

Lecture D2. Markov Reword Process2

Reinforcement learning Study

2021-02-21

차 례

Method 3- Analytic Solution p.17	2
Method4 - iterative solution - by fixed point	3
result plot	5
After 50 steps (coke only)	6
After 100 steps (coke only)	7

Method 3- Analytic Solution p.17

```
import numpy as np
P = np.array([[0.7,0.3],[0.5,0.5]])
R = np.array([1.5,1])[:, None] # return Column vector in 1D
gamma =0.9
v = np.dot(np.linalg.inv(np.identity(2)-gamma*P),R) #  $v = (I - \gamma P)^{-1} R$ 
print(v)
```

```
## [[13.35365854]
```

```
##  [12.74390244]]
```

Method4 - iterative solution - by fixed point

Implementation

```
import numpy as np
import pandas as pd

R = np.array([1.5,1])[ :, None] # return Column vector in 1D
P = np.array([[0.7,0.3],[0.5,0.5]])
gamma =0.9
epsilon = 10**(-8)
v_old = np.repeat(0,2)[ :,None]

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

print(v_new)

## [[13.35365845]
##  [12.74390235]]
```

The full iteration process

```
import numpy as np
import pandas as pd

R = np.array([1.5,1])[:, None] # return Column vector in 1D
P = np.array([[0.7,0.3],[0.5,0.5]])
gamma =0.9
epsilon = 10**(-8)
v_old = np.repeat(0,2)[:,None]

result=[]
while True:
    result.append(v_old.T)
    v_new =R+gamma*np.dot(P,v_old)
    if np.max(np.abs(v_new-v_old))> epsilon:
        v_old = v_new
        continue
    break

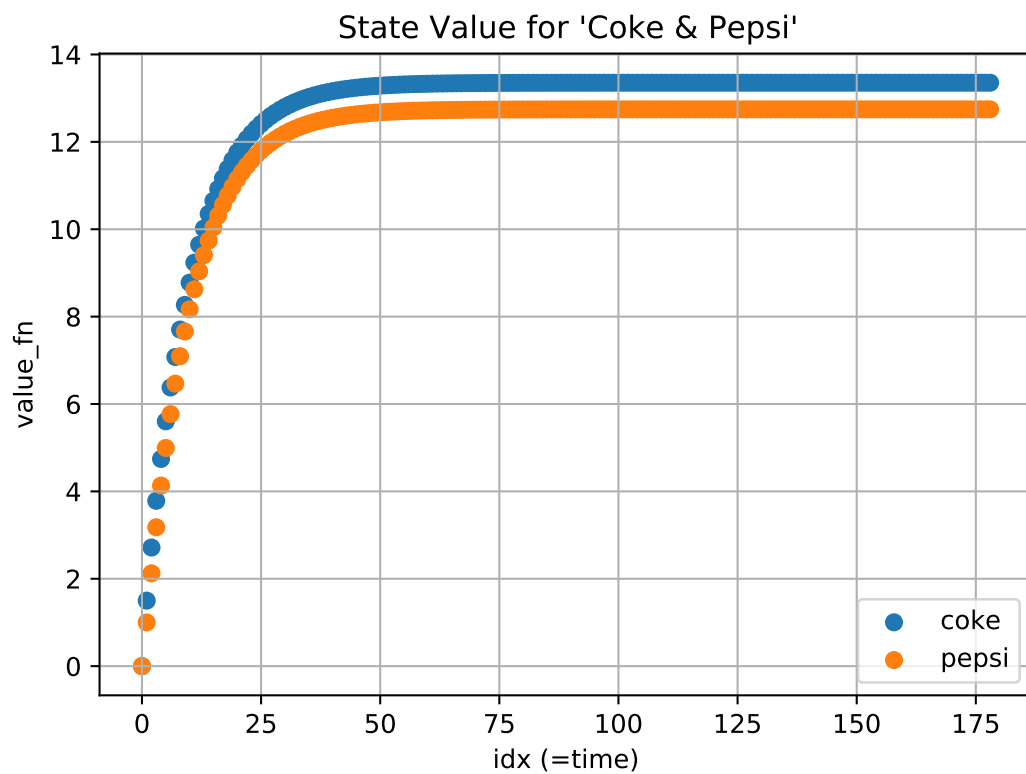
result=pd.DataFrame(np.array(result).reshape(len(result),2), columns = ['coke', 'pepsi'])

print(result)
```

```
##           coke      pepsi
## 0      0.000000  0.000000
## 1      1.500000  1.000000
## 2      2.715000  2.125000
## 3      3.784200  3.178000
## 4      4.742106  4.132990
## ..          ...        ...
## 174    13.353658  12.743902
## 175    13.353658  12.743902
## 176    13.353658  12.743902
## 177    13.353658  12.743902
## 178    13.353658  12.743902
##
## [179 rows x 2 columns]
```

