Ridge Regression

Presented By :
Aditya Sant
MSc Sem-3
11/1/2021

Multiple Linear Regression

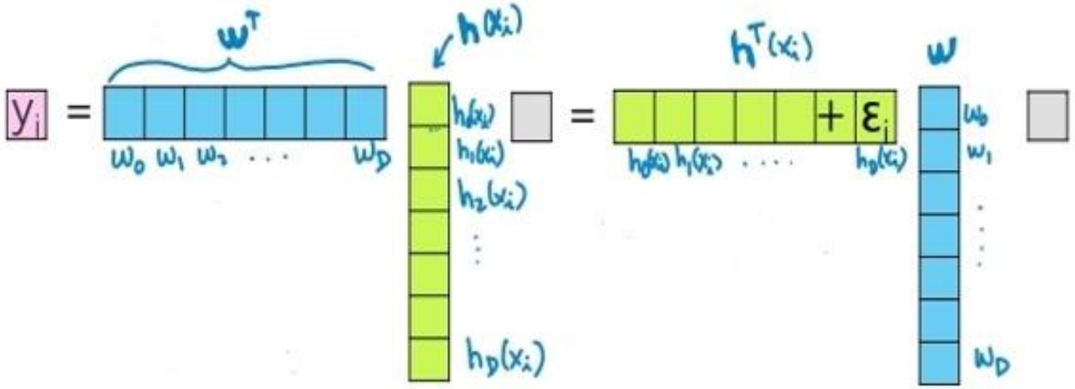
Model:

$$y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \dots + w_D h_D(x_i) + \varepsilon_i$$

$$= \sum_{j=0}^{D} w_j h_j(x_i) + \varepsilon_i$$

$$\mathrm{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

$$= (\mathbf{y} - \mathbf{H}\mathbf{w})^T (\mathbf{y} - \mathbf{H}\mathbf{w})$$



$$y_i = w_0 + w_1 x_i + \varepsilon_i$$

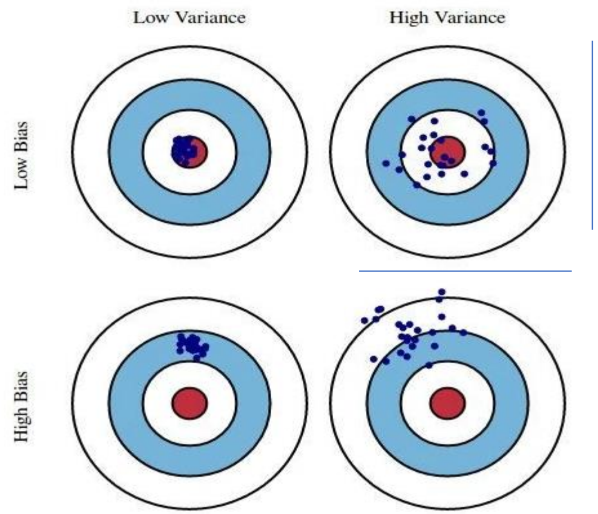
$$\boxed{\mathrm{RSS}(w_0, w_1) = \sum_{i=1}^{N} (y_i - [w_0 + w_1 x_i])^2}$$

If data are first centered about 0, then favoring small intercept not so worrisome

