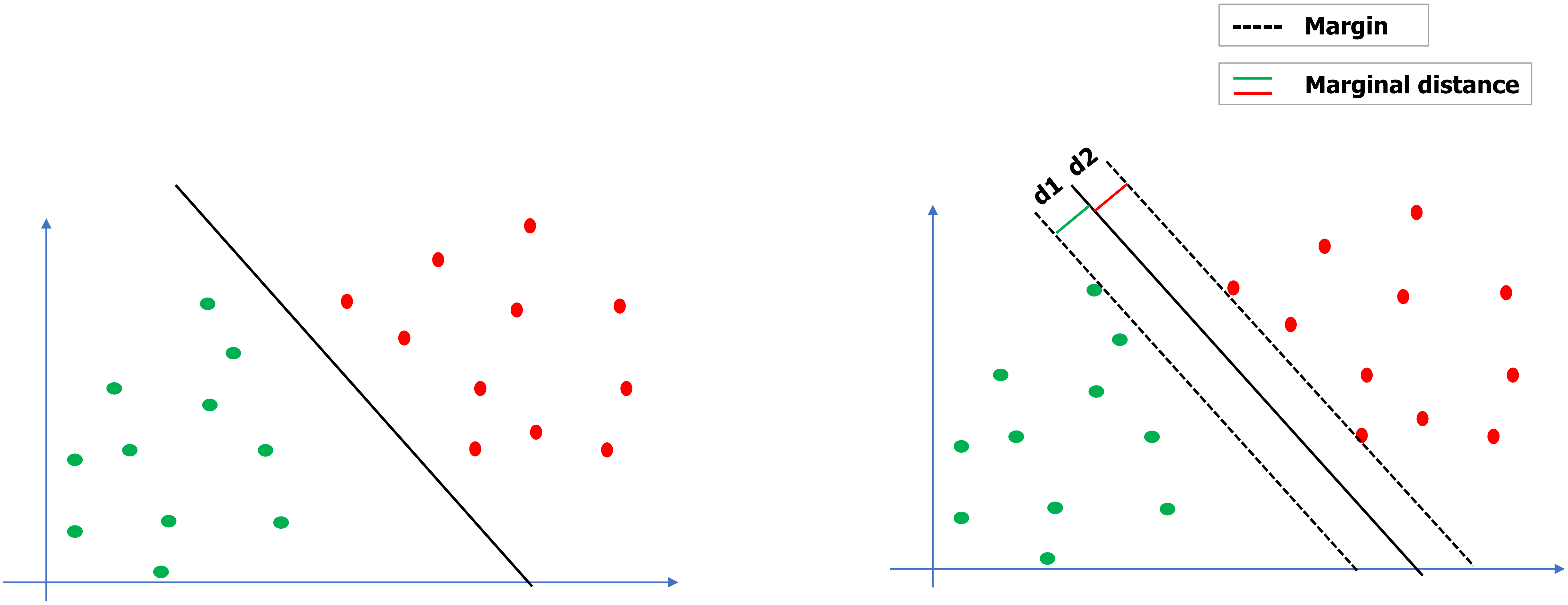
SVM
Support Vector Machines

SVM – Support Vector Machine

- Supervised machine learning algorithm
- Classification technique for **linear and non-linear** separable classes
- Alternate to Logistic regression
- Classification based on finding a **hyperplane** that maximises the margin between two classes
  - Mathematically speaking, SVM's are co-ordinates of the data/observations
- Mainly used in binary classification
  - Can be used in multiclass classification by implementing 'One-Vs-All' method
- SVM algorithm has a feature to ignore outliers
- Complex algorithm, computing resources high, but SVM performs very well

Let us consider a Binary Classification Problem that is Linearly separable



Margin
Marginal distance
d1
d2

The goal of SVM

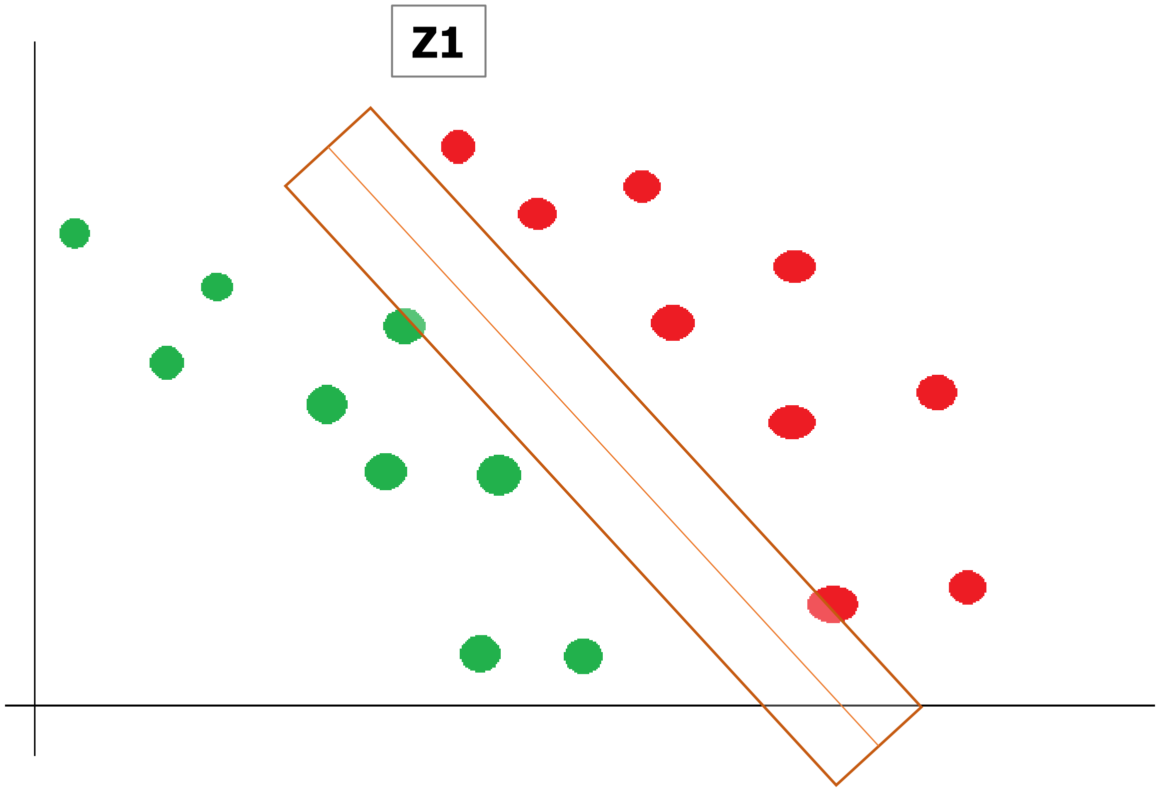
- Find a hyperplane that maximises the distance between the 2 nearest opposite classes

