Planning and learning with Tabular Methods

Ilija Simic

28.1.2020
Overview

1. Introduction

2. Dyna: Integrated Planning, Acting and Learning

3. Potential improvements

4. Decision-time planning
Models and planning

- What we assume by a model?
- Distribution vs. sample models
- Planning
- Structure of planning methods
- Learning vs. planning
Process
Algorithm

Tabular Dyna-Q

Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in S$ and $a \in A(s)$

Do forever:

(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \epsilon$-greedy($S, Q$)
(c) Execute action $A$; observe resultant reward, $R$, and state, $S'$
(d) $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
(e) $Model(S, A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Repeat $n$ times:
   $S \leftarrow$ random previously observed state
   $A \leftarrow$ random action previously taken in $S$
   $R, S' \leftarrow Model(S, A)$
   $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
Maze example
Wrong model

- Blocking maze
- Shortcut
Prioritized Sweeping

Idea - backward focusing

Prioritized sweeping for a deterministic environment

- Initialize $Q(s, a)$, $Model(s, a)$, for all $s, a$, and $PQueue$ to empty
- Do forever:
  1. $S \leftarrow$ current (nonterminal) state
  2. $A \leftarrow policy(S, Q)$
  3. Execute action $A$; observe resultant reward, $R$, and state, $S'$
  4. $Model(S, A) \leftarrow R, S'$
  5. $P \leftarrow \vert R + \gamma \max_a Q(S', a) - Q(S, A) \vert$
  6. If $P > \theta$, then insert $S, A$ into $PQueue$ with priority $P$
  7. Repeat $n$ times, while $PQueue$ is not empty:
     - $S, A \leftarrow$ first($PQueue$)
     - $R, S' \leftarrow Model(S, A)$
     - $Q(S, A) \leftarrow Q(S, A) + \alpha \{ R + \gamma \max_a Q(S', a) - Q(S, A) \}$
     - Repeat, for all $S, A$ predicted to lead to $S$:
       - $R \leftarrow$ predicted reward for $S, A, S$
       - $P \leftarrow \vert R + \gamma \max_a Q(S, a) - Q(S, A) \vert$
   - If $P > \theta$ then insert $S, A$ into $PQueue$ with priority $P$

Example 8.4 Prioritized Sweeping on Mazes

Prioritized sweeping has been found to dramatically increase the speed at which optimal solutions are found in maze tasks, often by a factor of 5 to 10. A typical example is shown below.

These data are for a sequence of maze tasks of exactly the same structure as the one shown in Figure 8.3, except that they vary in the grid resolution. Prioritized sweeping maintained a decisive advantage over unprioritized Dyna-Q. Both systems made at most $n = 5$ updates per environmental interaction.
Trajectory sampling
Real-time Dynamic Programming

<table>
<thead>
<tr>
<th></th>
<th>DP</th>
<th>RTDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average computation to convergence</td>
<td>28 sweeps</td>
<td>4000 episodes</td>
</tr>
<tr>
<td>Average number of updates to convergence</td>
<td>252,784</td>
<td>127,600</td>
</tr>
<tr>
<td>Average number of updates per episode</td>
<td>—</td>
<td>31.9</td>
</tr>
<tr>
<td>% of states updated ≤ 100 times</td>
<td>—</td>
<td>98.45</td>
</tr>
<tr>
<td>% of states updated ≤ 10 times</td>
<td>—</td>
<td>80.51</td>
</tr>
<tr>
<td>% of states updated 0 times</td>
<td>—</td>
<td>3.18</td>
</tr>
</tbody>
</table>

Start States: reachable from some start state under some optimal policy
Irrelevant States: unreachable from any start state under any optimal policy
Planning at Decision time

- Background planning
- Decision-time planning
Heuristic Search
Monte Carlo Tree Search

Rollout policy/Tree policy
The End