**Logistic Regression Algorithm**

Logistic Regression is a “Supervised machine learning” algorithm that can be used to model the probability of a certain class or event. It is used when the data is linearly separable and the outcome is binary or dichotomous in nature.

**Types of Logistic Regression**

* **Simple Logistic Regression:** a single independent is used to predict the output
* **Multiple logistic regression:** multiple independent variables are used to predict the output

**Extensions of Logistic Regression**

Although it is said Logistic regression is used for Binary Classification, it can be extended to solve multiclass classification problems.

**Multinomial Logistic Regression:** The output variable is discrete in three or more classes with no natural ordering.

Food texture: Crunchy, Mushy, Crispy

Hair color: Blonde, Brown, Brunette, Red​

**Ordered Logistic Regression:** Aka Ordinal regression model. The output variable is discrete in three or more classes with the ordering of the levels.

Customer Rating: extremely dislike, dislike, neutral, like, extremely like

Income level: low income, middle income, high income

**Intuition:** Two types of intuition to make you understand this algorithm. One is Geometric Intuition and second is Probabilistic Intuition. Here, I will discuss about Probabilistic Intuition.

**Note:** If you want to use logistic regression for classification problem then one thing keeps in mind that is your dataset should be linearly separable or most likely linearly separable. Because Logistic regression also try to find out a best fit line so that it can distinguish the two or multi-class.

**Best fit line**

**Simple Logistic Regression:**

There are some ways to distinguish class in logistic regression. One of them is perceptron trick. At first, I will discuss about perceptron trick. So, let’s discuss…

We know the formula about straight line for linear regression that is:

For logistic regression the formula will be:

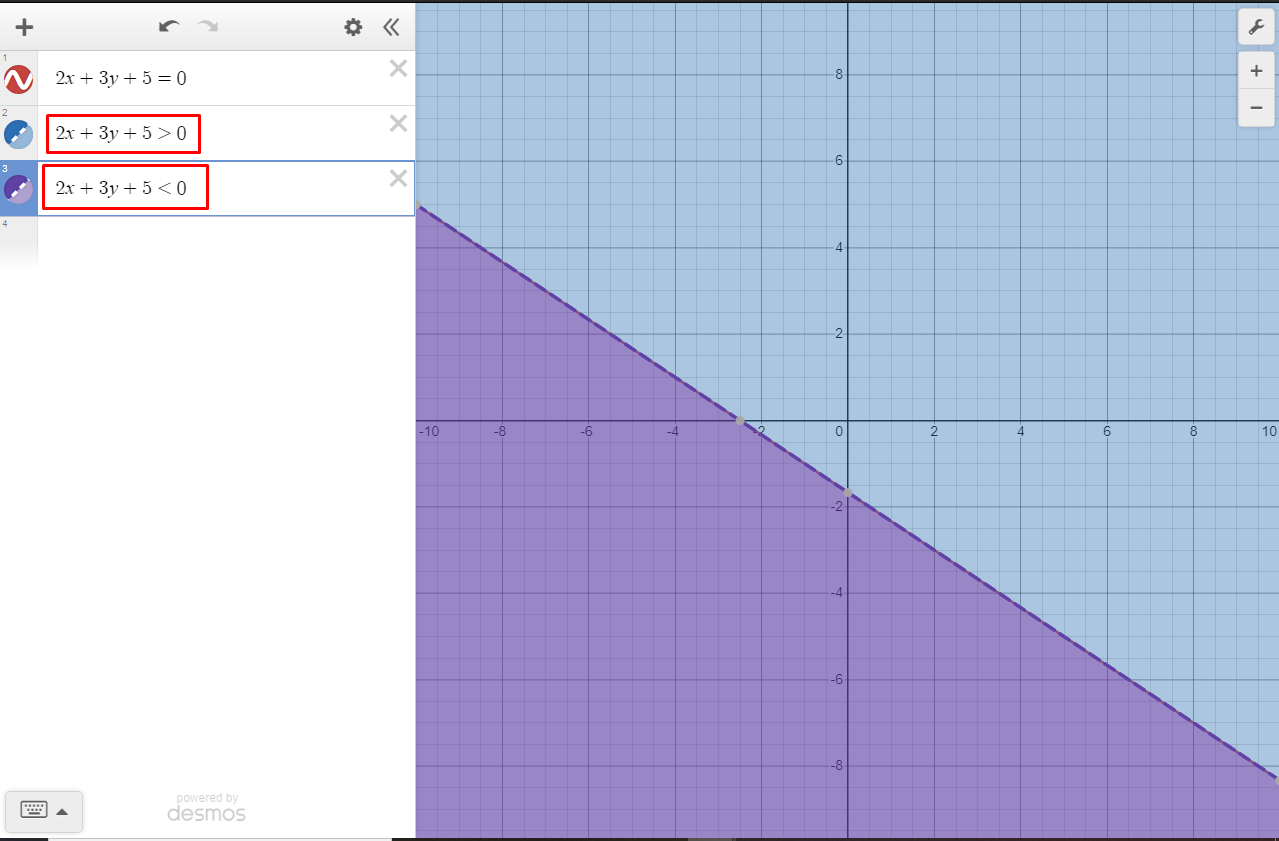
**We have to find out the value of ‘A’,’B’ and ‘c’.**

**Steps:**

* **At first, we assume random value of ‘A’, ‘B’ and ‘c’.**
* **We will use a loop (Ex: 1000 epochs).**
* **For every iteration we will pick a random data point.**
* **Based on that data point we will ask the line that the line is okay or not.**
* **If the data point in right side of the line, then line will be not change that means we won’t change the value of ‘A’, ‘B’ and ‘c’. If data point is missed classified then we will move the line towards the that point.**

**How to level regions:**

Now question easily arise how our line know that which side is positive regions and which side is negative regions. To identity this thing very easy. We can find out this thing from our line equations. Let’s give you a very good example.



