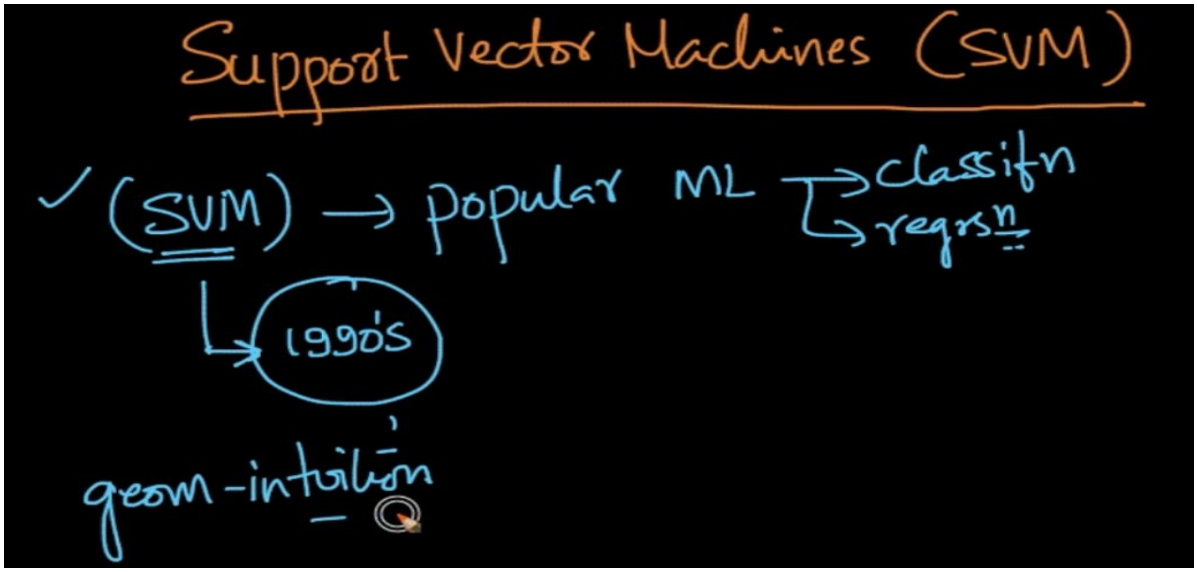
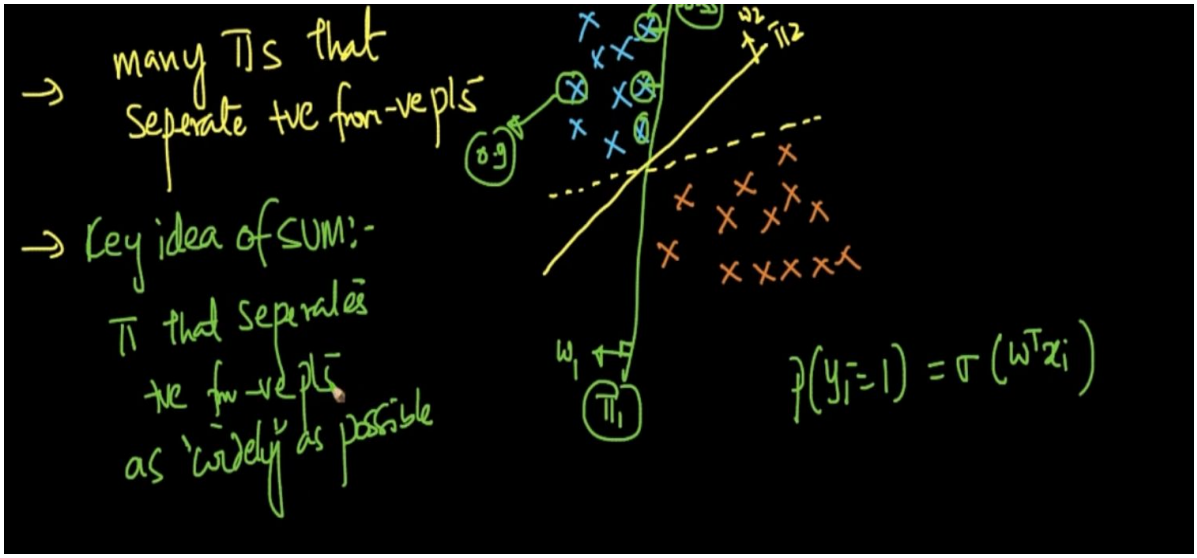
SUPPORT VECTOR MACHINES

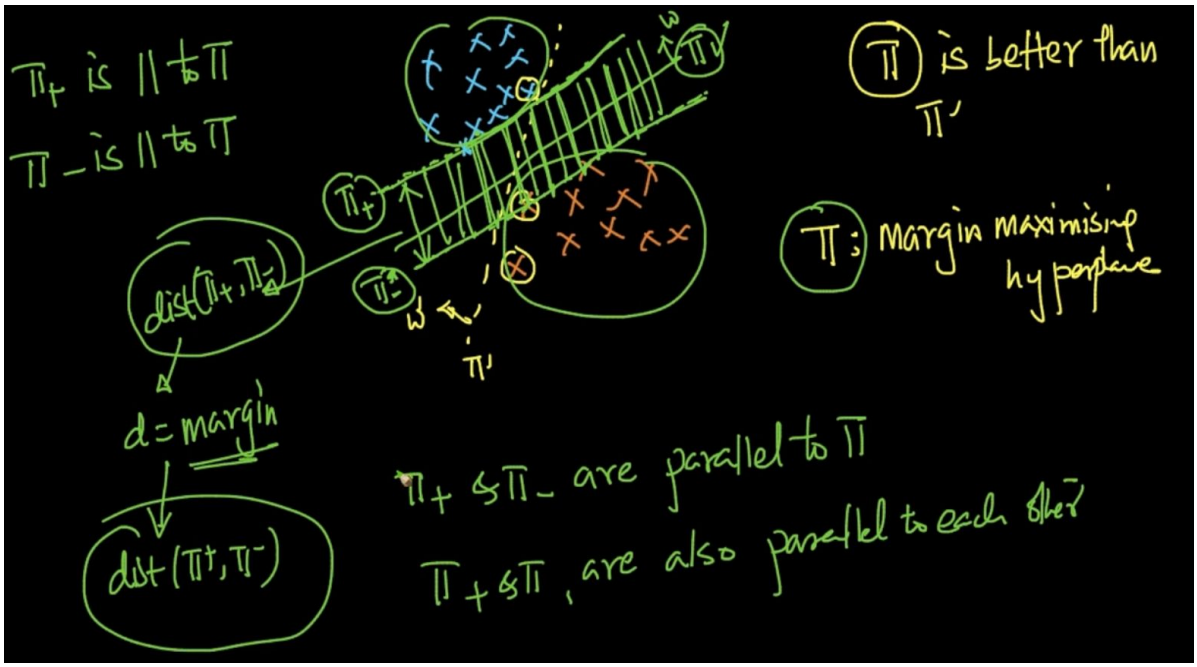
GEOMETRIC INTUITION



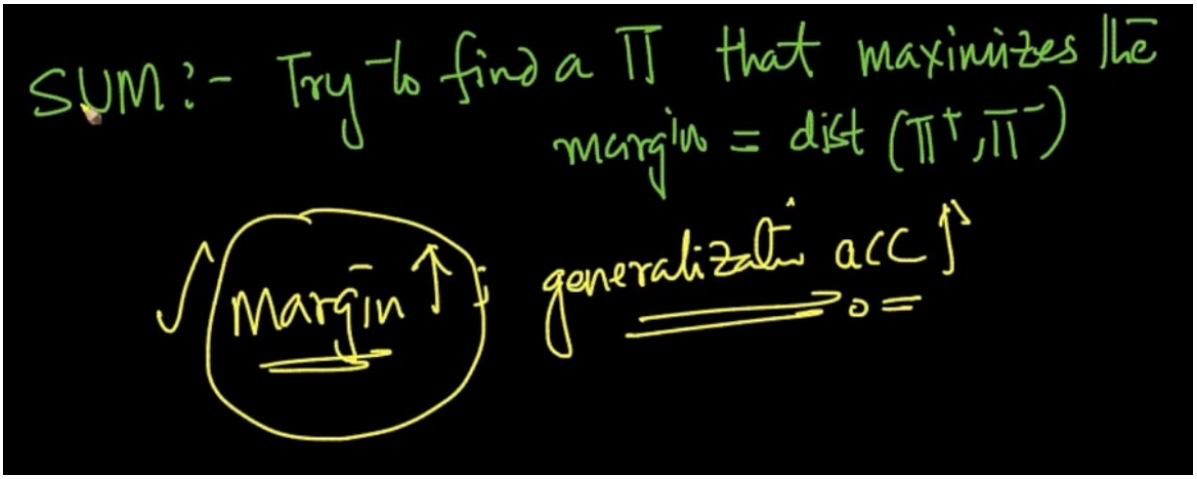
Support Vector Machines is a popular ML algorithm used for both Classification/ Regression



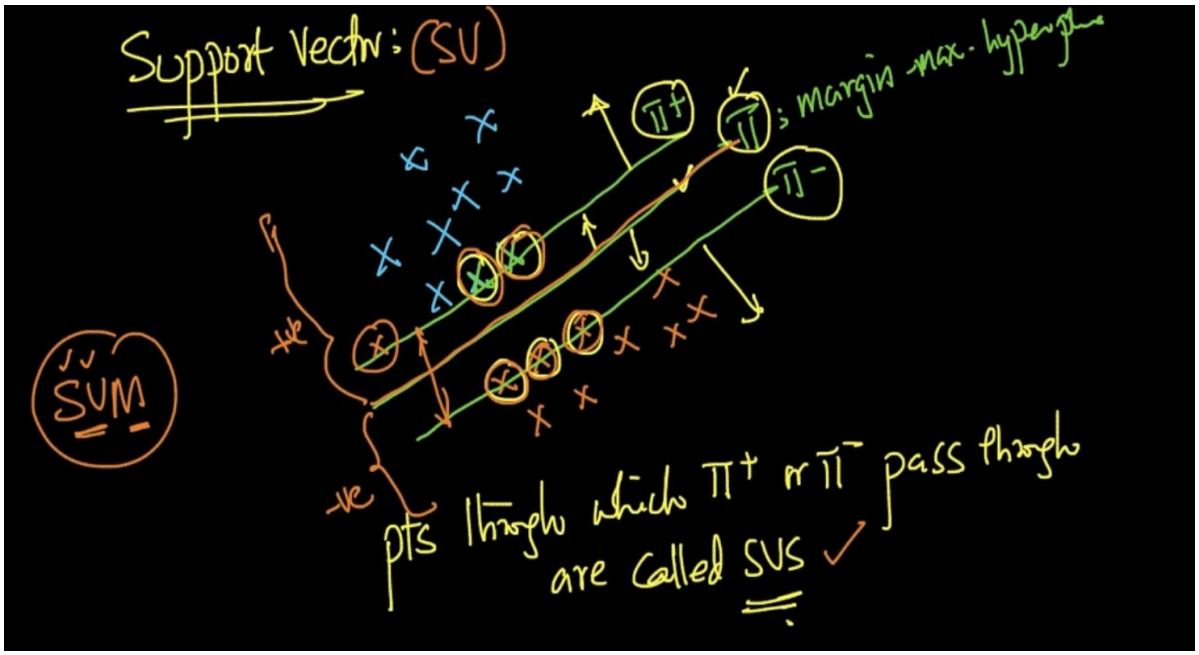
Dataset can be separated using many hyper-planes but if we take π_1 the closest points will've lesser probability if we take something like sigmoid function which takes distance into account. SO our SVM wants a π that seperates +ve and -ve as widely as possible



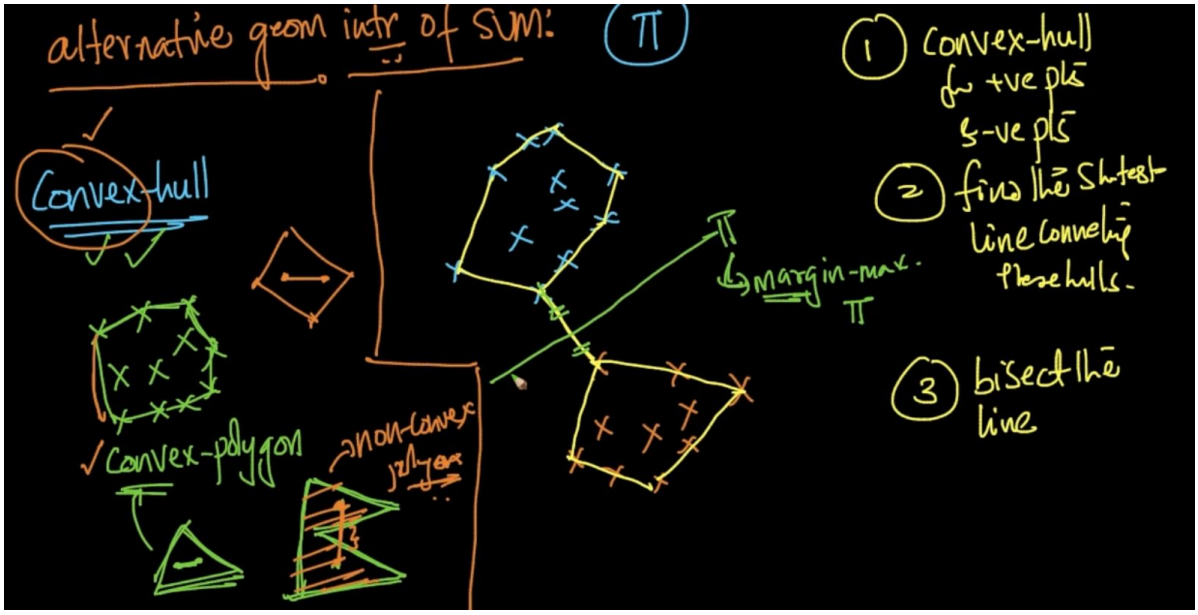
Our green line π is the margin maximizes hyperplane. If we take parallel lines which touch the *first points* in the +ve and -ve points. We calculate the distance which is known as margin between that two planes (π_+, π_-)



Svm tries to find the hyperplane π that maximizes the margin . If my margin is high then our chance of misclassifying decreases as margin increases our generalization accuracy i.e accuracy on unseen data increases

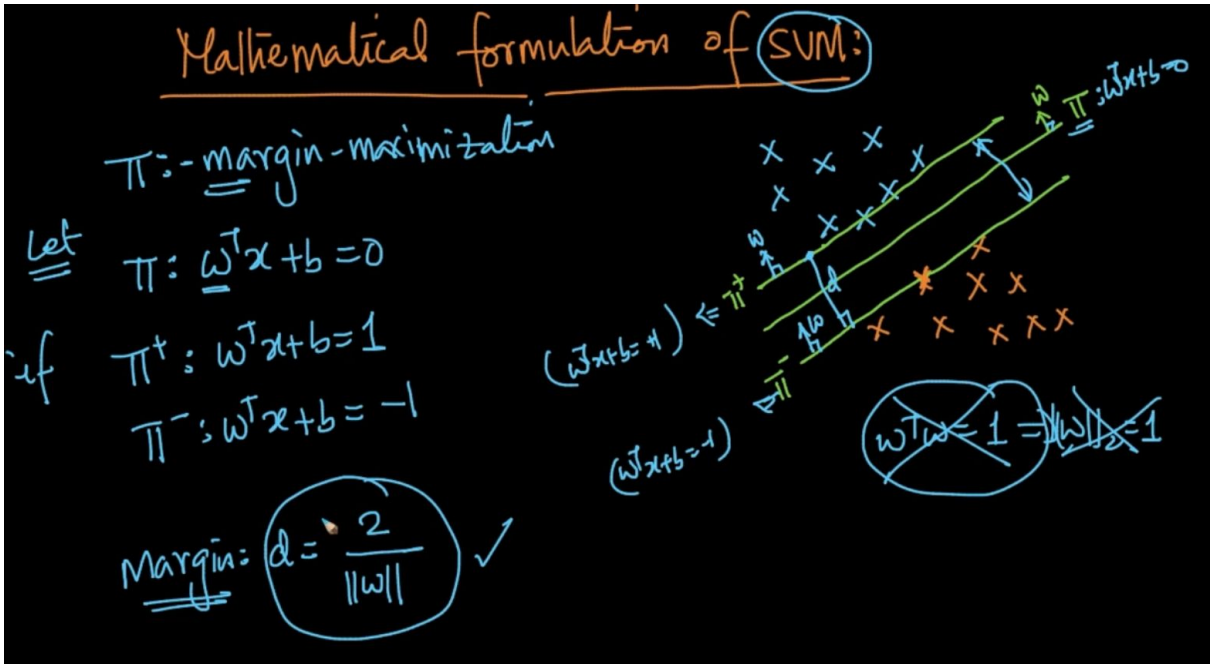


Support Vectors are the points through which π_+ or π_- passes through. If a new point comes it's classified on the basis of π: Margin max hyperplane and that's it

