Cross-Domain Transfer for Reinforcement Learning

Matthew E. Taylor MTAYLOR@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Department of Computer Sciences, The University of Texas at Austin

Abstract

A typical goal for transfer learning algorithms is to utilize knowledge gained in a source task to learn a target task faster. Recently introduced transfer methods in reinforcement learning settings have shown considerable promise, but they typically transfer between pairs of very similar tasks. This work introduces *Rule Transfer*, a transfer algorithm that first learns rules to summarize a source task policy and then leverages those rules to learn faster in a target task. This paper demonstrates that Rule Transfer can effectively speed up learning in Keepaway, a benchmark RL problem in the robot soccer domain, based on experience from source tasks in the gridworld domain. We empirically show, through the use of three distinct transfer metrics, that Rule Transfer is effective across these domains.

1. Introduction

Reinforcement learning (RL) methods excel at solving complex tasks with minimal feedback. *Transfer learning*, in an RL setting, typically attempts to decrease training time by learning a *source task* before learning the *target task*. While there have been a number of recent successes, most existing transfer methods focus on pairs of tasks that are closely related, such as different mazes where agents have the same sensors and actions available (Madden & Howley, 2004). Prior to this work, the most dissimilar source and target tasks we are aware of are pairs of tasks in a single domain with different reward structures, different actions, and/or different state variables (see, for instance, past transfer work in the robot soccer domain (Torrey et al., 2006)).

A more difficult challenge is to transfer between different domains, where we informally define a *domain* to be a setting for a group of semantically similar *tasks*. Such cross-domain transfer has been a long-term goal of transfer learning because it could allow transfer between significantly less similar tasks. While previous transfer work has focused on reducing training time by transferring from a simple to complex task in a single domain, a (potentially) more powerful way of simplifying a task is to formulate it as an abstraction in a different domain. This work will focus on demonstrating that cross-domain transfer is feasible and effective, where the source task is selected from the relatively simple gridworld domain and the target task is the more complex RL benchmark task of 3 vs. 2 Keepaway in the robot soccer domain.

We first introduce *Rule Transfer*, a novel domain-independent RL transfer method. In addition to succeeding at cross-domain transfer and having low computational requirements in practice, this method allows production rules (henceforth *rules*) to transfer knowledge between agents which may have different internal representations. Thus an agent may train very quickly with a simple internal representation in the source task, but a more complex agent in the target task could still benefit from the transfer.

This paper evaluates three different rule utilization schemes for Rule Transfer in Keepaway. We then empirically show that cross-domain transfer can effectively improve the speed of learning if the relationships between the tasks is known. Lastly, learning such relationships is an important open problem and this work also takes a step towards learning the mapping, illustrating a process by which it can be derived from high-level qualitative knowledge.

2. Rule Transfer

The following steps summarize Rule Transfer:

1. **Learn a policy ($\pi : S \mapsto A$) in the source task.** 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 rule learner is used to summarize the collected data to approximate the learned policy.
3. **Modify the decision list for use in the the target task** ($\texttt{Translate}(D_s) \rightarrow D_t$)**.** To allow the learned decision list to be applied in a target task that has different state variables and actions from the source task, the decision list must be translated before it can be used.
4. **Use D_t to learn a policy in the target task.** Section 2.2 will discuss how the transferred rules can be used in the target task to speed up learning.

The primary difference between this transfer method and previous work is that we leverage rules to provide an ab-

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

stract representation of a source task policy that is usable in the target task. We choose rules because rule learning is fast and well understood, and they are human readable. By using rules as an intermediate representation, we decouple the particular learning techniques used in the two tasks. Other intermediate representations, such as neural networks, are possible in principle. As long as rules may be abstracted from the source agent's behavior and leveraged by the target agent, agents in the two tasks may use different internal representations, as best suits their particular task.

2.1. Task-Specific Rule Translation

If the target task has different state variables or actions than the source task, an agent could not directly apply a learned decision list from the source task because the preconditions for the rules and/or actions recommended have changed. To define a relationship between two tasks, we define a pair of translation functions for actions and state variables, which we initially assume are provided. δ_A returns the target task action most similar to a source task action ($\delta_A(A_s) = A_t$) and δ_X returns the target task state variable most similar to a source state variable ($\delta_X(x_s) = x_t$). Together, these functions define how a pair of tasks are related.

`Translate()` procedurally modifies the source task decision list so that it can apply to a given target task by directly mapping state variables and actions between the tasks in the spirit of past transfer work (Soni & Singh, 2006; Torrey et al., 2006). In our experiments the source tasks' state variables and actions all had mappings into the target task. If source task state variables or actions had no correspondence in the target task, such preconditions or rules would be removed from the translated decision list.

2.2. Rule Utilization

To make Rule Transfer effective, we treat the translated decision list as *advice* rather as rules that must be followed. The agent benefits from the decision list initially, but refines its policy as it gathers more experience in the target task. This section introduces three advice utilization schemes. The first applies if the target task learner is using a value-function approximation method, like *temporal difference learning* (Sutton & Barto, 1998), but the second and third apply to other RL learning methods in principle.

