Simple Linear Regression from Scratch using Numpy

It is a very long article containing all the basics for Simple Linear Regression — From Theory to Practical. You can feel free to jump to any of the sections you want to read about. Given below are the sections:

1. Introduction
2. Dissecting Equation of Straight Line
3. Method of Least Squares
4. Model Metrics
5. Practical Coding using Numpy and Python Classes

Explore and Execute the code in Google Colab

Introduction

Let's take an example of Cricket to understand the concept of Linear Regression. But before jumping on it let us understand different scenarios.

We all know that pitch plays a major role in the game. If we play on a Hard Pitch, then we get extra bounce. Green pitch provides swing and skids. While red soil pitch or pitch after few days of rain provides extra spin. So can we say **that the pitch is related to how a person bats or how a bowler bowls**?

Pitches in Cricket

Next question is if it is related then what is the strength of relationship? Is **batting or bowling very highly related to the pitch or is there a normal relationship**? Can we say it will be very difficult for a batsman to play on a green pitch or it's just a random guess?

We know that different types of pitches are present in a single country itself. Can we say that **all the pitches present in different states will affect the batting or bowling? Or only some of the pitches**?

Next question that may come is can we quantify the relationship? This means **can we tell how much a specific type of pitch will affect the performance of a batsman or a bowler**? If pitch changes from green to hard, how much batting or bowling improves or deteriorates?

Now, if we know about all the above questions, can we predict the future? **Can we tell that based on a pitch how many runs a batsman will score or how many wickets a bowler will take? How accurately we can do this prediction**?

To understand and get answers for all the above questions there are various ways (algorithms). But, in this section we will try to understand the very first one, i.e. **Linear Regression.** Linear Regression tells that the answer to all the above questions is Linear. Now what does this statement means?

If the relationship between the pitch and the performance can have an approximate straight line relationship, then we can say that Linear Regression is the best tool to answer the above questions. To understand what I mean by this straight line relationship, let us first establish few assumptions:

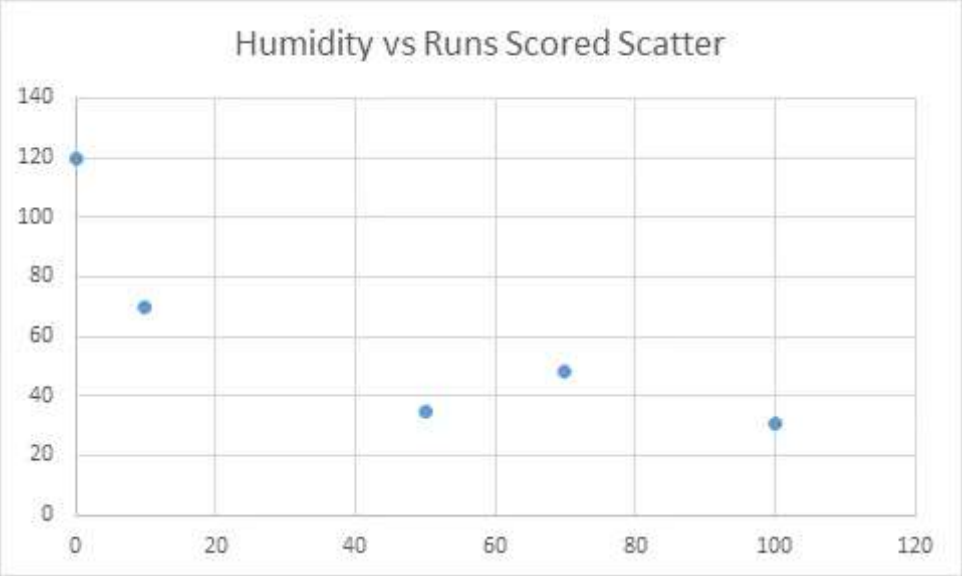
Assumptions

1. **0% water (humidity)** in pitch means **Hard Track**

2. **100% humidity** in pitch means **Spin Track**

3. **50% humidity** in pitch means **Green Track**

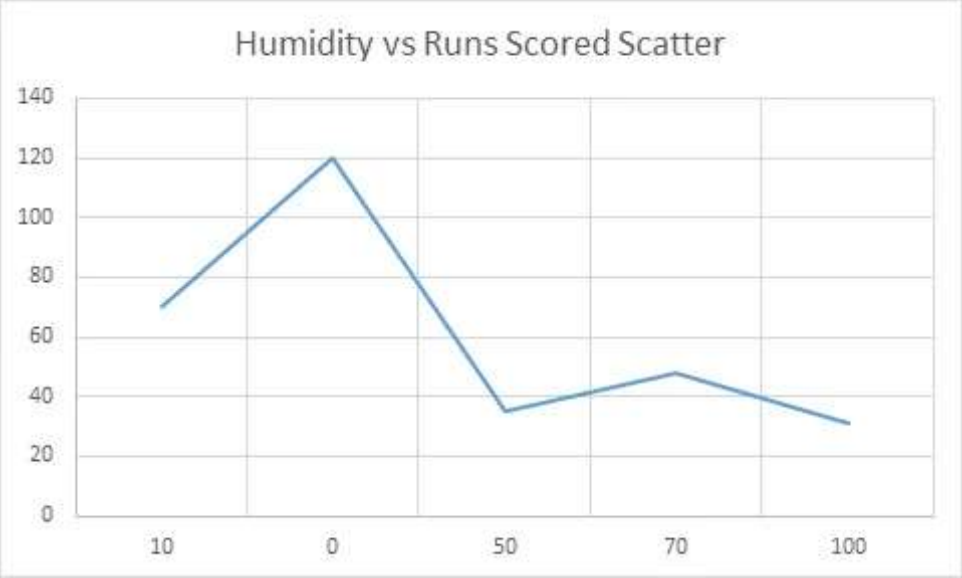
With these assumptions, if I know what is the percentage of humidity in the track, can I predict how many runs a batsman will score? Or how many wickets a bowler will take? Using a straight line graph. Let's understand this better with below graphs:



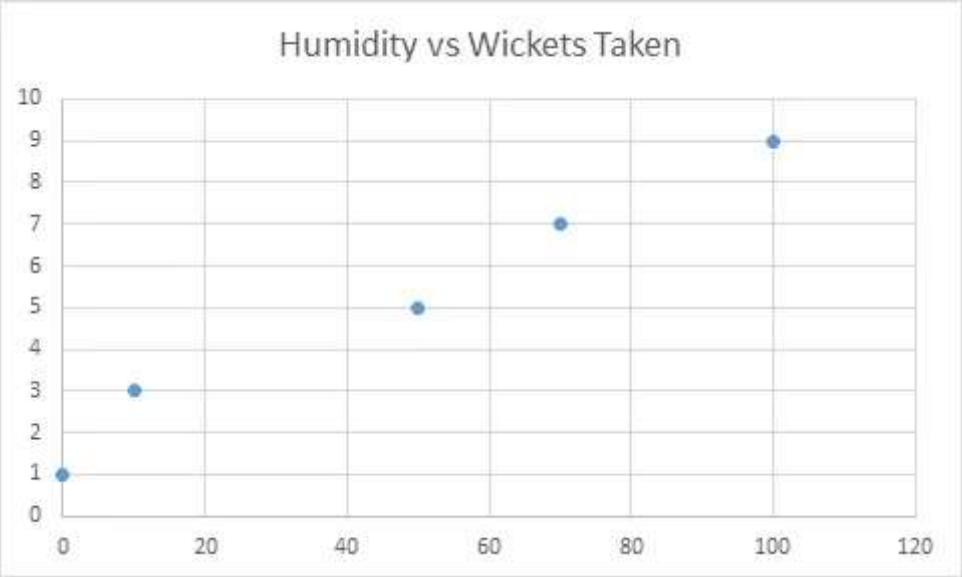
The above graph that we see tell us how much runs on an average a person has scored with varying values of humidity in the pitch. We can see that when humidity is around 0%, average runs scored is around 120. But if humidity increases to 100% the average runs come down to 31. Given below is the detailed table.

