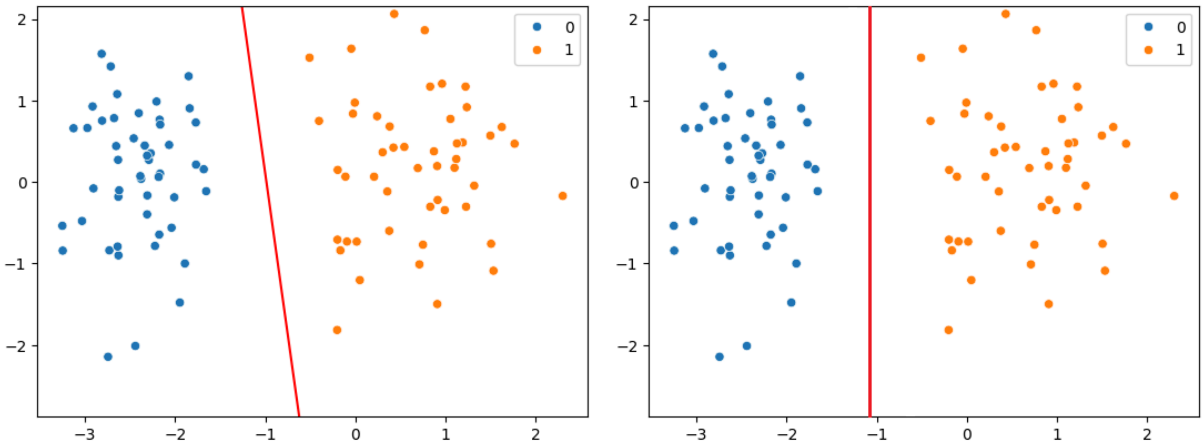
Introduction

In Logistic regression using perceptron, the problem we faced :



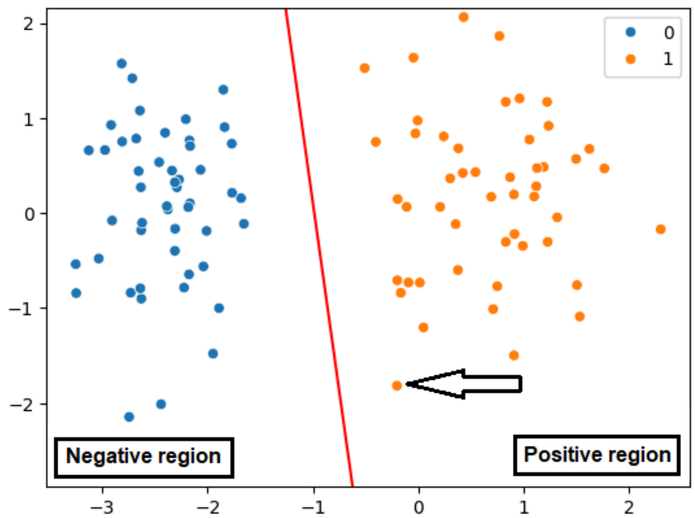
Our Perceptron Model Sklearn Model

So on the left, we want to make the regression line straight as much as possible.

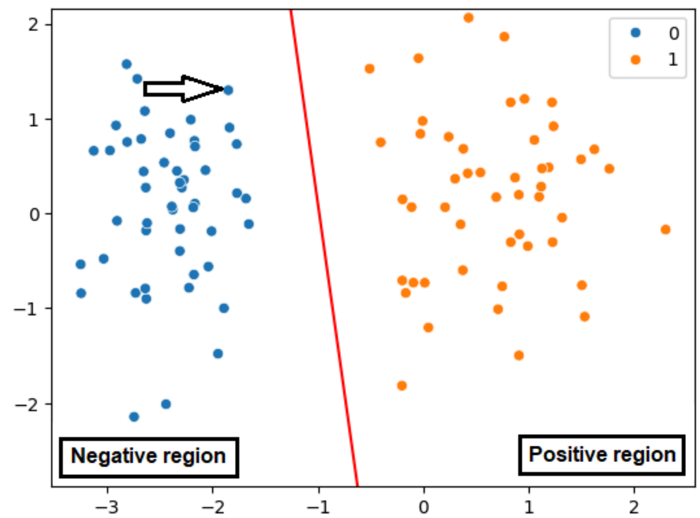
In the Perceptron algorithm, we move the red line for all the misclassified points.

$W_{new} = W_{old} + (y - \hat{y}) * \eta * X[random\_index]$ (from perceptron algorithm)

In that equation, when y and $\hat{y}$ are same (means the Point is not misclassified), $W_{new} = W_{old}$ i.e. the Red line doesn't move but now let's say even if the actual output(y) is same as the output returned by the Red line($\hat{y}$), we want to move the red line i.e.



