Homework 1 for **EECS E6720**

submitted to Professor John Paisley

Advait Rajagopal

1 October 2017

1 Question 1

This is an old problem often called the Monty Hall problem. We can solve this problem using Bayes theorem.
Before my friend has even started the game, her chance of winning a car is 1/3 as there are three doors with an equal probability of having the car behind them. Let us define the following events;

$$\begin{aligned} E_1 &= \text{Door 1 has the car behind it} \\ E_2 &= \text{Door 2 has the car behind it} \\ E_3 &= \text{Door 3 has the car behind it} \\ E &= \text{Event of finding the car behind the chosen door} \end{aligned}$$

So we know that $P(E_1) = P(E_2) = P(E_3) = 1/3$. This explains our prior probabilities. Now we need to develop our likelihood based on the evidence that we are presented which is that one door is opened. We know that the door with a car will never be opened. We also know that the door my friend picks is never opened. So there are only two doors that can be opened making the probability of evidence or the marginal likelihood $P(E) = 1/2$. This means that the event of finding a car behind a chosen door is $P(E)$.
Let us start with my friend picking door 1. Let us also say that the gameshow host opens door 3. Now we know that the host won't open the door with the car so we aren't interested in event E_3 any more. The likelihood is the probability gameshow host chooses door 3 given the car is behind the first door and is given by $P(E|E_1) = 1/2$. So let us calculate the posterior probability of the car being behind door 1 the originally picked door.

$$\begin{aligned} P(E_1|E) &= \frac{P(E|E_1)P(E_1)}{P(E)} \\ &= \frac{1/2 * 1/3}{1/2} \\ &= 1/3 \end{aligned}$$

So the probability of the car being behind the first door is 1/3 so if my friend stays with her original choice she has a one in third chance of winning the car.

Now the marginal likelihood $P(E)$ stays the same as this is independent of the hypothesis we are examining. The prior is also independent of any evidence so $P(E_2)$ this stays the same as well. Now suppose that my friend picks door 1 and the car is behind door 2, Monty cannot choose any of these doors and so will choose door 3. In this case my friend will definitely win because the likelihood is $P(E|E_2) = 1$. So given the prior and likelihood and evidence we update and calculate our posterior probability;

$$\begin{aligned} P(E_2|E) &= \frac{P(E|E_2)P(E_2)}{P(E)} \\ &= \frac{1 * 1/3}{1/2} \\ &= 2/3 \end{aligned}$$

So if my friend changes her choice after the first door is opened she doubles her chances of winning and this is the advice I'd give her if she called, **she should switch her choice**.

2 Question 2

We are given the following information about a parameter vector π. $\pi = (\pi_1, ..., \pi_K)$ and $\pi_k \geq 0$ such that $\sum_k \pi_k = 1$. We are also given the following information about the data X, where we are told that $X_i \sim$ Multinomial(π), i.i.d for $i = 1, .., N$.
If we are to find a conjugate prior for this likelihood then we need to ensure that the prior and the posterior have the same functional form or follow the same distribution. We find that using a Dirichlet prior on π will fulfill this requirement. If this is the case, our model is as follows;

$$\begin{aligned} \text{Prior} : \ & p(\pi|\alpha) \sim \text{Dirichlet}(\alpha) \\ \text{Likelihood} : \ & p(X|\pi) \sim \text{Multinomial}(\pi) \\ \text{Posterior} : \ & p(\pi|X, \alpha) \propto p(\pi|\alpha)p(X|\pi) \end{aligned}$$

We suppress the parameter α in the likelihood notation because α affects the data generating process only through the realization of π. We can show that the posterior is proportional to a Dirichlet distribution with the derivation that follows.
The prior distribution is given by;

$$p(\pi|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)...\Gamma(\alpha_K)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}$$

where;

$$\alpha_0 = \sum_{k=1}^{K} \alpha_K$$

The multinomial distribution's mass function can be written using the gamma function as;

$$\begin{aligned} p(X|\pi) &\propto \prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{x_{nk}} \\ &\propto \prod_{k=1}^{K}\pi_k^{(\sum_n x_{nk})} \\ &\propto \prod_{k=1}^{K}\pi_k^{m_k} \\ &= \binom{N}{m_1 m_2 ... m_K}\prod_{k=1}^{K}\pi_k^{m_k} \end{aligned}$$

