人工智能实验

贝叶斯网络

16 级计科教务 2 班

16337327

郑映雪

实验题目

贝叶斯网络

实验内容

算法原理

1、贝叶斯公式

贝叶斯公式来源于条件概率公式：

$$P(A|B)=\frac{P(A\cap B)}{P(B)}$$

其中P（A∩B）为 A 和 B 发生的联合概率，P（B）为 B 的边缘概率，P（A|B）又称为后验概率。联合概率又进行分解，得到贝叶斯公式如下：

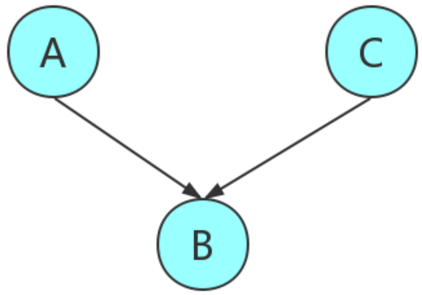
$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

2、贝叶斯网络

贝叶斯网络是一种概率图模型，用以模拟人类推理过程中因果关系，其结构是有向无环图。箭头连接的两个节点代表这两个随机变量具有因果关系，连接边的权值用条件概率表示。

贝叶斯网络有三种形式：head-to-head、tail-to-tail、head-to-tail。

head-to-head：



如图所示的“V”字形结构，在这种条件下，满足：

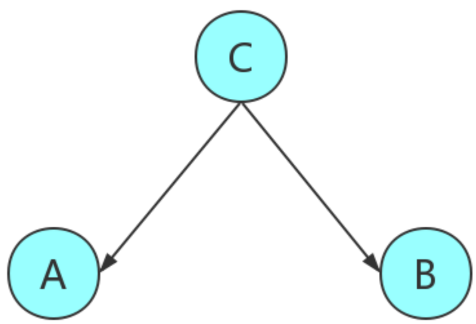
$$\sum P(A,B,C)=\sum P(A)P(C)P(B|A,C)$$

当 B 没有给定的情况下：

$$\mathrm{P(A,C)=P(A)P(C)}$$

即当 B 没有给定时，A、C 被阻断，此时它们条件独立。

tail-to-tail：



如图所示的结构，在这种条件下，满足：

$$\mathrm{P(A,B|C)=\frac{P(A,B,C)}{P(C)}}$$

又由乘法公式：

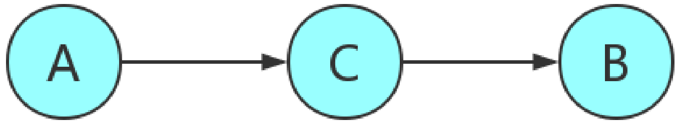
$$\mathrm{P(A,B,C)=P(C)P(A|C)P(B|C)}$$

得到：

$$\mathrm{P(A,B|C)=P(A|C)P(B|C)}$$

即，当 C 已知时，A 和 B 是条件独立的。

head-to-tail：



在 C 已知时：

$$
\begin{aligned}
P(A,B|C) &= \frac{P(A,B,C)}{P(C)} \\
&= \frac{P(A)P(C|A)P(B|C)}{P(C)} \\
&= \frac{P(A,C)P(B|C)}{P(C)} \\
&= P(A|C)P(B|C)
\end{aligned}
$$

