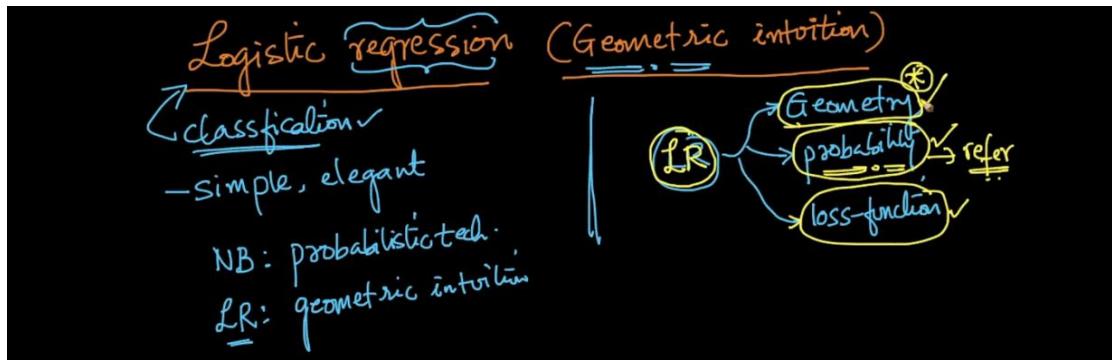
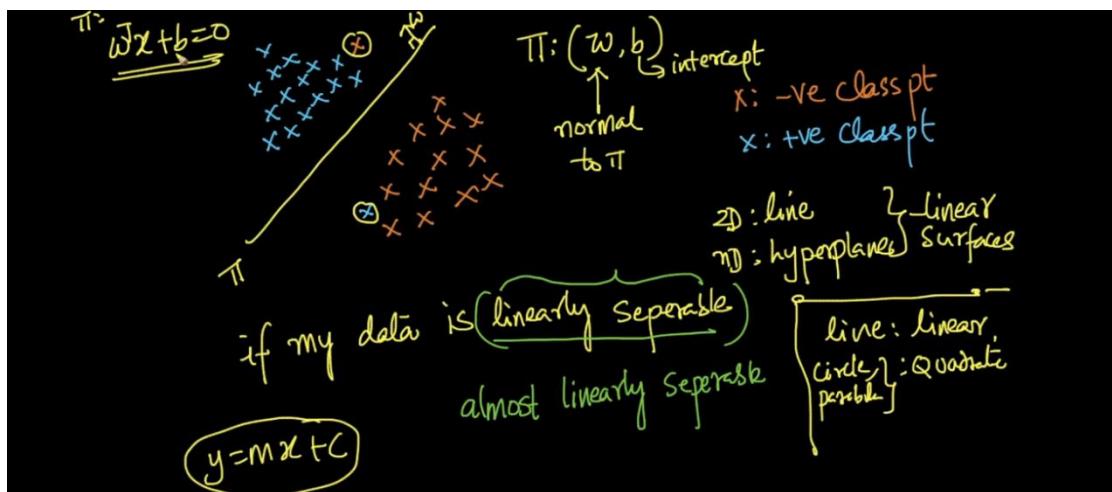


Logistic Regression :



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



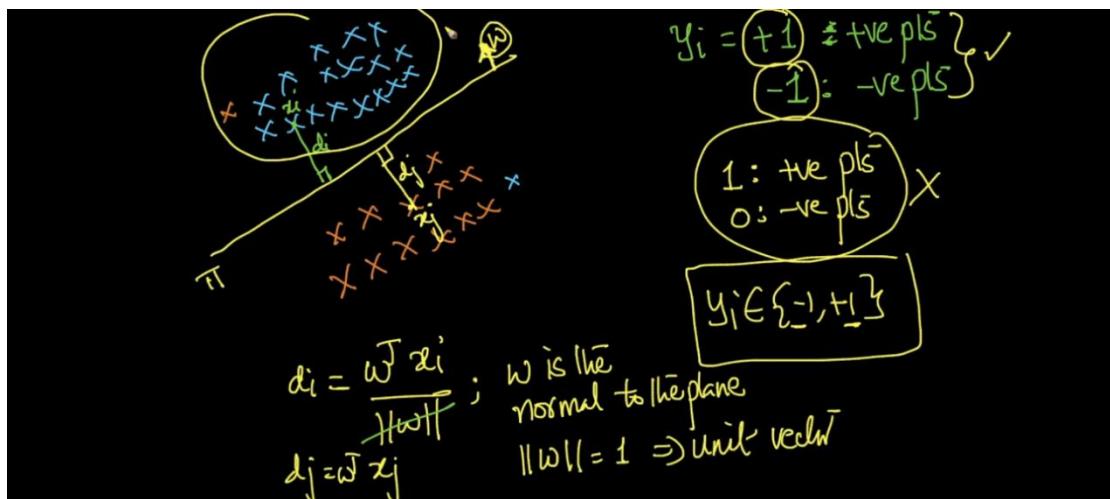
$w^T x + b$ if line passing through then b will be 0 .

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

if the π passes through origin:- $b=0$
 $w^T x = 0$

$$\pi: w^T x + b = 0$$

$w \in \mathbb{R}^d$; $w \in \mathbb{R}^d$ \rightarrow vec
 $b \in \mathbb{R}$ \leftarrow scalar



See above image lets take new 2 points x_1 and x_2 . now x_1 is in same direction of w hence it will + and x_2 opposite .

Case 1: $y_i * w^T x_i > 0 \rightarrow$ if $y_i = +1 \rightarrow +ve \text{ pt}$
 $y_i = +1$
 $w^T x_i > 0 \Rightarrow$ classifier is saying its the pt
 w is correctly classifying the pt

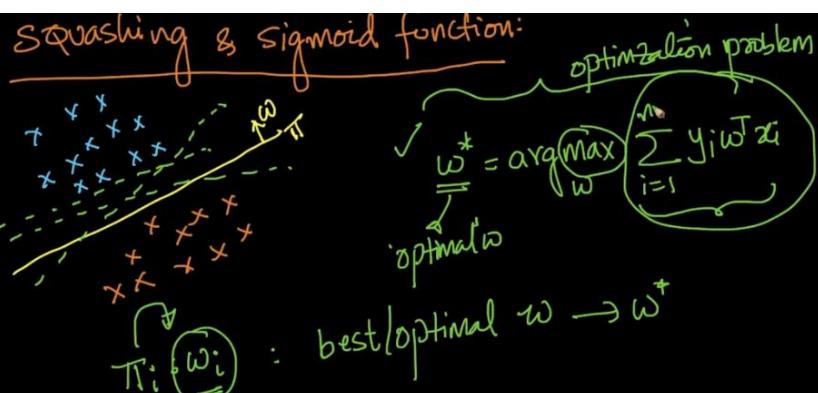
Case 2: $y_i = -1 : -ve \text{ pls}$
 $w^T x_i < 0 \Rightarrow$ LR concluding that x_i is a -ve pt
 $y_i * w^T x_i > 0$

both +ve & -ve pts
 $y_i w^T x_i > 0 \Rightarrow$ the LR model is correctly classifying the pt x_i

Case 3: $y_i = +1$ (+ve pt)
 $w^T x_i \leq 0 \Rightarrow$ LR is saying x_i is -ve class
 $y_i w^T x_i < 0 \Rightarrow$ $y_i = +1$ LR: -1 misclassified

Case 4: $y_i = -1 \Rightarrow$ -ve class
 $w^T x_i > 0 \Rightarrow$ LR is saying x_i is +ve pt
misclassified $y_i w^T x_i < 0$

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 I .it will perform very very bad . lets see below images .

Y will be +1 or -1

$$\underset{w}{\operatorname{argmax}} \sum_{i=1}^n y_i w^T z_i$$

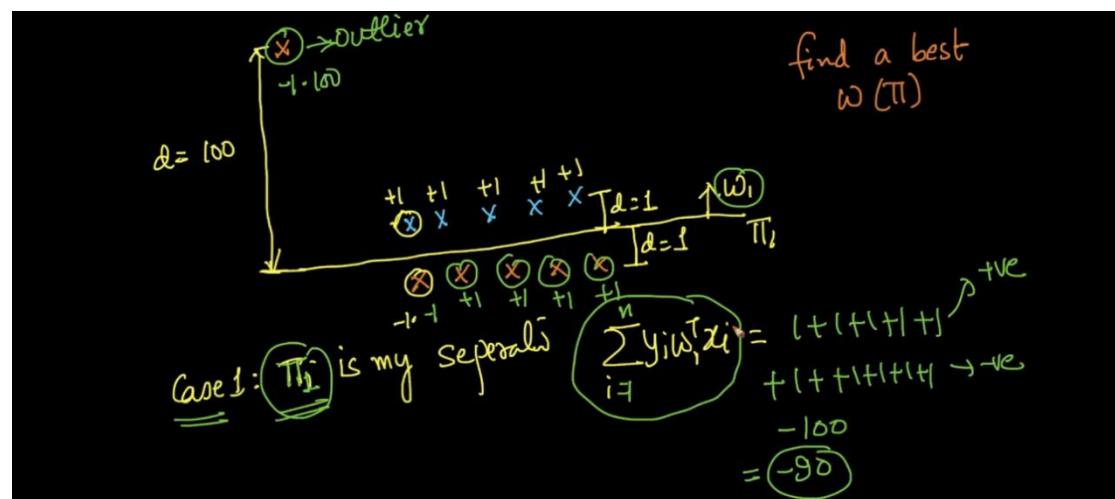
→ signed distance

$w^T z_i$ dist from x_i to π (w is a unit vector)

$y_i w^T z_i$: +ve $\Rightarrow \pi$ as defined by w correctly classifies x_i

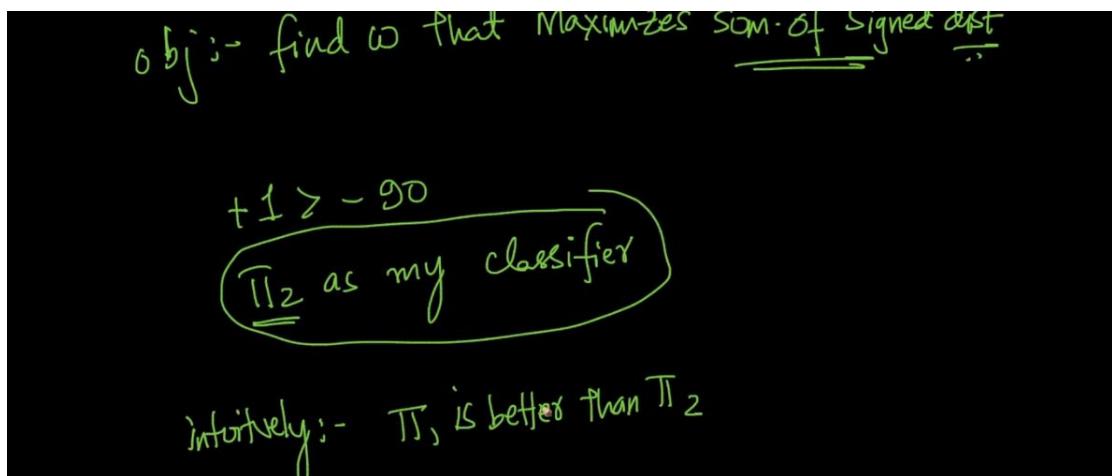
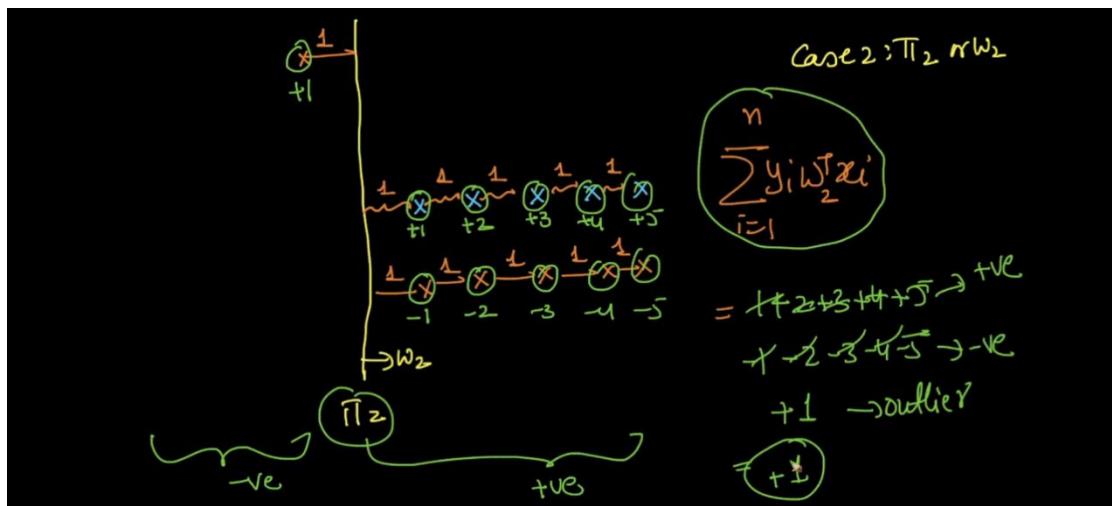
-ve \Rightarrow incorrectly classifies x_i

See case 1 :



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

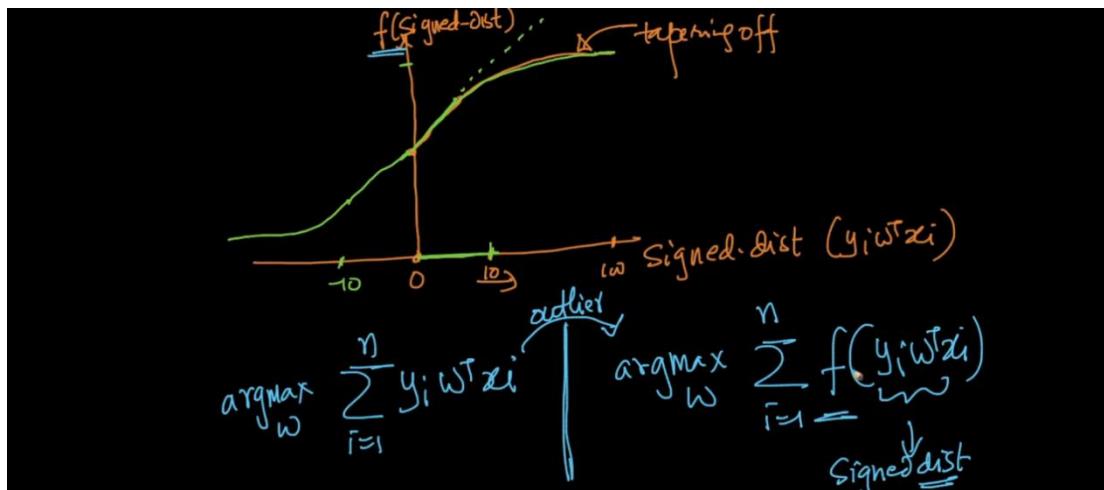
Since we get very bad value we will move or plane so next we get :



