```
---
title: 'Naive Bayes classification: spam/ham'
author: "Zijian Du"
date: "January 23, 2018"
output:
  html_document: default
  pdf_document: default
  word_document: default
---
```

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

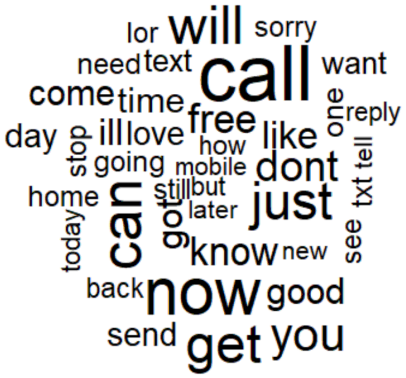
R Markdown
this project use Naive Bayes to implement spam classifier and evaluate the influence of random splitting of train/test split, as well as the influence of value of laplace on final test accuracy.

```{r}
# read in data
smsRaw = read.csv("sms_spam.csv", stringsAsFactors = FALSE)

# convert spam/ham to factor.
smsRaw$type=factor(smsRaw$type)

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

```
  ham spam
 4812  747
```

import wordcloud
```{r}
library(wordcloud)
#install.packages("tm")
```

load the wordcloud package and display wordfrequency
```{r}
wordcloud(smsRaw$text, max.words = 40)
```




```{r}
# 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

```

get document term matrix

```{r}
# create document term matrix
smsDtm <- DocumentTermMatrix(smsC, control = list(
tolower = TRUE,
removeNumbers = TRUE,
stopwords = TRUE,
removePunctuation = TRUE,
stemming = TRUE
))
dim(smsDtm)
```

```
[1] 5559 6971
```

Out of sample misclassification

```{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
```

frequency words and convert counts to binary value

```{r}
smsFreqWords = findFreqTerms(smsTrain, 5)
#words that appear at leat 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)
```

```{r}
library(e1071)
smsNB = naiveBayes(smsTrain, smsTrainy, laplace=1)
yhat = predict(smsNB,smsTest)
ctab = table(yhat,smsTesty)
ctab
misclass = (sum(ctab)-sum(diag(ctab)))/sum(ctab)
perspam = ctab[2,2]/sum(ctab[,2])
cat("misclass,perspam: ", misclass,perspam,"\n")
```

```
      smsTesty
yhat    ham spam
  ham  1202   29
  spam    5  154
misclass,perspam:  0.02446043 0.8415301
```

Now we iterate through random train test split and change the laplace value and report confusion matrixs and misclassification rate for each combination

```{r}
# sample train/test
trainfrac=.75
n=length(smsRaw$type)
nTrain = floor(trainfrac*n)
result_missclassification <-matrix(, nrow=10,ncol=10)
result_perspam <-matrix(, nrow=10,ncol=10)
for(traintest in 1:10)
{
  for (laplace in 1:10)
{
set.seed(traintest)
ii = sample(1:n,nTrain)
smsTrain = smsDtm[ii, ]
smsTest = smsDtm[-ii, ]
smsTrainy = smsRaw[ii, ]$type
smsTesty = smsRaw[-ii, ]$type
# freq words
smsFreqWords = findFreqTerms(smsTrain, 5) #words that appear at leat 5 times
smsFreqTrain = smsTrain[ , smsFreqWords]
smsFreqTest = smsTest[ , smsFreqWords]
# counts -> binary
smsTrain = apply(smsFreqTrain, MARGIN = 2, convertCounts)
smsTest = apply(smsFreqTest, MARGIN = 2, convertCounts)
```

```
smsNB = naiveBayes(smsTrain, smsTrainy, laplace)
#pred and misclass
yhat = predict(smsNB,smsTest)
ctab = table(yhat,smsTesty)
cat("Confusion matrix and mis-classification for ", traintest," th split and laplace = ",laplace,"\n")
print(ctab)
missclass = (sum(ctab)-sum(diag(ctab)))/sum(ctab)
perspam = ctab[2,2]/sum(ctab[,2])
result_missclassification[traintest, laplace] = missclass
result_perspam[traintest, laplace] = perspam
cat("misclass,perspam: ", missclass,perspam,"\n")
}
}
```

dataframe showing total missclassfication rate for combination of laplace and different splits

