ADMPL- Assignment 1

Steven Pavliga

2025-03-02

QA1. What is the main purpose of regularization when training predictive models?

Regularization encompasses various ways to control the complexity/variance in a model. It is typically done by adding a penalty of sorts to reduce complexity and/or overfitting. Reducing a model's complexity and fit can often improve generalization, enhancing the potential impact of the model.

QA2. What is the role of a loss function in a predictive model? And name two common loss functions for regression models and two common loss functions for classification models.

Loss functions essentially measure the distance between a model's predicted output and the actual value of that unit. Such distances can be measured in different ways. The goal of the loss function is to be minimized, resulting in the best fitting model.

Two loss functions for regression models are Mean Squared Error (MSE) and Root Mean Squared Error (RMSE). MSE takes the distance between the predicted output and the actual value, and squares the value. A distance of 2 would result in an MSE of 4, but a distance of 4 would result in 16- a much higher impact per unit of distance. Being said, higher distances have exponentially higher impacts. The total MSE is the sum of the distance of each unit squared. RMSE is the same, except the square root of the MSE is instead taken. RMSE will naturally be a smaller number which may be more useful in a real-world instance.

Two loss functions for classification models are log loss and hinge loss. Log loss takes into account how far the predicted output is from the actual value, in terms of the probability value. Log loss applies a heavy penalty when a probability value is strongly in one direction but the actual value is incorrect. Hinge loss does not take probability into account, and instead focuses on how close an output is to the decision value. Log loss takes even correct classifications into small consideration, however hinge loss does not take correct classifications into consideration.

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.

You would not be able to fully trust the model, primarily due to the danger of overfitting. With the complexity of the model as well as the size of the dataset, it's possible that the model will not be able to generalize to new data. A first resort would be to split the dataset into training and test partitions depending on how small the dataset actually is. Additionally, incorporating cross validation or regularization may be helpful.

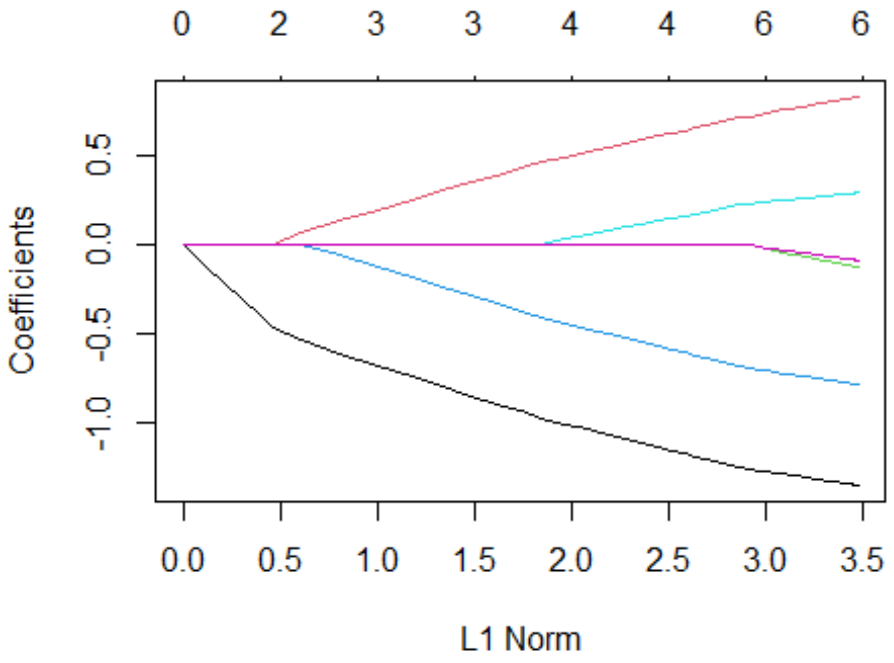
QA4. What is the role of the lambda parameter in regularized linear models such as Lasso or Ridge regression models?

Lambda essentially balances ensuring the model is well fit to the data, but also keeping parameters small to reduce complexity and prevent overfitting. As the lambda value increases, the size of the coefficients decrease and its effects become more noticeable in the model. Complexity of the model will continue to decrease, but after a point the model starts to run the risk of underfitting.

