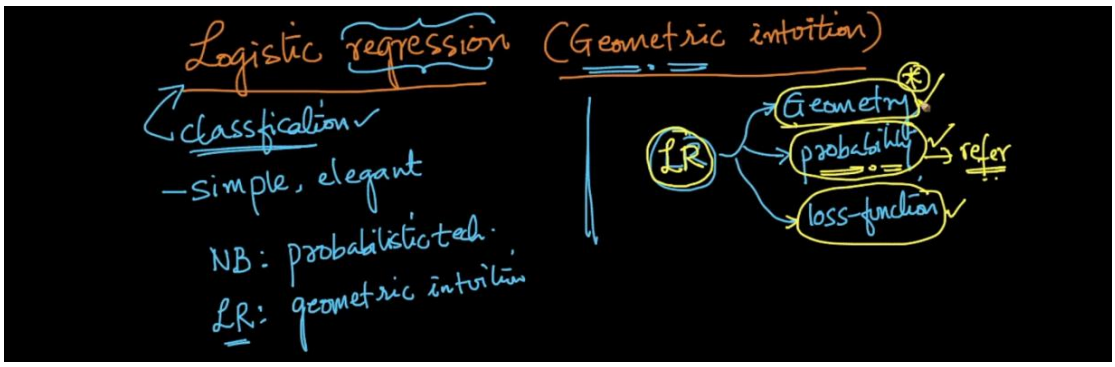
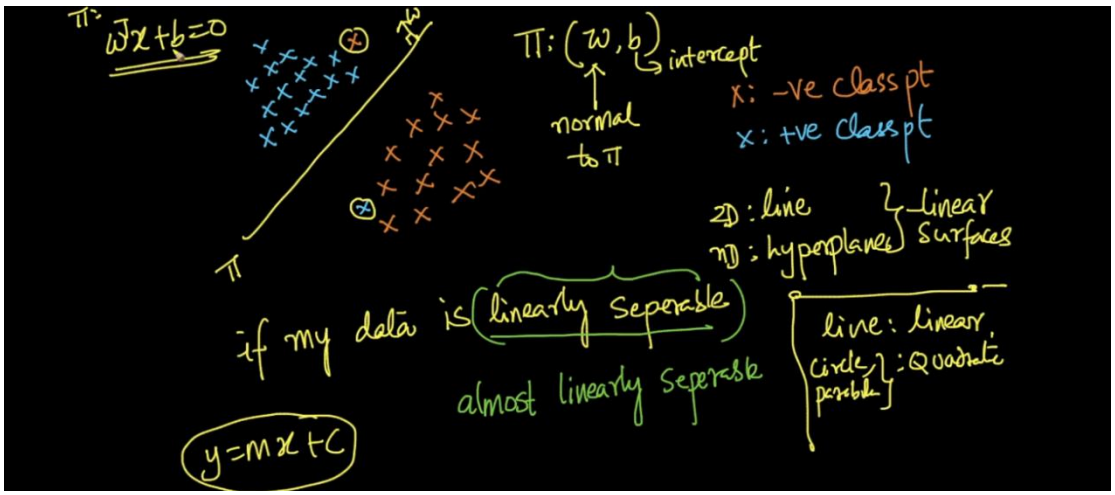
Logistic Regression :

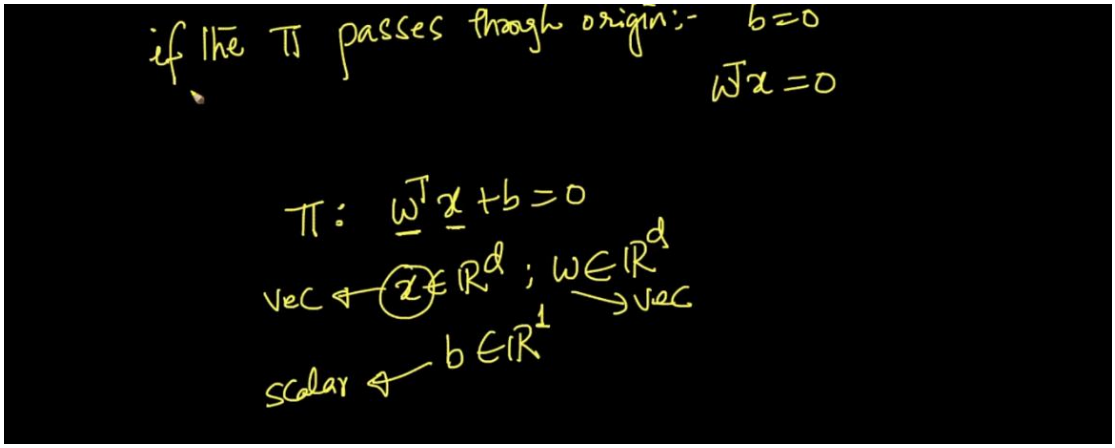


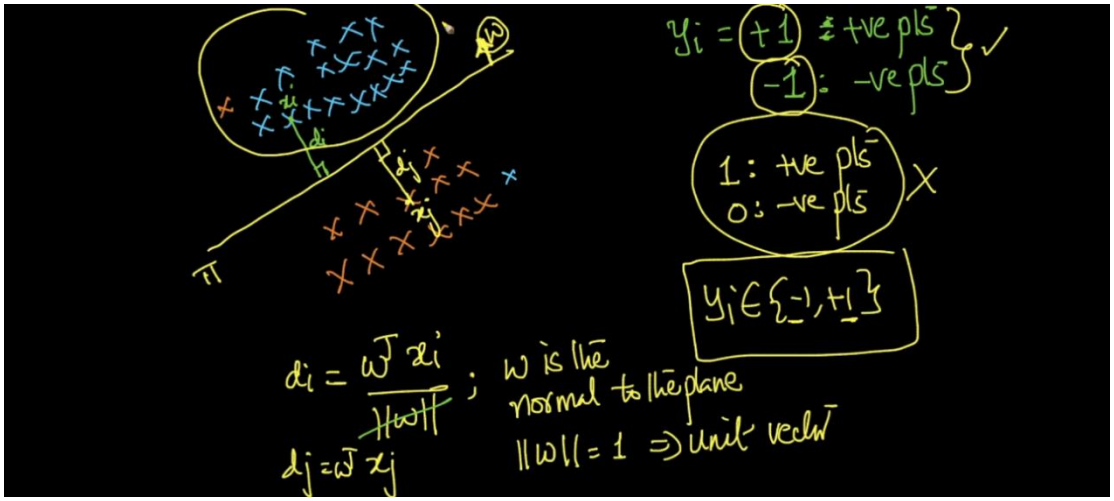
So in logistic regression our job is to chose appropriate plane means w .



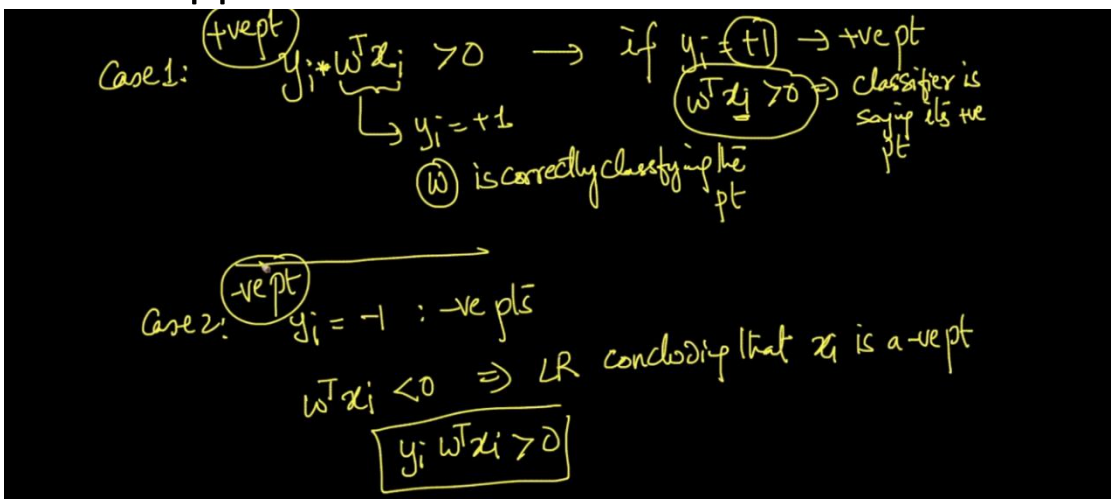
W t x + b if line passing through then b will be 0 .

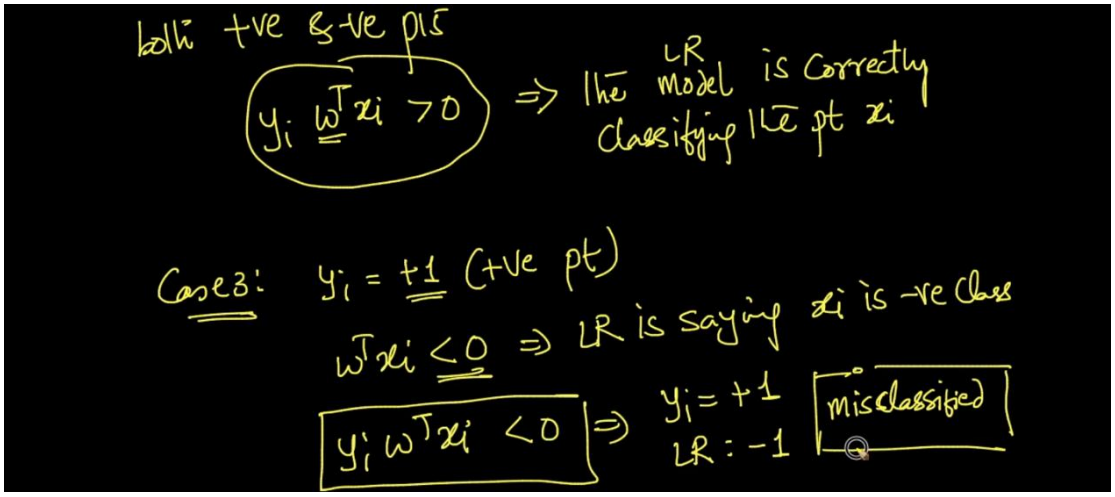
So main task is to find perfect plane which can easily separate data .

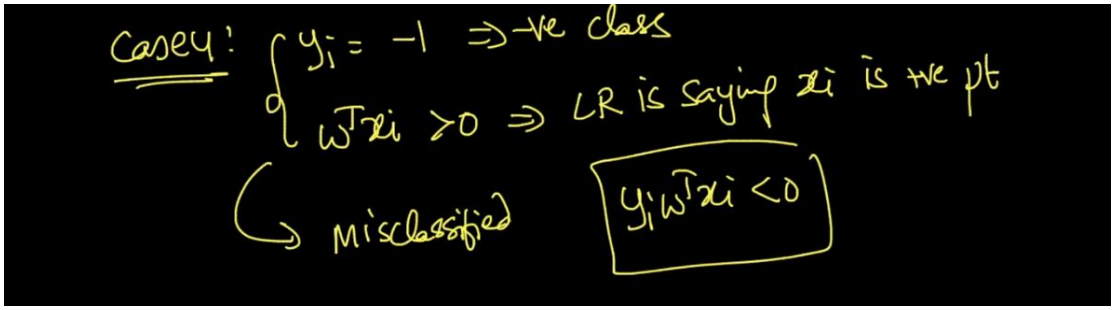




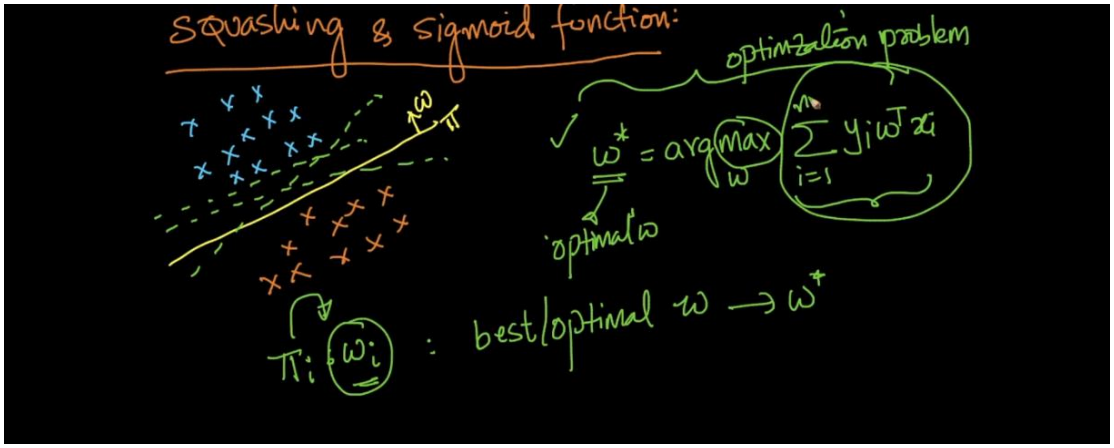
See above image lets take new 2 points x1 and x2 .now x1 is in same direction of w hence it will + and x2 opposite .







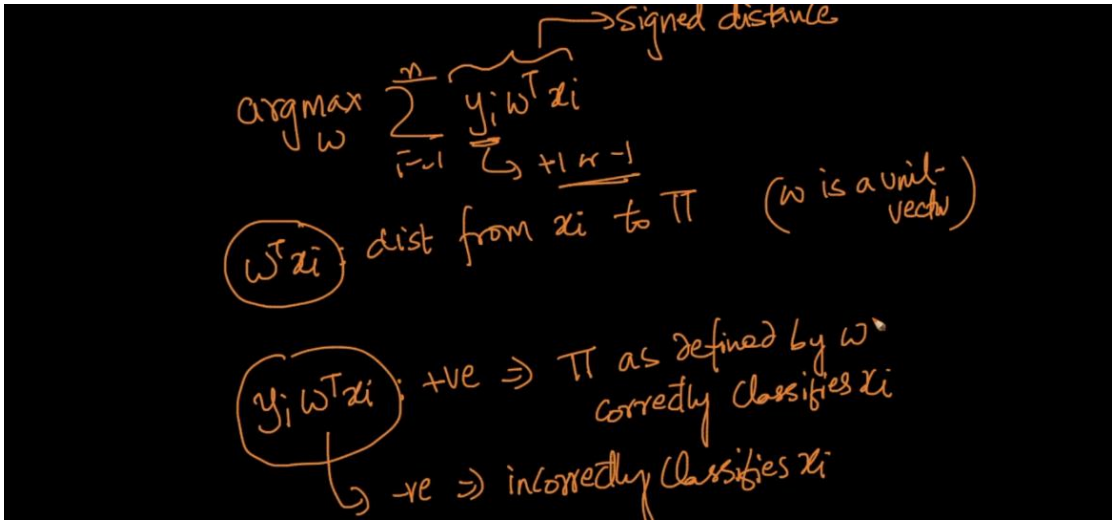
So finally we need best w so we can separate max points .



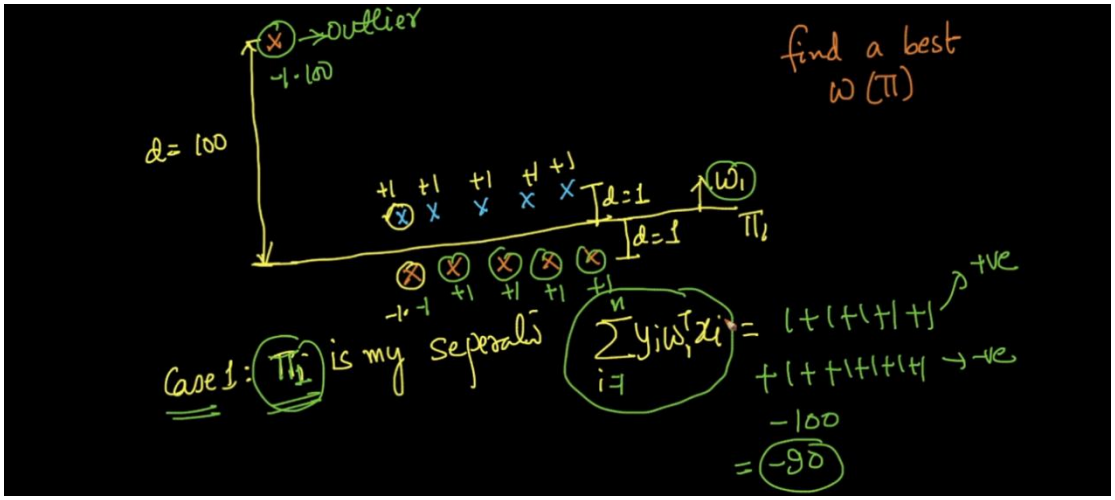
So see above image we found function of w we want to find max signed sum so we will get plane n good direction that's what we study .

But there is a big problem of this concept with out l .it will perform very very bad . lets see below images .

Y will be +1 or -1

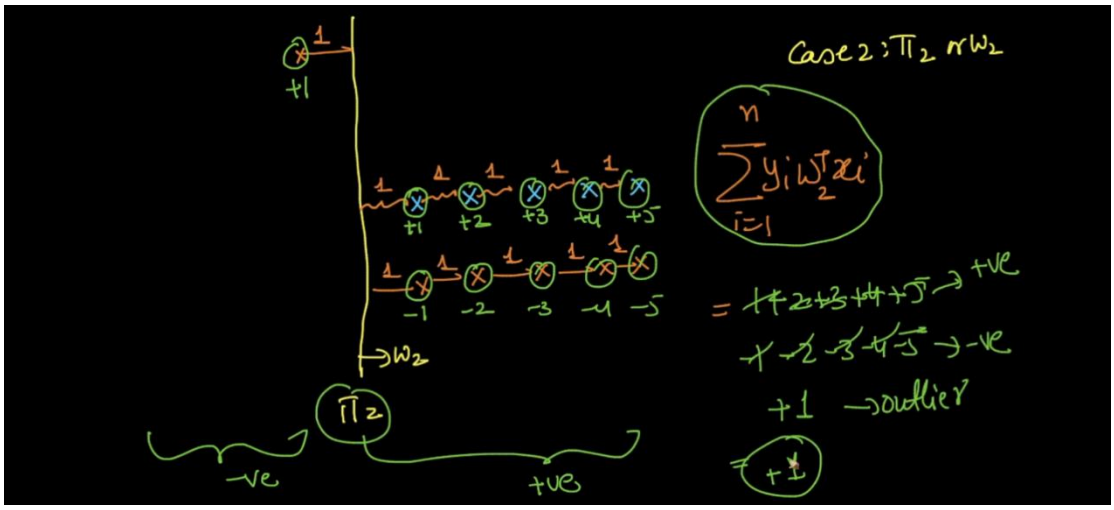


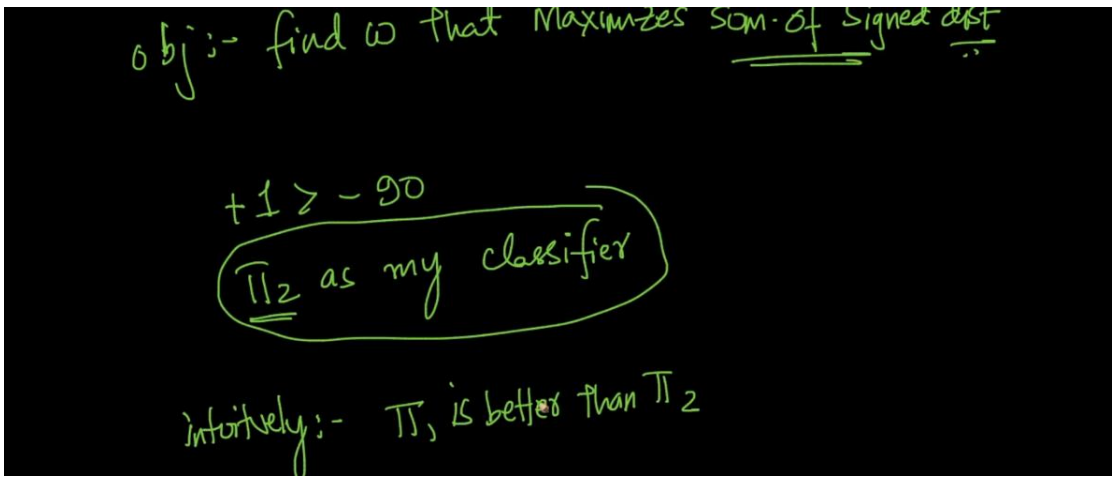
See case 1 :



Here we get very very small signed sum just because of only 1 out l.

Since we get very bad value we will move or plane so next we get :

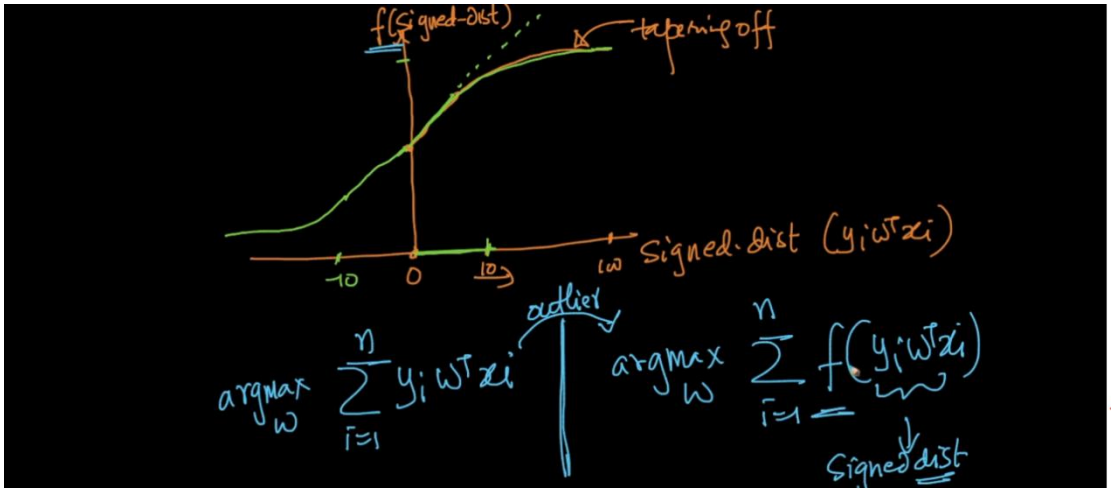




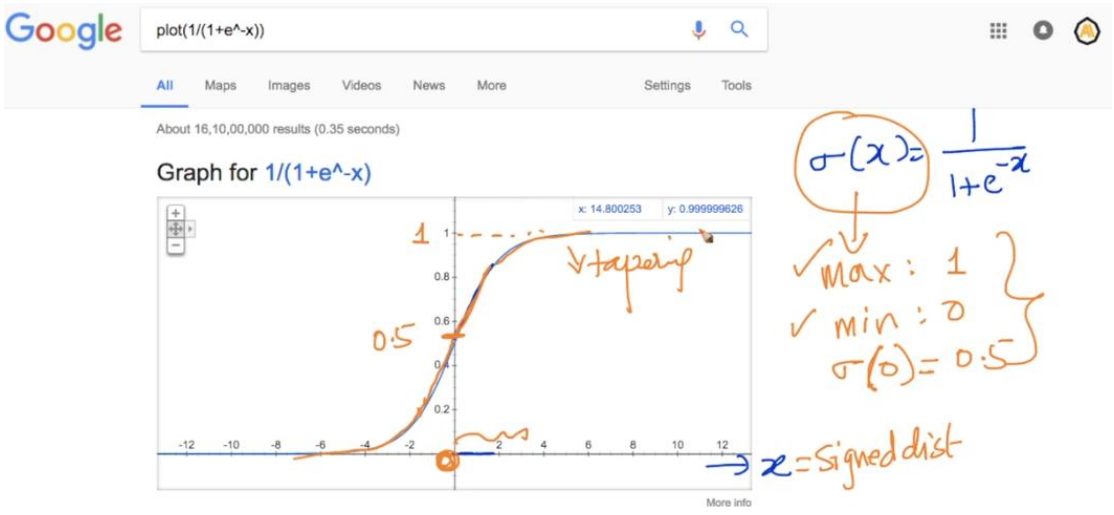
Practically pi 1 is best but we get wrong plane just because of 1 out l .

So we need any function which will handle this type of out l. fo

That function is called Sigmoid function .