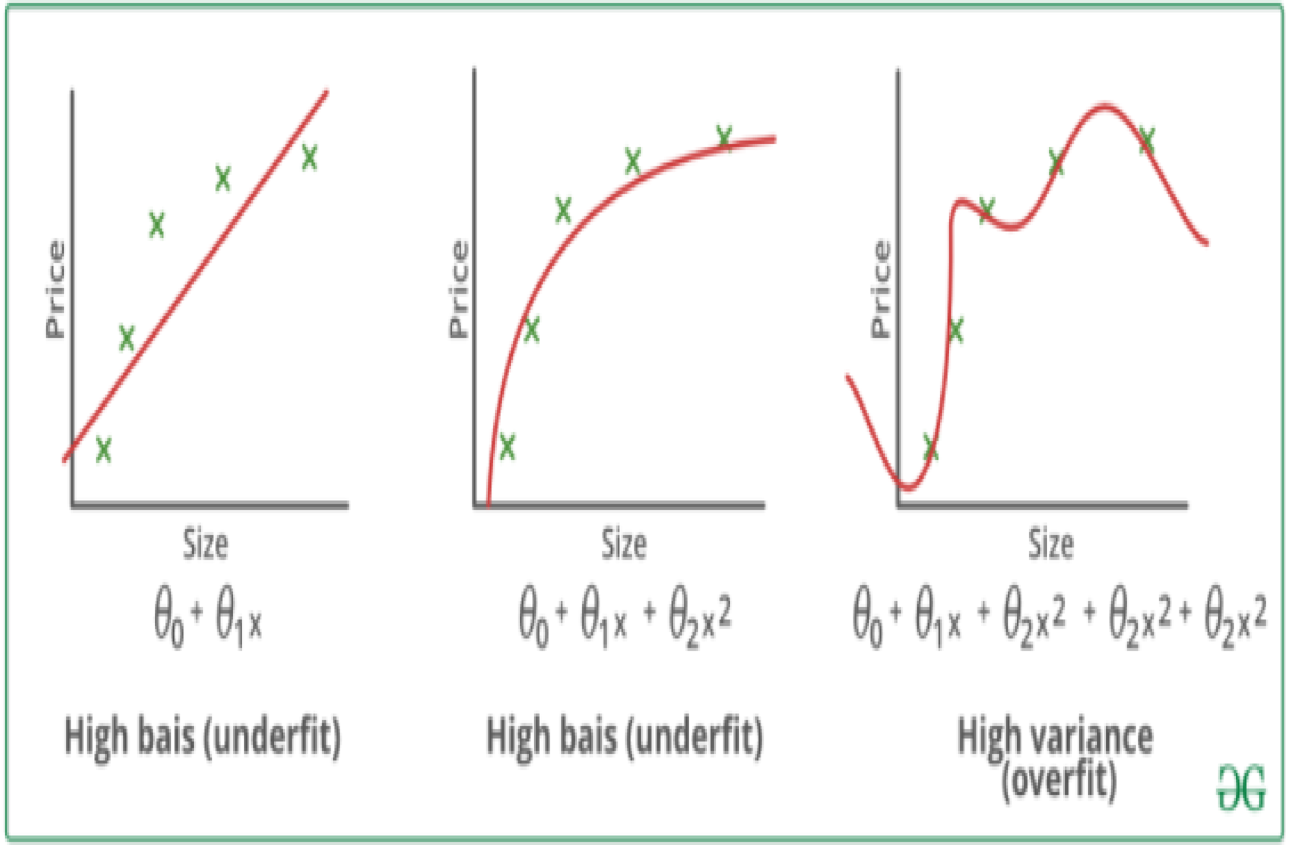
Step 1: Transform y to have 0 mean

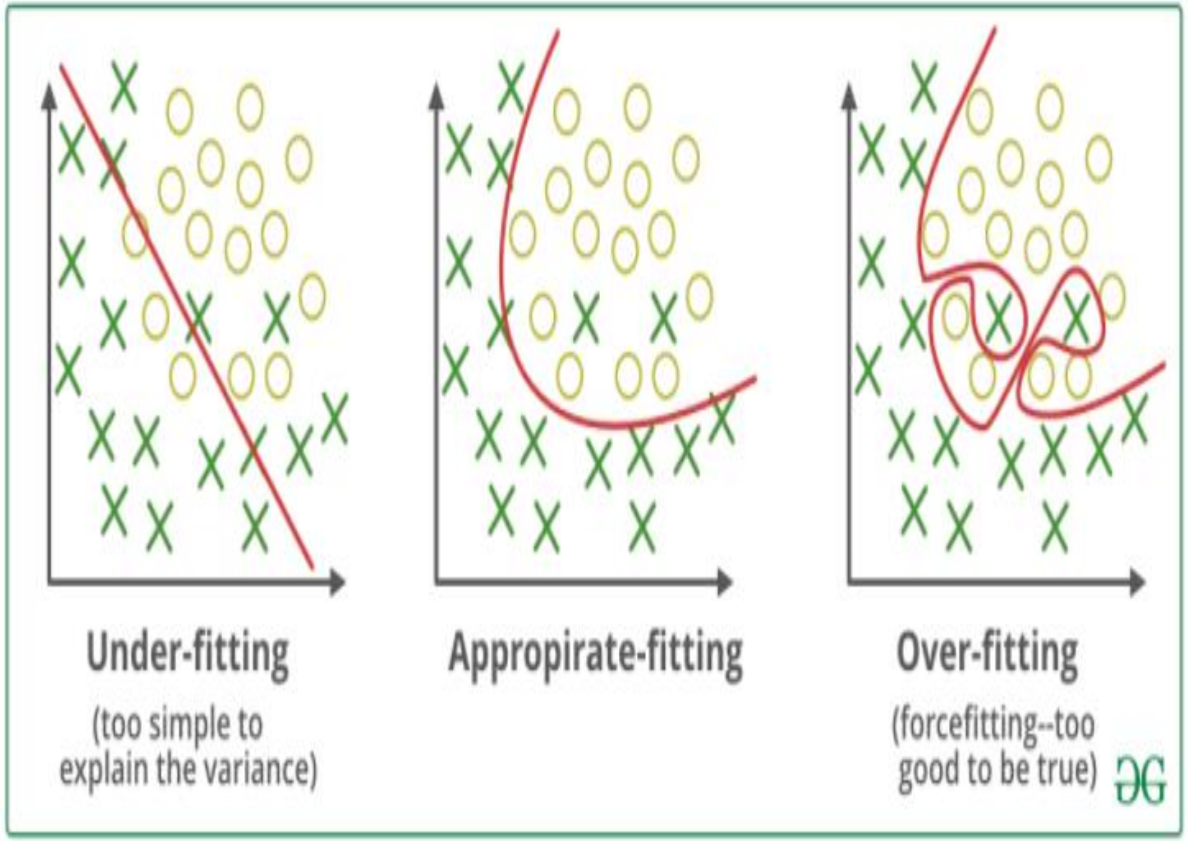
Step 2: Run ridge regression as normal

What is overfitting ?



Overfitting – High Variance and Low Bias





Main Idea Behind Ridge Regression

Adding term to cost-of-fit
to prefer small coefficients

Desired total cost format

Want to balance:

i. How well function fits data

ii. Magnitude of coefficients

Total cost =

$$\underbrace{\text{measure of fit}}_{\text{RSS}(\mathbf{w})} + \underbrace{\text{measure of magnitude of coefficients}}_{\|\mathbf{w}\|_2^2}$$

small # = good fit to training data

small # = not overfit

What if $\hat{\mathbf{w}}$ selected to minimize

$$\mathrm{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

tuning parameter = balance of fit and magnitude

Ridge regression
(a.k.a L_2 regularization)

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2$$

Cost function for simple linear model

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

Cost function for ridge regression

$$\|\boldsymbol{x}\| = \sqrt{\sum_{k=1}^{n} x_k^2}.$$

What do we achieve by Ridge Regression ?

- The penalty term (lambda) regularizes the coefficients such that if the coefficients take large values the optimization function is penalized.

- **Ridge Regression shrinks the coefficients and it helps to :**

1. **Reduce the model complexity**
2. **Reduce the effect of multi-collinearity (independent variables are correlated with each other)**

Bias-variance tradeoff

Large λ:

high bias, low variance

(e.g., $\hat{\mathbf{w}}$ =0 for $\lambda=\infty$)

In essence, λ controls model complexity

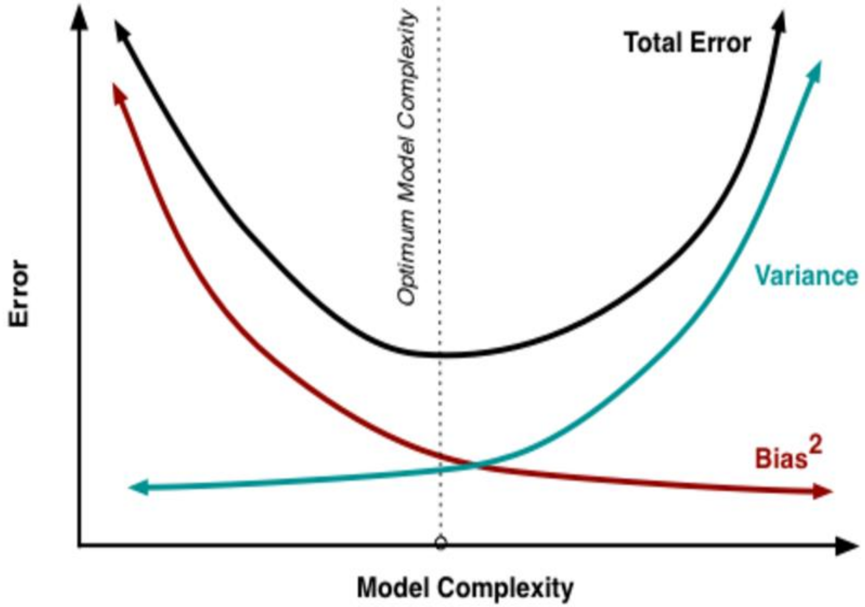
Small λ:

low bias, high variance

(e.g., standard least squares (RSS) fit of high-order polynomial for $\lambda=0$)

