

E09 Bayesian Network

17341015 Hongzheng Chen

November 7, 2019

Contents

1	Pomegranate Installation	2
2	Building Bayesian Network	2
3	Tasks	3
3.1	Burglary	3
3.2	Diagnosing	3
4	Codes and Results	6

1 Pomegranate Installation

Under Linux:

1. Install python first (**python 2**, not python 3).
2. Run `sudo apt-get install python-pip` to install pip.
3. Run `sudo pip install pomegranate` to install pomegranate.

```
al2017@osboxes:~$ pip
The program 'pip' is currently not installed. You can install it by typing:
sudo apt install python-pip
al2017@osboxes:~$ sudo apt install python-pip
[sudo] password for al2017:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-4.10.0-28 linux-headers-4.10.0-28-generic
  linux-headers-4.10.0-33 linux-headers-4.10.0-33-generic
  linux-headers-4.10.0-35 linux-image-4.10.0-28-generic
  linux-image-4.10.0-33-generic linux-image-4.10.0-35-generic
  linux-image-extra-4.10.0-28-generic linux-image-extra-4.10.0-33-generic
  linux-image-extra-4.10.0-35-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libxapian-dev libpython-all-dev libpython-dev libpython2.7-dev python-all
  python-all-dev python-dev python-pip python-pip-whl python-pkg-resources
  python-setuptools python-wheel python2.7-dev
Suggested packages:
  python-setuptools-doc
The following NEW packages will be installed:
  libxapian-dev libpython-all-dev libpython-dev libpython2.7-dev python-all
  python-all-dev python-dev python-pip python-pip-whl python-pkg-resources
  python-setuptools python-wheel python2.7-dev
0 upgraded, 13 newly installed, 0 to remove and 113 not upgraded.
Need to get 29.8 MB of archives.
After this operation, 45.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

```
al2017@osboxes:~$ sudo pip install pomegranate
The directory /home/al2017/.cache/pip/http or its parent directory is not owned by the current user and the
cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo,
you may want sudo's -H flag.
The directory /home/al2017/.cache/pip/ or its parent directory is not owned by the current user and caching w
heels has been disabled. check the permissions and owner of that directory. If executing pip with sudo, you na
y want sudo's -H flag.
Collecting pomegranate
  Downloading pomegranate-0.8.1-cp27-cp27mu-manylinux1_x86_64.whl (9.1MB)
100% |#####| 9.2MB 115kB/s
Collecting networkx<2.0,>=1.8.1 (from pomegranate)
  Downloading networkx-1.11-py2.py3-none-any.whl (1.3MB)
100% |#####| 1.3MB 478kB/s
Collecting numpy>=1.8.0 (from pomegranate)
  Downloading numpy-1.13.3-cp27-cp27mu-manylinux1_x86_64.whl (16.0MB)
100% |#####| 16.7MB 768kB/s
Collecting scipy>=0.17.0 (from pomegranate)
  Downloading scipy-1.0.0-cp27-cp27mu-manylinux1_x86_64.whl (46.7MB)
100% |#####| 46.7MB 388kB/s
Collecting joblib>=0.9.0b4 (from pomegranate)
  Downloading joblib-0.11-py2.py3-none-any.whl (176kB)
100% |#####| 184kB 703kB/s
Collecting decorator>=3.4.0 (from networkx<2.0,>=1.8.1->pomegranate)
  Downloading decorator-4.1.2-py2.py3-none-any.whl
Installing collected packages: decorator, networkx, numpy, scipy, joblib, pomegranate
Successfully installed decorator-4.1.2 joblib-0.11 networkx-1.11 numpy-1.13.3 pomegranate-0.8.1 scipy-1.0.0
You are using pip version 8.1.1, however version 9.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

Under Windows

You can also run `pip install pomegranate` if you have installed pip. If you don't know how to install pip, please click <https://jingyan.baidu.com/article/e73e26c0d94e0524adb6a7ff.html>.

For more, please click the homepage of Pomegranate - <https://github.com/jmschrei/pomegranate> for help.

