NAIVE BAYES

E. Fersini

PREDICTING SPAM/HAM

- Naive Bayes (NB) is a **classifying** algorithm which uses data about prior events to estimate the probability of future events.
- We will work on a text dataset to predict if an sms is spam or not

```
> sms = read.csv("spam.csv", stringsAsFactors= F)
```

- The data is simply 5574 records consisting of two fields, type (ham or spam, mostly ham) and the text.

```
> str(sms)
```

DEALING WITH TEXT

- Fix the factor variable

```
> sms$type = factor(sms$type)
```

- Install and load the package tm for dealing with text:

```
> library(tm)
```

- We need to build a **corpus**, which is a collection of documents, from the texts.

```
> sms_corpus = VCorpus(VectorSource(sms$text))
> print(sms_corpus)
> inspect(sms_corpus[1:3])
```

DEALING WITH TEXT

- We need to clean up the data for the aforementioned fluff, we'll use tm_map():

```
> corpus_clean = tm_map(sms_corpus, content_transformer(tolower))
```

```
> corpus_clean = tm_map(corpus_clean, removeNumbers)
```

```
> corpus_clean = tm_map(corpus_clean, removeWords, stopwords())
```

```
> corpus_clean = tm_map(corpus_clean, removePunctuation)
```

```
> corpus_clean = tm_map(corpus_clean, stripWhitespace)
```

DEALING WITH TEXT

- We will now tokenize each message into words to build the key structure for the analysis, a **sparse matrix** comprising:
  - the columns are the words in our corpus
  - the rows correspond to each text message
  - the cells are the number of times each word is seen in each message

- We use **DocumentTermMatrix()**

```
> corpus_clean = tm_map(corpus_clean, PlainTextDocument)
> dtm = DocumentTermMatrix(corpus_clean)
```

- Now we can split in train and test

DEALING WITH TEXT

- Split the raw data in train/test in a trivial way:

```
> sms.train = sms[1:4200, ] # about 75%
> sms.test = sms[4201:5574, ] # the rest
```

- Split the *document-term matrix*

```
> dtm.train = dtm[1:4200, ]
> dtm.test = dtm[4201:5574, ]
```

- *And finally the corpus*

```
> corpus.train = corpus_clean[1:4200]
> corpus.test = corpus_clean[4201:5574]
```

VISUALIZE TEXT DATA

- Visualising text data with wordclouds by installing and loading the library wordcloud

```
> library(wordcloud)
```

- Wordclouds show words in larger fonts if they're more frequent.
  - Let's build one for ham and one for spam to see if we can tell whether or not our NB classifier is likely to be successful.

```
> wordcloud(corpus.train, min.freq=40, random.order = FALSE)
```

```
> wordcloud(corpus.test, min.freq=40, random.order = FALSE)
```

REDUCE THE INPUT SPACE

- DTMs have more than 7000 columns - that’s way too much
  - Let’s eliminate words which appear in less than 5 SMS messages (about 0.1%). We’ll use tm’s findFreqTerms() function

```
> freq_terms = findFreqTerms(dtm.train, 5)
> reduced_dtm.train = DocumentTermMatrix(corpus.train,
list(dictionary=freq_terms))
> reduced_dtm.test = DocumentTermMatrix(corpus.test,
list(dictionary=freq_terms))
```

TRANSFORM TO NOMINAL

- Now, before training a Naïve Bayes classifier, we need to recall that it works on factors, but our DTM only has numerics.
  - Let’s define a function which converts counts to yes/no factor, and apply it to our reduced matrices.

```
>convert_counts = function(x) {
  x = ifelse(x > 0, 1, 0)
  x = factor(x, levels = c(0, 1), labels=c("No", "Yes"))
  return (x)
}
```

```
> reduced_dtm.train = apply(reduced_dtm.train, MARGIN=2,
convert_counts)
> reduced_dtm.test = apply(reduced_dtm.test, MARGIN=2,
convert_counts)
```

TRAINING NAIVE BAYES

- Now we can train our Naive Bayes model

```
> library(e1071)
```

```
> sms_classifier = naiveBayes(reduced_dtm.train, sms.train$type)
```

```
> sms_test.predicted = predict(sms_classifier, reduced_dtm.test)
```

ASSIGNMENT

- Define a NB model with different types of the input space
  - Discretize word counts
  - Remove features with variance equal to zero
  - Remove features with S.D. < -2σ and $>2\sigma$

- Estimate performance of different models and choose the optimal one in terms of accuracy
  - Please take care of the **confusion matrices**!!!
  - What's wrong?