In *The AAAI Spring Symposium, Help Me Help You -- Bridging the Gaps in Human-Agent Collaboration* (AAAI-11, SS05),
Palo Alto, CA, March 2011.

Using Human Demonstrations to Improve Reinforcement Learning

Matthew E. Taylor, Halit Bener Suay, and Sonia Chernova
Lafayette College, *taylorm@lafayette.edu*
Worcester Polytechnic Institute, {*benersuay, soniac*}*@wpi.edu*

Abstract

This work introduces *Human-Agent Transfer* (HAT), an algorithm that combines transfer learning, learning from demonstration and reinforcement learning to achieve rapid learning and high performance in complex domains. Using experiments in a simulated robot soccer domain, we show that human demonstrations transferred into a baseline policy for an agent and refined using reinforcement learning significantly improve both learning time and policy performance. Our evaluation compares three algorithmic approaches to incorporating demonstration rule summaries into transfer learning, and studies the impact of demonstration quality and quantity. Our results show that all three transfer methods lead to statistically significant improvement in performance over learning without demonstration.

Introduction

Agent technologies for virtual agents and physical robots are rapidly expanding in industrial and research fields, enabling greater automation, increased levels of efficiency, and new applications. However, existing systems are designed to provide niche solutions to very specific problems and each system may require significant effort. The ability to acquire new behaviors through learning is fundamentally important for the development of general-purpose agent platforms that can be used for a variety of tasks.

Existing approaches to agent learning generally fall into two categories: independent learning through exploration and learning from labeled training data. Agents often learn independently from exploration via *Reinforcement learning* (RL) (Sutton and Barto 1998). While such techniques have had great success in offline learning and software applications, the large amount of data and high exploration times they require make them intractable for real-world domains.

On the other end of the spectrum are *learning from demonstration* (LfD) algorithms (Argall et al. 2009). These approaches leverage the vast experience and task knowledge of a person to enable fast learning, which is critical in real-world applications. However, human teachers provide particularly noisy and suboptimal data due to differences in embodiment (e.g., degrees of freedom, action speed) and limitations of human ability. As a result, final policy performance achieved by these methods is limited by the quality of the dataset and the performance of the teacher.

This paper proposes a novel approach: use RL *transfer learning* methods (Taylor and Stone 2009) to combine LfD and RL and achieve both fast learning and high performance in complex domains. In transfer learning, knowledge from a *source task* is used in a *target task* to speed up learning. Equivalently, knowledge from a source agent is used to speed up learning in a target agent. For instance, knowledge has been successfully transferred between agents that balance different length poles (Selfridge, Sutton, and Barto 1985), that solve a series of mazes (Fernández and Veloso 2006), or that play different soccer tasks (Taylor, Stone, and Liu 2007; Torrey et al. 2005). The key insight of transfer learning is that previous knowledge can be effectively reused, even if the source task and target task are not identical. This results in substantially improved learning times because the agent no longer relies on an uninformed prior.

In this work, we show that we can effectively transfer knowledge from a human to an agent, even when they have different perceptions of state. Our method, *Human-Agent Transfer* (HAT): 1) allows a human teacher to perform a series of demonstrations in a task, 2) uses an existing transfer learning algorithm, *Rule Transfer* (Taylor and Stone 2007), to learn rule-based summaries of the demonstration, and 3) integrates the rule summaries into RL, biasing learning while allowing improvement over the transferred policy.

We perform empirical evaluation of HAT in a simulated robot soccer domain. We compare three algorithms for incorporating rule summaries into reinforcement learning, and compare learning performance for multiple demonstration source, quantity, and quality conditions. Our findings show statistically significant improvement in performance for all variants of HAT over learning with no prior.

Background

This section provides background on the three key techniques discussed in this paper: reinforcement learning, learning from demonstrations, and transfer learning.

Reinforcement Learning

Reinforcement learning is a common approach to agent learning from experience. We define reinforcement learning using the standard notation of Markov decision processes (MDPs). At every time step the agent observes its state $s \in S$ as a vector of k state variables such that $s = \langle x_1, x_2, \ldots, x_k \rangle$. 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 will attempt to maximize the long-term reward determined by the (initially unknown) reward and transition functions.

