SVM [Support Vector Machine]

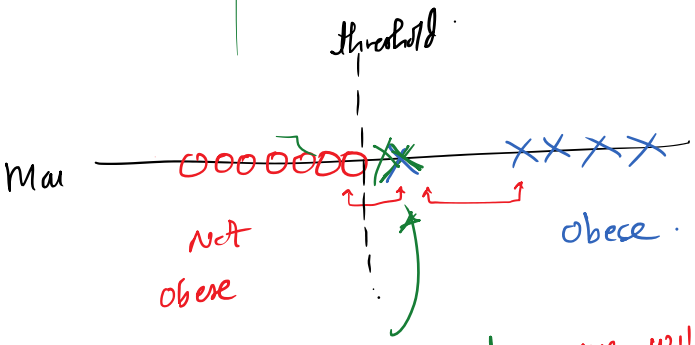
Consider → Mass of Bunch of mice →

threshold (randomly)

If mass of mice < threshold then declare → not obese

" " " > threshold " " → obese.

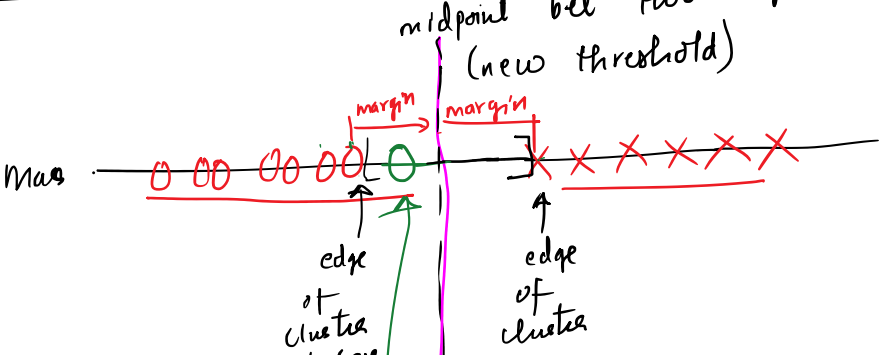
But Consider

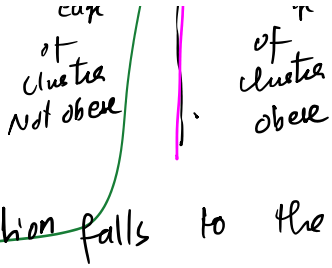


For this observation we will classify it as obese but it is actually closer to observations that are not obese

So the Threshold concept is preety lame.

We can do better on the original dataset when we focus on the edges of each cluster.





Here if any New observation falls to the left of the midpoint then classify it as not obese

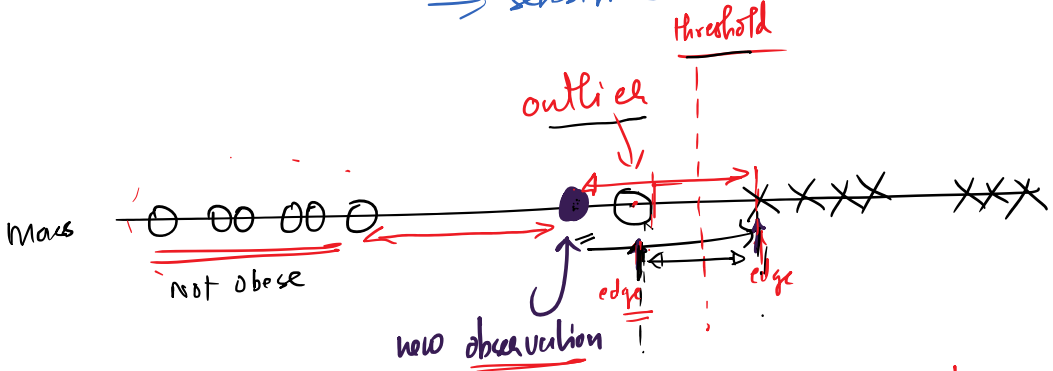
Margin → The shortest distance bet" the observations and the threshold is called margin.

Here Since threshold is midpoint the margin is same on both side.

When the threshold is Midway bet" the observations (edge observation of both cluster) then the margin is as large as it can be

This is known as Maximal Margin Classifier.

Consider the Training Data (Here Misclassification Not Allowed → Sensitive to Outliers → Hard Margin)

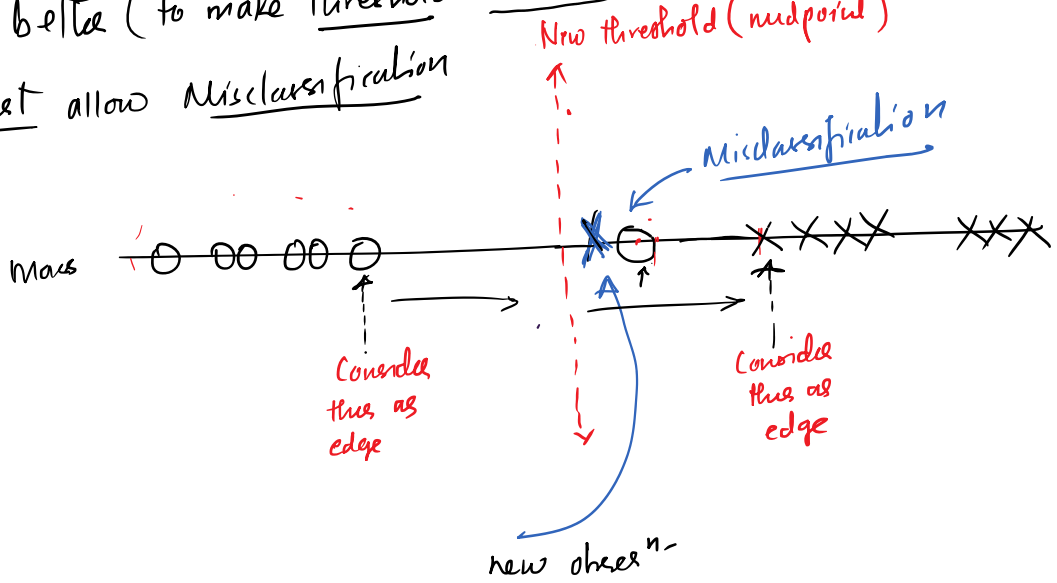


Here for New observation it will be classified as Not obese but we can see, it is closer to obese observations.

So Maximum Margin Classifier are super Sensitive to

outliers in Training Data and it makes it pretty ...

