Imitation Learning with Demonstrations and Shaping Rewards

Kshitij Judah and **Alan Fern** and **Prasad Tadepalli** and **Robby Goetschalckx**
School of EECS, Oregon State University, Corvallis, OR, USA, 97331
{judahk,afern,tadepall,goetschr}@eecs.oregonstate.edu

Abstract

Imitation Learning (IL) is a popular approach for teaching behavior policies to agents by demonstrating the desired target policy. While the approach has lead to many successes, IL often requires a large set of demonstrations to achieve robust learning, which can be expensive for the teacher. In this paper, we consider a novel approach to improve the learning efficiency of IL by providing a shaping reward function in addition to the usual demonstrations. Shaping rewards are numeric functions of states (and possibly actions) that are generally easily specified, and capture general principles of desired behavior, without necessarily completely specifying the behavior. Shaping rewards have been used extensively in reinforcement learning, but have been seldom considered for IL, though they are often easy to specify. Our main contribution is to propose an IL approach that learns from both shaping rewards and demonstrations. We demonstrate the effectiveness of the approach across several IL problems, even when the shaping reward is not fully consistent with the demonstrations.

1 Introduction

We consider teaching policies to agents to perform specific tasks. One framework for doing this is reinforcement learning (RL), where the learner is given a reward function, and learns a policy to maximize the reward. However, two drawbacks of this approach are that RL can suffer from poor scalability, and it can be difficult to design a reward function that leads RL to a specific desired behavior.

Imitation Learning (IL) is an alternative to RL that addressed both the scalability and reward-design issues. Rather than design a reward function, the teacher directly demonstrates the desired behavior. The learner then attempts to learn a policy that can mimic the expert demonstrations, and generalize to new situations. While the IL approach has often been successful, robust learning can often require a large number of demonstrations, which can be expensive to collect. The main contribution of this paper is a new IL approach that leverages shaping rewards in order to learn effectively from fewer demonstrations.

Shaping rewards are commonly used to speed up RL (Ng, Harada, and Russell 1999; Dorigo and Colombetti 1994; Mataric 1994; Randlov and Alstrom 1998), and can be viewed as rules-of-thumb that provide short-term information about the goodness/badness of states and actions. Thus, shaping rewards are often easier to learn from than the true reward function, which is often much longer term and infrequent. However, shaping rewards are generally a simplification of the true problem, and maximizing the shaping reward alone will not usually produce optimal policies under the true reward. Thus, in practice, shaping rewards are combined additively with the true reward during learning. Convergence to optimal policies can be guaranteed under certain restrictions (Ng, Harada, and Russell 1999).

Despite the wide use of reward shaping in RL, little prior work has consider leverage shaping rewards in an IL setting. As a motivating example, consider the IL benchmark of teaching car driving policies (Pomerleau 1989; Chernova and Veloso 2009; Ross and Bagnell 2010). A teacher can easily use shaping rewards to specify certain general rules-of-thumb about good/bad driving behaviors (e.g. avoiding collisions and staying on the road). Demonstrations might then be used to specify more detailed and specialized aspects of the desired behavior (e.g. lane and speed preferences). Ideally, if the shaping reward is mostly consistent with the target policy, we would hope an IL agent could learn from fewer demonstrations by biasing learning toward policies with higher shaping reward.

In this paper, we give a new IL algorithm called *Shaped IL (SHAIL)* for combining a shaping rewards and demonstration data. The key idea is to formulate a constrained optimization problem over policies, where the goal is to maximize shaping reward, subject to the constraint that the policy tends to agree with the demonstrations. This formulation ensures that we meet our learning goal of imitating the teacher, even when the shaping reward is not fully consistent with the teacher. At the same time, when the shaping reward is informative, by maximizing the shaping reward, the learner can be expected to require fewer demonstrations to identify the target policy. SHAIL addresses this problem via Lagrangian relaxation, which reduces the IL problem to a sequence of unconstrained problems. Our experimental results across a variety of IL problems show that SHAIL is able to use simple shaping rewards to significantly reduce the number of required demonstrations, and is robust to shaping rewards that are sometimes inconsistent with the teacher.

2 Related Work

The *value-based prior (VBP)* approach for IL (Syed and Schapire 2007) is a closely related approach where a reward function is used to define a prior over policies, such that policies with higher values have higher prior probabilities. Given demonstration data, learning involves finding a policy that maximizes the posterior. A major drawback of the approach is that the algorithm is designed for flat state spaces, and does not scale to large problems such as the domains used in our experiments.

