Ridge Regression

Nate Wells

Math 243: Stat Learning

October 28tht, 2020

Outline

In today's class, we will. . .

- Discuss Ridge Regression as an alternate method of model selection
- Implement Ridge Regression using the `glmnet` package

Section 1

Penalized Regression

Motivation 1

Suppose we wish to build a linear model for Y using predictors $X_1, \ldots, X_p$ using n observations.

Motivation 1

Suppose we wish to build a linear model for Y using predictors $X_1, \ldots, X_p$ using n observations.

Generally, under what circumstances will the **full model** perform well, compared to other subset models? *(Think about the Bias-Variance Tradeoff)*

Motivation 1

Suppose we wish to build a linear model for Y using predictors $X_1, \ldots, X_p$ using n observations.

Generally, under what circumstances will the **full model** perform well, compared to other subset models? *(Think about the Bias-Variance Tradeoff)*

- If irreducible error is small ($\epsilon \sim N(0, \sigma^2)$ with $\sigma \approx 0$)

Motivation 1

Suppose we wish to build a linear model for Y using predictors $X_1, \dots, X_p$ using n observations.

Generally, under what circumstances will the **full model** perform well, compared to other subset models? *(Think about the Bias-Variance Tradeoff)*

- If irreducible error is small ($\epsilon \sim N(0, \sigma^2)$ with $\sigma \approx 0$)
- If model variance is low and model bias is high.

Motivation 1

Suppose we wish to build a linear model for Y using predictors $X_1, \ldots, X_p$ using n observations.

Generally, under what circumstances will the **full model** perform well, compared to other subset models? *(Think about the Bias-Variance Tradeoff)*

- If irreducible error is small ($\epsilon \sim N(0, \sigma^2)$ with $\sigma \approx 0$)
- If model variance is low and model bias is high.

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is the best fitting linear model using X_1 and X_2.

Motivation 1

Suppose we wish to build a linear model for Y using predictors $X_1, \ldots, X_p$ using n observations.

Generally, under what circumstances will the **full model** perform well, compared to other subset models? *(Think about the Bias-Variance Tradeoff)*

- If irreducible error is small ($\epsilon \sim N(0, \sigma^2)$ with $\sigma \approx 0$)
- If model variance is low and model bias is high.

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is the best fitting linear model using X_1 and X_2.

How might the bias and variance of the following model compare?

$$\hat{y} = 10 + 0.01x_1 + 500x_2$$

Motivation 1

Suppose we wish to build a linear model for Y using predictors $X_1, \ldots, X_p$ using n observations.

Generally, under what circumstances will the **full model** perform well, compared to other subset models? *(Think about the Bias-Variance Tradeoff)*

- If irreducible error is small ($\epsilon \sim N(0, \sigma^2)$ with $\sigma \approx 0$)
- If model variance is low and model bias is high.

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is the best fitting linear model using X_1 and X_2.

How might the bias and variance of the following model compare?

$$\hat{y} = 10 + 0.01x_1 + 500x_2$$

When might this new model have lower MSE than the original model?

Motivation 2

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is again the best fitting linear model using X_1 and X_2.

Motivation 2

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is again the best fitting linear model using X_1 and X_2.

- Are we justified in saying that X_2 is a more important predictor than X_1?

Motivation 2

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is again the best fitting linear model using X_1 and X_2.

- Are we justified in saying that X_2 is a more important predictor than X_1?

Suppose we first standardize X_1 and X_2 by subtacting off their means and dividing by their standard deviations:

$$Z_1 = \frac{X_1 - \mu_1}{\sigma_1} \qquad Z_2 = \frac{X_2 - \mu_2}{\sigma_2}$$

Motivation 2

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is again the best fitting linear model using X_1 and X_2.

- Are we justified in saying that X_2 is a more important predictor than X_1?

