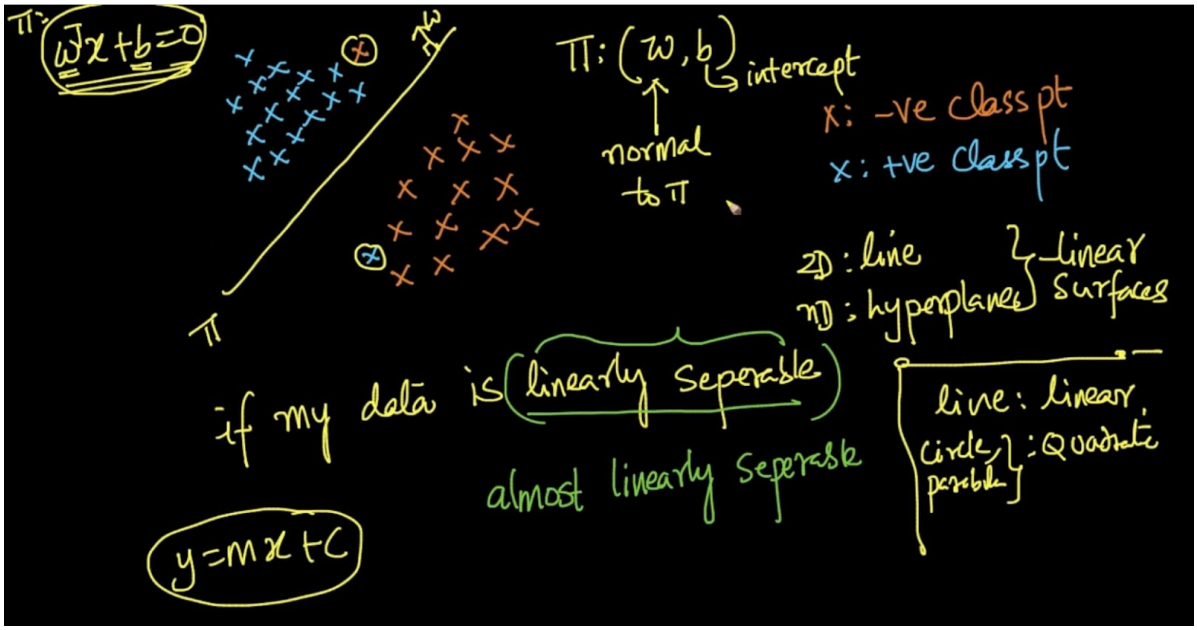
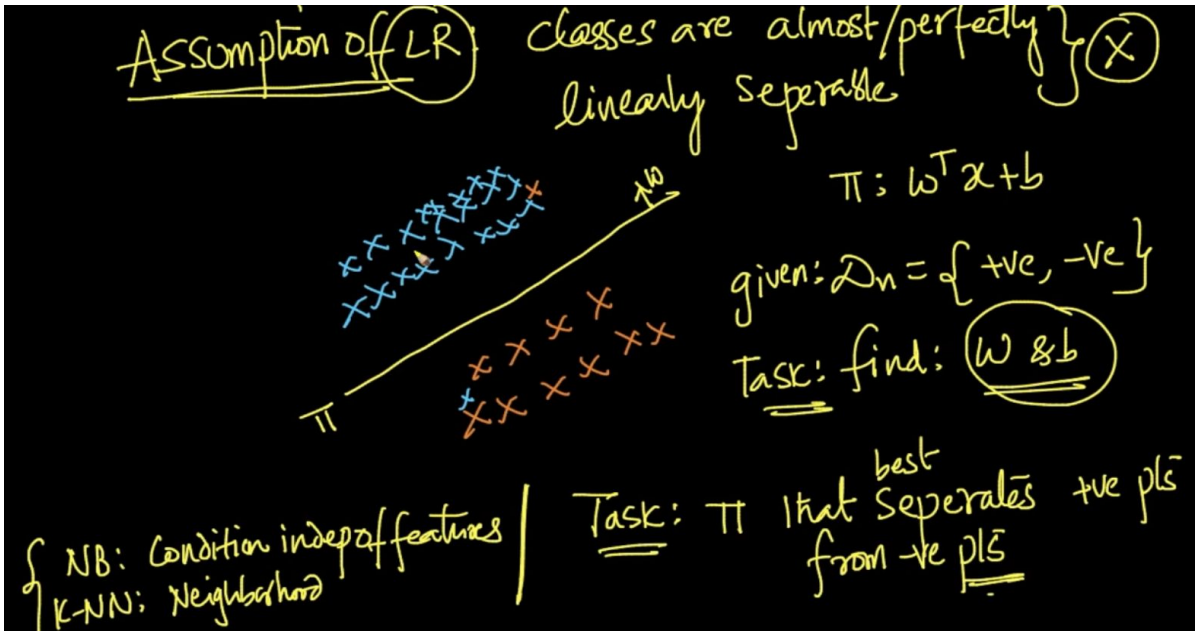
LOGISTIC REGRESSION

GEOMETRIC INTUITION

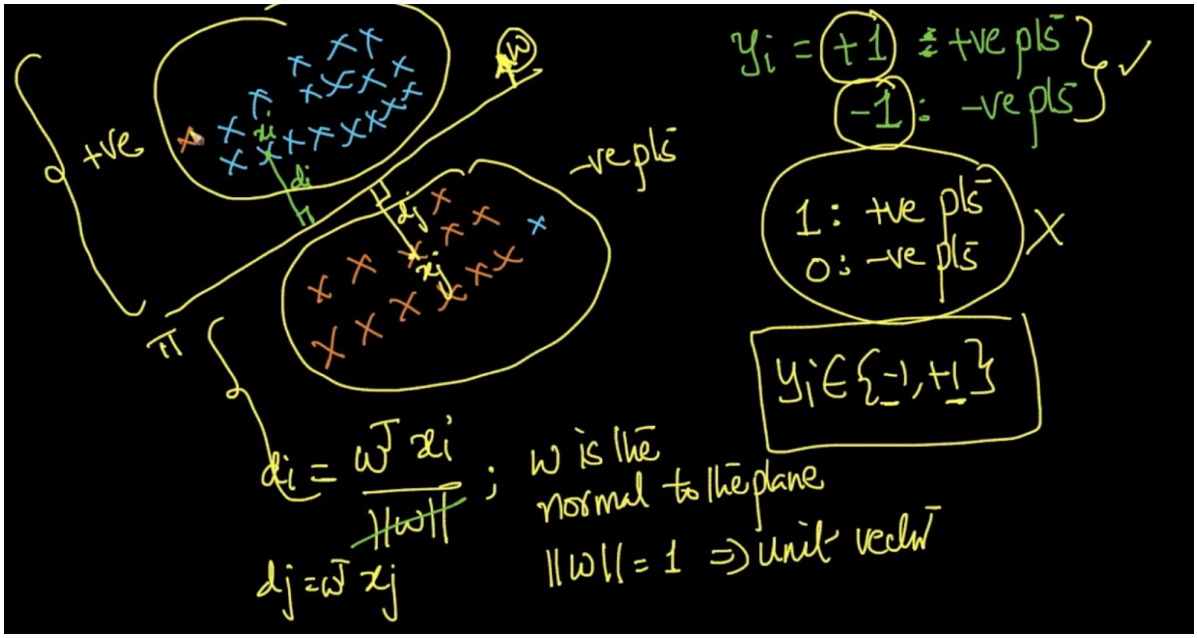


We need to find the hyperplane which makes the data linearly separable.

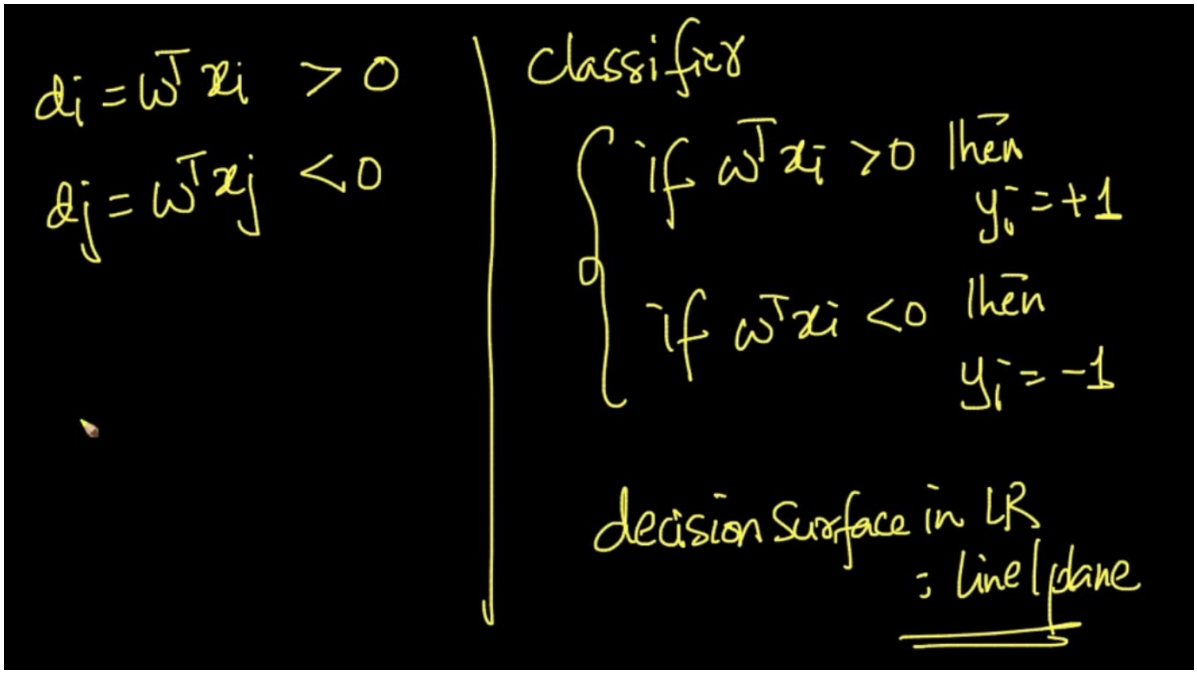
Equation of line/ hyperplane $\pi: w^T x + b = 0$



Assumption of Logistic Regression is that our data is perfectly linealrly separable. So, our task is to find the hyper-plane π that seperate both the +ve and -ve points.

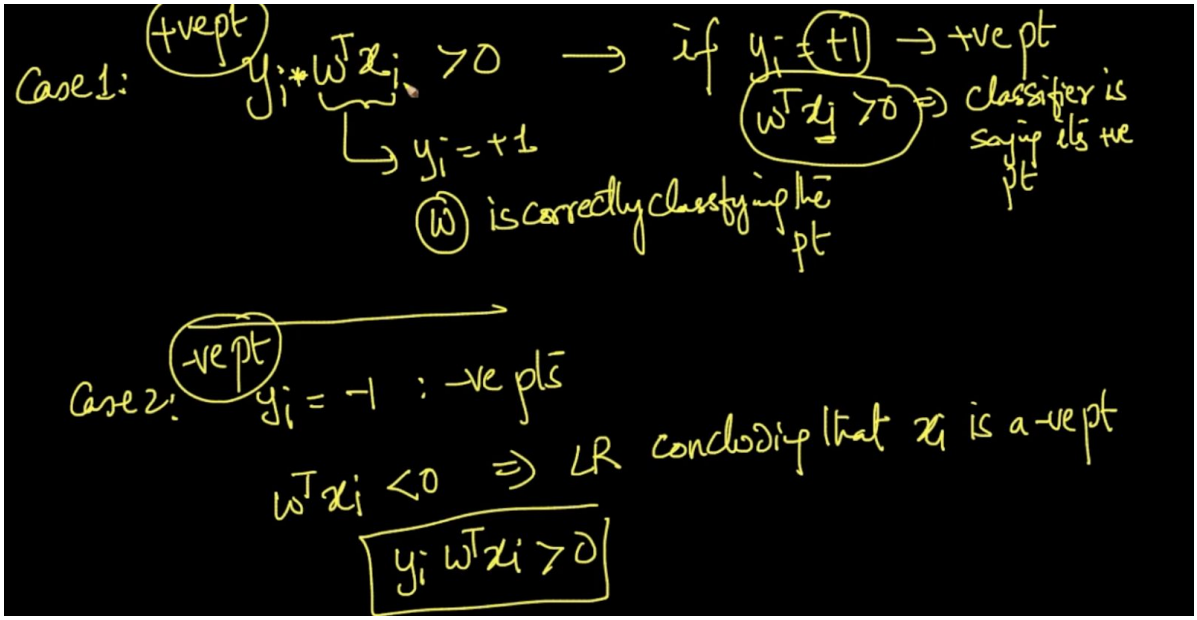


We are calculating distance d_i of a point x_i which is $d_i = w^T x_i$. We've labeled all the blue points as +ve points and the orange ones are -ve points.



Point x_i lies in the +ve region and the direction of normal w is also at +ve points direction

So , $w^T x_i > 0$. Hence we built a classifier saying that *if* $w^T x_i > 0$ *then* $y_i = +1$ *elif* $w^T x_i < 0$ then $y_i = -1$. Thats why we wanted an ideal plane π which is decision surface.



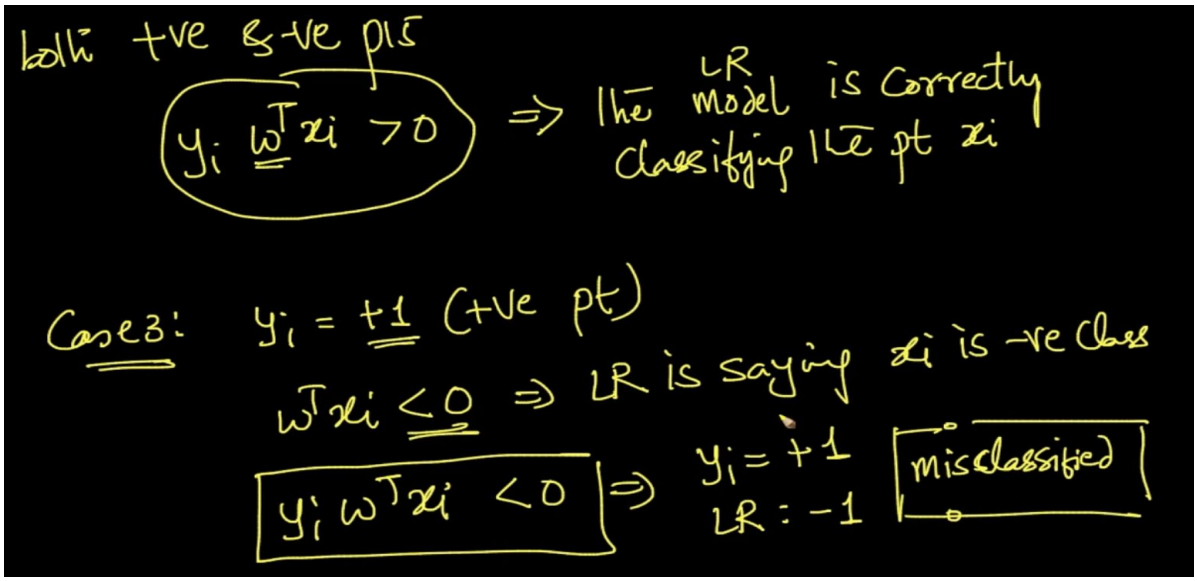
Case 1 : if y_i = +1 and $w^T x_i$ > 0 then the classifier is saying x_i is a +ve point. So, if

$y_i * w^T x_i$ > 0 and y_i = +1 then we can say that our normal **w** is correctly classifying the point

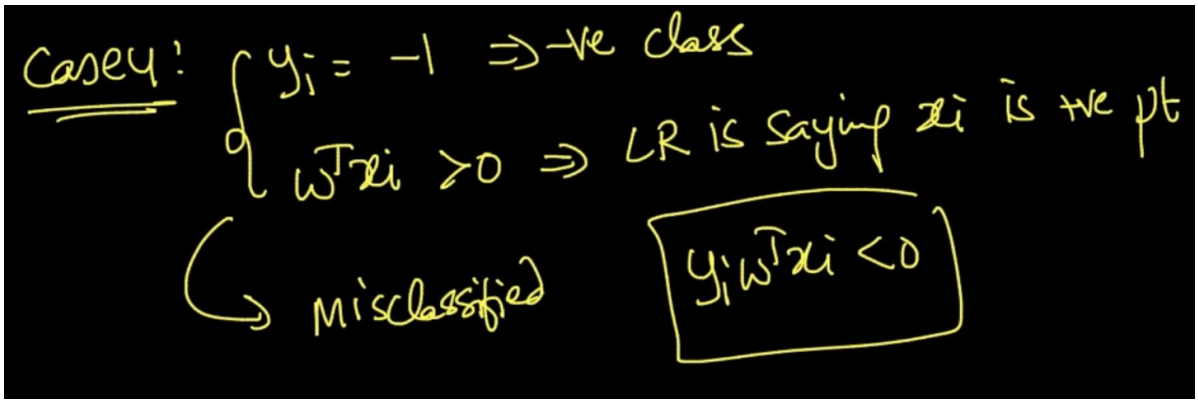
Case 2 : Same as case 1 except that case 2 is for -ve point . Here also $y_i * w^T x_i$ > 0.

NOTE : Both the cases are of **w** correctly classifying data points and in both $y_i * w^T x_i$ **> 0**

So, we want $y_i * w^T x_i$ greater than 0 to make sure that our classifier is working correctly

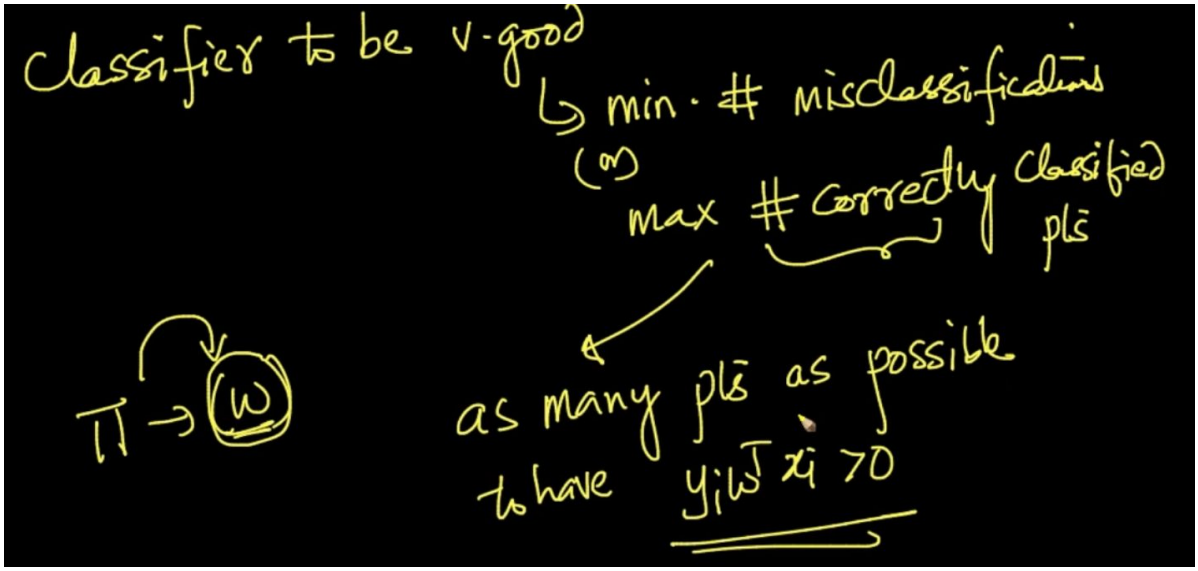


Case 3 : Here y_i = +ve point but $w^T x_i$ < 0 so $y_i * w^T x_i$ < 0 that means our Logistic Regression is misclassifying the data as LR : -1. Ex: Blue point in the orange points.

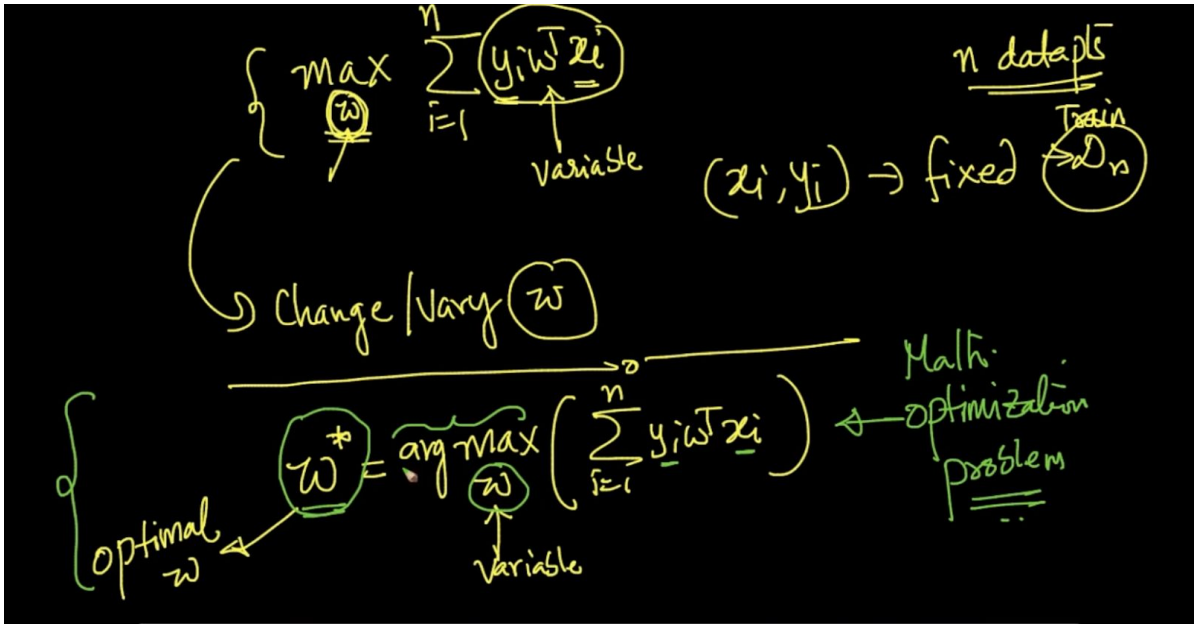


Case 4: Same as case 3. Here also $y_i * w^T x_i$ < 0.

NOTE : Case 3 and Case 4 both the cases are of **w** incorrectly classifying data points and in both $y_i * w^T x_i$ **< 0.** So, we want $y_i * w^T x_i < 0$ points to be less than 0 to make sure that our classifier is working correctly.



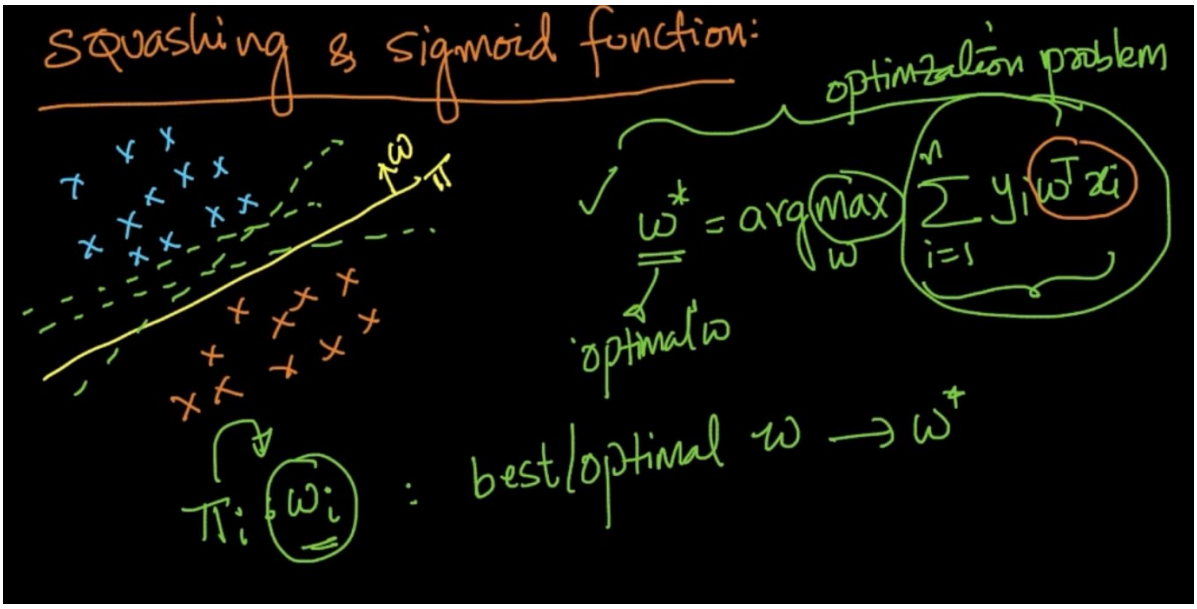
So in order for our classifier to be good we want as many points as possible to have $y_i * w^T x_i$ **> 0**



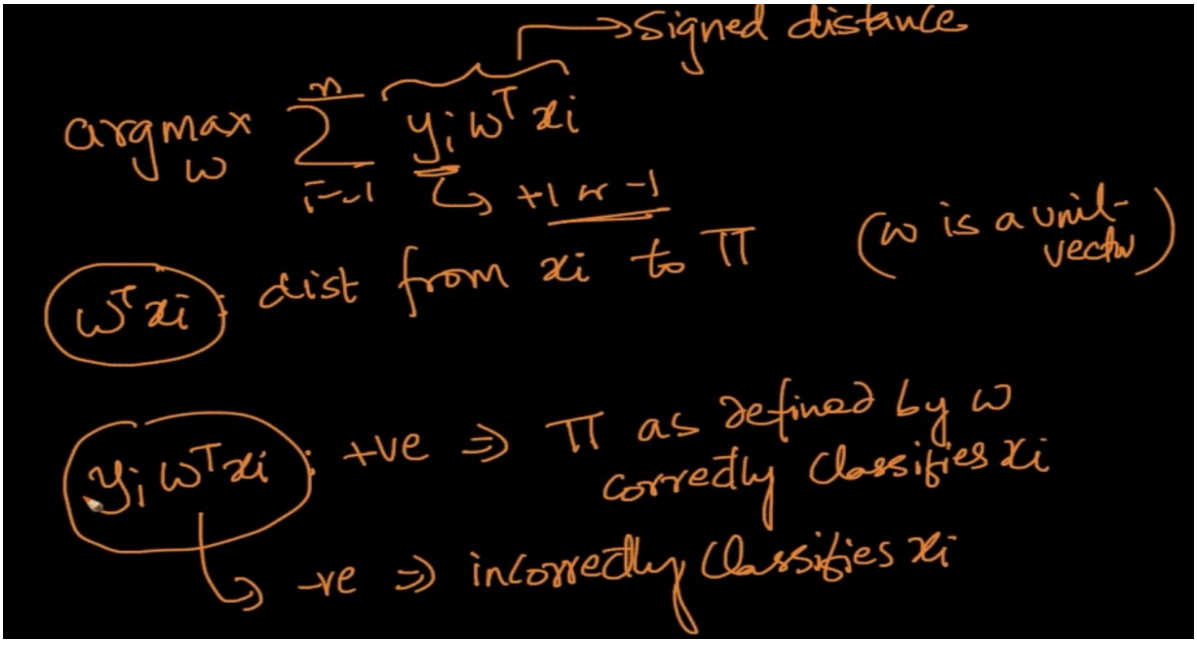
In this our x_i and y_i are fixed so the only thing that can be varied in $y_i * w^T x_i$ is w^T .