If the straight-line equation value greater than zero then right side of the line will be positive region and vice-versa.

**How to transformation of the line:**

Suppose, some positive points missed classify and those points lie on negative regions and also some negative points lie on positive regions. Then how we will take it in positive points in positive regions and negative points in negative regions. To do it we have to change the value of **‘A’, ‘B’ and ‘c’.** So, we have to transform our line in such a way so that we can achieve our goal. I am showing the technique in below:

red region (-ve) (assume)

Green region (+ve) (assume)

(1,3,1) (assume)

(4,5,1) (assume)

(assume)

As you can see here, there are two points are missed classify. One red point lies in green region and one green point lie red region. Now we have to move the line. How we will do that? Very simple, let’s see. We will add **1** at last position of coordinate of the point then we will subtract if the point lie in the positive region and we will add if the point lie in the negative region. Let’s see how calculation is happen:

Now, based on these two equations our line will be move or transform. But in machine learning we are not taking large step or large moving. So, we will use a learning rate parameter. Then the formula will be:

That means we will multiply a small number (learning rate) with all coordinate value then we subtract with coefficient.

**Algorithm**

|  |  |  |  |
| --- | --- | --- | --- |
| X0 | X1 | X2 | y |
| 1 | CGPA | IQ | PLACED |
| 1 | 7.5 | 81 | 1 |
| 1 | 8.9 | 109 | 1 |
| 1 | 7.00 | 75 | 0 |

This is our toy dataset. We are using this dataset for an example. Our equation was:

For this dataset we can write:

Now we will change the equation in different notation:

Or

line we can easily write as a matrix dot product.

**Pseudo code of algorithm:**

* set learning rate.
* For i in range(epoch):

Randomly select a data point

If

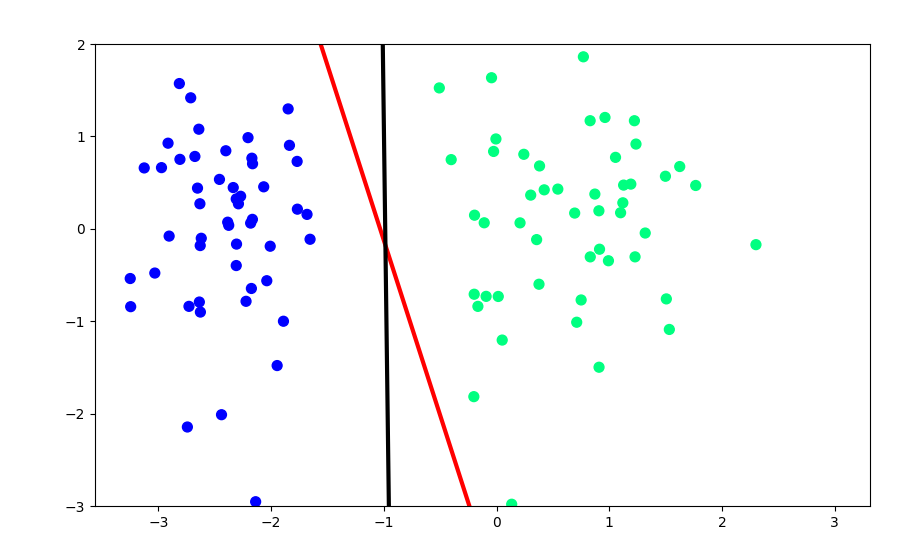
If

Used two condition we can use another way that is:

There are 4 conditions that a data point is correctly classified or missed classified the are:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | -1 |

If we put column value in our last equation then we will get previous two condition aquatically. That’s all about perceptron trick. But there are some limitations of perceptron trick. That why we will use **gradient descent** with **sigmoid function.**

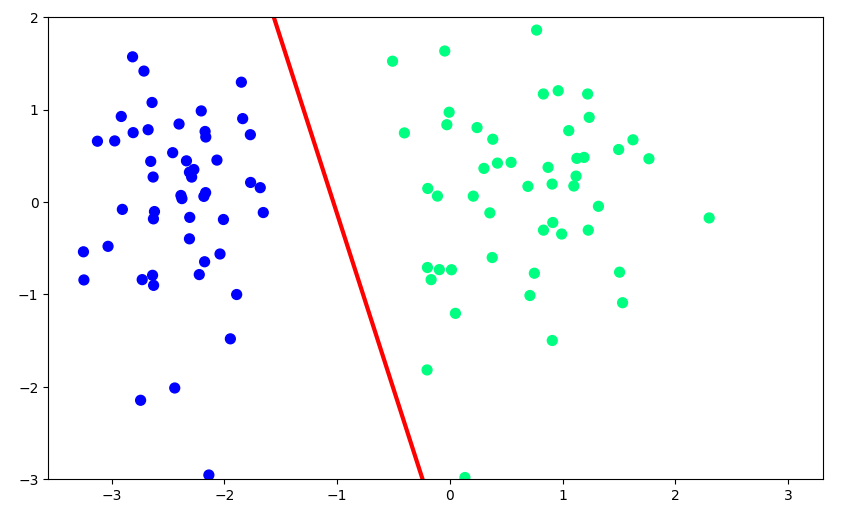


**To find out code:** [Click here](https://github.com/alaminbhuyan/ML-Algorithms-2023/blob/master/Algorithms/Logistic%20Regression/Perceptron%20Trick.ipynb)

In this picture the red line is using perceptron trick and the black line is scikit-learn learn library logistic regression algorithm. We can easily see there are some differences between two lines. Our perceptron trick line is not good as scikit-learn library logistic regression algorithm. So, what is the problem? Problem is we just consider missed classify data point to move our line. Here correctly classify data point is no say. Perceptron trick we just ask question by randomly select a data point that are you classify correctly or not. If correctly classify then we do nothing if not then we move the line.