Suppose we first standardize X_1 and X_2 by subtacting off their means and dividing by their standard deviations:

$$Z_1 = \frac{X_1 - \mu_1}{\sigma_1} \qquad Z_2 = \frac{X_2 - \mu_2}{\sigma_2}$$

- If we build a model and find $\hat{y} = 10 + 0.01z_1 + 1000z_2$, where Z_1 and Z_2 are standardized, are we now justified in saying that Z_2 is more important than Z_1?

Motivation 2

Suppose $\hat{y} = 10 + 0.01x_1 + 1000x_2$ is again the best fitting linear model using X_1 and X_2.

- Are we justified in saying that X_2 is a more important predictor than X_1?

Suppose we first standardize X_1 and X_2 by subtacting off their means and dividing by their standard deviations:

$$Z_1 = \frac{X_1 - \mu_1}{\sigma_1} \qquad Z_2 = \frac{X_2 - \mu_2}{\sigma_2}$$

- If we build a model and find $\hat{y} = 10 + 0.01z_1 + 1000z_2$, where Z_1 and Z_2 are standardized, are we now justified in saying that Z_2 is more important than Z_1?
  - How might the variance and bias of the following model compare to the standarized model?

$$\hat{y} = 10 + 0.02z_2 + 500z_1$$

Ridge Regression

Recall that least squares regression estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ for

$$\hat{y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

are obtained by finding the values of β that minimize

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

Ridge Regression

Recall that least squares regression estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ for

$$\hat{y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

are obtained by finding the values of β that minimize

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

To perform **Ridge Regression**, we instead find coefficients β that minimize

$$\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_i^2 \qquad \text{where } \lambda \geq 0 \text{ is tuning parameter}$$

Ridge Regression

Recall that least squares regression estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ for

$$\hat{y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

are obtained by finding the values of β that minimize

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

To perform **Ridge Regression**, we instead find coefficients β that minimize

$$\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_i^2 \qquad \text{where } \lambda \geq 0 \text{ is tuning parameter}$$

Why?

Ridge Regression

Recall that least squares regression estimates $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p$ for

$$\hat{y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

are obtained by finding the values of β that minimize

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

To perform **Ridge Regression**, we instead find coefficients β that minimize

$$\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_i^2 \qquad \text{where } \lambda \geq 0 \text{ is tuning parameter}$$

Why?

- The term $\lambda \sum_{i=1}^{p} \beta_i^2$ is the **shrinkage penalty**, and is small when the β are small.

Ridge Regression

Recall that least squares regression estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ for

$$\hat{y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

are obtained by finding the values of β that minimize

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

To perform **Ridge Regression**, we instead find coefficients β that minimize

$$\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_i^2 \qquad \text{where } \lambda \geq 0 \text{ is tuning parameter}$$

Why?

- The term $\lambda \sum_{i=1}^{p} \beta_i^2$ is the **shrinkage penalty**, and is small when the β are small.
- With a shrinkage penalty, the algorithm prefers models with lower coefficients.

Ridge Regression

Recall that least squares regression estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ for

$$\hat{y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

are obtained by finding the values of β that minimize

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

To perform **Ridge Regression**, we instead find coefficients β that minimize

$$\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_i^2 \qquad \text{where } \lambda \geq 0 \text{ is tuning parameter}$$

Why?

- The term $\lambda \sum_{i=1}^{p} \beta_i^2$ is the **shrinkage penalty**, and is small when the β are small.
- With a shrinkage penalty, the algorithm prefers models with lower coefficients.
- This tends to reduce variance, at the cost of increased bias.

Effects of the Tuning Parameter

Goal: Find β which minimize $\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_p^2$

Effects of the Tuning Parameter

Goal: Find β which minimize $\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_p^2$

What will happen to β_0 as $\lambda \to \infty$? As $\lambda \to 0$?

Effects of the Tuning Parameter

Goal: Find β which minimize $\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_p^2$

What will happen to β_0 as $\lambda \to \infty$? As $\lambda \to 0$?

