F2

Bongseokkim

2021-02-21

차 례

skiier.py(1)

```
import numpy as np
import pandas as pd

# Model
action_dict = {0:"n", 1:"s"}
Normal = 0
Speed = 1

states = np.arange(0,80,10)

P_normal = np.array([[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]])

P_speed = np.array([[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]])

R_s_a = 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]]

q_s_a_init = np.c_[np.repeat( 0.0, len( states ) ), np.repeat( 0.0, len( states ) )]
```

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


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


print(pd.DataFrame(pi_50, columns=['n','s'], index =states).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
def simul_path(pi, P_normal,P_speed, R_s_a):

    s_now = 0
    history_i = [str(s_now)]
    while s_now != 70:
        if np.random.uniform() < pi[np.where(states == s_now),Normal] :
            a_now = Normal
            P = P_normal

        else :
            a_now = Speed
            P = P_speed

        r_now = R_s_a[np.where(states == s_now)[0].item(),a_now]
        s_next = states[np.argmin(P[np.where(states == s_now),].cumsum() < np.random.uniform(0,1))]
        history_i.extend([action_dict[a_now], r_now, str(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)

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

    if np.random.uniform() < pi[np.where(states == s_now),Normal]:
        a_now = Normal
        P = P_normal
    else:
        a_now = Speed
        P = P_speed

    r_now = R_s_a[np.where(states == s_now)[0].item(),a_now]
    s_next = states[np.argmin(P[np.where(states == s_now),].cumsum() < np.random.uniform(0,1))]


    if np.random.uniform() < pi[np.where(states == s_next),Normal]:
        a_next = Normal

    else:
        a_next = Speed

    sarsa = [str(s_now), action_dict[a_now], r_now, str(s_next), action_dict[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[np.where(states== int(s)),list(action_dict.values()).index(a)] += alpha * (G - q_s_a_copy[

    return q_s_a_copy



q_s_a=pol_eval_MC( sample_path, q_s_a = q_s_a_init, alpha = 0.1 )
print(pd.DataFrame(q_s_a, columns=['n','s'], index =states))
```

```
##        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)

```
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[np.where(states== int(s)),list(action_dict.values()).index(a)]     +=alpha*(r+q_s_a_copy[np.wh

    return q_s_a_copy

q_s_a=pol_eval_TD(sample_step, q_s_a_init, alpha = 0.1)
print(pd.DataFrame(q_s_a, columns=['n','s'], index =states))
```

```
##       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[i] = 0
            pi_copy[i, np.argmax(q_s_a[i])] =1


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

print(pd.DataFrame(pol_imp(pi_speed, q_s_a, epsilon=0), columns=['n','s'], index =states))
```

```
##     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 "
```