Date: / /
Page No.

RIDGE AND LASSO REGRESSION

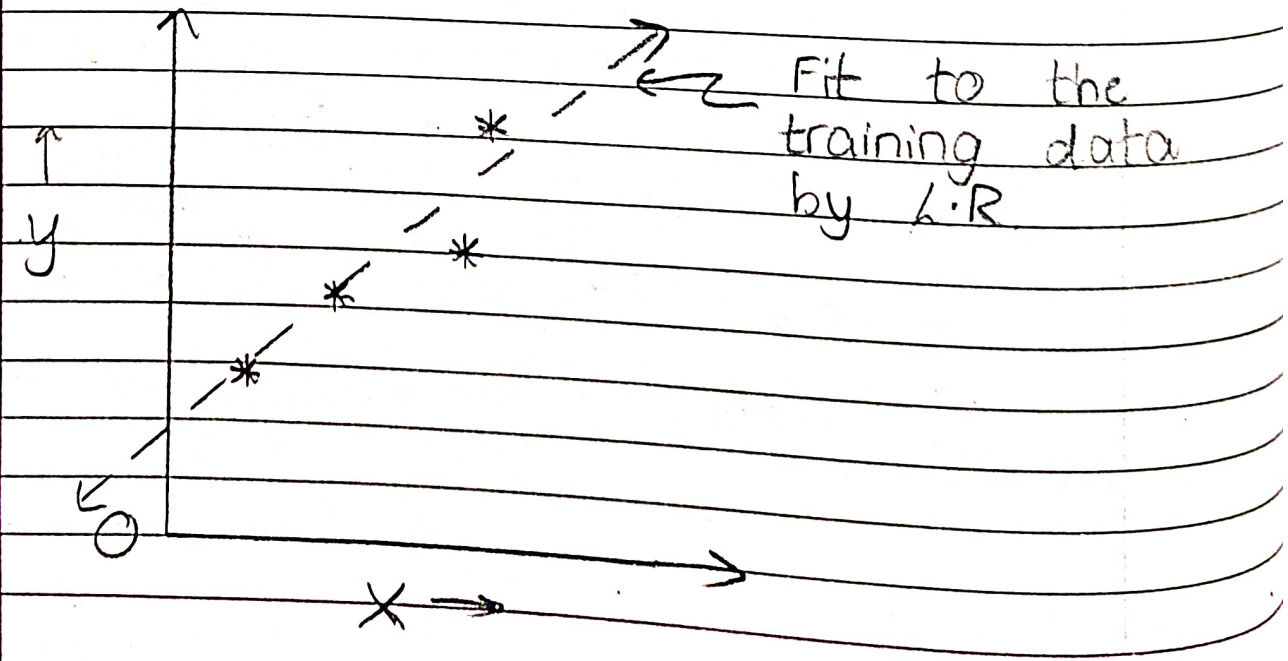
* RIDGE REGRESSION :

This is what linear regression looks like :

$$J(\theta) = \sum_{i=1}^{m} \left(h(x^{(i)}) - y^{(i)}\right)^2$$

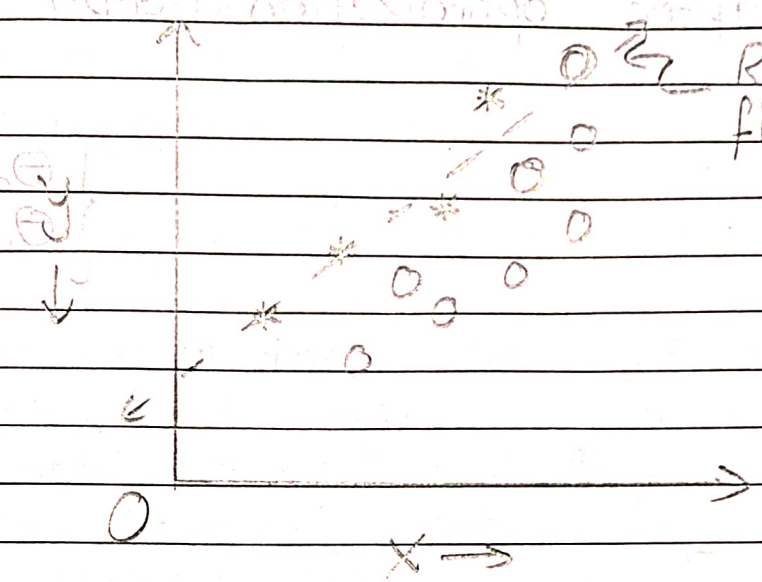
Our aim is to minimize this cost function and thus, linear regression aims at minimizing the squared error only. This means that if the data is non-representative or not sufficient enough, the model will not perform well during cross-validation and testing because it will sort-of overfit the training data and fail to generalize.