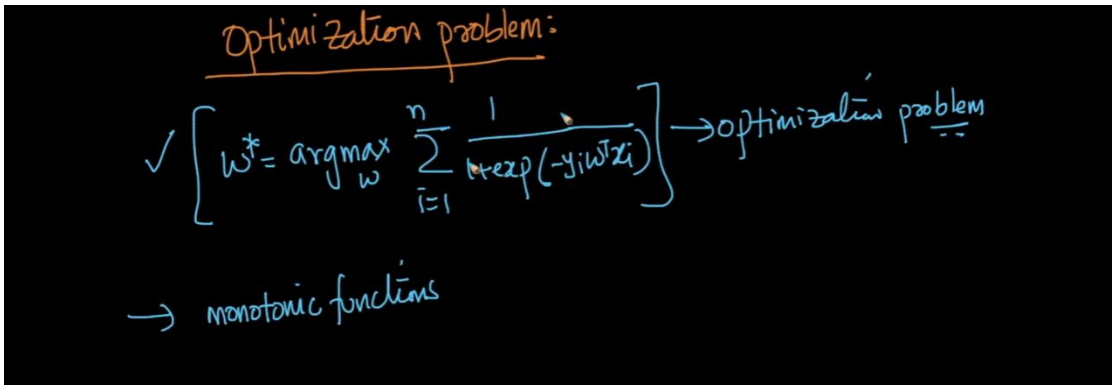
This function will squish out l. values from very large to small how ?



This function has great probabilistic interpret .

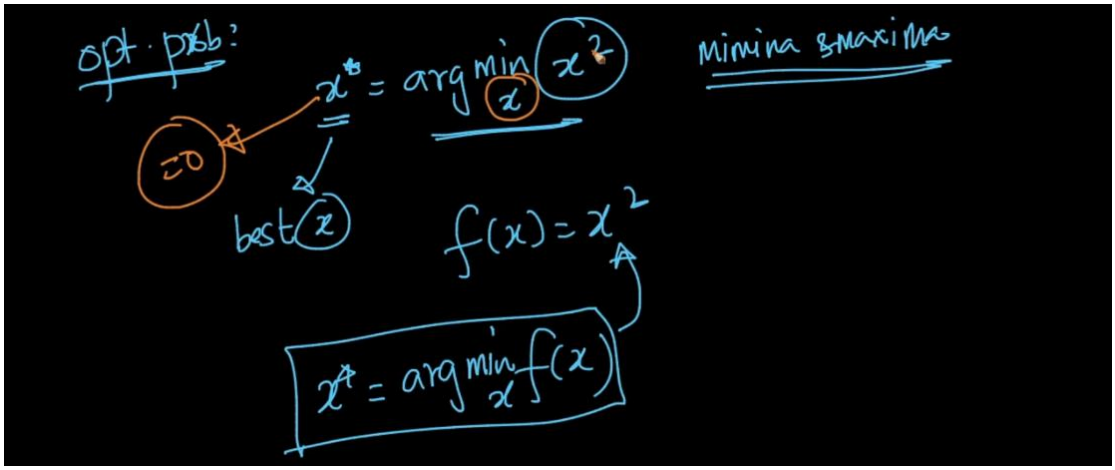
Lets say we are predicting y =1 for some x value

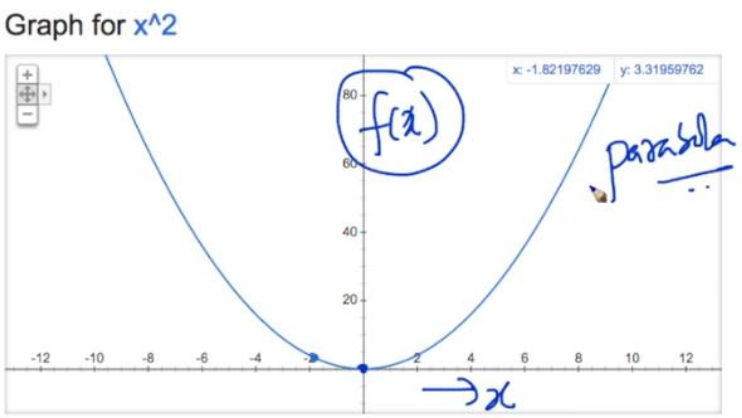
And lets say x is very large like out l . 12 in above example then this function return 99% prob value for belong to class 1 .



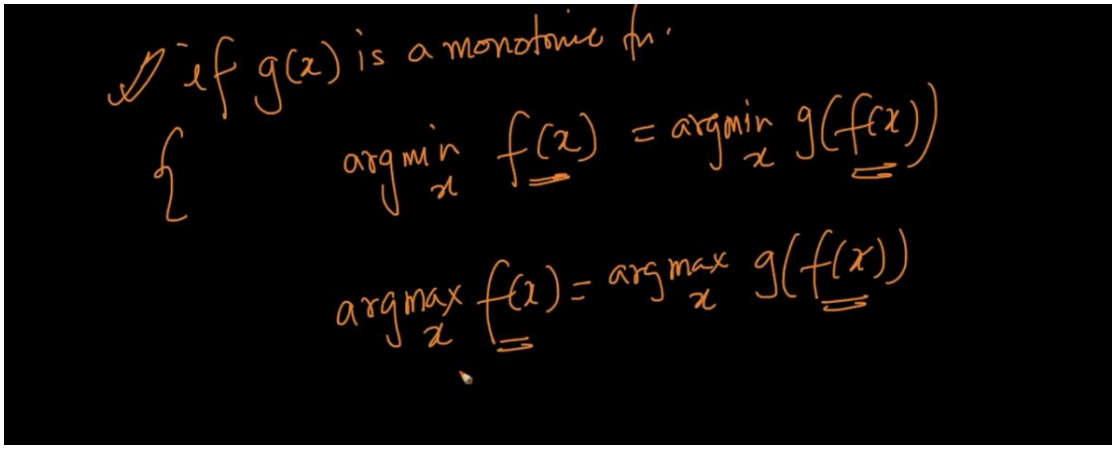
Log x is a monotonic function means as x inc log(x) also increase .

See below image that is opt problem it means find best x which minimize x2 .





See x = 0 which minimize f(x) .



geometry

$$w^* = \arg\min_w \sum_{i=1}^{n} \log\left(1 + \exp(-y_i w^T x_i)\right)$$

y_i: +1 or -1

prob. method

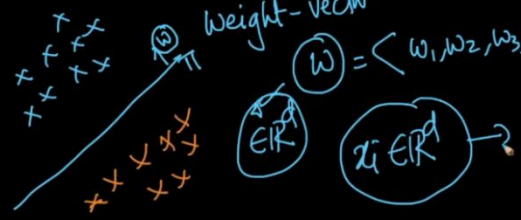
$$w^* = \arg\min_w \sum_{i=1}^{n} -y_i \log p_i - (1-y_i)\log(1-p_i)$$

y_i: +1 or 0

$$p_i = \sigma(w^T x_i)$$

Weight-Vector

$$w^* = \arg\min_w \sum_{i=1}^{n} \log\left(1+\exp(-y_i w^T x_i)\right)$$



$w = \langle w_1, w_2, w_3, w_4, \ldots w_d \rangle$

$w \in \mathbb{R}^d$

$x_i \in \mathbb{R}^d$

$w = \langle w_1, w_2, w_3 \ldots w_d \rangle$

$f_1, f_2, f_3, \ldots f_d$

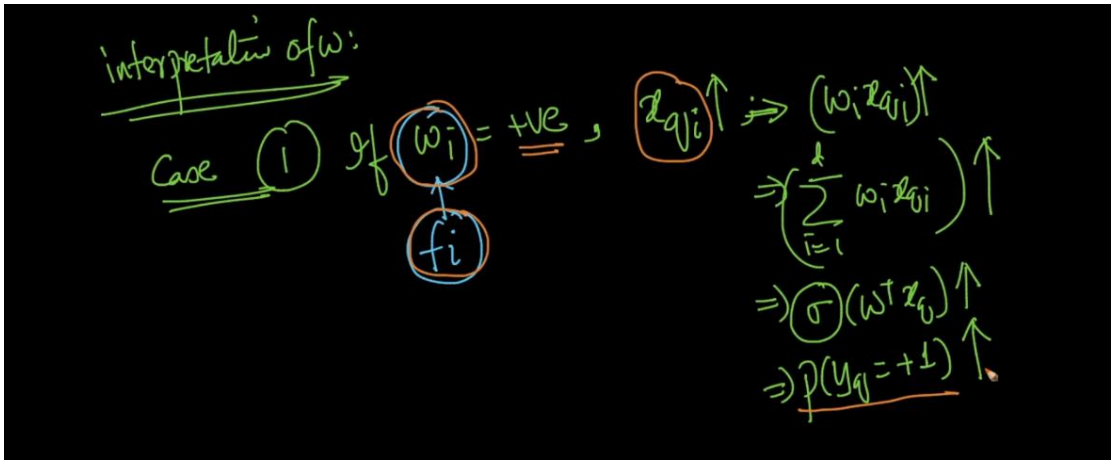
decision: $x_q \rightarrow y_q$

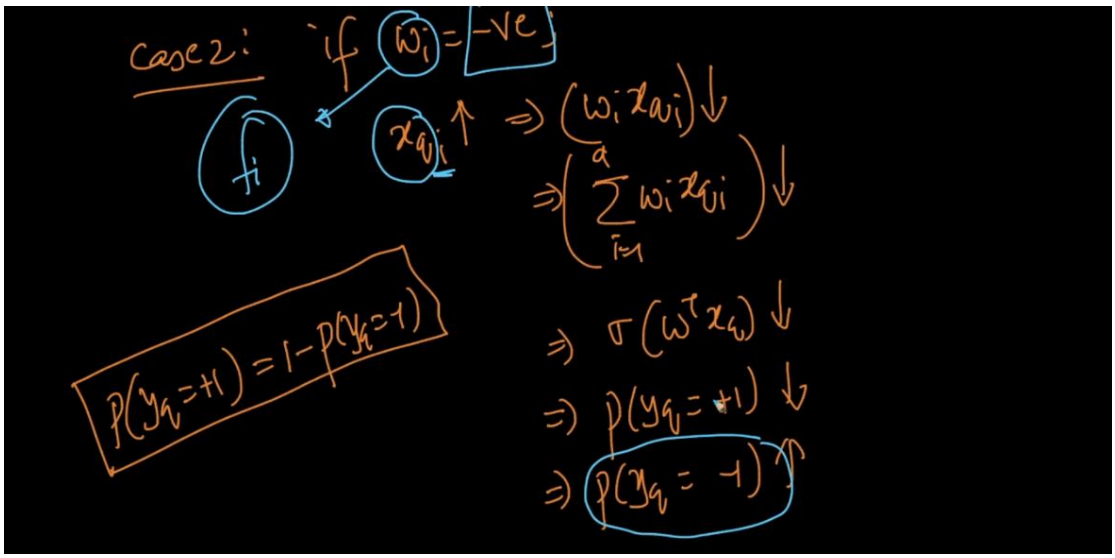
if $w^T x_q > 0$ then $y_q = +1$

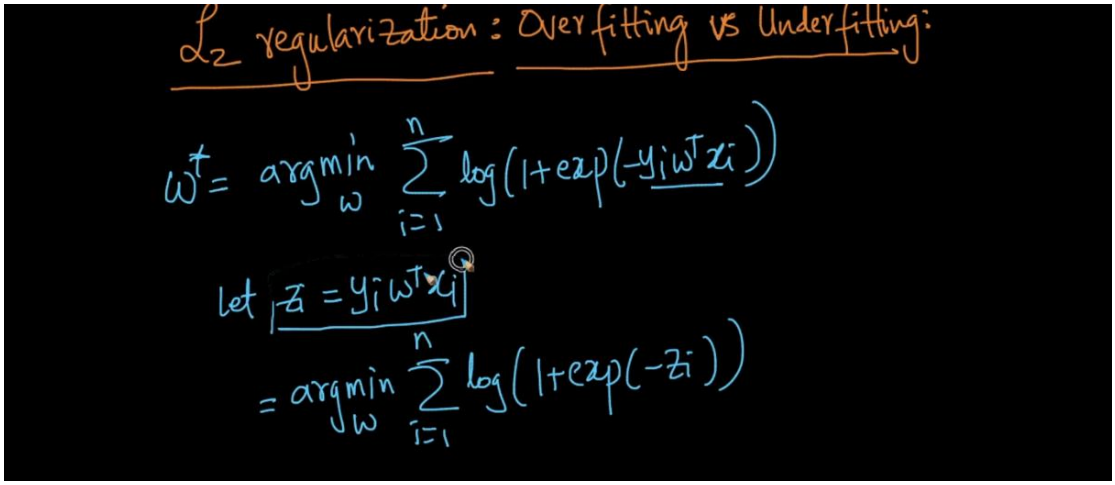
if $w^T x_q < 0$ then $y_q = -1$

Prob. interp: $\sigma(w^T x_q) = P(y_q = +1)$

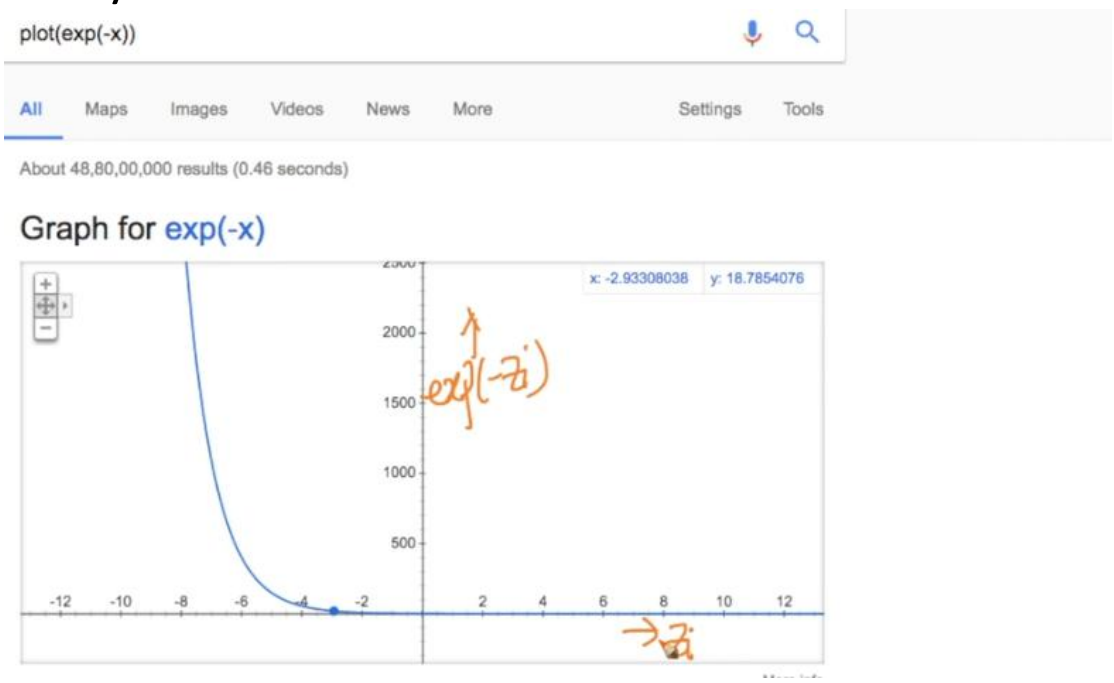
For each feature we have weight . we can find prob of new point will be + using sigmoid function .





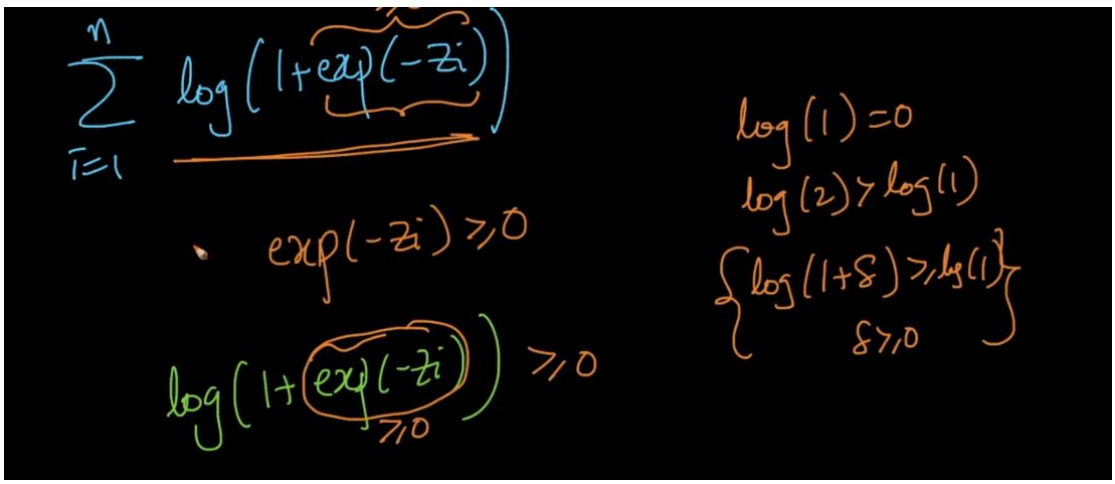


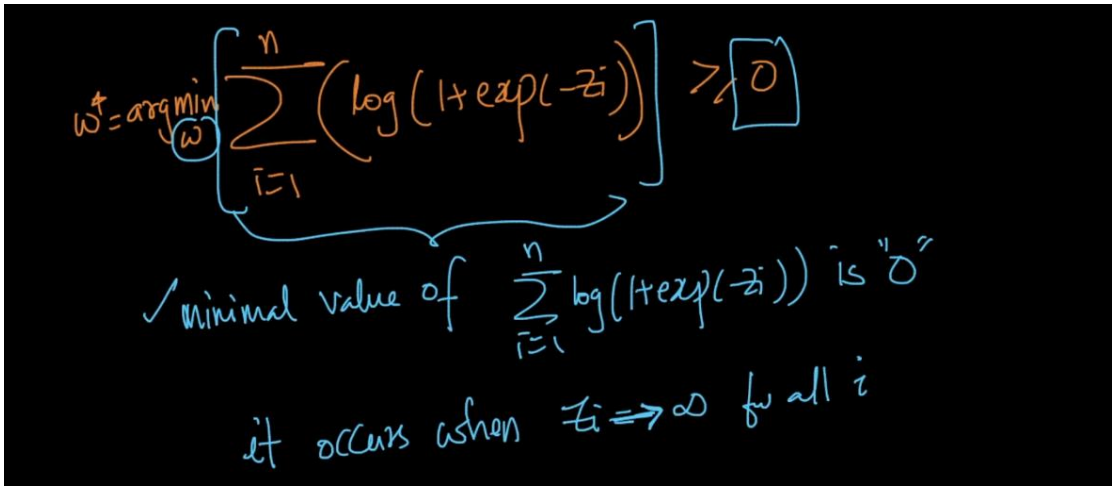
Lets take first exp part of equation .
Exp(-Z_I)
If we plot exp(-x) on Google we get this value will always >= 0 .



Now we know log is monotonic increasing function so log(2) > log(1) similarly ,

Log(1+e(-z)) is also always > =0 .

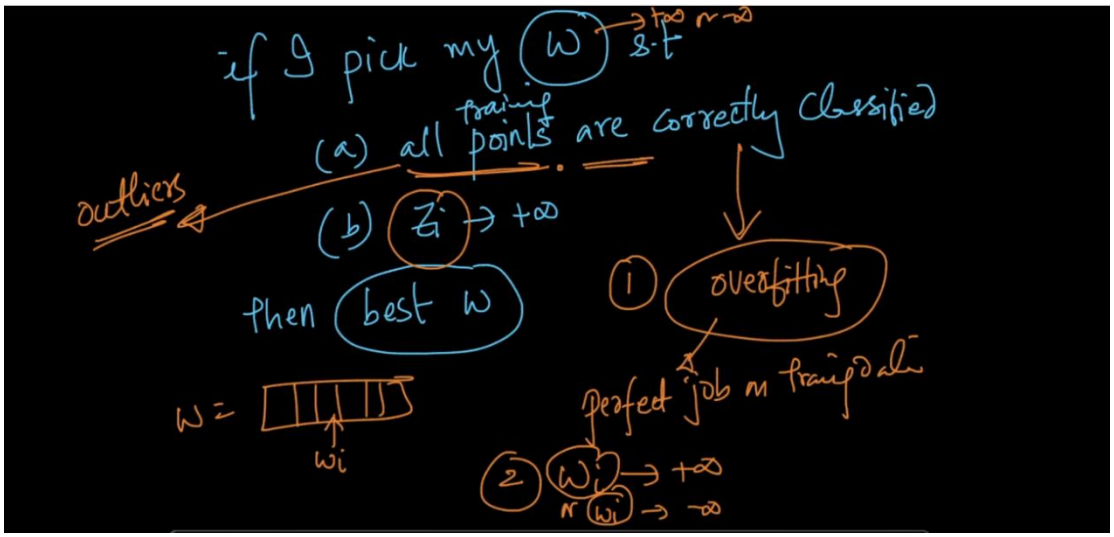




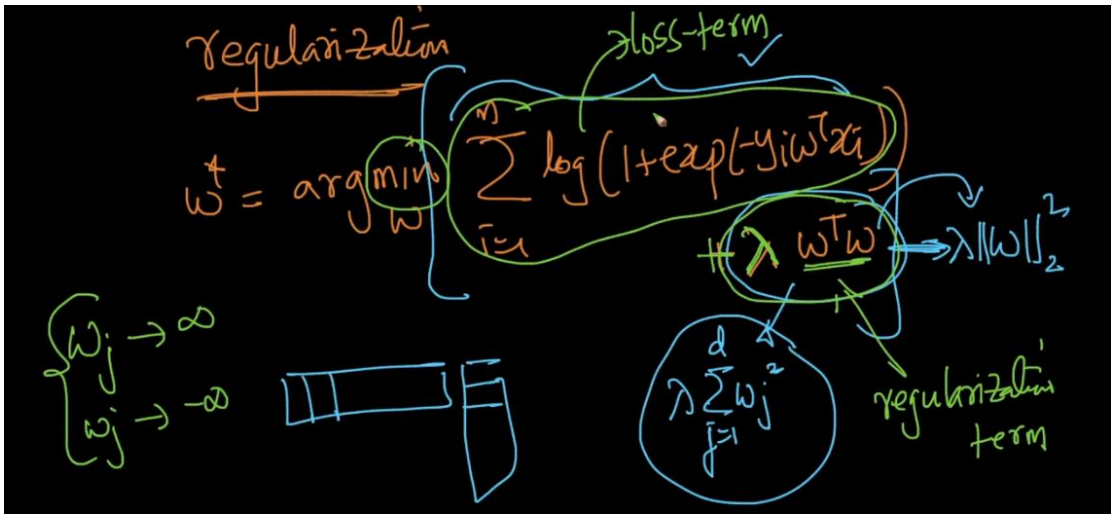
We want to find min value for w and min here is 0 .
We achieve these when sum of all z_I = inf .

So here x and y are fixed from train data we cant do anything with that only remaining is w so we modify w in such a way so we get sum of z = inf .

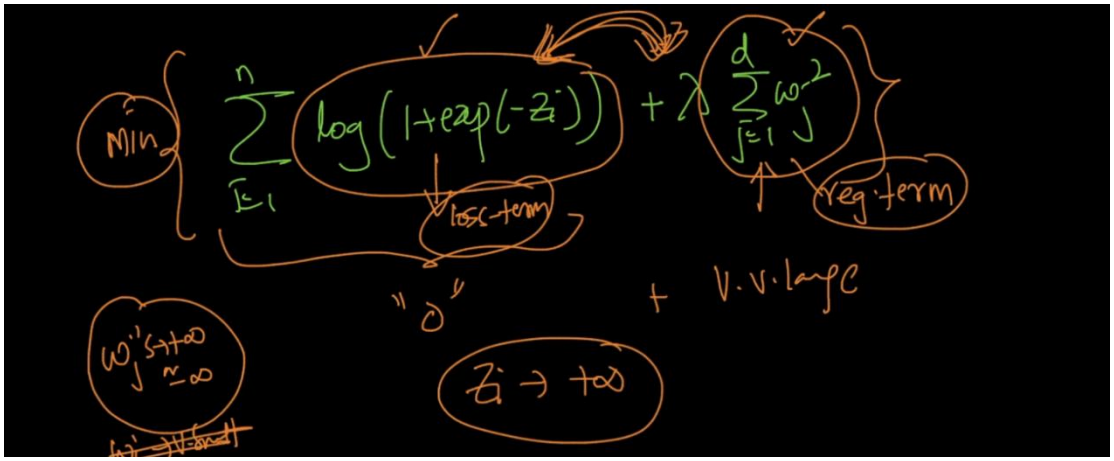
Lets assume we get perfect w but with that we get over fit issue also . because we are trying to fit perfectly out l. also so this will work good on train data but on test data it will fail this is overt fit issue .



