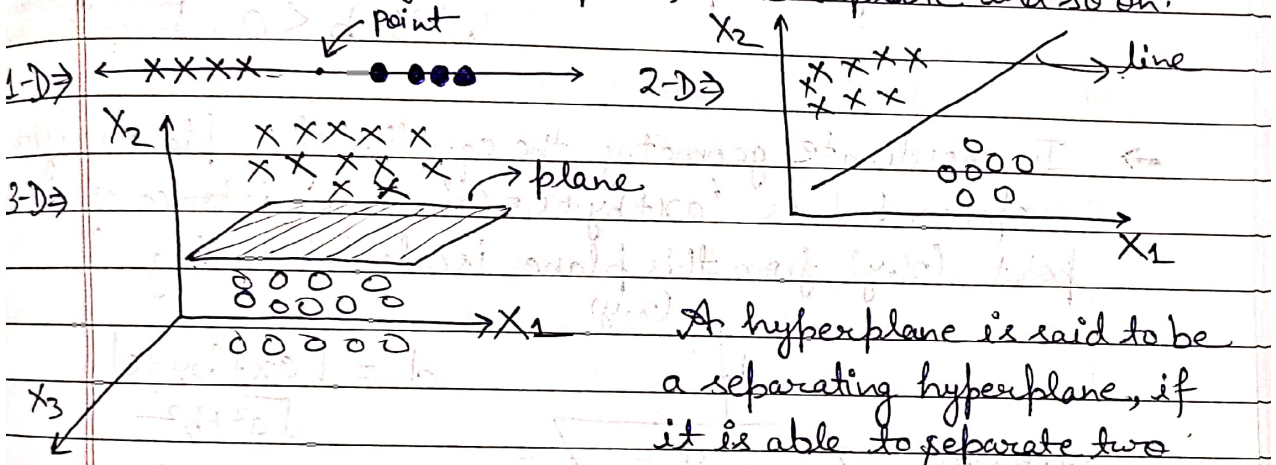
Support Vector Machine (SVM)

Support Vector Machines (SVMs) are powerful supervised learning models used primarily for classification, but can also be used for regression tasks. They are effective in high-dimensional spaces and versatile in handling both linear and non-linear data through the use of kernel functions.

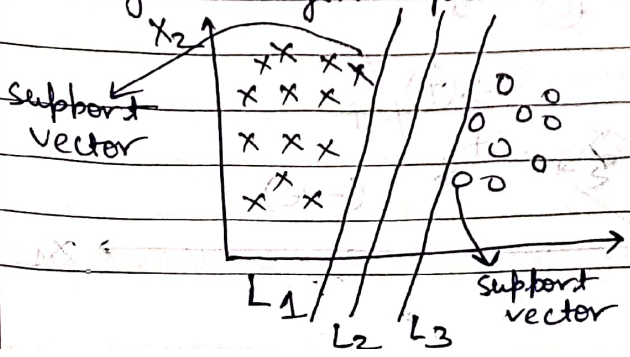
→ **Hyperplane:** A hyperplane is a plane of 'n-1' dimensions in a 'n' dimensional feature space, that separates two classes. For a 1-D space, it is a point, for 2-D, it is a line, for 3-D space, it is a plane and so on.



A hyperplane is said to be a separating hyperplane, if it is able to separate two classes. The main goal of SVM is to find an optimal hyperplane that best separates data points of different classes.

→ **Support Vectors:** These are the data points that are closest to the hyperplane and are critical in defining the position and orientation of the hyperplane. The SVM tries to **maximize** the **margin** between these support vectors and the hyperplane.

→ **Margin:** The margin is the distance between the hyperplane and the nearest data points from both the classes. A larger margin implies better generalization on unseen data.



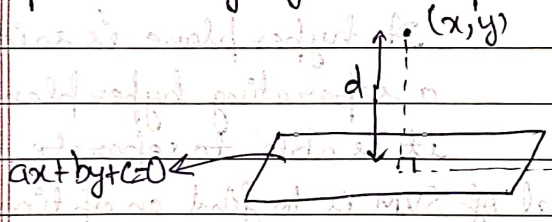
Out of three separating hyperplanes, L_1, L_2, L_3, L_3 is the best hyperplane because from that distance to support vectors is maximum.

