Solving Multicollinearity Problems: Ridge and Lasso Regression

Dr Austin R Brown

Introduction

- Recall from a prior class session that something we have to be mindful of in multiple linear regression (regardless if our goal is prediction or inference) is multicollinearity.
- While multicollinearity is not an assumption in the same way that normality and constant variance are (i.e., for inference to be valid), it can still cause us problems in terms of the precision of our coefficient estimates.
- Why is this?

Introduction

- Recall that the variance of an individual regression coefficient estimator is:

$$Var[\hat{\beta}_j] = \sigma^2 VIF_j$$

- where:

$$VIF_j = \frac{1}{1 - R_j^2}$$

Introduction

- Also recall that for an individual coefficient t-test, our test statistic is computed as:

$$t = \frac{\hat{\beta}_j}{\sqrt{MSE(VIF_j)}}$$

- So if VIF_j is large, it has an effect on both the precision of our estimates (as variance goes up, precision goes down), as well as the statistical power of our individual coefficient tests.
  - This is problematic!!

Introduction

- So what do we do? Occassionally it may be appropriate to delete predictor variables, but this should really be an exception and not the first solution.
- While traditional methods involve scaling and standardizing predictor variables, the modern solution to these problems is by using either <u>**Ridge**</u> or <u>**Lasso**</u> regression.
- Let's walk through an example using the `ISLR::Credit` dataset and then explain what each method does!

Example: Credit Data

- Suppose we want to predict credit card balances (Balance) using Income, Limit, Rating, and Student Status:

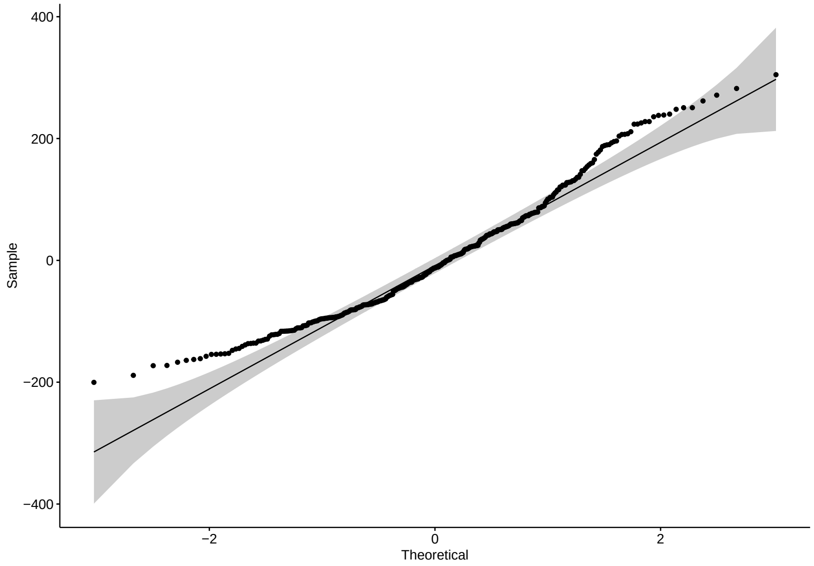
```
credit <- ISLR::Credit
credit_mod <- lm(Balance~Income+Limit+Rating+Student,
                 data=credit)
```

Check Assumptions: Normality

- Let's start with a check of normality, first using the QQ-plot:

```
library(ggpubr)
credit_mod$residuals |>
  ggqqplot()
```

Check Assumptions: Normality



Check Assumptions: Normality

- Okay, so it isn't perfect. Not bad, but we definitely have evidence of non-normality. Let's check on the K-S test:

```
ks.test(rstudent(credit_mod),"pnorm")
```

```

	Asymptotic one-sample Kolmogorov-Smirnov test

data:  rstudent(credit_mod)
D = 0.06793, p-value = 0.04986
alternative hypothesis: two-sided
```

- So we have evidence of slight non-normality, so that's something we need to be mindful of!
  - Let's move on to constant variance!

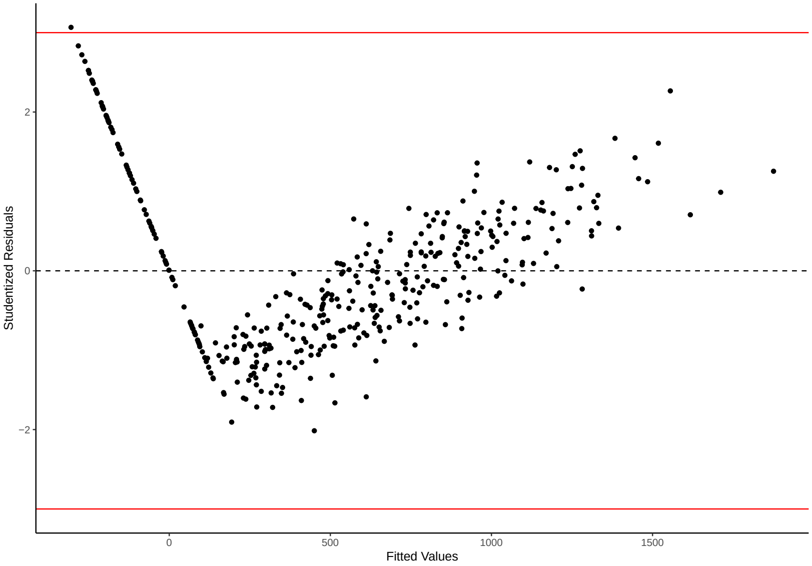
Check Assumptions: Constant Variance

- Let's evaluate constant variance using our scatterplot technique:

```r
library(moderndive)

credit_mod |>
  get_regression_points() |>
  mutate(`Studentized Residuals` = rstudent(credit_mod)) |>
  ggplot(aes(x=`Balance_hat`,y=`Studentized Residuals`)) +
  geom_point() +
  geom_hline(yintercept = 3, color='red') +
  geom_hline(yintercept = -3, color='red') +
  geom_hline(yintercept = 0, color='black',
             linetype='dashed') +
  labs(y="Studentized Residuals",
       x="Fitted Values") +
  theme_classic()
```

