Lecture Overview

Markov Decision Processes

- Formal Specification and example
- Policies and Optimal Policy
- Intro to Value Iteration
Combining ideas for Stochastic planning

• What is a key limitation of decision networks?
  
  *Represent (and optimize) only a fixed number of decisions*

• What is an advantage of Markov models?
  
  *The network can extend indefinitely*

**Goal:** represent (and optimize) an indefinite sequence of decisions
Decision Processes

Often an agent needs to go beyond a fixed set of decisions – Examples?

• Would like to have an ongoing decision process

Infinite horizon problems: process does not stop

Robot surviving on planet, Monitoring Nuc. Plant, ....

Indefinite horizon problem: the agent does not know when the process may stop

Finite horizon: the process must end at a give time $N$
Markov Models

- Markov Chains
- Hidden Markov Model
- Partially Observable Markov Decision Processes (POMDPs)
- Markov Decision Processes (MDPs)

 noisy actions rewards

 noisy observations
How can we deal with indefinite/infinite Decision processes?

We make the same two assumptions we made for:

The action outcome depends only on the current state

\[ P(S_{t+1} | S_t, A_t, S_{t-1}, A_{t-1}, \ldots) \]

Let \( S_t \) be the state at time \( t \) …

The process is stationary…

\[ P(S_{t+1} | S_t, A_t) \]

the same for all \( t \)

We also need a more flexible specification for the utility. How?

- Defined based on a reward/punishment \( R(s) \) that the agent receives in each state \( s \)

\[ s_0, s_1, \ldots, s_n \]

\[ r_0, r_1, \ldots, r_n \]
MDP graphical specification

Basically a MDP is a Markov Chain augmented with actions and rewards/values
When Rewards only depend on the state
Summary Decision Processes: MDPs

To manage an ongoing (indefinite… infinite) decision process, we combine….

Markov Chains & Decision Networks

Markovian
Stationary

Utility not just at the end

Sequence of rewards

Fully Observable
MDP: formal specification

For an MDP you specify:

- set $S$ of states and set $A$ of actions
- the process’ dynamics (or *transition model*)
  \[ P(S_{t+1} | S_t, A_t) \]
- The reward function
  - $R(s)$ is used when the reward depends only on the state $s$ and not on how the agent got there
  - More complex $R(s, a, s')$ describing the reward that the agent receives when it performs action $a$ in state $s$ and ends up in state $s'$
- Absorbing/stopping/terminal state

\[ \forall s, a, s' \quad P(s_{ab} | a, s_{ab}) = 1 \quad R(s_{ab}, a, s_{ab}) = 0 \]
Example MDP: Scenario and Actions

Agent moves in the above grid via actions Up, Down, Left, Right

Each action has:

- 0.8 probability to reach its intended effect
- 0.1 probability to move at right angles of the intended direction
- If the agent bumps into a wall, it says there

How many states? \[ \{(1,1), (1,2), \ldots, (2,4), (3,4)\} \]

There are two terminal states (3,4) and (2,4)
Example MDP: Rewards

\[ R(s) = \begin{cases} 
-0.04 & \text{(small penalty) for nonterminal states} \\ 
\pm 1 & \text{for terminal states} 
\end{cases} \]
Example MDP: Underlying Info structures

Four actions *Up, Down, Left, Right*

Eleven States: \{(1,1), (1,2), \ldots, (3,4)\}

**Table** $4 \times 11 \times 11 \quad P(S_{t+1} | S_t, A_t)$

<table>
<thead>
<tr>
<th>Up</th>
<th>1,1</th>
<th>2,1</th>
<th>1,2</th>
<th>3,1</th>
<th>Down</th>
<th>L</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>0.1</td>
<td>0.8</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2,1</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

**Diagram**

- States: Start (1,1)
- Actions: Up, Down, Left, Right
- Rewards: +1 at (3,4), -1 at (2,3)

**Note**

- Transition probabilities and rewards for each action and state transition.
Example MDP: Sequence of actions

The sequence of actions [Up, Up, Right, Right, Right] will take the agent in terminal state (3,4)...

A. always  B. never  C. Only sometimes

With what probability?

A. \((0.8)^5\)  B. \((0.8)^5 + ((0.1)^4 \times 0.8)\)  C. \(((0.1)^4 \times 0.8)\)
Example MDP: Sequence of actions

Can the sequence \([Up, Up, Right, Right, Right]\) take the agent in terminal state (3,4)?

\[(.8)^5\]

Can the sequence reach the goal in any other way?

\[(.1)^4 .8 \lessgtr \text{ with prob } \]

\[\text{yes no}\]
MDPs: Policy

• The robot needs to know what to do as the decision process unfolds…

• It starts in a state, selects an action, ends up in another state selects another action….

• Needs to make the same decision over and over: Given the current state what should I do?

• So a policy for an MDP is a single decision function $\pi(s)$ that specifies what the agent should do for each state $s$. 