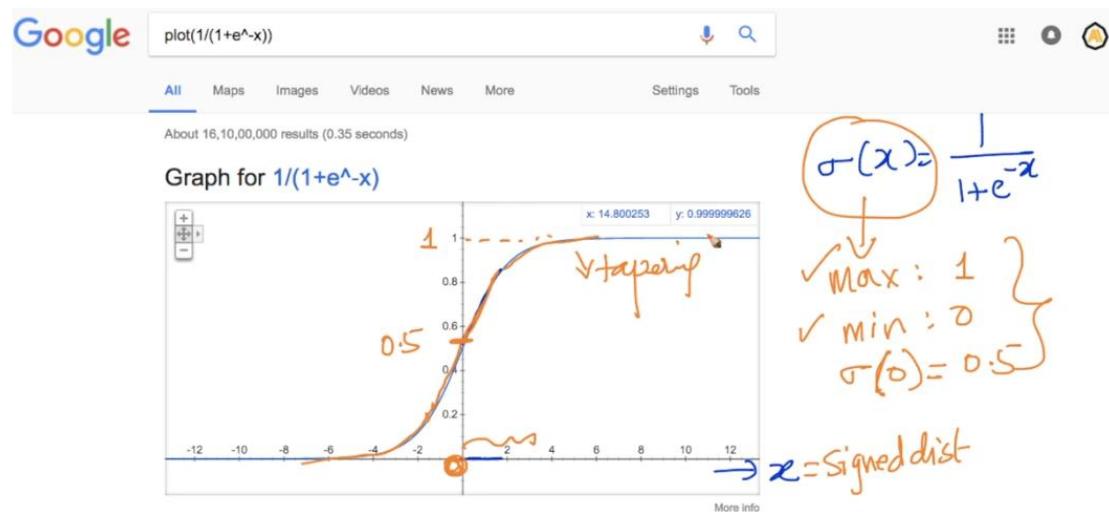
Practically π_1 is best but we get wrong plane just because of 1 outlier.

So we need any function which will handle this type of outlier. 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 .

Optimization problem:

$$\checkmark \left[w^* = \operatorname{argmax}_w \sum_{i=1}^n \frac{1}{1 + \exp(-y_i w^T x_i)} \right] \rightarrow \text{optimization problem}$$

→ monotonic functions

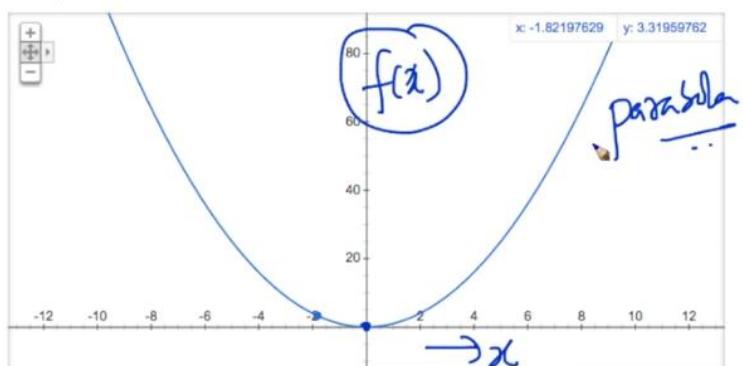
Log x is a monotonic function means as x inc $\log(x)$ also increase .



See below image that is opt problem it means find best x which minimize x^2 .

$\xrightarrow{\text{opt. prob.}}$ $x^* = \arg \min_x x^2$ Mimima & Maxima
 ≈ 0 $\xrightarrow{\text{best}} x$
 $f(x) = x^2$
 $x^* = \arg \min_x f(x)$

Graph for x^2



$$f(x) = x^2$$

See $x = 0$ which minimize $f(x)$.

If $g(x)$ is a monotonic fn.
 $\left\{ \begin{array}{l} \arg \min_x f(x) = \arg \min_x g(f(x)) \\ \arg \max_x f(x) = \arg \max_x g(f(x)) \end{array} \right.$

geometric

$$\hat{w} = \arg \min_w \sum_{i=1}^n \log (1 + \exp(-y_i w^\top x_i))$$

Prob. method

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

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

Weight-Vector

$$\hat{w} = \arg \min_w \sum_{i=1}^n \log (1 + \exp(-y_i (\hat{w}^\top x_i)))$$

weight-vecⁿ $w = \langle w_1, w_2, w_3, w_4, \dots, w_d \rangle$

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

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

$$f_1, f_2, f_3, \dots, f_d$$

decision: $x_Q \rightarrow y_Q$ $\begin{cases} \text{if } w^\top x_Q > 0 \text{ then } y_Q = +1 \\ \text{if } w^\top x_Q < 0 \text{ then } y_Q = -1 \end{cases}$

prob.interp: $\Pr(w^\top x_Q) = P(y_Q = +1)$

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

interpretation of ω :

Case 1 If $w_i = \text{ve}$, $x_{qj_i} \uparrow \Rightarrow (w_i x_{qj_i}) \uparrow$
 $\Rightarrow (\sum_{i=1}^n w_i x_{qj_i}) \uparrow$
 $\Rightarrow \sigma(w^T x_q) \uparrow$
 $\Rightarrow p(y_q = +1) \uparrow$

Case 2: If $w_i = -ve$
 $x_{qj_i} \uparrow \Rightarrow (w_i x_{qj_i}) \downarrow$
 $\Rightarrow (\sum_{i=1}^n w_i x_{qj_i}) \downarrow$
 $\Rightarrow \sigma(w^T x_q) \downarrow$
 $\Rightarrow p(y_q = +1) \downarrow$
 $\Rightarrow p(y_q = -1) \uparrow$

$p(y_q = +1) = 1 - p(y_q = -1)$

L_2 regularization : Overfitting vs Underfitting:

$$\omega^t = \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^n \log(1 + \exp(-y_i \underline{w^T x_i}))$$

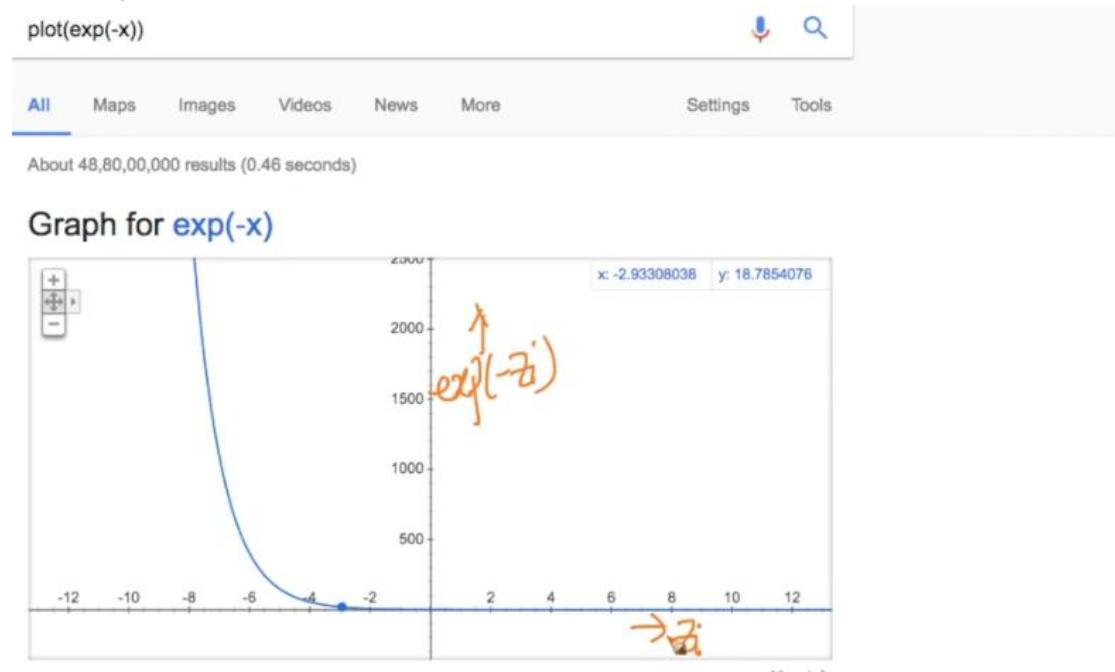
let $\underline{z_i} = y_i \underline{w^T x_i}$

$$= \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^n \log(1 + \exp(-z_i))$$

Lets take first exp part of equation .

$\text{Exp}(-Z_I)$

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 $>= 0$.

$$\sum_{i=1}^n \log \left(1 + \underbrace{\exp(-z_i)}_{>0} \right)$$

$\therefore \exp(-z_i) > 0$

$$\log \left(1 + \underbrace{\exp(-z_i)}_{>0} \right) > 0$$

$\log(1) = 0$
 $\log(2) > \log(1)$
 $\left\{ \begin{array}{l} \log(1+\delta) > \log(1) \\ \delta > 0 \end{array} \right\}$

$$w^* = \arg \min_w \left[\sum_{i=1}^n \left(\log(1 + \exp(-z_i)) \right) \right] > [0]$$

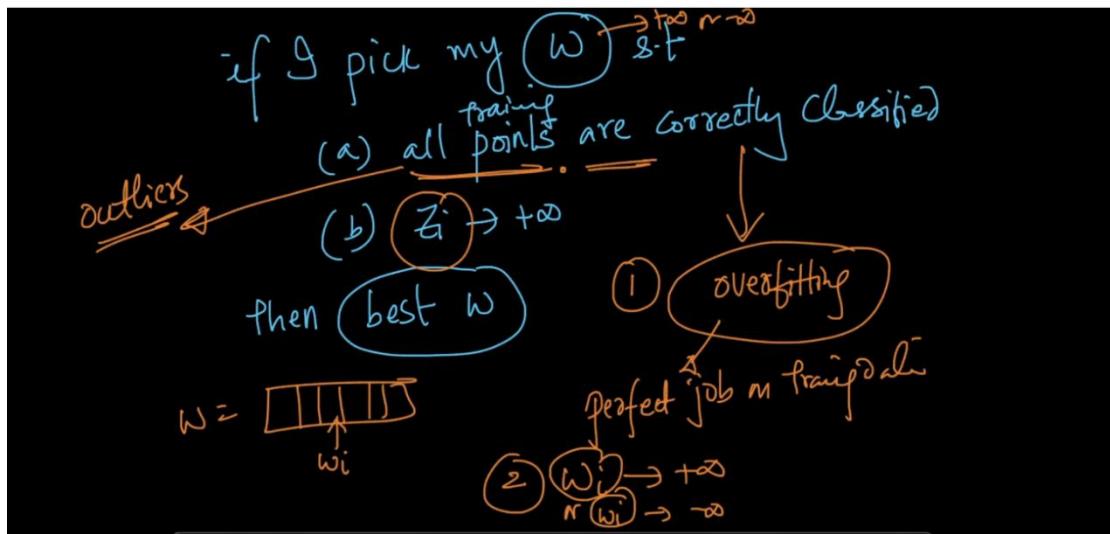
minimal value of $\sum_{i=1}^n \log(1 + \exp(-z_i))$ is "0"
it occurs when $z_i \Rightarrow \infty$ for all i

We want to find min value for w and min here is 0 .

We achieve these when sum of all $z_l = \infty$.

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 = ∞ .

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 over fit issue .



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

regularization

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

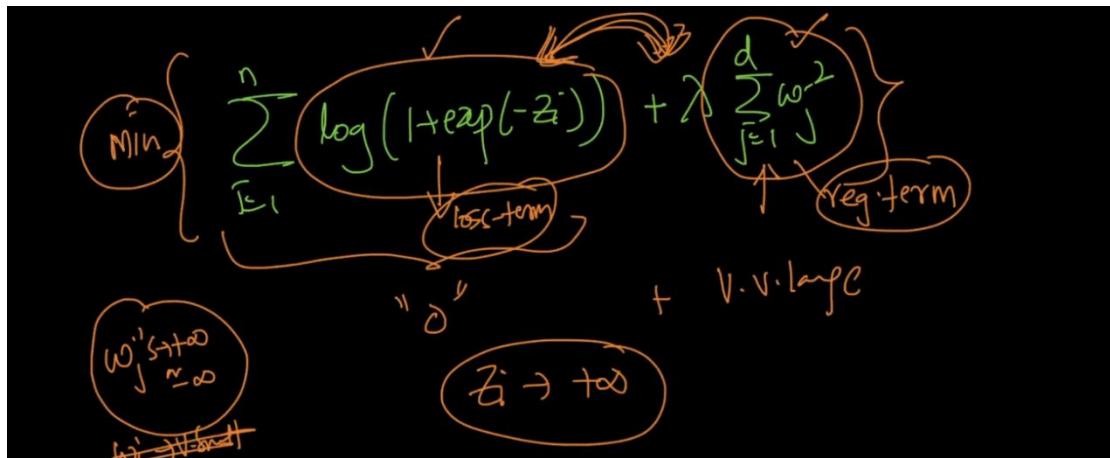
$w_j \rightarrow \infty$ $w_j \rightarrow -\infty$

loss-term $\sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i))$

$\lambda w^T w \rightarrow \|w\|_2^2$

$\lambda \sum_{j=1}^d w_j^2$ regularization term

Because of that new term it will never allow to become $+ \infty$ to w^* . It will always oppose.



