Naive Bayes

1. Introduction

The technique descended from the work of the 18th century mathematician Thomas Bayes, who developed foundational mathematical principles (now known as Bayesian methods) for describing the probability of events, and how probabilities should be revised considering additional information.

Classifiers based on Bayesian methods utilize training data to calculate an observed probability of each class based on feature values. When the classifier is used later on unlabeled data, it uses the observed probabilities to predict the most likely class for the new features. In fact, Bayesian classifiers have been used for:

- Text classification, such as junk email (spam) filtering, author identification, or topic categorization
- Intrusion detection or anomaly detection in computer networks
- Diagnosing medical conditions, when given a set of observed symptoms

Typically, Bayesian classifiers are best applied to problems in which the information from numerous attributes should be considered simultaneously to estimate the probability of an outcome. While many algorithms ignore features that have weak effects, Bayesian methods utilize all available evidence to subtly change the predictions.

1.1. Bayes' theorem

The relationships between dependent events can be described using Bayes' theorem, as shown in the following formula.

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

The notation P(A|B) can be read as the probability of event A given that event B occurred. This is known as **conditional probability**, since the probability of A is dependent (that is, conditional) on what happened with event B.

We can use spam and ham separate problem to introduce naive Bayes algorithm. For instance, if 10 out of 50 email messages are spam, then the probability of spam can be estimated as 20 percent. The notation P(A) is used to denote the probability of event A, as in P(spam) = 0.20. Given the value P(spam) = 0.20, we can calculate P(ham) = 1 – 0.20 = 0.80. This works because the events spam and ham are mutually exclusive and exhaustive. This means that the events cannot occur at the same time and are the only two possible outcomes. As shorthand, the notation P(¬A) can be used to denote the probability of event A not occurring, as in P(¬spam) = 0.80. In the following diagram, the rectangle represents the set of all possible outcomes for an email message. The circle represents the probability that the message is spam. The remaining 80 percent represents the messages that are not spam:

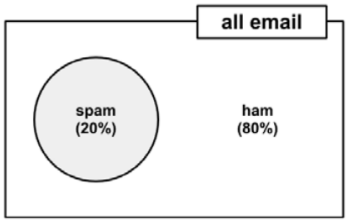

Figure 1

Consider, for instance, a second event based on the outcome that the email message contains the word Viagra. For most people, this word is only likely to appear in a spam message; its presence in a message is therefore a very strong piece of evidence that the email is spam. The preceding diagram, updated for this second event, might appear as shown in the following diagram:

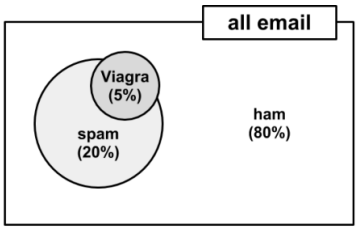

Figure 2

Notice in the diagram that the Viagra circle does not completely fill the spam circle, nor is it completely contained by the spam circle. This implies that not all spam messages contain the word Viagra, and not every email with the word Viagra is spam.

By applying Bayes' theorem to this evidence, we can compute a posterior probability that measures how likely the message is to be spam. If the posterior probability is greater than 50 percent, the message is more likely to be spam than ham, and it should be filtered. The following formula is the Bayes' theorem for the given evidence:

$$P(\text{spam} \mid \text{Viagra}) = \frac{P(\text{Viagra} \mid \text{spam})\,P(\text{spam})}{P(\text{Viagra})}$$

likelihood, prior probability, posterior probability, marginal likelihood

To calculate the components of Bayes' theorem, we must construct a frequency table (shown on the left in the following diagram) that records the number of times Viagra appeared in spam and ham messages. The frequency table can then be used to construct a likelihood table, as shown on right in the following diagram:

Frequency	Viagra: Yes	Viagra: No	Total
spam	4	16	20
ham	1	79	80
Total	5	95	100

Likelihood	Viagra: Yes	Viagra: No	Total
spam	4 / 20	16 / 20	20
ham	1 / 80	79 / 80	80
Total	5 / 100	95 / 100	100

The likelihood table reveals that P(Viagra|spam) = 4/20 = 0.20, indicating that the probability is 20 percent that a spam message contains the term Viagra. Additionally, since the theorem says that P(B|A) * P(A) = P(A ∩ B), we can calculate P(spam ∩ Viagra) as P(Viagra|spam) * P(spam) = (4/20) * (20/100) = 0.04. To compute the posterior probability, P(spam|Viagra), we simply take P(Viagra|spam) * P(spam) / P(Viagra), or (4/20) * (20/100) / (5/100) = 0.80. Therefore, the probability is 80 percent that a message is spam, given that it contains the word Viagra. Therefore, any message containing this term should be filtered.

