Decision Making in Robots and Autonomous Agents

Partial Observability and the POMDP Model
(Source: S. Thrun, Probabilistic Robotics; Y. Gal, A. Pfeffer, AAAI Tutorial)

Subramanian Ramamoorthy
School of Informatics

6 February, 2015
A Trace through an MDP

Environment: You are in state 65. You have 4 possible actions.
Agent: I’ll take action 2.
Environment: You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.
Agent: I’ll take action 1.
Environment: You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.
Agent: I’ll take action 2.
Environment: You received a reinforcement of 5 units. You are now in state 44. You have 5 possible actions.

What happens if agent doesn’t get, “You are now in state…” Instead, all the agent gets are, “You now see these observations…”
POMDP Problem

Belief state

Sufficient statistic (Markov property)

AGENT

OBSERVATIONS

STATE

WORLD + AGENT

ACTIONS
POMDPs

- In POMDPs we apply the very same idea as in MDPs.
- **Since the state is not observable**, the agent has to make its decisions based on the belief state which is a **posterior distribution over states**.
- Let $b$ be the belief of the agent about the state under consideration.
- POMDPs compute a **value function over belief space**:

$$V_T(b) = \gamma \max_u \left[ r(b, u) + \int V_{T-1}(b') p(b' | u, b) \, db' \right]$$
Some Problems to Consider

• Each belief is a probability distribution, thus, each value in a POMDP is a function of an entire probability distribution.

• This is problematic, since probability distributions are continuous.

• Additionally, we have to deal with the huge complexity of belief spaces.

• For finite worlds with finite state, action, and measurement spaces and finite horizons, however, we can effectively represent the value functions by piecewise linear functions.
An Illustrative Example

measurements | state $x_1$ | action $u_3$ | state $x_2$ | measurements

$z_1$ 0.7 | $x_1$ | 0.2 | $x_2$ | 0.3 $z_1$

$z_2$ 0.3 | $u_3$ | 0.8 | $u_3$ | 0.7 $z_2$

$u_1$ | $u_2$ | $u_1$ | $u_2$

payoff | 100 | 100 | 100 | -50

payoff

actions $u_1, u_2$
Payoff in POMDPs

- In MDPs, the payoff (or return) depended on the state of the system.
- In POMDPs, however, the true state is not exactly known.
- Therefore, we compute the expected payoff by integrating over all states:

\[
    r(b, u) = E_x[r(x, u)] \\
    = \int r(x, u) p(x) \, dx \\
    = p_1 \, r(x_1, u) + p_2 \, r(x_2, u)
\]
Payoffs in Our Example (1)

- If we are totally certain that we are in state $x_1$ and execute action $u_1$, we receive a reward of -100.
- If, on the other hand, we definitely know that we are in $x_2$ and execute $u_1$, the reward is +100.
- In between it is the linear combination of the extreme values weighted by the probabilities:

\[
r(b, u_1) = -100 p_1 + 100 p_2
\]

\[
= -100 p_1 + 100 (1 - p_1)
\]

\[
r(b, u_2) = 100 p_1 - 50 (1 - p_1)
\]

\[
r(b, u_3) = -1
\]
Payoffs in Our Example (2)

\[ r(b, u_1) \]

\[ r(b, u_2) \]

\[ r(b, u_3) \]

\[ V_1(b) = \max_u r(b, u) \]
The Resulting Policy for T=1

• Given we have a finite POMDP with T=1, we would use $V_1(b)$ to determine the optimal policy.

• In our example, the optimal policy for T=1 is

\[
\pi_1(b) = \begin{cases} 
  u_1 & \text{if } p_1 \leq \frac{3}{7} \\
  u_2 & \text{if } p_1 > \frac{3}{7} 
\end{cases}
\]

• This is the upper thick graph in the diagram.
Piecewise Linearity, Convexity

• The resulting value function $V_1(b)$ is the maximum of the three functions at each point

$$V_1(b) = \max_u r(b, u)$$

$$= \max \left\{ \begin{array}{ccc}
-100 p_1 & +100 (1 - p_1) \\
100 p_1 & -50 (1 - p_1) \\
-1 & 
\end{array} \right\}$$

• It is piecewise linear and convex.
Pruning

• If we carefully consider $V_1(b)$, we see that only the first two components contribute.
• The third component can therefore safely be pruned away from $V_1(b)$.

\[
V_1(b) = \max \begin{Bmatrix}
-100 p_1 + 100 (1 - p_1) \\
100 p_1 - 50 (1 - p_1)
\end{Bmatrix}
\]
Increasing the Time Horizon

• Assume the robot can make an observation before deciding on an action.
Increasing the Time Horizon

• Assume the robot can make an observation before deciding on an action.
• Suppose the robot perceives $z_1$ for which $p(z_1 \mid x_1) = 0.7$ and $p(z_1 \mid x_2) = 0.3$.
• Given the observation $z_1$ we update the belief using Bayes rule.