```{r}
print("total misclassification result for laplace-split combination")
result_missclassification<-data.frame(result_missclassification)
rownames(result_missclassification)<-c("split 1","split 2","split 3","split 4","split 5","split 6","split 7","split 8","split 9",
"split10")
colnames(result_missclassification)<-c("laplace 1","laplace 2","laplace 3","laplace 4","laplace 5","laplace 6","laplace 7","laplace
8","laplace 9", "laplace 10")
print(result_missclassification)
```

R Console

0.01438849
0.02733813
data.frame
10 x 10

	laplace 1 <dbl>	laplace 2 <dbl>	laplace 3 <dbl>	laplace 4 <dbl>	laplace 5 <dbl>	laplace 6 <dbl>	laplace 7 <dbl>	laplace 8 <dbl>	laplace 9 <dbl>
split 1	0.018705036	0.02230216	0.02733813	0.03309353	0.03956835	0.04676259	0.05179856	0.05899281	0.06978417
split 2	0.024460432	0.02877698	0.03525180	0.04028777	0.04748201	0.05539568	0.06834532	0.08129496	0.08705036
split 3	0.017985612	0.01942446	0.02446043	0.02733813	0.03237410	0.04028777	0.04676259	0.05971223	0.06906475
split 4	0.016546763	0.01870504	0.02446043	0.02949640	0.03381295	0.04316547	0.04820144	0.05827338	0.06187050
split 5	0.024460432	0.03021583	0.03525180	0.03956835	0.04244604	0.04676259	0.05323741	0.06187050	0.06762590
split 6	0.009352518	0.01438849	0.02158273	0.02446043	0.03381295	0.04388489	0.05251799	0.06546763	0.07625899
split 7	0.025179856	0.02733813	0.03237410	0.03453237	0.04028777	0.04532374	0.05755396	0.06546763	0.06906475
split 8	0.022302158	0.02949640	0.03525180	0.04172662	0.04388489	0.05323741	0.06043165	0.07194245	0.07841727
split 9	0.020143885	0.02446043	0.02805755	0.03381295	0.03741007	0.04244604	0.05035971	0.05971223	0.06618705
split10	0.026618705	0.03021583	0.03453237	0.04100719	0.04964029	0.05395683	0.06546763	0.07697842	0.08273381

1-10 of 10 rows | 1-10 of 10 columns

plot and visulize the missclassification

