F1

Reinforcement Learning Study

2021-01-28

차 례

Preperation

```
import numpy as np
import pandas as pd



states=np.arange(0,70+10,10).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)

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(len(stat

print("R_s_a:\n",R_s_a.T,"\n")
```

```
## R_s_a:
##       0   10   20   30   40   50   60   70
## n -1.0 -1.0 -1.0 -1.0  0.0 -1.0 -1.0  0.0
## s -1.5 -1.5 -1.5 -1.5 -0.5 -1.5 -1.5  0.0
```

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

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

print("Pi_speed:\n",pi_speed.T,'\n')
```

```
## Pi_speed:
##     0  10  20  30  40  50  60  70
## n  0   0   0   0   0   0   0   0
## s  1   1   1   1   1   1   1   1
```

```
print("Pi_50:\n",pi_50.T,'\n')
```

```
## Pi_50:
##       0   10   20   30   40   50   60   70
## n  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
## s  0.5  0.5  0.5  0.5  0.5  0.5  0.5  0.5
```

Simulator for Pi_speed

```
pi = pi_speed

np.random.seed(1234)
history =[]
MC_N = 10000

for MC_i in range(MC_N):
    s_now ="0"
    history_i =[s_now]

    while(s_now != "70"):
        if np.random.uniform(0,1) < pi.loc[s_now]['n']:
            a_now = "n"
            P = P_normal
        else:
            a_now = "s"
            P = P_speed

        r_now = str(R_s_a.loc[s_now][a_now])
        s_next = states[np.argmin(P.loc[s_now].cumsum() < np.random.uniform(0,1))]
        history_i.extend([a_now,r_now,s_next])
        s_now=s_next

    history.append(history_i)

history_speed =history

history_speed_20 = list(map(lambda x:",".join(x),history_speed[:20] ))

history_speed_20
```

```
##     ['0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',     '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-
1.5,70',    '0,s,-1.5,0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,40,s,-
0.5,60,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,40,s,-
0.5,60,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,10,s,-1.5,30,s,-1.5,50,s,-1.5,70',    '0,s,-1.5,20,s,-
1.5,40,s,-0.5,30,s,-1.5,50,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-
1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-
1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-
1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',    '0,s,-
1.5,20,s,-1.5,10,s,-1.5,30,s,-1.5,50,s,-1.5,70', '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70', '0,s,-
```

```
1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',   '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70',   '0,s,-
1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70']
```

Simulator Pi_50

```
pi = pi_50

np.random.seed(1234)
history =[]
MC_N = 10000

for MC_i in range(MC_N):
    s_now ="0"
    history_i =[s_now]

    while(s_now != "70"):
        if np.random.uniform(0,1) < pi.loc[s_now]['n']:
            a_now = "n"
            P = P_normal
        else:
            a_now = "s"
            P = P_speed

        r_now = str(R_s_a.loc[s_now][a_now])
        s_next = states[np.argmin(P.loc[s_now].cumsum() < np.random.uniform(0,1))]
        history_i.extend([a_now,r_now,s_next])
        s_now=s_next

    history.append(history_i)

history_50 =history

history_50_20=list(map(lambda x:",".join(x),history_50[:20] ))
history_pi_50 = list(map(lambda x:",".join(x),history_50))

history_50_20
```

```
## ['0,n,-1.0,10,n,-1.0,20,s,-1.5,40,n,0.0,50,s,-1.5,70', '0,n,-1.0,10,s,-1.5,30,n,-1.0,40,s,-
0.5,30,s,-1.5,50,n,-1.0,60,n,-1.0,70', '0,s,-1.5,20,n,-1.0,30,n,-1.0,40,s,-0.5,60,s,-1.5,70', '0,s,-
1.5,20,n,-1.0,30,n,-1.0,40,n,0.0,50,n,-1.0,60,s,-1.5,70', '0,n,-1.0,10,n,-1.0,20,n,-1.0,30,s,-
1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70', '0,n,-1.0,10,n,-1.0,20,n,-1.0,30,n,-1.0,40,n,0.0,50,s,-
1.5,70',    '0,n,-1.0,10,n,-1.0,20,n,-1.0,30,n,-1.0,40,n,0.0,50,s,-1.5,70',    '0,s,-1.5,20,s,-
1.5,40,s,-0.5,60,n,-1.0,70',    '0,s,-1.5,20,n,-1.0,30,s,-1.5,50,n,-1.0,60,s,-1.5,70',    '0,s,-
1.5,20,s,-1.5,40,n,0.0,50,n,-1.0,60,n,-1.0,70', '0,n,-1.0,10,s,-1.5,30,n,-1.0,40,s,-0.5,60,s,-
1.5,70',    '0,s,-1.5,20,n,-1.0,30,n,-1.0,40,n,0.0,50,n,-1.0,60,s,-1.5,70',    '0,n,-1.0,10,s,-
```

