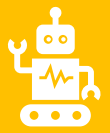
Support Vector Machine (SVM)

Machine Learning Approach

Support Vector Machines (SVM)

Originally developed in 1960's

Found again in 1990's



Now become popular – (Machine Learning)

Different from other machine learning algorithms

SVM is a *supervised* learning model

Introduction

A business organization receives lot of emails from customers every day. Some of these emails are complaints and should be answered very quickly.

Company would like to find a way to identify them quickly so that answer these email in priority.

Introduction

- Approach 1:
  - Create a label using keywords, for instance "**urgent", "complaint", "help**"
  - Drawback of this method
    - Need to identify all potential keywords (some angry users might use)
    - Probably miss some of them.
    - Over a time, keyword **list will probably become very messy,** and it will be **hard to maintain**.

Introduction

Approach 2: use a supervised machine learning algorithm.

Steps

- Need lot of emails, the more the better.
- Read the title of each email and classify it by saying "it is a complaint" or "it is not a complaint". It put a **label** on each email.
- **Train** a model on this dataset
- Assess the quality of the prediction (using cross validation)
- Use this model to **predict** if an email is a complaint or not.

SVMs - Support Vector Machines

SVM is used for classification

SVR (Support Vector Regression) is used for regression

Classification

In 1957, a simple linear model called the Perceptron was invented by Frank Rosenblatt to do classification.

A few years later, Vapnik and Chervonenkis, proposed another model called the "Maximal Margin Classifier", the SVM was born.

Then, in 1992, Vapnik et al. proposed the Kernel Trick, which allow to use the SVM to classify non-linear data.

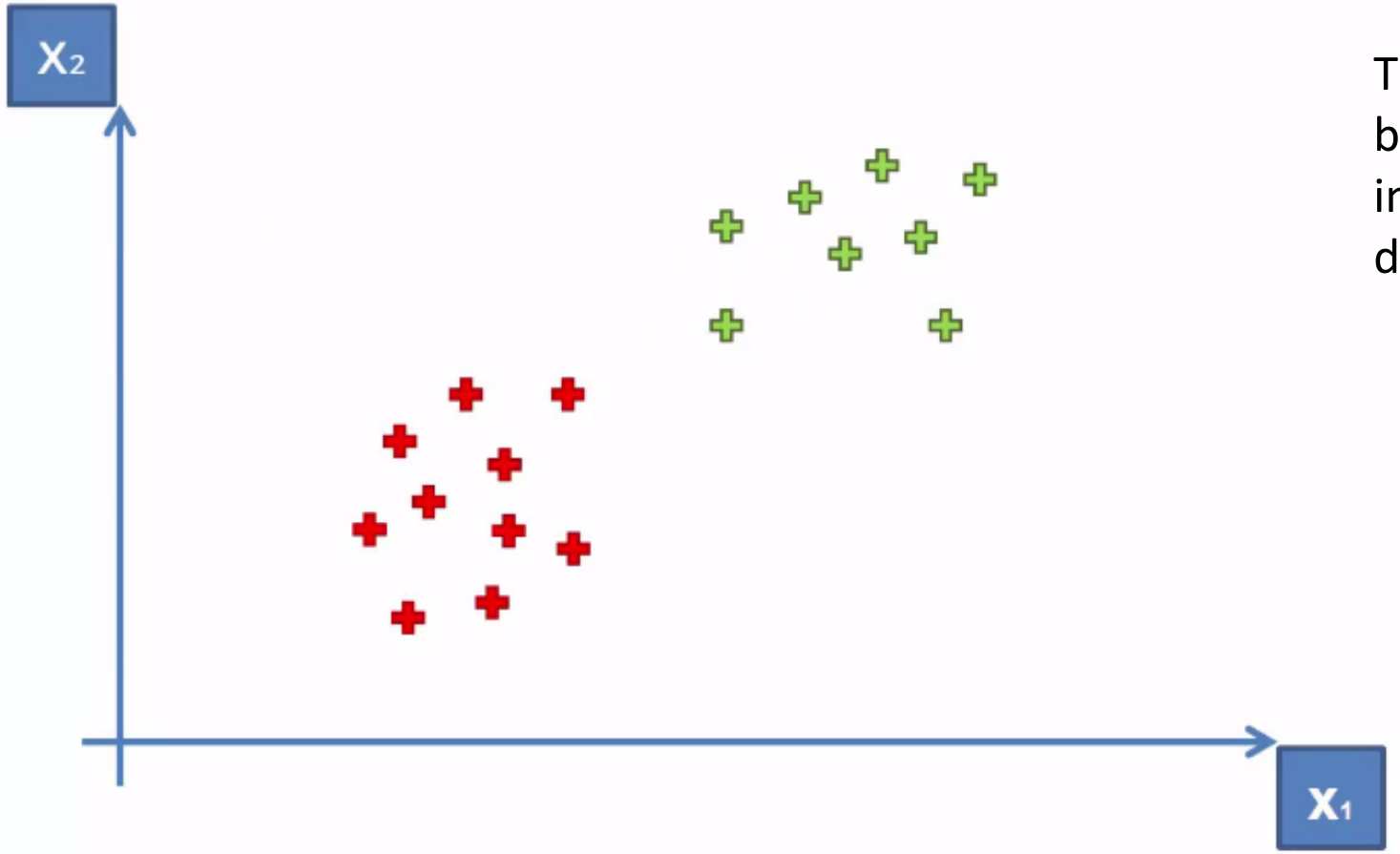
Classification

In 1995, Cortes and Vapnik introduced the **Soft Margin Classifier** which allows to **accept some misclassifications** when using a SVM.

What is a Support Vector Machine(SVM)?

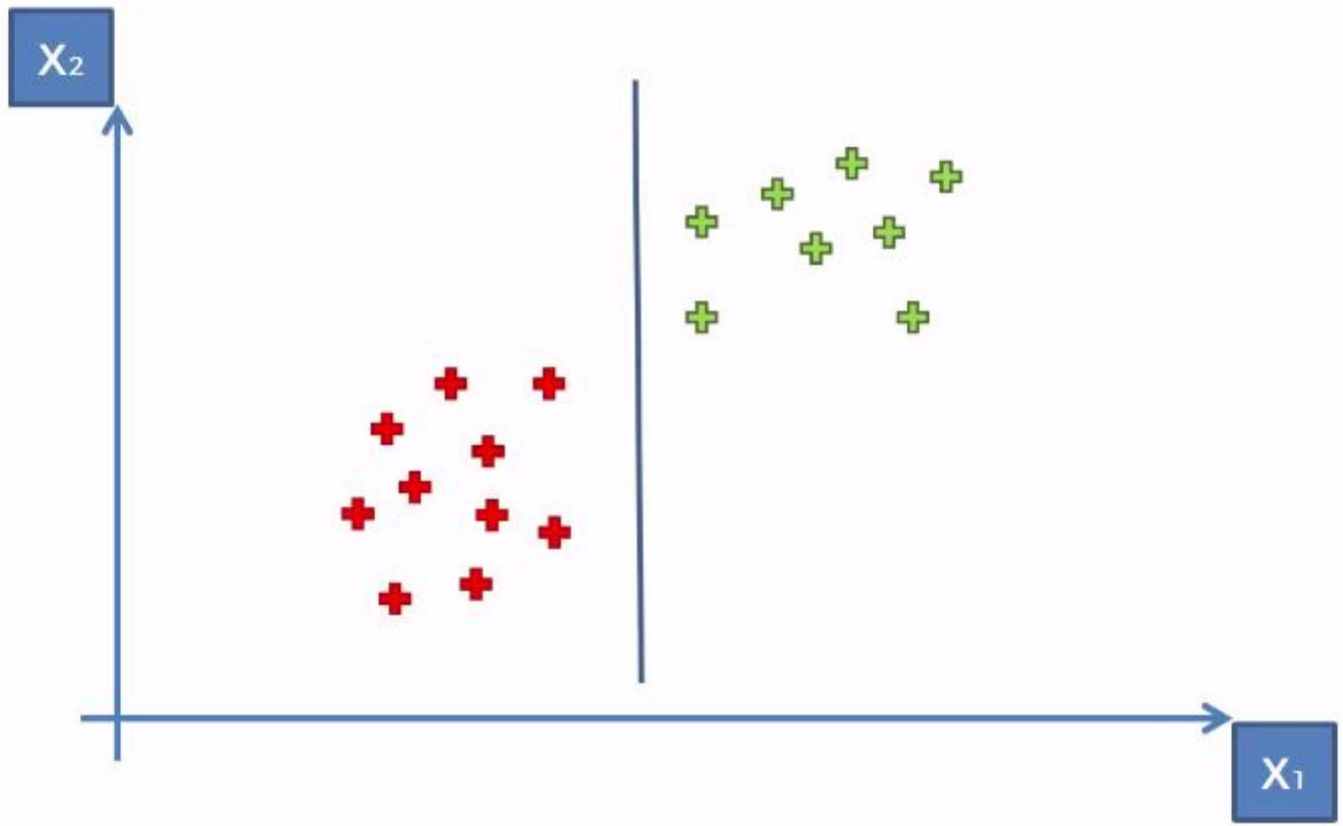
- It is a supervised machine learning problem where we try to find a hyperplane that best separates the two classes.

How to separate these points ?

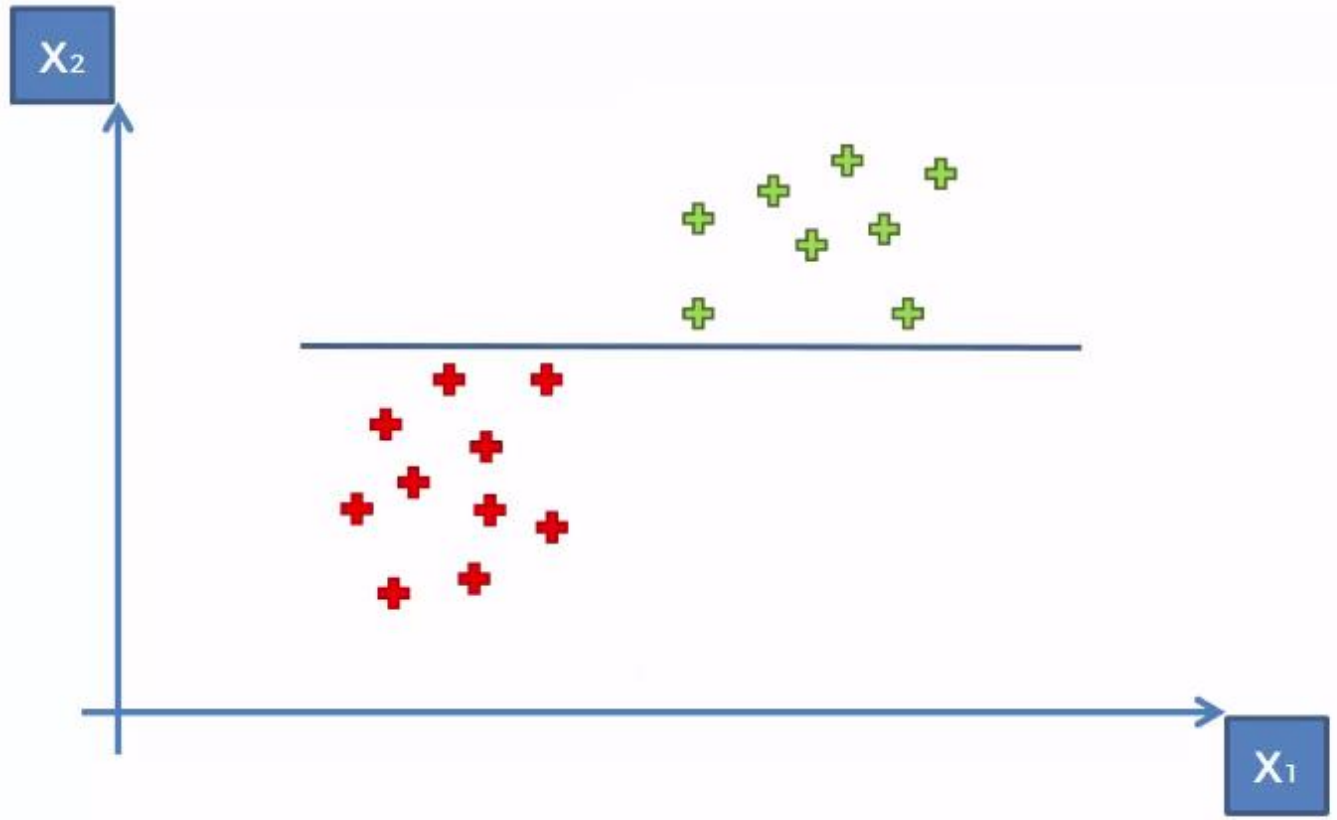


The decision boundary become very important for us to decide the new point

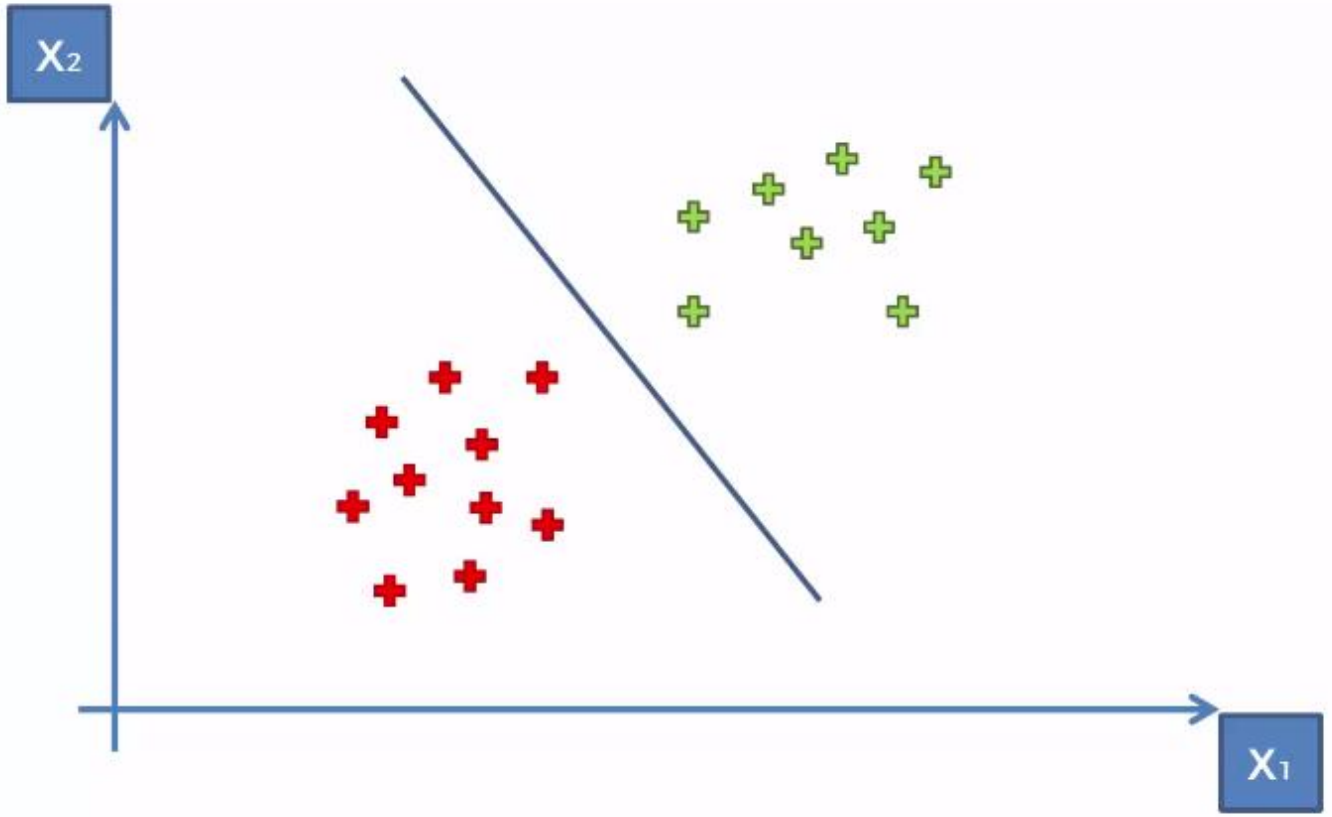
How to separate these points ?



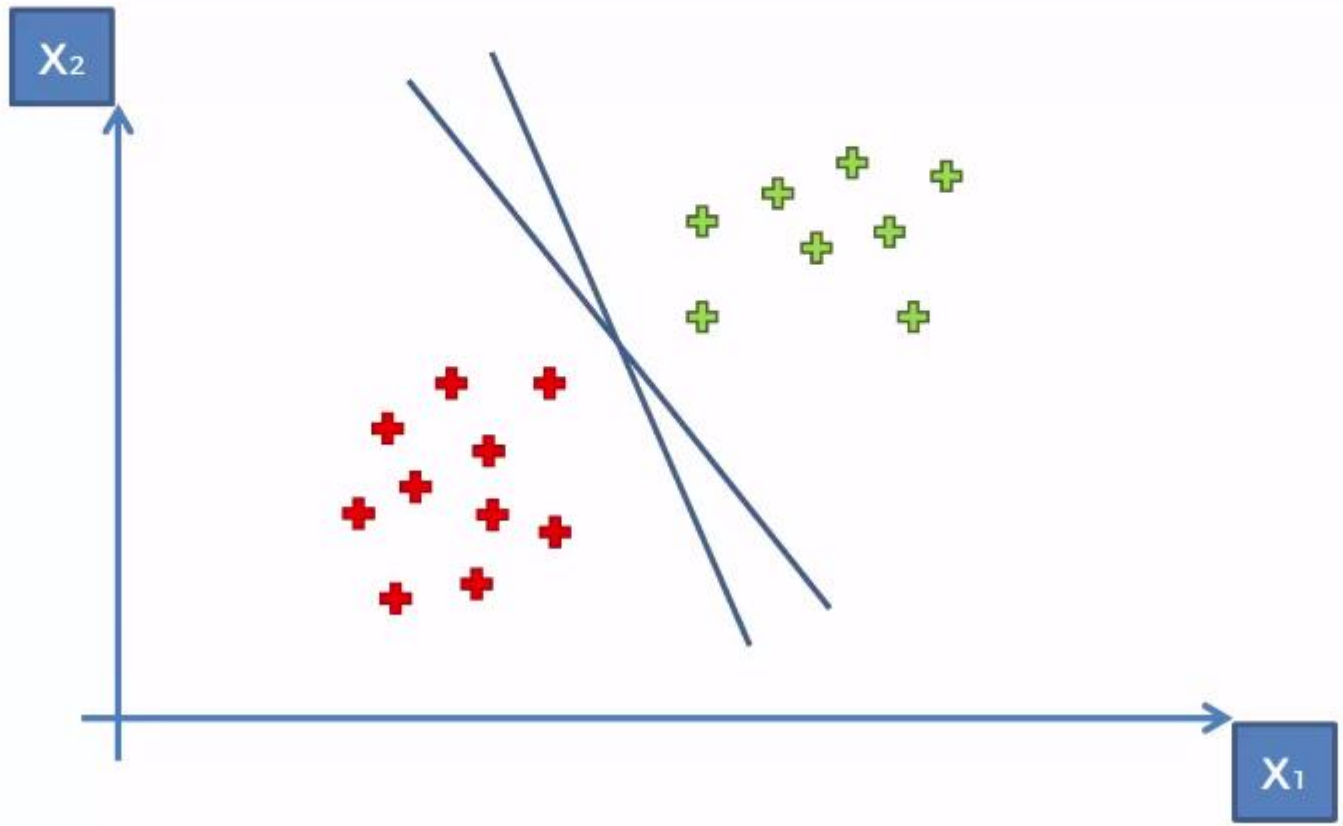
How to separate these points ?



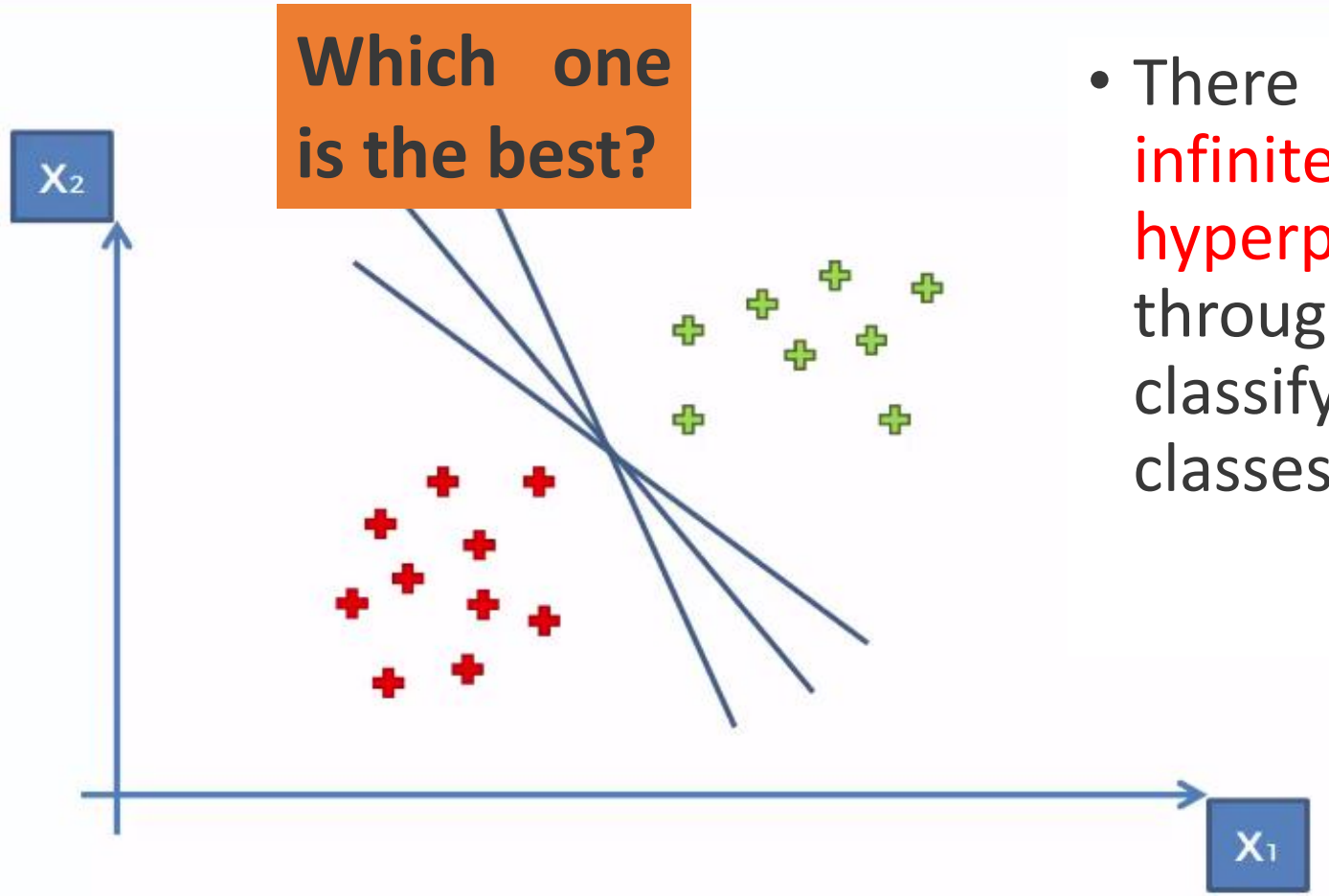
SVM



SVM



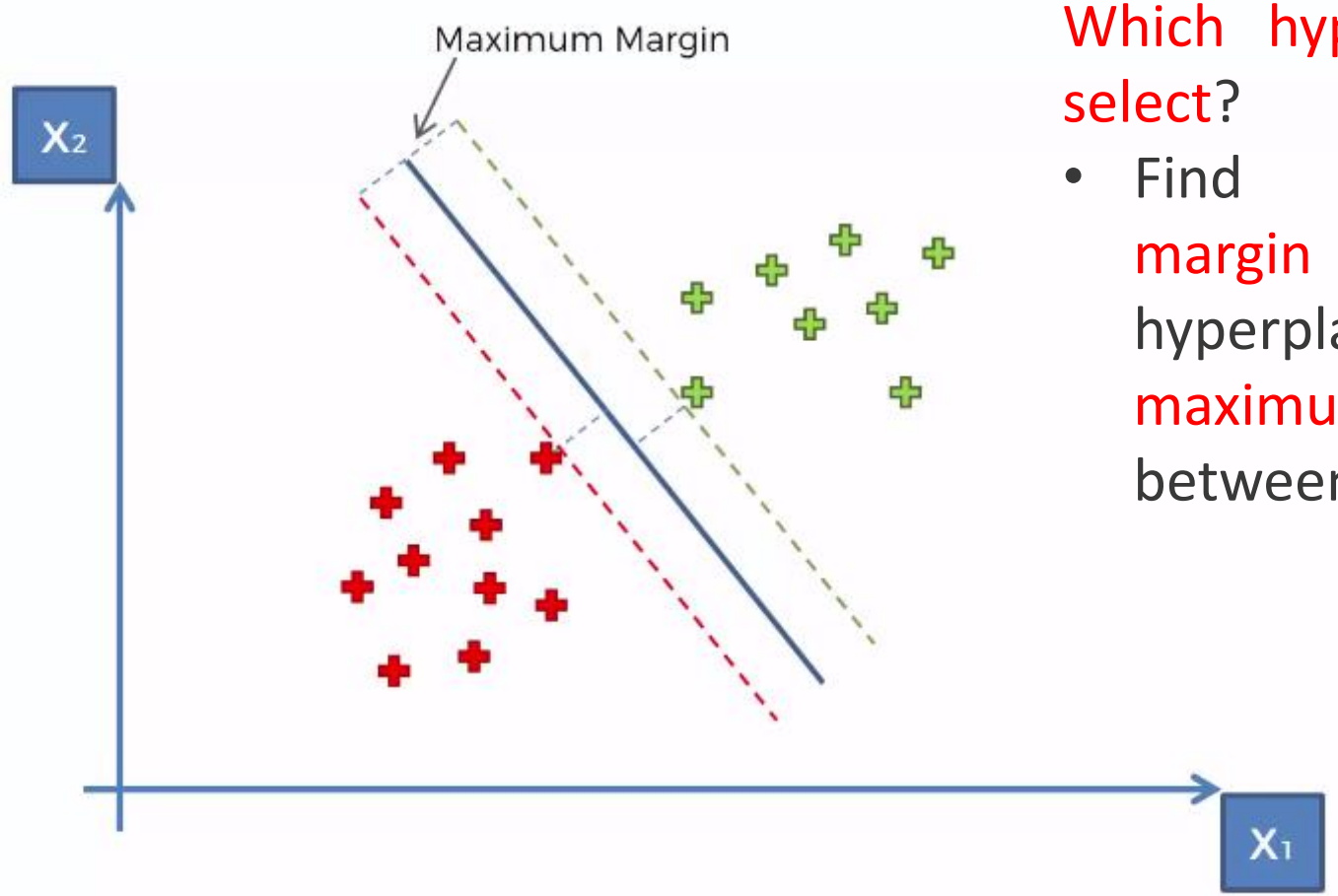
SVM



- There can be an infinite number of hyperplanes passing through a point and classifying the two classes perfectly.

Maximum Margin

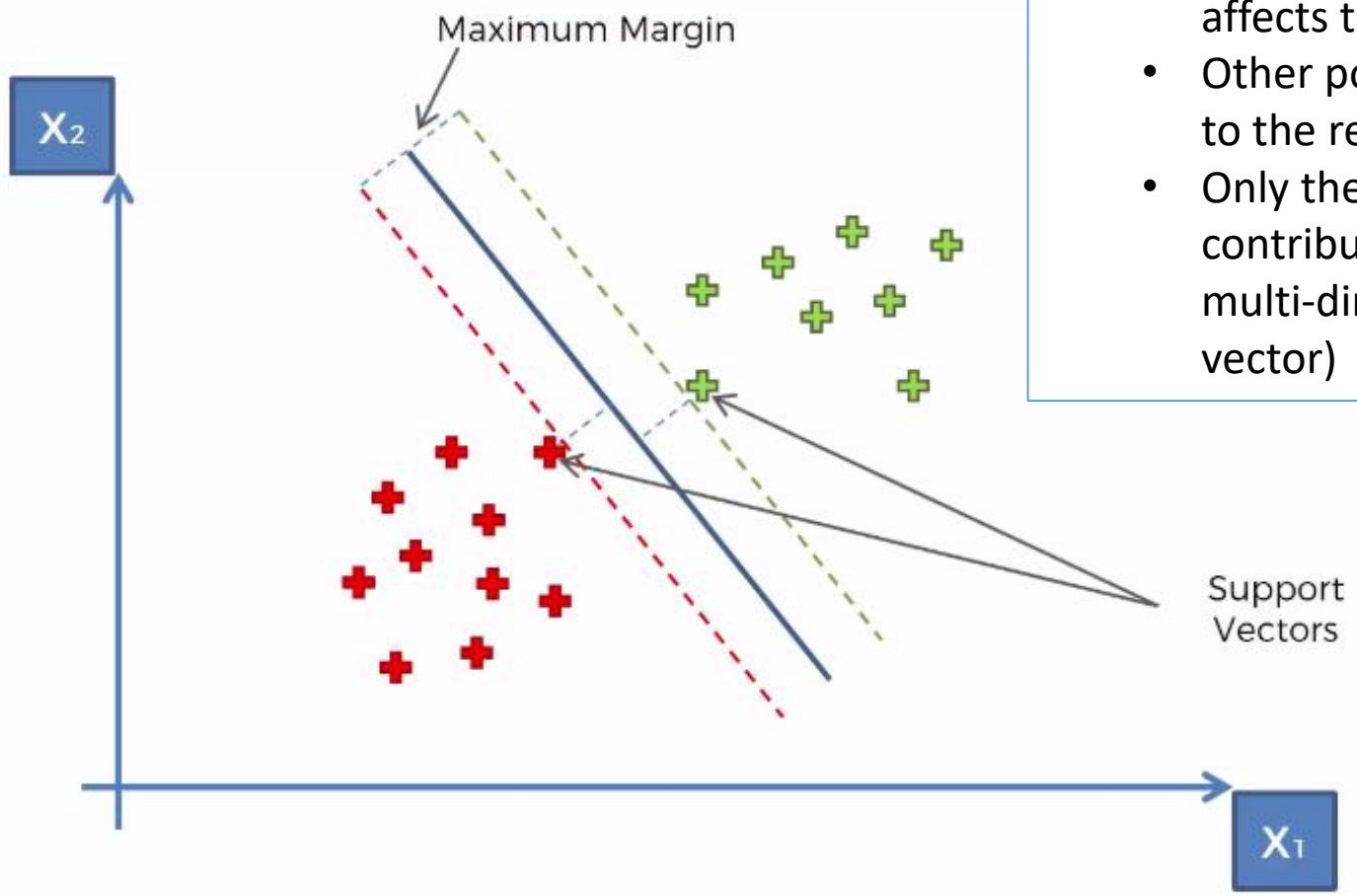
One with maximum margin is selected.



Which hyperplane does it select?

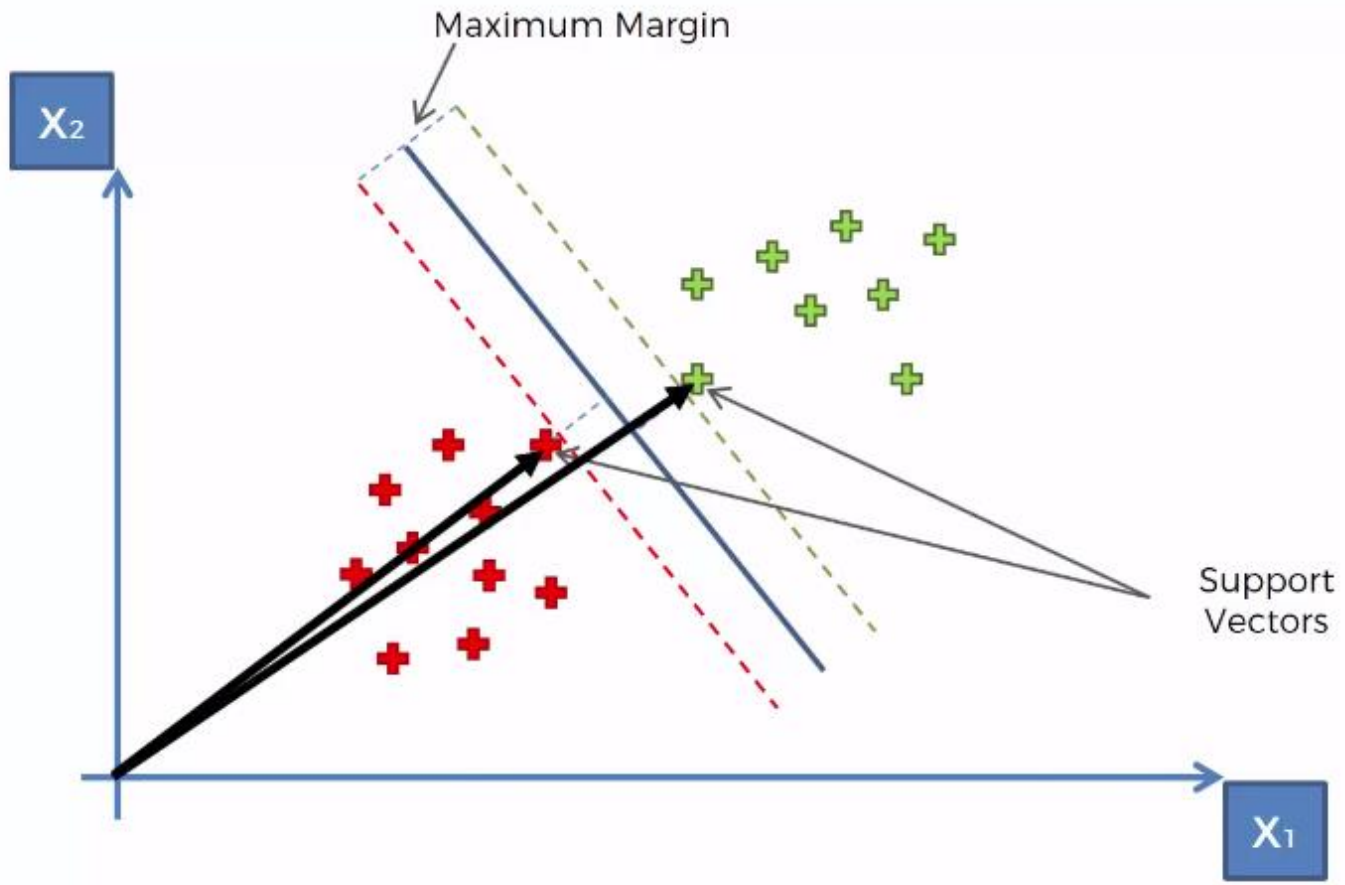
- Find the maximum margin between the hyperplanes that means maximum distances between the two classes.

