

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/340686790>

Detailed Derivation of The Linear Regression Model

Article · October 2019

CITATIONS

0

READS

2,015

1 author:



[Faris Alasmery](#)

King Fahd University of Petroleum and Minerals

2 PUBLICATIONS 2 CITATIONS

SEE PROFILE

Detailed Derivation of The Linear Regression Model

Faris Abdullah Alasmary

22nd of October 2019

1 Notation

Here are some notation conventions:

X = a **matrix** of features of all the examples in the dataset.

x = a **vector** of features of a single example.

$x^{(i)}$ = a **vector** that represents the i^{th} example from X .

$x_j^{(i)}$ = a **scalar** that represents the j^{th} input value (feature) of the i^{th} example vector from X .

For example,

$$X = \underbrace{\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}}_{m \times n}$$

where n is the number of input values (features) and m is the number of examples in the dataset.

If we select the 2nd example vector from the matrix X , the output will be:

$$x^{(2)} = \underbrace{\begin{bmatrix} x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \end{bmatrix}}_{1 \times n}$$

Finally, if we want to select the 3rd value from the vector $x^{(2)}$, then the result will be:

$$x_3^{(2)}$$

The equation of a line is typically written as:

$$y = mx + b \quad (1)$$

where m is the slope and b is the y -intercept. Similarly, we can say that the value of y in our dataset for a single example can be estimated as a linear combination of weighted features of that example, i.e.,

$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b = \left(\sum_{i=1}^n w_ix_i\right) + b \quad (2)$$

where n is the number of features of that example.

Sometimes the y -intercept b is written as w_0 and a new variable $x_0 = 1$ is associated with it. We can rewrite equation(2) as follows:

$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + w_0x_0 = \left(\sum_{i=1}^n w_ix_i\right) + w_0x_0 = \sum_{i=0}^n w_ix_i \quad (3)$$

If $n = 1$ then we will get back to the equation of a line

$$\hat{y} = w_1x_1 + w_0 = mx + b$$

From equation(2), if we put all x 's together in a single vector and all w 's together in another vector as shown below:

$$w = \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}}_{n \times 1} \quad x = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{n \times 1}$$

and by applying the *transpose* operation to the vector w , we will get the vector w^T as follows:

$$w = \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}}_{n \times 1}, \quad w^T = \underbrace{\begin{bmatrix} w_1 & w_2 & w_3 & \dots & w_n \end{bmatrix}}_{1 \times n}$$

we can rewrite equation(2) in the matrix form as follows:

$$\underbrace{\hat{y}}_{1 \times 1} = \underbrace{w^T \cdot x + b}_{1 \times 1} = \underbrace{\begin{bmatrix} w_1 & w_2 & w_3 & \dots & w_n \end{bmatrix}}_{1 \times n} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{n \times 1} = \underbrace{\sum_{i=1}^n w_i x_i + b}_{1 \times 1}$$

Likewise, we can rewrite equation(3) as shown below:

$$w = \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}}_{(n+1) \times 1} \quad x = \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{(n+1) \times 1}$$

and by applying the *transpose* operation to the vector w , we will get w^T vector as follows:

$$w = \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}}_{(n+1) \times 1} \quad w^T = \underbrace{\begin{bmatrix} w_0 & w_1 & w_2 & w_3 & \dots & w_n \end{bmatrix}}_{1 \times (n+1)}$$

Finally,

$$\underbrace{\hat{y}}_{1 \times 1} = \underbrace{w^T}_{1 \times 1} \cdot \underbrace{x}_{1 \times (n+1)} = \underbrace{\begin{bmatrix} w_0 & w_1 & w_2 & w_3 & \dots & w_n \end{bmatrix}}_{1 \times (n+1)} \cdot \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{(n+1) \times 1} = \underbrace{\sum_{i=0}^n w_i x_i}_{1 \times 1} \quad (4)$$

where n is the number of input values (features) and we added 1 to show that the vector size has increased since we added x_0 inside the vector. The same applies to vector w . The vector w is usually called the **weights vector**.

Note: the numbers under braces show the dimensions of matrices, i.e., *rows* \times *columns*.

Note: the vector is a special form of the matrix where the number of *rows* = 1 or the number of *columns* = 1. If the of *rows* = 1, this type of vectors is called a **row-vector**. On the other hand, if the number of *columns* = 1, it is called a **column-vector**.