Humidity	Runs
10	70
0	120
50	35
70	48
100	31

But, is the above values following straight line or approximate straight line relationship? Please look at the below line chart,



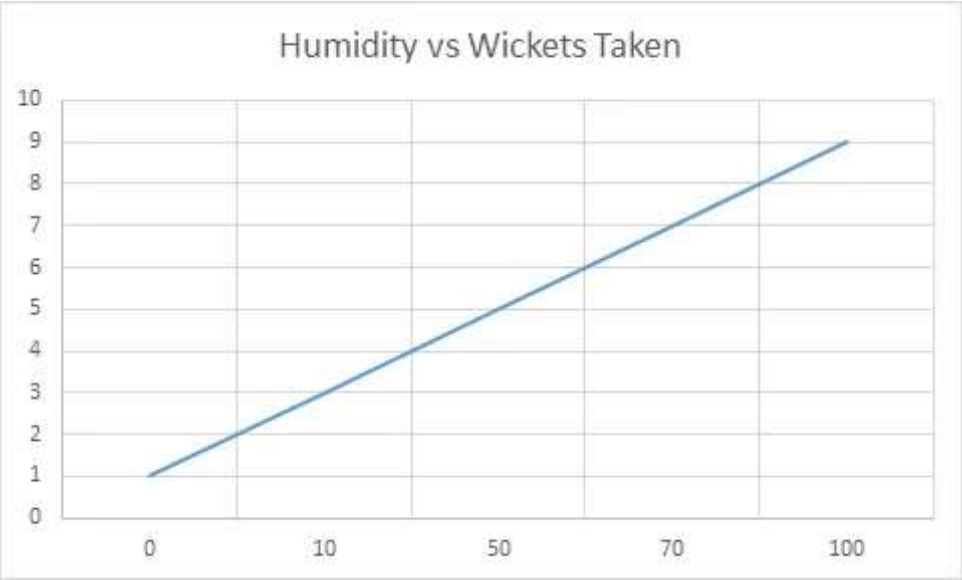
Above graph shows that it's not at all linear (Straight Line). Let us look at another example.



Above graph shows if the humidity is increasing how many wickets a bowler is taking. Given below are the detailed values.

Humidity	Wickets
0	1
10	3
50	5
70	7
100	9

By looking at the graph we can say that the relationship is approximately linear. Let us confirm with the help of line chart.



You can see that the line chart looks like a straight line. Hence, we can say that the relationship is linear and we can proceed with using Linear Regression as a Machine Learning tool.

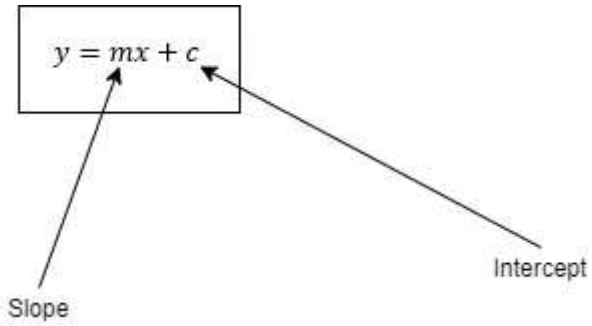
Now that we have understood what we mean by a Linear Relationship, let's dive deeper into Linear Regression Algorithm.

Let's go few years back, into our school days. If you remember there was something called as **Equation of a Straight Line**. Ring a bell? Now worries, I will help you out. If I know the equation of any straight line, I can find what values will be lying on that particular line. In our example, if we are able to find the equation of the straight line that we got for humidity vs wickets taken, **we will be able to find out how many wickets will be taken if I know the humidity**. Given below is the equation of straight line that we studied in our school days.

$$y = mx + c$$

Let us understand this equation in a better way.

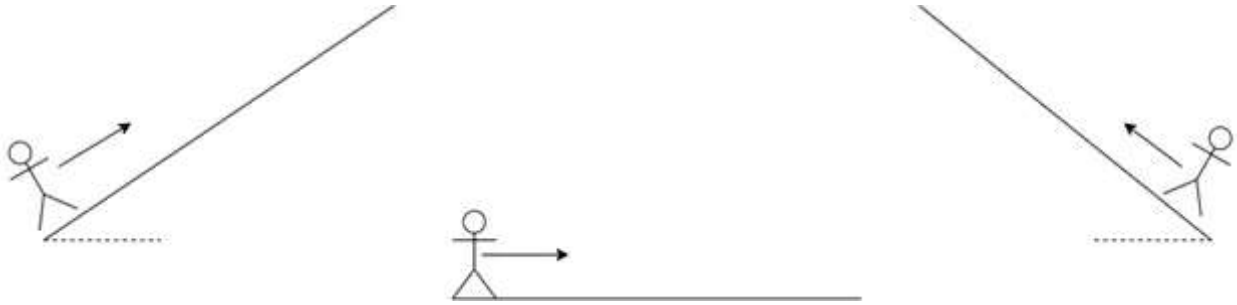
Dissecting Equation of a Straight Line



Mathematically speaking, slope tells us about how much change small change in x will bring on y. Intercept tells about at what point the straight line cuts the y-axis. Obviously, these definitions can be found on any text book or googling it. Let's understand in the layman terms what they mean.

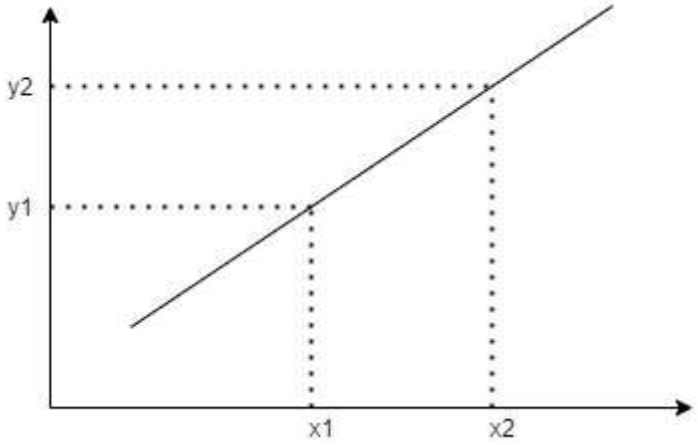
Slope

Tell me, is there any difference when you walk on a straight road, uphill or downhill? Yes, right? When you're going up the hill, energy required from you is more. If you're going down the hill, then energy required is less. While straight road means normal energy. So why three different energies? That's because of the slope. When going up or down, there is a slope which either hinders your movement (uphill) or helps you in the movement (downhill). In a straight road there is no slope, so no hindrance or help.



Slope can be found out by using following formulas:

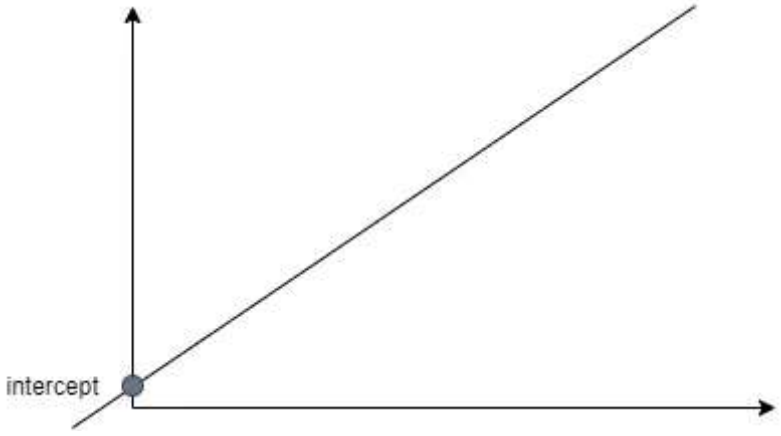
$$slope\ (m) = tan\theta = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$



Now, let's understand the concept of intercept.

