



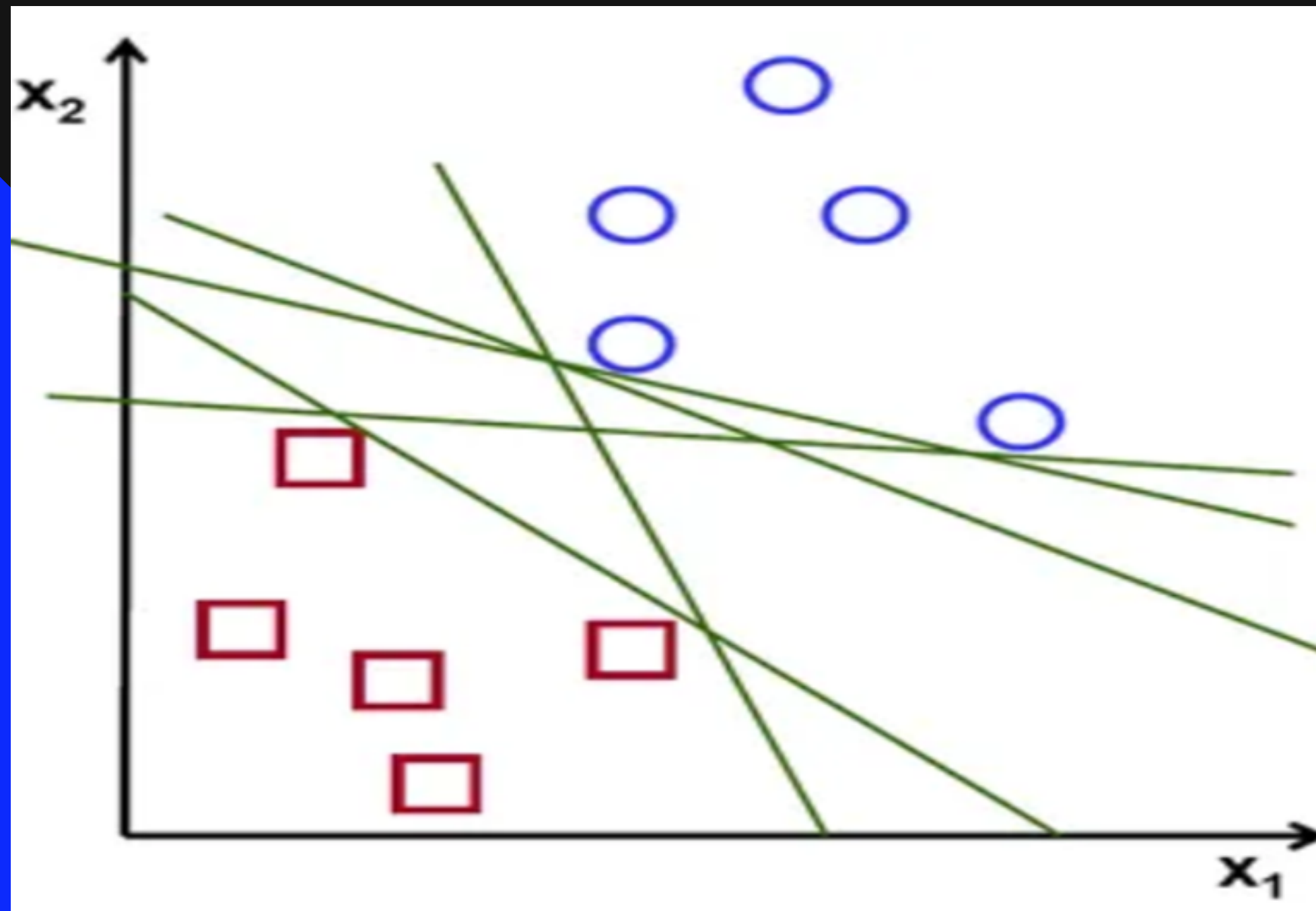
# Support Vector Machines



# what is SVM?

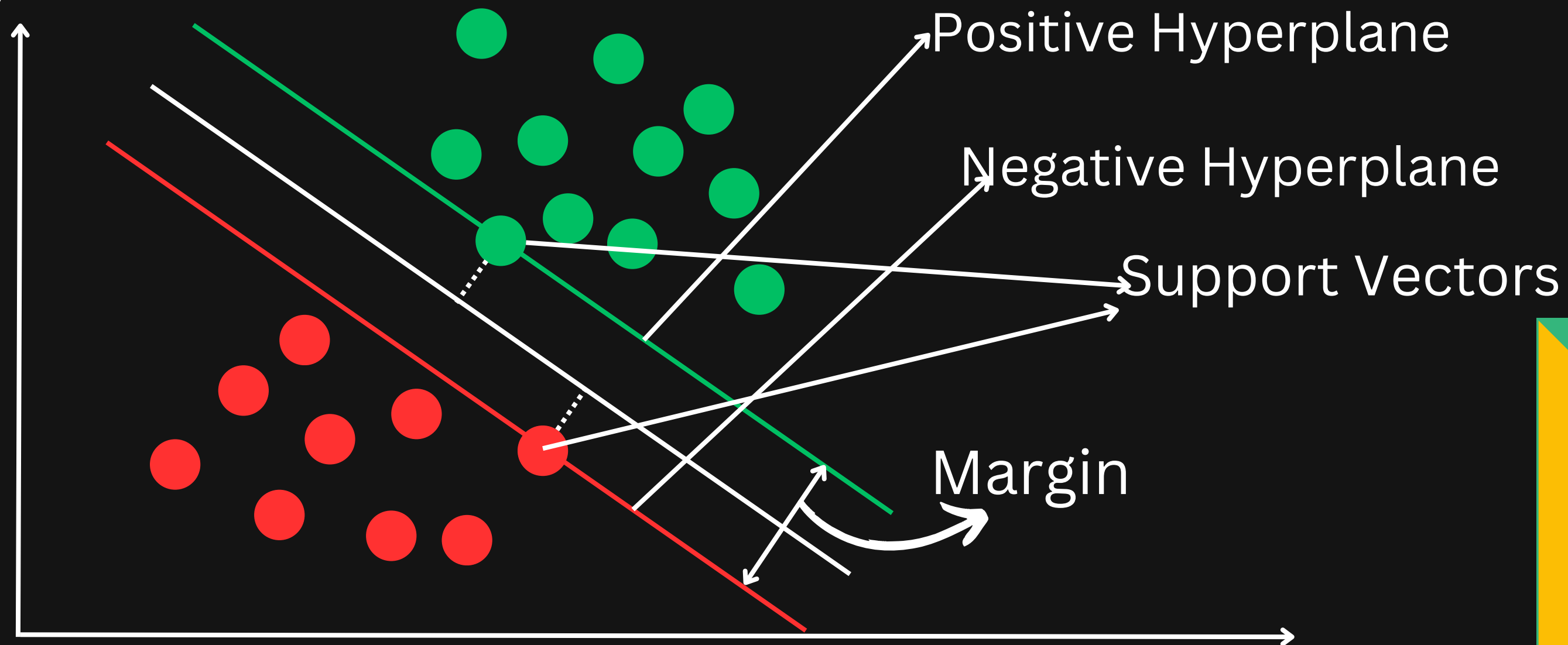
Support Vector Machine (SVM) is a powerful machine supervised learning algorithm used for linear or nonlinear classification, regression, and even outlier detection tasks.

we try to find a hyperplane that best separates the two classes.

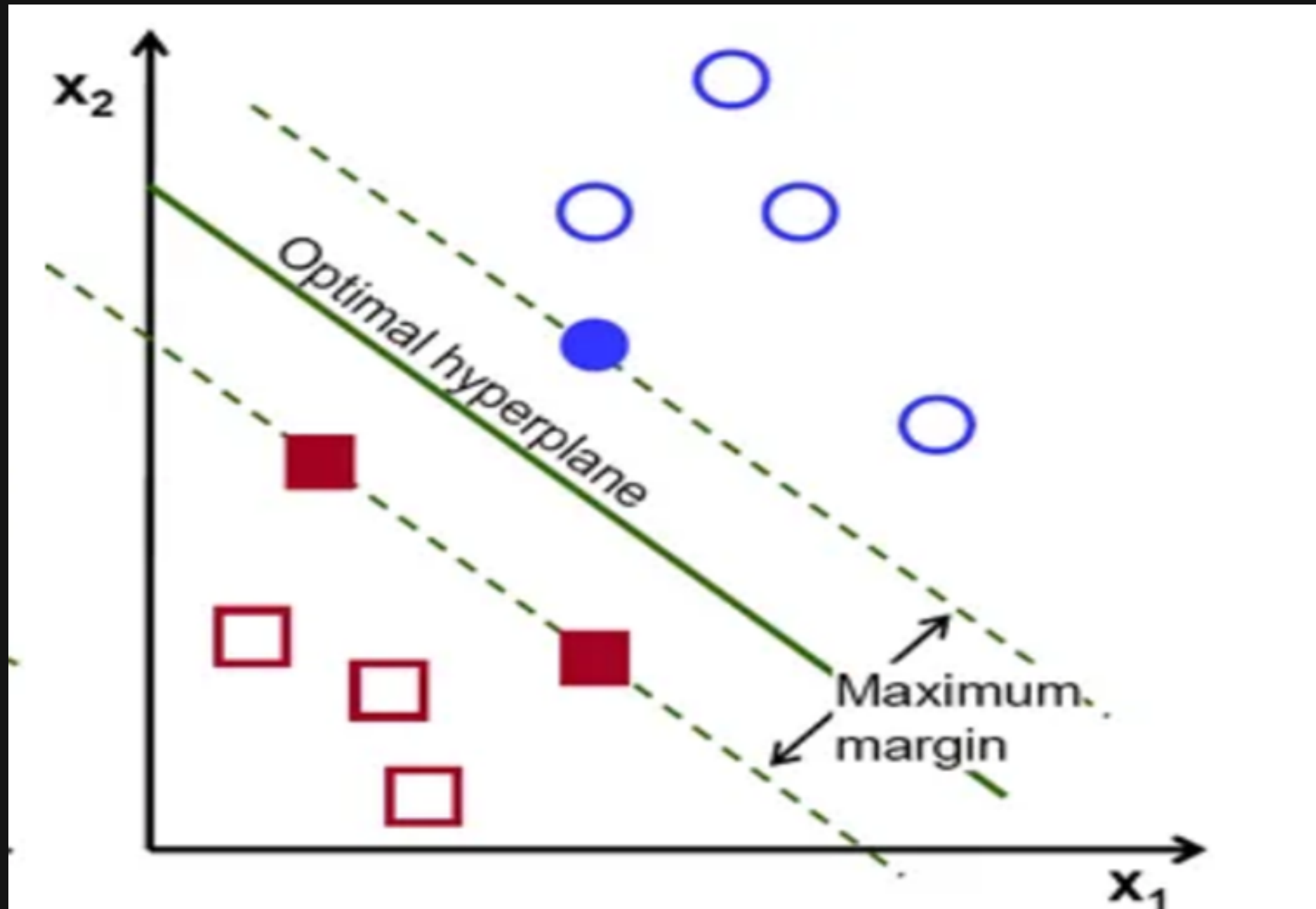


**Support Vectors:** These are the points that are closest to the hyperplane. A separating line will be defined with the help of these data points.