$$RSS(\mathbf{w}) + \lambda ||\mathbf{w}||_2^2$$

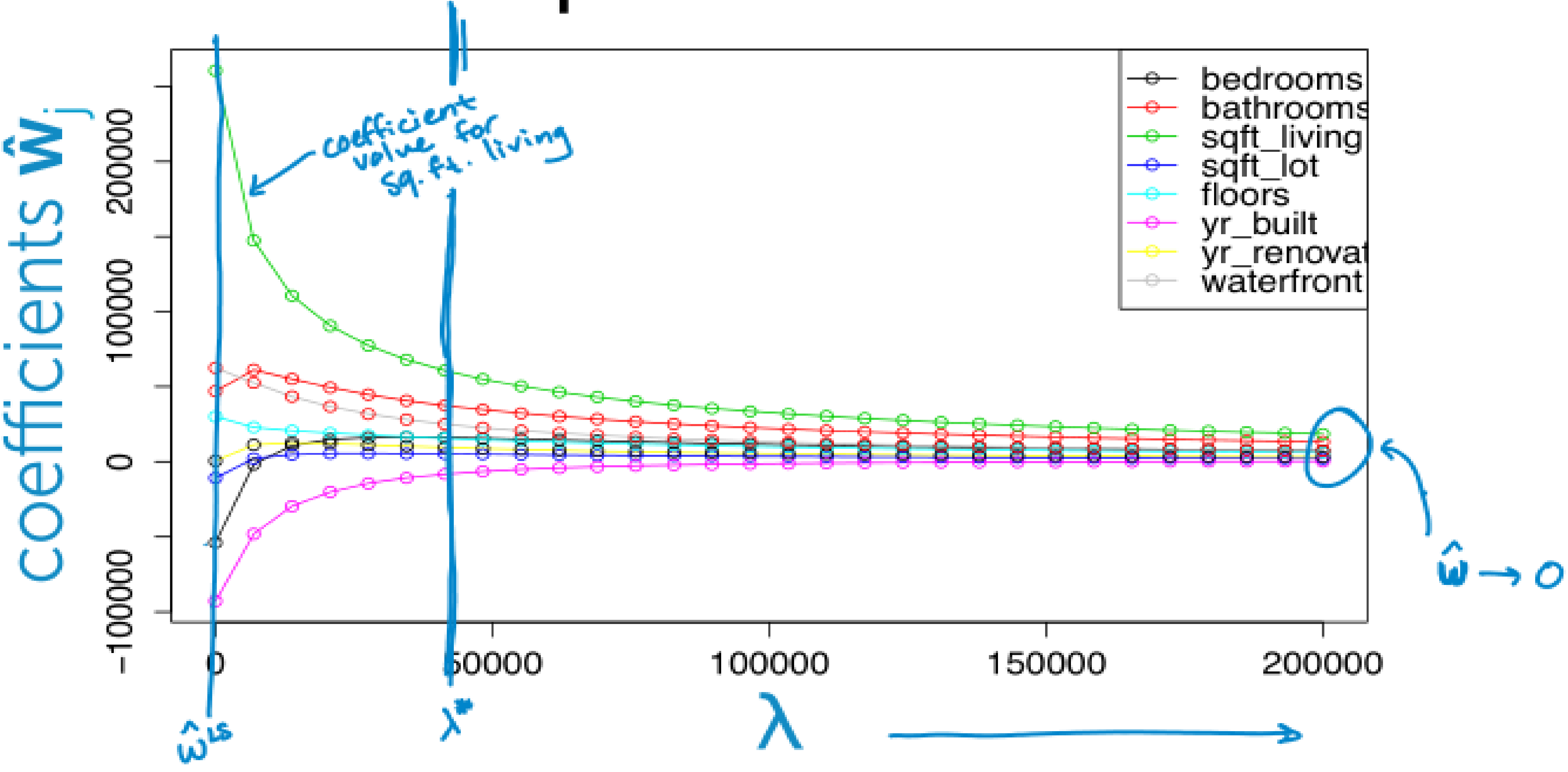
tuning parameter = balance of fit and magnitude



$$RSS(\mathbf{w}) + \lambda ||\mathbf{w}||_2^2$$

tuning parameter = balance of fit and magnitude

Coefficient path



1.
TO FIND THE ESTIMATE OF w

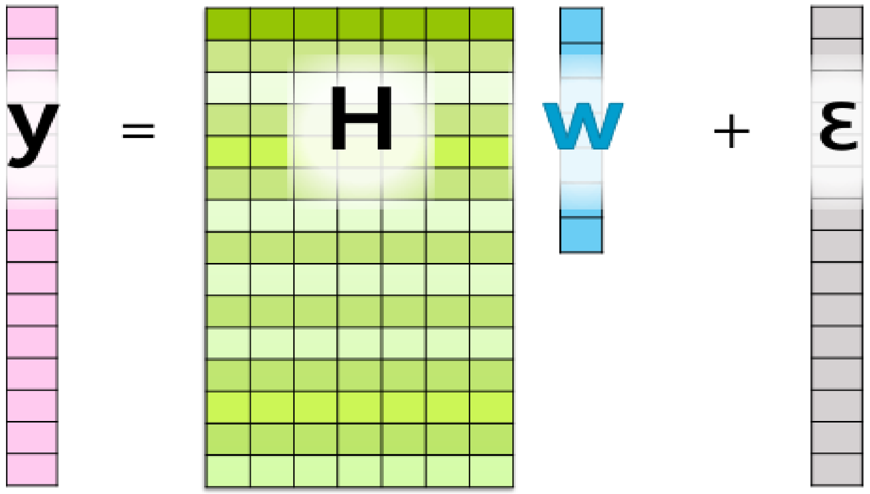
$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}\left(y_i - \sum_{j=0}^{p} w_j \times x_{ij}\right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

Cost function for ridge regression

Ridge regression

Recall matrix form of RSS

Model for all N observations together



$$\mathrm{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T\mathbf{w})^2$$

$$= (\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$

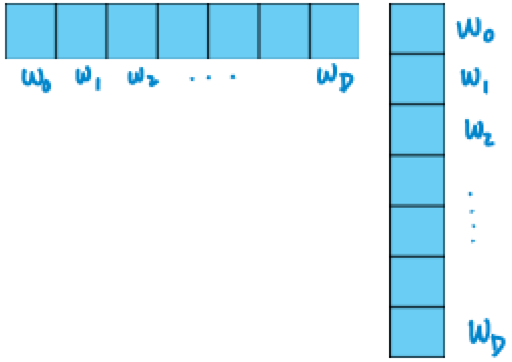
In matrix form, ridge regression cost is:

$$\mathrm{RSS}(\mathbf{w}) + \lambda\|\mathbf{w}\|_2^2$$

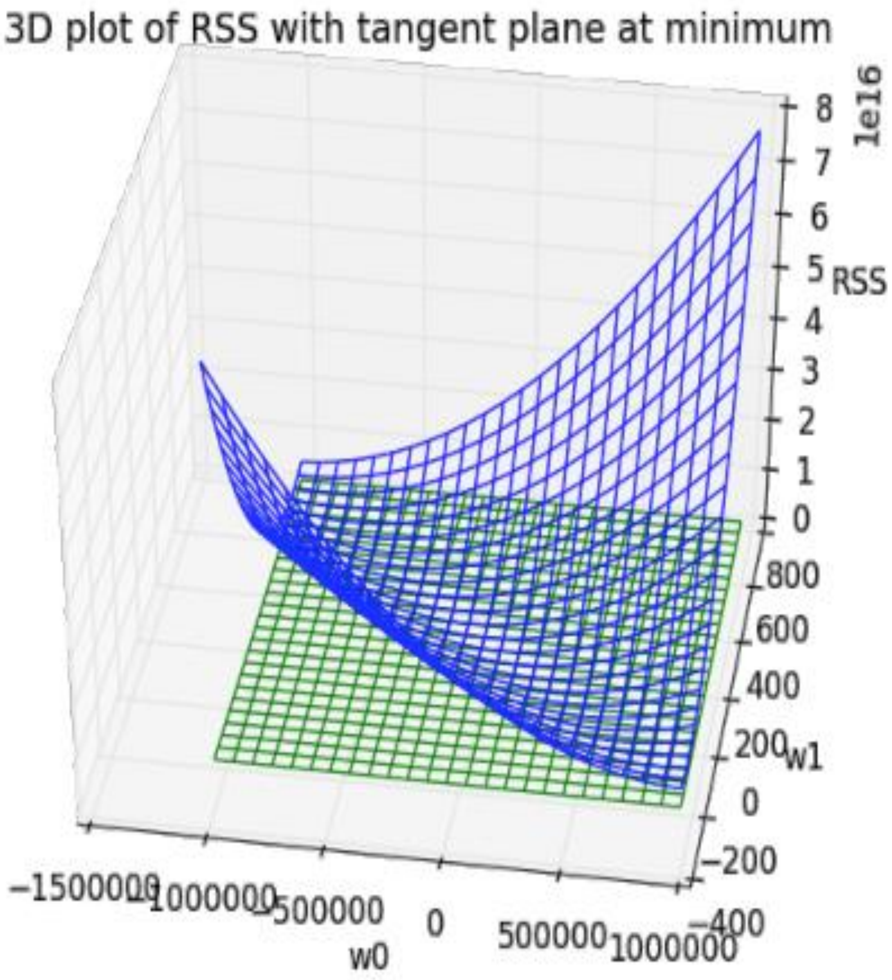
$$= (\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}$$

