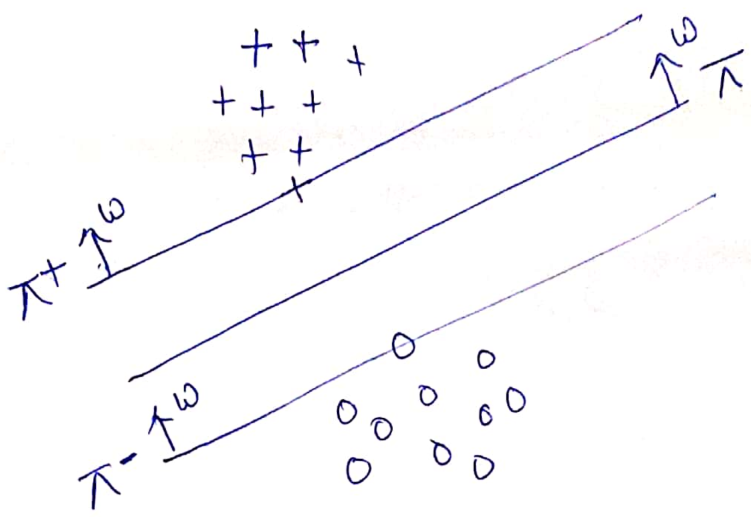
→ "Alternate Mathematical formulation of SVM"

We already know from geometric intuition of SVM, that, we want to find a hyperplane 'π' that does ["margin maximization"]



Suppose if 'π' is our best hyperplane, let's write hyperplane 'π' as w^Tx+b

$$\pi: w^Tx+b=0$$

Here "w" is $\perp$ to the hyperplane"

One thing that we quickly realize is, if π^+ is my positive hyperplane & π^- is my negative hyperplane & since 'w' is $\perp$ to 'π' & π^+, π^- & π are parallel to each other then "w" will also be $\perp$ to {'π^+' & 'π^-'}

Let's say "π^+" has this form.