So if we use

Lambda = 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 ,.

$$\min \left(\text{loss-fn over training data} + \text{reg.} \right)$$

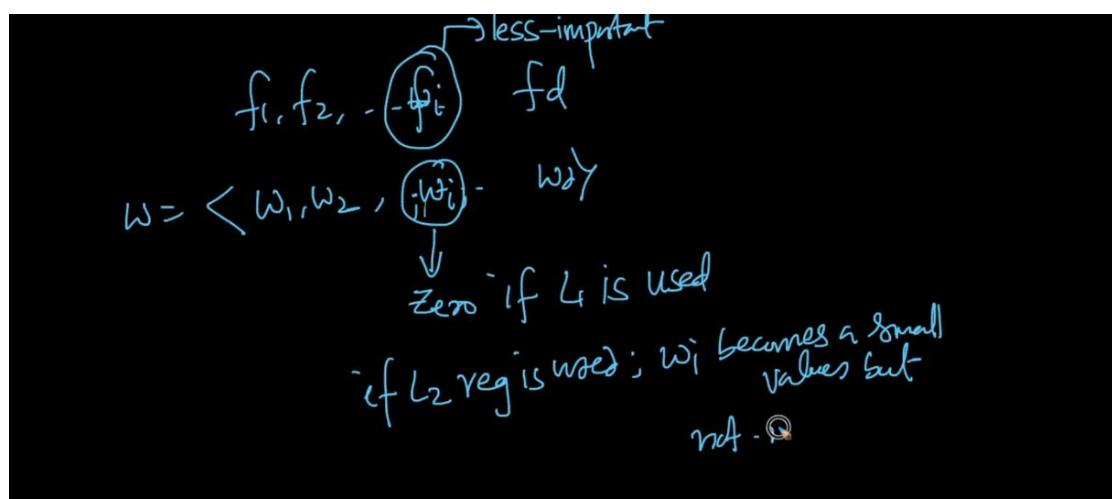
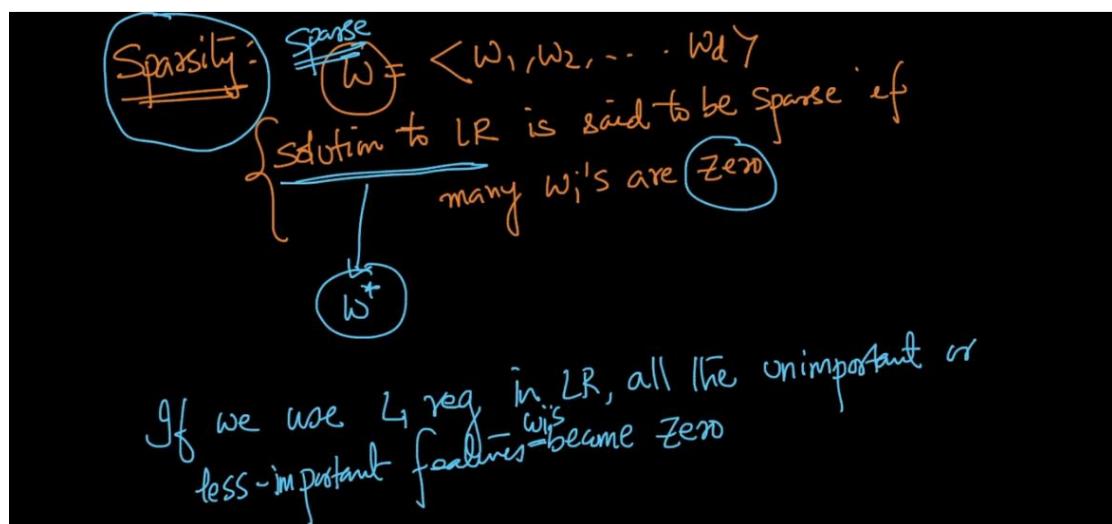
cross-validation

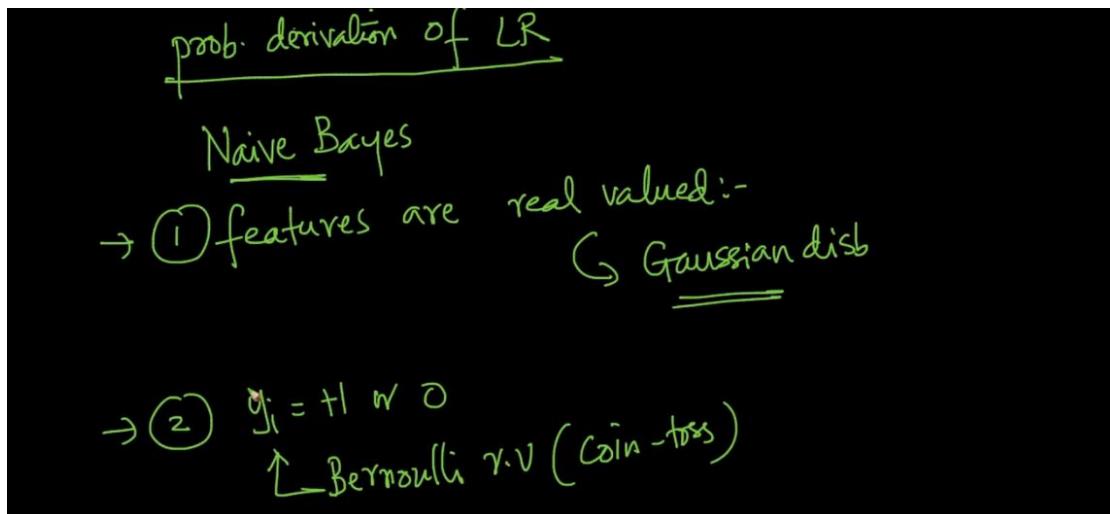
$\lambda = 0 \Rightarrow$ overfit \rightarrow high variance
 $\lambda = \text{v. large} \Rightarrow$ underfit \rightarrow high bias

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 .





Secure | https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

3.1 Form of $P(Y|X)$ for Gaussian Naive Bayes Classifier

Here we derive the form of $P(Y|X)$ entailed by the assumptions of a Gaussian Naive Bayes (GNB) classifier, showing that it is precisely the form used by Logistic Regression and summarized in equations (16) and (17). In particular, consider a GNB based on the following modeling assumptions:

- Y is boolean, governed by a Bernoulli distribution, with parameter $\pi = P(Y=1)$
- $X = \langle X_1 \dots X_n \rangle$, where each X_i is a continuous random variable
- For each X_i , $P(X_i|Y=y_k)$ is a Gaussian distribution of the form $N(\mu_{ik}, \sigma_i)$
- For all i and $j \neq i$, X_i and X_j are conditionally independent given Y

Below image both function work as same .

geom: $\omega^* = \arg \min_{\omega} \sum_{i=1}^n \log(1 + \exp(-y_i \omega^T x_i)) + \gamma_{reg}$

prob: $\omega^* = \arg \min_{\omega} \sum_{i=1}^n -y_i \log p_i - (1-y_i) \log(1-p_i) + \gamma_{reg}$

where $p_i = \sigma(\omega^T x_i)$

Proof :

Lets take case 1 :

$$Y = +$$

$$\text{Means } y_I(\text{Geo}) = +1$$

$$y_I(\text{prob}) = +1$$

$$\begin{array}{ll} \text{Case 1: } & y_i : +ve \\ \text{geom: -} & y_i = +1 \\ \text{prob: -} & y_i = +1 \end{array}$$

$$\begin{array}{ll} \text{geom: -} & \log\left(1 + \exp(-w^T x_i)\right) \\ \text{prob: -} & \left(-1 \cdot \log\left(\frac{1}{1 + \exp(-w^T x_i)}\right)\right) \\ & = \cancel{\log\left(1 + \exp(-w^T x_i)\right)} \quad \left[-\log(x) = \log\left(\frac{1}{x}\right) \right] \end{array}$$

After putting all values we get same result for both .

$$\begin{array}{ll} \text{Case 2: } & y_i = -ve \\ & y_i = -1 \xrightarrow{\text{geom}} \\ & y_i = 0 \xrightarrow{\text{prob}} \end{array}$$

$$\begin{array}{ll} \text{geom:} & \log\left(1 + \exp(w^T x_i)\right) \\ \text{prob:} & -1 \cdot \log\left(1 - \frac{1}{1 + \exp(-w^T x_i)}\right) \\ & \left(-1 \cdot \log\left(\frac{\exp(-w^T x_i)}{1 + \exp(-w^T x_i)}\right)\right) \end{array}$$

$$\begin{aligned} -\log(x) \\ = \log\left(\frac{1}{x}\right) \end{aligned}$$

$$\begin{aligned}
 &= \log \left(\frac{1 + \exp(-w^T x_i)}{\exp(-w^T x_i)} \right) \\
 &\quad \text{divide numerator & den by } (\exp(-w^T x_i)) \\
 &= \log \left(1 + \frac{1}{\exp(w^T x_i)} \right) \quad \boxed{\frac{1}{e^{-x}} = e^x} \\
 &= \underline{\log \left(1 + \exp(w^T x_i) \right)} \quad \circlearrowleft
 \end{aligned}$$

Loss minimization interpretation of LR

✓ geometric & probabilistic

loss-minimization

$$\left\{ \begin{array}{l} f(x_i) = w^T x_i \\ w^* = \arg \min_w \sum_{i=1}^n \log \left(1 + \exp(-y_i w^T x_i) \right) \\ z_i = y_i w^T x_i = \underline{y_i \cdot f(x_i)} \end{array} \right.$$

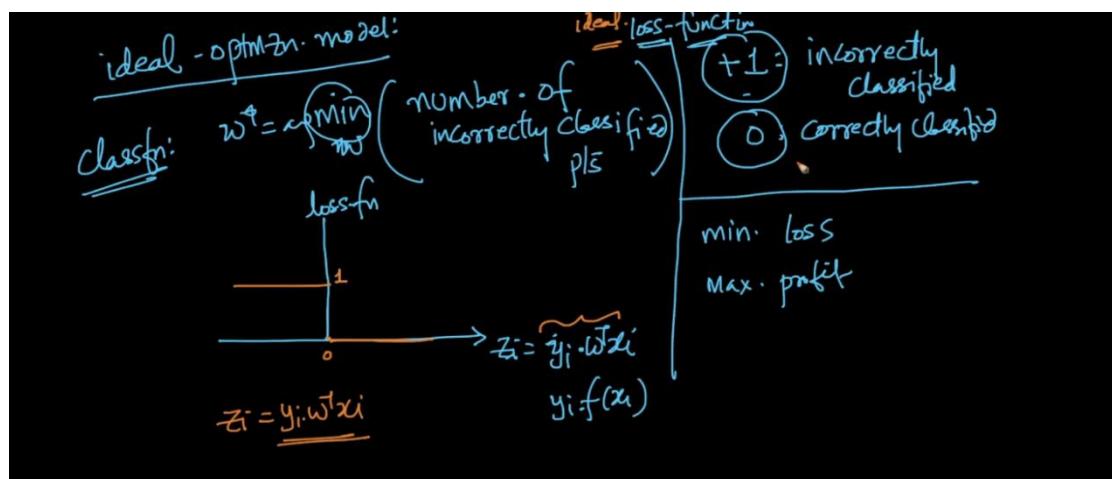
What is loss minimization ?

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 .

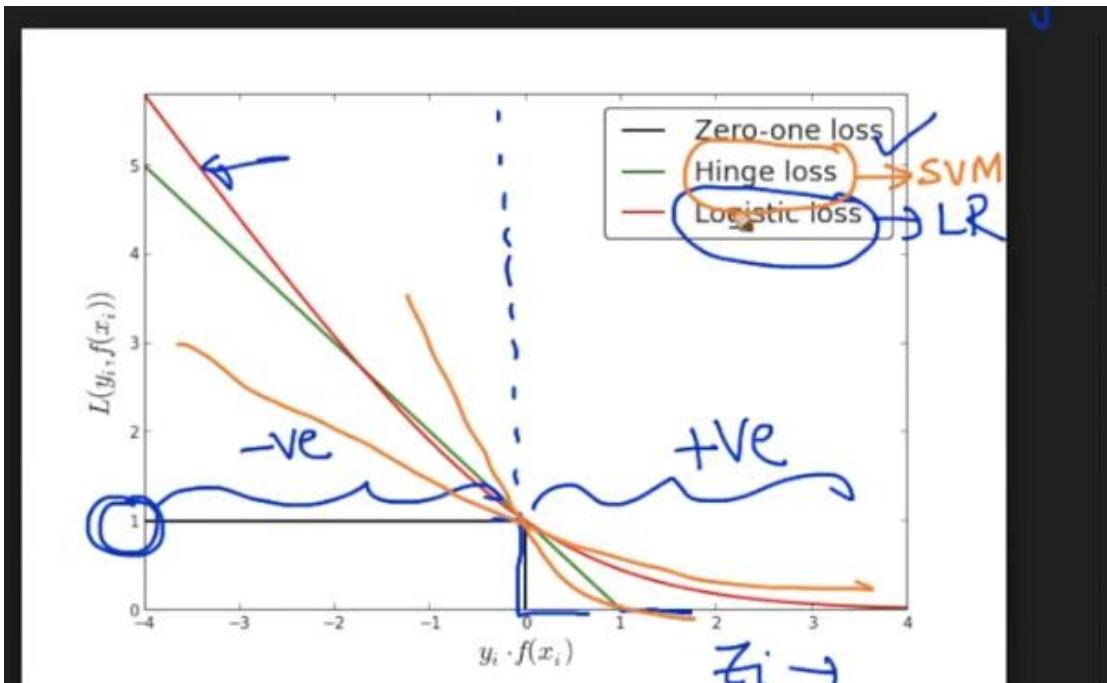
See below image 0-1 loss it is ideal loss function . Here lets say our $z > 0$ means points are correctly classified means no loss hence $z>0$ you see loss also 0 .

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 .

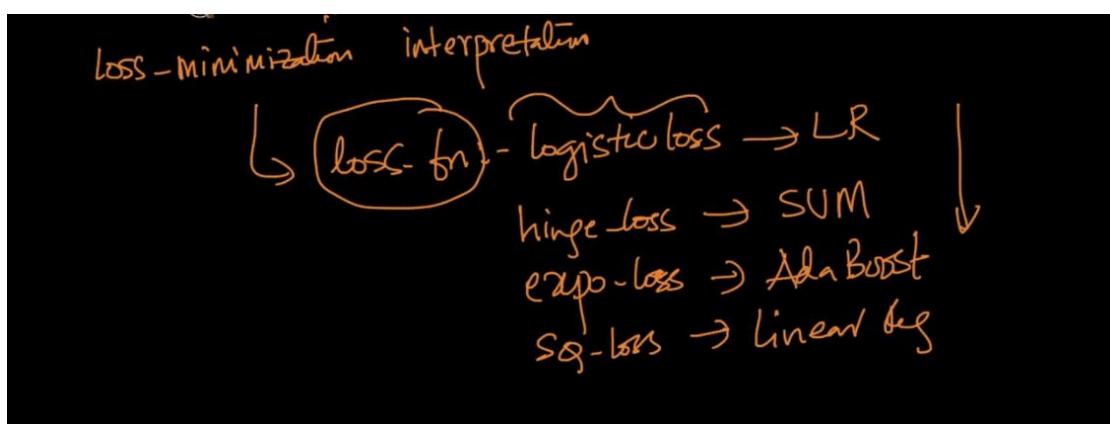


So we approximate this problem by using another loss ..



