Short & Sweet (Episode 5)

Support Vector Machine Algorithm

Support Vector Machine is another **supervised machine learning algorithm** which we also called **maximum margin classifier** because it is mostly used for classification purpose (mostly text based task) . In short, this algorithm tries to separate the target classes using building **hyperplane** . This algorithm is also flexible for linearly separable data as well as non-linear separable data. More benefits of this algorithm are :-

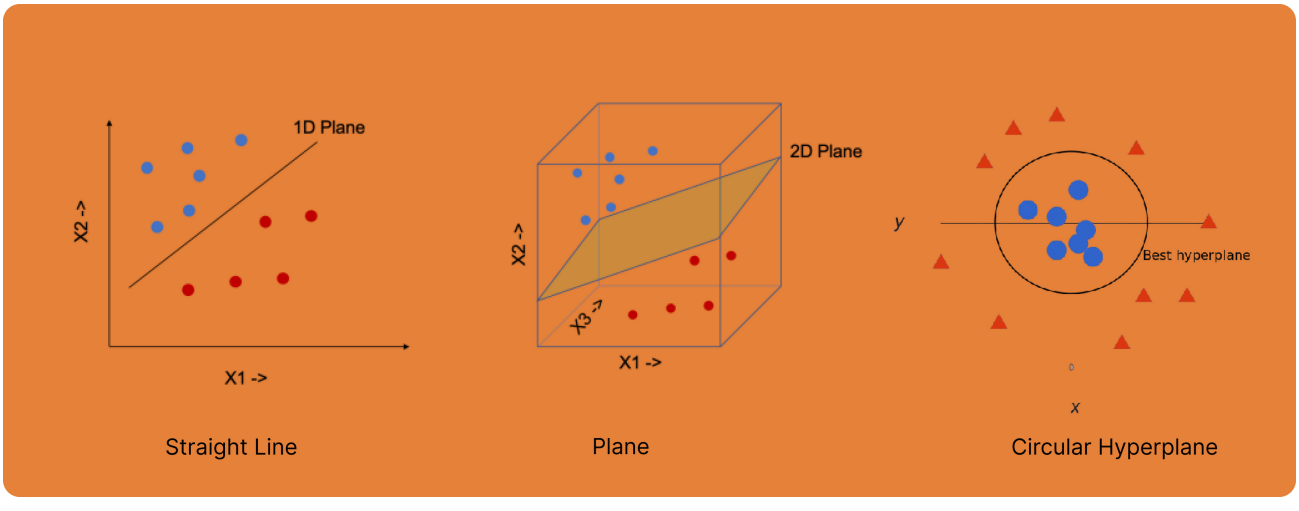
- Effective in High Dimensional spaces.
- Versatility with Kernels.
- Robust against overfitting.
- Effective in handling outliers.
- Memory Efficient

Now let's see how this algorithm works -

How It Works

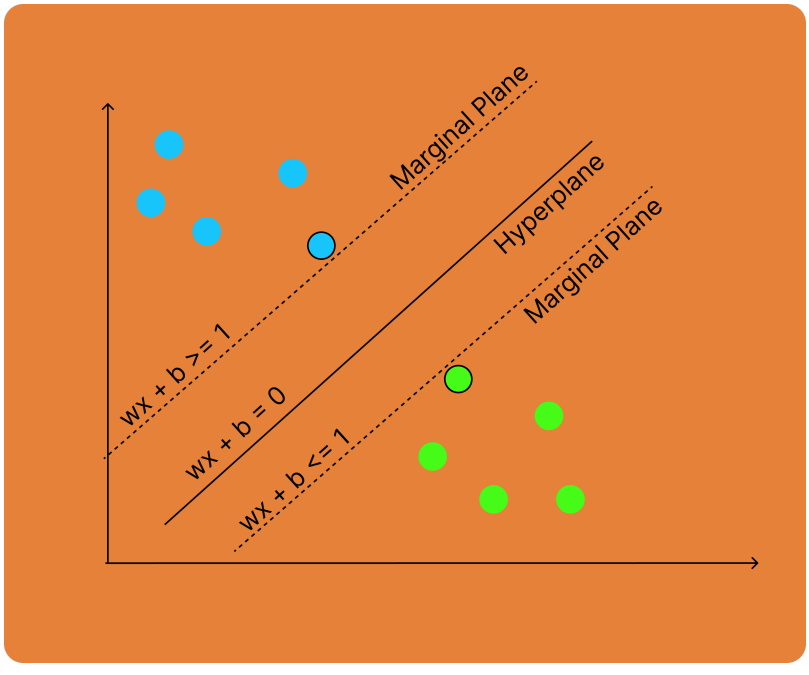
This algorithm tries to separate the data using **hyperplane -**

- Technically, **hyperplane** is a geometric concept in mathematics that refers to a flat, n-1 dimensional subspace within an **n-dimensional space**. In simpler terms, in a two-dimensional space (like a plane), a hyperplane is a **straight line**. In **higher dimensions**, it's a flat surface or a subspace that divides the space into two parts. For instance, in three dimensions, a hyperplane is a **flat plane**, while in higher dimensions, it's a **higher-dimensional flat surface**. This hyperplanes separates the target classes like some of the below examples :-