Eg: Training Dataset looks like this -



Date: / /
Page No.

However the entire dataset looks like this



Clearly, the model overfit the training in this case because, the training data itself is non-representative and sort of biased.

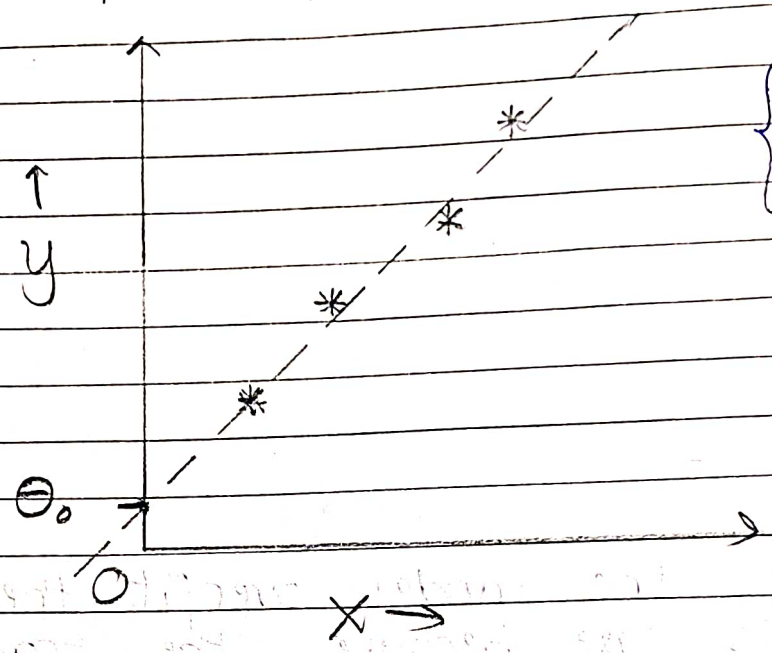
POSSIBLE SOLUTION:

Have a sort-of 'dial' or tweaker that helps us increase/decrease the weights/parameters so that the curve can generalize better. Let's understand this with a simple example:

Example- Let's assume that we have used the above mentioned data and trained Linear Regression Model with the following hypothesis

$$h_\theta(x) = \underbrace{\theta_0}_{\text{Bias}} + \underbrace{\theta_1 x}_{\text{Weight}}$$

Let's say the model fits the training data after optimization (G.D.) like this



Now, lets introduce a new term in the cost function because that is what needs to be minimized to get to a 'good fit'.

