Policy Transfer via Markov Logic Networks

Lisa Torrey and Jude Shavlik

University of Wisconsin, Madison WI, USA
ltorrey@cs.wisc.edu, shavlik@cs.wisc.edu

Abstract. We propose an improved method for transfer in reinforcement learning via Markov Logic Networks. Our goal is to extract relational knowledge from a source task and use it to speed up learning in a related target task. We do so by learning a Markov Logic Network that describes the source-task policy, and then using it for decision making in the early learning stages of the target task. Through experiments in the RoboCup simulated-soccer domain, we show that this approach outperforms a previous MLN transfer method that modeled a value function rather than a policy.

1 Introduction

The transfer of knowledge from one task to another is a desirable property in machine learning. Our ability as humans to transfer knowledge allows us to learn new tasks quickly by taking advantage of relationships between tasks. While many machine-learning algorithms learn each new task from scratch, there are also *transfer-learning* algorithms that can improve learning in a *target task* using knowledge from a previously learned *source task*.

In reinforcement learning [11] (RL), an agent navigates through an environment, sensing its state, taking actions, and trying to earn rewards. The *policy* of the agent determines which action it chooses in each step. An agent performing RL typically learns a *value function* to estimate the values of actions as a function of the current state, and its policy typically is to take the highest-valued action in all except occasional *exploration* steps.

In complex domains, RL can require many early episodes of nearly random exploration before acquiring a reasonable value function or policy. A common goal of transfer in RL is to shorten or remove this period of low performance. Recent research has yielded a wide variety of RL transfer algorithms to accomplish this goal [12]. In one category of methods, RL agents apply a source-task policy or value function at some point(s) while learning the target task. Approaches of this type vary in the representation of the source-task policy and in the timing and frequency of its application.

Madden and Howley [5] learn a set of rules to represent a source-task policy, and they use those rules only during exploration steps in the target task. Fernandez and Veloso [2] use the original representation of the source-task policy, and give the target-task agent a three-way choice between using the current target-task policy, using a source-task policy, and exploring randomly. Croonenborghs

Presented at the 19th International Conference on Inductive Logic Programming (ILP'09).

et al. [1] learn a relational decision tree to represent the source-task policy, and use the tree as a multi-step action that may be chosen in place of a target-task action in any step.

Our own work in this area has contributed several relational methods, in which the knowledge transferred is at the level of first-order logic, and is extracted from the source task with a relational learning method such as inductive logic programming [7] (ILP). We transfer several types of relational models.

In one approach [14], the transferred model is a first-order finite-state machine that we call a *relational macro*, and it represents a successful generalized source-task plan. In our most recent approach [13], the transferred model is a Markov Logic Network [8] (MLN), a statistical-relational model that combines first-order logic and probability, and it represents the source-task value function.

In this paper, we propose a new method for transfer via MLNs. Instead of transferring an MLN model that represents the value function, we transfer an MLN model that represents the policy. This is a more effective method, because while the source-task action choice is often a good choice for the target task, the numerical values of actions often differ between the two tasks, due to differences in reward functions or task difficulty. It is also more effective than our macro-transfer method in some cases. We show these comparisons with experiments in the simulated soccer domain RoboCup [6].

2 Reinforcement Learning in RoboCup

In one common form of RL called Q-learning [11], the value function learned by the agent is called a Q-function, and it estimates the value of taking an action from a state. The policy is to take the action with the highest Q-value in the current state, except for occasional exploratory actions taken in a small percent ϵ of steps. After taking an action and receiving some reward (possibly zero), the agent updates its Q-value estimates for the current state.

Stone and Sutton ([10]) introduced RoboCup [6] as an RL domain that is challenging because of its large, continuous state space and non-deterministic action effects. Since the full game of soccer is quite complex, researchers have developed several simpler games within the RoboCup simulator.

In M-on-N BreakAway (see Figure 1), the objective of the M reinforcement learners called *attackers* is to score a goal against $N-1$ hand-coded *defenders* and a *goalie*. The game ends when they succeed, when an opponent takes the ball, when the ball goes out of bounds, or after a time limit of 10 seconds. The learners receive a $+1$ reward if they score 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 center), pass to a teammate, or shoot (at the left, right, or center part of the goal).

