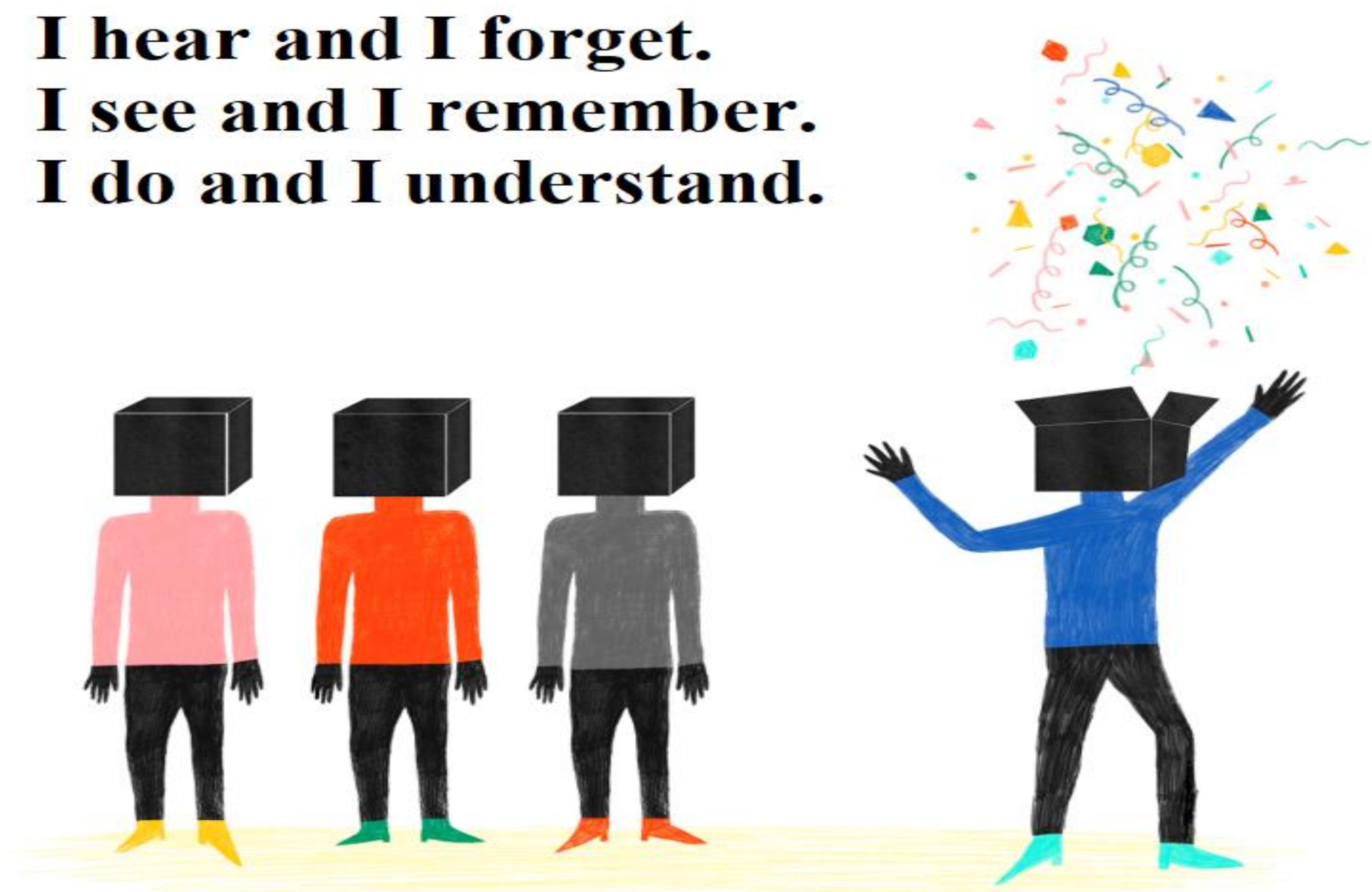
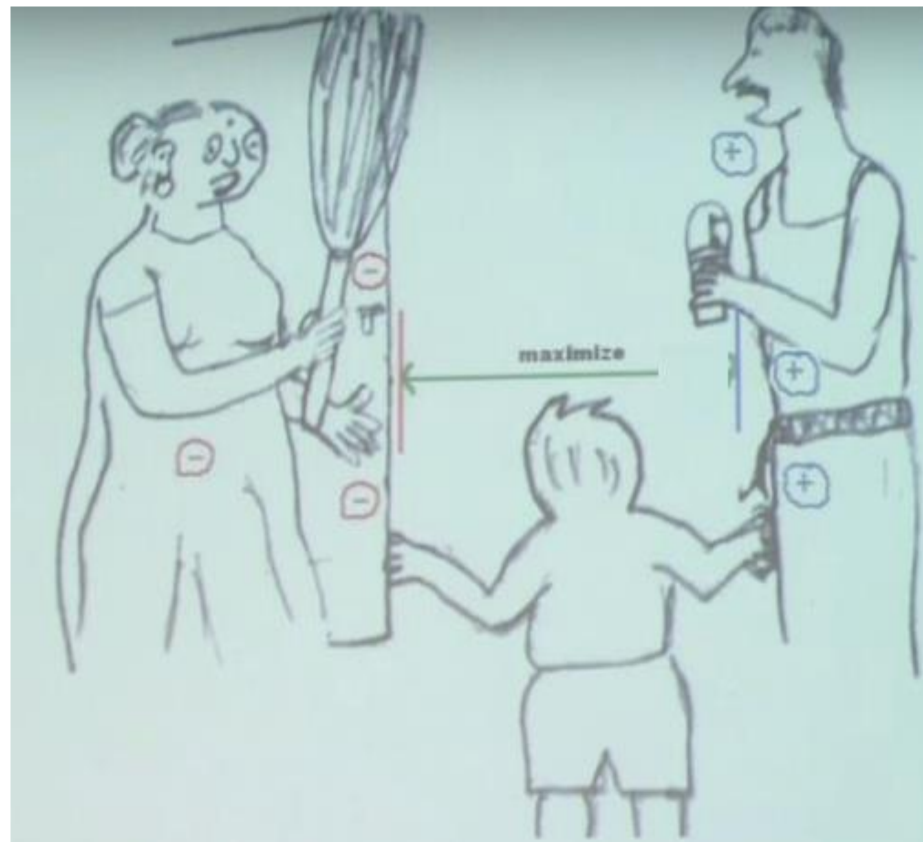


Chandan Verma

Corporate Trainer(Machine Learning,AI,Cloud Computing,IOT)

www.facebook.com/verma.chandan.070


maximize

Support Vector Machines

SVM is a supervised learning method

In this algorithm, a hyperplane is going to be selected that best separates the points in the space.

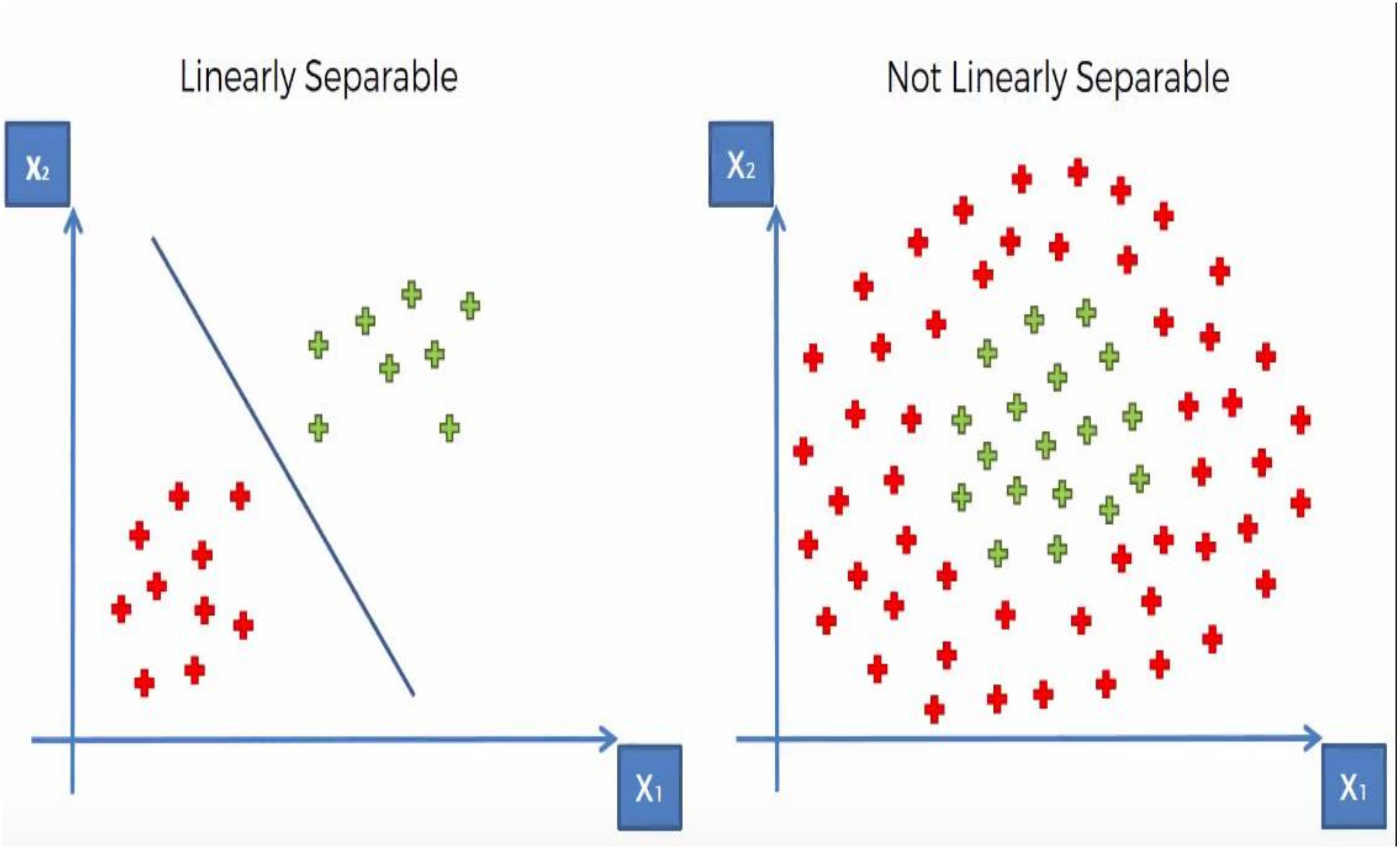
Like other classification algorithms, such as logistic regression or random forest, SVMs attempt to create a mathematical formula capable of dividing the data into the two categories.

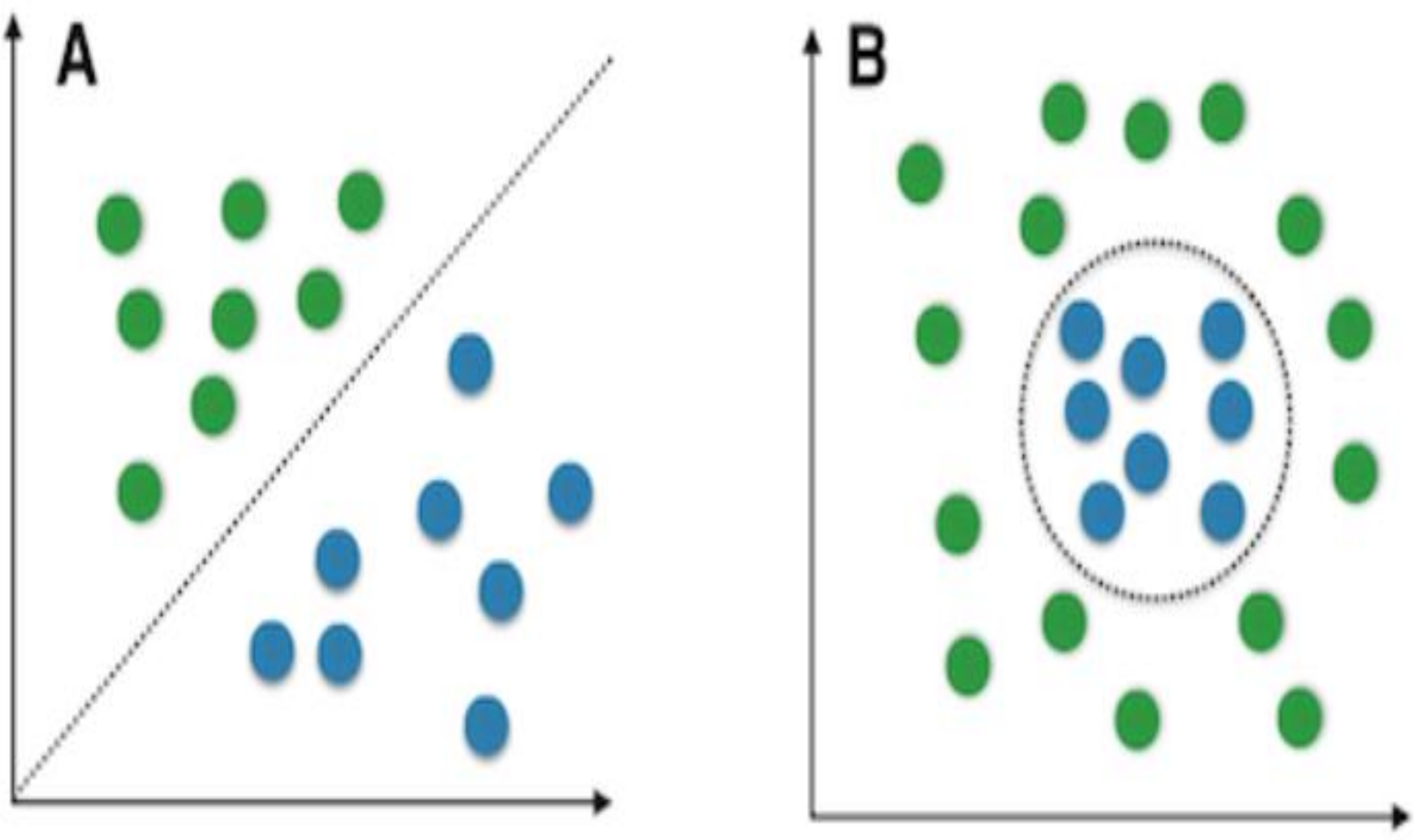
- in one dimension, is called a point
- in two dimensions, it is a line
- in three dimensions, it is a plane
- in more dimensions you can call it an hyperplane

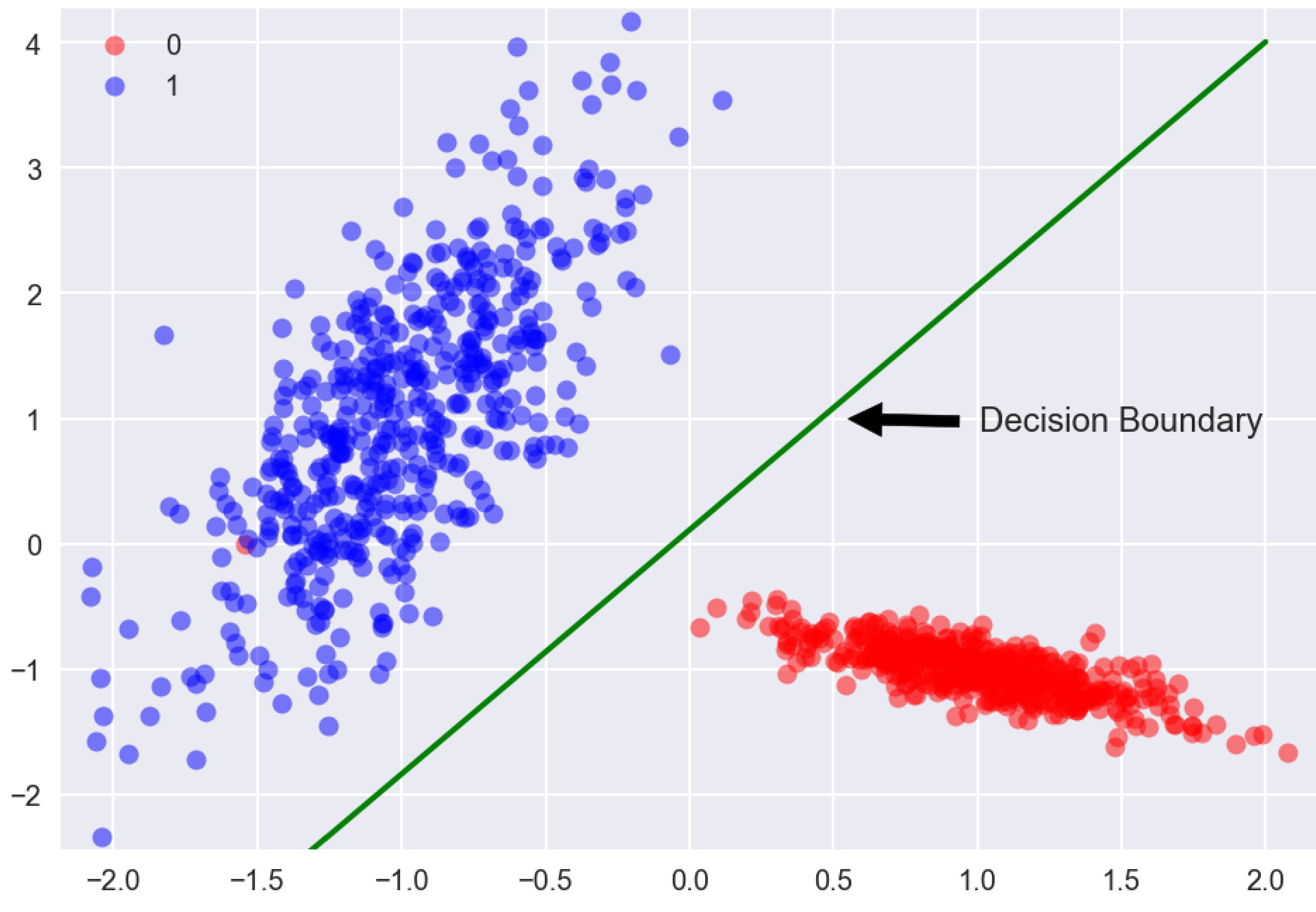
- **Separable case** – Infinite boundaries are possible to separate the data into two classes.
- **Non Separable case** – Two classes are not separated but overlap with each other.

support vector machine can work with any number of dimensions !

LINEAR SEPARABLE AND NON LINEAR SEPARABLE

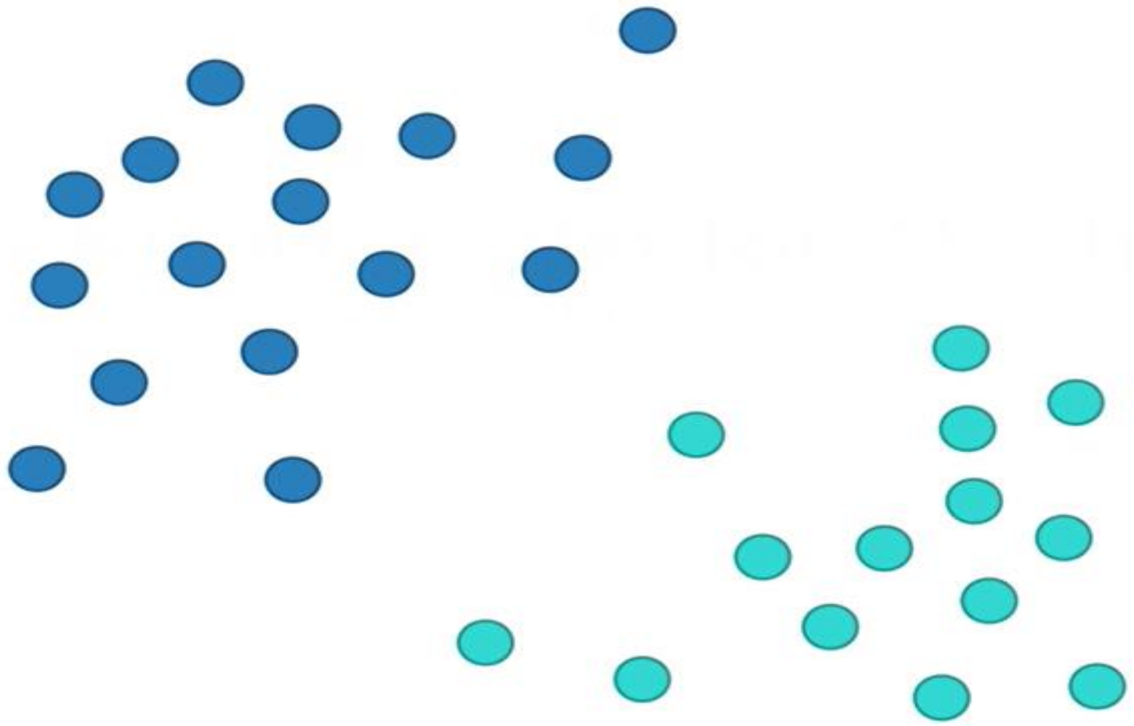


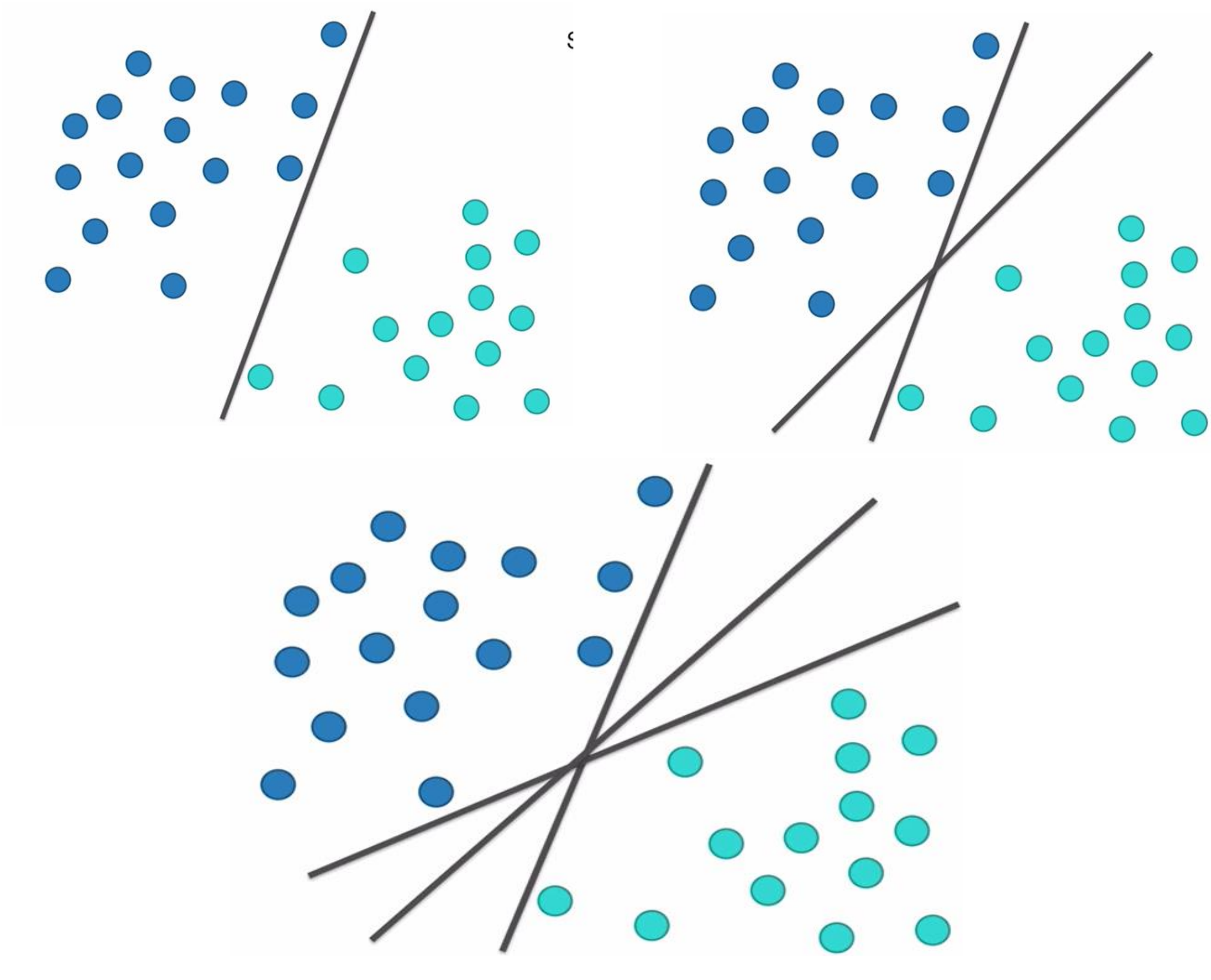

A
B



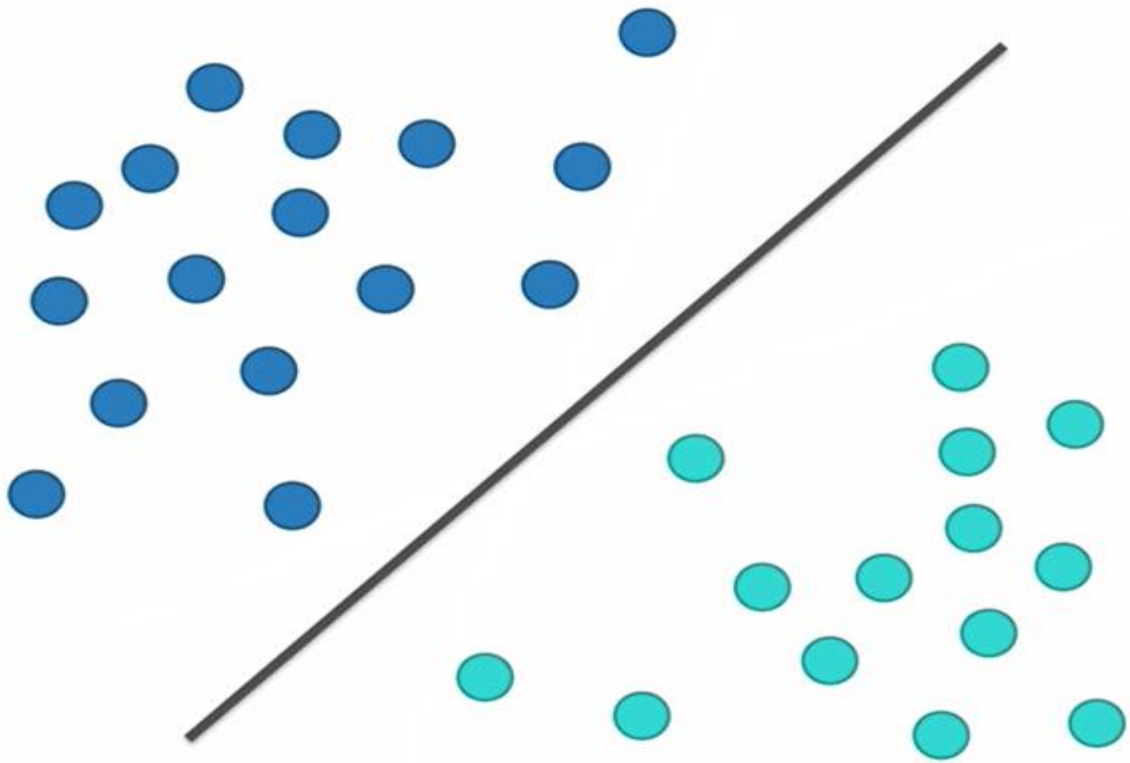
0
1
Decision Boundary
4
3
2
1
0
−1
−2
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Split Data