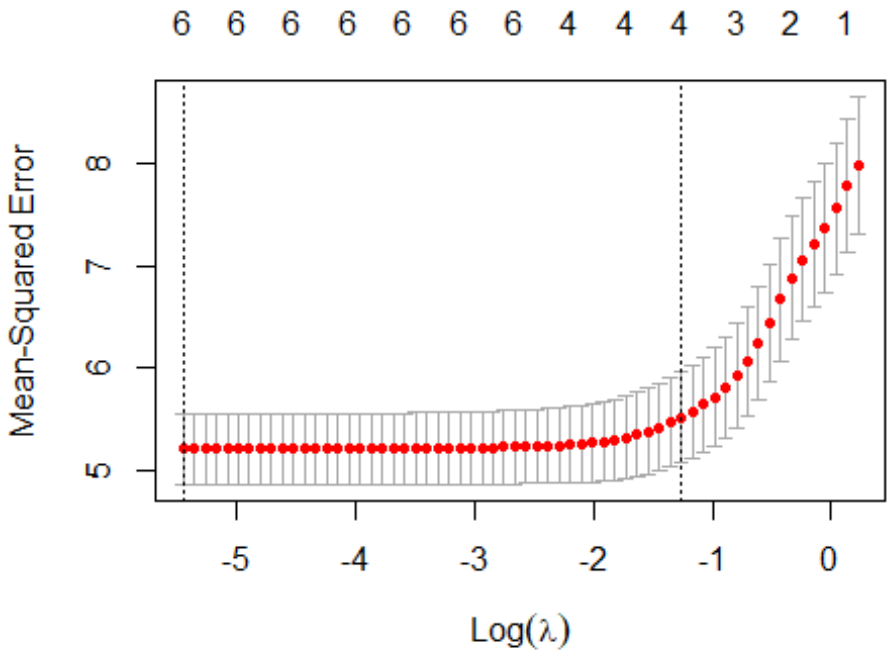
QB1. Build a Lasso regression model to predict Sales based on all other attributes ("Price", "Advertising", "Population", "Age", "Income" and "Education"). What is the best value of lambda for such a lasso model? (Hint1: Do not forget to scale your input attributes – you can use the caret preprocess() function to scale and center the data. Hint 2: glment library expect the input attributes to be in the matrix format. You can use the as.matrix() function for converting)

```
library(ISLR)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(glmnet)

## Warning: package 'glmnet' was built under R version 4.4.2

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

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

CarseatsPreprocessed <- preProcess(Carseats_Filtered[, -1], method = c("cente
r", "scale"))
CarseatsScaled <- predict(CarseatsPreprocessed, Carseats_Filtered)
```

```
x <- as.matrix(CarseatsScaled[, -1])
y <- CarseatsScaled$Sales

Fit <- glmnet(x, y)
plot(Fit)
```




```
set.seed(123)
LassoCV <- cv.glmnet(x, y, alpha = 1)
plot(LassoCV)
```




```
print(LassoCV$lambda.min)
```

```
## [1] 0.004305309
```

The best lambda value is 0.0043.

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

```
Coefficients <- coef(LassoCV, s = LassoCV$lambda.min)

CoefficientsPrice <- Coefficients["Price", ]
print(CoefficientsPrice)
```

```
## [1] -1.353834
```

The coefficient for price is -1.3538.

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?

```
Coefficients001 <- coef(LassoCV, s = 0.01)
PosCoefficients001 <- sum(Coefficients001 != 0) - 1
print(PosCoefficients001)
```

```
## [1] 6
```

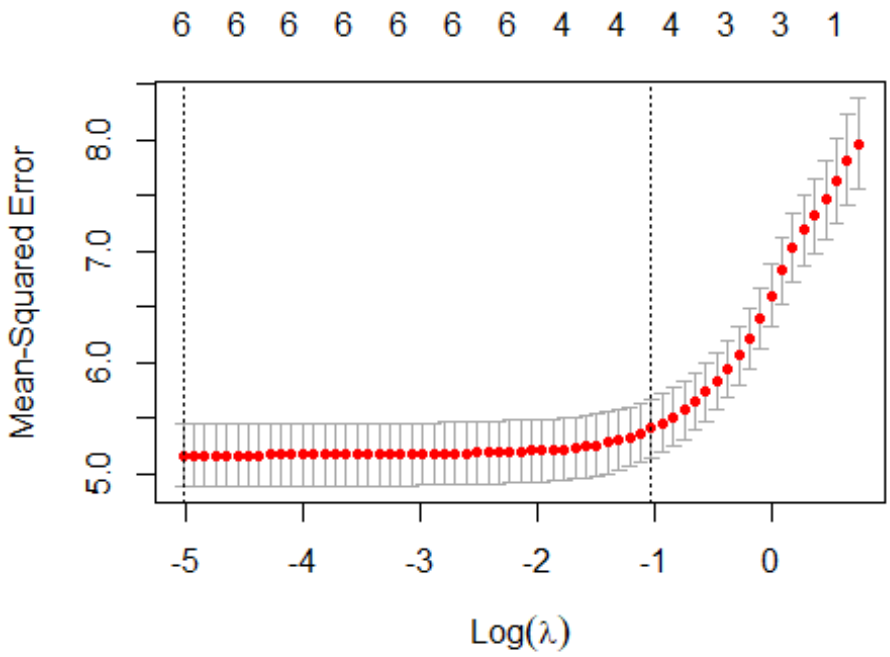
```
Coefficients01 <- coef(LassoCV, s = 0.1)
PosCoefficients01 <- sum(Coefficients01 != 0) - 1
print(PosCoefficients01)
```

```
## [1] 4
```

6 attributes remain at 0.01, with 4 remaining at 0.1. As lambda increases, less attributes will remain as the others will become zero. With fewer attributes, the model will become less complex.

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

```
ElasticNet <- cv.glmnet(x, y, alpha = 0.6)
plot(ElasticNet)
```




```
print(ElasticNet$lambda.min)
```

```
## [1] 0.006538062
```

The best value for lambda is 0.0065.