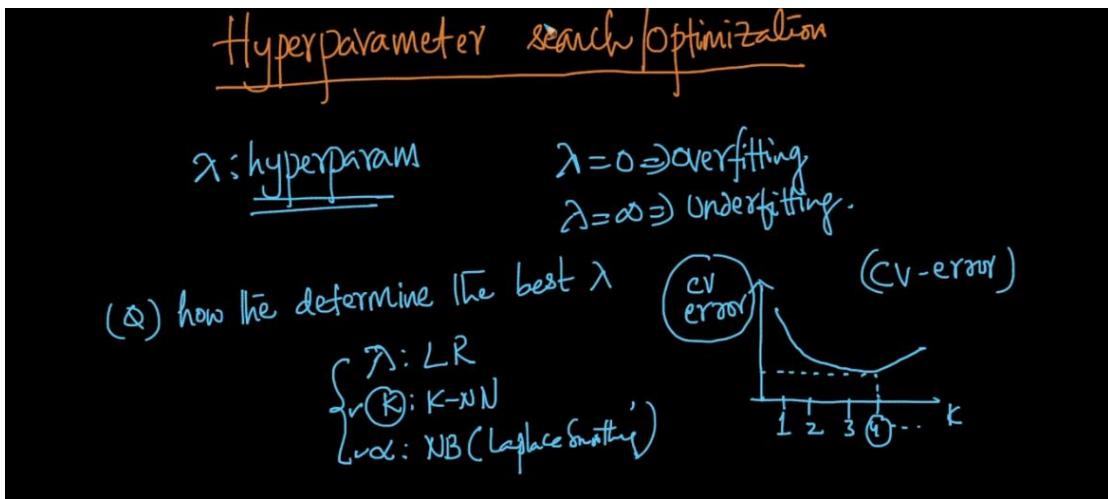
Called as logistic loss . if we use another loss hinge loss then we get SVM .

Just by changing loss we can learn diff model .

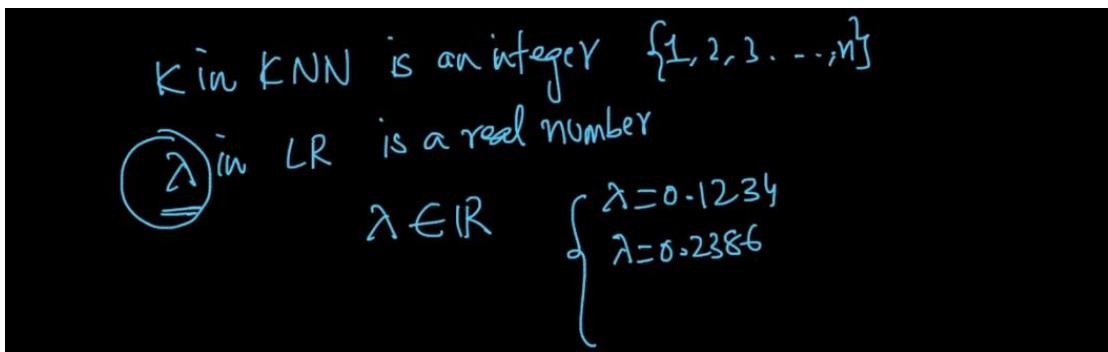


So how to chose correct lamda ?:

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

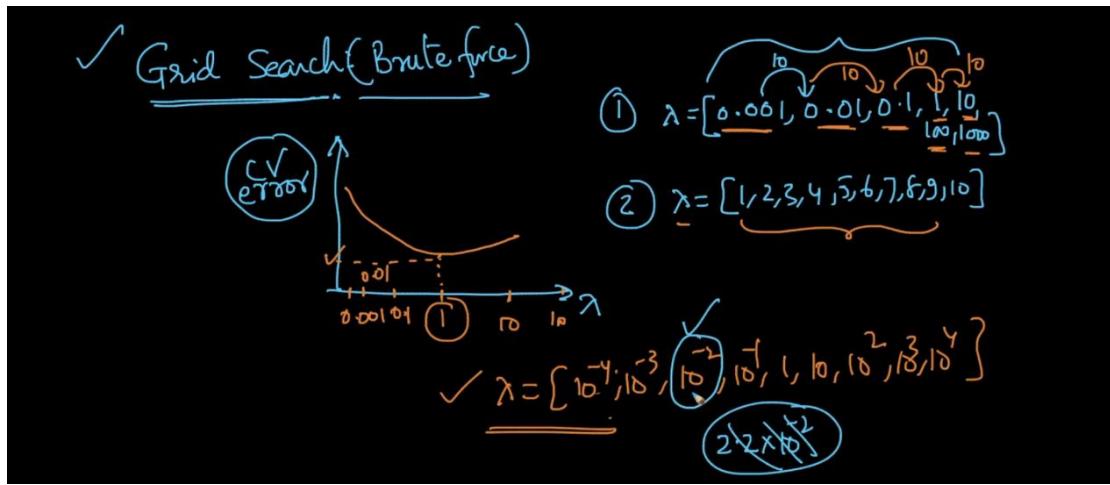


But for KNN values are like 1,2,3,..., n but for lamda values are real it can be anything ..
 So we using 2 methods :



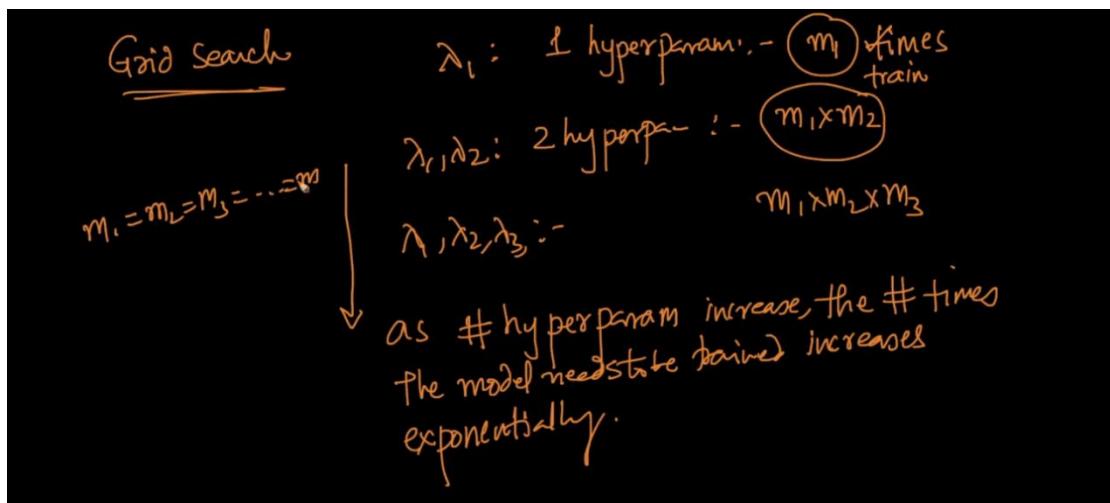
1. Grid search :

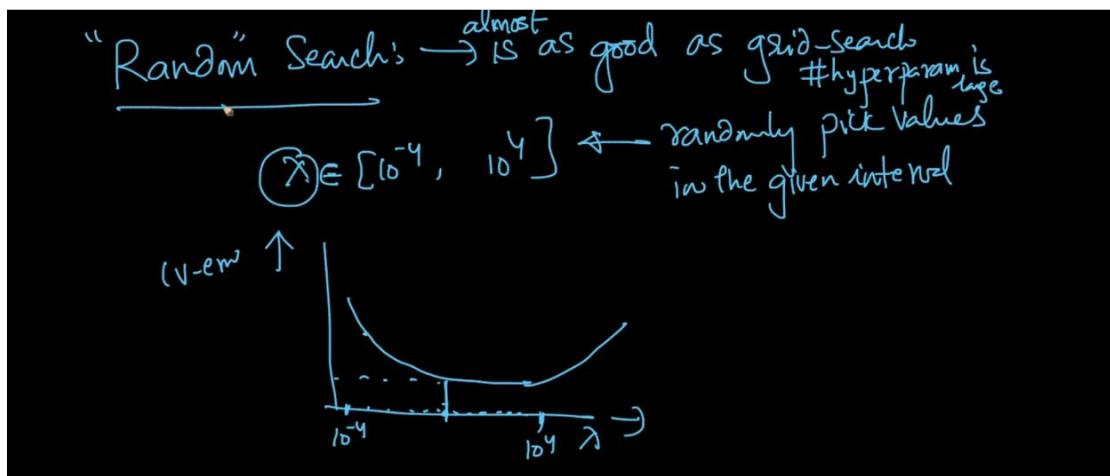
So here we give some range see in image and we run CV error graph for which error is less we pick that value as Lamda .



But here we have only one hyper parameter . lets say we took m_1 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 $m_1 * m_2$ 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 .

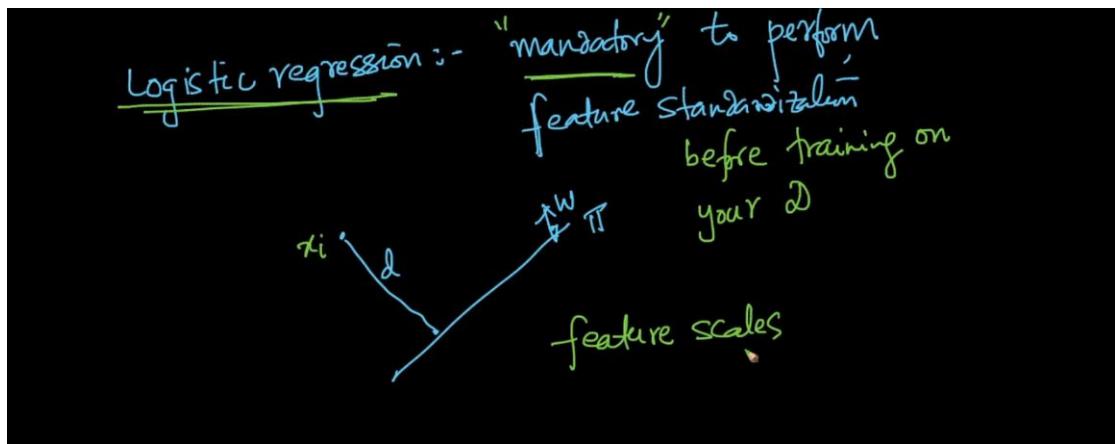
This will work too better than grid .

Column / feature standardization

$$z_{ij}' = \frac{z_{ij} - \mu_j}{\sigma_j}$$

\checkmark k-NN; - distances
✓ Standardize our feature

$\left\{ \begin{array}{l} \text{centering} \\ \text{scaling} \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} z_{ij}' = z_{ij} - \mu_j \\ \sigma_j \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} \text{standardization} \\ \text{mean-centering} \\ \text{scaling} \end{array} \right\}$



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 our data .

Feature Importance :

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

Feature importance & Model Interpretability

$f_1, f_2, \dots, f_j, \dots, f_d$

$w \rightarrow w_1, w_2, \dots, w_j, \dots, w_d$

$LR : GNB + \text{Bernoulli}$

assume: if all features are independent (Naive Bayes)

feature-imp: $w_j's$

$(K-NN)$:- feature-imp \rightarrow forward feature Sel(n.)
 (NB) :- $P(X_i | Y=+1)$ \rightarrow features which are important
 (LR) :- w_j 's \rightarrow to do

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.

$|w_j|$ = absolute value of weight corresponding to f_j

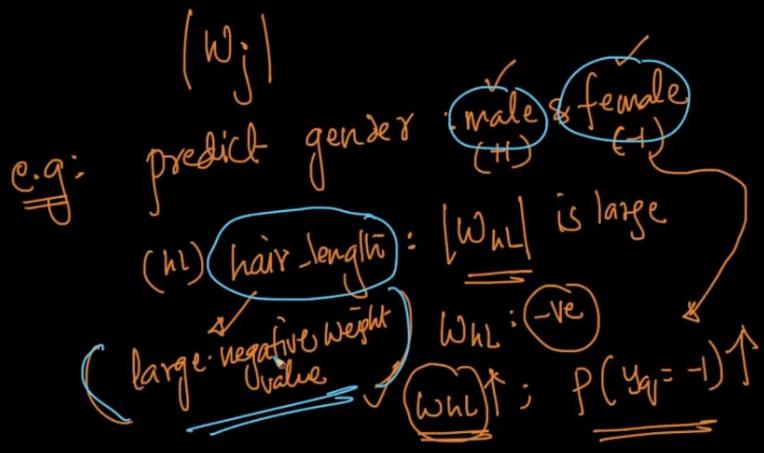
$|w_j| \uparrow ; (w^T x_q) \uparrow$

Case 1: $w_j = +ve$ & large; $\sum_{j=1}^d w_j \cdot x_{qj} \uparrow$ $\leftarrow w^T x_q$

$\hookrightarrow P(Y_q = +1) \uparrow$

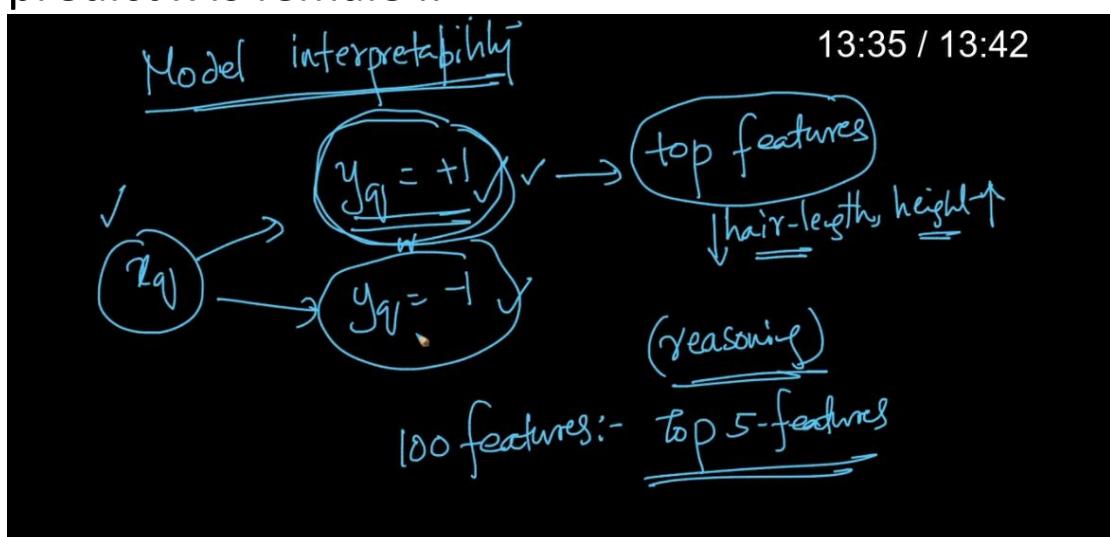
Case 2: w_j : -ve & large' :- $\sum_{j=1}^d w_j x_{qj} \uparrow$ $\leftarrow P(Y_q = -1) \uparrow$

determine the important features in LR



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 ..