$$\pi^+: w^Tx+b=1$$

& 'π^-' has this form

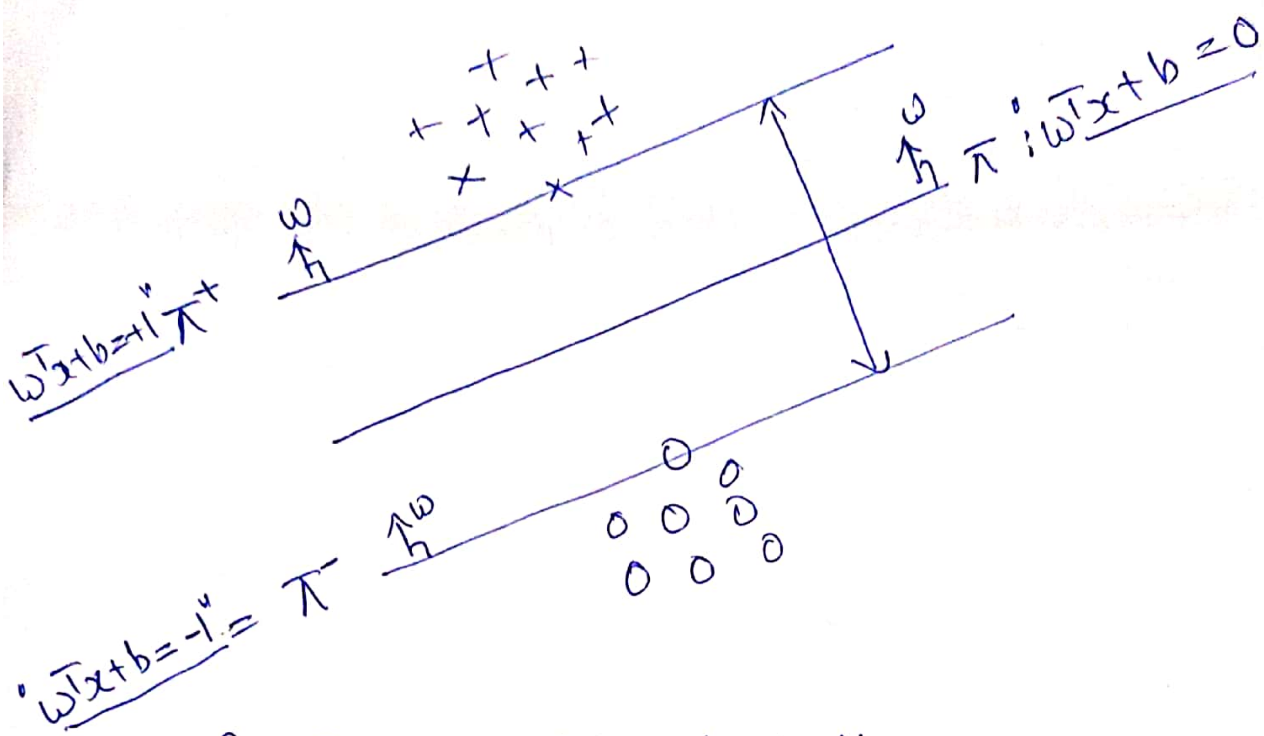
$$\pi^-: w^Tx+b=-1$$

Note :⇒ '$w^Tw \neq 1$'

['w'] is not a 'unit vector'

Scanned with CamScanner

let's assume that "w" is some vector & not necessarily a "unit vector" & it is $\perp$ to "π", "π^+" & "π^-".



In SVM we care about this margin.

The margin is $d = \left\{ \frac{2}{\|w\|} \right\}$ & "w" is some vector which {is $\perp$ to "π", "π^+" & "π^-"}

We want to find a "w^*" & "b^*" in such a way that the margin is maximized"

$$\left\{ (w^*, b^*) = \underset{w,b}{\operatorname{argmax}} \frac{2}{\|w\|} \right\}$$

