E1_jeong,wonryeol

Jeong, wonryeol

1/17/2021

Contents

21_page

```
import numpy as np
import pandas as pd
R_speed = np.array([-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0])
states = range(0,80,10)

P_speed = np.array([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]).reshape(8,8)
```

```
P_speed = pd.DataFrame(P_speed,columns = states)
gamma=1.0
epsilon=10**(-8)
v_old=np.zeros(8)
v_new=R_speed+np.dot(gamma*P_speed,v_old)
while np.max(np.abs(v_new-v_old))>epsilon:
    v_old=v_new
    v_new=R_speed+np.dot(gamma*P_speed, v_old)
print(v_new.T)
```

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

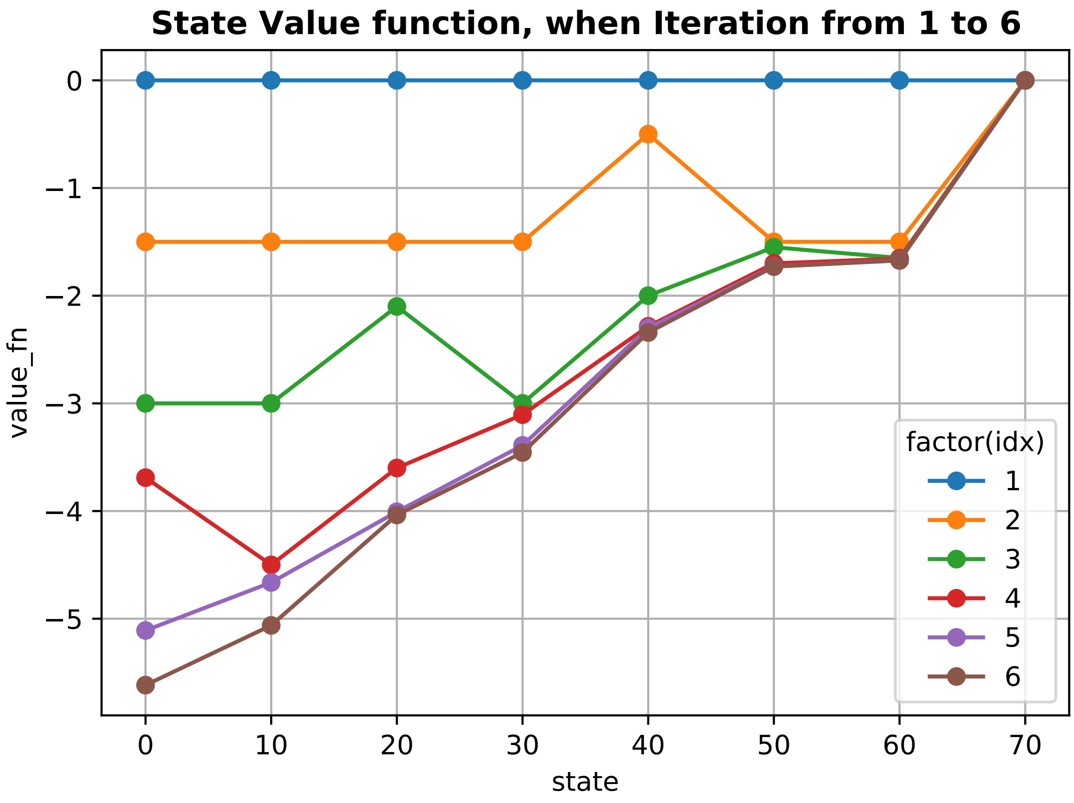
22_page

```
gamma=1.0
epsilon=10**(-8)
v_old=np.zeros(8)
v_new=R_speed+np.dot(gamma*P_speed,v_old)

results = np.vstack((v_old.T,v_new.T))
while np.max(np.abs(v_new-v_old))>epsilon:
    v_old=v_new
    v_new=R_speed+np.dot(gamma*P_speed, 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
```