So, our goal is to find the optimal w => w^* which gives us maximum in summation of $y_i * w^T x_i$ as more $y_i * w^T x_i > 0$ means more correctly classified data. Our maximum of summation gives that . So finding this optimal w is the crux of this algorithm called Logistic Regression

SIGMOID FUNCTION SQUASHING

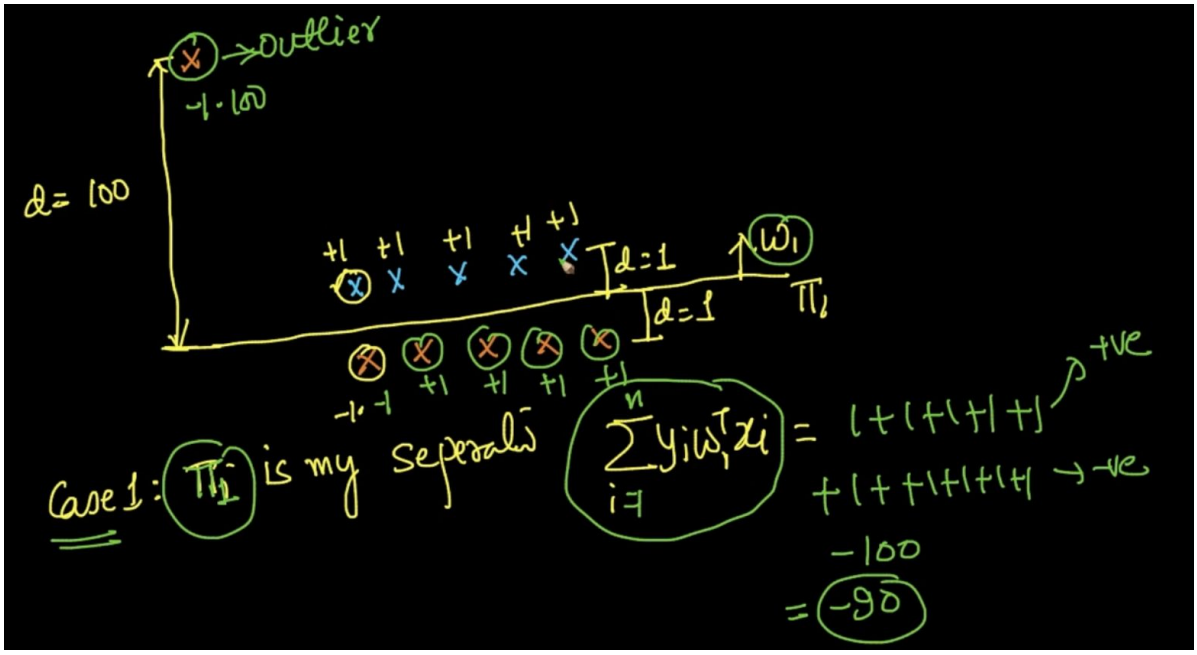


We need to find the optimal w -> w^*



y gives us sign of the point +1 or -1. $w^T x_i$ is the distance of data point x_i from plane π

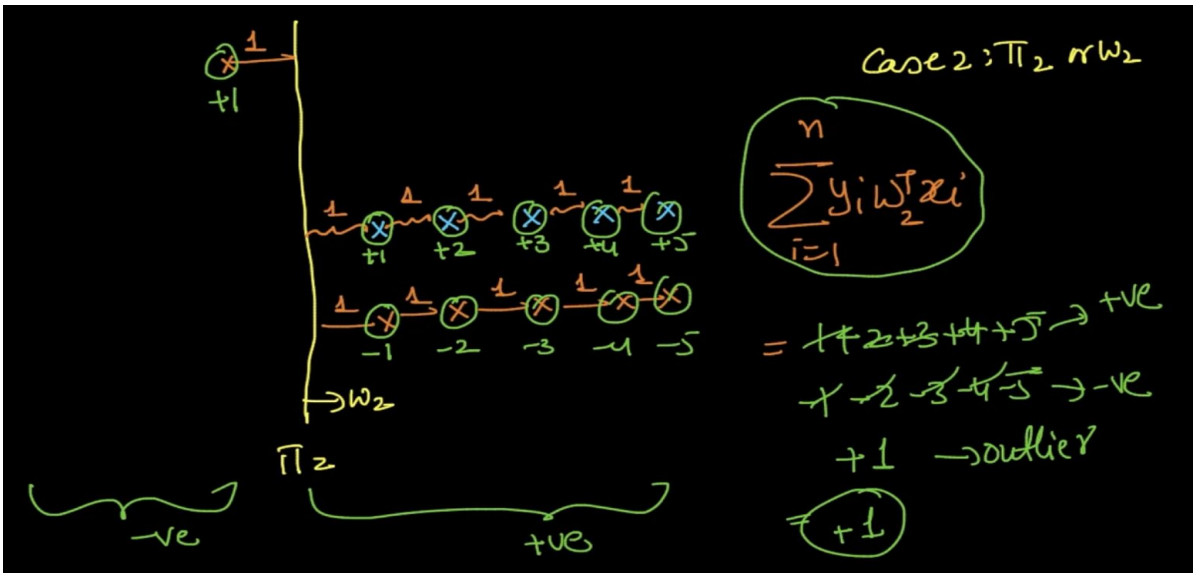
If $y_i w^T x_i$: +ve then our π defined by w correctly classifies x_i. So we want it to be maxed



Case 1: Here for +ve points(blue) distance d => $w^T x$ = 1 and y = +1 since positive

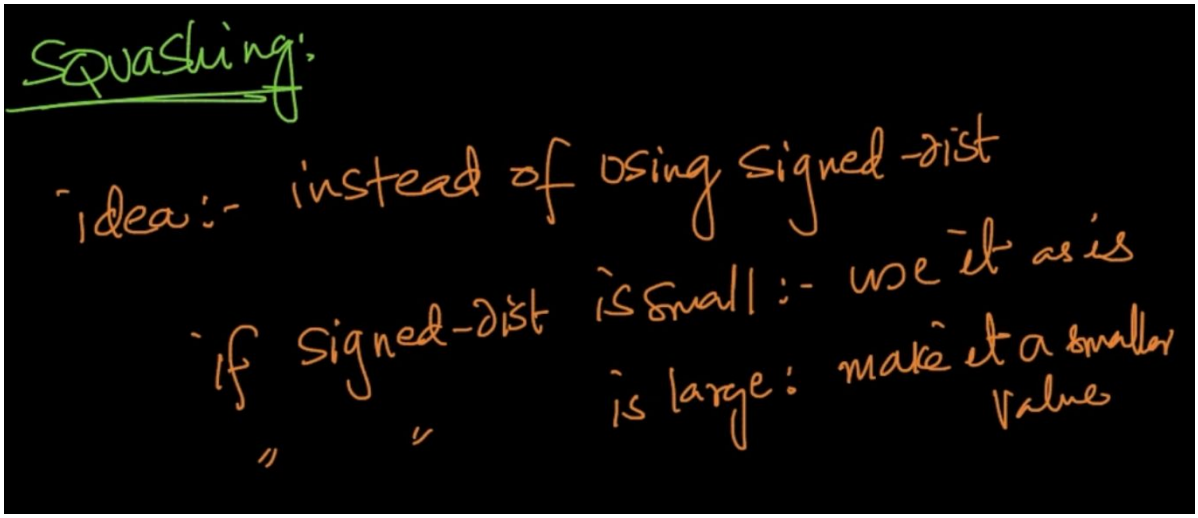
Here for -ve points(orange) distance d => $w^T x$ = -1 since opposite to direction of w and y = -1 since negative. For the outlier d = +100 since in the direction of w and -ve point

Therefore y = -1. $\sum_{i=1}^{n} y_i w^T x_i$ = -90

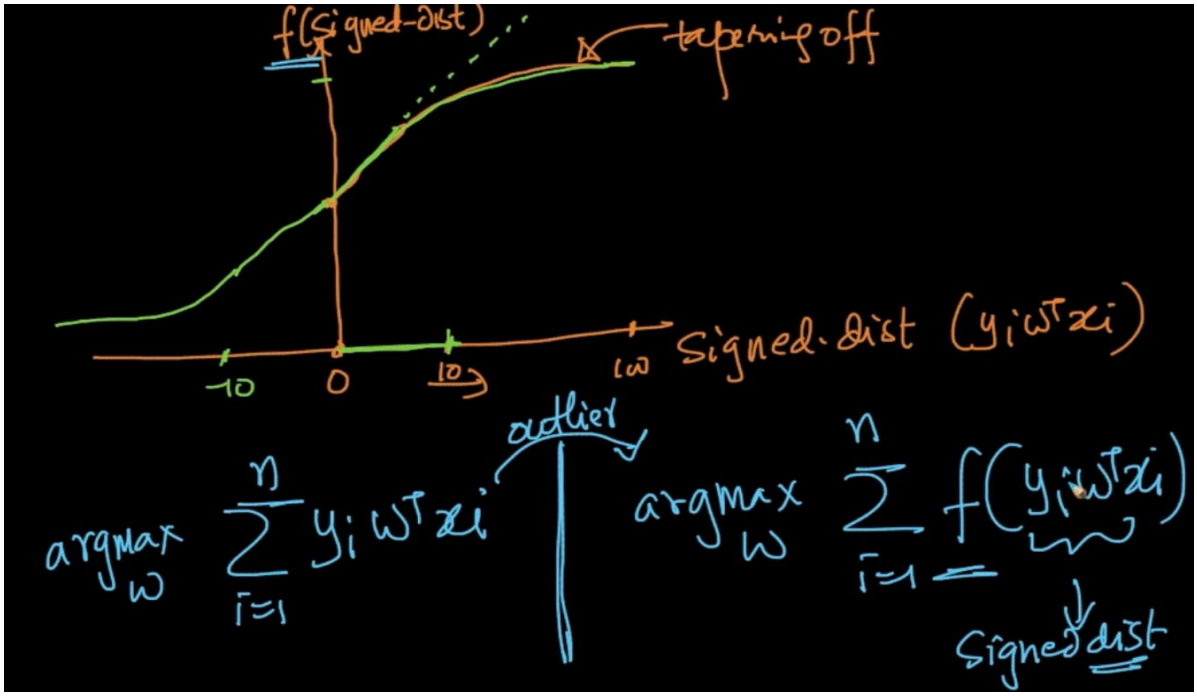


Case 2: Here the plane is separating the outlier and points. Here $\sum_{i=1}^{n} y_i w^T x_i$ = +1

We want $\sum_{i=1}^{n} y_i w^T x_i$ to be maxed so our model will say Case 2 but we know intuitively that the plane in Case 2 is doing a shit job as it's misclassifying more data. It's happening because of our outlier . So our max $\sum_{i=1}^{n} y_i w^T x_i$ isn't prone to outliers



So to deal with it we do squashing. It means if the distance is small use it as it is but if it's large make it a smaller value

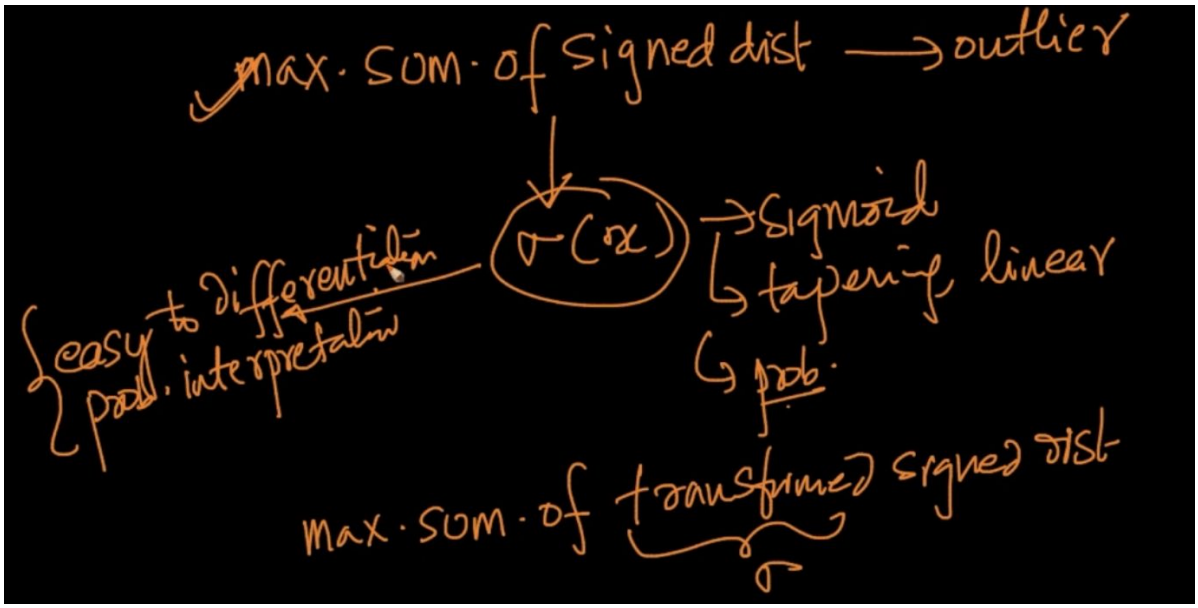


To get rid of the outlier we need a function such that after a certain distance ,10 here it'll taper off. That's max f($\sum_{i=1}^{n} y_i w^T x_i$). In the above example our outlier with distance 100 is changing our whole model so it should be tapered down . So we need to find that function

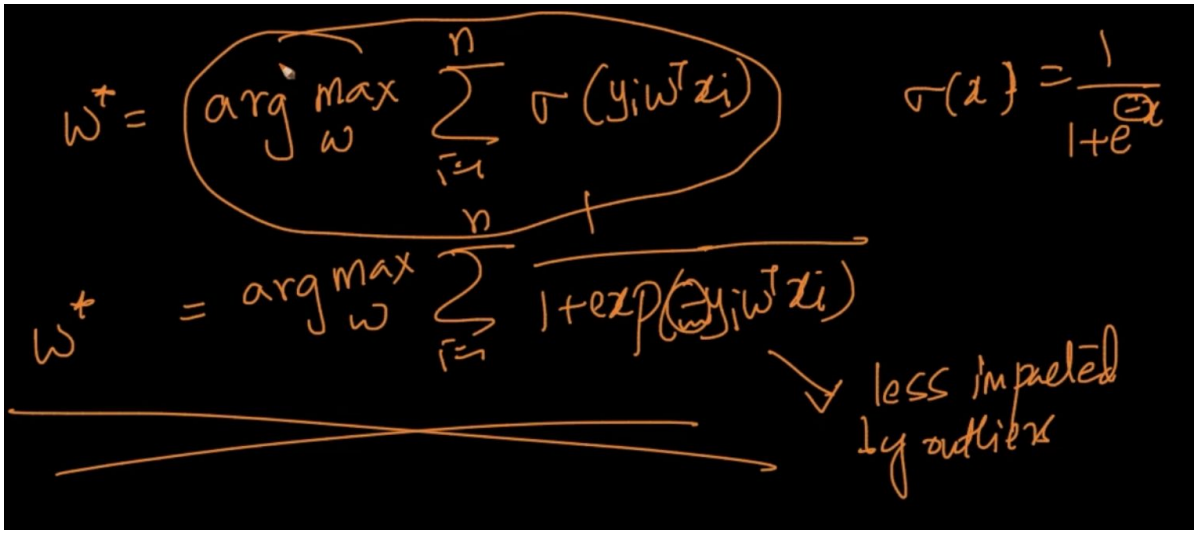
f

One such function is sigmoid function. We can see that till a certain value it's increasing linearly but after that it's getting tapered down and it's maxed value is 1 so if our x: signed distance > 3 here it's y will be 1

This sigmoid function has nice probabilistic interpretability properties . For ex: If the point is at plane then $w^T x_i$ = 0 therefore P($y_i = 1$) = 0.5 because we don't know whether it's +ve or -ve . If the distance $w^T x_i$ is large say 12 here our P($y_i = 1$) = 1.

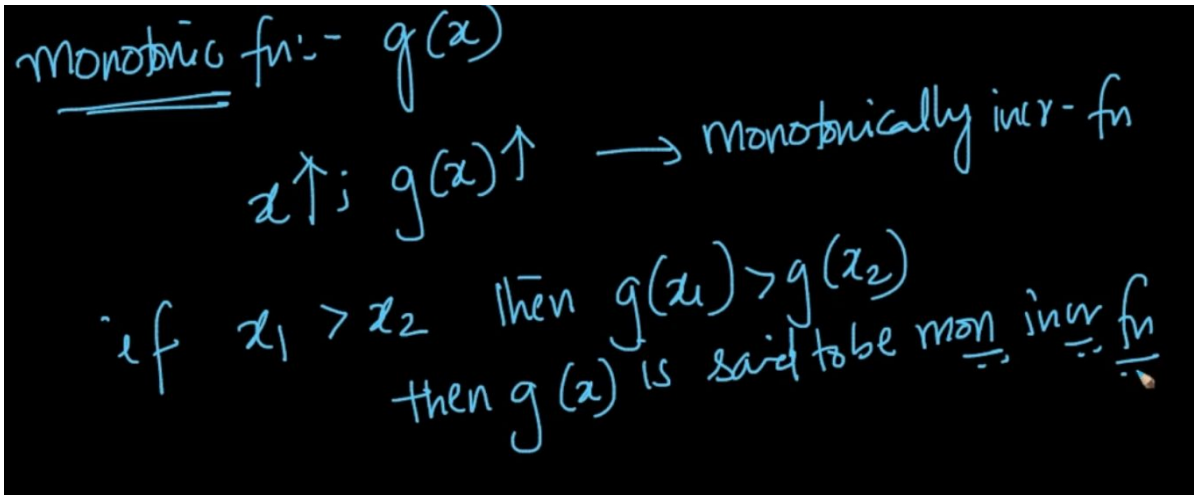


We wanted max sum of signed distance but it'd problem of outlier so we are using a function $\sigma(x)$ which has some nice properties mentioned above. So we want max sum of transformed signed distance with σ.

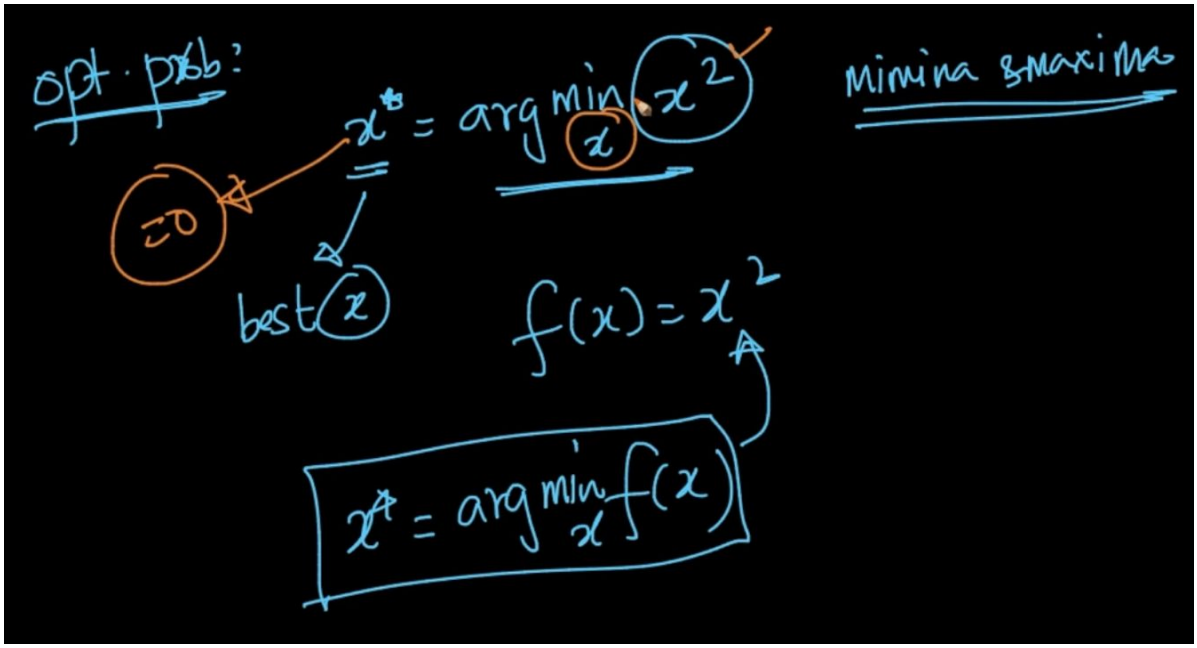


So our goal is to find optimal w : w^* = argmax $\sum_{i=1}^{n} \frac{1}{1+ exp(-y_i w^T x_i)}$

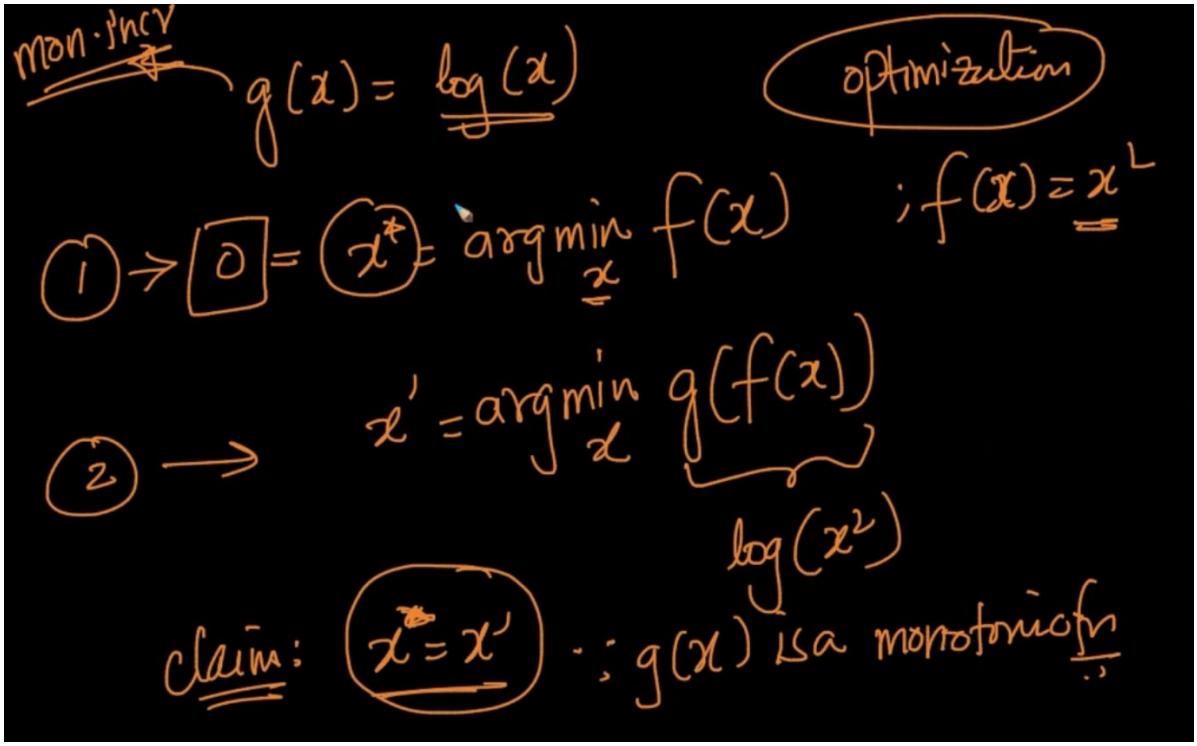
MATHEMATICAL FORMULATION OF OBJECTIVE FUNCTION



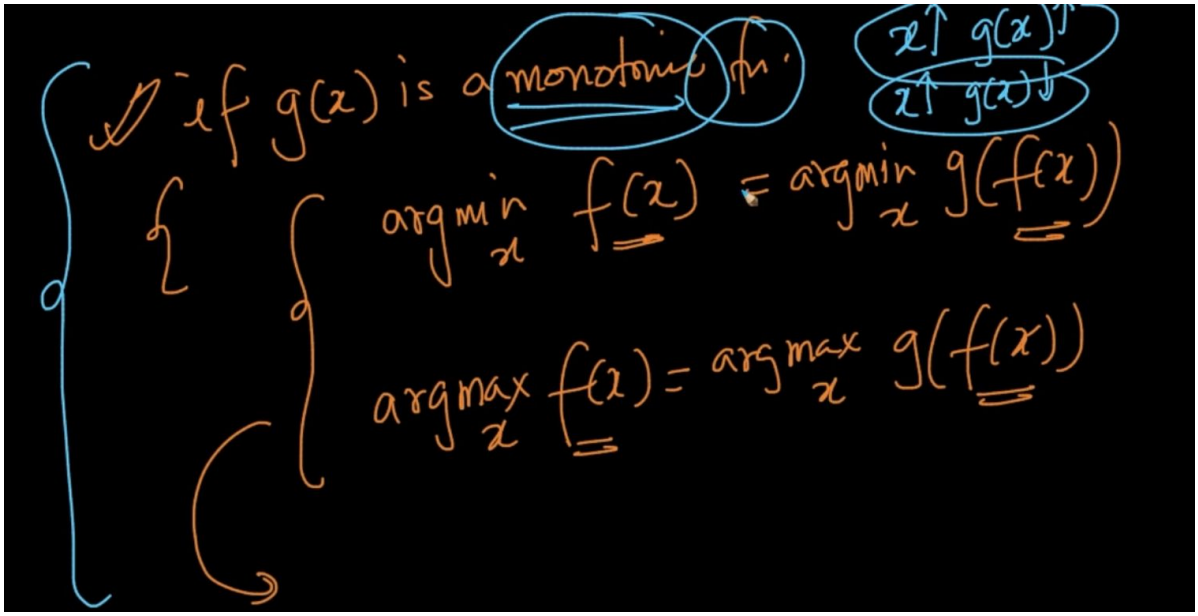
A monotonic function is a function which gets increased with x



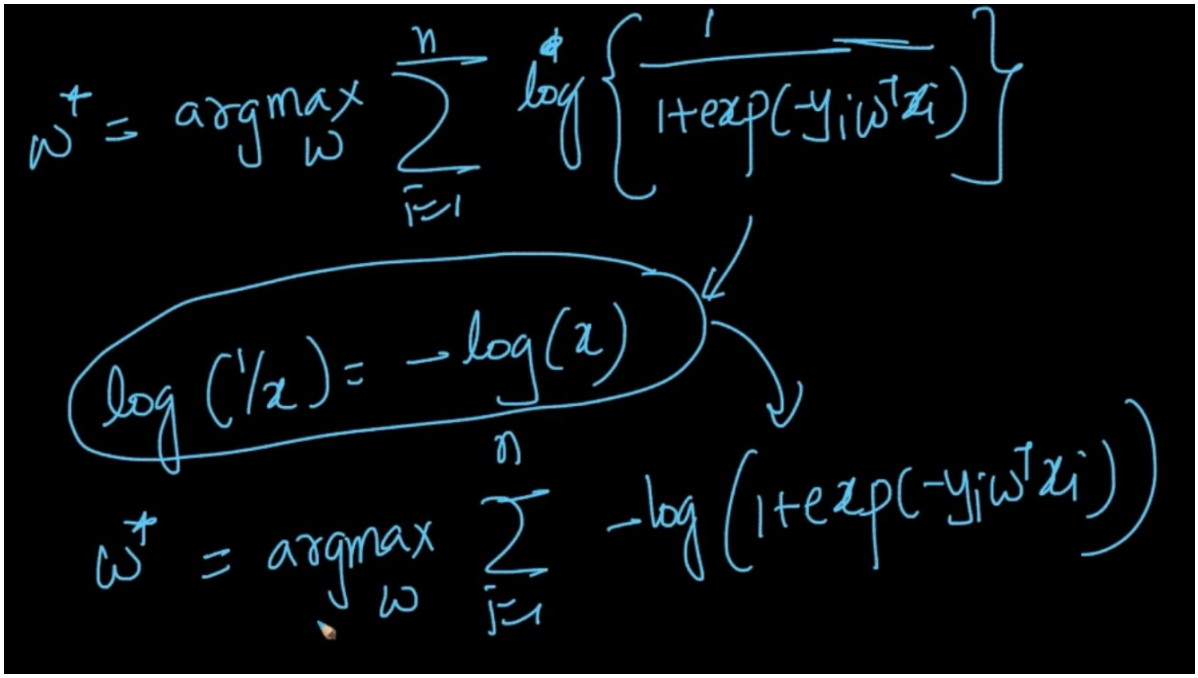
We are finding the minima of x^2 which is we can see from the plot is 0



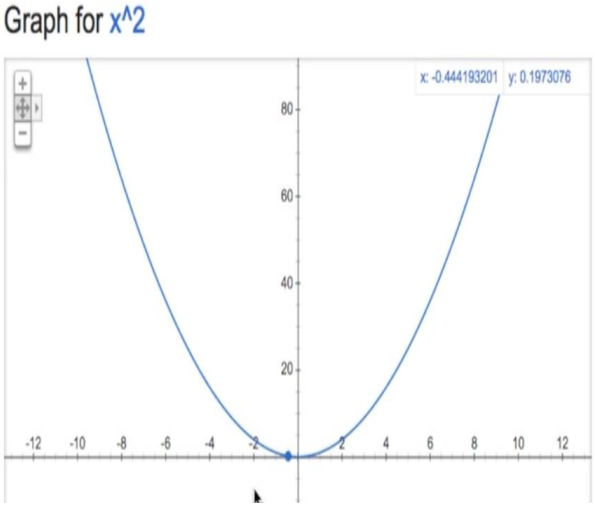
g(x) = log(x) which is a monotonically increasing function. We say here that x^* = x'

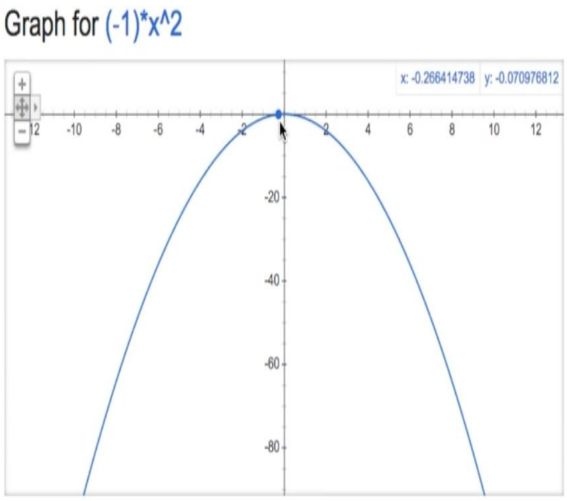


Here's the condition here.