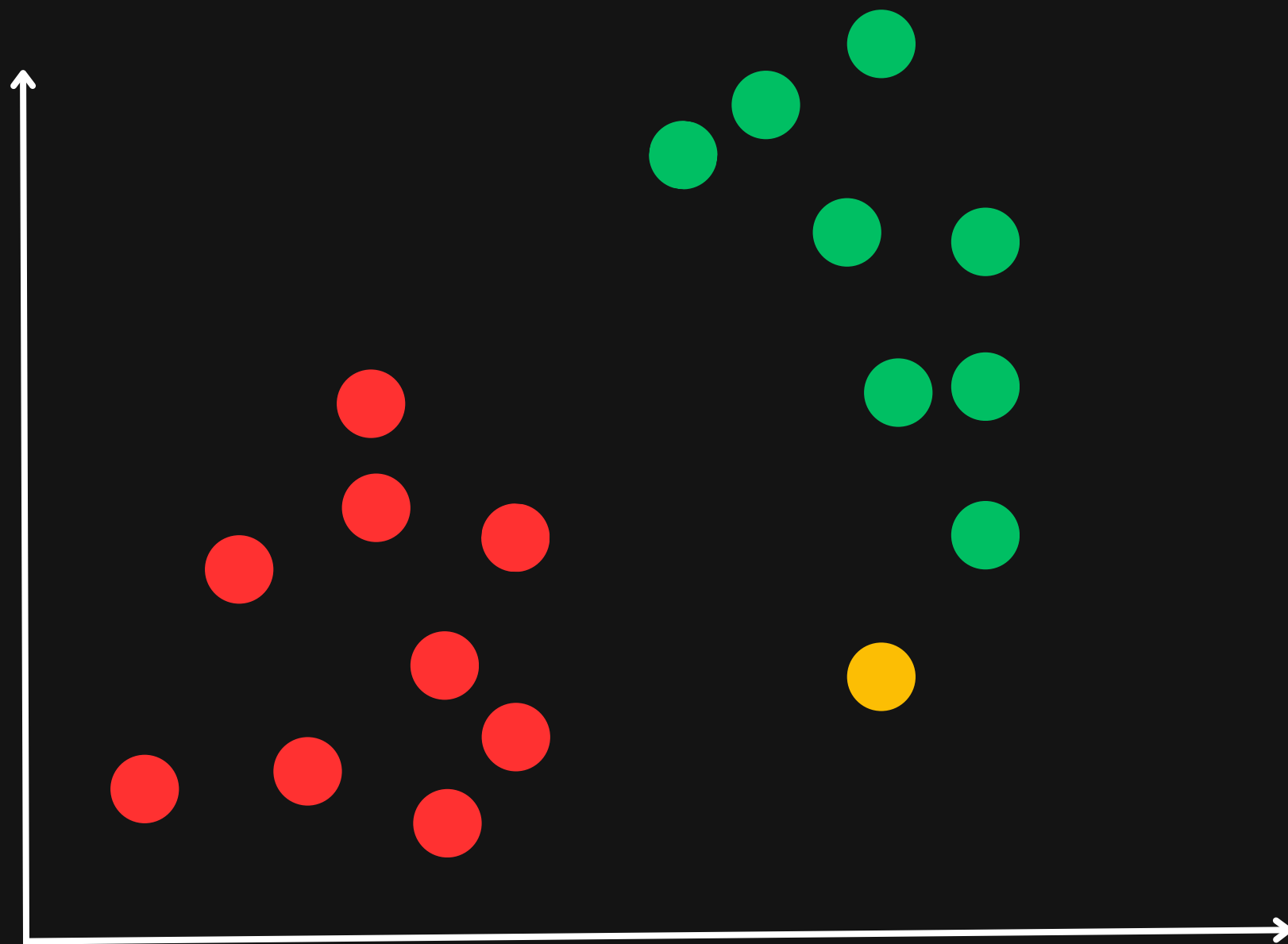
**Margin:** it is the distance between the hyperplane and the support vectors. In SVM large margin is considered a good margin. There are two types of margins hard margin and soft margin.