A learner chooses which action to take in a state via a policy, $\pi : S \mapsto A$. π is modified by the learner over time to improve performance, which is defined as the expected total reward. Instead of learning π directly, many 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. In this paper, agents learn using Sarsa (Rummery and Niranjan 1994; Singh and Sutton 1996), a well known but relatively simple temporal difference RL algorithm, which learns to estimate $Q(s, a)$. While some RL algorithms are more sample efficient than Sarsa, this paper will focus on Sarsa for the sake of clarity.

Learning from Demonstration

Learning from demonstration research explores techniques for learning a policy from examples, or demonstrations, provided by a human teacher. LfD can be seen as a subset of Supervised Learning, in that the agent is presented with labeled training data and learns an approximation to the function which produced the data.

Similar to reinforcement learning, learning from demonstration can be defined in terms of the agent's observed state $s \in S$ and executable actions $a \in A$. Demonstrations are recorded as temporal sequences of t state-action pairs $\{(s_0, a_0), ..., (s_t, a_t)\}$, and these sequences typically only cover a small subset of all possible states in a domain. The agent's goal is to generalize from the demonstrations and learn a policy $\pi : S \mapsto A$ covering all states that imitates the demonstrated behavior.

Many different algorithms for using demonstration data to learn π have been proposed. Approaches vary by how demonstrations are performed (e.g., teleoperation, teacher following, kinesthetic teaching, external observation), the type of policy learning method used (e.g., regression, classification, planning), and assumptions about degree of demonstration noise and teacher interactivity (Argall et al. 2009). Across these differences, LfD techniques possess a number of key strengths. Most significantly, demonstration leverages the vast task knowledge of the human teacher to significantly speed up learning either by eliminating exploration entirely (Grollman and Jenkins 2007; Nicolescu et al. 2008), or by focusing learning on the most relevant areas of the state space (Smart and Kaelbling 2002). Demonstration also provides an intuitive programming interface for humans, opening possibilities for policy development to non-agents-experts.

However, LfD algorithms are inherently limited by the quality of the information provided by the human teacher. Algorithms typically assume the dataset to contain high quality demonstrations performed by an expert. In reality, teacher demonstrations may be ambiguous, unsuccessful, or suboptimal in certain areas of the state space. A naïvely learned policy will likely perform poorly in such areas (Atkeson and Schaal 1997). To enable the agent to improve beyond the performance of the teacher, learning from demonstration must be combined with learning from experience. Most similar to our approach is the work of Smart and Kaelbling, which shows that human demonstration can be used to bootstrap reinforcement learning in domains with sparse rewards by initializing the action-value function using the observed states, actions and rewards (Smart and Kaelbling 2002). In contrast to this approach, our work uses demonstration data to learn generalized rules, which are then used to bias the reinforcement learning process.

Transfer Learning

The insight behind *transfer learning* (TL) is that generalization may occur not only within tasks, but also *across tasks*, allowing an agent to begin learning with an informative prior instead of relying on random exploration.

Transfer learning methods for reinforcement learning can transfer a variety of information between agents. However, many transfer methods restrict what type of learning algorithm is used by both agents (for instance, some methods require temporal difference learning (Taylor, Stone, and Liu 2007) or a particular function approximator (Torrey et al. 2005) to be used in both agents). However, when transferring from a human, it is impossible to copy a human's "value function" — both because the human would likely be incapable of providing a complete and consistent value function, and because the human would quickly grow wary of evaluating a large number of state-action pairs.

This paper uses *Rule Transfer* (Taylor and Stone 2007), a particularly appropriate transfer method that is agnostic to the knowledge representation of the source learner. The ability to transfer knowledge between agents that have different state representations and/or actions is a critical ability when considering transfer of knowledge between a human and an agent. The following steps summarize Rule Transfer:

1a: Learn a policy ($\pi : S \mapsto A$) in the source task.

1b: Generate samples from the learned policy. Record a number of (S, A) pairs while following the learned policy.

2: Learn a decision list ($D_s : S \mapsto A$) that summarizes the source policy.[1]

3: Use D_t to bootstrap learning of an improved policy in the target task.

Additional Related Work

We now briefly summarizes three additional related topics.

