Learning from Demonstration for Shaping through Inverse Reinforcement Learning

Halit Bener Suay
Worcester Polytechnic Institute
benersuay@wpi.edu

Tim Brys
Vrije Universiteit Brussel
timbrys@vub.ac.be

Matthew E. Taylor
Washington State University
taylorm@eecs.wsu.edu

Sonia Chernova
Georgia Institute of Technology
chernova@cc.gatech.edu

ABSTRACT

Model-free episodic reinforcement learning problems define the environment reward with functions that often provide only sparse information throughout the task. Consequently, agents are not given enough feedback about the fitness of their actions until the task ends with success or failure. Previous work addresses this problem with reward shaping. In this paper we introduce a novel approach to improve model-free reinforcement learning agents' performance with a three step approach. Specifically, we collect demonstration data, use the data to recover a linear function using inverse reinforcement learning and we use the recovered function for potential-based reward shaping. Our approach is model-free and scalable to high dimensional domains. To show the scalability of our approach we present two sets of experiments in a two dimensional Maze domain, and the 27 dimensional Mario AI domain. We compare the performance of our algorithm to previously introduced reinforcement learning from demonstration algorithms. Our experiments show that our approach outperforms the state-of-the-art in cumulative reward, learning rate and asymptotic performance.

Keywords

Learning and Adaptation; Reward structures for learning; Learning agent capabilities (agent models, communication, observation)

1. INTRODUCTION

Model-free episodic Reinforcement Learning (RL) problems require agents to explore the state space and iteratively find a solution, which can be costly in terms of learning time. These problems are typically defined by an environment reward that rarely changes throughout the state-space. For example, in the Cart-Pole domain [2, 21], the environment provides a reward of $r_{env} = +1$ after each step of the task until the terminal state, where the reward is $r_{env} = 0$ to emphasize that the cart has failed at balancing the pole. The information provided by the environment is *sparse* as the change in the reward signal is infrequent. The environment lets the agent know about an obstacle or the terminal state without providing much information right until then. On the one hand, it is easy to define a sparse reward function manually and the interpretation of rewards becomes very intuitive for humans. On the other hand, with this type of environment reward agents are not given enough information about how appropriate the chosen actions are throughout the task. A model-free RL agent can benefit immensely from having a *dense* reward function that provides more information about the task.

An extensively explored way to provide denser information is through *shaping* the existing reward function by adding an extra signal, thus providing the agent with a composite reward [14, 6, 4, 8, 25]. Research in reward shaping shows that potential based shaping functions provide a theoretically sound way of incorporating additional, informative reward functions in existing reinforcement learning problems without altering the problem definition [14]. Potential based shaping rewards can improve an agent's learning efficiency by reducing the number of episodes before convergence [14]. To give an example for the Cart-Pole problem, a potential based shaping reward could be the negative of the angular distance of the pole from the center at each step. There are multiple ways to define a shaping reward function. One way is to use a priori knowledge of the domain characteristics, however, doing so requires extensive knowledge of the state space and becomes infeasible as the number of state variables increases. Recent work has shown that shaping reward can also be effectively extracted from an expert's demonstration by computing the similarity between demonstrated states and observed states throughout learning [4].

Expert demonstrations have also been applied in the context of Markov decision processes for inverse reinforcement learning (IRL), that is, for extracting a reward function given observed optimal behavior [15]. In domains where the reward function is *unknown*, IRL algorithms are used with the purpose of obtaining a reward function for optimal control.

In this work we introduce Static IRL Shaping (SIS), and Dynamic IRL Shaping (DIS), two variations of a novel approach in which we combine reward shaping and inverse reinforcement learning. SIS and DIS improve upon previ-

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

ous work's use of an expert's demonstrations by capturing the demonstrator's general policy over the state-space for shaping, as opposed to using the observed demonstration samples directly. By using model-free IRL, we encode a demonstrator's task description as a heuristic which *complements* the environment reward function. Our approach consists of three main steps: i) collecting demonstrations, ii) recovering a potential function from demonstrations, and iii) using the recovered potential function for computing shaping rewards in reinforcement learning. Reward functions we recover let us compute a linear potential based shaping reward. It is important to note that, even though we use expert demonstrations in this paper, there is no strict requirement for the demonstrations to be optimal. We use reward shaping as a heuristic, while the agent still receives the information provided by the environment reward. This means that even if demonstrations are suboptimal, the agent will ultimately be able to learn the task. This is due to the guarantees of policy invariance provided by potential-based reward shaping approaches, and that reward shaping that is not potential-based may alter the optimal policy of the problem. Our goal is to show that using a model-free IRL technique, we can extract additional information for summarizing the task for a standard RL agent and the agent can use this information to learn the task faster. To the best of our knowledge, the use of IRL for potential-based shaping reward functions has not been investigated in the literature.

In order to show that our approach is scalable over different number of state variables, we evaluate our approach in two domains: a Maze domain and the Mario AI domain. We compare SIS and DIS with the standard SARSA(λ) and Q(λ)-learning agents, as well as with two previously introduced RL algorithms that leverage demonstration data. Our experiment results show that a) both of our approaches result in significantly (p-values < 0.0001) more cumulative reward in both domains compared to other techniques, b) DIS results in faster learning in both domains and c) both approaches have better asymptotic performance than the human demonstrator.

2. BACKGROUND

Our approach builds on several research areas; we provide a high-level overview of each below.

2.1 Reinforcement Learning

RL is the problem of choosing which actions to take in observed states. We define reinforcement learning using the standard notation of Markov Decision Processes (MDPs) [16, 22]. At every time step the agent observes its state $s \in S$ as a vector of k state variables such that $s = \langle x_1, \ldots, x_k \rangle$. The agent selects an action from the set of available actions A at every time step. An MDP's reward function $R : S \times A \mapsto \mathbb{R}$ and (stochastic) transition function $T : S \times A \mapsto S$ fully describe the system's dynamics. The agent attempts to maximize the long-term reward determined by the reward and transition functions that are initially unknown to the agent.

