Reinforcement Learning Task Clustering (RLTC)

James L. Carroll
Brigham Young University,
Provo Ut. 84601 USA,
james@jlcarroll.net

Todd Peterson
Utah Valley State College,
Orem, Ut. 84058 USA,
petersto@uvsc.edu

Kevin Seppi
Brigham Young University,
Provo Ut. 84601USA,
kseppi@cs.byu.edu

April 21, 2003

Abstract

This work represents the first step towards a task library system in the reinforcement learning domain. Task libraries could be useful in speeding up the learning of new tasks through task transfer. Related transfer can increase learning rate and can help prevent convergence to sub-optimal policies in reinforcement learning. Unrelated transfer can be extremely detrimental to the learning rate. Thus task transfer is useful in reinforcement learning if the source task and the target task are sufficiently related. Task similarity in reinforcement learning can be determined using many different similarity metrics, and simple clustering mechanisms can be applied to determine a set of related tasks. Invariants can be determined among the set of related tasks and then used in transfer. This paper uses information gathered from a set of simple grid world tasks to show that clustering of tasks based upon a similarity metric can be helpful in determining the set of source tasks which should be utilized in transfer.

1 Introduction

Reinforcement learning is a method for learning control by maximizing a reward function. Q-learning is one technique for reinforcement learning. In Q-learning the expected discounted reward for taking an action a in a state s is stored as what is known as a Q-Value denoted $Q(s, a)$. The Q-values are learned through temporal difference learning with repeated iterations with the world. Appropriate Q-values eventually backpropagate from the rewards throughout the rest of the state space. Unfortunately, Q-learning is often slow and fails to scale to more complicated problems.

Several methods for overcoming these problems have been proposed. In task transfer information is transferred from one task, known as the source task, to speed or otherwise improve the learning of another task, known as the target task. Thus the source task provides a type of learning bias for the second task. In general no restrictions are placed upon the relative complexities of the source and the target tasks. Shaping is a special case of task transfer which attempts to scale reinforcement learning to more complicated domains by first training an agent on simple tasks, and then attempting to use task transfer in order to transfer information to a more complex target task [1] [2]. The hope is that this will speed the learning of the more complicated target task.

Task transfer is more complex than simple function approximation, or generalization because the difference between tasks often can not be clearly parameterized. Further, the reward function itself can be completely unrelated from task to task. Several different mechanisms for task transfer in model free reinforcement learning have been proposed [3] [4] [5] [6] [7]. We assume a general understanding of these

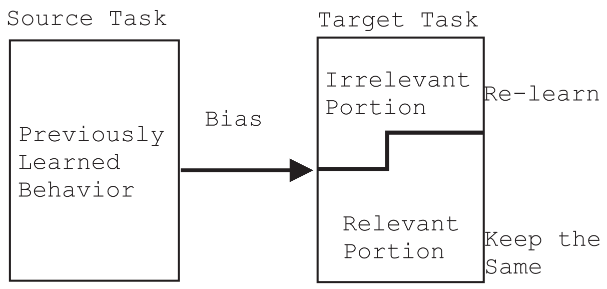

Figure 1: A source task can bias the learning of a target task.

techniques, and do not cover them in detail in the interest of space. Unfortunately, all these mechanisms work well for tasks that are related to each other, while performing poorly in situations where the two tasks are less related. In some situations this performance degradation is exponential on the number of differences, or the magnitude of the differences between the source and the target tasks [4]. This is because the agent must unlearn any information in the portion of the target task that is not similar to the source task (known as the "unlearning problem") and because during unlearning valid information is often accidentally lost in sections where the target task is similar to the source task (known as the "information loss problem"). Therefore, the bias must be flexible enough to allow adaptation in the section that must be unlearned, while not loosing information from the section where this bias could be useful [See Figure 1]. Unfortunately, this is an extremely difficult problem, and much research has gone into it. Most of this research has focused on the case where there is one source task and one target task.

2 Task Library System, The RLTC Algorithm

One of the major focuses of our investigation is to extend the task transfer mechanisms to the case where multiple source tasks from a task library are being used to bias the learning of one target task.

Sebastian Thrun has already shown how task clustering might work with the generation of a task library system in the domain of classification tasks [8] [9]. In reinforcement learning the situation is much more complex. This work extends Thrun's TC (Task Clustering) algorithm into the realm of reinforcement learning, thus the RLTC (Reinforcement Learning Task Clustering) algorithm.

In order to generate a reinforcement learning task library system the following must be accomplished.