1.2. Naive Bayes algorithm for spam classification

The naive Bayes (NB) algorithm describes a simple application using Bayes' theorem for classification. It is the most common, particularly for text classification where it has become the de facto standard. Strengths and weaknesses of this algorithm are as follows:

Table 1 Strengths and Weaknesses of naïve bayes method

Strengths	Weaknesses
• Simple, fast, and very effective • Does well with noisy and missing data • Requires relatively few examples for training, but also works well with very large numbers of examples • Easy to obtain the estimated probability for a prediction	• Relies on an often-faulty assumption of equally important • and independent features • Not ideal for datasets with large numbers of numeric features • Estimated probabilities are less reliable than the predicted classes

The naive Bayes algorithm is named as such because it makes a couple of "naive" assumptions about the data. In particular, naive Bayes assumes that all the features in the dataset are equally important and independent. These assumptions are rarely true in most of the real-world applications.

However, in most cases when these assumptions are violated, naive Bayes still performs well. This is true even in extreme circumstances where strong dependencies are found among the features. Due to the algorithm's versatility and accuracy across many types of conditions, naive Bayes is often a strong first candidate for classification learning tasks.

Note:

The exact reason why naive Bayes works well in spite of its faulty assumptions? One explanation is that it is not important to obtain a careful estimate of probability so long as the predicted class values are true. For instance, if a spam filter correctly identifies spam, does it matter that it was 51 percent or 99 percent confident in its prediction?

Let's extend our spam filter by adding a few additional terms to be monitored: money, groceries, and unsubscribe. The naive Bayes learner is trained by constructing a likelihood table for the appearance of these four words (W1, W2, W3, and W4), as shown in the following diagram for 100 emails:

	Viagra (W_1)		Money (W_2)		Groceries (W_3)		Unsubscribe (W_4)		
Likelihood	Yes	No	Yes	No	Yes	No	Yes	No	Total
spam	4 / 20	16 / 20	10 / 20	10 / 20	0 / 20	20 / 20	12 / 20	8 / 20	20
ham	1 / 80	79 / 80	14 / 80	66 / 80	8 / 80	71 / 80	23 / 80	57 / 80	80
Total	5 / 100	95 / 100	24 / 100	76 / 100	8 / 100	91 / 100	35 / 100	65 / 100	100

As new messages are received, the posterior probability must be calculated to determine whether they are more likely spam or ham, given the likelihood of the words found in the message text. For example, suppose that a message contains the terms Viagra and Unsubscribe, but does not contain either Money or Groceries. Using Bayes' theorem, we can define the problem as shown in the following formula, which captures the probability that a message is spam, given that Viagra = Yes, Money = No, Groceries = No, and Unsubscribe = Yes:

$$P(Spam|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4|spam)P(spam)}{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)}$$

For a number of reasons, this formula is computationally difficult to solve. As additional features are added, tremendous amounts of memory are needed to store probabilities for all the possible intersecting events. The work becomes much easier if we can exploit the fact that naive Bayes assumes independence among events. Specifically, naive Bayes assumes **class-conditional independence**, which means that events are independent so long as they are conditioned on the same class value. you may recall is P(A ∩ B) = P(A) * P(B), This results in a much easier-to-compute formulation, shown as follows:

$$P(Spam|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{P(W_1|spam)P(\neg W_2|spam)P(\neg W_3|spam)P(W_4|spam)P(spam)}{P(W_1)P(\neg W_2)P(\neg W_3)P(W_4)}$$

The result of this formula should be compared to the probability that the message is ham:

$$P(ham|W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{P(W_1|ham)P(\neg W_2|ham)P(\neg W_3|ham)P(W_4|ham)P(ham)}{P(W_1)P(\neg W_2)P(\neg W_3)P(W_4)}$$

Because the denominator is the same in both cases, it can be ignored for now. The overall likelihood of spam is then:

(4/20) * (10/20) * (20/20) * (12/20) * (20/100) = 0.012

While the likelihood of ham given this pattern of words is:

(1/80) * (66/80) * (71/80) * (23/80) * (80/100) = 0.002

Because 0.012 / 0.002 = 6, we can say that this message is six times more likely to be spam than ham. However, to convert these numbers to probabilities, we need one last step.