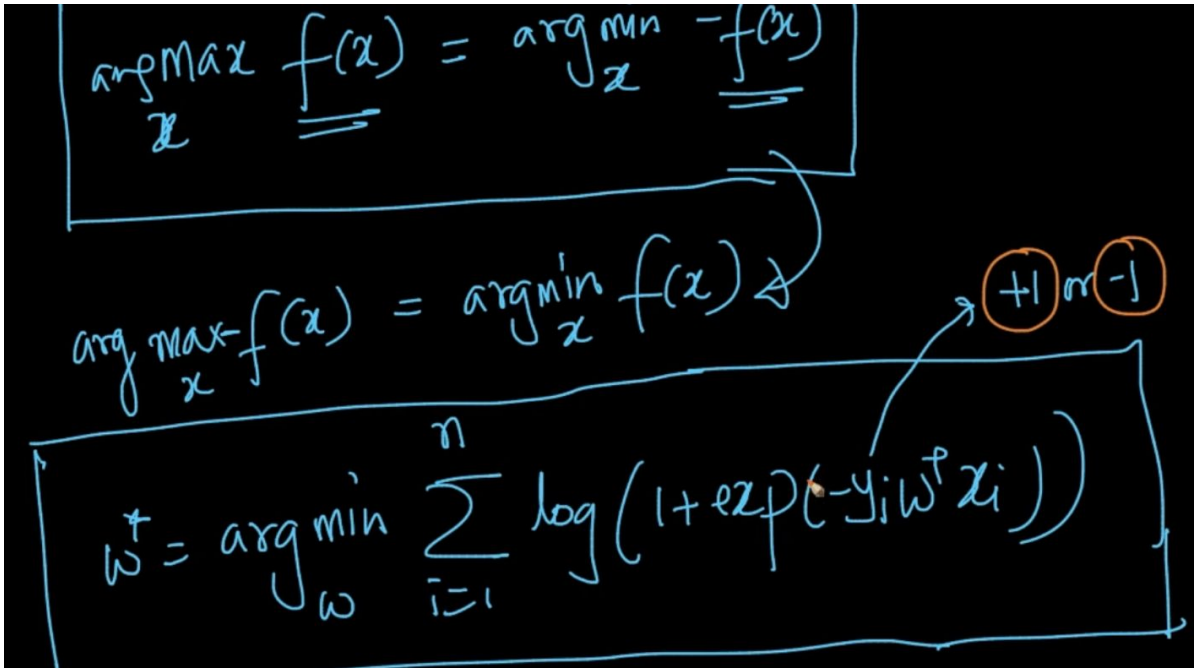


We have applied log to to the sigmoid function and getting that equation

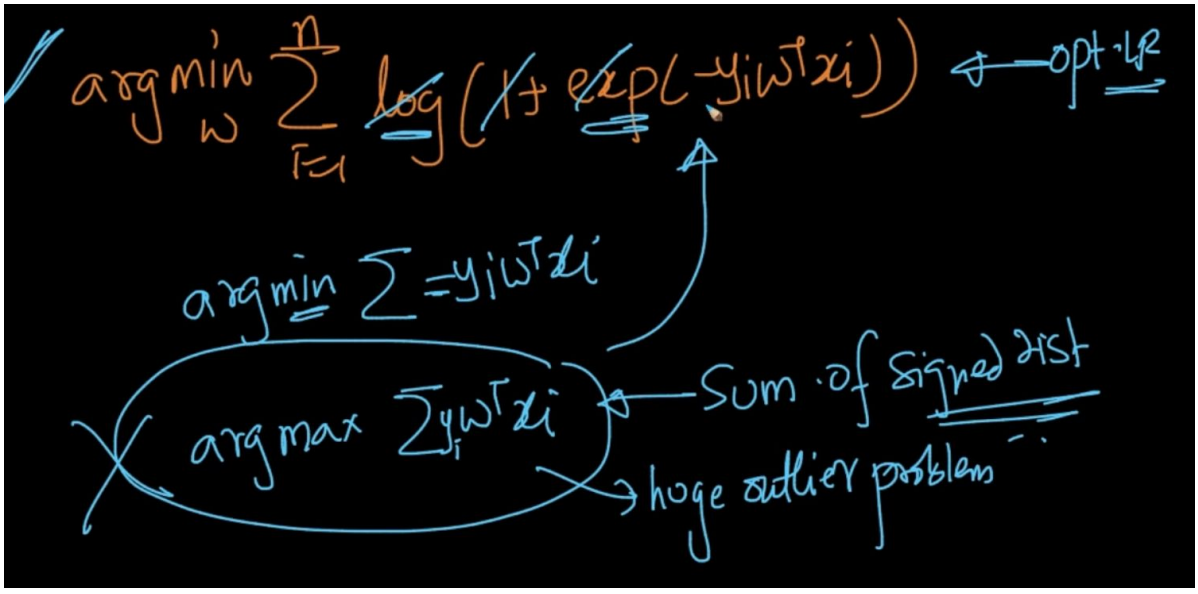




Minimum of x^2 = Maximum of $-x^2$. So we do the same to the above function as well

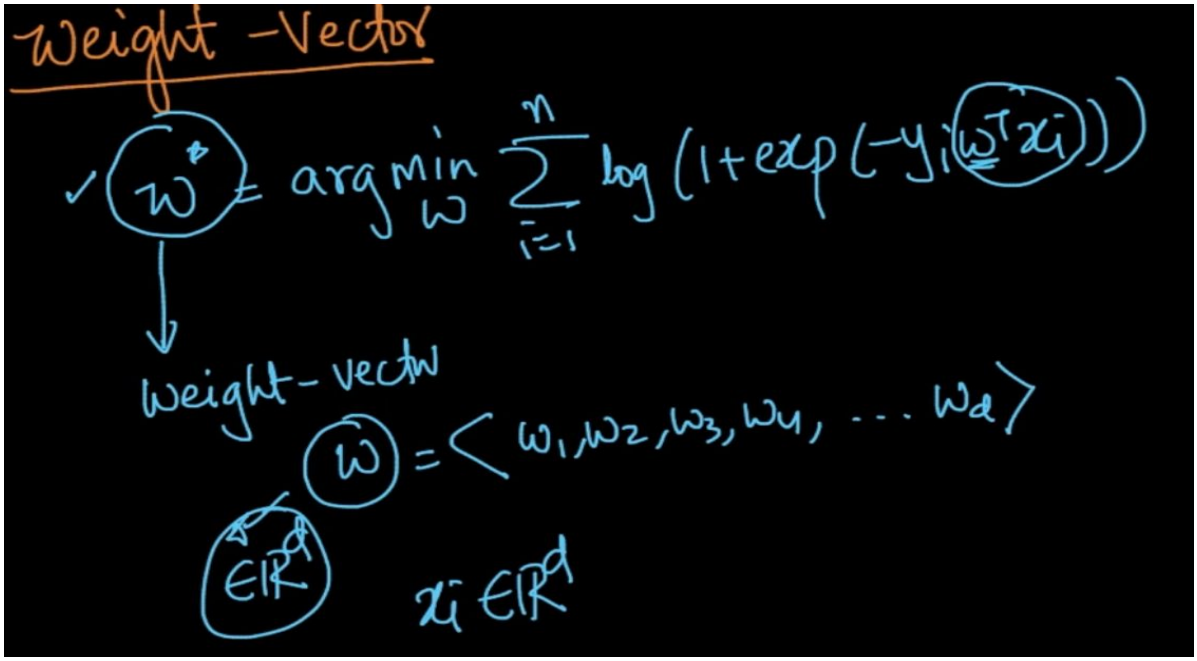


We want to find minimum(log(1 + exp($-y_i w^T x_i$)))

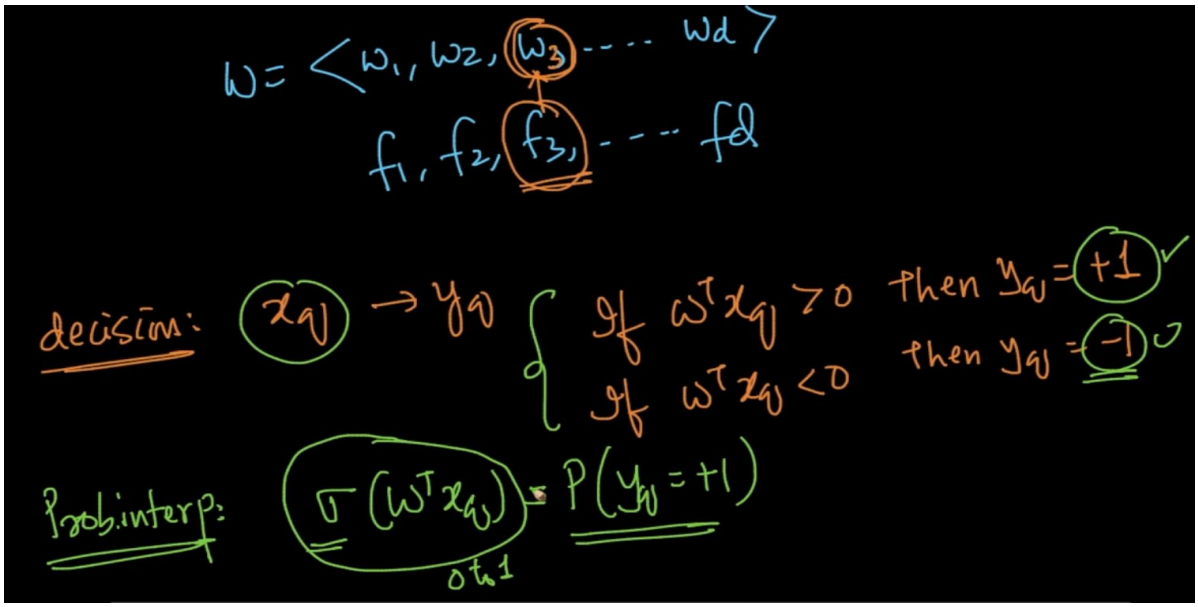


If we remove 1 from that we'll only get $y_i w^T x_i$ which is sum of signed distance but we are not solving that problem since it gives us a huge outlier problem

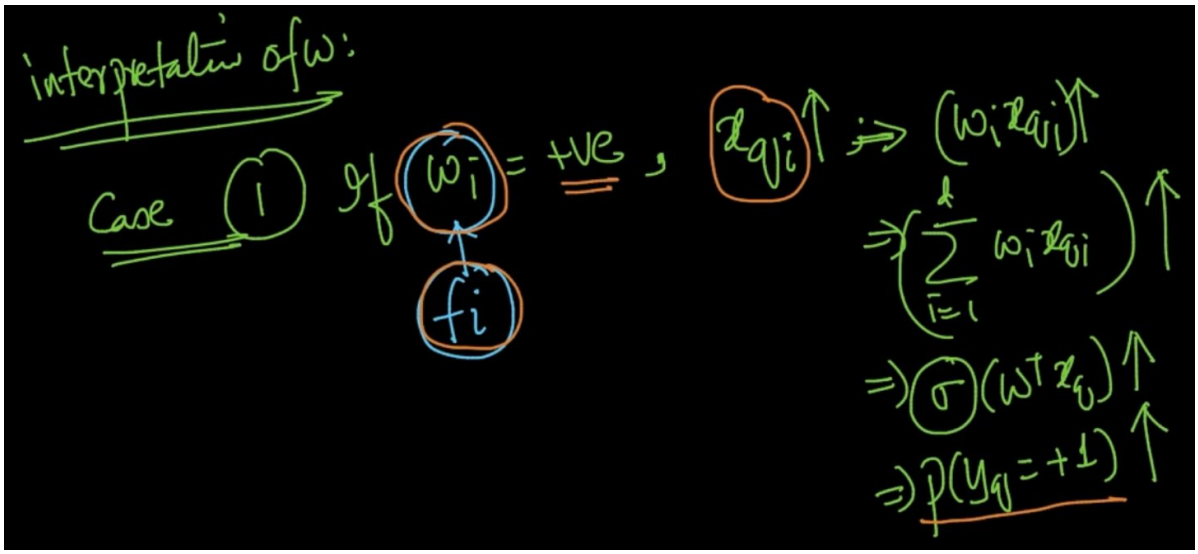
WEIGHT VECTOR



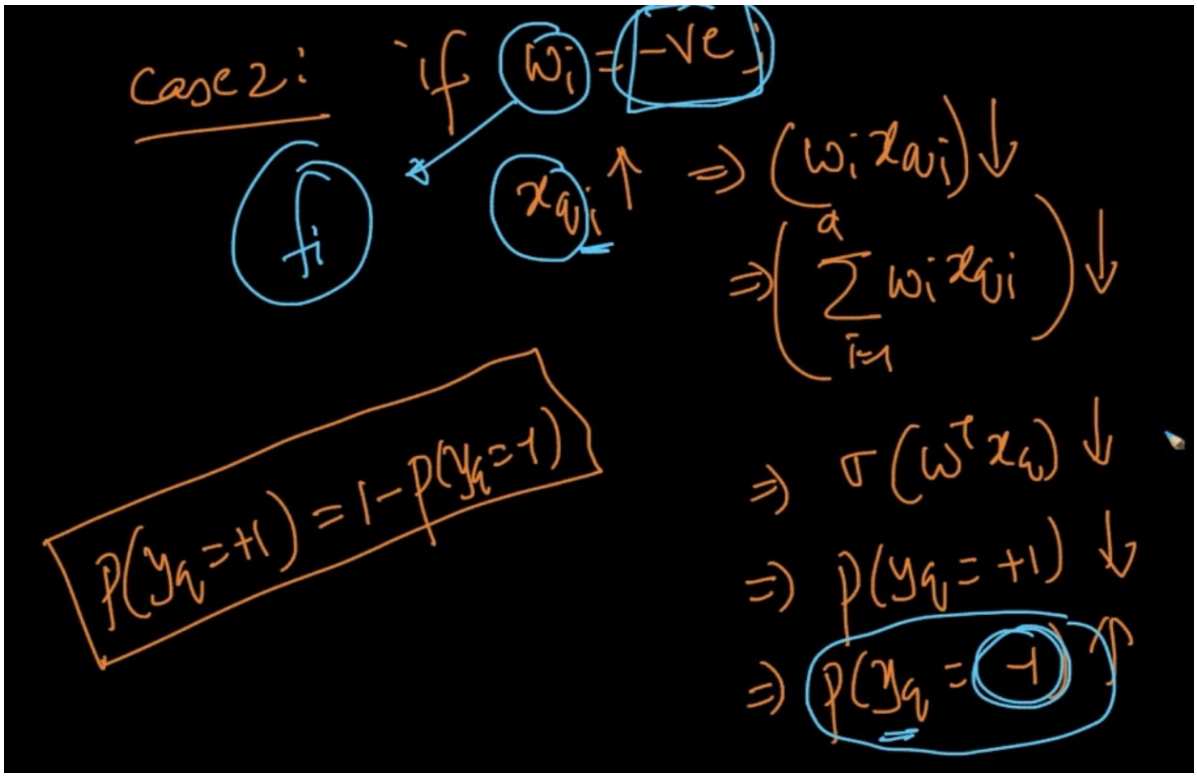
We want to get the optimal w which is also known as weight vector which has d-dimensions as well as x has d- dimensions



This is how we predict the class label y. We can also get probabilistic interpretation using sigmoid function

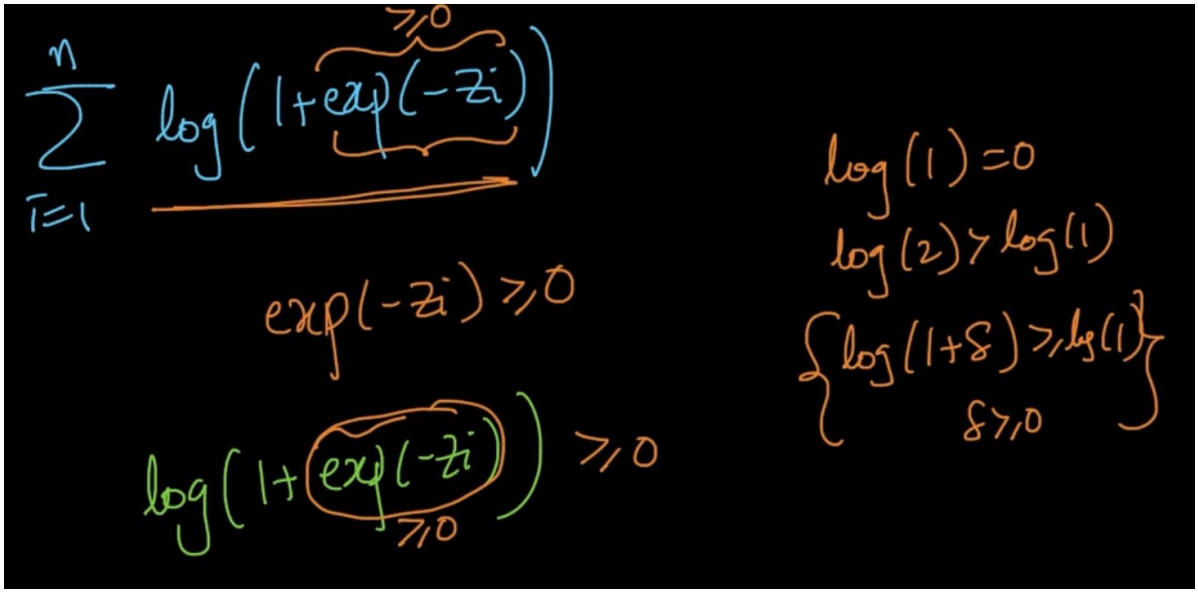


Case 1: If w_i corresponding to a feature f_i = +ve and x_{qi} ↑ then all the other things get increased as well. As x_{qi} ↑ => P(y_q = +1) ↑

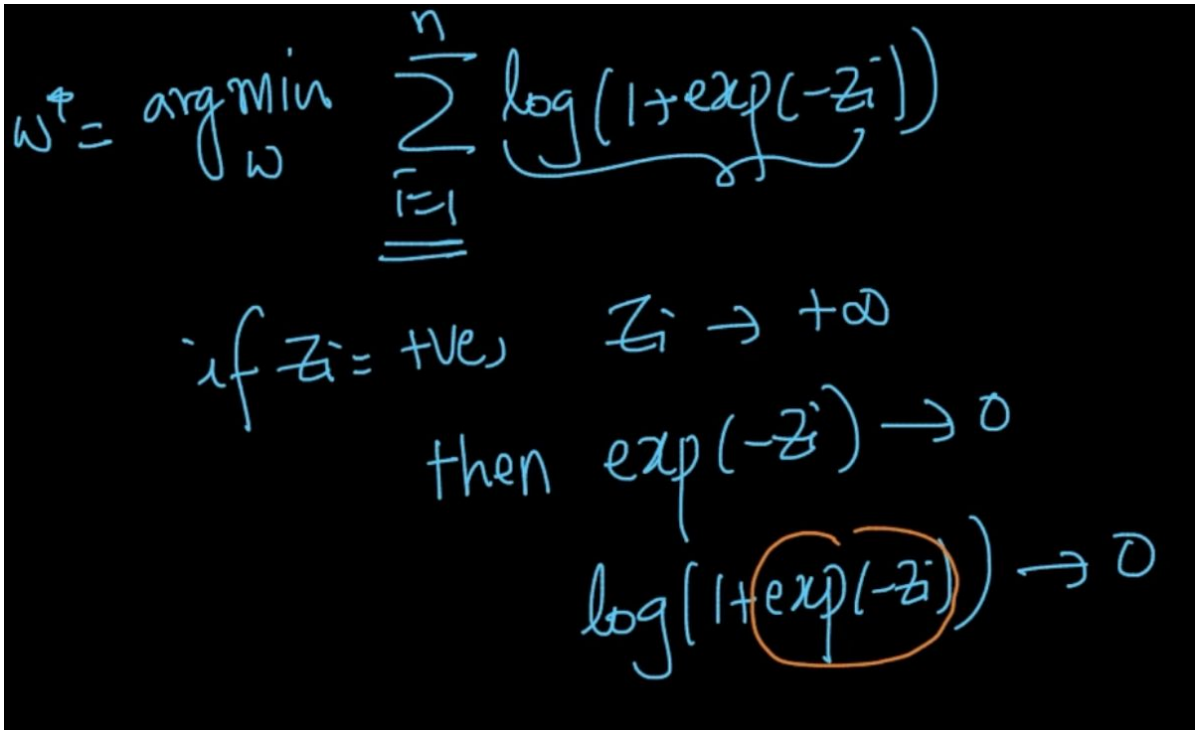


Case 2: If w_i = -ve and x_{qi} ↑ then all the other things will get decreased and P(y_q = +1) ↓ but P(y_q = -1) ↑

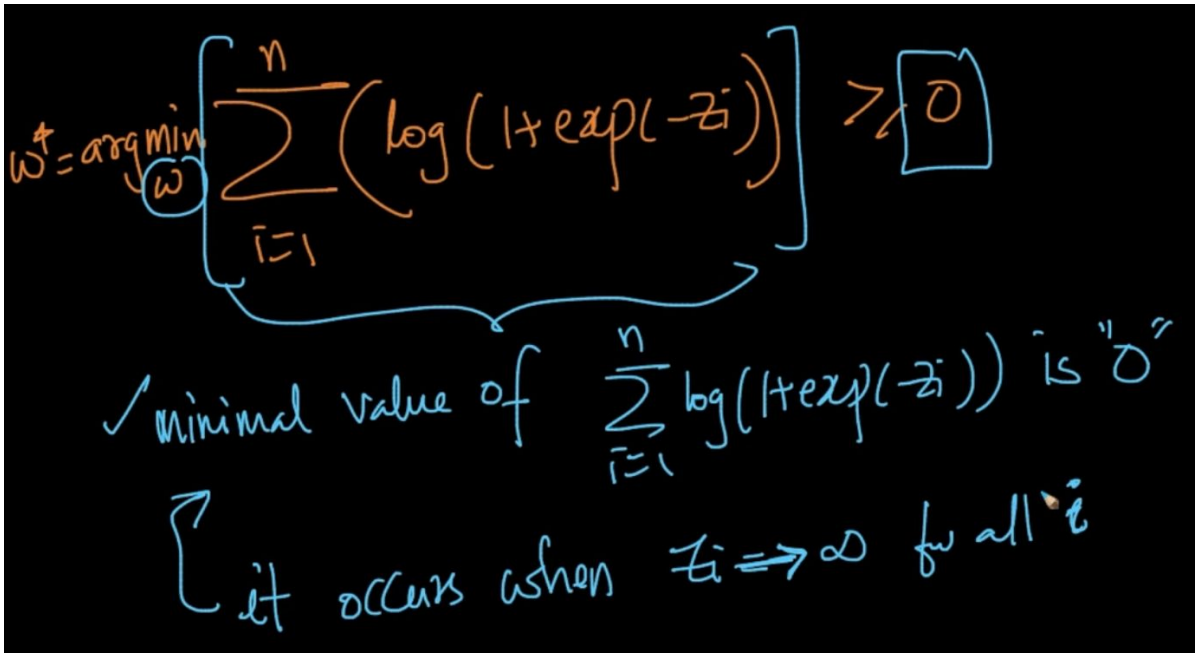
L2 REGULARIZATION OVERFITTING AND UNDERFITTING



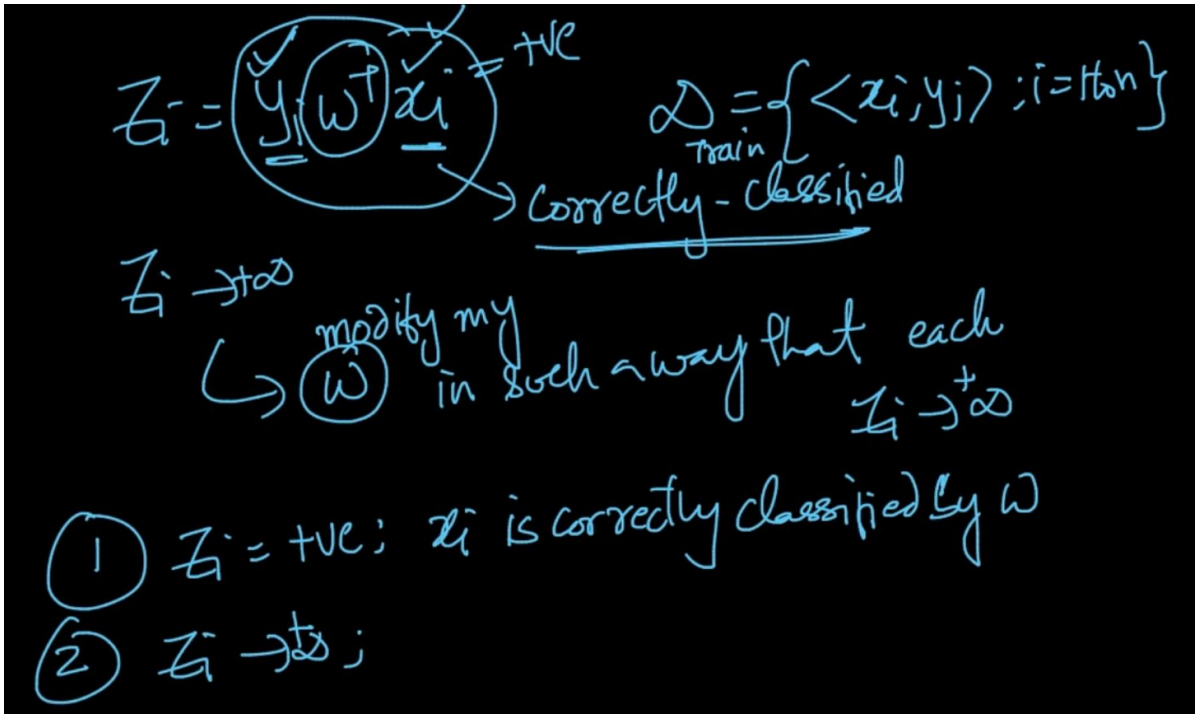
exp(-z) ≫ 0 and the value of our whole function > 0 since log(1 + δ) ≫ log(1) and δ = exp(-z)



