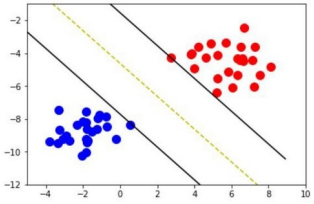
Support Vector Machine



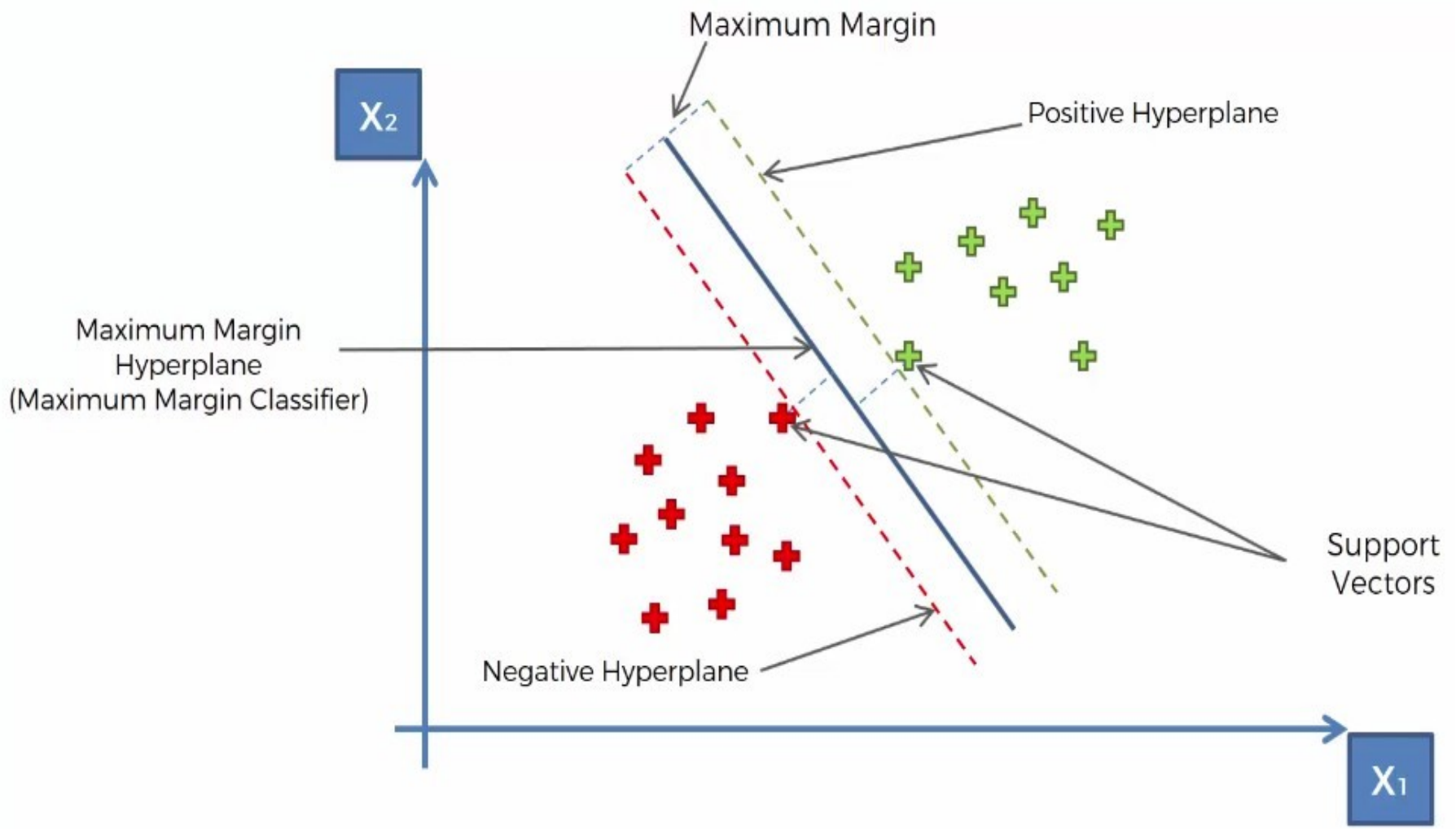
What is support vector?

- “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.
- In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.
- Then, we perform classification by finding the hyperplane that differentiate the two classes very well.

Support Vector Machine

- Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems.
- It can easily handle multiple continuous and categorical variables.
- SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error.
- The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.

Decision Vectors



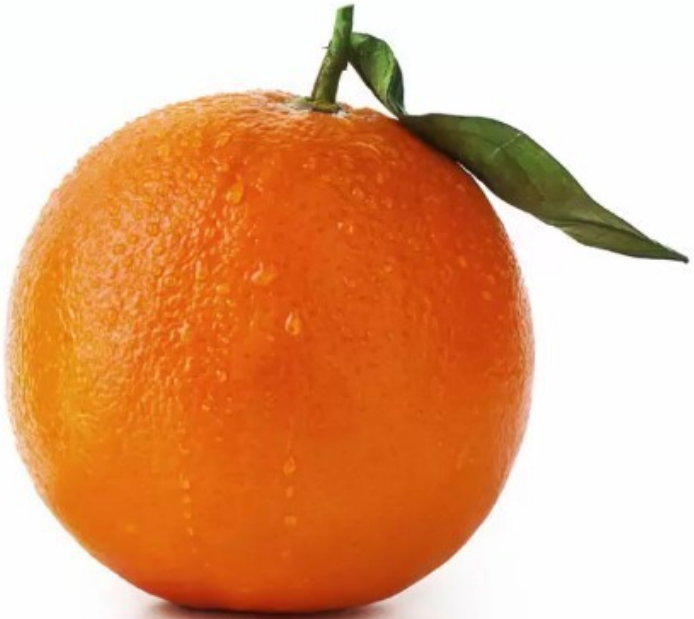
Definitions

- Support Vectors
  - Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.
- Hyperplane
  - A hyperplane is a decision plane which separates between a set of objects having different class memberships.