$$\|\mathbf{w}\|_2^2 = w_0^2 + w_1^2 + w_2^2 + \ldots + w_D^2$$

=



$= w^T w$



Scalar-valued multivariable function

$$\nabla f(\underbrace{x_0, y_0, \dots}_{\nabla f \text{ takes the same type of inputs as } f}) = \underbrace{\begin{bmatrix} \frac{\partial f}{\partial x}(x_0, y_0, \dots) \\ \frac{\partial f}{\partial y}(x_0, y_0, \dots) \\ \vdots \end{bmatrix}}_{\nabla f \text{ outputs a vector with all possible partial derivatives of } f.}$$

Notation for gradient, called “nabla”.

Here we want to minimize the Cost function so we will put the Gradient = 0

Gradient of ridge regression cost

$$\nabla \text{cost}(\mathbf{w}) = \nabla[\text{RSS}(\mathbf{w}) + \lambda||\mathbf{w}||_2^2] = \nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}]$$

$$= \underbrace{\nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})]}_{-2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})} + \lambda\underbrace{\nabla[\mathbf{w}^T\mathbf{w}]}_{2\mathbf{w}}$$

$$\nabla \text{cost}(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + 2\lambda\mathbf{w}$$
$$= -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + 2\lambda\mathbf{I}\mathbf{w}$$

$$\nabla \text{cost}(\mathbf{w}) = -\cancel{2}\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + \cancel{2}\lambda\mathbf{I}\mathbf{w} = 0$$

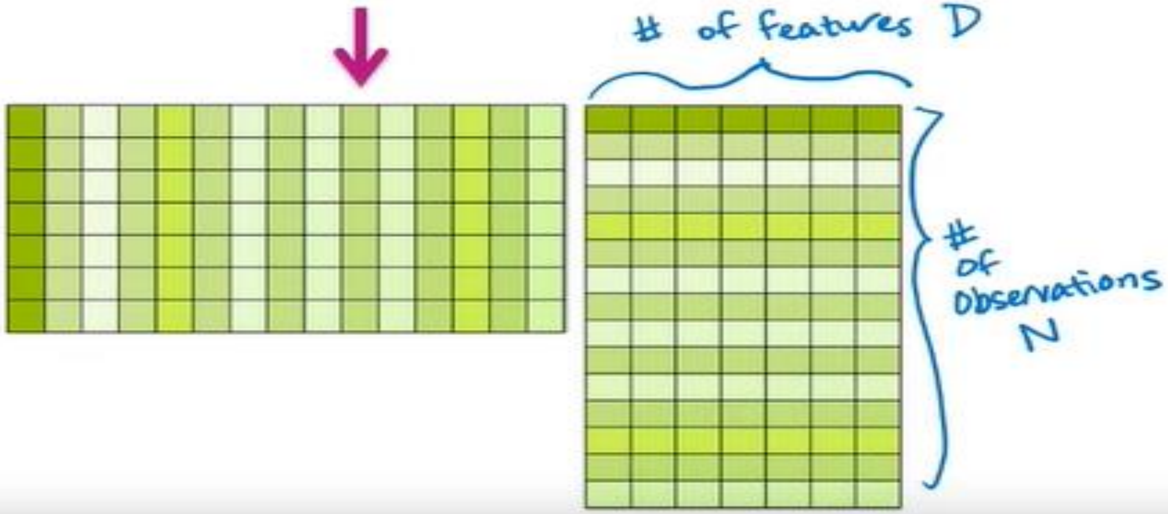
$$-H^T y + H^T H \hat{w} + \lambda I \hat{w} = 0$$

