Assignment I: MIS 64037

Eyob Tadele

March 13, 2022

PART A:

QA1: Purpose of regularization in training predictive models:

The main purpose of regularization is to optimize the performance on a training set in order to avoid underfitting and also penalize the model when it becomes too complex, leading to overfitting. In the case of overfitting, it is to prevent the likelihood that our predictive model performs too well on the training data, but not on an unseen/test data. In other words, the model is overfitting as it is also learning the noise from the training set. To prevent this, regularization techniques actively impede the model's ability to fit perfectly to the training data. It tends to make the model simpler, hence the name 'regular', where it becomes less specific to the training set and better able to generalize by more closely approximating the latent manifold of the data.

In short, regularization is used to calibrate predictive models in order to minimize the adjusted loss function and prevent overfitting and underfitting. Basically, starting with a slightly worse fit, regularization provides better long term predictions.

QA2: Role of loss function in predictive model and some of the common loss functions used for regression and classification models:

The main role of a loss function in predictive models is to evaluate how well it is doing in terms of being able to predict the expected outcome. Depending on the magnitude of deviation from the actual result, the loss function would be as big or small. With the help of an optimization function, the loss function will gradually learn to reduce the error in prediction.

Loss functions for Regression: since we deal with predicting a continuous value in regression, some of the common loss functions that could be applied are Mean Square Error(MSE) or Mean Absolute Error(MAE).

Loss functions for Classification: as we are trying to predict from a set of finite categorical values, Binary Cross-Entropy, or Hinge Loss functions can be used.

QA3: Consider the following scenario. You are building a classification model with many hyper parameters on a relatively small dataset. You will see that the training error is extremely small. Can you fully trust this model? Discuss the reason.

The simple answer is no. We can not fully trust this model. First, the mere fact that we have a small dataset increases the likelihood of overfitting on the training set, thereby seeing a very small training error. On top of that, we are building it on a model with many hyper-parameters (i.e., indicating the model is complex) and complex models with many parameters are prone to overfitting. The best way to handle it is by choosing a simpler model limiting its ability to see non-existent patterns and relationships in the small dataset. The concept of regularization can come handy in this case by making the model simpler.

QA4: Role of the lambda parameter in regularized linear models such as Lasso or Ridge regression models:

In its broad definition, lambda represents the penalty term in the regularized linear model-it determines how severe that penalty is. Meaning, the higher the penalty, the more it reduces the magnitude of coefficients (i.e., it shrinks the parameters). As a result, it is used to prevent multi-collinearity, and reduce model complexity by coefficient shrinkage. In both Lasso and Ridge regression models, the tuning parameter 'Lambda' is chosen through cross-validation. In Ridge, the goal is to make the fit small by making the residual sum or squares small, plus adding a shrinkage penalty(i.e., Lambda * sum of squares of the coefficients). This ensures that the coefficients that get too large are penalized. In the case of Lasso, when lambda is small, the result is essentially the least squares estimates. As lambda increases, shrinkage occurs so that variables that are at zero can be thrown away. This is an added advantage of Lasso, as it combines both shrinkage and variable selection. It is important to note here that the key difference between Ridge and Lasso Regression is that the former can only shrink the slope asymptotically close to zero, while the later can shrink the slope all the way to zero.