IL is related to inverse RL (IRL) (Ng and Russell 2000), where a teacher's reward function is estimated based on demonstrations and used to compute a policy. The *Bayesian IRL* approach (Ramachandran and Amir 2007) assumes a prior over reward functions, and attempts to find a reward function that maximizes the posterior given the demonstrations. The scalability of the approach is unclear, and has so far been applied only to small problems (up to 1000 states).

Prior work (Judah et al. 2010) studied how to incorporate an expert advice on actions into an RL agent. Similar to our work, the approach learns by combining a reward function with the teacher's direct advice about actions. However, their objective is to maximize reward, even if that means ignoring the teacher's action advice. This is the opposite of our IL objective, where the goal is to agree with the demonstrations and rewards are viewed as possibly fallible advice.

Other prior work allows a teacher to train an agent online by supplying numeric reward signals about an agent's actions (Knox and Stone 2009; 2010; 2012; Thomaz and Breazeal 2008), sometimes in combination with the true environmental reward. This mode of teaching is less direct and less informative than IL, though potentially more widely applicable when it is not feasible to provide demonstrations. However, the goal in these approaches is again to converge to an optimal policy with respect to the environmental reward, using the teacher reward as a way to accelerate learning. Hence these approaches are in contrast to our IL work. Similar comments hold for other work on advice in RL (Rosenstein and Barto 2004; Maclin et al. 2005).

3 Problem Setup

Imitation Learning. We consider the framework of Markov decision processes with no rewards (denoted MDP\R) and finite time horizon H. An MDP\R is a tuple $\langle S, A, T, I\rangle$, where S is a set of states, A is a finite set of actions, $T(s, a, s')$ denotes the probability of transitioning to state s' upon taking action a in state s, and I is the initial state distribution. A stationary policy π gives a probability distribution $\pi(s)$ over actions for each state s, where $\pi(s, a)$ denotes the probability of selecting action a in s. To handle large state spaces, we use parametric log-linear policies of the form $\pi_\theta(s, a) \propto \exp(f(s, a) \cdot \theta)$, where $f(s, a)$ is a feature vector of state-action pairs and θ is the weight vector.

In IL, the learner is provided with a set of demonstrations $\mathcal{D} = \{\mathcal{T}^i\}_{i=1}^N$ of a target policy π^*. Each demonstration $\mathcal{T}^i = (s_t^i, a_t^i)_{t=1}^H$ is a sequence of H state-action pairs starting at a state drawn from I with actions selected according to π^*. The goal is to learn a policy π that can accurately mimic π^*. A typical approach for learning such a policy is to supply the state-action pairs from $\mathcal{D}$ to a supervised classification learner and then let π be the learned classifier. This approach is justified by the fact that if π can accurately predict the actions of π^* with respect to the induced state distribution of π^*, then the long-term value of π will be close to that of π^* for any reward function (Ross and Bagnell 2010). In this paper, since we utilize probabilistic policies, we will measure accuracy of a policy π_θ in terms of its log-likelihood $L(\theta, \mathcal{D}) = \sum_{(s,a)\in\mathcal{D}} n_{s,a} \log \pi_\theta(s, a)$, where $n_{s,a}$ is the frequency of the state-action pair (s, a) in $\mathcal{D}$.

IL with Shaping Rewards. In this work, we are interested in using shaping rewards for accelerating IL. We assume that in addition to the set $\mathcal{D}$, the learner is also supplied with a shaping reward function R_s and a simulator M of the MDP\R. Here R_s is a function from states and actions to numeric rewards that is intended to reflect prior knowledge about the goodness/badness of being in different states and executing different actions from the perspective of the teacher's intended target behavior. We assume that the learner does not know or learn the transition model T. Rather, it has access to a simulator M that allows it to sample trajectories from the MDP in order to evaluate different policies. The assumption of having access to M is not too unrealistic because many domains of interest are either simulated (e.g. agents in video games) or have sophisticated simulators available (e.g. computer network simulators). Given M and R_s, we can define the *H-horizon shaping value* $V_{R_s}(\pi)$ of a policy π as the expected total shaping reward of trajectories that start in $s_1 \sim I$ and then execute π for H steps.

Note that access to R_s does not imply that π^* can be learned using pure RL. This is because shaping rewards seldom completely define the target policy as there will often be other policies that maximize the shaping value. Rather, the role of R_s is to encode extra information that we may have about π^*. For example, consider a situation where an agent will be solving multiple IL problems within a particular domain (e.g. car driving). R_s in this case can capture properties that are likely to be common across different target policies in the domain (e.g. collision avoidance), while demonstrations distinguish individual policies (e.g. lane preferences).

