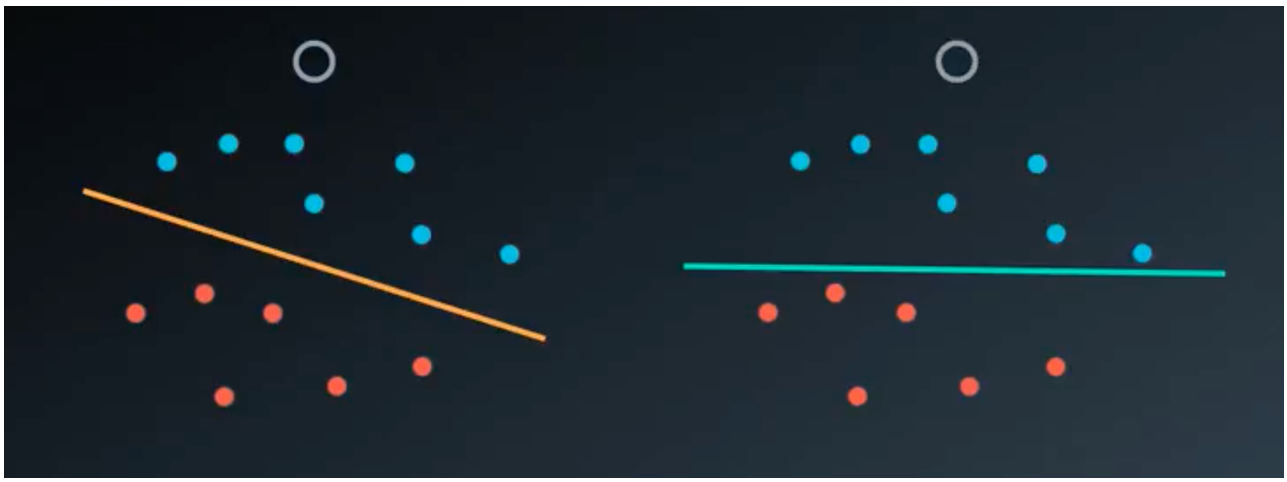
# 5-Supervised Classification Models -Support Vector Machines

**By:** Hamad Albgaie
Eng. Esraa Madhi

## What is Support Vector Machines (SVMs)?

- Support Vector Machines (SVMs) is a powerful algorithm for classification which **also finds the best decision boundaries** (the surface or line that separates different groups of data points in the feature space)



### What are the F1 scores for both separating lines? Which line performs better?

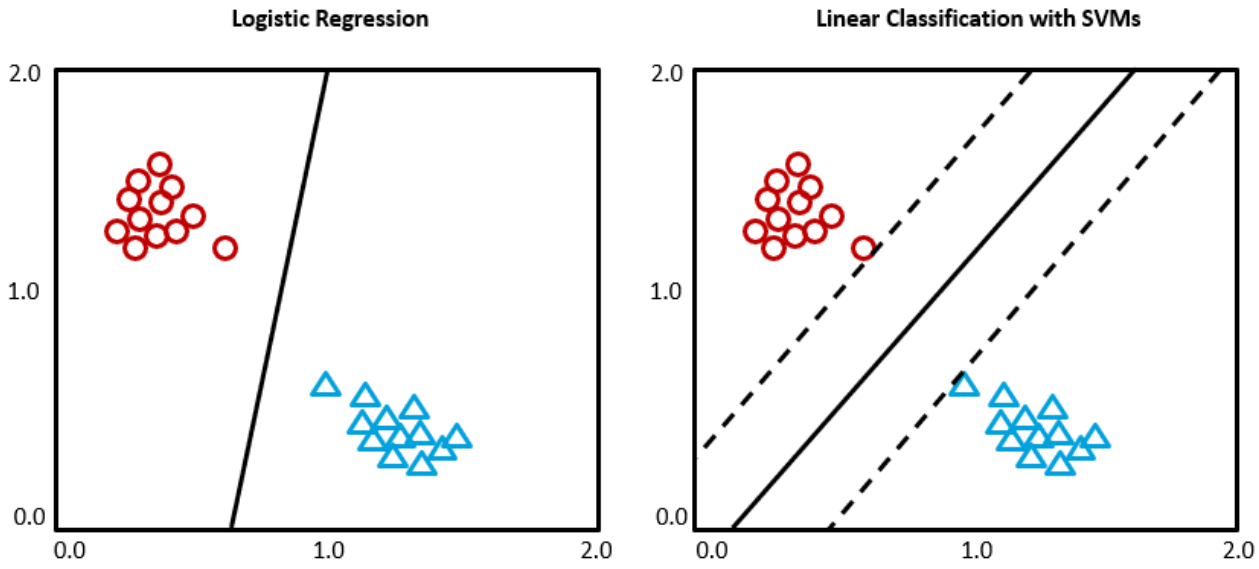
- Both have F1 score as 100%, But the **yellow line** is the best line, **Why?** As you can see, the yellow line **is pretty well-spaced** between the blue points and the red points. The green line is not, because it's **a little too close to some of the points on both sides**.

