Naive Bayes Classifier on Text data

CIS400/600 Fundamentals of Data and Knowledge Mining

February 23, 2017

1. To implement Naive Bayes classifier on text data, the necessary packages are `tm` for text mining and `e1071` for Naive Bayes classifier. To install and load these packages, the R commands are

   ```
   install.packages("tm")
   install.packages("e1071")
   library(tm)
   library(e1071)
   ```

   Necessary documentation about these packages can be found at `https://cran.r-project.org/web/packages/tm/tm.pdf` and `https://cran.r-project.org/web/packages/e1071/e1071.pdf`

2. The dataset provided along with the exercise is SMS SPAM dataset (`sms_spam.csv`). It is textual data which has unprocessed text that is user written. Dataset contains two columns viz. **type** and **text**. Attribute **"type"** indicates type of message i.e. if the message is **SPAM** or **HAM**. Attribute **"text"** is user composed text message. It is the task of student to train Naive Bayes classifier on this text data and classify the same.

3. Text data which is in CSV format is read into dataframe as follows

   ```
   sms_raw <- read.csv("sms_spam.csv", stringsAsFactors =
       FALSE)
   ```

4. In text mining, CORPUS refers to collection of documents. To process text, every individual document (here, message is a document) has to

be put into corpus. Bundle of documents constitute CORPUS. `sms_raw` is a dataframe that has two attributes viz. **"type"** and **"text"**. Attribute **"text"** indicates documents. These documents are to be converted to CORPUS

```
sms_corpus <- VCorpus(VectorSource(sms_raw$text))
```

5. Data cleaning is a priority if user intends to extract meaningful information. Text processing involves:

   (a) Removing Numbers form text

   (b) Removing punctuation

   (c) converting to lower case

   (d) Removing stopwords

   (e) Removing extra white-spaces

   For the above CORPUS `sms_corpus`, text processing involves above mentioned steps. R commands for the same are as follows:

   ```
   sms_corpus_clean <- tm_map(sms_corpus,
       content_transformer(tolower))
   sms_corpus_clean <- tm_map(sms_corpus_clean, removeNumbers)
   sms_corpus_clean <- tm_map(sms_corpus_clean,
       removePunctuation)
   ```

   User can read through the documentation of `tm` to explore additional attributes of `tm_map` method. The difference obtained by cleaning text to can be easily visualized using R commands

   ```
   lapply(sms_corpus[1:3], as.character)
   lapply(sms_corpus_clean[1:3], as.character)
   ```

6. For a classifier to be implemented, it is necessary that the data be placed under necessary attributes. Text data can be converted into similar representation using **Document Term Matrix** or **Term Document Matrix**. In **Document Term Matrix**, documents are represented along rows and terms (words) extracted from these documents

are placed along columns (vice-versa in Term-Document matrix). Document-Term matrix in R using the above generated CORPUS is obtained by R command

```
sms_dtm <- DocumentTermMatrix(sms_corpus_clean)
```

7. For the purpose of classification, data is split to training and test sets. For simplicity 75% data is chosen for training and rest is made into test set. R command for test-train split is

   ```
   # creating training and test datasets
   sms_dtm_train <- sms_dtm[1:4000, ]
   sms_dtm_test <- sms_dtm[4001:5559, ]
   ```

8. Extract class labels i.e. attribute **"type"** from SPAM dataset for both train and test datasets and save them to `sms_train_labels` and `sms_test_labels`.

9. Document-Term matrix when visualized, it can be seen that the matrix is sparse that is there are almost all zeros in the entries with 1s present at very few locations. Hence, it is algorithmically advantageous to work with frequent words, viz. words appearing in most of the documents. To extract frequent terms, say for instance words that occur a minimum of 5 times, `findFreqTerms` function is used

   ```
   freq_terms <- findFreqTerms(sms_dtm_train, 5)
   ```

10. Using these frequent terms (1106 in number) and 5559 documents of CORPUS, we can visualize the data as a matrix with 5559 entries along rows and 1106 entries along columns.

11. Using only frequent terms, the train and test data above viz. `sms_dtm_train` and `sms_dtm_test` are converted into Document-Term matrices

    ```
    # create DTMs with only the frequent terms
    sms_dtm_freq_train <- sms_dtm_train[ , sms_freq_words]
    sms_dtm_freq_test <- sms_dtm_test[ , sms_freq_words]
    ```

12. It is important to convert counts to factor and apply them to train and test sets. This function use is mandatory

```
convert_counts <- function(x)
{
x <- ifelse(x > 0, "Yes", "No")
}

sms_train <- apply(sms_dtm_freq_train, MARGIN = 2,
    convert_counts)

sms_test <- apply(sms_dtm_freq_test, MARGIN = 2,
    convert_counts)
```

13. Use `naiveBayes` classifier from `e1071` package to train the model on train dataset

```
sms_classifier <- naiveBayes(as.matrix(sms_train),
    as.factor(sms_train_labels))
```

14. Perform prediction on test set's document-term matrix as follows

```
sms_test_pred <- predict(sms_classifier,
    as.matrix(sms_test))
```

15. Confusion matrix is printed for the same as follows

```
table("Predicted" = sms_test_pred, "Actual" =
    sms_test_labels)
```