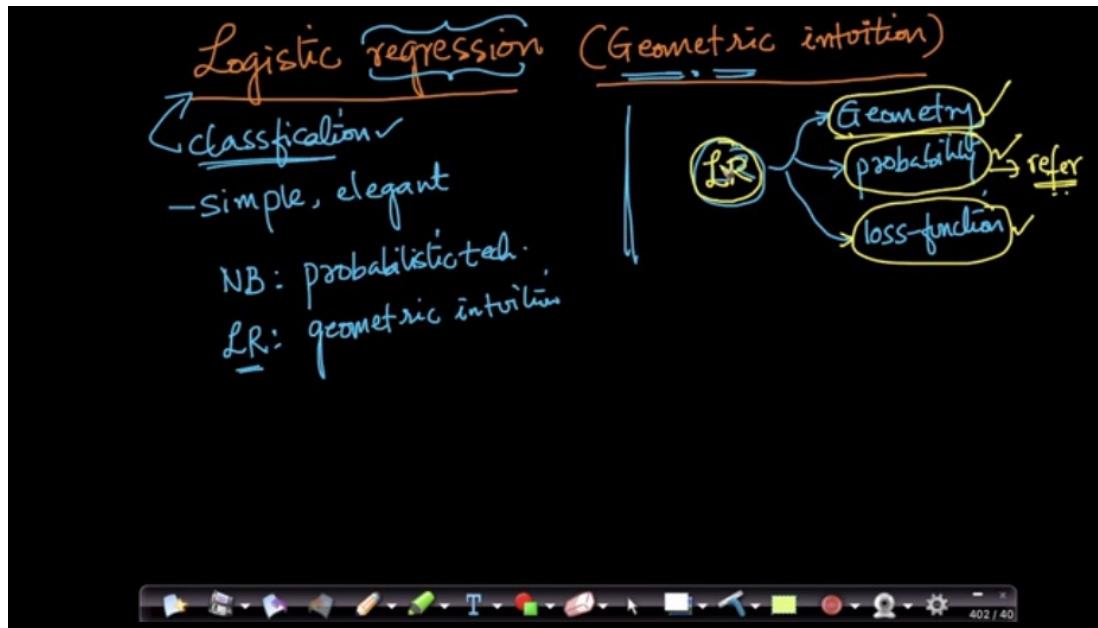


Notes

Logistic Regression

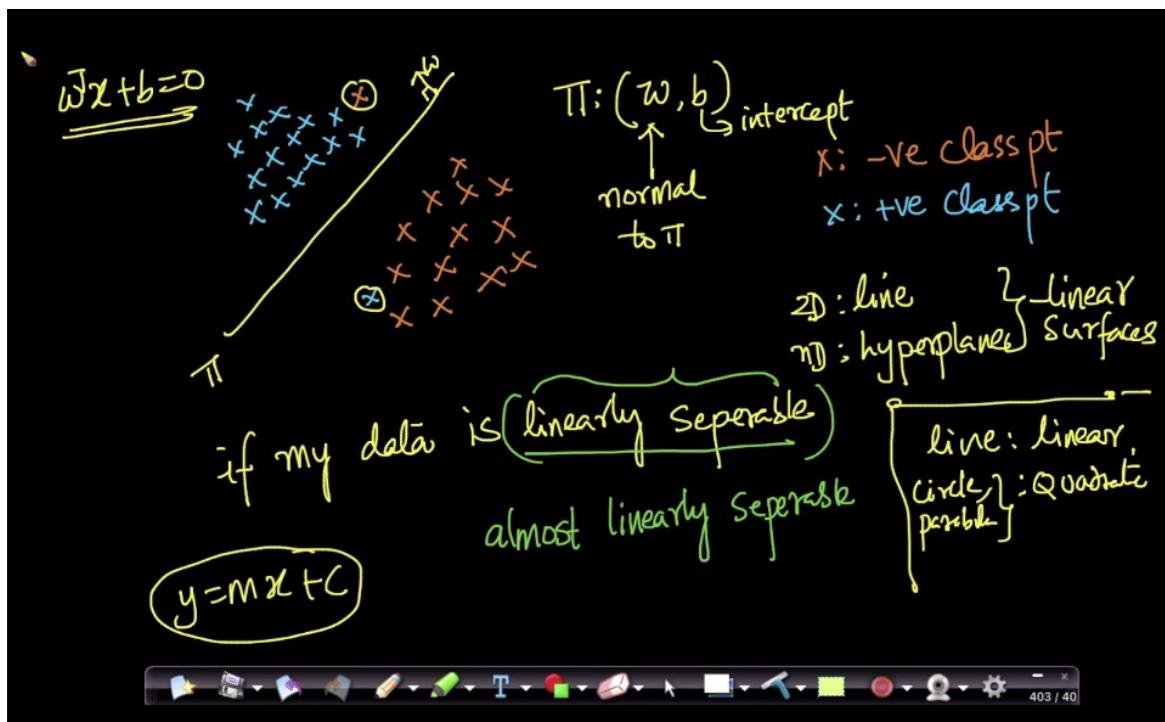
Geometric intuition of Logistic Regression:

LR can be derived using Geometry, probability and loss-function interpretation.



The two classes of points, we can draw a hyper plane in 2D to separate the points and hyper-plane in d – dimensional space.

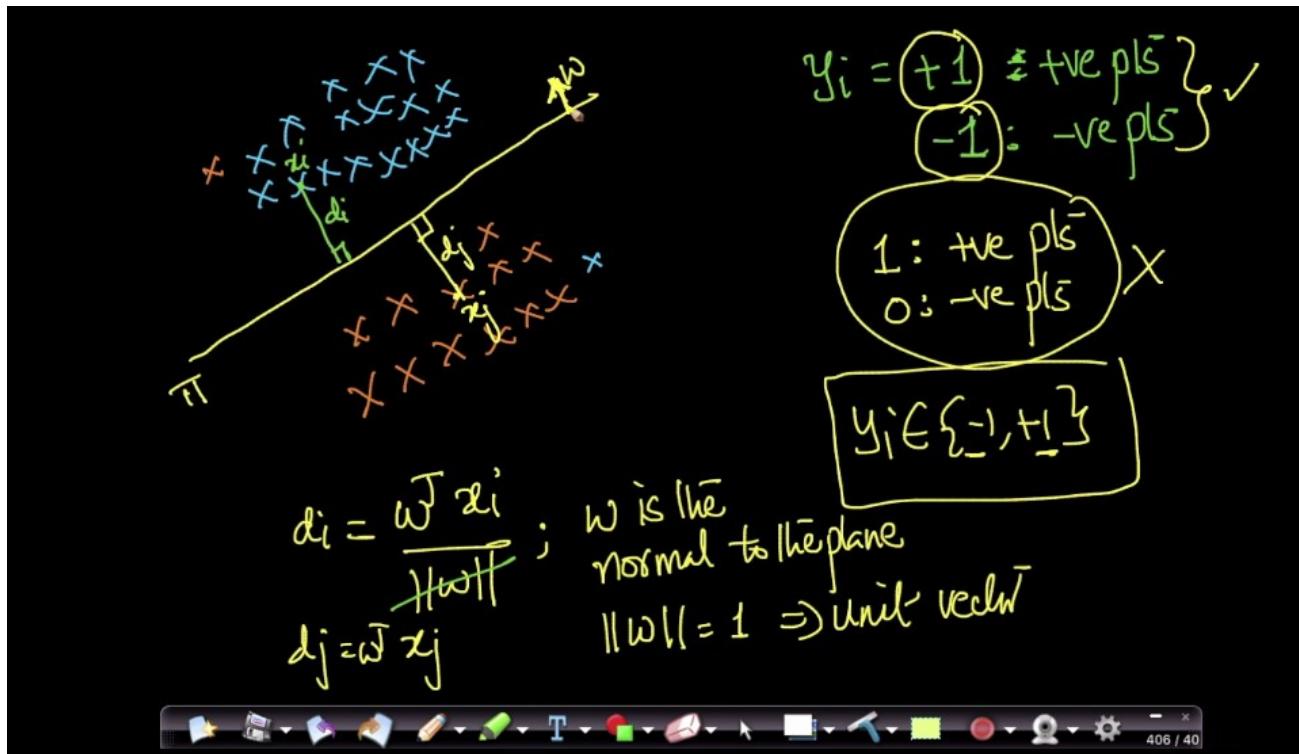
The data can be separated linearly and quadratic planes.



The equation of the plane is the product of the normal to the plane and the parameters of the plane plus the intercept term.

The equation of the line is $y=mx+b$.

Logistic regression assumes that the data is almost/perfectly linearly separable.



Here, w is the unit vector.

Note: The positive and negative points are denoted as +1 and -1 respectively.

The distance to the points x_i and x_j are computed where the distances are positive for x_i and negative for x_j .

Lets assume that the line passes through the origin.

$$\left. \begin{array}{l} d_i = \omega^\top x_i > 0 \\ d_j = \omega^\top x_j < 0 \end{array} \right\} \begin{array}{l} \text{classify} \\ \left\{ \begin{array}{l} \text{if } \omega^\top x_i > 0 \text{ then } y_i = +1 \\ \text{if } \omega^\top x_i < 0 \text{ then } y_i = -1 \end{array} \right. \end{array}$$

decision surface in LR
 \therefore line/plane

Different cases of optimization techniques.

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

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

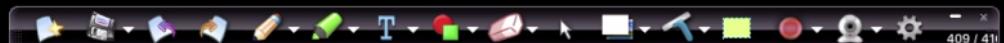


For both positive and negative points the $(y_i * w^T * x_i) > 0$, the model is correctly classifying the point x_i .

We want the point w_i that correctly classifies the point.

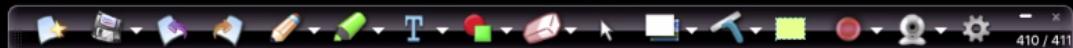
both +ve & -ve pts
 $\boxed{y_i w^T x_i > 0} \Rightarrow$ the LR model is correctly classifying the pt x_i

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



Case 4: $\begin{cases} y_i = -1 \Rightarrow \text{ne class} \\ w^T x_i > 0 \Rightarrow LR \text{ is saying } x_i \text{ is ne pt} \end{cases}$

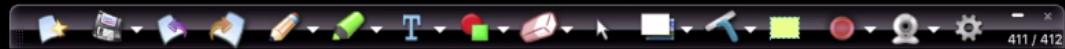
↳ Misclassified $y_i w^T x_i < 0$



Finally, we want our classifier to make minimum number of misclassifications.

Characteristics of LR:

Classifier to be v. good
 ↳ min. # misclassifications
 (m) max # correctly classified pls
 as many pls as possible
 to have $y_i w^T x_i > 0$



Mathematical formulation:

The objective function formulation.

$$\max_{\mathbf{w}} \sum_{i=1}^n (y_i \mathbf{w}^\top \mathbf{x}_i)$$

variable

$(x_i, y_i) \rightarrow$ fixed $\Rightarrow D_n$

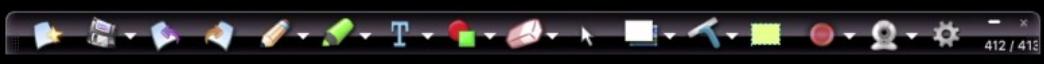
Change / Vary \mathbf{w}

$$\left\{ \begin{array}{l} \mathbf{w}^* = \arg \max_{\mathbf{w}} \left(\sum_{i=1}^n y_i \mathbf{w}^\top \mathbf{x}_i \right) \\ \text{variable} \end{array} \right.$$

Math. optimization problem

+ve: correctly classified
-ve: misclassified

Optimal \mathbf{w}



412 / 415

The ultimate goal is to find the optimal \mathbf{W} that maximizes the accuracy of the model.

Sigmoid function: Squashing

Our objective is

$$\arg \max_{\mathbf{w}} \sum_{i=1}^n y_i \mathbf{w}^\top \mathbf{x}_i$$

signed distance

$\mathbf{w}^\top \mathbf{x}_i$: dist from x_i to π (\mathbf{w} is a unit vector)

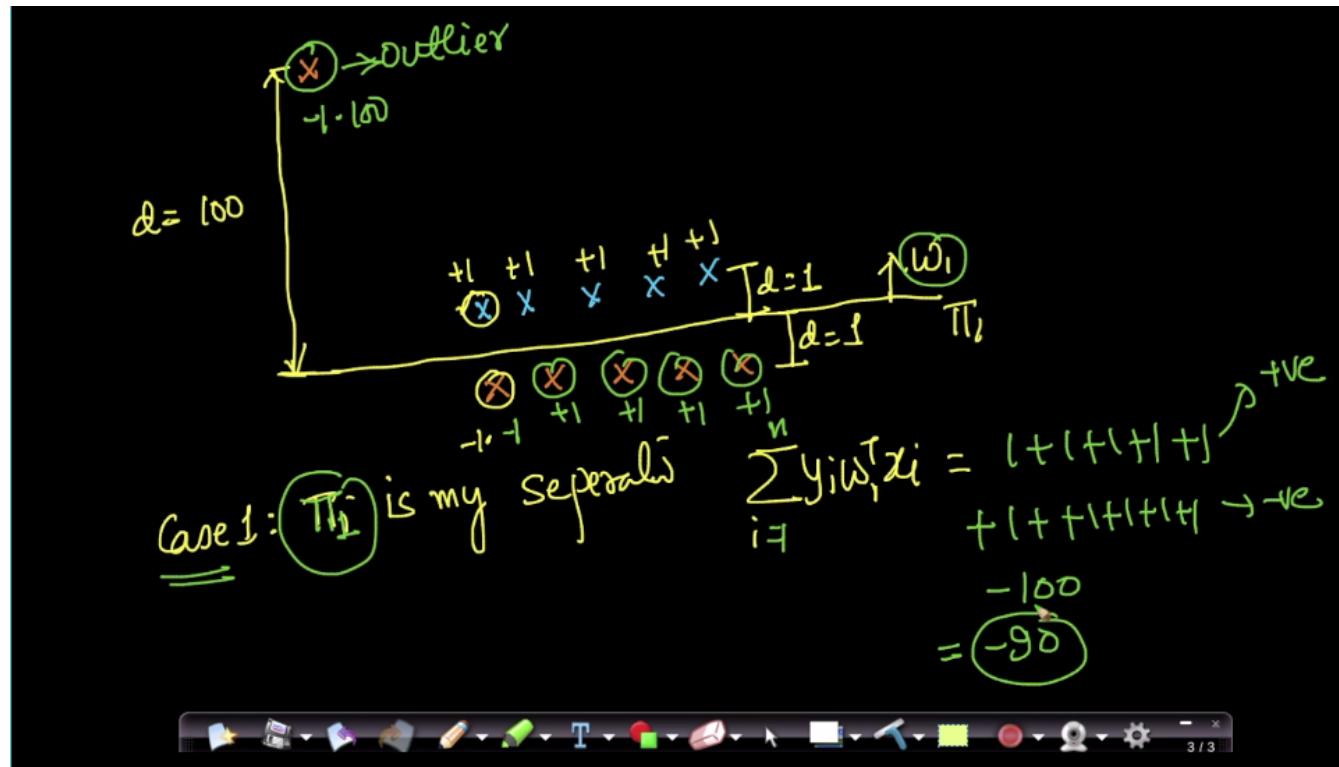
$y_i \mathbf{w}^\top \mathbf{x}_i$: +ve $\Rightarrow \pi$ as defined by \mathbf{w} correctly classifies x_i
-ve \Rightarrow incorrectly classifies x_i



2 / 2

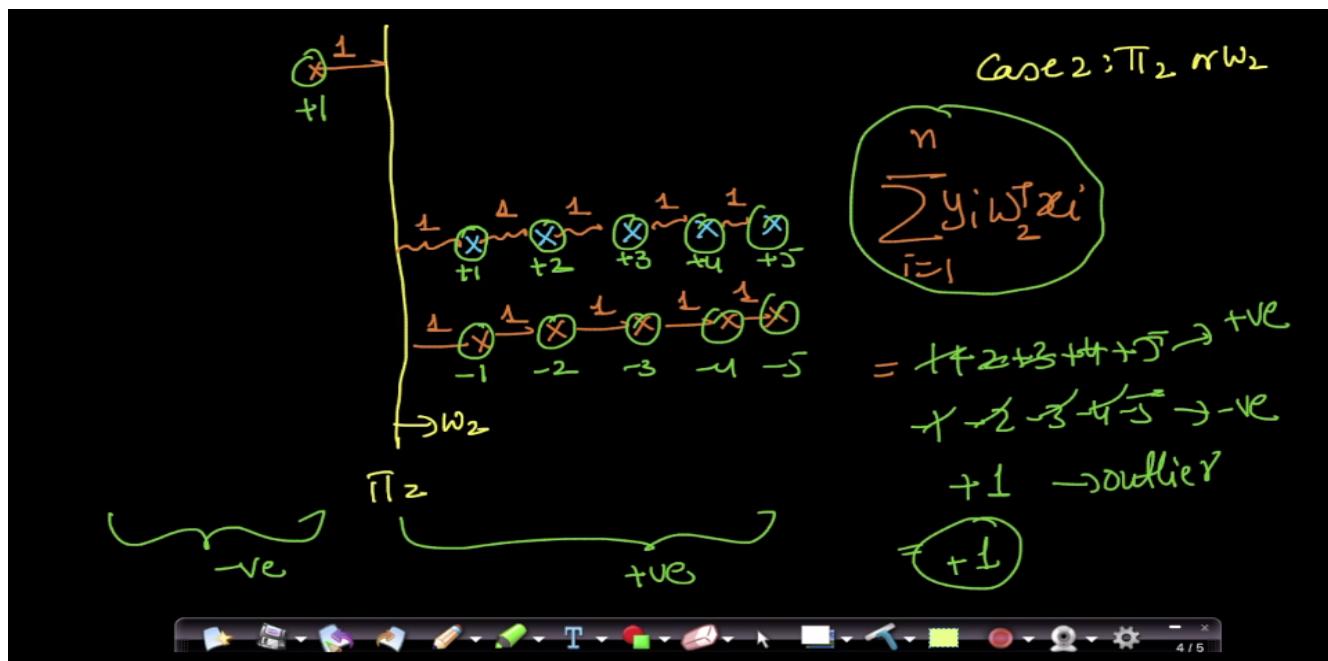
Edge cases:

All the points are assumed to be at the unit distance from the hyper plane, except one -ve point considered as outlier at a distance of 100 units. This point will mess up the objective function.