1. Methods for transferring information from multiple source tasks must be developed.
2. An appropriate distance metric (or distance metrics) must be determined for the various tasks.
3. Individual tasks should be clustered together with other tasks that have things in common.
4. Methods for automatically determining which of the previously learned clusters a new task belongs in before that task has been completely learned must be developed.

We leave the final item for future work, while showing methods for performing the first three of these goals.

2.1 Multiple Task Transfer

The actions of the learning agent with multiple source tasks are based upon an amalgamation of advice from the past tasks as well as the advice that the agent derives from its actual experience in the world. This can be thought of as a type of social welfare function.

There are several voting schemes that could be used in order to construct this social welfare function. One solution is to take the average of the advice of all the source tasks, and then use those values to initialize the Q-values for the new task. The agent could then update those values with its own experience (in the single source domain this is known as "direct transfer"). This method has the advantage of allowing the target's own experience to eventually take a greater role as time progresses, while the information from the source tasks take a greater role at first. Unfortunately, averaging policies can be dangerous especially when the tasks being averaged are not very similar.

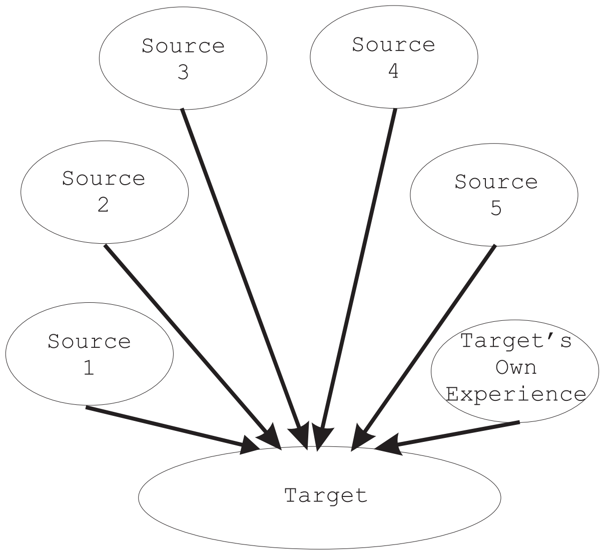

Figure 2: The actions of the target agent can be thought of as a social welfare function compiled from the advice of many source tasks and the target's own experience in the world.

Averaging tends to work better the more similar the source tasks are.

Another method of forming this social welfare function is to take only the invariants among the source tasks, and transfer those invariants as in direct transfer, while initializing the rest of the state space normally. Theoretically this will allow an agent to learn the task normally except in regions where all his previous sources agree. As before this technique will be more effective the more related the source tasks are to each other and to the target task. The more you can narrow down the number of source tasks the more invariants there will be to transfer.

2.2 Distance Metrics

In any task library there will be many tasks that are related to each other, and to the target task, but there will almost certainly be many tasks that are unrelated to the target. Since the transfer algorithms are all extremely sensitive to the amount of similarity that is present between the source and the target tasks, a method for determining the similarity of any two tasks is needed.

Finding an appropriate distance metric will be dependent on the types of tasks under consideration. Ideally we would have a set of task features that would form a vector v (and a corresponding space S) that would parsimoniously characterize the tasks to be clustered. Unfortunately given even human capacity to find and abstract task similarities, it is difficult or impossible to find such characterizations. Automating the development of this characterization is even more difficult. Fortunately there are clustering algorithms that will function so long as there exists at least one distance measure d between any two tasks in the library. Formally: $\forall_{ij} \exists d_{i,j}$. Where i and j are any two tasks in the library.

Two simple distance metrics that could be used are based upon 1) $R(s, a)$, the expected immediate reward for taking action a in state s and 2) the Q-values $Q(s, a)$ the expected discounted future reward for taking action a in state s.

By canonically ordering the R or Q values we can convert them into a task characterization vector. Although the space spanned by this vector is large, these two values sufficiently characterize the problem. In this space we can measure distance between tasks using the simple euclidian distance generating two separate distance measures D_Q and D_R. Which distance measure (or combination of distance measures) to use will depend upon the types of tasks in the library. Different cluster trees will be generated with each distance measure used, and each tree will hopefully capture a different set of features that tasks might have in common. Section 5 compares the results obtained using R-values as the task characterization with the results obtained using Q-values.

2.3 Clustering

One method for generating a useful sub-portion of source tasks from which to attempt transfer is to cluster the source tasks into sets of related tasks. Then the task of picking a set of related source tasks could be greatly simplified, perhaps even automated. Task Clustering is therefore the first step towards this automation. As the agent explores the space of a target task, and finds that this portion is similar to the same portion in a group of clustered tasks, then it will be reasonable to assume that the new target task might also share many similarities with this cluster of tasks in other parts of its state space.

