Ridge Regression for Better Usage

Kyoosik Kim
Jan 3 · 10 min read

The goal of this post is to let you better use ridge regression than just use what libraries provide. Then, *"What is Ridge Regression?"*. The simplest way to answer the question is "*Variation of Linear Regression*". The worst way is to start with the following mathematical equations not many can understand at first glance.

$$\hat{\beta}^{ridge} = \underset{\beta \in \mathbb{R}}{argmin} \|y - XB\|_2^2 + \lambda \|B\|_2^2$$

Bad news is that we still have to deal with it and good news is that we will not start with equations like that, albeit just not now. What I would like to begin with is 'Ordinary Least Squares (OLS)'. If you happened to have little or no background about linear regression, this video will help you get the sense of how it works using 'Least Square Method'. Now, you know that OLS is just like what we generally call 'Linear Regression', and I will use the term as such.

Before Moving On

In the next sections, I will take different approaches with various terms and figures. There are two things you would want to remember. One is that we don't like overfitting. In other words, **we always prefer a model that catches general patterns**. The other is that our goal is predicting it from new data, not specific data. Therefore, **model evaluation should be based on new data (testing set), not given data (training set)**. Also, I will use the following terms interchangeably.

- Independent Variable = Feature = Attribute = Predictor = X
- Coefficient = Beta = β
- Residual Sum of Squares = RSS

Why And Why Not OLS

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon \quad \longleftarrow \quad \textit{True Model}$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p \quad \longleftarrow \quad \textit{Estimated Model}$$

$$\hat{\beta}^{OLS} = (X^T X)^{-1} X^T Y \quad \longleftarrow \quad \textit{Matrix Equation}$$

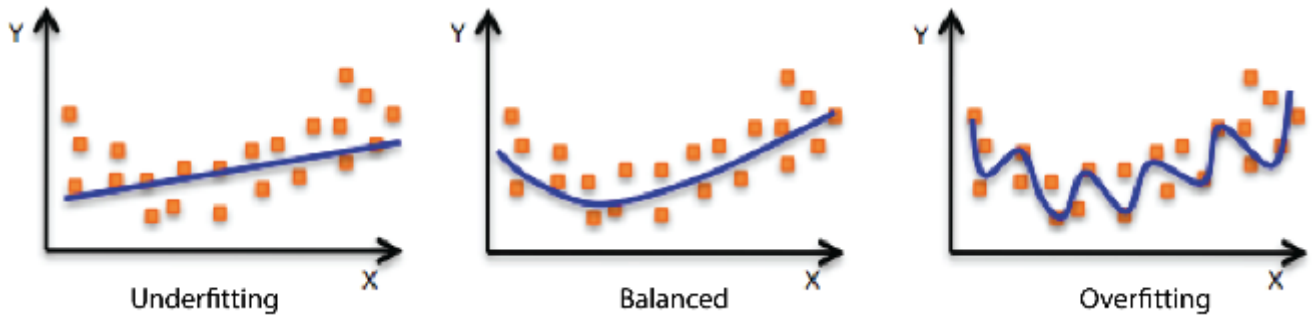
Least Square Method finds the Best and Unbiased Coefficients

You may know that least square method finds the coefficients that best fit the data. One more condition to be added is that it also finds the unbiased coefficients. Here unbiased means that **OLS doesn't consider which independent variable is more important than others**. It simply finds the coefficients for a given data set. In short, there is only one set of betas to be found, resulting in the lowest 'Residual Sum of Squares (RSS)'. The question then becomes *"Is a model with the lowest RSS truly the best model?"*.