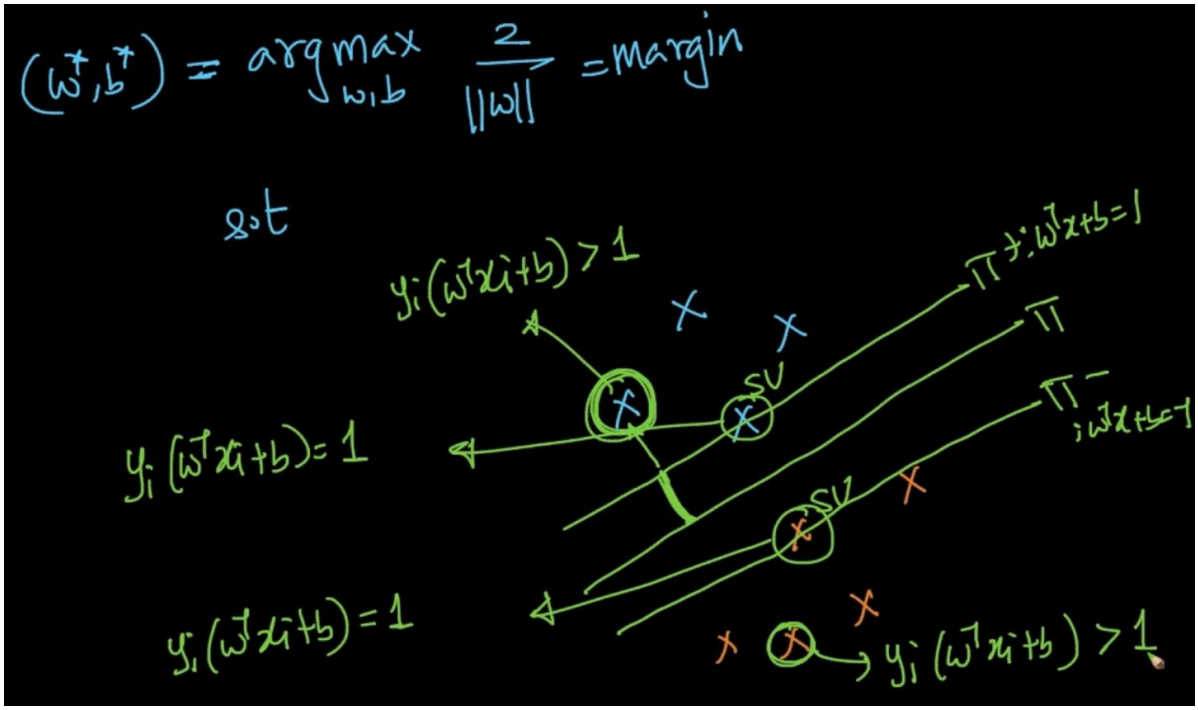


A convex hull is a shape inside which all the data points are there or the points are on the convex hull .Follow all the steps to perform given above to perform SVM

MATHEMATICAL DERIVATION

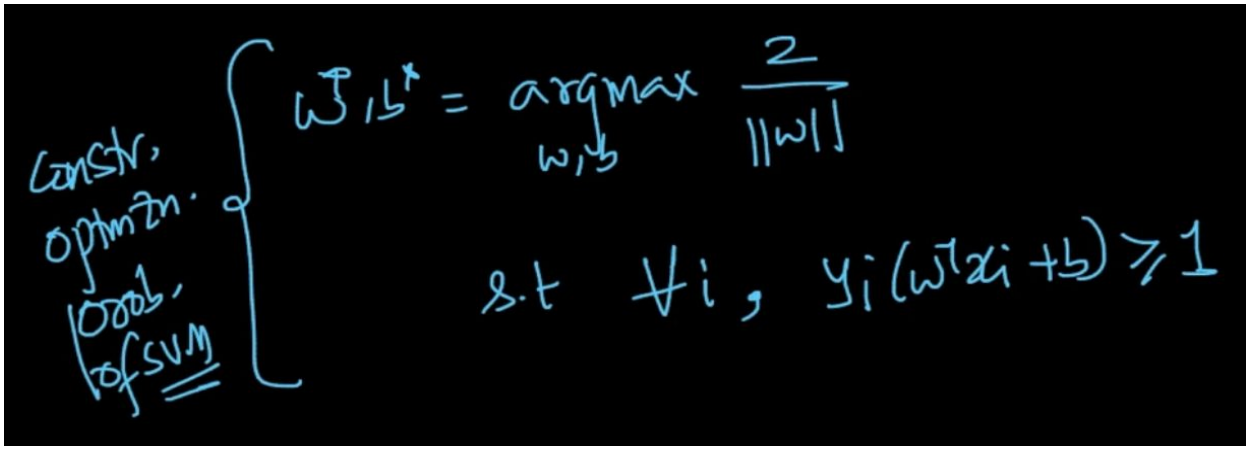


$\pi: w^T x$ + b = 0 and our $\pi^+: w^T x$ + b = 1 and $\pi^- : w^T x + b = -1$ so our margin i.e distance d = $\frac{2}{\|w\|}$. So we need to find (w^*, b^*) such that it maximizes $\frac{2}{\|w\|}$



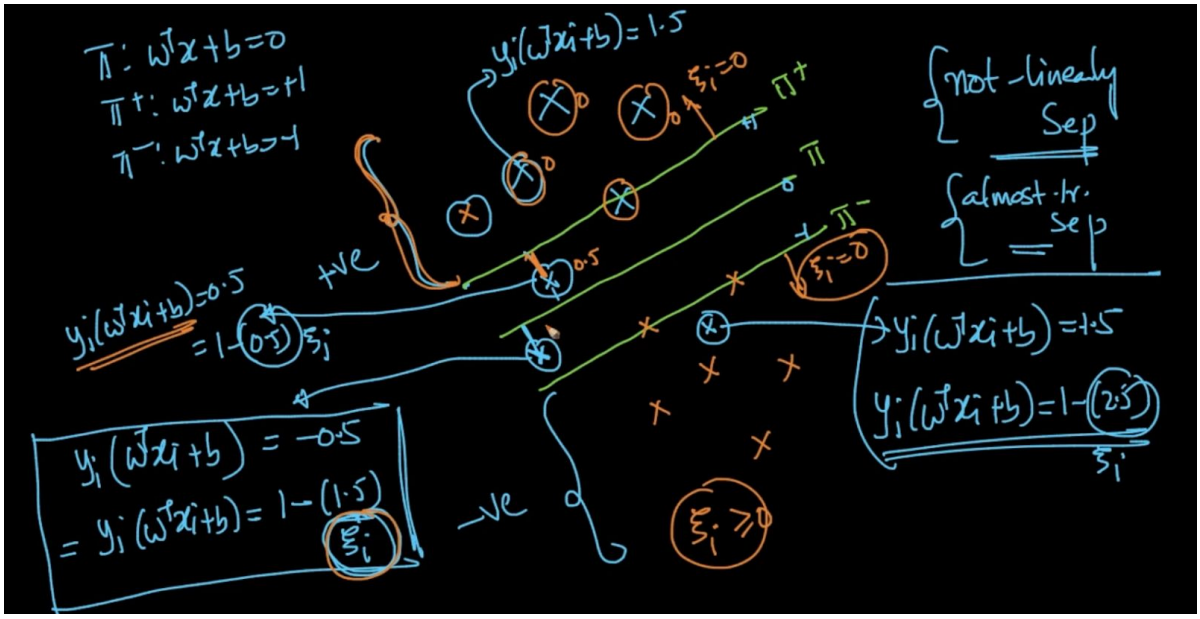
For support vectors : y_i ($w^T x$ + b) = 1

For data points greater than hyper-planes: y_i ($w^T x$ + b) > 1 as y will be -1 or +1 and ($w^T x$ + b) > 1 or ($w^T x$ + b) < -1



So we want to maximize the function with a constraint that for all i, y_i ($w^T x$ + b) $\geq$ 1

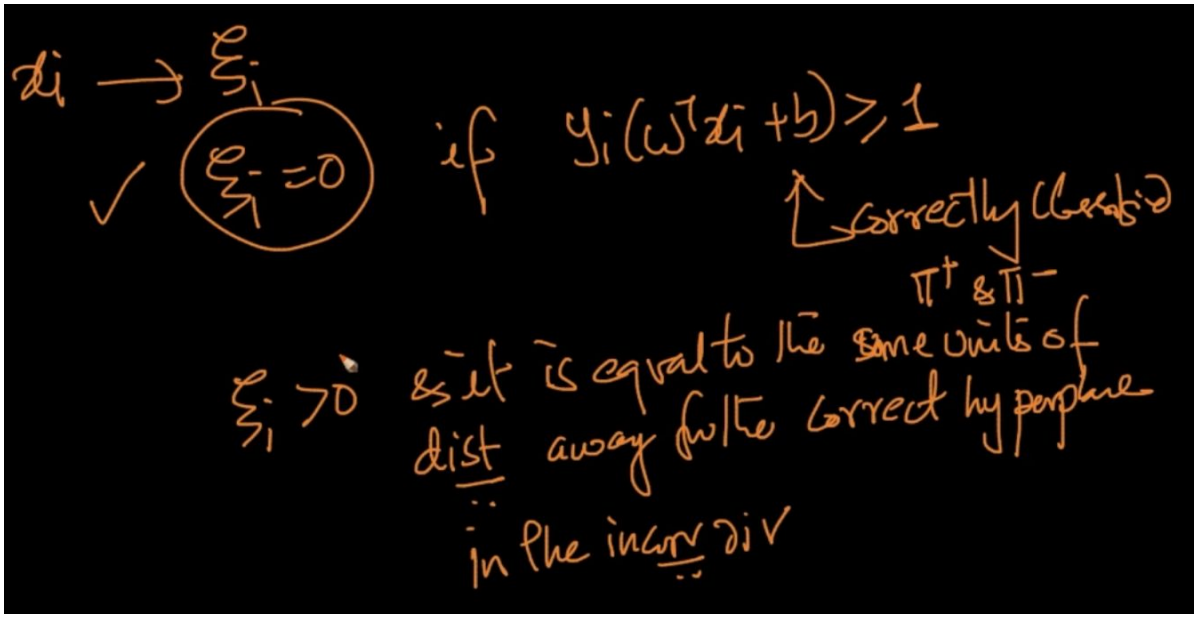
To satisfy this constraint our data should be linearly separable



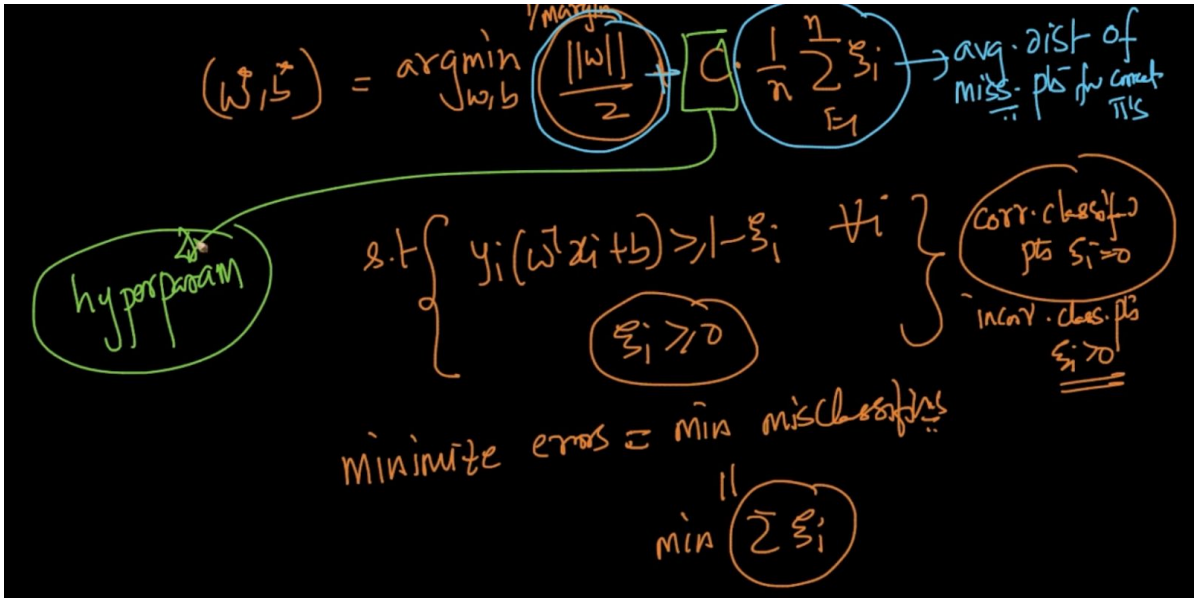
There can be cases like the above where a negative data is there in the positive part or vice versa. Here we'll never find the optimal w,b since it's not linearly separable.What do we do in these situations?

In the above , we can see that there's one +ve(blue) point amongst the negative data points. It's y_i ($w^T x$ + b) = -1.5 as y_i = 1 and ($w^T x$ + b) = -1.5 . It can also be written as y_i ($w^T x$ + b) = 1 - (2.5). Same logic can be applied for the points in between the hyper-planes as well . So that 2.5 can be our ζ . We use ζ for all the points that are not in the region of their correct class. It works like a loss ζ . In the region above the positive plane π^+ , ζ = 0. Same for the correctly classified -ve points in the negative hyperplane

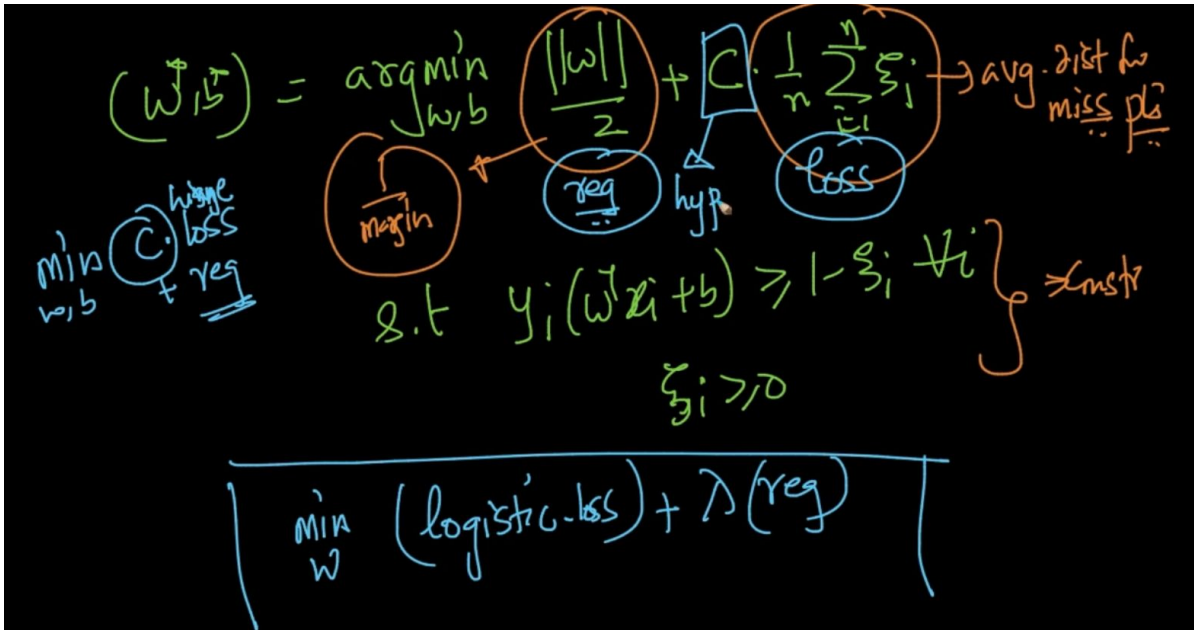
So, as $\zeta \uparrow$, the point is farther away from the correct hyper-plane . For ex: for the point which has ζ = 2.5 .2.5 is the distance from the correct plane



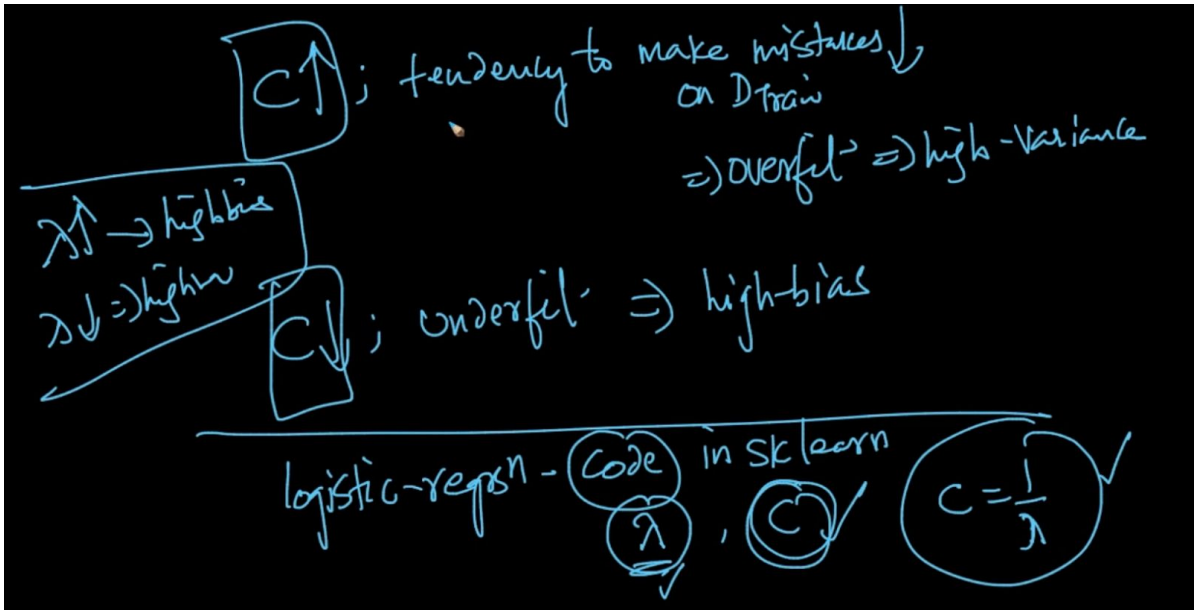
For every point x_i we've ζ_i and ζ_i = 0 if the point is correctly classified and ζ_i > 0 and it's equal to distance from the correct hyperplane and in the incorrect direction



Our margin was ||w||/2 and we want to maximize it/ So 2/ ||w|| is the inverse which we want to minimize. $\sum_{i=1}^{n} \zeta_i$: is what we want to minimae as well