* We can do better (to make threshold Insensitive to outliers)
For this we must allow Misclassification



Now for the new observation it will be classified as Obese

* When we pick a threshold that was Sensitive to outliers.
we say it has low bias
and it gives Error for new observation so we say
it has high Variance

* When we select a threshold that allow Misclassification.
then we say it has high Bias but correctly classifies
New observation so it has low Variance

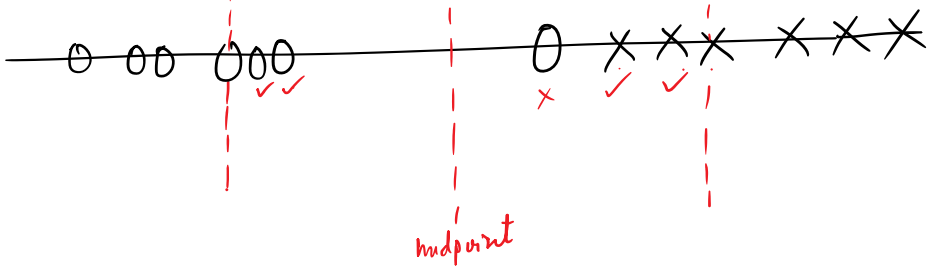
Soft Margin → When Misclassification is allowed then the distance bet^n threshold and observation is known as Soft Margin

We can ... named Misclassification Allowed, then How to know

known ... 0

We can have many Misclassification Allowed, then how to know which soft margin is better

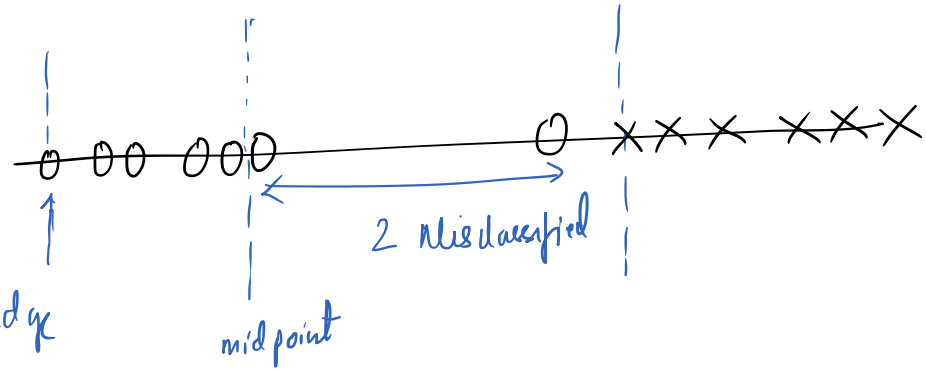
Consider the soft margin.



midpoint

4 are correctly classified
1 is Incorrectly classified.

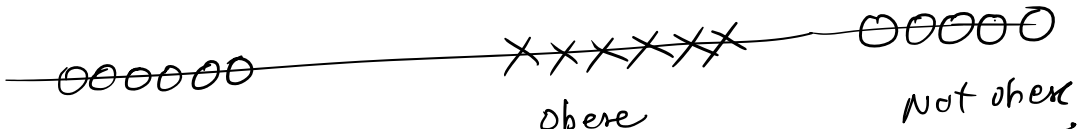
Also For this Soft Margin

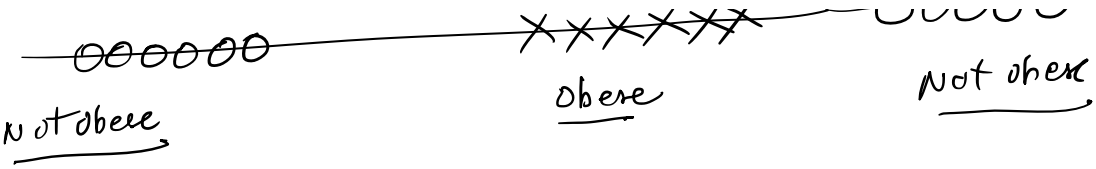


* We will determine which soft margin is better by Observing how many Misclassification each soft margin Allows.

Imp
* When we use Soft Margin to determine the location of threshold then we are Using "Soft Margin Classifier" Support Vector Classifier.

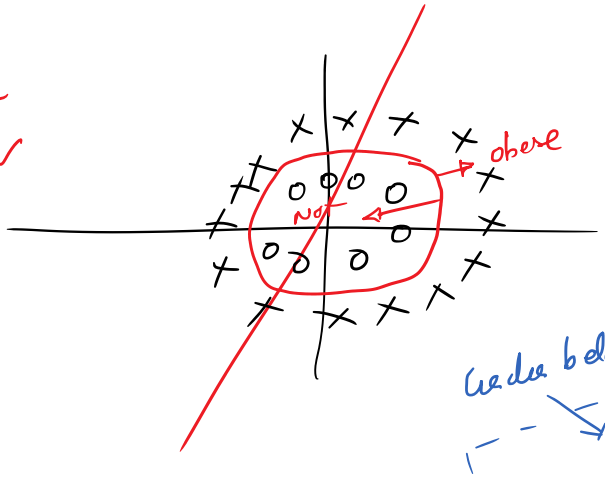
But if the Dataset is as.

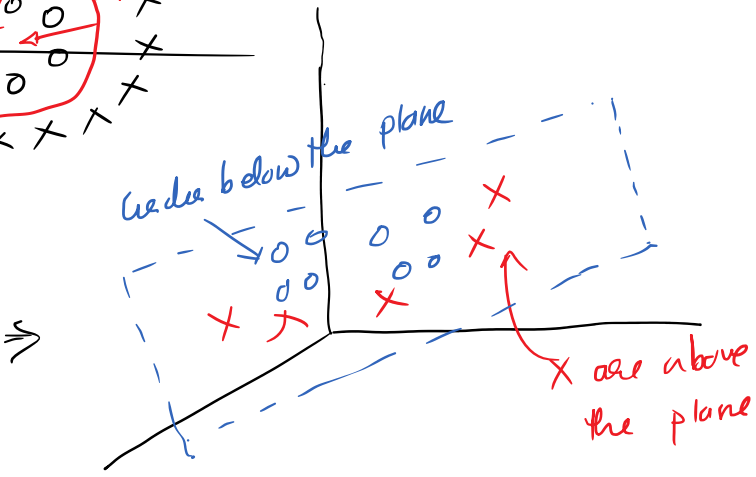




Here Soft Margin will not be efficient as the overlapping is very high.

We cannot have linear Separator Here





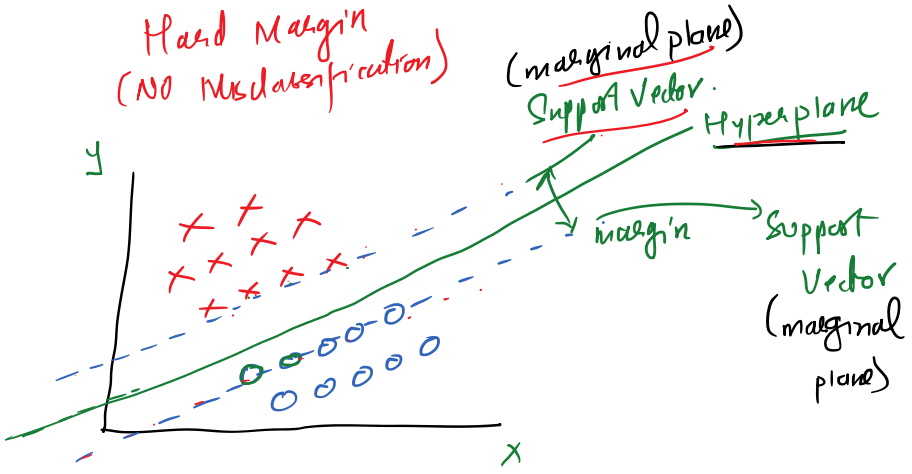
SVM → ML Algorithm

Task → Classification
Task → Regression.

