E1_Exercises

Kwon do yun

2021-01-22

차 례

P. 21

```
import numpy as np
import pandas as pd

R=np.hstack((np.repeat(-1.5,4),-0.5,np.repeat(-1.5,2),0)).reshape(-1,1)
states=np.arange(0,70+10,step=10)
P=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]])

P=pd.DataFrame(P,columns=states)
```

```
print(R)
```

```
## [[-1.5]
##  [-1.5]
##  [-1.5]
##  [-1.5]
##  [-0.5]
##  [-1.5]
##  [-1.5]
##  [ 0. ]]
```

```
print(P)
```

```
##     0    10   20   30   40   50   60   70
## 0  0.1  0.0  0.9  0.0  0.0  0.0  0.0  0.0
## 1  0.1  0.0  0.0  0.9  0.0  0.0  0.0  0.0
## 2  0.0  0.1  0.0  0.0  0.9  0.0  0.0  0.0
## 3  0.0  0.0  0.1  0.0  0.0  0.9  0.0  0.0
## 4  0.0  0.0  0.0  0.1  0.0  0.0  0.9  0.0
## 5  0.0  0.0  0.0  0.0  0.1  0.0  0.0  0.9
## 6  0.0  0.0  0.0  0.0  0.0  0.1  0.0  0.9
## 7  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

```
gamma=1.0
epsilon=10**(-8)

v_old=np.array(np.zeros(8,)).reshape(8,1)
v_new=R+np.dot(gamma*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)
print(v_new.T)
```

```
## [[-5.80592905 -5.2087811  -4.13926239 -3.47576467 -2.35376031 -1.73537603
##   -1.6735376   0.        ]]
```

Rewritten with intermediate saving

```
R=np.hstack((np.repeat(-1.5,4),-0.5,np.repeat(-1.5,2),0)).reshape(-1,1)
states=np.arange(0,70+10,step=10)
P=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]])
P=pd.DataFrame(P,columns=states)
gamma=1.0
epsilon=10**(-8)
v_old=np.array(np.zeros(8,)).reshape(8,1)
v_new=R+np.dot(gamma*P,v_old)
results=v_old.T
results=np.vstack((results,v_new.T))
while np.max(np.abs(v_new-v_old)) > epsilon:
    v_old=v_new
    v_new=R+np.dot(gamma*P, v_old)
    results=np.vstack((results,v_new.T))

results=pd.DataFrame(results, columns=states)
results.head()
```

```
##       0       10      20     30     40      50     60   70
## 0  0.000  0.0000  0.0000  0.000  0.000  0.0000  0.000  0.0
## 1 -1.500 -1.5000 -1.5000 -1.500 -0.500 -1.5000 -1.500  0.0
## 2 -3.000 -3.0000 -2.1000 -3.000 -2.000 -1.5500 -1.650  0.0
## 3 -3.690 -4.5000 -3.6000 -3.105 -2.285 -1.7000 -1.655  0.0
## 4 -5.109 -4.6635 -4.0065 -3.390 -2.300 -1.7285 -1.670  0.0
```

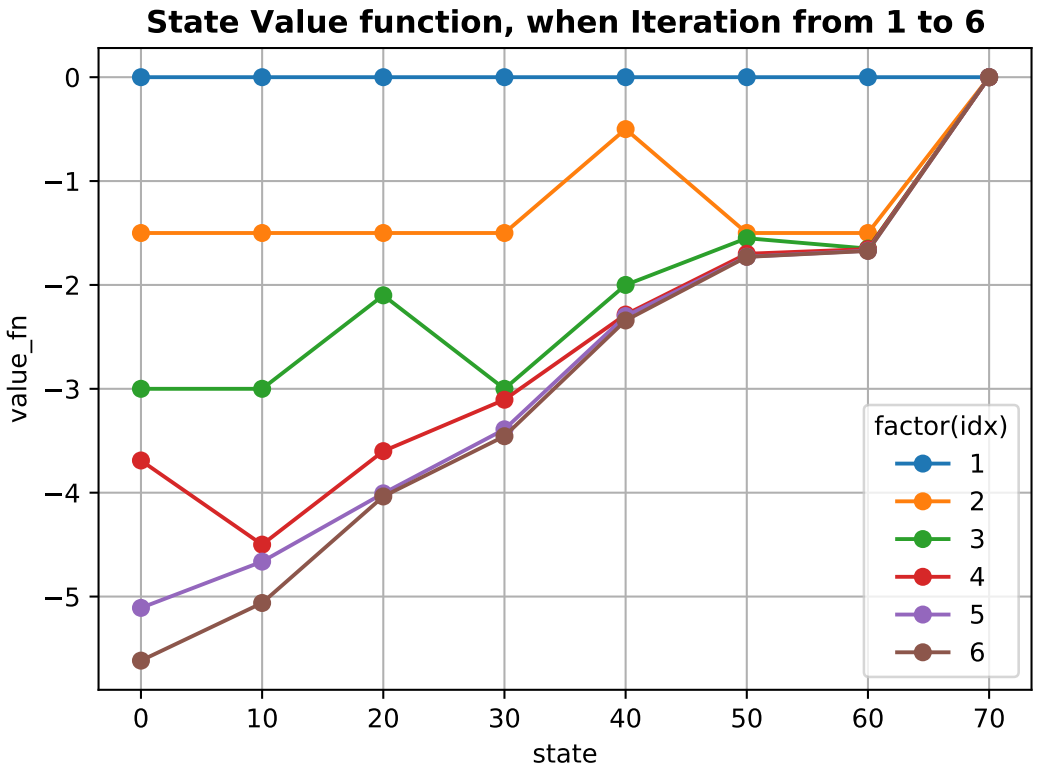
```
results.tail()
```