**Possible solution:**

Now we will consider missed classify data points and also correctly classify data points. In this technique correctly classify points push the line and missed classify points pull the line toward the data point. The push and pull by point will depend how close or how far from the line. If a data point is missed classify and it lies near to line then it is slowly pulling the line towards that point. If a missed classify point lies far from the line then it will pull the line towards very strong magnitude. If a point is correctly classified and it lies near to the line then it will push the line very strong magnitude forward direction. If a point is correctly classified and it lies far from the line then it will slowly push the line forward direction.



Slowly push (correctly classify)

Strongly push (correctly classify)

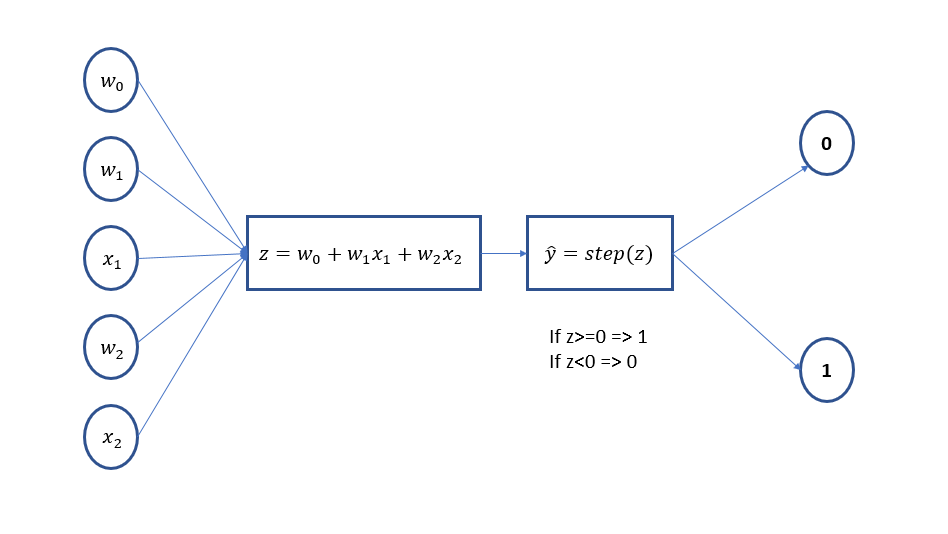
Slowly pull (missed classify)

Strongly pull (missed classify)

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 1 | 0 |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | -1 |

Our formula was:

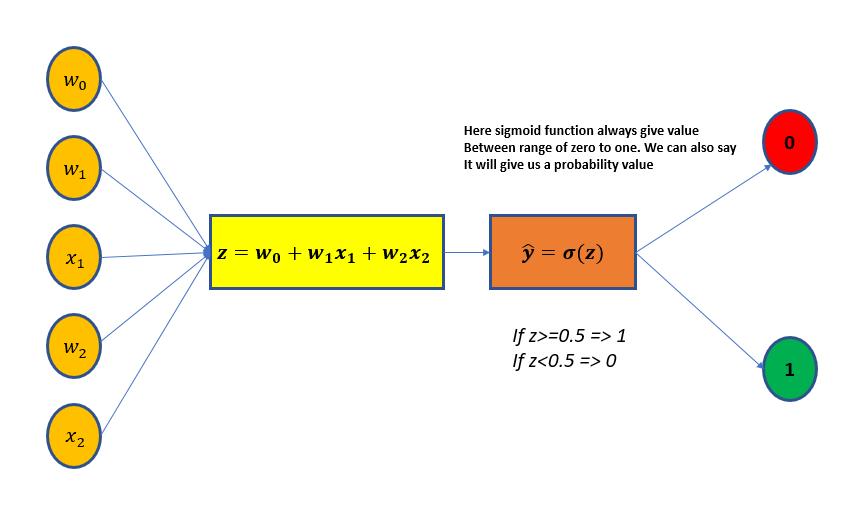
In this table we can easily see for first two row this will be zero. So that, old coefficient and new coefficient no change, they are same. So, our goal is to stop it to be zero. We have to use different function instead of step function.



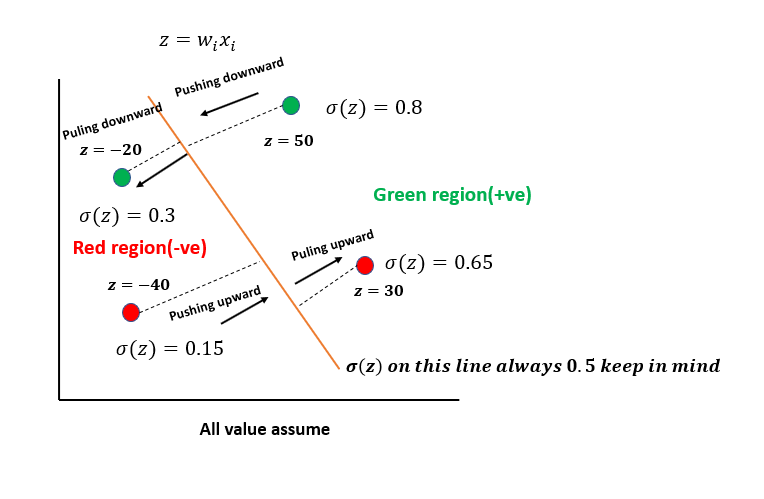
Look above diagram. Till now we have done this thing. Here, step function always gives zero or one. So for two cases will be zero. So, we have to replace the step function use here sigmoid function. The formula of sigmoid function is:

**We can easily understand this formula will give us value range of zero to one.**

If we used sigmoid function instead of step function then for two cases never be zero. Here, two case means if actual and predicted value is same in that time (actual - predicted) value will be zero. If we use sigmoid function for prediction (y\_hat) we will always get some value like, 0.56 or 0.12, it will never be zero. Now our diagram will be like this for prediction of a new query point:



Let’s prove that never be zero if we use sigmoid function. Here, I just take an example and assume all the thing for understanding purpose.



|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 0.8 | 0.2 |
| 0 | 0.65 | -0.65 |
| 1 | 0.3 | 0.7 |
| 0 | 0.15 | -0.15 |

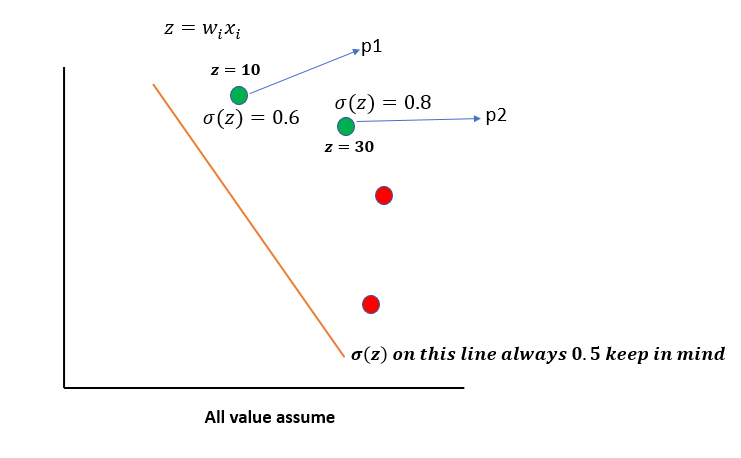
Now, based on our formula:

**We add some value with coefficient and this point is correctly classify. So, this point will be pushing the line to downward.**

**We subtract some value from coefficient and this point missed classified. So, this point will be pulling the line towards that point.**

**We add some value with coefficient and this point is missed classified. So, this point will be pushing the line to downward.**

**We subtract some value from coefficient and this point correctly classified. So, this point will be pushing the line upwards.**

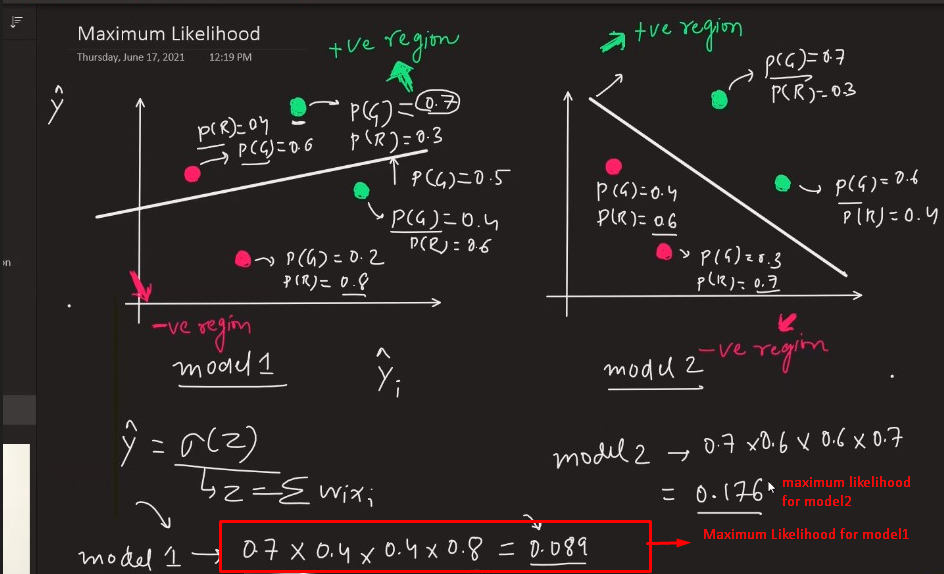


|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 0.6 | 0.4 |
| 1 | 0.8 | 0.2 |

**Here, so we can easily say the p1 will push the line very strongly than p2 point. Because we know if a point is correctly classified and if the point is near to the line than it will push the line very strongly than that point that are lies far away from the line.**

**Note:** we can’t do well only using sigmoid function we have to calculate loss function and also, we have to find out all exact coefficient value for that our loss function will be minimum. To minimize the loss function, we will use gradient descent algorithm.

**For Maximum Likelihood:**



We find out the probability for red and green point. For every point we find out the what will be the probability of red and what will be the probability of green a point. Then we calculate the maximum likelihood by multiplying the probability of two model. If the maximum likelihood probability of a model is larger than other model then larger maximum likelihood probability model is the best model. But here is one problem, as you can see above here we just multiply 4 number and the value is getting small. Now, you think if you have 300 value then how small will be your result. That why we will take cross entropy. Now what is cross entropy. Cross entropy is nothing but the summation of negative log of maximum likelihood. Here, we will try to minimize the cross entropy. It also know as log loss error function or binary cross entropy.