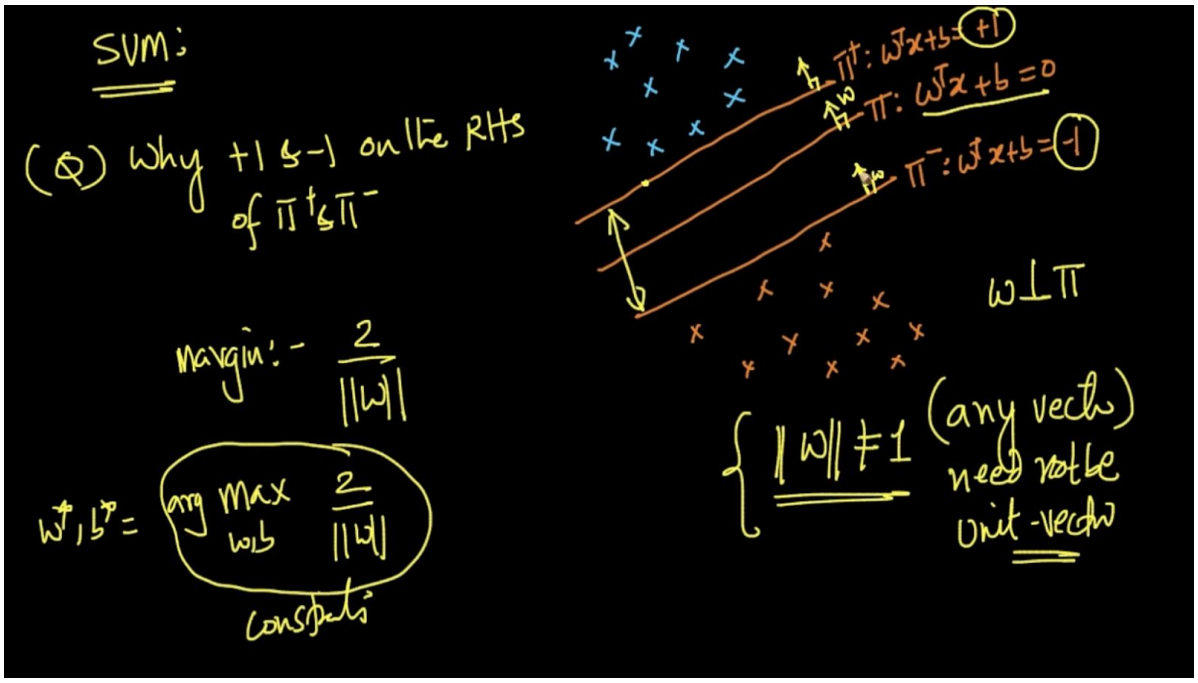


||w||/2 is our regularization and $\sum_{i=1}^{n} \zeta_i$: IS our Hinge-loss and C is hyperparameter

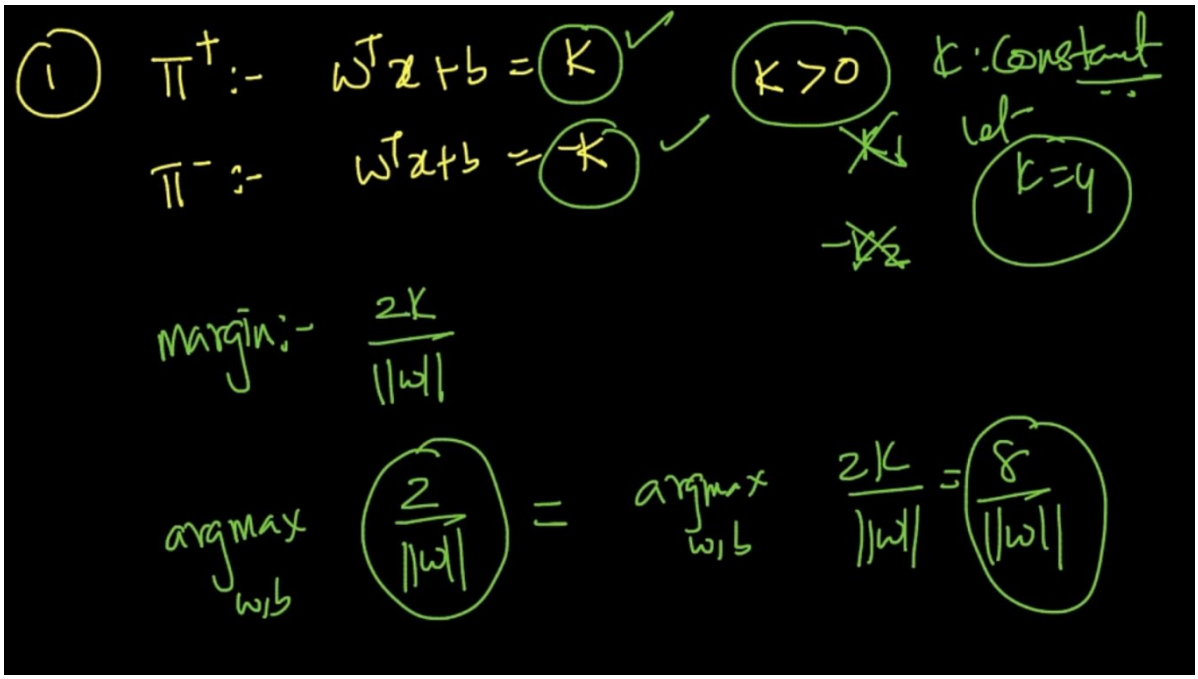


C = $\frac{1}{\lambda}$ so it works exactly opposite to λ as seen above

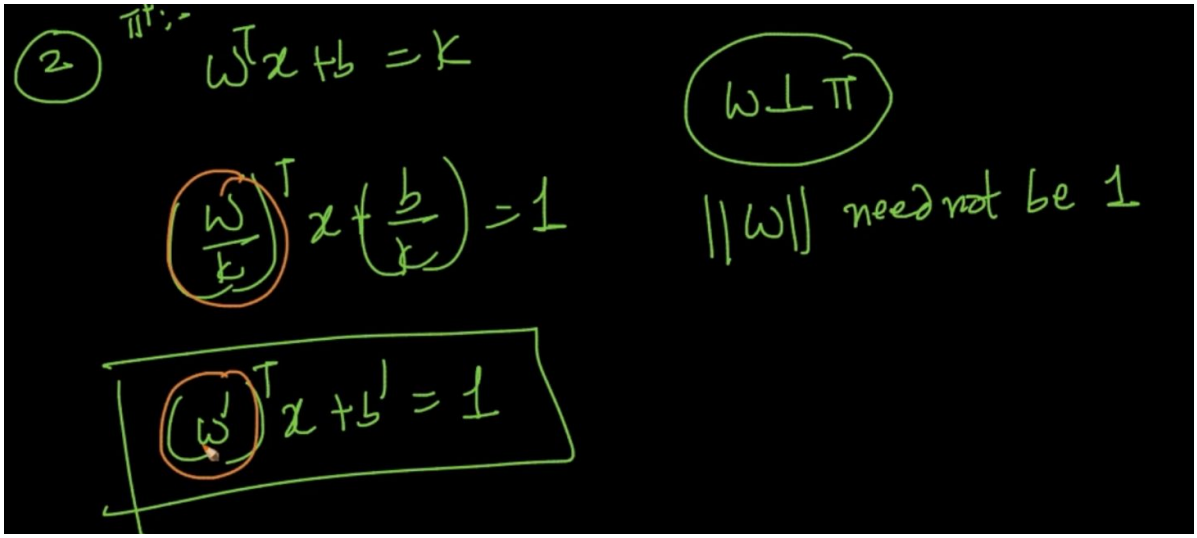
WHY WE TAKE +1 AND -1 FOR SUPPORT VECTOR PLANES



See, we said that $\|w\| \neq 1$ it just should be $w \perp \pi$.Remember our objective was to maximize the margin (2/ ||w|)| with some constraints which is basically that all the +ve points should be at one side and -ve points at other.



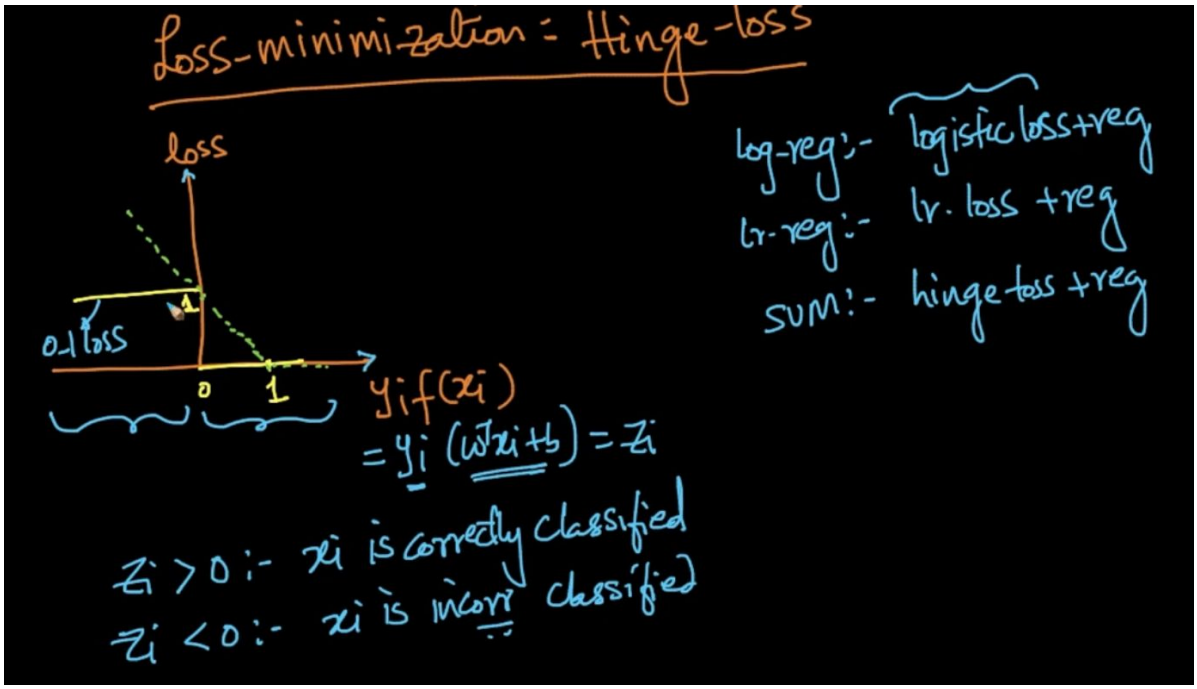
k can be anything there not just 1 but in the end it doesn't matter. Why can't we take k1, k2 separately because our whole idea of SVM was yo separate the +ve and -ve points well. It would be possible only if the hyperplanes $\pi^+ and\ \pi^-$ are equidistant from the optimal hyperplane π so it should be just k. Max 2/||w| = max (8/||w||) so it's the same.



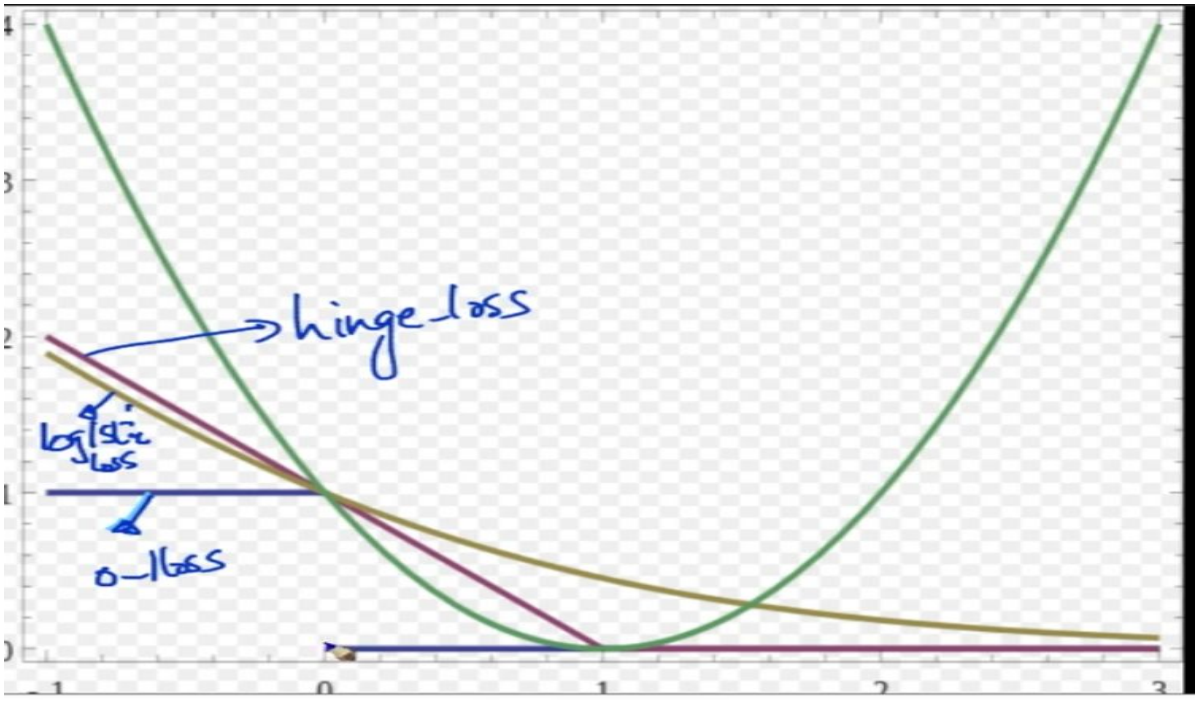
2nd interpretation is if we take $w^T x + b = k$ as $(\frac{w}{k})^T x + (\frac{b}{k}) = 1$ then also nothing would change since ||w|| need not be 1 it can be anything. So, $(\frac{w}{k})^T x + (\frac{b}{k})$ => $(w')^T x + b' = 1$

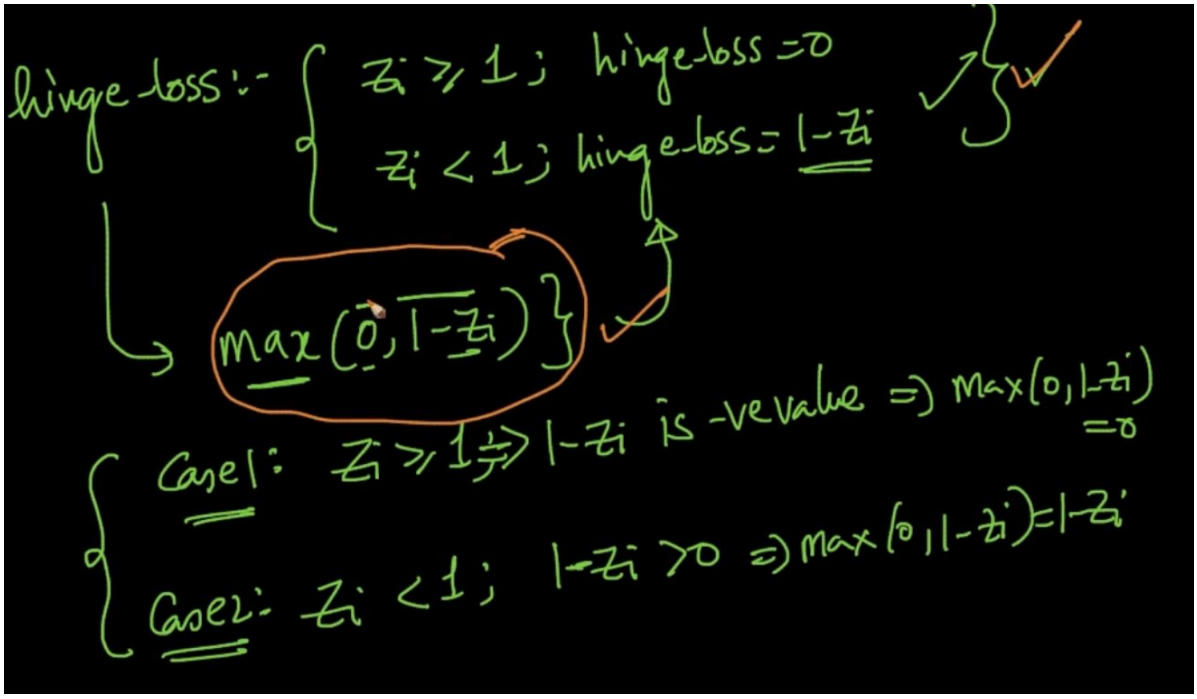
So the reason we are taking +1 and -1 is for simplifying the maths.

LOSS FUNCTION(HINGE LOSS) INTERPRETATION



$z_i = y_i(w^T x_i + b)$. With 0-1 loss, when $z_i > 0$: x_i is correctly classified since y is +ve and w and x are at same direction whereas when $z_i < 0$ x_i is incorrectly classified since w and x will be at opposite direction.