The probability of spam is equal to the likelihood that the message is spam divided by the likelihood that the message is either spam or ham:

0.012 / (0.012 + 0.002) = 0.857

Similarly, the probability of ham is equal to the likelihood that the message is ham divided by the likelihood that the message is either spam or ham:

0.002 / (0.012 + 0.002) = 0.143

Given the pattern of words in the message, we expect that the message is spam with 85.7 percent probability, and ham with 14.3 percent probability. Because these are mutually exclusive and exhaustive events, the probabilities sum up to one.

The naive Bayes classification algorithm we used in the preceding example can be summarized by the following formula.

$$P(C_L|F_1, ..., F_n) = \frac{1}{Z}p(C_L)\prod_{i=1}^{n} p(F_i|C_L)$$

1.3. The Laplace estimator

Let's look at one more example. Suppose we received another message, this time containing the terms: Viagra, Groceries, Money, and Unsubscribe. Using the naïve Bayes algorithm as before, we can compute the likelihood of spam as:

(4/20) * (10/20) * (0/20) * (12/20) * (20/100) = 0

And the likelihood of ham is:

(1/80) * (14/80) * (8/80) * (23/80) * (80/100) = 0.00005

Therefore, the probability of spam is:

0 / (0 + 0.0099) = 0

And the probability of ham is:

0.00005 / (0 + 0. 0.00005) = 1

The result is wrong obviously, because of some features are zero lead to total probability is zero. A solution to this problem involves using something called the Laplace estimator. The Laplace estimator essentially adds a small number to each of the counts in the frequency table, which ensures that each feature has a nonzero probability of occurring with each class. Typically, the Laplace estimator is set to 1, which ensures that each class-feature combination is found in the data at least once.

Let's see how this affects our prediction for this message. Using a Laplace value of 1, we add one to each numerator in the likelihood function. The total number of 1s must also be added to each denominator. The likelihood of spam is therefore:

$$(5/24) * (11/24) * (1/24) * (13/24) * (20/100) = 0.0004$$

And the likelihood of ham is:

$$(2/84) * (15/84) * (9/84) * (24/84) * (80/100) = 0.0001$$

This means that the probability of spam is 80 percent and the probability of ham is 20 percent; a more plausible result than the one obtained when Groceries alone determined the result.

2. Filtering mobile phone spam

The Naïve Bayes algorithm is very suit for filtering spam. We make the emails in the excel, the table like this:

Table 2 emails for spam classification

type	text
ham	Hope you are having a good week. Just checking in
ham	K..give back my thanks.
ham	Am also doing in cbe only. But have to pay.
spam	complimentary 4 STAR Ibiza Holiday or £10,000 cash needs your URGENT collection. 09066364349 NOW from Landline not to lose out! Box434SK38WP150PPM18+
spam	okmail: Dear Dave this is your final notice to collect your 4* Tenerife Holiday or #5000 CASH award! Call 09061743806 from landline. TCs SAE Box326 CW25WX 150ppm
ham	Aiya we discuss later lar... Pick u up at 4 is it?
ham	Are you this much buzy
……	……

We want to classify which email is spam, using R or Python can solve the Naïve Bayes problem.

2.1. Program with R

The program can be divided by five steps.

Step1: collecting data

The data set type should like table 2, one column is ‘type’, anther one is test.

Step2: exploring and preparing the data

The ‘type’ should be factor and read csv as ‘UTF-8’. Then make the text to a VCoupus type, and convert messages to lowercase, remove all numbers, remove punctuation, remove additional whitespace, remove the stop words. After preparing complete, creating a sparse matrix with DocumentTermMatrix command, and creating training set and test set. We can use word clouds function to visually depict the frequency at which words appear in text data. Creating indicator features for frequency words which appearing at least 5 times and change the sparse matrix number to ‘yes’ and ‘no’.

