Support vector machine (SVM).

- Let us first consider a linear classifier which is basically a linear discriminant function.
So if we have a input feature vector x & if we define a linear discriminant function $g(x)$ and for simplicity we consider a 2-class problem, two classes are c_1 & c_2.
So any unknown feature vector say x is to be classified as either belonging to class c_1 or to class c_2 and the linear discriminant function is $g(x)$.

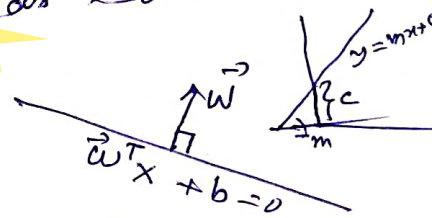
$\therefore g(x) = w^T x + b$ ← As linear fn.

w^T ↙ wt. vectors, x ↘ input feature vector, b → bias

$w^Tx+b=0 \Rightarrow$ st. line if x is 2D.
$w^Tx+b=0 \Rightarrow$ plane if x in 3-D

If x is 2-D feature vectors then $g(x)$ is a st. line
If x is 3-D " " $g(x)$ is a plane

If we increase dimension, i.e., if it is >3-D then the linear discriminant function (i.e., the linear equation) represents a hyperplane, where w is nothing but a vector perpendicular to that hyperplane (i.e. normal vector).



$\therefore \vec{w}$ represents the orientation of the hyperplane & b represents what is the position of the hyperplane in D-dimensional space, b is usually known as a bias term which is biasing the position of the hyperplane.

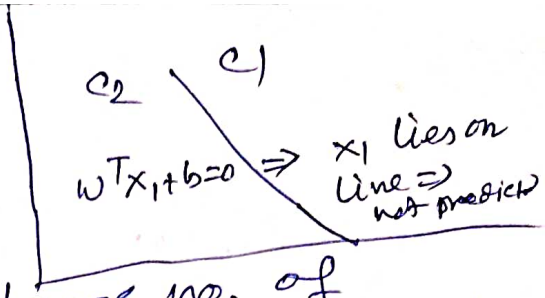
For classification problem, here for every feature vector x_1, we have to compute $g(x_1) = w^T x_1 + b$. If $g(x_1) > 0$ If x_1 lies on the +ve side of the hyperplane And if x_1 lies on -ve side of the hyperplane then $g(x_1) < 0$. And if x_1 lies on the hyperplane then $g(x_1) = 0$ i.e., $w^T x_1 + b = 0$.

$\therefore$ The hyperplane divides the D-dimensional space into two halves.

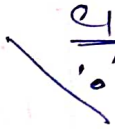
∴ The classifier rule is:

If $w^T x_1 + b > 0 \Rightarrow x_1 \in C1$

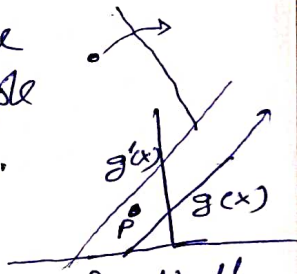
If $w^T x_1 + b < 0 \Rightarrow x_1 \in C2$



So for training we must have large no. of samples, some of which are of C1 & some of which are of C2. We have to train the model, i.e., estimate w & b such that certain objective function will be optimized. (Like some of square error). need to optimize. Training will be in iterative fashion.



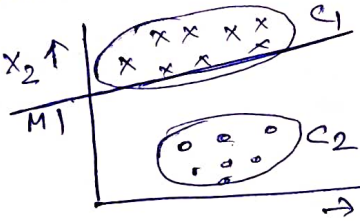
Let us consider any sample x_1 from class C1. Then we start with an initial value of $\vec{w}$ and b & compute $w^T x_1 + b$. If it should be ≥ 0. But if it is < 0 then we modify $\vec{w}$ & b in such a way that i.e., the position and orientation of the hyperplane is so modified that the sample x_1 moves to +ve side of the hyperplane.



P initially on left side of hyperplane g(x)
g(x) modified to g'(x) & P now lies on right side of g'(x).