In the last step we introduce the normalizing constant to complete the likelihood function. This represents the number of ways of partitioning N objects into K groups of size $m_1, m_2, ..., m_K$ and is given by;

$$\binom{N}{m_1 m_2 ... m_K} = \frac{N!}{m_1!m_2!..m_K!}$$

Now the posterior is proportional to the products of the prior and the likelihood;

$$\begin{aligned} p(\pi|X,\alpha) &\propto p(\pi|\alpha)p(X|\pi) \\ &\propto \prod_{k=1}^{K}\pi_k^{\alpha_k+m_k-1} \end{aligned}$$

Where we notice that this last line looks like a Dirichlet distribution if we can find the appropriate normalizing constant. The full Dirichlet distribution of the posterior becomes;

$$\begin{aligned} p(\pi|X,\alpha) &= \frac{\Gamma(\alpha_0+N)}{\Gamma(\alpha_1+m_1)...\Gamma(\alpha_K+m_K)}\prod_{k=1}^{K}\pi_k^{\alpha_k+m_k-1} \\ p(\pi|X,\alpha) &\sim Dirichlet(\pi|\boldsymbol{\alpha}+\mathbf{m}) \end{aligned}$$

We name this the **Dirichlet distribution**. The parameter $\mathbf{m} = (m_1, ..., m_k)^{\mathrm{T}}$. The α_k parameters can be interpreted as number of observations where $x_k = 1$.

3 Question 3

We are given a dataset $\{x_1, ...x_N\}$, where each $x \in \mathbb{N}$. It is i.i.d and follows a Poisson(λ) distribution. We model λ as $\lambda \sim$ Gamma(a, b).

3.1 Part A

The posterior distribution is $p(\lambda|X, a, b)$. The likelihood is $p(X|\lambda)$ and the prior distribution on the parameter of interest λ is $p(\lambda|a, b)$ and they are distributed as below;

$$\begin{aligned} \text{Prior}: \ & p(\lambda|a,b) \sim \text{Gamma}(a,b) \\ \text{Likelihood}: \ & p(X|\lambda) \sim \text{Poisson}(\lambda) \\ \text{Posterior}: \ & p(\lambda|X,a,b) \propto p(\lambda|a,b)p(X|\lambda) \end{aligned}$$

The prior distribution has the following functional form;

$$p(\lambda|a,b) = \frac{b^a}{\Gamma(a)}\lambda^{a-1}e^{-b\lambda}$$

The joint likelihood has the following functional form;

$$\begin{aligned} p(X|\lambda) &= \prod_{i=1}^{N} \frac{\lambda^{x_i}e^{-\lambda}}{x_i!} \\ &= \frac{\lambda^{\sum_i x_i} e^{-N\lambda}}{\prod_{i=1}^{N} x_i!} \end{aligned}$$

Then the posterior is given by the product of the prior and the joint likelihood;

$$\begin{aligned} p(\lambda|X,a,b) &\propto p(\lambda|a,b)p(X|\lambda) \\ &\propto \frac{b^a}{\Gamma(a)}\lambda^{a-1}e^{-b\lambda} \cdot \frac{\lambda^{\sum_i x_i} e^{-N\lambda}}{\prod_{i=1}^{N} x_i!} \\ &\propto \frac{b^a}{\Gamma(a)\prod_{i=1}^{N} x_i!}\lambda^{\sum_i x_i + a - 1} e^{-(N+b)\lambda} \\ &\propto \lambda^{\sum_i x_i + a - 1} e^{-(N+b)\lambda} \\ \implies p(X|\lambda) &\sim \text{Gamma}\Big(\sum_i x_i + a, N+b\Big) \end{aligned}$$

So we see that the posterior distribution $\mathbf{p}(\lambda|\mathbf{X}, \mathbf{a}, \mathbf{b})$ **follows a Gamma distribution** with the parameters as made explicit in the last line of the previous derivation. We can search for the appropriate normalizing constant but since we are interested in the distribution of λ we can disregard those terms for now and a distribution proportional to the true posterior is sufficient.

3.2 Part B

We are told the posterior predictive distribution is given by the following[1];

$$\begin{aligned} p(x^*|x_1,...,x_N) &= \int_0^\infty p(x^*|\lambda)p(\lambda|x_1,...,x_N)d\lambda \\ &= \int_0^\infty \left[\frac{\lambda^{x^*}e^{-\lambda}}{x^*!}\right] \cdot \left[\frac{(N+b)^{\sum_i x_i + a}}{\Gamma(\sum_i x_i + a)}\lambda^{\sum_i x_i + a - 1} e^{-(N+b)\lambda}\right] d\lambda \\ &= \frac{(N+b)^{\sum_i x_i + a}}{\Gamma(\sum_i x_i + a)\Gamma(x^*+1)} \int_0^\infty \lambda^{x^* + \sum_i x_i + a - 1} e^{-(N+b+1)\lambda} d\lambda \\ &= \frac{(N+b)^{\sum_i x_i + a}}{\Gamma(\sum_i x_i + a)\Gamma(x^*+1)} \cdot \frac{\Gamma(x^* + \sum_i x_i + a)}{(N+b+1)^{x^* + \sum_i x_i + a}} \\ &= \frac{\Gamma(x^* + \sum_i x_i + a)}{\Gamma(\sum_i x_i + a)\Gamma(x^*+1)} \left(\frac{N+b}{N+b+1}\right)^{\sum_i x_i + a} \left(\frac{1}{N+b+1}\right)^{x^*} \end{aligned}$$