Given demonstration data $\mathcal{D}$ and R_s, we want to learn a policy π that mimics the expert policy π^*, ideally, more efficiently than without R_s. Importantly, even if R_s is inconsistent with π^*, we still want to converge to π^*, ideally, nearly as efficiently as if R_s were not provided. That is, we are willing to ignore R_s when it conflicts with demonstrations. Given that goal, we seek a policy that maximizes the shaping value subject to remaining accurate with respect to the demonstration data, which is formalized as follows:

$$\hat{\theta} = \operatorname*{argmax}_\theta V_{R_s}(\pi_\theta) \text{ s.t. } L(\theta, \mathcal{D}) \geq \bar{L}, \qquad (1)$$

where $\bar{L}$ is a lower bound allowed on $L(\theta, \mathcal{D})$ that specifies the accuracy constraint. In our experiments, we automatically set $\bar{L}$ based on the log-likelihood L^* of a pure supervised learner, which ignores the shaping reward, and can be easily computed via convex optimization. We set $\bar{L} = L^* - \delta$, where $L^* = \max_\theta L(\theta, \mathcal{D})$ and δ is the slack allowed on the optimal log-likelihood value L^*. In all of our experiments, we set $\delta = 0.01 \cdot |\mathcal{D}|$, which in our experience

is a robust setting that results in minimal loss in training accuracy.

Theoretical Analysis. Ideally, we would like to guarantee that the performance of the learned policy should never be much worse than that of a pure IL learner that ignores the shaping reward, even when the shaping reward is misleading. At the same time, we would like to characterize how the efficiency of learning can be improved for high-quality shaping rewards. Here we address these questions by proving a PAC-style bound (Valiant 1984) related to learning policies according to optimization problem (1), which illustrates the dependence on the shaping reward.

We consider a simple learning setting where the learner considers a finite set of deterministic policies Π as its hypothesis space. It is further assumed that the teacher policy π^* is in Π. We first describe the PAC IL framework from prior work (Khardon 1999), which we extend to incorporate shaping rewards. Let D^{π^*} be a distribution over trajectories of π^* and define the trajectory error of a policy π, denoted as $e(\pi)$, to be the probability that π disagrees with π^* on any state of a random trajectory drawn from D^{π^*}. It is easy to show that if $e(\pi)$ is small then the value of π will be close to the value of π^* for any reward function. We consider bounding the performance of a consistent learner, which is provided a training set of $\mathcal{D}$ of N independent trajectories from D^{π^*}, and returns a policy $\pi \in \Pi$ that is consistent with each trajectory, which is always possible under our assumptions. Khardon showed that for any π returned by a consistent learner, with probability at least $(1-\delta)$, we have that $e(\pi) \leq \frac{1}{N}\ln(\frac{|\Pi|}{\delta})$ (Khardon 1999). This shows that the trajectory error decreases at a linear rate in the number of training trajectories and that there is a worst case logarithmic dependence on the size of the hypothesis space.

We now adapt this framework toward optimization problem (1), where the learner outputs a policy π that is both consistent with $\mathcal{D}$ (satisfying the loss constraint of (1)) and maximizes V_{R_s} among all consistent policies. We will call such a learner a *shaped consistent learner*. The impact of the shaping reward is to effectively reduce the space of viable hypotheses. We characterize this by defining $\Pi^{\pi^*}_{R_s}$ to be the set of hypotheses that are ranked equal to or higher than π^* according to V_{R_s}. $\Pi^{\pi^*}_{R_s}$ is well defined for a given target policy π^* and shaping reward function R_s. The following result gives an upper bound on the error of π.

Theorem 1. *Let L be a shaped consistent learner and assume that $\pi^* \in \Pi$. If L returns π given a set of N training trajectories drawn from D^{π^*}, then with probability at least $(1-\delta)$, we have $e(\pi) \leq \frac{1}{N}\ln(\frac{|\Pi^{\pi^*}_{R_s}|}{\delta})$.*

Proof (sketch). Similar to the classic Blumer bound proof (Blumer et al. 1987), the main idea is to bound the probability that any hypothesis in $\Pi^{\pi^*}_{R_s}$ whose error is more than some constant $\epsilon > 0$ is consistent with the demonstrations, which is at most $(1-\epsilon)^N$. Since $\pi^* \in \Pi$, shaped consistent learning only considers outputting policies in $\Pi^{\pi^*}_{R_s}$. Thus, via the union bound, the probability that at least one of those policies is consistent is at most $|\Pi^{\pi^*}_{R_s}|(1-\epsilon)^N \leq |\Pi^{\pi^*}_{R_s}|e^{-\epsilon N}$. The theorem is obtained by bounding this quantity by δ. □

Hence, we see that the worst case error bound is similar to that of a consistent learner, but is reduced by an amount $\frac{1}{N}\ln(|\Pi^{\pi^*}_{R_s}|/|\Pi|)$. This bound captures some of the intuitive properties of the shaped IL framework. First, note that since $\Pi^{\pi^*}_{R_s} \subseteq \Pi$, the worst-case PAC bound will never get worse with shaping. This holds even for completely uninformative shaping rewards (e.g. $R_s = 0$) or even a malicious shaping reward that ranks π^* as the worst. Second, in the case with an ideal shaping reward R_s such that π^* is the only maximizing policy, we get $|\Pi^{\pi^*}_{R_s}| = 1$, indicating that the learner will be able to learn with no training examples (i.e. the problem is equivalent to RL). In between these extremes, we see that according to the bound, the quality of the shaping reward can be measured according to the number of policies that are ranked below π^*, which agrees with intuition.