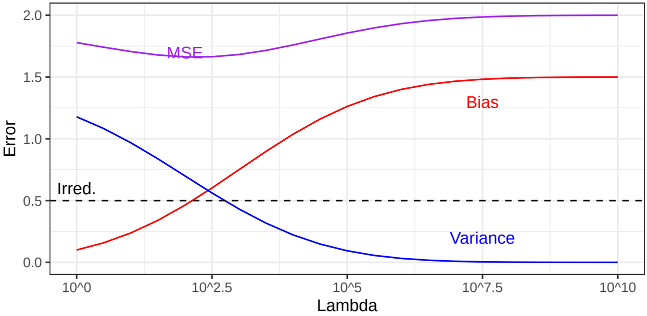
What happens to MSE as $\lambda \to 0$ or $\lambda \to 1$?

Effects of the Tuning Parameter

Goal: Find β which minimize $\mathrm{RSS} + \lambda \sum_{i=1}^{p} \beta_p^2$

What will happen to β_0 as $\lambda \to \infty$? As $\lambda \to 0$?

What happens to MSE as $\lambda \to 0$ or $\lambda \to 1$?



Standardization

Suppose X_1 and X_2 are non-standardized predictors and the best fitting linear model is

$$\hat{y} = 10 + 0.01x_1 + 1000x_2$$

Standardization

Suppose X_1 and X_2 are non-standardized predictors and the best fitting linear model is

$$\hat{y} = 10 + 0.01x_1 + 1000x_2$$

What type of models will ridge regression prefer (in terms of β_1 and β_2)?

Standardization

Suppose X_1 and X_2 are non-standardized predictors and the best fitting linear model is

$$\hat{y} = 10 + 0.01x_1 + 1000x_2$$

What type of models will ridge regression prefer (in terms of β_1 and β_2)?

- Recall the shrinkage penalty is $\lambda \sum_{i=1}^{2} \beta_i^2 = \lambda(1000^2 + 0.01^2)$
- Since $\beta_2 = 1000$ is much larger than $\beta_1 = 0.01$, ridge regression will prioritize reducing β_2 over β_1.

Standardization

Suppose X_1 and X_2 are non-standardized predictors and the best fitting linear model is

$$\hat{y} = 10 + 0.01x_1 + 1000x_2$$

What type of models will ridge regression prefer (in terms of β_1 and β_2)?

- Recall the shrinkage penalty is $\lambda \sum_{i=1}^{2} \beta_i^2 = \lambda(1000^2 + 0.01^2)$
- Since $\beta_2 = 1000$ is much larger than $\beta_1 = 0.01$, ridge regression will prioritize reducing β_2 over β_1.
- Will this actually produce good MSE for a fixed λ?

Standardization

Suppose X_1 and X_2 are non-standardized predictors and the best fitting linear model is

$$\hat{y} = 10 + 0.01x_1 + 1000x_2$$

What type of models will ridge regression prefer (in terms of β_1 and β_2)?

- Recall the shrinkage penalty is $\lambda \sum_{i=1}^{2} \beta_i^2 = \lambda(1000^2 + 0.01^2)$
- Since $\beta_2 = 1000$ is much larger than $\beta_1 = 0.01$, ridge regression will prioritize reducing β_2 over β_1.
- Will this actually produce good MSE for a fixed λ?
  - Only if standard deviation of x_2 is much, much larger than that of x_1

Standardization

Suppose X_1 and X_2 are non-standardized predictors and the best fitting linear model is

$$\hat{y} = 10 + 0.01x_1 + 1000x_2$$

What type of models will ridge regression prefer (in terms of β_1 and β_2)?

- Recall the shrinkage penalty is $\lambda \sum_{i=1}^{2} \beta_i^2 = \lambda(1000^2 + 0.01^2)$
- Since $\beta_2 = 1000$ is much larger than $\beta_1 = 0.01$, ridge regression will prioritize reducing β_2 over β_1.
- Will this actually produce good MSE for a fixed λ?
    - Only if standard deviation of x_2 is much, much larger than that of x_1

Ridge regression is most efficient if predictors are standardized first.

Standardization

Suppose X_1 and X_2 are non-standardized predictors and the best fitting linear model is

$$\hat{y} = 10 + 0.01x_1 + 1000x_2$$

What type of models will ridge regression prefer (in terms of β_1 and β_2)?

