Introduction to Artificial Intelligence
Uncertain Planning - MDP Algorithms

Janyl Jumadinova
November 11, 2016
Decision Processes

A Markov decision process augments a Markov chain with actions and reward values:
Markov Decision Process

Diagram:
- Agent
- Environment
- State $s_t$
- Reward $r_t$
- Action $a_t$
- Next State $s_{t+1}$
Example: to exercise or not?

States: \{fit, unfit\}

Actions: \{exercise, relax\}
Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

**States:** \{fit, unfit\}  **Actions:** \{exercise, relax\}
Example: to exercise or not?

Each week *Sam* has to decide whether to exercise or not:

**States:** \{fit, unfit\}  **Actions:** \{exercise, relax\}

### Dynamics:

| State  | Action | $P(\text{fit}|\text{State, Action})$ |
|--------|--------|----------------------------------|
| fit    | exercise | 0.99                             |
| fit    | relax     | 0.7                               |
| unfit  | exercise  | 0.2                               |
| unfit  | relax     | 0.0                               |

**Reward (does not depend on resulting state):**

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit</td>
<td>exercise</td>
<td>8</td>
</tr>
<tr>
<td>fit</td>
<td>relax</td>
<td>10</td>
</tr>
<tr>
<td>unfit</td>
<td>exercise</td>
<td>0</td>
</tr>
<tr>
<td>unfit</td>
<td>relax</td>
<td>5</td>
</tr>
</tbody>
</table>
Grid World Example
▶ The agent lives in a grid.
▶ Walls block the agent’s path.
The agent lives in a grid.

Walls block the agent’s path.

The agent’s actions do not always go as planned:

- 80% of the time, the action North takes the agent North (if there is no wall there).
- 10% of the time, North takes the agent West; 10% East.
The agent lives in a grid.

Walls block the agent’s path.

The agent’s actions do not always go as planned:

- 80% of the time, the action North takes the agent North (if there is no wall there).
- 10% of the time, North takes the agent West; 10% East.
- If there is a wall in the direction the agent would have been taken, the agent stays put.
The agent lives in a grid.
Walls block the agent’s path.
The agent’s actions do not always go as planned:
80% of the time, the action North takes the agent North (if there is no wall there).
10% of the time, North takes the agent West; 10% East.
If there is a wall in the direction the agent would have been taken, the agent stays put.
Big rewards come at the end.
Information Availability

What information is available when the agent decides what to do?

- **Fully-observable MDP** the agent gets to observe $S_t$ when deciding on action $A_t$. 
Information Availability

What information is available when the agent decides what to do?

- **Fully-observable MDP** the agent gets to observe $S_t$ when deciding on action $A_t$.

- **Partially-observable MDP (POMDP)** the agent has some noisy sensor of the state. It needs to remember its sensing and acting history.
A stationary policy is a function:

\[ \pi : S \rightarrow A \]

Given a state \( s \), \( \pi(s) \) specifies what action the agent who is following \( \pi \) will do.

An optimal policy is one with maximum expected discounted reward.
In an MDP, we want an optimal policy $\pi^* : S : H \rightarrow A$

- A policy $\pi$ gives an action for each state for each time
- An optimal policy maximizes expected sum of rewards
Solving MDPs

- In an MDP, we want an optimal policy $\pi^* : S : H \rightarrow A$
- A policy $\pi$ gives an action for each state for each time
- An optimal policy maximizes expected sum of rewards

**Contrast:** In deterministic, want an optimal plan, or sequence of actions, from start to a goal
Solutions

**Exact Methods:**
- Value Iteration
- Policy Iteration
- Linear Programming
Value Iteration Algorithm

\[ V_i^*(s) = \text{the expected sum of rewards accumulated when starting from state } s \text{ and acting optimally for a horizon of } i \text{ steps} \]

Start with \[ V_0^*(s) = 0 \] for all \( s \).
For \( i = 1, \ldots, H \)
Given \( V_i^* \), calculate for all states \( s \in S \):

\[ V_{i+1}^*(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + V_i^*(s') \right] \]

This is called a value update or Bellman update/back-up
Value Iteration in Gridworld

\( \text{noise} = 0.2, \gamma = 0.9, \) two terminal states with \( R = +1 \) and \( -1 \)
Value Iteration in Gridworld

\[ \text{noise} = 0.2, \gamma = 0.9, \text{ two terminal states with } R = +1 \text{ and } -1 \]
Value Iteration in Gridworld

\( \text{noise} = 0.2, \ \gamma = 0.9, \) two terminal states with \( R = +1 \) and \( -1 \)
Value Iteration in Gridworld

\[ \text{noise} = 0.2, \ \gamma = 0.9, \ \text{two terminal states with } R = +1 \text{ and } -1 \]
Value Iteration in Gridworld

\( \text{noise} = 0.2, \gamma = 0.9, \) two terminal states with \( R = +1 \) and \( -1 \)
Value Iteration in Gridworld

\( \text{noise} = 0.2, \gamma = 0.9, \) two terminal states with \( R = +1 \) and \( -1 \)
Value Iteration in Gridworld

\[ \text{noise} = 0.2, \gamma = 0.9, \text{two terminal states with } R = +1 \text{ and } -1 \]
Effect of discount, noise

(a) Prefer the close exit (+1), risking the cliff (-10)
(b) Prefer the close exit (+1), but avoiding the cliff (-10)
(c) Prefer the distant exit (+10), risking the cliff (-10)
(d) Prefer the distant exit (+10), avoiding the cliff (-10)

(1) $\gamma = 0.1$, noise = 0.5
(2) $\gamma = 0.99$, noise = 0
(3) $\gamma = 0.99$, noise = 0.5
(4) $\gamma = 0.1$, noise = 0