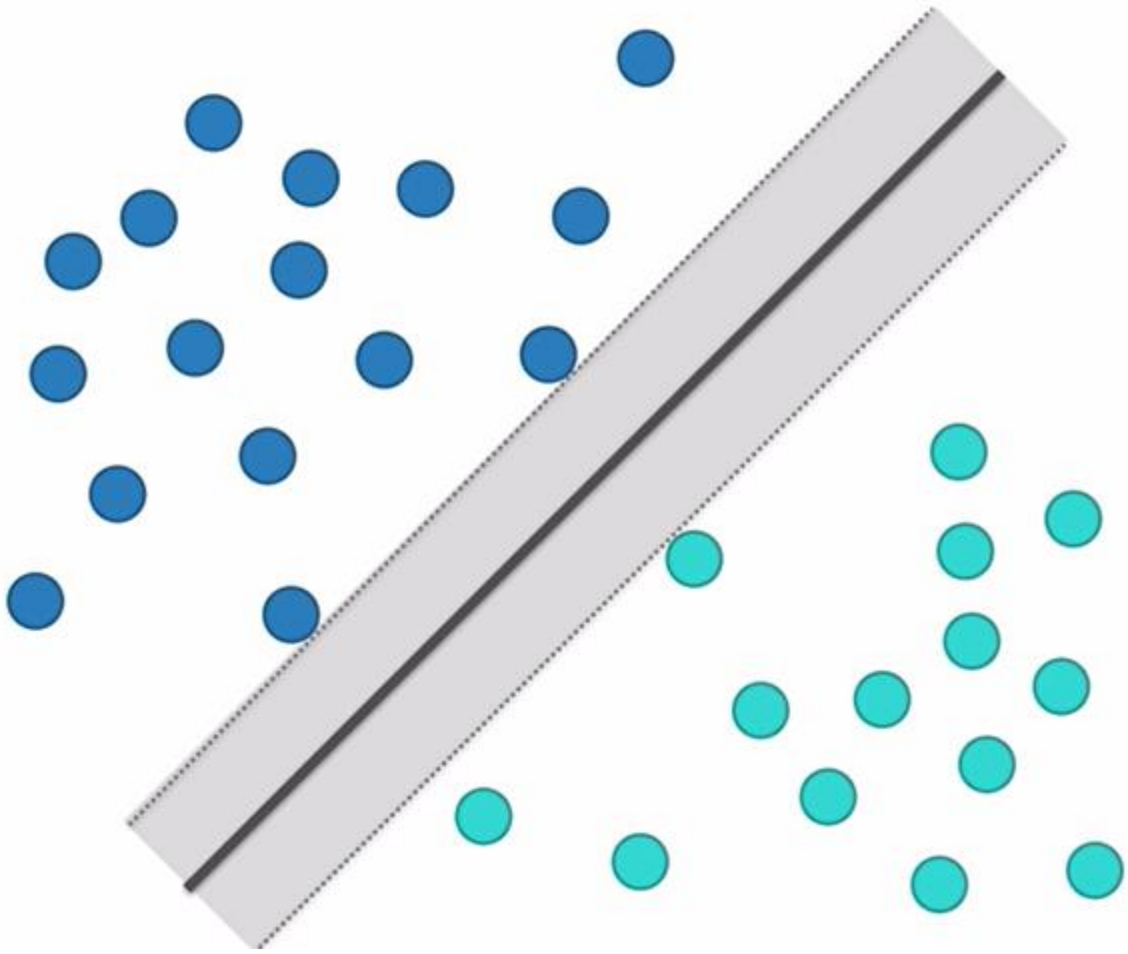


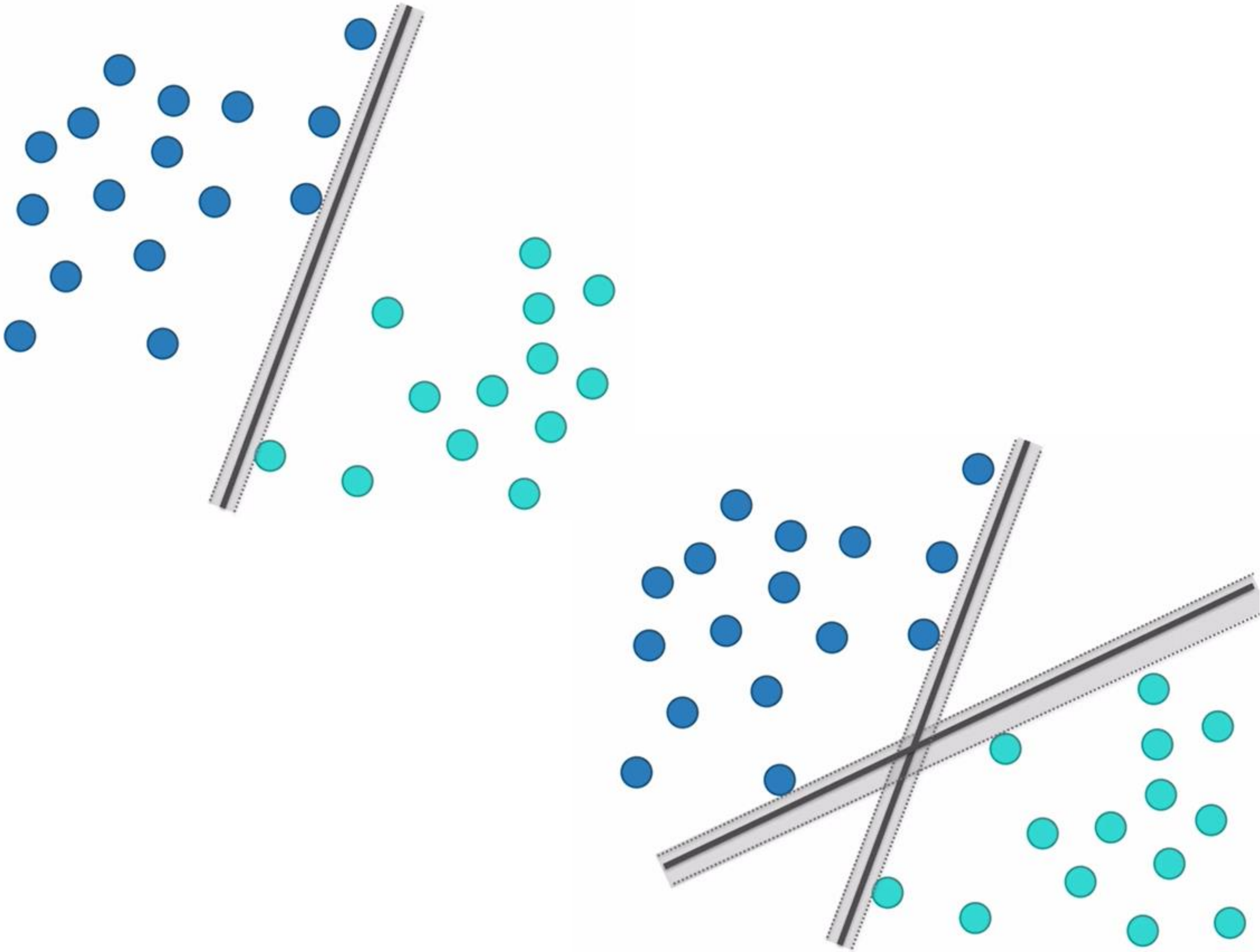


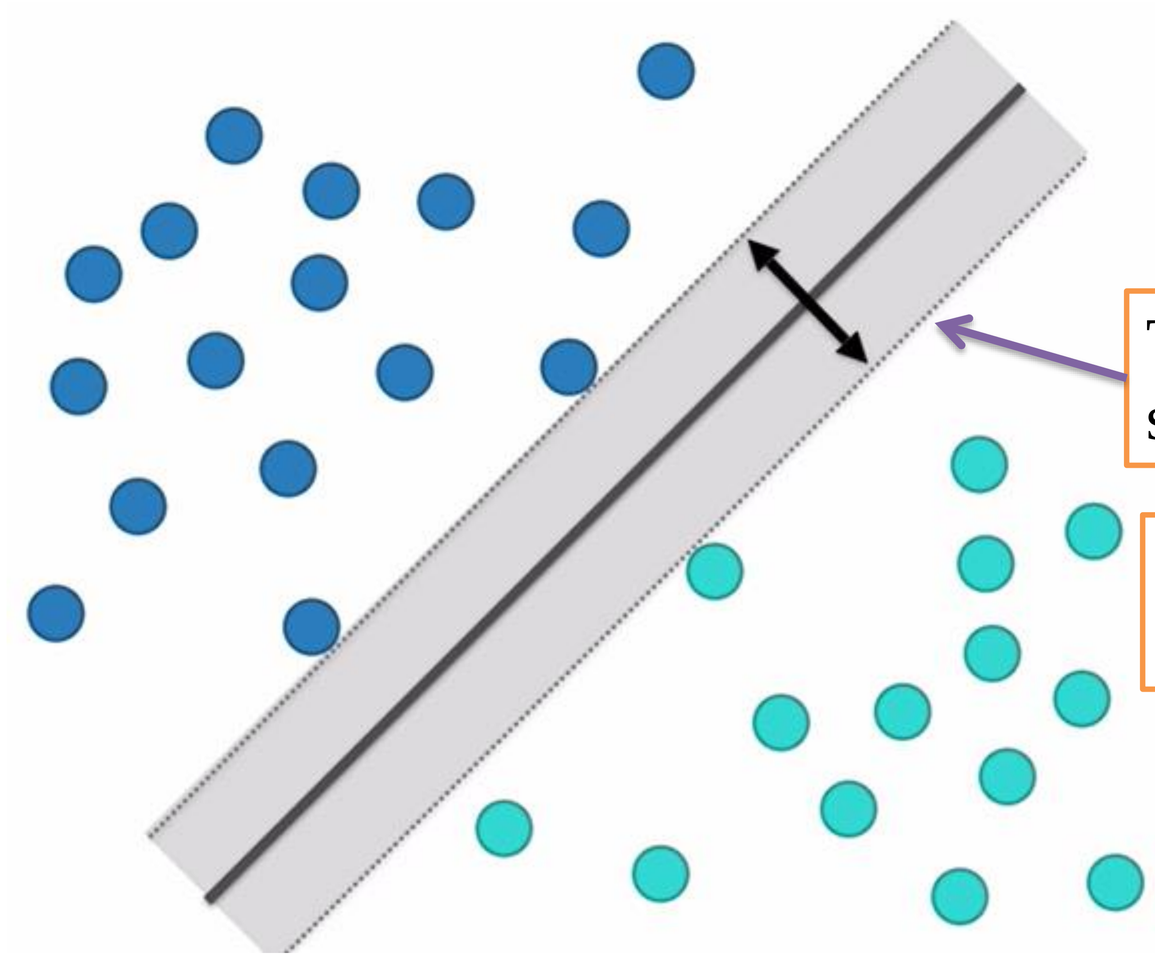
Split the Data Best Possible Way



Why Best Split

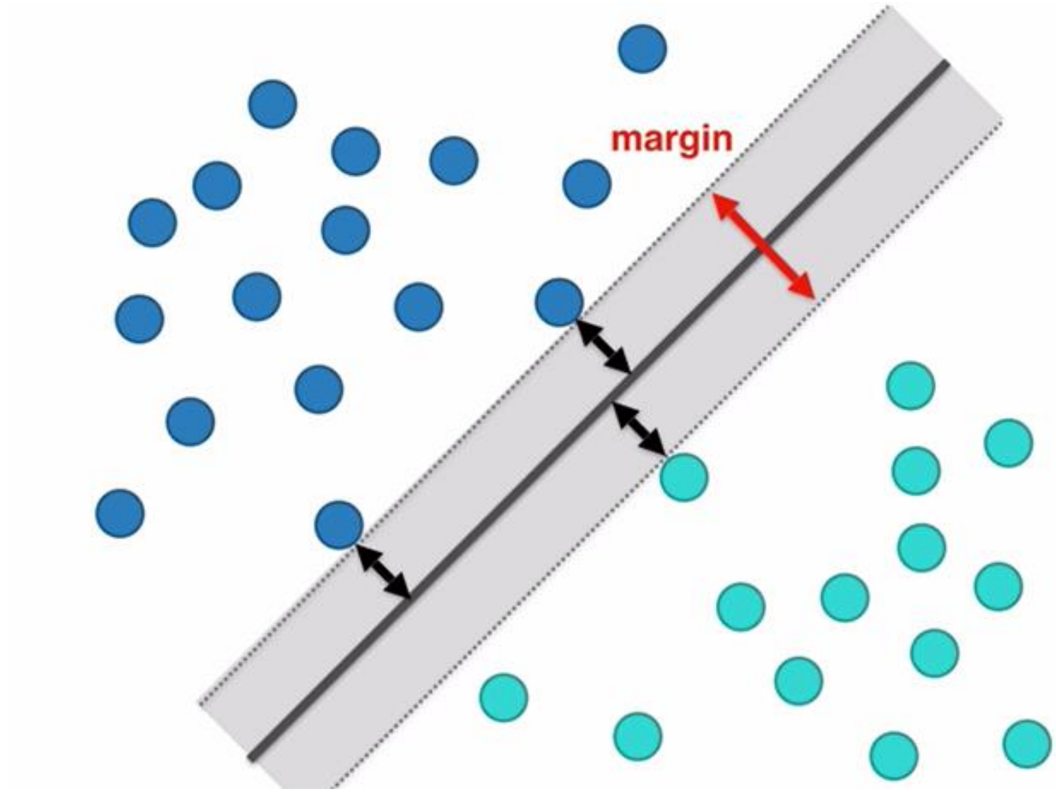




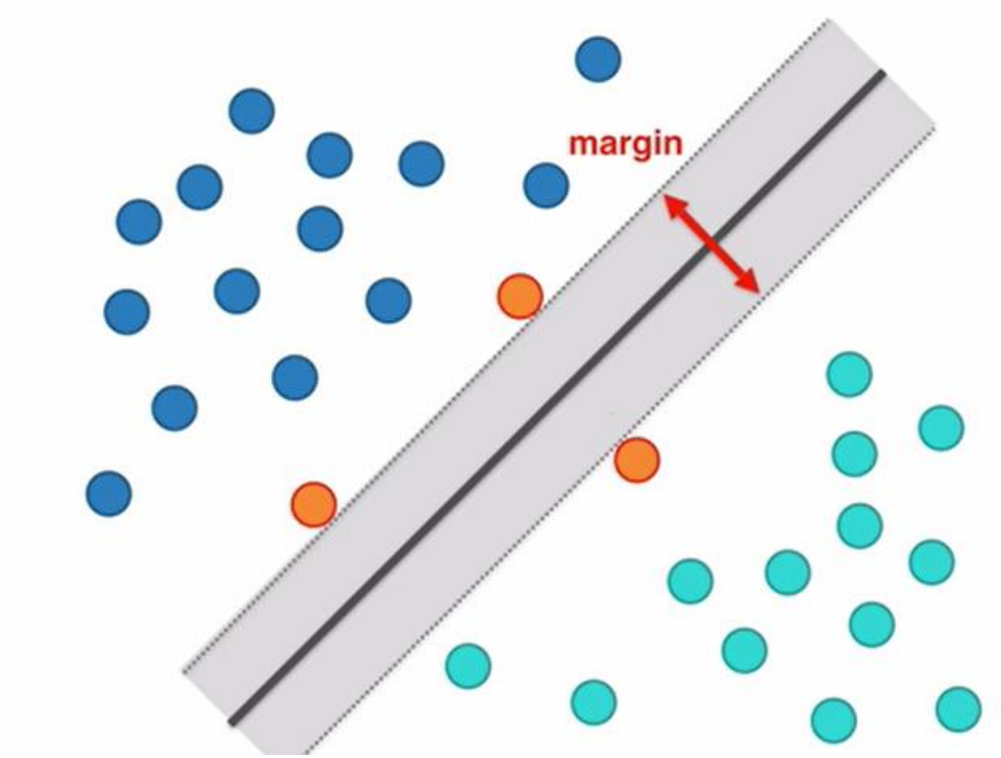


This is wider road that separates the two group

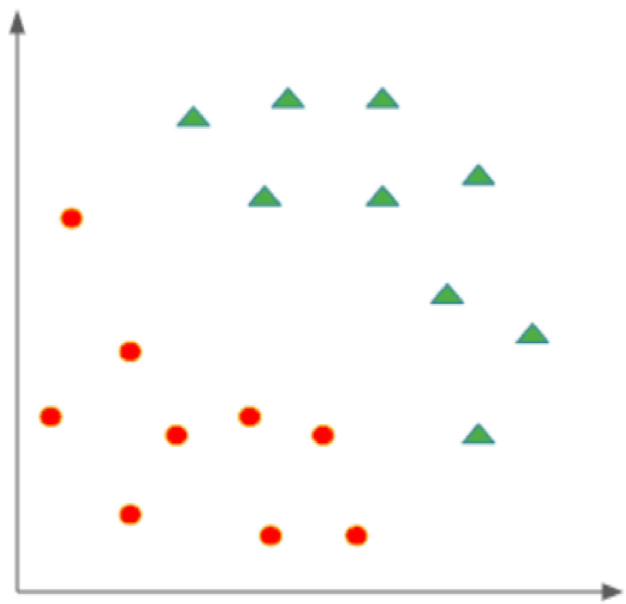
This is wider Margin that separates the two group

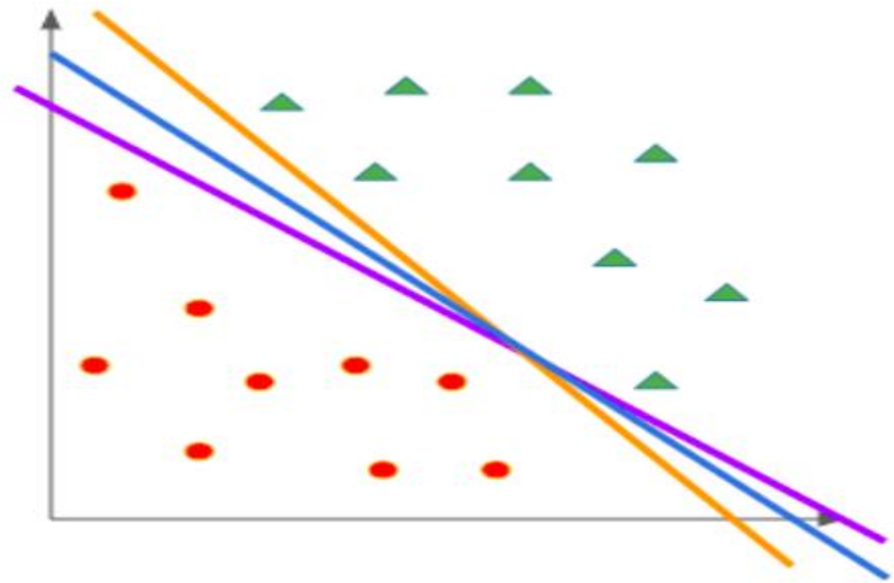


The distance between the point and the lines are as far as possible



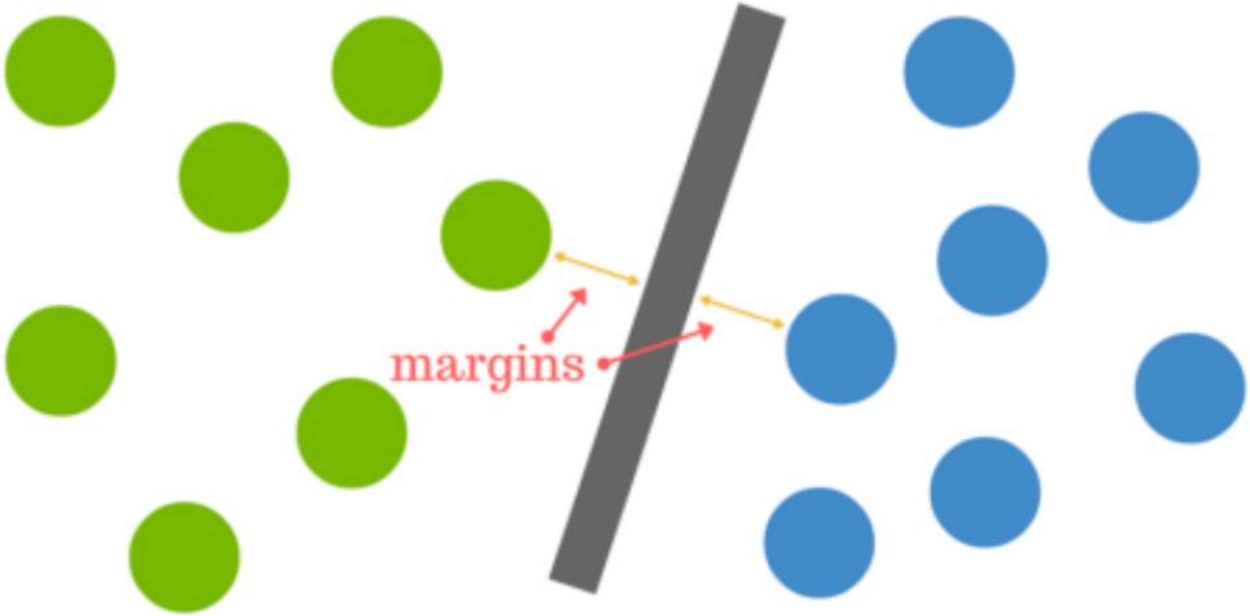
The distance between the support vector and the hyperplane are as far as possible





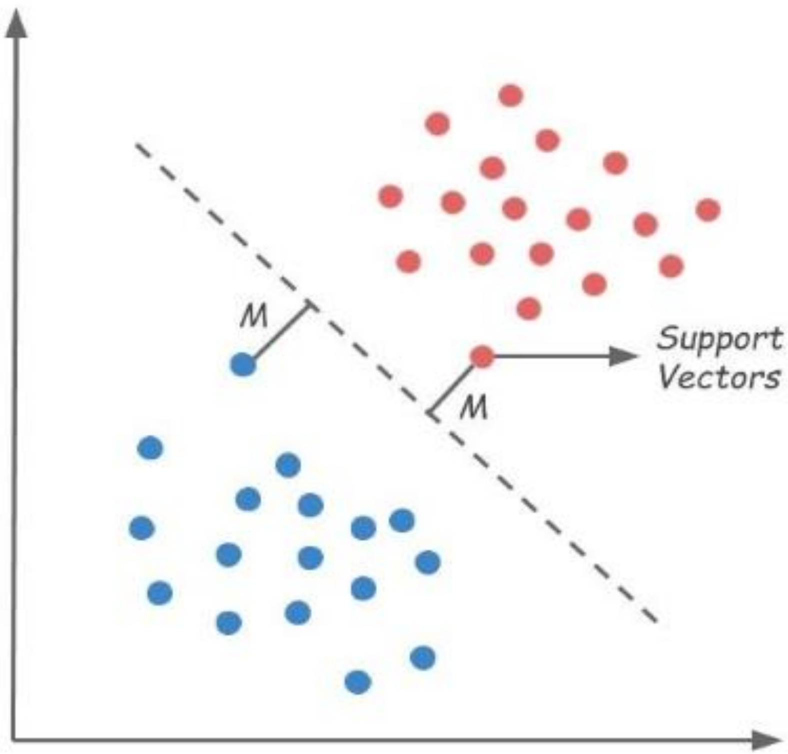
How do we choose the optimal separating line?