For that purpose we use regularization which is done by adding :



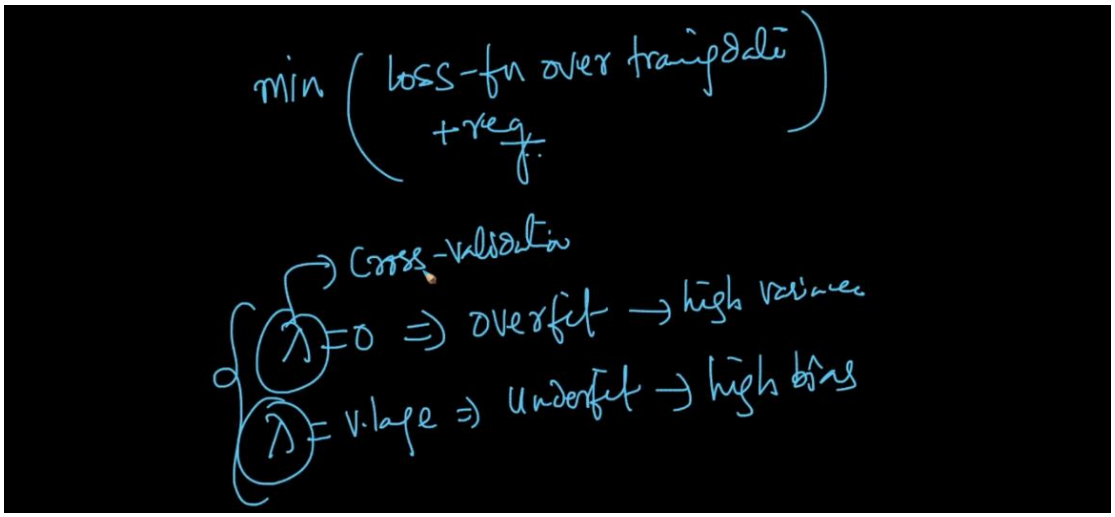
Because of that new term it will never allow to become + inf to w_I . it will always oppose .



So if we use

Lamda = 0 then we get over fit problem

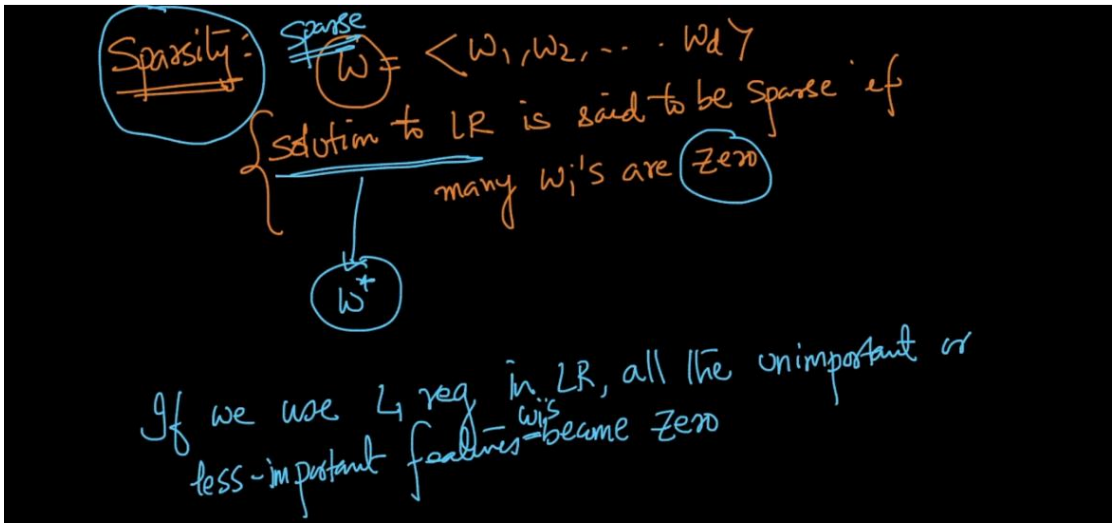
Lambda = very large then we will get our loss term very low and reg term very high so we will not use train data in that case and we get under fit problem ,.

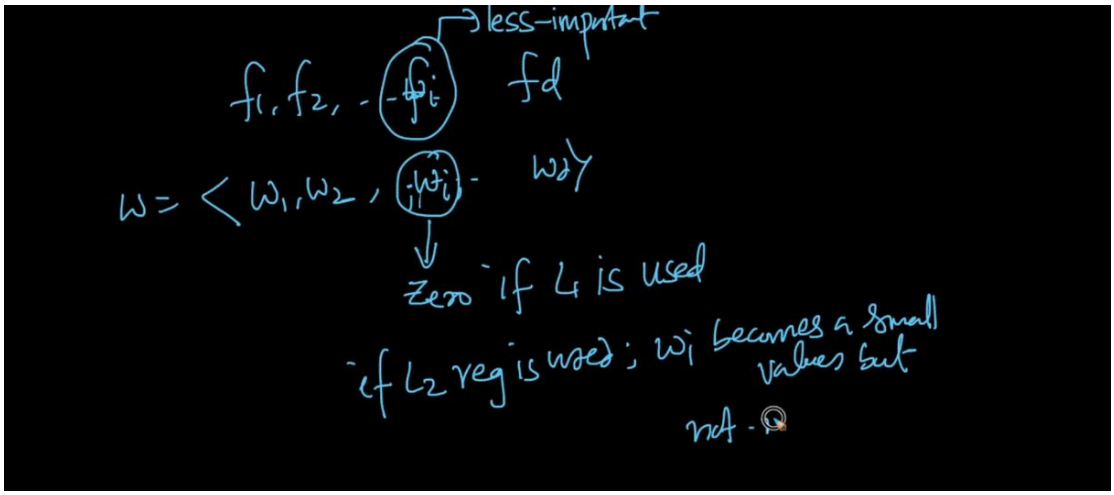


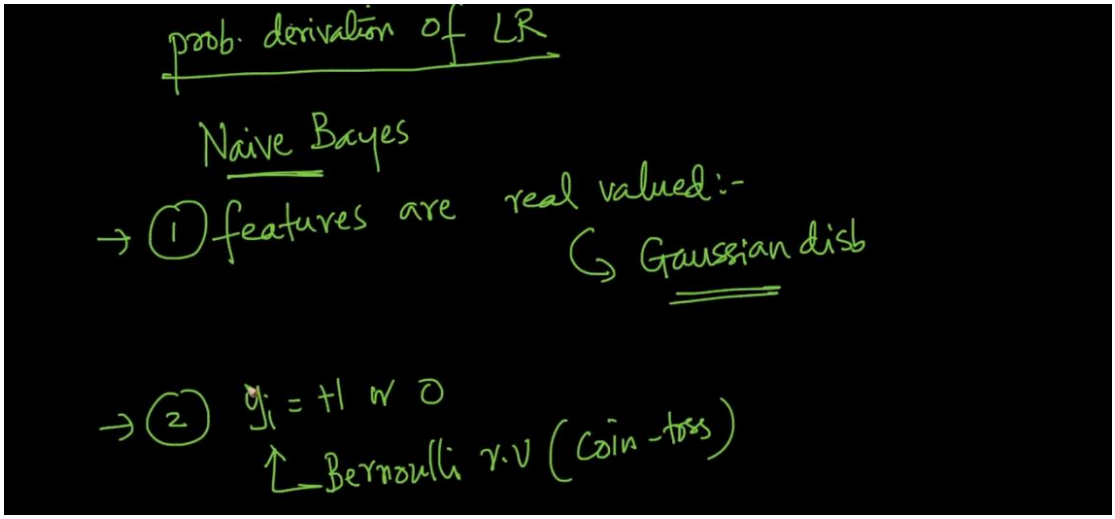
Another very popular alternative for L2 is L1 reg .

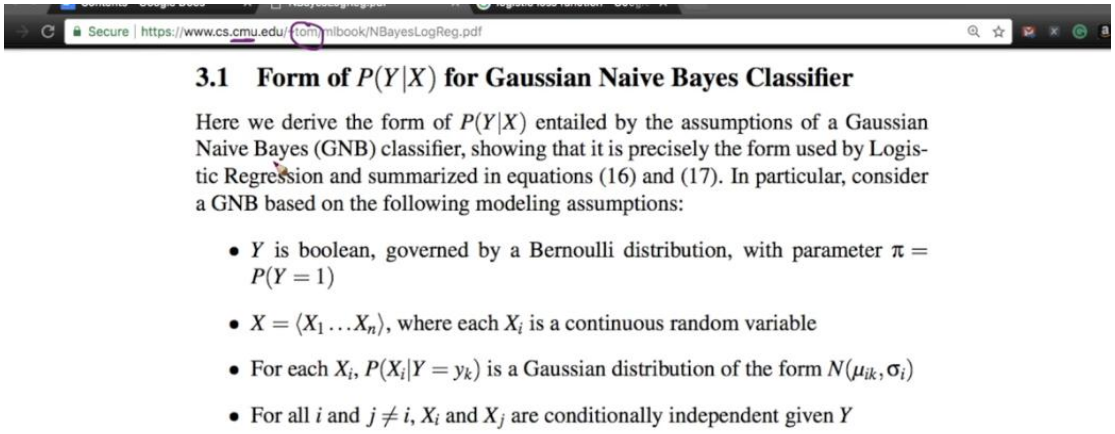
Because L1 have Sparse feature . We can say Regularization Sparse when it has maximum 0 values .

So if we use L1 then all unimportant features becomes zero .

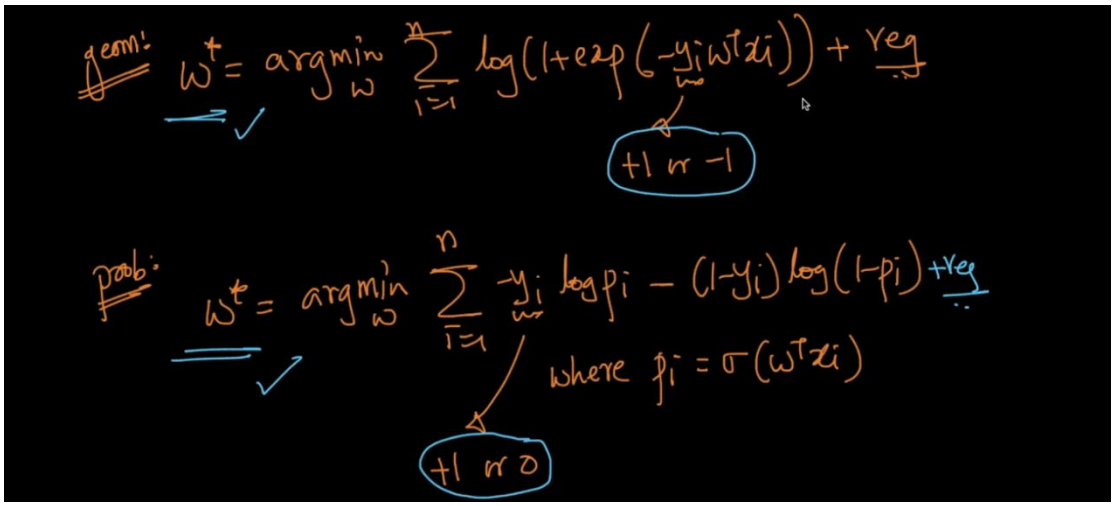








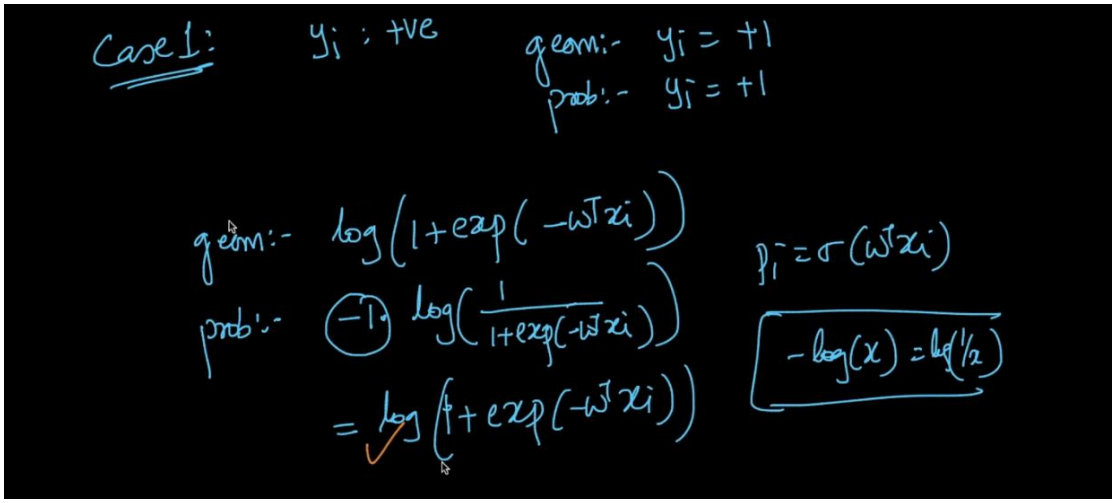
Below image both function work as same .



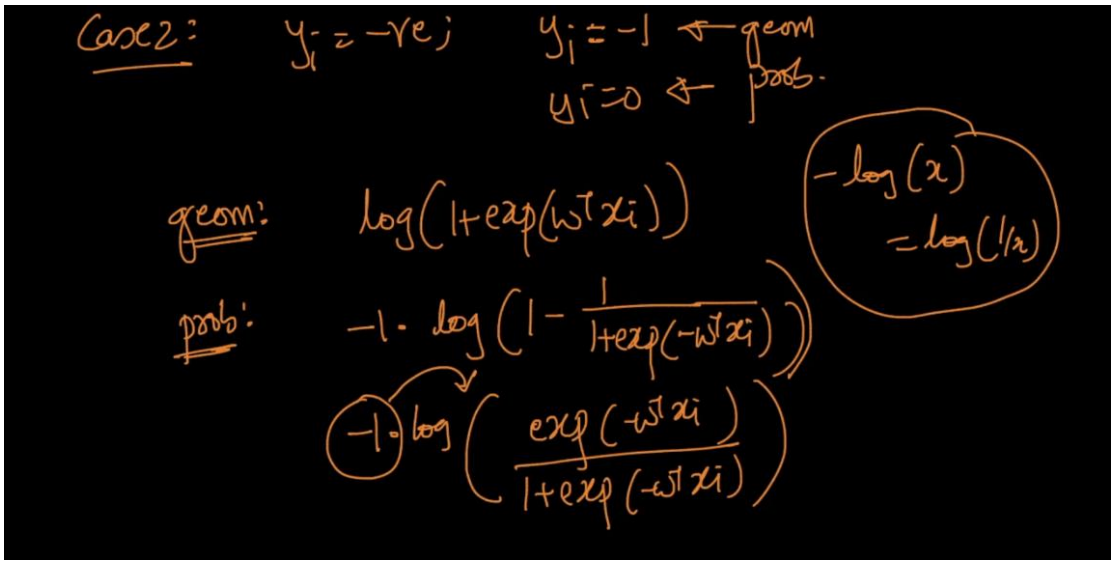
Proof :

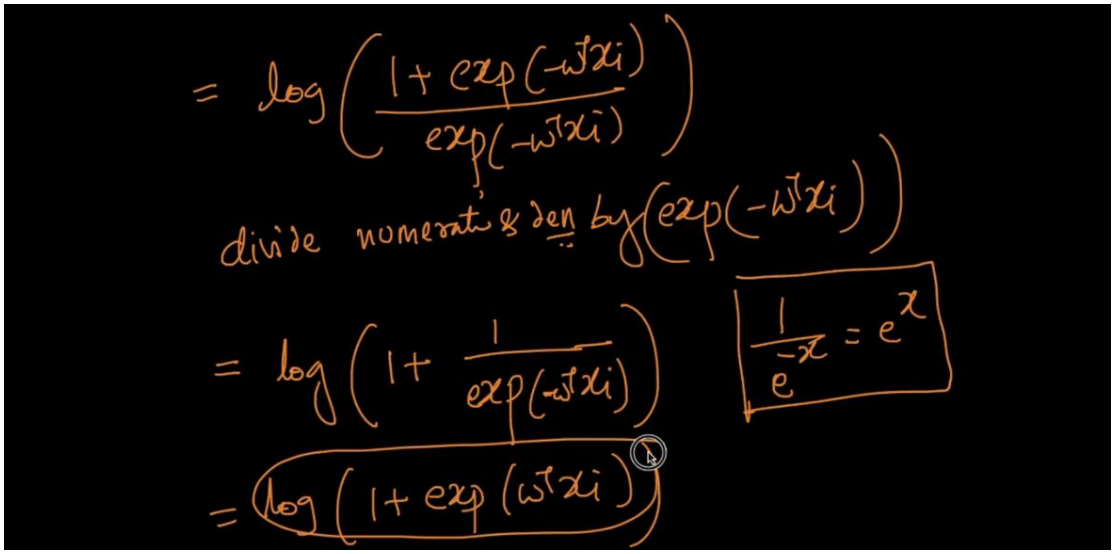
Lets take case 1 :

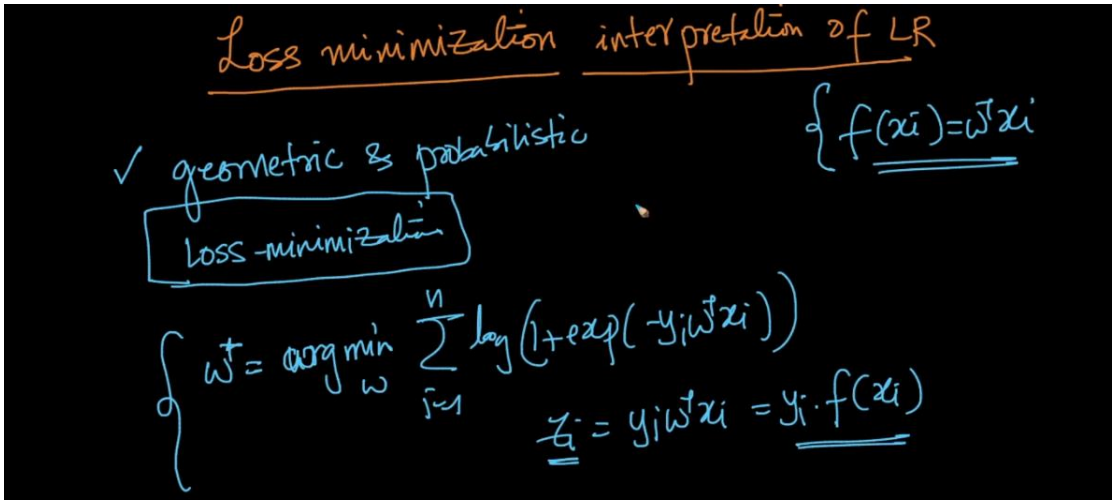
Y = +
Means y_I(Geo) = +1
Y_I(prob) = +1



After putting all values we get same result for both .







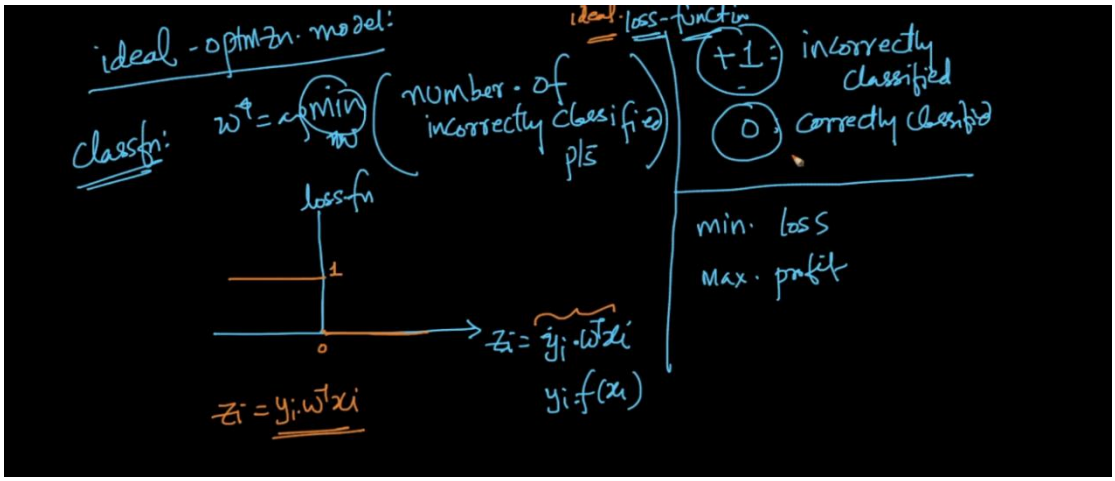
What is loss minimization ?

Means we want to find w so we get very less miss classification . so we want to minimize no of miss classified points that is called min loss .

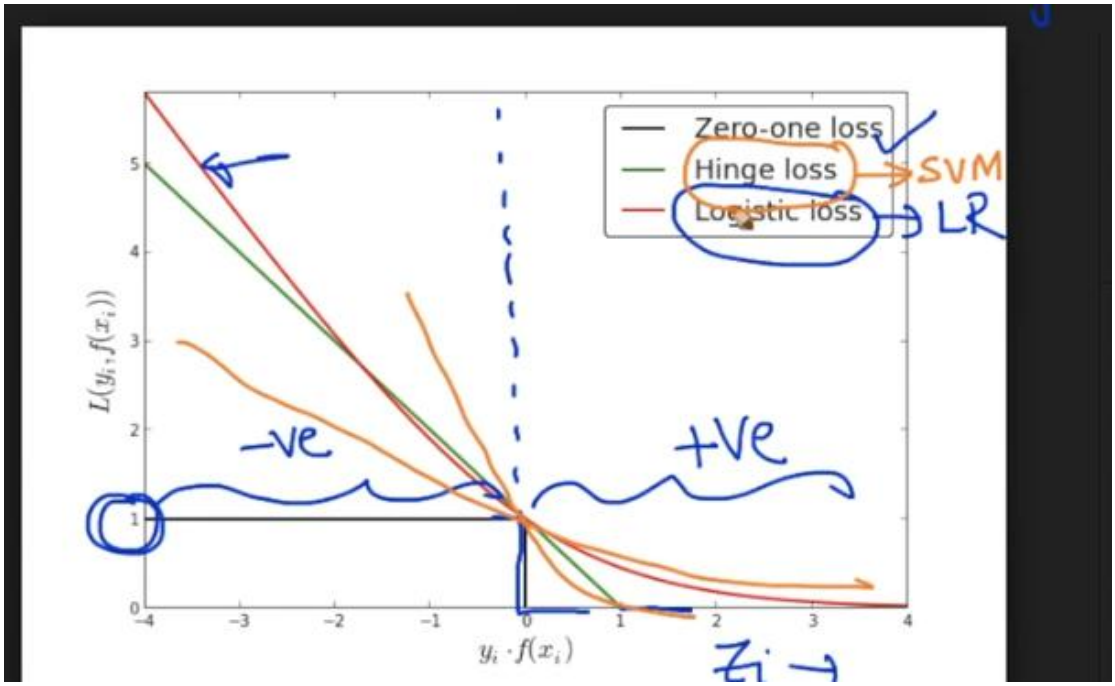
See below image 0-1 loss it is ideal loss function . Here lets say our z >0 means points are correctly classified means no loss hence z>0 you see loss also 0 .

If z < 0 means it is negative means points are miss classified . in this case we get loss = 1 .

So what value we get when z = 0 this is not defined in this loss .

