temporal midpoint, and classified the strategy used for the first half of the experiment, and that used for the second half. Table 4 shows how users changed strategies over time in volunteer studies 1 and 2. Overall, changing strategy was uncommon, with 84.7 % of training sessions in study 1 following a single strategy. There were however, a

[3] We exclude more data in the Mechanical Turk studies to remove participants who do the minimum amount of work to receive their compensation.

Table 4 The number of participants beginning a training session using one strategy *(rows)* and ending it using another *(columns)*. Entries on the diagonal indicate that no change occurred

Beginning Strategy	Ending strategy			
	Balanced feedback	Reward-focused	Punishment-focused	Inactive
(a) Volunteer study 1				
Balanced feedback	65	4	2	0
Reward-focused	10	52	1	1
Punishment-focused	2	1	4	1
Inactive	0	0	0	1
(b) Volunteer study 2				
Balanced feedback	17	2	0	0
Reward-focused	2	59	0	1
Punishment-focused	0	0	0	0
Inactive	0	1	1	0

number of cases where users switched from a reward-focused strategy to a balanced feedback strategy, which occurred in 6.9 % of training sessions in study 1. This change may have been an attempt by the users to preserve the desired behavior once it had been learned, that is, once the agent was taking mostly correct actions, incorrect actions were singled out for explicit punishment.

Our probabilistic model does not explicitly account for changes in strategy, though it could be extended to do so. While existing work has addressed trainers changing their strategies by actively encouraging users to give certain types of feedback [19], it may be more effective to integrate the notion of strategy change with an overall model of trainer feedback, such as the one presented here.

6.1 Effects of dog-training experience

As we are interested in the degree to which a participant's experience with training dogs influenced their strategy, we asked each user to rate their level of experience in dog training on a four-point scale from "None" to "I am an Expert." Many participants had no experience training dogs, and those that did varied in their degree of experience.

To visualize these results, we organize the data into a contingency table and depict it as a *residual mosaic plot* (see Fig. 3a). There are a few important things to note about such plots. The data is organized into boxes, with one column of boxes for each value of one of the categorical variables. The order of the boxes within each column follows the set of values of the other categorical variable. The area of a box in the plot indicates the number of responses in that category. The width of each box represents, in aggregate, the probability that a response will fall into that column, regardless of which row it is in, *e.g.*, $\Pr(\text{Experience} = \text{some})$.

The height of a box indicates the amount of data in that column when the value of the row is considered, *e.g.*, $\Pr(\text{Strategy} = \text{reward} - \text{focused}|\text{Experience} = \text{none})$. Thus, the more asymmetric any box is, the more it deviates from the expected value; tall thin rectangles indicate more data in that entry than expected and short wide rectangles indicate fewer data in that entry than expected.

Additionally, the color of an entry indicates whether or not the rectangular shape of an entry represents a significant deviation from the expected value. A shaded entry means that

