Relational Macros for Transfer in Reinforcement Learning

Lisa Torrey, Jude Shavlik, Trevor Walker
Computer Sciences Department
University of Wisconsin-Madison

Supported by DARPA grant HR0011-04-1-0007
and DARPA IPTO contract FA8650-06-C-7606

Richard Maclin
Computer Science Department
University of Minnesota-Duluth

Abstract

We describe an application of inductive logic programming to transfer learning. Transfer learning is the use of knowledge learned in a source task to improve learning in a related target task. The tasks we work with are in reinforcement learning domains. Our approach transfers relational macros, which are finite-state machines in which the transition conditions and the node actions are represented by first-order logical clauses. We use inductive logic programming to learn a macro that characterizes successful behavior in the source task, and then use the macro for decision-making in the early learning stages of the target task. Using experiments in the RoboCup simulated soccer domain, we show that this transfer method provides a substantial head start in the target task.

Transfer Learning

Agent learns Task A

↓

Agent encounters related Task B

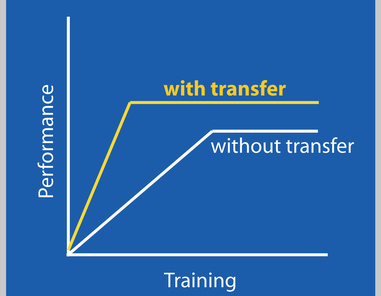
↓

Agent recalls relevant knowledge from Task A

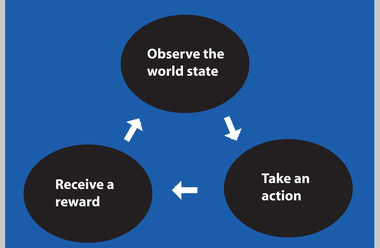
↓

Agent uses this knowledge to learn Task B quickly

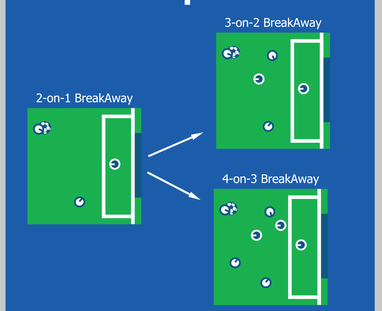
Goals of Transfer



Reinforcement Learning



RoboCup Domain



Relational Macros

- Relational macros are finite-state machines
- Nodes represent internal states of the agent in which independent policies apply
- Conditions for transitions and actions are sets of rules in first-order logic

Proposed Method

Learn a macro that describes a successful source-task strategy

↓

Demonstrate the strategy in the target task by executing the macro

↓

Continue learning the target task with standard RL

Learning Structure

- First learn an action pattern that reasonably separates good and bad games

```
macroSequence(Game) :-
    actionTaken(Game, StateA, move, ahead, StateB),
    actionTaken(Game, StateB, pass,  _,     StateC),
    actionTaken(Game, StateC, shoot, _,     gameEnd).
```

- This forms the node structure of the macro

move(ahead) → pass(Teammate) → shoot(GoalPart)

Learning Conditions

- Next learn the conditions for transitioning between nodes and choosing actions

move(ahead) → pass(Teammate)

```
transition(State) :-
    feature(State, distance(Teammate, goal)) < 15.
```

pass(Teammate)

```
action(State, pass(Teammate)) :-
    feature(State, angle(Teammate, me, Opponent)) > 30.
```

Choosing Examples to Learn Conditions

- Positive examples: States in successful games that followed the macro and took the step being learned
- Negative examples: States that followed the macro until the step being learned and then diverged
- Negative examples: States that took the action in the step being learned and immediately ended badly

Transferring a Macro

- Execute the macro strategy for 100 games in the target task to get a set of Q-value estimates
- Infer low Q-value estimates for actions not taken by the macro
- Compute an initial Q-function and then continue learning with standard RL

2-on-1 Macro

pass(Teammate)

↓

move(Direction)

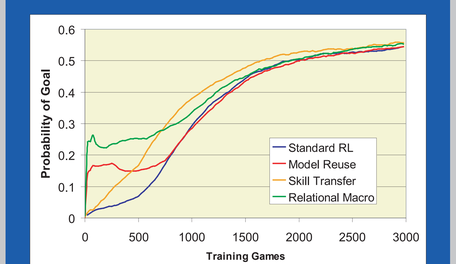
↓

shoot(goalRight)

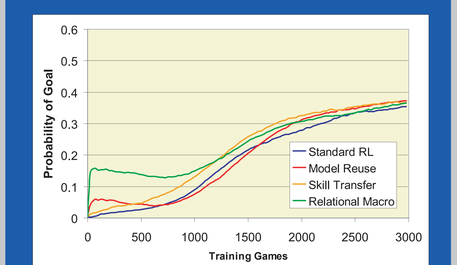
↓

shoot(goalLeft)

3-on-2 Results



4-on-3 Results



Evaluation

- This approach can significantly increase the initial performance in the target task
- It is a good choice if the source and target tasks have similar strategies
- The method can handle new elements being added to the target task, but not new objectives
- Future work may address transfer between tasks that share partial strategies