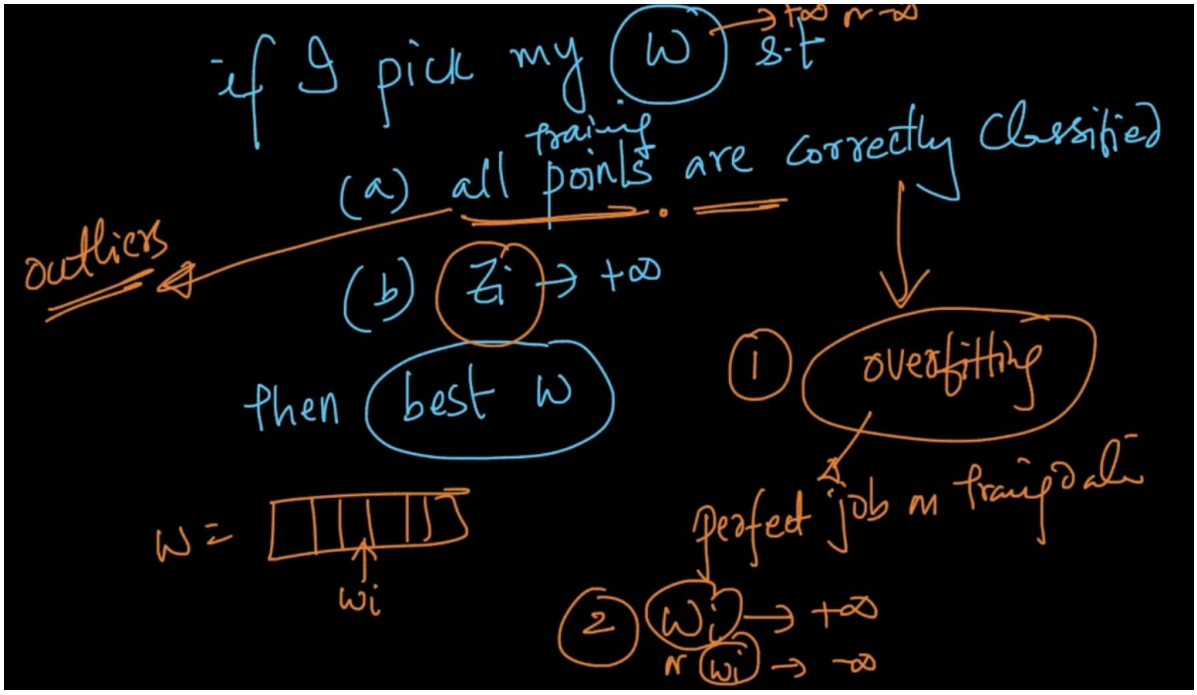
We want our z to be +ve for correct classification and if z -> ∞ then exp(-z) -> 0



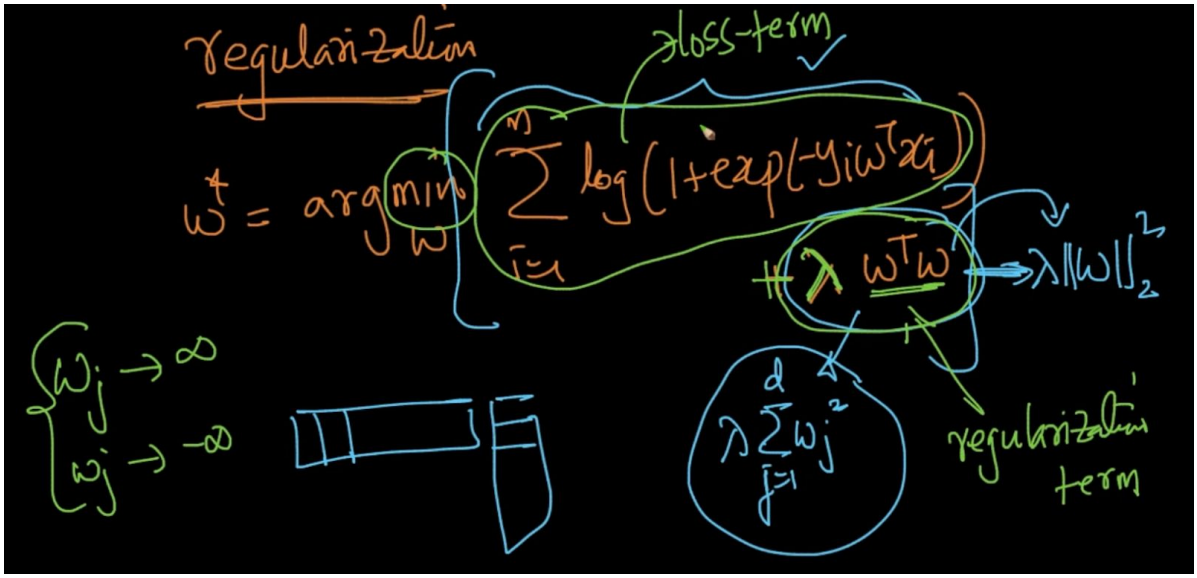
We want to minimize it and it's minimal value is 0 when z_i -> ∞ for all i



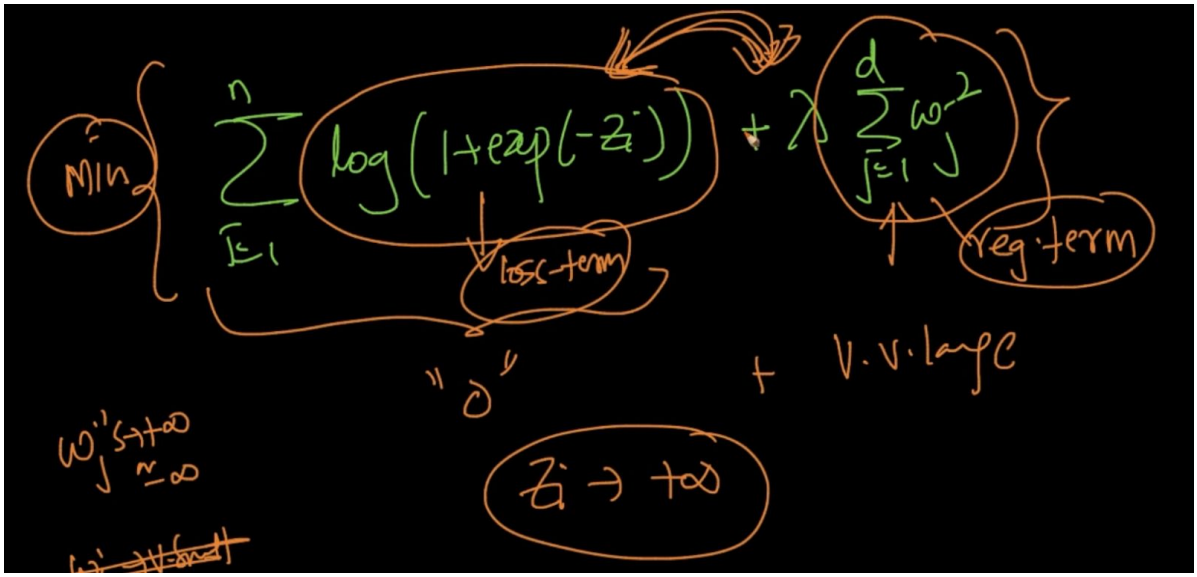
For z_i -> $+\infty$ we need to modify w such that z_i -> $+\infty$. SInce x and y are fixed w is the obly variable here



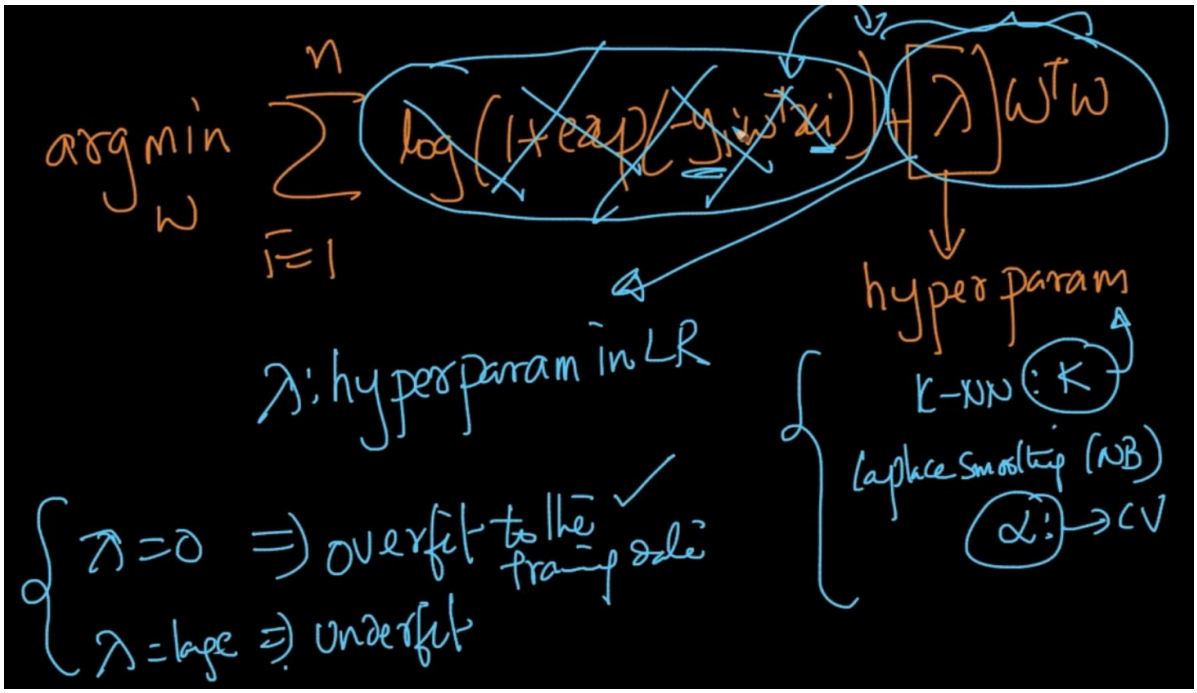
Since, our whole function is becoming 0 we are perfectly fitting the data . It means that it can be overfitting and since z -> + ∞ , w_i -> + ∞ so it's becoming too large



So we are adding $\lambda w^T w$ to our optimal w, known as regularization term. As w -> ∞ Then our regularization term will also be ∞ but we need to minimize it.

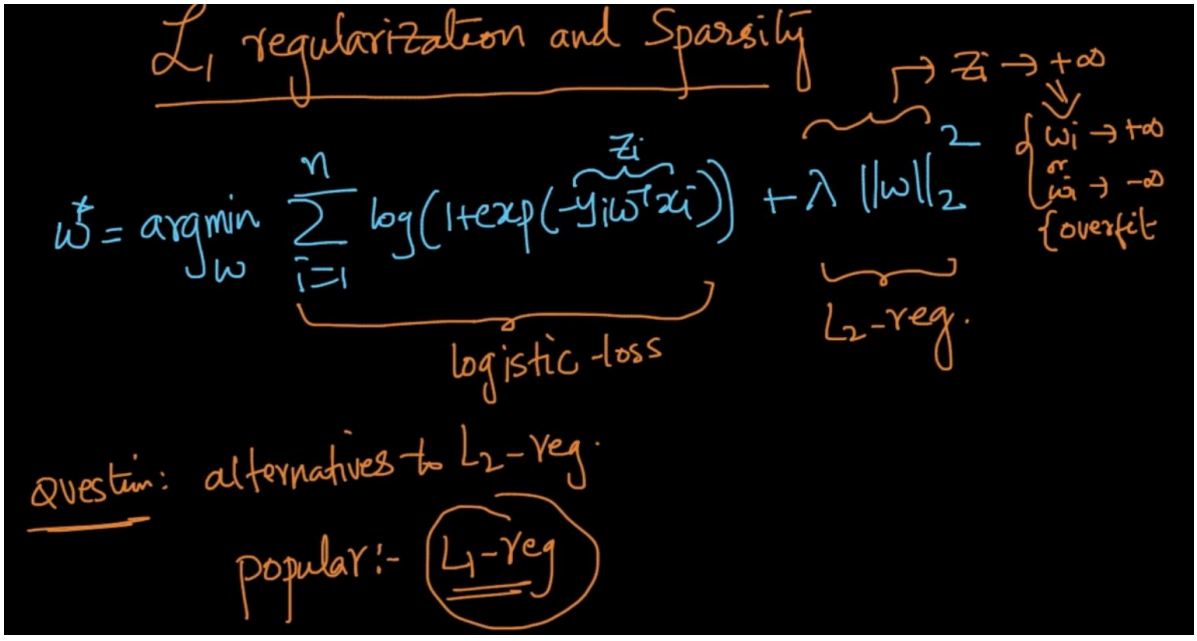


As w_j -> ∞ our log function will be 0 as exp(-z_i) but our regularization term will be very large and since we want to minimize the whole it won't happen as log function -> 0 means we are overfitting our data which we don't want. So both loss term and regular term are fighting with each other to minimize the whole

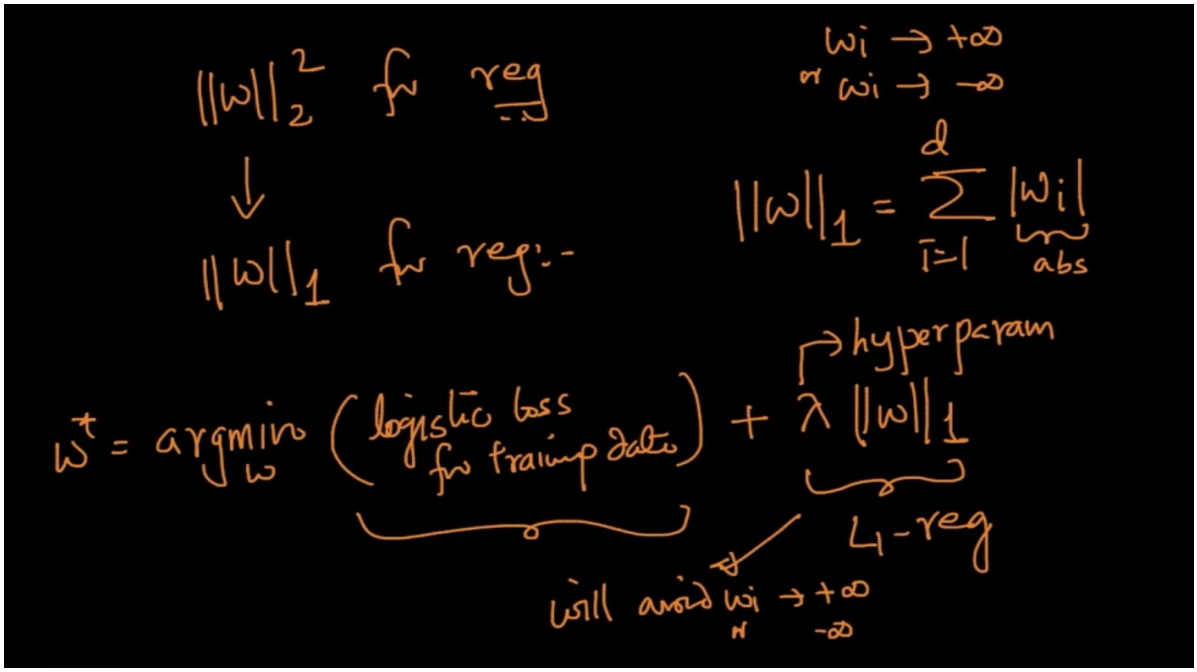


λ is hyper-parameter as λ= 0 => It'll mean overfit to Training data and if λ = large => Underfit since we are not using the training data i.e log thing and only w

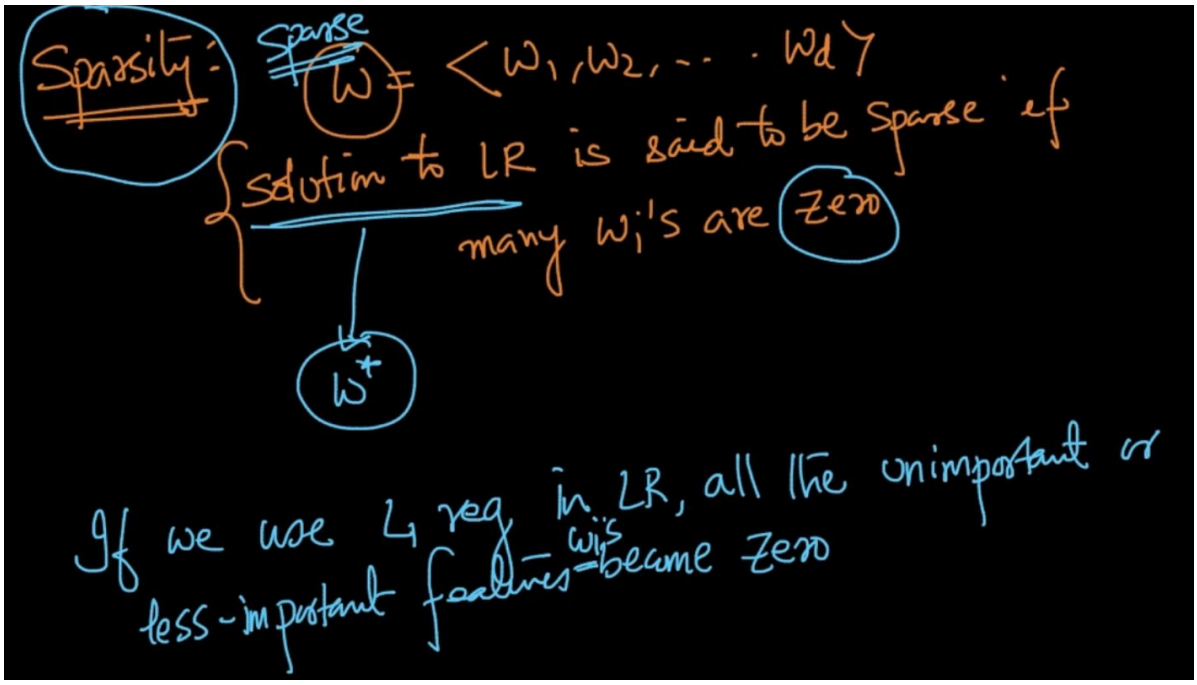
L1 REGULARIZATION AND SPARSITY



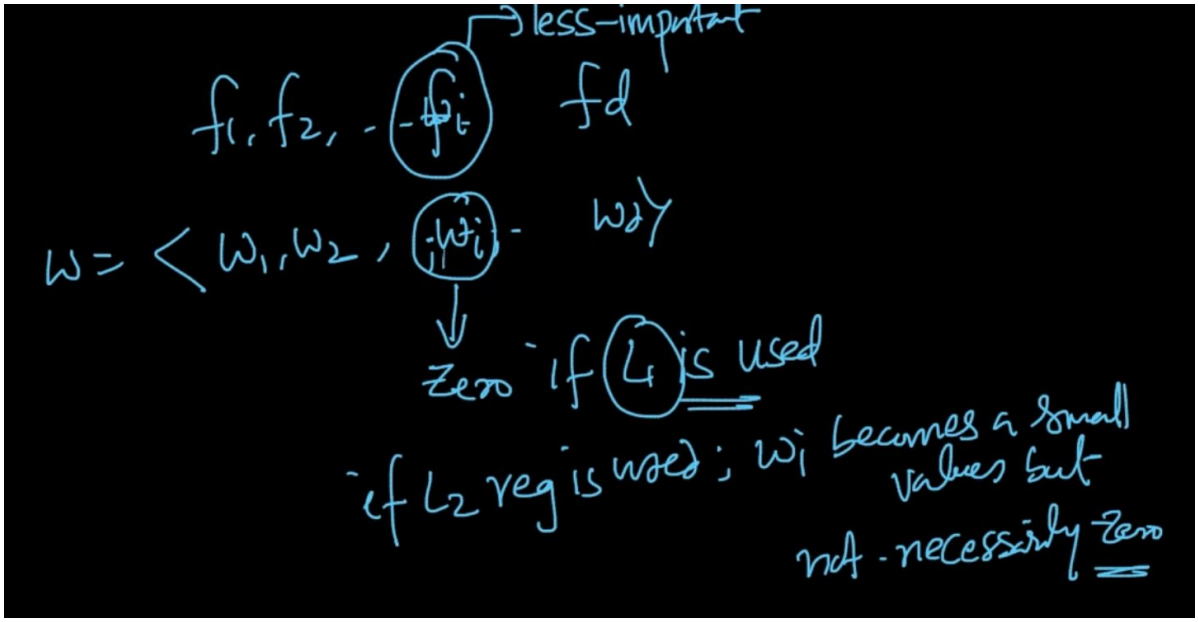
We'd seen L2 regularization now we see the other alternatives



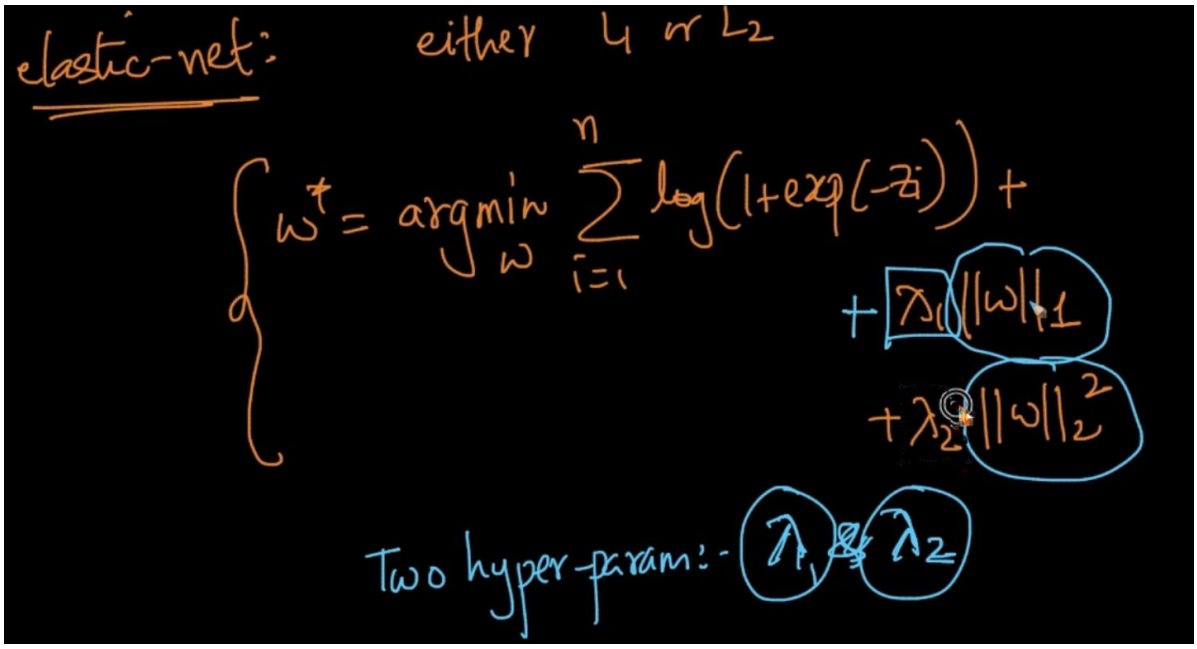
This is L1-Regularizer. It has absolute value of w. It's different advantage as compared to L2 which we'll see below



If we use L1 regularization then all the less important features become zero

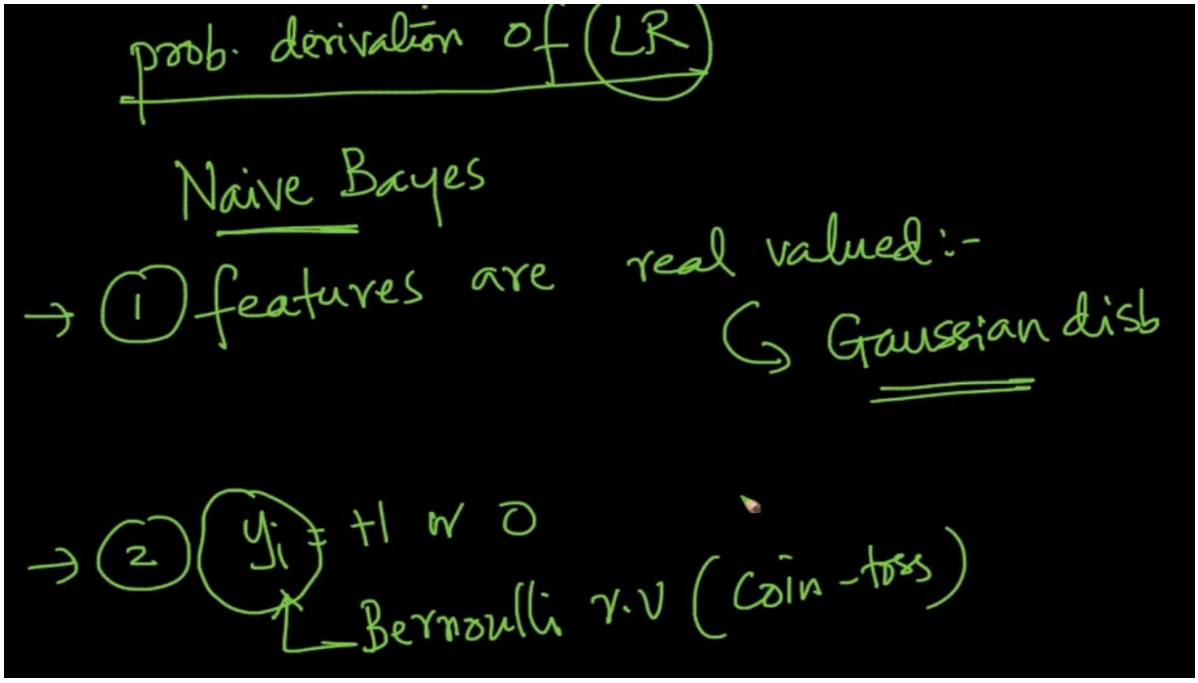


If we are using L1 regularization and feature f_1 is less important then it's corresponding w_i will be zero as compared to L2 where w_i becomes a small value

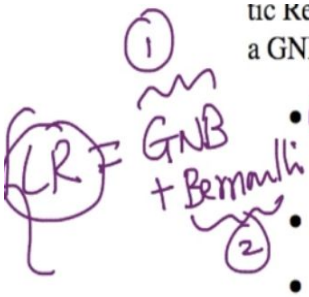


Elastic-Net is a combination of L1 and L2 regularization. It has two Hyper-parameters λ_1 and λ_2 which can be found out through cross-validation

