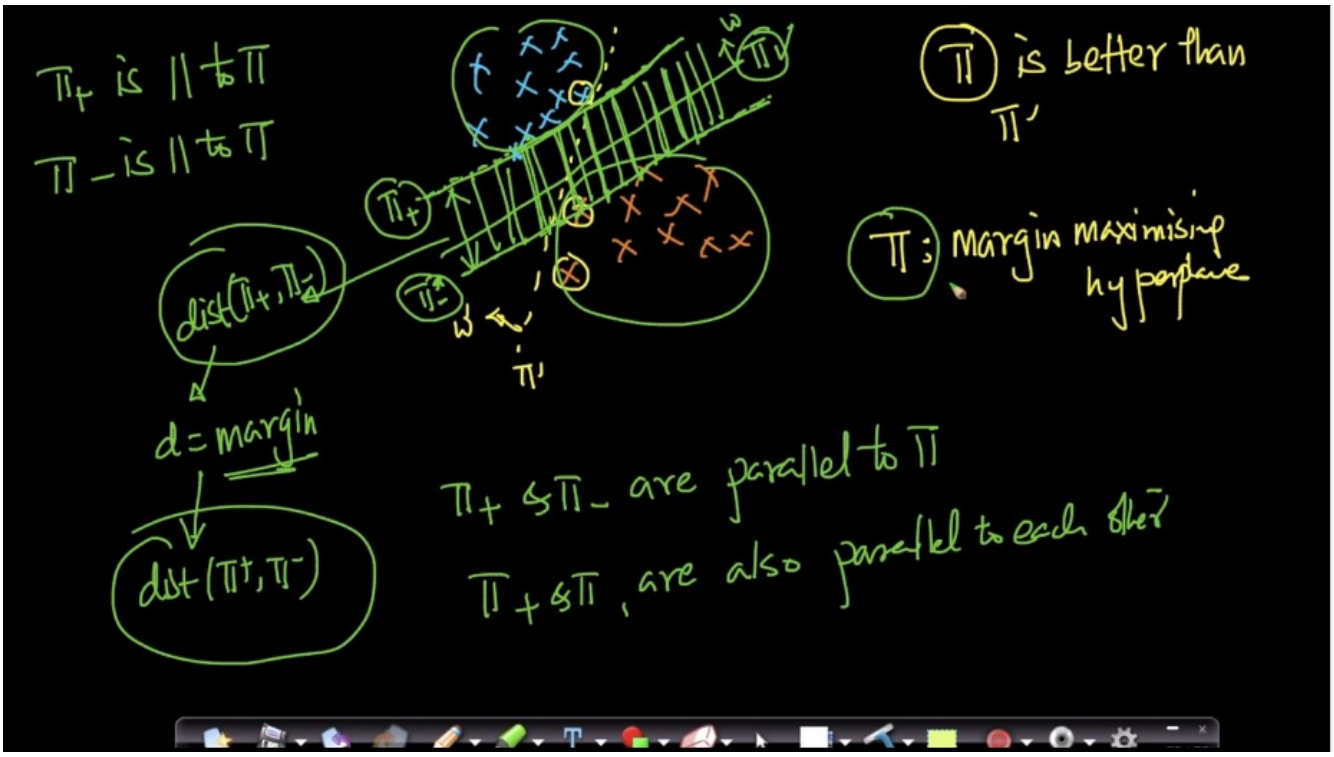
NOTES

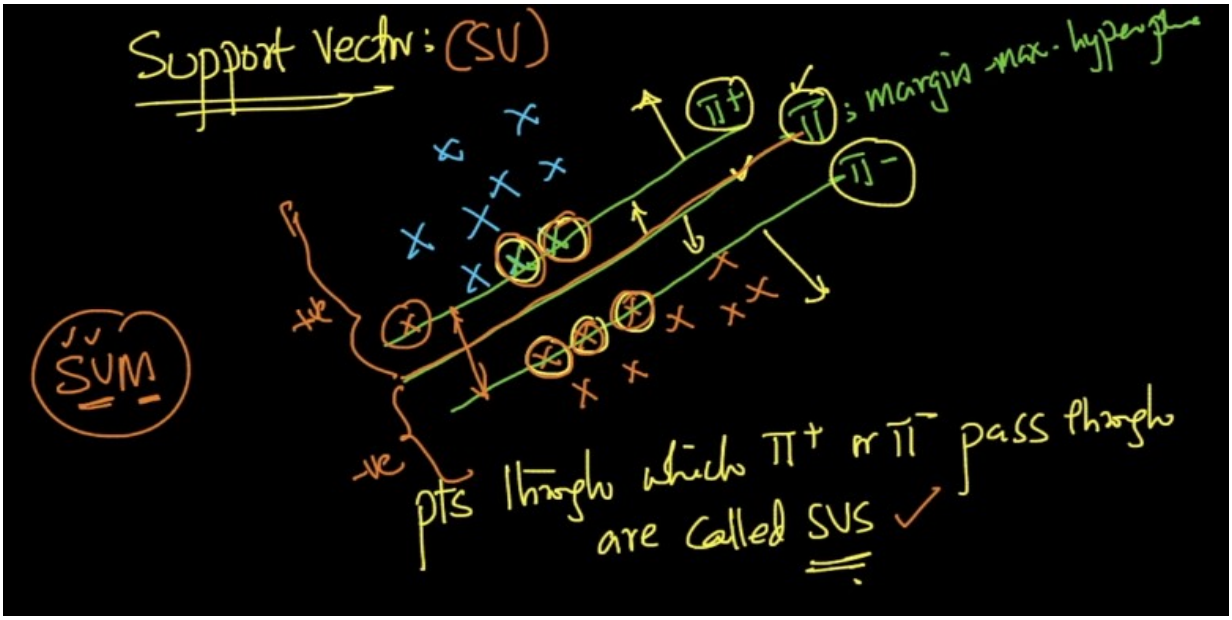
Support vector machines:

Let’s assume the data is linearly separable. There will be many hyper planes that can separate the data.

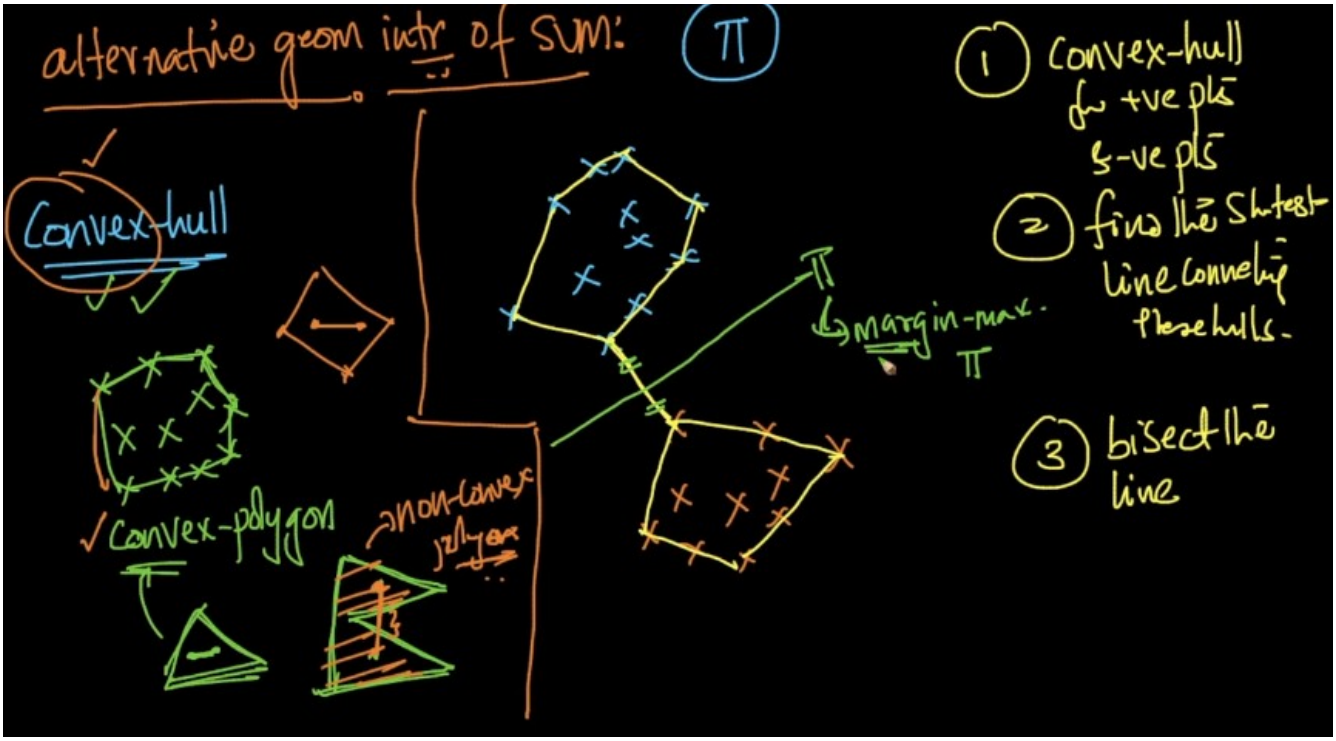
Fitting the hyper plane that can separate the two classes of data as widely as possible.



SVM’s try to find the hyper plane that maximizes the margin = dist(pi+, pi-), As margin increases the generalization error decrease.

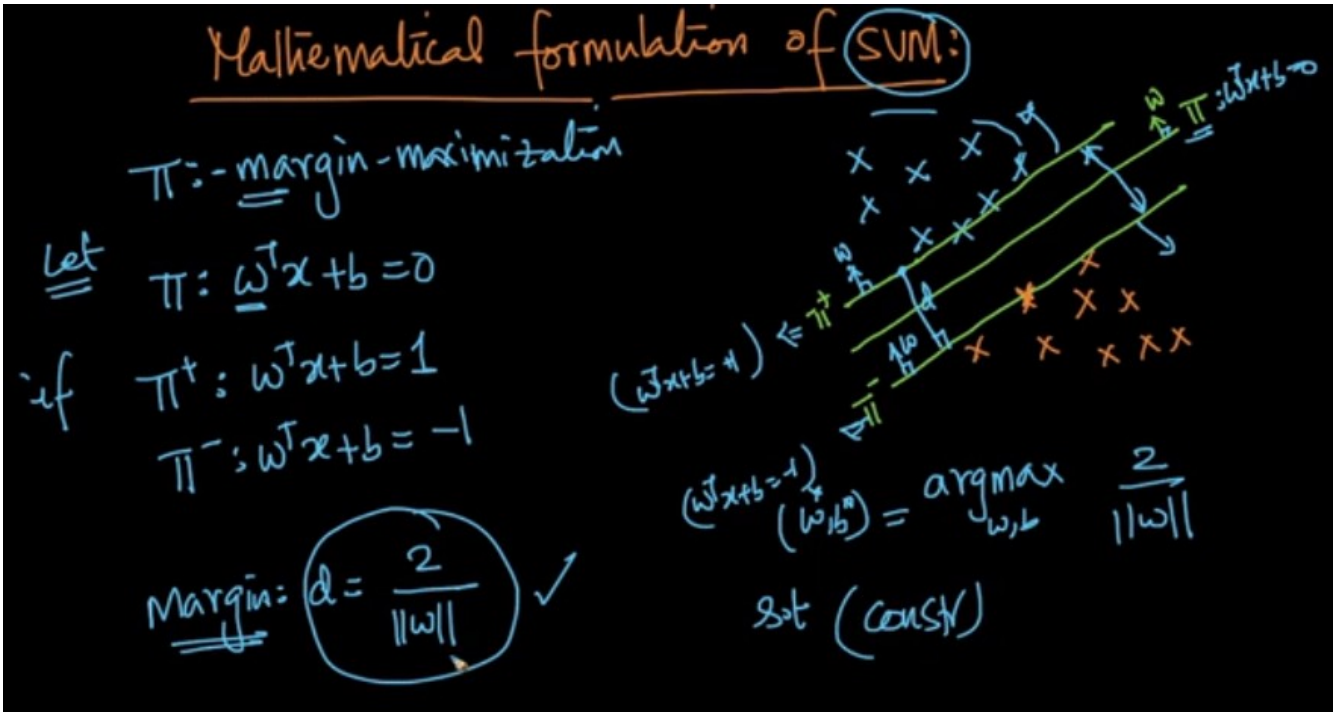


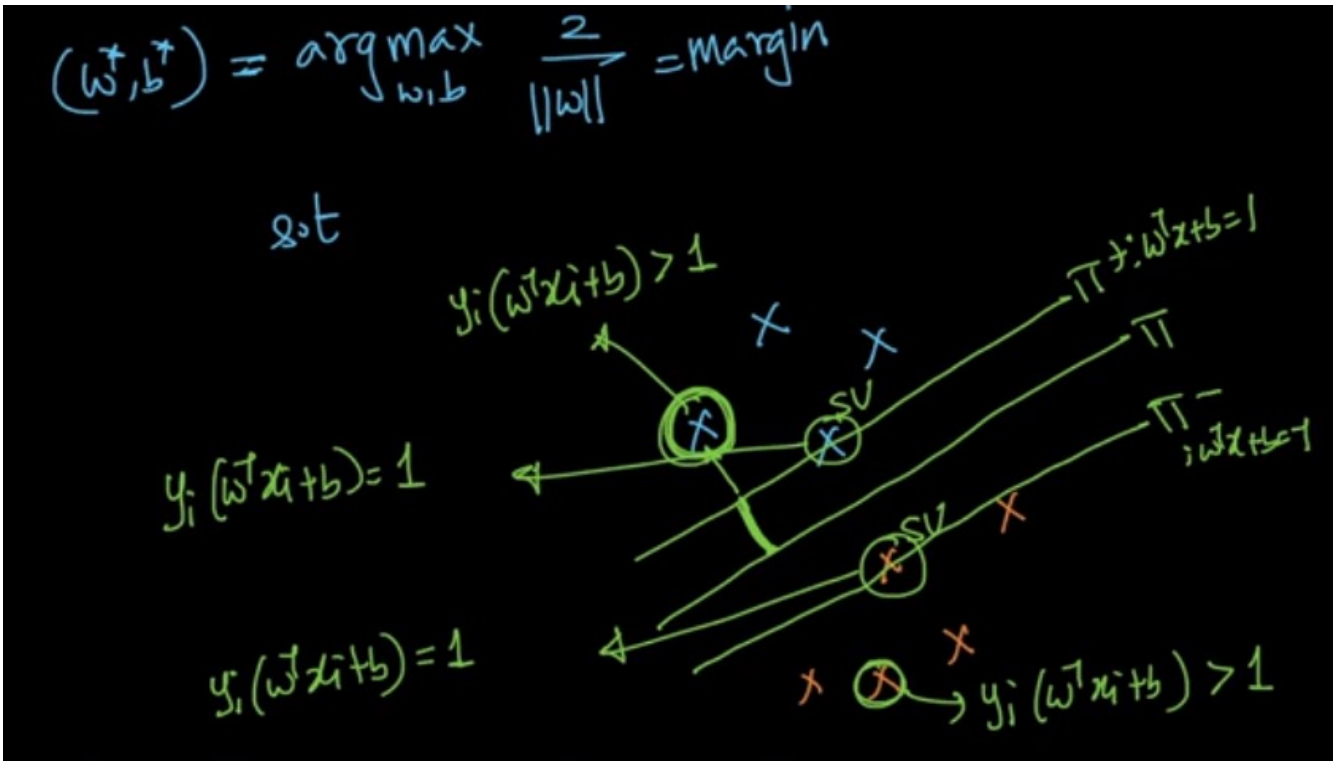
Alternative geometric intuition of sum:
First we make a convex polygon:
The points in the convex hull must be inside or on the convex hull.