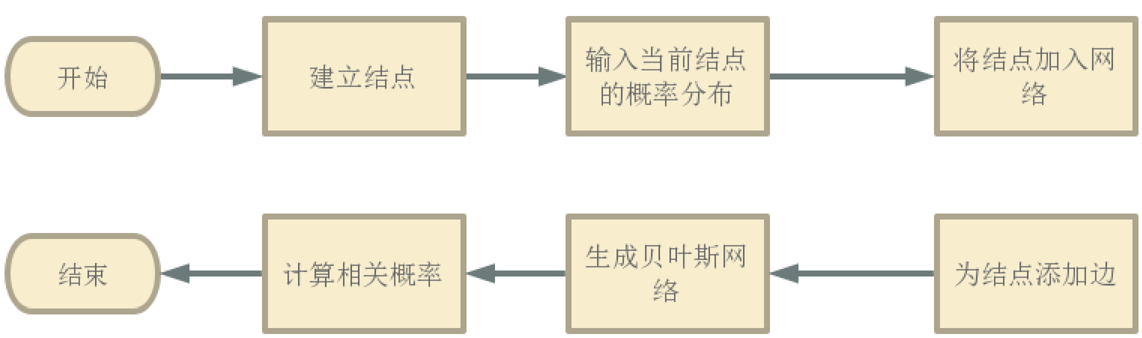
所以在 C 已知时，A 和 B 条件独立。

由这三种形式，贝叶斯网络可以基于结点之间的因果关系，简化很多概率的推算。

3、使用 pomegranate 构建贝叶斯网络

首先建立网络中所有的结点，对当前节点赋予权值（即输入当前节点的概率/条件概率），然后将这些节点加入网络中，添加每条边，最后生成网络。利用 probability 和 predict 函数+条件概率和联合概率公式可以计算概率。

流程图



关键代码

task1:

```python
from pomegranate import *

guest = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})
prize = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})
monty = ConditionalProbabilityTable(#monty 是指打开的门，它不能是奖品所在的门，也不能是客户打开的门
        [['A', 'A', 'A', 0.0],
         ['A', 'A', 'B', 0.5],#客户选择 A 门，奖品在 A 门，所以主持人打开 B 门的可能性是 0.5，其他以此类推
         ['A', 'A', 'C', 0.5],
         ['A', 'B', 'A', 0.0],
         ['A', 'B', 'B', 0.0],
         ['A', 'B', 'C', 1.0],
……………………………………………………………………………………
         ['C', 'C', 'C', 0.0]], [guest, prize])
#建立结点
s1 = Node(guest, name="guest")
s2 = Node(prize, name="prize")
s3 = Node(monty, name="monty")

model = BayesianNetwork("Monty Hall Problem")
model.add_states(s1, s2, s3)#建立贝叶斯网络结点
#添加边
model.add_edge(s1, s3)
model.add_edge(s2, s3)
model.bake()
#输出概率
print (model.probability([['A','C','B']]))
print (model.probability([['A','C','A']]))
```

task2

task2 在进行概率计算的时候，不能光代入函数，还要灵活变通，利用全概率公式和条件概率公式正确计算要求计算的概率。

```
from pomegranate import *
#be 的概率分布
burglary=DiscreteDistribution({'B':0.001,'~B':0.999})
earthquake=DiscreteDistribution({'E':0.002,'~E':0.998})
#a 的条件概率
alarm=ConditionalProbabilityTable([
    ['B','E','A',0.95],
    ['B','E','~A',0.94],
    ['B','~E','A',0.94],
    ['B','~E','~A',0.06],
    ['~B','E','A',0.29],
    ['~B','E','~A',0.71],
    ['~B','~E','A',0.001],
    ['~B','~E','~A',0.999]
],[burglary,earthquake])
#j 和 m 在 A 条件下的条件概率分布
johncalls=ConditionalProbabilityTable([
    ['A','J',0.9],
    ['A','~J',0.1],
    ['~A','J',0.05],
    ['~A','~J',0.95]
],[alarm])
marycalls=ConditionalProbabilityTable([
    ['A','M',0.7],
    ['A','~M',0.3],
    ['~A','M',0.01],
    ['~A','~M',0.99]
],[alarm])
#建立结点
sb=Node(burglary,name='burglary')
se=Node(earthquake,name='earthquake')
sa=Node(alarm,name='alarm')
sj=Node(johncalls,name='johncalls')
sm=Node(marycalls,name='marycalls')
#生成贝叶斯网络
model = BayesianNetwork("Alarm Question")
model.add_states(sb,se,sa,sj,sm)
#添加边
model.add_edge(sb,sa)
model.add_edge(se,sa)
model.add_edge(sa,sj)
model.add_edge(sa,sm)
```

```
model.bake()
#J&M 的概率为每一种 J&M 联合概率分布之和
print(model.probability([['B','E','A','J','M']])+model.probability([['~B','E','A','J','M']])+model.probability([['B','~E','A','J','M']])+model.probability([['B','E','~A','J','M']])
      +model.probability([['~B','~E','A','J','M']])+model.probability([['~B','E','~A','J','M']])+model.probability([['B','~E','~A','J','M']])+model.probability([['~B','~E','~A','J','M']]))
#输出联合概率
print(model.probability([['B','E','A','J','M']]))
#预测 A 的概率
print(model.predict_proba({'johncalls':'J','marycalls':'M'}))
#条件概率为~B 和 J 和~M 的联合概率除以~B 的概率
print((model.probability([['~B','E','A','J','~M']])+model.probability([['~B','~E','A','J','~M']])
      +model.probability([['~B','E','~A','J','~M']])+model.probability([['~B','~E','~A','J','~M']]))/0.999)
```

