

### Effect #1

When predictor variables are correlated, the estimated regression coefficient of any one variable depends on which other predictor variables are included in the model.

### Effect #2

When predictor variables are correlated, the precision of the estimated regression coefficients decreases as more predictor variables are added to the model.

### Effect #3

When predictor variables are correlated, the marginal contribution of any one predictor variable in reducing the error sum of squares varies depending on which other variables are already in the model.

### Effect #4

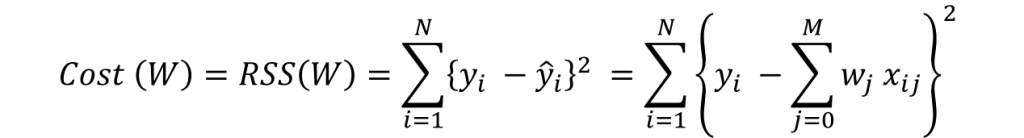
When predictor variables are correlated, hypothesis tests for βk = 0 may yield different conclusions depending on which predictor variables are in the model. (This effect is a direct consequence of the three previous effects.)

### Effect #5

High multicollinearity among predictor variables does not prevent good, precise predictions of the response within the scope of the model.

1. Simple Linear Regression

The objective function (also called as the cost) to be minimized is just the RSS (Residual Sum of Squares), i.e. the sum of squared errors of the predicted outcome as compared to the actual outcome. This can be depicted mathematically as:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq2.png)

In order to minimize this cost, we generally use a ‘gradient descent’ algorithm. I’ll not go into the details right now but you can refer this. The overall algorithm works as:

1. initialize weights (say w=0)

2. iterate till not converged

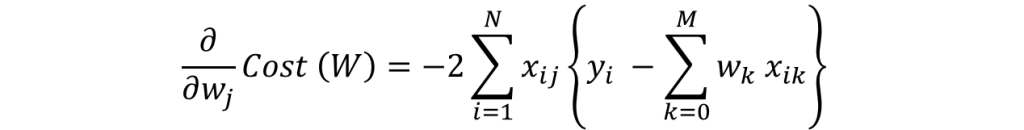
2.1 iterate over all features (j=0,1...M)

2.1.1 determine the gradient

2.1.2 update the jth weight by subtracting learning rate times the gradient

w(t+1) = w(t) - learning rate \* gradient

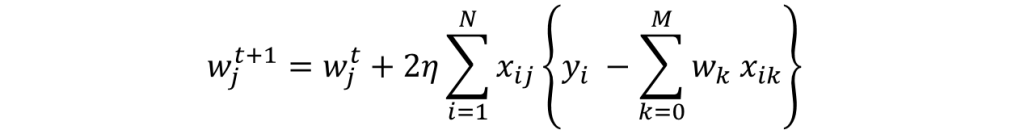
Here the important step is #2.1.1 where we compute the gradient. Gradient is nothing but a partial differential of the cost with respect to a particular weight (denoted as wj). The gradient for the jthweight will be:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq3_updated.png)

This is formed from 2 parts:

1. 2\*{..} : This is formed because we’ve differentiated the square of the term in {..}
2. -wj : This is the differentiation of the part in {..} wrt wj. Since its a summation, all other would become 0 and only wj would remain.

Step #2.1.2 involves updating the weights using the gradient. This update step for simple linear regression looks like:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq4-1.png)

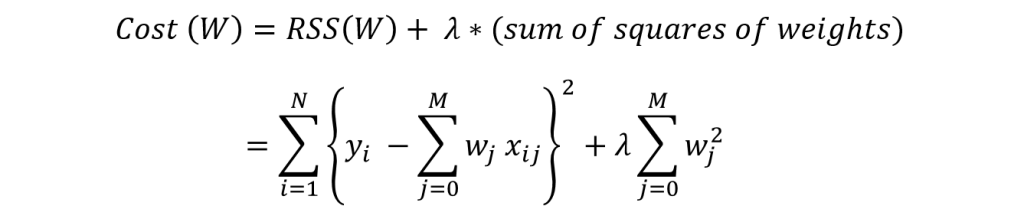
I hope you are able to follow along. Note the +ve sign in the RHS is formed after multiplication of 2 -ve signs. I would like to explain point #2 of the gradient descent algorithm mentioned above ‘**iterate till not converged**‘. Here convergence refers to attaining the optimum solution within pre-defined limit.