Initially, $J(\theta) = \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2$

simplifying, $J(\theta) = \Sigma(\hat{y} - y)^2$

Now, let's change this to

$J(\theta) = \Sigma(\hat{y} - y)^2 + \lambda(\text{slope})^2$

which will generalize to

$J(\theta) = \Sigma(\hat{y} - y)^2 + \lambda \sum_{j=1}^{n} (\theta_j^2)$

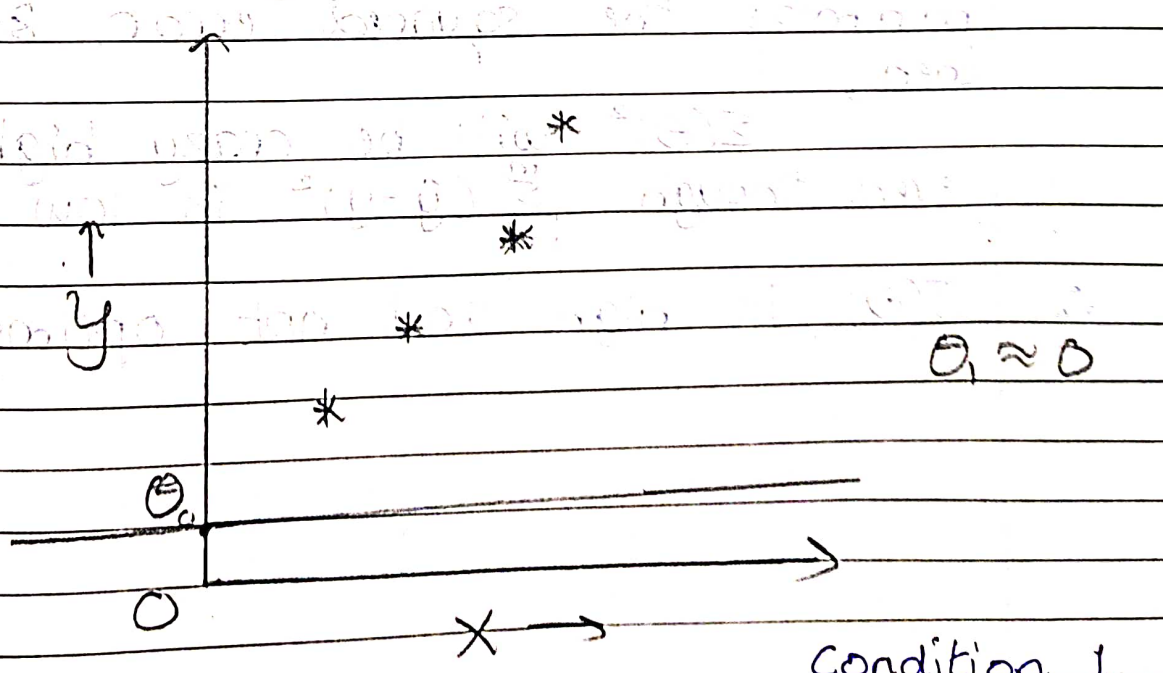
(Excluding the bias)

Here the lambda λ is called the regularization constant because it acts like a dial to determine the extent of regularization (decrease the weights)

What does this do ?

Since now the cost function depends on both the weights squared, and the squared error, thus to minimize the cost function, the algorithm will have to MINIMIZE THE SQUARED ERROR AS WELL AS DECREASE THE WEIGHT(S) TO GET TO OPTIMUM MINIMUM

Now, understand this, the algorithm can't simply ignore the squared error and minimize to weights to approach to 0 because that will lead to high overall cost because of the high squared error. This condition will be like this,



Condition 1.

Date: / /
Page No.

Note this:
The $\theta_i \approx 0$ but not equal to 0. This is because Ridge Regression uses

$$\sum_{j=1}^{n} (\theta_j)^2$$ and thus, once θ_i reaches a sufficiently low value like < 0.1 or so, $(\theta_i)^2 < 0.01$ and thus there is no need to optimize θ_i further.

In condition 1,

$\Sigma(\theta)^2 \to 0$ but,

$\sum_{j=1}^{m} (\hat{y} - y)^2$ is crazy high.

$\therefore$ $J(\theta)$ is high and not optimum.

Similarly, the algorithm can't simply ignore the θ's (weights) and just minimize the squared error. Because then,

$\Sigma(\theta)^2$ will be crazy high.
even though $\sum_{i=1}^{m} (\hat{y} - y)^2$ is low.

$\therefore$ $J(\theta)$ is high and not optimum.

