Linear Regression
(Problems & Solutions)

TIET, PATIALA

Problem 1: Multicollinearity

- In regression, "multicollinearity" refers to predictors that are correlated with other predictors.
- Multicollinearity occurs when our model includes multiple factors that are correlated not just to your response variable, but also to each other.
- In other words, it results when we have factors that are a bit redundant.
- Multicollinearity increases the standard errors of the coefficients.
- Increased standard errors in turn means that coefficients for some independent variables may be found not to be significantly different from 0.
- In other words, by overinflating the standard errors, multicollinearity makes some variables statistically insignificant when they should be significant.

Problem 2 & 3: Underfitting & Overfitting

- In order to understand the concept of overfitting and underfitting, consider following example (shown in Figure 1(a)) where we have to fit a regression function (hypothesis) that can predict the value of output variable (y) given input variable (x).
- Let us fit three regression hypothesis on the data as shown in Figure 1(b), 1(c), and 1 (d)

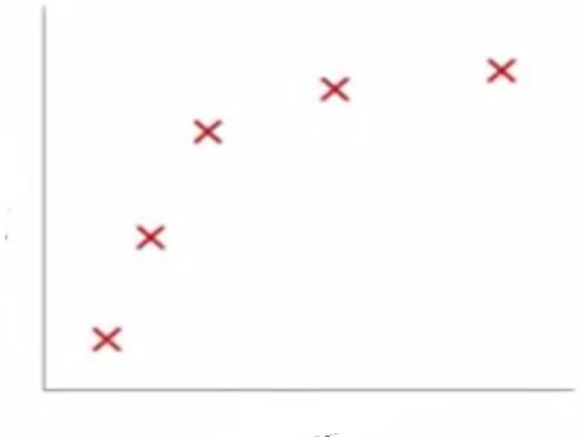


Figure 1(a)

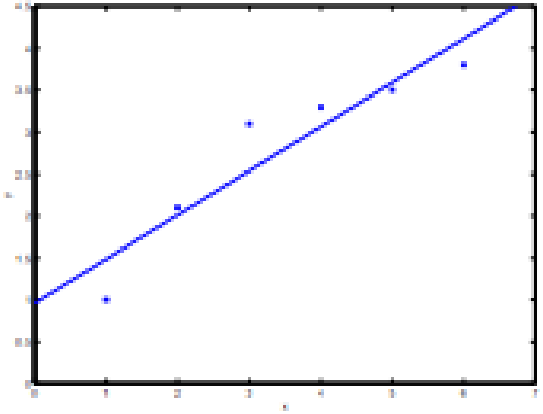


Figure 1(b)

$y^{\wedge} = \beta_0 + \beta_1 x$

(Underfit)

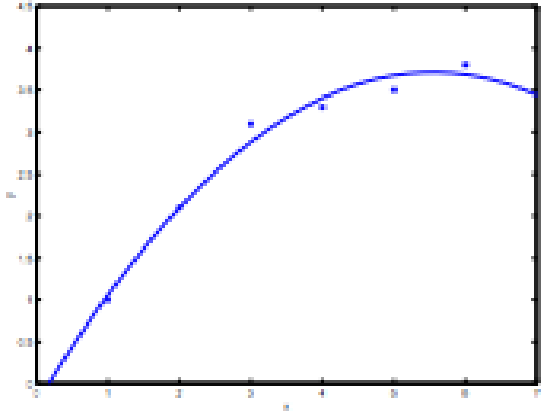


Figure 1(c)

$y^{\wedge} = \beta_0 + \beta_1 x + \beta_2 x^2$

(Good Fit)

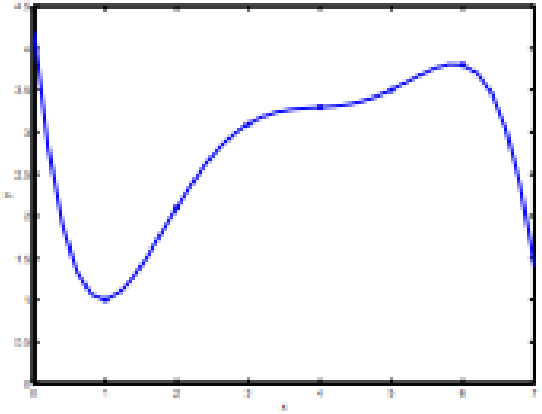


Figure 1(d)

$y^{\wedge} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$

(Overfit)

Underfitting

- The figure 1(a) shows the result of fitting a linear hypothesis $y^{\wedge} = \beta_0 + \beta_1 x$.
- We can see that the data doesn't really lie on straight line, and so the fit is not very good.
- The figure shows an instance of **underfitting**—in which the data clearly shows structure not captured by the model.
- **Underfitting, or high bias**, is when the form of our hypothesis function h maps poorly to the trend of the data.
- It is usually caused by a function that is too simple or uses too few features.
- Underfitting can be solved by increasing the number of features in our training data.

