

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Machine Learning Algorithms(16) — Support Vector Machine(SVM)



Kasun Dissanayake · [Follow](#)

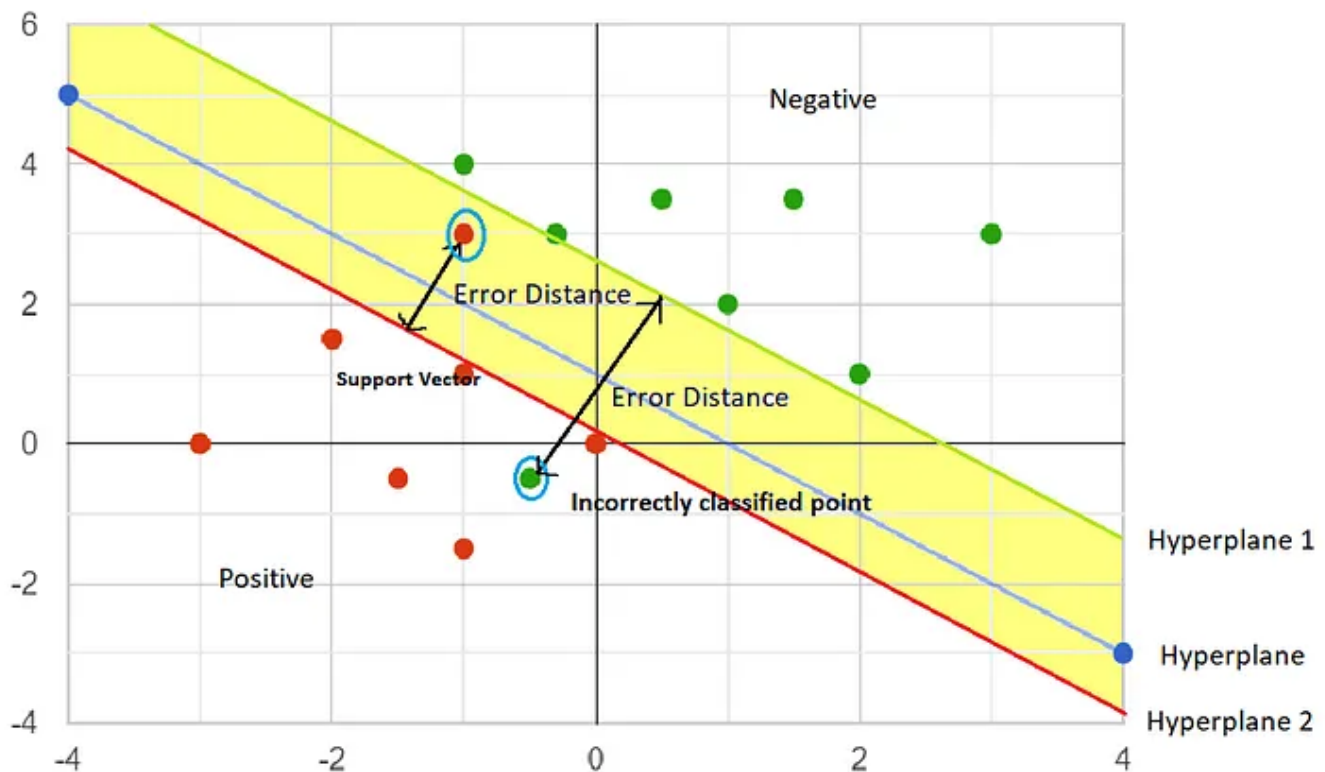
Published in Towards Dev · 12 min read · Feb 3, 2024



622



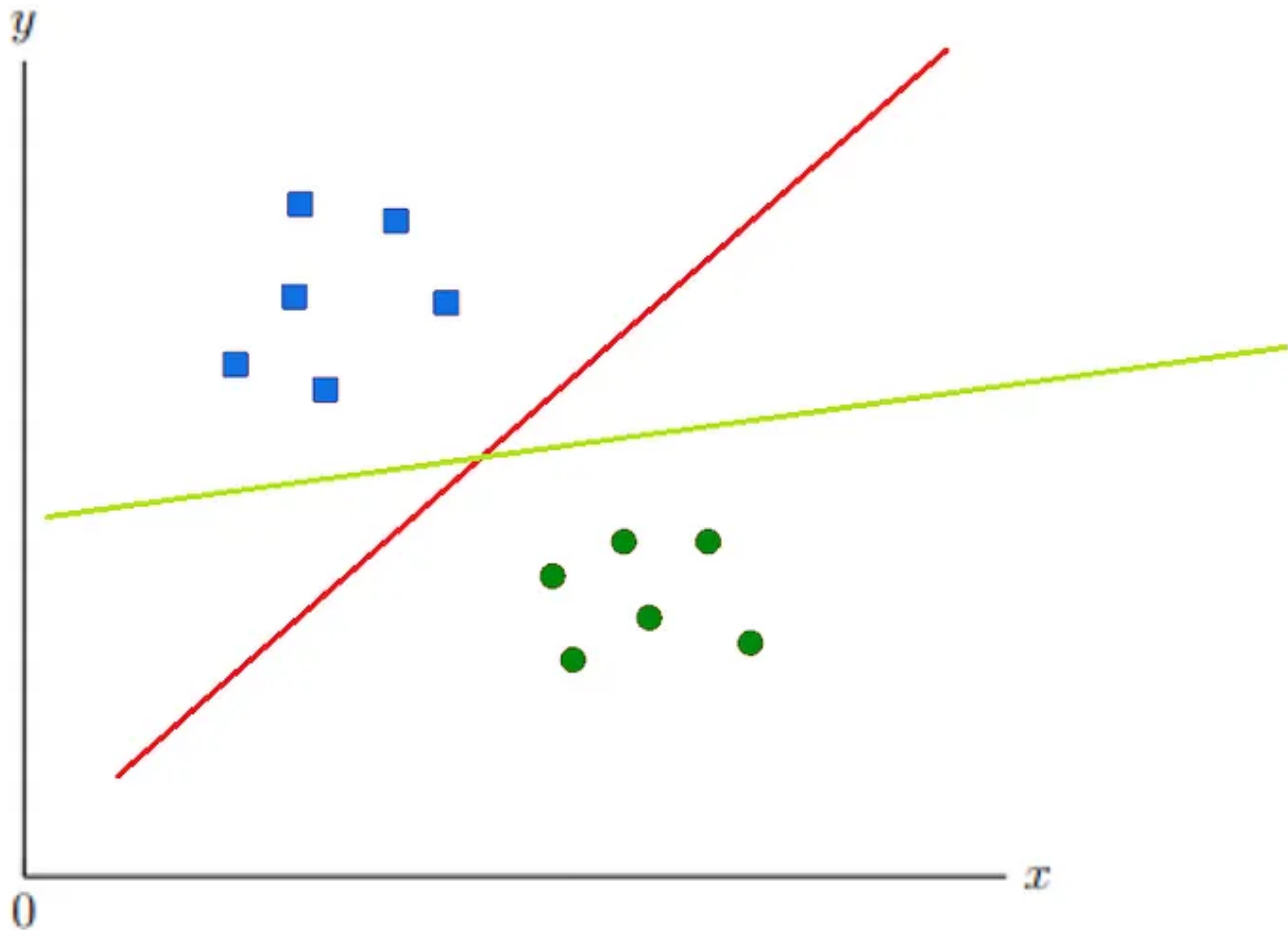
3



This article, delves into the topic of **Support Vector Machines(SVM)** in Machine Learning, covering the different types of SVM algorithms and how they function. SVM is a widely used **supervised machine learning algorithm** that can tackle classification and regression problems. To understand how SVM works, we first need to comprehend what supervised means — it involves providing label data as input to the algorithm.