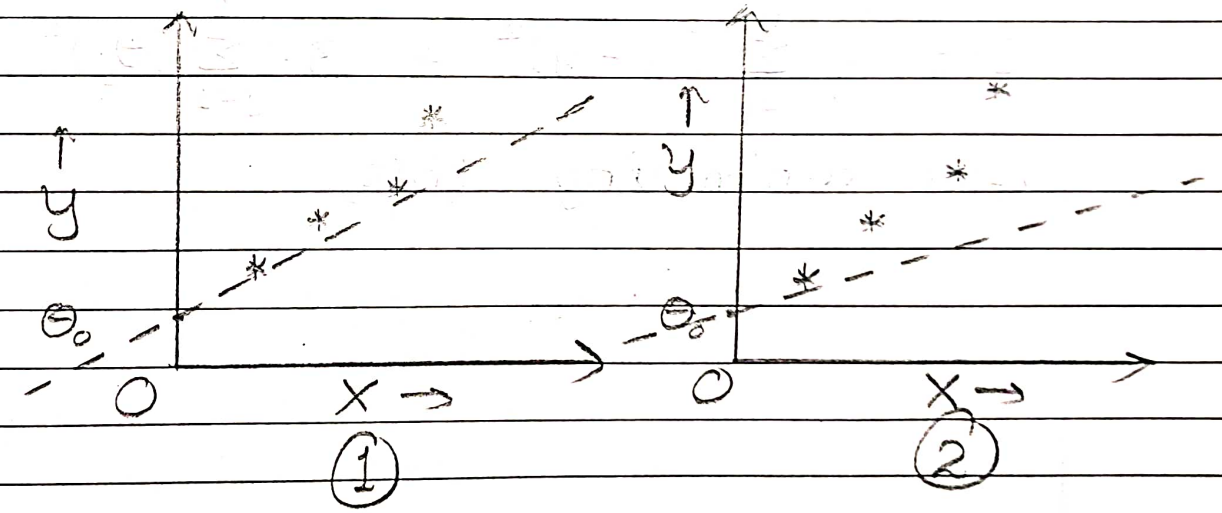
Therefore, the algorithm makes some trade-off between minimizing the first term (SE) and minimizing the second term (weights) and the extent of this tradeoff is determined by λ.

Eg: $J(\theta) = \{\Sigma(\hat{y}-y)^2\} + \lambda\{\Sigma(\theta)^2\}$

Case 1: $(\lambda = 1)$

$$\Sigma(\hat{y}-y)^2 + 1.\{\Sigma(\theta)^2\}$$

and after trying to minimize this we end up having $\theta_0 = 0.5$ (unchanged) and $\theta_1 = 0.7$. Refer fig (1).



Case 2: $(\lambda = 5)$

$$\Sigma(\hat{y}-y)^2 + 5.\{\Sigma(\theta)^2\}$$

Now, to minimize this the second term will play a major role as compared to case 1 because $\lambda = 5$ and the algo will have to make more efforts to minimize the weights as compared to the SE.

Therefore, an increase in λ forces the algo to reduce the weights more in order to minimize $J(\theta)$ and vice versa.

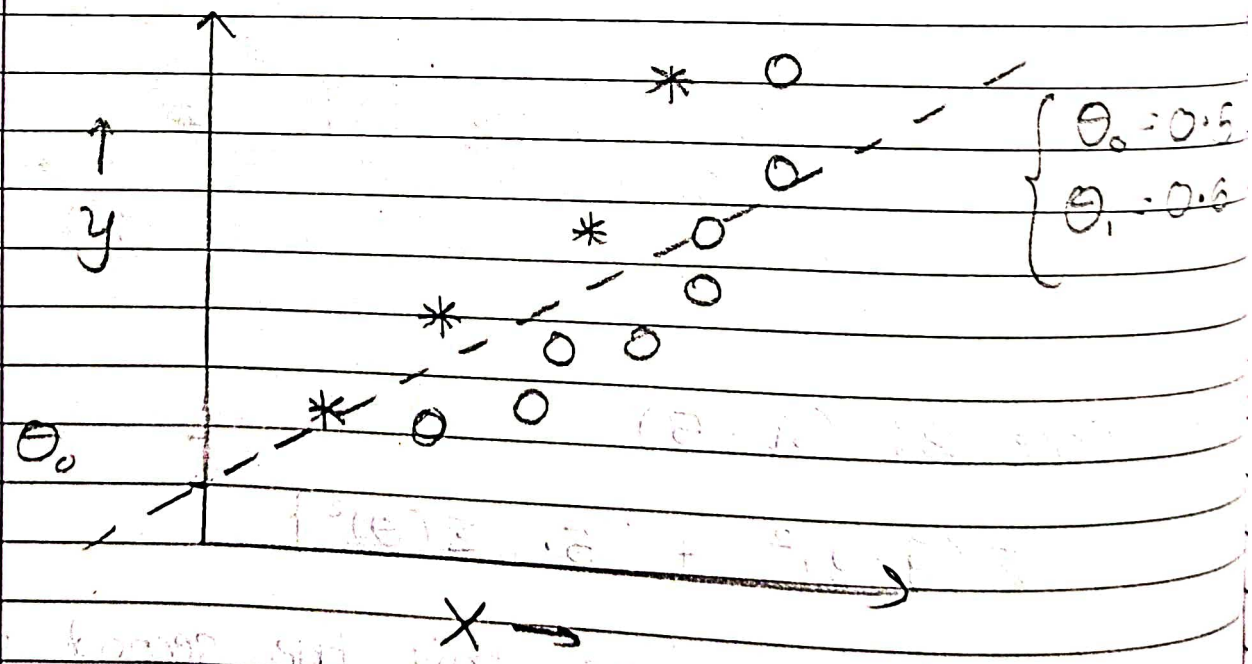
Now, what you can do is try different values of λ and run cross-validation tests to determine the best fit and generalization.