Support Vector Classifier.

Consider Binary Classification

2-D Data



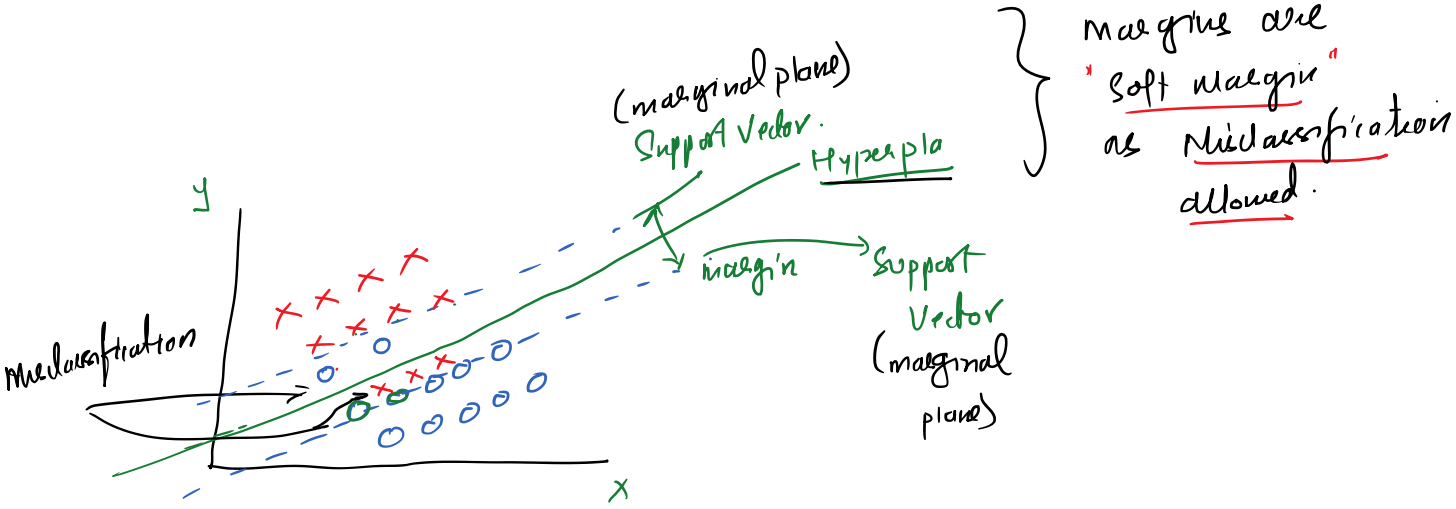
In logistic/linear ⇒ Single line (Hyperplane)

SVM ⇒ ① Support Vectors (Marginal plane)
② Hyperplane

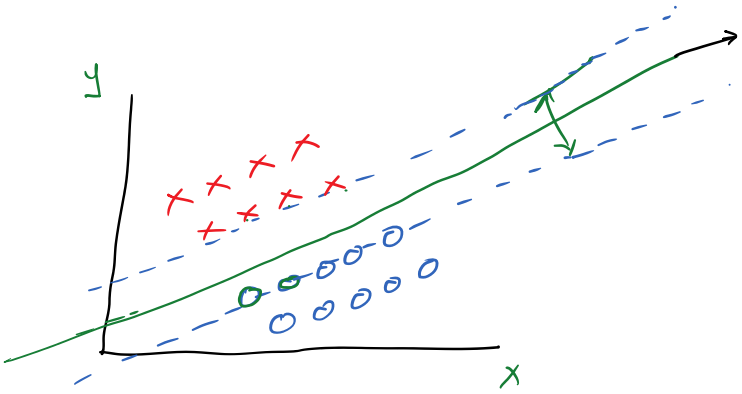
AIM → Create a Hyperplane and marginal planes such that the margin must be maximum.

* If we get perfect marginal plane and Hyperplane that can linearly separate the Data points then we call it as "Hard Marginal Plane" [No Error / Misclassification]

* If there are Misclassification allowed then the planes (marginal) will be known as "Soft marginal plane"



margins are "soft margin" as Misclassification allowed.