Our objective is to maximize the sum of signed distances.

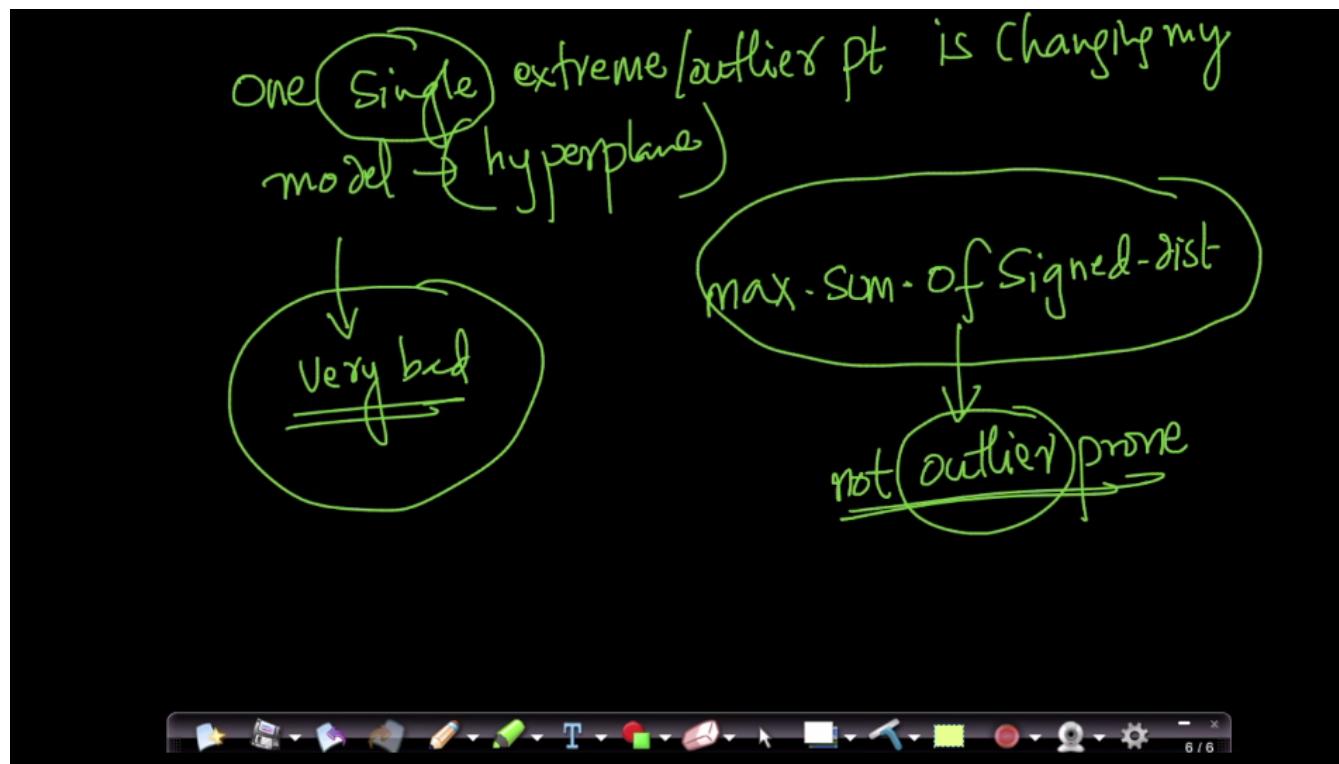
Edge case hyper plane:



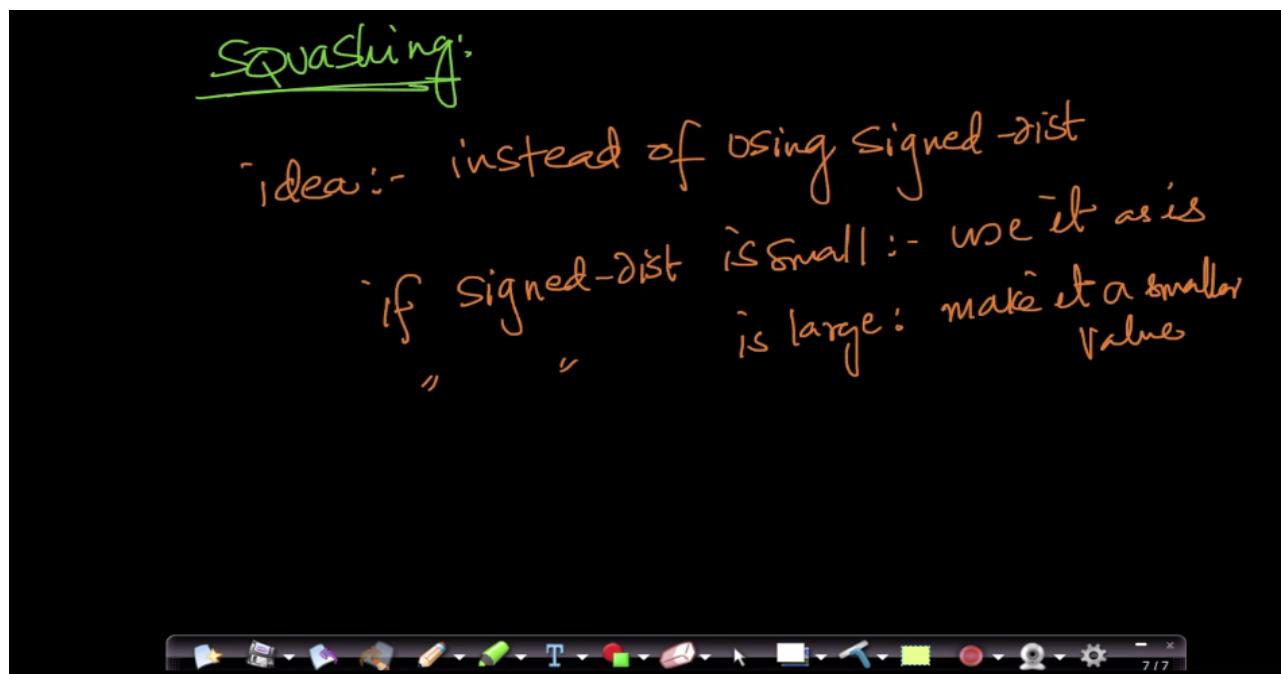
The sum of signed distance for **pi1** is less than the signed distance **pi2** then we will choose the **pi2**.

But geometrically the plane **pi2** is a terrible choice, because accuracy suffers.

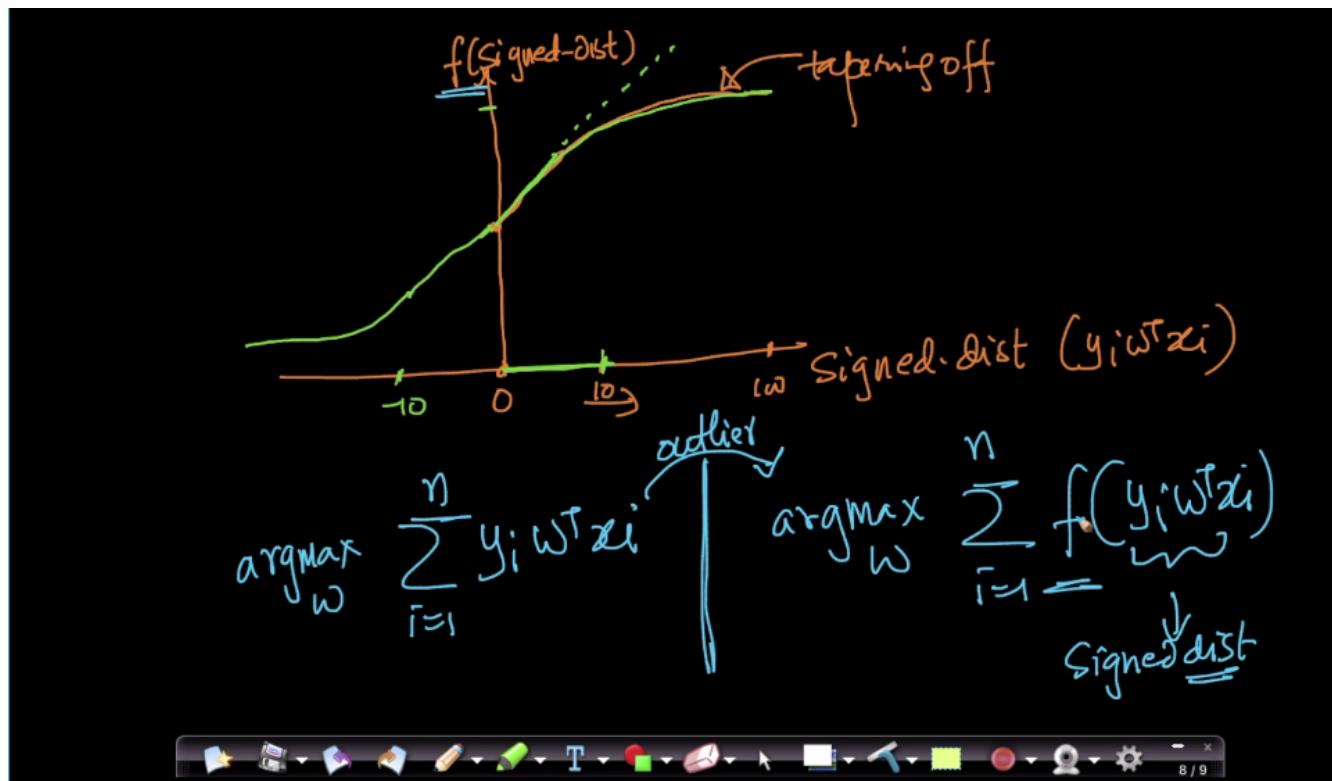
Intuitively, **pi1** is better than **pi2**.



In order to overcome this situation we will use the technique called squashing.

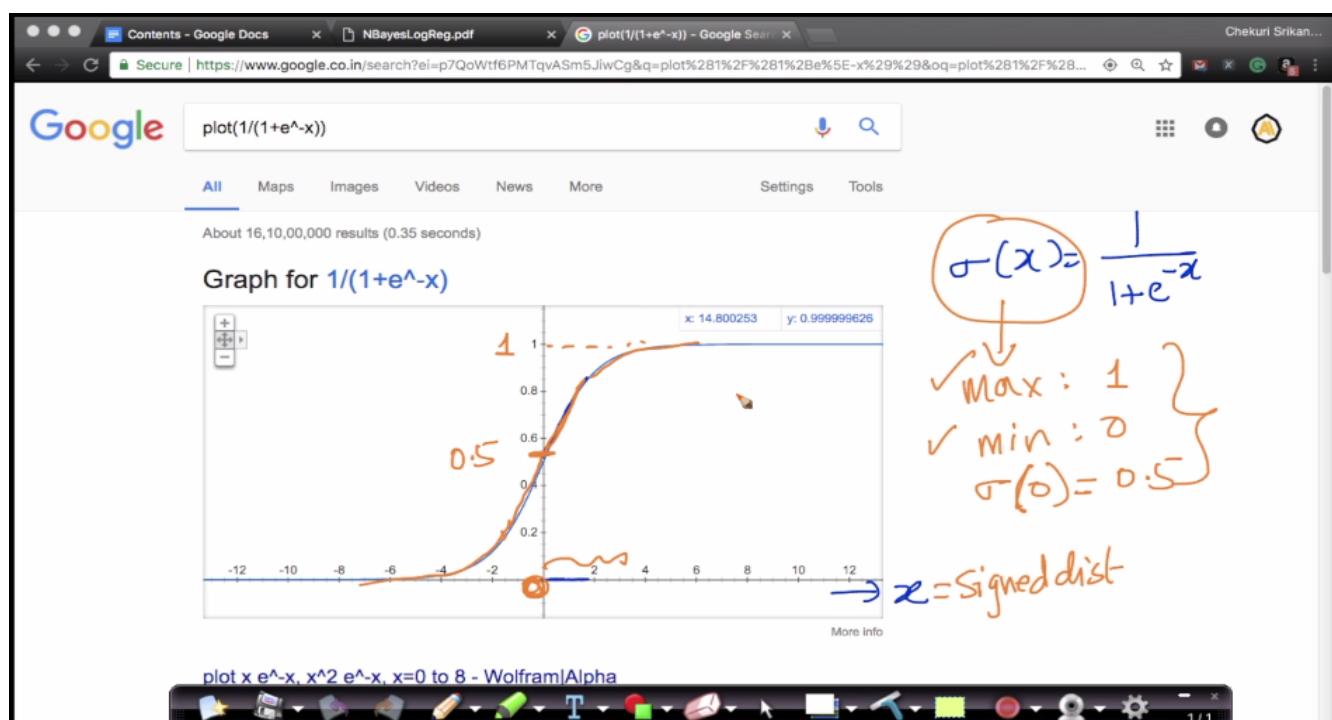


In this technique we will apply a function on top of signed distances to make it prone from the outlier effect.



The sigmoid function makes this possible.

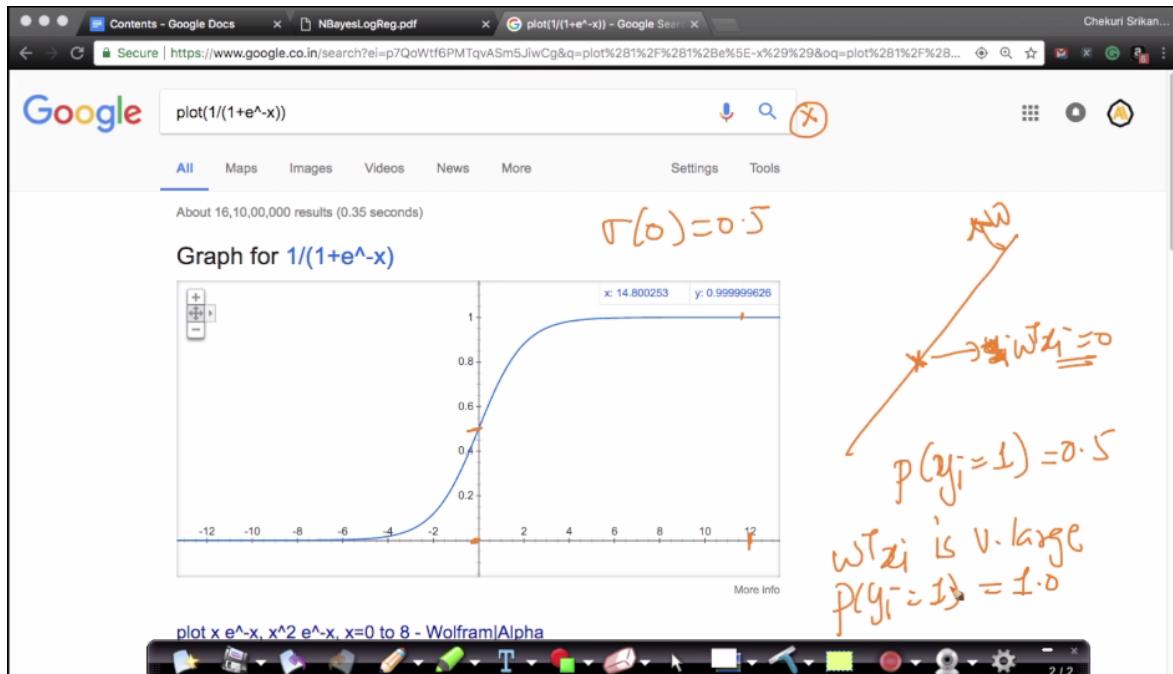
Properties of Sigmoid function:



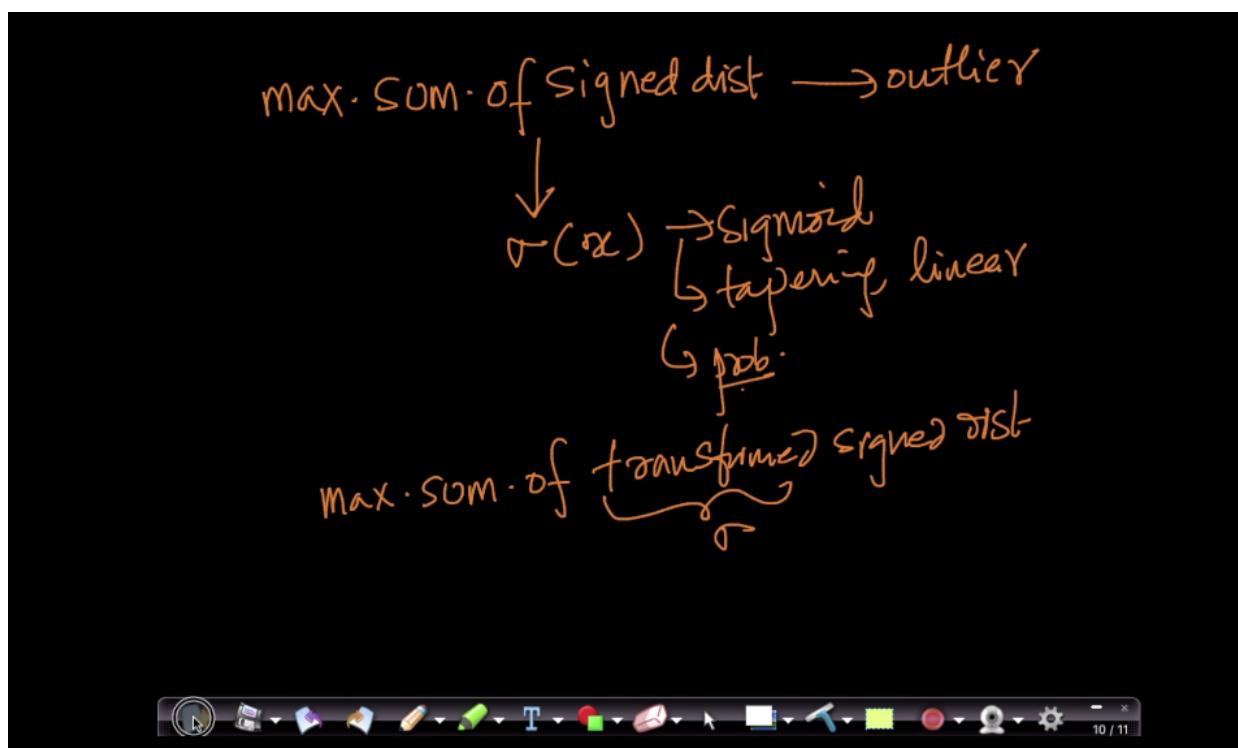
We choose sigmoid function because it has probabilistic interpretation.