Above we assumed that problem (1) can be solved exactly, which will often not be feasible in practice. A relaxation of the above considers a learner that may not exactly maximize V_{R_s}, but still outputs a consistent policy. In that case, if $|\Pi^{\pi^*}_{R_s}|$ is redefined to account for imperfect optimization of V_{R_s}, then the above result still holds, and in the worst case $|\Pi^{\pi^*}_{R_s}| = |\Pi|$, guaranteeing that in a worst-case PAC sense we do not perform worse than the pure IL, even with an imperfect optimizer.

4 Solution Approach

Optimization problem (1) is non-convex due to the $V_{R_s}(\theta)$ term. Thus, we focus on finding locally optimal solutions that work well in practice. In particular, we attempt to ensure that our returned solutions satisfy the accuracy constraints, while maximizing the objective as much as possible. As discussed above, as long as the constraints are satisfied, we can be confident that in the worst-case we will perform similarly to pure IL. Below we describe the *Shaped IL (SHAIL)* algorithm, first introducing the overall Lagrangian-relaxation approach to arrive at a sequence of unconstrained problems, and then giving an approach for the unconstrained problems.

Lagrangian Formulation. Lagrangian relaxation (LR) (Ahuja, Magnanti, and Orlin 1993) solves a constrained optimization problem via a series of unconstrained problems by removing the constraints, and adding them to the objective function as penalty terms weighted by Lagrange multipliers. This is useful when the unconstrained problems are easier to directly address compared to the constrained problem. For our constrained problem (1) (the primal problem), LR will relax the constraint by moving it to the objective function weighted by a Lagrange multiplier $\lambda \geq 0$. This yields the following unconstrained problem $P(\lambda)$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, V_{R_s}(\theta) + \lambda(L(\theta, \mathcal{D}) - \bar{L}), \qquad (2)$$

where λ controls how strongly we penalize solutions that violate the inequality constraint.

Importantly, λ can always be set large enough so that the resulting solution $\hat{\theta}$ is a feasible primal solution. This is due to our selection of $\bar{L}$, which is based on the best loss $L(\theta, \mathcal{D})$ achievable when ignoring the shaping reward. Thus, we know that there is always a feasible value of θ when λ

Algorithm 1 Pseudocode for Shaped IL (SHAIL)

Input: $\mathcal{D}$, R_s, parametric policy representation Θ, $\bar{L}$
Output: $(\hat{\theta}, V_{R_s}(\hat{\theta}), L(\hat{\theta}, \mathcal{D}))$, solution to problem (1)

$\lambda \leftarrow 1$
loop
  $\hat{\theta} \leftarrow \texttt{optimize}(\mathcal{D}, R_s, \lambda)$
  if $L(\hat{\theta}, \mathcal{D}) \geq \bar{L}$ **then** ▷ $\hat{\theta}$ is feasible
    $\lambda^u \leftarrow \lambda$, $\lambda^l \leftarrow \lambda^u/2$
    break
  $\lambda \leftarrow$ next higher λ
$\theta^* \leftarrow \hat{\theta}, V^* \leftarrow -\infty, L^* \leftarrow -\infty$
repeat
  $\lambda \leftarrow (\lambda^l + \lambda^u)/2$
  $\hat{\theta} \leftarrow \texttt{optimize}(\mathcal{D}, R_s, \lambda)$
  if $L(\hat{\theta}, \mathcal{D}) \geq \bar{L}$ **then** ▷ $\hat{\theta}$ is feasible
    $\lambda^u \leftarrow \lambda$
    if $V_{R_s}(\hat{\theta}) > V^*$ **then**
      $V^* \leftarrow V_{R_s}(\hat{\theta}), \theta^* \leftarrow \hat{\theta}, L^* \leftarrow L(\hat{\theta}, \mathcal{D})$
  else
    $\lambda^l \leftarrow \lambda$
until search is done
return (θ^*, V^*, L^*)

