E3 Python Code

Kang, Eui Hyeon

2021-01-21

차 례

Preparation

```
gamma=1
states=np.arange(0,70+10,10).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_normal
```

```
##     0  10  20  30  40  50  60  70
## 0   0   1   0   0   0   0   0   0
## 10  0   0   1   0   0   0   0   0
## 20  0   0   0   1   0   0   0   0
## 30  0   0   0   0   1   0   0   0
## 40  0   0   0   0   0   1   0   0
## 50  0   0   0   0   0   0   1   0
## 60  0   0   0   0   0   0   0   1
## 70  0   0   0   0   0   0   0   1
```

```
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)

P_speed
```

```
##       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
## 10  0.1  0.0  0.0  0.9  0.0  0.0  0.0  0.0
## 20  0.0  0.1  0.0  0.0  0.9  0.0  0.0  0.0
## 30  0.0  0.0  0.1  0.0  0.0  0.9  0.0  0.0
```

```
## 40  0.0  0.0  0.0  0.1  0.0  0.0  0.9  0.0
## 50  0.0  0.0  0.0  0.0  0.1  0.0  0.0  0.9
## 60  0.0  0.0  0.0  0.0  0.0  0.1  0.0  0.9
## 70  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0
```

```
R_s_a=pd.DataFrame(np.matrix([-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(stat

R_s_a
```

```
##     normal  speed
## 0     -1.0   -1.5
## 10    -1.0   -1.5
## 20    -1.0   -1.5
## 30    -1.0   -1.5
## 40     0.0   -0.5
## 50    -1.0   -1.5
## 60    -1.0   -1.5
## 70     0.0    0.0
```

Implementation

1. Initialize V

```
# 1. Initialize V

V_old=pd.DataFrame(np.repeat(0,len(states)).reshape(len(states),1),index=states)
V_old.T
```

```
##    0  10  20  30  40  50  60  70
## 0  0   0   0   0   0   0   0   0
```

2. Evaluate the Q-function

```
# 2. Evaluate the Q-function

q_s_a=R_s_a+np.c_[np.dot(gamma*P_normal,V_old),np.dot(gamma*P_speed,V_old)]

q_s_a
```

```
##     normal  speed
## 0     -1.0   -1.5
## 10    -1.0   -1.5
## 20    -1.0   -1.5
## 30    -1.0   -1.5
## 40     0.0   -0.5
## 50    -1.0   -1.5
## 60    -1.0   -1.5
## 70     0.0    0.0
```

3. Find the best action for each state

```
# 3. Find the best action for each state

V_new=np.matrix(q_s_a.apply(max,axis=1)).reshape(len(states),1)

V_new.T
```

```
## matrix([[-1., -1., -1., -1.,  0., -1., -1.,  0.]])
```

Value Iteration - Implementation

```
# Assigned are gamma, states, P_normal, P_speed, R_s_a

cnt=0
epsilon=10**(-8)

V_old=pd.DataFrame(np.repeat(0,len(states)).reshape(len(states),1),index=states)
results=V_old.T

while True:
    q_s_a=R_s_a+np.c_[np.dot(gamma*P_normal,V_old),np.dot(gamma*P_speed,V_old)]
    V_new=np.matrix(q_s_a.apply(max,axis=1)).reshape(len(states),1)

    if np.max(np.abs(V_new-V_old)).item() < epsilon :
        break

    results=np.r_[results, V_new.T]
    V_old=V_new

    cnt+=1

value_iter_process=results
results=pd.DataFrame(results, columns=states)

results.head(6)
```

```
##      0   10    20   30    40    50   60   70
## 0  0.0  0.0  0.00  0.0  0.00  0.00  0.0  0.0
## 1 -1.0 -1.0 -1.00 -1.0  0.00 -1.00 -1.0  0.0
## 2 -2.0 -2.0 -1.60 -1.0 -1.00 -1.50 -1.0  0.0
## 3 -3.0 -2.6 -2.00 -2.0 -1.50 -1.60 -1.0  0.0
## 4 -3.6 -3.0 -3.00 -2.5 -1.60 -1.65 -1.0  0.0
## 5 -4.0 -4.0 -3.24 -2.6 -1.65 -1.66 -1.0  0.0
```

```
results.tail(6)
```

```
##            0        10        20        30        40        50   60   70
## 16 -5.107742 -4.410773 -3.441077 -2.666666 -1.666667 -1.666667 -1.0  0.0
## 17 -5.107743 -4.410774 -3.441077 -2.666667 -1.666667 -1.666667 -1.0  0.0
## 18 -5.107744 -4.410774 -3.441077 -2.666667 -1.666667 -1.666667 -1.0  0.0
## 19 -5.107744 -4.410774 -3.441077 -2.666667 -1.666667 -1.666667 -1.0  0.0
```

```
## 20 -5.107744 -4.410774 -3.441077 -2.666667 -1.666667 -1.666667 -1.0  0.0
## 21 -5.107744 -4.410774 -3.441077 -2.666667 -1.666667 -1.666667 -1.0  0.0
```