A learner chooses which action to take in a state via a policy, $\pi : S \mapsto A$. Policy π is modified by the learner over time to improve performance, which is defined as the expected total reward. Instead of learning π directly, model-free RL algorithms instead approximate the action-value function, $Q : S \times A \mapsto \mathbb{R}$, which maps state-action pairs to the expected real-valued return.

2.2 Reward Shaping

Reward shaping, derived from behavioral psychology [20], is a way of incorporating prior knowledge in the learning process. A shaping reward F provides an agent with additional information after each state transition. With shaping rewards, an agent is rewarded for taking appropriate steps toward the goal. This extra reward supplements the environment reward which is typically defined as a sparse function for practical reasons.

The extra reward signal F, defined by agent designers, is added to the environment reward R, defined by problem designers. The agent learns a task using the shaped reward R':

$$R'(s, a, s') = R(s, a, s') + F(s, a, s') \quad (1)$$

F function usually encodes heuristic knowledge, and is intended to complement the typically less informative signal R. As we mentioned earlier, the agent's goal is defined by the reward function, and by manipulating the reward signal we may risk changing the task. Ng et al. [14] proved that potential-based shaping is a sound way to provide a shaping reward without changing the reinforcement learning problem.

2.3 Inverse Reinforcement Learning

IRL is the problem of learning a reward function using a set of observations from expert demonstrations [15, 1]. Two possible general representations of a task for MDPs are through defining a policy or a reward function. IRL algorithms aim to find a good representation of the demonstrated task by finding the most fitting reward function based on a set of demonstrations. Although there are different solutions for the IRL problem, many of them require the environment model (i.e. the transition function) to be known [17, 26, 11]. Recently Boularias et al. [3] and Klein et al. [10] introduced two different algorithms for model-free IRL. Boularias et al. introduced a sampling based algorithm Relative Entropy Inverse Reinforcement Learning (RE-IRL) that leverages statistical properties of human demonstrations [3]. Klein et al. introduced Cascaded Supervised IRL (CSI) which combines classification with regression in order to generate a reward signal based on human demonstrations [10]. In this work we use RE-IRL due to its simplicity and availability however a comparison of different IRL algorithms for our particular use can be a future investigation topic.

2.4 RL from Demonstration

RL from Demonstration (RLfD) is a learning setting for RL agents where both demonstrations and an environment reward signal are available. Algorithms use different methods to convert demonstrations into a representation that the agent will be able to use as heuristic information and the environment reward is treated as the ground truth for learning. Early work on Reinforcement Learning from Demonstration (RLfD) was presented by Schaal [19], in which the term described a set of approaches that leveraged initial training for learning a policy and a value function from demonstrations, followed by a "trial by trial learning" with RL.

Later, Taylor et al. introduced Human - Agent Transfer (HAT), a transfer learning algorithm that employs rule transfer [23] in order to summarize a set of human demonstrations for bootstrapping a reinforcement learning agent's performance. HAT follows a three step approach: i) A human demonstrator controls the agent and performs the task; ii)

Saved demonstrations are used as the input data for a rule learning algorithm; iii) A RL agent uses the summarized policy to initialize the state-action value function.

Most recently, Brys et al. introduced a new algorithm we refer to as Similarity Based Shaping (SBS) [4]. With the SBS algorithm Brys et al. took a different approach to investigate the use of prior knowledge as a potential-based shaping reward. The SBS algorithm uses human demonstrations as a potential function $\Phi^D(s,a)$ for potential-based shaping. With this approach each demonstrated state-action pair is assumed to be desirable for the agent, and using a similarity metric, the SBS algorithm computes a shaping reward F. During learning, the agent receives a shaped reward signal $R' = R + F$ which is the combination of the environment reward (i.e. ground truth) and the shaping reward. The SBS algorithm uses non-normalized multi-variate Gaussian distributions to compute the similarity between demonstrated states and observed states throughout learning.

HAT and SBS algorithms allow the use of multiple demonstrations, combining observations obtained from these demonstrations in different ways. In practice, the use of multiple demonstrations increases the possibility of recording noisy or sometimes conflicting state-action pairs. If that is the case, using demonstrations directly similar to the SBS algorithm may result in conflicting rewards for the agent. We use IRL because as a method it generalizes over multiple demonstrations. Our novel algorithms improve on the above techniques by leveraging IRL in combination with reward shaping. We compare our results to HAT and SBS.

3. METHODOLOGY

In this section, we present our approach for using a model-free IRL algorithm to obtain a shaping reward and speed up reinforcement learning. Our goal is to incorporate human task demonstration in an intuitive way and incorporate this knowledge in reinforcement learning without changing the task (i.e. the main goal). We aim to reinforce appropriate action selection as soon as an episode starts, much before the agent reaches the terminal state. Each shaping reward signal can be thought of as a *hint* for the agent to make better decisions. To the best of our knowledge, using a potential-based shaping reward function recovered with IRL has not been investigated in the literature. Our method has three main steps:

Collecting Expert Demonstrations: First, we collect observation data from a demonstrator. Each demonstration is a set of sequential observations in time. In this paper our focus is on episodic, simulated environments and we collect the agent's state vector as demonstrations. The demonstrator was an expert teacher, however this is not a requirement for our setup. We assume that the demonstrations are not malicious, that is, the intent of each demonstrated state transition is to perform well on the task. RE-IRL requires trajectories to have equal lengths. To equalize the length of demonstrations we process the demonstration data by repeating the last observation (in our first test domain) or pruning longer demonstrations (in the second test domain). Depending on the task, if demonstrating the goal state carries importance one could prefer to append a repetition of the goal state in the set of demonstrated observations, or otherwise prune the observations. In the end we obtain a set of demonstrations $\mathcal{T}$. The input for the next step (i.e. RE-IRL) is $\mathcal{T}$, where each trajectory $\tau \in \mathcal{T}$ is a set of observations of equal length, $\tau = \{s_1, \ldots, s_l\}$.