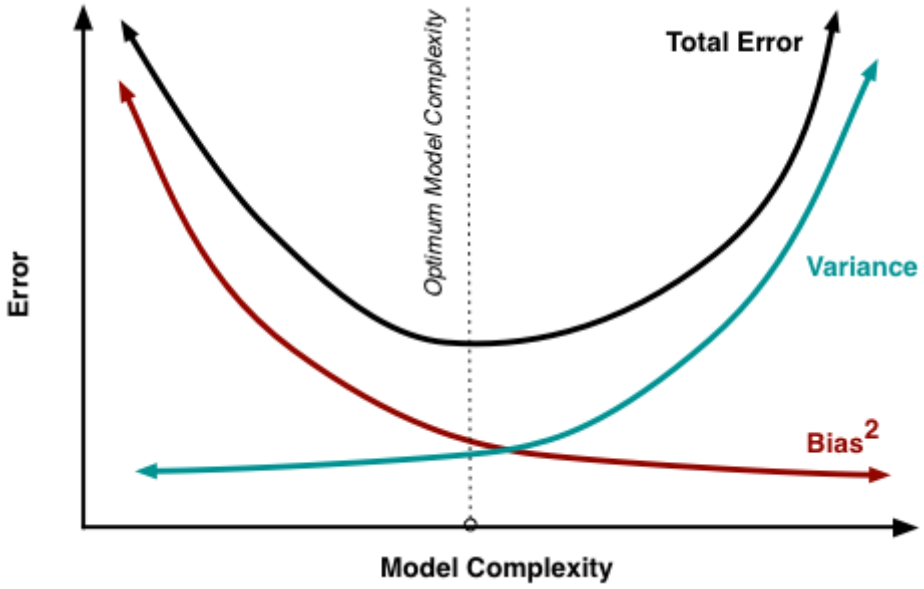
Bias vs. Variance

The answer for the question above is *"Not really"*. As hinted in the word 'Unbiased', we need to consider 'Bias' too. Bias means how equally a model cares about its predictors. Let's say there are two models to predict an apple price with two predictor 'sweetness' and 'shine'; one model is unbiased and the other is biased.

First, the unbiased model tries to find the relationship between the two features and the prices, just as the OLS method does. This model will fit the observations as perfectly as possible to minimize the RSS. However, this could easily lead to overfitting issues. In other words, the model will not perform as well with new data because it is built for the given data so specifically that it may not fit new data.



The biased model accepts its variables unequally to treat each predictor differently. Going back to the example, we would want to only care about 'sweetness' to build a model and this should perform better with new data. The reason will be explained after understanding **Bias vs. Variance**. If you're not familiar with the bias vs. variance topic, I strongly recommend you to watch this video that will give you insight.



It can be said that **bias is related with a model failing to fit the training set and variance is related with a model failing to fit the testing set**. Bias and variance are in a trade-off relationship over model complexity, which means that a simple model would have high-bias and low-variance, and vice versa. In our apple example, a model only considering 'sweetness' would not fit the training data as much as the other model considering both 'sweetness' and 'shine', but the simpler model will be better at predicting new data.

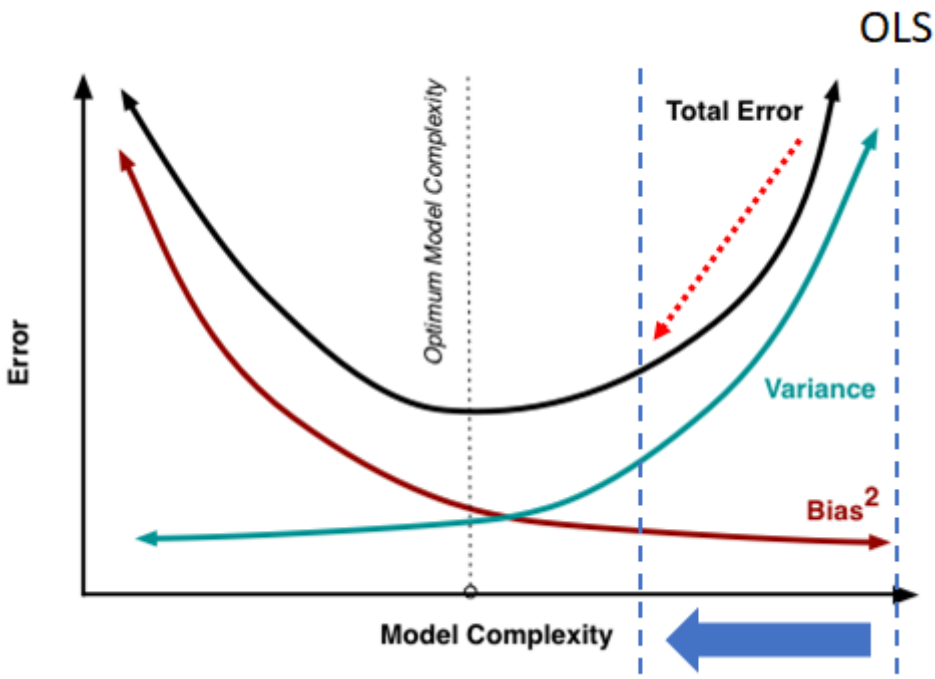
This is because 'sweetness' is a determinant of a price while 'shine' should not by common sense. We all know this as a human but mathematical models do not think like us and just calculate what's given until it finds some relationship between all the predictors and the independent variable to fit training data.