PROBABILISTIC DERIVATION OF LOGISTIC REGRESSION



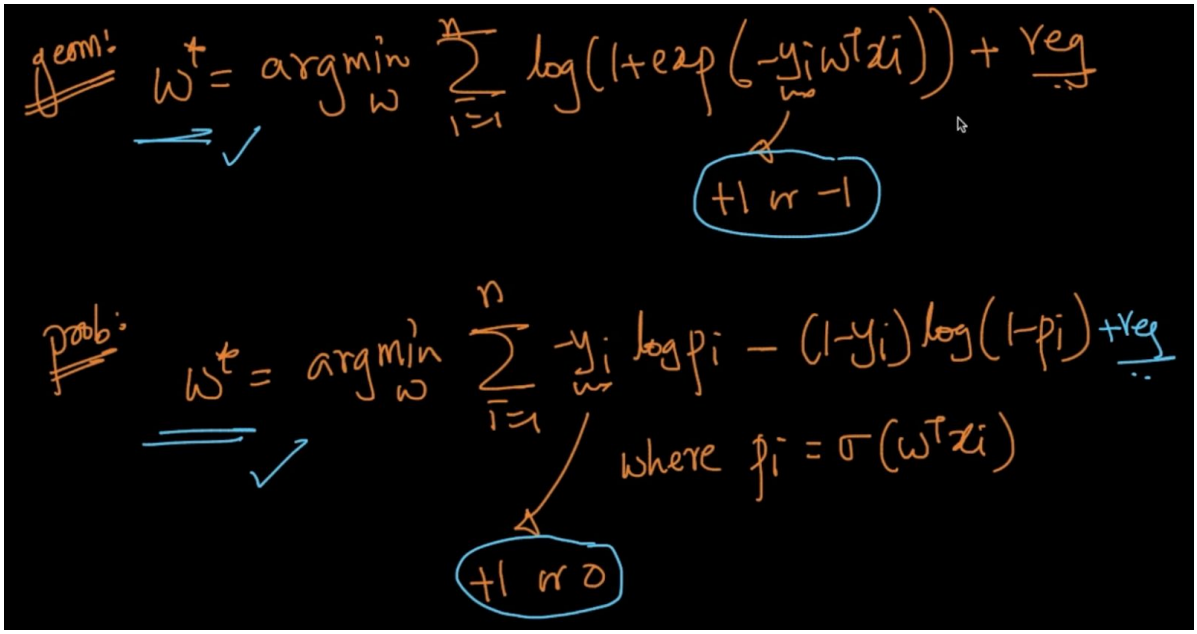
If these two points are true the Logistic Regression can be derived using probability



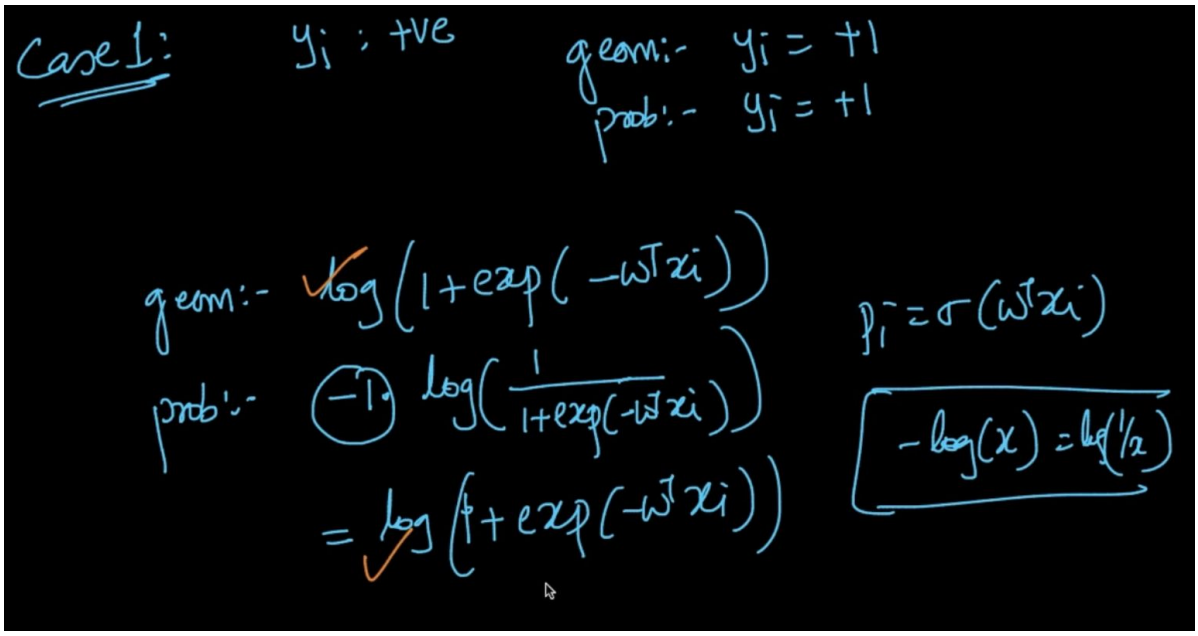
tic Regression and summarized in equations (16) and (17). In particular, consider a GNB based on the following modeling assumptions:

- Y is boolean, governed by a Bernoulli distribution, with parameter $\pi = P(Y = 1)$
- $X = \langle X_1 \ldots X_n \rangle$, 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

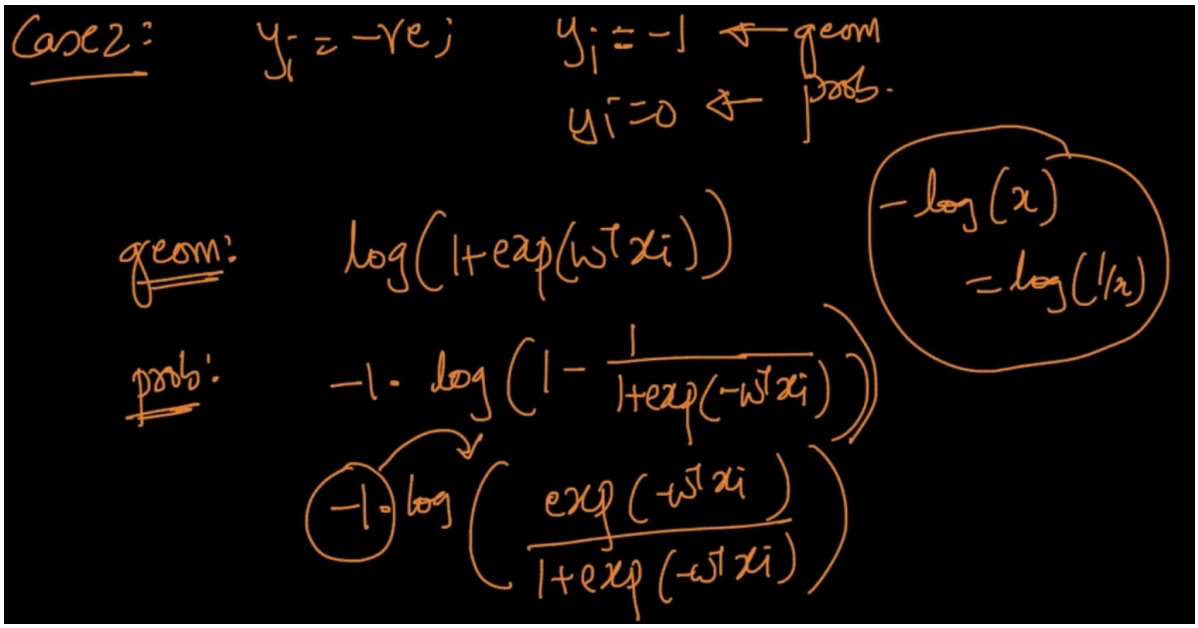
If the above assumptions are true we can easily derive i.e P(y = 1| X) = $\frac{1}{1+ exp(\sum_{i=1}^{n} w_i X_i)}$



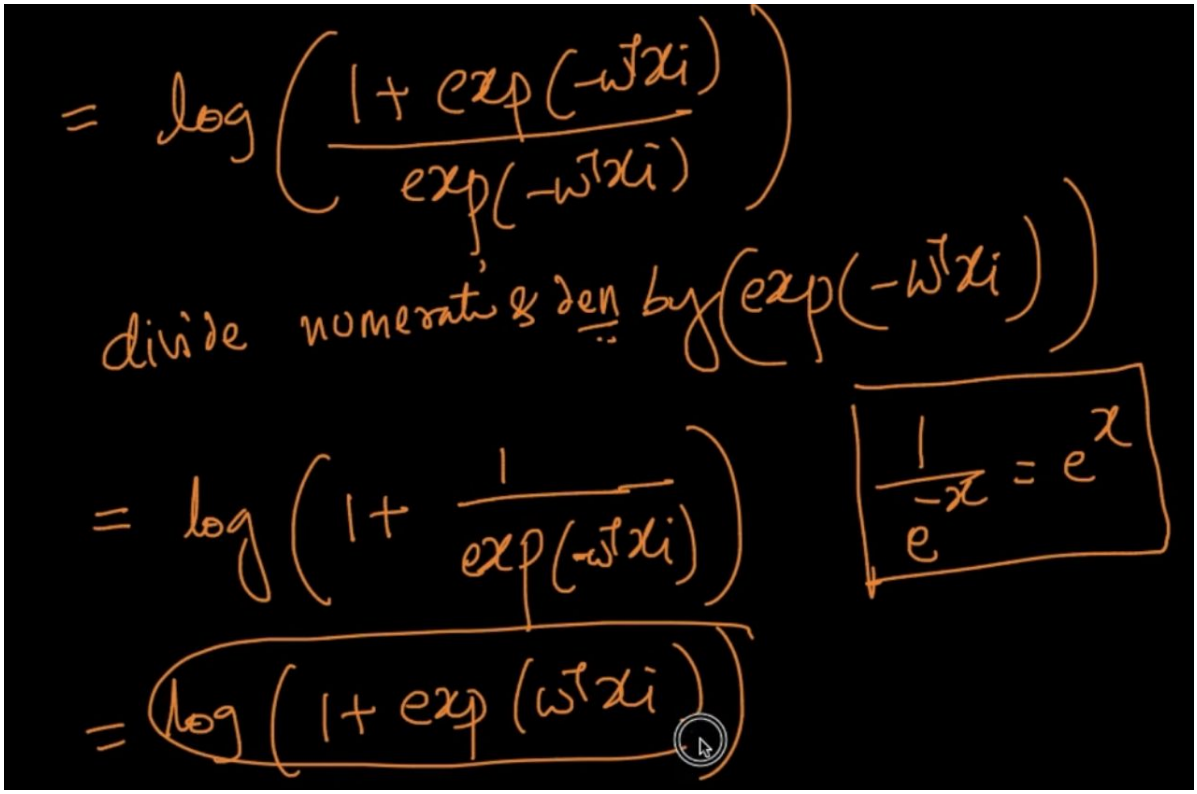
Both the formulae are same but in Geometric , y has +1 or -1 value and Probabilistic y is +1 or 0



Case 1: Here, y = +1 and if we do the math both will be same

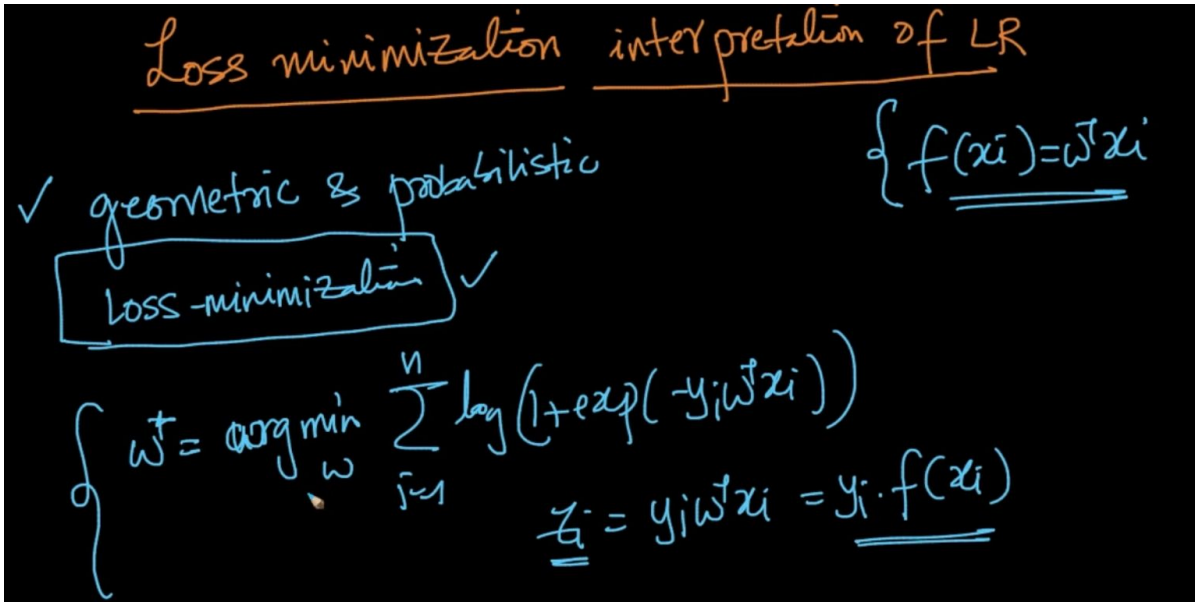


Case 2: Here, y = -1 in Geometry and y = 0 in probability. We are inverting the numerator and denominator using property -log(x) = log(1/x)

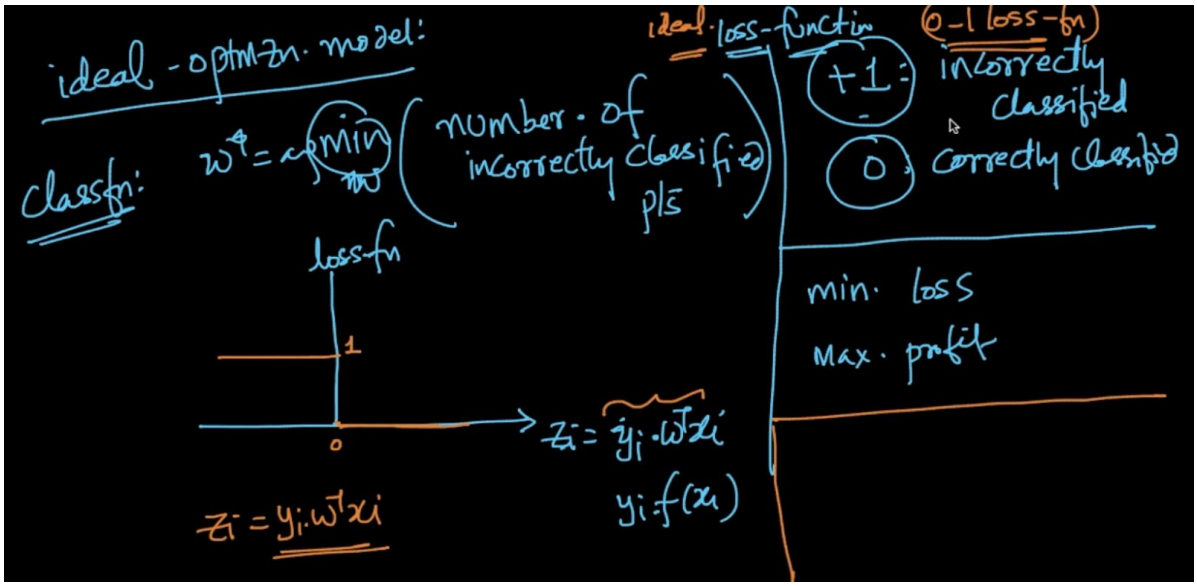


As we can see the Probabilistic measure is deriving to the same thing

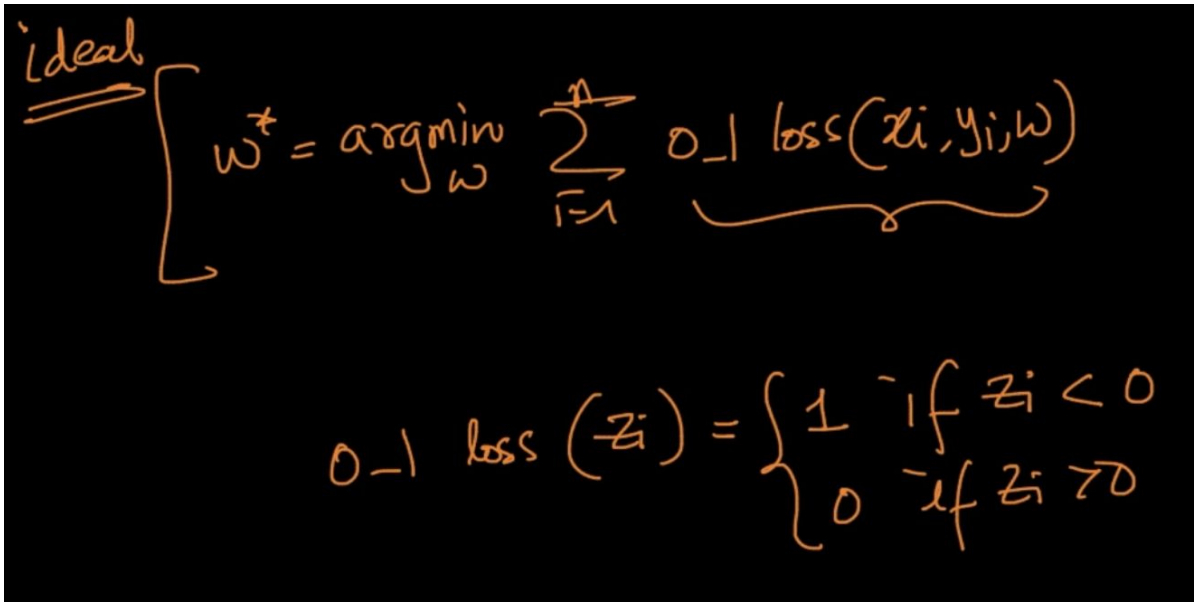
LOSS MINIMIZATION INTERPRETATION

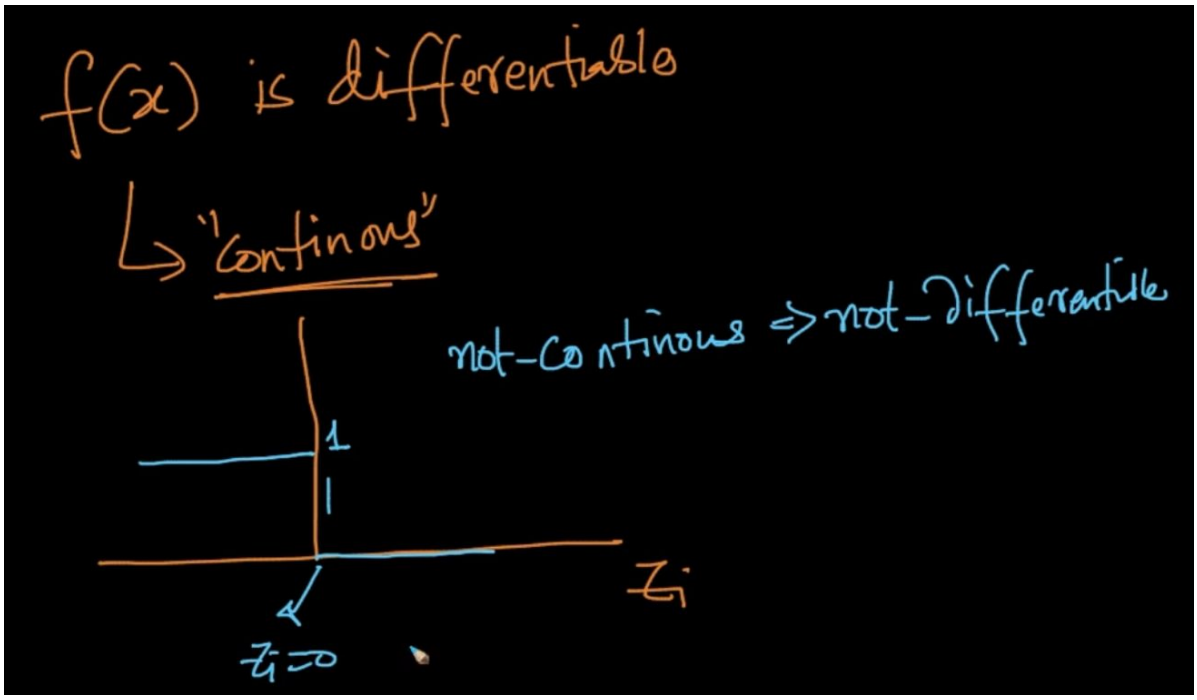


We'd seen Geometric and Probabilistic interpretation now we'll see loss minimization

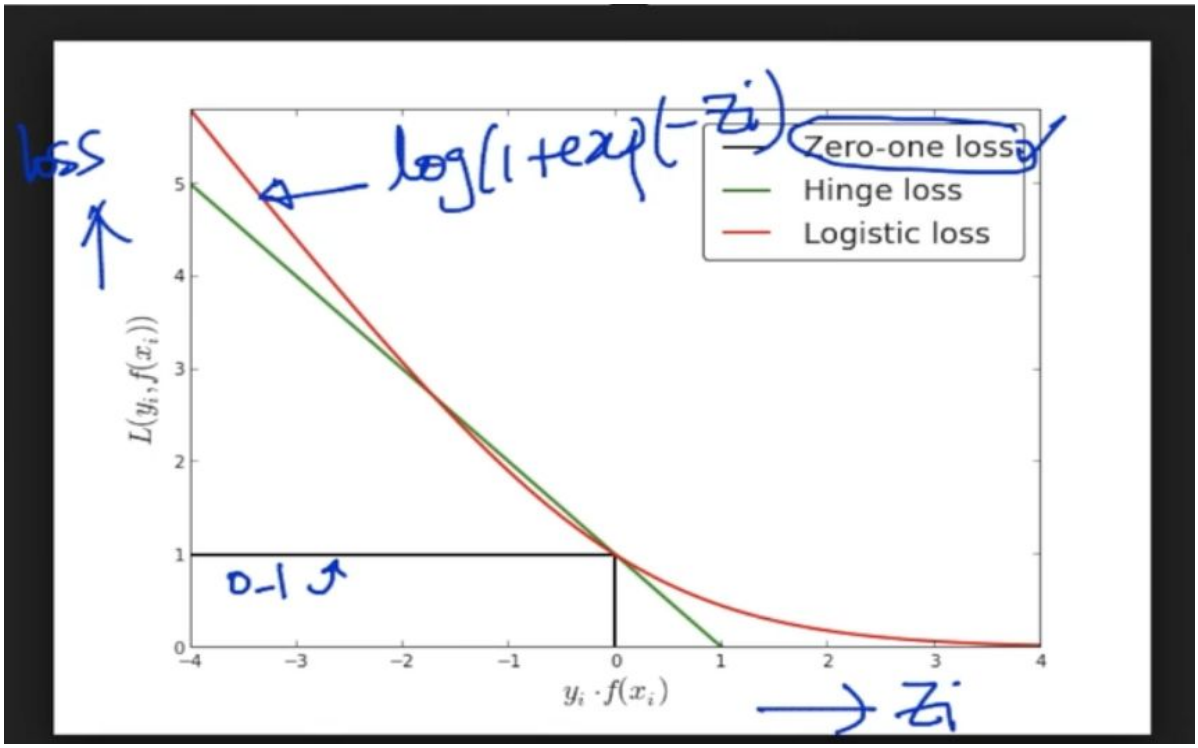


In our ideal optimization model we want to minimize the number of incorrectly classified points. Our loss function is saying that if +1: Then incorrect classification and 0 if correctly classified. So in the graph if z < 0 => 1 and if z > 0 then 0





But to solve the optimal model we need to differentiate the function f(x) and for function f(x) to be differentiable f(x) should be continuous

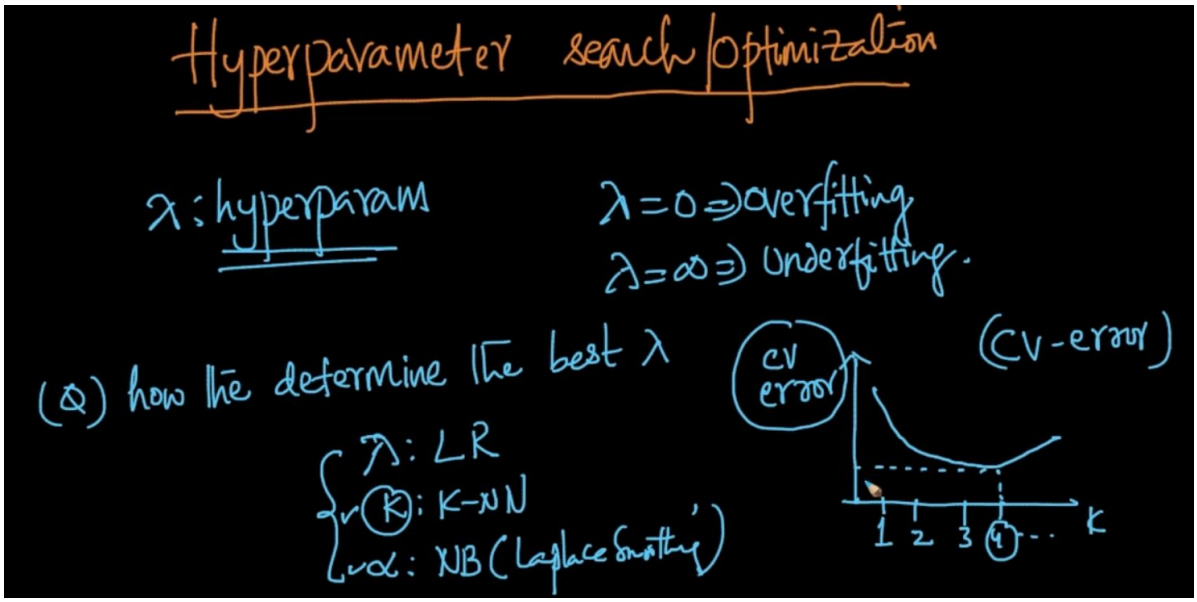


0-1 is not continuous hence not differentiable. So we'll approximate it by using continuous functions. We are using log-loss where y = log(1 + exp(-z)) where z = y.f(x) and f(x) = $w^T x$

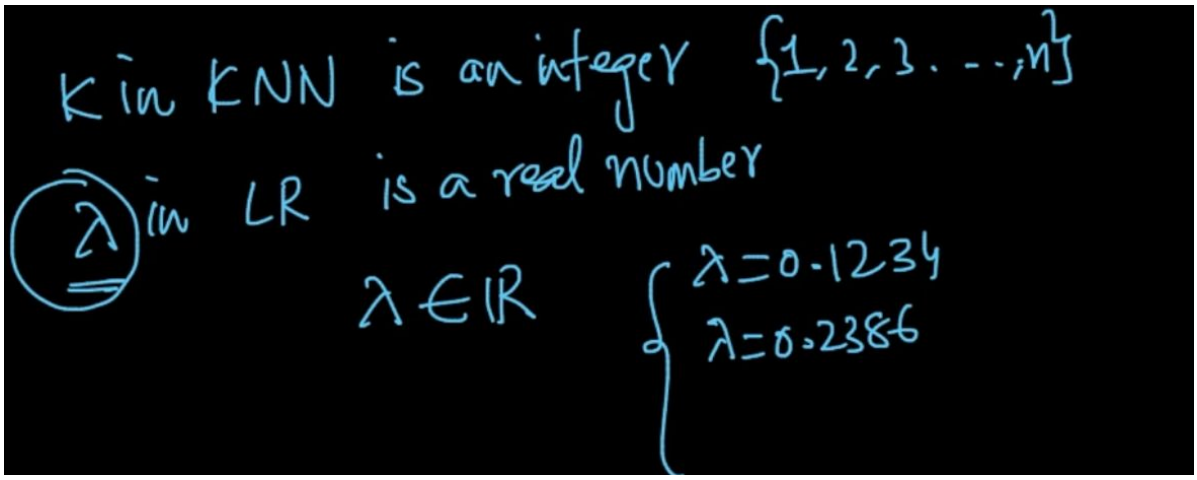
We could've drawn any other loss function as well but for logistic regression Logistic loss is used. Here, if z > 0 it's getting close to 0 after certain value but when z < 0 it's increasing sharply