The clustering algorithm creates a tree structure. Leaf nodes correspond to the tasks being clustered. Parent nodes link successively more dissimilar tasks, or clusters of tasks, as we move from the leaves to the root of the tree. The parents nodes in the tree represent the cluster containing all of the leaf nodes which are descendants of that parent.

The clustering algorithm takes as input a list of tasks and the distance between all possible pairs. That is, a list of tasks, $T_1 ... T_n$, and a corresponding array, D, $(n \times n)$ of distances $d_{i,j}$, where $1 \leq i, j \leq n$.

Note that we treat the tasks in the library as samples from a family (the ideal cluster) of tasks, A, represented by a region, S_A, in the space, S, which was described above. $C_A = \{C_i \mid i \in A\}$ represents the set of ALL tasks in the family, A, even ones we have never seen. We use $C_a = \{C_i \mid i \in a\}$ to represent the sample of tasks in the family A that are in the library.

For two clusters A and B with corresponding task samples C_a and C_b the distance between all possible pairs of sample tasks in the two cluster (the edges in a bipartite, complete graph, $K_{m,n}$, comprised of the m tasks in C_a and the n tasks in C_b) form a sample measurement, $d_{i,j}, i \in a, \ j \in b$ of the distance between the clusters A and B. We use the sample mean $(\mu_{a,b})$ to estimate the distance between clusters. Individual tasks can be treated as a cluster of size 1, so that the value $\mu_{a,b}$ can be used as an estimate for the distance between any two tasks, between a task and a cluster, or between a cluster and another cluster. As the number of tasks in C_a and C_b becomes large we can use a subset or sub-sample to compute the mean distance, and thus bound the computational complexity.

In each iteration of the algorithm we select $l^*, k^* = \arg\min_{l,k}(\mu_{l,k})$ (where l and k iterate over every cluster) to find the two clusters, C_{l*} and C_{k*}, which are closest to each other. These two clusters are used as the children of the next parent in what will grow into a binary tree. The two clusters C_{l*} and C_{k*} are then removed from the cluster list **C**, and are merged into a new cluster which is then added back into the list of clusters **C**. $\mu_{l,k}$ is then recomputed for the clusters in **C**. This process is repeated until all tasks have been connected into the binary tree the root of which is the only cluster left in **C**.

This binary tree could then be collapsed by merging parent and grandparent nodes which link clusters or individual tasks which are approximately the same distance apart. In future work we intend to explore the use of a t-tests on the distances to determine whether a particular cluster split is statistically significant, or whether it should be collapsed.

3 Methodology

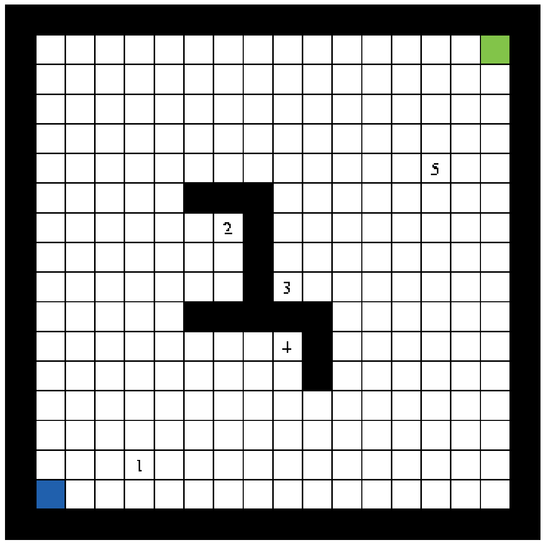

Figure 3: Simple grid-world.

We used a simple grid world [see Figure 3] in order to rapidly create a set of tasks that could be easily engineered to produce various levels of similarity. We produced a suite of thoroughly learned tasks in this world. Moving the goal various amounts, removing the obstacle in the center, or swapping the start and the goal produced various levels of similarity. There were four main types of tasks, tasks with the goal somewhere near the upper right, tasks with the goal somewhere near the bottom left, and tasks with and without the central obstacle. The hope was that the clustering algorithm would group these similar tasks together.

We then chose one of these tasks, and withheld it from the clustering algorithm to use as the target task later on. The other tasks were fed to the clustering algorithm and various clusters were produced which were then treated as secondary indexes into the list of learned tasks. Multiple cluster trees were produced using multiple distance metrics.

In order to form a baseline against which our other mechanisms could be judged we also tested transfer from individual tasks without any clustering algorithm. In order to use information from an individual source task to bias the learning of a new target task we used the well-understood "direct transfer" mechanism[4][3].