If the point is on the curve we cannot decide the probability of the class, sigmoid function gives the probability as 0.5.

If the signed distance is very large then the class will be near to the maximum of the sigmoid that is 1.



Our transformed objective function:



Transformed mathematical objective function.

$$\omega^+ = \arg \max_{\omega} \sum_{i=1}^n \sigma(y_i \omega^T x_i) \quad \sigma(x) = \frac{1}{1 + e^{-x}}$$
$$\omega^+ = \arg \max_{\omega} \sum_{i=1}^n \frac{1}{1 + \exp(-y_i \omega^T x_i)}$$

less impacted by outliers

Mathematical formulation of Objective function:

A function $g(x)$ is said to be monotonic function as x increases $g(x)$ increases.

$$\left\{ \begin{array}{l} \text{Monotonic fn:- } g(x) \\ x \uparrow; g(x) \uparrow \rightarrow \text{monotonically incr-fn} \\ \text{if } x_1 > x_2 \text{ then } g(x_1) > g(x_2) \\ \text{then } g(x) \text{ is said to be mon. incr fn} \end{array} \right.$$

Optimizing the $\text{fun}(\text{fun}(x))$:

We can claim that the value of the whole function that minimizes is equal to the value that minimizes the inner function. Since, both the functions are monotonic.

mon. func

$$g(x) = \underline{\log(x)}$$

$$\textcircled{1} \rightarrow \boxed{0} = (\underline{x^*}) = \arg \min_x f(x) ; f(x) = \underline{x^L}$$

$$\textcircled{2} \rightarrow x' = \arg \min_x \underbrace{g(f(x))}_{\log(x^L)}$$

claim: $\underline{x = x'}$ $\because g(x)$ is a monotonic fn

This is the function to be optimized.

$$w^* = \arg \max_w \sum_{i=1}^n \frac{1}{1 + \exp(-y_i w^T x_i)}$$

✓ $g(x) : \log(x) : \text{monotonic fn:-}$

$$w^* = \arg \max_w \sum_{i=1}^n \log \left(\sigma(y_i w^T x_i) \right)$$

Apply log to the objective function.

$$\omega^* = \underset{\omega}{\operatorname{argmax}} \sum_{i=1}^n \log \left\{ \frac{1}{1+\exp(-y_i \omega^T x_i)} \right\}$$

$$\log(1/x) = -\log(x)$$

$$\omega^* = \underset{\omega}{\operatorname{argmax}} \sum_{i=1}^n -\log \left(1 + \exp(-y_i \omega^T x_i) \right)$$



In optimization the value which maximizes the $f(x)$ is also minimizes the $-f(x)$.

$$\underset{x}{\operatorname{argmax}} f(x) = \underset{x}{\operatorname{argmin}} -f(x)$$

$$\underset{x}{\operatorname{argmax}} -f(x) = \underset{x}{\operatorname{argmin}} f(x) \rightarrow (+1)(-1)$$

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^n \log \left(1 + \exp(-y_i \omega^T x_i) \right)$$



Removing the 1 is will be more favorable, because the log and exp will cancel out each other.

$$\checkmark \arg \min_w \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) \rightarrow \text{optimal}$$

$$\arg \min \sum = y_i w^T x_i$$

$$\arg \max \sum y_i w^T x_i \rightarrow \begin{array}{l} \text{Sum of signed hist} \\ \text{huge outlier problem} \end{array}$$

Other formulation of the Objective function(probabilistic method). The both formulations are one and the same.

$$\stackrel{\text{geometry}}{\checkmark} \quad w^* = \arg \min_w \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) \rightarrow \text{+1 or -1}$$

$$\stackrel{\text{Prob. method}}{\checkmark} \quad w^* = \arg \min_w \sum_{i=1}^n (y_i \log p_i - (1-y_i) \log(1-p_i)) \quad p_i = \sigma(w^T x_i) \quad \boxed{\text{loss fn}}$$

Weight vector:

W is also a d dimensional vector.

Weight -Vector

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

↓

Weight -Vectw
 $w = \langle w_1, w_2, w_3, w_4, \dots, w_d \rangle$

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

x_i f_i R^d



Probabilistic interpretation:

$$w = \langle w_1, w_2, \underline{w_3}, \dots, w_d \rangle$$
$$f_1, f_2, \underline{f_3}, \dots, f_d$$

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: $\sum_{0 \neq i} (w^T x_i) = P(y_i = +1)$



For each feature we have the weight vector.

$$\omega = \langle \omega_1, \omega_2, \dots, \omega_d \rangle$$

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

decision: $x_q \rightarrow y_q$ { if $\omega^T x_q > 0$ then $y_q = +1$ ✓
if $\omega^T x_q < 0$ then $y_q = -1$ ✗

Prob.interp: $\sum_{i=1}^d (\omega_i x_{qi}) = P(y_q = +1)$

Interpretation of W:

Case – 1:

As w_i increases the point $x_q * w_i$ also increases and the probability of the point y_q will also increase

interpretation of ω :

Case (1) If $\omega_i = +ve$, $x_{qi} \uparrow \Rightarrow (\omega_i x_{qi}) \uparrow$
 $\Rightarrow \left(\sum_{i=1}^d \omega_i x_{qi} \right) \uparrow$
 $\Rightarrow \sum (\omega_i x_{qi}) \uparrow$
 $\Rightarrow P(y_q = +1) \uparrow$

Case – 2:

if w_i is negative the then $w_i \cdot x_{q_i}$ also decrease and the probability of y_{q_i} being 1 also decreases, which implies y_{q_i} being -1 also increases.

case 2: if $w_i = -ve$

f_i $x_{q_i} \uparrow \Rightarrow (w_i x_{q_i}) \downarrow \Rightarrow \left(\sum_{i=1}^n w_i x_{q_i} \right) \downarrow$

$p(y_{q_i} = +1) = 1 - p(y_{q_i} = -1)$

$\Rightarrow p(y_{q_i} = +1) \downarrow$

$\Rightarrow p(y_{q_i} = -1) \uparrow$

L2 Regularization: Over fitting and Under fitting:

Z is the signed distance.

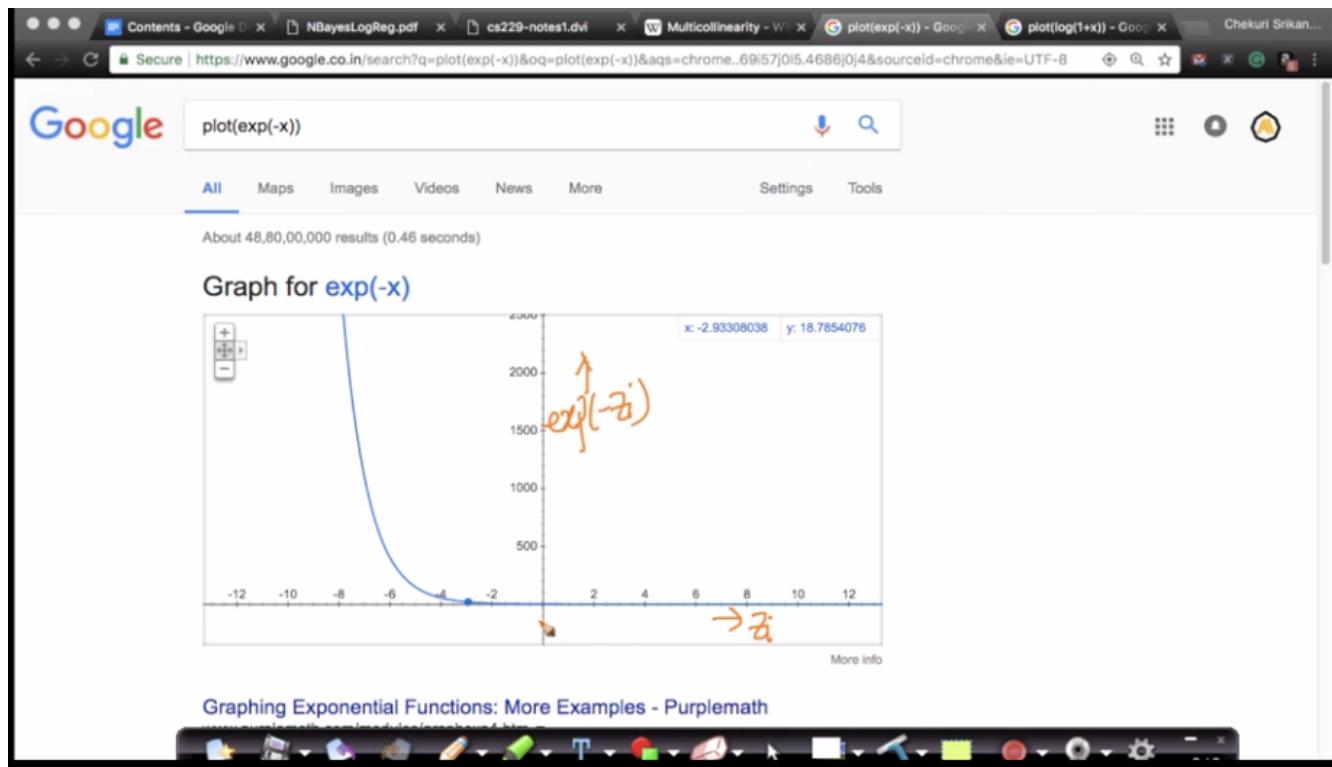
L_2 regularization: Overfitting vs Underfitting:

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

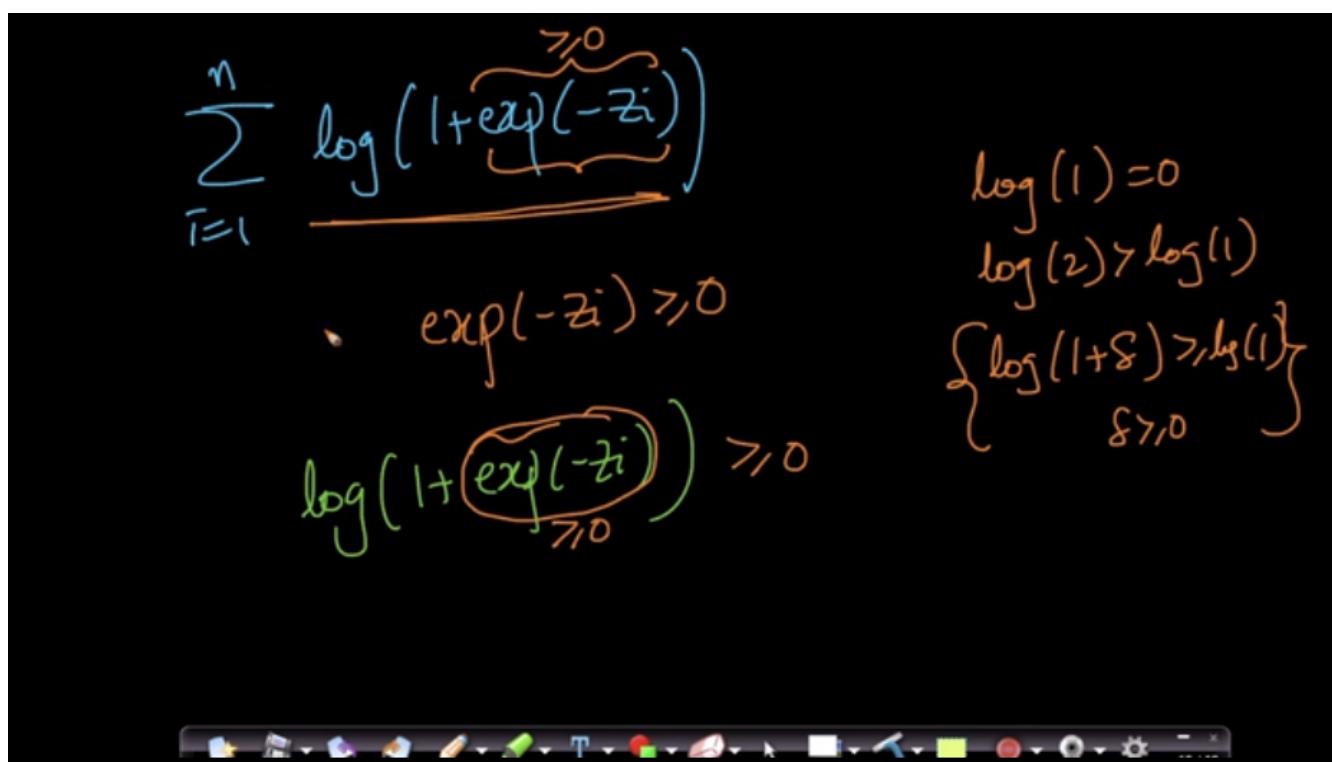
let $Z_i = y_i w^\top x_i$

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

In the Objective function, $\exp(-z_i)$ is always positive.



Interpretation of different functions.



The minimal value of the function occurs at Zero.

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

minimal value of $\sum_{i=1}^n \log(1 + \exp(-z_i))$ is "0"



If z_i tends towards infinite then $\exp(-z_i)$ tends towards 0

$$\omega^* = \arg \min_{\omega} \sum_{i=1}^n \log(1 + \exp(-z_i))$$

if $z_i = +ve$, $z_i \rightarrow +\infty$
then $\exp(-z) \rightarrow 0$

$$\log(1 + \exp(-z)) \rightarrow 0$$



The minimal value of the function is zero when z_i tends to infinite.

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

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

As the function above is monotonic function, the solution for the optimization problem will be w_i is either +infinite or -infinite and over fit to the data.

So, in order to overcome the problem we use the regularization term for making the objective function not equal to zero.

$$w^* = \arg \min_w \left[\sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) + \lambda \|w\|_2^2 \right]$$

regularization loss-term
 regularization term
 $w_j \rightarrow \infty$ $w_j \rightarrow -\infty$

The two terms in the function tries to attain the right value of the objective function.

$$J(\theta) = \sum_{i=1}^n \log(1 + \exp(-z_i)) + \frac{\lambda}{2} \sum_{j=1}^d w_j^2$$

Min

loss-term

reg-term

$w_j \rightarrow +\infty$ overfit

$z_i \rightarrow +\infty$ underfit

Lambda is the hyper parameter in Logistic regression.

If lambda = 0, then the model over fit the training data.

If lambda = large then the model will under fit the data.

$$\arg \min_{\theta} \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2} w^T w$$

λ : hyperparam in LR

$\lambda = 0 \Rightarrow$ overfit to the training set

$\lambda = \text{large} \Rightarrow$ underfit

hyper param

K-NN : K

Laplace Smoothing (NB)

$\alpha \rightarrow CV$

The general equation of optimization function in Machine Learning will be.

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

Right lambda is determined using cross-validation.

→ Cross-validation

$\lambda = 0 \Rightarrow$ overfit \rightarrow high variance

$\lambda = \text{large} \Rightarrow$ underfit \rightarrow high bias

L1 regularization and sparsity:

L1 regularization is the alternate to L2 regularization.