```{r}
print("total perspam accuracy result for laplace-split combination")
result_perspam<-data.frame(result_perspam)
rownames(result_perspam)<-c("split 1","split 2","split 3","split 4","split 5","split 6","split 7","split 8","split 9", "split10")
colnames(result_perspam)<-c("laplace 1","laplace 2","laplace 3","laplace 4","laplace 5","laplace 6","laplace 7","laplace 8","laplace
9", "laplace 10")
print(result_perspam)
```

R Console

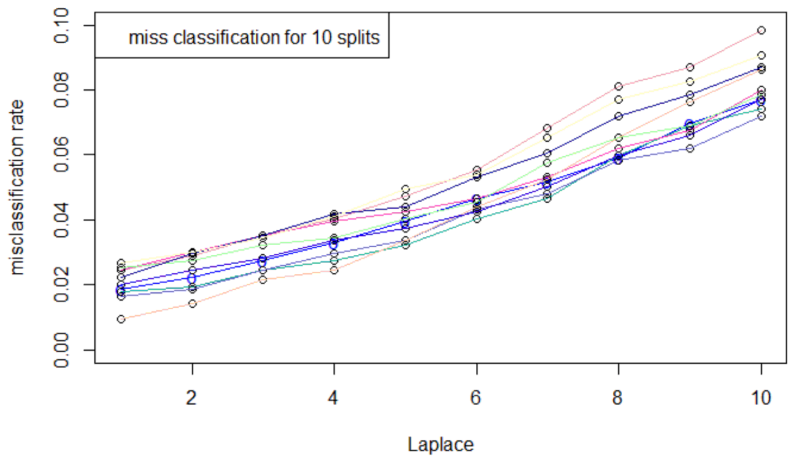
0.9072165
0.8097826
data.frame
10 x 10

	laplace 1 <dbl>	laplace 2 <dbl>	laplace 3 <dbl>	laplace 4 <dbl>	laplace 5 <dbl>	laplace 6 <dbl>	laplace 7 <dbl>	laplace 8 <dbl>	laplace 9 <dbl>
split 1	0.8728324	0.8497110	0.8092486	0.7630058	0.7109827	0.6531792	0.6011561	0.5433526	0.4508671
split 2	0.8333333	0.8080808	0.7626263	0.7272727	0.6767677	0.6262626	0.5353535	0.4444444	0.4040404
split 3	0.8720930	0.8546512	0.8313953	0.8023256	0.7616279	0.6976744	0.6453488	0.5406977	0.4651163
split 4	0.8870056	0.8700565	0.8361582	0.7853107	0.7514124	0.6779661	0.6384181	0.5593220	0.5310734
split 5	0.8287293	0.7955801	0.7569061	0.7292818	0.7016575	0.6574586	0.6132597	0.5414365	0.4972376
split 6	0.9381443	0.9072165	0.8608247	0.8402062	0.7783505	0.7061856	0.6391753	0.5463918	0.4690722
split 7	0.8315217	0.8097826	0.7771739	0.7663043	0.7228261	0.6739130	0.5815217	0.5217391	0.4891304
split 8	0.8605769	0.8269231	0.7836538	0.7403846	0.7259615	0.6586538	0.6105769	0.5336538	0.4855769
split 9	0.8546512	0.8197674	0.7906977	0.7441860	0.7151163	0.6686047	0.5988372	0.5232558	0.4709302
split10	0.8274112	0.7969543	0.7715736	0.7258883	0.6649746	0.6243655	0.5431472	0.4619289	0.4213198

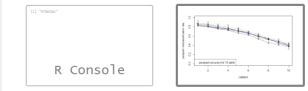
1-10 of 10 rows | 1-10 of 10 columns

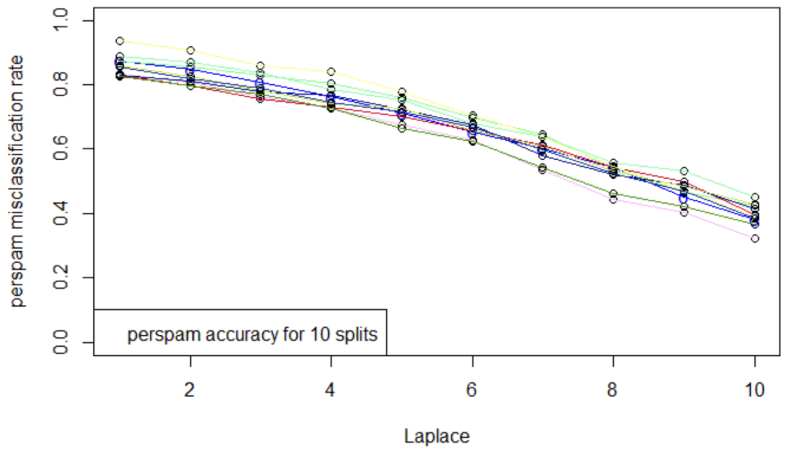
```{r}
library(igraph)
library(randomcoloR)
randomColor(count = 1, hue = c(" ", "random", "red", "orange", "yellow",
"green", "blue", "purple", "pink", "monochrome"), luminosity = c(" ",
"random", "light", "bright", "dark"))
x_laplace = c(1:10)
plot(x_laplace, result_missclassification[1,], xlab="Laplace",ylab="misclassification rate", type="o", col="blue", pch="o", lty=1,
ylim=c(0,0.1))
legend("topleft", "miss classification for 10 splits")
for (i in 2:10){
  points(x_laplace,result_missclassification[i,],col="black")
  lines(x_laplace, result_missclassification[i,], col=randomColor())
}
```

R Console



```
randomColor(count = 1, hue = c(" ", "random", "red", "orange", "yellow",
"green", "blue", "purple", "pink", "monochrome"), luminosity = c(" ",
"random", "light", "bright", "dark"))
x_laplace = c(1:10)
plot(x_laplace, result_perspam[1,], xlab="Laplace",ylab="perspam misclassification rate", type="o", col="blue", pch="o", lty=1,
ylim=c(0,1))
legend("bottomleft", "perspam accuracy for 10 splits")
for (i in 2:10)
{
  points(x_laplace,result_perspam[i,],col="black")
  lines(x_laplace, result_perspam[i,], col=randomColor())
}
```





Observations
1. final classification results do fluctuate with different train/test split, which can cause around 0.02 deviation for total missclassification
and around 0.1 deviation for perspam misclassification
2. with increase of laplace, total missclassification rate increases
3. with increase of laplace, perspam predicting accuracy decreases
Hence when doing laplace smoothing, we should use a small laplace value to filter out the zero conditional probability for unseen data while increasing the smoothing will make the classifier perform worse.