Multiple-Goal Reinforcement Learning with Modular Sarsa(O)

Nathan Sprague
Computer Science Department
University of Rochester
Rochester, NY 14627
sprague@cs. rochester. edu

Dana Ballard
Computer Science Department
University of Rochester
Rochester, NY 14627
dana@cs. rochester edu

Abstract

We present a new algorithm, GM-Sarsa(O), for finding approximate solutions to multiple-goal reinforcement learning problems that are modeled as composite Markov decision processes. According to our formulation different sub-goals are modeled as MDPs that are coupled by the requirement that they share actions. Existing reinforcement learning algorithms address similar problem formulations by first finding optimal policies for the component MDPs, and then merging these into a policy for the composite task. The problem with such methods is that policies that are optimized separately may or may not perform well when they are merged into a composite solution. Instead of searching for optimal policies for the component MDPs in isolation, our approach finds good policies in the context of the composite task.

keywords: reinforcement learning

1 Introduction

Traditional reinforcement learning algorithms can successfully solve small, single-goal tasks. The main challenge in the area of reinforcement learning is scaling up to larger and more complex problems. The scaling problem takes a number of forms. We may have a problem that has a very large state space, a problem that is best described as a set of hierarchically organized goals and subgoals, or a problem that requires the learning agent to address several tasks at once. It is the last form of scaling that this paper is concerned with.

The naive approach to learning to solve composite tasks is to create a state space that includes all of the information that is relevant to each sub-task. The agent would then learn in this joint space, receiving reward when any of the sub-goals are accomplished. The problem with this approach is that it suffers from the curse of dimensionality; as additional state dimensions are added for each new sub-task, the size of the joint state space grows exponentially.

A more promising approach to this sort of multiple-goal problem is to use the well known Q-learning algorithm to train one learning module to handle each of the sub-goals. The internal Q-values of the different learning modules can then be used to fairly distribute control among the modules. This approach has been independently explored in [Humphrys, 1996] and [Karlsson, 1997]. It is attractive in its simplicity, and it has shown good empirical performance in a number of domains.

In this paper we will highlight a previously unrecognized problem with existing modular Q-learning algorithms. Existing algorithms learn component policies that may be highly sub-optimal in the context of the composite task, because they do not take into account the fact that the component modules are forced to share control. We will show how to fix this problem by replacing the Q-learning with the closely related Sarsa(O) learning rule. The resulting algorithm shows improved performance on a large sample problem.

2 The problem formalized

The underlying formalism for many reinforcement learning algorithms is the Markov decision process. An MDP, denoted *M* is described by a 4-tuple (5, *A, T, R*), where *S* is the state space, *A* is the action space, and *T(s,* a, *s')* is the transition function that indicates the probability of arriving in state *s'* when action *a* is taken in state *s.* The reward function *R(s, a)* denotes the expected one-step payoff for taking action *a* in state *s.* The goal of reinforcement learning algorithms is to discover an optimal policy 7r*(s) that maps states to actions so as to maximize discounted long term reward.

Here we consider the problem of discovering a joint policy for a set of *N* MDP's $\{M_i\}_1^N$. Throughout we will use subscripts to distinguish the MDPs. These MDP's each have a distinct state space, but they share a common action space, and are required to execute the same action on each time step. This model is intended to map to the case of a single agent that is simultaneously faced with a set of different goals.

The *N* component MDPs implicitly define a larger composite MDP. Formally, the goal is to find the optimal policy for this composite MDP. The optimal composite policy is defined as the policy that maximizes summed discounted reward across the component MDPs.

The state space of the composite MDP is the cross product of the state spaces of the component MDPs: *S* — *S* x £2 x ... x *SN-* The composite reward function is defined as: *R(s, a)* = $\sum_{i=1}^{N} R_i(s_i, a)$. In the case where the component MDPs are independent, the composite transition function can be written as: $T(s,a,s') = \prod_{i=1}^{N} T_i(s_i,a,s_i')$. In the case where the component MDPs are not independent, the exact composite transition function will depend on on the particular dependencies between the models.

In theory, there is no reason that the composite MDP could not be solved directly using the traditional Q-learning algorithm. However, this is generally not practical because the size of the composite state space may grow exponentially with the number of component MDPs.