"Collinearity" (or) Multicollinearity

f.I :- features are indep; $|w_j|$ as f.I values

Collinearity:

\tilde{f}_i, \tilde{f}_j

s.t if, $\tilde{f}_i = \alpha \tilde{f}_j + \beta$

then \tilde{f}_i & \tilde{f}_j are collinear

Lets say we have 2 feature $f_1 = \text{height}$ and $f_2 = \text{weight}$.

If we get f_1 (h) by doing some multiplication with f_2 then we can say f_1 and f_2 are co linear.

M co linear also same like if we get f_1 by doing some multiplication with $f_2, f_3, f_4, \dots, f_n$ then we can say f_1 and others are m co linear ..

Multicollinearity: if f_1, f_2, f_3 & f_4 s.t

$$f_1 = \alpha_1 + \alpha_2 f_2 + \alpha_3 f_3 + \alpha_4 f_4$$

then f_1, f_2, f_3 & f_4 are said to be multicollinear

(Q) Why does $|w_j|$ not be useful as f.I if features are collinear

$$\mathcal{D} = \langle x_i, y_i \rangle_{i=1}^n$$

$$\omega^+ = \langle 1, 2, 3 \rangle ; \quad x_q = \langle x_{q_1}, x_{q_2}, x_{q_3} \rangle$$

$$f_1 f_2 f_3$$

$$\omega^T x_q = x_{q_1} + 2x_{q_2} + 3x_{q_3}$$

if $f_2 = 1.5f_1 \Rightarrow f_1 \text{ and } f_2 \text{ are collinear}$

$$\omega^T x_q = \underline{x_{q_1} + 3x_{q_1}} + 3x_{q_3} = 4x_{q_1} + 3x_{q_3} = \langle 4, 0, 3 \rangle$$

See lets say we have f_1, f_2, f_3 we did multiplication .

Then we put $f_2 = 1.5f_1$ just assume then we get :

$$2x_{q_2} = 2 * 1.5x_{q_1} = 3x_{q_1}$$

$$x_{q_1} + 3x_{q_1} + 3x_{q_3}$$

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

We get this difference just because of relation between f_1 and f_2 .

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 $f_2 = 0$ means it very worst feature and original matrix say it 2nd importance feature .

That's why we can't use co linear feature to decide feature importance .

$|w_j|$ as f.I

determine if features are multicollinear.

\rightarrow perturbation technique:

$x_{ij} + \epsilon$

small noise $N(0, D \cdot \frac{1}{\sigma})$

before perturbation: $w = \langle w_1, w_2, \dots, w_j, \dots, w_d \rangle$

after perturbation: $\tilde{w} = \langle \tilde{w}_1, \tilde{w}_2, \dots, \tilde{w}_j, \dots, \tilde{w}_d \rangle$

{ if $w_i \neq \tilde{w}_i$ differ significantly
then your features are collinear
 \downarrow
 $|w_j|$'s as f.I

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 them to find feature importance .

Small diff is OK .

Train & Run time Space & time complexity

$n = \# \text{ pts in } D_{\text{train}}$ $d = \text{dim}$

$O(nd)$ {

- Train LR:- $\hat{w}^t = \underset{w}{\operatorname{arg\min}} (\text{logistic loss} + \text{reg})$ $\geq \text{next-chapter optimztn}$
- Run Time (LR) $w^t = \langle w_1, w_2, \dots, w_d \rangle$

{

- Space: $O(d)$
- Time: $O(d)$

$\sum_{j=1}^d w_j x_{qj}$

$\begin{cases} w^T x_{qj} > 0 \rightarrow \text{+ve} \\ w^T x_{qj} < 0 \rightarrow \text{-ve} \end{cases}$

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

In L1 reg as we increase lamb sparsity 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 :

\checkmark Bias term

if d is large $d \approx 1000$ $\rightarrow \uparrow$; unaff

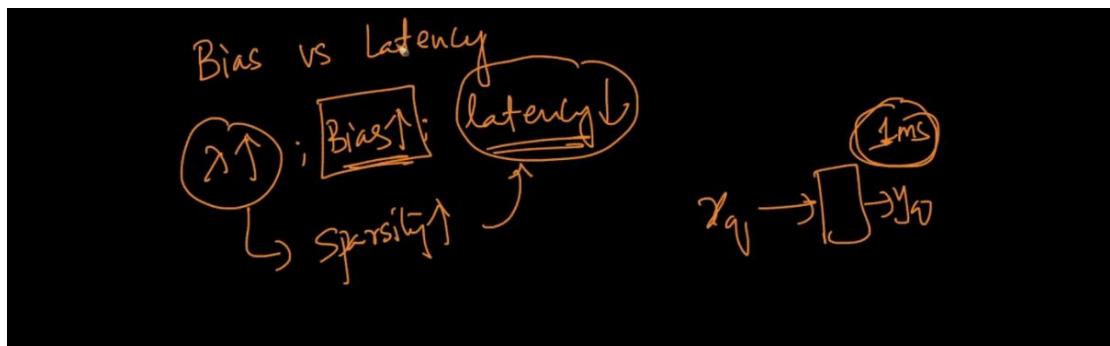
$w^T x_q$: - mult & additive 1000 mult & additions

\rightarrow L1 reg: - Sparsity (w_j 's corresponding to less important features = 0)

w^t : $\begin{matrix} 0 & 0 & 0 & 1 & 0 & \dots & 1 & 0 \end{matrix}$ \downarrow : reasonably

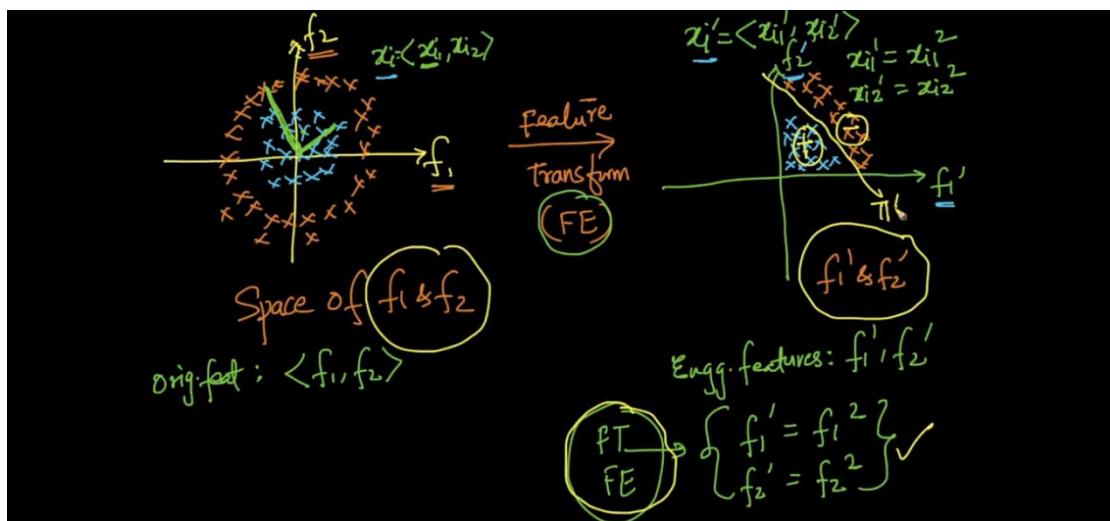
\uparrow : Sparsity \uparrow more of $w_j = 0$ $\{50\}$ mult & addns 1 ms

\uparrow : latency req



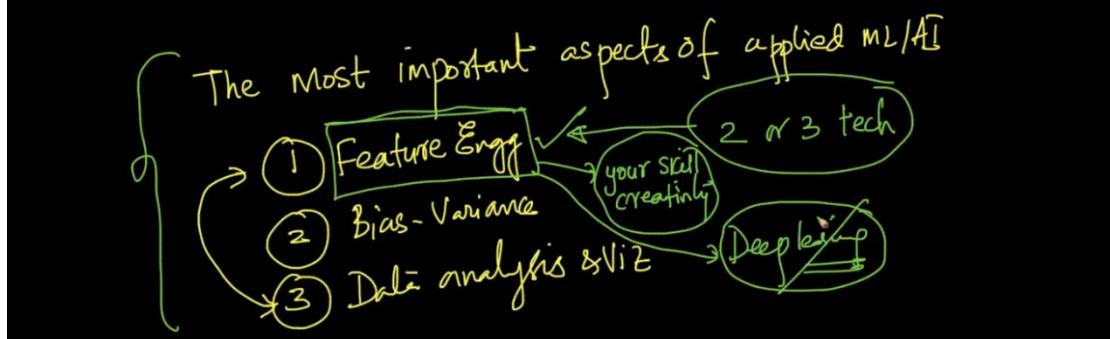
If manager say if your model take grater than 1 ms to run then I cant accept then it is better to go with high lamb so we get high bias with low latency . and it will be good instead of no model .

Feature Transformation :

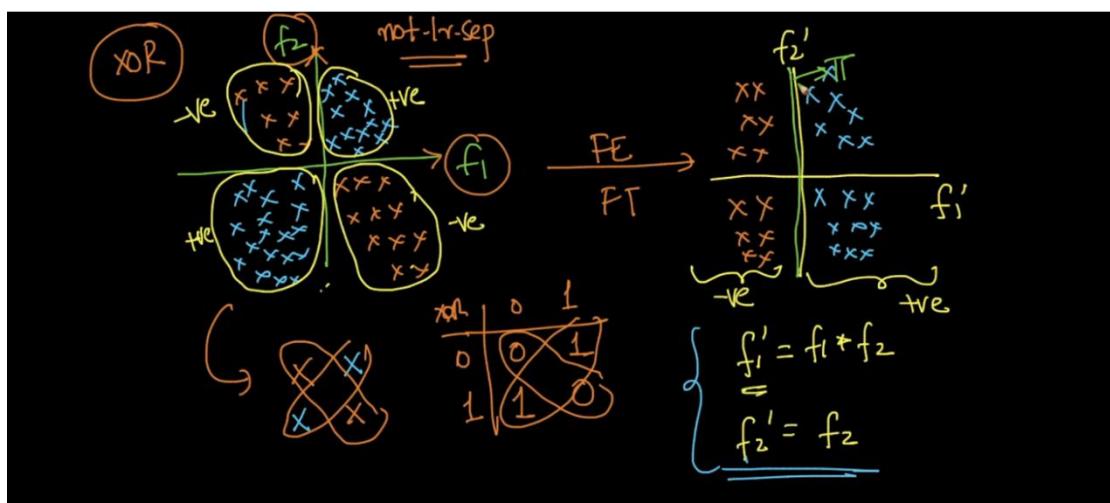


See above data is non linear we apply feature engineering to transform non linear to linear ..

(Q) how did you know which transform to apply?

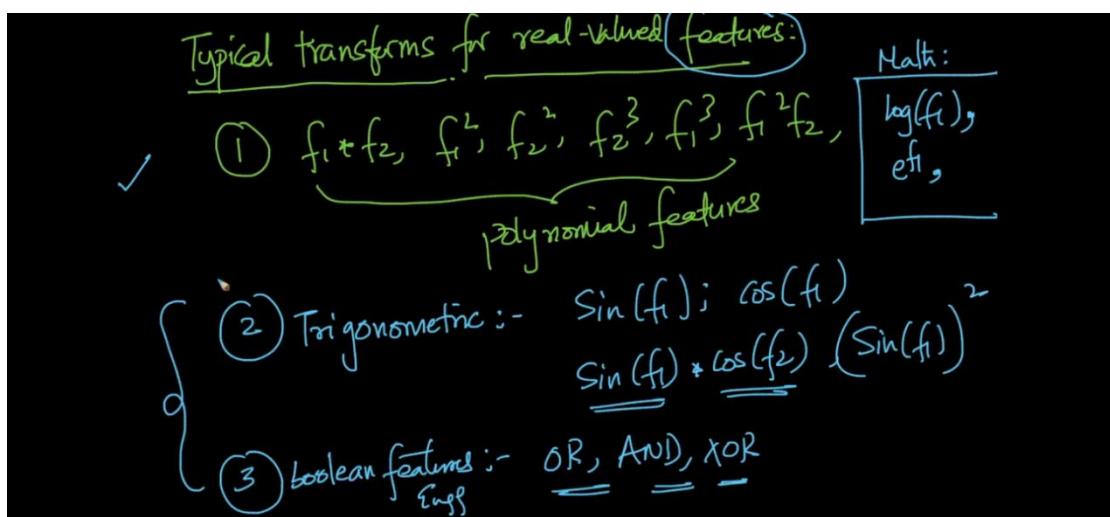
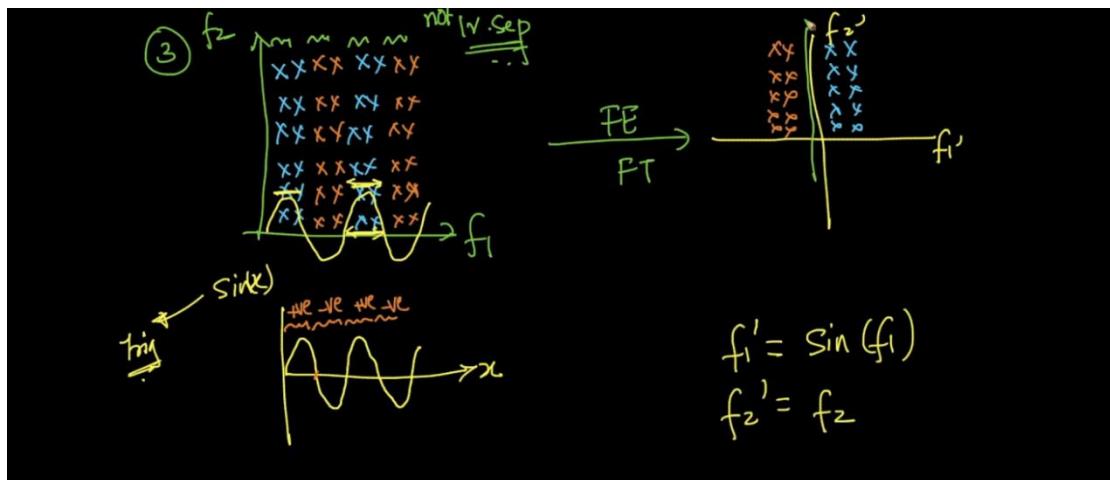


See below 2nd example of non linear to linear .



Here data is XOR we make $f_1 = f_1 * f_2$ so we get all positive points on one side on plane .