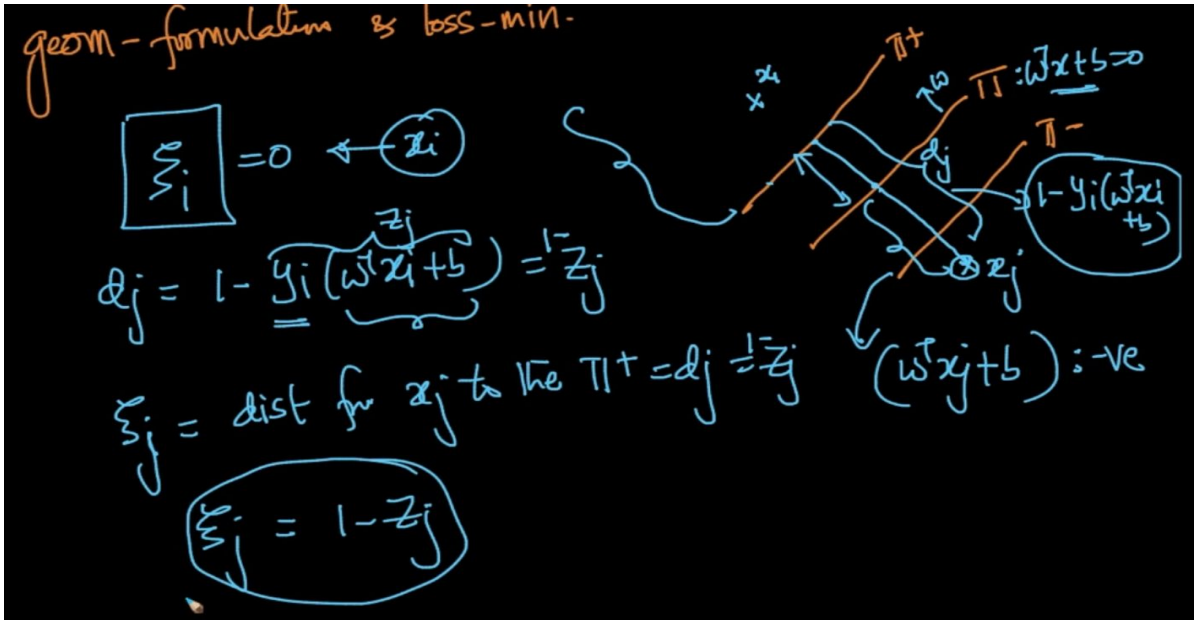


In Hinge Loss if $z_i \geq 1$ then it's 0 or if $z_i < 1$ it gets then loss increases

Hinge Loss : max(0, 1 - z_i)

Case 1: $z_i \geq 1$ => 1 - z_i is -ve value => max(0, 1 - z_i) = 0

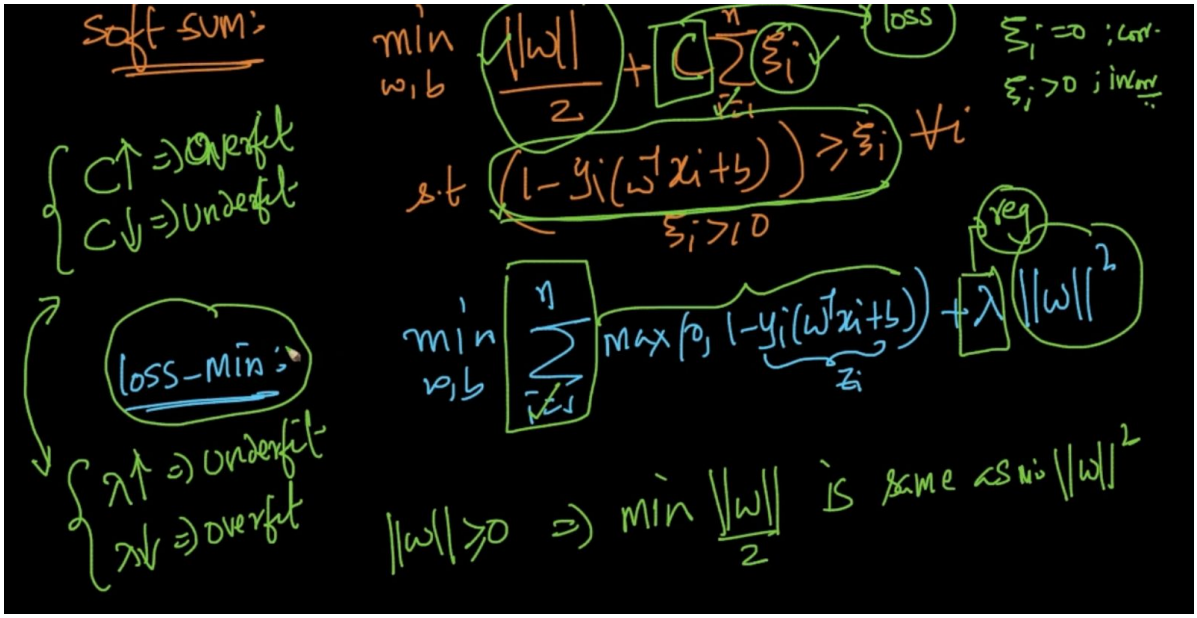
Case 2: $z_i < 1$ => 1 - $z_i > 0$ => max(0, 1 - z_i) = 1 - z_i



For x_i , ζ = 0 as it's correctly classified

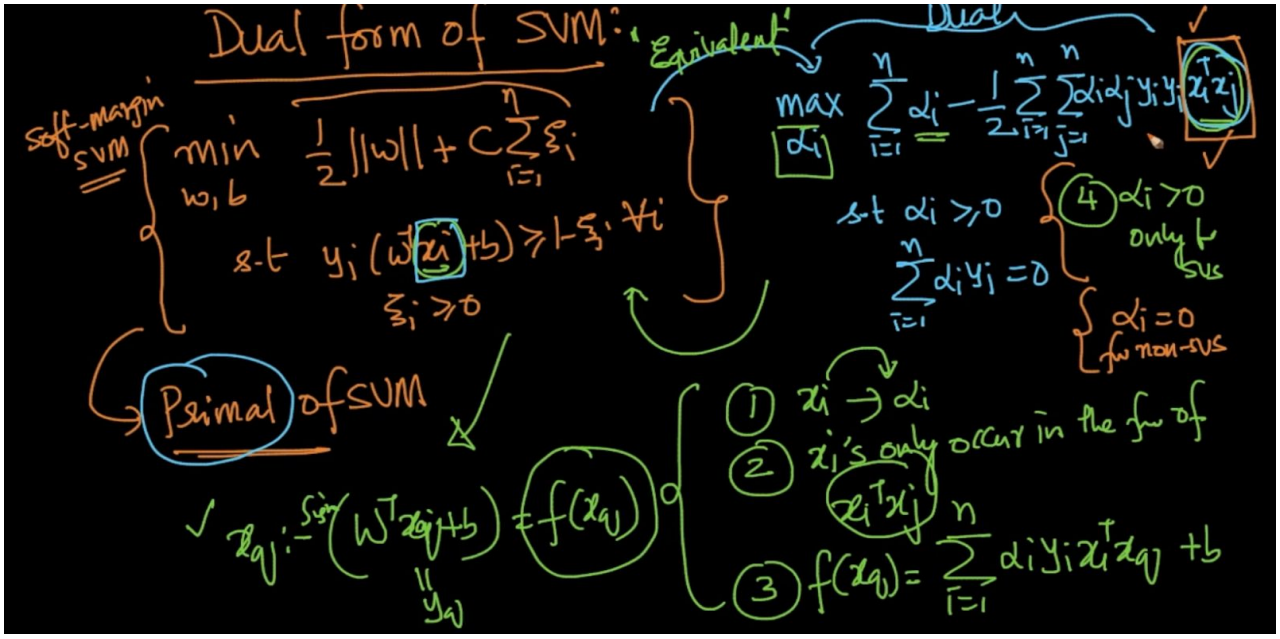
For x_j , dist from π : $w^T x + b$ => $y_j . w^T x_j + b$ but it's -ve since w and x are opposites

And from π^+, distance of x_j (d_j) = 1 - $y_j . (w^T x_j + b)$ since y = +ve and $w^T x$ = -ve and 1 is distance from hyperplane π for π^+ and $y_j . w^T x_j + b$ for x_j . So, d_j = 1 - z_j = ζ_j



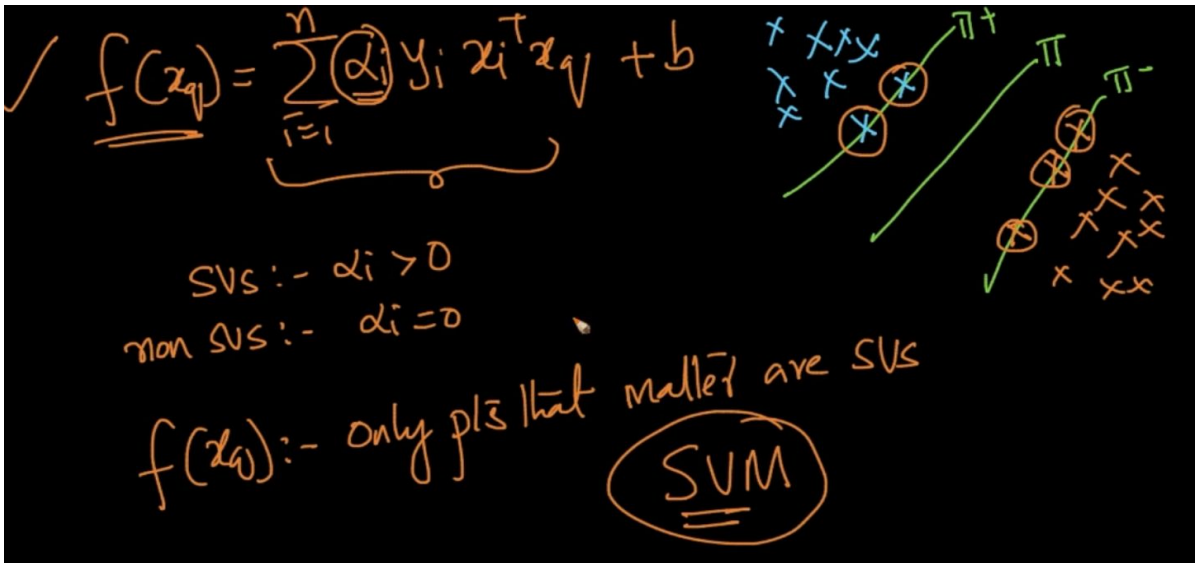
We are saying that Soft-SVM is equivalent to loss minimization formula it's just that 'C' is multiplied to loss and λ is multiplied to regularizer

DUAL FORM OF SVM



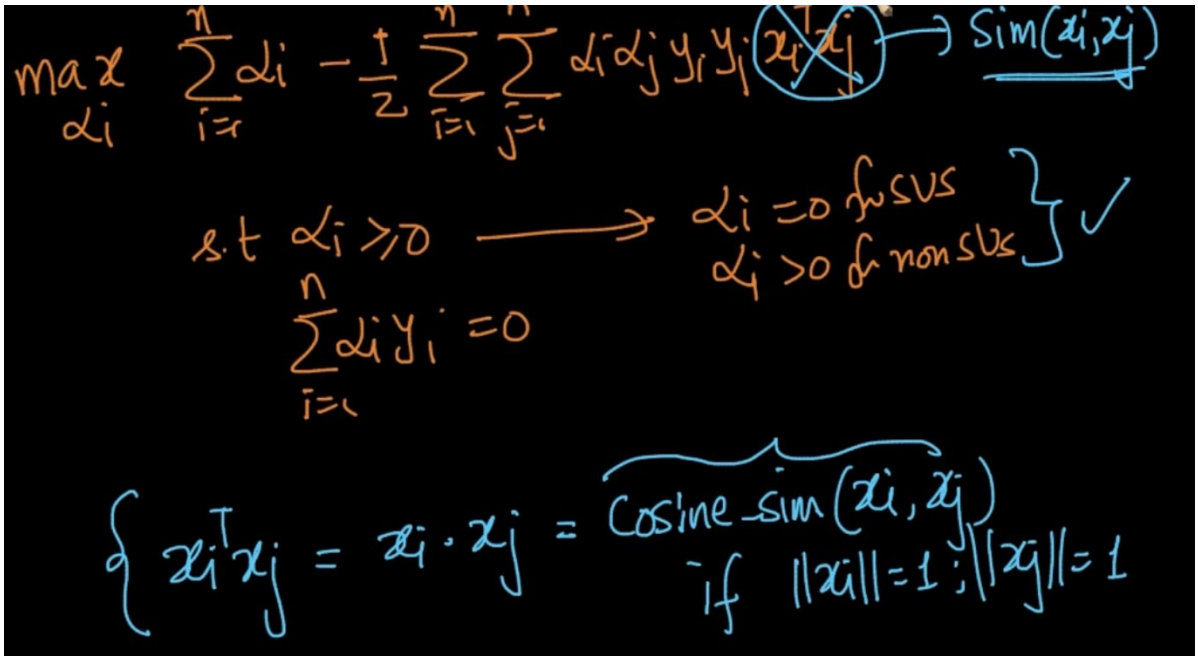
Our soft margin SVM can also be written as like mentioned in dual form . The rules are also written there.

PS - In dual form we are not dealing with x_i separately. It's $x_i^T x_j$



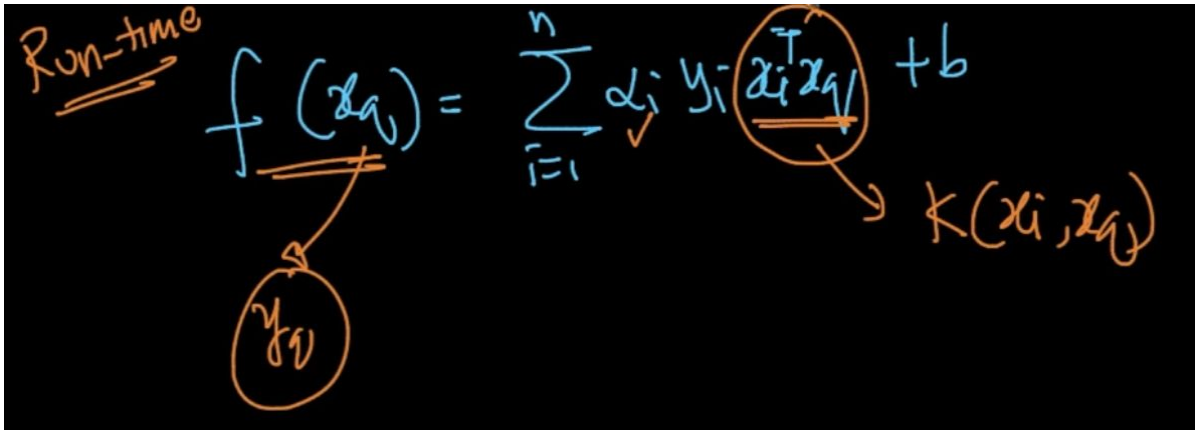
As the 4th pt say α_i > 0 only for support vectors for other points the whole product will be 0

So, the only points that matter are Support Vectors



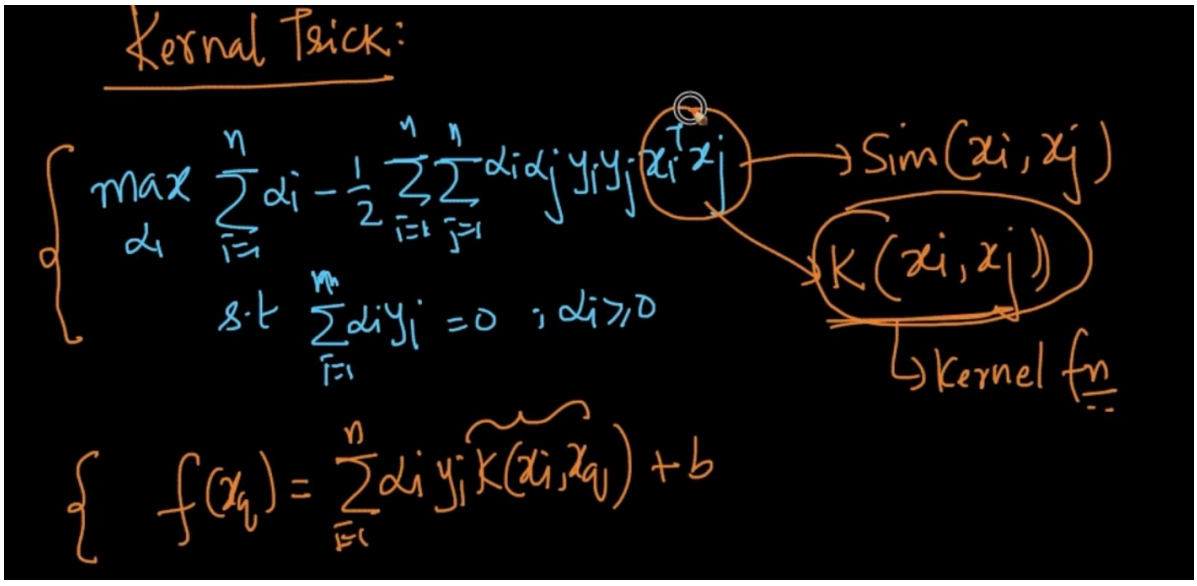
So you can replace $x_i^T x_j$ with Cosine -similarity (x_i, x_j). We can't do that with Soft -SVM

This part made SVM super-popular. We can represent Cosine -similarity (x_i, x_j) => K(x_i, x_j) which is also known as Kernel Function. We don't deal with x_i, x_j explicitly but with their dot product.