Definitions

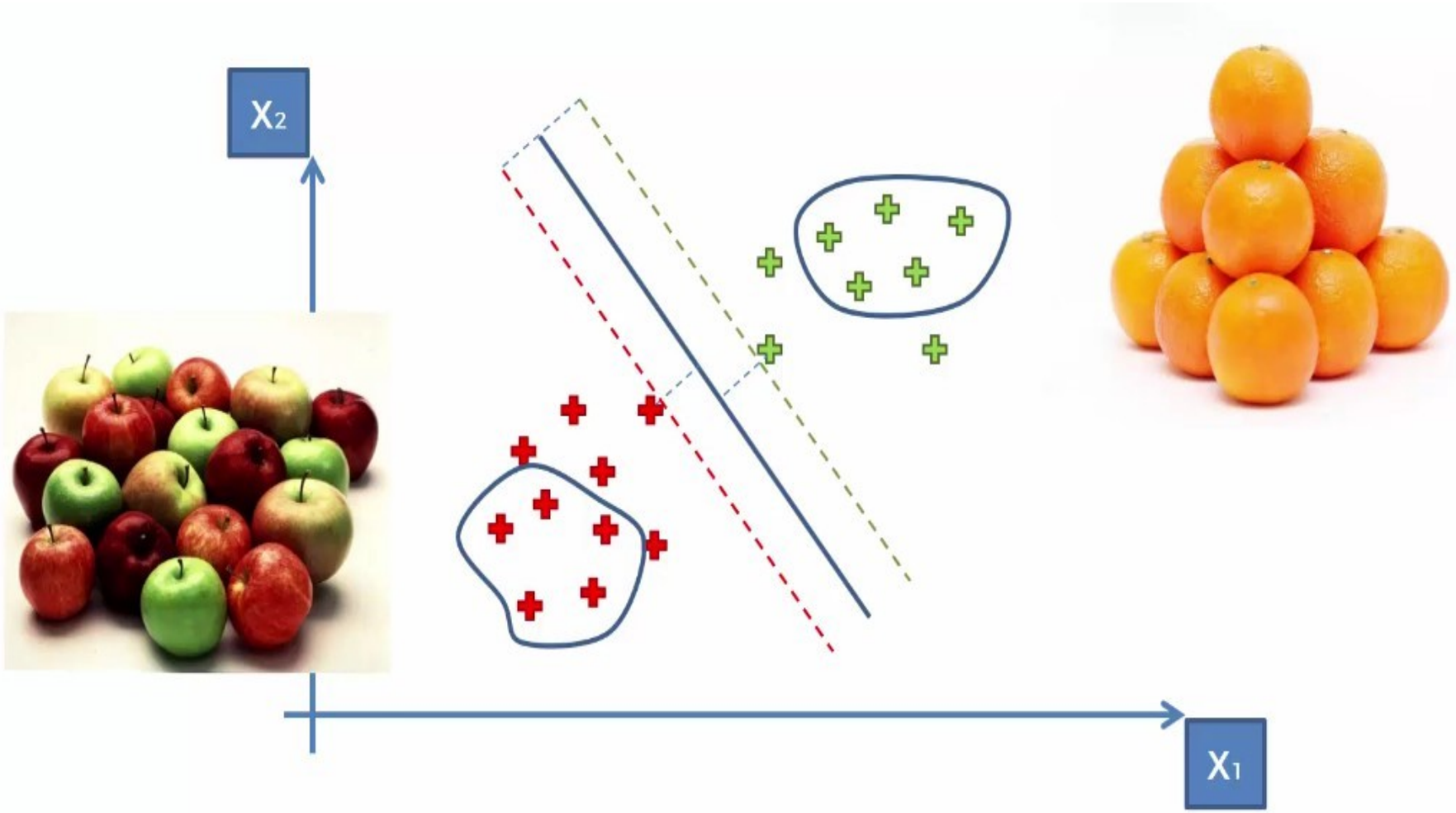
- Margin
  - A margin is a gap between the two lines on the closest class points.
  - This is calculated as the perpendicular distance from the line to support vectors or closest points.
  - If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

Why SVM is so special ?

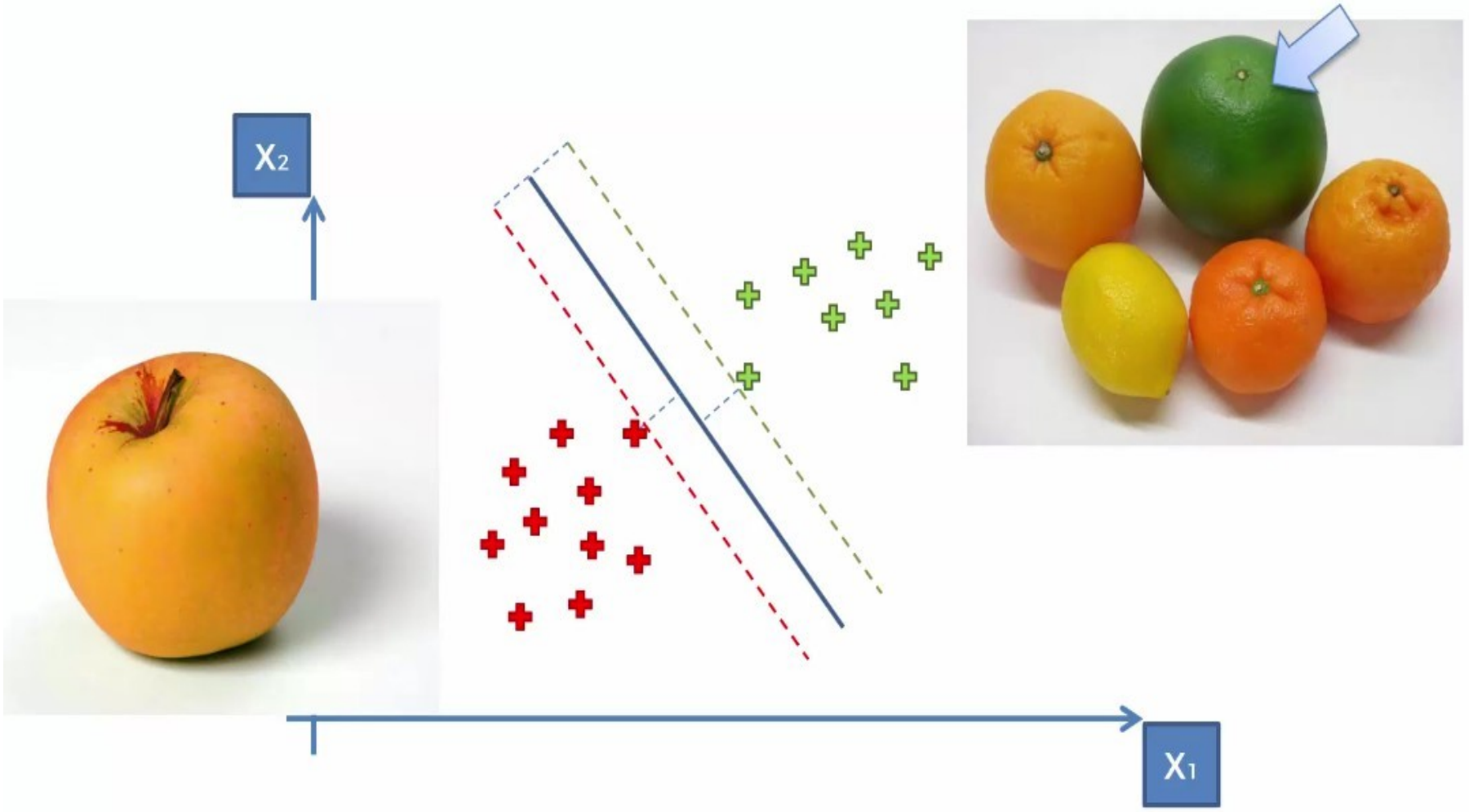


Example Reference: Super Data Science

How SVM works for this?



