In *The AAMAS workshop on Agents Learning Interactively from Human Teachers* (ALIHT-10), Toronto, Canada, May 2010.

Integrating Human Demonstration and Reinforcement Learning: Initial Results in Human-Agent Transfer

Matthew E. Taylor
University of Southern California
Los Angeles, CA
taylorm@usc.edu

Sonia Chernova
MIT Media Laboratory
Cambridge, MA
chernova@media.mit.edu

ABSTRACT

This work introduces *Human-Agent Transfer* (HAT), a method 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 can be transferred into a baseline policy for an agent, and reinforcement learning can be used to significantly improve policy performance. These results are an important initial step that suggest that agents can not only quickly learn to mimic human actions, but that they can also learn to surpass the abilities of the teacher.

1. 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) [35]. 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 most real-world domains.

On the other end of the spectrum are *learning from demonstration* (LfD) algorithms [1, 8, 10, 11, 17, 21]. These approaches leverage the vast experience and task knowledge of a person to enable fast learning, which is critical in real-world applications. However, the final policy performance achieved by these methods is limited by the quality of the dataset and the performance of the teacher. Human teachers provide especially noisy and suboptimal data due to differences in embodiment (e.g., degrees of freedom, action speed, precision) and limitations of human ability.

This paper proposes a novel approach: use RL *transfer learning* methods [38] to combine LfD and RL and achieve fast learning and high performance in a complex domain. 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 [29], that solve a series of mazes [9, 45], or that play different soccer tasks [39, 41, 42]. 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 (arbitrary) 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 (or agent) teacher to perform a series of demonstrations in a domain, 2) uses an existing transfer learning algorithm, *Rule Transfer* [36] to bias learning in an agent, and 3) allows the agent to improve upon the transferred policy using RL. HAT is empirically evaluated in a simulated robot soccer domain and the results serve as a positive proof of concept.

2. BACKGROUND

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

2.1 Reinforcement Learning

A common approach for an agent to learning from experience is reinforcement learning (RL). We define reinforcement learning using the standard notation of Markov decision processes (MDPs) [25]. 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 [26, 30], 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.

Although RL approaches have enjoyed multiple past successes (e.g., TDGammon [40], inverted Helicopter control [19], and agent locomotion [27]), they frequently take substantial amounts of data to learn a reasonable control policy. In many domains, collecting such data may be slow, expensive, or infeasible, motivating the need for ways of making RL algorithms more sample-efficient.

2.2 Learning from Demonstration

Learning from demonstration (LfD) is a growing area of machine learning research that explores techniques for learning a policy from examples, or demonstrations, provided by a human teacher. We define demonstrations as sequences of state-action pairs that are recorded by the agent while the teacher executes the desired behavior. LfD algorithms utilize this dataset of examples to derive a policy that reproduces the demonstrated behavior. Compared to exploration-based methods, such as reinforcement learning, demonstration learning reduces the learning time and eliminates the frequently difficult task of defining a detailed reward function [2, 32].

The field of learning from demonstration is broadly defined and many different algorithms have been proposed within its scope [1]. Approaches vary based on how demonstrations are performed (e.g., teleoperation [6, 10, 20], teacher following [21], kinesthetic teaching [12], external observation [17, 23]), and the type of policy learning method used (e.g., regression [5, 10], classification [6, 28], or planning [43, 44]).

Demonstration learning algorithms have been highly effective for real-world agent systems. 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 times either by eliminating exploration entirely [10, 20], or by focusing learning on the most relevant areas of the state space [32]. Demonstration also provides an intuitive programming interface for humans, opening possibilities for policy development to non-agents-experts.

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, however, 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 [2]. To enable the agent to improve beyond the performance of the teacher, learning from demonstration must be combined with learning from experience. Previous work by Smart and Kaelbling showed that human demonstration can be used to bootstrap reinforcement learning in domains with sparse rewards [32]. However, their evaluation was performed in relatively simple domains with small feature spaces, and the cost of refining the policy using RL in more complex domains has not been previously addressed.

2.3 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 agents 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 [39] or a particular function approximator [42] 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* [36], 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. Any type of reinforcement learning algorithm may be used.

1b: Generate samples from the learned policy After training has finished, or during the final training episodes, the agent records some number of interactions with the environment in the form of (S, A) pairs while following the learned policy.