3 Modular Q-Learning

Humphrys and Karlsson [Humphrys, 1996; Karlsson, 1997] independently developed similar approaches to the problem of multiple-goal reinforcement learning. The idea is that a separate learning module is created for each component MDP. The agent takes actions in the environment, and each module *i* is trained with the standard Q-learning update rule:

$$Q_i(s_i,a) \leftarrow (1-\alpha)Q_i(s_i,a)+\alpha(r_i+\gamma \max_{a'} Q_i(s_i',a')) \quad (1)$$

Where i\ is the immediate reward, *a* is the learning rate parameter, and γ is a discount factor applied to future rewards. In single goal reinforcement learning problems, these Q-values are used only to rank order the actions in a given state. The key observation here is that the Q-values can also be used in multiple-goal problems to indicate the degree of preference that modules have for different actions. There are several possible ways these values can be used to select a compromise action to execute. The different approaches will be referred to as action selection mechanisms.

Karlsson's suggestion, which he calls "greatest mass", is to generate an overall Q-value as a simple sum of the Q-values of the individual modules: *Q(s,a)* — $\sum_{i=1}^{n} Q_i(s_i,a)$. The action with the maximum summed value is then chosen to execute. We will refer to this approach as GM-Q for greatest mass Q-learning.

Humphrys considers the greatest mass approach, but raises the objection that the action with the highest sum may not be particularly good for any of the modules, with the result that no module is able to reach its goal. He explores several winner-take-all alternatives that constrain the chosen action to be optimal for at least one module. For a given state .s each of the N modules promotes its own action with a value $W_i(s_i)$. The module with the largest *W* value is then allowed to execute its preferred action.

The simplest method for generating the W-values, which we will refer to as Top-Q, is to set $W_t(s_t)$ = $\max_a Q_i(s_i,a)$, thus giving control to the module with the highest Q-value in the current state. This method suffers from the drawback that the module with the highest Q-value may have no preference over what action is chosen, while another module stands to lose a great deal if its action is not selected. The method sometimes exhibits reasonable performance, but this is strongly dependent on the structure of the reward functions.

A better alternative, referred to as negotiated W-learning, is to grant control to the module that stands to lose the most long term reward if it is not selected. This module can be discovered by examining the Q-values for the current state. Refer to [Humphrys, 1996] for a detailed description of the algorithm.

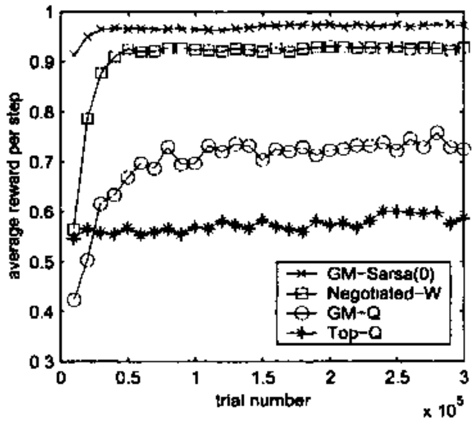

Figure 1: Results of one training run for GM-Sarsa(O) and three Q-learning based algorithms on a food gathering task. Training is divided into trials lasting 100 time steps. Data points are generated by suspending training every 10000 trials and computing the mean performance for 1000 trials without exploration. Each algorithm uses an e-grecdy exploration policy with e linearly reduced from .4 to 0 during the first half of trials. All algorithms use a fixed learning rate of .05, and a discount factor of .9.

3.1 The problem with modular Q-learning

Q-learning has some attractive qualities as a basis for multiple-goal reinforcement learning. Chief of these is the fact that it is an off-policy learning method. This means that Q-learning for a single MDP is guaranteed to converge to the optimal solution regardless of what policy is followed during training, as long as each state-action pair is visited infinitely often in the limit. This fact makes it easy to prove convergence results for the composite reinforcement learning algorithms introduced above. In particular, it is easy to see that each module is guaranteed to converge to the optimal policy and value function for its own MDP. Since the action selection mechanisms generate a policy deterministically from the component value functions, the composite policy is also guaranteed to converge, although there is no guarantee concerning the quality of the composite solution.

Unfortunately, the off-policy character of Q-learning is also a serious limitation. The difficulty is that the one-step value updates for each module are computed under the assumption that all future actions will be chosen optimally for that MDP. This assumption is not valid under the action selection mechanisms described above; future actions will represent some compromise policy in which the different modules share control. This means that the computed Q-values do not converge to the actual expected return under the composite policy. Instead, the max in equation (1) results in Q-values with a positive bias.