- **Why the distance is matter?** More distance = more sure to predict new data.

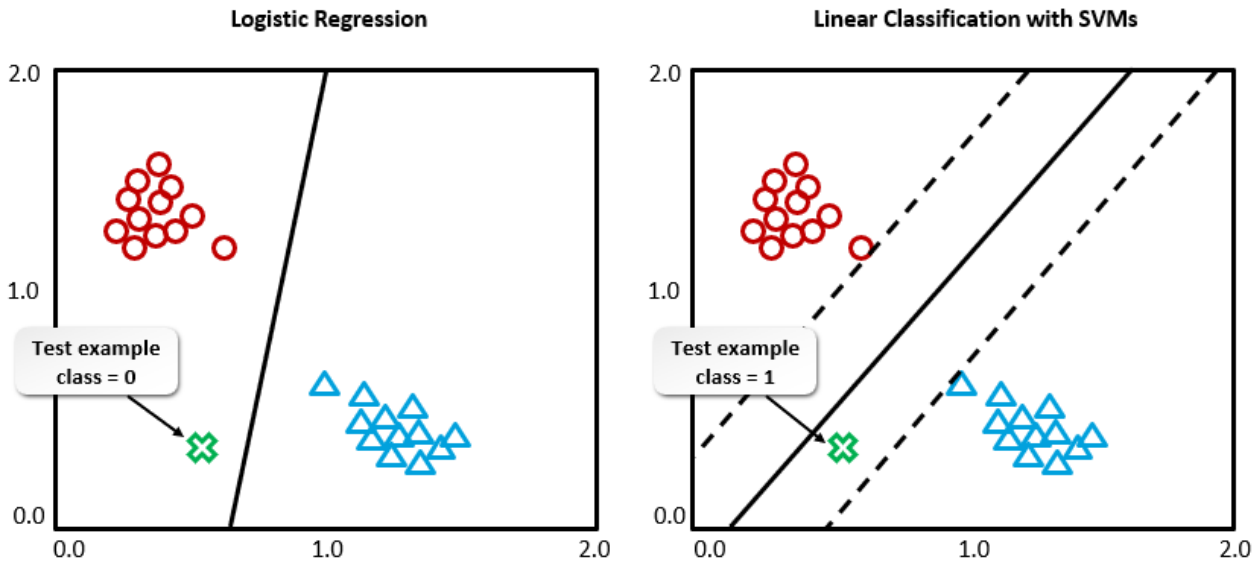
Look at all the distances between the closest point to one of the lines and compare them to the distances between the closest points to the other line. Then, we check which one of these closest points is actually farther from the line. This is the SVM.

## Isn't that typically how logistic regression establishes its decision boundary?

**Below, both models are built on the same data:**




**Let's try to predict unseen data point using the two models:**




SVMs are designed to find the best separating line (or hyperplane) that correctly classifies data into their respective categories, similar to logistic regression. However, SVMs also focus on **maximizing the margin,** which is the distance from the nearest data points, by applying two key techniques:

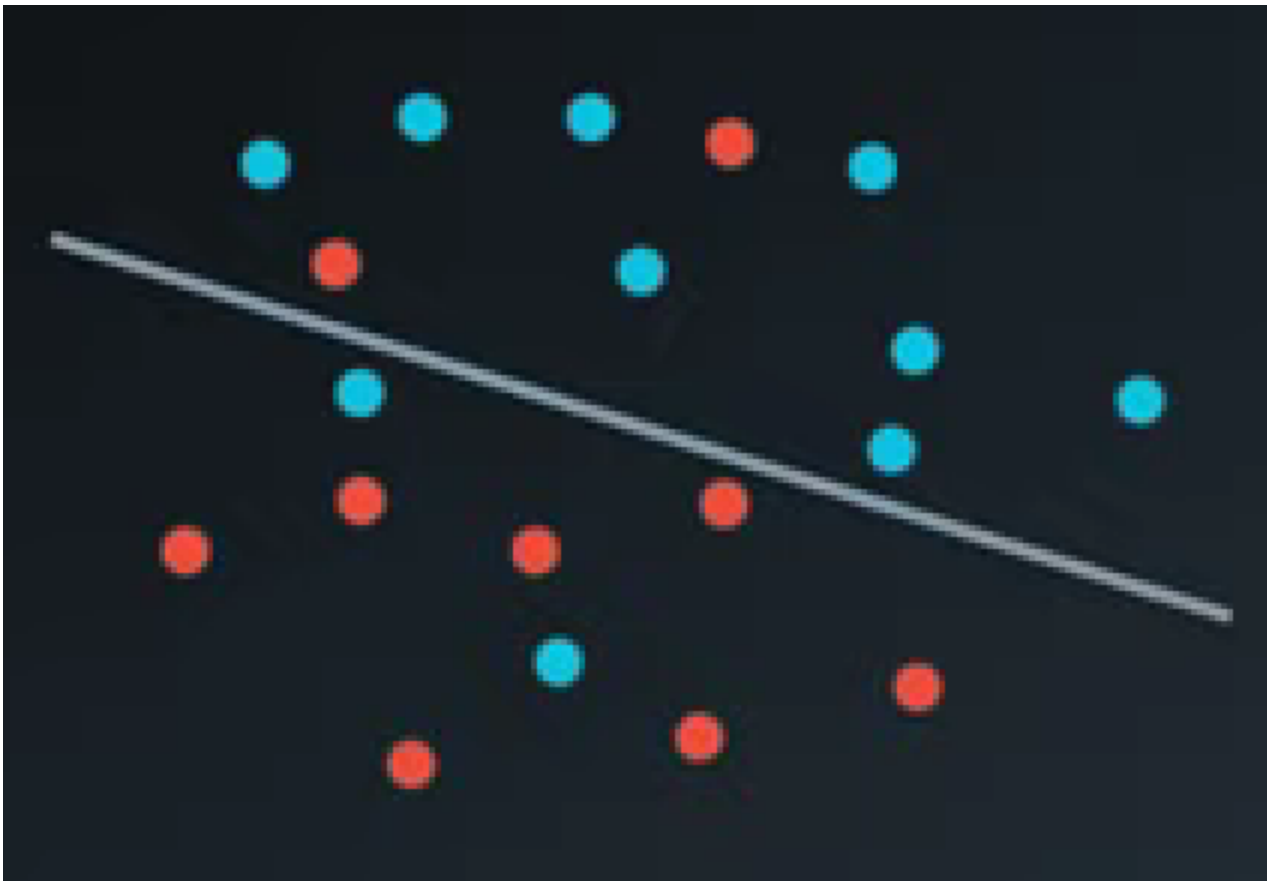
a. Adjusting the **Error function** to penalize smaller margins.
b. Utilizing only **the support vectors (nearest data points)** to define the decision boundary.