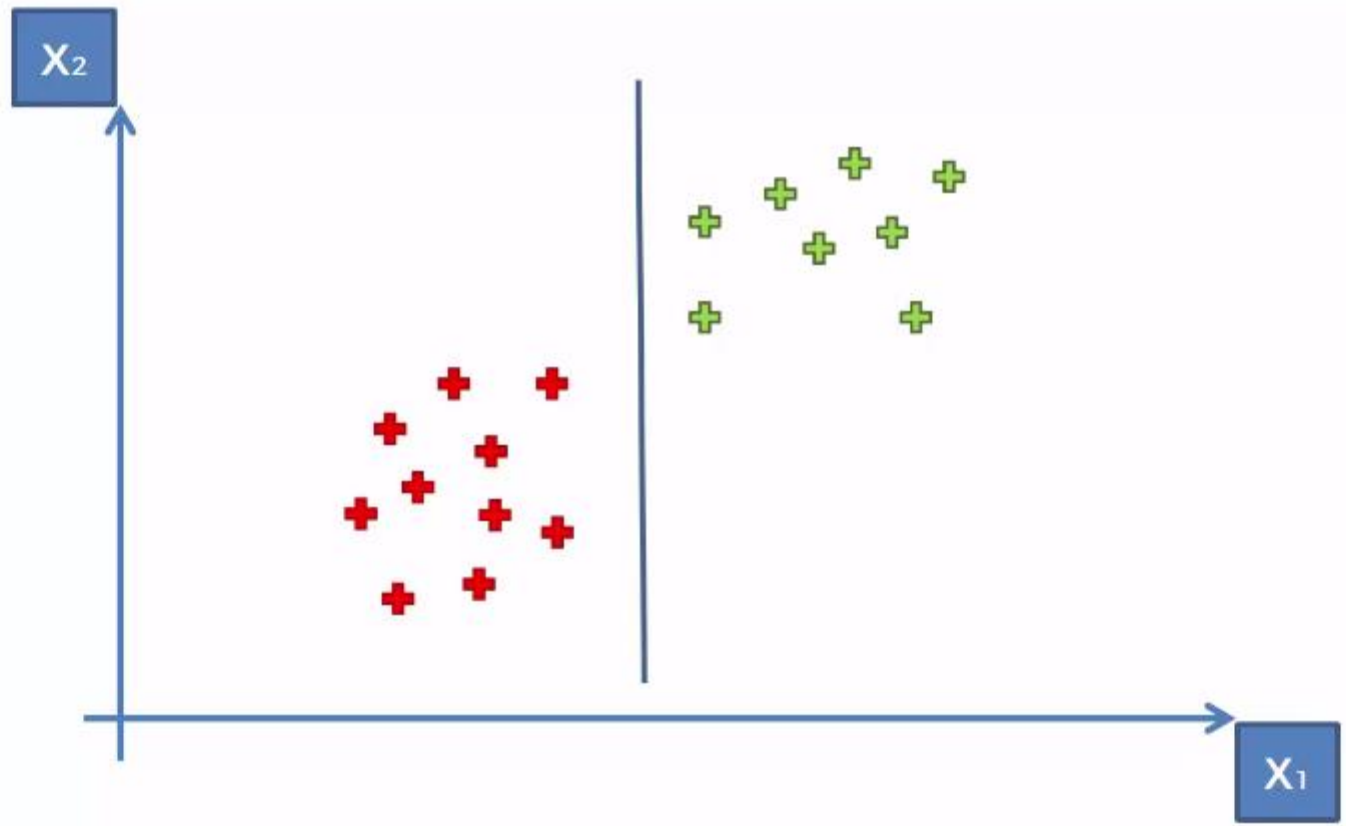
Support Vectors

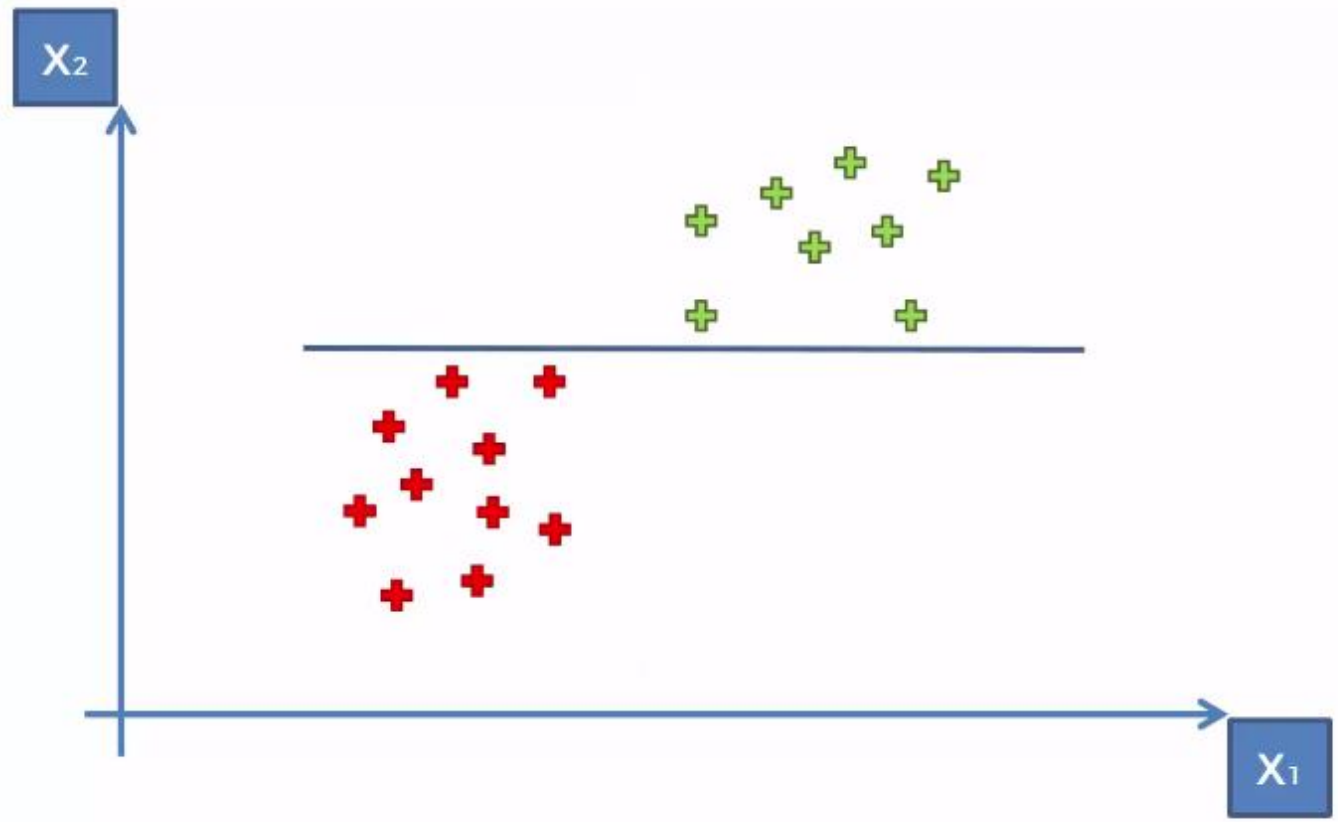


- The two points are called support vectors
  - Supporting the whole algorithm
  - Even other points changes, not affects the algorithm
  - Other points are not contributing to the result of the algorithm
  - Only these two points are contributing (support vectors – in multi-dimension point is called vector)

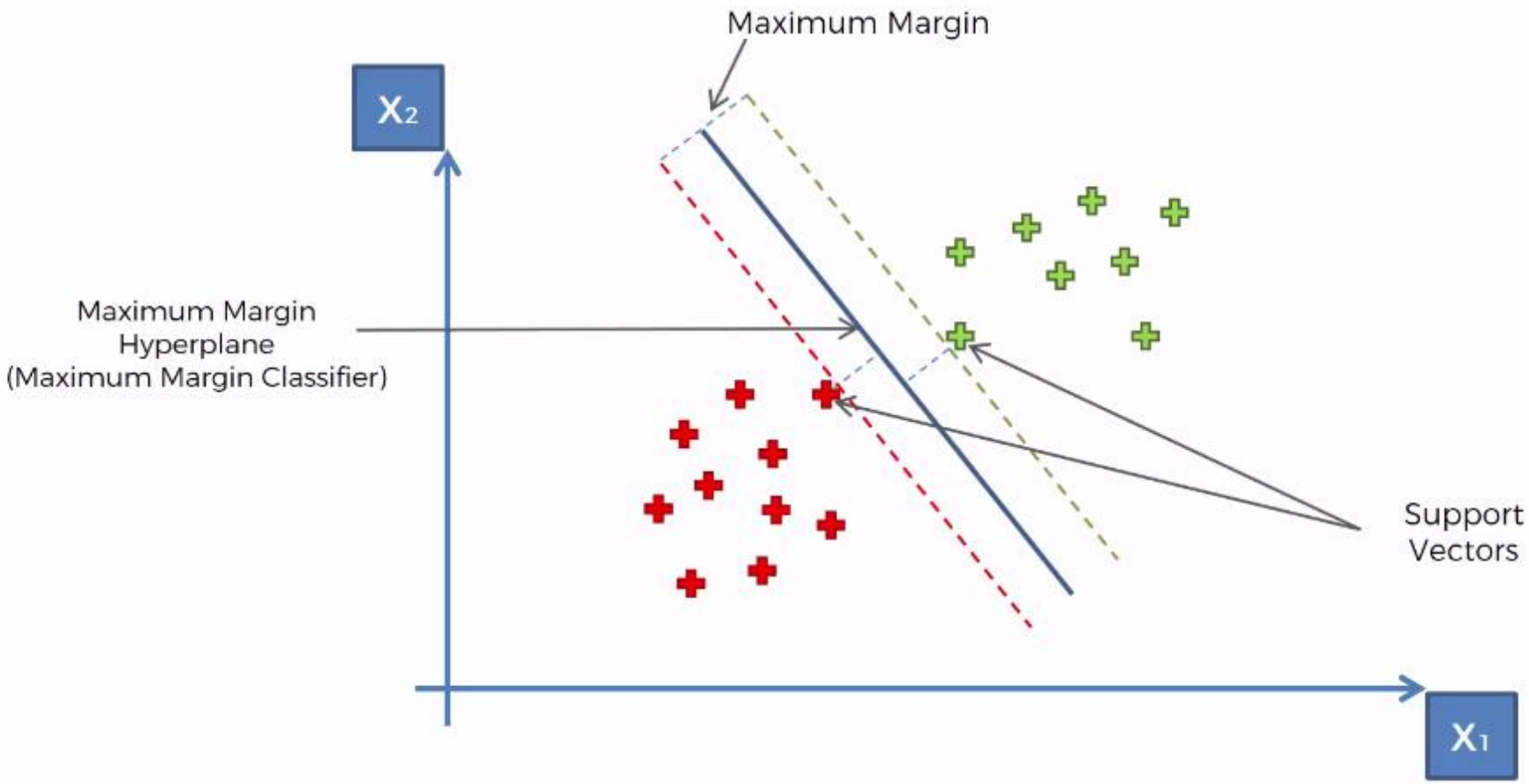
Support Vectors



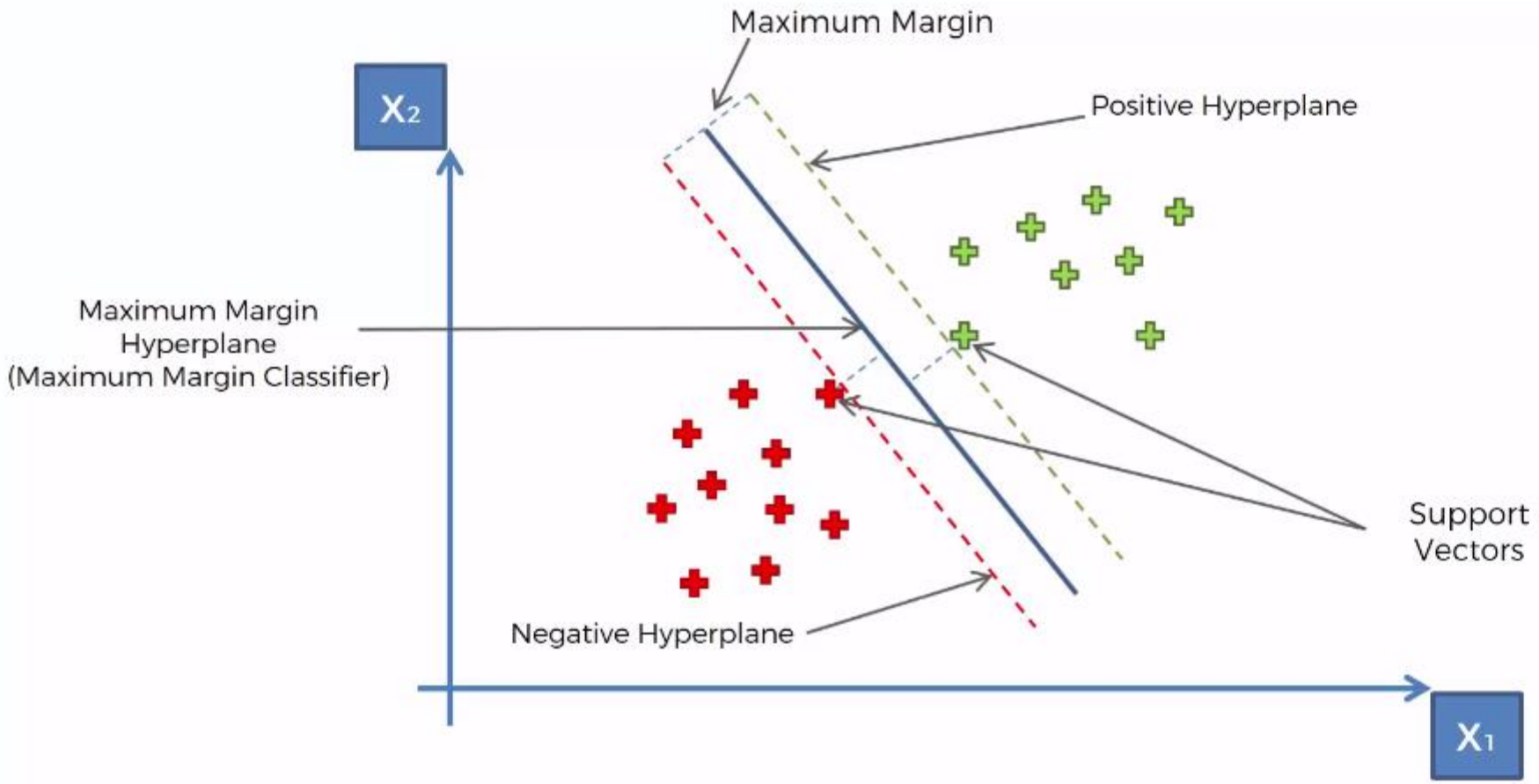

X2
X1


X2
X1

Hyperplanes



Hyperplanes



Types of Support Vector Machine (SVM) Algorithms

- **Linear SVM**:
  - When the data is perfectly linearly separable only then we can use Linear SVM.
  - Perfectly linearly separable means that the data points can be classified into 2 classes by using a single straight line (if 2D).

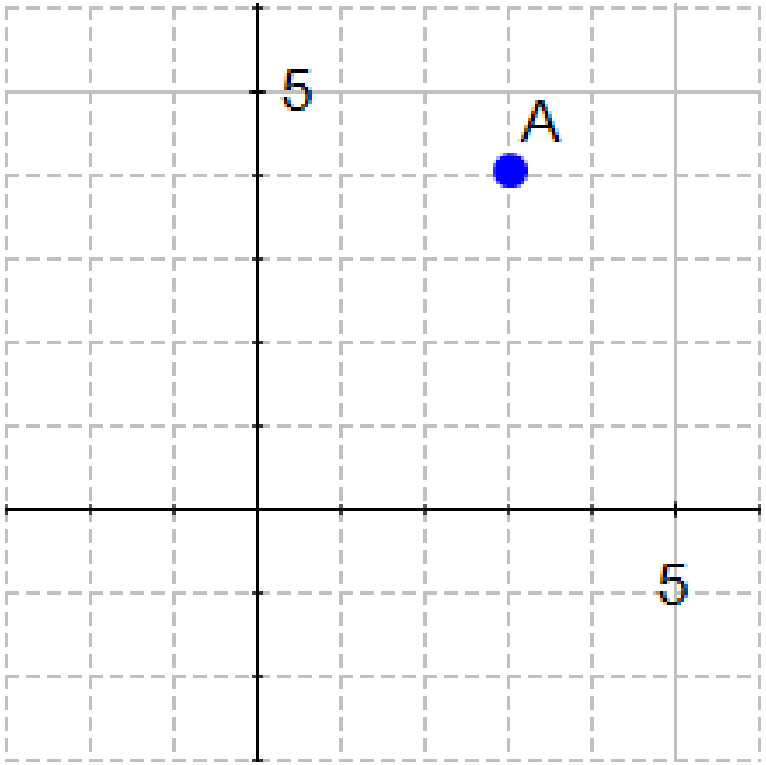
- **Non-Linear SVM**:
  - When the data is not linearly separable then we can use Non-Linear SVM, which means when the data points cannot be separated into 2 classes by using a straight line (if 2D) then we use some advanced techniques like kernel tricks to classify them.
  - In most real-world applications we do not find linearly separable datapoints hence we use kernel trick to solve them.

Mathematical Intuition Behind Support Vector Machine

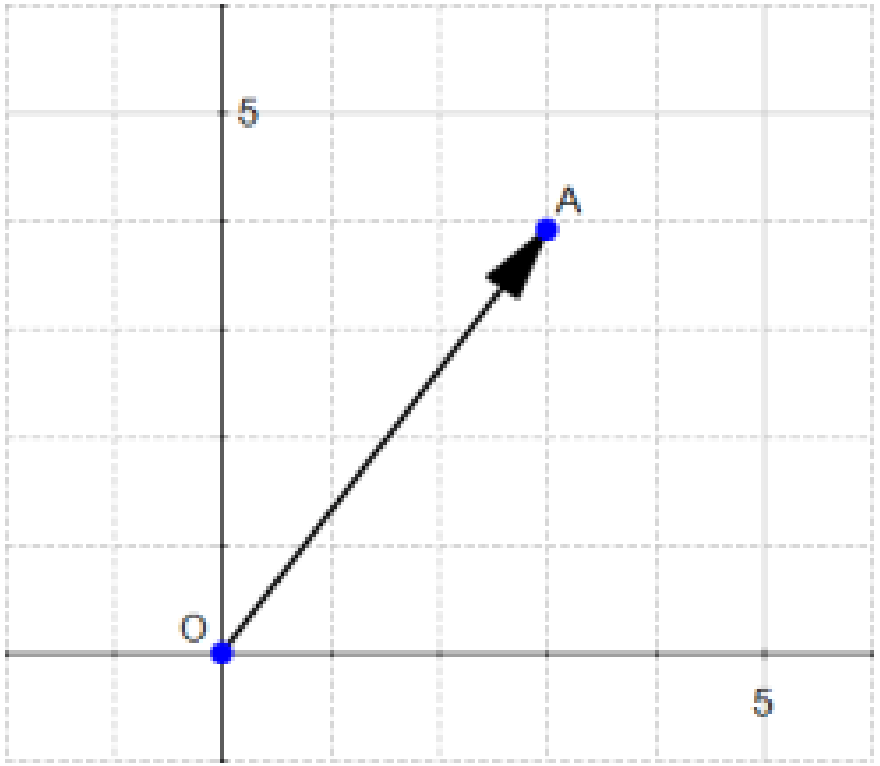
Vector

A vector is a quantity that has magnitude as well as direction and just like numbers we can use mathematical operations such as addition, multiplication.

If we define a point $A(3,4)$ in $\mathbb{R}^2$ we can plot it like this.



A vector in the plane, starting at the origin and ending at A



The magnitude

The magnitude or length of a vector X is written $\|X\|$ and is called its norm.

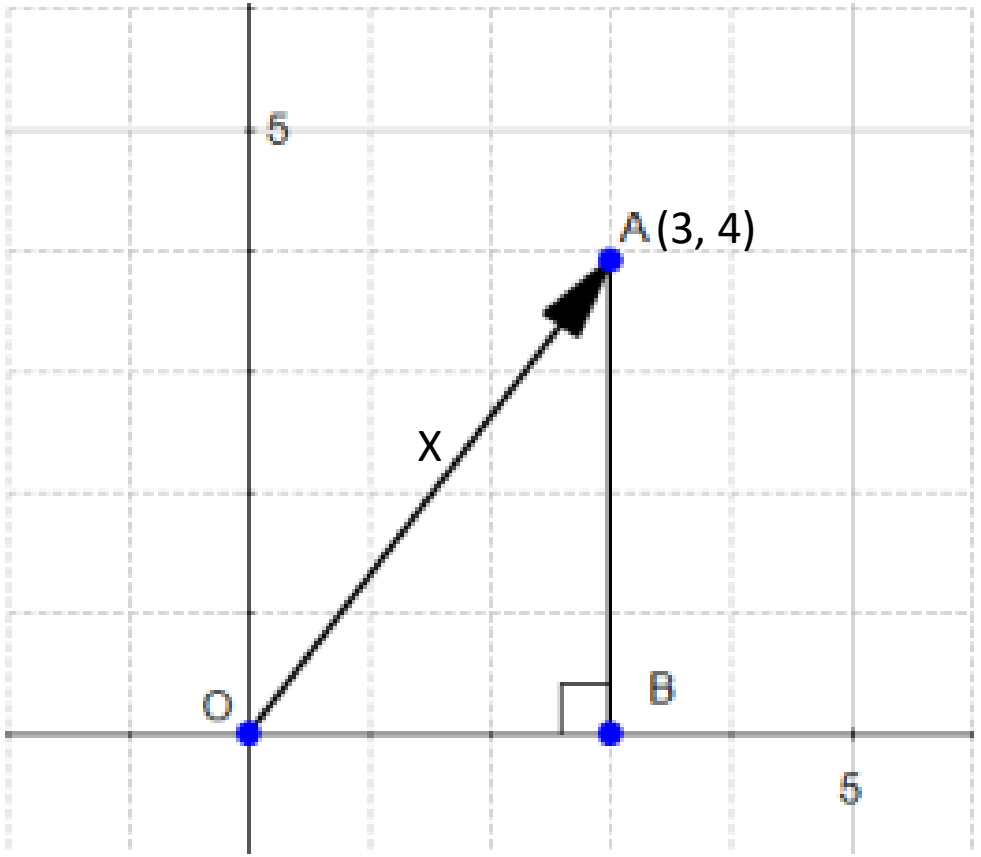
$$OA^2 = OB^2 + AB^2$$

$$OA^2 = 3^2 + 4^2$$

$$OA^2 = 25$$

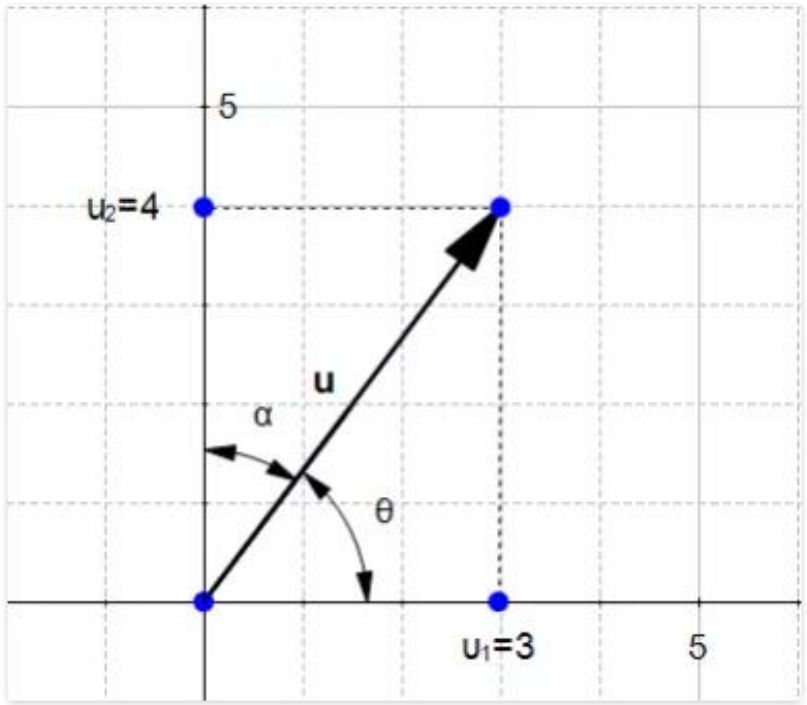
$$OA = \sqrt{25}$$

$$\|OA\| = OA = 5$$



The direction

*The **direction** of a vector* $\mathbf{u}(u_1, u_2)$ *is the vector* $\mathbf{w}\left(\frac{u_1}{\|u\|}, \frac{u_2}{\|u\|}\right)$



Understanding the definition

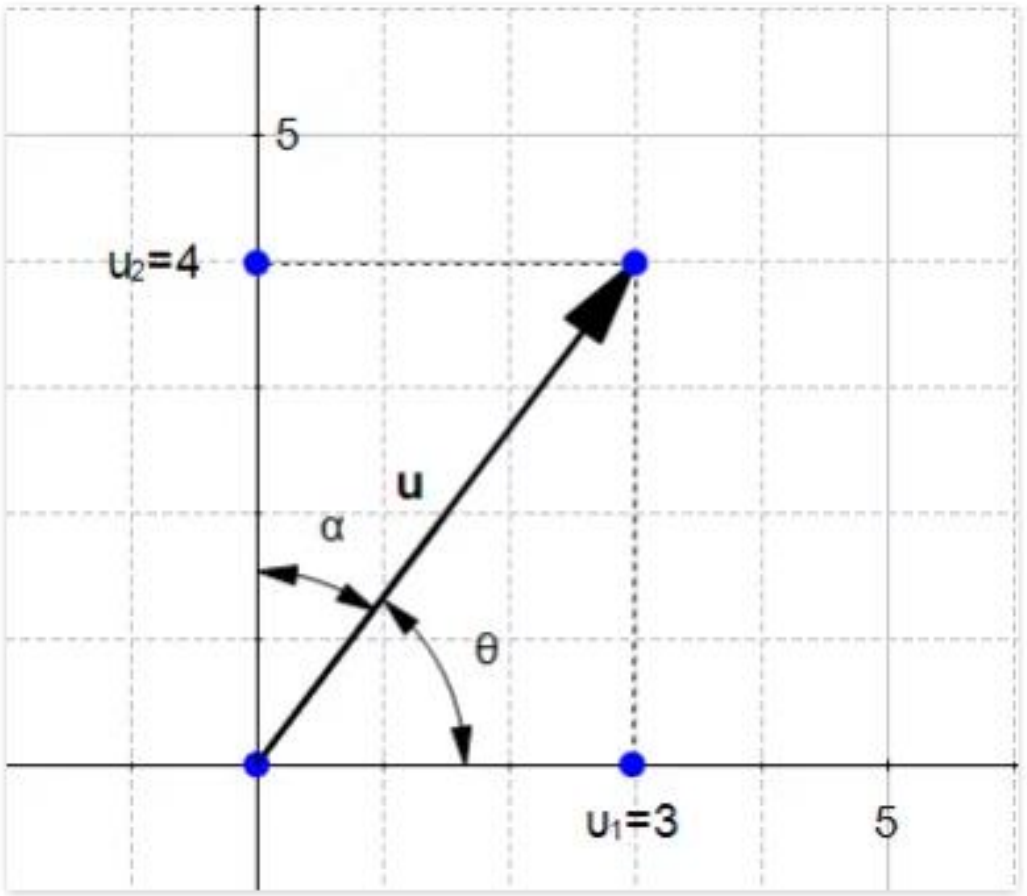
To find the direction of a vector, we need to use its angles.

vector $\mathbf{u}(u_1, u_2)$ with

$u_1 = 3$ and $u_2 = 4$

$$cos(\theta) = \frac{u_1}{\|u\|}$$

$$cos(\alpha) = \frac{u_2}{\|u\|}$$



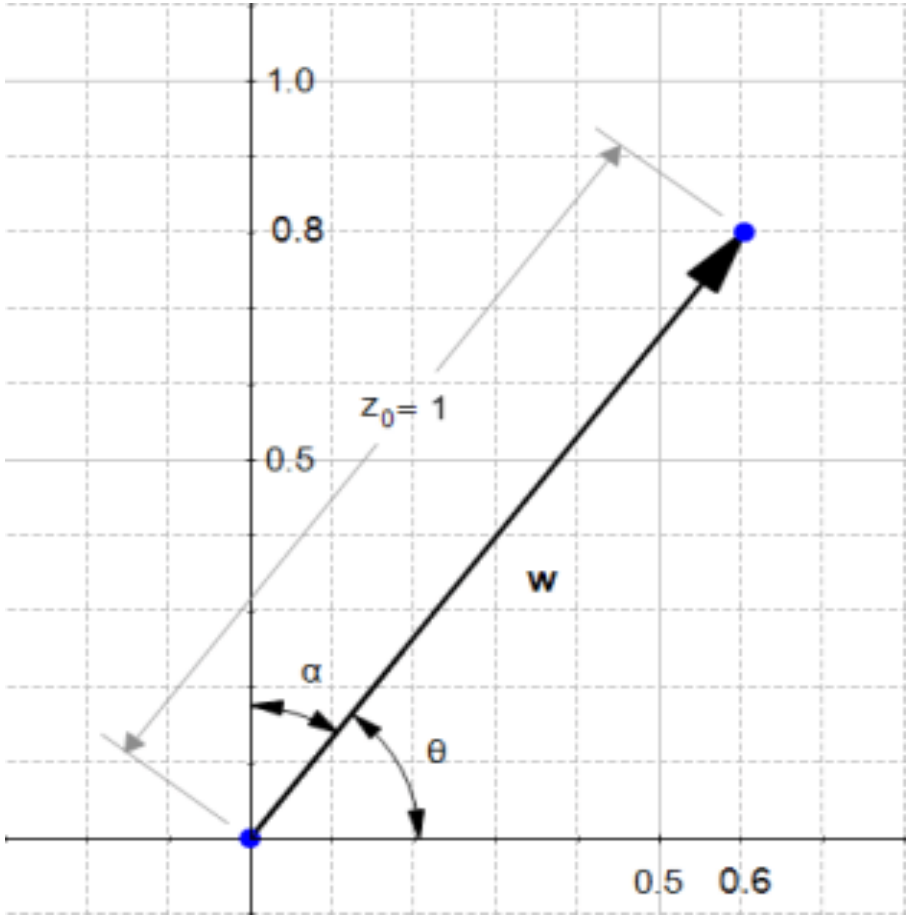
Computing the direction vector

We will now compute the direction of the vector $\mathbf{u}$ from Figure

$$cos(\theta) = \frac{u_1}{\|u\|} = \frac{3}{5} = 0.6$$

$$cos(\alpha) = \frac{u_2}{\|u\|} = \frac{4}{5} = 0.8$$

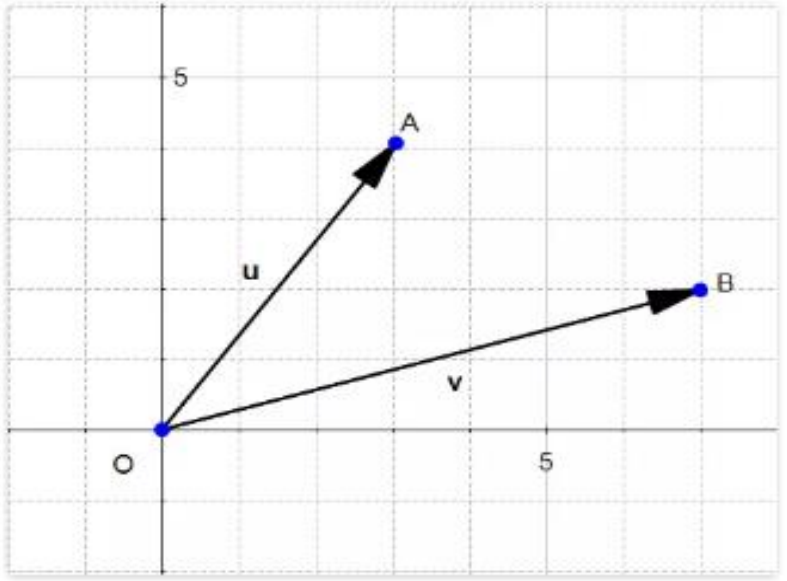
The direction of $\mathbf{u}(3,4)$ is the vector $\mathbf{w}(0.6, 0.8)$



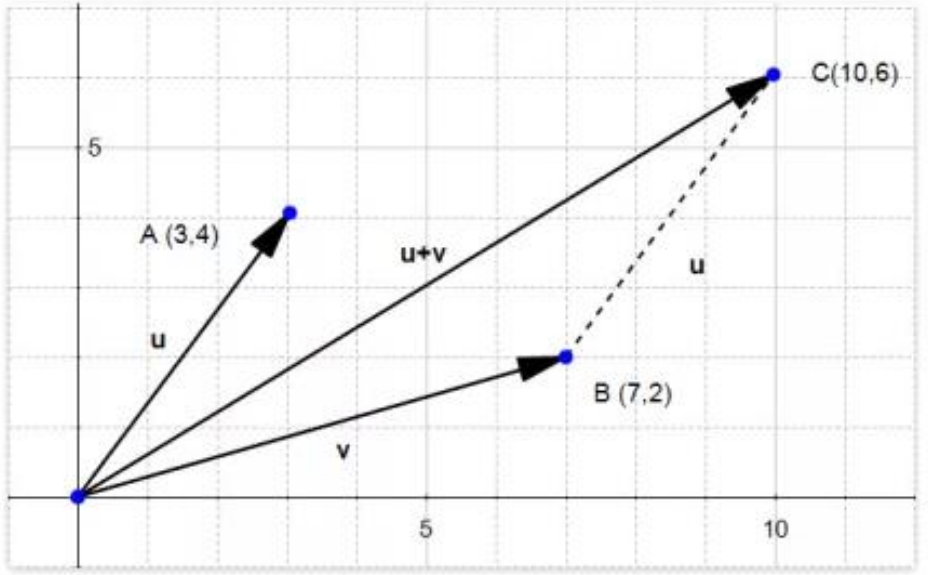
The sum of two vectors

Given two vectors $\mathbf{u}(u_1, u_2)$ and $\mathbf{v}(v_1, v_2)$ then :

$$\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2)$$

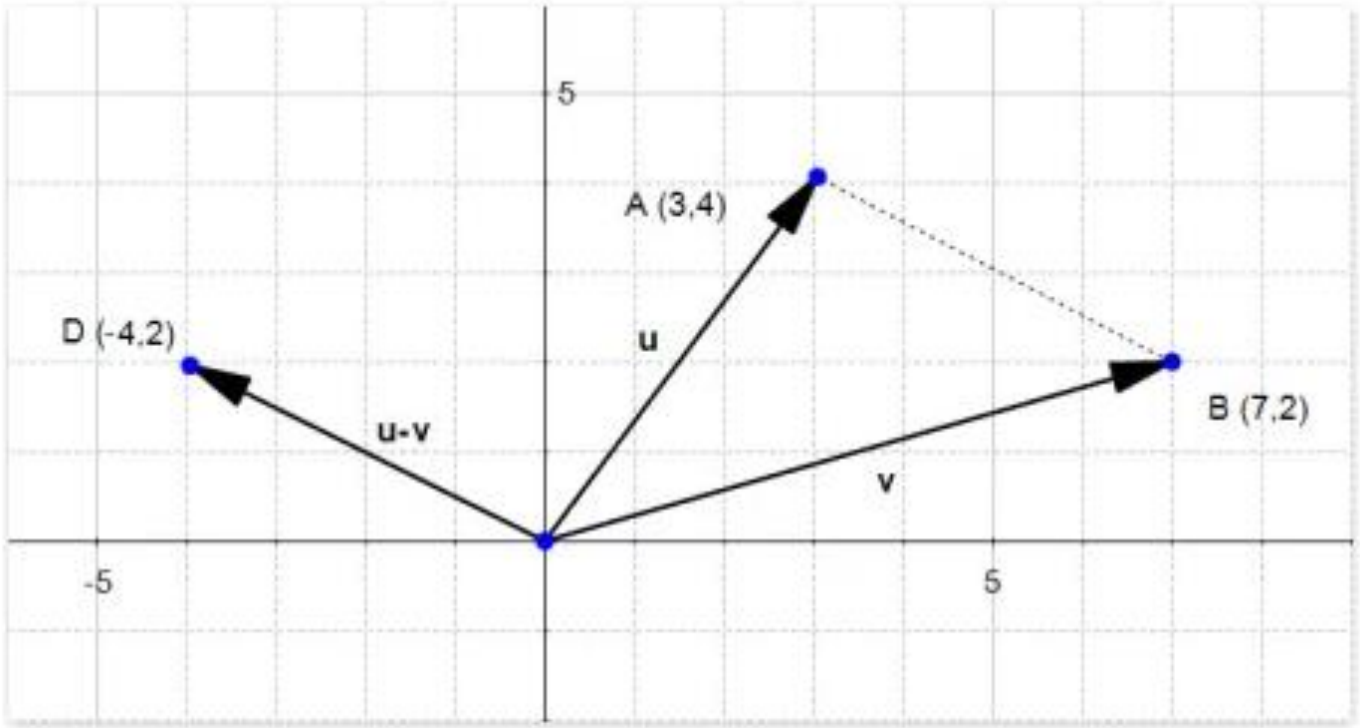


Which means that adding two vectors gives us **a third vector** whose coordinate are the sum of the coordinates of the original vectors.

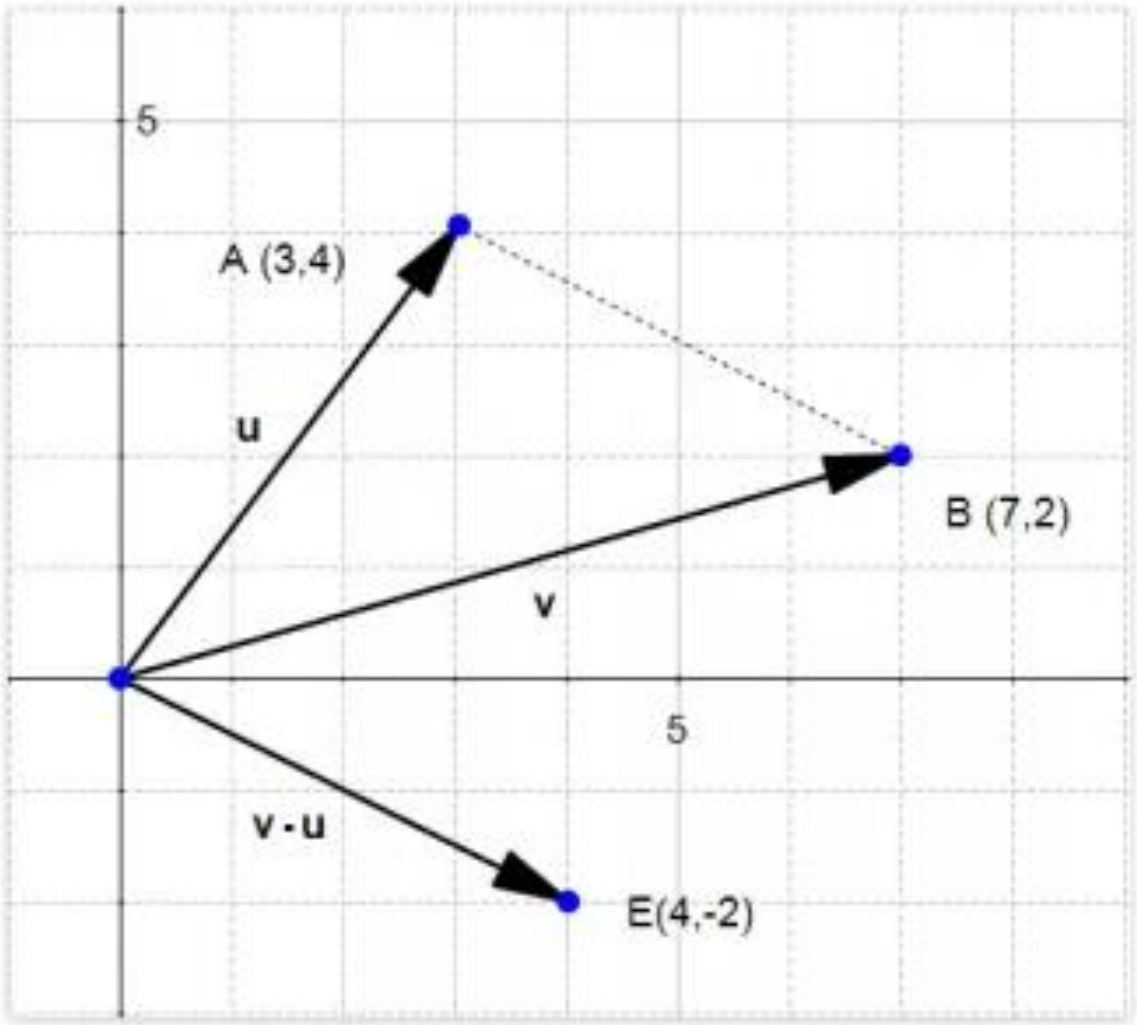


The difference between two vectors

$$\mathbf{u} - \mathbf{v} = (u_1 - v_1, u_2 - v_2)$$



$$\mathbf{v} - \mathbf{u} = (v_1 - u_1, v_2 - u_2)$$



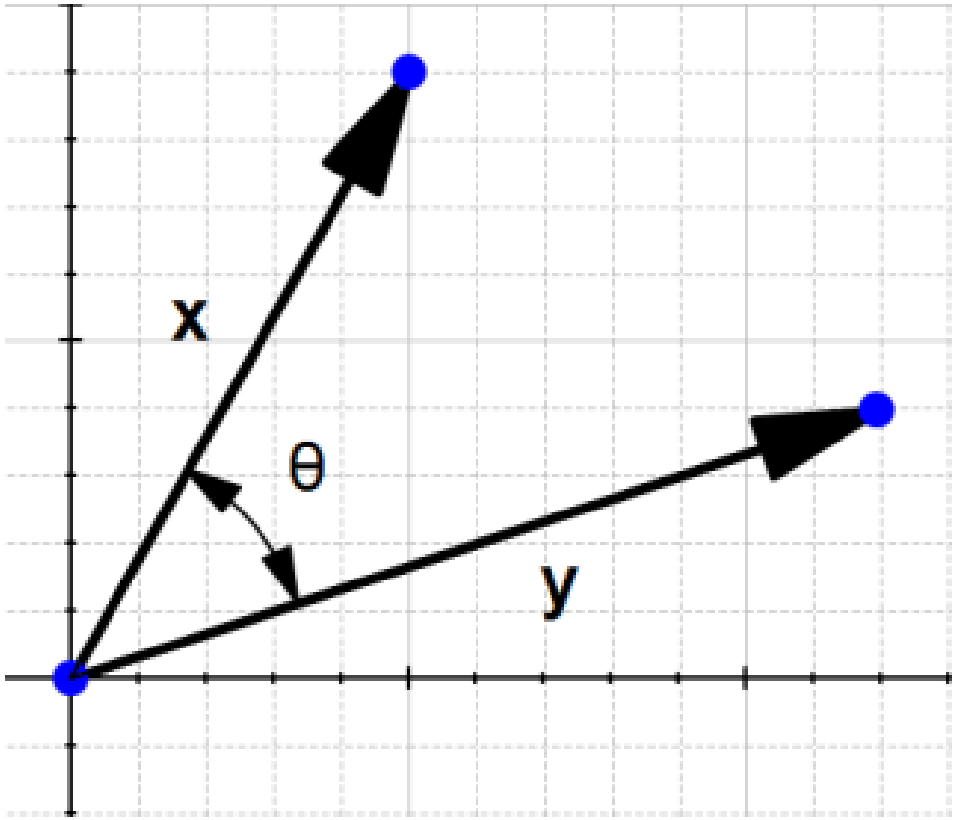
The dot product

- One **very** important notion to understand SVM is the dot product.