Note: Hyperplane → straight line

Eq → $y = c + mx$

OR $y = a_0 + a_1 x$

OR $y = b + w_1 x_1$

} Linear

Multi Linear Regression → Many features $x_1, x_2, x_3 \ldots$

each which weight w_1, w_2, w_3

$$y = w_1 x_1 + w_2 x_2 + \ldots\ldots w_n x_n + b.$$

$$\boxed{y = w^T x + b}$$

w^T ⇒ weight matrix

x = feature matrix

Normally Eq^n of line $y = mx + c$; — 1

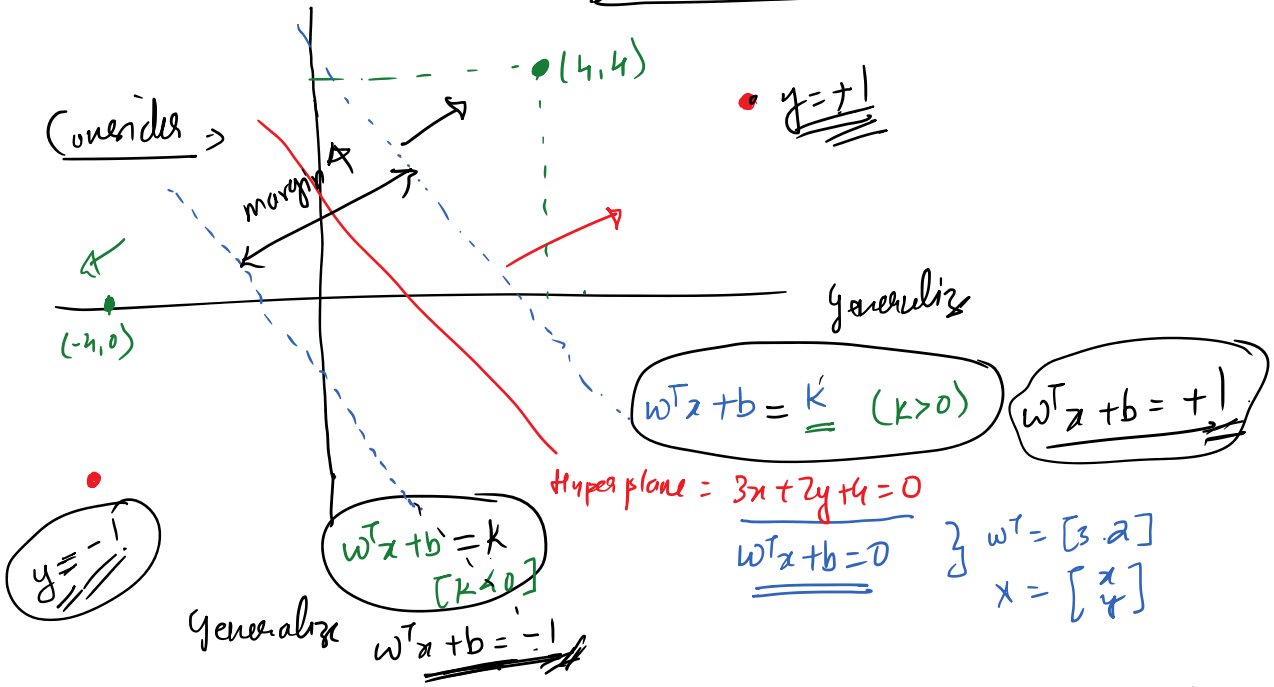
we have $ax + by + c = 0$; — 2

write (2) in form of (1)

$by = -ax - c$;

$$y = \left(\frac{-a}{b}\right)x - \left(\frac{c}{b}\right) \quad — (3)$$

from (1) & (3) $\quad m = -\frac{a}{b}, \quad c = -c/b$



Consider the above hyperplane Eq^n $3x + 2y + 4 = 0$ ⟶ in the form $w^Tx + b = 0$

For the point $(-4, 0)$ ⇒ substitute in Eq^n $\quad 3(-4) + 2(0) + 4$

$= -12 + 0 + 4 = -8 < 0$

For the point $(4, 4)$ ⇒ Substitute in Eq^n $\quad 3(4) + 2(4) + 4$

$= 12 + 8 + 4 = 24 > 0$

For any point lying above the Eq^n $w^T x + b = +1$ will give +ve value

Similarly any point lying below the Eq $w^T x + b = -1$ will give -ve value.

The Hyperplane is represented as $\Rightarrow$ $w^T x + b = 0$

Our Aim to draw two marginal planes (+ve & -ve side) and need to ensure the distance is maximum.

* The Eq^n of marginal plane on +ve side $w^T x + b = +1$
* the Eq^n of " " " -ve side $w^T x + b = -1$

* we want the distance bet^n them $\Rightarrow$

lets find diff $\Rightarrow$

$$w^T x_1 + b = +1$$
$$- \quad w^T x_2 + b = -1$$
$$+ \quad - \quad +$$
$$w^T (x_1 - x_2) = 2 \quad \text{---------- (4)}$$

w = coefficient = slope $\Rightarrow$ magitude $\|w\|$, Vector $\vec{w}$

we have to focus on Vector.

From w to get $\vec{w}$ $\Rightarrow \frac{w}{\|w\|} \Rightarrow \vec{w}$

From (4) $\quad w^T (x_1 - x_2) = 2$

To Convert into Vector Divide both side by $\|w\|$

$$\therefore \left(\frac{w^T}{\|w\|}\right)(x_1 - x_2) = \left(\frac{2}{\|w\|}\right)$$

$$\therefore \left(\frac{w^T}{\|w\|}\right)(x_1 - x_2) = \left(\frac{2}{\|w\|}\right)$$

difference/margin.

$$\vec{w}(x_1 - x_2) = \frac{2}{\|w\|}$$

We Need to Maximize Margin ⇒

$w^T x + b = 0$

AIM = Maximize$_{(w,b)}$ $= \frac{2}{\|w\|}$. By changing w & b

Constraint :

Such that

y_i (prediction)

$+1$ when $w^T x + b \geq 1$

when point lies outside $w^T x + b = +1$ + margin plane

then classification for the point ⇒ $+1$

-1 when $w^T x + b \leq -1$

i.e the point lies below the line $w^T x + b = -1$ -ve margin plane then y_i will be classified as -1

The above constraints are for Correctly classified Data points

∴ For all accurate/correctly classified data points

Final constraints wrt support vector classifier = $y_i \times (w^T x_i + b) \geq 1$

(y_i: observed, x_i: predicted)

Observed predicted

Final Cost Fn → This is Max Margin

Maximize $_{(w,b)}$ = $\frac{2}{\|w\|}$ subjected to constraint $y_i * (w^T x_i + b) \geq 1$

Can be also written as Since cost fn we express in terms of Minimize distance bet^n margin will increase

Minimize = $\frac{\|w\|}{2}$ subjed to $y_i * (w^T x_i + b) \geq 1$

But above is for correctly classified Points (No Misclassfn)

But In real world there is always Misclassification

we we must have about by using Hyperparameter

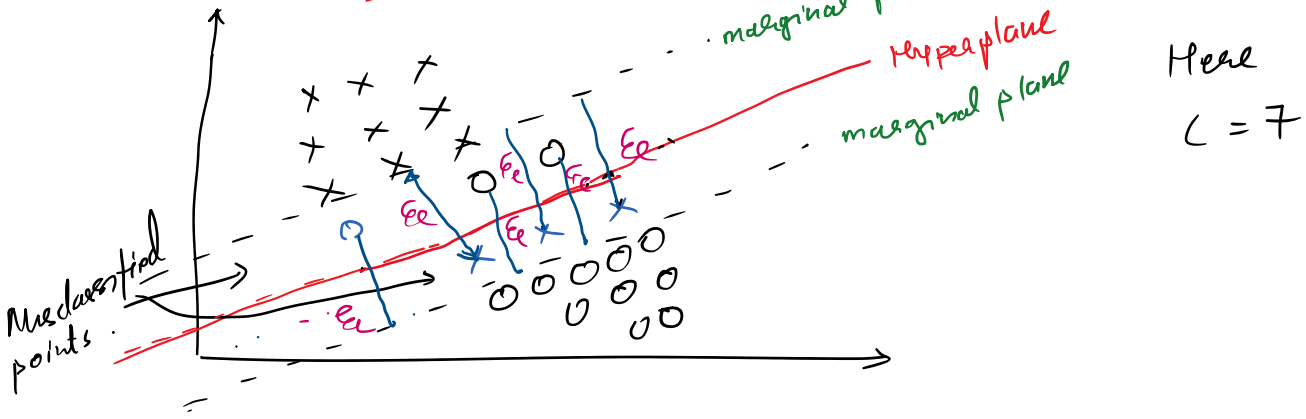
∴ Final cost Fn (For All Data points (correct/Incorrect classified).

Minimize $_{(w,b)}$ $\frac{\|w\|}{b} + C_i \sum_{i=1}^{C_i} \varepsilon_i$ Eta.

