Relational Skill Transfer via Advice Taking

Lisa Torrey LTORREY@CS.WISC.EDU
Jude Shavlik SHAVLIK@CS.WISC.EDU
Trevor Walker TWALKER@CS.WISC.EDU
University of Wisconsin, Madison WI 53706, USA
Richard Maclin RMACLIN@D.UMN.EDU
University of Minnesota, Duluth MN 55812, USA

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 about 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 might 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 at the goal.

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 also add supplementary *user advice* about new skills. The target task learner receives the advice and can follow it, refine it, or ignore it according to its value.

Appearing in the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning, June 29, Pittsburgh, PA. A longer version is UW ML Group Working Paper 06-2.

2. Reinforcement Learning in RoboCup

To demonstrate our approach, consider the RoboCup simulated soccer domain (see Figure 1). 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, called Mobile KeepAway, they can also move (inwards, outwards, clockwise and counterclockwise with respect to the field center). We developed this version because we expect its more realistic movement to generalize better to other RoboCup games.

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, 19 of them for 4-on-3 KeepAway, such as *distBetween(k0, Player)* and *angleDefinedBy(Keeper, k0, ClosestTaker)*. Note that for simplicity, we denote variable types through their names.

A second task is M-on-N BreakAway (Torrey et al., 2005), 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 a goal and 0 reward otherwise. 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 a goal section.

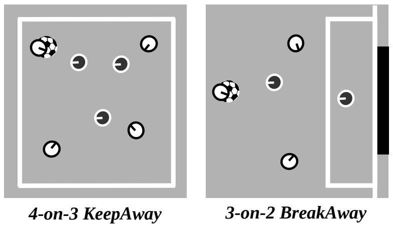
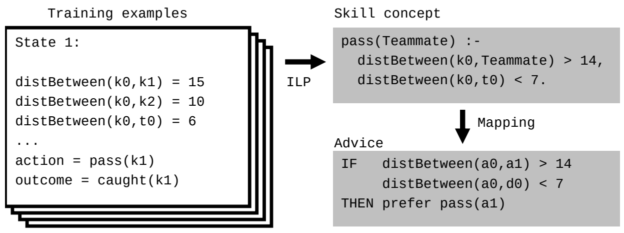


Figure 1. Snapshots of KeepAway and BreakAway games.



```
State 1:

distBetween(k0,k1) = 15
distBetween(k0,k2) = 10
distBetween(k0,t0) = 6
...
action = pass(k1)
outcome = caught(k1)
```

```
pass(Teammate) :-
  distBetween(k0,Teammate) > 14,
  distBetween(k0,t0) < 7.
```

```
IF   distBetween(a0,a1) > 14
     distBetween(a0,d0) < 7
THEN prefer pass(a1)
```

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

In the BreakAway state representation, the attackers are ordered by their distance to the learner *a0*, 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(a0, Player)* and *angleDefinedBy(GoalPart, a0, goalie)*.

In both tasks, following the approach of Stone and Sutton (2001) the features are discretized into 32 intervals called *tiles*, each of which is associated with a Boolean feature. For example, the tile *distBetween(a0, a1)*$_{[10,20]}$ takes the value 1 when *a1* is bewteen 10 and 20 units away from *a0* and 0 otherwise.

AI^2 uses the *SARSA* and TD(λ) algorithms for RL. It approximates the Q-function and incorporates advice using a linear optimization method called KBKR (Maclin et al., 2005). Our player code was developed from the University of Amsterdam Trilearn code base.

3. AI^2: Transferring Skills

We propose a transfer method that does not assume the two tasks have similarly structured Q-functions. 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 KeepAway games, AI^2 can learn the concept "states in which passing to a teammate is a good action."

To use AI^2, the user identifies which skills should be transferred, provides a mapping that relates objects in the source task to those in the target task, and optionally gives advice to guide transferred skills or provide new skills.

Given this information, AI^2 performs transfer automatically. From existing game traces in the source task, the system learns skill concepts and translates them into advice for the target task. It then applies both the transfer advice and the user advice to learning in the target task. Table 1 summarizes this AI^2 algorithm in high-level pseudocode. Figure 2 illustrates the transfer part of the algorithm.

Each advice item is a conjunction of conditions and a constraint to be applied if they are met, such as "prefer *pass(a1)* over move actions." Advice need not be followed exactly; it can be refined or even ignored if it disagrees with the learner's experience, using the advice-taking algorithm of Maclin et al. (2005). This provides some protection against imperfect transfer.

We anticipate that AI^2 would be used when a new task arises in a domain and data from an old task already exists. In the target task BreakAway, we assume the objective is to maximize the probability of scoring a goal after playing a given number of training games.

3.1. Learning Skills

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 passing to individual teammates.