Here will take log:

We know:

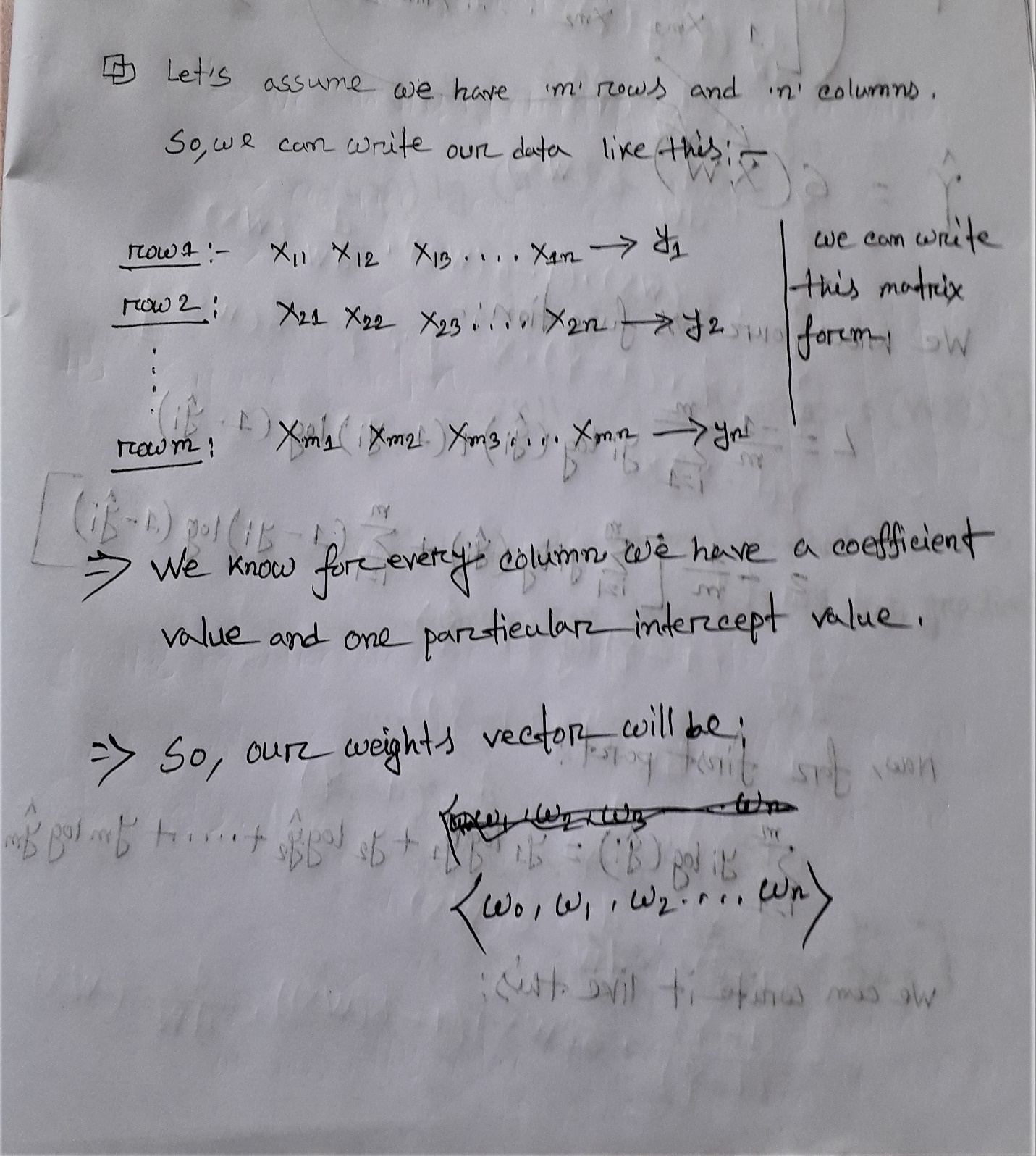
**We know log value of zero to one always negative number. So, we will calculate negative log.**

