# Towards Reinforcement Learning Representation Transfer

Matthew E. Taylor and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188
{mtaylor, pstone}@cs.utexas.edu

## ABSTRACT

Transfer learning problems are typically framed as leveraging knowledge learned on a source task to improve learning on a related, but different, target task. Current transfer methods are able to successfully transfer knowledge between agents in different reinforcement learning tasks, reducing the time needed to learn the target. However, the complimentary task of *representation transfer*, i.e. transferring knowledge between agents with different internal representations, has not been well explored. The goal in both types of transfer problems is the same: reduce the time needed to learn the target with transfer, relative to learning the target without transfer. This work introduces one such representation transfer algorithm which is implemented in a complex multiagent domain. Experiments demonstrate that transferring the learned knowledge between different representations is both possible and beneficial.

## 1. INTRODUCTION

Transfer learning is typically framed as leveraging knowledge learned on a *source task* to improve learning on a related, but different, *target task*. Past research [1, 4, 5, 11, 15] has demonstrated the possibility of achieving successful transfer between *reinforcement learning* (RL) [14] tasks. In this work we refer to such transfer learning problems as *task transfer*.

A key component of any reinforcement learning algorithm is the underlying *representation* used by the agent for learning (e.g. its function approximator or learning algorithm), and transfer learning approaches generally assume that the agent will use a similar (or even the same) representation to learn the target task as it used to learn the source. However, this assumption may not be necessary or desirable. This paper considers a related but distinct question: is it possible, and desirable, for agents to use different representations in the target and source? This paper defines and provides algorithms for this new problem of *representation transfer* (RT) and contrasts it with the more typical task transfer.

The motivation for transferring knowledge between tasks is clear: it may enable quicker and/or better learning on the target task after having learned on the source. Our primary motivation for RT in this paper is procedural. Suppose an agent has already been training on a source task with a certain learning method and function approximator (FA) but the performance is poor. A different representation could allow the agent to achieve higher performance. If experience is expensive (e.g. wear on the robot, data collection time, or cost of poor decisions) it is preferable to leverage the agent's existing knowledge to improve learning with the new representation and minimize sample complexity.

This paper's main contributions are to introduce representation transfer, to provide a novel RT algorithm, and to empirically demonstrate the efficacy of this algorithm in a complex multiagent RL domain, robot soccer Keepaway [12].


*AAMAS'07,* May 14–18, 2007, Honolulu, Hawai'i, USA.


## 2. RL BACKGROUND

In this work, we consider transfer in reinforcement learning domains. Following standard notation [14], we say that an agent exists in an environment and at any given time is in some state $s \in S$, beginning at $s_{initial}$. An agent's knowledge of the current state of its environment, $s \in S$ is a vector of k *state variables*, so that $s = x_1, x_2, \ldots, x_k$. The agent selects an action from available actions, $a \in A$. The agent then moves to a new state based on the transition function $T : S \times A \mapsto S$ and is given a real-valued reward for reaching the new state $R : S \mapsto \mathbb{R}$. Over time the agent learns a policy, $\pi : S \mapsto A$, to maximize the expected total reward.

A common approach to learning this policy is to first learn an action-value function, $Q : S \times A \mapsto \mathbb{R}$, which predicts the return for available actions from a given state. If the correct action-value function is known, the agent can act optimally by always selecting the actions with the largest Q-value. In large or continuous domains the action-value function must be approximated via some type of FA. One popular method to learning an action-value function is via temporal difference (TD) [14] methods.

## 3. OFFLINE RT

In this section we present an algorithm for addressing RT problems where the source and target representations differ. We define an agent's *representation* as the learning method used, the FA used, and the FA's parameterization. An an example, suppose an agent in the source uses Q-Learning with a neural network FA that has 20 hidden nodes. Our RT algorithm, *Offline RT* (ORT), may be used to transfer between different FAs (e.g. change to a radial basis function FA).

The key insight for ORT is that an agent using a source representation can record some information about its experience using the learned policy. The agent may record $s$, the perceived state; $a$, the action taken; $r$, the immediate reward; and/or $Q(s, a)$, the long-term expected return. Then the agent can learn to mimic this behavior in the target representation without the use of on-line training (i.e. without more interactions with the environment). The agent is



978-81-904262-7-5 (RPS) 

