STAT S4240 002, Homework 3

Brian Weinstein (bmw2148)

July 30, 2015

Problem 1: Naive Bayes Text Classification: Data Preparation

See hw03_q1.R for code.

(a) Text pre-processing

```
# load functions from hw03.R
source("hw03.R")

# preprocess text
preprocess.directory("datasets/FederalistPapers/fp_hamilton_train")
preprocess.directory("datasets/FederalistPapers/fp_hamilton_test")
preprocess.directory("datasets/FederalistPapers/fp_madison_train")
preprocess.directory("datasets/FederalistPapers/fp_madison_test")
```

(b) Loading the cleaned data

```
hamilton.train <- read.directory("datasets/FederalistPapers/fp_hamilton_train_clean")
hamilton.test <- read.directory("datasets/FederalistPapers/fp_hamilton_test_clean")
madison.train <- read.directory("datasets/FederalistPapers/fp_madison_train_clean")
madison.test <- read.directory("datasets/FederalistPapers/fp_madison_test_clean")
```

(c) Create a dictionary from all of the documents

```
dictionary <- make.sorted.dictionary.df(c(hamilton.train, hamilton.test,
                                          madison.train, madison.test))
```

(d) Creating document-term-matrices for each of the datasets

```
dtm.hamilton.train <- make.document.term.matrix(infiles=hamilton.train,
                                                dictionary=dictionary)
dtm.hamilton.test <- make.document.term.matrix(infiles=hamilton.test,
                                               dictionary=dictionary)
dtm.madison.train <- make.document.term.matrix(infiles=madison.train,
                                               dictionary=dictionary)
dtm.madison.test <- make.document.term.matrix(infiles=madison.test,
                                              dictionary=dictionary)
```

(e) Compute the log probabilities for the dictionary in each of the document datasets

```
mu=1/nrow(dictionary)

logp.hamilton.train <- make.log.pvec(dtm.hamilton.train, mu)
logp.hamilton.test <- make.log.pvec(dtm.hamilton.test, mu)
logp.madison.train <- make.log.pvec(dtm.madison.train, mu)
logp.madison.test <- make.log.pvec(dtm.madison.test, mu)
```

Problem 2: Naive Bayes Function

We first estimate the log priors based on the log of the proportion of training documents attributed to each author.

$$p(\text{author} = \texttt{author}) = \log\left(\frac{\#\text{ of training documents attributed to }\texttt{author}}{\text{total }\#\text{ of training documents}}\right)$$

Then, using (1) the log probabilities for the dictionary in a Hamilton-authored document and (2) the log probabilities for the dictionary in a Madison-authored document (as computed in **Problem 1**), we can input a new document-term-matrix and classify each document as belonging to one of the authors.

```
naive.bayes <- function(logp.hamilton.train, logp.madison.train,
                        log.prior.hamilton, log.prior.madison, dtm.test){
  # Performs naive bayes classification
  # Inputs:  logp.hamilton.train  :   vector of log probabilities of words
  #                                     occurring in the hamilton training data
  #          logp.madison.train   :   vector of log probabilities of words
  #                                     occurring in the madison training data
  #          log.prior.hamilton   :   the log prior of hamilton documents
  #          log.prior.madison    :   the log prior of madison documents
  #          dtm.test             :   a document-term-matrix to classify
  # Output:  Classification labels for each document in dtm.test

  # calculate the log posterior probabilities
  log.post.hamilton <- log.prior.hamilton + (dtm.test %*% logp.hamilton.train)
  log.post.madison <- log.prior.madison + (dtm.test %*% logp.madison.train)

  # compare the log posterior probabilities and assign to the author
  # with highest probability
  prediction <- data.frame(logPostHam=log.post.hamilton,
                           logPostMad=log.post.madison)
  prediction$pred <- (log.post.hamilton >= log.post.madison)
  prediction$pred <- gsub(TRUE, "Hamilton", prediction$pred)
  prediction$pred <- gsub(FALSE, "Madison", prediction$pred)

  # return a vector of the predictions
  return(prediction$pred)

}
```

Problem 3: Assessing Model Performance

Using the `confusionMatrix` function from the `caret` library:

- **Accuracy:** 63% accurate (% of the test papers that are classified correctly)
- **True Positive Rate:** 100% (Hamilton classified as Hamilton divided by the total amount of testing Hamilton papers)

- **True Negative Rate:** 9% (Madison classified as Madison divided by the total amount of testing Madison papers)
- **False Positive Rate:** 91% (Madison classified as Hamilton divided by the total amount of testing Madison)
- **False Negative Rate:** 0% (Hamilton classified as Madison divided by the total amount of testing Hamilton)

