E3_MDP Python

Jaemin Park

2021-01-22

차 례

p.7 Preparation

```
gamma = 1
states = np.arange(0,80,10)
P_normal = 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]])
P_speed = np.matrix([[0.1,0,0.9,0,0,0,0,0],[0.1,0,0,0.9,0,0,0,0],
[0,0.1,0,0,0.9,0,0,0],[0,0,0.1,0,0,0.9,0,0],[0,0,0,0.1,0,0,0.9,0],
[0,0,0,0,0.1,0,0,0.9],[0,0,0,0,0,0.1,0,0.9],[0,0,0,0,0,0,0,1]])

R_s_a = np.matrix([[-1,-1,-1,-1,0,-1,-1,0],[-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0]]).T
R_s_a = pd.DataFrame(R_s_a,states,["normal","speed"])
```

p.8 Implementation

1. Initialize V

```
V_old = np.zeros(len(states)).T
V_old = pd.DataFrame(V_old,states)
print(V_old.T)
```

```
##      0    10    20    30    40    50    60    70
## 0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
```

2. Evaluate the Q-function

```
q_s_a=R_s_a+np.hstack((np.dot(gamma*P_normal,V_old),np.dot(gamma*P_speed,V_old)))
print(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

```
V_new = np.matrix(q_s_a.apply(max,axis=1)).T
V_new = pd.DataFrame(V_new,states)
print(V_new.T)
```

```
##      0    10    20    30    40    50    60    70
## 0 -1.0 -1.0 -1.0 -1.0   0.0 -1.0 -1.0   0.0
```

p.11 Implementation

```
cnt = 0
epsilon = 10**(-8)
V_old = np.zeros(len(states)).T
V_old = pd.DataFrame(V_old,states)
results = V_old.T
while True:
    q_s_a=R_s_a+np.hstack((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)).T
    V_new = pd.DataFrame(V_new,states)
    if(np.linalg.norm(V_new-V_old)<epsilon):
        break
    results = np.vstack((results,V_new.T))
    V_old = V_new
    cnt+=1
results = pd.DataFrame(results,None,states)
print(results.head())
```

```
##      0    10    20    30    40     50    60    70
## 0  0.0   0.0   0.0   0.0   0.0   0.00   0.0   0.0
## 1 -1.0  -1.0  -1.0  -1.0   0.0  -1.00  -1.0   0.0
## 2 -2.0  -2.0  -1.6  -1.0  -1.0  -1.50  -1.0   0.0
## 3 -3.0  -2.6  -2.0  -2.0  -1.5  -1.60  -1.0   0.0
## 4 -3.6  -3.0  -3.0  -2.5  -1.6  -1.65  -1.0   0.0
```

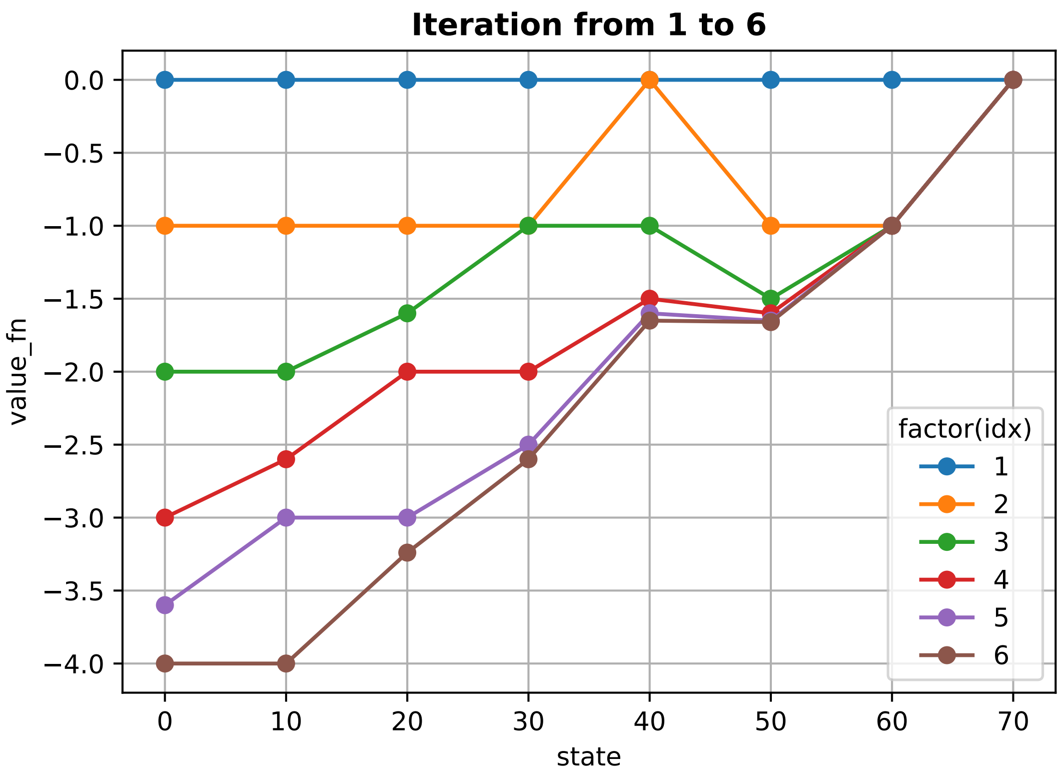
```
print(results.tail())
```

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

p. 13 Visualization

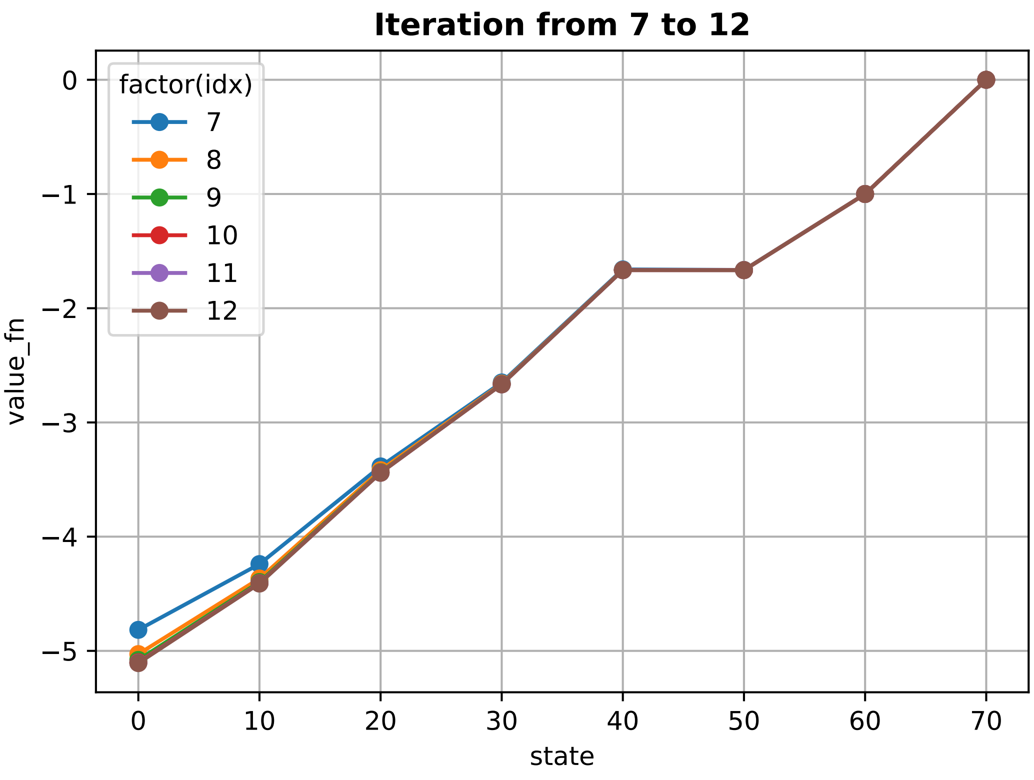
1. Iteration from 6 to 12