The recent work by Knox and Stone (Knox and Stone 2010) combines behavioral shaping with reinforcement learning. Their TAMER (Knox and Stone 2009) system learns to predict and maximize a reward that is interactively provided by a human. The learned human reward is combined in various ways with Sarsa(λ), providing significant improvements. The primary difference between HAT and this method is that we focus on leveraging human demonstration, rather than estimating and integrating a human reinforcement signal.

[1] Additionally, if the agents in the source and target task use different state representations or have different available actions, the decision list can be translated via inter-task mappings (Taylor and Stone 2007; Taylor, Stone, and Liu 2007).

The idea of transfer between a human and an agent is somewhat similar to *implicit imitation* (Price and Boutilier 2003), in that one agent teaches another how to act in a task, but HAT does not require the agents to have the same (or very similar) representations.

High-level advice and suggestions have also been used to bias agent learning. Such advice can provide a powerful learning tool that speeds up learning by biasing the behavior of an agent and reducing the policy search space. However, existing methods typically require either a significant user sophistication (e.g., the human must use a specific programming language to provide advice (Maclin and Shavlik 1996)) or significant effort is needed to design a human interface (e.g., the learning agent must have natural language processing abilities (Kuhlmann et al. 2004)). Allowing a teacher to demonstrate behaviors is preferable in domains where demonstrating a policy is a more natural interaction than providing such high-level advice.

Methodology

In this section we present HAT, our approach to combining LfD and RL. HAT consists of three steps, motivated by those used in Rule Transfer:

Demonstration The agent performs the task under the teleoperated control by a human teacher, or by executing an existing suboptimal controller. During execution, the agent records all state-action transitions. Multiple task executions may be performed.

Policy Summarization HAT uses the state-action transition data recorded during the Demonstration phase to derive rules summarizing the policy. These rules are used to bootstrap autonomous learning.

Independent Learning The agent learns independently in the task via reinforcement learning, using the policy summary to bias its learning. In this step, the agent must balance exploiting the transferred rules with attempting to learn a policy that outperforms the transferred rules.

In contrast to transfer learning, HAT assumes that either 1) the demonstrations are executed on the same agent, in the same task, as will be learned in the Independent Learning phase, or that 2) any differences between the agent or task in the demonstration phase are small enough that they can be ignored in the independent learning phase. Instead of transferring between different tasks, HAT focuses on transferring between different agents with different internal representations. For instance, it is not possible to directly use a human's "value function" inside an agent because 1) the human's knowledge is not directly accessible and 2) the human has a different state abstraction than the agent.

We next present three different ways that HAT can use a decision list to improve independent learning.

Value Bonus

The intuition behind the *Value Bonus* method (Taylor and Stone 2007) is similar to that of shaping in that the summarized policy is used to add a reward bonus to certain human-favored actions. When the agent reaches a state and calculates $Q(s, a)$, the Q-value of the action suggested by the summarized policy is given a constant bonus (B). For the first C episodes, the learner is forced to execute the action suggested by the rule set. This is effectively changing the initialization of the Q-value function, or (equivalently) providing a shaping reward to the state-action pairs that are selected by the rules.

We use $B = 10$ and $C = 100$ to be consistent with past work (Taylor and Stone 2007); the Q-value for the action chosen by the summarized policy will be given a bonus of +10 and agents must execute the action chosen by the summarized policy for the first 100 episodes.

Extra Action

The *Extra Action* method (Taylor and Stone 2007) augments the agent so that it can select a *pseudo-action*. When the agent selects this pseudo-action, it executes the action suggested by the decision list. The agent may either execute the action suggested by the transferred rules, or it can execute one of the "base" MDP actions. Through exploration, the RL agent can decide when it should follow the transferred rules or when it should execute a different action (e.g., the transferred rules are sub-optimal). Were the agent to always execute the pseudo-action, the agent would never learn but would simply mimic the demonstrated policy.

As with the Value Bonus algorithm, the agent initially executes the action suggested by the decision list, allowing it to estimate the value of the decision list policy. We again set this period to be 100 episodes ($C = 100$).