$$p'_{1} = \frac{0.7 p_{1}}{p(z_1)}$$
$$p'_{2} = \frac{0.3(1 - p_{1})}{p(z_1)}$$

$$p(z_1) = 0.7 p_{1} + 0.3(1 - p_{1}) = 0.4 p_{1} + 0.3$$
Value Function

$V_1(b)$

$b' (b|z_1)$

$x_1$

$x_2$

$x_1$

$V_1(b|z_1)$
Increasing the Time Horizon

- Assume the robot can make an observation before deciding on an action.
- Suppose the robot perceives $z_1$ for which $p(z_1 | x_1) = 0.7$ and $p(z_1 | x_2) = 0.3$.
- Given the observation $z_1$ we update the belief using Bayes rule.
- Thus $V_1(b | z_1)$ is given by

$$V_1(b | z_1) = \max \left\{ -100 \cdot \frac{0.7 p_1}{p(z_1)} + 100 \cdot \frac{0.3 (1-p_1)}{p(z_1)}, 100 \cdot \frac{0.7 p_1}{p(z_1)} - 50 \cdot \frac{0.3 (1-p_1)}{p(z_1)} \right\}$$

$$= \frac{1}{p(z_1)} \max \left\{ -70 p_1 + 30 (1-p_1), 70 p_1 - 15 (1-p_1) \right\}$$

6/2/2016
Expected Value after Measuring

- Since we do not know in advance what the next measurement will be, we have to compute the expected belief

\[
\overline{V}_1(b) = E_z[V_1(b \mid z)] = \sum_{i=1}^{2} p(z_i) V_1(b \mid z_i)
\]

\[
= \sum_{i=1}^{2} p(z_i) V_1 \left( \frac{p(z_i \mid x_1) p_1}{p(z_i)} \right)
\]

\[
= \sum_{i=1}^{2} V_1 \left( p(z_i \mid x_1) p_1 \right)
\]
Expected Value after Measuring

• Since we do not know in advance what the next measurement will be, we have to compute the expected belief

\[
\bar{V}_1(b) = E_z[V_1(b \mid z)] = \sum_{i=1}^{2} p(z_i) V_1(b \mid z_i) = \max \left\{ \begin{array}{cc}
-70 p_1 & +30 (1 - p_1) \\
70 p_1 & -15 (1 - p_1)
\end{array} \right\} + \max \left\{ \begin{array}{cc}
-30 p_1 & +70 (1 - p_1) \\
30 p_1 & -35 (1 - p_1)
\end{array} \right\}
\]
Resulting Value Function

- The four possible combinations yield the following function which then can be simplified and pruned.

\[
\bar{V}_1(b) = \max \begin{cases} 
-70 p_1 & +30 (1 - p_1) & -30 p_1 & +70 (1 - p_1) \\
-70 p_1 & +30 (1 - p_1) & +30 p_1 & -35 (1 - p_1) \\
+70 p_1 & -15 (1 - p_1) & -30 p_1 & +70 (1 - p_1) \\
+70 p_1 & -15 (1 - p_1) & +30 p_1 & -35 (1 - p_1) 
\end{cases}
\]

\[
= \max \begin{cases} 
-100 p_1 & +100 (1 - p_1) \\
+40 p_1 & +55 (1 - p_1) \\
+100 p_1 & -50 (1 - p_1) 
\end{cases}
\]
Value Function

\[
p(z_1) V_1(b|z_1)
\]

\[
p(z_2) V_2(b|z_2)
\]

\[
\bar{V}_1(b)
\]
State Transitions (Prediction)

• When the agent selects \( u_3 \) its state potentially changes.

• When computing the value function, we have to take these potential state changes into account.

\[
p'_1 = E_x[p(x_1 \mid x, u_3)] \\
= \sum_{i=1}^{2} p(x_1 \mid x_i, u_3) p_i \\
= 0.2p_1 + 0.8(1 - p_1) \\
= 0.8 - 0.6p_1
\]
State Transitions (Prediction)

\[ p'_1 = E_x[p(x_1 \mid x, u_3)] \]
\[ = \sum_{i=1}^{2} p(x_1 \mid x_i, u_3)p_i \]
\[ = 0.2p_1 + 0.8(1 - p_1) \]
\[ = 0.8 - 0.6p_1 \]
Resulting Value Function after executing $u_3$

- Taking the state transitions into account, we finally obtain.

$$
\bar{V}_1(b) = \max \left\{ \begin{array}{cccc}
-70 \ p_1 & +30 \ (1 - p_1) & -30 \ p_1 & +70 \ (1 - p_1) \\
-70 \ p_1 & +30 \ (1 - p_1) & +30 \ p_1 & -35 \ (1 - p_1) \\
+70 \ p_1 & -15 \ (1 - p_1) & -30 \ p_1 & +70 \ (1 - p_1) \\
+70 \ p_1 & -15 \ (1 - p_1) & +30 \ p_1 & -35 \ (1 - p_1) \\
\end{array} \right\}
$$

$$
= \max \left\{ \begin{array}{cc}
-100 \ p_1 & +100 \ (1 - p_1) \\
+40 \ p_1 & +55 \ (1 - p_1) \\
+100 \ p_1 & -50 \ (1 - p_1) \\
\end{array} \right\}
$$

$$
\bar{V}_1(b \mid u_3) = \max \left\{ \begin{array}{cc}
60 \ p_1 & -60 \ (1 - p_1) \\
52 \ p_1 & +43 \ (1 - p_1) \\
-20 \ p_1 & +70 \ (1 - p_1) \\
\end{array} \right\}
$$
Value Function after executing $u_3$