4 Modular Sarsa(O)

A possible solution is to the problem of positive bias is to replace Q-learning with an on-policy learning algorithm. In particular we will explore the use of Sarsa(O) [Rummery and

Niranjan, 1994; Singh and Sutton, 1996; Sutton, 1996]. The update rule for Sarsa(O) is:

$$Q_i(s_i, a) \leftarrow (1 - \alpha)Q_i(s_i, a) + \alpha(r_i + \gamma Q_i(s'_i, a')) \quad (2)$$

This update rule is virtually identical to that for Q-learning except that the max over Q-values on the right has been replaced with the Q-value of the state action pair that is actually observed on the next step. For the case of single MDPs Sarsa(O) has been proved to converge to the optimal policy as long as the exploration rate is asymptotically decayed toward zero according to an appropriate schedule [Singh *et al,* 2000].

The key observation for our purposes is that, since Sarsa(O) is an on-policy method, it does not suffer from the problem of positive bias. Since updates are based on the actions that are actually taken, rather than on the best possible action, we expect Sarsa(O) based modules to discover Q-values that are closer to the true expected return under the composite policy.

Any of the action selection mechanisms from Section 3 could be recast to use Sarsa(O) rather than Q-learning to train the modules. However, we focus on the method of greatest mass. We refer to the resulting algorithm as GM-Sarsa(O). Recall that the goal is to maximize the summed reward across all of the component MDPs. Assuming that we have trustworthy utility estimates from each of the modules, it makes sense to choose the action that has the highest summed utility across all of the modules. By definition, this is the action that will lead to the greatest summed long term reward. This reasoning did not hold under Q-learning, because the utility estimates were inaccurate under the composite policy.

Thus far we have no convergence proof for the GM-Sarsa(O) algorithm. Refer to the associated technical report [Sprague and Ballard, 2003] for a discussion of the possible convergence characteristics.

5 Examples

Figure 1 demonstrates the performance of GM-Sarsa(O) on a sample composite task (the task is adapted from [Singh and Cohn, 1998]). The goal of the agent in this task is to gather stationary food items while avoiding a predator in a 5 x 5 grid. There are three food items present at all times.

The agent moves in any of the eight possible directions at each time step. A random move is made with a probability of .1. If the agent contacts any of the food items it receives a reward of 1.0, and the item is randomly moved to a new position. The agent receives a reward of .5 for every time step that it avoids the predator. The predator moves deterministically one position toward the agent on every other time step.

The positions of the food items as well as the positions of the agent and predator result in 25^5 ` 10 million distinct states. This is too large for a monolithic tabular learning algorithm to be practical. The task is a good candidate for a modular reinforcement learning algorithm because it can be decomposed into several small MDPs. One MDP describes the agent's interaction with the predator, and three MDPs describe the interaction with the food items. Each of these component MDPs has $25^2 = 625$ states. Figure 1 shows the performance of GM-Sarsa(O) as well as the three Q-learning based algorithms from Section 3 on this task. Of the four algorithms, GM-Sarsa(O) exhibits the best performance.

6 Conclusion

We have presented a method for learning approximately optimal policies for a certain class of composite Markov decision processes. Empirical results demonstrate that our approach performs better than a number of existing algorithms. Future work will focus on proving convergence results for our algorithm. A longer version of this paper, including a discussion of related work is available as [Sprague and Ballard, 2003].

Acknowledgments

This material is based upon work supported by a grant from the Department of Education under grant number P200A000306, a grant from the National Institutes of Health under grant number 5P41RR09283 and a grant from the National Science Foundation under grant number EI A-0080124. 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 above mentioned institutions.

References

[Humphrys, 1996] M. Humphrys. Action selection methods using reinforcement learning. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior,* pages 135-144, Cambridge, MA, 1996.

[Karlsson, 1997] J. Karlsson. *Learning to Solve Multiple Goals.* PhD thesis, University of Rochester, 1997.

[Rummery and Niranjan, 1994] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-1NFENG/TR 166, Cambridge University Engineering Department, 1994.

[Singh and Cohn, 1998] S. Singh and D. Cohn. How to dynamically merge markov decision processes. In *Advances in Neural Information Processing Systems,* volume 10, 1998.

[Singh and Sutton, 1996] S. Singh and R. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning,* 22(1-3), 1996.

[Singh *et al,* 2000] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning,* 2000.

[Sprague and Ballard, 2003] N. Sprague and D. Ballard. Multiple-goal reinforcement learning with modular sarsa(O). Technical Report 798, University of Rochester Computer Science Department, 2003.

[Sutton, 1996] R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems,* volume 8, 1996.