F1 Python

Jaemin Park

2021-01-25

차 례

p.9 Preparation

```
states = np.arange(0,80,10)
P_normal = 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]])
P_speed = np.matrix([[0.1,0,0.9,0,0,0,0,0],[0.1,0,0,0.9,0,0,0,0],
[0,0.1,0,0,0.9,0,0,0],[0,0,0.1,0,0,0.9,0,0],[0,0,0,0.1,0,0,0.9,0],
[0,0,0,0,0.1,0,0,0.9],[0,0,0,0,0,0.1,0,0.9],[0,0,0,0,0,0,0,1]])
P_normal = pd.DataFrame(P_normal,states,states)
P_speed = pd.DataFrame(P_speed,states,states)
R_s_a = np.matrix([[-1,-1,-1,-1,0,-1,-1,0],[-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]]).T
R_s_a = pd.DataFrame(R_s_a,states,["n","s"])

pi_speed = np.hstack((np.repeat(0,len(states)).reshape(8,1),np.repeat(1,len(states)).reshape(8,1)))
pi_speed = pd.DataFrame(pi_speed,states,["n","s"])

pi_50 = np.hstack((np.repeat(0.5,len(states)).reshape(8,1),np.repeat(0.5,len(states)).reshape(8,1)))
pi_50 = pd.DataFrame(pi_50,states,["n","s"])
pi_speed.T
```

```
##    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
```

```
pi_50.T
```

```
##     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
```

p.11 Simulator π^{speed}

```
pi = pi_speed
history = []
MC_N = 10000
for MC_i in range(MC_N):
    s_now = 0
    history_i = []
    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 = R_s_a.loc[s_now][a_now]
        s_next = states[np.argmin(P.loc[s_now].cumsum()<np.random.uniform(0,1))].item()
        history_i.extend([s_now,a_now,r_now])
        s_now = s_next
    history.append(history_i)

history_speed = history
print(pd.DataFrame(np.matrix(history_speed[:20])).T)
```

```
##                                                    0
## 0   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 30, s, ...
## 1   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 2   [0, s, -1.5, 0, s, -1.5, 20, s, -1.5, 40, s, -...
## 3   [0, s, -1.5, 0, s, -1.5, 20, s, -1.5, 40, s, -...
## 4   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 5   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 6   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 7   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 8   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 9   [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 10  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 11  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 12  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 30, s, ...
## 13  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 14  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
```

```
## 15  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 16  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 17  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 18  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
## 19  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, s, ...
##
## D:\miniconda3\envs\r-reticulate\lib\site-packages\numpy\matrixlib\defmatrix.py:145: VisibleDeprec
##   arr = N.array(data, dtype=dtype, copy=copy)
```

p.13 Simulator π^{50}

```
pi = pi_50
history = []
MC_N = 10000
for MC_i in range(MC_N):
    s_now = 0
    history_i = []
    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 = R_s_a.loc[s_now][a_now]
        s_next = states[np.argmin(P.loc[s_now].cumsum()<np.random.uniform(0,1))].item()
        history_i.extend([s_now,a_now,r_now])
        s_now = s_next
    history.append(history_i)

history_50 = history
print(pd.DataFrame(np.matrix(history_50[:20])).T)
```

```
##                                                    0
## 0   [0, n, -1.0, 10, n, -1.0, 20, n, -1.0, 30, n, ...
## 1   [0, s, -1.5, 20, s, -1.5, 40, n, 0.0, 50, s, -...
## 2   [0, n, -1.0, 10, n, -1.0, 20, n, -1.0, 30, n, ...
## 3   [0, n, -1.0, 10, n, -1.0, 20, n, -1.0, 30, n, ...
## 4   [0, s, -1.5, 20, n, -1.0, 30, s, -1.5, 20, n, ...
## 5   [0, s, -1.5, 20, s, -1.5, 40, n, 0.0, 50, s, -...
## 6   [0, s, -1.5, 20, s, -1.5, 40, n, 0.0, 50, s, -...
## 7   [0, s, -1.5, 20, n, -1.0, 30, n, -1.0, 40, n, ...
## 8   [0, s, -1.5, 20, n, -1.0, 30, s, -1.5, 50, s, ...
## 9   [0, n, -1.0, 10, s, -1.5, 30, n, -1.0, 40, n, ...
## 10  [0, s, -1.5, 20, n, -1.0, 30, s, -1.5, 50, s, ...
## 11  [0, s, -1.5, 20, s, -1.5, 40, n, 0.0, 50, n, -...
## 12  [0, s, -1.5, 20, s, -1.5, 40, s, -0.5, 60, n, ...
## 13  [0, n, -1.0, 10, s, -1.5, 30, n, -1.0, 40, s, ...
## 14  [0, n, -1.0, 10, s, -1.5, 30, n, -1.0, 40, n, ...
```

```
## 15  [0, s, -1.5, 20, n, -1.0, 30, n, -1.0, 40, s, ...
## 16  [0, n, -1.0, 10, n, -1.0, 20, n, -1.0, 30, s, ...
## 17  [0, n, -1.0, 10, s, -1.5, 30, n, -1.0, 40, s, ...
## 18  [0, s, -1.5, 20, n, -1.0, 30, s, -1.5, 50, n, ...
## 19  [0, s, -1.5, 20, n, -1.0, 30, n, -1.0, 40, s, ...
##
## D:\miniconda3\envs\r-reticulate\lib\site-packages\numpy\matrixlib\defmatrix.py:145: VisibleDeprec
##   arr = N.array(data, dtype=dtype, copy=copy)
```