2: Learn a decision list ($D_s : S \mapsto A$) that summarizes the source policy. After the data is collected, a propositional rule learner is used to summarize the collected data to approximate the learned policy.[1] This decision list is used as a type of inter-lingua, allowing the following step to be independent of the type of policy learned (step 1a).

3: Use D_t to bootstrap learning of an improved policy in the target task. Our previous work [36] has suggested that providing the agent a pseudo-action, which when selected always executes the action suggested by the decision list, is an effective method for allowing the agent to both exploit the transferred knowledge, as well as learn when to ignore the knowledge (by selecting one of the base actions in the MDP).

2.4 Additional Related Work

Learning from demonstration and transfer learning work has been discussed earlier. This section briefly summarizes three additional lines of related work.

Within psychology, *behavioral shaping* [31] is a training procedure that uses reinforcement to condition the desired behavior in a human or animal. During training, the reward signal is initially used to reinforce any tendency towards the correct behavior, but is gradually changed to reward successively more difficult elements of the task. Shaping methods with human-controlled rewards have been successfully demonstrated in a variety of software agent applications [3, 13]. In contrast to shaping, LfD allows a human to demonstrate complete behaviors, which may contain much more information than simple positive/negative rewards.

Most similar to our approach is the recent work by Knox and Stone [15] which combines shaping with reinforcement learning. Their TAMER [14] 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 human reinforcement.

Transfer learning problems are typically framed as leveraging knowledge learned on a source task to improve learning on a related, but different, target task. *Representation transfer* [37] examines the complimentary task of transferring knowledge between agents with different internal representations (i.e., the function approximator or learning algorithm) of the *same* task. This idea is somewhat similar to *implicit imitation* [24], in that one agent teaches another how to act in a task. Allowing for such shifts in representation gives additional flexibility to an agent designer; past experience may be transferred rather than discarded if a new representation is desired. Representation transfer is similar in spirit to HAT in that both the teacher and the learner function in the same task, but very different techniques are used since the human's "value function" cannot be directly examined.

High-level advice and suggestions have also been used to bias agent learning. Such advice can provide a powerful learning technique that speeds up learning by moulding the behavior of an agent

[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 [36, 39] (as step 2b). For the current paper, this translation is not necessary, as the source and target agents operate in the same domain.

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 [18]) or significant effort is needed to design a human interface (e.g., the learning agent must have natural language processing abilities [16]). 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.

3. METHODOLOGY

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

Phase 1: Source 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 (similar to rule transfer's step 1b).

Phase 2: Policy Transfer HAT uses the state-action transition data recorded during Phase 1 to derive rules summarizing the policy (similar to rule transfer step 2). These rules are used to bootstrap autonomous learning.

Phase 3: Independent Learning The agent learns independently in the task via reinforcement learning, using the transferred policy to bias its learning (similar to rule transfer step 3). In this phase, the agent initially executes actions based solely on the transferred rules so that it learns the value of the transferred policy. After this initial training, the agent is allowed to either execute the action suggested by the transfered rules, or it can execute one of the 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 transfered rules are sub-optimal).

4. EXPERIMENTAL VALIDATION

This section first discusses Keepaway [34], a simulated robot soccer domain, explains the experimental methodology used to test HAT, and then reports on results in this domain that confirm the efficacy of our method.

4.1 Keepaway

In this section we discuss *Keepaway*, 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 $\times$ 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 (see Figure 1) as defined elsewhere [34]. 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

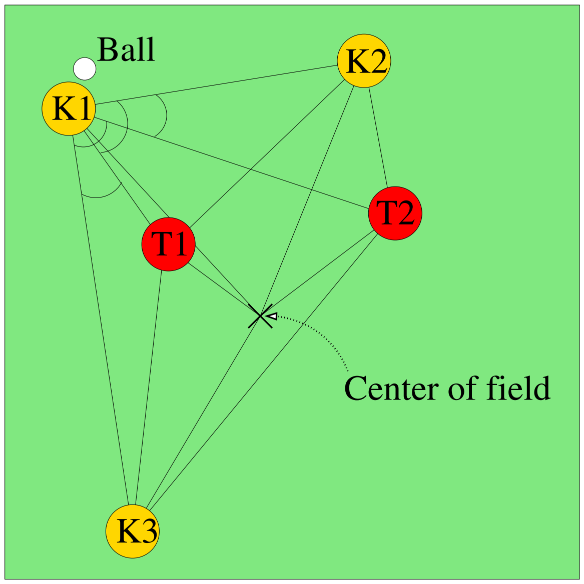

Figure 1: This diagram shows the distances and angles used to construct the 13 state variables used for learning with 3 keepers and 2 takers. Relevant objects are the 3 keepers (K) and the two takers (T), both ordered by distance from the ball, and the center of the field.

and the remaining players follow a handcoded strategy to try to get open for a pass.

