**LOGISTICS REGRESSION**

Logistic regression is the multi-class classification technique used in Machine learning to predict the class label i.e, Yi where Yi belongs to some class but not real value.

Basically it is used for i)Binary classification but we can extend this to ii)Multiclass classification by “one vs rest” or “one vs all”

Why we are using logistics regression in binary classification instead of linear regression?

If we have outliers in our data , in linear regression the line equation will not be the best fit line(completely deviated) i.e, it will produce more error because of outlier .So that we are applying squashing technique to dampen the error which is called logistic regression(here we are applying sigmoid function)

**Assumption**

The biggest assumption before solving it is our data is linearly separable or almost linearly separable.

**Note:**

If we are having this assumption then the model will be call parametric model

If there is no assumption then it is non parametric model

Here the line equation is

y=mx+c

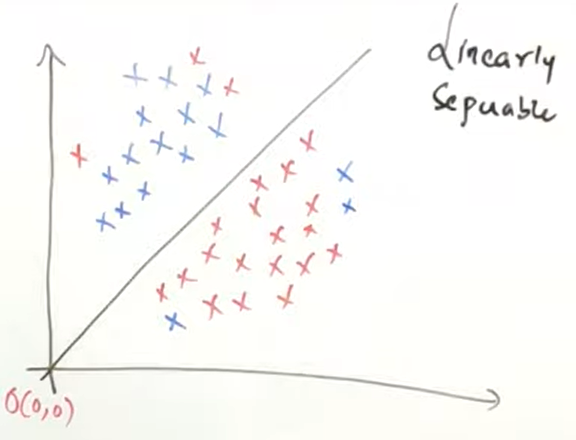
m=slope , c=intercept

or

y=Bo+B1x

or

y= W(Transpose)X + b



We need to find or apply W(Transpose)or m (slope) to give the best fit line.i.e, W should be optimal

The y(output) value considered as +1 or -1 (not 1 or 0)

Positive points considered as +1

Negative points condidered as -1

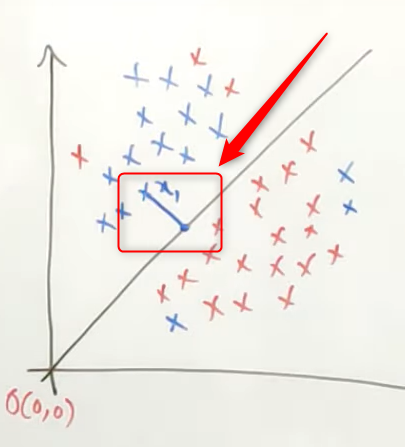
consider the line passing through origin then the intercept is 0, so the equation will be

Y=mx or y =W(Transpose)X – which is distance between the data point and the plane

The distance(W(Transpose)X) will be positive if we calculate above the plane and the distance(W(Transpose)X) will be negative if it is below the plane

**Case 1**

Consider one +ve datapoint which is above the plane

****

The output y is +1 because the positive points considered as +1

Y=+1

The distance (W(Transpose)X) is positive because the data point is above the plane

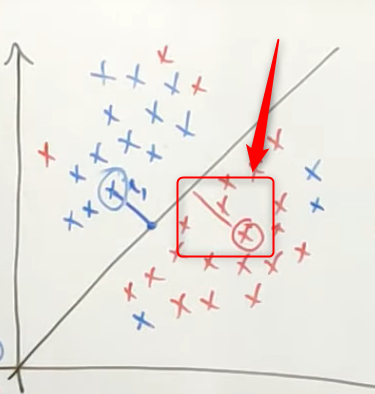
WtX>0

So the output \* distance will be positive value

**Y\* WtX>0 – classified correctly**

**Case 2**

Consider one -ve datapoint which is below the plane



The output y is -1 because the negative points considered as -1

Y=-1

The distance (W(Transpose)X) is negative because the data point is below the plane