2 Background

Let the matrix X be the features matrix and the vectors y and \hat{y} be the target values vector and the predicted values vector, respectively. Let the vector w be the weights vector (model). Suppose we have m different examples and each example x has n features that are arranged as a row-vector. Suppose we stack all examples over each other, we will get a matrix like:

$$X = \underbrace{\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}}_{m \times n}, \quad y = \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}}_{m \times 1}$$

Each row from the matrix X is associated with a value from the vector y of the same row. For example, the third example $x^{(3)}$ from the matrix X is associated with the value $y^{(3)}$ from the vector y .

In equation(3), we introduced a new variable x_0 to rewrite the equation of line(2) in the summation form. We will add x_0 to every example in the matrix X .

$$X = \underbrace{\begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ x_0^{(3)} & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}}_{m \times (n+1)}$$

In the previous matrix, we can replace x_0 with 1 to get the following matrix:

$$X = \underbrace{\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}}_{m \times (n+1)}$$

if we multiple the matrix X with the vector w , we will get \hat{y} value that approximates the target y value for each example in matrix X

$$\underbrace{\hat{y}}_{m \times 1} = \underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} = \underbrace{\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \dots & x_n^{(3)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_n^{(m)} \end{bmatrix}}_{m \times (n+1)} \cdot \underbrace{\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ \dots \\ w_n \end{bmatrix}}_{(n+1) \times 1} \quad (5)$$

the previous matrix multiplication can be written in details as:

$$\underbrace{\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(3)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix}}_{m \times 1} = \underbrace{\begin{bmatrix} w_0 + w_1x_1^{(1)} + w_2x_2^{(1)} + w_3x_3^{(1)} + \dots + w_nx_n^{(1)} \\ w_0 + w_1x_1^{(2)} + w_2x_2^{(2)} + w_3x_3^{(2)} + \dots + w_nx_n^{(2)} \\ w_0 + w_1x_1^{(3)} + w_2x_2^{(3)} + w_3x_3^{(3)} + \dots + w_nx_n^{(3)} \\ \vdots \\ w_0 + w_1x_1^{(m)} + w_2x_2^{(m)} + w_3x_3^{(m)} + \dots + w_nx_n^{(m)} \end{bmatrix}}_{m \times 1} = \underbrace{\begin{bmatrix} \sum_{i=0}^n w_i x_i^{(1)} \\ \sum_{i=0}^n w_i x_i^{(2)} \\ \sum_{i=0}^n w_i x_i^{(3)} \\ \vdots \\ \sum_{i=0}^n w_i x_i^{(n)} \end{bmatrix}}_{m \times 1}$$

We can see clearly that equation(5) is a generalization, written in the form of matrix, of equation(4). Each row in the last vector shows an equation of line corresponds to the same row in the X matrix.

Our Goal: we want to find the vector w , that is used in equation(5), such that the values of the vector \hat{y} are as close as possible to the values of the vector y .

One way to measure the "closeness" of two values is to subtract them from each other. The smaller the result is, the closer the two values are to each other. If the result is zero, it means that the two values are identical. The difference between a value of the vector y and the corresponding value of the same row in the vector \hat{y} is called the **residual error**. For example, if the values of the 4th row from the vectors y and \hat{y} are $y^{(4)} = 5$ and $\hat{y}^{(4)} = 1$, then the residual error is:

$$\text{residual error} = y^{(4)} - \hat{y}^{(4)} = 5 - 1 = 4$$

the same concept can be applied on the vectors and we will get a vector of residual errors. For example, if we have the two vectors y and \hat{y} with the following values:

$$y = \underbrace{\begin{bmatrix} 5 \\ 2 \\ 7 \end{bmatrix}}_{3 \times 1}, \quad \hat{y} = \underbrace{\begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix}}_{3 \times 1}$$

then the vector of residual errors is:

$$\underbrace{\text{residual error}}_{3 \times 1} = \underbrace{y - \hat{y}}_{3 \times 1} = \underbrace{\begin{bmatrix} 5 \\ 2 \\ 7 \end{bmatrix}}_{3 \times 1} - \underbrace{\begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix}}_{3 \times 1} = \underbrace{\begin{bmatrix} 2 \\ -2 \\ 0 \end{bmatrix}}_{3 \times 1}$$

we can find a single number (scalar) to represent the total error by summing up all the values in the vector of residual errors. This scalar is called the **residual sum**. The residual sum of the previous example is:

$$\underbrace{\text{residual sum}}_{1 \times 1} = 2 + (-2) + 0 = 0$$

the result shows that the total error equals 0 which is not true. The total error should be 4 since the difference between 5 and 3 is 2 and the difference between 2 and 4 is 2. Since the values are the same and the signs are opposite, the sum was 0. To solve this problem, we have either to take the sum of absolute values or to take the sum of squared values of all elements in the vector of residual errors. We will go with the second option since the square function is a differentiable function which is not the case with the absolute value function. Therefore, the new error vector will be as shown below after we apply **element-wise** squaring function:

$$\underbrace{\text{residual error}^2}_{1 \times 1} = \underbrace{(y - \hat{y})^2}_{3 \times 1} = \left(\underbrace{\begin{bmatrix} 5 \\ 2 \\ 7 \end{bmatrix}}_{3 \times 1} - \underbrace{\begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix}}_{3 \times 1} \right)^2 = \left(\underbrace{\begin{bmatrix} 2 \\ -2 \\ 0 \end{bmatrix}}_{3 \times 1} \right)^2 = \underbrace{\begin{bmatrix} 2^2 \\ -2^2 \\ 0^2 \end{bmatrix}}_{3 \times 1} = \underbrace{\begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix}}_{3 \times 1}$$

Finally, the new error scalar is called the **residual sum of squares (RSS)** and it will be:

$$\text{residual sum of squares} = 2^2 + (-2)^2 + 0^2 = 4 + 4 + 0 = 8$$

We can get the **sum of squares** of any vector by multiplying the transposed of that vector with the vector itself. For example, if we have the vector x and its transposed vector x^T

$$x^T = \underbrace{[x_1 \quad x_2 \quad x_3 \quad \dots \quad x_n]}_{1 \times n}, \quad x = \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{n \times 1}$$

the sum of squares will be:

$$\begin{aligned} \underbrace{\text{sum of squares}}_{1 \times 1} &= \underbrace{x^T}_{1 \times n} \cdot \underbrace{x}_{n \times 1} = \underbrace{[x_1 \quad x_2 \quad x_3 \quad \dots \quad x_n]}_{1 \times n} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}}_{n \times 1} \\ &= x_1 \cdot x_1 + x_2 \cdot x_2 + \dots + x_n \cdot x_n = \sum_{i=1}^n x_i^2 \end{aligned} \quad (6)$$

we can rewrite the residual sum of squares (RSS) using the same concept that is represented in equation(6)

$$\underbrace{RSS}_{1 \times 1} = \underbrace{\begin{bmatrix} 2 & -2 & 0 \end{bmatrix}}_{1 \times 3} \cdot \underbrace{\begin{bmatrix} 2 \\ -2 \\ 0 \end{bmatrix}}_{3 \times 1} = 2^2 + (-2)^2 + 0^2 = 4 + 4 + 0 = 8$$

Therefore, the general vectorized formula of **residual sum of squares (RSS)** can be derived using the same technique shown in equation(6) and using \hat{y} vector shown in equation(5)

$$\underbrace{\text{residual error}}_{m \times 1} = \underbrace{\text{target values}}_{m \times 1} - \underbrace{\text{predicted values}}_{m \times 1} = \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix}}_{m \times 1} - \underbrace{\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(3)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix}}_{m \times 1} = \underbrace{\begin{bmatrix} y^{(1)} - \hat{y}^{(1)} \\ y^{(2)} - \hat{y}^{(2)} \\ y^{(3)} - \hat{y}^{(3)} \\ \vdots \\ y^{(m)} - \hat{y}^{(m)} \end{bmatrix}}_{m \times 1} = \underbrace{\begin{bmatrix} e^{(1)} \\ e^{(2)} \\ e^{(3)} \\ \vdots \\ e^{(m)} \end{bmatrix}}_{m \times 1}$$

$$\underbrace{RSS}_{1 \times 1} = \underbrace{e^T}_{1 \times m} \cdot \underbrace{e}_{m \times 1} \quad (7)$$

