F4 Python Code

Kang, Eui Hyeon

2021-02-22

차 례

Preparation

```
states=np.arange(0,70+10,10).astype('str')
states
```

```
## array(['0', '10', '20', '30', '40', '50', '60', '70'], dtype='<U11')
```

```
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_normal
```

```
##     0  10  20  30  40  50  60  70
## 0   0   1   0   0   0   0   0   0
## 10  0   0   1   0   0   0   0   0
## 20  0   0   0   1   0   0   0   0
## 30  0   0   0   0   1   0   0   0
## 40  0   0   0   0   0   1   0   0
## 50  0   0   0   0   0   0   1   0
## 60  0   0   0   0   0   0   0   1
## 70  0   0   0   0   0   0   0   1
```

```
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)

P_speed
```

```
##       0   10   20   30   40   50   60   70
## 0   0.1  0.0  0.9  0.0  0.0  0.0  0.0  0.0
```

```
## 10  0.1  0.0  0.0  0.9  0.0  0.0  0.0  0.0
## 20  0.0  0.1  0.0  0.0  0.9  0.0  0.0  0.0
## 30  0.0  0.0  0.1  0.0  0.0  0.9  0.0  0.0
## 40  0.0  0.0  0.0  0.1  0.0  0.0  0.9  0.0
## 50  0.0  0.0  0.0  0.0  0.1  0.0  0.0  0.9
## 60  0.0  0.0  0.0  0.0  0.0  0.1  0.0  0.9
## 70  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

```
R_s_a=pd.DataFrame(np.c_[[-1,-1,-1,-1,0,-1,-1,0],[-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]], index=states, colum
```

```
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
```

```
q_s_a_init=pd.DataFrame(np.c_[np.repeat(0.0,len(states)), np.repeat(0.0,len(states))], index=states, columns=
```

```
q_s_a_init.T
```

```
# Policy
```

```
##      0   10   20   30   40   50   60   70
## n  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
## s  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

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

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

```
pi_50.T
```

```
# simul_step()
```

```
##      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
```

```
def simul_step(pi, s_now, P_normal, P_speed, R_s_a):
    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()

    if np.random.uniform(0,1) < pi.loc[s_next]['n']:
        a_next='n'
    else:
        a_next='s'

    sarsa=[s_now, a_now, r_now, s_next, a_next]

    return sarsa

sample_step=simul_step(pi=pi_speed, s_now='0', P_normal=P_normal, P_speed=P_speed, R_s_a=R_s_a)
sample_step

# pol_eval_TD
```

```
## ['0', 's', -1.5, '20', 's']
```

```
def pol_eval_TD(sample_step, q_s_a, alpha):
    s=sample_step[0]
    a=sample_step[1]
    r=sample_step[2]
    s_next=sample_step[3]
    a_next=sample_step[4]

    q_s_a.loc[s][a]=q_s_a.loc[s][a]+alpha*(r+q_s_a.loc[s_next][a_next]-q_s_a.loc[s][a])

    return q_s_a
```

```
# pol_eval_Q

def pol_eval_Q(sample_step, q_s_a, alpha):
    s=sample_step[0]
    a=sample_step[1]
    r=sample_step[2]
    s_next=sample_step[3]
    a_next=sample_step[4]

    q_s_a.loc[s][a]=q_s_a.loc[s][a]+alpha*(r+max(q_s_a.loc[s_next])-q_s_a.loc[s][a])
    return q_s_a

# pol_imp()

def pol_imp(pi, q_s_a, epsilon): # epsilon=exploration_rate
    for i in range(0, pi.shape[0]):
        if np.random.uniform(0,1)>epsilon: #exploitation
            pi.iloc[i]=0
            pi.iloc[i][q_s_a.iloc[i].idxmax()]=1
        else:
            pi.iloc[i]=1/q_s_a.shape[1]

    return pi
```

Q - Learning

```
from time import time
from copy import deepcopy

num_ep=10**5
beg_time=time()
q_s_a=deepcopy(q_s_a_init)
pi=deepcopy(pi_50)
exploration_rate=1

for epi_i in range(1,num_ep+1):
    s_now='0'
    while s_now!='70':
        sample_step=simul_step(pi, s_now, P_normal, P_speed, R_s_a)
        q_s_a=pol_eval_Q(sample_step, q_s_a, alpha=max(1/epi_i, 0.05))
        if epi_i%100 == 0 :
            pi=pol_imp(pi, q_s_a, epsilon=exploration_rate)

        s_now=sample_step[3]
        exploration_rate=max(exploration_rate*0.9995, 0.001)

end_time=time()

q_s_a.T
```

```
##           0        10        20        30        40        50        60   70
## n -5.665373 -4.665789 -3.667342 -2.652668 -1.656261 -1.999813 -1.000000  0.0
## s -5.302711 -4.814238 -3.849196 -3.228926 -1.942221 -1.646434 -1.671092  0.0
```

```
print(end_time-beg_time)
```

```
## 841.2134251594543
```

```
pi.T
```

```
##      0   10   20   30   40   50   60   70
## n  0.0  1.0  1.0  1.0  1.0  0.0  1.0  1.0
## s  1.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0
```

