Application of Linear Algebra in Multiple Linear Regression.

Thiago Marques Martins

Faculty of Engineering, Rio de Janeiro State University, Rio de Janeiro, Brazil, thiagoann@gmail.com.

Abstract. It is often common for scientists to come across a huge set of data with many variables after conducting a certain experiment and their objective is to analyse and conclude afterwards. However, these results are many times nonlinear, and a definitive answer is not always straightforward. The goal of this study was to discuss the application of Linear Algebra in Multiple Linear Regression models, thus problems like this mentioned in the beginning can be solved in a much faster way using matrices and their operations. Additionally, the knowledge of computer programming can also be of great advantage, for some programming languages, such as Python, already have most of the methods implemented, therefore making the entire process much faster. Based on that, a set of data was extracted and, using both techniques described, it was possible to combine them and build a model in which our data fitted the dotted line perfectly. This indicates an excellent model and shows us that Linear Algebra can be an extremely useful tool when it comes to building a good linear model, applying our prior knowledge of mathematics plus coding, hence, obtaining our result.

Keywords. Linear Algebra, Multiple Linear Regression, Least Square Estimators, Python, matrices, residuals, predictions, Coefficient of Determination

1. Introduction

Multiple Linear Regression consists of finding a linear relationship between a set of data containing the features, x variables, and the targets, y variable. In order to establish this relationship, we need to find the least square estimators. To do that, we would normally work with systems of equations, which can be exceedingly difficult when working with a large set of data. On the other hand, we will make use of Linear Algebra, a mathematical tool that uses matrices and their operations to solve this type of problem.

2. Research Methods

The study was conducted by using a set of data from a textbook [1]. A sample of it can be seen in Table 1. By developing a Python program [2] using the concepts of Linear Algebra, the results were obtained. Columns *x1*, *x2* and *x3* correspond to three different semiconductors, respectively: Emitter-RS, Base-RS and Emitter-to-Base (RS). As for the *y* column, it will store the values for the device HFE [1].

Tab. 1 – Semiconductor Data

y	x1	x2	x3
128.4	14.62	226	7
52.62	15.63	220	3.375
113.9	14.62	217.4	6.375
98.01	15	220	6
139.9	14.5	226.5	7.625

In subsection 2.1, it is the description of the method to find the Multiple Linear Regression equation.

2.1 Multiple Linear Regression Equation

A thorough review of the theory was helpful to understand the whole concept of applying Linear Algebra in Multiple Linear Regression. The problem comes down to two simple equations, the first one describing how the model will look like, according to Equation 1, and the other one showing the formula to find the beta vector, according to Equation (3). They summarize our whole problem.

Equation (1) is the basis of Multiple Linear

Regression, which $\beta 1, \beta 2, \beta 3, \ldots, \beta p$ correspond to the parameters, and ε is the residuals. In statistical problems, this last one is quite common to appear, so we must take it into account. As for Equation (2), it is a simplified version of Equation 1, for it shows how our model will look like when putting it into a matrix form. To find $\hat{\beta}$, we need to minimize the residuals by applying the first order condition, but this is something that will not be covered in this paper. To see more about it, have a look at [3].

$$Y = \beta 0 + \beta 1 * x1 + \beta 2 * x2 + \ldots + \beta p * xp + \epsilon \quad (1)$$

$$Y = \hat{\beta} X + \epsilon \quad (2)$$

This is when Equation (3), called the Normal Equation, is going to show up. It tells us how to find the beta vector and the way to do it is applying matrix operations, such as transposition and inversion, as well as other ones. We shall take a careful look at how it works.

$$\hat{\beta} = (X^T X)^{-1} X^T Y \quad (3)$$

A matrix and a vector will be needed for this problem, X and Y, respectively. For the model to work, X must have more rows than columns. It is called the model matrix, and it will have fixed values, which will be the independent ones, whereas vector Y is the vector of targets, which contains the dependent values, and they are random. Notice that an extra column containing only ones was included in our matrix X. According to [4], this is because our function $\beta 0 + \beta 1 * x1 + \beta 2 * x2 + \ldots + \beta p * xp$ is linear in β, but not in x, due to $\beta 0$. By adding the extra column, our model becomes linear in the feature space. In our problem, X will hold the values corresponding to *x1*, *x2* and *x3*, and vector Y the values for the *y*.