Probabilistic Policy Reuse

The third method used is *Probabilistic Policy Reuse* (PPR), based on the π-reuse Exploration Strategy (Fernández, García, and Veloso 2010; Fernández and Veloso 2006). In PPR, the agent will reuse a policy with probability ψ, explore with probability ϵ, and exploit the current policy with probability $1 - \psi - \epsilon$. By decaying ψ over time, the agent can initially leverage the decision list, but then learn to improve on it if possible. PPR is similar to the more recent TAMER+RL method #7 (Knox and Stone 2010): the agent tries to execute the action suggested by the learned human shaping reward, rather than follow a transferred policy.

Experimental Validation

This section first discusses Keepaway (Stone, Sutton, and Kuhlmann 2005), a simulated robot soccer domain, and then explains the experimental methodology used for evaluation.

Keepaway

Keepaway is a domain with a continuous state space and significant amounts of noise in the agent's actions and sensors. One team, the *keepers*, attempts to maintain possession of the ball within a 20m × 20m region while another team, the *takers*, attempts to steal the ball or force it out of bounds. The simulator places the players at their initial positions at the start of each episode and ends an episode when the ball leaves the play region or is taken away from the keepers.

The keeper with the ball has the option to either pass the ball to one of its two teammates or to hold the ball. In *3 vs.*

2 Keepaway (3 keepers and 2 takers), the state is defined by 13 hand-selected state variables, as defined in (Stone, Sutton, and Kuhlmann 2005). The reward to the learning algorithm is the number of time steps the ball remains in play after an action is taken. The keepers learn in a constrained policy space: they have the freedom to decide which action to take only when in possession of the ball. Keepers not in possession of the ball are required to execute the `Receive` macro-action in which the player who can reach the ball the fastest goes to the ball and the remaining players follow a handcoded strategy to try to get open for a pass.

For policy learning, the Keepaway problem is mapped onto the discrete-time, episodic RL framework. As a way of incorporating domain knowledge, the learners choose not from the simulator's primitive actions but from a set of higher-level macro-actions implemented as part of the player (Stone, Sutton, and Kuhlmann 2005). These macro-actions can last more than one time step and the keepers have opportunities to make decisions only when an on-going macro-action terminates. Keepers can choose to `Hold` (maintain possession), $\texttt{Pass}_1$ (pass to the closest teammate), and $\texttt{Pass}_2$ (pass to the further teammate). Agents then make decisions at discrete time steps (when macro-actions are initiated and terminated).

To learn Keepaway with Sarsa, each keeper is controlled by a separate agent. Many kinds of function approximation have been successfully used to approximate an action-value function in Keepaway, but a Gaussian Radial Basis Function Approximation (RBF) has been one of the most successful (Stone et al. 2006). All weights in the RBF function approximator are initially set to zero; every initial state-action value is zero and the action-value function is uniform. Experiments in this paper use the public versions 11.1.0 of the RoboCup Soccer Server (Noda et al. 1998), and 0.6 of UT-Austin's Keepaway players (Stone et al. 2006).

Experimental Setup

In the Demonstration phase of HAT, Keepaway players in the simulator are controlled by the teacher using the keyboard. This allows a human to watch the visualization and instruct the keeper with the ball to execute the `Hold`, $\texttt{Pass}_1$, or $\texttt{Pass}_2$ actions. During demonstration, we record all (s, a) pairs selected by the teacher. It is worth noting that the human has a very different representation of the state than the learning agent. Rather than observing a 13 dimensional state vector, the human uses a visualizer of the soccer field. It is therefore critical that whatever method used to glean information about the human's policy does not require the agent and the human to have identical representations of state.

To be consistent with past work (Stone et al. 2006), our Sarsa learners use $\alpha = 0.05$, $\epsilon = 0.10$, and RBF function approximation. After conducting initial experiments with five values of ψ, we found that $\psi = 0.999$ was at least as good as other possible settings. In the Policy Summarization Phase, we use a simple propositional rule learner to generate a decision list summarizing the policy (that is, it learns to generalize which action is selected in every state). For these experiments, we use JRip, as implemented in Weka (Witten and Frank 2005).

Finally, when measuring speedup in RL tasks, there are many possible metrics. In this paper, we measure the success of HAT along three related dimensions. The initial performance of an agent in a target task may be improved by transfer. Such a *jumpstart* (relative to the initial performance of an agent learning without the benefit of any prior information), suggests that transferred information is immediately useful to the agent. In Keepaway, the jumpstart is measured as the average episode reward (corresponding to the average episode length in seconds), averaged over 1,000 episodes without learning. The jumpstart is a particularly important metric when learning is slow and/or expensive.