p.17 Implementation 1 π^{speed} (vectorized)

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(states)*2))).reshape(len(states),2), index=states, col
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(MC_N):
    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']+=pd.Series(history_i)[range(j+2,len(history_i)-1,3)].a

        else:
            pol_eval.loc[history_i[j]]['sum']+=0
print(pol_eval.T)
```

```
##             0       10       20      30      40      50      60   70
## count  11239.0   998.0  10303.0  1886.0  9545.0  2537.0  8548.0  0.0
## sum   -48405.0 -3737.5 -27179.5 -3722.5 -8184.0  -556.5 -1446.0  0.0
```

```
pol_cal=pd.DataFrame(pol_eval['sum']/pol_eval['count'])
print(pol_cal.T)
```

```
##          0        10        20        30        40        50        60  70
## 0 -4.306878 -3.74499 -2.638018 -1.973754 -0.857412 -0.219354 -0.169162 NaN
```

p.19 Implementation 2 π^{speed} (running estimate)

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(states)*2))).reshape(len(states),2), index=states, col
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
```

```
for MC_i in range(MC_N):
    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']

        # return is the new info
        if j < len(history_i):
            new_info=pd.Series(history_i)[range(j+2,len(history_i)-1,3)].astype('float').sum()

        else:
            new_info=0

        # update the last estimate with new info
        alpha=1/current_cnt
        pol_eval.loc[history_i[j]]['est']+=alpha*(new_info-pol_eval.loc[history_i[j]]['est'])


np.round(pol_eval.T,2)
```

```
##              0       10        20       30       40       50       60   70
## count  11239.00  998.00  10303.00  1886.00  9545.00  2537.00  8548.00  0.0
## est       -4.31   -3.74     -2.64    -1.97    -0.86    -0.22    -0.17  0.0
```

p.21 Implementation 3 π^{50} (vectorized)

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(states)*2))).reshape(len(states),2), index=states, col
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(MC_N):
    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']+=pd.Series(history_i)[range(j+2,len(history_i)-1,3)].a

        else:
            pol_eval.loc[history_i[j]]['sum']+=0
pol_eval.T
```

```
##             0        10       20       30      40      50      60   70
## count  10859.0   5871.0   8132.0   7043.0  7598.0  7233.0  7059.0  0.0
## sum   -50421.5 -22299.0 -22650.5 -14449.0 -5620.0 -4740.5  -760.0  0.0
```

```
pol_cal=pd.DataFrame(pol_eval['sum']/pol_eval['count'])
print(pol_cal.T)
```

```
##          0        10        20        30        40        50        60  70
## 0 -4.643291 -3.79816 -2.785354 -2.051541 -0.739668 -0.655399 -0.107664 NaN
```

p.23 Implementation 4 π^{50} (running estimate)

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(states)*2))).reshape(len(states),2), index=states, col
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
```

```
for MC_i in range(MC_N):
    history_i=history_50[MC_i]

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

        # return is the new info
        if j < len(history_i):
            new_info=pd.Series(history_i)[range(j+2,len(history_i)-1,3)].astype('float').sum()

        else:
            new_info=0

        # update the last estimate with new info
        alpha=1/current_cnt
        pol_eval.loc[history_i[j]]['est']+=alpha*(new_info-pol_eval.loc[history_i[j]]['est'])


np.round(pol_eval.T,2)
```

```
##              0       10       20       30       40       50       60   70
## count  10859.00  5871.0  8132.00  7043.00  7598.00  7233.00  7059.00  0.0
## est       -4.64    -3.8    -2.79    -2.05    -0.74    -0.66    -0.11  0.0
```