- Recall the shrinkage penalty is $\lambda \sum_{i=1}^{2} \beta_i^2 = \lambda(1000^2 + 0.01^2)$
- Since $\beta_2 = 1000$ is much larger than $\beta_1 = 0.01$, ridge regression will prioritize reducing β_2 over β_1.
- Will this actually produce good MSE for a fixed λ?
  - Only if standard deviation of x_2 is much, much larger than that of x_1

Ridge regression is most efficient if predictors are standardized first.

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{\hat{\sigma}_j}$$

Where x_{ij} is the ith observation of the jth predictor, $\bar{x}_j$ is the sample mean of the jth predictor, and $\hat{\sigma}_j$ is the sample st. dev. of the jth predictor.

Section 2

Ridge Regression in R

The Data

The video_games dataset contains sales information and attributes for a random selection of 1000 video games.

- Suppose we want to predict Global_Sales based on NA_Sales, EU_Sales, JP_Sales, Critic_Score, User_Score, and Rating.

The Data

The `video_games` dataset contains sales information and attributes for a random selection of 1000 video games.

- Suppose we want to predict `Global_Sales` based on `NA_Sales`, `EU_Sales`, `JP_Sales`, `Critic_Score`, `User_Score`, and `Rating`.

```
## # A tibble: 6 x 8
##   Name     Global_Sales NA_Sales EU_Sales JP_Sales Critic_Score User_Score Rating
##   <chr>           <dbl>    <dbl>    <dbl>    <dbl>        <dbl>      <dbl> <chr>
## 1 Rayman~          0.07     0.06     0.02    0               75        7.9 E
## 2 Army o~          0.28     0.11     0.11    0.01            58        6.7 M
## 3 Forza ~          1.4      0.5      0.78    0.01            86        8.2 E10+
## 4 Street~          4.16     2.03     1.04    0.580           94        7.3 T
## 5 X-Men ~          0.2      0.15     0.03    0               55        8.1 E10+
## 6 Bomber~          0.08     0.06     0.02    0               70        7.9 E
```

The Data

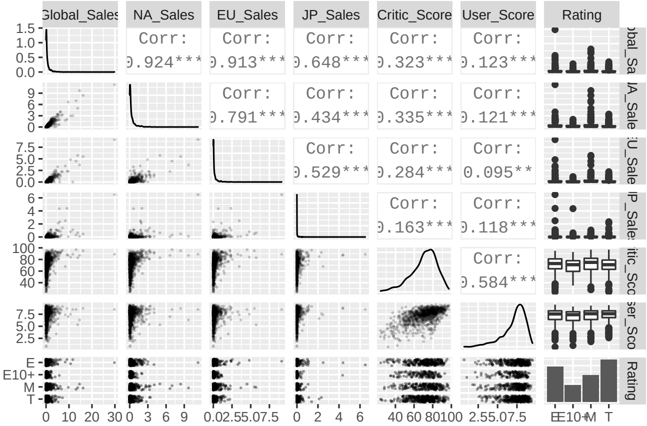
The `video_games` dataset contains sales information and attributes for a random selection of 1000 video games.

- Suppose we want to predict `Global_Sales` based on `NA_Sales`, `EU_Sales`, `JP_Sales`, `Critic_Score`, `User_Score`, and `Rating`.

```
## # A tibble: 6 x 8
##   Name    Global_Sales NA_Sales EU_Sales JP_Sales Critic_Score User_Score Rating
##   <chr>          <dbl>    <dbl>    <dbl>    <dbl>        <dbl>      <dbl> <chr>
## 1 Rayman~         0.07     0.06     0.02    0               75        7.9 E
## 2 Army o~         0.28     0.11     0.11    0.01            58        6.7 M
## 3 Forza ~         1.4      0.5      0.78    0.01            86        8.2 E10+
## 4 Street~         4.16     2.03     1.04    0.580           94        7.3 T
## 5 X-Men ~         0.2      0.15     0.03    0               55        8.1 E10+
## 6 Bomber~         0.08     0.06     0.02    0               70        7.9 E
```

- What are some possible problems with the full model?