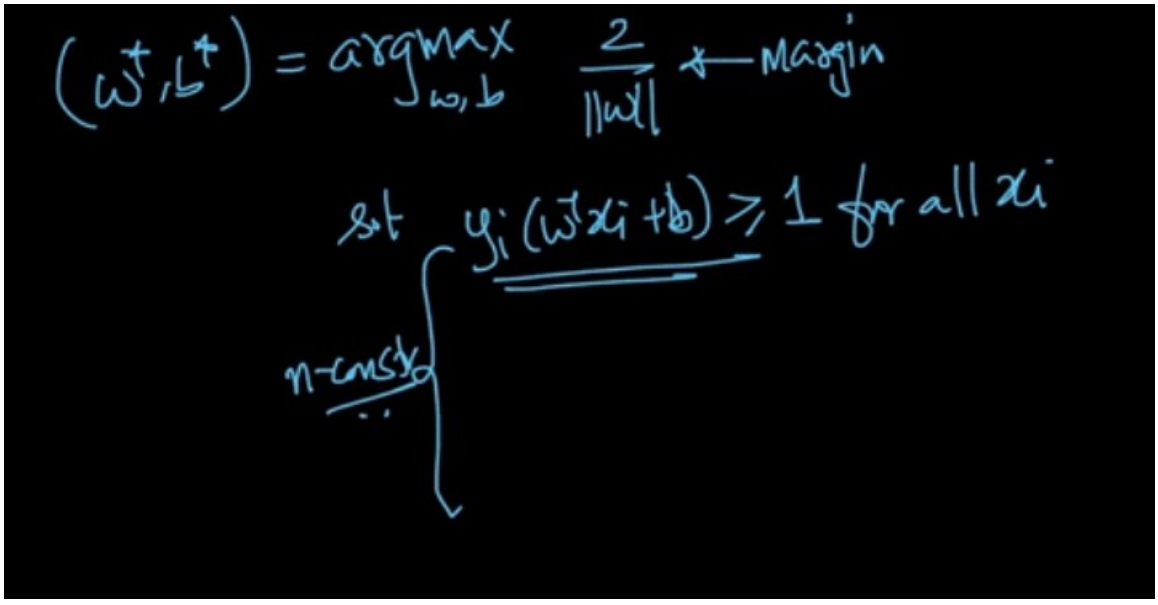
Mathematical derivation:

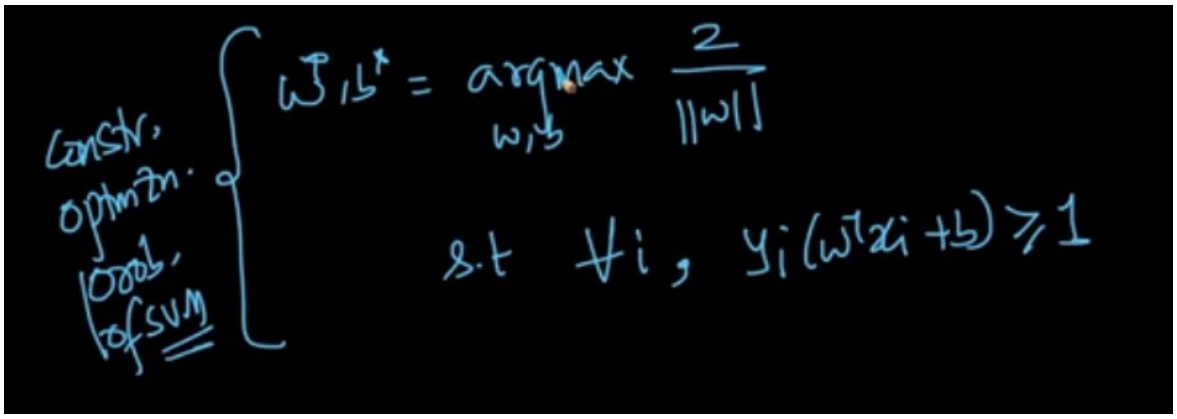
Objective: We want to find a hyper plane that does margin maximization.





The optimization function is find w*, b*.

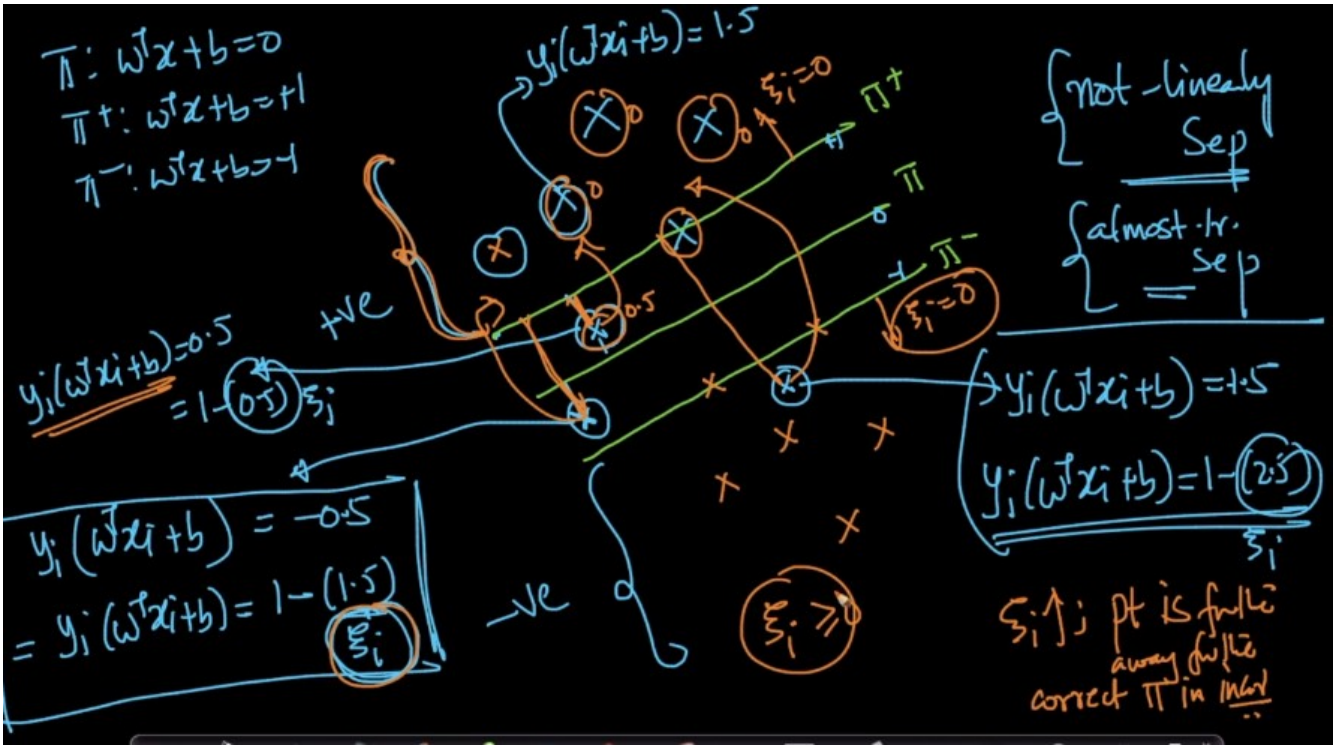


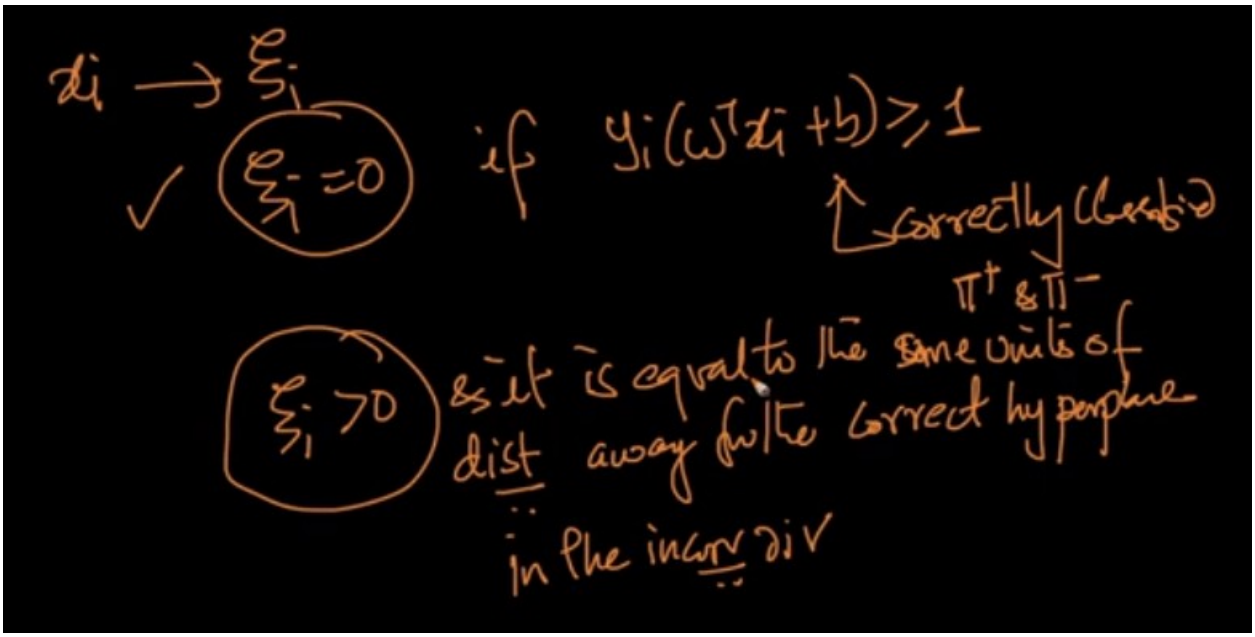


This works only when the data is linearly separable, this is also called as hard margin SVM.

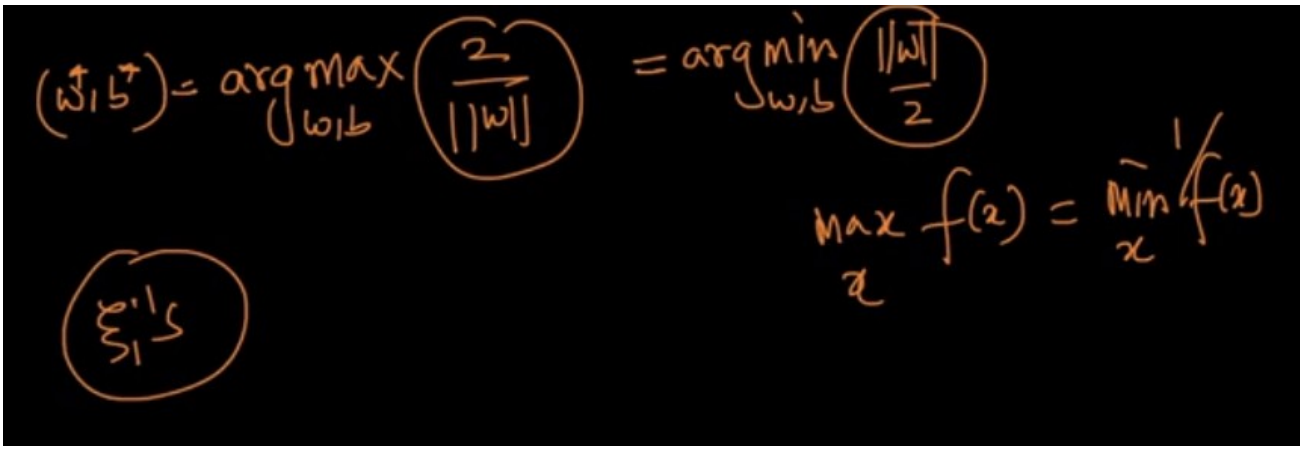
Edge cases:

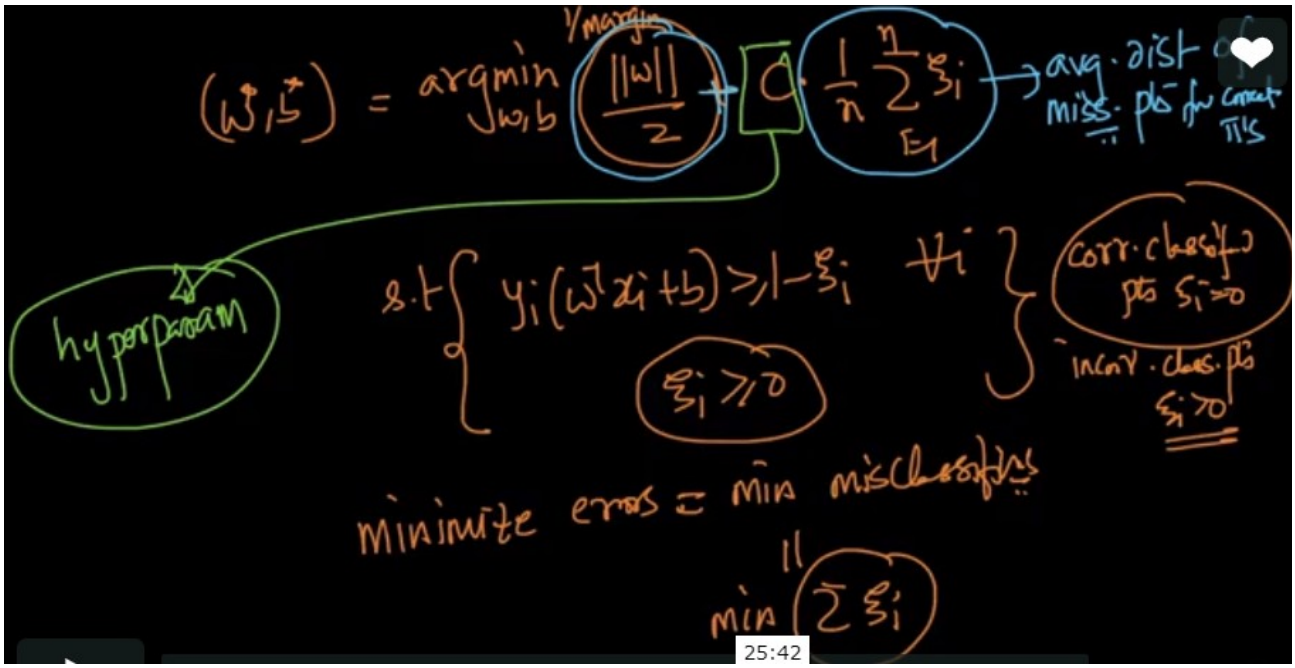
Not – linearly separable data, alternate formulation of the edge points. As zeta(i) increases the points will be away from the SVM.



