Planning, Learning and Decision Making: Homework 3. Partially observable Markov decision problems

Mykhaylo Marfeychuk, Roy De Prins

06 November 2018

1 Excercise 1) a

State Space = {A, B1, B2, C, D, E, F}

Action Space = {a, b, c}

Observation Space = {A, B, C, D, E, F}

2 Excercise 1) b

Probability matrix action 'a':

	A	B1	B2	C	D	E	F
A	0.0	0.5	0.5	0.0	0.0	0.0	0.0
B1	0.0	0.0	0.0	0.0	0.0	1.0	0.0
B2	0.0	0.0	0.0	0.8	0.0	0.0	1.0
C	0.0	1.0	0.0	0.0	0.0	0.0	0.0
D	0.0	0.0	1.0	0.0	0.0	0.0	0.0
E	1.0	0.0	0.0	0.0	0.0	0.0	0.0
F	1.0	0.0	0.0	0.0	0.0	0.0	0.0

Probability matrix action 'b':

	A	B1	B2	C	D	E	F
A	0	0.5	0.5	0	0	0	0
B1	0	0	0	0	0	0	1
B2	0	0	0	0	0	1	0
C	0	1	0	0	0	0	0
D	0	0	1	0	0	0	0
E	1	0	0	0	0	0	0
F	1	0	0	0	0	0	0

Probability matrix action 'c':

	A	B1	B2	C	D	E	F
A	0	0.5	0.5	0	0	0	0
B1	0	0	0	1	0	0	0
B2	0	0	0	0	1	0	0
C	0	1	0	0	0	0	0
D	0	0	1	0	0	0	0
E	1	0	0	0	0	0	0
F	1	0	0	0	0	0	0

Observation Matrix for 'a', 'b' and 'c':

	A	B	C	D	E	F
A	1	0	0	0	0	0
B1	0	1	0	0	0	0
B2	0	1	0	0	0	0
C	0	0	1	0	0	0
D	0	0	0	1	0	0
E	0	0	0	0	1	0
F	0	0	0	0	0	1

Cost Function:

	a	b	c
A	1	1	1
B1	1	1	1
B2	1	1	1
C	1	1	1
D	1	1	1
E	1	1	1
F	0	0	0

3 Excercise 1) c

We calculated the Belief at time step t+1 with Python code, which gave the following output:

Belief at t+1 for action 'a': [0. 0. 0. 0. 0. 0.5 0.5]
Belief at t+1 for action 'b': [0. 0. 0. 0. 0. 0.5 0.5]
Belief at t+1 for action 'c': [0. 0. 0. 0.5 0.5 0. 0.]

```python
import numpy as np
pA = np.array([[0,0.5,0.5,0,0,0,0],
               [0,0,0,0,0,1,0],
               [0,0,0,0,0,0,1],
               [0,1,0,0,0,0,0],
               [0,0,1,0,0,0,0],
               [1,0,0,0,0,0,0],
               [1,0,0,0,0,0,0]])

pB = np.array([[0,0.5,0.5,0,0,0,0],
               [0,0,0,0,0,0,1],
               [0,0,0,0,0,1,0],
               [0,1,0,0,0,0,0],
               [0,0,1,0,0,0,0],
               [1,0,0,0,0,0,0],
               [1,0,0,0,0,0,0]])

pC = np.array([[0,0.5,0.5,0,0,0,0],
               [0,0,0,1,0,0,0],
               [0,0,0,0,1,0,0],
               [0,1,0,0,0,0,0],
               [0,0,1,0,0,0,0],
               [1,0,0,0,0,0,0],
               [1,0,0,0,0,0,0]])

observation_matrix = np.array([ [1,0,0,0,0,0],
                                [0,1,0,0,0,0],
                                [0,1,0,0,0,0],
                                [0,0,1,0,0,0],
                                [0,0,0,1,0,0],
                                [0,0,0,0,1,0],
                                [0,0,0,0,0,1]])

belief = np.array([0.0,0.5,0.5,0.0,0.0,0.0,0.0])

obsA = np.identity(7)
obsB = obsA
obsC = obsA

def belief_update(belief, probability_action, observation):
    sum = 0
    t1 = np.dot(belief, probability_action)
    t2 = observation
    numerator= np.dot(t1, t2)
    for x in np.nditer(numerator):
        sum += x
    return numerator/sum

print("Belief at t+1 for action 'a':",belief_update(belief, pA, obsA))
print("Belief at t+1 for action 'b':",belief_update(belief, pB, obsB))
print("Belief at t+1 for action 'c':",belief_update(belief, pC, obsC))
```

Figure 1: The Python program for calculating the updated belief.