It is checked using the value of gradient. If the gradient is small enough, that means we are very close to optimum and further iterations won’t have a substantial impact on coefficients. The lower-limit on gradient can be changed using the ‘**tol**‘ parameter.

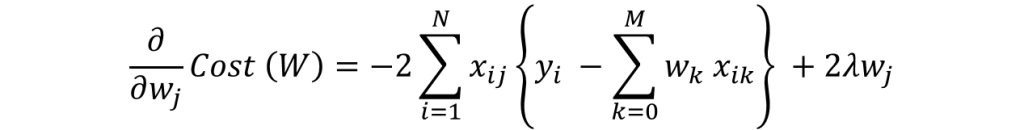
Lets consider the case of ridge regression now.

2. Ridge Regression

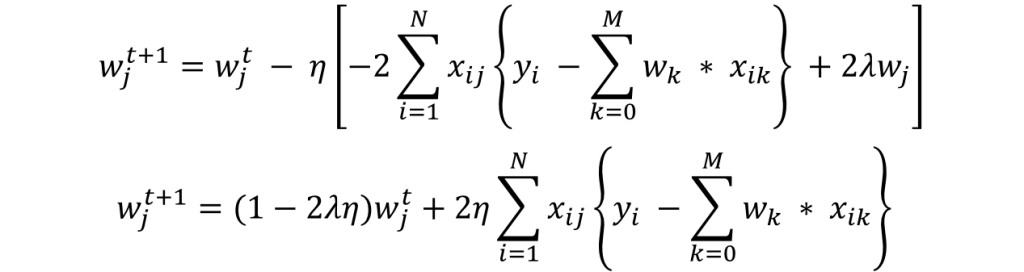
The objective function (also called the cost) to be minimized is the RSS(Residual sum of squares) plus the sum of square of the magnitude of weights. This can be depicted mathematically as:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq5-1.png)

In this case, the gradient would be:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq6-1.png)

Again in the regularization part of gradient, only wj remains and all other would become zero. The corresponding update rule is:

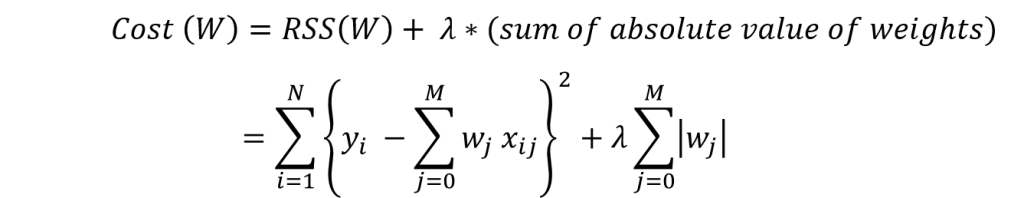
[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq7-1.png)

Here we can see that second part of the RHS is same as that of simple linear regression. Thus, ridge regression is equivalent to reducing the weight by a factor of (1-2λη) first and then applying the same update rule as simple linear regression. I hope this gives some intuition into why the coefficients get reduced to small numbers but never become zero.

