Lecture lnotes2 Mars Rover(MDP)

Bong Seok Kim

2021-01-28

차 례

Example of a Markov decision process : Mars Rover

As an example of a MDP, consider the example given in Figure 3. The agent is again a Mars rover whose state space is given by S = {S1, S2, S3, S4, S5, S6, S7}. The agent has two actions in each state called 'try left' and 'try right', and so the action space is given by A = {TL, TR}. Taking an action always succeeds, unless we hit an edge in which case we stay in the same state. This leads to the two transition probability matrices for each of the two actions as shown in Figure 3. The rewards from each state are the same for all actions, and is 0 in the states {S2, S3, S4, S5, S6}, while for the states S1, S7 the rewards are 1, 10 respectively. The discount factor for this MDP is some $\gamma \in$ [0, 1].

Preparation

State space = {S1,S2,S3,S4,S5,S6,S7}

action = {try left, try right} ={TL,TR} action always succeeds unless hit an edge

Reward = S1, S7 1, 10 respectively else 0

Transition Probabilty ,$P^a_{ss'}$ in Figure 3

Pi S->A

```
import numpy as np
import pandas as pd
```

```
states = ['S1','S2','S3','S4','S5','S6','S7']

pi_TL = np.c_[np.repeat(1,len(states)), np.repeat(0,len(states))]
pi_TL = pd.DataFrame(pi_TL, columns=['TL','TR'],index= states)

pi_TR = np.c_[np.repeat(0,len(states)), np.repeat(1,len(states))]
pi_TR = pd.DataFrame(pi_TR, columns=['TL','TR'],index= states)

pi_50 = np.c_[np.repeat(0.5,len(states)), np.repeat(0.5,len(states))]
pi_50 = pd.DataFrame(pi_50, columns=['TL','TR'],index= states)



pi_TL
```

```
##     TL  TR
## S1   1   0
## S2   1   0
## S3   1   0
## S4   1   0
## S5   1   0
## S6   1   0
## S7   1   0
```

P^pi SXA -> S'

```
P_TL = np.matrix([
                  [1,0,0,0,0,0,0],
                  [1,0,0,0,0,0,0],
                  [0,1,0,0,0,0,0],
                  [0,0,1,0,0,0,0],
                  [0,0,0,1,0,0,0],
                  [0,0,0,0,1,0,0],
                  [0,0,0,0,0,1,0],
                 ])

P_TR = np.matrix([[0,1,0,0,0,0,0],
                  [0,0,1,0,0,0,0],
                  [0,0,0,1,0,0,0],
                  [0,0,0,0,1,0,0],
                  [0,0,0,0,0,1,0],
                  [0,0,0,0,0,0,1],
```

```
                    [0,0,0,0,0,0,1],

                   ])

P_TL
```

```
## matrix([[1, 0, 0, 0, 0, 0, 0],
##         [1, 0, 0, 0, 0, 0, 0],
##         [0, 1, 0, 0, 0, 0, 0],
##         [0, 0, 1, 0, 0, 0, 0],
##         [0, 0, 0, 1, 0, 0, 0],
##         [0, 0, 0, 0, 1, 0, 0],
##         [0, 0, 0, 0, 0, 1, 0]])
```

```
def transition(given_pi,states, P_TL, P_TR):
    P_out = np.zeros(shape=(7,7))

    for i in range(len(states)):
        action_dist = given_pi.iloc[i,:]

        P= action_dist['TL']*P_TL + action_dist['TR']*P_TR
        P_out[i,]=P[i,]

    return P_out

transition(pi_TL, states=states, P_TL=P_TL, P_TR=P_TR)
```

```
## array([[1., 0., 0., 0., 0., 0., 0.],
##        [1., 0., 0., 0., 0., 0., 0.],
##        [0., 1., 0., 0., 0., 0., 0.],
##        [0., 0., 1., 0., 0., 0., 0.],
##        [0., 0., 0., 1., 0., 0., 0.],
##        [0., 0., 0., 0., 1., 0., 0.],
##        [0., 0., 0., 0., 0., 1., 0.]])
```

R^pi S->R

```
R_s_a = pd.DataFrame(np.array([[1,0,0,0,0,0,10],
                                  [1,0,0,0,0,0,10]]).T,columns=['TL','TR'],index=states)

def reward_fn(given_pi):
```

```
    R_s_a = pd.DataFrame(np.array([[1,0,0,0,0,0,10],
                                  [1,0,0,0,0,0,10]]).T,columns=['TL','TR'],index=states) # all action got sam
    R_pi = np.sum(R_s_a*given_pi, axis=1)

    return R_pi

reward_fn(pi_TR)
```

```
## S1     1
## S2     0
## S3     0
## S4     0
## S5     0
## S6     0
## S7    10
## dtype: int64
```