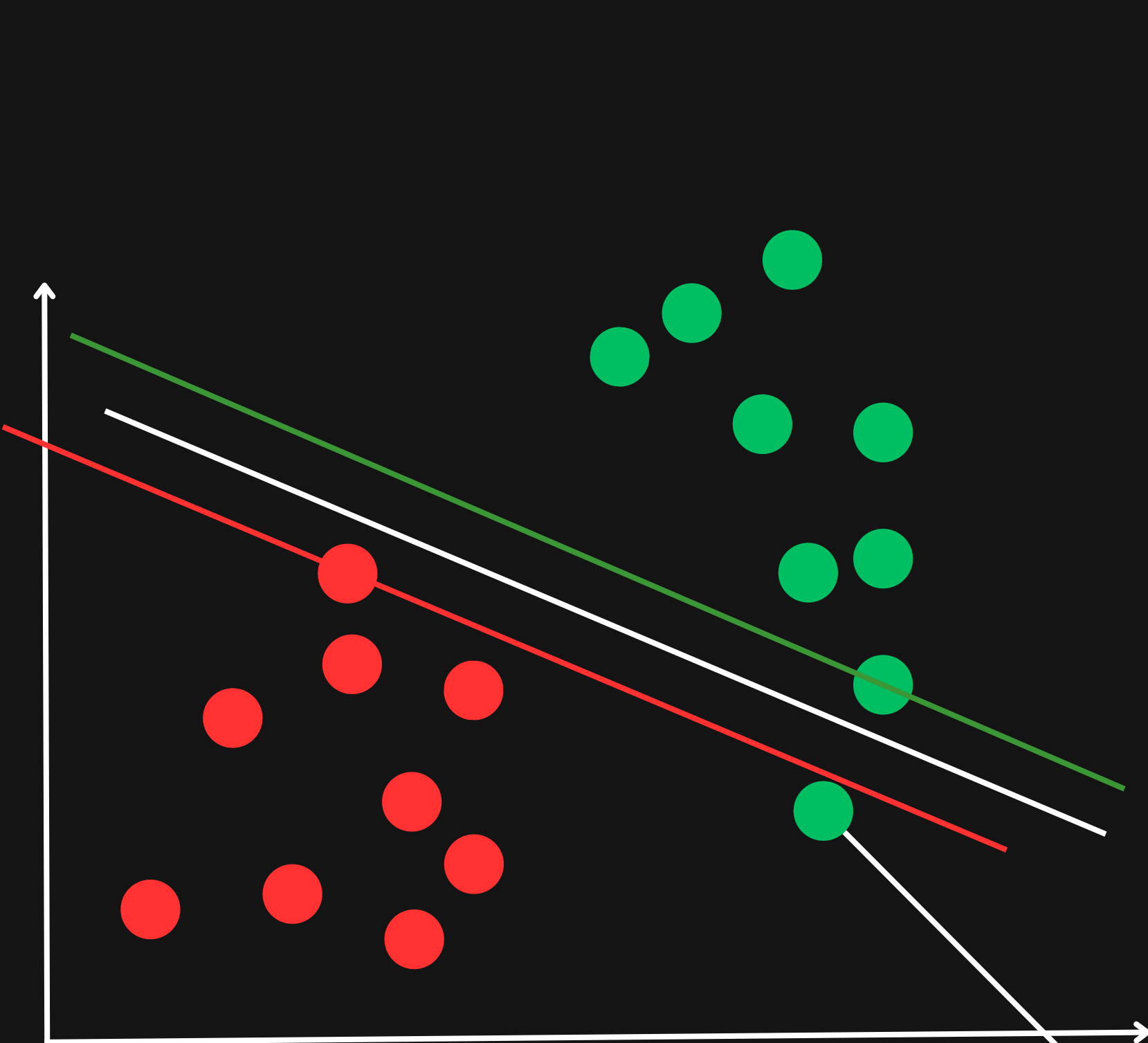


The objective is to find a hyperplane with the maximum margin

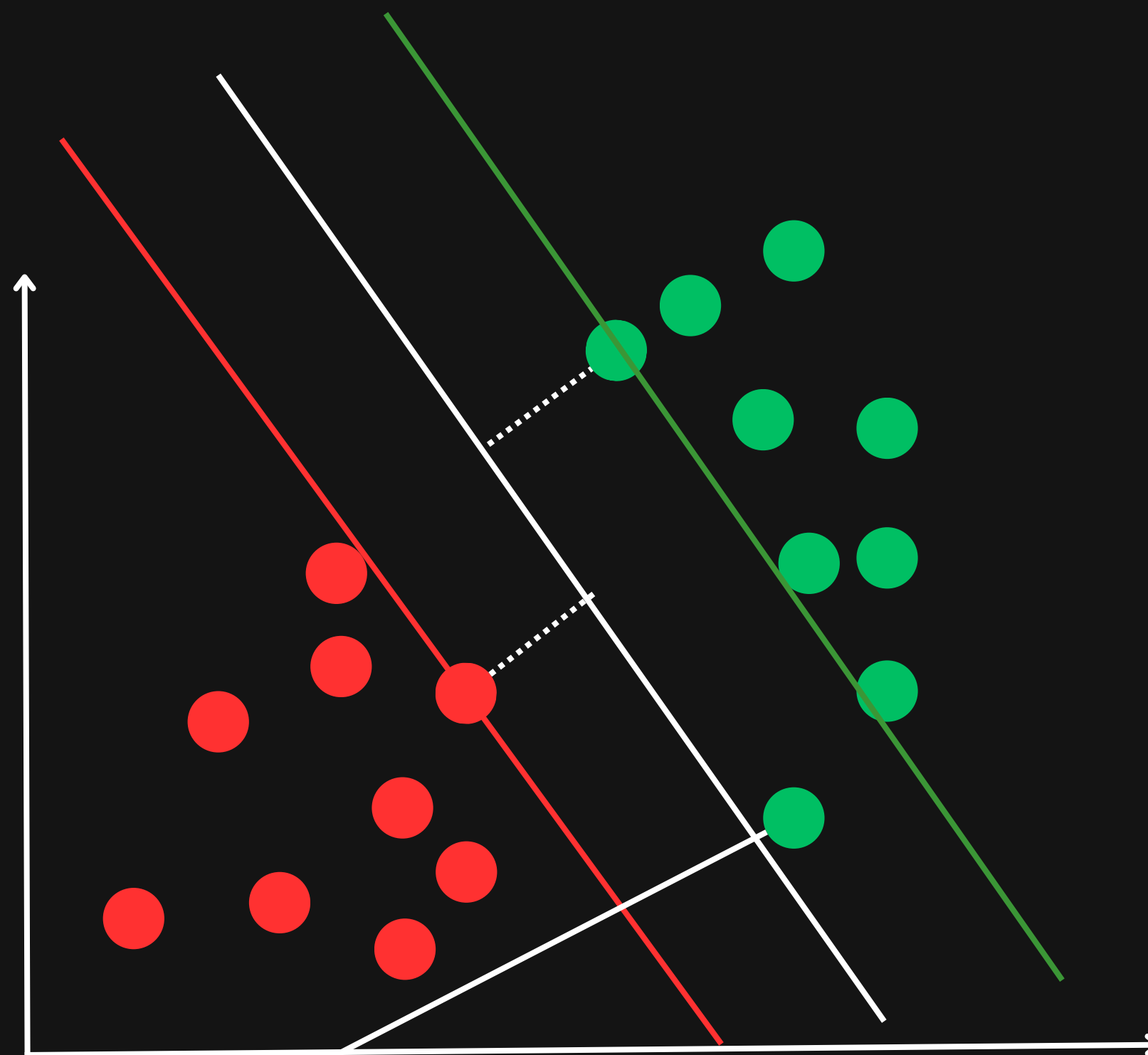


Predict which  
class the yellow  
dot belongs to





Incorrectly classifies the test datapoint

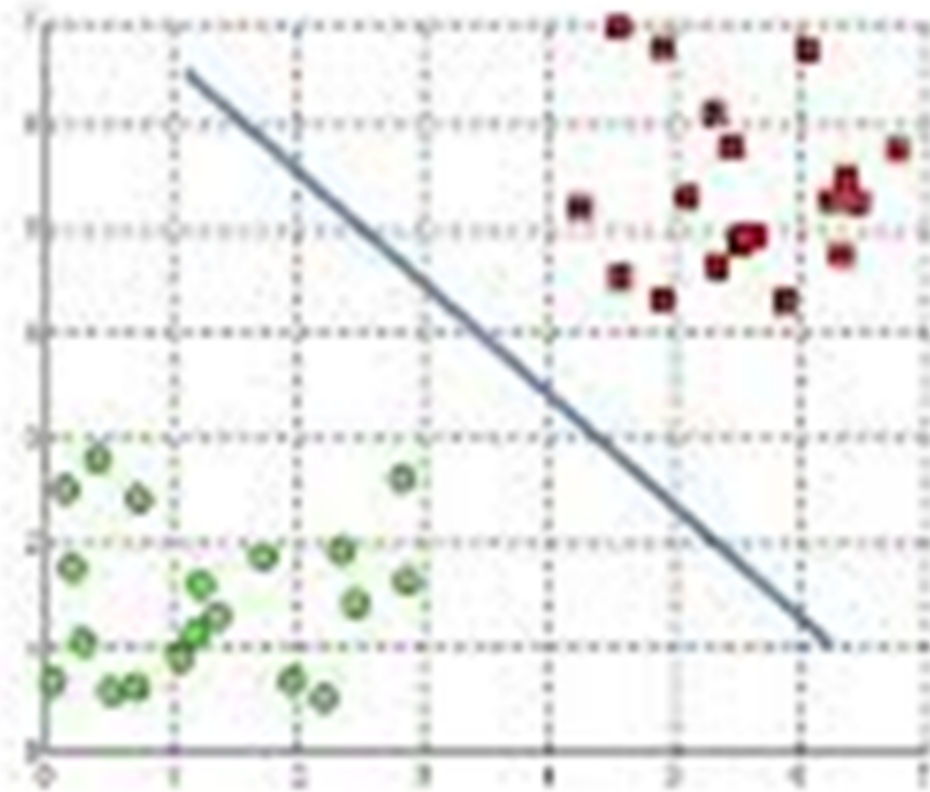


Correctly classifies the test datapoint

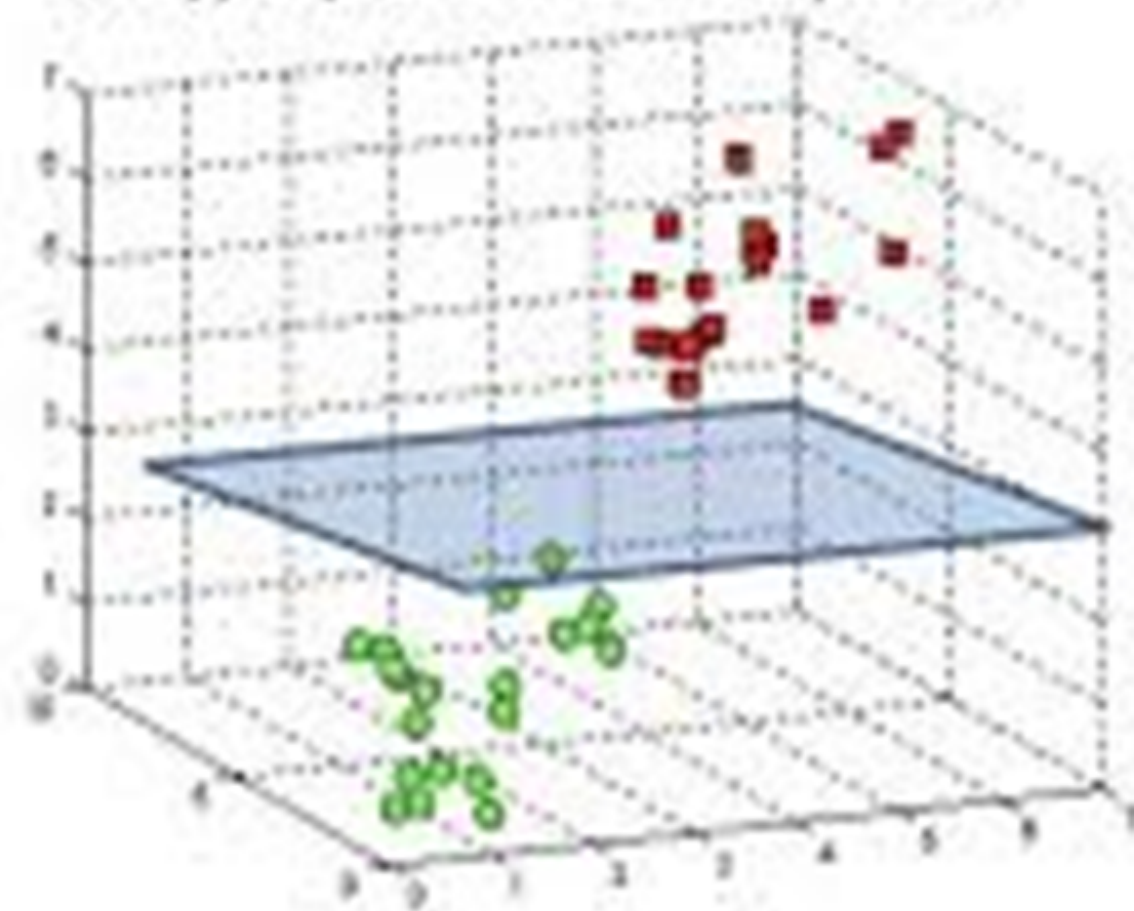
Test datapoint

The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane.

A hyperplane in  $\mathbb{R}^2$  is a line



A hyperplane in  $\mathbb{R}^3$  is a plane





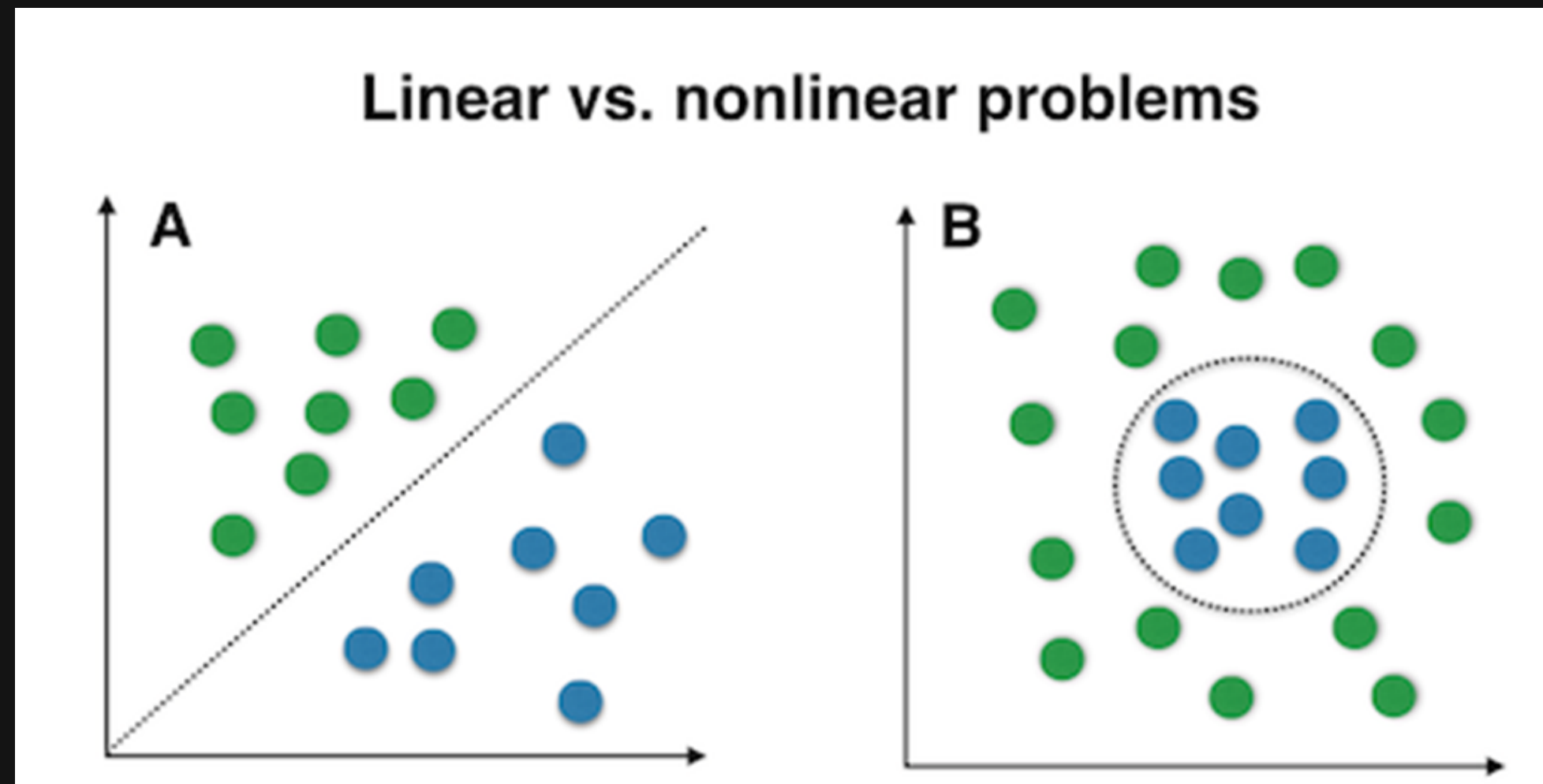
# Linear and Non Linear SVMs

## Linear SVM

- Used when data is perfectly linearly separable
- Data points can be classified into 2 classes by using a single straight line(if 2D).

## Non-Linear SVM

- Used when the data is not linearly separable
- Data points cannot be separated into 2 classes by using a straight line (if 2D)
- We use advanced techniques like kernel tricks to classify them.