L2 norm is nothing but the Manhattan distance.

$$\|w\|_2^2 \text{ fn reg}$$

\downarrow

$$\|w\|_1 \text{ for reg.}$$

$\|w\|_1 = \sum_{i=1}^d |w_i|_{\text{abs}}$

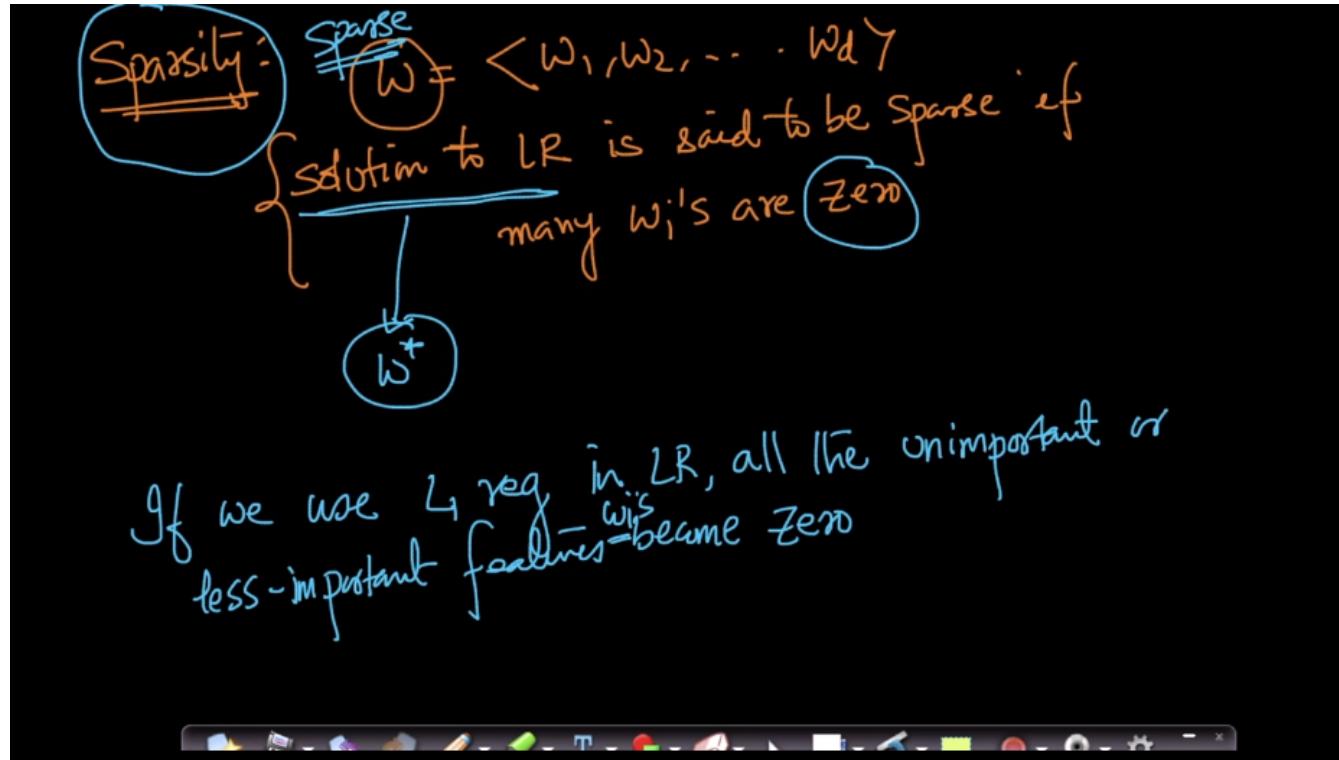
$w^* = \underset{w}{\operatorname{argmin}} \left(\underbrace{\text{logistic loss}_{\text{fn training data}}}_{\text{will avoid } w_i \rightarrow \pm\infty} + \lambda \underbrace{\|w\|_1}_{L1-\text{reg}} \right)$

L1 regularization serves the same purpose of the L2 regularization. Since it uses the absolute value of W.

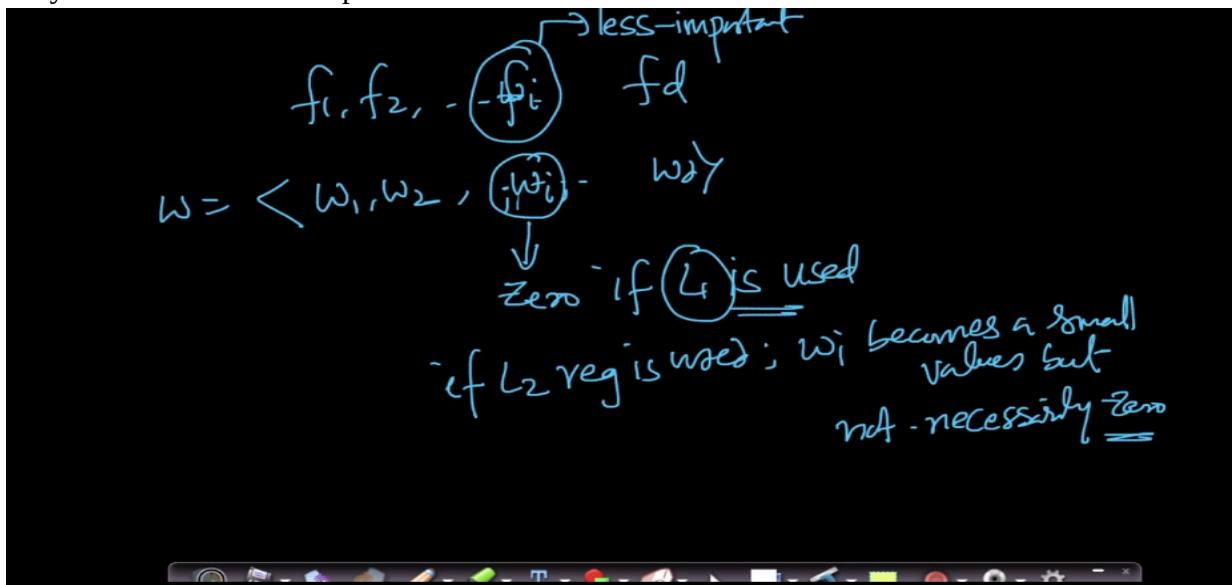
It has one disadvantage.

Sparsity- The vector is said to be sparse if many of the values of the vector are zeros.

If we use L1 regularization all the less important features become Zeros.



If any of the feature less important the w_i value for the feature becomes Zero.



There is elastic – net implementation of regularization in which they take both L1 and L2 norm.
In this case we have two hyper parameters.

elastic-net: either L1 or L2

$$\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^n \log(1 + \exp(-x_i^\top \omega)) + \lambda_1 \|\omega\|_1 + \lambda_2 \|\omega\|_2^2$$

Two hyper-param: - λ_1 & λ_2

Probabilistic Interpretation: Gaussian Naive Bayes for logistic regression:

The random variable for coin tosses can be used as Bernoulli random variable.

~~geom:~~ $\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^n \log(y_i \omega^\top x_i) + \text{reg}$

~~prob:~~ $\omega^* = \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^n -y_i \log p_i - (1-y_i) \log(1-p_i) + \text{reg}$

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

+1 or 0

The both formulations are one and the same.

When y_1 is positive.

Case 1: $\underline{y_i : +ve}$

geom:- $y_i = +1$
prob:- $y_i = +1$

geom:- $\log\left(1 + \exp(-w^T x_i)\right)$
prob:- $-1 \cdot \log\left(\frac{1}{1 + \exp(-w^T x_i)}\right)$
 $= \log\left(1 + \exp(-w^T x_i)\right)$

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

$-\log(x) = \log(1/x)$

When y_i is negative.

Case 2: $\underline{y_i = -ve}$

$y_i = -1 \leftarrow$ geom
 $y_i = 0 \leftarrow$ prob.

geom: $\log\left(1 + \exp(w^T x_i)\right)$

prob: $-1 \cdot \log\left(1 - \frac{1}{1 + \exp(-w^T x_i)}\right)$
 $= -1 \cdot \log\left(\frac{\exp(-w^T x_i)}{1 + \exp(-w^T x_i)}\right)$

$-\log(x) = \log(1/x)$

$$= \log \left(\frac{1 + \exp(-w^T x_i)}{\exp(-w^T x_i)} \right)$$

divide numerator & den by $\exp(-w^T x_i)$

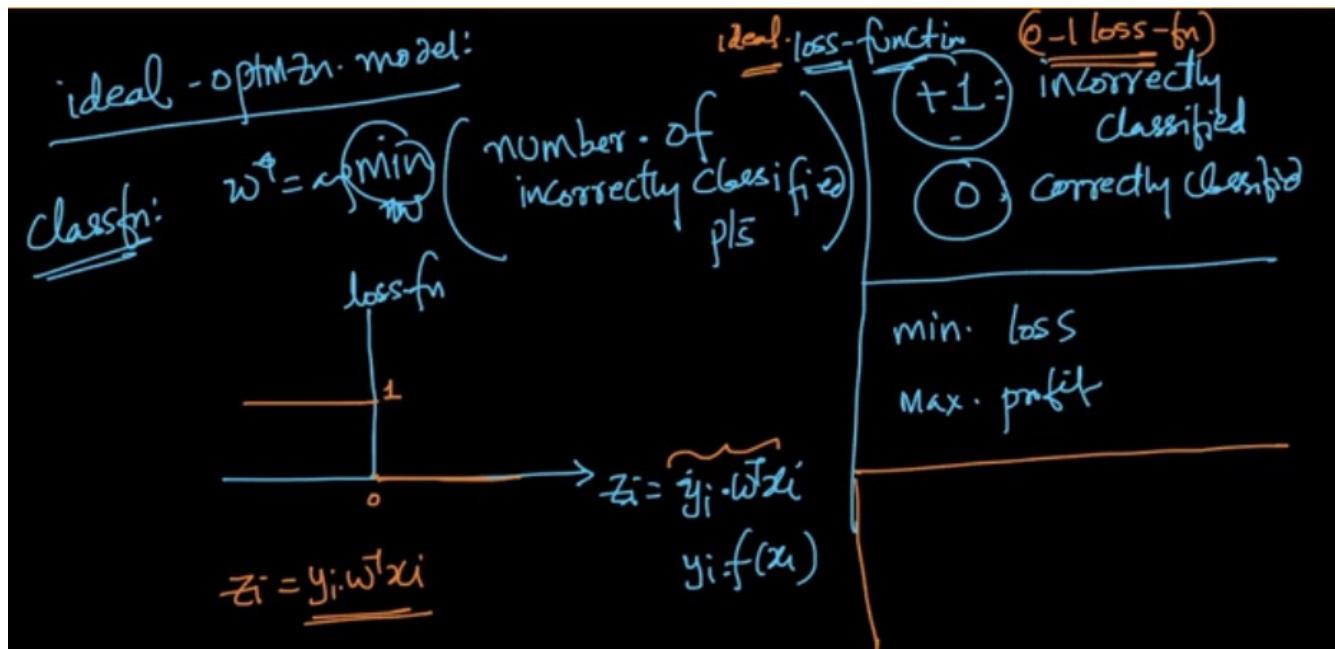
$$= \log \left(1 + \frac{1}{\exp(w^T x_i)} \right)$$

$\frac{1}{e^{-x}} = e^x$
④

$$= \log \left(1 + \exp(w^T x_i) \right)$$

Loss minimization interpretation:

An ideal optimization model can be found by minimizing the miss-classification.



Mathematical formulation

ideal

$$\hat{\omega}^* = \underset{\omega}{\operatorname{argmin}} \sum_{i=1}^n \underbrace{o_1 \text{ loss}(x_i, y_i, \omega)}_{}$$
$$o_1 \text{ loss}(z_i) = \begin{cases} 1 & \text{if } z_i < 0 \\ 0 & \text{if } z_i \geq 0 \end{cases}$$

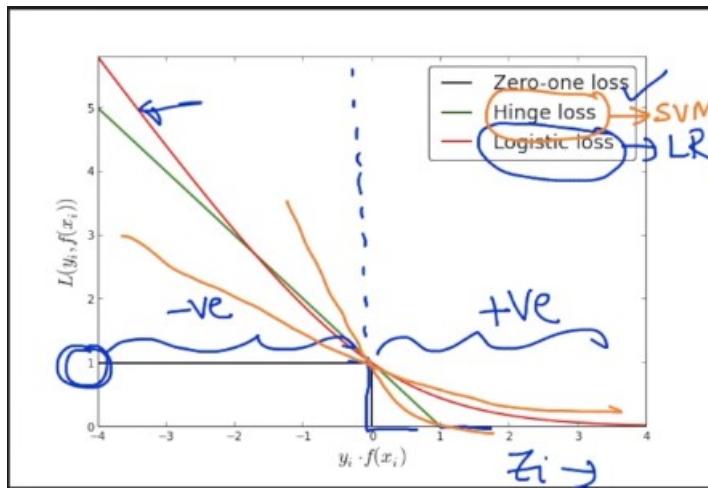

To solve optimization problems in machine learning we use calculus.

solve optimization problems in ML
↳ "differentiation" in calculus.
↳ next chapter

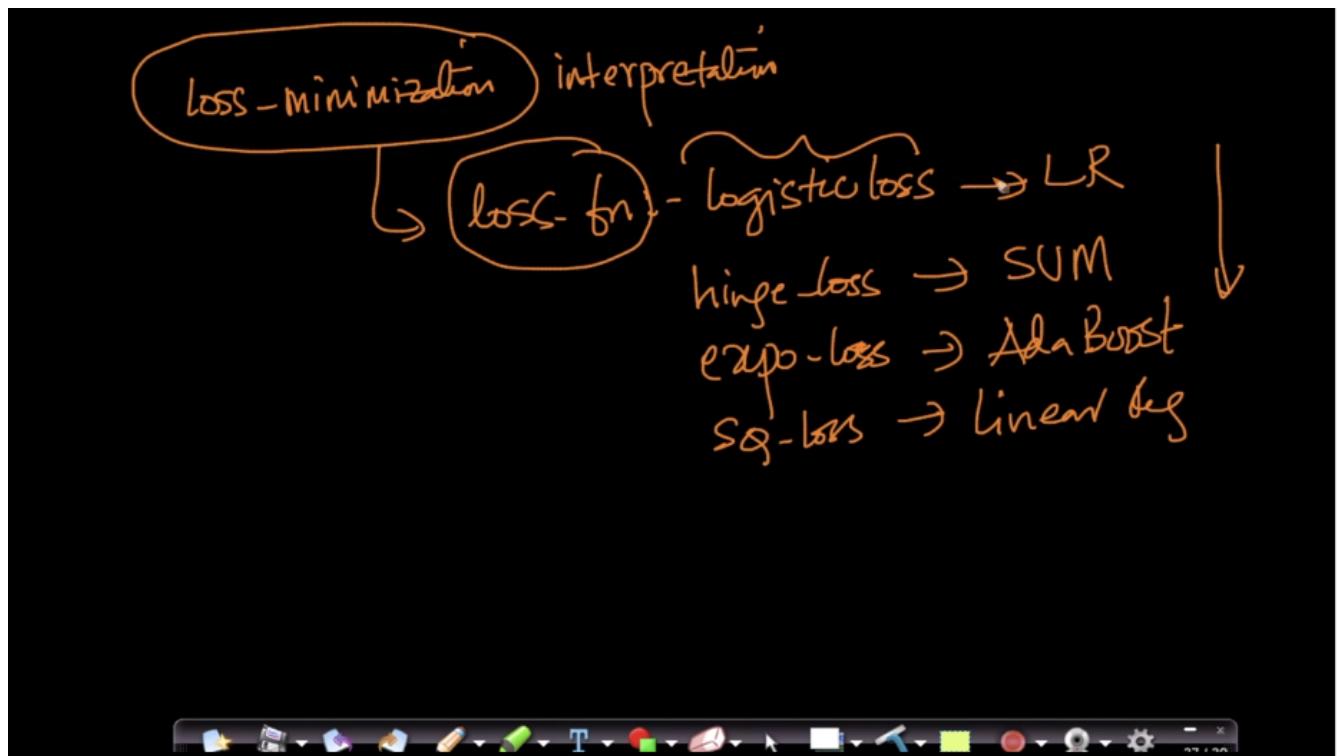
loss function is also called as 0 – 1 loss.

We need continuous functions to differentiate the loss function. Since, we cannot differentiate the 0-1 loss we will use approximate to obtain **logistic loss**.

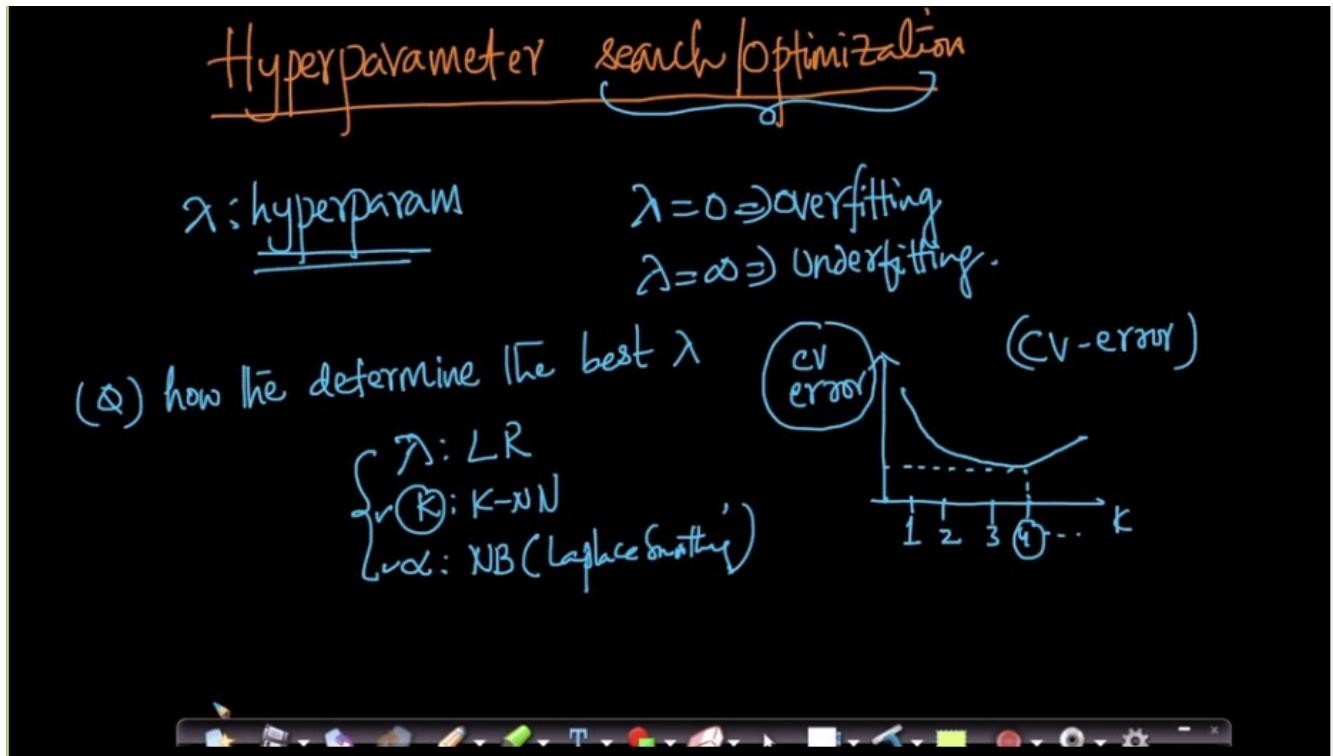
The second form of approximate the loss is **hinge loss**.



Various loss function minimization leads to multiple machine learning algorithms.