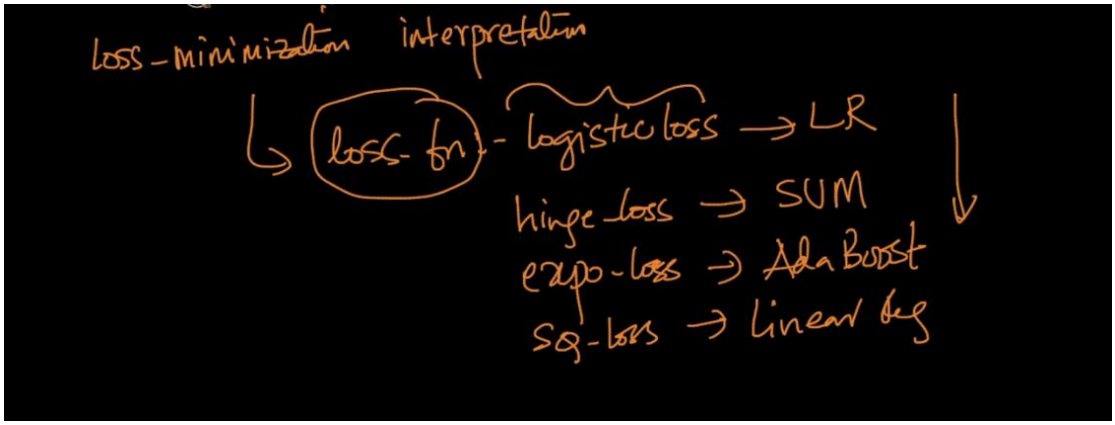


So we approximate this problem by using another loss ..



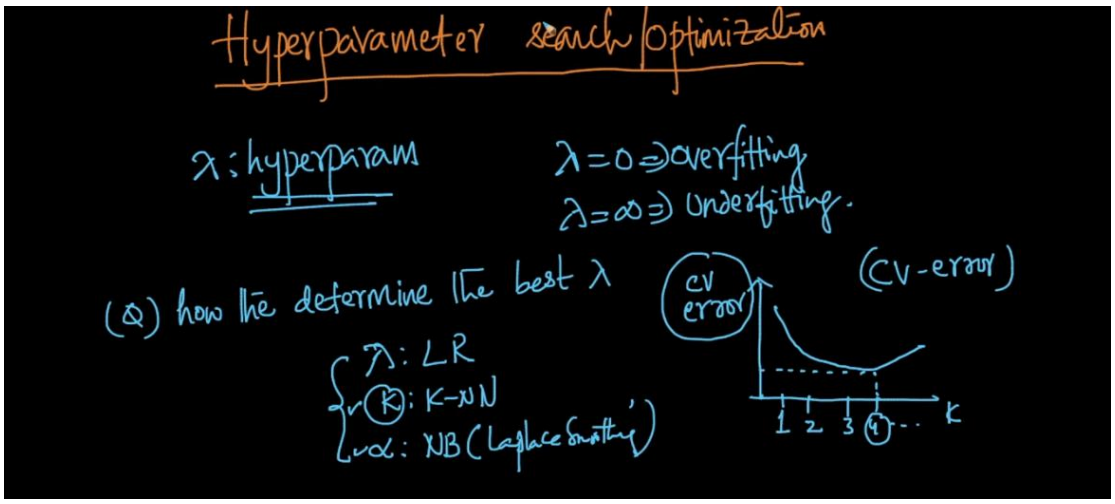
Called as logistic loss . if we use another loss hinge loss then we get SVM .

Just by changing loss we can learn diff model .

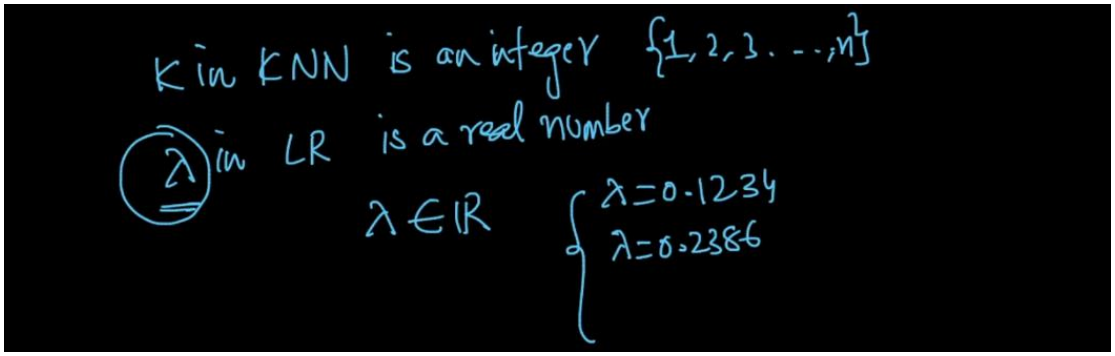


So how to chose correct lamda ?:

In KNN we use cross validation error graph and we pick that K value for which loss is very less .

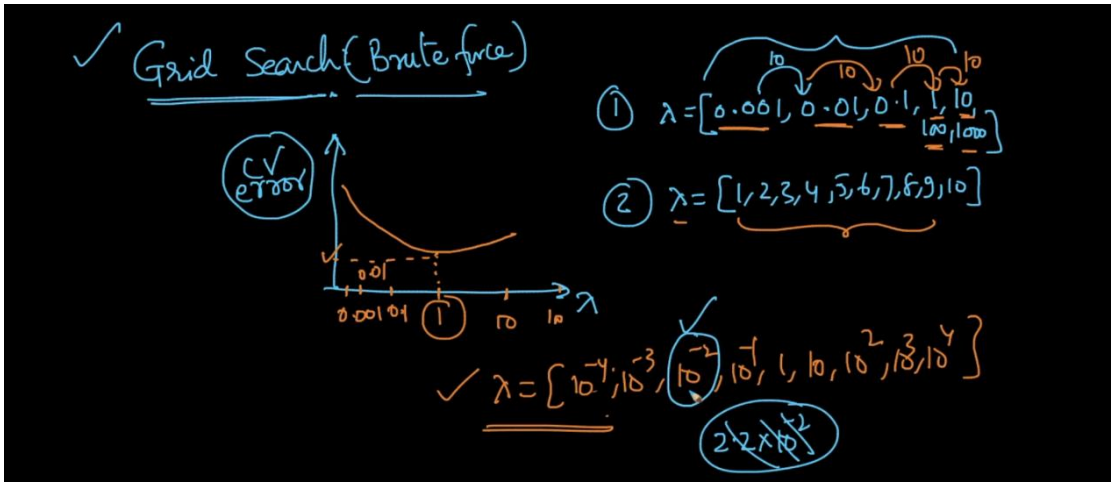


But for KNN values are like 1,2,3,….. n but for lamda values are real it can be anything ..
So we using 2 methods :



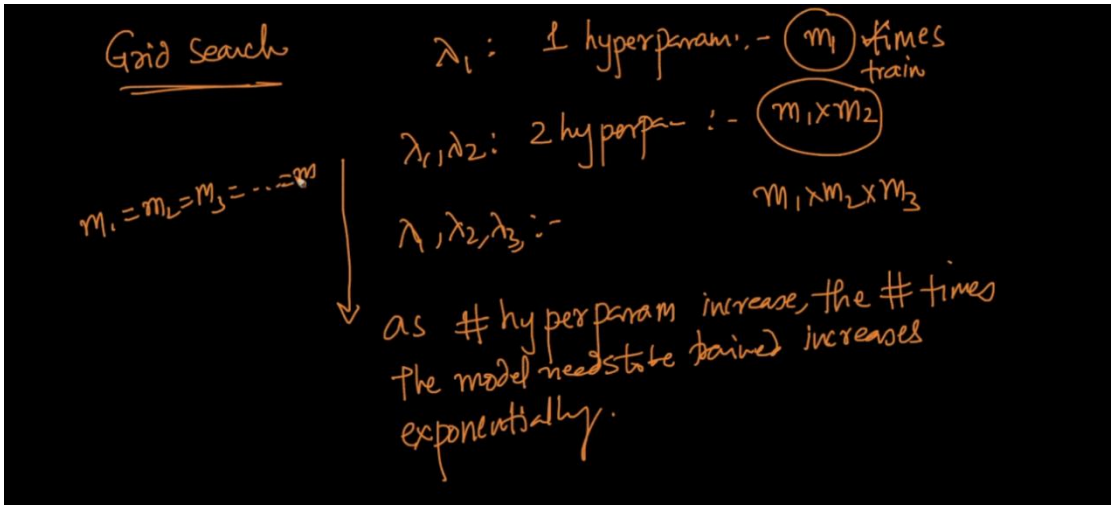
1. Grid search :

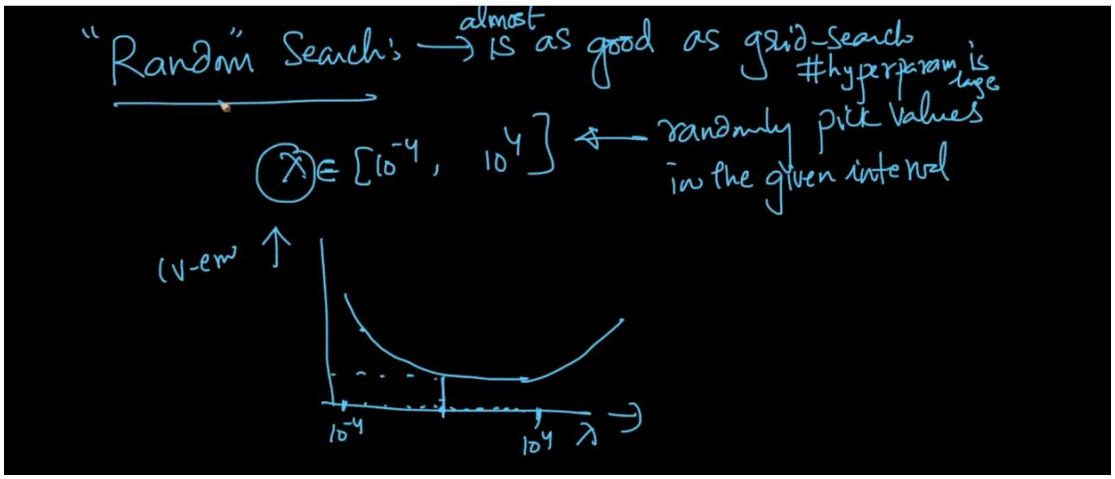
So here we give some range see in image and we run CV error graph for which error is less we pick that value as Lamda .



But here we have only one hyper parameter . lets say we took m1 possibilities any range so that much time we need to train our model to draw CV error graph .

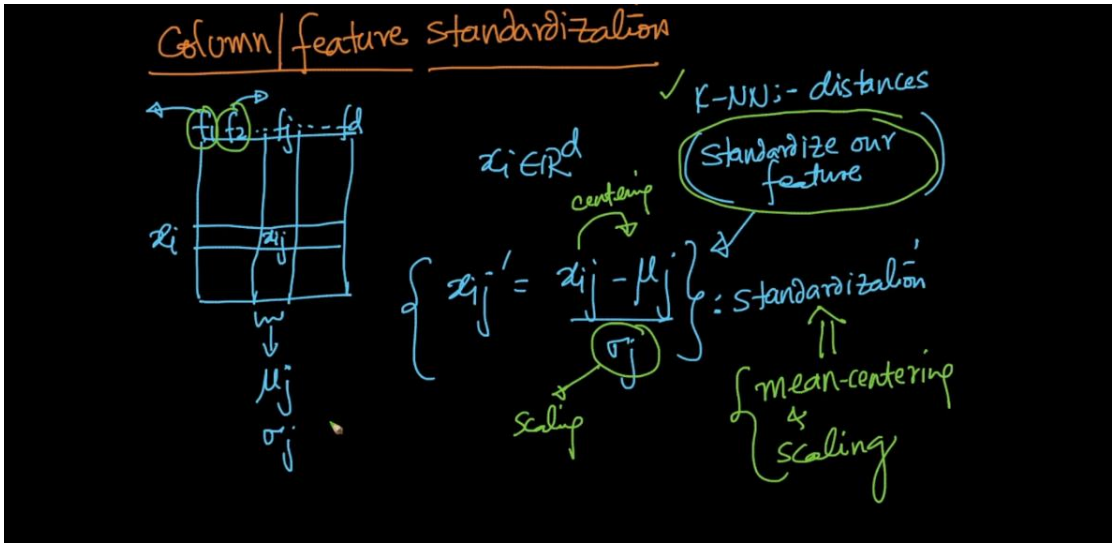
And if we have 2 parameter then we need m1*m2 times train this will increase time complexity very huge .

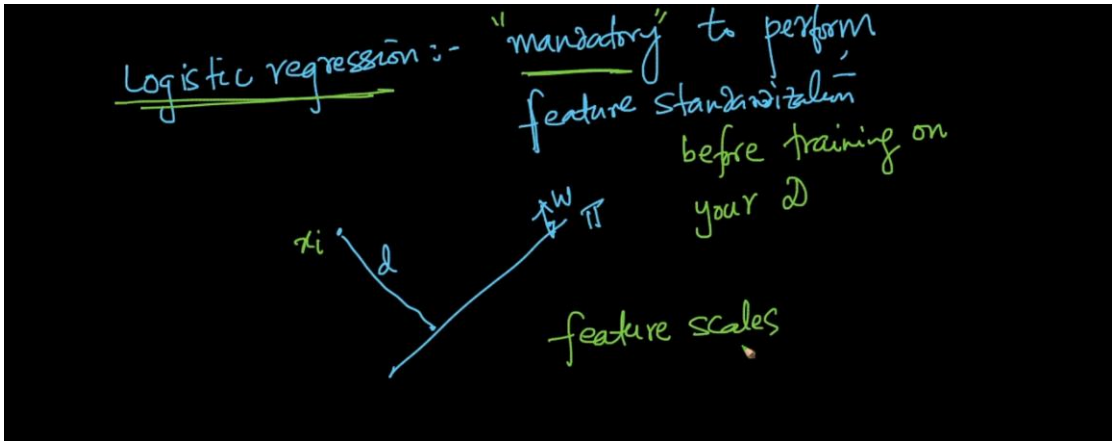




In this method we randomly pick m points from given range and will draw CV error and pick value for which error is less .

This will work too better than grid .

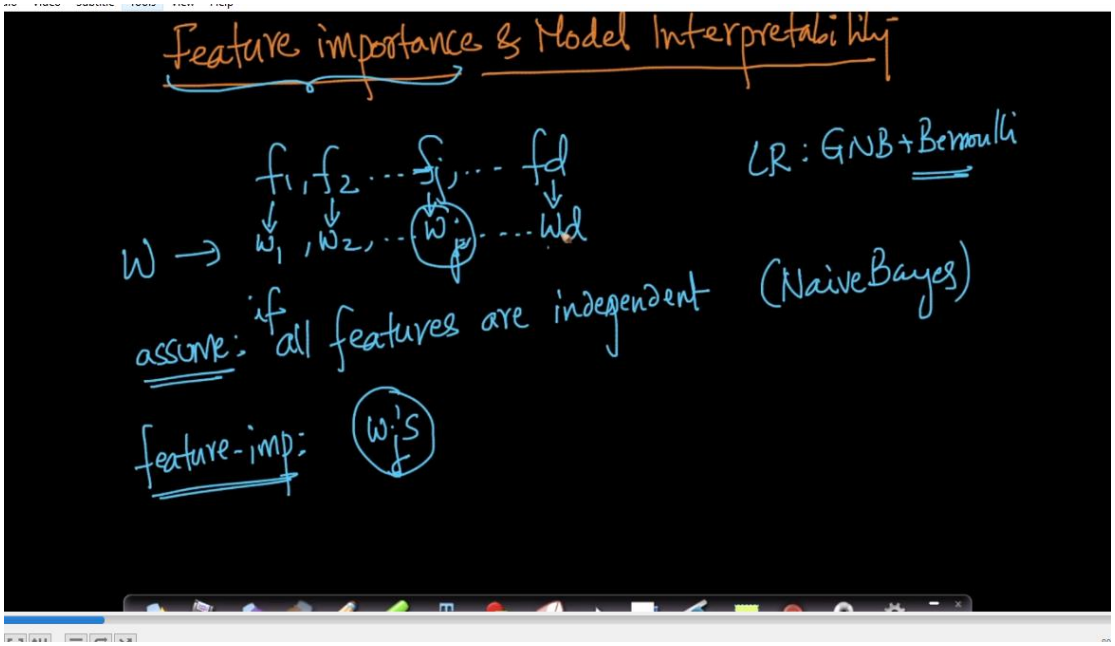


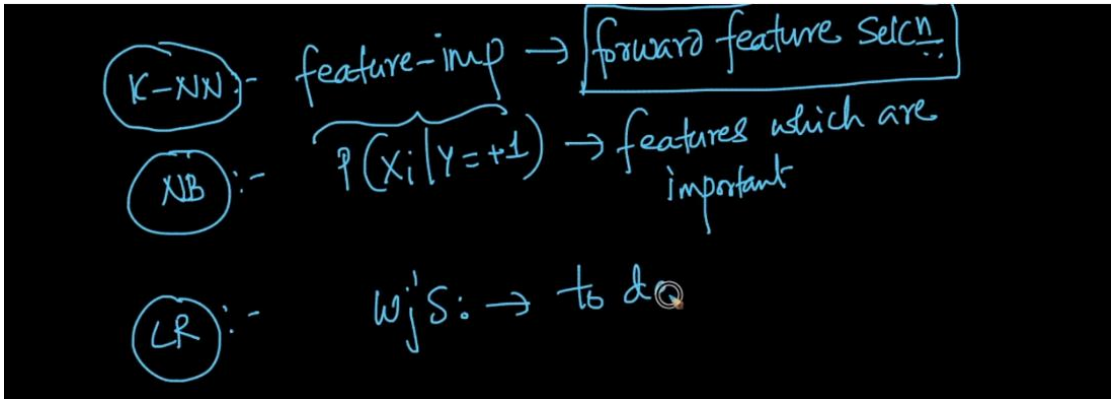


In logistic regression we are cal dist so it can be affected by scale of data like m ,cm , that's why before train it is mandatory to standardize out data .

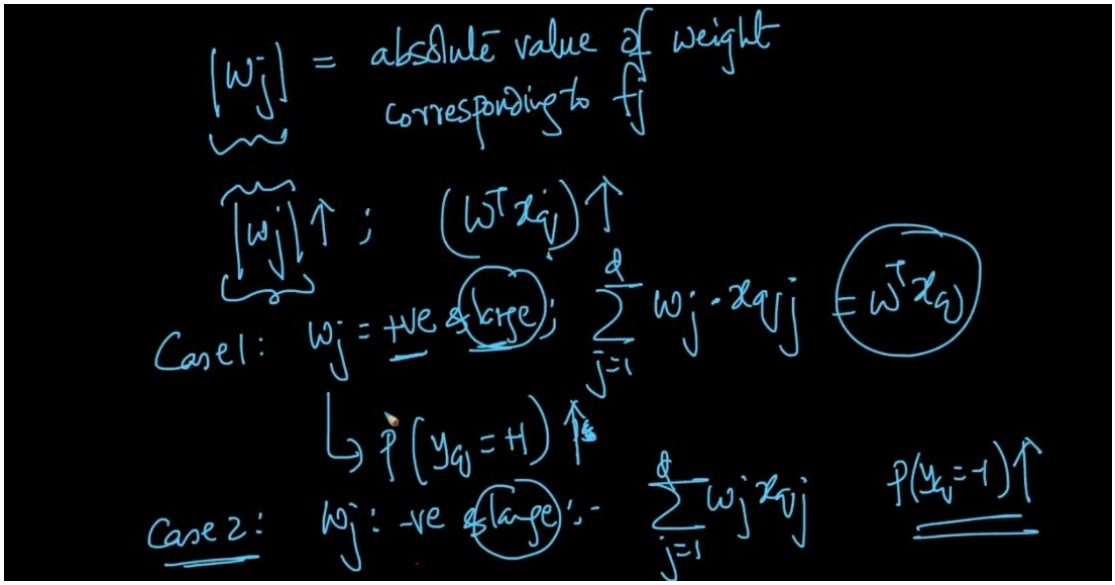
Feature Importance :

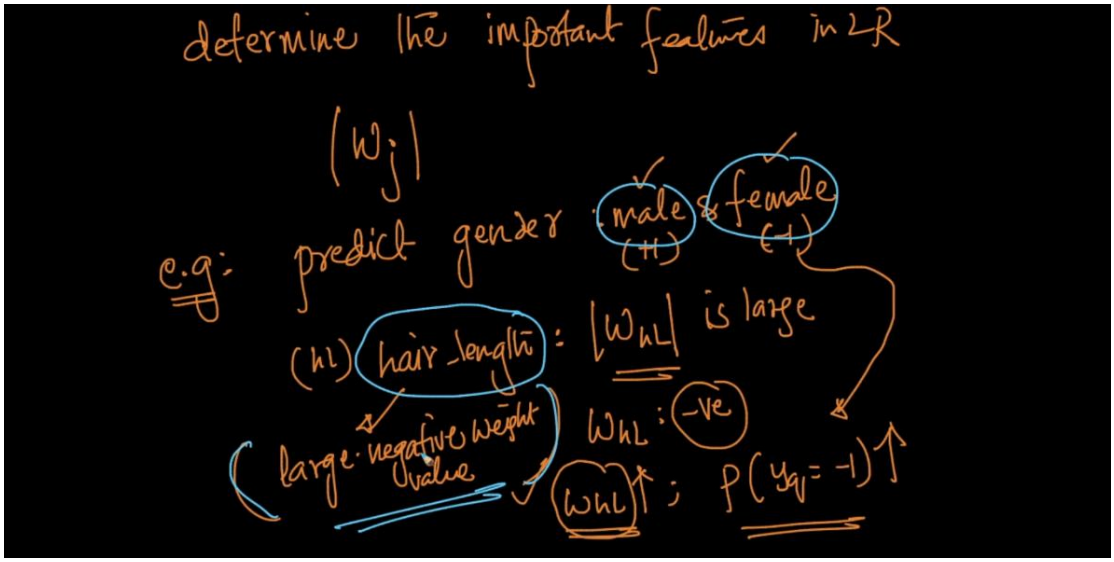
If all features are independent then only below tech will be useful .





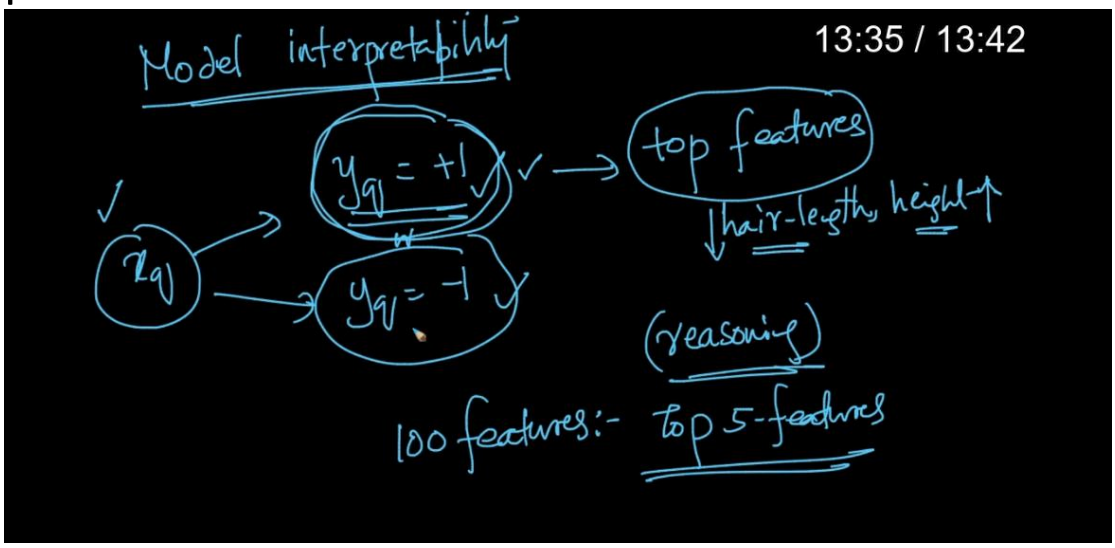
If w is high and positive then probability of that new x belongs to + is high . so just by seeing weight value we can say which feature is imp.

