A Framework for Transfer in Reinforcement Learning

George Konidaris GDK@CS.UMASS.EDU
Autonomous Learning Laboratory, Computer Science Dept., University of Massachusetts at Amherst, 01003 USA

Abstract

We present a conceptual framework for transfer in reinforcement learning based on the idea that related tasks share a common space. The framework attempts to capture the notion of tasks that are related (so that transfer is possible) but distinct (so that transfer is non-trivial). We define three types of transfer (knowledge, skill and model transfer) in terms of the framework, and illustrate them with an example scenario.

1. Introduction

One aspect of human problem-solving that remains poorly understood is the ability to appropriately generalise knowledge and skills learned in one task and apply them to improve performance in another.

Although reinforcement learning researchers study algorithms for improving task performance with experience, we do not yet understand how to effectively *transfer* learned skills and knowledge from one problem setting to another. It is not even clear which problem sequences allow transfer, which do not, and which do not need to. Although the idea behind transfer in reinforcement learning is intuitively clear, no definition or framework exists that usefully formalises the notion of "related but distinct" tasks—tasks that are similar enough to allow transfer but different enough to require it.

In this paper we present a framework for thinking about the transfer problem for reinforcement learning agents. The framework is based on the idea that related tasks share a common space, and it attempts to capture the notion of related but distinct tasks.

2. Related Tasks Share a Common Space

Successful transfer requires an agent that must solve a sequence of tasks that are related but distinct—different, but not so different that experience in one is irrelevant to experience in another. We propose that what makes a sequence of tasks related is the existence of a feature set that is shared and retains the same semantics across tasks. These features model the common elements between tasks, and we call the space generated by them an *agent-space* because it is associated with an agent, not an individual task.

The agent also requires a descriptor that is sufficient to distinguish Markov states in each individual task. This induces a task-specific Markov Decision Process (MDP) with a set of actions that are common across the sequence (because the agent does not change) but with a set of states, transition probabilities and rewards that refer to a particular task. We call each of these spaces a *problem-space*.[1] The core idea of our framework is that task learning takes place in problem-space, but transfer takes place in agent-space.

Consider a class of environments generated by an underlying parameterized environmental model (e.g., gridworlds with goal, size and obstacle location parameters) where the agent must solve a sequence of task instances $E^1, ..., E^n$ obtained by setting the model parameters. The agent may employ various observation functions that map an environment (described by its parameters) and its state to a real-valued vector. Particularly important are the reward functions, $r^1, ..., r^n$, which map each state in each environment to a single real-valued reward.

An observation function is a problem-space generator for environment E^m if the descriptors (s_t^m at time t) it produces are Markov for E^m. In general, an ideal problem-space is discrete and small. If we choose to use a function approximator, it should preferably be be smooth with respect to r^m (so that similar states have similar values).

An observation function is an agent-space generator if it is defined over and returns a descriptor of the same form for all of the environments. This requires an agent-space observation function that is a function of an environment description and its state, rather than just state as is standard

[1] We recognize that not all solution methods require the Markov property, but we treat this as the ideal case. More generally problem-space must provide enough structure to allow a solution to be found.

Appearing in the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning, June 29, Pittsburgh, PA.

in reinforcement learning. In general, an ideal agent-space contains all useful commonalities but may not necessarily be Markov.

We note that in some cases the agent-space and problem-spaces used for a sequence of tasks may be related, e.g. each problem-space is formed by appending a task-specific amount of memory to agent-space. However, in general it may not possible to recover an agent-space descriptor from a problem-space descriptor, or vice versa. They are separate observation functions which must be designed (or learned outside of the reinforcement learning process) with different objectives.

We define a sequence of tasks to be *related* if that sequence has an agent-space, i.e. if a set of (given or engineered) features exist in all of the tasks. We define the sequence to be *reward-linked* if every task has the same reward observation function ($r^m \equiv r, \forall m$) so that rewards are allocated for the same types of interactions in all tasks (e.g., reward is always x for finding food). Note that here r is a function of an environment description and its state (rather than only state as is standard in reinforcement learning), so it can be defined across multiple state spaces.

If a sequence of tasks is related we may be able to perform effective transfer by taking advantage of the shared space. If no such space exists there is no common way (however abstract, noisy or lossy) of describing states across the tasks. Without such a descriptor, we cannot perform transfer between arbitrary tasks in the sequence without more information.

If we can find an agent-space that is also a problem-space for every task in the sequence, then we can treat the sequence as a set of tasks in the same space and perform transfer directly by learning about the structure of this space. If in addition the sequence is reward-linked then the tasks are not distinct and transfer is trivial because we can view them as a single problem. However, there may be cases where a shared problem-space exists but results in slow learning, and using task-specific problem-spaces coupled with a transfer mechanism is more practical.