4 Results

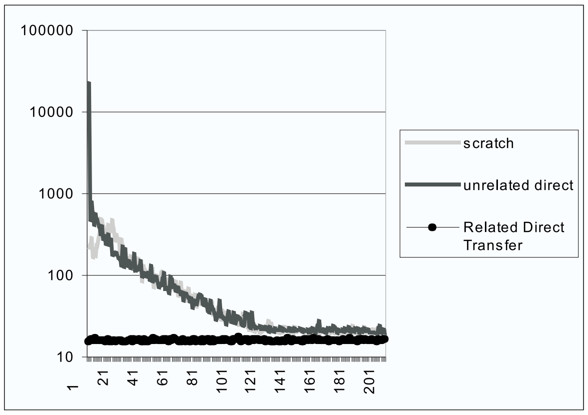

Figure 4: Shows the advantage of transfer if the two tasks are related, and the danger in transfer if the two tasks are unrelated. The left axis represents the average number of steps to the goal, while the right axis represents each epoch.

We first tested transfer from individual tasks. Figure 4 demonstrates the value of transfer, showing that a task can be learned much faster using transfer from a related task while demonstrating the potential damage that can be produced if transfer is indiscriminately used from an unrelated task. For our related task we moved the goal one step from its initial position, creating two tasks that were as similar as possible without being identical. Interestingly because the source task had been thoroughly explored, its application in transfer prevented the convergence to a sub optimal policy that is so common when learning from scratch with a "choose best" exploration policy. For our unrelated tasks we swapped the start and the goal, generating two tasks that were as dissimilar as possible. Notice that the scale of this graph is logarithmic. In unrelated transfer more time was spent in the first trial while unlearning the incorrect portion of the task than in all the rest of the trials combined. This result motivated the exploration of task clustering.

The clustering algorithm functioned as expected creating very logical clustering trees, one for each distance metric. The tree based upon the Q-value distance metric correctly separated out tasks that had the goal in the upper right from those tasks that had the goal in the bottom left. These categories of tasks were then further broken down into tasks that had the obstacle from those tasks that had no obstacle [see figure 5]. When the R-Values were used as the distance metric the tasks were much more sensitive the presence of the obstacle in the center, moving this difference further up the tree [see figure 6].

Once these clusters were produced we tested transfer using both invariants and average Q-values from from various clusters. We found that the invariants from the top of the cluster tree built from the Q-values didn't generate any useful information because there were no invariants at this level. There were invariants in the cluster tree based upon the R-Values, even at the top of the cluster tree where all the tasks were included. This was because the bounding wall was invariant among all tasks, and the R-Values easily captured this feature, while the Q-values did not. When invariant information was transferred from related branches of the cluster tree a substantial speedup in learning rate was detected. Transferring information from the R-Values substantially increased the agent's fault tolerance, reducing

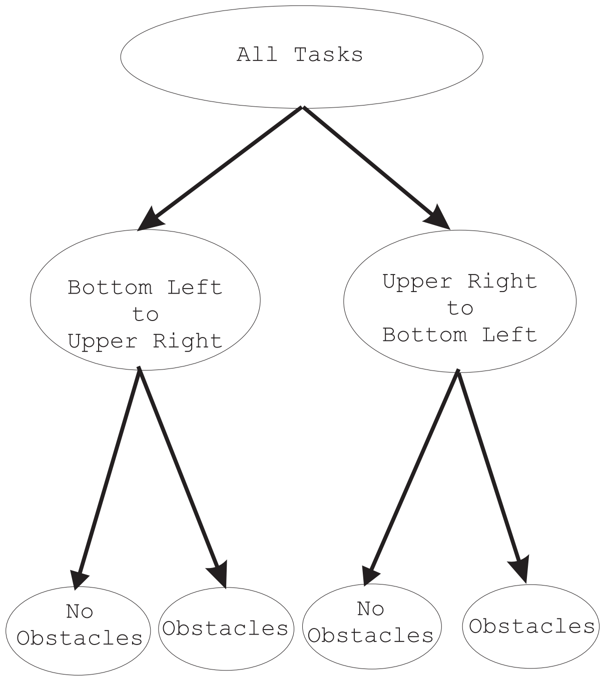

Figure 5: Cluster tree 1, based on the mean squared error of the Q-values.

