605_HW8.Rmd

Kumudini Bhave

April 1, 2017

Bayes Theorem & Contingency Tables

Problem Set 1 :

Your colleague either commutes by train or by the bus. 20 days of the month, she takes the train and the remaining 10 days she takes the bus. If she takes the train, she reaches work on time with a probability of 0:9. If she takes the bus, she frequently gets stuck in traffic and reaches work on time with a probability of 0:5. Given that she was on time today, what is the probability that she took the bus to work today?Your colleague either commutes by train or by the bus. 20 days of the month, she takes the train and the remaining 10 days she takes the bus. If she takes the train, she reaches work on time with a probability of 0:9. If she takes the bus, she frequently gets stuck in traffic and reaches work on time with a probability of 0:5. Given that she was on time today, what is the probability that she took the bus to work today?

Solution :

Probability of taking bus = 10 / 30 = 1/3

Probability of taking train = 20/ 30 = 2/3

Probability of timely arrival with bus = 0.5

Probability of timely arrival with train = 0.9

Total days on time when taken train = 0.9 * 20 = 18 days Total days on time when taken bus = 0.5 * 10 = 5 days

Total days on time = 18 + 5 = 23

Probability of the event of being on time , taken a bus is =

(no of days on time when taken a bus)/total days on time

5 / 23 =0.21739

Verifying with Bayes theorem

E : Event on time X : Vehicle taken bus / train X =x (taken bus) X != x (taken train)

P(X = x|e) =

```
P(e|X = x) * P(X = x)
---------------------
          P(e)
```

which is also

```
 P(e|X = x) * P(X = x)
-------------------------
 P(e|X = x) * P(X = x) + P(e|X != x) * P(X != x)


0.5 * (1/3)
-------------------------
(0.5 * (1/3) + 0.9 * (2/3))

    0.16666
-------------------------
   (0.16666 + 0.6)
```

= 0.16666/ 0.76666 =0.2174

Problem Set 2 : In the Grade Network that we looked at in the notes, what happens to the probability of Difficulty of Course when you present the evidence that the received recommendation letter was good? In addition, now present the evidence that both SAT scores were good and the letter of recommendation was good, What is the probability of the Difficulty of Course now? You should use the gRain package in R to build your network and perform these calculations. You may need to install RBGL package from BioConductor in R to get gRain working. See http://www.bioconductor.org/packages/release/bioc/html/RBGL.html for instructions on RBGL. Please submit your assignment as an R markdown document.

Solution :

Enlisting the SAT grade parameters

We have level of **Difficulty**

Difficulty	Level
Yes	Y
No	N

We have level of **Intelligence**

Intelligence	Level
High	H
Low	L

We have level of **Grade**

Difficulty / Intelligence	Level High	Level Low
D= No , I = Low	0.4	0.6
D = No, I = High	0.99	0.01
D = Yes, I = Low	0.2	0.8
D = Yes, I = High	0.9	0.1

We have level of **SAT Score**

Intelligence	Level High	Level Low
I = High	0.8	0.2
I = Low	0.1	0.9

We have level of **Letter Of Recommendation**

Grade	Level High	Level Low
G = High	0.95	0.05
G = Low	0.1	0.9

```
knitr::opts_chunk$set(message = FALSE, echo = TRUE)


# Library for graphical network of data
library(gRain)
```