**Note*: We assume that 'sweetness' and 'shine' are not correlated

Where Ridge Regression Comes Into Play

Looking at *Bias vs. Variance* figure, the Y-axis is 'Error' which is the 'Sum of Bias and Variance'. Since both of them are basically related with failing, we would like to minimize those. Now taking a second look at the figure closely, you will find that the

spot the total error is lowest is somewhere in the middle. This is often times called 'Sweet Spot'.



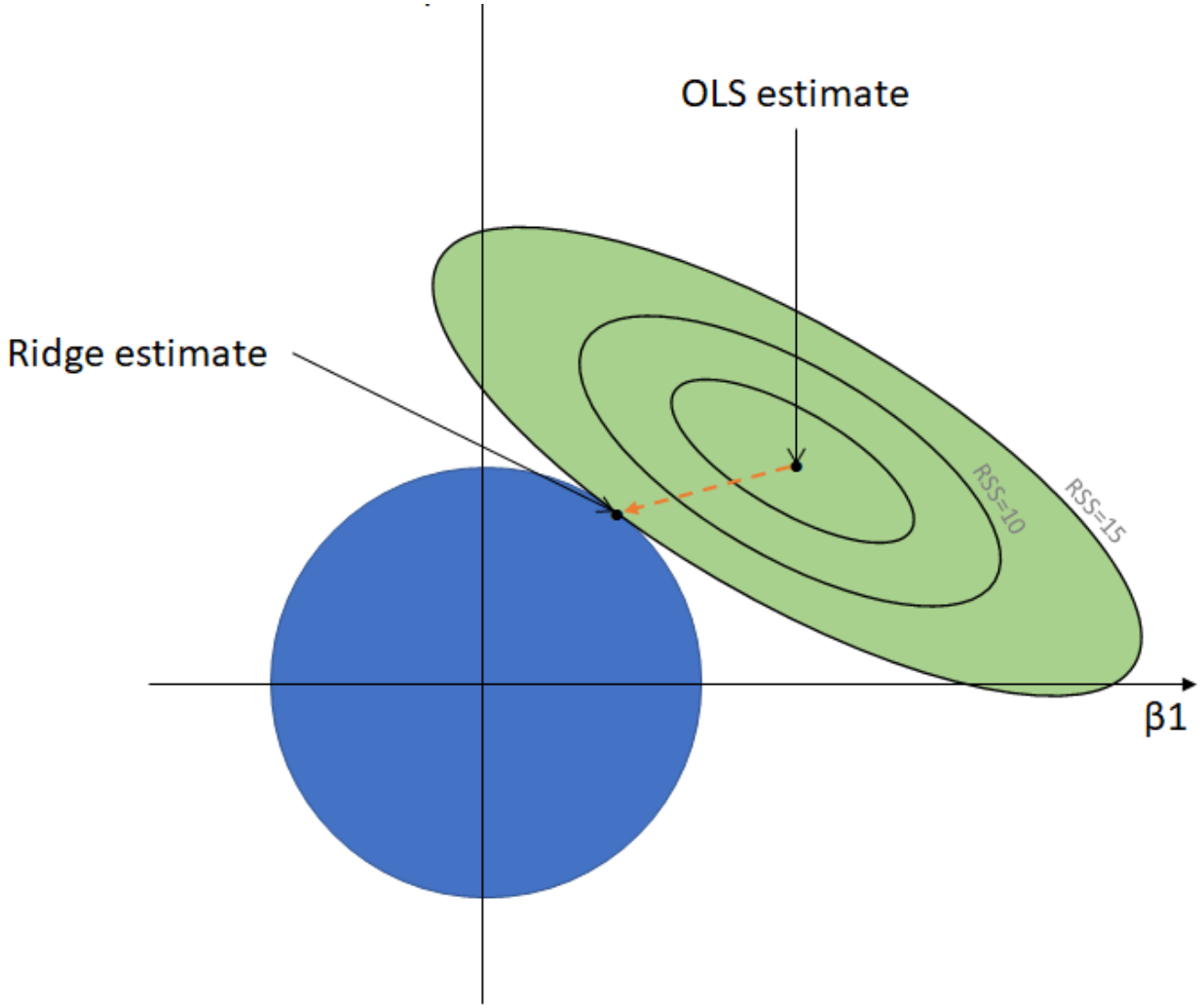
Let's recall that OLS treats all the variables equally (unbiased). Therefore, an OLS model becomes more complex as new variables are added. It can be said that an OLS model is always on the rightest of the figure, having the lowest bias and the highest variance. It is fixed there, never moves, but we want to move it to the sweet spot. This is when ridge regression would shine, also referred to as *Regularization*. **In ridge regression, you can tune the lambda parameter so that model coefficients change**. This can be best understood with a programming demo that will be introduced at the end.