Hyper parameter search: Grid Search and Random Search

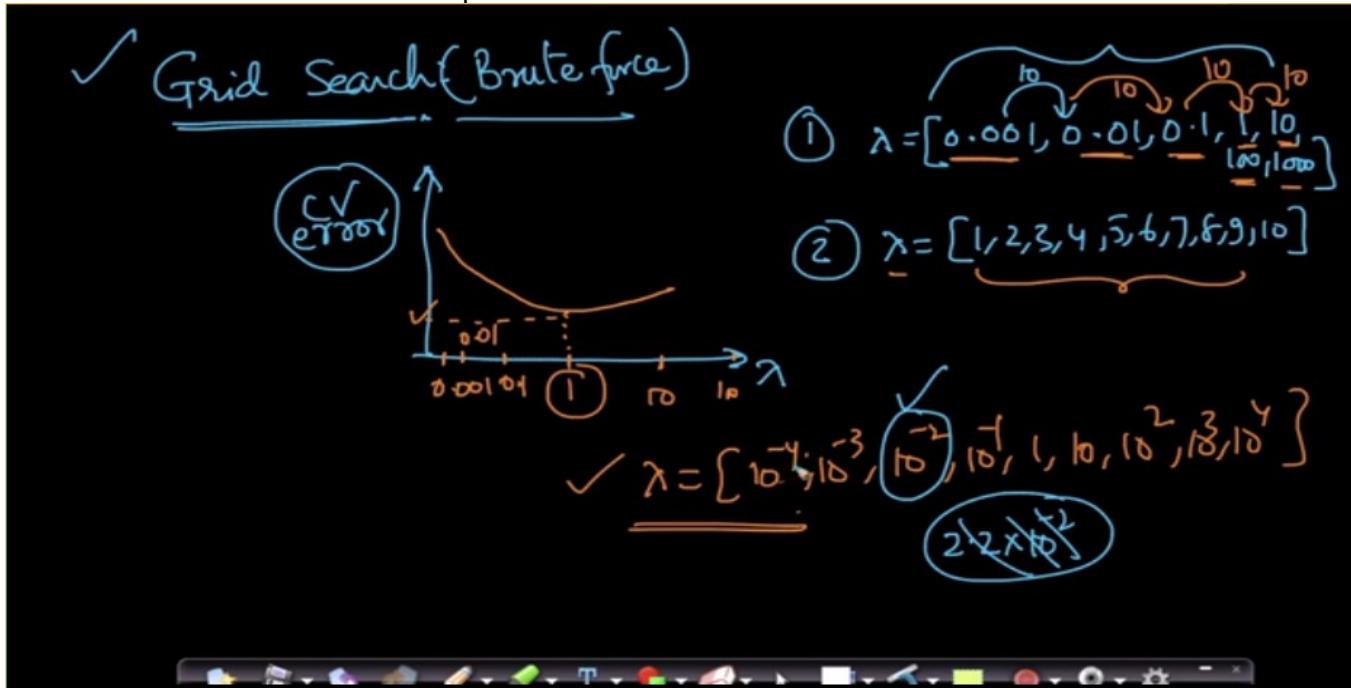


K in KNN is an integer $\{1, 2, 3, 4, 5, \dots, n\}$

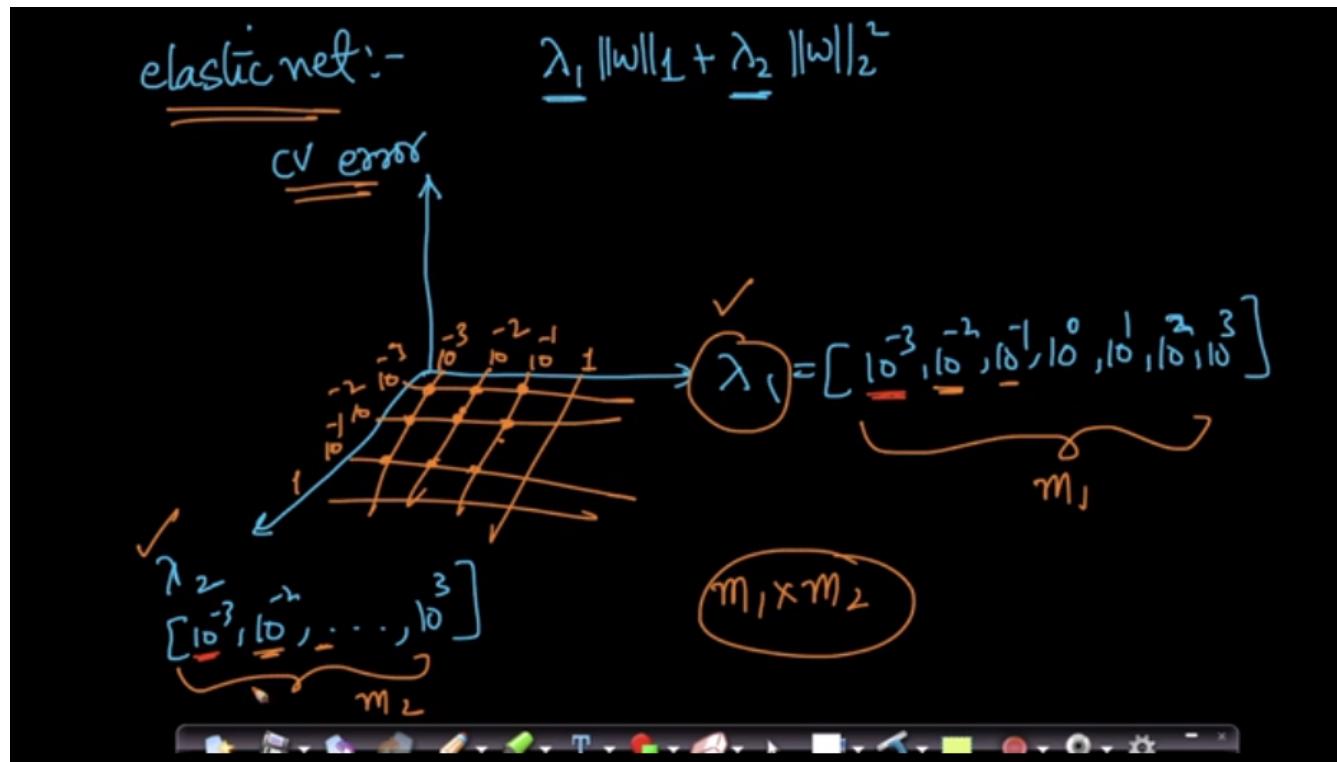
Lambda in the LR is a real number

Grid Search is the technique to find the Hyper parameters.

Grid Search is a brute force technique.



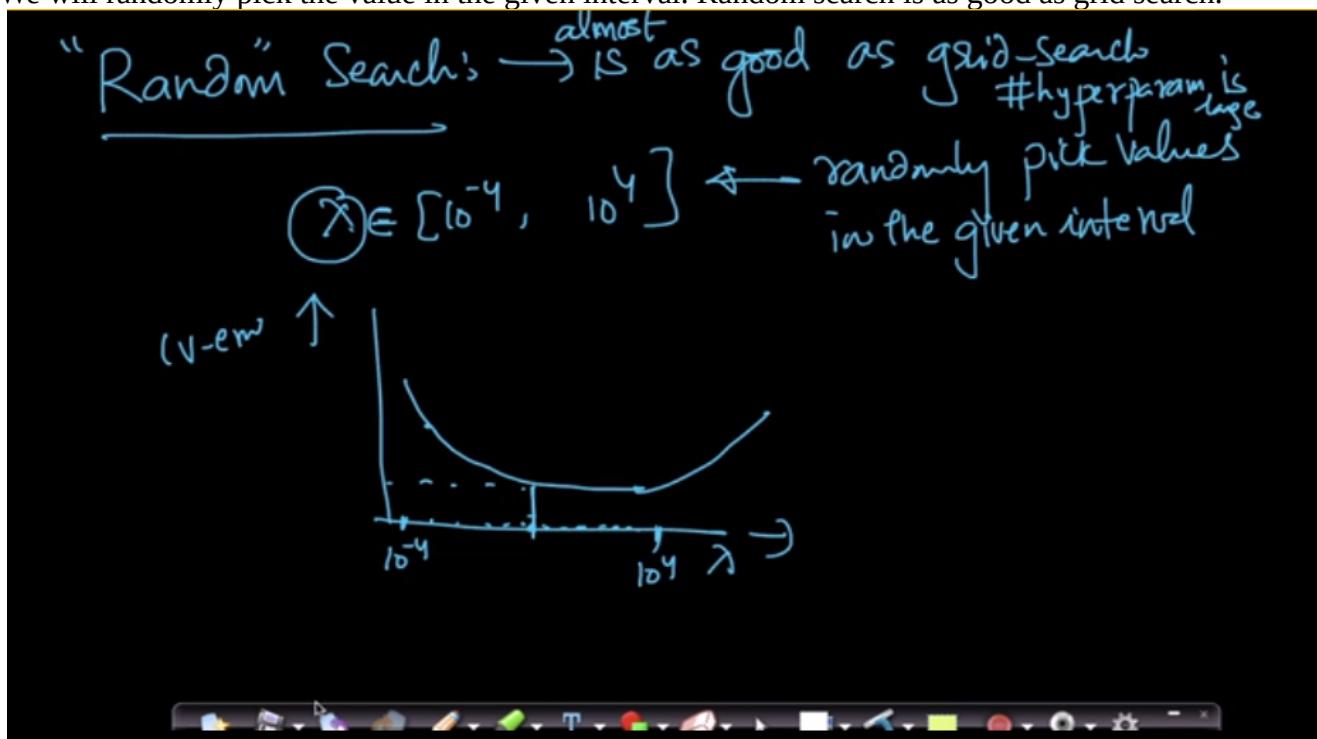
For elastic net hyper parameter choosing:



Grid search is not good in case of many hyper parameters.

There is an alternative to the grid search called random search.

We will randomly pick the value in the given interval. Random search is as good as grid search.



Column Standardization:

Column / feature standardization

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

$\left\{ z_{ij}' = \frac{x_{ij} - \mu_j}{\sigma_j} \right\}$: standardization

K-NN: - distances
(standardize our feature)

Even in Logistic regression it is **mandatory to perform feature (or) column standardization.**

Distances are impacted by the feature scales.

Feature importance and model interpretability:

If all features are independent

Feature importance & Model Interpretability

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

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

$LR: GNB + \underline{\text{Bernoulli}}$

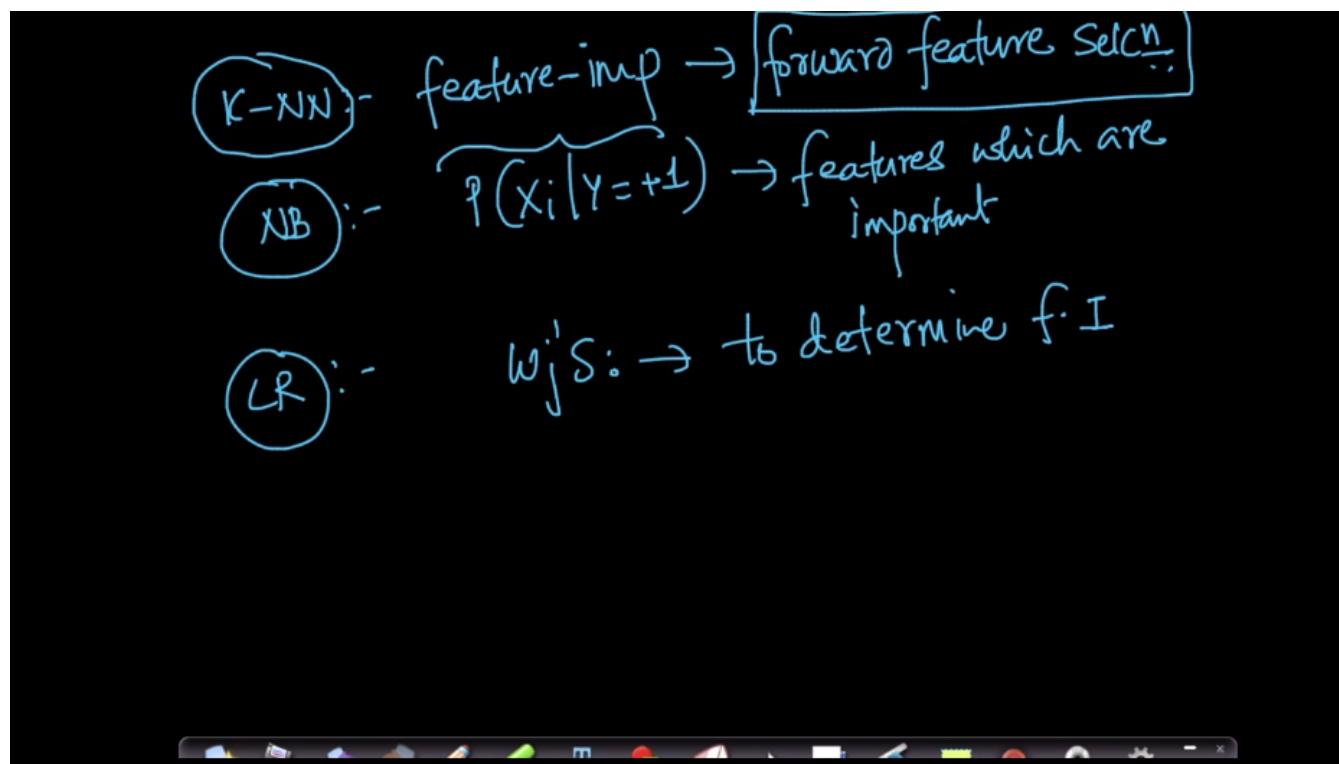
assume: if all features are independent (Naive Bayes)

feature-imp: $w_j's$

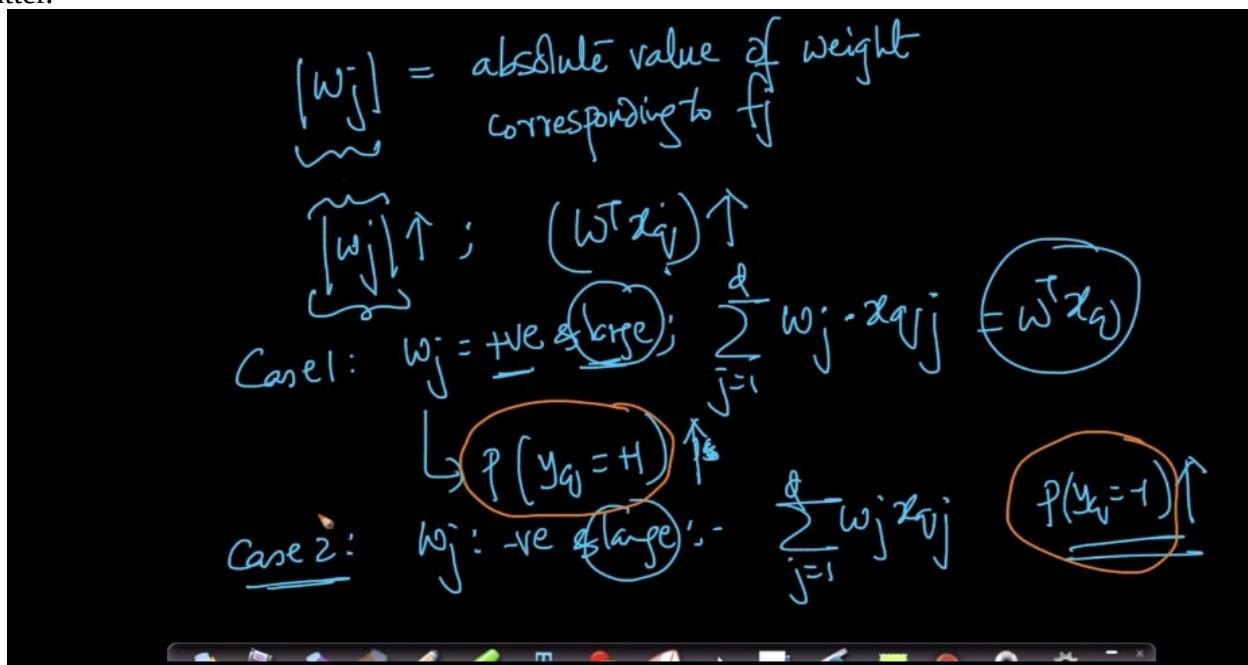
In K nearest neighbors feature importance can be found using forward feature selection.

In NB we can get the features importance by using the probability scores.

FFS is general technique.



A feature can be found using the absolute value of the weight, the positive and negative values does not matter.