- The margin signifies a generalised model that would give a more accurate model

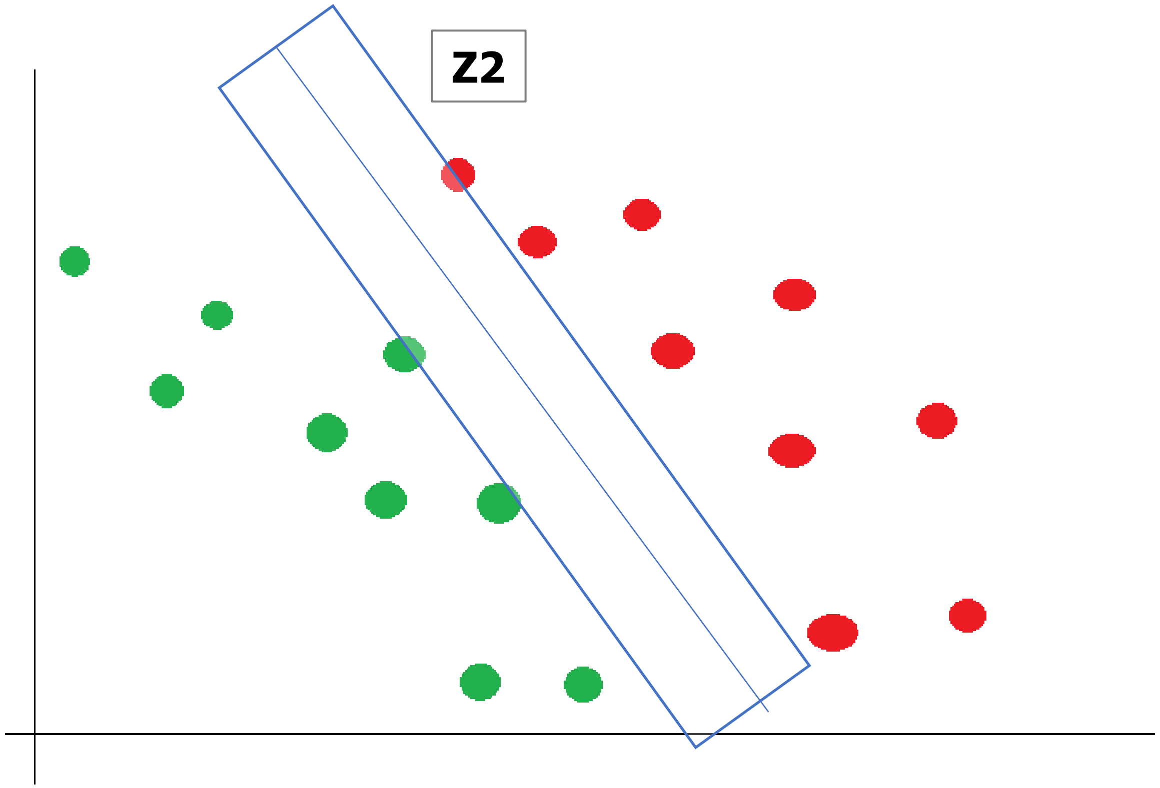
- The hyperplane leaves the widest possible "cushion" between input points from two classes. Trade-off between
  - "narrow cushion, little / no mistakes"
  - "wide cushion, quite a few mistakes"

- A narrow margin may do a good job at separating the training classes, but it is prone to misclassifications of the test data

Z2 > Z1



$w^T.x+b <= -1$ *for Green classes*



$w^T.x+b >= 1$ *for Red classes*

What are Support Vectors ?