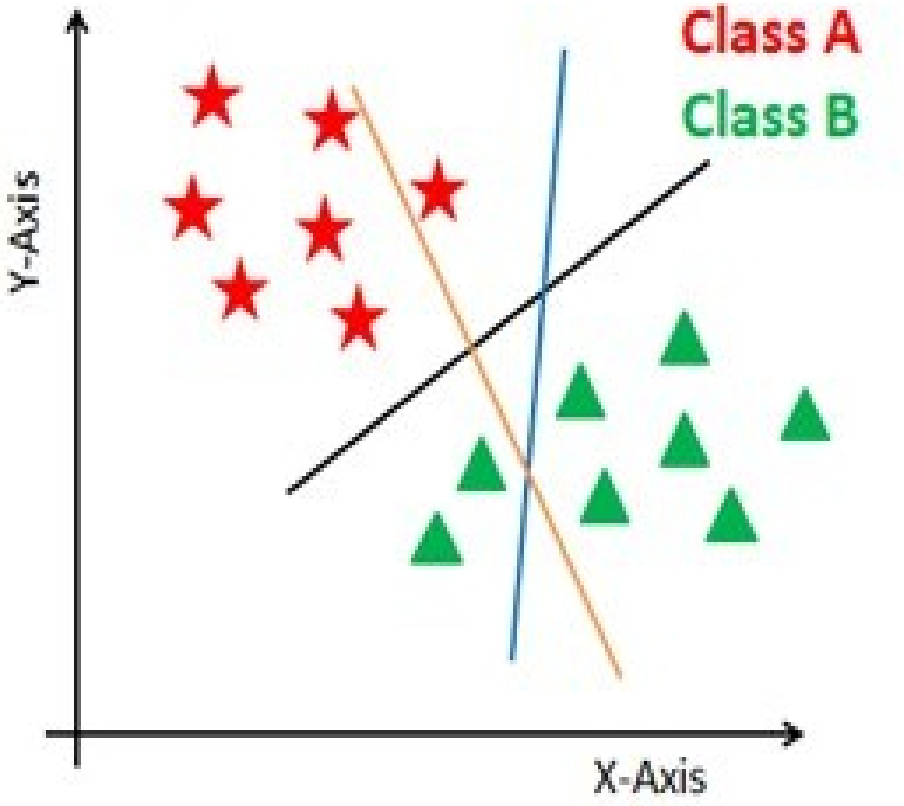
How classification will work ?

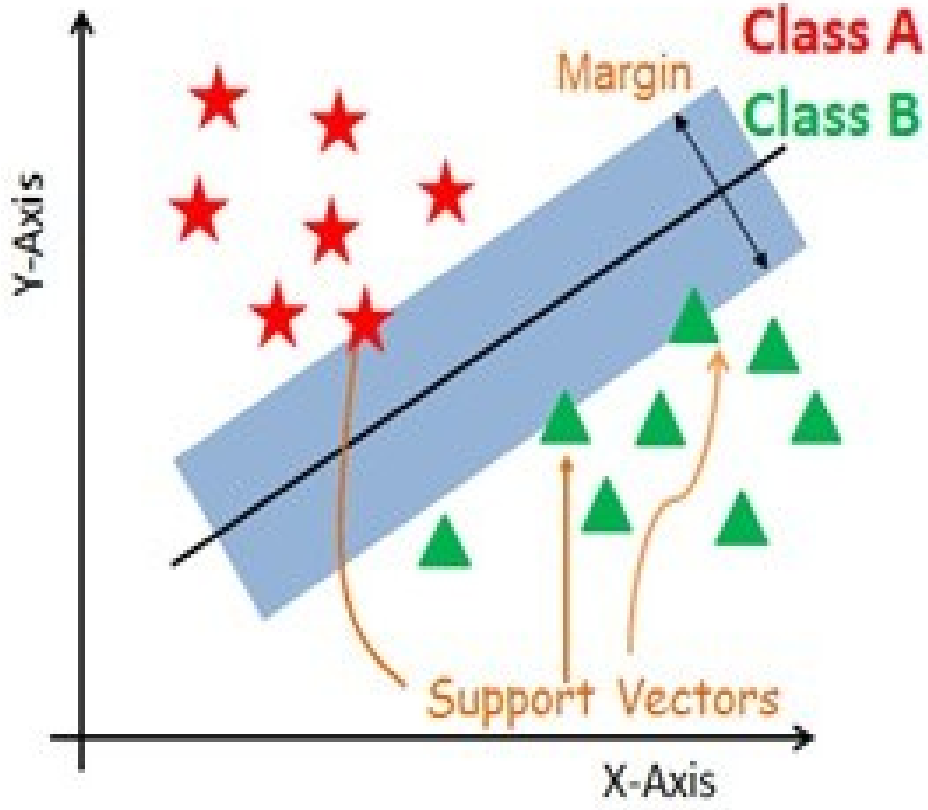


How SVM works ?

- The main objective is to segregate the given dataset in the best possible way.
- The distance between the either nearest points is known as the margin.
- The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset.
- SVM searches for the maximum marginal hyperplane in the following steps:
  - Generate hyperplanes which segregates the classes in the best way.
  - Select the right hyperplane with the maximum segregation from the either nearest data points.

How SVM works ?

