Foundations of Agents: Practical Assignment 1

Kevin Trebing (i6192338)

September 26, 2018

1 Tower of Hanoi problem

In a Tower of Hanoi problem the agent needs to move disks of different sizes from one pin to another pin. Furthermore, the disks need to be in the order that a smaller disk needs to be on top of a bigger one. Only one disk can be moved at a time and only the topmost disk can be moved. In our problem we have 3 pins and two disks. The starting position is pin 1 and the smaller disk is on the bigger disk. The goal is to move the disks to pin 3.

1.1 Description of States

Notation:

- a = small disk,
- b = big disk,
- 1 = pin1,
- 2 = pin2,
- 3 = pin3,
- ab = a is on b

We have 12 different possible states:

State	Pin		
s_0	ab1	2	3
s_1	1	ab2	3
s_2	1	2	ab3
s_3	ba1	2	3
s_4	1	ba2	3
s_5	1	2	ba3
s_6	b1	a2	3
s_7	a1	b2	3
s_8	b1	2	a3
s_9	a1	2	b3
s_{10}	1	a2	b3
s_{11}	1	b2	a3

1.2 Description of Actions

We have 6 different actions that the agent can take.

Action	effect
a_1	move a to pin1
a_2	move a to pin2
a_3	move a to pin3
b_1	move b to pin1
b_2	move b to pin2
b_3	move b to pin3

2 Optimal policy

The optimal policy describes for every state the best action the agent should take.

- $\pi(s_0) = a_2$
- $\pi(s_1) = a_1$
- $\pi(s_2) = a_2$
- $\pi(s_3) = b_3$
- $\pi(s_4) = b_3$
- $\pi(s_5) = b_1$ according to PI and b_2 according to VI
- $\pi(s_6) = b_3$
- $\pi(s_7) = b_3$
- $\pi(s_8) = a_2$
- $\pi(s_9) = a_3$
- $\pi(s_{10}) = a_3$
- $\pi(s_{11}) = a_1$

The difference in $\pi(s_5)$ arises because of different utilities of the states. Using action b_1 would transition into state s_8 and using action b_2 would transition into state s_{11}. Below we can see that policy iteration gives the state s_8 a higher utility than s_{11}, vice versa for value iteration.

3 Utility

The utility of each state given the optimal policy. The utility is calculated the following formula:

$$u(s, \pi) = r(s, \pi(s)) + \gamma \cdot \sum_{s' \epsilon S} t(s, \pi(s), s') \cdot u(s', \pi) \tag{1}$$

Using the values of the policy-iteration:

- $u(s_0) = 73.59$
- $u(s_1) = 62.27$
- $u(s_2) = 0.00$
- $u(s_3) = 86.00$
- $u(s_4) = 86.63$
- $u(s_5) = 53.27$
- $u(s_6) = 85.91$
- $u(s_7) = 85.81$
- $u(s_8) = 74.31$
- $u(s_9) = 98.79$
- $u(s_{10}) = 98.79$
- $u(s_{11}) = 74.11$

Using value-iteration the following utility-values arise:

- $u(s_0) = 75.31$
- $u(s_1) = 75.39$
- $u(s_2) = 0.00$
- $u(s_3) = 86.75$
- $u(s_4) = 86.75$
- $u(s_5) = 66.77$
- $u(s_6) = 85.93$
- $u(s_7) = 85.93$
- $u(s_8) = 74.48$
- $u(s_9) = 98.79$
- $u(s_{10}) = 98.79$
- $u(s_{11}) = 75.39$

4 Conclusion

The policies for value-iteration and policy-iteration differed in one state (s_5). But apart from that they were similar. Interestingly, the policy-iteration was significantly faster than value-iteration: value-iteration needed about 0.016s to converge whereas policy-iteration only needed 0.0036s to converge. Furthermore, policy-iteration delivers a more accurate result.

5 Note

The Hanoi.py file requires Python 3.6.