```
##           0         10        20        30       40        50        60   70
## 18 -5.805929 -5.208781 -4.139262 -3.475765 -2.35376 -1.735376 -1.673538  0.0
## 19 -5.805929 -5.208781 -4.139262 -3.475765 -2.35376 -1.735376 -1.673538  0.0
## 20 -5.805929 -5.208781 -4.139262 -3.475765 -2.35376 -1.735376 -1.673538  0.0
## 21 -5.805929 -5.208781 -4.139262 -3.475765 -2.35376 -1.735376 -1.673538  0.0
## 22 -5.805929 -5.208781 -4.139262 -3.475765 -2.35376 -1.735376 -1.673538  0.0
```

Plots iteration 1 to 6

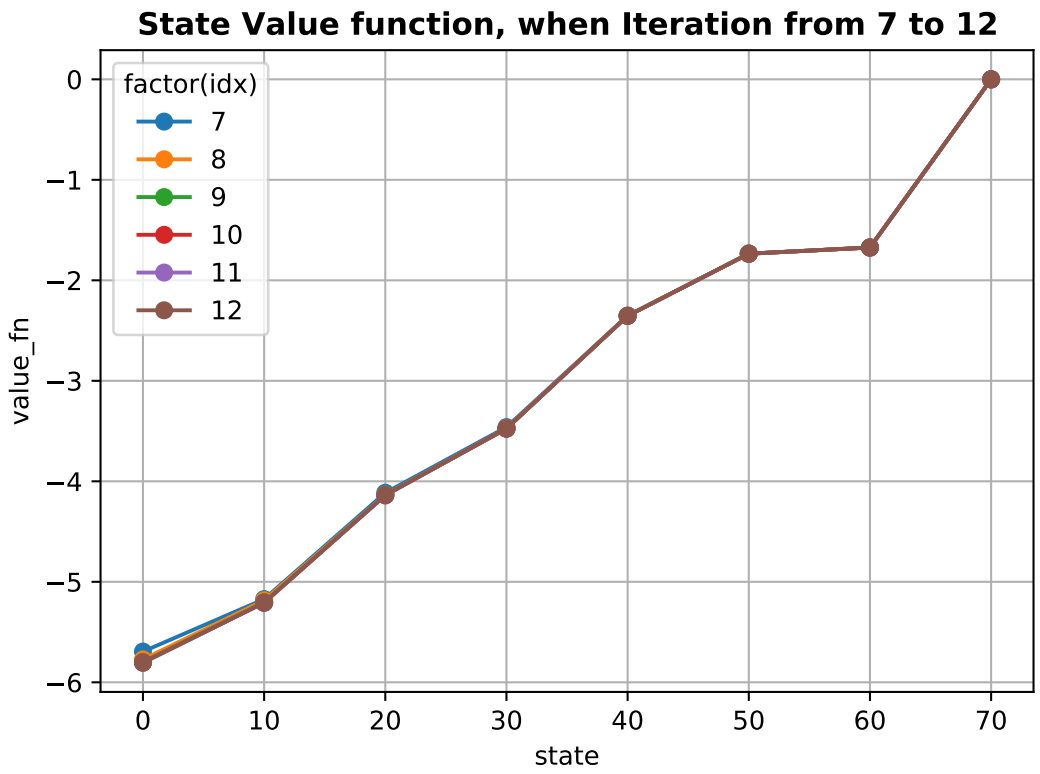
```
import matplotlib.pyplot as plt