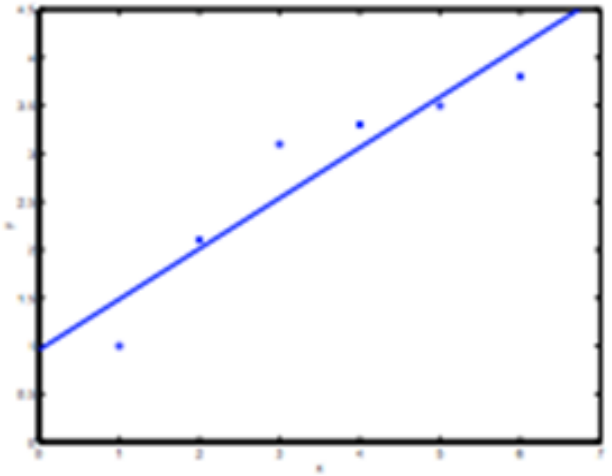


Figure 1(b)
$y^{\wedge} = \beta_0 + \beta_1 x$
(Underfit)

Overfitting

- The figure 1(d) shows the result of fitting a high order polynomial hypothesis $y^{\wedge} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$.
- The figure is an example of **overfitting.**
- **O**verfitting, or high variance, is caused by a hypothesis function that fits the available data but does not generalize well to predict new data.
- It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.
- For example, a quadratic fit shown in Figure 1(c) (slide number 3) is a good fit as it fits well to data and generalizes well.

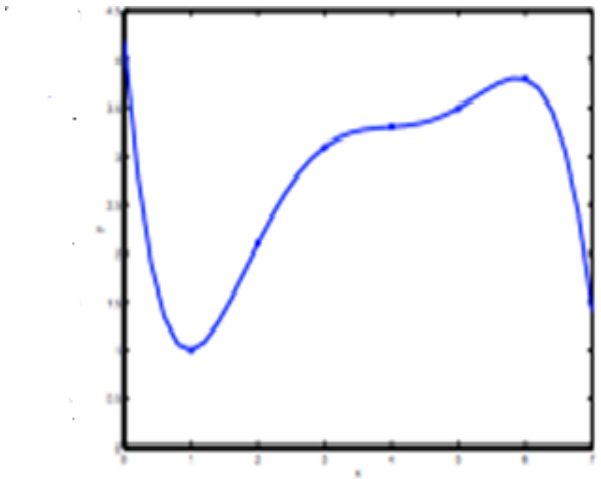


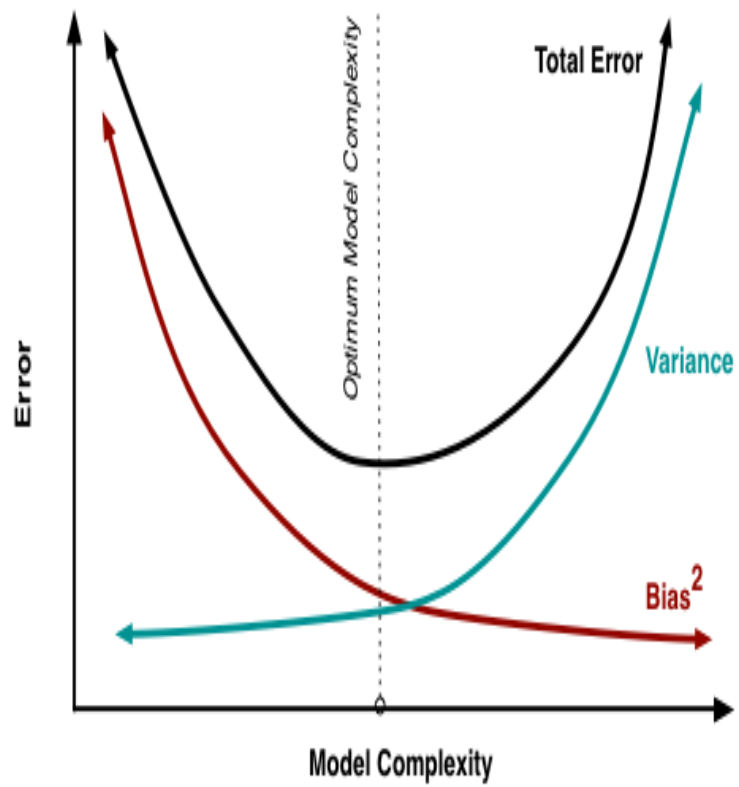
Figure 1(d)