Exploration



Pre-Processing

First, we perform some preliminary data formatting:

Pre-Processing

First, we perform some preliminary data formatting:

```
x<-model.matrix(Global_Sales ~. - Name, data = video_games)[,-1]
y<-video_games$Global_Sales
head(x)
```

```
##   NA_Sales EU_Sales JP_Sales Critic_Score User_Score RatingE10+ RatingM RatingT
## 1     0.06     0.02     0.00           75        7.9          0       0       0
## 2     0.11     0.11     0.01           58        6.7          0       1       0
## 3     0.50     0.78     0.01           86        8.2          1       0       0
## 4     2.03     1.04     0.58           94        7.3          0       0       1
## 5     0.15     0.03     0.00           55        8.1          1       0       0
## 6     0.06     0.02     0.00           70        7.9          0       0       0
```

Pre-Processing

First, we perform some preliminary data formatting:

```
x<-model.matrix(Global_Sales ~. - Name, data = video_games)[,-1]
y<-video_games$Global_Sales
head(x)
```

```
##   NA_Sales EU_Sales JP_Sales Critic_Score User_Score RatingE10+ RatingM RatingT
## 1     0.06     0.02     0.00           75        7.9          0       0       0
## 2     0.11     0.11     0.01           58        6.7          0       1       0
## 3     0.50     0.78     0.01           86        8.2          1       0       0
## 4     2.03     1.04     0.58           94        7.3          0       0       1
## 5     0.15     0.03     0.00           55        8.1          1       0       0
## 6     0.06     0.02     0.00           70        7.9          0       0       0
```

- The `model.matrix` function creates a `matrix` of predictors and converts all categorical variables to dummy variables

Pre-Processing

First, we perform some preliminary data formatting:

```
x<-model.matrix(Global_Sales ~. - Name, data = video_games)[,-1]
y<-video_games$Global_Sales
head(x)
```

```
##   NA_Sales EU_Sales JP_Sales Critic_Score User_Score RatingE10+ RatingM RatingT
## 1     0.06     0.02     0.00           75        7.9          0       0       0
## 2     0.11     0.11     0.01           58        6.7          0       1       0
## 3     0.50     0.78     0.01           86        8.2          1       0       0
## 4     2.03     1.04     0.58           94        7.3          0       0       1
## 5     0.15     0.03     0.00           55        8.1          1       0       0
## 6     0.06     0.02     0.00           70        7.9          0       0       0
```

- The `model.matrix` function creates a `matrix` of predictors and converts all categorical variables to dummy variables

We also create vector `grid` of suitable tuning parameters λ.

```
## [1] 10000000000  7564633276  5722367659  4328761281  3274549163  2477076356
```

The `glmnet` package

We use the `glmnet` function in the `glmnet` package in order to perform Ridge Regression for a variety of values of the tuning parameter λ.

The glmnet package

We use the glmnet function in the glmnet package in order to perform Ridge Regression for a variety of values of the tuning parameter λ.

```
library(glmnet)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

The glmnet package

We use the glmnet function in the glmnet package in order to perform Ridge Regression for a variety of values of the tuning parameter λ.

```
library(glmnet)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

- Applying the coef function to the glmnet object gives a matrix of regression coefficients, one column for each value of lambda and one row for each predictor (and intercept)

The glmnet package

We use the glmnet function in the glmnet package in order to perform Ridge Regression for a variety of values of the tuning parameter λ.

```
library(glmnet)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

- Applying the coef function to the glmnet object gives a matrix of regression coefficients, one column for each value of lambda and one row for each predictor (and intercept)
- The alpha argument in glmnet determines the type of penalty
  - alpha = 0 corresponds to Ridge Regression. alpha = 1 corresponds to LASSO

The glmnet package

We use the glmnet function in the glmnet package in order to perform Ridge Regression for a variety of values of the tuning parameter λ.