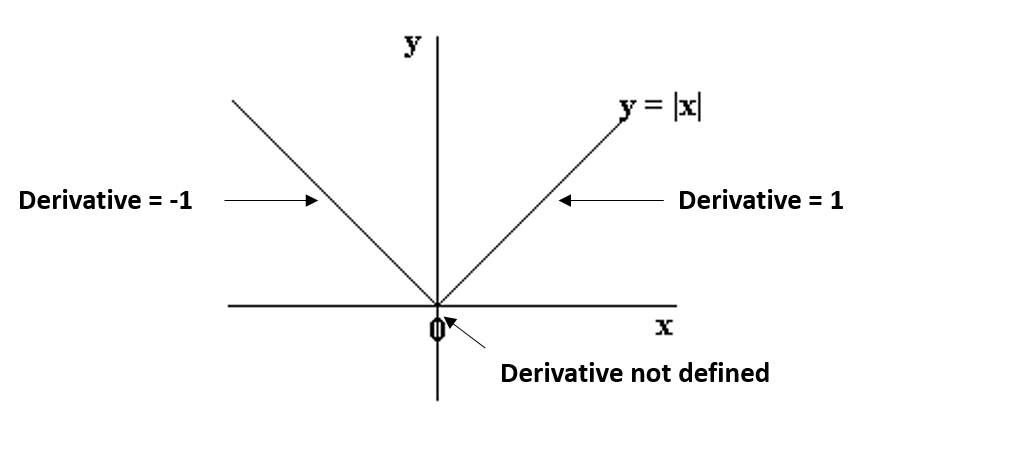
Note that the criteria for convergence in this case remains similar to simple linear regression, i.e. checking the value of gradients. Lets discuss Lasso regression now.

3. Lasso Regression

The objective function (also called the cost) to be minimized is the RSS plus the sum of absolute value of the magnitude of weights. This can be depicted mathematically as:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq8-1.png)

In this case, the gradient is not defined as the absolute function is not differentiable at x=0. This can be illustrated as:

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/fig1.png)

We can see that the parts on the left and right side of 0 are straight lines with defined derivates but the function can’t be differentiated at x=0. In this case, we have to use a different technique called as **coordinate descent** which is based on the concept of sub-gradients. One of the coordinate descent follows the following algorithms (this is also the default in sklearn):

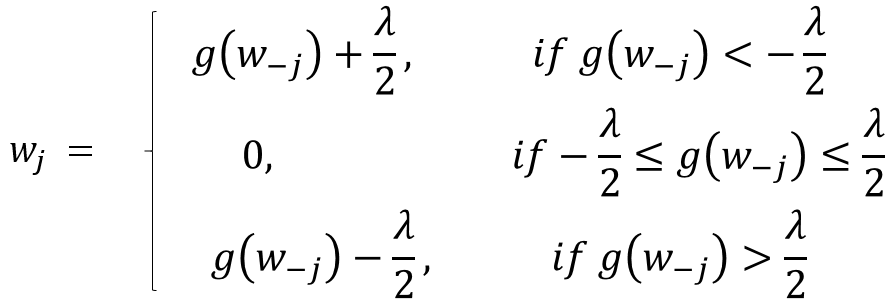
1. initialize weights (say w=0)

2. iterate till not converged

2.1 iterate over all features (j=0,1...M)

2.1.1 update the jth weight with a value which minimizes the cost

#2.1.1 might look too generalized. But I’m intentionally leaving the details and jumping to the update rule:

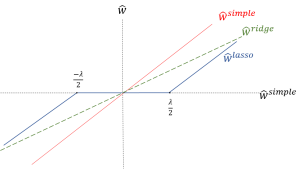
[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/eq9.png)

Here **g(w-j)** represents (but not exactly) the difference between actual outcome and the predicted outcome considering **all EXCEPT the jth variable**. If this value is small, it means that the algorithm is able to predict the outcome fairly well even without the jth variable and thus it can be removed from the equation by setting a zero coefficient. This gives us some intuition into why the coefficients become zero in case of lasso regression.

In coordinate descent, checking convergence is another issue. Since gradients are not defined, we need an alternate method. Many alternatives exist but the simplest one is to **check the step size of the algorithm**. We can check the maximum difference in weights in any particular cycle over all feature weights (#2.1 of algo above).