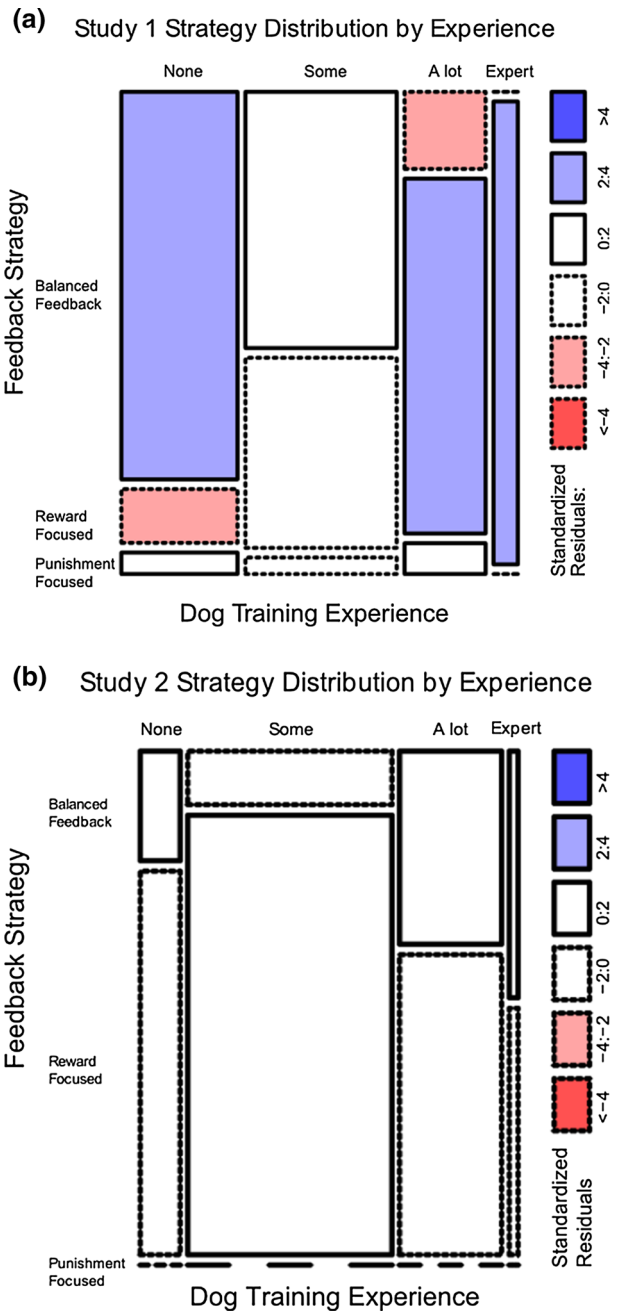

Fig. 3 Mosaic plots (generated with the R language) with Pearson residuals for strategies in the volunteer studies, grouped by dog-training experience (Note that *boxes* with *solid borders* indicate a deviation *above* the expected value, while *boxes* with *dotted borders* indicate a deviation *below* the expected value). Users with no experience were more likely to use balanced-feedback strategy, users with some experience were more likely to use a reward-focused strategy. For volunteer study 1, differences were 2–4 standard deviations from expected (significant with $p < 0.05$) **a** Volunteer study 1 **b** Volunteer study 2

the value that box represents is more than two standard deviations above (or below) the expected value, and is therefore significant with $p < 0.05$. If the border of the cell is solid, then the deviation is above the expected value, if it is dashed, it is below expected.

Figure 3a shows the relationship between dog-training experience and the employed feedback strategy in a mosaic plot, for participants in volunteer study 1. As a common approach to dog training is to only use positive feedback, we expected that users with dog-training experience would be more likely to use reward-focused strategies than those without experience.

Indeed, in volunteer study 1, we found that the more dog-training experience a user had, the more likely they were to use a reward-focused strategy. This relationship was found to be statistically significant at the 95 % confidence level. However, this relationship did not appear as strong in volunteer study 2 in which users with at least some experience were very likely to use reward-focused strategies (Fig. 3b). This difference likely reflects differences in the distribution of participants between the two studies, with the second study having only four participants with no training experience.

Both volunteer studies 1 and 2 specifically recruited participants with dog-training experience, and that choice almost certainly affected the observed frequency of different strategies. The Mechanical Turk studies, however, should have no bias towards users with training experience.

6.2 Effect of agent appearance

The Mechanical Turk studies focused primarily on how different aspects of the training task and the interface would affect the training strategies used. AMT 1, 2 and 3 considered the question of whether the appearance of the agent would affect the distribution of strategies used, either because users believed that an agent resembling a dog would respond better to strategies that are effective with real dogs, or because the appearance of an animal made users more averse to giving punishment. Recall that the Mechanical Turk studies asked the user to teach the same behavior as in the volunteer studies, but varied the sprites between a dog/rat, robot/battery, snake/bird, or arrow/box.

As shown in Table 3, the distribution of strategies in AMT 1, 2 and 3 was relatively insensitive to the agent's appearance. Fisher's exact test shows that the number of times each of the four strategies was used was not significantly different ($p > 0.21$) between subjects training the dog and those training the robot. Similarly, we did not see differences in strategies between the snake and the arrow ($p = 0.10$).

Despite a lack of statistically significant findings, there is some weak evidence that the learning agent's sprite did influence trainers' choices of strategies. Consider Fig. 4, which shows the distribution of dog-training experience for those trainers that used a reward-focused strategy, grouped by sprite. What is interesting to note is that participants with dog training experience used reward-focused strategies in roughly equal proportion when training the dog and the robot; however, for participants without dog training experience, it appears a higher percentage used the reward-focused strategy on the dog when compared to the robot. One plausible explanation is that empathy toward the dog caused users to avoid explicit punishment, even if they were unfamiliar with dog-training techniques.

6.3 Effect of feedback from the agent

We also consider how having the agent or the training interface provide some feedback to the trainer might influence their choice of strategy. AMT 4 and 5, and AMT 7 and 8, looked at the

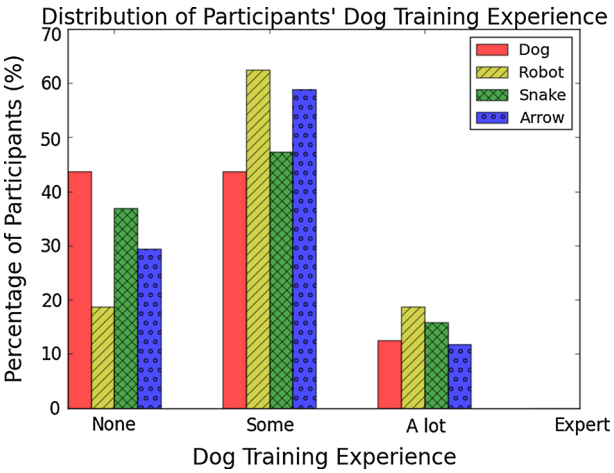

Fig. 4 The distribution of participants in AMT 1, 2 and 3 who used a reward-focused strategy, based on their experience with dog training, grouped by the sprite they were training

Table 5 Breakdown of strategies used when training a dog with policy-accuracy displayed and a dog with sound, as well as when training a dog with the $0.25 performance bonus and with the increased $0.75 bonus