Visualization

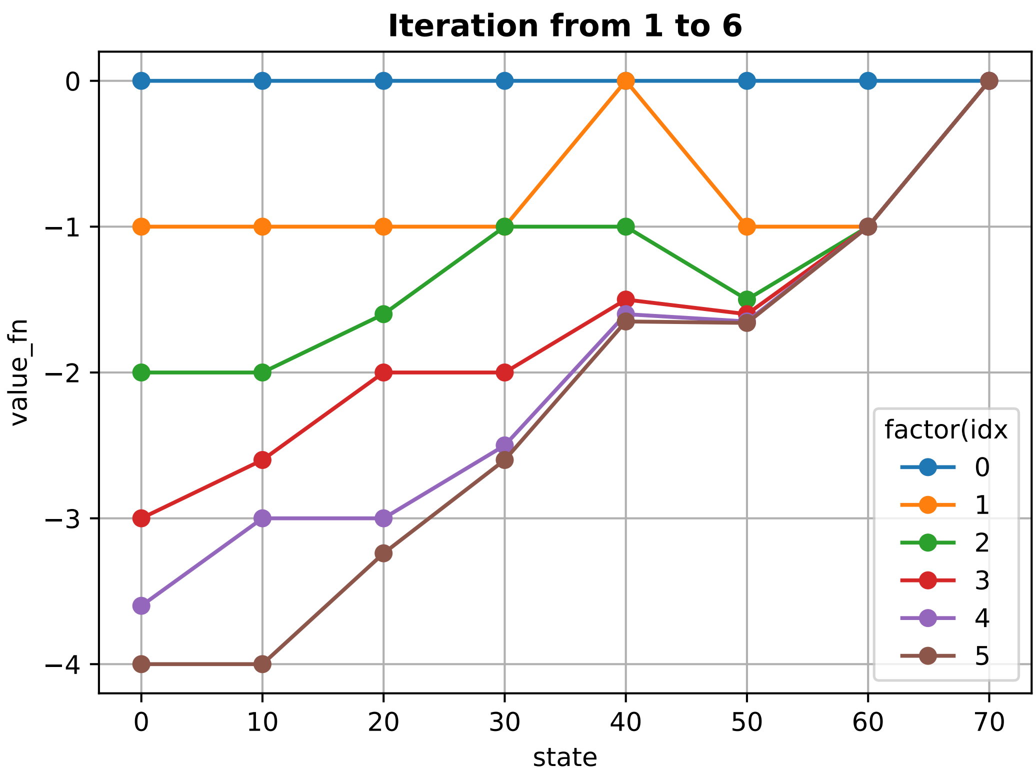
1. Iteration from 6 to 12

```
for i in range(6):
    plt.plot(results.columns,results.iloc[i], label=i,marker='o')

plt.grid(True)
plt.legend(title='factor(idx')
plt.xlabel('state')
plt.ylabel('value_fn')
plt.title('Iteration from 1 to 6', fontweight='bold')
plt.yticks([0,-1,-2,-3,-4])
```

```
## ([<matplotlib.axis.YTick object at 0x000000002C96DCF8>, <matplotlib.axis.YTick object at 0x000000002C96D8D0>
```

```
plt.show()
```



