Learning a Transfer Function for Reinforcement Learning Problems

Tom Croonenborghs[1], **Kurt Driessens**[2] **and Maurice Bruynooghe**[2]
1: Biosciences and Technology Department, KH Kempen University College, Geel, Belgium
2: Declarative Languages and Artificial Intelligence Group, Katholieke Universiteit Leuven (KUL), Belgium
tom.croonenborghs@khk.be
{kurt.driessens,maurice.bruynooghe}@cs.kuleuven.be

Abstract

The goal of transfer learning algorithms is to utilize knowledge gained in a source task to speed up learning in a different but related target task. Recently, several transfer methods for reinforcement learning have been proposed. A lot of them require a mapping that relates features from the source task to those of the target task, and most of the time it is the task of a domain expert to hand code these mappings.

This paper proposes a method to learn such a mapping automatically from interactions with the environment, using a probability tree to represent the probability that the optimal action learned in the source task is useful in the target task. Preliminary experiments show that our approach can learn a meaningful mapping that can be used to speed up learning through the execution of transferred actions during exploration.

Introduction

Inductive transfer or transfer learning is concerned with the connection between learning in different but related contexts. In real life, the beneficial effects of transfer learning are obvious. Examples include learning to drive a bus after having learned to drive a car or learning Dutch after having learned German (except of course for native Dutch speaking people). In a machine learning context, transfer learning is concerned with the added benefits that learning one task can have on a different, but probably related task.

An area where transfer learning is particularly important is the domain of Reinforcement Learning. In reinforcement learning (Sutton & Barto 1998), an agent can observe its world and perform actions in it. The agent's learning task is to maximize the reward he obtains. Reinforcement learning is often approached as a tabula rasa learning technique, where at the start of the learning task, the agent has no or little information and is forced to perform random exploration. As a consequence, reinforcement learning in complex domains quickly becomes infeasible or too slow in practice. Leveraging knowledge could significantly increase the learning speed and thereby expand the applicability of reinforcement learning approaches.

Recently, several approaches have been proposed to transfer knowledge between different reinforcement learning tasks. Often, a user defined mapping is used to relate the new task to the task for which a policy was already learned (Taylor, Whiteson, & Stone 2007; Torrey *et al.* 2006). There has been some work on learning such a mapping. For example, in the work by Liu and Stone (Liu & Stone 2006) a graph-matching algorithm is used to find similarities between state variables in the source and target task. This approach however needs a complete and correct transition model for both tasks. A similar approach is used by (Kuhlmann & Stone 2007) in the context of General Game Playing.

In this paper, transfer learning is achieved by considering *transfer actions*, i.e. actions transferred from the source task to the target task during the exploration phase of the learning. To decide which action to transfer, the agent learns a function that predicts for each source action the probability that executing the transfered action is at least as good as executing the action which is best according to the agent's current utility function and selects the one with the highest probability.

The rest of the paper is structured as follows: the next section introduces the notion of transfer during exploration and the transfer function. It also describes how the transfer function can be learned from simple interaction with the environment without the need for an external oracle to define a feature mapping. Afterwards we present some preliminary experiments that serve as a proof of concept. We review some related work and directions for future work at the end of the paper.

Using Exploration to Transfer Knowledge

We consider a standard reinforcement learning problem (Sutton & Barto 1998) where an agent interacts with its environment. The environment is defined by a set of states $s \in \mathcal{S}$, a number of executable actions $a \in \mathcal{A}$, an unknown state-transition function $\delta(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ that represent the probability that taking action a in state s will result in a transition to state s' and a reward function $r(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. When the reward function is nondeterministic, we will use the expected rewards $R(s, a) = E_{s,a}[r(s, a)]$. The state, action, and reward at time t are denoted $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$ and r_t (or R_t) $\in \mathbb{R}$.