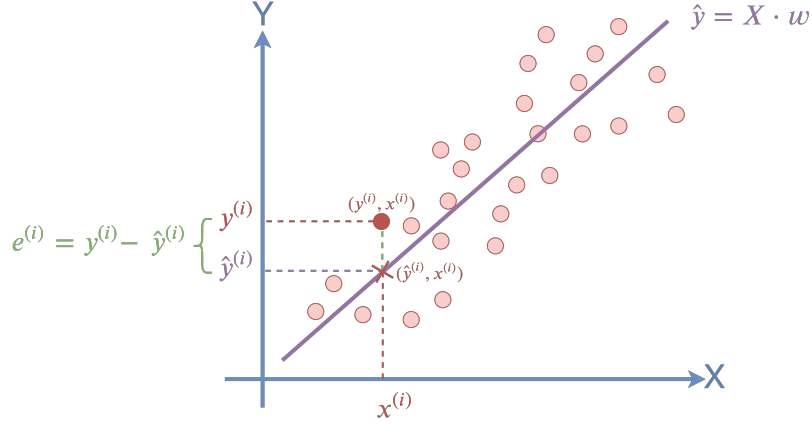


Figure 1: the error of the i^{th} example

3 Mathematical Derivation

We will use equation(7) that we developed previously to find the optimal weights vector w that minimizes the residual error. In this section, the variable m refers to the number of examples in the dataset and the variable n shows the number of features of each example.

Note: To keep track of matrices dimensions, we will show the dimensions under each term.

The dimensions of residual sum of squares (RSS) are 1×1 since we know that it is a scalar number.

$$\underbrace{RSS}_{1 \times 1} = \underbrace{e^T}_{1 \times m} \cdot \underbrace{e}_{m \times 1}$$

and we know from equation(7) that

$$\underbrace{e}_{m \times 1} = \underbrace{y}_{m \times 1} - \underbrace{\hat{y}}_{m \times 1} = \underbrace{(y - \hat{y})}_{m \times 1}$$

hence,

$$\underbrace{RSS}_{1 \times 1} = \underbrace{e^T}_{1 \times m} \cdot \underbrace{e}_{m \times 1} = \underbrace{(y - \hat{y})^T}_{1 \times m} \cdot \underbrace{(y - \hat{y})}_{m \times 1} \quad (8)$$

Also, we know from equation(5) that

$$\underbrace{\hat{y}}_{m \times 1} = \underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} = \underbrace{(X \cdot w)}_{m \times 1}$$

by substituting \hat{y} with $(X \cdot w)$ in equation(8)

$$\underbrace{RSS}_{1 \times 1} = \underbrace{\left(\underbrace{y}_{m \times 1} - \underbrace{\left(\underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)}_{m \times 1} \right)}_{1 \times m}^T \cdot \underbrace{\left(\underbrace{y}_{m \times 1} - \underbrace{\left(\underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)}_{m \times 1} \right)}_{m \times 1} \quad (9)$$

From equation(9), by distributing the transpose operation over the terms inside parentheses and applying the transpose operation rules

$$\underbrace{RSS}_{1 \times 1} = \underbrace{\left(\underbrace{y^T}_{1 \times m} - \underbrace{\left(\underbrace{w^T}_{1 \times (n+1)} \cdot \underbrace{X^T}_{(n+1) \times m} \right)}_{1 \times m} \right)}_{1 \times m} \cdot \underbrace{\left(\underbrace{y}_{m \times 1} - \underbrace{\left(\underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)}_{m \times 1} \right)}_{m \times 1} \quad (10)$$

and by distributing sums over products, i.e., multiplying each term from the first operand of the product to all terms in the second operand of the product and sum them up

$$\begin{aligned} \underbrace{RSS}_{1 \times 1} &= \underbrace{\left(\underbrace{y^T}_{1 \times m} \cdot \underbrace{y}_{m \times 1} \right)}_{1 \times 1} - \underbrace{\left(\underbrace{y^T}_{1 \times m} \cdot \underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)}_{1 \times 1} - \underbrace{\left(\underbrace{w^T}_{1 \times (n+1)} \cdot \underbrace{X^T}_{(n+1) \times m} \cdot \underbrace{y}_{m \times 1} \right)}_{1 \times 1} \\ &\quad + \underbrace{\left(\underbrace{w^T}_{1 \times (n+1)} \cdot \underbrace{X^T}_{(n+1) \times m} \cdot \underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)}_{1 \times 1} \quad (11) \end{aligned}$$