Policy evaluation

```
def policy_eval(given_pi,gamma=0.99):
    R = reward_fn(given_pi).values.reshape(7,1)
    P = transition(given_pi,states, P_TL = P_TL, P_TR = P_TR)

    gamma = gamma
    epsilon = 10**(-8)

    v_old=np.repeat(0,7).reshape(7,1)
    v_new = R+gamma*np.dot(P, v_old)

    while np.max(np.abs(v_new-v_old))>epsilon:
        v_old=v_new
        v_new=R+np.dot(gamma*P,v_old)


    return v_new
```

```
policy_eval(pi_TL, gamma=0.9).astype(int)
```

```
## array([[ 9],
##        [ 8],
##        [ 8],
```

```
##        [ 7],
##        [ 6],
##        [ 5],
##        [15]])
```

```
policy_eval(pi_TL, gamma=0).astype(int)
```

```
## array([[ 1],
##        [ 0],
##        [ 0],
##        [ 0],
##        [ 0],
##        [ 0],
##        [10]])
```

```
policy_eval(pi_TL, gamma=0.1).astype(int)
```

```
## array([[ 1],
##        [ 0],
##        [ 0],
##        [ 0],
##        [ 0],
##        [ 0],
##        [10]])
```

Policy Improvement

```
gamma=0.1
V_old = policy_eval(pi_TL)
pi_old = pi_TL
q_s_a = R_s_a + np.c_[gamma*np.dot(P_TL,V_old), gamma*np.dot(P_TR,V_old)]
q_s_a
```

```
##           TL         TR
## S1  11.00000  10.900000
## S2  10.00000   9.801000
## S3   9.90000   9.702990
## S4   9.80100   9.605960
## S5   9.70299   9.509900
## S6   9.60596  10.414801
## S7  19.50990  20.414801
```

```
pi_new_vec=q_s_a.idxmax(axis=1)
pi_new_vec
```

```
## S1    TL
## S2    TL
## S3    TL
## S4    TL
## S5    TL
## S6    TR
## S7    TR
## dtype: object
```

```
pi_new = pd.DataFrame(np.zeros(shape=(pi_old.shape)),columns=['TL','TR'])


for i in range(len(pi_new_vec)):
    pi_new.iloc[i][pi_new_vec[i]]=1

pi_new
```

```
##     TL   TR
## 0  1.0  0.0
## 1  1.0  0.0
## 2  1.0  0.0
## 3  1.0  0.0
## 4  1.0  0.0
## 5  0.0  1.0
## 6  0.0  1.0
```

```
def policy_imporve(V_old, pi_old, R_s_a=R_s_a, gamma = gamma, P_TL = P_TL, P_TR = P_TR):

    q_s_a=R_s_a + np.c_[np.dot(P_TL,V_old),np.dot(P_TR,V_old)]
    pi_new_vec=q_s_a.idxmax(axis=1)
    pi_new = pd.DataFrame(np.zeros(shape=(pi_old.shape)),columns=['TL','TR'],index=states)

    for i in range(len(pi_new_vec)):
        pi_new.iloc[i][pi_new_vec[i]]=1

    return pi_new
```