Here, after trying $\lambda = 0.1, 0.5, 1, 2, \ldots$

we find that $\lambda = 3$ is helping us minimize the test set error, so we use that.

$$\therefore \quad J(\theta) = \sum_{i=1}^{m} (\hat{y} - y)^2 + 3 \cdot \sum_{j=1}^{1} (\theta_j)^2$$

after minimizing this,



Therefore, Ridge Regression is used to decrease the variance of our model by decreasing the weights. So by starting with what might look like worse fits to the training data, it can prove to be better in the long term, only because Linear Regression on poor quality training data leads to poor predictions.

This applies for data with more factors / features as well, if you have 4 features, say $\theta_1, \theta_2, \theta_3$ & θ_4,

$$J(\theta) = \sum_{i=1}^{m} (\hat{y} - y)^2 + \lambda(\theta_1^2 + \theta_2^2 + \theta_3^2 + \theta_4^2)$$

Thus, minimizing every weight and thus, if a particular weight has a very huge value, say $\theta_2 = 50$,

(Note that slope = $\tan\theta$ & $0 \leq \tan\theta < \infty$)

Then $\lambda \times \theta_2^2 = 2500\lambda$ which is even more huge, so the algo will try to minimize this a little bit more (in magnitude) as compared to the other already-small weights, and of course you can change λ as you wish.

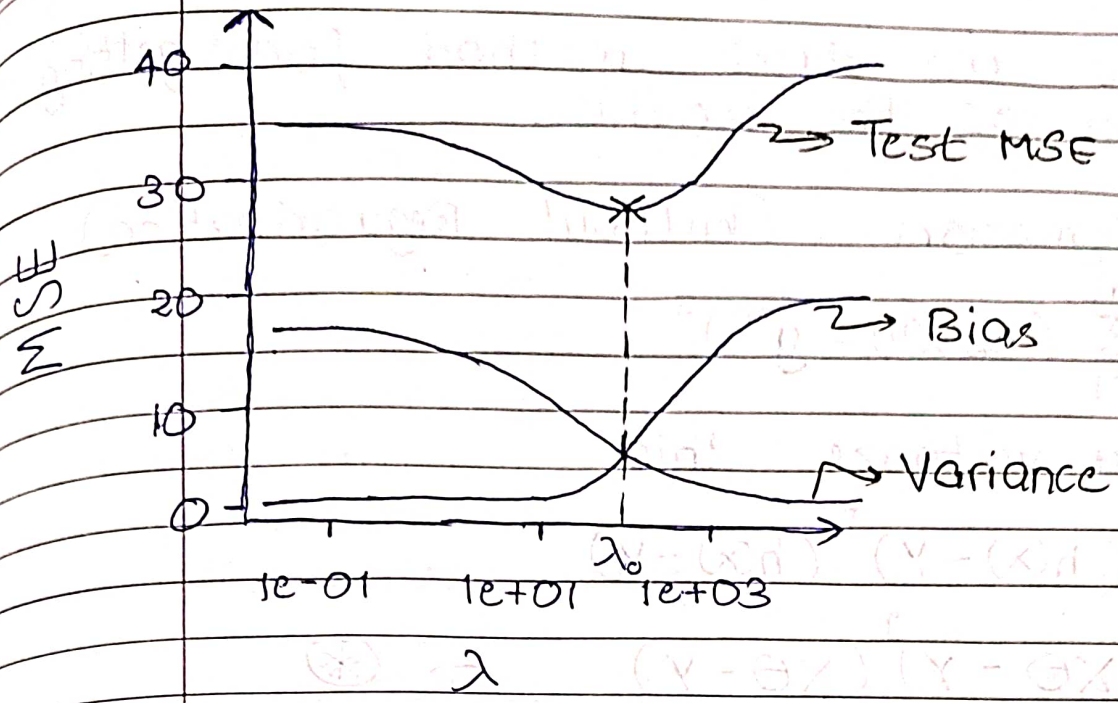
Date: / /
Page No.