Classification problems, where the target has a finite number of possibilities, are tackled with SVM. For instance, if you want to determine whether the mail is a scam, there are only two possibilities. On the other hand, regression problems deal with continuous target labels. For example, if you need to predict an employee's salary increase based on their performance, the salary increase would be continuous. SVM algorithms are generally utilized for classification challenges in machine learning.

The objective of the SVM algorithm is to **create a decision boundary or line** that can effectively segregate a given data set into different classes. Once the decision boundary is established, new examples can be classified into the appropriate classes with relative ease. In the SVM algorithm, this decision boundary is known as a **hyperplane**. The challenge then becomes drawing this hyperplane with precision.



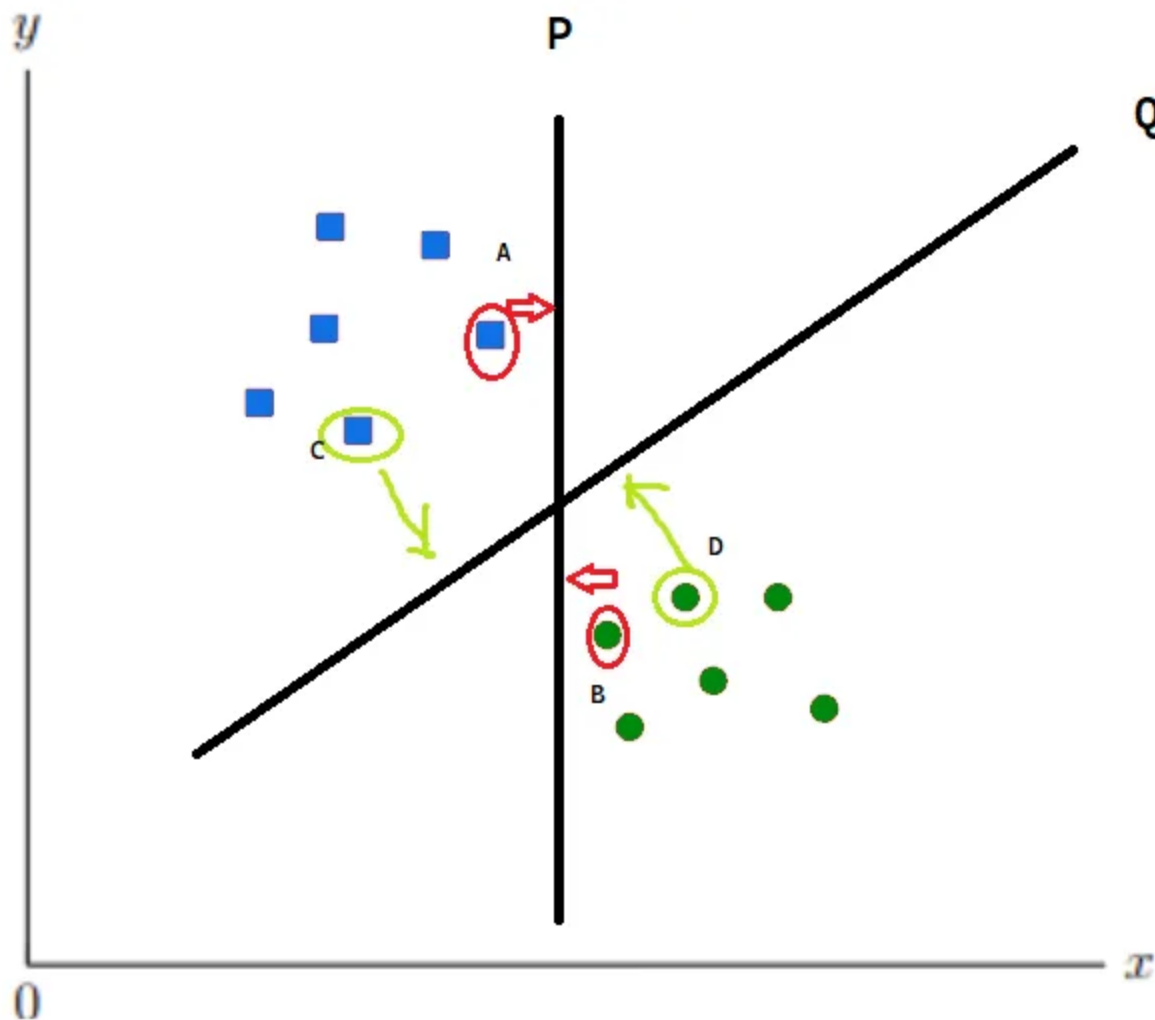
Hyperplanes

Suppose we are given a data set with two classes drawn in **blue and green colors**, respectively. The data set is **linearly separable**, meaning we can draw a straight line to separate the two classes. However, we need to determine which line is the **best fit or the optimal boundary**. In the given diagram, two lines are drawn in **red and green** colors, respectively. In this case, we need to draw all possible lines to determine the one with the best accuracy. This line is called the **Hyperplane**, whose dimension depends on the features present in the data set. If there are only two features, we can draw a straight line, but if there are more than two features, we need to draw a plane, which will be the **two-dimensional hyperplane** in this case.

To ensure the maximum margin, drawing the hyperplane is crucial. Later on, I will delve into the specifics of how to achieve this maximum margin.

When drawing the hyperplane we need to consider **finding the nearest points on both sides of the hyperplane**. Those points we called **Support Vectors**.

Support vectors are data points that are **closer to the hyperplane** and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the **margin**(we will learn this thing later) of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.



