Memory-guided Exploration in Reinforcement Learning

James L. Carroll, Todd S. Peterson & Nancy E. Owens

Machine Intelligence, Learning, and Decisions Laboratory
Brigham Young University Provo Ut. 84601 USA
james@jlcarroll.net, todd@cs.byu.edu, owens@cs.byu.edu

Abstract

The life-long learning architecture attempts to create an adaptive agent through the incorporation of prior knowledge over the lifetime of a learning agent. Our paper focuses on task transfer in reinforcement learning and specifically in Q-learning. There are three main model free methods for performing task transfer in Q-learning: direct transfer, soft transfer and memory-guided exploration. In direct transfer Q-values from a previous task are used to initialize the Q-values of the next task. Soft transfer initializes the Q-values of the new task with a weighted average of the standard initialization value and the Q-values of the previous task. In memory-guided exploration the Q-values of previous tasks are used as a guide in the initial exploration of the agent. The weight that the agent gives to its past experience decreases over time. We explore stability issues related to the off-policy nature of memory-guided exploration and compare memory-guided exploration to soft transfer and direct transfer in three different environments.

1 Introduction

There are several major problems that have been encountered by researchers as they have attempted to create intelligent agents. These problems mostly center on the availability of good data, training time, and overfitting.

[12] has identified the following four major machine learning bottlenecks: knowledge (a need for accurate information and models), engineering (information must be made computer accessible), tractability (complexity of robot domains), and precision (imprecise sensors and actuators cause differences between the real world and models, simulators, or plans).

1.1 Proposed solution

One approach to overcoming these problems is with the "life-long learning" paradigm for machine learning [12][2][11][5]. In the life-long learning approach an agent encounters many different tasks over its lifetime. These tasks are often related to each other. Through the appropriate transfer of knowledge it is possible for an agent to learn each new task with greater efficiency. Less training examples would be required because the agent could use past experience to fill in the holes in its training set. Accurate models would be learned, and transferred. Such knowledge would be computer accessible because it would be learned, and the need to hard code such information would be reduced. By transferring information from simpler problems to more complex problems the tractability issues would be significantly reduced. Learning tends to adapt automatically to precision problems in the sensors and actuators. In short many of the problems in machine learning could be solved or simplified through the life-long learning paradigm.

For any life-long learning system to succeed it will be important for an agent to be able to transfer information successfully between tasks. Task transfer requires the agent to automatically determine which features from previous tasks are significant to the new task to be learned, and which are not. Appropriate task transfer is a complex problem and it lies at the very heart of life-long learning. This paper focuses on task transfer in reinforcement learning, specifically in Q-learning.

2 Current Transfer Methods

If the agent's actuators and sensors remain constant, there are two situations that could be encountered when attempting task transfer. Either the environment will have changed, or the reward structure will have changed [12]. If the environment has changed, the state transition probabilities will no longer be the same. Alternately if the reward structure has changed, the state transition probabilities will be the same but the reward received for taking those transitions will have changed. Combinations of these two situations can also be encountered. If the agent itself changes, then the actual state space can grow or contract (if preceptors are added or removed), or the agent's action space can grow or contract (if actuators are added or removed).

Task transfer mechanisms can be thought of as algorithms that 1) attempt to determine invariants among all tasks, 2) generalize invariant information among various groups of tasks, 3) determine group membership of new tasks, 4) determine task specific information, and 5) allow specialization in those areas where the tasks differ. In general such algorithms should allow the agent to learn a new task with greater speed than learning that task from scratch. When encountering a task unlike any the agent has encountered before we should allow the agent to take slightly longer than when learning from scratch. This is understandable because we would expect the agent to explore areas of similarity between the current and past tasks first. However we believe that any appropriate task transfer mechanism should be guaranteed to eventually learn any learnable task, regardless of the amount of incorrect or irrelevant prior information given to the agent.

2.1 Q-learning

Q-learning is a reinforcement learning method where the agent stores the expected discounted reward for performing action a in state s $Q(s_t, a_t)$ updated by

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

where R is the reward and t is the time step, α is the learning rate and γ is the discount factor. Q-values are typically initialized to some constant value $Q(s, a) = I$.

Some methods of incorporating prior knowledge in Q-learning have involved building a model of the environment [7], biasing the inputs [9] or changing other aspects of a function approximator [4]. [8] suggested that transfer of information from one task to another in Q-learning can be accomplished by stringing together solutions to elemental sequential sub tasks. Unfortunately this requires *design intervention.*[1] to determine each elemental task.