[1]note that $n! = \Gamma(n+1)$

This is the pmf of a negative binomial distribution $p(x^*|x_1, ..., x_n) \sim \text{NB}(\sum_i x_i + a, N + b)$.

4 Question 4

4.1 Part A

I implement the naive Bayes classifier for spam filtering in R, which is my preferred language. The code is appended to the end of this document as well as the source code is attached with the submission.

4.2 Part B

The table below is the confusion matrix. I drop 7 emails in the test set that resulted in NaN values for the likelihood. This makes the total number of "test" emails 454. After this I have an accuracy of 90%.

Prediction

		spam	**ham**	**total**
	spam	TP = 172	FN = 9	181
Truth	**ham**	FP = 33	TN = 240	273
	total	205	249	

4.3 Part C

I pick three misclassified emails and make the desired plot. I arbitrarily pick two false positives and one false negative. I plot these in Figure 1. I notice that when the probability of spam (λ_1) and probability of ham (λ_0) peak together, there is a high concentration of that word. This means the black, red and blue lines often peak together. I suspect this is what confuses the algorithm.

Table 1: Predictive Probabilities

Email	spam	ham
155	1.608e-30	1
168	1.676e-33	1
372	1	7.073e-24

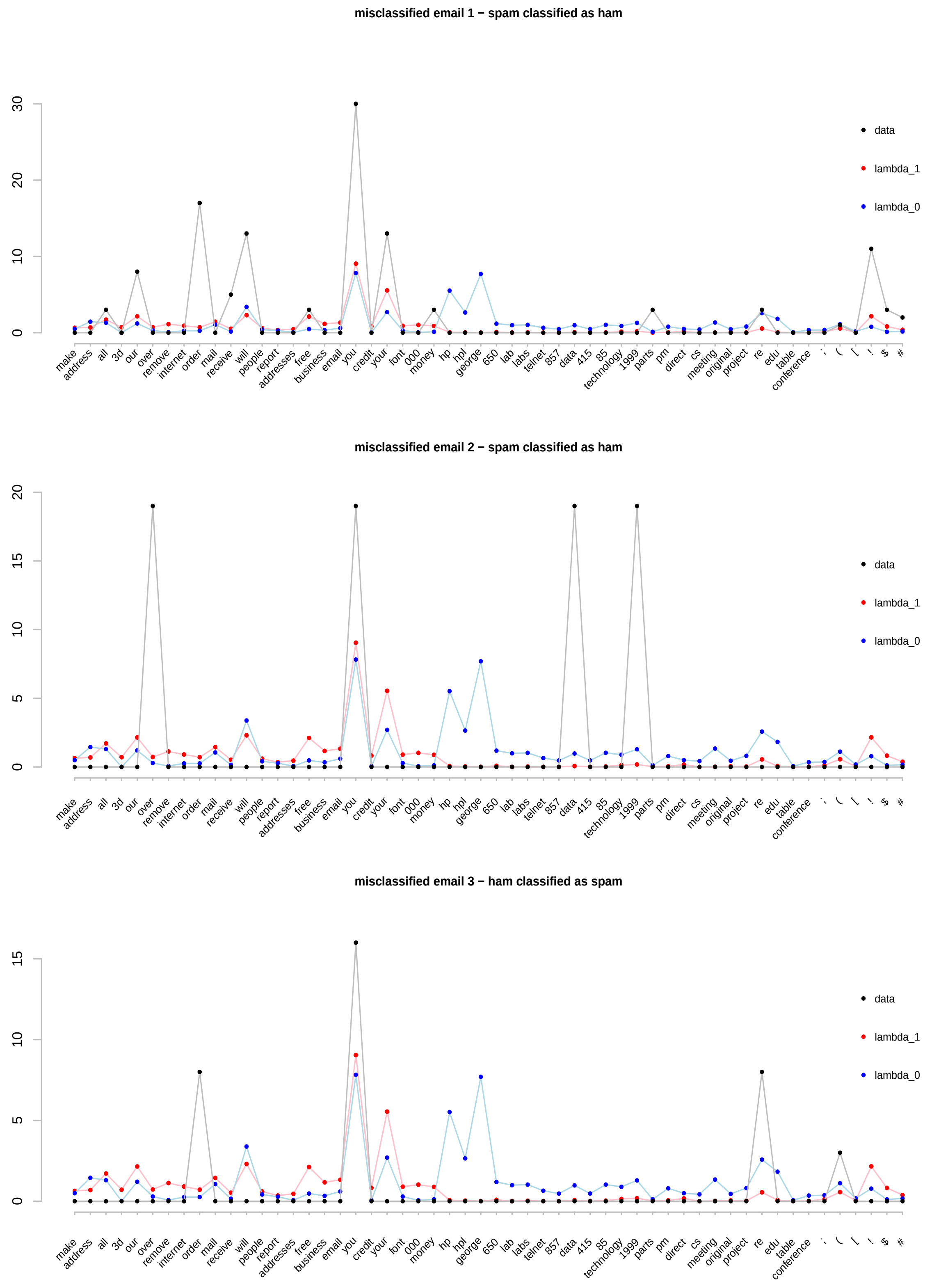

Figure 1: *The black dots indicate the feature value corresponding to each word in the vocabulary. There are 54 of these words. The red and blue dots indicate posterior values of* $E[\lambda_1]$ *and* $E[\lambda_0]$ *respectively.*