Experiment	Performance bonus	Training conditions	Balanced feedback	Reward-focused	Punishment-focused	Inactive
AMT 1	$0.25	Base	151 (85 %)	25 (14 %)	1 (.5 %)	1 (.5 %)
AMT 4	$0.25	Policy accuracy	32 (84 %)	3 (8 %)	1 (2.7 %)	2 (5.3 %)
AMT 5	$0.25	Audible response	18 (72 %)	6 (24 %)	0 (0 %)	1 (4 %)
AMT 6	$0.75	Base	46 (88 %)	5 (10 %)	0 (0 %)	1 (2 %)
AMT 7	$0.75	Policy accuracy	38 (79 %)	7 (15 %)	1 (2 %)	2 (4 %)
AMT 8	$0.75	Audible response	33 (72 %)	11 (24 %)	1 (2 %)	1 (2 %)

effects of providing feedback to the trainer in various forms. AMT 4 and AMT 7 displayed the current percentage of the learner's policy that was correct, while AMT 5 and AMT 8 had the dog give an audible cry in response to punishment. As AMT 7 and 8 increased the participants performance bonus to $0.75 from $0.25, we consider AMT 6 to be a baseline against which to compare AMT 7 and 8, while AMT 1 with the dog sprite would be a baseline for AMT 4 and5, as it only had a $0.25 performance bonus.

Table 5 summarizes the frequency of strategies used by the human trainers in these studies. The results are in line with the other Mechanical Turk studies, with the dominant training strategy being balanced feedback, followed in popularity by reward-focused. Fisher's exact test shows that the number of times each of the four strategies was used was not significantly significant ($p = 0.25$) between AMT 4 and AMT 5, nor were the differences between AMT 6, 7 and 8 significant ($p = 0.35$). We do note however that the ratio of balanced feedback to reward-focused strategies is smallest (three to one) for experiments where the dog gave an audible cry in response to punishment, which suggests that more participants chose a reward-focused strategy when the dog gave such a response than when it did not. This would be in line with our expectation, since the audible cry could lead human trainers to empathize with the learner, and so give fewer punishments during training.

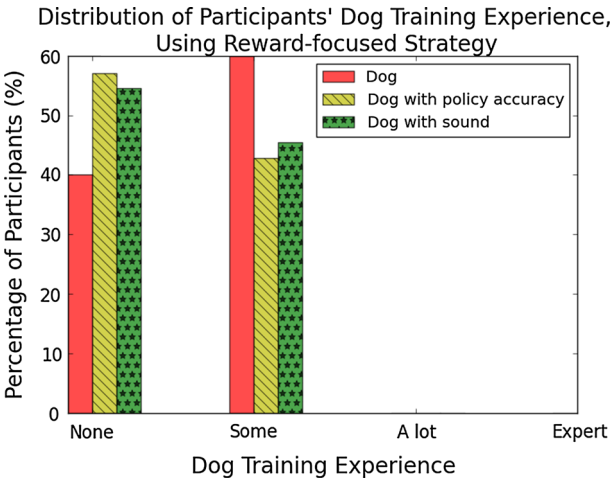

Fig. 5 The distribution of participants in AMT 6–8 who used a reward-focused strategy based on their experience with dog training, grouped by different training conditions

Figure 5 shows the distribution of participants who used a reward-focused strategy based on their experience with dog training. Fisher's exact test shows that the difference was not statistically significant ($p = 1$). We still find that more trainers without any dog training experience chose to use reward-focused strategies in the dog with policy accuracy and dog with audible response training conditions compared against the control condition. We can therefore conclude that different training factors did influence workers' choices of training strategies. That means, if the accuracy of the learned policy was shown to the trainer, or if the dog gave an audible cry after being punished, human trainers were more likely to use reward-focused.

6.4 Trainer mistakes

One of the main assumptions of our probabilistic model is that trainers can make mistakes when providing feedback (the ϵ parameter, discussed in Sect. 3.2). The results of both the volunteer studies and the Mechanical Turk studies demonstrate that trainer errors are common, and that any approach to learning from feedback must therefore be able to recover from such errors.

Note that since we cannot know if a user made a mistake for actions that did not receive feedback, we can only estimate ϵ from cases in which explicit feedback is provided. We estimated the average ϵ for participants in volunteer studies 1 and 2 combined to be 0.085 on a 0 to 1 probability scale. In AMT 1, where agents were represented as both dogs and as robots, the estimated average ϵ was 0.034.

The comments made by some of the participants suggest possible sources of error. One participant explained, "... i [sic] kept getting mixed up at first and hitting the wrong buttons...", suggesting that error could be reduced with a clearer interface design and more user practice. Another user commented, "At first it got frustrating because my timing was off on the reward and punishment. That doesn't help the dog and they become afraid and stay away because they are confused." Our model does not currently account for errors in the timing of feedback. This problem, however, may be mitigated by taking the weighted average of feedback over a longer time window, as was done in some related work [15].

7 Performance of SABL and I-SABL

In addition to exploring the distribution of trainer strategies, the two volunteer studies were used to evaluate the performance of the SABL and I-SABL algorithms. In addition to these studies, we also conducted experiments using SABL and I-SABL with simulated trainers that generated feedback according to our probabilistic model. The results in this section will show that learning the trainer's strategy, and using that knowledge to interpret the lack of feedback, can improve learning performance, at least when a large number of users follow reward-focused training strategies.

