Lecture C1.Discrete Time Markov Chain 1

Baek, Jong min

2021-01-06

차 례

Exercise 2

Suppose $\mathbb{P}(S_0 = c)$ = 0.6 and $\mathbb{p}(S_0 = p)$ = 0.4 then what is $\mathbb{P}(S_1 = c)$ = ?

Solution 1)

$$[0.6\ 0.4] \begin{bmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \end{bmatrix} = [0.62\ 0.38]$$

Solution 2)

$$\begin{aligned} \mathbb{P}(S_1 = c) &= \mathbb{P}(S_1 = c, S_o = c) + \mathbb{P}(S_1 = c, S_o = p) \\ &= \mathbb{P}(S_1 \mid S_o = c)\mathbb{P}(S_o = c) + \mathbb{P}(S_1 = c \mid S_o = p)\mathbb{P}(S_o = p) \\ &= 0.7 \times 0.6 + 0.5 \times 0.4 \\ &= 0.62 \end{aligned}$$

Exercise 3

Suppose $\mathbb{P}(S_o = c)$ = 0.6 and $\mathbb{P}(S_o = p)$ = 0.4, then what is $\mathbb{P}(S_2 = c)$ = ?

$$[0.6\ 0.4] = \begin{bmatrix} 0.7 & 0.3 \\ 0.5 & 0.5 \end{bmatrix}^2 = [?\ ?]$$

$$[0.6\ 0.4] = \begin{bmatrix} 0.64 & 0.36 \\ 0.6 & 0.4 \end{bmatrix} = [0.624\ 0.376]$$

Exercise 4

Suppose $S_o = p$, then what is $\mathbb{P}(S_2 = p)$ = ?

$$\begin{aligned} S_o &= p \\ &= \mathbb{P}(S_1 = c) = 0.5 \\ &= \mathbb{P}(S_1 = p) = 0.5 \\ [0\,1](P^2) &= [0\ 1] \begin{bmatrix} 0.64 & 0.36 \\ 0.6 & 0.4 \end{bmatrix} \\ &= [0.6\,0.4] \end{aligned}$$

Page.25

```
def soda_simul(this_state):
  u = np.random.uniform(size=1)
  if this_state == 'c':
    if u <= 0.7:
      next_state = 'c'
    else:
      next_state = 'p'
  else:
    if u <= 0.5:
      next_state = 'c'
    else:
      next_state = 'p'
  return next_state
```

```
for i in range(1,6):
  path = 'c'
  for n in range(1,10):
    this_state = path[-1]
    next_state = soda_simul(this_state)
    path += next_state
  print(path)
```

```
## cccccccccc
## cppccccccc
## cccpcppcpp
## cccpcpppcc
## cppccccccc
```

Page.26

To address the send question regarding expected spending, we certainly need more than 5 path
Let's do it with 10,000 Monte-Carlo simulation
We need cost evaluating function that calculates cost for each path

```
def cost_eval(path):
  cost_one_path = path.count('c')*1.5 + path.count('p')*1
  return cost_one_path
```

```
MC_N = 10000
spending_records = np.repeat(0,MC_N)
for i in range(0,MC_N):
  path = 'c'
  for t in range(1,10):
    this_state = path[-1]
    next_state = soda_simul(this_state)
    path+= next_state
  spending_records[i] = cost_eval(path)
```

```
np.mean(spending_records)
```

```
## 13.0933
```

C1.Rmd

```
"Hello"
```

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