In 2-D, the equation of a line (hyperplane) is given as—
$W^T X + b = 0$, where W is the weight vector (in 2D, it has two components, $W = [w_1, w_2]$), X is the input vector (in 2D, $X = [x_1, x_2]$), b is the bias term/intercept.
In a binary classification problem, the goal is to separate two classes ($y = +1$ and $y = -1$), the decision boundary should satisfy the condition:

$$W^T X + b = 0$$

such that for all points: $W^T X + b \geq 0$, for all point belongs to positive class
$W^T X + b < 0$, for all points belong to negative class.

→ In coordinate geometry, the equation of a plane is often represented as '$ax + by + c = 0$', and distance of a point (x, y) from this plane is calculated using:

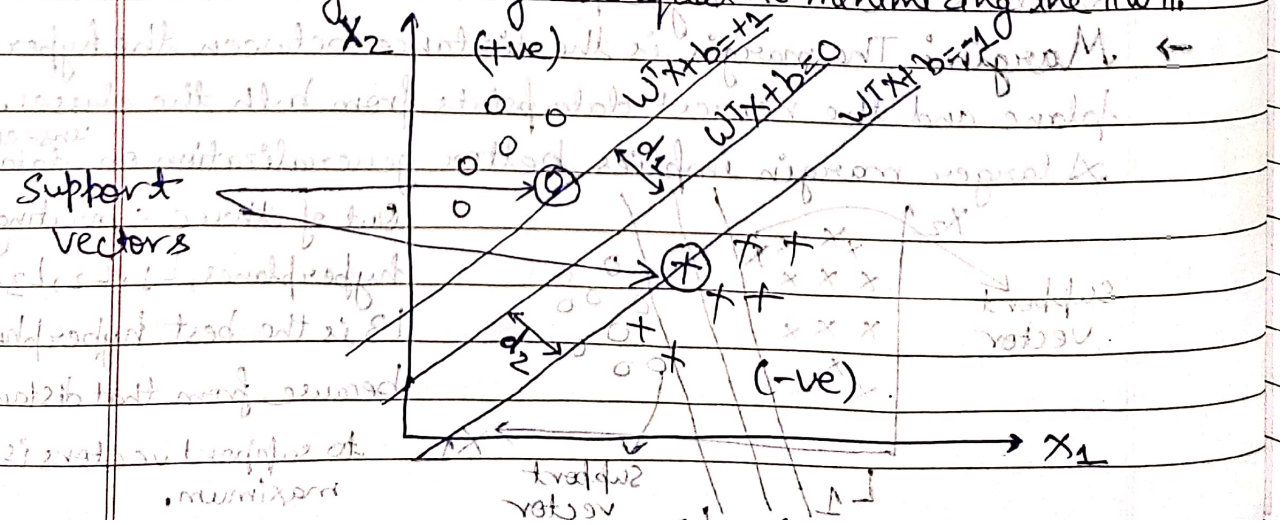


$$d = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}}$$

In our case, the distance can be represented as

$$d = \frac{W^T X + b}{\|W\|_2}$$ → L2 Norm

Our goal is to maximize the distance between the plane and the support vectors. The distance/margin is inversely proportional to norm of the weight vector $\|W\|$. Therefore, maximizing the margin is equal to minimizing the $\|W\|$.



$$d_1 = \frac{|W^T x_{(+ve)} + b|}{\|W\|} = \frac{1}{\|w\|}$$, for positive support vector

$$d_2 = \frac{|W^T x_{(-ve)} + b|}{\|W\|} = \frac{|-1|}{\|w\|} = \frac{1}{\|W\|}$$, for negative support vector.

distance $d = d_1 + d_2 = \frac{2}{\|W\|_2}$ → L2 Norm.