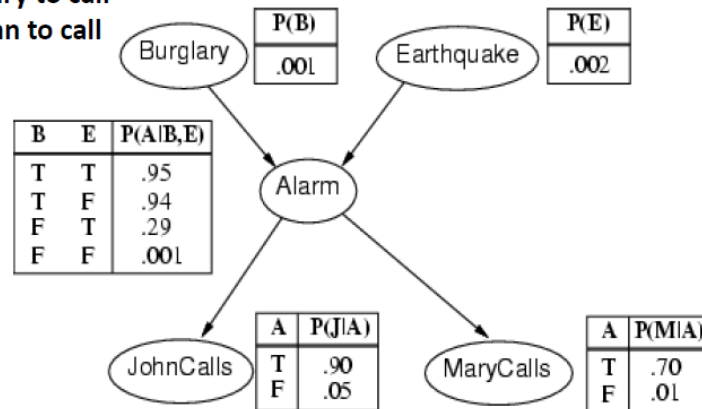
```
PS C:\Users\Lau ChiuSui\Desktop\pomegranate-0.8.1> pip install pomegranate
Collecting pomegranate
  Downloading pomegranate-0.8.1-cp27-cp27m-win32.whl (3.4MB)
100% |#####| 3.4MB 227kB/s
Collecting scipy>=0.17.0 (from pomegranate)
  Downloading scipy-1.0.0-cp27-none-win32.whl (26.4MB)
100% |#####| 26.4MB 40kB/s
Collecting joblib>=0.9.0b4 (from pomegranate)
  Downloading joblib-0.11-py2.py3-none-any.whl (176kB)
100% |#####| 184kB 957kB/s
Collecting networkx<2.0,>=1.8.1 (from pomegranate)
  Downloading networkx-1.11-py2.py3-none-any.whl (1.3MB)
100% |#####| 1.3MB 463kB/s
Requirement already satisfied: numpy>=1.8.0 in d:\softwares\python27\lib\site-packages (from pomegranate)
Collecting decorator>=3.4.0 (from networkx<2.0,>=1.8.1->pomegranate)
  Downloading decorator-4.1.2-py2.py3-none-any.whl
Installing collected packages: scipy, joblib, decorator, networkx, pomegranate
Found existing installation: networkx 2.0
Uninstalling networkx-2.0:
  Successfully uninstalled networkx-2.0
Successfully installed decorator-4.1.2 joblib-0.11 networkx-1.11 pomegranate-0.8.1 scipy-1.0.0
```

2 Building Bayesian Network

Please refer to Tutorial 4 Bayesian Networks.pdf. I will explain it in class.

- A burglary can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

Note that these tables only provide the probability that X_i is true.
(E.g., $Pr(A \text{ is true} | B, E)$)
The probability that X_i is false is 1- these values



3 Tasks

3.1 Burglary

Please code to calculate:

1. $P(A)$
2. $P(J\bar{M})$
3. $P(A|J\bar{M})$
4. $P(B|A)$
5. $P(B|J\bar{M})$
6. $P(J\bar{M}|\bar{B})$

```
P(Alarm) =
0.002516442

P(J&&~M) =
0.050054875461

P(A | J&&~M) =
0.0135738893313

P(B | A) =
0.373551228282

P(B | J&&~M) =
0.0051298581334

P(J&&~M | ~B) =
0.049847949
```

3.2 Diagnosing

Variables and their domains

```
(1)PatientAge:['0-30','31-65','65+']
(2)CTScanResult:['Ischemic Stroke','Hemorrhagic Stroke']
(3)MRIScanResult: ['Ischemic Stroke','Hemorrhagic Stroke']
(4)StrokeType: ['Ischemic Stroke','Hemorrhagic Stroke', 'Stroke Mimic']
(5)Anticoagulants: ['Used','Not used']
(6)Mortality:['True', 'False']
(7)Disability: ['Negligible', 'Moderate', 'Severe']
```

CPTs

Note: [CTScanResult, MRIScanResult,StrokeType] means:

$P(\text{StrokeType}=\dots \mid \text{CTScanResult}=\dots \wedge \text{MRIScanResult}=\dots)$

```
(1)
[PatientAge]

['0-30', 0.10],
['31-65', 0.30],
['65+', 0.60]

(2)
[CTScanResult]

['Ischemic Stroke',0.7],
[ 'Hemorrhagic Stroke',0.3]

(3)
[MRIScanResult]

['Ischemic Stroke',0.7],
[ 'Hemorrhagic Stroke',0.3]

(4)
[Anticoagulants]

['Used',0.5],
['Not used',0.5]

(5)
[CTScanResult, MRIScanResult,StrokeType]

['Ischemic Stroke','Ischemic Stroke','Ischemic Stroke',0.8],
['Ischemic Stroke','Hemorrhagic Stroke','Ischemic Stroke',0.5],
[ 'Hemorrhagic Stroke','Ischemic Stroke','Ischemic Stroke',0.5],
[ 'Hemorrhagic Stroke','Hemorrhagic Stroke','Ischemic Stroke',0],

['Ischemic Stroke','Ischemic Stroke','Hemorrhagic Stroke',0],
['Ischemic Stroke','Hemorrhagic Stroke','Hemorrhagic Stroke',0.4],
[ 'Hemorrhagic Stroke','Ischemic Stroke','Hemorrhagic Stroke',0.4],
[ 'Hemorrhagic Stroke','Hemorrhagic Stroke','Hemorrhagic Stroke',0.9],

['Ischemic Stroke','Ischemic Stroke','Stroke Mimic',0.2],
['Ischemic Stroke','Hemorrhagic Stroke','Stroke Mimic',0.1],
[ 'Hemorrhagic Stroke','Ischemic Stroke','Stroke Mimic',0.1],
[ 'Hemorrhagic Stroke','Hemorrhagic Stroke','Stroke Mimic',0.1],

(6)
```

```
[StrokeType, Anticoagulants, Mortality]

['Ischemic Stroke', 'Used', 'False',0.28],
['Hemorrhagic Stroke', 'Used', 'False',0.99],
['Stroke Mimic', 'Used', 'False',0.1],
['Ischemic Stroke','Not used', 'False',0.56],
['Hemorrhagic Stroke', 'Not used', 'False',0.58],
['Stroke Mimic', 'Not used', 'False',0.05],

['Ischemic Stroke', 'Used' , 'True',0.72],
['Hemorrhagic Stroke', 'Used', 'True',0.01],
['Stroke Mimic', 'Used', 'True',0.9],
['Ischemic Stroke', 'Not used' , 'True',0.44],
['Hemorrhagic Stroke', 'Not used', 'True',0.42 ],
['Stroke Mimic', 'Not used', 'True',0.95]
```

(7)

```
[StrokeType, PatientAge, Disability]

['Ischemic Stroke', '0-30','Negligible', 0.80],
['Hemorrhagic Stroke', '0-30','Negligible', 0.70],
['Stroke Mimic', '0-30', 'Negligible',0.9],
['Ischemic Stroke', '31-65','Negligible', 0.60],
['Hemorrhagic Stroke', '31-65','Negligible', 0.50],
['Stroke Mimic', '31-65', 'Negligible',0.4],
['Ischemic Stroke', '65+' , 'Negligible',0.30],
['Hemorrhagic Stroke', '65+' , 'Negligible',0.20],
['Stroke Mimic', '65+' , 'Negligible',0.1],

['Ischemic Stroke', '0-30' , 'Moderate',0.1],
['Hemorrhagic Stroke', '0-30' , 'Moderate',0.2],
['Stroke Mimic', '0-30' , 'Moderate',0.05],
['Ischemic Stroke', '31-65', 'Moderate',0.3],
['Hemorrhagic Stroke', '31-65', 'Moderate',0.4],
['Stroke Mimic', '31-65', 'Moderate',0.3],
['Ischemic Stroke', '65+' , 'Moderate',0.4],
['Hemorrhagic Stroke', '65+' , 'Moderate',0.2],
['Stroke Mimic', '65+' , 'Moderate',0.1],

['Ischemic Stroke', '0-30' , 'Severe',0.1],
['Hemorrhagic Stroke', '0-30' , 'Severe',0.1],
['Stroke Mimic', '0-30' , 'Severe',0.05],
['Ischemic Stroke', '31-65', 'Severe',0.1],
['Hemorrhagic Stroke', '31-65', 'Severe',0.1],
['Stroke Mimic', '31-65', 'Severe',0.3],
['Ischemic Stroke', '65+' , 'Severe',0.3],
['Hemorrhagic Stroke', '65+' , 'Severe',0.6],
['Stroke Mimic', '65+' , 'Severe',0.8]
```