We can determine the important features by determining each of the feature weights.

determine the important features in LR

(w_j)

e.g. predict gender: male (+1) & female (-1)

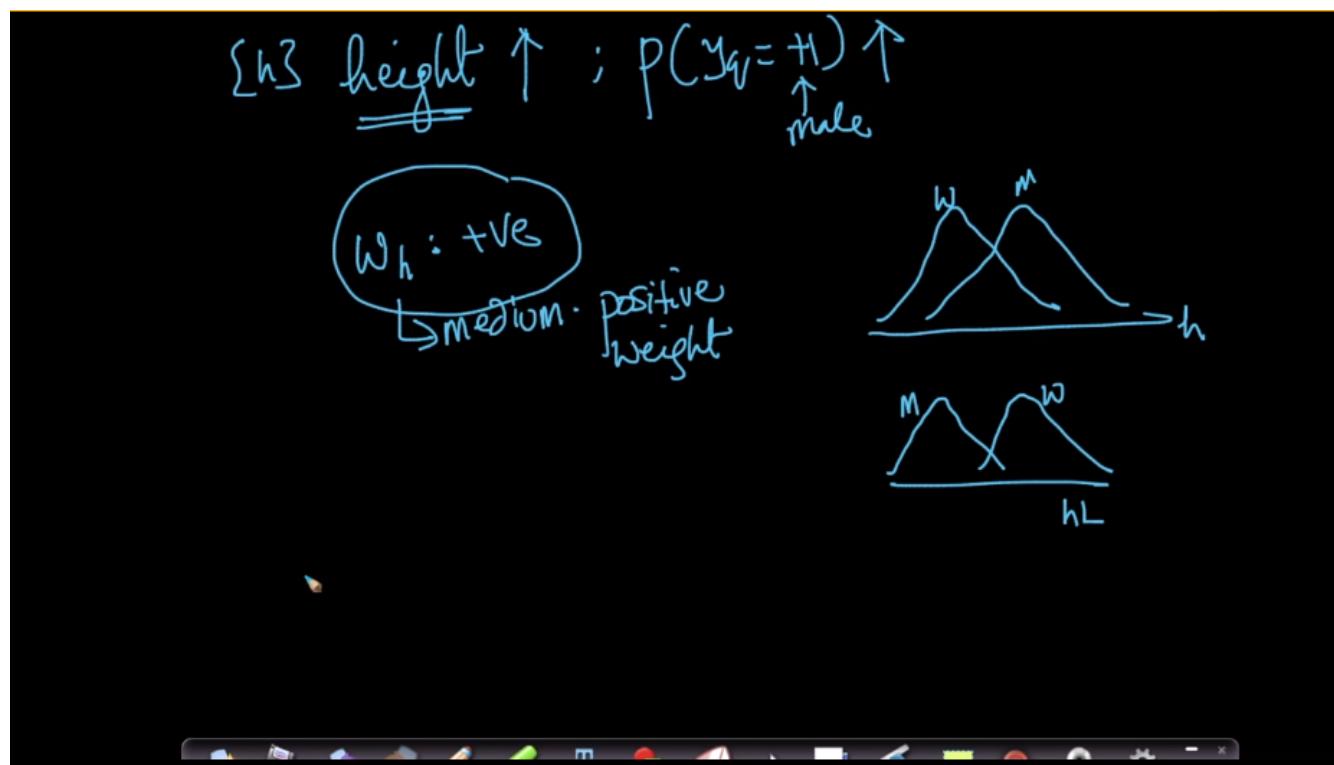
(w_{hL}) hair length: $|w_{hL}|$ is large

large negative weight ($w_{hL} \downarrow$) $P(y_g = -1) \uparrow$



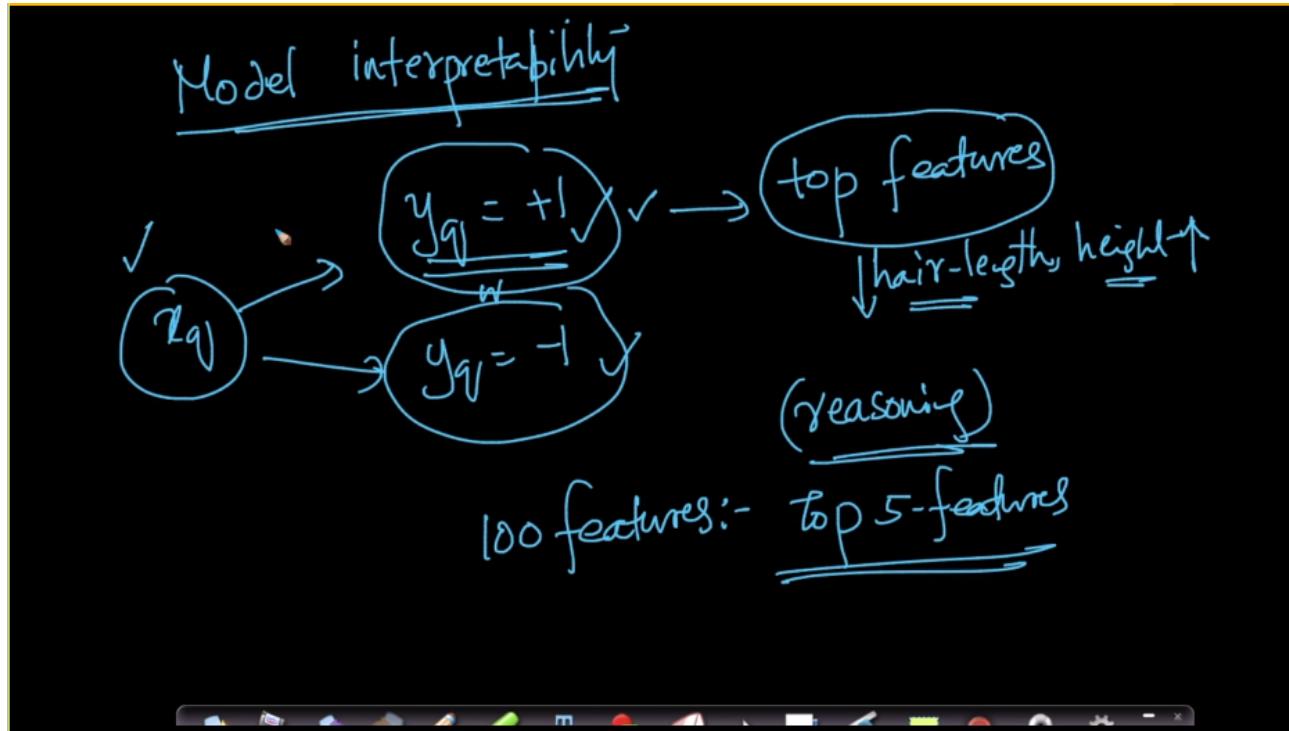
Females tend to have more hair length, negative weight will increase because the hair length of female is more.

For height we will get minimum weight to height.



Model interpretability:

The top features are concluded using more absolute value of the weight to the feature.

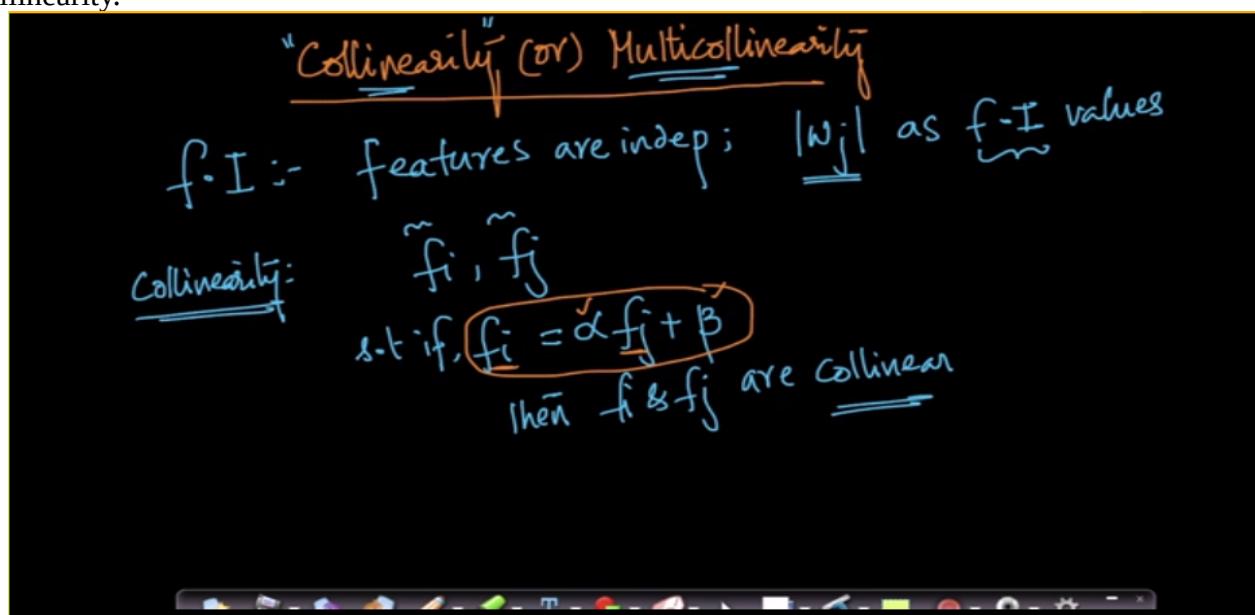


Col linearity (or) multicollinearity:

If features are independent we can use the absolute weight values for feature importance.

Features are not independent in collinearity and multi-collinearity.

Collinearity: If the features are represented in a linear relationship then this relationship is called collinearity.



Multicollinearity: In this several features are connected with a linear relationship.

Multicollinearity: if $f_1, f_2, f_3 \& f_4$ s.t
 $f_1 = \underline{\alpha_1} + \underline{\alpha_2}f_2 + \underline{\alpha_3}f_3 + \underline{\alpha_4}f_4$
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

$|w_i|$ is not useful for col-linear features, because there is a relationship between the features.

If there is a relationship between f_1 and f_2 then the function changes, with the relation of the features.

$$\mathcal{D} = \langle x_i, y_i \rangle_{i=1}^n$$
$$w^+ = \langle \underbrace{1, 2, 3}_{f_1, f_2, f_3} \rangle ; x_{qj} = \langle x_{q1}, x_{q2}, x_{q3} \rangle$$

$$w^T x_{qj} = x_{q1} + 2x_{q2} + 3x_{q3}$$

if $f_2 = 1.5f_1 \Rightarrow f_1 \& f_2$ are collinear

$$w^T x_{qj} = \underline{x_{q1}} + 3\underline{x_{q1}} + 3x_{q3} = 4x_{q1} + 3x_{q3} = \langle 4, 0, 3 \rangle$$

Since, the features are co linear the feature importance changes.

$\checkmark \quad w^t = \langle 1, 2, 3 \rangle$ f_3 is the most imp

$\checkmark \quad \tilde{w} = \langle 4, 0, 3 \rangle$ f_1 is the most imp

Same classifier $\therefore f_2 = 1.5f_1$

$x_q \rightarrow (w^T x_q)$

If features are collinear (or) multicollinear then the weight vector changes arbitrarily.

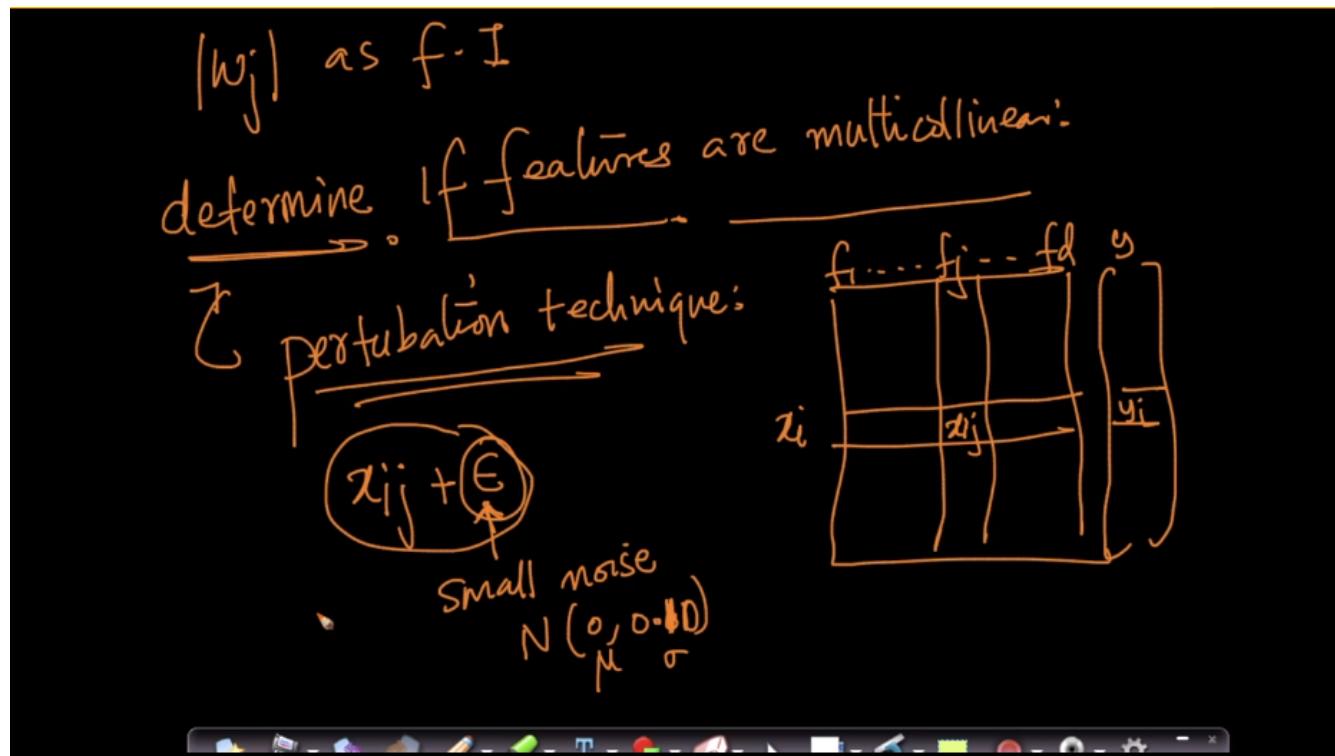
Conclusion:

if features are collinear
↓
weight vector can change arbitrarily.
↓
 $|w_j|$ cannot be used as f. I

Determining if features are multi collinear or not?

This can be determined by using perturbation technique.

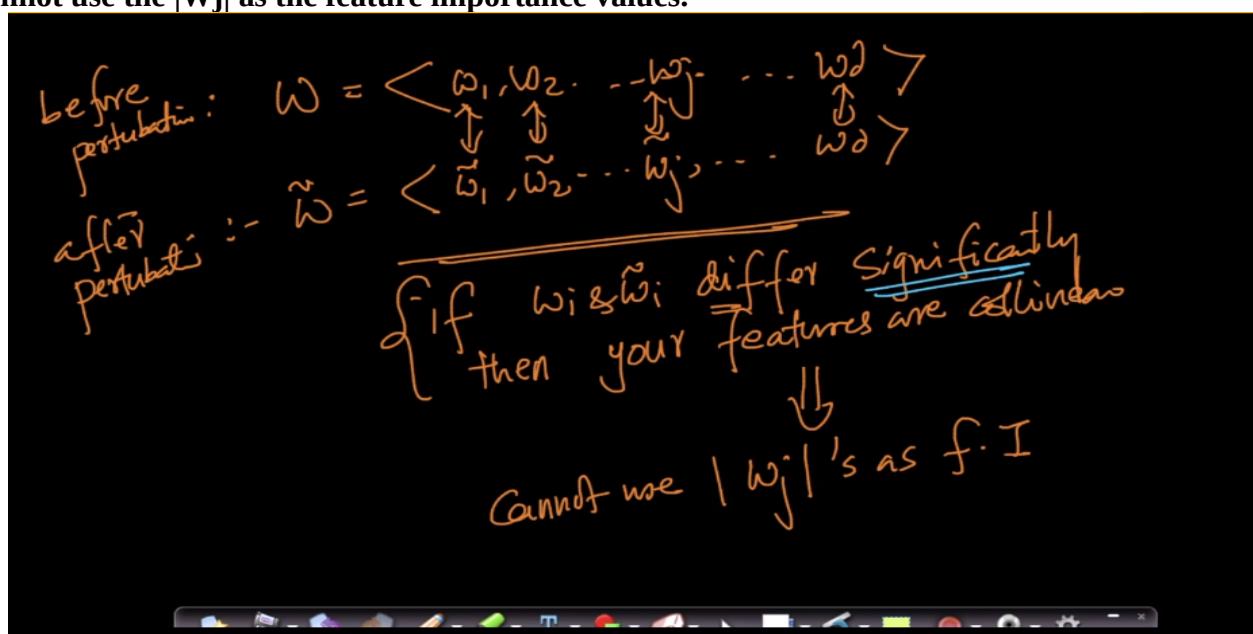
In this technique we will add small noise to each data point in the feature space.



Technique:

Before perturbation we will compute the weights of the feature W' vector and after the perturbation W'' .

If w' and w'' differ significantly then we can conclude that features are collinear, then we cannot use the $|w_j|$ as the feature importance values.



If weights cannot be used in case of col-linear features, THERE IS ALWAYS FORWARD FEATURE SELECTION technique that can be implemented.

Time and space complexity of Logistic regression:

We only need to store the weight vector which of size 'd'.