Value Bonus uses the transferred decision list, D_t, to determine which target task action the decision list would recommend in the current state. The computed Q-value of this recommended action then receives a "bonus" so that it is increased by some constant. Actions recommended by D_t are initially more likely to be selected, but the bonus can be negated over time through learning.

Extra Action adds an action to the target task. When the target task agent selects this pseudo-action, the agent executes the action recommended by D_t. The learner treats this pseudo-action as a normal action. To bias the learner towards this action, the agent is forced to execute the pseudo-action for a constant number of episodes at the beginning of training in the target task. Afterward, assuming pessimistic initialization, the pseudo-action will have higher Q-values than all other actions, which causes the agent to initially perform recommended actions, but over time learning can override this bias. For instance, in regions of the state space in which the advice is appropriate, the agent will learn to select the pseudo-action, while in other regions of the state space where the advice is non-optimal, the agent must learn to intelligently choose between all the actions.

Extra Variable adds an extra state variable to the target task's state description. This variable takes on the value of the index for the action recommended by D_t. To assist the agent in learning the importance of this variable, we again initially force the agent to choose the action recommended by D_t. An agent quickly learns the importance of this state variable, but it can still learn to ignore the state variable when the advice is sub-optimal.

3. Task Descriptions

In this section we first describe 3 vs. 2 Keepaway, an RL benchmark task (Stone et al., 2006) in the domain of robot soccer. Next we introduce Ringworld and Knight Joust, novel tasks in the gridworld domain which are designed to exhibit similarities to Keepaway.

3.1. Keepaway

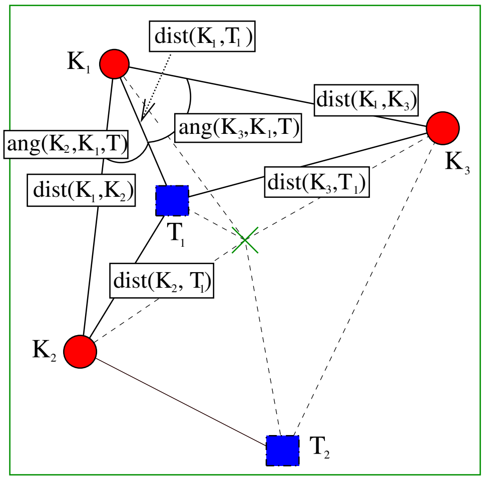

Figure 1. The 13 state variables used for learning 3 vs. 2 Keepaway, 7 of which are labeled with solid lines. There are 11 distances to players and the center of the field, as well as 2 angles along passing lanes.

Keepaway tasks in the RoboCup simulated Soccer domain are characterized by stochastic actions, noisy observations, and a continuous state space. In 3 vs. 2 Keepaway, a team of 3 *keepers* tries to possess a ball in a square area while 2 *takers* work to foil the keepers (see Figure 1). Over time the keepers attempt to learn to possess the ball longer, increasing the average episode length. The players in our experiments are based on version 0.6 of the benchmark players distributed by UT Austin (Stone et al., 2006) and freely-available Soccer Server version 9.4.5.

In 3 vs. 2 Keepaway, three keepers are initially placed in three corners of a 25m $\times$ 25m field and a ball is placed near one of the keepers. Two takers are placed in the fourth corner. When an episode starts, the three keepers attempt to keep control of the ball by passing among themselves and moving to open positions. The keepers receive a reward of +1 for every time step that the ball remains in play. The episode finishes when a taker gains control of the ball or the ball is kicked out of bounds. The episode is then reset with a random keeper placed near the ball.

Keepers can choose from 3 macro-actions when possessing the ball: $A = \{hold, Pass_1, Pass_2\}$. Keepers which do not possess the ball follow a hand-coded policy which attempts to capture an open ball or moves to get open for a pass. Takers follow a fixed strategy and do not learn. The agent's state is defined by 13 variables, as is shown by line segments and angles in Figure 1. These variables describe relevant distance and angles of the keepers $K_1 - K_n$, the takers $T_1 - T_m$, and the center of the playing region, C. Keepers and takers are ordered by increasing distance from the ball. Learning in this multi-agent domain is complicated by a continuous state space and (simulated) noise in agent sensors and actuators.

Keepers learn using Sarsa (Rummery & Niranjan, 1994; Singh & Sutton, 1996) for estimating the action-value function. Because Keepaway has a continuous state space, some kind of function approximation is necessary. We utilized a radial basis function approximator (Sutton & Barto, 1998), as was done previously in this domain (Stone et al., 2006).

3.2. Ringworld

Ringworld is a novel task that is situated on a grid[1] with 0.01 meter tiles. There is no noise in agents' perceptions and the player receives a reward of +1 for every time step it is not captured by the opponent. The opponent always moves towards the player. When the episode starts, the player is randomly assigned two possible "Run Targets," which lie on a static ring. At every timestep, the player may either stay in its current location, or choose to "run" to one of the two targets. If the player runs, it moves at twice the speed of the opponent to the run target. If the opponent does not intercept the player, as determined by the transition function, two new random run targets on the ring are chosen for the player and the episode continues. As the opponent approaches the player, either when the player is standing still or while running, the chance of capture increases. Thus the only stochasticity in the environment is the randomness associated with the probability of capture. The player learns to maximize average episode length by using Sarsa with a Q-value table. $A = \{Stay, Run_{Near}, RunFar\}$ and the state is represented by 5 distances and 2 angles (see Figure 2).