4.4 Part D

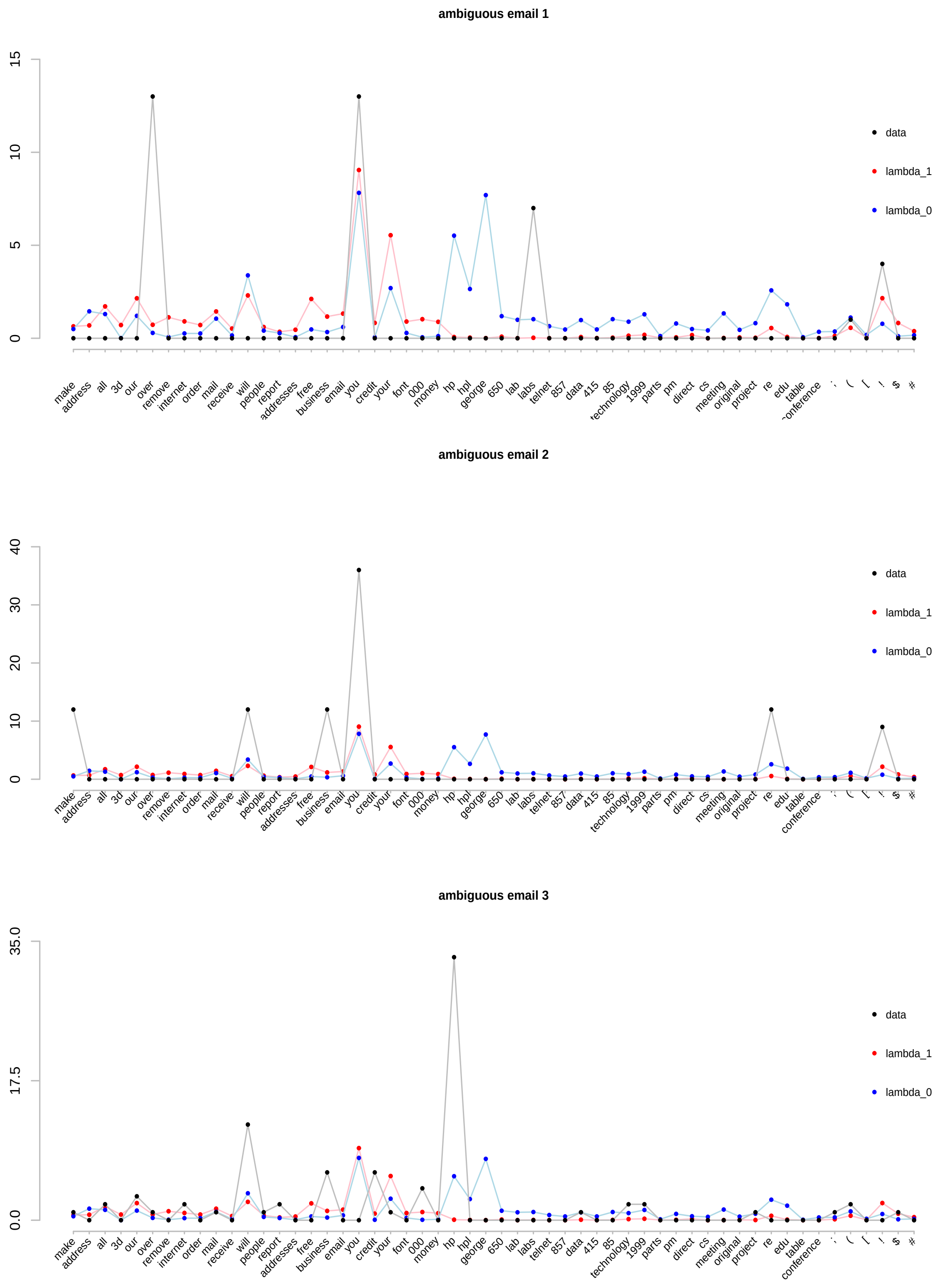

Figure 2: *The black dots indicate the feature value corresponding to each word in the vocabulary. There are 54 of these words. The red and blue dots indicate posterior values of* $E[\lambda_1]$ *and* $E[\lambda_0]$ *respectively.*

I pick 3 ambiguously classified emails (code appended) and plot their features along with probability of spam (λ_1) and probability of ham (λ_0) in Figure 2. The predictive probabilities of spam and ham are displayed in Table 2.

Table 2: Predictive Probabilities

Email	spam	ham
4	5.2e-01	0.47
130	2.2e-05	0.99
266	9.9e-01	1.3e-04

5 Code

Listing 1: Reading in data

```
#rm(list = ls())
#packages
library(dplyr)
##############################
list.files()
setwd("/Users/Advait/Desktop/New_School/Fall17/
␣␣␣␣␣␣␣␣␣Bayes_ML/HW_1/EECS6720-hw1-data")
#54 should be the number of words and X
   is called a 54 dimensional vector.

#load the train feature matrix
X_train <- read.csv("X_train.csv",
                    header = F)
dim(X_train)

#load the test feature matrix
X_test <- read.csv("X_test.csv",
                   header = F)
dim(X_test)

#load the test labels
y_train <- read.csv("label_train.csv",
                   header = F)
dim(y_train)

#load the test labels
y_test <- read.csv("label_test.csv",
                   header = F)
dim(y_test)
#vector of 460 rows
```

Listing 2: The naive Bayes classifier

```
#########################
#Naive Bayes Classifier#
#########################
adv8_naiveBayes <- function(new_data,
                            data){
  spam_counter <- vector()
  ham_counter <- vector()
  spam_probs <- vector()
  ham_probs <- vector()
  a <- 1
  b <- 1
  e <- 1
  f <- 1
  N <- 4140
  spam_features <- data[data[,1] == 1,][-1]
  spam_labels <- data[data[,1] == 1,][1]
  ham_features <- data[data[,1] == 0,][-1]
  ham_labels <- data[data[,1] == 0,][1]