Check Assumptions: Constant Variance



Check Assumptions: Constant Variance

- We very clearly have non-constant variance. But confirming using the Breush-Pagan Test:

```
library(lmtest)

credit_mod |>
  bptest()
```

```

	studentized Breusch-Pagan test

data:  credit_mod
BP = 110.2, df = 4, p-value < 2.2e-16
```

Checking Assumptions: Multicollinearity

- Okay, so we have pretty substantial issues with normality and constant variance we need to be aware of.
- What about multicollinearity?
- Contextually, we know that we probably have some degree of collinearity considering that income and credit limit are typically positively related.
- But let's check using the techniques we learned!

Checking Assumptions: Multicollinearity

```r
library(olsrr)

credit_mod |>
  ols_vif_tol()
```

```
   Variables   Tolerance        VIF
1     Income 0.371555337   2.691389
2      Limit 0.006186284 161.647942
3     Rating 0.006207596 161.092961
4 StudentYes 0.995980106   1.004036
```

Ridge Regression

- Since VIF of greater than 10 is considered problematic, these values in excess of 100 are quite severe and something we need to address.
- So how do we do it?
- One clever technique is by using a method called **ridge regression**.

Ridge Regression

- Since one of the main issues with multicollinearity is that our $X^T X$ matrix becomes nearly non-invertable, we can add a small constant, $lambda \geq 0$, to the $X^T X$ matrix. This makes our normal equations:

$$(X^T X + \lambda I)\hat{\beta}_R = X^T Y$$

- Remember, the diagonals of the $(X^T X)^{-1}$ are the VIF values for each of the predictors. So we can sort of think of the diagonals of the $X^T X$ matrix as the inverse of VIF.
  - This implies that if VIF is big that their inverse will be small.

Ridge Regression

- So by adding a small constant to the diagonals, we're sort of intentionally preventing multicollinearity. So our ridge regression coefficient estimates will be:

$$\hat{\beta}_R = (X^T X + \lambda I)^{-1} X^T Y$$

- Slick solution, right? So what's the tradeoff? Here, we're trading a little bit of bias for a more precise estimate.
  - We know how VIF is being "brute forced" downward. How does the ridge regression coefficient estimates become biased?

Ridge Regression

$$E[\hat{\beta}_R] = E[(X^TX + \lambda I)^{-1}X^TY]$$

$$= (X^TX + \lambda I)^{-1}X^TE[Y]$$

$$= (X^TX + \lambda I)^{-1}X^TX\beta \neq \beta$$

- Because the expected value of the estimate does not equal the parameter it is estimating, the estimator is said to be biased.

Ridge Regression

- What we need to know then is how to choose λ. The typical procedure we use is to try several different values of λ and see at what value our regression coefficients begin to stabilize.

- We typically use a plot called the “ridge trace” to ascertain the value, but can also use built in R functions to help us decide.

- Let's see how we do this in R.

Ridge Regression

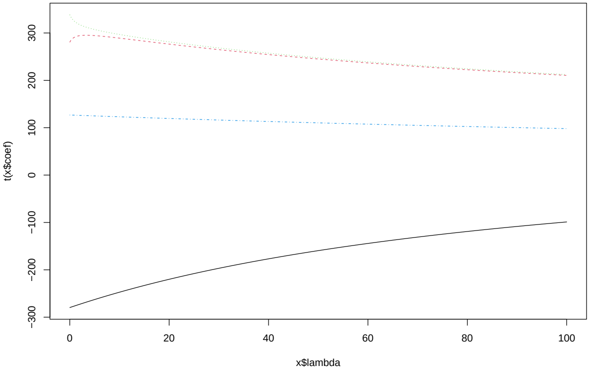
- Let's use the `lm.ridge` function which is part of the `MASS` package to do this:

```r
library(tidyverse)
library(MASS)
ridge <- lm.ridge(Balance~Income+Limit+Rating+Student,
                  data=credit,lambda=seq(0.01,100,0.01))
```

Ridge Regression

- Then, we can plot our ridge trace to visually identify the optimum value of λ:

```
plot(ridge)
```



Ridge Regression

- Visually, it looks like maybe around 50-60 is where the estimates begin to stabilize.
- We can also estimate our value of λ using the `select` function which is part of the `MASS` package:

```
MASS::select(ridge)
```

```
modified HKB estimator is 0.07100619
modified L-W estimator is 0.1016979
smallest value of GCV  at 1.08
```

Ridge Regression

- So using $\lambda = 1.08$, let's refit the model and check out our coefficient estimates:

```
## Let's use the GCV estimator lambda = 1.08 ##

lambda <- 1.08

## Refit Ridge ##

refit_ridge <- lm.ridge(Balance~Income+Limit+Rating+Student,
                        data=credit,lambda=lambda)

## Get Coefficients ##

round(coef(refit_ridge),4)
```

Ridge Regression

```
                 Income        Limit       Rating StudentYes 
-509.9428       -7.8376       0.1265       2.0934   421.4066 
```