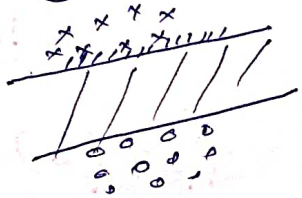
Similarly if we take a input feature vector X_2 of class C_2, the same situation will happen.

This is done in case of linear discriminant function.

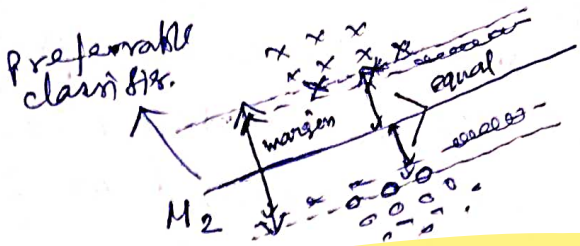


Now let x_1 & x_2 are two features & the samples are of class C1 & C2. And curve Linear classifier (M1) is as shown in the diagram. Here M1 correctly classified all samples. But is that classifier is desired? Answer is NO: why?

It is not desired because M1 gives a large bias in favour of C2, whereas it puts a penalty against class C1. The reason is → the entire space between two type of samples is given to C2 & only a small margin is given to C1.



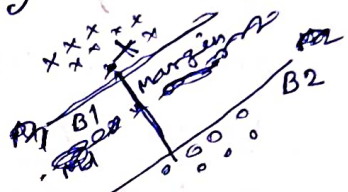
M_2 is more preferable classifier because the training vectors of both C_1 & C_2 are equally apart from the classifier.



The SVM actually tries to find a classifier which will be positioned in a form like M2.

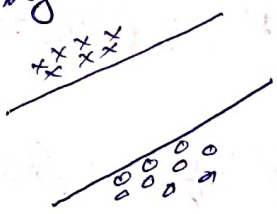
Let us see how such a SVM can actually be designed.

So the basic aim of the SVM is: If a sample x is one side of boundary (B2) and it seems to be saved if its distance from B2 is larger. If the distance is very small (for B1) then a little disturbance / noise can push x to move to other side of classifier which implies a misclassification. So it is desirable that the margin of x from the classifier will be quite large so that the sample x will be in the same side of the classifier until a large disturbance/noise is applied.



∴ Large distance from classifier ⇒ Safe sample.

The SVM tries to design the classifier that maximizes the distance of the separating boundary between two classes C_1 & C_2.



$\vec{x_i} = (x_{i1}\ x_{i2} \ldots x_{id})$ & $W = (w$

∴ If $x_i \in C_1 \Rightarrow W^T x_i + b > 0$

& If $x_i \in C_2 \Rightarrow W^T x_i + b < 0$

	x_1 x_2 ... x_d	Y
x_1		
⋮		
x_i	x_{i1} x_{i2} ... x_{id}	Y_i

As SVM is supervised learning so we know the class belongingness of every x_i. ~~x_i x_i~~ Let class belongingness of x_i is Y_i ∴ $Y_i = \pm 1$

$+1 \Rightarrow C_1$ class

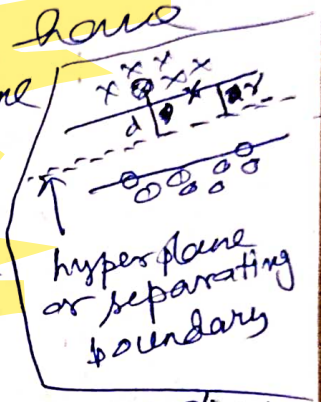
$-1 \Rightarrow C_2$ class

so for any $(x_i\ Y_i)$ $Y_i(W^T x_i + b) > 0$ since if $Y_i = +1$ then $W^T x_i + Y_i > 0$

& if $Y_i = -1$ then $W^T x_i + Y_i < 0$

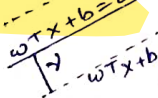
using this concept, ie, $y_i(w^T x_i + b) > 0$, we are designing the classifier. [$\Gamma = 0$ only when x_i lies on classifier]
Once we set w & b, for an unknown feature vector p, we don't have y_i, so we compute $w^T p + b$
If $w^T p + b > 0$ then we classify p as to class c_1
and if $w^T p + b < 0$ then we classify p to class c_2.