- Support Vectors are the data points through which the marginal lines passes

- The number of SV's can vary depending upon the data

General equation

$y = w_1x_1 + w_2x_2 + \ldots\ldots + b$

Where

w: weight associated with feature x

Y > +1 : Positive Class

Y < -1 : Negative Class

We can generalise the equation as

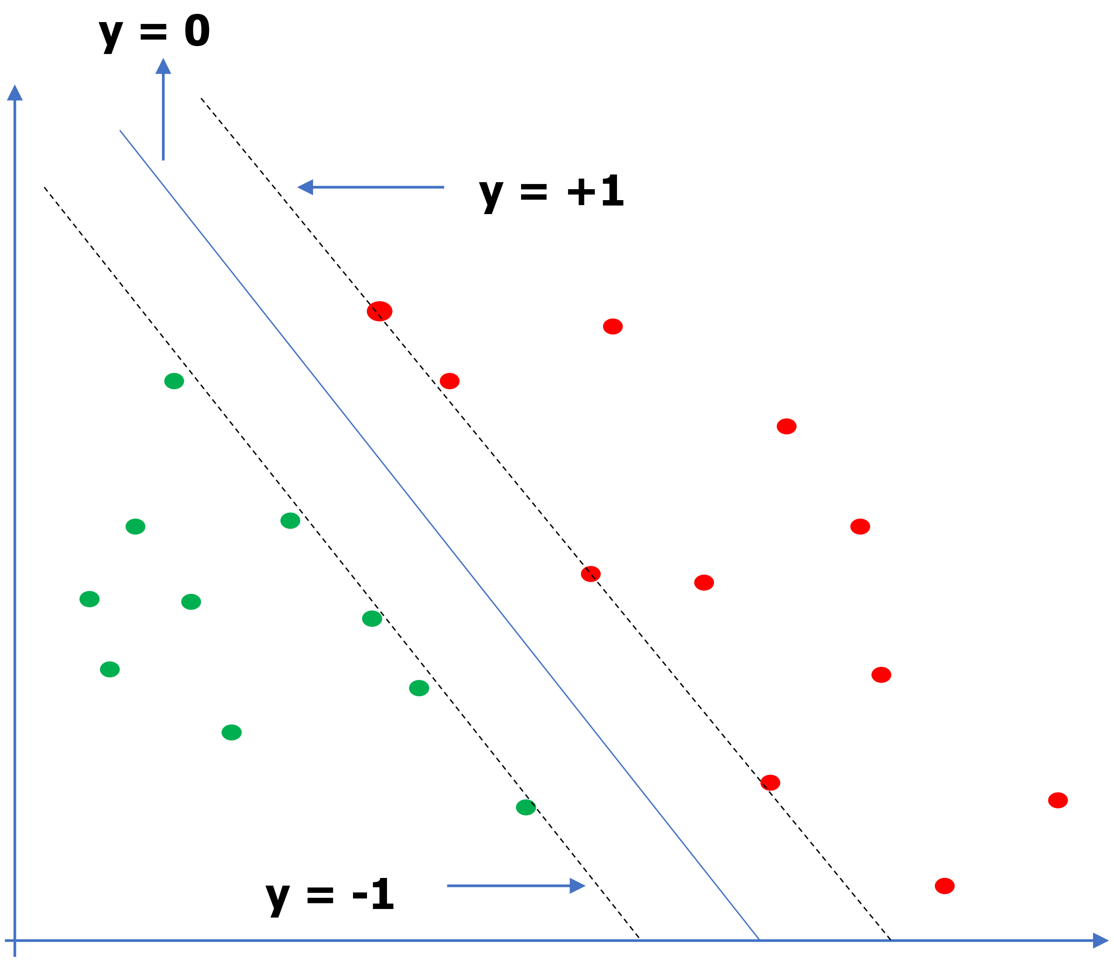
$y = w^Tx + b$

where

w: weights of the features

x: input features

b: constant



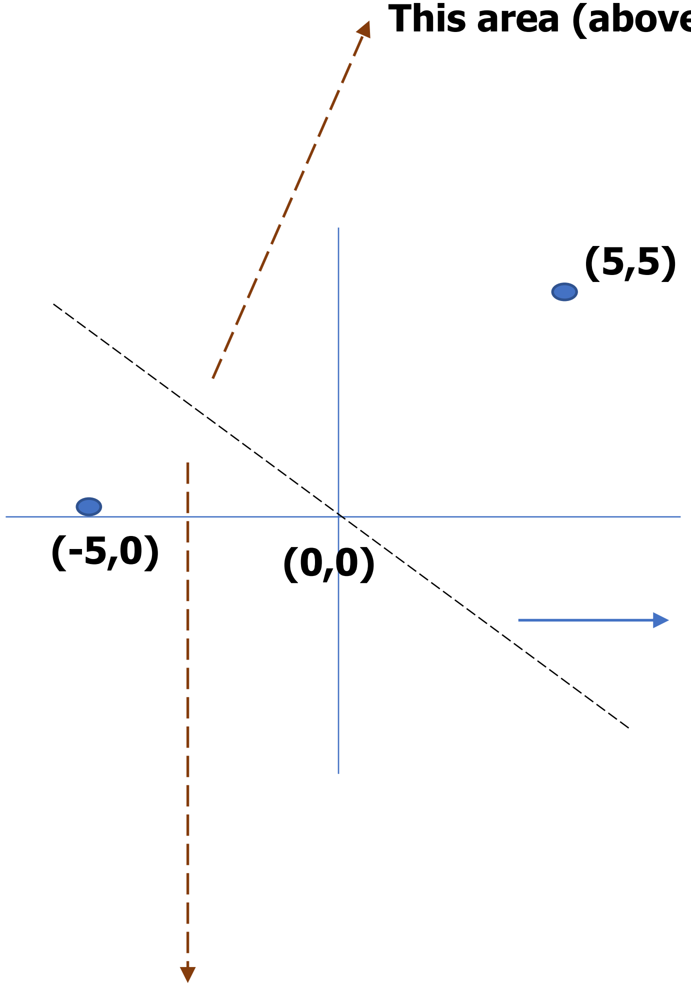
Assume m=-1
b=0 (since line passes through the origin

y = mx + b

Here, weights **(w)** are **m (-1) and b (0)**
w = [-1, 0]

Consider 2 data points
X_1 = [-5,0], X_2 = [5,5]

This area (above the line) is always -ve



$$y_2 = w^T x_2 + b = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} 5 & 0 \end{bmatrix} = -5$$

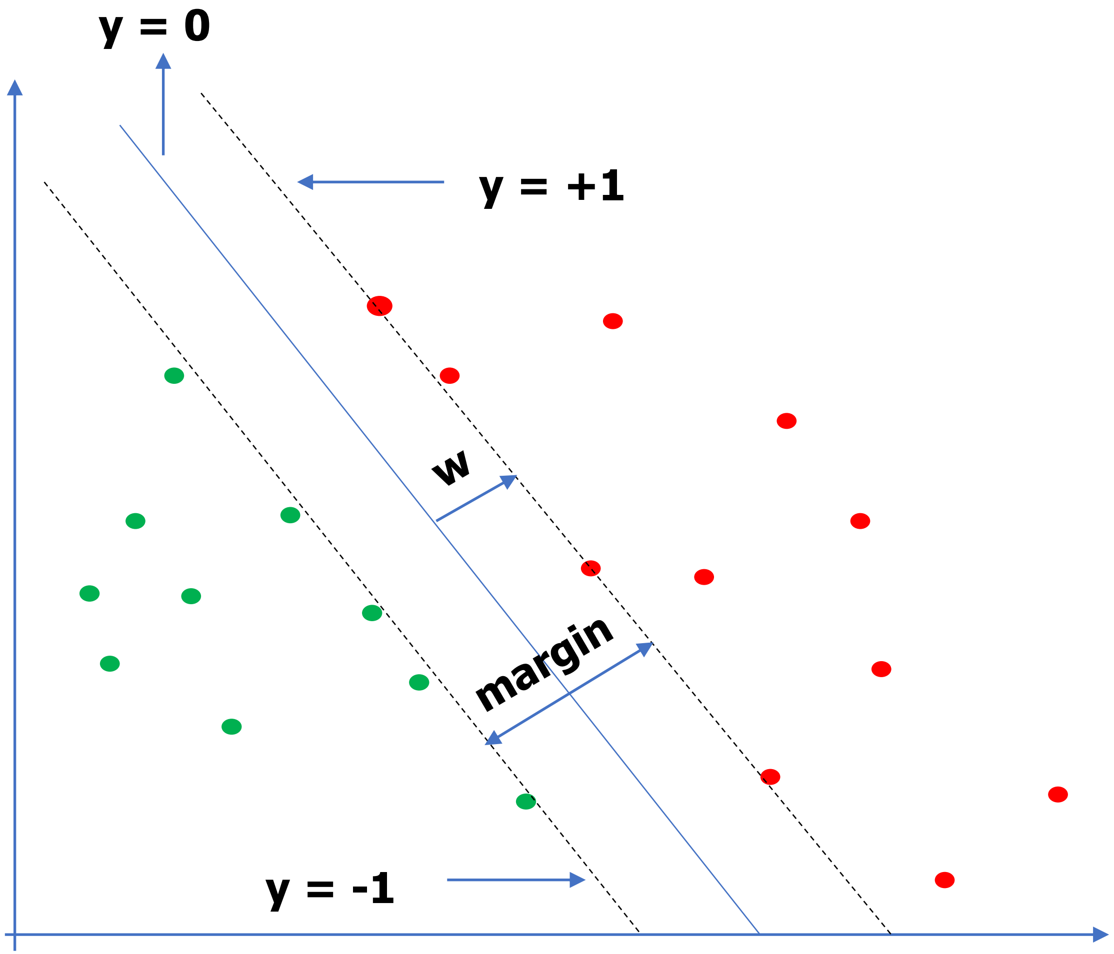
$$y_1 = w^T x_1 + b = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} -5 & 0 \end{bmatrix} = 5$$