Recovering a Linear Function: Once we obtain $\mathcal{T}$, we use RE-IRL and compute the vector of reward feature weights $\mathbf{w}$ to use in $R_{IRL}(s)$. RE-IRL recovers a reward function $R_{IRL}(s,a)$ for the set of demonstrations $\mathcal{T}$ as a linear combination of observed reward features $\mathbf{f} = \langle f_1, \ldots, f_n \rangle$, weighted with $\mathbf{w}$. The output is the vector of feature weights $\mathbf{w} = \langle w_1, \ldots, w_n \rangle$:

$$R_{IRL}(s) = \sum_{i=1}^{n} \omega_i f_i(s) \tag{2}$$

where, f_i is ith reward feature, ω_i is ith feature weight, s is the state vector for the current state of the agent. Each reward feature is computed from the state vector s and the mapping from state variables to reward features is typically domain dependent, defined by agent designers.

In our representation, we only use the state variables to represent the path of the agent and the policy of the demonstrator, hence the potential function is independent of the actions. RE-IRL obtains the weight vector $\mathbf{w}$ by finding a distribution $\mathcal{P}$ that minimizes the KL-Information loss $I(\mathcal{P}, \mathcal{Q})$ over trajectories. KL-Information loss $I(\mathcal{P}, \mathcal{Q})$ is the information lost when the distribution $\mathcal{Q}$ is used to approximate the distribution $\mathcal{P}$ [5]. For this work we are interested in transferring the expert demonstration policy from humans to agents, hence we use the distribution $\mathcal{Q}$ as the distribution of observed expert trajectories in state space. RE-IRL tries to find a vector $\mathbf{w}$ that minimizes the KL-Information loss. With $\mathbf{w}$, during learning, when $\mathbf{f_{observed}} \approx \mathbf{f_{demonstration}}$ feature weights return a higher reward to reinforce the agent's state-action values. For the full derivation of the solution we refer readers to [3]. An explanation of RE-IRL with a pseudo-code for the algorithm is also presented by Muelling et al. [13].

Using The Recovered Function: Last, we compute potential-based shaping rewards that the agent receives during learning. For this step we tested the use of the recovered linear function as a traditional, static potential function and also as a dynamic potential function. To implement potential-based shaping we define the above potential function $\Phi : S \mapsto \mathbb{R}$ over the state space, and define the shaping reward F as the difference between the potential of the new state s' and the old state s:

$$F(s,a,s') = \gamma\Phi(s') - \Phi(s) \tag{3}$$

where $0 \leq \gamma \leq 1$ is the discount factor which describes the present value of future rewards and potentials [22]. The effect of shaping is that the agent's exploration is less random and the agent is biased towards states with high potential. In this work we use inverse reinforcement learning to obtain a linear function $\Phi(s) = R_{IRL}(s)$ from expert teacher demonstrations. We call this approach Static IRL Shaping (SIS). We name the approach *static* since the potential function does not change over time.

The formulation given in Eq.(3) is limited in that the shaping reward is dependent only upon the states encountered by the agent and not its actions. Wiewiora et al. [25] extend the definition of F and Φ to state-action pairs (s,a) as:

$$F(s,a,s',a') = \gamma\Phi(s',a') - \Phi(s,a) \tag{4}$$

This formulation allows more information to be incorporated, as it is not limited to states but uses action choices for computing the shaping reward.

Devlin et al. [6] extended Ng's potential-based reward shaping to dynamic potential-based shaping, allowing the shaping function to change over time:

$$F(s,t,s',t') = \gamma\Phi(s',t') - \Phi(s,t) \quad (5)$$

By changing the shaping function over time, this formulation allows incorporating learning in the shaping function based on agent's experience in order to improve the shaping function. Harutyunyan et al. [8] combine these two extensions into *dynamic shaping* over state-action pairs:

$$F(s,a,t,s',a',t') = \gamma\Phi(s',a',t') - \Phi(s,a,t) \quad (6)$$

Importantly, the above variants of shaping all preserve the guarantees such as policy invariance and consistent Nash Equilibria proven by Ng et al [14], and Devlin and Kudenko [6]. Using dynamic shaping, any expert provided reward function $R^\dagger$ can be transformed into a potential-based shaping function by learning a secondary Q-function $\Phi^\dagger$, which serves as the potential function for shaping. The formulation below defines the dynamic potential-based shaping reward. The secondary value function must be learned on-policy, with a technique such as SARSA [18]:

$$\Phi^\dagger(s,a) \leftarrow \Phi^\dagger(s,a) + \beta\delta^{\Phi^\dagger} \quad (7)$$

Where β is the learning rate, and $\delta^{\Phi^\dagger}$ is defined as the temporal difference (TD) error of the state transition:

$$\delta^{\Phi^\dagger} = r^{\phi^\dagger} + \gamma\Phi^\dagger(s',a') - \Phi^\dagger(s,a) \quad (8)$$