This is what we want to find

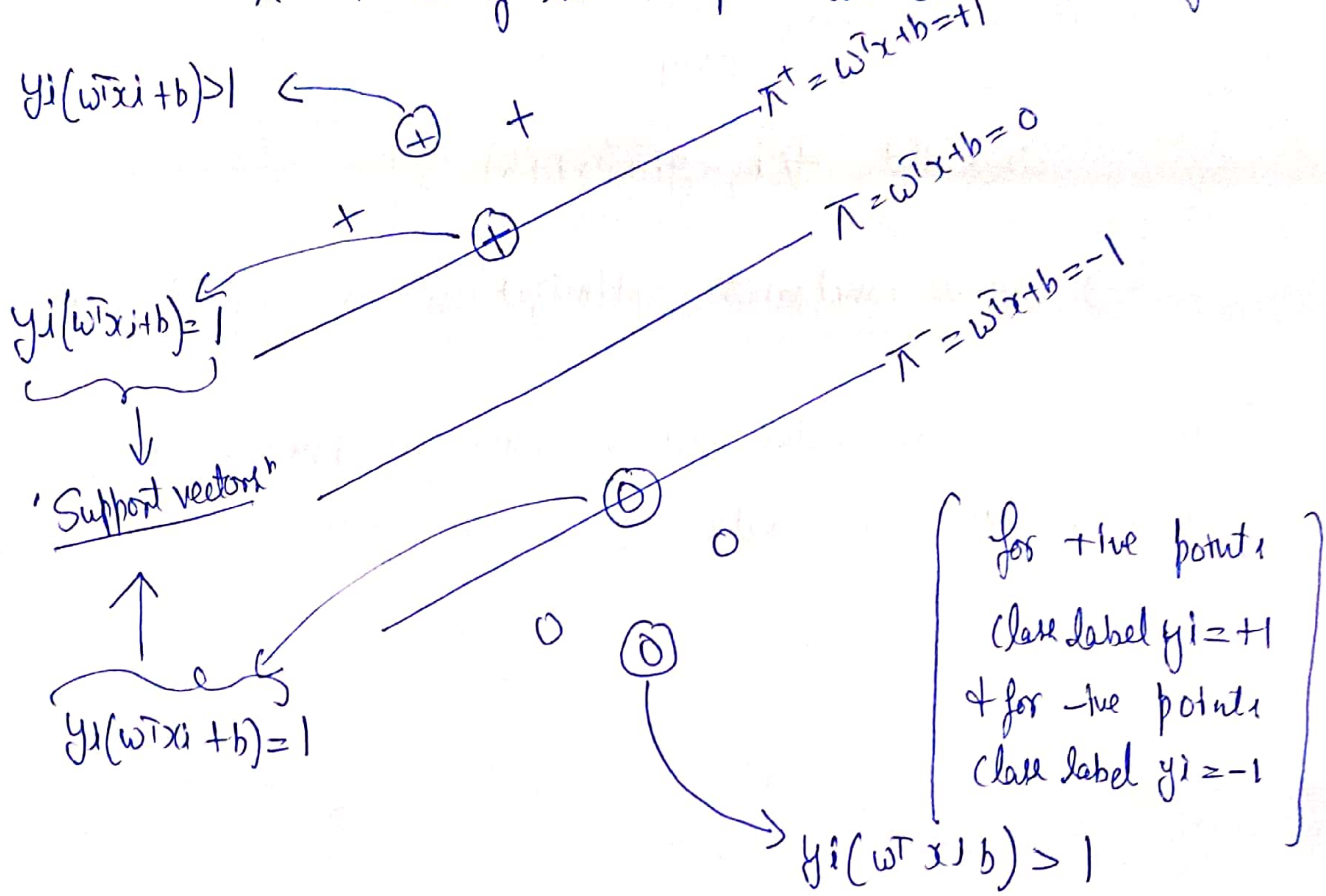
Constraint such that All the +ve points are on the side of "π^+" & all the -ve points are on the side of "π^-"

Scanned with CamScanner

There are some constraint with.

$$(w^*, b^*) = \underset{w,b}{\operatorname{argmax}} \ \frac{2}{\|w\|} \rightarrow \text{margin}$$

such that any of the '+ve point' is on the side of 'π^+' & all of the -ve points are on the side of 'π^-'



Now the constraint that we have is :-
& our optimization problem will eventually looks like.

$$\boxed{\begin{aligned} (w^*, b^*) &= \underset{w,b}{\operatorname{argmax}} \ \frac{2}{\|w\|} \rightarrow \text{margin} \\ \text{such that} \quad & y_i(w^Tx_i+b) \geq 1 \quad \text{for all } x_i\text{'s} \end{aligned}}$$

Note :- It is exactly equal to '1' for Support vectors & for Non-support vectors, it is greater than '1'

Scanned with CamScanner

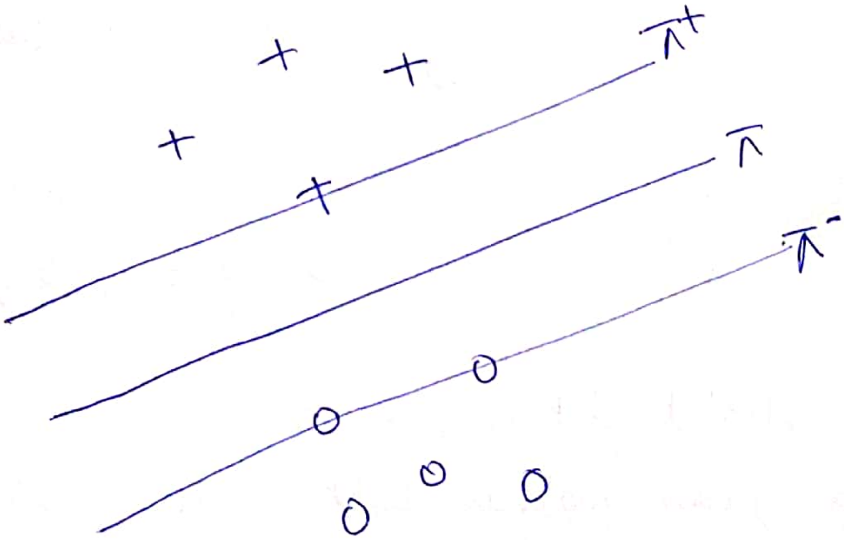
So, literally it is not one constraint, it is n constraints
becoz we have 'n' points in our training data