[1] An agent sees an average of only 8065 distinct states over the course of a 25,000 episode learning trial.

This gridworld task was constructed so that it would have similarities to 3 vs. 2 Keepaway. For instance, the 7 state variables were chosen to be similar to the state variables in Keepaway. The width of the ring (9.5 m) was selected so that the distance between runs is similar, on average, to the distance between keepers. The transition function that determines if the episode ends (which takes as input the state variable $dist(P, O)$: *the distance between the player and opponent*) was based on the observed likelihood that a Keepaway episode ends (i.e. the probability that an episode ends given $dist(K_1, T_1)$). While it would be impossible to recreate many of the dynamics associated with a complex, stochastic, and continuous task, we designed Ringworld to capture some of Keepaway's characteristics.

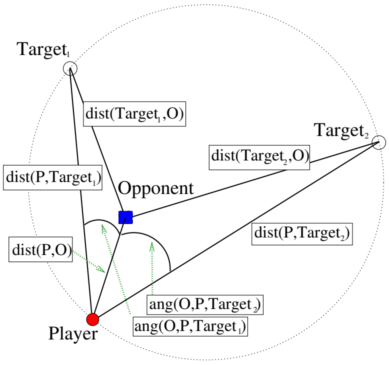

Figure 2. Ringworld: The player may stay still or run to 1 of 2 possible targets. The episode ends when the opponent captures the player. The underlying 0.1m grid is not shown.

3.3. Knight Joust

Knight Joust is also situated in the gridworld domain but was designed to be less similar to Keepaway than Ringworld, and more simple than Ringworld. In this task the player begins on one end of a 25m $\times$ 25m board, the opponent begins on the other, and the players alternate moves. The player's goal is to reach the opposite end of the board without being touched by the opponent (see Figure 3); the episode ends if the player reaches the goal line or the

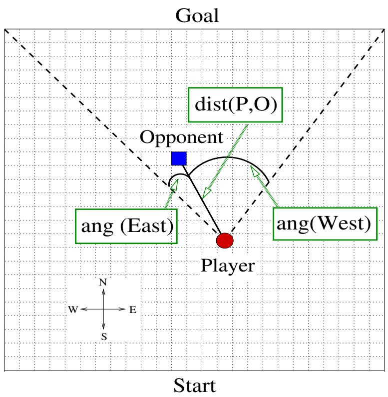

Figure 3. Knight Joust: The player attempts to reach the goal end of a a 25 $\times$ 25 grid-world while the opponent attempts to touch the player.

opponent is on the same square as the player. The state space is discretized into 1m squares[2] and again there is no noise in the perception. The player's state variables are composed of the distance from the player to the opponent, and two angles which describe how much of the goal line is viewable by the player.

if opponent is E of player **then**
  Move W with probability 0.9
else if opponent is W of player **then**
  Move E with probability 0.9
if opponent is N of player **then**
  Move S with probability 1.0
else if opponent is S of player **then**
  Move N with probability 0.8

Figure 4. Knight Joust opponent policy

The player may deterministically move one square North, or perform a knight's jump, where the player moves one square North and two East or West: $A = \{Forward, Jump_{West}, Jump_{East}\}$. The opponent may move in any of 8 directions and follows a fixed stochastic policy[3], shown in Figure 4.

The player receives a reward of +20 every time it takes the forward action, a +20 upon reaching the goal line, and 0 otherwise. The player uses Sarsa with a tabular representation to learn in this task. While this task is quite dissimilar from Keepaway, note that there are some similarities, such as favoring larger distances between player and opponent.

4. Utilizing Rules Effectively

To determine reasonable settings for the different rule utilization methods outlined in Section 2.2, we analyze Rule Transfer by using 3 vs. 2 Keepaway as the source *and* target task. We first train in 3 vs. 2 for five simulator hours (roughly 1,300 episodes). Next, JRip, an implementation of RIPPER (Cohen, 1995) included in Weka (Witten & Frank, 2005), learns a decision list summarizing the source task policy[4]. Lastly, we utilize the decision list in a new instance of Keepaway.

We will compare the different rule utilization methods to learning without transfer via three metrics:

1. Initial Performance: Measure the average hold time at time = 0.
2. Asymptotic Performance: Measure the average hold time after learning has plateaued. We measure this after 40 simulator hours because initial informal experiments showed that learning without transfer has plateaued by this time.
3. Accumulated Reward: Approximate the area under the curve by summing the average reward accumulated by a particular trial at every hour: $\sum_{t=0}^{40}(\text{average reward at time t})$.