Since the transpose of a scalar is the same scalar and by taking the transpose of the term

$$\underbrace{\left(\underbrace{y^T}_{1 \times m} \cdot \underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)}_{1 \times 1} = \underbrace{\left(\underbrace{y^T}_{1 \times m} \cdot \underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)^T}_{1 \times 1} = \underbrace{\left(\underbrace{w^T}_{1 \times (n+1)} \cdot \underbrace{X^T}_{(n+1) \times m} \cdot \underbrace{y}_{m \times 1} \right)}_{1 \times 1}$$

hence; we can simplify equation(11) by plugging in the previous result in

$$\begin{aligned} \underbrace{RSS}_{1 \times 1} &= \underbrace{\left(\underbrace{y^T}_{1 \times m} \cdot \underbrace{y}_{m \times 1} \right)}_{1 \times 1} - 2 \cdot \underbrace{\left(\underbrace{w^T}_{1 \times (n+1)} \cdot \underbrace{X^T}_{(n+1) \times m} \cdot \underbrace{y}_{m \times 1} \right)}_{1 \times 1} \\ &+ \underbrace{\left(\underbrace{w^T}_{1 \times (n+1)} \cdot \underbrace{X^T}_{(n+1) \times m} \cdot \underbrace{X}_{m \times (n+1)} \cdot \underbrace{w}_{(n+1) \times 1} \right)}_{1 \times 1} \end{aligned} \quad (12)$$

For simplicity, all underbraces will be removed from the previous equation

$$RSS = y^T \cdot y - 2 \cdot (w^T \cdot X^T \cdot y) + (w^T \cdot X^T \cdot X \cdot w) \quad (13)$$

Recall that the RSS is a function that represents the sum of the difference between the vectors y and \hat{y} , i.e., the total error. We want to find the best values for the vector w that minimizes the RSS error. To do that, we will take the partial derivative of the RSS error function with respect to each element in the vector w and set it to zero. The vector of all partial derivatives is called the **gradient** and it is denoted with the symbol ∇_w .

$$\nabla_w RSS = \underbrace{\begin{bmatrix} \frac{\partial RSS}{\partial w_0} \\ \frac{\partial RSS}{\partial w_1} \\ \frac{\partial RSS}{\partial w_2} \\ \vdots \\ \frac{\partial RSS}{\partial w_{(n+1)}} \end{bmatrix}}_{(n+1) \times 1}$$

3.1 The Normal Equation (Closed-Form Solution)

We will calculate the gradient of the RSS function with respect to the vector w and set it to zero.

$$\nabla_w RSS = \nabla_w (y^T \cdot y - 2 \cdot (w^T \cdot X^T \cdot y) + (w^T \cdot X^T \cdot X \cdot w)) = 0 \quad (14)$$

Since the derivative of a sum is the sum of the derivatives, we can rewrite equation(14) as follows:

$$\nabla_w RSS = \nabla_w (y^T \cdot y) - 2 \cdot \nabla_w (w^T \cdot X^T \cdot y) + \nabla_w (w^T \cdot X^T \cdot X \cdot w) = 0 \quad (15)$$

The gradient of the first term of equation(15) is zero since it is a constant term with respect to the vector w . The gradient of the second term is $-2 \cdot (X^T \cdot y)$. The last term is in a form called the **quadratic form**. The following equations show the quadratic form and its gradient, respectively.

$$\begin{aligned} f(B) &= B^T \cdot A \cdot B \\ \nabla_B f &= 2 \cdot A \cdot B \end{aligned} \quad (16)$$

Let $A = X^T \cdot X$ and $B = w$; hence,

$$\nabla_w (w^T \cdot X^T \cdot X \cdot w) = 2 \cdot X^T \cdot X \cdot w$$

and by substituting all gradients back into the $\nabla_w RSS$ equation(15), we will get:

$$\nabla_w RSS = 0 - 2 \cdot (X^T \cdot y) + (2 \cdot X^T \cdot X \cdot w) = 0 \quad (17)$$

$$(2 \cdot X^T \cdot X \cdot w) = 2 \cdot (X^T \cdot y)$$

$$\begin{aligned} X^T \cdot X \cdot w &= X^T \cdot y \\ w &= (X^T \cdot X)^{-1} \cdot X^T \cdot y \end{aligned} \tag{18}$$

The previous equation(18) computes the best values of the vector w . This solution is called the **closed-form solution** or the **normal equation**.