The Keepaway problem maps fairly directly 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 [34]. 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. The macro-actions (`Hold`, `Pass`$_1$, and `Pass`$_2$ in 3 vs. 2) that the learners select among can last more than one time step, and the keepers have opportunities to make decisions only when an on-going macro-action terminates. To handle such situations, it is convenient to treat the problem as a *semi-Markov decision process*, or SMDP [4, 25], where agents reason over multi-step macro actions. 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 [33]. 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 version 9.4.5 of the RoboCup Soccer Server [22], and version 0.6 of UT-Austin's Keepaway players [33].

4.2 Experimental Setup

When measuring speedup in RL tasks, there are many possible metrics. In this paper, we measure the success of HAT along two (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 *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 dependant 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 (to help combat the high noise in the Keepaway domain).

In this work, we consider two types of policies which can bootstrap learning (Phase 1 of HAT).

1. Previous work [34] defined a policy that was hand-tuned to play 3 vs. 2 Keepaway. This static policy performs significantly better than allowing the keepers to select actions randomly, but players that learn can surpass its performance.

2. In the simulator, Keepaway players can be controlled by 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.

In experiments, we record all (s, a) pairs selected by the hand-tuned policy and from a human's control. It is worth noting that while the hand-tuned policy uses the same state variables (i.e., representation of state) that the target task learning agent uses, the human has a very different representation. Rather than observing a 13 dimensional state vector, the human uses a visualizer (Figure 2), which contains more detailed information. This additional information may or may not be useful for executing a high-performing policy, but it is 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 evaluate HAT, we compare the outcome from four distinct experiments.

1. "No Prior": The agent learns the task using Sarsa with an uninitialized (arbitrary) Q-value function.
2. "20 Episodes: Hand-coded Policy": Allow the hand-coded agent to demonstrate its policy for 20 episodes, transfer this information to the target task agents, and then continue learning with Sarsa.
3. "10 Episodes: Human Training": Allow a human to demonstrate a policy for 10 episodes, transfer this information to the target task agents, and then continue learning with Sarsa.
4. "18 Episodes: Human Training": Allow a human to demonstrate a policy for 18 episodes, transfer this information to the target task agents, and then continue learning with Sarsa.

It is worth noting that while keepaway learning trials are measured in simulator hours, the above three demonstration periods are significantly shorter. For example, it takes less than three simulator minutes for the hand-coded policy to demonstrate 10 episodes of 3 vs. 2 Keepaway.

In Phase 2, 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, an implementation of RIPPER [7] included in Weka [46].

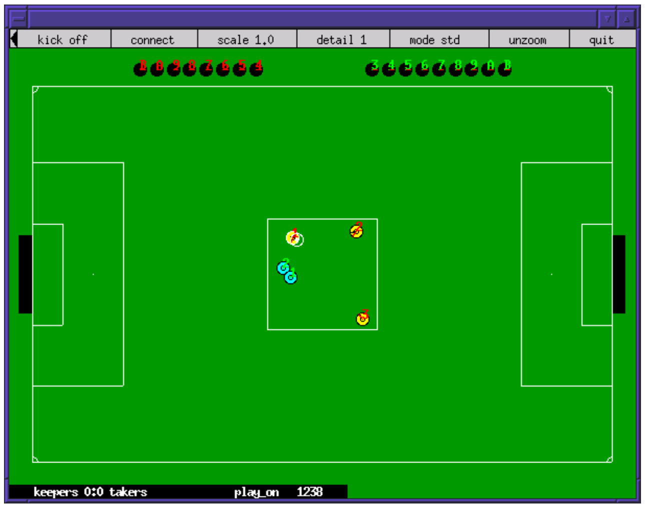

Figure 2: This figure shows a screenshot of the visualizer used for the human to demonstrate a policy in 3 vs. 2 Keepaway. The human controls the keeper with the ball (shown as a hollow white circle) by telling the agent when, and to whom, to pass. When no input is received, the keeper with the ball executes the `Hold` action, attempting to maintain possession of the ball.