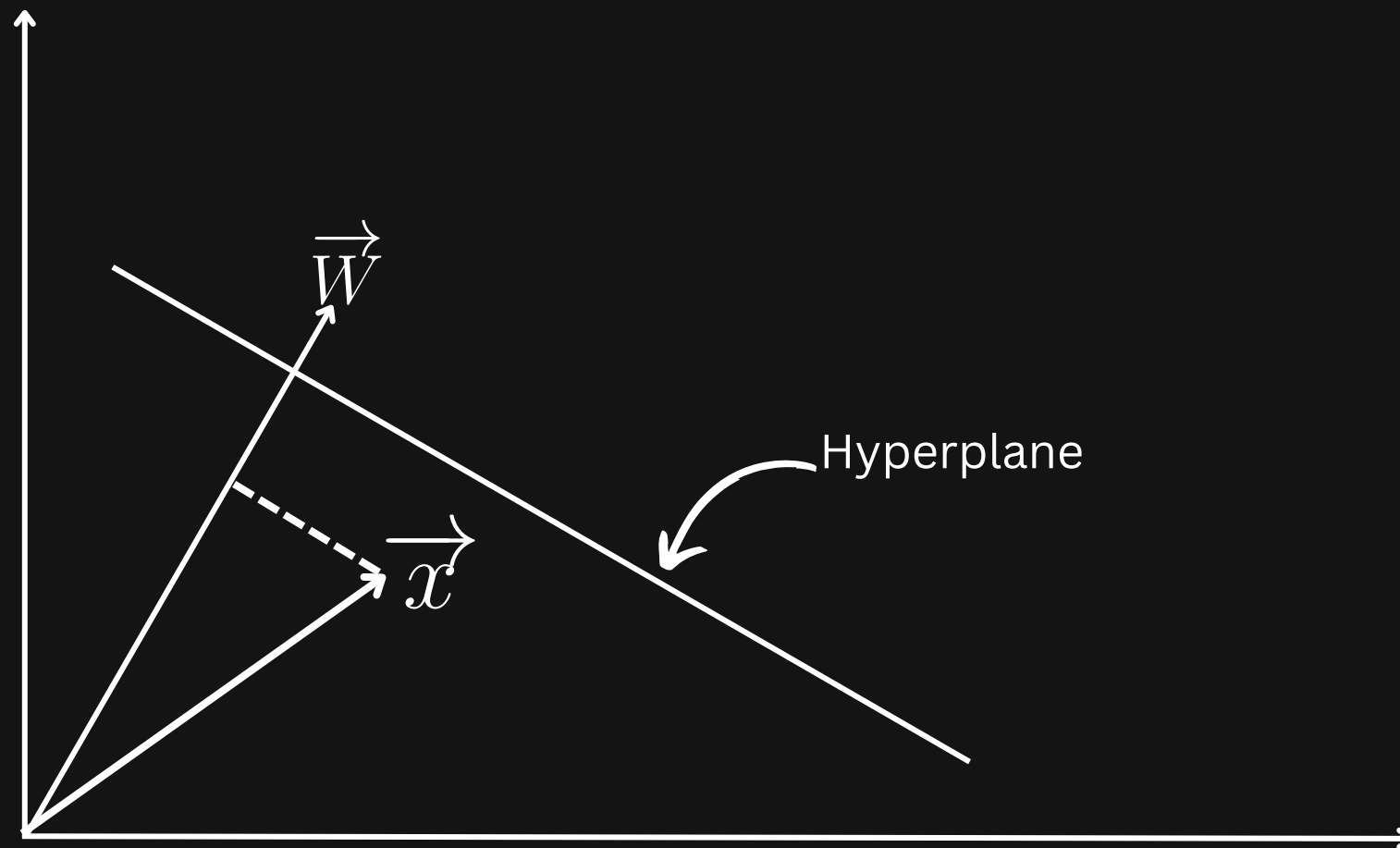


Here  $\vec{w}$  is a vector that is perpendicular to the hyperplane  
and  $\vec{x}$  is the position vector of a datapoint

$\vec{x} \cdot \vec{w} = c$  (*the point lies on the decision boundary*)

$\vec{x} \cdot \vec{w} > c$  (positive samples)

$\vec{x} \cdot \vec{w} < c$  (negative samples)



Lets define two new terms  $b$  ( $b = -c$ ) and  $y$ ,  
where

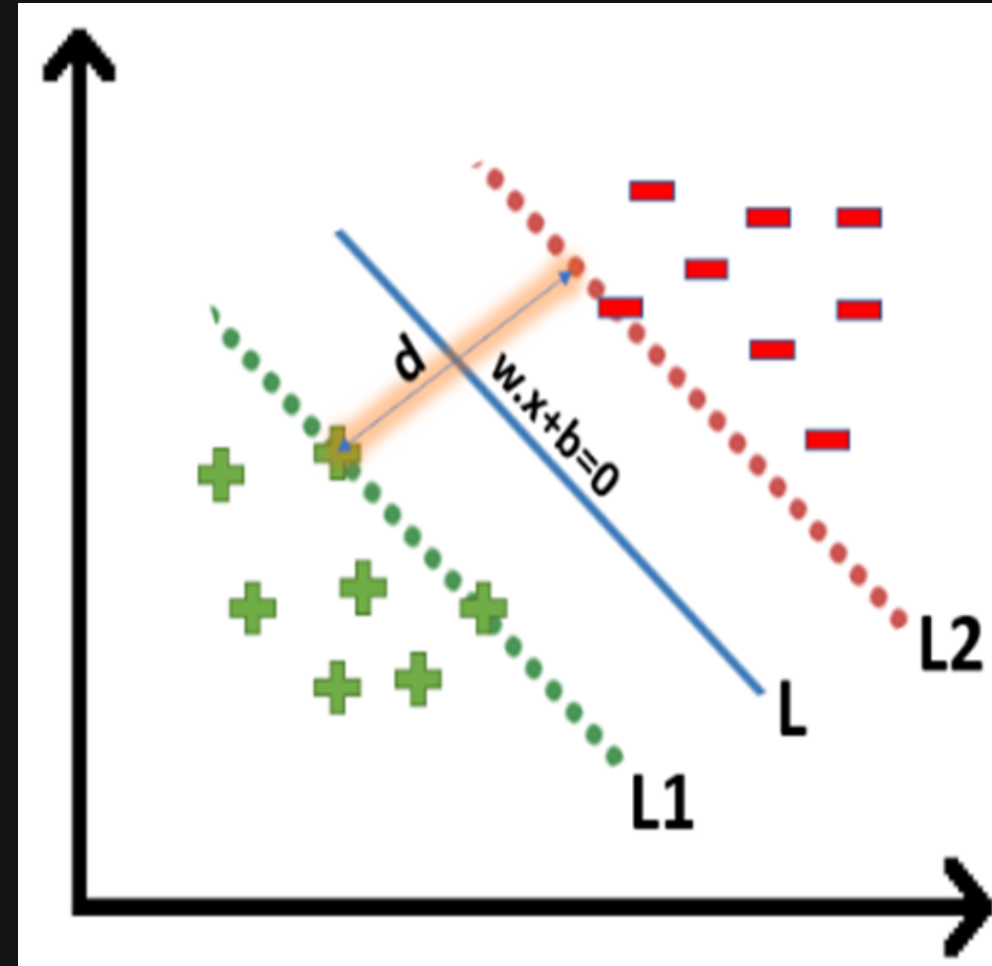
$$y = \begin{cases} +1 & \text{if } \vec{X} \cdot \vec{w} + b > 0 \\ -1 & \text{if } \vec{X} \cdot \vec{w} + b < 0 \end{cases}$$

The equation of:

hyperplane is  $\vec{X} \cdot \vec{w} + b = 0$

L1 is  $\vec{X} \cdot \vec{w} + b = -1$ ,

L2 is  $\vec{X} \cdot \vec{w} + b = +1$



So here  $y = +1$  for all datapoints in the red class and  $y$  is  $-1$  for all datapoints in the green class

$\vec{W}$  is  $m\hat{i} + n\hat{j}$ ,  $\vec{X}$  is  $x\hat{i} + y\hat{j}$ . So  $\vec{W} \cdot \vec{X} + b$  can be written as  $mx + ny + b$

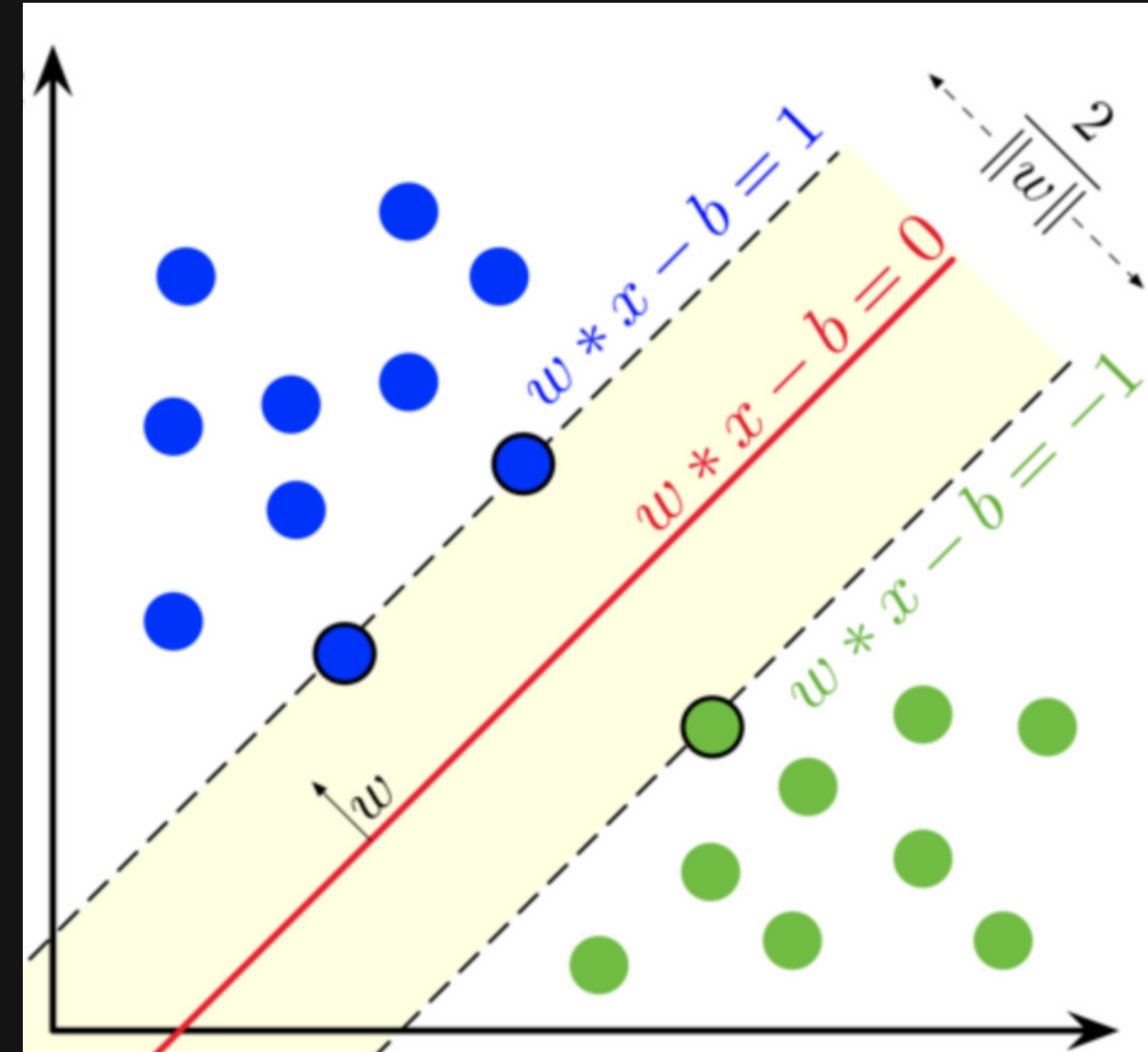
L1:  $mx + ny + b = -1$


L2:  $mx + ny + b = +1$

The distance  $d$  between L1 and L2 is given by  $\frac{(b+1)-(b-1)}{\sqrt{m^2+n^2}} = \frac{2}{\|\vec{W}\|}$

So we want the maximum distance 'd' such that  $y_i (\vec{W} \cdot \vec{X}_i + b) \geq 1$ .