Figure 5 shows the performance averaged over 10 learning curves for learning without transfer with a 1000 episode sliding window. One learning curve shows the performance when always utilizing the rules ("Always Use Rules"). Additionally, the three rule utilization methods are shown.

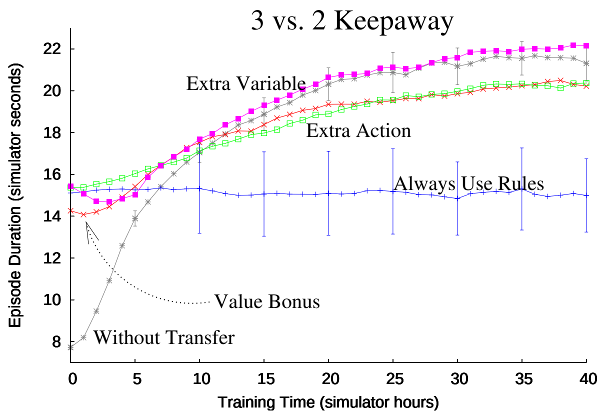

Figure 5. This graph shows the average of 10 independent learning trials for learning without transfer and for using rules after learning for five simulator hours (not shown): without further learning, with a Value Bonus of +10, with Extra Action after 100 episodes, and with Extra Variable after 100 episodes. Learning without transfer and using rules without learning have standard error bars in 5 hour increments.

Table 1 details the results, which show that while the transfer metrics are affected by the relevant rule utilization parameters, each rule utilization method has a wide range of effective parameters.

5. Cross-Domain Transfer Results

In the previous section we determined appropriate advice utilization settings via three transfer metrics. In this section we apply those same settings while using Ringworld and Knight Joust as source tasks and Keepaway as the target. This section demonstrates that transfer from Ringworld is able to significantly improve all three transfer metrics, that Rule Transfer settings are not particularly brittle, and that transfer from Knight Joust is able to significantly improve one of the transfer metrics.

5.1. Ringworld to 3 vs. 2 Keepaway

In this section we first detail how to perform Rule Transfer between Ringworld and Keepaway. We compare the results from using the three different advice utilization schemes and show that Extra Action is superior. Lastly, we perform a set of experiments to show that Rule Transfer, while it has many parameters, is not particularly sensitive to these parameters' settings.

[2]A learner in Knight Joust sees an average of only 601 distinct states over a 50,000 episode learning trial

[3]If the opponent could move in any direction with probability 1.0, the player would never succeed in reaching the goal line. The player can only take a limited number of jumps before hitting the edge of the board and must hope that the opponent "stumbles" so that the player can pass it.

[4]RIPPER is a simple propositional rule learner that can learn a decision list. If additional representational power were needed, an ILP rule learner like Aleph (Srinivasan, 2001) could be used, but we found the additional complexity unnecessary.

Keepaway to Keepaway			
	Initial Performance	Asymptotic Performance	Accumulated Reward
Without Transfer			
	7.0 ± 0.7	19.4 ± 2.0	688.4 ± 68.7
Added Constant	Value Bonus		
1	12.3 ± 1.7	17.1 ± 1.7	630.1 ± 61.3
5	12.7 ± 1.9	18.2 ± 1.8	666.0 ± 66.4
10	13.0 ± 1.8	18.4 ± 2.2	686.4 ± 77.5
20	12.6 ± 1.7	16.6 ± 1.7	611.9 ± 65.3
50	12.8 ± 1.9	13.9 ± 1.9	534.2 ± 73.8
Initial Episodes	Extra Action		
0	7.6 ± 1.7	19.0 ± 2.1	648.2 ± 72.1
50	13.8 ± 2.1	18.3 ± 2.0	676.1 ± 75.4
100	14.0 ± 2.4	18.5 ± 2.0	688.2 ± 75.7
250	13.9 ± 2.3	18.4 ± 1.9	675.3 ± 69.0
500	13.7 ± 2.2	18.1 ± 1.9	678.8 ± 72.2
1000	13.4 ± 2.0	17.9 ± 2.1	648.2 ± 72.1
Initial Episodes	Extra Variable		
0	7.0 ± 0.7	20.0 ± 2.0	691.4 ± 68.8
50	13.6 ± 2.0	19.9 ± 2.0	715.9 ± 70.2
100	14.0 ± 2.3	20.1 ± 2.1	726.0 ± 72.4
250	13.7 ± 2.1	19.9 ± 2.1	717.6 ± 74.6
500	13.6 ± 2.2	20.2 ± 2.0	729.2 ± 73.8
1000	13.7 ± 2.4	17.4 ± 6.3	637.4 ± 207.6

Table 1. Results compare three different rule utilization schemes where each row represents 10 independent tests. The three rule metrics from Section 4 are shown in columns 2-4. The first column contains the constant added to the recommended action in Value Bonus, or the number of episodes the learner in initially forced to select the recommended action for Extra Action and Extra Variable.

Agents learn for 25,000 episodes in Ringworld and then record 20,000 (S, A) pairs, which takes less than 1,000 episodes. After JRip learns a decision list, and the rules are transformed via `Translate()` and the cross-domain mappings (Table 2), the decision list is utilized to learn Keepaway.