$$H^T H \underline{\hat{w}} + \lambda I \underline{\hat{w}} = H^T y$$

$$(H^T H + \lambda I)\hat{w} = H^T y$$

$$\hat{w}^{ridge} = (H^T H + \lambda I)^{-1} H^T y$$

$$\hat{\mathbf{w}}^{LS} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$



2.
TO FIND LAMBDA

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

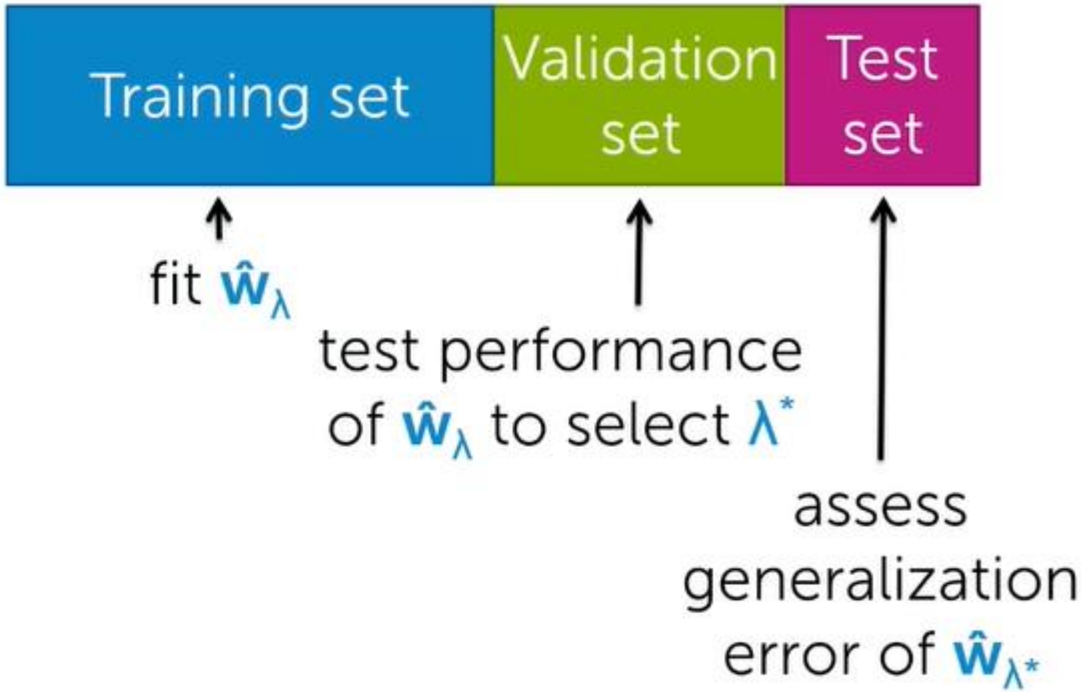
Cost function for ridge regression

$$RSS(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

tuning parameter = balance of fit and magnitude

How to choose λ

If sufficient amount of data…



Choosing the validation set

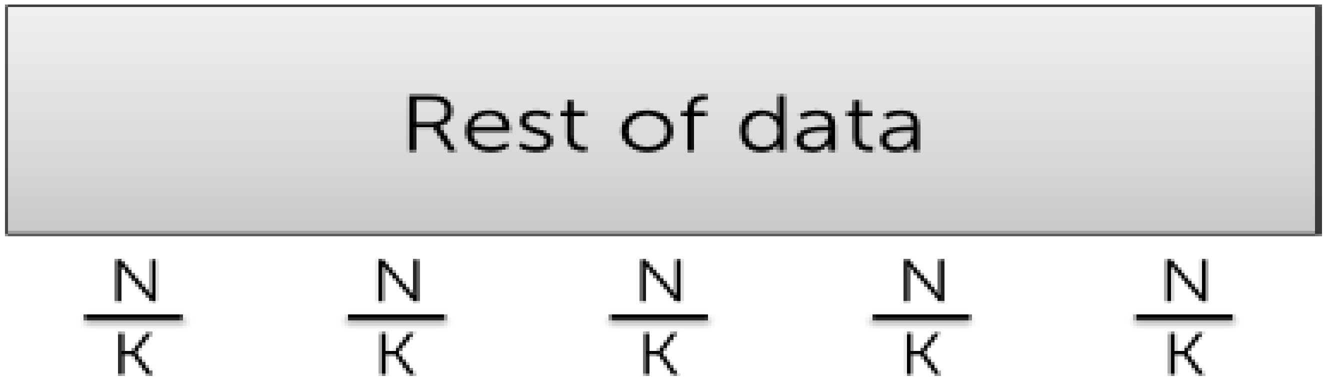
small validation set

Which subset should I use?

ALL!

average performance over all choices

K-fold cross validation

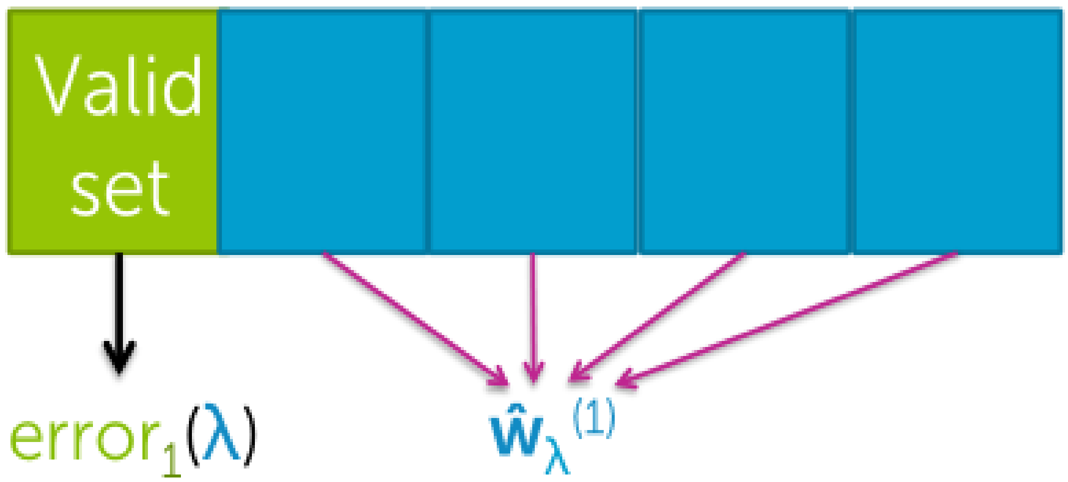


Preprocessing:

Randomly assign data to K groups

(use same split of data for all other steps)

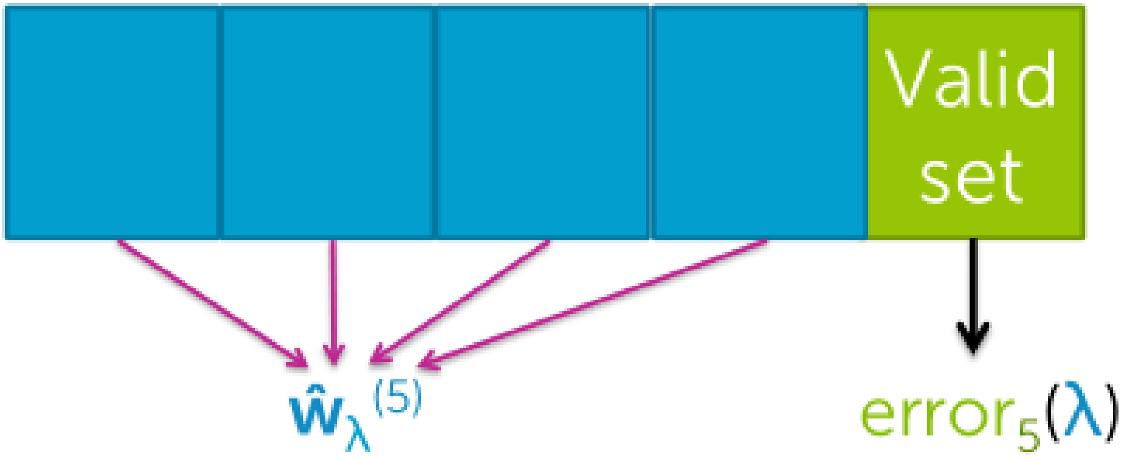
K-fold cross validation



For k=1,…,K

1. Estimate $\hat{\mathbf{w}}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $error_k(\lambda)$

K-fold cross validation

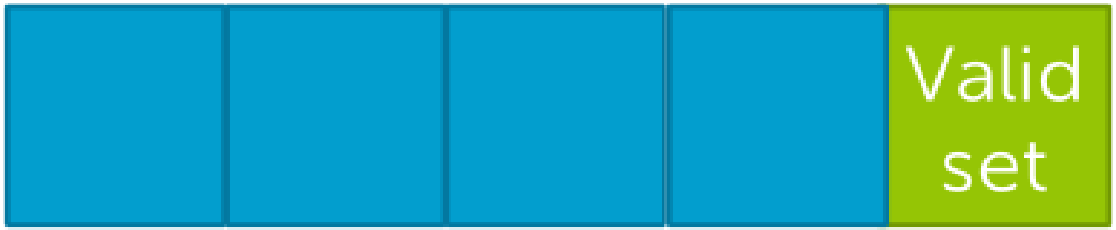


For k=1,…,K

1. Estimate $\hat{\mathbf{w}}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $\text{error}_k(\lambda)$

Compute average error: $CV(\lambda) = \frac{1}{K}\sum_{k=1}^{K}\text{error}_k(\lambda)$

K-fold cross validation



Repeat procedure for each choice of λ

$\boxed{\text{Choose } \lambda^* \text{ to minimize } CV(\lambda)}$

What value of K?