C_i = No of misclassified

Hyperparameters.

C_i = How many points we can allow Misclassified.



C_i ⇒ It decreases Overfitting problem by allowing Misclassification.

$\sum \varepsilon_i$ ⇒ Sum of distance of Misclassified points from corresponding margin.

misclassified

$\Sigma \xi_i$ ⇒ Sum of distance of misclassified points

margin.

∴ The Cost F^n for SVC is

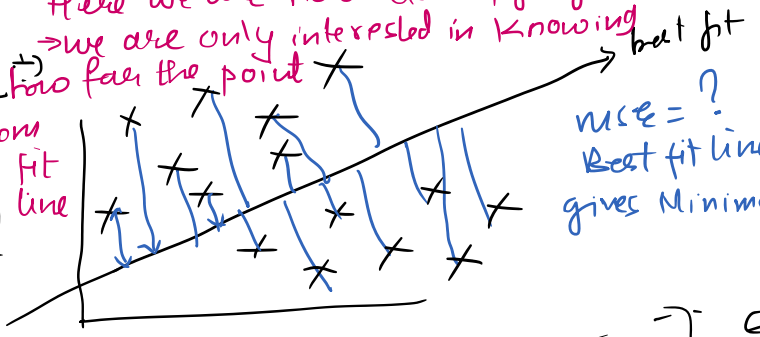
$$\text{Minimize}_{(w,b)} \quad \frac{\|w\|}{2} + C \sum_{i=1}^{n} |\xi_i|$$

//

Here Support Vector used for Classifi^n.

Support Vector for Regression →

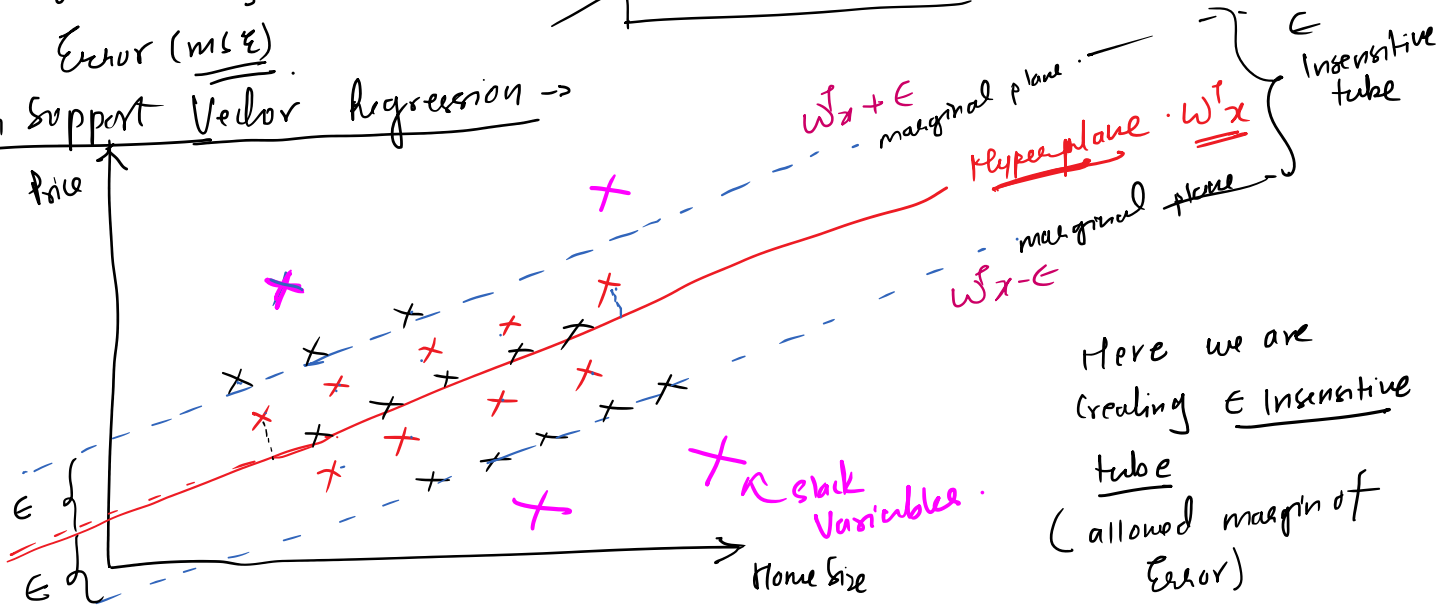
Here we are not classifying
→ we are only interested in knowing how far the point is from best fit line



* In linear Regression - best fit line we minimize Mean Squared Error (MSE)

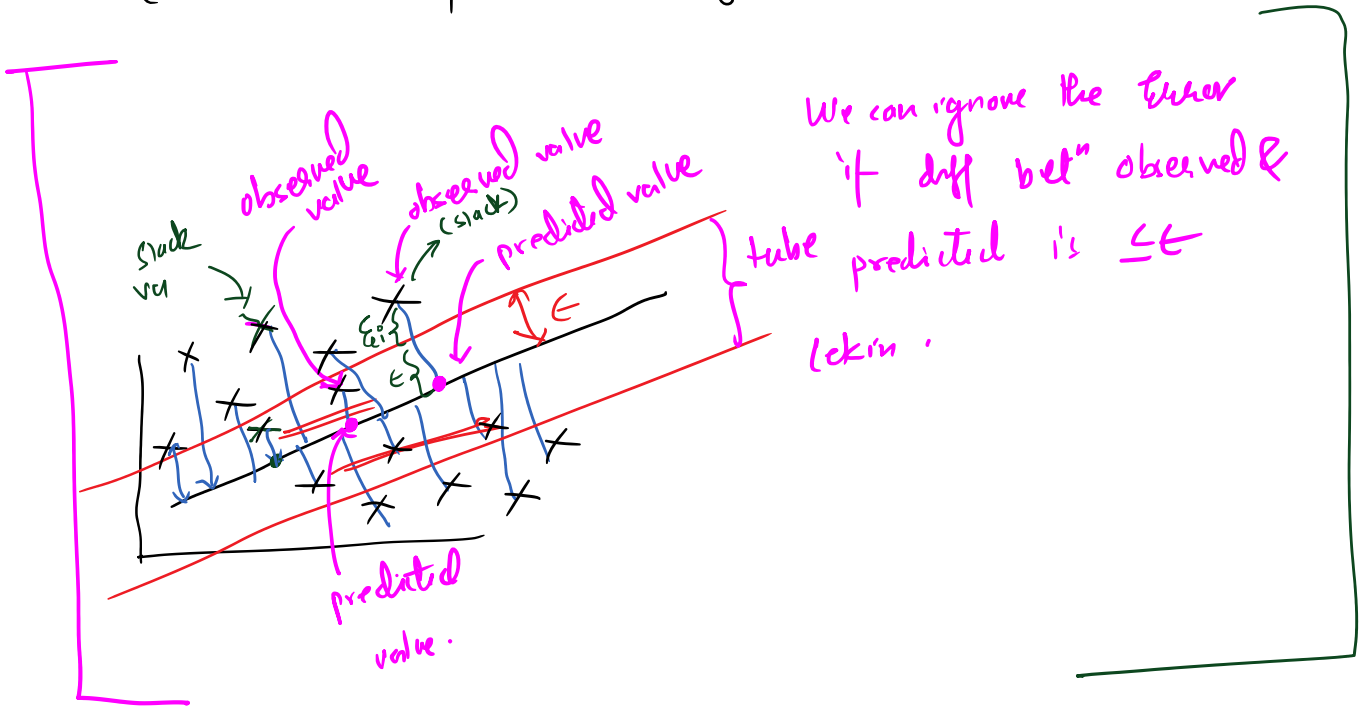
MSE = ?
Best fit line that gives Minimum MSE