Table 3 shows one of the main results of this paper; all three rule utilization methods can significantly increase all three transfer metrics. Furthermore, the asymptotic performance

Cross-Domain Mappings for Ringworld to Keepaway

Ringworld	**Keepaway**
δ_A	
Stay	Hold Ball
Run_{Near}	$Pass_1$: Pass to K_2
Run_{Far}	$Pass_2$: Pass to K_3
δ_X	
$dist(P, O)$	$dist(K_1, T_1)$
$dist(P, Target_1)$	$dist(K_1, K_2)$
$dist(Target_1, O)$	Min($dist(K_2, T_1)$, $dist(K_2, T_2)$)
$ang(O, P, Target_1)$	Min($ang(K_2, K_1, T_1)$ $ang(K_2, K_1, T_2)$)
$dist(P, Target_2)$	$dist(K_1, K_3)$
$dist(Target_2, O)$	Min($dist(K_3, T_1)$, $dist(K_3, T_2)$)
$ang(O, P, Target_2)$	Min($ang(K_3, K_1, T_1)$, $ang(K_3, K_1, T_2)$)

Table 2. This table describes the cross-domain mapping used by `Translate()` to modify a decision list learned in Ringworld so that it can apply to Keepaway.

Ringworld to Keepaway			
	Initial Performance	Asymptotic Performance	Accumulated Reward
Without Transfer			
	7.8 ± 0.1	21.6 ± 0.8	756.7 ± 21.8
Added Constant	Value Bonus		
5	**11.1 ± 1.4**	19.8 ± 0.6	722.3 ± 24.3
10	**11.5 ± 1.7**	**22.2 ± 0.8**	**813.7 ± 23.6**
Initial Episodes	Extra Action		
100	**11.9 ± 1.8**	**23.0 ± 0.5**	**842.0 ± 26.9**
250	**11.8 ± 1.9**	**23.0 ± 0.8**	**827.4 ± 33.0**
Initial Episodes	Extra Variable		
100	**11.8 ± 1.9**	21.9 ± 0.9	**784.8 ± 27.0**
250	**11.7 ± 1.8**	**22.4 ± 0.8**	**793.5 ± 22.2**

Table 3. A comparison of three rule utilization schemes to learning Keepaway without transfer. Each row is the average of 20 independent trials and shows the standard error (note that the top line row uses the same settings as learning without transfer in Table 1 but with more trials). Numbers in **bold** are statistically better than learning without transfer at the 95% level, as determined via a Student's t-test.

IF ((dist(K_1, T_1) $<=$ 4) AND
(Min($dist(K_3, T_1)$, $dist(K_3, T_2)$) $>=$ 12.8) AND
($ang(K_3, K_1, T) >= 36$)) THEN Pass to K_3

Figure 6. An example transformed rule from Ringworld that would be difficult for a human to generate from domain knowledge alone.

is not adversely effected by Rule Transfer for the best parameter settings. These results show that the Extra Action rule utilization method is slightly superior to the other two methods and confirm that cross-domain transfer can be effective at increasing the speed of learning in Keepaway. Figure 7 shows learning in Keepaway without transfer and when using Extra Action Rule Transfer from Ringworld.

An example of the type of knowledge transferred from Ringworld to Keepaway, consider the transformed rule in Figure 6 from one trial. This rule demonstrates that the agent has learned that it should pass if a taker is close, there isn't a taker very close to the target teammate, and the passing angle to the teammate is open.

To further determine the robustness of transfer from Ringworld to Keepaway, we perform a series of additional studies, shown in Table 4, to determine the sensitivity of Rule Transfer to various parameter settings. First we try learning for different amounts of time in Ringworld. At 20,000 episodes the learning in Ringworld has not plateaued and it is not surprising that the initial performance in Keepaway is therefore slightly reduced. Different ring diameters make the task less similar to Keepaway, but all diameters do successfully improve one or more transfer metrics relative to learning without transfer.

We also performed experiments that examined the robustness of rule learning for transfer. In the first experiment we recorded different amounts of Ringworld data; less

Ringworld Sensitivity Analysis

Param	Initial Performance	Asymptotic Performance	Accumulated Reward
Episodes of Ringworld Training before Recording Data			
20,000	10.1 ± 1.7	21.8 ± 1.3	762.5 ± 44.1
25,000	11.9 ± 1.8	23.0 ± 0.5	842.0 ± 26.9
30,000	12.0 ± 1.7	20.7 ± 5.0	793.9 ± 47.8
Ringworld's Ring Diameter (m)			
7.5	14.8 ± 2.4	20.0 ± 1.5	748.2 ± 53.6
8.5	13.5 ± 1.5	21.1 ± 1.2	776.8 ± 45.2
9.5	11.9 ± 1.8	23.0 ± 0.5	842.0 ± 26.9
10.5	9.4 ± 1.0	21.5 ± 1.3	757.7 ± 42.4
11.5	8.2 ± 1.3	20.1 ± 1.6	705.0 ± 41.6
Amount of recorded Ringworld Data			
5,000	12.2 ± 1.2	20.6 ± 4.9	765.1 ± 59.4
20,000	11.9 ± 1.8	23.0 ± 0.5	842.0 ± 26.9
40,000	11.2 ± 1.3	21.4 ± 1.3	776.4 ± 46.6
JRip Settings			
N=2, O=2	13.7 ± 1.7	20.9 ± 1.3	767.3 ± 44.3
N=100, O=2	10.7 ± 1.5	21.6 ± 1.2	784.3 ± 49.9
N=100, O=10	11.9 ± 1.8	23.0 ± 0.5	842.0 ± 26.9
N=2, O=10	14.0 ± 1.8	20.9 ± 1.3	763.3 ± 44.7