A common model free task transfer method in Q-learning is to use the Q-values from the first task to initialize the Q-values of the second task. This method is called direct transfer of Q-values. Other model free methods for task transfer in Q-learning that we developed include memory-guided exploration [6] and what we call "soft transfer." We discuss these three methods and their applications in three different environments.

[1]manual intervention by the designer of the system

2.1.1 Direct Transfer. One simple way to transfer information from one Q-learning task to another is by using the Q-values from the first task to initialize the Q-values of the second task, then allowing normal updates to adjust to any differences between the two tasks.

If the problems are too dissimilar the agent will spend too much time unlearning the incorrect portion of the policy causing the agent to learn the new task slower than learning it from scratch [6][1]. Also as the agent unlearns this incorrect information it often unlearns correct information as well. This usually happens as γ discounts Q-values along the path to the goal and before the reward from the new goal has back-propagated sufficiently to overcome this situation. We call this the unlearning problem. For direct transfer to function well the two tasks must be sufficiently similar, which limits the applicability of the method.

2.1.2 Soft Transfer. One way to make direct transfer more robust to differences in the past task is through soft transfer. Soft transfer preserves the current policy while softening the Q-values making them easier to change. This is a similar idea to that of avoiding overfitting in many other machine learning problems. This is accomplished by initializing the Q-values using a weighted average between the old Q-values and the standard tabula rasa I thus:

$$Q_0^{new}(s, a) = (1 - W)I + W(Q_F^{old}(s, a)),$$

where Q_0 is the initial Q-value, and Q_F is the final Q-value. If $W = 1$ this is the same as direct transfer and if $W = 0$ then agent learns from scratch. By picking $0 < W < 1$ the agent can unlearn the incorrect portions of its policy easier. Care needs to be taken to set this parameter appropriately. If W is too high the agent will spend too much time unlearning the incorrect information in the past task, if it is too low the agent will loose useful information.

2.1.3 Memory-guided Exploration. Memory-guided exploration stores two or more sets of Q-values $Q^{old}(s, a)$ and $Q^{new}(s, a)$. One set of Q-values represents the agent's current experience in the new task. The other set of Q-values represents the stored memory of a past task. The old Q-values are never changed, preventing the accidental loss of pertinent information. The new Q-values are updated normally as the agent explores its environment, reducing the time necessary to "unlearn" the incorrect portions of the policy [6].

A process similar to memory-guided exploration was independently developed by [3]. They used a switching mechanism to choose between the agents prior task and the current task. In our version actions are chosen based on a weighted sum of random noise, with the prior and current Q-values

$$a_t = max_a[W \times Q^{old}(s_t, a) + (1 - W)Q^{new}(s_t, a) + \tau],$$

where W (the weights of the old Q-values) and τ (the random exploration noise) decay over time. Thus the old Q-values are used only to guide the agent's initial exploration. Memory-guided exploration is an off-policy controller [10], and this raises convergence issues. These issues can be overcome, and we will discuss them in greater detail later. However once W decays sufficiently all the standard convergence proofs hold, thus this method is guaranteed to eventually converge.

3 Methodology

We tested these task transfer methods on three different Markov decision tasks. These tasks illustrate different properties of various transfer mechanisms. For each task we will discuss the issues of unlearning, stability and convergence.

3.1 Tree-structured Decision Task

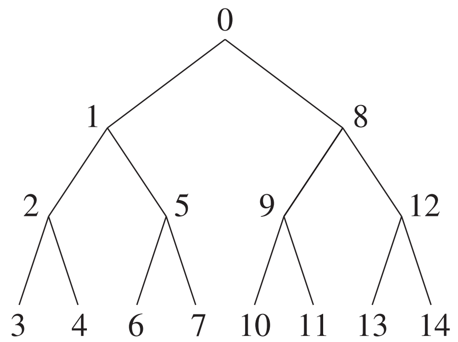

Figure 1: Simple analysis task. Agent starts task at node 0 and terminates at one of the nodes at the bottom of the tree.

Figure 1 shows a simple decision task. In this world the agent starts at node 0 and terminates at one of the leaf nodes of the tree. The goal is placed in one of the leaf nodes, and the agent's task is to find it. The agent receives a reward of +1 if it finds the goal and a reward of 0 if the agent terminates in a non-goal state. Once the agent reaches a leaf node the task begins over, and the agent is placed at the top of the tree. The transitions are deterministic. The tasks varied in depth d and branching factor b and can thus be made of arbitrary complexity.