The agent selects which action to execute in a state following a policy function $\pi(s) : \mathcal{S} \rightarrow \mathcal{A}$. During exploration of the environment the agent uses an exploration strategy indicated by π_e that guarantees that each state-action combination has a non-zero probability of being visited. The utility of a state-action pair $Q^\pi(s,a)$ for a policy π is defined as the expected discounted sum of future reward, starting from state s by selecting action a and following the actions indicated by policy π afterwards:

$$Q^\pi(s,a) = E_{\pi,\delta,r}\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i} | s_t = s, a_t = a\right] \tag{1}$$

where $\gamma \in [0,1[$ is the discount factor.

Transfer Actions

To incorporate knowledge transfer into the exploration of the reinforcement learning agent, the standard exploration policy π_e of the agent is altered. In our initial efforts, we decided to emulate ϵ-based exploration to accomplish this integration. With a given probability of ϵ_t the agent will select and execute a *transfer action* $\pi_t(s)$ instead of the normal exploration action $\pi_e(s)$. In lieu of the learning speedup obtained with guidance from experts in complex reinforcement learning tasks (Driessens & Dzeroski 2004), we expect these transfer actions to help with learning, under the condition that the transfer actions are chosen appropriately.

Assume the agent has learned a policy π_s that performs well on the source task. To decide which transfer action to perform in a state t of the target problem, we will employ a *transfer function* $p(s,t)$ that represents for each source state s the probability that the best action for s is at least as good for t as the best action according to the current approximation of the utility function. This function allows the agent to select a transfer action according to the policy learned on the source task, for that state s from the source task that maps best onto the target state t. Thus, the transfer action performed in state t (i.e., $\pi_t(t)$) is the action performed by policy π_s in the state s for which $p(s,t)$ is maximal, i.e.,

$$\pi_t(t) = \pi_s(\operatorname*{argmax}_s p(s,t)) \tag{2}$$

with π_s the source task policy.

Note that we assume for simplicity that actions in the source task are executable in the target task. In the current setup, if this is not the case, we could generate a negative reward when the agent tries to execute a non-available action, and it would quickly learn that the transfer function for the state that suggested this action should be low. In future work, we will extend our approach so that the transfer function maps (state,action)-pairs and is thereby able to incorporate the relation between actions.

Learning the Transfer Function

The aim of this work is to avoid the need for a human expert that defines a mapping function between the source and the target task, but to be able to learn this mapping automatically. In the setup described above, this means we have to learn the transfer function $p(s,t) : \mathcal{S}_s \times \mathcal{S}_t \rightarrow [0,1]$, where $\mathcal{S}_S$ is the set of states of the source task and $\mathcal{S}_t$ is the set of

Algorithm 1 Transfer Learning by Exploration

```
1: initialize the Q-function hypothesis $\hat{Q}_0$
2: initialize the transfer function $\hat{P}_0$
3: $e \leftarrow 0$
4: repeat {for each episode}
5:   $Ex_Q \leftarrow \emptyset$
6:   $Ex_P \leftarrow \emptyset$
7:   generate a starting state $s_0$
8:   $i \leftarrow 0$
9:   repeat {for each step of episode}
10:    if $rand() < \epsilon_t$ then {select transfer action}
11:      $a_i = \pi_s(\operatorname*{argmax}_t p(t,s))$
12:      $t_i = \operatorname*{argmax}_t p(t,s)$ {remember transfer state}
13:    else {select exploration action}
14:      $a_i = a$ with prob. $\frac{e^{\frac{\hat{Q}_e(s,a)}{\tau}}}{\sum_{b \neq a} e^{\frac{Q_e(s,b)}{\tau}}}$
15:    end if
16:    take action $a_i$, observe $r_i$ and $s_{i+1}$
17:    $i \leftarrow i + 1$
18:  until $s_i$ is terminal
19:  $q_{i-1} \leftarrow r_{i-1}$
20:  for $j = i - 2$ to 0 do
21:    $q_j \leftarrow r_j + \gamma q_{j+1}$
22:    generate example $x_q = (s_j, a_j, q_j)$
23:    $Ex_Q \leftarrow Ex_Q \cup \{x_q\}$
24:    if $a$ was selected as transfer action, i.e.
       $a_j = \pi_s(\text{argmax}_t\ p(t, s_j))$ then
25:      if $q_j \geq max_a \hat{Q}_e(s_j, a)$ then
26:        generate example $x_p = (t_j, s_j,$ 'transfer'$)$
27:      else
28:        generate example $x_p = (t_j, s_j,$ 'no transfer'$)$
29:      end if
30:      $Ex_P \leftarrow Ex_P \cup \{x_p\}$
31:    end if
32:  end for
33:  Update $\hat{Q}_e$ using $Ex_Q$ to produce $\hat{Q}_{e+1}$
34:  Update $\hat{P}_e$ using $Ex_P$ to produce $\hat{P}_{e+1}$
35:  $e \leftarrow e + 1$
36: until no more episodes
```