$y = w^T x + b$

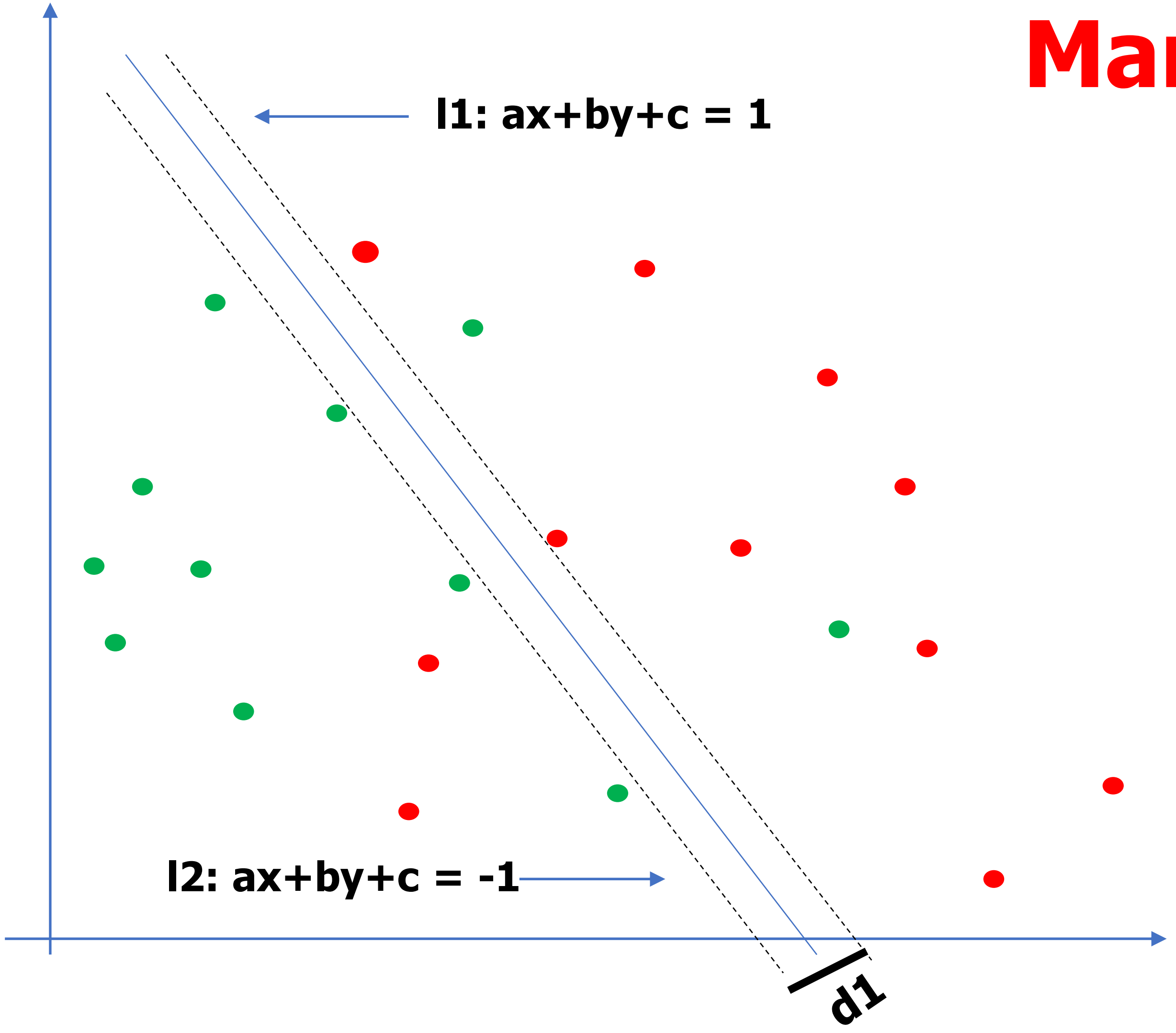
This area (under the line) is always +ve

- Vector w = (w1,w2 ….) is perpendicular to the decision boundary

- w -> normal vector
- b -> scalar



Margin Error



Let the equations of the lines be
l1: ax+by+c = 1
l2: ax+by+c = -1

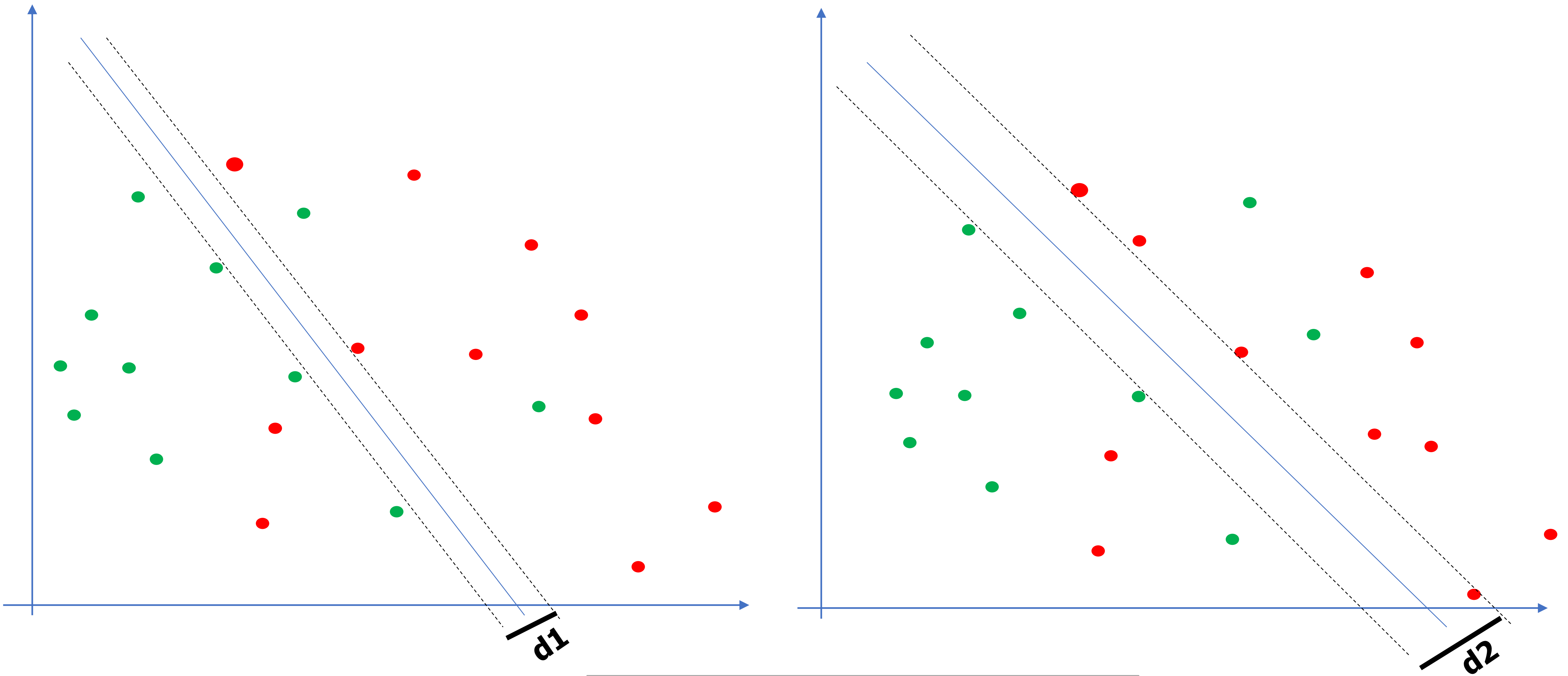
Margin (d1) = Perpendicular distance joining each line

Formula (d1) = 2 / $\sqrt{a^2 + b^2}$

Margin Error: $a^2 + b^2$

Goal of SVM is to minimise this error

Which line is better depends on
whether we need too many classification errors or
too many margin errors



Which line is better ?