Algorithm 2 Optimizer for $V_{R_s}(\theta) + \lambda L(\theta, \mathcal{D})$

1: **procedure** OPTIMIZE($\mathcal{D}$, R_s, λ)
2: Set learning rate α, discount factor γ
3: $\theta^* \leftarrow \theta^0$, $J^* \leftarrow -\infty$
4: **for** some # of random restarts **do**
5: $\theta \leftarrow \theta^{rand}$
6: **for** some # of iterations **do**
7: ▷ run an episode in the MDP
8: Initialize eligibility trace z
9: Initialize MDP to a random state
10: **for** $t = 1$ to H **do**
11: $s \leftarrow \texttt{getCurrentState}()$
12: $a \leftarrow \texttt{takeAction}(\theta)$
13: $r \leftarrow R_s(s, a)$
14: $z \leftarrow \gamma * z + \nabla(\log(\pi_\theta(s, a)))$
15: $\theta \leftarrow \theta + \alpha * r * z$
16: ▷ run SGA
17: **for** each (s, a) in $\mathcal{D}$ **do**
18: $g \leftarrow \nabla(L(\theta, (s, a)))$
19: $\theta \leftarrow \theta + \alpha * \lambda * g$
20: **if** $V_{R_s}(\theta) + \lambda L(\theta, \mathcal{D}) > J^*$ **then**
21: $J^* \leftarrow V_{R_s}(\theta) + \lambda L(\theta, \mathcal{D})$
22: $\theta^* \leftarrow \theta$
23: **return** θ^*

is large enough, because the second term in (2) dominates. Here we assume a black box optimizer for (approximately) solving $P(\lambda)$, which we describe in the next section.

If we let $J(\lambda)$ denote the optimal objective value of $P(\lambda)$, it is easy to show that for any fixed value of λ, $J(\lambda)$ is an upper bound to the optimal value of the primal problem (1). Therefore, we can approximate the optimal value of the primal problem (and the θ at which the optimal occurs) by minimizing this upper bound. This minimization is captured by the following optimization problem (the dual problem):

$$\min J(\lambda) \text{ s.t. } \lambda \geq 0.$$

Because our Lagrangian dual problem optimizes over just one variable, SHAIL can perform a line search over different values of λ, using our black box optimizer for $P(\lambda)$ and find the value that minimizes $J(\lambda)$. As is typical in LR, the minimizing λ is not guaranteed to result in a feasible θ for the primal problem. Thus, during the line search SHAIL keeps track of the best feasible solution uncovered with respect to V_{R_s} and returns that solution.

Algorithm 1 gives pseudocode for SHAIL's line search component, where $\texttt{optimize}(\mathcal{D}, R_s, \lambda)$ denotes the black box optimizer for $P(\lambda)$. The line search starts with a small value for λ and increases it until finding a multiplier value λ^u that results in a feasible primal. Then a halving line search is conducted over λ values, using λ^u as an upper bound, to find the best minimizer of $J(\lambda)$.

Optimization of $P(\lambda)$. We now consider the procedure $\texttt{optimize}(\mathcal{D}, R_s, \lambda)$, which optimizes $V_{R_s}(\theta) + \lambda L(\theta, \mathcal{D})$. Since this objective is non-convex with respect to θ due to the V_{R_s} term, we follow a gradient-based optimization approach with random restarts to combat local optima. The best solution across restarts is returned as the final answer. Specifically, we use stochastic gradient ascent (SGA), which performs a sequence of many cheap gradient-based parameter updates, each based on an individual term of the objective. This process continues until a stopping condition is met. In our case, the objective consists of the $V_{R_s}(\theta)$ term, and the $N = |\mathcal{D}|$ components of the $L(\theta, \mathcal{D})$ term. Further, $V_{R_s}(\theta)$ is an expectation over random outcome sequences, and hence can be implicitly decomposed into one term per possible outcome sequence. Each iteration of our algorithm performs a gradient update for a randomly sampled component of the $V_{R_s}(\theta)$ term (i.e. a trajectory in the MDP) followed by updates for the individual components of $L(\theta, \mathcal{D})$. We present this routine in algorithm 2.

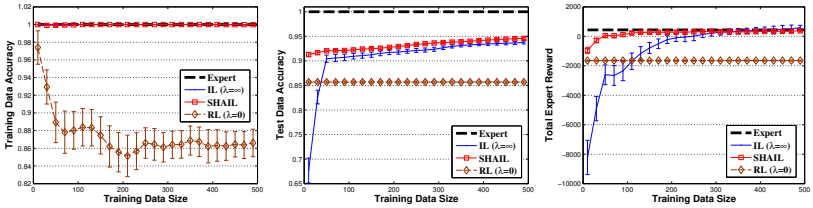
For the $V_{R_s}(\theta)$ component, we run one episode of the policy gradient RL algorithm OLPOMDP (Baxter, Bartlett, and Weaver 2001) (lines 8-15). OLPOMDP is a well-known stochastic optimization algorithm for RL. Next, we do SGA updates for each component of the likelihood term $L(\theta, \mathcal{D})$, for which the gradients can be computed in closed form. This effectively loops over the entire set of state-action pairs in the demonstration data, updating θ for each one (lines 17-19). This entire process is repeated for some number of iterations during which we keep track of the best policy found.

