Inotes example_손민상

Son Min Sang

2021-01-29

차 례

$\pi : S \rightarrow A$

```
import numpy as np
import pandas as pd

action = ["TL","TR"]
states = ["S1","S2","S3","S4","S5","S6","S7"]

pi_TL=pd.DataFrame(np.c_[np.repeat(1,len(states)),np.repeat(0,len(states))],index=states,columns=action)

pi_TR=pd.DataFrame(np.c_[np.repeat(0,len(states)),np.repeat(1,len(states))],index=states,columns=action)

pi_50=(pi_TL+pi_TR)/2

pi_50
```

```
##      TL   TR
## S1  0.5  0.5
## S2  0.5  0.5
## S3  0.5  0.5
## S4  0.5  0.5
## S5  0.5  0.5
## S6  0.5  0.5
## S7  0.5  0.5
```

$R^{\pi} : S \rightarrow \mathbb{R}$

```
import numpy as np
import pandas as pd

R_s_a = pd.DataFrame(np.array([[1,1,0,0,0,0,0],[0,0,0,0,0,10,10]]).T,columns=["TL","TR"],index=states)
def reward_fn(given_pi):
    R_s_a = pd.DataFrame(np.array([[1,1,0,0,0,0,0],[0,0,0,0,0,10,10]]).T,columns=["TL","TR"],index=states)
    R_pi = np.sum(R_s_a*given_pi, axis=1)
    return R_pi

R_s_a
```

```
##     TL  TR
## S1   1   0
## S2   1   0
## S3   0   0
## S4   0   0
## S5   0   0
## S6   0  10
## S7   0  10
```

$P^{\pi} : S \times A \rightarrow S$

```
P_TL = np.array([
                 [1,0,0,0,0,0,0],
                 [1,0,0,0,0,0,0],
                 [0,1,0,0,0,0,0],
                 [0,0,1,0,0,0,0],
                 [0,0,0,1,0,0,0],
                 [0,0,0,0,1,0,0],
                 [0,0,0,0,0,1,0],
                ])
P_TR = np.array([[0,1,0,0,0,0,0],
                 [0,0,1,0,0,0,0],
                 [0,0,0,1,0,0,0],
                 [0,0,0,0,1,0,0],
                 [0,0,0,0,0,1,0],
                 [0,0,0,0,0,0,1],
                 [0,0,0,0,0,0,1],
```

```
                ])

P_TL
```

```
## array([[1, 0, 0, 0, 0, 0, 0],
##        [1, 0, 0, 0, 0, 0, 0],
##        [0, 1, 0, 0, 0, 0, 0],
##        [0, 0, 1, 0, 0, 0, 0],
##        [0, 0, 0, 1, 0, 0, 0],
##        [0, 0, 0, 0, 1, 0, 0],
##        [0, 0, 0, 0, 0, 1, 0]])
```

```
import numpy as np
import pandas as pd

def transition(given_pi,states, P_TL, P_TR):
    P_out = pd.DataFrame(np.zeros(shape=(len(states),len(states))))

    for s in range(len(states)):
        action_dist = given_pi.iloc[s,:]

        P= action_dist['TL']*P_TL + action_dist['TR']*P_TR
        P_out[s]=P[:,s]

    return P_out
transition(pi_TL, states=states, P_TL=P_TL, P_TR=P_TR)
```

```
##    0  1  2  3  4  5  6
## 0  1  0  0  0  0  0  0
## 1  1  0  0  0  0  0  0
## 2  0  1  0  0  0  0  0
## 3  0  0  1  0  0  0  0
## 4  0  0  0  1  0  0  0
## 5  0  0  0  0  1  0  0
## 6  0  0  0  0  0  1  0
```

$R^{\pi}S \rightarrow R$

```
R_s_a = pd.DataFrame(np.array([[1,1,0,0,0,0,0],[0,0,0,0,0,10,10]]).T,columns=["TL","TR"],index=states)

def reward_fn(given_pi):
```

```
    R_s_a = pd.DataFrame(np.array([[1,1,0,0,0,0,0],[0,0,0,0,0,10,10]]).T,columns=["TL","TR"],index=states)
    R_pi = np.sum(R_s_a*given_pi, axis=1)

    return R_pi
reward_fn(pi_TR)
```

```
## S1     0
## S2     0
## S3     0
## S4     0
## S5     0
## S6    10
## S7    10
## dtype: int64
```

Policy evaluation

```
def policy_eval(given_pi,gamma=0.99):
    R = reward_fn(given_pi).values.reshape(7,1)
    P = transition(given_pi,states, P_TL = P_TL, P_TR = P_TR)
    gamma = gamma
    epsilon = 10**(-8)

    v_old=np.repeat(0,7).reshape(7,1)
    v_new = R+gamma*np.dot(P, v_old)

    while np.max(np.abs(v_new-v_old))>epsilon:
        v_old=v_new
        v_new=R+np.dot(gamma*P,v_old)


    return v_new
```

```
policy_eval(pi_TL, gamma=0.9).astype(int)
```

```
## array([[9],
##        [9],
##        [8],
##        [8],
##        [7],
##        [6],
```

```
##        [5]])
```

```
policy_eval(pi_TL, gamma=0).astype(int)
```

```
## array([[1],
##        [1],
##        [0],
##        [0],
##        [0],
##        [0],
##        [0]])
```

```
policy_eval(pi_TL, gamma=0.1).astype(int)
```

```
## array([[1],
##        [1],
##        [0],
##        [0],
##        [0],
##        [0],
##        [0]])
```