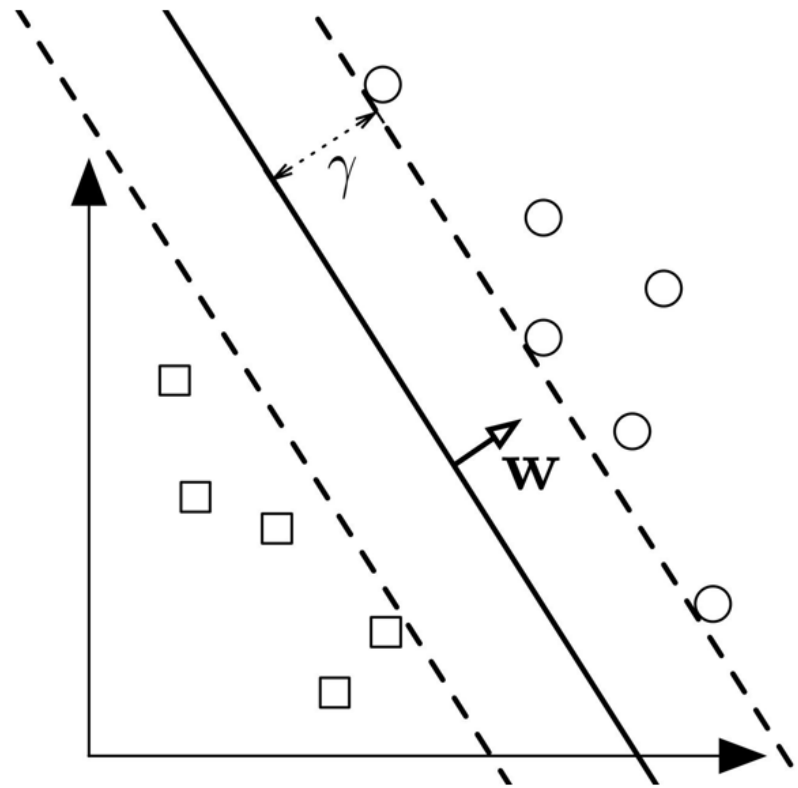
SVMs find the best line that separates the data into categories by maximizing the orthogonal distance between the nearest points of each category



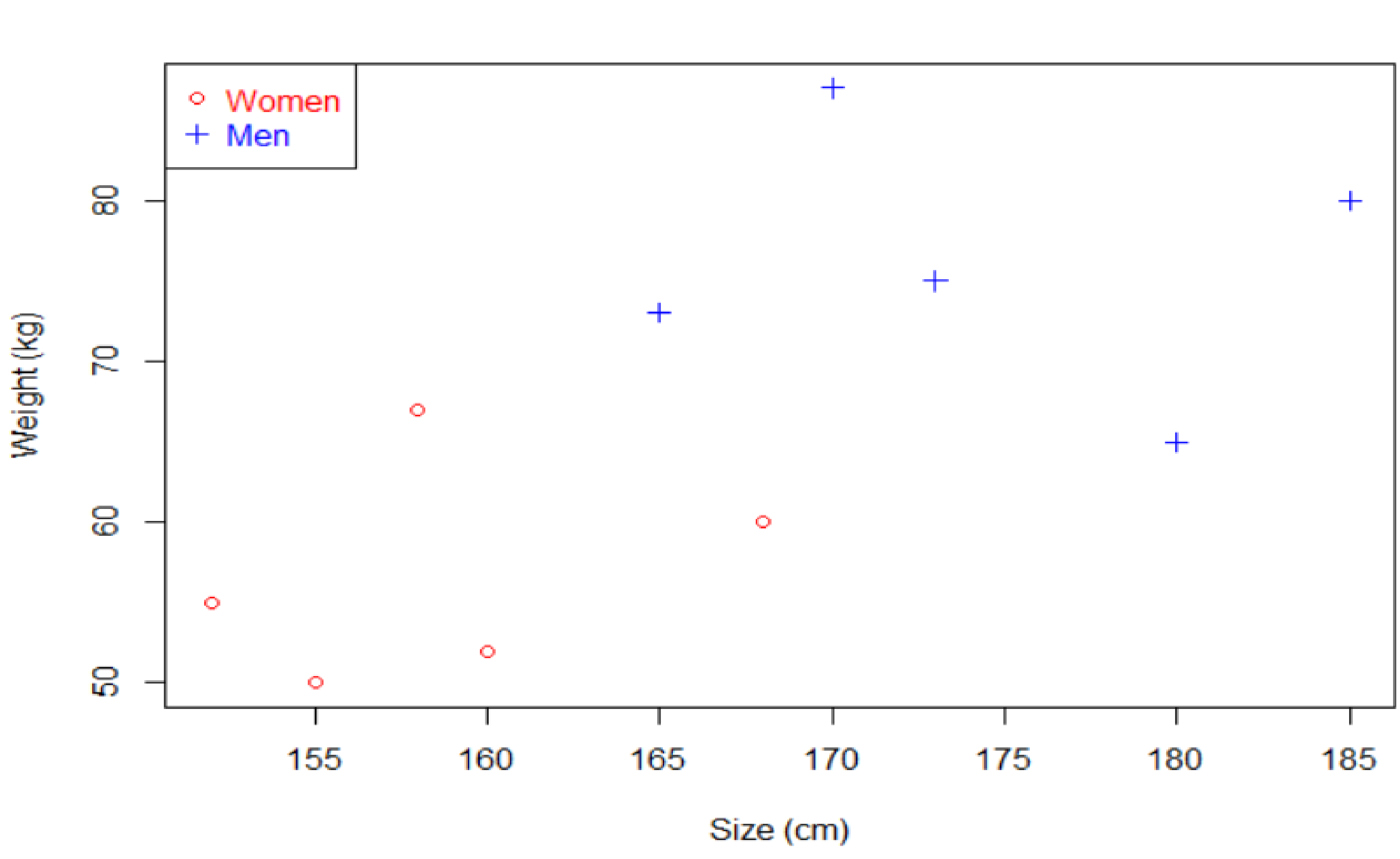
The distance between the hyperplane and the nearest data point from either set is known as the margin.

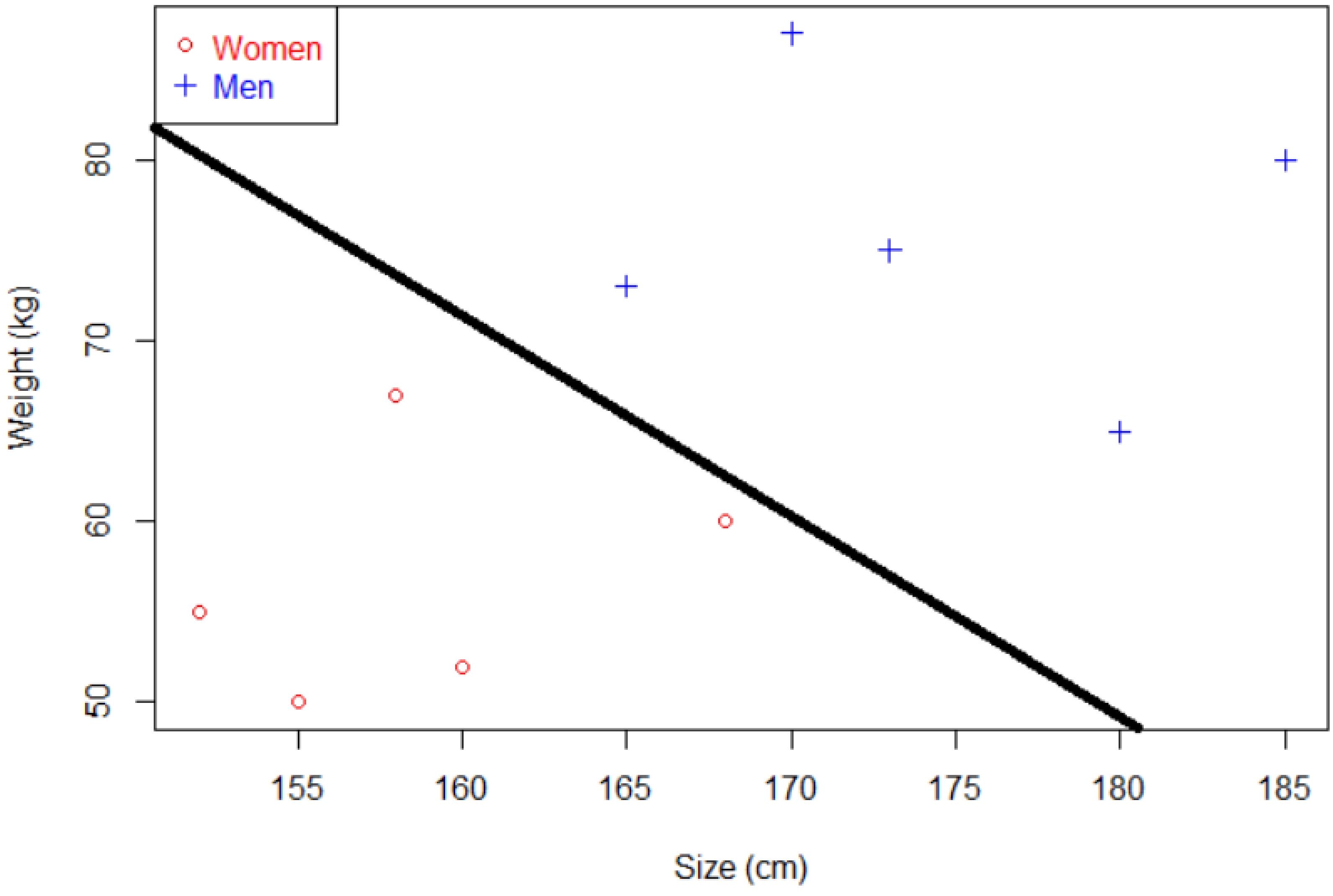
The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.


M
M
Support
Vectors


γ
$\mathbf{w}$

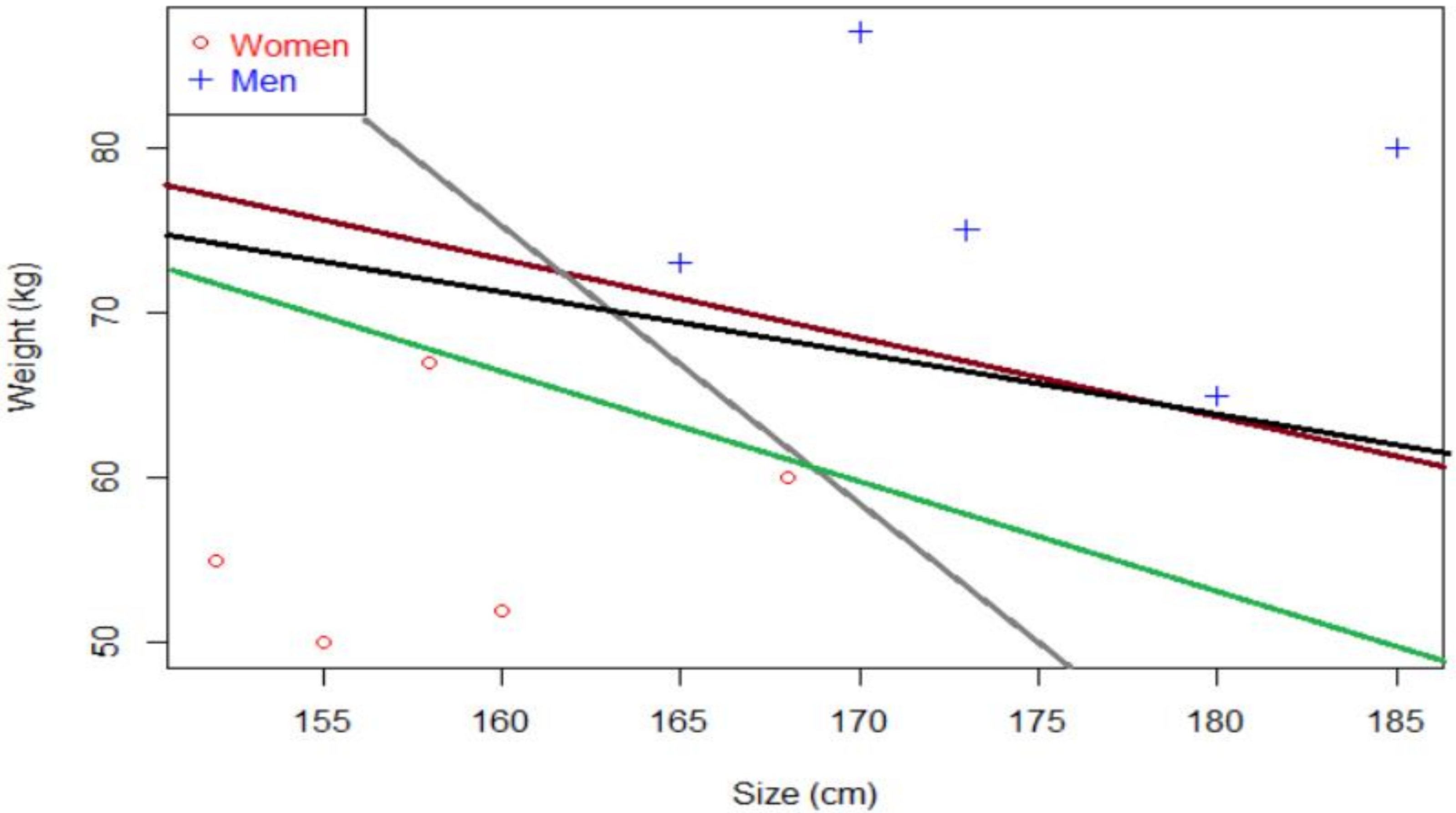
Separating hyperplane?

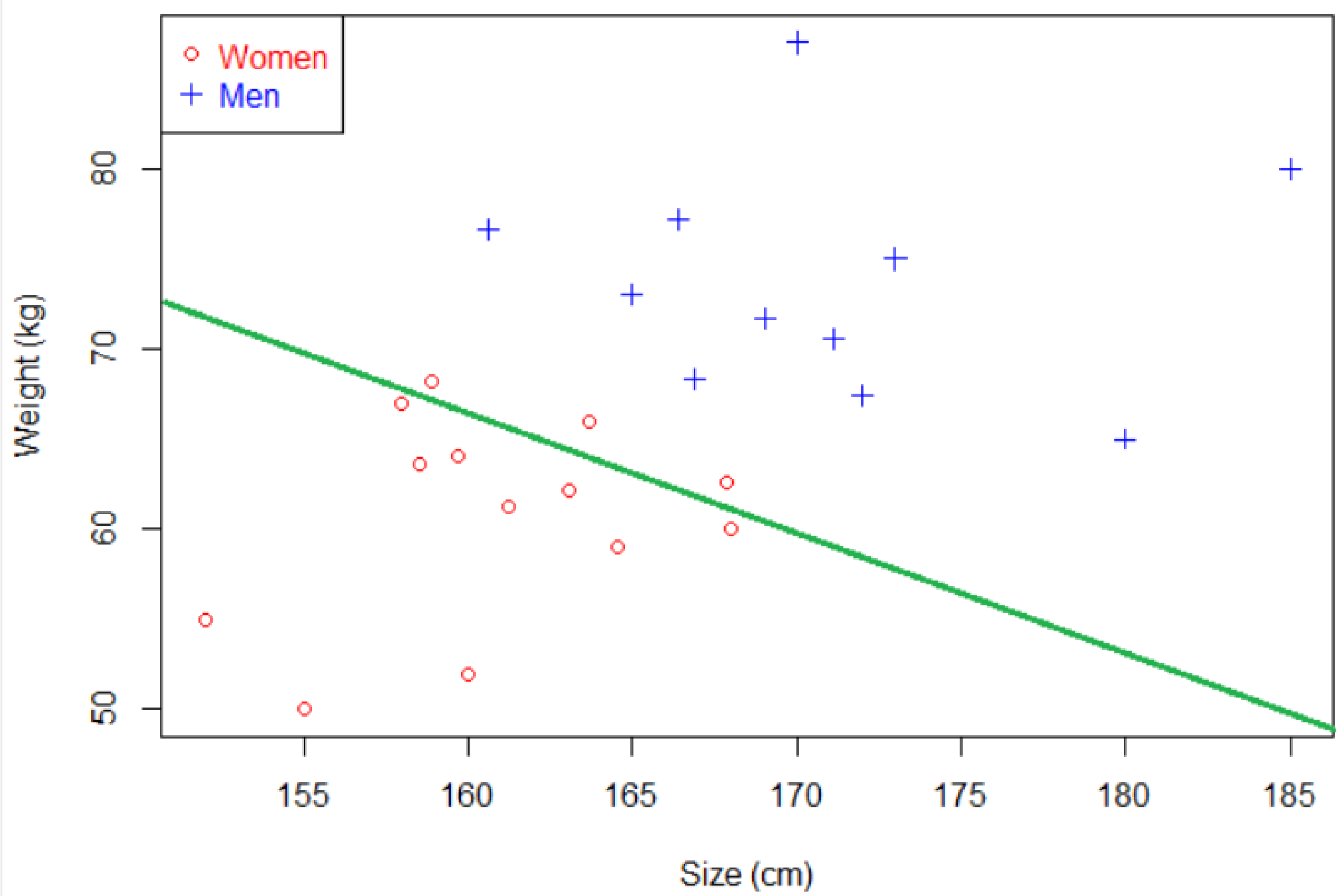




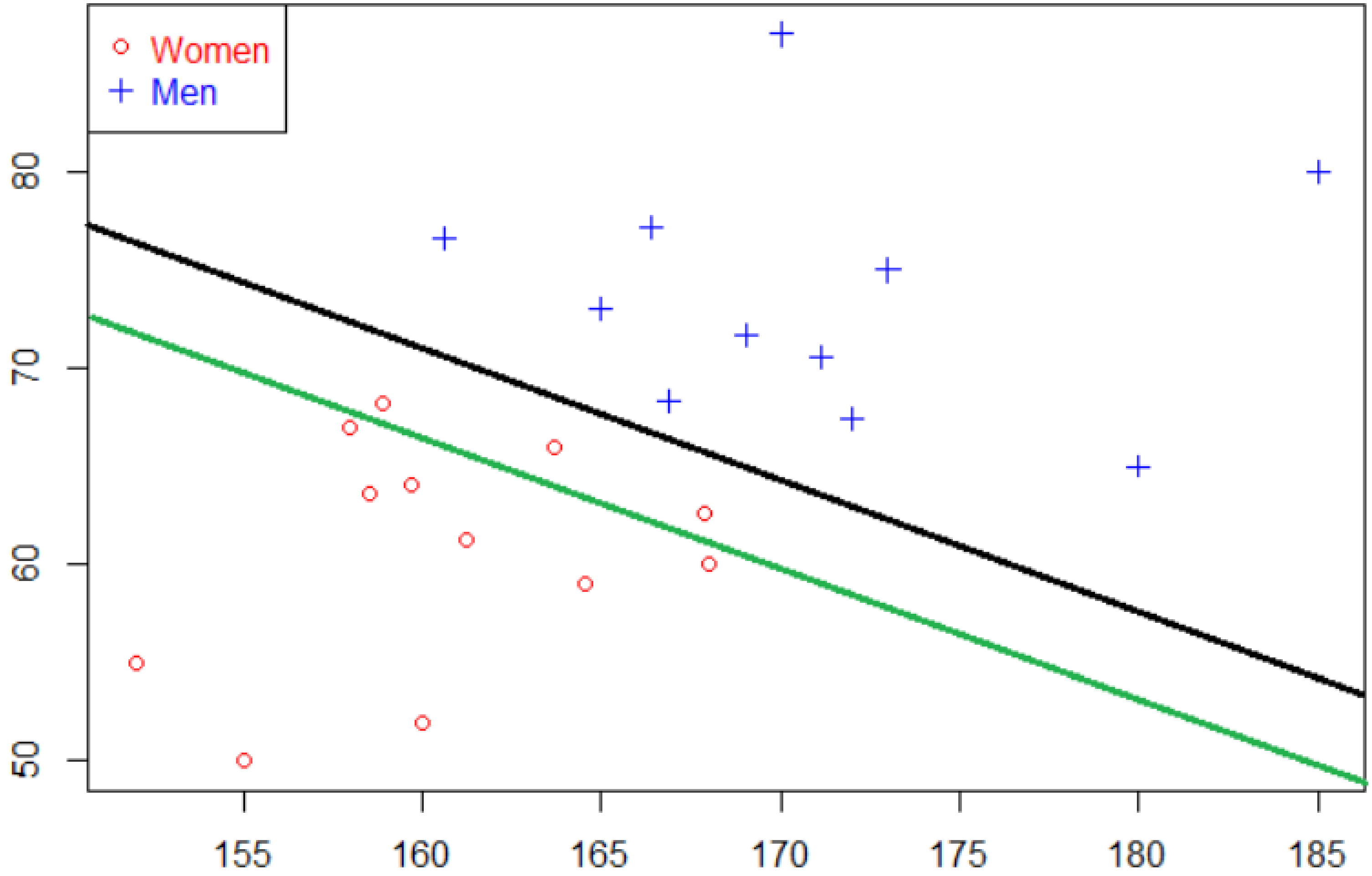
Women
Men
Weight (kg)
50
60
70
80
155
160
165
170
175
180
185
Size (cm)