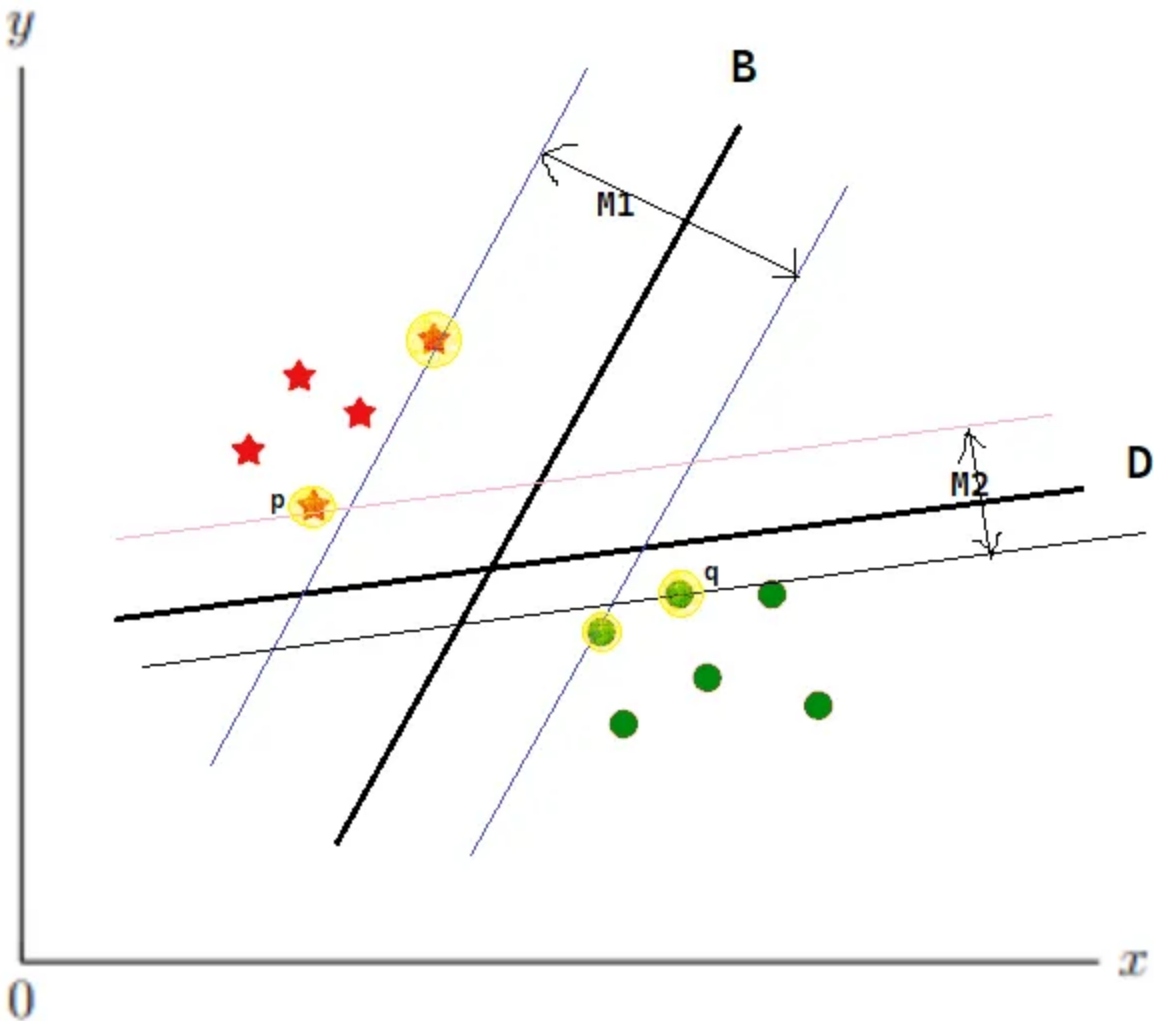
In this graph, A and B points are nearest for this P hyperplane. So A and B are referred to as **support vectors** to the P hyperplane in this case.

And C and B are the other nearest points to the Q hyperplane.

What are the different types of SVM?

- **Linear SVM** — Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as **Linear SVM classifier**.

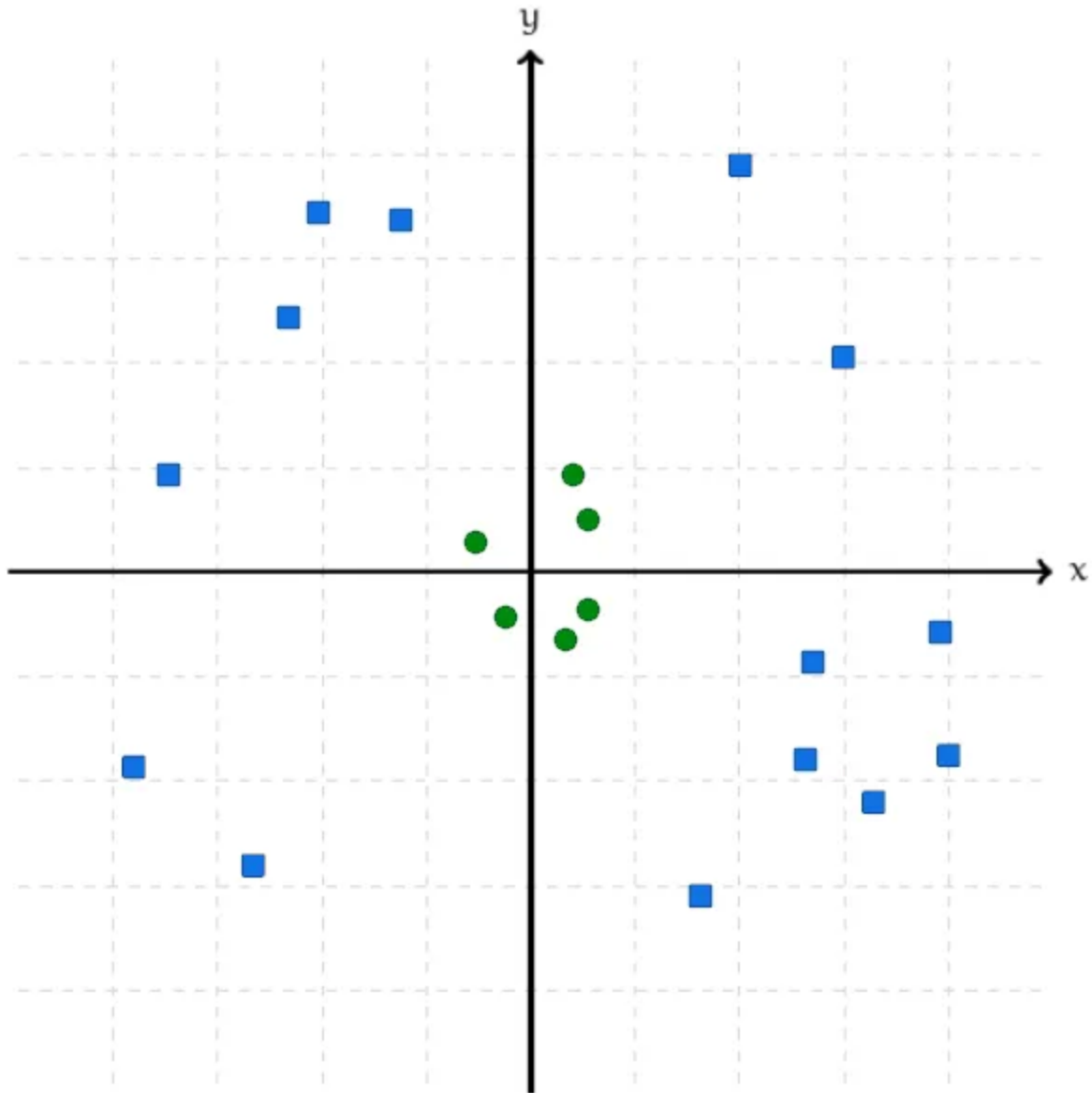
Suppose we have a dataset that has two classes(stars and dots) and the dataset has two features **x** and **y**. we can separate this dataset using **B** and **D** straight lines. When we draw multiple straight lines to separate the data which one has to be considered as the **hyperplane**. To understand we can use this example.



B and D hyperplane and find the maximum margin

Assume **D** as the one hyperplane and these **p** and **q** are the support vectors. So draw a *parallel line* to a hyperplane with the support of support vectors. And the **M2** distance we need to calculate. This distance we called *Margin*. Similarly, we need to do this to **B** hyperplane also. And calculate the margin **M1**. When comparing the **M1** and **M2**, the **M2 margin is smaller** when compared to the **M1**. So using this we can say **B is the best fit line(hyperplane)** which divides the data into two classes. We should do this calculation for all possible hyperplanes and the one who gives the **maximum margin** will be the hyperplane.

- **Non-linear SVM** — Non-linear SVM is used for non-linearly separated data, which means if *a dataset cannot be classified by using a straight line*, then such data is termed as **Non-linear data**, and the classifier is called a **Non-linear SVM classifier**.

