Gov 2018: Lab 3 LASSO

Adeline Lo Your name:

Tuesday February 8, 2022

Question 1. Setting up New York Times Annotated Corpus

Question 1.1

Today, we are going to analyze the New York Times Annonated Corpus. From Canvas please download `NYT.RData` and load the file.

This loads a list, `nyt_list`, with the following components:

- train : the document term matrix for the training set
- train_label: an indicator equal to 1 if the story comes from the national desk for each document in the training set
- test: the document term matrix for the test set.
- test_label: an indicator equal to 1 if the story comes from the national desk for each document in the test set

We will work with `train` and `train_label` to build our prediction models. We will use the `test` set to test the fit of our model.

Put these components in individual objects (name each component as a separate object that is easy to understand for you and the reader).

```
# Load library and tidyverse
library(tidyverse)
load("NYT.RData")

# Reading in data, with train and test as tibbles
train <- as_tibble(nyt_list[[1]])
test <- as_tibble(nyt_list[[3]])
train_label <- nyt_list[[2]]
test_label <- nyt_list[[4]]
```

Question 1.2

Print the dimensions of the train and test set. What is the ratio of n to the number of covariates?

Note that the `train` and `test` matrices do not contain a column for the labels. Combine the dtm and labels into two data frames for the train set and for the test set.

```
# Printing dimensions
print("Train dims")
```

```
## [1] "Train dims"
```

```
dim(train)
```

```
## [1]  200 1000
```

```
print("Test dims")
```

```
## [1] "Test dims"
```

```
dim(test)
```

```
## [1]   88 1000
```

```
# Printing ratios
print("Train ratio")
```

```
## [1] "Train ratio"
```

```
nrow(train)/dim(train)[[2]]
```

```
## [1] 0.2
```

```
print("Test ratio")
```

```
## [1] "Test ratio"
```

```
nrow(test)/dim(test)[[2]]
```

```
## [1] 0.088
```

```
# Add labels to dataset
train <- train %>%
  mutate(train_label = train_label)
test <- test %>%
  mutate(test_label = test_label)
```

2. Linear Probability Model

Question 2.1

We are ready to apply a linear probability model to perform classification. Using the `lm` function regress `train_label` against all the words in train. To do this, note that you can include all the variable in a regression using the following syntax: `full_reg <- lm(train_label ~ . , data = train.df)`

The `~.` tells R to include all the variables in the data frame.

Analyze the coefficients from full_reg , what do you notice? Specifically, what happens to the number of coefficients in the model?

```
# Make model
full_reg <- lm(train_label ~ . , data = train)
#summary(full_reg)
```

Many of the coefficient values are listed as NA in the summary(), with a message stating that "801 not defined because of singularities". I believe this means that over 4/5ths of the words were dropped because they were very rare, too rare to be useful in detecting a trend.

Question 2.2

We are now going to make predictions using the training data and the test data and compare their properties.

Using the `predict` function, make predictions for all observations in the training set. Then, classify the documents as national or not using a threshold of 0.5. Assess your classification to the actual data. Create a 2x2 table of the predicted train labels and true train label and note your findings.

```
library(gt)
```

```r
set.seed(12019)

# Make preds
train$preds <- predict(full_reg, train)

# Round preds based on threshold
train <- train %>%
  mutate(preds = ifelse(preds > 0.5, 1, 0))

# Creating table of 1 and 0s
preds_1 <- sum(train$preds)
preds_0 <- 200 - sum(train$preds)
label_1 <- sum(train$train_label)
label_0 <- 200 - sum(train$train_label)

# Formatting the table
tibble(value = c(1,0),
       preds = c(preds_1, preds_0),
       labels = c(label_1, label_0)) %>%
  gt()
```

value	preds	labels
1	50	50
0	150	150

```r
# Calculating prediction accuracy
train %>%
  mutate(correct = ifelse(preds == train_label, 1, 0)) %>%
  summarise(accuracy = sum(correct)/n())
```

```
## # A tibble: 1 x 1
##   accuracy
##      <dbl>
## 1        1
```

The model predicted exactly the same as the true labels (50 1s, 150 0s), with an accuracy of 1. This is because it is predicting the data it was trained on.