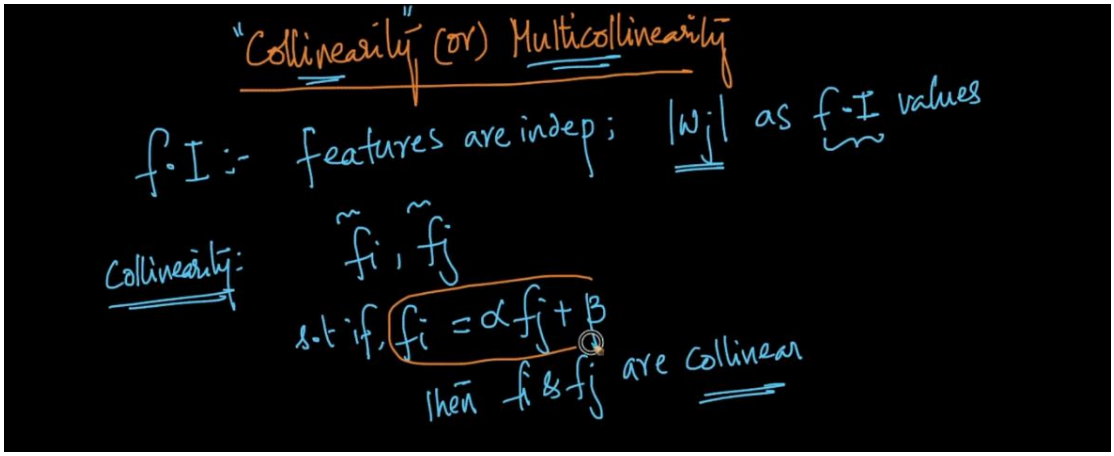




Lets take hair length as feature so W of this is very high (-) because female is (-) class here so as W is large we get large prob of new x is female .

In this way we can find best features from data . Interpret is also high because we can proof easily that x person has very high hair length and medium height and because of this my model predict x Is female ..

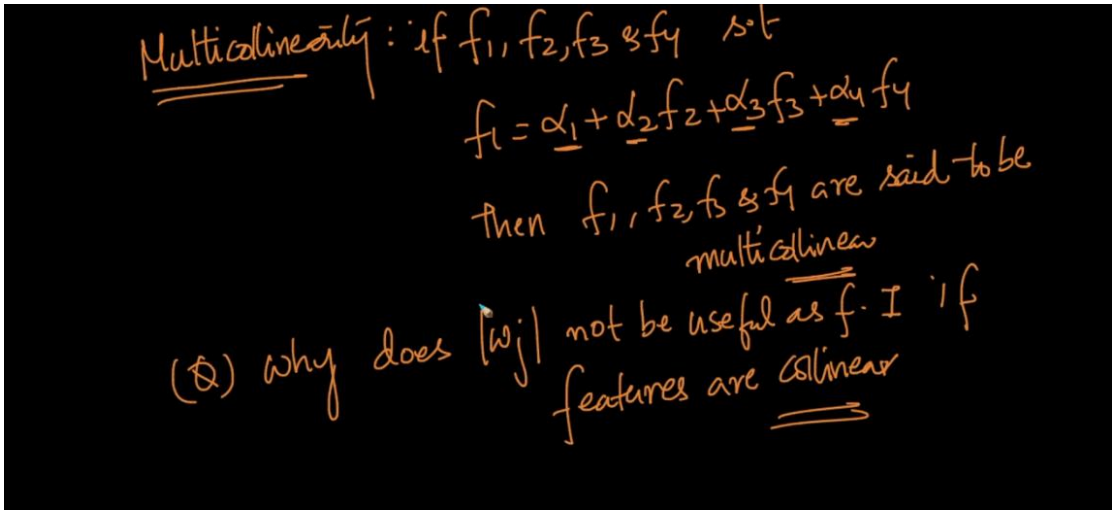


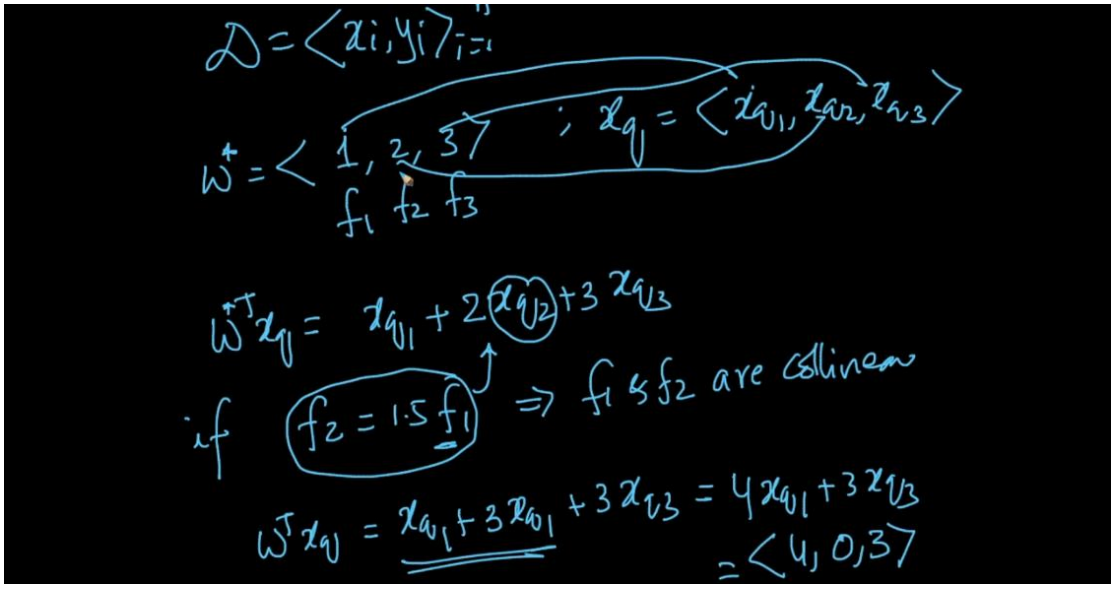


Lets say we have 2 feature f1 = height and f2 = weight .

If we get f1 (h) by doing some multiplication with f2 then we can f and f2 are co linear .

M co linear also same like if we get f1 by doing some multiplication with f2,f3,f4 ….. f_n then we can say f1 and others are m co linear ..





See lets say we have f1,f2,f3 we did multiplication .

Then we put f2 = 1.5f1 just assume then we get :
2xq2 = 2*1.5xq1 = 3xq1
Xq1+ 3xq1+3xq3

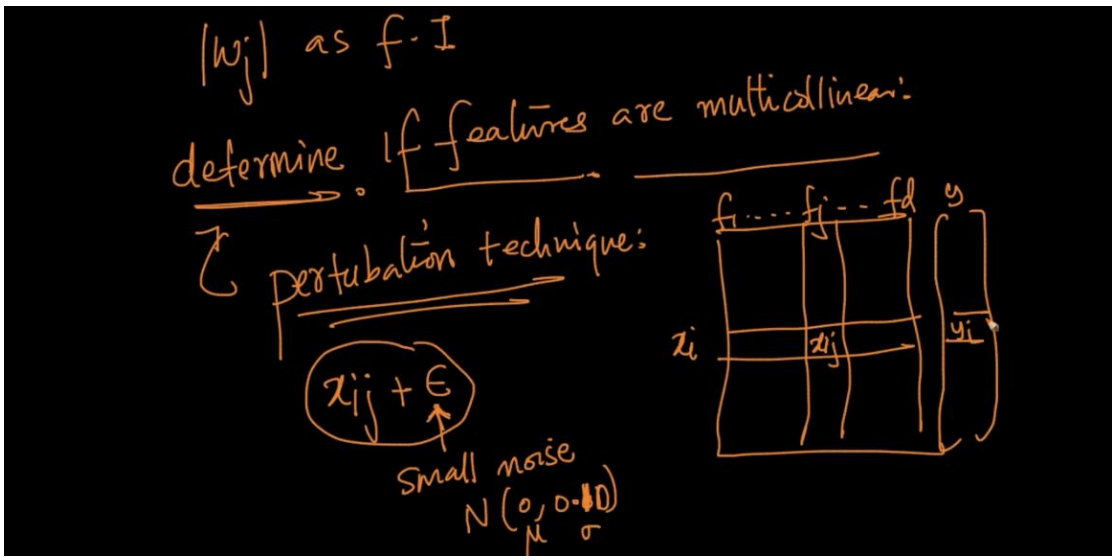
So we get matric 4,0,3 and original was 1,2,3

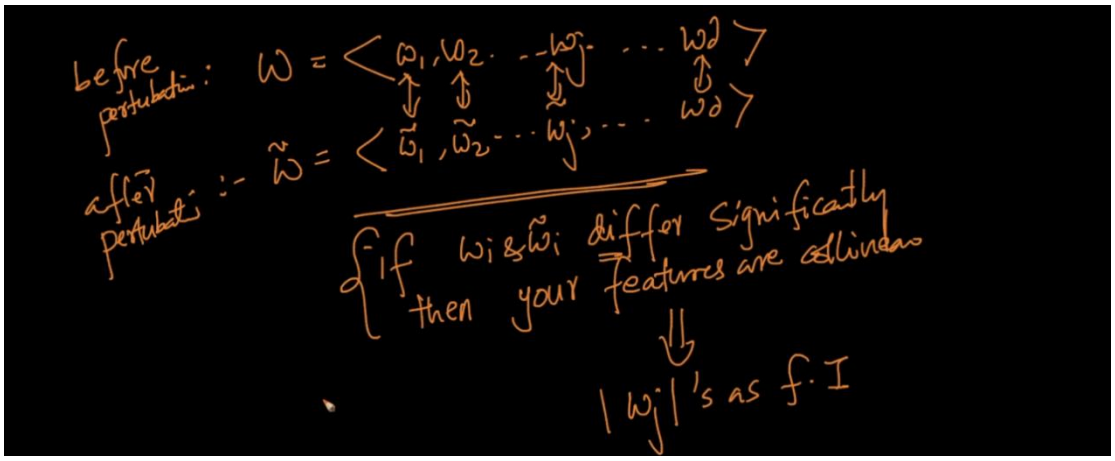
We get this difference just because of relation between f1 and f2 .

So from this our whole logic will change if we use this matrix to tell feature importance we will get very worst result .

See as per new matrix we say f2 = 0 means it very worst feature and original matrix say it 2nd importance feature .

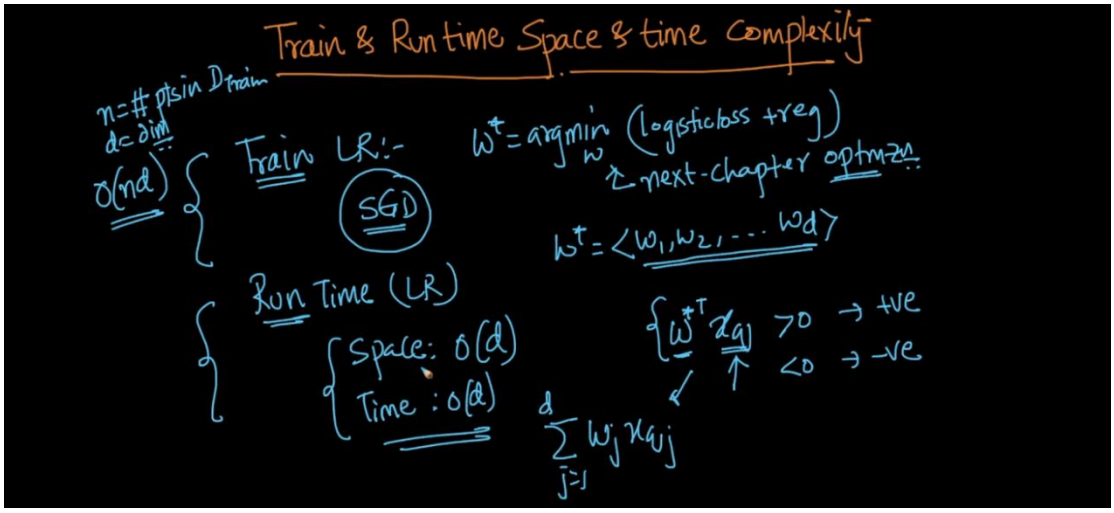
That’s why we cant use co linear feature to decide feature importance .





So given train data we train model once then we add some less noise in data and will train again then we compare both weights if difference between them is high then they are co linear don’t use then to find feature importance .

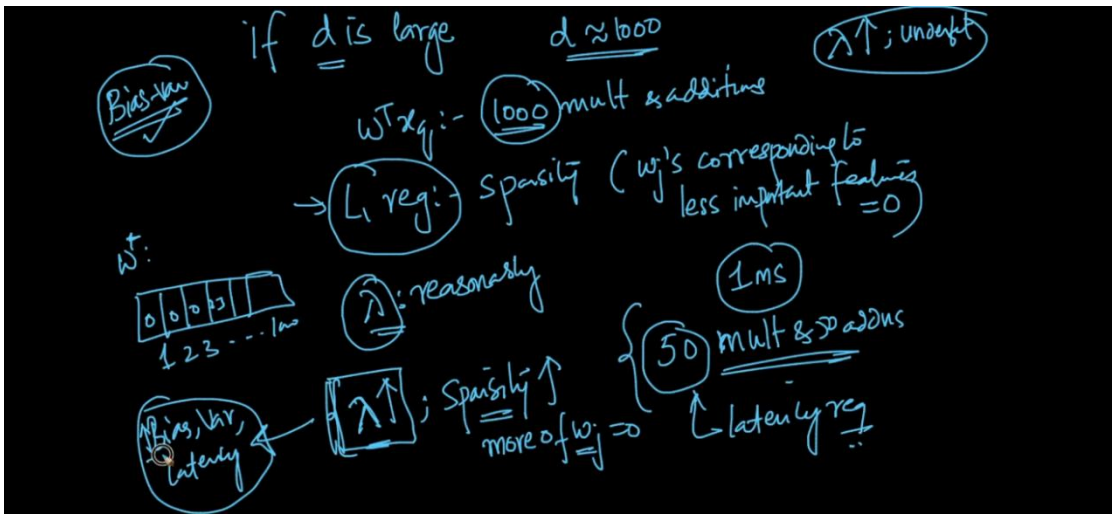
Small diff is OK .

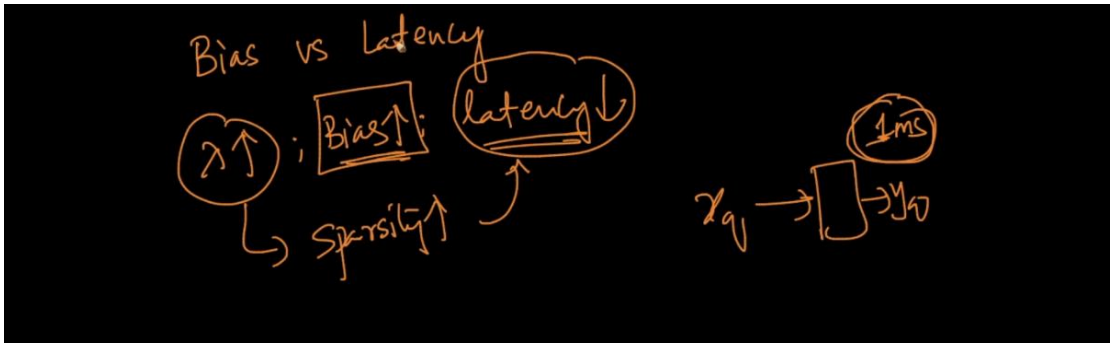


If d is small then time and space complexity is very low .

In L1 reg as we increase lamb sparcity also increase so we get lots of features with 0 value and remaining are only imp features .

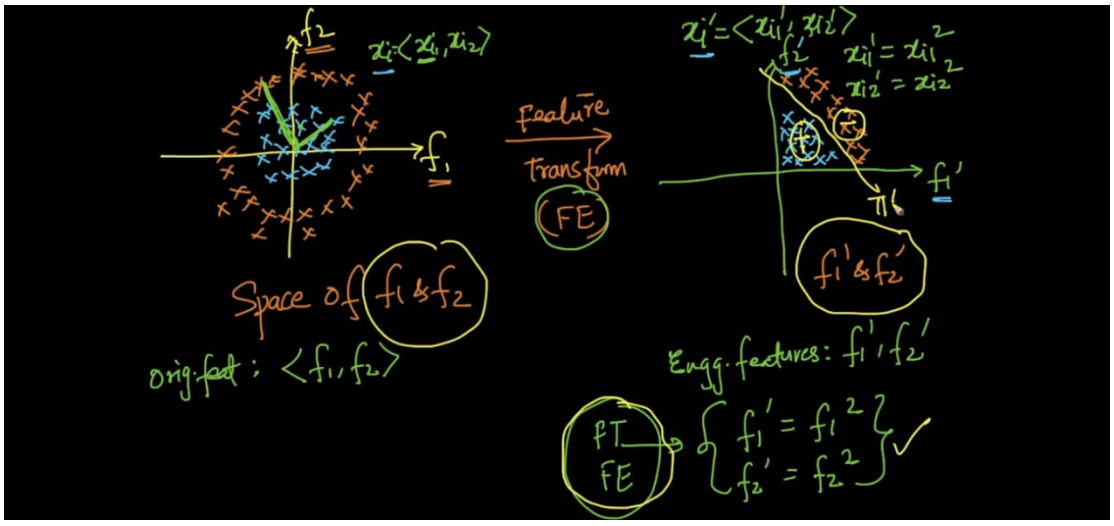
So if d is very large we can use L1 to reduce dimension :



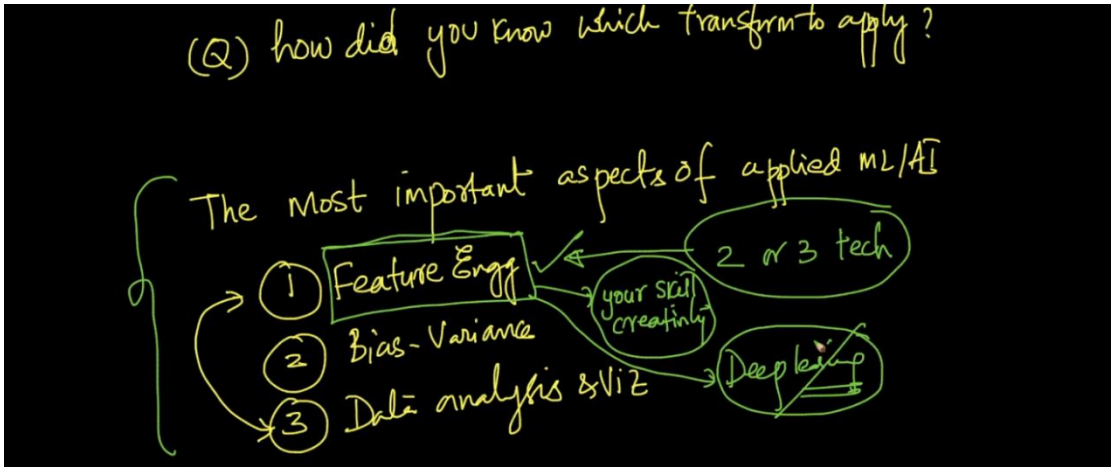


If manager say if your model take grater than 1 ms to run then I cant accept then it is better to go with high lamb so we get high bias with low latency . and it will be good instead of no model .

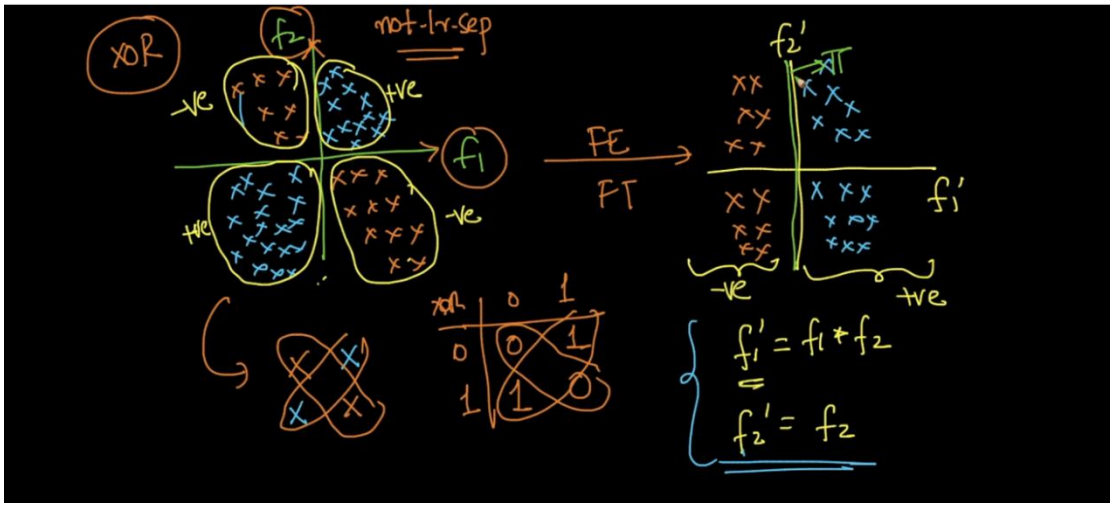
Feature Transformation :



See above data is non linear we apply feature engineering to transform non linear to linear ..

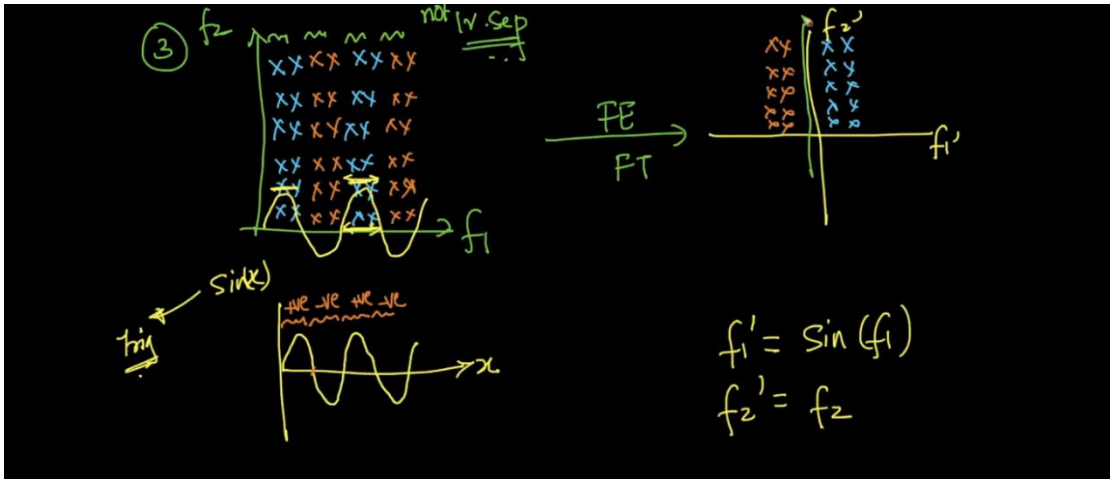


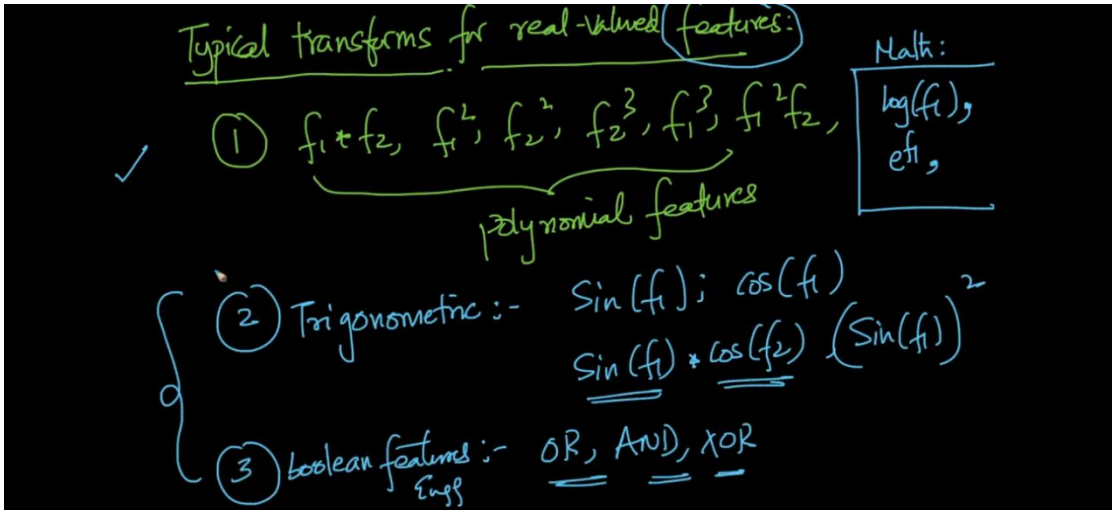
See below 2nd example of non linear to linear .