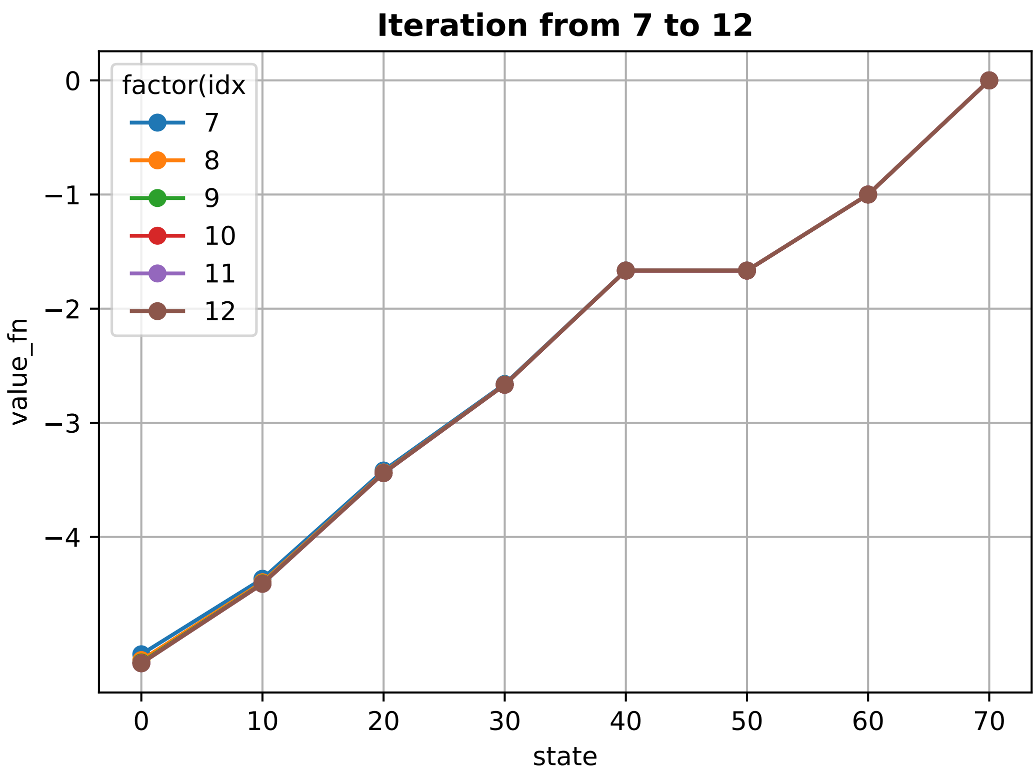
2. Iteration from 7 to 12

```
for i in range(7,13):
    plt.plot(results.columns,results.iloc[i], label=i,marker='o')

plt.grid(True)
plt.legend(title='factor(idx')
plt.xlabel('state')
plt.ylabel('value_fn')
plt.title('Iteration from 7 to 12', fontweight='bold')
plt.yticks([0,-1,-2,-3,-4])
```

```
## ([<matplotlib.axis.YTick object at 0x000000002DBB85F8>, <matplotlib.axis.YTick object at 0x000000002DBB81D0>,
```

```
plt.show()
```