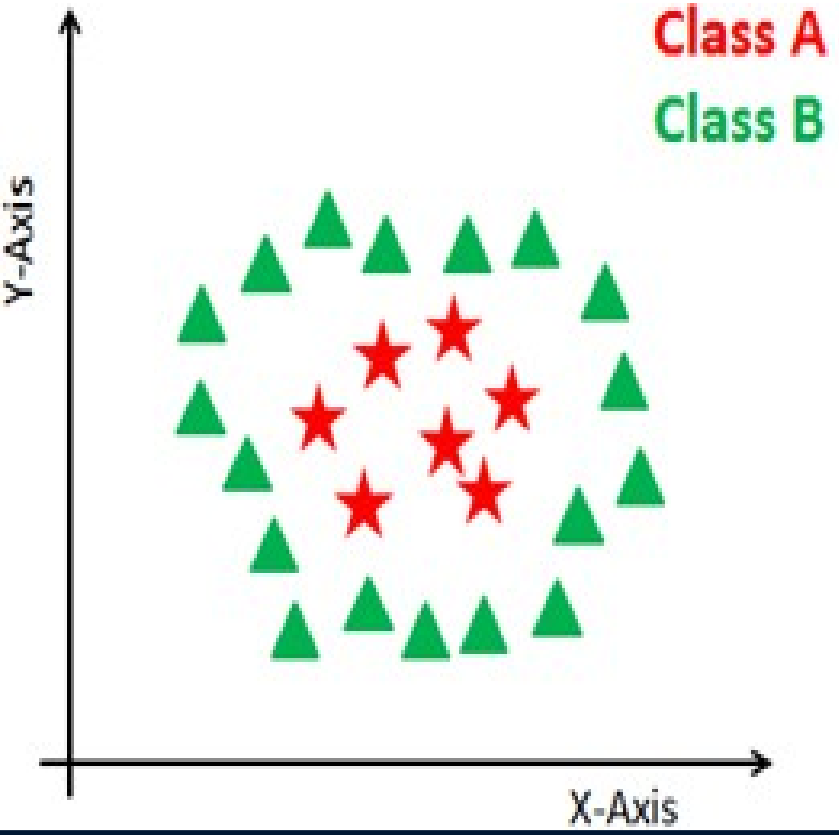


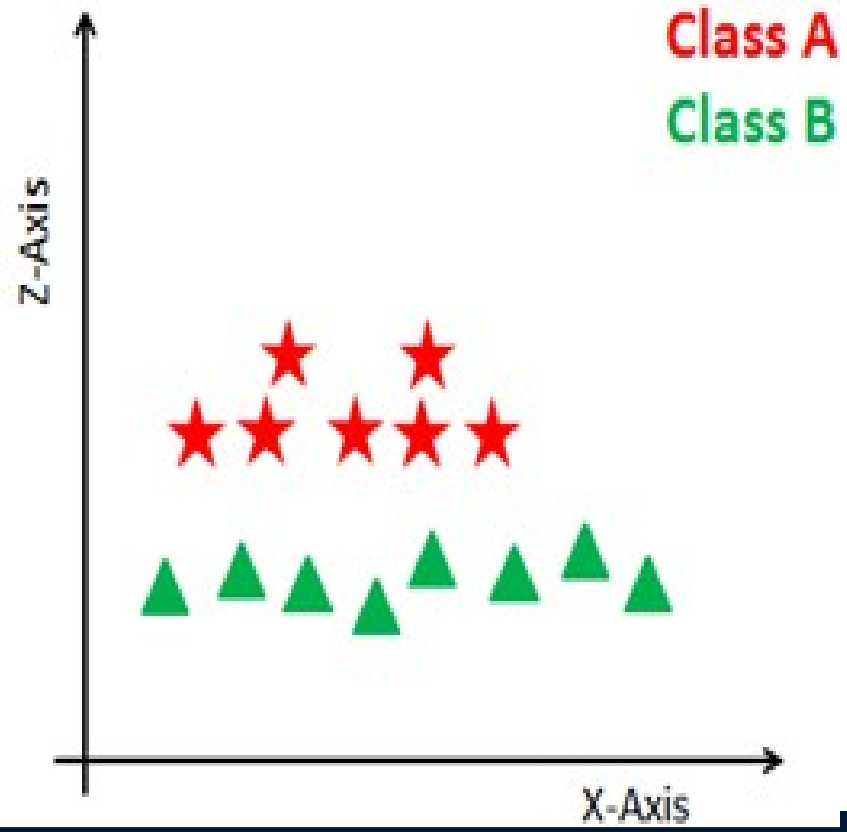


Non-linear and inseparable planes

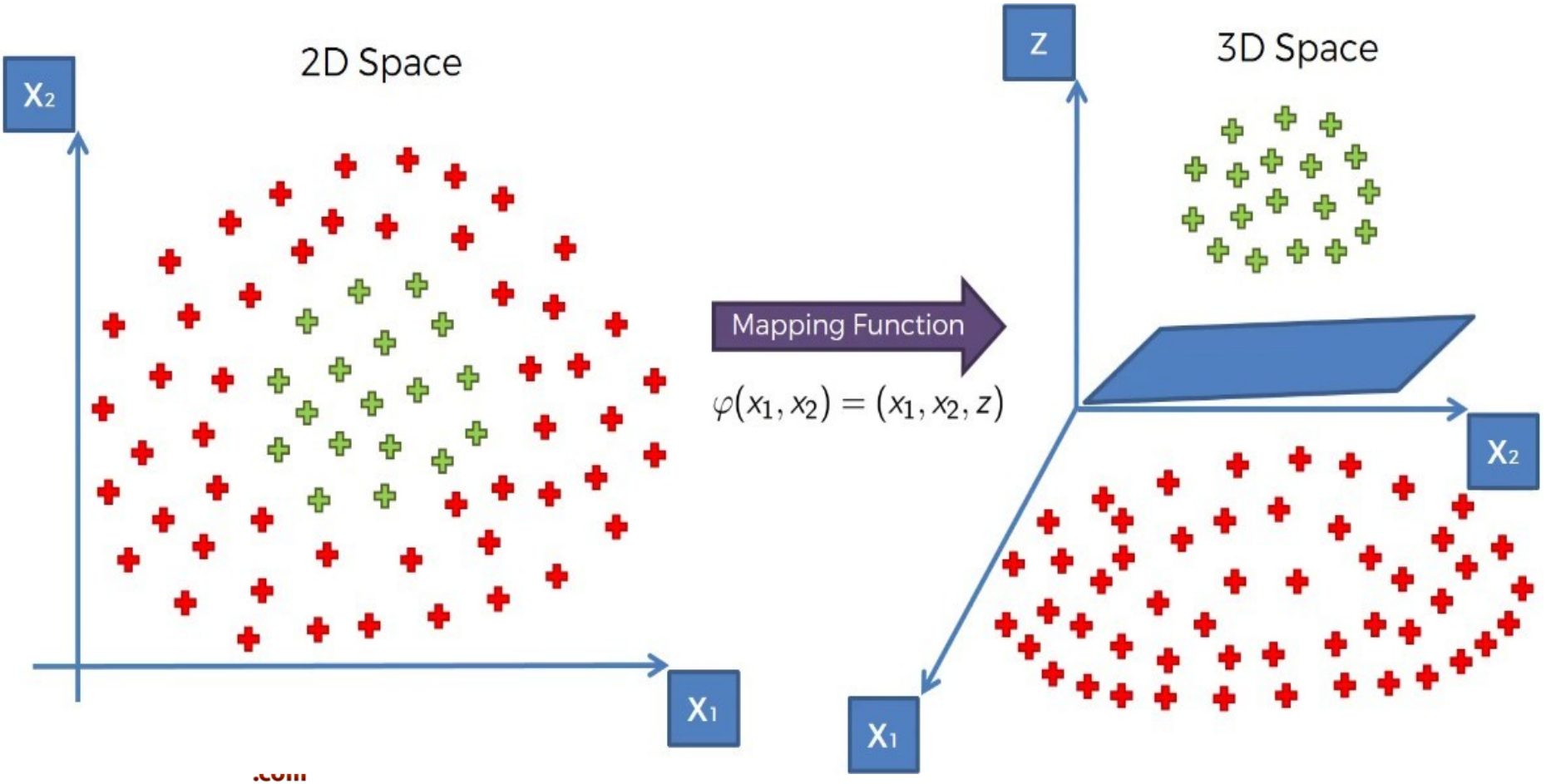
- Some problems can’t be solved using linear hyperplane.
- In such situation, SVM uses a kernel trick to transform the input space to a higher dimensional space as shown on the right.
- The data points are plotted on the x-axis and z-axis (Z is the squared sum of both x and y: z=x^2=y^2).
- Now you can easily segregate these points using linear separation.

