Ridge Regression

XI(✗) – X’s are not independent (are correlated)

Ridge Regression

- **OLS** finds regression coefficients that **minimize** the SSE:

$$SSE = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{1i} - \beta_2 x_{2i} - etc.)^2$$

- **Ridge** regression finds coefficients that **minimize**:

$$SSE(R) = SSE + \text{shrinkage penalty} = SSE + \lambda\,(\beta_1^2 + \beta_2^2 + \beta_3^2 + etc.)$$

- This seems like a complicated idea but the **concept** is simple:
  - **Ridge** regression fits a line that **minimizes SSE(R)**
  - That is, **Ridge** minimizes **SSE plus** a **penalty**
  - We can vary the penalty λ thus **controlling** the **shrinkage**
  - If we set **$\lambda = 0$**, Ridge minimizes **SSE** → same as **OLS**
  - If we set λ **very large**, then the resulting **β's** have to be **very small** → i.e., we **shrink** the coefficients
  - So if **$\lambda = \infty$** Ridge yields the **null model $y = \beta_0$**
  - The goal is to **select** the λ that **minimizes** the **Test MSE**

When/How to use Ridge Regression

- **Ridge** regression is particularly useful for **large models** with **dimensionality** issues (e.g., multi-collinearity)
- Ridge regression **coefficients** are **biased** because the OLS regression line is altered through the shrinking
- But it tends to **lower** the **variance** of the estimated coefficients when there are **dimensionality** issues
- **OLS** is **"scale invariant"**; this means that if we re-scale a variable unit (e.g., from feet to meters, from dollars to thousands), the regression coefficients will simply change proportionally.
- Because the penalty in **Ridge** regression is based on the sum of the squared coefficients**, re-scaling** will **change** the **results** disproportionately
- Therefore, it is standard practice to:
  - ✓ **Standardize** the predictors in Ridge regression models
  - ✓ **Compare** Ridge regression with several **λ's** to other models (e.g., OLS) with **cross-validation** measures of the **Test MSE**

R Tips

`Glmnet(){glmnet}` → Package for **Ridge** and LASSO regressions. Note: `glmnet()` function has a different syntax and it requires that we define the predictor variables as a matrix **`X`** and the response as a vector **`Y`**

`X=model.matrix(y~x1+x2+etc.,data=dataName)` → Creates **`X`** matrix using only independent variables

`Y=dataName$y` → Creates **`Y`** vector using column `y`

`ridge.fit=glmnet(X,Y,alpha=0, lambda=0)` → **`alpha=0`** fits a **Ridge** regression; **`lambda=0`** fits an OLS regression (i.e., no shrinkage)

`ridge.fit=glmnet(X,Y,alpha=0, lambda=1000)` → a lot of shrinkage

`ridge.fit=glmnet(X,Y,alpha=0, lambda=1000000)` → as lambda gets very large (approaches ∞) most coefficients are shrunk to almost 0, yielding a **null** model (i.e., **just the intercept**)

```
ridge.fit=glmnet(X,Y,alpha=0,
          lambda=c(0,10,100,1000, 1000000))
```
→ run multiple shrinkage values of `lambda`

`coef(ridge.fit)` → Lists all ridge coefficients sorted from the largest to lowest `lambda`

Kogod School
of
Business