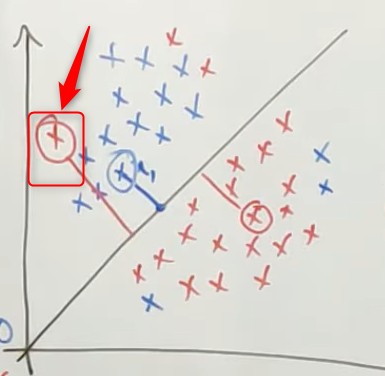
WtX<0

So the “output \* distance” will be positive value i.e (-1)\*(-1)=+1

**Y\* WtX>0 – classified correctly**

**Case 3**

Consider one -ve datapoint which is above the plane



The output y is -1 because the negative points considered as -1

Y=-1

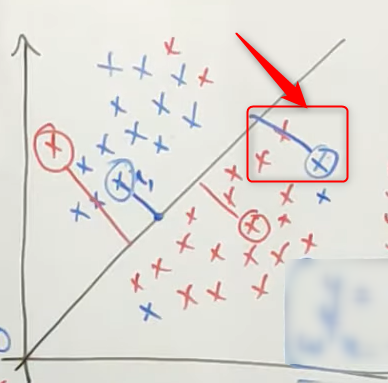
The distance (W(Transpose)X) is positive because the data point is above the plane

WtX>0

So the “output \* distance” will be negative value i.e (-1)\*(+1)=-1

**Y\* WtX<0 – Misclassified**

Consider one +ve datapoint which is below the plane



The output y is +1 because the positive points considered as +1

Y=+1

The distance (W(Transpose)X) is negative because the data point is below the plane

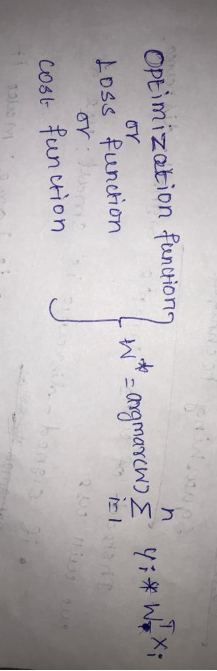
WtX<0

So the “output \* distance” will be negative value i.e (+1)\*(-1)=-1

**Y\* WtX<0 – Misclassified**

**Observation**

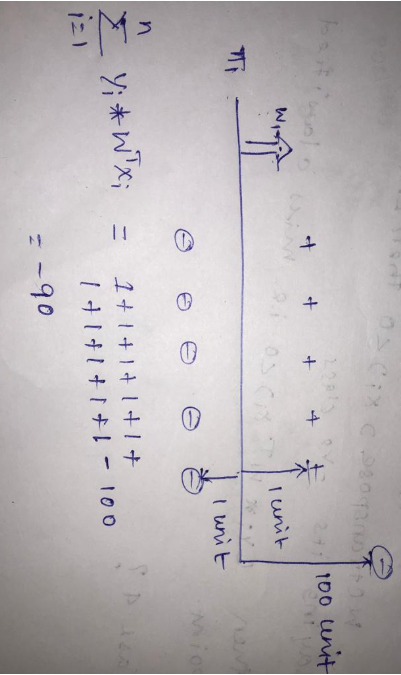
We need to have maximum classified data points or minimum misclassified data points , (maximum positive points or minimum loss )then only the line is considered as best fit line which is separating the data points correctly.

