Projektaufgaben Block 3

Carlo Michaelis, 573479; David Hinrichs, 572347; Lukas Ruff, 572521

10 Januar 2017

1 SPAM vs. HAM: Naive Bayes

In dieser Aufgabe beschäftigen wir uns mit dem Spam vs. Ham Klassifizierungsproblem.

1.1 Einlesen der Daten

```
# Read train/test data and dictionary
colFeatures <- c("docID", "wordID", "wordCount")

trainFeatures <- read.table(file = './data/train-features.txt',
                            col.names = colFeatures)
testFeatures <- read.table(file = './data/test-features.txt',
                           col.names = colFeatures)

YTrain <- read.table(file = './data/train-labels.txt', col.names = "Y")
YTest <- read.table(file = './data/test-labels.txt', col.names = "Y")

# Read dictionary
colDict <- c("wordID", "word")
dict <- read.table(file = './data/dictionary.txt', col.names = colDict)
```

1.2 Erzeugen der Featurematrizen

```
# Build (sparse) feature matrices
XTrain <- sparseMatrix(i = trainFeatures$docID, j = trainFeatures$wordID,
                       x = trainFeatures$wordCount)
XTest <- sparseMatrix(i = testFeatures$docID, j = testFeatures$wordID,
                      x = testFeatures$wordCount)
```

1.3 Erzeuge Wahrscheinlichkeiten für den Naive-Bayes-Classifier

```
fnNBTrain <- function(X, Y) {
  # This function generates the probabilities for the Naive-Bayes-Classifier.
  #
  # Args:
  #   X: Matrix of features
  #   Y: Vector of labels
  #
  # Returns:
  #   A list containing the following elements:
```

```r
  #     $phiSpam:   Probabilities for words occuring in SPAM message
  #     $phiHam:    Probabilities for words occuring in HAM message
  #     $gammaSpam: The relative frequency of SPAM messages
  #     $gammaHam:  The relative frequency of HAM messages

  nWords <- length(Y)
  indSpam <- as.logical(Y)
  indHam <- !(indSpam)
  NBClass <- list()

  # Generate probabilities
  NBClass$phiSpam <- (colSums(X[indSpam, ]) + 1) / (sum(X[indSpam, ]) + nWords)
  NBClass$phiHam <- (colSums(X[indHam, ]) + 1) / (sum(X[indHam, ]) + nWords)

  # Get relative frequencies
  NBClass$gammaSpam <- sum(indSpam)
  NBClass$gammaHam <- sum(indHam)

  return(NBClass)
}

NBClass1 <- fnNBTrain(XTrain, YTrain[[1]])
```

Zähler und Nenner wurden derart angepasst, dass für den Fall, dass keine Trainingsdaten vorliegen, für die bedingten Verteilungen der Wörter in einem Dokument a priori diskrete Gleichverteilungen mit Wahrscheinlichkeiten $\frac{1}{|V|} = \frac{1}{2500}$ angenommen werden.

1.4 Vorhersage auf den Testdaten

```r
fnNBPredict <- function(X, NBClass) {
  # This function predicts one of the two classes (SPAM or HAM) for some test
  # data X using a Naive-Bayes-Classifier trained by fnNBTrain.
  #
  # Args:
  #   X:        Matrix of features
  #   NBClass:  A list returned by fnNBTrain which contains
  #               $phiSpam:   Probabilities for words occuring in SPAM message
  #               $phiHam:    Probabilities for words occuring in HAM message
  #               $gammaSpam: The relative frequency of SPAM messages
  #               $gammaHam:  The relative frequency of HAM messages
  #
  # Returns:
  #   A vector with predictions for each example.

  # Predict labels (using logarithm)
  postSpam <- rowSums(t(t(X) * log(NBClass$phiSpam))) + log(NBClass$gammaSpam)
  postHam <- rowSums(t(t(X) * log(NBClass$phiHam))) + log(NBClass$gammaHam)
  pred <- (postSpam > postHam) * 1

  return(pred)
}
```

```r
# Predict test labels
predTest1 <- fnNBPredict(XTest, NBClass1)

# Number of errors
nErr1 <- sum(YTest[[1]] != predTest1)
```

Insgesamt gibt es lediglich 6 falsche Klassifikationen, was einer Fehlerrate von 2.31% entspricht.