As the secondary Q-function $\Phi^\dagger$ gets updated after each step the shaping function becomes:

$$F = \gamma\Phi^\dagger(s',a') - \Phi^\dagger(s,a) \quad (9)$$

By substitution we obtain:

$$\delta_t^{\Phi^\dagger} = r^{\phi^\dagger} + F \quad (10)$$

By definition, the temporal difference error (i.e. the error between the estimated Q-value and the actual return) goes to 0 at the time of convergence. Thus for $\delta^{\Phi^\dagger} = 0$:

$$r^{\phi^\dagger} + F = 0 \quad (11)$$

And finally at the time of convergence the potential-based shaping function becomes:

$$F(s,a) = R^\dagger(s,a) = -r^{\phi^\dagger} \quad (12)$$

When the secondary value function has converged (i.e. when $\delta^{\Phi^\dagger} = 0$), the main value function $Q(s,a)$ will be supplied with a potential-based reward shaping that is equivalent to the reward function $R^\dagger$. Even before convergence, $\Phi^\dagger$ provides useful information about $R^\dagger$, just like the main Q- function provides information about good policies before converging to an optimal policy. As implied by the last equation above, we learn the secondary Q-function $\Phi^\dagger$ using the negative of the expert defined reward $r^{\phi^\dagger} = -R^\dagger$. In our work, $R^\dagger(s,a,s') = R_{IRL}(s)$ is the reward function we recover using inverse reinforcement learning. We call this approach Dynamic IRL Shaping (DIS), as the potential function changes over time based on the learned, secondary Q-function which uses Equation 2 as its reward input.

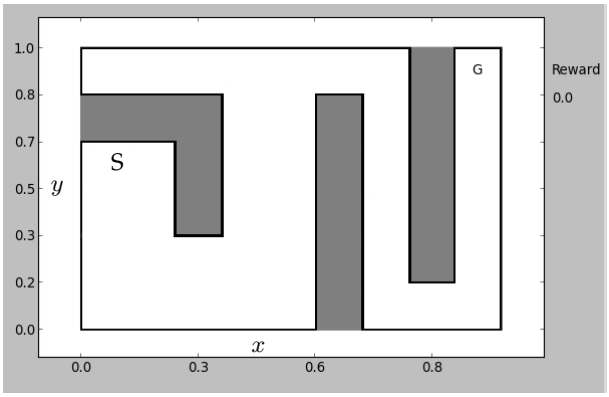

Figure 1: Maze domain. The state-space consists of two continuous variables x, y that define the horizontal and vertical location of the agent in the maze with reference to the lower-left corner. The agent starts from S and its goal is to navigate around the gray walls and reach the goal destination G.

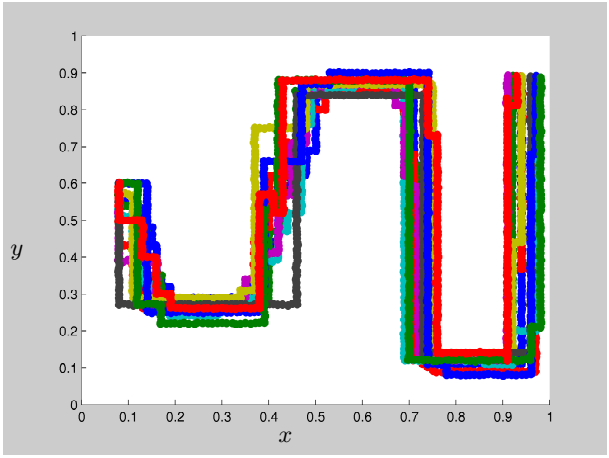

Figure 2: Suboptimal demonstrations in the Maze domain.

4. EXPERIMENTAL VALIDATION

In order to validate our approach we compare the performance of our algorithms against HAT with value-bonus [24] and SBS in two domains: a relatively simple Maze domain, and the Mario AI [9] domain. For both domains, we investigate the initial performance, the asymptotic performance, and the total cumulative reward for evaluation.

In both domains we have 5 agents: a standard RL agent, a SIS agent, a DIS agent, a HAT agent and a SBS agent. For the Maze domain we use SARSA(λ) as the standard learner and the underlying learning algorithm for all other agents. For the Mario AI domain we use Q(λ)-learning as the standard learner and the underlying learning algorithm for all other agents (with the exception of the secondary learner in DIS, which has to be a SARSA learner).

4.1 Maze

The Maze domain shown in Fig.1 is a modified version of the Dyna Maze domain introduced by Sutton and Barto [22]. In this domain the goal of the agent is to start from the position marked with *S*, navigate around the walls shown as

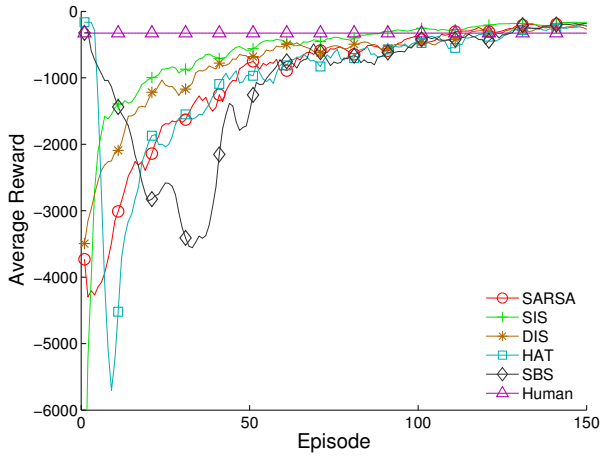

Figure 3: Maze domain experiment results.

Table 1: Average initial (first episode) performance, asymptotic performance, and cumulative reward with standard deviations for all agents in the Maze domain. Maximum values are shown in bold. Higher values are better. SAR: SARSA, HMN: average of the reward human demonstrator received in 10 demonstrations.