**In this way, SVM enhances the efficiency of predicting unseen data points.**

---

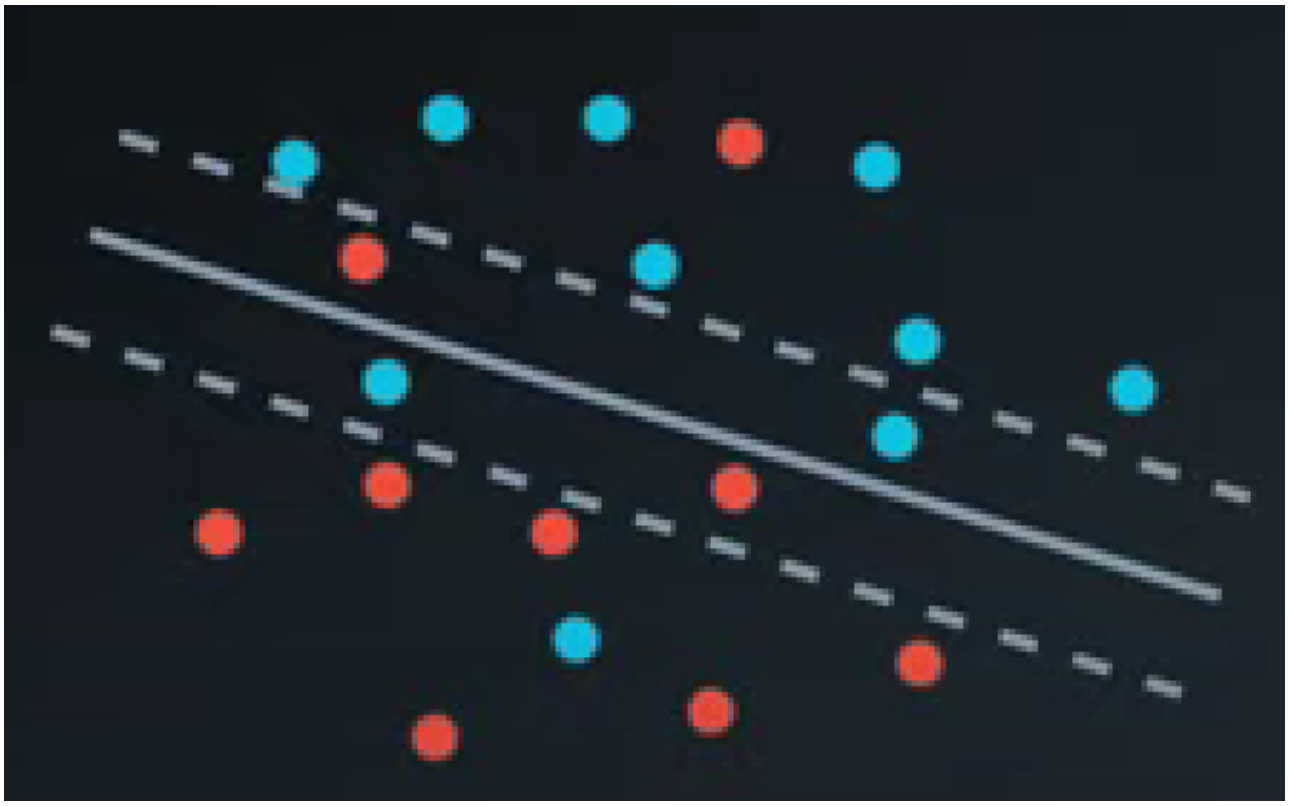
## Let's see SVM in Action:

Any classification method will split the dataset in this way



But SVM will split them in a different way:

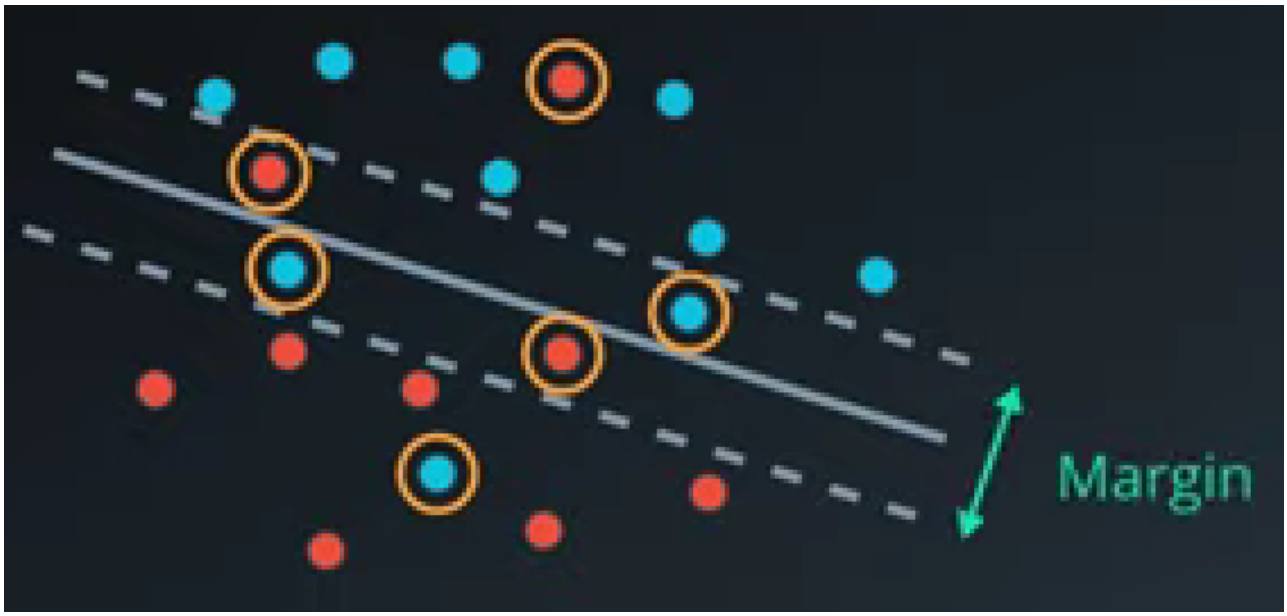
- At a high level, SVM is splitting the data with a line, and add two more lines, one in the class 0 area, and the other one in the class 1 area.