Geometric Understanding of Ridge Regression

Many times, a graphic helps to get the feeling of how a model works, and ridge regression is not an exception. The following figure is the geometric interpretation to compare OLS and ridge regression.

Contours and OLS Estimate

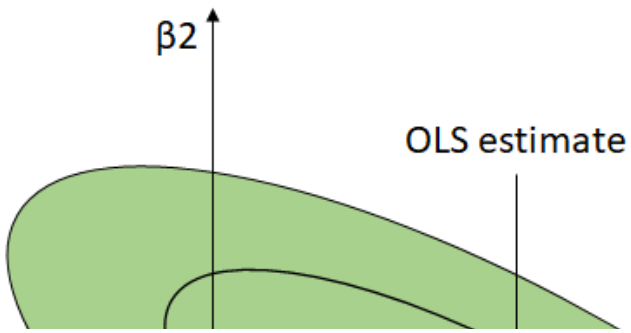
Each contour is a connection of spots where the RSS is the same, centered with the OLS estimate where the RSS is the lowest. Also, the OLS estimate is the point where it best fits the training set (low-bias).

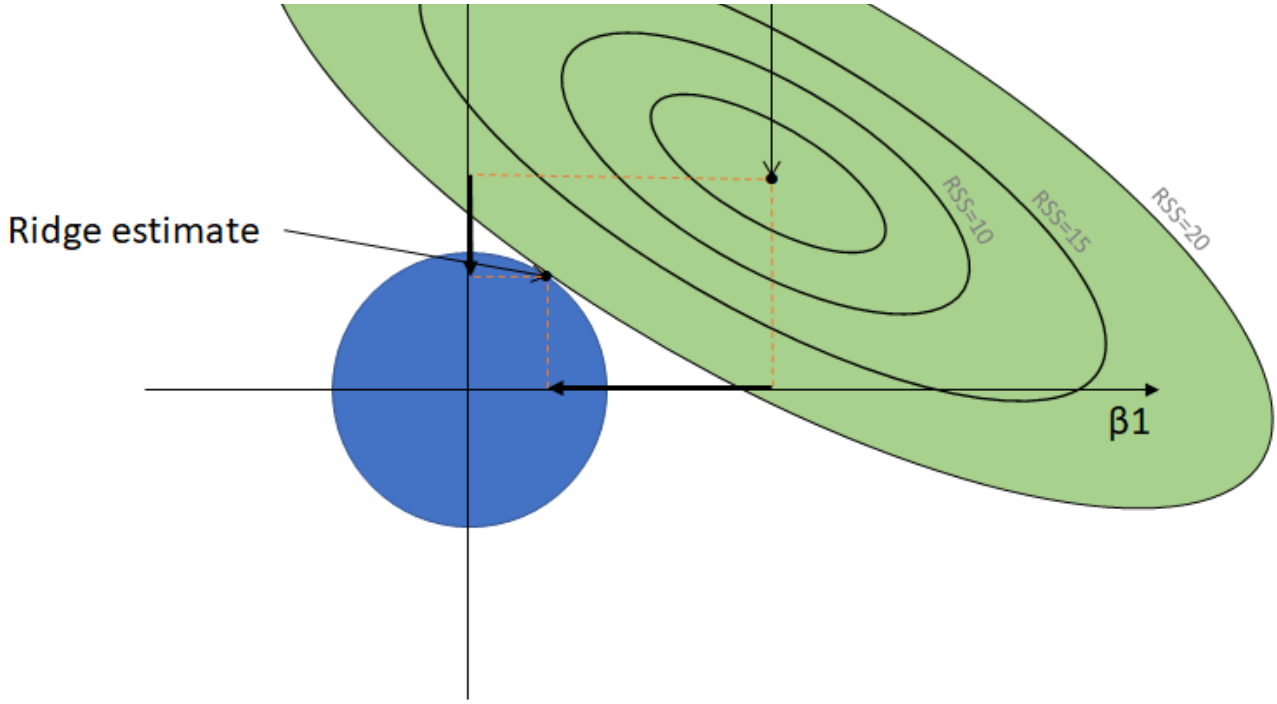


Circle and Ridge Estimate

Unlike the OLS estimate, the ridge estimate changes as the size of the blue circle changes. It is simply where the circle meets the most outer contour. How ridge regression works is how we tune the size of the circle. The key point is that **'s change at a different level**.

