Homework 2, STATS 315A

Stanford University, Winter 2019

Joe Higgins

Question 2

This question will have you write an R function to efficiently perform forward stepwise linear regression. It requires that you understand the material on pages 13–15 of the chapter 3 lecture notes. The setup is linear regression with an NÖp matrix X and a response vector y. You always include an intercept in your models, which is not included in X.

(a) Let $\tilde{X}$ be the matrix X with each of the columns mean centered. What is the fitted intercept in the regression of y on $\tilde{X}$? How do the coefficients of X in the regression of y on X compare with the coefficients of $\tilde{X}$?

```
#stuff goes here
```

Question 7

7. Obtain the zipcode train and test data from the ESL website.

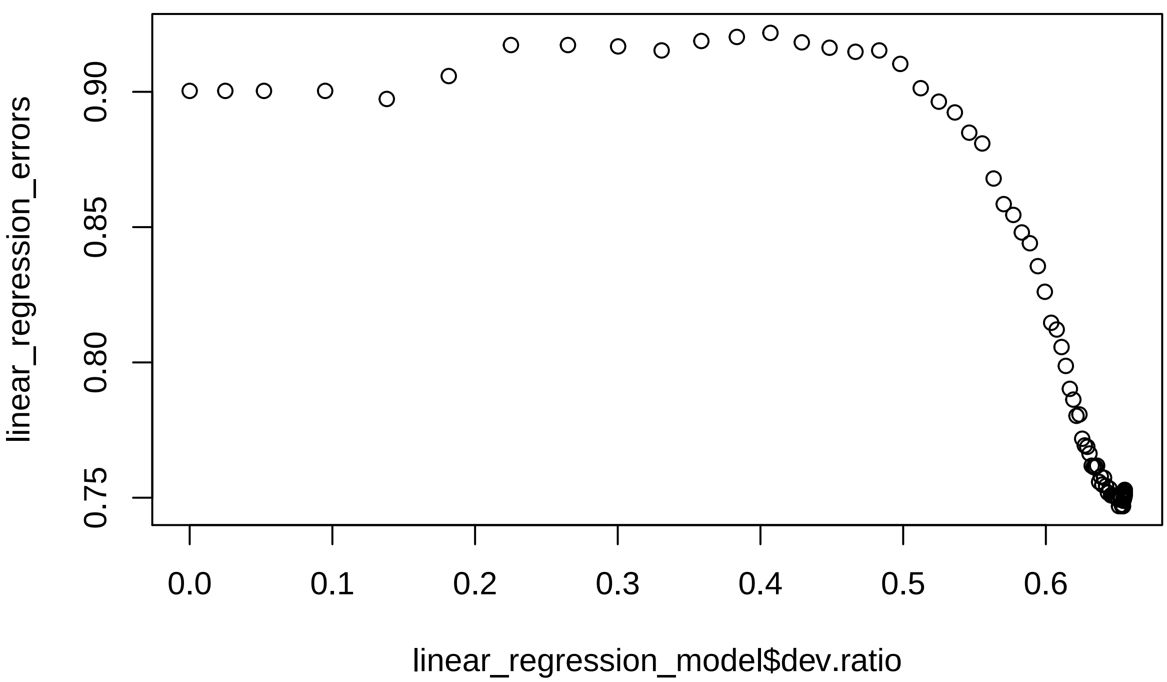
i. Compare the test performance of a) linear regression b) linear discriminant analysis and c) multiclass linear logistic regression.

ii. For a) and c), use the package glmnet (available in R, matlab and python) to run elastic-net regularized versions of each (use $\alpha = 0.3$). For these two, plot the test error as a functions of the training R2 for a) and D2 for c) (% training deviance explained).