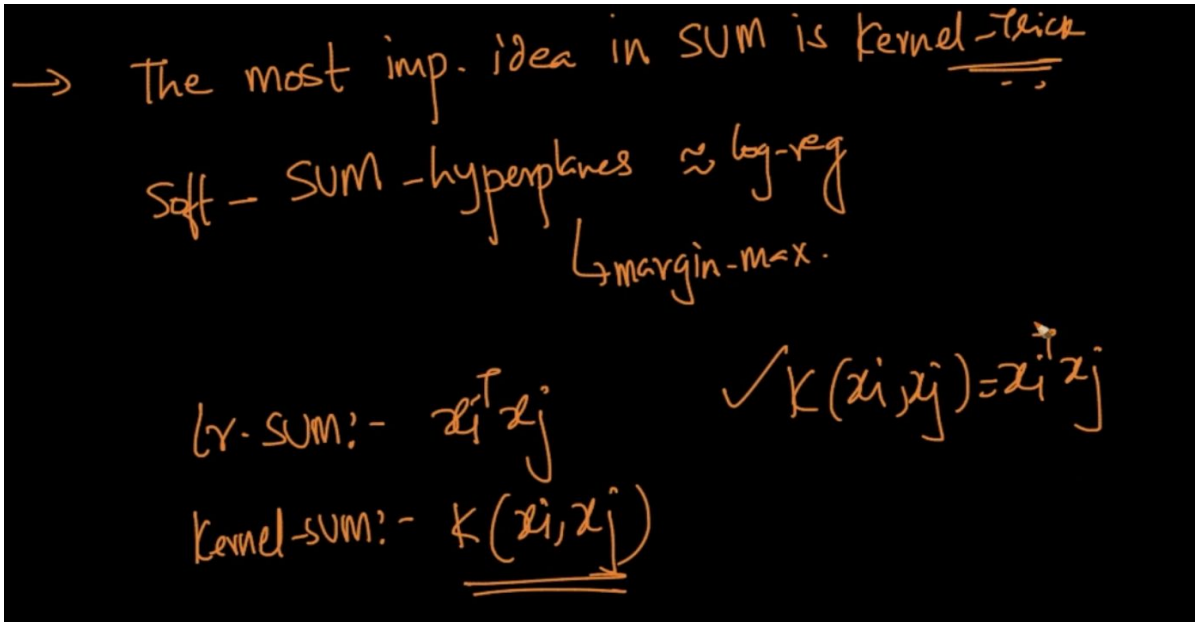


This is same in Run-Time for SVM

KERNEL TRICK FOR SVM

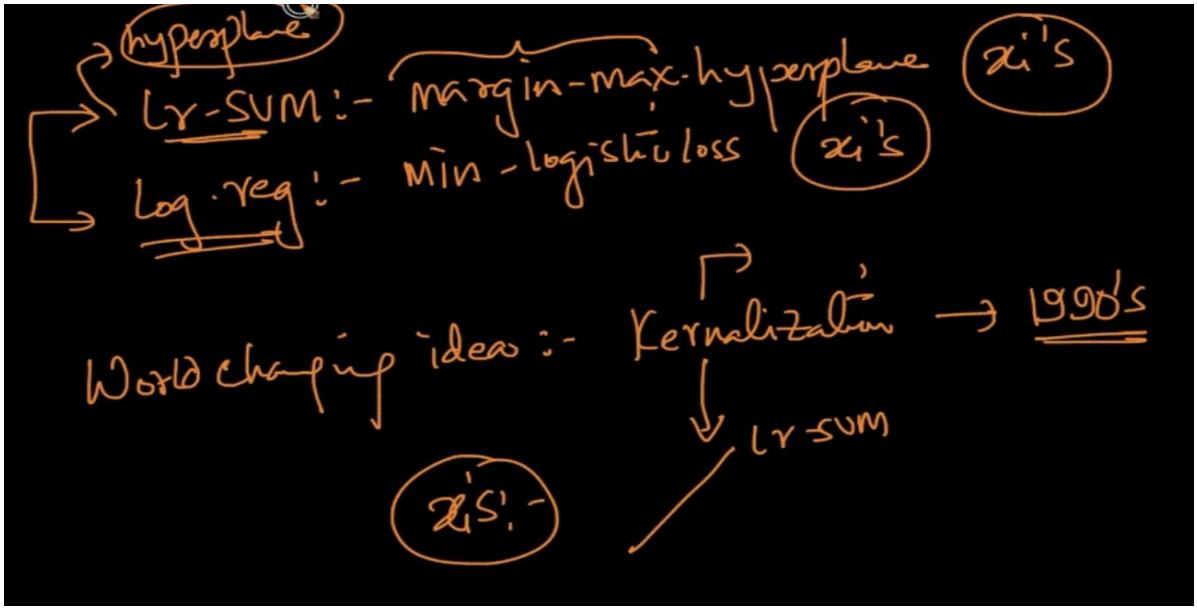


We'd seen that $x_i^T x_j$ can be replaced with K(x_i, x_j) . It is called Kernel Trick

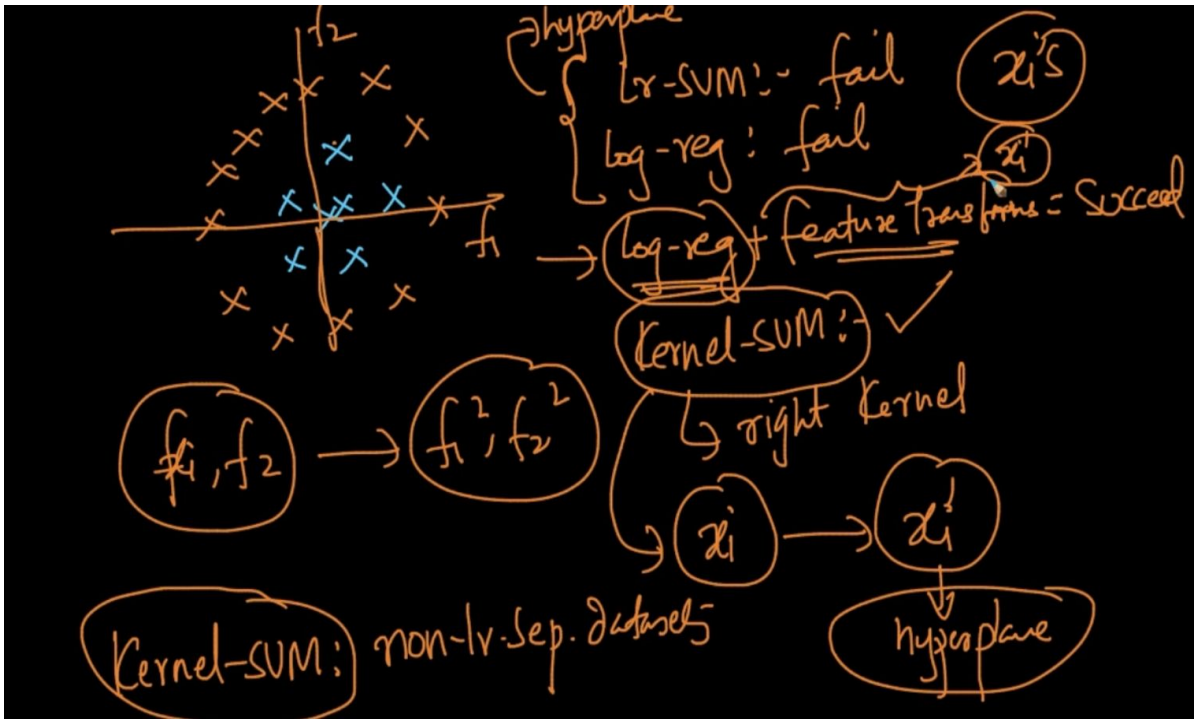


Kernel-Trick is one of the most important idea in SVM.

If K(x_i, x_j) = $x_i^T x_j$ then it is simply linear SVM but if we are using a function K(x_i, x_j) then Kernel-SVM

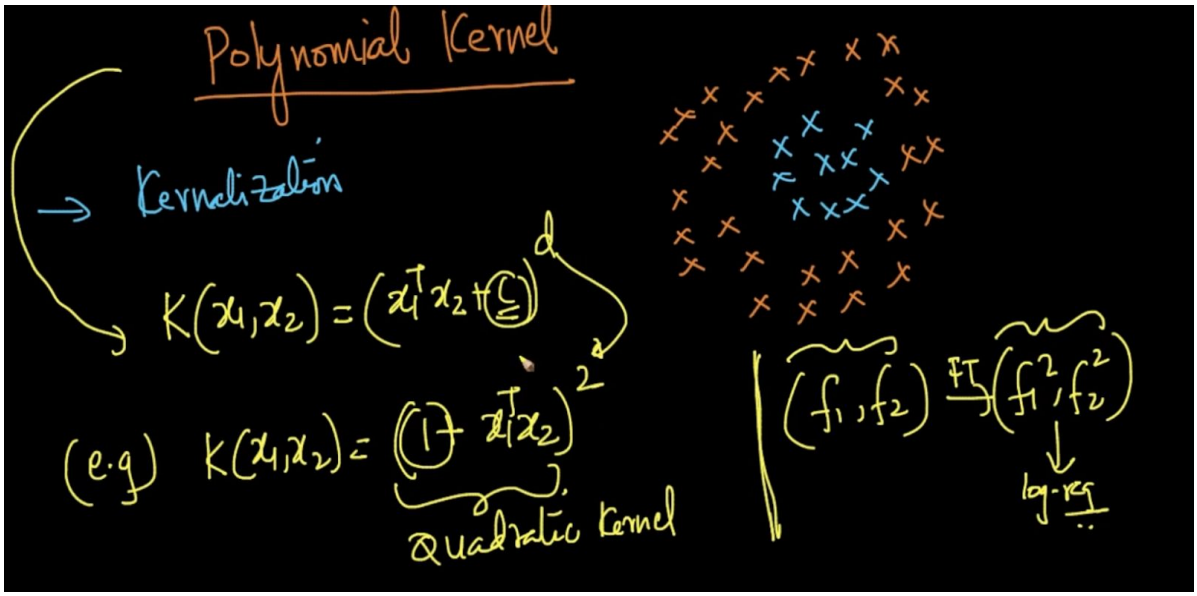


In Linear SVM we try to find the hyperplane which maximizes margin but it wasn't a world changing idea but Kernalization is



As we can see the above data isn't seperable so our Linear SVM won't be able to find the margin. So it fails.. If we apply Kernel- SVM with the right Kernel then we can find the right hyperplane. Hence we can infer Kernel SVM works on non-linearly separated data

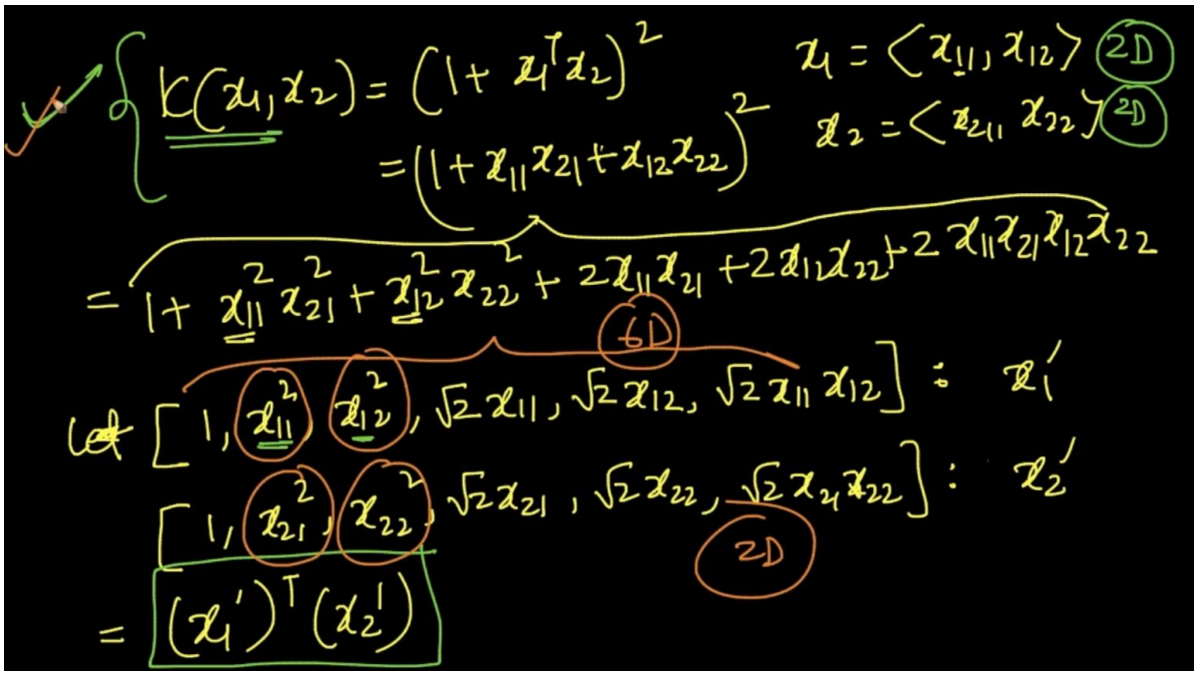
POLYNOMIAL KERNEL



We'd seen that with features f_1, f_2 our logistic regression couldn't find a hyperplane to separate it so we apply feature transformation to get to a feature space where a hyperplane can divide the hyperplane

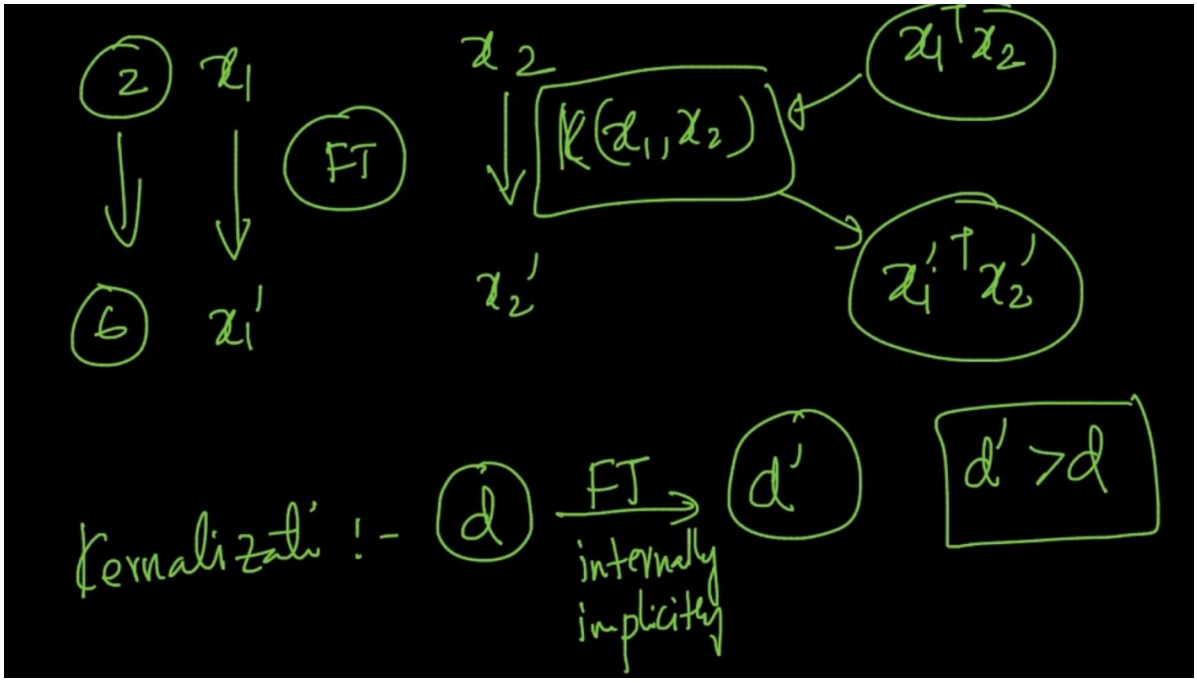
In SVM we can do that with kernelization. K(x_1, x_2) = $(x_1^T x_2 + c)^d$ where d - degree of polynomial and c- constant

K(x_1, x_2) = $(x_1^T x_2 + 1)^2$ is called a quadratic kernel . It's a polynomial kernel with d = 2, c = 1



We are just expanding the formula . Our x_1, x_2 has 2 dimensions as seen. So after expanding it we'll be getting $x'_1,\ x'_2$ so we can $(x'_1)^T\ (x'_2)$

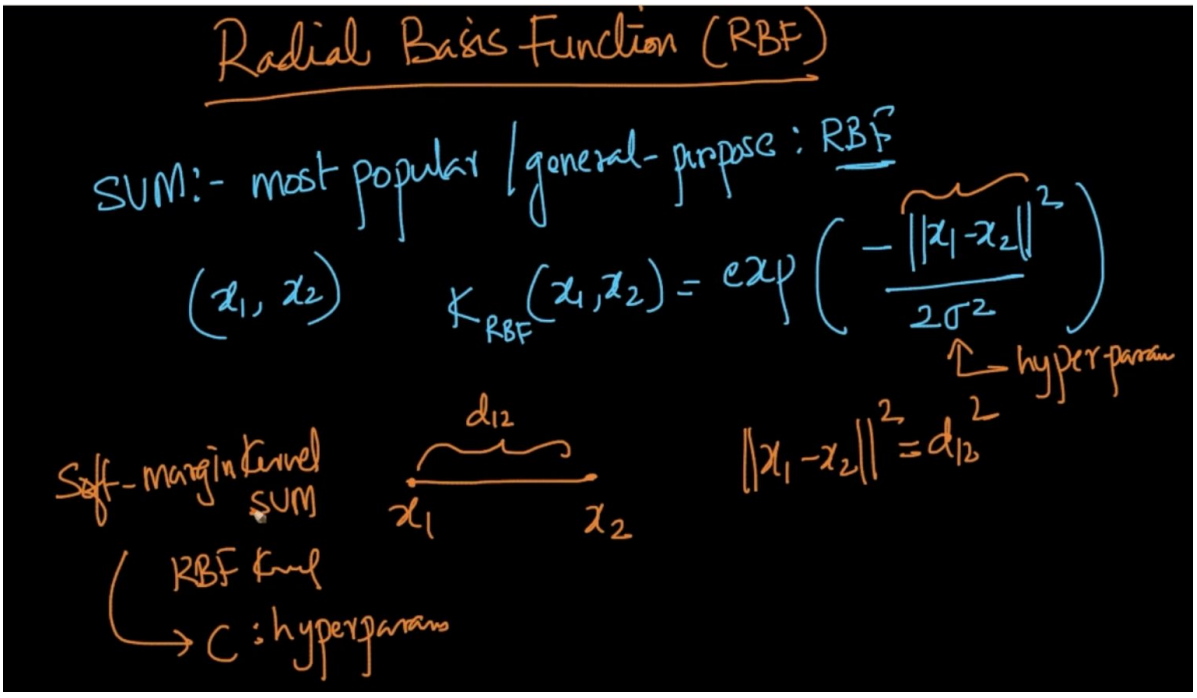
Note: Our 2D data is converted to 6D



With Kernelization , $x_1^T x_2$ => K(x_1, x_2) => $(x'_1)^T\ (x'_2)$. With it our d- dimensional space we get to a new d'-dimensional space where d' > d

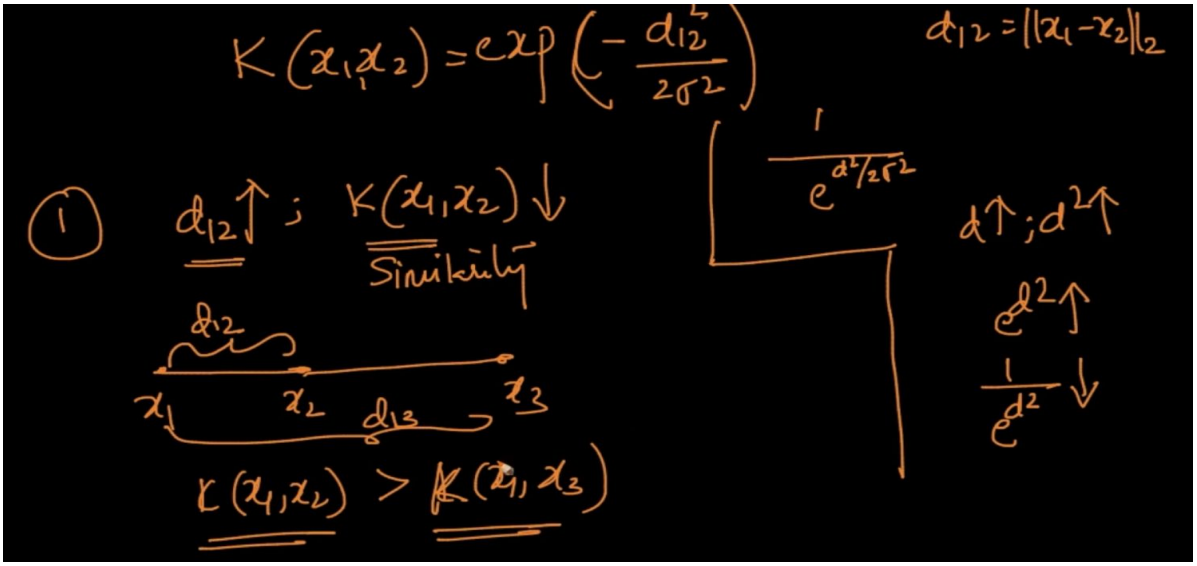
PS - So in SVM our most important work is to find the right kernel in dataset. Ex: For the above concentric circle dataset a polynomial kernel with degree 2 is the right kernel