In the first phase of training the agent is trained with the goal in a random location. Once the agent has learned this task the task is changed slightly by moving the reward a small distance. In this second phase the agent's task is to adapt to this new situation. The closer the new goal is to the old goal the more the policies of the two tasks will overlap and the more "similar" the two problems will be.

This task is important because of the theoretical framework it provides for analyzing various transfer mechanisms. This type of task is similar to problems like chess, where similar states have vastly different utilities, and the agent is not allowed to return to prior states. States near the old goal often have relatively low Q-values even though there is a substantial overlap in the policy necessary to reach those states. This makes task transfer extremely difficult, and many task transfer algorithms perform differently in this environment than they would in environments where the Q-values near the old goal are comparatively high. For this reason we believed that directly transferring the Q-values from the first phase to this second phase would cause the agent to perform poorly in this environment while memory-guided exploration would perform well [6].

Because of the unidirectional nature of this task, convergence was not an issue in this world like it was in some of the other worlds we investigated.

3.2 Simple Grid Task

We also tested the agent in a simple accessible maze world (see Figure 2). This world has very different properties than the decision task. States near the old goal tend to have relatively high Q-values, and therefore we would expect direct transfer to perform better in this world than it does in the decision world.

The agent's task was to navigate from the bottom left to the goal, while avoiding the walls. A reward of a -1 was given when the agent hit a wall, and a reward of a $+1$ was given when the agent found the goal. Both a stochastic and a deterministic version of this world were tested.

To test task transfer the agent initially learned this

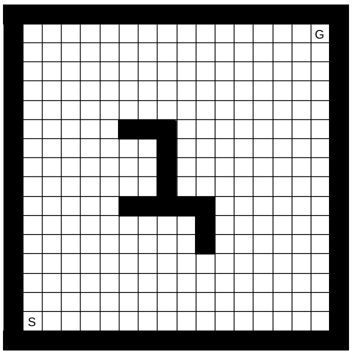

Figure 2: Grid world. Versions of this world were tested with obstacles, without obstacles, and of varying widths.

task with the goal in the upper left then the new goal was moved various distances to the right. This allowed us to test transfer mechanisms on problems of varying similarity. The further to the left the new goal is the more similar the two problems will be. As the goal is moved to the right, the problems become increasingly dissimilar until the goal is in the upper right (as shown in Figure 2).

The ability of the agent to return to previously visited states created divergence problems with some off-policy controllers that were not present in the decision world. This allowed us to test the stability of various algorithms generally, and memory-based exploration specifically. These problems will be discussed in greater detail in section 4.

3.3 Nomad II Robots

In order to test transfer mechanisms in a more realistic setting, the agent was trained in a simulator for Nomad II Robots on a wall avoidance task. This task was then transferred to a wall following task. The environment was simply a large empty room. The agent's sensors consisted of eight sonars. This environment is approximately continuous, and the state space is large, therefore the agent used a CMAC as a function approximator. The memory-guided convergence issues encountered in the grid were not manifest in this environment.

4 Convergence Issues

The off-policy nature of memory-guided exploration caused three specific problems: 1) required exploration bias 2) unvisited transitions and 3) local exploration vs. global exploitation. These issues needed to be dealt with before results could be obtained in the grid world for memory-guided exploration.

4.1 Required exploration bias

Memory-guided exploration requires an additional bias towards unexplored areas. W decays asymptotically toward 0. The purpose of decaying W is to allow the agent to explore additional areas of the state space. However the agent will still have some bias towards the previous policy because W will not fall all the way to zero. In order to insure sufficient exploration as W decays, some additional weaker bias toward the unexplored areas of the state space must dominate, otherwise the agent will never leave the location of the previous goal.

This problem is best illustrated when $I = 0$, and can be easily solved by choosing $I > 0$. This biases the agent's exploration away from areas where he has already been because the Q-values will be discounted each time they are visited. If I is chosen to be 0 this can not happen because $0 \times \gamma = 0$. Alternately, any other type of exploration bias could be employed, such as recency or counter-based exploration [13]. Care needs to be taken to ensure that this additional bias is initially smaller than the bias towards the prior policy.

4.2 Unvisited transitions

Off-policy controllers can diverge from the true Q-values, unless the controller is a random Monte Carlo controller [10]. This happens because the prior knowledge bias causes the agent to visit the states according to a different distribution than would the policy that maximizes the current reward function [3]. Effectively certain transitions never get updated.