Though the results presented in Sect. 6 show that balanced feedback training strategies were the most common overall (except in volunteer study 2), there are a number of reasons to believe that algorithms which explicitly consider trainer strategy would be effective with users who do not use such a strategy as frequently. For one, while balanced strategies were the most common, we did observe a significant number of users following reward-focused strategies, including in the Mechanical Turk studies. As the simulated trainer experiments will show, SABL (assuming a reward-focused strategy) and I-SABL do not perform significantly worse than SABL (assuming a balanced feedback strategy) when the trainer is actually following a balanced feedback strategy. Therefore, we argue that SABL and I-SABL can improve performance for users following reward-focused strategies without significantly impacting performance for users following balanced feedback strategies. In addition, because of the way we classify strategies as balanced-feedback versus reward-focused or punishment-focused, it is still possible for a user following a balanced feedback strategy to have some bias towards providing implicit feedback for correct or incorrect actions (if for example they always give explicit feedback for incorrect actions, but sometimes fail to give feedback for correct actions), and so those users may still benefit from the SABL/I-SABL algorithms.

7.1 Reward based algorithms, M_{-0} and M_{+0}

To evaluate the SABL and I-SABL algorithms, we compare them against two algorithms (M_{-0} and M_{+0}) which are meant to be representative of algorithms from the literature which treat human feedback as being representative of numeric reward. M_{-0} and M_{+0} both map each feedback to a numeric reward value, $+1$ for positive feedback, and -1 for negative feedback. Both algorithms maintain a table containing the average reward value received for observation/action pair with a value of zero for any state action pair that has not yet been encountered during learning. Unlike SABL and I-SABL, M_{-0} and M_{+0} use the cumulative value of all feedback given during an episode. Both algorithms take the action which has the highest average reward of all actions for the current state. M_{-0} and M_{+0} differ in how they handle cases where no feedback is given. M_{-0} is designed to be most similar to the TAMER framework [17] and ignores episodes without feedback, making no changes to its value estimates in that case. M_{+0} is designed to be similar to the COBOT system [11] in how it handles episodes without feedback, treating no feedback as a reward value of zero. Therefore, with M_{+0}, value estimates for actions will return to zero after enough episodes with no feedback.

As M_{-0} ignores episodes without feedback, there is no way for it to interpret the lack of feedback using knowledge of the trainers strategy. M_{+0} could, however, be modified to learn from the lack of feedback, by assigning a positive or negative reward value to episodes without feedback, depending on the trainer's strategy. The M_{+0} algorithm would still behave differently from SABL, even if both assumed the same strategy. If, for example, both assumed a punishment-focused strategy (and the trainer actually followed such a strategy), then M_{+0} should assign a positive reward value (one less that the value assigned to explicit positive

feedback) to episodes without feedback. If, based on explicit feedback, both algorithms had identified the correct action for a state, then their subsequent actions for that state would yield little explicit feedback. After each episode without feedback, SABL would increase the estimated likelihood of the action being correct, while M_{+0} would move its reward estimate for the action closer to the reward value associated with the lack of feedback. This could have the effect of reducing that action's reward estimate, and actually cause the agent to eventually select a different action, which would be undesirable if the action is actually correct.

We do not, however, explore the possibility of using M_{+0} with strategy knowledge here, and to our knowledge a similar approach has not been described in the literature. As stated above, in these experiments we always assume that M_{+0} assigns a reward value of zero to episodes without feedback.

7.2 User studies

Volunteer studies 1 and 2, in addition to evaluating the types of strategies used by human trainers, also evaluate the performance (in terms of the time required to learn the target policy) of SABL and I-SABL, both against each other and against existing approaches. Volunteer study 1 compared SABL against M_{-0} and M_{+0}, and had 126 users, of which 71 completed training at least one learner. Volunteer study 2 compared I-SABL against SABL and had 43 users, of which 26 completed training at least one learner. In both of these studies, the base SABL learner assumed that the trainer followed a balanced feedback strategy, that is, $\mu^+ = \mu^-$.

Our performance measure was the average number of steps it took each agent to reach each of four criteria. Three of the criteria were when the learner's estimate of the policy was 50, 75, and 100 % correct. The fourth criterion was the number of steps before the user terminated the experiment. Results from the first user study show that learners using SABL tended to outperform those using M_{-0} and M_{+0}. Figure 6a shows the number of steps to reach each of the four criteria. The bars for SABL are lower than their counterparts for the other algorithms, showing that on average the SABL learner took fewer steps to reach the 75, 100 %, and the user termination criteria. Unpaired two sample t-tests show that the differences between the SABL learner and the M_{-0} and M_{+0} learners, for the 75, 100 % and termination criteria, were statistically significant ($p < 0.05$). In addition, a larger percentage of sessions using SABL reached 50, 75, and 100 % policy correctness than using M_{-0} or M_{+0}. Pearson's χ^2 tests show that the differences between the number of times the SABL learner and the M_{-0} and M_{+0} learners reached the 100 % criteria were statistically significant ($p < 0.01$), with the SABL, M_{-0} and M_{+0} learners reaching 100 % correctness 53, 17 and 19 % of the time respectively.