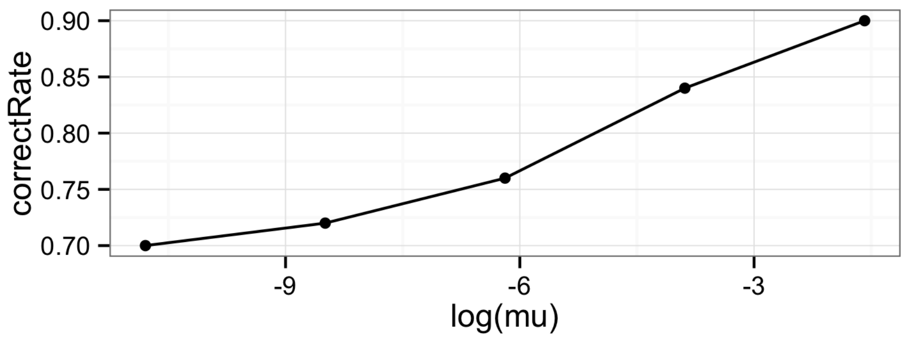
```
> confusionMatrix(data=predictions$pred,
+                 reference=predictions$trueValue,
+                 dnn=c("Prediction", "True Value"),
+                 positive="Hamilton")
Confusion Matrix and Statistics

          True Value
Prediction Hamilton Madison
  Hamilton       16      10
  Madison         0       1

               Accuracy : 0.6296
                 95% CI : (0.4237, 0.806)
    No Information Rate : 0.5926
    P-Value [Acc > NIR] : 0.427258

                  Kappa : 0.106
 Mcnemar's Test P-Value : 0.004427

            Sensitivity : 1.00000
            Specificity : 0.09091
         Pos Pred Value : 0.61538
         Neg Pred Value : 1.00000
             Prevalence : 0.59259
         Detection Rate : 0.59259
   Detection Prevalence : 0.96296
      Balanced Accuracy : 0.54545

       'Positive' Class : Hamilton
```

Problem 4: 5-fold Cross Validation

(a) For each value of $\mu \in \left\{\frac{0.1}{|D|}, \frac{1}{|D|}, \frac{10}{|D|}, \frac{100}{|D|} \frac{1000}{|D|}\right\}$, where $|D| = 4875$ is the size of our dictionary; estimations of the the correct classification rate, the false negative rate, and the false positive rate using 5-fold cross-validation are outlined below. For each of the metrics, the table indicates the value for each of the 25 tests and the graph indicates averages over the 5 tests for each choice of μ.

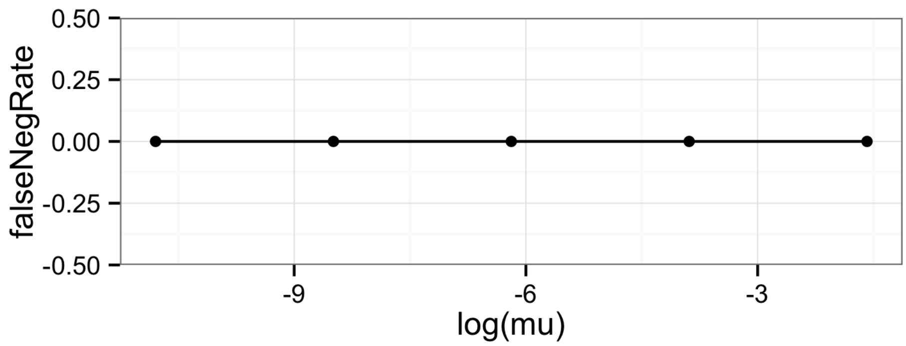
Correct Rate:

	fold1	fold2	fold3	fold4	fold5
mu1	0.7	0.7	0.7	0.7	0.7
mu2	0.8	0.7	0.7	0.7	0.7
mu3	0.8	0.7	0.7	0.8	0.8
mu4	0.9	0.7	0.8	0.8	1.0
mu5	1.0	0.8	0.9	0.8	1.0



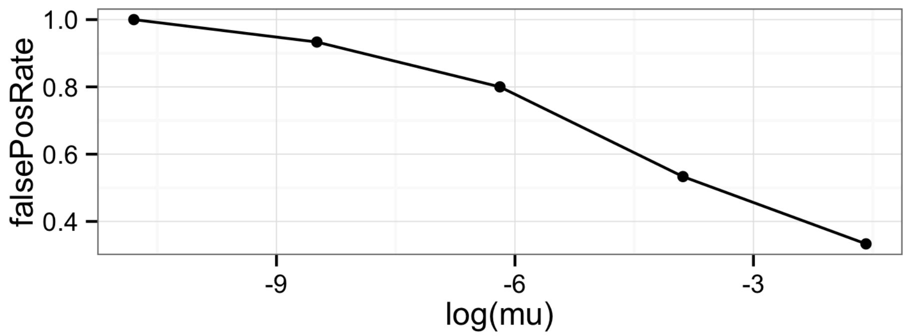
False Negative Rate:

	fold1	fold2	fold3	fold4	fold5
mu1	0	0	0	0	0
mu2	0	0	0	0	0
mu3	0	0	0	0	0
mu4	0	0	0	0	0
mu5	0	0	0	0	0



False Positive Rate:

	fold1	fold2	fold3	fold4	fold5
mu1	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
mu2	0.6666667	1.0000000	1.0000000	1.0000000	1.0000000
mu3	0.6666667	1.0000000	1.0000000	0.6666667	0.6666667
mu4	0.3333333	1.0000000	0.6666667	0.6666667	0.0000000
mu5	0.0000000	0.6666667	0.3333333	0.6666667	0.0000000



(b) Based on the values measured on the training set (using 5-fold cross validation), the best value for μ is `mu5`$= \frac{1000}{|D|} \approx 0.205$. At this value we maximize the accuracy and minimize the false positive rate, with no increase in the false negative rate.

(c) For each value of μ, train on the full training set and test on the full testing set. Summarize the correct classification rate, the false negative rate and the false positive rate in three graphs. Does your answer from (b) still seem the best value? Why or why not?

(d) How close are the rates estimated from cross-validation to the true rates on the testing set (give percentage error)? What could account for dierences? Give one way the differences between the cross-validation rate estimates and the rates on the training sets could be minimized.