states from the target task, from interaction with the environment.

At the end of every learning episode in the target task, a number of learning examples for the transfer function can be generated. For every transfer action a_t the agent executed (in a state s) during the episode, he can compare the quality of the transfer action with the quality of his current policy using two different utility values. On the one hand, a Q-value $Q_{MC}(s, a_t)$ can be obtained by backtracking the Q-values from the end of that episode to the step where a_t is executed. This is comparable to a Monte Carlo estimate of this Q-value. On the other hand we let the agent learn a Q-value approximation $\hat{Q}$ of its current policy using a standard Q-learning type algorithm with generalization.

The generated learning example for each executed transfer action then consists of the states in both the source and target task and a label for the example:

"transfer" if $Q_{MC}(s, a_t) \geq max_a \hat{Q}(s, a)$ and
"no transfer" otherwise

Using a learning algorithm able to predict class probabilities, the learning agent can generate a generalized probability function that approximates the intended transfer function p by predicting the probability that a new state-state pair will belong to the class "transfer".

Note that, in practice, we are not interested in the exact probability value as defined by the function p. The purpose of p is to generate a positive effect of the transferred actions on the learning speed.

An overview of the reinforcement learning algorithm with transfer by exploration can be found in Algorithm 1. Note that we used a SARSA temporal difference learner (Rummery & Niranjan 1994) (line 21), but another reinforcement learning technique could be used instead.

Preliminary Experiments

We implemented a prototype of the suggested approach to act as a proof of principle. We decided to use a first order tree learning algorithm TILDE (Blockeel & De Raedt 1998) to learn the transfer function p. The TILDE system is able to predict class probabilities as required in our setup (Fierens *et al.* 2005). To approximate the utility values, we use the SARSA temporal difference approach (Sutton & Barto 1998) and an incremental regression tree algorithm TG (Driessens, Ramon, & Blockeel 2001) to generalise the utility function. These choices are not specific to the approach, only to our current implementation.

Experimental Setup

To evaluate our implementation, we ran experiments in a simple grid based world, depicted in Figure 1. The rooms are connected with colored doors which the agent (indicated with the diamond labelled "a") can only pass if he possesses a key of the same color as the door (depicted with colored circles). The primitive actions available to the agent include four movement actions (up, down, left and right) and a pickup action that picks up the key located at the agent's location (if applicable). The agent can execute at most 500 actions per episode and receives a reward of 1 if he exits the last room and 0 otherwise. The state representation includes the dimensions of the different rooms, the locations and colors of the doors, the location and colors of the keys, the keys the agent possesses, the agent's location and the goal location. The location consists of two coordinates where a coordinate is determined by the relative position in the room and a certain offset for every room as indicated in Figure 1.