- Support Vectors: are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.

- Margin: The distance between the hyperplane and the nearest data point from either set.

But what about the misclassified points? that the data inside the margin? **We make them part of our classification error.**




**Our Goal:**
**We aim to identify the optimal line that separates two classes, minimizing the number of misclassified points and maximizing the margin between them.**

---

## SVM in Math:

Suppose we have the following dataset with two features (x1, x2) and two classes (0 and 1):

| x1 | x2 | y |
|---|---|---|
| 1 | 3 | 0 |
| 2 | 2 | 0 |
| 3 | 1 | 1 |
| 4 | 3 | 1 |

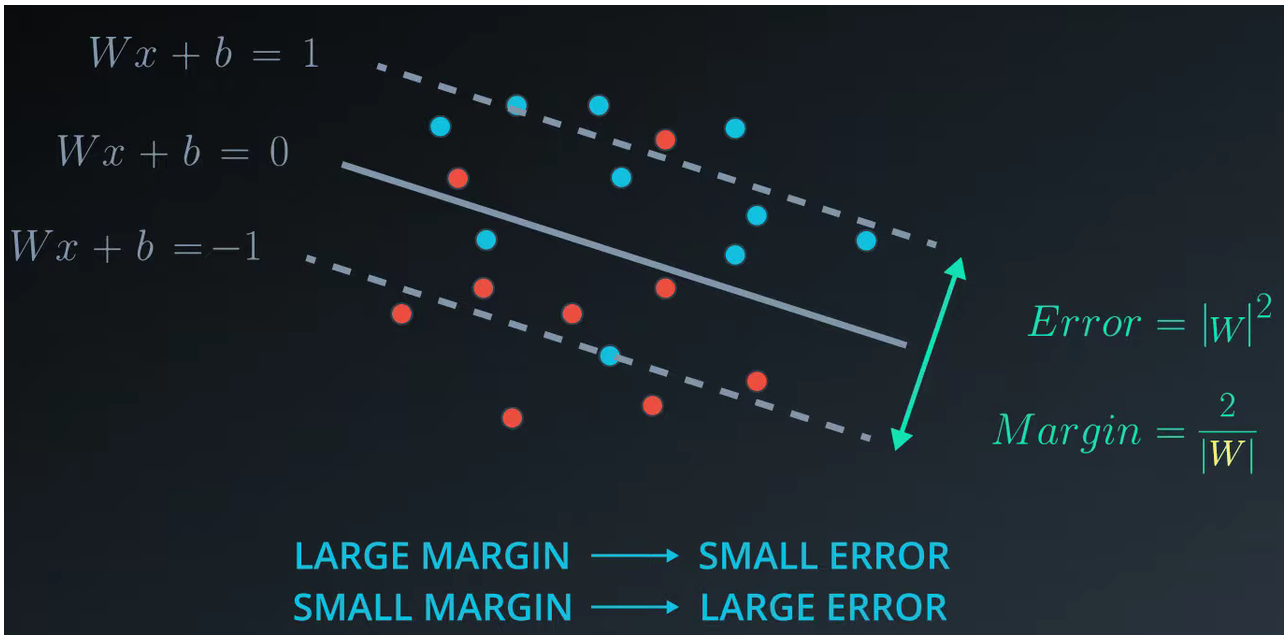
For binary classification using SVM, we will take few assumptions that we relabel our classes as (+1, -1)