Figure 2 shows the state features for BreakAway, which mainly consist of distances and angles between players and the goal. They are shown in logical notation since we perform transfer learning in first-order logic; our basic RL algorithm uses the grounded versions of these literals. Capitalized atoms indicate

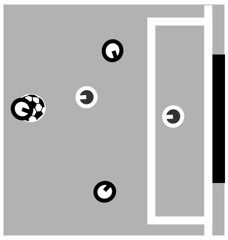

Fig. 1. Snapshot of a 3-on-2 BreakAway game. The attacking players have possession of the ball and are maneuvering against the defending team towards the goal.

distBetween(a0, Player)
distBetween(a0, GoalPart)
distBetween(Attacker, goalCenter)
distBetween(Attacker, ClosestDefender)
distBetween(Attacker, goalie)
angleDefinedBy(topRight, goalCenter, a0)
angleDefinedBy(GoalPart, a0, goalie)
angleDefinedBy(Attacker, a0, ClosestDefender)
angleDefinedBy(Attacker, a0, goalie)
timeLeft

Fig. 2. The features that describe a BreakAway state in their first-order logical form, where variables are capitalized.

typed variables, while constants and predicates are uncapitalized. The attackers (labeled *a0*, *a1*, etc.) are ordered by their distance to the agent in possession of the ball (*a0*), as are the non-goalie defenders (*d0*, *d1*, etc.).

Our basic RL algorithm uses a $SARSA(\lambda)$ variant of Q-learning [11] and employs a support vector machine (SVM) for Q-function approximation [4]. It relearns the SVM Q-function after every batch of 25 games. The exploration rate ϵ begins at 2.5% and decays exponentially over time. Stone and Sutton ([10]) found that discretizing the continuous features into Boolean interval features called *tiles* is necessary for learning in RoboCup; following this approach, we create 32 tiles per feature.

Agents in the games of 2-on-1 and 3-on-2 BreakAway take between 1000 and 3000 training episodes to reach a performance asymptote in our system. The differences in the numbers of attackers and defenders cause substantial differences in their optimal policies, particularly since there is a type of player entirely missing in 2-on-1 (the non-goalie defender). However, the tasks do have the same objective, and transfer between them should improve learning.

3 Policy Transfer via MLNs

The Markov Logic Network [8] (MLN) is a model that combines first-order logic and probability. It expresses concepts with first-order rules, as in standard ILP, but it also puts weights on the rules to indicate how important they are. While standard ILP rulesets can only predict a concept to be true or false, an MLN can estimate the probability that a concept is true, by comparing the total weight of satisfied rules to the total weight of unsatisfied rules.

Formally, a Markov Logic Network is a set of first-order logic formulas M, with associated real-valued weights W, that provides a template for a Markov network. With an MLN (M, W), one can calculate the conditional probability of any node in the network given *evidence* about the truth values of other nodes.

Table 1. Our algorithm for learning an MLN policy from a source task.

```
Let $M = \emptyset$                          // This will be the set of MLN formulas
Let $G$ be the set of high-reward source-task games
For each source-task action $a$
    Let $Pos$ be the set of states in $G$ games in which $a$ was taken
    Let $Neg$ be the set of states in $G$ games in which $a$ was not taken
    Learn rules with Aleph to distinguish $Pos$ from $Neg$
    Let $R$ be the rules sorted by decreasing precision on the training set
    Let $S = \emptyset$                      // This will be the final ruleset for action a
    For each rule $r \in R$                  // Add the rule if it improves the F score
        Let $T = S \cup \{r\}$
        If $F(T) > F(S)$ then set $S \longleftarrow T$
    Set $M \longleftarrow M \cup S$
Learn MLN weights $W$ for the formulas $M$ with Alchemy
Return the MLN described by $(M, W)$
```

In MLN policy transfer, the formulas M recommend actions. For example, an MLN formula recommending the *pass* action in 2-on-1 BreakAway is:

IF distBetween(a0, GoalPart) ≥ 27
AND angleDefinedBy(topRightCorner, goalCenter, a0) ≤ 75
AND distBetween(a0, Teammate) ≥ 9
AND angleDefinedBy(Teammate, a0, goalie) ≥ 25
THEN pass(Teammate)

Each action may have many such formulas with different weights. In the Markov network produced by these formulas, there is a node for each action and a node for each grounded literal. To use the MLN (M, W), a target-task agent evaluates the conditional probability of each action node given the values of all the evidence nodes. These represent the probabilities that a source-task agent would choose each action. The target-task agent chooses the action whose node has highest probability.

Our algorithm for learning an MLN policy is summarized in Table 1. It begins by learning rules such as the example above using the ILP system Aleph [9]. We define positive examples for an action as states in high-reward games in which that action was taken, and negative examples as states in high-reward games in which a different action was taken. We do not use low-reward games because we cannot be sure which action(s) were responsible for the low reward. In BreakAway, high-reward games are those that scored a goal; other tasks could require different definitions.

From the rules generated by Aleph, our algorithm selects a final ruleset for each action. It does so using an efficient method that approximately optimizes for both precision and recall: sorting rules by decreasing precision and greedily adding them to the final ruleset if they improve its F score. The combined rulesets for all the actions form the set of formulas M in the MLN. The algorithm learns weights W for these formulas using the scaled conjugate-gradient method in the Alchemy MLN system [3].