Table 4. This table shows Ringworld transfer with Extra Action rule usage after forcing the action advised by D_t for 100 episodes. The settings used previously (in Table 3) are shown in bold for comparison, each row is the average over 20 independent trials, and the standard error is shown.

data would force more generalization while more data may cause overfitting. The last sensitivity analysis tried varying the parameters to JRip, again showing that the performance of the 4 metrics is not particularly sensitive to the rule learning settings as they all outperform learning without transfer. N is the minimum number of instances a rule must cover (JRip default = 2) and O is the number of optimization runs to increase generality (JRip default = 2). Thus, while there are a number of parameters to set in Rule Transfer, the parameters proved easy to set in practice and were not critical to the method's success.

5.2. Knight Joust to 3 vs. 2 Keepaway

In this section we present the results for transferring from Knight Joust to Keepaway using the cross-domain mappings in Table 5. Briefly, the intuition is that the forward action is similar to the hold ball action because the player should take it whenever possible. Note that the we have made West in the Knight Joust correspond to K_2 and East correspond to K_3, but either is reasonable, as long as the state variables and actions are consistent. When it is "too dangerous," the player instead jumps to the side, similar to passing the ball. We first train the Knight Joust players for 50,000 episodes, as initial experiments showed that learners generally stopped learning after roughly this many episodes. The advice is utilized by Extra Action Rule Transfer in Keepaway (informal experiments showed that Value Bonus and Extra Variable under-performed Extra Action, as in Ringworld) and other parameters are unchanged from the previous section. The results from these experiments are presented in Table 6.

The Knight Joust task is less similar to Keepaway than

Cross-Domain Mappings for Knight Joust to Keepaway

Knight Joust	**Keepaway**
δ_A	
Forward	Hold Ball
$Jump_{West}$	Pass to closest keeper
$Jump_{East}$	Pass to furthest keeper
δ_X	
$dist(P,O)$	$dist(K_1,T_1)$
ang(West)	Min($ang(K_2,K_1,T_1)$ $ang(K_2,K_1,T_2)$)
ang(East)	Min($ang(K_3,K_1,T_1)$, $ang(K_3,K_1,T_2)$)

Table 5. This table describes the cross-domain mapping used by `Translate()` to modify a decision list learned in the Knight Joust so that it can apply to Keepaway.

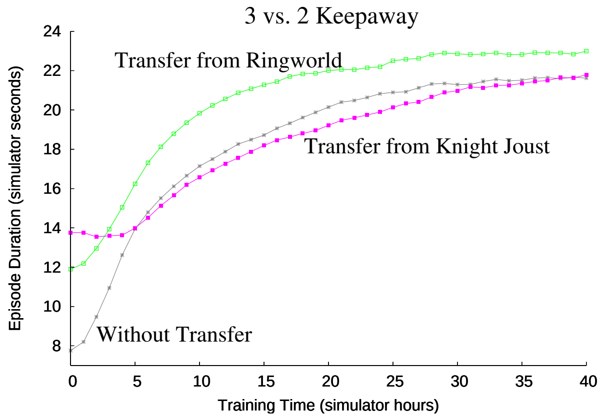

Figure 7. Learning curves in 3 vs. 2 Keepaway, averaged over 20 trials, showing learning without transfer, learning with Extra Action from Ringworld after 100 episodes of following the rule-suggested actions, and learning with Extra Action from Knight Joust after 100 episodes of following the rule-suggested actions.

Ringworld. There are many fewer state variables, a less similar transition function, and a very different reward structure. However, information from Knight Joust can significantly improve the initial performance of Keepaway players because very basic information, such as that it is desirable to maximize the distance to the opponent, will initially cause the players to perform better than acting randomly. The other two transfer metrics are not improved by transfer, however. We hypothesize that this is because the transferred knowledge, while allowing the agents to perform better than acting randomly, does not bias the learner towards an optimal policy and thus the rules are less helpful

Knight Joust into Keepaway

Param	Initial Performance	Asymptotic Performance	Accumulated Reward
Without Transfer			
	7.8 ± 0.1	21.6 ± 0.8	756.7 ± 21.8
Extra Action			
100	**13.8 ± 1.1**	21.8 ± 1.2	758.5 ± 29.3
250	**13.5 ± 0.9**	21.6 ± 0.9	747.9 ± 25.3

Table 6. Transferring from Knight Joust to Keepaway significantly improves the initial performance, but the other two metrics are not improved. All results are averaged over 20 independent trials and the standard error is shown. Numbers in **bold** are statistically different from learning without transfer at the 95% level, as determined via a Student's t-test.

after learning in the target task. In informal experiments, after 40 hours of training in Keepaway, agents that transferred advice from Ringworld were following the advice for 90% of the actions, while agents that transferred advice from Knight Joust were following the advice for only 85% of the actions, indicating that the Ringworld advice was more useful in Keepaway.