optimal separating hyperplane



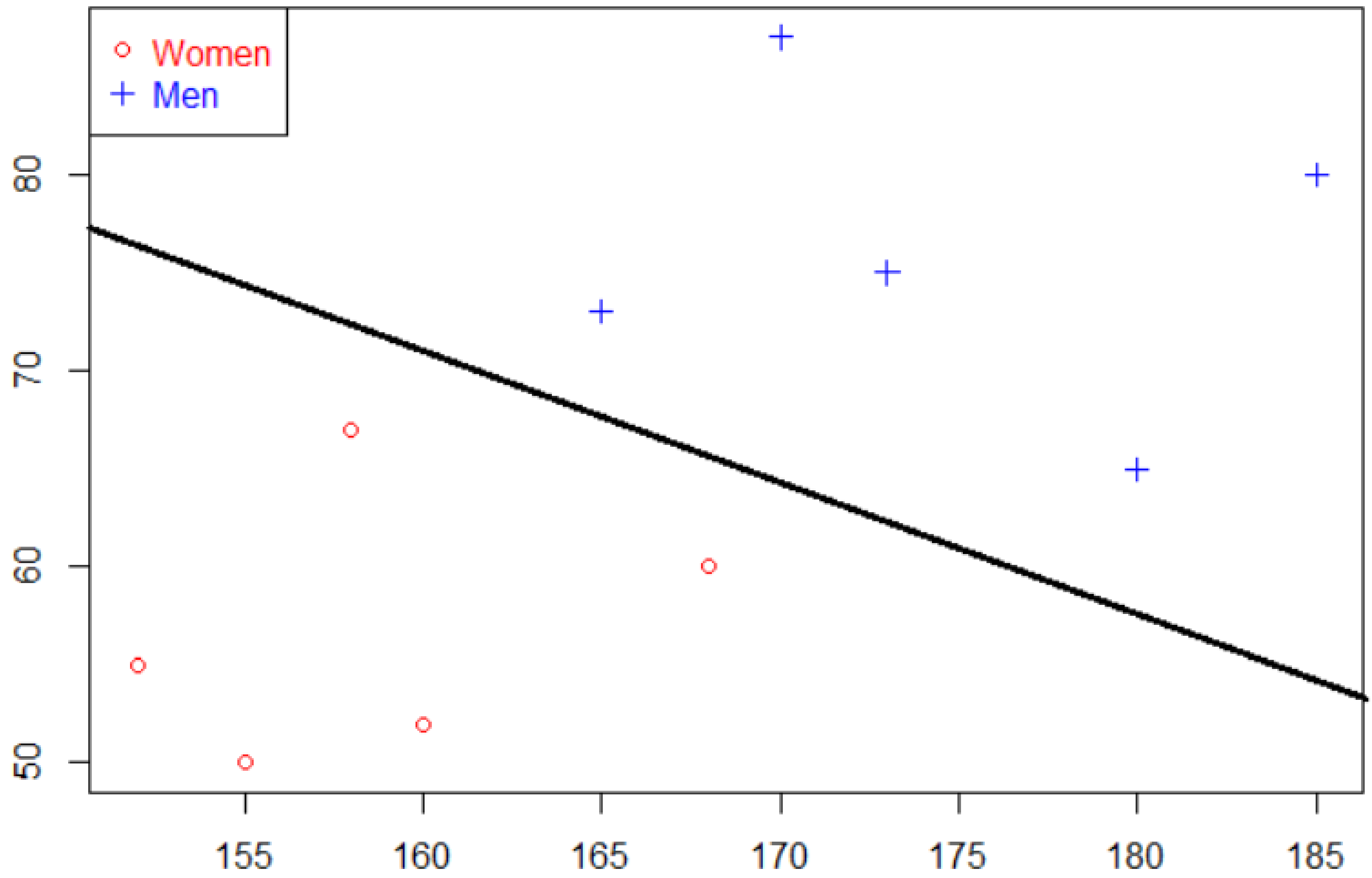


This hyperplane does not generalize well



The black hyperplane classifies more accurately than the green one

hyperplane as far as possible from data points from each category

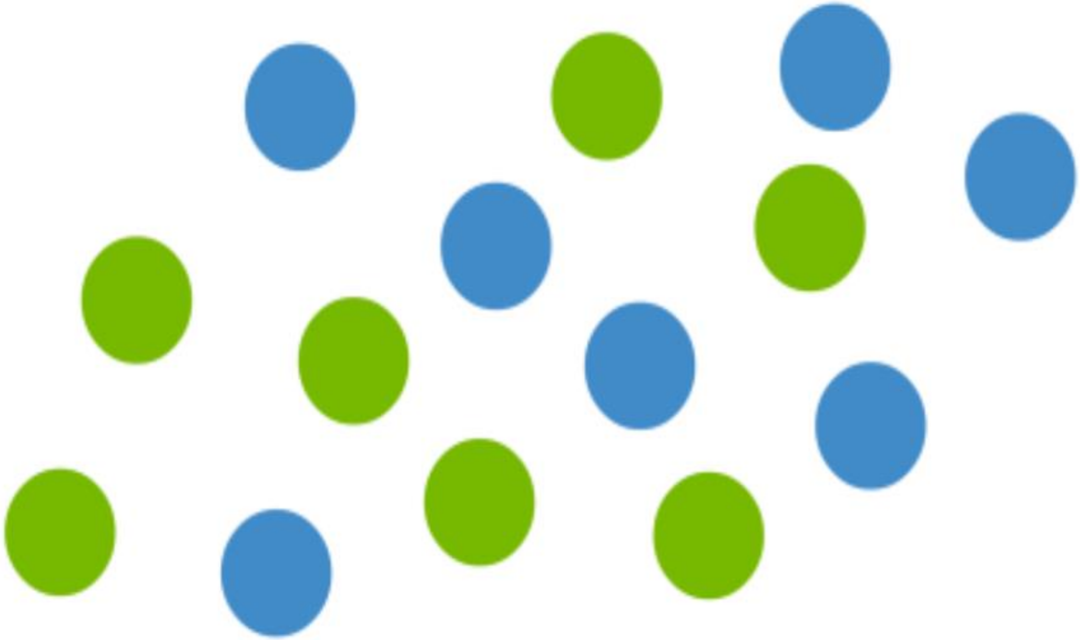


Support Vector Machines

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems.

SVMs are based on the idea of finding a **hyperplane** that best divides a dataset into two classes.

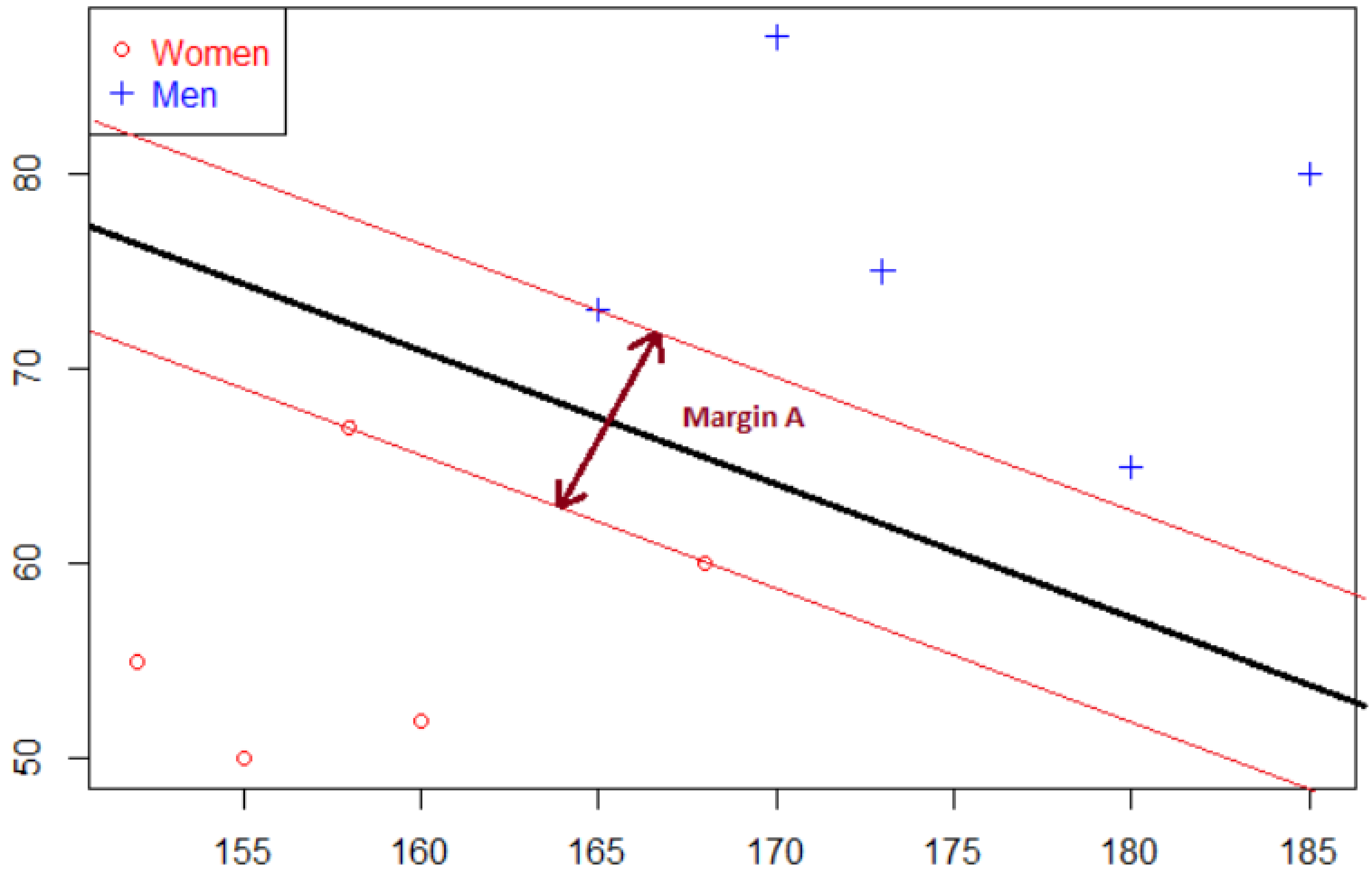
Hyperplane as a line that linearly separates and classifies a set of data.



In order to classify a dataset like the one above it's necessary to move away from a 2d view of the data to a 3d view.

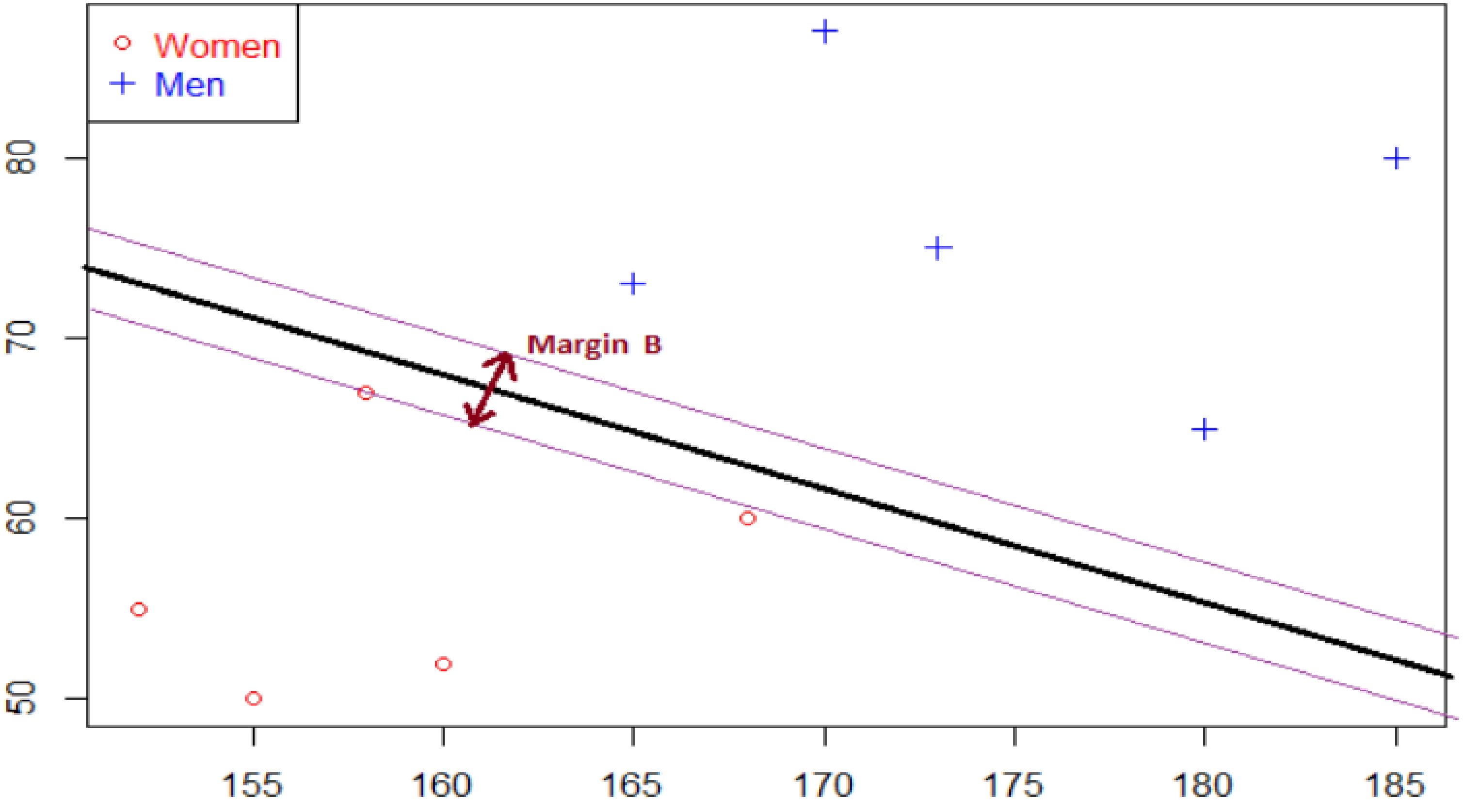
objective of a SVM is to **find the optimal separating hyperplane**:

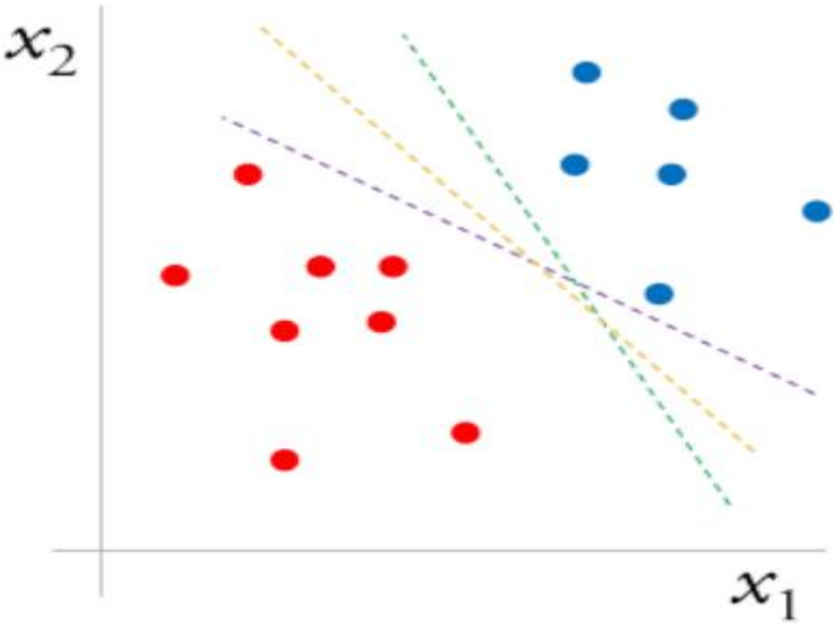
- because it correctly classifies the training data
- and because it is the one which will generalize better with unseen data



Basically the margin is a no man's land. There will never be any data point inside the margin.

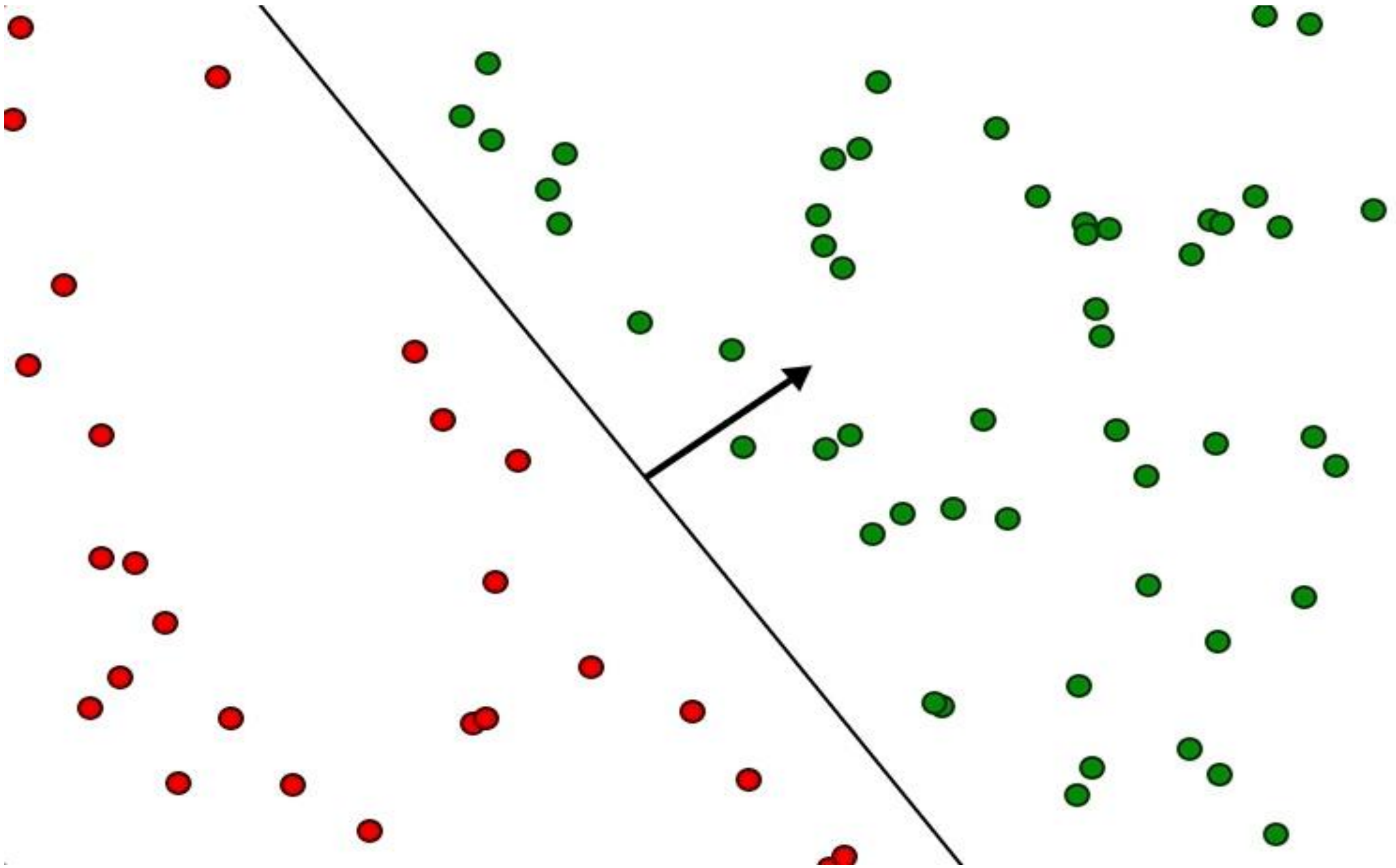
The optimal hyperplane will be the one with the biggest margin. **objective of the SVM** is to find the optimal separating hyperplane which maximizes the margin of the training data.

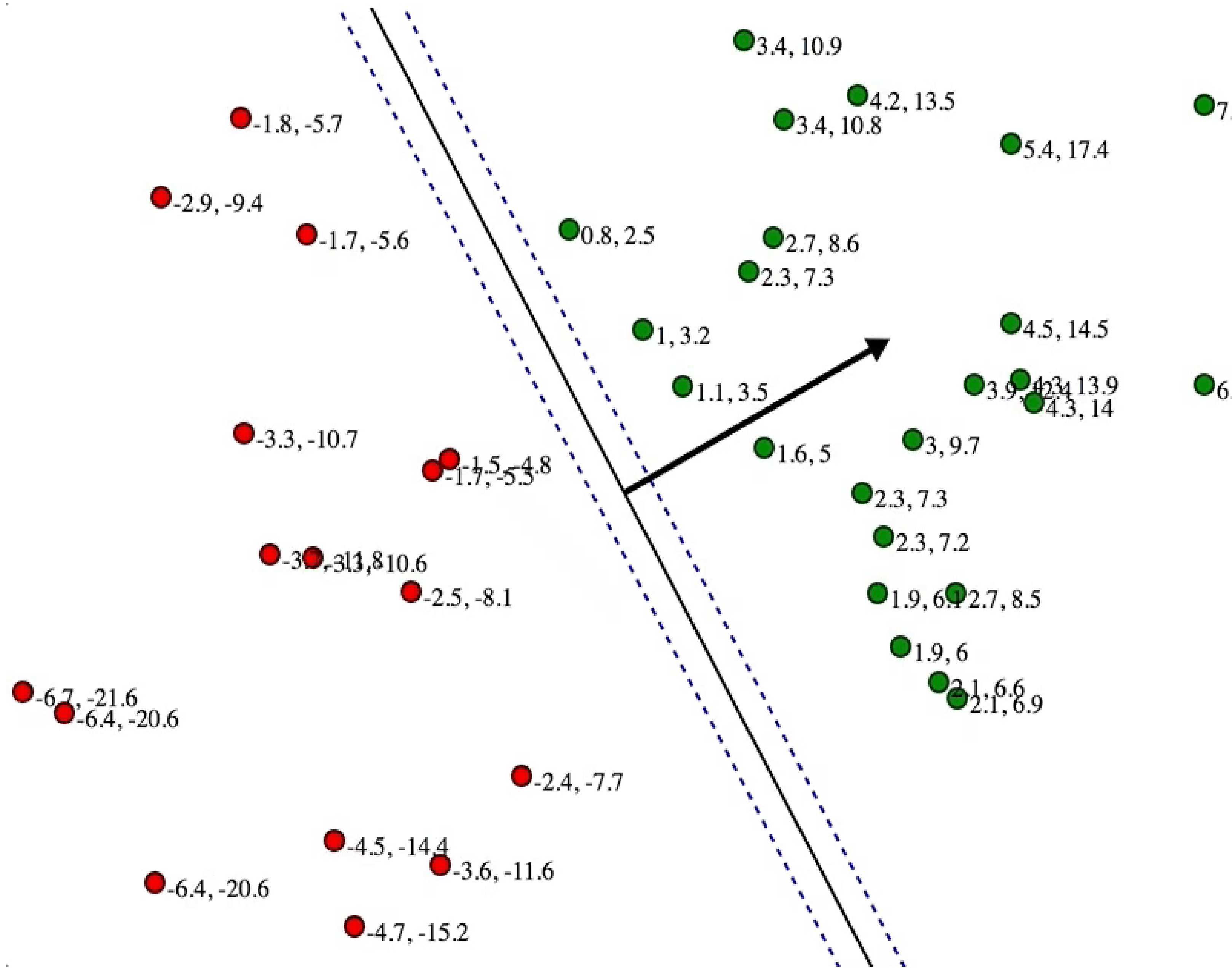




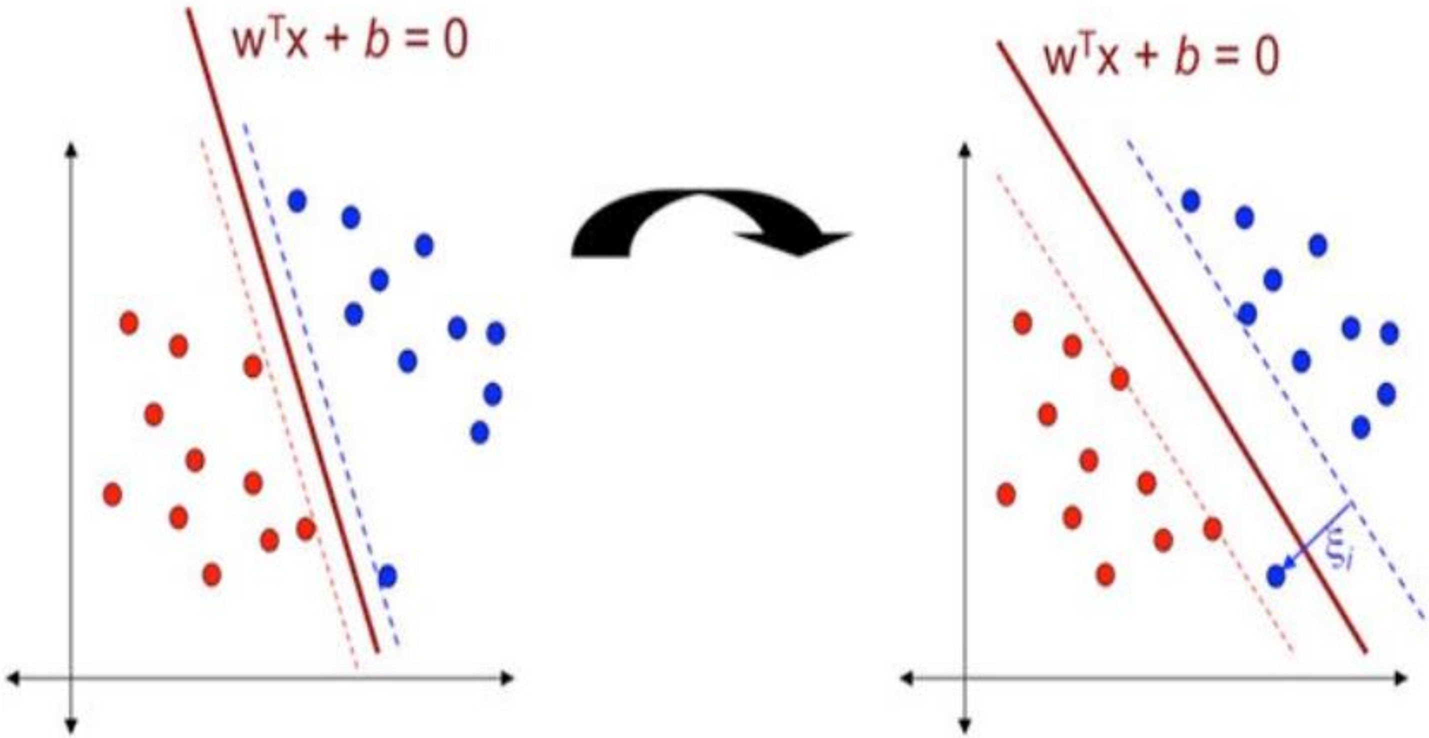
There are many possible decision boundaries that would perfectly separate the two classes, but an SVM will choose the line in 2-d (or “hyperplane”, more generally) that maximizes the margin around the boundary.

Intuitively, we can be very confident about the labels of points that fall far from the boundary, but we're less confident about points near the boundary.