In SVM we have the objective that we have to maximize the distance of the hyperplane or the separating boundary from each of the feature vector, so that every feature vector feels that they are safer so far this classifier is concerned.



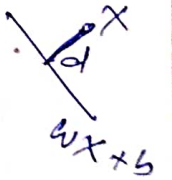
To do that, we have made a modification in our linear classifier design. Earlier we have considered →
$w^T x_i + b > 0$ if $x_i \in c_1$
& $w^T x_i + b < 0$ if $x_i \in c_2$

Now we consider
If $x_i \in c_1$ then $w^T x_i + b \geq \gamma$
If $x_i \in c_2$ then $w^T x_i + b \leq -\gamma$



where γ = some margin which is nothing but a measure of distance of x_i from the separating plane.

So if we have a hyperplane whose equation is $w^T x + b = 0$ then distance of the point x from the hyperplane is $d = \frac{|w^T x + b|}{\|w\|}$



$\therefore$ so we want $d = \frac{|w^T x + b|}{\|w\|} \geq \gamma \Rightarrow -\gamma \geq \frac{w^T x + b}{\|w\|} \geq \gamma$

Our decision regarding the correct classification of x is independent of scaling vector $\vec{w}$ because $\vec{w}$ tells only about the orientation of the plane. So whatever scaling factor is applied to $\vec{w}$, our decision remains the same. $\therefore$ we may write $w^T x + b \geq \gamma \|w\| \Rightarrow \frac{w^T x + b}{\|w\|} \geq 1$

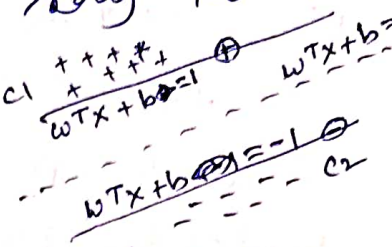
And by proper scaling, we may set, $w^T x + b \geq 1$

[$-\gamma \geq \frac{w^T x + b}{\|w\|} \Rightarrow \frac{w^T x + b}{\|w\|} \leq -\gamma$
$\Rightarrow w^T x + b \leq -\gamma \|w\|$ if $-1 \leq -\gamma \|w\|$
i.e., if $\frac{1}{\|w\|} \leq \gamma$ $\Rightarrow \gamma \geq \frac{1}{\|w\|}$]

36/1

So ~~afte~~ in this situation we can say that for every feature vector x,

$$w^T \cdot x + b \geq 1 \text{ if } x \in c_1$$
$$\leq -1 \text{ if } x \in c_2$$



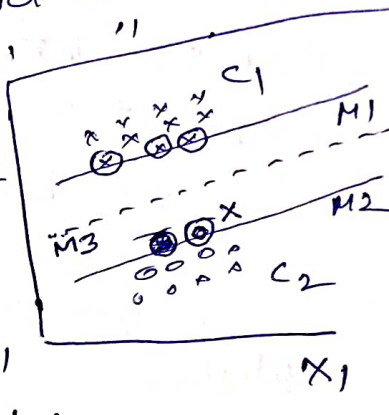
∴ During learning process, for every vector x_i always $y_i (w^T \cdot x_i + b) \geq 1$

$y_i (w^T \cdot x_i + b) = 1$ if x_i is a support vector and

$y_i (w^T \cdot x_i + b) > 1$ if x_i is not a " "

So what is the support vector?

Let us consider 2-D feature vectors with features x_1 & x_2. Then the samples are plotted. If we consider a st. line M1 where 3 samples of class c_1 are passing through M1.



Similarly, ~~to~~ 2 samples of class c_2 are passing through another st. line M2.

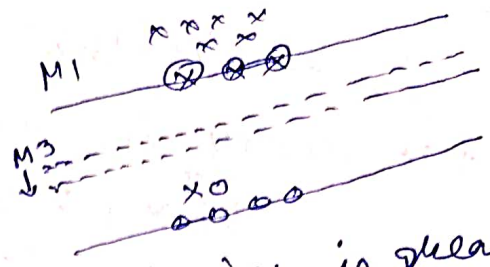
So these 3 samples of the training set are just on boundary. If any disturbance occurs then there is the possibility that these samples may be misclassified.