task3

task3 仍旧需要利用公式进行概率计算，其贝叶斯网络将在实验结果中画出。

```
from pomegranate import *
#图：C&M->S,A&S->M2,S&P->D
#pcma 概率
patientage = DiscreteDistribution({'0-30': 0.1, '31-65': 0.3, '65+': 0.6})
ctscanresult = DiscreteDistribution({'I': 0.7, 'H': 0.3})
mriscanresult = DiscreteDistribution({'I': 0.7, 'H': 0.3})
anticoagulants = DiscreteDistribution({'U': 0.5, 'N': 0.5})
#s 条件概率
stroketype=ConditionalProbabilityTable(
    [['I','I','I',0.8],
     ['I','H','I',0.5],
……………………………………
     ['H','H','S',0.1]
     ],
    [ctscanresult,mriscanresult]
)
#m2 条件概率
```

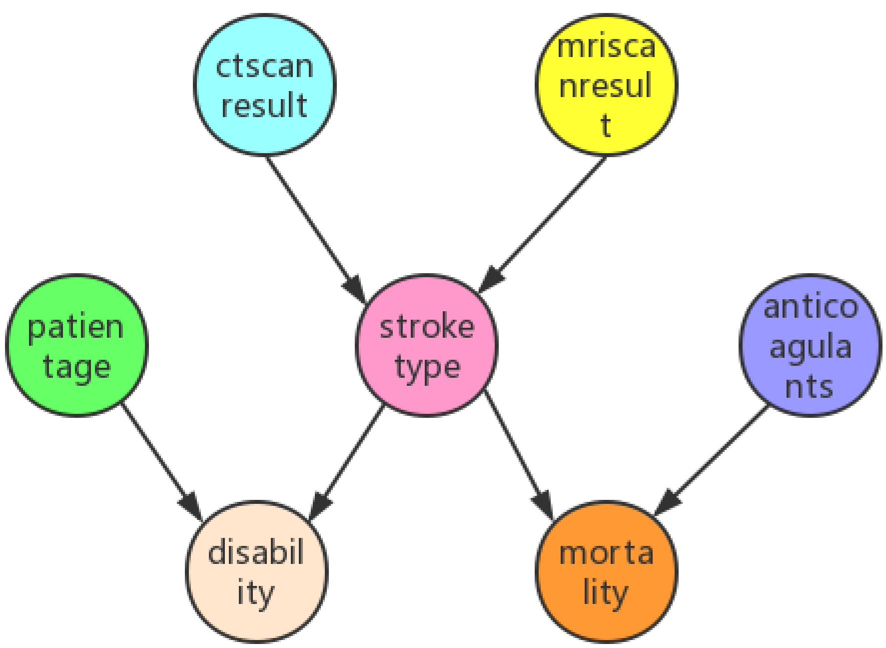
```
mortality=ConditionalProbabilityTable(
    [['I','U','F',0.28],
     ['H','U','F',0.99],
     ['S','U','F',0.1],
……………………………………………………
     ['S','N','T',0.95]
     ],
    [stroketype,anticoagulants]
)
#d 条件概率
disability=ConditionalProbabilityTable(
    [
        ['I','0-30','N',0.8],
        ['H','0-30','N',0.7],
        ['S','0-30','N',0.9],
……………………………………………………………………
        ['S', '65+', 'S', 0.8],
    ],
    [stroketype,patientage]
)
#添加结点
sp=Node(patientage,name='patientage')
sc=Node(ctscanresult,name='ctscanresult')
sm=Node(mriscanresult,name='mriscanresult')
ss=Node(stroketype,name='stroketype')
sa=Node(anticoagulants,name='anticoagulants')
sd=Node(disability,name='disability')
sm2=Node(mortality,name='mortality')
#构建贝叶斯网络
model=BayesianNetwork('task3')
model.add_states(sp,sc,sm,ss,sa,sd,sm2)
#添加边
model.add_edge(sc,ss)
model.add_edge(sm,ss)
model.add_edge(sa,sm2)
model.add_edge(ss,sm2)
model.add_edge(ss,sd)
model.add_edge(sp,sd)
#生成网络
model.bake()
#预测概率
print(model.predict_proba({'patientage':'0-30','ctscanresult':'I'}))
```

```python
print('--------')