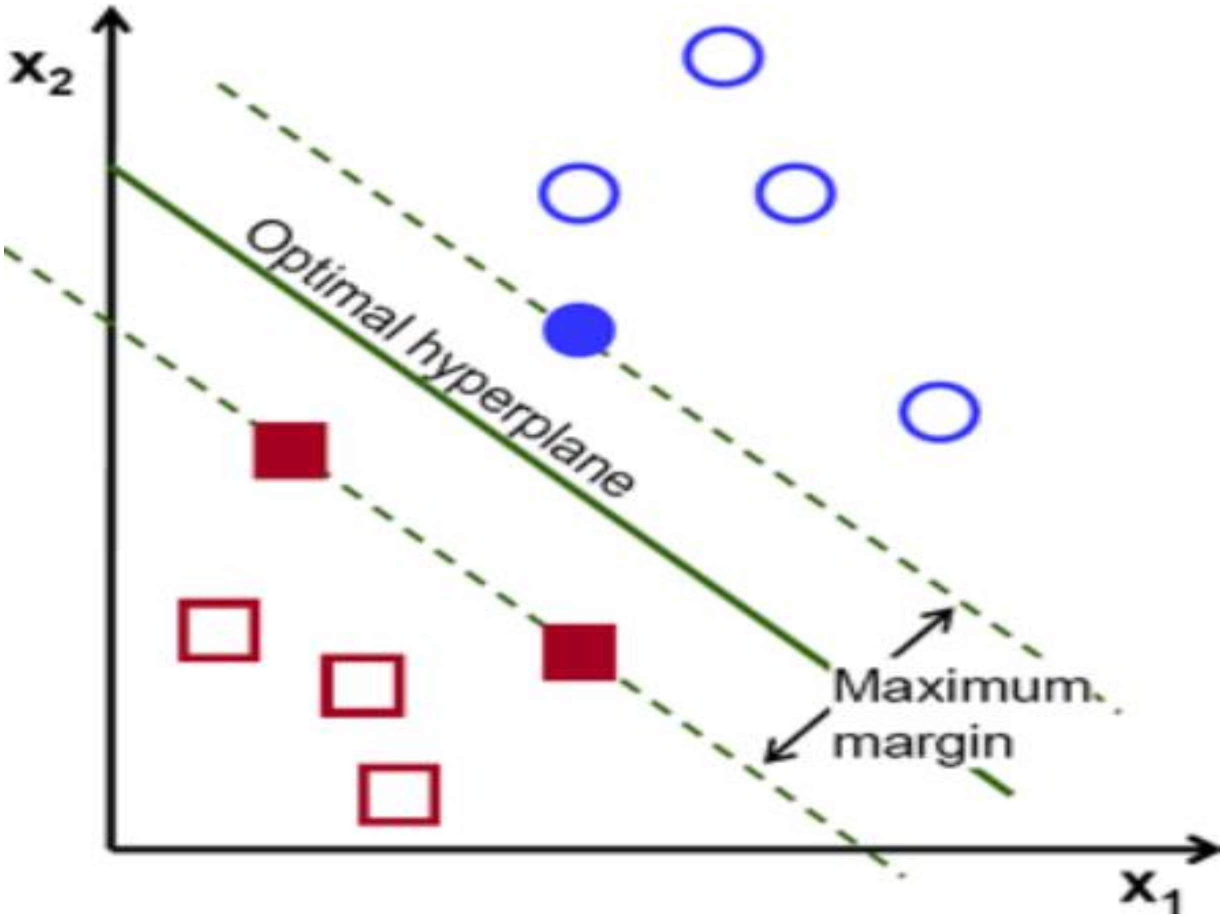



3.4, 10.9
4.2, 13.5
3.4, 10.8
5.4, 17.4
-1.8, -5.7
-2.9, -9.4
-1.7, -5.6
0.8, 2.5
2.7, 8.6
2.3, 7.3
1, 3.2
4.5, 14.5
1.1, 3.5
4.3, 13.9
4.3, 14
-3.3, -10.7
1.6, 5
3, 9.7
-1.5, -4.8
-1.7, -5.5
2.3, 7.3
2.3, 7.2
-2.5, -8.1
2.7, 8.5
1.9, 6
2.1, 6.9
-6.7, -21.6
-6.4, -20.6
-2.4, -7.7
-4.5, -14.4
-3.6, -11.6
-6.4, -20.6
-4.7, -15.2

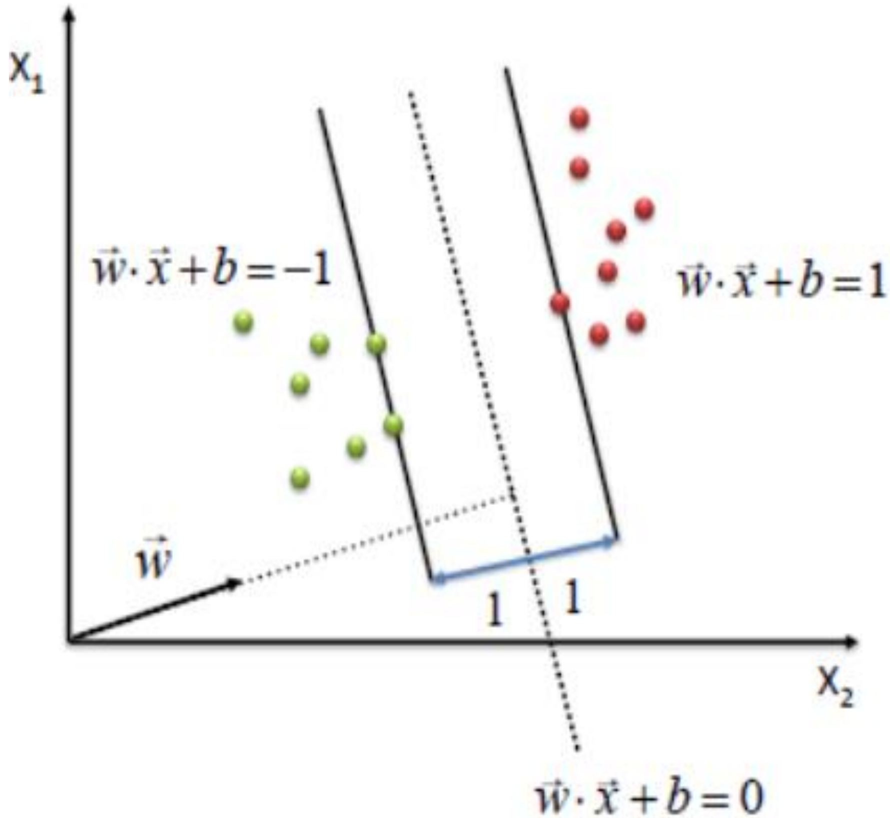
For Linear Separable Data



Maximizing the margin



optimal hyperplane

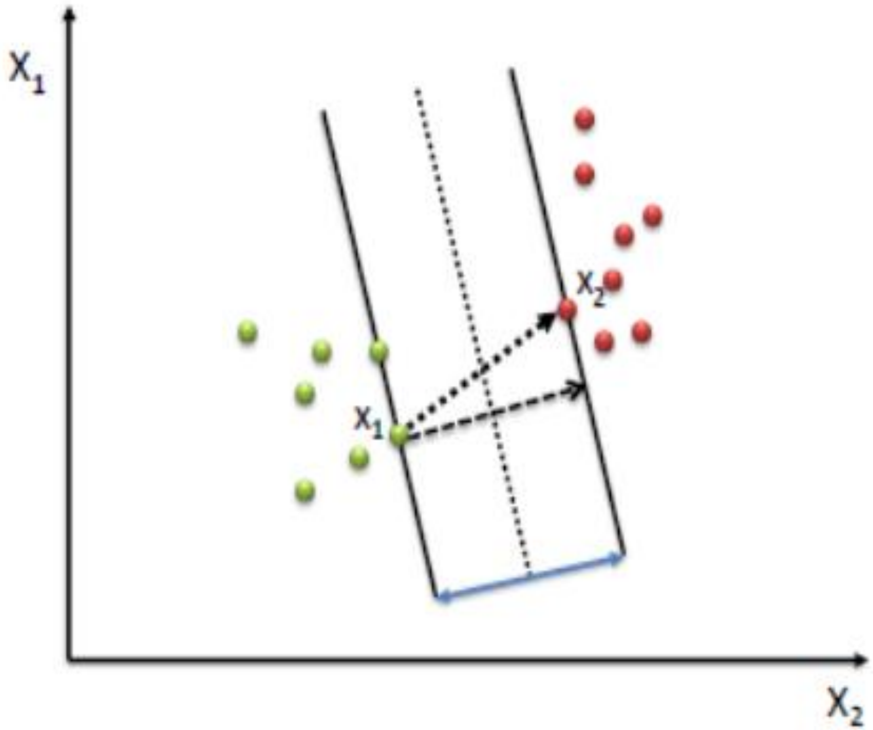


$$\boxed{\max \frac{2}{\|w\|}}$$

$s.t.$

$(w\cdot x+b)\geq 1, \forall x$ of class 1

$(w\cdot x+b)\leq -1, \forall x$ of class 2



$$\frac{w}{\|w\|}\cdot(x_2 - x_1) = \text{width} = \frac{2}{\|w\|}$$

$$w \cdot x_2 + b = 1$$

$$w \cdot x_1 + b = -1$$

$$w \cdot x_2 + b - w \cdot x_1 - b = 1 - (-1)$$

$$w \cdot x_2 - w \cdot x_1 = 2$$

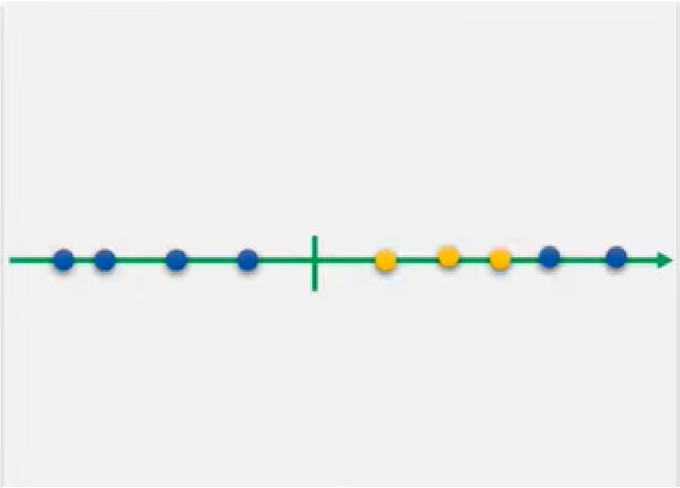
$$\frac{w}{\|w\|}(x_2 - x_1) = \frac{2}{\|w\|}$$

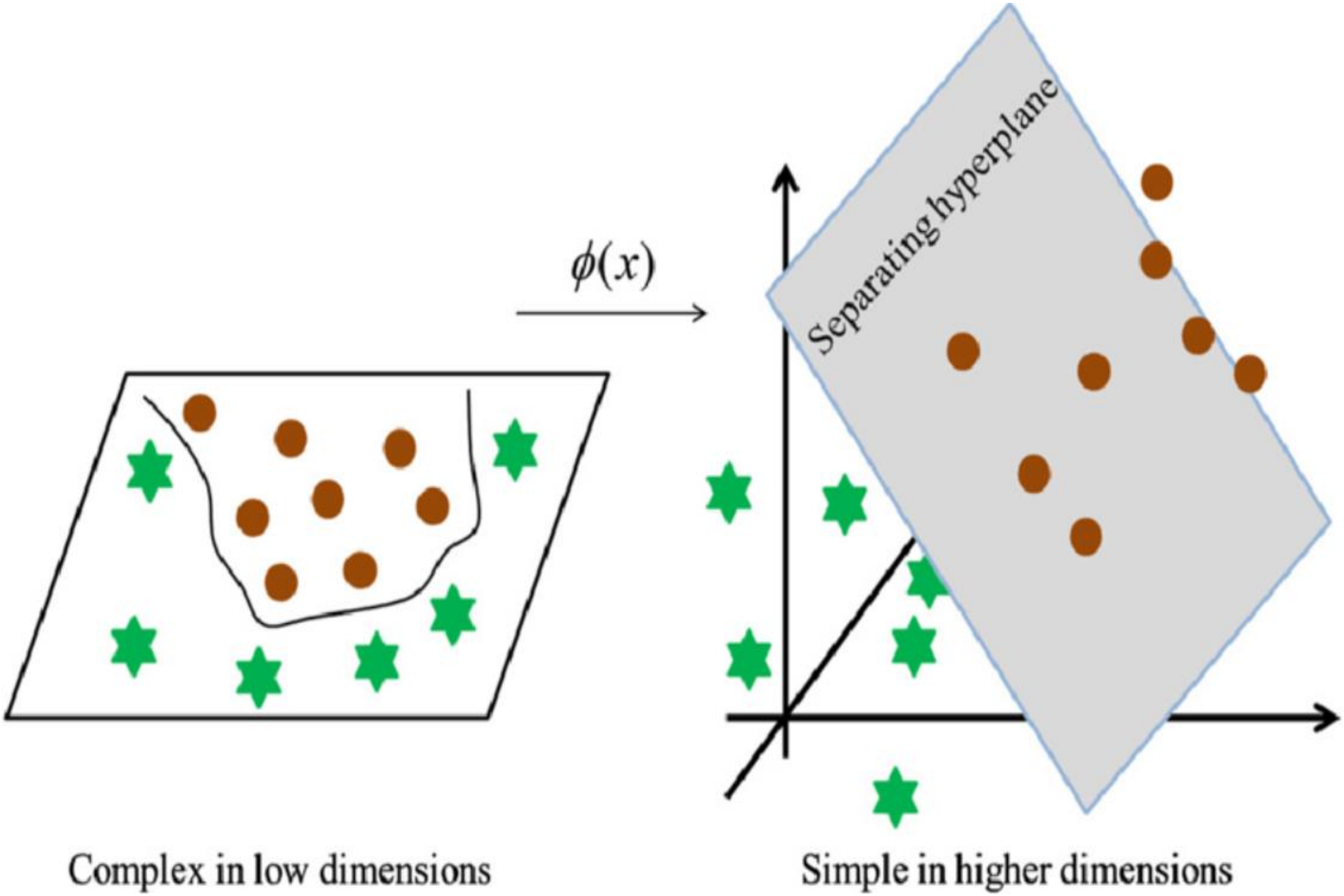
SVM-Basic Technique

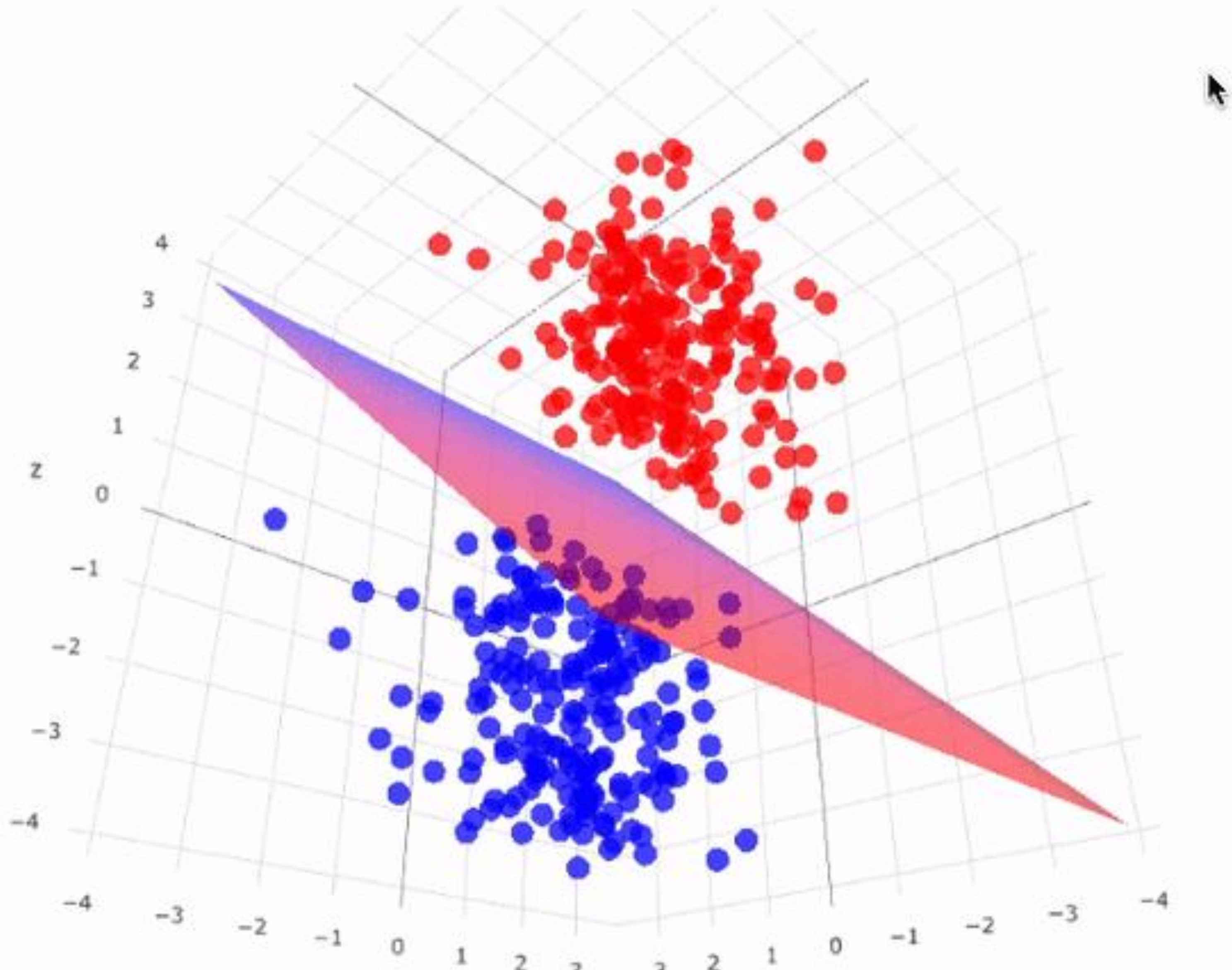
- Transformation
- Illusion
- Separation.

Transformation

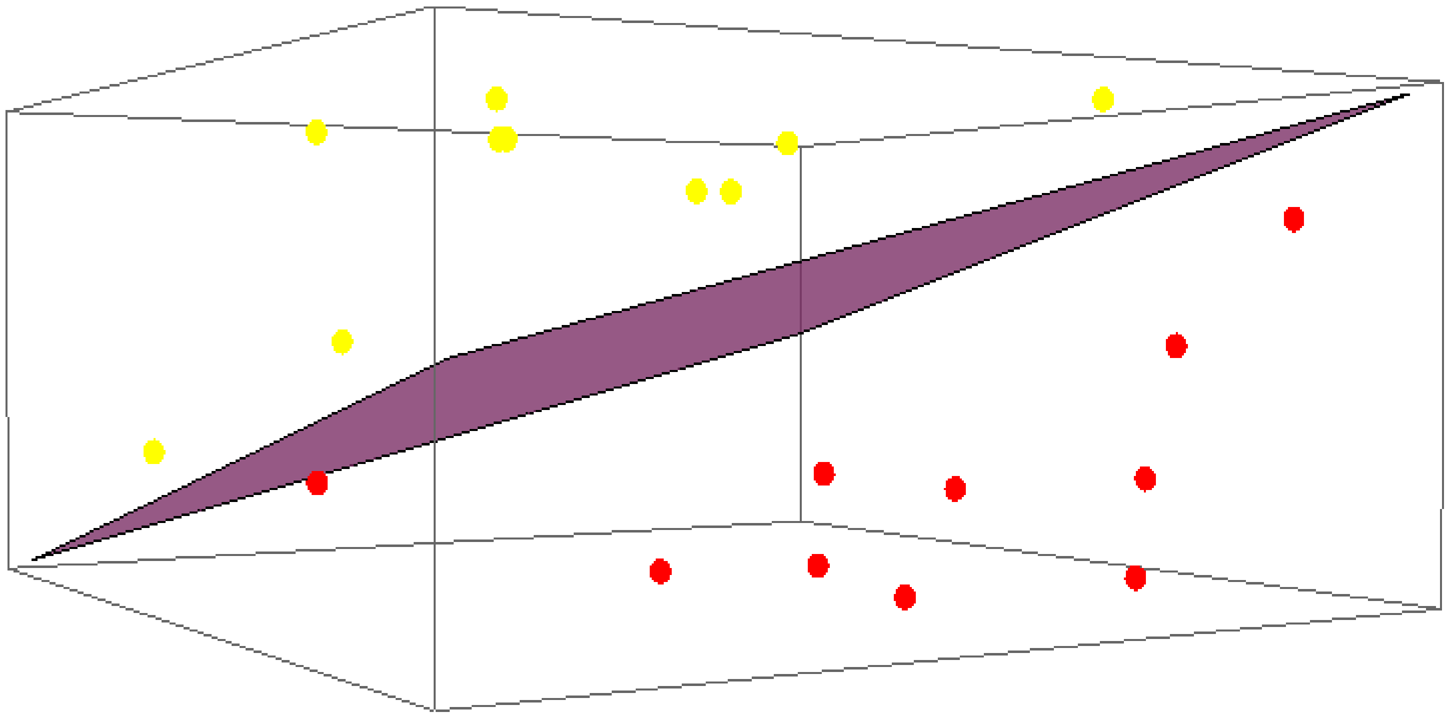
Transformation: If the classes are not linearly separable in a two-dimensional space, SVM uses a higher dimensional space to draw the separating hyperplane(s) between the classes of the target variable. The key point to note here, is, **when unable to separate linearly, it transforms the data to a higher dimension space to get the required separation.**

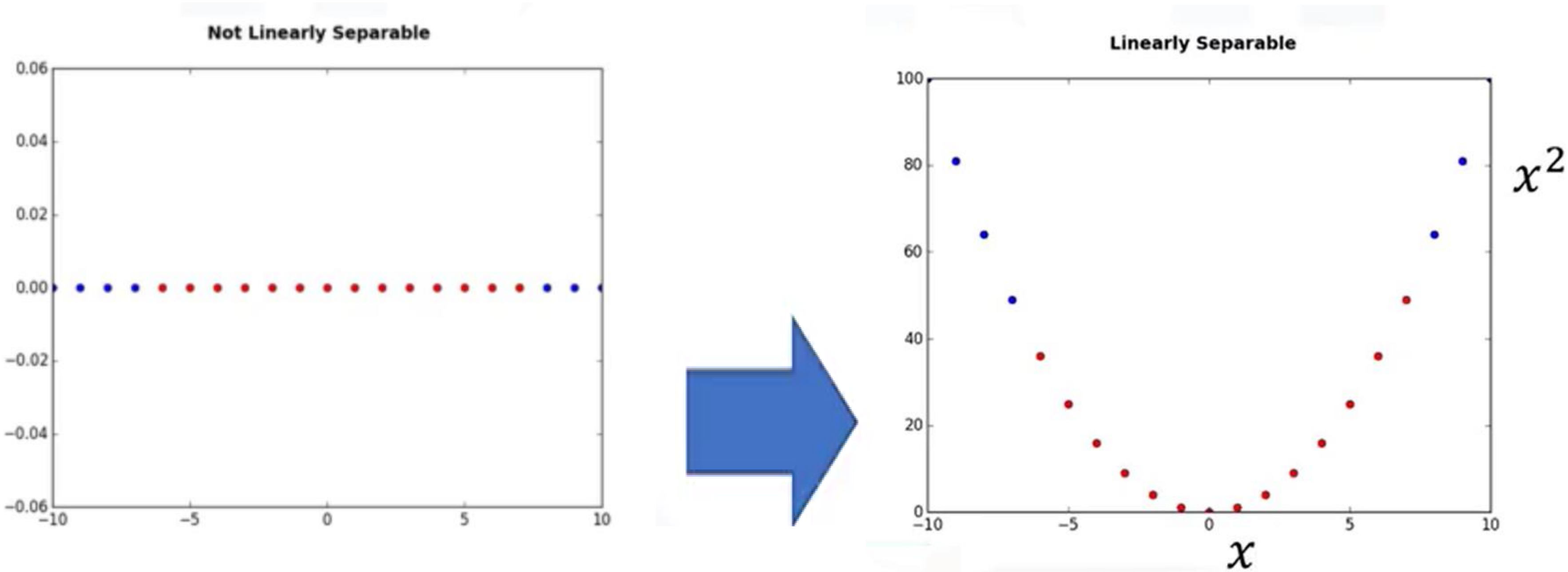



$\phi(x)$
Separating hyperplane
Complex in low dimensions
Simple in higher dimensions