In Phase 3, this decision list is loaded by all three keepers, after which they learn and act independently. The decision list is treated as a pseudo-action [36], which the agent may select, and then execute the action indicated by the decision list. For the first 100 episodes, all keepers are forced to execute this pseudo-action, attempting to mimic the policy demonstrated in Phase 1. During these 100 episodes, the keepers learn the value of the transferred decision list.

After the first 100 episodes, keepers can select from the three MDP-level actions (`Hold`, $\texttt{Pass}_1$, or $\texttt{Pass}_2$ actions) and the pseudo-action, which executes the action suggested by the decision list for the current state. The agent is free to explore (using ϵ-greedy exploration), allowing it to discover the value of executing actions that disagree with the transferred decision list. Specifically, over time, the agent learns to execute actions in areas of the state space that differ from that suggested by the decision list when the demonstrated policy's actions are sub-optimal. (Were the agent to always execute the pseudo-action, the agent would never learn but would simply mimic the policy demonstrated in Phase 1.)

4.3 Experimental Results

This section presents preliminary results showing that HAT is effective by using demonstration and Rule Transfer to bootstrap RL in Keepaway agents.

Figure 3 compares the performance of the four experimental settings discussed above. Each experiment was run five times and the performance was analyzed at every hour, using a 1,000 episode sliding window. For readability only one line per experiment is shown, the average of the five trials, and error bars show the standard error of the five trials. The "No Prior" line shows the performance of agents learning without the benefit of transfer. The other three lines show the performance of HAT after demonstration by a hand-coded policy and by a human. Table 1 compares the four experiments according to their jumpstart and total reward. A Student's t-test suggests[2] that all jumpstarts are statistically significant ($p < 0.05$), relative to learning with no prior. Note that a jumpstart

[2]Note that 5 trials is not sufficient for the normality assumption

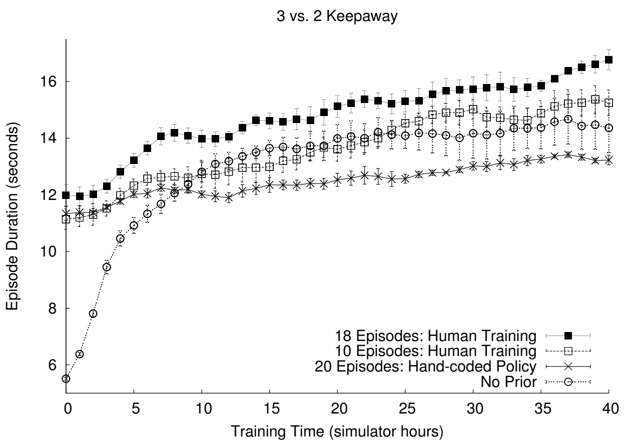

Figure 3: This graph summarizes performance of Sarsa learning in Keepaway using four different settings, averaged over five trials each. Error bars show the standard error in the performance.

Table 1: This table shows the jumpstart and total reward metrics for 3 vs. 2 Keepaway.

Method	Jumpstart	Total Reward
No Prior	0	531
20 Episodes, Hand-coded	5.8	512
10 Episodes, Human	5.6	559
18 Episodes, Human	6.5	606

of roughly six seconds is also "practically significant," as policies learned from scratch reach an average possession time of 14 seconds per episode after training. Only the difference in total reward between no prior and 18 episodes of human training is statistically significant ($p < 0.05$).

These results show that the "No Prior" agents initially perform very poorly, learning a reasonable control policy only after spending significant amounts of time exploring the environment. In all three cases, HAT is able to improve the jumpstart of the learner. This shows that the demonstrated policy is indeed useful to the agent during initial learning. Such a result is particularly important when training is slow and/or expensive — for the first 9 simulator hours, the HAT keepers dominate the keepers learning with an uninformed prior.

In terms of both the jumpstart and total reward, human demonstrations were more useful than the hand-coded policy, likely because the human was able to achieve higher performance using keyboard control than the hand-coded policy. Using more human demonstrations achieved higher performance, likely because the extra data allowed the decision list learned in Phase 2 to more accurately approximate the demonstrated policy.

Transferring information via HAT from both the hand-coded policy and the human results in significant improvements over learning without prior knowledge.

