Efficient Reinforcement Learning using Relational Aggregation

Martijn van Otterlo OTTERLO@CS.UTWENTE.NL
TKI, Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE, Enschede, The Netherlands

1. Introduction

Much research in Reinforcement Learning (RL) has focused on learning *algorithms* and *generalization* using simple representation languages for states and actions. Recently, there is much interest in various kinds of *abstraction*. Abstractions over time or primitive actions, e.g. in *hierarchical RL*, are useful ways to abstract over specific sub-actions or time. Currently there is also interest in using more powerful representation *languages* for abstraction in RL, in which subsets of first-order logic are used for representing sets of states and actions. For an overview of these methods, see (van Otterlo., 2003; van Otterlo, 2002).

Some issues in relational representations were recently addressed in both *decision theoretic planning* and (model-free) RL. The work by Dzeroski et al. (Dzeroski, 2002) on *Relational Reinforcement Learning* (RRL) uses an online tree induction algorithm that was upgraded to relational representations. RRL induces a relational representation of the value function, thereby generalizing over the state-action space by using predicates and variables. The work by Morales (Morales, 2003) uses a relational representation for states and actions too, but does not induce a representation of the state space. Instead, it uses a *predefined* partitioning of the state space and predefined (global) actions that can be applied.

Here we present a new method for RL using a relational representation that is also predefined, but we use *local* actions (i.e. defined relatively to some (abstract) state). Our representation involves identifiable states and actions as exact *abstractions* of parts of the underlying MDP. With this we can extend our approach by learning a model and by using model-based RL algorithms such as *prioritized sweeping* (PS).

2. Approach

A *relationally factored* MDP (RMDP) $M =< P, A, D, T, R >$, where P is a set of predicate templates (*fluents*), A is a set of action templates and D is a domain of objects. The state space $\mathcal{S}$ is defined as a subset of the set of interpretations over P and D. Let $\mathcal{A}$ be the set of interpretations over A and D (i.e. the set of ground actions), then $T: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$ and $R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. Solving the RMDP involves finding a policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$ that maximizes expected rewards. In this paper we do this by learning a value function.

In order to solve an RMDP, a representation is defined as follows. Formally, an abstracted state-action space Υ is a set of *regions*, defined by $\Upsilon = \{(\sigma, \Omega)\}$ in which σ is a relational expression describing an abstract state and Ω a set of abstract actions that can be applied in that state. If $\rho \in \Upsilon$ is a region, we use ρ^σ and ρ^Ω as notation for the two parts of a region ρ. For example (A, B and C are variables), $\rho^\sigma = \exists A, B, C\ [\ on(A, B) \wedge on(B, C) \wedge on(C, f) \wedge \neg(A = B) \wedge \neg(B = C)]$ and $\rho^\Omega = \{move(A, f)\}$ for some blocks world example. Value functions are defined for Υ: if $\rho \in \Upsilon$, then $Q(\rho^\sigma, \omega)$ is the Q-value for some action $\omega \in \rho^\Omega$ and $V(\rho^\sigma) = max_{\omega \in \rho^\Omega} Q(\rho^\sigma, \omega)$.

Algorithm 1 Main Reinforcement Learning Loop

```
Require: environment is initialized etc.
 1: for all episodes do
 2:   Initialize start state
 3:   while NOT ((end of episode) OR
      (maximum number of steps reached)) do
 4:     $s$ is current ground state
 5:     Find covering region: $\rho \in \Upsilon$ for which $s \vdash \rho^\sigma$
 6:     Take action set $\rho^\Omega$
 7:     if exploration then
 8:       exploration strategy chooses $\omega$ from $\rho^\Omega$
 9:     else
10:       $\omega = \arg\max_{\omega' \in \rho^\Omega} Q(\rho^\sigma, \omega')$
11:     end if
12:     get set of substitutions $\theta = \{\theta_i \mid s \vdash_{\theta_i} \rho^\sigma\}$
13:     take random substitution $\theta_i$ from $\theta$
14:     apply ground action $a = \omega\theta_i$
15:     observe new state $s'$ and reward $r$
16:     Find new region: $\rho' \in \Upsilon$ for which $s' \vdash \rho'^\sigma$
17:     $Q(\rho^\sigma, \omega) = Q(\rho^\sigma, \omega) +$
        $\alpha(r + \gamma \max_{\omega' \in \rho'^\Omega} Q(\rho'^\sigma, \omega') - Q(\rho^\sigma, \omega))$.
18:   end while
19: end for
```

Interaction with the environment happens in terms of *ground representations* (e.g. no variables are used). In the RL cycle of perceiving states, doing actions and getting rewards, all states and actions are ground. The learner perceives a state that consists of all simple features that are true in the current state, i.e. a state is a *first-order interpretation*. For a blocks world, just the *on(·,·)* and *clear(·)* relations are used for states. An example ground state is $\{on(a,b), on(b,c), on(b,f), clear(a), clear(f)\}$. The abstract states are specified using the simple features and variables, but can make use of *background knowledge* relations as well. The relation *height(X,N)* - saying the X is the top of stack of height N - is not provided in the ground state, but it can be defined in terms of simple state features like *on* and *clear*. Background knowledge, if available, enables powerful abstractions.