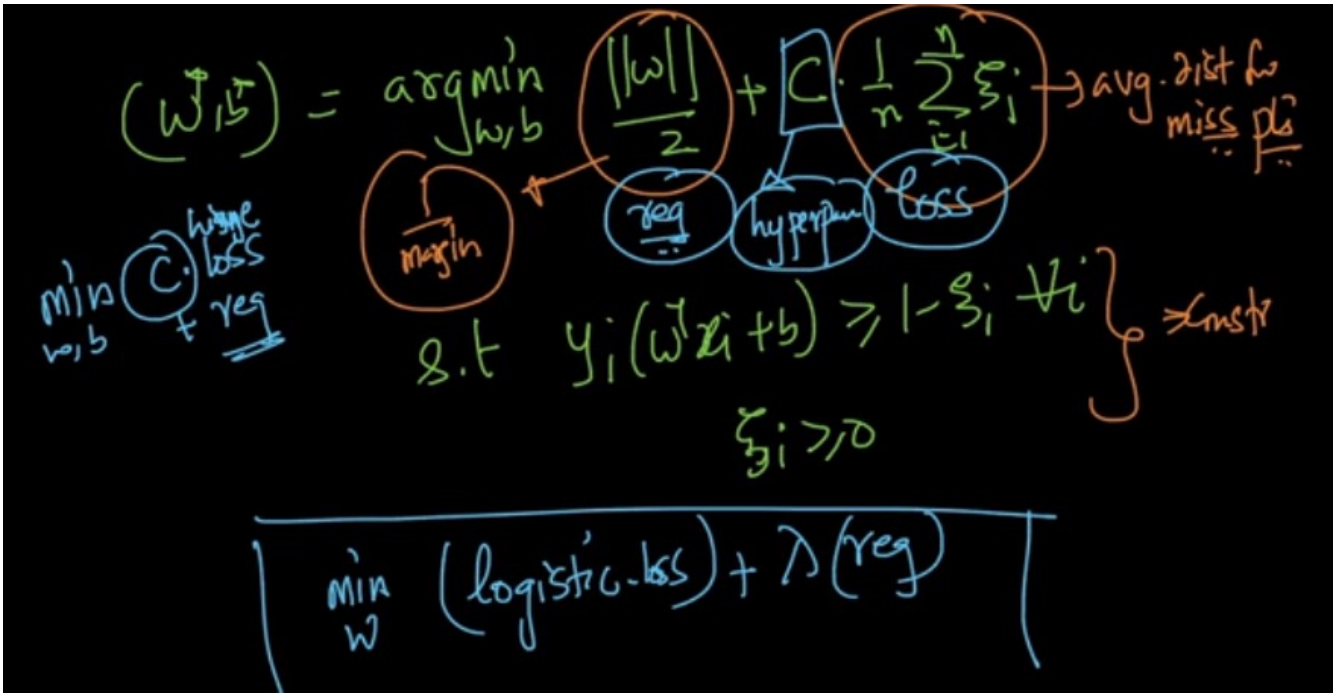


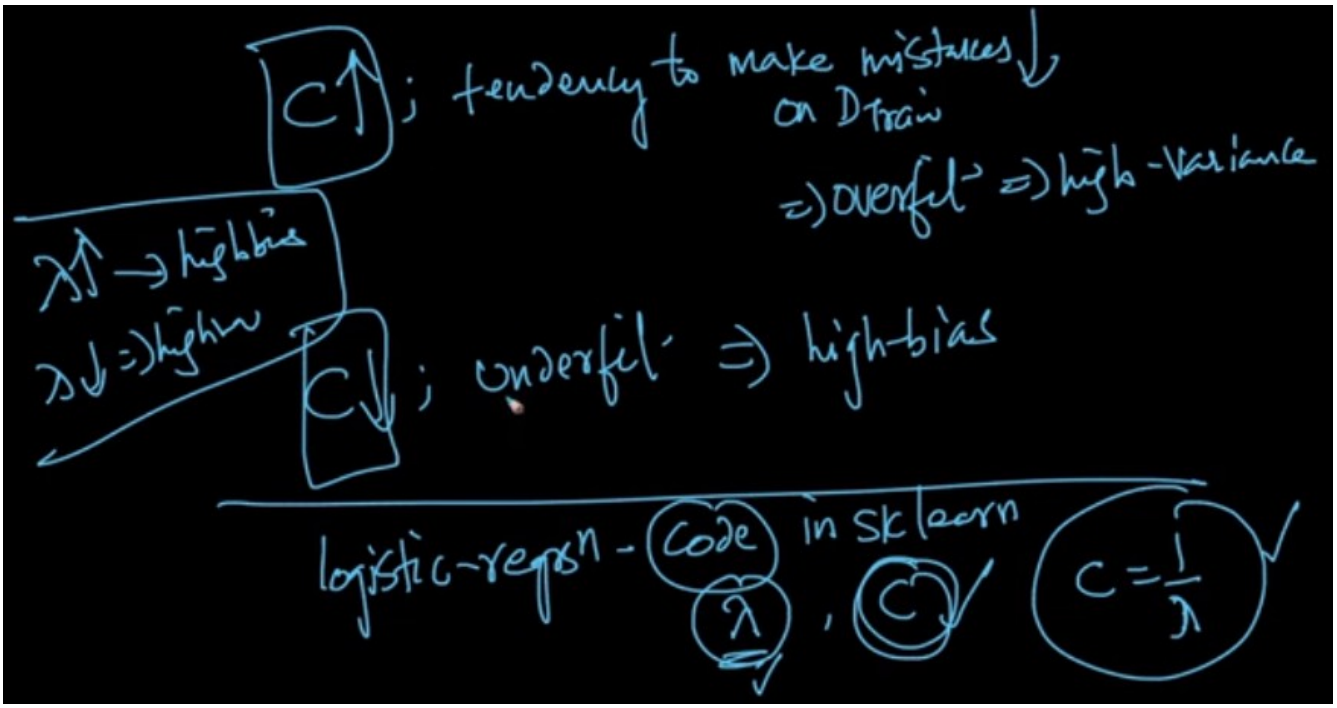
Changes to the formulation.



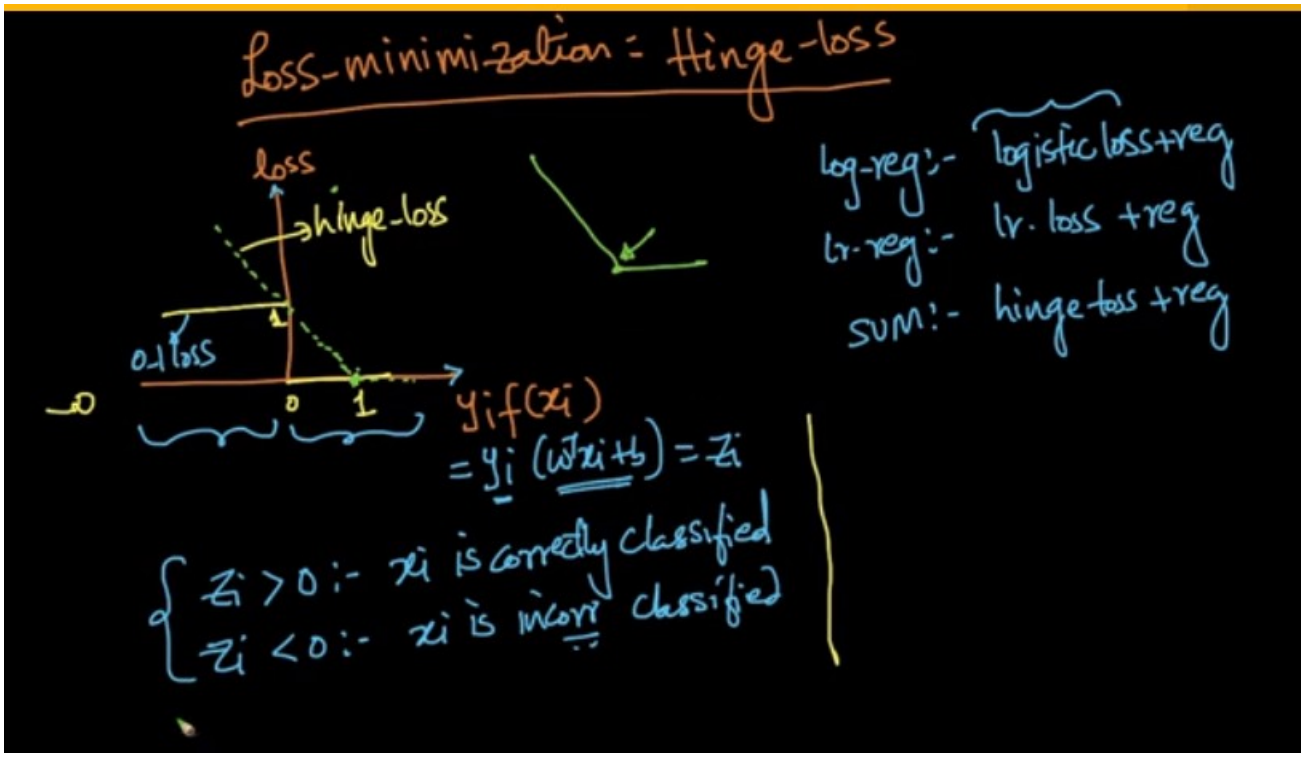


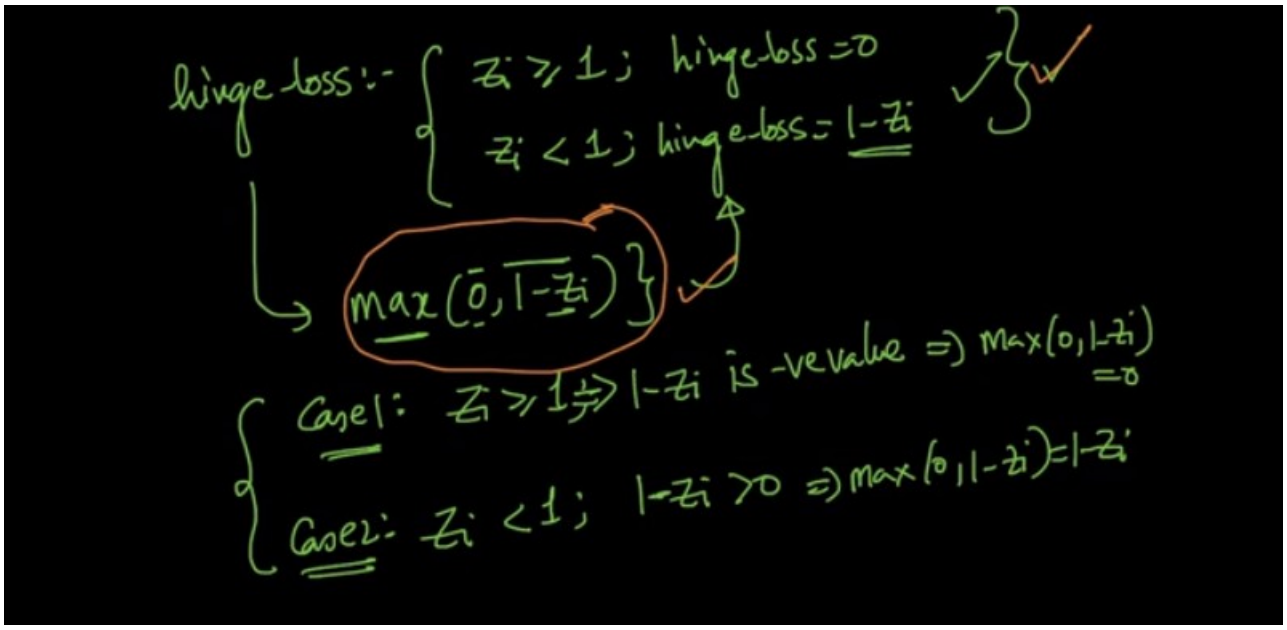
Modified loss function:



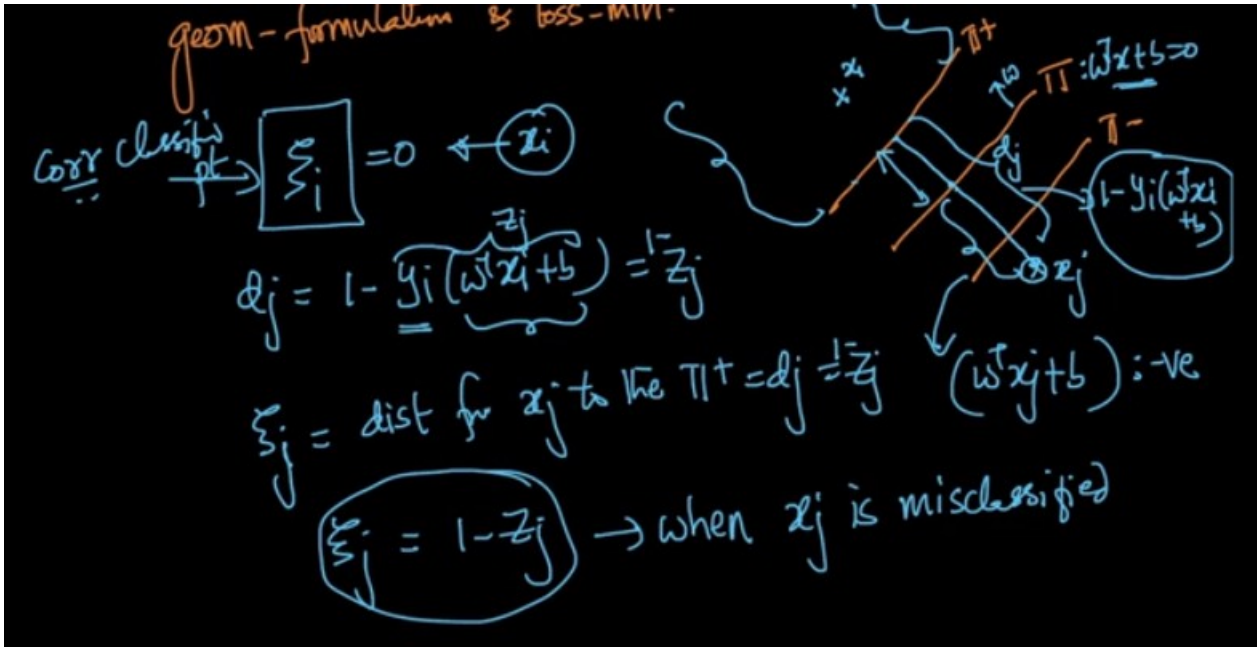


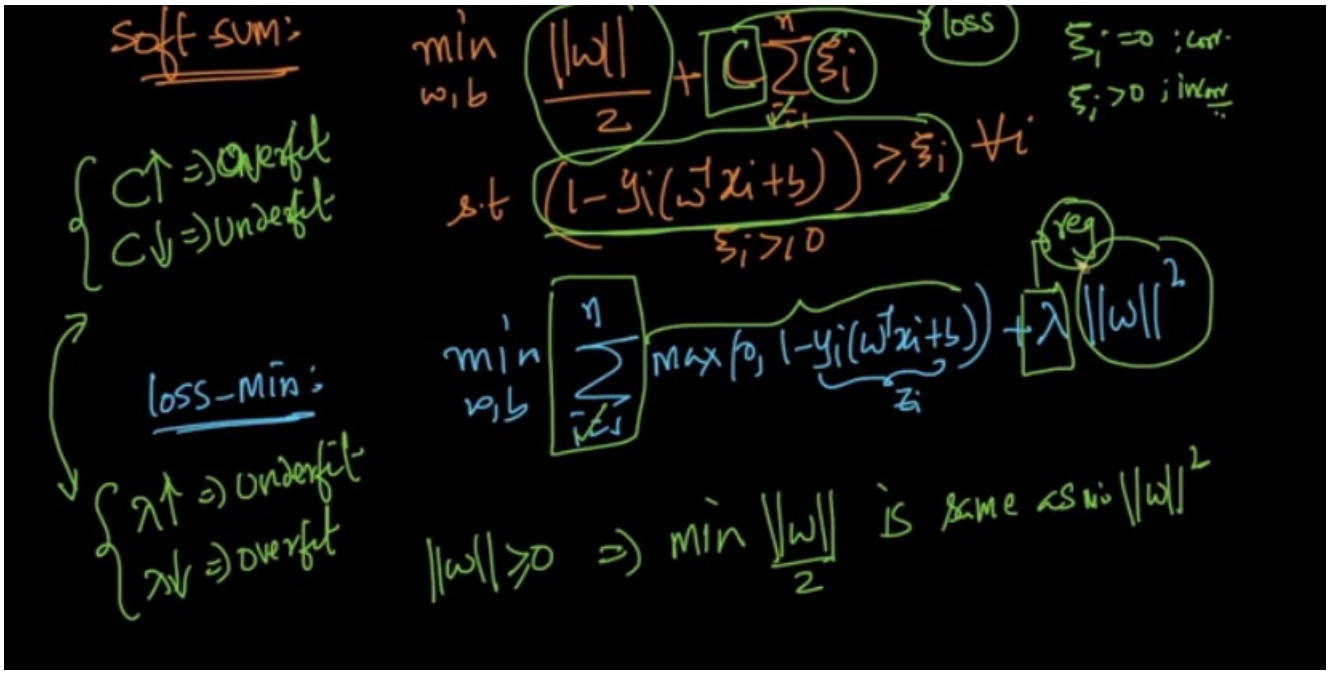
Loss function(Hings Loss) based interpretation:



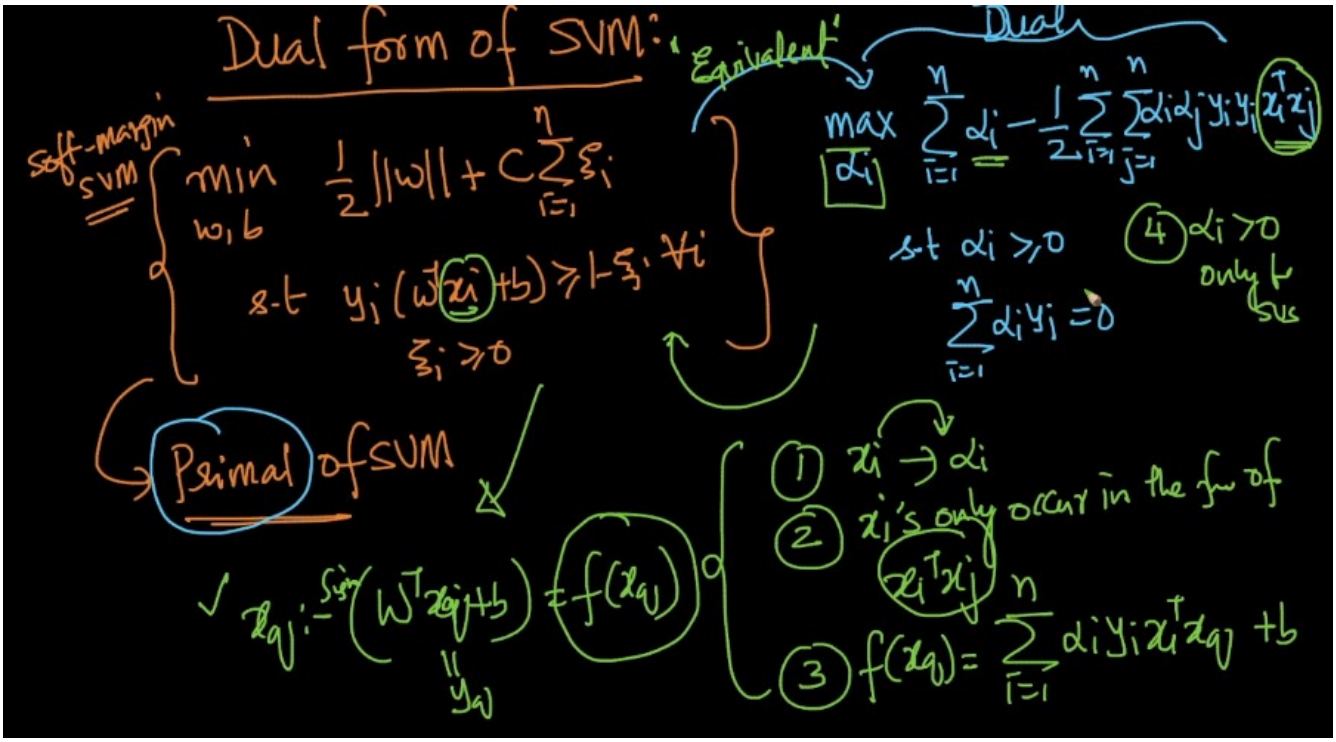


This is alternate formulation of Hinge Loss.

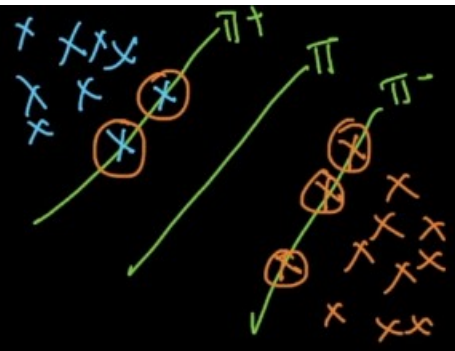




Dual for of SVM formulation:



✓ $f(x_q) = \sum_{i=1}^{n} \alpha_i y_i x_i^T x_q + b$



SVs :- $\alpha_i > 0$

non SVs :- $\alpha_i = 0$

$f(x_q)$:- only pts that matter are SVs

SVM

Kernal Trick:

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \underbrace{x_i^T x_j}_{\rightarrow Sim(x_i, x_j)} \rightarrow K(x_i, x_j)$$

$$s.t \sum_{i=1}^{m} \alpha_i y_i = 0 \; ; \alpha_i \geq 0$$

↳ Kernel fn

$$f(x_q) = \sum_{i=1}^{n} \alpha_i y_i \overbrace{K(x_i, x_q)} + b$$

→ The most imp. idea in SVM is kernel-trick

Soft-SVM-hyperplanes ≈ log-reg

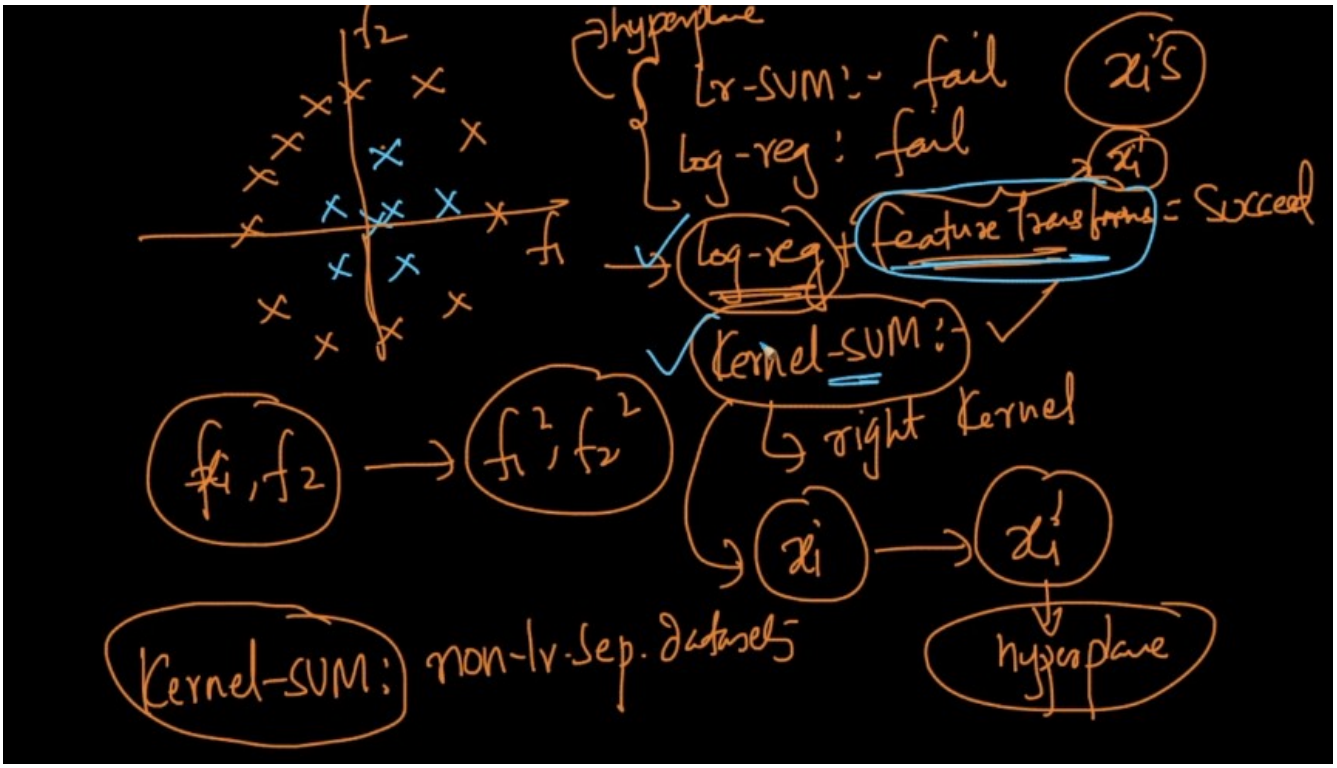
↳ margin-max.

$K(x_i, x_j) = x_i^T x_j$

Lr. SVM:- $x_i^T x_j$

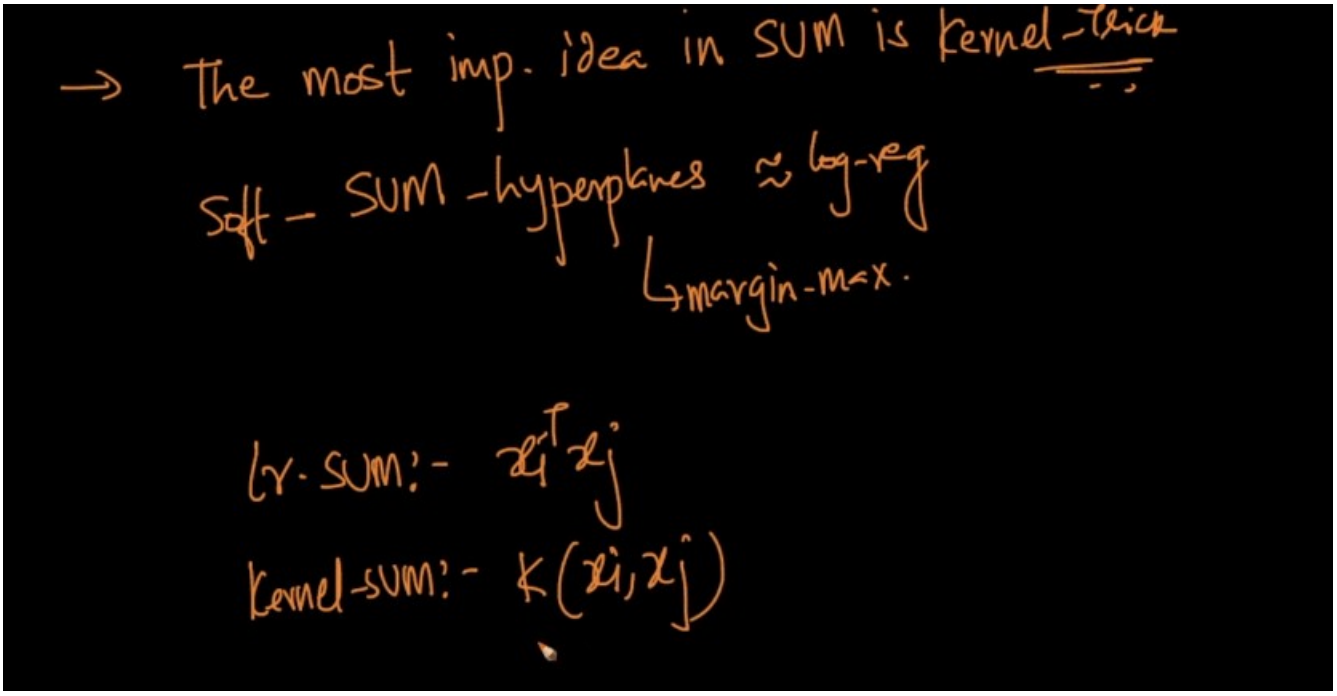
Kernel-SVM:- $K(x_i, x_j)$

Failure cases of SVM:

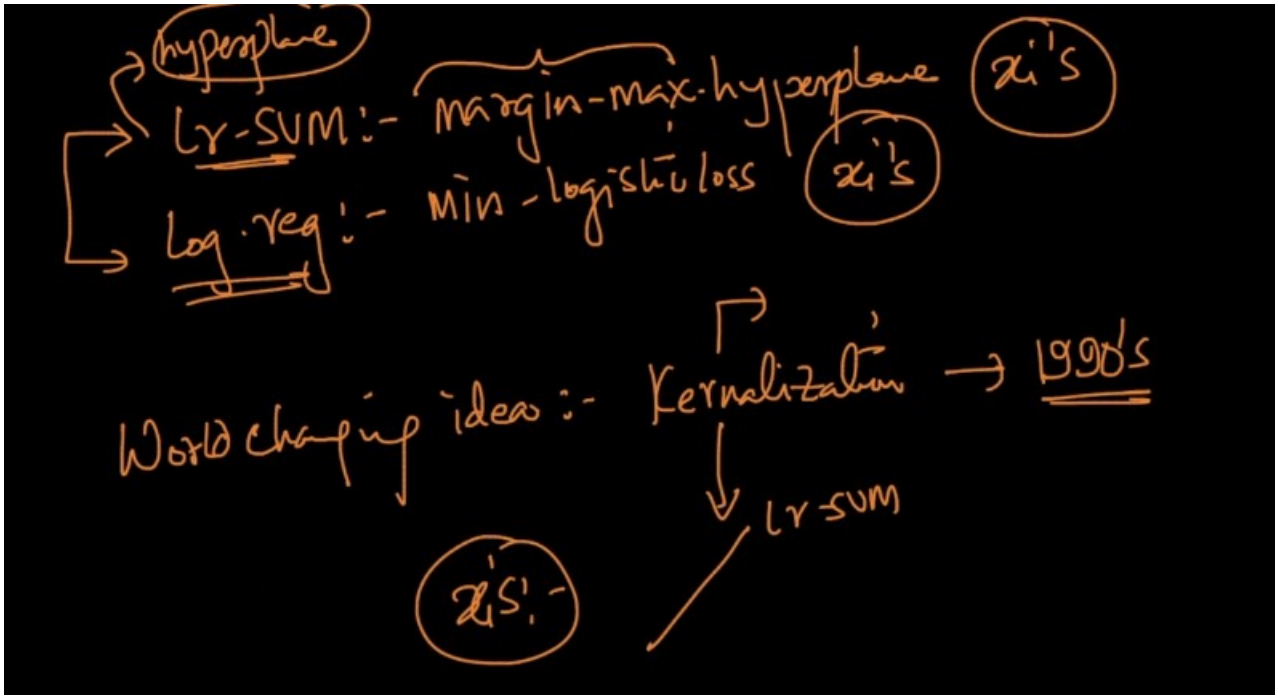


Kernalization makes SVM handle non – linearly separable data sets.

