Intelligent Systems (AI-2)

Computer Science cpsc422, Lecture 4

Sep, 11, 2019
Announcements

Office Hours have been posted

- Giuseppe Carenini  ICICS (CICSR) 105  Wed 1-2
- Grigorii Guz  ICCS X 237, Th 2-3  (already starting this week!)
- Daniele Reda  ICCS X 237, Tu 4-5
- Matthew Wilson  ICCS X 237, Mon 5-6  (right after class)

What to do with readings? In a few lectures we will discuss the first research paper. Instructions on what to do are available on the course webpage.

Discussions on Piazza (84%) - Signup Link:
piazza.com/ubc.ca/winterterm12019/cpsc422101
Just an example of popularity of these R&R methods: In 2016 I was program co-chair of the SigDial Conference.

Four papers using (PO)MDP & Reinforcement Learning!

Four papers in 2017 as well......
Lecture Overview

Markov Decision Processes

• Some ideas and notation
• Finding the Optimal Policy
  • Value Iteration
• From Values to the Policy
  • Rewards and Optimal Policy
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 agents bumps into a wall, it says there

Eleven states

Two terminal states (4,3) and (4,2)
Example MDP: Rewards

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

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

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

➢ Example for state (1,1)

\[
V(1,1) = -0.04 + 1 \times \max \begin{bmatrix}
0.8V(1,2) + 0.1V(2,1) + 0.1V(1,1) & UP \\
0.9V(1,1) + 0.1V(1,2) & LEFT \\
0.9V(1,1) + 0.1V(2,1) & DOWN \\
0.8V(2,1) + 0.1V(1,2) + 0.1V(1,1) & RIGHT
\end{bmatrix}
\]

(sorry (column, row) to indicate state)
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 \( V \) values and the corresponding optimal policy
Let $V^{(i)}(s)$ be the utility of state $s$ at the $i^{th}$ iteration of the algorithm.

Start with arbitrary utilities on each state $s$: $V^{(0)}(s)$

Repeat simultaneously for every $s$ until there is “no change”

$$V^{(k+1)}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^{(k)}(s')$$

True “no change” in the values of $V(s)$ from one iteration to the next are guaranteed only if run for infinitely long.

- In the limit, this process converges to a unique set of solutions for the Bellman equations.
- They are the total expected rewards (utilities) for the optimal policy.
Example

Suppose, for instance, that we start with values $V^{(0)}(s)$ that are all 0.

Iteration 0

\[
\begin{array}{c|c|c|c|c}
3 & 0 & 0 & 0 & +1 \\
2 & 0 & [ & 0 & -1 \\
1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Iteration 1

\[
\begin{array}{c|c|c|c|c}
3 & 0 & 0 & 0 & +1 \\
2 & 0 & 0 & [ & 0 \\
1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
V^{(1)}(1,1) = -0.04 + 1 \times \max \begin{bmatrix}
0.8V^{(0)}(1,2) + 0.1V^{(0)}(2,1) + 0.1V^{(0)}(1,1) \\
0.9V^{(0)}(1,1) + 0.1V^{(0)}(1,2) \\
0.9V^{(0)}(1,1) + 0.1V^{(0)}(2,1) \\
0.8V^{(0)}(2,1) + 0.1V^{(0)}(1,2) + 0.1V^{(0)}(1,1)
\end{bmatrix}
\]

\[
V^{(1)}(1,1) = -0.04 + \max \begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

CPSC 422, Lecture 4
Let’s compute $V^{(1)}(3,3)$

\[
V^{(1)}(3,3) = -0.04 + 1 \times \max \left[ 0.8V^{(0)}(3,3) + 0.1V^{(0)}(2,3) + 0.1V^{(0)}(4,3),
0.8V^{(0)}(2,3) + 0.1V^{(0)}(3,3) + 0.1V^{(0)}(3,2),
0.8V^{(0)}(3,2) + 0.1V^{(0)}(2,3) + 0.1V^{(0)}(4,3),
0.8V^{(0)}(4,3) + 0.1V^{(0)}(3,3) + 0.1V^{(0)}(3,2) \right]
\]

\[
V^{(1)}(3,3) = -0.04 + \max \left[ 0.8 \times 0, 0, 0, 0.8 \times 0.76 \right]
\]

Iteration 0

\[
\begin{array}{cccc}
3 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{array}
\]

Iteration 1

\[
\begin{array}{cccc}
3 & 0 & 0 & 0.76 \\
2 & 0 & 0 & 0 \\
1 & -0.04 & 0 & 0 \\
\end{array}
\]
Example (cont’d)

➢ Let’s compute $V^{(1)}(4,1)$

\[
V^{(1)}(4,1) = -0.04 + \max \begin{bmatrix}
0.8V^{(0)}(4,2) + 0.1V^{(0)}(3,1) + 0.1V^{(0)}(4,1) & UP \\
0.8V^{(0)}(3,1) + 0.1V^{(0)}(4,2) + 0.1V^{(0)}(4,1) & LEFT \\
0.9V^{(0)}(4,1) + 0.1V^{(0)}(3,1) & DOWN \\
0.9V^{(0)}(4,1) + 0.1V^{(0)}(4,2) & RIGHT
\end{bmatrix}
\]

\[
V^{(1)}(4,1) = -0.04 + \max \begin{bmatrix}
-0.8 & UP \\
-0.1 & LEFT \\
0 & DOWN \\
-0.1 & RIGHT
\end{bmatrix}
\]
After a Full Iteration