Algorithm 1 shows the main algorithm for standard Q-learning over an abstract state space. Notice that the main differences with standard Q-learning are the random action choice in line 13 and the covering test by *proving* ($\vdash$). Also different is that regions have their own set of actions. Each action is dependent on the representation of the region it is in.

By learning a model of the underlying RMDP, i.e. approximate transition probabilities and expected rewards for Υ, we can speed up learning the value function. For this we modified line 17 of algorithm 1 into a call to a PS algorithm[1] with ρ as argument.

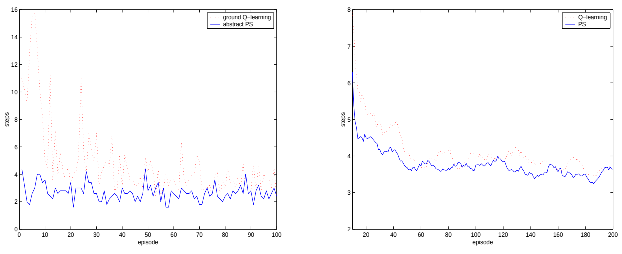

Figure 1. (a) Q-learning on a ground state space vs. PS on an abstract state space for a four blocks world (averaged over 5 runs). Notice the large variance in ground learning. Learning over the abstract space converges after only a few episodes. (b) Q-learning vs. PS on the same abstract five blocks world (mean over the last 10 episodes, averaged over 10 runs).

We have performed some experiments in simple blocks world problems and the game of Tic-Tac-Toe (see figures 1 and 2). To give some impression, in the 4-blocks world a ground representation consists of 240 state-action-pairs (SAP), but in our representation just 12. For Tic-Tac-Toe a ground representation has roughly 6000 states and many more SAP's, whereas we have only 15 states and 41 SAP's for reasonable play.

[1]We use Wiering's modified PS algorithm, adapted for our specific (action) representation.

3. Conclusions and Further Research

The experiments show that i) the use of our relational representation significantly reduces the number of episodes needed to learn a task (because of the reduction of the size of the state space) and ii) because our representation enables the learning of a transition model for the underlying MDP, it also enables us to use PS to speed up learning of the value function.

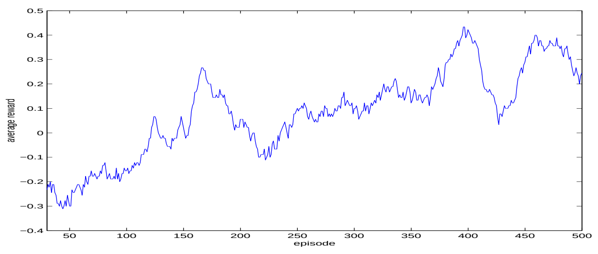

Figure 2. Sample graph of learning Tic-Tac-Toe on an abstract space (mean over 3 runs). The data points are moving averages over 30 games.

Working with relational representations is computationally demanding. For example, proving coverage of a state is harder in a relational language than for propositional languages. Also, the combination of value-based methods (RL and other stochastic approximation techniques) and logical induction methods such as *inductive logic programming* has not been studied much. Nevertheless, the benefits of relational languages, i.e. compact state space, comprehensibility, applicability in large (relational) state spaces etc. (see also (van Otterlo, 2002)) will outweigh this computational burden in the end.

Our current and future work deals specifically with a second disadvantage, that is the *predefined* state space. We are working on methods to adaptively learn a representation *while learning* (e.g. like in RRL) and we have some initial work ongoing in bottom-up generalization, using information from the value function and transition probabilities. Unlike RRL, which learns an abstraction of the value function, we want to learn a representation of the state space and learn a value function for this.

References

Dzeroski, S. (2002). Relational reinforcement learning for agents in worlds with objects. *In Proceedings of the AISB'02 Symposium on Adaptive Agents and Multi-Agent Systems* (pp. 1–8).

Morales, E. (2003). Scaling up reinforcement learning with a relational representation. Published at a local workshop in Sidney (in january).

van Otterlo, M. (2002). Relational expressions in reinforcement learning: Review and open problems. *Proceedings of the ICML'02 Workshop on Development of Representations.*

van Otterlo., M. (2003). *Efficient reinforcement learning using relational aggregation* (Technical Report 2003-XX). Department of Computer Science, University of Twente, The Netherlands. forthcoming.