Now that we have seen a full description of how to get to the Multiple Linear Regression equation, we will learn step by step which operations are done in Equation 3.

1. X^T – transposition of matrix X, i.e., the rows and columns are swapped.
2. $X^T X$ – matrices X^T and X are multiplied obeying the rules of matrix multiplication.
3. $(X^T X)^{-1}$ – the previous operation will result in a new matrix which will be inversed.
4. $X^T Y$ – same thing done in step 2, except now matrices X^T and Y are multiplied.
5. The matrices obtained in steps 3 and 4 will be multiplied thus resulting in the beta vector.

We have finished analysing the entire process, including the use of matrix operations with the purpose of finding our least square estimators. We will now have a look at how our results look like, applying this process in a Python code carefully developed.

3. Discussion and Results

3.1 The Code

All the process was done in a worldwide popular programming language, Python. Now, Python has many built-in functions and modules that could help us solve our problem in very few lines. Instead, my intention was to develop the complete process from scratch, only using the module NumPy to multiply our matrices and solve our equation. A few parts of the code will be shown below, along with their explanations. As it has already been said, the rest of the it can be seen on [2].

Part 1 – Adding the extra column with just ones:

```
X = np.insert(X,0,1,axis=1)
```

Part 2 – the Normal Equation and how it was done:

```
'''

THIS IS WHERE WE'LL BE APPLYING OUR ALGORITHM USING LINEAR ALGEBRA

WITH THE HELP OF PYTHON LIBRARY NUMPY.

- X_TRANSPOSE = X.T => TRANSPOSES MATRIX X

- XtX = np.dot(X_transpose,X) => FUNCTION dot() DOES THE MATRIX MULTIPLICATION

- THE SAME THING FOR XtY

- beta = np.linalg.solve(XtX,XtY) => FUNCTION SOLVE() SOLVES A MATRIX EQUATION

JUST LIKE THE ONE WE HAVE IN OUR PROBLEM AND RETURNS THE BETA VECTOR

'''

X_transpose = X.T

XtX = np.dot(X_transpose,X)

XtY = np.dot(X_transpose,y)

beta = np.linalg.solve(XtX,XtY)

print(f'y = {beta[0]} + {beta[1]}*x1 + {beta[2]}*x2 + {beta[3]}*x3')
```

3.2 Least Square Estimators

Having written the code and run it, it will return the following coefficients: $\beta 0 = 47.17$, $\beta 1 = -9.74$, $\beta 2 = 0.43$ and $\beta 3 = 18.24$. Therefore, our equation will be:

$$y = 47.17 - 9.74 * x1 + 0.43 * x2 + 18.24 * x3 \quad (4)$$

Equation (4) is our Multiple Linear Regression final equation. These values, when put inside the beta

vector, are called the least square estimators and they will be named $\hat{\beta}\,0, \hat{\beta}\,1, \hat{\beta}\,2, \ldots, \hat{\beta}\,p$. They must obey two particularly important properties. The first one is regarding the expected value and the other has to do with the covariance. We shall look at them further in this article.

3.3 Graph

Other than looking at numbers, it is also a great idea to see how the result looks graphically. This allows us to have a visual representation of our model. Notice that the data points, in red, are remarkably close to the fitted line, in green, which is an excellent sign. If the points were scattered, that is, far from the straight line, this would mean that the model is not good, therefore changes would have to be made. Figure 1 below shows the linear correlation between the semiconductors and the device. The red points are plotted based on the relationship between the predicted values and the real ones. As for the dashed green line, it fits these plots.