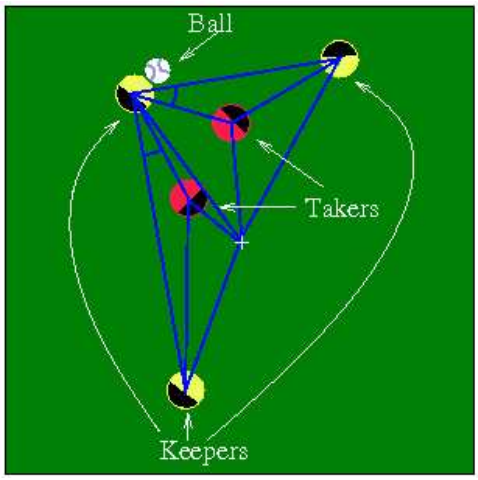


**Figure 1: 3 Keepers play against 2 Takers. A Keeper's state is composed of 11 distances to players and the center of the field as well as 2 angles along passing lanes.**

then able to learn better performance faster than if it had learned the target representation without transfer. This paper will focus on changes in the source and target representation's function approximator and we leave transferring between different learning methods to future work.

Algorithm 1 describes RT for value function methods with different FAs. The agent saves $n$ (state, action, Q-value) tuples and then trains offline with the target representation to predict those saved Q-values, given the corresponding state. Here offline training utilizes a TD update, but the target Q-values are set by the recorded experience.

**Algorithm 1** ORT: Value Functions

1: Train a source representation until reaching a performance or time threshold
2: Record $n$ ($s$, $a$, $q(s_i, a_i)$) tuples while the agent acts
3: **for** all $n$ tuples **do**
4: Train offline with target representation, learning to predict $Q_{target}(s_i, a_i) = q(s_i, a_i)$ for all $a \in A$
5: Train on-line using the target representation

## 4. 3 VS. 2 KEEPAWAY

To test the efficacy of RT we consider the RoboCup simulated soccer Keepaway domain. We use a setup similar to past research [13] and agents based on version 0.6 of the benchmark players distributed[1] by UT-Austin [12]. This section discusses the Keepaway domain and the RL methods used in our experiments.

This multiagent domain has noisy sensors and actuators, and enforces a hidden state so that agents can perceive only a partial world view at any time. In *Keepaway*, one team of *keepers* attempt to possess a ball on a field, while another team of *takers* attempt to steal the ball or force it out of bounds. Keepers that make better decisions about their actions are able to maintain possession of the ball longer and thus have longer average possession *episodes*. Three keepers play against two takers in Figure 1.

[1] http://www.cs.utexas.edu/~AustinVilla/sim/Keepaway/

The agents choose from a set of higher-level macro-actions implemented as part of the player, rather than controlling individual actuators. These macro-actions can last more than one time step, and keepers make decisions only when a macro-action terminates. The macro-actions are Hold Ball, Get Open, Receive, and Pass [13]. A keeper in 3 vs. 2 Keepaway in possession of the ball may choose to either hold the ball or pass it to a teammate: $A$ = {*hold*, *passToTeammate1*, *passToTeammate2*}. Otherwise, keepers execute Receive, the macro-action that causes the keeper who can reach the ball the fastest to approach the ball while the remaining players follow a hand-coded strategy to try to get open for a pass. Takers follow a static hand-coded policy.

In 3 vs. 2 Keepaway, three keepers are initially placed in three corners of a 20m $\times$ 20m field with a ball near one of the keepers. Two takers are located in the fourth corner. When an episode starts, the keepers attempt to control the ball by passing among themselves and moving to open positions. The keeper's world state is defined by 13 variables, as shown in Figure 1. The keepers receive a +1 reward for every time step the ball remains in play. The episode finishes when a taker gains control of the ball or the ball is kicked out of bounds. Further details of the Keepaway domain can be found elsewhere [13].

Experiments presented in this paper use Sarsa [9, 10] with $\epsilon$-greedy exploration. We use a learning rate of 0.05 and an $\epsilon$ of 0.01 to be consistent with past research [12]. We set $\gamma$ to 1 so that the the episodic task is undiscounted [13]. Because the state features are continuous, we use RBFs and neural network function approximators, also consistent with past work [12].

## 5. RESULTS

In this section we present empirical results showing that ORT for Value Functions can improve training in 3 vs. 2 Keepaway between Sarsa agents that utilize neural networks and RBFs.

Learning curves presented in the section each average ten independent trials. The x-axis shows the number of Soccer Server simulator hours and where the simulator can be sped up by roughly a factor of two so that wall clock time is roughly half of the simulator time. the y-axis shows the average performance of the keepers by showing the average episode length in simulator seconds. Error bars show one standard deviation. All parameters chosen in this section were selected via experimentation with a small set of initial test experiments.

