电子科技大学
格拉斯哥学院
Glasgow College, UESTC

KNN, SVM, and Decision Tree

Supervised Learning

- Computation methods used in the supervised learning process
  - Regression
  - Naïve Bayes
  - K-Nearest Neighbors
  - Support Vector Machines
  - Decision Tree
  - Neural Networks

K-Nearest Neighbors (KNN)

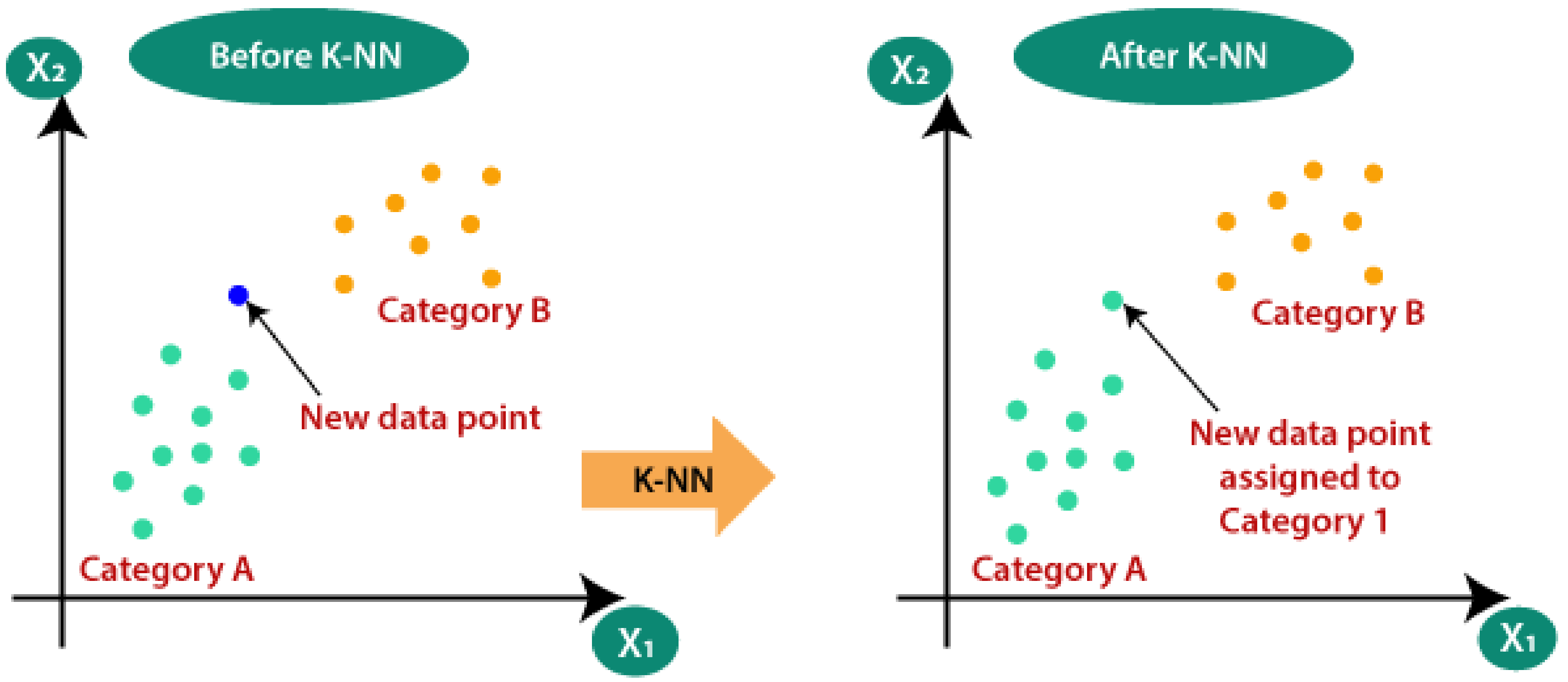
K-Nearest Neighbors

- The K-nearest neighbors (KNN) algorithm is easy-to-implement supervised machine learning algorithm, and has been used for both regression and classification.

- The KNN algorithm relies on labelled input data to learn a function that produces an appropriate output when given new unlabelled data.