In Support Vector Regression →



Here we are creating ϵ Insensitive tube (allowed margin of Error)

All the points inside ϵ Insensitive tube are not Considered for Calculating Error



We can ignore the Error if diff bet" observed & predicted is $\leq \epsilon$ lekin.

For a point x_i within ϵ Insensitive tube

$|y_i - w^T x_i| \leq \epsilon$

for a point x_i ...

$|y_i^o - w^T x_i^o| \leq \epsilon$

y_i^o = observed value

$w^T x_i^o$ = predicted value.

Constraint if $|y_i^o - w^T x_i^o| \leq \epsilon$ then we can say the prediction is good &. we will not consider the difference in error calculation.

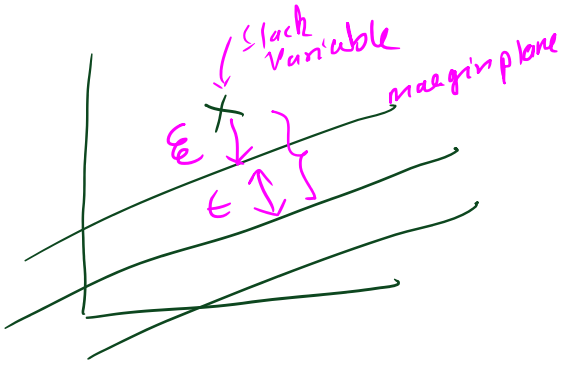
There are points outside marginal plane (slack Variables)

∴ we need to calculate distance (error) of slack Variables (observed value y_i^o) from Hyperplane (predicted value)

$= \epsilon + \xi_i^o$

ϵ: distance bet^n marginal plane & Hyperplane

ξ_i^o: distance bet^n marginal plane & slack Variable.



Cost J^n = Minimize$_{(w,b)}$ $\frac{\|w\|}{2} + C_i^o * \sum_{i=1}^{\epsilon} \xi_i^o$ — Hinge Loss.

Constraint: $|y_i^o - w_i x_i^o| \leq \epsilon + |\xi_i|$

i.e distance bet^n observed & predicted value must be

$\leq \epsilon + \xi_i^o$

* As Compared to linear Regression, the sum is allowing some margin (ϵ) for error as buffer

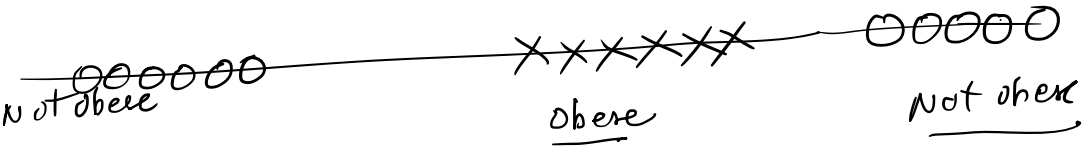
margin (ϵ) for error as buffer

* Here basically the error is distance bet^n slack variable (variable outside the tube) and the tube (ϵ)

ξ_i

SVM (Support Vector Machine) →

if the Dataset is as.

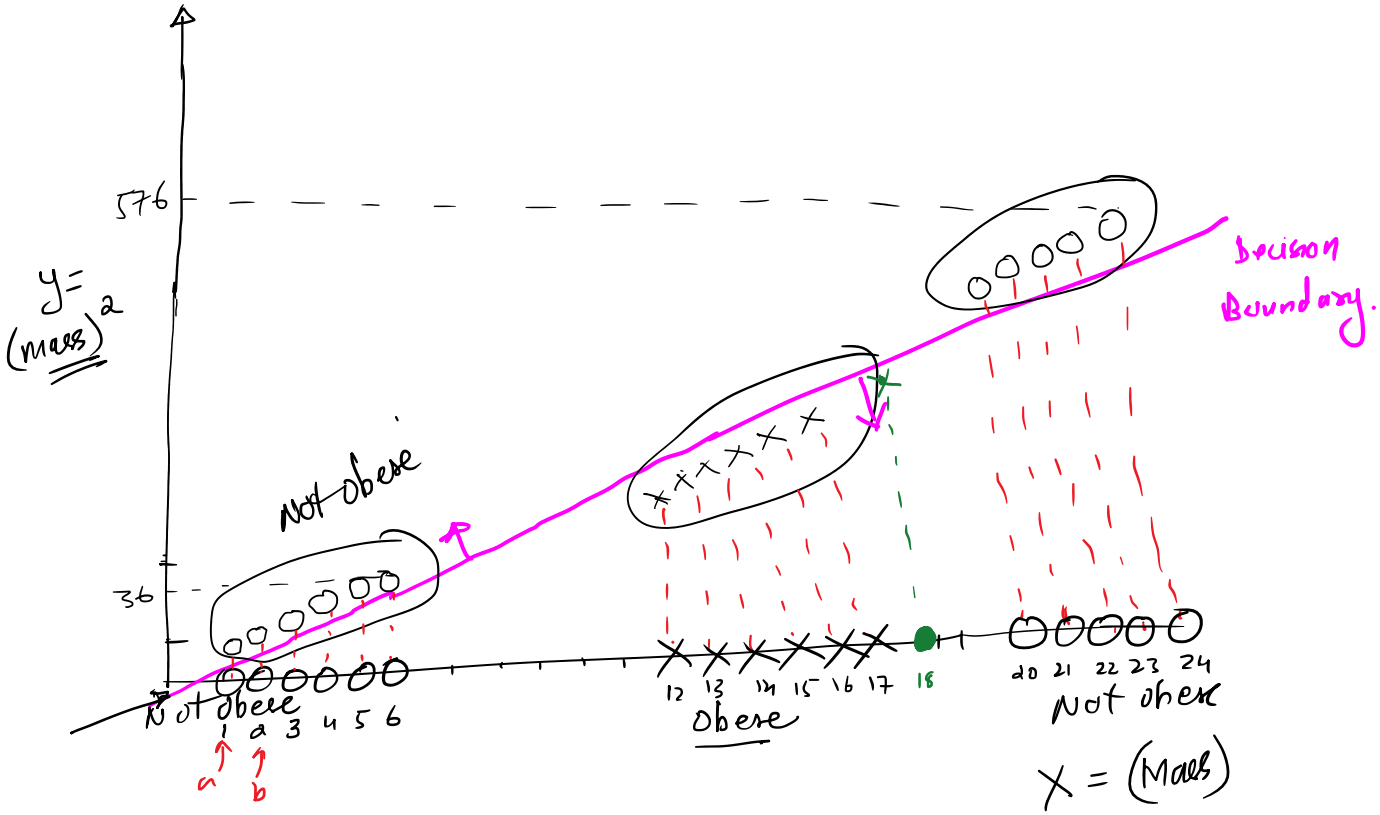


Here Soft Margin will not be efficient as the overlapping is very high.

