Coursework 1

Jianhong Wang CID: 01235001

November 3, 2016

Anouncement: every transition matrix and reward matrix in this answer sheet is row notation, which means row represents current state and column represents successor state.

1 Question 1

1. The transition matrix for $\pi(s) = Left$ is:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	1	0	0	0	0	0	0
s_2	0.8	0.2	0	0	0	0	0
s_3	0	0.8	0.2	0	0	0	0
s_4	0	0	0.8	0.2	0	0	0
s_5	0	0	0	0.8	0.2	0	0
s_6	0	0	0	0	0.8	0.2	0
s_7	0	0	0	0	0	0	1

The transition matrix for $\pi(s) = Right$ is:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	1	0	0	0	0	0	0
s_2	0	0.2	0.8	0	0	0	0
s_3	0	0	0.2	0.8	0	0	0
s_4	0	0	0	0.2	0.8	0	0
s_5	0	0	0	0	0.2	0.8	0
s_6	0	0	0	0	0	0.2	0.8
s_7	0	0	0	0	0	0	1

The matrix for reward function of $\pi(s) = Left$ is:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	0	0	0	0	0	0	0
s_2	-10	-10	0	0	0	0	0
s_3	0	1	1	0	0	0	0
s_4	0	0	1	1	0	0	0
s_5	0	0	0	1	1	0	0
s_6	0	0	0	0	1	1	0
s_7	0	0	0	0	0	0	0

The matrix for reward function of $\pi(s) = Right$ is:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	0	0	0	0	0	0	0
s_2	0	-1	-1	0	0	0	0
s_3	0	0	-1	-1	0	0	0
s_4	0	0	0	-1	-1	0	0
s_5	0	0	0	0	-1	-1	0
s_6	0	0	0	0	0	10	10
s_7	0	0	0	0	0	0	0

The terminates of all of matrices above are s_1 and s_7.

2.

$$V(s) = \Sigma_{s' \in S} P_{ss'} r_{ss'} + \gamma \Sigma_{s' \in S} P_{ss'} v(s') \quad (1)$$

	$V(s_1)$	$V(s_2)$	$V(s_3)$	$V(s_4)$	$V(s_5)$	$V(s_6)$	$V(s_7)$
initial	0	0	0	0	0	0	0
iteration 1	0	-10	1	1	1	1	0
iteration 2	0	-10.5	-0.95	1.25	1.25	1.25	0
iteration 3	0	-10.525	-1.1475	0.8725	1.3125	1.3125	0

3. The deriviation shows below:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
initial	0	0	0	0	0	0	
iteration 1	0	-10	1	1	1	1	0
iteration 2	0	-10.5	-0.95	1.25	1.25	1.25	0
iteration 3	0	-10.525	-1.1475	0.8725	1.3125	1.3125	0
*iteration*4	0	-10.5263	-1.1624	0.8141	1.2401	1.3281	0
*iteration*5	0	-10.5263	-1.1634	0.8082	1.2248	1.3144	0
*iteration*6	0	-10.5263	-1.1634	0.8077	1.2229	1.3107	0
*iteration*7	0	-10.5263	-1.1634	0.8077	1.2227	1.3101	0

Here γ is the same value as Question 2.
Meanwhile, assume that we always choose left action just as the question 1 subpart 2.

4. The full policy is:

when $p < 0.12$,
$\pi(Right \mid s_1) = 0$ and $\pi(Left \mid s_1) = 1$
$\pi(Right \mid s_2) = 1$ and $\pi(Left \mid s_2) = 0$
$\pi(Right \mid s_3) = 0$ and $\pi(Left \mid s_3) = 1$
$\pi(Right \mid s_4) = 0$ and $\pi(Left \mid s_4) = 1$
$\pi(Right \mid s_5) = 1$ and $\pi(Left \mid s_5) = 0$
$\pi(Right \mid s_6) = 1$ and $\pi(Left \mid s_6) = 0$
$\pi(Right \mid s_7) = 0$ and $\pi(Left \mid s_7) = 1$

when $p \geq 0.12$,
$\pi(Right \mid s_1) = 0$ and $\pi(Left \mid s_1) = 1$

$\pi(Right \mid s_2) = 1$ and $\pi(Left \mid s_2) = 0$
$\pi(Right \mid s_3) = 0$ and $\pi(Left \mid s_3) = 1$
$\pi(Right \mid s_4) = 0$ and $\pi(Left \mid s_4) = 1$
$\pi(Right \mid s_5) = 0$ and $\pi(Left \mid s_5) = 1$
$\pi(Right \mid s_6) = 1$ and $\pi(Left \mid s_6) = 0$
$\pi(Right \mid s_7) = 0$ and $\pi(Left \mid s_7) = 1$

$\pi(a \mid s_5)$ will change from $\pi(Right \mid s_5) = 1$ and $\pi(Left \mid s_5) = 0$ to $\pi(Right \mid s_5) = 0$ and $\pi(Left \mid s_5) = 1$ when p is greater equal than 0.12.
Since the state values of terminals are always zero, either action for these two states will be the optimal policy. Here we choose $\pi(Left) = 1$ to be the optimal policy for terminal states.
Here γ is the same value as Question 2.