- KNN is a non-parametric classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

KNN - Intuition

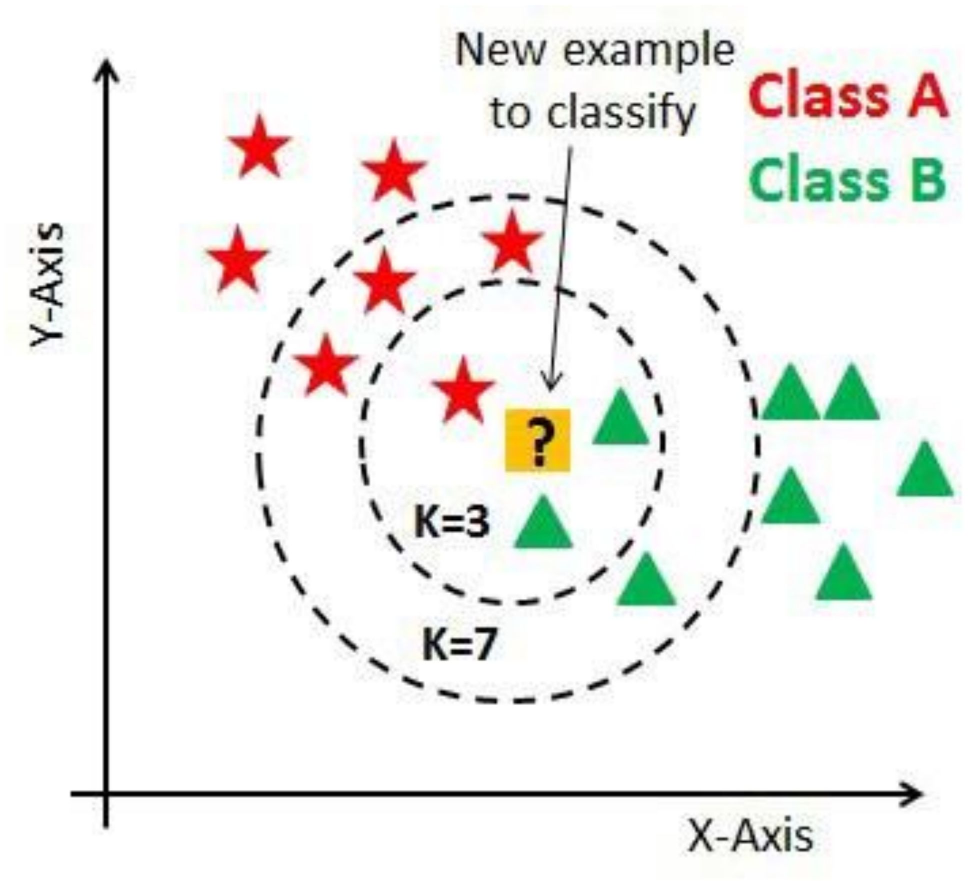


- Suppose there are two categories, i.e., Category *A* and Category *B*, and we have a new data point x_1, so this data point will lie in which of these categories?

Source: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

KNN - Intuition

- KNN tries to predict the correct class by calculating the distance between the test data and all the training points (from all classes).
- Select the *K*, number of points, closest to the test data.
- Assign the new data points to class *A* or *B* according to the majority point.

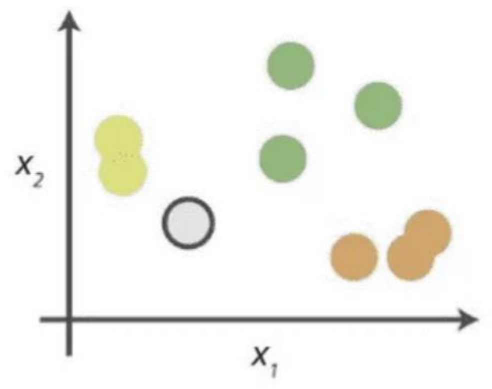


KNN Algorithm

- The K-NN working can be explained on the basis of the below algorithm:
  - **Step 1:** Select the number *K* of the neighbors.
  - **Step 2:** Calculate the Euclidean distance of *K* number of neighbors.
  - **Step 3:** Take the *K* nearest neighbors as per the calculated Euclidean distance.
  - **Step 4:** Among these *K* neighbors, count the number of the data points in each category.
  - **Step 5:** Assign the new data points to that category for which the number of the neighbor is maximum.
  - **Step 6:** Our model is ready.

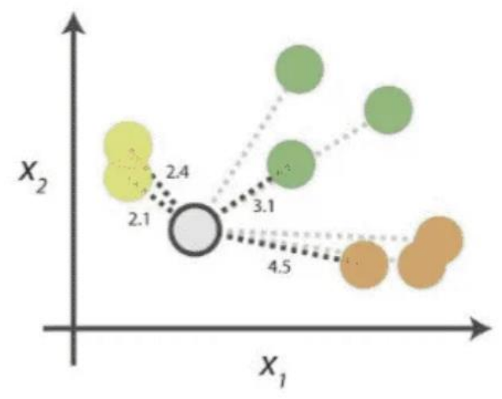
How does the KNN Algorithm Work?