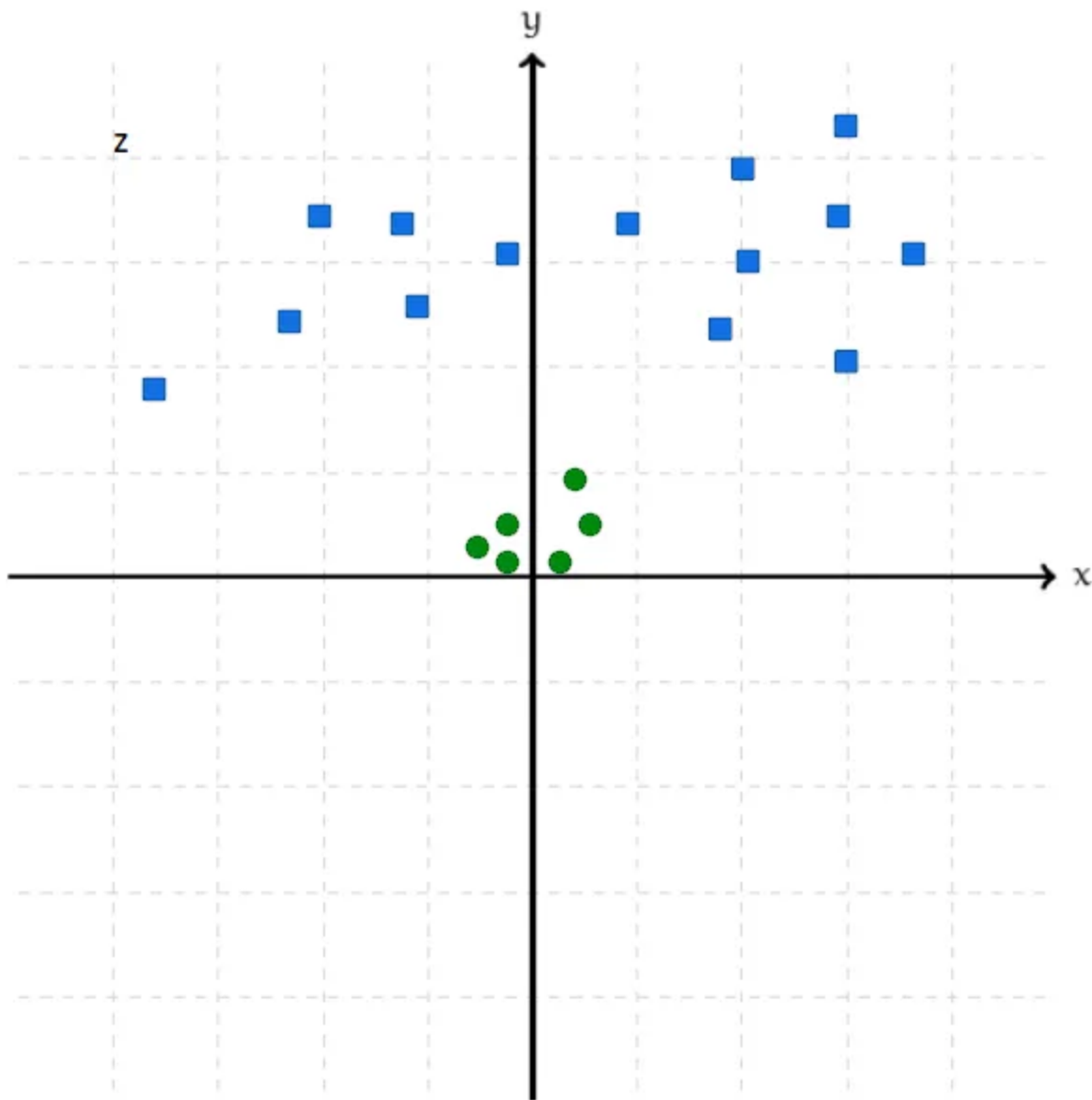


Here we are not able to draw a straight line to device these data into 2 classes. We need to convert particular data into linear data. For that, we can use a **mapping function**.

It can be calculated as:

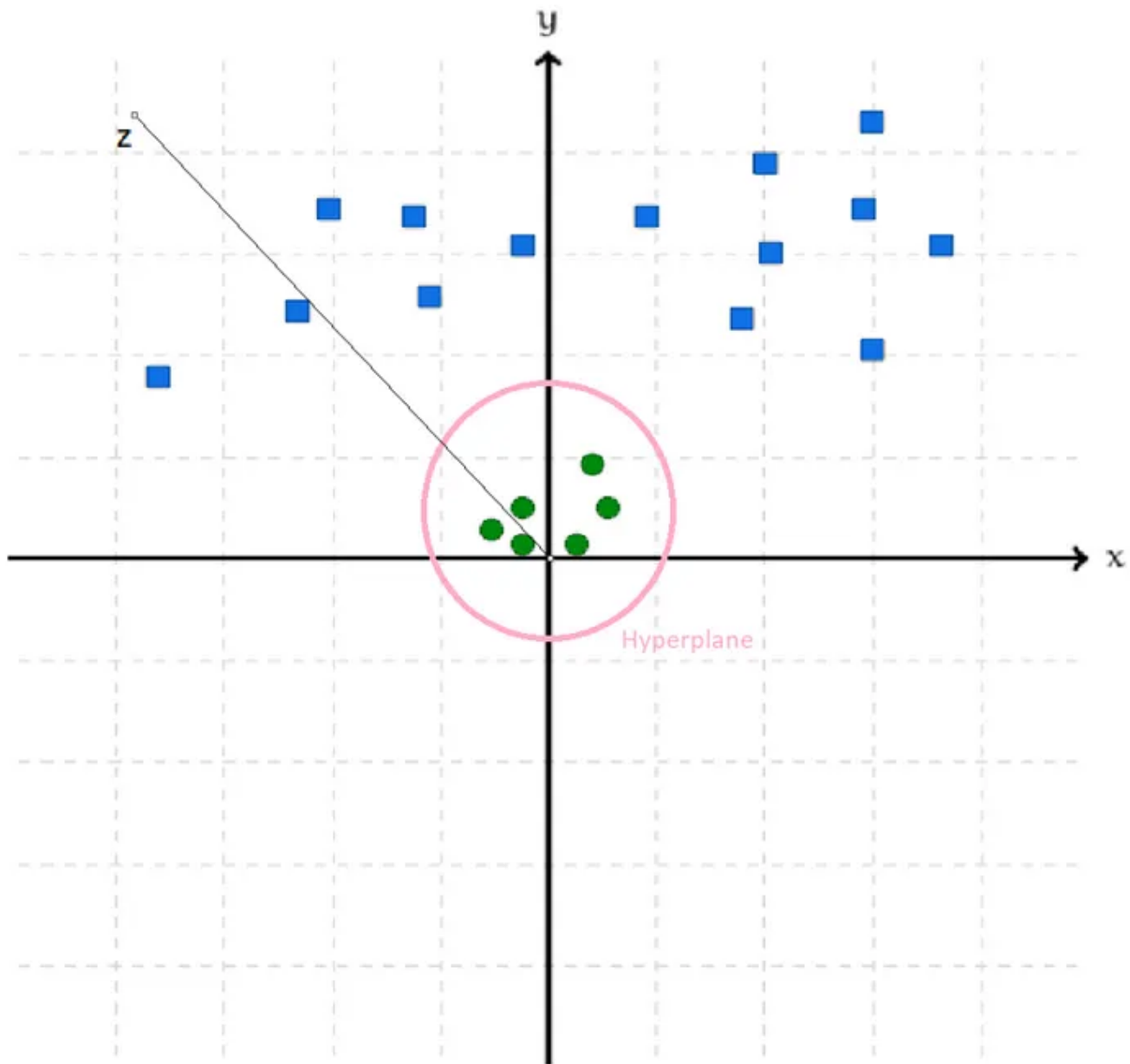
$$z = x^2 + y^2$$

Once we apply this all the data points will be like this. And we will get another axis called z .