```
1.5,30,n,-1.0,40,n,0.0,50,s,-1.5,70',   '0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70',   '0,n,-
1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70',   '0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70',   '0,s,-
1.5,20,s,-1.5,40,n,0.0,50,n,-1.0,60,n,-1.0,70', '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,n,-1.0,70', '0,n,-
1.0,10,n,-1.0,20,s,-1.5,40,n,0.0,50,n,-1.0,60,s,-1.5,70', '0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-
1.5,70']
```

Implementation 1 - pi_speed(vectorized)

```
pol_eval= pd.DataFrame(np.zeros(shape=(len(states),2)), index=states, columns=['count','sum'])

pol_eval.T
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## sum    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
for MC_i in range(len(history_speed)):
    history_i = history_speed[MC_i]

    for j in range(0, len(history_i),3):
        pol_eval.loc[history_i[j]]['count']+=1

        if j < len(history_i):
            pol_eval.loc[history_i[j]]['sum']+= np.sum(np.array(history_i[j+2:len(history_i)-1:3]).astype(flo

        else :
            pol_eval.loc[history_i[j]['sum']] +=0

pol_eval.T
```

```
##              0      10       20      30       40      50       60       70
## count  11225.0  1076.0  10291.0  1887.0   9485.0  2563.0   8563.0  10000.0
## sum   -65136.0 -5619.0 -42703.0 -6539.0 -22275.5 -4472.5 -14355.0      0.0
```

```
pol_eval['sum']/pol_eval['count']
```

```
## 0    -5.802762
## 10   -5.222119
## 20   -4.149548
## 30   -3.465289
## 40   -2.348498
## 50   -1.745025
## 60   -1.676398
## 70    0.000000
## dtype: float64
```

Implementation 2 Pi^speed (runing estimate)

```
pol_eval= pd.DataFrame(np.zeros(shape=(len(states),2)), index=states, columns=['count','est'])

pol_eval.T
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## est    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
new_info=0
for MC_i in range(len(history_speed)):
    history_i = history_speed[MC_i]

    for j in range(0, len(history_i),3):
        # update count
        pol_eval.loc[history_i[j]]['count']+=1
        current_cnt = pol_eval.loc[history_i[j]]['count']

        if j < len(history_i):
            new_info = np.sum(np.array(history_i[j+2:len(history_i)-1:3]).astype(float))

        else :
            new_info = 0

        alpha = 1/current_cnt
        pol_eval.loc[history_i[j]]['est']+=alpha*(new_info-pol_eval.loc[history_i[j]]['est'])

pol_eval
```

```
##       count       est
## 0   11225.0 -5.802762
## 10   1076.0 -5.222119
## 20  10291.0 -4.149548
## 30   1887.0 -3.465289
## 40   9485.0 -2.348498
## 50   2563.0 -1.745025
## 60   8563.0 -1.676398
## 70  10000.0  0.000000
```

Implementation 3 Pi^50 (vectorized)

```
pol_eval= pd.DataFrame(np.zeros(shape=(len(states),2)), index=states, columns=['count','sum'])


for MC_i in range(len(history_speed)):
    history_i = history_50[MC_i]
    for j in range(0, len(history_i),3):
        pol_eval.loc[history_i[j]]['count']+=1

        if j < len(history_i):
            pol_eval.loc[history_i[j]]['sum']+= np.sum(np.array(history_i[j+2:len(history_i)-1:3]).astype(flo

        else :
            pol_eval.loc[history_i[j]['sum']] +=0

pol_eval.T
```

```
##              0       10       20       30       40       50      60       70
## count  10863.0   5792.0   8140.0   7121.0   7549.0   7363.0  6991.0  10000.0
## sum   -64904.5 -29662.5 -33549.0 -24133.0 -15410.0 -14874.5 -9436.5      0.0
```

