Logistic Regression and Text Analytics

Olena Smotrova

24/02/2018

Contents

Logistic Regression

We consider the problem of predicting a binary response Y using multiple p predictors $X_1, X_2, ..., X_p$. Logistic regression models the probability that Y belongs to a particular category.

$$p(X) = Pr(Y = 1|X) \tag{1}$$

For convinience we are using the generic 0/1 coding for response. In logistic regression, we use the logistic function

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}} \tag{2}$$

We use the maximum likelihood method to estimate $\beta_0, \beta_1, ...\beta_p$. See [1] for more details.

Text Representation with Bag-of-Words Model

In oder to use linear classifier on textual data set, we need to transform our data into numeric data. A popular and simple method is called the bag-of-words model of text. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things: a vocabulary of known words and a measure of the presence of known words. It is called a "bag" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document.

```
## [1] "What a waste of money and time!."
```

```
## [1] "And the sound quality is great."
```

```
##     Terms
## Docs and money time waste what great quality sound the
##    1   1     1    1     1    1     0       0     0   0
##    2   1     0    0     0    0     1       1     1   1
```

As the vocabulary size increases, so does the vector representation of documents. There are simple text transforming techniques that can be used to reduce to the size of the vocabulary:

- Ignoring case
- Ignoring punctuation
- Ignoring stop words, like "a," "of," etc.
- Reducing words to their stem

Text Analysis

Let us consider a text data set consists of 3000 sentences which come from reviews on imdb.com, amazon.com, and yelp.com. Each sentence is labeled according to whether it comes from a positive review or negative review.

```
## 'data.frame':    3000 obs. of  2 variables:
##  $ message: chr  "So there is no way for me to plug it in here in the US unless I go by a converter.
##  $ labels : int  0 1 1 0 1 0 0 1 0 0 ...
```

For text mining we will use `tm` R library. We need to create a collection of documents (`Corpus`). The `tm` package provides the function to do this.

```
corpus = Corpus(VectorSource(sentences$message))
```

Next step is text transformation: eliminate extra space, convert to lower case, remove punctuation and stop words, stem document.

```
# eliminate extra white spaces
corpus <- tm_map(corpus, stripWhitespace)

# Convert to lower case
corpus = tm_map(corpus, content_transformer(tolower))

# Remove punctuation
corpus = tm_map(corpus, removePunctuation)

stop_words = stopwords("english")
# Remove stopwords in our documents
corpus = tm_map(corpus, removeWords, stop_words)

# Stem document
corpus = tm_map(corpus, stemDocument)
```

After text cleaning, we create a term-document matrix.

```
dtm = DocumentTermMatrix(corpus)
inspect(dtm)
```

```
## <<DocumentTermMatrix (documents: 3000, terms: 4045)>>
## Non-/sparse entries: 17880/12117120
## Sparsity           : 100%
## Maximal term length: 32
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   film good great like movi one phone place time work
##   2244    1    0     1    0    0   1     0     0    0    0
##   2376    0    2     0    0    1   0     0     0    0    0
##   2391    0    0     0    0    0   1     0     0    0    0
##   2422    0    0     0    0    1   0     0     0    0    0
##   2429    0    0     0    1    1   0     0     0    0    0
##   2470    0    0     0    0    1   0     0     0    0    0
##   2477    0    0     0    0    0   0     0     0    0    0
##   2621    0    0     0    0    0   0     0     0    0    0
##   2622    0    0     0    0    0   0     0     0    1    0
##   2805    1    0     0    0    0   0     0     0    0    0
```

Size of our 'bag' is

```
dtm$ncol
```

```
## [1] 4045
```

To get the frequency of occurrence of each word in the corpus, we simply sum over all rows to give column sums. First 20 most frequently words are

```
sort(colSums(as.matrix(dtm)), decreasing = TRUE)[1:20]
```

```
##   good  great   movi   film  phone    one   like   work   time  place
##    226    208    208    184    173    147    142    141    136    126
##   food   just servic realli    bad   love    use   well   dont    get
##    125    119    107    103    101     93     93     87     85     81
```

Term document matrix tends to be very big. We could reduce matrix size without loosing important information. To do this we remove sparse terms, i.e., terms occurring only in very few documents.

```
sparse = removeSparseTerms(dtm, 0.995)
inspect(sparse)
```

```
## <<DocumentTermMatrix (documents: 3000, terms: 231)>>
## Non-/sparse entries: 8743/684257
## Sparsity           : 99%
## Maximal term length: 10
## Weighting          : term frequency (tf)
## Sample             :
##       Terms
## Docs   film good great like movi one phone place time work
##   121     0    0     0    0    0   2     1     0    0    0
##   2422    0    0     0    0    1   0     0     0    0    0
##   2429    0    0     0    1    1   0     0     0    0    0
##   2431    2    0     0    1    0   1     0     0    0    0
##   2600    0    0     0    1    0   0     0     0    0    0
##   2804    0    0     0    2    0   0     0     0    0    0
##   2883    0    0     0    0    1   0     0     0    0    0
##   407     0    0     0    0    0   0     1     0    1    0
##   717     0    0     0    0    0   1     0     0    0    1
##   99      0    0     0    0    0   1     0     0    0    0
```

We have been almost done for apllying logistic regerssion to the text data. The last step is to add predicted variable to the term document matrix.

```
# Convert to a data frame
sentencesSparse = as.data.frame(as.matrix(sparse))

# Make all variable names R-friendly
colnames(sentencesSparse) = make.names(colnames(sentencesSparse))

# Add dependent variable
sentencesSparse$labels = sentences$labels
```

Split the data into training and test sets.

```
library(caTools)

split = sample.split(sentencesSparse$labels, SplitRatio = 0.83333)
```

```r
train = subset(sentencesSparse, split==TRUE)
test = subset(sentencesSparse, split==FALSE)
```

Fit a logistic regression model

```r
log.mod = glm(labels~., data = train, family = 'binomial')

pred.log.train = predict(log.mod, type = 'response' )
pred.log.test = predict(log.mod, newdata = test, type = 'response' )

pred.labels.train = ifelse(pred.log.train > 0.5, 1, 0)
pred.labels.test = ifelse(pred.log.test > 0.5, 1, 0)
```

Confusion matrix for the training data and training error rate are

```r
table(pred.labels.train, train$labels)
```

```
##
## pred.labels.train    0    1
##                 0 1058  282
##                 1  192  968
```

```r
mean(train$labels != pred.labels.train)
```

```
## [1] 0.1896
```

Confusion matrix for the test data and test error rate are

```r
table(pred.labels.test, test$labels)
```

```
##
## pred.labels.test   0   1
##                0 186  59
##                1  64 191
```

```r
mean(test$labels != pred.labels.test)
```

```
## [1] 0.246
```

References

[1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning with Applications in R*, **2013**.