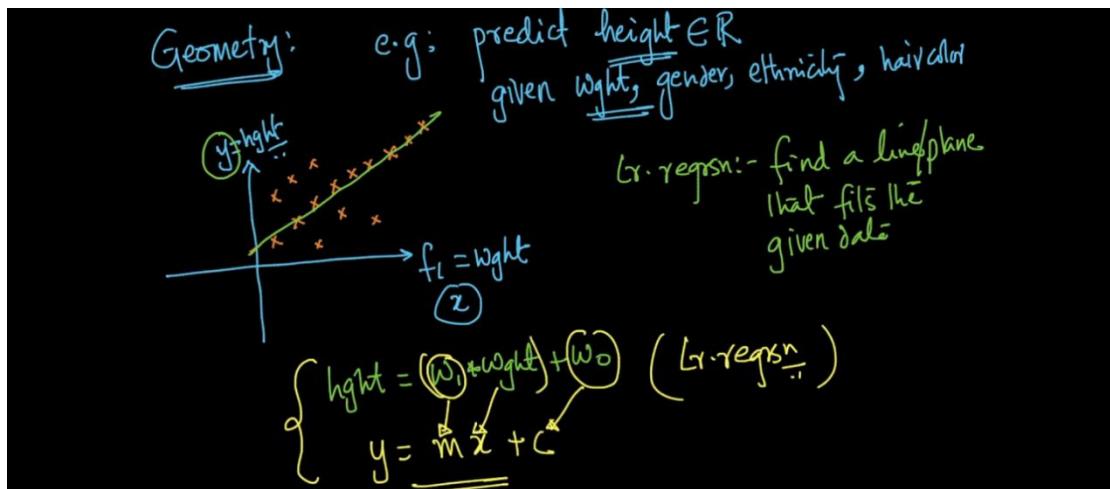
See below 3rd example here we apply sin because see data is in sin shape . so we get all positive and negative separate .



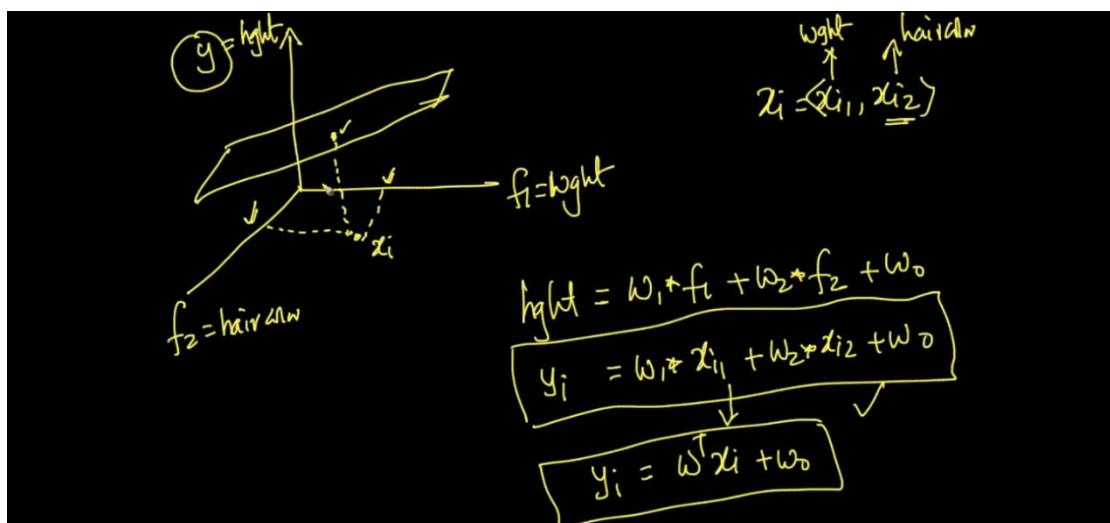
We can lots of diff technique for feature transform above are same famous methods.

Linear Regression :

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



So we want to find best w_1 and w_0 for 2d data .



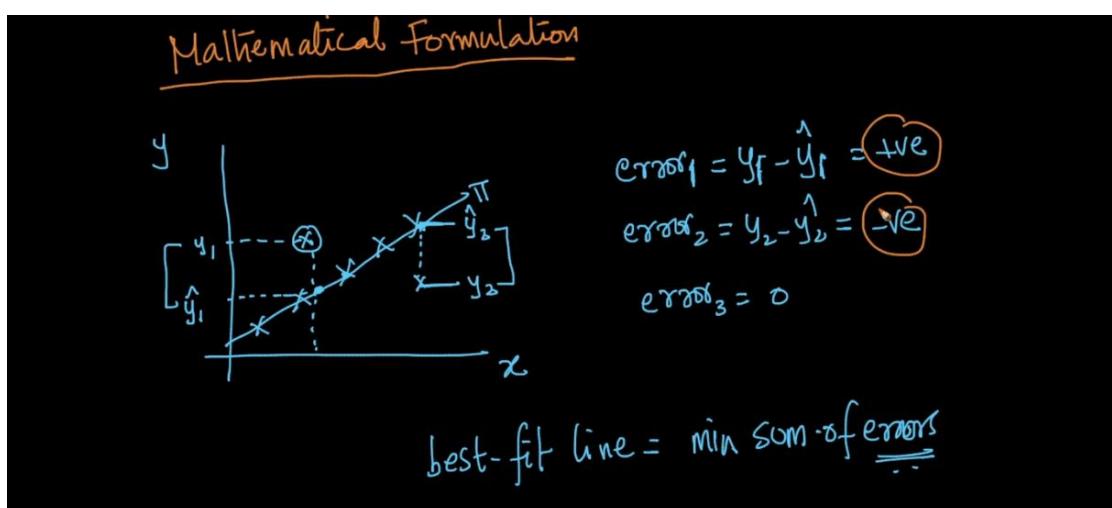
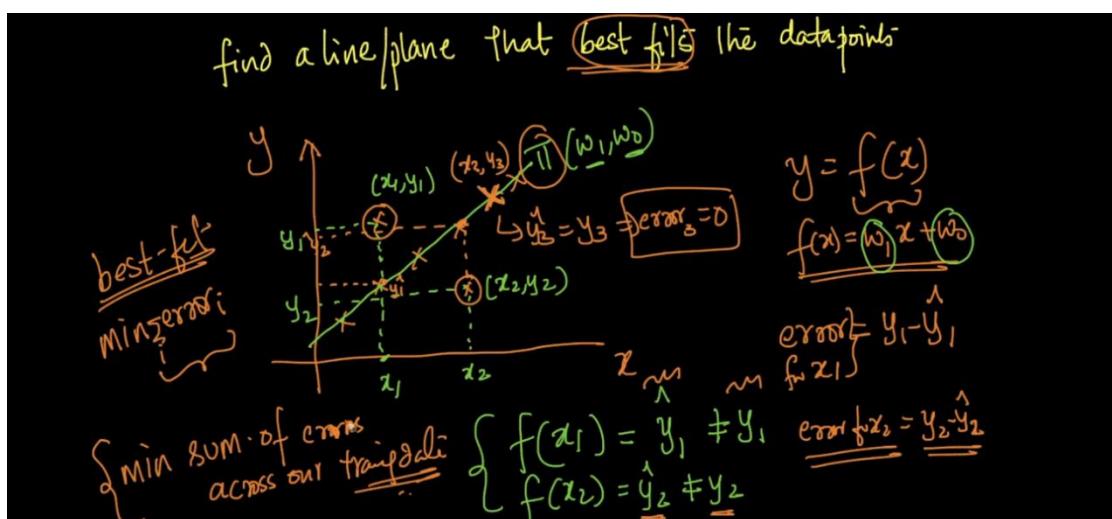
For 3 dimension we just extend formula .

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 $= Y^{\wedge}$

Original output for C = Y

So error = $Y - Y^A$



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

Lr. regsn: $\rightarrow \underline{\text{OLS}}, \underline{\text{LLS}}$

$$\left\{ \begin{array}{l} (\hat{w}, \hat{w}_0) = \underset{w, w_0}{\arg \min} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ \text{vector} \quad \text{scalar} \\ \hat{y}_i = f(x_i) = w^T x_i + w_0 \end{array} \right.$$

optimzn-pds $(w^*, w_0^*) = \underset{w, w_0}{\arg \min} \sum_{i=1}^n \left\{ y_i - \underbrace{(w^T x_i + w_0)}_{\text{sq-loss}} \right\}^2$

Regularize is same like Logistic only diff is square loss.

regularization:

$$(w^*, w_0^*) = \underset{w, w_0}{\arg \min} \sum_{i=1}^n \left\{ y_i - (w^T x_i + w_0) \right\}^2 + \lambda \|w\|_2^2$$

Logistic-reg^n: $L_2\text{-reg}$ or $L_1\text{-reg}$ or elastic net

prob. interpretation:

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

Cases: (Lr-regn)

Imbalanced data:- upsampling, down sampling
✓ feat. imp & inter. predictability: - not multicollinear
log-reg: $\rightarrow \text{feat} \rightarrow w_j$

✗ no-multi class:-
feature Engg. / feature transforms \rightarrow log. regn

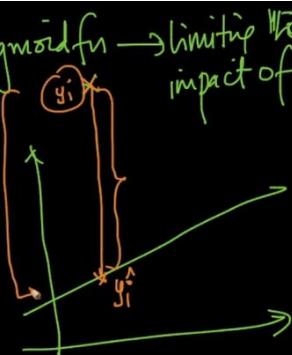
For out l. we used sigmoid for logistic regression
here what we will do ?

Outliers: Log-regn:- Sigmoid fn \rightarrow limiting the impact of outliers

Lr-regn:-

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2$$

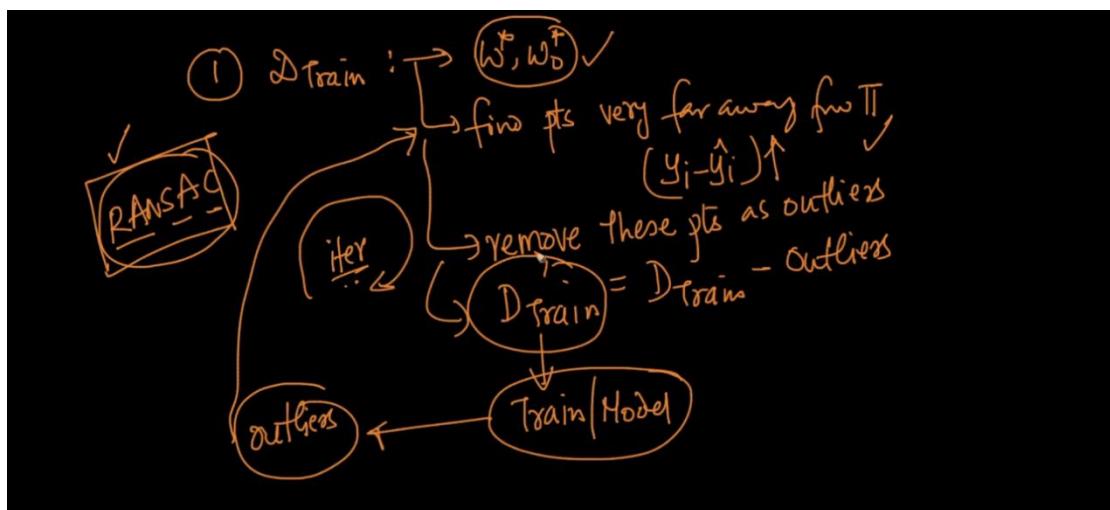
SQ-loss:- outlier



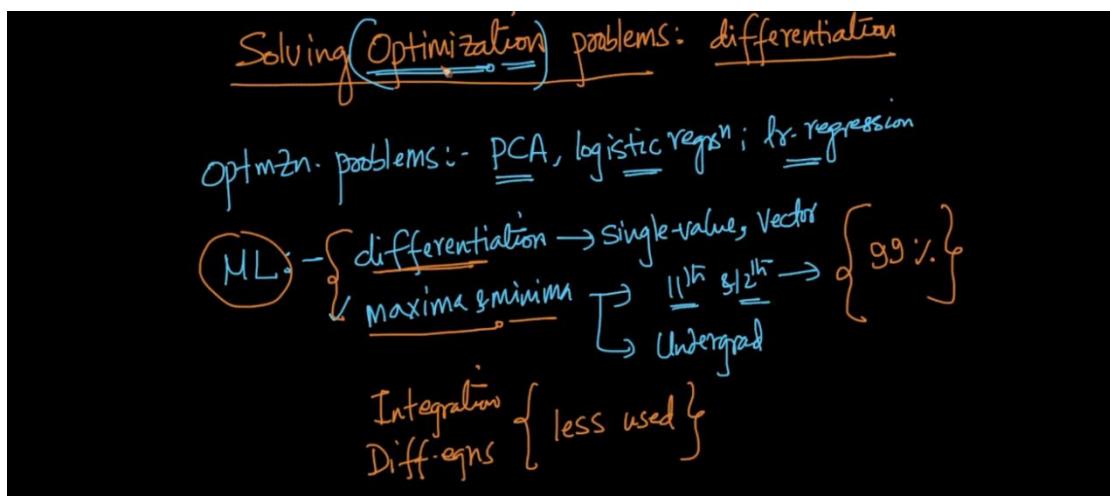
As because of square loss it will be very high for out l. in case of linear regression ..

