Inotes2_MDP_Exercises

Kwon do yun

2021-02-01

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 $\gamma \in [0, 1]$.

$\pi : S \to A$

```
import numpy as np
import pandas as pd

states=["S1","S2","S3","S4","S5","S6","S7"]
action=["TL","TR"]

pi_TL=pd.DataFrame(np.c_[np.repeat(1,len(states)),np.repeat(0,len(states))], index=states, columns=action)
pi_TR=pd.DataFrame(np.c_[np.repeat(0,len(states)), np.repeat(1,len(states))],index=states, columns=action)
pi_50 =(pi_TL+pi_TR)/2

pi_50
```

```
##       TL   TR
## S1   0.5  0.5
## S2   0.5  0.5
## S3   0.5  0.5
## S4   0.5  0.5
## S5   0.5  0.5
## S6   0.5  0.5
## S7   0.5  0.5
```

$R^\pi : S \to \mathbb{R}$

```
R_s_a=pd.DataFrame(np.matrix([[1,1,0,0,0,0,0],[0,0,0,0,0,10,10]]).T,columns=action,index=states)

def reward_fn(given_pi):
    R_pi = np.sum(R_s_a*given_pi, axis=1)
    return R_pi.values.reshape([7,1])

reward_fn(pi_50)
```

```
## array([[0.5],
##        [0.5],
##        [0. ],
##        [0. ],
##        [0. ],
##        [5. ],
##        [5. ]])
```

$P^\pi : S \times A \rightarrow S$

```
P_TL = np.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],
                 ])



P_TR = np.array([[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_50 = (P_TL + P_TR)/2

def transition(given_pi,states, P_TL, P_TR):
    P_out = pd.DataFrame(np.zeros(shape=(len(states),len(states))))

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

        P= action_dist['TL']*P_TL + action_dist['TR']*P_TR
        P_out[s]=P[:,s]
    return P_out

transition(pi_50, states=states, P_TL=P_TL, P_TR=P_TR)
```

```
##      0    1    2    3    4    5    6
## 0  0.5  0.5  0.0  0.0  0.0  0.0  0.0
## 1  0.5  0.0  0.5  0.0  0.0  0.0  0.0
## 2  0.0  0.5  0.0  0.5  0.0  0.0  0.0
## 3  0.0  0.0  0.5  0.0  0.5  0.0  0.0
## 4  0.0  0.0  0.0  0.5  0.0  0.5  0.0
## 5  0.0  0.0  0.0  0.0  0.5  0.0  0.5
## 6  0.0  0.0  0.0  0.0  0.0  0.5  0.5
```

Policy evaluation

```
def policy_eval(given_pi,gamma,states,P_TL,P_TR):
    R=reward_fn(given_pi)
    P=transition(given_pi,states,P_TL,P_TR)
    gamma = gamma
    epsilon = 10**(-8)

    v_old=np.matrix([0,0,0,0,0,0,0]).T
    v_new = R + np.dot(gamma*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
```

```
gamma=0.99
print(policy_eval(pi_50,gamma,states,P_TL,P_TR))
```

```
## [[140.35814869]
##  [142.18356586]
##  [145.8712773 ]
##  [152.50588326]
##  [162.22141617]
##  [175.21414942]
##  [181.64555239]]
```

```
print(policy_eval(pi_TL,gamma,states,P_TL,P_TR))
```

```
## [[99.99999901]
##  [99.99999901]
##  [98.99999901]
##  [98.00999901]
##  [97.02989901]
##  [96.05960001]
##  [95.099004  ]]
```

```
print(policy_eval(pi_TR,gamma,states,P_TL,P_TR))
```

```
## [[950.99004891]
##  [960.59600901]
##  [970.29899901]
##  [980.09999901]
##  [989.99999901]
##  [999.99999901]
##  [999.99999901]]
```

```
gamma=0
print(policy_eval(pi_50,gamma,states,P_TL,P_TR))
```

```
## [[0.5]
##  [0.5]
##  [0. ]
##  [0. ]
##  [0. ]
##  [5. ]
##  [5. ]]
```

```
print(policy_eval(pi_TL,gamma,states,P_TL,P_TR))
```

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

```
print(policy_eval(pi_TR,gamma,states,P_TL,P_TR))
```

```
## [[ 0]
##  [ 0]
##  [ 0]
##  [ 0]
##  [ 0]
##  [10]
##  [10]]
```

Policy imporve

```
def policy_imporve(V_old,pi_old,gamma):
    q_s_a = R_s_a + np.c_[np.dot(gamma*P_TL,V_old),np.dot(gamma*P_TR,V_old)]
    pi_new=pd.DataFrame(np.zeros(pi_old.shape), index=pi_old.index, columns=pi_old.columns)
    idx = q_s_a.idxmax(axis=1).values
    count = 0
    for i in states:
        pi_new.loc[i][idx[count]] = 1
        count +=1
    return pi_new
```

```
gamma=0.99
pi_old = pi_TL
V_old = policy_eval(pi_TL,gamma,states,P_TL,P_TR)
q_s_a = R_s_a + np.c_[np.dot(gamma*P_TL,V_old),np.dot(gamma*P_TR,V_old)]
pi_new = policy_imporve(V_old,pi_old,gamma)
pi_new
```

```
##      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
```

```
gamma=0.99
pi_old = pi_TR
```

```
V_old = policy_eval(pi_TR,gamma,states,P_TL,P_TR)
q_s_a = R_s_a + np.c_[np.dot(gamma*P_TL,V_old),np.dot(gamma*P_TR,V_old)]
pi_new = policy_imporve(V_old,pi_old,gamma)
pi_new
```

```
##      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
```

```
gamma=0.99
pi_old = pi_50
V_old = policy_eval(pi_50,gamma,states,P_TL,P_TR)
q_s_a = R_s_a + np.c_[np.dot(gamma*P_TL,V_old),np.dot(gamma*P_TR,V_old)]
pi_new = policy_imporve(V_old,pi_old,gamma)
pi_new
```

```
##      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
```

```
gamma=0
pi_old = pi_TL
V_old = policy_eval(pi_TL,gamma,states,P_TL,P_TR)
q_s_a = R_s_a + np.c_[np.dot(gamma*P_TL,V_old),np.dot(gamma*P_TR,V_old)]
pi_new = policy_imporve(V_old,pi_old,gamma)
pi_new
```

```
##      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
```

```
gamma=0
pi_old = pi_TR
V_old = policy_eval(pi_TR,gamma,states,P_TL,P_TR)
q_s_a = R_s_a + np.c_[np.dot(gamma*P_TL,V_old),np.dot(gamma*P_TR,V_old)]
pi_new = policy_imporve(V_old,pi_old,gamma)
pi_new
```

```
##       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
```

```
gamma=0
pi_old = pi_50
V_old = policy_eval(pi_50,gamma,states,P_TL,P_TR)
q_s_a = R_s_a + np.c_[np.dot(gamma*P_TL,V_old),np.dot(gamma*P_TR,V_old)]
pi_new = policy_imporve(V_old,pi_old,gamma)
pi_new
```

```
##       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
```

```
"Inotes2_MDP_Exercises"
```