****

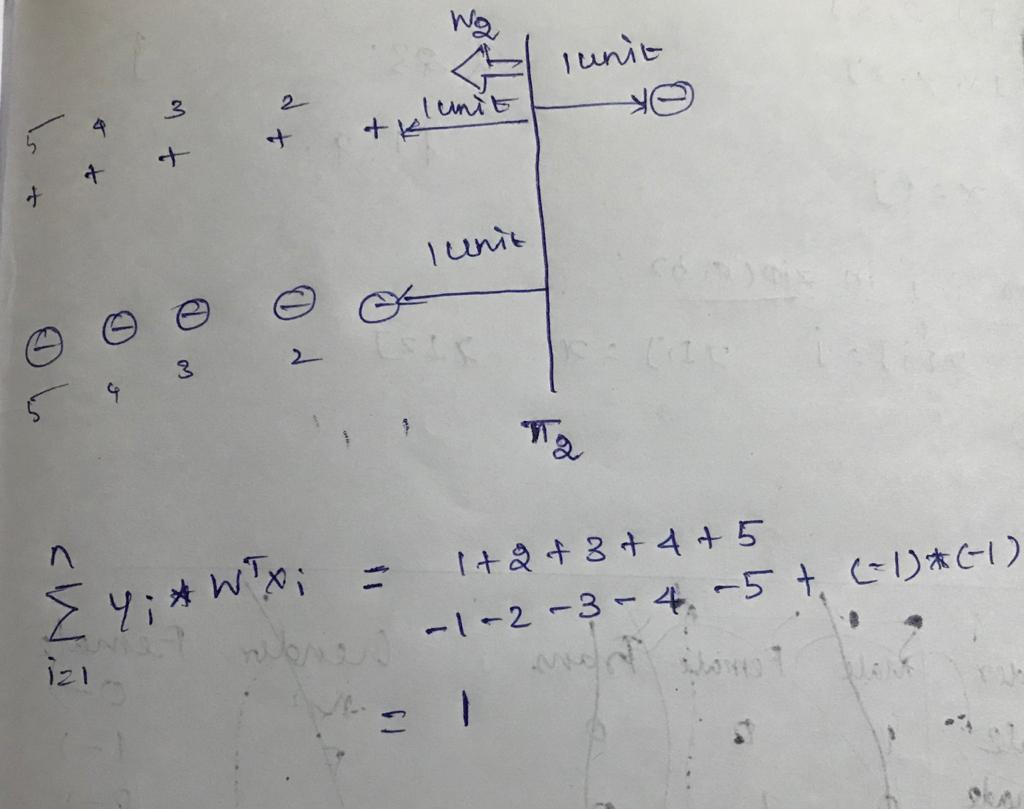
Update w that we get the best fit line i.e, w should be optimal

**Impact of the outliers**

**Case1**

****

**Case 2**



Considering the both cases logically case 1 line is best fit line which is classifying most of the data points correctly but due to one outlier, the result coming in negative. So it will neglect case 1 best fit line

To remove the effect of outlier we need to apply squashing technique

**SQUASHING TECHNIQUE**

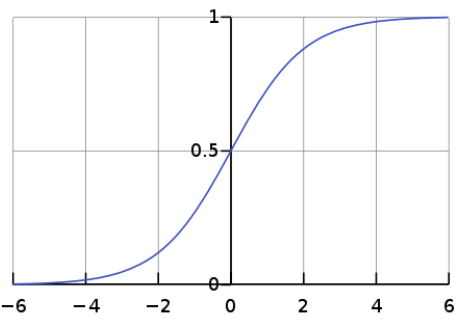
Instead of using simple signed distance we will use

* If signed distance is small use as it is
* If signed distance is large make it as small value and use it

We want a function when its value is small :increase it linearly and when its value becomes large tapper it off

