Probabilistic Planning

Finding a sequence of actions to achieve some goal.

As before:
• Generalize logic to probabilities.
• Probabilistic planning.

This results in a harder planning problem.

In particular, we can no longer compute straight-line plans.
Probabilistic Planning

Recall:
- Problem has a state.
- State has the Markov property.

Markov Decision Processes (MDPs):
- Canonical probabilistic planning formulation.
- Problem has a set of states.
- Actions cause stochastic transitions.
- Actions have costs/rewards.

MDPs

An MDP is a tuple:

\(< S, A, R, T, \gamma >\)

\(S\): set of states.
\(A\): set of actions.
\(R\): reward function \(R(s, a, s') \to \mathbb{R}\)
\(T\): transition function \(T(s'|s, a) \to [0, 1]\)
\(\gamma\): discount factor \(\gamma \in (0, 1]\)

Some states “absorbing”.

Example

![Diagram of a grid world with probabilities and rewards]

Example

![Diagram of a simple MDP with states A, B, C and transition probabilities]

\(0.8\) \(r=-2\)

\(0.2\) \(r=-5\)
MDPs

Our goal is to find a policy:

\[ \pi : S \rightarrow A \]

That maximizes return: expected sum of rewards.
(equiv: min sum of costs)

\[ \sum_{i=1}^{\infty} \mathbb{E}[\gamma^i r_i] \]

Policies and Plans

Compare a policy:
• An action for every state.

… with a plan:
• A sequence of actions.

Why the difference?

Planning

So our goal is to produce optimal policy.

\[ \pi^*(s) = \max_{\pi} R^\pi(s) \]

Define the value function to estimate this quantity:

\[ V^\pi(s) = \mathbb{E} \left[ \sum_{i=0}^{\infty} \gamma^i r_i(s_i) \right] \]

How to find \( V \)?

Bellman

Bellman's equation is a condition that must hold for \( V \):

\[ V^\pi(s) = r(s, \pi(s)) + \gamma \max_a \sum_{s'} T(s'|s, a) V^\pi(s') \]

reward

value of this state

expected value of next state
Value Iteration

This gives us an algorithm for learning the value function for a specific given fixed policy:

Repeat:
• Make a copy of the VF.
• For each state in VF, assign value using BE.
• Replace old VF.

This is known as value iteration. (In practice, only adjust “reachable” states.)

Why do we care so much about VF?

Policy Iteration

We can adjust the policy given the VF:

$$\pi(s) = \max_a \left[ R(s, a) + \gamma \sum_{s'} T(s'|s, a)V(s') \right]$$

Adjust policy to be greedy w.r.t VF.

We can alternate value and policy iteration.
Surprising results:
• This converges even if alternate every step.
• Converges to optimal policy.
• Converges in polynomial time.

Elevator Scheduling

Crites and Barto (1985):
• System with 4 elevators, 10 floors.
• Realistic simulator.
• 46 dimensional state space.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AvgWait</th>
<th>QueueWait</th>
<th>SystemTime</th>
<th>Percent&gt;30 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECTOR</td>
<td>30.3</td>
<td>164.3</td>
<td>39.5</td>
<td>12.40</td>
</tr>
<tr>
<td>RUFF</td>
<td>32.6</td>
<td>88.1</td>
<td>39.8</td>
<td>3.18</td>
</tr>
<tr>
<td>LAP</td>
<td>39.8</td>
<td>87.4</td>
<td>43.9</td>
<td>4.92</td>
</tr>
<tr>
<td>BASIC RUFF</td>
<td>39.9</td>
<td>87.5</td>
<td>44.7</td>
<td>4.94</td>
</tr>
<tr>
<td>FIM</td>
<td>39.6</td>
<td>88.4</td>
<td>43.4</td>
<td>3.15</td>
</tr>
<tr>
<td>EMA</td>
<td>40.1</td>
<td>88.7</td>
<td>32.3</td>
<td>3.13</td>
</tr>
<tr>
<td>BLD</td>
<td>18.3</td>
<td>100.0</td>
<td>48.4</td>
<td>2.80</td>
</tr>
<tr>
<td>BLDm</td>
<td>18.6</td>
<td>145.2</td>
<td>42.7</td>
<td>2.49</td>
</tr>
</tbody>
</table>

MicroMAP

“Drivers and Loads” (trucking), CASTLE lab at Princeton

“the model was used by 20 of the largest truckload carriers to dispatch over 66,000 drivers”
Back to PDDL

Note that an MDP does not contain the structure of PDDL.

If we wish to combine that structure and probabilistic planning, we can use a related language called PPDDL - probabilistic problem domain definition language.

PPDDL Operators

Now operators have probabilistic outcomes:

```
(:action move_left
  :parameters (x, y)
  :precondition (not (wall(x-1, y)))
  :effect (probabilistic
    0.8 (and (at(x-1)) (not at(x)) (decrease (reward) 1))
    0.2 (and (at(x+1)) (not(at(x))(decrease (reward) 1))
  )
)
```

PPDDL

Instead of computing a plan, we again need a policy.

Most planners:
- Take as input PPDDL.
- Use a simulator.
- Compute policy for states reachable from start.
- Are evaluated jointly with simulator.

Robot Planning

One more common type of planning left!