Naive Bayes

Carlos Carvalho and Rob McCulloch

1. Bayes Rule Classification
2. Naive Bayes Classification
3. Sentiment Analysis: Spam or Ham

1. Bayes Rule Classification

In linear regression we are trying to predict a *numeric* Y given *numeric* $x = (x_1, x_2, \ldots, x_p)$.

Let's consider the *very important* problem of predicting a categorical Y.

Many methods can be viewed as an attempt to estimate:

$$p(y|x)$$

the conditional distribution of Y given $X = x$.

For example, in logistic regression Y is 0 or 1 and we have:

$$p(Y = 1|x) \sim \text{Bernoulli}(p(x)),\ p(x) = \frac{e^{\beta_0 + \sum \beta_j x_j}}{1 + e^{\beta_0 + \sum \beta_j x_j}}.$$

$\beta_0 + \sum_{j=1}^{p} \beta_j x_j = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p.$

An alternative approach is to estimate the full joint distribution of (X, Y) by estimating the marginal for Y ($p(y)$) and the conditional for X ($p(x|y)$).

We then have the joint via:

$$p(x, y) = p(y)\, p(x|y).$$

And classification is then obtained from Bayes Theorem:

$$p(y|x) = \frac{p(y)p(x|y)}{p(x)}$$

As usual we can predict the most probable y or make a decision based on the probabilities.

For discrete y and x we have:

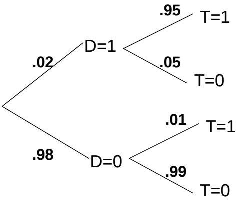
$$p(y|x) = \frac{p(y,x)}{p(x)} = \frac{p(y,x)}{\sum_y p(y,x)} = \frac{p(y)p(x|y)}{\sum_y p(y)p(x|y)}$$

For binary y (y is 0 or 1) we have:

$$p(Y=1|x) = \frac{p(Y=1)\,p(x|Y=1)}{p(Y=0)\,p(x|Y=0) + p(Y=1)\,p(x|Y=1)}$$

Recall our Disease testing example.

Let $D = 1$ indicate you have a disease.
Let $T = 1$ indicate that you test positive for it
We know the marginal of D and the conditional of T given D.



$\Longrightarrow$

		D	
		0	1
T	0	.9702=.98*.99	.001
	1	.0098	.019

$$p(Y=1|x) = \frac{p(Y=1)\,p(x|Y=1)}{p(Y=0)\,p(x|Y=0) + p(Y=1)\,p(x|Y=1)}$$

In the disease testing example Y is D and X is T:

$$p(D=1|T=1) = \frac{p(T=1|D=1)p(D=1)}{p(T=1|D=1)p(D=1)+p(T=1|D=0)p(D=0)}$$

$$p(D=1|T=1) = \frac{0.019}{(0.019+0.0098)} = 0.66$$

Odds Ratios

Note that for binary Y, a nice way to look at Bayes theorem is with the *odds ratio*:

$$\frac{p(Y=1|x)}{p(Y=0|x)} = \frac{p(Y=1)}{p(Y=0)} \frac{p(x|Y=1)}{p(x|Y=0)}$$

$\frac{p(Y=1)}{p(Y=0)}$: the prior odds ratio.

$\frac{p(x|Y=1)}{p(x|Y=0)}$: the likelihood ratio.

$\frac{p(Y=1|x)}{p(Y=0|x)}$: the posterior odds ratio.

Disease testing:
posterior odds: $.66/(1\text{-}.66) = 1.941176$
prior odds: $.02/.98 = 0.02040816$
likelihood ratio: $.95/.01 = 95$

prior odds x likelihood ratio: $.0204{*}95 = 1.938$

Probability from odds:

$$p(Y=1|x) = \frac{p(Y=1)\,p(x|Y=1)}{p(Y=0)\,p(x|Y=0) + p(Y=1)\,p(x|Y=1)}$$

Divide top and bottom by $p(Y=0)\,p(x|Y=0)$:

$$p(Y=1|x) = \frac{odds}{1+odds}$$

Disease testing:
$1.938/(1+1.938) = 0.6596324$

2. Naive Bayes Classification

Naive Bayes classification uses the Bayes Theorem approach to classification.