Question 2.3

Now, use the model to make a prediction for the *test* data and classify using a 0.5 threshold.

Assess the accuracy of your classification by comparing it to the actual test data. What do you notice? What would happen if you randomly guessed the test labels using a prior on the probability of 1 as the proportion of 1s in the train labels? Remember to `set.seed(12019)`. Compare your findings between the two methods.

```r
set.seed(12019)

# Make preds
test$preds <- predict(full_reg, test)

# Round preds based on threshold
test <- test %>%
  mutate(preds = ifelse(preds > 0.5, 1, 0))
```

```
# Creating table of 1 and 0s
preds_1 <- sum(test$preds)
preds_0 <- 88 - sum(test$preds)
label_1 <- sum(test$test_label)
label_0 <- 88 - sum(test$test_label)

# Formatting table
tibble(value = c(1,0),
       preds = c(preds_1, preds_0),
       labels = c(label_1, label_0)) %>%
  gt()
```

value	preds	labels
1	40	26
0	48	62

```
# calculating prediction accuracy
test %>%
  mutate(correct = ifelse(preds == test_label, 1, 0)) %>%
  summarise(accuracy_model = sum(correct)/n())
```

```
## # A tibble: 1 x 1
##   accuracy_model
##            <dbl>
## 1          0.455
```

```
set.seed(12019)

# Finding the prior based on the train labels
sum(train$train_label)/nrow(train)
```

```
## [1] 0.25
```

```
# Sampling with 0.25 probability of a 1

prior_list <- c(1, 0, 0, 0)

test$prior <- sample(prior_list, replace = TRUE, size = 88)

# Creating table of 1 and 0s
preds_1 <- sum(test$prior)
preds_0 <- 88 - sum(test$prior)
label_1 <- sum(test$test_label)
label_0 <- 88 - sum(test$test_label)

# Formatting table
tibble(value = c(1,0),
       preds = c(preds_1, preds_0),
       labels = c(label_1, label_0)) %>%
  gt()
```

value	preds	labels
1	18	26

0	70	62

```
# Calculating accuracy
test %>%
  mutate(correct = ifelse(prior == test_label, 1, 0)) %>%
  summarise(accuracy_prior = sum(correct)/n())
```

```
## # A tibble: 1 x 1
##   accuracy_prior
##            <dbl>
## 1          0.727
```

Interestingly, using a simple prior was more accurate, 72.7% vs. 45.5%. We can see from the tables that this greater accuracy was because the model predicted 1s far more often than the 'prior' rate of 1s in the training set.

3. Fit LASSO regression

Question 3.1

We are going to use the `glmnet` library to fit the LASSO regression. Load the package.

The syntax for the glmnet model is as follows: `lasso <- glmnet(x = train, y = train_label)`

This defaults to linear regression. To do logistic regression you can fit the same model, but add `lasso_logist <- glmnet(x = train, y = train_label, family = 'binomial')`