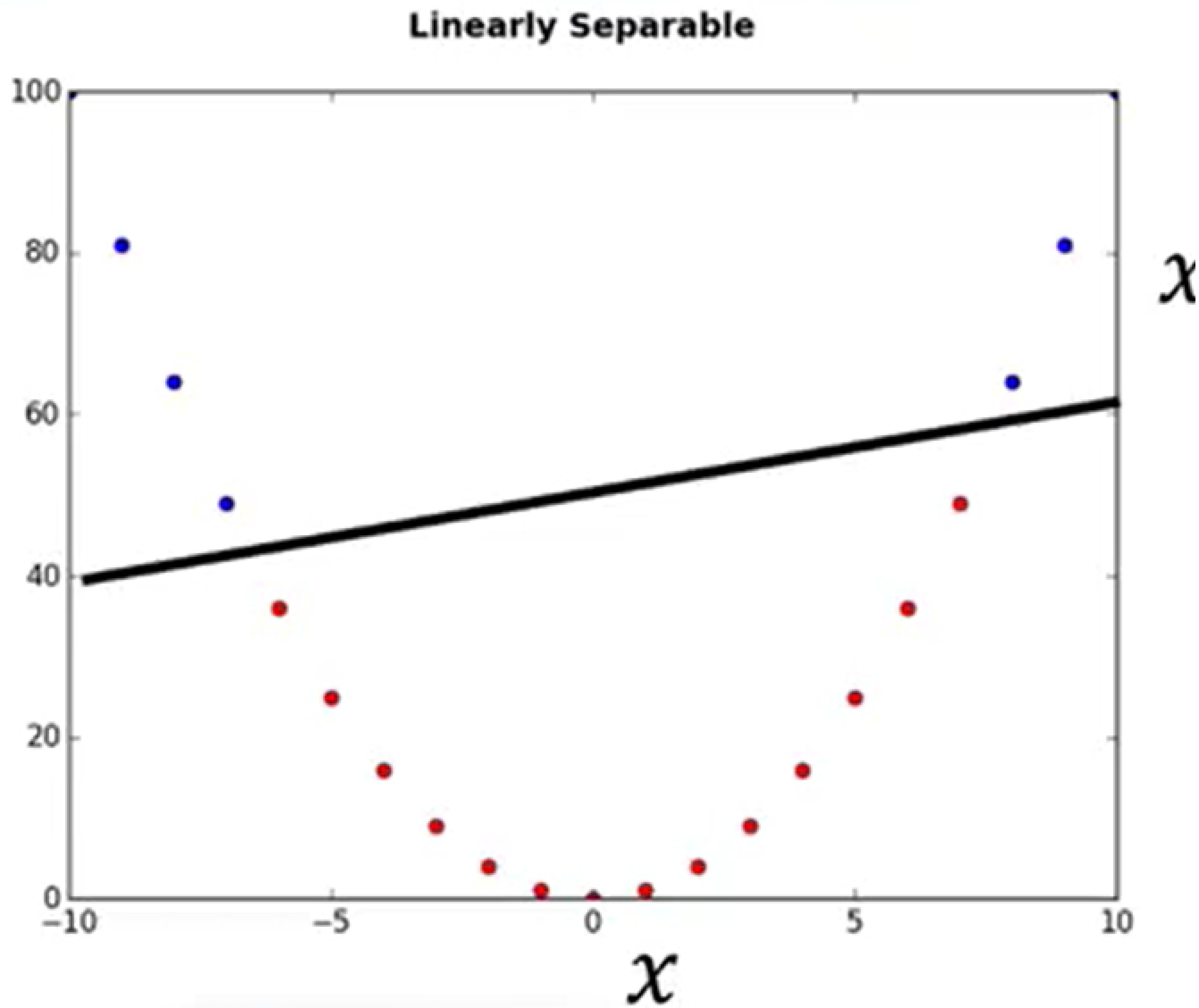


4
3
2
1
z
0
−1
−2
−3
−4
−4 −3 −2 −1 0 1 2 3
3 2 1 0 −1 −2 −3 −4



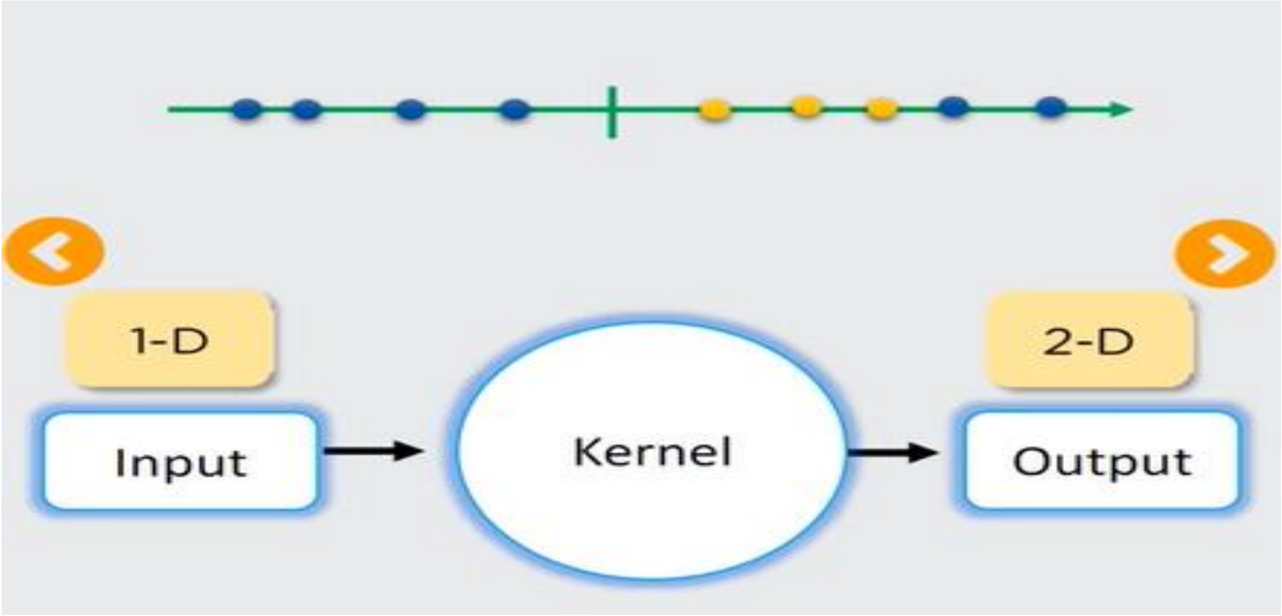

Not Linearly Separable
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−10
−5
0
5
10
Linearly Separable
100
80
60
40
20
0
−10
−5
0
5
10
x^2
x

$$\phi(x) = [x, x^2]$$



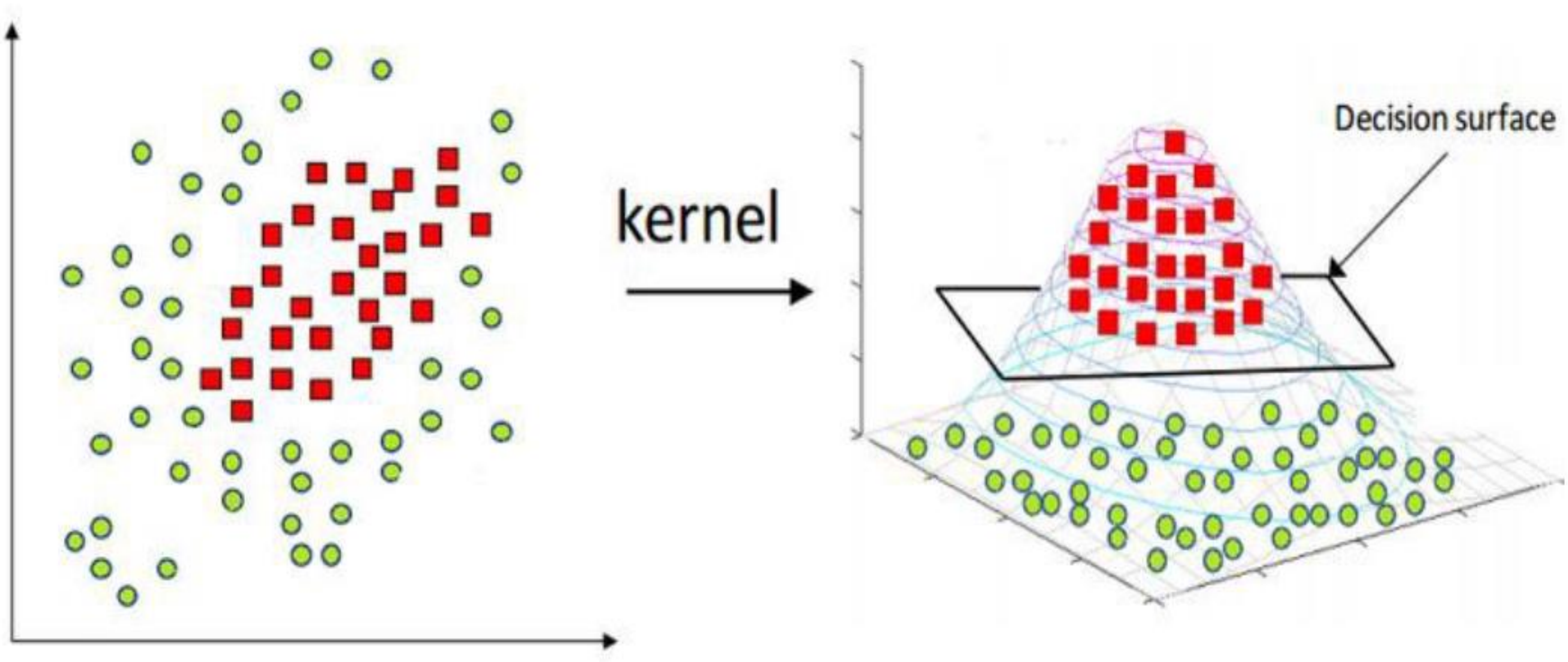
Linearly Separable
100
80
60
40
20
0
−10
−5
0
5
10
x
x

kernel trick

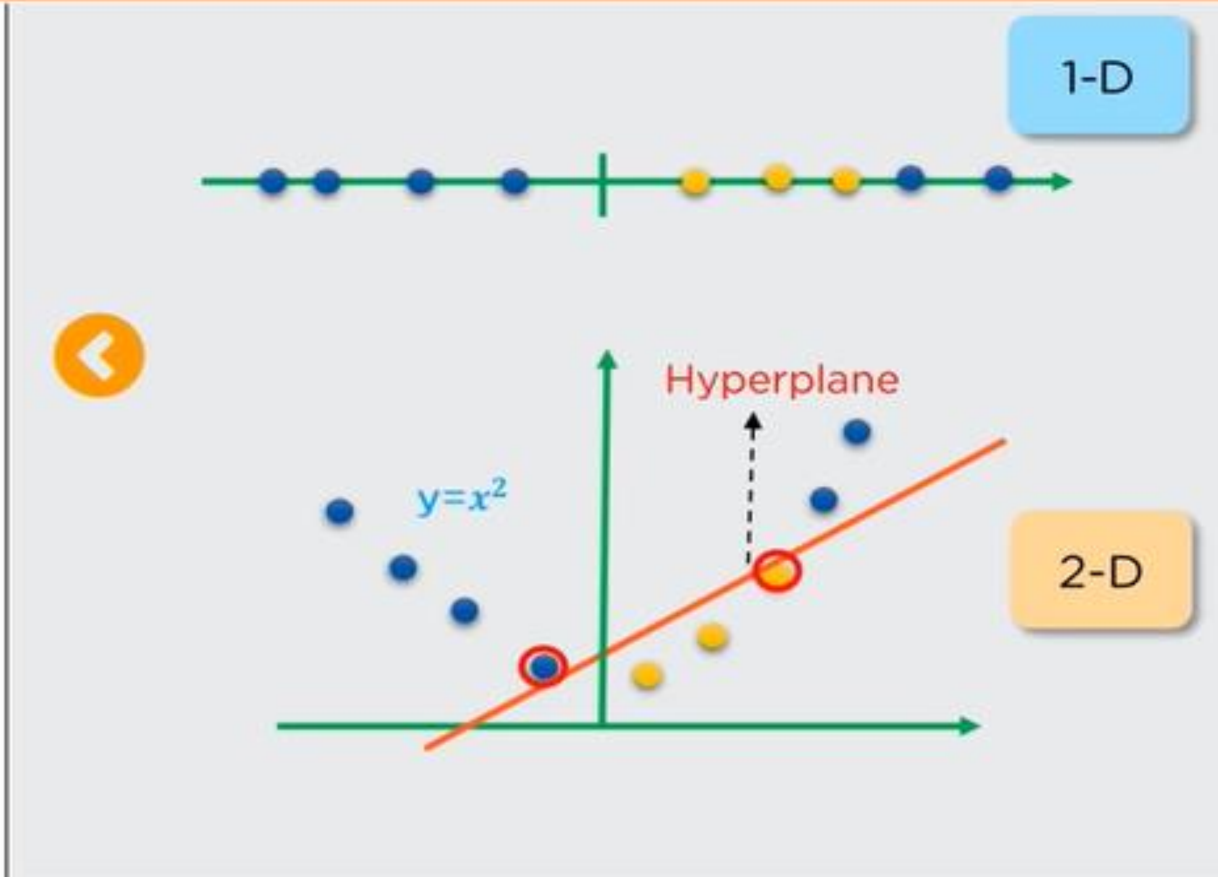


The idea is that our data which is not linearly separable in our 'n' dimensional space may be linearly separable in a higher dimensional space.

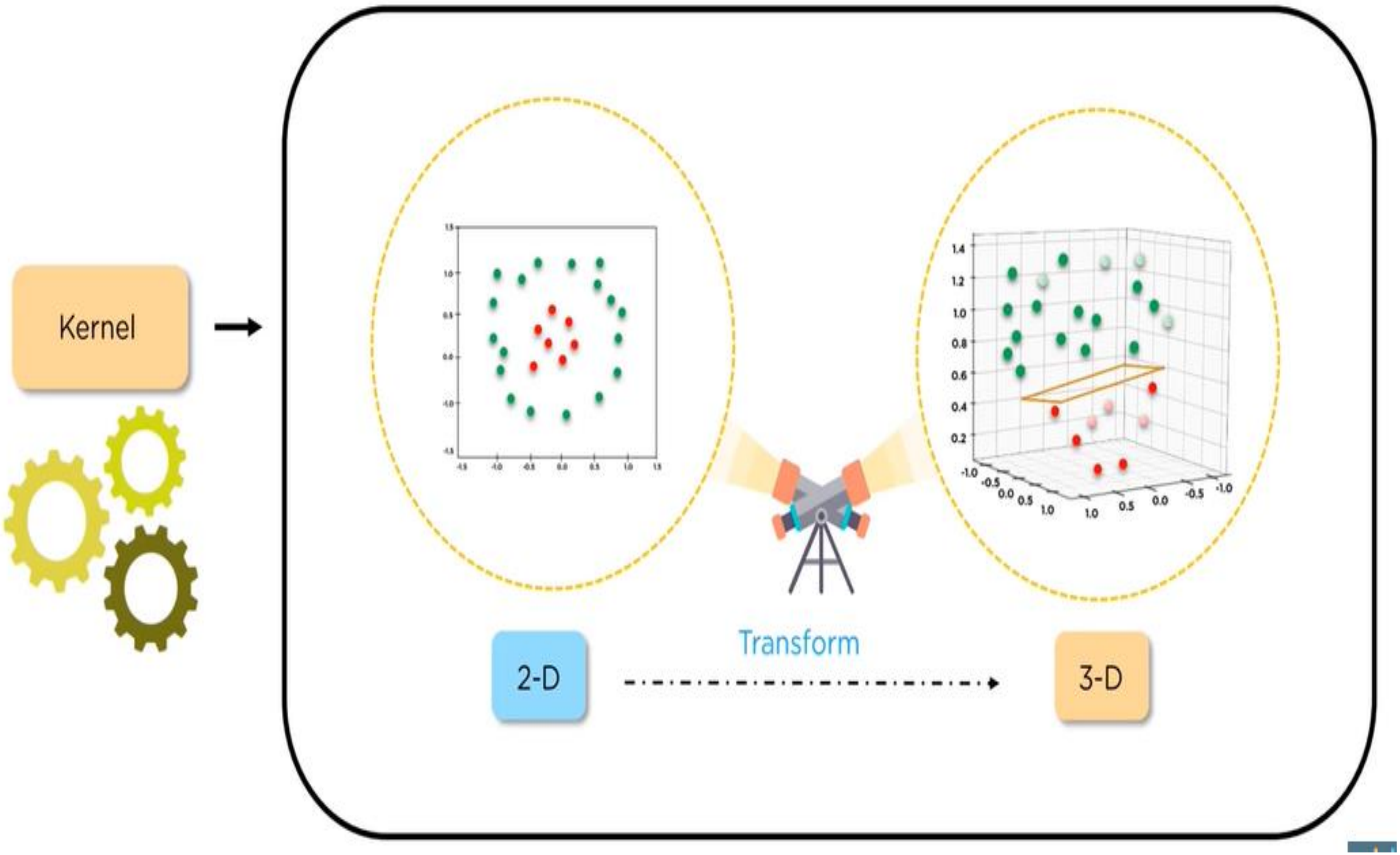
when unable to separate linearly, it transforms the data to a higher dimension space to get the required separation

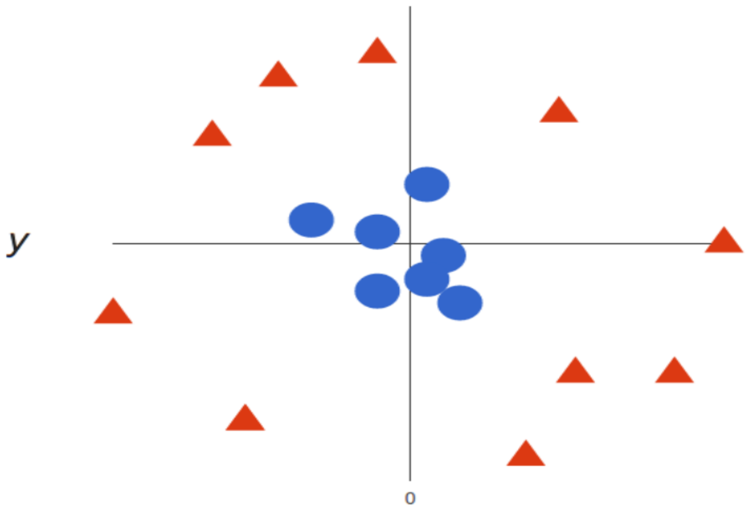


Transformation




Segregate the two classes
Sample Dataset
Not an optimal hyperplane


Kernel
2-D
Transform
3-D



Kernel is a way of computing the dot product of two vectors **x** and **y** in some (possibly very high dimensional) feature space, which is why kernel functions are sometimes called “generalized dot product”.

classify nonlinear data by cleverly mapping our space to a higher dimension.

Type of Kernel

Linear Kernel A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

$$K(x, xi) = sum(x * xi)$$

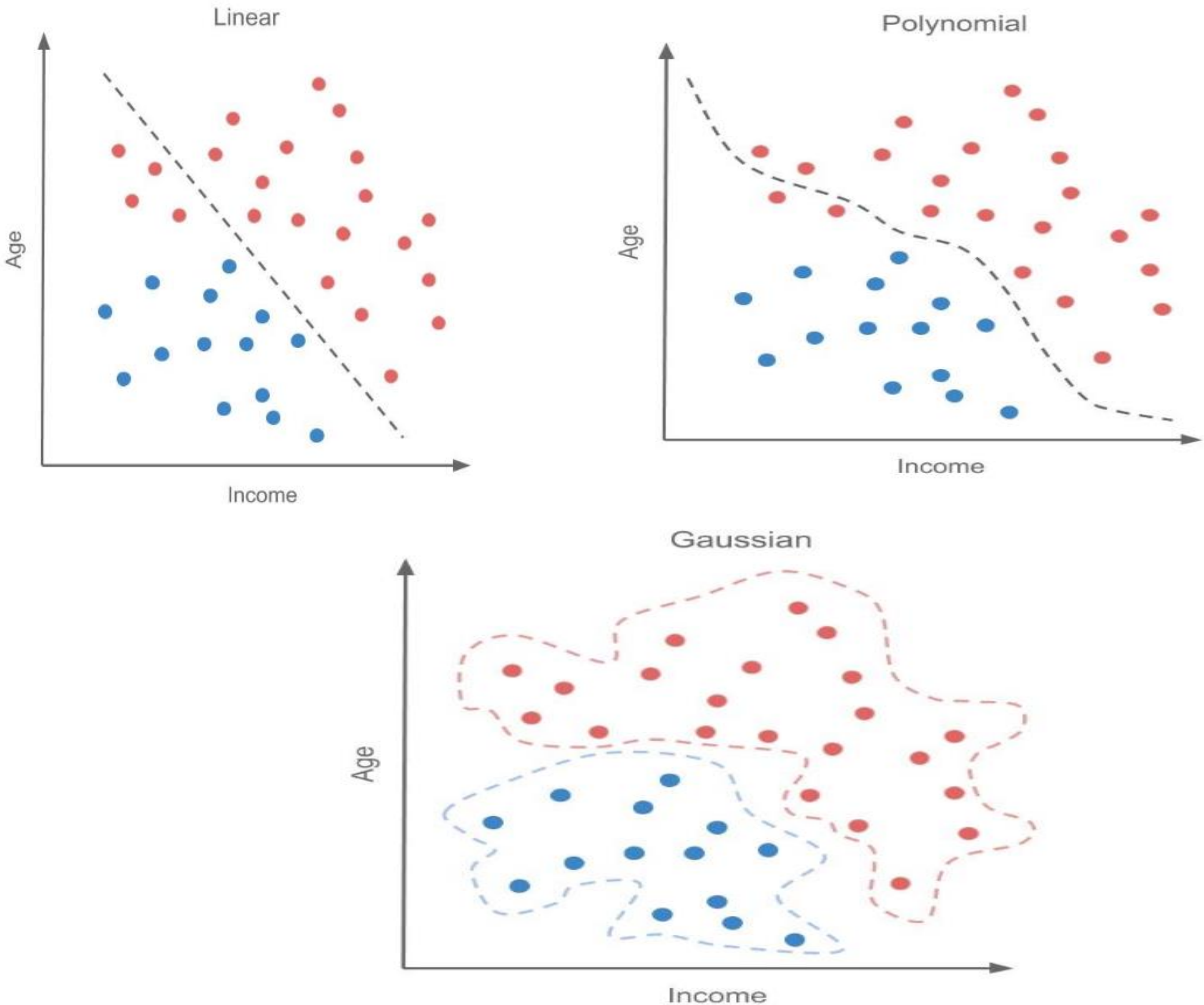
Polynomial Kernel A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

$$\boxed{K(x,xi) = 1 + sum(x * xi)^d}$$

Radial Basis Function Kernel The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

K(x,xi) = exp(-gamma * sum((x – xi^2))

Here gamma is a parameter, which ranges from 0 to 1. A higher value of gamma will perfectly fit the training dataset, which causes over-fitting. Gamma=0.1 is considered to be a good default value. The value of gamma needs to be manually specified in the learning algorithm


Linear
Age
Income
Polynomial
Age
Income
Gaussian
Age
Income

Separation

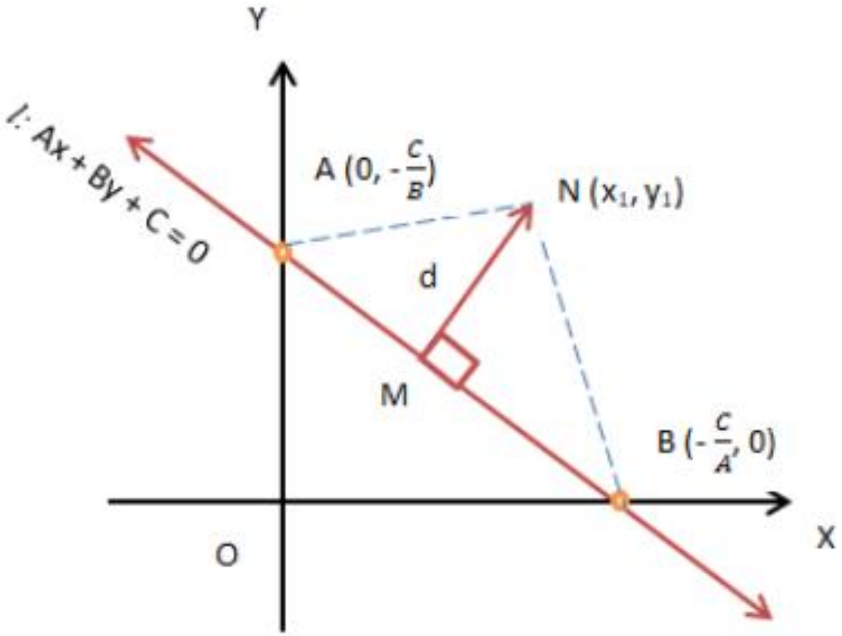
The underlying concept in SVM is the ability of the algorithm to draw the widest possible margin between the classes of the target variable i.e. the observations lying on one side of the margin belong to one class and those lying on the other side of the margin belong to the other class. The wider is the margin, the bigger is the distance between the classes and higher is the confidence level of the classifier.