3.2 The Gradient Descent (Iterative Solution)

We can see from the normal equation(18) that we need to compute the inverse matrix of $X^T \cdot X$ which is too expensive to compute for large matrices since the time complexity of the inverse matrix algorithm is $O(n^2 \cdot m)$ (assuming that $m > n$ and the used algorithm is Moore–Penrose pseudoinverse algorithm). The gradient descent algorithm is widely used to compute the vector w because it is much cheaper to compute compared with the normal equation.

3.2.1 The Update Rule

Let w_{old} represents the current values of the vector w and w_{new} represents the updated values of the vector w after running gradient descent algorithm for one iteration. Also, let Δw be the vector that represents the difference between the vectors w_{new} and w_{old} as follows:

$$\underbrace{\Delta w}_{(n+1) \times 1} = \underbrace{w_{new}}_{(n+1) \times 1} - \underbrace{w_{old}}_{(n+1) \times 1}$$

we can also say that:

$$\underbrace{w_{new}}_{(n+1) \times 1} = \underbrace{w_{old}}_{(n+1) \times 1} + \underbrace{\Delta w}_{(n+1) \times 1}$$

There are basically two main steps to do in gradient descent algorithm:

Step 1: Compute the gradient of the $RSS(w_{old})$ function

Step 2: Update the current w_{old} vector such that:

$$RSS(w_{old} + \Delta w) < RSS(w_{old})$$

Explanation: The second step says that we want to take a small step Δw in some direction such that the new place that we land in, $w_{new} = w_{old} + \Delta w$, causes the RSS error function evaluated at $w_{new} = w_{old} + \Delta w$ to be **less than**

the previous evaluation of the function at w_{old} . This means that each step Δw that the gradient descent algorithm takes should minimize the *RSS* error function.

For the first step, we will use equation(17) that we derived previously to compute the gradient of the RSS function with respect to the vector w . For the second step, we will use the Taylor series expansion of the function *RSS* to derive the **update rule** that is used in the gradient descent algorithm. Recall that the Taylor series expansion for a univariate function $f(x)$ is defined as follows:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots$$

where $f(x)$ is infinitely differentiable at the real number a .

The multivariate version of the Taylor series is defined as follows:

$$f(x) = f(a) + \frac{1}{1!}\nabla_a f(a)^T(x-a) + \frac{1}{2!}(x-a)^T H_a(x-a) + \dots$$

where $\nabla_a f(a)$ is the gradient of the function $f(x)$ evaluated at the vector a and H_a is the Hessian matrix evaluated at the same vector a .

If we select the first two terms, then we will get

$$f(x) \approx f(a) + \nabla_a f(a)^T(x-a) \quad (19)$$

This is called the **first-order** Taylor series approximation.

Now, we want to approximate the error function *RSS* at the vector w_{old} . We will replace the function and the variables as follows:

$$x = w_{new}$$

$$a = w_{old}$$

$$\Delta w = w_{new} - w_{old}$$

$$w_{new} = w_{old} + \Delta w$$

$$f = RSS$$

and by substituting the previous variables in equation(19), we will get

$$RSS(w_{new}) \approx RSS(w_{old}) + \nabla_{w_{old}} RSS(w_{old})^T (w_{new} - w_{old})$$

which can also be written as:

$$\underbrace{RSS(w_{old} + \Delta w)}_{1 \times 1} \approx \underbrace{RSS(w_{old})}_{1 \times 1} + \underbrace{\nabla_{w_{old}} RSS(w_{old})^T}_{1 \times (n+1)} \cdot \underbrace{\Delta w}_{(n+1) \times 1} \quad (20)$$

$\underbrace{\hspace{10em}}_{1 \times 1}$

To make sure that equation(20) satisfies the condition:

$$RSS(w_{old} + \Delta w) < RSS(w_{old})$$

we can see that the term $RSS(w_{old})$ is always **positive** since it represents the sum of squared errors which will never be negative. Therefore, the term $\nabla_{w_{old}} RSS(w_{old})^T \Delta w$ must be **negative** to ensure that the RSS error value decreases after each iteration.

$$RSS(w_{old} + \Delta w) \approx \underbrace{RSS(w_{old})}_{\text{always positive}} + \underbrace{\nabla_{w_{old}} RSS(w_{old})^T \Delta w}_{\text{must be negative}}$$

Since $\nabla_{w_{old}} RSS(w_{old})^T \cdot \Delta w$ is the inner product of the vectors $\nabla_{w_{old}} RSS(w_{old})$ and Δw and from the inner product properties, we can rewrite the previous expression as follows:

$$\nabla_{w_{old}} RSS(w_{old})^T \Delta w = |\nabla_{w_{old}} RSS(w_{old})| |\Delta w| \cos(\theta)$$

where θ is the angle between the vectors $\nabla_{w_{old}} RSS(w_{old})$ and Δw .