Let's say $\beta 1$ is 'shine' and $\beta 2$ is 'sweetness'. As you can see, ridge $\beta 1$ relatively drops more quickly to zero than ridge $\beta 2$ does as the circle size changes (compare the two figures). The reason why this happens is because the β's change differently by the RSS. More intuitively, the contours are not circles but ellipses positioned tilted.





Ridge β's can never be zero but only *converge* to it, and this will be explained in the next with the mathematical formula. Although a geometric expression like this explains a main idea pretty well, there is a limitation too that we can't express it over 3-dimension. So, it all comes down to mathematical expressions.

Mathematical Formula

We've seen the equation of multiple linear regression both in general terms and matrix version. It can be written in another version as follows.

$$\underset{\beta \in \mathbb{R}}{argmin} \sum [y_i - \hat{y}_i] = \underset{\beta \in \mathbb{R}}{argmin} \sum [y_i - (\beta_0 + \beta_1 x_1 + \beta x_2 + \cdots + \beta x_p)]^2$$

Here *argmin* means 'Argument of Minimum' that makes the function attain the minimum. In the context, it finds the β's that minimize the RSS. And we know how to get the β's from the matrix formula. Now, the question becomes "What does this have to do with ridge regression?".

$$\beta_0^2 + \beta_1^2 + \cdots + \beta_p^2 \leq C^2$$

Again, ridge regression is a variant of linear regression. The term above is the ridge constraint to the OLS equation. We are looking for the β's but they now must meet the above constraint too. Going back to the geometric figure, the C is equivalent to the radius of the circle, thus, the β's should fall in the circle area, probably somewhere on the edge.

Vector Norm

We still want to understand the very first equation. To do so, we need to brush up on vector norm, which is nothing but the following definition.

$$\|B\|_2 = \sqrt{\beta_0^2 + \beta_1^2 + \cdots + \beta_p^2}$$

The subscription 2 is as in 'L2 norm', and you can learn more about vector norms here. We only care about L2 norm at this moment, so we can construct the equation we've already seen. The following is the simplest but still telling the same as what we have been discussing. Notice the first term in the following equation is basically OLS, and then the second term with lambda is what makes ridge regression.

$$\hat{\beta}^{ridge} = \underset{\beta \in \mathbb{R}}{argmin} \|y - XB\|_2^2 + \lambda \|B\|_2^2$$

What We Really Want to Find

The term with lambda is often called 'Penalty' since it increases RSS. We iterate certain values onto the lambda and evaluate the model with a measurement such as 'Mean Square Error (MSE)'. So, the lambda value that minimizes MSE should be selected as the final model. This **ridge regression model is generally better than the OLS model in prediction**. As seen in the formula below, ridge β's change with lambda and becomes the same as OLS β's if lambda is equal to zero (no penalty).

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

Why It Converges to Zero But Not Becomes Zero

Deploying the matrix formula we saw previously, the lambda ends up in denominator. It means that if we increase the lambda value, ridge β's should decrease. But ridge β's can't be zeros no matter how big the lambda value is set. That is, ridge regression gives different importance weights to the features but does not drop unimportant features.

$$(\cdots + \frac{1}{\lambda} + \cdots)X^T Y$$

Demo with Dataset

The dataset 'Boston House Price' from *sklearn* library is used for demonstration. There are more than a dozen features which are explained on this metadata. The following *python* libraries are needed throughout the demonstration.