Intercept

Intercept is also called as y-intercept or just a constant. It is a fixed value that we get when all the values of x (i.e. x1, x2, x3... etc.) are zero. In itself it is not meaningful, but it helps in making your predictions unbiased. We will look at that in subsequent sections. As of now let us visualize the intercept concept.



Coming back to Cricket Example

Now, how can we get the equation of straight line for our cricket example? We can use the following equation to get the predicted wickets value,

$$\overline{wickets} = c + m * humidity$$

Using the above equation, we can find the prediction of wickets based on humidity. But, finding the values of c and m is a little complex part in Linear Regression. This is because we need to find the best straight line that leads us to most accurate predictions. Since, at the start of the algorithm we don't have a straight line, or by end of the algorithm we will get multiple straight lines (We will understand this

later), hence getting immediate value of c and m will not be possible. Therefore instead of representing them with c and m, we represent it with B0 and B1. Therefore, the same equation becomes,

$$wickets = \beta_o + \beta_1 * humidity$$

If we generalize the above equation, equation of Linear Regression becomes,

$$y = \beta_o + \beta_1 * x$$

Finding the Coefficients

We have seen in the example discussed in the previous section that if I know the values of B0 and B1 then I can make the predictions for wickets. These values are called as **parameters** or **coefficients**. For a normal straight lines these values are equal to the values of intercept and slope. But, when we talk about Linear Regression, we need to find the best values of these parameters. For that we have a specific approach called as the **Method of Least Squares**. Let us discuss this method.

Method of Least Squares

Before we start looking at this method, we need to understand errors. Errors, also called as Residuals, are the difference between the actual value and the predicted value. So , in our example of predicting wickets, suppose we randomly give some value to B0 and B1.

$$\beta_o = 0.5$$
$$\beta_1 = 1$$

Now, if we say that the pitch has a humidity of 50%. Then, by putting the values in the formula we get:

$$y = 0.5 + 1 * 0.5$$

This gives us a prediction of 1. This means that if the humidity is 50% then the number of wickets taken by the bowler will be 1. But we have seen the actual table and it says that if the humidity is 50% then the number of wickets taken by a bowler would be 5. The value 5 here is called as the **actual value** while the number 1 is the

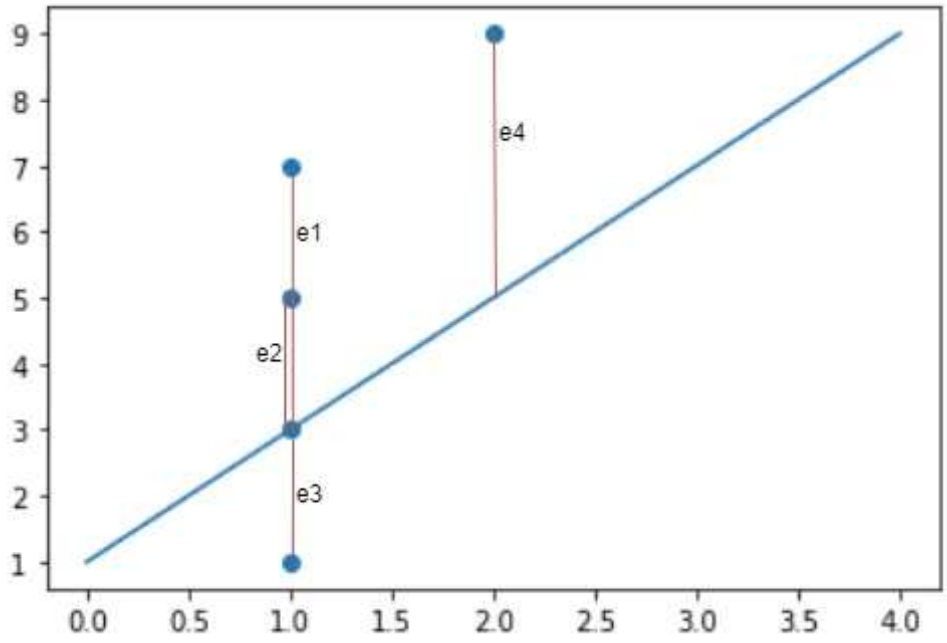
predicted value. We can see that our prediction is not correct. There is an error in the prediction. This error is equal to:

$$error\ (e) = actual - predicted = 5 - 1 = 4$$

Similarly, for all the values of humidity present in the table, we can find out the predicted values, and hence the error. The table below shows the actual and predicted values and their respective errors.

Humidity	Wickets	Prediction (Rounded Off)	Error
0	1	0.5 = 1	0
10	3	0.6 = 1	2
50	5	1	4
70	7	1.2 = 1	6
100	9	1.5 = 2	7

In the table above we can see that there are a lot of errors/residuals. If we plot these values of errors on the line that we have got, the graph will show the predicted points and the errors, as given below,



Since we have understood the concept of errors/residuals, now let's move on to finding out the values of coefficients. The main aim of this algorithm is to find the best values for the coefficients that gives us the minimum error.

Let's square all the errors and add them,

$$e_1^2 + e_2^2 + e_3^2 + e_4^2 = 0^2 + 2^2 + 4^2 + 6^2 + 7^2 = 0 + 4 + 16 + 36 + 49 = 105$$

The above value of 56 is called as the **Squared Error.** Now let's find the mean of above value,

$$\frac{105}{5} = 21$$

The value of 14 is called as the **Mean Squared Error.** Now let's find the mean of above value,

$$\sqrt{Mean\ Squared\ Error} = \sqrt{14} = 3.74$$

This value of 3.74 is called as the **Root Mean Squared Error.** We have to minimize this error to get the best value of the coefficients.

Minimization of Error

The above mathematical calculations that we have seen, we can depict it with the help of a standardized equation,

$$\sum_{i=1}^{5} e_i^2$$

Above equation is the standardized version of Squared Error. Let's find mean squared error now,

$$\frac{1}{5}\sum_{i=1}^{4} e_i^2$$

This is the standardized equation of Mean Squared Error. As of now in our example the number of data points are 4. But, in real life scenario there can be a lot of data points. Let's represent it with the letter N. Then the formula becomes,

$$\frac{1}{N}\sum_{i=1}^{N} e_i^2$$

Let's go back and revisit error. Error was defined as difference between Actual and Predicted value. If we represent actual value with y and predicted value with y_cap. Then error can be represented as,

$$prediction = \hat{y} = \beta_0 + \beta_1 X$$

Where in our example y was Wickets and x was Humidity. As we have 4 values, the predicted value will be,

$$\hat{y_1} = \beta_0 + \beta_1 X_1$$
$$\hat{y_2} = \beta_0 + \beta_1 X_2$$
$$\hat{y_3} = \beta_0 + \beta_1 X_3$$
$$\hat{y_4} = \beta_0 + \beta_1 X_4$$

So, the error formula can be re-represented as,

$$\sum_{i=1}^{N} (y_i - \hat{y_i})^2$$

And this can again be represented as,

$$E = \frac{1}{N}\sum_{i=1}^{N} (y - \beta_0 - \beta_1 X_i)^2$$

This is the generalized equation of **Mean Squared Error.** This is the final equation that needs to be minimized to get the best values of the coefficients.