- If we have two vectors **x** and **y** and there is an angle **θ** between them, their **dot** product is :

$$x \cdot y = \|x\| \, \|y\| \cos(\theta)$$

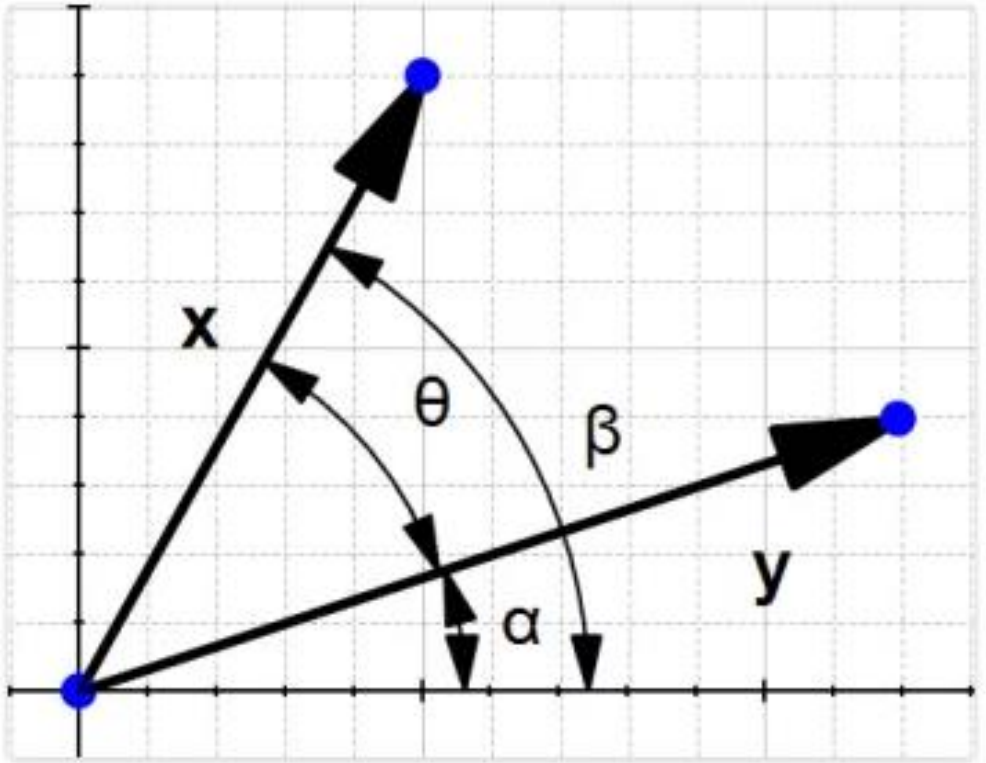
$$\cos(\theta) = \frac{adjacent}{hypotenuse}$$



$x \cdot y = \|x\| \, \|y\| \cos(\theta)$

$$\theta = \beta - \alpha$$

$$cos(\beta - \alpha) = cos(\beta)cos(\alpha) + sin(\beta)sin(\alpha)$$



$$cos(\beta) = \frac{adjacent}{hypotenuse} = \frac{x_1}{\|x\|}$$

$$sin(\beta) = \frac{opposite}{hypotenuse} = \frac{x_2}{\|x\|}$$

$$cos(\alpha) = \frac{adjacent}{hypotenuse} = \frac{y_1}{\|y\|}$$

$$sin(\alpha) = \frac{opposite}{hypotenuse} = \frac{y_2}{\|y\|}$$

$$cos(\theta) = cos(\beta - \alpha) = cos(\beta)cos(\alpha) + sin(\beta)sin(\alpha)$$

$$cos(\theta) = \frac{x_1}{\|x\|}\frac{y_1}{\|y\|} + \frac{x_2}{\|x\|}\frac{y_2}{\|y\|}$$

$$cos(\theta) = \frac{x_1y_1 + x_2y_2}{\|x\|\|y\|}$$

If we multiply both sides by $\|x\|\|y\|$ we get:

$$\|x\|\|y\|cos(\theta) = x_1y_1 + x_2y_2$$

Which is the same as :

$$\|x\|\|y\|cos(\theta) = \mathbf{x} \cdot \mathbf{y}$$

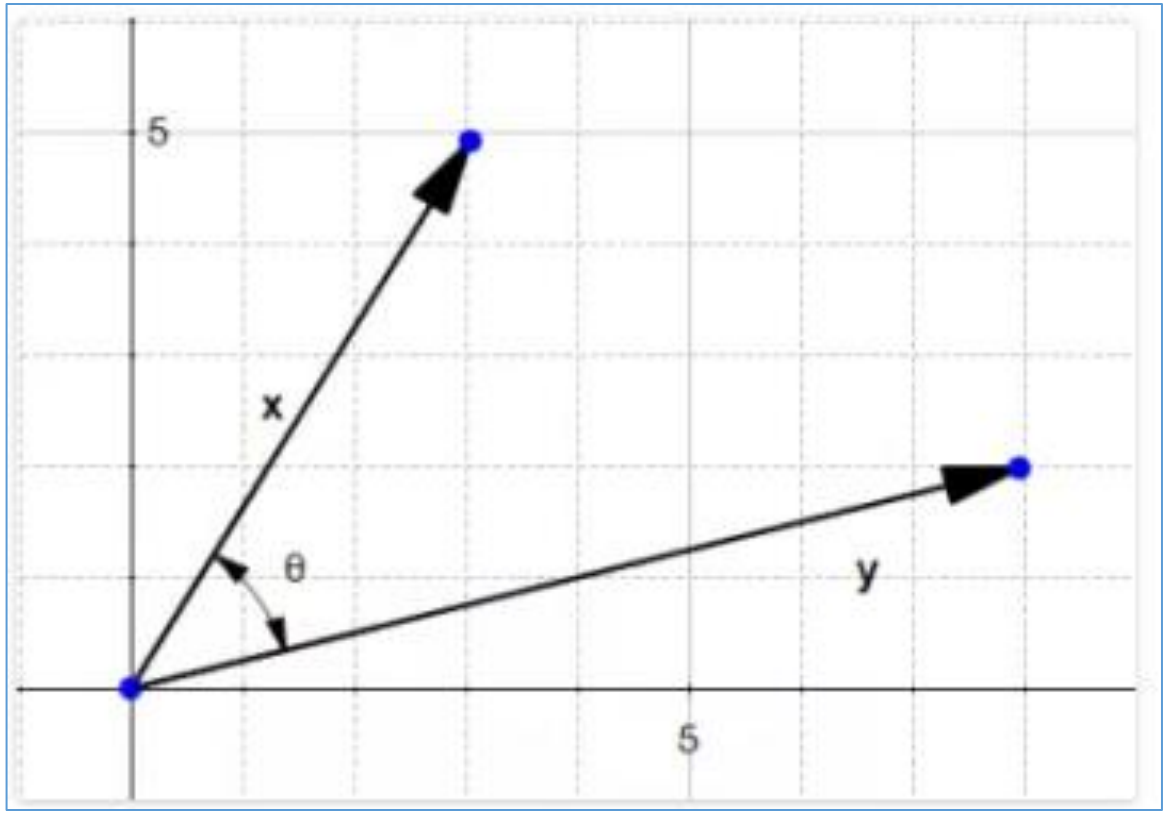
We just found the geometric definition of the dot product !

Eventually from the two last equations we can see that :

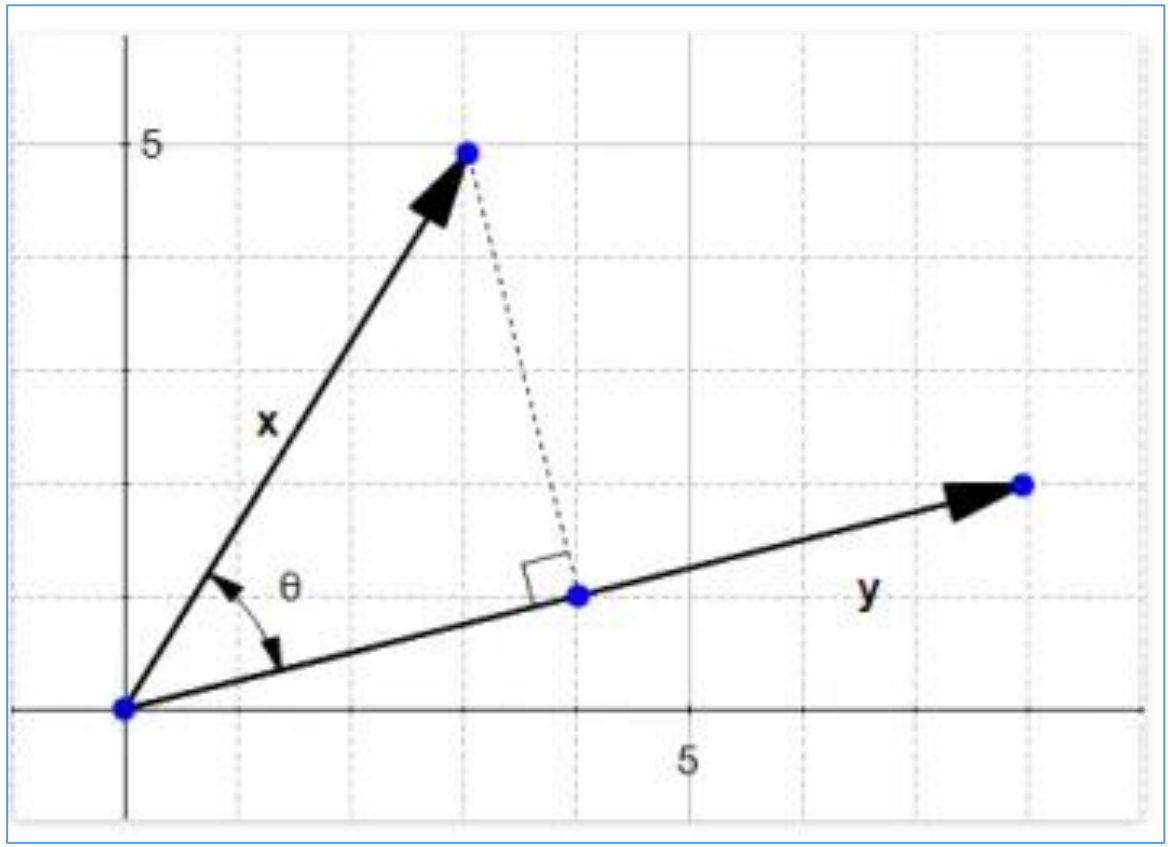
$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 = \sum_{i=1}^{2} (x_i y_i)$$

The orthogonal projection of a vector

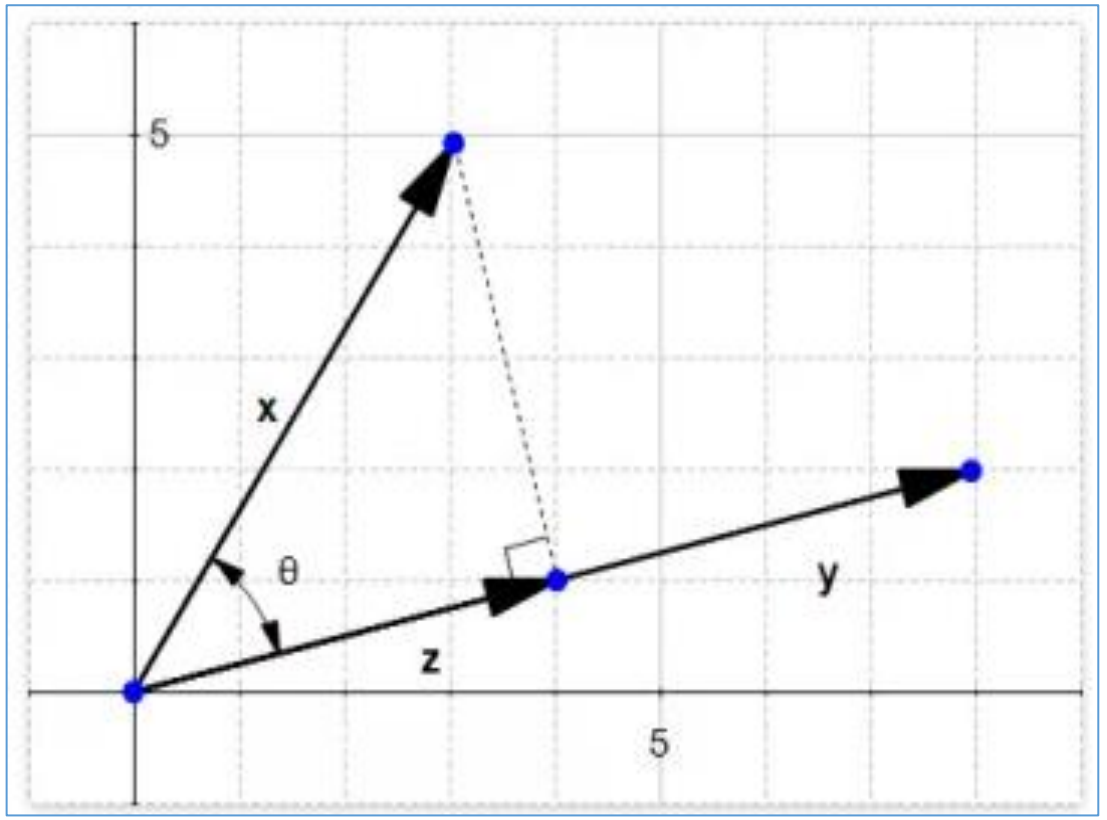
Given two vectors **x** and **y**, we would like to find the orthogonal projection of **x** onto **y**:



Project the vector x onto y:



This gives us the vector **z**:



$$\cos(\theta) = \frac{||z||}{||x||} \qquad ||x|| \cos(\theta) = ||z|| \qquad \cos(\theta) = \frac{x.y}{||x|| \, ||y||}$$

We replace cos(Θ) in equation:

$$||x|| \cos(\theta) = ||z||$$

$$\cos(\theta) = \frac{x.y}{||x|| \, ||y||}$$

$$||z|| = ||x|| \frac{x.y}{||x|| \, ||y||}$$

$$||z|| = \frac{x.y}{||y||}$$

If we define the vector **u** as the direction of **y** then

$$\mathbf{u} = \frac{y}{||y||}$$

Since this vector **z** is in the same direction as **y** it has the direction **u**

$$\mathbf{u} = \frac{z}{||z||}$$

$$z = ||z|| \, \mathbf{u}$$

And we can say:
The vector **z** = (**u** . **x**) **u** is the orthogonal projection of **x** onto **y.**

SVM Hyperplane

Understanding the equation of the Hyperplane

Equation of a line is $\boldsymbol{y = ax + b}$

Equation of a hyperplane is defined by $\boldsymbol{w^T x = 0}$

How does these two forms relate ?

$\boldsymbol{y = ax + b}$

$\boldsymbol{y - ax - b = 0}$

$Given\ two\ vectors\ \mathbf{w} = \begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$ and $\mathbf{x} = \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$\boldsymbol{w^T x = -b * (1) + (-a) * x + 1 * y}$

$\boldsymbol{w^T x = y - ax - b}$

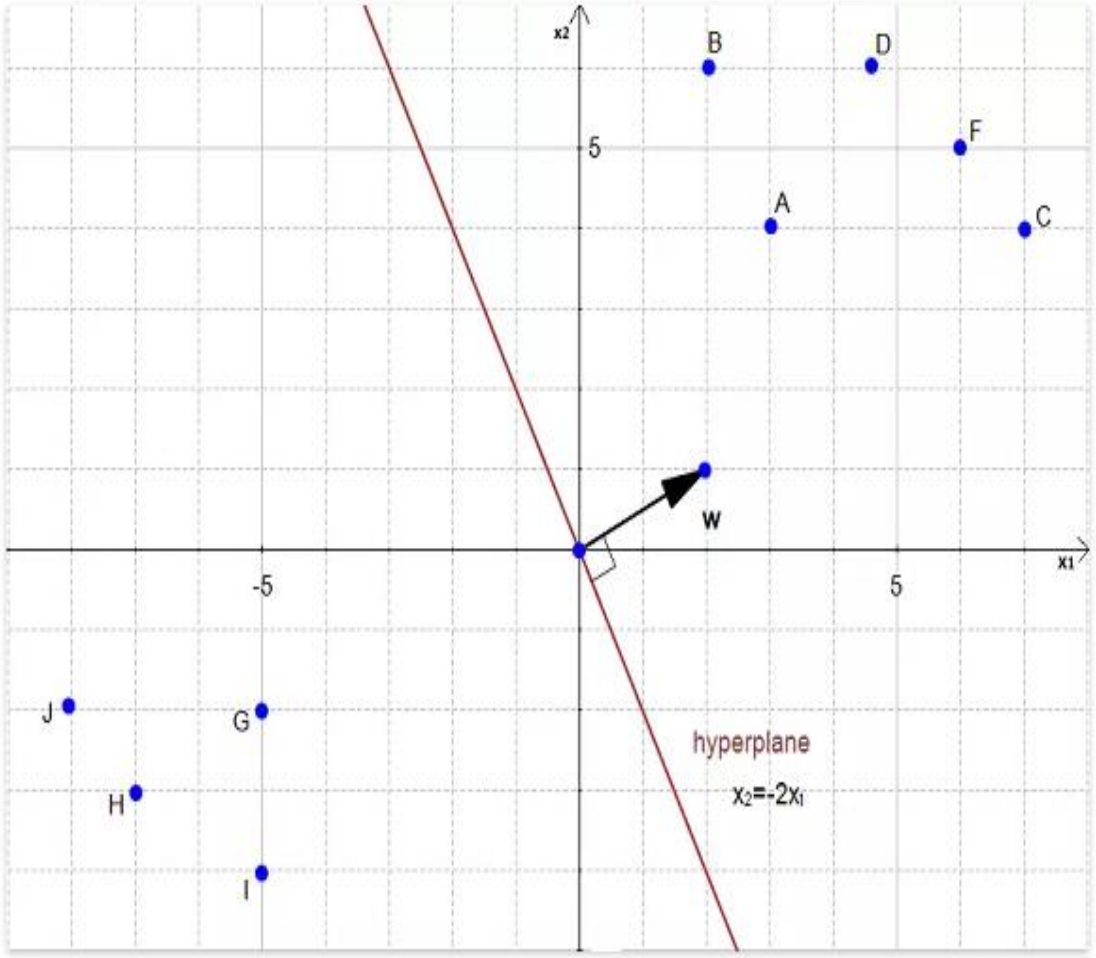
The two equations are just different ways of expressing the same thing.

$w^T x = 0$ is preferred for the following two reasons:

- it is easier to work in **more than two dimensions** with this notation
- the vector **W** will always be **normal to the Hyperplane**

Compute the distance from a point to the hyperplane

We have a Hyperplane, which separates two group of data.



Equation of the Hyperplane is $w^T x = 0$

To simplify this example, we have set $w_0 = 0$.

$$x_2 = -2x_1$$

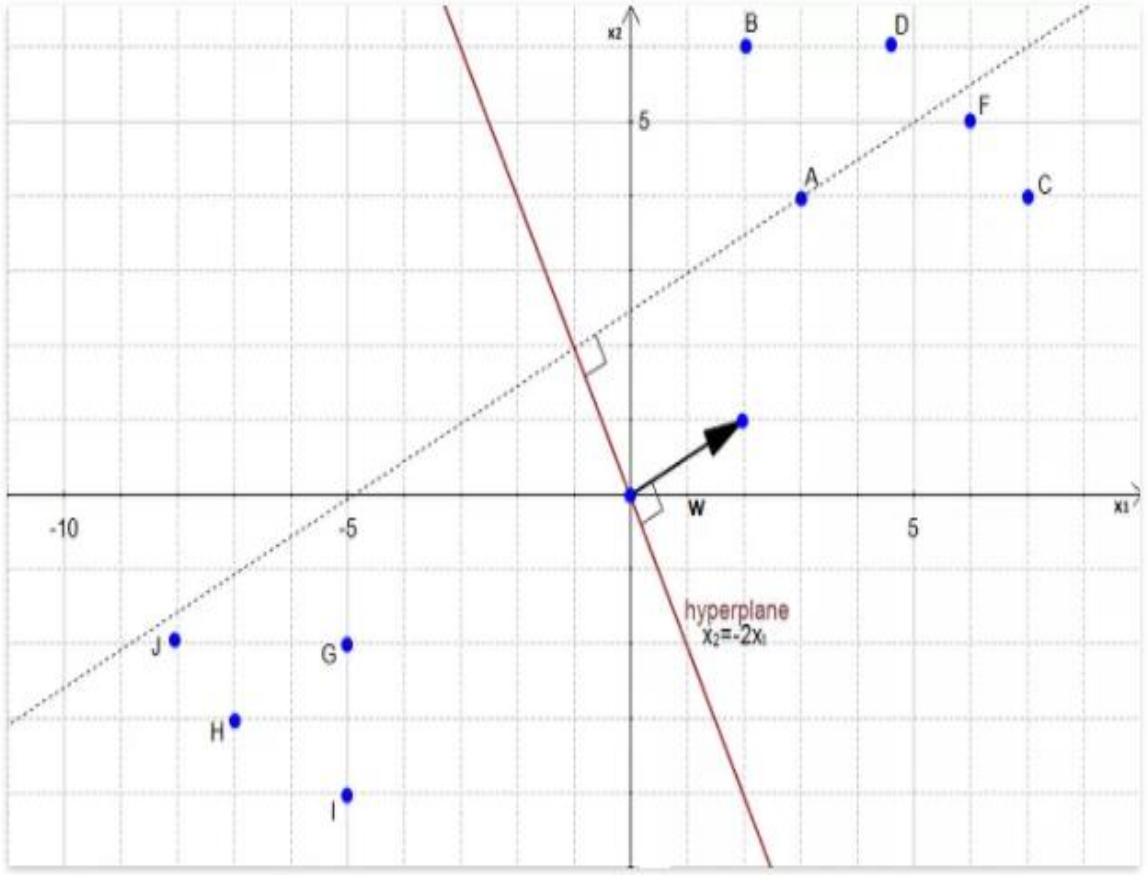
$$x_2 + 2x_1 = 0$$

$$w^T x = 0$$

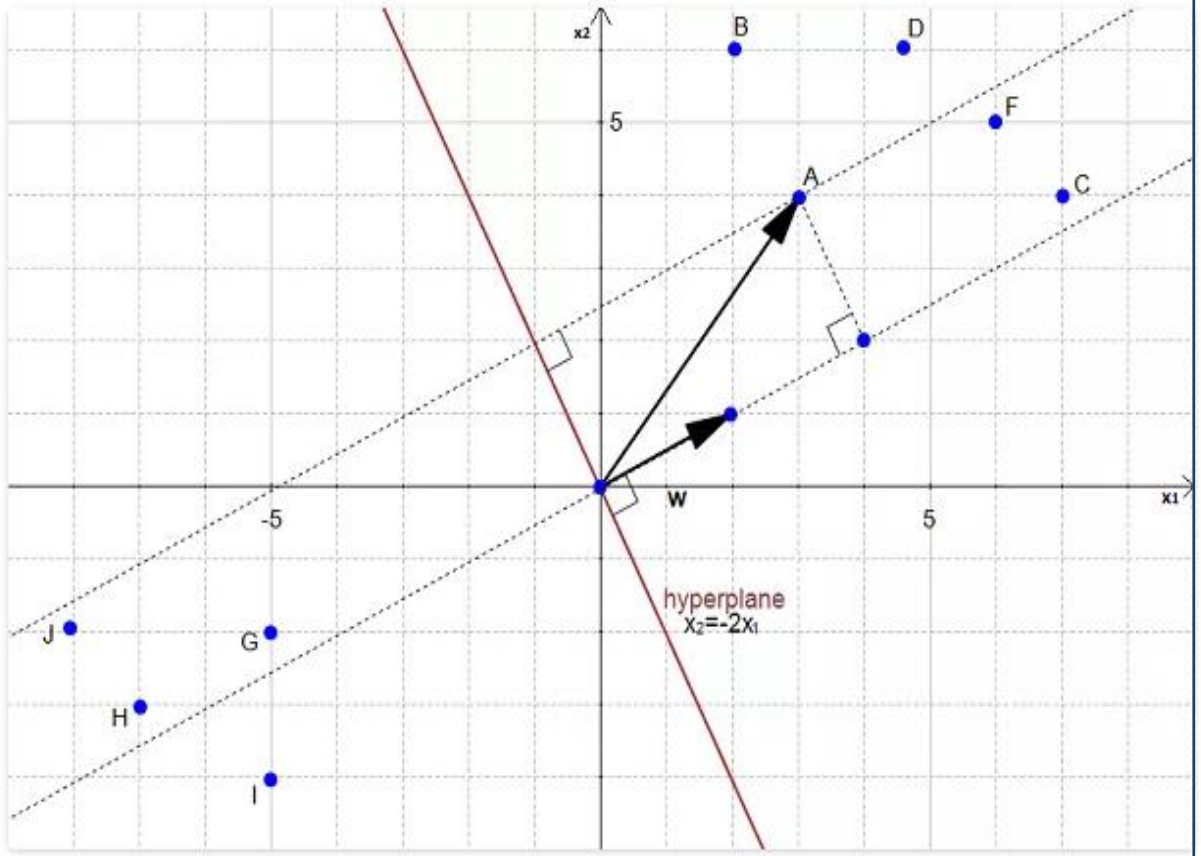
From these two vectors

$w = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ and $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

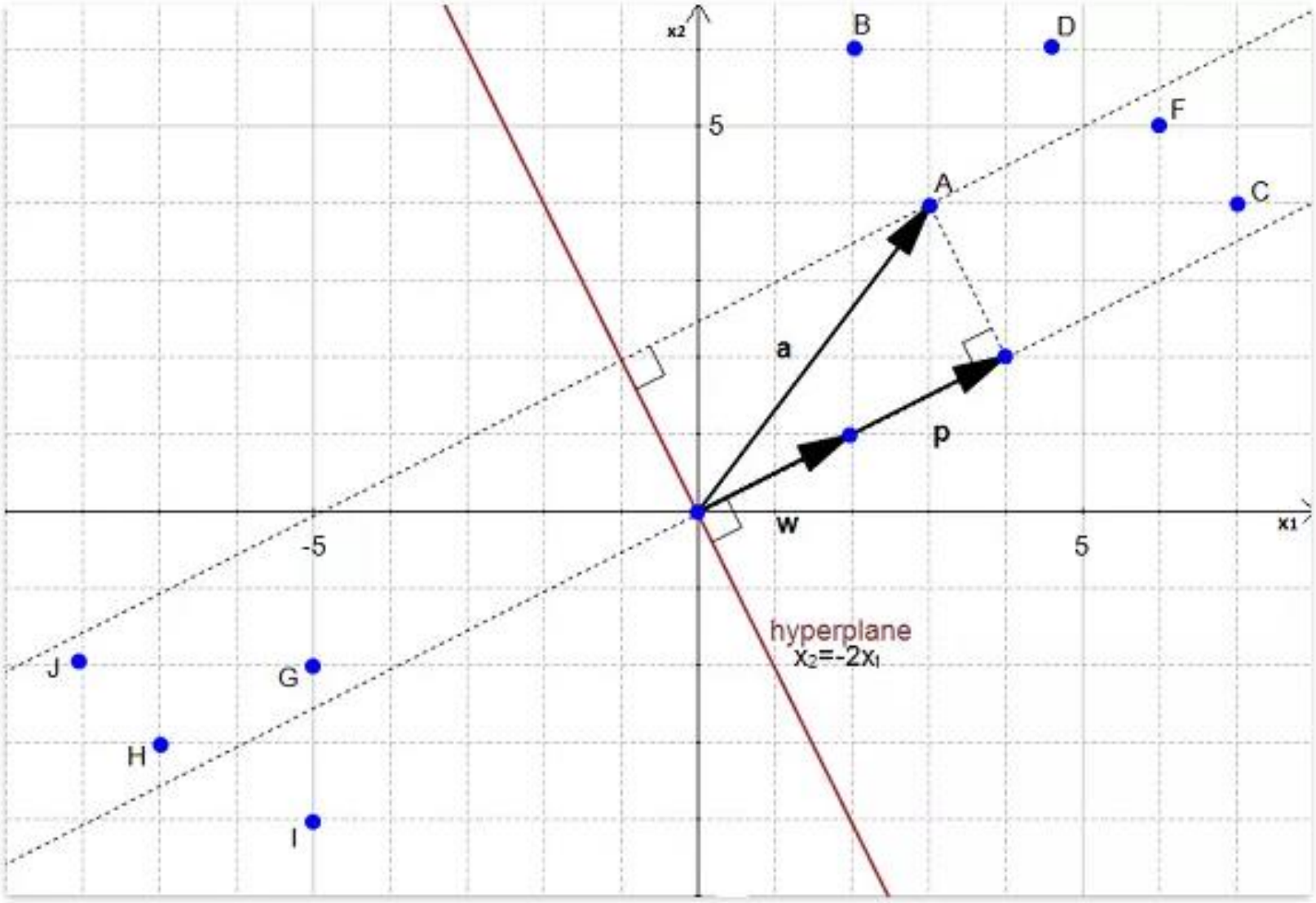
- We would like to compute the distance between the point **A(3, 4)** and the **hyperplane**.
- The dashed line is the distance between **A** and its projection onto the **hyperplane**.



- We can view the point **A** as a vector from the origin to **A**. if we project it onto the normal vector **w**.



We get the vector **p**.



- Our goal is to find the distance between the point **A(3, 4)** and the hyperplane.
- We can see figure that this distance is the same thing as **||p||**.
- We start with two vectors, **w = (2, 1)** which is normal to the hyperplane, and **a=(3, 4)** which is the vector between the origin and A.

Magnitude $||w|| = \sqrt{2^2 + 1^2} = \sqrt{5}$

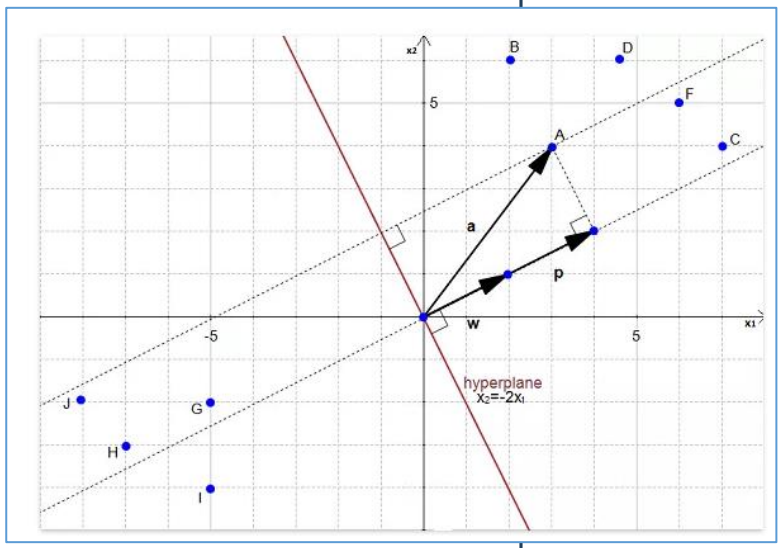
Let the vector **u** be the direction of **w**

$$u = (\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}})$$

p is the orthogonal projection of **a** onto **w**

$$p = (u.a)u$$

$$p = \left((\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}})(3,4)\right)u$$



$$p = \left((\frac{2}{\sqrt{5}} * 3 + \frac{1}{\sqrt{5}} * 4)\right)u$$

$$p = \left((\frac{6}{\sqrt{5}} + \frac{4}{\sqrt{5}})\right)u$$

$$p = \left(\frac{10}{\sqrt{5}}\right)u$$

$$p = \left(\frac{10}{\sqrt{5}}\right) * (\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}})$$

$$p = \left(\frac{20}{5}, \frac{10}{5}\right) \qquad p = (4,2)$$

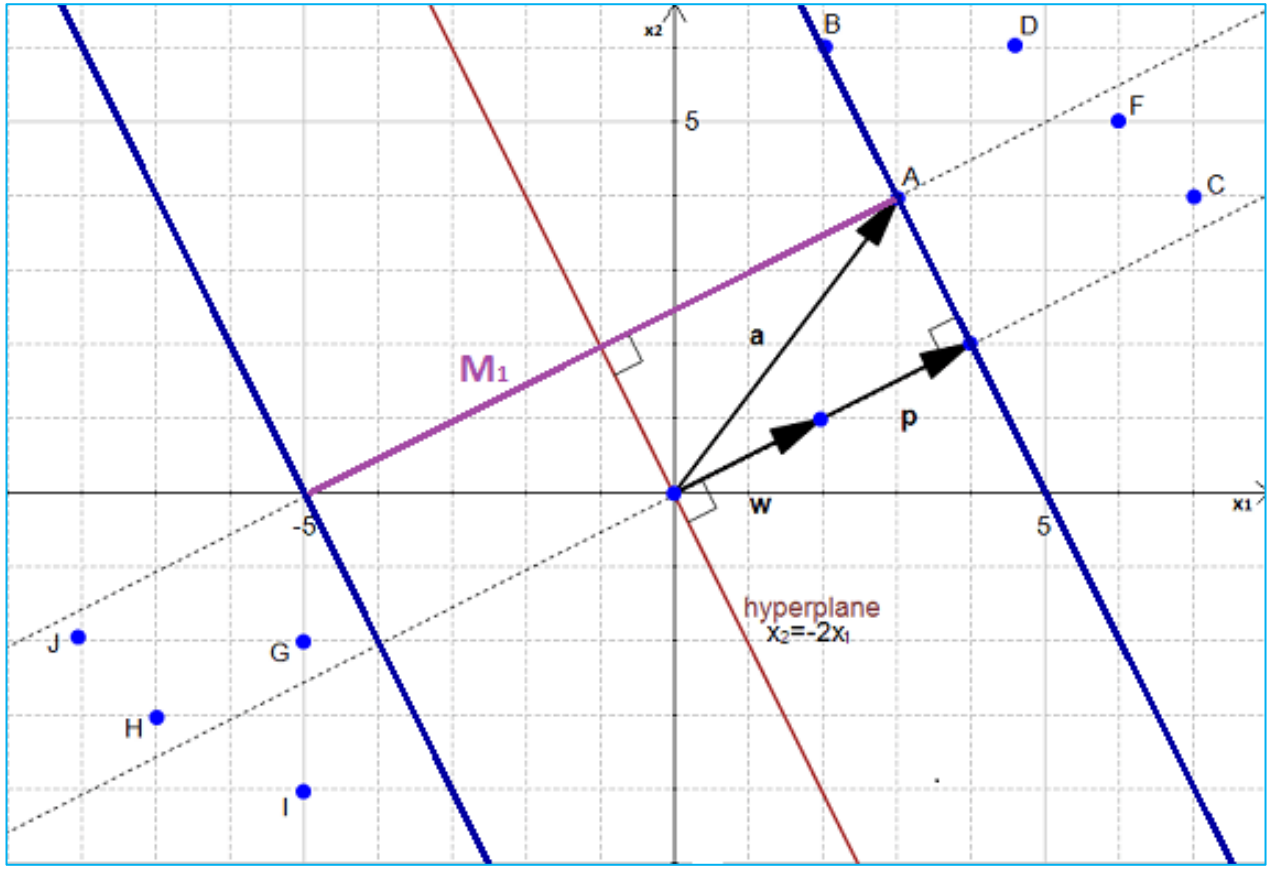
$$||p|| = \sqrt{4^2 + 2^2} = 2\sqrt{5}$$

Compute the margin of the hyperplane

Now we have the distance $||\boldsymbol{p}||$ *between* $\boldsymbol{A}$ *and the hyperplane,*

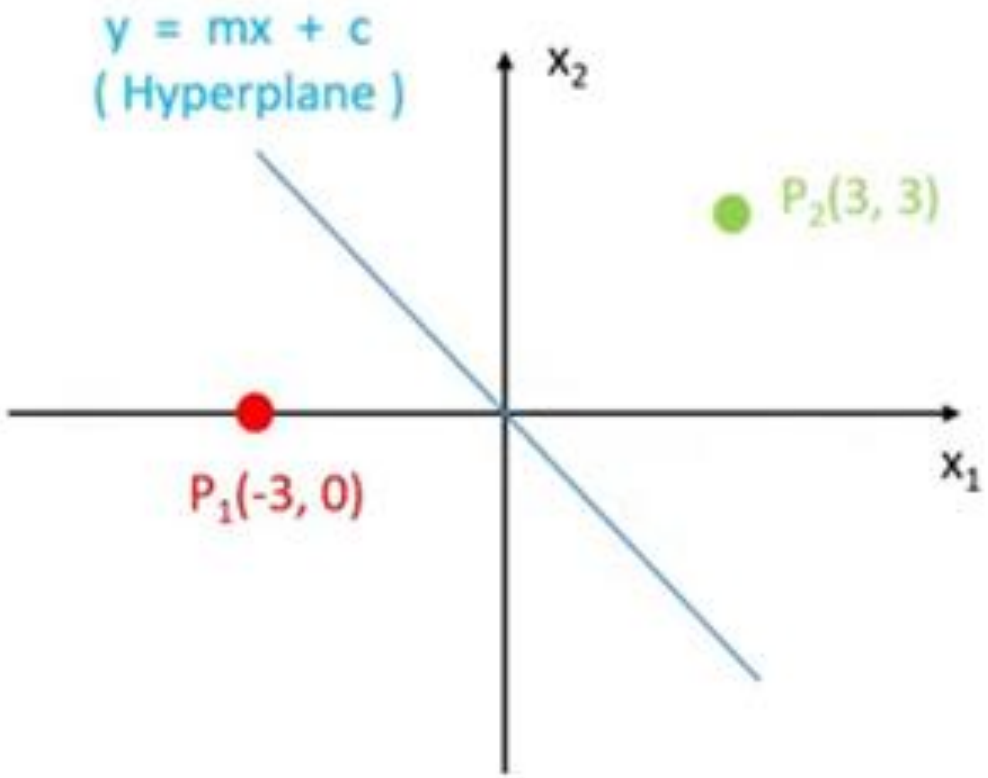
The margin is defined by

$$\boldsymbol{margin} = 2||\boldsymbol{p}|| = 2 * 2\sqrt{5} = 4\sqrt{5}$$



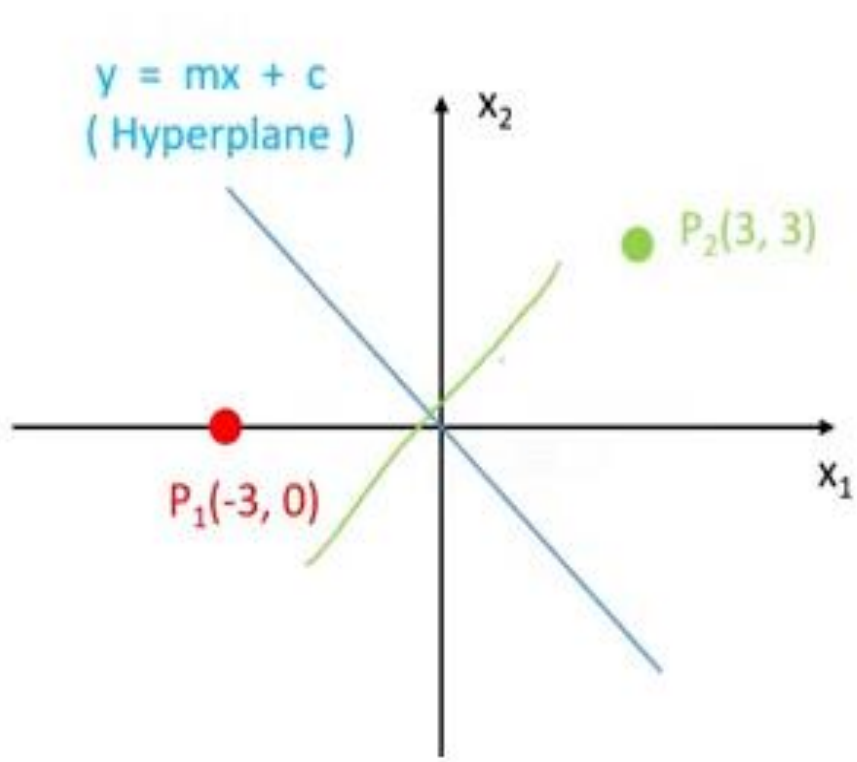
Classification of Data points

Classification of Data points: How to classify the points as +ve or –ve.