$\operatorname{argmin}(w, b) \frac{\|\vec{w}\|}{2}$  such that  $y_i (\vec{W} \cdot \vec{X}_i + b) \geq 1$





In real-life applications we don't find datasets which are linearly separable, what we'll find is either an almost linearly separable dataset or a non-linearly separable dataset.

So what do we do with such data??

Lets introduce two concepts:

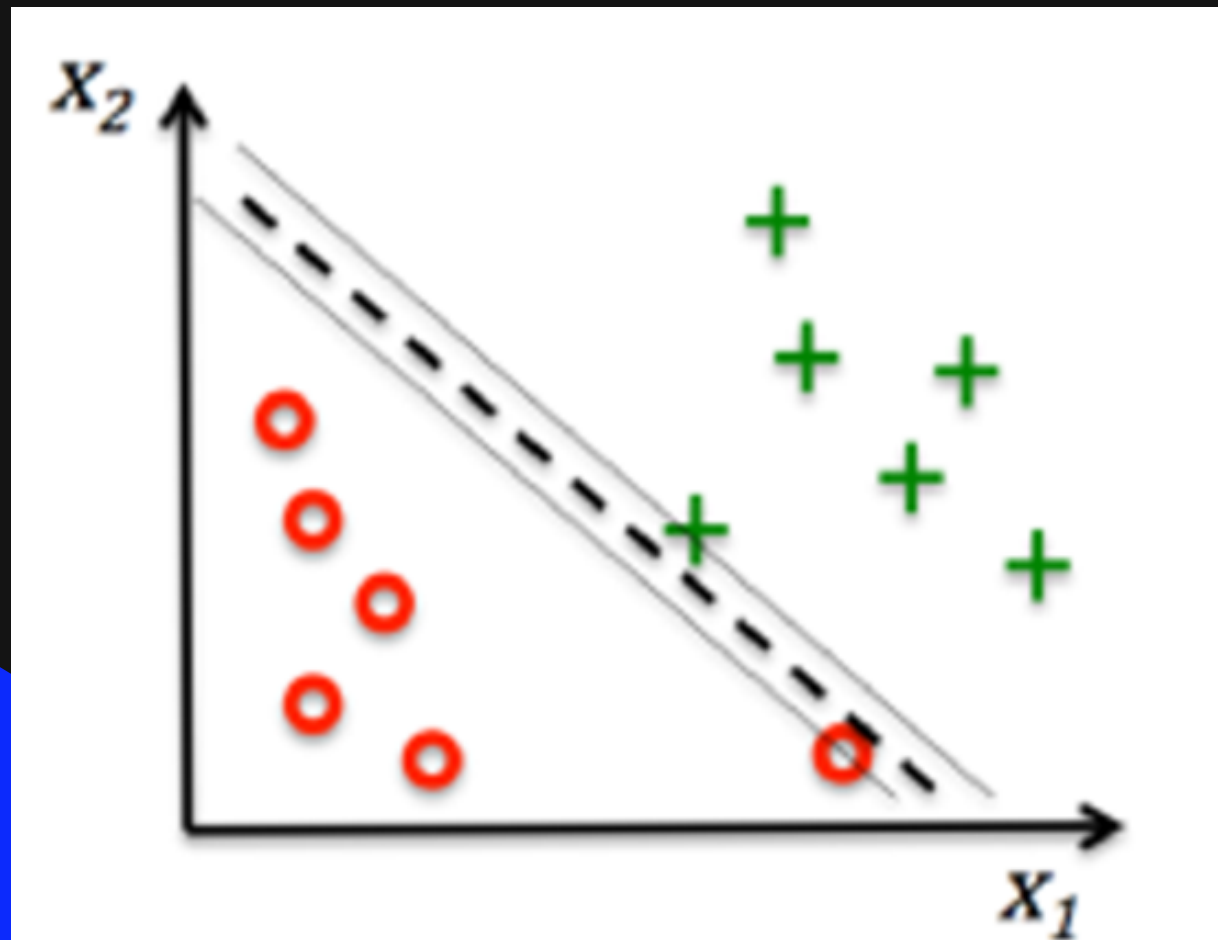
1) Soft Margin

2) Kernels

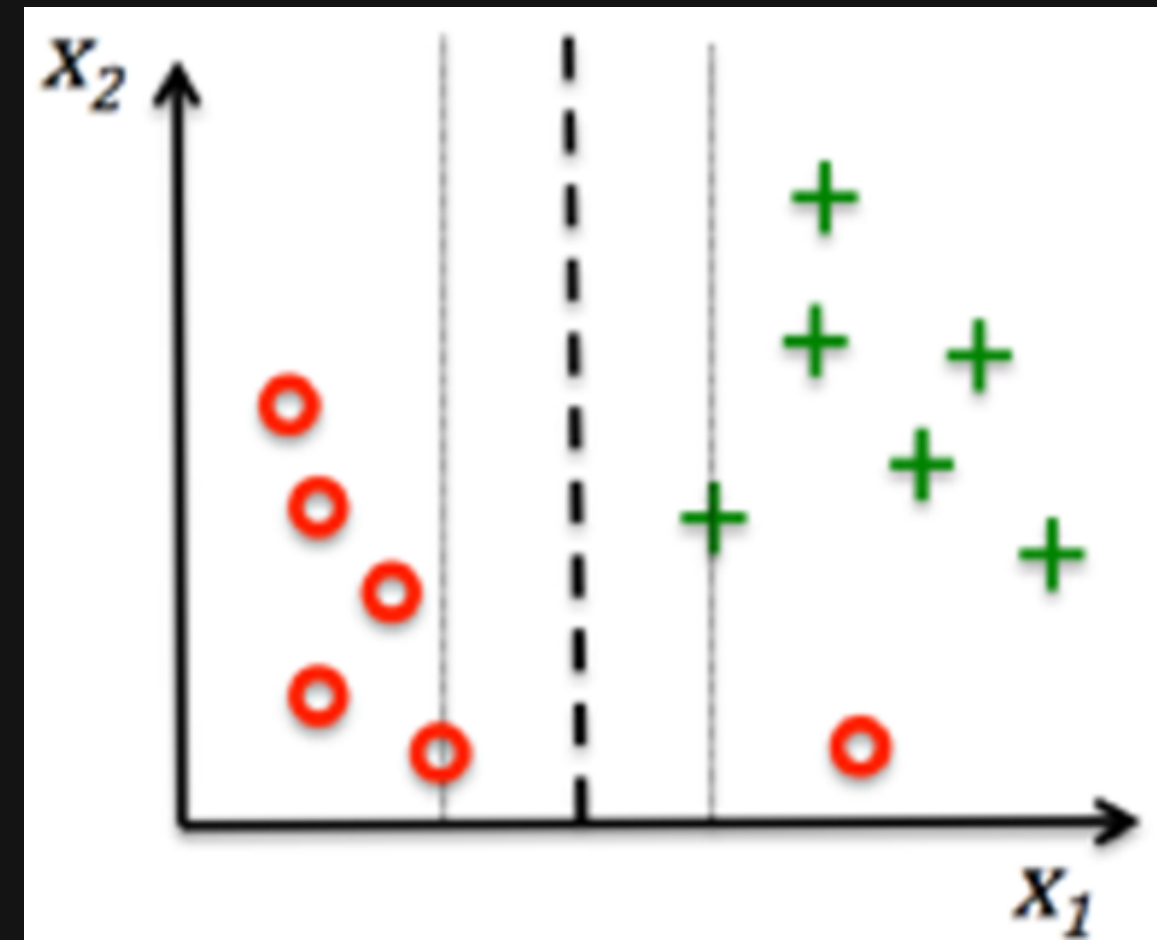


# SOFT MARGIN

Soft margin SVM allows some misclassification to happen by relaxing the hard constraints of Support Vector Machine. By doing this we prevent overfitting.