Probably, the most amazing feature of Ridge Regression is that it works well when we are short of data. Even if we have 100 features and just 50 training samples, RR works because it will deal with the weights (slopes) of the model then, rather than first focusing on minimizing squared error.

SOME LIMITATIONS:

① Can't help in removing the unnecessary features as it never reduces / leads to weights being equal to 0. It just decreases the complexity of the model and thus, is not good for feature reduction.

② I feel that, in case we want a particular feature or a few features to remain as 'important' as they are, we can't do that unless we remove them from the cost function or manually multiply their weights with a small value to decrease the dependence of $J(\theta)$ on them.



As can be seen, the tradeoff between the bias and the variance is actually good for the long term generalization. For only a small increase in bias the variance drops significantly and thus, the MSE drops considerably as λ increases from 0 to 10. After this point, the decrease variance gradually slows down and the bias starts increasing rapidly. The sweet spot is achieved at about $\lambda = \lambda_0$.

Also, the test MSE is basically f(the bias + the variance)

Test MSE = f(Bias + Variance)

Page No.

Now, let's use direct method for getting the values of the weights.

① Linear Regression (Without Regularization)

$$J(\theta) = \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

When you vectorize this,

$$J(\theta) = (h(x) - Y)^T (h(x) - Y)$$

$$J(\theta) = (X\theta - Y)^T (X\theta - Y) \quad \text{— } (*)$$

where X is your input matrix (mxn) and θ is your parameter vector (nx1)

Our aim is to minimize this cost function (*):

$$\min_\theta J(\theta) = \min_\theta (X^T\theta^T - Y^T)(X\theta - Y)$$

This is wrong!

Remember, $(A+B)^T = A^T B^T$

$(AB)^T = B^T A^T$

$$\Rightarrow \min_\theta J(\theta) = \min_\theta (\theta^T X^T - Y^T)(X\theta - Y)$$

$$\Rightarrow \min_\theta (\theta^T X^T X\theta) - (\theta^T X^T Y) - (Y^T X\theta) + (Y^T Y)$$

Date: / /
Page No.

Let:

$$(\theta^T X^T X \theta) - (\theta^T X^T Y) - (Y^T X \theta) + (Y^T Y) = \mathcal{L}$$

where $\mathcal{L} \in \mathbb{R}$ because note that as soon as we vectorized our cost function, the cost function is basically a summation of a vector product its transpose. So each term in $\mathcal{L}$ is a scalar quantity. Thus, $\mathcal{L} \in \mathbb{R}$

Now, minimizing requires finding partial derivatives.

