Text as Data: Homework 4

Jeff Ziegler

August 22, 2017

In this problem set we're going to make a slight detour from text as data to predict student's drinking habits. While this might seem like a big departure, we're going to learn more about some methods that are fundamental to classification problems. Further, we're going to see that the methods for learning for text as data problems are more generally useful. Trust me, this is going to help you analyze text better (but always keep in mind what would be similar...) The data come from a public health study of Portugese students. You can read more about the variables (and their interpretation) here. We're going to model the sum of weekday and weekend drinking activity. The data are stored in `StudentDrinking.RData` on canvas. The dependent variable is, `alcohol`, which is a measure of alcohol consumption. The bigger `alcohol` is, the more students drink. The covariates are stored in `X`.

1 Comparing Coefficients from OLS, LASSO, Ridge, and Elastic Net

We first want to explore the behavior of OLS, LASSO, and Ridge applied to the data.

i) Fit a linear regression of alcohol on the covariates in the included data

ii) Using `cv.glmnet` fit a LASSO regression of alcohol on the covariates

iii) Using `cv.glmnet` fit a Ridge regression of alcohol on the covariates

iv) Using `cv.glmnet` fit an elastic-net regression of alcohol on the covariates, with $\alpha = 0.5$. Explain what $\alpha = 0.5$ implies about the model you're fitting.

```
# Problem 1

# load libraries and .csv files
library(glmnet); library(data.table); library(randomForest)
load("~/Documents/Git/WUSTL_textAnalysis/StudentDrinking.RData")

# i) fit a linear regression of alcohol on the covariates in the included data
olsModel <- lm(alcohol~X)
# ii) fit a lasso regression of alcohol on the covariates
# alpha = 1 for lasso, alpha=0 for ridge
```

```r
lassoModel <- cv.glmnet(x = X, y = alcohol, alpha = 1)
# iii  fit a ridge regression of alcohol on the covariates
ridgeModel <- cv.glmnet(x = X, y = alcohol, alpha = 0)
# iv) fit an elastic-net regression of alcohol on the covariates (alpha = 0.5)
elasticModel <- cv.glmnet(x = X, y = alcohol, alpha = 0.5)
# alpha = 0.5 implies a "balance" between the penalty occurred
# under ridge (lamba*sum(beta_j)) and lasso (lamba*sum(abs(beta_j)))
```

v) Using your models from (i-iv) let's examine the behavior of the coefficient on `male` as λ increases

a) Suppose `glmnet.obj` contains the results from applying `cv.glmnet`. To obtain the coefficient values for the sequence of λ values tested in `cv.glmnet`, we use the coefficient function `coef(glmnet.obj, s = glmnet.obj$lambda)`. Use this function to obtain a matrix of coefficients for the models used in (ii-iv).