3. Types of Transfer

Given an agent solving n problems with respective state spaces $S_1, ..., S_n$, we view the ith state in S_j as consisting of the following attributes:

$$s_i^j = (d_i^j, c_i^j, r_i^j),$$

where d_i^j is the problem-space state descriptor (sufficient to distinguish this state from the others in S_j), c_i^j is an agent-space sensation, and r_i^j is the reward obtained at the state.

3.1. Knowledge Transfer

We can use reinforcement learning during each task S_j to learn a value function V_j:

$$V_j : d_i^j \mapsto v_i^j,$$

where v_i^j is the expected return for action from state s_i^j. This function is not portable between tasks because the form and meaning of d (as a problem-space descriptor) may change from one task to another. However, the form and meaning of c (as an agent-space descriptor) does not change, so we can perform knowledge transfer across the sequence by introducing a function L that estimates return for a state given the agent-space descriptor received there:

$$L : c_i^j \mapsto v_i^j.$$

L will only be a useful predictor of reward when there is a consistent relationship between some aspect of agent-space and reward across the sequence of tasks. Thus, knowledge transfer will only work in reward-linked tasks, where we can expect such a relationship because reward is always allocated for the same types of interactions.

Once an agent has completed task S_j and has learned V_j, it can use its (c_i^j, v_i^j) pairs as training examples for a supervised learning algorithm to learn L. Alternatively, learning could occur online during each task.

After training, L can be used to estimate a value for newly observed states in problem-space, thus providing a good initial estimate for V that can be refined using reinforcement learning. Alternatively (and equivalently), L could be used as an external shaping function (Ng et al., 1999).

Konidaris and Barto (2006a) show that such an agent is able to significantly improve its performance on a reference task after experience on even a single small training task.

3.2. Skill Transfer

Often, reinforcement learning agents are either given or learn a set of macro-actions to reduce the time required to solve their task. The options framework (Sutton et al., 1999) provides methods for learning and planning using *options* (macro-actions) in the standard reinforcement learning framework (Sutton & Barto, 1998). Each option o (as usually defined in problem-space) consists of the following components:

$$\begin{array}{lll} \pi_o : & (d_i^j, a) & \mapsto [0,1] \\ I_o : & d_i^j & \mapsto \{0,1\} \\ \beta_o : & d_i^j & \mapsto [0,1]. \end{array}$$

where π_o is the *option policy* (giving action probabilities at each state in which the option is defined), I_o is the *initiation* *set*, which is 1 for the states the option can be started from and 0 elsewhere, and β_o is the *termination condition*, which gives the probability of the option terminating in each state.

We can define portable agent-space options as:

$$\begin{array}{lll} \pi_o: & (c_i^j, a) & \mapsto [0,1] \\ I_o: & c_i^j & \mapsto \{0,1\} \\ \beta_o: & c_i^j & \mapsto [0,1]. \end{array}$$

Although the agent then learns its option policies in a different space from its task policy, it receives both agent-space and problem-space descriptors at each state so both policies can be updated simultaneously.

Konidaris and Barto (2006b) show that agents learning portable options are able to improve their performance in a reference problem through experience in other problems, until they approach the performance of agent with perfect prespecified problem-space options.

3.3. Model Transfer

Finally, it may be useful to learn a model of agent-space:

$$P: (c_i^j, a, c_k^j) \mapsto [0,1],$$

where P is probability of moving from one agent-space descriptor to another, given action a. This would primarily be useful for using model-based reinforcement learning methods to compute L or an agent-space option policy offline. Alternatively, it could be used to predict the results of executing an option and coupled with knowledge transfer to estimate the value of first executing an option.

4. An Illustrative Example

Consider a mobile robot, equipped with a laser range finder and pressure, light and temperature gauges, that must find heat sources in a sequence of different buildings.

Because its laser-range finder readings are noisy and non-Markov, the robot will need to construct a map of each building as it explores in order to search it efficiently. The resulting pose variables in a particular building's map are sufficient to form a problem-space for that building, but since all the buildings it is likely to see (and the location of the heat source in them) are different and unknown in advance, pose variables cannot be used for transfer. However, the robot's sensors retain their meaning across all buildings, so they form an agent-space for the building sequence.

The robot could attempt to learn a relationship between its sensors and the reward obtained when it finds the heat source (and hopefully learn that temperature is a heuristic for distance to the heat source), thus performing knowledge transfer. This would enable it to to later find heat sources in larger buildings in less time, but would not by itself be enough to find heat sources in new buildings because heat sensor readings are not Markov in problem-space.

The robot could also learn options using only the laser range finder, corresponding to actions like moving to the nearest door, thus performing skill transfer. Because these options are based solely on sensations in agent space without referencing problem-space (any individual metric map), they can be used to speed up learning and exploration in any building that the robot encounters in the future.

Finally, the robot could learn a model predicting the change in sensor outputs given an action, thus performing model transfer. For example, it could learn a forward model of its laser-range finder, and thus be able to learn agent-space options using model-based reinforcement learning methods.