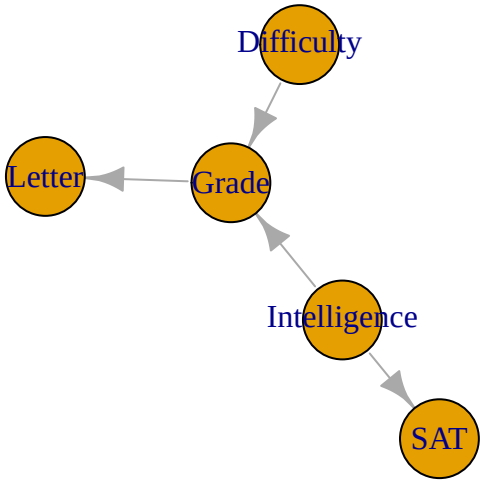
```
#  Specifying conditional probability tables


# specifying levels
yn <- c("Yes","No")
hl <- c("High","Low")

d <- cptable(~Difficulty, values = c(0.7,0.3), levels = yn)
i <- cptable(~Intelligence, values = c(0.2,0.8), levels = hl)
#g.di <- cptable(~Grade|Difficulty : Intelligence, values = c(0.4, 0.6, 0.99, 0.01, 0.2, 0.8, 0.9, 0.1)
g.di <- cptable(~Grade|Difficulty : Intelligence, values = c( 0.9, 0.1, 0.99, 0.01, 0.2, 0.8, 0.4, 0.6)


s.i <- cptable(~SAT|Intelligence, values = c(0.8, 0.2, 0.1, 0.9), levels = hl)
l.g <- cptable(~Letter|Grade, values = c(0.95, 0.05, 0.1, 0.9), levels=hl)


# Compiliing list of conditional probability tables and forming network

 plist <- compileCPT(list(d, i, g.di, s.i, l.g))
 plist
```

```
## CPTspec with probabilities:
##  P( Difficulty )
##  P( Intelligence )
##  P( Grade | Difficulty Intelligence )
##  P( SAT | Intelligence )
##  P( Letter | Grade )
```

```
gradenw <- grain(plist)
summary(gradenw)
```

```
## Independence network: Compiled: FALSE Propagated: FALSE
##  Nodes : chr [1:5] "Difficulty" "Intelligence" "Grade" "SAT" ...
```

```
# Plot model
 iplot(gradenw)
```




```
# Query model g
 querygrain(gradenw, nodes=c("Difficulty","Intelligence" , "Grade", "SAT", "Letter"))
```

```
## $Difficulty
## Difficulty
## Yes  No
## 0.7 0.3
##
## $Intelligence
## Intelligence
## High  Low
##  0.2  0.8
##
## $Grade
## Grade
##   High    Low
## 0.3934 0.6066
##
## $SAT
## SAT
## High  Low
## 0.24 0.76
##
```

```
## $Letter
## Letter
##    High     Low
## 0.43439 0.56561
```

```
plist$Difficulty
```

```
## Difficulty
## Yes  No
## 0.7 0.3
## attr(,"class")
## [1] "parray" "array"
```

```
plist$Intelligence
```

```
## Intelligence
## High  Low
##  0.2  0.8
## attr(,"class")
## [1] "parray" "array"
```

```
plist$Grade
```

```
## , , Intelligence = High
##
##       Difficulty
## Grade  Yes   No
##   High 0.9 0.99
##   Low  0.1 0.01
##
## , , Intelligence = Low
##
##       Difficulty
## Grade  Yes  No
##   High 0.2 0.4
##   Low  0.8 0.6
##
## attr(,"class")
## [1] "parray" "array"
```

```
plist$SAT
```

```
##       Intelligence
## SAT    High Low
##   High  0.8 0.1
##   Low   0.2 0.9
## attr(,"class")
## [1] "parray" "array"
```

```
plist$Letter
```

```
##       Grade
## Letter High Low
##   High 0.95 0.1
##   Low  0.05 0.9
## attr(,"class")
## [1] "parray" "array"
```

```
# Now we can add some data and repeat the query again.
# We try find that given Letter recommendation as High what is the probaility of Difficult of Course

gradenw_find1 <- setFinding(gradenw, nodes="Letter", states="High")
summary(gradenw_find1)
```

```
## Independence network: Compiled: TRUE Propagated: TRUE
##  Nodes : chr [1:5] "Difficulty" "Intelligence" "Grade" "SAT" ...
##  Number of cliques:                3
##  Maximal clique size:              3
##  Maximal state space in cliques:   8
##    nodes is.hard.evidence hard.state
## 1 Letter             TRUE       High
```

```
## Query model given parameter values (=prior) + some data
querygrain(gradenw_find1, nodes=c("Difficulty"))
```

```
## $Difficulty
## Difficulty
##       Yes        No
## 0.6268561 0.3731439
```

```
# Now we can add some data and repeat the query again.
# We try find that given SAT score High, Letter recommendation as High,  what is the probaility of Diff

# First finding the SAT High stats
gradenw_find2 <- setFinding(gradenw, nodes="SAT", states="High")
summary(gradenw_find2)
```

```
## Independence network: Compiled: TRUE Propagated: TRUE
##  Nodes : chr [1:5] "Difficulty" "Intelligence" "Grade" "SAT" ...
##  Number of cliques:                3
##  Maximal clique size:              3
##  Maximal state space in cliques:   8
##   nodes is.hard.evidence hard.state
## 1   SAT             TRUE       High
```

```
# Finding the Letter High stats from the earlier findings
gradenw_find2 <- setFinding(gradenw_find2, nodes="Letter", states="High")
summary(gradenw_find2)
```

```
## Independence network: Compiled: TRUE Propagated: TRUE
##  Nodes : chr [1:5] "Difficulty" "Intelligence" "Grade" "SAT" ...
##  Number of cliques:                3
##  Maximal clique size:              3
##  Maximal state space in cliques:   8
##    nodes is.hard.evidence hard.state
## 1    SAT             TRUE       High
## 2 Letter             TRUE       High
```

```
# Querying for probability for Difficulty , given SAT and Letter data from the latest findings
querygrain(gradenw_find2, nodes=c("Difficulty"))
```

```
## $Difficulty
## Difficulty
##       Yes        No
## 0.6676522 0.3323478
```