The *final reward* acquired by the algorithm at the end of the learning process (at 30 simulator hours in this paper) indicates the best performance achieved by the learner. This value is computed by taking the average of the final 1,000 episodes to account for noise in the Keepaway domain.

The *total reward* accumulated by an agent (i.e., the area under the learning curve) may also be improved. This metric measures the ability of the agent to continue to learn after transfer, but is heavily dependent on the length of the experiment. In Keepaway, the total reward is the sum of the average episode durations at every integral hour of training:

$$\sum_{t:0 \rightarrow n} (\text{average episode reward at training hour } t)$$

where the experiment lasts n hours and each average reward is computed by using a sliding window over the past 1,000 episodes.[2]

Empirical Evaluation

This section presents results showing that HAT can effectively use human demonstration to bootstrap RL in Keepaway agents.

To begin, we recorded a demonstration from *Subject A* which lasted for 20 episodes (less than 3 minutes). Next, we used JRip to summarize the policy with a decision list. The following rules were learned, where *state*$_k$ represents the k^{th} state variable, as defined in the keepaway task:

$$\begin{array}{lll} \textit{if} & (state_{11} \geq 74.84 \textit{ and } state_3 \leq 5.99 \textit{ and} & \\ & \quad state_{11} \leq 76.26) & \rightarrow \textit{Action} = 1 \\ \textit{elseif} & (state_{11} \geq 53.97 \textit{ and } state_4 \leq 5.91 \textit{ and} & \\ & \quad state_0 \geq 8.45 \textit{ and } state_8 \leq 7.06) & \rightarrow \textit{Action} = 1 \\ \textit{elseif} & (state_3 \leq 4.84 \textit{ and } state_0 \geq 7.33 \textit{ and} & \\ & \quad state_{12} \geq 43.66 \textit{ and } state_8 \leq 5.57) & \rightarrow \textit{Action} = 2 \\ \textit{else} & & \rightarrow \textit{Action} = 0 \end{array}$$

While not the focus of this work, we found it interesting that the policy was able to be summarized with only four rules, obtaining over 87% accuracy on when using stratified cross-validation.

[2] Recall that the reward in Keepaway is +1 per time step, or $\frac{1}{10}$ of a simulator second. Thus, the reward for the first hour of training is always $60 \times 60 \times 10 = 36000$ — a metric for the total reward over time must account for the reward *per episode* and simply summing the total amount of reward accrued is not appropriate.

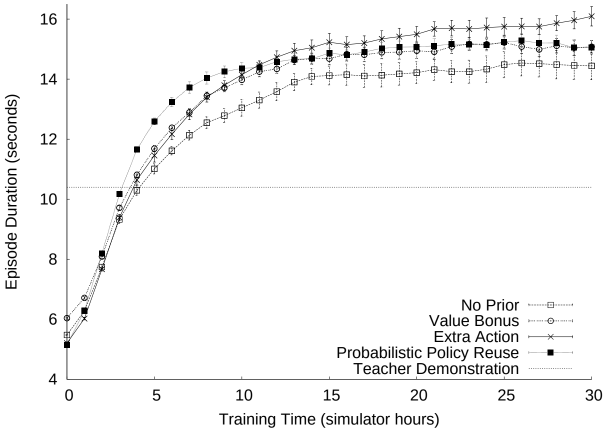

Figure 1: This graph summarizes performance of Sarsa learning in Keepaway using four different algorithms. One demonstration of 20 episodes was used for all three HAT learners. Error bars show the standard error in the performance.

Method	Jumpstart	Final	Total Reward
No Prior	N/A	14.3	380
Value Bonus	**0.57**	15.1	**401**
Extra Action	**-0.29**	**16.0**	**407**
PPR	**-0.30**	15.2	**411**

Table 1: This table shows the jumpstart, final reward and total reward metrics for Figure 1. Values in **bold** have statistically significant differences in comparison to the No Prior method ($p < 0.05$).

Finally, agents are trained in 3 vs. 2 Keepaway without using transfer rules (No Prior), using the Value Bonus, using the Extra Action, or using the PPR method. All learning algorithms were executed for 30 simulator hours (processor running time of roughly 2.5 hours) to ensure convergence.