RADIAL BASIS FUNCTION KERNEL (RBF)

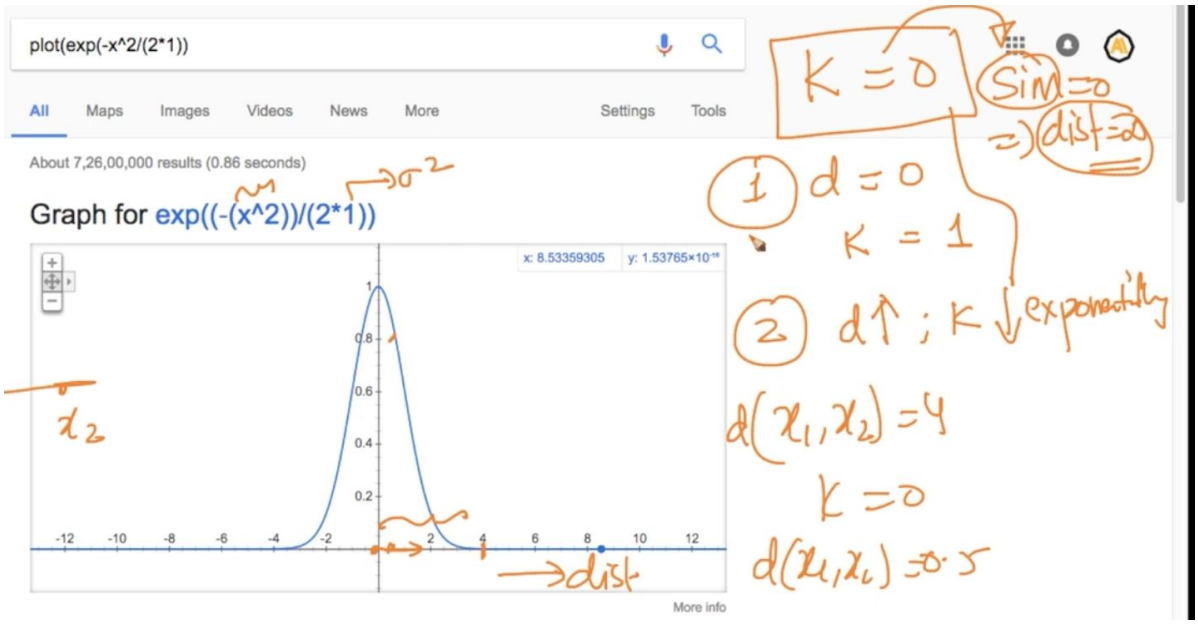


RBF Kernel is the most popular kernel in SVM. $K_{RBF}(x_1,x_2)$ = exp ($\frac{-||x_1-x_2||^2}{2\sigma^2}$) where

$||x_1-x_2||$ = Distance between 2 points = d_{12} and σ => hyperparameter



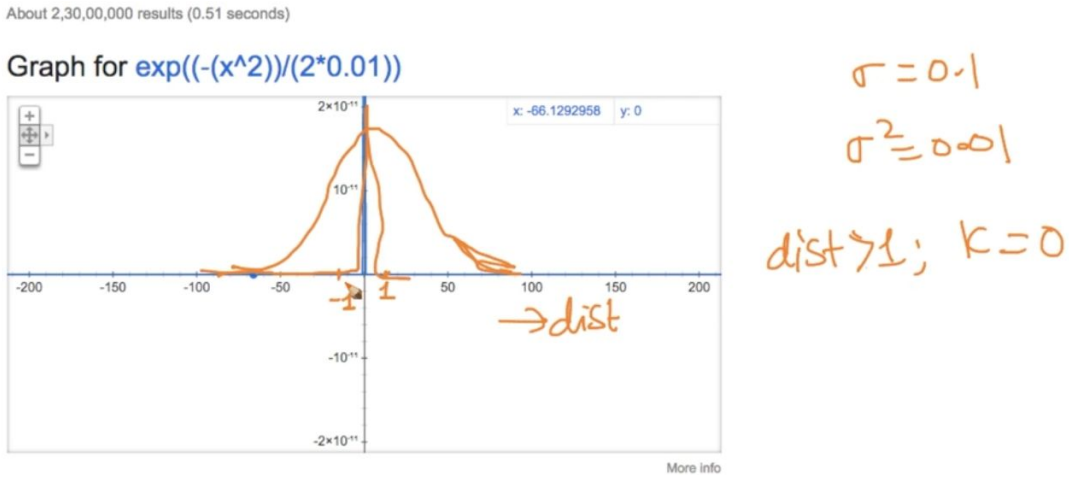
As $d_{12}\uparrow$; $K_{RBF}(x_1,x_2)\downarrow$ i.e similarity $\downarrow$. In fig , K(x_1,x_2) > K(x_1,x_3) as $d_{12} < d_{13}$



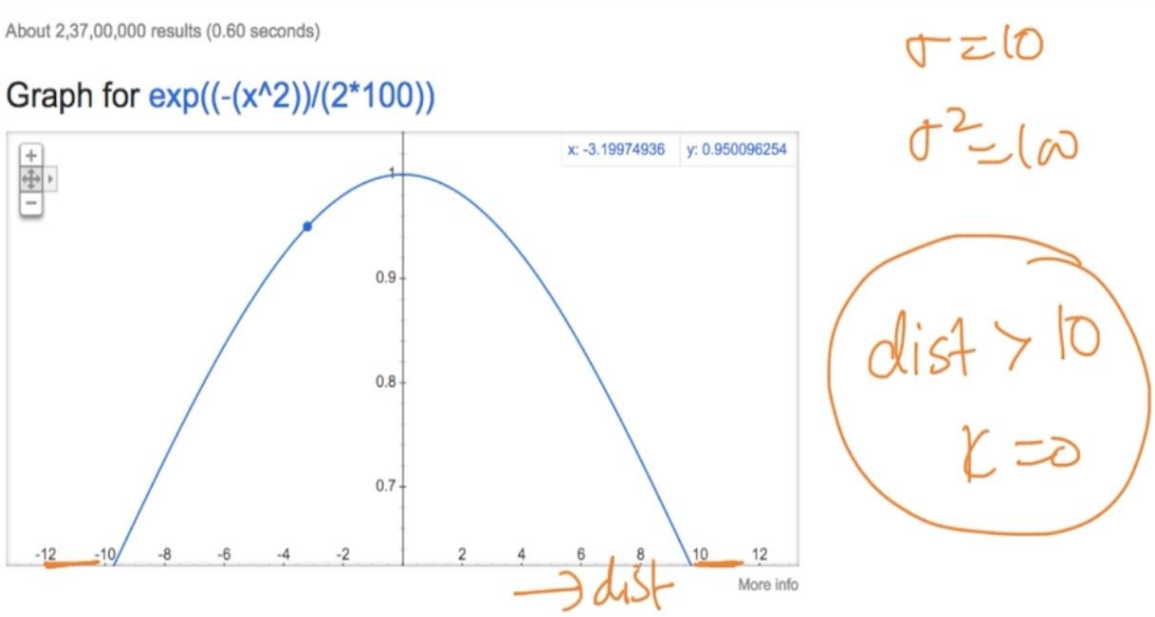
Here, $\sigma = 1$. So when d = 0 , K = 1 but when d ↑; K ↓ exponentially. As we can see as the distance increases the Kernel function's value gets decreased.

NOTE - We can intuitively say that Kernel is like Similarity function i.e when two points are similar their value is greater and distance is like dissimilarity i.e if distance is greater the points are dissimilar

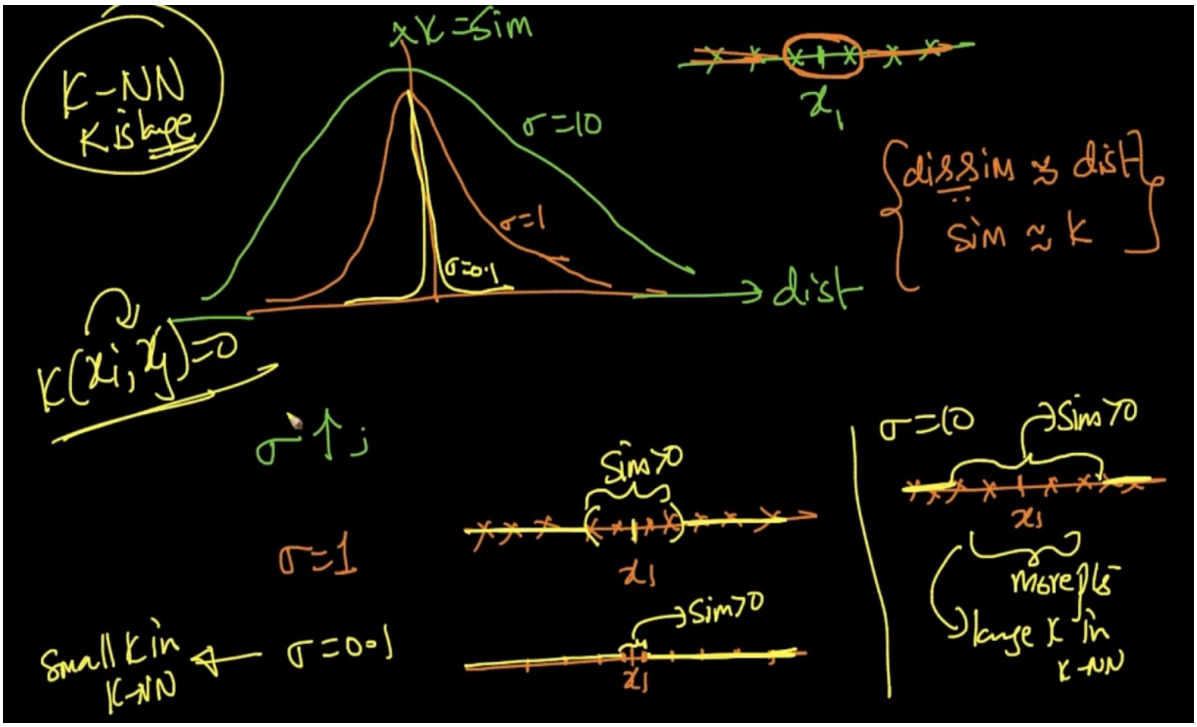
NOTE 2 - RBF ~ Gaussian PDF



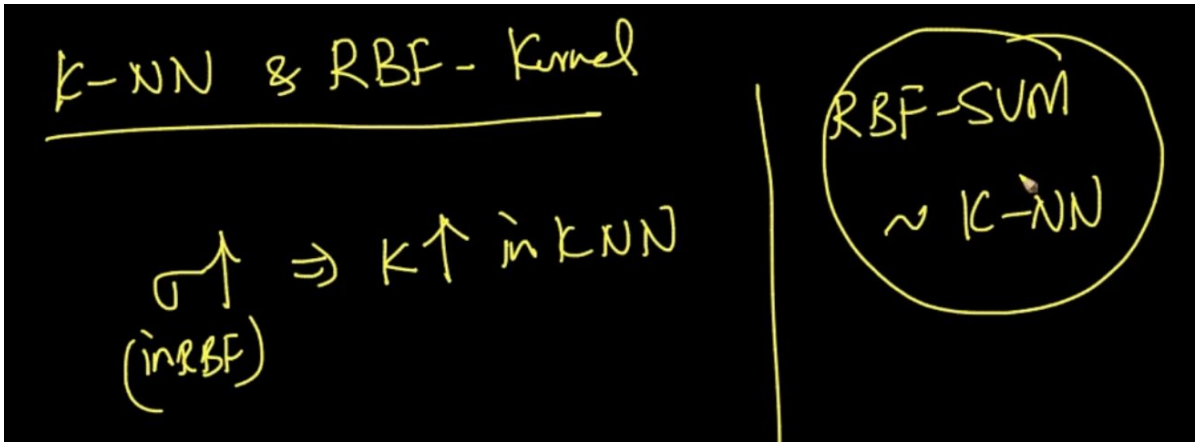
When we change σ = 1 to 0.1 the graph gets more steep. Even when dist > 1 ; K = 0



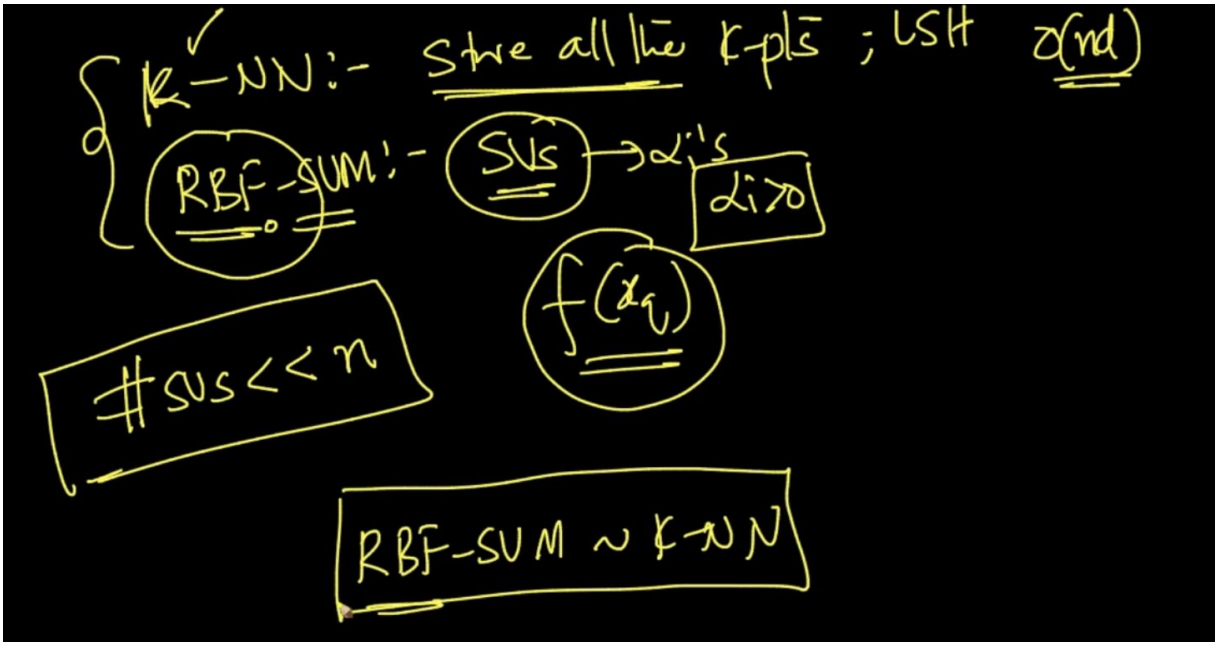
For σ = 10



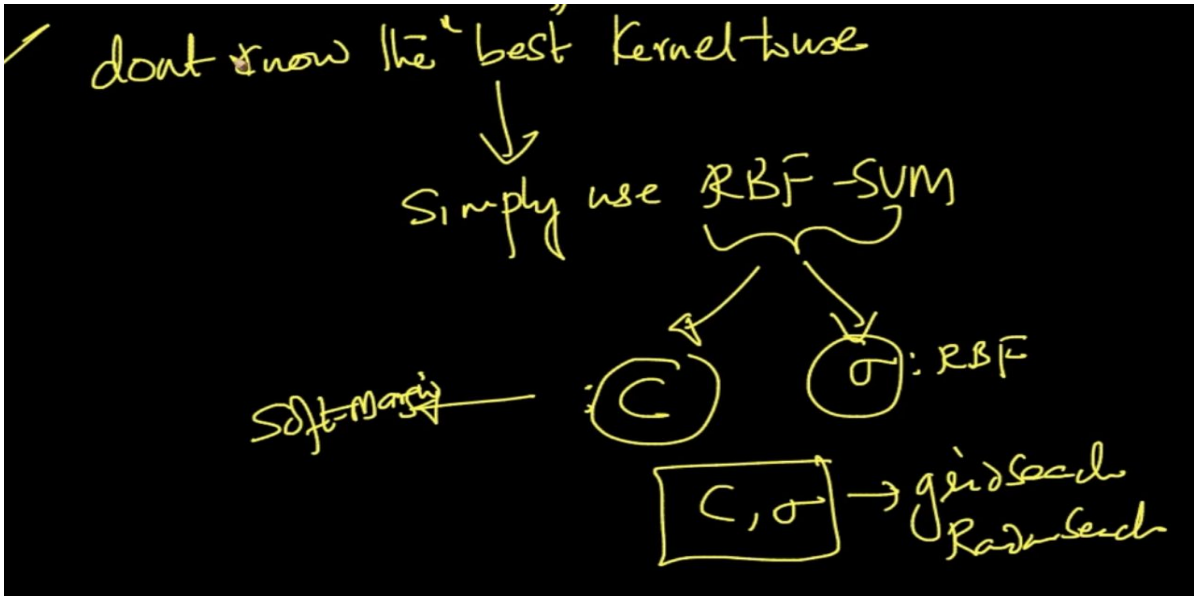
When σ is small we take only the points which are very close and we can take more points when σ is large . So σ can be compared with k in K-NN



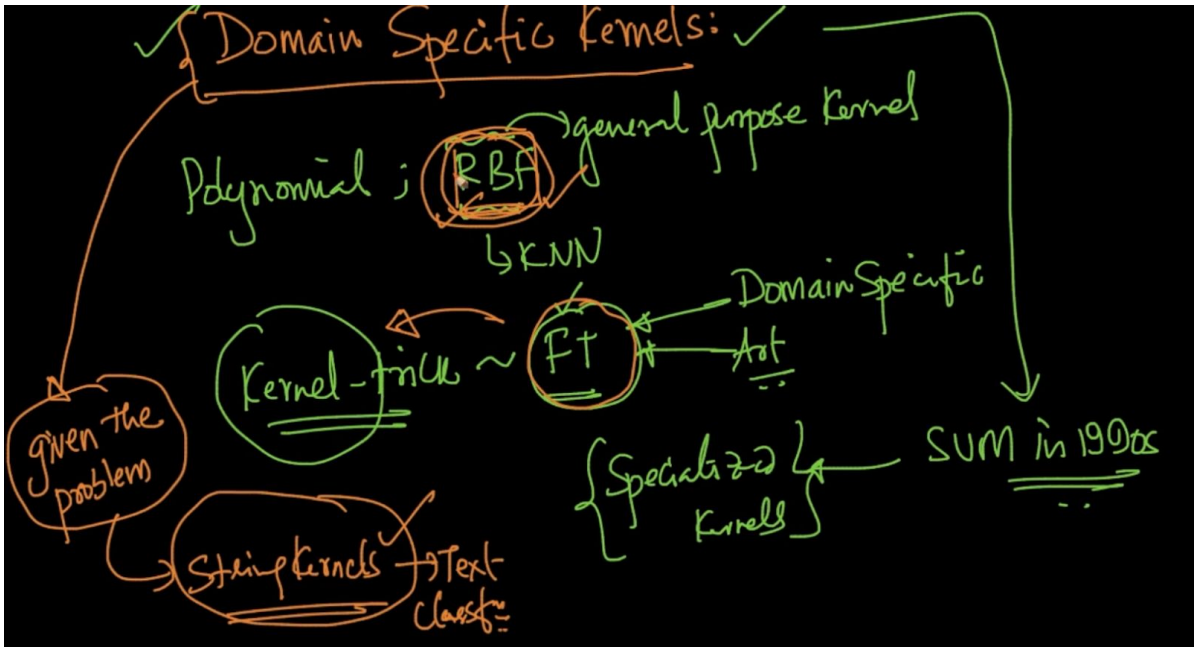
So we can say that RBF-SVM ~ k-NN



In K-NN we store all the k-Points but in RBF-SVM we store only the support vectors because for $\alpha > 0$ we only need Support Vectors.