Train & Run time Space & time Complexity

$n = \# \text{ pts in } D_{\text{train}}$
 $d = \text{dim}$
 $O(nd)$

Train LR :- SGD $\hat{w}^t = \underset{w}{\operatorname{arg\min}} (\text{logistic loss} + \text{reg})$
 $\xrightarrow{\text{next chapter optimztn}}$

Run Time (LR)

$\left\{ \begin{array}{l} \text{Space: } O(d) \\ \text{Time: } O(d) \end{array} \right. \quad \sum_{j=1}^d w_j x_{qj}$

$w^t = \langle \underline{w_1}, \underline{w_2}, \dots, \underline{w_d} \rangle$

$\begin{cases} w^t x_q > 0 \rightarrow +ve \\ w^t x_q < 0 \rightarrow -ve \end{cases}$

If the 'd' is very small then the latency of the result is low for the query point.

if d is small ($d=30$)
 $\rightarrow LR$ is v.v.good for low-latency application

$x_q \rightarrow 1 \text{ ms}$ (y_q) ✓

favourite algo
internet
companies

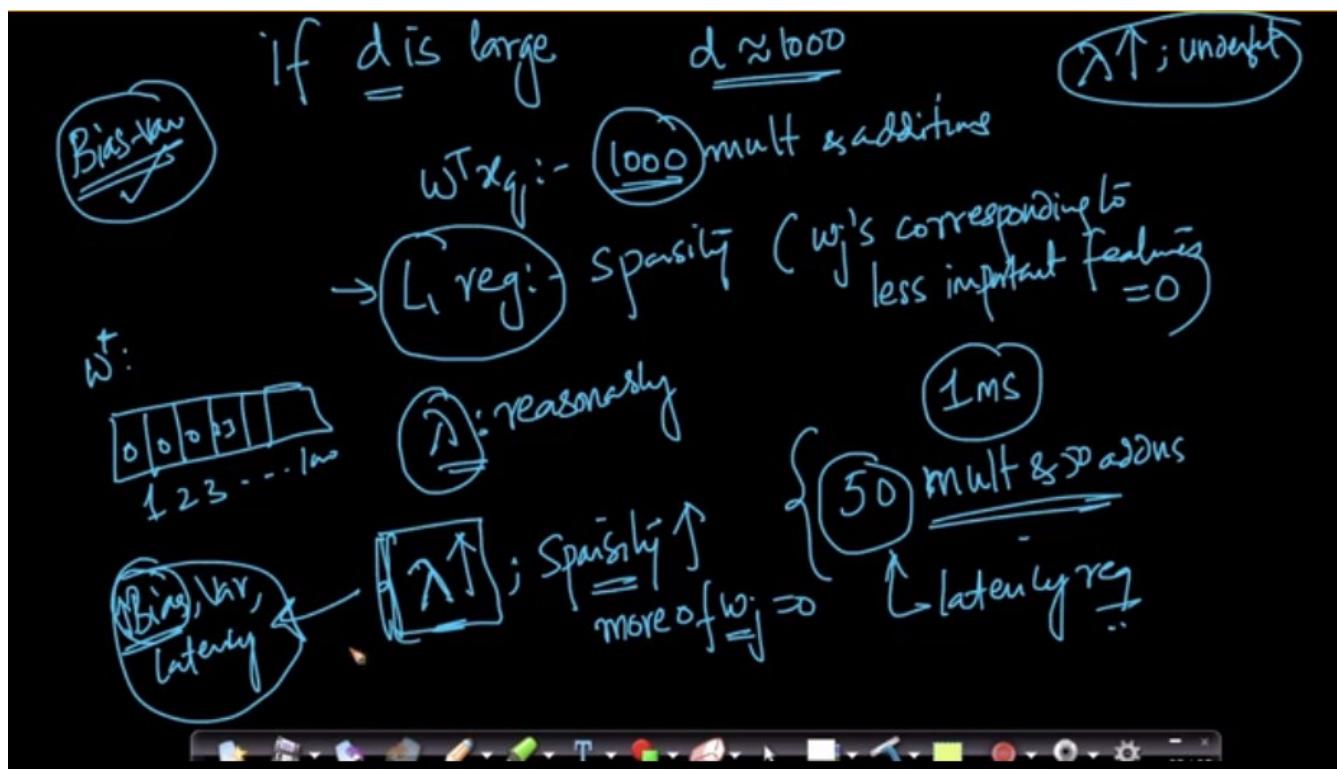
(30 multiplications)
+ 29 additions

\rightarrow memory efficient. $x_q \rightarrow \boxed{\quad} \rightarrow y_q$

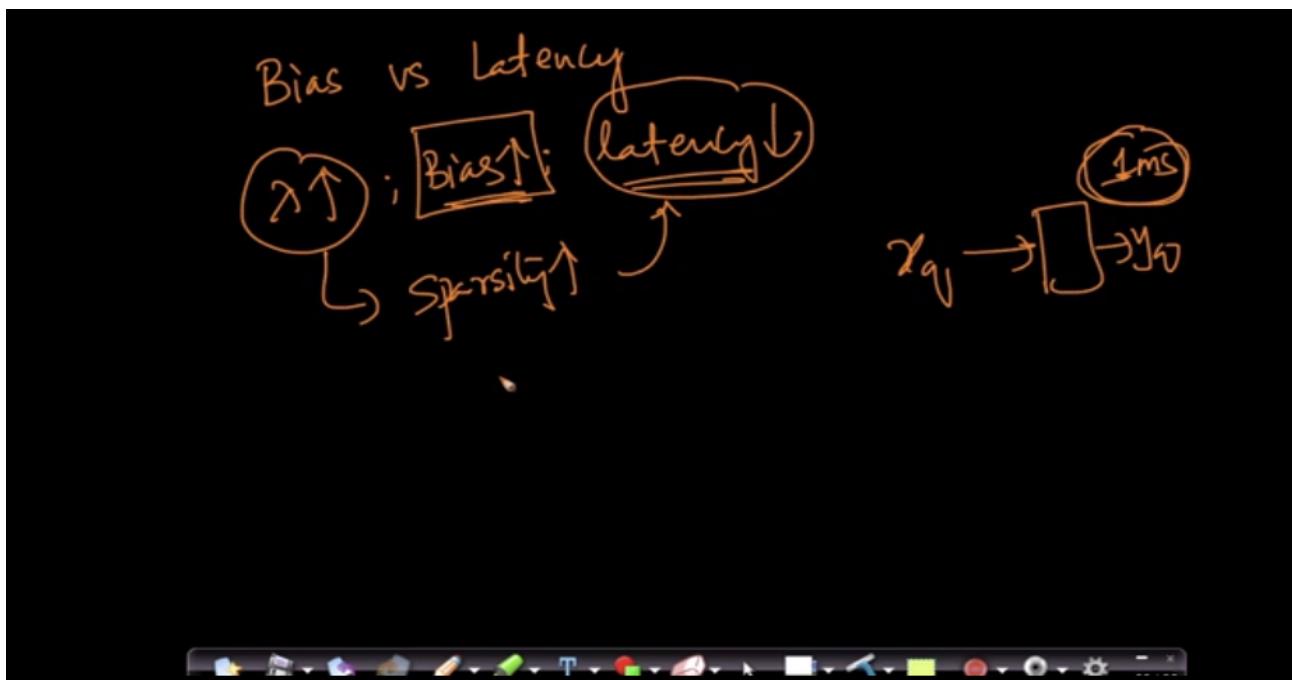
w^t

If 'd' is very large, we need 1000's of features. We use L1 regularization to make feature importance.

For using L1 regularization we need to take care of the sparsity, lambda, bias and variance for making the model.



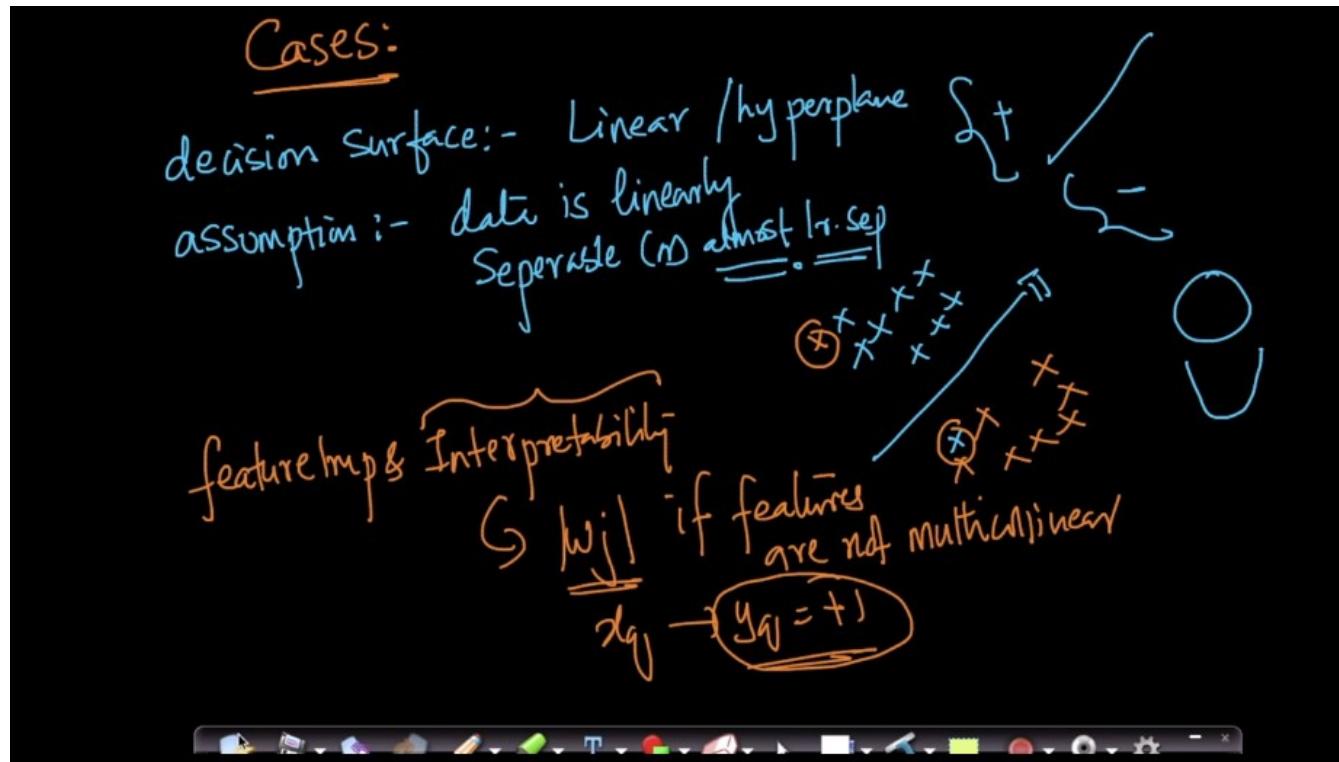
In some cases, having the working model is more important than the not working model.



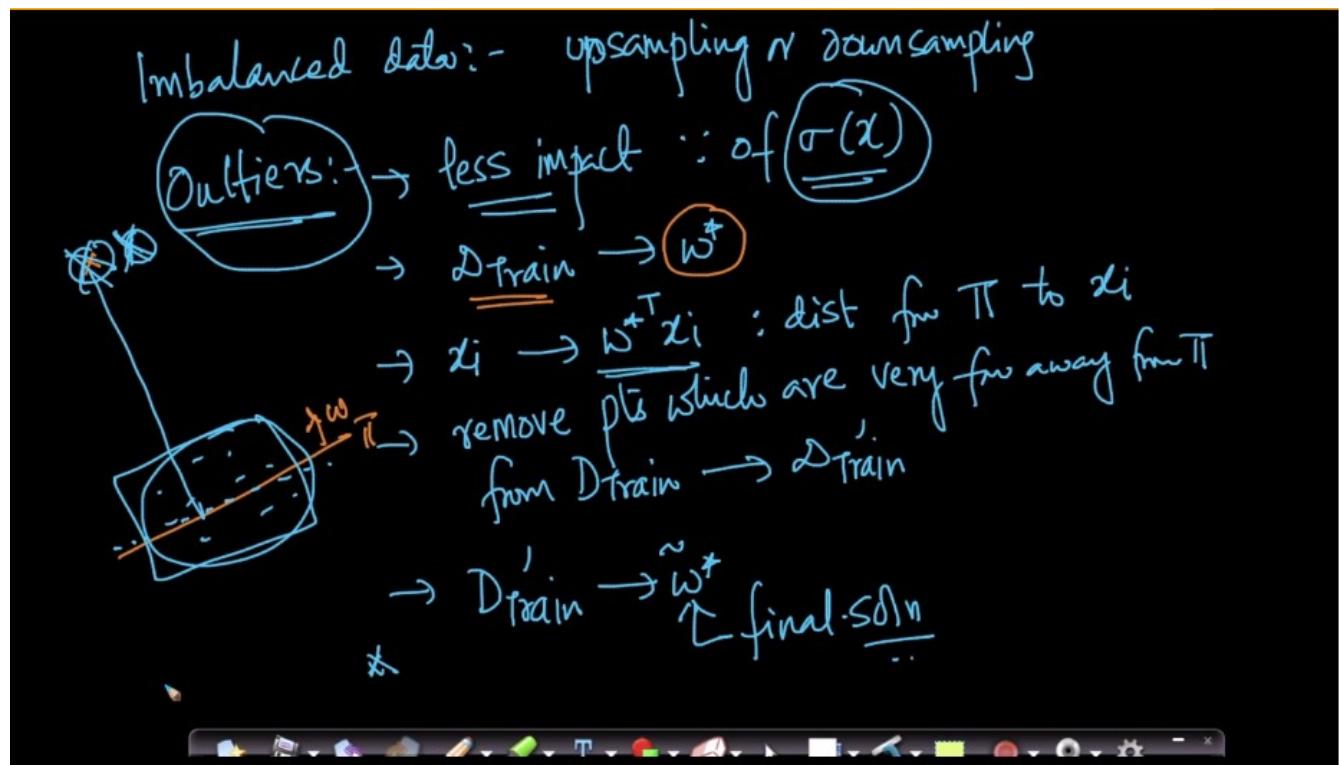
Real world Cases:

Decision surface: Linear / hyper-plane

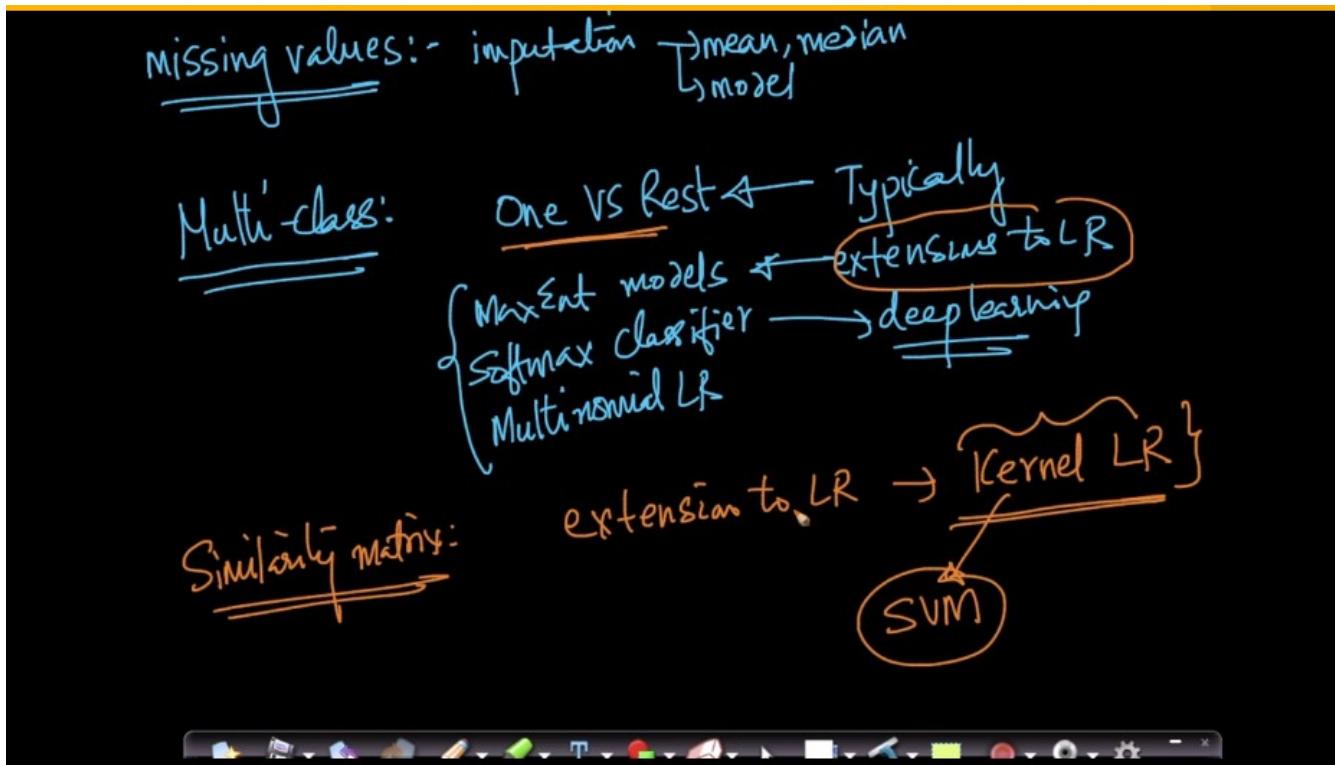
Assumption: Data is linearly separable (or) almost linearly separable.



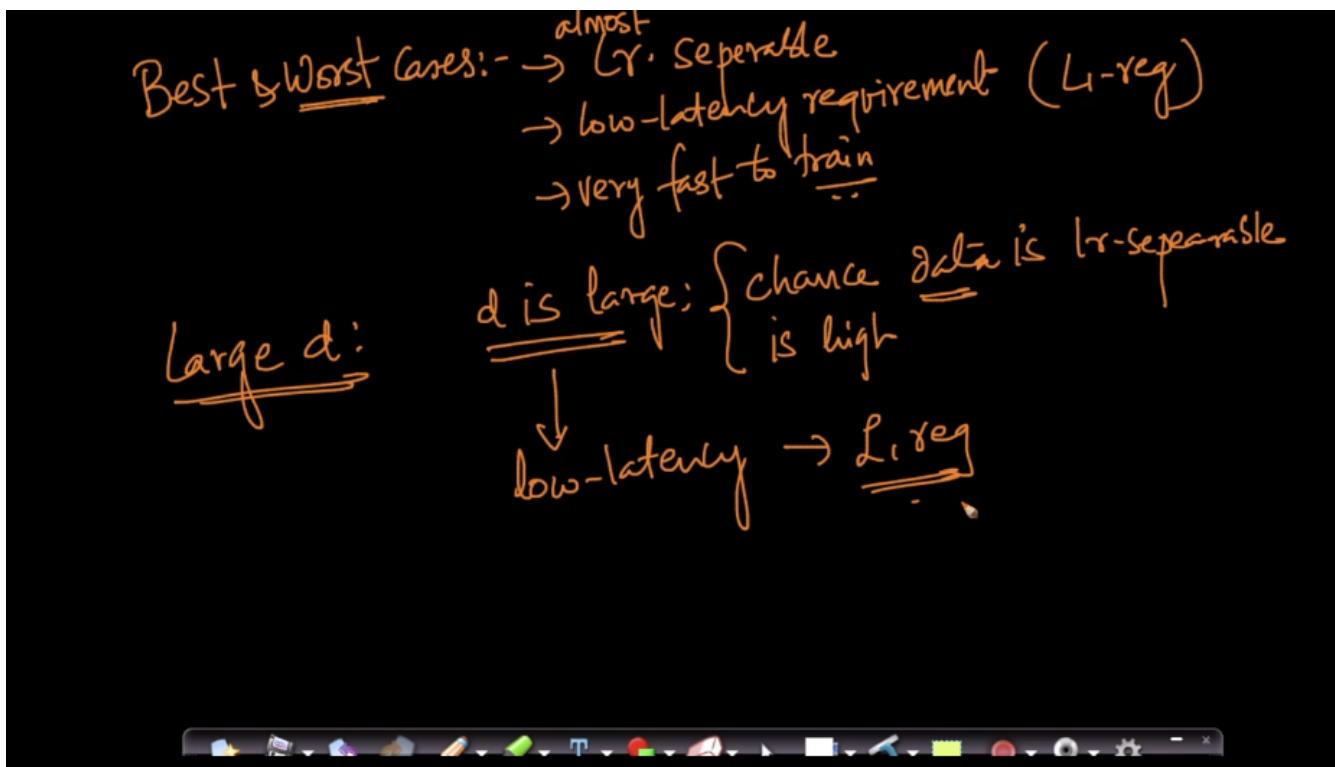
Imbalanced data: We use up sampling or down sampling.



