MDP Case (D case)

Kang, Eui Hyeon

2021-02-15

차 례

Fat Samuel 2
Preparation 3
Functions 5
Policy Evaluation 6
Policy Improvement 7

Fat Samuel

Samuel is 99 kilograms. He loves eating, but he knows gaining more weight is not good for his health. Now, He decided to lose weight for his lovely family. The goal is to lose up to 90 kilograms. There are two ways for losing weight.

1. Controlling Diet : It helps lose 1kg a week.

2. Doing Sports : It helps lose 2kg a week.

Of course, method 2 looks good. However, There is a hazard gaining 3kg due to the yo-yo phenomenon with probability of 0.2 Controlling diet costs 30,000 won a week and doing sports will incur a cost of 40,000 won a week. What kind of policy should I choose to lose weight? (Fortunately, when he decided to lose weight, his weight doesn't over 99kg.)

Preparation

```
gamma=1
states=np.arange(99,89,-1).astype('str')

P_diet=pd.DataFrame(np.array([[0,1,0,0,0,0,0,0,0,0], # 99kg
                [0,0,1,0,0,0,0,0,0,0], # 98kg
                [0,0,0,1,0,0,0,0,0,0], # 97kg
                [0,0,0,0,1,0,0,0,0,0], # 96kg
                [0,0,0,0,0,1,0,0,0,0], # 95kg
                [0,0,0,0,0,0,1,0,0,0], # 94kg
                [0,0,0,0,0,0,0,1,0,0], # 93kg
                [0,0,0,0,0,0,0,0,1,0], # 92kg
                [0,0,0,0,0,0,0,0,0,1], # 91kg
                [0,0,0,0,0,0,0,0,0,1]]), index=states, columns=states) # 90kg

P_diet
```

```
##     99  98  97  96  95  94  93  92  91  90
## 99   0   1   0   0   0   0   0   0   0   0
## 98   0   0   1   0   0   0   0   0   0   0
## 97   0   0   0   1   0   0   0   0   0   0
## 96   0   0   0   0   1   0   0   0   0   0
## 95   0   0   0   0   0   1   0   0   0   0
## 94   0   0   0   0   0   0   1   0   0   0
## 93   0   0   0   0   0   0   0   1   0   0
## 92   0   0   0   0   0   0   0   0   1   0
## 91   0   0   0   0   0   0   0   0   0   1
## 90   0   0   0   0   0   0   0   0   0   1
```

```
P_sport=pd.DataFrame(np.array([[0.2,0,0.8,0,0,0,0,0,0,0], # 99kg
                [0.2,0,0,0.8,0,0,0,0,0,0], # 98kg
                [0.2,0,0,0,0.8,0,0,0,0,0], # 97kg
                [0.2,0,0,0,0,0.8,0,0,0,0], # 96kg
                [0,0.2,0,0,0,0,0.8,0,0,0], # 95kg
                [0,0,0.2,0,0,0,0,0.8,0,0], # 94kg
                [0,0,0,0.2,0,0,0,0,0.8,0], # 93kg
                [0,0,0,0,0.2,0,0,0,0,0.8], # 92kg
                [0,0,0,0,0,0.2,0,0,0,0.8], # 91kg
                [0,0,0,0,0,0,0,0,0,1]]), index=states, columns=states) # 90kg
```

```
P_sport
```

```
##      99   98   97   96   95   94   93   92   91   90
## 99  0.2  0.0  0.8  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## 98  0.2  0.0  0.0  0.8  0.0  0.0  0.0  0.0  0.0  0.0
## 97  0.2  0.0  0.0  0.0  0.8  0.0  0.0  0.0  0.0  0.0
## 96  0.2  0.0  0.0  0.0  0.0  0.8  0.0  0.0  0.0  0.0
## 95  0.0  0.2  0.0  0.0  0.0  0.0  0.8  0.0  0.0  0.0
## 94  0.0  0.0  0.2  0.0  0.0  0.0  0.0  0.8  0.0  0.0
## 93  0.0  0.0  0.0  0.2  0.0  0.0  0.0  0.0  0.8  0.0
## 92  0.0  0.0  0.0  0.0  0.2  0.0  0.0  0.0  0.0  0.8
## 91  0.0  0.0  0.0  0.0  0.0  0.2  0.0  0.0  0.0  0.8
## 90  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

```
R_s_a=pd.DataFrame(np.c_[np.repeat(-3,len(states)), np.repeat(-4,len(states))], index=states, columns=['diet'

R_s_a.loc['90']=0

R_s_a
```

```
##     diet  sport
## 99    -3     -4
## 98    -3     -4
## 97    -3     -4
## 96    -3     -4
## 95    -3     -4
## 94    -3     -4
## 93    -3     -4
## 92    -3     -4
## 91    -3     -4
## 90     0      0
```

Functions

```
def transition(given_pi, states, P_diet, P_sport):
    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['diet']*P_diet+action_dist['sport']*P_sport
        P_out.loc[s]=P.loc[s]

    return P_out

def reward_fn(given_pi):
    R_s_a=pd.DataFrame(np.c_[np.repeat(-3,len(states)), np.repeat(-4,len(states))], index=states, columns=['d

    R_s_a.loc['90']=0

    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_diet=P_diet, P_sport=P_sport)

    gamma=1.0
    epsilon=10**(-8)

    v_old=np.repeat(0,10).reshape(10,1)
    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
```

Policy Evaluation

```
# Policy Evaluation

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

policy_eval(pi_50).T
```

```
## array([[-30.11305886, -27.48442842, -24.64884691, -21.62174771,
##         -18.31666794, -14.88026965, -11.57022569,  -7.82568028,
##          -4.98802696,   0.        ]])
```

Policy Improvement

```
# Policy Improvement

def policy_improve(V_old, pi_old, R_s_a, gamma, P_diet, P_sport):
    q_s_a=R_s_a+np.c_[np.dot(gamma*P_diet,V_old), np.dot(gamma*P_sport, V_old)]

    pi_new_vec=q_s_a.idxmax(axis=1)
    pi_new=pd.DataFrame(np.zeros(pi_old.shape), index=pi_old.index, columns=pi_old.columns)

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

    return pi_new



pi_old=pi_50
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_diet=P_diet, P_sport=P_sport)

pi_new
```

```
##      diet  sport
## 99   0.0    1.0
## 98   0.0    1.0
## 97   1.0    0.0
## 96   1.0    0.0
## 95   1.0    0.0
## 94   1.0    0.0
## 93   1.0    0.0
## 92   0.0    1.0
## 91   1.0    0.0
## 90   1.0    0.0
```