* Whenever we have lot of Misclassification (Overlapping)

then Max Margin Classifier and Support Vector Classifier cannot be used.

In such case we will use Support Vector Machine.



Here x = Mass

$y = (\text{Mass})^2$

from above By increasing dimension of data from 1D to 2D by using Square (Here) function we can linearly separate the data points.

We can also use it to predict for new observation.
→ For new observation take square of observation.

SVM (Main Idea)

* Start data with low Dimension (given)
* Now move data to Higher Dimension
* Find SV classifier to classify the data into diff classes.

Question Arises → Why Square why not Cube?
ie How to decide, how to Transform the data.

[SVM uses Kernel function to Systematically find SV classifier in Higher Dimension]

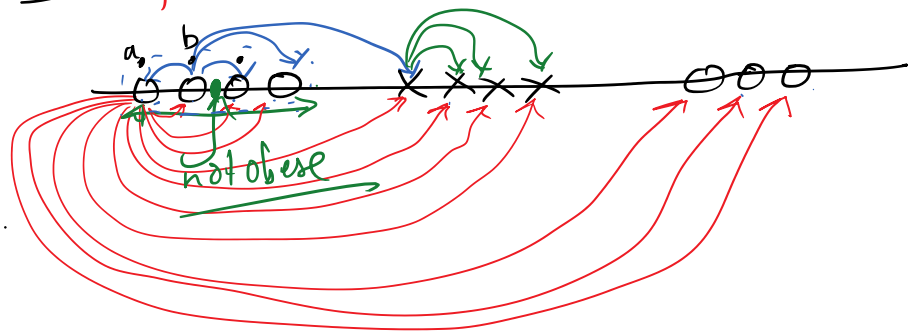
* Imp
* Kernel function actually never does any transformation in Higher Dimension.
* Instead it calculate Relation bet^n observations and visualizes them in Higher Dimension. SVC uses this Visualizn for classification.
* This is Known as Kernel Trick

* Let us use Polynomial Kernel that has parameter 'd' ie d is for degree of Polynomial.

↑ Let us use [illegible]

Skands for degree of Polynomial.

if d=1 → the Polynomial Kernel computes relationship bet" each pair of observation in 1D



Here All points with All other points relation is used.

Here d = best found by Cross Validation.

* We have many Kernel fn

Most Commonly Used are →

① Polynomial Kernel ✓

② Radial Basis Fn ✓

Polynomial Kernel → It calculate Higher Dimensional Relationship.

The kernel that transforms 1D to 2D is Polynomial Kernel

It may look like $(a * b + r)^d$

* a and b are two diff observations in dataset (any two points in dataset).
* r = coefficient of Polynomial.
* d = degree of Polynomial.

Find relationship bet^n observation in higher dimension.

[Here we use SVM with Polynomial Kernel to compute Relationship bet^h observations in Higher Dimension and then find Good Classifier]

Let $r = 1/2$ $d = 2$ } value of r and d is determined by cross validation.

$$(a * b + 1/2)^2 = (a * b + 1/2) \cdot (a * b + 1/2)$$

$$= a^2 b^2 + \frac{1}{2} ab + \frac{1}{2} ab + 1/4.$$

$$= a^2 b^2 + ab + 1/4.$$

$$= (a, a^2, 1/2) \cdot (b, b^2, 1/2)$$

This polynomial is equal to this Dot product.

First observation x-axis values

Second observ^n. x axis value

higher dimension Visual^n y-axis value.

[Dot product is Sum of 1st term Multiplied, 2nd term Multiplied and so on]

* The Dot product gives us high Dimensional coordinates from Data.

Here $1/2$ is third axis but values are same so can ignore

Here $(a+b\times\gamma)^d$ is used to get higher Dimension Rel^n of two data points a & b.

Note

Since $= (a*b+1/2)^2 = (a, a^2, 1/2)\cdot(b, b^2, 1/2)$

$\Rightarrow$ We can observe that all we need to do calculate the higher dimensional relationship is to calculate dot products bet^n each pair of points.

when $\gamma=1$, $d=2$

$(a*b+\gamma)^d = (a*b+1)^2$

$= a^2b^2 + 2ab + 1$

$= 2ab + a^2b^2 + 1$

$= (\sqrt{2}a, a^2, 1)\cdot(\sqrt{2}b, b^2, 1)$

Now

for these two points, we want to find high dimensional relationship bet^n these two points

now $(a*b+1/2)^2 = (9\times14+1/2)^2 = 126.5^2 = \underline{16002.25}$

(bcoz we have to do this for all data points pair)

is the one 2-D relationship we need to solve for SV classifier even though we already did not transform data in 2-Dimension.

* Once we decide value of γ and d using Cross Validation we can find higher dimensional relationship bet^n every pair of data points.

pair of data points.

Radial Kernel → Works in infinite Dimension.

Question ↳ What are the parameters?

↳ How it calculates the relationship in Higher Dimension.

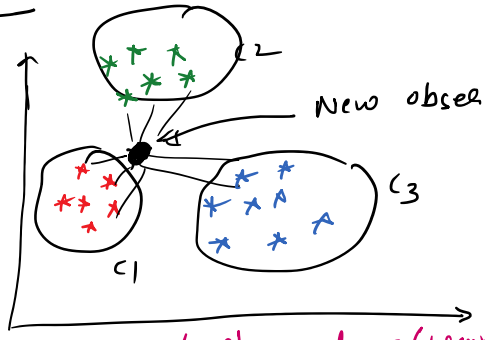
↳ It visualizes relationship of observations in Infinite Dimension.

* One way to deal with overlapping data is to use SVM with Radial Kernel.

$e^{-\gamma(a-b)^2}$ } Radial Basis f^n (RBF)

→ Since RBF finds SV classifier in Infinite Dimension it is not possible to visualize what it does.

→ But when applied to New observation. it behaves like weighted Nearest Neighbour model



* Radial Kernel checks influence of the Nearest Neighbours in New observations and accordingly makes classification.

[* closest observation (nearest Neighbour will have lot of influence on how we classify new observations as compared to far off observations]

* Let's talk about how radial Kernel determines how much influence each observation in Training Dataset has on classifying new observations.