```
library(glmnet)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

- Applying the coef function to the glmnet object gives a matrix of regression coefficients, one column for each value of lambda and one row for each predictor (and intercept)
- The alpha argument in glmnet determines the type of penalty
  - alpha = 0 corresponds to Ridge Regression. alpha = 1 corresponds to LASSO
- By default, glmnet standardizes observations. To use unstandardized obs. add standardize = FALSE

The glmnet package

We use the glmnet function in the glmnet package in order to perform Ridge Regression for a variety of values of the tuning parameter λ.

```
library(glmnet)
ridge_mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

- Applying the coef function to the glmnet object gives a matrix of regression coefficients, one column for each value of lambda and one row for each predictor (and intercept)
- The alpha argument in glmnet determines the type of penalty
  - alpha = 0 corresponds to Ridge Regression. alpha = 1 corresponds to LASSO
- By default, glmnet standardizes observations. To use unstandardized obs. add standardize = FALSE
- Here, we gave a specific range of values for the tuning parameter. But if no lambda value is supplied, the function will automatically select a range.

Manipulating output of glmnet

```
coef(ridge_mod)[,1:4]
```

```
## 9 x 4 sparse Matrix of class "dgCMatrix"
##                         s0            s1            s2            s3
## (Intercept)   8.162200e-01  8.162200e-01  8.162200e-01  8.162200e-01
## NA_Sales      3.412480e-10  4.511098e-10  5.963407e-10  7.883273e-10
## EU_Sales      4.792437e-10  6.335319e-10  8.374919e-10  1.107115e-09
## JP_Sales      5.824358e-10  7.699458e-10  1.017823e-09  1.345502e-09
## Critic_Score  7.209905e-12  9.531071e-12  1.259951e-11  1.665582e-11
## User_Score    2.582969e-11  3.414533e-11  4.513812e-11  5.966994e-11
## RatingE10+   -5.247474e-11 -6.936852e-11 -9.170109e-11 -1.212235e-10
## RatingM       7.685481e-11  1.015975e-10  1.343060e-10  1.775446e-10
## RatingT      -6.427287e-11 -8.496495e-11 -1.123187e-10 -1.484787e-10
```

Manipulating output of glmnet

```
coef(ridge_mod)[,1:4]
```

```
## 9 x 4 sparse Matrix of class "dgCMatrix"
##                         s0            s1            s2            s3
## (Intercept)   8.162200e-01  8.162200e-01  8.162200e-01  8.162200e-01
## NA_Sales      3.412480e-10  4.511098e-10  5.963407e-10  7.883273e-10
## EU_Sales      4.792437e-10  6.335319e-10  8.374919e-10  1.107115e-09
## JP_Sales      5.824358e-10  7.699458e-10  1.017823e-09  1.345502e-09
## Critic_Score  7.209905e-12  9.531071e-12  1.259951e-11  1.665582e-11
## User_Score    2.582969e-11  3.414533e-11  4.513812e-11  5.966994e-11
## RatingE10+   -5.247474e-11 -6.936852e-11 -9.170109e-11 -1.212235e-10
## RatingM       7.685481e-11  1.015975e-10  1.343060e-10  1.775446e-10
## RatingT      -6.427287e-11 -8.496495e-11 -1.123187e-10 -1.484787e-10
```

```
ridge_mod$lambda[75]
```

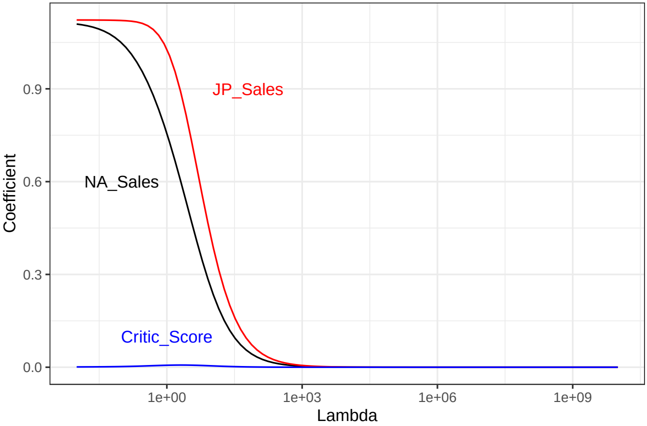
```
## [1] 10.72267
```