If this is lower than ‘tol’ specified, algo will stop. The convergence is not as fast as gradient descent and we might have to set the ‘max\_iter’ parameter if a warning appears saying that the algo stopped before convergence. This is why I specified this parameter in the Lasso generic function.

Lets summarize our understanding by comparing the coefficients in all the three cases using the following visual, which shows how the ridge and lasso coefficients behave in comparison to the simple linear regression case.

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/01/fig2.png)

Apologies for the lack of visual appeal. But I think it is good enough to re-inforced the following facts:

1. The ridge coefficients are a reduced factor of the simple linear regression coefficients and thus never attain zero values but very small values
2. The lasso coefficients become zero in a certain range and are reduced by a constant factor, which explains there low magnitude in comparison to ridge.

Before going further, one important issue in case of both ridge and lasso regression is **intercept handling**. Generally, regularizing the intercept is not a good idea and it should be left out of regularization. This requires slight changes in the implementation, which I’ll leave for you to explore.

If you’re still confused and things are a bit fuzzy, I recommend taking the course on Regression which is part of the [Machine Learning Specialization](https://www.coursera.org/specializations/machine-learning) by University of Washington at Coursera.

Now, lets come to the concluding part where we compare the Ridge and Lasso techniques and see where these can be used.

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?) As we observed earlier, some of the coefficients become exactly zero, which is equivalent to the particular feature being excluded from the model.

Traditionally, techniques like **stepwise regression** were used to perform feature selection and make parsimonious models. But with advancements in Machine Learning, ridge and lasso regression provide very good alternatives as they give much**better output**, require **fewer tuning parameters**and can be **automated** to a large extend.

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.

Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick   
while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s)   
can be continuous or discrete, and nature of regression line is linear.

Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X)   
using a best fit straight line (also known as regression line).

Ridge Regression

Ridge Regression is a technique used when the data suffers from multicollinearity ( independent variables are highly correlated).  
In multicollinearity, even though the least squares estimates (OLS) are unbiased, their variances are large which deviates the observed value   
far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.

Above, we saw the equation for linear regression. Remember? It can be represented as:

y=a+ b\*x

This equation also has an error term. The complete equation becomes:  
y=a+b\*x+e (error term), [error term is the value needed to correct for a prediction error between the observed and predicted value]  
=> y=a+y= a+ b1x1+ b2x2+....+e, for multiple independent variables.

In a linear equation, prediction errors can be decomposed into two sub components. First is due to the biased and second is due to the variance. Prediction error can occur due to any one of these two or both components. Here, we’ll discuss about the error caused due to variance.

Ridge regression solves the multicollinearity problem through shrinkage parameter λ (lambda). Look at the equation below.

Ridge

In this equation, we have two components. First one is least square term and other one is lambda of the summation of β2 (beta- square) where β is the coefficient. This is added to least square term in order to shrink the parameter to have a very low variance.

Important Points:  
•The assumptions of this regression is same as least squared regression except normality is not to be assumed  
•It shrinks the value of coefficients but doesn’t reaches zero, which suggests no feature selection feature  
•This is a regularization method and uses l2 regularization.

Lasso Regression

Similar to Ridge Regression, Lasso (Least Absolute Shrinkage and Selection Operator) also penalizes the absolute size of the regression coefficients.   
In addition, it is capable of reducing the variability and improving the accuracy of linear regression models.   
Look at the equation below: LassoLasso regression differs from ridge regression in a way that it uses absolute values in the penalty function,  
instead of squares. This leads to penalizing (or equivalently constraining the sum of the absolute values of the estimates)  
values which causes some of the parameter estimates to turn out exactly zero. Larger the penalty applied,  
further the estimates get shrunk towards absolute zero. This results to variable selection out of given n variables.

Important Points:  
•The assumptions of this regression is same as least squared regression except normality is not to be assumed  
•It shrinks coefficients to zero (exactly zero), which certainly helps in feature selection  
•This is a regularization method and uses l1 regularization  
•If group of predictors are highly correlated, lasso picks only one of them and shrinks the others to zero