We use kernels in the dual form of SVM to make different relationships other than the default linear assumption.

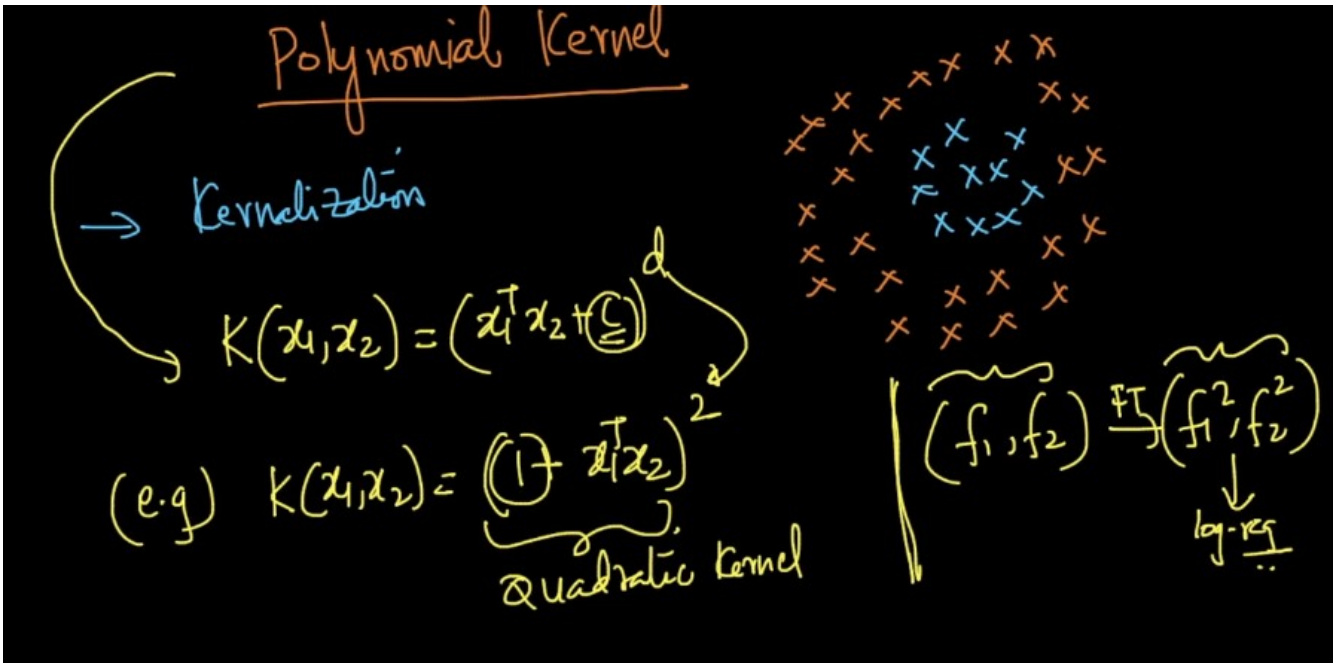


The key idea of SVM's is Kernel trick.

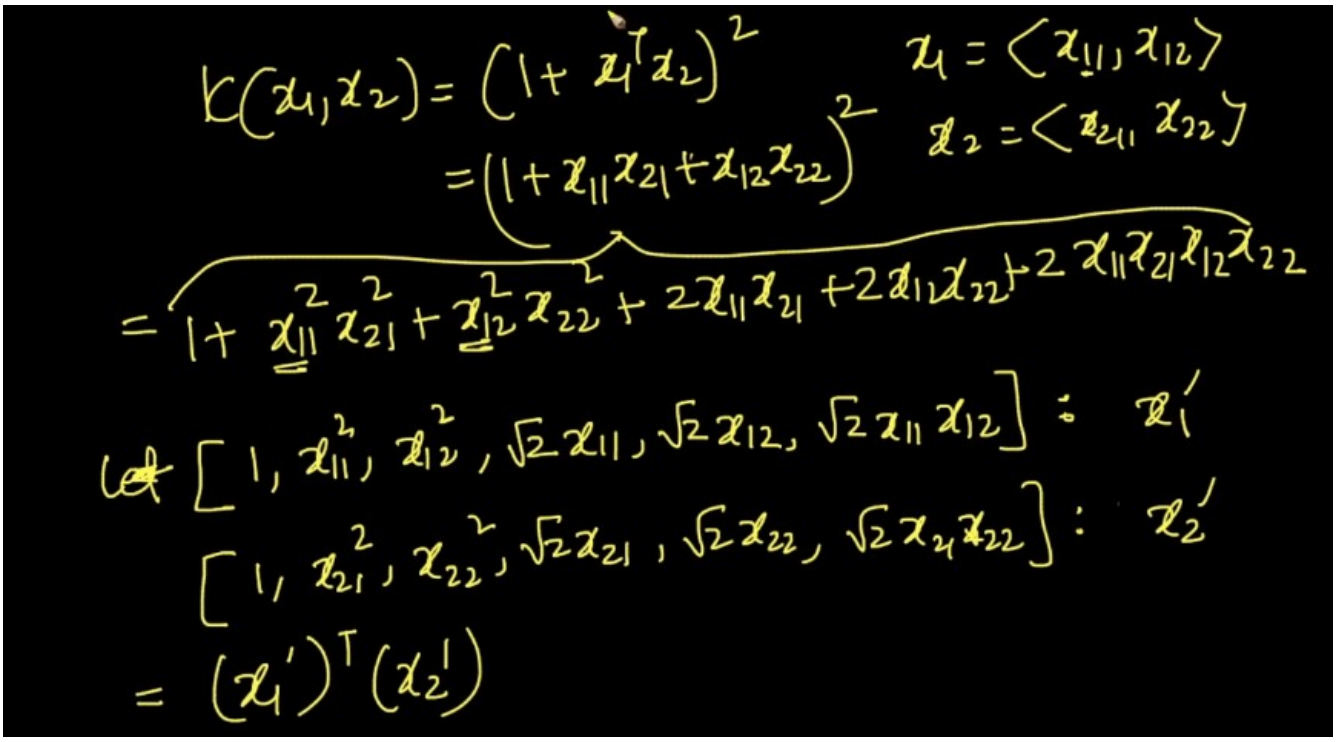


Non – linearly seperable data sets can be classified using SVM and kernels.

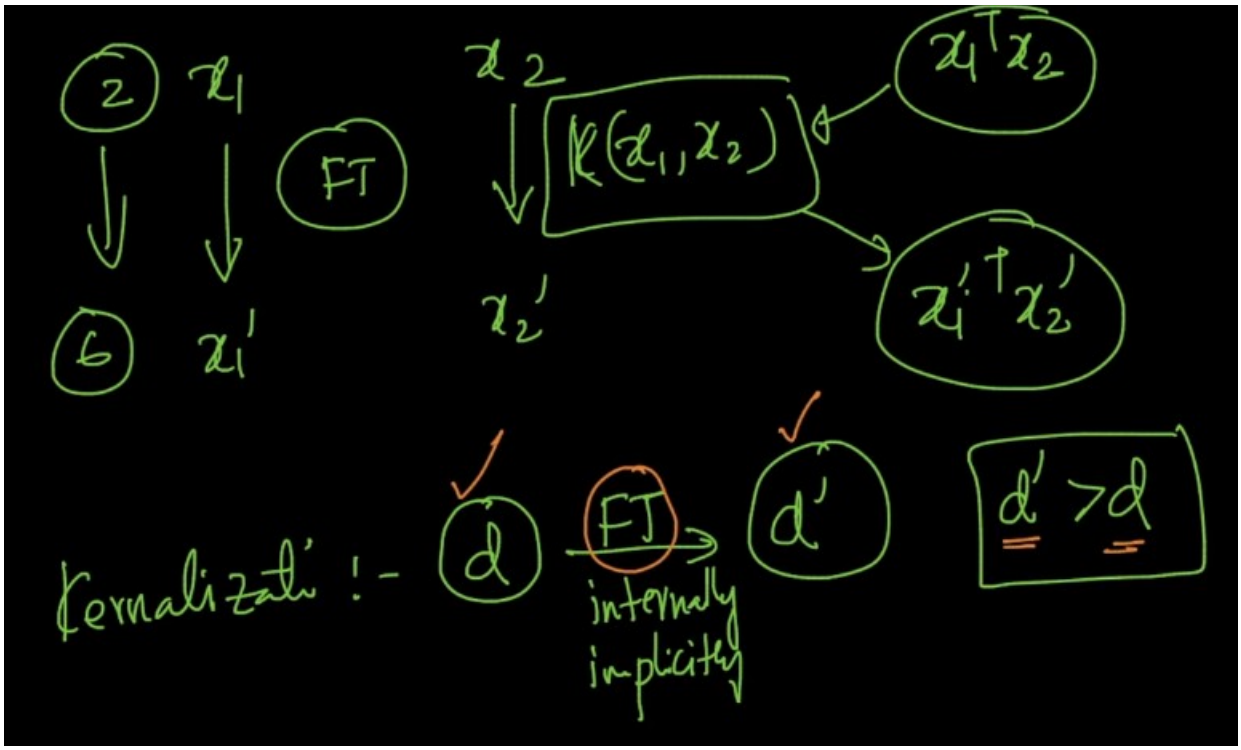
Polynomial kernel:



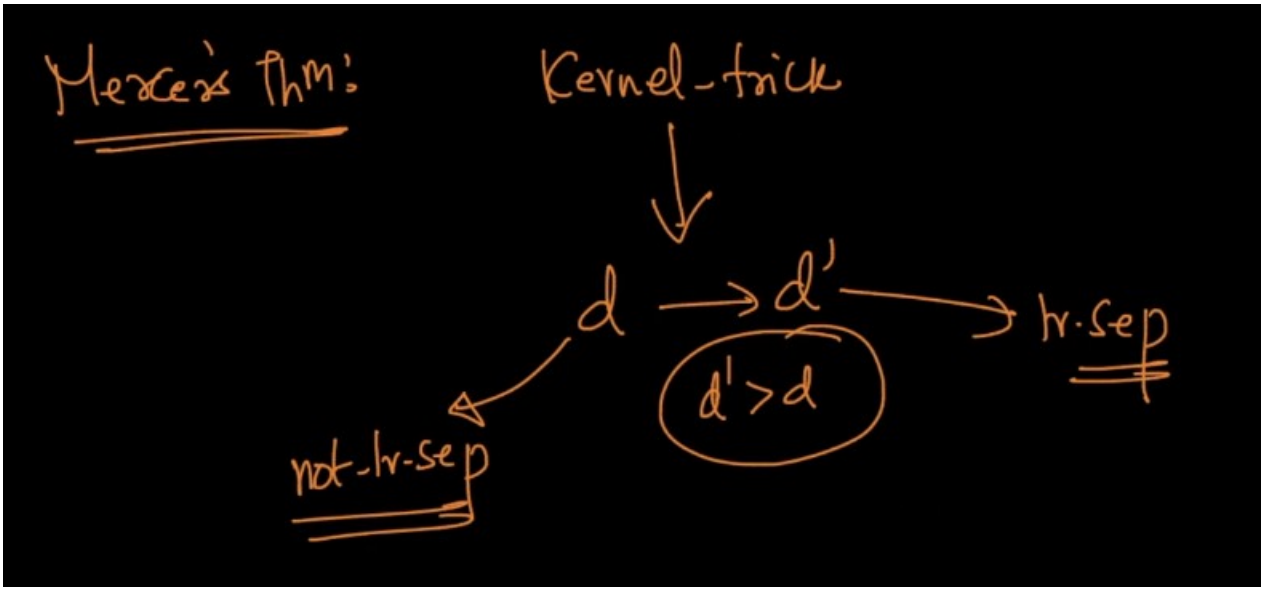
Example:



Kernelization to the feature transform internally using the kernel trick.



The data will become separable after kernel trick. After applying mercers theorem the data will be more linearly separable transforming to d' dimensional space.

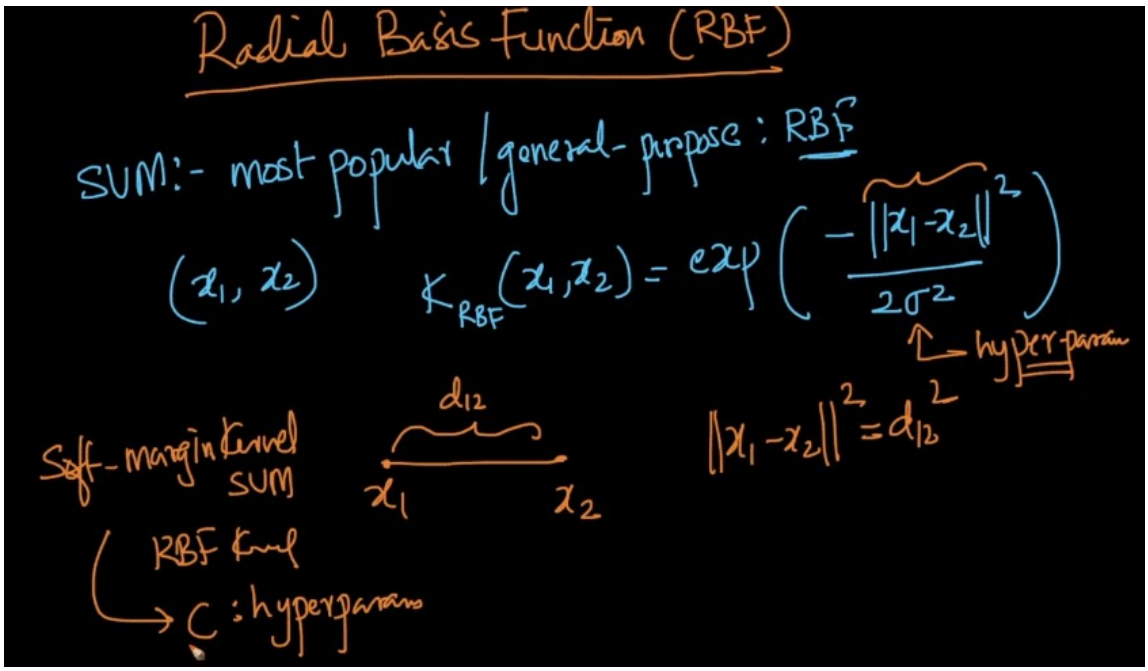


Kernel trick does the transformation internally than that of the feature transform that is done externally in logistic regression.