Figure 1 compares the performance of the four methods, averaged over 10 independent trials. Using 20 episodes of transferred data from *Subject A* with HAT can improve the jumpstart, the final reward, and the cumulative reward. The horizontal line in the figure shows the average duration of the teacher's demonstration episodes; all four of the RL-based learning methods improve upon and outperform the human teacher. The performance of the different algorithms is measured quantitatively in Table 1, where significance is tested with a Student's t-test.

While the final reward performance of the all four methods is very similar (only Extra Action has a statistically significant improvement over No Prior), the total reward accumulated by all three algorithms is significantly higher than with No Prior learning. This result is an indication that although the same final performance is achieved in the long term because the learning algorithm is able to learn the task in all cases, high performance is achieved *faster* by using a small number of demonstrations. This difference can be best observed by selecting an arbitrary threshold of episode duration and comparing the number of simulation hours each algorithm takes to achieve this performance. In the case of a threshold of 14 seconds, we see that No Prior learning takes 13.5 hours, compared to 10.1, 8.57 and 7.9 hours for Value Bonus, Extra Action, and PPR, respectively. These results show that transferring information via HAT from the human results in significant improvements to learning.

The following sections we will explore how performance changes with different types or amounts of demonstration. In all further experiments we use the PPR method as it was not dominated by either of the other two methods. Additionally, in some trials with other methods we found that the learner could start with a high jumpstart but fail to improve as much as other trials. We posit this is due to becoming stuck in a local minimum. However, because ψ explicitly decays the effect from the rules, this phenomena was never observed when using PPR.

Comparison of Different Teachers

Above, we used a single demonstration data set to evaluate and compare three algorithms for incorporating learned rules into reinforcement learning. In this section, we examine how demonstrations from different people impact learning performance of a single algorithm, PPR. Specifically, we compare three different teachers:

1. *Subject A* has many years of research experience with the Keepaway task. (The same as Figure 1.)
2. *Subject B*: is new to Keepaway, but practiced for approximately 100 games before recording demonstrations.
3. *Subject C*: is an expert in LfD, but is new to Keepaway, practicing only 10 games.

Each teacher recorded 20 demonstration episodes while trying to play Keepaway to the best of their ability. Figure 2 summarizes the results and compares performance of using these three demonstration sets against learning the Keepaway task without a prior. All reported results are averaged over 10 learning trials. Table 2 presents summary of the results, highlighting statistically significant changes in bold.

All three HAT experiments outperformed learning without a bias from demonstration, with statistically significant improvements in total reward. However, as in any game, different Keepaway players have different strategies. While some prefer to keep the ball in one location as long as possible, others pass frequently between keepers. As a result, demonstrations from three different teachers led to different learning curves. Demonstration data from *Subjects A* and *C* resulted in a low jumpstart, while *Subject B*'s demonstration gave the learner a significant jumpstart early in the learning process. The final reward also increased for all three HAT trials, with statistically significant results in the case of *Subjects B* and *C*. These results indicate that HAT is robust to demonstrations from different people with varying degrees of task expertise.

Future Work and Conclusion

This paper empirically evaluates HAT in the Keepaway domain, showing that just a few minutes of human demonstration can increase the learning rate of the task by several simulation hours. We evaluated three different variants which used different methods to bias learning with the human's demonstration. All three methods performed statistically significantly better than learning without demonstra-

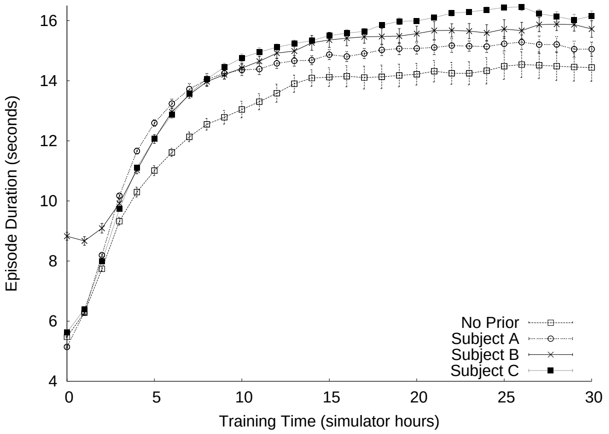

Figure 2: This graph summarizes performance of no prior learning and Probabilistic Policy Reuse learning using demonstrations from three different teachers. Each teacher demonstrated for 20 episodes; the standard error over 10 trials is shown.