Remember,

$$X = \begin{bmatrix} — & x^{(1)} & — \\ — & x^{(2)} & — \\ | & | & | \\ — & x^{(m)} & — \end{bmatrix} \quad (m \times n+1)$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad (n+1)$$

$$Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad (m)$$

Computing the derivative

$$\nabla_\theta J(\theta) = \nabla_\theta \{\theta^T X^T X\theta - \theta^T X^T Y - Y^T X\theta + Y^T Y\}$$

Some rules of matrix calculus :

① $\nabla_\theta a^T\theta = a$

② $\nabla_\theta \theta^T a = a$

③ $\nabla_\theta \theta^T A\theta = 2A\theta$ where $A\ (n\times n)$

$$\nabla_\theta J(\theta) = 2X^T X\theta - \nabla_\theta(\theta^T X^T Y + \theta^T X^T Y) + 0$$

($Y^T X\theta$ is a scalar and is same as $\theta^T X^T Y$ $_{(1\times n)(n\times m)(m\times 1)}$)
$_{(1\times n)(m\times n)(n\times 1)}$

$$\Rightarrow \nabla_\theta J(\theta) = 2X^T X\theta - \nabla_\theta(2\theta^T X^T Y)$$

$$\Rightarrow \boxed{\nabla_\theta J(\theta) = 2X^T X\theta - 2X^T Y} = 0$$

$$\Rightarrow X^T X\theta = X^T Y$$

$$\Rightarrow (X^T X)^{-1}(X^T X\theta) = (X^T X)^{-1}(X^T Y)$$

$$\Rightarrow (X^T X)^{-1}(X^T X)(\theta) = (X^T X)^{-1}(X^T Y)$$

$$\rightarrow \boxed{\theta = (X^T X)^{-1}(X^T Y)} \rightarrow \text{Closed Form Solution}$$

Date: / /
Page No.

② Ridge Regression :

$$J(\theta) = (X\theta - Y)^T(X\theta - Y) + \lambda(\theta^T\theta)$$

$$J(\theta) = (\theta^T X^T - Y^T)(X\theta - Y) + \lambda(\theta^T\theta) \quad \text{— (*)}$$

$$\nabla_\theta J(\theta) = \nabla_\theta \{\theta^T X^T X\theta - Y^T X\theta - \theta^T X^T Y + Y^T Y + \lambda\theta^T\theta\}$$

$$\Rightarrow \boxed{(2X^TX\theta - 2X^TY + 0 + 2\lambda\theta) = \nabla_\theta J(\theta)}$$

$$\Rightarrow \nabla_\theta J(\theta) = 0$$

$$\Rightarrow X^TX\theta + \lambda\theta = X^TY$$

$$\Rightarrow (X^TX + \lambda I)\theta = X^TY$$

$$\Rightarrow \boxed{\theta = (X^TX + \lambda I)^{-1} X^TY} \rightarrow \text{closed solution.}$$

Now, as you can see, if $\lambda = 0$, the Ridge coefficient θ becomes the same as Linear coefficient described before.

Q. One question that remains to be answered is that why haven't we penalized the intercept θ_0?

Ans: A very intuitive explanation for the same can be, the cost function will start 'contradicting itself'.

If $J(\theta) = \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)}) + \lambda \sum_{j=0}^{n} (\theta_j)^2$

Then consider the fact, that if we add a constant c to all y's, β_0 must also increase by c and thus all the $\hat{y}^i$'s will also increase by c. But, since we have penalized θ_0 / β_0, thus θ_0 won't increase as much as c and thus $\hat{y}_i$ won't increase by c and thus the model will become less sensitive in choosing the correct origin.

So, finally where does Ridge Regression shine and where it doesn't?

It shines, when:

① You want to kind-of keep the number of features (complexity) high so that your model has higher degrees of freedom, but at the same time don't want your model to be too complex and overfit your data causing high variance.

Date: / /
Page No.

② When the effect of outliers in your training set is making your model go crazy and you want to bring it a little down so that the features suffering from it have their weights toned down.

③ When $n \approx m$ or $n > m$, your least squares model might not even have a unique solution, then the bias-variance tradeoff will pay off.

④ Rather than selecting the best sub-set of features (decreasing the complexity) for which time complexity is $O(2^n)$, you will spend less time optimizing your λ and getting the same high-complexity model with a tuned regularization.