So, the final problem that we have is :-

$$\left\{ w^*, b^* = \underset{w, b}{\arg\max} \frac{2}{\|w\|} \quad \text{such that } \forall i,\ y_i(w^T x_i + b) \geq 1 \right\}$$

→ This is constraint optimization problem of SVM

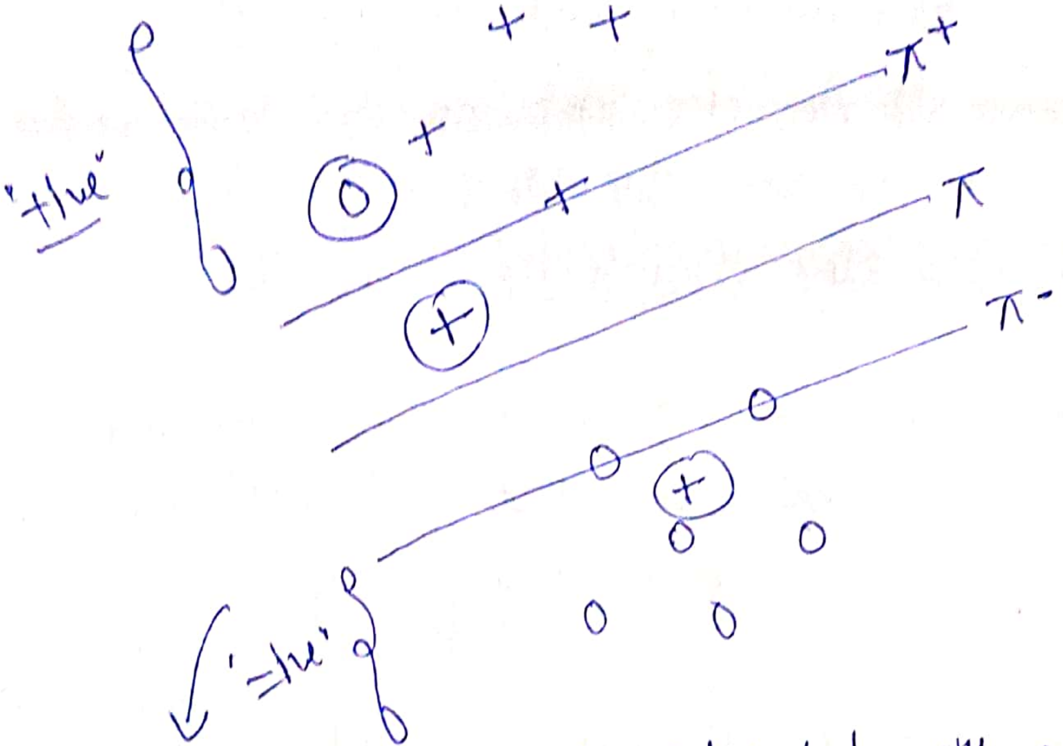
There is one fundamental problem with this formulation.
What if we have our data like shown below :-



→ This works when our data is linearly separable.
becoz the constraint that we have is, that every +ve point should be in the upper region of "π^+" & every negative point should be in the lower region of 'π^-'

There should be no +ive & -ive points in the opposite directions. Actually there should be no point between the planes also (in the margin area). (5)

Now what if we have a '-ive' point on the upper side of 'π⁺' or a '+ive' point on the lower side of 'π⁻'



This dataset can't be separated with a hyperplane

These three encircled points in above case will never satisfy the constraints

So, if we try to solve the problem (optimization problem) for a dataset like shown above which is not linearly separable but it is almost linearly separable except for just a few points, most of the points are ok.
In such a case you may never be able to find a 'w & b' that satisfy these condition.