Non-linear and inseparable planes



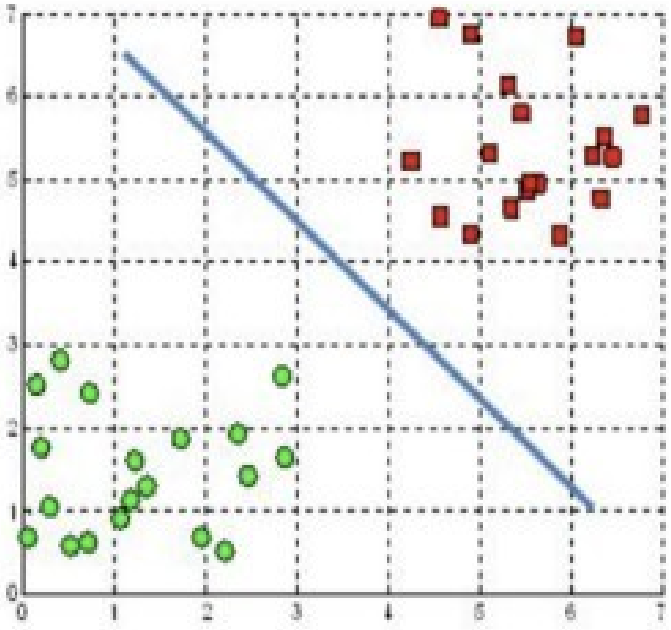


High Dimensional Space Mapping

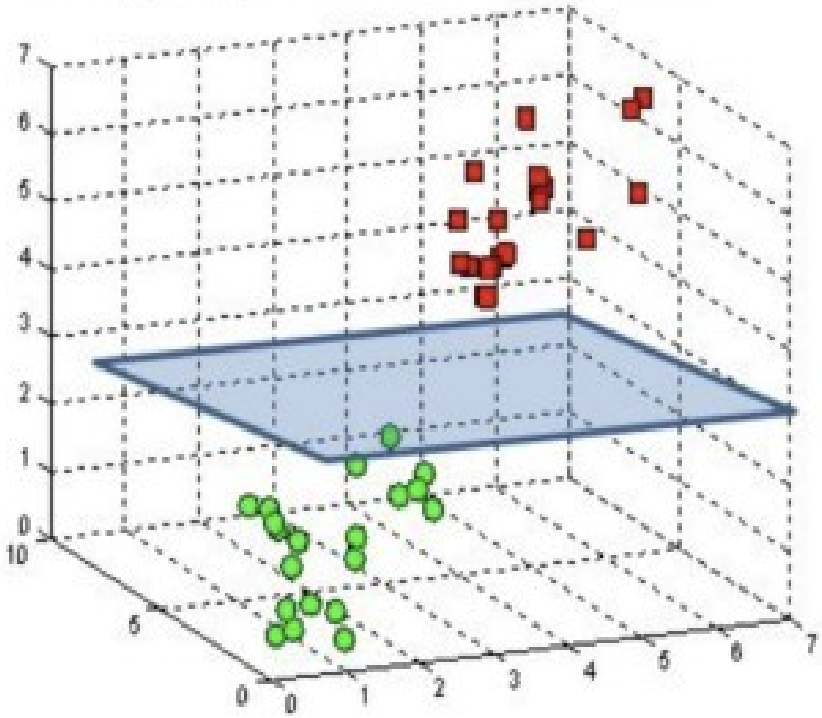


High Dimensional Space Mapping

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



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



Categories of SVM

1. Based on Learning Type

(a) Binary SVM (Hard Margin & Soft Margin)

- Used for **binary classification** (two-class problems).
- Finds the best **hyperplane** that maximizes the margin between two classes.

☑ **Hard Margin SVM**

❖ Assumes **linearly separable** data.

❖ Strictly maximizes margin without allowing misclassification.

Issue: Does not work well with noisy data.

☑ **Soft Margin SVM**

✓ Allows **some misclassification** for better generalization.

✓ Introduces **slack variables** to balance margin size and classification errors.

➢ Works well with **real-world data** that may have noise.

Hard Margin vs. Soft Margin SVM

- SVM is used for **classification** by finding the optimal hyperplane that separates classes.
- The margin refers to the distance between the **support vectors** (closest points) and the decision boundary.

1. Hard Margin SVM

☑ **Assumption:** The data is **linearly separable** (no overlap between classes).
☑ **Goal:** Maximizes the margin while ensuring **no misclassification** (100% accuracy on training data).
🚨 **Limitations:**
Fails when data is **not perfectly separable** (outliers make it infeasible).
Very sensitive to noise, as a single misclassified point can drastically affect the decision boundary.
📌 **Use Case:** Works well for **clean datasets** with **clear class separation** (e.g., basic text classification problems)

Contd.

✅ **Mathematical Formulation:**

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

Subject to constraints:

$$y_i(\mathbf{w}^T x_i + b) \geq 1, \quad \forall i$$

where:

- $\mathbf{w}$ = weight vector (defines the hyperplane),
- b = bias term,
- y_i = class label ($+1$ or -1),
- x_i = input feature vector.

2. Soft Margin SVM

✅ **Assumption:** The data is **not perfectly separable** (some overlap between classes).

✅ **Goal:** Maximizes the margin while allowing **some misclassification** using **slack variables** ξ_i to balance margin width and classification errors.

✅ **Mathematical Formulation:**

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

Subject to constraints:

$$y_i(\mathbf{w}^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

where:

- ξ_i = slack variables (allow misclassification),
- C = **regularization parameter** (controls trade-off between margin size and misclassification).