Let slope, m = - 1

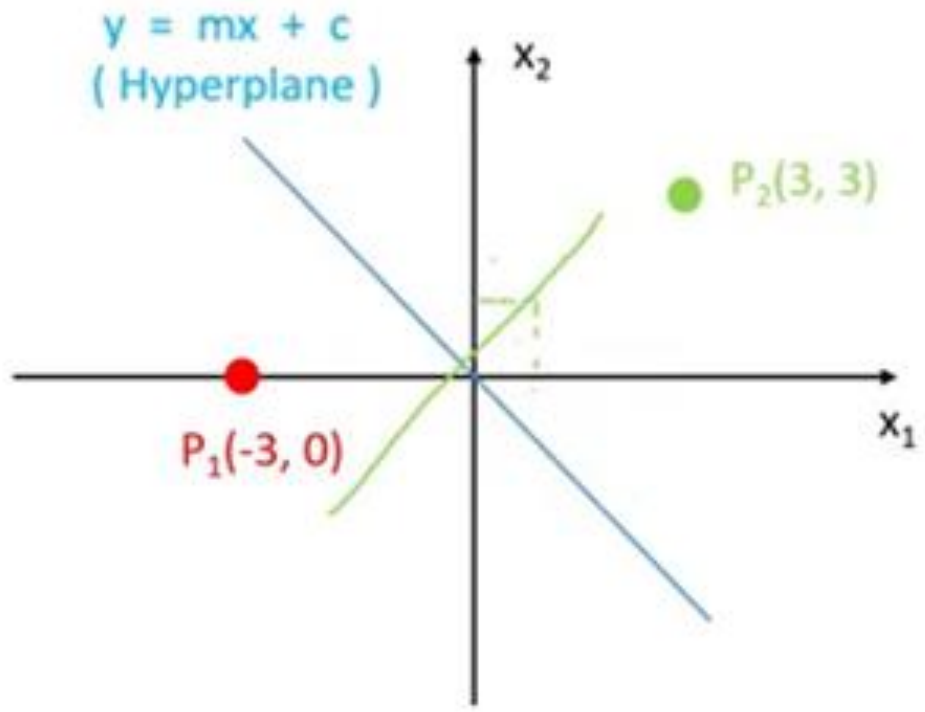
Intercept, c = 0



Let slope, m = - 1

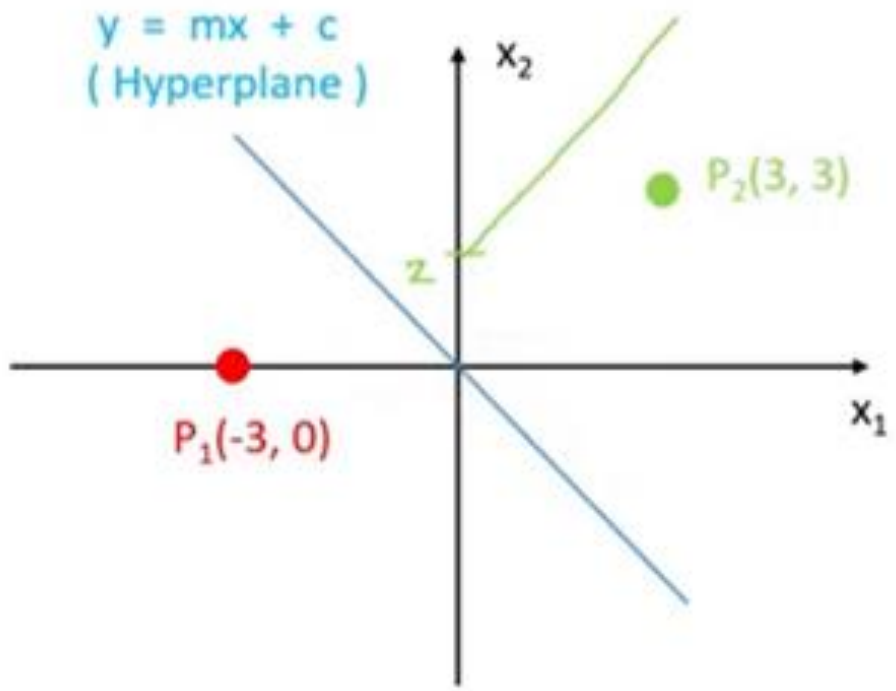
Intercept, c = 0

Classification of Data points: How to classify the points as +ve or –ve.



Let slope, m = - 1

Intercept, c = 0



Let slope, m = - 1

Intercept, c = 0

Classification of Data points: How to classify the points as +ve or –ve.

$y = mx + c$

$m = -1$ & $c = 0$

$y - mx - c = 0$

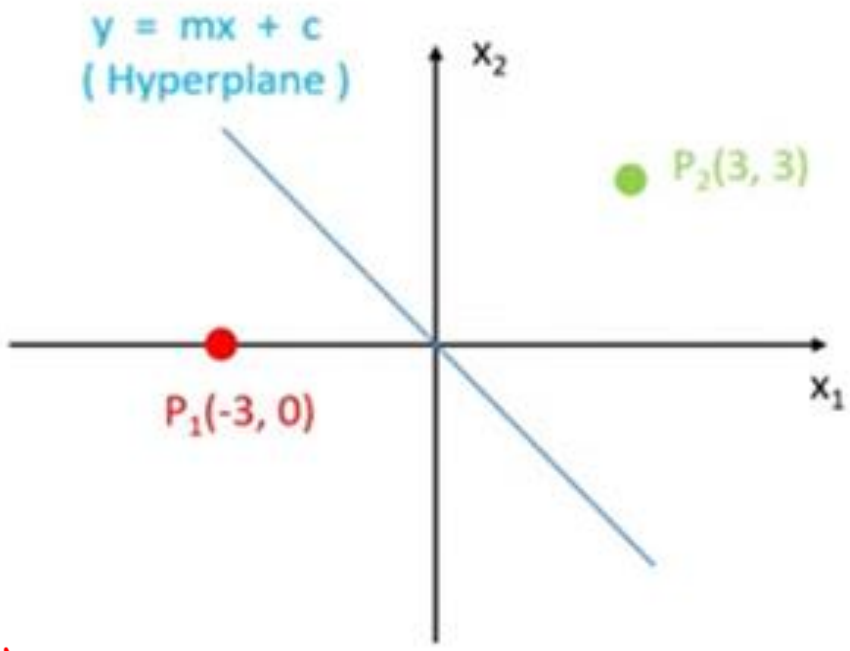
$y + x = 0$

$w = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$x = \begin{pmatrix} x \\ y \end{pmatrix}$

$w^T x = (1, 1) \begin{pmatrix} x \\ y \end{pmatrix}$ → data point value P_1, P_2

$= x + y$



Consider $P_1(-3, 0)$

$w^T x \Rightarrow w^T P_1$

$= (1, 1) \begin{pmatrix} -3 \\ 0 \end{pmatrix}$

$= -3$

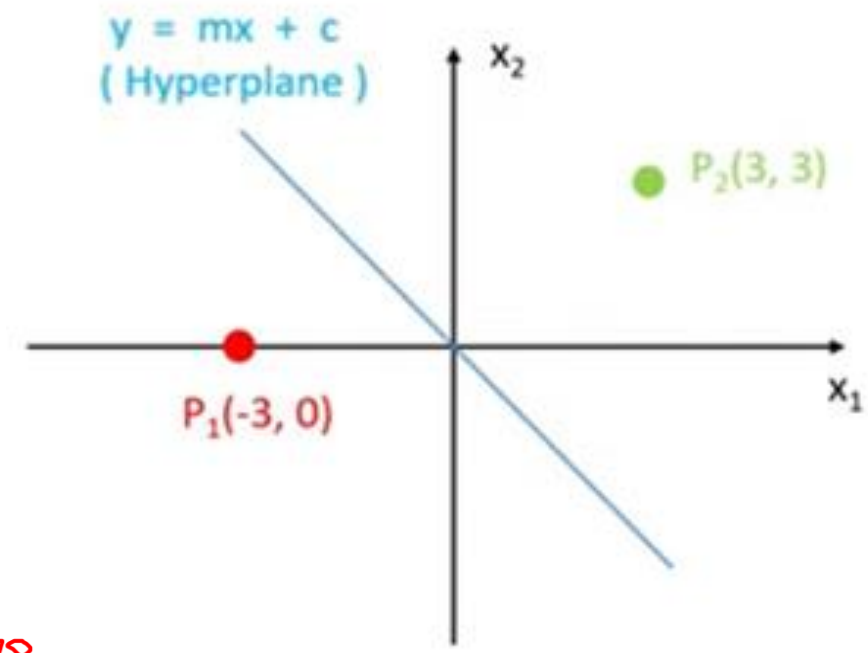
$w^T x$ for $P_1(-3, 0)$ is **−ve**

For $P_2[3, 3]$

$w^T x = (1, 1) \begin{pmatrix} 3 \\ 3 \end{pmatrix} = 6$

= +ve

Classification of Data points: How to classify the points as +ve or –ve.

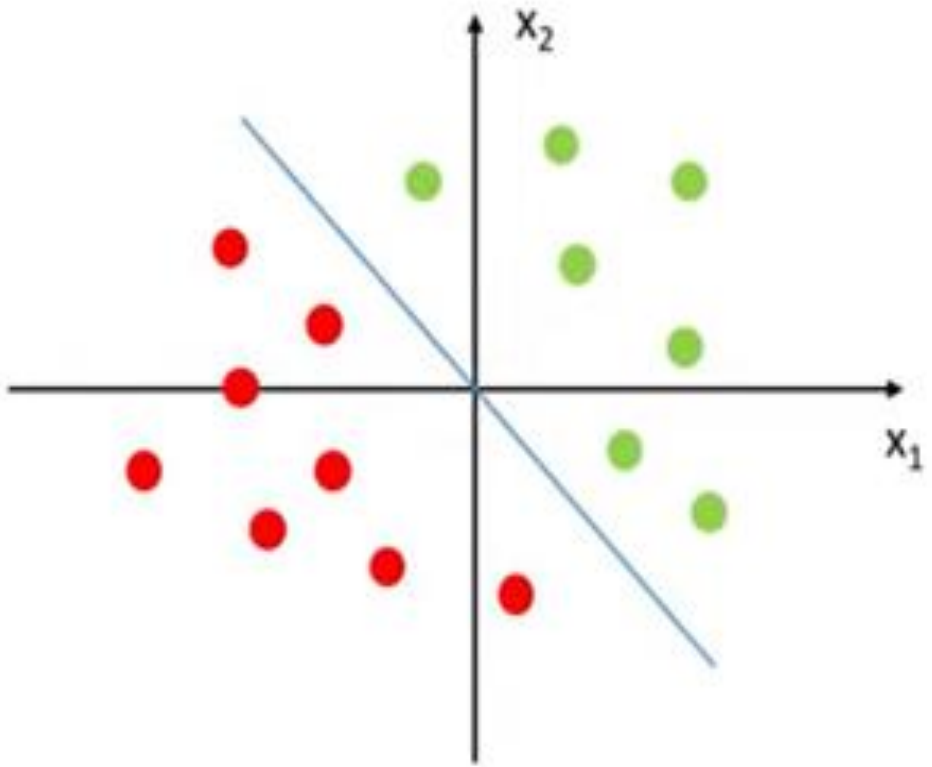


All points above the hyperplane are +ve

All points below the hyperplane are -ve

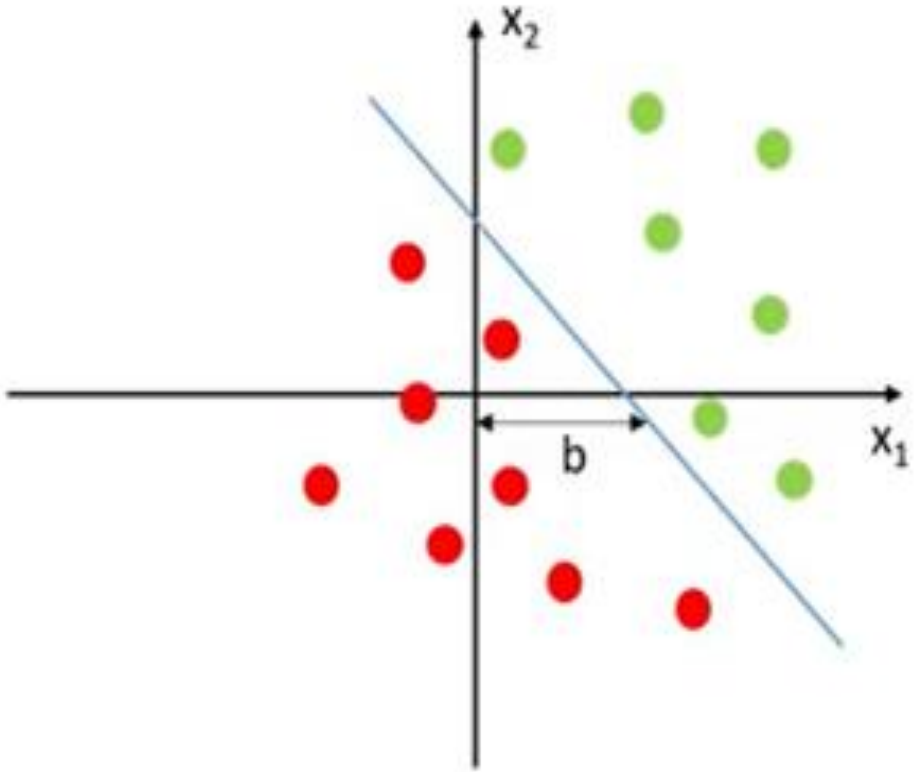
Let slope, m = - 1

Intercept, c = 0



$w^T x = \text{Label}$

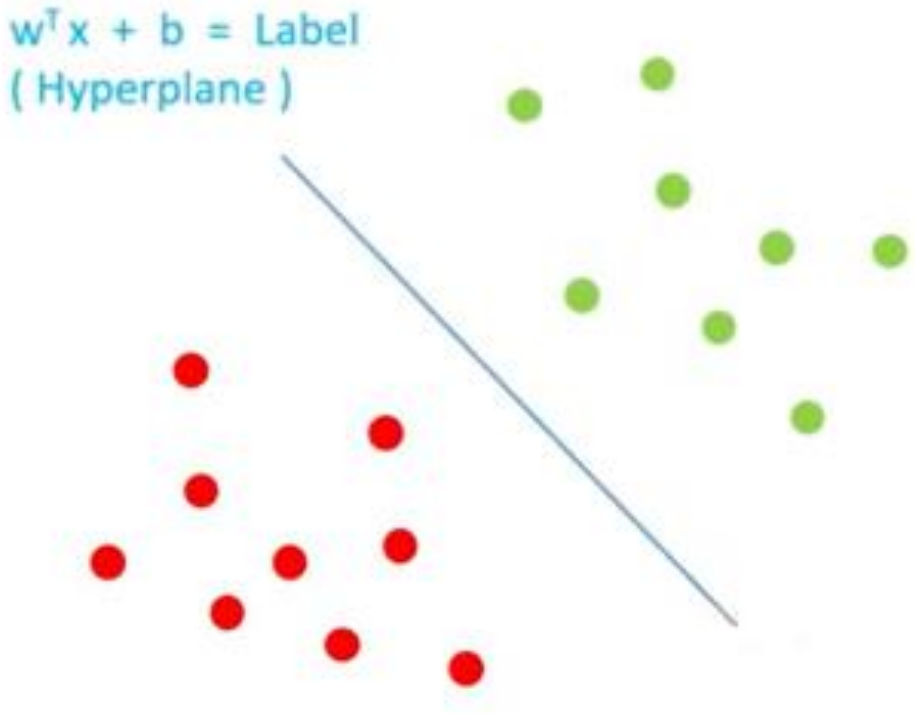
Label is the class of the data point
(red or green in this case)

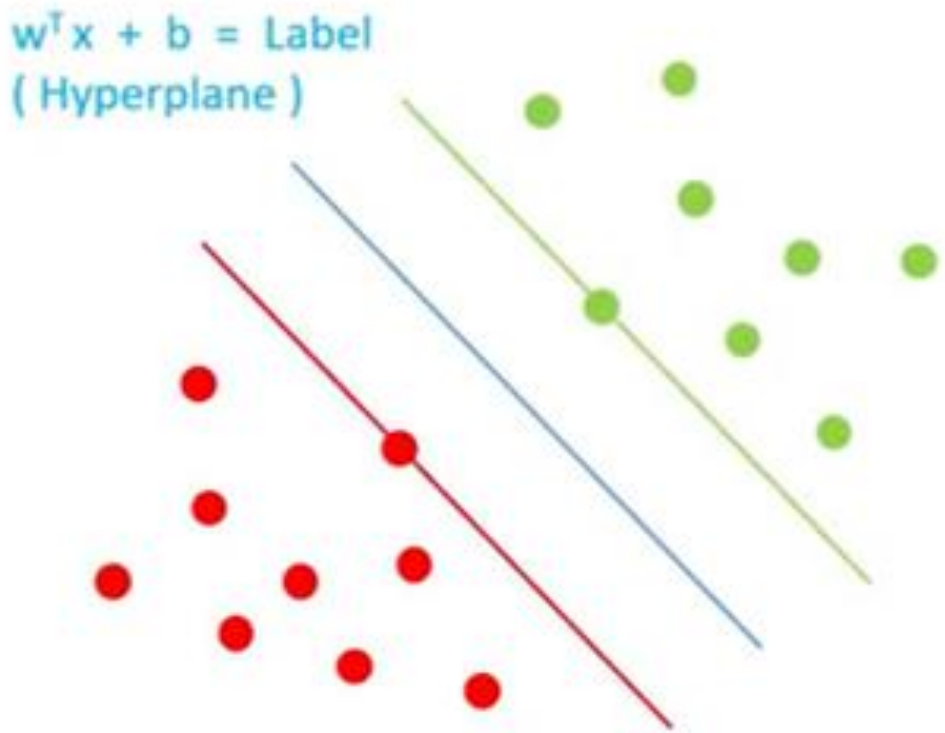


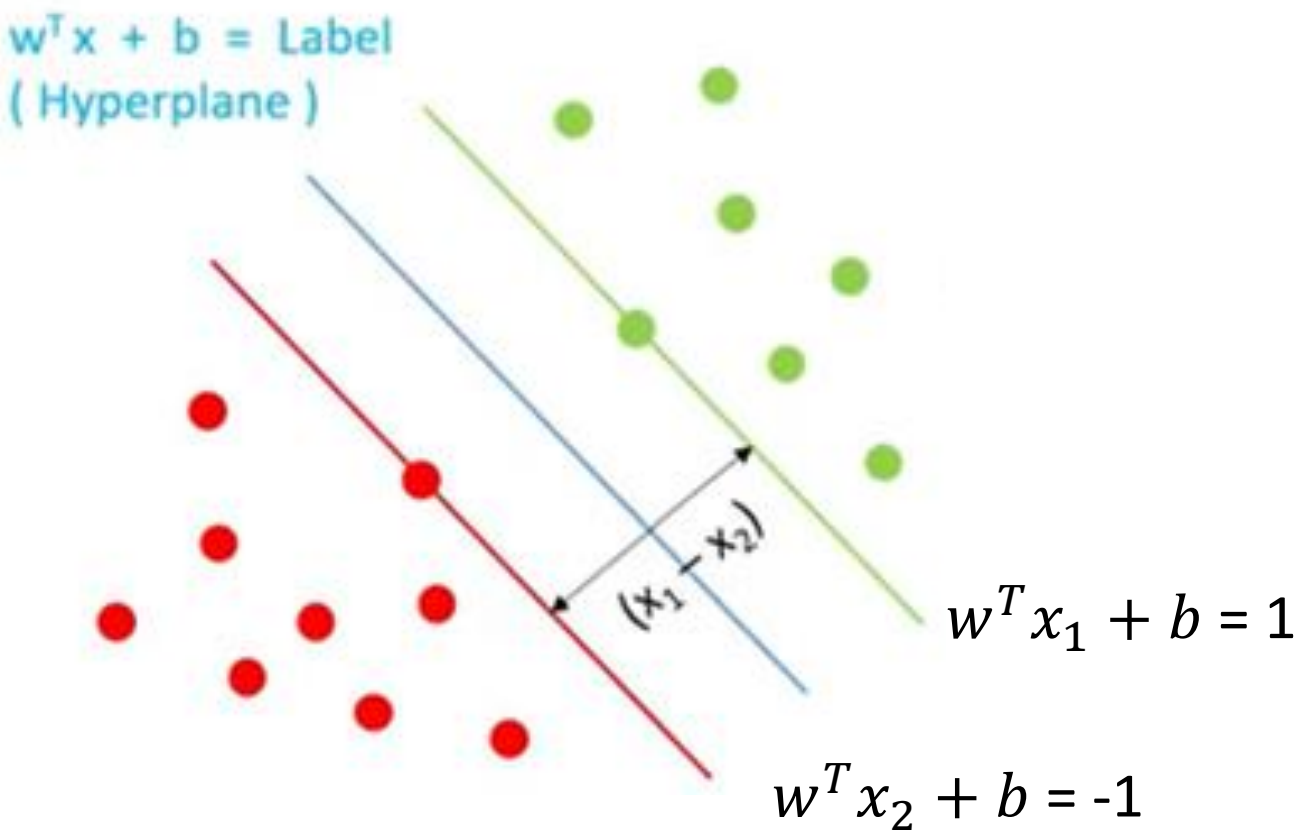
$w^T x + b = \text{Label}$

bias

Optimization for Maximum margin:



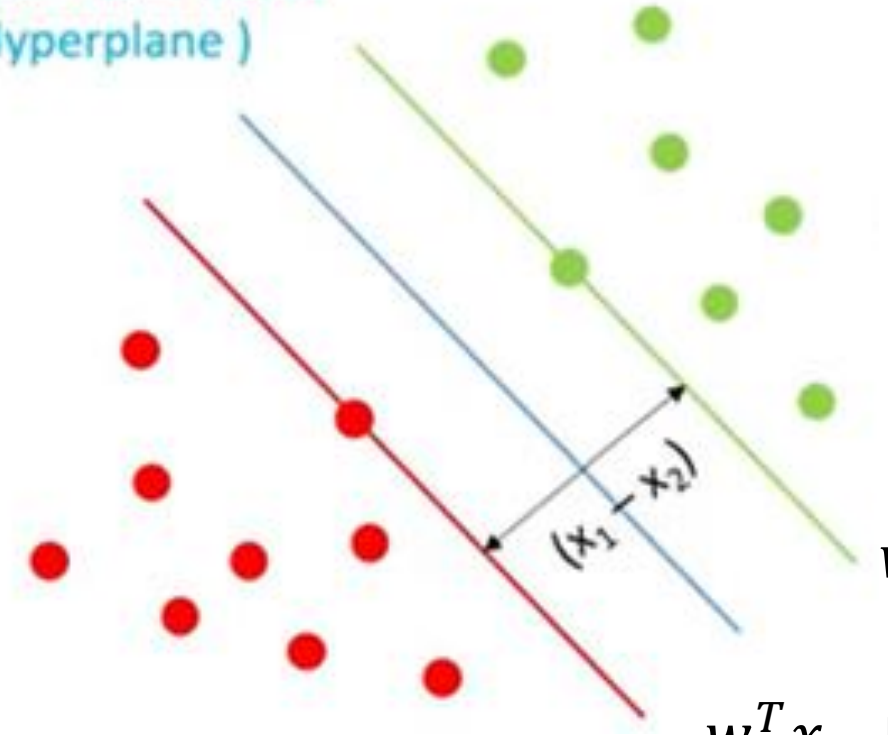




$w^T x_1 + b = 1$

$w^T x_2 + b = -1$

$w^T x + b$ = Label
(Hyperplane)



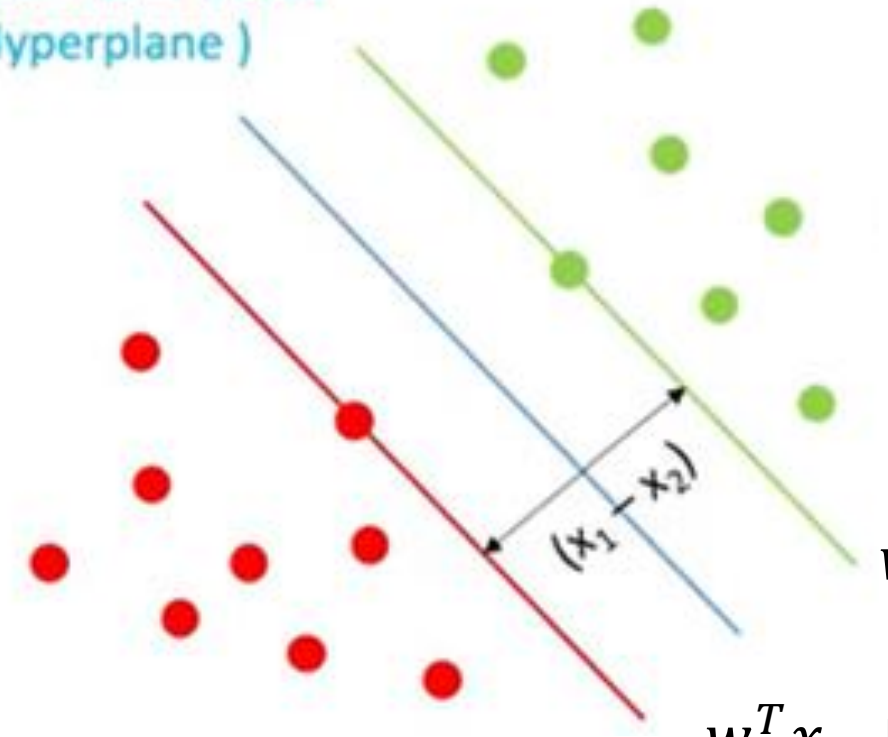
$$w^T x_1 + b = 1$$

$$w^T x_2 + b = -1$$

$$w^T x_1 + b = 1$$

$$(-)\ w^T x_2 + b = -1$$

$w^T x + b$ = Label
(Hyperplane)



$$w^T x_1 + b = 1$$

$$w^T x_2 + b = -1$$

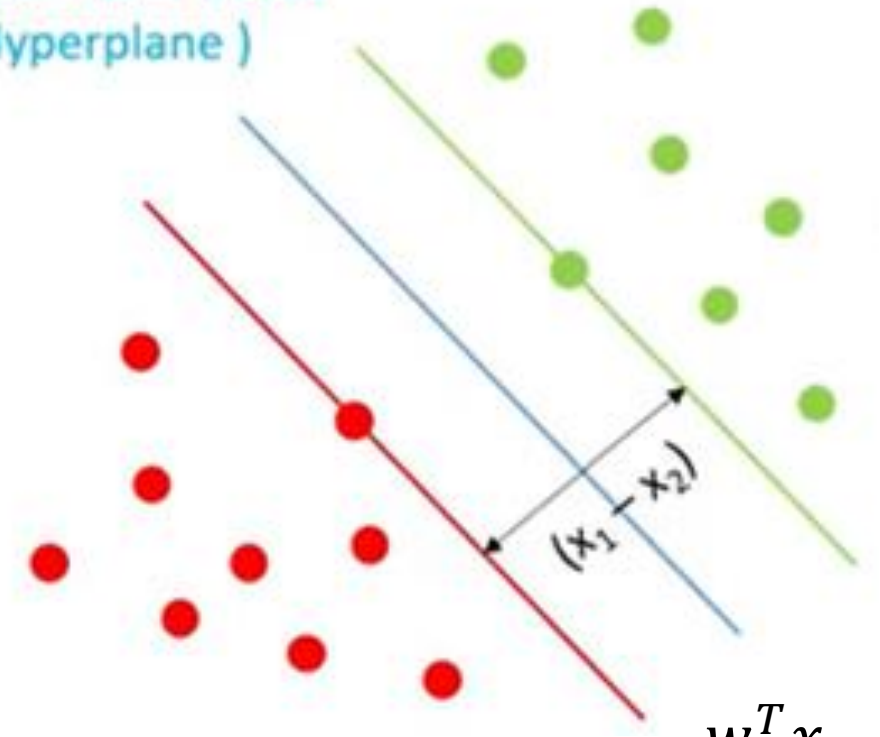
$$w^T x_1 + b = 1$$

$$(-)\ w^T x_2 + b = -1$$

$$w^T (x_1 - x_2) = 2$$

Optimization for Maximum margin:

$w^T x + b$ = Label
(Hyperplane)



$$w^T x_1 + b = 1$$

$$w^T x_2 + b = -1$$

$$w^T x_1 + b = 1$$

$$(-)\quad w^T x_2 + b = -1$$

$$w^T (x_1 - x_2) = 2$$

Divide by $||w||$

(magnitude of the vector)

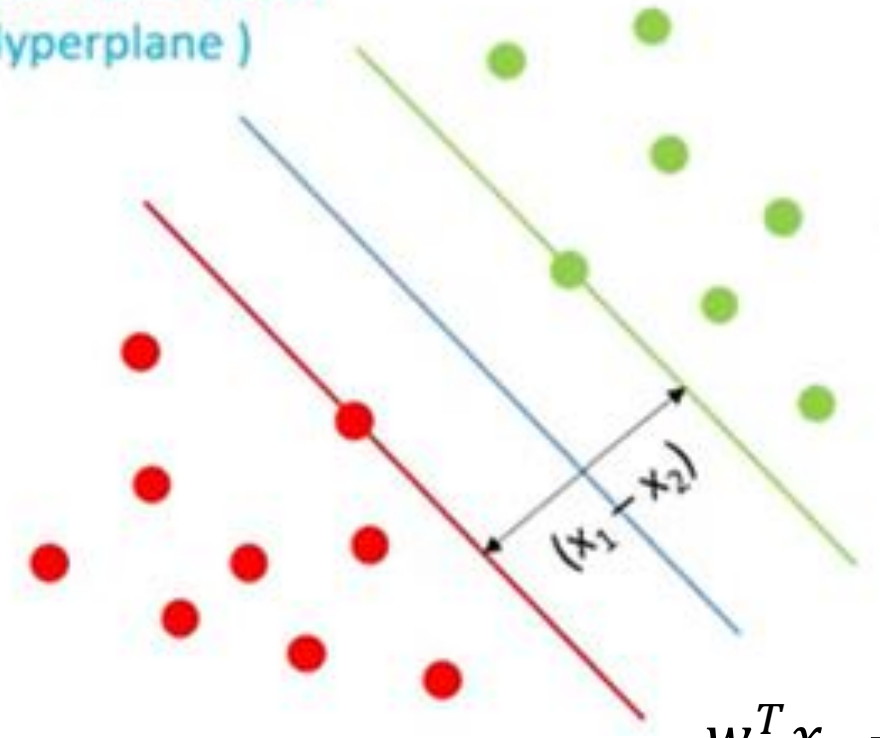
$$\frac{w^T}{||w||}(x_1 - x_2) = \frac{2}{||w||}$$

$$(x_1 - x_2) = \frac{2}{||w||} \quad \text{(margin)}$$

Optimization for Maximum margin:

$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases} \quad \text{(Label)}$$

$w^T x + b$ = Label
(Hyperplane)



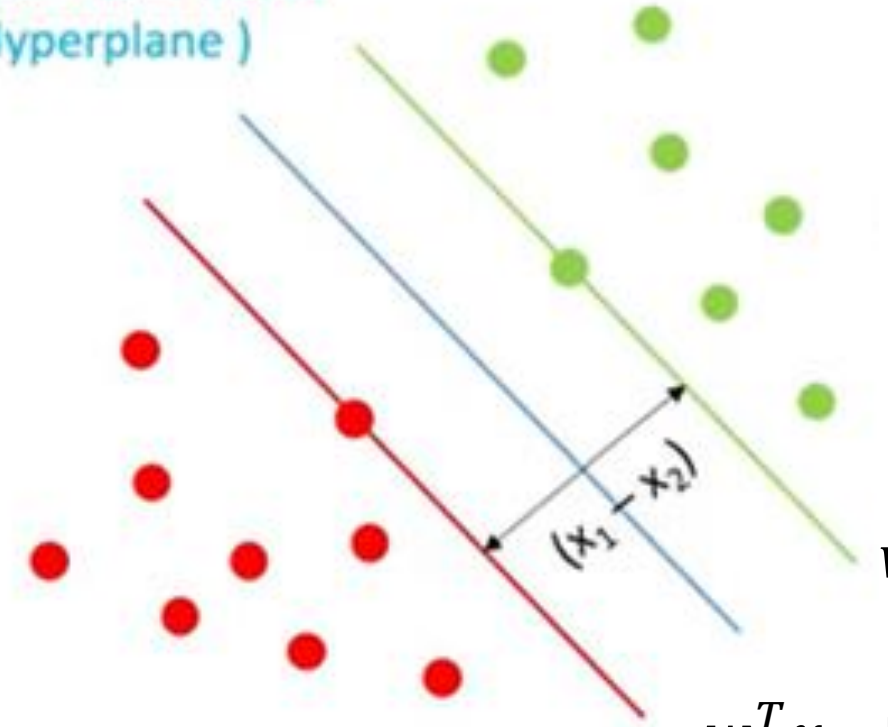
$w^T x_1 + b = 1$

$w^T x_2 + b = -1$

$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases} \quad \text{(Label)}$$

$$(x_1 - x_2) = \frac{2}{||\, w \,||} \quad \text{(margin)}$$

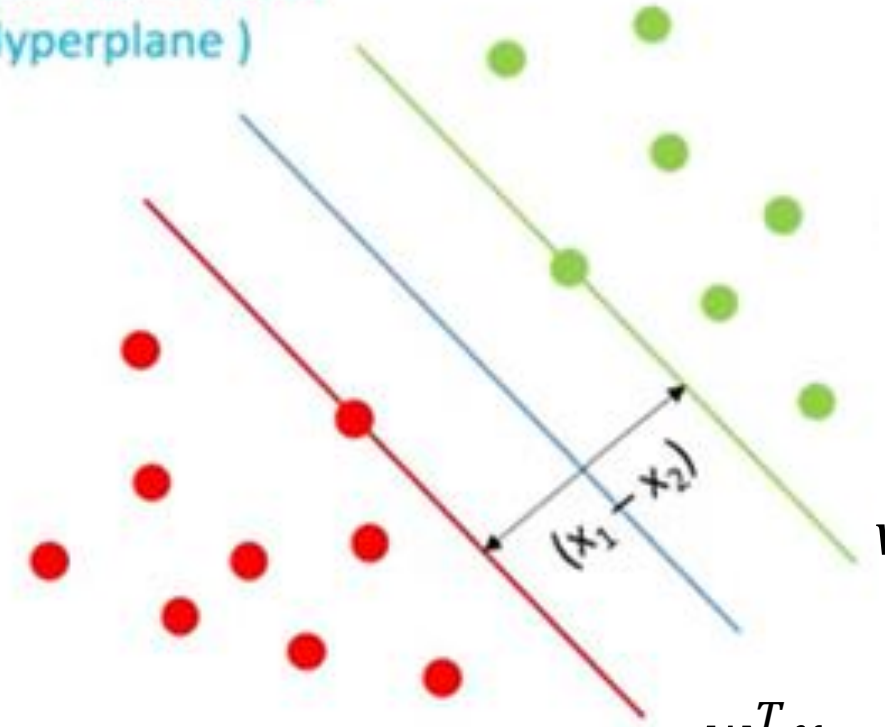
$w^T x + b$ = Label
(Hyperplane)



$w^T x_1 + b = 1$

$w^T x_2 + b = -1$

$w^T x + b$ = Label
(Hyperplane)



$$w^T x_1 + b = 1$$

$$w^T x_2 + b = -1$$

$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases} \quad \text{(Label)}$$

$$(x_1 - x_2) = \frac{2}{||w||} \quad \text{(margin)}$$

$$\max \left(\frac{2}{||w||} \right) \quad \text{Such that,}$$

$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

Soft Margin SVM

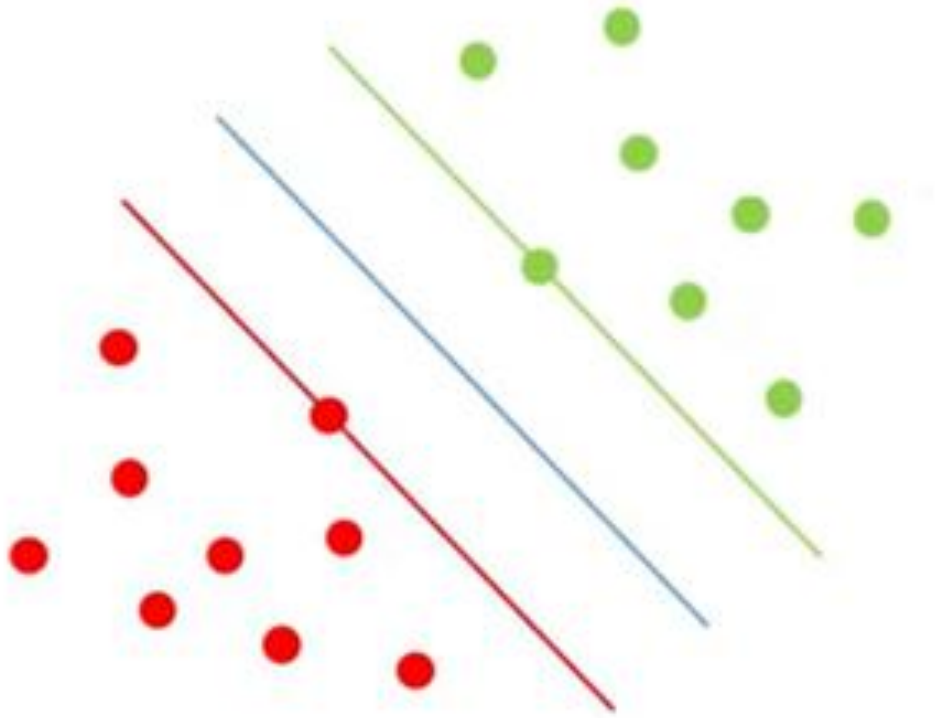
- In real-life applications, we rarely encounter datasets that are perfectly linearly separable.
- Instead, we often come across datasets that are either nearly linearly separable or entirely non-linearly separable.
- Unfortunately, the trick demonstrated above for linearly separable datasets is not applicable in these cases.
- This is soft margin classifier - Support Vector Machines (SVM) come into play.