The advantages of SVM

- High-Dimensionality - The SVM is an effective tool in high-dimensional spaces, which is particularly applicable to document classification and sentiment analysis where the dimensionality can be extremely large.
- Memory Efficiency
- Versatility - Class separation is often highly non-linear.

Perpendicular Distance of Point from a Line

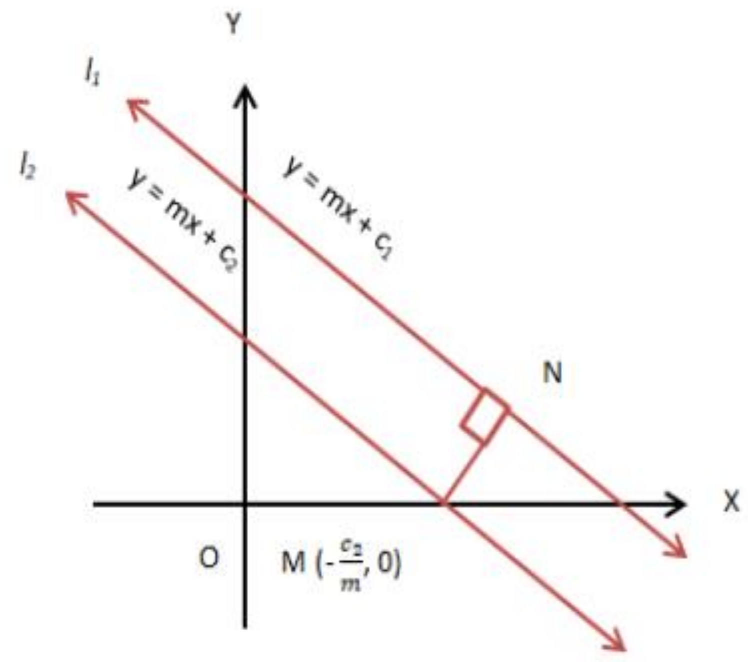
The distance from a point (m, n) to the line Ax + By + C = 0 is given by:



$$d = \frac{|Am + Bn + C|}{\sqrt{A^2 + B^2}}$$

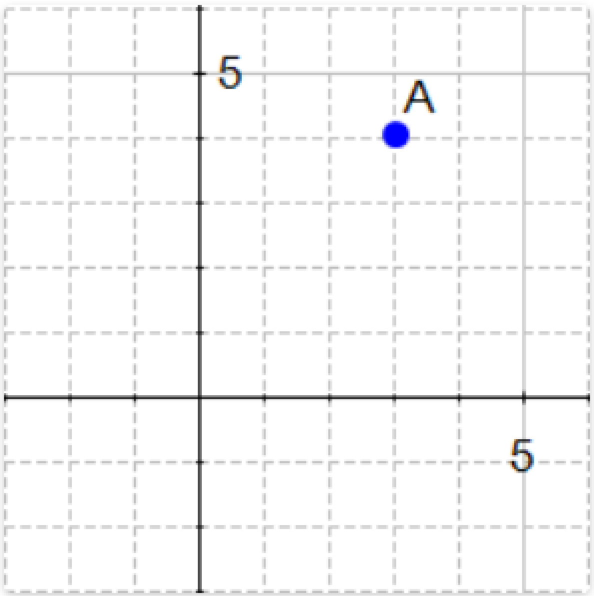
Distance between Two Parallel Lines

$$y = mx + c_1 \ldots \text{(I)}$$
$$y = mx + c_2 \ldots \text{(II)}$$

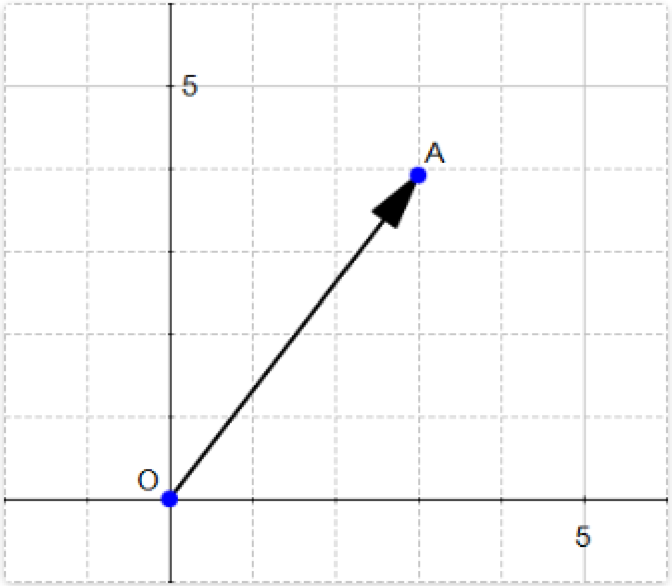


$$\boxed{d = |C_1 - C_2| / (A^2 + B^2)^{½}}$$

Vector



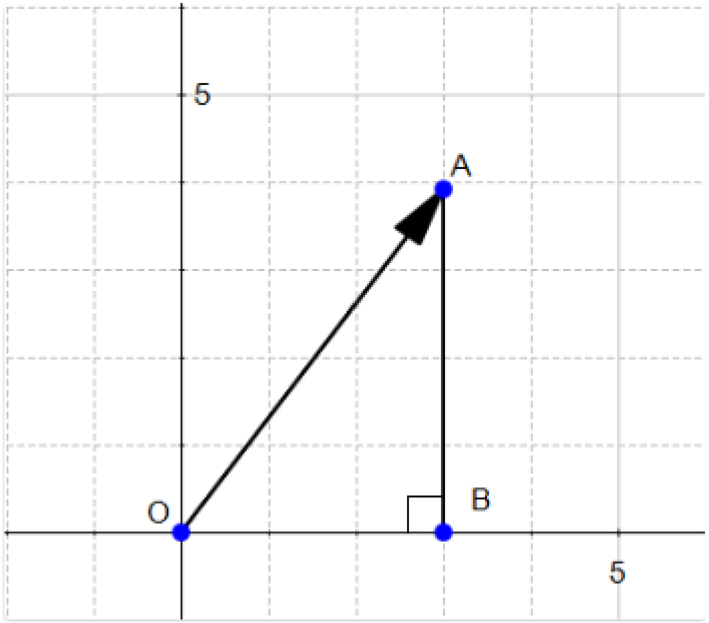
x=(x1,x2),x≠0,inR^2specifies a vector in the plane, namely the vector starting at the origin and ending at x.



A vector is an object that has both a magnitude and a direction.

The magnitude

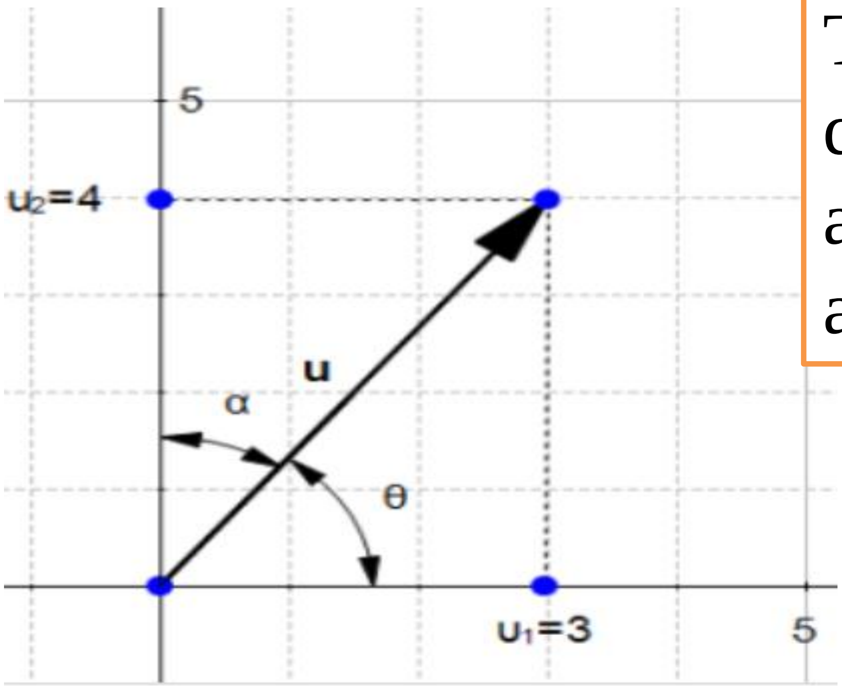
The magnitude or length of a vector x is written ‖x‖and is called its norm. For our vector ‖OA‖ is the length of the segment OA



The direction

The direction of a vector u(u1,u2)is the vector w(u1/‖u‖,u2‖u‖)

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

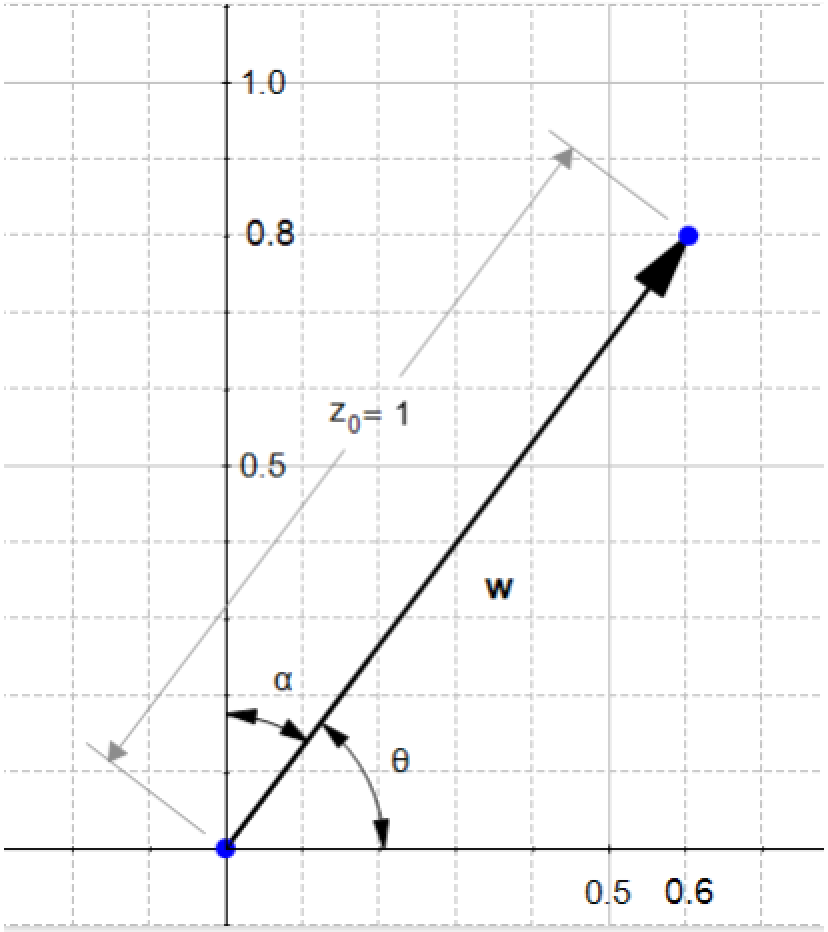


The direction of the vector u is defined by the cosine of the angle θ and the cosine of the angle α

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

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

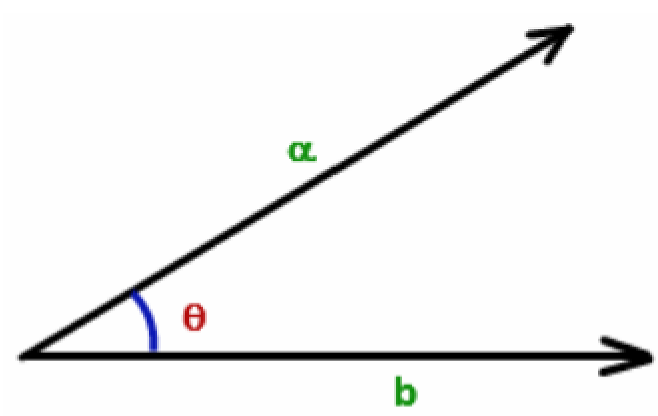
The direction of u(3,4)is the vector w(0.6,0.8)



We can see that w as indeed the same look as u except it is smaller. Something interesting about direction vectors like w is that their norm is equal to 1. That's why we often call them unit vectors.

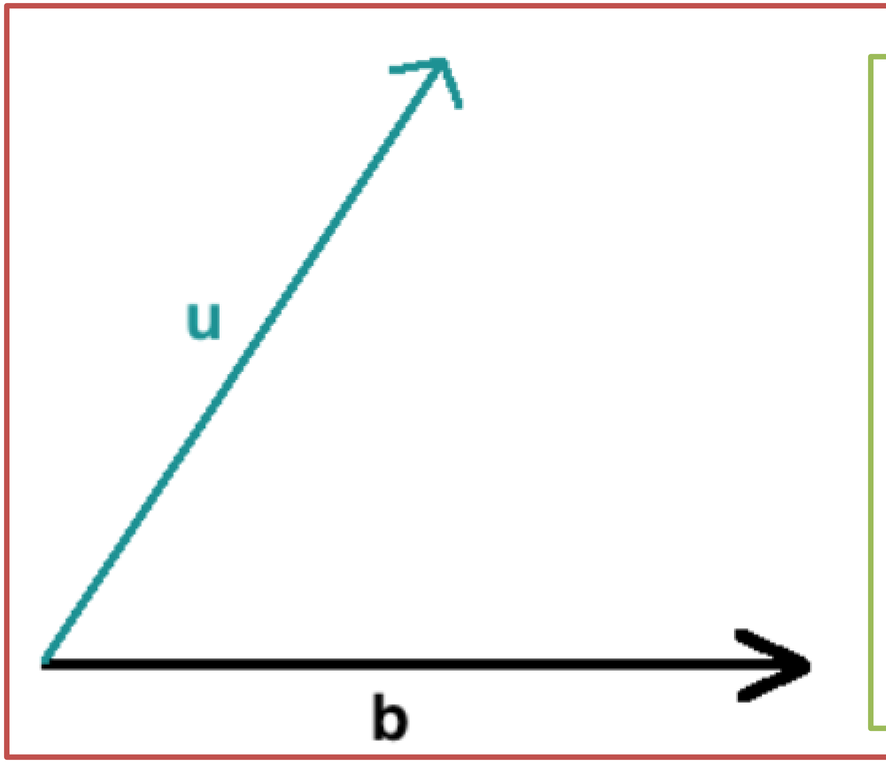
The dot product

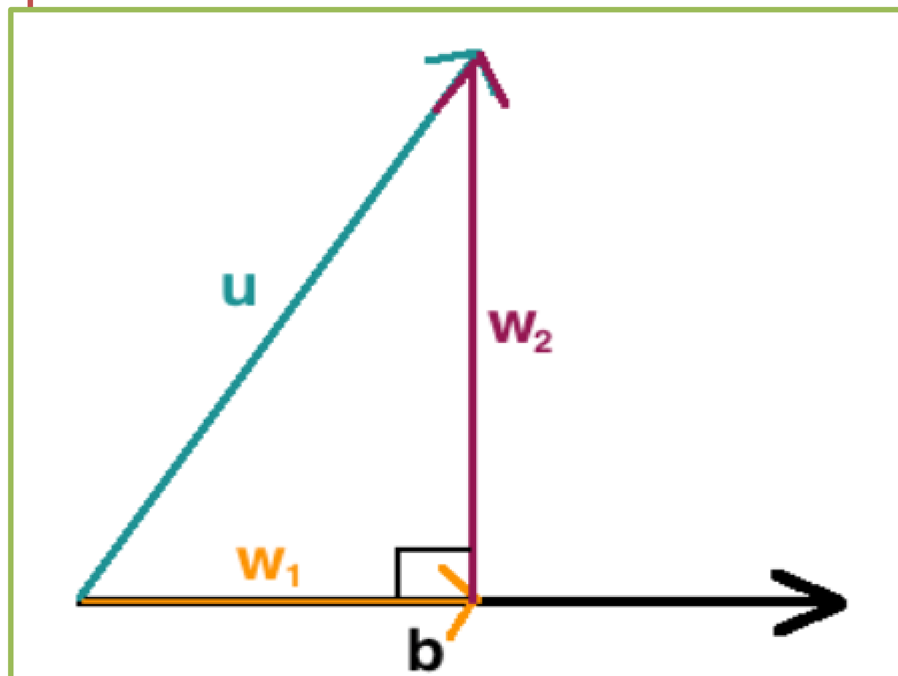
The dot product (also sometimes called the scalar product) is a mathematical operation that can be performed on any two vectors with the same number of elements.

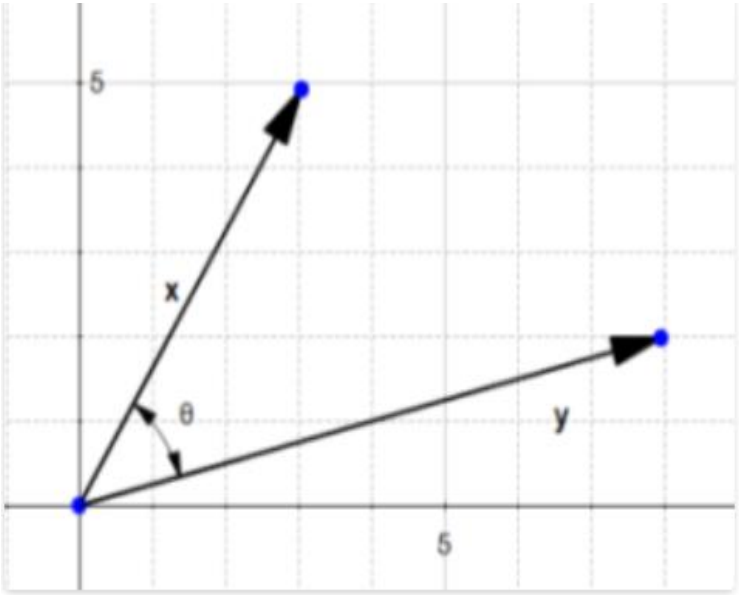


$$a.b = |a|\ |b|\ \mathrm{Cos}\theta$$

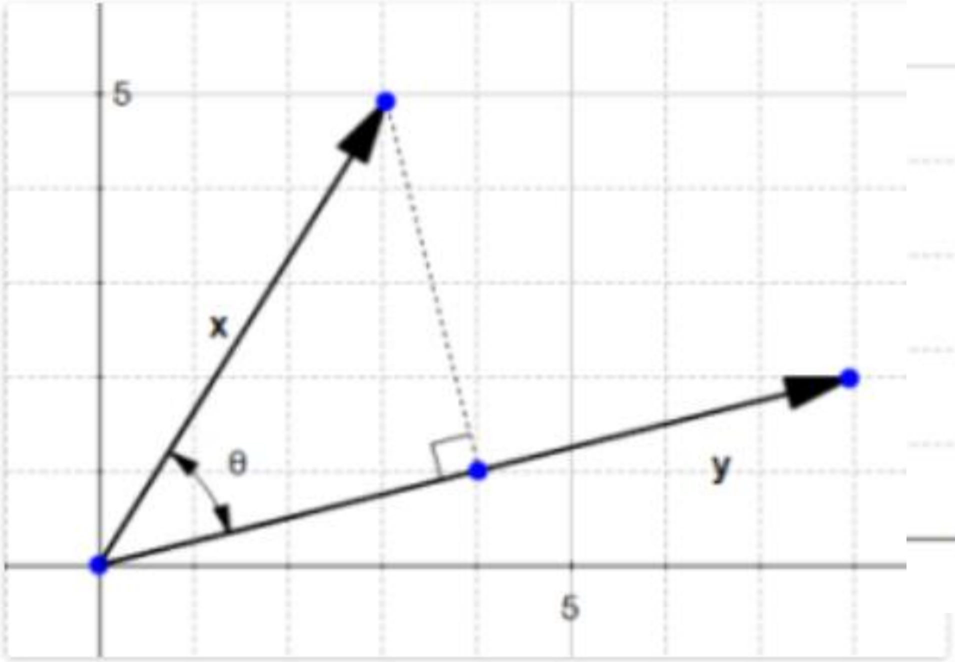
Orthogonal Projections

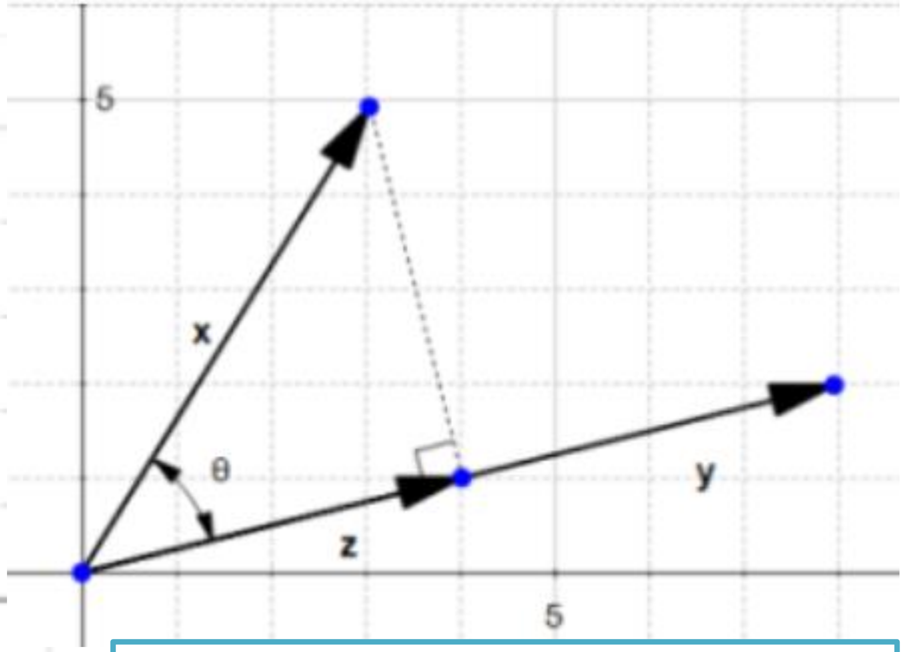






To do this we project the vector x onto y





z is the projection of x onto y

$$cos(\theta) = \frac{||z||}{||x||}$$

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

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

$$||z|| = ||x|| \frac{\mathbf{x} \cdot \mathbf{y}}{||x||||y||}$$

$$||z|| = \frac{\mathbf{x} \cdot \mathbf{y}}{||y||}$$

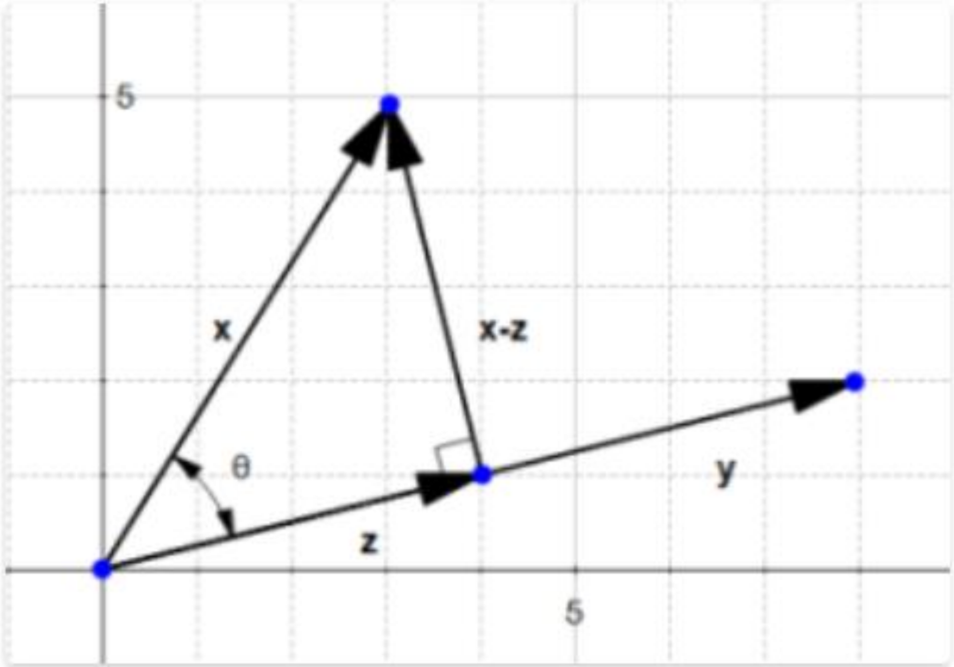
If we define the vector u as the direction of y then

$$\mathbf{u} = \frac{\mathbf{y}}{||y||} \qquad \mathbf{z} = \left\|z\right\|\mathbf{u}$$

The vector $\mathbf{z} = (\mathbf{u} \cdot \mathbf{x})\mathbf{u}$ *is the orthogonal projection of* $\mathbf{x}$ *onto* $\mathbf{y}$.