Fit a LASSO linear regression.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-3
```

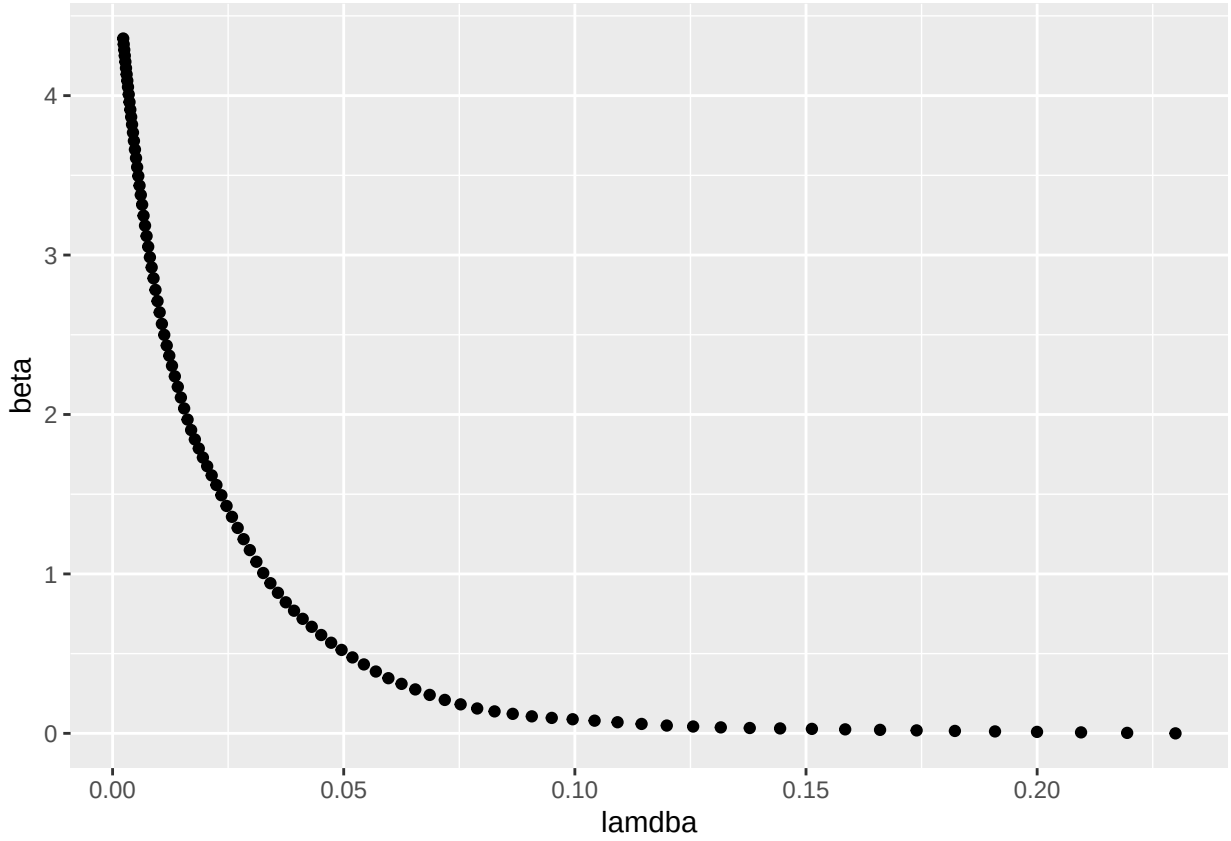
```
# Pulling out train and test set
train_df <- nyt_list[[1]]
test_df <- nyt_list[[3]]

# Defining lasso model
# Must seperate true labels and x's for this func
lasso <- glmnet(x = train_df, y = train_label)
```

Question 3.2

The LASSO function automatically fits the model for several values of λ, and produces β values for all covariates for each value of λ all of which is found in the object `lasso$beta`.

Sum up the absolute values of `lasso$beta` for each column. Plot that against `lasso$lambda`. What generally happens as λ increases?

```
# Plotting lambdas against betas
tibble(beta = colSums(abs(lasso$beta)),
       lamdba = lasso$lambda) %>%
  ggplot(aes(x = lamdba, y = beta)) +
  geom_point()
```



As lambda increases, beta generally decreases.

Question 3.3

There are different methods to selecting lambda, which we set aside for another day. Today, we're going to set a particular value of lambda arbitrarily and then assess its performance. We will set lambda to 0.05.

Formulate predictions for the training set using the following syntax: `lasso_pred <- predict(lasso, newx=train, s = 0.05 )`

- `lasso` is the lasso regression
- `newx` are the values you want to predict
- `s` is the value of lambda.

Classify the observations using a threshold of 0.5. Then assess the accuracy of those predictions by comparing them to the training set labels and create a confusion matrix. Do the same but use a threshold of prior information on the training set – the proportion of 1s. Which threshold is better?

