F2 Python Code

Kang, Eui Hyeon

2021-02-05

차 례

Skiier.py(1) 2
Skiier.py(2) 4
Skiier.py(3) 5
Skiier.py(4) 6
Skiier.py(5) 7
Skiier.py(6) 8
Skiier.py(7) 9

Skiier.py(1)

```
# Model

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
```

```
##      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
```

Skiier.py(2)

```
# Policy

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=['n

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
```

Skiier.py(3)

```python
# simul_path()

def simul_path(pi, P_normal, P_speed, R_s_a):
    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=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

    return history_i

sample_path=simul_path(pi=pi_speed,P_normal=P_normal,P_speed=P_speed,R_s_a=R_s_a)
sample_path
```

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

Skiier.py(4)

```python
# simul_step()

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
```

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

Skiier.py(5)

```
# pol_eval_MC()
from copy import deepcopy

def pol_eval_MC(sample_path, q_s_a, alpha):
    qsa=deepcopy(q_s_a)
    for j in range(0, len(sample_path)-1, 3):
        s=sample_path[j]
        a=sample_path[j+1]
        G=sum([sample_path[g] for g in range(j+2, len(sample_path)-1, 3)])

        qsa.loc[s][a]=q_s_a.loc[s][a]+alpha*(G-q_s_a.loc[s][a])

    return qsa

q_s_a=pol_eval_MC(sample_path=sample_path, q_s_a=q_s_a_init, alpha=0.1)
q_s_a
```

```
##       n     s
## 0   0.0 -0.50
## 10  0.0  0.00
## 20  0.0 -0.35
## 30  0.0  0.00
## 40  0.0 -0.20
## 50  0.0  0.00
## 60  0.0 -0.15
## 70  0.0  0.00
```

Skiier.py(6)

```
# pol_eval_TD()

def pol_eval_TD(sample_step, q_s_a, alpha):
    qsa=deepcopy(q_s_a)

    s=sample_step[0]
    a=sample_step[1]
    r=sample_step[2]
    s_next=sample_step[3]
    a_next=sample_step[4]

    qsa.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 qsa

q_s_a=pol_eval_TD(sample_step, q_s_a_init, alpha=0.1)
q_s_a
```

```
##       n     s
## 0   0.0 -0.15
## 10  0.0  0.00
## 20  0.0  0.00
## 30  0.0  0.00
## 40  0.0  0.00
## 50  0.0  0.00
## 60  0.0  0.00
## 70  0.0  0.00
```

Skiier.py(7)

```
# pol_imp()

def pol_imp(pi, q_s_a, epsilon): # epsilon=exploration_rate
    pi=deepcopy(pi)

    # VERSION 1
    for i in range(0, pi.shape[0]):
        if np.random.uniform(0,1)>epsilon: #exploitation
            cols=['n','s']
            maxVal=q_s_a.iloc[i].idxmax()
            cols.remove(maxVal)

            pi.iloc[i][maxVal]=1
            pi.iloc[i][cols[0]]=0
        else:
            pi.iloc[i]=1/q_s_a.shape[1]
    '''
    # VERSION 2
    # 칼럼이 여러개인 경우 적용방안
    pi=q_s_a.apply(lambda x: (x.rank(method='first', ascending=False)), axis=1).apply(lambda x:(x==x.min())+0

    for i in range(0, pi.shape[0]):
        if np.random.uniform(0,1) < epsilon :
            pi.iloc[i]=1/q_s_a.shape[1]
    '''
    return pi

pi=pol_imp(pi=pi_speed, q_s_a=q_s_a, epsilon=0)
pi.T
```

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