Representation Transfer via Elaboration

Matthew E. Taylor and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188
{mtaylor, pstone}@cs.utexas.edu
http://www.cs.utexas.edu/{~mtaylor,~pstone}

Introduction

A key component of any reinforcement learning (RL) algorithm is the underlying *representation* used by the agent for learning (e.g. the parameterization of its function approximator). *Transfer learning* tasks typically look at speeding up a target task after learning in a source task. This paper considers a different, but related, question: is it possible, and desirable, for agents to transfer from a source representation to a target representation? *Elaboration*, presented below, is a *representation transfer* (RT) algorithm that may allow an agent to learn faster than learning with a single representation.

One motivation for such flexibility is learning speed: changing representations partway through learning may allow agents to achieve better performance in less time. SOAR (Laird, Newell, & Rosenbloom 1987) can use multiple descriptions of planning problems and search problems, generated by a human user, for just this reason. Other work (Sherstov & Stone 2005) suggests that gradually reducing the generalization of a function approximator may improve the speed of learning. In this paper we suggest that it is advantageous to modify the internal representation while learning in some RL tasks, relative to using a fixed representation, so that higher performance is achieved more quickly.

Elaboration

Elaboration is a type of representation transfer where the function approximator (FA) is changed to allow for more representational power over time. Consider the decision of whether to represent state variables conjunctively or independently. A linear interpolation of different state variables may be faster to learn, but a conjunctive representation has more descriptive power. Using Elaboration, the agent can learn with a simple representation initially and then switch to a more complex representation later. Thus the agent can reap the benefits of fast initial training without suffering from decreased asymptotic performance.

Algorithm 1 describes the process for transferring between value function representations with different parameterizations of state variables, e.g. FAs with different dimensionalities. The weights (parameters) of a learned FA are used as needed when the agent learns a target value function representation. If the target representation must calculate $Q(s, a)$ using a weight which is set to the default value rather than a learned one, the agent uses the source representation to set the weight. Using this process, a single weight from the source representation can be used to set multiple weights in the target representation.

Algorithm 1 Elaboration

1: **while** Source task performance or time threshold not reached **do**
2: Source task agent trains on task with FA_S
3: **while** Target task agent trains on a task with FA_T **do**
4: **if** $Q(s, a)$ needs to use at least one uninitialized weight in FA_T **then**
5: Find the set of weights W that would be used to calculate $Q(s, a)$ with FA_S
6: Set any remaining uninitialized weight(s) in FA_T needed to calculate $Q(s, a)$ to the average of the weights in W

XOR Keepaway

To test the efficacy of RT we consider the RoboCup simulated soccer Keepaway domain. We use a setup similar to past research (Stone, Sutton, & Kuhlmann 2005) and agents based on version 0.6 of the benchmark players distributed by UT-Austin (Stone *et al.* 2006). In 3 vs. 2 Keepaway, three keepers on a 20m $\times$ 20m field attempt to control the ball while 2 takers attempt to capture the ball or kick it out of bounds. The keeper's world state is defined by 13 variables Takers, and keepers that do not possess the ball, follow static hand-coded policies.

A keeper possessing the ball in the standard 3 vs. 2 Keepaway task may choose to either hold the ball or pass it to a teammate: A = {*hold*, *passToTeammate1*, *passToTeammate2*}. In XOR Keepaway, the 3 vs. 2 Keepaway task is modified[1] so that the effect of agents' actions are changed. We define *good pass*, which executes the pass action and additionally disables the takers for 2 seconds. *Bad pass* causes the keeper's pass to travel directly to the closest taker. These effects are triggered based on the agent's chosen pass action and 4 state variables; keepers must learn a policy that considers these four state variables conjunctively in order to avoid the *bad pass* action. Thus an agent's representation must be capable of learning an "exclusive or" to achieve top performance. A linear (non-conjunctive) representation can learn quickly but is eventually outperformed by a more complex representation.

[1]In informal experiments, Elaboration did not improve performance on 3 vs. 2 Keepaway, likely because the simple task can be learned well when all state variables are considered independently (Stone, Sutton, & Kuhlmann 2005).

The XOR Keepaway Task is learned using Sarsa (Rummery & Niranjan 1994; Singh & Sutton 1996), a well understood TD method that estimates the action-value function to predict the long-term expected return of taking a particular action in a particular state. We rely on CMACs (Albus 1981), a tile-coding function approximator that allows us to discretize a continuous state space by using tilings while maintaining the capability to generalize via multiple overlapping tilings. The number of tiles and width of the tilings define our representation parameterization and are hardcoded, dictating which state values will activate which tiles. The function approximation is learned by changing how much each tile contributes to the output of the function approximator. By default, all the CMAC's weights are initialized to zero. There can either be a separate CMAC for each state feature so that each is independent[2], or the CMACs can tile multiple state features together conjunctively.