Margin Error defines the best line
Larger distance (d) means less margin error; Smaller distance (d) means larger margin error

To find the **marginal distance**

$\mathbf{w^T x_1 + b = 1 \rightarrow (1)}$
$\mathbf{w^T x_2 + b = -1 \rightarrow (2)}$

Subtracting (2) from (1)
$\mathbf{w^T(x_2 - x_1) = 2}$

Dividing both sides by ||w|| (length of the normal vector)

$\mathbf{w^T (x_2 - x_1) / ||w|| = 2 / ||w||}$
SVM Optimisation function

subject to the constraint
y = +1 when $w^Tx + b >= +1$
= -1 when $w^Tx + b <= -1$

The goal is to maximise ||w||

This is done by changing the **(w,b)** values

Multiplying the constraints with the class labels, we can rewriting the condition as

Y * $w^Tx + b >= +1$
Y * $w^Tx + b >= +1$

This indicates correct classification

Rewriting the maximisation problem into a minimisation problem, we get

$\mathbf{\max 2 / ||w|| \rightarrow \min \frac{1}{2} (||w||)^2}$

Subject to the condition
Y $(w^Tx + b) -1 >= 0$

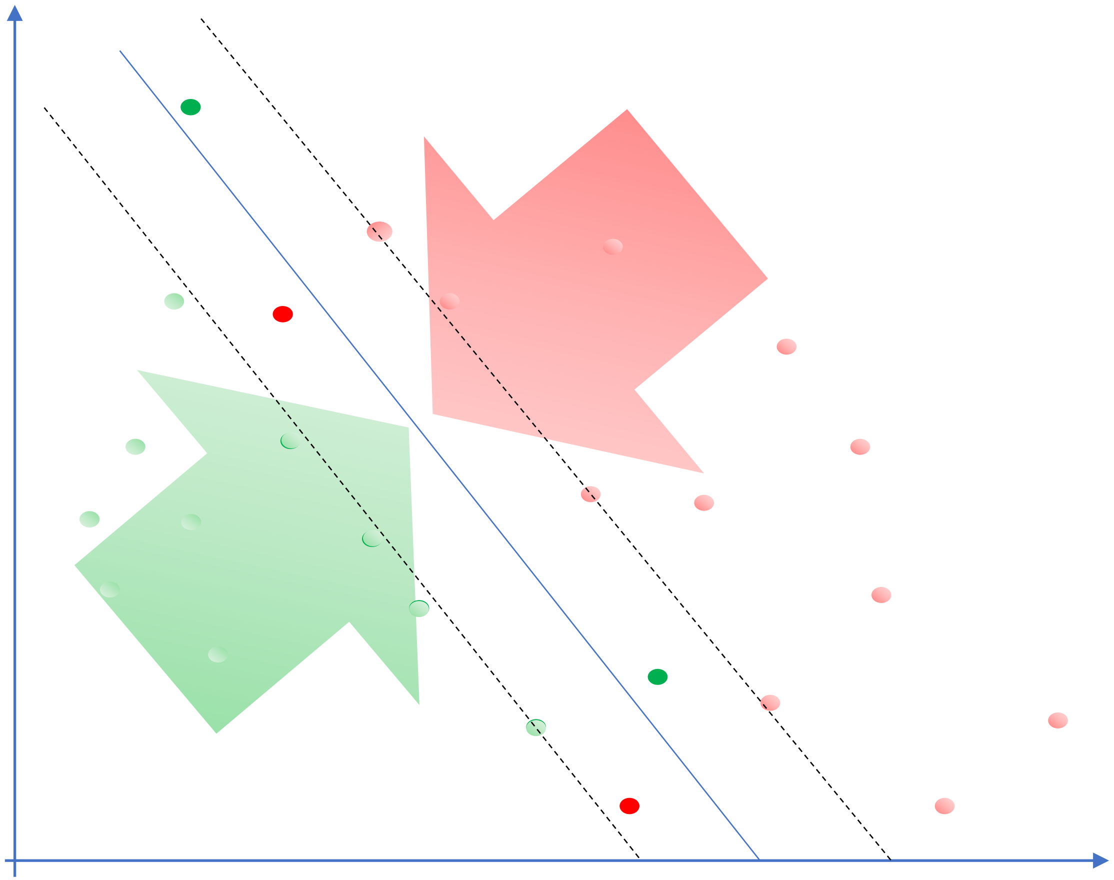
Solving this will give the value for w and b

- Real life scenario does not usually have such linearly separable points
- Prone to misclassifications and errors
- Errors doesn’t mean change the margin

Rewriting the equation, we get

$$\min \tfrac{1}{2} (\|w\|) + C \sum \zeta$$

where
C = number of errors
ζ = value of the errors



Cost Parameter

- C controls the cost of misclassification on the training data
- C → how many errors are there
- (C * Classification Error) + (Margin Error)
- Value of C
  - ✓ **Small C**
    - Cost of misclassification low ("too strict")
    - Large Margin Error
  - ✓ **Large C**
    - Cost of misclassification high and potentially overfit ("too loose")
    - Low Margin Error
- The goal is to find the balance between "not too strict" and "not too loose"
- Cross-validation and resampling are good ways to finding the best C

C and Gamma

- For linear models, only C needs to be optimized
- For RBF models, both C and Gamma parameters need to be optimized
- C and Gamma values can change based on the dataset / problem dataset
- For starters, the estimates could range from
  - C -> 0.1 – 100
  - Gamma -> 0.0001 - 10

Find the best Kernel and other parameters

Cost

- Known as the Penalty parameter **(C)**
- Controls the cost of misclassification on the training data
- High C → more data points chosen as support vectors
  - ➢ High variance : Low Bias → Overfit
- Low C → less data points chosen as support vectors
  - ➢ Low variance : High Bias → Underfit