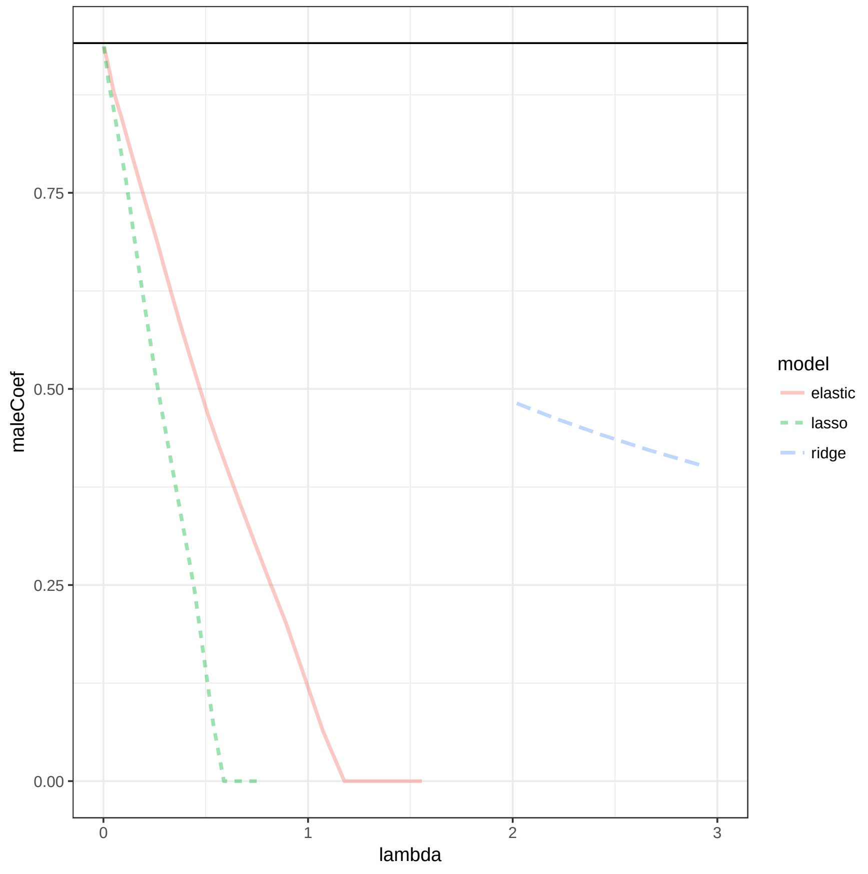
b) Using the matrix for each method, plot the coefficient on `male` against the value of λ from the models in ii-iv. Include the coefficient from OLS as a flat line. What do you notice as λ increases?

```r
# v) a) task: obtain a matrix of coefficients for the models used in (ii-iv).
lassoLambda <- data.table("maleCoef" = t(as.matrix(coef(lassoModel, s =
    lassoModel$lambda)))[,3],
                          "lambda"=lassoModel$lambda, "model"="lasso")
ridgeLambda <- data.table("maleCoef" = t(as.matrix(coef(ridgeModel, s =
    ridgeModel$lambda)))[1:65,3],
                          "lambda"=ridgeModel$lambda[1:65], "model"="ridge")
elasticLambda <- data.table("maleCoef" = t(as.matrix(coef(elasticModel, s =
    elasticModel$lambda)))[1:65,3],
                            "lambda"=elasticModel$lambda[1:65], "model"="
    elastic")

# b) plot the coefficient on male against the value of lambda from the models
    in ii-iv
# and include the coefficient from OLS as a flat line
# merge data
plotDataframe <- merge(lassoLambda, ridgeLambda, by=c("maleCoef", "lambda", "
    model"), all=T)
plotDataframe <- cbind(olsModel$coefficients[3], merge(elasticLambda,
    plotDataframe, by=c("maleCoef", "lambda", "model"), all=T))

pdf("~/Documents/Git/WUSTL_textAnalysis/HW4lambda.pdf")
# we can see from the plot that as lambda --> 0, we get closer to the ols coef
# which is displayed as the solid, horizontal black line at 0.94
ggplot(plotDataframe, aes(x = lambda, y = maleCoef, col = model, linetype =
    model)) +
  geom_line(lwd = 1, alpha=.4) + geom_hline(yintercept=plotDataframe$V1) +
  theme_bw() + scale_x_continuous(limits = c(0, 3))
dev.off()
```

Figure 1: The estimated coefficient of `male` as λ varies.