The target-task learner uses the MLN (M, W) to choose actions during its first 100 episodes. In the remaining episodes, it continues learning the target task normally, choosing actions with its developing Q-function. This *demonstration* approach in the target task follows our previous algorithms [13, 14].

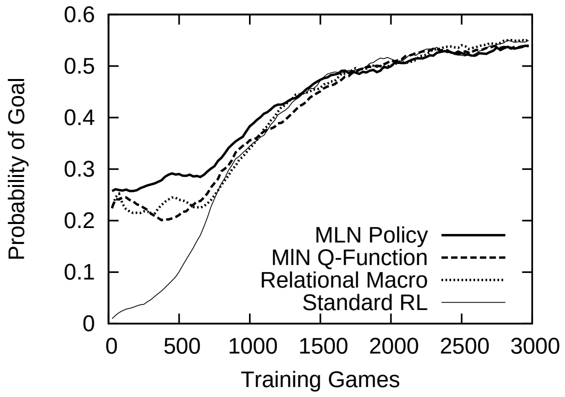

Fig. 3. Probability of scoring a goal in 3-on-2 BreakAway with standard RL, relational macro transfer from 2-on-1 BreakAway, MLN Q-function transfer from 2-on-1 BreakAway, and MLN policy transfer from 2-on-1 BreakAway.

4 Experimental Results

To test MLN policy transfer, we learn MLNs from source tasks in 2-on-1 BreakAway and transfer them to 3-on-2 BreakAway. Figure 3 shows the performance of MLN policy transfer compared to RL without transfer, macro transfer [14], and MLN Q-function transfer [13]. Each curve in the figure is an average of 25 runs and has points averaged over the previous 250 games to smooth over the high variance in the RoboCup domain. The transfer curves consist of five target-task runs generated from each of five source-task runs, to account for variance in both stages of learning.

MLN policy transfer is more effective than both of our previous approaches in this transfer scenario. The results are statistically significant; the area under the curves for MLN policy transfer is larger than for all the others ($p < 0.05$). MLN policy transfer may perform better because it combines the best aspects of MLN Q-function transfer (the use of a strong statistical-relational model) with the best aspects of macro transfer (the transfer of policy knowledge rather than value-function knowledge).

5 Conclusions and Future Work

We propose an improved algorithm for transfer in reinforcement learning via Markov Logic Networks, by representing the source-task policy rather than its value function. Through experiments in a complex domain, we show that this method outperforms MLN Q-function transfer and can also outperform another previous method.

Future work in this area could focus on revision of the transferred model– whether it is a macro, MLN, or other type of model– after the initial demonstration episodes. Our methods currently revert to standard RL, but they could instead learn by incrementally revising the source-task knowledge.

A related area of potential work is MLN-based relational reinforcement learning. Domains like RoboCup could benefit from relational RL, which would provide substantial generalization over objects and actions. The main challenge to overcome in performing relational RL in such a complex domain is the computational cost of learning MLN structures and weights.

6 Acknowledgements

This research is supported by DARPA grants HR0011-07-C-0060 and FA8650-06-C-7606.

References

1. T. Croonenborghs, K. Driessens, and M. Bruynooghe. Learning relational skills for inductive transfer in relational reinforcement learning. In *International Conference on Inductive Logic Programming*, Corvallis, OR, 2007.
2. F. Fernandez and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Conference on Autonomous Agents and Multi-Agent Systems*, Hakodate, Japan, 2006.
3. S. Kok, P. Singla, M. Richardson, and P. Domingos. The Alchemy system for statistical relational AI. Technical report, University of Washington, 2005.
4. R. Maclin, J. Shavlik, L. Torrey, and T. Walker. Knowledge-based support vector regression for reinforcement learning. In *IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, Edinburgh, Scotland, 2005.
5. M. Madden and T. Howley. Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, 21:375–398, 2004.
6. I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
7. L. De Raedt. *Logical and Relational Learning.* Springer, 2008.
8. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
9. A. Srinivasan. The Aleph manual, 2001.
10. P. Stone and R. Sutton. Scaling reinforcement learning toward RoboCup soccer. In *International Conference on Machine Learning*, Williamstown, MA, 2001.
11. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.
12. L. Torrey and J. Shavlik. Transfer learning. In E. Soria, J. Martin, R. Magdalena, M. Martinez, and A. Serrano, editors, *Handbook of Research on Machine Learning Applications.* IGI Global, 2009.
13. L. Torrey, J. Shavlik, S. Natarajan, P. Kuppili, and T. Walker. Transfer in reinforcement learning via Markov Logic Networks. In *AAAI Workshop on Transfer Learning for Complex Tasks*, Chicago, IL, 2008.
14. L. Torrey, J. Shavlik, T. Walker, and R. Maclin. Relational macros for transfer in reinforcement learning. In *International Conference on Inductive Logic Programming*, Corvallis, OR, 2007.