Different loss function for different ML algorithms

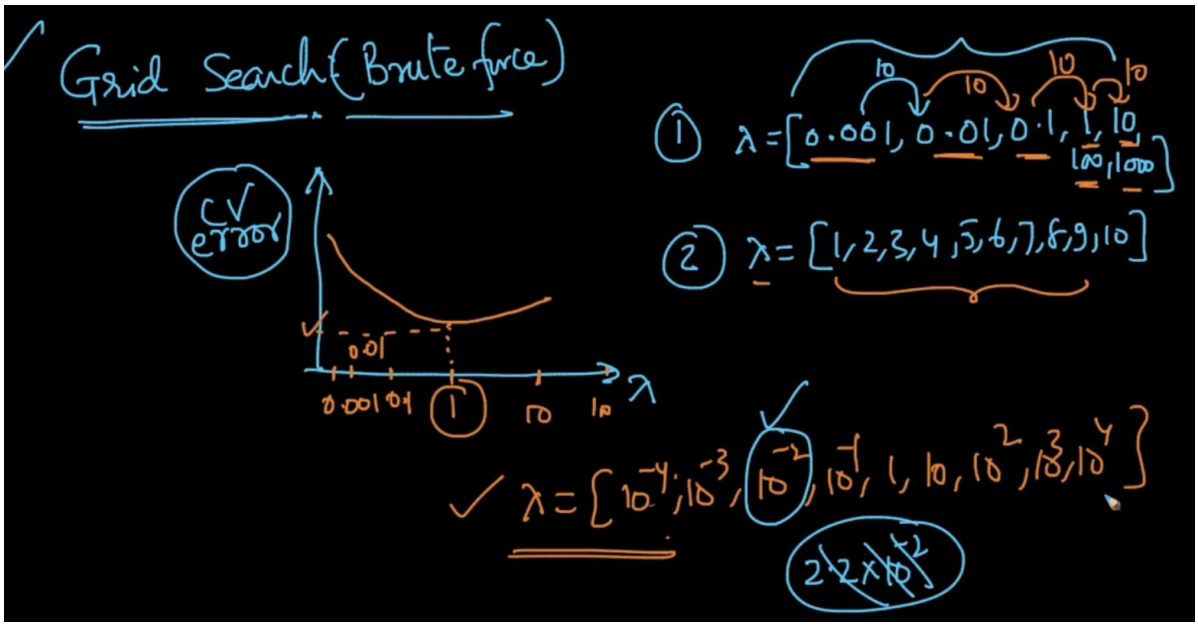
HYPERPARAMETER SEARCH/OPTIMIZATION



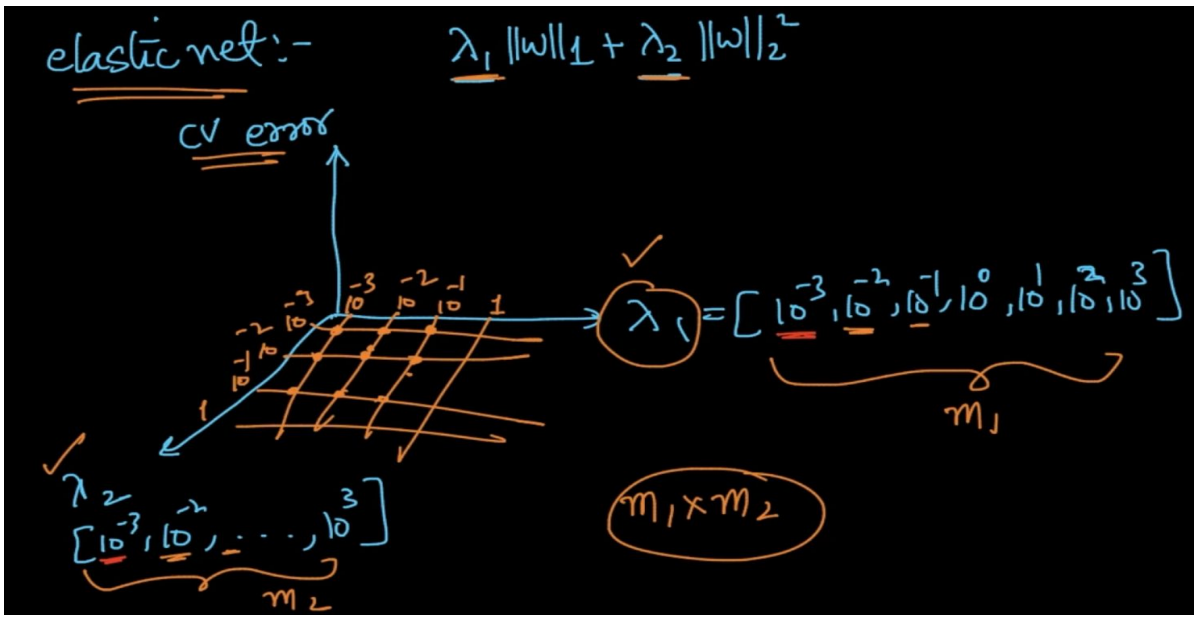
We want to identify the best λ for Logistic Regression. In k-NN, we found the best k by plotting the CV-error vs k graph and checking whichever k gives lowest CV error is our k



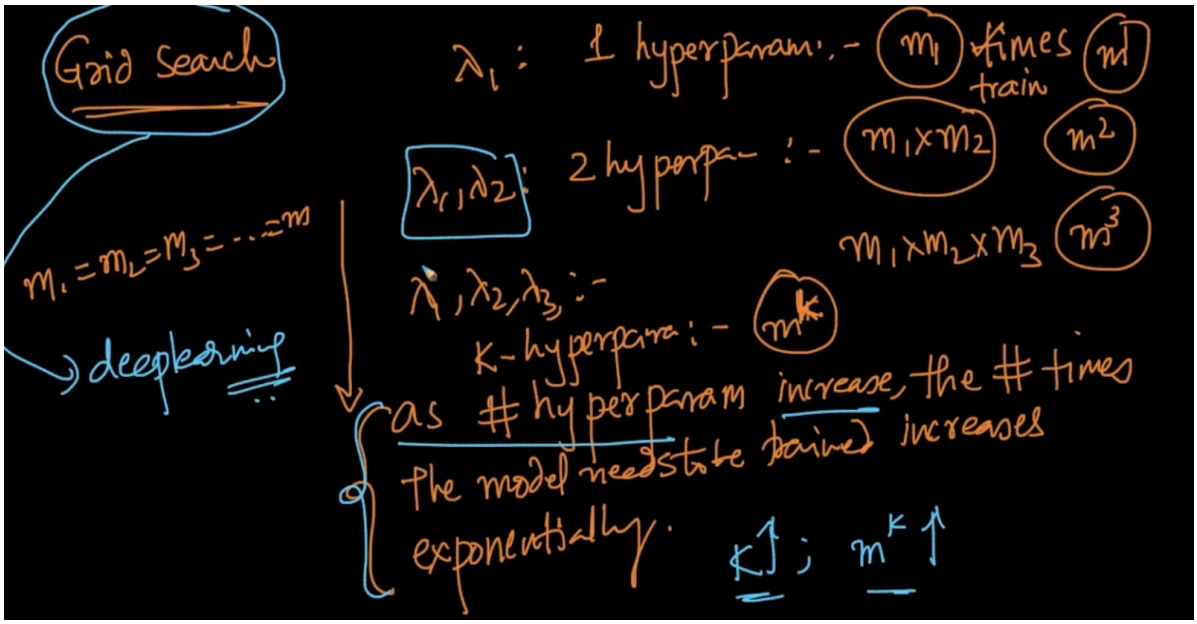
But our k in k-NN is an integer but our λ in LR is a real number



In order to tackle that real value situation of λ we'll do a Brute Force technique called Grid Search where we create a window from λ =[0.001,0.01, …..100, 1000] and we plot the CV error v/s λ graph and select the best λ which has lowest CV error

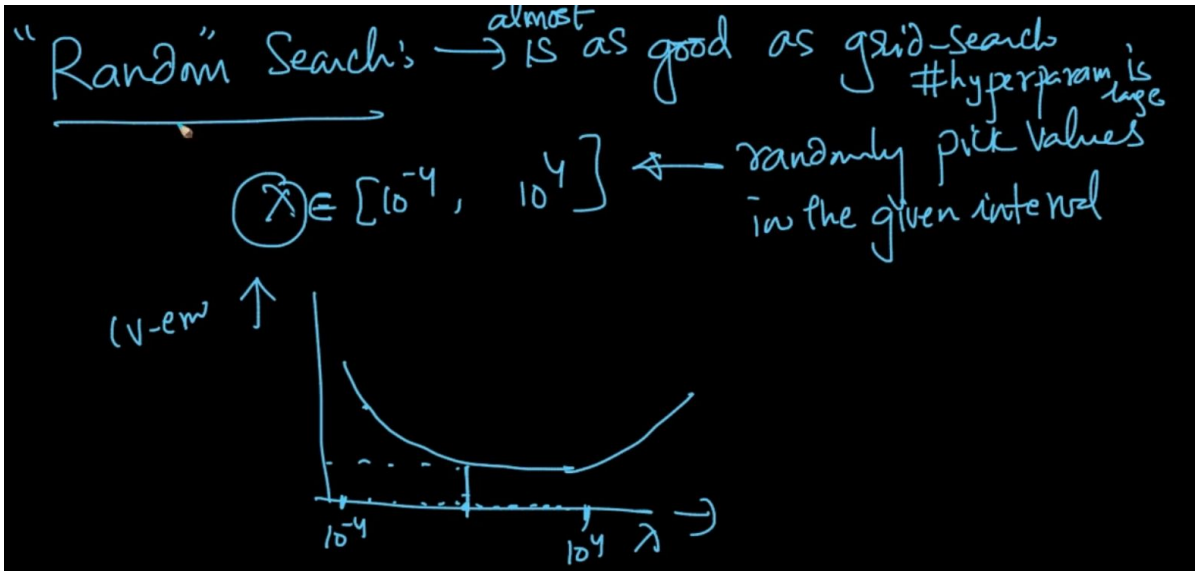


If we've more than one parameters, lets say above. We'll plot the graph of λ_1 v/s λ_2 v/s CV error to select the best λ_1 , λ_2 which has lowest CV error. If λ_1 has m_1 dimensions and λ_2 has m_2 dimensions so total calculations will be $m_1 * m_2$



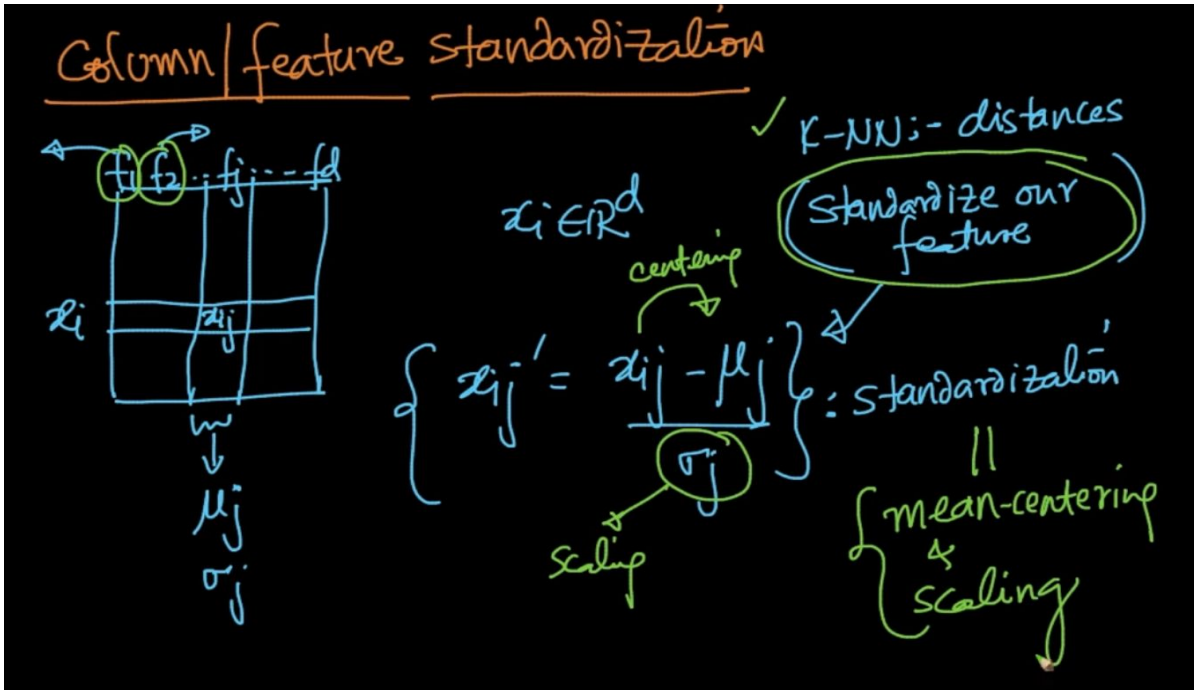
There's one problem in Grid Search. It's that as number of hyperparameters increase The no. of times the model needs to be trained is increased exponentially

Ex : For 3 hyperparameters with m-dimensions has m^3 calculations.

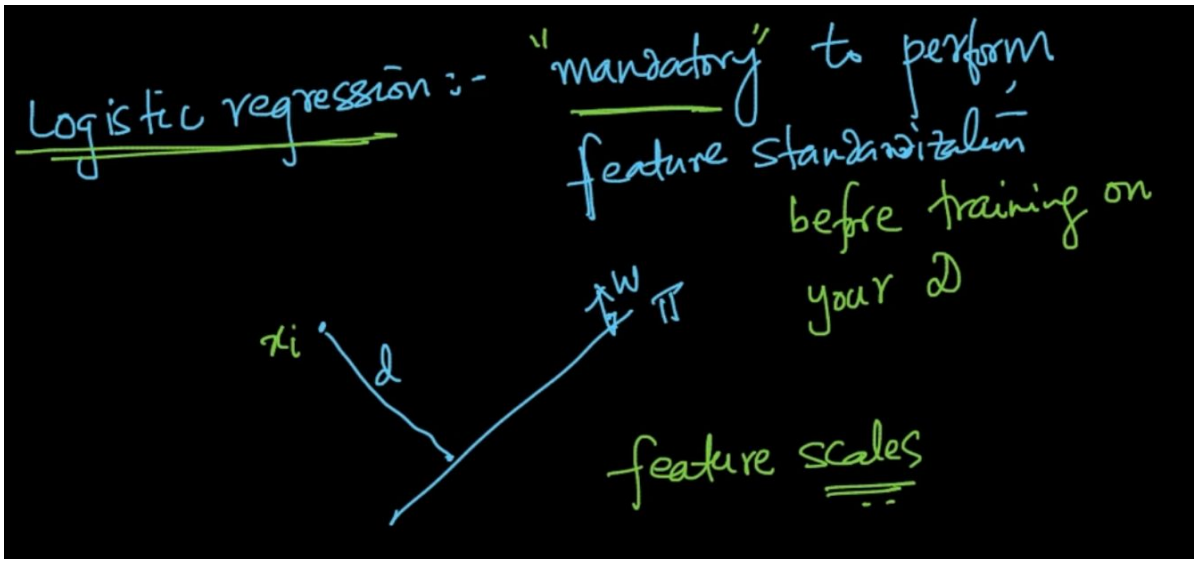


There's another technique called Random Search in which wee select an interval and from that interval we randomly pick values and get the value of λ with the lowest CV error. It's said that it's as good as Grid search . Used when hyperparameters are large

COLUMN / FEATURE STANDARDIZATION

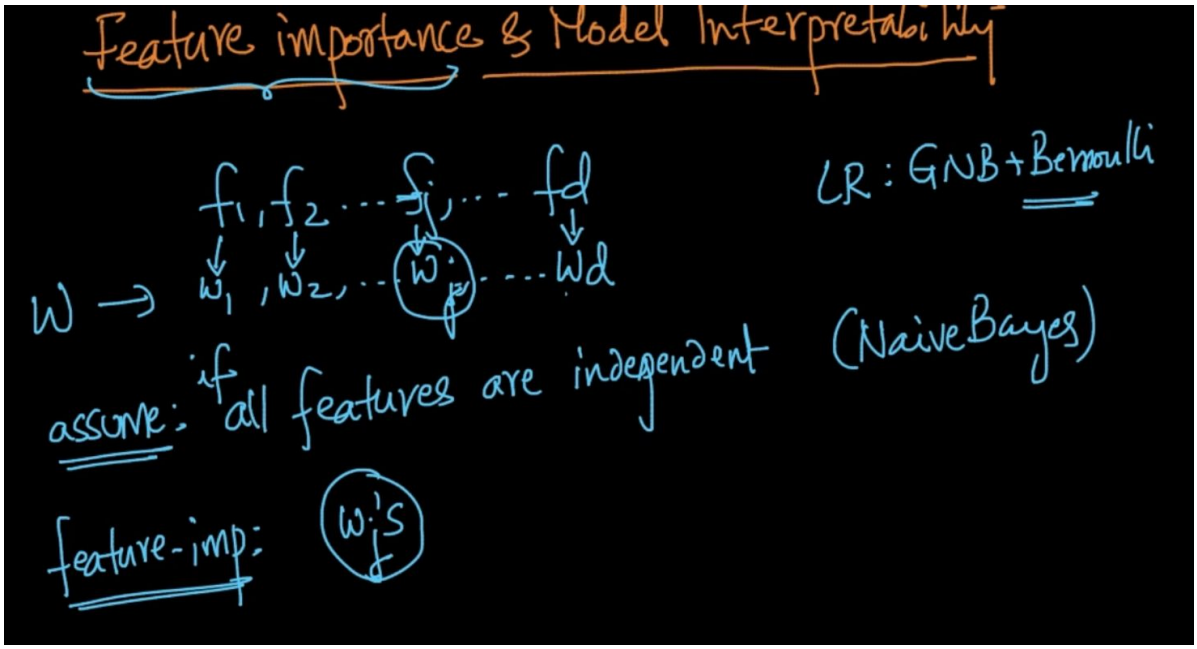


We are performing Column standardization by mean-centering and scaling



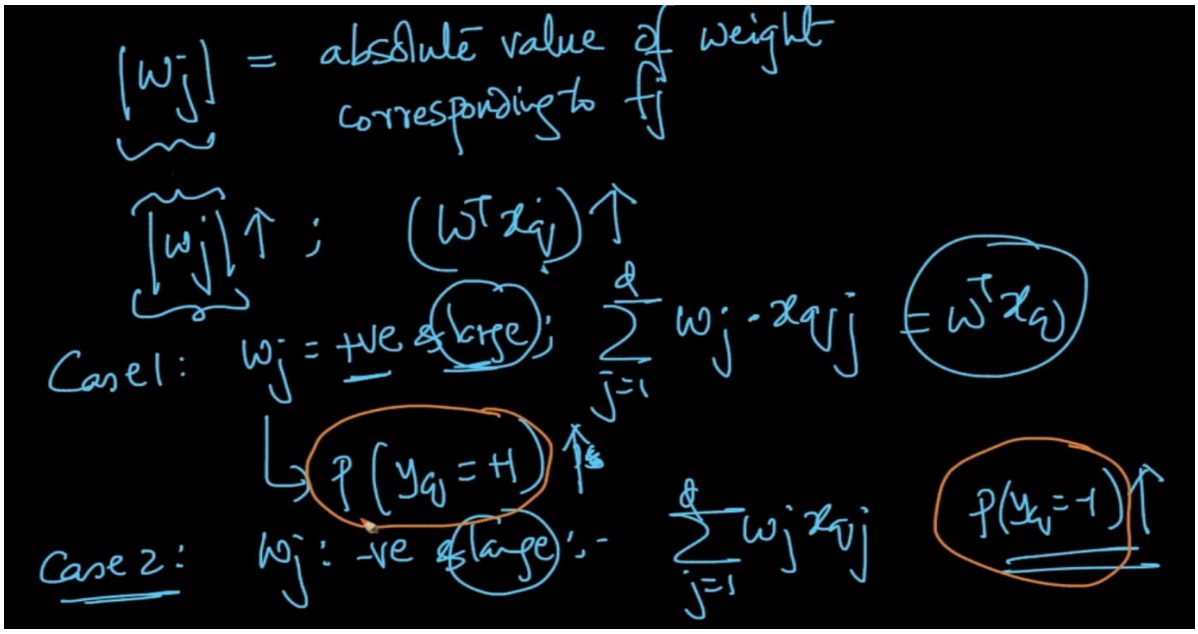
In Logistic Regression, it's mandatory to perform Column Standardization because if two features are at different scales then it can create problems as distances can be affected by scales

FEATURE IMPORTANCE AND FEATURE INTERPRETABILITY

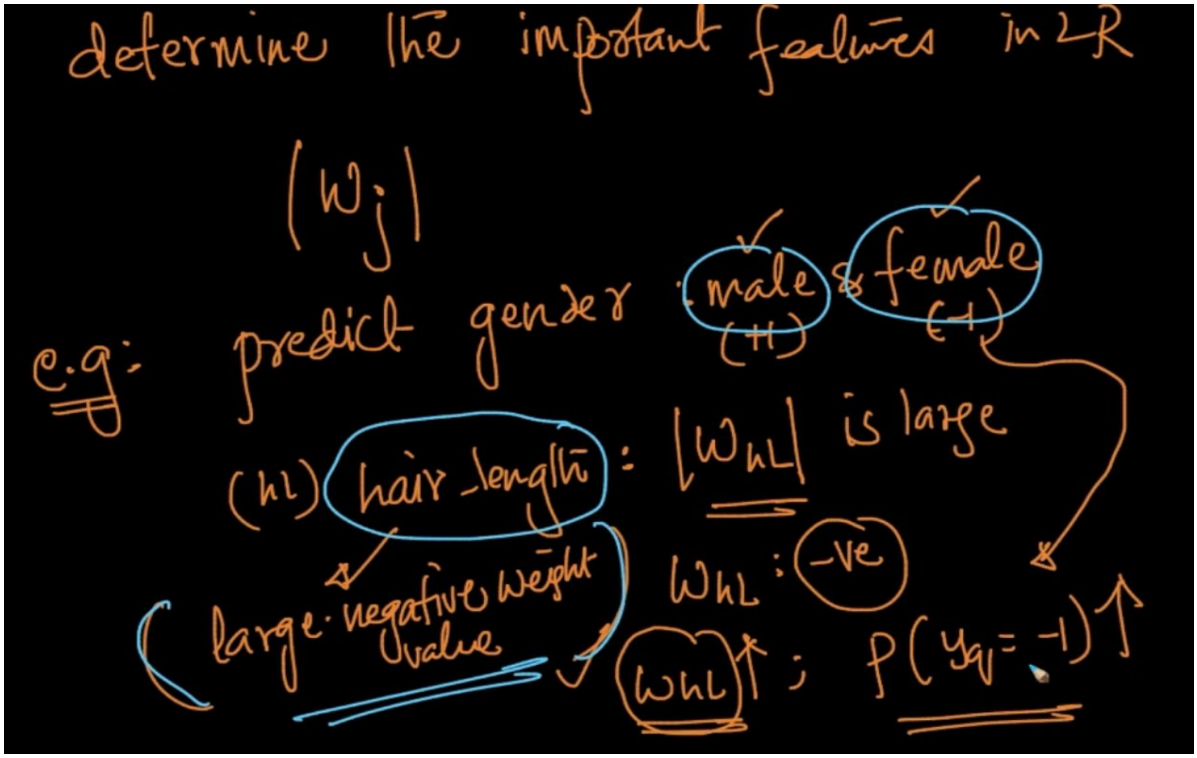


We can decide the important features by using their corresponding w's

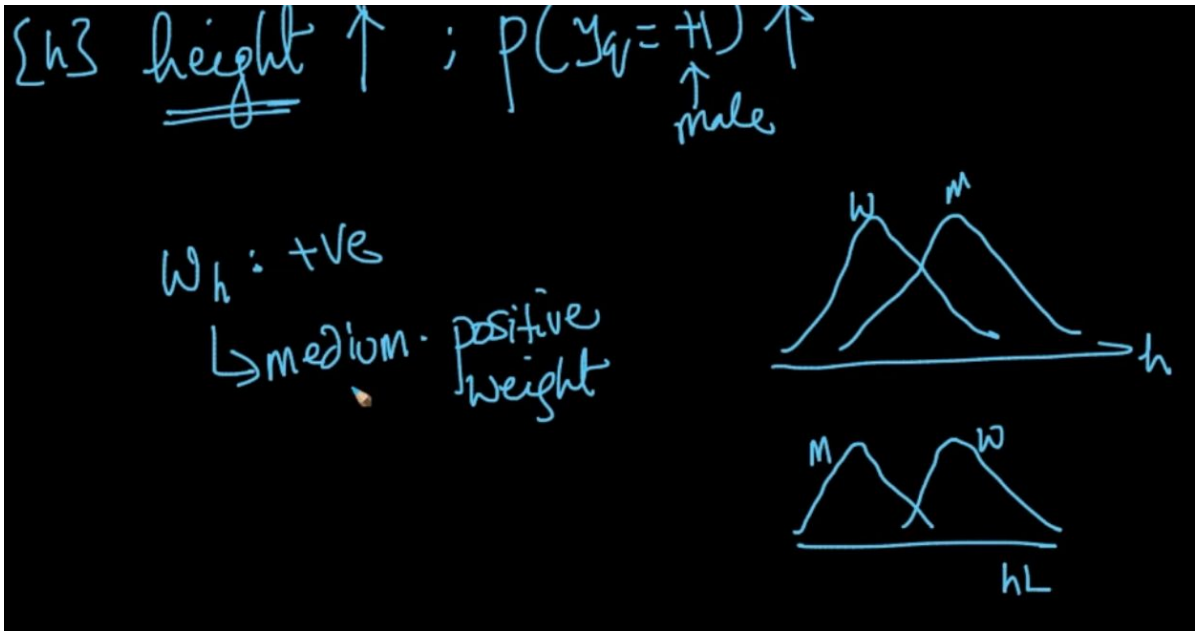
Note- We assume here that all features are independent just like in Naive Bayes



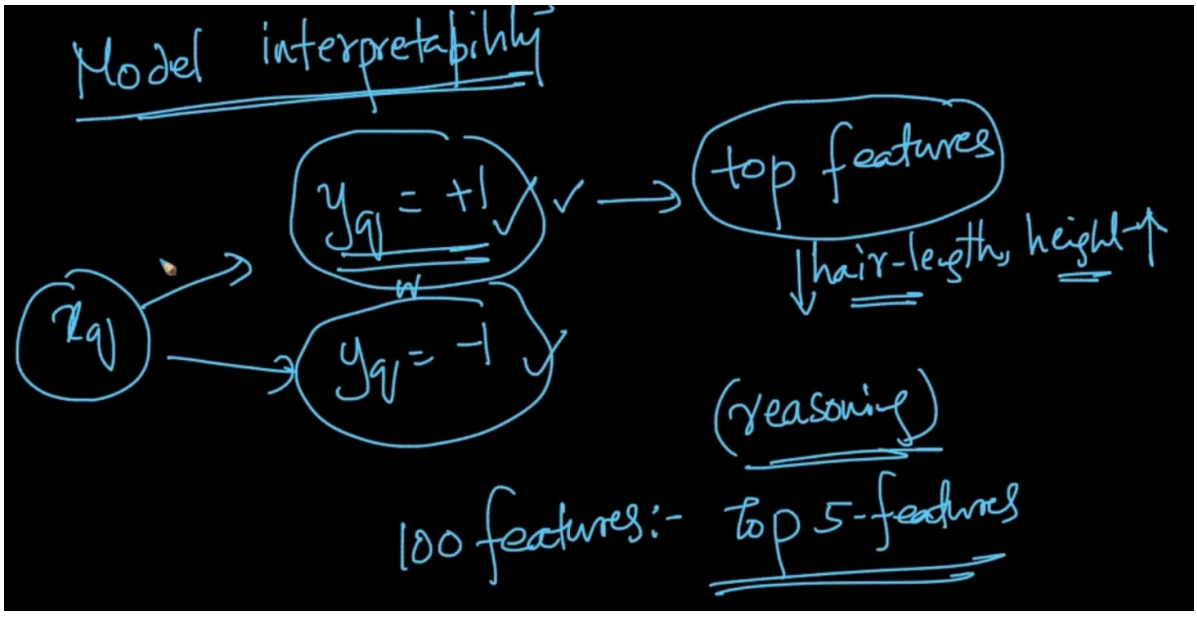
Feature importance of feature f_j is dependent on absolute value corresponding weight vector w_j. If $|w_j|$ is large and positive .It'll also contribute to $w^T x_q$ therefore P(y_q = +1) is also increased. Same analogy for case 2