5 Experiments

We evaluate SHAIL in three domains: Car driving, Cartpole, and an IL formulation of handwriting recognition. In each, the learner is given a shaping reward function that encodes simple knowledge about good behavior. In this paper, all demonstrations are provided by simulated experts.

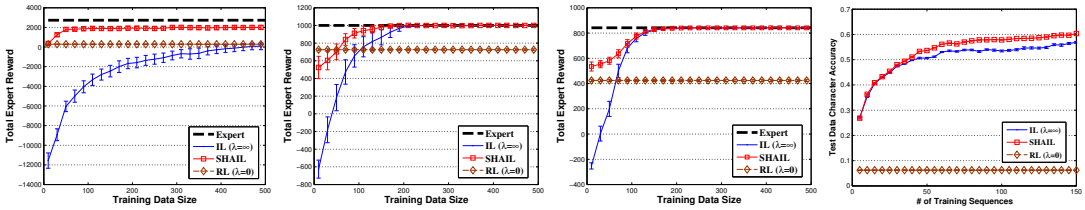
We compare SHAIL with the following baselines: 1) *Expert*, the (ground truth) entity we are trying to imitate, 2) *IL*, traditional IL which learns using only the demonstrations via Algorithm 2 without any shaping reward (effectively setting $\lambda = \infty$), 3) *RL*, which learns only by maximizing the shaping rewards via Algorithm 2 with $\lambda = 0$, which is identical to running the OLPOMDP algorithm.

We measure the performance of each algorithm accord-



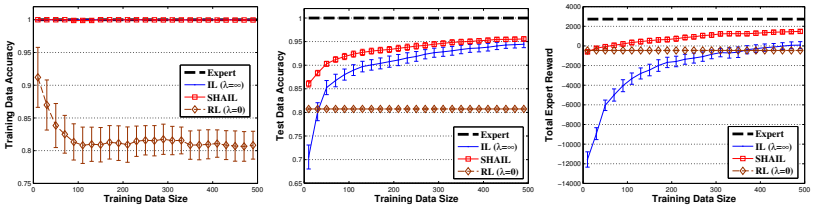
(a) (b) (c)

Figure 1: Driving-Expert1: (a) Training accuracy (b) Test accuracy (c) Total expert reward.



(a) (b) (c) (d)

Figure 2: SHAIL results: (a) Driving-Expert2 (b) CartPole-Expert1 (c) CartPole-Expert2 (d) Handwriting.



(a) (b) (c)

Figure 3: Driving-Expert2 with inconsistent shaping reward: (a) Training accuracy (b) Test accuracy (c) Total expert reward.

ing to three different criteria: 1) Training accuracy, which measures the classification accuracy of the learned policy on the training (demonstration) set. The purpose behind measuring training accuracy is to evaluate the ability of the learned policy to reproduce the specific demonstrations of the teacher. In practice, this is important, since as noted in prior work (Judah et al. 2010), users become quite frustrated when the learning system is clearly contradicting the specific examples shown to it. 2) Test accuracy, which measures the generalization capability of the learned policy using a test dataset drawn from independently generated teacher trajectories. 3) Average total expert reward accumulated during a test episode with respect to the "true expert reward function". Here the true expert reward is the reward function that the expert policy is intended to optimize, which is unknown to the learners and always more complex than the shaping reward used in our experiments. Evaluating learned policies w.r.t. the expert reward is a common practice in IL, since it gives another indication of how well the learned policies actually perform when executed in the environment, which the test accuracy only indirectly measures. In all of our experiments, we created each expert by maximizing a particular reward function via RL (sometimes an expensive process), so the expert reward was precisely known.

Each learner's policy is represented via a log-linear policy representation (a.k.a. linear logistic regression classifier) using features of state-action pairs where features correspond to state variables (details are in (Judah 2014)). We report results in the form of learning curves along with 95% confidence intervals. To generate a learning curve, we incrementally provide demonstration data to the learner and allow the learner to learn from it. We then evaluate the performance according to the above mentioned criteria. We repeat the experiment 50 times. The final learning curve is the average of the individual curves. We next describe our domains and the experts followed by the experimental results.