```
for i in range(0,6):
    plt.plot(states,results.iloc[i],marker='o',label=i+1)
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.show()
```



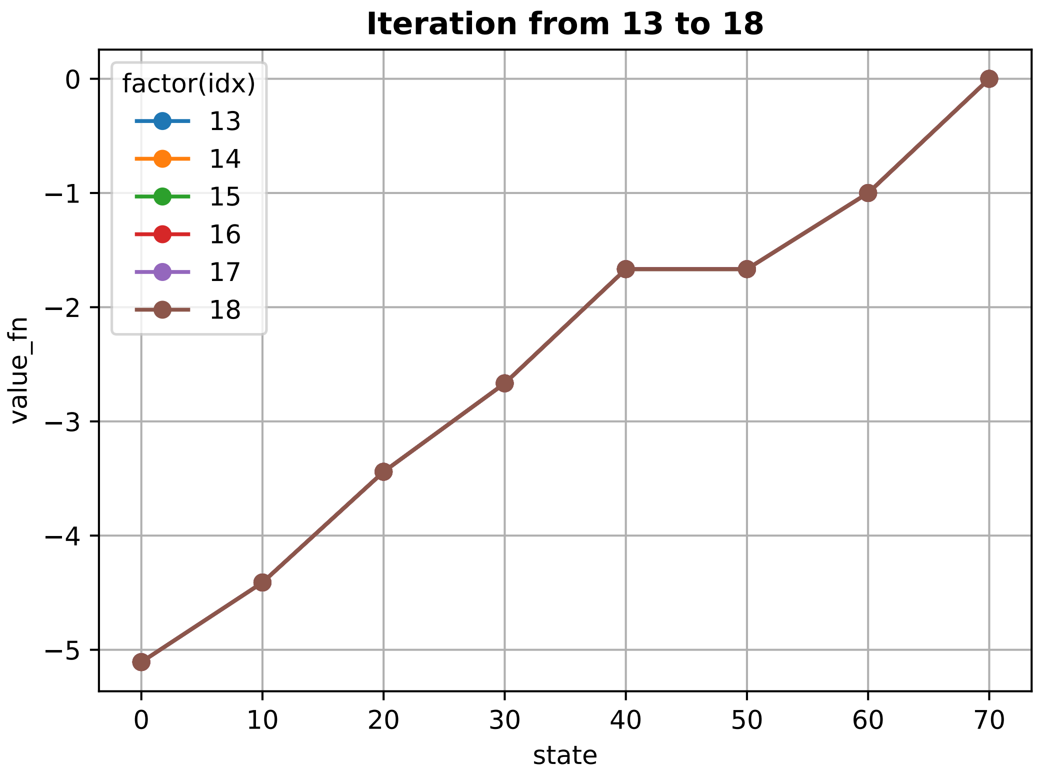
2. Iteration from 7 to 12

```
for i in range(6,12):
    plt.plot(states,results.iloc[i],marker='o',label=i+1)
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.show()
```



3. Iteration from 13 to 18

```
for i in range(12,18):
    plt.plot(states,results.iloc[i],marker='o',label=i+1)
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.show()
```



p.18 Optimal Value function -> Optimal policy

```
V_opt = results.tail(1).T
print(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.hstack((np.dot(gamma*P_normal,V_opt), np.dot(gamma*P_speed, V_opt)))
print(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=pd.DataFrame(np.zeros((len(states),2)), states, ['normal','speed'])

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

print(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
```