The above depiction of data points is linearly separable and can be separated by a straight-line hyperplane. Now we can draw all possible lines that will separate data into two classes and we need to identify **one straight line that will give the maximum margin** and that will be considered as the best hyperplane.

This representation is in 3-D with a z-axis. In 2-D, the graph looks like this:-



Non-linear SVM hyperplane

This is what a non-linear SVM does! It takes the data points to a higher dimension to be linearly separable in that dimension, and then the algorithm classifies them.

Maths behind the Support Vector Machine Classifier

Let's say we have two features **X1** and **X2**. We are trying to plot the data points in these two features in a graph. Let's say we have two points **P1** and **P2**.

$$P1 - (-3, 0) \quad P2 - (3, 3)$$

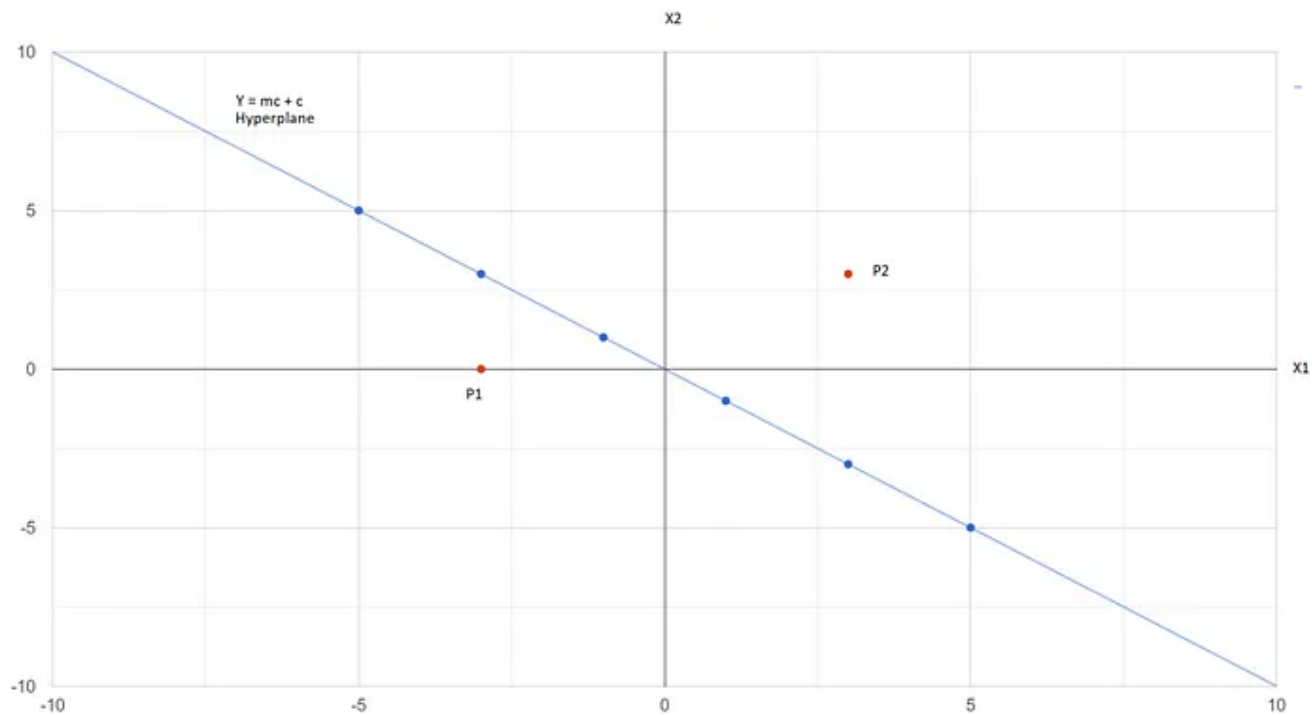
To draw a hyperplane that can separate these two points is a **straight line** because this is a 2-dimensional space. Let's take this line(blue color line) which will act as our hyperplane for our supporting machine classifier. We know that the equation of a straight line is,

$$y = mx + c$$

In the below scatter plot diagram,

Slope = **-1** (because of the negative slope)

Intercept = **0** (because **this** line go through the origin of the graph)



Hyperplane

I am going to take these two parameters **slop and Intercept** and store them in a variable called **W**. **W** is the *weight* vector that contains the coefficients of **x** and **y**.

For example, for a line given in the diagram

$$-x - y = 0,$$

W → parameter of the line

$$(-1, -1)$$

We can form the equation like this ,

$$-x - y + b = 0,$$

$$W = [-1, -1], X = [x, y] \text{ and bias } b = 0$$

We can represent the same equation in the following form:

$$f(x) = W \cdot \text{Transpose} \cdot X$$

- *Let's take the $P1(-3, 0)$ point and we try to find the corresponding $W \cdot \text{Transpose value}$ for that.*

I do a simple matrix multiplication. For that, I need W^t (transpose of w) and multiply it with x to find the $W^t \cdot X$ value.

Why do we use W^t value?

We are trying to project the $P1$ point into the Hyperplane. For that we need to use matrix multiplication(Projection of vectors).

```
W = [-1, -1]
Wt = [-1 ]
      [-1 ]

X = [x, y] = [-3  0]

Wt . X =  [-1 ]
          [-1 ] . [-3  0 ]

Wt . x  = +3 (Positive value)
```

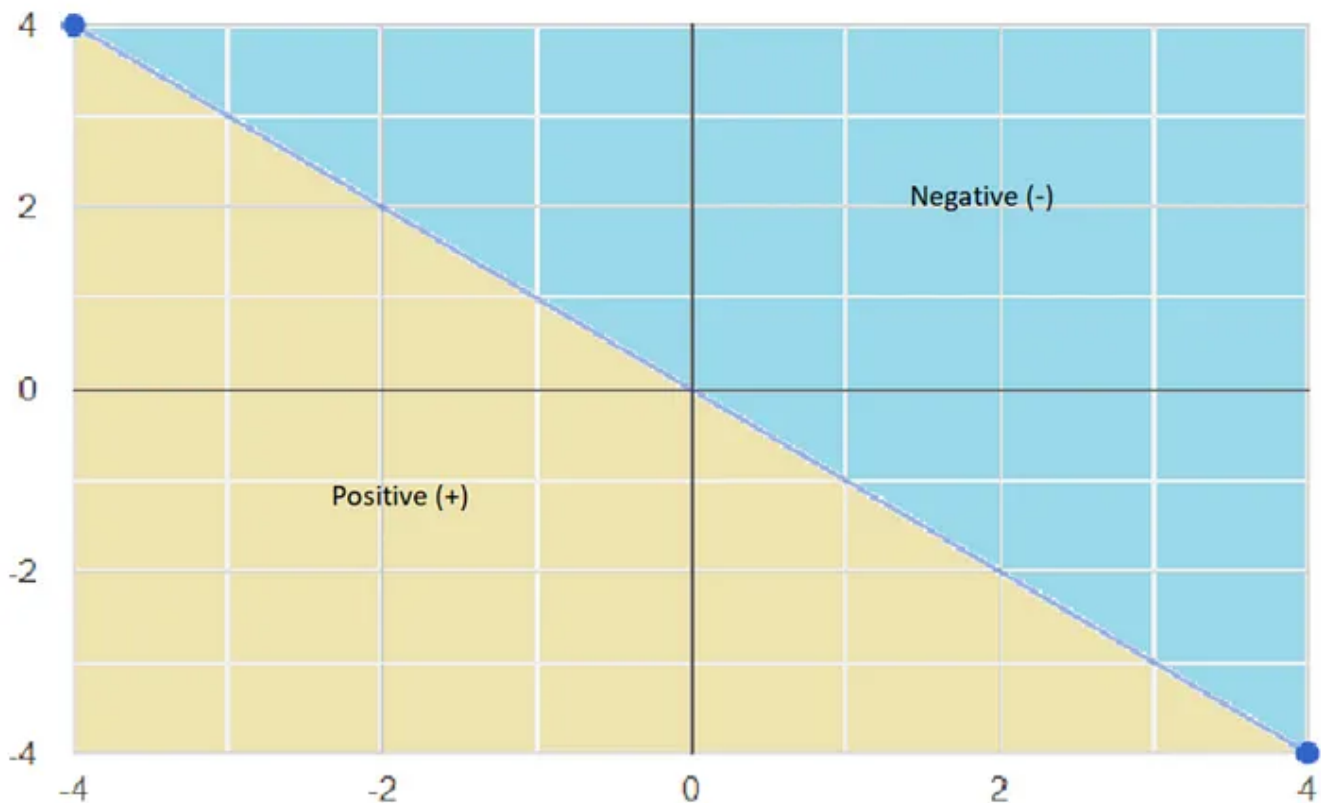
Now you can see the value of multiplication is 3. It is a **positive number**. So we can say that all the points that lie on the left side of the hyperplane will have positive values.

