CUNY School of Professional Studies

SPS.CUNY.EDU

Lecture 08

2020 Spring Data-622

Naive Bayes

Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Multivariate Conditional Probability

https://math.stackexchange.com/questions/301207/why-is-px-yz-pyx-zpxz

CUNY EMAIL ISL Statistics How to Stop Skype Fr Attributes: Defining C Fundry & Yesc

Search on Mathematics...

From

$$P(X,Y|Z) = \frac{P(X,Y,Z)}{P(Z)}$$

$$P(Y|X,Z) = \frac{P(X,Y,Z)}{P(X,Z)}$$

$$P(X|Z) = \frac{P(X,Z)}{P(Z)}$$

we have

$$P(Y|X,Z)P(X|Z) = \frac{P(X,Y,Z)}{P(X,Z)}\frac{P(X,Z)}{P(Z)} = \frac{P(X,Y,Z)}{P(Z)} = P(X,Y|Z).$$

This is also intuitive: The probability that X and Y happen if we know that Z happens is the same as the probability that X happens when we know that Z happens and that then Y happens when we know that X and Z happen.

share cite improve this answer

answered Feb 12 '13 at 14:19

Hagen von Eitzen

306k 25 291 537

Multivariate Joint Probability AKA Chain Rule

https://en.wikipedia.org/wiki/Chain_rule_%28probability%29#More_than_two_random_variables

CUNY EMAIL ISL Statistics How to Stop Skype Fr Attributes: Defining C Fundry & Yesco - Firm Module 8 - G

Repeating this process with each final term creates the product:

$$\mathrm{P}\left(\bigcap_{k=1}^{n} X_k\right) = \prod_{k=1}^{n} \mathrm{P}\left(X_k \,\middle|\, \bigcap_{j=1}^{k-1} X_j\right)$$

Example [edit]

With four variables ($n = 4$), the chain rule produces this product of conditional probabilities:

$$\begin{aligned}
\mathrm{P}(X_4, X_3, X_2, X_1) &= \mathrm{P}(X_4 \mid X_3, X_2, X_1) \cdot \mathrm{P}(X_3, X_2, X_1) \\
&= \mathrm{P}(X_4 \mid X_3, X_2, X_1) \cdot \mathrm{P}(X_3 \mid X_2, X_1) \cdot \mathrm{P}(X_2, X_1) \\
&= \mathrm{P}(X_4 \mid X_3, X_2, X_1) \cdot \mathrm{P}(X_3 \mid X_2, X_1) \cdot \mathrm{P}(X_2 \mid X_1) \cdot \mathrm{P}(X_1)
\end{aligned}$$

Conditional Independence

https://www.probabilitycourse.com/chapter1/1_4_4_conditional_independence.php 67%

0 Preface
1 Basic Concepts
1.0 Introduction
1.1 Introduction
1.2 Review of Set Theory
1.3 Random Experiments and Probabilities
1.4 Conditional Probability
1.4.0 Conditional Probability
1.4.1 Independence
1.4.2 Law of Total Probability
1.4.3 Bayes' Rule
1.4.4 Conditional Independence

1.4.4 Conditional Independence

As we mentioned earlier, almost any concept that is defined for probability can also be extended to conditional probability. Remember that two events A and B are independent if

$$P(A \cap B) = P(A)P(B), \quad \text{or equivalently, } P(A|B) = P(A).$$

We can extend this concept to conditionally independent events. In particular,

Definition 1.2
Two events A and B are **conditionally independent** given an event C with $P(C) > 0$ if

$$P(A \cap B|C) = P(A|C)P(B|C) \tag{1.8}$$

Recall that from the definition of conditional probability,

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

if $P(B) > 0$. By conditioning on C, we obtain

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

if $P(B|C), P(C) \neq 0$. If A and B are conditionally independent given C, we obtain

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

Thus, if A and B are conditionally independent given C, then

$$P(A|B,C) = P(A|C) \tag{1.9}$$

Thus, Equations 1.8 and 1.9 are equivalent statements of the definition of conditional independence. Now let's look at an example.

Bayes → Naive

Introduction

posterior

LIKELIHOOD

This is a simple probabilistic **classifier** based on the **Bayes** theorem, from the Wikipedia article. This project contains source files that can be included in any C# project.

$$P(W|Q) = \frac{P(Q|W)P(W)}{P(Q)} = \frac{P(Q|W)P(W)}{P(Q|W)P(W) + P(Q|M)P(M)}$$

Q data
W class

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

Bayes

total Prob

The **Bayes**ian **Classifier** is capable of calculating the most probable output depending on the input. It is possible to add new raw data at runtime and have a better probabilistic **classifier**. A **naive Bayes classifier** assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of other features, a **naive Bayes classifier** considers all of these properties to independently contribute to the probability that this fruit is an apple.

That is why the model is naive...