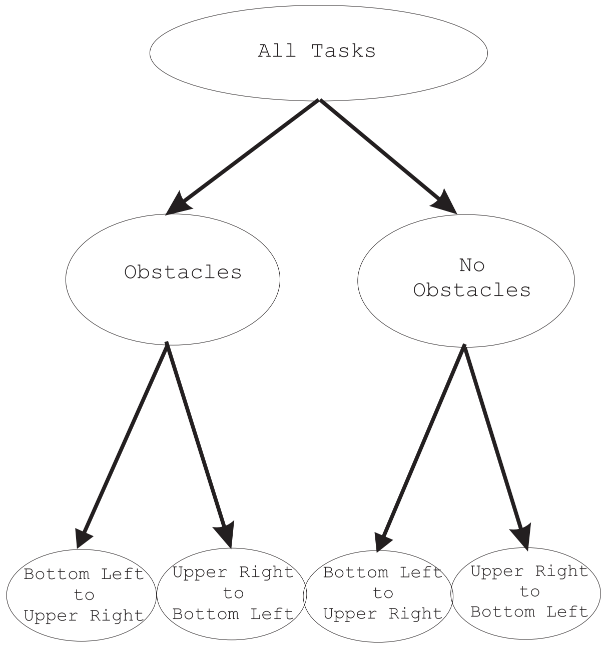
All Tasks
Obstacles
No Obstacles
Bottom Left to Upper Right
Upper Right to Bottom Left
Bottom Left to Upper Right
Upper Right to Bottom Left

Figure 6: Cluster tree 2, based on the mean squared error of the R-Values.

the number of negative rewards encountered during training. If the proper sub-branch of the R-Value cluster tree was used the number of negative rewards received by the agent was reduced to 0.

In addition to invariant transfer we also tested the use of the average Q-value from the the appropriate branch of the tree built from the Q-values for the initialization a target task. This gave a substantial increase in learning rate [shown in 7]. We also attempted transfer from from the top of the Q-value tree (effectively using all tasks as if they had not been clustered). Unsurprisingly, it was determined that the average Q-values of all tasks generated a nonsensical initial policy which was detrimental to learning.

5 Conclusions

We have shown that related transfer can be extremely helpful, while unrelated transfer can be detrimental. We have also shown that reinforcement learning tasks can be effectively clustered, and that the transfer across invariants in such a cluster can be beneficial if the cluster is sufficiently related and if the target belongs to the particular cluster chosen.

This research has potential in the production of a truly autonomous agent that can learn many tasks over its lifetime. Such an agent could store these tasks in a task library, and work on clustering these tasks offline. Then when a new task must be learned the agent could attempt to use information from these past tasks online in order to learn the new task faster.

6 Future Work

We eventually hope to automate the process of selecting an appropriate cluster to transfer from. Determining the relatedness between a target task and a cluster of source tasks should be reducible to a simple localization problem. This problem should be easy to solve in the case of the R-Values, while solving this problem with Q-values may be more difficult because

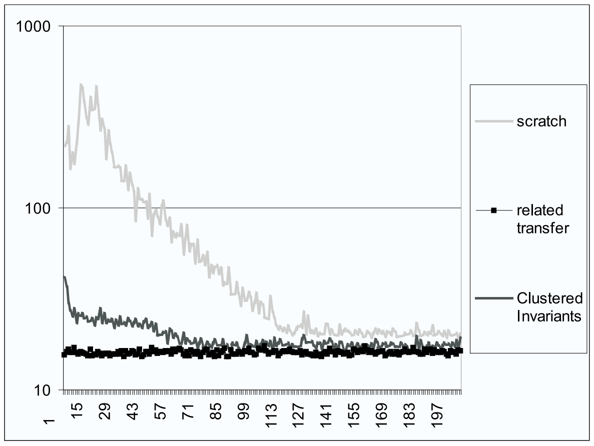

Figure 7: Transferring information from the clustered tasks rather than from all tasks avoided the disadvantages found in unrelated transfer, while allowing most of the speedup from related transfer.

the Q-values will not approach their correct values until the entire task has been learned. This problem should be our next area of research, because by combining that tool with the research described in this paper the entire task library process could be automated. This would have extremely significant ramifications for such techniques as shaping.

Another area of research that should be explored is piecewise transfer, where one source task may be used in a portion of the target task, while another source task may be used in another portion of the target task [10]. Related to this idea is the notion that tasks may have one thing in common with one cluster of tasks, while having something else in common with another cluster, perhaps even another cluster produced using a different distance metric. It may be possible to combine this information from different source clusters in order to rapidly produce a new task.

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

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

[5] 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.

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

[7] 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.

[8] S. Thrun and J. O'Sullivan, "Clustering learning tasks and the selective cross-task transfer of knowledge," 1995.

[9] Sebastian Thrun and Joseph O'Sullivan, "Discovering structure in multiple learning tasks: The TC algorithm," in *International Conference on Machine Learning*, 1996, pp. 489–497.

[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.