E2

Jeong, wonryeol

2021-01-22

Contents

4_page

```
import numpy as np
```

```
import pandas as pd
```

```
gamma = 1
states = np.arange(0,80,10).astype('str')
P_normal=pd.DataFrame(np.matrix([[0,1,0,0,0,0,0,0],
                                 [0,0,1,0,0,0,0,0],
                                 [0,0,0,1,0,0,0,0],
                                 [0,0,0,0,1,0,0,0],
                                 [0,0,0,0,0,1,0,0],
                                 [0,0,0,0,0,0,1,0],
                                 [0,0,0,0,0,0,0,1],
                                 [0,0,0,0,0,0,0,1]]), index=states,columns=states)
P_speed=pd.DataFrame(np.matrix([[.1,0,.9,0,0,0,0,0],
                                [.1,0,0,.9,0,0,0,0],
                                [0,.1,0,0,.9,0,0,0],
                                [0,0,.1,0,0,.9,0,0],
                                [0,0,0,.1,0,0,.9,0],
                                [0,0,0,0,.1,0,0,.9],
                                [0,0,0,0,0,.1,0,.9],
                                [0,0,0,0,0,0,0,1]]), index=states, columns=states)

def transition(given_pi, states, P_normal, P_speed):
    P_out=pd.DataFrame(np.zeros((len(states),len(states))),index=states, columns=states)

    for s in states:
        action_dist=given_pi.loc[s]
        P=action_dist['normal']*P_normal+action_dist['speed']*P_speed
        P_out.loc[s]=P.loc[s]

    return P_out
```

```
R_s_a=pd.DataFrame(np.array([-1,-1,-1,-1,0.0,-1,-1,0,
                              -1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]).reshape(len(states),2,order='F'),c
R_s_a
```

```
##     normal  speed
## 0     -1.0   -1.5
## 10    -1.0   -1.5
## 20    -1.0   -1.5
## 30    -1.0   -1.5
## 40     0.0   -0.5
## 50    -1.0   -1.5
## 60    -1.0   -1.5
## 70     0.0    0.0
```

```
def reward_fn(given_pi):
    R_s_a=pd.DataFrame(np.matrix([-1,-1,-1,-1,0.0,-1,-1,0,-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]).reshape

    R_pi=np.asarray((given_pi*R_s_a).sum(axis=1)).reshape(-1,1)

    return R_pi



def policy_eval(given_pi):
    R=reward_fn(given_pi)
    P=transition(given_pi, states=states, P_normal=P_normal, P_speed=P_speed)

    gamma=1.0
    epsilon=10**(-8)
    v_old=np.repeat(0,8).reshape(8,1)
    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
```

```
pi_speed=pd.DataFrame(np.c_[np.repeat(0,len(states)), np.repeat(1,len(states))],index=states, columns=[
policy_eval(pi_speed).T
```

```
## array([[-5.80592905, -5.2087811 , -4.13926239, -3.47576467, -2.35376031,
##         -1.73537603, -1.6735376 ,  0.        ]])
```

```
pi_50=pd.DataFrame(np.repeat(0.5,len(states)*2).reshape(8,2),index=states, columns=['normal','speed'])
policy_eval(pi_50).T
```

```
## array([[-5.96923786, -5.13359222, -4.11995525, -3.38922824, -2.04147003,
##         -2.02776769, -1.35138838,  0.        ]])
```

12__page Implementation

```
V_old = policy_eval(pi_speed)
pi_old = pi_speed
q_s_a = R_s_a + np.c_[np.dot(gamma*P_normal,V_old),np.dot(gamma*P_speed,V_old)]
q_s_a
```

```
##       normal     speed
## 0  -6.208781 -5.805929
## 10 -5.139262 -5.208781
## 20 -4.475765 -4.139262
## 30 -3.353760 -3.475765
## 40 -1.735376 -2.353760
## 50 -2.673538 -1.735376
## 60 -1.000000 -1.673538
## 70  0.000000  0.000000
```

```
idxmax = q_s_a.idxmax(axis=1).tolist()
count = 0
pi_new = pd.DataFrame(np.zeros(16).reshape(8,2),index = q_s_a.index,columns = q_s_a.columns)
for i in q_s_a.index.tolist():

    pi_new.loc[i][idxmax[count]] = 1
    count +=1

pi_new
```

```
##     normal  speed
## 0      0.0    1.0
## 10     1.0    0.0
## 20     0.0    1.0
## 30     1.0    0.0
## 40     1.0    0.0
## 50     0.0    1.0
## 60     1.0    0.0
## 70     1.0    0.0
```

policy_improve()

```
def policy_improve(v_old,pi_old,R_s_a,gamma,P_normal,P_speed):
    q_s_a=R_s_a+np.c_[np.dot(gamma*P_normal,V_old), np.dot(gamma*P_speed, V_old)]

    idxmax = q_s_a.idxmax(axis=1).tolist()
    count = 0
    pi_new = pd.DataFrame(np.zeros(16).reshape(8,2),index = q_s_a.index,columns = q_s_a.columns)
    for i in q_s_a.index.tolist():

        pi_new.loc[i][idxmax[count]] = 1
        count +=1

    return pi_new
```