$y^{\wedge} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$

(Overfit)

Bias and Variance Tradeoff

- If we have less number of features i.e., low model complexity, the fit is underfit with high bias error.
- the bias is an error from a faulty assumption in the learning algorithm.
- when the bias is too large, the algorithm would be able to correctly model the relationship between the features and the target outputs.
- As we keep on increasing the model complexity (number of features) bias decreases but variance increases (overfit).
- Variance is an error resulting from fluctuations int he training dataset.
- A high value for variance would cause the algorithm may capture the most data points put would not be generalized enough to capture new data points.
- The tradeoff between bias and variance can be controlled using controlling the complexity of model.



Solutions to Regression Problems

- The problems of multicollinearity, and overfitting can be controlled by handling the complexity of the model.
- The model complexity can be controlled using any of these features:

1. **Remove highly correlated predictors from the model**
   - Manually select which features to keep.
   - Use feature selection methods to choose features that maximizes relevancy or minimizes redundancy.
   - Use feature extraction techniques models such as PCA, SVD, and LDA.
2. **Regularization**

- Keep all the features, but reduce the magnitude of parameters
- Regularization works well when we have a lot of slightly useful features.

Regularization/ Shrinkage

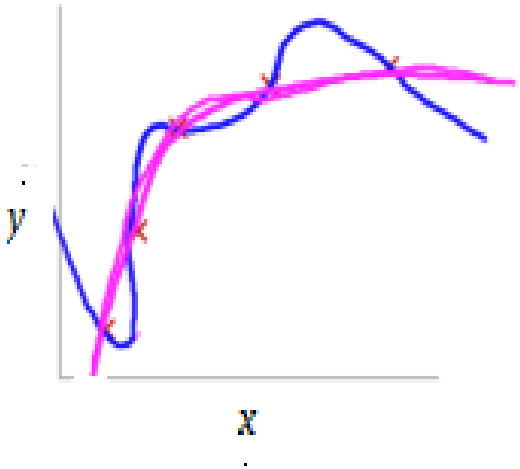
- Regularization is a technique used for tuning the function by adding an additional penalty term in the error function that reduces the magnitude of parameters .
- The additional term controls the excessively fluctuating function such that the coefficients don’t take extreme values.
- This technique of keeping a check or reducing the value of error coefficients are called **shrinkage methods**.
- If we have overfitting from our hypothesis function (as shown in Figure 1(d)), we can reduce the weight of some of the terms in our function carry by increasing their cost.

Regularization/ Shrinkage Contd…

- Without actually getting rid of these features or changing the form of our hypothesis, we can instead modify our **cost function**:

$$Cost\ Function = J(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4) + 5000\ \beta_3^2 + 5000\beta_4^2$$

- We've added two extra terms at the end to inflate the cost of $\beta_3\ and\ \beta_4$.
- Now, in order for the cost function to get close to zero, we will have to reduce the values of $\beta_3\ and\ \beta_4$ to near zero.
- This will in turn greatly reduce the values of $\beta_3\ and\ \beta_4$ in our hypothesis function.
- As a result, we see that the new hypothesis (depicted by the pink curve) looks like a quadratic function but fits the data better due to the extra small terms $\beta_3 x^3 + \beta_4 x^4$



Regularization in Linear Regression

- In regression models, we do not know which regression coefficients we should shrink by adding their penalty in the cost function.
- So, the general tendency of applying regularization in regression is to shrink the weight (regression coefficients) of all the input variables.
- Two most commonly used regularization in linear regression are:

1. Ridge Regression (L2 Normalization)
2. Least Absolute Selection and Shrinkage Operator (LASSO) Regression (L! Normalization)

Ridge Regression

- Ridge regression performs '**L2 regularization**', i.e. it adds a factor of sum of squares of coefficients in the optimization objective.
- Thus, ridge regression optimizes the following:

$$\boldsymbol{Cost\ (Objective) Function = Mean\ Square\ Error + \lambda (\text{sum of square of coefficients})}$$

$$\textbf{i.e., } \boldsymbol{J = \frac{1}{2n}\sum_{i=1}^{n}((y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \beta_3 x_{i3} - \cdots \ldots \ldots \ldots - \beta_k x_{ik})^2 + \lambda \sum_{i=0}^{k} \beta_i^2)}$$

where n are the total number of training examples; k are the number of features; β_j represents regression coefficients of jth input variable and λ is the regularization parameter.