Now that the above concepts are understood, let's revisit our school mathematics. We studied calculus in detail, when we were in 11th Standard. There was a basics concept that if we want to minimize any equation then we differentiate it once and then make it equal to 0. Same thing we will do for the generalized mean squared

error equation. We will differentiate it once and put it equal to 0. This will give us the value of B0 and B1.

But, in out equation we have two parameters. So we will do partial differentiation, one with respect to B0 and other with respect to B1. These differentiations can be represented by,

$$\frac{dE}{d\beta_0} = 0$$

And

$$\frac{dE}{d\beta_1} = 0$$

Now, to solve the equation in detail requires detailed calculus, matrix and determinants. So we will not jump inside the mathematics. If you want to understand in detail, go to the appendix-1. Let us see what is the final output that we get once we solve the above calculus equations. The final formulas of B0 and B1 are given below.

$$\beta_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

We have introduced two new terms here — x_bar and y_bar. These are nothing but mean of all the x values and all the y values. In our cricket example,

$$\bar{x} = mean(all\ the\ humidity\ values\ in\ data)$$
$$\bar{y} = mean(all\ the\ wickets\ values\ in\ data)$$

Once we know the B1 value, we can proceed and get the B0 value using the formula below,

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

And hence we finish the concept of finding the coefficient using Least Squares Method. Let's check it out and try this in our example of Cricket.

Lease Squares Application on Cricket Example

Let's start with finding the

value. First step is finding the mean of humidity and wickets.

$$mean(humidity) = \bar{y} = \frac{1}{5}(0 + 10 + 50 + 70 + 100) = 56$$

$$mean(wickets) = \bar{x} = \frac{1}{5}(1 + 3 + 5 + 7 + 9) = 5$$

Now, we have to subtract these values from each values of x and y, and then multiply and add them. Let's start step by step.

Step 1: Finding difference of means for x and y

In the first step we will focus on the numerator part of the equation. The table given below shows the calculation of difference of means of **x (Humidity)** and **y (Wickets).**

Humidity(x)	Wickets(y)	Mean(x)	Mean(y)	Mean Difference of X Humidity (x) - Mean(x)	Mean Difference of Y Wickets(y) - Mean(y)
0	1	46	5	-46	-4
10	3	46	5	-36	-2
50	5	46	5	4	0
70	7	46	5	24	2
100	9	46	5	54	4

In the next step we have to find the multiplication of these mean differences and add them all.

Step 2: Summation of Product of Mean Differences

The table below shows and extra column which we have got by multiplying the Mean Difference column of X and Y. Next we need to add all the values of the last column.

Humidity(x)	Wickets(y)	Mean(x)	Mean(y)	Mean Difference of X Humidity (x) - Mean(x)	Mean Difference of Y Wickets(y) - Mean(y)	Product of Mean Difference
0	1	46	5	-46	-4	184
10	3	46	5	-36	-2	72
50	5	46	5	4	0	0
70	7	46	5	24	2	48
100	9	46	5	54	4	216

Therefore,

$$\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) = 184 + 72 + 0 + 48 + 216 = 520$$

Now that we have got the Numerator part of the equation, let us focus on the denominator part.

Step 3: Finding the Summation of Squared Difference of Means of X

In the table below, the fourth column shows the Mean Difference of X while the fifth column finds the square of mean difference.

Humidity(x)	Wickets(y)	Mean(x)	Mean Difference of X Humidity (x) - Mean(x)	Mean Difference Squared
0	1	46	-46	2116
10	3	46	-36	1296
50	5	46	4	16
70	7	46	24	576
100	9	46	54	2916

Therefore,

$$\sum_{i=1}^{N}(x_i - \bar{x})^2 = 2116 + 1296 + 16 + 576 + 2916 = 6920$$

Step 4: Finding the value of B1

As we remember the formula,

$$\beta_1 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Therefore,

$$\beta_1 = \frac{520}{6920} = 0.08$$

Now that we have got the value of β_1, let's find the value for β_0

Step 5: Finding the value of B0

If we try to recollect the formula,

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

We already know the value of $\bar{x}$ and $\bar{y}$ which is 46 and 5.

Therefore,

$$\beta_0 = 5 - 0.08 * 46 = 1.32$$

Step 6: The Final Equation

As we know the final value of β_1 and β_0, which is 0.08 and 1.32, the final equation becomes,

$$\boldsymbol{y = 1.32 + 0.08x}$$

Testing the New Error

Based on our new equation, we can get predicted value and that can be compared with the actual value. This will give us the error. Then we can use the generalized equation of Root Mean Squared Error to find the final error. The table given below shows the errors for each prediction,

Humidity	Wickets	Prediction	Error	Squared Error
0	1	1.32	-0.3	0.1024
10	3	2.12	0.88	0.7744
50	5	5.32	-0.3	0.1024
70	7	6.92	0.08	0.0064
100	9	9.32	-0.3	0.1024

If we find the sum of squared of all the errors, we get

$$SSE\ (Sum\ of\ Squared\ Errors) = 0.1 + 0.8 + 0.1 + 0.01 + 0.1 = 1.09$$

Next, let's find the mean of sum of squared errors,

$$MSE\ (Mean\ Squared\ Errors) = \frac{1.09}{5} = 0.22$$

Finally, let's find the root mean squared error,

$$RMSE\ (Root\ Mean\ Squared\ Error) = \sqrt{0.22} = 0.47$$

Final Analysis

When we randomly assigned values for B0 and B1 we saw that the RMSE value was above 3. But, when we used the method of least squares to find the best value of the coefficients, RMSE value got reduced to 0.47. So, definitely the method of least squares do help in minimizing the error and getting the best coefficients. You can think that is it the only method for finding the best values? No, there are many other methods that we will look at once we proceed with our discussions on different algorithms.

How to know that our model is the best model

In the above sections we saw how we can use Linear Regression to make predictions. In our case we predicted number of wickets taken with the help of humidity of the pitch. This scenario where we only use one variable (humidity) for prediction is called as **Simple Linear Regression.**

We have seen that if the value of RMSE is low for a Linear Regression model then it is considered as good. But how much low? This is something which is difficult to quantify. A low value for one problem statement can be very high value for other. Suppose we are predicting how much percentage of cancer has been spread inside a patient's body? Suppose, using the method of least squares we got the value 2.11. So, 2.11 is the lowest value for Cancer problem statement but the same value is considered very high when we talk of cricket problem statement (which has an RMSE of 0.47).

So, are there any ways with which we can decide about the model's performance? And which can be generalized across all the problem statements? Yes, there are some of the approaches that can be used for this analysis. They are listed below:

1. Residual Standard Error

2. R2 Statistic

3. Adjusted R2 Statistic

Adjusted R2 Statistic is used in Multiple Linear Regression and we will discuss about it in the next section. Let's look at the first two statistics.

Residual Standard Error

In the problem statement of cricket that we solved, we got the RMSE as 0.47. So this means that even though our predictions are very close to the actual values, still there is some error present. If the RMSE value is coming as 0, then our predictions are exactly equal to the actual values. But, in real life scenario it's not possible. Therefore, we can re-write the equation of Linear Regression as,

$$y = \beta_0 + \beta_1 X + \epsilon$$

Where e represents error which is always present in our model. If are able to find the standard deviation of this error term, then we will be able to know that approximately by how much value our predictions will differ from that actual values. This value of standard deviation is called as **Residual Standard Error**.