Formally, the best approximation occurs for validation sets of size 1 ($K=N$)

leave-one-out cross validation

Computationally intensive

- requires computing N fits of model per λ

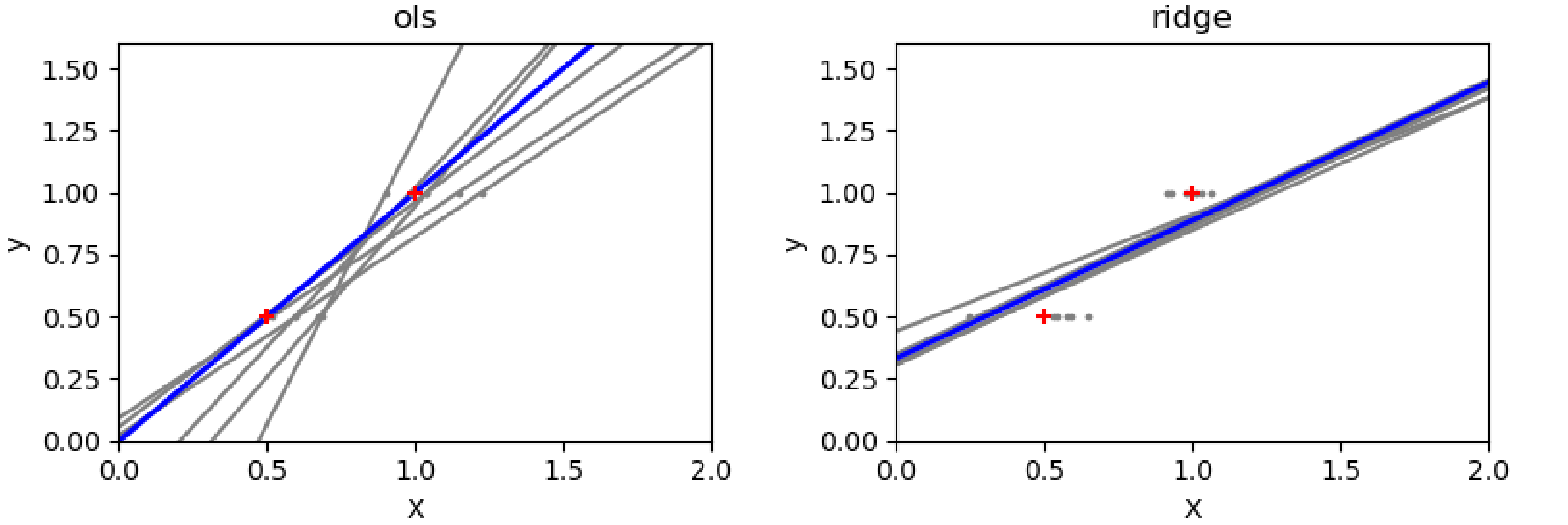
Typically, $K=5$ or 10

5-fold CV

10-fold CV

OLS v/s RIDGE

Ridge regression is basically minimizing a penalised version of the least-squared function. The penalising `shrinks` the value of the regression coefficients. Despite the few data points in each dimension, the slope of the prediction is much more stable and the variance in the line itself is greatly reduced, in comparison to that of the standard linear regression



End Of Theoretical Discussion

Example in Python – Predict Price

```
There are 13 independent variables in each case of the dataset. They are:
CRIM - per capita crime rate by town
ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS - proportion of non-retail business acres per town.
CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX - nitric oxides concentration (parts per 10 million)
RM - average number of rooms per dwelling
AGE - proportion of owner-occupied units built prior to 1940
DIS - weighted distances to five Boston employment centres
RAD - index of accessibility to radial highways
TAX - full-value property-tax rate per $10,000
PTRATIO - pupil-teacher ratio by town
B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT - % lower status of the population
```

1 Linear Regression

[11]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression

lin_regressor=LinearRegression()
mse=cross_val_score(lin_regressor,X,y,scoring='neg_mean_squared_error',cv=5)␣
↪#cv-cross-validation splitting strategy
mean_mse=np.mean(mse)
print(mean_mse)
```

```
-37.13180746769922
```

2 Ridge Regression

[14]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

ridge=Ridge()
parameters={'alpha':
↪[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100]} #Note Let␣
↪alpha = "lambda"
ridge_regressor=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error',cv=5)
ridge_regressor.fit(X,y)
```

