Efficient Reinforcement Learning using Relational Aggregation

Martijn van Otterlo
Language, Knowledge and Interaction Group
Department of Computer Science
University of Twente
The Netherlands

Sixth European Workshop on Reinforcement Learning (EWRL-6)

Outline

• Generalization and Abstraction in RL
• Relational MDP’s
• Existing techniques for solving RMDP’s
• Efficient RL on fixed Representations
• Possibilities for learning the representation
• Further Research
Abstraction and Generalization

- Generalization (NN, MLP, DT) prop, AV representations
- Hierarchical (MaxQ, options, HQ) Abstraction over time and/or subactions
- Logical Representation Languages Compact, powerful Relational domains

Gains: compactness, efficiency Challenge relating values of underlying structure to aggregations Possibly create partial observability

Bazen, van Otterlo et al (2001)

Relationally Factored MDP’s

An RMDP $M = <P, D, A, T, R>$

- $P$: set of predicate templates, e.g. on(.,.), clear(.,.), height(.,.)
- $A$: set of action templates, e.g. move(.,.)
- $D$: domain of objects, e.g. (a, b, c, table)

State space $S$ is a subset of all first-order interpretations over $P$ and $D$.
The action-space $A'$ consists of all interpretations over $A$ and $D$.

Examples: (on(a,b),clear(a),on(b,table)), move(a,table)

$T$ and $R$ are defined as $T: S \times A' \rightarrow [0,1]$ and $R: S \times A' \rightarrow \mathbb{R}$

Goal: find an optimal policy $\pi: S \rightarrow A$

RMDP’s are very large $|S| = \prod_{i=1}^{k_i} \alpha_i^{|D|}$ ($n = \text{arity}, |D| = k$) (BW: 10 blocks $\rightarrow 60M$)

logical language as a compact way of specifying $T, R, \pi, V$ and $Q$. 

Bazen, van Otterlo et al (2001)
DP-based approaches to RMDP’s

- Variety of logical formalisms
- DP-algorithms can be applied
  - Deduction of (value) regions from known \( T \) and \( R \)
  - Regression (weakest precondition)
- Induce policy from \((s,a,q)\)-samples
- Heuristic search for policies

<table>
<thead>
<tr>
<th>Authors</th>
<th>Approach</th>
<th>Generalization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boutilier et al. (2001)</td>
<td>Sym DP, VI</td>
<td>SC</td>
<td>Regression, Simplification</td>
</tr>
<tr>
<td>Yoon et al. (2002)</td>
<td>≤ Upgrade</td>
<td>DL</td>
<td>Heuristic Search, Rule Learning</td>
</tr>
<tr>
<td>Grossmann et al. (2003)</td>
<td>Sym DP, VI</td>
<td>FC</td>
<td>Regression, Simplification</td>
</tr>
<tr>
<td>Fern et al. (2003)</td>
<td>API</td>
<td>DL</td>
<td>Rollout, Upgrading</td>
</tr>
<tr>
<td>Mausam et al. (2003)</td>
<td>≤ Upgrade</td>
<td>HL</td>
<td>SPUDD, Logical regression</td>
</tr>
<tr>
<td>Guestrin et al. (2003)</td>
<td>Upgrade</td>
<td>PRM</td>
<td>Static Fluents</td>
</tr>
</tbody>
</table>

RL approaches to RMDP’s

- [Lecoutre] (’01), RRL, RRL-TG (’98, ’01) (Dzeroski,Dressens)
  - Induction of abstract Q-function from samples
  - Problems: online building, sampling, relational regression
- RRL-RIB (’03) (Dzeroski,Dressens)
  - Nearest neighbor on relational representations
  - Distance measure highly domain-specific
- Kernel RRL (’03) (Gartner et al.)
  - Product graph kernel for structured data
- RQ (’03) (Mamer)
  - Q-learning, fixed state, abstract states
  - Definitions of action preconditions are known

Difficulties
- Induction of representations? (T, R unknown)
- How to generalize?
- ...
Compact Representation for RMDPs
Existing work mostly Q-function abstraction
Compact representation of a state-action space (MDP) enables
- model-estimation + efficient learning (PS)
- possible usage planning
- adaptive resolution using states

Compact state space \( Y = \{ (\sigma, \Omega) \} \) where
\( \sigma = \) aggregate state, \( \Omega = \) compact action space
\( Y \) Consists of regions with two parts \( \rho^a \) and \( \rho^b \)
Q-table stores values for \((\rho^a, \rho^b)\)-pairs \((a \in \rho^a, b \in \rho^b)\)

Background knowledge + variables

\[
\text{state}([\text{on}(A,B), \text{on}(B,\text{table}), \text{on}(C,\text{table}), \neg (A==B), \neg (B==C)])
\]
\[
\text{action}(\text{move}(A,C)), \text{action}(\text{move}(C,A)), \text{action}(\text{move}(A,\text{table}))
\]

