Lab Exercise Solutions

07 March 2022

Sentiment analysis using LASSO

Sentiment analysis is a method for measuring the positive or negative valence of language. In this problem, we will use movie review data to create scale of negative to positive sentiment ranging from 0 to 1.

In this exercise, we will do this using a logistic regression model with ℓ_1 penalty (the lasso) trained on a corpus of 25,000 movie reviews from IMDB.

First, lets install and load packages.

```
#install.packages("doMC", repos="http://R-Forge.R-project.org")
#install.packages("glmnet")
#install.packages("quanteda")
#install.packages("readtext")

library(doMC)
library(glmnet)
library(quanteda)
library(readtext)
```

In this first block, I have provided code that downloads the preprocessed data into a matrix of term counts (columns) for each document (rows). This matrix is named `dfm`. Each document is labeled 0 or 1 in the document variable `sentiment`: positive or negative sentiment respectively.

```
options(timeout=max(300, getOption("timeout")))
download.file("https://github.com/lse-my474/pset_data/raw/main/12500_dtm.rds", "12500_dtm.rds")
download.file("https://github.com/lse-my474/pset_data/raw/main/6250_dtm.rds", "6250_dtm.rds")
download.file("https://github.com/lse-my474/pset_data/raw/main/3125_dtm.rds", "3125_dtm.rds")
```

Below is starter code to help you properly train a lasso model using the `.rds` files generated in the previous step. As you work on this problem, it may be helpful when troubleshooting or debugging to reduce `nfolds` to 3 or change N to either 3125 or 6250 to reduce the time it takes you to run code. You can also choose a smaller N if your machine does not have adequate memory to train with the whole corpus.

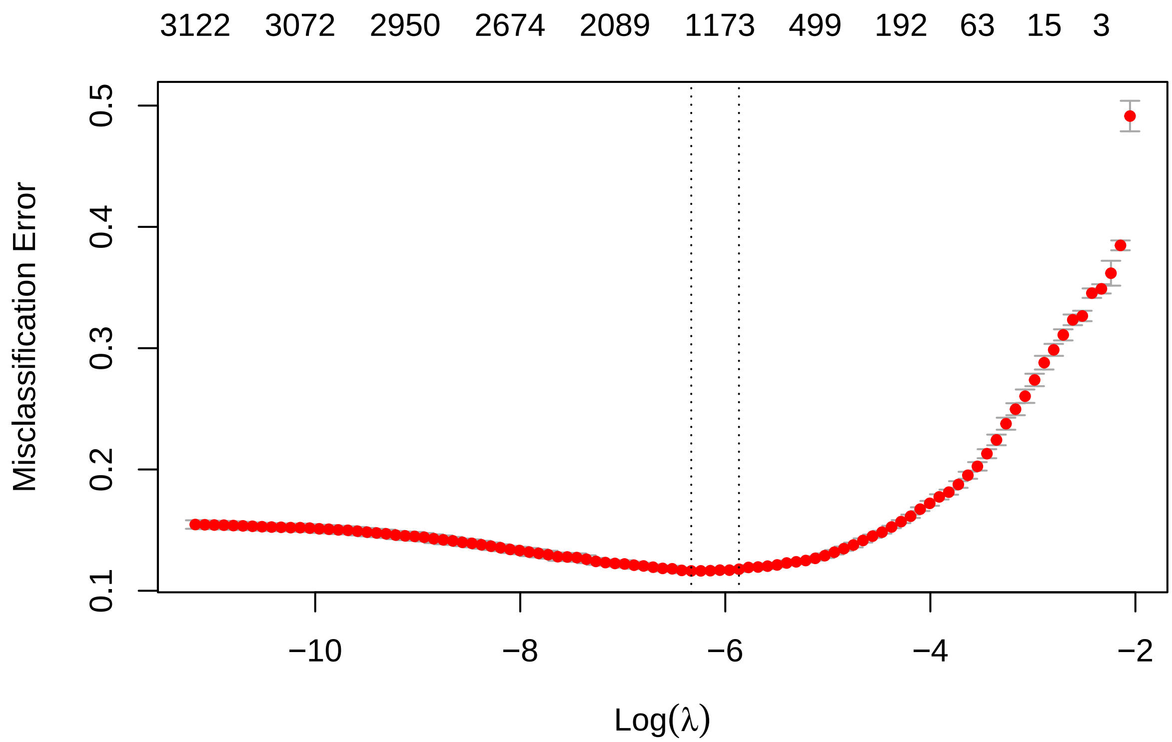
```
# change N to 3125 or 6250 if computation is taking too long
N <- 12500

dfm <- readRDS(paste(N, "_dtm.rds", sep=""))
tr <- 1:(N*2) # indexes for training data

registerDoMC(cores=5) # trains all 5 folds in parallel (at once rather than one by one)
mod <- cv.glmnet(dfm[tr,], dfm$sentiment[tr],
                 nfolds=5, parallel=TRUE, family="binomial", type.measure="class")
```