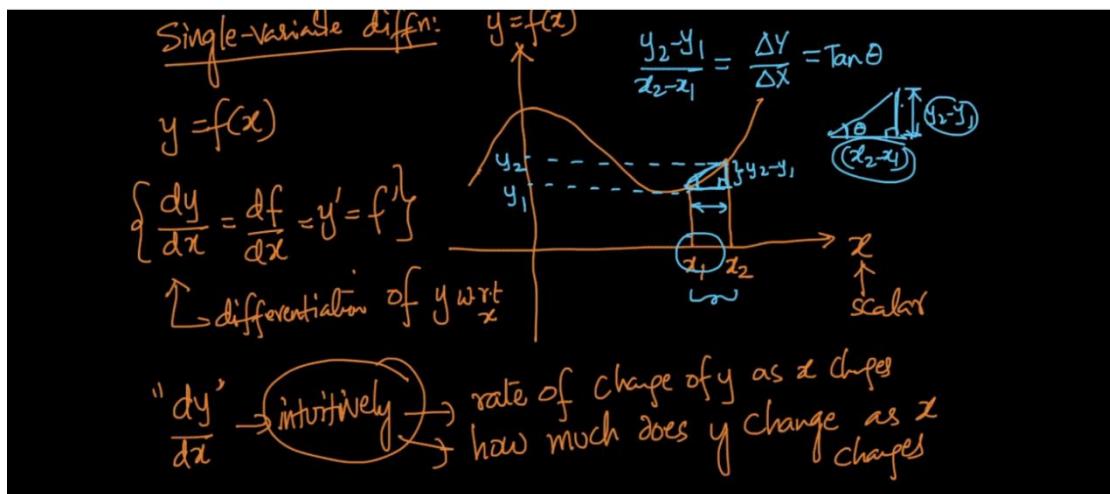
So we will just take train data cal w and w0 so we get plane or line then we cal distance if $(y - y^)$ is

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



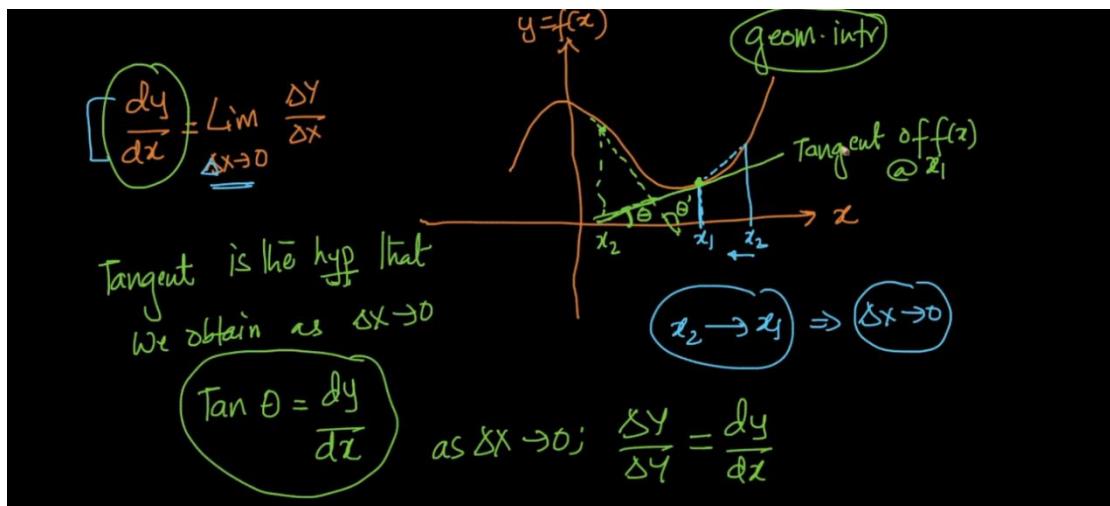
Optimization :





Diff means how much does y change if change in x .

Geometrically we just try to find slope of tangent.

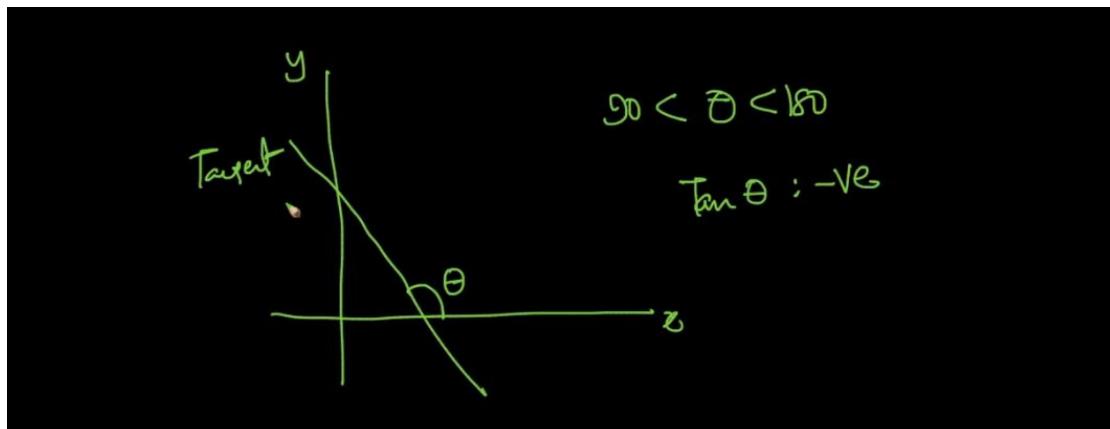
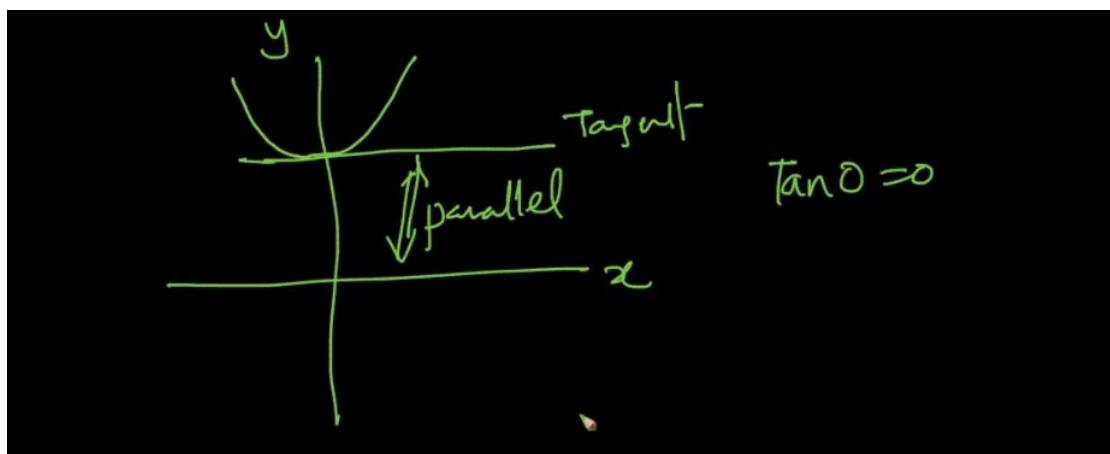
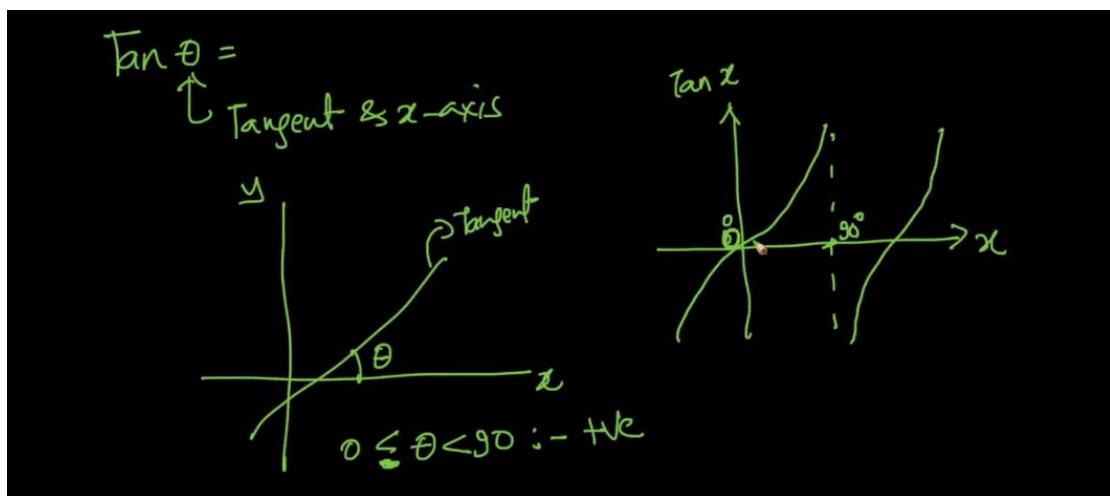


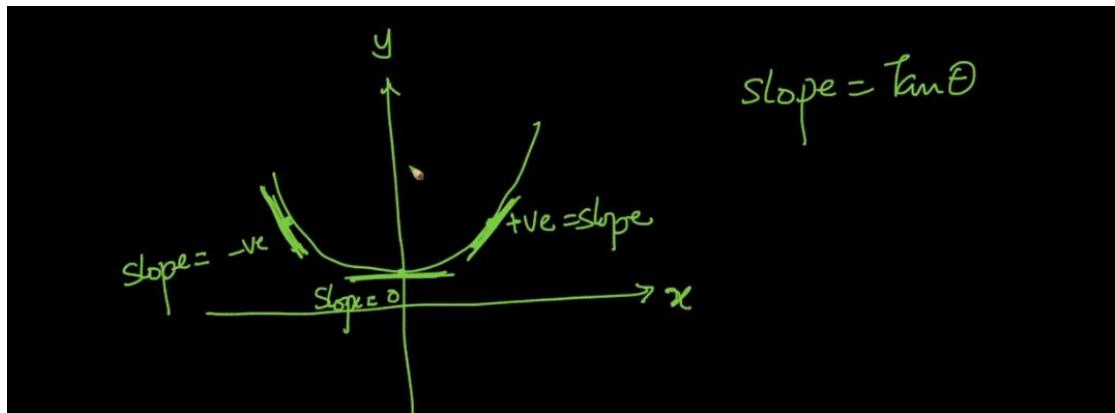
$$\Delta x = x_1 - x_2$$

$$\Delta y = y_1 - y_2$$

Limit $x \rightarrow 0$ means if difference between x_1 and x_2 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 .





basic :-

$$\frac{d}{dx}(x^n) = nx^{n-1} ; \quad \frac{d}{dx}(x^2) = 2x$$

$$\frac{d}{dx}(c) = 0 \quad \frac{d}{dx}(3x) = 3$$

$$\frac{d}{dx}(cx^n) = cnx^{n-1}$$

$$\frac{d}{dx} \log(x) = \frac{1}{x}$$

Chain-rule :-

Jeff. dean

$$\frac{d}{dx} f(g(x)) = \frac{df}{dg} \cdot \frac{dg}{dx}$$

$$f(g(x)) = (a-bx)^2$$

$$g(x) = (a-bx) = z$$

$$f(x) = x^2$$

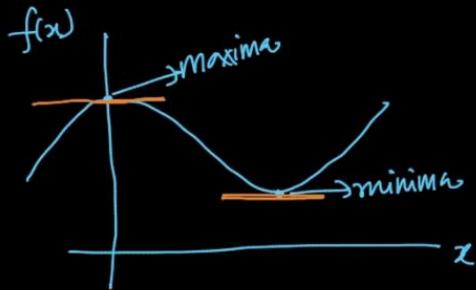
$$\frac{df(g(x))}{dx} = -2b(a-bx)$$

$$\frac{df}{dg} = \frac{d}{dx}(a-bx) = \frac{d}{dx}(a) - \frac{d}{dx}(bx) = -b$$

Slope is nothing but $\tan(\text{tangent line and } x\text{ axis angle})$..

Maxima & Minima:

at minima &
maxima;
(slope = 0)



For minim and Maxim slope = 0 .

$$f(x) = x^2 - 3x + 2$$

Maxima and/or minima

$$\text{Eqn: } \boxed{\frac{df}{dx} = 0} \Rightarrow \text{slope} = 0 \quad @ x = \frac{3}{2}; \text{slope} = 0$$

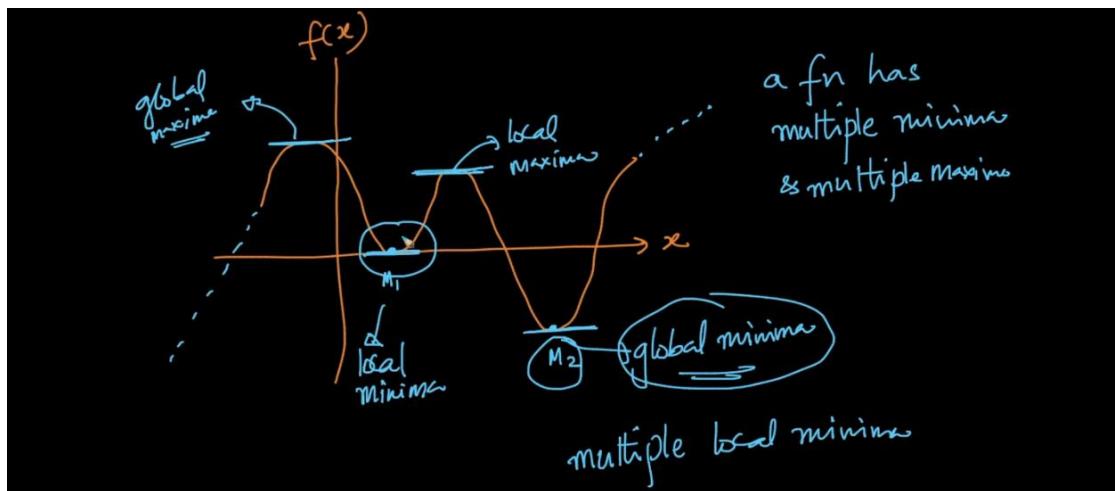
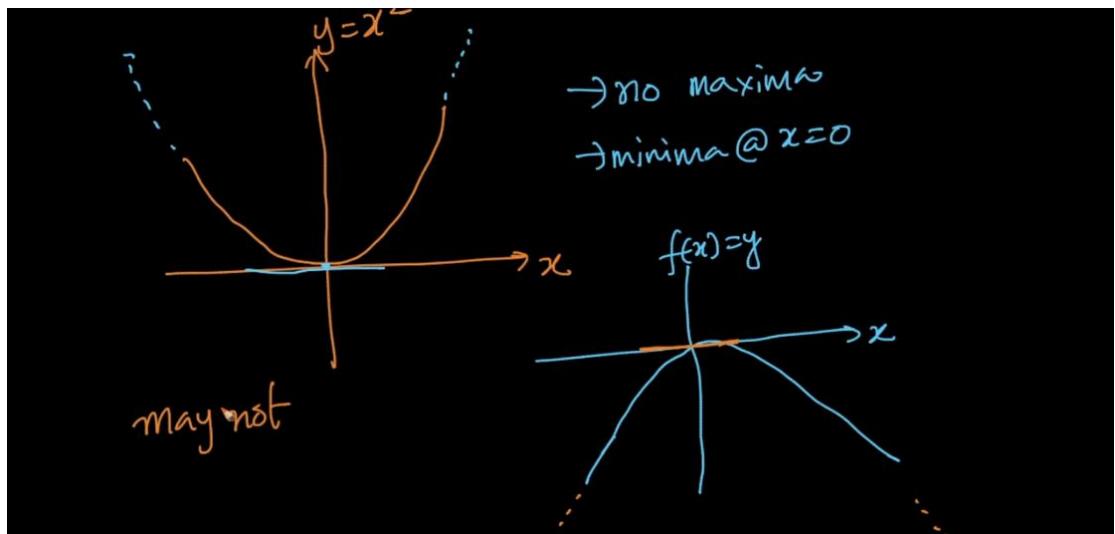
$$\frac{df}{dx} = 2x - 3 = 0 \Rightarrow \boxed{x = \frac{3}{2}} \quad \left| \begin{array}{l} (Q) f(1.5) = -0.25 \\ f(1) = 0 \\ f(1.5) < f(1) \\ \Rightarrow 1.5 \text{ cannot be maxima} \end{array} \right.$$

minima @ x = 1.5

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.