**Full code* can be found at my github

```
# dataframe and plot
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

# dataset
from sklearn.datasets import load_boston
# scaling and dataset split
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
# OLS, Ridge
from sklearn.linear_model import LinearRegression, Ridge
# model evaluation
from sklearn.metrics import r2_score, mean_squared_error
```

ridge_libraries.py hosted with ♡ by GitHub view raw

Now the dataset is loaded, subsequently, the features should be standardized. Since ridge regression shrinks coefficients by penalizing, the features should be scaled for start condition to be fair. This post explains some more details about this issue.

```
# load dataset
house_price = load_boston()
df = pd.DataFrame(house_price.data, columns=house_price.feature_names)
df['PRICE'] = house_price.target
```

```
# standardize and train/test split
house_price.data = preprocessing.scale(house_price.data)
X_train, X_test, y_train, y_test = train_test_split(
    house_price.data, house_price.target, test_size=0.3, random_state=10)
```

house_price_load.py hosted with ♡ by GitHub view raw

Next, we can iterate the lambda values ranged from 0 to 199. Note that the coefficients at lambda equal to zero ($x = 0$) are the same with the OLS coefficients.

```
# initialize
ridge_reg = Ridge(alpha=0)
ridge_reg.fit(X_train, y_train)
ridge_df = pd.DataFrame({'variable': house_price.feature_names, 'estimate': ridge_reg.coef_})
ridge_train_pred = []
ridge_test_pred = []

# iterate lambdas
for alpha in np.arange(0, 200, 1):
    # training
    ridge_reg = Ridge(alpha=alpha)
    ridge_reg.fit(X_train, y_train)
    var_name = 'estimate' + str(alpha)
    ridge_df[var_name] = ridge_reg.coef_
    # prediction
    ridge_train_pred.append(ridge_reg.predict(X_train))
    ridge_test_pred.append(ridge_reg.predict(X_test))

# organize dataframe
ridge_df = ridge_df.set_index('variable').T.rename_axis('estimate').rename_axis(None, 1).reset_
```

iterate_lambda.py hosted with ♡ by GitHub view raw

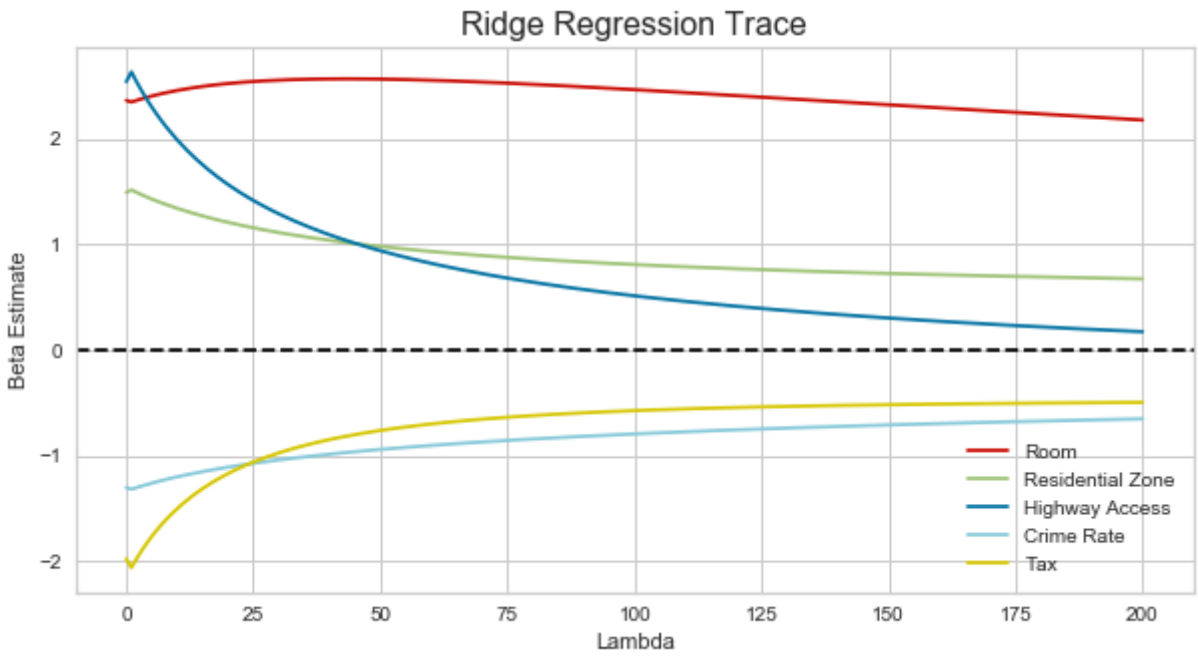
Now, we can draw plot from the data frame. Only five attributes are selected for better visualization.

```
# plot betas by lambda
fig, ax = plt.subplots(figsize=(10, 5))
ax.plot(ridge_df.RM, 'r', ridge_df.ZN, 'g', ridge_df.RAD, 'b', ridge_df.CRIM, 'c', ridge_df.TAX,
ax.axhline(y=0, color='black', linestyle='--')
ax.set_xlabel("Lambda")
ax.set_ylabel("Beta Estimate")
ax.set_title("Ridge Regression Trace", fontsize=16)
ax.legend(labels=['Room','Residential Zone','Highway Access','Crime Rate','Tax'])
```

```
ax.grid(True)
```