What is a right Kernel to apply?

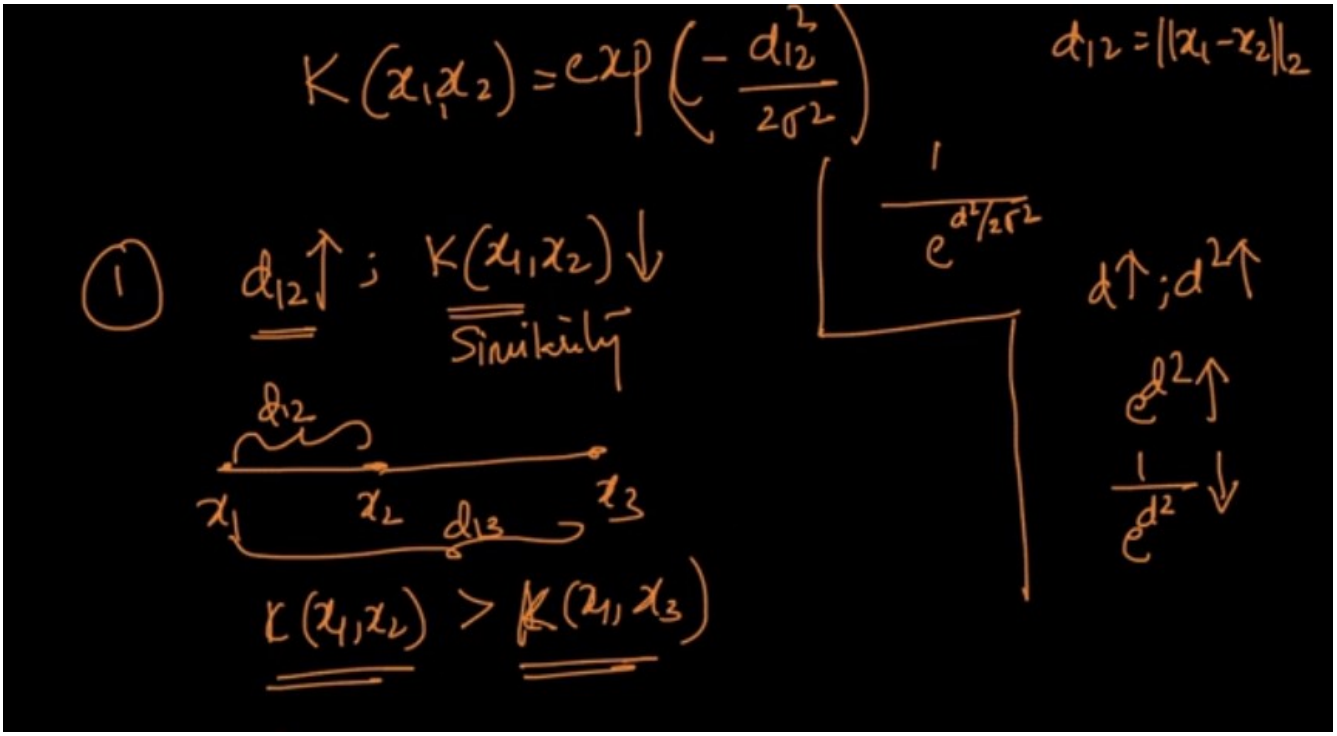
The problem of explicit feature transform is changed to find the right kernel in SVM than that of Logistic regression.

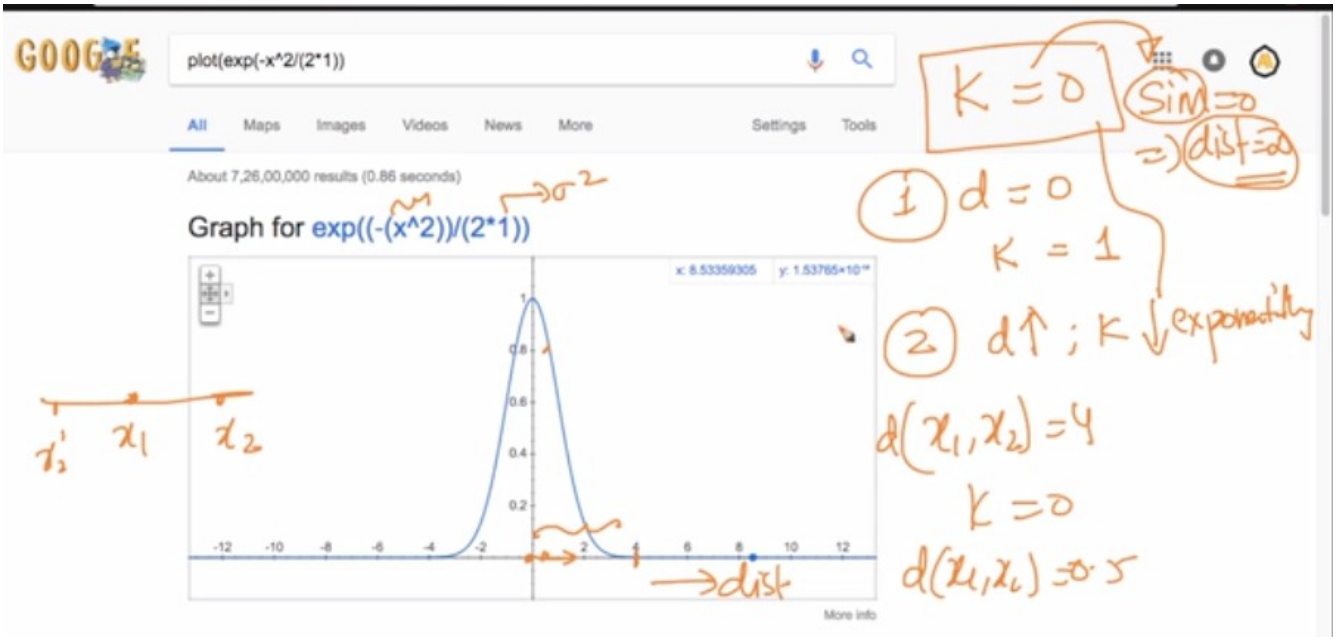
RBF – kernel:



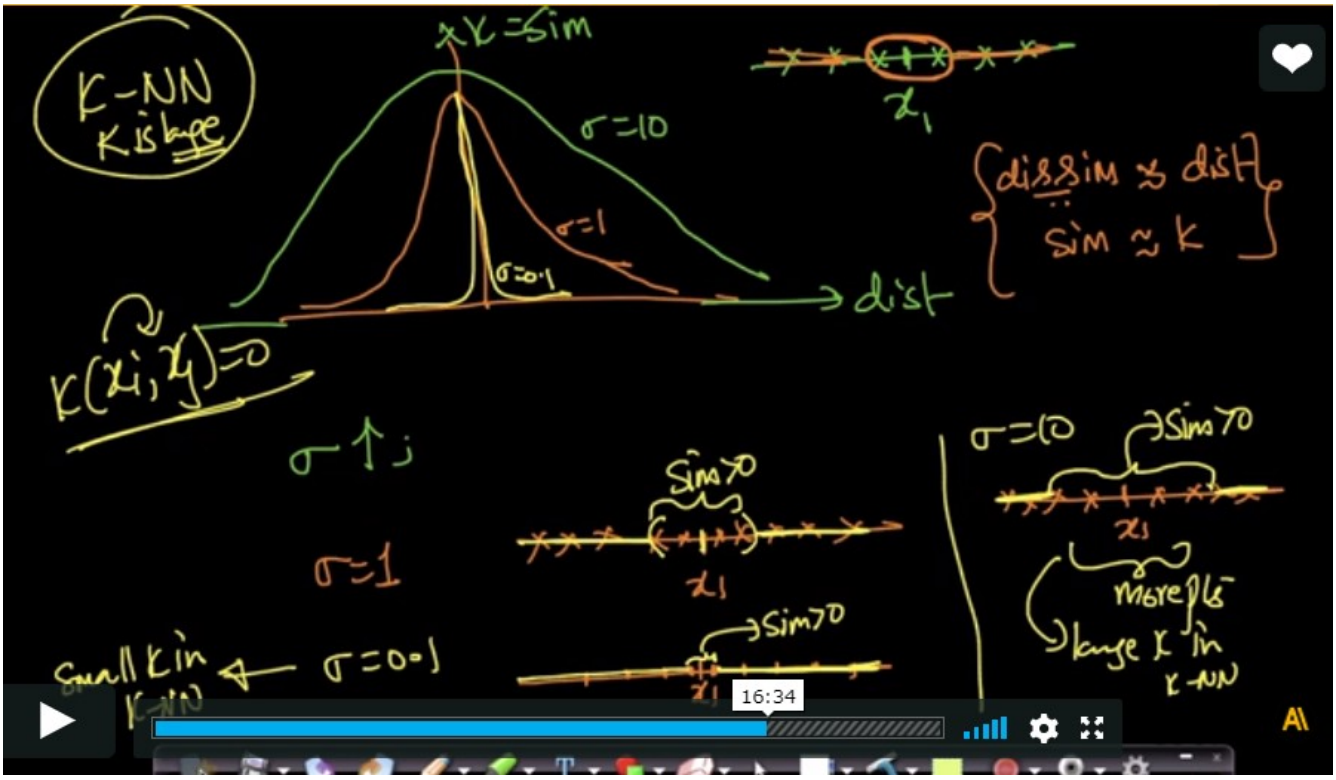
Here sigma is the hyper parameter in RBF. As distance between the points increases then the kernel decreases.

Kernels work like similarity distance measures.

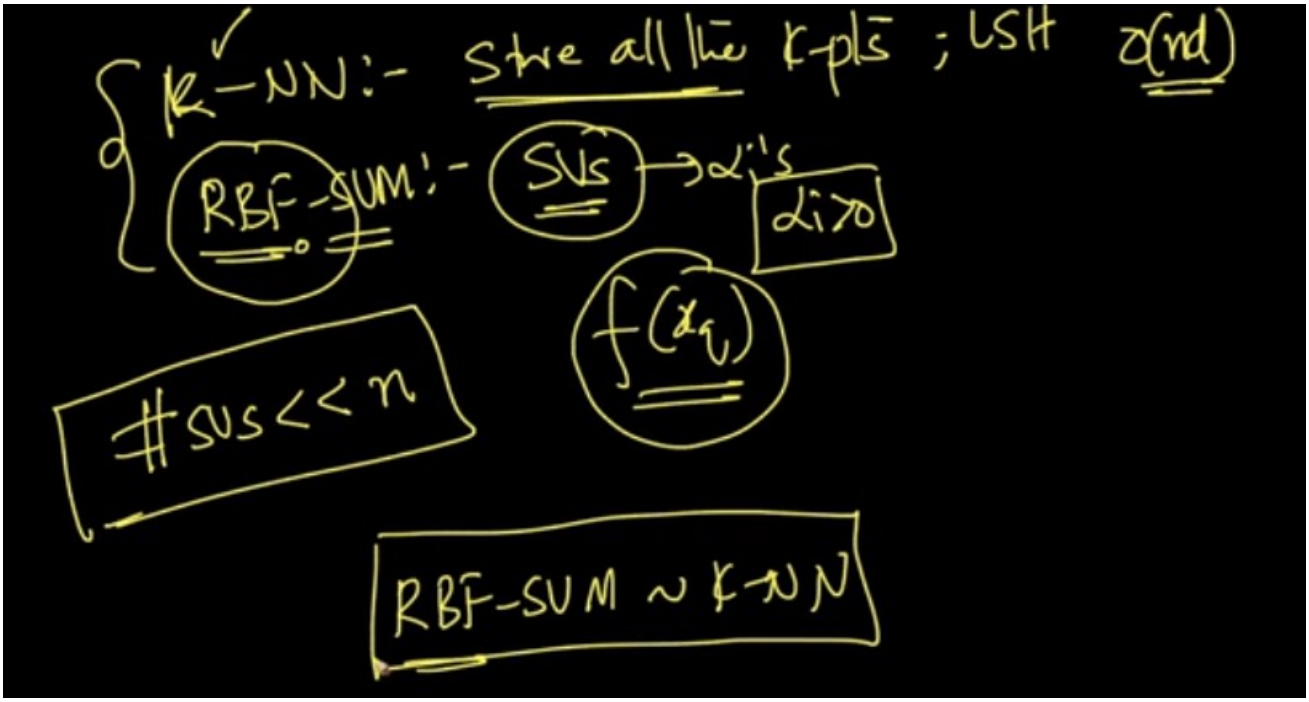




Kernels measure the similarity.
Distance measure the distance between the data points.

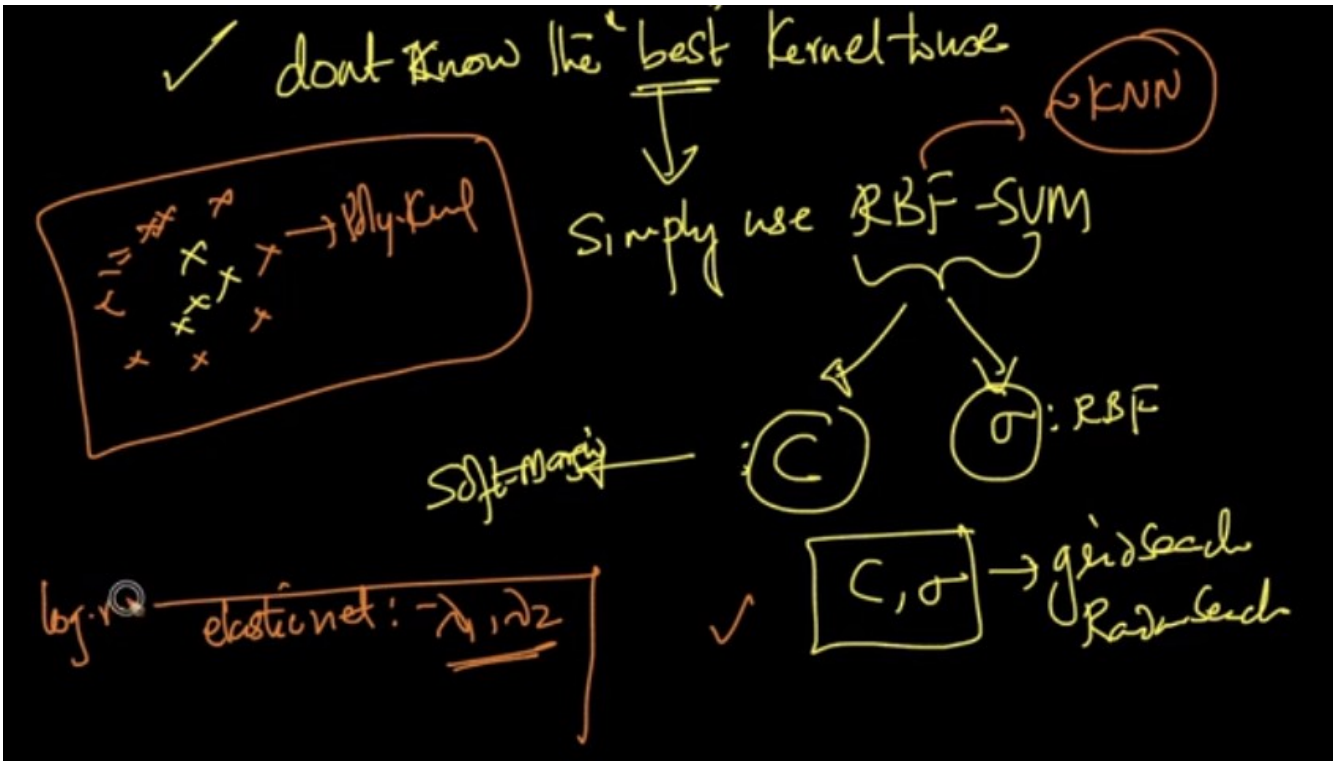


KNN and RBF- Kernel



If there is no clue of choosing the kernel, RBF will be the good choice.