As an example, the memory biases the agent away from obstacles (especially when τ is low or 0). This behavior is one of the major advantages of this method. We wanted the agent's memory to be strong enough to insure that the agent wouldn't explore transitions that used to map to obstacles until it had first explored all other options. The Q-values mapping to obstacles will remain at i and the Q-values surrounding obstacles will therefore not drop below $I \times \gamma^d$ where d is the distance from the obstacle. This means that when these un-updated Q-values are the maximum for that state, they cause the Q-values of all neighboring states to be inflated. This effect remains until W drops sufficiently to allow these states to be visited. Effectively this creates an artificial bias towards states next to obstacles because transitions into those obstacles are never updated.

It was not immediately clear whether these issues would effect memory-guided exploration because the agent's memory of old tasks only effects the agent's initial exploration. Convergence is assured once W decays to insignificance [3]. However we found that the agent's behavior was often divergent until the decay of W, at which time the agent would behave like an agent learning from scratch. Although the agent eventually converged, the advantages of the transfer were nullified.

There are two solutions to this problem. The first is to allow the old Q-values to bias the update of the new Q-values by the same amount that the old Q-values affect exploration. The update equation becomes:

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

Where $Q^C = W * Q^{old}(s_{t+1}, a) + (1-W)Q^{new}(s_{t+1}, a)$. In effect this creates a new "composite" on-policy controller.

The other option is to keep local values for W. This is effective because divergent behavior manifests itself as repeated visits to the same few states in an infinite loop. Because W decays each time the agent visits a state, the local values of W can be decayed in areas of divergence while remaining high for the rest of the state space. This allows the agent to behave like an on-policy controller in areas of divergence, while allowing the agent the exploration advantages of behaving as an off-policy controller in areas where divergence is not an issue.

4.3 Local Exploration vs. Global Exploitation

There is another reason to keep W local. The true max old Q-values are approximately γ^d where d is the shortest path to the old goal. This means that in areas of the state space that are far from the old goal, the difference between the Q-values of the best choice and the second best choice is often very small, whereas these differences are comparatively high near the old goal. Therefore the agent's memory bias is stronger near the old goal than it is in states further away.

If W is not local the agent will stay near the old goal until the global W has decayed sufficiently for the agent's exploration bias to overcome the agent's strong memory bias at the old goal location. Because the agent's memory bias is weaker near the start, when the agent is moved to the start it will begin learning tabula rasa. Thus the agent has lost any exploratory advantage that could have been drawn from its memory bias in the rest of the state space.

Keeping W local to each state solves this problem. When W is local the agent moves to the old goal, and stays there until the local W decays sufficiently at which time the agent begins exploration, while W remains high and allows the agent to exploit the old policy in other areas of the state space.

5 Results

5.1 Decision World Results

As noted in [6] memory-guided exploration performed substantially better than direct transfer in this world. This occurred because the actual Q-values of all transitions that did not terminate in a goal state were 0. With direct transfer the agent often lost all information along the path to the old goal. This is the unlearning problem discussed in [6]. This means that even when the tasks are very similar direct transfer performs slightly worse than learning from scratch. Memory-guided exploration on the other hand performs an order of magnitude better, and only performed slightly worse than tabula rasa learning when the new task was completely dissimilar to the old task.

5.2 Grid World Results

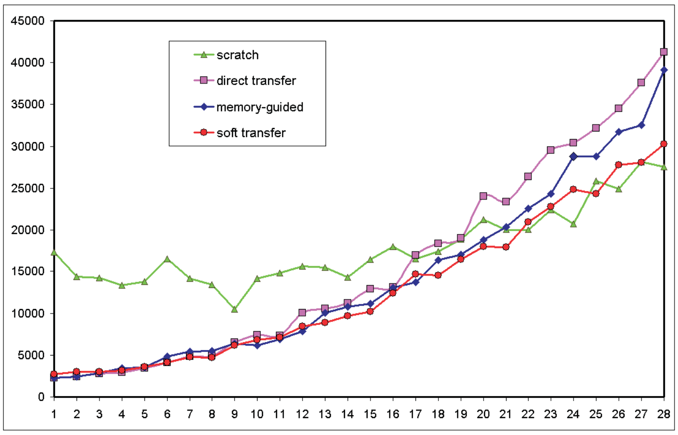

Figure 3: Steps to find goal 150 times over distance goal was moved for all 3 transfer versions and learning from scratch in the deterministic version of the grid world.

Local values of W were used with memory-guided exploration, and for soft transfer we chose a value .5 for W. In the deterministic version of the world, without any obstacles all three methods of task transfer were superior to learning the task from scratch until the tasks became sufficiently dissimilar (see Figure 3). This is in contrast to the results in the decision world, where direct transfer was worse even when the tasks were similar. In the grid world with the increasing dissimilarity of the tasks, direct transfer performed poorly, memory-guided exploration performed better, and soft transfer performed best. We also tested this world with obstacles and with stochastic transitions with similar results.

