SVM NOTES BY SOHAM

Let’s start with SVM today

So SVM is a type of SUPERVISED classification problem where we are primarily able to do binary classification .

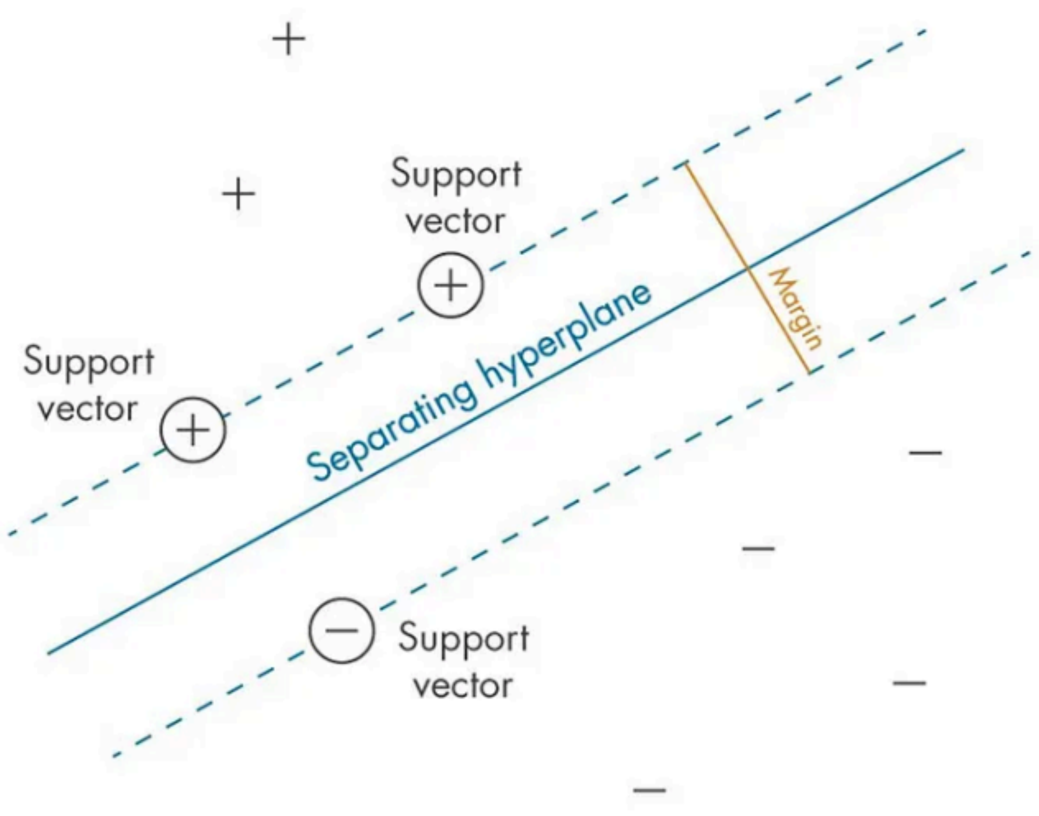
TERMINOLOGIES:

HYPERPLANE: One that separates two classes, could be just a line in case of 2D representation and in 3D it would be a plane

SUPPORT VECTORS: Closest points to decision boundaries that help us to create decision boundaries.

MARGIN: Distance between the hyperplane and the support vectors.

OUR PRIMARY AIM IS TO FORM A ROAD LIKE STRUCTURE HAVING HYPERPLANE EXACTLY IN BETWEEN AND MAXIMUM MARGINS ON BOTH ENDS.



Let us look at the equation for a straight line with slope m and intercept c.

The equation becomes : mx + c = 0

(To notice: we have fit a straight/linear line which is 1-D in a 2-D space)

SVM NOTES BY SOHAM

The hyperplane equation dividing the points (for classifying) can now easily be written as:

H: wT(x) + b = 0 ;

Here: b = Intercept and bias term of the hyperplane equation

In D dimensional space, the hyperplane would always be D -1 operator.