Ridge Regression using Gradient Descent

- We know in gradient descent optimization, we update the regression coefficients as follows:

$$\beta_j = \beta_j - \alpha \frac{\partial(J(\beta))}{\partial \beta_j} \; for \; j = 0,1,2, \dots \dots . . k$$

- For Ridge Regression, cost function is given by:

$$J = \frac{1}{2n} \sum_{i=1}^{n} ((y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \beta_3 x_{i3} - \cdots \dots \dots \dots . . - \beta_k x_{ik})^2 + \lambda \sum_{i=0}^{k} \beta_i^2)$$

- The gradient of cost function w.r.t β's is given by:

$$\frac{\partial J}{\partial \beta_0} = \frac{1}{n} \sum_{i=1}^{n} ((\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots \dots \dots \dots . . + \beta_k x_{ik} - y_i) + \lambda \, \beta_0)$$

Ridge Regression using Gradient Descent Contd…

Similarly, $\frac{\partial J}{\partial \beta_1} = \frac{1}{n}\sum_{i=1}^{n}((\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots \ldots \ldots \ldots + \beta_k x_{ik} - y_i) \times x_{i1} + \lambda\beta_1)$

$$\frac{\partial J}{\partial \beta_2} = \frac{1}{n}\sum_{i=1}^{n}((\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots \ldots \ldots \ldots + \beta_k x_{ik} - y_i) \times x_{i1} + + \lambda\, \beta_2)$$

$$\frac{\partial J}{\partial \beta_3} = \frac{1}{n}\sum_{i=1}^{n}((\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots \ldots \ldots \ldots + \beta_k x_{ik} - y_i) \times x_{i3} + + \lambda\beta_3)$$

$$\vdots$$

In general, $\boldsymbol{\frac{\partial J}{\partial \beta_j} = \frac{1}{n}\sum_{i=1}^{n}((\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots \ldots \ldots \ldots + \beta_k x_{ik} - y_i) \times x_{ij} + \lambda\, \beta_j)}$

Ridge Regression using Gradient Descent Contd…

- Therefore, the regression coefficients are updated as:

$$\beta_j = \beta_j - \frac{\alpha}{n}\sum_{i=1}^{n}((\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots \ldots \ldots \ldots + \beta_k x_{ik} - y_i) \times x_{ij} + \lambda \beta_j)$$

or $$\boldsymbol{\beta_j = \beta_j\left(1 - \frac{\alpha\lambda}{n}\right) - \frac{\alpha}{n}\sum_{i=1}^{n}(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots \ldots \ldots \ldots + \beta_k x_{ik} - y_i) \times x_{ij}}$$

Since the factor $\boldsymbol{\left(1 - \frac{\alpha\lambda}{n}\right)}$ is less than 1, therefore, the algorithm, will keep on shrinking the values of the regression coefficients, and will handle the problem of overfitting.

Ridge Regression using Least Square Error Fit

- In least Square error fit, we represent error in matrix form as:

$$\epsilon = y - X\beta$$

- Where $\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$; y= $\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$; $\beta = \begin{bmatrix} \beta_0 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$

and $X = \begin{bmatrix} 1 & x_{11} & x_{12} \cdots & x_{1k} \\ 1 & x_{21} & x_{22} \cdots & x_{2k} \\ \vdots & \vdots & \vdots \ \vdots & \vdots \\ 1 & x_{n1} & x_{n2} \cdots & x_{nk} \end{bmatrix}$

$$\boldsymbol{Cost\ (Objective) Function = Mean\ Square\ Error + \lambda(\text{sum of square of coefficients})}$$

Ridge Regression using Least Square Error Fit Contd....