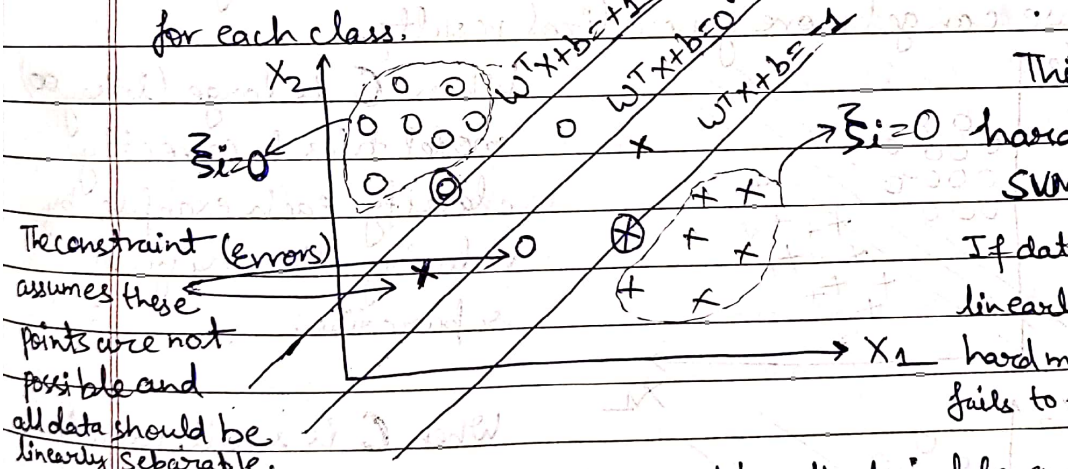
Thus, optimization problem can be formulated as –

final SVM objective

$$\min_{W,b} \frac{1}{2}\|W\|_2^2$$, subject to the constraint $y_i(W^T x_i + b) \geq 1 \quad \forall_i$

Here, y_i is the label of the data point X_i (either +1 or -1)

However, this constrained optimization problem is not straight forward, and directly solving it can lead to a non-convex function because of the hard-margin constraints. The constraints we have defined assumes that there should be no data points beyond the support vectors for each class.



The constraint assumes these points are not possible and all data should be linearly separable.

This is called hard-margin SVM.

If data is not linearly separable hard margin SVM fails to find a solution

But in real-word data may not be that simple and such cases can happen because of presence of outliers or noise. **Soft Margin SVM** allows for some misclassification by allowing / introducing slack variable ξ_i (zeta) that measures the degree to which a data point violates the margin constraint.

* The goal of soft margin SVM is to find a balance between maximizing the margin and minimizing the classification error (misclassifications). This is controlled by a **C-parameter**, which determines the trade-off between having a larger

margin and allowing more misclassifications.
For these misclassifications / error points, we want to add some penalty to our SVM objective which is given as —

hinge loss

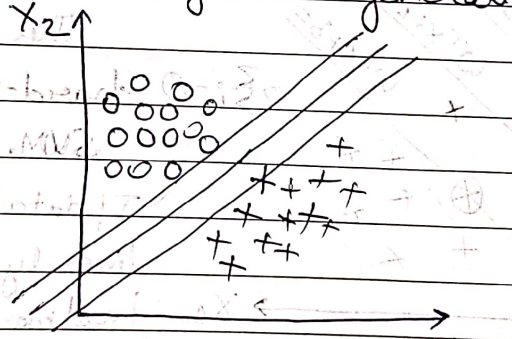
Primal objective

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi_i, \text{ subject to } y_i(w^T x_i + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \; \forall i$$

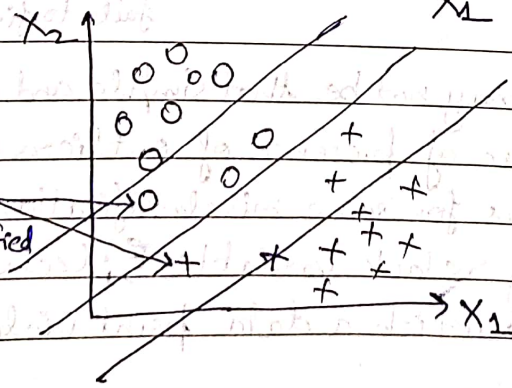
- Here, $\xi_i \geq 0$ are the slack variables that allow for some margin violations.
- C is a regularization parameter that controls the penalty for misclassifications:

→ A larger C puts more emphasis on minimizing misclassifications (i.e., fewer misclassified points but possibly a small margin) which can be prone to overfitting.