Agents	μ_{init}	μ_{asymp}	μ_{cumul}
SAR	-3,730.36	-222.52	-151,525.96
SIS	-7,973.16	**-165.88**	**-93,591.96**
DIS	-3,494.08	-267.88	-113,689.12
HAT	**-164.68**	-222.36	-150,141.28
SBS	-324.04	-185.84	-159,634.52
HMN	-327.5	-327.5	

dark gray cells and reach the goal state marked with G. In our version the state-space consists of two continuous variables $0 \leq x, y < 1.0$. The agent starts at $s = (x : 0.08, y : 0.6)$, and chooses one of four available actions: left, right, up, and down. Step size for each action is 0.02. After each step the agent receives an environment reward of -1. If the agent tries to get outside the domain boundaries in any direction, or hits a wall, it receives an environment reward of -10. The terminal condition is $(x \geq 0.9)$ && $(y \geq 0.9)$ with an environment reward of $+10$. This is an episodic task and each episode starts form the initial state and ends at the goal state (or terminates with failure after 6000 steps).

In order to evaluate our approach we collected 10 suboptimal demonstrations in the Maze domain. The demonstrator was the first author who also designed and implemented the modifications to the original Dyna Maze domain. Fig. 2 shows the suboptimal demonstrations. Features we recorded during demonstrations were the horizontal and vertical location of the agent at each time-step, that is, $s_t(x_t, y_t)$ for each observation s_t. The shortest demonstration took 328 steps to reach the goal and the longest demonstration was 353 steps long ($\mu = 337.5$ and $\sigma = 10.71$ steps). None of the demonstrations had the agent collide with the walls or the borders of the maze. For RE-IRL, we appended the last observation of the shorter demonstrations repeatedly in order to equalize the lengths of the demonstrations. We used the following parameters for our agents: number of tiles = 32, $\alpha = 0.25/\#tiles$, $\beta = 0.05/\#tiles$, $\gamma = 1.0$, $\lambda = 0.25$, with $\epsilon-$greedy action selection where $\epsilon = 0.02$. Finally, we used a scaling of 100 for all shaping reward signals, except for the HAT agent where the scaling was 10.

Table 2: Statistical analysis for the Maze domain results show p-values for two-sample *t*-tests. SAR stands for SARSA. Columns show the statistical significance of the difference between the initial (first episode) performance, asymptotic (last episode) performance, and cumulative reward (i.e. total reward throughout the trial).* p $\leq$ 0.05, p $\leq$ 0.01,*** p $\leq$ 0.001,**** p $\leq$ 0.0001.**

Agents	Initial	Asymp.	Cumulative
SAR vs SIS	3.8929e-16****	0.051603	1.4592e-44****
SAR vs DIS	0.62928	0.56365	8.3354e-33****
SAR vs HAT	1.5995e-13****	0.99673	0.51621
SAR vs SBS	1.6913e-12****	0.26082	0.022387*
DIS vs HAT	3.4262e-13****	0.56402	1.0199e-22****
DIS vs SIS	9.6501e-18****	0.17313	5.2131e-26****
DIS vs SBS	4.1881e-12****	0.28159	5.4586e-18****
SIS vs HAT	3.0772e-94****	0.060007	4.3662e-32****
SIS vs SBS	2.597e-55****	0.32015	9.6945e-25****
SBS vs HAT	0.062916	0.27499	0.015977*

Fig. 3 shows the agents' performance in 150 episodes, averaged over 25 trials. The vertical axis shows the agents' performance in terms of total reward received per episode and the horizontal axis shows the number of episodes. Each line shows the performance of a different agent. For comparison we also show human demonstrator's performance averaged over 10 demonstrations. Average initial (first episode), asymptotic (last episode), and cumulative reward are detailed in Table 1.

First, we note that all five algorithms are able to achieve a similar asymptotic performance in this domain, although they reach it at different rates. For example, if we consider the number of episodes each algorithm takes to surpass the performance of the demonstrator, then we see that SARSA(λ), SIS, DIS, HAT and SBS require 110, 90, 115, 135, and 136 episodes, respectively. In summary, SIS learns the task remarkably faster than all other methods.

The performance of the different techniques also varies largely during the learning process. The standard SARSA(λ) agent starts from $-3,730$ and reaches its asymptotic performance after 130 episodes. The SIS approach starts with the worst performance, $-7,973$, however is the fastest to surpass the demonstrator's performance only after 90 episodes. In comparison the DIS approach starts with an initial performance of $-3,494$ and learns the task comparably fast.

The HAT algorithm, initializes the Q-function with the potential function at the very beginning. The first column of Table 1 shows that the initialization results in high performance during first episodes.However later on, as the agent probabilistically explores the state-space its performance degrades over time. It is important to point out that the performance boost gained by amount of initial jump-start is dependent of the demonstration quality and demonstrator's ability. Taylor et al. [24] have investigated the impact of demonstration quality and demonstrator expertise. HAT suffers from a performance loss after about 10 episodes due to the negative step reward. The agent starts with the "good" policy, however receives negative reward at every step and thus starts

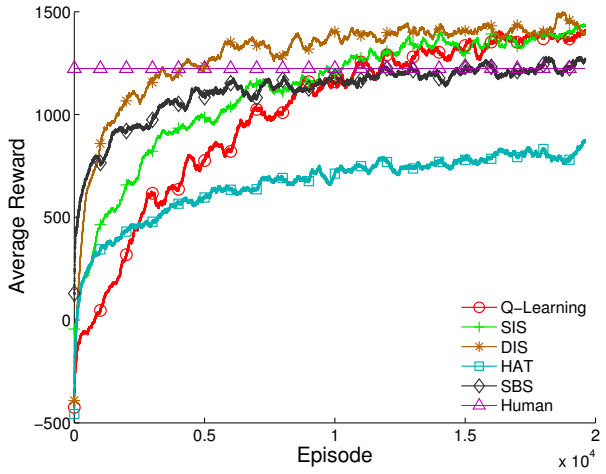

Figure 4: Mario domain experiment results.

to explore beyond the good policy. After exploring different states only to learn that they were "bad" states, the agent recovers and improves its performance to catch-up with the learning speed of the standard SARSA(λ) agent. This finding is in parallel with Taylor et al.'s [24] observations, in the sense that human demonstrations do not hurt the agent's final performance.