$$\boldsymbol{Cost\ (Objective) Function = Mean\ Square\ Error + \lambda}\textbf{(sum of square of coefficients)}$$

$$J(\beta) = \frac{1}{2n}\sum_{i=1}^{n}\left(\epsilon_i^2 + \lambda\sum_{j=1}^{k}\beta_j^2\right) = \frac{1}{2n}(\epsilon^T\epsilon + \lambda\beta^T\beta)$$

$$= \frac{1}{2n}((y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta)$$

$$= \frac{1}{2n}((y^T - \beta^T X^T)(y - X\beta) + \lambda\beta^T\beta)$$

$$J(\beta) = \frac{1}{2n}(y^T y - \beta^T X^T y - y^T X\beta + \beta^T X^T X\beta + \lambda\beta^T\ \beta)$$

$$\boldsymbol{J(\beta)} = \frac{1}{2n}\ (\boldsymbol{y^T y - 2y^T X\beta + \beta^T X^T}X\beta + \lambda\beta^T\ \beta)$$

[Because $y^T X\beta$ and $\beta^T X^T y$ is always equal with only one entry. The square error function is minimized using **second derivative test**]

Ridge Regression using Least Square Error Fit Contd....

- **Step 1: Compute the partial derivate of J(β) w.r.t β**

$$\frac{\partial \mathrm{J}(\beta)}{\partial \beta} = \frac{1}{2n}\frac{\partial (y^T y - 2y^T X\beta + \beta^T X^T\, X\beta + \lambda\beta^T\, \beta)}{\partial \beta}$$

$$= \frac{1}{2n} \times \frac{\partial y^T y}{\partial \beta} - \frac{\partial 2y^T X\beta}{\partial \beta} + \frac{\partial \beta^T X^T\, X\beta}{\partial \beta} + \frac{\partial \lambda\beta^T \beta}{\partial \beta}$$

$$= \frac{1}{2n} \times \left(0 - 2X^T y\frac{\partial \beta}{\partial \beta} + \frac{\partial \beta^T X^T\, X\beta}{\partial \beta} + \frac{\partial \lambda\beta^T \beta}{\partial \beta}\right) [Because\ \frac{\partial AX}{\partial X} = A^T]$$

$$= \frac{1}{2n} \times (-2X^T y + 2X^T\, X\beta + 2\lambda I\beta) = -X^T y + X^T\, X\beta + \lambda I\beta$$

$$[Because\ \frac{\partial X^T AX}{\partial X} = 2AX]$$

Ridge Regression using Least Square Error Fit Contd....

- **Step 2: Compute $\hat{\beta}$ for β for which $\frac{\partial J(\beta)}{\partial \beta} = 0$**

$$-X^T y + X^T X\hat{\beta} + \lambda I\hat{\beta} = 0$$

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

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

- **Step 3: Compute $\frac{\partial^2 J(\beta)}{\partial \beta^2}$ and prove it to be minimum for $\hat{\beta}$**

$$\frac{\partial^2 J(\beta)}{\partial \beta^2} = \frac{\partial(-X^T y + X^T X\beta + \lambda I\beta)}{\partial \beta} = 0 + 2\,XX^T + 0 = +ve$$

- Thus, β is updated as $\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$. It will solve the problem of overfitting and multicollinearity as $|X^T X + \lambda I|$ will not be zero for correlated features.

LASSO Regression

- LASSO regression performs '**L1 regularization**', i.e. it adds a factor of sum of absolute values of coefficients in the optimization objective.

- Thus, ridge regression optimizes the following:

$$\boldsymbol{Cost\ (Objective) Function = Mean\ Square\ Error + \lambda (\text{sum of absolute values of coefficients})}$$

$$\textbf{i.e., } \boldsymbol{J = \frac{1}{2n}\sum_{i=1}^{n}((y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \beta_3 x_{i3} - \cdots \ldots\ldots\ldots\ldots - \beta_k x_{ik})^2 + \lambda \sum_{i=0}^{k} |\beta_i|)}$$