2 Question 2

1. This question requests to list a return function and resolve it:

$$R(T) = r_{T+1} + \gamma r_{T+2} + ... + \gamma^k r_{T+k+1} \tag{2}$$
$$= 1 + 0.8 + ... + 0.8^k \tag{3}$$
$$= \frac{(1 - 0.8^{k+1})}{1 - 0.8} \tag{4}$$
$$\tag{5}$$

Since this is an infinite sequence, so k tends to ∞:

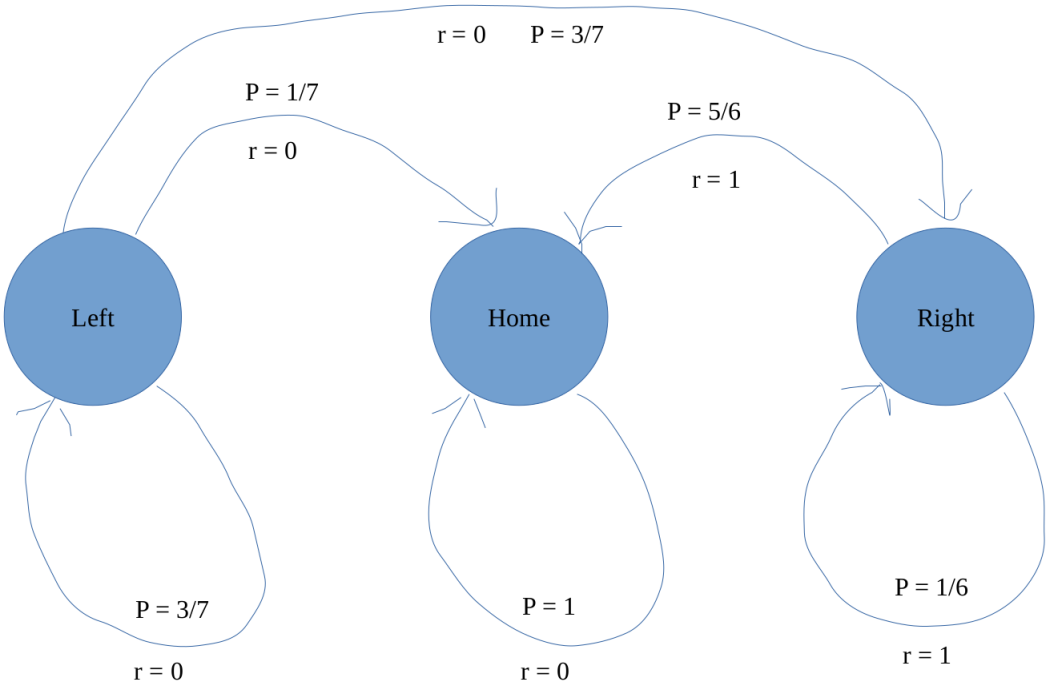
$$R(T) = lim_{k \to \infty} \frac{(1 - 0.8^{k+1})}{1 - 0.8} = \frac{1}{0.2} = 5 \tag{6}$$

2. (a) The transition matrix should be:

	$Left$	$Right$	$Home$
$Left$	$\frac{3}{7}$	$\frac{3}{7}$	$\frac{1}{7}$
$Right$	0	$\frac{1}{6}$	$\frac{5}{6}$
$Home$	0	0	1

The reward function matrix should be:

	$Left$	$Right$	$Home$
$Left$	0	0	0
$Right$	0	1	1
$Home$	0	0	0



To infer the transition matrix, the first step is to count up the frequency from each state to another state, then computing the probability. To infer the reward function matrix, the thing is to find the immediate reward value appears in the sample for each state to another state.

(b) After observation, it can be found that:

$$V(s = Right) = \frac{1}{6}\gamma V(s^{'} = Right) + 1 \tag{7}$$

$$V(s = Left) = \frac{3}{7}V(s^{'} = Left) + \frac{3}{7}V(s^{'} = Right)\gamma \tag{8}$$

$$V(s = Home) = 0 \tag{9}$$

Assume that the state value can be converged, so:

$$V(s = Right) = \frac{1}{6}\gamma V(s = Right) + 1 \tag{10}$$

$$= \frac{6}{6 - \gamma} \tag{11}$$

Since all of state values should converge together finally, so:

$$V(s^* = Left) = \frac{3}{7}V(s^* = Left) + \frac{3}{7}V(s^* = Right)\gamma \tag{12}$$

