Lecture C2.Discrete Time Markov Chain 2

Baek, Jong min

2021-01-09

차 례

Method 1-eigen-decomposition

Remark 3(Page 12)

```
import numpy as np
eig = np.linalg.eig
P = np.array([0.7, 0.5, 0.3, 0.5]).reshape(2,2)
print('Eigenvalue : {}'.format(eig(P)[0]))
```

```
## Eigenvalue : [1.  0.2]
```

```
print('Eigenvector : {}'.format(eig(P)[1]))
```

```
## Eigenvector : [[ 0.85749293 -0.70710678]
##  [ 0.51449576  0.70710678]]
```

```
x_1 = eig(P)[1]
x_1 = x_1[:,0]
x_1
```

```
## array([0.85749293, 0.51449576])
```

```
v = x_1/np.sum(x_1)
print('stationary distribution : {}'.format(v))
```

```
## stationary distribution : [0.625 0.375]
```

Remark 5(Page 15)

```
P = np.array([0.7, 0.5, 0.3, 0.5]).reshape(2,2)
n = len(P) # nrow
I = np.identity(n)
A = np.c_[(P-I).T,np.array([1,1])]
b = np.array([0,0,1])
print(A)
```

```
## [[-0.3  0.3  1. ]
##  [ 0.5 -0.5  1. ]]
```

```
print(b)
```

```
## [0 0 1]
```

```
v = np.linalg.solve(np.dot(A,A.T), np.dot(A,b.T))
v
```

```
## array([0.625, 0.375])
```

Limiting probabilities

Motivation (Page 17)

```
from numpy.linalg import matrix_power
P = np.array([0.7,0.5,0.3,0.5]).reshape(2,2).T
print(P)
```

```
## [[0.7 0.3]
##  [0.5 0.5]]
```

```
print(matrix_power(P,2))
```

```
## [[0.64 0.36]
##  [0.6  0.4 ]]
```

```
print(matrix_power(P,3))
```

```
## [[0.628 0.372]
##  [0.62  0.38 ]]
```

```
print(matrix_power(P,4))
```

```
## [[0.6256 0.3744]
##  [0.624  0.376 ]]
```

```
print(matrix_power(P,20))
```

```
## [[0.625 0.375]
##  [0.625 0.375]]
```

Page 19

The limiting distribution may or may not exist. For example,

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

```
## array([[0, 1],
##        [1, 0]])
```

```
matrix_power(P,2)
```

```
## array([[1, 0],
##        [0, 1]])
```

```
matrix_power(P,3)
```

```
## array([[0, 1],
##        [1, 0]])
```

C2.Rmd

```
"Hello"
```

```
## [1] "Hello"
```