Simple Example

\[
\text{State}([\text{on}(a,b), \text{on}(b,\text{table}), \text{clear}(a)])
\]
\[
\text{Action}(\text{move}(a,\text{table}))
\]

\[
\text{State}([\text{on}(b,a), \text{on}(a,\text{table}), \text{clear}(b)])
\]
\[
\text{Action}(\text{move}(a,\text{table}))
\]

Background knowledge
- towerHeight\((X,N)\) \(\iff\) clear\((X)\), towerHeight2\((X,N)\).
- towerHeight2\((X,N)\) \(\iff\) on\((X,Y)\), towerHeight2\((Y,N2)\), \(N2\) is \(N2+1\).
- towerHeight2\((X,1)\) \(\iff\) on\((X,\text{table})\).
Algorithm

For all episodes do
    initialize start state
    while not stop_criterion do
        s is current ground state
        find \( p \in \Psi \) for which \( s \vdash p \)
        choose \( \rho \) (expl. Strat)
        get subs \( \Theta = \{ \rho \} \vdash \omega \)
        take random \( \theta \), from \( \Theta \)
        ground action \( a \vdash \omega \) the
        apply action, observe \( s \) and \( r \)
        find \( Q(p, \omega) = Q(p, \omega) + \alpha(r + \gamma \max_{s' \in \Psi} Q(p, \omega') - Q(p, \omega)) \)
        end while
    End for

\((\mathbf{\alpha, \Omega}) \) is an MDP

Estimate a transition and reward model

Note: each state has its own action space

Do Prioritized Sweeping to speed up learning

\[ \text{State(\{on(a,table),on(b,table),on(c,table),}
\text{clear(a),clear(b),clear(c)\})} \]
\[ \Downarrow \]
\[ \text{State(\{clear(X),clear(Y)\})} \]
\[ \text{Action(move(X,Y),}\}
\text{\{X/D1,Y/D2 \mid D1,D2 \in D\})} \]
\[ \Downarrow \]
\[ \text{Action(move(a,b))} \]
\[ \Downarrow \]
\[ \text{State(\{on(a,b),on(b,table),on(c,table),}
\text{clear(a),clear(c)\})} \]

Some Consequences and Problems

- Action-variables have to be used in state descriptions
  - BK can be ‘too’ powerful
- Predefined space realistic?
  - For Blocks worlds elegant
  - More difficult for larger domains (tic-tac-toe)
  - Possibly introduce Partial Observability
- Exact partition?
  - Maybe difficult to assure
  - Partial solution by Decision Lists
- Computational complexity
  - Working with relational languages is hard
  - Value learning is efficient (PS), but coverage (\( \downarrow \)) is not

\[ \text{State(\{nrTowers(3)\})} \]
\[ \Downarrow \]
\[ \text{Action(move(\_,\_))} \]
Experiments
Small blocks worlds
(3,4,5 blocks)
4 blocks $\Rightarrow$ (#s, #sap):
(73,240) ground
(5,12) compact

Tic-tac-toe

\[
\begin{array}{ccc}
+ & + & + \\
+ & + & + \\
+ & + & + \\
\end{array}
\]

Tic-tac-toe $\Rightarrow$ (#s, #sap):
(26000,>>10000) ground
(15,41) compact

Actions relative to patterns (e.g. line, fork $\Rightarrow$ BK)

How to learn the representation?

- Upgrading Adaptive Resolution techniques
  - Learn Representation and value function
  - Handful existing methods
    (ELF,G,Grossmann,Finton,Reynolds,etc.)
  - Upgrading difficult in relational RL (splitting statistics)

- Bottom-up Generalization from samples (and traces)
  - RL$\Rightarrow$model-building$\Rightarrow$RL$\Rightarrow$model-building$\Rightarrow$
  - A learned model can help
  - We are currently investigating this option
Bottom-Up Generalization

- Theoretically well-founded way of (logical) induction
- Avoids the problems of top-down learning (e.g. a tree)

- No room for details, but an example:

\[
\text{lgg}([\text{state}([\text{on}(a,b),\text{on}(b,\text{table}),\text{clear}(a))],\text{action}(\text{move}(a,\text{table})))], \\
\text{([state}([\text{on}(b,a),\text{clear}(b),\text{on}(a,\text{table})],\text{action}(\text{move}(b,\text{table})))]) = \\
\text{[state}[\text{on}(A,B),\text{on}(a,C),\text{on}(b,D),\text{on}(B,\text{table}),\text{clear}(A)], \\
\text{Action}(\text{move}(A,\text{table})))]
\]

Similar to defining the state space representation by hand

Further Research

- Summary:
  (1) new representation modeling states and actions, not just the Q-function,
  (2) efficient learning due i) to smaller state space and ii) model estimation and PS. Drawback: fixed representation.

And now:
- Analysis, comparison, experiments, efficiency
- Bottom-up generalization in \(\theta\)-subsumption space for adaptive resolution RL
- Relational (/hierarchical) RL into (logic-based) agent programming languages".

* See further "A Characterization of Sapient Agents", (2003), KIMAS’03
  M. van Otterlo, M.A. Wiering, M. Dastani and J.-J.Ch. Meyer