$$= \frac{16\gamma}{7(6 - \gamma)(1 - \frac{3\gamma}{7})} \tag{13}$$

3 Appendices

1. This is the code for solving out the Question 1 subpart 3:

```
import numpy as np
from numpy import linalg as li

def state_value_left(v):

    #The transition matrix of action Left
    P = np.zeros((7,7))
    for i in range(0, 7):
        if i == 0 or i == 6:
            P[i][i] = 1
        else:
```

```
            P[i][i] = 0.2
            P[i][i-1] = 0.8

    #The transition matrix of reward function of action Left
    r = np.zeros((7,7))
    for i in range(0, 7):
        if i == 0 or i == 6:
            pass
        elif i == 1:
            r[i][i] = -10
            r[i][i-1] = -10
        else:
            r[i][i] = 1
            r[i][i-1] = 1

    #compute \Sigma_{s' \in S} P_{ss'}r_{ss'}
    A = np.dot(P, r.transpose())
    part_1 = np.array([A[0][0], A[1][1], A[2][2], A[3][3],
    A[4][4], A[5][5], A[6][6]])

    #compute \gamma \Sigma_{s' \in S} P_{ss'} v(s')
    part_2 = 0.25 * np.dot(P, v)

    #combine two partitions together
    v = part_1.reshape((7,1)) + part_2

    return v

    if __name__ == "__main__":

        #initialise the state value vector
        v_l = np.zeros((7, 1))
        v = np.zeros((7, 1))


        i = 0
        #set the threshold to terminate the loop
        theta = 0.0001
```

```
temp = np.zeros((7, 1))
while True:
    temp = v_l
    #print "iteration ", i, ":\n", v_l.reshape((1,7))[0]
    #compute a state value
    v_l = state_value_left(v_l)
    #compare with the previous value
    delta = temp - v_l
    delta = [abs(x) for x in delta]
    if np.amax(delta) < theta:
        break
    i+=1

print "This is the vector of Left: ", v_l.reshape((1, 7))[0
```

2. This is the code for solving out the Question 1 subpart 4:

```
import numpy as np
from numpy import linalg as li

def state_value_left(v, p):

    #The transition matrix of action Left
    P = np.zeros((7,7))
    for i in range(0, 7):
        if i == 0 or i == 6:
            P[i][i] = 1
        else:
            P[i][i] = p
            P[i][i-1] = 1 - p

    #The transition matrix of reward function of action Left
    r = np.zeros((7,7))
    for i in range(0, 7):
        if i == 0 or i == 6:
            pass
        elif i == 1:
            r[i][i] = -10
            r[i][i-1] = -10
        else:
            r[i][i] = 1
            r[i][i-1] = 1

    #compute \Sigma_{s' \in S} P_{ss'}r_{ss'}
    A = np.dot(P, r.transpose())
    part_1 = np.array([A[0][0], A[1][1], A[2][2], A[3][3],
    A[4][4], A[5][5], A[6][6]])

    #compute \gamma \Sigma_{s' \in S} P_{ss'} v(s')
    part_2 = 0.25 * np.dot(P, v)

    #combine two partitions together
```

```
    v = part_1.reshape((7,1)) + part_2

    return v

def state_value_right(v, p):

    #The transition matrix of action Right
    P = np.zeros((7,7))
    for i in range(0, 7):
        if i == 0 or i == 6:
            P[i][i] = 1
        else:
            P[i][i] = p
            P[i][i+1] = 1 - p

    #The transition matrix of reward function of action Right
    r = np.zeros((7,7))
    for i in range(0, 7):
        if i == 0 or i == 6:
            pass
        elif i == 5:
            r[i][i] = 10
            r[i][i+1] = 10
        else:
            r[i][i] = -1
            r[i][i+1] = -1

    #compute \Sigma_{s' \in S} P_{ss'}r_{ss'}
    A = np.dot(P, r.transpose())
    part_1 = np.array([A[0][0], A[1][1], A[2][2], A[3][3],
    A[4][4], A[5][5], A[6][6]])

    #compute \gamma \Sigma_{s' \in S} P_{ss'} v(s')
    part_2 = 0.25 * np.dot(P, v)

    #combine two partitions together
    v = part_1.reshape((7,1)) + part_2
```

```
    return v

if __name__ == "__main__":

    #initialise the state value vector
    v_l = np.zeros((7, 1))
    v = np.zeros((7, 1))

    p = np.arange(0, 1, 0.01)

    for j in range(0, 100):
        state = np.zeros((1,7))
        c = 0
        #set the threshold to terminate the loop
        theta = 0.0001
        temp = np.zeros((7, 1))
        while True:
            v_ = np.zeros((7,1))
            temp = v
            #print "iteration ", c, ":\n", v.reshape((1,7))[0]
            #compute a state value
            v_r = state_value_right(v, p[j])
            v_l = state_value_left(v, p[j])
            #compare state value
            for i in range(0, 7):
                if v_r[i][0] > v_l[i][0]:
                    state[0][i] = -1
                    v_[i][0] = v_r[i][0]
                else:
                    state[0][i] = 1
                    v_[i][0] = v_l[i][0]

            v = v_
            #compare with the previous value
            delta = temp - v
            delta = [abs(x) for x in delta]
            if np.amax(delta) < theta:
                break
```

```
        c+=1

    print "The value of p is: ", p[j]
    print "This is the action vector: ", state
```