```
# Making predictions on train data
lasso_pred <- predict(lasso, newx=train_df, s = 0.05)

rows <- nrow(lasso_pred)
```

```r
#Making pred and true df
predictions <- tibble(preds = lasso_pred[,1],
       true = train$train_label) %>%
  mutate(preds = ifelse(preds > 0.5, 1, 0))

# Accuracy with a 0.5 threshold
accuracy_thres_50 <- predictions %>%
  mutate(correct = ifelse(true == preds, 1, 0)) %>%
  summarise(accuracy_thres_50 = sum(correct)/200)

accuracy_thres_50
```

```
## # A tibble: 1 x 1
##   accuracy_thres_50
##               <dbl>
## 1             0.905
```

```r
# Accuracy with a 0.25 threshold
accuracy_thres_25 <- tibble(preds = lasso_pred[,1],
       true = train$train_label) %>%
  mutate(preds = ifelse(preds > 0.25, 1, 0)) %>%
  mutate(correct = ifelse(true == preds, 1, 0)) %>%
  summarise(accuracy_thres_25 = sum(correct)/200)

accuracy_thres_25
```

```
## # A tibble: 1 x 1
##   accuracy_thres_25
##               <dbl>
## 1             0.915
```

```r
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
# Confusion matrix
confusionMatrix(table(predictions$preds, predictions$true))
```

```
## Confusion Matrix and Statistics
##
##
##       0   1
##   0 149  18
##   1   1  32
##
##                Accuracy : 0.905
##                  95% CI : (0.8556, 0.9418)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : 2.272e-08
##
```

```
##                   Kappa : 0.7143
##
##  Mcnemar's Test P-Value : 0.0002419
##
##             Sensitivity : 0.9933
##             Specificity : 0.6400
##          Pos Pred Value : 0.8922
##          Neg Pred Value : 0.9697
##              Prevalence : 0.7500
##          Detection Rate : 0.7450
##    Detection Prevalence : 0.8350
##       Balanced Accuracy : 0.8167
##
##        'Positive' Class : 0
##
```

Using the threshold of 0.25 very slightly improves the accuracy. from 90.5 to 91.5.

Question 3.4

Now formulate predictions for the test set, classify the documents as national or not with a threshold using the prior proportion of 1 labels in the training set as well as 0.5, and assess the accuracy of those predictions by comparing them to the test set labels. What do you notice about the quality of the predictions from LASSO relative to the predictions from OLS?

```
# Making test preds
lasso_pred <- predict(lasso, newx=test_df, s = 0.05)

rows <- nrow(lasso_pred)

# Accuracy threshold at 0.5
accuracy_thres_50 <- tibble(preds = lasso_pred,
       true = test$test_label) %>%
  mutate(preds = ifelse(preds > 0.5, 1, 0)) %>%
  mutate(correct = ifelse(true == preds, 1, 0)) %>%
  summarise(accuracy_thres_50 = sum(correct)/88)

accuracy_thres_50
```

```
## # A tibble: 1 x 1
##   accuracy_thres_50
##               <dbl>
## 1             0.830
```

```
# Accuracy threshold at 0.25
accuracy_thres_25 <- tibble(preds = lasso_pred,
       true = test$test_label) %>%
  mutate(preds = ifelse(preds > 0.25, 1, 0)) %>%
  mutate(correct = ifelse(true == preds, 1, 0)) %>%
  summarise(accuracy_thres_25 = sum(correct)/88)

accuracy_thres_25
```

```
## # A tibble: 1 x 1
##   accuracy_thres_25
##               <dbl>
```

```
## 1             0.830
```

The accuracies came out to be the same regardless of the threshold being 0.5 and 0.25. Overall, these LASSO predictions are a vast improvement upon the linear model, which had a sub-50% prediction accuracy on the test data compared to the 80%+ prediction accuracies of LASSO.