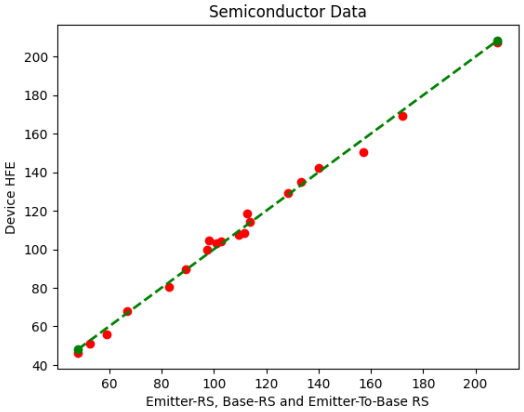

Fig. 1 – Semiconductor Data – Multiple Linear Regression

We have seen the Multiple Linear Regression equation and how it looks like in a linear graph. We will now analyse our residuals, the properties of the least square estimators, the Coefficient of Determination and predictions that could be made using our model.

3.4 Residuals

The difference between the observation y_i and the fitted value $\hat{y}_i$ is a residual, say, $e_i = y_i - \hat{y}_i$ [1]. As well as the graph seen above, residuals are also an excellent way to see how good our model is. Remember that our goal in Multiple Linear Regression is to minimize our residuals, so that we have the best model. Table 2 shows five residuals. The rest can be seen in the Python code on [2].

Tab.2 - Residuals

Residuals
-0.9003915
1.832658673
-0.31871551
-6.7838394
-2.18117015

3.4.1 Estimating σ^2

Regarding the residuals, there is one parameter related to it that is interesting to analyse and that is their variance, which we normally call σ^2(sigma-squared). Now, σ^2 is an unbiased estimator, which means that the expected value of the parameter is equal to its real one. To calculate it, we first need to calculate the Sum of Squares Errors (*SSE*). When dealing with statistics, we apply the following formula:

$$SSE = \sum_{I=1}^{n} \epsilon^2 \quad (5)$$

On the other hand, we will use the same matrix operations as seen previously to find the beta vector. Equation (5) can be looked at as the multiplication between the residuals vector transposed and its own. Knowing that $\boldsymbol{\epsilon}$ is $\boldsymbol{Y} - \hat{y}$ and that is equal to y-$X\widehat{\boldsymbol{\beta}}$, the result will be:

$$SSE = Y^T Y - \hat{\beta}\, X^T Y \quad (6)$$

Equation 6 is just as the same as Equation 5 but working with matrices. After doing this, we divide the result by n-p, which n equals the number of columns we have, and p equals the number of parameters βk. Given n=20 and p = 4 ($\beta 0, \beta 1, \beta 2$, β3), Equation (7) gives us our result.

$$\sigma^2 = \frac{SSE}{n-p} = 12.108 \quad (7)$$

Therefore, our variance of the residuals will be 12.108.

3.5 Properties of the Least Square Estimators

3.5.1 Expected Value (E)

In probability and statistics, an expected value is the mean value of a range of data. In Multiple Linear Regression, we want the expected value of our least square estimators equal to the regression coefficient. Now, when this happens, we say that our estimator is unbiased. The demonstration below is based on the same one seen in [1] and uses Equation 3 to do it. In the end, it really shows that the expected value of the least square estimators is, in fact, unbiased estimators of the regression coefficients.

$$E(\hat{\beta}) = E[((X^T X)^{-1} X^T Y)] \quad (8)$$
$$E(\hat{\beta}) = E[((X^T X)^{-1} X^T (\hat{\beta} X + \epsilon)] \quad (8.1)$$
$$E(\hat{\beta}) = E[(X^T X)^{-1} X^T X\, \beta + (X^T X)^{-1} X^T \epsilon)] \quad (8.2)$$

$$E(\hat{\beta}) = E[(X^TX)^{-1}X^TX\ \beta] + E[(X^TX)^{-1}X^T\ \epsilon)] \quad (8.3)$$

Considering that the expected value of the sum equals the sum of the expected values, we can separate Equation (8.2) into two, resulting in Equation (8.3). It is known that E(e) = 0 and $X^TX)^{-1}X^TX$ equals the identity matrix, I, Therefore, our result will be:

$$E(\hat{\beta}) = E[I\ \beta] + 0 \quad (8.4)$$
$$E(\hat{\beta}) = \beta \quad (8.5)$$

Therefore, we can see that the expected value of our estimators is, as matter of fact, unbiased.