Elastic nets have two hyper parameters

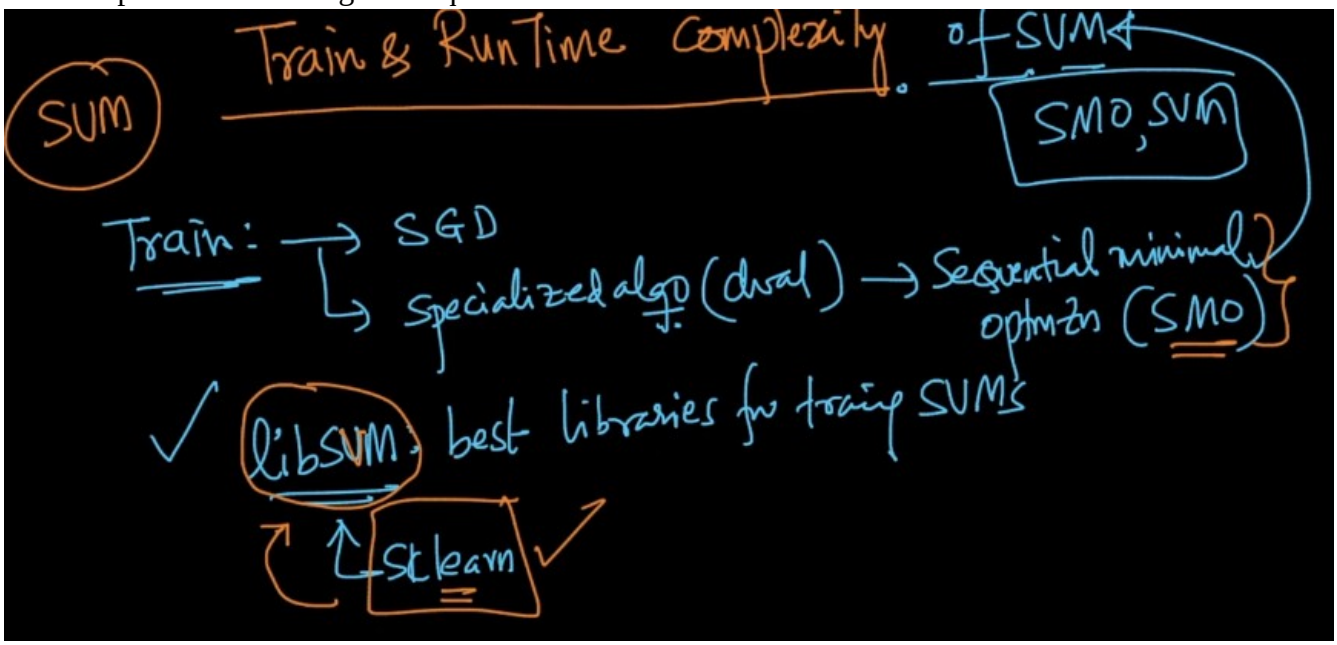


Domain specific Kernels:

There area lot of specify kernels,

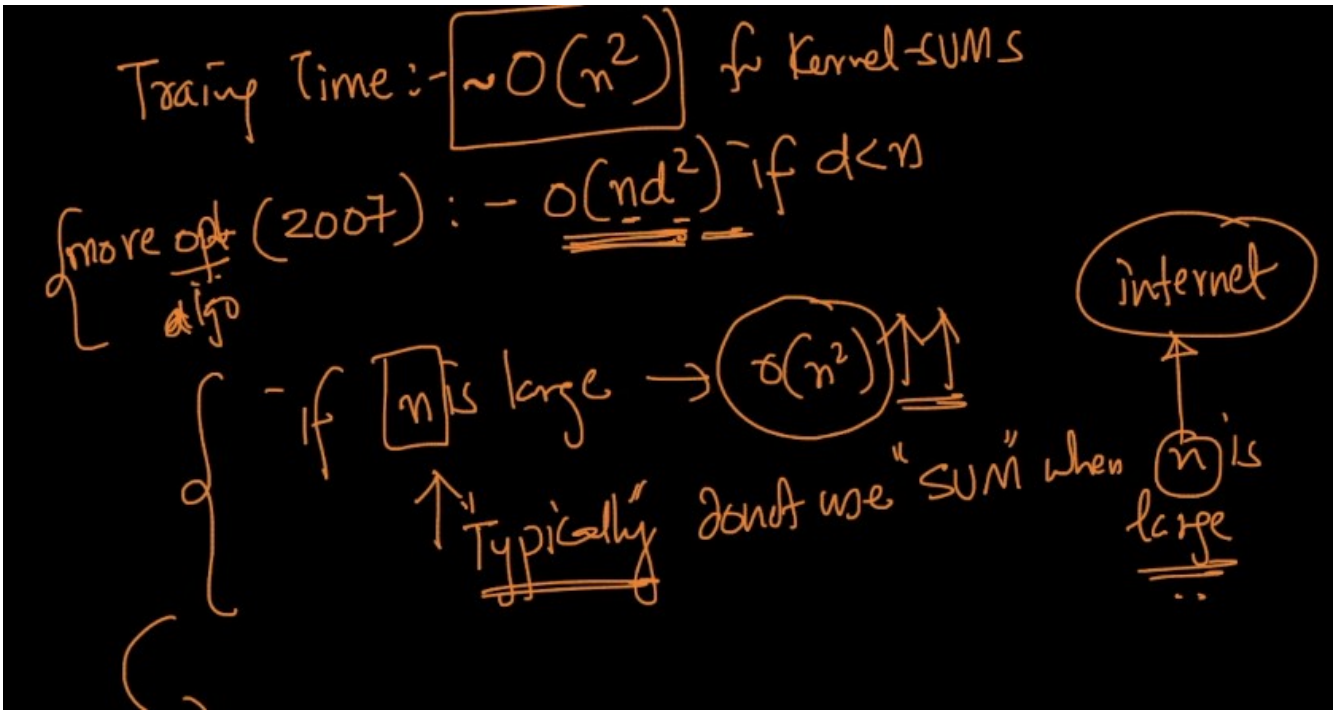
String kernels – If the features contains text.
Geonmic kernels
Graph kernels

Train and run time complexities of SVM.
There are specialized training techniques for SVM such as libsvm.

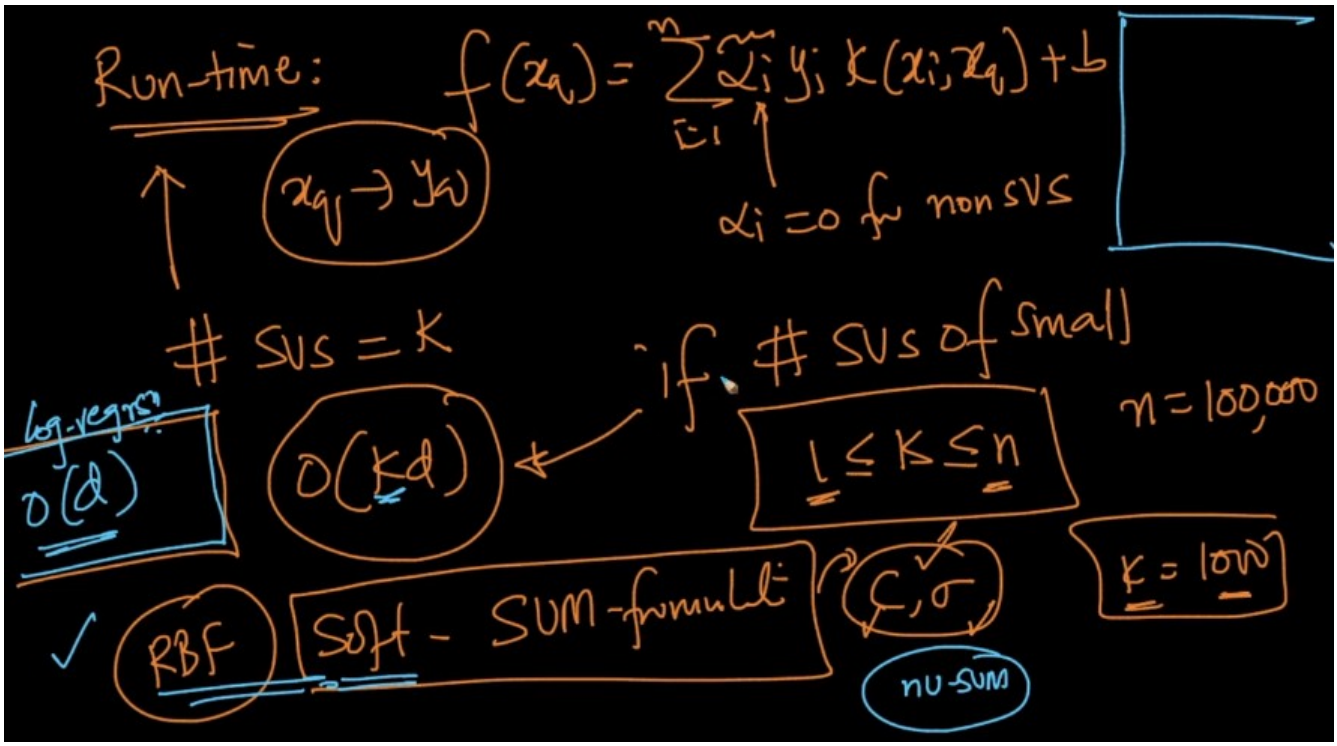


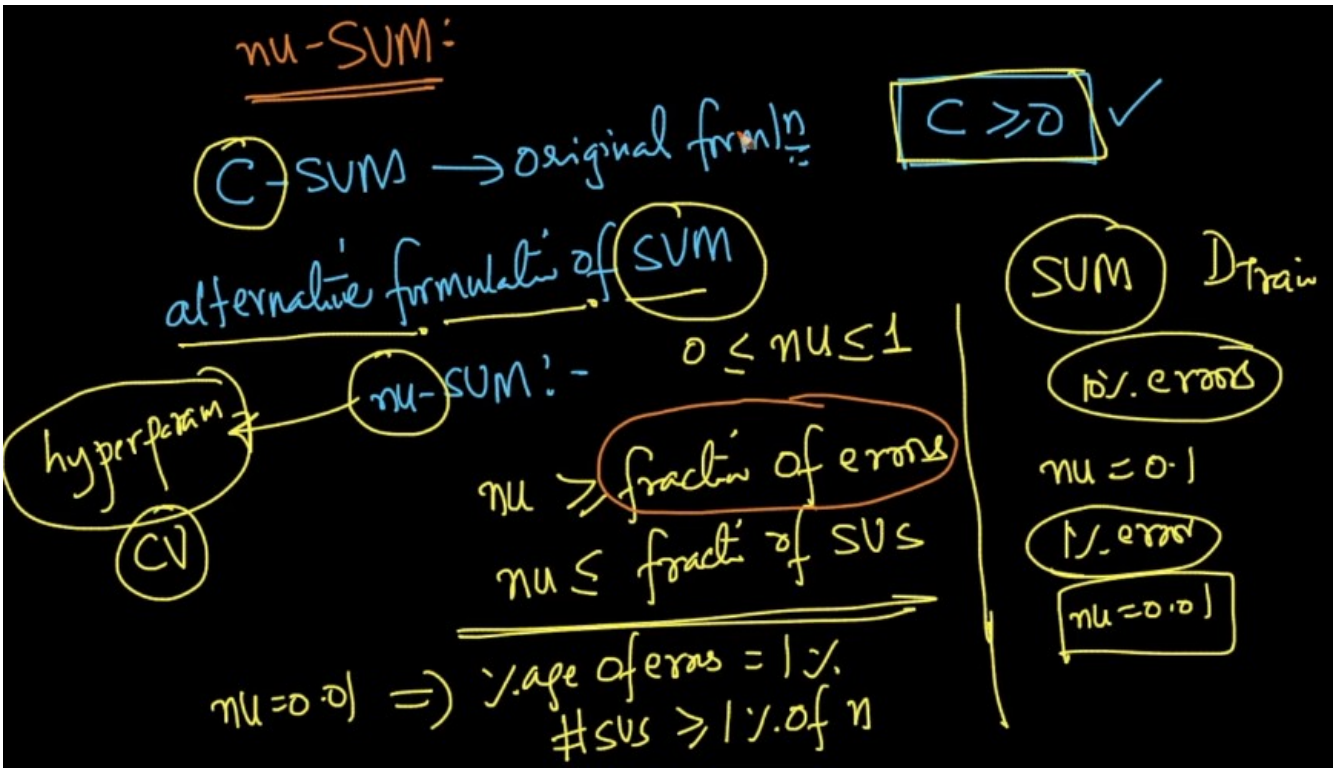
The training time of the SVM is roughly around O(n^2).

Typically, we don’t use SVM when number of data points are large.