A simple NB example

Sex classification

Problem: classify whether a given person is a male or a female based on the measured features. The features include height, weight, and foot size.

```
dftrain<-data.frame(gender=c('M','M','M','M','F','F','F','F'),
 height=c(6,5.92,5.58,5.92,5,5.5,5.42,5.75),
 weight=c(180,190,170,165,100,150,130,150),
 foot=c(12,11,12,10,6,8,7,9),stringsAsFactors=F)
```

Training

Example training set is shown below.

sex	height (feet)	weight (lbs)	foot size (inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

8 data points,2 classes, 3 predictors

A simple NB example-CONTD

The **classifier** created from the training set using a Gaussian distribution assumption would be:

sex	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

Let's say we have equiprobable classes so P(male)= P(female) = 0.5. There was no identified reason for making this assumption so it may have been a bad idea. If we determine P(C) based on frequency in the training set, we happen to get the same answer.

```
male_mean<-unlist(apply(dftrain[dftrain$gender=='M',2:4],2,mean))
 male_mean
 male_sd<-unlist(apply(dftrain[dftrain$gender=='M',2:4],2,sd))
 female_sd<-unlist(apply(dftrain[dftrain$gender=='F',2:4],2,sd))
 female_mean<-unlist(apply(dftrain[dftrain$gender=='F',2:4],2,mean))
 dfparameters<-data.frame(rbind(male_mean,male_sd))
 dfparameters<-rbind(dfparameters,female_mean,female_sd)
rownames(dfparameters)<-c("male_mean","male_sd","female_mean","female_sd")
dfparameters<-cbind(dfparameters,gender=c('M','M','F','F'))
```

A simple NB example-CONTD

https://www.codeproject.com/articles/318126/naive-bayes-classifier

Raman Kannan - Outlook ... | Naive Bayes Classifier - Cod... | read.dta function | R Documen... | Raman Kannan - Outlook Web...

Inbox (10,016) - rk2153@g... | The data warehouse toolki... | Welcome · Advanced R | Seeing Theory | admin - Shortcut | Libraries - Shortcut

Below is a sample to be classified as a male or female.

sex	height (feet)	weight (lbs)	foot size (inches)
sample	6	130	8

We wish to determine which posterior is greater, male or female. For the classification as male, the posterior is given by:

$$posterior(male) = \frac{P(male)\, p(height|male)\, p(weight|male)\, p(footsize|male)}{evidence}$$

For the classification as female, the posterior is given by:

$$posterior(female) = \frac{P(female)\, p(height|female)\, p(weight|female)\, p(footsize|female)}{evidence}$$

The evidence (also termed normalizing constant) may be calculated since the sum of the posteriors equals one.

$$evidence = P(male)\, p(height|male)\, p(weight|male)\, p(footsize|male) + P(female)\, p(height|female)\, p(weight|female)\, p(footsize|female)$$

The evidence may be ignored since it is a positive constant. (Normal distributions are always positive.) We now determine the sex of the sample.

P(male) = 0.5

A simple NB example-CONTD

$$p(\text{height}|\text{male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6-\mu)^2}{2\sigma^2}\right) \approx 1.5789$$

, where $\mu = 5.855$ and $\sigma^2 = 3.5033e - 02$ are the parameters of normal distribution which have been previously determined from the training set. Note that a value greater than 1 is OK here – it is a probability density rather the probability, because height is a continuous variable.

p(weight | male) = 5.9881e-06

p(foot size | male) = 1.3112e-3

posterior numerator (male) = their product = 6.1984e-09

P(female) = 0.5

p(height | female) = 2.2346e-1

p(weight | female) = 1.6789e-2

p(foot size | female) = 2.8669e-1

posterior numerator (female) = their product = 5.3778e-04

Since posterior numerator is greater in the female case, we predict the sample is female.

A simple NB example-CONTD

```
given<-c(6,130,8)
as.character(unique(dfparameters$gender)[[which.max(
unlist(lapply(lapply(
lapply(1:2,FUN=function(x,v=given,df=dfparameters){
cid=x
meanrow=(x-1)*2+1
sdrow=(x-1)*2+2
prob<-unlist(lapply(1:length(v),FUN=function(x,vi=v,dfi=df)
{
value=vi[[x]]
mu=dfi[meanrow,x]
sigma=dfi[sdrow,x]
rval<-(1/sqrt(2*22*sigma^2/7))*exp((-1*(value-mu)^2)/(2*sigma^2))
rval
}))
prob}
)
,cumprod),FUN=function(x)x[[length(x)]])))]])
```

A simple NB example-CONTD

```
given<-c(6,130,8)
as.character(unique(dfparameters$gender)[[which.max(
unlist(lapply(lapply(
lapply(1:2,FUN=function(x,v=given,df=dfparameters){
cid=x
meanrow=(x-1)*2+1
sdrow=(x-1)*2+2
prob<-unlist(lapply(1:length(v),FUN=function(x,vi=v,dfi=df)
{
value=vi[[x]]
mu=dfi[meanrow,x]
sigma=dfi[sdrow,x]
rval<-(1/sqrt(2*22*sigma^2/7))*exp((-1*(value-mu)^2)/
(2*sigma^2))
rval
}))
prob}
)
,cumprod),FUN=function(x)x[[length(x)]])))]])
```