In the second study, we compared I-SABL against SABL using the same performance criteria to test whether inferring trainers' strategies improves learning performance. Figure 6b shows the number of steps for each algorithm to reach the criteria. Of interest are the very small (statistically insignificant) differences between SABL and I-SABL for the 50 and 75 % policy correctness criteria. The difference becomes much larger at the 100 % and user-selected termination criteria, where I-SABL reaches each criteria in significantly fewer steps. This is expected, as improvements in learning performance for I-SABL will be most pronounced when the agent has received enough feedback for some observations to infer the trainer's strategy. Unpaired t-tests show these performance differences are statistically significant, with $p = 0.01$ for the 100 % and $p < 0.05$ for the termination criteria. A larger percentage of sessions using I-SABL reached 50, 75, and 100 % policy correctness before termination than using SABL. Pearson's χ^2 tests show that the differences between the number of times the I-SABL learner and the SABL learner reached the 100 % criteria

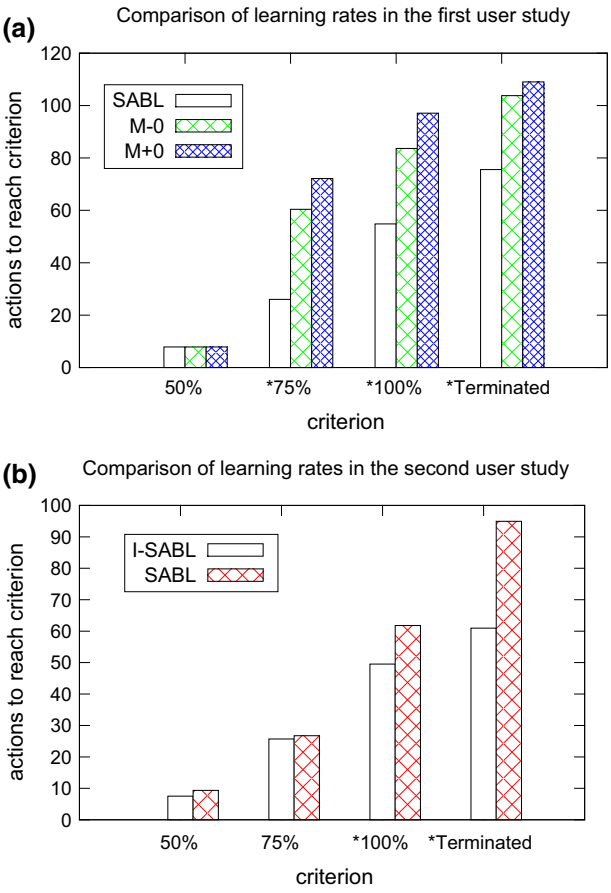

Fig. 6 Average number of episodes required to learn a policy that was correct for at least 50, 75, or 100 % of observations, and until the participants terminated the session. (*Asterisk* indicates that differences were statistically significant for that *column*) **a** First user study, comparing SABL, $M_{\_0}$ and M_{+0} **b** Second user study, comparing SABL and I-SABL

were significant ($p < 0.01$), with the I-SABL learner reaching 100 % policy correctness 50 % of the time, and the SABL learner reaching it 23 % of the time, respectively.

We note that SABL took more episodes on average to learn in volunteer study 2 than it did in volunteer study 1. We attribute this difference to the fact that users with dog training experience, who were much more common in study 2 than in study 1, were more likely to use a reward-focused training strategy. As the SABL algorithm assumed a balanced feedback strategy, it ignored episodes without feedback, and so performed more poorly under reward-focused strategies which provided fewer explicit feedbacks. This does however raise the question of how the M_{-0} and M_{+0} algorithms would have performed in this study. As users gave less explicit feedback in volunteer study 2, we suggest that M_{-0} and M_{+0} would have suffered a similar reduction in performance as SABL, since they too cannot directly consider trainer strategy (though M_{+0} does include episodes without feedback in its value estimate). As M_{-0} and M_{+0} were not evaluated in volunteer study 2, however, we cannot compare their performance to that of SABL in that study, nor can we directly compare their performance to the performance of I-SABL.

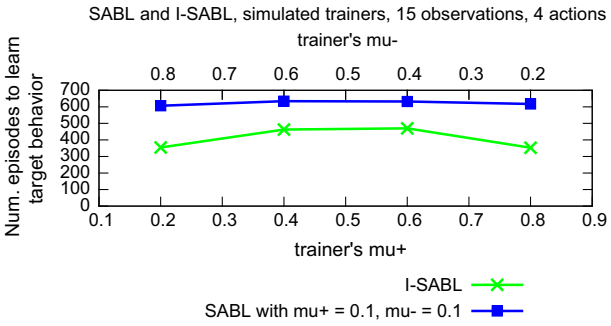

Fig. 7 Performance of I-SABL and SABL (μ^-=μ^+=0.1) with simulated trainers. The *bottom* x-axis is the trainer's μ^+, the *top* x-axis is μ^-, and the y-axis is the number of episodes to find the target policy. As the difference between μ^+ and μ^- grows, so too does the performance difference between SABL and I-SABL

7.3 Simulated trainer experiments

To help understand how strategy inference allows I-SABL to outperform SABL, we ran several experiments with simulated trainers in contextual bandit domains, comparing I-SABL against SABL (with SABL's μ^+=μ^-=0.1). The simulated trainer chose a target policy at random, and generated feedback using the same probabilistic model underlying SABL and I-SABL. We tested each learning agent on tasks consisting of 2, 5, 10, 15 and 20 observations and 2, 3, or 4 actions. These experiments were conducted for a range of pairs of μ^+ and μ^- values for the simulated trainer. Each μ parameter was varied, from 0.0 to 0.8, such that μ^-+μ^+≤1. The trainer's error rate was ϵ=0.2, matching SABL and I-SABL's assumed value. Learners in these studies took actions at random but kept an estimate of the most likely policy.