Schauen wir uns an, welche Wörter besonders gute Indikatoren für SPAM oder HAM sind. Dazu betrachten wir jeweils die 25 Wörter mit den größten $\Phi_{i|1}$ bzw. $\Phi_{i|0}$:

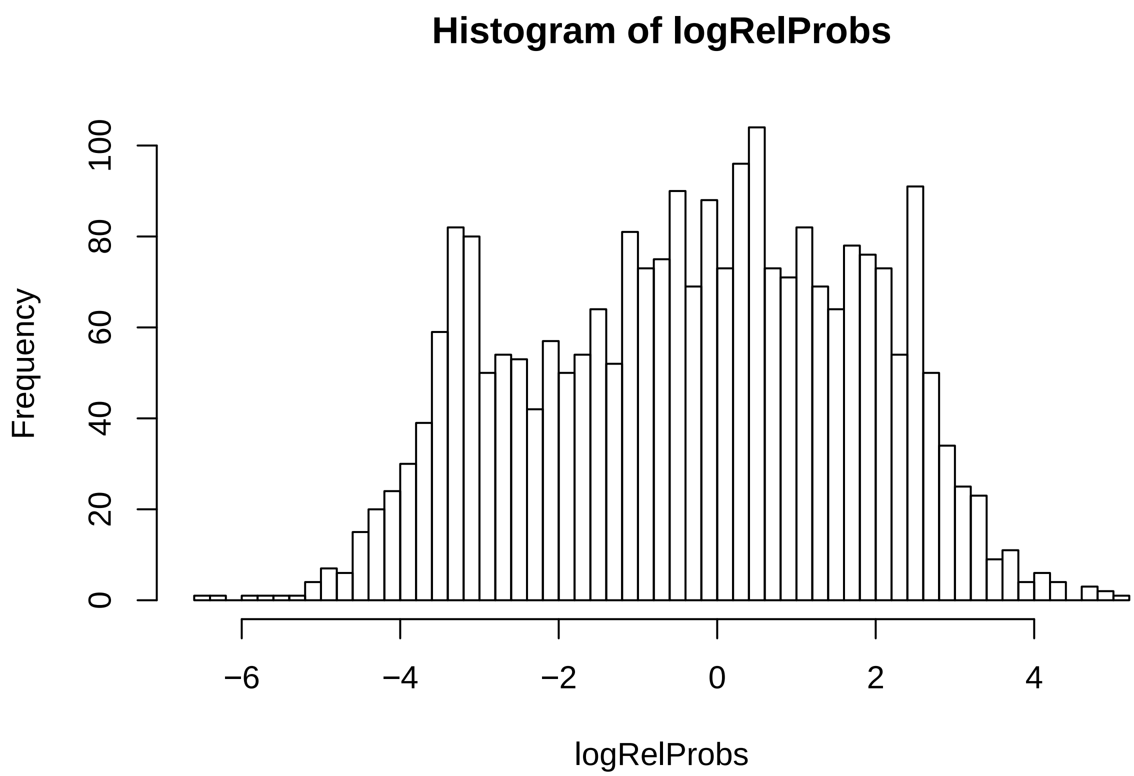
```r
par(mfrow = c(1,2))

# Plot SPAM indicators
wordcloud(dict$word[rank(-NBClass1$phiSpam) <= 50],
          NBClass1$phiSpam[rank(-NBClass1$phiSpam) <= 50],
          scale = c(2.5, 0.2), main = "SPAM indicators")

# Plot HAM indicators
wordcloud(dict$word[rank(-NBClass1$phiHam) <= 50],
          NBClass1$phiHam[rank(-NBClass1$phiHam) <= 50],
          scale = c(2.5, 0.2))
```

Figure 1: Wordclouds for SPAM (left) and HAM (right) indicators.