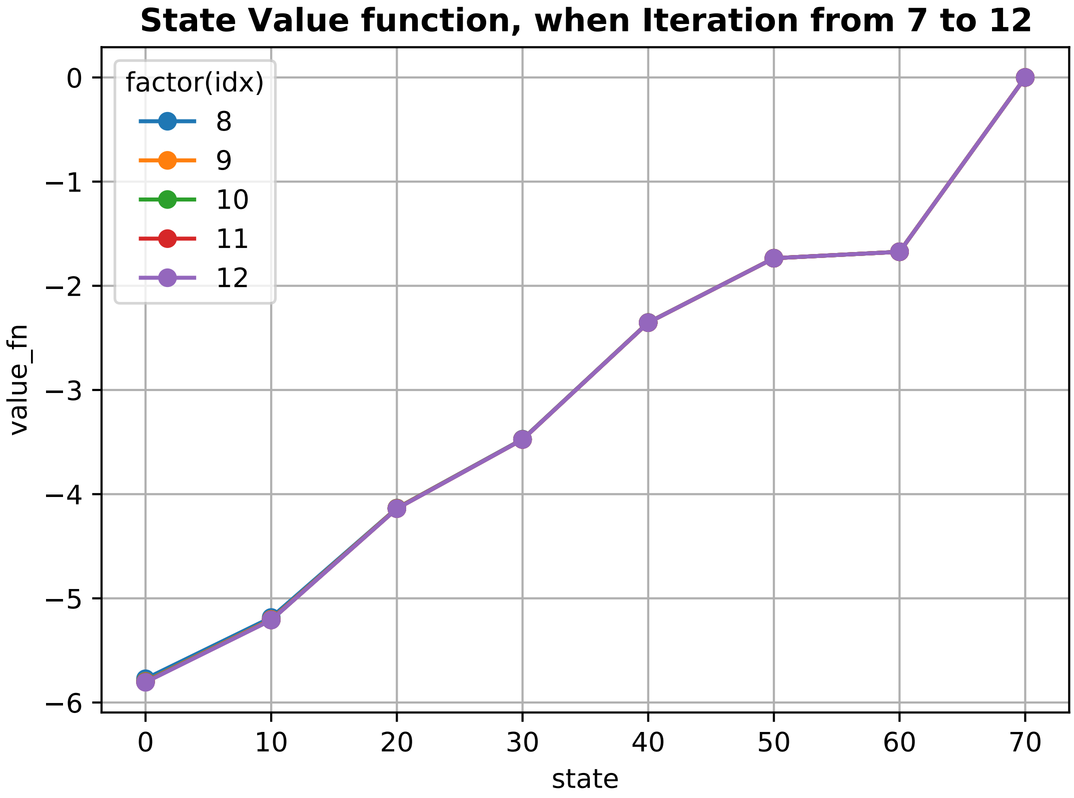
Iteration from 1 to 6

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



plt.rcParams["figure.figsize"] = (14,14)
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()
```



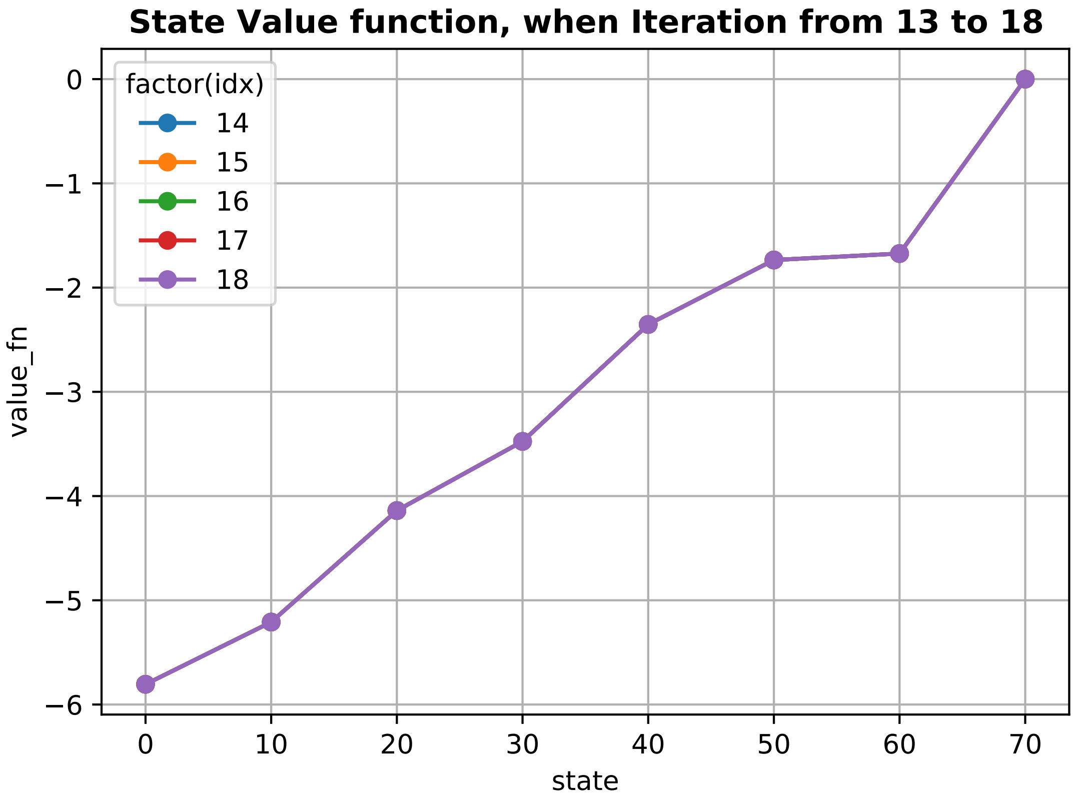
Iteration from 7 to 12