  #the prior part
  #(spam,ham)
  p_ystar <- rep(0, 2)
  p_ystar[1] <- (e + dim(spam_features)[1])/(N + e + f)
  p_ystar[2] <- (f + dim(ham_features)[1])/(N + e + f)

  #starting a for loop for x_star
  for(i in 1:dim(new_data)[1]){
  x_star <- new_data[i,]

  #for spam emails
  summation_x_1 <- colSums(spam_features[,1:ncol(spam_features)])
  spam_numerator <- x_star + summation_x_1 + a
  spam_denominator_1 <- summation_x_1 + a
  spam_denominator_2 <- x_star + 1
  p_x_star_1 <- (lgamma(spam_numerator)
                 + (summation_x_1)*log((N + b)/(N+b+1))
                 + x_star*log(1/(N+b+1))
                 - lgamma(spam_denominator_1)
                 - lgamma(spam_denominator_2))
  p_y_star_1 <- exp(sum(p_x_star_1))*p_ystar[1]
```

Listing 3: The naive Bayes classifier contd.

```
 #for ham emails
  summation_x_0 <- colSums(ham_features[,1:ncol(ham_features)])
  ham_numerator <- x_star + summation_x_0 + a
  ham_denominator_1 <- summation_x_0 + a
  ham_denominator_2 <- x_star + 1
  p_x_star_0 <- (lgamma(ham_numerator)
                + (summation_x_1)*log((N + b)/(N+b+1))
                + x_star*log(1/(N+b+1))
                - lgamma(ham_denominator_1)
                - lgamma(ham_denominator_2))
  p_y_star_0 <- exp(sum(p_x_star_0))*p_ystar[2]

  #return information
  p_spam <- p_y_star_1/(p_y_star_1 + p_y_star_0)
  p_ham <- p_y_star_0/(p_y_star_1 + p_y_star_0)

  # pred_prob <- c(p_spam,p_ham)
  if(is.nan(p_spam) == TRUE | is.nan(p_ham) == TRUE){
    cat("email␣number",i,"is␣messed␣up","\n")
    next
  }
  spam_probs <- c(spam_probs,p_spam)
  ham_probs <- c(ham_probs,p_ham)
  if(p_spam >= p_ham){
    spam_counter <- c(spam_counter,1)
    ham_counter <- c(ham_counter,1000)
  }
  if(p_spam < p_ham){
    ham_counter <- c(ham_counter,1)
    spam_counter <- c(spam_counter,1000)
  }
  }
  return(cbind(spam_counter,ham_counter,spam_probs,ham_probs))
}
```