Similarly, the SBS approach implements Q-function initialization and starts with a high performance which degrades over time with exploration. After about 136 episodes the SBS agent reaches the demonstrator's performance and soon after converges to its asymptotic performance of -186.

Using the rewards agents collected per trial, we performed two-sample *t*-tests to evaluate whether the agents' initial performance, asymptotic performance and cumulative reward were significantly different from each other (Table 2, sample size: 25 trials).

The difference in asymptotic performance is not significant as can be obvious from Fig. 3. That being said, we would like to underline that all methods have surpassed human demonstrator's average performance. Especially SIS results in about a 50% increase in asymptotic performance after 150 episodes w.r.t the demonstrator's performance (second column of Table 1).

As for the initial performance data, there wasn't a significant difference between the DIS and the SARSA agents. However, the HAT and the SBS agents performed significantly better than the SARSA agent.

Finally, both the SIS and the DIS agents collected significantly more reward throughout 150 episodes (i.e. cumulative reward) than the SARSA, the HAT and the SBS agents.

In summary, the SIS agent shows the steepest learning curve by converging to a stable policy fastest and collects the highest cumulative reward of approximately -93,000 after 150 episodes. The DIS agent performs comparably good with a better initial performance than SIS, however collects less cumulative reward, approximately -113,000.

4.2 Mario AI

Super Mario Bros is a popular 2-D side-scrolling video game developed by the Nintendo Corporation. In this work we use the publicly available Mario AI version released by Karakovskiy and Togelius [9]. There are many possible representations of the state-space in this domain and here in order to test our approach in a large state-space, we use 27 discrete state-variables. Boolean state variables *jump*, *ground*, and *shoot* define whether Mario is able to jump, is standing on a tile, or is able to shoot fireballs. We use *x-dir* and *y-dir* variables that can take on either one of $\{-1, 0, 1\}$ values to define the direction of the agent's movements. The immediate surrounding of the agent is discretized in 8 grid cells. For each grid cell we have a state variable *close-enemies* defining whether there is a enemy in the matching cell. We also keep track of the grid cells that are one step farther than the immediate cells, and we have 8 boolean variables called *mid-enemies*. We keep four boolean state variables *obstacle* to identify whether the four vertical cells to the immediate right of Mario are occupied by obstacles such as pipes. Finally we have two variables *closest-enemy-x* and *closest-enemy-y* that define the Euclidean distance between the agent and the closest enemy within a 22x22 grid. This state space is inspired by Liao's work [12]. We note that this domain is a relatively high dimensional domain compared to most standard RL benchmarking domains. The main reason of our evaluation in the Mario AI domain is to show that our model-free approach is scalable. The agent can choose one of 12 actions combining one of each options of the following three sets: {left, right, no-action}, {jump, do not jump}, {run, do not run}. The goal of the agent is to maximize the total reward it receives throughout the level. The agent receives a reward of $+10$ for stepping on an enemy, $+58$ for eating a mushroom, $+64$ for eat-

Table 3: Performance metrics for the agents in Mario AI: initial performance, asymptotic performance, cumulative reward (average of 10 trials). Maximum values are shown in bold. Higher values are better.

Agents	μ_{init}	μ_{asymp}	μ_{cumul}
QLR	-424.6	1,410.4155	20,101,973.6
SIS	-43.6	**1,431.5328**	22,341,947.6
DIS	-391.2	1,398.7714	**25,725,731.6**
HAT	-457	867.2241	13,250,664.4
SBS	**129.8**	1,267.8642	22,332,848.2
HMN	1,224	1,224	

Table 4: Statistical analysis for the Mario AI domain results show p-values for two-sample *t*-tests. QLR stands for Q-Learning. Columns show the statistical significance of the difference between the initial performance, asymptotic (final) performance, and cumulative reward (i.e. total reward throughout the trial).

Agents	Initial	Asymp.	Cumulative
QLR vs SIS	0.018698*	0.99448	0.0010416**
QLR vs DIS	0.65987	0.11448	5.5167e-09****
QLR vs HAT	0.60982	0.71677	9.0692e-12****
QLR vs SBS	0.050229	0.059223	0.00014223***
DIS vs HAT	0.40846	0.23382	1.6551e-18****
DIS vs SIS	0.037257*	0.094503	1.9024e-06****
DIS vs SBS	0.067898	9.1049e-05****	1.9184e-08****
SIS vs HAT	0.012506*	0.70797	4.121e-15****
SIS vs SBS	0.56634	0.040042*	0.98207
SBS vs HAT	0.039996*	0.023199*	4.0285e-21****

ing a fire-flower, $+16$ for collecting a coin, $+24$ for finding a hidden block, $+1024$ for successfully finishing the level, -42 for getting hurt by an enemy (e.g. losing fire-flower), and -512 for dying. The maximum number of steps per episode is limited by the game's timer and the default timeout is 200 seconds. In the context of potential-based reward shaping, recent work by Harutyunyan et al. [7] implements and evaluates their reward shaping framework [8] in Mario AI domain as well.

We used the 10 demonstrations' first 953 observations for RE-IRL, and used the same demonstrations in full length for all other methods. For RE-IRL, the 10th longest demonstration was 953 observations long and we pruned the first nine demonstrations at this length to keep the demonstration length consistent. We used the following parameters for our agents: number of tiles = 32, $\alpha = 0.01/\#tiles$, $beta = 0.05$, $\gamma = 0.9$, $\lambda = 0.5$, with $\epsilon-$greedy action selection where $\epsilon = 0.05$, and $scaling = 1$ for the shaping reward.