```
coef(ridge_mod)[,75]
```

```
##  (Intercept)     NA_Sales     EU_Sales     JP_Sales Critic_Score   User_Score
##  0.239041734  0.234144568  0.325084954  0.386330686  0.004219627  0.012114496
##   RatingE10+      RatingM      RatingT
## -0.030652286  0.042868156 -0.037871886
```

The effects of ridge regression

```
dd<-data.frame(b_NA_Sales<- coef(ridge_mod)[2,],
               b_JP_Sales<- coef(ridge_mod)[4,],
               b_Critic = coef(ridge_mod)[5,],
               grid)
```



Optimizing the tuning parameter

Which performs better, the model with $\lambda \approx 10000$ or $\lambda \approx 10$?

```
set.seed(11)
video_games_trn<-video_games %>% sample_frac(.75)
video_games_tst<-anti_join(video_games, video_games_trn)

x_trn<-model.matrix(Global_Sales ~. - Name, data = video_games_trn)[,-1]
y_trn<-video_games_trn$Global_Sales

x_tst<-model.matrix(Global_Sales ~. - Name, data = video_games_tst)[,-1]
y_tst<-video_games_tst$Global_Sales

ridge_mod_trn <- glmnet(x_trn, y_trn, alpha = 0, lambda = grid)

pred_10 <- predict(ridge_mod_trn, s = 75, newx = x_tst)
pred_10k <- predict(ridge_mod_trn, s = 50, newx = x_tst)
```

Optimizing the tuning parameter

Which performs better, the model with $\lambda \approx 10000$ or $\lambda \approx 10$?

```
set.seed(11)
video_games_trn<-video_games %>% sample_frac(.75)
video_games_tst<-anti_join(video_games, video_games_trn)

x_trn<-model.matrix(Global_Sales ~. - Name, data = video_games_trn)[,-1]
y_trn<-video_games_trn$Global_Sales

x_tst<-model.matrix(Global_Sales ~. - Name, data = video_games_tst)[,-1]
y_tst<-video_games_tst$Global_Sales

ridge_mod_trn <- glmnet(x_trn, y_trn, alpha = 0, lambda = grid)

pred_10 <- predict(ridge_mod_trn, s = 75, newx = x_tst)
pred_10k <- predict(ridge_mod_trn, s = 50, newx = x_tst)
```

```
MSE_10<-mean( (y_tst - pred_10)^2 )
MSE_10k<-mean( (y_tst - pred_10k)^2 )
data.frame(MSE_10, MSE_10k)
```

```
##     MSE_10  MSE_10k
## 1 1.364439 1.291877
```

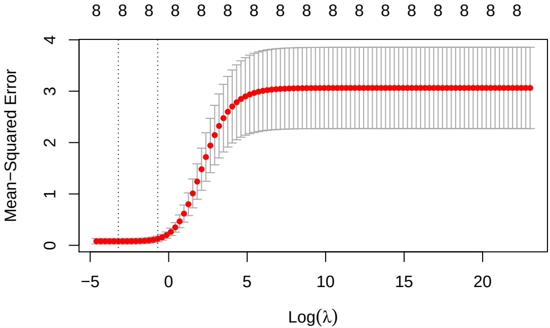
Optimizing the tuning parameter, cont'd

We can use the `cv.glmnet` function to perform cross-validation to compare MSE across all values of λ

Optimizing the tuning parameter, cont'd

We can use the `cv.glmnet` function to perform cross-validation to compare MSE across all values of λ

```
set.seed(1010)
my_cv<-cv.glmnet(x, y, alpha = 0, lambda = grid)
plot(my_cv)
```




```
best_L<-my_cv$lambda.min
best_L
```

```
## [1] 0.04037017
```