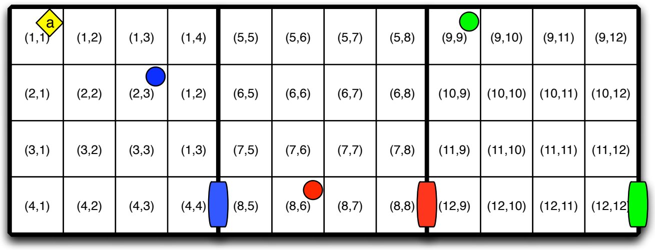

Figure 1: Grid based world.

In the source task, the agent had (successfully) learned how to navigate to the bottom right location in a four by four grid.

Results

As a first experiment, instead of continuously learning the transfer function, we learned a single transfer function once with TILDE (Blockeel & De Raedt 1998) based on learning examples created during the first 100 learning episodes in the target task and actions transferred from random source states. The algorithm was able to learn both the shift in coordinates between the different rooms and that successful transfer is very unlikely if the agent does not have the key needed to leave the current room. We then restarted the agent in the environment with the learned transfer function.

Figure 2 shows the average reward and Figure 3 the number of actions per episode of a SARSA-learning agent, both with and without transfer. The numbers are obtained by freezing the current utility function and following a greedy test policy for 100 episodes every 50 episodes. We show results averaged over 10 runs.

Discussion

As shown, the approach with transfer reaches the goal state in a higher percentage of trials and uses less actions per episode to reach it. The graph does not show the 100 learning episodes used to generate the learning examples to learn the transfer function which is an added cost in our current implementation and should be taken into account when interpreting the learning speed. In future work, we would like to employ an online, incremental learning algorithm to learn the transfer function and to eliminate this extra cost.

Related Work

A lot of transfer learning approaches use a mapping to relate the new task to the task for which a policy was already learned. This mapping relates features from the old task to

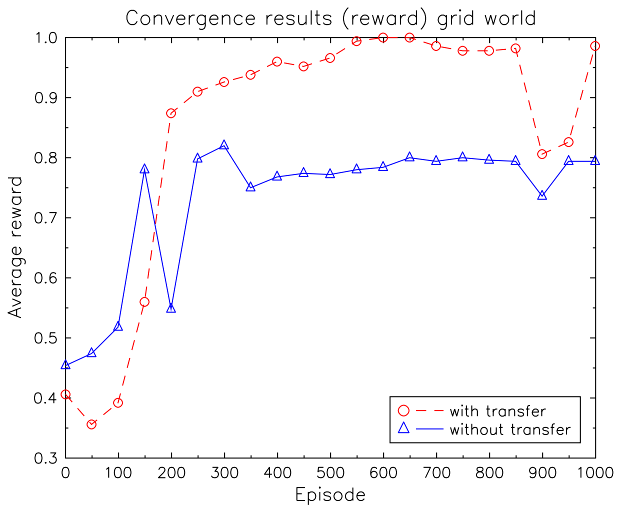

Figure 2: Average reward with and without transfer function.

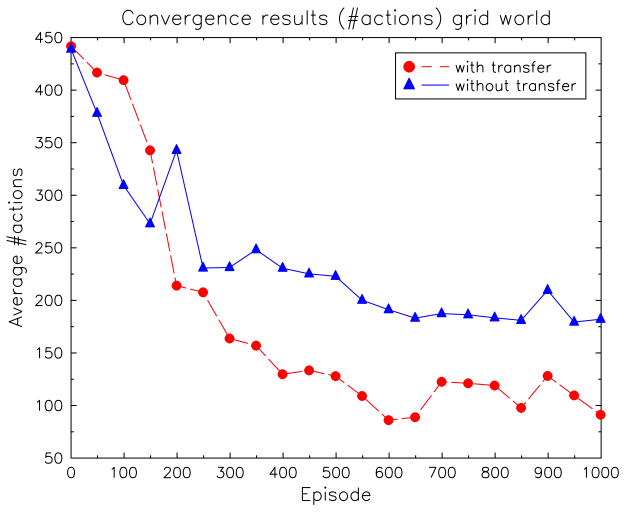

Figure 3: Average number of actions with and without transfer function.

features from the new task. How this mapping is then used to transfer knowledge differs between approaches. Quite often, the previously learned policy is used to guide exploration in the new task. In the approach of Madden and Howley (Madden & Howley 2004), the learner will, after gaining enough experience in a simple task using a standard tabular Q-learning algorithm, use a symbolic learner and a propositional representation of the task to build a generalized policy based on the learned Q-values. This policy is then used to aid exploration in a more difficult version of the task. This is related to our approach, but does not use an explicit representation of the transferability of the given policy for a state.