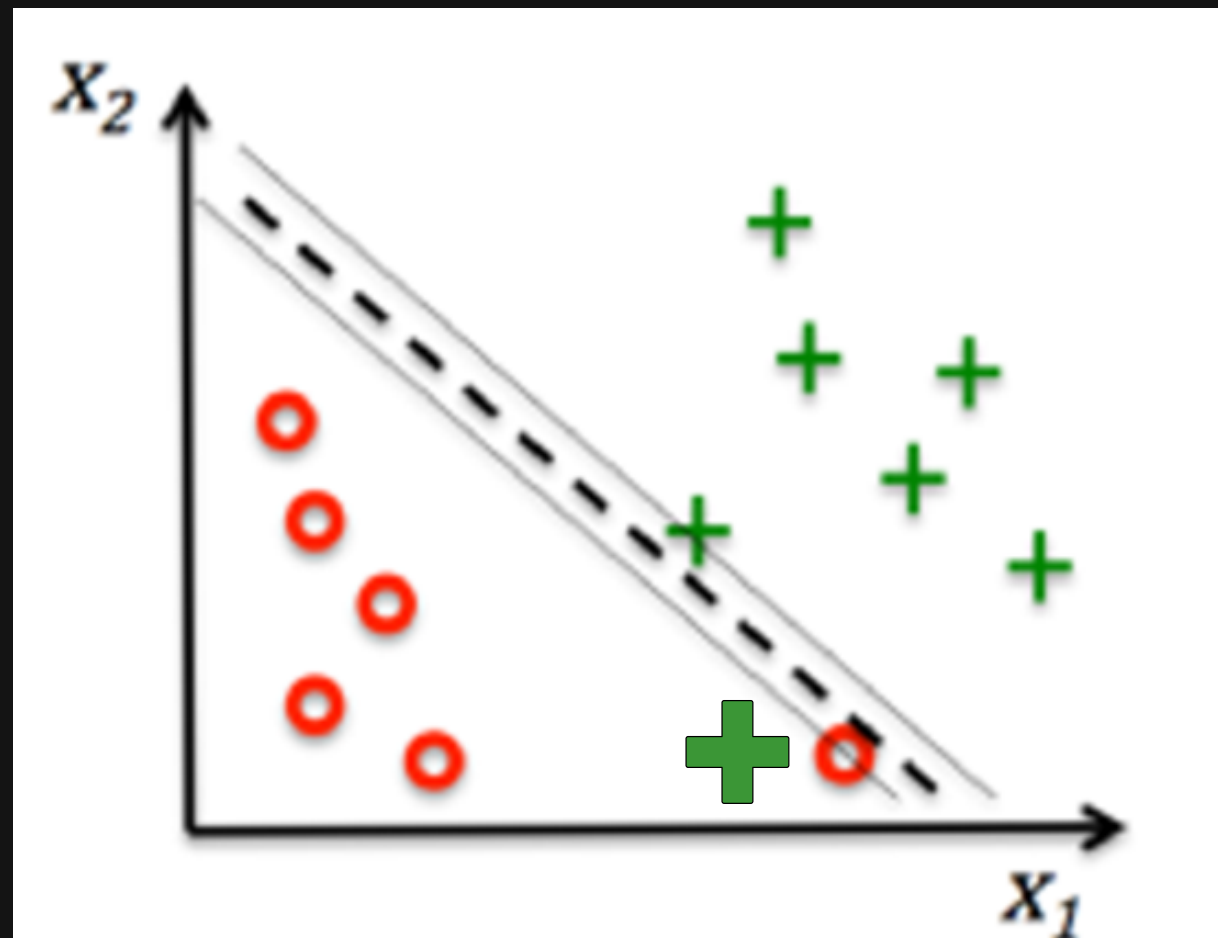


Correctly classifies all the training datapoints

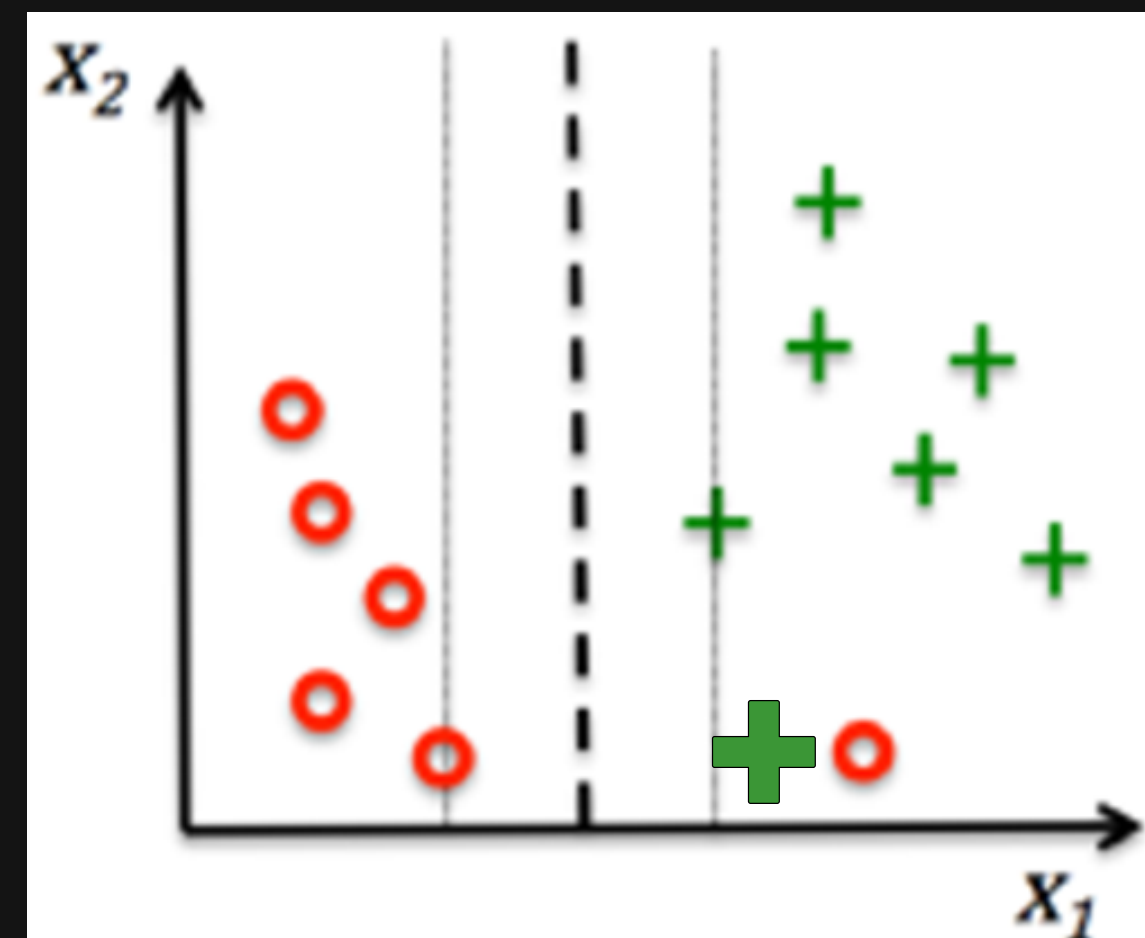


Incorrectly classified one training datapoint

Lets add a new(test) datapoint and see what happens....



Incorrectly classifies the test datapoint



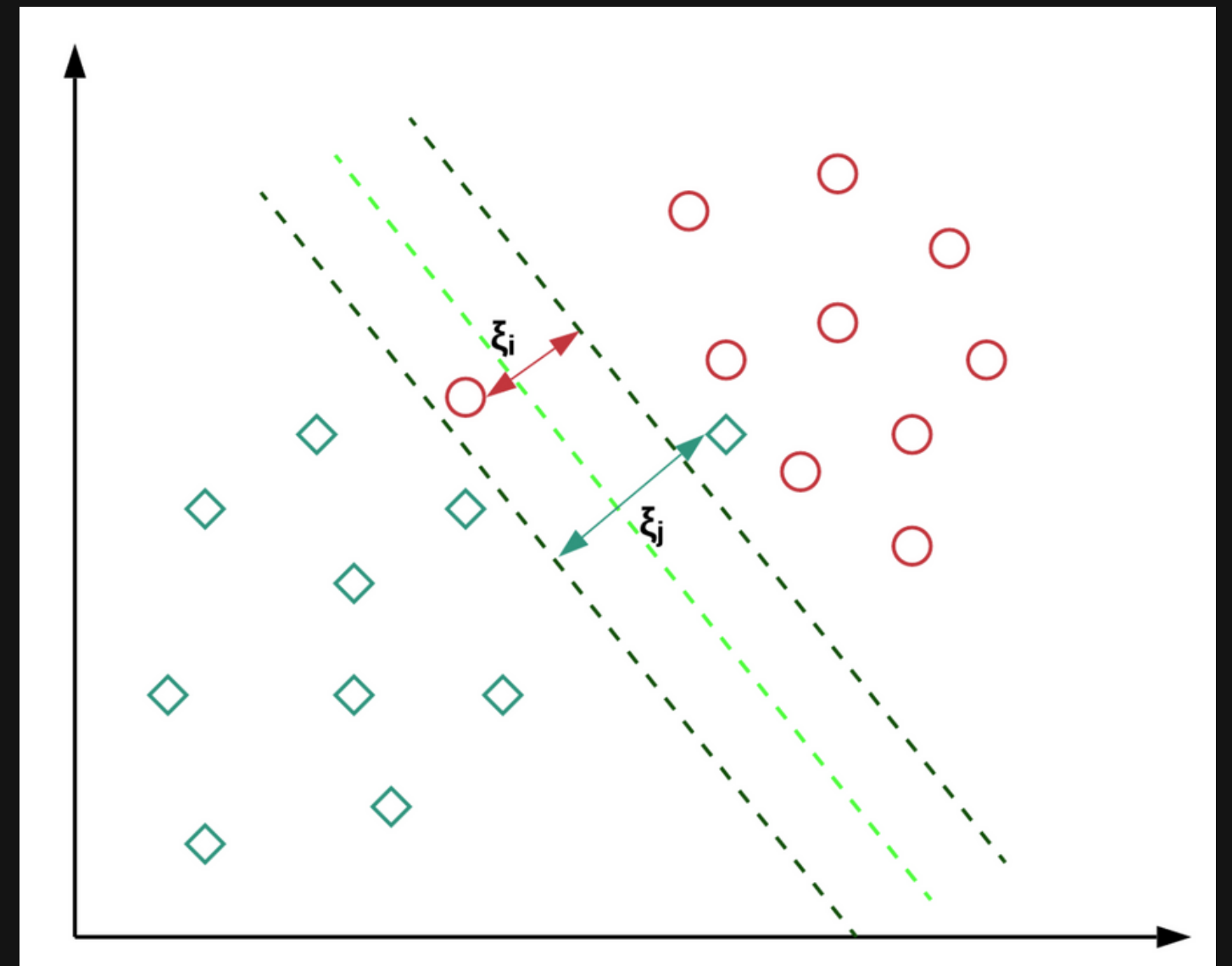
Correctly classifies the test datapoint

- we add two more terms to this equation which is  $\zeta$  and multiply that by a hyperparameter 'c'.
- For all the correctly classified points our zeta will be equal to 0
- and for all the incorrectly classified points the zeta is simply the distance of that particular point from its correct hyperplane .

So now what were trying to do is to minimize  $\frac{||w||}{2} + c \sum_{i=1}^n \zeta_i$

$$\operatorname{argmin}(w^*, b^*) \frac{||w||}{2} + c \sum_{i=1}^n \zeta_i$$

$$\text{Cost function} = \frac{1}{2} ||w||^2 + C \sum_{i=1}^n \max(0, 1 - y_i(\vec{w} \cdot \vec{x}_i + b))$$







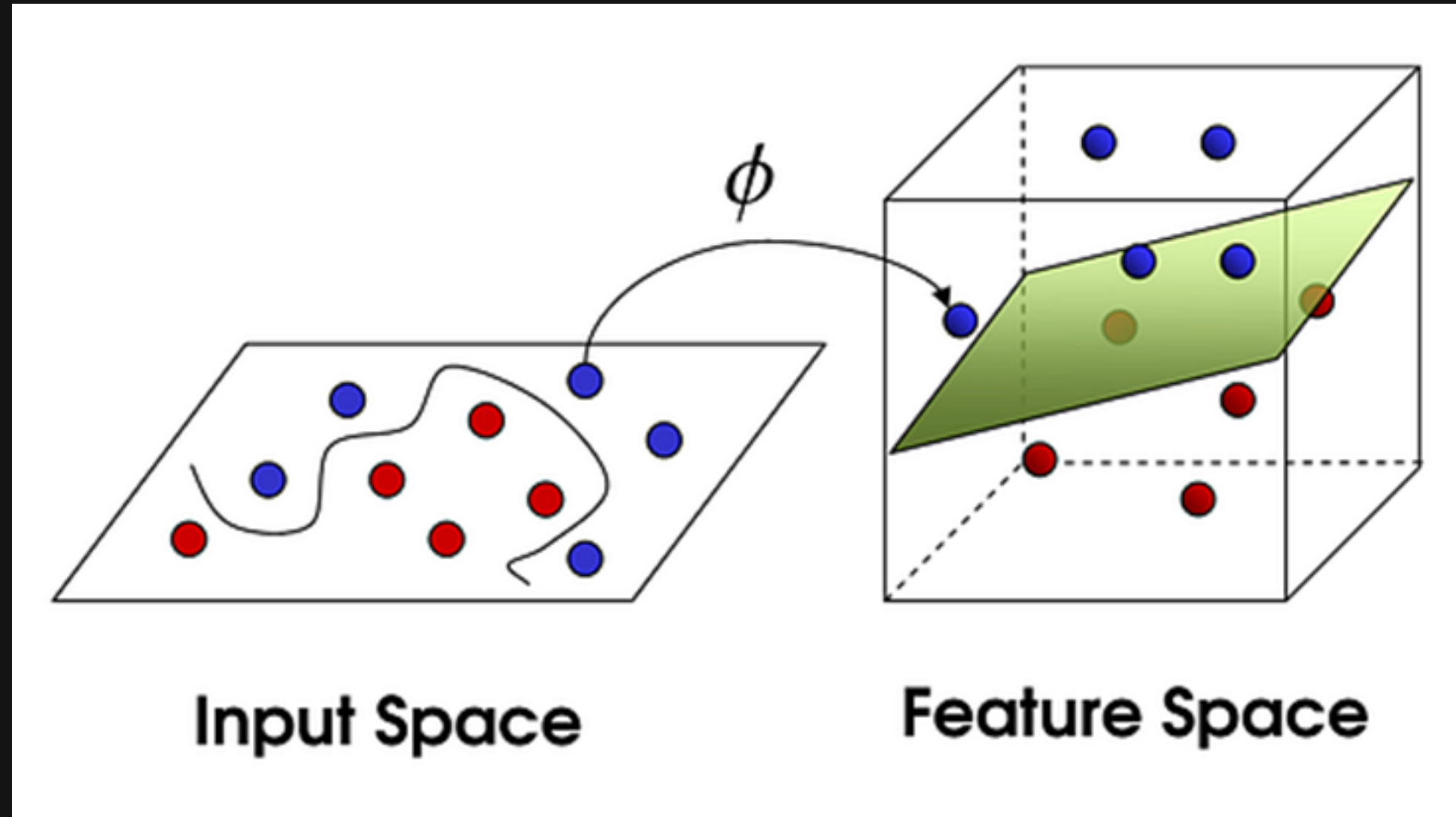
## Large value of 'c'

- Low training error
- Acts like a hard margin
- May overfit the training data

## Small value of 'c'

- Higher training error
- Acts like a soft margin
- May underfit the training data

# Kernels in Support Vector Machine



So what we're doing here is taking points from a 2d space and mapping it to a 3d space. This then allows us to find a hyperplane. However for more complex data we'll have to map our points to higher dimensions and that leads to unnecessary computational costs.