2 Cross-Validation, Super Learning and Ensembles

We're going to assess the performance of five models, an unweighted average, and a super-learning average of the methods.

i) First, set the first 20 rows to the side for use as the validation set.

ii) We'll first estimate the (unconstrained) super learner weights.

   a) On the training data (all but the first 20 rows) perform ten fold cross validation, including (1) linear regression, (2) LASSO, (3) Ridge, (4) Elastic-Net, and (5) Random Forest. Obtain 5 predictions for each observation in the training set, one from each observation

   b) Regress the dependent variable on the out of sample prediction, (without including an intercept).

   c) Store those weights as $\boldsymbol{w}$

```r
# Problem 2

set.seed(4)
# i) create the validation set
alcoholValid <- alcohol[c(1:20)]; xValid <- X[c(1:20),]
alcoholTraining <- alcohol[-c(1:20)]; xTraining <- X[-c(1:20),]
# w/ the training data (all but the first 20 rows) perform 10 fold CV
    including:
# create 10 folds
folds <- sample(1:10, length(alcoholTraining), replace=T)
# create vectors to fill with predicted values
olsPredictions <- c(); lassoPredictions <- c(); ridgePredictions <- c();
elasticPredictions <- c(); randomPredictions <- c()
# for each fold
for(i in 1:10){
  # find which observations are included in fold and which aren't
  trainingData <- which(folds!=i)
  testData <- which(folds==i)
  # (1) linear regression
  olsTrain <- lm(alcoholTraining[trainingData] ~ xTraining[trainingData, ])
  # get predicted value
  olsPredictions[testData] <- predict(olsTrain, newdata=as.data.frame(
    xTraining[testData, ]))
  # (2) lasso
  # alpha = 1 for lasso, alpha=0 for ridge
  lassoTrain <- cv.glmnet(y = alcoholTraining[trainingData], x = xTraining[
    trainingData, ], alpha = 1)
  # get predicted value
  lassoPredictions[testData] <- predict(lassoTrain, newx= xTraining[testData,
    ], s = lassoTrain$lambda.min)
  # (3) ridge
  ridgeTrain <- cv.glmnet(y = alcoholTraining[trainingData], x = xTraining[
    trainingData, ], alpha = 0)
  # get predicted value
  ridgePredictions[testData] <- predict(ridgeTrain, newx= xTraining[testData,
    ], s = ridgeTrain$lambda.min,type = "class")
  # (4) elatist-net
  elasticTrain <- cv.glmnet(y = alcoholTraining[trainingData], x = xTraining[
    trainingData, ], alpha = 0.5)
  # get predicted value
  elasticPredictions[testData] <- predict(elasticTrain, newx= xTraining[
    testData, ], s = elasticTrain$lambda.min,type = "class")
  # (5) random forest
  randomTrain <- randomForest(y = alcoholTraining[trainingData], x = xTraining
    [trainingData, ])
  # get predicted value
  randomPredictions[testData] <- predict(randomTrain, newdata=as.data.frame(
    xTraining[testData, ]))
}
#
# obtain weights
```

```
modelWeights <- lm(alcoholTraining ~ cbind(olsPredictions, lassoPredictions,
    ridgePredictions,
                                              elasticPredictions,
    randomPredictions) -1)$coefficients
```

Table 1: Model weights from regression of alcohol on the out of sample predictions for each model.

	Estimated coefficient ($\boldsymbol{w}$)	Std. Error
OLS Predictions	0.0433	0.0631
Lasso Predictions	0.1050	3.7391
Ridge Predictions	-0.4370	0.7474
Elastic Predictions	0.9345	4.2089
Random Predictions	0.3639	0.1797