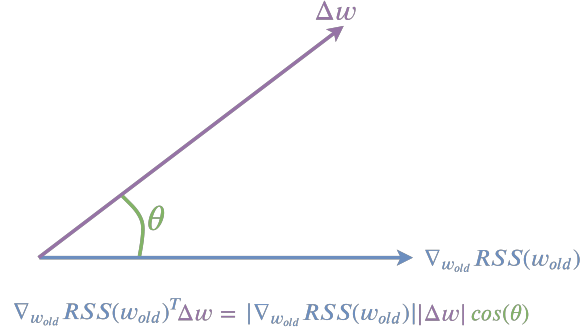


Figure 2: the inner product formula

If we further manipulate the previous equation, we will get the following:

$$\cos(\theta) = \frac{\nabla_{w_{old}} RSS(w_{old})^T \Delta w}{|\nabla_{w_{old}} RSS(w_{old})| |\Delta w|} \quad (21)$$

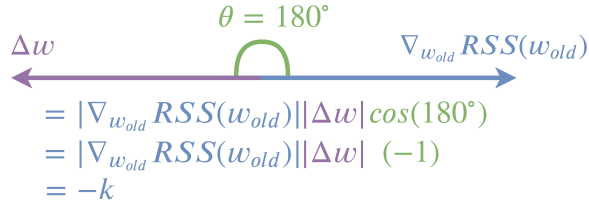
and we know that $\cos(\theta)$ is bounded between -1 and 1:

$$-1 \leq \cos(\theta) \leq 1 \quad (22)$$

for simplicity, let $k = |\nabla_{w_{old}} RSS(w_{old})| |\Delta w|$. If we substitute the value of $\cos(\theta)$ from equation(21) into equation(22) and multiply by k , we will get the following equation:

$$-k \leq \nabla_{w_{old}} RSS(w_{old})^T \Delta w \leq k \quad (23)$$

We can see clearly from equation(23) that the maximum value we can reach is k that corresponds to 1 in equation(22) when the angle $\theta = \cos^{-1}(1) = 0^\circ$. On the other hand, the minimum value is $-k$ that corresponds to -1 in equation(22) when the angle $\theta = \cos^{-1}(-1) = 180^\circ$. Since we want the value to be as small as possible, we should take $-k$ which means that the angle between the vectors $\nabla_{w_{old}} RSS(w_{old})$ and Δw must be $\theta = \cos^{-1}(-1) = 180^\circ$.



$$\begin{aligned} & \Delta w \quad \theta = 180^\circ \quad \nabla_{w_{old}} RSS(w_{old}) \\ & = |\nabla_{w_{old}} RSS(w_{old})| |\Delta w| \cos(180^\circ) \\ & = |\nabla_{w_{old}} RSS(w_{old})| |\Delta w| (-1) \\ & = -k \end{aligned}$$

Figure 3: the inner product formula

One of the vectors properties is that when you multiply a vector with -1 you will get the same vector pointing to the opposite direction.

From the previous findings, we conclude that the vector Δw should point to the opposite direction of the vector $\nabla_{w_{old}} RSS(w_{old})$, i.e.;

$$\Delta w = -\nabla_{w_{old}} RSS(w_{old})$$

if we replace Δw with its value $w_{new} - w_{old}$, then we will get our **update rule**:

$$\begin{aligned} w_{new} - w_{old} &= -\nabla_{w_{old}} RSS(w_{old}) \\ w_{new} &= w_{old} - \nabla_{w_{old}} RSS(w_{old}) \end{aligned} \quad (24)$$

3.2.2 The Learning Rate

Let's depict adding the two vectors $\nabla_{w_{old}} RSS(w_{old})$ and w_{old} .