E.x: If we are predicting gender : Male = +1 and Female = -1 and we've a feature Hair Length(hL) and if $|W_{hL}|$ is large and hair_length has a large negative weight value because as W_{hL} increases P(y_q = -1) also increases so it's an important feature to classify male and female

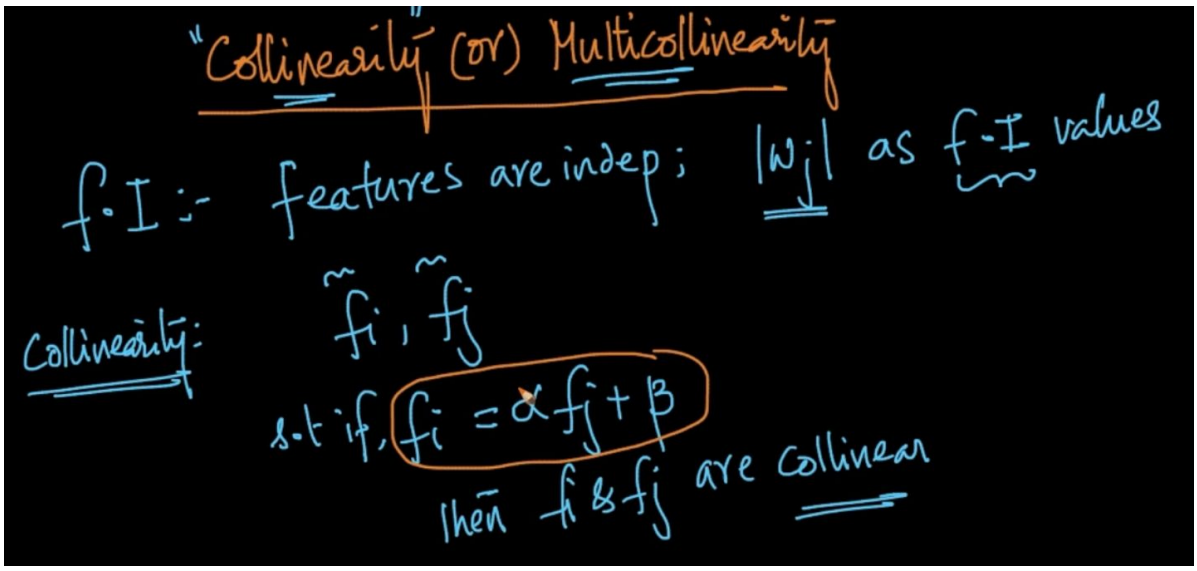


If we are taking height as feature , weight W_h : =ve but medium positive not large positive because if we plot the distributions between male and female hL has greater seperatibility But height doesn't have that

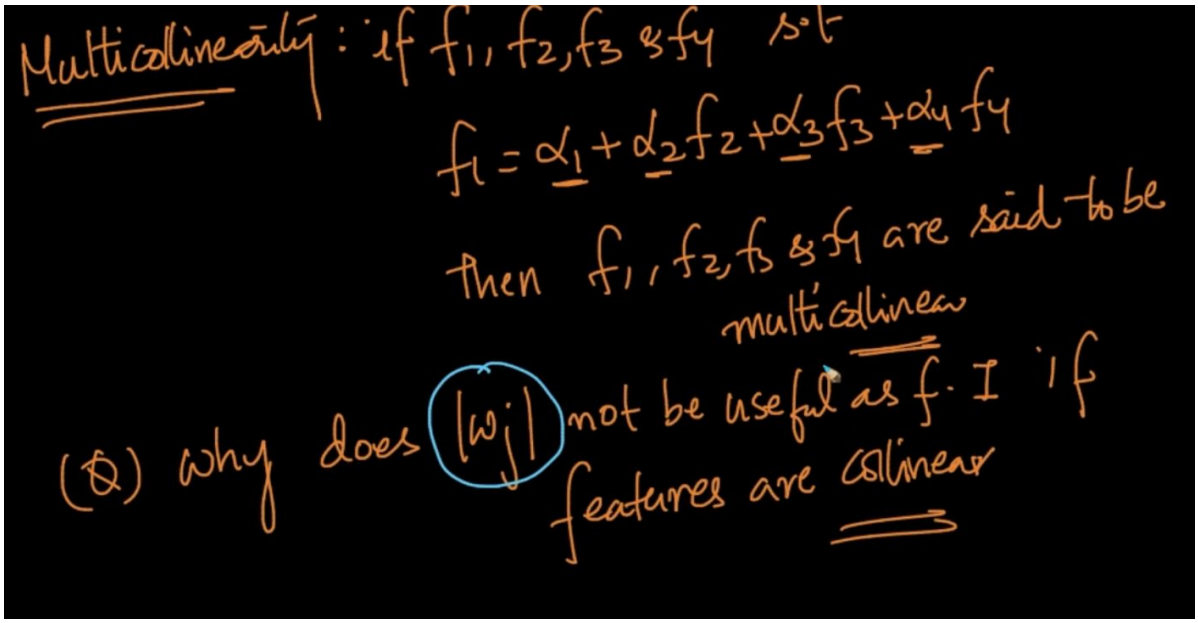


It also gives us Model Interpretability because if hair-length is low and height is high then we can say that y_q = +1 saying that it's a male. So we select the top features from the absolute value of the corresponding w. If they are large then that feature is important

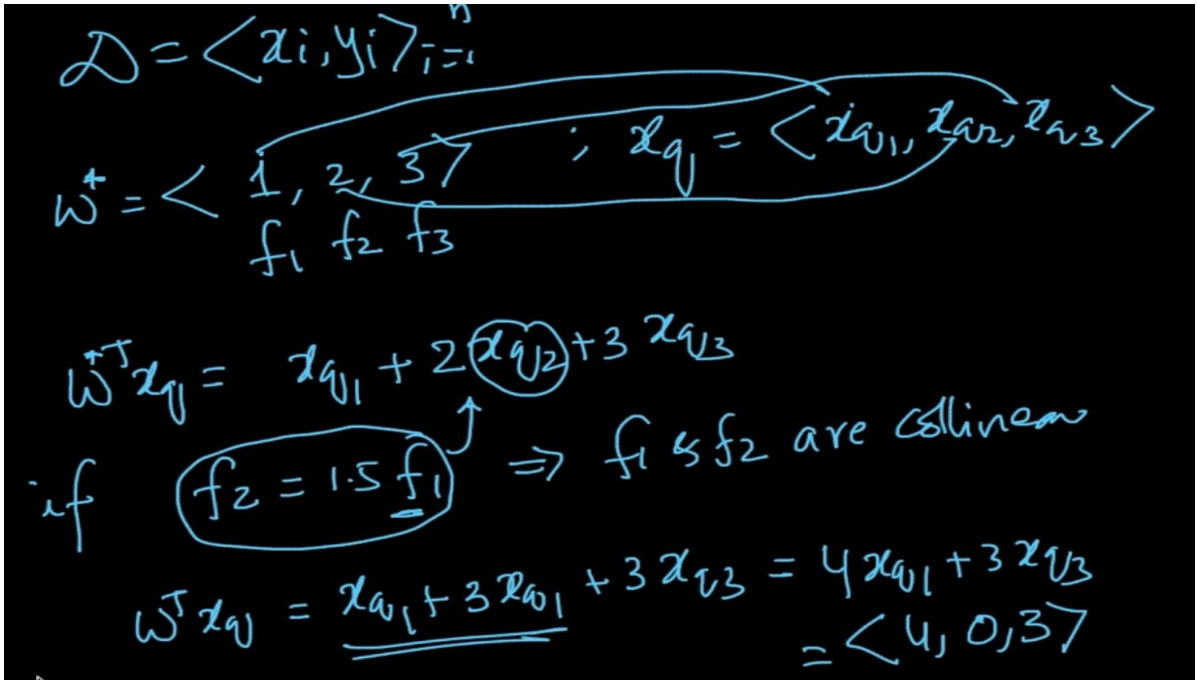
COLLINEARITY OF FEATURES



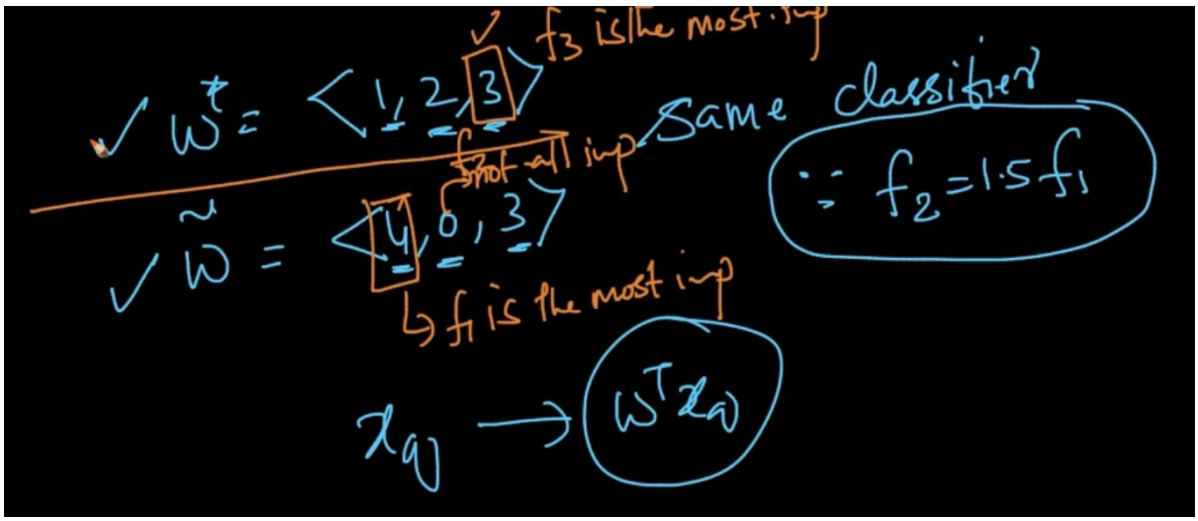
Features f_i , f_j are collinear if f_i = αf_j + β where α & β are constants



We'll see that $|w_j|$ for a feature f_j is not as useful for feature importance if features are collinear



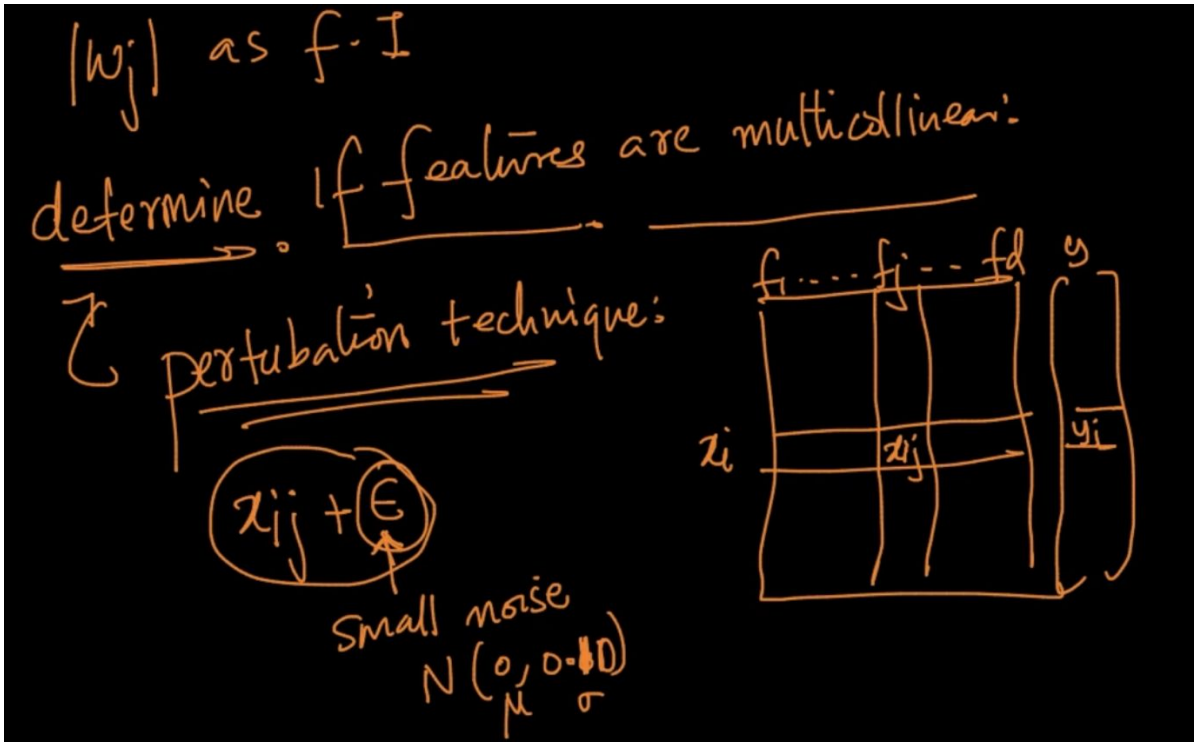
Our weight - vector w^* = <1, 2, 3> here but f_2 = 1.5 f_1 therefore they are collinear so our weight vector will get converted to $\overline{w}$ = <4, 0, 3>



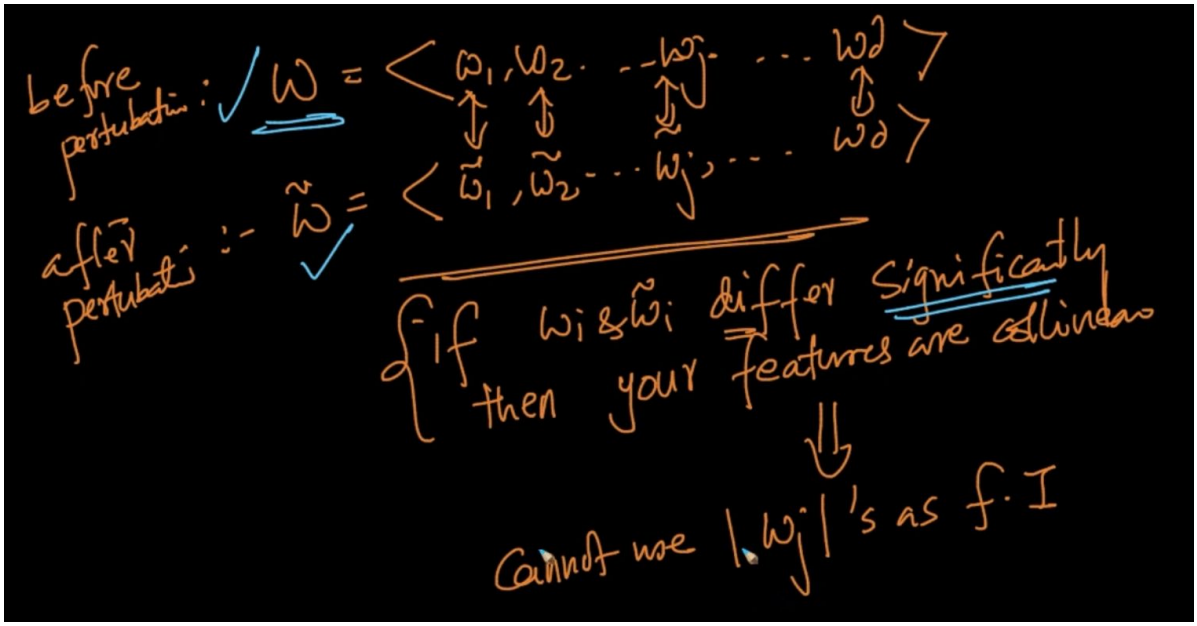
In w^* , f_3 is the most important feature as it has the highest absolute weight vector value

But in $\overline{w}$ f_1 is the most important feature and both w's give the same classifier since features are collinear

NOTE - If features are collinear then weight vectors can change arbitrarily so |w| cannot be be used for Feature Importance as seen from the example above

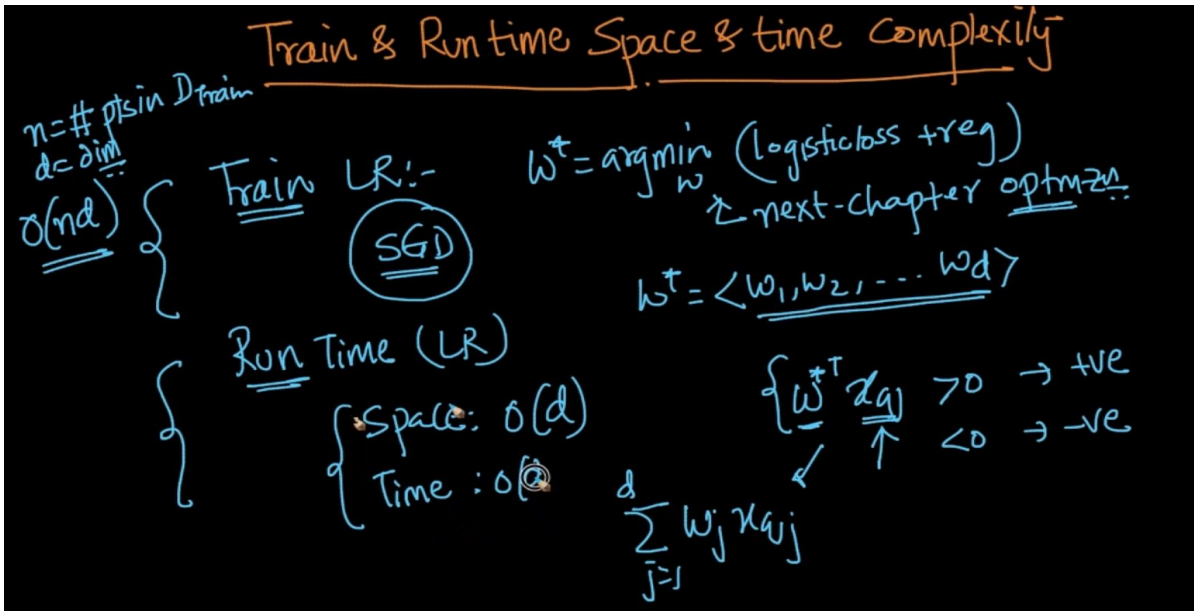


To determine the features are multicollinear we use pertubation technique in which we add a small noise ε to the data point x_{ij} which has small variance

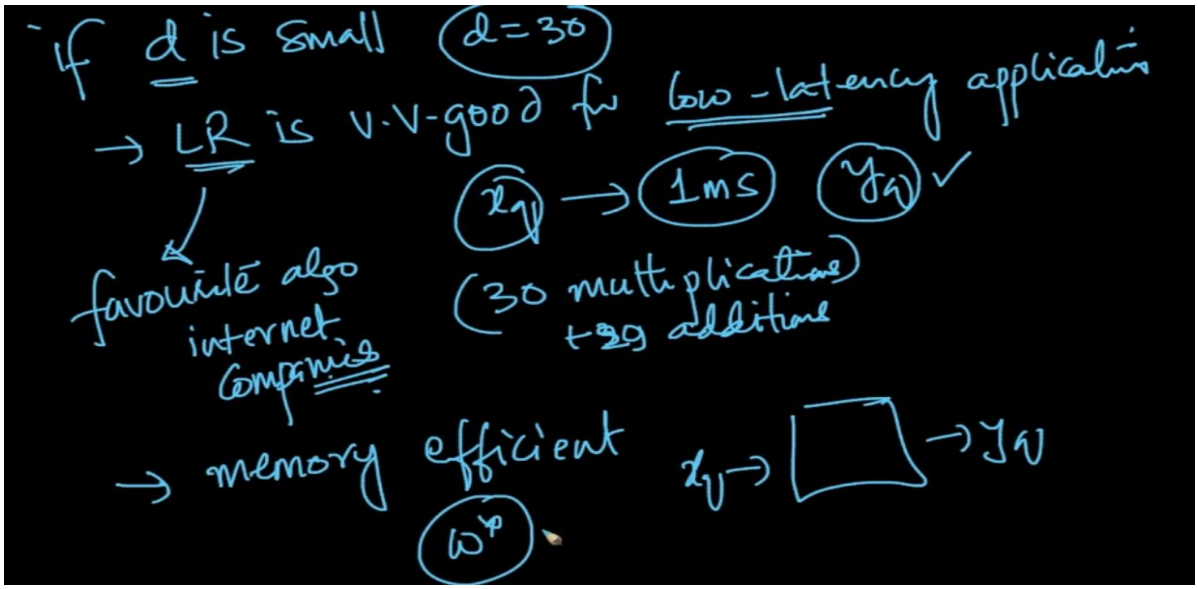


So we compare the weight vectors before and after the perturbation technique. If the difference is very significant then the features are collinear. So | w_j | cannot be used

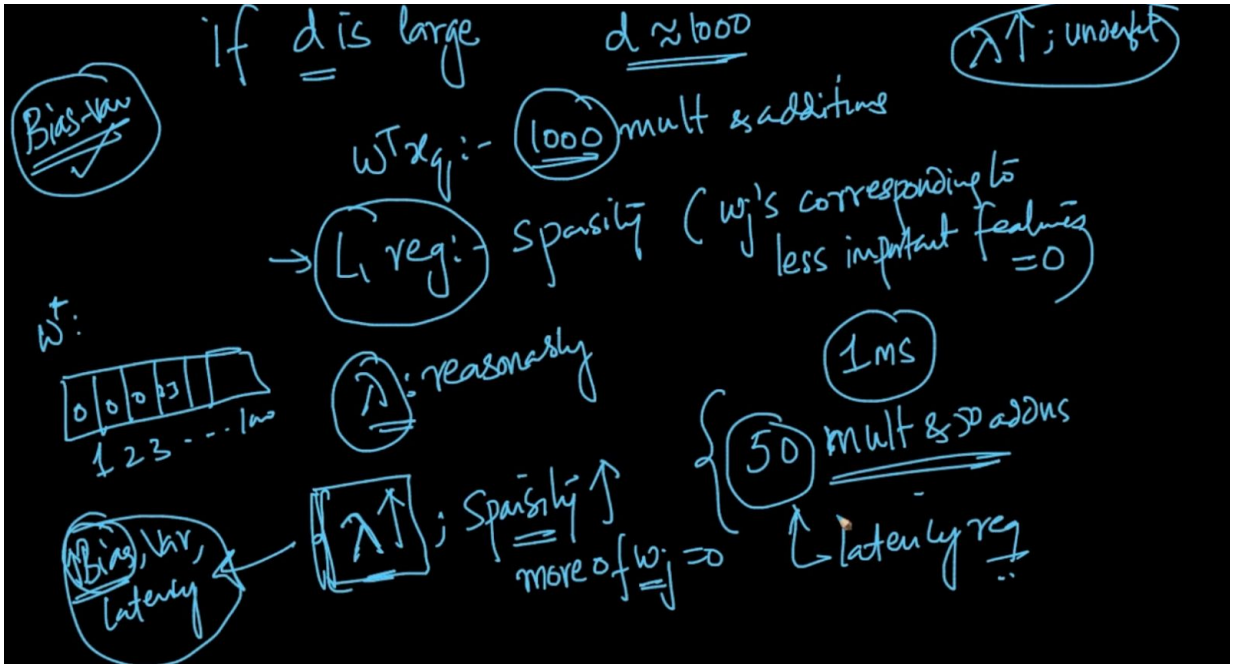
TRAIN & RUNTIME SPACE COMPLEXITY



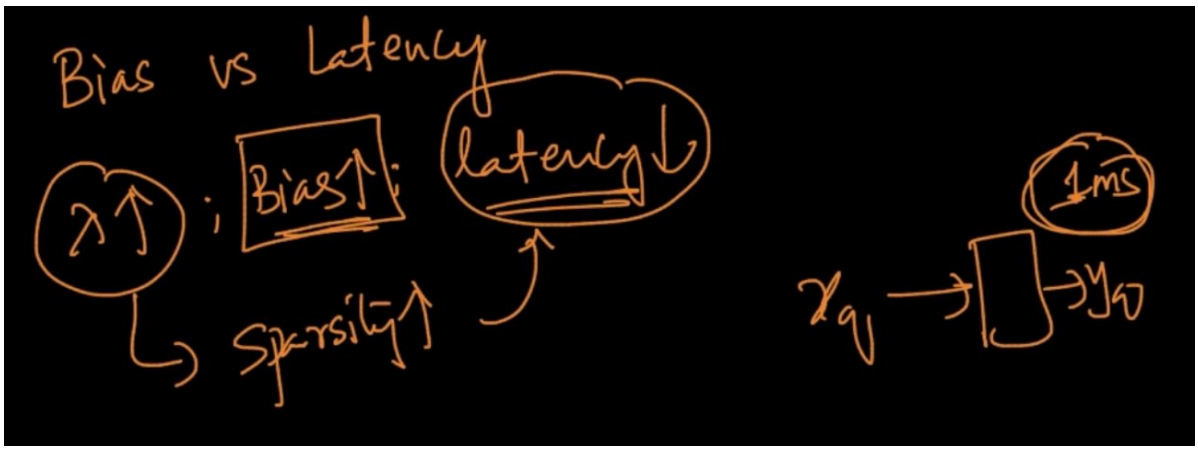
In Runtime, we need to only store the optimal w^*, if it has d- dimensions then it's Space complexity will be O(d) because x_q is given at run time. It's time complexity will also be O(d) as only d - multiplications and d-1 addition



If d is small, Logistic Regression is very good for low latency. Ex: If d = 30 then 30 multiplications and 29 additions. It's also memory efficient as only need to store w^*