Step3: training a model on the data

Use package e1071 to do the training.

Step4: evaluating model performance

Using predict() function and CrossTable() function to evaluate the result.

Step5: improving model performance

Set a value for Laplace estimator and train again.

The flowing table of naïve Bayes for filtering spam is depicted by picture 3.

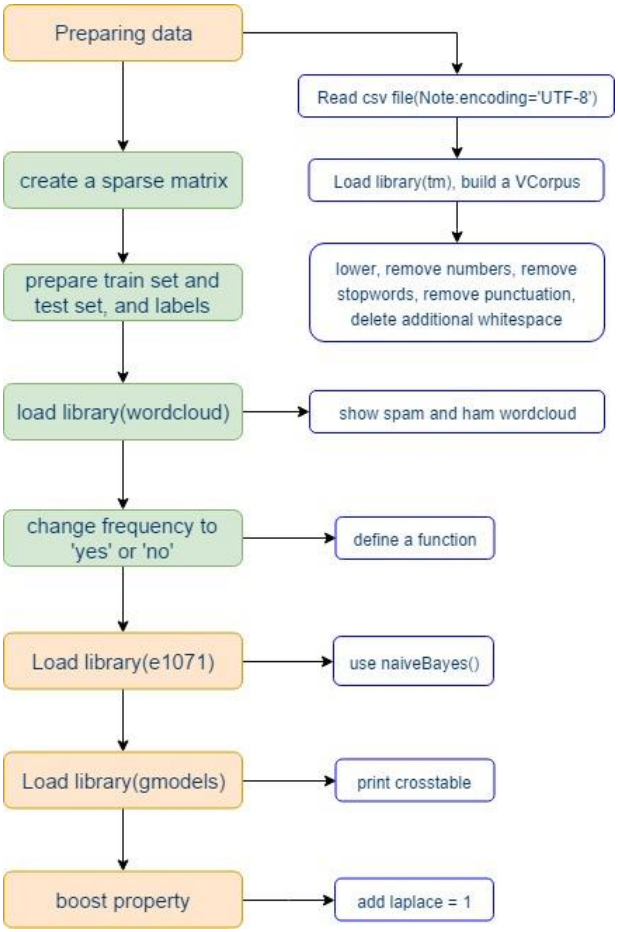

Figure 3 Naïve Bayes method for filtering spam

Naïve Bayes classification syntax

Using the naiveBayes() function in the e1071 package
Building the classifier: m <- naiveBayes(train, class, laplace=0) • train is a data frame or matrix containing training data • class is a factor vector with the class for each row in the training data • laplace is a number to control the laplace estimator (by default, 0) The function will return a naïve Bayes model object that can be used to make predictions. **Making predictions**: p <- predict(m, test, type='class') • m is a model trained by the naiveBayes() function • test is a data frame or matrix containing test data with the same features as the training data used to build the classifier. • type is either "class" or "raw" and specifies whether the predictions should be the most likely class value or the raw predicted probabilities.

The function will return a vector of predicted class values or raw predicted probabilities depending upon the value of the type parameter.

Example:

sms_classifier <- naiveBayes(sms_train, sms_type)

sms_prediction <- predict(sms_classifier, sms_test)