The tricky part is that we would like this to work for large x!!!

$$x = (x_1, x_2, \ldots, x_p)$$

where p may be large !!!!

In our application we will have $p = 1,136$!!

How do we get $p(x|y)$ from the data when p is large???

Naive Bayes classification simplifies the problem by assumming that the elements of $X = (X_1, X_2, \ldots, X_p)$ are *conditionally independent* given Y:

$$p(x, y) = p(y)\, p(x \mid y) = p(y) \prod_i p(x_i|y)$$

Each coordinate x_i of x gets to multiply in it's own contribution of evidence about y depending on how likely x_i would be if $Y = y$.

For example, suppose we just have $x = (x_1, x_2)$ and each x is binary (0 or 1).

$p(Y = 1 | X_1 = 1, X_2 = 0) =$

$$= \frac{p(Y = 1)p(X_1 = 1, X_2 = 0 | Y = 1)}{p(Y = 1)p(X_1 = 1, X_2 = 0 | Y = 1) + p(Y = 0)p(X_1 = 1, X_2 = 0 | Y = 0)}$$

$$= \frac{p(Y = 1)p(X_1 = 1 | Y = 1)p(X_2 = 0 | Y = 1)}{p(Y = 1)p(X_1 = 1 | Y = 1)p(X_2 = 0 | Y = 1) + p(Y = 0)p(X_1 = 1 | Y = 0)p(X_2 = 0 | Y = 0)}$$

Same idea works with p x variables instead of 2 !!!
You just have to estimate $p(X_i = 1 | y)$ for each i !!!

NB has some key advantages:

- We only have to estimate the low dimension $p(x_i|y)$ instead of the high dimensional $p(x|y)$!!!!
- Many small bits of information from each x_i can be combined.
- It is simple.

The main disadvantage is that the conditional independence assumption often seems inappropriate. However, *this does not seem to keep from working very well in practice !!!*

According to Mladen Kolar,

> *NB is the single most used classifier out there. NB often performs well, even when the assumption is violated.*

3. Sentiment Analysis: Spam or Ham

Sentiment analysis tries to understand text documents.

A popular approach is to combine "bag of words" with NB.

Each word in the document provides an additional independent piece of evidence about the kind of document it is.

A simple example is trying to classify the document as spam or not: "spam or ham".

Bag of words means just that, we ignore the order of the words.

The document:

> *When the lecture is over, remember to wake up the person sitting next to you in the lecture room.*

is the same as the document,

> *in is lecture lecture next over person remember room sitting the the the to to up wake when you*

SMS Spam Data:

Note: this follows Chapter 4 of "Machine Learning with R", by Brett Lanz.

Note: sms: short message service.

Have 5,559 sms text message documents.

Each one is labelled as spam or ham.

Here is the first (ham) and fourth (spam) observation:

```
> smsRaw[1,]
  type                                             text
1  ham Hope you are having a good week. Just checking in
> smsRaw[4,]
  type
4 spam

4 complimentary 4 STAR Ibiza Holiday or £10,000 cash needs your URGENT collection.
09066364349 NOW from Landline not to lose out! Box434SK38WP150PPM18+
```

Work flow:

- clean: tolower, kill numbers, punctuation, stopwords
- stem: (help,helped,helping,helps) becomes (help,help,help,help)
- tokenization: split a document up into single words (or "tokens" or "terms").
- document term matrix (DTM): rows indicate documents columns are counts for terms.
- train/test split.
- throw away low count terms.
- convert DTM to indicators: Yes if the word (term) is in the document, 0 else.
- do Naive Bayes!!

Note:

Most of the work is processing the data !!!!!

This is often the case in real world applications.

Getting the data into a form that allows you to analysize it is time consuming and **very** important.

Garbage in, garbage out !!!

In data science we talk about things like “data wrangling” and “feature engineering”.