So we should place our classifier somewhere in the middle of M1 & M2. So M3 be the position of the classifier which will be more reliable or it is called more generalized classifier. So the risk of misclassification will be less.

∴ the position of M3 depends on the position of these 3 (3+1) feature vectors or training samples. And it does not depend on the position of other feature vectors. Even if we remove some other feature vectors from the training set, the position of the hyperplane (M3) remains the same. But if we remove, say x (lies on M2) then our hyperplane

is going to be different,

So these 3+1 = 4 feature vectors are called support vectors, others are not.



∴ A SVM is a linear machine whose design is greatly influenced by the positions of the support vectors, and are not that much influenced by the non-support vectors. ok.

So we have $y_i(W^T x_i + b) = 1$ if x_i is a support vector.

∵ we want margin γ as maximum as possible

where, $\frac{W^T \cdot X + b}{\|W\|} \geq \gamma$ (see next page *)

So we have to minimize W and maximize b.

So when we try to find the support vectors, we have to minimize W and maximize b.

$\frac{w_1 x_1 + w_2 x_2 + b}{\sqrt{w_1^2 + w_2^2}} \geq \gamma$

$\|W\|^2 = w_1^2 + w_2^2$

minimize $\|W\| = \frac{1}{2} W \cdot W$

To minimize W: It is same as, ~~if~~ we want to minimize a function, $\Phi(W) = W^T W = W \cdot W$

$= \frac{1}{2} W \cdot W$ (for mathematical convenience)

∴ We need to minimize $\Phi(W) = \frac{1}{2} W \cdot W$

Trivial value of $\vec{W}$ is $\vec{0}$, but we are not looking for it.

$x_i = (x_{i1}\ x_{i2} \cdots x_{id})$

$W = (w_1\ w_2 \cdots w_d)$

$0\ 0 \cdots 0$

d no wts.

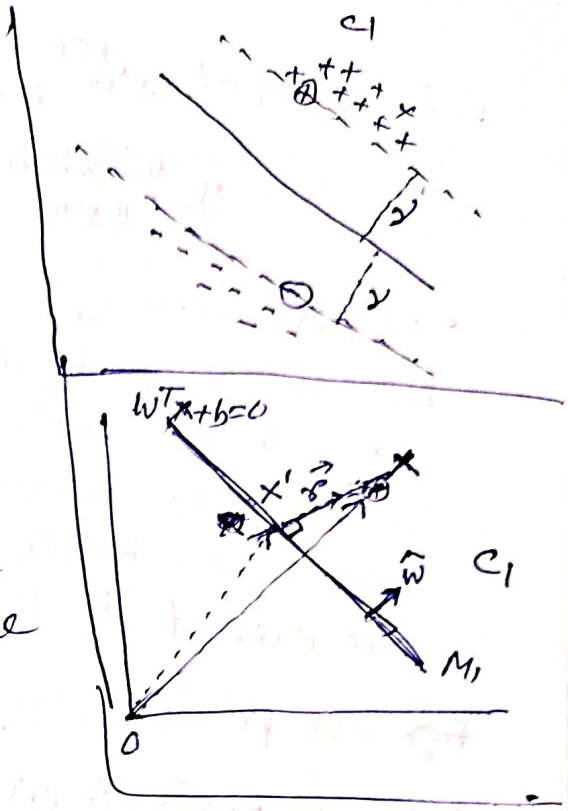
Here the constraint is $y_i(W \cdot x_i + b) = 1$

So we have to minimize W or $\Phi(W) = \frac{1}{2} W \cdot W$

Subject to the constraint $y_i(W \cdot x_i + b) \geq 1$ but as support vectors satisfy equality so we consider the constraint $y_i(W \cdot x_i + b) = 1$.

So it is a constrained optimization problem. This problem can be converted to an unconstrained optimization problem by using the Lagrangian multiplier.