Calculation

Please code to calculate the following probability value:

$p_1 = P(\text{Mortality}=\text{'True'} \mid \text{PatientAge}=\text{'31-65'} \wedge \text{CTScanResult}=\text{'Ischemic Stroke'})$

$p_2 = P(\text{Disability}=\text{'Moderate'} \mid \text{PatientAge}=\text{'65+'} \wedge \text{MRIScanResult}=\text{'Hemorrhagic Stroke'})$

$p_3 = P(\text{StrokeType}=\text{'Stroke Mimic'} \mid \text{PatientAge}=\text{'65+'} \wedge \text{CTScanResult}=\text{'Hemorrhagic Stroke'} \wedge \text{MRIScanResult}=\text{'Ischemic Stroke'})$

$p_4 = P(\text{Anticoagulants} = \text{'Not used'} \mid \text{PatientAge} = \text{'0-30'})$

```
ai2017@osboxes:~$ python diagnose.py
p1= 0.59485
p2= 0.26
p3= 0.1
p4= 0.5
```

Please solve the 2 tasks and hand in a file named E09_YourNumber.pdf, and send it to ai_201901@foxmail.com

4 Codes and Results

The codes use Python 3!

```
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/E09_BN$ python burglary.py
P(A) = 0.0025164420000000935
P(J && M) = 0.050054875461000355
P(A | J && M) = 0.01357388933131146
P(B | A) = 0.3735512282818995
P(B | J && M) = 0.005129858133403528
P(J && M | B) = 0.0498479490000000266
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/E09_BN$
```

burglary.py

```
from pomegranate import *

burglary = DiscreteDistribution( {'T':0.001, 'F':0.999} )
earthquake = DiscreteDistribution( {'T':0.002, 'F':0.998} )
alarm = ConditionalProbabilityTable(
    [['T','T','T',0.95],
     ['T','F','T',0.94],
     ['F','T','T',0.29],
     ['F','F','T',0.001],
     ['T','T','F',0.05],
     ['T','F','F',0.06],
     ['F','T','F',0.71],
     ['F','F','F',0.999]], [burglary, earthquake])
johncalls = ConditionalProbabilityTable(
    [['T','T',0.90],
     ['F','T',0.05],
     ['T','F',0.10],
     ['F','F',0.95]], [alarm])
marycalls = ConditionalProbabilityTable(
    [['T','T',0.70],
     ['F','T',0.01],
     ['T','F',0.30],
     ['F','F',0.99]], [alarm])

s1 = State(burglary, name="burglary")
s2 = State(earthquake, name="earthquake")
s3 = State(alarm, name="alarm")
s4 = State(johncalls, name="johncalls")
s5 = State(marycalls, name="marycalls")
```

```

model = BayesianNetwork("Burglary")

model.add_states(s1,s2,s3,s4,s5)

model.add_transition(s1,s3)
model.add_transition(s2,s3)
model.add_transition(s3,s4)
model.add_transition(s3,s5)
model.bake()

marginals = model.predict_proba({})
# P(A)
print("P(A) = {}".format(marginals[2].parameters[0]["T"]))
# P(J&&~M) = P(J|~M)P(~M)
j_nm = model.predict_proba({'marycalls':'F'})[3].parameters[0]["T"] * marginals[4].
    ↪ parameters[0]["F"]
print("P(J && ~M) = {}".format(j_nm))
# P(A|J&&~M)
print("P(A | J && ~M) = {}".format(model.predict_proba({'johncalls':'T','marycalls':'F'})
    ↪ [2].parameters[0]["T"]))
# P(B|A)
print("P(B | A) = {}".format(model.predict_proba({'alarm':'T'})[0].parameters[0]["T"]))
# P(B|J&&~M)
b_c_j_nm = model.predict_proba({'johncalls':'T','marycalls':'F'})[0].parameters[0]["T"]
print("P(B | J && ~M) = {}".format(b_c_j_nm))
# P(J&&~M|~B) = P(~B && J && ~M) / P(~B)
#             = P(~B | J && ~M) P(J && ~M) / P(~B)
#             = (1- P(B | J && ~M)) P(J && ~M) / P(~B)
print("P(J && ~M | ~B) = {}".format((1-b_c_j_nm) * j_nm / marginals[0].parameters[0]["F"]))