[14]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10,
                                   20, 30, 35, 40, 45, 50, 55, 100]},
             scoring='neg_mean_squared_error')
```

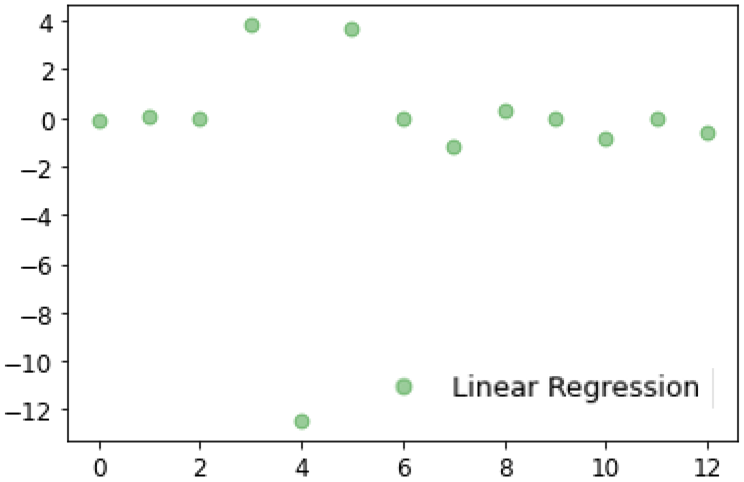
[13]:
```
print(ridge_regressor.best_params_) #Best value of lambda
print(ridge_regressor.best_score_) #Mean Squared Error
```

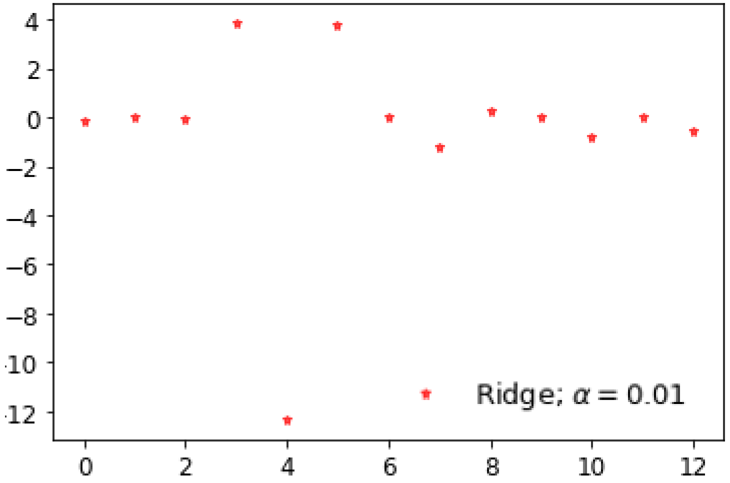
```
{'alpha': 100}
-29.90570194754033
```

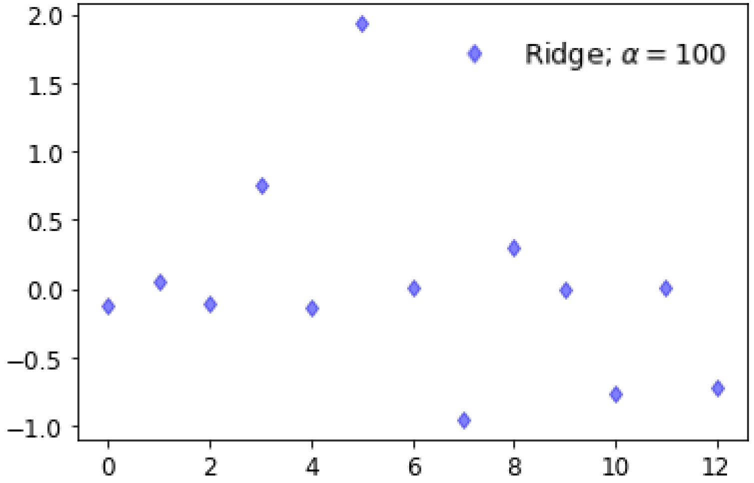
Ridge Regression performes better than Linear regression as -29.905 is more closer to 0 than -37.131.

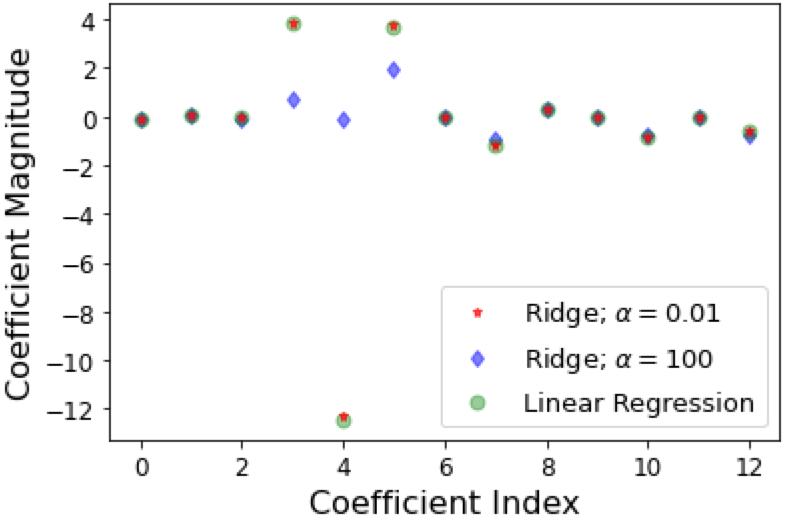
```
# higher the alpha value, more restriction on the coefficients;
# low alpha > more generalization,
# in this case linear and ridge regression resembles
```

Alpha = Lambda λ









In X axis we plot the coefficient index and, for Boston data there are 13 features (for Python 0th index refers to 1st feature)

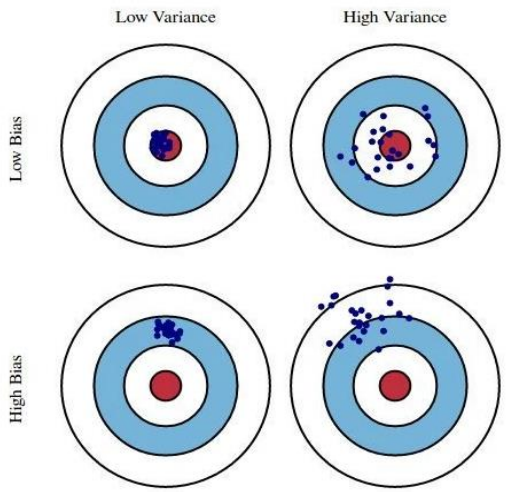
For low value of (0.01), when the coefficients are less restricted, the magnitudes of the coefficients are almost same as of linear regression.

For higher value of (100), we see that for coefficient indices 3,4,5 the magnitudes are considerably less compared to linear regression case.

This is an example of shrinking coefficient magnitude using Ridge regression.

$$\mathrm{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

Is there something more ?



- There are three popular regularization techniques, each of them aiming at decreasing the size of the coefficients:
  - Ridge Regression, which penalizes sum of squared coefficients (L2 penalty).
  - Lasso Regression, which penalizes the sum of absolute values of the coefficients (L1 penalty).
  - Elastic Net, a convex combination of Ridge and Lasso.

$$Y = X\beta + \varepsilon,$$

$$\varepsilon \sim N(0, \sigma^2).$$

$$L_{OLS}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 = ||y - X\hat{\beta}||^2$$

$$L_{ridge}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda\sum_{j=1}^{m}\hat{\beta}_j^2 = ||y - X\hat{\beta}||^2 + \lambda||\hat{\beta}||^2.$$

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda\sum_{j=1}^{m}|\hat{\beta}_j|.$$

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}|\hat{\beta}_j|\right),$$

References :

- https://www.coursera.org/learn/ml-regression
- https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net
- https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b
- https://en.wikipedia.org/wiki/Tikhonov_regularization
- https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
- https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols_ridge_variance.html

Thank you for your attention

Extra Slides

Origin : Tikhonov regularization

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i,$$

$$\mathrm{y} = \mathrm{X}\beta + \varepsilon,$$

$$\mathbf{X}\boldsymbol{\beta} = \mathbf{y},$$

$$\mathrm{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1p} \\ X_{21} & X_{22} & \cdots & X_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{np} \end{bmatrix}, \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Such a system usually has no exact solution, so the goal is instead to find the coefficients - Beta which fit the equations "best", in the sense of solving the quadratic minimization problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\arg\min}\, S(\boldsymbol{\beta}),$$

$$\hat{\boldsymbol{\beta}} = (\mathrm{X}^{\mathsf{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathsf{T}}\mathbf{y}.$$

A special case of Tikhonov regularization, known as **ridge regression** is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters

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

$$Bias(\hat{\beta}_{ridge}) = -\lambda(X'X + \lambda I)^{-1}\beta,$$

$$Var(\hat{\beta}_{ridge}) = \sigma^2(X'X + \lambda I)^{-1}X'X(X'X + \lambda I)^{-1}.$$

Generic basis expansion

Model:

$$y_i = \underline{w_0 h_0(x_i)} + w_1 h_1(x_i) + \ldots + w_D h_D(x_i) + \varepsilon_i$$
$$= \sum_{j=0}^{D} w_j h_j(x_i) + \varepsilon_i$$

feature 1 = $h_0(x)$…often 1 (constant)

feature 2 = $h_1(x)$… e.g., x

feature 3 = $h_2(x)$… e.g., x^2

…

feature D+1 = $h_D(x)$… e.g., x^p

If data are first centered about 0, then favoring small intercept not so worrisome

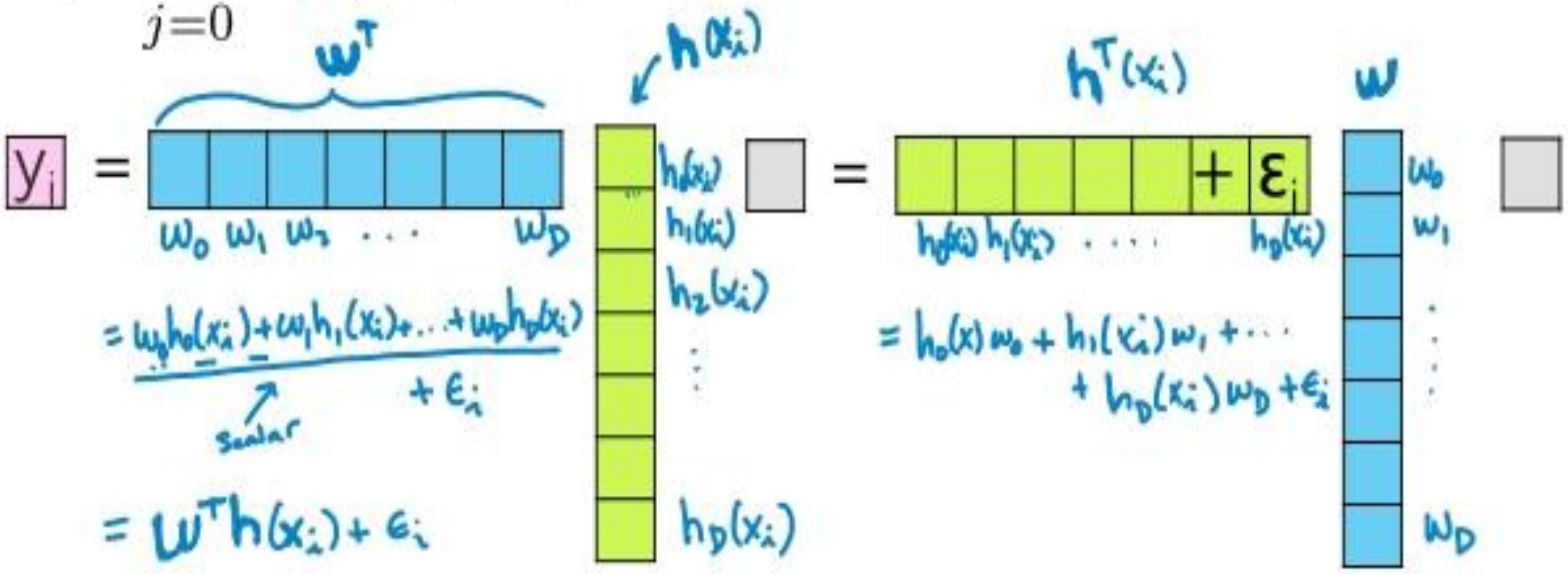
Step 1: Transform y to have 0 mean

Step 2: Run ridge regression as normal

Rewrite in matrix notation

For observation i

$$y_i = \sum_{j=0}^{D} w_j \, h_j(\mathbf{x}_i) + \varepsilon_i$$



RSS in matrix notation

$$\text{RSS}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

$$= (\mathbf{y} - \mathbf{H}\mathbf{w})^T (\mathbf{y} - \mathbf{H}\mathbf{w})$$

residual_1	residual_2	residual_3	...	residual_N

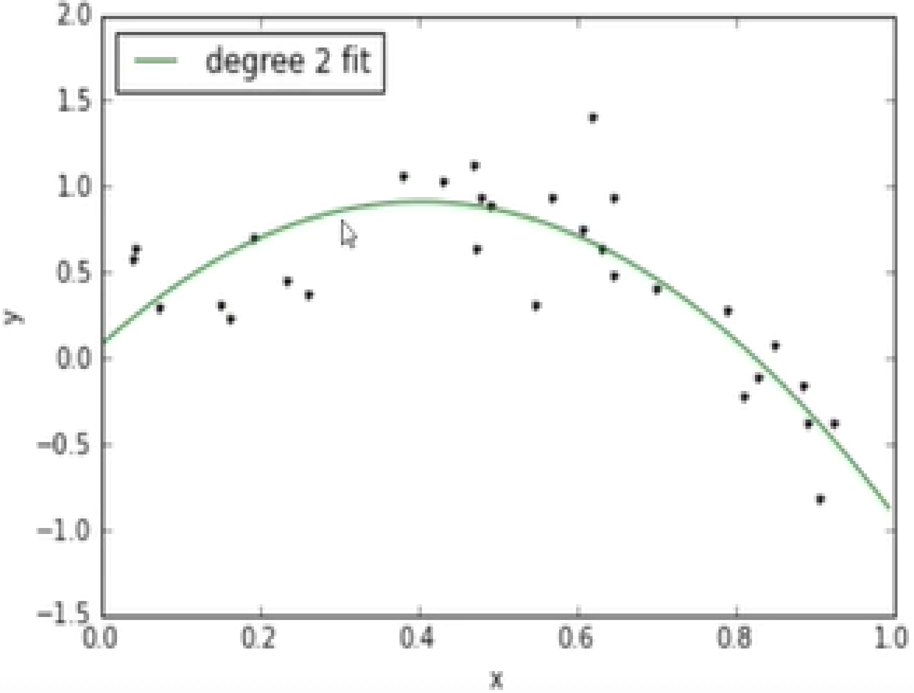
residual_1
residual_2
residual_3
...
residual_N

Symptom of overfitting

Often, overfitting associated with very large estimated parameters $\hat{\mathbf{w}}$