- *Let's take the $P2(3, 3)$ point and we try to find the corresponding y value for that.*

```
Wt . x  = [-1 ]
          [-1 ] . [3  3]
```

$$Wt \cdot x = -6 \text{ (Negative value)}$$

Now you can say, for all the points that lie on the left side of the hyperplane $Wt \cdot x$ value will be Positive and For all the points that lie on the right side of the hyperplane $Wt \cdot x$ value will be Negative.



Clasity data points to Negative and Positive

This is how our support vector machine classifier will try to classify data points. Wt will act as a **Label**.

But in some cases, our hyperplane does not pass through the origin and the equation will be $f(x) = W \cdot \text{Transpose} \cdot X + B$,

//Where W is the weight vector which contains the coefficients of x and y .

For example, for a line given by $x - 2y + 3 = 0$,

$W = [1, -2]$, $X = [x, y]$ and bias $b = 3$

//Where W is the weight vector containing the coefficients of features X , Y , and

For example, for the equation of a line, $2x + 3y + 4z + 3 = 0$,

$W = [2, 3, 4]$, $X = [x, y, z]$ and bias, $B = 3$

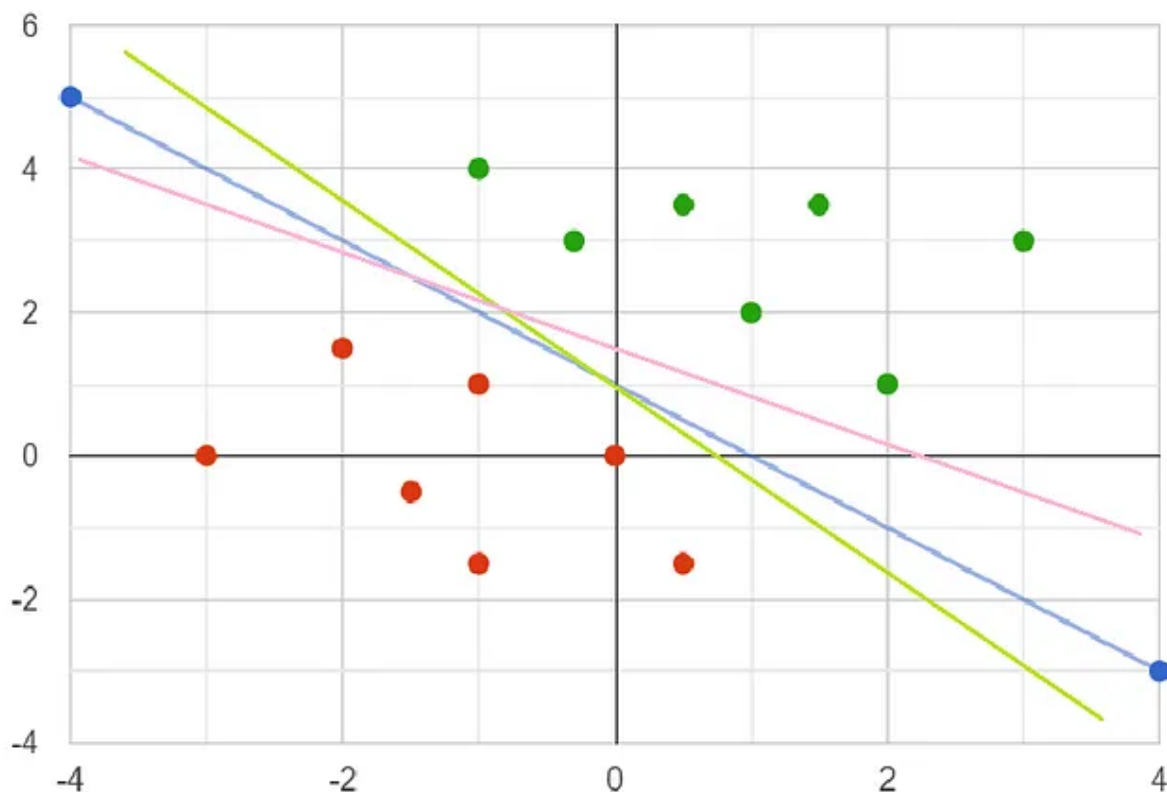
//Where W and X are vectors of dimension ' n ' which is the number of features of

// W is the weight vector that contains the coefficients of corresponding feature

// Also, it is always normal to the given plane and thus gives an idea about the

// B is the bias which describes the position of the plane in an n -dimensional hy

You know that we can draw any number of hyperplanes to divide data into classes. But we need to draw the **best hyperplane**. For that, we can use the **margin and the support vectors**. To make a good prediction for the best hyperplane your margin value should be maximum.



Many Maginal Lines can be drawn

Optimization for a Maximum margin

To find the marginal distance we can find the support vectors and create the parallel lines to the hyperplane.