```

```

chhzh123@DESKTOP-PV2UBJL: /mnt/d/Assignments/ArtificialIntelligence/E09_BN
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/E09_BN$ python diagnosing.py
0.5948499999999999
0.26000000000000001
0.10000000000000004
0.5
chhzh123@DESKTOP-PV2UBJL:/mnt/d/Assignments/ArtificialIntelligence/E09_BN$

```

diagnosing.py

```

from pomegranate import *

PatientAge = DiscreteDistribution({'0-30':0.10, '31-65':0.30, '65+':0.60})
CTScanResult = DiscreteDistribution({'Ischemic Stroke':0.7, 'Hemorrhagic Stroke':0.3})
MRIScanResult = DiscreteDistribution({'Ischemic Stroke':0.7, 'Hemorrhagic Stroke':0.3})
Anticoagulants = DiscreteDistribution({'Used':0.5, 'Not used':0.5})

StrokeType = ConditionalProbabilityTable(
    [['Ischemic Stroke','Ischemic Stroke','Ischemic Stroke',0.8],
     ['Ischemic Stroke','Hemorrhagic Stroke','Ischemic Stroke',0.5],
     ['Hemorrhagic Stroke','Ischemic Stroke','Ischemic Stroke',0.5],
     ['Hemorrhagic Stroke','Hemorrhagic Stroke','Ischemic Stroke',0],
     ['Ischemic Stroke','Ischemic Stroke','Hemorrhagic Stroke',0],
     ['Ischemic Stroke','Hemorrhagic Stroke','Hemorrhagic Stroke',0.4],
     ['Hemorrhagic Stroke','Ischemic Stroke','Hemorrhagic Stroke',0.4],
     ['Hemorrhagic Stroke','Hemorrhagic Stroke','Hemorrhagic Stroke',0.9],