Clean and Stem

Here are the first two documents:

```
> smsRaw$text[1]
[1] "Hope you are having a good week. Just checking in"
> smsRaw$text[2]
[1] "K..give back my thanks."
```

Here are the first 2 docs after cleaning.
smsCC is the cleaned data in the Corpus data structure from the `tm` R package. smsCC is for sms data as a Cleaned Corpus.

```
> smsCC[[1]][1]
$content
[1] "hope good week just check"

> smsCC[[2]][1]
$content
[1] "kgive back thank"
```

Tokenize and get DTM

Tokenization gives us 6518 words (or terms) from all the 5,559 sms documents.

The i^{th} row of the DTM gives us the count for each term in document i.

```
> print(dim(smsDtm))
[1] 5559 6518
> library(slam) #for col_sums
> summary(col_sums(smsDtm)) #summarize total time a term is used.
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.000   6.776   4.000 658.000
> terms = smsDtm$dimnames$Terms
> nterm = length(terms)
> set.seed(14)
> ii = sample(1:nterm,20)
> terms[ii]
 [1] "effect"      "pinku"       "wikipediacom" "mundh"       "wwwsmsconet"
 [6] "marsm"       "voic"        "itz"          "logo"        "hip"
[11] "transfr"     "colin"       "leo"          "technolog"   "text"
[16] "scratch"     "graze"       "prolli"       "tech"        "ofsi"
```

Average word frequencies for ham observations:

Average word frequencies for spam observations:

If the word (term) "call" is in a document does that make it more or less likely that it is spam?

Train/Test

We split our data into train/test:
train: we estimate/learn/train our model using the training data.
test: see how well we predict on the test data.

```
#train and test
# creating training and test datasets
smsTrain = smsDtm[1:4169, ]
smsTest  = smsDtm[4170:5559, ]

# also save the labels
smsTrainy = smsRaw[1:4169, ]$type
smsTesty  = smsRaw[4170:5559, ]$type

> prop.table(table(smsTrainy))
smsTrainy
      ham      spam
0.8647158 0.1352842
> prop.table(table(smsTesty))
smsTesty
      ham      spam
0.8683453 0.1316547
```

Throw Away Terms with Low Frequency

```
# save frequently-appearing terms to a character vector
smsFreqWords = findFreqTerms(smsTrain, 5)

> str(smsFreqWords)
 chr [1:1136] "abiola" "abl" "abt" "accept" "access" "account" ...

> length(smsFreqWords)
[1] 1136

# create DTMs with only the frequent terms
smsFreqTrain = smsTrain[ , smsFreqWords]
smsFreqTest = smsTest[ , smsFreqWords]

> dim(smsFreqTrain)
[1] 4169 1136
> dim(smsTest)
[1] 1390 1136
```

Convert Counts to Indicators

Convert number of times a term is in a document to just whether or not it is in the document.

```
#convert counts to if(count>0) (yes,no)
convertCounts <- function(x) {
  x <- ifelse(x > 0, "Yes", "No")
}
# apply() convert_counts() to columns of train/test data
# these are just matrices
smsTrain = apply(smsFreqTrain, MARGIN = 2, convertCounts)
smsTest  <- apply(smsFreqTest, MARGIN = 2, convertCounts)

> dim(smsTrain)
[1] 4169 1136
> smsTrain[1:3,1:5]
    Terms
Docs abiola abl  abt  accept access
   1 "No"   "No" "No" "No"   "No"
   2 "No"   "No" "No" "No"   "No"
   3 "No"   "No" "No" "No"   "No"
```

We are ready for NB!!!

```
library(e1071)
smsNB = naiveBayes(smsTrain, smsTrainy)

> smsNB$tables[1:3]
$abiola
         abiola
smsTrainy          No         Yes
     ham  0.998058252 0.001941748
     spam 1.000000000 0.000000000

$abl
         abl
smsTrainy          No         Yes
     ham  0.994729542 0.005270458
     spam 1.000000000 0.000000000

$abt
         abt
smsTrainy          No         Yes
     ham  0.995839112 0.004160888
     spam 1.000000000 0.000000000
```

The tables are our $p(x_i|y)$ terms !!
y is ham or spam and x_i are the words(terms): *abiola*, *abl*, *abt*, ...