Here data is XOR we make f1 = f1*f2 so we get all positive points on one side on plane .

See below 3rd example here we apply sin because see data is in sin shape . so we get all positive and negative separate .

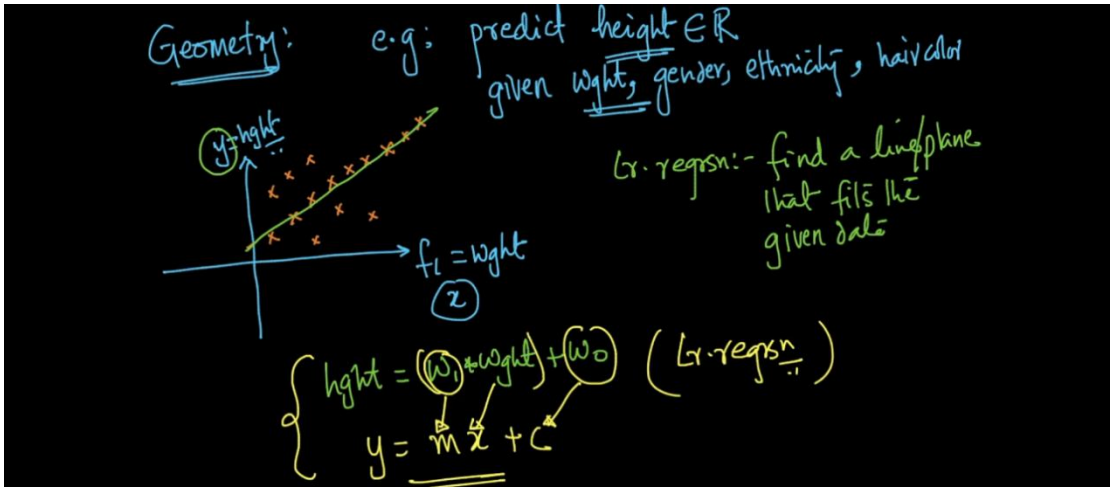




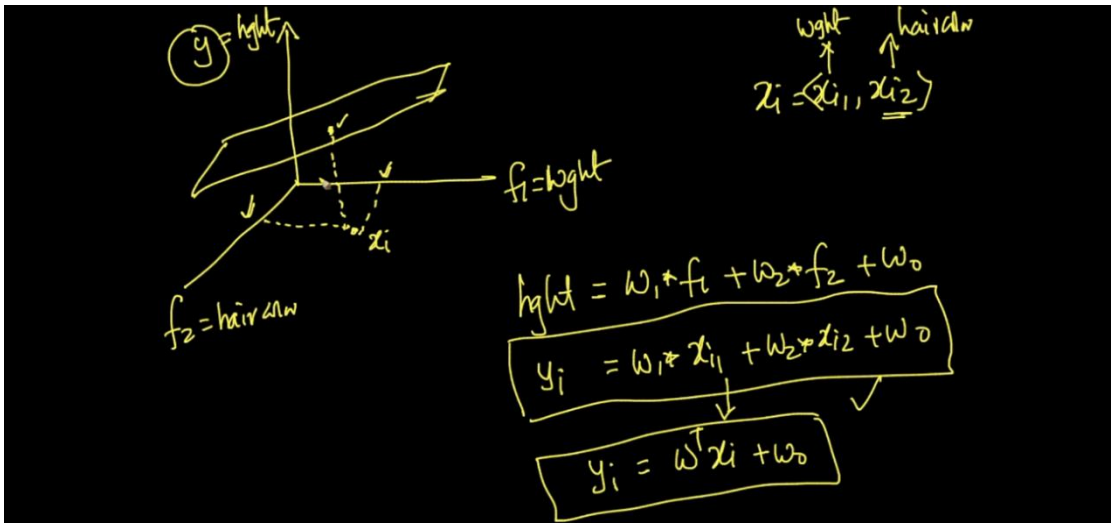
We can lots of diff technique for feature transform above are same famous methods.

Linear Regression :

In LR we try to find perfect fit line to a data .



So we want to find best w1 and w0 for 2d data .

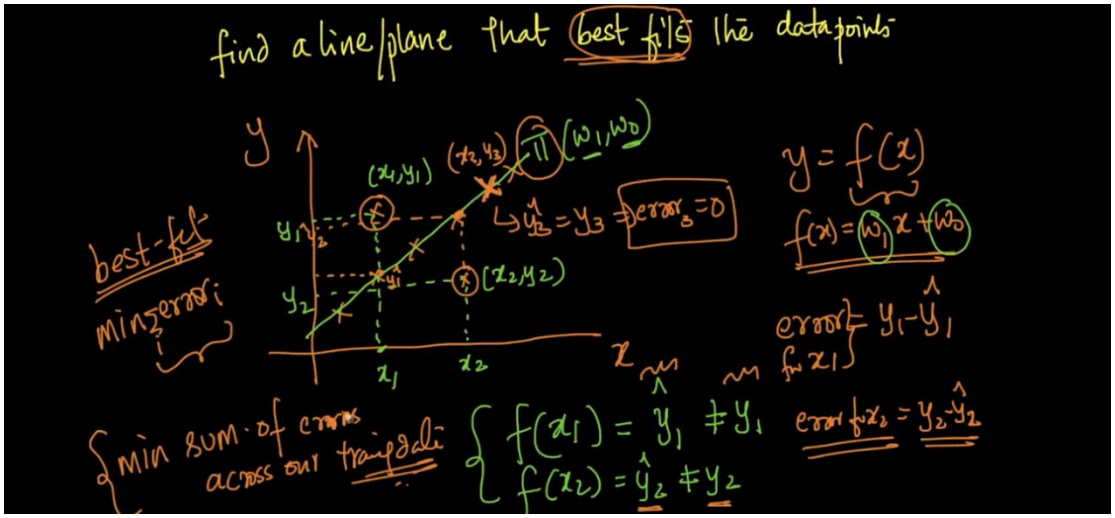


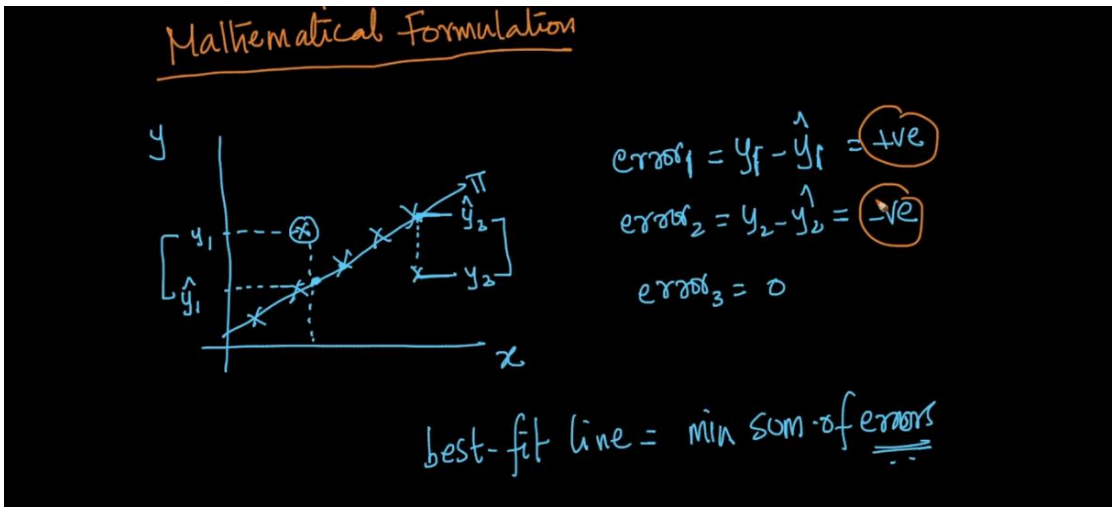
For 3 dimension we just extend formula .

Lets say we got fit line after that if we see below image some points are away from line so that is error .

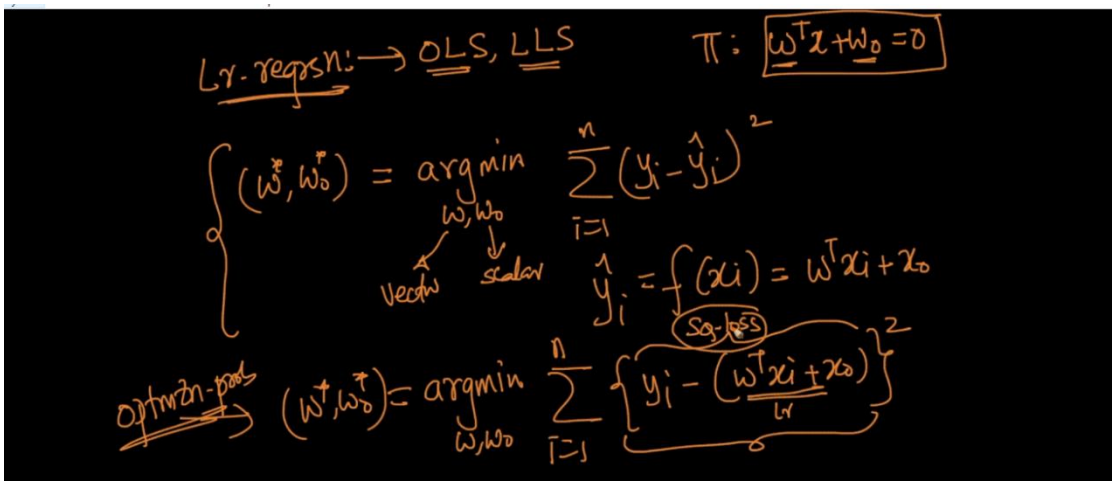
Lets say predicted value = Y^
Original output for C = Y

So error = Y-Y^

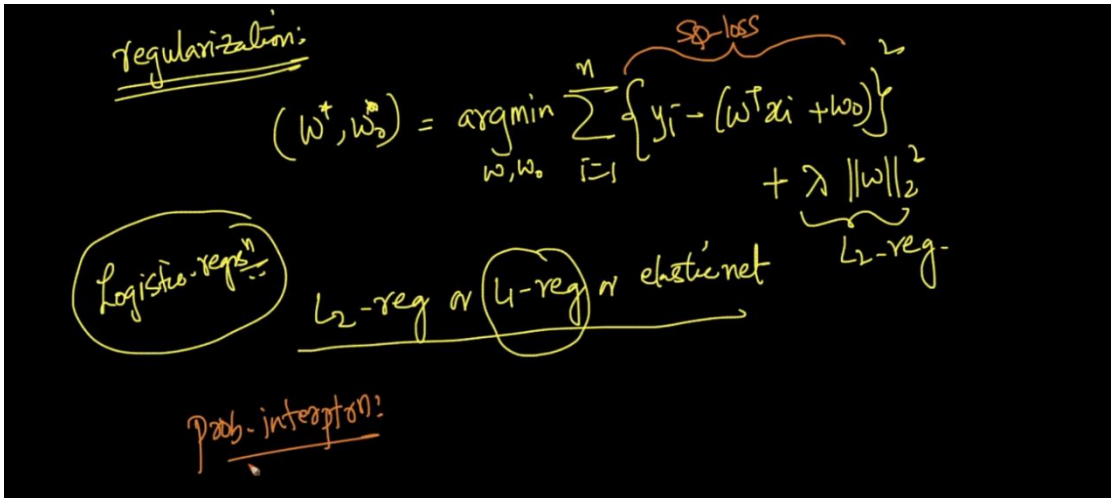




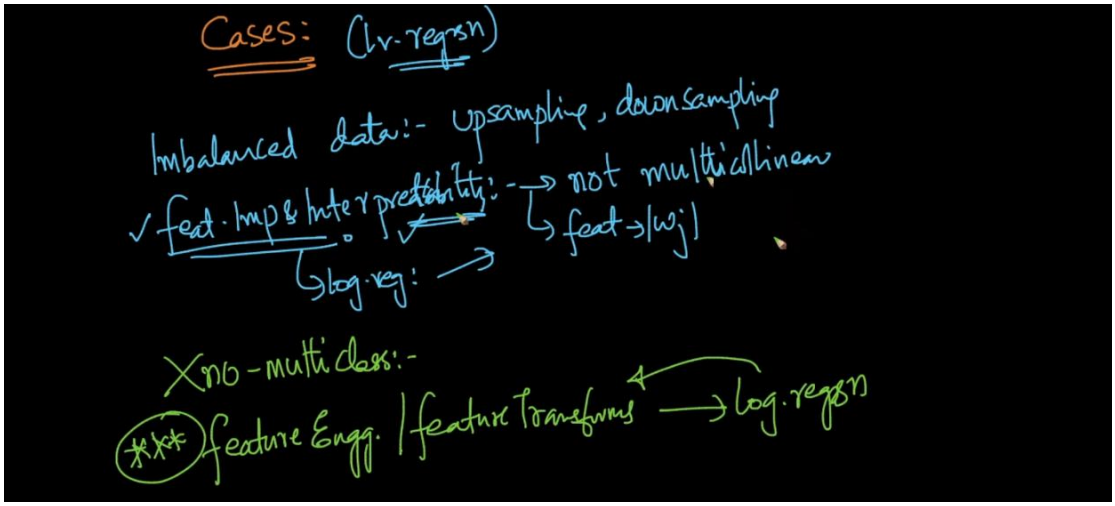
See we get + and - error also so we make it as square .



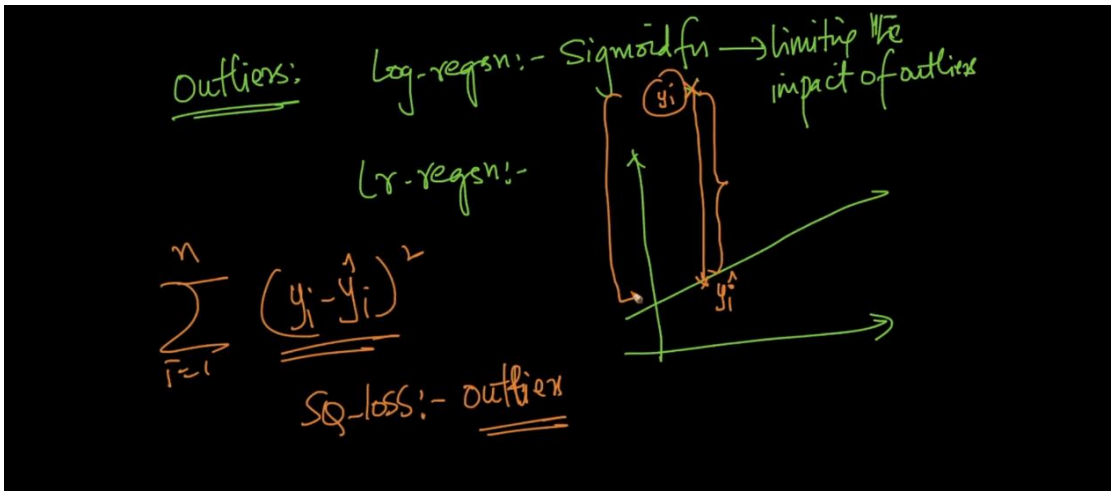
Regularize is same like Logistic only diff is square loss.



All tech of logistic regression can be applied for linear like imbalanced data ,regularization,feature importance .. except Outliers only .



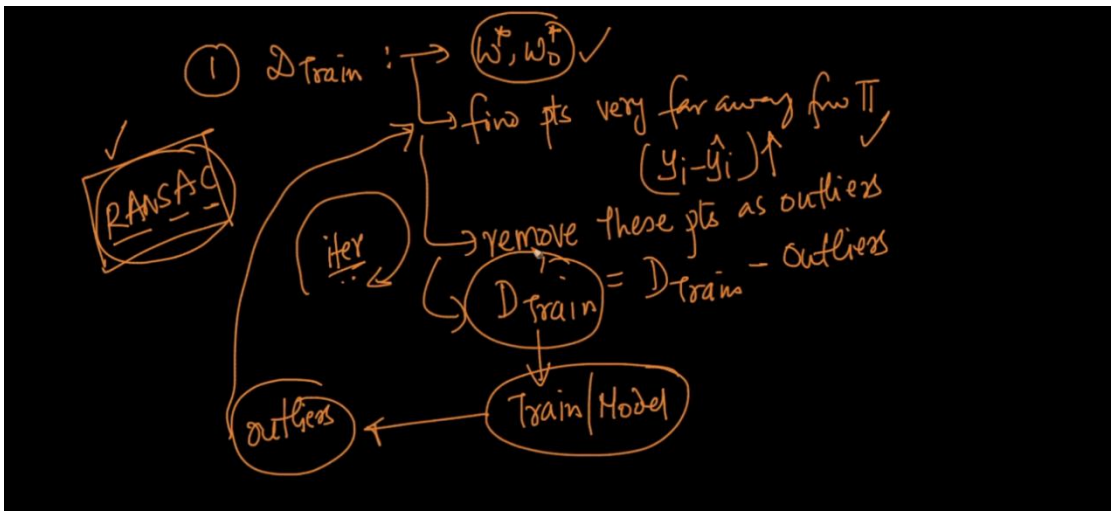
For out l . we used sigmoid for logistic regression here what we will do ?



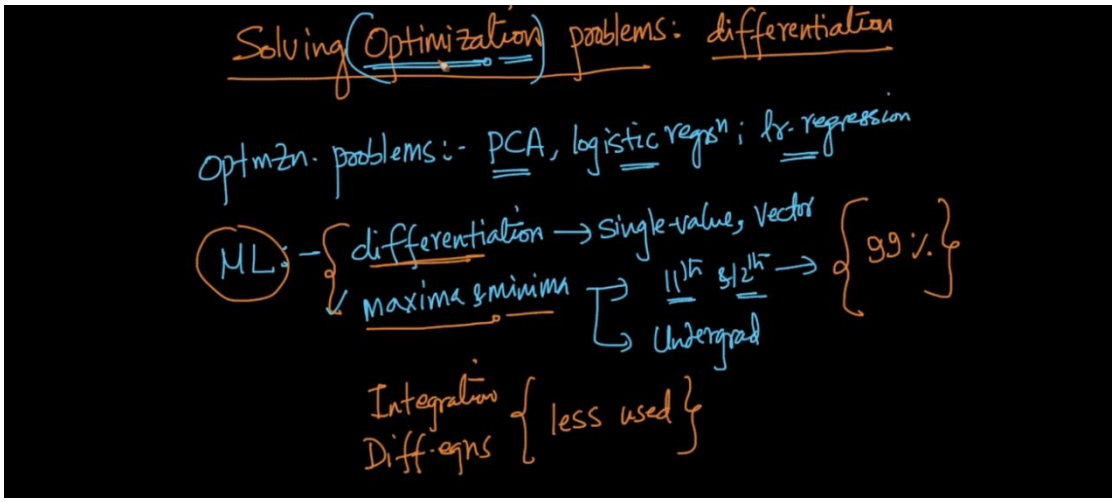
As because of square loss it will be very high for out l. in case of linear regression ..

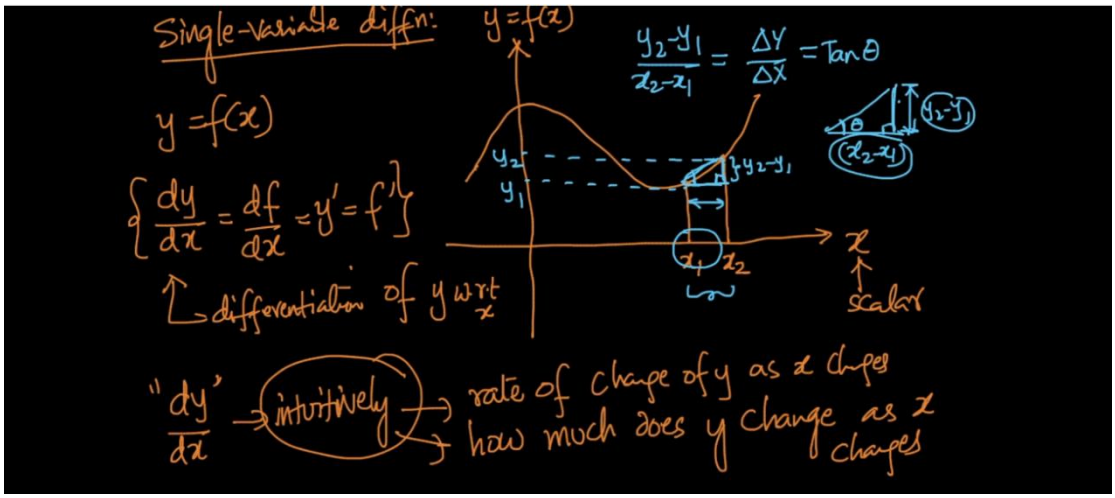
So we will just take train data cal w and w0 so we get plane or line then we cal distance if (y-y^) is

very large means error is very large then we just remove that point we will repeat this process until remove all out l.

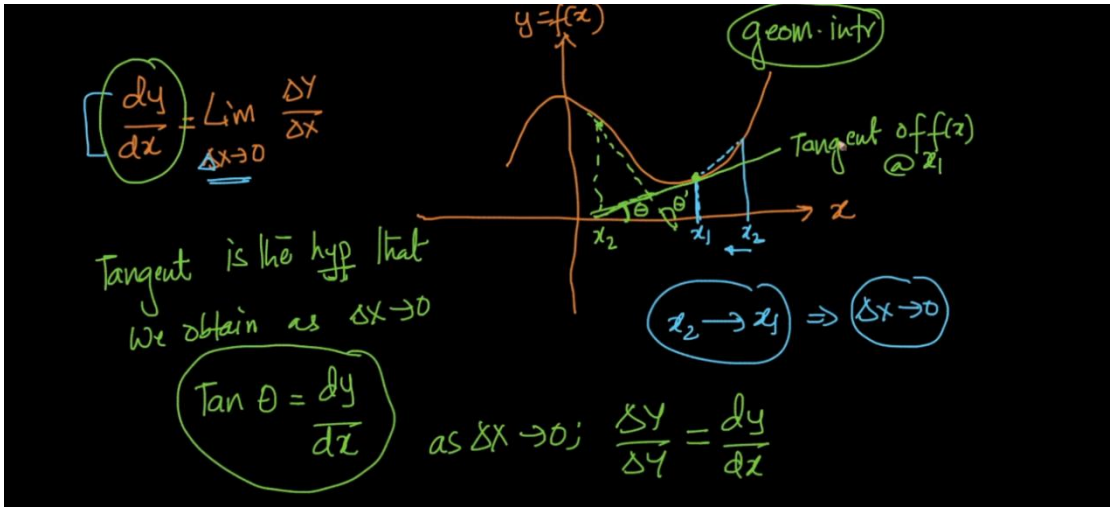


Optimization :





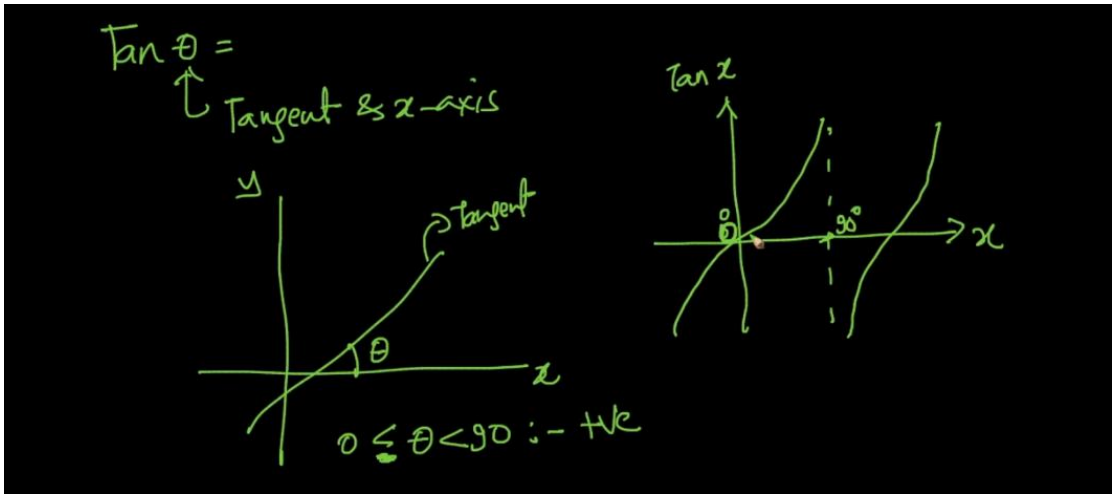
Diff means how much does y change if change in x .
Geometrically we just try to find slope of tangent .