Listing 4: Setting up data and calling classifier function

```
#setting up data
spam <- cbind(y_train,X_train)
colnames(spam)[1] <- "labels"

#predictions
predictions <- adv8_naiveBayes(new_data = X_test,
                               data = spam)

#deleting the messed up emails as informed by my function
y_test <- y_test[-c(164,228,264,326,350,383,403),]
length(y_test)

#making  a data frame of prediction related info
predictions <- cbind(y_test,predictions)
colnames(predictions) <- c("labels","spam","ham","spam_probs",
                                        "ham_probs")
predictions <- data.frame(predictions)
str(predictions)
```

Listing 5: The confusion matrix

```
###################
#confusion matrix
###################
#a11 = actual spam & predicted spam
dim(predictions[predictions$labels == 1 & predictions$spam == 1,])[1]

#a12 = actual spam & predicted ham
dim(predictions[predictions$labels == 1 & predictions$ham == 1,])[1]

#a21 = actual ham & predicted spam
dim(predictions[predictions$labels == 0 & predictions$spam == 1,])[1]

#a22 = actual ham & predicted ham
dim(predictions[predictions$labels == 0 & predictions$ham == 1,])[1]
```

Listing 6: Compute posterior estimates of lambda_1 and lambda_0

```
#calculating posterior values of lambda_0 and lambda_1
  from training data
a <- 1
b <- 1
N <- 4140

#spam emails
X_train_1 <- spam[spam[,1] == 1,][-1]
dim(X_train_1)
sum_X_train_1 <- colSums(X_train_1[,1:ncol(X_train_1)])
lambda_1 <- (sum_X_train_1 + a)/(N + b)

#ham emails
X_train_0 <- spam[spam[,1] == 0,][-1]
dim(X_train_0)
sum_X_train_0 <- colSums(X_train_0[,1:ncol(X_train_0)])
lambda_0 <- (sum_X_train_0 + a)/(N + b)
```

Listing 7: Picking misclassified emails

```
#now picking three misclassified emails
predictions[predictions$labels == 1 & predictions$ham == 1,]
#email - spam classified as ham

predictions[predictions$labels == 0 & predictions$spam == 1,]
#email - ham classified as spam

#pulling features out
x_1 <- X_test[168,]
x_1 <- unlist(x_1)
x_2 <- X_test[155,]
x_2 <- unlist(x_2)
x_3 <- X_test[372,]
x_3 <- unlist(x_3)
```

Listing 8: Plotting misclassified

```
#plotting ----
x_axis_labels <- read.csv("README.csv",
                          skip = 1)
colnames(x_axis_labels) <- "xlab"

par(col = "gray",
    mfcol = c(3,1),
    mar = c(4,4,4,2))
#email 1
plot(lambda_1,
     pch = 16,
     cex = 0.7,
     col = "red",
     xaxt = 'n',
     yaxt = 'n',
     xlab = NA,
     ylab = NA,
     main = "misclassified␣email␣1␣-␣spam␣classified␣as␣ham",
     cex.main = 0.8,
     ylim = c(0,36),
     bty = 'n')
axis(1, at=1:54,
     labels=F,
     cex.axis = 0.7,
     las =2,
     col = "gray",
     tck = -0.008)
axis(2,at = seq(0,30, length.out = 4),
     col = "gray")
text(x = seq(1:54),
     y = -2.4,
     labels = x_axis_labels$xlab,
     srt=45,
     adj = 1,
     xpd = T,
     cex = 0.8,
     col = "black")
lines(lambda_1,
      col = "pink")
```