We ran 10 trials for all five agents' performance evaluation. The level our agent played in Mario was designed as an episodic task where the agent finishes a level either with success or the level ends with failure with the agent's death, or a timeout of 200 seconds. Every episode, the agent plays a randomly generated level, starting from a randomly selected mode (small, large, fire-mario).

Fig. 4 shows the agents' performance in 20,000 episodes, averaged over 10 trials. The vertical axis shows the average total reward (i.e. points) agents collected during each episode. The horizontal axis shows the number of episodes. Each line shows the performance of an agent. Again, for comparison, we also show the human demonstrator's performance. Table 3 shows the average initial performance, asymptotic performance, and cumulative reward for each agent. On average the demonstrator scored 1224 points in this domain.

Unlike the previous domain not every agent was able to achieve the same asymptotic performance after 20,000. Three agents' asymptotic performance (SIS, DIS, and Q(λ)) is significantly better than the remaining two (HAT and SBS). Again, if we consider the number of episodes each agent takes to surpass the performance of the user, then we see that Q(λ), SIS, DIS, and SBS require $11,000$, $10,000$, $4,000$ and $12,000$ episodes respectively. The HAT agent was unable to reach or surpass the demonstrator's performance.

Similar to our previous experiment we observe a great difference in the learning process. With the exception of the SBS agent, most agents start with similar initial performance and negative total reward for their first several hundred episodes. The SBS agent leverages Q-function initialization and starts with a positive initial performance. The initial performance of the SIS agent was significantly better than the standard Q(λ)-learning agent, the DIS agent and the HAT agent. That said, the SIS agent's asymptotic performance was only better compared to the SBS and the HAT agents.

In terms of cumulative reward, the SIS agent performed significantly better than all other agents but the DIS agent. Table 3 shows that the DIS agent has the best overall cumulative reward performance, exceeding a cumulative reward of 25M in 20,000 episodes. The DIS agent also is the fastest to reach to the demonstrator's performance after about $4,000$ episodes, reaching the best asymptotic performance after about $8,000$ episodes before any of the other methods even reach the level of the demonstrator.

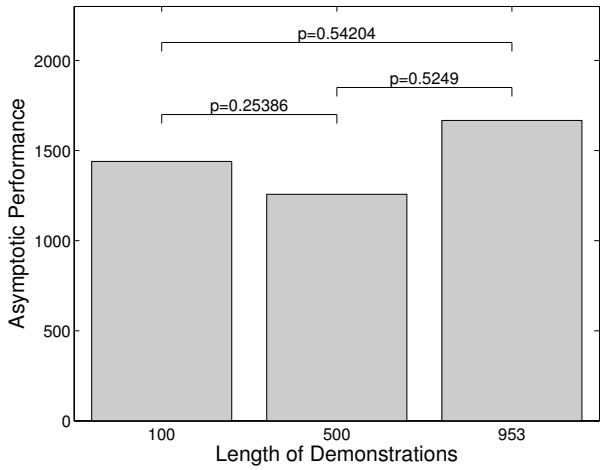

Figure 5: Asymptotic performance of the DIS agent after 20,000 episodes in Mario AI. Here the performance is a function of the length of demonstrations (the number of state-action pairs recorded per demonstration). Each condition has 10 demonstrations. P-values obtained with two-sample *t*-tests between marked conditions for 10 trials. Values for the bar plot are the average of 10 trials.

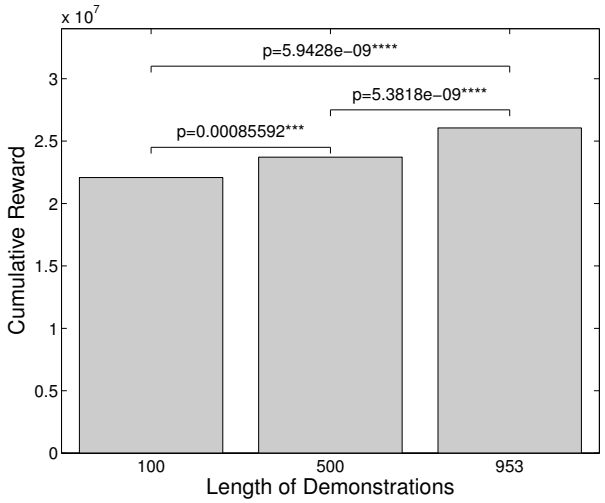

Figure 6: Cumulative reward the DIS agent received during 20,000 episodes in Mario AI. Here the cumulative reward is a function of the length of demonstrations (the number of state-action pairs recorded per demonstration). Each condition has 10 demonstrations. P-values obtained with two-sample *t*-tests between marked conditions for 10 trials.

The SBS algorithm shows quick convergence however its final performance remains limited at about 1,267 points, very close to demonstrator's performance, after 20,000 episodes. This is because SBS takes observed state's similarity with demonstrations and rewards the agent if the agent acts similar to demonstrations. As pointed out by Brys et al. [4] SBS performs better with minimum amount of consistent demonstrations (even with a single demonstration in the case of Cart-Pole domain). Although given enough number of episodes the algorithm is guaranteed to reach the underlying Q(λ)-learning's performance, suboptimal demonstrations affect the

speed of learning. Finally, the HAT agent performs slightly better than standard Q(λ)-learning only for the first 250 episodes. This is because it is difficult to summarize the demonstrator's policy based on a limited number of observations. This shows the slight performance boost that policy transfer brings, however for the rest of the trials, HAT agent shows slow improvement resulting in the significantly worst performance after 20,000 episodes.

