Lecture C2. Discrete Time Markov Chain2 Solution

Reinforcement Learning Study

2021-01-12

차 례

Method 1 -eigen-decomposition p.12

in R

```
P <- array(c(0.7, 0.5, 0.3, 0.5),dim =c(2,2))
eigen(t(P)) #eigen-decomposition for P^t
```

```
## eigen() decomposition
## $values
## [1] 1.0 0.2
##
## $vectors
##           [,1]       [,2]
## [1,] 0.8574929 -0.7071068
## [2,] 0.5144958  0.7071068
```

```
x_1 <-eigen(t(P))$vectors[,1]
x_1
```

```
## [1] 0.8574929 0.5144958
```

```
v<-x_1/sum(x_1)
v
```

```
## [1] 0.625 0.375
```

in Python

```
import numpy as np
P = np.array([[0.7, 0.3],[0.5, 0.5]])
egien_value, egien_vector = np.linalg.eig(P.T) #eigen-decomposition for P^t
print("egien_value :\n",egien_value)
```

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

```
print("egien_vector :\n",egien_vector)
```

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

```
x_1=egien_vector[:,0]
print(x_1)
```

```
## [0.85749293 0.51449576]
```

```
v=x_1/np.sum(x_1)
print(v)
```

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

Method 2- system of linear equation p.15

in R

```
P<- array(c(0.7, 0.5, 0.3, 0.5), dim =c(2,2))
n<- nrow(P)
I<-diag(n)
A<-cbind(P-I,rep(1,n))
b<-array(c(rep(0,n),1),dim =c(1, n+1))
A
```

```
##      [,1] [,2] [,3]
## [1,] -0.3  0.3    1
## [2,]  0.5 -0.5    1
```

```
b
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    1
```

```
v <- solve(A %*%t(A),A%*%t(b))
v
```

```
##       [,1]
## [1,] 0.625
## [2,] 0.375
```

in Python

```python
import numpy as np
P=np.array([[0.7, 0.3],[0.5, 0.5]])
n=len(P) # n=|S|
I=np.identity(n) # identity matrix
A=np.c_[P-I,np.repeat(1,n)] ## np._c_ equalt to cbind in r
b=np.append(np.repeat(0,n), np.array(1))
print(A)
```

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

```python
print(b)
```

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

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

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

Limiting Probability -Motivation p.17

in R

```r
library(expm)
```

```
## Loading required package: Matrix

##

## Attaching package: 'expm'

## The following object is masked from 'package:Matrix':
##
##     expm
```

```r
P <- array(c(0.7,0.5,0.3,0.5), dim = c(2,2))
P %*% P
```

```
##      [,1] [,2]
## [1,] 0.64 0.36
## [2,] 0.60 0.40
```

```r
P %^% 3
```

```
##       [,1]  [,2]
## [1,] 0.628 0.372
## [2,] 0.620 0.380
```

```r
P %^% 4
```

```
##        [,1]   [,2]
## [1,] 0.6256 0.3744
## [2,] 0.6240 0.3760
```

```r
P %^% 20
```

```
##       [,1]  [,2]
## [1,] 0.625 0.375
## [2,] 0.625 0.375
```

in Python

```
import numpy as np
from numpy.linalg import matrix_power

P = np.array([[0.7,0.5],[0.3,0.5]])
print(P)
```

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

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

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

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

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

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

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

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

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

Limiting Probability- p.19

The limiting distribution may or may not exit. For example

in r

```r
library(expm)
P<-array(c(1,0,0,1), dim=c(2,2))
P
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```r
P %^% 2
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```r
P %^% 3
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

in Python

```
from numpy.linalg import matrix_power

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

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

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

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

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

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

```
"Done, Lecture C2. Discrete Time Markov Chain2 "
```

```
## [1] "Done, Lecture C2. Discrete Time Markov Chain2 "
```