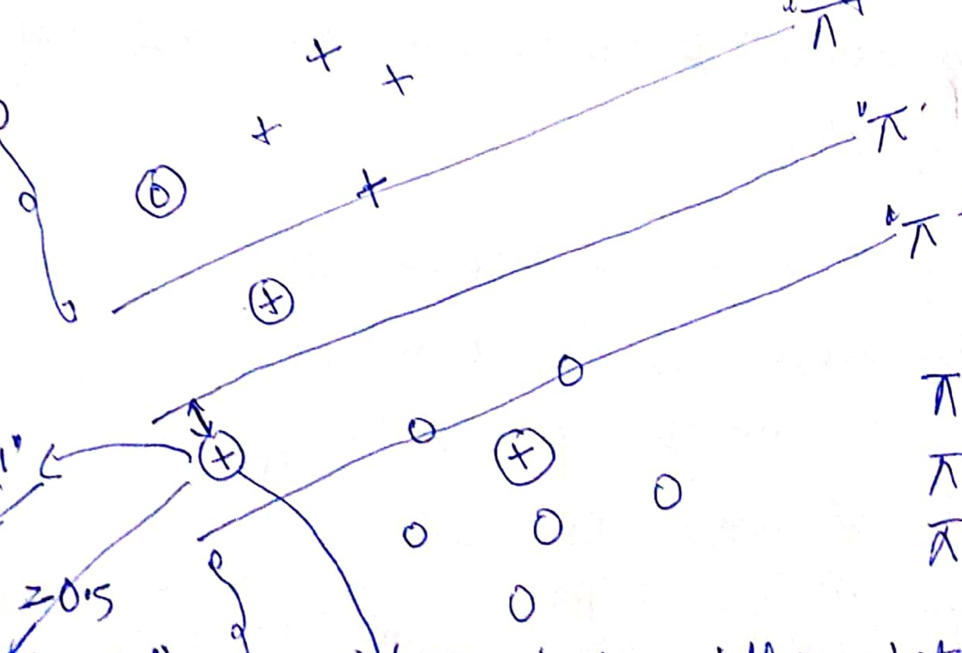
Scanned with CamScanner

It is impossible to find something like that, which satisfy this constraint.

This optimization problem or this diagrammatic representation, where all the +ive points are on the upper side of 'π^+' & all the -ive points are on the lower side of 'π^-'. This formulation is called ["hard-margin SVM"].
becoz we are saying all the +ive points are on one side & all the -ive points are on other side & nothing else & we are imposing this thing through the constraint.

Now for almost linearly separable data, let's see can we somehow modify this formulation of ["SVM"]. slightly to find a hyperplane for almost linearly separable cases also.

Let's take each of these misclassified points & try to do something



$\pi : w^Tx + b = 0$

$\pi^+ : w^Tx + b = +1$

$\pi^- : w^Tx + b = -1$

'$y_i = +1$'