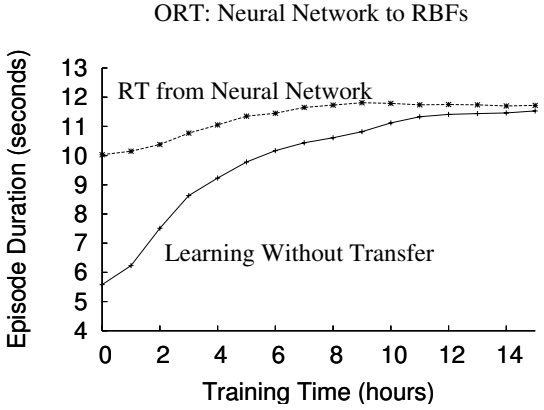


**Figure 2: RBF players utilize ORT from neural networks to outperform RBF players learning without transfer.**

To demonstrate intra-policy transfer, we first train Sarsa players using a neural network on 3 vs. 2 Keepaway for 20 simulator hours and then record 20,000 tuples, which took roughly 1.0 simulator hour. TD-RBF players are then trained offline by iterating over all tuples 5 times and updating $Q(s_i, a)$ where $a \in A$. Using Algorithm 1, this process takes roughly 8 minutes of wall clock time. Figure 5 shows that RT from neural network players outperforms RBF players learning without transfer. T-tests confirm that differences graphed are statistically significant for times less than 11 simulator hours. These results demonstrate that if a neural network player has been trained, ORT can be used to improve the performance of RBF players. By using ORT, both the computational complexity (wall clock time) and sample complexity (simulator time) are significantly reduced. This experiment's compliment, transferring from RBF players to neural network players, yields similar results but is omitted due to space constraints.

## 6. RELATED AND FUTURE WORK

The idea of using multiple representations to solve a problem is not new. Kaplan's production system [3] was able to simulate the representation shift that humans often undergo when solving the *mutilated checkerboard* [7] problem. Other work [2] used libraries of problem solving and "problem description improvement" algorithms to automatically change representations in planning problems. *Implicit imitation* [8] allows an RL agent to train while watching a mentor with similar actions, but this method does not directly address internal representation differences. Additionally, all training is done on-line; therefore agents using imitation do not initially perform better than learning without transfer. Our method of training offline from saved experience is more similar to the idea of *replayed TD* [6], a method to improve the rate of learning by reusing experience in a single agent.

None of these methods directly address the problem of transferring knowledge between different representations in an RL setting. By using RT methods like ORT, different representations can be leveraged so that better performance can be more quickly learned, which could be used in conjunction with existing RL speedup methods.

This paper has focused on changes in the function approximator between the source and target. In the future, we would like to extend this work to situations where the representation differers in learning method (i.e. use an agent that learns with SARSA to speed up a policy search learner). Additionally, we would like to attempt to find methods which are able to perform both representation transfer and task transfer, ideally simultaneously.

## 7. CONCLUSION

This paper presents the problem of representation transfer and an algorithm that successfully transfers knowledge between different internal representations. We have presented empirical results in Keepaway demonstrating that RT allows an agent training with RBF function approximation to learn faster when it utilizes knowledge gathered by an agent with neural network function approximation.

## Acknowledgments

We would like to thank Cynthia Matuszek, Shimon Whiteson, Andrew Dreher, Bryan Klimt, Nate Kohl, and anonymous reviewers for helpful comments and suggestions. This research was supported in part by DARPA grant HR0011-04-1-0035, NSF CAREER award IIS-0237699, and NSF award EIA-0303609.

## 8. REFERENCES

[1] F. Fernandez and M. Veloso. Learning by probabilistic reuse of past policies. In *Proc. of the 6th International Conference on Autonomous Agents and Multiagent Systems*, 2006.

[2] E. Fink. Automatic representation changes in problem solving. Technical Report CMU-CS-99-150, Depart. of Computer Science, Carnegie Mellon University, 1999.

[3] C. A. Kaplan. Switch: A simulation of representational change in the mutilated checkboard problem. Technical Report C.I.P. 477, Department of Psychology, Carnegie Mellon University, 1989.

[4] G. Konidaris and A. Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd Internation Conference on Machine Learning*, pages 489–496, 2006.

[5] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.

[6] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. In *National Conference on Artificial Intelligence*, pages 768–773, 1991.

[7] J. McCarthy. A tough nut for proof procedures. Technical Report Sail AI Memo 16, Computer Science Department, Stanford University, 1964.

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

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

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

[11] V. Soni and S. Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, July 2006.

[12] P. Stone, G. Kuhlmann, M. E. Taylor, and Y. Liu. Keepaway soccer: From machine learning testbed to benchmark. In I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020, pages 93–105. Springer Verlag, Berlin, 2006.

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

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

[15] M. E. Taylor, P. Stone, and Y. Liu. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.