Car Driving. The driving domain is a simple simulation of multi-lane highway driving that has been commonly used in IL (Abbeel and Ng 2004). We use a recent implementation (Cohn, Durfee, and Singh 2011), where there are three traffic lanes and two shoulder lanes. The learner controls the movement of the car across lanes (including the shoulder lane) at a constant speed. The other cars move at a randomly chosen constant speed in fixed lanes. We allow each episode to run for 200 time steps. We created two experts to learn from in this domain. *Driving-Expert1* was the result of training a policy using SARSA(λ) with linear function approximation. At a high level, the reward function used for training Driving-Expert1 is to exhibit generally good driving behavior but to prefer the rightmost lane, not to tailgate, and to follow other cars at a safe distance. *Driving-Expert2* was trained with a reward function that allows for a riskier driving behavior in that it prefers to tailgate small sized cars, but avoids tailgating larger cars. The shaping reward used for both experts gives penalties for driving in the shoulder lane and for collisions, which agrees with most target behaviors.

Cart-Pole. Cart-pole is a well-known RL benchmark involving balancing a pole on a moving cart. We created two

different expert policies that can keep the pole balanced, but do so with different preferences. *CartPole-Expert1* is a traditional cart-pole agent that keeps the pole balanced while keeping the cart position in the range $[-1.0, 1.0]$. The reward function used to train this behavior gives $+1$ for each step the pole is kept balanced and the cart is in range, and -1 otherwise. *CartPole-Expert2* has a preference to keep the pole balanced in a goal location at -1.0 and so must move from the initial location of 0.0 to -1.0, and remain close to -1.0 while balancing the pole. The reward used to train the expert gives $+2$ when in the goal location with a balanced pole, $+1$ if the pole is balanced elsewhere, and -1 if the pole is not balanced. The shaping reward used in the main experiments only gives rewards related to pole balancing ($+1$ for balanced and -1 otherwise), and not the position of the cart.

Handwriting Recognition. We applied SHAIL to handwriting recognition using the data set from Tasker et al. (Taskar, Guestrin, and Koller 2004). In this task, we are given handwritten words and asked to assign one of the 26 character labels to each letter of the word in the left-to-right order. This problem can be viewed as IL (see for example (Daumé III, Langford, and Marcu 2009)), where at each time step (letter location), the learner has to execute the correct action (i.e. predict correct label) given the current state. The state consists of features describing the input (the current letter) and the previous C predictions made by the learner (the prediction context). In our experiments, we use $C = 1$. The expert in this domain is derived from the labels in the training set, giving the true characters. The dataset has 6600 words divided into 10 folds. We used the first fold to produce demonstration data and the remaining folds as test data.

The shaping reward is based on the linguistic knowledge of the lowercase-by-lowercase character bigram frequencies in the English language, taken from the work of Jones and Mewhort (Jones and Mewhort 2004). In particular, we exploit the fact that there are many character bigrams that are quite rare, and do not occur in commonly used English words. The shaping reward function penalizes (reward=-100) policies that output rare bigrams, and otherwise does not assign rewards for non-rare bigrams (reward=0). We defined rare bigrams by thresholding based on frequency.

Experimental Results. Figure 1 shows the results when learning to imitate Driving-Expert1. SHAIL performs nearly as well as the expert and IL in terms of training accuracy. At the same time, in terms of test accuracy, it performs better than IL, especially with little training data due to the shaping reward. This shows that SHAIL achieves better generalization to unseen states while staying competitive with IL on the training data. RL performs worse than SHAIL and IL in terms of training accuracy, since it does not consult the training data and relies on only the shaping reward. It achieves some generalization and starts out better than IL in terms of test accuracy since the shaping reward provides non-trivial information about the expert policy. However, it remains inferior to SHAIL, which finds a better trade-off between the demonstration data and shaping reward. Furthermore, RL is soon overtaken by IL as the training data grows and captures the information in the shaping reward signal. SHAIL is also superior in terms of the expert reward.

Figure 2(a),(b) and (c) shows the results when learning to imitate other experts. Due to space constraints, we only show performance in terms of total expert reward, noting that similar trends were seen in the training and test accuracy results. As seen with Driving-Expert1, SHAIL again remains an overall superior method throughout the learning curve. Figure 2(d) shows the results for handwriting recognition. RL performs very poorly since the linguistic knowledge in the shaping reward does not allow it to learn how to relate image features to characters. The Expert performance is always 1.0 and not shown. We see that SHAIL performs better than IL as the learning curve progresses. Somewhat unexpectedly, SHAIL did not show an advantage over IL early in the training. We believe that this is because initially IL has not seen labels spanning all characters, and hence it is unable to predict those characters in the output, which dominates the error rate with or without shaping. However, once the training data grows and begins to span all character labels, IL starts predicting the rare bigrams more aggressively. Our approach, in contrast, has learned not to predict the rare bigrams based on the supplied linguistic knowledge, and this results in improved character recognition rate.