Soft Margin SVM

- To handle non-linear data, we modify that equation in such a way that it allows few misclassifications that means it allows few points to be wrongly classified.

- We know that ***max[f(x)]*** can also 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.

Maximum margin without overfitting:



$$\max \left(\frac{2}{||w||} \right) \quad \text{Such that,}$$

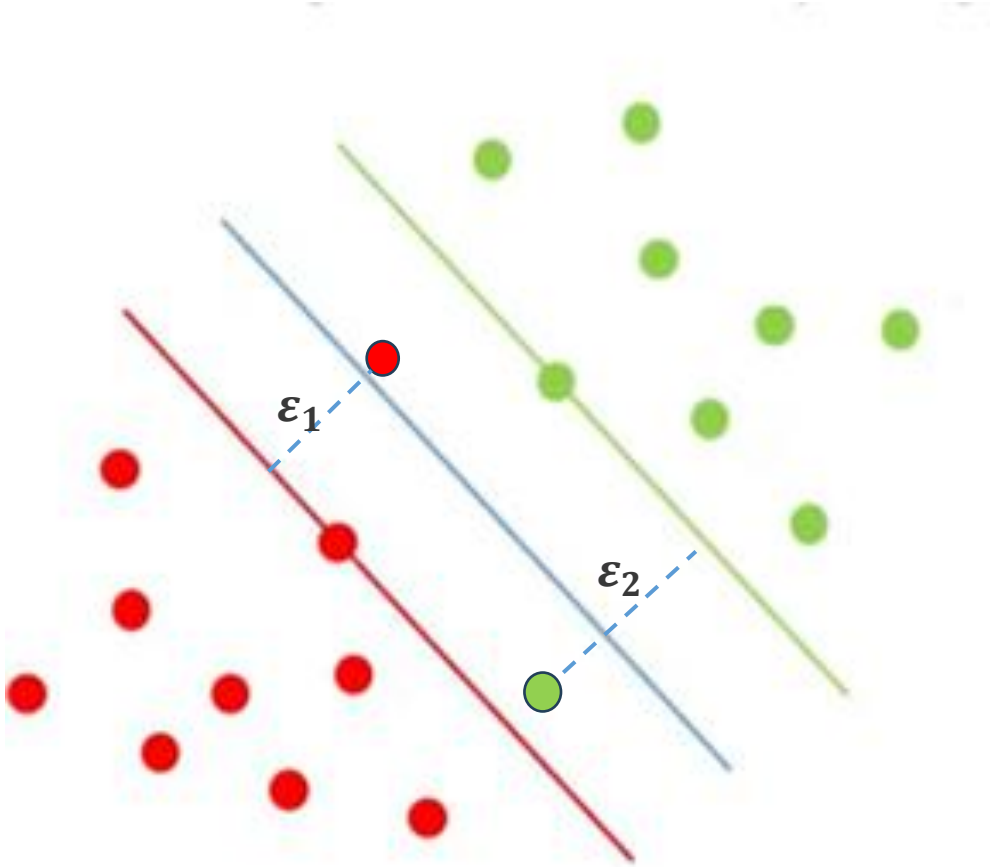
$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

This means

$$\min \left(\frac{||w||}{2} \right) + c * \Sigma \varepsilon_i$$

To make a soft margin equation we add 2 more terms to this equation which is ε and multiply that by a **hyperparameter '*c*'**

- For all the ***correctly classified*** points ε will be equal to 0 and for all the ***incorrectly classified*** points the ε is simply the distance of that particular point from its correct hyperplane.

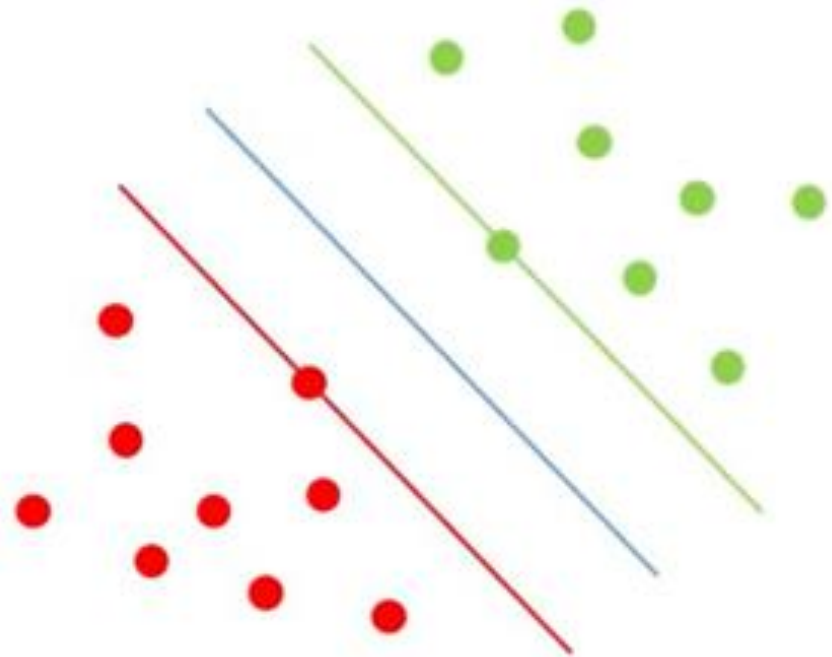


$$\max \left(\frac{2}{||w||} \right) \quad \text{Such that,}$$

$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

$$\min \left(\frac{||w||}{2} \right) + c * \sum \varepsilon_i$$

Maximum margin without overfitting:



$$\max \left(\frac{2}{||w||} \right) \quad \text{Such that,}$$

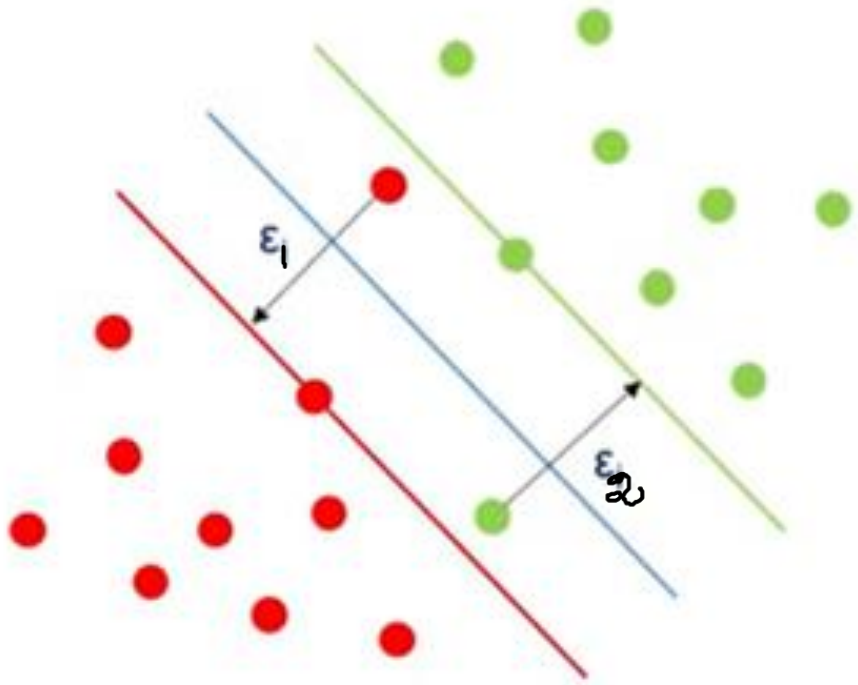
$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

$$\min \left(\frac{||w||}{2} \right) + c * \Sigma \varepsilon_i$$

c --> Number of errors

ε_i --> Error magnitude

Maximum margin without overfitting:



$$\max \left(\frac{2}{||w||} \right) \quad \text{Such that,}$$

$$y_i = \begin{cases} -1, & w^T x_1 + b \leq -1 \\ 1, & w^T x_1 + b \geq 1 \end{cases}$$

$$\min \left(\frac{||w||}{2} \right) + c * \Sigma \varepsilon_i$$

c --> Number of errors = 2

ε_i --> Error magnitude

Example

Linear Example

Suppose we are given the following positively labeled data points in $\Re^2$:

$$\left\{\begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ -1 \end{pmatrix}, \begin{pmatrix} 6 \\ 1 \end{pmatrix}, \begin{pmatrix} 6 \\ -1 \end{pmatrix}\right\}$$

and the following negatively labeled data points in $\Re^2$ (see Figure 1):

$$\left\{\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}\right\}$$

Linear Example

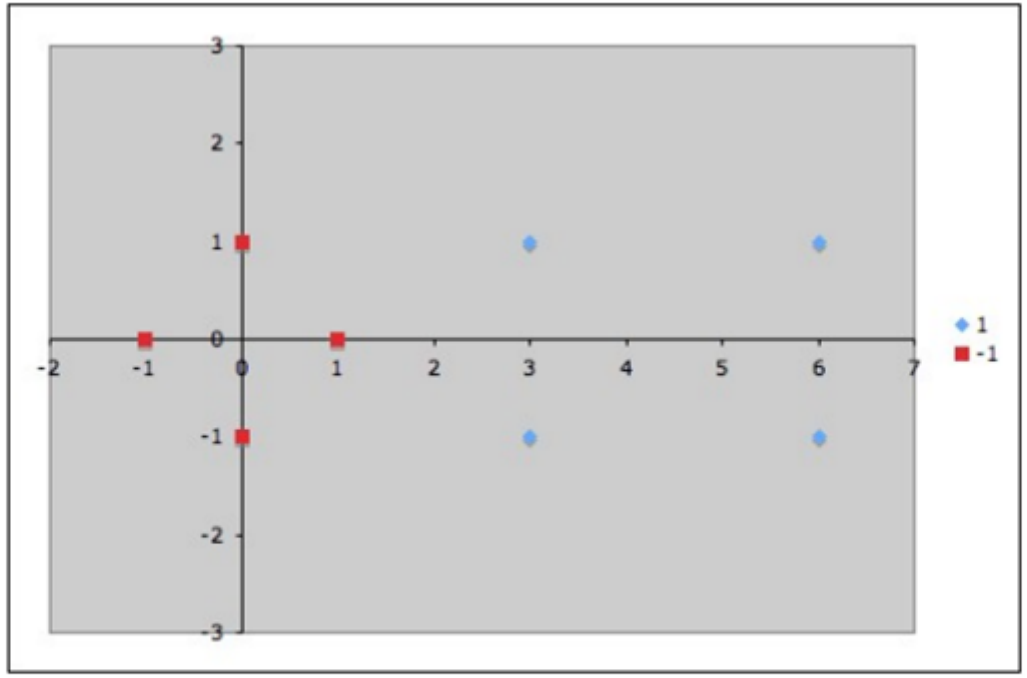


Figure 1: Sample data points in $\Re^2$. Blue diamonds are positive examples and red squares are negative examples.

Linear Example

- Discover a simple SVM that accurately discriminates the two classes.
- Since the data is linearly separable, we can use a linear SVM.
- There are three support vectors (see Figure 2)

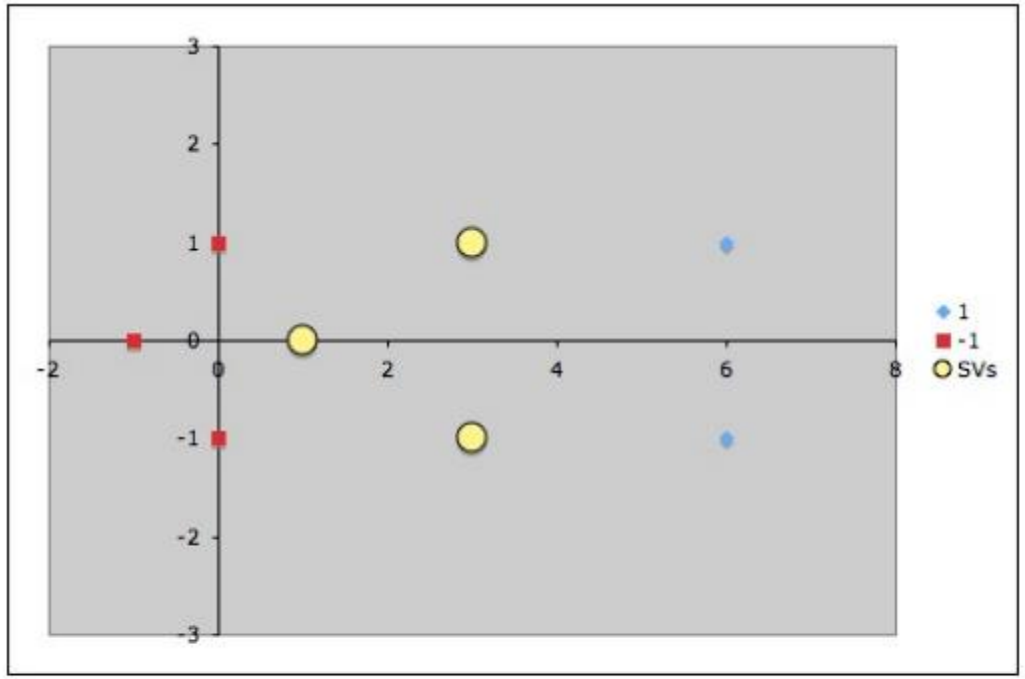


Figure 2: The three support vectors are marked as yellow circles.

Implementation in Python

Implementation of SVM in Python & R Steps

- Importing the Dataset
- Splitting the Dataset into the Training set and Test set
- Feature Scaling
- Fitting Classifier to Training Set
- Predicting the Test Set result

Python Code

```
# Data Preprocessing

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

	User ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	0
2	15810944	Male	35	20000	0
3	15668575	Female	26	43000	0
4	15603246	Female	27	57000	0
5	15804002	Male	19	76000	0
6	15728773	Male	27	58000	0
7	15598044	Female	27	84000	0
8	15694829	Female	32	150000	1
9	15600575	Male	25	33000	0
10	15727311	Female	35	65000	0
11	15570769	Female	26	80000	0
12	15606274	Female	26	52000	0

Showing 1 to 12 of 400 entries

	Age	EstimatedSalary	Purchased
1	19	19000	0
2	35	20000	0
3	26	43000	0
4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
9	25	33000	0
10	35	65000	0
11	26	80000	0
12	26	52000	0

Showing 1 to 12 of 400 entries

Python Code

```
# Data Preprocessing

# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state = 0)
```

	Age	EstimatedSalary	Purchased
1	19	19000	0
3	26	43000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
10	35	65000	0
11	26	80000	0
13	20	86000	0
14	32	18000	0
15	18	82000	0
16	29	80000	0
17	47	25000	1

Showing 1 to 12 of 300 entries

	Age	EstimatedSalary	Purchased
2	35	20000	0
4	27	57000	0
5	19	76000	0
9	25	33000	0
12	26	52000	0
18	45	26000	1
19	46	28000	1
20	48	29000	1
22	47	49000	1
29	29	43000	0
32	27	137000	1
34	28	44000	0

Showing 1 to 12 of 100 entries

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

	Age	EstimatedSalary	Purchased
1	-1.76554750	-1.47334137	0
3	-1.09629664	-0.78837605	0
6	-1.00068938	-0.36027273	0
7	-1.00068938	0.38177303	0
8	-0.52265305	2.26542765	1
10	-0.23583125	-0.16049118	0
11	-1.09629664	0.26761214	0
13	-1.66994024	0.43885347	0
14	-0.52265305	-1.50188159	0
15	-1.86115477	0.32469259	0
16	-0.80947485	0.26761214	0
17	0.91145593	-1.30210004	1

Showing 1 to 12 of 300 entries

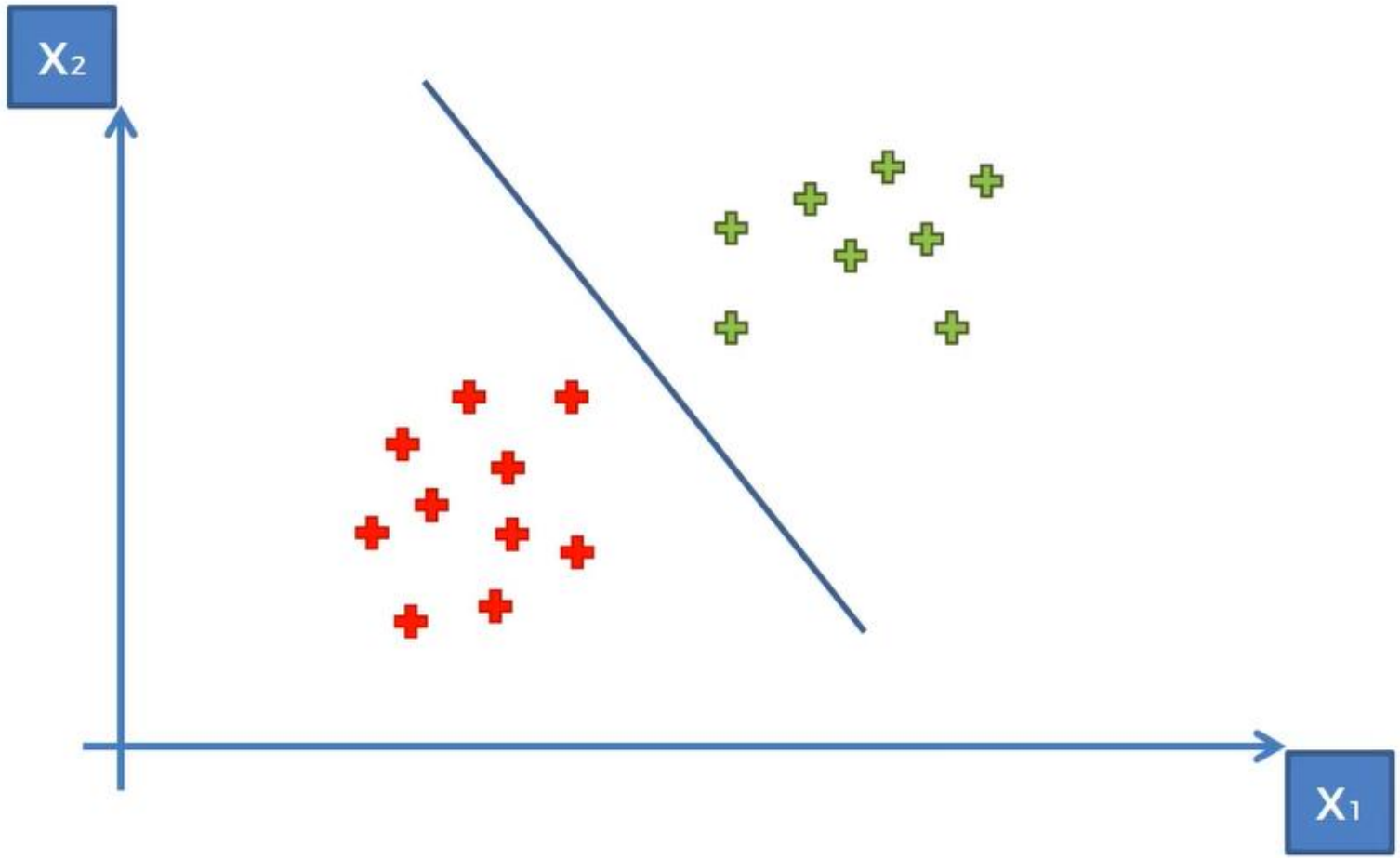
	Age	EstimatedSalary	Purchased
2	-0.30419063	-1.51354339	0
4	-1.05994374	-0.32456026	0
5	-1.81569686	0.28599864	0
9	-1.24888202	-1.09579256	0
12	-1.15441288	-0.48523366	0
18	0.64050076	-1.32073531	1
19	0.73496990	-1.25646596	1
20	0.92390818	-1.22433128	1
22	0.82943904	-0.58163769	1
29	-0.87100546	-0.77444577	0
32	-1.05994374	2.24621408	1
34	-0.96547460	-0.74231109	0

Showing 1 to 12 of 100 entries

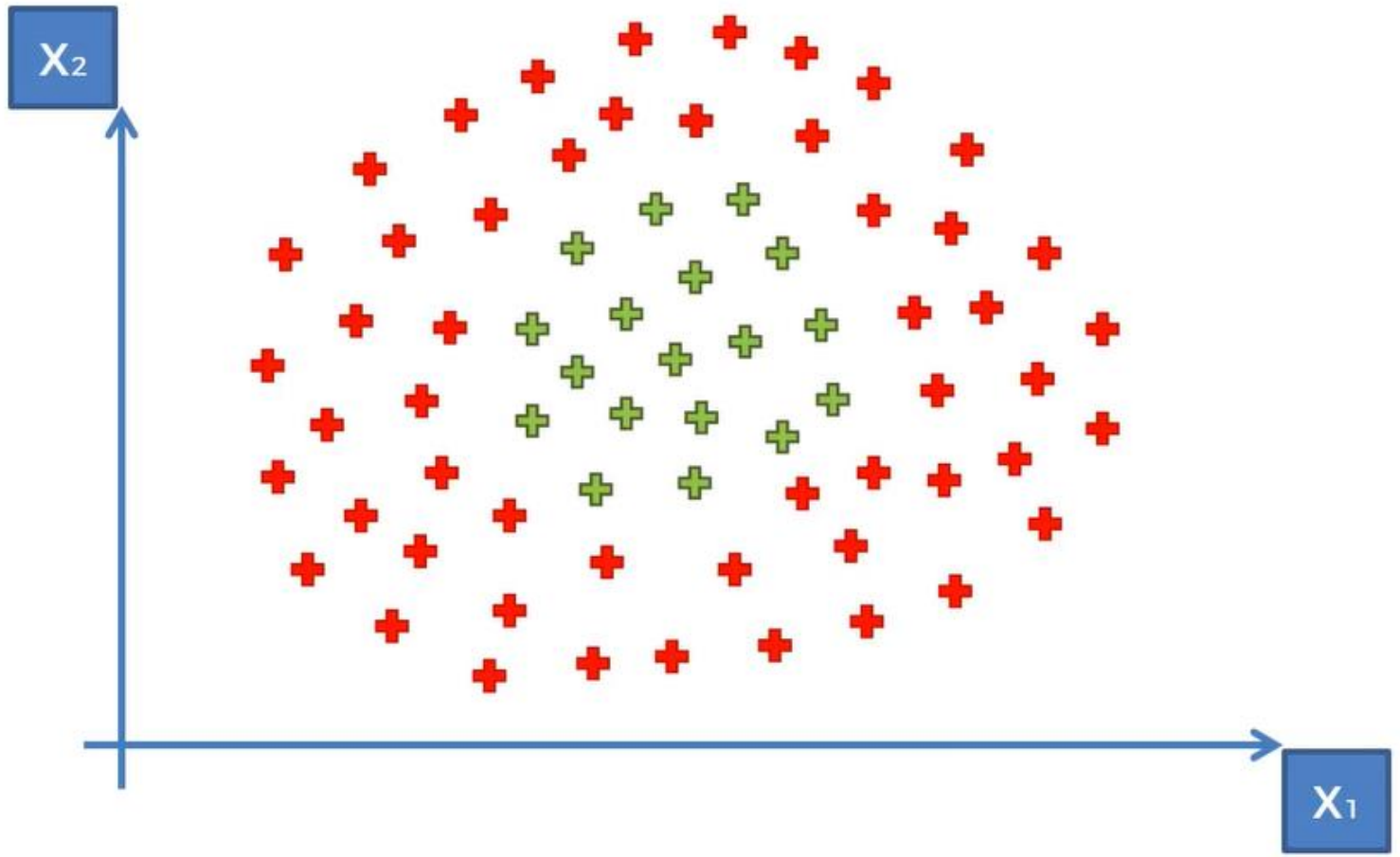
```
# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train,  y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Kernel SVM Intuition

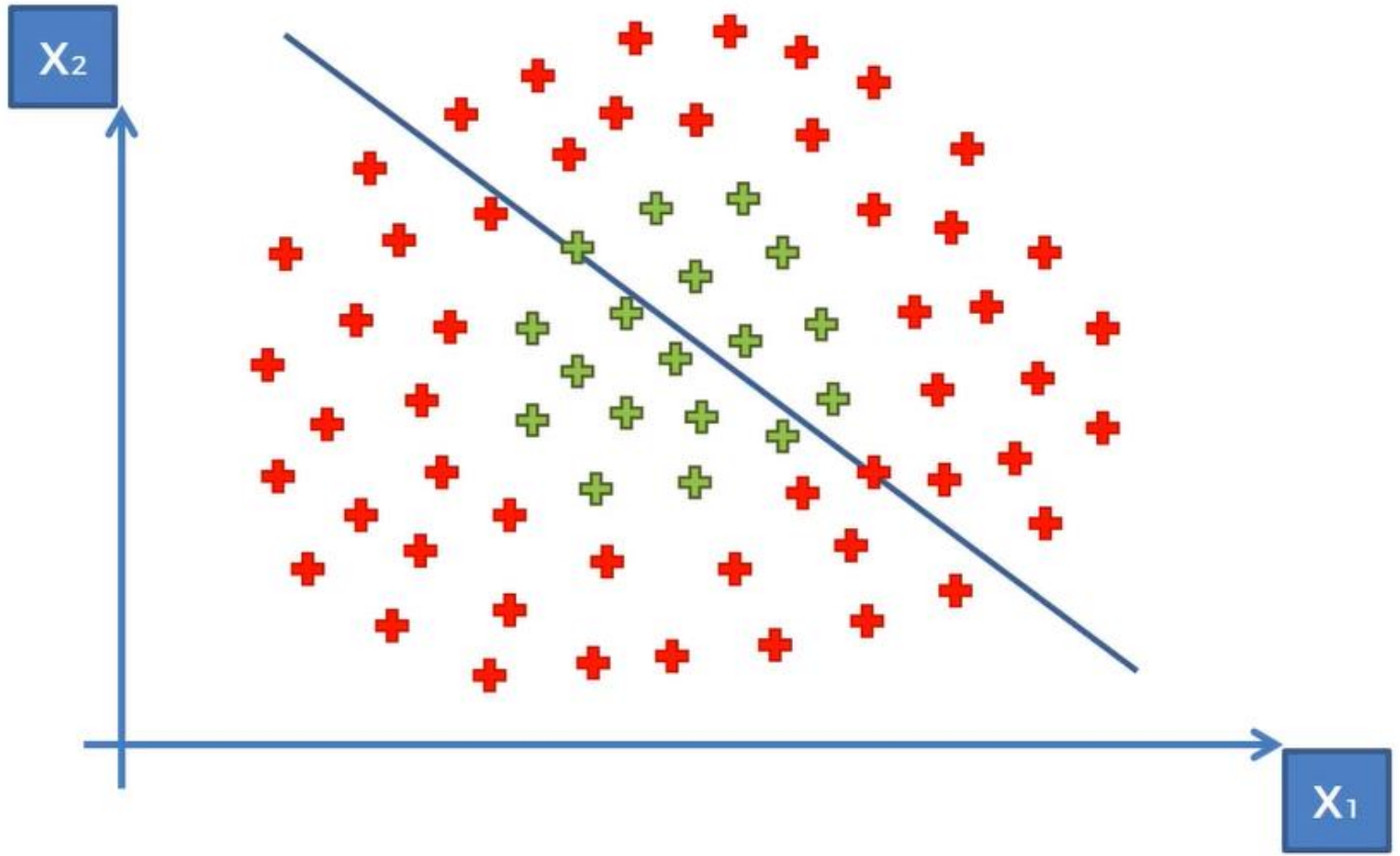
SVM separates well these points



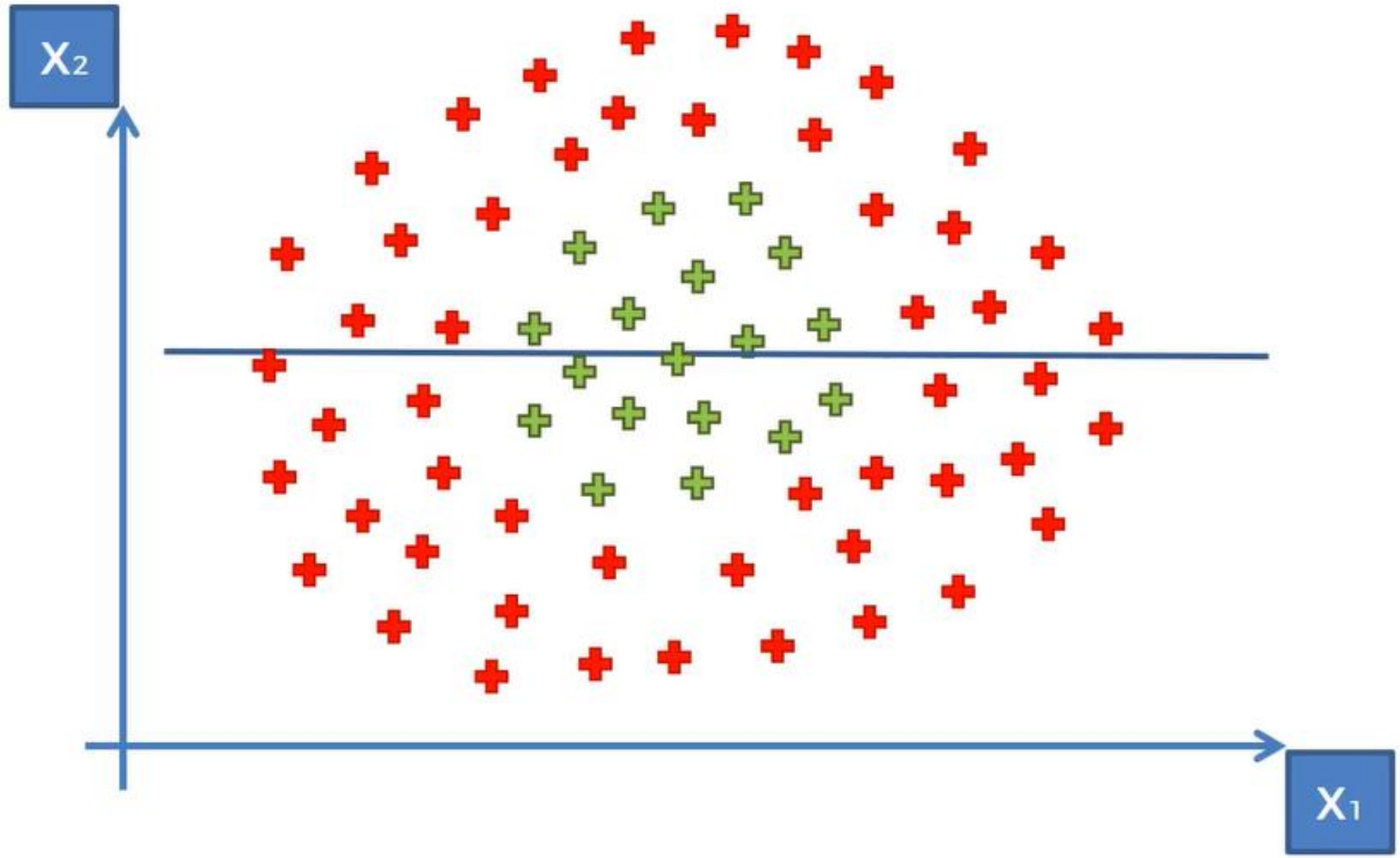
What about these points ?



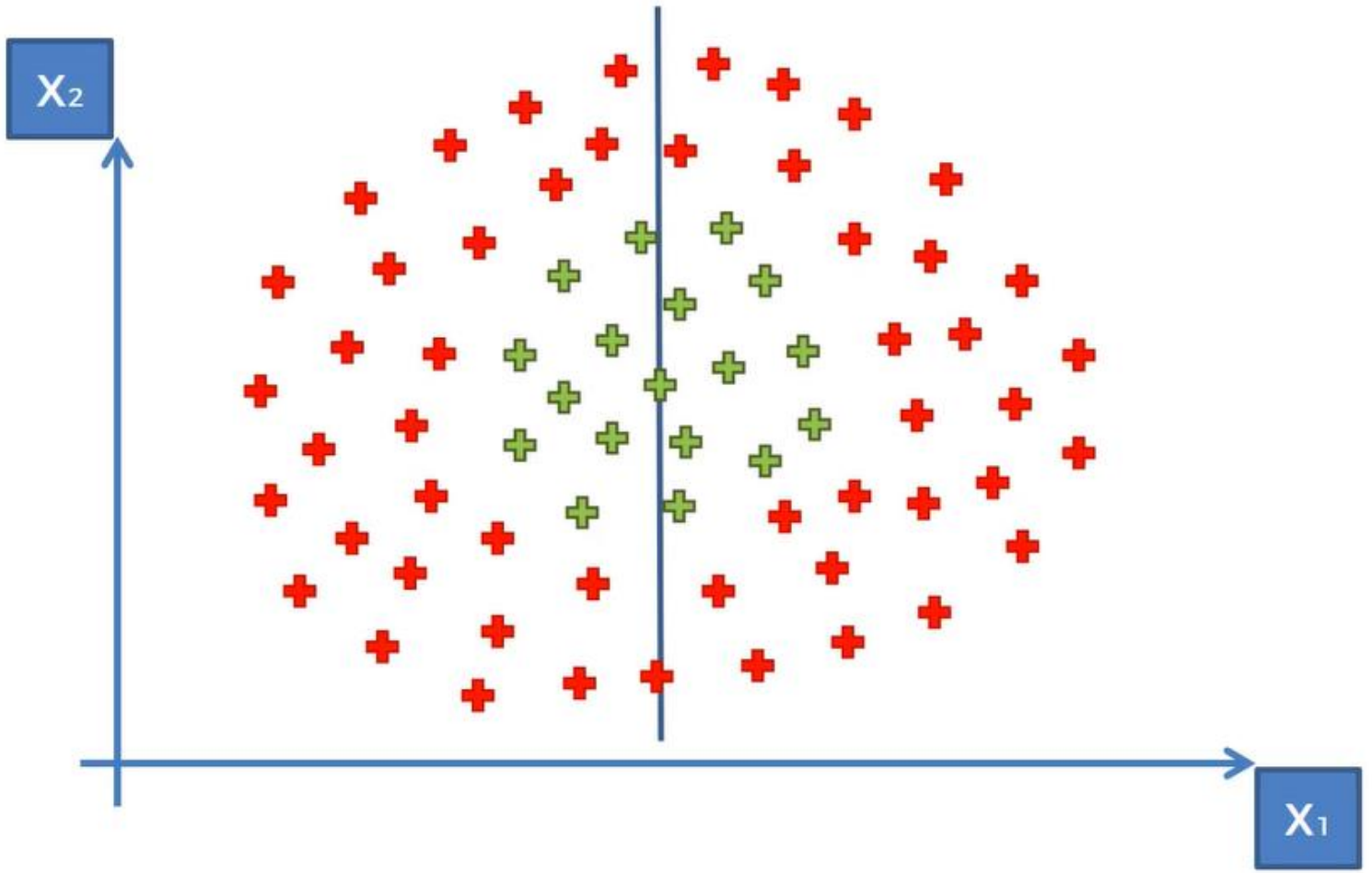
What about these points ?



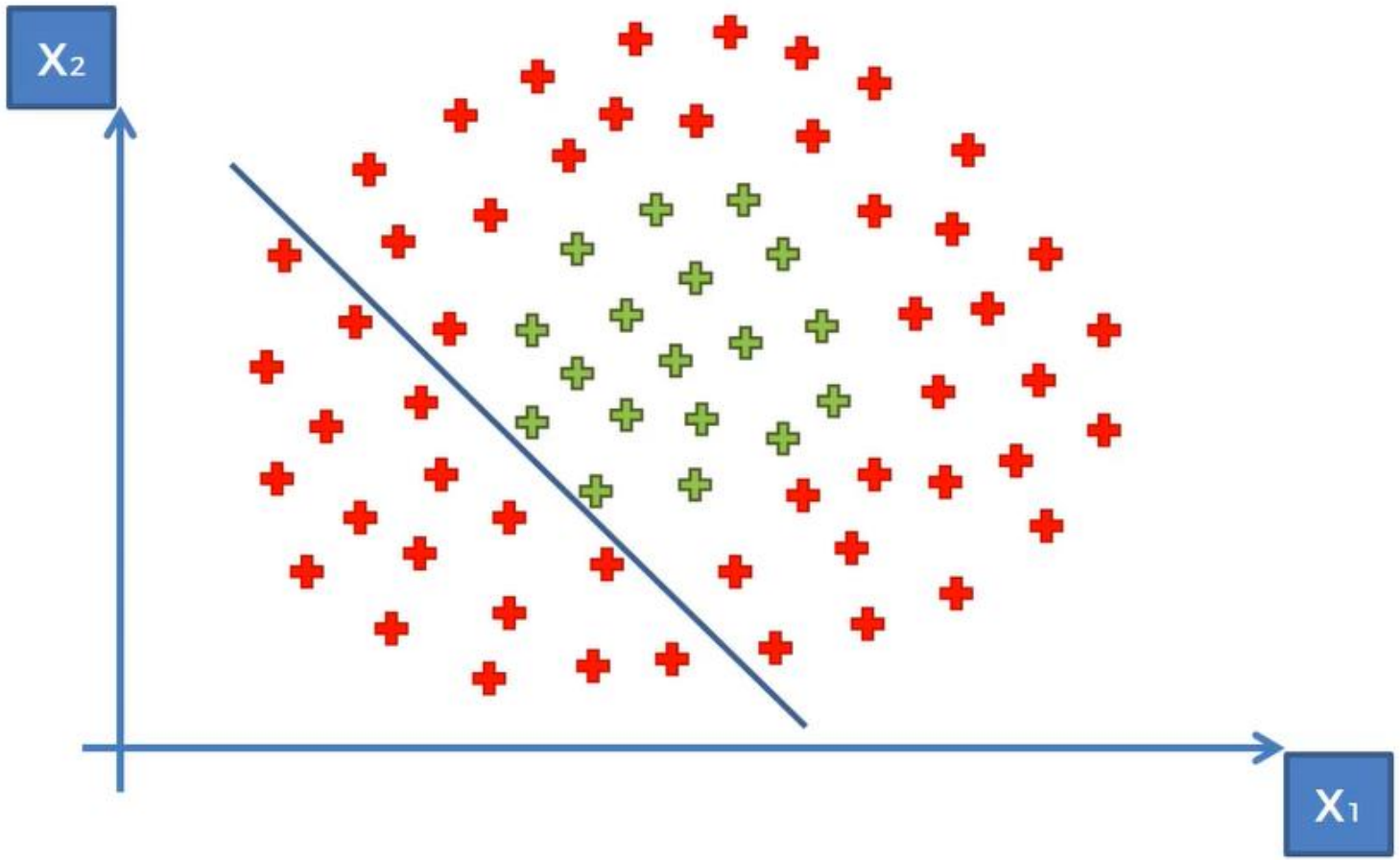
What about these points ?



