Linear Regression

Shuvrajeet Das

May 7, 2021

Abstract

Linear regression is an important technique. Its basis is illustrated here, and various derived values such as the standard deviation from regression and the slope of the relationship between two variables are shown. The way to study residuals is given, as well as information to evaluate auto-correlation.

1 Introduction

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called simple linear regression; for more than one, the process is called multiple linear regression.

2 Explanation

2.1 MATHS BEHIND IT(normal method)

In linear regression, we obtain an estimate of the unknown variable (denoted by y; the output of our model) by computing a weighted sum of our known variables (denoted by x_i; the inputs) to which we add a bias term.

$$y = b + \sum_{i=1}^{n} x_i.w_i \tag{1}$$

where n is the number of data points.

Adding a bias is the same thing as imagining we have an extra input variable that's always 1 and using only the weights. We will consider this case to make the math notation a little easier.

$$y = \sum_{i=1}^{n} x_i.w_i \tag{2}$$

Where x_0 is always 1, and x_0 is our previous b.

For making it easier, we will transform it in matrix equation.

$$y = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \tag{3}$$

Now, the above equation is just for one data point. If we want to compute the outputs of more data points at once, we can concatenate the input rows into one matrix which we will denote by X. The weights vector will remain the same for all those different input rows and we will denote it by w. Now y will be used to denote a column-vector with all the outputs instead of just a single value.

$$y = Xw \tag{4}$$

To learn the weights, we need a dataset in which we know both x and y values, and based on those we will find the weights vector.

For obtaining weights the equation can be:

$$w = X^{-1}y \tag{5}$$

Most of the time the requirement for the solution above will not hold. Most of the time, our data points will not perfectly fit a line. There will be some random noise around our regression line, and we will not be able to obtain an exact solution for w. However, we will try to obtain the best possible solution for w so that the error is minimal.

This defines that y doesn't belong to the column space of X. So, instead of y, we will use the projection of y onto the column space of X. This is the closest vector to y that also belongs to the column space of X. If we multiply (on the left) both sides by the transpose of X, we will get an equation in which this projection is considered.

$$\begin{aligned} y &= Xw \\ X^T.| \quad y &= Xw \\ X^Ty &= X^TXw \\ (X^TX)^{-1}.| \quad X^Ty &= X^TXw \\ (X^TX)^{-1}X^Ty &= w \end{aligned} \tag{6}$$

i.e we get, $\boxed{w = (X^TX)^{-1}X^Ty}$

2.2 MATHS BEHIND IT(gradient descent method)

The equation of our line of best fit is given by this equation. We call this our hypothesis, represented by the function h(x).

$$h(x) = \sum_{i=0}^{n} \theta_i x_i \tag{7}$$

Where $x_0 = 1$,

Thus, in multivariate (dealing with multiple variable, x) linear regression, we must find a way to tweak these thetas to get a hyper plane that best represents the hyper plane of best fit. Here, θ_0 is the bias, in that it shifts the entire line by default. θ_1,θ_2,..,θ_n represents the weight, or how much does the input x impact the output y.

The mean squared error (cost function) finds the average squared difference between our predictions, and the actual y values in our training set. Our goal is to minimize the total cost.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2 \tag{8}$$

Summarising everything

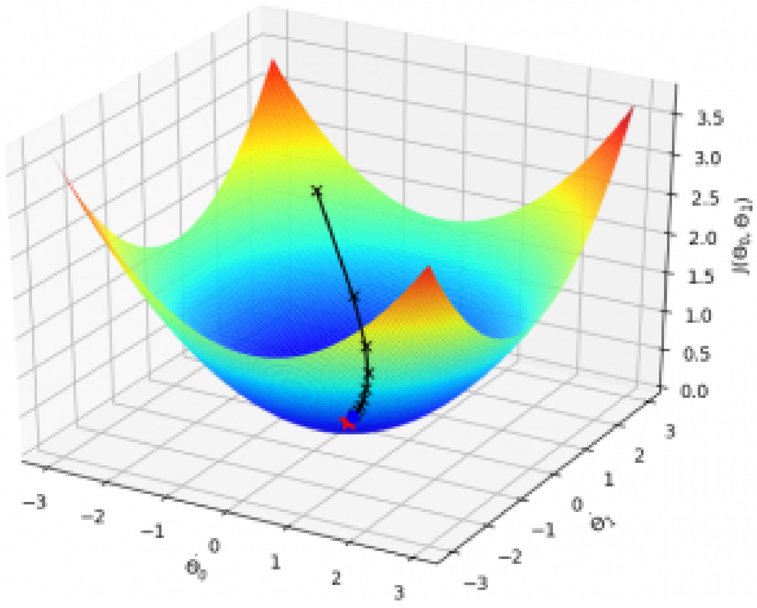
Hypothesis: $h(x) = \sum_{i=0}^{n} \theta_i x_i$
Parameters: θ
Cost Function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h(x_i) - y_i)^2$
Goal: minimize$J(\theta)$

Gradient Descent Algorithm:

Gradient descent is an iterative algorithm that randomly initializes somewhere on this plane. It then adjusts the values of θ_1 and θ_0 until it gets to a point on the plane where the cost function is low.

