Fixed vs. Dynamic Sub-transfer in Reinforcement Learning

James L. Carroll, Todd Peterson
Brigham Young University,
3350 Talmage bldg
Provo Ut. 84601 USA
james@jlcarroll.net, todd@cs.byu.edu

April 30, 2002

Abstract

We survey various task transfer methods in Q-learning and present a variation on fixed sub-transfer which we call dynamic sub-transfer. We discuss the benefits and drawbacks of dynamic sub-transfer as compared with the other transfer methods, and we describe qualitatively the situations where this method would be preferred over the fixed version of sub-transfer. We test this method against several other transfer methods in a simple three room grid world where portions of the source's policy are relevant to the target task and other portions are not. In this situation we found that dynamic sub-transfer converged to the optimal solution, avoiding the sub-optimality inherent in fixed sub-transfer, while also avoiding some of the convergence problems often experienced by fixed sub-transfer.

1 Introduction

In a reinforcement learning context task transfer is the process whereby information from one task, called the source task, is used by another task, known as the target task. Typically this is done by allowing the source task to bias the learning of the target task in the hope that this shared information can improve performance on the target task. Task transfer has many practical uses in reinforcement learning.

One use for task transfer is a process known as shaping [1][2]. Shaping increases the complexity of the tasks that an agent can acquire by learning simple tasks first. Through task transfer the source information from the simple task is used to aid in the learning of the next task which is more complex. This process can then be repeated in order to build up increasingly complex behaviors.

Task transfer is also important in the "lifelong learning" [3][4] paradigm for machine learning. In the lifelong learning approach, an agent encounters a variety of tasks over its lifetime. The agent can then apply information from tasks that it has already learned in order to speed the learning of each subsequent task. This approach can potentially reduce the training time and increase the adaptability of a reinforcement-learning agent. This approach is more complex than simple shaping because there is no constraint on the order in which tasks of various complexities are encountered.

In section 2 we discuss the previous work that has been done in task transfer, first summarizing Q-learning in general, and then giving a brief overview of several previous task transfer methods. We then discuss two of those methods, direct transfer and fixed sub-transfer, in greater detail. Section 3 introduces a new transfer method which we call dynamic sub-transfer. Section 4 describes our experimental methodology whereby we will compare these transfer mechanisms. Section 5 summarizes our experimental results. Sections 6 and 7 give our conclusions and discuss future work that we plan to do in this area.

2 Previous Work

Our reinforcement learning research has focused on a subset of reinforcement learning known as Q-learning. In Q-learning, the agent stores the online discounted expected reward for performing each action a in state s at time step t, and these "Q-values" are updated according to the equation:

$$\Delta Q(s_t, a_t) = \alpha[R(s_t, a_t) + \gamma max_a Q(s_{t+1}, a)],$$

where α is the learning rate and γ is a discount factor. Before learning can begin these Q-values must be initialized to some initial value which we call I. This value is important because it is often manipulated in the task transfer process. We shall discuss how this is usually done in greater detail later.

Many methods have been proposed to accomplish task transfer in Q-learning. Some methods are model-based and focus on transferring action models and other model information from one task to another. Other transfer mechanisms are model-free. We have focused our research on the model-free techniques. Some of the past model free techniques that are related to our research are:

- Direct-transfer[5][6][7][8][9] which uses the learned Q-values from the source task as the I for the target task
- Soft-transfer[7][8] which uses a weighted average of direct transfer and learning from scratch to make the transferred policy more adaptable
- Memory-guided exploration[7][8] which uses the past task to guide the initial exploration of the agent in the target task
- Fixed sub-transfer[5][6] which is a piecewise transfer mechanism that uses a portion of the source policy in the target task.

This paper introduces a new model-free technique for task transfer known as dynamic sub-transfer. Because direct transfer and fixed sub-transfer are directly related to our work on dynamic sub-transfer, we will first discuss those methods in more detail before we introduce dynamic sub-transfer.

2.1 Direct Transfer

Direct transfer of Q-values is the most straightforward method of performing task transfer in Q-learning. Rather than using a fixed value for I, Direct transfer takes the final Q-values from the source task, and initializes the target task with the Q-values from the earlier source task.

$$\forall s, \forall a, I_{target}(s, a) = Q_{source}(s, a).$$

In some situations direct transfer can perform poorly. When tasks are sufficiently dissimilar, direct transfer of Q-values can be much worse than learning from scratch [7]. This is because it often takes longer to unlearn the incorrect portion of the prior policy than it would take to learn the entire policy from scratch. This is known as the "unlearning" problem. Another problem with direct transfer is the "information loss" problem. Until the Q-values converge to their final values, valuable information can be lost. Even in the similar portion of the Q-space where correct information was transferred, incorrect updates can mar this correctly transferred information. This means that the agent sometimes loses all the relevant transferred information, while attempting to unlearn the irrelevant portion. We describe this situation in detail and give examples where this effect can be seen in [7].