→ A smaller C allows for a larger margin but potentially more misclassified points. At the cost of some error, we can get more generalized results.



When C is large (let's say ∞) model tries to correctly classify each example, by reducing the margin of separation.



When C is small, we allow some misclassifications, thus margin of separation can be maximized helping us to get more generalized result.

Our primal objective function has some constraints so we cannot directly use it for finding the global minima by reducing the distance of margin.

Instead we need to constrained optimization problem into an unconstrained optimization problem.
We use the concept of **Lagrange's Multiplier** using which we can optimize our objective without thinking about the constraints

→ Using Lagrange's Multiplier, our primal objective can be written as —

$$L = \frac{\|W\|^2}{2} - \sum \alpha_i [y_i (W^T X_i + b) - 1]$$

The next step is to eleminate W & b from Lagrangian by setting the derivatives of L w.r.t. W & b to zero leading to a **dual problem**:

$$\frac{\partial L}{\partial W} = \bar{W} - \sum \alpha_i y_i \bar{X_i} = 0$$

$$\Rightarrow \boxed{\bar{W} = \sum \alpha_i y_i \bar{X_i}}$$ This tells us that vector $\bar{W}$ can be obtained by a linear sum of the sample vectors. At the end of optimization, a lot of α_i is going to be 0 and non-zero α_i corresponds to support vectors.

$\frac{\partial L}{\partial b} = \sum \alpha_i y_i = 0$, If we substitute these values of $\bar{W}$ & $\sum \alpha_i y_i = 0$ in the Lagrangean' equation we will get

$$L = -\sum \alpha_i - \frac{1}{2} \sum_{i=1} \sum_{j} \alpha_i \alpha_j y_i y_j \bar{X_i} \cdot \bar{X_j}$$

This signifies that optimization depends on dot product of pairs of samples.

* **SVM in case of non-linearly separable data:**
When data is not linearly separable, then SVM use **Kernel Trick** which helps SVM to project the data in a higher dimensional space, without explicitly computing the coordinates in that space which enables SVM to classify data points and handle non-linear decision boundaries.

Ⓚ From Langrangean eqⁿ, we inferred that decision function / optimization was only dependent on the dot product of $\bar{X}_i \cdot \bar{X}_j$, thus we can use any transformation function / kernel function to represent the dot product of $\bar{X}_i \cdot \bar{X}_j$. Thus Lagrange eqⁿ in this term can be expressed as — Kernel function

$$L = -\sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\bar{X}_i \cdot \bar{X}_j)$$

here $K(\bar{X}_i, \bar{X}_j)$ is the kernel function that computes the dot product in the higher dimensional space, allowing the SVM to find a non-linear decision boundary in the original feature space.

* **Common kernel functions are —**

(1) **Linear Kernel:** Formula : $K(\bar{X}_i, \bar{X}_j) = (X_i, X_j)$
The kernel corresponds to the standard dot product in the original feature space. It is equivalent to using no kernel at all and is used when the data is linearly separable.

(2) **Polynomial Kernel:** Formula : $K(X_i, X_j) = (\langle X_i, X_j \rangle + c)^d$
This kernel maps the data into a higher dimensional polynomial space. The parameter 'd' determines the degree of polynomial, and c is a constant that trades off the influence of higher order and lower order terms.

(3) **Radial Basis Function (RBF) / Gaussian Kernel:**
Formula : $K(X_i, X_j) = \exp(-\gamma \|X_i - X_j\|^2)$
The kernel maps the data into an infinite dimensional space and is widely used for non-linear problems. The parameter γ controls the width of the gaussian function determining how far the influence of a single training example reaches.

(4) **Sigmoid Kernel :** Formula : $K(X_i, X_j) = \tanh(\alpha \langle X_i, X_j \rangle + c)$
This kernel is inspired by neural networks, where tanh is the activation function.

Advantages of Kernel Trick:

1. No explicit mapping → The kernel trick avoids the explicit computation of co-ordinates in the higher dimensional space, reducing computational complexity.
2. It can be used for wide range of non-linear classification problems.