Listing 9: Plotting misclassified contd.

```
points(lambda_1,
       col = "red",
       pch = 16,
       cex = 0.7)
lines(lambda_0,
      col = "lightblue")
points(lambda_0,
       col = "blue",
       pch = 16,
       cex = 0.7)
lines(x_1,
      col = "gray")
points(x_1,
       pch = 16,
       cex = 0.7,
       col = "black")
legend("topright",
       legend = c("data","lambda_1","lambda_0"),
       pch = 16,
       col = c("black","red","blue"),
       bty = 'n',
       cex = 0.7,
       text.col = "black",
       y.intersp = 0.3)

#email 2
plot(lambda_1,
     pch = 16,
     cex = 0.7,
     col = "red",
     xaxt = 'n',
     yaxt = 'n',
     xlab = NA,
     ylab = NA,
     main = "misclassified␣email␣2␣-␣spam␣classified␣as␣ham",
     cex.main = 0.8,
     ylim = c(0,20),
     bty = 'n')
```

Listing 10: Plotting misclassified contd.

```
axis(1, at=1:54,
     labels=F,
     cex.axis = 0.7,
     las =2,
     col = "gray",
     tck = -0.008)
axis(2,at = seq(0,20, length.out = 5),
     col = "gray")
text(x = seq(1:54),
     y = -2.4,
     labels = x_axis_labels$xlab,
     srt=45,
     adj = 1,
     xpd = T,
     cex = 0.8,
     col = "black")
lines(lambda_1,
      col = "pink")
points(lambda_1,
       col = "red",
       pch = 16,
       cex = 0.7)
lines(lambda_0,
      col = "lightblue")
points(lambda_0,
       col = "blue",
       pch = 16,
       cex = 0.7)
lines(x_2,
      col = "gray")
points(x_2,
       pch = 16,
       cex = 0.7,
       col = "black")
```

Listing 11: Plotting misclassified contd.

```
legend("topright",
       legend = c("data","lambda_1","lambda_0"),
       pch = 16,
       col = c("black","red","blue"),
       bty = 'n',
       cex = 0.7,
       text.col = "black",
       y.intersp = 0.3)
#email 3
plot(lambda_1,
     pch = 16,
     cex = 0.7,
     col = "red",
     xaxt = 'n',
     yaxt = 'n',
     xlab = NA,
     ylab = NA,
     main = "misclassified␣email␣3␣-␣ham␣classified␣as␣spam",
     cex.main = 0.8,
     ylim = c(0,17),
     bty = 'n')
axis(1, at=1:54,
     labels=F,
     cex.axis = 0.7,
     las =2,
     col = "gray",
     tck = -0.008)
axis(2,at = seq(0,15, length.out = 4),
     col = "gray")
text(x = seq(1:54),
     y = -2.4,
     labels = x_axis_labels$xlab,
     srt=45,
     adj = 1,
     xpd = T,
     cex = 0.8,
     col = "black",
     bty = 'n')
lines(lambda_1,
      col = "pink")
```

Listing 12: Plotting misclassified contd.

```
points(lambda_1,
       col = "red",
       pch = 16,
       cex = 0.7)
lines(lambda_0,
      col = "lightblue")
points(lambda_0,
       col = "blue",
       pch = 16,
       cex = 0.7)
lines(x_3,
      col = "gray")
points(x_3,
       pch = 16,
       cex = 0.7,
       col = "black")
legend("topright",
       legend = c("data","lambda_1","lambda_0"),
       pch = 16,
       col = c("black","red","blue"),
       bty = 'n',
       cex = 0.7,
       text.col = "black",
       y.intersp = 0.3)
```

Listing 13: Picking ambiguous emails

```
##############
##ambiguous##
##############
predictions

#a11 = actual spam & predicted spam
predictions[predictions$labels == 1 & predictions$spam == 1,]
#a22 = actual ham & predicted ham
predictions[predictions$labels == 0 & predictions$ham == 1,]

#######
predictions$abs_1 <- abs(predictions$ham_probs - 0.5)

#writing a while loop to pick the ambiguous emails
loc_min <- rep(NA,3)
min_diff <- rep(NA,3)
pred_ambig <- matrix(NA, nrow = 3, ncol = 2)
x_ambig <- matrix(NA, nrow = 3, ncol = 54)

index <- 1
while(index <= 3){
  min_diff[index] <- min(predictions$abs_1)
  loc_min[index] <- which.min(predictions$abs_1)
  predictions$abs_1[loc_min[index]] <- max(predictions$abs_1)
   if((predictions$labels == 1 & predictions$spam == 1)
   || (predictions$labels == 0 & predictions$ham == 1)){
    x_ambig[index,] <- unlist(X_test[loc_min[index],])
    pred_ambig[index,1] <- predictions$spam_probs[loc_min[index]]
    pred_ambig[index,2] <- predictions$ham_probs[loc_min[index]]
    index <- index + 1
   }
}
```