The results show that I-SABL is able to take advantage of information from episodes where no explicit feedback is given. Figure 7 shows two curves representing the number of steps it took the SABL and I-SABL agents to find the correct policy, for varying μ parameters. The difference in performance between I-SABL and SABL increases (in favor of I-SABL) as the trainer's μ parameters diverge from the balanced strategy that SABL assumes. In addition, I-SABL compares well to SABL even when the trainer follows a balanced strategy.

8 SABL in sequential domains

Results presented thus far show SABL and I-SABL in contextual bandit domains, where the agent can observe the world, but it's actions have no effect on the probability of subsequent states of the world. We can also apply these algorithms to sequential decision making domains. For efficiency, we limit the set of policies considered by SABL and I-SABL, by assuming that the trainer teaches an optimal policy for some set of conditions. In a grid world, for example, the trainer could teach the agent to reach some goal location.

We tested SABL and I-SABL for sequential domains in a 15 by 15 grid world with a simulated trainer. The algorithms considered 48 possible goal states, as well as two special kinds of "obstacles"—states the agent could move in to or out of but may have needed to avoid—depending on the particular obstacle condition. There were four different obstacle conditions (no obstacles, avoid type one, avoid type two, avoid both types), resulting in $48 \times 4 = 192$ possible optimal policies. Figure 8 shows the grid world used. Note that the learners did not actually receive any information about the goal or obstacles from the

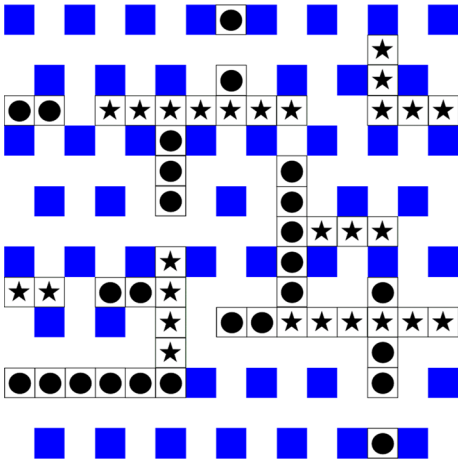

Fig. 8 The sequential domain. *Blue squares* represent possible goal states, *circles* represent obstacles of type one and *stars* represent obstacles of type two (Color figure online)

environment, and so could only learn the correct behavior based on trainer feedback. In the sequential case, SABL and I-SABL simply assumed that the trainer's target policy was one of the 192 possible optimal policies.

In this case SABL and I-SABL only considered a small, finite set of possible μ parameter combinations, representing balanced, reward-focused, and punishment-focused trainer strategies. Additionally, to leverage this simplification rather than use EM on the entire feedback history at each step, we adapted I-SABL to update its prior belief in each strategy and policy to the posterior probability distribution given by the most recent feedback and the current distribution over trainer strategies. Trainer strategies were defined by $\{\mu^+, \mu^-\} = \{0.1, 0.1\}$ for the balanced feedback strategy, $\{\mu^+, \mu^-\} = \{0.1, 0.9\}$ for the reward-focused strategy, and $\{\mu^+, \mu^-\} = \{0.9, 0.1\}$ for the punishment-focused strategy. We did not consider the inactive strategy, as it was uncommon in the user study. For all strategies, $\epsilon = 0.05$, which is lower than the assumed error rate used in our user studies, but is closer to the actual error rate observed in those studies. We should note that the values of the μ parameters were chosen to represent strategies that strongly prefer explicit feedback in all cases, or extremely reward or punishment focused strategies. This was done both to highlight differences between the learning algorithms, and because such strong preferences were observed in the user studies.

Table 6 summarizes the results for all algorithm and trainer strategy pairs. For all simulated trainers, I-SABL and SABL using the correct feedback strategy identified the intended policy the fastest, again demonstrating that I-SABL does not suffer significantly from initial uncertainty about the trainer strategy. When the simulated trainer used a balanced strategy, SABL using incorrect strategy assumptions performed worse, but not significantly worse, likely due the fact that the simulated trainer almost always gave explicit feedback. Regardless of their strategy assumption, SABL learners always interpret explicit feedback in the same way. However, when the trainer does not employ a balanced strategy, incorrect SABL assumptions will be more problematic. If SABL assumes a balanced feedback strategy while the trainer follows a reward-focused strategy, the policy can be learned, but more steps are needed to do so because many steps receive no explicit feedback and so are ignored. If SABL assumes the opposite strategy (e.g., assuming punishment-focused when it is actually reward-focused), then the agent may never learn the correct policy. Assuming the opposite strategy likely performs so poorly because it misinterprets what a lack of feedback means. If SABL assumes a punishment-focused strategy when it's actually a reward-focused strategy, it will interpret the lack of feedback when its action is incorrect as evidence that it is correct.