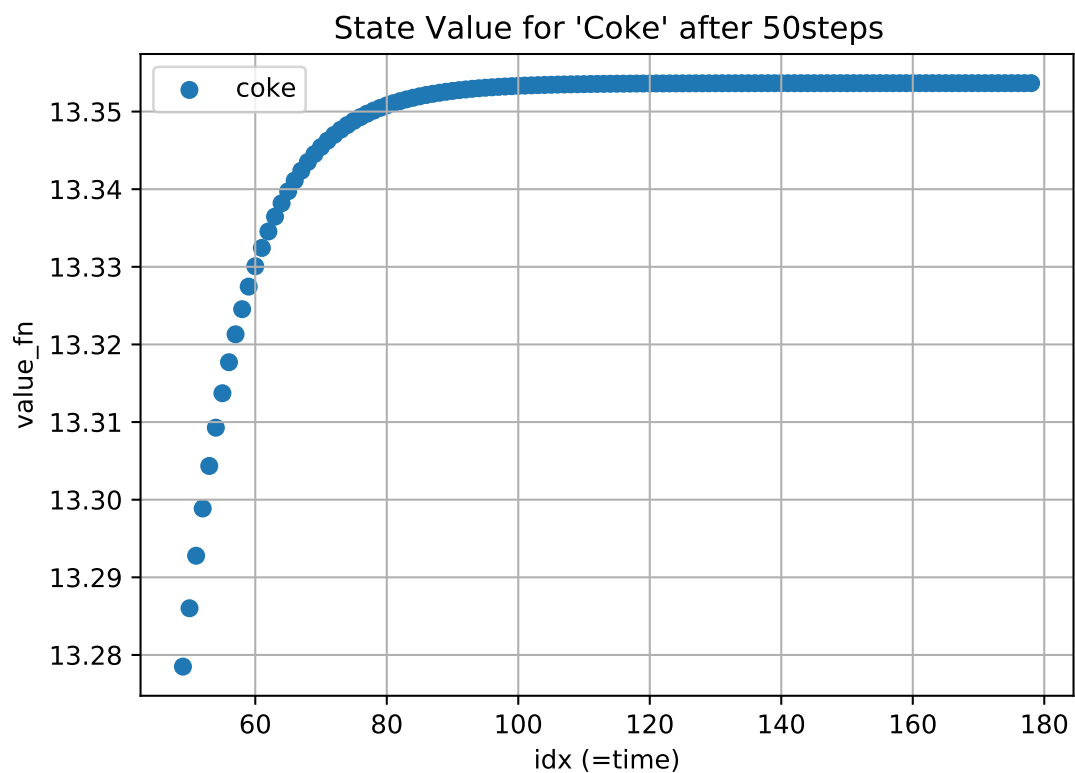
result plot

```
import matplotlib.pyplot as plt
plt.scatter(result.index, result['coke'],label='coke')
plt.scatter(result.index, result['pepsi'],label='pepsi')
plt.grid(True)
plt.title("State Value for 'Coke & Pepsi'")
plt.ylabel('value_fn')
plt.xlabel('idx (=time)')
plt.legend()
plt.show()
```



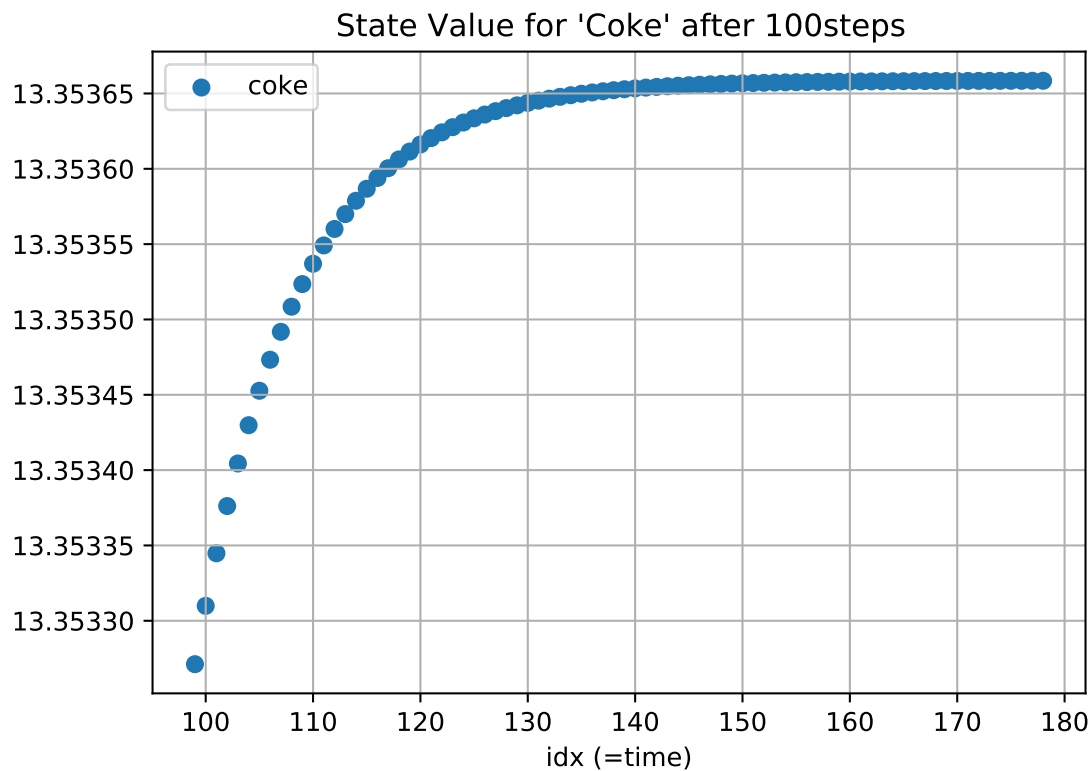
After 50 steps (coke only)

```
plt.scatter(result.index[49:], result['coke'][49:],label='coke')
plt.grid(True)
plt.title("State Value for 'Coke' after 50steps")
plt.ylabel('value_fn')
plt.xlabel('idx (=time)')
plt.legend()
plt.show()
```



After 100 steps (coke only)

```
plt.scatter(result.index[99:], result['coke'][99:],label='coke')
plt.grid(True)
plt.title("State Value for 'Coke' after 100steps")
ax=plt.gca()
ax.get_yaxis().get_major_formatter().set_useOffset(False)
plt.ylabel('value_fn')
plt.xlabel('idx (=time)')
plt.legend()
plt.show()
```



"Done, Lecture D2. Markov Reword Process2 "

```
## [1] "Done, Lecture D2. Markov Reword Process2 "
```