1.5 Feature Engineering

Für die Klassifizierung von SPAM und HAM bedeutunglose Wörter sollten in SPAM- und HAM-Nachrichten relativ betrachtet ähnlich häufig vorkommen. Um Ähnlichkeit zu quantifizieren, können wir das Verhältnis von $\Phi_{i|1}$ zu $\Phi_{i|0}$ betrachten. Logarithmieren liefert eine bessere Skalierung.

```r
logRelProbs <- log(NBClass1$phiSpam/NBClass1$phiHam)
hist(logRelProbs, breaks = 50)
```




```
# Remove some words
indMeaningful <- !((logRelProbs >= (-1.5)) & (logRelProbs <= 1.5))

# Test classification after removal of meaningless words
NBClass2 <- fnNBTrain(XTrain[, indMeaningful], YTrain[[1]])
predTest2 <- fnNBPredict(XTest[, indMeaningful], NBClass2)

# New classification error
nErr2 <- sum(YTest[[1]] != predTest2)
```

Durch die Entfernung bedeutungsloser Wörtern hat sich die Anzahl der Fehler von 6 auf 3 halbiert.

```
# TODO: Implementation of additional features (e.g. number of words in doc)
# Maybe hints on: https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
```

```
# TODO: Implementation of m-fold cross validation
# Maybe plots for train/test error in dependence of training set size.
# Discussion on model generalization and overfitting.
```

2 SPAM vs. HAM: Lineare Regression & Lasso

2.1 Kleinste-Quadrate-Schätzer und Ridge-Regression-Schätzer

2.1.1 a) Explizite Bestimmung des Ridge-Regression-Schätzers

Wir nehmen ein lineare Modell $Y = X\beta + \epsilon$ an, wobei $\epsilon \sim N(0, \sigma^2 I_n)$. Zunächst betrachten wir den Kleinste-Quadrate-Schätzer $\hat{\beta}$ und formulieren seine explizite Darstellung (siehe Folien):

$$\hat{\beta} = \underset{b \in \mathbb{R}^p}{\operatorname{argmin}} \|Y - Xb\|^2 = (X^T X)^{-1} X^T Y$$

Der Ridge-Regression-Schätzer, ist mit der l^2-Norm wie folgt definiert, wobei wir den zu minimierenden Ausdruck mit Q bezechnen:

$$\hat{\beta}^{RR} = \underset{b \in \mathbb{R}^p}{\operatorname{argmin}} Q = \underset{b \in \mathbb{R}^p}{\operatorname{argmin}} \left(\|Y - Xb\|^2 + \lambda \|b\|_{l^2}^2 \right)$$

Durch Umstellung erhalten wir:

$$\begin{aligned}
Q &= \|Y - Xb\|^2 + \lambda \|b\|_{l^2}^2 \\
&= (Y - Xb)^T (Y - Xb) + \lambda b^T b \\
&= (Y^T - b^T X^T)(Y - Xb) + \lambda b^T b \\
&= Y^T Y - b^T X^T Y - Y^T X b + b^T X^T X b + \lambda b^T b \\
&= Y^T Y - 2b^T X^T Y + b^T X^T X b + \lambda b^T b
\end{aligned}$$

Durch Ableitung erhalten wir den Ridge-Regression-Schätzer:

$$\begin{aligned}
\frac{\partial Q}{\partial b} &= -2X^T Y + 2X^T X b + 2\lambda b \overset{!}{=} 0 \\
\Leftrightarrow\ & X^T Y = (X^T X + \lambda I_p) b \\
\Leftrightarrow\ & \hat{\beta}^{RR} = (X^T X + \lambda I_p)^{-1} X^T Y
\end{aligned}$$