```python
import matplotlib.pyplot as plt
for i in range(7,12):
    plt.plot(states, results.iloc[i], marker='o', label=str(i+1))


plt.rcParams["figure.figsize"] = (14,14)
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()
```



Iteration from 13 to 18

```
import matplotlib.pyplot as plt
for i in range(13,18):
    plt.plot(states, results.iloc[i], marker='o', label=str(i+1))

plt.rcParams["figure.figsize"] = (14,14)
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()
```



For pi__50

```
P_normal = 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)
states = range(0,80,10)
```

```

pi_50 = np.array([0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5]).reshape(8,2)

pi_50
```

```
## array([[0.5, 0.5],
##        [0.5, 0.5],
##        [0.5, 0.5],
##        [0.5, 0.5],
##        [0.5, 0.5],
##        [0.5, 0.5],
##        [0.5, 0.5],
##        [0.5, 0.5]])
```

Page__33 transition

```

def transition(given_pi,X_normal,X_speed):
    X_out = np.zeros(X_normal.shape)

    for i in range(0,given_pi.shape[1]):


        X_out[i] = given_pi[0][i]*X_normal[i] + given_pi[1][i]*X_speed[i]

    return X_out
```

Page_34 test1

```
given_pi = np.array([0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1]).reshape(2,8)

Pi_speed = transition(given_pi,P_normal,P_speed.values)
Pi_speed
```

```
## array([[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. ]])
```

Page_35 test2

```
given_pi = np.array([0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5]).reshape(2,8)


P_50 = transition(given_pi,P_normal,P_speed.values)
P_50
```

```
## array([[0.05, 0.5 , 0.45, 0.  , 0.  , 0.  , 0.  , 0.  ],
##        [0.05, 0.  , 0.5 , 0.45, 0.  , 0.  , 0.  , 0.  ],
##        [0.  , 0.05, 0.  , 0.5 , 0.45, 0.  , 0.  , 0.  ],
##        [0.  , 0.  , 0.05, 0.  , 0.5 , 0.45, 0.  , 0.  ],
##        [0.  , 0.  , 0.  , 0.05, 0.  , 0.5 , 0.45, 0.  ],
##        [0.  , 0.  , 0.  , 0.  , 0.05, 0.  , 0.5 , 0.45],
##        [0.  , 0.  , 0.  , 0.  , 0.  , 0.05, 0.  , 0.95],
##        [0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 1.  ]])
```

Page_39 Implementation, finally

```
def policy_eval (given_pi):
    R_normal = np.array([-1,-1,-1,-1,0,-1,-1,0])
    R_speed = np.array([-1.5,-1.5,-1.5,-1.5,-0.5,-1.5,-1.5,0])

    P_normal = 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)
    P_speed = np.array([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]).reshape(8,8)


    R_50 = transition(given_pi,R_normal,R_speed)

    P_50 = transition(given_pi,P_normal,P_speed)
    gamma=1.0
    epsilon=10**(-8)
    v_old=np.zeros(8)
    v_new=R_50+np.dot(gamma*P_50,v_old)

    results = np.vstack((v_old.T,v_new.T))
    while np.max(np.abs(v_new-v_old))>epsilon:
        v_old=v_new
        v_new=R_50+np.dot(gamma*P_50, v_old)


    return v_new.T
```

```
pi_50 = np.array([0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5]).reshape(2,8)

policy_eval(pi_50)
```

```
## array([-5.96923786, -5.13359222, -4.11995525, -3.38922824, -2.04147003,
##        -2.02776769, -1.35138838,  0.        ])
```