Gamma

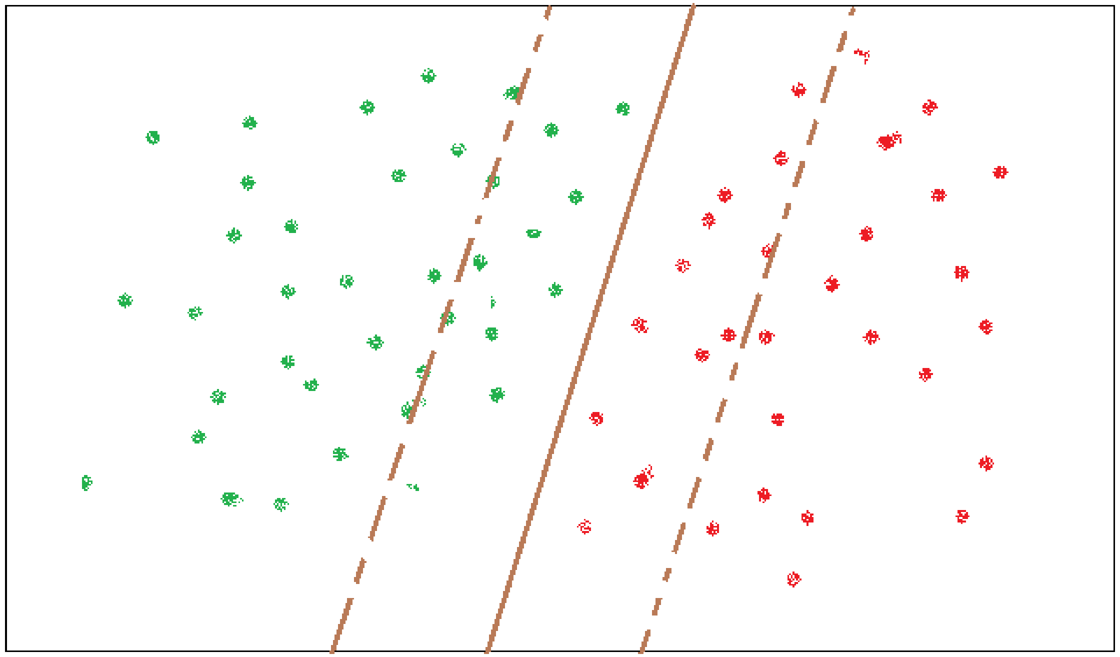
- Influence of data points on the decision boundary
- Shape of the decision boundary line depends on gamma
  - ➢ High Gamma → decision boundary depends on data points near the decision boundary
  - ➢ Low Gamma → decision boundary depends on far away points

- The goal of SVM is to find a hyperplane that would leave the widest possible "cushion" between input points from two classes.

- There is a trade-off between "narrow cushion, little / no mistakes" and "wide cushion, quite a few mistakes".

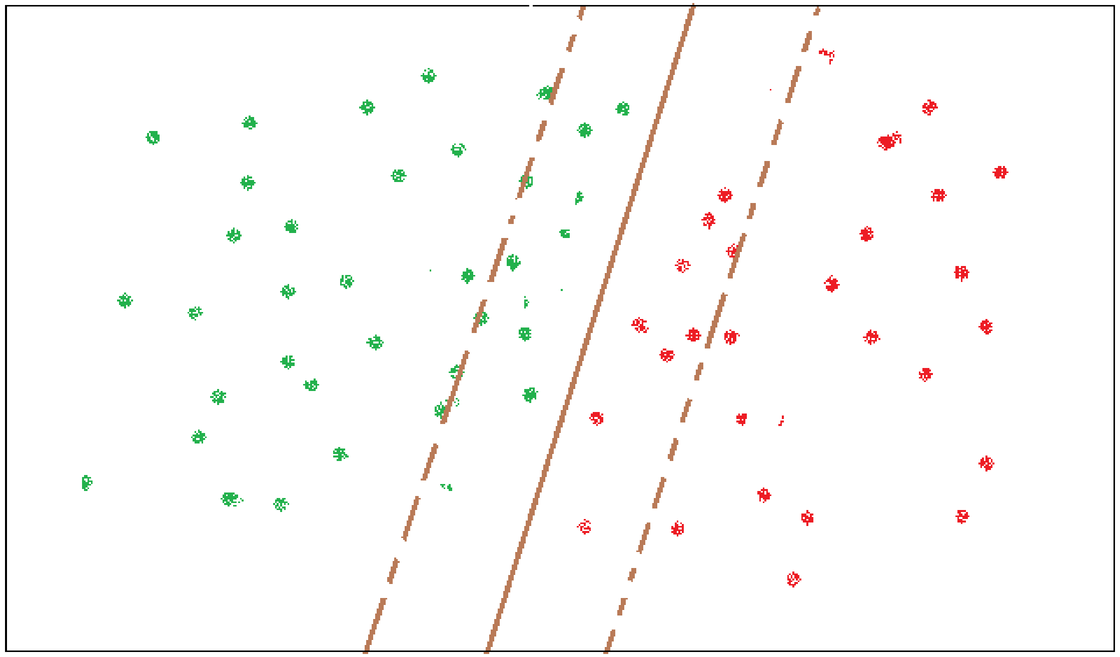
C Parameter

Predictions with C=0.1



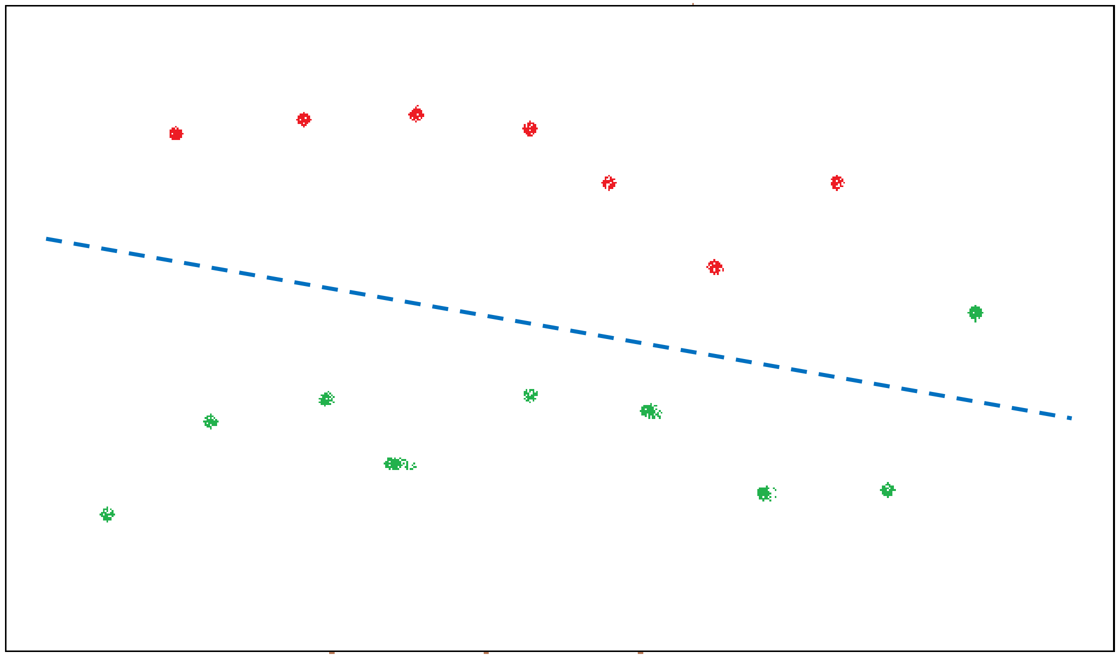
- Large margin
- More generalised model
- May have classification errors