print(model.predict_proba({'patientage':'65+','mriscanresult':'I'}))
print('--------')
print(model.predict_proba({'patientage':'65+','ctscanresult':'H','mriscanresult':'I'}))
print('--------')
print(model.predict_proba({'patientage':'0-30','anticoagulants':'U','stroketype':'S'}))
print('--------')
#联合概率
print(model.probability([['0-30','I','H','S','U','S','F']]))
```

实验结果及分析

实验结果展示

1、task3 的贝叶斯网络



2、task 中各概率结果

task1: P([‘A’ , ’ C’ , ’ B’])   P([‘A’ , ’ C’ , ’ A’])

```
0.111111111111
0.0
```

task2:

P(JohnCalls, MaryCalls)

```
0.002084101129
```

P(Burglary, Earthquake, Alarm, JohnCalls, MaryCalls)

```
1.197e-06
```

P(Alarm | JohnCalls, MaryCalls)

```
"parameters" :[
    {
        "A" :0.7606917140152378,
```

P(JohnCalls, ¬ MaryCalls | ¬Burglary)

```
0.049847949
```

task3:

p1 = P(Mortality='True' | PatientAge='0-30' , CTScanResult='Ischemic Stroke')

```
"parameters" :[
    {
        "T" :0.5948499999999999,
```

p2 = P(Disability=' Severe ' | PatientAge='65+' , MRIScanResult=' Ischemic Stroke ')

```
"parameters" :[
    {
        "S" :0.421,
```

p3 = P(StrokeType='Stroke Mimic' | PatientAge='65+' , CTScanResult='Hemmorraghic Stroke' , MRIScanResult='Ischemic Stroke')

```
"parameters" :[
    {
        "S" :0.10000000000000045,
```

p4 = P(Mortality='False' | PatientAge='0-30', Anticoagulants=’ Used’ , StrokeType='Stroke Mimic')

```
"F" :0.1000000000000002
```

p5 = P(PatientAge='0-30', CTScanResult='Ischemic Stroke', MRIScanResult=' 'Hemmorraghic Stroke' , Anticoagulants=’ Used’ , StrokeType='Stroke Mimic' , Disability=' Severe' , Mortality ='False')

```
5.25e-06
```

思考题

K2 算法有什么改进之处?

为了计算更加简便，评分函数可以变为对数计算。同时，K2 算法对变量的顺序有较高的敏感度，可以输入不同的几个顺序，比较之下取最优。