Lecture F2. MDP without Model 1

Baek, Jong min

2021-02-15

차 례

skiier.R(1)

```
states = np.arange(0,80,10).astype(str)
p_normal = pd.DataFrame(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
]).reshape(8,8),index=states, columns=states)
p_speed = pd.DataFrame(np.array([
.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,
]).reshape(8,8),index=states, columns=states)
```

```
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(8,2,orc
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 = np.vstack([np.zeros(len(states)),np.zeros(len(states))]).T
q_s_a_init = pd.DataFrame(q_s_a_init,index=states,columns=['n','s'])
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 = pd.DataFrame(np.c_[np.zeros(len(states)),np.repeat(1,len(states))],columns=['n','s'],index=states)
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 = pd.DataFrame(np.c_[np.repeat(0.5,len(states)),np.repeat(0.5,len(states))],columns=['n','s'],index=sta
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 = 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(np.cumsum(p.loc[s_now]) < np.random.uniform(0,1))]
    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)
print(sample_path)
```

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

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(np.cumsum(p.loc[s_now]) < np.random.uniform(0,1))]
  if np.random.uniform(0,1) < pi.loc[s_next,'n']:
    a_next = 'n'
  else:
    a_next = 's'
  sarsa = np.hstack([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)
sample_step
```

```
## array(['0', 's', '-1.5', '20', 's'], dtype='<U32')
```

skiier.R(5)

```
def pol_eval_MC(sample_path, q_s_a, alpha):
  for j in range(0,len(sample_path)-1,3):
    s = sample_path[j]
    a = sample_path[j+1]
    G = np.sum(np.asarray(sample_path)[range(j+2,len(sample_path)-1,3)].astype('float'))
    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)

```
q_s_a = np.vstack([np.zeros(len(states)),np.zeros(len(states))]).T
q_s_a = pd.DataFrame(q_s_a,index=states,columns=['n','s'])
```

```
def pol_eval_TD(sample_step,q_s_a,alpha):
  s = sample_step[0]
  a = sample_step[1]
  r = sample_step[2].astype('float')
  s_next = sample_step[3]
  a_next = sample_step[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_step,q_s_a,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)

```
def pol_imp(pi,q_s_a,epsilon):
  for i in range(len(pi_speed)):
    if np.random.uniform(0,1) > epsilon :
      pi.iloc[i] = 0
      pi.iloc[i,np.argmax(q_s_a.iloc[i])] = 1
    else:
      pi.iloc[i] = 1/len(q_s_a.columns)
  return pi

pi = pol_imp(pi_speed,q_s_a,epsilon=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
```

F2.Rmd

```
"Hello"
```

```
## [1] "Hello"
```