plt.plot(states,results.iloc[0],marker='o',label='1')
plt.plot(states,results.iloc[1],marker='o',label='2')
plt.plot(states,results.iloc[2],marker='o',label='3')
plt.plot(states,results.iloc[3],marker='o',label='4')
plt.plot(states,results.iloc[4],marker='o',label='5')
plt.plot(states,results.iloc[5],marker='o',label='6')
plt.grid(True)
plt.legend(title='factor(idx)')
plt.xlabel('state')
plt.ylabel('value_fn')
plt.title('State Value function, when Iteration from 1 to 6',fontweight='bold')
plt.show()
```



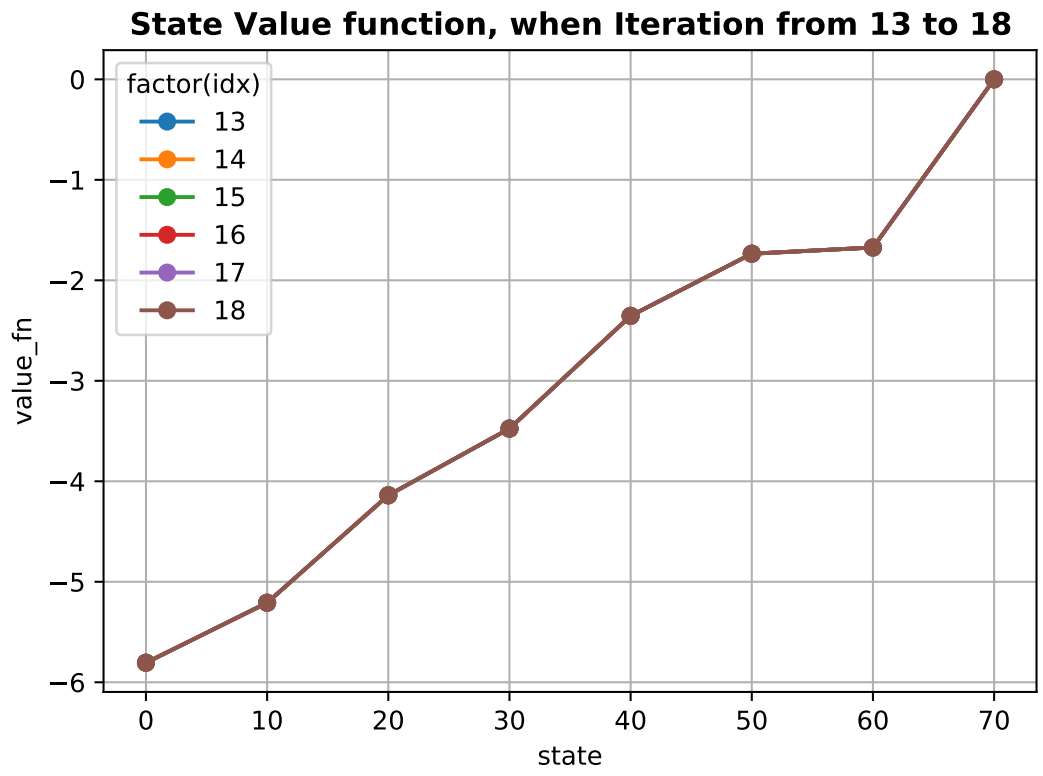
Plots iteration 7 to 12

```
plt.plot(states,results.iloc[6],marker='o',label='7')
plt.plot(states,results.iloc[7],marker='o',label='8')
plt.plot(states,results.iloc[8],marker='o',label='9')
plt.plot(states,results.iloc[9],marker='o',label='10')
plt.plot(states,results.iloc[10],marker='o',label='11')
plt.plot(states,results.iloc[11],marker='o',label='12')
plt.grid(True)
plt.legend(title='factor(idx)')
plt.xlabel('state')
plt.ylabel('value_fn')
plt.title('State Value function, when Iteration from 7 to 12',fontweight='bold')
plt.show()
```



Plots iteration 13 to 18

```
plt.plot(states,results.iloc[12],marker='o',label='13')
plt.plot(states,results.iloc[13],marker='o',label='14')
plt.plot(states,results.iloc[14],marker='o',label='15')
plt.plot(states,results.iloc[15],marker='o',label='16')
plt.plot(states,results.iloc[16],marker='o',label='17')
plt.plot(states,results.iloc[17],marker='o',label='18')
plt.grid(True)
plt.legend(title='factor(idx)')
plt.xlabel('state')
plt.ylabel('value_fn')
plt.title('State Value function, when Iteration from 13 to 18',fontweight='bold')
plt.show()
```




```
"E1_Exercises"
```