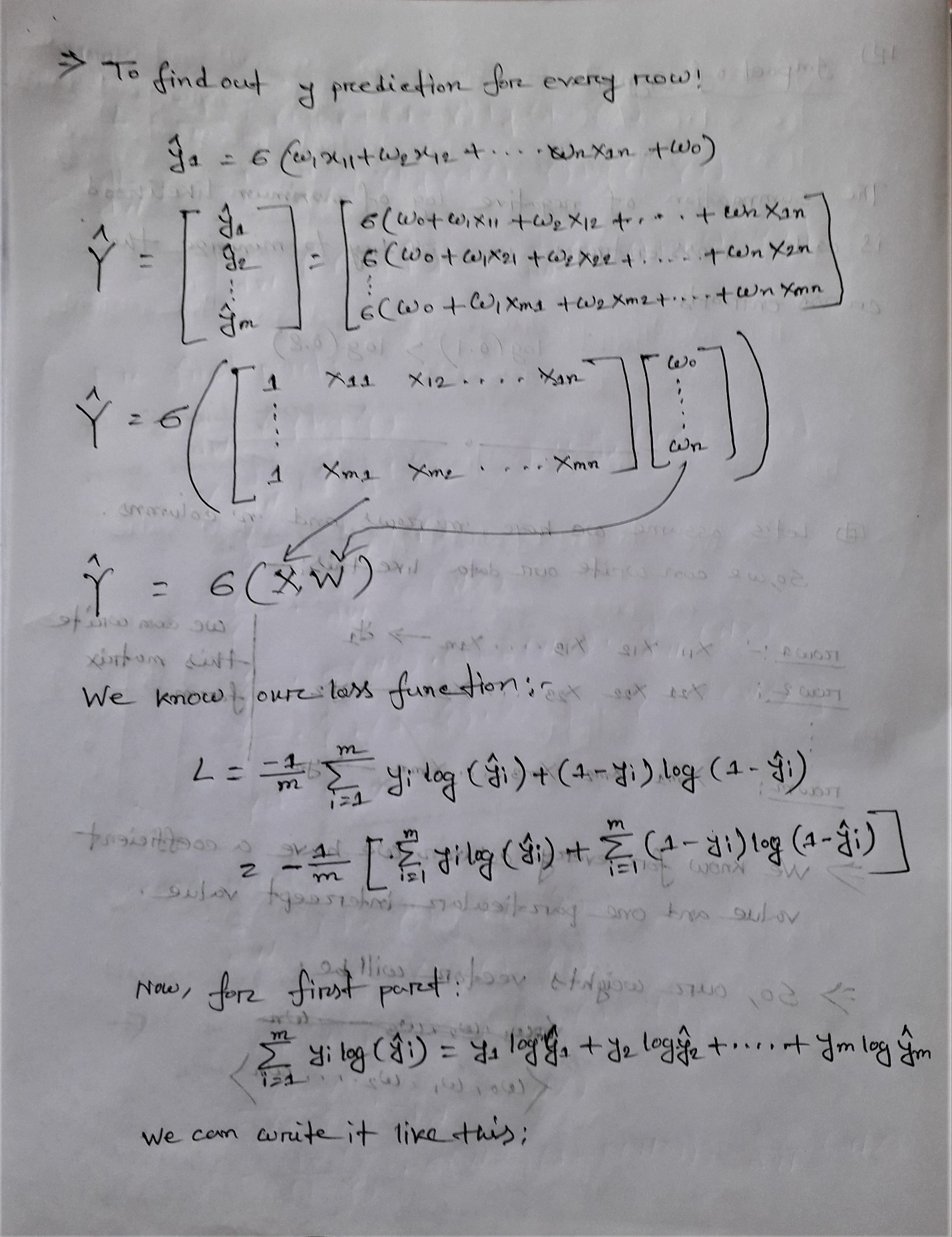
**We have to minimize of log loss or cross entropy because log (0.1) > log (0.8).**

