
Relational Skill Transfer via Advice Taking

Lisa Torrey

University of Wisconsin, Madison WI 53706, USA

Jude Shavlik

University of Wisconsin, Madison WI 53706, USA

Trevor Walker

University of Wisconsin, Madison WI 53706, USA

Richard Maclin

University of Minnesota, Duluth MN 55812, USA

LTORREY@CS.WISC.EDU

SHAVLIK@CS.WISC.EDU

TWALKER@CS.WISC.EDU

RMACLIN@D.UMN.EDU

Abstract

We describe a reinforcement learning system that transfers relational skills from a previously learned source task to a related target task. The system uses inductive logic programming to analyze experience in the source task, and transfers rules for when to take actions. The target task learner accepts these rules through an advice-taking algorithm. Our system also accepts human-provided advice, which can guide the application of transferred skills and provide instruction about new skills in the target task.

1. Introduction

Machine learning tasks are often addressed independently, under the implicit assumption that each new task has no relation to the tasks that came before. In some domains, particularly reinforcement learning (RL) ones, this assumption is false since tasks in the same domain tend to have similarities. We view these similarities as shared *skills*. Our goal is to transfer shared skills from a source task in order to speed up learning in a new but similar target task.

For example, suppose an RL soccer player has learned, in a source task, to keep the ball away from its opponents by passing to its teammates. In the target task, suppose it must keep the ball away from its opponents in order to shoot goals. If this player started with the passing skills from the source task, it could focus on the new skills and master the target task more quickly. A human observer might also be able to give tips on new aspects of the problem, such as when to shoot.

We present a system for interacting with RL agents in this manner, called AI^2 (Advice via Induction and Instruction). Our system constructs relational *transfer advice* by using inductive logic programming to analyze experience in the source task and learn skills in first-order logic. The user can add supplementary *user advice* to guide transfer to the target task.

2. Reinforcement Learning in RoboCup

To demonstrate our approach, we introduce the RoboCup simulated soccer domain. In the task of M -on- N KeepAway, the objective of the M reinforcement learners called *keepers* is to keep the ball away from N hand-coded players called *takers*. The learners receive a +1 reward for each time step their team keeps the ball. In the original KeepAway task of Stone and Sutton (2001), the keeper who has the ball can choose only to hold it or pass to a teammate. In our version, they can also move (inwards, outwards, clockwise and counterclockwise with respect to the field center).

In the KeepAway state representation, the keepers are ordered by their distance to the learner $k0$, as are the takers. The features include distances and angles between players (13 of them for 3-on-2 KeepAway), such as $distBetween(k0, Player)$ and $angleDefinedBy(Keeper, k0, ClosestTaker)$. (Note that for simplicity, we denote variable types through their names throughout the paper.)

A second task is M -on- N BreakAway, where the objective of the M reinforcement learners called *attackers* is to score a goal against $N - 1$ hand-coded *defenders* and a hand-coded *goalie*. The learners receive a +1 reward for doing so. The attacker who has the ball may choose to move (ahead, away, left, or right with respect to the goal), pass to a teammate, or shoot (at the left, right, or center part of the goal).

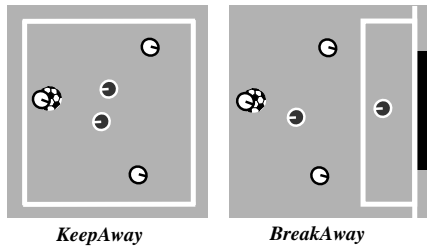


Figure 1. Samples of KeepAway and BreakAway games.

In the BreakAway state representation, the attackers are ordered by their distance to the learner a_0 , as are the defenders. The features include distances and angles between players and goal sections (21 of them for 3-on-2 BreakAway), such as $distBetween(a_0, Player)$ and $angleDefinedBy(GoalPart, a_0, goalie)$.

In both tasks, the features are discretized into 32 intervals called *tiles*, each of which is associated with a Boolean feature. A tile feature takes value 1 when the numeric value falls into its interval, and 0 otherwise. For example, the tile $distBetween(a_0, a_1)_{[0,20]}$ takes value 1 when a_1 is less than 20 units away from a_0 .