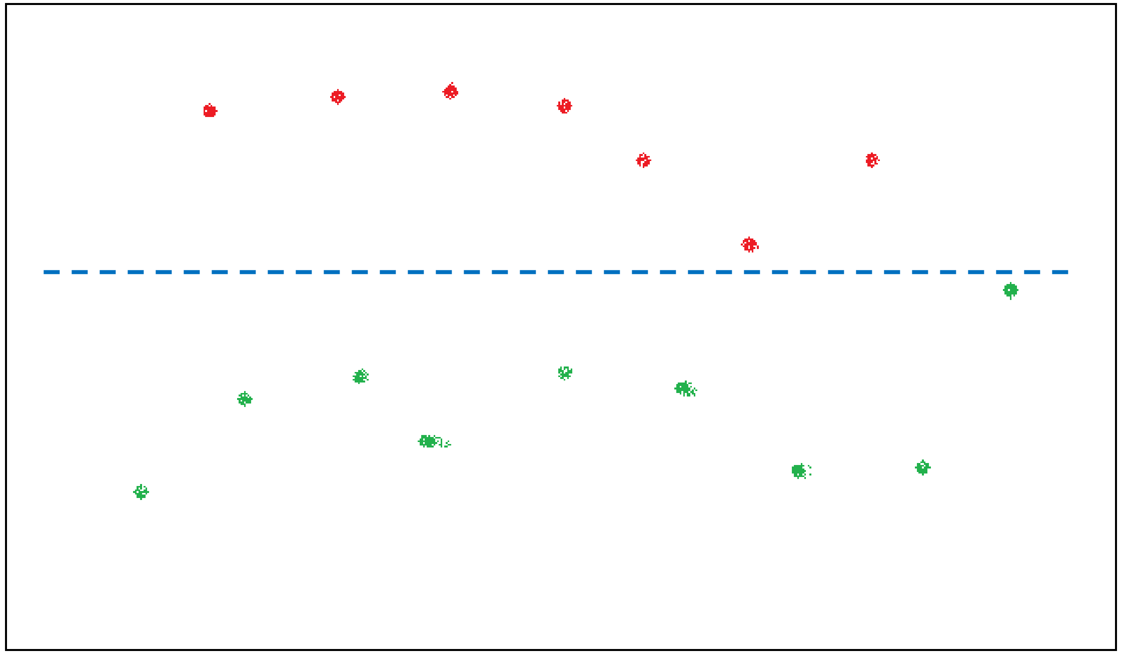
Predictions with C=100



- Smaller margin
- May not have any classification errors

- Given 2 models, assume there are 2 boundary lines
- Which separating line is better ?





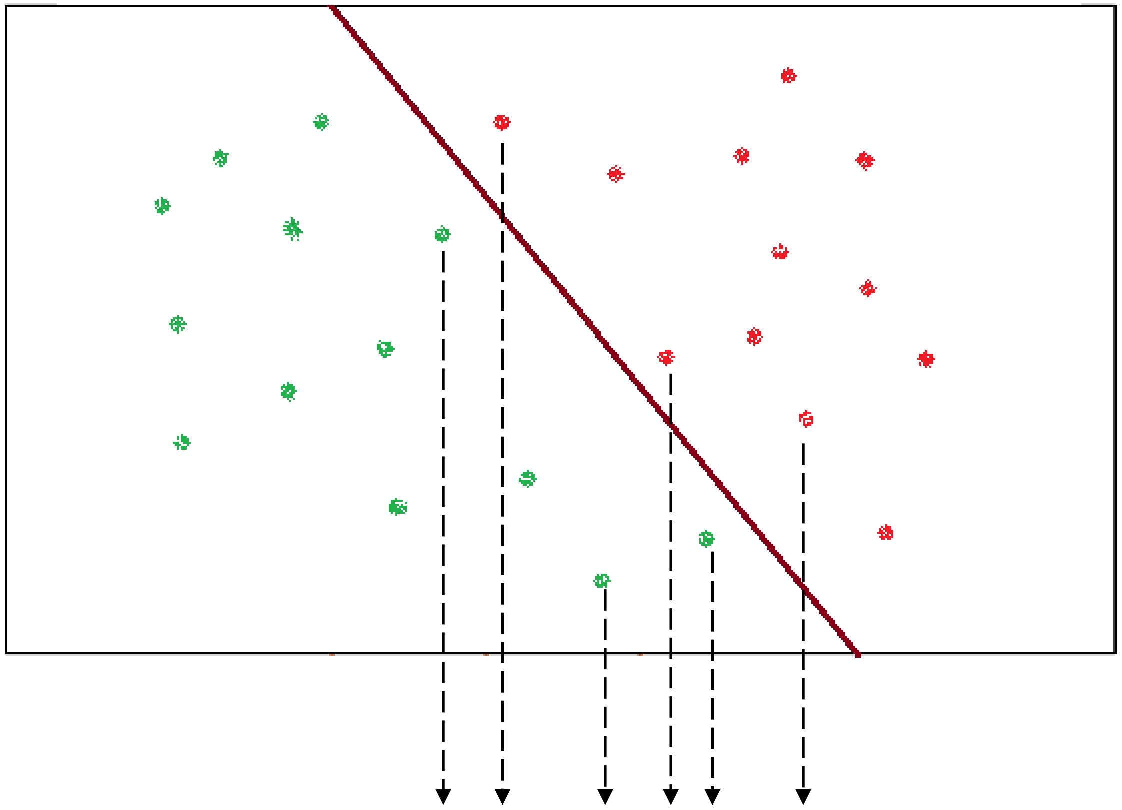
Gamma Parameter

- Used mainly with non-linear SVM (where data cannot be linearly separated due to high dimensions)