$e^{-\gamma(a-b)^2}$ Here a and b are two observations. (let say dosages)

$\Rightarrow (a-b)^2$ ⇒ diff b/w the measurement is squared giving us the squared distance b/w the two observations.

but say a and b are

let say a and b are different data points. (different dosages)

two observations.

* So the amount of influence that one observation has on another is function of $(a-b)^2$. [squared distance].

Ex

(effective) xxxx 0000 dosage very high not effective

0000 dosage very less not effective

γ = determined by Cross validation [It scales the distance hence influence].

$= \frac{1}{\text{no of features} * \sigma^2}$

= Scaler that defines how much influence a single training example (point) has.

The observation a = 2.5 and b = 4 are closest to the new observation [These observations are relatively near]

Radial Kernel uses this classification for new observation

let $\gamma = 1$ ← γ scales the distance

$e^{-1(2.5-4)^2} = 0.11$ — (1)

Here low scale, low distance, overall high influence

When γ is ↑ it scales (multiplies) distance bet^n a & b. So effectively effects the amount of influence the two points have on each other.

let $\gamma = 2$

$e^{-2(2.5-4)^2} = 0.01$

[0.11 is high dimensional relationship bet^n observations]

If two observation are a = 2.5 b = 16 ← Here the observation are relatively far from each other.

let $\gamma = 1$ $e^{-1(2.5-16)^2} = e^{-182.25}$ — (2)

= ≈ close to zero

[Here relationship is closed to zero as the points are far from each other].

From (1) & (2)

If points are close enough then we have high influence

and if points are far off then we have low

Just like Polynomial Kernel when we plug the values in Radial Kernel we get High Dimensional Relationship.

Note →

To understand How Radial Basis Kernel works in Infinite Dimensions:

Let us take Polynomial Kernel →

$(a*b+r_1)^d$ = $r_1 = 0$, $d = 2$ ⇒ $(a*b)^d = (a)^d \cdot (b)^d$

⇒ So now coordinate is only the Square of orginal measurement on orginal axis [No new axis generated].

when $a = 2.5$, $b = 4$ → $a^2 = 6.25$, $b^2 = 16$.

When $r = 0$, $d = 3$ ⇒ $a^3 \cdot b^3$.

$r = 0$, $d = 1$ $a \cdot b$.

$r = 0$, $d = 2$ $a^2 \cdot b^2$

When $r_1 = 0$ leaves data in same 1 dimensional irrespective of values of d.

To add $d = 1, d = 2$ ⇒ $a^1 \cdot b^1 + a^2 \cdot b^2$ ⇒ $(a \cdot a^2) \cdot (b \cdot b^2)$

Thus it 2-D Transformation.

Here actually no transformation was done.

* Just Solved the dot product to get High Dimensional

* Just some relationship.

Lets do for ∞ dimension ⇒

$a^1b^1 + a^2b^2 + \ldots + a^\infty b^\infty \Rightarrow (a^1, a^2, a^3 \ldots a^\infty) \cdot (b^1, b^2, \ldots b^\infty)$

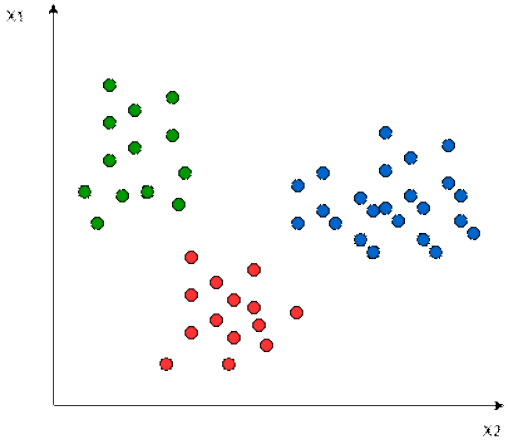
This gives us Dot product with Cordinate for Infinite no of Dimensions.

* This is what Radial Kernel Does

Multiclass Classification Using SVM

- In its most simple type, SVM doesn't support multiclass classification natively.
- It supports binary classification and separating data points into two classes.
- For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.
- The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes.
- **This is called a *One-to-One* approach, which breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of classes.**
- **Another approach one can use is *One-to-Rest*. In that approach, the breakdown is set to a binary classifier per each class.**
- A single SVM does binary classification and can differentiate between two classes.
- So that, according to the two breakdown approaches, to classify data points from classes data set:
- In the *One-to-Rest* approach, the classifier can use SVMs. Each SVM would predict membership in one of the classes.
- In the *One-to-One* approach, the classifier can use SVMs.

Let's take an example of 3 classes classification problem; green, red, and blue, as the following image:

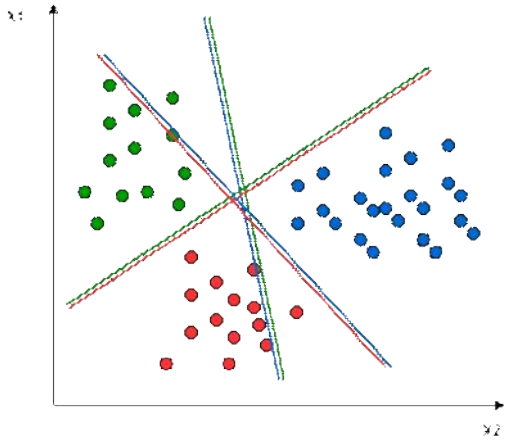


Applying the two approaches to this data set results in the followings:

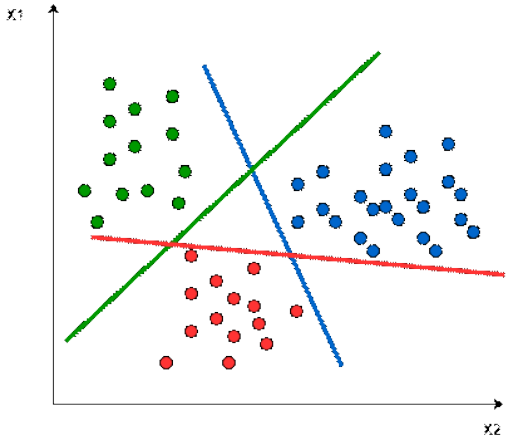
In the *One-to-One* approach, we need a hyperplane to separate between every two classes, neglecting the points of the third class.

This means the separation takes into account only the points of the two classes in the current split.

For example, the red-blue line tries to maximize the separation only between blue and red points. It has nothing to do with green points:



In the *One-to-Rest* approach, we need a hyperplane to separate between a class and all others at once. This means the separation takes all points into account, dividing them into two groups; a group for the class points and a group for all other points. For example, the green line tries to maximize the separation between green points and all other points at once:



One of the most common real-world problems for multiclass classification using SVM is text classification. For example, classifying news articles, tweets, or scientific papers.