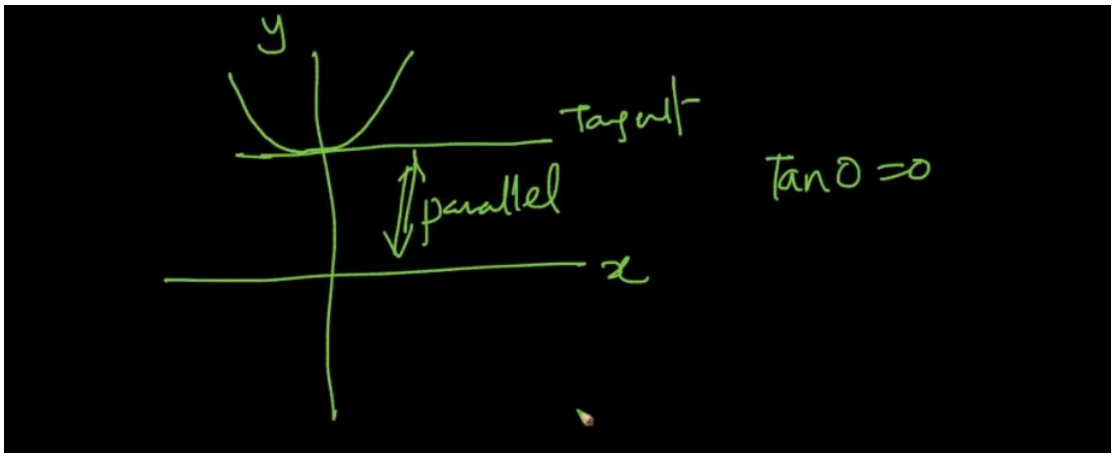


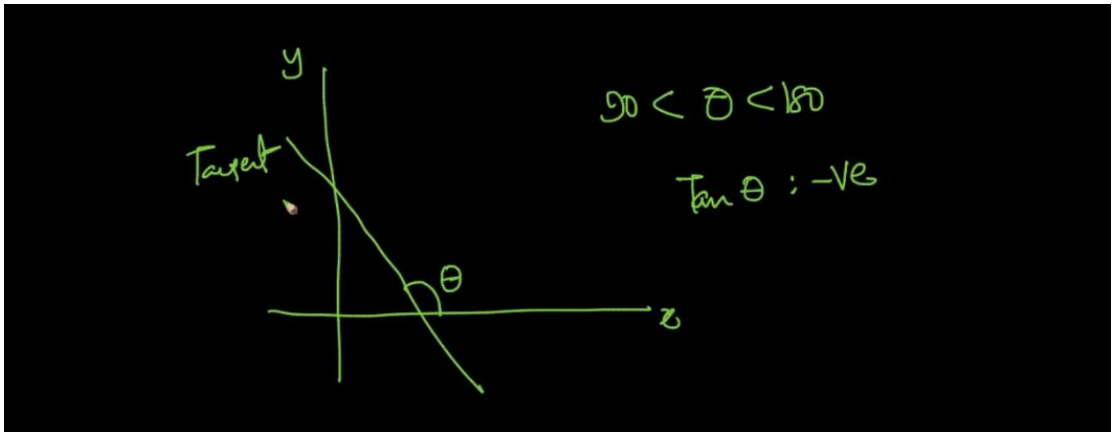
Delta x = x1-x2
Delta y = y1-y2

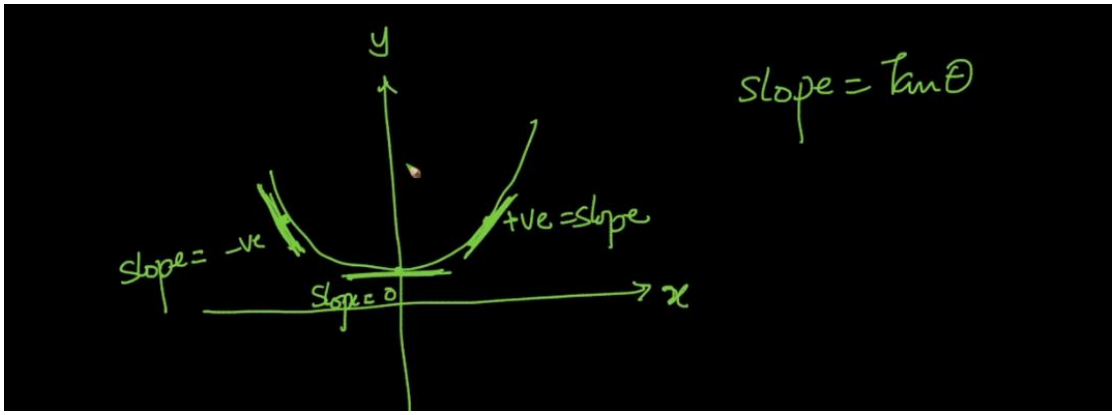
Limit x->0 means if difference between x1 and x2 is very very small close to zero then we will straight line slope . and we just find tan of it .

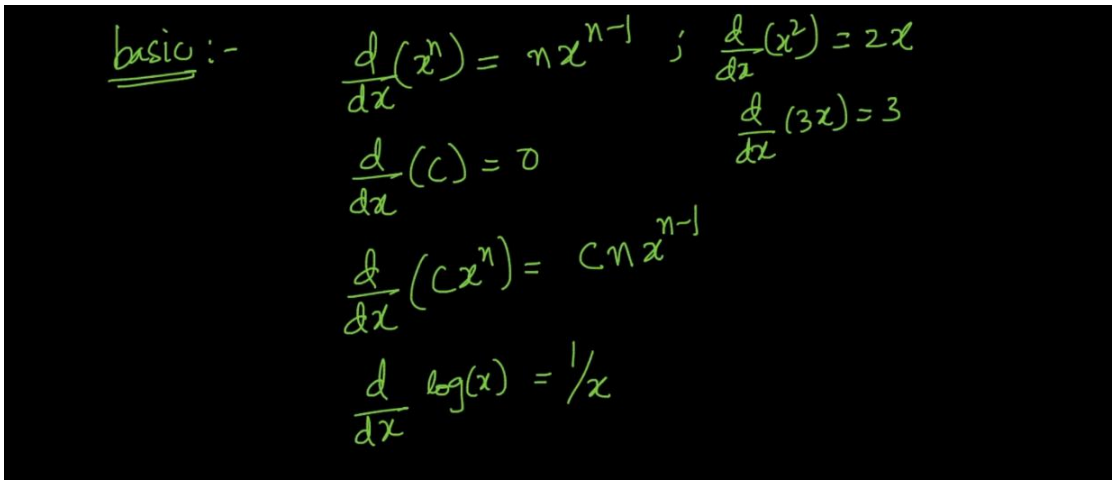
If theta between 0 to 90 values is positive .

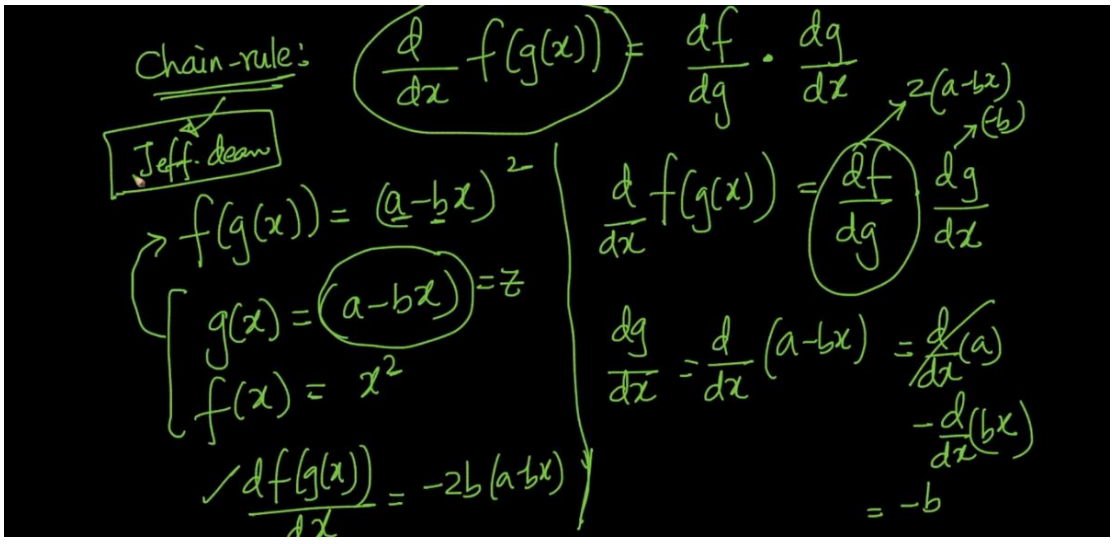






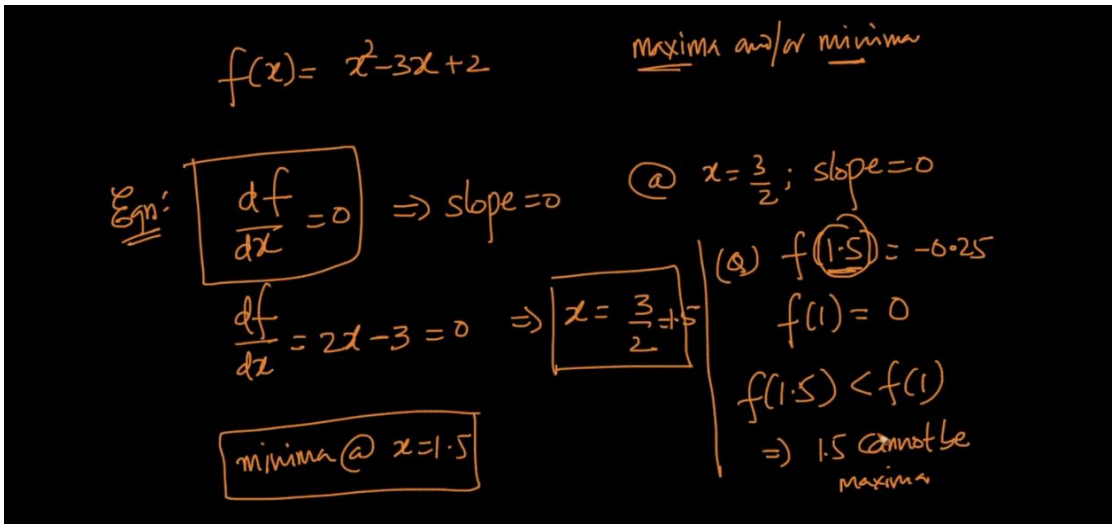






Slope is nothing but tan (tangent line and x axis angle) ..

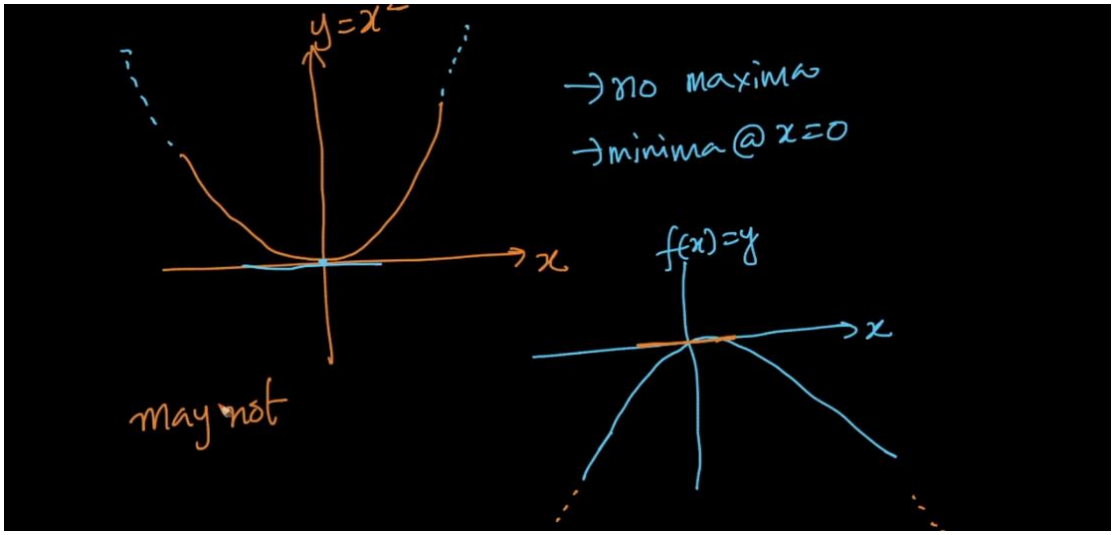
For minim and Maxim slope = 0 .

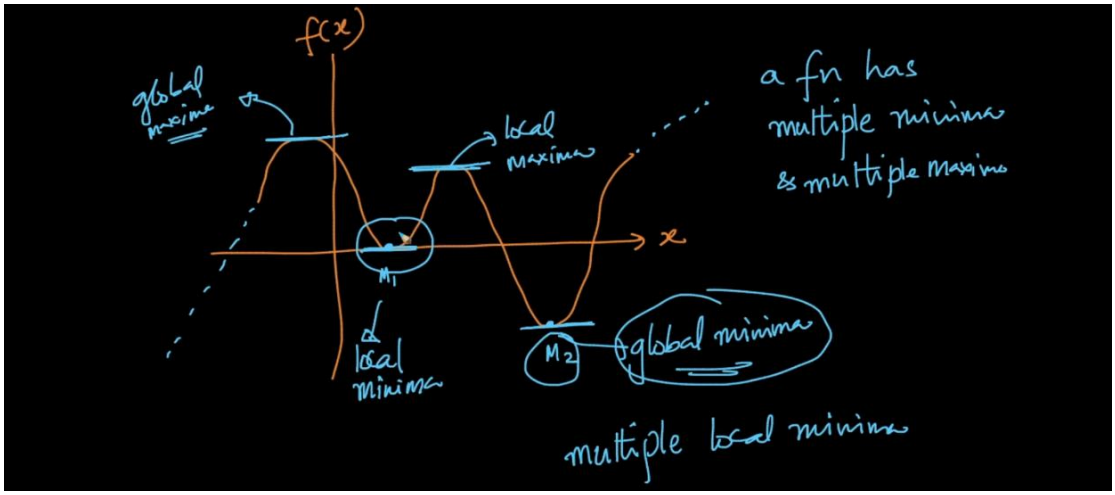


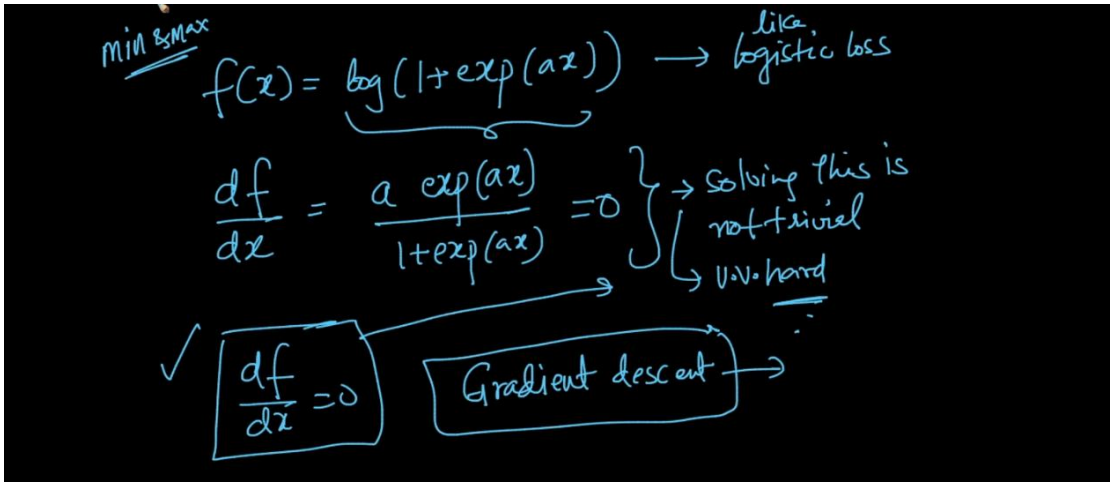
After taking derivative we get x = 3/2 the we put that value in same equation we get f(x) .

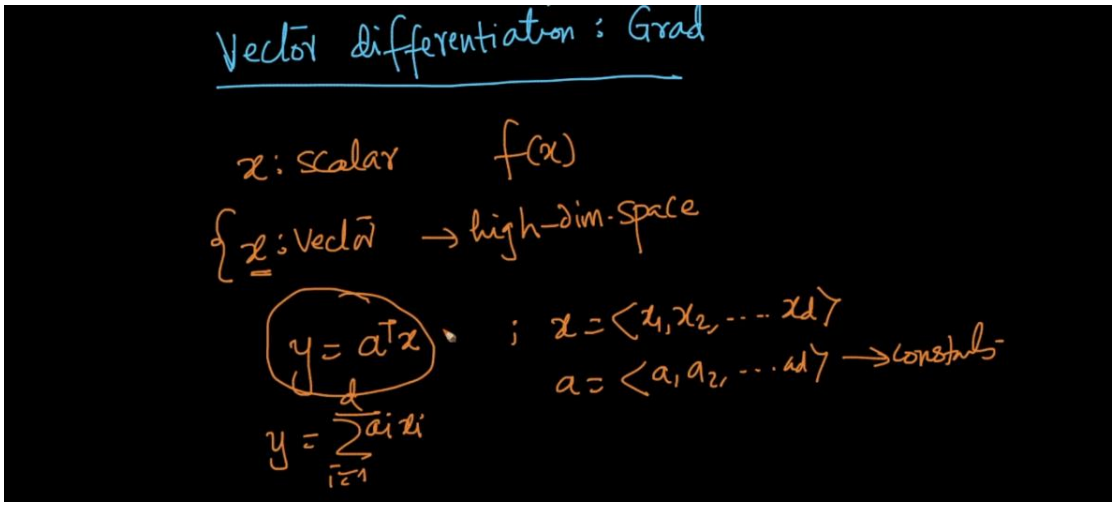
Now we will take very close value less than 1.5 lets say we take 1 then after putting 1 we f(1) = 0 which is > f(1.5) ..

That means f(1.5) is smallest values and that is our minim.


y=x
x
→ no maxima
→ minima @ x=0
f(x)=y
x
may not


f(x)
global maxima
local maxima
x
M1
local minima
M2
global minima
a fn has multiple minima & multiple maxima
multiple local minima

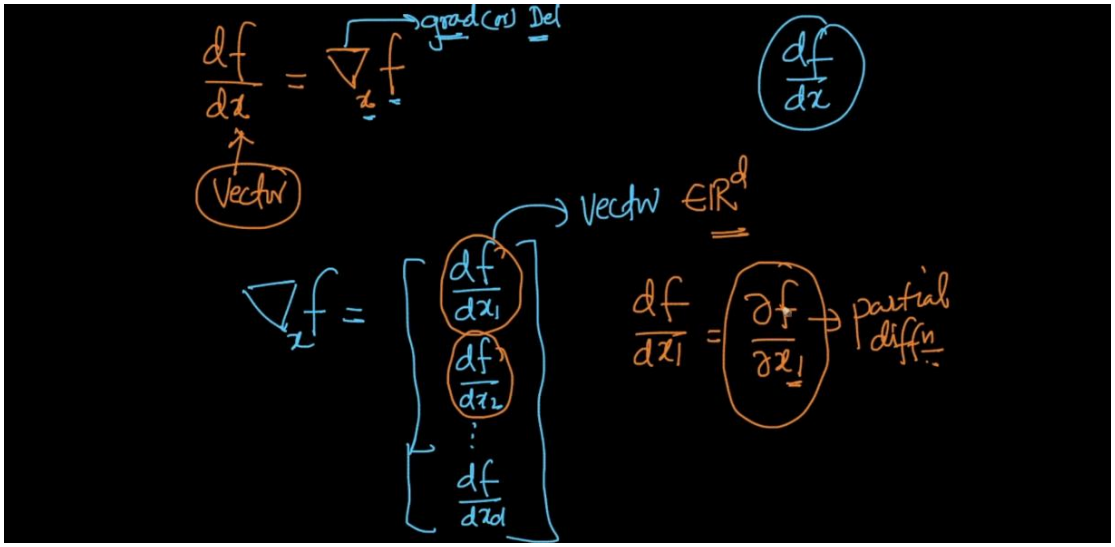


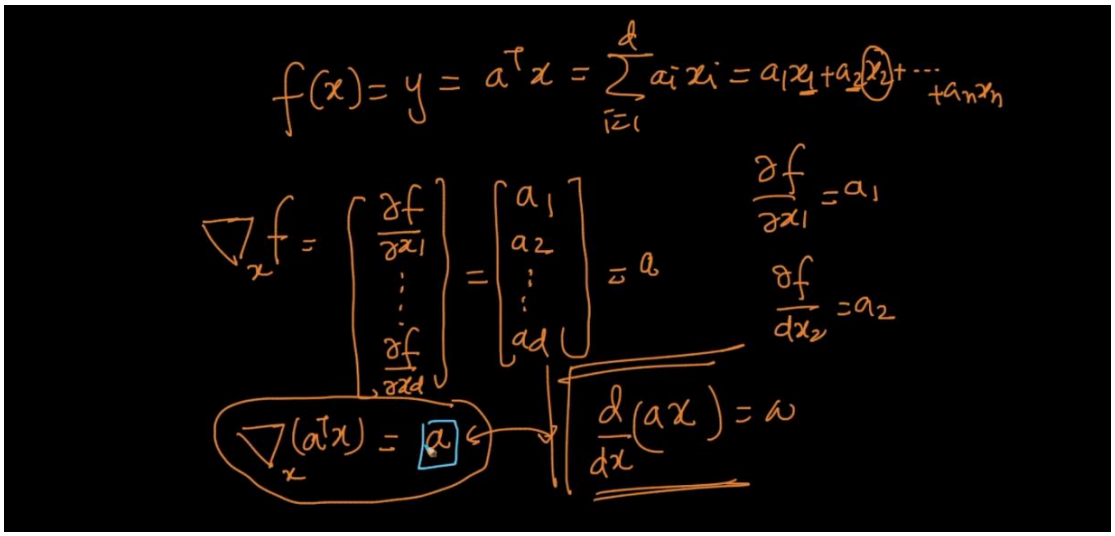


Now for scalar we can directly cal derivatives but now see above image a and x are vectors .

So we use grad to cal derivatives .