3.5.2 Covariance (cov)

This second property deals with the covariance. When working with regression models, we will be working with a matrix that derives from the multiplication of two matrices we have already seen in Equation 3. It is $(X^TX)^{-1}$. When doing this operation, we will notice that the result will be a symmetric matrix, which has the a_{ij} elements equal to the a_{ji} ones. More than that, if we multiply it by σ^2, there will be a new matrix in which its diagonal elements will tell us the variances of $\hat{\beta}\,0, \hat{\beta}\,1, \hat{\beta}\,2, \ldots, \hat{\beta}\,k$, and the off-diagonal ones will store the covariances. Based on the calculations shown in [1], the same thing was done using Python on [2].

3.6 Coefficient of Determination (R^2)

A parameter that we use when we want to know whether our model is good or not is called the Coefficient of Determination. This is a parameter that varies between 0 and 1. According to [1], the R^2 is often referred to as the amount of variability in the data explained or account by the regression model. To calculate it, we will be using two metric variables, one of which we have already seen when calculating σ^2, the so-called SSE in Equation (6), and a new one called SST (Sum of Squares Total). Now, when calculating SST, we will need the average of the values in vector Y and the vector itself. The formula will be the square sum of their difference, from the first value to the Nth one.

$$SST = \sum_{I=1}^{N}(Y_i - \bar{Y})^2 \quad (9)$$

Having calculated SST, we can now find R^2, which can be calculated by the subtraction of 1 and the ratio of SSE and SST. Mathematically, that will be:

$$R^2 = 1 - \frac{SSE}{SST} (10)$$

The entire calculation was done in [2]. After done, it was found out that R^2 = 0.994, which shows that our model has 99.4% variability in the data.

3.7 Predictions

With the points and the residuals, we can make predictions. According to [5], when one uses MLR for prediction, one is using a sample to create a regression equation that would optimally predict a particular phenomenon within a particular population. This leads to the conclusion that they help us know the future results of our problem. Table 3 shows five. As well as the residuals, the rest of the predictions can be seen on [2].

Tab.3 – Predictions

Predictions
129.30039315
50.78734327
114.21871551
104.7938394
142.08117015

All the results above help us to have a basic understanding of what the results will be. For example, if we consider x1 = 14.5, x2 = 220 and x3=5, we will have:

$$\hat{y} = 47.17 - 9.74 * 14.5 + 0.43 * 220 + 18.24 * 5$$
$$\hat{y} = 91.42398577$$

4. Conclusion

In this paper, we discussed about the use of Linear Algebra in Multiple Linear Regression. Linear Algebra is a field of mathematics used in many areas and Regression models is one of them. Following a complete review of the theory and using a set of data to illustrate our problem, we could see that the calculations become much easier when making use of matrix operations, such as transposition and inversion, instead of calculating all the statistical parameters that are needed to make the model. With the use of Python, an excellent tool for Linear Algebra problems and especially Linear Regression ones as well, the entire process becomes much faster for, as seen in the code referenced, the programming language already has many of the methods built in.

5. Acknowledgement

I would like to express my deepest gratitude to Professor Hugo de Souza Oliveira, Ph.D., who reviewed this work and gave me useful feedback.

6. References

[1] Montgomery D., Runger G. *Applied Statistics and Probability for Engineers*. John Wiley and Sons, Incorporated, Hoboken; 2013; 920 p.

[2] Martins T. Application of Linear Algebra in Multiple Linear Regression. Available from: Application of Linear Algebra in Multiple Linear Regression.ipynb - Colab (google.com)

[3] Woolridge J. *Introductory to Econometrics: A Modern Approach*. Cengage Learning, Boston;

2020; 816 p.

[4] Kroese P., Botev Z., Taimre T., Vaisman R. *Data Science and Machine Learning: Mathematical and Statistical Methods*. CRC Press, Boca Raton; 2019; 532 p.

[5] Osborne, J. Prediction in Multiple Linear Regression. *Practical Assessment, Research and Evaluation*. 2000; 7(2).