That is $p(abiola = Yes \,|\, y = ham) = 0.001941748$.

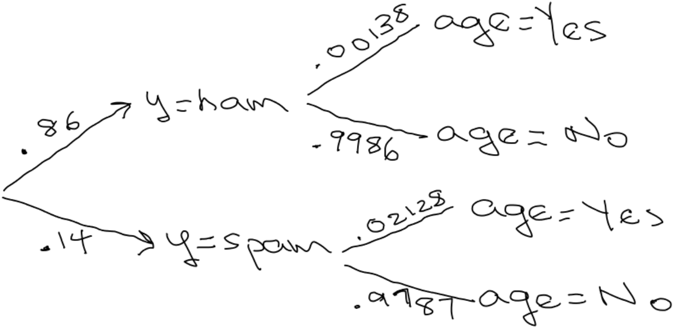
What is the probability an sms is spam given the word age is in it

What is $p(y = spam|age = Yes)$?

Let's use $p(y = spam) = .14$, the training data proportion.

And age given y is exactly the table:

```
$age
         age
smsTrainy          No         Yes
     ham  0.998613037 0.001386963
     spam 0.978723404 0.021276596
```



		y	
		ham	spam
age	No	*	*
	Yes	.86×.00138	.14×.02128

$$p(y = spam | age = Yes) =$$

$$\frac{p(y{=}spam)p(age{=}Yes|y{=}spam)}{p(y{=}spam)p(age{=}Yes|y{=}spam)+p(y{=}ham)p(age{=}Yes|y{=}ham)}$$

```
> .14*0.021276596/(.14*0.021276596 + .86*0.001386963)
[1] 0.7140633
```

```
In [1]: priodds = .14/.86

In [2]: likerat = 0.021276596/0.001386963

In [3]: priodds
Out[3]: 0.16279069767441862

In [4]: likerat
Out[4]: 15.340420761044093

In [5]: postodds = priodds*likerat

In [6]: postodds
Out[6]: 2.497277798309504

In [7]: pspam = postodds/(1+postodds)

In [8]: pspam
Out[8]: 0.7140633207681201
```

age was 15 times more likely to be in the message if it was spam!!

Now let's use two words and Naive Bayes !!!

```
$age
        age
smsTrainy          No         Yes
     ham  0.998613037 0.001386963
     spam 0.978723404 0.021276596

$adult
        adult
smsTrainy          No         Yes
     ham  0.999445215 0.000554785
     spam 0.994680851 0.005319149
```

What is $p(y = spam|age = Yes, adult = Yes)$?
(The prob the sms is spam given the word age is in it and the word adult is in it).

$$p(y = spam|age = Yes, adult = Yes) =$$

$$\frac{p(y=spam)p(age=Yes|y=spam)p(adult=Yes|y=spam)}{p(y=spam)p(age=Yes|y=spam)p(adult=Yes|y=spam)+p(y=ham)p(age=Yes|y=ham)p(adult=Yes|y=ham)}$$

```
> .14*0.021276596*0.005319149/(.14*0.021276596*0.005319149 + .86*0.001386963*0.000554785)
[1] 0.9599091
```

Ok, let's try it with all the terms (words) !!!

Out of Sample Confusion Matrix

```
yhat = predict(smsNB,smsTest)

library(gmodels)
CrossTable(yhat, smsTesty,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

```
             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1201 |        30 |      1231 |
             |     0.995 |     0.164 |           |
-------------|-----------|-----------|-----------|
        spam |         6 |       153 |       159 |
             |     0.005 |     0.836 |           |
-------------|-----------|-----------|-----------|
Column Total |      1207 |       183 |      1390 |
             |     0.868 |     0.132 |           |
-------------|-----------|-----------|-----------|
```

Missclassification rate:

$36/1390 = 0.02589928$

% spam detected: .836.

```
> 153/(153+30)
[1] 0.8360656
```