2.1.2 b) Erwartungswert und Varianz

Wir beginnen mit dem Erwartungswert des Kleinste-Quadrate-Schätzers $\hat{\beta}$. Mit der Linearität des Erwartungswertes, sowie $\mathbb{E}[\epsilon] = 0$ aus der Annahme, gilt:

$$\begin{aligned}
\mathbb{E}[\hat{\beta}] &= \mathbb{E}[(X^T X)^{-1} X^T Y] \\
&= (X^T X)^{-1} X^T \mathbb{E}[X\beta + \epsilon] \\
&= (X^T X)^{-1} X^T X \beta + (X^T X)^{-1} X^T \mathbb{E}[\epsilon] = \beta
\end{aligned}$$

Bei der Bestimmung der Varianz von $\hat{\beta}$ nutzen wir erneut, dass der Ewartungswert der Fehler Null ist. Mit dem Verschiebungssatz gilt daher $\operatorname{Var}(\epsilon) = \mathbb{E}[\epsilon \epsilon^T] = \sigma^2 I_p$. Für die Varianz des Schätzers gilt:

$$
\begin{aligned}
\mathrm{Var}(\hat\beta) &= \mathbb{E}[(\hat\beta-\beta)(\hat\beta-\beta)^T] \\
&= \mathbb{E}\Big[\big((X^TX)^{-1}X^TY-\beta\big)\big((X^TX)^{-1}X^TY-\beta\big)^T\Big] \\
&= \mathbb{E}\Big[\big((X^TX)^{-1}X^T(X\beta+\epsilon)-\beta\big)\big((X^TX)^{-1}X^T(X\beta+\epsilon)-\beta\big)^T\Big] \\
&= \mathbb{E}\Big[\big(\beta+(X^TX)^{-1}X^T\epsilon-\beta\big)\big(\beta+(X^TX)^{-1}X^T\epsilon-\beta\big)^T\Big] \\
&= \mathbb{E}\big[(X^TX)^{-1}X^T\epsilon\epsilon^TX(X^TX)^{-1}\big] \\
&= (X^TX)^{-1}X^T\mathbb{E}[\epsilon\epsilon^T]X(X^TX)^{-1} \\
&= (X^TX)^{-1}X^T\mathrm{Var}(\epsilon)X(X^TX)^{-1} \\
&= (X^TX)^{-1}X^T\sigma^2I_nX(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}
\end{aligned}
$$

Der Erwartungswert des Ridge-Resgression-Schäzers $\hat\beta^{RR}$ folgt analog zum obigen Vorgehen:

$$
\begin{aligned}
\mathbb{E}[\hat\beta^{RR}] &= \mathbb{E}[(X^TX+\lambda I_p)^{-1}X^TY] \\
&= (X^TX+\lambda I_p)^{-1}X^T\mathbb{E}[X\beta+\epsilon] \\
&= (X^TX+\lambda I_p)^{-1}X^TX\beta+(X^TX+\lambda I_p)^{-1}X^T\mathbb{E}[\epsilon] \\
&= (X^TX+\lambda I_p)^{-1}X^TX\beta
\end{aligned}
$$

Für $\lambda\to 0$ gilt $\mathbb{E}[\hat\beta]=\mathbb{E}[\hat\beta^{RR}]=\beta$. Der Ridge-Regression-Schätzer entspricht dann dem Kleinste-Quadrate-Schätzer und wird entsprechend erwartungstreu. Für $\lambda\to\infty$ steigt der Bias des Ridge-Regression-Schätzers mit steigendem λ.

Die Varianz wird ebenfalls analog bestimmt:

$$
\begin{aligned}
\mathrm{Var}(\hat\beta^{RR}) &= \mathbb{E}[(\hat\beta^{RR}-\mathbb{E}[\hat\beta^{RR}])(\hat\beta^{RR}-\mathbb{E}[\hat\beta^{RR}])^T] \\
&= \mathbb{E}\Big[\big((X^TX+\lambda I_p)^{-1}X^TY-(X^TX+\lambda I_p)^{-1}X^TX\beta\big)\big((X^TX+\lambda I_p)^{-1}X^TY-(X^TX+\lambda I_p)^{-1}X^TX\beta\big)^T\Big] \\
&= \mathbb{E}\Big[\big((X^TX+\lambda I_p)^{-1}X^T\epsilon\big)\big((X^TX+\lambda I_p)^{-1}X^T\epsilon\big)^T\Big] \\
&= (X^TX+\lambda I_p)^{-1}X^T\mathbb{E}[\epsilon\epsilon^T]X(X^TX+\lambda I_p)^{-1} \\
&= \sigma^2(X^TX+\lambda I_p)^{-1}X^TX(X^TX+\lambda I_p)^{-1}
\end{aligned}
$$