Program:

```
# read "sms_spam.csv" file,
# we must add encoding='UTF-8'otherwise we can't use DocumentTermMatrix funciton later
sms_raw <- read.csv("sms_spam.csv", stringsAsFactors = FALSE,encoding='UTF-8')

# rename
names(sms_raw)[1] <- 'type'

# check the variables
str(sms_raw$type)
table(sms_raw$type)

# change spam and ham to factor
sms_raw$type <- factor(sms_raw$type)

# import text mining package "tm"
library(tm)

# build a Volatiel Corpora containing the SMS messages in the training data
sms_corpus <- VCorpus(VectorSource(sms_raw$text))

# check corpora
print(sms_corpus)
inspect(sms_corpus[1:3])
view1[[1]]$content
view1[[2]]$content
view1[[3]]$content
# we can also use as.character(sms_corpus[[1]]) to check the content
as.character(sms_corpus[[1]])
# and also can use lapply(sms_corpus[1:3], as.character) to check the content
lapply(sms_corpus[1:3], as.character)

# convert all of the SMS messages to lowercase and remove any numbers:
sms_corpus_clean <- tm_map(sms_corpus, content_transformer(tolower))
sms_corpus_clean <- tm_map(sms_corpus_clean, removeNumbers)

# delete stop words
sms_corpus_clean <- tm_map(sms_corpus_clean, removeWords, stopwords())

# define replace function
# replacePunctuation <- function(x) { gsub("[[:punct:]]+", " ", x) }

# remove punctuation
sms_corpus_clean <- tm_map(sms_corpus_clean, removePunctuation)

# install SnowballC package, which can handle same word with different type
library(SnowballC)
wordStem(c("learn", "learned", "learning", "learns"))

# make different words' type to single one
sms_corpus_clean <- tm_map(sms_corpus_clean, stemDocument)

# remove additional whitespace, leave a single one
sms_corpus_clean <- tm_map(sms_corpus_clean, stripWhitespace)


# create a sparse matrix
sms_dtm <- DocumentTermMatrix(sms_corpus_clean)

# create train set and test set
sms_dtm_train <- sms_dtm[1:4169, ]
sms_dtm_test  <- sms_dtm[4170:5559, ]

# keep classical label
sms_train_labels <- sms_raw[1:4169, ]$type
sms_test_labels  <- sms_raw[4170:5559, ]$type

# word cloud
library(wordcloud)
# wordcloud example, input 26 letters, the frequency is 1 to 26 according to a to z.
# wordcloud(c(letters), seq(1, 26),random.order=F)
# wordcloud example, show three words which the most frequency
# wordcloud(iris$Species, max.words=3)

# wordcloud can support corpus type. a word appear more than 50 times can be showed in cloud
wordcloud(sms_corpus_clean, min.freq = 50, random.order = FALSE)

# make the train set into two sets, discovery the word cloud
spam <- subset(sms_raw, type == "spam")
ham  <- subset(sms_raw, type == "ham")
# the typeface is 0.5 to 3
wordcloud(spam$text, max.words = 40, scale = c(3, 0.5))
wordcloud(ham$text, max.words = 40, scale = c(3, 0.5))

# find and save these words which frequency greater than 5
sms_freq_words <- findFreqTerms(sms_dtm_train, 5)

# use the high frequency words to build a new sparse matrix
sms_dtm_freq_train <- sms_dtm_train[ , sms_freq_words]
sms_dtm_freq_test <- sms_dtm_test[ , sms_freq_words]

# change frequency to "yes" and "no"
convert_counts <- function(x) {
```

```
  x <- ifelse(x > 0, "Yes", "No")
}

# make the change, margin=2 means use for columns
sms_train <- apply(sms_dtm_freq_train, MARGIN = 2,
convert_counts)
sms_test  <- apply(sms_dtm_freq_test, MARGIN = 2,
convert_counts)


# use e1071 package to do naive byes analyzation
library(e1071)
sms_classifier <- naiveBayes(sms_train, sms_train_labels)

# evaluate the performance
sms_test_pred <- predict(sms_classifier, sms_test)

# use crosstable to show the performance
library(gmodels)
CrossTable(sms_test_pred, sms_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))

# boost property
sms_classifier2 <- naiveBayes(sms_train, sms_train_labels, laplace =
1)
sms_test_pred2 <- predict(sms_classifier2, sms_test)
CrossTable(sms_test_pred2, sms_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

Result:

1) No Laplace estimator

Cross table:

```
Total Observations in Table:  1390


             | 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 |           |
-------------|-----------|-----------|-----------|
```

2) Add Laplace estimator

Cross table:

```
Total Observations in Table:  1390


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1202 |        28 |      1230 |
             |     0.996 |     0.153 |           |
-------------|-----------|-----------|-----------|
        spam |         5 |       155 |       160 |
             |     0.004 |     0.847 |           |
-------------|-----------|-----------|-----------|
Column Total |      1207 |       183 |      1390 |
             |     0.868 |     0.132 |           |
-------------|-----------|-----------|-----------|
```

When use the Laplace estimator, the result becomes better.