A simple NB example-CONTD

```
>
> given<-c(6,130,8)
> as.character(unique(dfparameters$gender)[[which.max(
+ unlist(lapply(lapply(
+ lapply(1:2,FUN=function(x,v=given,df=dfparameters){
+ cid=x
+ meanrow=(x-1)*2+1
+ sdrow=(x-1)*2+2
+ prob<-unlist(lapply(1:length(v),FUN=function(x,vi=v,dfi=df)
+ {
+ value=vi[[x]]
+ mu=dfi[meanrow,x]
+ sigma=dfi[sdrow,x]
+ rval<-(1/sqrt(2*22*sigma^2/7))*exp((-1*(value-mu)^2)/(2*sigma^2))
+ rval
+ }))
+ prob}
+ )
+ ,cumprod),FUN=function(x)x[[length(x)]]))))]])
[1] "F"
>
```

How can we iteraate through all the observations in dftrain?

A simple NB example-CONTD

```
unlist(
apply(dftrain[,-1],1,FUN=function(x)
{
given=x
#above segment
))
```

```
> unlist(
+ apply(dftrain[,-1],1,FUN=function(x)
+ {
+ given=x
+ as.character(unique(dfparameters$gender)[[which.max(
+ unlist(lapply(lapply(
+ lapply(1:2,FUN=function(x,v=given,df=dfparameters){
+ cid=x
+ meanrow=(x-1)*2+1
+ sdrow=(x-1)*2+2
+ prob<-unlist(lapply(1:length(v),FUN=function(x,vi=v,dfi=df)
+ {
+ value=vi[[x]]
+ mu=dfi[meanrow,x]
+ sigma=dfi[sdrow,x]
+ rval<-(1/sqrt(2*22*sigma^2/7))*exp((-1*(value-mu)^2)/(2*sigma^2))
+ rval
+ }))
+ prob}
+ )
+ ,cumprod),FUN=function(x)x[[length(x)]])))]])
+ })
+ )
[1] "M" "M" "M" "M" "F" "F" "F" "F"
>
> dftrain[,1]
[1] "M" "M" "M" "M" "F" "F" "F" "F"
>
```

Is it possible to verify our implementaton with standard package NB Implementation.

A simple NB example-CONTD

```
require(e1071)
nbmodel<-naiveBayes(gender~.,data=dftrain)
nb.predicted<-predict(nbmodel,dftrain[,-1],type='raw')
nb.predicted
unlist(lapply(apply(nb.predicted,1,which.max),
FUN=function(x)names(as.data.frame(nb.predicted))
[[x]]))
#confusion matrix for training error

table(dftrain$gender,unlist(lapply(apply(nb.predicted,1,w
hich.max),
FUN=function(x)names(as.data.frame(nb.predicted))
[[x]])))
# never seen before data
dftest<-data.frame(height=6,weight=130,foot=8)
nbtest.pred<-predict(nbmodel,dftest,type='raw')
nbtest.pred
dftest<-data.frame(height=6,weight=130,foot=8)
predict(nbmodel,dftest,type='raw')
```

```
> require(e1071)
> nbmodel<-naiveBayes(gender~.,data=dftrain)
> nb.predicted<-predict(nbmodel,dftrain[,-1],type='raw')
> #nb.predicted
> unlist(lapply(apply(nb.predicted,1,which.max),
+ FUN=function(x)names(as.data.frame(nb.predicted))[[x]]))
[1] "M" "M" "M" "M" "F" "F" "F" "F"
> #confusion matrix for training error
>  table(dftrain$gender,unlist(lapply(apply(nb.predicted,1,which.max),
+ FUN=function(x)names(as.data.frame(nb.predicted))[[x]])))

    F M
  F 4 0
  M 0 4
> # never seen before data
> dftest<-data.frame(height=6,weight=130,foot=8)
> nbtest.pred<-predict(nbmodel,dftest,type='raw')
> nbtest.pred
             F            M
[1,] 0.9999885 1.152307e-05
> dftest<-data.frame(height=6,weight=130,foot=8)
> predict(nbmodel,dftest,type='raw')
             F            M
[1,] 0.9999885 1.152307e-05
> 
```

References

https://en.wikipedia.org/wiki/Chain_rule_%28probability%29#More_than_two_random_variables
https://math.stackexchange.com/questions/301207/why-is-px-yz-pyx-zpxz
https://corporatefinanceinstitute.com/resources/knowledge/other/total-probability-rule/
https://www.probabilitycourse.com/chapter1/1_4_4_conditional_independence.php
https://www.codeproject.com/articles/318126/naive-bayes-classifier
https://rpubs.com/riazakhan94/naive_bayes_classifier_e1071

Review

We have seen 3 different probabilistic classifiers:
Logistic, LDA and NaiveBayes

Since these algorithms model a probability distribution needing Only a few parameters for the PDF,they are also called parametric models.

Is it possible to estimate class given data without assuming PDF. Our focus will be such classifiers which do not assume any PDF and calculate based on logic and instances.

Practice … Practice … Practice