The formula for calculating RSE is given by,

$$RSE = \sqrt{\frac{1}{N-2} * SSE}$$

We have already seen the formula of SS0045 and we know that its value was 21. So the RSE for Cricket example will be,

$$RSE = \sqrt{\frac{1}{5-2} * 1.09} = \sqrt{0.36} = 0.60$$

So, the above value can be inferred as that the model will deviate from the actual values by approximately 0.06 (approximately 1 we can say) wickets. If we speak percentage-wise, then we know that the mean of wickets is 5. So, percentage RSE will be,

$$RSE_{\%} = \frac{0.60}{5} * 100 = 12\%$$

This deviation is less. So, using the concept of RSE we can say that our model's predictions were good and the model is also very good. RSE is also used for understanding **Overfitting** and **Underfitting** concepts, which we will discuss later.

Let's Start the Fun. Let's Code

There are a lot of packages present that can be used to make a Linear Regression model, the most famous one being Scikit-Learn. But, we will build the code from scratch first to understand all the formulas that we saw, using Numpy. Then we will explore Scikit-Learn to do the same.

Let's go step by step now.

Step 1: Store the Data in a Variable

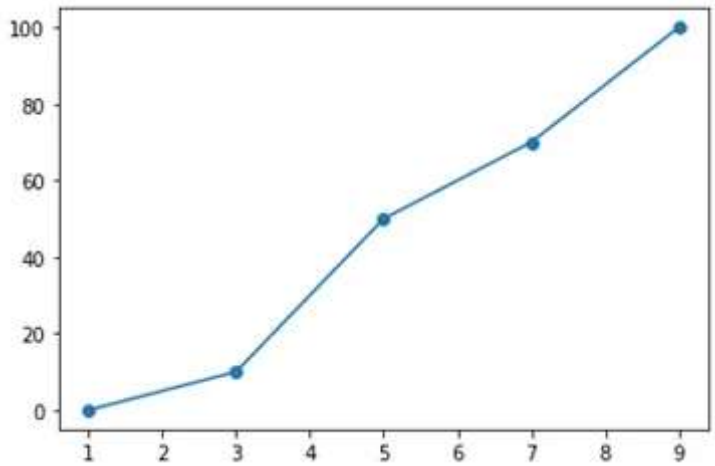
We have our Humidity and Wickets data. Let's store it two different variables.

```
humidityData = np.array([0,10,50,70,100])
wicketsData = np.array([1,3,5,7,9])
```

Step 2: Visualizing the Scatter Plot for the dataset

```
import matplotlib.pyplot as plt
plt.scatter(wicketsData, humidityData)
plt.plot(wicketsData, humidityData)
plt.show()
```

The above code snippet created scatter plot for the data and then draws a line plot connecting the lines. Given below is the visualization that we see,



Step 3: Calculating the Slope

```
import numpy as np
meanDifferenceX = X - np.mean(humidityData)
meanDifferenceY = Y - np.mean(wicketsData)
```

Numpy is the mathematical package used for most of the mathematical operations performed on arrays. Using the mean function of Numpy, we can find the mean of X and Y. Then we can subtract the mean from the original data. This leads to a matrix subtraction, where from all the elements of X and Y the mean is subtracted.

This is one of the benefits of using Numpy. If we use normal python code, then we'll have to write multiple for loops to do the same. But, Numpy does everything using Matrix multiplications and hence the process becomes much faster.

Now that we have the mean difference of X and Y we can find the numerator by doing matrix multiplication of the differences and then adding them all. Code snippet given below helps to accomplish this.

```
meanDifferenceMultiplication = np.multiply(meanDifferenceX,
meanDifferenceY)
numerator = np.sum(meanDifferenceMultiplication)
```

np.multiply helps in doing the basic multiplication of two matrices. Then np.sum adds all the elements present in the matrix we got through multiplication. This gives us the numerator part.

Similarly, for getting the denominator, we will find the square of the mean difference of X and then add all the elements. Code snippet below shows the same,

```
denominator = np.sum(np.square(meanDifferenceX))
```

np.square can be used to find the square of the elements of a matrix. Now that we have the numerator and the denominator, we can divide them and find the value of

```
b_1 = np.divide(numerator,denominator)
```

np.divide can be used for normal division or matrix division. As both numerator and denominator contains a scalar value, it will give us one single value.

Step 4: Calculating the Intercept

```
b_0 = np.mean(Y) - np.multiply(b_1,np.mean(X))
```

Now all that is left is doing the predictions. First, let us look at the complete structured code of the discussion that we have done till now.

Complete Structured Code

I have moved the complete code inside a class called Linear Regression. The code snippet is given below,

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
humidityData = np.array([0,10,50,70,100])
wicketsData = np.array([1,3,5,7,9])
class LinearRegression:
def __init__(self, humidityData, wicketsData):
self.X = humidityData
self.Y = wicketsData
def slope(self):
meanDifferenceX = self.X - np.mean(self.X)
meanDifferenceY = self.Y - np.mean(self.Y)
numerator = np.sum(np.multiply(meanDifferenceX, meanDifferenceY))
denominator = np.sum(np.square(meanDifferenceX))
b_1 = np.divide(numerator,denominator)
```

```
    return b_1

    def train(self):

    self.b_1 = round(slope(self.X, self.Y),2)

    self.b_0 = round(np.mean(self.Y) - round(self.b_1,2) *
    np.mean(self.X),2)

    print("**********************************************************
    ********")

    print("B_1 Value is: ", self.b_1, "\n")

    print("**********************************************************
    ********")

    print("B_0 Value is: ", self.b_0, "\n")

    print("**********************************************************
    ********")

    def predict(self, X):

    output = np.add(self.b_0, np.multiply(self.b_1, np.array(X)))

    return output
```

First, let's create the object of the class.

```
lr = LinearRegression(humidityData, wicketsData)
```

Now, let's call the train() function,

```
lr.train()
```

This will give us the following output,

```
**************************************************************************
B_1 Value is:  0.08

**************************************************************************
B_0 Value is:  1.32

**************************************************************************
```

Finally, let's make our predictions,

```
lr.predict(humidityData)
```

Predictions are given below,

```
array([1.32, 2.12, 5.32, 6.92, 9.32])
```

If you match it with the table that we created in this chapter, it gives us the same result. Finally, let's move on to the metrics and check the model's performance.

Model Metrics

First let us create a function for calculating Residual Standard Error.

Residual Standard Error using Python

So, if we make this formula in Numpy, it will become,

```
def rse(predictions, Y):
error = Y - predictions
squaredError = np.square(error)
sse = np.sum(squaredError)
rse = np.sqrt(np.divide(sse,len(Y)-2))
return rse
```

First we find the error by subtracting predictions from actuals. Then we square these errors and add them to get our SSE. Finally, we divide the SSE with total elements minus two. We can get the total number of data points by finding length of Y. Later, whatever is the answer we find the square root of it. We can add this section of code as well in our base class. Update code is given below.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
humidityData = np.array([0,10,50,70,100])
wicketsData = np.array([1,3,5,7,9])
```

```
class LinearRegression:

def __init__(self, humidityData, wicketsData):

self.X = humidityData

self.Y = wicketsData

def slope(self):

meanDifferenceX = self.X - np.mean(self.X)

meanDifferenceY = self.Y - np.mean(self.Y)

numerator = np.sum(np.multiply(meanDifferenceX, meanDifferenceY))

denominator = np.sum(np.square(meanDifferenceX))

b_1 = np.divide(numerator,denominator)

return b_1

def train(self):

self.b_1 = round(self.slope(),2)

self.b_0 = round(np.mean(self.Y) - round(self.b_1,2) *
np.mean(self.X),2)

print("**********************************************************
********")

print("B_1 Value is: ", self.b_1, "\n")

print("**********************************************************
********")

print("B_0 Value is: ", self.b_0, "\n")

print("**********************************************************
********")

def predict(self, X):

self.predictions = np.add(self.b_0, np.multiply(self.b_1,
np.array(X)))

return self.predictions

def sumSquaredErrors(self):

error = self.Y - self.predictions

squaredError = np.square(error)

self.sse = np.sum(squaredError)

return self.sse

def rse(self):

sse = self.sumSquaredErrors()

rse = np.sqrt(np.divide(sse,len(self.Y)-2))
```

```
print(f"The model is deviating {round(rse/np.mean(self.Y),2)*100}%
from the mean")
```

Now, after we have trained the model and got the predictions, we can try to get the RSE. For this we will call the function, as given in the code snippet below,

```
lr.rse()
```

This will give me the following results,

```
The model is deviating 12.0% from the mean
```

Again, the value is same as the value that we got manually. Next, let us calculate R2 Statistic

R2 Statistic

RSE is a good statistic but it tells us only in terms of Y variable, which means in our example RSE only tells us about the model in terms of Wickets taken. But, when we talk about the variance present in our model, it fails to explain.