- RBF (Radial Basis Function) – most commonly used non-linear kernel in SVM

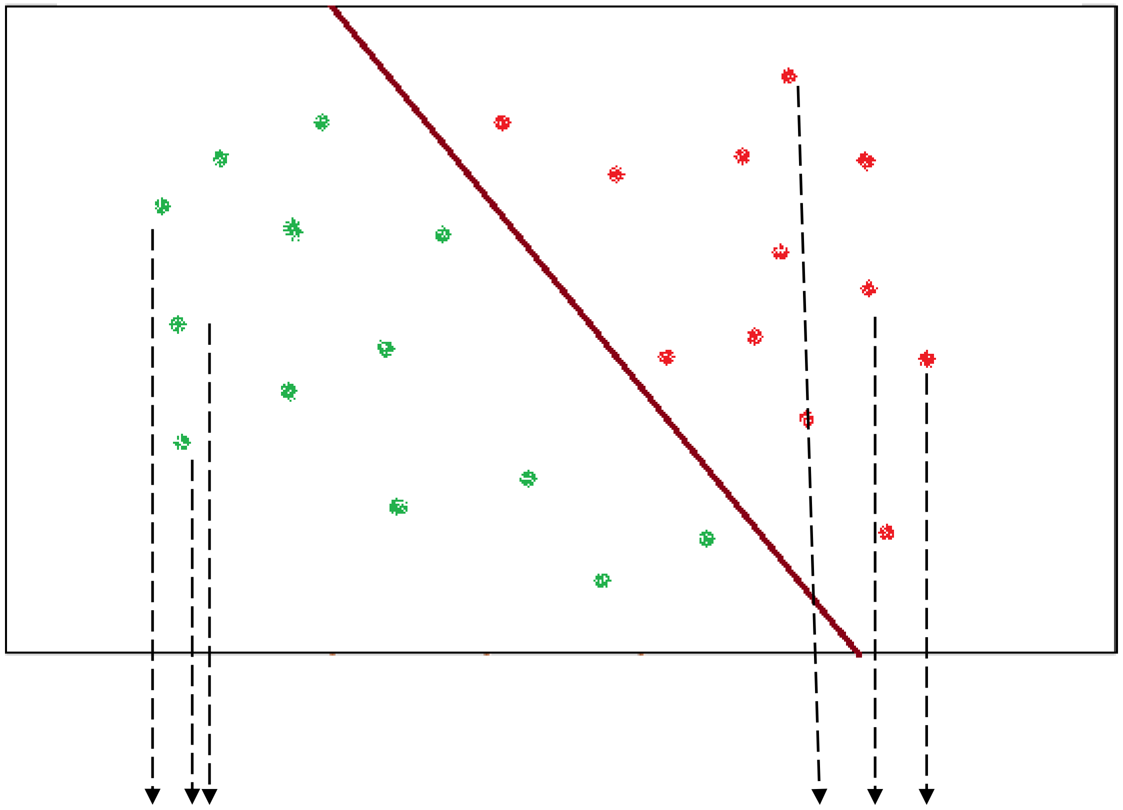
- Influence of a single training example (definition according to SVM documentation)
- Low Gamma
  - Training data has a far reach
  - Large similarity radius that groups more points

- High Gamma
  - Training data has a close reach
  - Points need to be very close to group them in the same class
  - Models with high gamma values usually overfits



- **High Gamma**

Decision boundary depends on these closest points



- **Low Gamma**

Decision boundary depends on these far away points

Find the best Kernel and other parameters

Kernel

- Kernels are mathematical functions
- Measures the similarity between 2 data points
- Sometimes, it is difficult to draw decision boundary
- This kernel technique is black-box

Kernel types

- RBF (Radial Basis Kernel Function) (observations > features)
- Linear Kernel (features > observations)

Non-Linear classification / Kernel Trick

- Uses "kernel" technique to convert non-linear classes to linear classes to fit multi-classes
  - Quite efficient in multi-class prediction

- Uses higher dimension feature space for calculation (i.e. converting non-linear separable classes to separable classes)

- SVM is popular as it works efficiently in large datasets having multi-classes

- Algorithm to arrive at an optimum hyperplane can be computationally expensive and time consuming

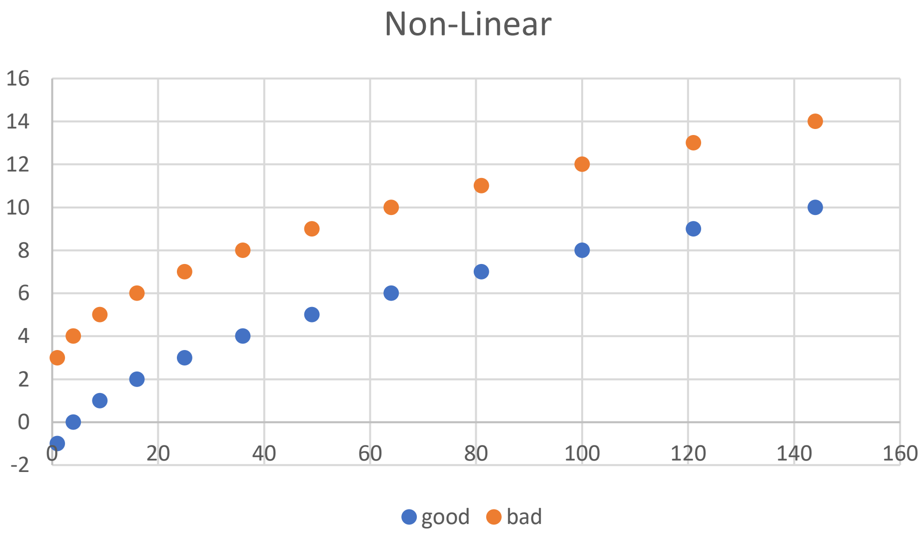
- More features and more observations complicate the algorithm

- Choice of Kernel for non-linear datasets
  - A big challenge
  - Black-box performance
  - Uses complex data transformation techniques

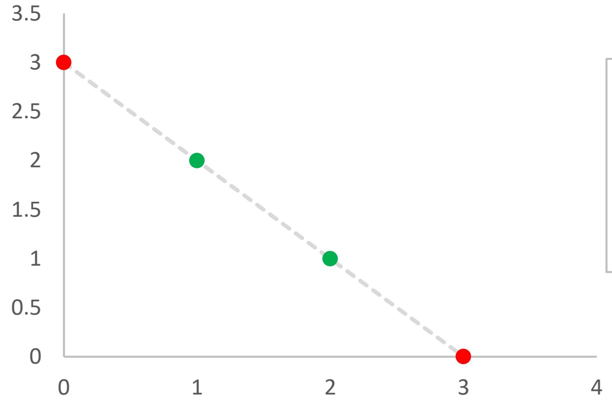
x	good	bad
1	-1	3
4	0	4
9	1	5
16	2	6
25	3	7
36	4	8
49	5	9
64	6	10
81	7	11
100	8	12
121	9	13
144	10	14

Square root of x

x	good	bad
1	-1	3
2	0	4
3	1	5
4	2	6
5	3	7
6	4	8
7	5	9
8	6	10
9	7	11
10	8	12
11	9	13
12	10	14



x	y
0	3
1	2
2	1
3	0

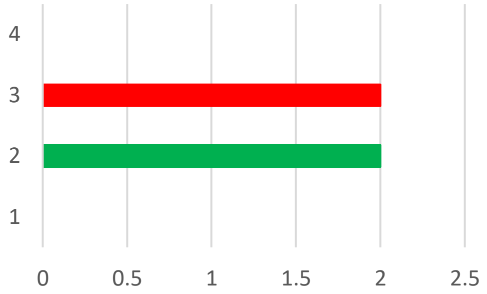


How to separate the Red Class from Green ?

transformation →

x	y	x+y	x*y	x^2
0	3	3	0	0
1	2	3	2	1
2	1	3	2	4
3	0	3	0	9

x * y



x	y	x*y	(x,y,x*y)
0	3	0	(0,3,0)
1	2	2	(1,2,2)
2	1	2	(2,1,2)
3	0	0	(3,0,0)