# So what do kernels do?


After a lot of math... we realized that the equation of the hyperplane is essentially a function of the dot product of vectors in the input space. which looks something like this

$$\sum y_i \alpha_i K(x_i, x) + b$$

Where  $K$  is the kernel function given by:

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

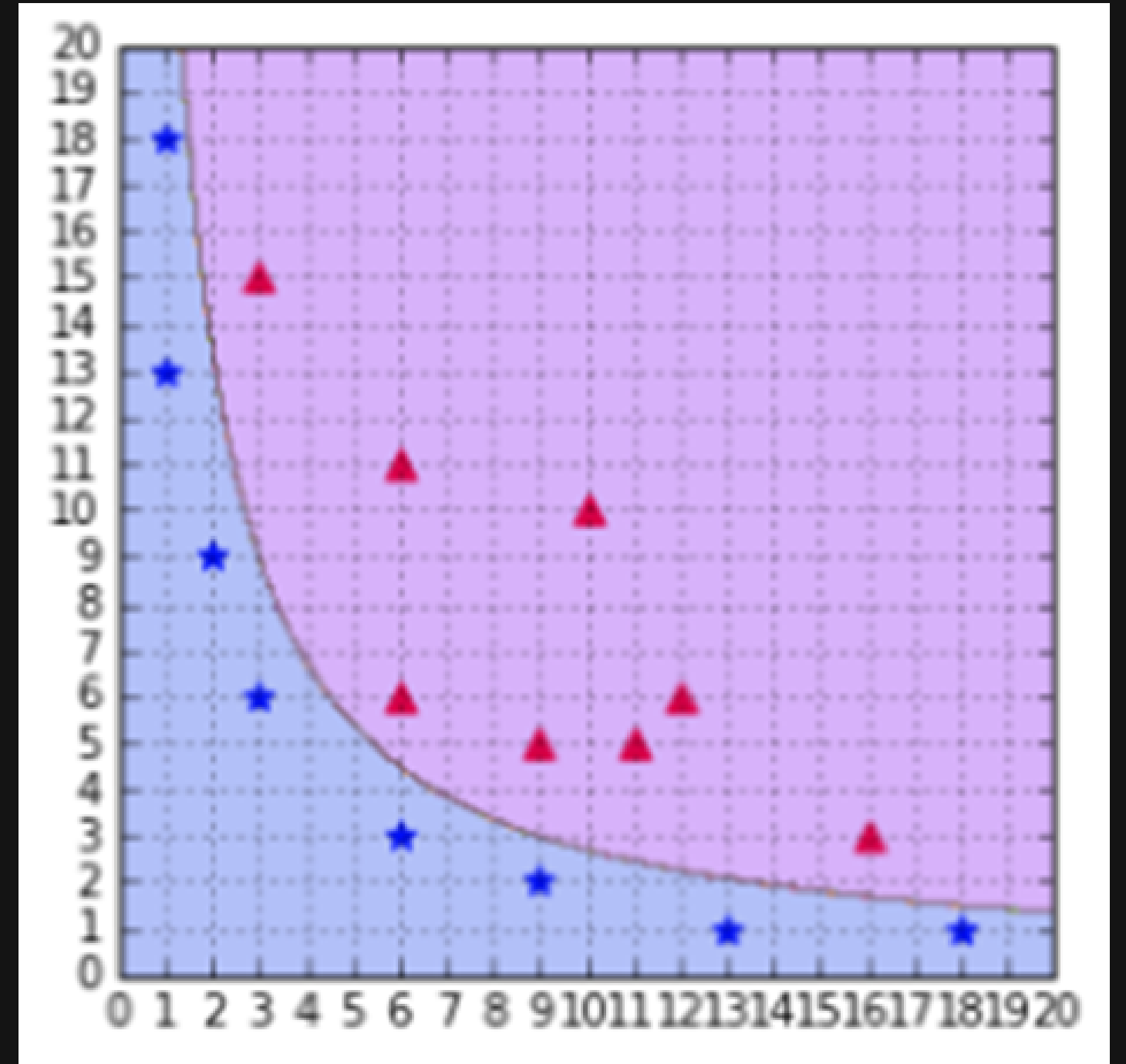
This means that there is no real need to calculate the corresponding new coordinates in the feature space. We can avoid all the unnecessary computational cost by directly using the kernel function :)



# Polynomial kernel

$$(\gamma \langle x, x' \rangle + r)^d$$

- $x, x'$  are vectors in the original space.
- Computes the dot product of the two vectors and hence finds similarities between vectors in the original space.
- It then uses this to create a polynomial boundary.



# Hyperparameter tuning

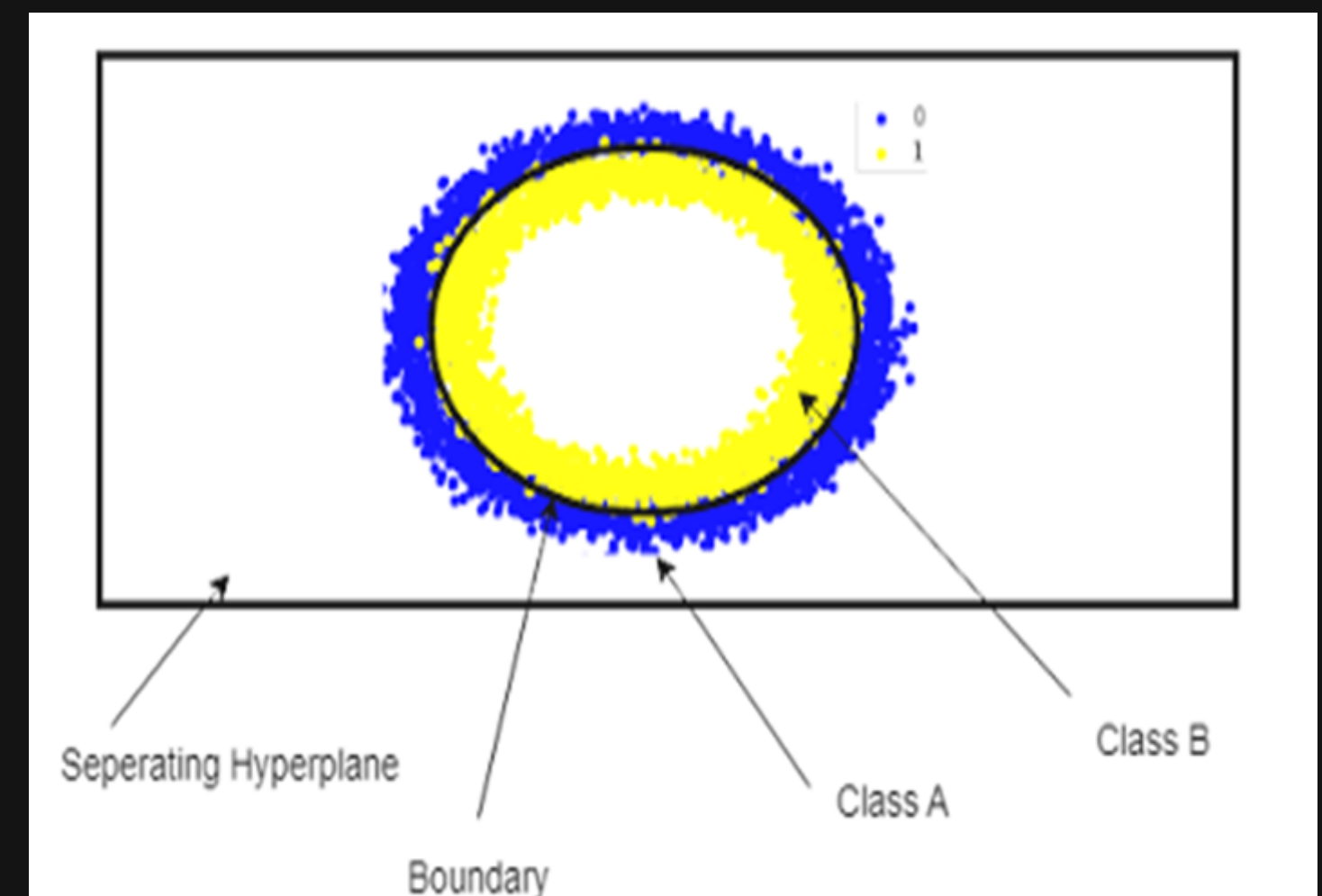
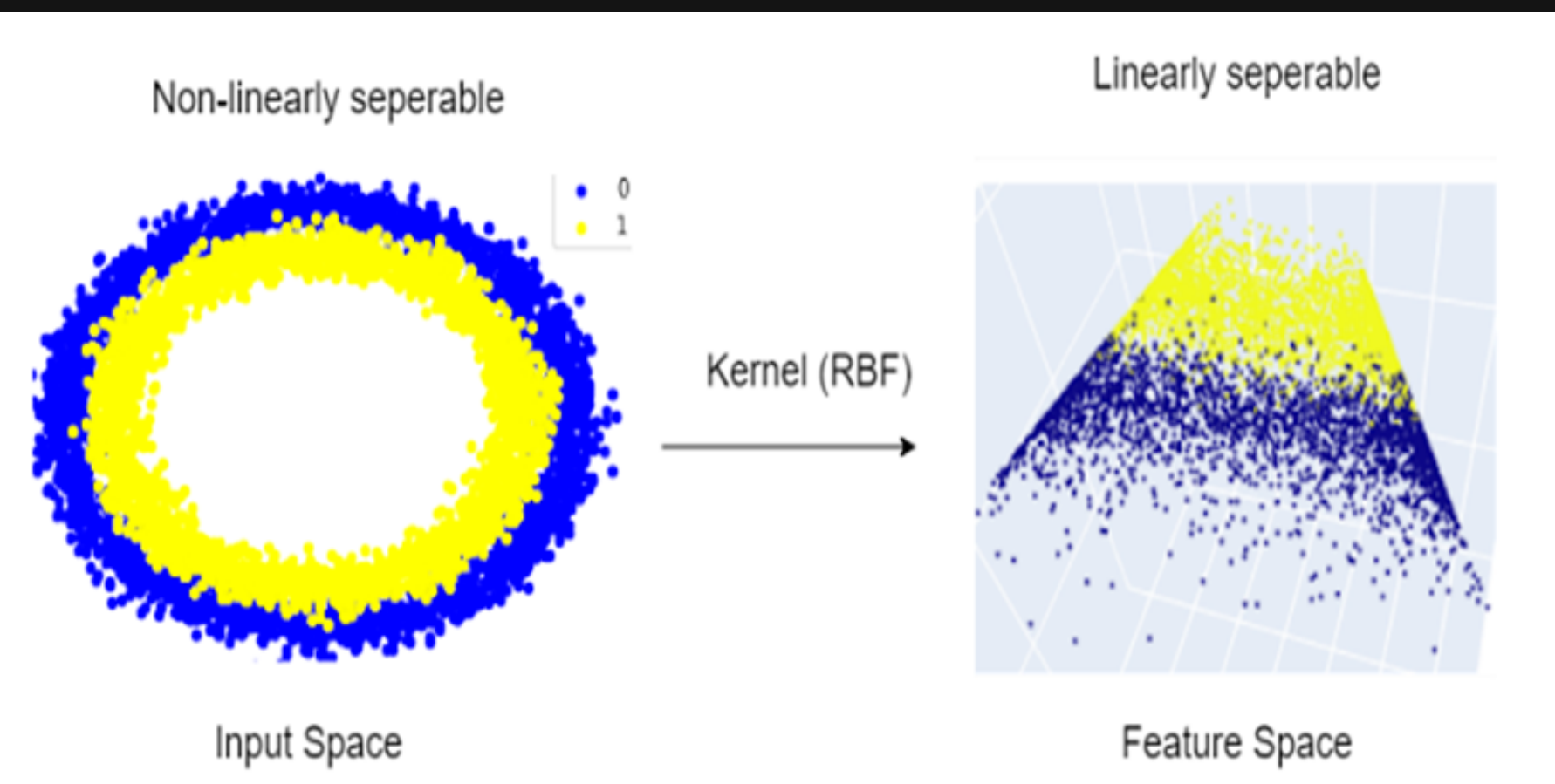
- A large  $c$  value will give a low training error but may overfit.
- A small  $c$  value will give a high training error but may underfit.
- The degree  $d$  of the polynomial can be used to control the complexity of the model.
- Higher degree results in a more complex model that may overfit the data.
- lower degree results in a simpler model that may underfit the data.