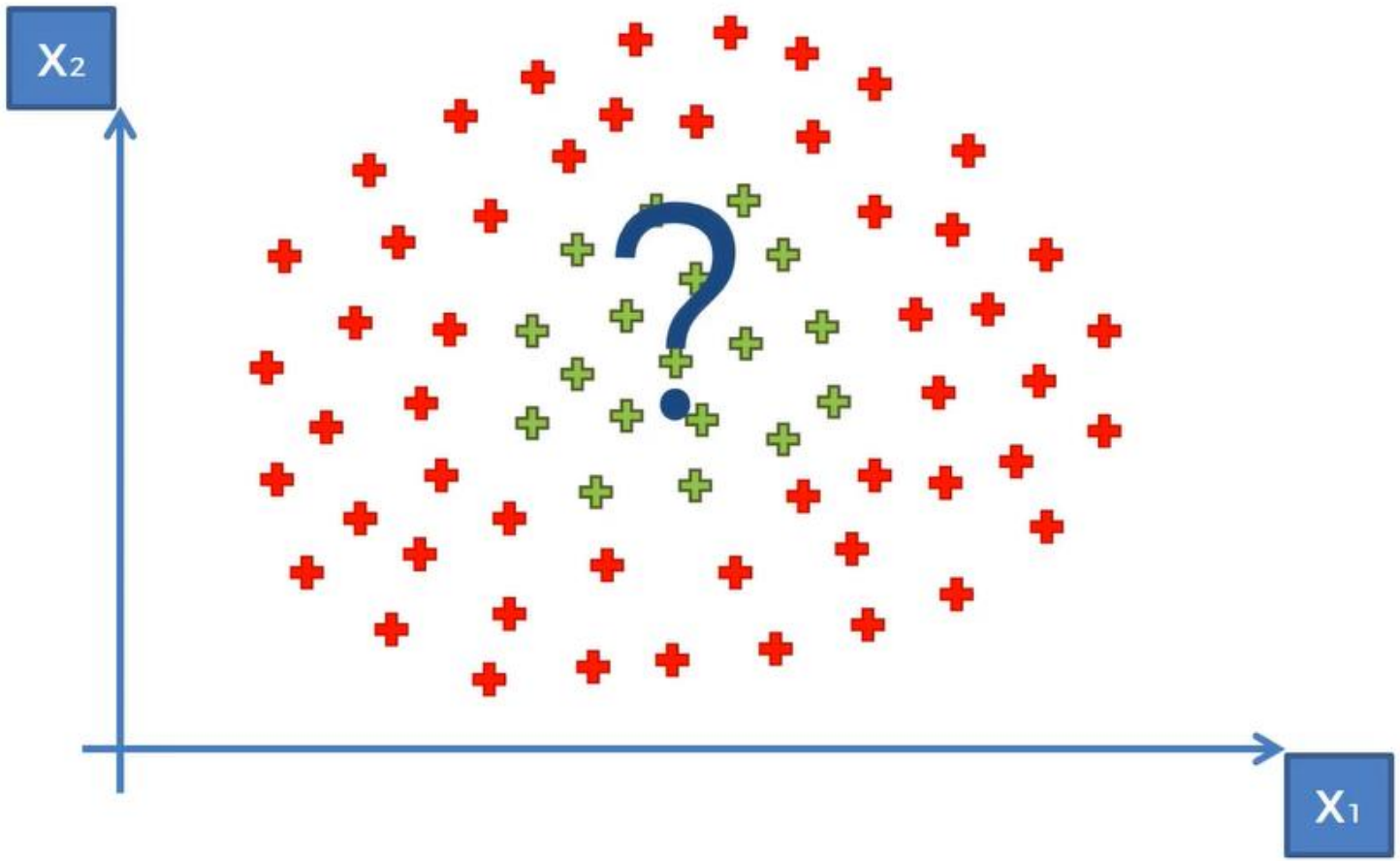
What about these points ?



What about these points ?



What about these points ?

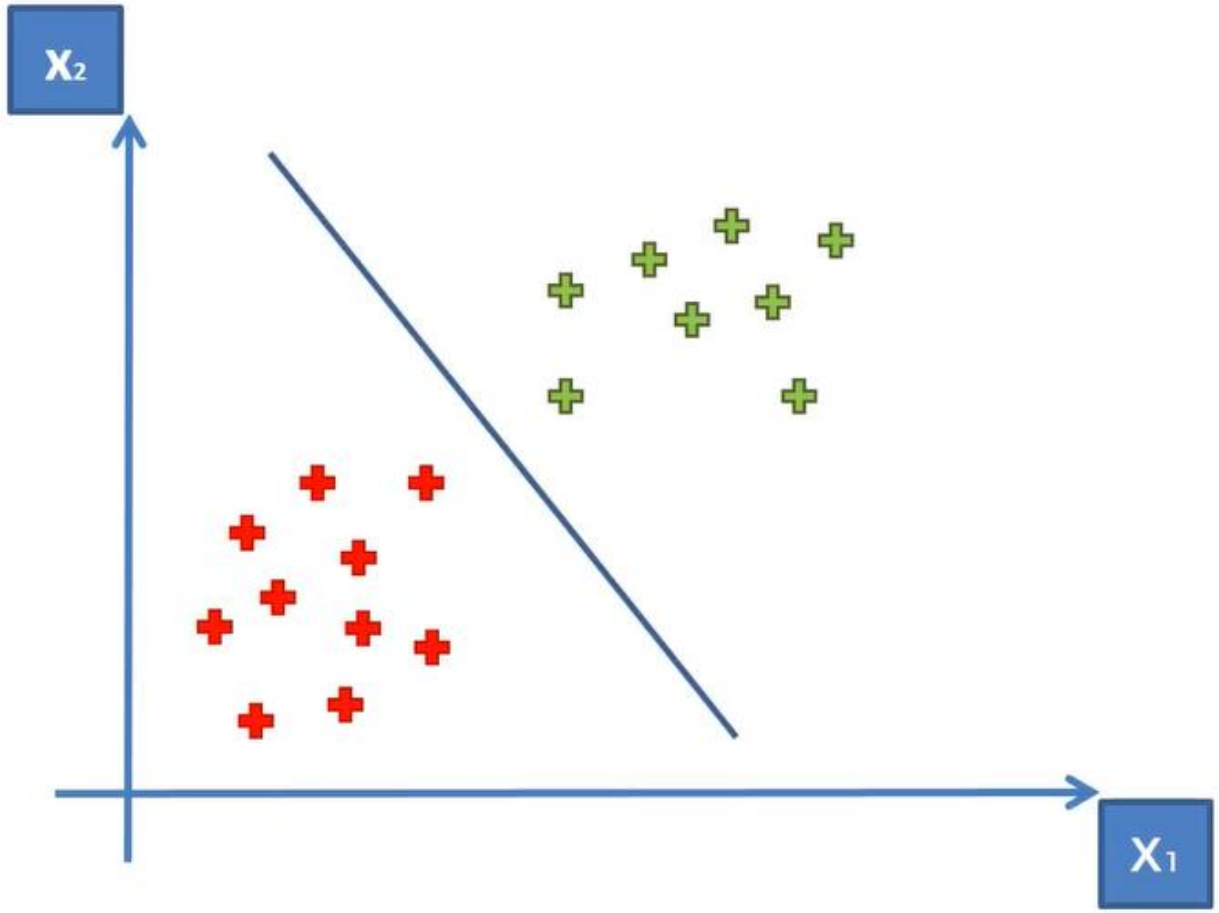


Why ?

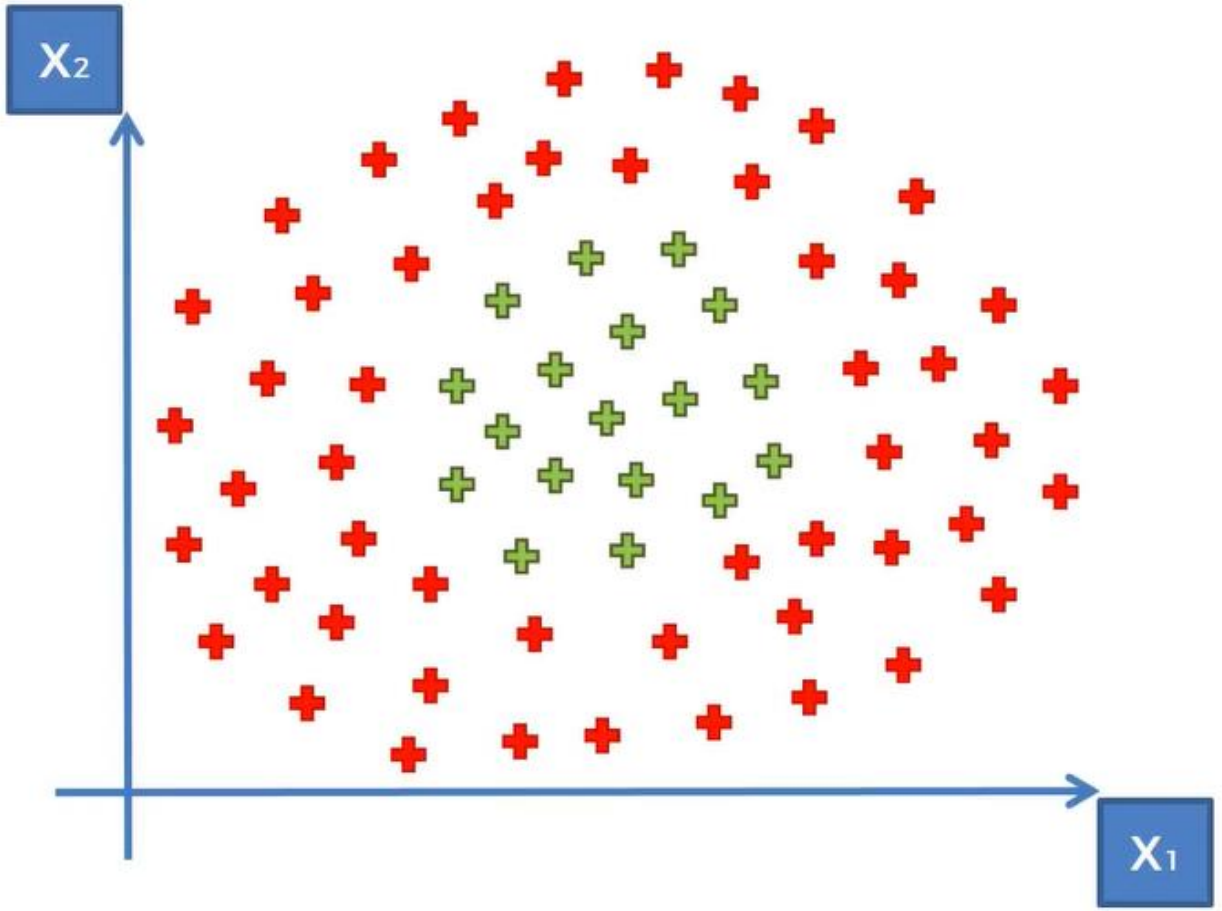
Because the data points are not LINEARLY SEPARABLE

