F1 Python ver

Lee SungHo

2021-02-01

Contents

page 9 Preparation

```
import numpy as np
import pandas as pd

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)

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
print(R_s_a.T)
```

```
##      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=[

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

print(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
```

```
print(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
```

page 11 simulator 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=list(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(np.cumsum(P.loc[s_now,]) < 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
func=np.vectorize(lambda x: ','.join(x))
pd.Series(func(history_speed[:20]))
```

```
## 0             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 1             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 2     0,s,-1.5,0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1....
## 3             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 4             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 5             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 6     0,s,-1.5,20,s,-1.5,10,s,-1.5,30,s,-1.5,50,s,-1...
## 7     0,s,-1.5,20,s,-1.5,40,s,-0.5,30,s,-1.5,50,s,-1...
## 8             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 9             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 10            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 11            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 12            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 13            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 14            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 15    0,s,-1.5,20,s,-1.5,10,s,-1.5,30,s,-1.5,50,s,-1...
## 16            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 17            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 18            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## 19            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## dtype: object
##
## C:\Users\LEESUN~1\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\numpy\core\_asarray.py:83: V
##   return array(a, dtype, copy=False, order=order)
```

page 13 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=list(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))].item()
        history_i.extend([a_now,r_now,s_next])
        s_now=s_next

    history.append(history_i)

history_50=history
func=np.vectorize(lambda x: ','.join(x))
pd.Series(func(history_50[:20]))
```

```
## 0     0,n,-1.0,10,n,-1.0,20,s,-1.5,40,n,0.0,50,s,-1....
## 1     0,n,-1.0,10,s,-1.5,30,n,-1.0,40,s,-0.5,30,s,-1...
## 2     0,s,-1.5,20,n,-1.0,30,n,-1.0,40,s,-0.5,60,s,-1...
## 3     0,s,-1.5,20,n,-1.0,30,n,-1.0,40,n,0.0,50,n,-1....
## 4     0,n,-1.0,10,n,-1.0,20,n,-1.0,30,s,-1.5,20,s,-1...
## 5     0,n,-1.0,10,n,-1.0,20,n,-1.0,30,n,-1.0,40,n,0....
## 6     0,n,-1.0,10,n,-1.0,20,n,-1.0,30,n,-1.0,40,n,0....
## 7             0,s,-1.5,20,s,-1.5,40,s,-0.5,60,n,-1.0,70
## 8     0,s,-1.5,20,n,-1.0,30,s,-1.5,50,n,-1.0,60,s,-1...
## 9     0,s,-1.5,20,s,-1.5,40,n,0.0,50,n,-1.0,60,n,-1....
## 10    0,n,-1.0,10,s,-1.5,30,n,-1.0,40,s,-0.5,60,s,-1...
## 11    0,s,-1.5,20,n,-1.0,30,n,-1.0,40,n,0.0,50,n,-1....
## 12    0,n,-1.0,10,s,-1.5,30,n,-1.0,40,n,0.0,50,s,-1....
## 13            0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70
## 14            0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70
## 15            0,n,-1.0,10,s,-1.5,30,s,-1.5,50,s,-1.5,70
## 16    0,s,-1.5,20,s,-1.5,40,n,0.0,50,n,-1.0,60,n,-1....
## 17            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,n,-1.0,70
## 18    0,n,-1.0,10,n,-1.0,20,s,-1.5,40,n,0.0,50,n,-1....
## 19            0,s,-1.5,20,s,-1.5,40,s,-0.5,60,s,-1.5,70
## dtype: object
##
## C:\Users\LEESUN~1\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\numpy\core\_asarray.py:83:
```

```
##   return array(a, dtype, copy=False, order=order)
```

page 17 Implementation 1 π^{speed} (vectorized)

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(states)*2))).reshape(len(states),2), index=states, columns
print(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)].astyp

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

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

```
##           0         10         20         30         40         50         60   70
## 0 -5.802762 -5.222119 -4.149548 -3.465289 -2.348498 -1.745025 -1.676398  0.0
```

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

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(states)*2))).reshape(len(states),2), index=states, columns
print(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'])


print(pol_eval.T)
```

```
##                  0            10  ...           60       70
## count  11225.000000  1076.000000  ...  8563.000000  10000.0
## est       -5.802762    -5.222119  ...    -1.676398      0.0
##
## [2 rows x 8 columns]
```

page 21 Implementation 3 π^{50} (vectorized)

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

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

```
##           0        10        20       30        40        50        60   70
## 0 -5.974823 -5.121288 -4.121499 -3.38899  -2.04133 -2.020168 -1.349807  0.0
```

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

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


print(pol_cal.T)
```

```
##           0         10         20        30        40         50         60   70
## 0 -5.974823 -5.121288 -4.121499 -3.38899 -2.04133 -2.020168 -1.349807  0.0
```

page 36 Implementation 5 pi speed

```
import pandas as pd
import numpy as np

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

print(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 episode_i in range(len(history_speed)):
  history_i = history_speed[episode_i]

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

    #build TD target
    if(j < len(history_i)-3):
      TD_tgt = float(history_i[j+2])+pol_eval.loc[history_i[j+3]]['est']

    else:
      TD_tgt = 0

    # TD-updating
    alpha = 1/current_cnt

    pol_eval.loc[history_i[j]]['est'] += alpha*(TD_tgt - pol_eval.loc[history_i[j]]['est'])

print(pol_eval.T)
```

```
##                  0           10  ...           60       70
## count  11225.000000  1076.000000  ...  8563.000000  10000.0
## est       -5.738838    -5.186466  ...    -1.675699      0.0
##
## [2 rows x 8 columns]
```

page 37 Implementation 6 pi 50

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

print(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 episode_i in range(len(history_50)):
  history_i = history_50[episode_i]

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

    #build TD target
    if(j < len(history_i)-3):
      TD_tgt = float(history_i[j+2])+pol_eval.loc[history_i[j+3]]['est']

    else:
      TD_tgt = 0

    # TD-updating
    alpha = 1/current_cnt

    pol_eval.loc[history_i[j]]['est'] += alpha*(TD_tgt - pol_eval.loc[history_i[j]]['est'])

print(pol_eval.T)
```

```
##                  0           10           20  ...           50           60       70
## count  10863.00000  5792.000000  8140.000000  ...  7363.000000  6991.000000  10000.0
## est       -5.84492    -5.052485    -4.079273  ...    -2.026683    -1.351198      0.0
##
## [2 rows x 8 columns]
```