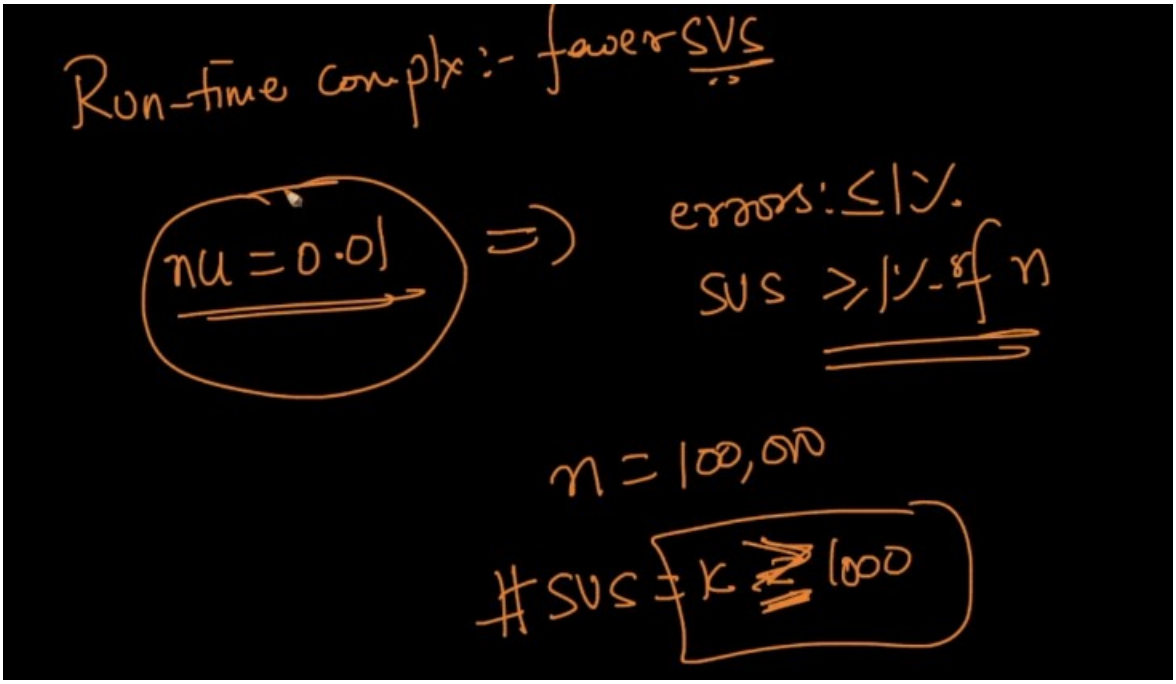


Run time complexity:

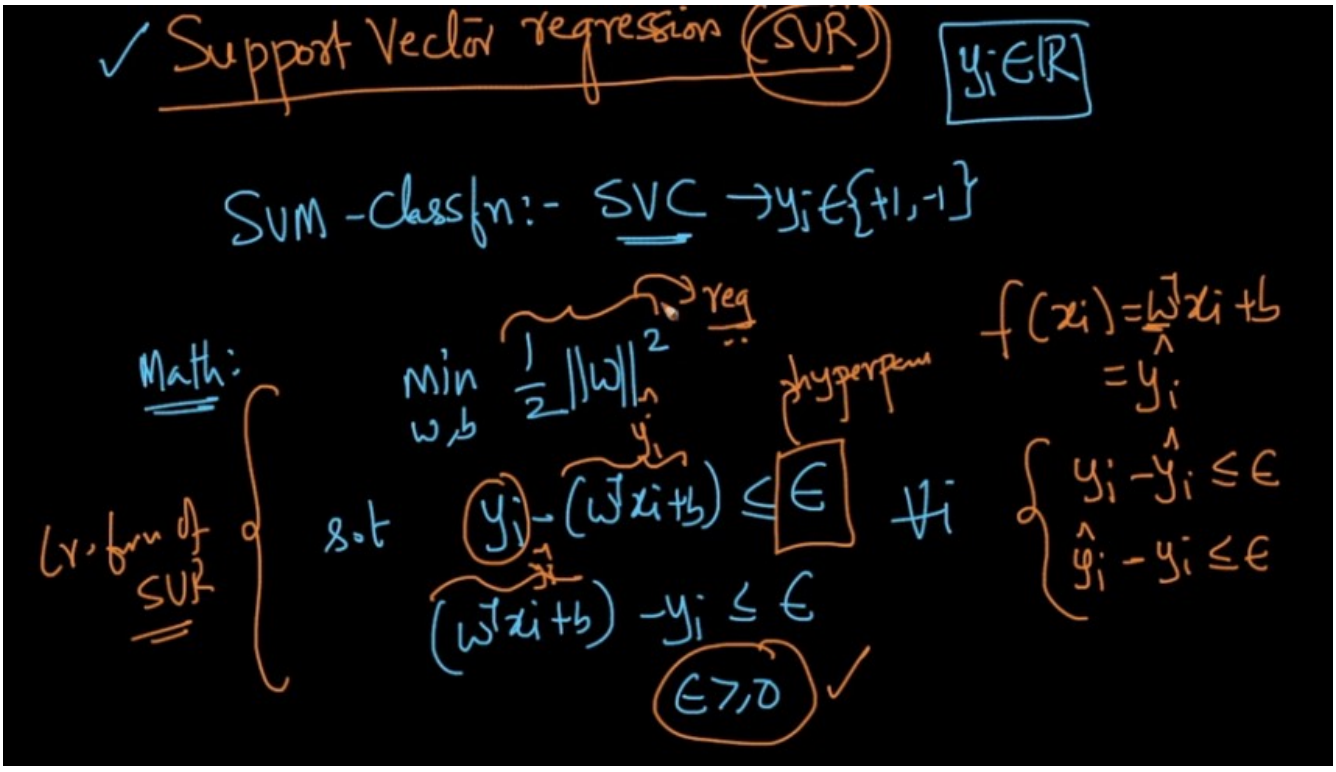




nu is the value of the error made by the SVM.

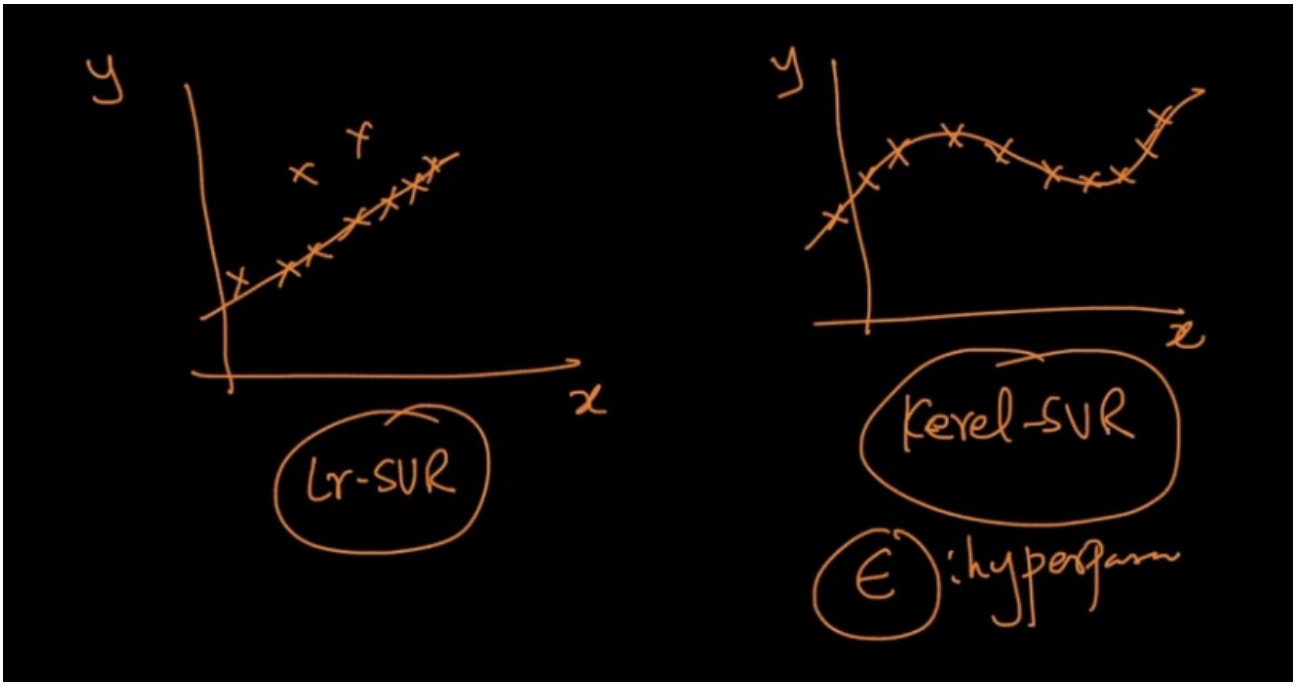


SVM regression:



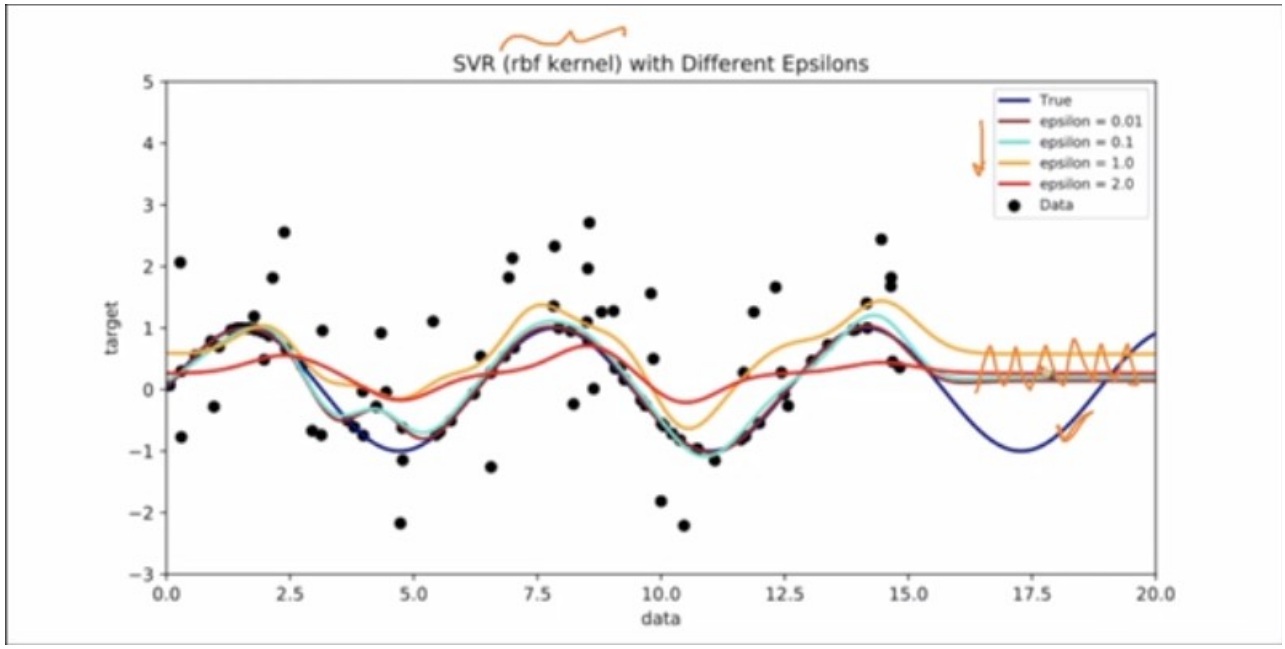
Here, we obtain the difference between the yi and yi^ and made to less than or equal to epsilon, where epsilon is a hyper parameter.

Where the ||w||^2 can be kernalized.



If epsilon is low then the error are low on the training data, over fitting will increase.
If epsilon in high then errors on training data will increase, that means under fit the data.

RBS represents the KNN-reg roughly.



As epsilon is very less then the likelihood of making mistakes decreases, then the model will over fit the data, choosing the right value of epsilon helps in making the model to generalize well.

Cases:

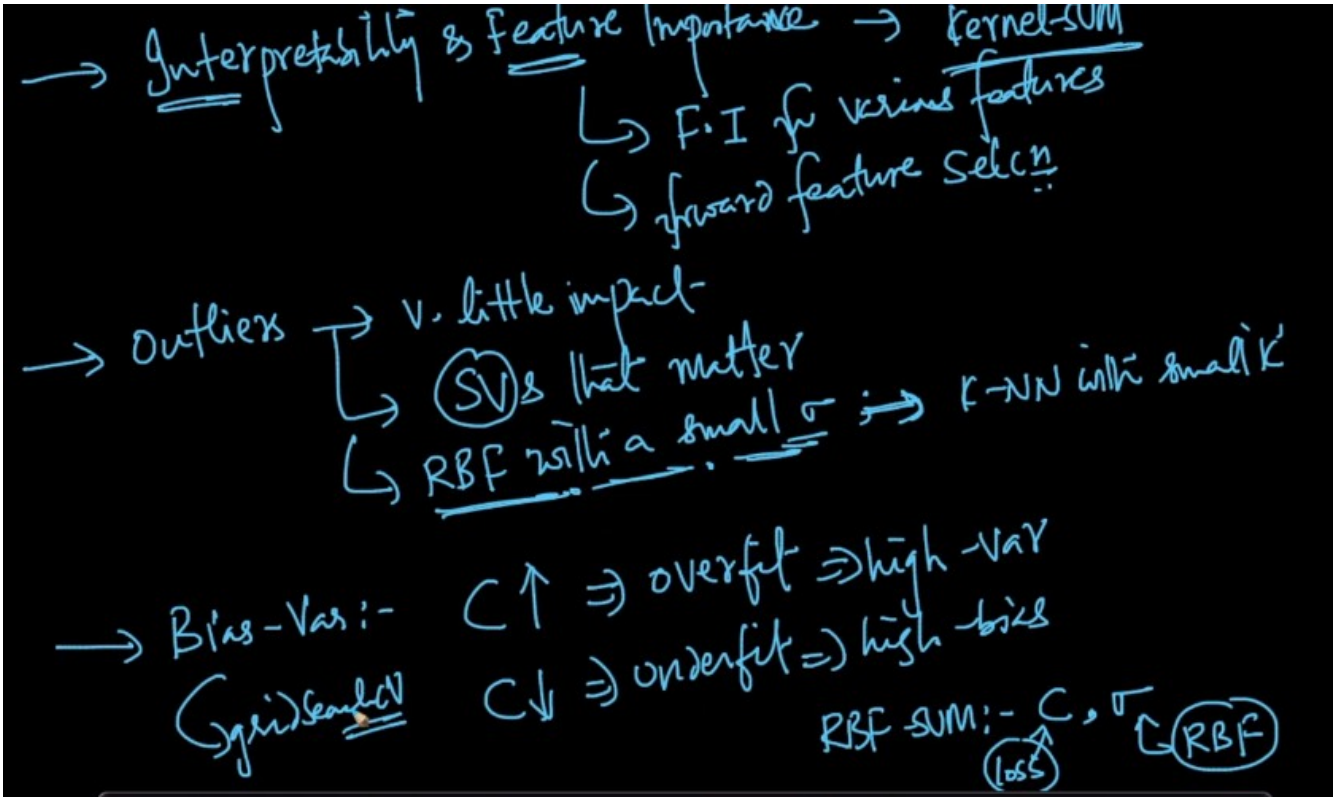
feature engineering: Kernel design, finding the right kernel.
Decision surface: Lr sum – Hyperplane.
Kernel - sum

Cases in SVM:

SVM's can work with similarity matrix.

Interpretability: For kernel SVM's we cannot get the feature importance.

Outliers: Outliers have little impact because we only consider the support vectors, RBF kernels get influenced by outliers, because it behaves as KNN.



The best case for SVM is when we have right kernel.