Listing 14: Plotting ambiguous emails

```
###ambiguous email plotting ----

x_1 <- x_ambig[1,]
x_2 <- x_ambig[2,]
x_3 <- x_ambig[3,]

par(col = "gray",
    mfcol = c(3,1),
    mar = c(4,4,4,2))
#email 1
plot(lambda_1,
     pch = 16,
     cex = 0.7,
     col = "red",
     xaxt = 'n',
     yaxt = 'n',
     xlab = NA,
     ylab = NA,
     main = "ambiguous␣email␣1",
     cex.main = 0.8,
     ylim = c(0,15),
     bty = 'n')
axis(1, at=1:54,
     labels=F,
     cex.axis = 0.7,
     las =2,
     col = "gray",
     tck = -0.008)
axis(2,at = seq(0,15, length.out = 4),
     col = "gray")
text(x = seq(1:54),
     y = -2.4,
     labels = x_axis_labels$xlab,
     srt=45,
     adj = 1,
     xpd = T,
     cex = 0.8,
     col = "black")
lines(lambda_1,
      col = "pink")
```

Listing 15: Plotting ambiguous emails contd.

```
points(lambda_1,
       col = "red",
       pch = 16,
       cex = 0.7)
lines(lambda_0,
      col = "lightblue")
points(lambda_0,
       col = "blue",
       pch = 16,
       cex = 0.7)
lines(x_1,
      col = "gray")
points(x_1,
       pch = 16,
       cex = 0.7,
       col = "black")
legend("topright",
       legend = c("data","lambda_1","lambda_0"),
       pch = 16,
       col = c("black","red","blue"),
       bty = 'n',
       cex = 0.7,
       text.col = "black",
       y.intersp = 0.3)

#email 2
plot(lambda_1,
     pch = 16,
     cex = 0.7,
     col = "red",
     xaxt = 'n',
     yaxt = 'n',
     xlab = NA,
     ylab = NA,
     main = "ambiguous␣email␣2",
     cex.main = 0.8,
     ylim = c(0,48),
     bty = 'n')
```

Listing 16: Plotting ambiguous emails contd.

```
axis(1, at=1:54,
     labels=F,
     cex.axis = 0.7,
     las =2,
     col = "gray",
     tck = -0.008)
axis(2,at = seq(0,40, length.out = 5),
     col = "gray")
text(x = seq(1:54),
     y = -2.4,
     labels = x_axis_labels$xlab,
     srt=45,
     adj = 1,
     xpd = T,
     cex = 0.8,
     col = "black")
lines(lambda_1,
      col = "pink")
points(lambda_1,
       col = "red",
       pch = 16,
       cex = 0.7)
lines(lambda_0,
      col = "lightblue")
points(lambda_0,
       col = "blue",
       pch = 16,
       cex = 0.7)
lines(x_2,
      col = "gray")
points(x_2,
       pch = 16,
       cex = 0.7,
       col = "black")
```

Listing 17: Plotting ambiguous emails contd.

```
legend("topright",
       legend = c("data","lambda_1","lambda_0"),
       pch = 16,
       col = c("black","red","blue"),
       bty = 'n',
       cex = 0.7,
       text.col = "black",
       y.intersp = 0.3)
#email 3
plot(lambda_1,
     pch = 16,
     cex = 0.7,
     col = "red",
     xaxt = 'n',
     yaxt = 'n',
     xlab = NA,
     ylab = NA,
     main = "ambiguous␣email␣3",
     cex.main = 0.8,
     ylim = c(0,35),
     bty = 'n')
axis(1, at=1:54,
     labels=F,
     cex.axis = 0.7,
     las =2,
     col = "gray",
     tck = -0.008)
axis(2,at = seq(0,35, length.out = 3),
     col = "gray")
text(x = seq(1:54),
     y = -2.4,
     labels = x_axis_labels$xlab,
     srt=45,
     adj = 1,
     xpd = T,
     cex = 0.8,
     col = "black",
     bty = 'n')
lines(lambda_1,
      col = "pink")
```

Listing 18: Plotting ambiguous emails contd.

```
points(lambda_1,
       col = "red",
       pch = 16,
       cex = 0.7)
lines(lambda_0,
      col = "lightblue")
points(lambda_0,
       col = "blue",
       pch = 16,
       cex = 0.7)
lines(x_3,
      col = "gray")
points(x_3,
       pch = 16,
       cex = 0.7,
       col = "black")
legend("topright",
       legend = c("data","lambda_1","lambda_0"),
       pch = 16,
       col = c("black","red","blue"),
       bty = 'n',
       cex = 0.7,
       text.col = "black",
       y.intersp = 0.3)
```