🔹 **Effect of C:**

- **High C → Strict, fewer misclassifications but smaller margin.**
- **Low C → Larger margin, more misclassifications allowed.**

📌 **Use Case:** Works well for **real-world data** (e.g., spam detection, image recognition), where **some noise and misclassification are acceptable**.

Contd.

Key Differences:

Feature	Hard Margin SVM	Soft Margin SVM
Misclassification	Not allowed	Allowed
Use case	Linearly separable data	Non-linearly separable data
Sensitivity to Outliers	Very high	Handles outliers better
Slack Variables (ξ)	Not used	Used
Regularization Parameter (C)	Not required	Required

Contd.

(b) Multi-Class SVM

Standard SVM is binary, but extensions allow handling multiple classes using techniques like:

- ✓ **One-vs-One (OvO)**: Trains C(C−1)/2 classifiers for C classes.
- ✓ **One-vs-All (OvA)**: Trains CCC classifiers, each distinguishing one class from the rest.

☑ **Use Case:**

✓ **Handwriting recognition (e.g., digits 0-9)**

✓ **Speech classification**

Contd.

2. Based on Kernel Type

☑ Linear SVM

- Uses a **linear kernel** when data is **linearly separable**.
- Fast and interpretable.

❖ Example: **Text classification (spam detection).**

☑ Non-Linear SVM

✓ Uses **non-linear kernels** like **RBF, polynomial, or sigmoid**.

✓ Helps separate complex patterns in data.

❖ Example: **Image recognition, medical diagnosis.**

Contd.

3. Based on Objective

(a) Classification SVM

✓ The standard **SVM for classification** tasks.

✓ Separates data into **discrete classes**.

✓ Uses **support vectors** to define decision boundaries.

☑ Example:

❖ **Email spam detection** (Spam vs. Not Spam).

❖ **Medical diagnosis** (Cancerous vs. Non-cancerous cells).

Contd.

b) Regression SVM (Support Vector Regression - SVR)

✓ Instead of classifying, **SVR predicts continuous values**.

✓ Uses an **epsilon-tube** to fit a function with minimal error.

☑ Example:

❑ **Stock price prediction.**

❑ **Weather forecasting**

c) One-Class SVM (Anomaly Detection)

✓ Used for **outlier detection** or identifying anomalies.

✓ Trains on normal data and detects **deviations** as anomalies.

☑ Example:

❑ **Fraud detection** in banking.

❑ **Intrusion detection** in cybersecurity

Support Vector Machine - Kernel, Nonlinear Classification, and multi-class (Basic concept)

- ✓ Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification tasks, and it can also be applied to regression.
- ✓ It works by finding the hyperplane that best separates the classes of data in a high-dimensional space.
- ✓ Let's break down the key concepts of kernels, nonlinear classification, and multi-class classification in SVM:

1. Kernel in SVM:

- SVM works by finding a linear hyperplane to separate classes of data. However, many real-world problems involve data that is not linearly separable.
- To handle this, SVM uses a technique called the **kernel trick**.
- **Kernel Trick:** This is a mathematical technique that transforms data into a higher-dimensional space where it is easier to find a hyperplane to separate the classes. The kernel function computes the dot product of data points in the higher-dimensional space without explicitly mapping the data to that space, saving computational resources.

Types of Kernels:

Linear Kernel: For linearly separable data.

Polynomial Kernel: For problems where the decision boundary is nonlinear but can be approximated by polynomials.

Radial Basis Function (RBF) Kernel (Gaussian): Effective for high-dimensional and non-linear problems. It maps data into an infinite-dimensional space.

Sigmoid Kernel: Uses a sigmoid function for transforming the data, similar to a neural network activation function.

2. Nonlinear Classification:

- When data is not linearly separable in the original input space, SVM can still perform classification by applying a kernel function to transform the data into a higher-dimensional feature space where a linear separation is possible.

For example:

- ✓ If we have two classes that cannot be separated by a straight line (e.g., a circular pattern), applying a non-linear kernel like the **RBF kernel** maps the data points into a higher-dimensional space where a hyperplane can separate them linearly

Multi-class Classification:

SVM is inherently a binary classifier, meaning it can distinguish between two classes. However, real-world problems often require multi-class classification (more than two classes). There are two common approaches to extend SVM to multi-class problems:

- **One-vs-One (OvO):** For `k` classes, this method constructs a binary SVM for each pair of classes, resulting in $\frac{k(k-1)}{2}$ classifiers. Each classifier is trained to distinguish between a pair of classes.
- **One-vs-All (OvA):** This method creates `k` classifiers, one for each class, where each classifier distinguishes between the target class and all other classes. This leads to a more manageable number of classifiers, i.e., `k` classifiers.

Summary:

- ✓ **Kernel trick** allows SVM to handle non-linearly separable data by mapping it into a higher-dimensional space.
- ✓ **Nonlinear classification** is achieved by applying appropriate kernels like polynomial or RBF.
- ✓ **Multi-class classification** can be handled using strategies like one-vs-one or one-vs-all, as SVM is naturally a binary classifier.

Kernel

- Kernels are a set of functions used to transform data from lower dimension to higher dimension and to manipulate data using dot product at higher dimensions.
- The use of kernel is to apply transformation to data & perform classification at the higher dimension.

SVM Kernels

- The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form.
- SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space.
- In other words, you can say that it converts non-separable problem to separable problems by adding more dimension to it.
- It is most useful in non-linear separation problem.
- Kernel trick helps you to build a more accurate classifier.

Kernel Types

- ✓ In SVM, **kernels** transform input data into higher-dimensional feature spaces, allowing SVM to find optimal decision boundaries even for non-linearly separable data.
- ✓ Below are the four primary types of kernels used in SVM:

- Linear Kernel
- Polynomial Kernel
- Radial Basis Function Kernel
- Sigmoid Kernel

1. **Linear Kernel**

- Formula:

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

- Use Case:
  - Best for linearly separable data.
  - Works well with high-dimensional datasets where data is already well-separated.
- Example:

  If we have two vectors $x_1 = (1, 2)$ and $x_2 = (3, 4)$, their dot product is:

$$K(x_1, x_2) = 1 \times 3 + 2 \times 4 = 3 + 8 = 11$$

2. **Polynomial Kernel**

- **Formula**:

$$K(x_i, x_j) = (x_i^T x_j + c)^d$$

- **Parameters**:
  - d: Degree of the polynomial.
  - c: A constant that controls the influence of higher-degree terms.
- **Use Case**:
  - Useful for non-linear data when relationships are polynomial.
  - Higher-degree polynomials create more complex decision boundaries.