36/2



Let M_1 is the hyperplane & x is a feature vector of C1 class.
∴ position vector of x is $\vec{OX}$.
Draw $\perp^r$ from x to hyperplane say it meets at x'
∴ position vector of x' = $\vec{Ox'}$
Let $x'x = \vec{r}$
The hyperplane is $w^T x + b = 0$ where $\hat{w}$ is the unit vector normal to the hyperplane.

$$\therefore \vec{r} = x'x = ox - ox' = \vec{x} - \vec{x'}$$

(*) for x', $w^T x' + b = 0$
$\vec{r}$ is $\perp^r$ to hyperplane, so $\vec{r}$ is ∥l to $\hat{w}$
Let # $|\vec{r}| = \gamma$, margin for x [As we are interested for margin]

$$\therefore \vec{r} = \gamma \hat{w} = \gamma \frac{\vec{w}}{\|\vec{w}\|}$$

$$\therefore \vec{r} = \vec{x} - \vec{x'} \text{ gives } \vec{x'} = \vec{x} - \vec{r} = \vec{x} - \frac{\gamma \vec{w}}{\|\vec{w}\|}$$

$$\therefore w^T x' + b = 0 \Rightarrow w^T\left(\vec{x} - \frac{\gamma \vec{w}}{\|\vec{w}\|}\right) + b = 0$$

$$\Rightarrow w^T x - \frac{\gamma \vec{w}^T \vec{w}}{\|\vec{w}\|} + b = 0$$

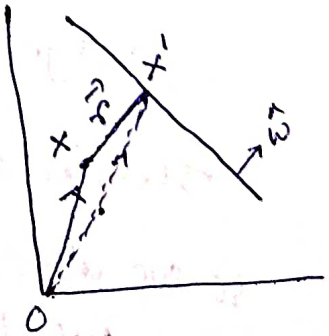
$$\Rightarrow w^T x - \gamma \|\vec{w}\| + b = 0$$

$$\Rightarrow \gamma = \frac{w^T x + b}{\|\vec{w}\|}$$

∵ $w^T w = \|w\|^2$

$\|w\| = \sqrt{w_1^2 + w_2^2 + \cdots}$

Similarly if x is in class C2:



$$\vec{r} = xx' = o\vec{x'} - o\vec{x} = x' - x$$

For x', $w^T x' + b = 0$. $\vec{r} = \gamma \hat{w}$ & $x' = \vec{r} + \vec{x}$
$= \gamma \frac{\vec{w}}{\|w\|} + x$

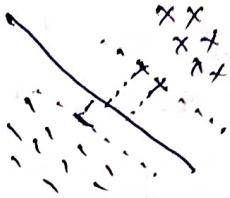
$$\therefore w^T x' + b = 0 \Rightarrow w^T\left(\vec{x} + \frac{\gamma \vec{w}}{\|w\|} + b\right) = 0$$

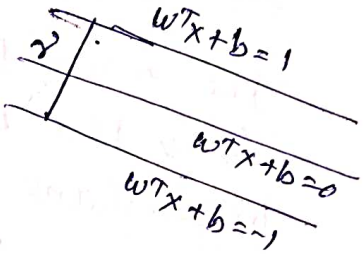
$$\Rightarrow w^T x + \gamma \frac{w^T w}{\|w\|} + b = 0 \Rightarrow w^T x + \gamma \|w\| + b = 0$$

$$\Rightarrow \gamma = -\frac{w^T x + b}{\|w\|}$$

∴ In general, the margin for any point (may be in c_1 or c_2) is $\gamma = y \cdot \frac{w^T x + b}{\|w\|}$ (y = label of x if x in $c_1 \Rightarrow y = +1$ & x in $c_2 \Rightarrow y = -1$)

The objective of SVM is to maximize this margin. We can compute the margin for every data point but cannot maximize all margins. We try to maximize the margins for the data points of both classes which are closest to the hyper plane.





for +ve points (in c_1), $\gamma = \frac{w^T x + b}{\|w\|}$

∴ for +ve points (in c_1) on margin (= support vectors),

$\gamma = \frac{1}{\|w\|}$ [∵ $w^T x + b = 1$ for those points]

Similarly, for -ve points on margin,

$\gamma = -\frac{w^T x + b}{\|w\|} = \frac{-(-1)}{\|w\|}$ [∵ for -ve points on margin (support vectors), $w^T x + b = -1$]

$= \frac{1}{\|w\|}$.

∴ Total margin is $= \frac{2}{\|w\|}$.