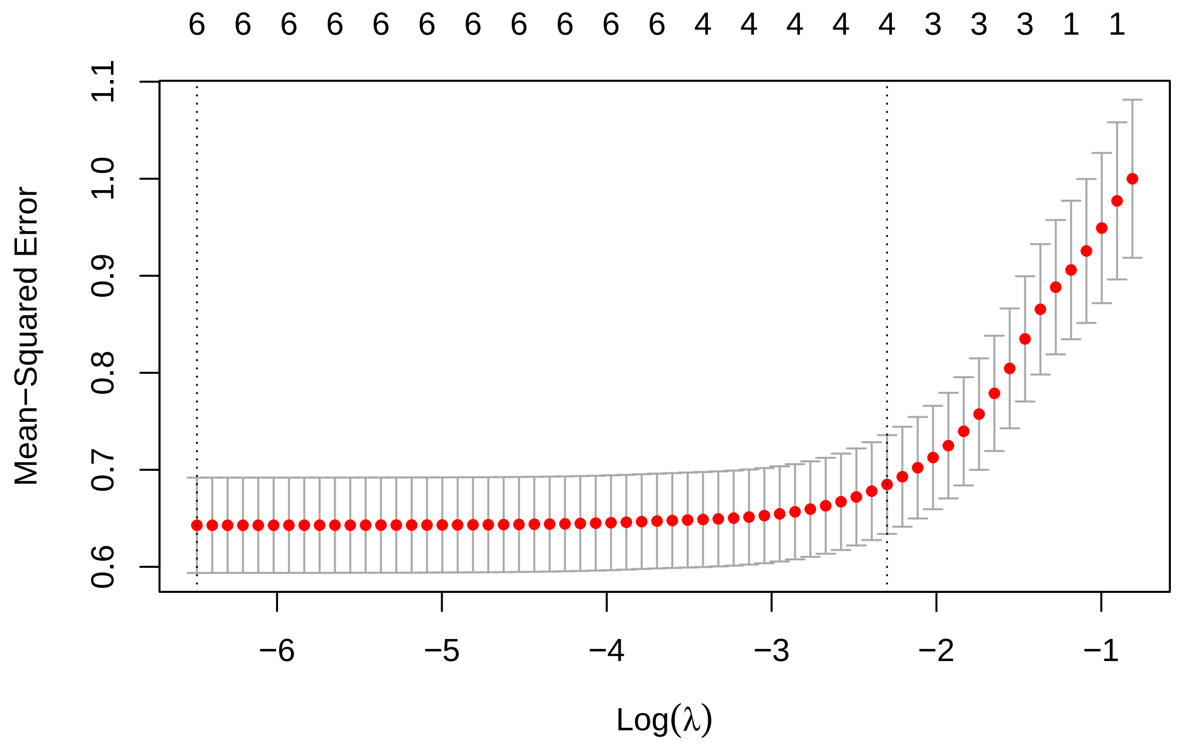
PART B:

QB1: Build a Lasso regression model to predict Sales based on all other attributes ("Price", "Advertising", "Population", "Age", "Income" and "Education"). What is the best value of lambda for such a lasso model?

```r
library(ISLR)
library(dplyr)
library(glmnet)
library(caret)

Carseats_Filtered <- Carseats %>% select("Sales","Price","Advertising","Population","Age","Income","Edu

set.seed(420)
# creating a normalized model on the filtered Carseats dataset
norm_model <- preProcess(Carseats_Filtered, method = c("center","scale"))
# applying scaling on the dataset
Carseats_normalized <- predict(norm_model,Carseats_Filtered)

# selecting the features and labels separately, and changing them into a matrix format
Carseats_x <- as.matrix(Carseats_normalized[,-c(1)])
Carseats_y <- as.matrix(Carseats_normalized[,1])
```

Building a Lasso regression model

```r
set.seed(420)
Cs_lasso_cv <- cv.glmnet(
  x = Carseats_x,
  y = Carseats_y,
  alpha = 1
)
# plotting the graph
plot(Cs_lasso_cv)
```



```r
# optimal lambda value
Cs_lasso_cv$lambda.min
```

```
## [1] 0.001524481
```

The best/optimal value for this lasso model is 0.001524481

QB2: What is the coefficient for the price (normalized) attribute in the best model (i.e. model with the optimal lambda)?

```
# identifying the coefficients for the best model with the optimal lambda value
coef(Cs_lasso_cv, s = "lambda.min")
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  3.723500e-16
## Price       -4.793834e-01
## Advertising  2.932098e-01
## Population  -4.624934e-02
## Age         -2.792202e-01
## Income       1.024459e-01
## Education   -3.223128e-02
```

Coefficient for the Price is -0.4793834

QB3: How many attributes remain in the model if lambda is set to 0.01? How that number changes if lambda is increased to 0.1? Do you expect more variables to stay in the model (i.e., to have non-zero coefficients) as we increase lambda?

```
coef(Cs_lasso_cv, s = 0.01)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  3.686308e-16
## Price       -4.696889e-01
## Advertising  2.815718e-01
## Population  -3.323443e-02
## Age         -2.693300e-01
## Income       9.585212e-02
## Education   -2.330455e-02
```

```
coef(Cs_lasso_cv, s = 0.1)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  3.499749e-16
## Price       -3.691394e-01
## Advertising  1.839178e-01
## Population   .
## Age         -1.684796e-01
## Income       1.925921e-02
## Education    .
```

1. Lambda at 0.01: All the attributes still remain in the model.

2. Lambda at 0.1: Population and Education are dropped from the model. The rest remain in the model. As the value of lambda increases, more and more variables/attributes start to drop off (i.e., there will be less variables)

QB4: Build an elastic-net model with alpha set to 0.6. What is the best value of lambda for such a model?

```
set.seed(420)
Cs_elastic_cv <- cv.glmnet(
  x = Carseats_x,
  y = Carseats_y,
  alpha = 0.6
)
Cs_elastic_cv$lambda.min
```

```
## [1] 0.002315083
```

The optimal/best lambda for the elastic-net model is 0.002315083