PES University, Bangalore

Established under Karnataka Act No. 16 of 2013

UE20CS312 - Data Analytics - Worksheet 5 Course instructor: Gowri Srinivasa, Professor Dept. of CSE, PES University

Designed by Yashas Kadambi, Dept. of CSE - yashasks@pesu.pes.edu

Markov Chains and AB Testing

Prerequisites

- Revise the following concepts
    - Markov Chains
    - Markov Chains with Absorbing states
    - Calculation of eventual probability of aborbtion
    - A/B Testing
- Install the following software
    - pandas
    - numpy

Points

The problems in this worksheet are for a total of 10 points with each problem having a different weightage.

- Problem 1: 2 points
- Problem 2: 4 points
- Problem 3: 4 points

Scenario 2

Its a freezing day in New york, Commisioner Wench has sent a report to Captain Holt that the 99th precinct has much lower reported crimes compared to other precincts. Upon Analysis Captain Holt decides to add feedback unit along with the 4 major units to analyse this descripency. All the units are mentioned below

1. Major Crimes

2. Traffic
3. General crimes
4. Feedback
5. Theft

The initial probablity of a person going to a particular unit on a particular day is given as follows

Major Crimes	Traffic	General crimes	Feedback	Theft
0.3	0.4	0.1	0.15	0.05

To measure how many people will go to the feedback unit, the personel files of all employees are give to the ***Move-o-Tron 99*** and it gives us the following matrix which shows us the probability of people moving from one unit to another on a particular day . It is known that the ***Move-o-Tron 99*** alwasy outputs matices which follow a first order Markov chain.

	Major Crimes	Traffic	General crimes	Feedback	Theft
Major Crimes	0.002	0.666	0.31	0.0	0.022
Traffic	0.466	0.102	0.222	0.0	0.21
General crimes	0.022	0.11	0.502	0.0	0.366
Feedback	0.0	0.0	0.0	1.0	0.0
Theft	0.11	0.122	0.066	0.0	0.702

As the people of New York are smart the will learn where all the units are present and hence the next days (day 1) distribution will be the distribution present at the end of the current day (day 0). Captain holt want to check if the matrix given by the ***Move-o-Tron*** can be used to model the footfall.

Problem 1 (2 points)

1. What technique can be used to model the probability of people going to the correct unit to report thier crime after N days? (0 points)
   - (pi)i * Pij to the power of N
2. Is the chain irreducible? Justify (0.5 point)

   A chain is said to be irreducible if every state can be reached from every other state.Therefore, for the given matrix the feedback state has all the outgoing transitions as zero.
3. What will be the intital probability of a person going to a particular unit after 1 day, 2 days, 10 days, 1000 days and 1001 days. (1 point)

- use the given code

Hint: Use the Chapman−Kolmogorov relationship

```
# C = A.B
matrix_C = np.dot(matrix_A, matrix_B)

# C = A.(B^4) can be replaced by
matrix_C = matrix_A
for _ in range(4):
    matrix_C = np.dot(matrix_C, matrix_B)
```

4. What can you say about the markov chain from state of initial probability of a person going to a particular unit after 1000 and 1001 days? (0.5 points)
   - The values are almost similar for 1000 and 1001 days so we see a convergence.

```
# Importing Libraries
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
# encoding the probabilities as a numpy array
trans_array = np.array([
    [0.002, 0.666, 0.31, 0, 0.022],
    [0.466, 0.102, 0.222, 0, 0.21],
    [0.022, 0.11, 0.502, 0, 0.366],
    [0, 0, 0, 1, 0],
    [0.11, 0.122, 0.066, 0, 0.702]
])
# ensures that the probabilities sum to 1
assert(trans_array[0].sum() == 1.0)
assert(trans_array[1].sum() == 1.0)
assert(trans_array[2].sum() == 1.0)
assert(trans_array[3].sum() == 1.0)
assert(trans_array[4].sum() == 1.0)
```

```
# encoding the initial probability of as a numpy array
state = np.array([[0.3, 0.4, 0.1, 0.15, 0.05]]) #initial day =0 pi
assert(state[0].sum() == 1.0)
```

```
# C = A.B
matrix_A=state
matrix_B=trans_array
matrix_C = np.dot(matrix_A, matrix_B)
matrix_C = matrix_A
for _ in range(4):
    matrix_C = np.dot(matrix_C, matrix_B)

matrix_C
```

```
array([[0.12164599, 0.16713173, 0.21490282, 0.15      , 0.34631946]])
```

```
for _ in range(1000):
    matrix_C = np.dot(matrix_C, matrix_B)
matrix_C
```

```
array([[0.1214373 , 0.16411091, 0.19739717, 0.15      , 0.36705462]])
```

```
for _ in range(1001):
    matrix_C = np.dot(matrix_C, matrix_B)
matrix_C
```

```
array([[0.1214373 , 0.16411091, 0.19739717, 0.15      , 0.36705462]])
```

After analysing the model Captain holt calls the squad and educates them to ask people to give feedbacks. And the details of the squad are given to the ***Move-o-Tron 99***. After reanalyising the report the ***Move-o-Tron 99*** gave out a new Matrix, which is shown below

	Major Crimes	Traffic	General crimes	Feedback	Theft
Major Crimes	0.002	0.666	0.01	0.02	0.302
Traffic	0.466	0.102	0.02	0.032	0.38
General crimes	0.0	0.0	1.0	0.0	0.0
Feedback	0.0	0.0	0.0	1.0	0.0
Theft	0.11	0.122	0.066	0.004	0.698

Considering the new report the model has to be re-evaluated. The initial probablity of a person going to a particular unit on a particular day remains the same.