3. Transferring Skills

Because RL agents learn to take actions, a natural human interpretation of RL is that the agents acquire *skills*. However, what they typically acquire is a Q -function, which does not readily translate into discrete skills. Some researchers have therefore proposed methods for transferring an entire Q -function or policy directly (Taylor & Stone, 2005; Torrey et al., 2005).

In AI^2 , we present an approach that assumes no relationship between the Q -functions of the two tasks. Instead, it transfers relational skills by analyzing games played in the source task. Games are collections of state-action pairs, where the action can be viewed as the classification of the state. AI^2 uses these pairs as training examples to learn to classify states. For example, from traces of KeepAway games, AI^2 can learn the concept “states in which passing to a teammate is a good action.”

To use our system, the user identifies which skills should be transferred and provides a mapping that relates logical objects in the source task to those in the target task. From existing game traces in the source task, the system learns skill concepts and translates them into advice for the target task. After giving the user a chance to add new advice, it applies all of the advice to learning in the target task using the advice-taking algorithm in Maclin et al. (2005).

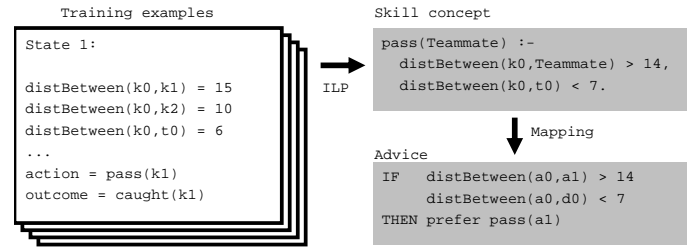


Figure 2. Example showing how AI^2 transfers skills.

Each advice item is a conjunction of conditions and a constraint to be applied if they are met, such as “prefer $pass(a_1)$ over move actions.” Under the conditions, AI^2 prefers the advised action over the other actions that existed in the source task.

3.1. Learning Rules

AI^2 uses inductive logic programming (ILP) to learn skill concepts as first-order rules, which we believe generalize well. For example, the rule $pass(Teammate)$ may capture the essential elements of the passing skill better than rules for individual teammates.

AI^2 uses the Prolog-based Aleph software package (Srinivasan, 2001), which conducts a random search in the hypothesis space. It selects the rule it finds with the highest $F(\beta)$ score (a generalization of the more familiar $F(1)$ metric, with $\beta^2 = 0.1$). Before searching, it collects states from games in the source task and sorts them into positive and negative examples.

In a positive example, several conditions must be met: the skill was performed, the desired outcome occurred, the expected Q -value (using the most recent Q -function) is above a minimum score $minQ^+$ and is at least $ratio^+$ times the expected Q -values of other actions. In a negative example, some other action was performed, the highest Q -value is above a minimum score $minQ^-$, and the expected Q -value of the skill being learned is at most $ratio^-$ times the highest and is below a maximum score $maxQ^-$.

Table 1. The AI^2 algorithm.

GIVEN	
	Game traces from source task
	Skills to be transferred
	Object mapping between tasks
DO	
FOR each skill:	
	Collect training examples
	Find rules with Aleph
	Select rule with highest $F(\beta)$ score
	Translate rule into advice for target task
	Add any extra advice from user
	Learn target task with advice

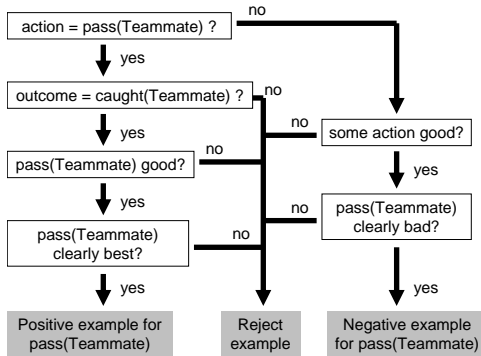


Figure 3. Example of how AI^2 selects training examples.