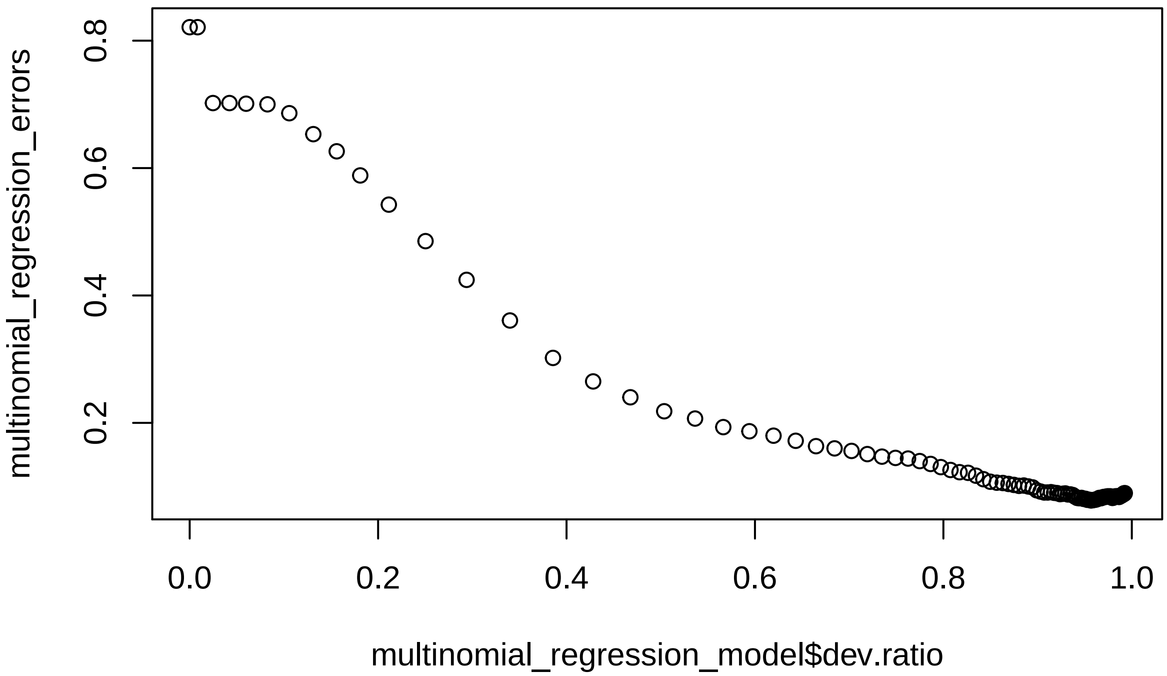
iii. In ii., what is the optimization problem being solved?

```
rm(list = ls())

read_data_as_matrix <- function(file_string){
  table <- read.table(file_string)
  output <- matrix(, nrow=dim(table)[1], ncol=dim(table)[2])
  for(i in 1:ncol(table)){
    output[,i] <- table[,i]
  }
  return(output)
}

get_accuracy <- function(y_hat, y){
  correct <- y_hat == y
  pct_correct <- sum(correct)/length(correct)
  return(pct_correct)
}

train <- read_data_as_matrix("zip.train")
y_train <- train[,1]
x_train <- train[,2:dim(train)[2]]
test <- read_data_as_matrix("zip.test")
y_test <- test[,1]
x_test <- test[,2:dim(test)[2]]
```

```
linear_regression_model <- glmnet(x_train, y_train, family=c("gaussian"), alpha = 0.3)
lda_model <- lda(y_train ~ ., data=data.frame(x_train), na.action="na.omit", CV=FALSE)
multinomial_regression_model <- glmnet(x_train, y_train, family=c("multinomial"), alpha = 0.3)

linear_regression_output <- predict(linear_regression_model, x_test, type=c("response"))
linear_regression_prdct <- round(linear_regression_output)
linear_regression_prdct[linear_regression_prdct == -1] <- 0
multinomial_regression_output <- predict(multinomial_regression_model, x_test, type=c("response"))
multinomial_regression_prdct <- apply(
  multinomial_regression_output, 3, function (x) apply(
    x, 1, function(y) which.max(y) - 1
  )
)
lda_output <- predict(lda_model, data.frame(x_test))
lda_prdct <- lda_output$class


linear_regression_errors <- apply(linear_regression_prdct, 2, function(x) 1 - get_accuracy(x, y_test))
multinomial_regression_errors <- apply(multinomial_regression_prdct, 2, function(x) 1 - get_accuracy(x,
lda_accuracy <- get_accuracy(lda_prdct, y_test)

#is dev ratio the R^2 for linear regression?
#what is the optimization being done...
plot(linear_regression_model$dev.ratio, linear_regression_errors)
```




```
plot(multinomial_regression_model$dev.ratio, multinomial_regression_errors)
```



multinomial_regression_errors
0.2
0.4
0.6
0.8
0.0
0.2
0.4
0.6
0.8
1.0
multinomial_regression_model$dev.ratio