F2 Python

Jaemin Park

2021-02-06

차 례

skiier.R(1)

```
states = np.arange(0,80,10)
P_normal = 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]])
P_speed = 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]])

P_normal = pd.DataFrame(P_normal,states,states)
P_speed = pd.DataFrame(P_speed,states,states)

R_s_a = np.matrix([[-1,-1,-1,-1,0,-1,-1,0],[-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]]).T
R_s_a = pd.DataFrame(R_s_a,states,["n","s"])

q_s_a_init = np.hstack((np.zeros(len(states)).reshape(8,1),np.zeros(len(states)).reshape(8,1)))
q_s_a_init = pd.DataFrame(q_s_a_init,states,["n","s"])
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.R(2)

```
pi_speed = np.hstack((np.zeros(len(states)).reshape(8,1),np.repeat(1,len(states)).reshape(8,1)))
pi_speed = pd.DataFrame(pi_speed,states,["n","s"])
print(pi_speed.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  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0
```

```
pi_50 = np.hstack((np.repeat(0.5,len(states)).reshape(8,1),np.repeat(0.5,len(states)).reshape(8,1)))
pi_50 = pd.DataFrame(pi_50,states,["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.R(3)

```
def simul_path(pi,P_normal,P_speed,R_s_a):
    s_now = 0
    history_i = []
    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([s_now,a_now,r_now])
        s_now = s_next
    return(history_i)

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

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

skiier.R(4)

```
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_speed, 0, P_normal, P_speed, R_s_a) # a.k.a. sarsa
print(sample_step)
```

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

skiier.R(5)

```
def pol_eval_MC(sample_path, q_s_a, alpha):
    for j in range(0,len(sample_path),3):
        s = sample_path[j]
        a = sample_path[j+1]
        G = pd.Series(sample_path)[range(j+2,len(sample_path)-1,3)].astype('float').sum()
        q_s_a.loc[s][a] = q_s_a.loc[s][a] + alpha*(G-q_s_a.loc[s][a])
    return q_s_a
q_s_a = pol_eval_MC(sample_path,q_s_a_init,0.1)
print(q_s_a)
```

```
##        n     s
## 0   0.0 -0.35
## 10  0.0  0.00
## 20  0.0 -0.20
## 30  0.0  0.00
## 40  0.0 -0.05
## 50  0.0  0.00
## 60  0.0  0.00
## 70  0.0  0.00
```

skiier.R(6)

```
def pol_eval_TD(sample_path, q_s_a, alpha):
    s = sample_path[0]
    a = sample_path[1]
    r = float(sample_path[2])
    s_next = sample_path[3]
    a_next = sample_path[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
q_s_a = pol_eval_TD(sample_path,q_s_a_init,0.1)
print(q_s_a)
```

```
##       n      s
## 0   0.0 -0.485
## 10  0.0  0.000
## 20  0.0 -0.200
## 30  0.0  0.000
## 40  0.0 -0.050
## 50  0.0  0.000
## 60  0.0  0.000
## 70  0.0  0.000
```

skiier.R(7)

```
def pol_imp(pi,q_s_a,epsilon):

    for i in range(pi.shape[0]):
        if(np.random.uniform(0,1)>epsilon):
            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)
pi = pol_imp(pi_speed, q_s_a, 0)
print(pi)
```

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