Related to this, Fernandèz and Veloso (Fernández & Veloso 2006) re-use the learned policy as one option of a probabilistic exploration strategy which has a choice over exploiting its current learned policy, choosing a random exploration action or exploit the previously learned policy. The use of a similarity measure between policies allows for the discovery of classes of similar policies and, as a consequence, a basis (or library) of core policies, which can be used to study structural aspects of the learning domain. Currently, the application of this approach is limited to simple domains such as navigational problems. The authors state that applying their approach to more complex domains will require a mapping such as the one we learn in this work.

Torrey et al. (Torrey *et al.* 2005) use transfer learning to generate advice to speed up reinforcement learning. They use a SVM regression approach to approximate the Q-function of a task. The advice consists of relative information about Q-values of different actions as learned in the original task. The Q-value estimates of the original task are again translated to the new domain using a human designed mapping of state-features and actions. The advice is incorporated into the new task by adding the information about Q-values as soft-constraints to the linear optimization problem that approximates the Q-function for the next task. In follow up work (Torrey *et al.* 2006), the advice is generated from the original Q-function automatically through the use of a relational rule learner and extended with user-defined advice.

Taylor et al. (Taylor, Whiteson, & Stone 2006) use neural networks to represent policies in an evolutionary algorithm. After learning on one task, the resulting population of neural networks is restructured to fit the new task using a human designed mapping of features and actions between the two domains. These newly constructed neural networks are then used as the starting population for a genetic algorithm to optimize the policies towards the new task.

Liu and Stone (Liu & Stone 2006) address the problem of finding mappings between the source and target task. They apply a specialized version of the structure mapping theory, a psychological and computational theory about analogy making, to find similarities in the tasks based on similar patterns in their state transitions. Their algorithm needs as input almost a full specification of the source and target task while our approach assumes no prior knowledge.

Kuhlmann and Stone (Kuhlmann & Stone 2007) also made progress toward the complete automation of a domain mapping for value function transfer learning. They use a graph-based approach to identify similar games based on a canonical form of games in the General Game Playing (GGP) domain. As they apply their approach in GGP, they have full descriptions of the tasks available.

Talvitie and Singh (Talvitie & Singh 2007) use an experts algorithm to select the best policy amongst a number of given policies. By creating different target task policies using the source task policy and different mappings, they can hence find the best mapping. They however need to create different mappings by hand and are not able to alter the resulting policy.

The most related to our work is the work of Taylor et al. (Taylor, Whiteson, & Stone 2007). Their approach requires the state variables to be arranged into task-independent clusters which describe different objects in the domain. For every object in the domain, they learn a classifier that predicts the index of a particular state variable or action given a transition (state, action, next_state, reward) in the source task (minus the target of the classifier). They then apply the learned classifier to transitions in the target task and consider the prediction as a similar source action or state variable for that transition. A winner-takes-all voting scheme is then used to determine the actual mapping between actions and state variables. One difference with our approach is that we do not need the semantic knowledge of state-variable groupings. Furthermore, we believe that a lot of useful mappings cannot be identified by similarities in state transitions only, as is e.g. the case in our grid based world. Another difference is that our approach directly learns a mapping between states which might discover more complicated patterns compared to learning a mapping for every state variable independently.