$$\min_{\alpha} \text{log}(1 + \exp(\alpha x)) \rightarrow \text{logistic loss}$$

$$\frac{df}{dx} = \frac{\alpha \exp(\alpha x)}{1 + \exp(\alpha x)} = 0 \quad \begin{cases} \text{Solving this is} \\ \text{not trivial} \\ \text{u.v. hard} \end{cases}$$

✓ $\frac{df}{dx} = 0$ Gradient descent →

Vector differentiation : Grad

x : scalar $f(x)$

\underline{x} : vector → high-dim-space

$y = \underline{a}^T \underline{x}$; $\underline{x} = \langle x_1, x_2, \dots, x_d \rangle$
 $a = \langle a_1, a_2, \dots, a_d \rangle \rightarrow \text{constants}$

$$y = \sum_{i=1}^d a_i x_i$$

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 x_1, x_2, \dots, x_d

$$\frac{df}{dx} = \nabla_x f$$

grad (or) Del

\uparrow Vector

$$\nabla_x f = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \\ \vdots \\ \frac{df}{dx_d} \end{bmatrix}$$

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

$$\frac{df}{dx_i} = \left(\frac{\partial f}{\partial x_i} \right) \rightarrow \text{partial diffn}$$

$$f(x) = y = a^T x = \sum_{i=1}^d a_i x_i = a_1 x_1 + a_2 x_2 + \dots + a_n x_n$$

$$\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix} = a$$

$$\frac{\partial f}{\partial x_1} = a_1$$

$$\frac{\partial f}{\partial x_2} = a_2$$

$$\nabla_x (a^T x) = a \quad \boxed{a}$$

$$\boxed{\frac{d(a^T x)}{dx} = a}$$

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 .

$$L(\omega) = \sum_{i=1}^n \log \left(\frac{1}{1 + \exp(-y_i \omega^\top x_i)} \right) + \lambda \frac{\omega^\top \omega}{2}$$

$\langle x_i, y_i \rangle \rightarrow \text{constants} \rightarrow D_{\text{train}}$

$$\nabla_{\omega} L = \frac{(y_i x_i) \exp(-y_i \omega^\top x_i)}{1 + \exp(-y_i \omega^\top x_i)} + 2\lambda \omega = 0$$

(Gradient descent)

Gradient Descent :

Gradient descent algorithm:

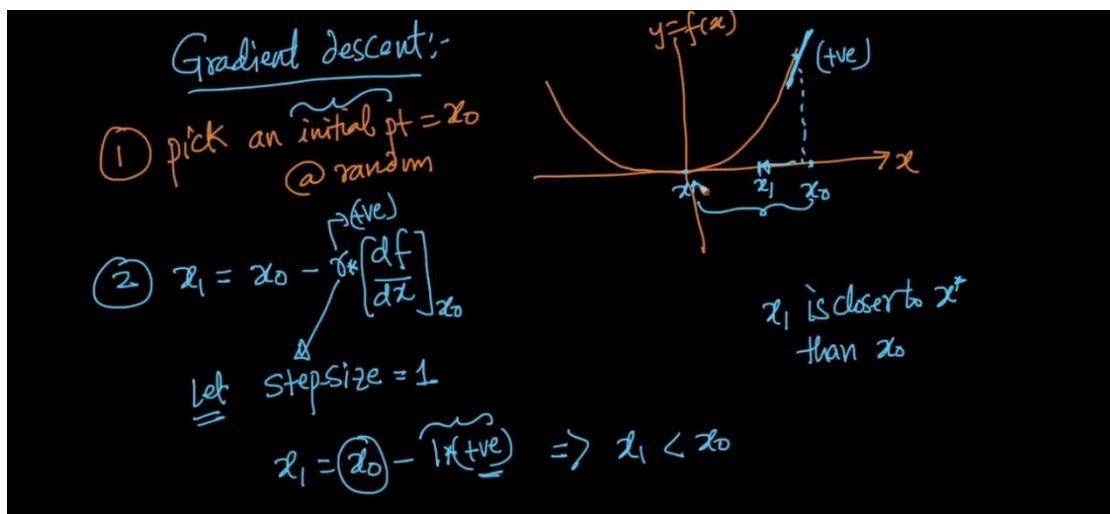
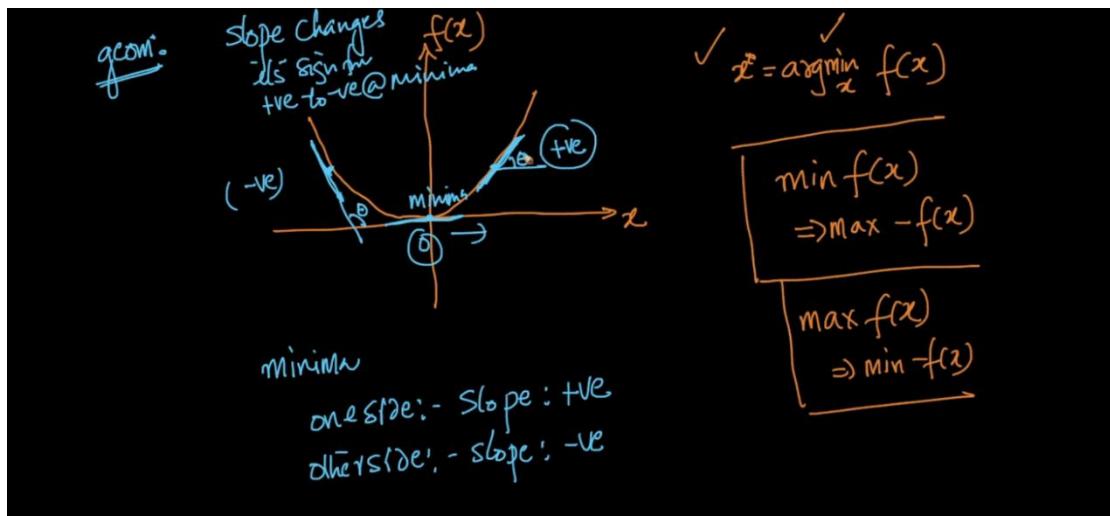
maxima & minima:

$$\boxed{\frac{df}{dx} = 0} ; \boxed{\nabla_x f = 0} \quad ; \quad \boxed{x^* = \arg \min_x f(x)}$$

→ iterative algorithm :-

- $x_0 \leftarrow$ first guess of x^*
- $x_1 \leftarrow$ iteration 1
- $x_2 \leftarrow$ iteration 2
- \vdots
- $x^* \leftarrow x_k \leftarrow$ iter k

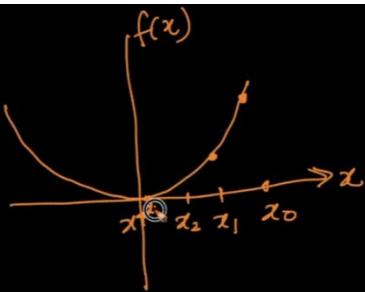
We take any random value as x_0 then our algorithm find itself best min .



Similarly we will find x_2, x_3, \dots, x_n by running in loop. We are taking derivative means we will get slope . here we took random point x_0 and we assume that it is max point .

Now as per our formula X_0 is max so are subtracting new number from this max number so next number will less than x_0 .

$$③ x_2 = x_1 - \gamma \left[\frac{df}{dx} \right]_{x_1}$$



iteration :-

$$\checkmark x_{i+1} = x_i - \gamma \left[\frac{df}{dx} \right]_{x_i}$$

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

$x_0, x_1, x_2, x_3, \dots, x_k, x_{k+1}$

$$\checkmark x_k = x_{k-1} - \gamma \left[\frac{df}{dx} \right]_{x_{k-1}}$$

if $(x_{k+1} - x_k)$ is v-small
then terminate @ $\boxed{x^* = x_k}$

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 .

remedy for oscillation:-
 change γ with each iteration
 → one tech:- reduce γ with each iteration
 (γ = function of iteration number)
 $\gamma = h(i)$ iteration s.t $i \uparrow; \gamma \downarrow$

Gradient Descent For Linear Regression :

Gradient descent for Linear Regression:-

$$w^* = \arg \min_w \sum_{i=1}^n (y_i - w^T x_i)^2$$

↑
 not using w_0
 ↗ no-reg

$$L(w) = \sum_{i=1}^n (y_i - w^T x_i)^2 \quad \langle x_i, y_i \rangle \rightarrow D_{train}$$

$$\nabla_w L = \sum_{i=1}^n \left\{ 2(y_i - w^T x_i)(-x_i) \right\}$$

① pick a random vector $w_0 = \begin{pmatrix} \dots \\ \dots \end{pmatrix}$

$$w_1 = w_0 - \gamma * \sum_{i=1}^n (-2x_i)(y_i - w_0^T x_i)$$

$$w_2 = w_1 - \gamma * \sum_{i=1}^n (-2x_i)(y_i - w_1^T x_i)$$

In this way we apply GD to Linear regression and we find perfect w . first we take w_0 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 .

✓ Stochastic Gradient descent (SGD) → the most imp opt

Linear regression:-

$$\text{GD: } w_{j+1} = w_j - \gamma * \sum_{i=1}^n (-2x_i)(y_i - w_j^T x_i)$$

when n is large, takes a lot of time

SGD: $w_{j+1} = w_j - \gamma * \sum_{i=1}^K (-2x_i)(y_i - w_j^T x_i)$

$1 \leq K \leq n$; pick a random set of k -pls

DL: Adam, Adagrad

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 .

GD: - $(n=1M)$ $\boxed{100}$ iterations to converge x^*

SGD: - $K=1000$; if ≥ 100 iterations x^*
 (500)

SGD: $K=10$; $\boxed{1000}$; x^*
 $K = \# \text{ of random pls that you @ each iter for update}$

why does L_1 -reg. create Sparsity:

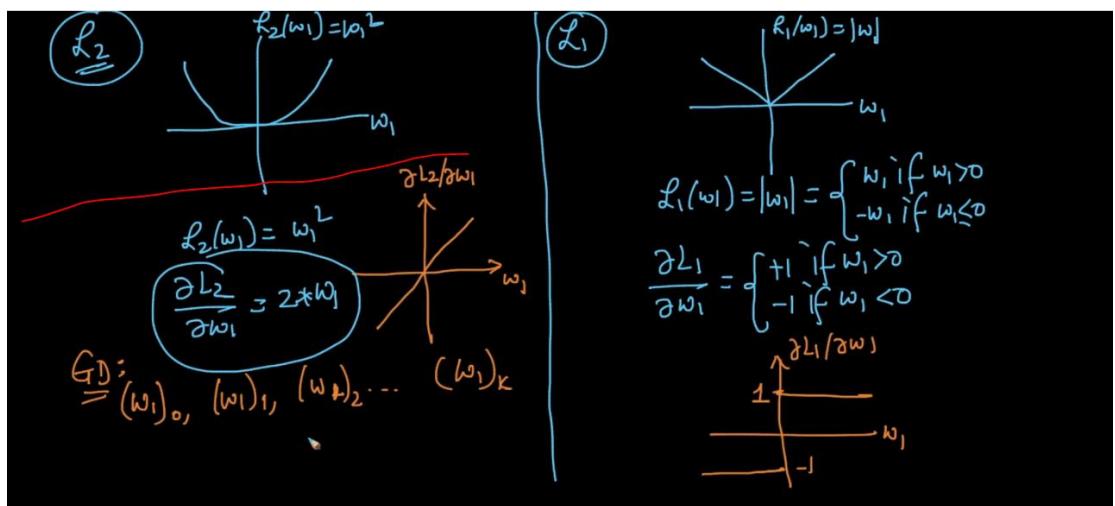
Logistic regn. :- L_1 reg creates Sparsity in w
as compared to L_2 reg
 $w = \langle w_1, w_2, \dots, w_d \rangle$

See formulation of L2 and L1 :

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

(L_2) $w = (w_1, w_2, \dots, w_d)$ $\min_w (\text{loss} + \lambda \ w\ _2^2)$ $\min_{w_1, w_2, \dots, w_d} (w_1^2 + w_2^2 + \dots + w_d^2)$ $\min_{w_1} w_1^2 = L_2(w_1)$	(L_1) $\min_w (\text{loss} + \lambda \ w\ _1)$ $\min_{w_1, w_2, \dots, w_d} (w_1 + w_2 + \dots + w_d)$ $\min_{w_1} w_1 \rightarrow L_1(w_1)$
---	---

Just for simplicity take w_1 for both L_1 and L_2 and find derivative .



See w_1 for L_1 is just absolute value . absolute means $w_1 = w_1$ if $w_1 > 0$
 $w_1 = -w_1$ if $w_1 < 0$

Now we will find derivative of it .

See $L_2 w_1$ and $L_1 w_1$ we have very small difference in equation . Gradient of L_1 is constant hence it will towards 0 but gradient of L_2 always change

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

$$\begin{aligned}
 & L_2: (w_1)_{j+1} = (w_1)_j - \gamma \left[\frac{\partial L_2}{\partial w_1} \right]_{w_1=j} \\
 & \text{Let } (w_1)_j = 0.05 \\
 & (w_1)_{j+1} = (w_1)_j - \gamma (2 \times w_1) \\
 & (w_1)_{j+1} = 0.05 - 0.01 \\
 & \text{Let } \gamma = 0.01 \\
 & (w_1)_{j+1} = 0.05 - 0.001 \\
 & (w_1)_{j+1} = 0.049
 \end{aligned}$$

$$\begin{aligned}
 & L_1: (w_1)_{j+1} = (w_1)_j - \gamma \left(\frac{\partial L_1}{\partial w_1} \right)_{w_1=j} \\
 & (w_1)_{j+1} = (w_1)_j - \gamma (0.05 - 0.01) \\
 & (w_1)_{j+1} = (w_1)_j - 0.04 \\
 & \text{Let } \gamma = 0.01 \\
 & (w_1)_{j+1} = (w_1)_j - 0.001 \\
 & (w_1)_{j+1} = 0.049
 \end{aligned}$$

V quickly:- $\left\{ \begin{array}{l} \text{less chance of } w_1 = 0 \text{ @ end of k-iter} \\ L_2 \text{ reg. doesn't change the value of } w_1 \text{ from one iteration to another} \end{array} \right.$
 { chance that $(w_1)_{j+1}$ are more at 1/e end of k-iter } $\left\{ \begin{array}{l} L_1 \text{ reg. continues to constantly reduce } w_1 \text{ towards } w_1^*(=0) \end{array} \right.$