After an analysis of learning curves and the agents' performance, we also briefly investigated the effect of the length of demonstrations on the asymptotic performance and the cumulative reward. For this set of experiments, we chose the Mario AI domain as it is a harder task and we used the DIS agent as it was the fastest learner with the highest cumulative reward in our previous experiment.

Fig. 5 shows the average asymptotic performance results for 10 trials using 100, 500 and 953 observations for 10 demonstrations. We did *not* find a significant difference between the asymptotic performance data for varying length of demonstrations. We conclude that this is because the DIS agent learns using a secondary Q-function, and even if the demonstrations do not contain enough information, in this case $20,000$ episodes were enough for using the environment reward to eventually achieve the same asymptotic performance. However, we did find a significant difference in the cumulative reward of the agents. Fig. 6 shows that as the number of observations per demonstration increase, the cumulative reward significantly increases. The results in Fig. 5 and Fig. 6 imply that the length of demonstrations do affect the learning *speed* without affecting the asymptotic performance of the DIS agent.

5. CONCLUSION

In this paper we introduced a novel, three step approach to use human demonstrations for potential-based reward shaping in reinforcement learning. Specifically, with our method, we collect demonstrations in a domain, we then use the demonstrations to recover a linear function using inverse reinforcement learning, finally we use this recovered function to compute a shaping reward for potential-based shaping in reinforcement learning. Our approach is model-free and scalable. To show the scalability of our approach we ran two sets of experiments in a simple two dimensional continuous Maze domain, and a complex, 27 dimensional Mario domain with discrete state-variables. We compared our algorithm to standard RL and two previously introduced reinforcement learning from demonstration algorithms. Our experiments show that our approach results in a significant increase in cumulative reward collected by the agents. Our agents' asymptotic performance surpasses the human demonstrator's average performance. One of our approaches, Static IRL Shaping results in faster learning with higher cumulative reward in the simpler Maze domain, whereas our second approach, Dynamic IRL Shaping results in faster learning with higher cumulative reward in the more complicated Mario AI domains. One possible explanation for this behavior is that the Maze task was not complex enough to observe the benefits of the secondary policy the DIS algorithm learns. This indicates that in simple domains, learning a reward function by demonstrations can be enough to improve a learner's performance compared to previously introduced transfer learning techniques. It is also likely that the high dimensionality and the state representation of the Mario AI domain makes it difficult for the IRL algorithm to recover a suitable reward function for all sections of the state space. The intuition behind the DIS algorithm is that the secondary policy can generalize over the state space and provide a more informative, adaptive shaping reward signal for the agent. Which may explain why the DIS agent performs better in this domain. Finally we also briefly investigated the effect of the number of observations on learning. We found that our approach learns faster as the demonstrations contain more observations as this results in a more complete representation of the task.

We believe that using demonstrations originating either from a human teacher or another agent, is a promising direction of research for incorporating a priori knowledge in reinforcement learning. For future work, we are investigating the impact of demonstration *quality* on learning and the use of multiple, locally weighted, learned shaping rewards.

Acknowledgments

Halit Bener Suay's work is supported by the Office of Naval Research award N000141410795. Tim Brys is funded by a PhD scholarship of the Research Foundation Flanders (FWO). This research has taken place in part at the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL research is supported in part by grants AFRL FA8750-14-1-0069, AFRL FA8750-14-1-0070, NSF IIS-1149917, NSF IIS-1319412, USDA 2014-67021-22174, and a Google Research Award.

REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Twenty-first international conference on Machine learning - ICML '04*, New York, New York, USA, 2004.

[2] A. Barto, R. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-13(5):834–846, 1983.

[3] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 182–189, 2011.

[4] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé. Reinforcement learning from demonstration through shaping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[5] K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer, 2002.

[6] S. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 433–440. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[7] A. Harutyunyan, T. Brys, P. Vrancx, and A. Nowé. Shaping mario with human advice. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1913–1914, Richland, SC, 2015.

[8] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowe. Expressing arbitrary reward functions as potential-based advice. In *AAAI Conference on Artificial Intelligence*, 2015.

[9] S. Karakovskiy and J. Togelius. The mario ai benchmark and competitions. *Computational Intelligence and AI in Games, IEEE Transactions on*, 4(1):55–67, 2012.

[10] E. Klein, B. Piot, M. Geist, and O. Pietquin. *ECML, Prague, Czech Republic, 2013*, A Cascaded Supervised Learning Approach to Inverse Reinforcement Learning, pages 1–16. Springer Berlin Heidelberg, 2013.

[11] S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*, pages 41–48, 2012.

[12] Y. Liao, K. Yi, and Z. Yang. Cs229 final report reinforcement learning to play mario. Technical report, Stanford University, 2012.

[13] K. Muelling, A. Boularias, B. Mohler, B. Schölkopf, and J. Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological cybernetics*, 108(5):603–19, Oct. 2014.

[14] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999.

[15] A. Y. Ng and S. J. Russell. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 663–670. Morgan Kaufmann Publishers Inc., 2000.

[16] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[17] N. Ratliff, J. Bagnell, and M. Zinkevich. Maximum margin planning. *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[18] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, University of Cambridge, Department of Engineering, 1994.

[19] S. Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems 9*, pages 1040–1046. MIT Press, 1997.

[20] B. F. Skinner. *The behavior of organisms: An experimental analysis.* Appleton-Century, 1938.

[21] R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts - Amherst, 1984.

[22] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.

[23] M. E. Taylor and P. Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the 24th international conference on Machine learning*, pages 879–886. ACM, 2007.

[24] M. E. Taylor, H. B. Suay, and S. Chernova. Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 617–624. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[25] E. Wiewiora, G. W. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 792–799, 2003.

[26] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum Entropy Inverse Reinforcement Learning. In *AAAI*, pages 1433–1438, 2008.