a. Plot misclassification error for all values of λ chosen by `cv.glmnet`. How many non-zero coefficients are in the model where misclassification error is minimized? How many non-zero coefficients are in the model one standard deviation from where misclassification error is minimized? Which model is sparser?

```
plot(mod)
```




```
print(mod)
```

```
##
## Call:  cv.glmnet(x = dfm[tr, ], y = dfm$sentiment[tr], type.measure = "class",      nfolds = 5, paral
##
## Measure: Misclassification Error
##
##       Lambda Index Measure       SE Nonzero
## min 0.001779    47  0.1164 0.001996    1440
## 1se 0.002832    42  0.1178 0.002194    1006
```

There are 1440 non-zero coefficients in the minimum lambda model and 1006 in the 1 s.e. model. The 1 s.e. model is sparser because it has fewer non-zero coefficients due to having a higher value of lambda.

b. According to the estimate of the test error obtained by cross-validation, what is the optimal λ stored in your `cv.glmnet()` output? What is the CV error for this value of λ? *Hint: The vector of λ values will need to be subsetted by the index of the minimum CV error.*

```
lam_min <- which(mod$lambda == mod$lambda.min)
lam_min
```

```
## [1] 47
```

```
cv_min <- mod$cvm[lam_min]
cv_min
```

```
## [1] 0.11636
```

c. What is the test error for the λ that minimizes CV error? What is the test error for the 1 S.E. λ? How well did CV error estimate test error?

```r
pred_min <- predict(mod, dfm[-tr,], s="lambda.min", type="class")
mean(pred_min != dfm$sentiment[-tr])
```

```
## [1] 0.11552
```

```r
lam_1se <- which(mod$lambda == mod$lambda.1se)
pred_1se <- predict(mod, dfm[-tr,], s="lambda.min", type="class")
mean(pred_1se != dfm$sentiment[-tr])
```

```
## [1] 0.11552
```

C.V. error estimated test error very closely.

d. Using the model you have identified with the minimum CV error, identify the 10 largest and the 10 smallest coefficient estimates and the features associated with them. Do they make sense? Do any terms look out of place or strange? In 3-5 sentences, explain your observations. *Hint: Use* ***`order()`****,* ***`head()`****, and* ***`tail()`****. The argument* ***`n=10`*** *in the* ***`head()`****, and* ***`tail()`*** *functions will return the first and last 10 elements respectively.*

```r
beta <- mod$glmnet.fit$beta[,lam_min]
ind <- order(beta)

head(beta[ind], n=10)
```

```
##          waste disappointment        unfunny    forgettable         poorly
##      -1.498757      -1.362972      -1.323662      -1.288455      -1.277560
##          worst      obnoxious      pointless          awful          trite
##      -1.272199      -1.267985      -1.090637      -1.075391      -1.057022
```

```r
tail(beta[ind], n=10)
```

```
##        hooked extraordinary      captures      funniest     excellent
##     0.7677677     0.7939172     0.8558665     0.8578376     0.8597383
##      troubled             8   wonderfully             7    refreshing
##     0.9598468     1.0038663     1.1200738     1.1958341     1.5274383
```

The largest magnitude positive and negative coefficients overall make a good deal of sense. I see that the number eight is an important feature, which might be due to a rating out of ten by the reviewer. The word "troubled" stands out as well. This could be related to the importance of conflict in good story-telling. Overall, the weights for each of these terms provide a sanity check that our model is capturing sentiment.