2.2 Fixed Sub-transfer

Another model-free method for task transfer we call fixed sub-transfer. This method was introduced by Bowling and Veloso [6][5] and attempts to overcome the difficulties inherent in direct transfer. This method grew out of the SKILLS algorithm, which was introduced by Thrun and Schwartz in 1995 [10].

The SKILLS algorithm attempts to find partially defined action policies called skills which occur in more than one task. Skills are found using a description length argument. The SKILLS algorithm minimizes a function of the form

$$E = PL + \mu * DL,$$

where PL is a performance loss, and DL is a description length. By minimizing E this algorithm effects a

piecewise decomposition across multiple tasks. The skills so identified are the skills that are applicable across many separate tasks. It would be reasonable to use these partially defined policies over multiple tasks.

In fixed sub-transfer a portion of the source task is used in the target task. The portions that are used by the target task are those that have been deemed "similar." From the perspective of the target task the source task and the target task share the same Q-values in the similar section, and the Q-values are not updated in that section (they are fixed). In the portion that is not deemed similar, the agent initializes its Q-values to some I and learns these sections from scratch[6]. We have dubbed this technique fixed sub-transfer because only a portion of the source task is used by the target task and the Q-values in the similar portion are fixed.

There are several advantages to this approach aside from a reduced description length. This method avoids the unlearning problem because the sections that are not similar are not transferred. The information loss problem is also avoided because the correct portions of the policy are fixed.

However there are several drawbacks to this approach:

- Some method is required to determine which sub-skill(s) should be used by the target task. This usually requires "design intervention" because until the new task is learned it is difficult to determine its similarity with the skills which the agent already knows.
- Any technique that fixes portions of the Q-space is prone to divergence along the boundary between the fixed portions of the policy and the unfixed portion.
- Fixing portions of the policy often leads to a sub-optimal end policy. The sub-optimality of such a solution has been quantitatively bounded by [6][5].

Bowling and Veloso never mention how they dealt with the divergence issues. We chose the simple solution of updating all transitions that cross back into the fixed portion with an expected discounted reward of 0. This "Quick fix" would not work in all situations and these divergence issues are a major drawback of fixed sub-transfer. These problems should be more thoroughly addressed before this method could be used extensively in real world applications. As will be shown, dynamic sub-transfer elegantly sidesteps these divergence issues.

The design intervention necessary in fixed sub-transfer is problematic, but is not insurmountable, especially if the task has been broken down into sub-tasks by the SKILLS algorithm. It should be possible to notice from simple observation which of the automatically extracted skills are still valid in the new context. An example where this can be done, previously discussed by[6], is robot soccer. In robot soccer a user might notice that a policy learned in a simulator performs well in the real world when the agent is not shooting the ball, and is at a distance from the wall, but performs poorly otherwise. The sections away from the ball and the wall can therefore be deemed "similar."

3 Dynamic Sub-transfer

In dynamic sub-transfer the agent transfers information in the portions of the state space that are considered similar while initializing the rest of the state space to a fixed I. The portions that are transferred are not fixed as in fixed sub-transfer, but allowed to adjust normally.

We hypothesized that most of the speedup found in fixed sub-transfer came from its initialization, and not from the fixing of the policy; therefore we believed that this method would retain most of the learning-rate improvements seen with fixed sub-transfer while ensuring convergence, and removing any sub-optimality in the final solution.

The dynamic sub-transfer algorithm specifically deals with the problems of divergence and sub-optimality associated with fixed sub-transfer. To date this approach doesn't deal with the necessity of design intervention inherent in all sub-transfer methods. The user must still manually decide which portions of the state space to consider similar.

4 Methodology

To visualize how these transfer methods function we chose a simple illustrative task. We used the simple stochastic three-room maze world first introduced by Sebastian Thrun [10] and used by Bowling and Veloso [6][5] [see Figure 1]. This world has multiple positions for the start and the goal. We first learned the task with the start and the goal in the bottom positions and then moved the start and the goal to their counterpart positions toward the top.

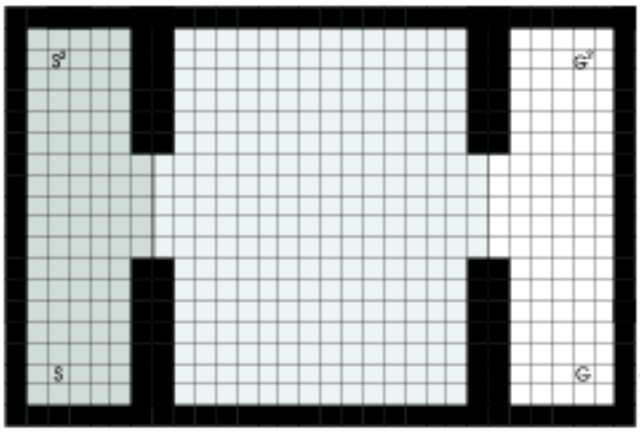

Figure 1: Simple three room stochastic environment.