Now

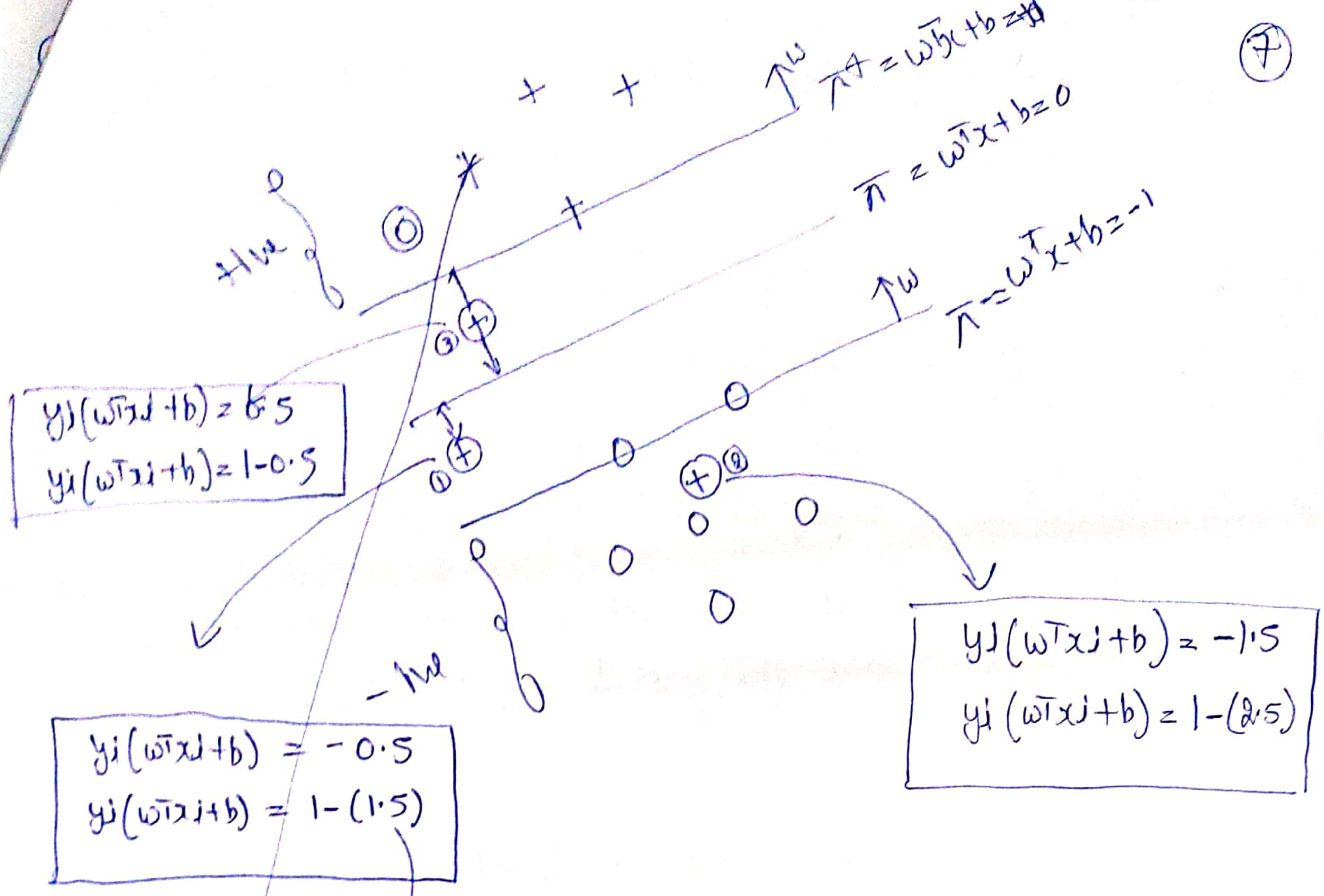
$(y_i w^T x_i + b = 0.5$

-ive

$(y_i w^T x_i + b) = 1 - (1.5)$

This point is halfway between π & π^-

Scanned with CamScanner



let us call this term as ξ_i

for ① ξ_i is 1.5 for ② ξ_i is 2.5 & for ③ ξ_i is 0.5.

& for correct point ξ_i will be "0" becoz it is already greater than '1'

So, what we do is, we create a new variable called "ξ_i"

If a +ve point lies in correct region, its ξ_i will be '0' same is with -ve point, if it lies in correct region. i.e below π plane.