3.2. Mapping Rules

To produce advice for the new task, the system translates source-task objects into target-task objects based on the user-provided mapping. Next it instantiates rules like $pass(Teammate)$ into specific rules like $pass(a1)$. Finally, it propositionalizes any conditions that contain variables. For example, a rule for 3-on-2 BreakAway might contain this condition:

$$\text{distBetween}(a0, \text{Attacker}) < 20$$

This is effectively a disjunction of conditions: either the distance to $a1$ or the distance to $a2$ is less than 20. Although disjunctions are not part of the advice language, AI^2 does have a way to represent them. Recall that each feature range is divided into Boolean tiles that take value 1 when the feature value falls into their interval and 0 otherwise. The system can express the disjunction by requiring at least one of two tiles to be active:

$$\text{distBetween}(a0, a1)_{[0,20]} + \text{distBetween}(a0, a2)_{[0,20]} \geq 1$$

If these tile boundaries do not exist in the target task, AI^2 adds new tile boundaries to the feature space.

3.3. Editing and Adding Rules

Through this learning and mapping process, our system produces user-friendly, interpretable rules. Their intuitive nature makes it easy for a user to add to the rules AI^2 produces, guiding the application of the transfer advice.

For example, the passing skills transferred from KeepAway to BreakAway make no distinction between passing towards the goal and away from the goal. Since the new objective is to score goals, players should clearly prefer passing towards the goal. A user could provide this guidance by instructing the system to add the following condition to the $pass(Teammate)$ rule:

$$\text{distBetween}(Teammate, \text{goal}) < \text{distBetween}(a0, \text{goal})$$

Users might also want to suggest entirely new rules. This is particularly useful for new skills that are not being transferred from the source task. For example, users could provide simple rules for the shoot actions in BreakAway.

4. Empirical Results

We present results for transferring the skill $pass(Teammate)$ from our version of 4-on-3 KeepAway to 3-on-2 BreakAway. AI^2 found the following rule (in Prolog notation):

```

pass(Teammate) :-
    distBetween(k0, Teammate) > 14,
    angleDefinedBy(Teammate, k0, ClosestTaker) ∈ [30, 150],
    distBetween(k0, Taker) < 7,
    distBetween(k0, Player) < 11.
    
```

This rule produces one item of transfer advice for each teammate:

```

IF    distBetween(a0, a1) > 14 AND
      angleDefinedBy(a1, a0, closestTakerToA1) ∈ [30, 150] AND
      distBetween(a0, d0) < 7 AND
      distBetween(a0, a1)[0,11] + distBetween(a0, a2)[0,11]
      + distBetween(a0, d0)[0,11] ≥ 1
THEN  prefer pass(a1) over move actions

IF    distBetween(a0, a2) > 14 AND
      angleDefinedBy(a2, a0, closestTakerToA2) ∈ [30, 150] AND
      distBetween(a0, d0) < 7 AND
      distBetween(a0, a1)[0,11] + distBetween(a0, a2)[0,11]
      + distBetween(a0, d0)[0,11] ≥ 1
THEN  prefer pass(a2) over move actions
    
```

To encourage passing towards the goal, we assume the user adds the extra constraint described in Section 3.3. Finally, we assume the user provides approximations of new skills in BreakAway as follows:

```

IF    distBetween(a0, goalLeft) < 10 AND
      angleDefinedBy(goalLeft, a0, goalie) > 40
THEN  prefer shoot(goalLeft) over all actions

IF    distBetween(a0, goalRight) < 10 AND
      angleDefinedBy(goalRight, a0, goalie) > 40
THEN  prefer shoot(goalRight) over all actions

IF    distBetween(a0, goalCenter) > 10
THEN  prefer moveAhead over shoot actions
    
```

We compare learning in BreakAway with and without this advice to evaluate its effects. To analyze the individual contributions of transfer advice and user advice, we also compare learning with each advice type separately. Each curve is an average of 5 independent runs, and each data point is smoothed over 1000 games.

With the full set of advice, the scoring probability is significantly higher at the 95% confidence level, based on unpaired t -tests on points on these curves, at nearly every data point. Learners without advice take about 4500 games to begin scoring a goal 50% of the time; with AI^2 they reach this level in 2500 games.

