Naive Bayes Text Classification in R

Rob McCulloch

1/22/2018

Contents

Overview

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

Let's review the Naive Bayes analysis of the sms text data done in R.

A sms document is *ham* if it is a message you want, (sent by someone you know) and *spam* if it is sent by an advertiser or intruder.

The goal is to be able to guess (predict) whether a text message document is ham or spam from the contents of the document.

Each document is a sms (short message service, a short little text message).

Each document in our data set is *labelled* as ham or spam.

So y=ham/spam and x=the document.

We have *classification problem*, we are trying to predict a categorical binary y from x.

We will:

- read in the data
- clean the text documents (e.g get rid of stop words)
- get a document term matrix
- throw out infrequently occurring terms
- convert counts to a binary indicating whether or not a term is in the document
- train/test split
- fit Naive Bayes on train, predict on test
- get our out-of-sample miss-classification rate

Read in sms Data and Wrangle

Read in Data

Let's read in the data and then have a quick look at it.

We convert the *type* variable which is ham/spam into a factor, which is the R way of saying it is a categorical variable.

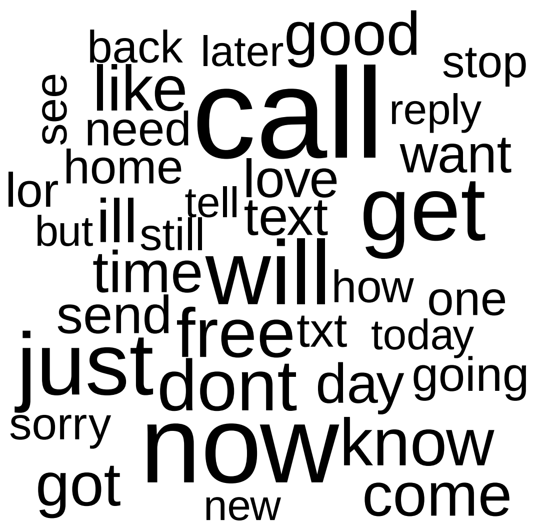
```
# read in data
smsRaw = read.csv("http://www.rob-mcculloch.org/data/sms_spam.csv", stringsAsFactors = FALSE)
# convert spam/ham to factor.
smsRaw$type = factor(smsRaw$type)
```

Let's have a quick look at the data.

```
#smRaw is a data frame
dim(smsRaw)
names(smsRaw)
smsRaw$type[1:5]
smsRaw$text[1:5]

#look at y=type
print(table(smsRaw$type))

#look at x=words
library(wordcloud)
wordcloud(smsRaw$text, max.words = 40)
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):
## transformation drops documents
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
## Warning in wordcloud(smsRaw$text, max.words = 40): can could not be fit on
## page. It will not be plotted.
## Warning in wordcloud(smsRaw$text, max.words = 40): you could not be fit on
## page. It will not be plotted.
## Warning in wordcloud(smsRaw$text, max.words = 40): mobile could not be fit
## on page. It will not be plotted.
```



Make and Clean Corpus

We use the `tm` package to clean up the text.

```
# build a corpus using the text mining (tm) package
library(tm)
library(SnowballC)
#volatile (in memory corpus from vector of text in R
smsC = VCorpus(VectorSource(smsRaw$text))
# clean up the corpus using tm_map()
smsCC = tm_map(smsC, content_transformer(tolower)) #upper -> lower
smsCC = tm_map(smsCC, removeNumbers) # remove numbers
smsCC = tm_map(smsCC, removeWords, stopwords()) # remove stop words
smsCC = tm_map(smsCC, removePunctuation) # remove punctuation
smsCC = tm_map(smsCC, stemDocument) #stemming
smsCC = tm_map(smsCC, stripWhitespace) # eliminate unneeded whitespace
```

```
#see the first text message after cleaning
smsCC[[1]]$content
## [1] "hope good week just check"
# first message before cleaning
smsRaw$text[1]
## [1] "Hope you are having a good week. Just checking in"
```

Get Document Term Matrix

The document term matrix with have rows corresponding to documents and columns corresponding to terms. The (i, j) element of the matrix is the number of times the j^{th} term is in the i^{th} document.

```r
# create Document Term Matrix
smsDtm = DocumentTermMatrix(smsCC)
dim(smsDtm)
## [1] 5559 6559
```

Split Data into Train/Test, Throw Out Infrequent Terms, Convert Term Counts to Binary

Train and Test

We divide our data in train and test sub-samples.

We use the train data to estimate or *train* our classifier and we evaluate its performance on the test data.

```r
# creating training and test datasets
smsTrain = smsDtm[1:4169, ]
smsTest  = smsDtm[4170:5559, ]
smsTrainy = smsRaw[1:4169, ]$type
smsTesty  = smsRaw[4170:5559, ]$type
cat("training fraction is: ",4169/5559,"\n")
## training fraction is:  0.749955
```

Freq Words and Convert Counts to Binary

We throw out the terms (columns of the Dtm) such that the term comes up less than 5 times over all documents.

Then we convert the count to a simple binary indicator Yes/No indicating whether or not the term is in the document.

```r
smsFreqWords = findFreqTerms(smsTrain, 5) #words that appear at least 5 times
smsFreqTrain = smsTrain[ , smsFreqWords]
smsFreqTest = smsTest[ , smsFreqWords]

convertCounts <- function(x) {
  x <- ifelse(x > 0, "Yes", "No")
}
# apply() convert_counts() to columns of train/test data
smsTrain = apply(smsFreqTrain, MARGIN = 2, convertCounts)
smsTest  = apply(smsFreqTest, MARGIN = 2, convertCounts)

#check basic properties
dim(smsTrain)
## [1] 4169 1139
is.matrix(smsTrain)
## [1] TRUE
smsTrain[1:3,1:5]
```

```
##     Terms
## Docs £wk  €~m  €~s  abiola abl
##    1 "No" "No" "No" "No"   "No"
##    2 "No" "No" "No" "No"   "No"
##    3 "No" "No" "No" "No"   "No"
```

Naive Bayes and Missclassification

Now we are ready to do Naive Bayes classification using the `e1071` R package.

```
library(e1071)
smsNB = naiveBayes(smsTrain, smsTrainy, laplace=1)
yhat = predict(smsNB,smsTest)
ctab = table(yhat,smsTesty)
ctab
##       smsTesty
## yhat    ham spam
##   ham  1202   28
##   spam    5  155
misclass = (sum(ctab)-sum(diag(ctab)))/sum(ctab)
perspam = ctab[2,2]/sum(ctab[,2])
cat("misclass,perspam: ", misclass,perspam,"\n")
## misclass,perspam:  0.02374101 0.8469945
```