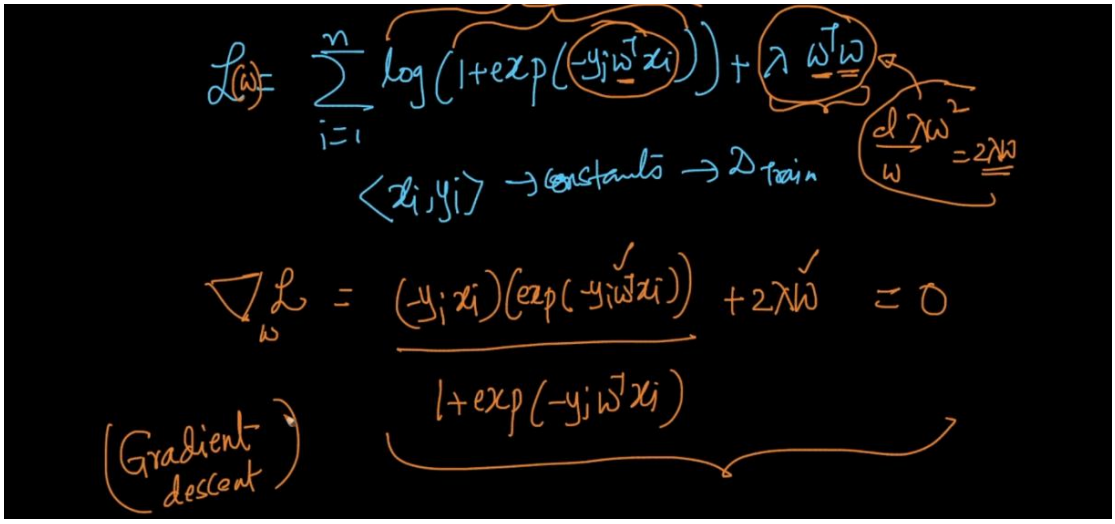
See below image we are taking derivative of f with each x like x1,x2 ….. x_d



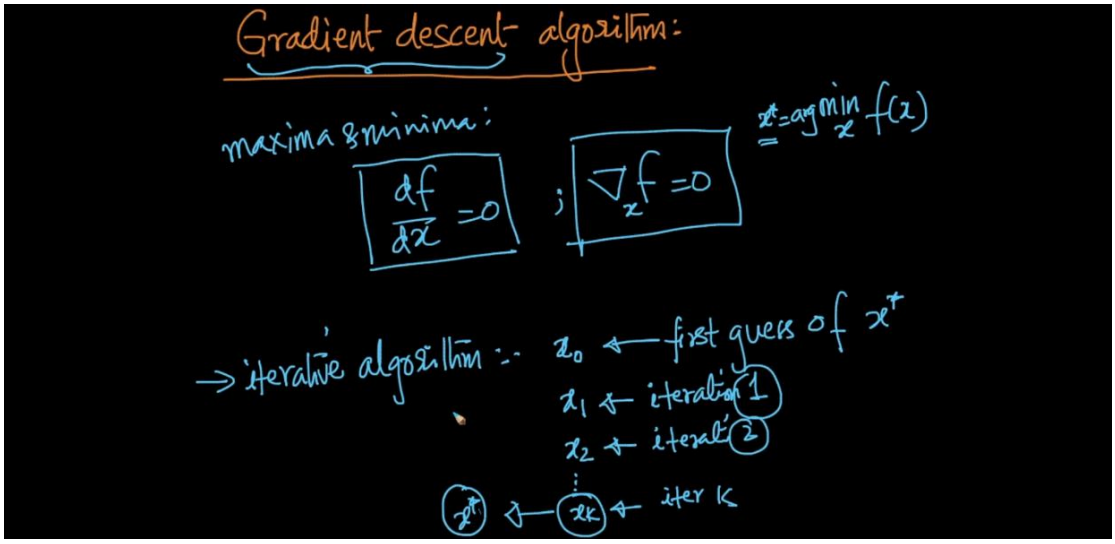


So grad of A trans X is = vector A ..

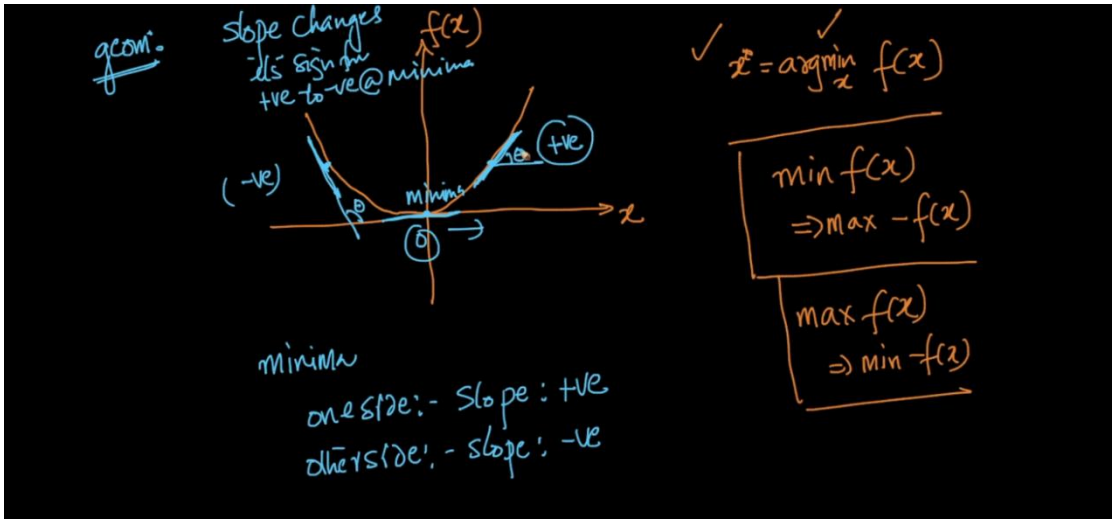
Just take logistic example here we cal grad pf whole equation and if we see solving that vector big equation is very complex work that's why we use Gradient descent .

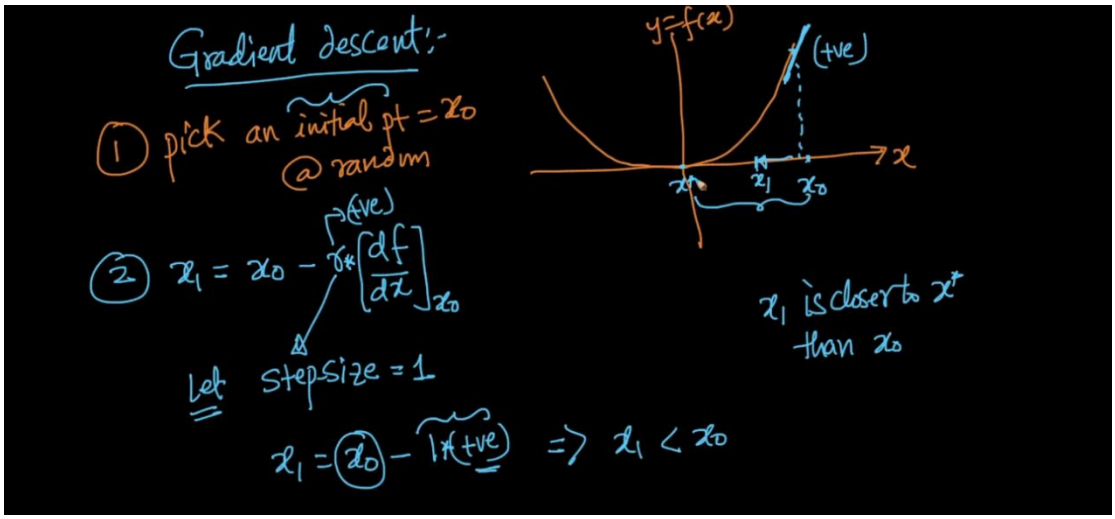


Gradient Descent :



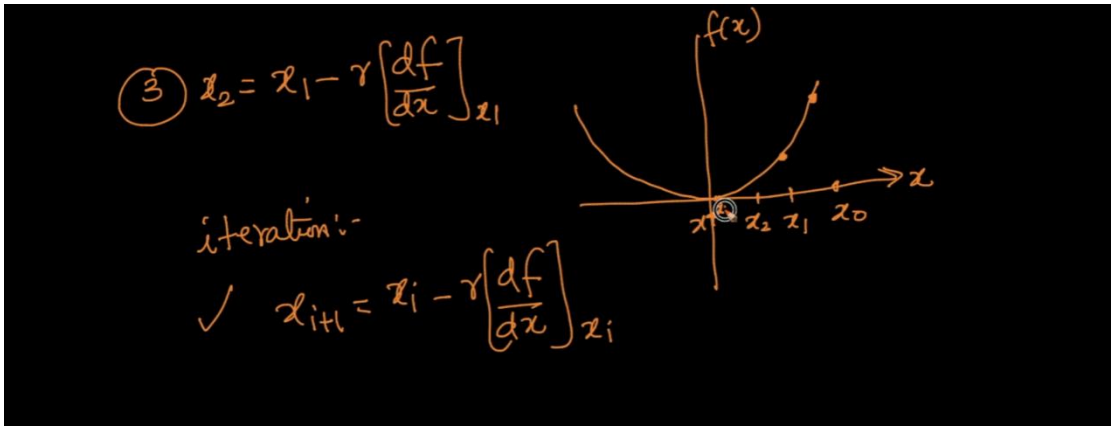
We take any random value as x0 then our algorithm find itself best min .



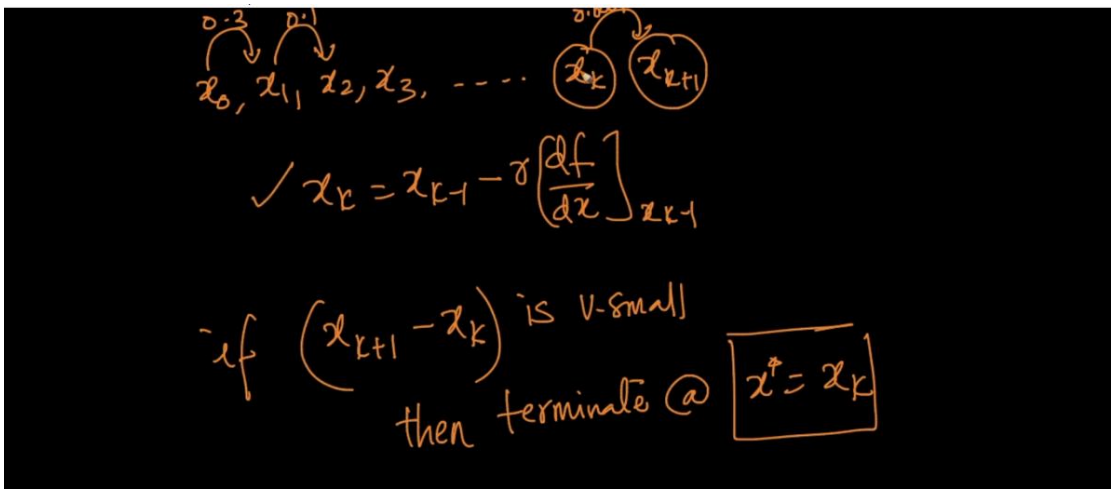


Similarly we will find x2,x3 …. x_n by running in loop. We are taking derivative means we will get slope . here we took random point x0 and we assume that it is max point .

Now as per out formula X0 is max so are subtracting new number from this max number so next number will less than x0 .

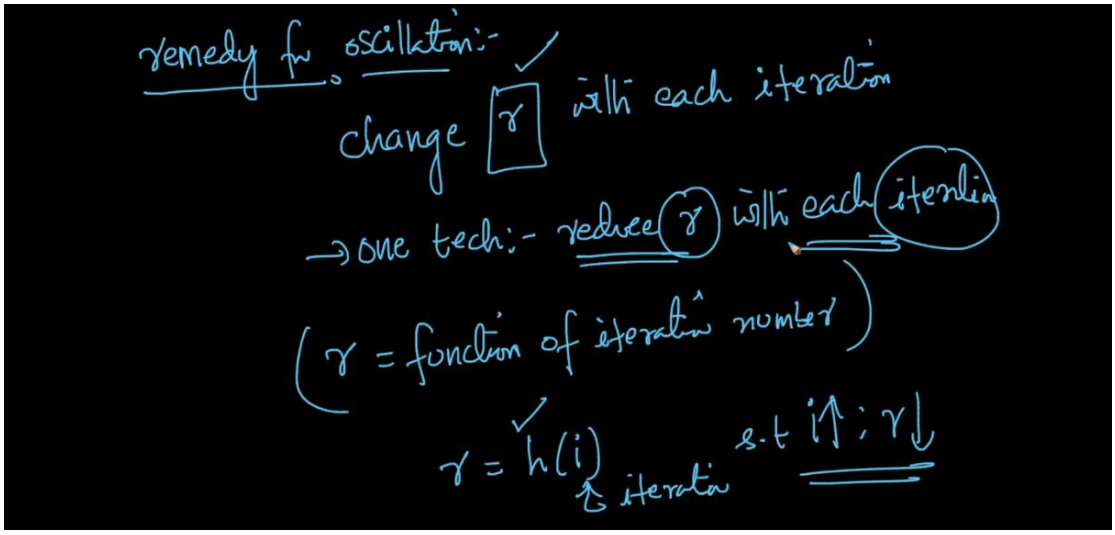


After some point we will see diff is very very small then we stop loop and declare that point as mini .

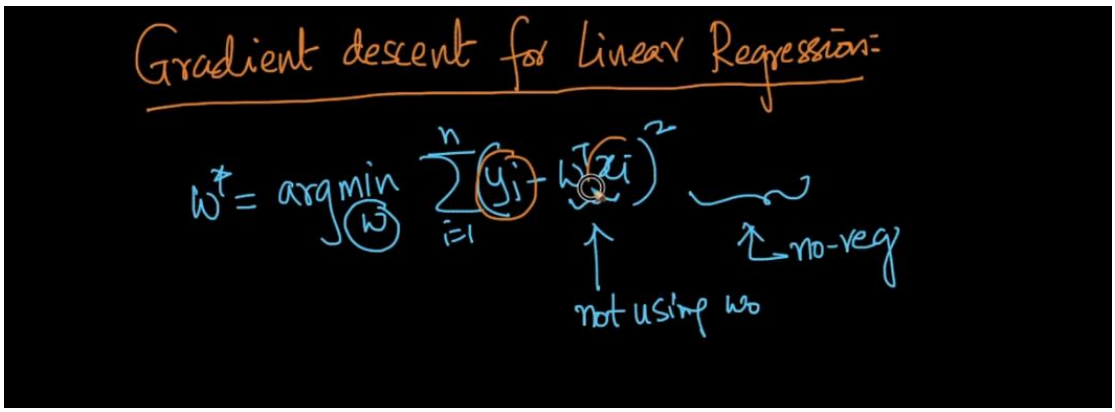


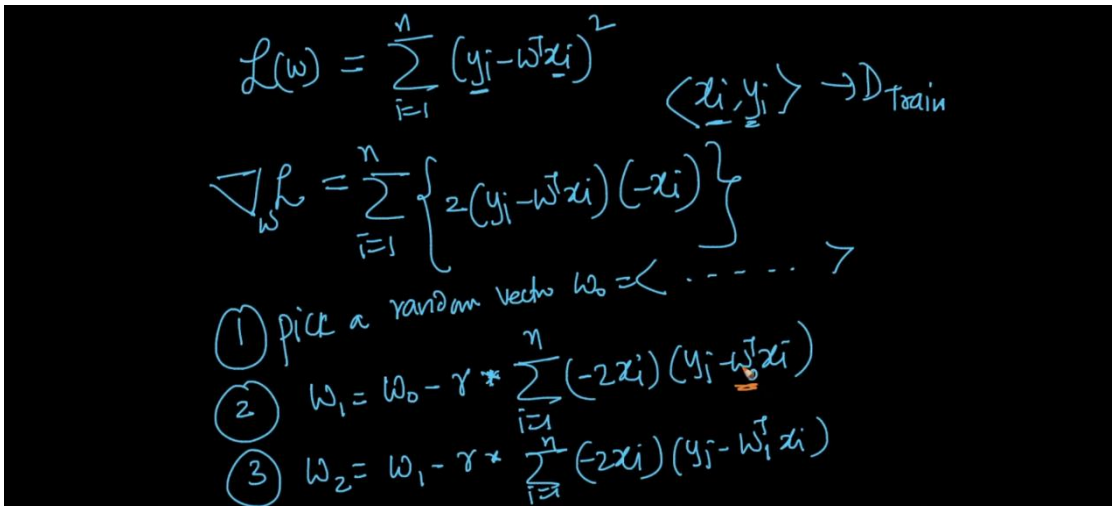
Problem with fixed learning rate is we will never reach at min we will face only oscillation .

So to avoid this problem we have to reduce learning rate after each iteration .



Gradient Descent For Linear Regression :



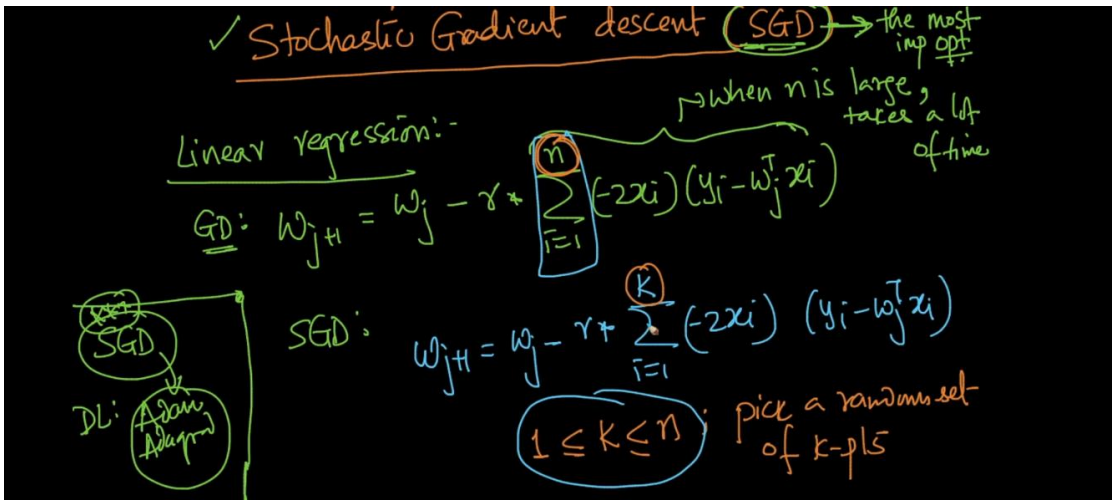


In this way we apply GD to Linear regression and we find perfect w . first we take w0 as random then our algorithm will find best W .

The only problem is see for each step we have to cal 1 to n sum and if lets say n = 1 Million which is possible in ML then it will be very very time consuming so we use advance method called Stochastic GD .

So we do in Stochastic GD we will take random set lets k from all n points .

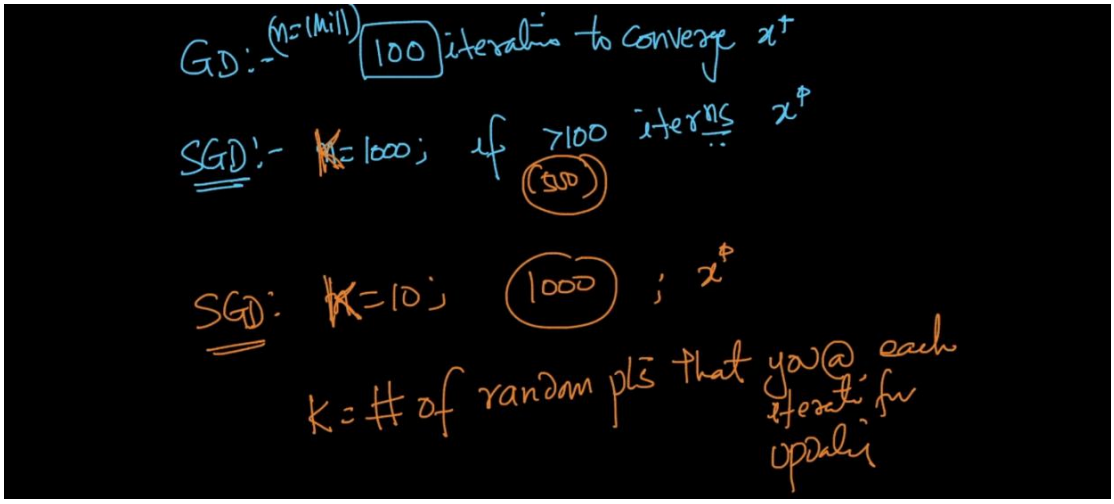
Lets say we have 1M points we take k = 1000 random for each iteration make sure you will change k randomly after each iteration .

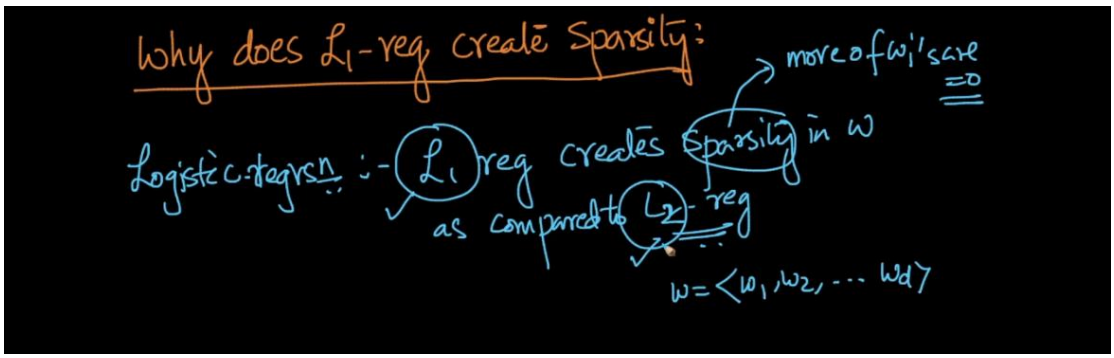


As we are using too much less data in SGD than GD we will get result after lots of iteration .

Lets say GD with n = 1M take 100 Iteration to find minim .

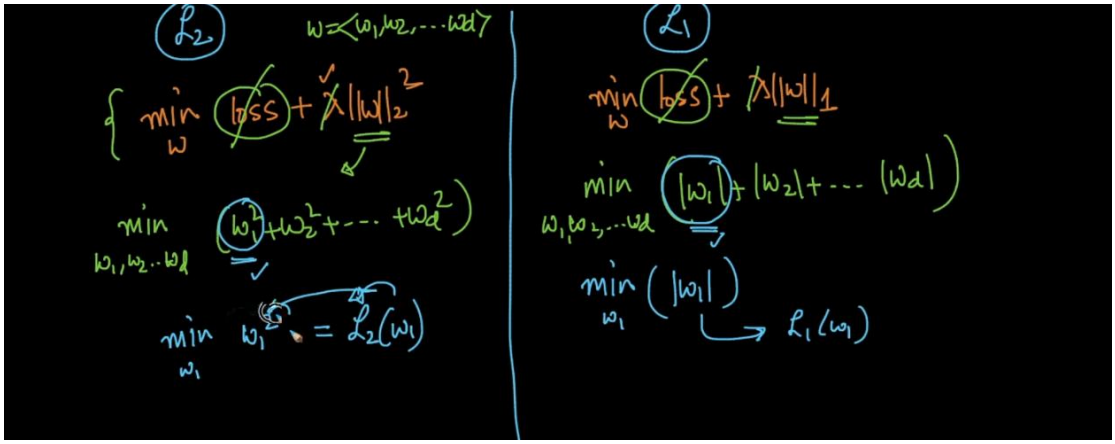
Then SGD with k = 1000 take may be 500 iteration to reach or find min but still its too good because see diff between n and k its huge .



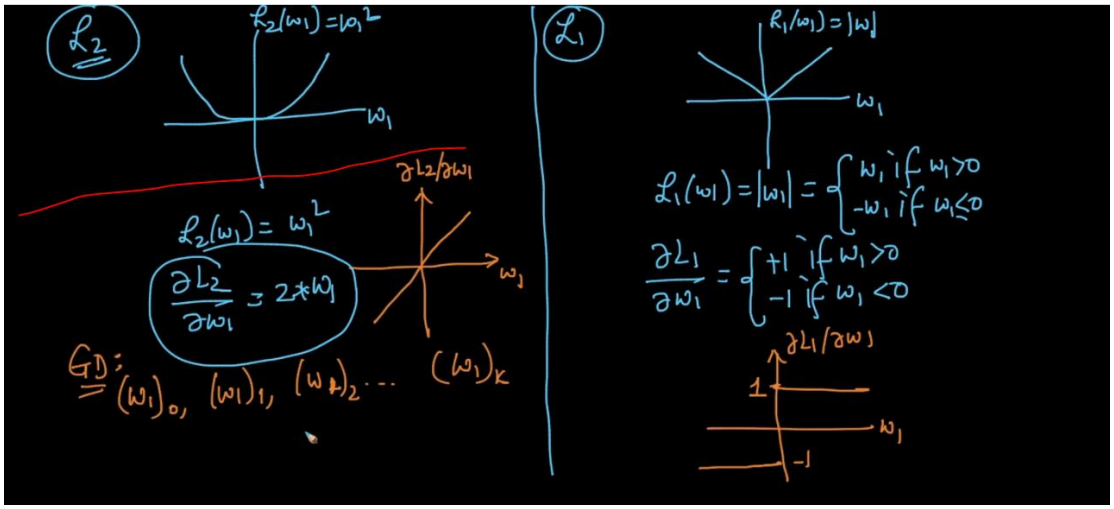


See formulation of L2 and L1 :

Here we want to find best w . See below image of L2 and L1 formula .



Just for simplicity take w1 for both l1 and l2 and find derivative .



See w1 for L1 is just absolute value . absolute means w1 = w1 if w1 > 0
W1 = -w1 if w1 < 0

Now we will find derivative of it .

See l2 w1 and l1 w1 we have very small difference in equation . Gradient of L1 is constant hence it will towards 0 but gradient of L2 always change

see graph so after some point we will see very small difference between w_j+1 and w_j.