F2

Bongseokkim

2021-02-10

차 례

skiier.py(1)

```
import numpy as np
import pandas as pd

# Model
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( [[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]] ), index = states, columns = states )

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, columns = ['n', 's'] )



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

```
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
```

```
print(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', 's'] )



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
```

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


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
```

skiier.py(3)

```
def simul_path(pi, P_normal, P_speed, R_s_a):
    s_now = "0"
    history_i = [s_now]

    while s_now != "70":
        if np.random.uniform() < 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() )]
        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)
print(sample_path)
```

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

skiier.py(4)

```
# simul_step()
def simul_step(pi, s_now, P_normal, P_speed, R_s_a):
    if np.random.uniform() < 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() )]

    if np.random.uniform() < 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_speed, "0", P_normal, P_speed, R_s_a )
print( sample_step )
```

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

skiier.py(5)

```
def pol_eval_MC(sample_path, q_s_a, alpha):
    q_s_a_copy= q_s_a.copy()

    for j in range( 0,len( sample_path ) - 1, 3 ):
        s = sample_path[j]
        a = sample_path[j + 1]
        G = np.sum(np.array(sample_path[j + 2:len( sample_path )-1:3]).astype( float ) )

        q_s_a_copy.loc[s,a] += alpha * (G - q_s_a_copy.loc[s, a])

    return q_s_a_copy



q_s_a=pol_eval_MC( 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)

```python
def pol_eval_TD(sample_step, q_s_a, alpha):
    q_s_a_copy= q_s_a.copy()
    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_copy.loc[s,a] +=alpha*(r+q_s_a_copy.loc[s_next, a_next]-q_s_a_copy.loc[s,a])

    return q_s_a_copy

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)

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



        else:
            # exploration
            pi_copy.iloc[i] = 1/q_s_a.shape[1]

    return pi_copy

pol_imp(pi_speed, q_s_a, epsilon=0)
```

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

```
"Done "
```

```
## [1] "Done "
```