Policy iteration

```
pi_old = pi_TL
gamma =0.99

cnt = 0

while True :
    print("-------------------")
    print(cnt,"-th iteration")
    print(pi_old)
    V_old = policy_eval(pi_old)
    pi_new  = policy_imporve(V_old, pi_old, R_s_a=R_s_a, gamma = gamma, P_TL = P_TL, P_TR = P_TR)

    if(np.sum((pi_old==pi_new).values) != pi_new.shape[0]*pi_new.shape[1]):
        cnt+=1
        pi_old=pi_new
        continue
    break
```

```
## -------------------
## 0 -th iteration
##      TL  TR
## S1    1   0
## S2    1   0
## S3    1   0
## S4    1   0
## S5    1   0
## S6    1   0
## S7    1   0
## -------------------
## 1 -th iteration
##       TL   TR
## S1   1.0  0.0
## S2   1.0  0.0
## S3   1.0  0.0
## S4   1.0  0.0
## S5   1.0  0.0
## S6   0.0  1.0
## S7   0.0  1.0
## -------------------
```

```
## 2 -th iteration
##      TL   TR
## S1  1.0  0.0
## S2  1.0  0.0
## S3  1.0  0.0
## S4  1.0  0.0
## S5  0.0  1.0
## S6  0.0  1.0
## S7  0.0  1.0
## -------------------
## 3 -th iteration
##      TL   TR
## S1  1.0  0.0
## S2  1.0  0.0
## S3  1.0  0.0
## S4  0.0  1.0
## S5  0.0  1.0
## S6  0.0  1.0
## S7  0.0  1.0
## -------------------
## 4 -th iteration
##      TL   TR
## S1  1.0  0.0
## S2  1.0  0.0
## S3  0.0  1.0
## S4  0.0  1.0
## S5  0.0  1.0
## S6  0.0  1.0
## S7  0.0  1.0
## -------------------
## 5 -th iteration
##      TL   TR
## S1  1.0  0.0
## S2  0.0  1.0
## S3  0.0  1.0
## S4  0.0  1.0
## S5  0.0  1.0
## S6  0.0  1.0
## S7  0.0  1.0
## -------------------
## 6 -th iteration
##      TL   TR
```

```
## S1  0.0  1.0
## S2  0.0  1.0
## S3  0.0  1.0
## S4  0.0  1.0
## S5  0.0  1.0
## S6  0.0  1.0
## S7  0.0  1.0
```

```
print("-------------------")
```

```
## -------------------
```

```
print(policy_eval(pi_new,0.1).astype(int))
```

```
## [[ 1]
##  [ 0]
##  [ 0]
##  [ 0]
##  [ 0]
##  [ 1]
##  [11]]
```

Value imporvement

```
cnt=0
gamma=0.99
epsilon=10**(-6)
V_old=pd.DataFrame(np.repeat(0,len(states)).reshape(len(states),1),index=states)
results=V_old.T
while True:
    q_s_a=R_s_a + np.c_[gamma*np.dot(P_TL,V_old),gamma*np.dot(P_TR,V_old)]
    V_new=np.matrix(q_s_a.apply(max,axis=1)).reshape(len(states),1)

    if np.max(np.abs(V_new-V_old)).item() < epsilon :
        break

    results=np.r_[results, V_new.T]
    V_old=V_new



    cnt+=1
```

```
value_iter_process = results

results = pd.DataFrame(results, columns=states)

results
```

```
##                S1          S2          S3  ...          S5          S6          S7
## 0        0.000000    0.000000    0.000000  ...    0.000000    0.000000    0.000000
## 1        1.000000    0.000000    0.000000  ...    0.000000    0.000000   10.000000
## 2        1.990000    0.990000    0.000000  ...    0.000000    9.900000   19.900000
## 3        2.970100    1.970100    0.980100  ...    9.801000   19.701000   29.701000
## 4        3.940399    2.940399    1.950399  ...   19.503990   29.403990   39.403990
## ...           ...         ...         ...  ...         ...         ...         ...
## 1600   942.480046  950.989946  960.595906  ...  980.099896  989.999896  999.999896
## 1601   942.480047  950.989947  960.595907  ...  980.099897  989.999897  999.999897
## 1602   942.480048  950.989948  960.595908  ...  980.099898  989.999898  999.999898
## 1603   942.480049  950.989949  960.595909  ...  980.099899  989.999899  999.999899
## 1604   942.480050  950.989950  960.595910  ...  980.099900  989.999900  999.999900
##
## [1605 rows x 7 columns]
```

```
V_opt=value_iter_process[-1]
q_s_a=R_s_a+np.c_[np.dot(gamma*P_TL,V_opt.T),np.dot(gamma*P_TR,V_opt.T)]
q_s_a
```

```
##             TL          TR
## S1  934.055249  942.480051
## S2  933.055249  950.989951
## S3  941.480051  960.595911
## S4  950.989951  970.298901
## S5  960.595911  980.099901
## S6  970.298901  989.999901
## S7  990.099901  999.999901
```

```
pi_opt_vec=q_s_a.idxmax(axis=1)

pi_opt = pd.DataFrame(np.zeros((len(states),2)), index=states, columns=['TL','TR'])

for i in states:
    pi_opt.loc[i][[pi_opt_vec][0][i]]=1
```

```r
pi_opt
```

```
##       TL   TR
## S1  0.0  1.0
## S2  0.0  1.0
## S3  0.0  1.0
## S4  0.0  1.0
## S5  0.0  1.0
## S6  0.0  1.0
## S7  0.0  1.0
```

```r
"Done, Lecture E1.MDP with Model1 "
```

```
## [1] "Done, Lecture E1.MDP with Model1 "
```