So far the shaping rewards have been consistent with our experts. Now we consider a partially inconsistent shaping reward in the driving domain. Recall that Driving-Expert2 attempts to avoid collisions and driving in the shoulder lane, but has a preference to tailgate small sized cars. We modified the shaping reward function in the driving domain to add an additional penalty for any type of tailgating, where previously tailgating was not part of the shaping reward. This type of partial inconsistency between shaping rewards and the target policy should be expected in practice.

The results are shown in Figure 3. Even though SHAIL has suffered a degradation in performance due to inconsistent reward function, it continues to perform well relative to both IL and RL. SHAIL remains comparable to IL in training accuracy, and outperforms it in test accuracy and total expert reward. This shows that SHAIL can overcome inconsistent shaping rewards, and even leverage the consistent aspects of such shaping rewards to improve over pure IL.

6 Summary

There has been little work on incorporating shaping rewards into IL, despite the fact that shaping rewards are often easy to specify. A main contribution of this paper was to introduce a constrained optimization formulation for integrating shaping rewards into the IL framework. We described the SHAIL algorithm for addressing this problem and showed that it can effectively exploit shaping rewards in order to improve the data efficiency of IL. Further, SHAIL was designed to be robust to incorrect shaping rewards and was experimentally shown to perform as well as traditional IL, even when the shaping reward conflicts with the training demonstrations. We also presented a theoretical result that provides first insight into when can IL benefit from shaping rewards.

7 Acknowledgments

The authors acknowledge the support of the Office of Naval Research under the ATL grant N00014-11-1-0106.

References

Abbeel, P., and Ng, A. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Ahuja, R. K.; Magnanti, T. L.; and Orlin, J. B. 1993. *Network flows: theory, algorithms, and applications*. Prentice hall.

Baxter, J.; Bartlett, P. L.; and Weaver, L. 2001. Experiments with infinite-horizon, policy-gradient estimation. *JAIR*.

Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. K. 1987. Occam's razor. *Inf. Process. Lett.*

Chernova, S., and Veloso, M. 2009. Interactive policy learning through confidence-based autonomy. *JAIR*.

Cohn, R.; Durfee, E.; and Singh, S. 2011. Comparing action-query strategies in semi-autonomous agents. In *AAAI*.

Daumé III, H.; Langford, J.; and Marcu, D. 2009. Search-based structured prediction. *Machine learning*.

Dorigo, M., and Colombetti, M. 1994. Robot shaping: Developing autonomous agents through learning. *Artificial intelligence*.

Jones, M. N., and Mewhort, D. J. 2004. Case-sensitive letter and bigram frequency counts from large-scale english corpora. *Behavior Research Methods, Instruments, & Computers*.

Judah, K.; Roy, S.; Fern, A.; and Dietterich, T. 2010. Reinforcement learning via practice and critique advice. In *AAAI*.

Judah, K. 2014. *New Learning Modes for Sequential Decision Making*. Phd thesis, Oregon State University. http://hdl.handle.net/1957/47464.

Khardon, R. 1999. Learning to take actions. *Machine Learning*.

Knox, W. B., and Stone, P. 2009. Interactively shaping agents via human reinforcement: The TAMER framework. In *K-CAP*.

Knox, W. B., and Stone, P. 2010. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *AAMAS*.

Knox, W. B., and Stone, P. 2012. Reinforcement learning from simultaneous human and MDP reward. In *AAMAS*.

Maclin, R.; Shavlik, J.; Torrey, L.; Walker, T.; and Wild, E. 2005. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *AAAI*.

Mataric, M. J. 1994. Reward functions for accelerated learning. In *ICML*.

Ng, A., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *ICML*.

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

Pomerleau, D. A. 1989. Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document.

Ramachandran, D., and Amir, E. 2007. Bayesian inverse reinforcement learning. In *IJCAI*.

Randlov, J., and Alstrom, P. 1998. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*.

Rosenstein, M., and Barto, A. 2004. Supervised actor-critic reinforcement learning. In Si, J.; Barto, A.; Powell, W.; and Wunsch, D., eds., *Learning and Approximate Dynamic Programming: Scaling Up to the Real World*.

Ross, S., and Bagnell, J. 2010. Efficient reductions for imitation learning. In *AISTATS*.

Syed, U., and Schapire, R. E. 2007. Imitation learning with a value-based prior. In *UAI*.

Taskar, B.; Guestrin, C.; and Koller, D. 2004. Max-margin markov networks. In *NIPS*.

Thomaz, A., and Breazeal, C. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*.

Valiant, L. G. 1984. A theory of the learnable. *Commun. ACM*.