∴ we want to maximize margin ⇒ we want to maximize $\frac{2}{\|w\|}$

∴ we have to minimize, $\phi(w) = \|w\| \Rightarrow \frac{1}{2}\|w\|^2 = \frac{1}{2} w \cdot w$

∴ optimization formulation is:

Minimize $\frac{1}{2}\|w\|^2$
w, b

Subject to constraints, $y_i (w^T x_i + b) \geq 1$, for $i = 1, 2, \ldots m$ → {this constraints ensures that data are correctly classified}

[∵ ~~we are considering only the support vectors~~]

This is a constrained optimization problem. So we cannot apply here gradient descent method.

[For regression or perceptron we have unconstrained optimization]

36/3

We can't apply linear programing, as though constraints are linear ~~but is~~ but objective function is non-linear (quadratic).

∴ we have quadratic objective function with m-inequality constraints. To solve this we have many methods (like in MATLAB → quadprog, in C CVXOPT (Convex optimization package), sure that some package in python should exist.

Need to supply initial weights and constraints to get the result

But this is not the good solution.

Another best Technique is used Lagrangian ~~method~~ Multiplier method.

Method 1: for solving constrained optimization problems of differentiable functions.

This convert the constrained opt. meth to unconstrained opt. problem

$$\underset{x,y}{\text{minimize}}\ f(x,y) = 2 - x^2 - 2y^2 \quad \text{(quadratic)}$$
$$\text{subject to } g(x,y) = x + y - 1 = 0 \quad \text{(equality)}$$

A Lagrangian multiplier (α) combine the two equations into one: $\underset{x,y,\alpha}{\text{minimize}}\ L(x,y,\alpha) = f(x,y) - \alpha\, g(x,y)$

Whatever soln. we are getting will satisfy the constraints.

Method 2: for - m - constraints

$$\min_{x,y} f(x,y) = x^2 + y^2$$
$$\text{s.t}\quad g_1(x,y) = x + 1 = 0 \Rightarrow \min_{x,y,\alpha} L(x,y,\alpha) = f(x,y) - \sum_{i=1}^{m} \alpha_i g_i(x,y)$$
$$g_m(x,y) = \dots = 0$$

Method 3 Handling Inequality constraints:

$$\underset{x,y}{\text{minimize}}\ f(x,y) = x^2 + y^2$$
$$\text{s.t}\quad g(x) = x^2 - 1 \geq 0$$

Here, inequality constraints are transferred as constraints on α

$g(x) \geq 0 \Rightarrow \alpha \geq 0$; $g(x) \leq 0 \Rightarrow \alpha \leq 0$ and

$g(x) = 0 \Rightarrow \alpha$ is unconstrained.

$\therefore \min_{x,y,\alpha} L(x,y,\alpha) = f(x,y) + \alpha\, g(x)$

subject to $\alpha \geq 0$

$\therefore$ it is still a constrained optimization problem but simpler one.

This helps to solve our SVM problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2 = \frac{1}{2} w^T w$$

$$s.t \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, 2, \cdots, m$$

Introducing Lagrangian multipliers $\alpha_1, \alpha_2, \cdots \alpha_m$ we get

$$\min_{w,b,\alpha} L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left[y_i(w^T x_i + b) - 1 \right]$$

$$s.t \quad \alpha_i \geq 0$$

$\Longrightarrow$ Here also we may use package and get soln.

(*) we define a Lagrangian function which is to be minimized

$$\min_{w,b} L(w,b) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i \left[y_i(w^T x_i + b) - 1 \right]$$

$$s.t \quad \alpha_i \geq 0$$

we will use dual programing problem so its dual is:

$$\max_{\alpha_i} \phi(\alpha) = \max_{\alpha} \min_{w,b} L(w,b)$$

$$= \max_{\alpha_i} \left\{ \min_{w,b} \left[\frac{1}{2}\|w\|^2 - \sum \alpha_i \left[(w^T x_i + b) - 1 \right] \right] \right\}$$

$$\sum_{i=1}^{m} \alpha_i y_i (w \cdot \vec{x}_i) = \alpha_1 y_1 (w \cdot x_1) + \alpha_2 y_2 (w \cdot x_2) + \cdots + \alpha_m y_m (w \cdot x_m)$$