5. FUTURE WORK AND CONCLUSION

This paper has introduced HAT, a novel method to combine learning from demonstration with reinforcement learning by leveraging an existing transfer learning algorithm. Initial empirical results in the Keepaway domain have shown that HAT can improve learning by using demonstrations generated by a hand-coded policy or by a human.

In order to better understand HAT and possible variants, future work will address the following questions:

- Why do keepers have trouble improving their performance after using HAT to learn from hand-coded policy demonstrations? Is this a statistical quirk, is the hand-coded policy near a local maximum, or is there another (currently unknown) effect at work?
- How does the quality of the demonstration affect learning?
- How does the quantity or state space coverage of demonstrations affect learning?
- Rather than performing 1-shot transfer, could HAT be extended so that the learning agent and teacher could iterate between learning autonomously and providing additional demonstrations?
- Are there other transfer techniques that would better allow an agent to learn from a recorded demonstration?
- 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 have different actions? How similar do the source task and target tasks need to be for effective learning improvement?
- Is using a pseudo-action efficient? Previous work [36] suggested that using the pseudo-action was superior to a set of possible transfer learning variants, but this should be re-investigated in the context of human-agent transfer.
- Could we combine these techniques with inverse reinforcement learning? For instance, it could be that the human is maximizing a different reward function, which accounts in part for the human's higher performance.

Acknowledgements

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

6. REFERENCES

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

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

[3] B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M. P. Johnson, and B. Tomlinson. Integrated learning for interactive synthetic characters. *ACM Trans. Graph.*, 21(3):417–426, 2002.

[4] S. J. Bradtke and M. O. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In *NIPS*, 1995

[5] B. Browning, L. Xu, and M. Veloso. Skill acquisition and use for a dynamically-balancing soccer robot. In *AAAI*, 2004.

[6] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Inelligence Research*, 34(1):1–25, 2009.

[7] W. W. Cohen. Fast effective rule induction. In *ICML*, 1995.

used by a t-test and technically a more sophisticated statistical test should be used.

[8] Y. Demiris and A. Billard. *Special Issue on Robot Learning by Observation, Demonstration and Imitation*. IEEE Transaction on Systems, Man and Cybernetics, 2006.

[9] F. Fernandez and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *AAMAS*, 2006.

[10] D. H. Grollman and O. C. Jenkins. Dogged learning for robots. In *ICRA*, 2007.

[11] D. H. Grollman and O. C. Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *ICRA*, 2008.

[12] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6):1463–1467, Dec. 2008.

[13] F. Kaplan, P.-Y. Oudeyer, E. Kubinyi, and A. Miklosi. Robotic clicker training. In *Robotics and Autonomous Systems*, 38(3-4):197 – 206, 2002.

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

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

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

[17] A. Lockerd and C. Breazeal. Tutelage and socially guided robot learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

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

[19] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.

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

[21] M. N. Nicolescu and M. J. Mataric. Methods for robot task learning: Demonstrations, generalization and practice. In *AAMAS*, 2003.

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

[23] N. Pollard and J. K. Hodgins. Generalizing demonstrated manipulation tasks. In *Workshop on the Algorithmic Foundations of Robotics*, 2002.

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

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

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

[27] M. Saggar, T. D'Silva, N. Kohl, and P. Stone. Autonomous learning of stable quadruped locomotion. In *RoboCup-2006: Robot Soccer World Cup X*, 2007.

[28] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006.

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

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

[31] B. F. Skinner. *Science and Human Behavior*. Colliler-Macmillian, 1953.

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

[33] P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu. Keepaway soccer: From machine learning testbed to benchmark. In *RoboCup-2005: Robot Soccer World Cup IX*, 2006.

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

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

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

[37] M. E. Taylor and P. Stone. Representation transfer for reinforcement learning. In *AAAI 2007 Fall Symposium on Computational Approaches to Representation Change during Learning and Development*, 2007.

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

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

[40] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[41] L. Torrey, J. W. Shavlik, T. Walker, and R. Maclin. Relational macros for transfer in reinforcement learning. In *ILP*, 2007.

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

[43] M. van Lent and J. E. Laird. Learning procedural knowledge through observation. In *K-CAP*, 2001.

[44] H. Veeraraghavan and M. Veloso. Learning task specific plans through sound and visually interpretable demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2599–2604, Sept. 2008.

[45] A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *ICML*, 2007.

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