Method	Jumpstart	Final	Total Reward
No Prior	N/A	14.3	380
Subject A	**-0.30**	15.2	**411**
Subject B	**3.35**	**15.7**	**423**
Subject C	**0.15**	**16.2**	**424**

Table 2: This table shows the jumpstart, final reward and total reward metrics for Figure 2, where all HAT methods use Probabilistic Policy Reuse with 20 episodes of demonstrated play. Values in **bold** have statistically significant differences in comparison to the No Prior method.

tion. Probabilistic Policy Reuse consistently performed at least as well as the other methods, likely because it explicitly balances exploiting the human's demonstration, exploring, and exploiting the learned policy. Additional evaluation using demonstrations from different teachers.

One of the key strengths of this approach is its robustness. It is able to take data of good or poor quality and use it well without negative effects. This is very important when learning from humans because it can naturally handle the noisy, suboptimal data that usually occurs with human demonstration, allowing non-expert users to successfully train agents.

In order to better understand HAT and possible variants, the following questions should be explored in future work:

- Can we identify the characteristics that make some sets of demonstrations lead to better learning performance?
- Rather than performing 1-shot transfer, could HAT be extended so that the learning agent and teacher could iterate between RL and providing demonstrations?
- In this work, the human teacher and the learning agent had different representations of state. Will HAT still be useful if the teacher and agent are performing different tasks? How similar does the demonstrated task need to be to the autonomous learning task for HAT to be effective?

Acknowledgements

The authors would like to thank Shivaram Kalyanakrishnan for sharing his code to allow a human to control the keepers via keyboard input and W. Bradley Knox for useful comments and suggestions.

References

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

Atkeson, C. G., and Schaal, S. 1997. Robot learning from demonstration. In *ICML*.

Fernández, F., and Veloso, M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *AAMAS*.

Fernández, F.; García, J.; and Veloso, M. 2010. Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems* 58(7):866–871.

Grollman, D. H., and Jenkins, O. C. 2007. Dogged learning for robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

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

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

Kuhlmann, G.; Stone, P.; Mooney, R. J.; and Shavlik, J. W. 2004. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *Proceedings of the AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*.

Maclin, R., and Shavlik, J. W. 1996. Creating advice-taking reinforcement learners. *Machine Learning* 22(1-3):251–281.

Nicolescu, M.; Jenkins, O.; Olenderski, A.; and Fritzinger, E. 2008. Learning behavior fusion from demonstration. *Interaction Studies* 9(2):319–352.

Noda, I.; Matsubara, H.; Hiraki, K.; and Frank, I. 1998. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence* 12:233–250.

Price, B., and Boutilier, C. 2003. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research* 19:569–629.

Rummery, G., and Niranjan, M. 1994. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG-RT 116, Engineering Department, Cambridge University.

Selfridge, O. G.; Sutton, R. S.; and Barto, A. G. 1985. Training and tracking in robotics. In *IJCAI*.

Singh, S., and Sutton, R. S. 1996. Reinforcement learning with replacing eligibility traces. *Machine Learning* 22:123–158.

Smart, W. D., and Kaelbling, L. P. 2002. Effective reinforcement learning for mobile robots. In *ICRA*.

Stone, P.; Kuhlmann, G.; Taylor, M. E.; and Liu, Y. 2006. Keepaway soccer: From machine learning testbed to benchmark. In Noda, I.; Jacoff, A.; Bredenfeld, A.; and Takahashi, Y., eds., *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020.

Stone, P.; Sutton, R. S.; and Kuhlmann, G. 2005. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior* 13(3):165–188.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to Reinforcement Learning*. MIT Press.

Taylor, M. E., and Stone, P. 2007. Cross-domain transfer for reinforcement learning. In *ICML*.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(1):1633–1685.

Taylor, M. E.; Stone, P.; and Liu, Y. 2007. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research* 8(1):2125–2167.

Torrey, L.; Walker, T.; Shavlik, J. W.; and Maclin, R. 2005. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *ECML*.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.