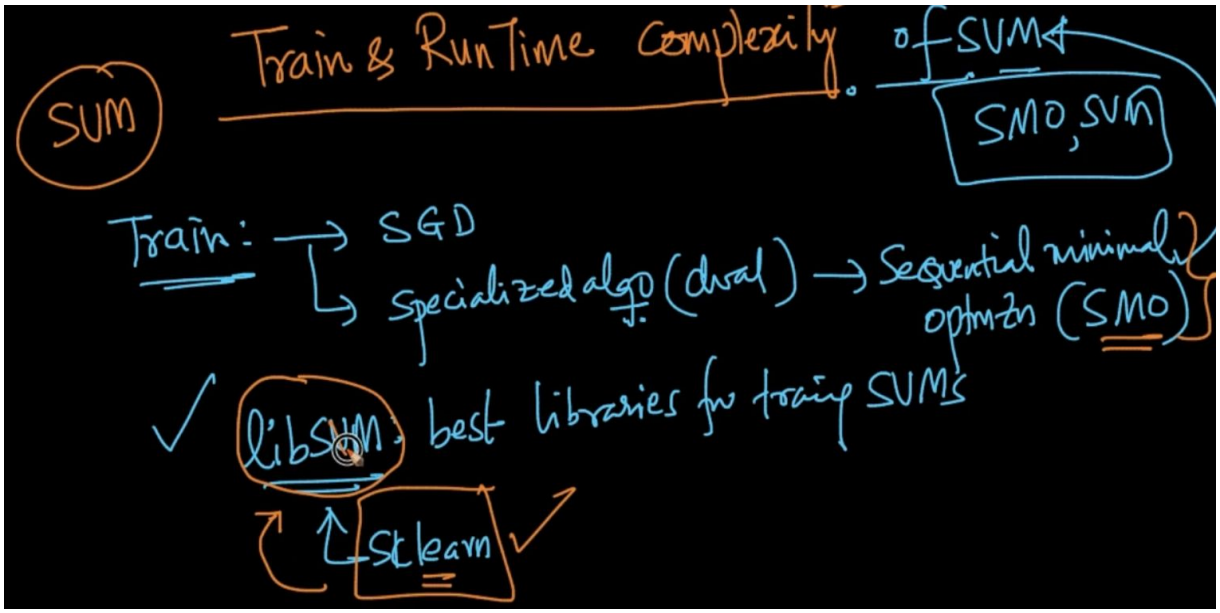


So it can work pretty well in large datasets .Hence , if we don't know which Kernel to use we use RBF-SVM . We do grid search/ random search for both C, σ

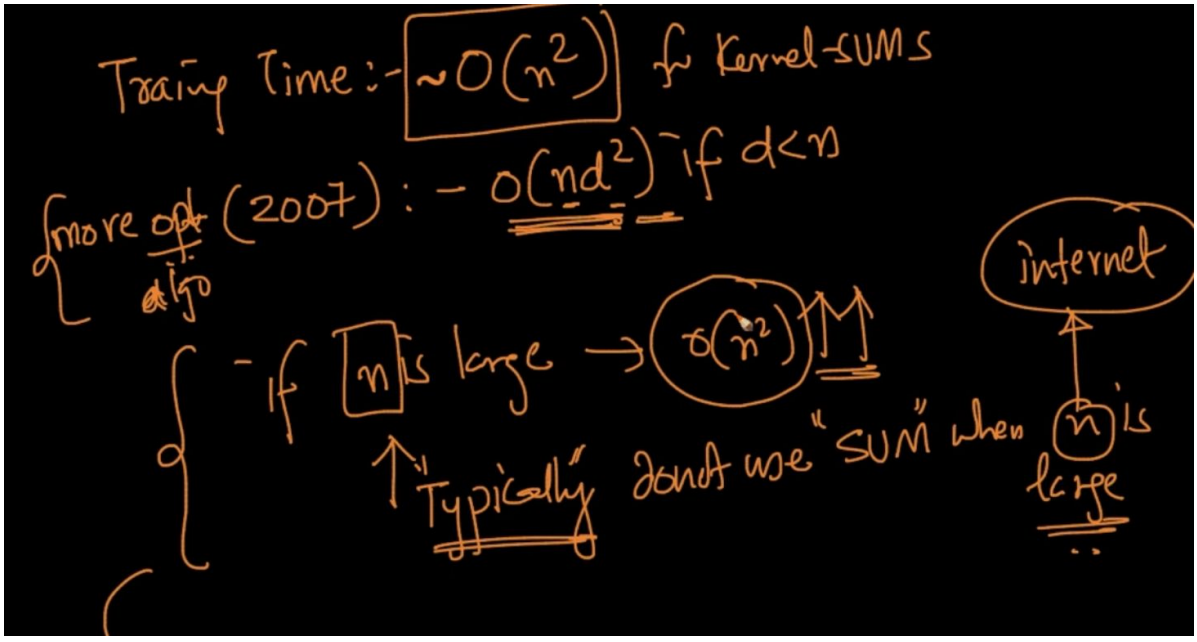


There are domain specific kernels for SVM i.e String Kernel for Text Classification, genome Kernel for Bioinformatics and so on...So, always look the appropriate Kernel i.e if we have Amazon Reviews data it is a text classification task so we can use String Kernel and our RBF-Kernel is obviously there for the backup plan

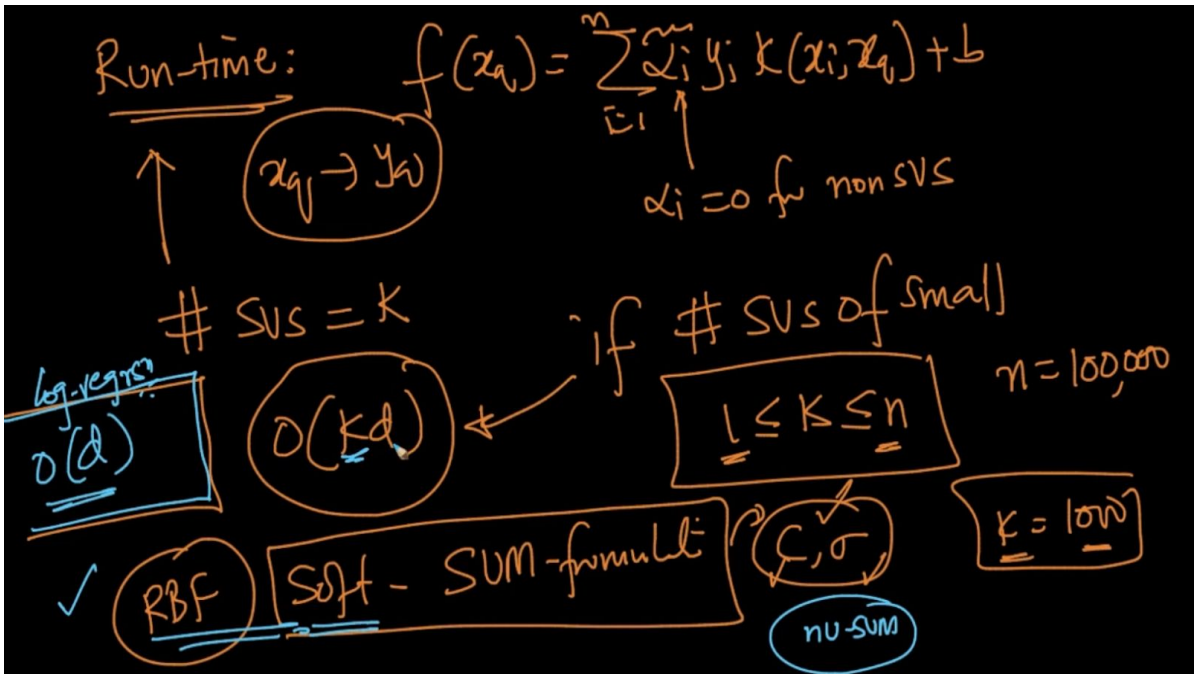
TRAIN AND RUN-TIME COMPLEXITIES



libSVM is one of the best library for training SVM



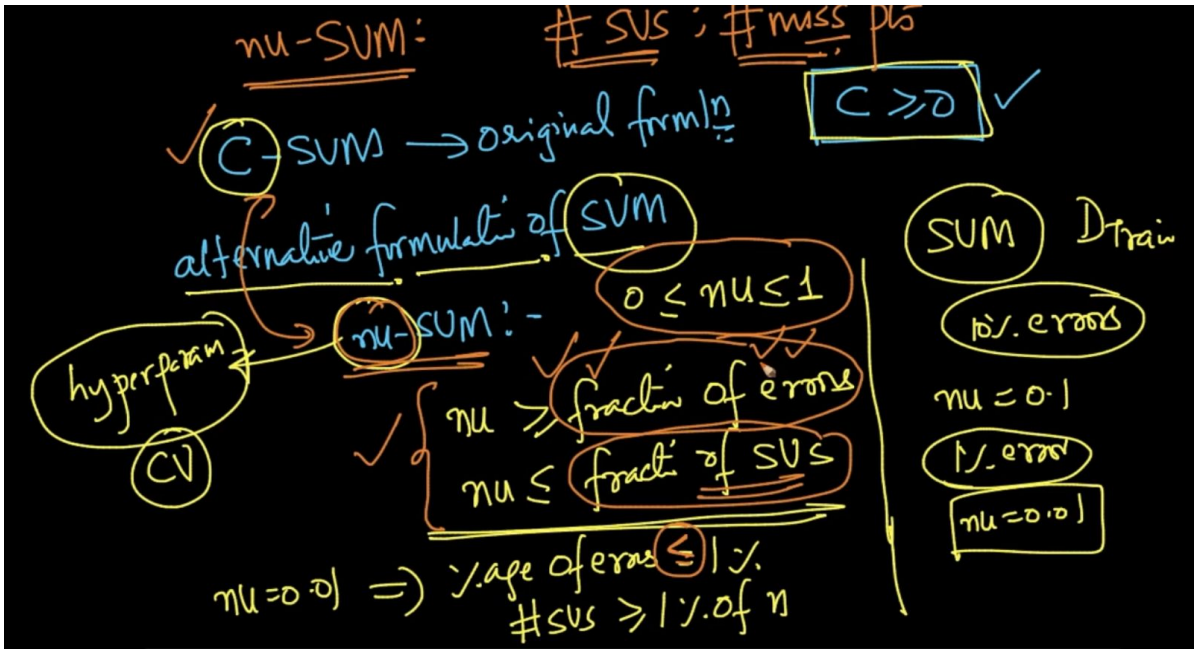
Training Time : O(n^2) for Kernel- SVMs so when n is large we typically don't use it since it takes significant time



Runtime Complexity : O(kd) where k - No. of Support Vectors and d- No. of dimensions

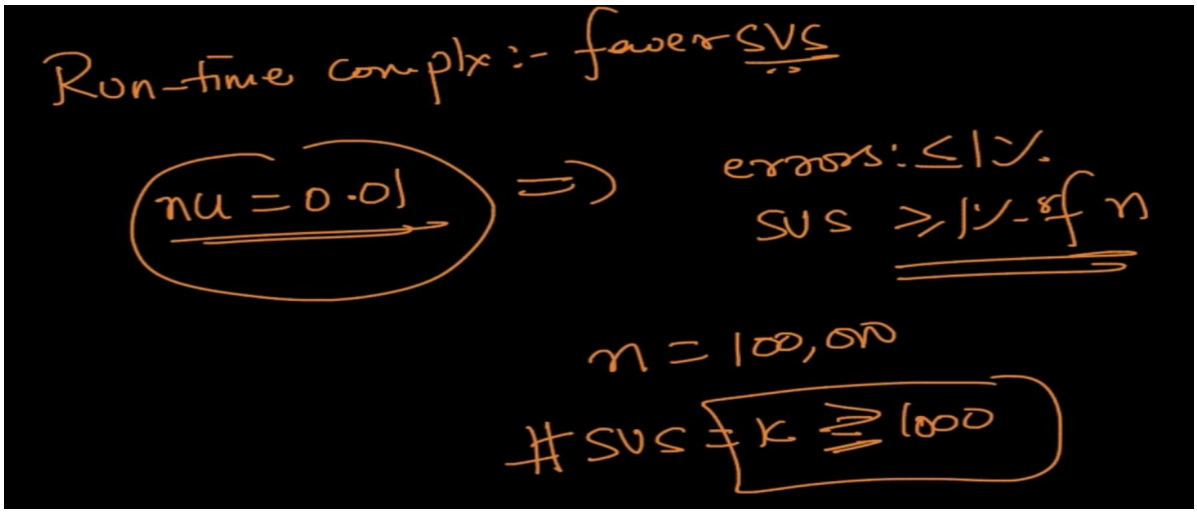
For logistic Regression it’s O(d) which is less than SVM

nu-SVM CONTROL ERRORS AND SUPPORT VECTORS



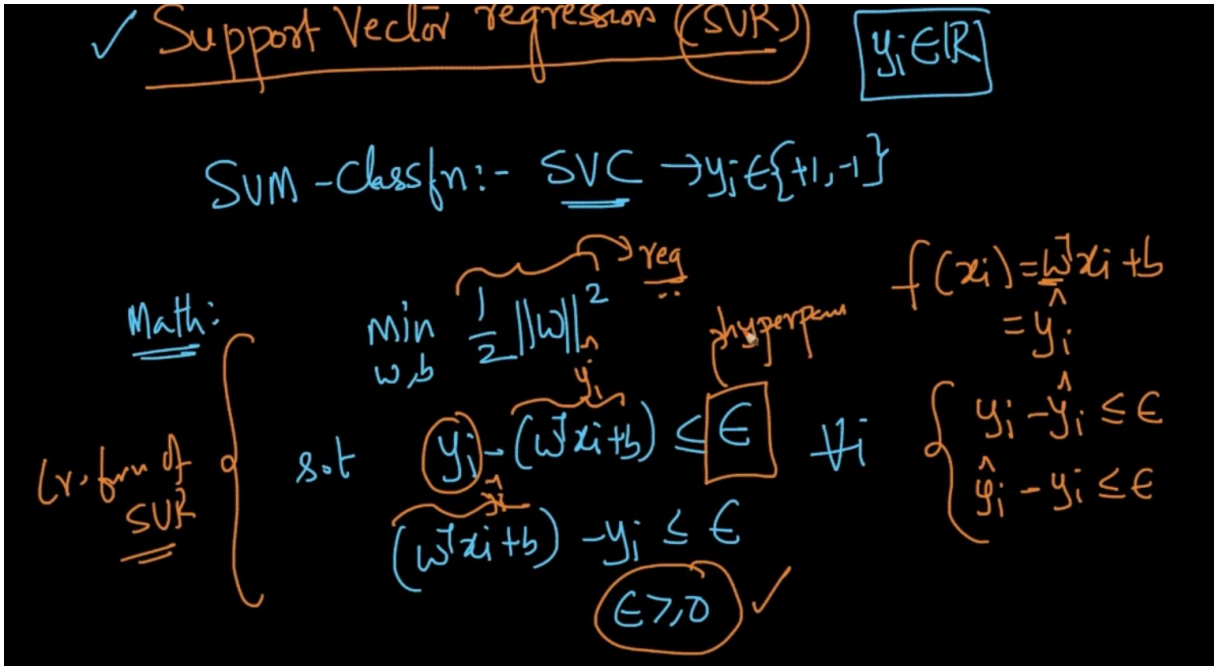
We studied C-SVM till now but C $\geq$ 0 so we can't get an accurate C so we use nu-SVM where 0 $\leq$ nu $\leq$ 1 and nu $\geq$ fraction of errors, nu $\leq$ fraction of Support Vectors

So , nu is the threshold of errors i.e suppose we want just 10 % errors just use nu = 0.1