Intuitively, what gradient descent does is upon each iteration (each star on the graph) it ‘looks around a full 360 degrees’ and chooses the direction of steepest (most efficient) descent. It then takes a baby step in that direction, and reiterates again, until it converges — when it gets to a point where the cost can't go any lower. Below are the mathematical formulas that describe how these steps are taken (note that the formulas on the right are directly equivalent, but have the derivative term, the fraction thing, calculated).

Figure 1: Gradient Descent.

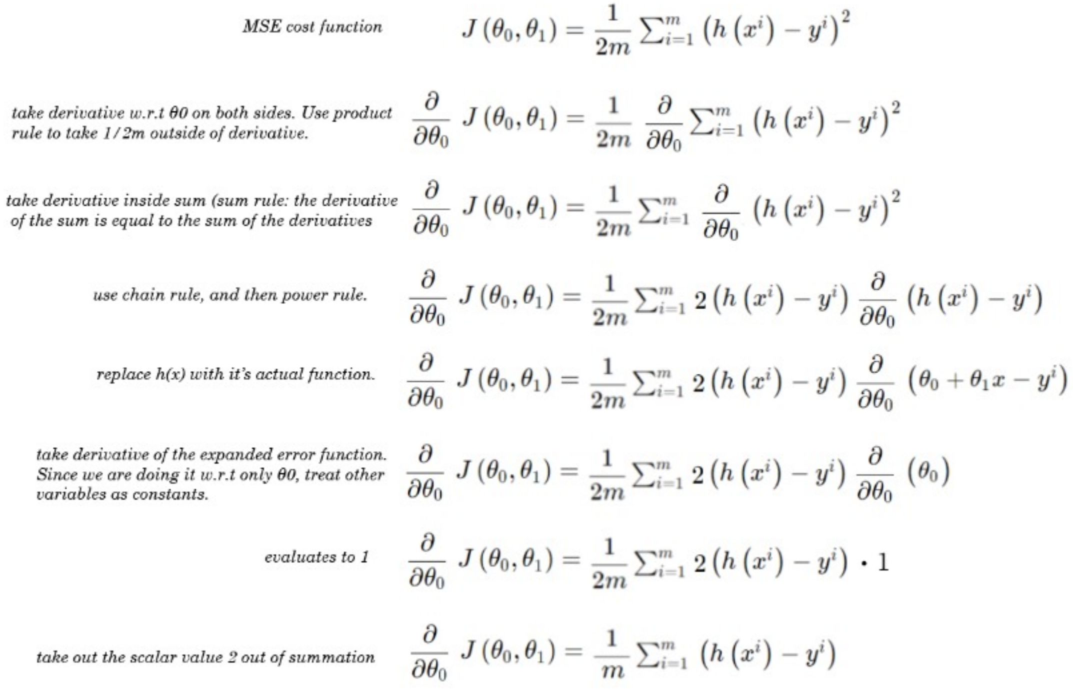


The algorithm can be defined as:

$$\begin{gathered}\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta) \\ \theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta) \\ \vdots \\ \theta_n := \theta_n - \alpha \frac{\partial}{\partial \theta_n} J(\theta)\end{gathered} \tag{9}$$

If we were to think of this as a vector operation, it could be adjusting a theta vector with θ by the negative gradient scaled by some learning rate α.

$$\begin{bmatrix}\theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m\end{bmatrix} - \begin{bmatrix}\frac{\partial}{\partial \theta_0} J(\theta) \\ \frac{\partial}{\partial \theta_1} J(\theta) \\ \frac{\partial}{\partial \theta_2} J(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} J(\theta)\end{bmatrix} \alpha \tag{10}$$



MSE cost function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h\left(x^i\right) - y^i\right)^2$$

take derivative w.r.t θ0 on both sides. Use product rule to take 1/2m outside of derivative.

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \frac{\partial}{\partial \theta_0} \sum_{i=1}^{m} \left(h\left(x^i\right) - y^i\right)^2$$

take derivative inside sum (sum rule: the derivative of the sum is equal to the sum of the derivatives

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_0} \left(h\left(x^i\right) - y^i\right)^2$$

use chain rule, and then power rule.

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} 2\left(h\left(x^i\right) - y^i\right) \frac{\partial}{\partial \theta_0} \left(h\left(x^i\right) - y^i\right)$$

replace h(x) with it's actual function.

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} 2\left(h\left(x^i\right) - y^i\right) \frac{\partial}{\partial \theta_0} \left(\theta_0 + \theta_1 x - y^i\right)$$

take derivative of the expanded error function. Since we are doing it w.r.t only θ0, treat other variables as constants.

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} 2\left(h\left(x^i\right) - y^i\right) \frac{\partial}{\partial \theta_0} \left(\theta_0\right)$$

evaluates to 1

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} 2\left(h\left(x^i\right) - y^i\right) \cdot 1$$

take out the scalar value 2 out of summation

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h\left(x^i\right) - y^i\right)$$

The derivation is pretty simple year 1 calculus stuff, except for the brief use of multivariable calculus in determining the second product after using the chain rule. This technically only derives the partial derivative segment of the overall update rule, but the rest.

3 Conclusion

Thus finding all the deriatives we can find all the weights and reach our minimum loss.