That's the reason why R2 Statistics value comes handy, as it tells about the proportion of variance explained by our model, that is present in our data. As we know from statistics that variance tells about the data deviation from the mean, in terms of the squared value. So, more variance explained by our model, better is the model.

Formula-wise, R2 can be defined as,

$$R^2\ Statistic = 1 - \frac{RSS}{TSS}$$

When we talk about RSS, it is same as SSE that we saw earlier, given by the formula,

$$RSS = SSE = \sum (y - \hat{y})^2$$

TSS is called as Total Sum of Squares, which is similar to the above formula, but instead of having predictions, we take the mean.

$$TSS = \sum (y - \bar{y})^2$$

So, the final formula becomes,

$$R^2 = 1 - \frac{\Sigma (y - \hat{y})^2}{\Sigma (y - \bar{y})^2}$$

So, in other terms we can say that R2 statistic tells about the proportion of variance present in Y (Wickets) that is explained by X (Humidity). The value of R2 lies between 0 to 1. Closer to 1 means more variance is explained, closer to 0 means less variance is explained.

Suppose, we get the R2 statistics value as 0.42, then we can say that 42% of the variance present in our Y variable is explained by the X variable using our model. Now let us see what is the R2 statistic value for our cricket example. Instead of using excel, let's use python. So first we will create the formula and then we will try to find the value of R2 statistic.

```
def r2Statistic(predictions, Y):
RSS = np.sum(np.square(Y - predictions))
TSS = np.sum(np.square(Y - np.mean(Y)))
r2 = 1 - (RSS/TSS)
return r2
```

When we run this code we get the R2 value of **97.28%.** This means that R2 Statistic also says that our model is pretty good. ***(Remember that our data is dummy data and not taken from actual real-life dataset. We will apply our code on an existing dataset later to see our model performance)***

Linear Regression | Machine Learning | Numpy | Python | Scratch

Follow

Written by Himanshu Singh

246 Followers

ML Consultant, Researcher, Founder, Author, Trainer, Speaker, Story-teller Connect with me on LinkedIn: https://www.linkedin.com/in/himanshu-singh-2264a350/

More from Himanshu Singh

 Himanshu Singh

Exploring Asynchronous Requests in Python with Flask and Gevent

In the world of web development, it's important to ensure that our applications are responsive and performant, especially when handling...