If a +ve point lies anywhere else other than the region specified. then its ξ_i will be +ve

let's assume ξ_i to be ≥ 0"

So, if $\xi_i \uparrow$ then the point is farther away from the correct hyperplane in incorrect direction.

So, for every point "x_i" we are creating "ξ_i" such that $\xi_i = 0$ if $y_i(w^T x_i + b) \geq 1$

which means they are correctly classified. as per not 'π' but as per 'π^+' & 'π^-'

But $\xi_i > 0$ & it is equal to some units of distance away from the correct hyperplane either π^+ or π^- in the incorrect direction.

ξ_i's are telling us whether a point is correctly classified or not & how far it is away from the correct hyperplane in the incorrect direction.

Now let us formulate our optimization function

Our initial formulation is, we have to find (w^* & b^*) such that $\frac{2}{\|w\|}$ (marginal distance) gets maximized

$$(w^*, b^*) \quad \underset{w,b}{\operatorname{argmax}} \frac{2}{\|w\|}$$

& maximizing this is same as minimizing $\frac{\|w\|}{2}$

$$(w^*, b^*) \quad \underset{w,b}{\operatorname{argmin}} \frac{\|w\|}{2}$$

because

$\max f(x) == \min \frac{1}{f(x)}$

Scanned with CamScanner

Now let us see using ξ_i's how can we write our whole optimization problem

$$\left\{ (w^*, b^*) = \underset{w,b}{\operatorname{argmin}} \left(\frac{\|w\|}{2} \right) + C \cdot \frac{1}{n} \sum_{i=1}^{n} \xi_i \right\}$$

1/margin

such that $y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i$

& $\xi_i > 0$

we have seen that for every misclassified point. we can write $y_i(w^T x_i + b) = 1 - \xi_i$ where ξ_i is +ive

∴ for all misclassified points, we can write it as

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad \text{where } \xi_i \text{ is +ive}$$

Here what is happening is, for correctly classified points 'ξ_i' will be equal to '0'.

Now what do we want to minimize,

we want to minimize the errors or we want to minimize misclassifications.

Minimizing misclassification means that, since $\xi_i > 0$ for all misclassified points. This means we want to minimize the sum of 'ξ_i's'

So, in our objective function, earlier we have only margin

Scanned with CamScanner

Now along with margin if I say that I want to minimize the average distance (becoz ξ_i represent the distance of incorrectly classified points for correct hyperplane) in opposite direction).

$\frac{1}{n}\sum_{i=1}^{n}\xi_i$ → This is the avg. distance of misclassified points, becoz for all the correctly classified points ξ_i will be equal to '0'.

& we want to minimize ~~that~~

'C' here is hyperparameter.

Here $\frac{||w||}{2}$ is $\frac{1}{\text{margin}}$

$\frac{1}{n}\sum_{i=1}^{n}\xi_i$ → avg. distance for misclassified points

We can think of this → $\left(\frac{1}{n}\sum_{i=1}^{n}\xi_i\right)$ as a loss becoz we want to minimize it. We want to minimize the no. of misclassified points. So whenever there is a misclassified point 'ξ_i' is greater than '0'. This is basically a loss to the model that we want to minimize

'C' here is hyperparameter. C is the

as C↑; we are giving more importance to not making errors. As C↑ we are saying, we don't want to make mistakes..

C↑; tendency to make mistakes on Dtrain reduces
~~high variance model~~ → This means we are going to overfit.

C↓; $\frac{||w||}{2}$ will get more importance, & we have a tendency to underfit.
↳ high bias

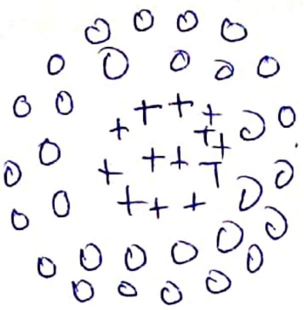
This formulation of SVM is called "soft-margin SVM"

Hard-margin SVM's donot allow errors.

but a soft-margin SVM' says make errors but minimize them.