| x1 | x2 | y |
|---|---|---|
| 1 | 3 | +1 |
| 2 | 2 | +1 |

| | | |
|---|---|---|
| 3 | 1 | -1 |
| 4 | 3 | -1 |

In this way, boundary line (hyperplane):
$\mathbf{w} \cdot \mathbf{x} + b = 0$
and other 2 lines:
$\mathbf{w} \cdot \mathbf{x} + b = 1$ , lies in the area of class 1
$\mathbf{w} \cdot \mathbf{x} + b = -1$ , lies in the area of class -1

- **On the hyperplane:** For any point **x** lying exactly on the hyperplane, the equation equals zero. This reflects a position on the boundary between classes where the algorithm is exactly uncertain whether **x** belongs to one class or the other.
- **Above the hyperplane:** For points where $\mathbf{w} \cdot \mathbf{x} + b >= 0$, these are classified into one class (often conventionally positive, or "+1").
- **Below the hyperplane:** For points where$\mathbf{w} \cdot \mathbf{x} + b < 0$, these are classified into the opposite class (often conventionally negative, or "-1").




In the equation $z = \mathbf{w} \cdot \mathbf{x} + b$ , **(The weight vector components $b_1, b_2$ and the bias term $b_0$**

**are initially assigned random values).** After that, we will use **gradient decent and partial derivative of loss function to find best parameters** for boundary line $z = \mathbf{w} \cdot \mathbf{x} + b$ (get the change amount of $(b_0, b_1, b_2)$)

$$\text{Error Function} = \text{misclassification error} + \text{margin error}$$

$$\text{margin error} = ||W||^2$$

$$\text{misclassification error (hinge loss)} = C \sum_{i=1}^{n} L(y_i, z_i)$$
$$= (\max(0, 1 - y_i z_i))$$

$$\text{Error Function} = C \sum_{i=1}^{n} L(y_i, z_i) + ||W||^2$$

**Where:**

- ( C ) is the regularization parameter that controls the trade-off between achieving a low hinge loss and keeping the model weights small to avoid overfitting.
- $(L(y_i, z_i))$ is the hinge loss for each individual training example ( $(x_i, y_i)$ )
  - $y_i$ is the true label of the data point, typically +1 or -1.
  - $z_i$ is the prediction of the model for the input *x*. the output of the decision function $\mathbf{w} \cdot \mathbf{x} + b$
- $(||W||^2)$ is the ( L2 ) norm of the weight vector, which is used for regularization.
- ( n ) is the number of training samples.

**Calculation Example:**

| x1 | x2 | y |
|---|---|---|
| 1 | 3 | +1 |
| 2 | 2 | +1 |
| 3 | 1 | -1 |
| 4 | 3 | -1 |

1. Assume we have finished training SVM to fit the best decision boundary the we got ( W = [2, -3] ) ( b = 0.5 )

2. let's try to predict the above target $y$
3. Calculate: $z = 2x_1 - 3x_2 + 0.5$

| x1 | x2 | y | $z$ |
|---|---|---|---|
| 1 | 3 | +1 | -6.5 |
| 2 | 2 | +1 | -1.5 |
| 3 | 1 | -1 | 3.5 |
| 4 | 3 | -1 | -0.5 |

4. $y =$
    - If $(z_i > 0)$, then $(y = +1)$.
    - If $(z_i < 0)$, then $(y = -1)$.
    - If $(z_i = 0)$, typically means the point is exactly on the decision boundary (though how this is handled can depend on the specific implementation and context.

| x1 | x2 | y | $z$ | $\hat{y}$ |
|---|---|---|---|---|
| 1 | 3 | +1 | -6.5 | -1 |
| 2 | 2 | +1 | -1.5 | -1 |
| 3 | 1 | -1 | 3.5 | 1 |
| 4 | 3 | -1 | -0.5 | -1 |