- **Example**:

  If $x_1 = (1, 2)$ and $x_2 = (3, 4)$, with $d = 2$ and $c = 1$:

$$K(x_1, x_2) = (11 + 1)^2 = 144$$

3. Sigmoid Kernel

- **Formula**:

$$K(x_i, x_j) = \tanh(\alpha x_i^T x_j + c)$$

- **Parameters**:
  - α: Scaling factor.
  - c: Shift parameter.
- **Use Case**:
  - Inspired by neural networks (similar to an activation function).
  - Not commonly used due to instability with certain parameters.

- **Example**:

  If $x_1 = (1, 2)$ and $x_2 = (3, 4)$, with $\alpha = 0.01$ and $c = 1$:

$$K(x_1, x_2) = \tanh(0.01 \times 11 + 1) = \tanh(1.11) \approx 0.805$$

4.Radial Basis Function (RBF) Kernel

- **Formula**:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

- **Parameter**:
  - γ (Gamma): Controls the influence of a single training example.
    - **High** γ → Closer decision boundary.
    - **Low** γ → Smoother, generalized boundary.
- **Use Case**:
  - Best for complex, highly non-linear data.
  - Maps data into an infinite-dimensional space.

- **Example**:
  If $x_1 = (1, 2)$ and $x_2 = (3, 4)$, with $\gamma = 0.5$:

$$K(x_1, x_2) = \exp(-0.5 \times ((1-3)^2 + (2-4)^2)) = e^{-2} \approx 0.1353$$

Comparison of Kernels

- ✓ Each kernel has its strengths depending on the dataset.
- ✓ **Linear kernels** are simple and interpretable,
- ✓ while **RBF kernels** are powerful for non-linear data.

Kernel Type	Best Used For	Complexity	Hyperparameters
Linear	Linearly separable data	Low	None
Polynomial	Data with polynomial relationships	Medium-High	d, c
RBF	Highly non-linear data	High	γ
Sigmoid	Neural network-like behavior	Medium	α, c

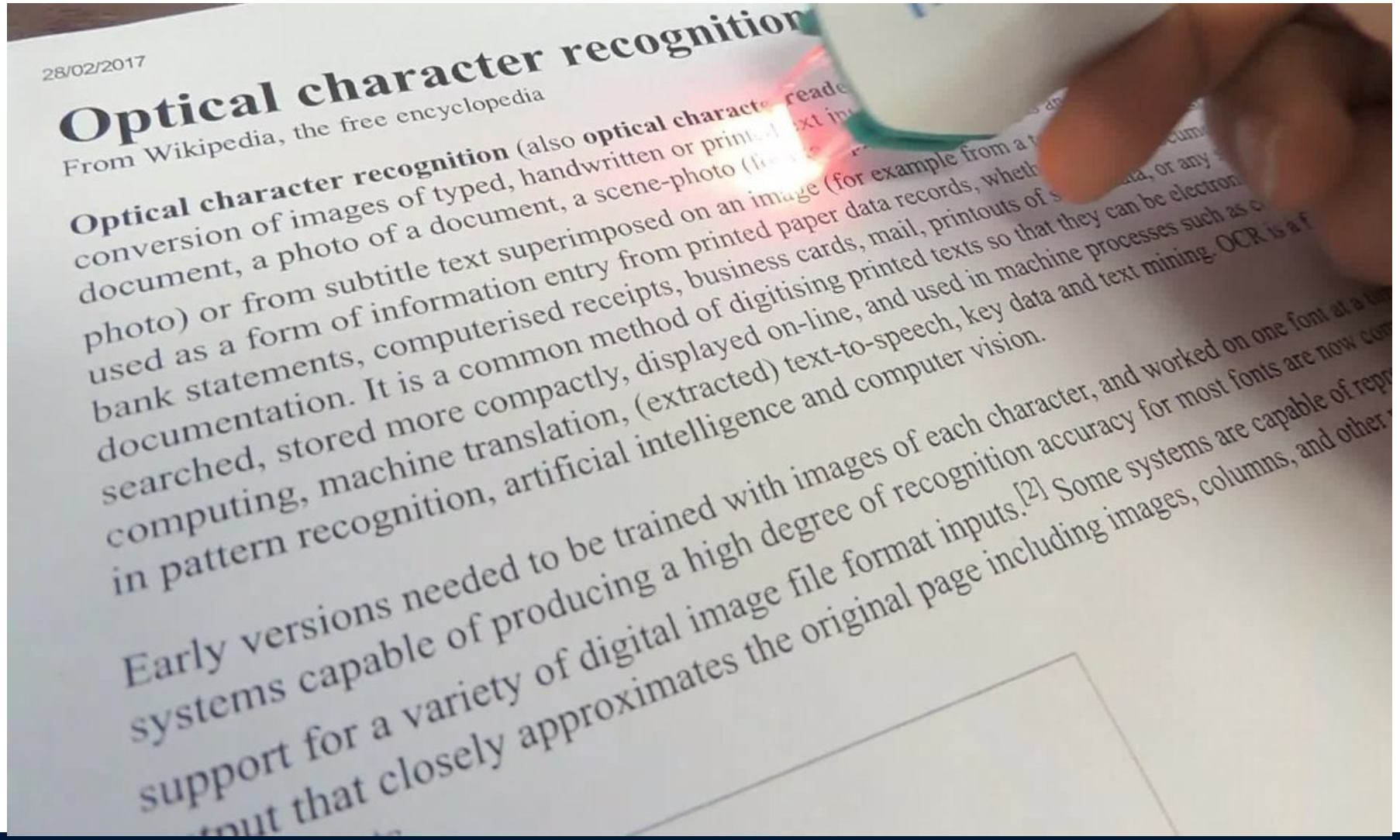
Radial Basis Function Kernel

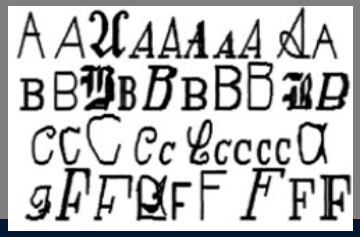
- The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

$$K(x,xi) = \exp(-\text{gamma} * \text{sum}((x - xi^2))$$

- 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.

Example: OCR


28/02/2017
Optical character recognition
From Wikipedia, the free encyclopedia
Optical character recognition (also optical
conversion of images of typed, handwritten
document, a photo of a document, a
photo) or from subtitle text superimposed
used as a form of information entry from
bank statements, computerised receipts,
documentation. It is a common method
searched, stored more compactly, displayed
computing, machine translation, (extracted)
in pattern recognition, artificial intelligence
Early versions needed to be trained with images of each character, and worked on one font at a
systems capable of producing a high degree of recognition accuracy for most fonts are now
support for a variety of digital image file format inputs.[2] Some systems are capable of
that closely approximates the original page including images, columns, and other



Example: Image Processing

- Image processing is a difficult task for many types of machine learning algorithms.
- The relationships linking patterns of pixels to higher concepts are extremely complex and hard to define.
- For instance, it's easy for a human being to recognize a face, a cat, or the letter "A", but defining these patterns in strict rules is difficult.
- Furthermore, image data is often noisy. There can be many slight variations in how the image was captured, depending on the lighting, orientation, and positioning of the subject.