Double Q - Learning

```
def pol_eval_Q(sample_step, q_s_a, alpha):
    s=sample_step[0]
    a=sample_step[1]
    r=sample_step[2]
    s_next=sample_step[3]
    a_next=sample_step[4]

    q_s_a.loc[s][a]=q_s_a.loc[s][a]+alpha*(r+max(q_s_a.loc[s_next])-q_s_a.loc[s][a])
    return q_s_a

def pol_eval_dbl_Q(sample_step, q_s_a_1, q_s_a_2, alpha):
    s=sample_step[0]
    a=sample_step[1]
    r=sample_step[2]
    s_next=sample_step[3]

    if np.random.uniform(0,1) < 0.5 :
        q_s_a_1.loc[s][a]=q_s_a_2.loc[s][a]+alpha*(r+q_s_a_2.loc[s_next][q_s_a_1.loc[s_next].idxmax()]-q_s_a_
    else:
        q_s_a_2.loc[s][a]=q_s_a_2.loc[s][a]+alpha*(r+q_s_a_1.loc[s_next][q_s_a_2.loc[s_next].idxmax()]-q_s_a_

    return q_s_a_1, q_s_a_2
```

Double Q-Learning (11p)

```
num_ep=10**5
beg_time=time()
q_s_a_1=deepcopy(q_s_a_init)
q_s_a_2=deepcopy(q_s_a_init)
pi=deepcopy(pi_50)
exploration_rate=1

for epi_i in range(1, num_ep+1):
    s_now='0'
    while s_now!='70':
        sample_step=simul_step(pi, s_now, P_normal, P_speed, R_s_a)
        q_s_a_1, q_s_a_2=pol_eval_dbl_Q(sample_step, q_s_a_1, q_s_a_2, alpha=max(1/epi_i, 0.05))

        if epi_i % 100 == 0:
            pi=pol_imp(pi, q_s_a_1+q_s_a_2, epsilon=exploration_rate)

        s_now=sample_step[3]
        exploration_rate=max(exploration_rate*0.9995, 0.001)

end_time=time()

pd.DataFrame(((q_s_a_1+q_s_a_2)/2), index=states, columns=['n','s']).T
```

```
##           0        10        20        30        40        50       60   70
## n -5.710681 -4.678038 -3.672418 -2.677539 -1.865379 -1.997438 -1.00000  0.0
## s -5.389653 -4.762882 -3.803913 -3.344512 -1.636887 -1.762391 -1.62411  0.0
```

```
print(end_time-beg_time)
```

```
## 959.0306012630463
```

```
pi.T
```

```
##      0   10   20   30   40   50   60   70
## n  0.0  1.0  1.0  1.0  0.0  0.0  1.0  1.0
## s  1.0  0.0  0.0  0.0  1.0  1.0  0.0  0.0
```

Exercise (case : epi_i%500==0)

```
num_ep=10**5
beg_time=time()
q_s_a_1=deepcopy(q_s_a_init)
q_s_a_2=deepcopy(q_s_a_init)
pi=deepcopy(pi_50)
exploration_rate=1

for epi_i in range(1, num_ep+1):
    s_now='0'
    while s_now!='70':
        sample_step=simul_step(pi, s_now, P_normal, P_speed, R_s_a)
        q_s_a_1,q_s_a_2=pol_eval_dbl_Q(sample_step, q_s_a_1, q_s_a_2, alpha=max(1/epi_i, 0.05))

        if epi_i % 500 == 0:
            pi=pol_imp(pi, q_s_a_1+q_s_a_2, epsilon=exploration_rate)

        s_now=sample_step[3]
        exploration_rate=max(exploration_rate*0.9995, 0.001)

end_time=time()

pd.DataFrame(((q_s_a_1+q_s_a_2)/2), index=states, columns=['n','s']).T
```

```
##           0        10        20        30        40        50       60   70
## n -5.630276 -4.533584 -3.603864 -2.590510 -1.747020 -1.939772 -1.00000  0.0
## s -5.375942 -4.764001 -3.884199 -3.225211 -1.679693 -1.664859 -1.49285  0.0
```

```
print(end_time-beg_time)
```

```
## 913.1761200428009
```

```
pi.T
```

```
##      0   10   20   30   40   50   60   70
## n  0.0  1.0  1.0  1.0  0.0  0.0  1.0  1.0
## s  1.0  0.0  0.0  0.0  1.0  1.0  0.0  0.0
```

Exercise (case : alpha=max(n, 0.1))

```
num_ep=10**5
beg_time=time()
q_s_a_1=deepcopy(q_s_a_init)
q_s_a_2=deepcopy(q_s_a_init)
pi=deepcopy(pi_50)
exploration_rate=1

for epi_i in range(1, num_ep+1):
    s_now='0'
    while s_now!='70':
        sample_step=simul_step(pi, s_now, P_normal, P_speed, R_s_a)
        q_s_a_1,q_s_a_2=pol_eval_dbl_Q(sample_step, q_s_a_1, q_s_a_2, alpha=max(1/epi_i, 0.1))


        if epi_i % 100 == 0:
            pi=pol_imp(pi, q_s_a_1+q_s_a_2, epsilon=exploration_rate)

        s_now=sample_step[3]
        exploration_rate=max(exploration_rate*0.9995, 0.001)

end_time=time()

pd.DataFrame(((q_s_a_1+q_s_a_2)/2), index=states, columns=['n','s']).T
```

```
##           0        10        20        30        40        50        60   70
## n -6.001441 -5.000000 -4.000000 -3.000000 -2.000000 -2.000000 -1.000000  0.0
## s -6.037380 -5.007525 -4.128738 -3.754061 -2.213914 -2.107194 -1.855157  0.0
```

```
print(end_time-beg_time)
```

```
## 1083.7254917621613
```

```
pi.T
```

```
##      0   10   20   30   40   50   60   70
## n  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
## s  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```