For example, for 2-D space, a hyperplane is a straight line (1-D).

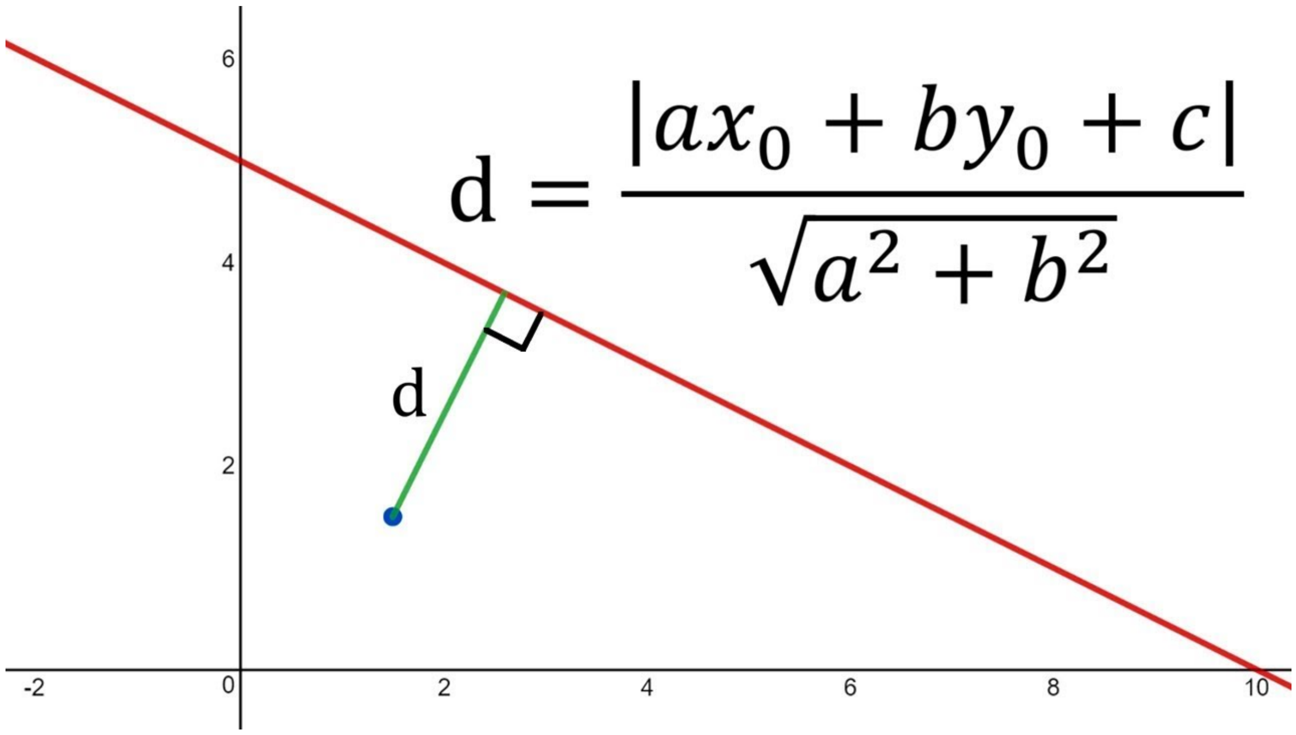
The equation of the 2 lines formed by SVMs are:

wT(x) + b = 1 for the positive class

wT(x) + b = -1 for the negative class

NOW WE NEED TO MAXIMISE THE MARGIN:

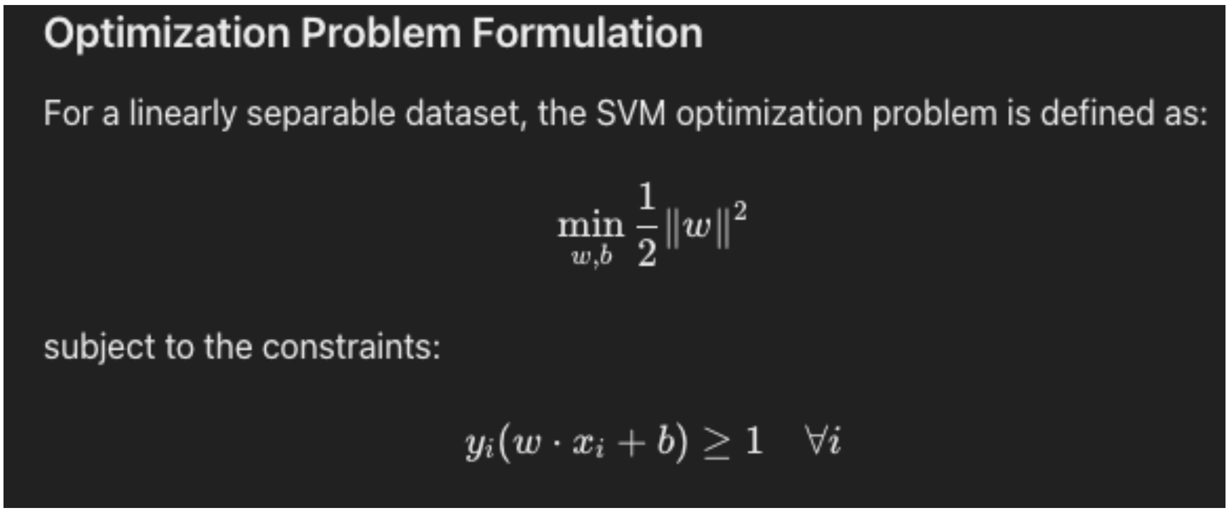
For that first we would find the distance between the hyperplane and the line formed by the SVM



According to the distance between two parallel lines or a point and the line formula we find the length of margin from the hyperplane to be 1/|w| so we would simply multiply it by 2 to find distance between the two lines created by SVMs of opposite classes.

Now length of margin is 2/|w| to maximise length we would minimise the ‘w’.

Now awe would achieve this by:



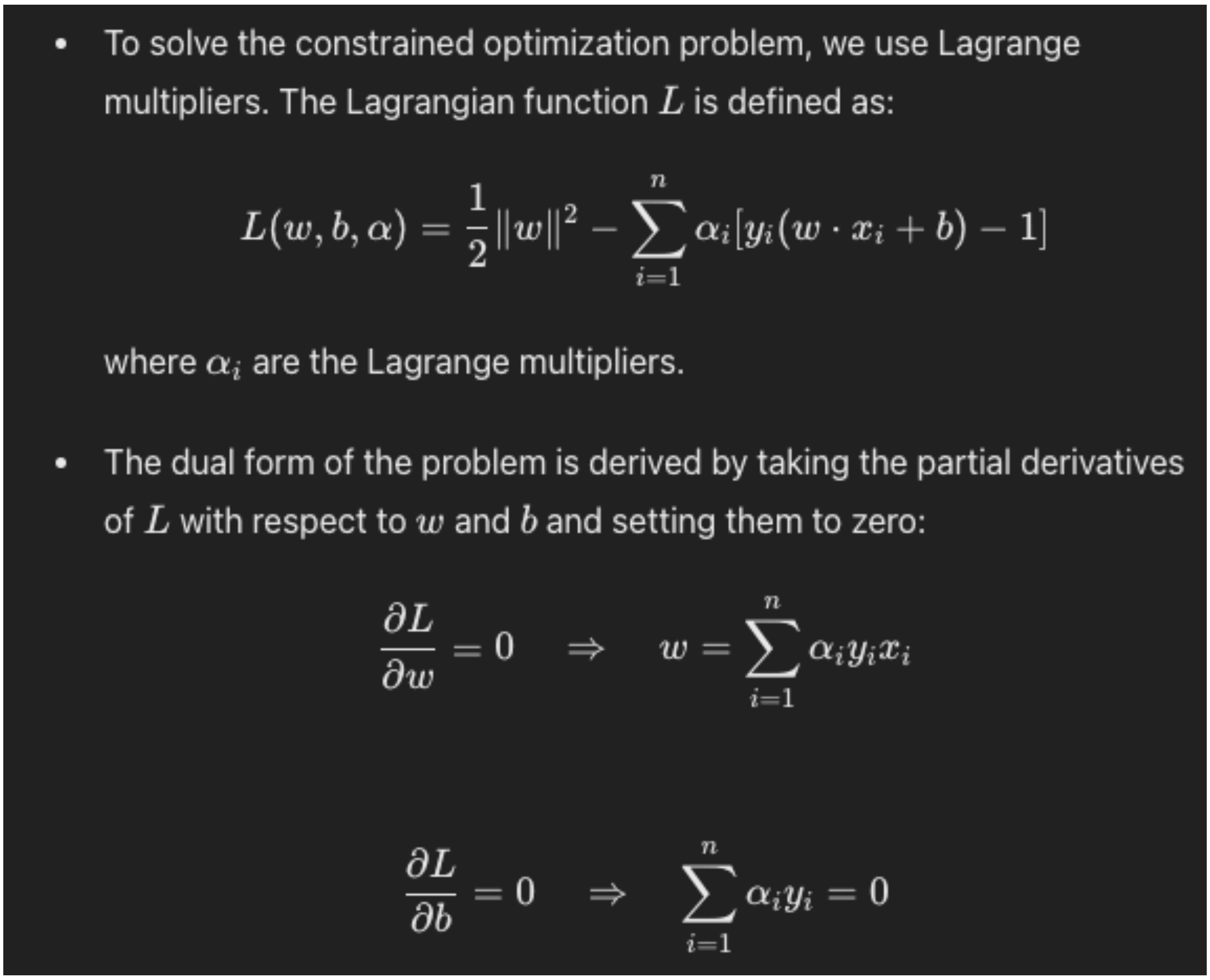
Optimization Problem Formulation

For a linearly separable dataset, the SVM optimization problem is defined as:

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

subject to the constraints:

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



- To solve the constrained optimization problem, we use Lagrange multipliers. The Lagrangian function L is defined as:

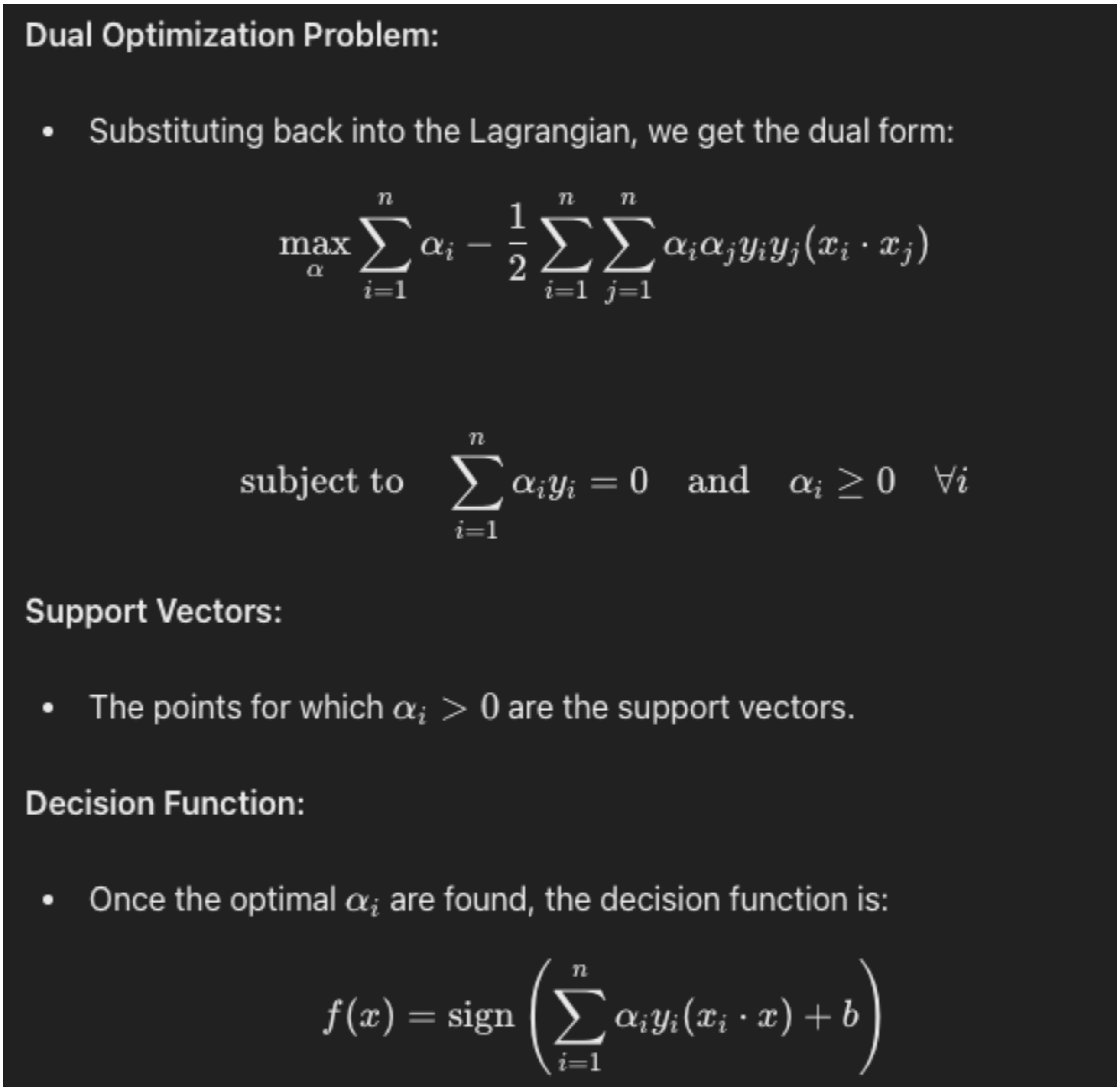
$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w \cdot x_i + b) - 1]$$

where α_i are the Lagrange multipliers.

- The dual form of the problem is derived by taking the partial derivatives of L with respect to w and b and setting them to zero:

$$\frac{\partial L}{\partial w} = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$



Dual Optimization Problem:

- Substituting back into the Lagrangian, we get the dual form:

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{and} \quad \alpha_i \geq 0 \quad \forall i$$

Support Vectors:

- The points for which $\alpha_i > 0$ are the support vectors.

Decision Function:

- Once the optimal α_i are found, the decision function is:

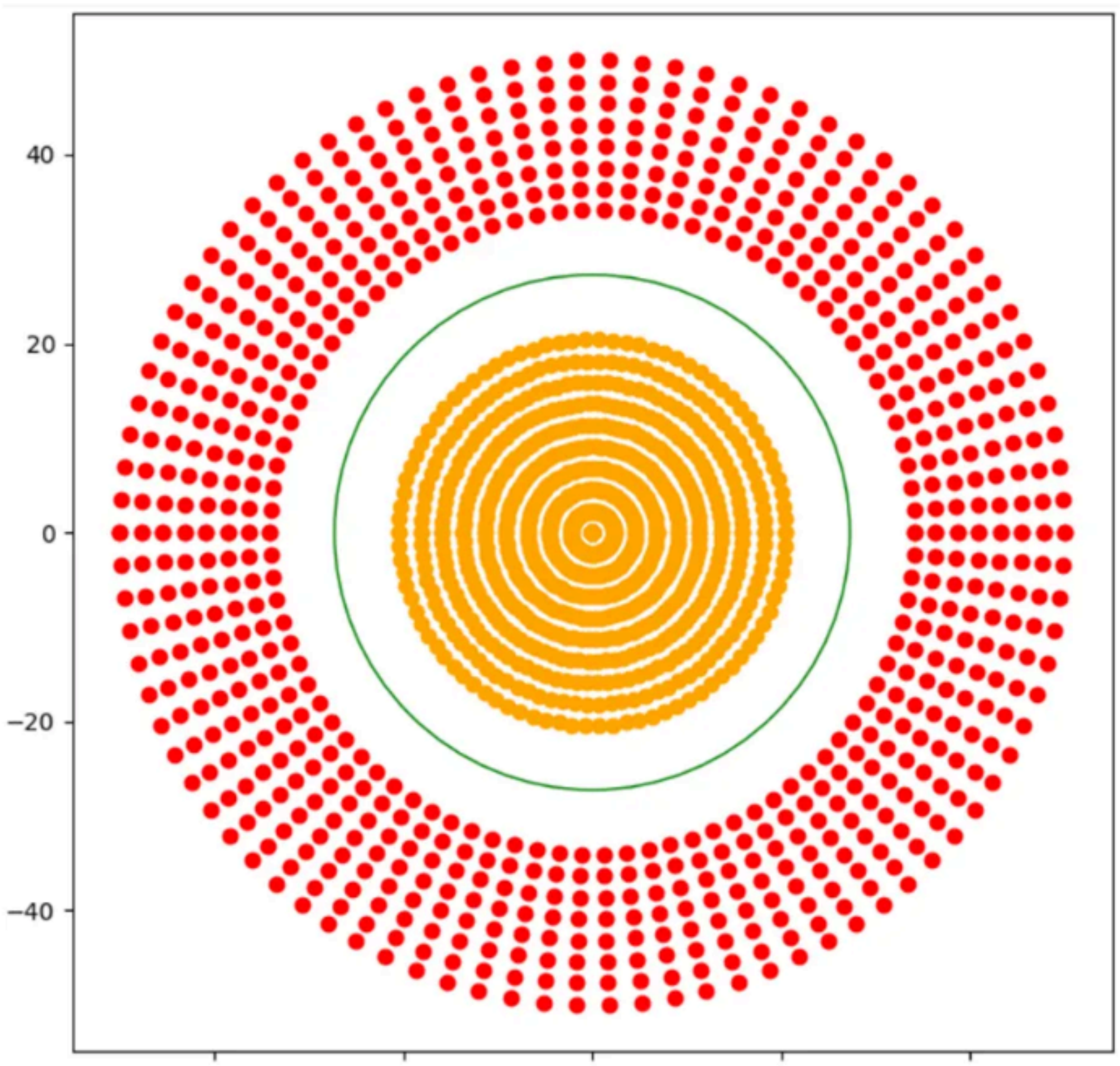
$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i (x_i \cdot x) + b \right)$$

Non-linear SVM and Kernel Trick

Kernel Trick:

For non-linear data, SVM uses the kernel trick to map the input features into a higher-dimensional space where the data becomes linearly separable.

LIKE FOR EXAMPLE WHERE HYPERPLANE IS NOT VERY EASY TO MAKE IN THE CASE OF THE BELOW EXAMPLE:

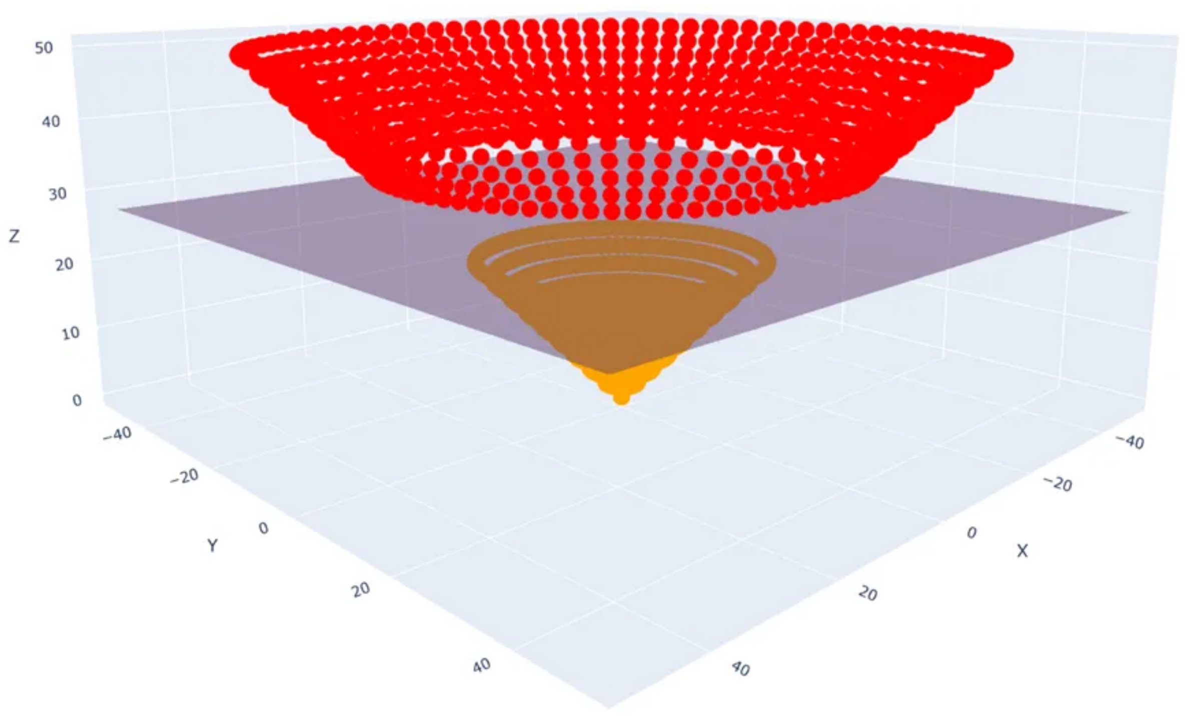


A dataset which is not linearly separable.

WE NEED TO ADD AN EXTRA DIMENSION TO CREATE THE GREEN HYPERPLANE :

What is the Kernel Trick?

SVM NOTES BY SOHAM



The kernel trick is a clever mathematical shortcut used in machine learning to handle data that isn't easily separated by a straight line. Instead of trying to draw a straight line through complicated data, the kernel trick lets us draw a line in a more complex, high-dimensional space without actually having to transform the data into that space.

Why is it Useful?

1. **Solving Non-linear Problems:** Some datasets can't be separated by a simple straight line. The kernel trick helps create complex boundaries that can separate such data effectively.