$$= \alpha_1 y_1 \left[\left(\sum \alpha_i y_i \vec{x}_i \right) \cdot \vec{x}_1 \right] + \alpha_2 y_2 \left[\left(\sum \alpha_i y_i \vec{x}_i \right) \cdot \vec{x}_2 \right] + \cdots$$

$$= \alpha_1 y_1 \left[\alpha_1 y_1 \vec{x}_1 + \alpha_2 y_2 \vec{x}_2 + \cdots \right] \cdot \vec{x}_1 + \quad \Downarrow \quad + \cdots$$

$$= \alpha_1 y_1 \left(\alpha_1 y_1 \vec{x}_1 \cdot \vec{x}_1 + \alpha_2 y_2 \vec{x}_2 \cdot \vec{x}_1 + \cdots \right) \Downarrow \quad + \cdots$$

$$= \sum \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$$

$$\vec{w} \cdot \vec{w} = \left(\sum \alpha_i y_i \vec{x}_i \right) \cdot \left(\sum \alpha_j y_j \vec{x}_j \right) = \sum \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$$

1 see later.

37.

So we can define a Lagrangian function as

$$L(W,b) = \frac{1}{2}(W\cdot W) - \sum \alpha_i [Y_i(W\cdot X_i + b) - 1] \quad \text{---(1)}$$

(minimize, maximize) and we have to optimize the Lagrangian multiplier.

where α_i = Lagrangian multiplier.

∴ For optimization of this Lagrangian L we know that this optimization has to be done by computing $\frac{\partial L}{\partial W}$ & $\frac{\partial L}{\partial b}$

Now expanded form of L is

$$L(W,b) = \frac{1}{2}(W\cdot W) - \sum \alpha_i Y_i (W\cdot X_i) - \sum \alpha_i Y_i b + \sum \alpha_i \quad \text{---(2)}$$

$$\therefore \frac{\partial L}{\partial b} = -\sum \alpha_i Y_i = 0 \quad \text{for optimization.} \quad \text{---(3)}$$

∴ One constraint is: $\sum_{i=1}^{m} \alpha_i Y_i = 0$ where m = no. of feature vectors.

Now $\frac{\partial L}{\partial W} = ?$

$$L(W,b) = \frac{1}{2}(W\cdot W) - \sum \alpha_i Y_i (W\cdot X_i) - \sum \alpha_i Y_i b + \sum \alpha_i$$

$$\therefore \frac{\partial L}{\partial W} = W - \sum \alpha_i Y_i X_i = 0 \Rightarrow$$

$$W = \sum_{i=1}^{m} \alpha_i Y_i \vec{X}_i \quad \text{---(4)}$$ Another constraint

Now putting W from (4) in (2) we get the Lagrangian expression:

$$L = \frac{1}{2}(W\cdot W) - \sum \alpha_i Y_i (W\cdot X_i) - \underbrace{\sum \alpha_i Y_i b}_{=0 \text{ using (3)}} + \sum \alpha_i$$

$$= \frac{1}{2} \sum_{\substack{i=1\ldots m \\ j=1\ldots m}} \alpha_i \alpha_j Y_i Y_j (\vec{X}_i \cdot \vec{X}_j) - \sum \alpha_i \alpha_j Y_i Y_j (\vec{X}_i \cdot \vec{X}_j) + \sum \alpha_i$$

$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j Y_i Y_j (\vec{X}_i \cdot \vec{X}_j) \quad \text{---(5)}$$

∴ we have to maximize this Lagrangian L with different values of α (which are the Lagrangian multipliers & they are always positive).

Maximize because we apply dual method see later.

So we have one constraint, $\alpha_i \geq 0$. —(6)
& the other constraint is $\sum_{i=1}^{m} \alpha_i y_i = 0$

∴ We have to find out the Lagrangian multipliers which maximize the expression L.
When we try to maximize L, it is quite likely that some of the Lagrangian multipliers will be equal to 0, some value will be very high.
⇒ So If some $\alpha_i = 0$ that indicates that the corresponding training feature vectors x_i is not a support vector. So it does not influence the position of the hyperplane.
⇒ If an α_i is very high ⇒ the corresponding x_i has a high influence over the position of my decision surface or the hyper plane.
⇒ If α_i is extraordinarily high, & we can infer that corresponding x_i is a spurious point, which might have occurred due to noise. It is a disturbed point or outlier.
All these different types of interpretations we can have over α_i's.