Auch hier gilt für $\lambda\to 0$, dass $\mathrm{Var}[\hat\beta]=\mathrm{Var}[\hat\beta^{RR}]=\sigma^2(X^TX)^{-1}$. Für $\lambda\to\infty$ wird die Varianz des Ridge-Regression-Schätzers mit zunehmendem λ immer kleiner.

2.1.3 c) Mittlerer quadratischer Fehler

Der mittlere quadratische Fehler eines Schätzers $\hat\rho$ für den Parameter ρ kann mittels Bias-Varianz-Zerlegung wie folgt formuliert werden:

$$
\mathbb{E}[\|\hat\rho-\rho\|^2]=\mathbb{E}[(\hat\rho-\rho)^T(\hat\rho-\rho)]=(\mathbb{E}[\hat\rho]-\rho)^2+\mathrm{Var}(\hat\rho)=\mathrm{Bias}(\hat\rho)+\mathrm{Varianz}(\hat\rho)
$$

Für den Kleinste-Quadrate-Schätzer $\hat{\beta} = (X^T X)^{-1} X^T Y$ ergibt sich damit:

$$\mathbb{E}[\|\hat{\beta} - \beta\|^2] = (\beta - \beta)^2 + \mathrm{Var}(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$$

Deutlich wird, dass der Schätzer keinen Bias aufweist, d.h. $\mathrm{Bias}(\hat{\beta}) = 0$.

Für den Ridge-Resgression-Schätzer $\hat{\beta}^{RR} = (X^T X + \lambda I_p)^{-1} X^T Y$ gilt:

$$\mathbb{E}[\|\hat{\beta}^{RR} - \beta\|^2] = \left[(X^T X + \lambda I_p)^{-1} X^T X \beta - \beta\right]^2 + \sigma^2 (X^T X + \lambda I_p)^{-1} X^T X (X^T X + \lambda I_p)^{-1}$$

Der Ridge-Regression-Schätzer besitzt für $\lambda \neq 0$ einen Bias. Gleichzeitig wird die Varianz im Vergleich zum Kleinste-Quadrate-Schätzer kleiner. Damit "erkauft" man sich bei der Ridge-Regression eine geringere Varianz, der Preis dafür ist jedoch eine Verzerrung der Schätzung. Für ein gut gewähltes λ kann dieser "Deal" vorteilhaft sein, z.B. wenn die Varianz des Kleinste-Quadrate-Schätzers so groß ist, dass keine vernünftige Aussage getroffen werden kann.

2.2 Klassifikation mittels Lasso-Schätzer

```r
fnLRPredict <- function(X, Y, XTest) {
  # This function predicts the labels for the test set using Lasso regression
  #
  # Args:
  #   X: Matrix of training features
  #   Y: Vector of training labels
  #   XTest: Vector of test features
  #
  # Returns:
  #   Vector of predicted labels from test features

  # fit spam data via lasso regularization and use binomial function
  fitGlm <- glmnet(X, Y, family = "binomial")

  # do cross validation to find optimal lambda
  cv <- cv.glmnet(X, Y)

  # predict test data
  YPred <- predict(fitGlm, XTest, type="response", s=cv$lambda.min)

  # classify test data
  YPred[YPred >= 0.5] <- 1
  YPred[YPred < 0.5] <- 0

  return(YPred)
}

YPred <- fnLRPredict(XTrain, YTrain[[1]], XTest)

# error rate (compare predicted labels with true labels)
sum(YPred != YTest)/length(YTest[[1]])
```

```
## [1] 0.03461538
```