→ 'Polynomial Kernel"

let's take an example where we have a bunch of 'true points,' surrounded by a bunch of 'negative points" & the datasets looks like two concentric circles.



In logistic regression, we can separate these points by feature transformation trick.

$$(f_1, f_2) \xrightarrow{\text{'FT'}} \underbrace{(f_1^2, f_2^2)}$$

↳ & finally we can separate them using a line.

Now let's look a polynomial kernel :→

The general definition of a polynomial kernel is

given two datapoints (x_1 & x_2) the general polynomial kernel is $(x_1^T x_2 + c)^d$.

$$K(x_1, x_2) = (x_1^T x_2 + c)^d$$

where 'c' & 'd' are constants.

Scanned with CamScanner

let's take an example of a quadratic kernel.

Eg:→ $K(x_1, x_2) = \underbrace{(1 + x_1^T x_2)^2}$

↓ Quadratic kernel

Here $c = 1$ & $d = 2$

If we apply this, let's see what is $K(x_1, x_2)$

$$K_\phi(x_1, x_2) = (1 + x_1^T x_2)^2$$

$$= \left(1 + [x_{11}, x_{12}] \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix}\right)^2$$

$$= (1 + x_{11}x_{21} + x_{12}x_{22})^2$$

Let's assume

$x_1 = \langle x_{11}, x_{12} \rangle$

↓

vector of two points

$x_2 = \langle x_{21}, x_{22} \rangle$

$$= \underbrace{1 + x_{11}^2 x_{21}^2 + x_{12}^2 + x_{22}^2 + 2x_{11}x_{21} + 2x_{12}x_{22} + 2x_{11}x_{21}x_{12}x_{22}}$$

↓ This can be represented as a product of vectors.

let $[1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12}] : x_1'$

$[1, x_{21}^2, x_{22}^2, \sqrt{2}x_{21}, \sqrt{2}x_{22}, \sqrt{2}x_{21}x_{22}] : x_2'$

→ vectors

Now we can show that this product we have written above is equivalent to $\{(x_1')^T (x_2')\}$

If we look at it carefully, we will find out x_1 & x_2 are in 2d & now for x_1' we have term only x_{11} & x_{12} & for x_2' we have only x_{21} & x_{22} terms

(1) So imagine if we are given "x_1" & "x_2" we have transformed them into x_1' & x_2'

$$\begin{array}{ccc} 'x_1' & & 'x_2' \\ \downarrow & & \downarrow \\ 'x_1'' & & x_2' \end{array}$$

Now instead of doing $\boxed{x_1^T x_2}$ we can do $\boxed{x_1'^T x_2'}$

↳ This is equivalent to "feature transform"

So, what 'kernalization' is doing internally is exactly equal to "feature transformation"

Kernelization takes "d-dimensional data" & does a feature transformation internally & implicitly

Kernelization :- $d \xrightarrow[\text{internally \& implicitly}]{FT} d'$

Feature Transformation :- $d \xrightarrow[\text{Explicitly}]{FT} d'$

Using kernel trick, we are converting d dimensional points into d' dimensional points where $d' > d$ typically

Scanned with CamScanner

x_1' is 6-d data

x_2' is 6-d data

So using kernel trick we went from 2d data to '6-d data'

$$x_1' = [1, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}, \sqrt{2}x_{12}, \sqrt{2}x_{11}x_{12}]$$

$$x_2' = [1, x_{21}^2, x_{22}^2, \sqrt{2}x_{21}, \sqrt{2}x_{22}, \sqrt{2}x_{21}x_{22}]$$

Since in these we have squared terms ($x_{11}^2, x_{12}^2, x_{21}^2, x_{22}^2$) & these terms are very similar to (f_1^2, f_2^2) terms the data will become separable.

Scanned with CamScanner