5. $\text{Accuracy} = \frac{1}{4} = 0.25$

**Side exercise: calculate error function for the above example**

$$\text{Error Function} = \text{misclassification error} + \text{margin error}$$

$$\text{Error Function} = C \sum_{i=1}^{n} L(y_i, z_i) + ||W||^2$$

- ( C = 1 ) (Hyperparameter, C should be tuned using cross validation)
- $||W||^2 = 2^2 + (-3)^2 = 4 + 9 = 13$
- $\sum_{i=1}^{n} L(y_i, z_i) = (\max(0, 1 - y_i z_i)$

| x1 | x2 | y | z | Hinge Loss |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | 3 | +1 | -6.5 | 7.5 |
| 2 | 2 | +1 | -1.5 | 2.5 |
| 3 | 1 | -1 | 3.5 | 0 |
| 4 | 3 | -1 | -0.5 | 0.5 |

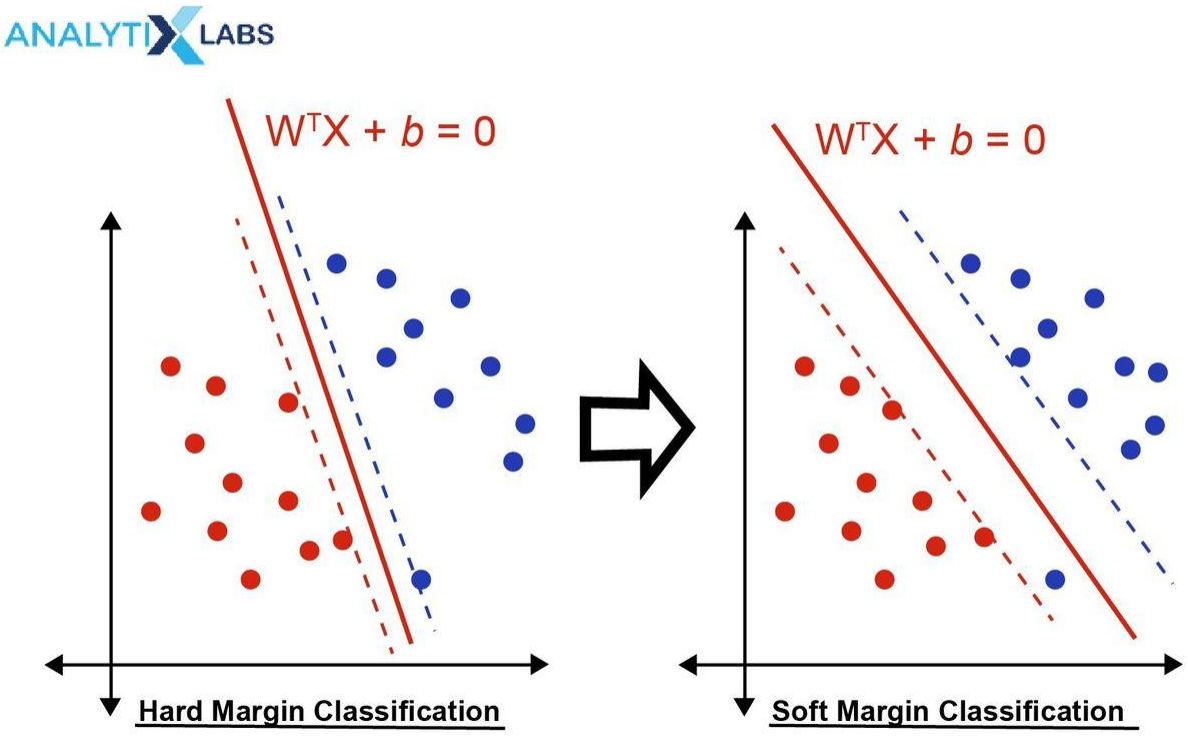
Total Hinge Loss = $\sum_{i=1}^{4} L(y_i, z_i) = 7.5 + 2.5 + 0 + 0.5 = 10.5$

5. Error function $= 1 \cdot 10.5 + 13 = 23.5$

**What is the range of $C$ parameter?**

1. **Hard Margin:**
In this case, the goal is to find a hyperplane that completely separates the data without any violations (i.e., no points are misclassified or within the margin). Here, *C* is typically set to a very high value (theoretically *C*=∞). In practical implementations, you might not be able to set *C* to infinity, but setting it to a very high value would force the optimization to prioritize maximizing the margin with no violations.