To get our optimization function, there are a few constraints to consider. That constraint is that **“We’ll calculate the distance (d) in such a way that no positive or negative point can cross the margin line”**. Let’s write these constraints mathematically:

```
//For all the Green points
W.Transpose. X1 + B <= -1

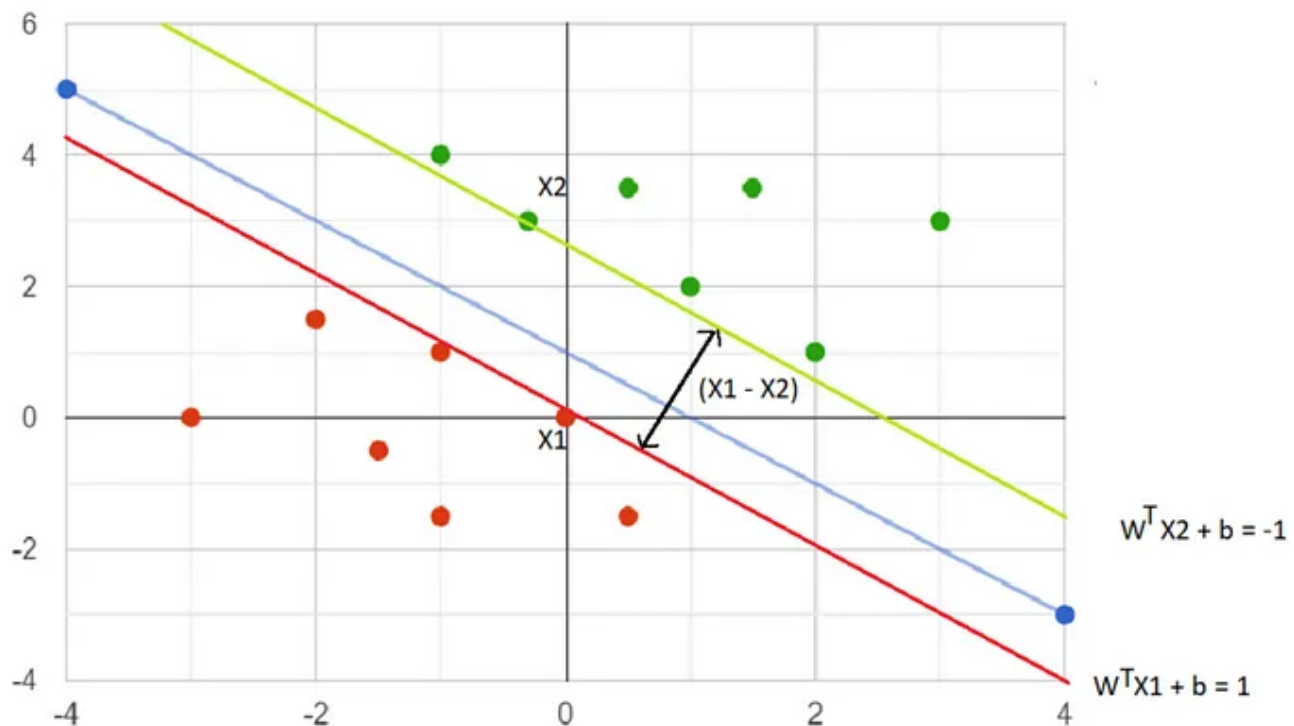
//For all the Red points
W.Transpose. X1 + B >= 1
```

Rather than taking 2 constraints forward, we’ll now try to simplify these two constraints into 1. We assume that negative classes have $y=-1$ and positive classes have $y=1$.

We can say that for every point to be correctly classified this condition should always be true:

```
 $y_i (W \cdot X_1 + B) \geq 1$ 
```

Suppose an X_1 red point is correctly classified that means it will follow $W \cdot \text{Transpose} \cdot X_1 + B$, if we multiply this with $y=1$ we get this same equation mentioned above. Similarly, if we do this with a green point with $y=-1$ we will again get this equation. Hence, we can say that we need to maximize (d) such that this constraint holds.



We will take 2 support vectors, 1 from the negative class (Green— X_1) and 2nd from the positive class (Red— X_2). The distance between these two vectors X_1 and X_2 will be $(X_1 - X_2)$ vector. What we need is the shortest distance between these two points which can be found using $f(x) = W \cdot \text{Transpose} \cdot X + B$.

Red support vector = X_1
 Green Support Vector = X_2

Marginal Distance (d) = $X_1 - X_2$
 *** Now we need to increase this marginal distance value.


```

// And the Left side(Red) of the Hyperplane we can get positive numbers so that
W.Transpose. X1 + B <= -1

// And the Right side(Green) of the Hyperplane we can get negative numbers so th
W.Transpose. X2 + B >= 1

//Let's take this W.Transpose. X1 + B is equal to 1 possibility,
X1-> W.Transpose. X1 + B = 1
X1 ->W.Transpose. X1 + B - 1 = 0
X2 ->W.Transpose. X2 + B = -1
X2 ->W.Transpose. X2 + B + 1 = 0

Finding the Marginal Distance
=====
X1 = X2

W.Transpose. X1 + B -1 - (W.Transpose. X2 + B + 1)
W.Transpose. X1 - W.Transpose. X2 - 2

W.Transpose. X1 - W.Transpose. X2 = 2
//Since W.Transpose is a vector we can divide it by ||w|| which is magnitude of
W.Transpose. X1 - W.Transpose. X2 = 2
W.Transpose. (X1 - X2) = 2
//Divide by ||w||
W.Transpose. (X1 - X2) / ||W|| = 2 / ||W||
(X1 - X2) = 2 / ||W||

-----
Marginal Distance = (X1 - X2) = Max [ 2 / ||W|| ]
                  such that, yi(W.Transpose. X1 + B) >= 1
-----

```

We have now found our optimization function but there is a catch here we don't find this type of perfectly linearly separable data in the industry, there is hardly any case we get this type of data and hence we fail to use this condition we proved here. The type of problem that we just studied is called **Hard Margin SVM** now we shall study soft margin which is similar to this but there are a few more interesting tricks we use in **Soft Margin SVM**.

Soft Margin SVM

Real-world data sets are rarely perfectly linearly separable, making it difficult to apply the trick demonstrated in the previously shown data sets. In these cases, support vector machines become crucial in machine learning.

Open in app ↗



Search

Write



equation, we modify it to allow for a few points to be classified incorrectly.

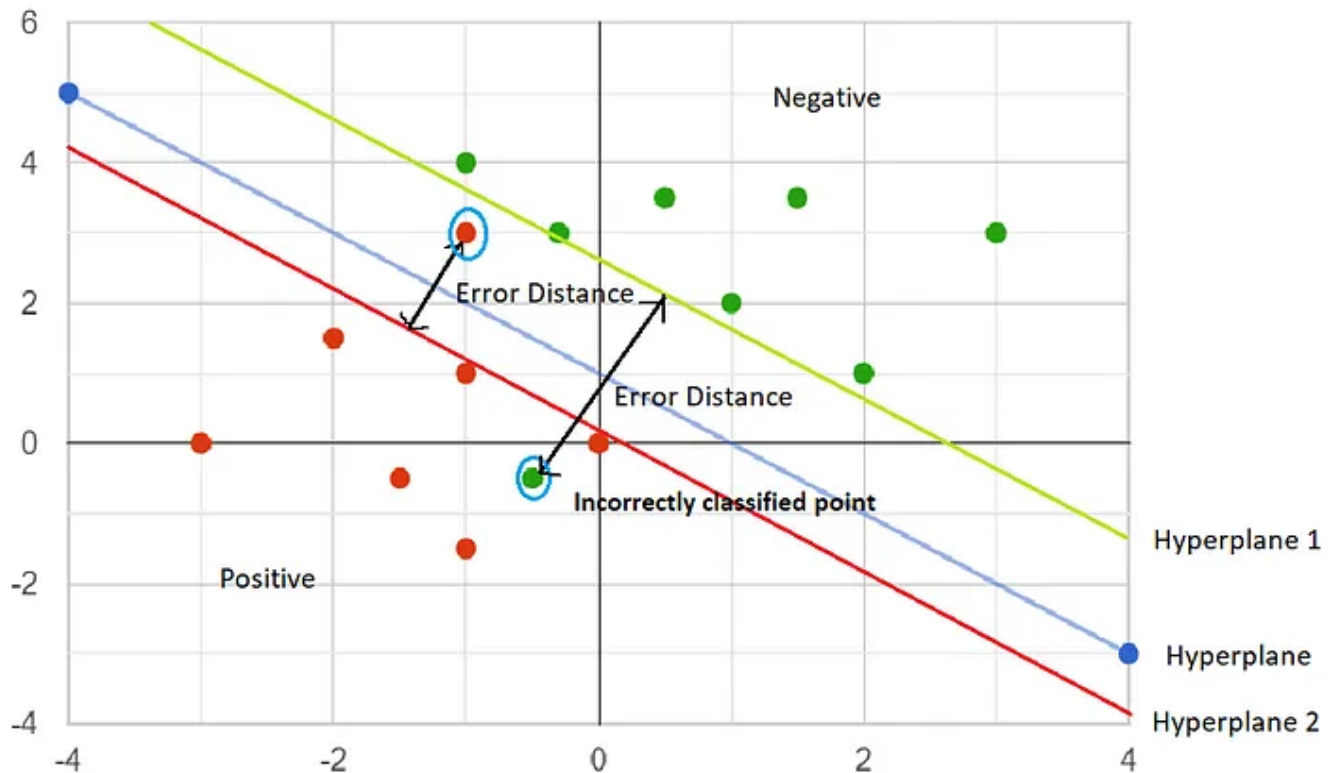
We can also $\max[f(x)]$ be written as $\min[1/f(x)]$, it is common practice to minimize a cost function for optimization problems; therefore, we can invert the function