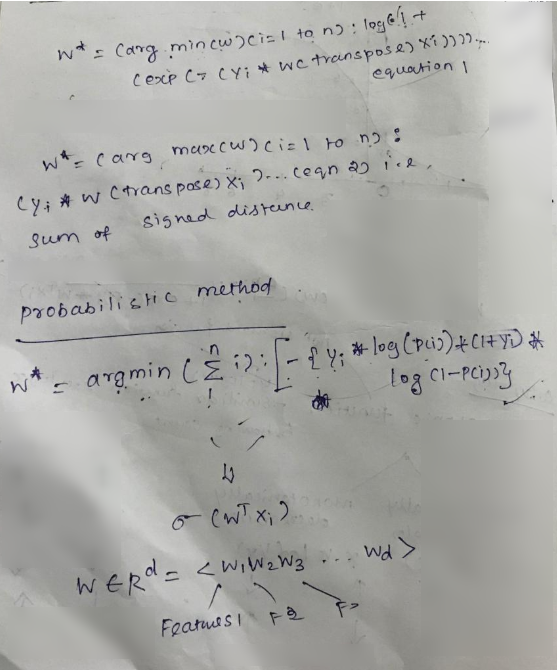
One such function is sigmoid function

Another reason for choosing sigmoid function , we can differentiate sigmoid function



**Sig(t) = 1/(1+e-t)**

Sigmoid function will return the value which is monotonically increasing



**OVERFIT VS UNDERFIT VS GENERALIZED-FIT**



By looking at the graph on the left side we can predict that the line does not cover all the points shown in the graph. Such model tend to cause underfitting of data .It also called High Bias.

Note:- Bias – error in training dataset

Variance – error in test dataset

Where as the graph on right side, shows the predicted line covers all the points in graph. In such condition you can also think that it’s a good graph which cover all the points. But that’s not actually true, the predicted line into the graph covers all points which are noise and outlier. Such model are also responsible to predict poor result due to its complexity.It is also called High Variance.

Now, Looking at the middle graph it shows a pretty good predicted line. It covers majority of the point in graph and also maintains the balance between bias and variance.

**REGULARIZATION**

Regularization is a process of adding bias or penalty or loss to our model in order to avoid overfitting

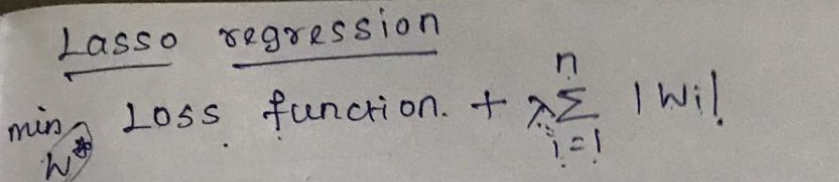
If we have many feature Overfit will happen

If we have less feature then underfit will happen

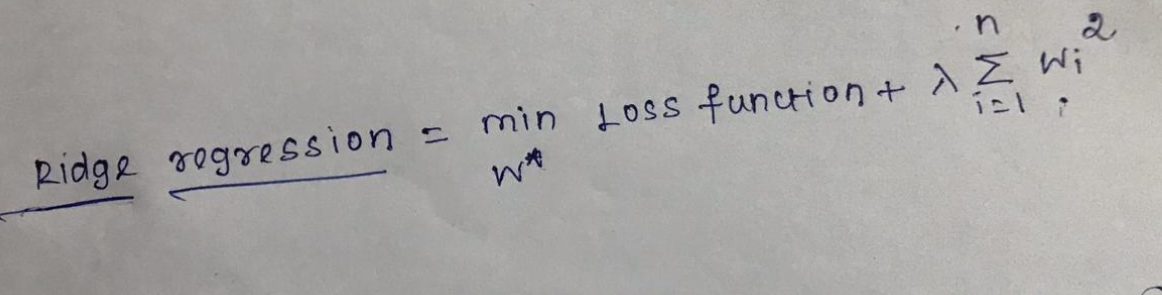
To solve the problem of overfitting in our model we need to increase flexibility of our model. But too much of his flexibility can also spoil our model, so flexibility should such that it is optimal value. To increase flexibility we can use regularization technique.

They are three types of regularization technique to overcome overfitting.

a) L1 regularization (also called Lasso regularization / penalization.)



b) L2 regularization (also called Ridege regularization/ penalization.)



c) Elastic net

**LOGISTIC REGRESSION**

1)Initialize the weight

2)Calculate loss

3)dL/dw updation step

Wnew = Wold – dL/dW

4)Repeat updation until convergence