5 min read · Jul 4, 2023

 15  1 

Himanshu Singh

Day 4: Building Multi-do RAG and packaging using Streamlit

This is part of the series—10 days of Retrieval Augmented Generation

6 min read · Feb 4, 2024

43

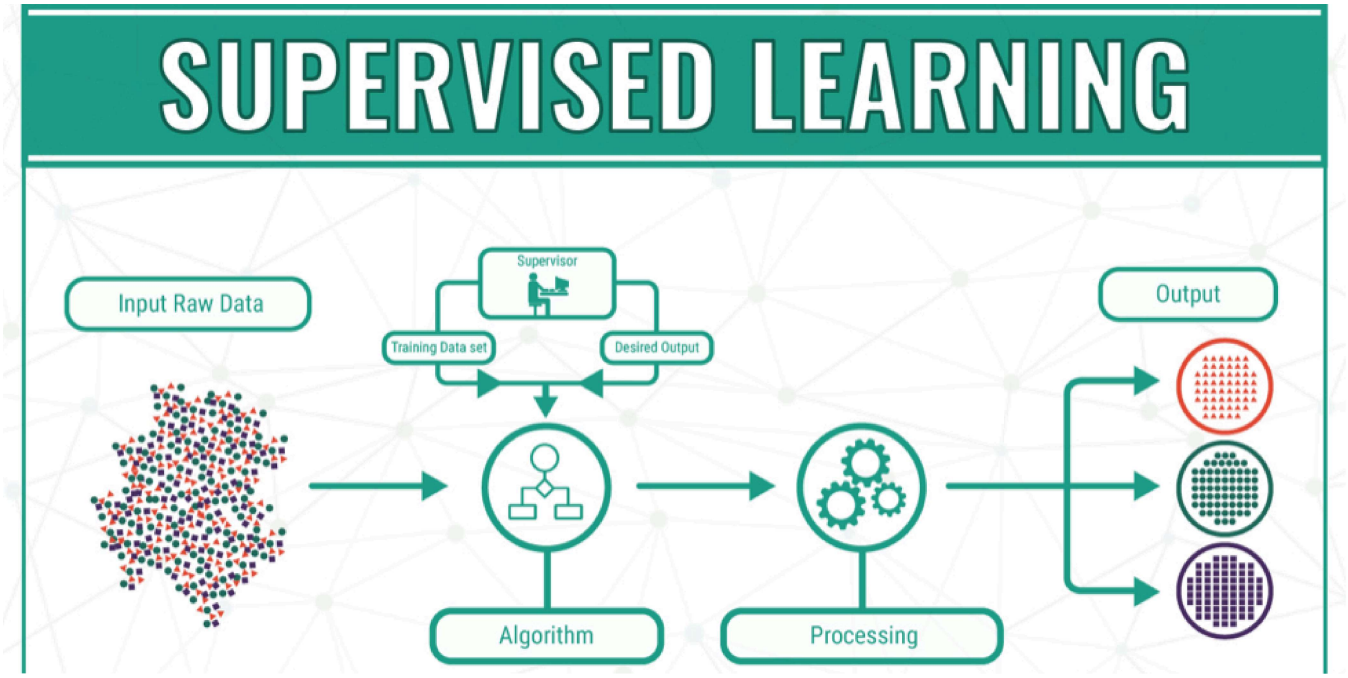
Himanshu Singh

Day 6: Building complete RAG pipeline in Azure

This is part of the series—10 days of Retrieval Augmented Generation

5 min read · Feb 9, 2024

20



Himanshu Singh

Supervised Learning Methods using Python

Machine Learning can be classified as of three types:-

8 min read · Jun 7, 2018

127 1

See all from Himanshu Singh

Recommended from Medium

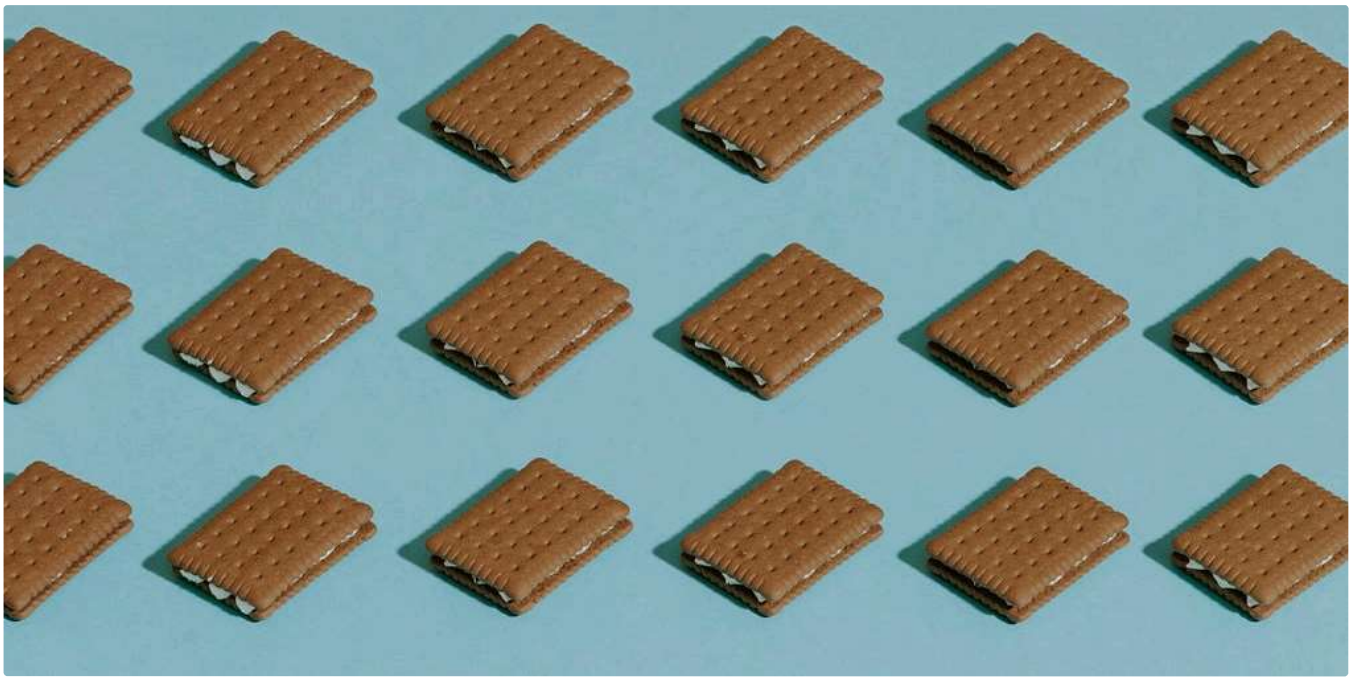
Mansi in Code Like A Girl

Coding Your First Neural Network FROM SCRATCH

A step by step guide to building your own Neural Network using NumPy.

9 min read · Oct 26, 2023

192 1



V VAV Data

Numpy Functions—3

discussion about numpy functions used in python.