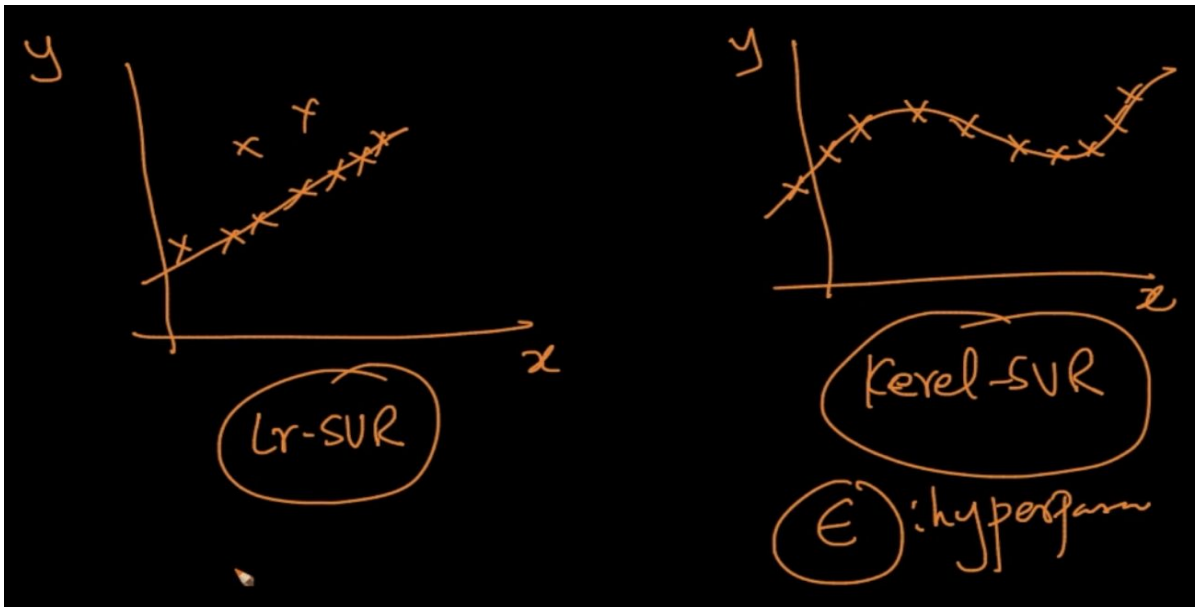


In run-time complexity we want fewer SVs so if nu = 0.01 it guarantees errors $\leq$ 1 % and also Support Vectors $\geq$ 1 % of n so if n = 100,000 nu = 0.01 , Support Vectors will guaranteed be $\geq$ 1000

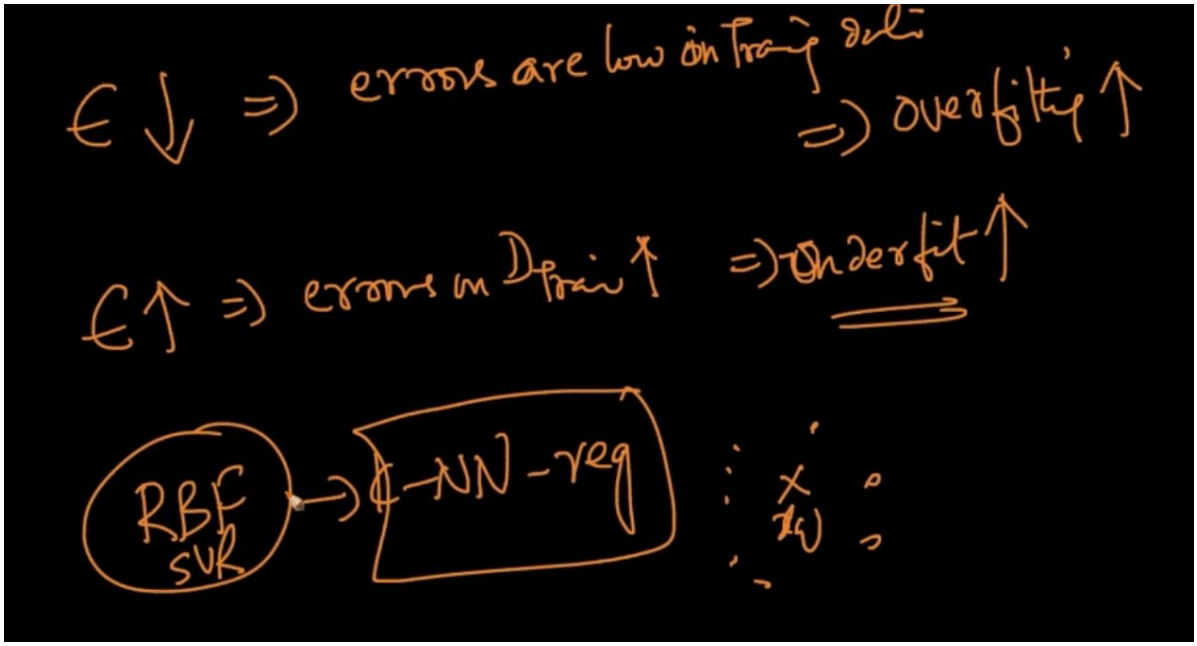
SVM REGRESSION



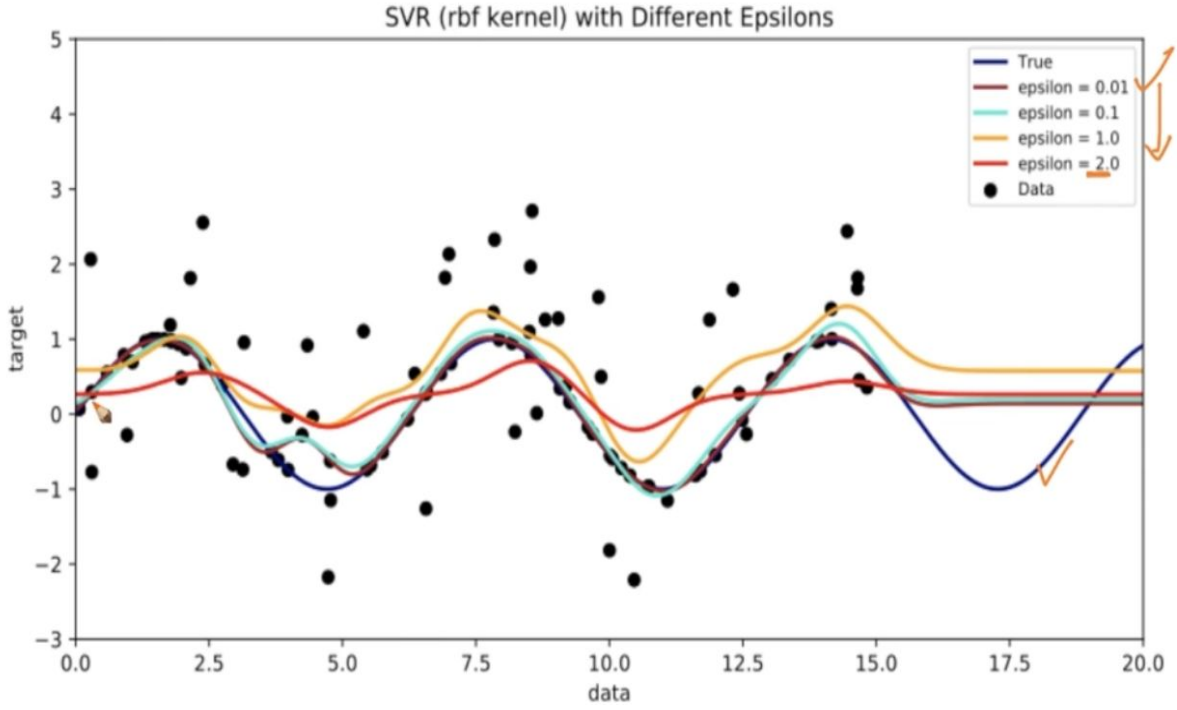
In SVM-Regression, min $\frac{1}{2}\|w\|^2$ => regularizer such that $y_i - (w^T x_i + b) \leq \in$ and $(w^T x_i + b) - y_i \leq \in$. $\in \geq 0$ and $\in$ is a hyper-parameter



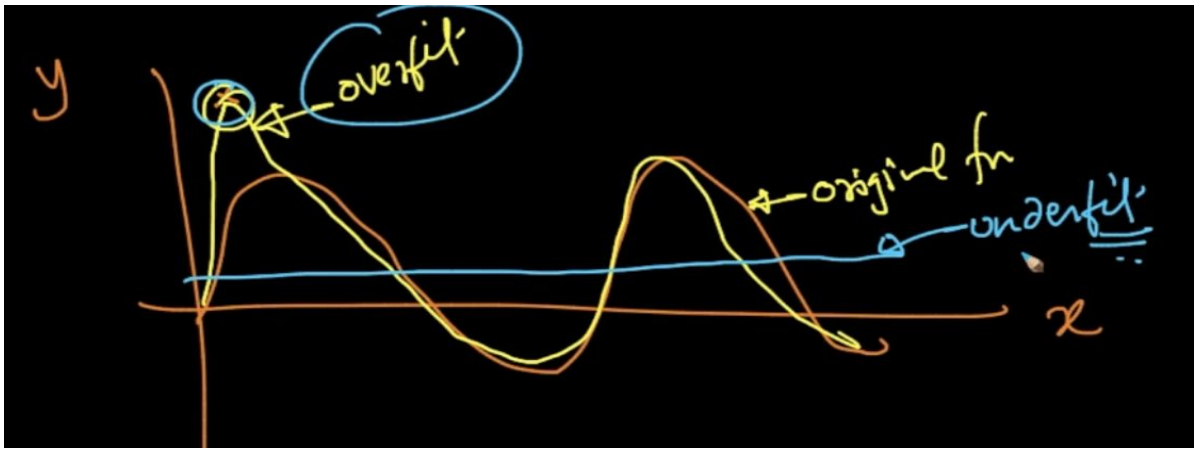
Linear SVR can only fit linear data and Kernel-SVR can fit any non linear data



Effect of $\in$ on errors

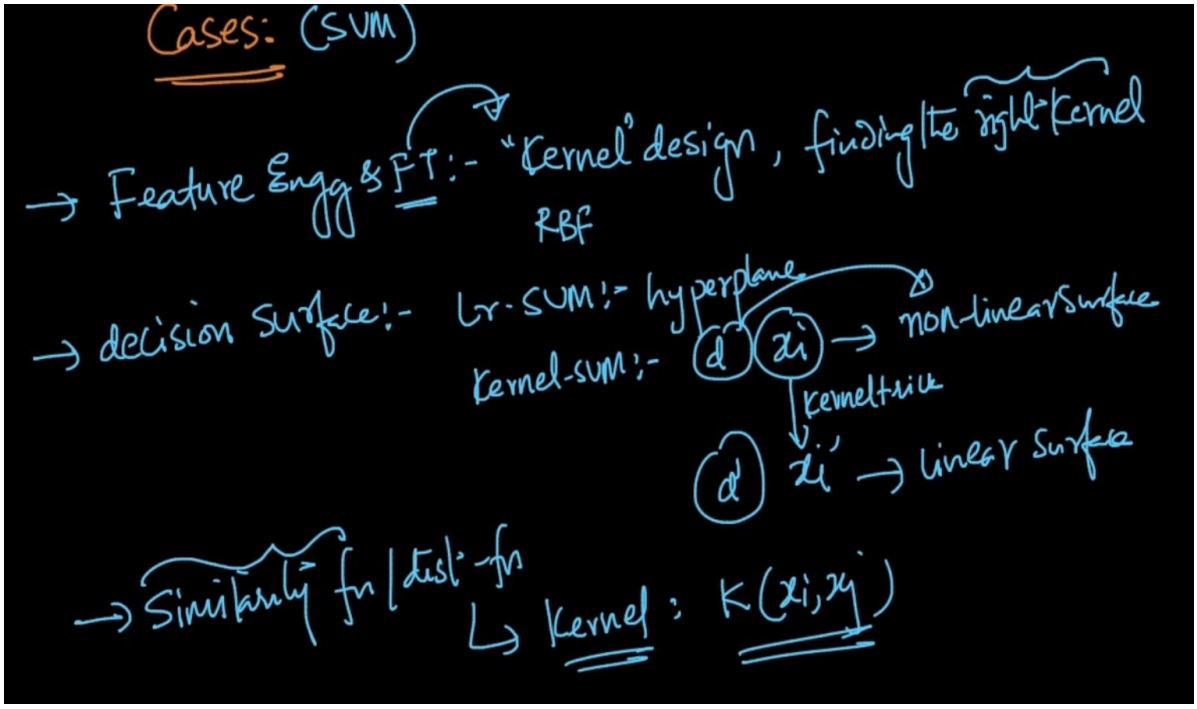


Brown line is overfitting , it has $\in$ = 0.01 and Red line is underfitting $\in$ = 2.0



When overfitting happens the function distorts for a single point and in underfitting It just doesn't care.

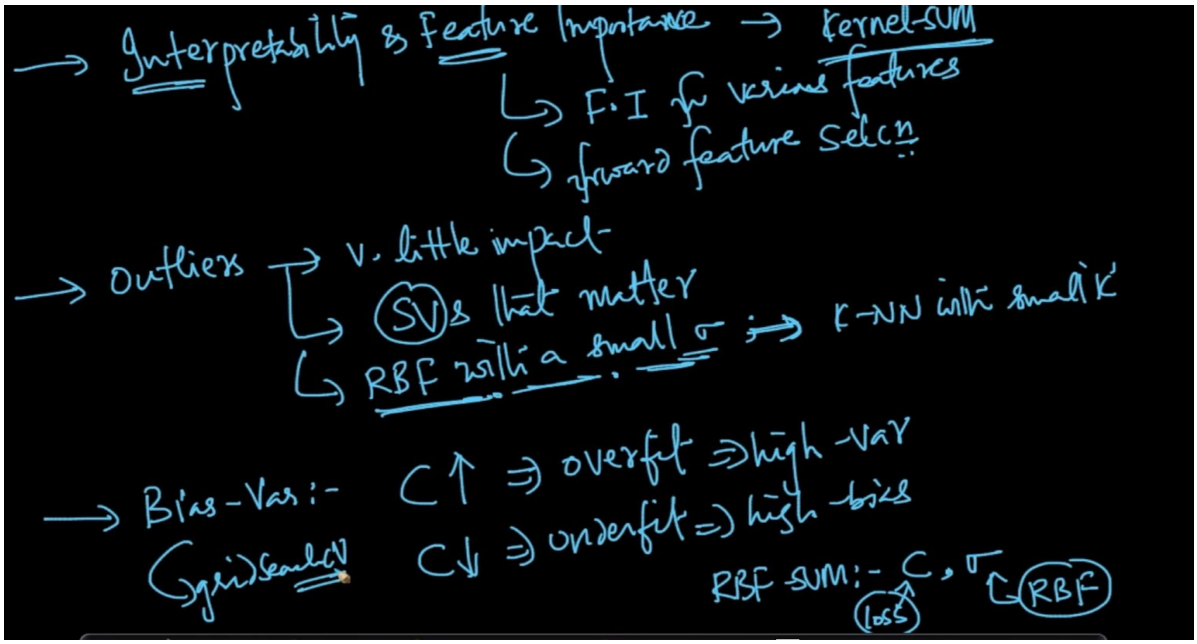
CASES



1) Feature Transformation - In SVM, feature transformation is changed to Kernel Design/finding the right kernel

2) Decision Surface - In kernel-SVM we don't get a linear hyperplane with x_i but with kernel trick it is transformed to x'_i to a different dimensional which is linearly seperable

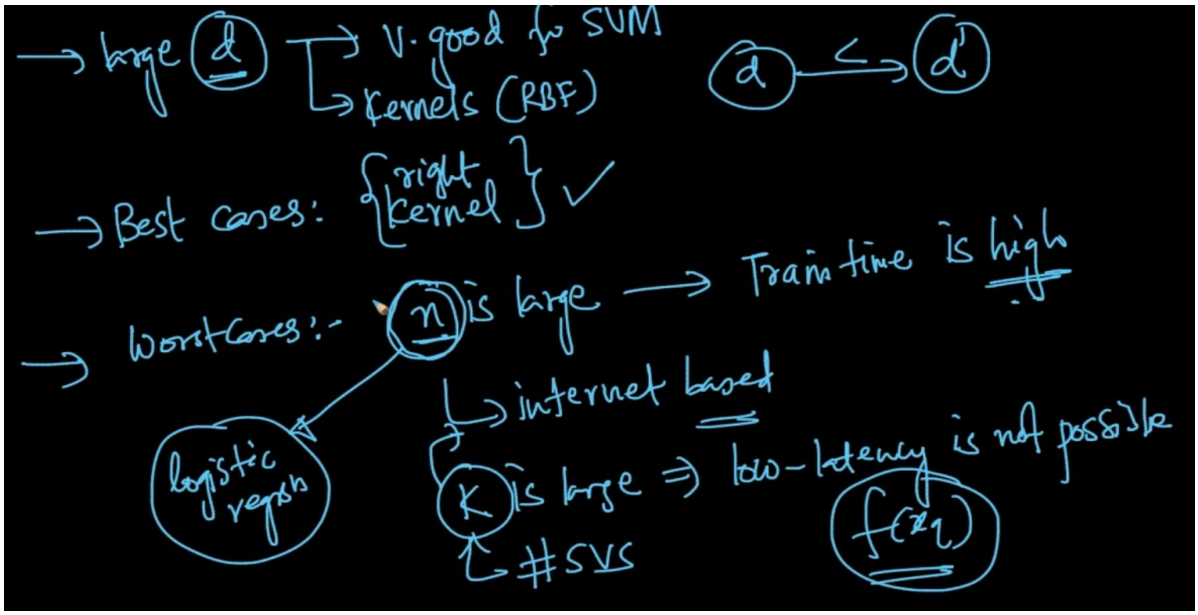
3) Similarity Function - If it's given then it's very easy to convert into Kernel



4) Interpretability and Feature importance is very difficult in Kernel-SVM. For Linear SVM we can always apply the same methods as logistic regression

5) Outliers has very little impact on SVM

6) Bias Variance - We can find the ideal parameters : C, σ using gridSearchCv



When we have a large dimensional data SVM works like a charm. Best case for SVM is Finding the right Kernel and worst case is when ‘n’ or K is large