2. **Soft Margin:**
For a **soft margin**, some misclassifications are allowed to enable the model to generalize better on unseen data, especially when the data is not linearly separable. In this case, *C* is set to a finite,




https://youtu.be/_YPScrckx28

https://youtu.be/_YPScrckx28

---

## What if we have non linear problem?

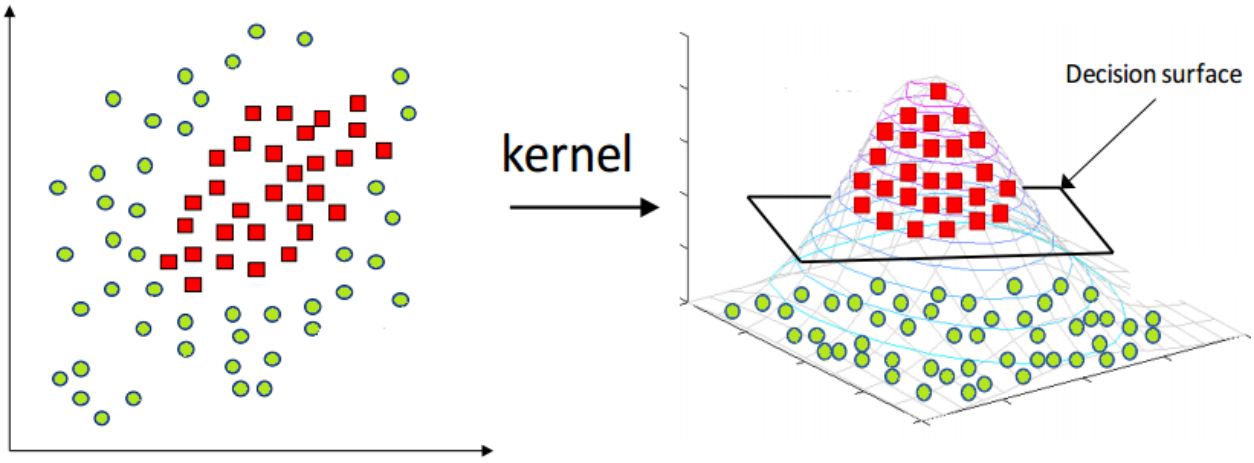
https://youtu.be/Q7vT0--5VII

https://youtu.be/Q7vT0--5VII

## 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).




https://youtu.be/3liCbRZPrZA

https://youtu.be/3liCbRZPrZA

**How can we split the classes?** Using some advanced techniques like **kernel tricks** to classify them.

It is a group of mathematical methods for mapping the data to a higher-dimensional feature space to efficiently representing non-linearly separable data in higher-dimensional space.

In the example above, we have 2D-space that is non-linear. We can convert it into a 3D-Space. Thus we put the points of class zero below, and the points of class 1 above.

## Kernels Types:

1. Linear Kernel
2. Polynomial Kernel
3. Radial Basis Function (RBF) Kernel or Gaussian Kernel

**Choosing a Kernel Function**
Choosing the right kernel function and its parameters is crucial for the performance of an SVM model. Here are some guidelines on how to select a kernel:

- **Data Linearity**: Start with a linear kernel, especially if the number of features is high and the data is suspected to be linearly separable. It's the simplest and often used when there are many features.
- **Non-linearity in Data**: If the data is not linearly separable, consider non-linear kernels. The RBF kernel is often a good first choice because it has fewer hyperparameters (just ( \gamma )) and can model complex boundaries.
- **Polynomial Relationships**: If there is a reason to believe that the relationship between the features is polynomial, testing a polynomial kernel with different degrees can be insightful.
- **Performance and Overfitting**: High-dimensional kernels like RBF and polynomial can lead to overfitting, especially with small training datasets or when chosen parameters (like ( \gamma ) and ( d )) are extreme.
- **Cross-Validation**: Use cross-validation to empirically test which kernel and parameter settings (like $(C)$, $(\gamma)$, $(d)$, $(r)$) work best for your specific dataset.
- **Computational Resources**: Non-linear kernels are generally more computationally intensive than linear kernels. If you are working with a very large dataset and runtime is a concern, starting with a linear kernel or using dimensionality reduction techniques before applying SVM might be advisable.

---

**Pros**

- Accuracy
- Works well on smaller cleaner datasets

- It can be more efficient because it uses a subset of training points

**Cons**

- Isn’t suited to larger datasets as the training time with SVMs can be high
- Less effective on noisier datasets with overlapping classes