That orange point is correctly classified but we want to push the red line towards its opposite region, Negative region to make the red line straight as much as possible.



That blue point is correctly classified but we want to push the red line towards its opposite region, Negative region to make the red line straight as much as possible.

That means $(y - \hat{y})$ can't be 0 if we want to move the red line even for correctly classified points. That also means y and $\hat{y}$ can't be the same value at the same time. y is fixed as it is given by the dataset. We can only bring changes in $\hat{y}$ as we've to calculate it manually.

y	$\hat{y}$	y - $\hat{y}$	line_move_towards
0	0	0	remains same
1	1	0	remains same
1	0	$\boxed{1}$	Negative region
0	1	$\boxed{-1}$	Positive region

From this :

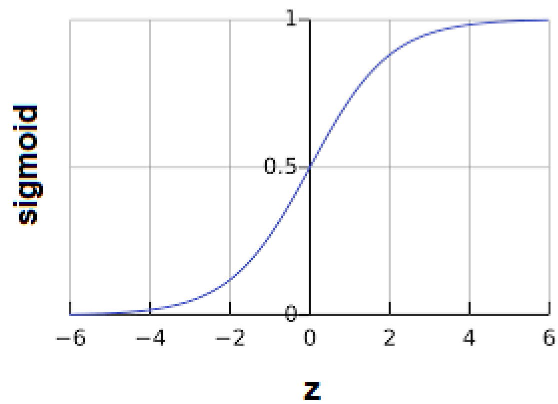
y	$\hat{y}$	y - $\hat{y}$	line_move_towards
0	0	0	remains same
1	1	0	remains same

To this : (this is what want) [0.2 and 0.7 are assumed)

y	$\hat{y}$	y - $\hat{y}$	line_move_towards
0	0.2	-0.2	Positive region
1	0.7	0.3	Negative region

To bring such changes in $\hat{y}$, we use here **sigmoid** function :

$$\text{Sigmoid}(z) = \frac{1}{1 + e^{-z}}$$



Why sigmoid? Because for z = 0, sigmoid returns 0.5. As the value of z increases, the sigmoid output goes close to 1 but never 1. As the value of z decreases, the sigmoid output goes close to 0 but never 0. As the sigmoid never becomes 0 or 1, so $\hat{y}$ will never be 0 or 1 AND thus ($y - \hat{y}$) will never be 0.

Wait! How did we calculate $\hat{y}$ in perceptron algorithm?

def line_decison(line):
    return 1 if line > 0 else 0 # i.e. return 1 if the Point is in Positive Region, else 0.
$\hat{y}$ = line_decison(WX)   (W = [w0, w1, w2 … wm]
                                          X = [x0, x1, x2 … xm] for one point/row)

The new changes would be, sending the Line to a sigmoid function :

def sigmoid(line):
return 1 + (1 + e^{-line})
$\hat{y}$ = sigmoid(WX)