6. Learning a Rule Translation Function

Thus far, this work has relied on hand-coded cross-domain mappings δ_A and δ_X, as has much of the past transfer learning research. Learning cross-domain mappings in their entirety is an open and very challenging research problem, and is beyond the scope of this paper. In this section, we sketch a method by which a cross-domain mapping from Ringworld can be partially learned starting from some knowledge about the qualitative characteristics of the source domain. This method increases the degree of automation of Rule Transfer by removing the requirement for some given knowledge, though at the expense of requiring different, arguably more intuitive, information. Although this section focuses on using Ringworld as a source task, a similar process could be followed for Knight Joust.

An analysis of Ringworld results in observations A-C, and we make assumption D for feasibility:

- A An opponent approaches the player, as shown by a *distance to opponent* state variable, eventually ending the episode when the player fails to move.
- B As the player moves, the opponent will continue moving toward the player.
- C The action Run_{Near} takes longer to complete than the action Run_{Far}.
- D The learner has a sense of "identity" and can distinguish between other objects in the world. We accomplish this by assigning unique identifiers (UIDs) to all objects in a task. The agent initially knows only its own UID and is able to determine the UIDs that are used to compute each state variable.

Given these observations, we construct the following procedure under the assumption that this process could map Ringworld to multiple target tasks, and that constructing such a procedure is simpler than creating a mapping for a given target (or impossible if the target is unknown beforehand). Players in the target (3 vs. 2 Keepaway) first record $(S, A, time)$ tuples (for 60 episodes or roughly 7 simulator minutes). They then determine the mapping by the following procedure:

1. **Identify the state variable(s) that are near zero when the episode ends.** Observation A tells us that the Ringworld state variable $dist(P, O)$ should map to this state variable. We find that, on average, $dist(K_1, T_1)$ = 0.48 ± 0.2 and $dist(K_1, T_2)$ = 0.94 ± 0.4 when the episode ends after $Hold$ (all other state variables are at least an order of magnitude larger, on average).
2. **Identify the action that causes the target task variable(s) from step 1 to most consistently decrease.** Also by observation A, we find that $Stay$ should be mapped to $Hold$ in Keepaway because both $dist(K_1, T_1)$ and $dist(K_1, T_2)$ decrease during this action on average with a small variance, while this is not true for the actions $Pass_1$ and $Pass_2$.
3. **Identify the state variables that correspond to $\boldsymbol{dist(P, Target_1)}$ and $\boldsymbol{dist(P, Target_2)}$ and the actions that correspond to $\boldsymbol{Run_{Near}}$ and $\boldsymbol{Run_{Far}}$.**
   (a) Step 1 identified the UIDs of the two opponents. Using assumption D, we search all Keepaway state variables and consider only those that include the player's UID but not the opponents' UIDs: $dist(T_1, C)$, $dist(T_2, C)$, $dist(K_2, T)$, and Min($dist(K_3, T_1)$, $dist(K_3, T_2)$).
   (b) Next, consider the state before an action (either $Pass_1$ or $Pass_2$) is executed. Before the action, some state variable will be consistently greater than the value of $dist(P, O)$ after the action (Observation B). For Keepaway, $dist(K_2, T)$ is greater than $dist(P, O)$ after $Pass_1$ (the distance decreases by 6.8m on average) and Min($dist(K_3, T_1)$, $dist(K_3, T_2)$) is greater than $dist(P, O)$ after taking the action $Pass_2$ (an average decrease of 10.8m).
   (c) Observation C allows us to sort the two pass actions by how long they take: the action $Pass_1$ takes an average of 7.4 time steps while $Pass_2$ takes an average of 9.5 time steps. This allows us to decide that Run_{Near} should map to $Pass_1$ and Run_{Far} should map to $Pass_2$. Sorting out the pass actions also allows us to map $dist(P, Target_1)$ to $dist(K_2, T)$ and $dist(P, Target_2)$ to Min($dist(K_3, T_1)$, $dist(K_3, T_2)$).
4. **Identify $\boldsymbol{dist(P, Target_1)}$ and $\boldsymbol{dist(P, Target_2)}$.** Step 3 identified the UIDs of $Target_1$ and $Target_2$ and the UID of the player is known (assumption D). In Keepaway, the UIDs of $Target_1$ and the player identify $dist(P, Target_1)$ as $dist(K_1, K_2)$ and the UIDs of $Target_2$ and the player identify $dist(P, Target_2)$ as $dist(K_1, K_3)$.
5. **Identify $\boldsymbol{ang(O, P, Target_1)}$ and $\boldsymbol{ang(O, P, Target_2)}$.** Similar to Step 4, using the known UIDs of the player, the enemies, and the run targets, we identify $ang(O, P, Target_1)$ as $ang(T, K_1, K_2)$ and $ang(O, P, Target_2)$ as $ang(T, K_1, K_3)$.