$$(X1 - X2) = \text{Min} [||W|| / 2]$$

such that, $y_i(W \cdot \text{Transpose} \cdot X1 + B) \geq 1$

//To make a soft margin equation we add 2 more terms to this equation which is E
 $\text{Min} [||W|| / 2] + c \cdot \sum_{i=1}^n E_i$

For all the correctly classified points, our error will be equal to 0. For all the incorrectly classified points, the **error(E)** is simply the distance of that particular point from its correct hyperplane. That means if we see the wrongly classified green points, the value of error will be the distance of these points from hyperplane 1, and for only classified red points, the error will be that point from hyperplane 2.



$\text{Min} [||W|| / 2] + \text{Classification Error}$
 $\text{Min} [||W|| / 2] + c. \quad i = 1 \rightarrow n \quad \Sigma E$

c = Number of incorrectly classified points

So now we can say that our that are,
 SVM Error = Margin Error + Classification Error

//The higher the margin, the lower would-be margin error, and vice versa.

We'll **increase 'c'** to decrease **Classification Error** but if you want that your margin should be maximized then the value of '**c**' should be **minimized**. That's why '**c**' is a **hyperparameter** and we find the optimal value of '**c**'.

This is all about SVM and I hope this post helped in clarifying stuff about SVM. See you in another article.

Thank you!

Svm

Hyperplane

Support Vector Machine

Classification

Machine Learning



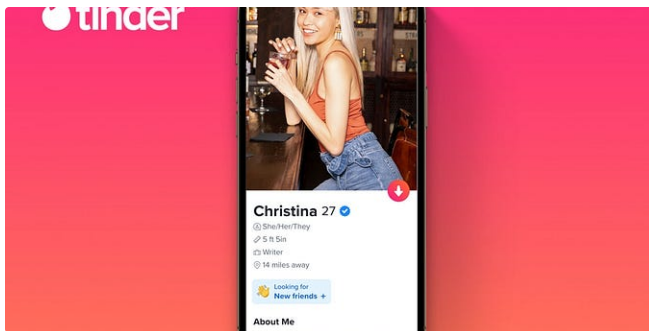
Written by Kasun Dissanayake

Follow

2.3K Followers · Writer for Towards Dev

Senior Software Engineer at IFS R & D International || Former Software Engineer at Pearson Lanka || Former Associate Software Engineer at hSenid Mobile

More from Kasun Dissanayake and Towards Dev



Kasun Dissanayake



Enigma of the Stack in Towards Dev

Tinder — Fully explained System Design and Architecture

In this article, I am going to explain about the interesting topic you all want to know. That is...

16 min read · Nov 14, 2023



542



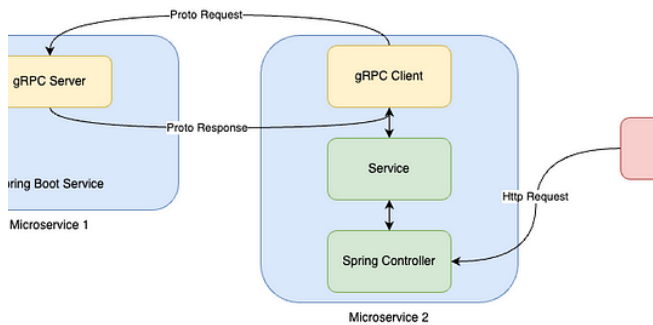
1



459



6

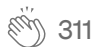


M K Pavan Kumar in Towards Dev

Building Scalable Microservices with gRPC, Spring Boot, and Maven

This article stands as a very good guide to understand and implement gRPC concepts...

8 min read · Jan 20, 2024



311



2



194



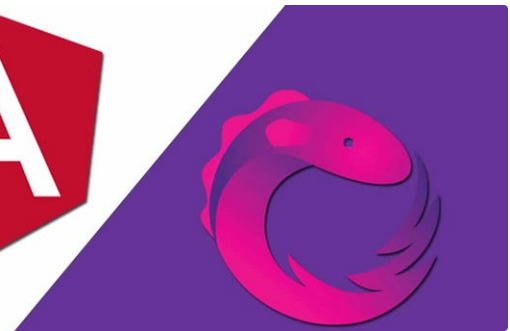
1



12 Top Free and Fun APIs Every Developer Needs for Side Projects

Elevate Your Side Projects with Versatile and Engaging APIs

★ · 7 min read · Jan 11, 2024

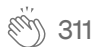


Kasun Dissanayake in Nerd For Tech

What are Observables in Angular?

In this tutorial, I am going to explain What is Observable and Where/When we use it. We...

6 min read · May 30, 2022



311



2



194



1



See all from Kasun Dissanayake

See all from Towards Dev

Recommended from Medium




 Jason Roell

Ultimate Python Cheat Sheet: Practical Python For Everyday...

This Cheat Sheet was born out of necessity. Recently, I was tasked with diving into a new...

33 min read · Jan 30, 2024

 2.2K  28  

 Tessa Rowan

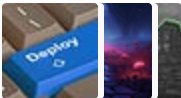
How a Simple Countdown Timer Website Making \$10,000 Per...

From Idea to Income

🌟 · 4 min read · Feb 10, 2024

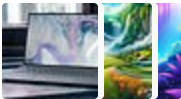
 8.2K  92  

Lists



Predictive Modeling w/ Python

20 stories · 937 saves



Natural Language Processing

1225 stories · 709 saves



Practical Guides to Machine Learning

10 stories · 1101 saves



The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 315 saves



Liu Zuo Lin in Level Up Coding

7 Things I Wish I Knew Earlier About Python Classes

Day 90 of experimenting with video content:

★ · 5 min read · Feb 17, 2024



1.3K



7



Long Nguyen

Building a Recurrent Neural Network From Scratch

In this blog post, we will explore Recurrent Neural Networks (RNNs) and the...

14 min read · Jan 28, 2024



77



Kelvin Lu in Towards AI

Full-Stack Data Scientist?

Research of the emerging full-stack data scientists, data science engineering...

9 min read · Feb 11, 2024



1.1K



10



See more recommendations