D_case

Wonryeol, Jeong

2021-02-15

Contents

Case introduction **2**

Problem discription **2**

Implement 3

New Situation(Experiment) **5**

Case introduction

Problem discription

There are three ways to get to 90 meters in total 90 meters. interval, running ,rest. Probability is below ## Preparation and description note

```
import numpy as np
import pandas as pd
gamma = 1
states = np.arange(0,100,10).astype('str')
P_running=pd.DataFrame(np.matrix([[0,1,0,0,0,0,0,0,0,0],
                                  [0,0,1,0,0,0,0,0,0,0],
                                  [0,0,0,1,0,0,0,0,0,0],
                                  [0,0,0,0,1,0,0,0,0,0],
                                  [0,0,0,0,0,1,0,0,0,0],
                                  [0,0,0,0,0,0,1,0,0,0],
                                  [0,0,0,0,0,0,0,1,0,0],
                                  [0,0,0,0,0,0,0,0,1,0],
                                  [0,0,0,0,0,0,0,0,0,1],
                                  [0,0,0,0,0,0,0,0,0,1],


                                  ]), index=states,columns=states)

P_rest=pd.DataFrame(np.matrix([[1,0,0,0,0,0,0,0,0,0],
                                  [0,1,0,0,0,0,0,0,0,0],
                                  [0,0,1,0,0,0,0,0,0,0],
                                  [0,0,0,1,0,0,0,0,0,1],
                                   [0,0,0,0,1,0,0,0,0,0],
                                  [0,0,0,0,0,1,0,0,0,0],
                                  [0,0,0,0,0,0,1,0,0,0],
                                  [0,0,0,0,0,0,0,1,0,0],
                                  [0,0,0,0,0,0,0,0,1,0],
                                  [0,0,0,0,0,0,0,0,0,1]

                                  ]), index=states,columns=states)

P_interval=pd.DataFrame(np.matrix([
                                    [0,.3,0,.7,0,0,0,0,0,0],
                                   [0,0,.3,0,.7,0,0,0,0,0],
                                   [0,0,0,.3,0,.7,0,0,0,0],
                                   [0,0,0,0,.3,0,.7,0,0,0],
                                   [0,0,0,0,0,.3,0,.7,0,0],
                                   [0,0,0,0,0,0,.3,0,.7,0],
                                   [0,0,0,0,0,0,0,.3,0,.7],
                                   [0,0,0,0,0,0,0,0,.3,.7],
                                   [0,0,0,0,0,0,0,0,0,1],
                                   [0,0,0,0,0,0,0,0,0,1]
]), index=states, columns=states)

q_s_a_init = pd.DataFrame(np.zeros((len(states),3)),states,["running","interval","rest"])
```

R and Pi(probability for strategy) are below

```
# reword
R_s_a=pd.DataFrame(np.array([-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
                             -2,-2,-2,-2,-2,-2,-2,-2,-2,-2,
                              0.1,.1,.1,.1,.1,.1,.1,.1,.1,.1]).reshape(len(states),3,order='F'),columns
R_s_a.T
```

```
##             0   10   20   30   40   50   60   70   80   90
## running  -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
## interval -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0
## rest      0.1  0.1  0.1  0.1  0.1  0.1  0.1  0.1  0.1  0.1
```

```
# pi_50
pi=pd.DataFrame(np.array([0.45,0.9,0.1]*10).reshape(10,3),index=states, columns=['running','interval','
```

```
pi.T
```

```
##              0    10    20    30    40    50    60    70    80    90
## running   0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45  0.45
## interval  0.90  0.90  0.90  0.90  0.90  0.90  0.90  0.90  0.90  0.90
## rest      0.10  0.10  0.10  0.10  0.10  0.10  0.10  0.10  0.10  0.10
```

Implement

```
def simul_path(pi,P_interval,P_running,P_rest,R_s_a):
    s_now = "0"
    history_i = [s_now]
    while s_now != '90':
        if np.random.uniform(0,1,1) < pi.loc[s_now,"running"] :


            a_now = "running"
            P = P_running
        elif np.random.uniform(0,1,1) < pi.loc[s_now,"interval"] :
            a_now = "interval"
            P = P_interval
        else:
            a_now = "rest"
            P = P_rest


        r_now = R_s_a.loc[s_now,a_now]

        s_next = pd.Series(np.cumsum(P.loc[s_now,])<np.random.uniform(0,1)).idxmin()
        history_i.extend([a_now,r_now,s_next])
        s_now = s_next

    return history_i
```

```
sample_path = simul_path(pi,P_interval,P_running,P_rest,R_s_a)
sample_path
```

```
## ['0', 'running', -1.0, '10', 'interval', -2.0, '40', 'interval', -2.0, '50', 'interval', -2.0, '80',
```

```
def pol_eval_MC(sample_path, q_s_a, alpha ):
    Q_s_a = q_s_a.copy()
    for j in range(0,len(sample_path)-1,3):

        s = sample_path[j]

        a = sample_path[j+1]

        G = pd.Series(sample_path)[list(range(j+2,len(sample_path),3))].astype('float').sum()

        Q_s_a.loc[s,a] = Q_s_a.loc[s,a] +alpha*(G- Q_s_a.loc[s,a])

    return Q_s_a

q_s_a = pol_eval_MC(sample_path,q_s_a_init,alpha = 0.1)
q_s_a
```

```
##     running  interval  rest
## 0      -0.8       0.0   0.0
## 10      0.0      -0.7   0.0
## 20      0.0       0.0   0.0
## 30      0.0       0.0   0.0
## 40      0.0      -0.5   0.0
## 50      0.0      -0.3   0.0
## 60      0.0       0.0   0.0
## 70      0.0       0.0   0.0
## 80     -0.1       0.0   0.0
## 90      0.0       0.0   0.0
```

```
# Skiier.R(7)
def pol_imp(pi,q_s_a,epsilon):
    Pi = pi.copy()
    for i in list(pi.index):
        if np.random.uniform(0,1,1) > epsilon:

            Pi.loc[i] = 0
            Pi.loc[i,q_s_a.loc[i].idxmin()]=1
            if i == '90':
                print(Pi.loc[i,q_s_a.loc[i].idxmax()
                ])

        else:
            Pi.loc[i,:] = 1/q_s_a.shape[1]
    return Pi

pi = pol_imp(pi,q_s_a, 0)
```

```
## 1.0
```

```
pi
```

```
##     running  interval  rest
## 0       1.0       0.0   0.0
## 10      0.0       1.0   0.0
## 20      1.0       0.0   0.0
## 30      1.0       0.0   0.0
## 40      0.0       1.0   0.0
## 50      0.0       1.0   0.0
## 60      1.0       0.0   0.0
## 70      1.0       0.0   0.0
## 80      1.0       0.0   0.0
## 90      1.0       0.0   0.0
```

New Situation(Experiment)

- Hard to make a code… I'm trying to make a code

One person infected corona spreads to an average of three people. if vaccine A is introduced, propagation power an average of 0.1 persons, and cost 10 persons. If vaccine B is introduced, the propagation power will be average of 0.3 persons and cost is 5.Derive the results through Td learning.