# RBF(radial basis function) kernel

The RBF kernel function for two points  $X_1$  and  $X_2$  computes the similarity or how close they are to each other. This kernel can be mathematically represented as follows:

$$K(X_1, X_2) = \exp(-\gamma \|X_1 - X_2\|^2)$$

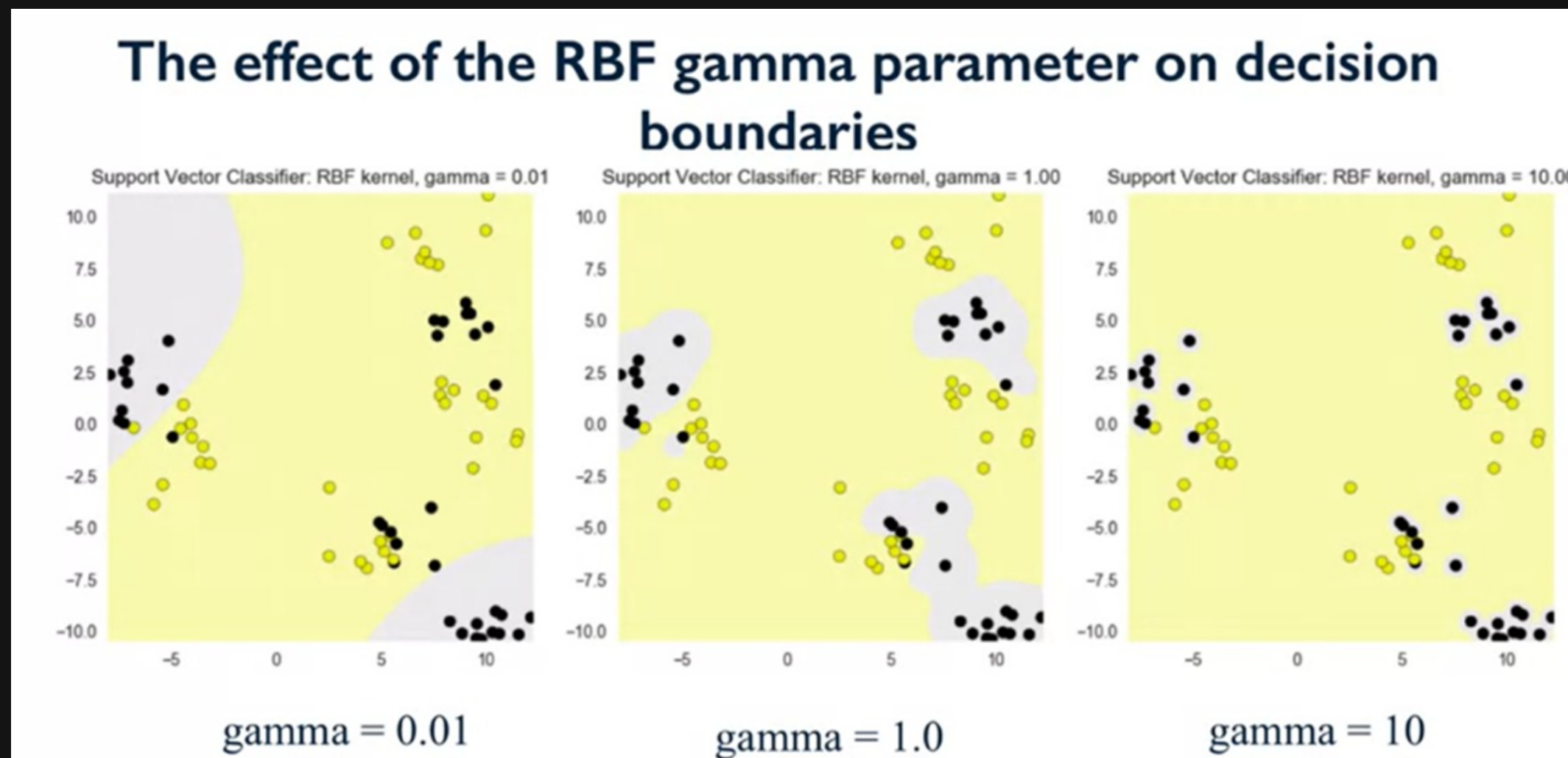
Where  $\|X_2 - X_1\|$  is the Euclidean ( $L_2$ -norm) Distance between two points  $X_1$  and  $X_2$



# Hyperparameter tuning

The Effect of  $c$  and Gamma:

- High  $c$ , better classification of training examples
- Low  $c$ , smoother boundary
- gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected.
- Higher Gamma value tends to overfit.
- Lower Gamma value tends to underfit.



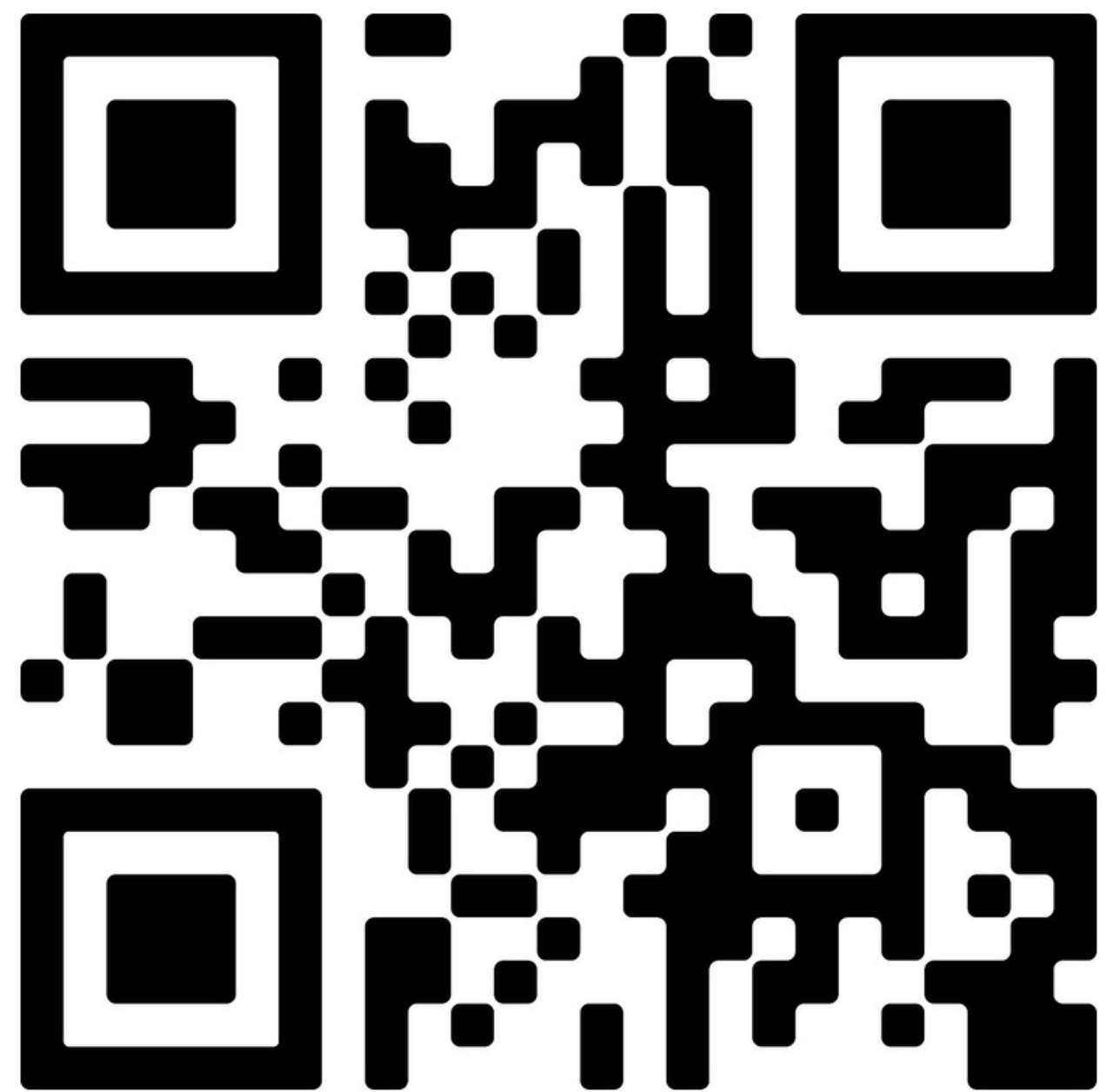


## Advantages of Support Vector Machine:

- 1.SVM works relatively well when there is a clear margin of separation between classes.
- 2.SVM is more effective in high dimensional spaces.
- 3.SVM is relatively memory efficient
- 4.The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem.
- 5.It scales relatively well to high dimensional data.

## Disadvantages of Support Vector Machine:

- 1.SVM algorithm is not suitable for large data sets.
- 2.SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.
- 3.As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.
- 4.Choosing a “good” kernel function is not easy.
- 5.The hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact



Lets move to the code implementation