Table 6 For all algorithm and simulated trainer pairs tested, the average number of steps before the agent correctly identified the intended policy as the most likely, and the average number of explicit feedbacks that were provided before the intended task was identified as the most likely

Trainer's strategy	Learning algorithm	Identify policy	95 % Confidence interval	# Explicit feedbacks	95 % Confidence interval
Balanced feedback	I-SABL	44.4	±11.7	39.1	±10.4
	SABL—balanced feedback	46.7	±9.3	40.5	±8.1
	SABL—reward-focused	67.3	±21.1	60.0	±19.3
	SABL—punishment-focused	65.6	±20.6	58.1	±18.5
Reward-focused	I-SABL	68.7	±20.5	54.1	±17.7
	SABL—balanced feedback	152.8	±27.9	71.4	±18.2
	SABL—reward-focused	65	±23.8	50.8	±20.4
	SABL—punishment-focused	N/A	N/A	N/A	N/A
Punishment-focused	I-SABL	76.2	±25.4	14.8	±3.9
	SABL—balanced feedback	190.9	±27.3	37.4	±4.5
	SABL—reward-focused	N/A	N/A	N/A	N/A
	SABL—punishment-focused	51.3	±17.9	11.1	±2.8

"N/A" indicates that the algorithm was unable to learn the correct policy in the majority of training runs

In these results, it is also interesting to note how few explicit feedbacks are required for I-SABL and SABL (with a correct strategy assumption) to learn when the trainer follows a punishment-focused strategy. As it learns, more of the agent's actions are correct, resulting in less explicit feedback; since I-SABL (and SABL assuming a punishment-focused strategy) correctly interpret this lack of explicit feedback as positive, it does not hinder learning. In these experiments the actual and assumed error rates ϵ were relatively low at 0.05. While a higher error rate would certainly have meant that each algorithm would take longer to learn (more mistakes would need to be corrected, requiring more time and more feedback), it is unclear how the error rate would affect the relative performance of SABL and I-SABL. While more erroneous feedback could reduce the quality of I-SABL's estimate of the trainer's strategy, knowledge of trainer strategy could also allow I-SABL to more quickly recover from mistakes.

9 Future directions

This work has only considered cases where the state and action spaces of the task domain are discrete, such that the trainer's desired behavior can be represented simply as a list of

each state with its correct action. In many real world domains with large or continuous state spaces, it may not be possible for the target policy to be represented in such an explicit way, or it may be difficult for the agent to demonstrate every state action pair possible. In such cases, it may be necessary for the policy to be represented by some parametric function approximator that can handle continuous state features, and that allows for some degree of action generalization between states. While this work has considered only discrete policies, we suggest that it would be possible to learn such continuous, parametric policy representations under the SABL/I-SABL framework. For example, policies in continuous spaces can be represented as multilayer perceptrons, such that learning the target policy involves finding weight and bias parameters of the network that minimize or maximize some objective function based on examples of that policy [26]. The policy likelihood function that SABL attempts to maximize could be used as an objective function for training such a network via backpropagation. Similarly, the expected likelihood function maximized in each iteration of I-SABL's expectation-maximization algorithm could be used as the objective function for training a network, such that I-SABL would attempt to find a maximum likelihood estimate of the network parameters, rather than of the policy itself.

Real world domains are also often sequential in nature, such that information about the dynamics of the environment can be used to help identify the target policy. While this work considers learning from feedback in sequential domains, by combining our learning framework with algorithms for inverse reinforcement learning, we note that the applicability of such IRL algorithms to many real world domains often depends on their ability to compute good, if not optimal, policies in those domains. Much of the work in inverse reinforcement learning has been restricted to cases where the state space is discrete, where algorithms such as value iteration, policy iteration, or linear programming can be used to compute optimal policies [21,22]. Even when reward functions can be modeled based on continuous features, the underlying domain may still need to be discretized during the planning phase [1]. Similarly, our maximum likelihood IRL approach relies on the existence of an efficient planning algorithm for the task domain, which in our experiments is discrete. Future work might consider how SABL and I-SABL could be combined with IRL algorithms that are better suited to continuous domains [28]. Extensions of our learning algorithms to continuous domains, however, are beyond the scope of this work and are reserved for future studies.

As discussed earlier in this work, there are many aspects of trainer strategies which are not accounted for in our model. For one, we have no explicit model of how a trainer's strategy can change over time, or what such changes in and of themselves are meant to convey to the learning agent. In the survey section of the user studies, some participants discussed how they changed strategy over time. One participant explained, "I rewarded for every time the dog faced the side the rat came from. I ignored incorrect responses. As the dog became better and better at heading to the rat side, I implemented random reward for correct responses and continued to ignore incorrect responses."

Another participant similarly believed that rewards would be more useful early, while punishments would be better later on, "I allowed for mistakes in the beginning because the dog was 'new' to the task. I rewarded any successful attempts with increasing amounts of reward up to 5 rewards per successful guess. Then, I applied punishments to correct mistakes since the dog had 'learned' what the correct action was." In both of these cases, the meaning of the lack of feedback changes over time, going from implicitly negative to neutral or implicitly positive. Our model could be extended to account for this change, so as to correctly interpret the lack of feedback throughout the training session.

Future work could also consider how to handle various types of trainer error. Specifically, we currently have no way to account for delay in the feedback given that might cause it to

be associated with the wrong action. There is also the possibility that feedback has different interpretations for different parts of a task, that is, for different subsets of states and actions. Developing models of feedback that account for these more complex and variable training strategies would allow us to build learning agents that could better adapt to the user's strategy.