```
print_coefficients(model)
```

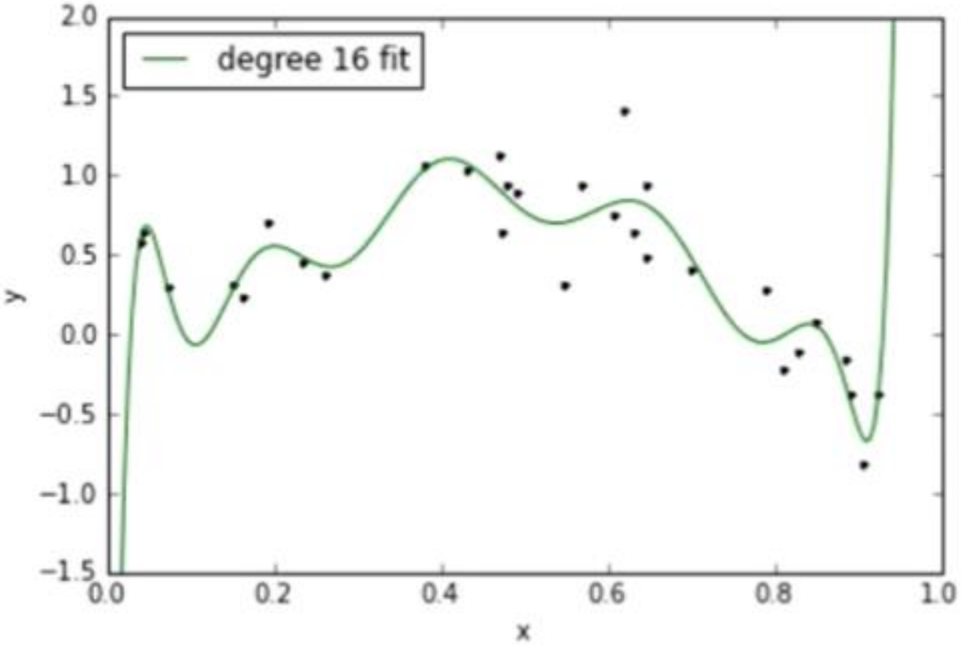
```
Learned polynomial for degree 2:
        2
-5.129 x + 4.147 x + 0.07471
```



30 points drawn from the sinusoid $y = \sin(4x)$:

Here we will apply **Polynomial Regression**.
The Images below show the result :

```
Learned polynomial for degree 16:
            16             15             14             13
2.583e+06 x  - 1.092e+07 x  + 1.443e+07 x  + 1.873e+06 x
             12             11             10             9
 - 2.095e+07 x  + 1.295e+07 x  + 9.366e+06 x  - 1.232e+07 x
             8             7             6             5             4
 - 2.544e+06 x + 1.181e+07 x - 9.325e+06 x + 3.887e+06 x - 9.666e+05 x
             3             2
 + 1.441e+05 x - 1.215e+04 x + 506.6 x - 7.325
```



A1. The linear regression model is "linear in parameters."

A2. There is a random sampling of observations.

A3. The conditional mean should be zero.

A4. There is no multi-collinearity (or perfect collinearity).

A5. Spherical errors: There is homoscedasticity and no autocorrelation

A6: Optional Assumption: Error terms should be normally distributed.

Assumptions

The assumptions are the same as those used in regular multiple regression: linearity, constant variance (no outliers), and independence. Since ridge regression does not provide confidence limits, normality need not be assumed.