Example: Data Collection

- When OCR software first processes a document, it divides the paper into a matrix such that each cell in the grid contains a single glyph, which is just a term referring to a letter, symbol, or number.
- Next, for each cell, the software will attempt to match the glyph to a set of all characters it recognizes.
- Finally, the individual characters would be combined back together into words, which optionally could be spell-checked against a dictionary in the document's language.

The Dataset

- We'll use a dataset donated to the UCI Machine Learning Data Repository (http://archive.ics.uci.edu/ml) by W. Frey and D. J. Slate.
- The dataset contains 20,000 examples of 26 English alphabet capital letters as printed using 20 different randomly reshaped and distorted black and white fonts.
- The following figure, published by Frey and Slate, provides an example of some of the printed glyphs.
- Distorted in this way, the letters are challenging for a computer to identify, yet are easily recognized by a human being:

Why KKT Conditions in SVM?

- ✓ In **SVM**, we aim to find a **maximum margin hyperplane** while ensuring correct classification.
- ➢ This leads to a **constrained quadratic optimization problem**.
- ❖ **KKT conditions** help solve this problem efficiently.

Support Vector Machine - KKT Condition

- ✓ The **Karush-Kuhn-Tucker (KKT) conditions** are a set of mathematical conditions used to solve optimization problems that are subject to inequality constraints, and they are fundamental to the optimization process in Support Vector Machines (SVM).
- ✓ **Context of KKT in SVM:**
- In the case of SVM, we are trying to find a hyperplane that maximizes the margin (distance between the closest points of the two classes).
- This is done through an optimization problem with constraints, and the KKT conditions help in solving this problem.
- ➢ In SVM, the objective is to **maximize the margin** while ensuring that the data points are correctly classified.
- ➢ This can be formulated as a **constrained optimization problem**. The Lagrangian formulation of the problem leads to the need to satisfy the KKT conditions.

The Optimization Problem in SVM:

Given Given a set of training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ (feature vectors) and y and $y_i \in \{-1, 1\}$ (class labels), the SVM optimization problem is:

Maxin Maximize:

$$\frac{1}{2}\|\mathbf{w}\|^2$$

Subje Subject to:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i$$

Where:

- $\mathbf{w}$ is the weight vector (normal vector to the hyperplane).
- b is the bias term (offset from the origin).
- y_i is the class label of the i-th sample.
- $\mathbf{x}_i$ is the feature vector of the i-th sample.

This optimization problem is subject to **inequality constraints**, which makes it a **constrained optimization problem**.

The KKT Conditions:

The **KKT conditions** provide a set of necessary conditions for a solution to be optimal in this constrained optimization problem. These conditions are:

1. **Primal feasibility:** The constraints must be satisfied.

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \quad \forall i$$

2. **Dual feasibility:** The Lagrange multipliers associated with the constraints must be non-negative.

$$\alpha_i \geq 0 \quad \forall i$$

   where α_i are the Lagrange multipliers for the constraints.

3. **Complementary slackness:** For each constraint, either the Lagrange multiplier is zero, or the constraint is active (i.e., the sample lies on the margin).

$$\alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right) = 0 \quad \forall i$$

   This condition means that if a sample is not on the margin, the Lagrange multiplier $\alpha_i = 0$, implying that the constraint is not active for that sample. If a sample lies on the margin, $\alpha_i > 0$.

4. **Stationarity:** The gradient of the Lagrangian with respect to the primal variables (i.e., $\mathbf{w}$ and b) must be zero. The Lagrangian for the SVM is given by:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right)$$

   The stationarity condition involves taking partial derivatives of the Lagrangian with respect to $\mathbf{w}$ and b and setting them to zero:

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \text{and} \quad \frac{\partial L}{\partial b} = 0$$

Understanding the Conditions:

- **Primal feasibility** ensures that the data points are correctly classified with a margin of at least 1.
- **Dual feasibility** ensures that the Lagrange multipliers are non-negative, indicating that each constraint is either satisfied or the margin is maximized.
- **Complementary slackness** ensures that the Lagrange multiplier for any sample that is not on the margin is zero, and for samples on the margin, the multiplier is positive.
- **Stationarity** ensures that we are at an optimal point where the gradient of the Lagrangian is zero.

Expresses w as a **linear combination** of **support vectors**.

Contd.

Summary of KKT Conditions in SVM:

In SVM optimization, the KKT conditions are critical for finding the optimal separating hyperplane. The conditions ensure that:

- The hyperplane separates the classes with the maximum margin.
- The constraints are satisfied or active only for those points that are critical (support vectors).

By solving these conditions, we can derive the optimal values for the weight vector $\mathbf{w}$ and the bias term b, as well as determine the support vectors and the associated Lagrange multipliers α_i.

Numerical Example: Finding Optimal Margin in SVM

Given Data Points

We have two classes (**+1 and -1**) in a **2D space**:

Point (x_1, x_2)	Class (y)
(2, 3)	+1
(3, 3)	+1
(2, 1)	-1
(3, 1)	-1

CONTD.

Step 1: Define the Optimal Hyperplane

The equation of the decision boundary (hyperplane) in **linear SVM** is:

$$w_1 x_1 + w_2 x_2 + b = 0$$

Let's assume **optimal weights** after training are:

$$w = [1, 1], \quad b = -4$$

So, the decision boundary equation is:

$$x_1 + x_2 - 4 = 0$$

CONTD.

Step 2: Find the Margin

The margin is defined as:

$$\text{Margin} = \frac{2}{||w||}$$

where $||w||$ is the **L2 norm** (magnitude) of the weight vector:

$$||w|| = \sqrt{w_1^2 + w_2^2} = \sqrt{1^2 + 1^2} = \sqrt{2}$$

$$\text{Margin} = \frac{2}{\sqrt{2}} = \sqrt{2} \approx 1.414$$

Contd.

Step 3: Check Margin Trade-off

- If we **increase margin**, it may **include misclassified points**.
- If we **reduce margin**, we may **overfit** to training data.

To adjust margin, **tune the** `C` **parameter** in soft-margin SVM:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum \xi_i$$

- **Higher C** → Smaller margin, fewer misclassifications.
- **Lower C** → Larger margin, allows some misclassification.

References:

- https://mitu.co.in
- https://superdatascience.com
- https://stackabuse.com/
- https://jakevdp.github.io
- https://towardsdatascience.com
- https://www.datacamp.com
- http://scikit-learn.org/