10 Conclusion

This work has demonstrated that, when considering the problem of learning from trainer feedback, significant improvements in learning performance can be achieved by applying relatively simple models of trainer strategy. We have also shown that there is significant variability in trainer strategy that can be exploited. We can draw two main conclusions from the empirical results presented in this work.

1. Human trainers use a variety of strategies when training virtual agents, and may change strategy while training. These results suggest that trainers' choices of strategies can be influenced by the trainers' backgrounds, and, at least to some degree, by the nature of the training task itself. The different strategies followed by trainers necessitate different interpretations of cases where no feedback is given, with the lack of feedback indicating a correct action under some strategies, and an incorrect action under others.
2. We have presented two probabilistic inference algorithms, SABL and I-SABL, which explicitly take trainer strategy into account. We have demonstrated with real users that these algorithms can learn behaviors with fewer trainer feedbacks than algorithms based on a numerical interpretation of feedback, and we have demonstrated that I-SABL in particular is able to adapt to trainers' strategies online, and so is able to learn more efficiently by correctly interpreting what the lack of feedback means.

Based on these results we argue that incorporating similar, though potentially more general models of human feedback into systems designed to learn and reason in more complicated, real-world environments will allow us to build robots and virtual agents that can learn useful behaviors efficiently, and in a way that is intuitive for the average user who has little or no backgrounds in programming or artificial intelligence.

Acknowledgments This work was supported in part by Grants IIS-1149917 and IIS-1319412 from the National Science Foundation.

References

1. Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*.
2. Argall, B., Browning, B., & Veloso, M. (2007). Learning by demonstration with critique from a human teacher. In *Proceedings of the ACM/IEEE International Conference on human-robot interaction* (pp. 57–64).
3. Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.
4. Atkeson, C. G., & Schaal, S. (1997). Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on machine learning*.
5. Cakmak, M., & Lopes, M. (2012). Algorithmic and human teaching of sequential decision tasks. In *Proceedings of the Twenty-sixth AAAI Conference on artificial intelligence* (pp. 1536–1542).
6. Chernova, S., & Veloso, M. (2007). Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Conference on autonomous agents and multiagent systems*.

7. Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38.
8. Griffith, S., Subramanian, K., Scholz, J., Isbell, C., & Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. *Advances in Neural Information Processing Systems*, 26.
9. Heer, J., Good, N. S., Ramirez, A., Davis, M., & Mankoff, J. (2004). Presiding over accidents: system direction of human action. In *Proceedings of the SIGCHI Conference on human factors in computing systems* (pp. 463–470).
10. Hiby, E., Rooney, N., & Bradshaw, J. (2004). Dog training methods: Their use, effectiveness and interaction with behaviour and welfare. *Animal Welfare*, *13*(1), 63–70.
11. Isbell, C., Shelton, C., Kearns, M., Singh, S., & Stone, P. (2001). A social reinforcement learning agent. In *Proceedings of the Fifth International Conference on Autonomous Agents* (pp. 377–384).
12. Judah, K., Fern, A., Tadepalli, P., & Goetschalckx, R. (2014). Imitation learning with demonstrations and shaping rewards. In *Proceedings of the twenty-eighth AAAI Conference on Artificial Intelligence* (pp. 1890–1896).
13. Judah, K., Roy, S., Fern, A., & Dietterich, T. G. (2010). Reinforcement learning via practice and critique advice. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (pp. 481–486).
14. Khan, F., Zhu, X., & Mutlu, B. (2011). How do humans teach: On curriculum learning and teaching dimension. In *Proceedings of the Twenty-fifth Annual Conference on Neural Information Processing Systems* (pp. 1449–1457).
15. Knox, W., Stone, P., & Breazeal, C. (2013). Training a robot via human feedback: A case study. *Social Robotics, Lecture Notes in Computer Science*, *8239*, 460–470.
16. 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*, *4*(4), 409–421.
17. Knox, W. B., & Stone, P. (2009). Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the Fifth International Conference on Knowledge Capture* (pp. 9–16).
18. Knox, W. B., Taylor, M. E., & Stone, P. (2011). Understanding human teaching modalities in reinforcement learning environments: A preliminary report. In *Proceedings of the Workshop on Agents Learning Interactively from Human Teachers (at IJCAI-11)*.
19. Li, G., Hung, H., Whiteson, S., & Knox, W. B. (2013). Using informative behavior to increase engagement in the TAMER framework. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems* (pp. 909–916).
20. Lu, T., Pal, D., Pal, M. (2010). Contextual multi-armed bandits. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*.
21. Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 663–670).
22. Ramachandran, D. (2007). Bayesian inverse reinforcement learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*.
23. Skinner, B. F. (1938). *The behavior of organisms: An experimental analysis*. New York: Appleton-Century.
24. Skinner, B. F. (1953). *Science and human behavior*. New York: Macmillan.
25. Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge: MIT Press.
26. Tesauro, G. (1990). Neurogammon: A neural-network backgammon program. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 33–39). IEEE.
27. Thomaz, A. L., Breazeal, C. (2006). Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the Twenty-first AAAI Conference on Artificial Intelligence* (pp. 1000–1005).
28. Ziebart, B. D., Maas, A., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-third AAAI conference on artificial intelligence*.