- Only the state one step away from a positive reward (3,3) has gained value, all the others are losing value.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-0.04</td>
<td>-0.04</td>
<td>0.76</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>-0.04</td>
<td>0.04</td>
<td>-0.04</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>-0.04</td>
<td>-0.04</td>
<td>-0.04</td>
<td>-0.04</td>
</tr>
</tbody>
</table>

Iteration 1
Some steps in the second iteration

**Iteration 1**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-.04</td>
<td>-.04</td>
<td>0.76</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>-.04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-.04</td>
<td>-.04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Iteration 2**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-.04</td>
<td>-.04</td>
<td>0.76</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>-.04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>-0.08</td>
<td>-0.04</td>
<td>-0.04</td>
</tr>
</tbody>
</table>

\[ V^{(2)}(1,1) = -0.04 + 1 \times \max \begin{bmatrix} 0.8V^{(1)}(1,2) + 0.1V^{(1)}(2,1) + 0.1V^{(1)}(1,1) & UP \\ 0.9V^{(1)}(1,1) + 0.1V^{(1)}(1,2) & LEFT \\ 0.9V^{(1)}(1,1) + 0.1V^{(1)}(2,1) & DOWN \\ 0.8V^{(1)}(2,1) + 0.1V^{(1)}(1,2) + 0.1V^{(1)}(1,1) & RIGHT \end{bmatrix} \]

\[ V^{(2)}(1,1) = -0.04 + \max \begin{bmatrix} -.04 & UP \\ -.04 & LEFT \\ -.04 & DOWN \\ -.04 & RIGHT \end{bmatrix} = -0.08 \]
Example (cont’d)

Let’s compute \( V^{(2)}(2,3) \)

\[
V^{(2)}(2,3) = -0.04 + 1 \times \max \begin{bmatrix}
0.8V^{(1)}(2,3) + 0.1V^{(1)}(1,3) + 0.1V^{(1)}(3,3) & \text{UP} \\
0.8V^{(1)}(1,3) + 0.1V^{(0)}(2,3) + 0.1V^{(1)}(2,3) & \text{LEFT} \\
0.8V^{(1)}(2,3) + 0.1V^{(1)}(1,3) + 0.1V^{(1)}(3,3) & \text{DOWN} \\
0.8V^{(1)}(3,3) + 0.1V^{(1)}(2,3) + 0.1V^{(1)}(2,3) & \text{RIGHT} 
\end{bmatrix}
\]

\[
V^{(1)}(2,3) = -0.04 + (0.8 \times 0.76 + 0.2 \times -0.04) = 0.56
\]

Steps two moves away from positive rewards start increasing their value.
Note that values of states at different distances from (4,3) accumulate negative rewards until a path to (4,3) is found.
Value Iteration: Computational Complexity

Value iteration works by producing successive approximations of the optimal value function.

\[ \forall s : V^{(k+1)}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)V^{(k)}(s') \]

What is the complexity of each iteration?

A. \( O(|A|^2|S|) \)  
B. \( O(|A||S|^2) \)  
C. \( O(|A|^2|S|^2) \)

…or faster if there is sparsity in the transition function.
Relevance to state of the art MDPs

Value Iteration (VI) forms the basis of most of the advanced MDP algorithms that we discuss in the rest of the book. ........
Lecture Overview

Markov Decision Processes

• ……

• Finding the Optimal Policy
  • Value Iteration

• From Values to the Policy

• Rewards and Optimal Policy
Value Iteration: from state values $V$ to $\pi^*$

Now the agent can choose the action that implements the **MEU principle**: maximize the expected utility of the subsequent state.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.812</td>
<td>0.868</td>
<td>0.912</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>0.762</td>
<td>0.660</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.705</td>
<td>0.655</td>
<td>0.611</td>
<td>0.388</td>
</tr>
</tbody>
</table>
Value Iteration: from state values $V$ to $\pi^*$

- Now the agent can choose the action that implements the MEU principle: maximize the expected utility of the subsequent state

$$\pi^*(s) = \arg \max_a \sum_{s'} P(s'| s, a)V_{\pi^*}(s')$$

- Expected value of following policy $\pi^*$ in $s'$
- States reachable from $s$ by doing $a$
- Probability of getting to $s'$ from $s$ via $a$
Example: from state values $V$ to $\pi^*$

$$\pi^*(s) = \arg \max_a \sum_{s'} P(s'|s, a) V^\pi^*(s')$$

➢ To find the best action in (1,1)

$$\pi^*(1,1) = \arg \max \begin{bmatrix}
0.8V(1,2) + 0.1V(2,1) + 0.1V(1,1) \\
0.9V(1,1) + 0.1V(1,2) \\
0.9V(1,1) + 0.1V(2,1) \\
0.8V(2,1) + 0.1V(1,2) + 0.1V(1,1)
\end{bmatrix}$$
Optimal policy

➢ This is the policy that we obtain....
Learning Goals for today’s class

You can:

Define/read/write/trace/debug the Value Iteration (VI) algorithm. Compute its complexity.

• Compute the Optimal Policy given the output of VI

• Explain influence of rewards on optimal policy
• **Read Textbook** 9.5.6 Partially Observable MDPs

• **Also Do Practice Ex. 9.C**

http://www.aispace.org/exercises.shtml