iii) Now, fit all 5 models from (ii)-(a) to the entire training data set and predict the drinking level from the validation set (the data put off to the side).

```
# iii) fit all 5 models from (ii)-(a) to the entire training data set
# and predict the drinking level from the validation set
# (1) linear regression
olsTrain <- lm(alcoholTraining ~ ., data=as.data.frame(cbind(alcoholTraining,
    xTraining)))
# get predicted value
olsPredictions <- predict(olsTrain, newdata=as.data.frame(xValid))
# (2) lasso
# alpha = 1 for lasso, alpha=0 for ridge
lassoTrain <- cv.glmnet(y = alcoholTraining, x = xTraining, alpha = 1)
# get predicted value
lassoPredictions <- predict(lassoTrain, newx= xValid, s = lassoTrain$lambda.
    min)
# (3) ridge
ridgeTrain <- cv.glmnet(y = alcoholTraining, x = xTraining, alpha = 0)
# get predicted value
ridgePredictions <- predict(ridgeTrain, newx= xValid, s = ridgeTrain$lambda.
    min,type = "class")
# (4) elatist-net
elasticTrain <- cv.glmnet(y = alcoholTraining, x = xTraining, alpha = 0.5)
# get predicted value
elasticPredictions <- predict(elasticTrain, newx= xValid, s = elasticTrain$
    lambda.min,type = "class")
# (5) random forest
randomTrain <- randomForest(y = alcoholTraining, x = xTraining)
# get predicted value
randomPredictions <- predict(randomTrain, newdata=as.data.frame(xValid))

# predict the drinking level from the validation set (the data put off to the
    side).
```

```r
predictedAlcohol <- round(cbind(alcoholValid, olsPredictions, lassoPredictions
    , ridgePredictions, elasticPredictions, randomPredictions))
# check how many observations each model predicted correctly
```

Table 2: Predicted alcohol category by model (rounded to nearest integer).

Alcohol (validation set)	OLS preds	Lasso preds	Ridge preds	Elastic-Net preds	Random Forest preds
2	4	4	4	4	4
2	3	3	3	3	3
5	2	3	3	3	3
2	2	2	2	2	3
3	2	3	3	3	3
3	3	3	3	3	4
2	5	5	4	5	4
2	4	4	4	4	4
2	3	3	3	3	3
2	2	2	3	2	3
3	3	3	3	3	3
2	2	2	2	2	3
4	4	4	4	4	3
3	4	4	4	4	3
2	3	3	3	3	3
3	4	4	4	4	5
3	3	3	3	3	3
2	2	2	2	2	3
6	6	5	6	5	7
4	4	4	4	4	4

iv) Obtain two ensemble predictions.

a) Take the unweighted average of the predictions from the methods

b) Take the weighted average, using the weights $\boldsymbol{w}$.

```r
# iv) Obtain two ensemble predictions:
# a) unweighted average of the predictions from the methods
unweightedModelAverage <- round((rowSums(predictedAlcohol[,-1]))/dim(
    predictedAlcohol[,-1])[2])
# b) weighted average, using the weights
weightedModelAverage <- round(rowSums(sweep(predictedAlcohol[,-1], MARGIN=2,
                                            as.numeric(modelWeights),'*')))
```

v) You should have 7 predictions. Store those in a matrix and report the correlation between the predictions

vi) Using the average absolute difference as a loss function assess the performance of each method. Which method performs best? Which performs the worst?

The average absolute difference for method k is defined as

$$L(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}_k) \quad = \quad \sum_{i=1}^{N_{\text{validation}}} \frac{|Y_i - \widehat{Y}_{ik}|}{N_{\text{validation}}}$$

where $N_{\text{validation}}$ refers to the number of observations in the validation set $\widehat{\boldsymbol{Y}}_k$ refers to the predictions from the k^{th} method,

```r
# v) store predictions in matrix
allModelPredictions <- cbind(cbind(olsPredictions, lassoPredictions, ridgePredictions,
                elasticPredictions, randomPredictions, unweightedModelAverage, weightedModelAverage))

# vi) for each method, determine average absolute difference
# sum(abs(alcoholValid_i - modelPrediction_ik)/nrow(alcoholValid))
for(i in 1:dim(allModelPredictions)[2]){
  print(sum(abs(alcoholValid[1:20] - allModelPredictions[1:20,i])/length(alcoholValid)))
}
```

Table 3: Average absolute difference for each method.

Model	$L(\boldsymbol{Y}, \widehat{\boldsymbol{Y}}_k)$
OLS	0.838
Lasso	0.847
Ridge	0.836
Elastic-Net	0.845
Random Forest	0.999
Unweighted model average	0.7
Weighted model average	0.7