plot_beta.py hosted with ♡ by GitHub view raw

'Room' should be the best indicator for house price by intuition. This is why the line in red does not quite shrink over iteration. On the contrary, 'Highway Access' (blue) decreases remarkably, which means the feature loses its importance as we seek more general models.



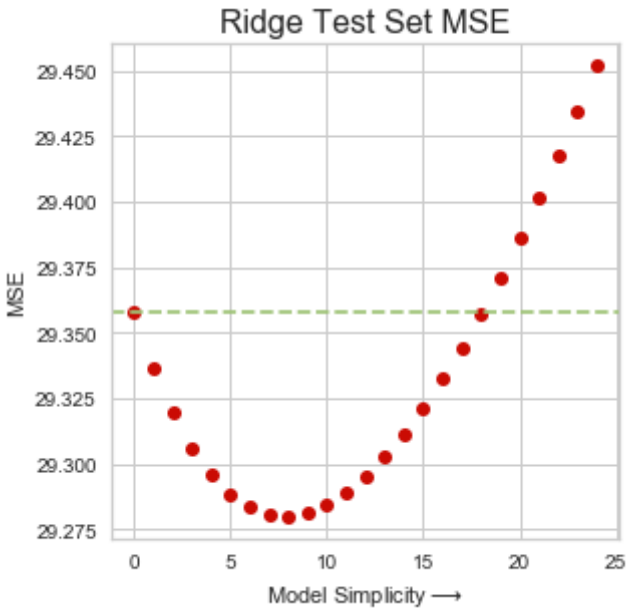
The similar patterns are seen from the rest converging to zero, the black dotted line. If we increase the lambda more and more (extremely biased), then only 'Room' would stay significant, which makes sense again because the number of rooms must explain the most.

```
# MSE of Ridge and OLS
ridge_mse_test = [mean_squared_error(y_test, p) for p in ridge_test_pred]
ols_mse = mean_squared_error(y_test, ols_pred)

# plot mse
plt.plot(ridge_mse_test[:25], 'ro')
plt.axhline(y=ols_mse, color='g', linestyle='--')
plt.title("Ridge Test Set MSE", fontsize=16)
plt.xlabel("Model Simplicity$\longrightarrow$")
plt.ylabel("MSE")
```

plot_mse.py hosted with ♡ by GitHub view raw

The code snippet above draws MSE traced by lambda. Since the model becomes simpler (=biased) as a bigger value is set to lambda, the X-axis represents model simplicity from left to right.



The green dotted line is from OLS on the graph above with the X-axis being drawn by increasing lambda values. The MSE values decreases in the beginning as the lambda value increases, which means the model prediction is improved (less error) to a certain point. In short, **an OLS model with some bias is better at prediction than the pure OLS model**, we call this modified OLS model as the ridge regression model.

Conclusion

We looked into ridge regression at different angles from mathematical formula, matrix format, to geometric expression. Through these, we could understand that ridge regression is basically a linear regression with penalty. Through the demonstration, we confirmed that **there is no equation for finding the best lambda**. Thus, we needed to iterate a series of values and evaluate prediction performances with MSE. By doing so, we found that the ridge regression model performs better than the plain linear regression model for prediction.

- OLS simply finds the best fit for given data
- Features have different contributions to RSS
- Ridge regression gives a bias to important features
- MSE or R-square can be used to find the best lambda

Good Reads

Welcome to STAT 501! | STAT 501

This is the STAT 501 online course materials website. There are lots of examples, notes, and...

onlinecourses.science.psu.edu

sklearn.linear_model.Ridge — scikit-learn 0.20.0 documentation

This model solves a regression model where the loss function is the linear least squares function...

scikit-learn.org

IPython Cookbook - 8.1. Getting started with scikit-learn

IPython Cookbook,

ipython-books.github.io

Machine Learning Ridge Regression Linear Regression Bias Vs Variance Python

About Help Legal