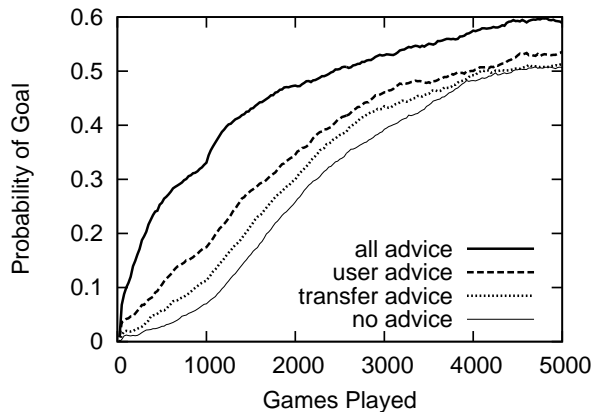


Figure 4. Learning curves for 3-on-2 BreakAway with several combinations of transfer and user advice.

The combination of transfer advice and user advice improves performance substantially more than either type does alone; these differences are also significant at the 95% confidence level at nearly every data point.

5. Related Work

Our approach builds on methods for providing advice to RL agents. Driessens and Dzeroski (2002) use human guidance to create a partial initial Q -function in relational RL. Kuhlmann et al. (2004) propose rule-based advice to increase Q -values by a fixed amount. Maclin et al. (2005) present the Pref-KBKR method for specifying advice as action preferences.

Another aspect of our work is extracting explanatory rules from complex functions. Sun (2002) studies rule learning from neural-network based reinforcement learners. Fung et al. (2005) investigate extracting rules from support vector machines.

We also address knowledge transfer in RL. Singh (1992) studies transfer of knowledge between sequential decision tasks. Taylor and Stone (2005) copy initial Q -functions to transfer between KeepAway games of different sizes. Torrey et al. (2005) introduce transfer from KeepAway to BreakAway using advice extracted from the Q -function.

6. Conclusions and Future Work

Reinforcement learners can benefit significantly from the user-guided transfer of skills from a previous task. We have presented the AI^2 system, which transfers structured skills by learning first-order rules from agent behavior and allowing user guidance. This system assumes no relationship between the Q -functions

of the source and target tasks. Our experimental results demonstrate the effectiveness of this approach in a complex RL domain.

A challenge that we have encountered in RL transfer learning is that differences in action sets and reward structures between the source and target task make it difficult to transfer even shared actions. Changing the game objective or adding a new action changes the meaning of a shared skill. We have addressed this problem with user guidance, relying on human domain knowledge to apply transferred skills appropriately. However, we would prefer to minimize the user’s role in transfer.

We believe that the underlying issue is the separation of *general* from *specific*. In RL transfer learning we want to transfer only general aspects of skills in a domain, filtering out task-specific aspects. Our use of ILP to learn general, first-order skill concepts is a step towards this goal. A future step we are considering is learning skills from multiple games in a domain, which we believe may lead to even more general rules and therefore better transfer.

References

- Driessens, K., & Dzeroski, S. (2002). Integrating experimentation and guidance in relational reinforcement learning. *Proc. ICML '02*.
- Fung, G., Sandilya, S., & Rao, B. (2005). Rule extraction from linear support vector machines. *Proc. KDD '05*.
- Kuhlmann, G., Stone, P., Mooney, R., & Shavlik, J. (2004). Guiding a reinforcement learner with natural language advice. *AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*.
- Maclin, R., Shavlik, J., Torrey, L., Walker, T., & Wild, E. (2005). Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. *Proc. AAAI '05*.
- Singh, S. (1992). Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning 8 (3-4)*, 323-339.
- Srinivasan, A. (2001). The Aleph manual.
- Stone, P., & Sutton, R. (2001). Scaling reinforcement learning toward RoboCup soccer. *Proc. ICML '01*.
- Sun, R. (2002). Knowledge extraction from reinforcement learning. *New learning paradigms in soft computing*, 170-180.
- Taylor, M., & Stone, P. (2005). Behavior transfer for value-function-based reinforcement learning. *Proc. AAMAS '05*.
- Torrey, L., Walker, T., Shavlik, J., & Maclin, R. (2005). Using advice to transfer knowledge acquired in one reinforcement learning task to another. *Proc. ECML '05*.