Linear Separability

Linearly Separable

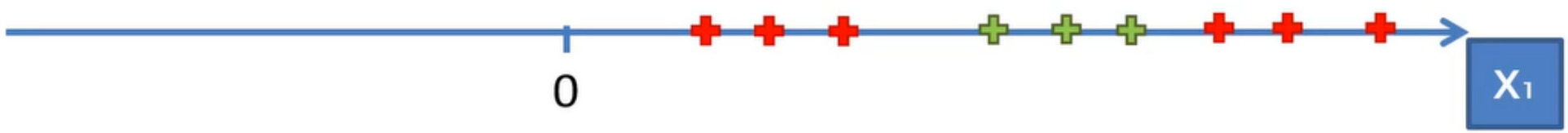


Not Linearly Separable



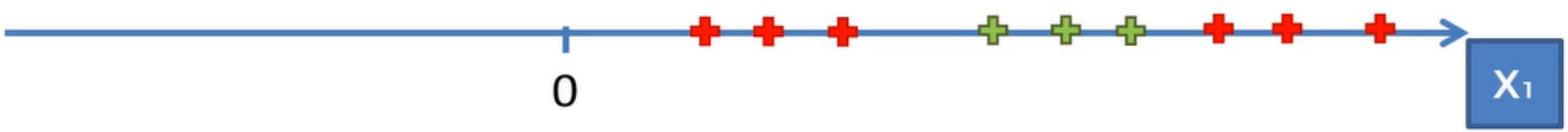
A Higher-Dimensional Space

Mapping to a Higher Dimension



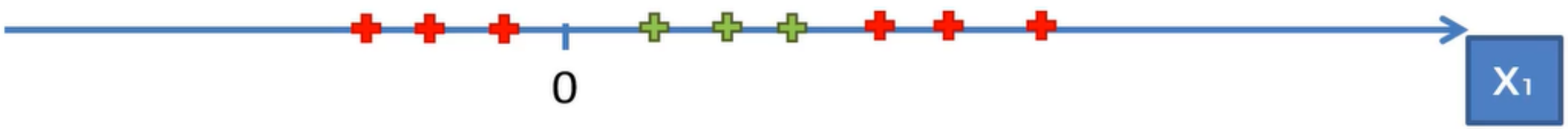
Mapping to a Higher Dimension

Mapping function → $f = x - 5$



Mapping to a Higher Dimension

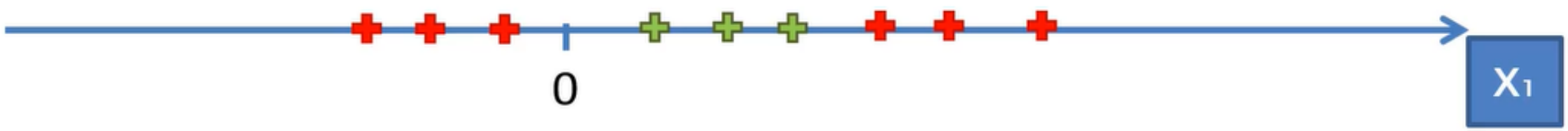
$f = x - 5$



Mapping to a Higher Dimension

$f = x - 5$

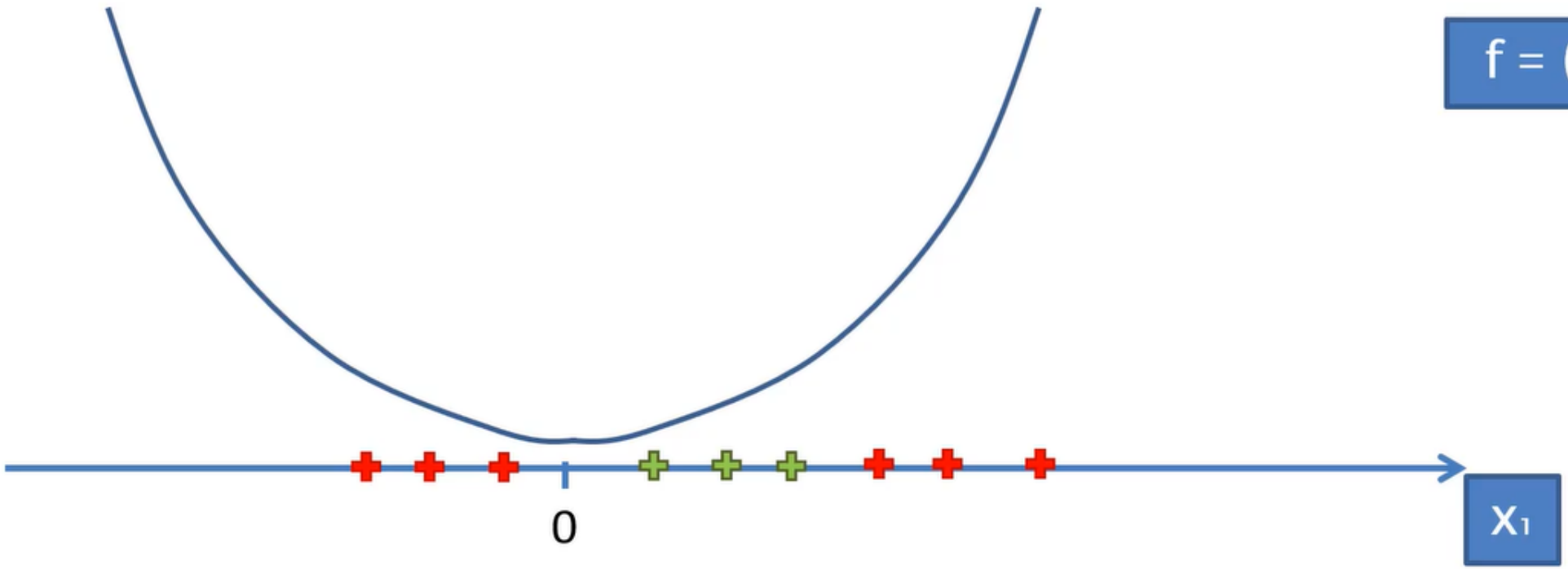
$f = (x - 5)^2$



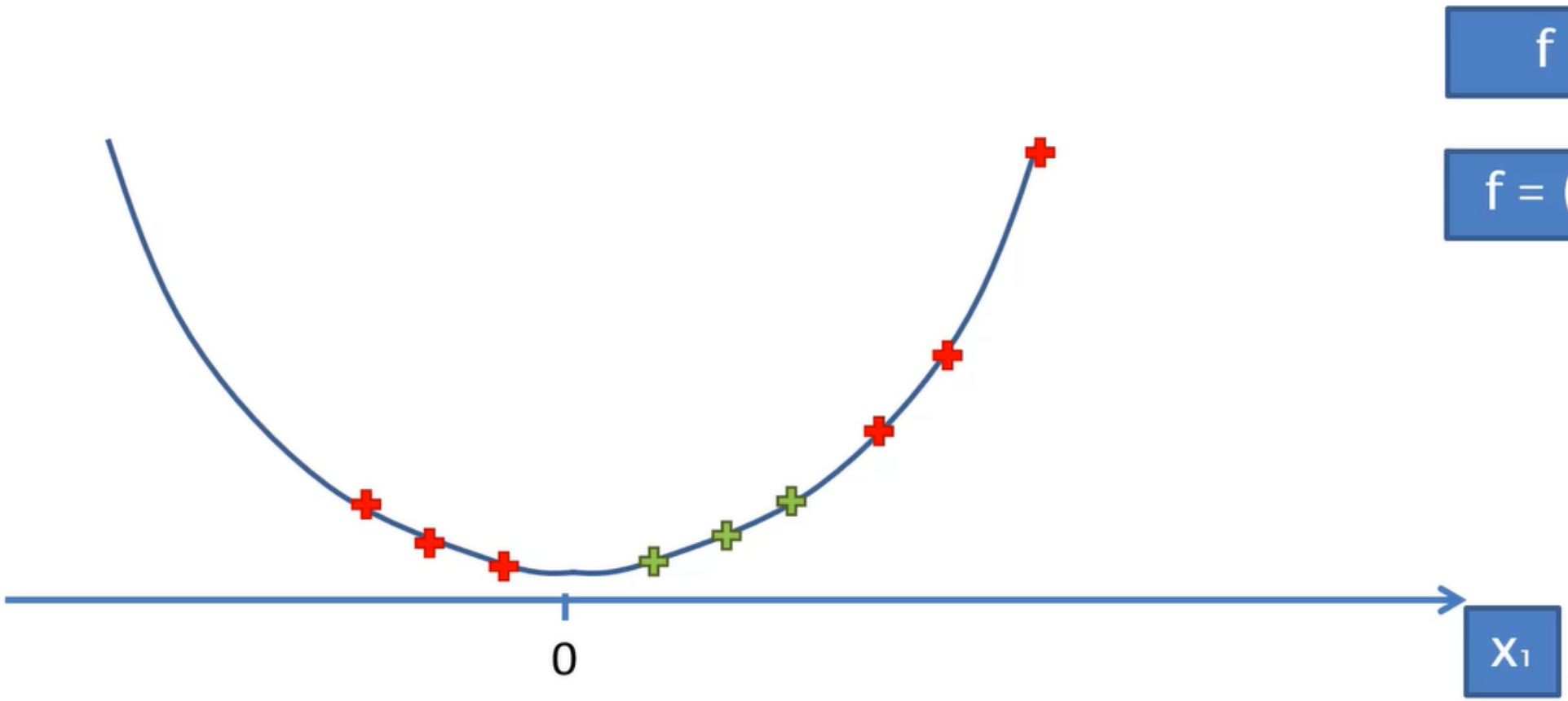
Mapping to a Higher Dimension

f = x - 5

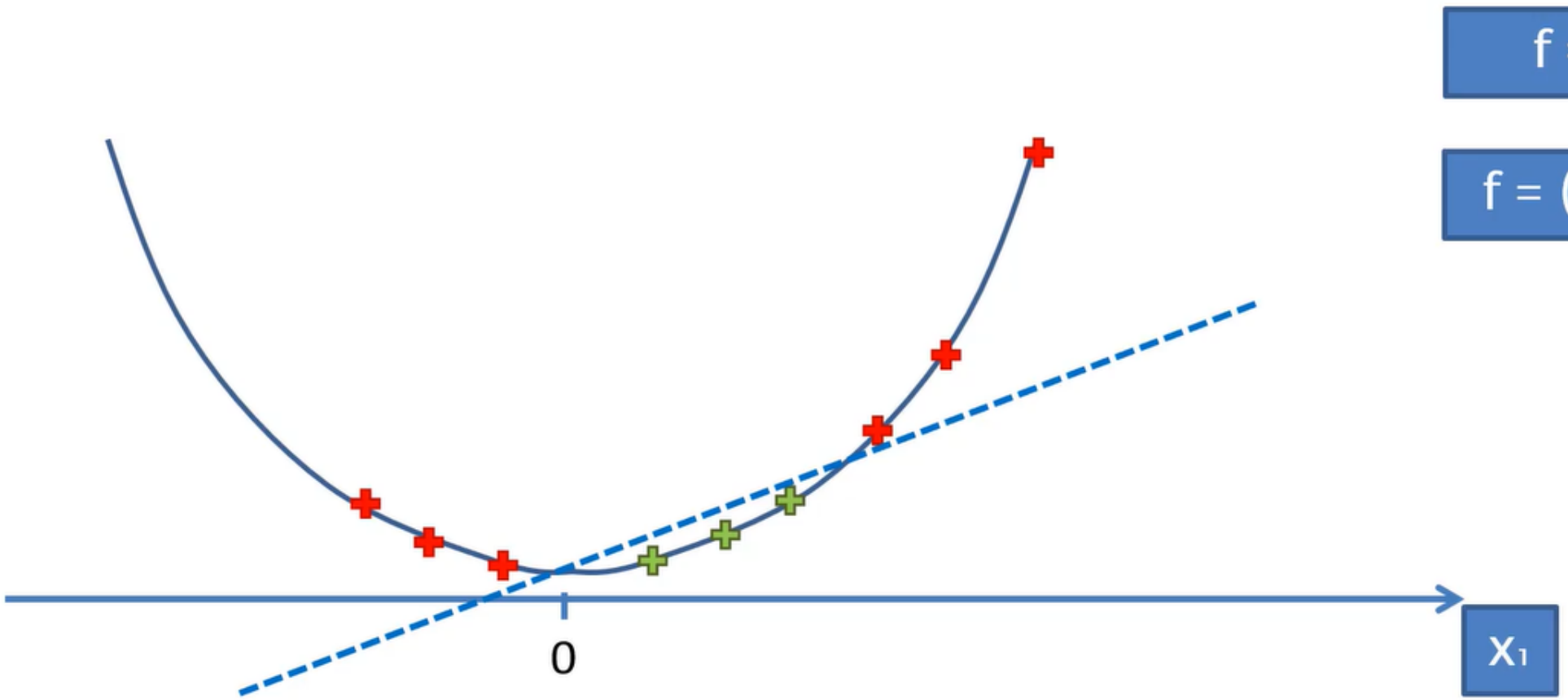
f = (x – 5)^2



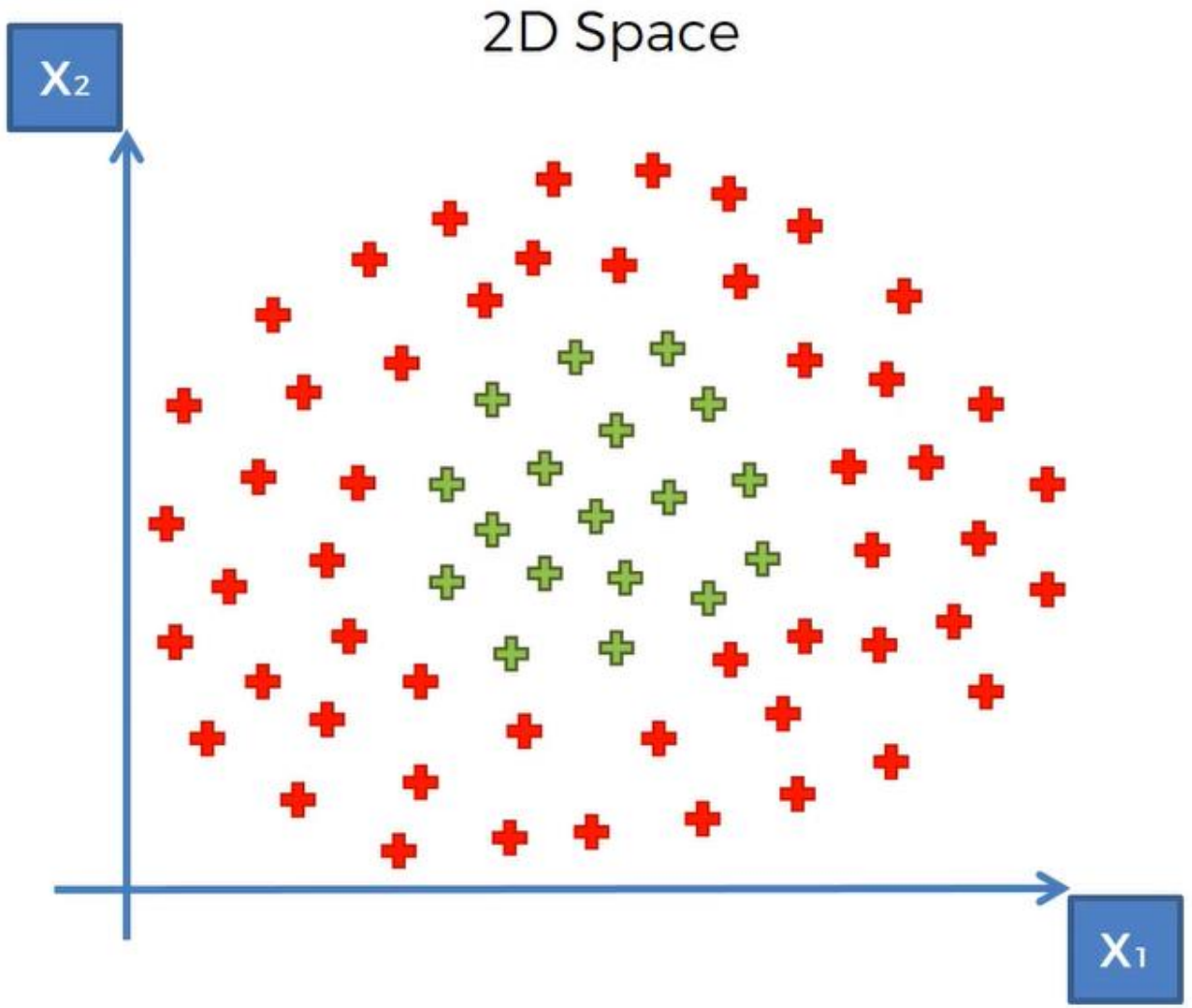
Mapping to a Higher Dimension



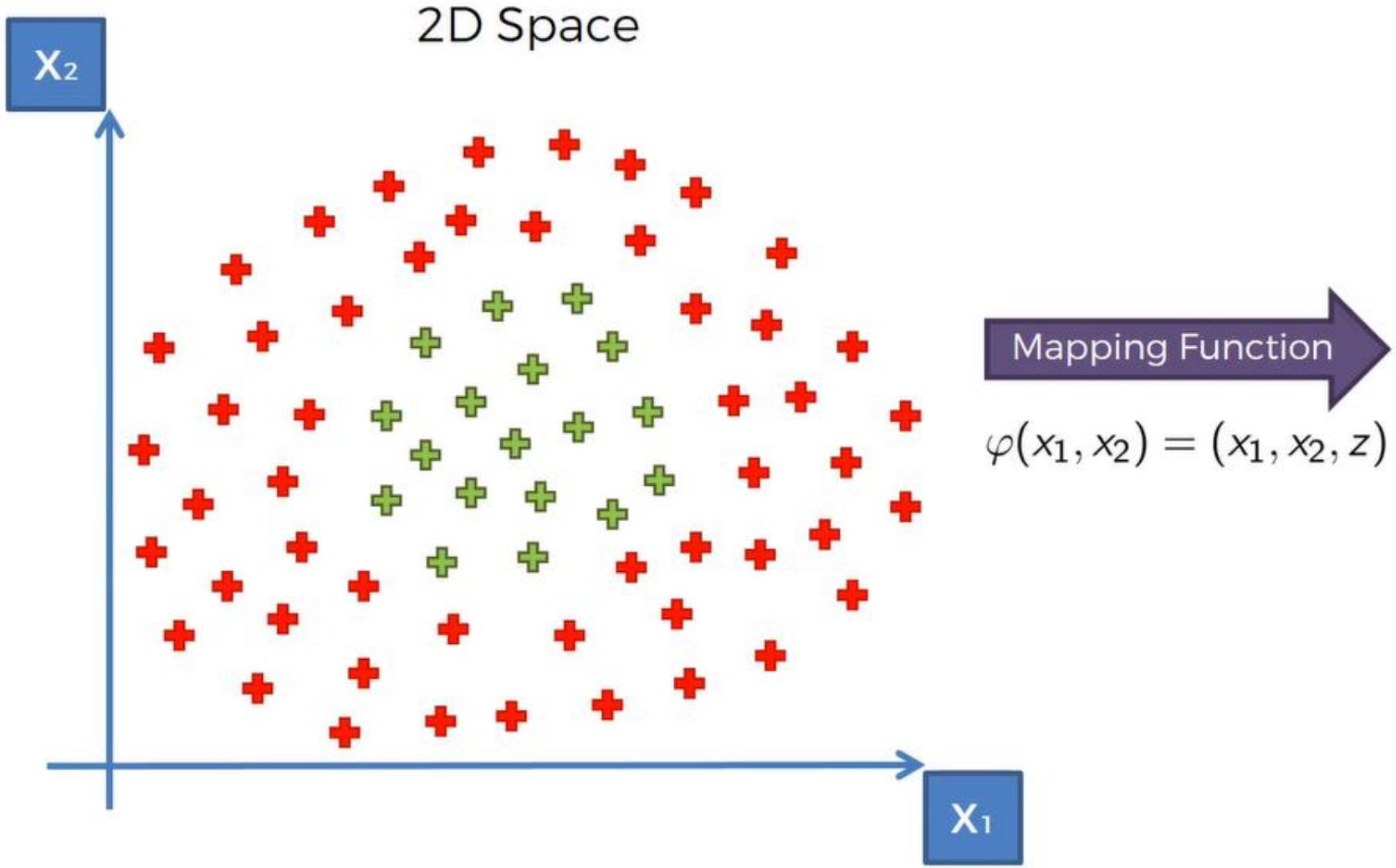
Mapping to a Higher Dimension



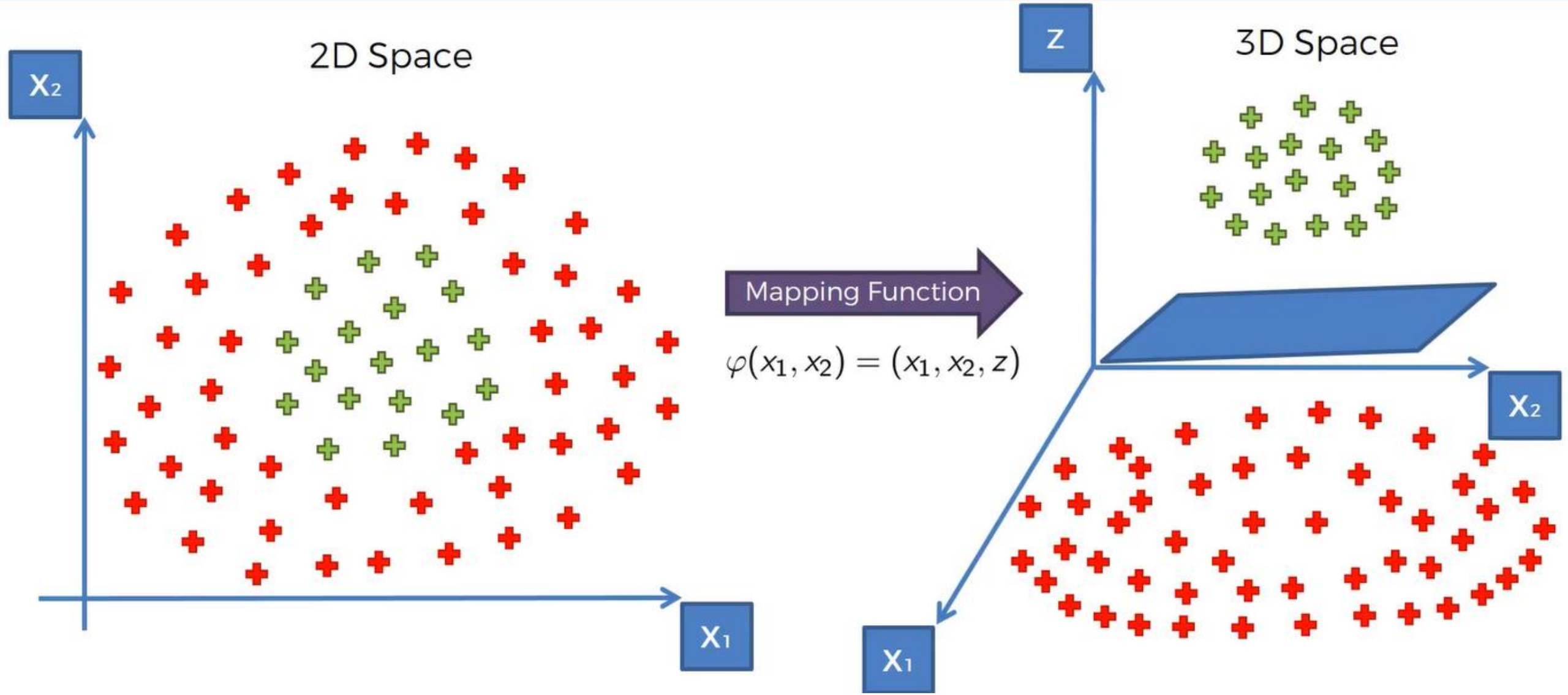
Mapping to a Higher Dimension



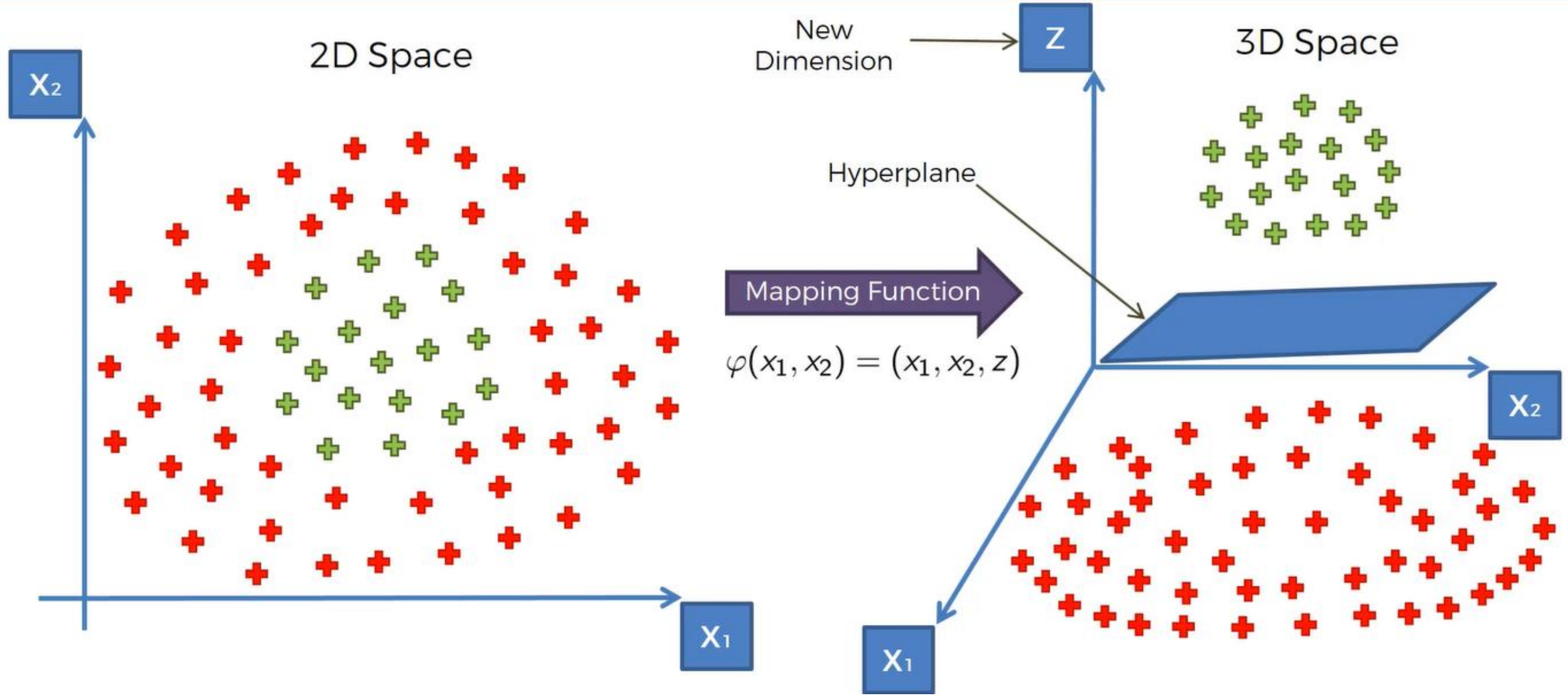
Mapping to a Higher Dimension



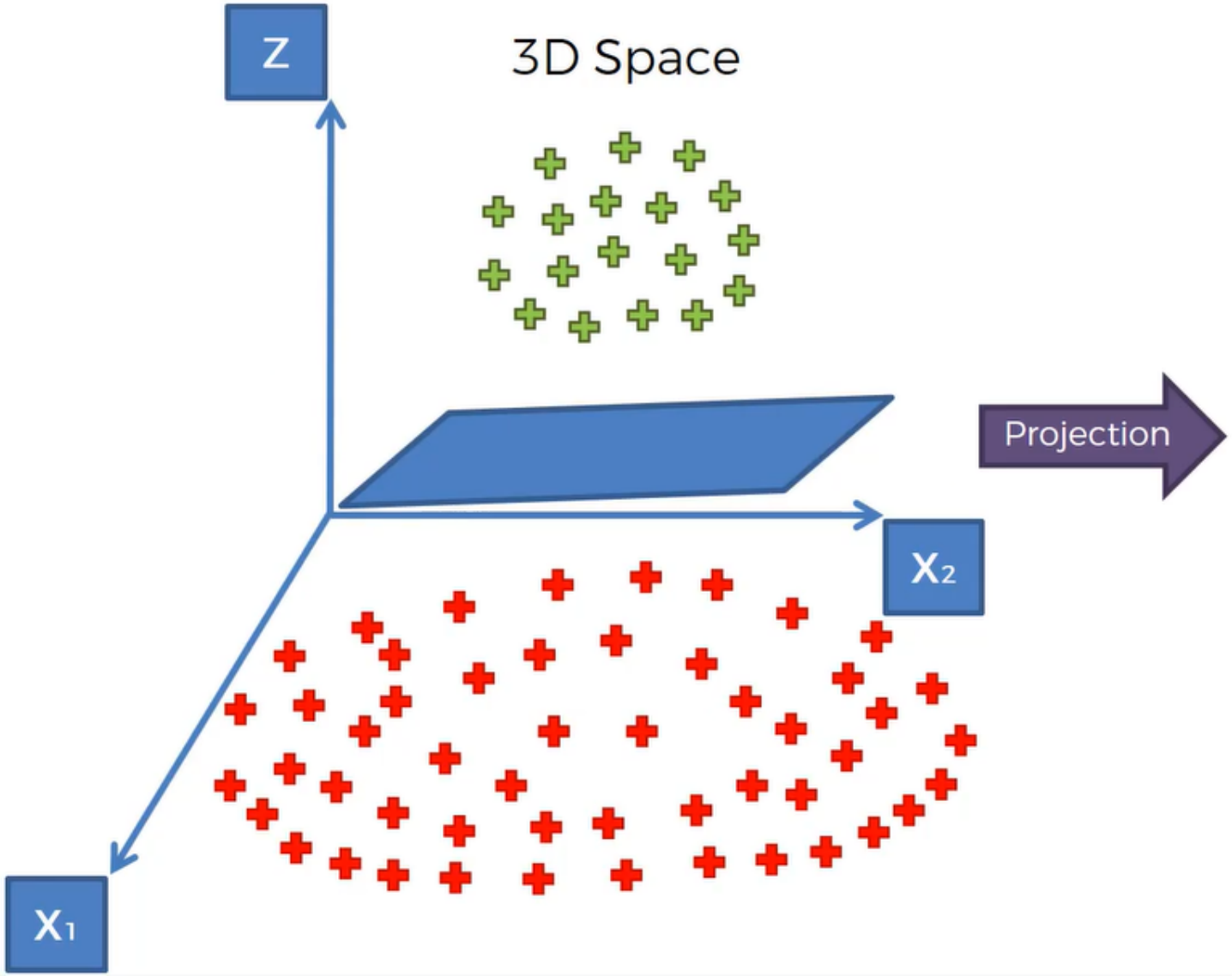
Mapping to a Higher Dimension



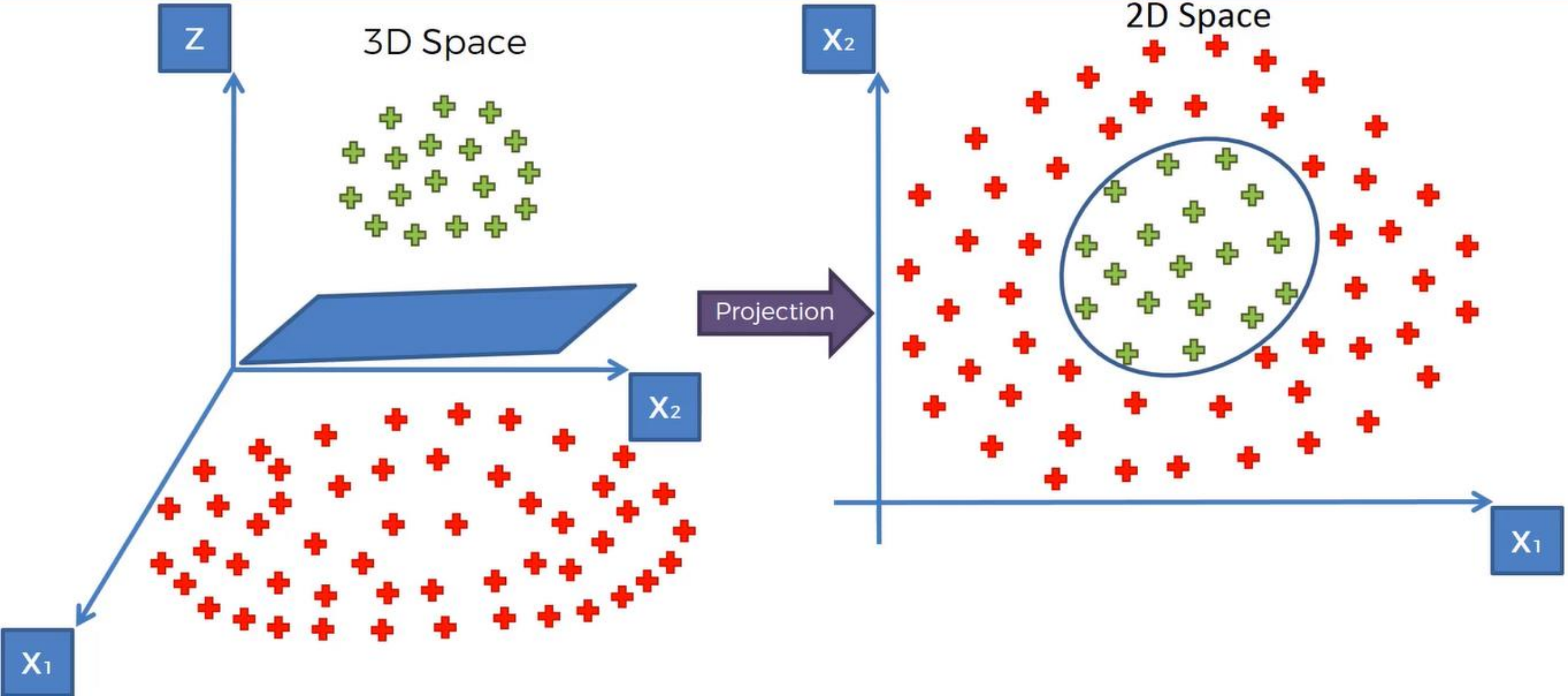
Mapping to a Higher Dimension



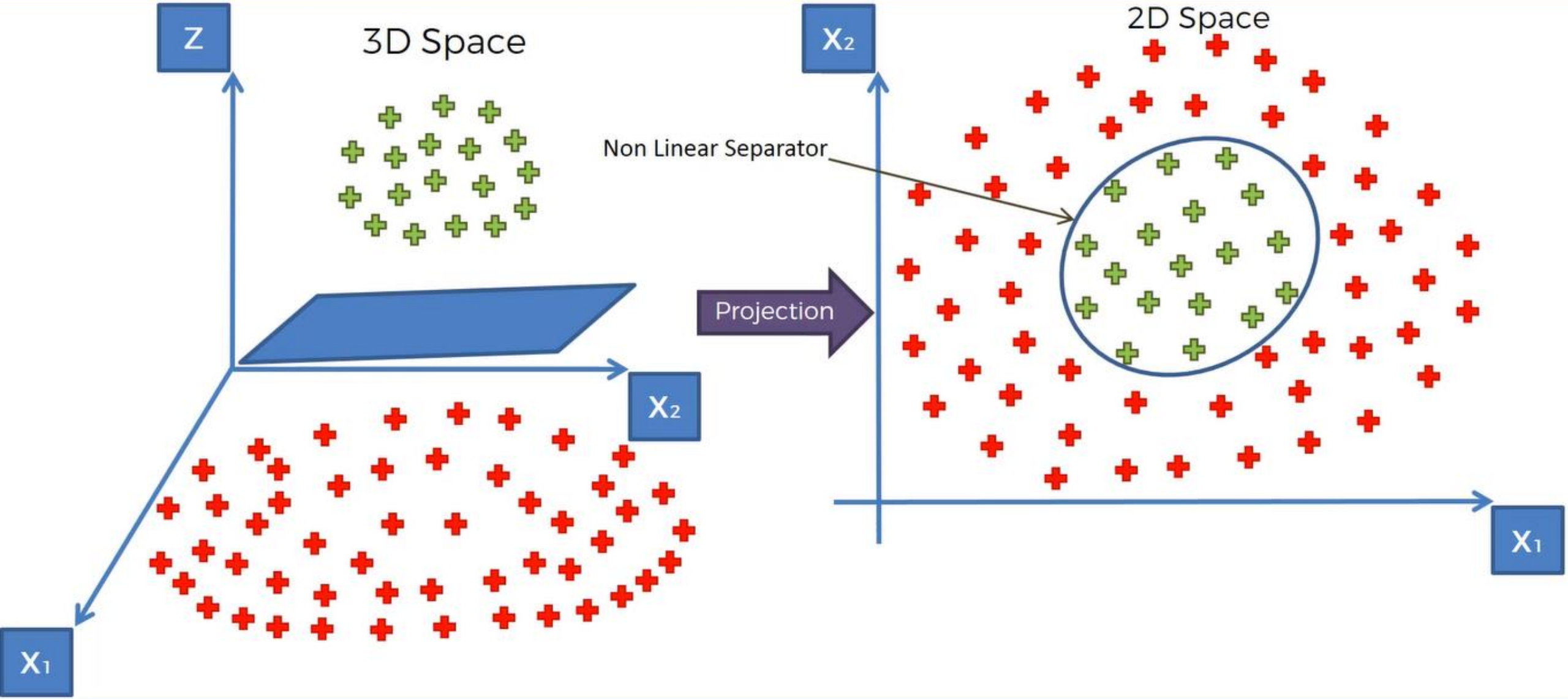
Projecting back to 2D Space



Projecting back to 2D Space



Projecting back to 2D Space



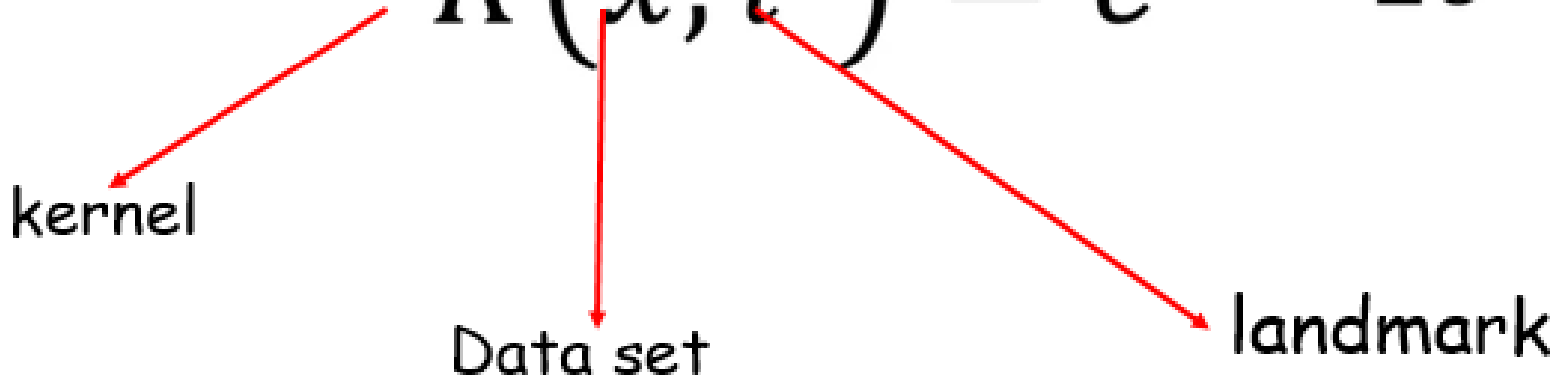
But there is a catch...

Mapping to a Higher Dimensional Space can be highly compute-intensive

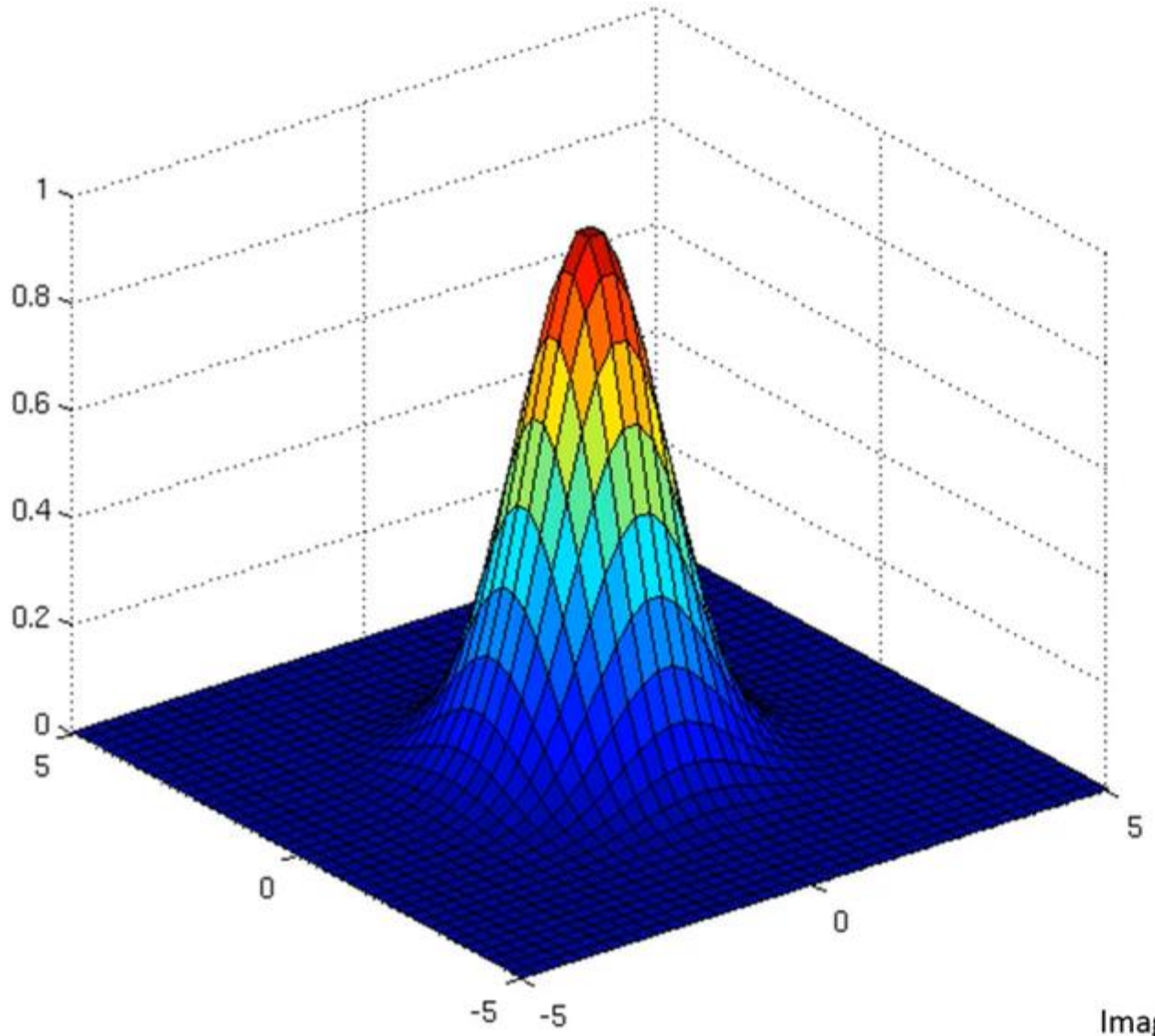
The Kernel Trick

The Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$



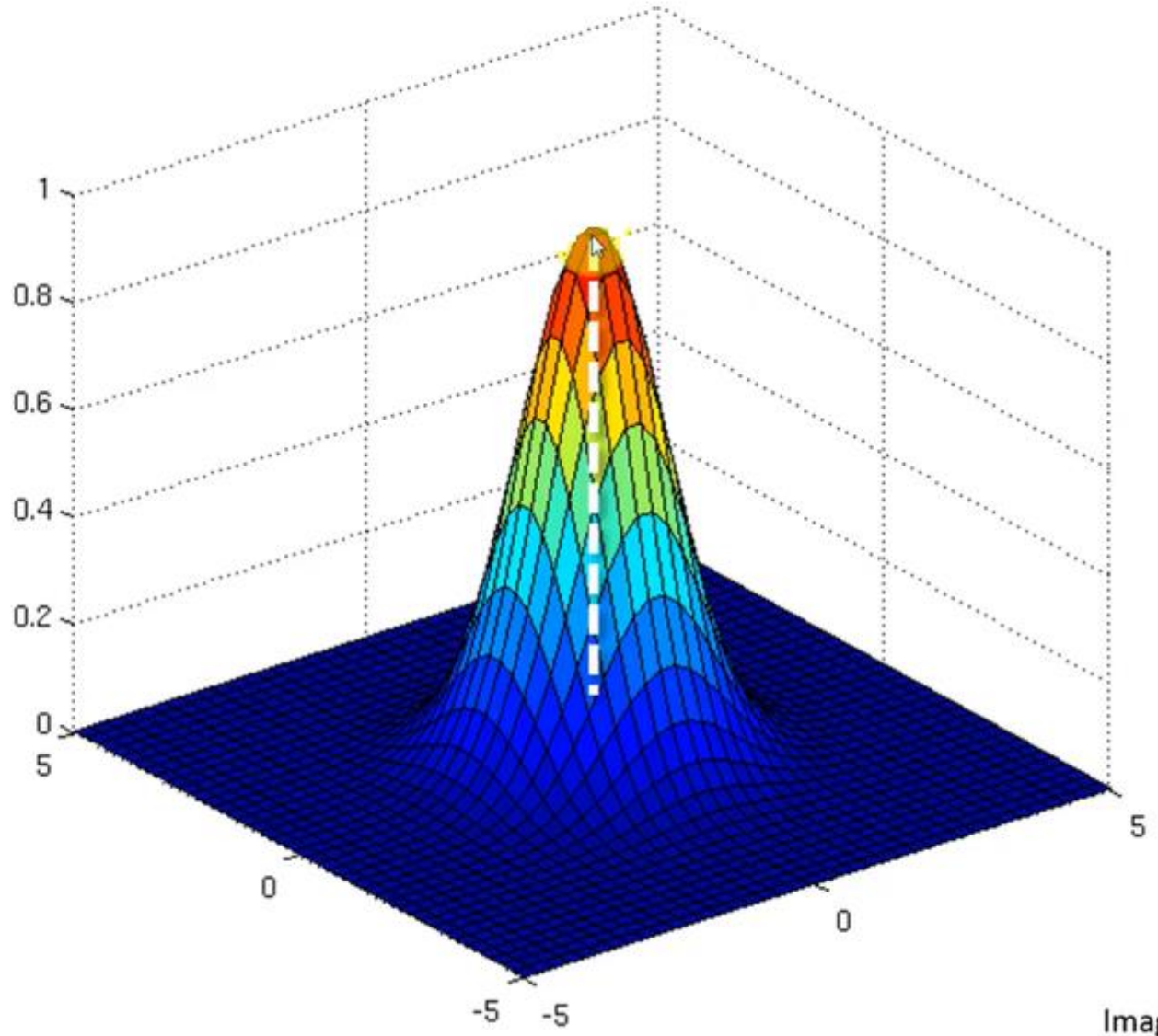
The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

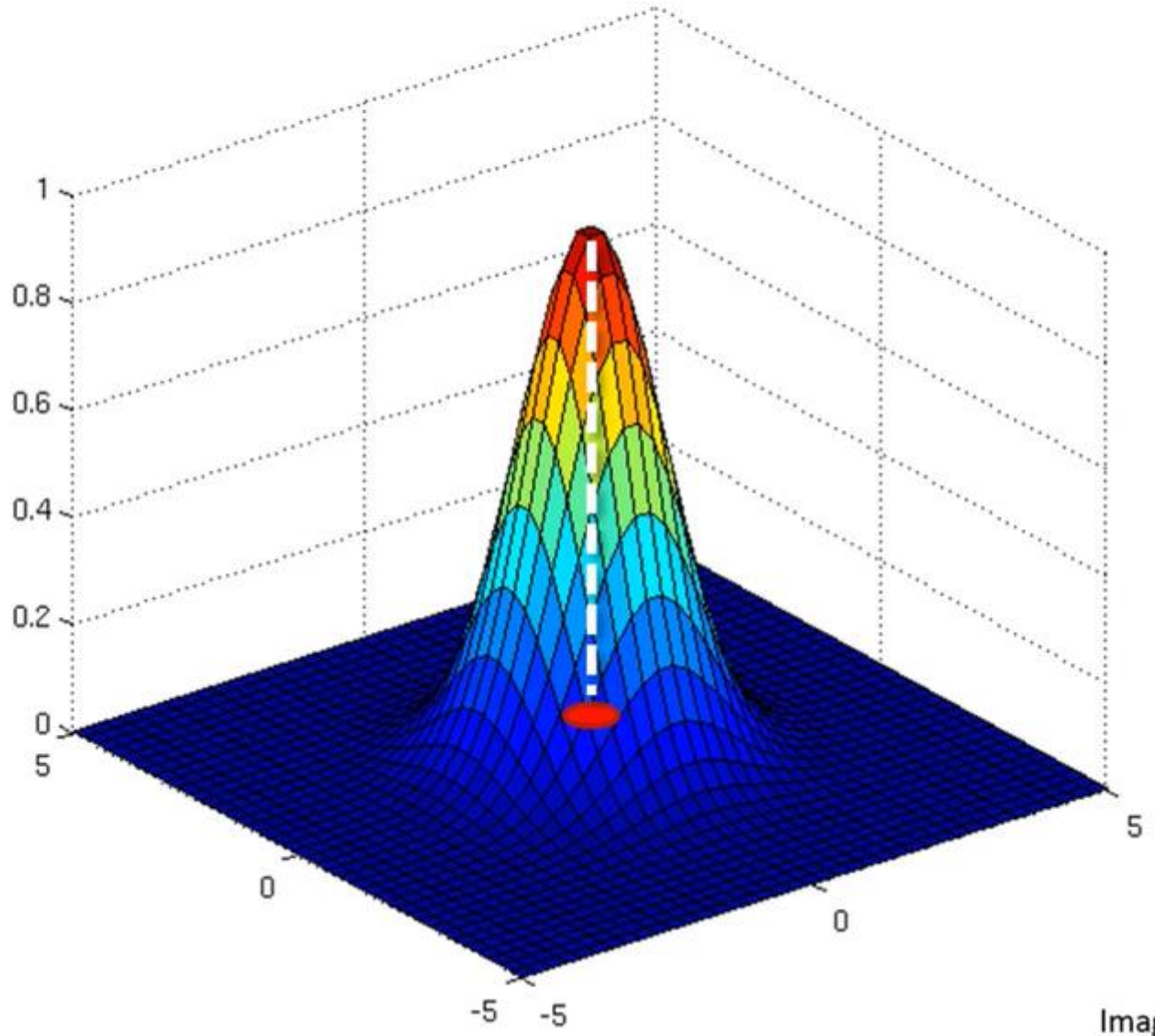
The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

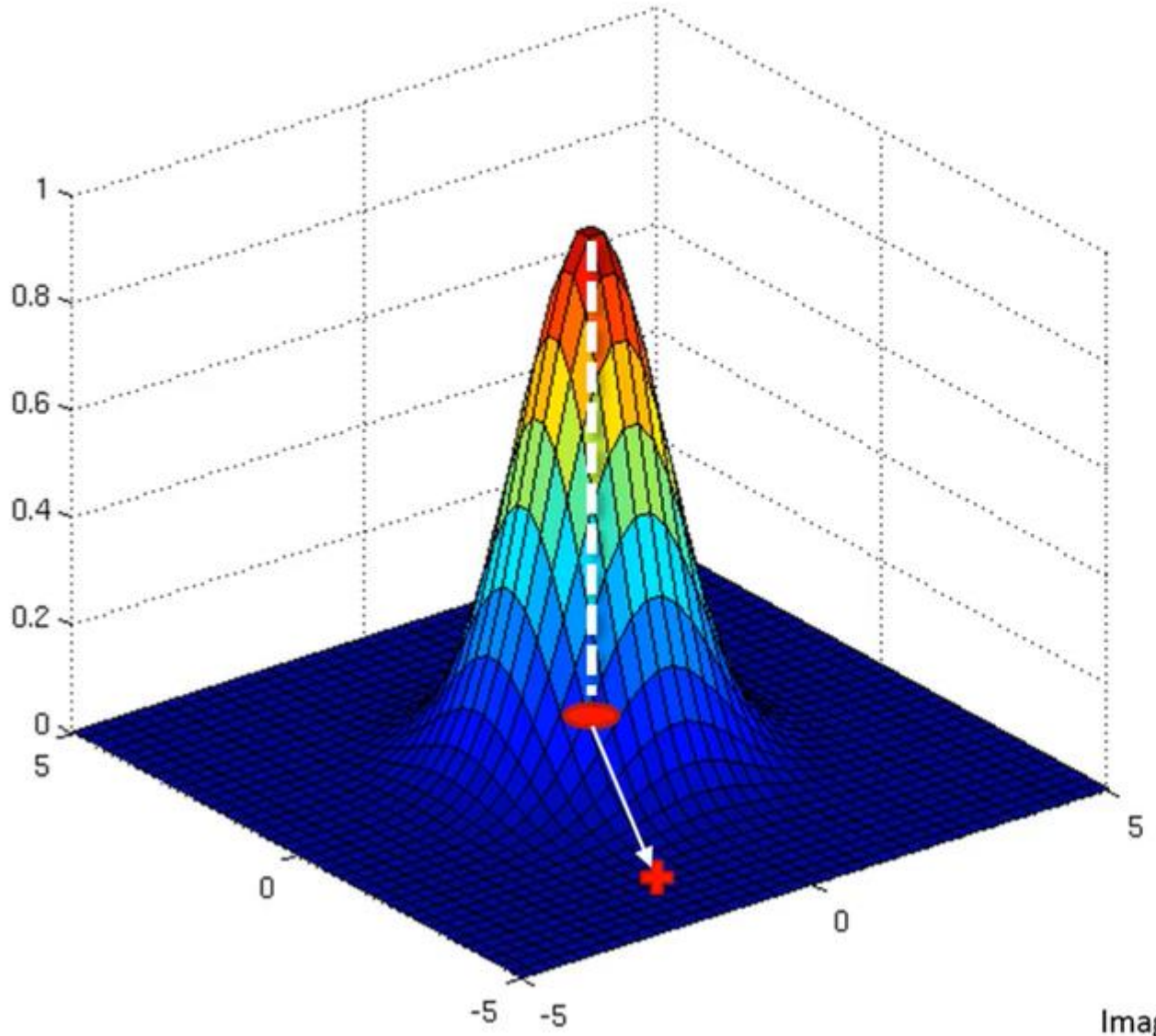
The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

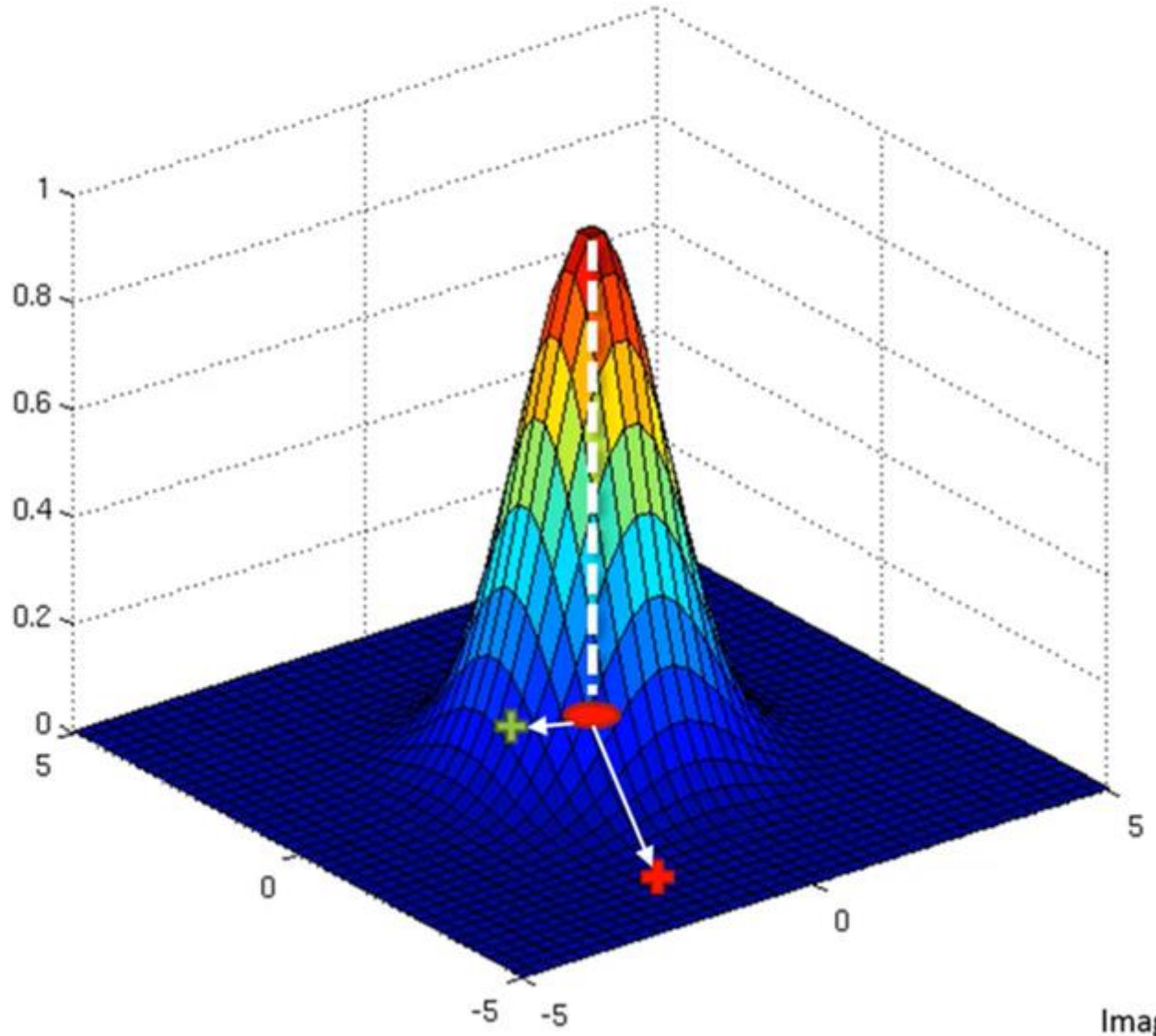
The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

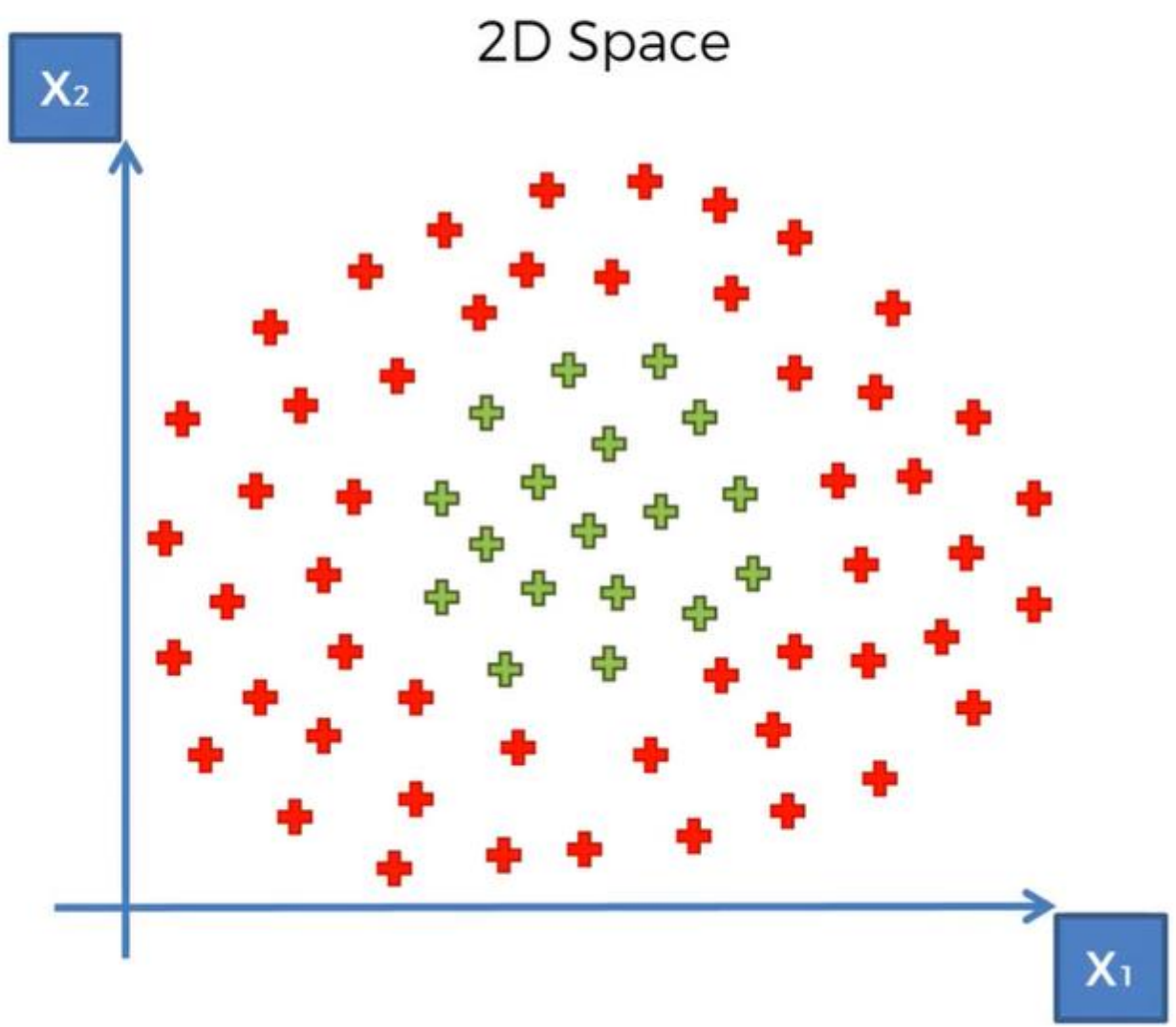
The Gaussian RBF Kernel

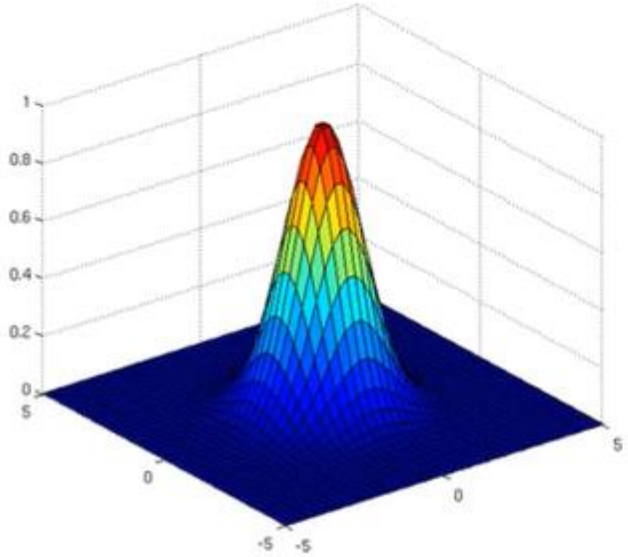


$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$

Image source: http://www.cs.toronto.edu/~duvenaud/cookbook/index.html

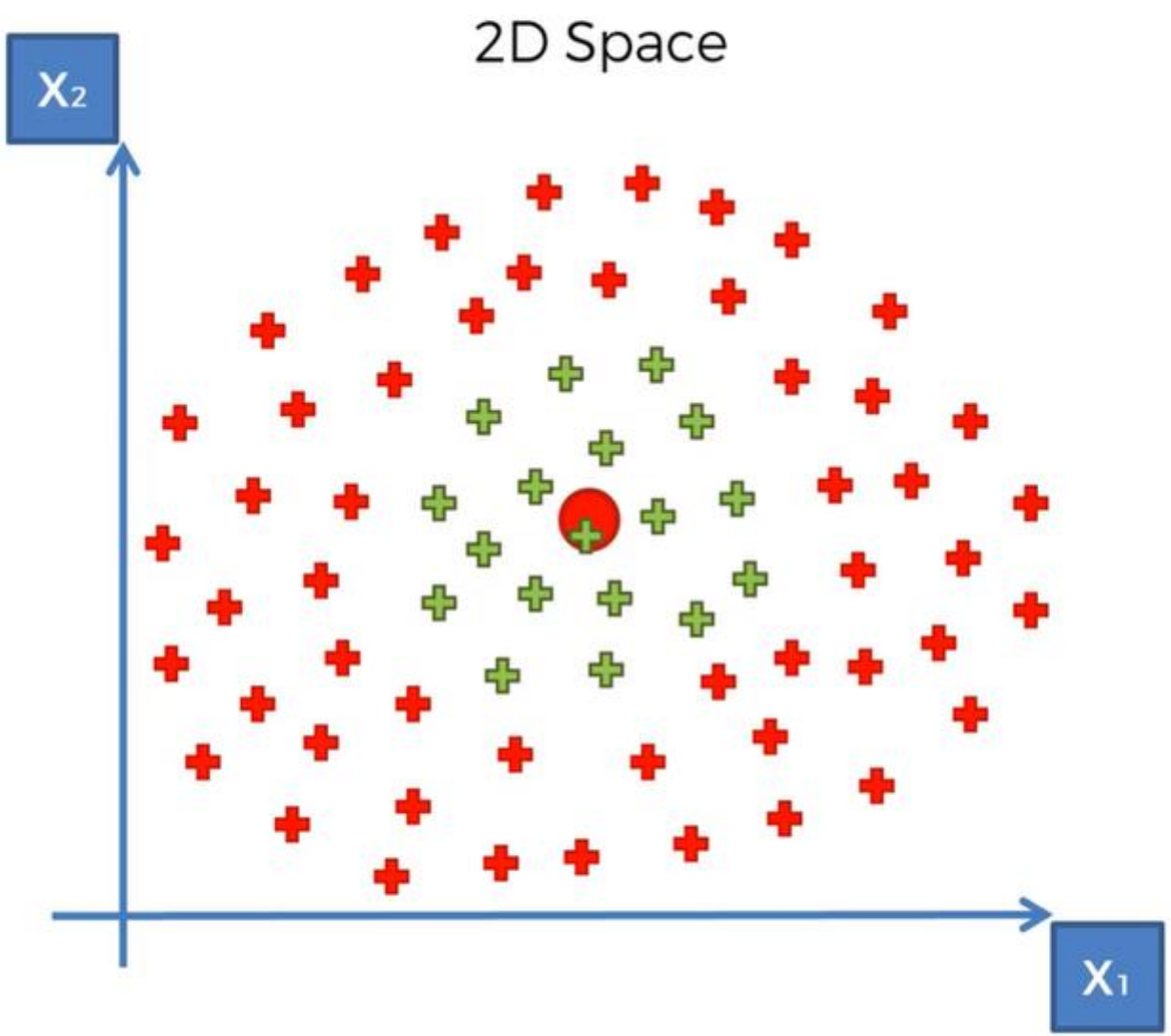
The Gaussian RBF Kernel

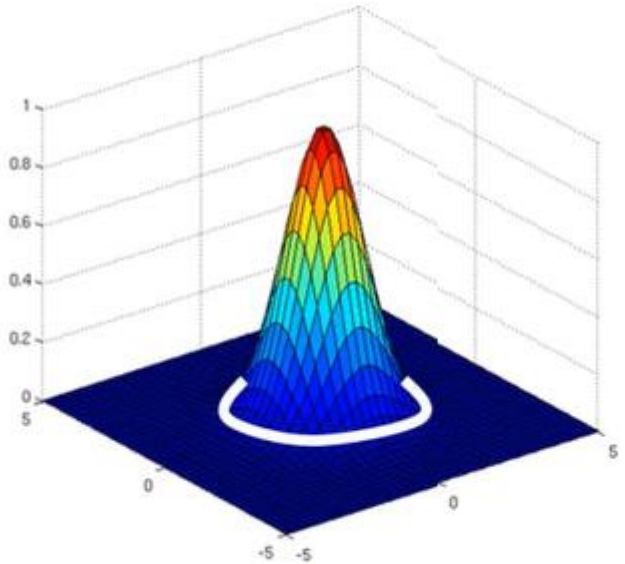




$$K\left(\vec{x}, \vec{l}^{i}\right) = e^{-\frac{\left\|\vec{x}-\vec{l}^{i}\right\|^{2}}{2\sigma^{2}}}$$