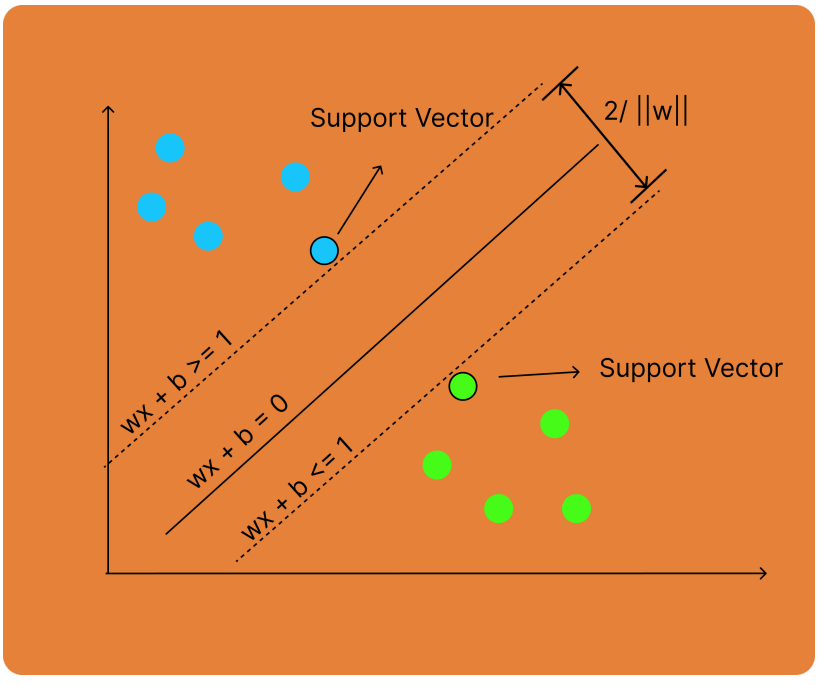


- Let's know about the **mathematical concepts** behind this algorithm -
  - As we know the equation of line which is **y = wx + b,** modify it little bit like below -
    - **y = wx + b**

- **wx - y + b = 0**
- Now we know x and y is a part of same vector which is line, plane or anything. So let **(x,y)** be **X** and **(w,-1)** be **W**. After this , it will look like **WX + b = 0**.
- So we know, this is some thing to build line, plane or higher dimensional plane but how this support vector machine is different.
- We'll create two more hyperplanes which are near to the extreme points of each target class. We called those hyperplanes, **marginal plane**.
- Equations for this two marginal planes are - **WX + b >= 1** && **WX + b <= 1**. Here 1 doesn't meant for any constant value it defines that If **WX + b** is greater than **0** means it falls in one class said as **+1** else falls in another class said as **- 1**. We'll have more constant ranges if we are performing multi-classification.



- So basically what we want is to maximize the distance between these two marginal planes such that hyperplane can classify the data points with accuracy. Distance can also be called as margin and that's why we call this algorithm **maximum margin classifier**.
- As I've said these marginal planes are the planes created nearest to the extreme points which are called **support vectors**.
- Now on subtracting marginal planes equation, we get the value of margin that we want to maximize right . You can see the calculation below -
  - On subtracting **wx + b = 1** && **wx + b = -1** , we will get the marginal value as below -



$$\text{Distance} = \frac{|1-(-1)|}{\sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}} = \frac{2}{\|w\|}$$

- Constraints are defined as -

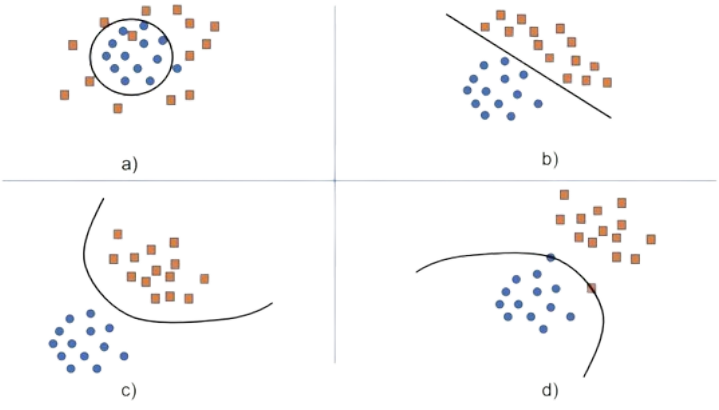
$$h(x_i) = \begin{cases} +1 & \text{if } w \cdot x_i + b \geq 0 \\ -1 & \text{if } w \cdot x_i + b < 0 \end{cases}$$

$$y_i(w \cdot x_i + b) \geq 1$$

- We'll eventually tries to find the optima value of weights and bias such that marginal value gets its maximized value and correctly classifies the target class.
- To optimize, we'll use **perceptron trick** and **lagrange interpolation** which needs a discussion in another episode.

What is hard margin and soft margin ?

In **support vector machine**, we have mostly two patterns of data, one which is linear or **perfectly separable** through hyperplane and another is **non-separable** data points which is the actual real world case. So if data is perfectly separable through line or plane we can call it a **hard margin** and if data has so many outliers, resulted hyperplanes is higher dimensional like circle, parabola, hyperbola etc. known as **soft margin**.



What is Kernels Trick ?

Now, think once if we have complicated data, how we can manage to fit the soft margin to it. That's where the **Kernel trick** comes into action. **Kernel trick** applies some transformation to data and make it more higher dimensional so that it will be separable at good amount. To be more technical, The "kernel trick" is a technique used in machine learning, particularly in Support Vector Machines (SVMs), to handle non-linearly separable data by implicitly mapping the input data into a higher-dimensional space. This method allows linear algorithms to perform effectively in this transformed space without explicitly calculating the coordinates of the data in that space.

In simpler terms, the kernel trick allows SVMs to find a linear decision boundary in a higher-dimensional space without having to explicitly transform the data into that space, which can be computationally expensive, especially for high-dimensional data.

Some techniques you can go through -

- **Linear Kernel**: (equivalent to the standard dot product).

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

- **Polynomial Kernel**:

$$K(x_i, x_j) = (x_i)^T \cdot x_j + c)^d$$

- **Radial Basis Function (RBF) Kernel (Gaussian Kernel)**:

$$K(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel)^2)$$