· 2 min read · 4 days ago

1

Lists

Predictive Modeling w/ Python

20 stories · 1034 saves

Practical Guides to Machine Learning

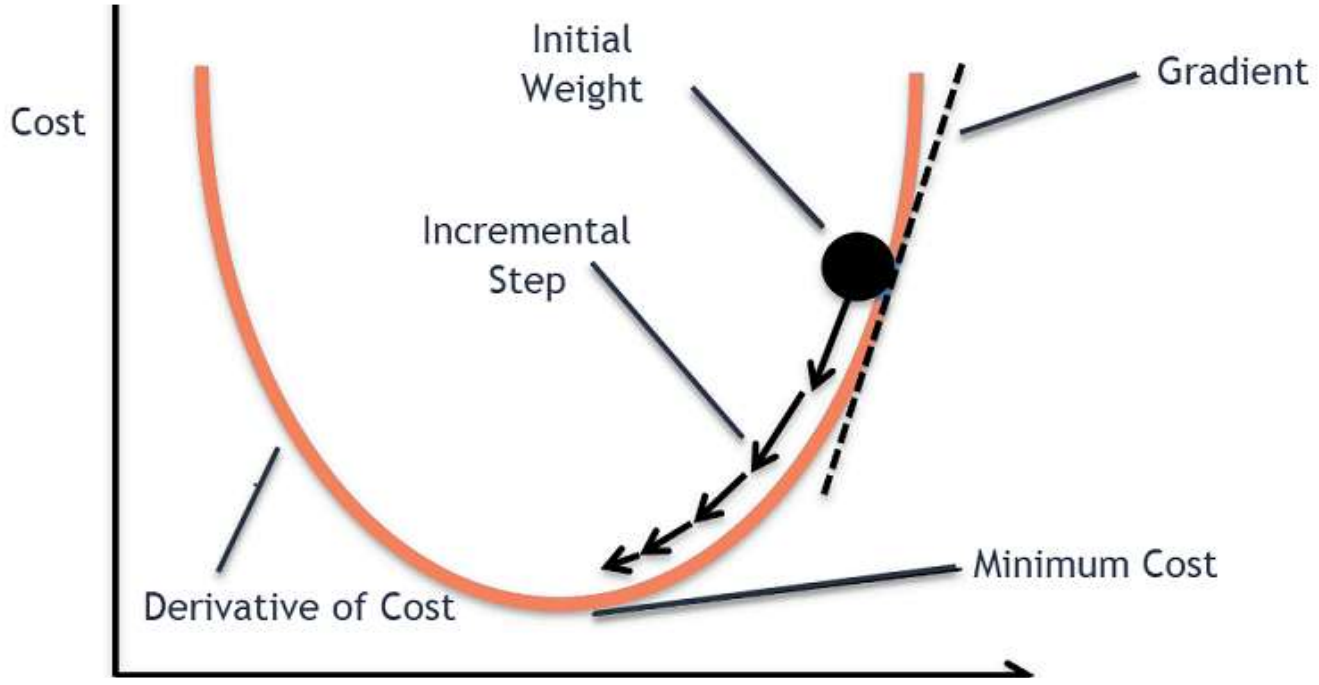
10 stories · 1236 saves

Coding & Development

11 stories · 522 saves

Natural Language Processing

1319 stories · 811 saves



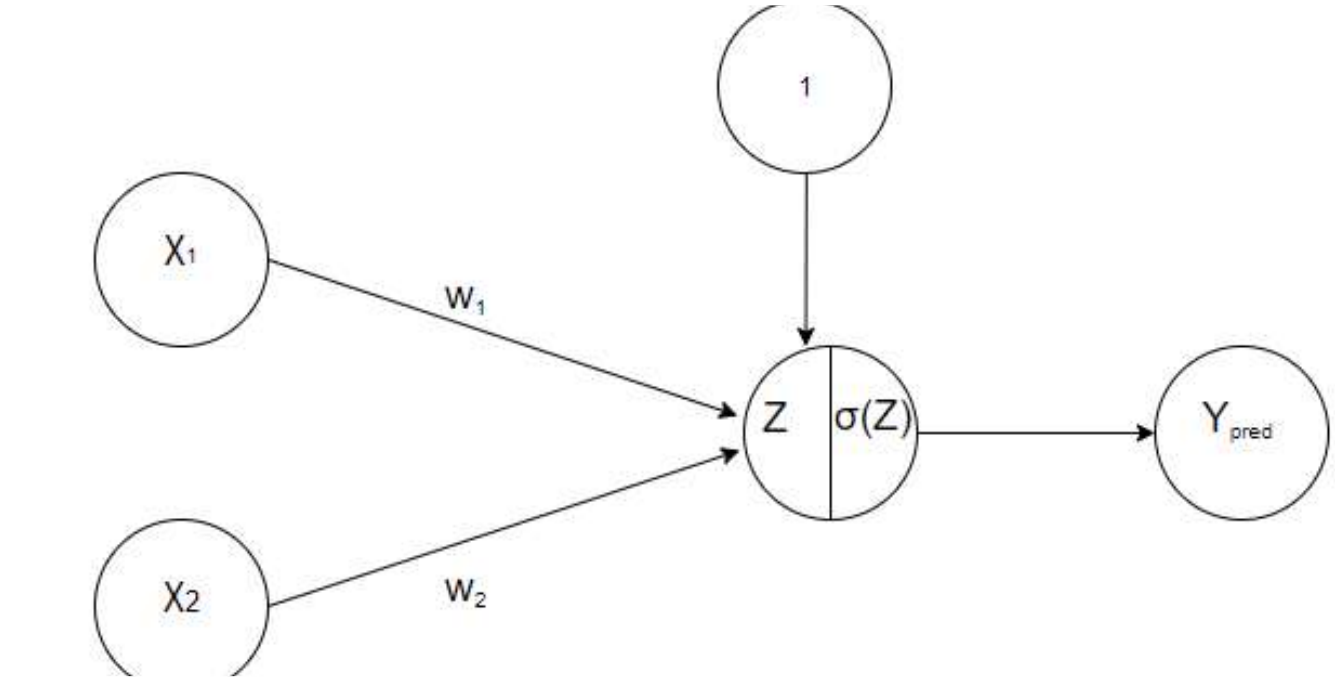
Prasad Meesala

Variants of Gradient Descent and Their Implementation in Python from Scratch

Hello everyone, Do you like math ? Whatever it may be, this article is for you. In this article, I'm going to give a brief overview of...

13 min read · Nov 22, 2023

75



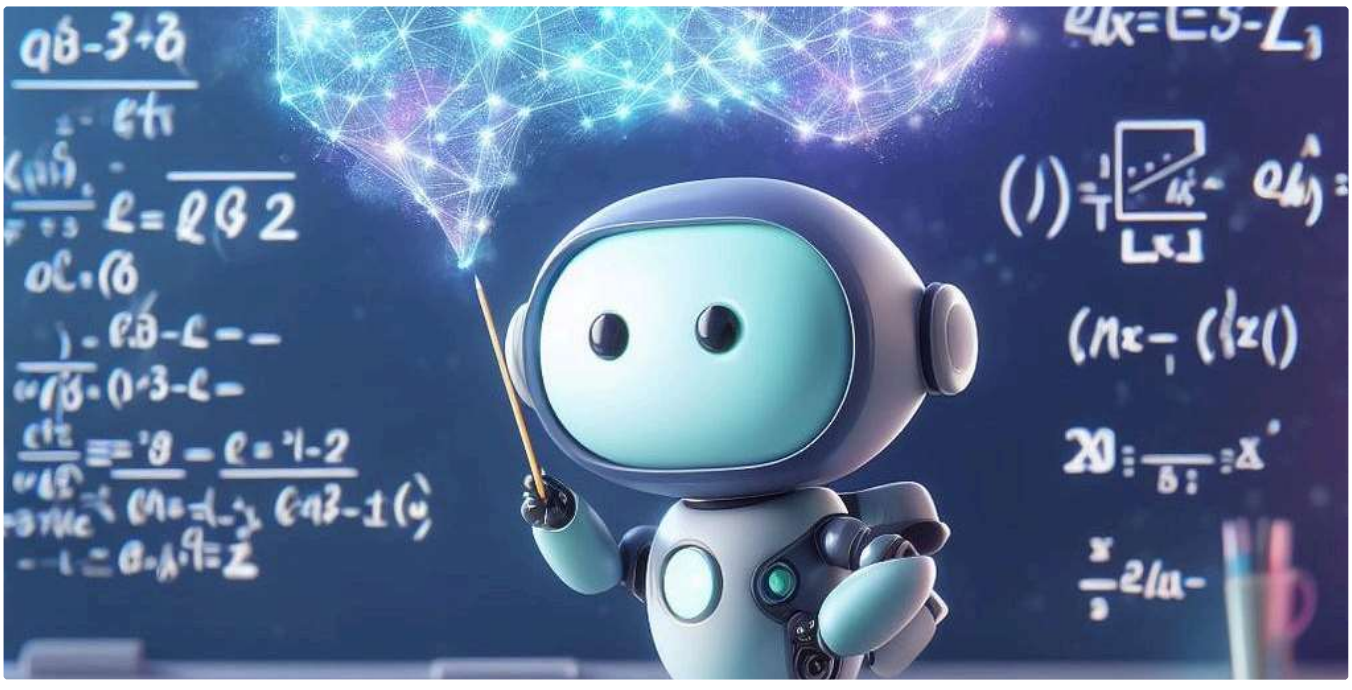
Kamel SOUDANI

Building an Artificial Neural Network Model by hand

ANN with one hidden layer and one formal neuron trained using backpropagation algorithm

8 min read · Jan 9, 2024

3



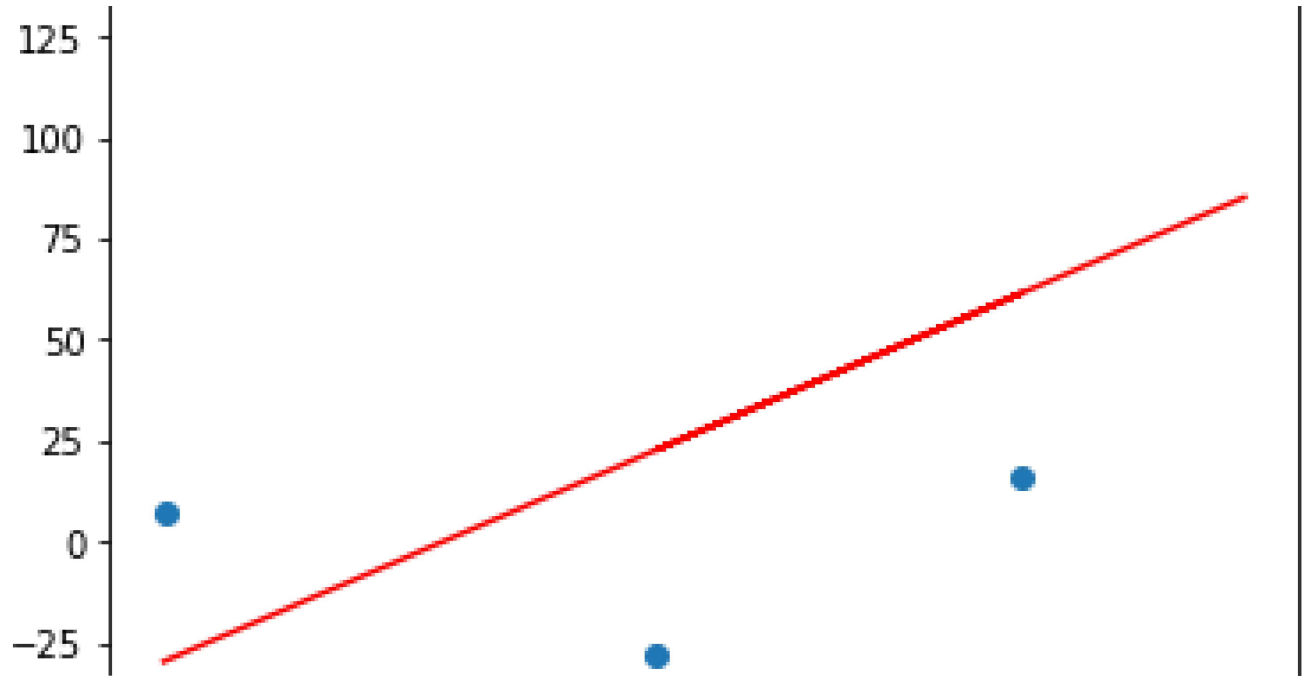
Senash Thenuja

Math Behind Backpropagation and Gradient Descent

Learn how Neural Networks are trained under the hood...

6 min read · Nov 12, 2023

164



HRJEET SINGH

Gradient descent

late-1800's—Neural Networks appear as an analogy to biological systems

3 min read · Feb 2, 2024

103

See more recommendations