The policies learned when the start and the goal are moved are nearly identical in the first room. In the second room the two policies are similar, and in the third room the policies are different. This synthetic environment models what happens in transfer when a portion of a policy changes while another portion of the policy remains the same, the situations where the sub-transfer techniques are useful.

5 Results

5.1 Direct Transfer

We felt that direct transfer should perform better than learning from scratch in this world despite others' experiments to the contrary [6] because the difference between the two policies is not great. We found that direct transfer did indeed outperform learning from scratch [see Figure 2]. However, other transfer methods that can outperform direct transfer need to be developed because others have shown that under other conditions direct transfer performs much worse than *tabula rasa* learning [7]. Furthermore, the direct transfer results in the three room experiment still leave much room for improvement.

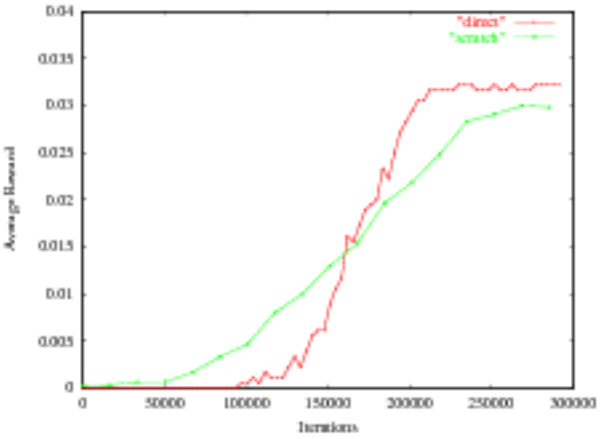

Figure 2: Direct transfer as compared to learning from scratch.

5.2 Fixed Sub-transfer

We compared our results for learning from scratch and direct transfer to the results obtained from fixed sub-transfer by fixing the policy in the left room only, and then by fixing the left two rooms [See Figure 3]. The results were similar to those obtained by [6] who performed the same experiment. Notice that when the first room is fixed there is a substantial improvement in learning rate. When the second room is fixed this improvement is even greater but the policy learned is sub-optimal.

In fixed sub-transfer the majority of the speedup in the learning rate comes from the initialization. In fixed sub-transfer direct transfer of Q-values is effectively performed on the portions of the policy that are similar, while the rest of the Q-space is initialized to a fixed I. This removes the difficulty inherent in the unlearning problem discussed in section 2. The

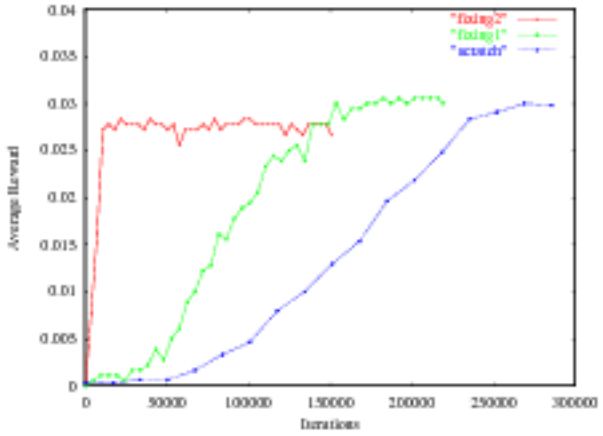

Figure 3: Fixed Sub-transfer: fixing the first two rooms, fixing the first room only, as compared to learning from scratch.

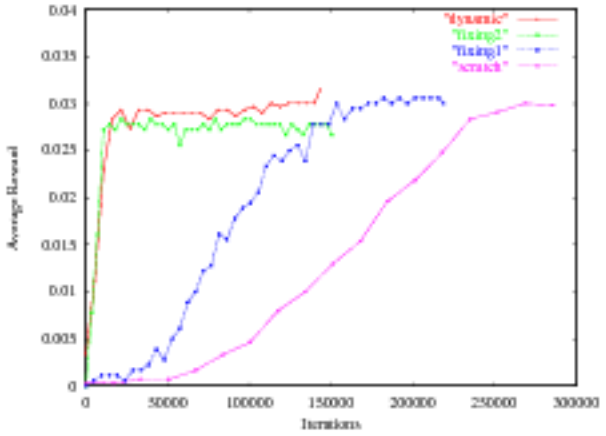

Figure 4: Fixed sub-transfer: fixing the first room, and first two rooms vs. dynamic sub-transfer and learning from scratch.

fact that the similar portions of the state space are fixed has less to do with the speedup, as information loss is not a major problem in this world. These observations lead us to dynamic sub-transfer.

