F1 Python ver

Lee SungHo

2021-02-01

Contents

Environment

```
import pandas as pd
import numpy as np


action = ['move_left','move_not', 'move_right']
state = [1,2,3,4,5,6,7] # s1 ~ s7

P_ML = pd.DataFrame(np.matrix([[0,0,0,0,0,0,0],
                               [1,0,0,0,0,0,0],
                               [0,1,0,0,0,0,0],
                               [0,0,1,0,0,0,0],
                               [0,0,0,1,0,0,0],
                               [0,0,0,0,1,0,0],
                                [0,0,0,0,0,1,0]
                                ]),index = state , columns = state)

P_MR = pd.DataFrame(np.matrix([[0,1,0,0,0,0,0],
                               [0,0,1,0,0,0,0],
                               [0,0,0,1,0,0,0],
                               [0,0,0,0,1,0,0],
                              [0,0,0,0,0,1,0],
                                [0,0,0,0,0,0,1],
                               [0,0,0,0,0,0,0]
                                ]),index = state , columns = state)

P_MN = pd.DataFrame(np.matrix([[1,0,0,0,0,0,0],
                               [0,1,0,0,0,0,0],
                               [0,0,1,0,0,0,0],
                               [0,0,0,1,0,0,0],
                              [0,0,0,0,1,0,0],
                                [0,0,0,0,0,1,0],
                               [0,0,0,0,0,0,1]
                                ]),index = state , columns = state)



pi_mars =pd.DataFrame(np.matrix([np.repeat(0.4,len(state)),np.repeat(0.2,len(state)),np.repeat(0.4,len(s

pi_mars['move_left'][1] = 0
pi_mars['move_not'][1] = 0.6

pi_mars['move_right'][7] = 0
pi_mars['move_not'][7] = 0.6

print(pi_mars.T)

##               1    2    3    4    5    6    7
## move_left   0.0  0.4  0.4  0.4  0.4  0.4  0.4
## move_not    0.6  0.2  0.2  0.2  0.2  0.2  0.6
## move_right  0.4  0.4  0.4  0.4  0.4  0.4  0.0

reward = np.array([1,0,0,0,0,0,10])
```

```
print(reward)
```

```
## [ 1  0  0  0  0  0 10]
```

Simulator

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

for MC_i in range(MC_N):
  s_now = 4 # Start s4
  history_i = [4]
  count = 0

  while count < 10 :

    probability = np.random.uniform(0,1)

    if probability < pi.loc[s_now]['move_left']:
      a_now = 'move_left'
      P = P_ML
      s_next = s_now - 1

    elif probability >= pi.loc[s_now]['move_left'] and probability < (pi.loc[s_now]['move_left'] + pi.l

      a_now = 'move_not'
      P = P_MN
      s_next = s_now

    else:
      a_now = 'move_right'
      P = P_MR
      s_next = s_now + 1


    r_now = reward[s_now-1]
    history_i.extend([a_now,r_now,s_next])
    s_now = s_next
    count+=1

  history.append(history_i)


history[-5:]

## [[4, 'move_left', 0, 3, 'move_left', 0, 2, 'move_left', 0, 1, 'move_not', 1, 1, 'move_right', 1, 2,
```

Implementation 1 (vectorized)

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(state)*2))).reshape(len(state),2), index=state, columns=[
print(pol_eval.T)
```

```
##          1    2    3    4    5    6    7
## count  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
```

```
for MC_i in range(MC_N):
  history_i = history[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)].sum()

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

print(pol_eval.T)
```

```
##              1        2        3         4         5         6         7
## count   8759.0  12168.0  19057.0   29616.0   19243.0   12303.0    8854.0
## sum    16875.0  13878.0  28059.0  130448.0  124280.0  126723.0  170551.0
```

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

```
##          1         2         3         4         5          6          7
## 0  1.92659  1.140533  1.472372  4.404646  6.458452  10.300171  19.262593
```

Implementation 2 (vectorized)

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(state)*2))).reshape(len(state),2), index=state, columns=[
print(pol_eval.T)
```

```
##          1    2    3    4    5    6    7
## count  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
```

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

```
##      count        est
## 1   8759.0   1.926590
## 2  12168.0   1.140533
## 3  19057.0   1.472372
## 4  29616.0   4.404646
## 5  19243.0   6.458452
## 6  12303.0  10.300171
## 7   8854.0  19.262593
```

page 36 Implementation 3

```
pol_eval=pd.DataFrame(np.matrix(np.zeros((len(state)*2))).reshape(len(state),2), index=state, columns=[
print(pol_eval.T)
```

```
##          1    2    3    4    5    6    7
## count  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
```

```
for episode_i in range(len(history)):
  history_i = history[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'])

pol_eval
```

```
##      count        est
## 1   8759.0   2.705149
## 2  12168.0   1.788965
## 3  19057.0   1.996642
## 4  29616.0   3.456291
## 5  19243.0   6.536708
## 6  12303.0  12.466257
## 7   8854.0  24.314566
```