Problem 2 (4 points)

1. Is the chain irreducible? Justify (0.5 point)
    - It is still not reducible as we cant reach the feedback state from the general crimes.
2. What will be the intital probability of a person going to a particular unit after 1 day, 2 days, 10 days, 1000 days and 1001 days. (1 point)

   Hint: Use the Chapman−Kolmogorov relationship

- The initial probability will be the matrix_C that we got from q3 of problem 1

3. What can you say about the markov chain from state of intital probability of a person going to a particular unit after 1000 and 1001 days? (0.5 points)
   - from previous values they seem to converge
4. Summer Edgecombe is Confidential Informant (CI) to the Officer Kimbal Cho and comes in every day to the police station. If on the first day she goes to the Major crimes Unit what will be the probability that she gives a feedback? (2 points)
   - initial state vector looks like [1,0,0,0,0]
   - eventual prob of feedback : that is expected duration of reaching a state (find canonical marix ------- etc.)

```
# np.delete()
# https://note.nkmk.me/en/python-numpy-delete/#:~:text=Using%20the%20NumPy%

print(a)
# [[ 0  1  2  3]
#  [ 4  5  6  7]
#  [ 8  9 10 11]]

print(np.delete(a, 1, 0))
# [[ 0  1  2  3]
#  [ 8  9 10 11]]

print(np.delete(a, 1, 1))
# [[ 0  2  3]
#  [ 4  6  7]
#  [ 8 10 11]]

# Deleting multiple rows or columns
print(np.delete(a, [0, 3], 1))
# [[ 1  2]
#  [ 5  6]
#  [ 9 10]]

# Deleting rows and columns
print(np.delete(np.delete(a, 1, 0), 1, 1))
# [[ 0  2  3]
#  [ 8 10 11]]
```

```
            # matrix multiplication or cross pdt C = A*B
            matrix_C = matrix_A @ matrix_B # matrix_C = np.matmul(matrix_A, matrix_B)
```

```
# encoding the probabilities as a numpy array
trans_array = np.array([
    [0.002, 0.666, 0.01, 0.020, 0.302],
    [0.466, 0.102, 0.02, 0.032, 0.38],
    [0.0, 0.0, 1, 0.0, 0.0],
    [0, 0, 0, 1, 0],
    [0.11, 0.122, 0.066, 0.004, 0.698]
])

# ensures that the probabilities sum to 1
assert(trans_array[0].sum() == 1.0)
assert(trans_array[1].sum() == 1.0)
assert(trans_array[2].sum() == 1.0)
assert(trans_array[3].sum() == 1.0)
assert(trans_array[4].sum() == 1.0)

# encoding the initial probability of as a numpy array
state = np.array([[0.3, 0.4, 0.1, 0.15, 0.05]])
assert(state[0].sum() == 1.0)
```

```
matrix_A=state
matrix_B=trans_array
matrix_C = np.dot(matrix_A, matrix_B)

matrix_C = matrix_A

for _ in range(1):
    matrix_C = np.dot(matrix_C, matrix_B)
matrix_C
```

```
     array([[0.1925, 0.2467, 0.1143, 0.169 , 0.2775]])
```

```
matrix_A=state
matrix_B=trans_array
matrix_C = np.dot(matrix_A, matrix_B)

matrix_C = matrix_A

for _ in range(2):
    matrix_C = np.dot(matrix_C, matrix_B)
matrix_C
```

```
     array([[0.1458722, 0.1872234, 0.139474 , 0.1818544, 0.345576 ]])
```

```
matrix_A=state
```

```
matrix_B=trans_array
matrix_C = np.dot(matrix_A, matrix_B)

matrix_C = matrix_A

for _ in range(10):
    matrix_C = np.dot(matrix_C, matrix_B)
matrix_C
```

```
array([[0.07693332, 0.09688565, 0.3362719 , 0.24330485, 0.24660428]])
```

```
matrix_A=state
matrix_B=trans_array
matrix_C = np.dot(matrix_A, matrix_B)

matrix_C = matrix_A

for _ in range(1000):
    matrix_C = np.dot(matrix_C, matrix_B)
matrix_C
```

```
array([[8.97378472e-28, 1.13004326e-27, 6.60595331e-01, 3.39404669e-01,
        2.87688168e-27]])
```

```
matrix_A=state
matrix_B=trans_array
matrix_C = np.dot(matrix_A, matrix_B)

matrix_C = matrix_A

for _ in range(1001):
    matrix_C = np.dot(matrix_C, matrix_B)
matrix_C
```

```
array([[8.44851901e-28, 1.06389804e-27, 6.60595331e-01, 3.39404669e-01,
        2.70848815e-27]])
```

High Convergence is seen.