5.3 Nomad II Robot Results

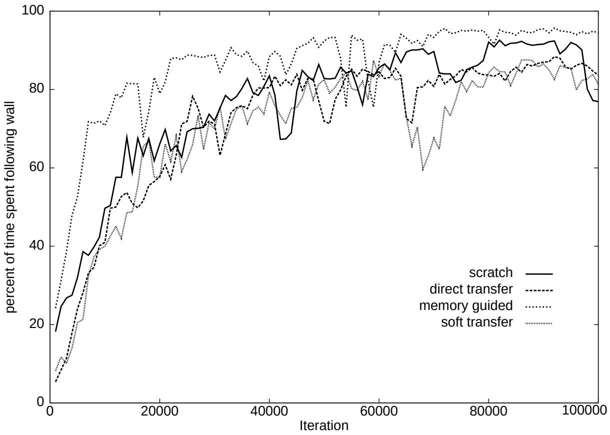

Figure 4: Average of 3 runs each, for learning from scratch, direct transfer, soft transfer and memory guided exploration on the Nomad II simulator transferring wall avoidance to wall following.

In the Nomad II simulator we found that memory-guided exploration performed better than learning the task from scratch (see Figure 4). Interestingly enough soft transfer and direct transfer were worse than learning the task from scratch in this environment. These results can vary somewhat from run to run. We believe that this was due to differences in the wall avoidance policy learned in the previous task. There are many effective policies for wall avoidance that could be learned. One policy for wall avoidance might transfer to wall following better than another, and different mechanisms for task transfer may transfer one policy better than another.

6 Conclusions and future work

Memory guided exploration performed better than tabula rasa learning in almost every situation tested, where the more similar the situation the greater the effect. Memory guided exploration was also superior to the direct transfer of Q-values. More research is needed to determine when soft transfer performs better. This paper has focused on model free methods, model based methods should work well with task transfer and should be explored further.

Additional methods for soft transfer and memory-guided exploration could be explored such as initializing W such that the values of W closer to the old goal are low, while values of W closer to the start are high. Also, it may not be necessary to keep a unique value for W in each state, a local function approximator may be sufficient. Memory-guided exploration could also serve as a method for creating an advice taking agent, where the agent's initial exploration is handled through direct control. Methods that use multiple past tasks should be explored, and memory-guided exploration should be expanded to handle multiple past tasks.

References

[1] M. Bowling and M. Veloso. Reusing learned policies between similar problems. In *Proceedings of the AIIA-98 Workshop on new trends in robotics.* Padua, Italy, 1998.

[2] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[3] Kevin Dixon, Richard Malak, and Pradeep Kosla. Incorporating prior knowledge and previously learned information into reinforcement learning agents. Technical report, Institute for Complex Engineered Systems, Carnegie Mellon, 2000.

[4] Tom M. Mitchell and Sebastian Thrun. Explanation based learning: A comparison of symbolic and neural network approaches. In *International Conference on Machine Learning*, pages 197–204, 1993.

[5] J. O'Sullivan. Transferring learned knowledge in a lifelong learning mobile robot agent. In *7th European Workshop on Learning Robots*, 1999.

[6] Todd Peterson, Nancy Owens, and James Carroll. Automated shaping as applied to robot navigation. In *ICRA2001*, San Diego, CA, 2001. In Press.

[7] J. A. Meyer R. Pfeiffer, B. Blumberg and Jurgen Schmidhuber S. W. Wilson (eds.) Marco Wiering. Efficient model-based exploration. In *Proceedings of tje Fifth International Conference on Simulation of Adaptive Behavior.*, 1998.

[8] S. P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.

[9] P. Stone and M. Veloso. Layered learning. In *Eleventh Europeaning Conference on Machine Learning*, 2000.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning An Introduction.* MIT Press, Cambridge Mass, 1998.

[11] Sebastian Thrun and Tom M. Mitchell. Learning one more thing. Technical report cmu-cs-94-184, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1994.

[12] Sebastian Thrun and Tom M.Mitchell. Lifelong robot learning. Technical report iai-tr-93-7, Institute for Informatics III, University of Bonn, Bonn, Germany, July 1993.

[13] Sebastian B. Thrun. The role of exploration in learning control. In D. White and D. Sofge, editors, *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches.* Van Nostrand Reinhold, Florence, Kentucky 41022, 1992.