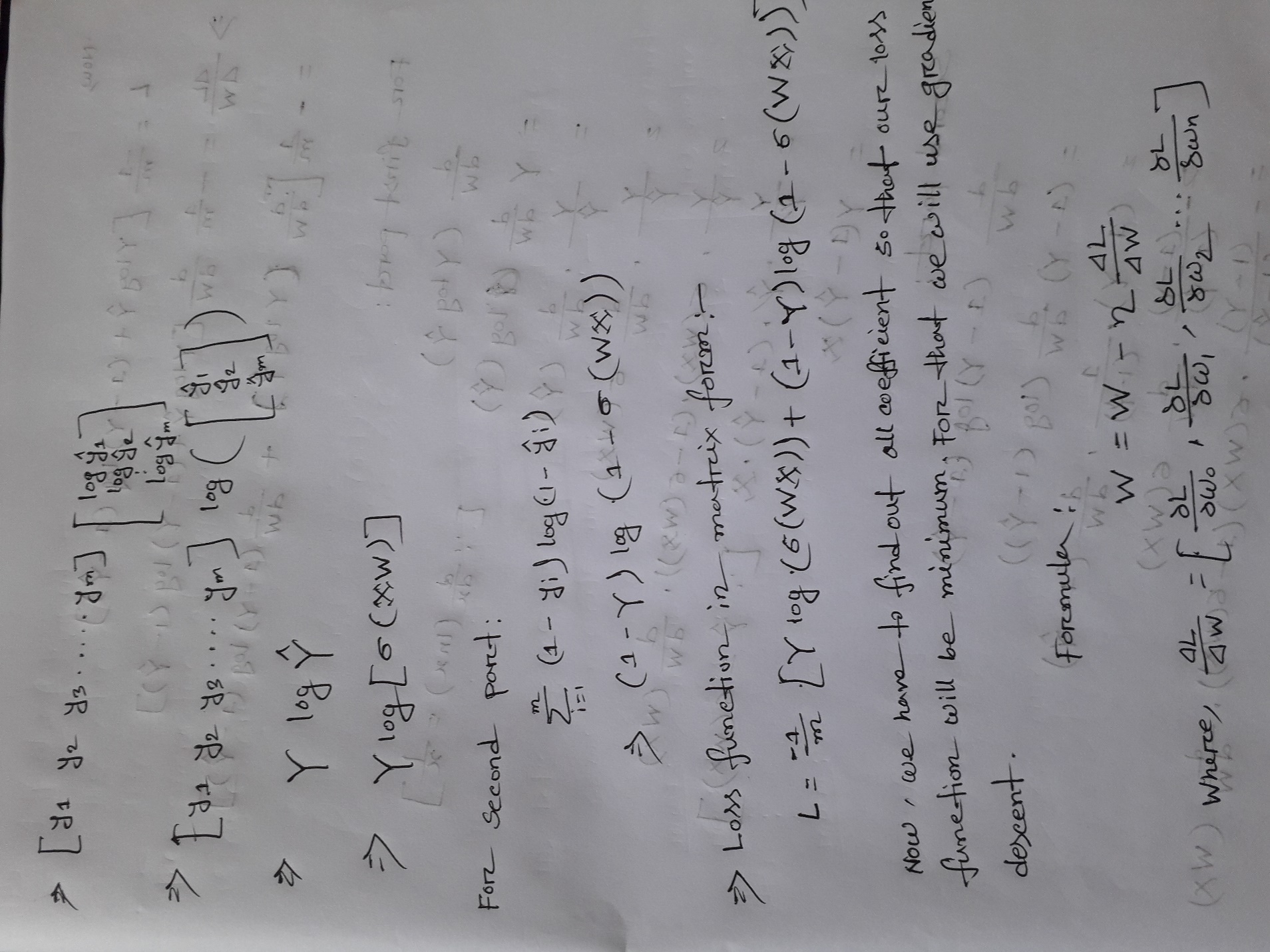
Or **mean log loss:**

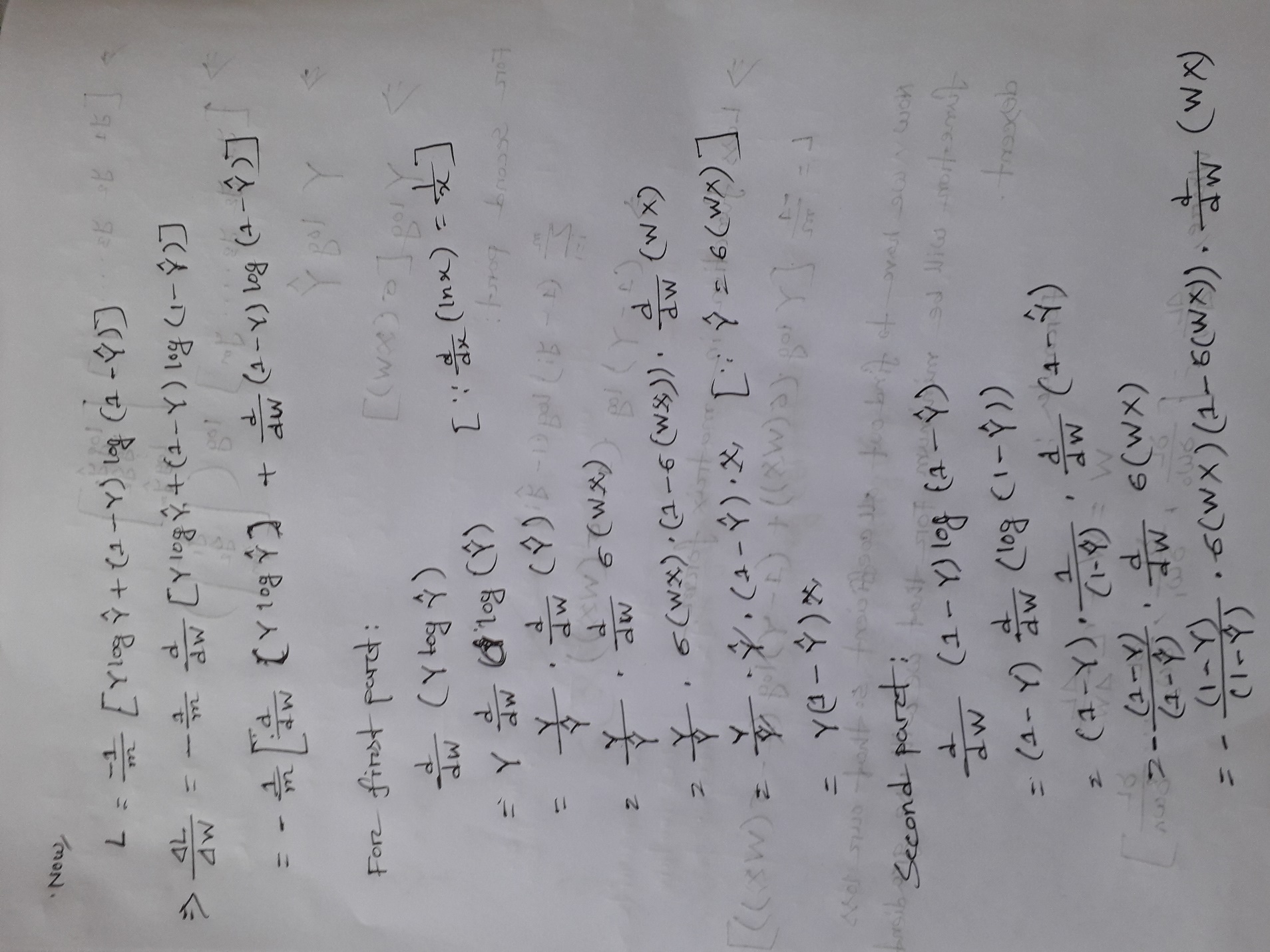
Now, our goal is to minimize the loss function **using gradient descent**. That means we have to find out the coefficient vlaue so that our loss function will be minimum.