Why orthogonal projection?

it allows us to compute the distance between x and the line which goes through y.



$$\|x - z\| = \sqrt{(3-4)^2 + (5-1)^2} = \sqrt{17}$$

Definition: If $\vec{u} = \overrightarrow{w_1} + \overrightarrow{w_2}$, $\overrightarrow{w_1} \| \vec{b}$ and $\overrightarrow{w_1} \perp \overrightarrow{w_2}$, then $\overrightarrow{w_1}$ the **Orthogonal Projection of** $\vec{u}$ **Along** $\vec{b}$ denoted $w_1 = \text{proj}_{\vec{b}}\vec{u} = \frac{(\vec{u}\cdot\vec{b})}{\|\vec{b}\|^2}\vec{b}$, and $\overrightarrow{w_2}$ is the **Vector Component of** $\vec{u}$ **Orthogonal to** $\vec{b}$ and $\overrightarrow{w_2} = \vec{u} - \text{proj}_{\vec{b}}\vec{u}$.

If $\vec{u} = \overrightarrow{w_1} + \overrightarrow{w_2}$, $\overrightarrow{w_1} \| \vec{b}$ and $\overrightarrow{w_1} \perp \overrightarrow{w_2}$ then $\|\text{proj}_{\vec{b}}\vec{u}\| = \|\vec{u}\|\,\|\vec{b}\|\cos\theta$.

The SVM hyperplane

y=ax+b

$$y = ax + b$$

$$\mathbf{w}^T\mathbf{x} = 0$$

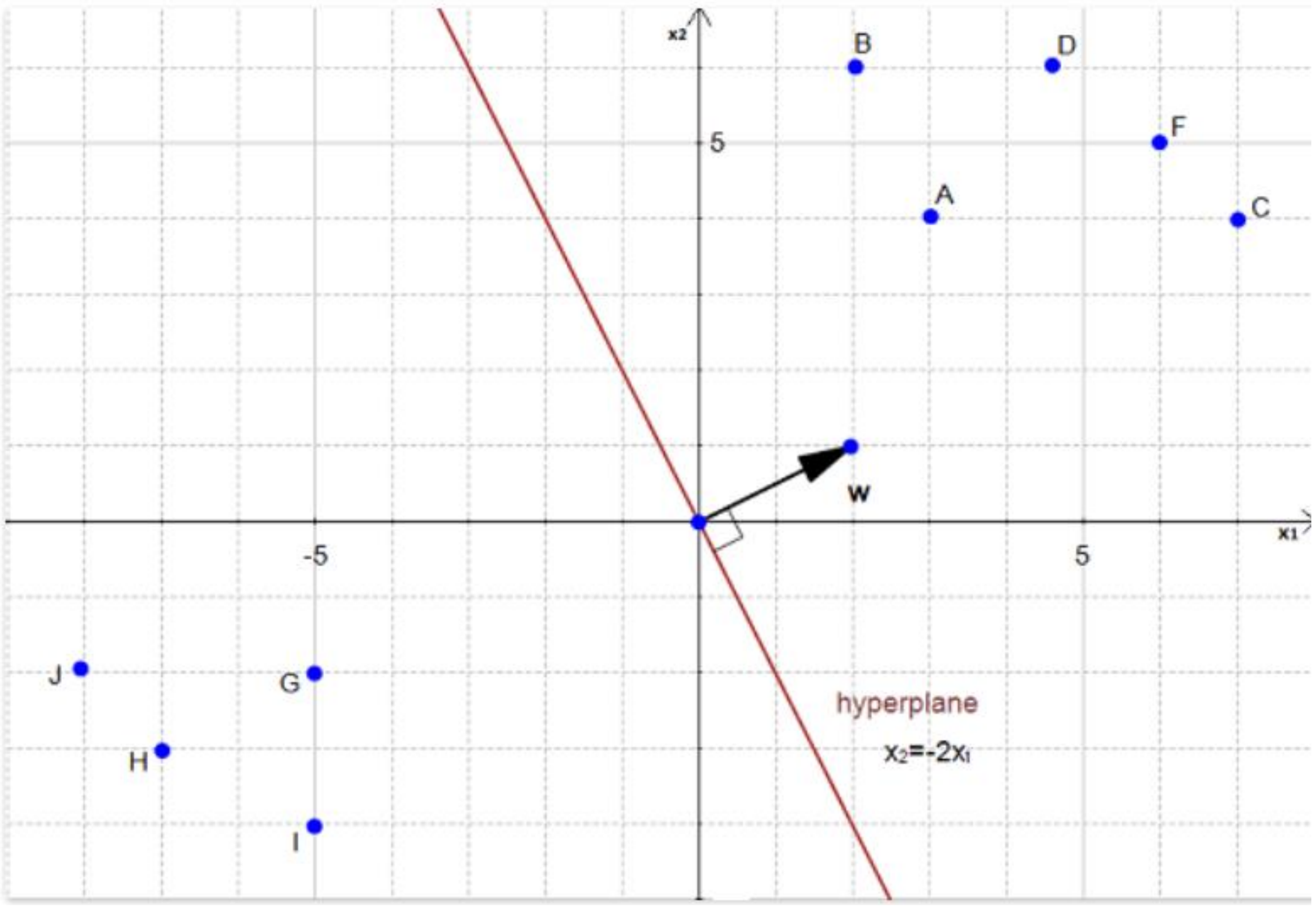
$$y - ax - b = 0$$

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

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

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

Compute the distance from a point to the hyperplane

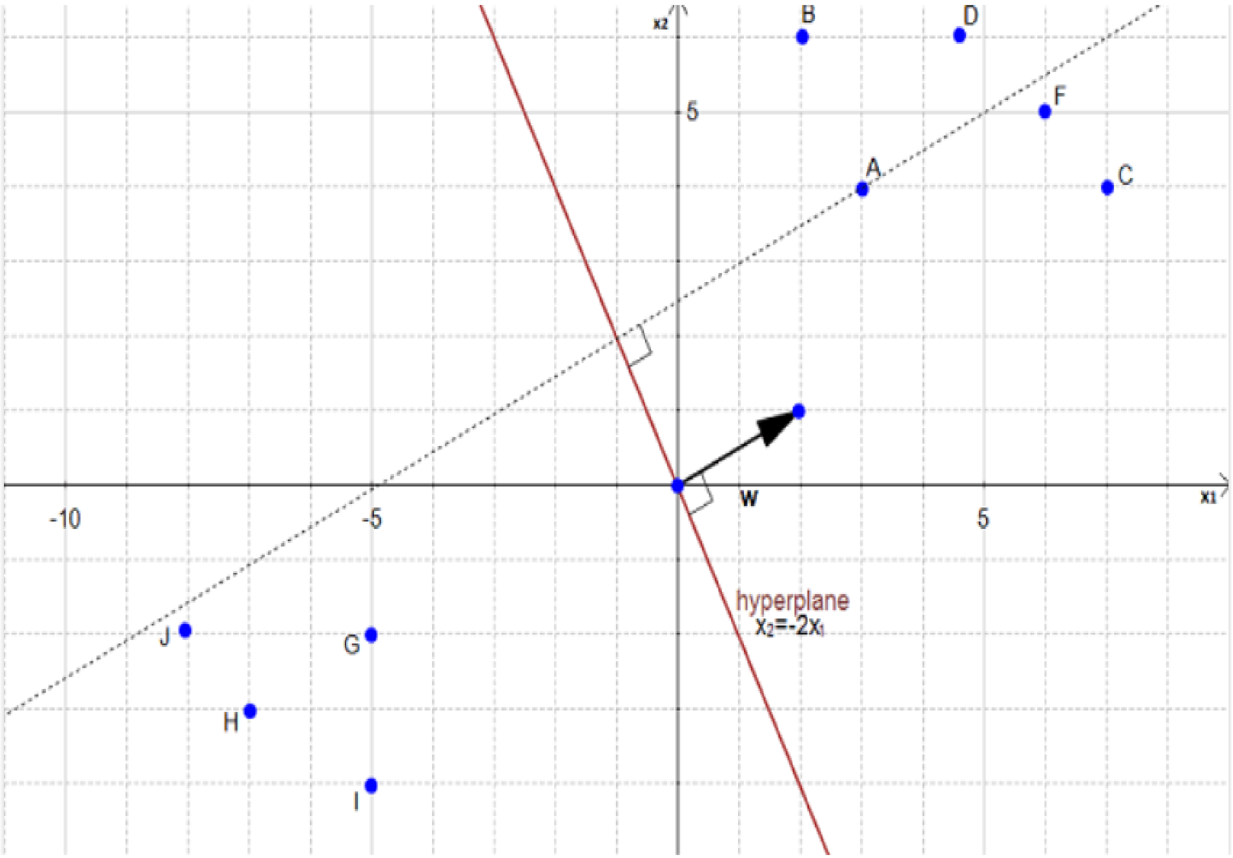


x2=−2x1

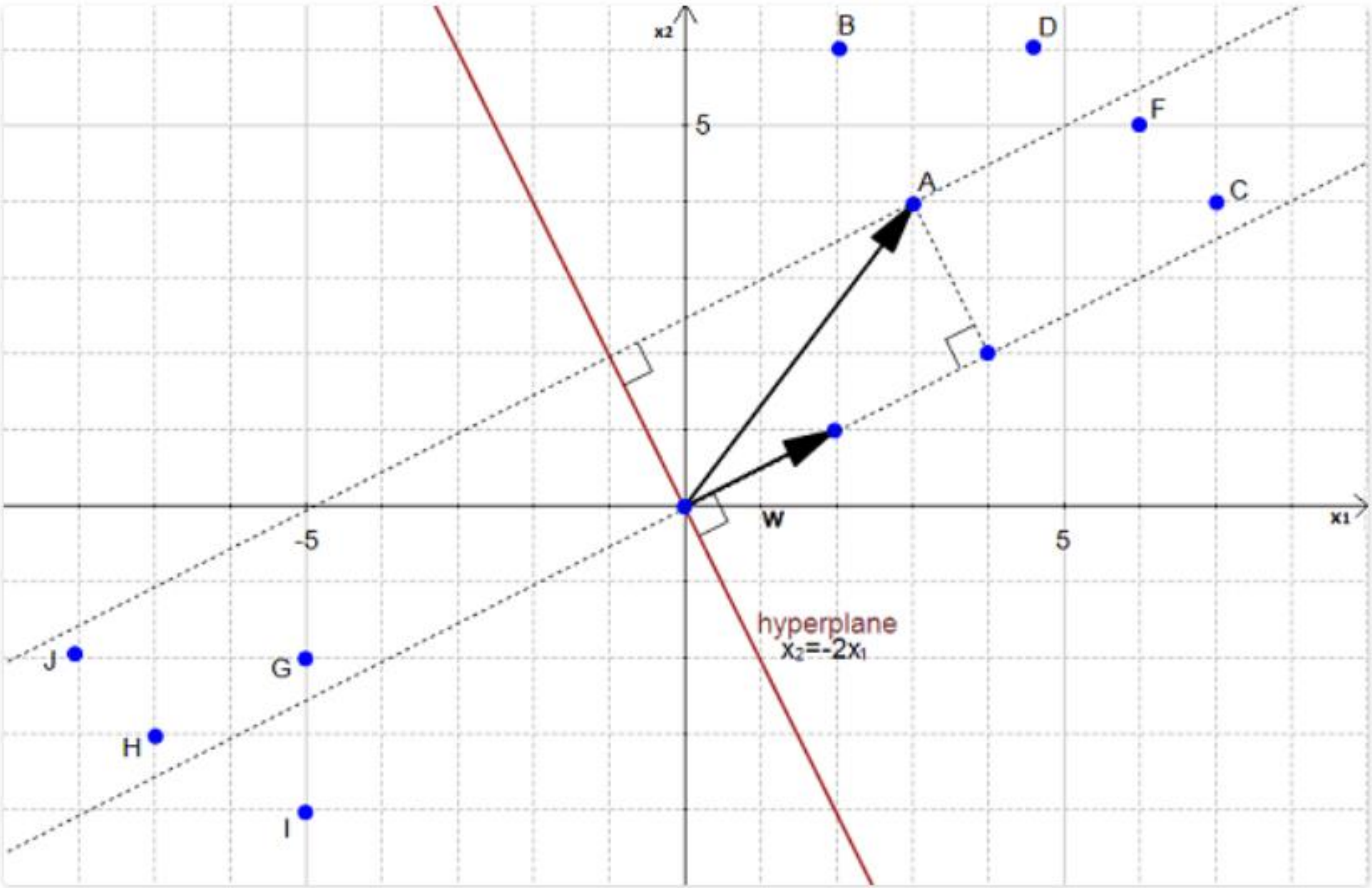
We would like to compute the distance between the point A(3,4)and the hyperplane.

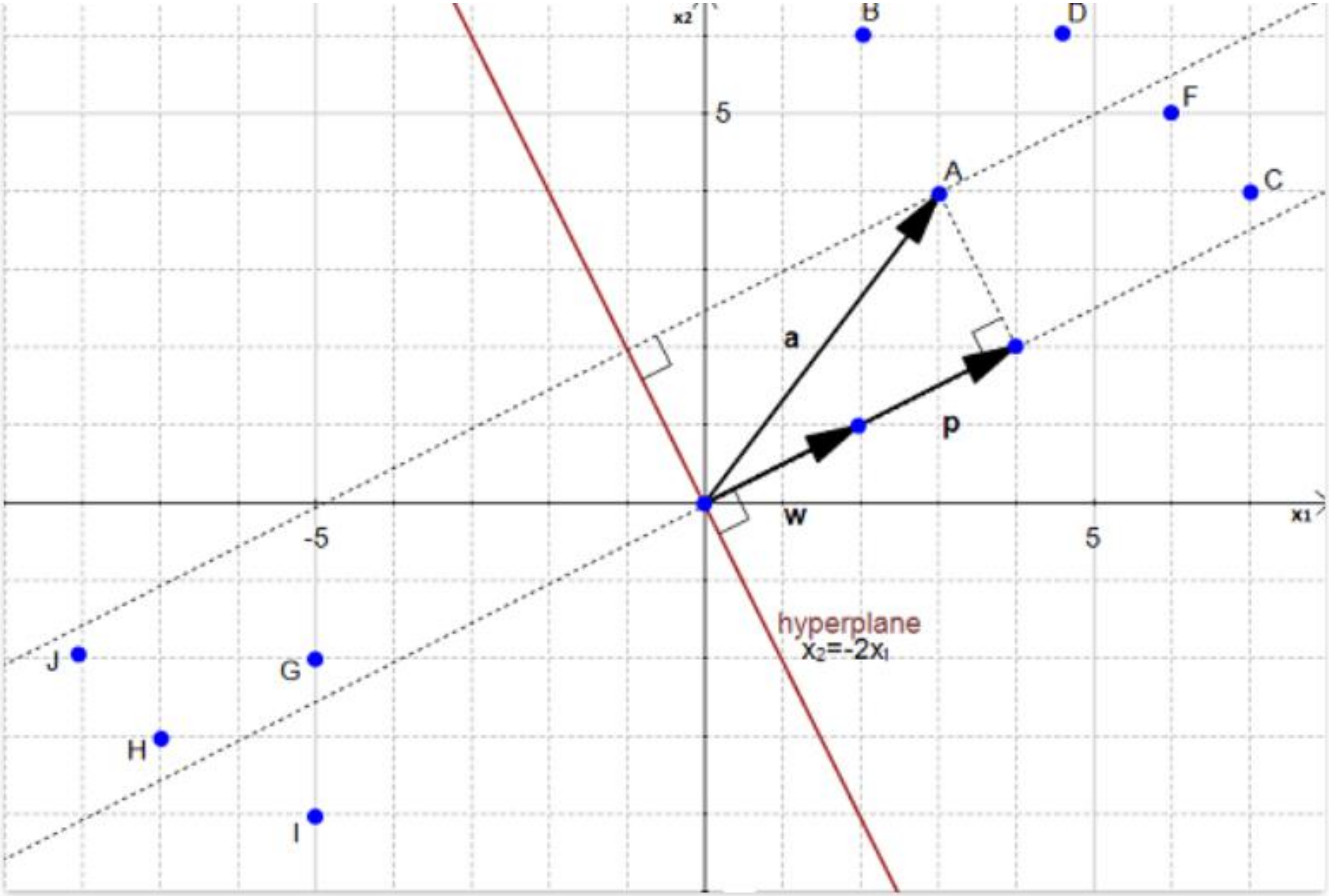
$$\mathbf{w}^T\mathbf{x} = 0$$

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



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





p is the projection of a onto w

Our goal is to find the distance between the point A(3,4)and the hyperplane. 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.

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

Let the vector u be the direction of w

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

P is the orthogonal projection of a onto w so :

$$\mathbf{p} = (\mathbf{u} \cdot \mathbf{a})\mathbf{u}$$

Compute the margin of the hyperplane

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

$$\mathbf{p} = (3 \times \frac{2}{\sqrt{5}} + 4 \times \frac{1}{\sqrt{5}})\mathbf{u}$$

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

$$\mathbf{p} = \frac{10}{\sqrt{5}}\mathbf{u}$$

$$\mathbf{p} = (\frac{10}{\sqrt{5}} \times \frac{2}{\sqrt{5}}, \frac{10}{\sqrt{5}} \times \frac{1}{\sqrt{5}})$$

$$\mathbf{p} = (\frac{20}{5}, \frac{10}{5})$$

$$\mathbf{p} = (4, 2)$$

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

Linear Separating Hyperplanes

The linear separating hyperplane is the key geometric entity that is at the heart of the SVM.
if we have a high-dimensional feature space, then the linear hyperplane is an object one dimension lower than this space that divides the feature space into two regions.

If we consider a real-valued p.
p-dimensional feature space, known mathematically as R^P.
then our linear separating hyperplane is p−1dimensional space embedded within it

If we consider an element of our p-dimensional feature space :x=(x1,...,xp)∈R^p
then we can mathematically define an affine hyperplane by the following equation

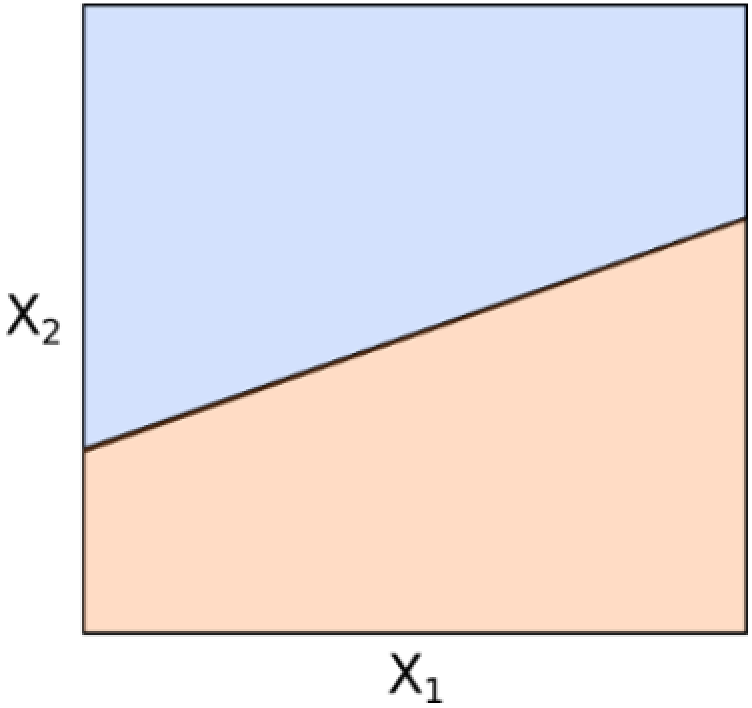
$$b_0 + b_1 x_1 + \ldots + b_p x_p = 0$$

$$b_0 + \sum_{j=1}^{p} b_j x_j = 0$$

Notice however that this is nothing more than a multi-dimensional dot product

$$\vec{b} \cdot \vec{x} + b_0 = 0$$

If an element x∈Rp satisfies this relation then it lives on the p−1-dimensional hyperplane. This hyperplane splits the p-dimensional feature space into two classification regions.



$$\vec{b} \cdot \vec{x} + b_0 > 0$$

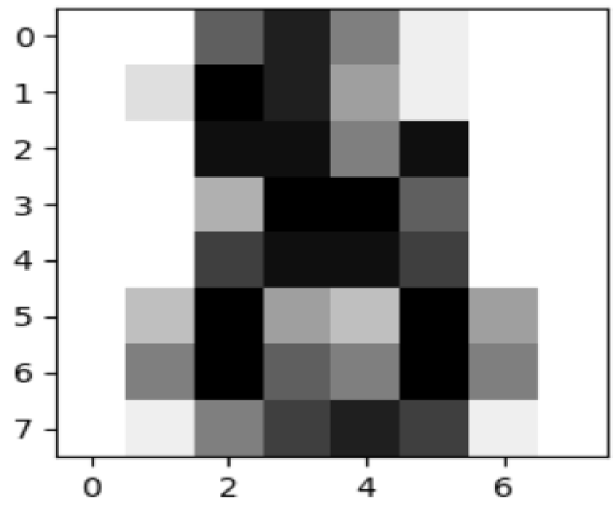
$$\vec{b} \cdot \vec{x} + b_0 < 0$$

Application of Support Vector Machines

- Face Detection
- Text & Hypertext Categorization
- Classification of Images
- Bioinformatics

Digit Dataset for Recognizing hand-written digits

This dataset is made up of 1797 8x8 images. Each image, like the one shown below, is of a hand-written digit. In order to utilize an 8x8 figure like this, we'd have to first transform it into a feature vector with length 64.



UCI Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Pen-Based Recognition of Handwritten Digits Data Set

Download: Data Folder, Data Set Description

Abstract: Digit database of 250 samples from 44 writers

Data Set Characteristics:	Multivariate	**Number of Instances:**	10992	**Area:**	Computer
Attribute Characteristics:	Integer	**Number of Attributes:**	16	**Date Donated**	1998-07-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	157562

http://archive.ics.uci.edu/ml/datasets/PenBased+Recognition+of+Handwritten+Digits

It is 250 samples from 44 writers.
The samples written by 30 writers are used for training, digits written by the other 14 are used for writer independent testing.

Training 7494
Testing 3498

Number of Attributes 16 input+1 class attribute
All input attributes are integers in the range 0..100.
The last attribute is the class code 0..9

Class Distribution

Class: No of examples in training set

```
0:  780
1:  779
2:  780
3:  719
4:  780
5:  720
6:  720
7:  778
8:  719
9:  719
```

Class: No of examples in testing set

```
0:  363
1:  364
2:  364
3:  336
4:  364
5:  335
6:  336
7:  364
8:  336
9:  336
```

sklearn.datasets.load_digits

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers 0-16

```
from sklearn.datasets import load_digits
digitdata=load_digits()
digitdata.keys()
digitdata.data.shape
digitdata.target
digitdata.target_names
digitdata.images
```

```
import matplotlib.pyplot as plt
plt.matshow(digitdata.images[0])
plt.show()
```

Digits Recognition using SVM

```
#Package
from sklearn import datasets
from sklearn.svm import SVC
```

```
#load the Digit Data
mydigitdata=datasets.load_digits()
x_feature=mydigitdata.data
y_target=mydigitdata.target
```

```
#Modeiling
mysvc=SVC(gamma=.001)
mymodel=mysvc.fit(x_feature,y_target)
#Testing
mymodel.predict(x_feature[-1])
```

```
from sklearn import datasets
from sklearn.svm import SVC
from scipy import misc
```

```
#load the image
image=misc.imread("8.jpg")
x_feature.shape
#resize the image
image=misc.imresize(image,(8,8))
image.dtype
mydigitdata.images.dtype
image=image.astype(mydigitdata.images.dtype)
image.dtype
```

```
image
x_feature[-1]
#scale
image=misc.bytescale(image,high=16,low=0)
image
```

```
x_test=[]
for eachRow in image:
    for eachpixel in eachRow:
        x_test.append(sum(eachpixel)/3.0)

x_test
```

```
mymodel.predict(x_test)
```

```
from sklearn import datasets
from sklearn.svm import SVC
from scipy import misc
import numpy as np
mydigitdata=datasets.load_digits()
x_feature=mydigitdata.data
y_target=mydigitdata.target
mysvc=SVC(gamma=.001)
mysvc.fit(x_feature,y_target)
img=misc.imread("4.png")
img=misc.imresize(img,(8,8))
#print(img)
#print(img.dtype)
#print(mydigitdata.images.dtype)
```

```
#print(x_feature.shape)
#print(mysvc.predict([x_feature[-1]]))
img=img.astype(mydigitdata.images.dtype)
#print(img.dtype)
#print(img)
#print(x_feature[-1])
img=misc.bytescale(img,high=16,low=0)
x_test=[]
for eachRow in img:
    for eachpixel in eachRow:
        x_test.append(sum(eachpixel)/3.0)

#print(x_test)
print(mysvc.predict([x_test]))
```