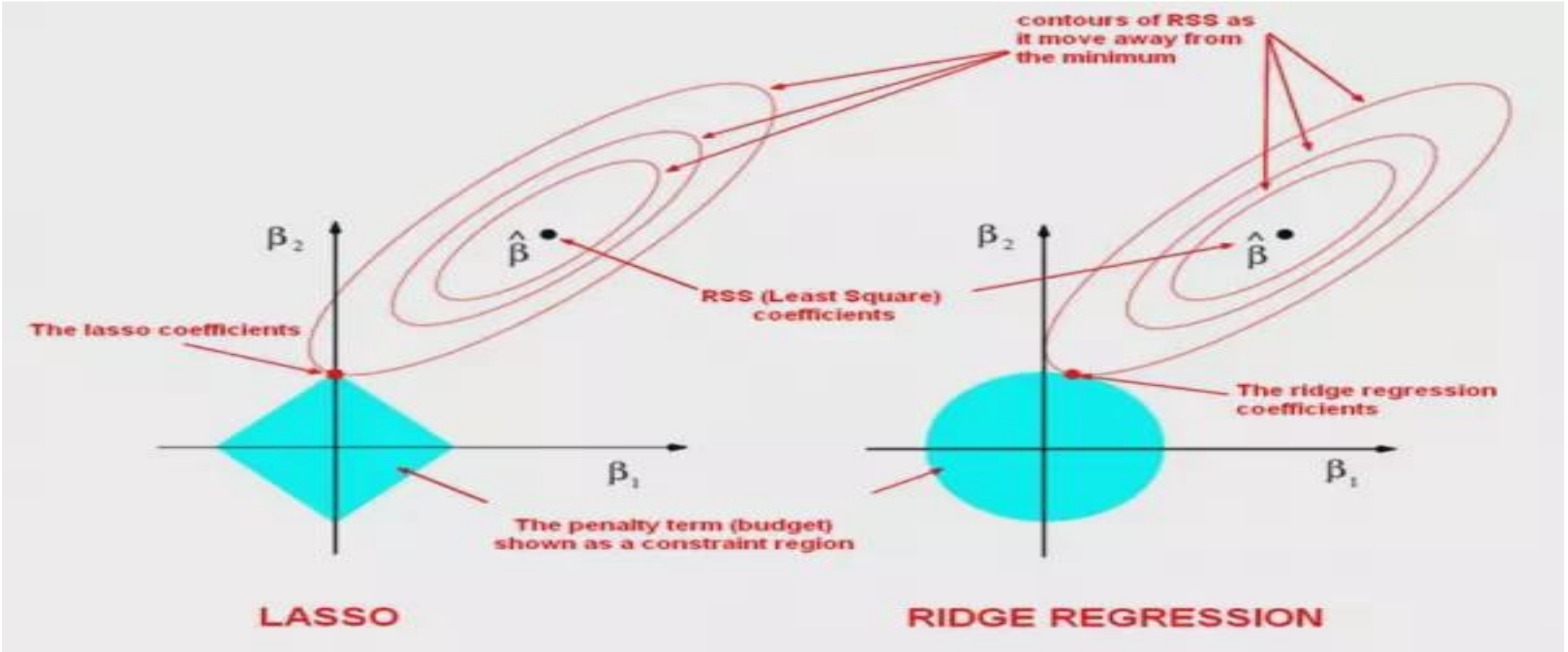
where n are the total number of training examples; k are the number of features; β_j represents regression coefficients of jth input variable and λ is the regularization parameter.

Ridge vs LASSO

1. Key Difference

- **Ridge:**
  - It includes all (or none) of the features in the model.
  - Thus, the major advantage of ridge regression is coefficient shrinkage and reducing model complexity.
- **LASSO:**
  - Along with shrinking coefficients, lasso performs feature selection as well. (Remember the '*selection*' in the lasso full-form)
  - Some of the coefficients become exactly zero, which is equivalent to the particular feature being excluded from the model.

Ridge vs. LASSO



Ridge vs LASSO

2. Typical Use Cases

- **Ridge:**
  - It is majorly used to *prevent overfitting*.
  - Since it includes all the features, it is not very useful in case of exorbitantly high features, say in millions, as it will pose computational challenges.
- **LASSO:**
  - Since it provides *sparse solutions*, it is generally the model of choice (or some variant of this concept) for modelling cases where the #features are in millions or more.
  - In such a case, getting a sparse solution is of great computational advantage as the features with zero coefficients can simply be ignored.

Ridge vs LASSO

3. Presence of Highly Correlated Features

- **Ridge:**
  - It generally works well even in presence of highly correlated features as it will include all of them in the model but the coefficients will be distributed among them depending on the correlation.
- **LASSO:**
  - It arbitrarily selects any one feature among the highly correlated ones and reduced the coefficients of the rest to zero.
  - Also, the chosen variable changes randomly with change in model parameters. This generally doesn't work that well as compared to ridge regression.