Results

To master the XOR Keepaway task we use Sarsa to learn with CMAC FAs, with both independently and conjunctively tiled parameterizations. The independently tiled players use 13 separate CMACs, one for each state feature. The conjunctively tiled players use 10 separate CMACs, 9 of which independently tile state features and the last conjunctively tiles the 4 remaining state variables. We train the independently tiled players for 20 hours and then save the weights in their CMAC FAs. To get a small performance improvement, we set all zero weights to the average weight value, a method previously shown to improve CMAC performance (Taylor, Stone, & Liu 2005). We then train conjunctively tiled CMAC players, using the previously learned weights as needed as per Algorithm 1.

Agents learn best when the four relevant state features are conjunctively tiled: Figure 1 shows that players learning with conjunctive FAs outperform the players using independently tiled FAs. However, initially, agents using independently tiled state features are able to learn faster. Agents trained with independent CMACs for 20 hours can then transfer to a conjunctive representation via Algorithm 1, significantly outperforming players that only use the independent representation. A series of Student's t-tests confirm that Elaboration outperforms the conjunctively tiled players initially and then, after 55 hours of total training time, the conjunctively tiled players perform comparably. A series of Student's t-tests also show that the Elaboration trials outperform the individually tiled players after 40 hours of total training time. Thus, in the XOR Keepaway task, using Elaboration to transfer knowledge between two different representations outperforms using either representation alone for the the equivalent amount of time and allows the agents to achieve higher performance faster.

Related and Future Work

The idea of using multiple representations to solve a problem is not new. For instance, SOAR (Laird, Newell, & Rosenbloom 1987) uses multiple descriptions of planning problems to help with search and learning. Sherstov and Stone (2005) show that a tile coding may be changed gradually over time by adding tilings so that an agent may learn faster. Our work, rather than gradually changing a FA over time, aims to transfer a batch of knowledge from one FA to another.

[2]This approach to function approximation in the RoboCup soccer domain is detailed elsewhere (Stone, Sutton, & Kuhlmann 2005).

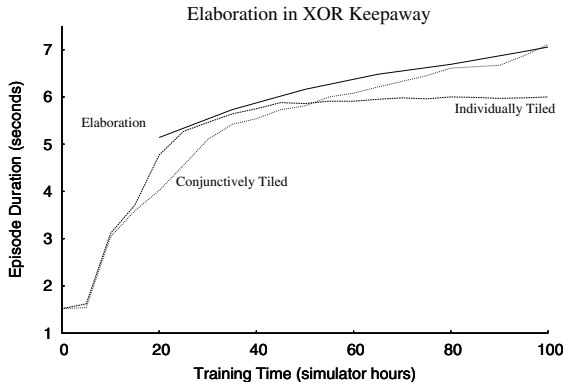

Figure 1: Averages from 10 independent trials of each of the 3 learning methods confirms that Elaboration outperforms learning with individually tiled CMACs and partially conjunctive CMACs.

One immediate goal is to demonstrate that RT algorithms may transfer between different FAs, such as from a CMAC to a neural network, rather than just different parameterizations of a single FA. In the future we would plan to test RT in more domains and with more representations. We would also like to discover ways of determining both *what* representation an agent should transfer to and *when* training is sufficient in the source representation to provide maximum benefit.

Acknowledgments

We would like to thank Cynthia Matuszek, Shimon Whiteson, Andrew Dreher, Bryan Klimt, Nate Kohl, and 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

Albus, J. S. 1981. *Brains, Behavior, and Robotics*. Peterborough, NH: Byte Books.

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. SOAR: An architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.

Rummery, G., and Niranjan, M. 1994. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG-RT 116, Engineering Department, Cambridge University.

Sherstov, A. A., and Stone, P. 2005. Function approximation via tile coding: Automating parameter choice. In Zucker, J.-D., and Saitta, I., eds., *SARA 2005*, volume 3607 of *Lecture Notes in Artificial Intelligence*. Berlin: Springer Verlag. 194–205.

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

Stone, P.; Kuhlmann, G.; Taylor, M. E.; and Liu, Y. 2006. Keepaway soccer: From machine learning testbed to benchmark. In Noda, I.; Jacoff, A.; Bredenfeld, A.; and Takahashi, Y., eds., *RoboCup-2005: Robot Soccer World Cup IX*, volume 4020. Berlin: Springer Verlag. 93–105.

Stone, P.; Sutton, R. S.; and Kuhlmann, G. 2005. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior* 13(3):165–188.

Taylor, M. E.; Stone, P.; and Liu, Y. 2005. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*.