5.3 Dynamic Sub-transfer

We tested dynamic sub-transfer and found that it had a learning rate comparable to those obtained through the fixed sub-transfer strategy in the three-room problem. The dynamic version lagged just a few time steps behind fixed sub-transfer because it wasted some time with the information loss problem. More importantly, the agent achieved an optimal policy over time [See Figure 4]. Thus, in the three room problem, at least, dynamic sub-transfer is preferable. The amount of time wasted by information loss was small and could possibly have been recovered by temporarily fixing the similar portion, and then releasing it.

6 Conclusions

In cases where the designer knows which portions of the policy are similar beforehand the sub-transfer methods are preferable to direct transfer, soft transfer, or memory guided exploration, while the latter methods are preferable when such design intervention needs to be avoided. Fixed sub-transfer allows an agent to use one skill to represent the policy in multiple tasks. In many cases dynamic sub-transfer is preferable to fixed sub-transfer in that it learns just as quickly, but doesn't converge to a sub optimal policy.

In worlds where states have vastly different Q-values in comparison to their neighbors information loss is more of a problem, and fixed-transfer may perform better than dynamic sub-transfer [7]. In such situations a hybrid method may be most effective. It is also possible to fix portions of the policy initially, and then release them later, allowing the agent to adjust after the new portion has been learned. This avoids the sub optimality found in fixed sub-transfer while retaining all of the speedup in the learning rate associated with fixed sub-transfer.

7 Future Work

Methods for automatically determining sub-problem similarity need to be developed, and hybrids of the above methods should be explored. In worlds where information loss is a serious issue, portions of the pol-

icy should be fixed until the agent has learned the rest of the state space to avoid information loss. These portions could then be released to allow the agent to adjust, thereby avoiding sub-optimality while retaining a fast learning rate. This process could perhaps be automated by watching the change in average reward, and releasing the fixed portions when it levels out.

A better method for dealing with the divergence issues involved with fixed sub-transfer needs to be developed. This is important because the current methods only function in select, carefully controlled situations.

Once an adequate feel for the various strengths and weaknesses of these methods has been reached and their effects have been quantified, we believe that the future course of this research should be to automate the entire transfer mechanism. When an agent encounters a new situation it should automatically determine that a new task is necessary. Then the agent should automatically choose the best transfer mechanism for the current situation, and adapt that mechanism as more information becomes available. Any such agent should incorporate the model-based approaches to task transfer as well as the model-free approaches.

References

[1] B. F. Skinner, *The Behavior of Organisms: An Experimental Analysis*, Prentice Hall, Englewood Cliffs, New Jersey, 1938.

[2] B. F. Skinner, *Science and Human Behavior.*, Colliler-Macmillian, New York, 1953.

[3] Sebastian Thrun and Tom M. Mitchell, "Lifelong robot learning," *Technical Report, IAI-TR-93-7.*

[4] Rich Caruana, "Multitask learning," in *Learning to Learn*, Lorien Pratt Sebastian Thrun, Ed., chapter 5, pp. 95–1133. Kluwer Academic Publsishers, Norwell Massachusetts, 1998.

[5] Michael H. Bowling and Manuela M. Veloso, "Bounding the suboptimality of reusing subproblem," in *IJCAI*, 1999, pp. 1340–1347.

[6] Mike Bowling and Manuela Veloso, "Reusing learned policies between similar problems," in *Proceedings of the AI*AI-98 Workshop on New Trends in Robotics*, Padua, Italy, October 1998.

[7] Nancy Owens Todd Peterson and James L. Carroll, "Automated shaping as applied to robot navigation," in *IEEE International Conference on Robotics and Automation*, Korea, 2001.

[8] Todd Peterson James L. Carroll and Nancy Owens, "Memory-guided exploration in reinforcement learning," in *In IJCNN2001*, Washington, D.C., 2001.

[9] Kevin R. Dixon, Richard J. Malak, and Pradeep K. Khosla, "Incorporating prior knowledge and previously learned information into reinforcement learning agents," in *Institute for Complex Engineered Systems Technical Report Series*, Carnegie Mellon University, January 2000.

[10] Sebastian Thrun and Anton Schwartz, "Finding structure in reinforcement learning," in *Advances in Neural Information Processing Systems 7*, D. Touretzky G. Tesauro and T. Leen, Eds., 1995, p. pages 385392.