Conclusions

We proposed a first step towards automatically learning a mapping from states of the source task to states of the target task for inductive transfer in reinforcement learning problems without the need for prior knowledge about the tasks. We suggested a transfer learning approach that performs transfer through guided exploration and defined a transfer function that represents the probability that a source state can be mapped to a target state. We implemented a proof of principle for the approach using SARSA as a reinforcement learning technique and probability trees as the transfer function. Preliminary experimental results show that the approach can yield a significant speedup for the learning process in the target task.

Future Work

Besides evaluating our approach in more detail, we would like to investigate a number of possible extensions of the proposed approach. One possible direction for further research is incorporating the quality of possible transfer in the agent's exploration policy to avoid negative transfer. At the moment, we employ an ϵ-greedy type approach, which is certainly suboptimal and does not make use of the learned probabilities of the transfer function. For example in our experimental setup, if the agent is not carrying the correct key to open the next door, $p(s, t)$ will be low for all $s \in \mathcal{S}_s$ and transferring actions will not be beneficial.

We would also like to substitute the batch learning of the transfer function as employed in our experiments, by a continuous, incremental approach.

Another important direction for further research is the incorporation of the action description into the transfer function. This will open the door towards relational reinforcement learning problems where the relations between states and actions define the transferability of a learned policy. We will certainly look into probabilistic relational learning techniques to model the probabilities of the transfer function.

Acknowledgements

Kurt Driessens is a post-doctoral research fellow of the Research Foundation - Flanders (FWO).

References

Blockeel, H., and De Raedt, L. 1998. Top-down induction of first order logical decision trees. *Artificial Intelligence* 101(1-2):285–297.

Driessens, K., and Dzeroski, S. 2004. Integrating guidance into relational reinforcement learning. *Machine Learning* 57(3):271–304. URL = http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=40988.

Driessens, K.; Ramon, J.; and Blockeel, H. 2001. Speeding up relational reinforcement learning through the use of an incremental first order decision tree learner. In De Raedt, L., and Flach, P., eds., *Proceedings of the 12th European Conference on Machine Learning*, volume 2167 of *Lecture Notes in Artificial Intelligence*, 97–108. Springer-Verlag.

Fernández, F., and Veloso, M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 720–727. New York, NY, USA: ACM Press.

Fierens, D.; Ramon, J.; Blockeel, H.; and Bruynooghe, M. 2005. A comparison of approaches for learning probability trees. In *Proceedings of 16th European Conference on Machine Learning*, volume 3720 of *Lecture Notes in Artificial Intelligence*, 556–563.

Kuhlmann, G., and Stone, P. 2007. Graph-based domain mapping for transfer learning in general games. In Kok, J. N.; Koronacki, J.; de Mántaras, R. L.; Matwin, S.; Mladenic, D.; and Skowron, A., eds., *Proceedings of the 18th European Conference on Machine Learning*, volume 4701 of *Lecture Notes in Computer Science*, 188–200. Springer.

Liu, Y., and Stone, P. 2006. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, 415–20.

Madden, M. G., and Howley, T. 2004. Transfer of Experience Between Reinforcement Learning Environments with Progressive Difficulty. *Artif. Intell. Rev.* 21(3-4):375–398.

Rummery, G., and Niranjan, M. 1994. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press.

Talvitie, E., and Singh, S. 2007. An experts algorithm for transfer learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Taylor, M. E.; Whiteson, S.; and Stone, P. 2006. Transfer learning for policy search methods. In *ICML workshop on Structural Knowledge Transfer for Machine Learning*, 1–4.

Taylor, M. E.; Whiteson, S.; and Stone, P. 2007. Transfer via inter-task mappings in policy search reinforcement learning. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*.

Torrey, L.; Walker, T.; Shavlik, J.; and Maclin, R. 2005. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the 16th European Conference on Machine Learning*, 412–424.

Torrey, L.; Shavlik, J.; Walker, T.; and Maclin, R. 2006. Skill acquisition via transfer learning and advice taking. In *Proceedings of the 17th European Conference on Machine Learning*, 425–436.