```

```

    ['Ischemic Stroke', 'Ischemic Stroke', 'Stroke Mimic', 0.2],
    ['Ischemic Stroke', 'Hemorrhagic Stroke', 'Stroke Mimic', 0.1],
    ['Hemorrhagic Stroke', 'Ischemic Stroke', 'Stroke Mimic', 0.1],
    ['Hemorrhagic Stroke', 'Hemorrhagic Stroke', 'Stroke Mimic', 0.1]], [CTScanResult,
    ⇨ MRIScanResult])
Mortality = ConditionalProbabilityTable(
    [['Ischemic Stroke', 'Used', 'False', 0.28],
    ['Hemorrhagic Stroke', 'Used', 'False', 0.99],
    ['Stroke Mimic', 'Used', 'False', 0.1],
    ['Ischemic Stroke', 'Not used', 'False', 0.56],
    ['Hemorrhagic Stroke', 'Not used', 'False', 0.58],
    ['Stroke Mimic', 'Not used', 'False', 0.05],
    ['Ischemic Stroke', 'Used', 'True', 0.72],
    ['Hemorrhagic Stroke', 'Used', 'True', 0.01],
    ['Stroke Mimic', 'Used', 'True', 0.9],
    ['Ischemic Stroke', 'Not used', 'True', 0.44],
    ['Hemorrhagic Stroke', 'Not used', 'True', 0.42],
    ['Stroke Mimic', 'Not used', 'True', 0.95]], [StrokeType, Anticoagulants])
Disability = ConditionalProbabilityTable(
    [['Ischemic Stroke', '0-30', 'Negligible', 0.80],
    ['Hemorrhagic Stroke', '0-30', 'Negligible', 0.70],
    ['Stroke Mimic', '0-30', 'Negligible', 0.9],
    ['Ischemic Stroke', '31-65', 'Negligible', 0.60],
    ['Hemorrhagic Stroke', '31-65', 'Negligible', 0.50],
    ['Stroke Mimic', '31-65', 'Negligible', 0.4],
    ['Ischemic Stroke', '65+', 'Negligible', 0.30],
    ['Hemorrhagic Stroke', '65+', 'Negligible', 0.20],
    ['Stroke Mimic', '65+', 'Negligible', 0.1],
    ['Ischemic Stroke', '0-30', 'Moderate', 0.1],
    ['Hemorrhagic Stroke', '0-30', 'Moderate', 0.2],
    ['Stroke Mimic', '0-30', 'Moderate', 0.05],
    ['Ischemic Stroke', '31-65', 'Moderate', 0.3],
    ['Hemorrhagic Stroke', '31-65', 'Moderate', 0.4],
    ['Stroke Mimic', '31-65', 'Moderate', 0.3],
    ['Ischemic Stroke', '65+', 'Moderate', 0.4],
    ['Hemorrhagic Stroke', '65+', 'Moderate', 0.2],
    ['Stroke Mimic', '65+', 'Moderate', 0.1],
    ['Ischemic Stroke', '0-30', 'Severe', 0.1],
    ['Hemorrhagic Stroke', '0-30', 'Severe', 0.1],
    ['Stroke Mimic', '0-30', 'Severe', 0.05],
    ['Ischemic Stroke', '31-65', 'Severe', 0.1],
    ['Hemorrhagic Stroke', '31-65', 'Severe', 0.1],
    ['Stroke Mimic', '31-65', 'Severe', 0.3],
    ['Ischemic Stroke', '65+', 'Severe', 0.3],
    ['Hemorrhagic Stroke', '65+', 'Severe', 0.6],
    ['Stroke Mimic', '65+', 'Severe', 0.8]], [StrokeType, PatientAge])

s1 = State(PatientAge, name="PatientAge")
s2 = State(CTScanResult, name="CTScanResult")
s3 = State(MRIScanResult, name="MRIScanResult")
s4 = State(StrokeType, name="StrokeType")
s5 = State(Anticoagulants, name="Anticoagulants")
s6 = State(Mortality, name="Mortality")
s7 = State(Disability, name="Disability")

model = BayesianNetwork("Diagnosing")

```



```

model.add_states(s1,s2,s3,s4,s5,s6,s7)

model.add_transition(s2,s4)
model.add_transition(s3,s4)

model.add_transition(s4,s6)
model.add_transition(s5,s6)

model.add_transition(s1,s7)
model.add_transition(s4,s7)

model.bake()

marginals = model.predict_proba({})

p1 = model.predict_proba({'PatientAge':'31-65','CTScanResult':'Ischemic Stroke'})[5].
    ↪ parameters[0]["True"]
p2 = model.predict_proba({'PatientAge':'65+','MRIScanResult':'Hemorrhagic Stroke'})[6].
    ↪ parameters[0]["Moderate"]
p3 = model.predict_proba({'PatientAge':'65+','CTScanResult':'Hemorrhagic Stroke',
    ↪ MRIScanResult':'Ischemic Stroke'})[3].parameters[0]["Stroke Mimic"]
p4 = model.predict_proba({'PatientAge':'0-30'})[4].parameters[0]["Not used"]
print(p1)
print(p2)
print(p3)
print(p4)

```