One step improvement from π^{speed}

```
pi_old = pi_speed
V_old = policy_eval(pi_old)
pi_new = policy_improve(V_old,pi_old,R_s_a,gamma,P_normal,P_speed)
pi_old
```

```
##     normal  speed
## 0        0      1
## 10       0      1
## 20       0      1
## 30       0      1
## 40       0      1
## 50       0      1
## 60       0      1
## 70       0      1
```

```
pi_new
```

```
##     normal  speed
## 0      0.0    1.0
## 10     1.0    0.0
## 20     0.0    1.0
## 30     1.0    0.0
## 40     1.0    0.0
## 50     0.0    1.0
## 60     1.0    0.0
## 70     1.0    0.0
```

16_page

Step0

```
# step0
pi_old=pi_speed
pi_old
```

```
##    normal  speed
## 0       0      1
## 10      0      1
## 20      0      1
## 30      0      1
## 40      0      1
## 50      0      1
## 60      0      1
## 70      0      1
```

Step1

```
# step1
V_old=policy_eval(pi_old)
pi_new=policy_improve(V_old, pi_old=pi_old, R_s_a=R_s_a, gamma=gamma, P_normal=P_normal, P_speed=P_spee
pi_old=pi_new
pi_old
```

```
##    normal  speed
## 0     0.0    1.0
## 10    1.0    0.0
## 20    0.0    1.0
## 30    1.0    0.0
## 40    1.0    0.0
## 50    0.0    1.0
## 60    1.0    0.0
## 70    1.0    0.0
```

Step2

```
# step2
V_old=policy_eval(pi_old)
pi_new=policy_improve(V_old, pi_old=pi_old, R_s_a=R_s_a, gamma=gamma, P_normal=P_normal, P_speed=P_spee
pi_old=pi_new
pi_old
```

```
##    normal  speed
## 0     0.0    1.0
## 10    0.0    1.0
## 20    0.0    1.0
## 30    1.0    0.0
## 40    1.0    0.0
## 50    0.0    1.0
## 60    1.0    0.0
## 70    1.0    0.0
```

Step3

```
# step3
V_old=policy_eval(pi_old)
pi_new=policy_improve(V_old, pi_old=pi_old, R_s_a=R_s_a, gamma=gamma, P_normal=P_normal, P_speed=P_speed
pi_old=pi_new
pi_old
```

```
##     normal  speed
## 0      0.0    1.0
## 10     0.0    1.0
## 20     0.0    1.0
## 30     1.0    0.0
## 40     1.0    0.0
## 50     0.0    1.0
## 60     1.0    0.0
## 70     1.0    0.0
```

19_page

Policy iteration process π^{speed}

```
pi_old=pi_speed
count=0
while True:
    print(count,'-th iteration')
    print(pi_old.T)

    V_old=policy_eval(pi_old)

    pi_new=policy_improve(V_old, pi_old=pi_old, R_s_a=R_s_a, gamma=gamma, P_normal=P_normal, P_speed=P_s

    if pi_new.equals(pi_old)==True:
        break

    pi_old=pi_new
    count+=1
```

```
## 0 -th iteration
##         0   10   20   30   40   50   60   70
## normal  0   0   0   0   0   0   0   0
## speed   1   1   1   1   1   1   1   1
## 1 -th iteration
##           0    10    20    30    40    50    60    70
## normal  0.0  1.0  0.0  1.0  1.0  0.0  1.0  1.0
## speed   1.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0
## 2 -th iteration
##           0    10    20    30    40    50    60    70
## normal  0.0  0.0  0.0  1.0  1.0  0.0  1.0  1.0
## speed   1.0  1.0  1.0  0.0  0.0  1.0  0.0  0.0
```

```
policy_eval(pi_new).T
```

```
## array([[-5.1077441 , -4.41077441, -3.44107744, -2.66666667, -1.66666667,
##         -1.66666667, -1.        ,  0.        ]])
```

Policy iteration process π^{50}

```
pi_old=pi_50
count=0
while True:
    print(count,'-th iteration')
    print(pi_old.T)

    V_old=policy_eval(pi_old)

    pi_new=policy_improve(V_old, pi_old=pi_old, R_s_a=R_s_a, gamma=gamma, P_normal=P_normal, P_speed=P_s

    if pi_new.equals(pi_old)==True:
        break

    pi_old=pi_new
    count+=1
```

```
## 0 -th iteration
##           0   10   20   30   40   50   60   70
## normal  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
## speed   0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
## 1 -th iteration
##           0   10   20   30   40   50   60   70
## normal  0.0  1.0  0.0  1.0  1.0  0.0  1.0  1.0
## speed   1.0  0.0  1.0  0.0  0.0  1.0  0.0  0.0
## 2 -th iteration
##           0   10   20   30   40   50   60   70
## normal  0.0  0.0  0.0  1.0  1.0  0.0  1.0  1.0
## speed   1.0  1.0  1.0  0.0  0.0  1.0  0.0  0.0
```

```
policy_eval(pi_new).T
```

```
## array([[-5.1077441 , -4.41077441, -3.44107744, -2.66666667, -1.66666667,
##         -1.66666667, -1.        ,  0.        ]])
```