Using the Prolog-based Aleph software package (Srinivasan, 2001), AI^2 conducts a random search in the hypothesis space. It selects the rule it finds with the highest F(β) score (a generalization of the more familiar F(1) metric; we use $\beta^2 = 0.1$). To produce datasets for this search, it selects positive and negative examples from source task games.

To be a positive example, a state must meet several conditions: the skill was performed, the desired outcome occurred, the expected Q-value (using the most recent Q-function) is above a minimum score $minQ_{pos}$

Table 1. The AI^2 algorithm.

```
GIVEN
    Game traces from source task
    Skills to be transferred
    Object mapping between tasks
    User advice (optional)
DO
    FOR each skill:
        Collect training examples
        Learn rules with Aleph
        Select rule with highest $F(\beta)$ score
        Translate rule into transfer advice
    Learn target task with all advice
```

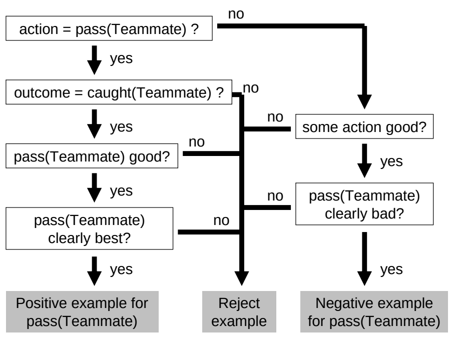

action = pass(Teammate) ?
no
yes
outcome = caught(Teammate) ?
no
yes
pass(Teammate) good?
no
no
some action good?
yes
pass(Teammate) clearly best?
no
no
pass(Teammate) clearly bad?
yes
yes
Positive example for pass(Teammate)
Reject example
Negative example for pass(Teammate)

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

and is at least $ratio_{pos}$ 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 $minQ'_{neg}$, and the expected Q-value of the skill being learned is at most $ratio_{neg}$ times the highest and is below a maximum score $maxQ_{neg}$. The standard ratio values in AI^2 are $ratio_{pos} = 1.10$ and $ratio_{neg} = 0.90$, and the other parameters are set so that there are at least 100 positive and 100 negative examples. Figure 3 illustrates this sorting process with an example from RoboCup.

3.2. Mapping Skills

To produce advice for the new task, the system translates source-task objects into target-task objects based on the user-provided mapping. (Learning this mapping automatically is an interesting topic that we have not yet addressed.) Next it instantiates skills 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:

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

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. This way advice from the source task can be expressed exactly without knowing the target-task feature space when learning the source task.

3.3. User Advice

As part of the mapping from one task to another, users can optionally provide additional advice. This advice can guide the application of transferred skills. 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)* skill:

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

Users may also provide advice on entirely new skills that are not being transferred from the source task. For example, users could provide simple rules for the *shoot* actions in BreakAway. User advice is not necessary for AI^2 to perform transfer, but it provides a natural and powerful way for users to interact with the transfer process.

4. Empirical Results

We present results for transferring the skill *pass(Teammate)* from 4-on-3 Mobile KeepAway to 3-on-2 BreakAway. AI^2 learned the following skill (in Prolog notation):

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

This skill produces one item of transfer advice for each teammate. 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 in Figure 4 is an average of 10 independent runs, and each data point is smoothed over 500 games.

With transfer advice alone (this does not include the constraint for passing forward) the scoring probability

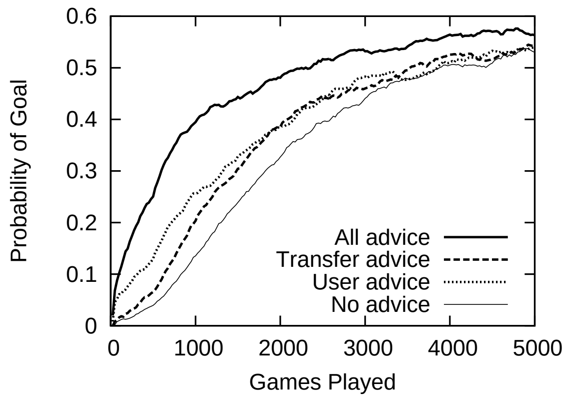


Figure 4. Learning curves for 3-on-2 BreakAway with combinations of transfer and user advice.

is higher at the 90% confidence level, based on unpaired t-tests, between 200 and 2500 games played. With the full set of advice, scoring is more probable at the 95% confidence level at nearly every point. The full AI^2 system also performs significantly better than user or transfer advice alone at the 95% confidence level. Transfer and user guidance together produce synergy, performing better than the sum of their parts.

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.

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 constructed 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 does not assume a similar reward structure between the source and target tasks and provides some robustness to imperfect transfer through advice-taking. 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, using human domain knowledge to help apply transferred skills. 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* knowledge. 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.

7. Acknowledgements

This research is partially supported by DARPA grant HR0011-04-1-0007 and United States Naval Research Laboratory grant N00173-06-1-G002.

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.*