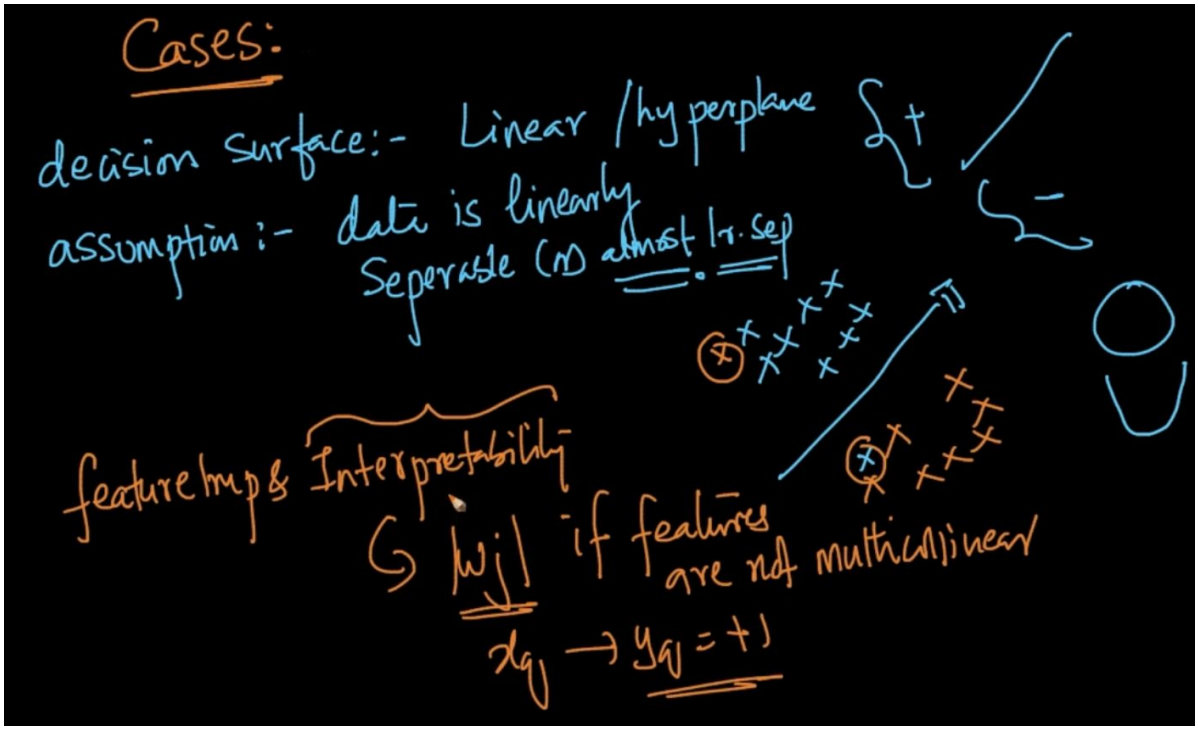


If d is large then there'll be slightly more time as 1000 multiplications and additions but we know that L_1 regularization we can create sparsity i.e w_j corresponding to less important features = 0. If the required latency is only achieved by 50 multiplications and 50 additions per microsecond then we increase the λ as increasing the λ increases sparsity but it means also increase in bias means underfitting . So, we need to find the reasonable λ

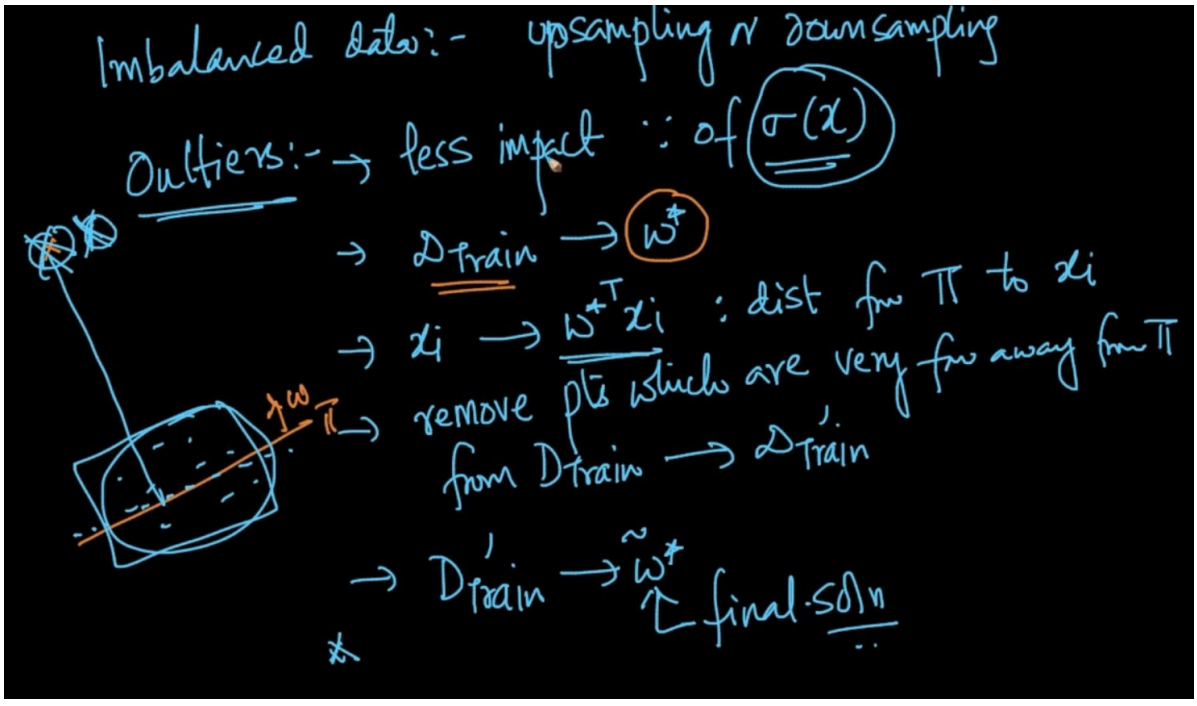


So there's a tradeoff we do between Bias and Latency that we've to do to achieve our goals.Logistic Regression one of the most popular algorithm since it has low Latency

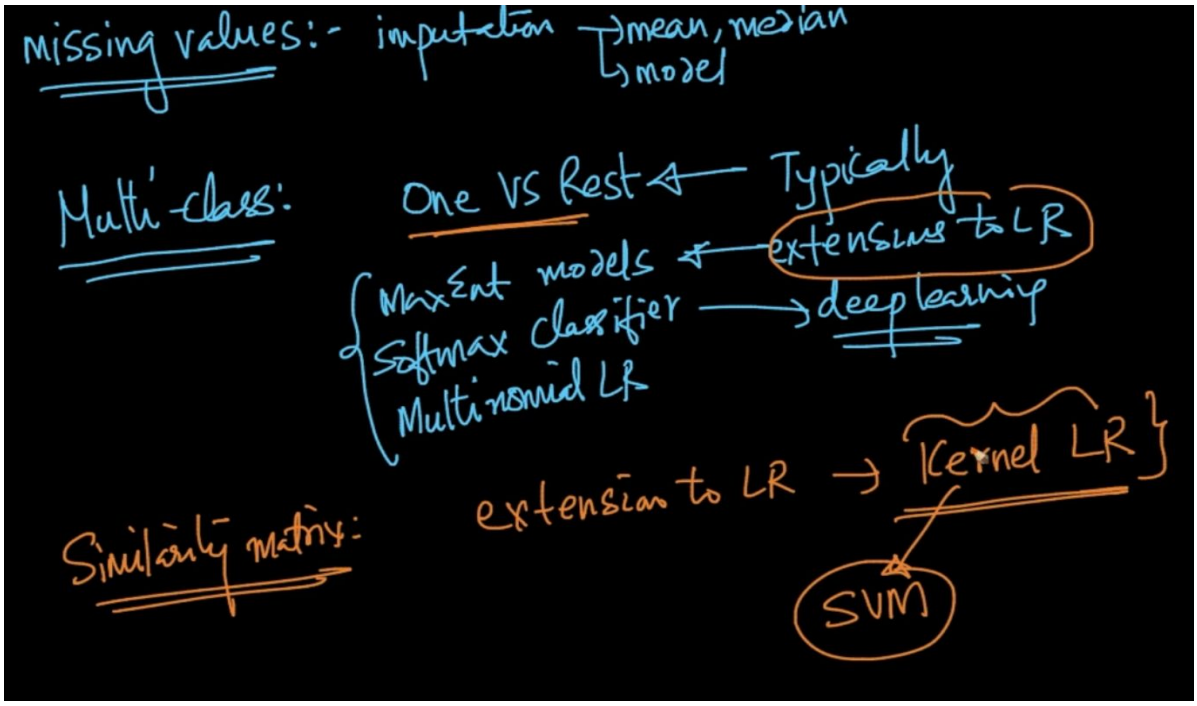
REAL WORLD CASES



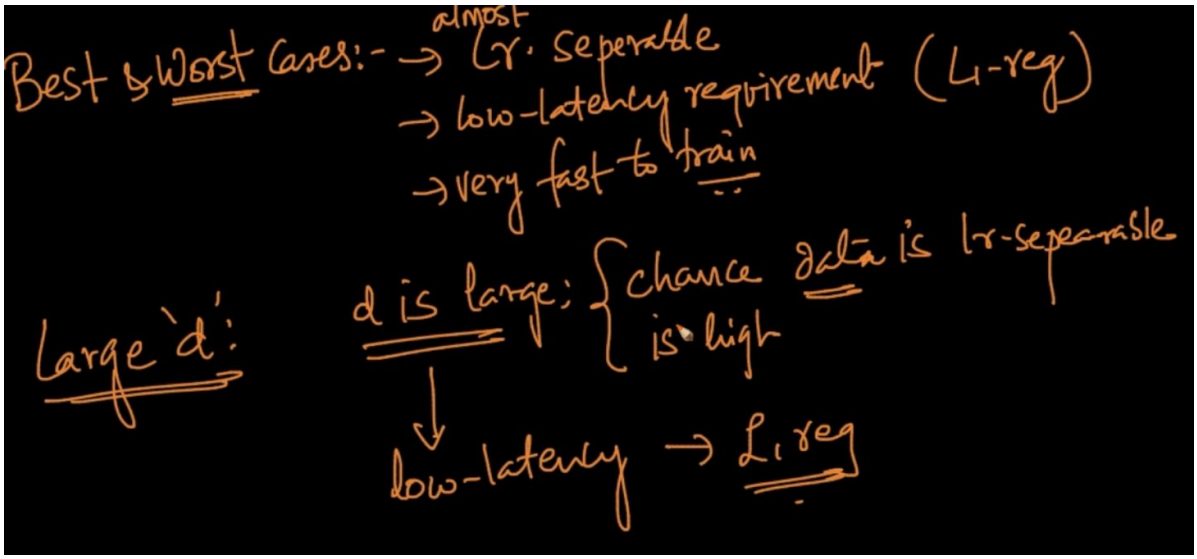
Assumption :- Our data is linearly separable. Feature Importance: $|w_j|$ i.e it depends on the absolute value of the weight vector i'th dimension but our features are not multi-collinear



If we've outliers then also it has less impact because of sigmoid function. We'll get the optimal w^*. If the data point x_i has great distance from hyper-plane π then we remove that data point and then train again . Our new $\overline{w*}$ is our final solution

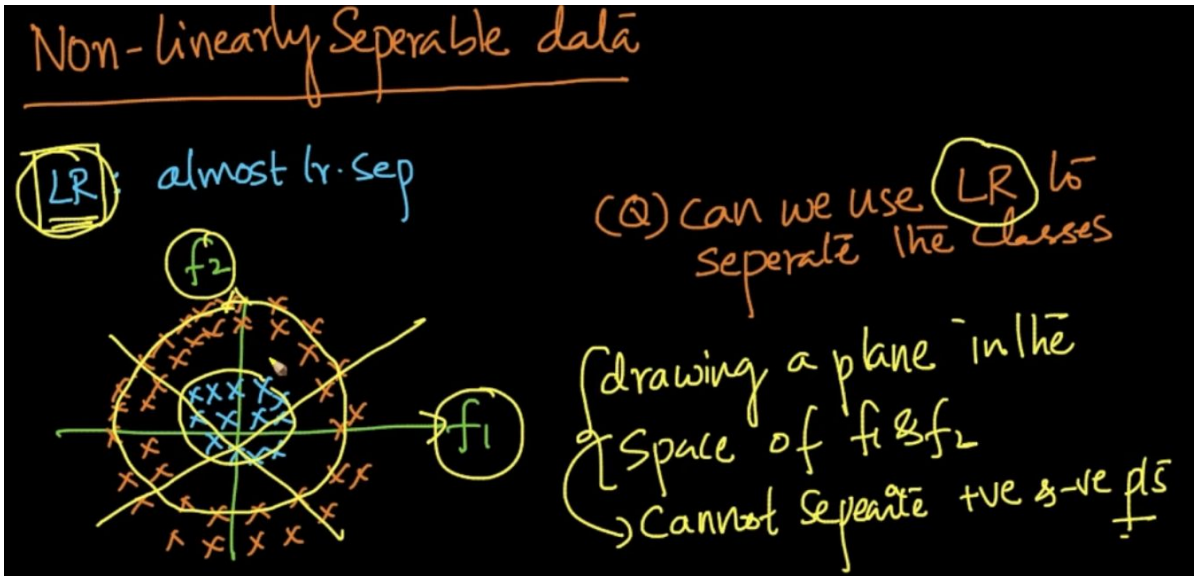


Other cases where Logistic Regression can be applied

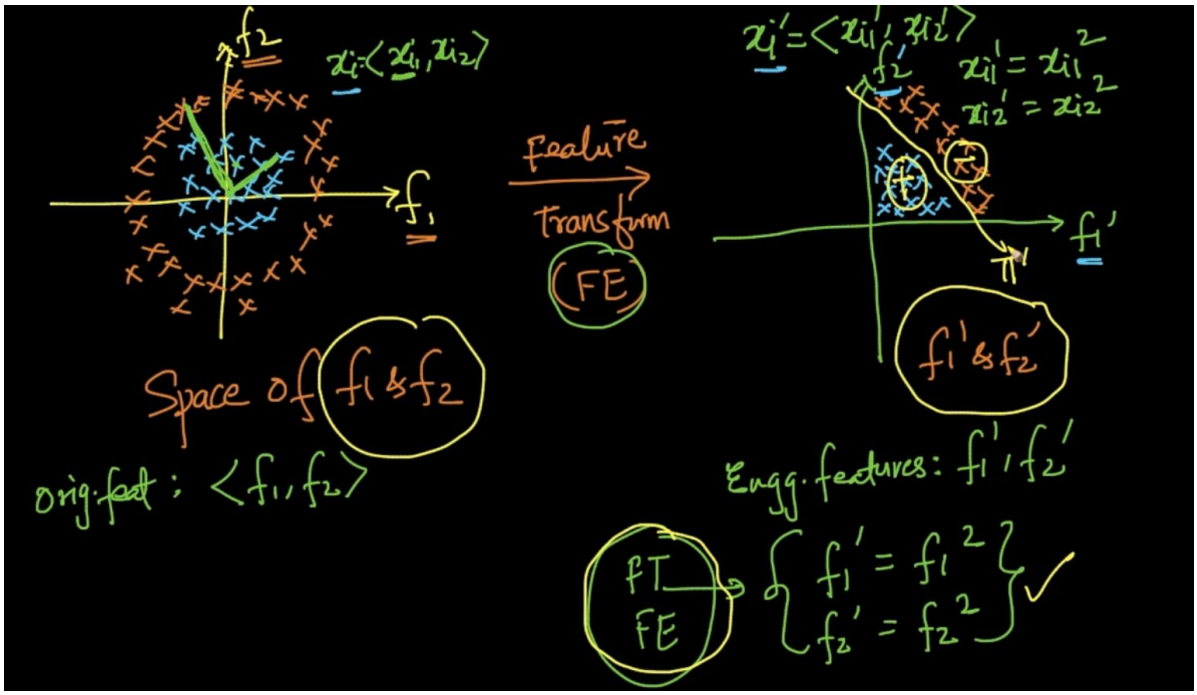


We've seen the best cases above but even if 'd' is large then chance that data is linearly separable is high and if requirement is low latency then L1-regularization

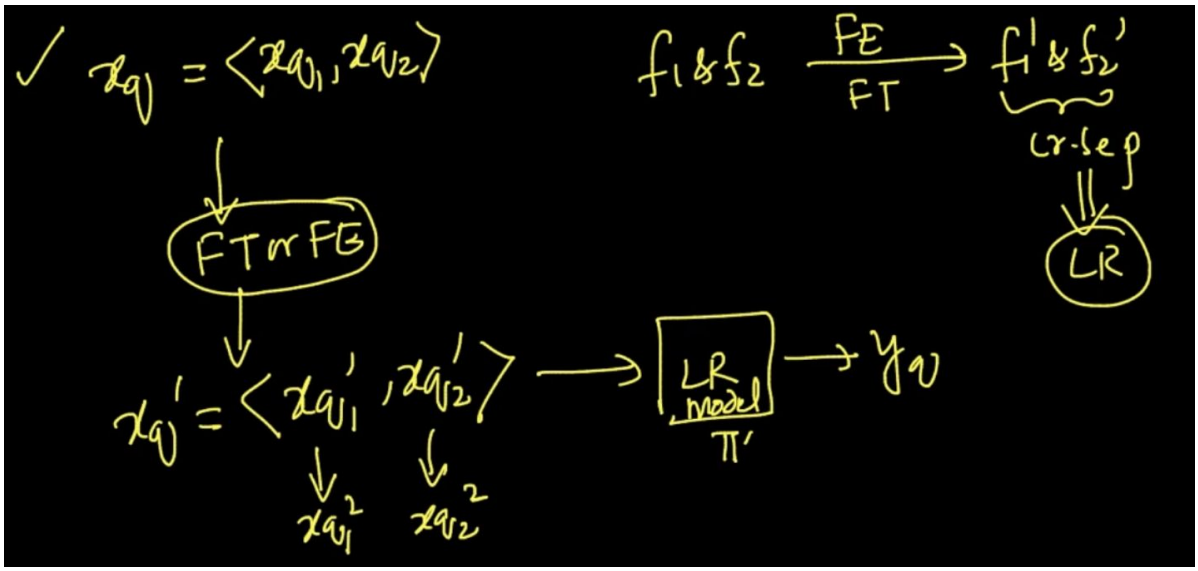
FEATURE ENGINEERING



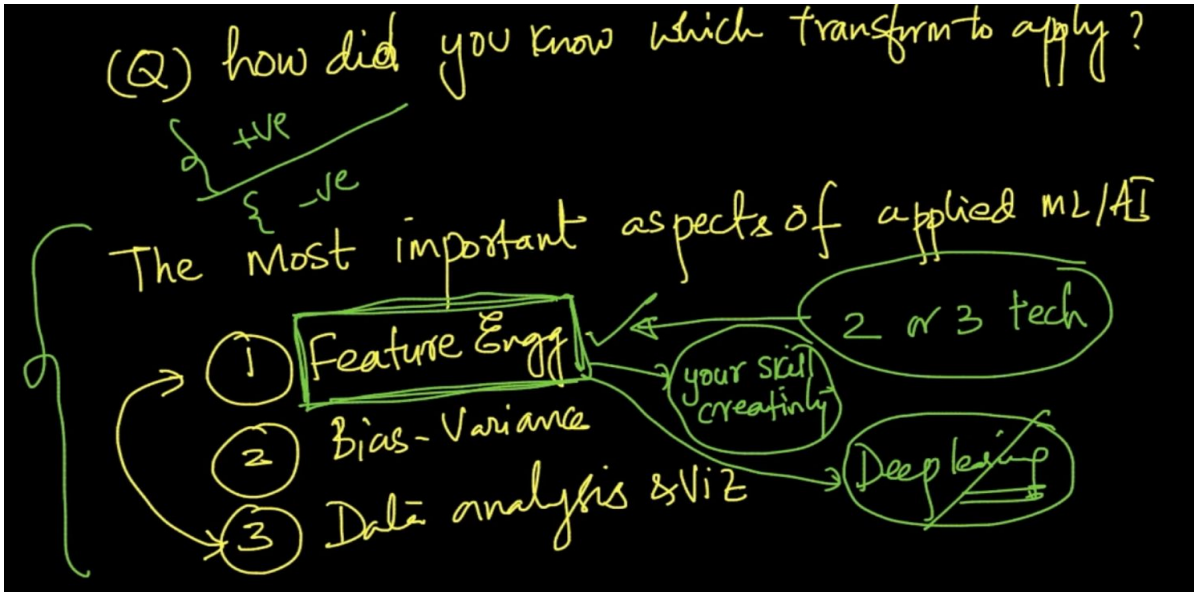
We apply Logistic to the data which is Linearly separable but in the above data Drawing a plane/line cannot separate +ve and -ve points. So, the question is can we apply LR to separate the classes?



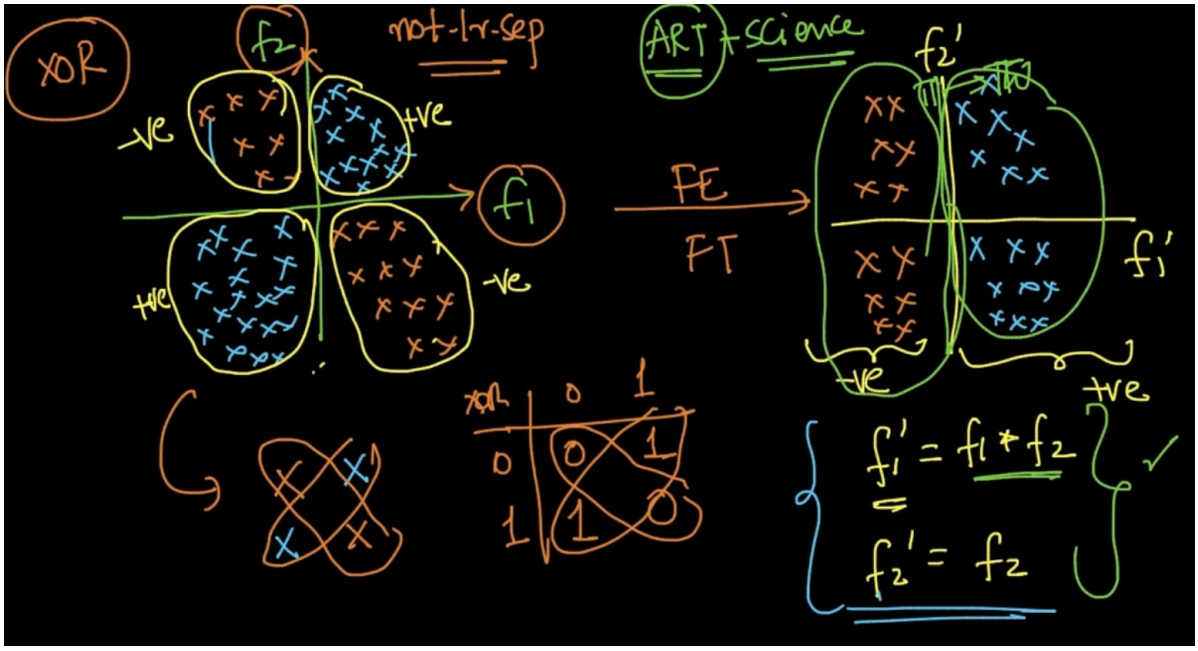
We are applying Feature Transformation by taking the features to a different space f'_1 and f'_2. Here, $f'_1 = f_1^2$ and $f'_2 = f_2^2$. As the distance of -ve point is more than +ve points they'll be clustered far away as seen but now they can be separated with a hyper-plane π. So Logistic Regression definitely works here



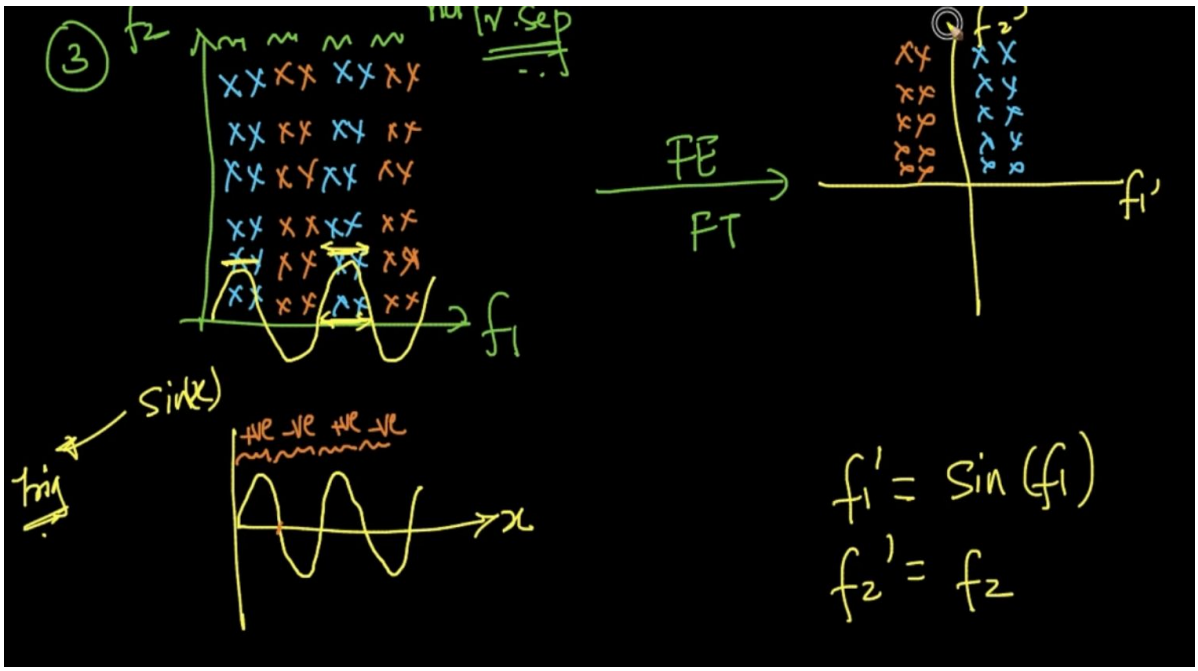
Feature Engineering is done on the features taking them to a different space which are linearly separable and Logistic Regression can be applied there



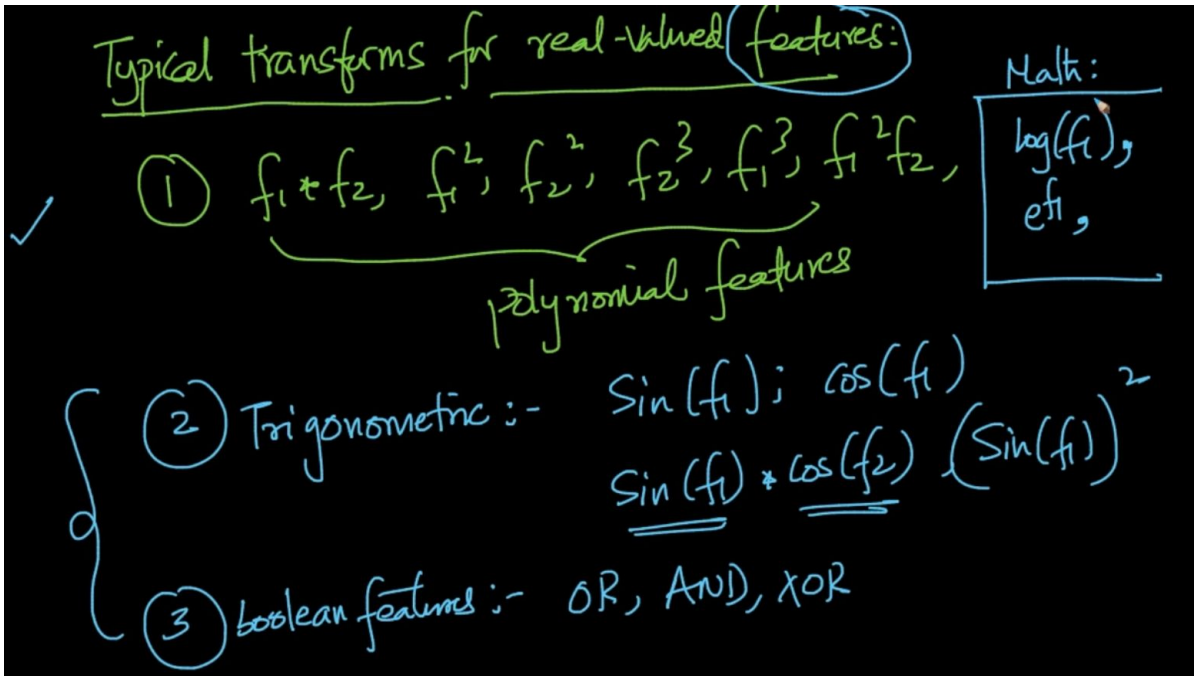
Feature Engineering is the important aspect of applied Machine Learning



This is XOR dataset. To apply feature engineering , f1' = f1 * f2, f2' = f2. When we do this all the blue points become +ve and the orange points become -ve so can be linearly separated



Here, feature engineering is done by applying sin() function as we can see all the blue points are in positive region of sinusoidal wave and orange points in negative region . So, it can be separated after that as seen above



These are the typical transformation methods applied in Feature Engineering.