QP: quadratic programming problems

∴ Maximize $L = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$
s.t constraints $\alpha_i \geq 0$ &
$\sum \alpha_i y_i = 0$

using QP we obtain α_i's. With these α_i's
The solution gives us the value of $w = \sum \alpha_i y_i \vec{x}_i$ [from (4)] is computed and used in our decision making process.
So our classification decision will be as follows;
for an unknown feature vector z, the classification decision

$$D(z) = \text{sgn}(\vec{w} \cdot \vec{z} + b) = \text{sgn}\left(\sum_{j=1}^{m} \alpha_j y_j (\vec{x}_j \cdot \vec{z}) + b\right)$$

∴ If $D(z) > 0$ then $z \in C_1$
< 0 then $z \in C_2$

∴ ~~Steps of SVM design~~

If the original set of feature vectors in lower dimensional space is not linearly separable, then we don't have the SVM because in SVM we assume that the samples are linearly separable. Said they are not linearly separable in lower dimensional space, we have to cast these feature vectors into higher dimensional space by using functions like radial basis functions, which are called kernel functions. So once we cast them into higher dimensional space, then by taking the samples in the higher dimensional space, we can design the SVM. Then for classification of unknown samples, say z, we have to cast them into same higher dimensional space. And after casting, we have to compute $D(z)$ and if $D(z)$ is +ve then z will be in class c_1 and if -ve then in class c_2.

So we need two terms:

i) w which has been obtained by optimization as discussed.

ii) the value of b, how we may set it?

$$D(z) = \text{sign}\left(\sum \alpha_j y_j \vec{x}_j \cdot \vec{z} + b\right)$$

The value of b can be computed as:

(As this is the margins)

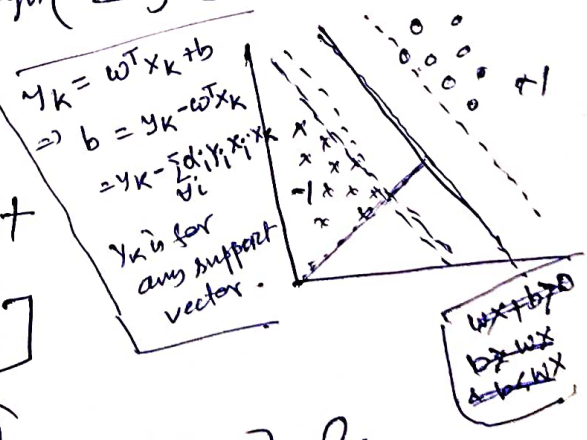
$$b = \frac{1}{2}\left[\min_{i|y_i=+1}\left(\sum \alpha_i y_i (x_i \cdot x_j)\right) + \max_{i|y_i=-1}\left(\sum \alpha_i y_i (x_i \cdot x_j)\right)\right]$$

$y_k = w^T x_k + b$

$\Rightarrow b = y_k - w^T x_k$

$= y_k - \sum \alpha_i y_i x_i \cdot x_k$

y_k is for any support vector.



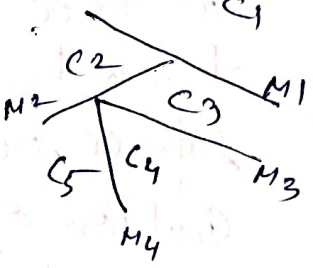
Thus $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ and $b =$ ↑ are used for classification of an unknown feature vector z. This is how our SVM works.

∴ SVM is a linear m/c that helps us placing the separating boundary between 2-classes or simply states where our hyperplane should be placed so that the classifier we get is more robust or more generalized.

* Now instead of a 2-class problem, if we have a multiclass problem, then we have multiple number of SVM,

i) one SVM tells us whether a sample belongs to class c_1 or does not in c_1.

ii) another SVM tells us whether a sample belongs to class c_2 or does not in c_2.



∴ we have (P-1) SVM if it is a P-class problem, i.e., we have (P-1) linear classifiers.