Regular logistic regression does not support similarity matrix. We use Kernel LR.



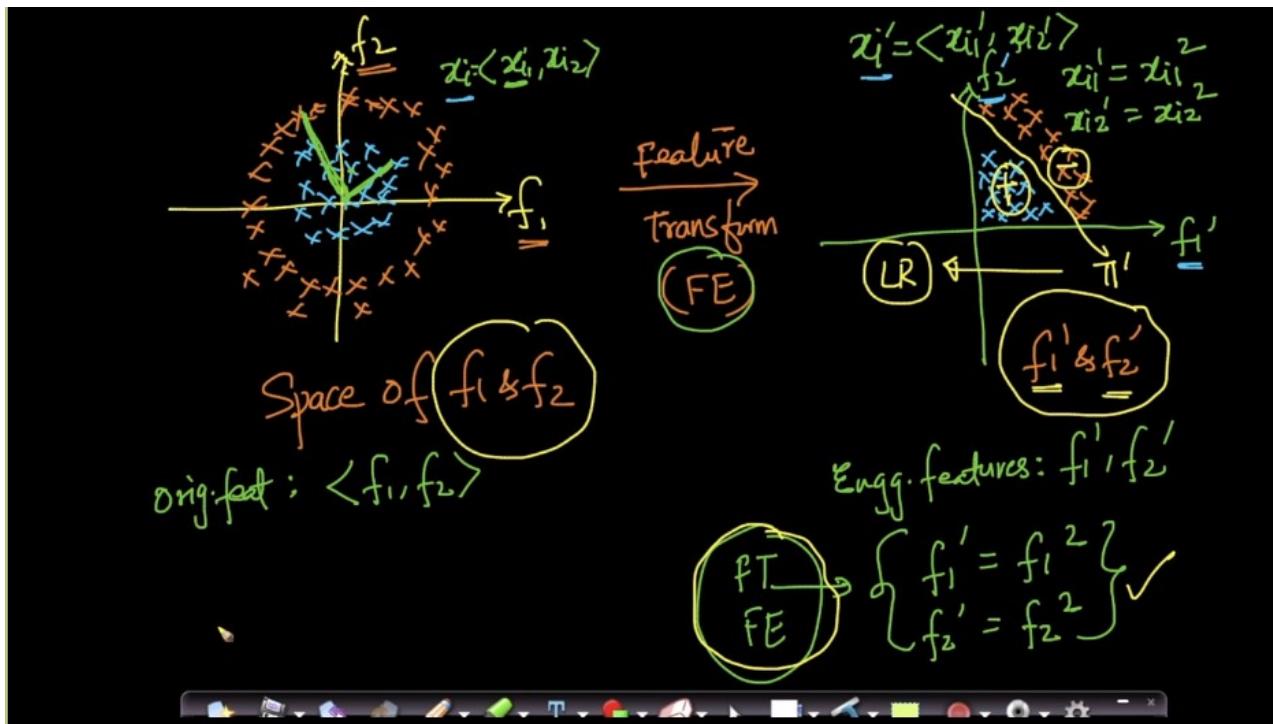
Best and worst cases:



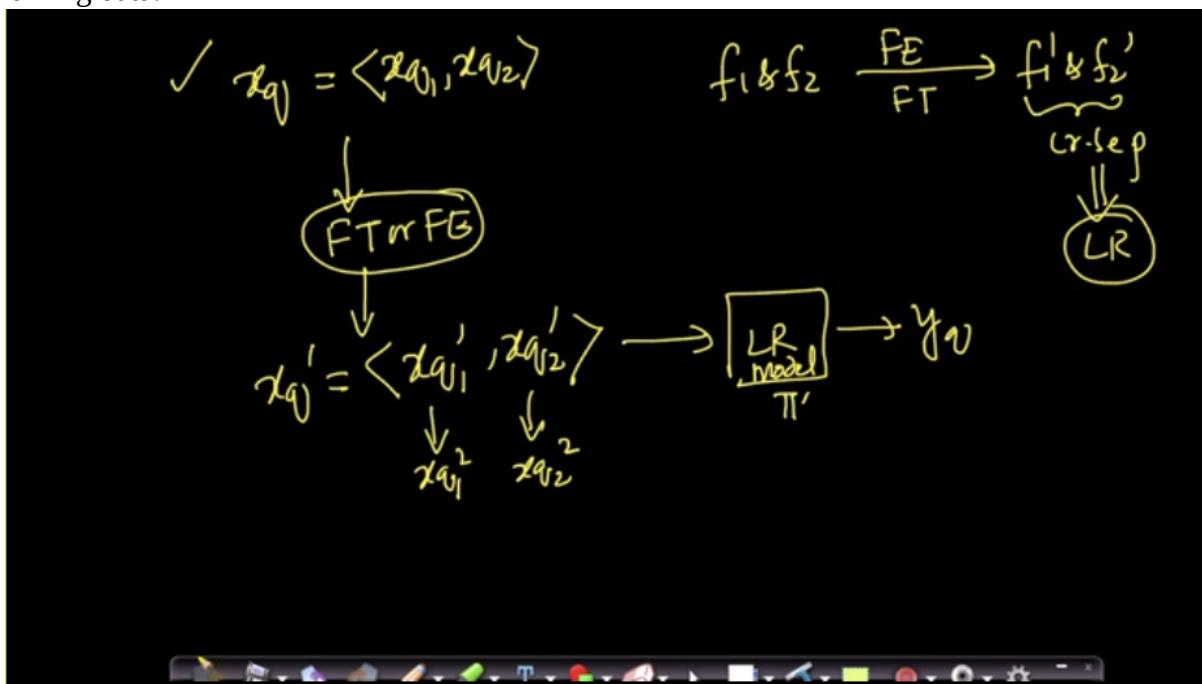
Non-linearly separable data and feature engineering:

Tweaking the data to some extent we can get the features that can be separable by logistic regression.

IMPORTANT: CASE - 1

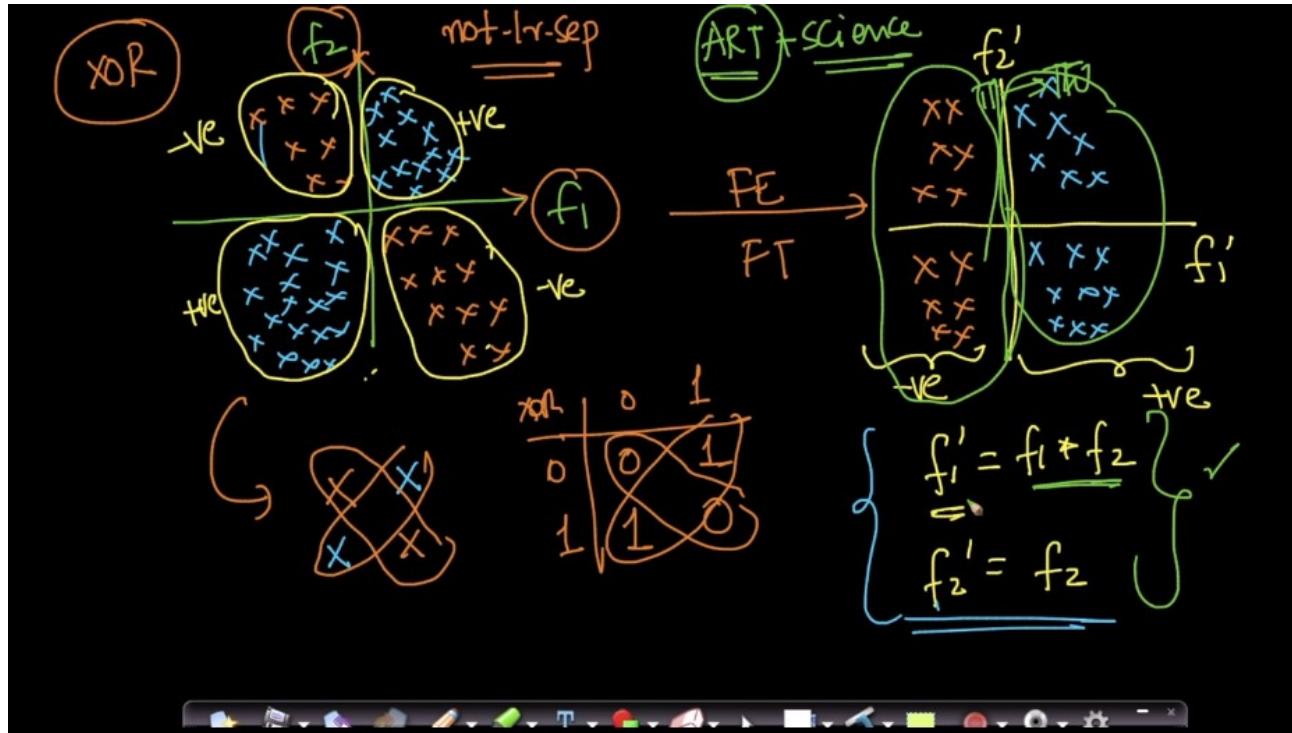


Even for the query point we will transform the features in the same way of transformation done to training data.

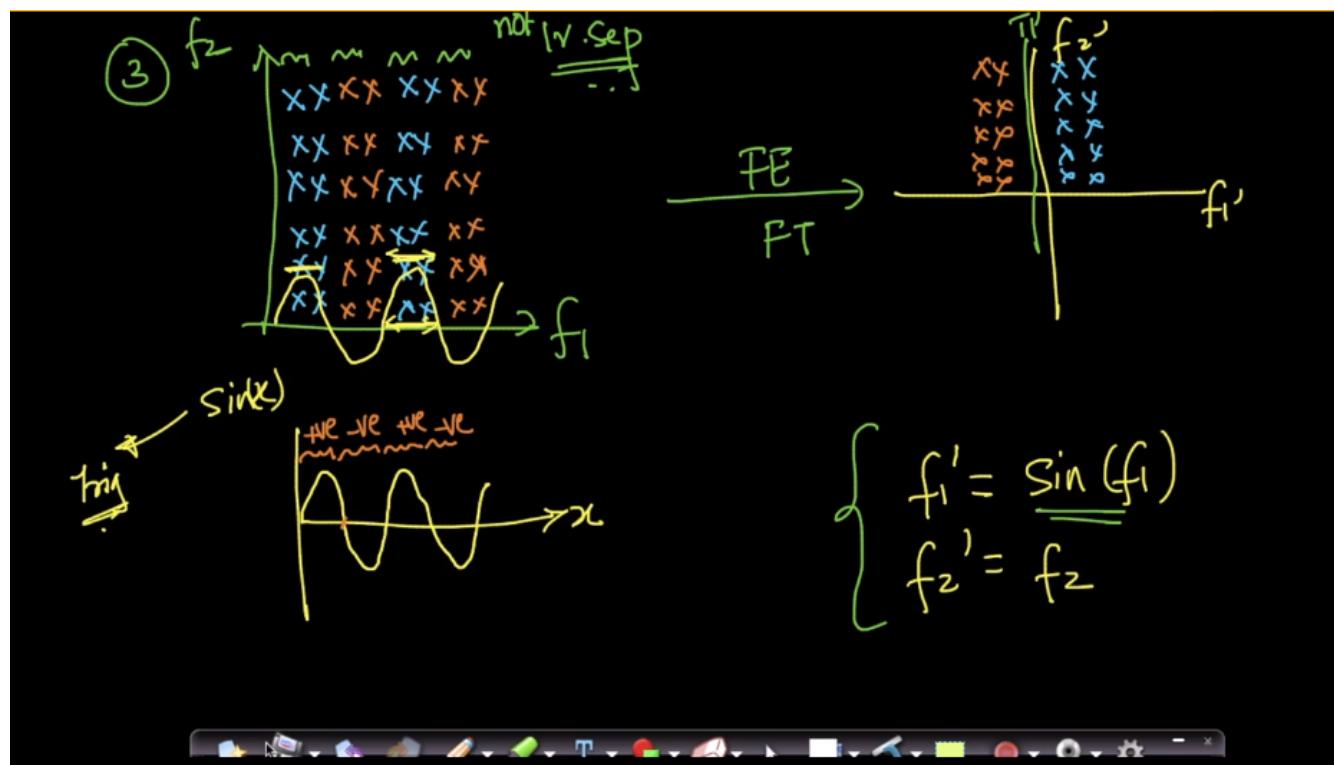


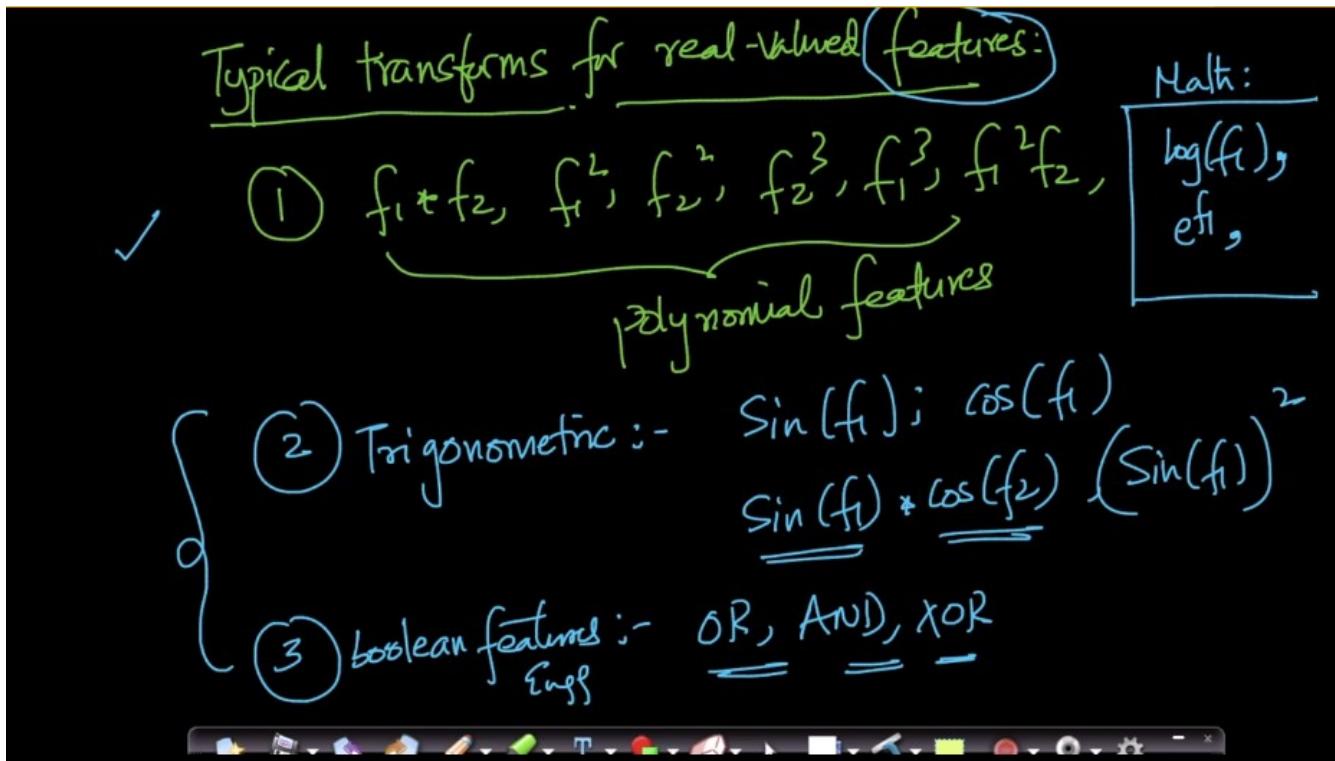
How to find which feature transformation to apply?
This feature transformation can only come from experience.

Another case: CASE - 2



CASE - 3





Sklearn implementation of Logistic regression:

scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

sklearn.linear_model.LogisticRegression

class sklearn.linear_model.LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag' and 'lbfgs' solvers. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted (and copied).

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty.

Read more in the User Guide.

Parameters: **penalty** : str, 'l1' or 'l2', default: 'l2'

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only L2 penalties.

New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

Exercise:

Ex: Logistic Regression on Amazon Reviews dataset:

$$\hat{\omega} = \arg \min_{\omega} \text{Log-loss} + \lambda \text{reg}$$

- ① Train, CV, Test
- ② GridSearchCV & RandomSearchCV to find optimal λ
- ③ L_2 reg & L_1 reg, \rightarrow high dimension data
- ④ L_1 reg; $\lambda \uparrow$; error \uparrow , sparsity = # non-zero elements in $\hat{\omega} \uparrow$

Generalized linear models:

Multi nominal distribution: It is for the multi class classification.

Linear regression: probabilistic assumption.

Poisson regression: Poisson distribution- How many visitors will visit the restaurant.