```
pol_eval['sum']/pol_eval['count']
```

```
## 0    -5.974823
## 10   -5.121288
## 20   -4.121499
## 30   -3.388990
## 40   -2.041330
## 50   -2.020168
## 60   -1.349807
## 70    0.000000
## dtype: float64
```

Implementation 4 Pi^50 (runing estimate)

```
pol_eval= pd.DataFrame(np.zeros(shape=(len(states),2)), index=states, columns=['count','est'])

pol_eval.T
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
## est    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
new_info=0
for MC_i in range(len(history_speed)):
    history_i = history_50[MC_i]

    for j in range(0, len(history_i),3):
        # update count
        pol_eval.loc[history_i[j]]['count']+=1
        current_cnt = pol_eval.loc[history_i[j]]['count']

        if j < len(history_i):
            new_info = np.sum(np.array(history_i[j+2:len(history_i)-1:3]).astype(float))

        else :
            new_info = 0

        alpha = 1/current_cnt
        pol_eval.loc[history_i[j]]['est']+=alpha*(new_info-pol_eval.loc[history_i[j]]['est'])

pol_eval
```

```
##       count       est
## 0   10863.0 -5.974823
## 10   5792.0 -5.121288
## 20   8140.0 -4.121499
## 30   7121.0 -3.388990
## 40   7549.0 -2.041330
## 50   7363.0 -2.020168
## 60   6991.0 -1.349807
## 70  10000.0  0.000000
```

implementation 5 - pi^speed with tempral diffrence

```
pol_eval= pd.DataFrame(np.zeros(shape=(len(states),2)), index=states, columns=['count','est'])

pol_eval.T
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## est    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
new_info=0
for MC_i in range(len(history_speed)):
    history_i = history_speed[MC_i]

    for j in range(0, len(history_i),3):
        # update count
        pol_eval.loc[history_i[j]]['count']+=1
        current_cnt = pol_eval.loc[history_i[j]]['count']

        if j < len(history_i)-3:
            TD_target = np.array(history_i[j+2]).astype('float')+pol_eval.loc[history_i[j+3]]['est']

        else :
            TD_target = 0

        alpha = 1/current_cnt
        pol_eval.loc[history_i[j]]['est']+=alpha*(TD_target-pol_eval.loc[history_i[j]]['est'])

pol_eval
```

```
##       count       est
## 0   11225.0 -5.738838
## 10   1076.0 -5.186466
## 20  10291.0 -4.128507
## 30   1887.0 -3.479288
## 40   9485.0 -2.349398
## 50   2563.0 -1.746177
## 60   8563.0 -1.675699
## 70  10000.0  0.000000
```

implementation 6 - pi^50 with tempral diffrence

```
pol_eval= pd.DataFrame(np.zeros(shape=(len(states),2)), index=states, columns=['count','est'])

pol_eval.T
```

```
##          0   10   20   30   40   50   60   70
## count  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## est    0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

```
new_info=0
for MC_i in range(len(history_speed)):
    history_i = history_50[MC_i]

    for j in range(0, len(history_i),3):
        # update count
        pol_eval.loc[history_i[j]]['count']+=1
        current_cnt = pol_eval.loc[history_i[j]]['count']

        if j < len(history_i)-3:
            TD_target = np.array(history_i[j+2]).astype('float')+pol_eval.loc[history_i[j+3]]['est']

        else :
            TD_target = 0

        alpha = 1/current_cnt
        pol_eval.loc[history_i[j]]['est']+=alpha*(TD_target-pol_eval.loc[history_i[j]]['est'])

pol_eval
```

```
##       count       est
## 0   10863.0 -5.844920
## 10   5792.0 -5.052485
## 20   8140.0 -4.079273
## 30   7121.0 -3.372556
## 40   7549.0 -2.038181
## 50   7363.0 -2.026683
## 60   6991.0 -1.351198
## 70  10000.0  0.000000
```

```
"Done "
```

```
## [1] "Done "
```