Problem 3 (4 points)

It seems that there is a bug in ***Move-o-Tron 99*** which makes general crimes and feedback units as absorbing states. After updating the software of ***Move-o-Tron 99***, Captain Holt wants to find out the effect that Amy Santiago has on the probability of a person giving feedback. So one matrix is generated including Santiago and the other one without.

Matrix 1 (With Santiago)

	Major Crimes	Traffic	General crimes	Feedback	Theft
Major Crimes	0.002	0.232	0.31	0.434	0.022
Traffic	0.426	0.102	0.222	0.04	0.21

	Major Crimes	Traffic	General crimes	Feedback	Theft
General crimes	0.03	0.11	0.2	0.294	0.366
Feedback	0.003	0.176	0.321	0.3	0.2
Theft	0.11	0.422	0.166	0.1	0.202

Matrix 2 (Without Santiago)

	Major Crimes	Traffic	General crimes	Feedback	Theft
Major Crimes	0.11	0.222	0.092	0.374	0.202
Traffic	0.03	0.11	0.2	0.294	0.366
General crimes	0.002	0.232	0.31	0.434	0.022
Feedback	0.466	0.102	0.02	0.032	0.38
Theft	0.003	0.176	0.321	0.3	0.2

1. How can you find out the effect that Santiago has on the probability of feedback? (1 point)
    - We can find out the effect of Santiago by considering one case that considers feedbacks with or without Amy.
2. What effect does Santiago have on the probability of getting feedback? (1 point)
    - The probabilities seem to be lesser.

      Note: The initial probablity of a person going to a particular unit on a particular day remains the same
3. Name the test Captain Holt is performing. (0.5 points)
    - Captain Holt is performing A/B Test.

Lina Ginetti reports to Captain Holt that the there two kinds of days in the precicnt *"There are normal days and then there are days where workflow is dismal with a tiny dash of pathetic."*. Captain Holt decided to sample the initial probablity of a person going to a particular unit on a good day and a bad day.

4. Without the information about these inital probabilities, can you tell if there is any difference in the probability of getting a feedback? Explain. (1.5 points)
    - There is not much difference in the values we got.

```
# encoding the probabilities as a numpy array
# With Santiago
trans_array_with_amy = np.array([
    [0.002, 0.232, 0.31, 0.434, 0.022],
    [0.426, 0.102, 0.222, 0.04, 0.21],
    [0.03, 0.11, 0.20, 0.294, 0.366],
    [0.003, 0.176, 0.321, 0.3, 0.2],
    [0.11, 0.422, 0.166, 0.1, 0.202]
])
```

```
# Without Santiago
trans_array_without_amy = np.array([
    [0.11, 0.222, 0.092, 0.374, 0.202],
    [0.03, 0.11, 0.20, 0.294, 0.366],
    [0.002, 0.232, 0.31, 0.434, 0.022],
    [0.466, 0.102, 0.02, 0.032, 0.38],
    [0.003, 0.176, 0.321, 0.3, 0.2]
])

# ensures that the probabilities sum to 1
assert(trans_array_with_amy[0].sum() == 1.0)
assert(trans_array_with_amy[1].sum() == 1.0)
assert(trans_array_with_amy[2].sum() == 1.0)
assert(trans_array_with_amy[3].sum() == 1.0)
assert(trans_array_with_amy[4].sum() == 1.0)

assert(trans_array_without_amy[0].sum() == 1.0)
assert(trans_array_without_amy[1].sum() == 1.0)
assert(trans_array_without_amy[2].sum() == 1.0)
assert(trans_array_without_amy[3].sum() == 1.0)
assert(trans_array_without_amy[4].sum() == 1.0)

# encoding the initial probability of as a numpy array
state = np.array([[0.3, 0.4, 0.1, 0.15, 0.05]])
assert(state[0].sum() == 1.0)



matrix_A=state
matrix_B=trans_array_with_amy
matrix_C = np.dot(matrix_A, matrix_B)

matrix_C = matrix_A
for _ in range(4):
    matrix_C = np.dot(matrix_C, matrix_B)

matrix_C
```

```
array([[0.11933933, 0.20723382, 0.23591313, 0.21620945, 0.22130427]])
```

```
matrix_A=state
matrix_B=trans_array_without_amy
matrix_C = np.dot(matrix_A, matrix_B)

matrix_C = matrix_A
for _ in range(4):
    matrix_C = np.dot(matrix_C, matrix_B)

matrix_C
```

```
array([[0.14642713, 0.16228218, 0.18628577, 0.26274735, 0.24225758]])
```