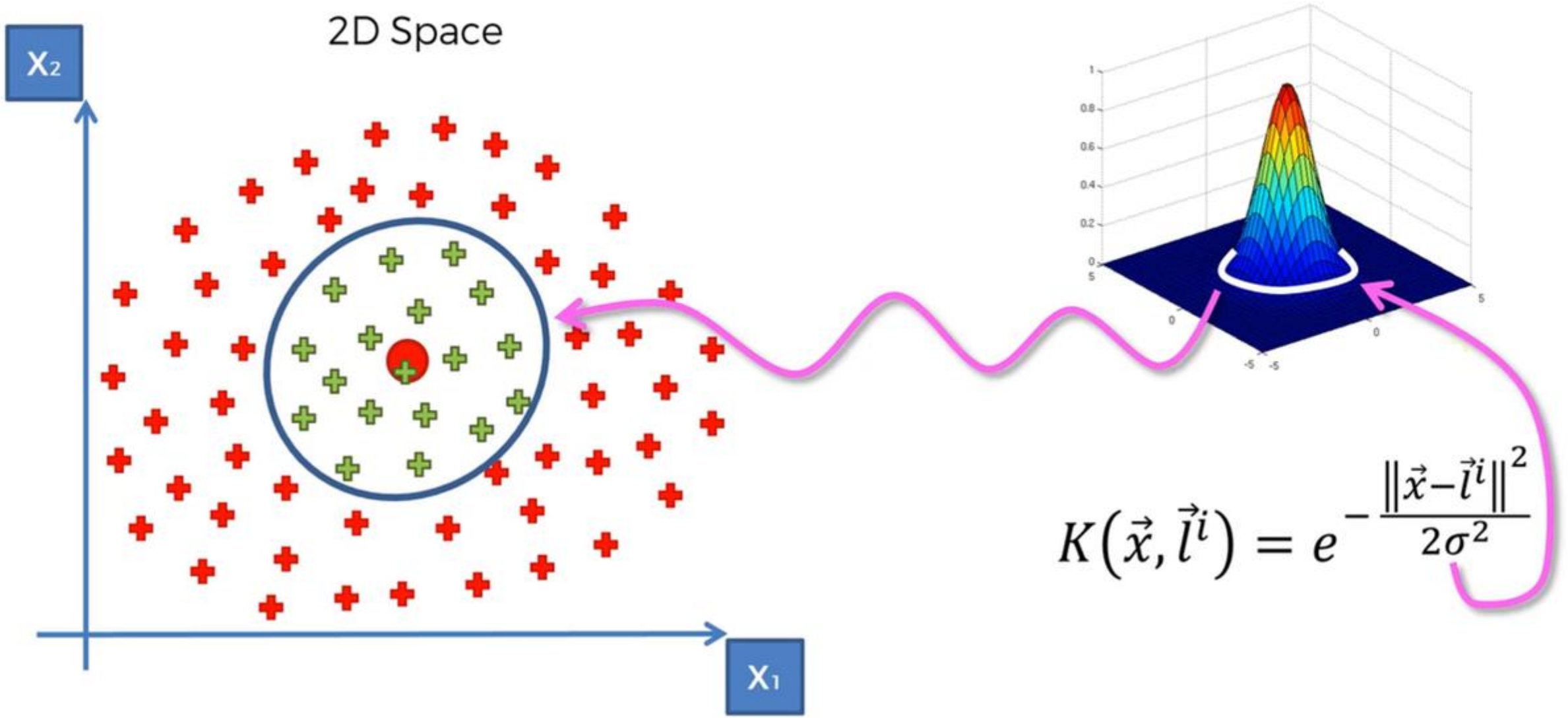
The Gaussian RBF Kernel



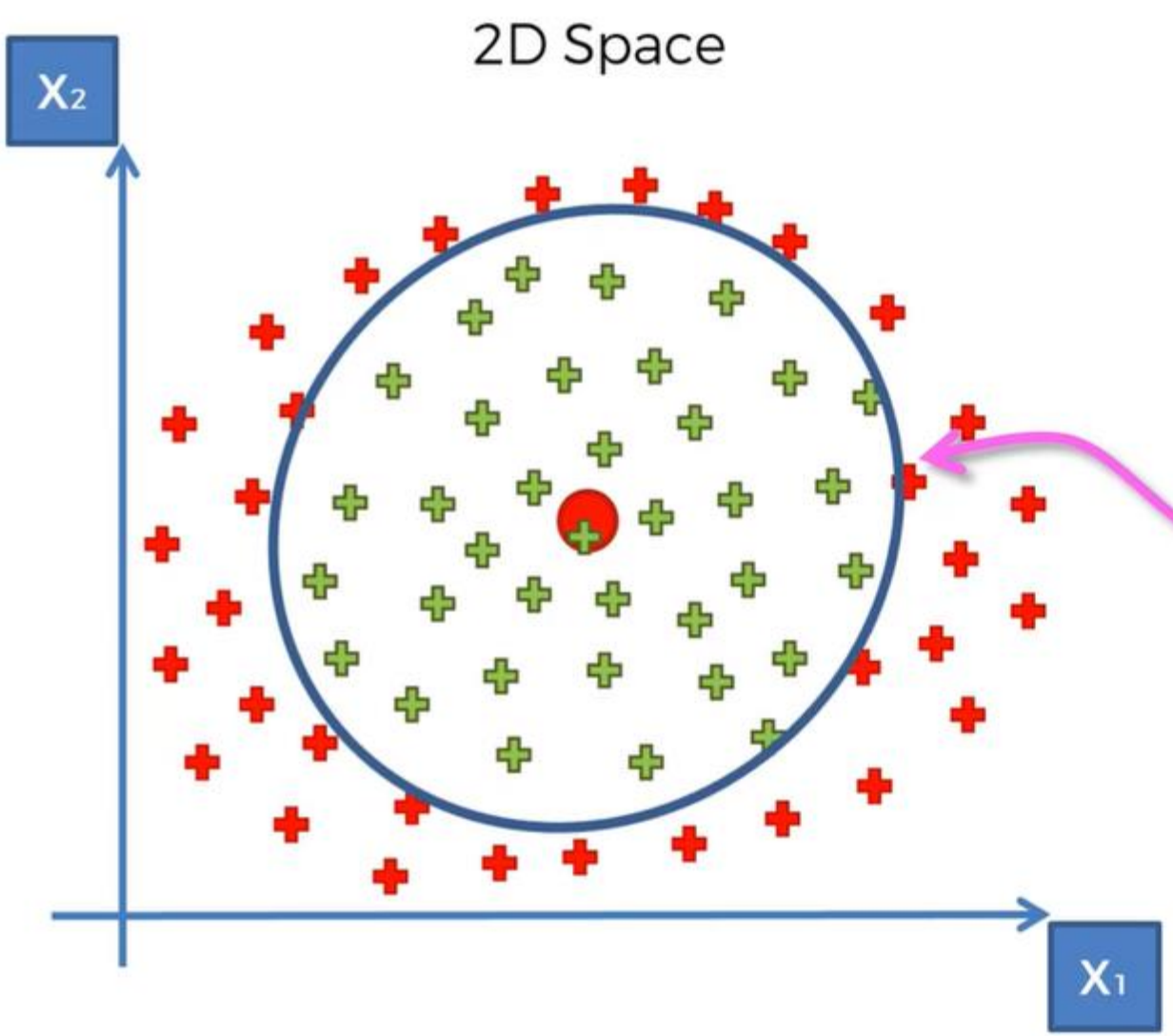


$$K\left(\vec{x}, \vec{l}^{i}\right) = e^{-\frac{\left\|\vec{x}-\vec{l}^{i}\right\|^{2}}{2\sigma^{2}}}$$

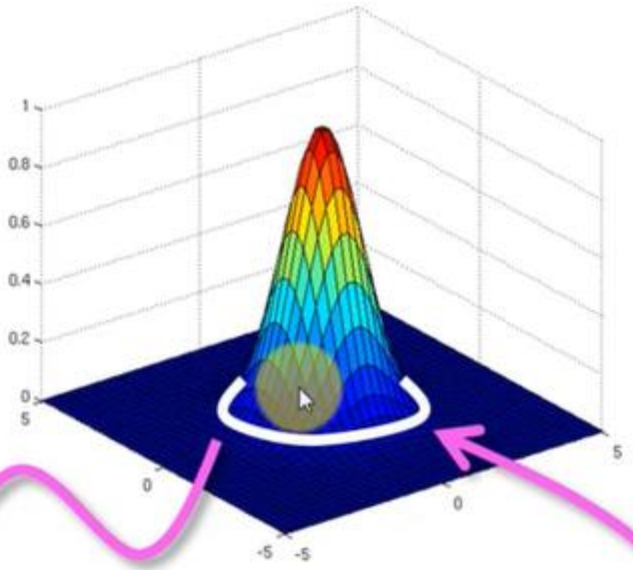
The Gaussian RBF Kernel



The Gaussian RBF Kernel

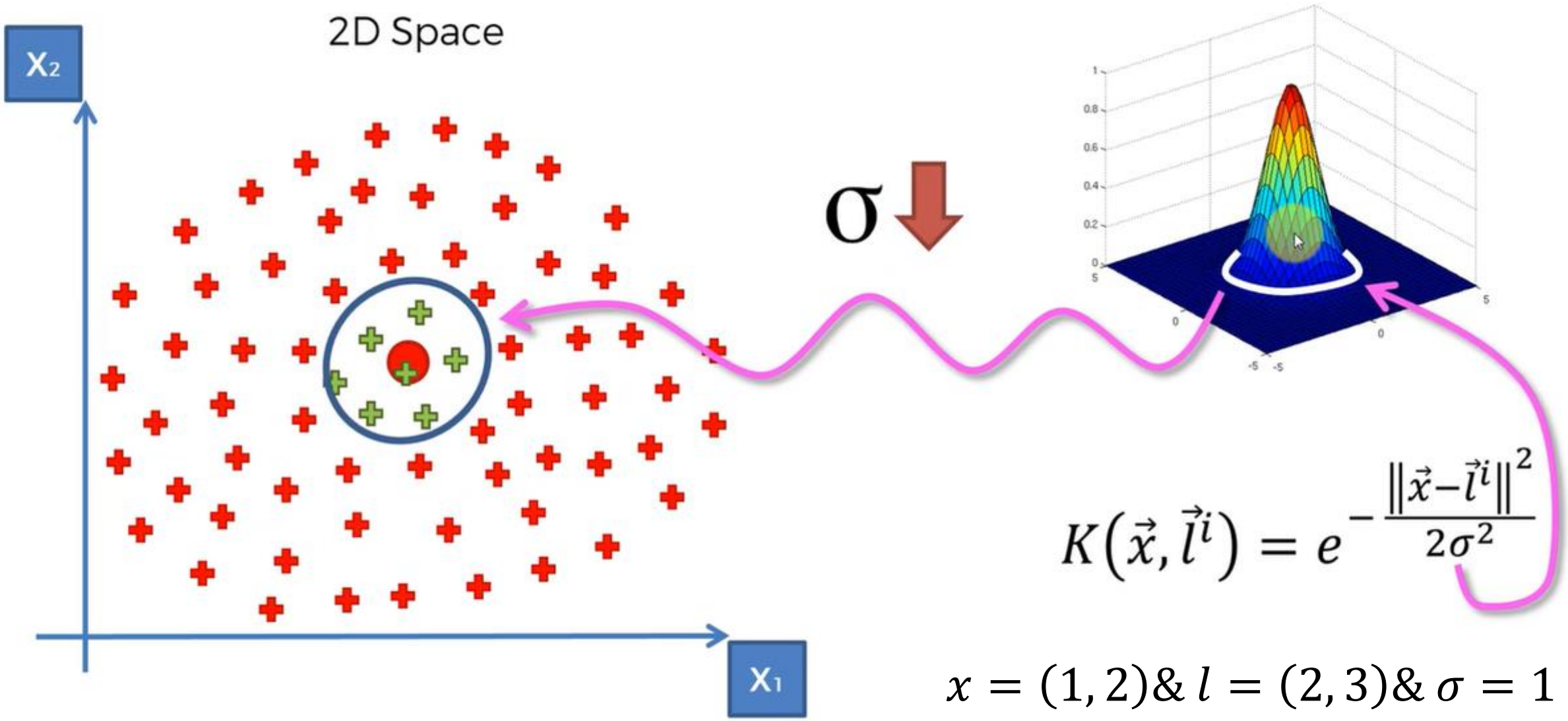


σ

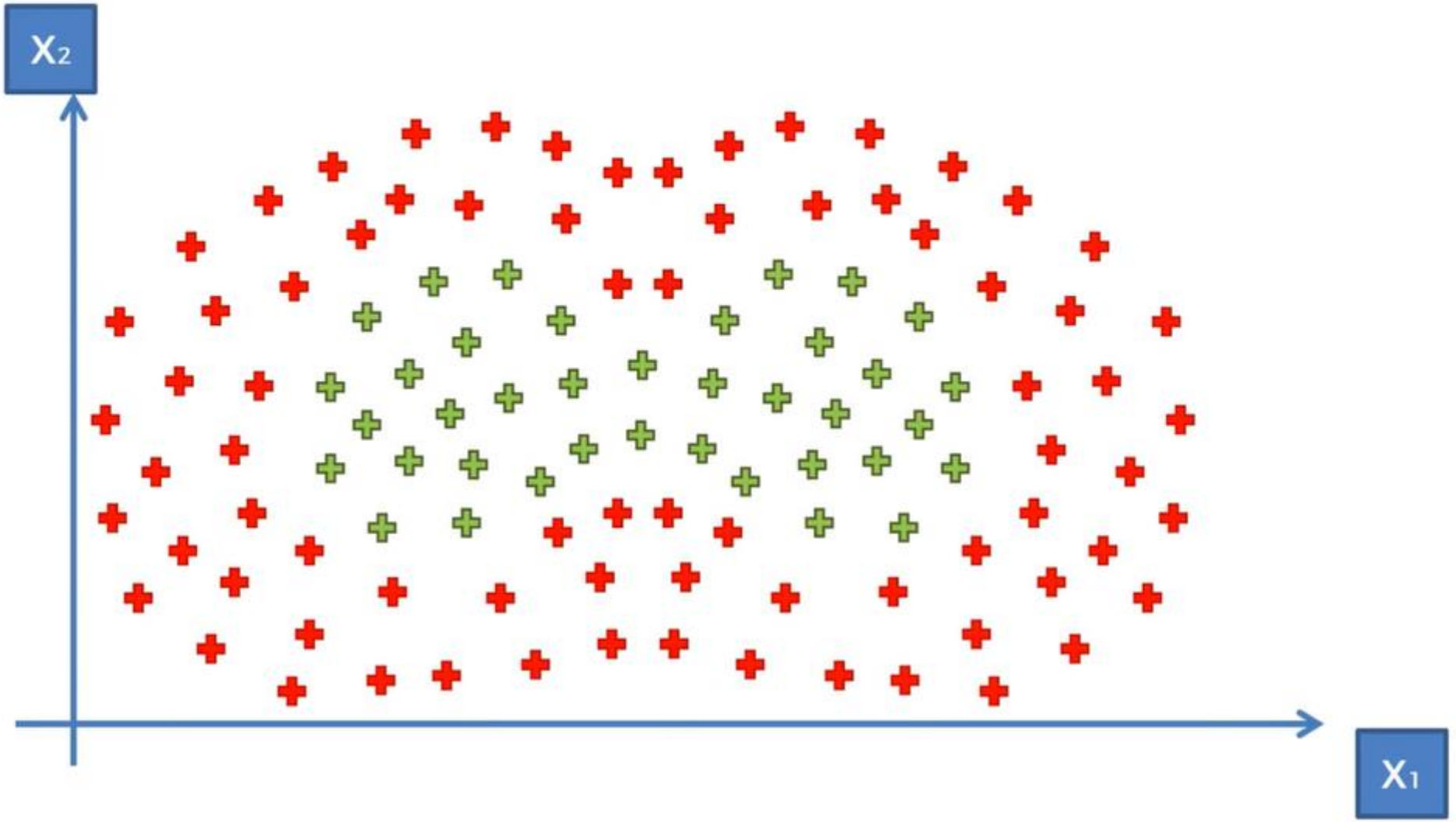


$$K\left(\vec{x}, \vec{l}^{i}\right) = e^{-\frac{\left\|\vec{x}-\vec{l}^{i}\right\|^{2}}{2\sigma^{2}}}$$

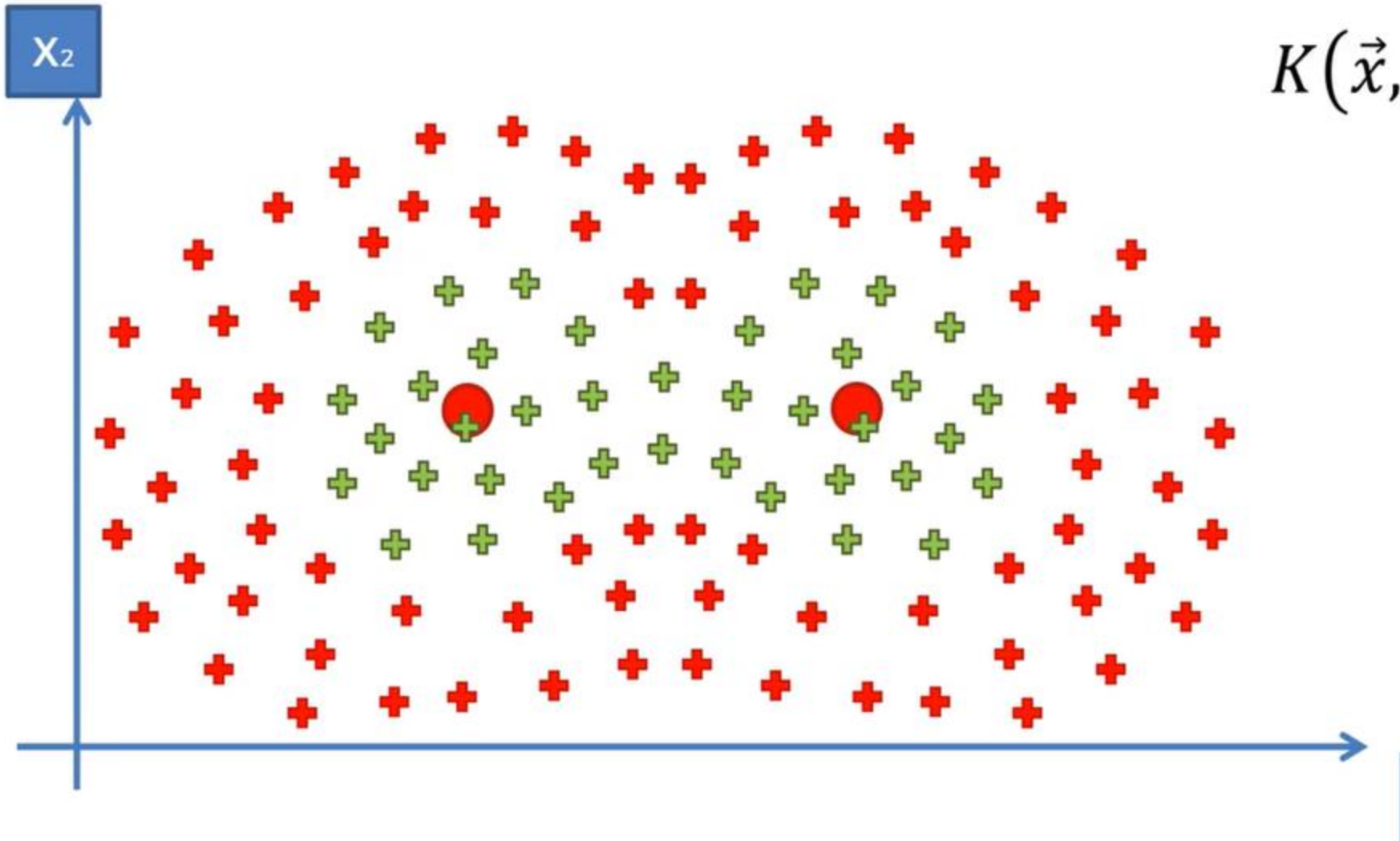
The Gaussian RBF Kernel



The Gaussian RBF Kernel



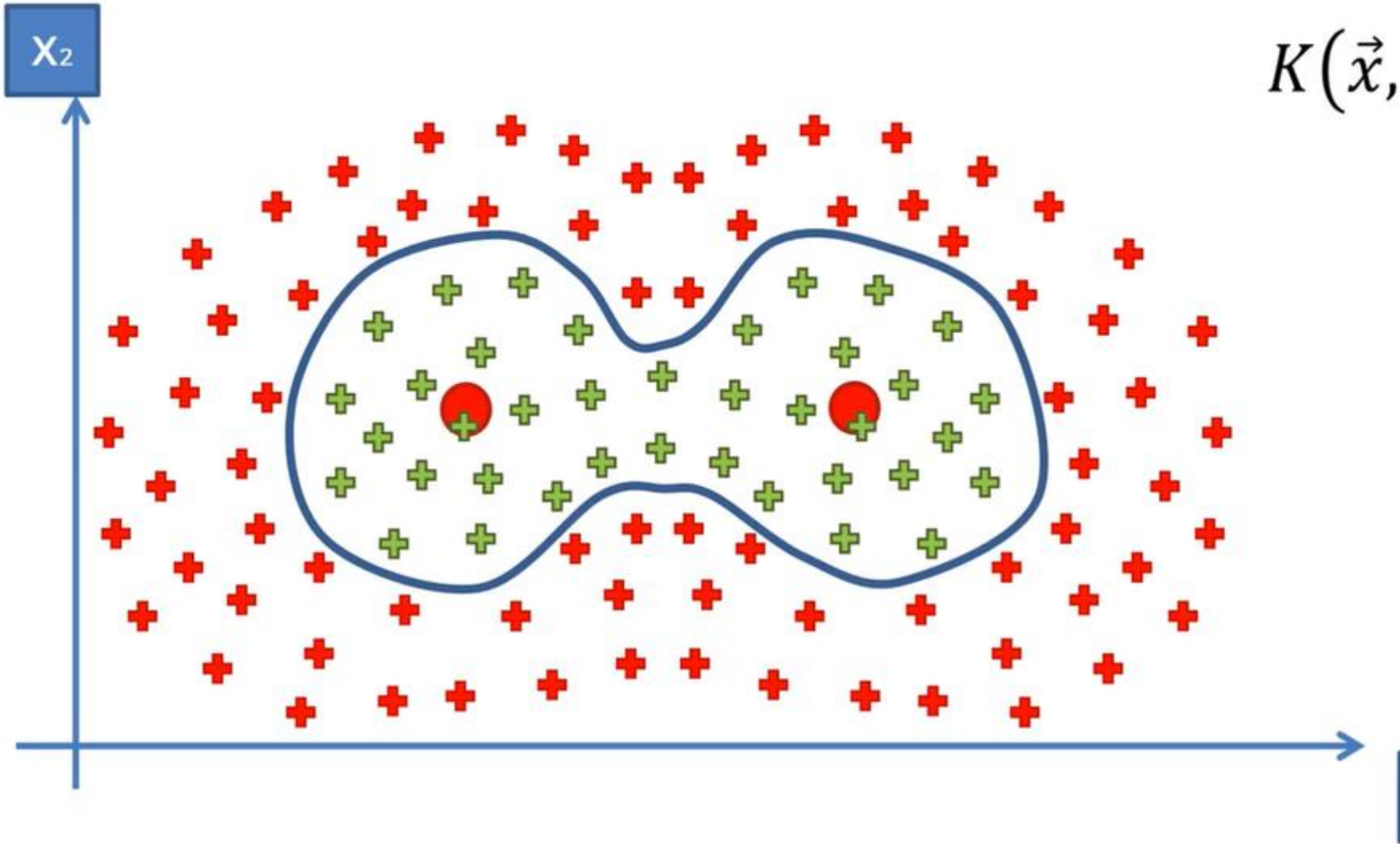
The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

(Simplified Formula)

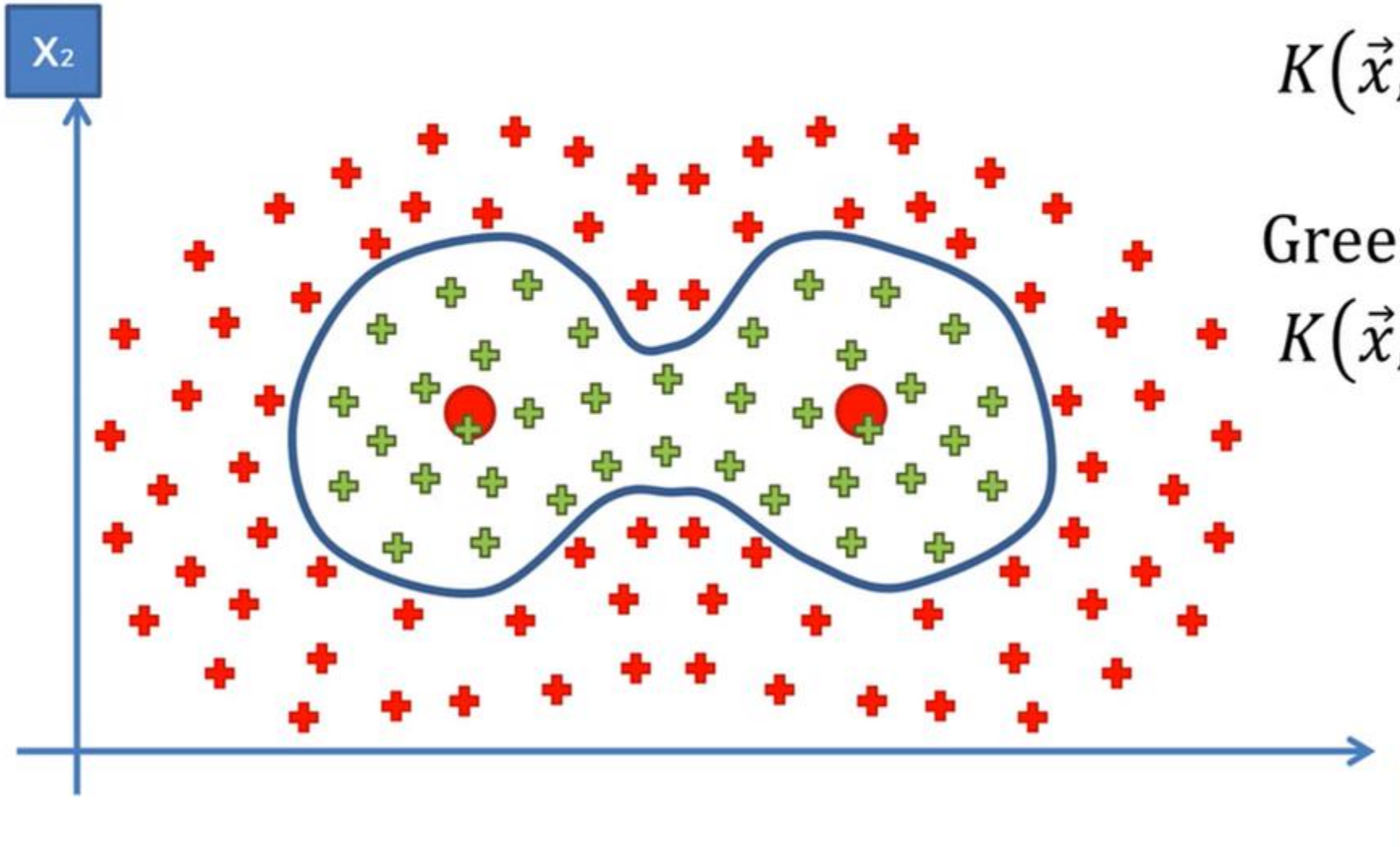
The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

(Simplified Formula)

The Gaussian RBF Kernel



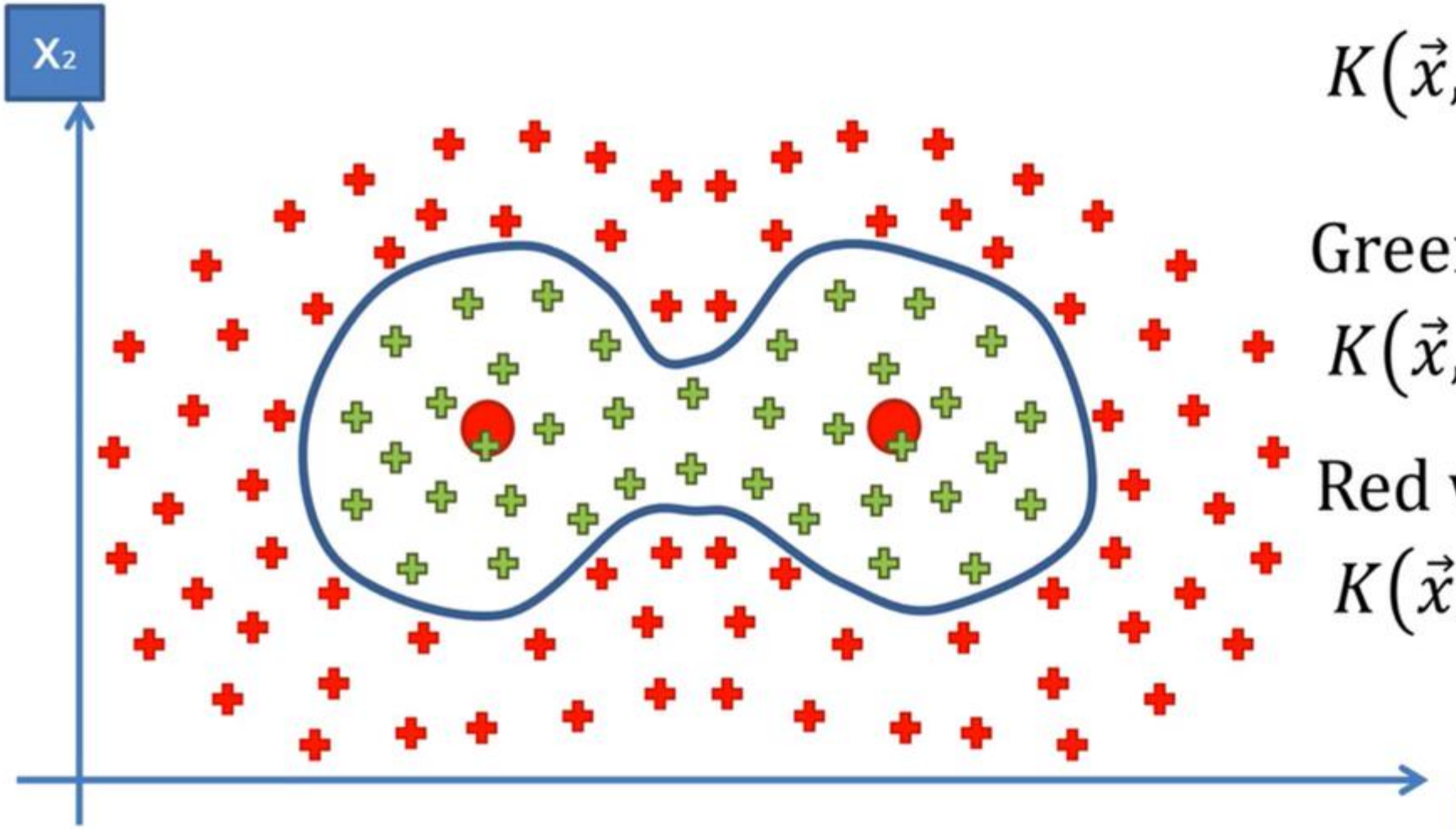
$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

(Simplified Formula)

Green when:

$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) > 0$$

The Gaussian RBF Kernel



$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2)$$

(Simplified Formula)

Green when:

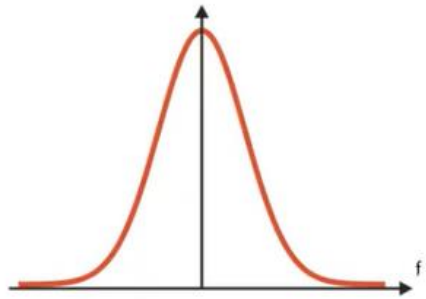
$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) > 0$$

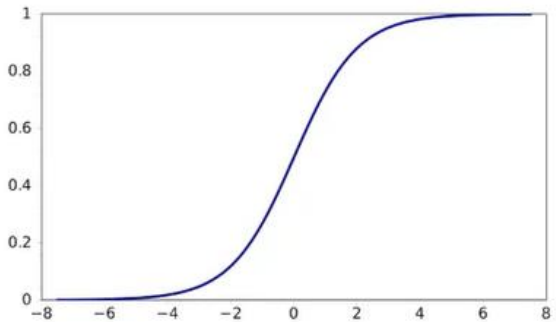
Red when:

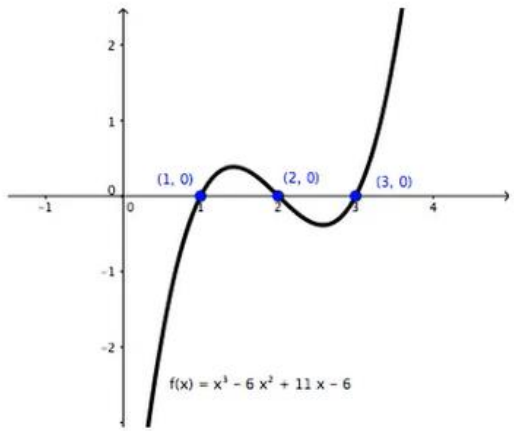
$$K(\vec{x}, \vec{l}^1) + K(\vec{x}, \vec{l}^2) = 0$$

$$x = (1, 2) \& \; l = (2, 3) \& \; \sigma = 1$$

Types of Kernel Functions







Gaussian RBF Kernel

$$K\left(\vec{x}, \vec{l}^{i}\right) = e^{-\frac{\left\|\vec{x}-\vec{l}^{i}\right\|^{2}}{2\sigma^{2}}}$$

Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^{T}Y + r)$$

Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^{T}Y + r)^{d}, \gamma > 0$$

SVM

Nonlinear Example

Implementation of Kernel SVM in Python

Python Code

```
# Data Preprocessing

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

	User ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	0
2	15810944	Male	35	20000	0
3	15668575	Female	26	43000	0
4	15603246	Female	27	57000	0
5	15804002	Male	19	76000	0
6	15728773	Male	27	58000	0
7	15598044	Female	27	84000	0
8	15694829	Female	32	150000	1
9	15600575	Male	25	33000	0
10	15727311	Female	35	65000	0
11	15570769	Female	26	80000	0
12	15606274	Female	26	52000	0

Showing 1 to 12 of 400 entries

	Age	EstimatedSalary	Purchased
1	19	19000	0
2	35	20000	0
3	26	43000	0
4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
9	25	33000	0
10	35	65000	0
11	26	80000	0
12	26	52000	0

Showing 1 to 12 of 400 entries

Python Code

```
# Data Preprocessing

# Splitting the dataset into the Training set and Test set

from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state = 0)
```

	Age	EstimatedSalary	Purchased
1	19	19000	0
3	26	43000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
10	35	65000	0
11	26	80000	0
13	20	86000	0
14	32	18000	0
15	18	82000	0
16	29	80000	0
17	47	25000	1

Showing 1 to 12 of 300 entries

	Age	EstimatedSalary	Purchased
2	35	20000	0
4	27	57000	0
5	19	76000	0
9	25	33000	0
12	26	52000	0
18	45	26000	1
19	46	28000	1
20	48	29000	1
22	47	49000	1
29	29	43000	0
32	27	137000	1
34	28	44000	0

Showing 1 to 12 of 100 entries

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

	Age	EstimatedSalary	Purchased
1	-1.76554750	-1.47334137	0
3	-1.09629664	-0.78837605	0
6	-1.00068938	-0.36027273	0
7	-1.00068938	0.38177303	0
8	-0.52265305	2.26542765	1
10	-0.23583125	-0.16049118	0
11	-1.09629664	0.26761214	0
13	-1.66994024	0.43885347	0
14	-0.52265305	-1.50188159	0
15	-1.86115477	0.32469259	0
16	-0.80947485	0.26761214	0
17	0.91145593	-1.30210004	1

Showing 1 to 12 of 300 entries

	Age	EstimatedSalary	Purchased
2	-0.30419063	-1.51354339	0
4	-1.05994374	-0.32456026	0
5	-1.81569686	0.28599864	0
9	-1.24888202	-1.09579256	0
12	-1.15441288	-0.48523366	0
18	0.64050076	-1.32073531	1
19	0.73496990	-1.25646596	1
20	0.92390818	-1.22433128	1
22	0.82943904	-0.58163769	1
29	-0.87100546	-0.77444577	0
32	-1.05994374	2.24621408	1
34	-0.96547460	-0.74231109	0

Showing 1 to 12 of 100 entries

```
# Fitting SVM to the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train,  y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Applications of Support Vector Machine

Face detection

Text and hypertext categorization

Classification of images

Bioinformatics


Thank you