CPSC 422, Lecture 3
How to evaluate a policy
(in essence how to compute $V^\pi(s)$ brute force)

A policy can generate a set of state sequences with different probabilities

Each state sequence has a corresponding reward. Typically the (discounted) sum of the rewards for each state in the sequence

$$\sum ((1,1) \rightarrow (1,1) \rightarrow (2,1) \rightarrow (3,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (3,4))$$

$$+ \cdot 72$$
MDPs: expected value/total reward of a policy and optimal policy

Each sequence of states (environment history) associated with a policy has

- a certain **probability** of occurring
- a given amount of total **reward** as a function of the rewards of its individual states

**Expected value / total reward**

\[
\sum P(s_0, s_1, \ldots, s_{\text{terminal}}) \times \sum R(s_0) \ldots R(s_T)
\]

For all the sequences of states generated by the policy, we sum the product of its probability times its reward.

**Optimal policy** is the policy that maximizes **expected total reward**
Lecture Overview

Markov Decision Processes
• Formal Specification and example
• Policies and Optimal Policy
• Intro to Value Iteration
Sketch of ideas to find the optimal policy for a MDP (Value Iteration)

We first need a couple of definitions

- **$V^\pi(s)$**: the expected value of following policy $\pi$ in state $s$
- **$Q^\pi(s, a)$**, where $a$ is an action: expected value of performing $a$ in $s$, and then following policy $\pi$.

Can we express $Q^\pi(s, a)$ in terms of $V^\pi(s)$?

\[
Q^\pi(s, a) = V^\pi(s) + R(s) \quad \text{(A)}
\]

\[
Q^\pi(s, a) = R(s) + \sum_{s' \in X} P(s' | s, a) \cdot V^\pi(s') \quad \text{(B)}
\]

\[
Q^\pi(s, a) = R(s) + \sum_{s' \in X} V^\pi(s') \quad \text{(C)}
\]

D. None of the above

X: set of states reachable from $s$ by doing $a$
Discounted Reward Function

- Suppose the agent goes through states $s_1, s_2, ..., s_k$ and receives rewards $r_1, r_2, ..., r_k$

- We will look at discounted reward to define the reward for this sequence, i.e. its utility for the agent

\[ \gamma \text{ discount factor, } 0 \leq \gamma \leq 1 \]

\[ U[s_1, s_2, s_3, ...] = r_1 + \gamma r_2 + \gamma^2 r_3 + ..... \]
Sketch of ideas to find the optimal policy for a MDP (Value Iteration)

We first need a couple of definitions

- $V^\pi(s)$: the expected value of following policy $\pi$ in state $s$
- $Q^\pi(s, a)$, where $a$ is an action: expected value of performing $a$ in $s$, and then following policy $\pi$.

We have, by definition

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) V^\pi(s')$$

- reward obtained in $s$
- Discount factor
- states reachable from $s$ by doing $a$
- Probability of getting to $s'$ from $s$ via $a$
- expected value of following policy $\pi$ in $s'$
Value of a policy and Optimal policy

We can also compute $V^\pi(s)$ in terms of $Q^\pi(s, a)$

$$V^\pi(s) = Q^\pi(s, \pi(s))$$

For the optimal policy $\pi^*$ we also have

$$V^{\pi^*}(s) = Q^{\pi^*}(s, \pi^*(s))$$
Value of Optimal policy

\[ V^{\pi^*}(s) = Q^{\pi^*}(s, \pi^*(s)) \]

Remember for any policy \( \pi \)

\[ Q^{\pi}(s, \pi(s)) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) \times V^{\pi}(s') \]

But the Optimal policy \( \pi^* \) is the one that gives the action that maximizes the future reward for each state

\[ Q^{\pi^*}(s, \pi^*(s)) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) \times V^{\pi^*}(s') \]

So...

\[ V^{\pi^*}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) \times V^{\pi^*}(s') \]
Value Iteration Rationale

- Given $N$ states, we can write an equation like the one below for each of them:

$$V(s_1) = R(s_1) + \gamma \max_a \sum_{s'} P(s'|s_1, a)V(s')$$

$$V(s_2) = R(s_2) + \gamma \max_a \sum_{s'} P(s'|s_2, a)V(s')$$

- Each equation contains $N$ unknowns – the $V$ values for the $N$ states

- $N$ equations in $N$ variables (Bellman equations): It can be shown that they have a unique solution: the values for the optimal policy

- Unfortunately the $N$ equations are non-linear, because of the max operator: Cannot be easily solved by using techniques from linear algebra

- **Value Iteration Algorithm**: Iterative approach to find the optimal policy and corresponding values
Learning Goals for today’s class

You can:

- Compute the probability distribution on states given a sequence of actions in an MDP
- Define a policy for an MDP
- Define and Justify a discounted reward function
- Derive the Bellman equations on which Value Iteration is based (we will likely finish this in the next lecture)
Read textbook
• 9.5.3 Value Iteration

TODO for Wed
CPSC 322 Review “Exam”

https://forms.gle/SpQwrXfonTZrVf4P7

Based on CPSC 322 material

- Logic
- Uncertainty
- Decision Theory

Review material (e.g., 322 slides from 2017):