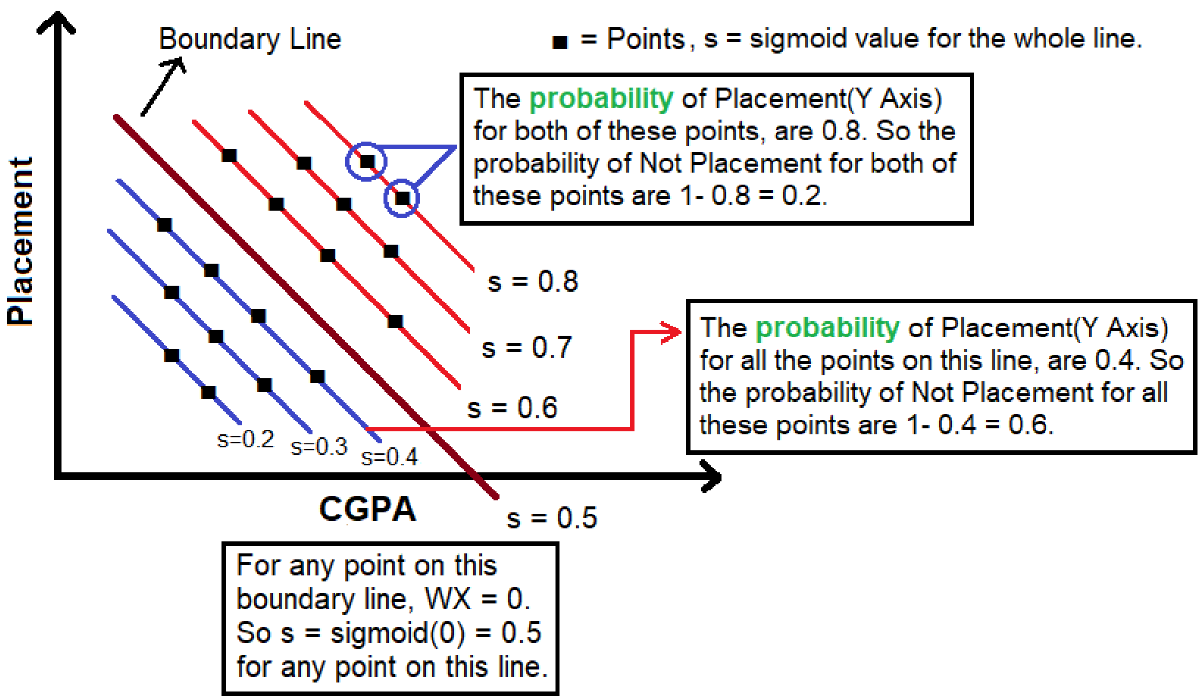
More Insights of Sigmoid Function

Watch how **sigmoid represents probability** and how the sigmoid value has an impact on pushing the line from [15.56 to 38:25](). (You may skip it as I've noted the below info from that campusx video)

Sigmoid Values represent Probabilities :

—--

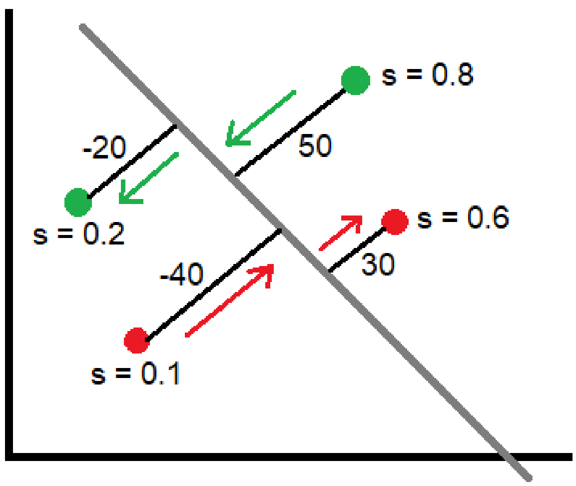
Because the sigmoid value's range is (0, 1), it can be described as Probabilities.



You may ask, why is the Sigmoid Value the same for all the points on a line.Okay, what do we feed the Sigmoid Function? The line(WX) i.e. `sigmoid(np.dot(W, x))`. Then sigmoid returns a sigmoid value for the whole line i.e. for all the points on that line.

Sigmoid value's impact on 4 type points :

—---



y - ŷ

----

1 - 0.8 = 0.2
0 - 0.6 = -0.4
1 - 0.2 = 0.8
0 - 0.1 = -0.1

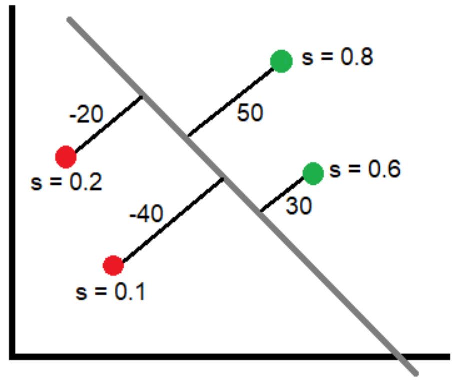
As you can see, there is no 0 output.

0.2 and 0.8 are **positive**. That means the line will go **down** from its previous position, either towards the point or the opposite of the point.

-0.4 and -0.1 are **negative**. That means the line will go **up** from its previous position, either towards the point or the opposite of the point.

Sigmoid value's magnitude :

—--------------------------------------



y - $\hat{y}$
----

1 - 0.8 = 0.2 → **w = w + 0.2 * ηx**
1 - 0.6 = 0.4 → **w = w + 0.4 * ηx**

+ means the line will go down from its previous position. if ηx = 10, then

0.2 * 10 = 2 and 0.4 * 10 = 4

This 2 and 4 are for the distance 50 and 30 respectively.
So lesser the distance, greater the force to push below.

0 - 0.2 = -0.2 → **w = w - 0.2 * ηx**
0 - 0.1 = -0.1 → **w = w - 0.1 * ηx**

\- means the line will go **up** from its previous position. if w = 10 and ηx = 10, then

10 - 0.2 * 10 = **8** and 10 - 0.4 * 10 = 6

This 8 and 6 are for the distance -20 and -40 respectively.
So lesser the distance, greater the force to push up.

Ultimately for **correctly classified point** : The **lesser** the distance, the **larger** the force that point use on the line.