Thus, using source task observations, we are able to map the source task onto Keepaway, using recorded data to construct a cross-domain mapping very similar to the hand-coded mapping from Table 2. The only difference is that the learned cross-domain mapping maps $dist(P, O)$ onto both $dist(K_1, T_1)$ and $dist(K_1, T_2)$, but since the takers' policy in 3 vs. 2 Keepaway causes them to stay very close together, in practice these two mappings are equivalent.

We emphasize that our choice of observations describing the source task are as important as the method for using them. If, for example, one of the observations were that $dist(P, Target_1)$ did not ever change when taking the $Stay$ action, $dist(P, Target_1)$ would not map to any state variable in Keepaway because all state variables change over time, regardless of action, due to noise in the sensors. However, in some cases, such as this one, we would be able to detect that the source task observations were violated. In these cases, it would be possible to consider a different set of observations for the source task, consider a different source task for transfer, or simply learn Keepaway without transfer. Thus we believe this method provides a reasonable solution for solving the learned mapping problem when the source task is well understood but the target is unknown.

7. Related Work

While a number of methods can successfully transfer between pairs of tasks in the same domain, the main novelty of this work is to show that inter-*domain* transfer is not only feasible, but beneficial. Additionally, we extend previous work on using advice to improve performance by comparing multiple rule utilization strategies. Most similar to our Value Bonus method is work by Kuhlmann et al. (2004) which gives a bonus to actions which are suggested by hand-coded rules. Other work (Madden & Howley, 2004) has shown that learned rules can be used to initialize Q-values when visiting a novel state, but such a method is not directly applicable in continuous domains. Learned advice has also been successfully used as soft constraints to help initialize a target task's function approximator off-line before learning (Torrey et al., 2006).

Other work has focused on learning task relationships. For instance, some work has used homomorphisms (Soni & Singh, 2006) to generate and empirically test a number of possible relationships. If a human can define both tasks can be defined in terms of *qualitative dynamic Bayes networks*, Liu and Stone (2006) showed that a graph matching method can automatically find task similarities.

8. Conclusion and Future Work

In this work we have introduced Rule Transfer along with three different advice utilization methods. Using three transfer metrics, Rule Transfer significantly improved learning in robot soccer after first learning in a gridworld task. In addition to hand-coding an cross-domain mapping function, we give evidence such a mapping may be learned from observed task data.

The Ringworld task was constructed directly from data gathered in the target task while the Knight Joust task was chosen as intuitively related to Keepaway. In future work, rather than constructing such tasks by hand, we would like to automatically construct such source tasks.

The cross-domain transfer experiments in this paper begin to demonstrate the flexibility of rule transfer; agents were able to transfer knowledge successfully irrespective of the underlying function approximator's representation. In the future we would like to further exploit this flexibility by transferring between agents with more dissimilar internal representations and learning methods. Finally, if both the source and target task had a finite number of states, the policies could be transferred directly. We plan to test this and determine the effect of using rules as an intermediary when they are not required for transfer.

Acknowledgments

We would like to thank Raymond Mooney, Cynthia Matuszek, Lilyana Mihalkova, Nick Jong, and the 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.

References

Cohen, W. W. (1995). Fast effective rule induction. *International Conf. on Machine Learning* (pp. 115–123).

Kuhlmann, G., Stone, P., Mooney, R., & Shavlik, J. (2004). Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. *The AAAI-2004 Workshop on Supervisory Control of Learning and Adaptive Systems*.

Liu, Y., & Stone, P. (2006). Value-function-based transfer for reinforcement learning using structure mapping. *Proc. of the 21st National Conference on Artificial Intelligence*.

Madden, M. G., & Howley, T. (2004). Transfer of experience between reinforcement learning environments with progressive difficulty. *Artif. Intell. Rev.*, *21*, 375–398.

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems* Technical Report CUED/F-INFENG-RT 116). Engineering Dept., Cambridge University.

Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replaceing eligibility traces. *Machine Learning*, *22*, 123–158.

Soni, V., & Singh, S. (2006). Using homomorphisms to transfer options across continuous reinforcement learning domains. *Proc. of the 21st National Conference on Artificial Intelligence*.

Srinivasan, A. (2001). The aleph manual.

Stone, P., Kuhlmann, G., Taylor, M. E., & Liu, Y. (2006). Keepaway soccer: From machine learning testbed to benchmark. In I. Noda, A. Jacoff, A. Bredenfeld and Y. Takahashi (Eds.), *RoboCup-2005: Robot soccer world cup IX*, vol. 4020, 93–105. Berlin: Springer Verlag.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning*. MIT Press.

Torrey, L., Shavlik, J. W., Walker, T., & Maclin, R. (2006). Skill acquisition via transfer learning and advice taking. *ECML* (pp. 425–436). Springer.

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.