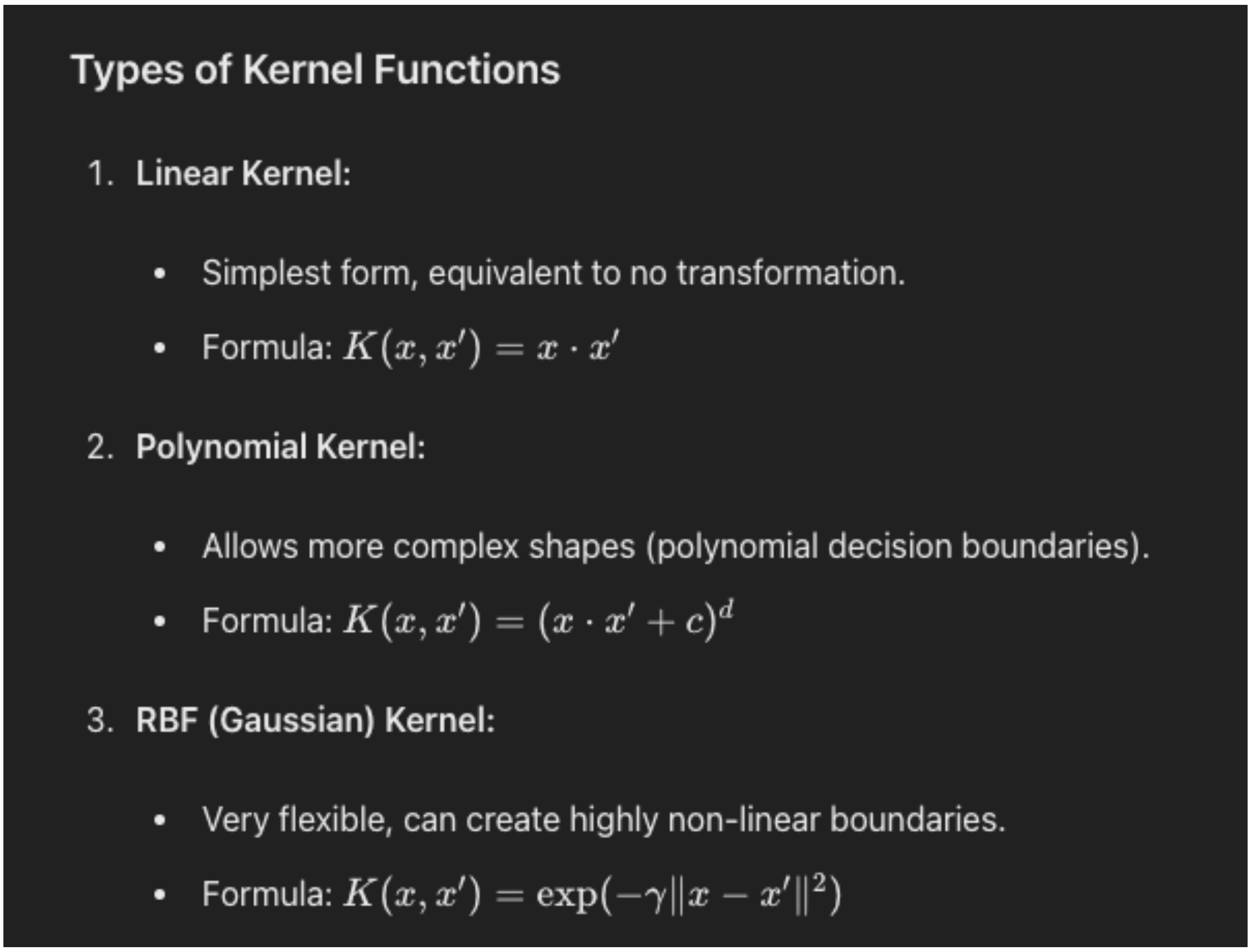
2. **Efficiency:** Directly transforming data into higher dimensions would be computationally intensive. The kernel trick allows us to work in high-dimensional spaces without the heavy computation, saving time and resources. It works by computing the similarities within dots by finding dot products between the images of all pairs of data points in this high-dimensional space using a kernel function.

How Does It Work?

1. **Original Space:** Imagine you have data points on a 2D plane that form two concentric circles. A straight line can't separate these points.

2. **High-dimensional Space:** If you could magically lift the points into a 3D space, suddenly a flat plane (like a sheet of paper) might be able to separate the points perfectly.

3. **Kernel Function:** Instead of actually lifting the points into 3D, you use a kernel function. This function calculates the "dot product" (a measure of similarity) of points as if they were in the higher-dimensional space, without ever needing to explicitly compute their positions in that space.



Types of Kernel Functions

1. **Linear Kernel:**
   - Simplest form, equivalent to no transformation.
   - Formula: $K(x, x') = x \cdot x'$
2. **Polynomial Kernel:**
   - Allows more complex shapes (polynomial decision boundaries).
   - Formula: $K(x, x') = (x \cdot x' + c)^d$
3. **RBF (Gaussian) Kernel:**
   - Very flexible, can create highly non-linear boundaries.
   - Formula: $K(x, x') = \exp(-\gamma \|x - x'\|^2)$

THIS IS THE END CONCLUDING HOW THE HYPERPLANE IS MADE ON BASIS OF THE LONGEST MARGIN.