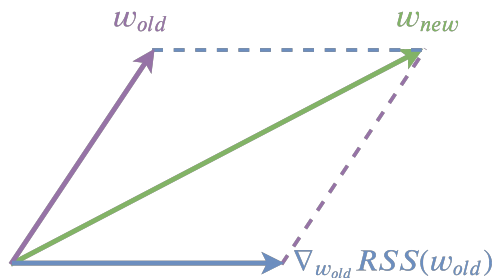


Figure 4: the resultant vector

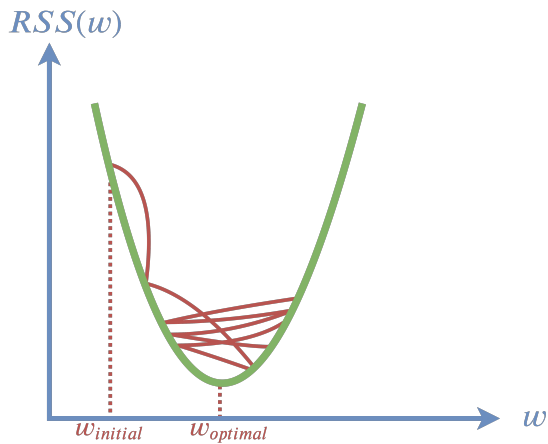


Figure 5: gradient overshooting

It can be seen clearly that the result of the addition is a vector of a larger magnitude compared to the original operands. This will cause the gradient to overshoot the minimum and diverge. To solve this problem, we need to multiply the gradient with a small scalar number $0 < \alpha < 1$ to shrink the resultant vector magnitude.

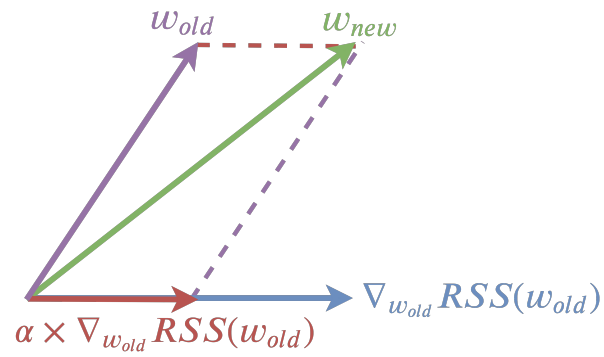


Figure 6: the resultant vector after multiplying by α

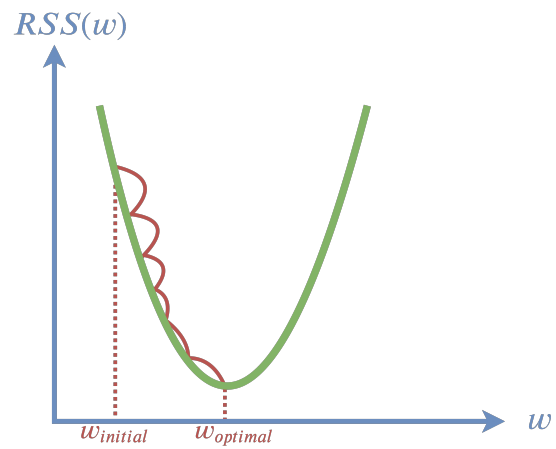


Figure 7: reaching the optimal weights after using a good learning rate

Finally, if we put everything together, we will get the gradient descent algorithm.

Algorithm 1 Gradient Descent Algorithm

w_{old} = a randomly initialized vector of size $(n + 1) \times 1$

w_{new} = a vector of zeros of size $(n + 1) \times 1$

α = the learning rate

ϵ = tolerance

while $\|w_{new} - w_{old}\|_2 > \epsilon$ **do**

$w_{old} = w_{new}$

$\nabla_{w_{old}} RSS = X^T \cdot X \cdot w_{old} - X^T \cdot y$

$w_{new} \leftarrow w_{old} - \alpha \cdot \nabla_{w_{old}} RSS$

end while

3.3 The Implementation Code

You can find the implementation of this paper on my github account:

<https://github.com/farisalasmaary/linear-regression-numpy/>

References

- [1] Shahbaz Khan, *Mathematical Intuition behind Gradient Descent*. 2019.
”<https://towardsdatascience.com/mathematical-intuition-behind-gradient-descent-flb959a59e6d>”
- [2] Michael Orlitzky, *The derivative of a quadratic form*.
”http://michael.orlitzky.com/articles/the_derivative_of_a_quadratic_form.xhtml”
- [3] Wikipedia, *Moore–Penrose inverse*.
”https://en.wikipedia.org/wiki/Moore–Penrose_inverse”
- [4] Wikipedia, *Taylor series*.
”https://en.wikipedia.org/wiki/Taylor_series”