$\bar{V}_1(b)$

$\bar{V}_1(b \mid u_3)$
Value Function for $T=2$

- Taking into account that the agent can either directly perform $u_1$ or $u_2$ or first $u_3$ and then $u_1$ or $u_2$, we obtain (after pruning)

$$
\bar{V}_2(b) = \max \left\{ \begin{array}{ccc}
-100p_1 & +100(1-p_1) \\
100p_1 & -50(1-p_1) \\
51p_1 & +42(1-p_1) 
\end{array} \right\}
$$
Graphical Representation of $V_2(b)$

The outcome of measurement is important here.
Deep Horizons and Pruning

• We have now completed a full backup in belief space.
• This process can be applied recursively.
• The value functions for $T=10$ and $T=20$ are:
Why Pruning is Essential

- Each update introduces additional linear components to $V$.
- Each measurement squares the number of linear components.
- Thus, an un-pruned value function for $T=20$ includes more than $10^{547,864}$ linear functions.
- At $T=30$ we have $10^{561,012,337}$ linear functions.
- The pruned value functions at $T=20$, in comparison, contains only 12 linear components.
- The combinatorial explosion of linear components in the value function are the major reason why POMDPs are impractical for most applications.
POMDPs as Bayesian Networks
Influence Diagrams [Howard & Matheson ‘84]

- Influence Diagrams extend Bayesian Networks for decision making.
- *Rectangles* are decisions; *ovals* are chance variables; *diamonds* are utility functions.
- Graph topology describes decision problem.
- Each node specifies a probability distribution (CPD) given each value of parents.
Influence Diagrams (ID)

- Parents of decisions (informational parents) represent observations.
- Parents of chance nodes represent probabilistic dependence.
- Parents of utility nodes represent the parameters of the utility functions.
- A strategy for a decision is a function from its informational parents to a choice for the decision. For each observation, a *pure* strategy prescribes a single choice of action for an agent.
Bob observes the weather forecast before deciding whether to carry an umbrella to work. Bob wishes to stay dry, but carrying an umbrella around is annoying.
Influence Diagram for Umbrella Example

<table>
<thead>
<tr>
<th>Weather</th>
<th>Forecast</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>0.7</td>
<td>-10</td>
</tr>
<tr>
<td>rain</td>
<td>0.3</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>Forecast</th>
<th>Umbrella</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>TRUE</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td>sun</td>
<td>FALSE</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>rain</td>
<td>TRUE</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>rain</td>
<td>FALSE</td>
<td>-10</td>
<td></td>
</tr>
</tbody>
</table>

6/2/2016
Extended Umbrella Example

The newspaper forecast is more reliable, but costs money, decreasing Bob’s utility by 10 units. There are now two decisions:

– Buying a newspaper
– Carrying an umbrella

<table>
<thead>
<tr>
<th>Weather</th>
<th>Forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>0.8</td>
</tr>
<tr>
<td>rain</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weather</th>
<th>NP</th>
<th>Umbrella</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>TRUE</td>
<td>TRUE</td>
<td>-20</td>
</tr>
<tr>
<td>sun</td>
<td>TRUE</td>
<td>FALSE</td>
<td>90</td>
</tr>
<tr>
<td>rain</td>
<td>TRUE</td>
<td>TRUE</td>
<td>90</td>
</tr>
<tr>
<td>rain</td>
<td>TRUE</td>
<td>FALSE</td>
<td>-20</td>
</tr>
<tr>
<td>...</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
</tbody>
</table>
New decision node *Newspaper* added, that affects the forecast and Bob’s annoyance level.
IDs and BNs

• A new chance node implements a strategy $s$ for decision $D$ if it has the same informational parents as $D$ and chooses the same action as does $s$ for each instantiation of the parents of $D$.

• A chance node implements a utility node $V$ if it assigns probability 1 to the value associated with the utility node for each instantiation of the parents.
A BN implements an ID given strategies for all decisions if it implements all decisions and chance nodes.
Converting IDs to Decision-trees

1. Traverse ID top-down.
2. If node is a decision or an informational parent
   - create vertex in decision tree.
   - create edges and label with node values.
3. Compute probability of each value of informational parents and annotate edge.
4. Label leaves with expected utility for agent given a path instantiating values for all decisions and informational parents.

Steps 3 and 4 need to ‘query’ Bayesian network
Converting ID to Decision Tree: Umbrella Example

Disadvantage: We lose graph structure
MDP as an Influence Diagram
POMDP as an Influence Diagram