F2 Wonryeol,Jeong

Jeong, wonryeol

2021-02-14

Contents

Preparation

```
import numpy as np
import pandas as pd
gamma = 1
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([[.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)

q_s_a_init = pd.DataFrame(np.zeros((len(states),2)),states,["n","s"])

def transition(given_pi, states, P_normal, P_speed):
    P_out=pd.DataFrame(np.zeros((len(states),len(states))),index=states, columns=states)
```

```
    for s in states:
        action_dist=given_pi.loc[s]
        P=action_dist['normal']*P_normal+action_dist['speed']*P_speed
        P_out.loc[s]=P.loc[s]

    return P_out
```

```
# reword
R_s_a=pd.DataFrame(np.array([-1,-1,-1,-1,0.0,-1,-1,0,
                             -1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]).reshape(len(states),2,order='F'),c
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
```

```
# pi_speed
pi_speed=pd.DataFrame(np.c_[np.repeat(0,len(states)), np.repeat(1,len(states))],index=states, columns=[
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
pi_50=pd.DataFrame(np.repeat(0.5,len(states)*2).reshape(8,2),index=states, columns=['n','s'])

np.cumsum(pi_50.loc['10',])
```

```
## n    0.5
## s    1.0
## Name: 10, dtype: float64
```

```
np.where(pi_50 == .5)
```

```
## (array([0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7]), array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
```

```
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
```

```
np.random.uniform(0,1,1)
```

```
## array([0.70923826])
```

Skiier.R(3)

```
# Skiier.R(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(0,1,1) < 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 = pd.Series(np.cumsum(P.loc[s_now,])<np.random.uniform(0,1)).idxmin()
        history_i.extend([a_now,r_now,s_next])
        s_now = s_next

    return history_i

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

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

Skiier.R(4)

```
# Skiier.R(4)
def simul_step(pi,s_now,P_normal,P_speed,R_s_a):
    if np.random.uniform(0,1,1) < 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 = pd.Series(np.cumsum(P.loc[s_now,])<np.random.uniform(0,1)).idxmin()

    if np.random.uniform(0,1,1) < pi.loc[s_now,"n"] :
        a_next = "n"

    else:
        a_next = "s"

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



sample_path_td = simul_step(pi_speed,'0',P_normal,P_speed,R_s_a)
sample_path_td
```

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

Skiier.R(5)

```
# Skiier.R(5)
## pol_eval_MC()
def pol_eval_MC(sample_path, q_s_a, alpha ):
    Q_s_a = q_s_a.copy()
    for j in range(0,len(sample_path)-1,3):

        s = sample_path[j]

        a = sample_path[j+1]

        G = pd.Series(sample_path)[list(range(j+2,len(sample_path),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,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.R(6)

```
# Skiier.R(6)
## pol_eval_TD()
def pol_eval_TD(sample_path, q_s_a, alpha ):
    Q_s_a = q_s_a.copy()
    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_td,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.R(7)

```
# Skiier.R(7)
def pol_imp(pi,q_s_a,epsilon):
    Pi = pi.copy()
    for i in list(pi.index):
        if np.random.uniform(0,1,1) > epsilon:

            Pi.loc[i] = 0
            Pi.loc[i,np.argmax(q_s_a.loc[i])]=1
            if i == '70':
                print(Pi.loc[i,np.argmax(q_s_a.loc[i])])

        else:
            Pi.loc[i,:] = 1/q_s_a.shape[1]
    return Pi

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

```
## 1.0
```

```
pi
```

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