5. Related Work

Most prior research on transfer in reinforcement learning involves either finding useful options (Bernstein, 1999; Pickett & Barto, 2002; Thrun & Schwartz, 1995) or building structured representations (Mahadevan, 2005; Van Roy, 1998) of a single state space, to speed up learning for tasks in the same state space but with different reward functions.

Taylor and Stone (2005) perform knowledge transfer from one instance of a multi-agent task to another with more agents (and hence a different state space). However, their approach uses a hand-coded transfer function that must be manually constructed for every pair of tasks in a sequence.

In supervised machine learning, transfer is viewed as an inductive bias mechanism that speeds learning and improves generality when learning multiple related tasks from the same feature set, demonstrating performance improvements using several methods (Thrun, 1996; Caruana, 1997; Evgeniou et al., 2005).

Baxter (2000) provides a theoretical framework for inductive bias learning over a distribution of related tasks. Silver and Mercer (2001) consider tasks related with respect to a learning algorithm when multitask learning using them improves its performance. In contrast, Ben-David and Schuller (2003) define related tasks to be those generated from each other under a given set of transformations. These methods all assume a single feature space for every task.

6. Discussion

Our framework requires the identification of a suitable agent-space to facilitate transfer, but it does not specify how that space is identified. This results in a design problem that is similar to that of standard state space design, but researchers in the reinforcement learning community have so far developed significant expertise at designing problem-spaces, not agent-spaces. We expect that the two design problems are equivalently difficult, and that with practice finding agent-spaces for related tasks will become easier.

It may be possible to automatically construct such spaces given a sequence of environments, but that will likely require explicit descriptions of the environments and their regularities across the sequence of tasks—information that is not available to reinforcement learning agents.

The idea of an agent-centric representation is closely related to the notion of deictic or ego-centric representations (Agre & Chapman, 1987), where objects are represented from the point of view of the agent rather than in some global frame of reference. We expect that for most problems, especially in robotics, agent-space representations will be egocentric, except in manipulation tasks, where they will likely be object-centric. In problems involving spatial maps, we expect that the difference between problem-space and agent-space will be closely related to the difference between allocentric and egocentric representations of space (Guazzelli et al., 1998).

7. Summary

We presented a conceptual framework for transfer in reinforcement learning. The framework attempts to capture the idea that for tasks to be related but distinct they must share an common space (an agent-space) but have different individual state spaces (problem-spaces). We defined three types of transfer (knowledge, skill and model transfer) in terms of the framework and illustrated them with an example transfer scenario.

Acknowledgments

We would like to thank Andy Barto, Pippin Wolfe and Özgür Şimşek for their useful comments. George Konidaris was supported by the National Science Foundation under Grant No. CCF-0432143. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Agre, P., & Chapman, D. (1987). Pengi: An implementation of a theory of activity. *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI 87)* (pp. 268–272).

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Ben-David, S., & Schuller, R. (2003). Exploiting task relatedness for multiple task learning. *Proceedings of the The Sixteenth Annual Conference on Learning Theory* (pp. 567–580).

Bernstein, D. (1999). *Reusing old policies to accelerate learning on new MDPs* (Technical Report UM-CS-1999-026). Department of Computer Science, University of Massachusetts at Amherst.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Evgeniou, T., Micchelli, C., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, *6*, 615–637.

Guazzelli, A., Corbacho, F., Bota, M., & Arbib, M. (1998). Affordances, motivations, and the world graph theory. *Adaptive Behavior*, *6*, 433–471.

Konidaris, G., & Barto, A. (2006a). Autonomous shaping: Knowledge transfer in reinforcement learning. *Proceedings of the Twenty Third International Conference on Machine Learning*.

Konidaris, G., & Barto, A. (2006b). *Building portable options: Skill transfer in reinforcement learning* (Technical Report UM-CS-2006-17). Department of Computer Science, University of Massachusetts Amherst.

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. *Proceedings of the Twenty Second International Conference on Machine Learning (ICML 05)*.

Ng, A., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: theory and application to reward shaping. *Proceedings of the 16th International Conference on Machine Learning* (pp. 278–287).

Pickett, M., & Barto, A. (2002). Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. *Proceedings of the Nineteenth International Conference of Machine Learning (ICML 02)* (pp. 506–513).

Silver, D., & Mercer, R. (2001). Selective functional transfer: Inductive bias from related tasks. *Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing* (pp. 182–189).

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Taylor, M., & Stone, P. (2005). Behavior transfer for value-function-based reinforcement learning. *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 53–59).

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *Advances in Neural Processing Systems 8* (pp. 640–646).

Thrun, S., & Schwartz, A. (1995). Finding structure in reinforcement learning. *Advances in Neural Information Processing Systems* (pp. 385–392). The MIT Press.

Van Roy, B. (1998). *Learning and value function approximation in complex decision processes*. Doctoral dissertation, Massachusetts Institute of Technology.