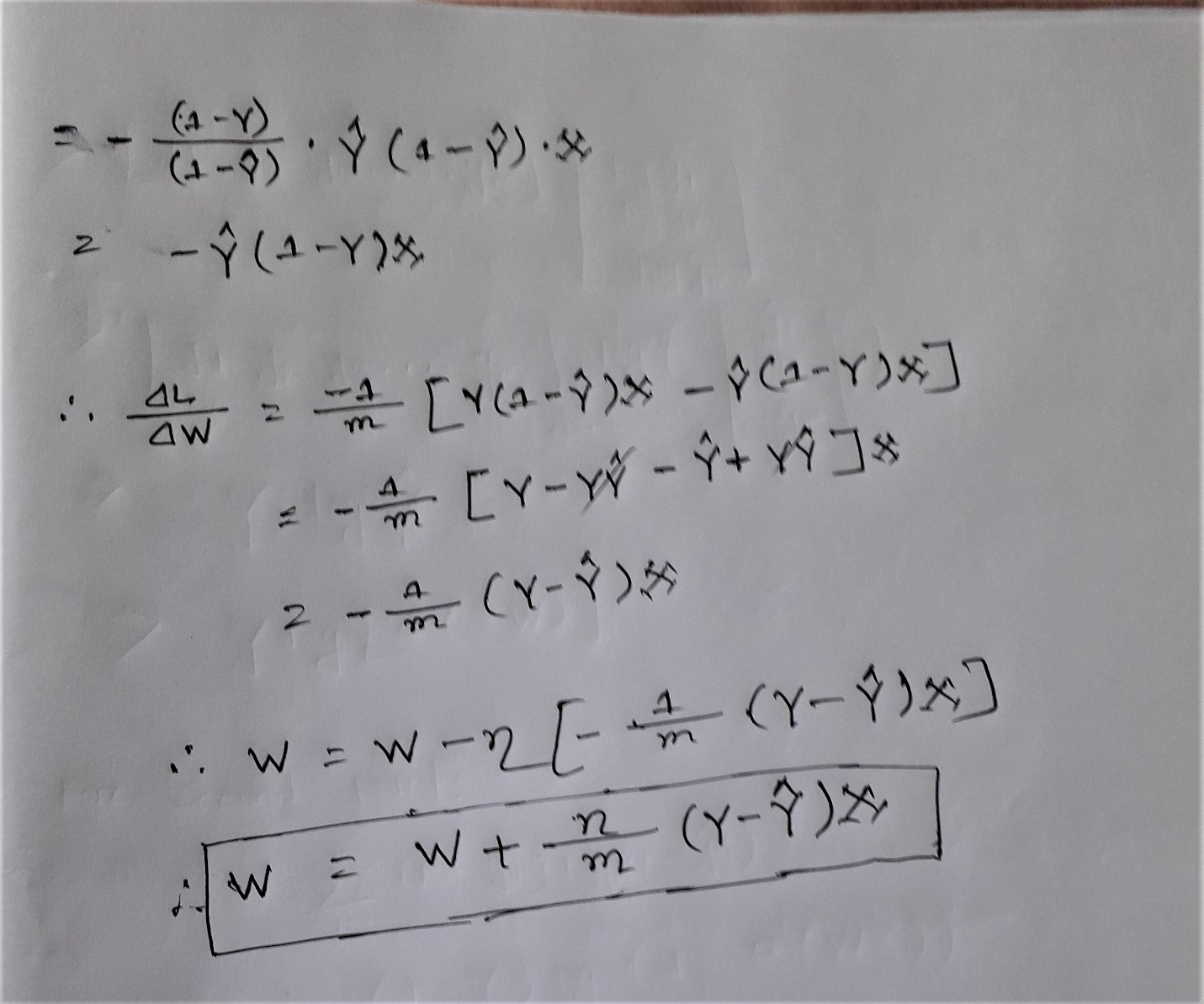
**Gradient Descent**





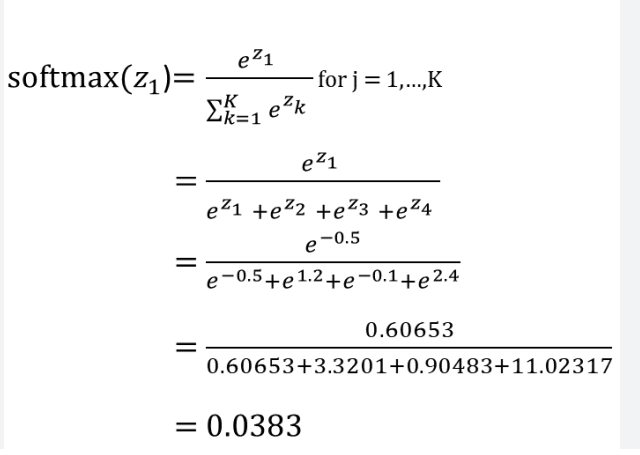






**Multiclass Logistic Regression or Multinomial Logistic Regression:**

For multinomial logistic regression we used **SoftMax function**. SoftMax function give us probability of every class. The formula is:



**Training Intuition:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CGPA** | **IQ** | **PLACE** | **Class=1** | **Class=2** | **Class=3** |
| **7** | **98** | **1 (yes)** | **1** | **0** | **0** |
| **6** | **87** | **2 (No)** | **0** | **1** | **0** |
| **5.6** | **78** | **3 (Optout)** | **0** | **0** | **1** |

We just encode our target class and create three columns for three classes. Now it is a binary logistic problem. Then we will train three different logistic regression model for [cgpa, iq, class=1], [cgpa, iq, class=2], [cgpa, iq, class=3]. After training we will get nine coefficient they are: ,,. If the number of classes increase our columns will be increase then number of coefficients will be increase.

**Prediction:**

Let, our new query point is: {7, 70}

Now we will pass this ‘z’ value to SoftMax function:

Let’s suppose the output will be: then our predicted class will be ‘yes’ because its probability is higher than others.

Note: Of course, here also apply gradient descent and update the coefficient value. But in this process there is a problem that is as number of class is increase this approach will be slow because our number of column will be increase. So, scikit-learn library used different approach. They use different loss function so that this multiclass classification does at a time.

The loss function is:

The second summation is of all class for every row.