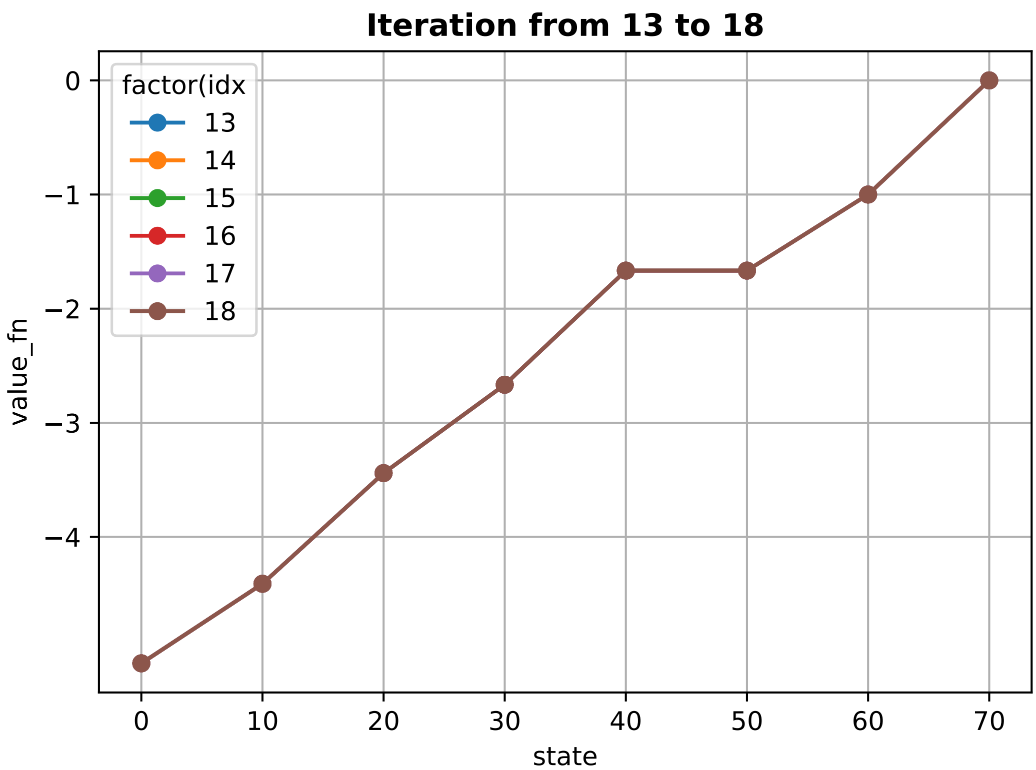
3. Iteration from 13 to 18

```
for i in range(13,19):
    plt.plot(results.columns,results.iloc[i], label=i,marker='o')

plt.grid(True)
plt.legend(title='factor(idx')
plt.xlabel('state')
plt.ylabel('value_fn')
plt.title('Iteration from 13 to 18', fontweight='bold')
plt.yticks([0,-1,-2,-3,-4])
```

```
## ([<matplotlib.axis.YTick object at 0x000000002C963550>, <matplotlib.axis.YTick object at 0x000000002C963D68>
```

```
plt.show()
```



Optimal Value function → Optimal policy

```
V_opt=results.tail(1).T
V_opt.T
```

```
##              0         10         20         30         40         50    60    70
## 21 -5.107744 -4.410774 -3.441077 -2.666667 -1.666667 -1.666667 -1.0  0.0
```

```
q_s_a=R_s_a+np.c_[np.dot(gamma*P_normal,V_opt), np.dot(gamma*P_speed, V_opt)]
q_s_a
```

```
##       normal     speed
## 0   -5.410774 -5.107744
## 10 -4.441077 -4.410774
## 20 -3.666667 -3.441077
## 30 -2.666667 -3.344108
## 40 -1.666667 -1.666667
## 50 -2.000000 -1.666667
## 60 -1.000000 -1.666667
## 70  0.000000  0.000000
```

```
pi_opt_vec=q_s_a.idxmax(axis=1)
pi_opt_vec
```

```
## 0      speed
## 10     speed
## 20     speed
## 30    normal
## 40    normal
## 50     speed
## 60    normal
## 70    normal
## dtype: object
```

```
pi_opt=pd.DataFrame(np.zeros((len(states),2)), index=states, columns=['normal','speed'])

for i in range(len(pi_opt_vec)):
    pi_opt.iloc[i][pi_opt_vec[i]]=1

pi_opt.T
```

```
##            0    10    20    30    40    50    60    70
```

```
## normal  0.0  0.0  0.0  1.0  1.0  0.0  1.0  1.0
## speed   1.0  1.0  1.0  0.0  0.0  1.0  0.0  0.0
```