0. Look at the data



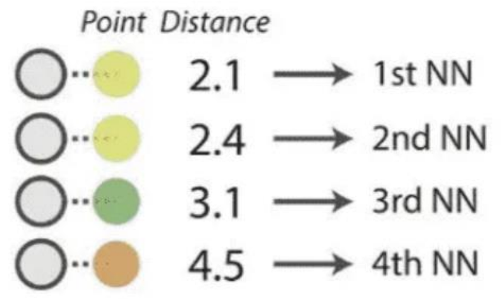
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances



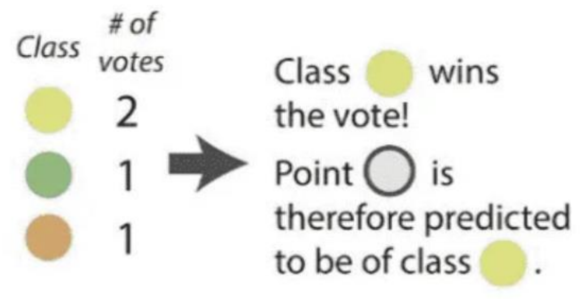
Start by calculating the distances between the grey point and all other points.

2. Find neighbours



Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

3. Vote on labels

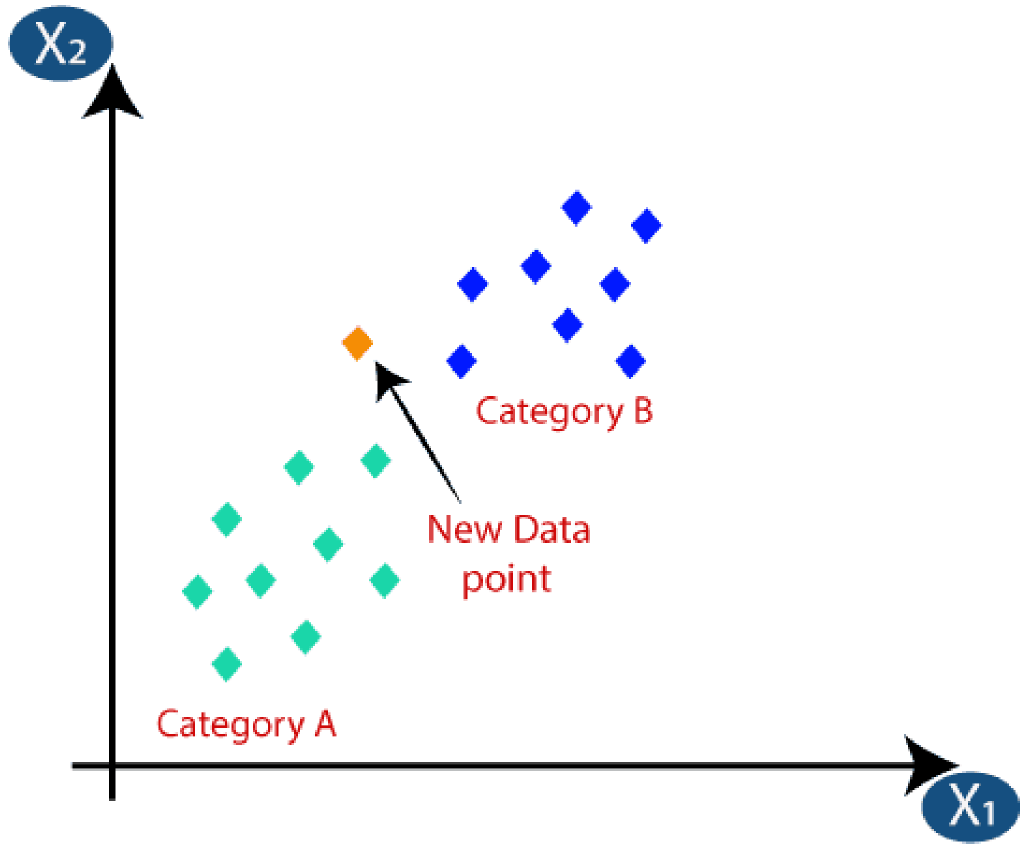


Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4#:~:text=The%20KNN%20algorithm%20calculates%20the,'K'%20selected%20training%20points.

KNN Procedure: an Example

- Step 1: Suppose we have a new data point, and we need to put it in the required category.
- Firstly, we need to choose the number of neighbors i.e., K =?
- Let’s choose K=5 fo.r this example



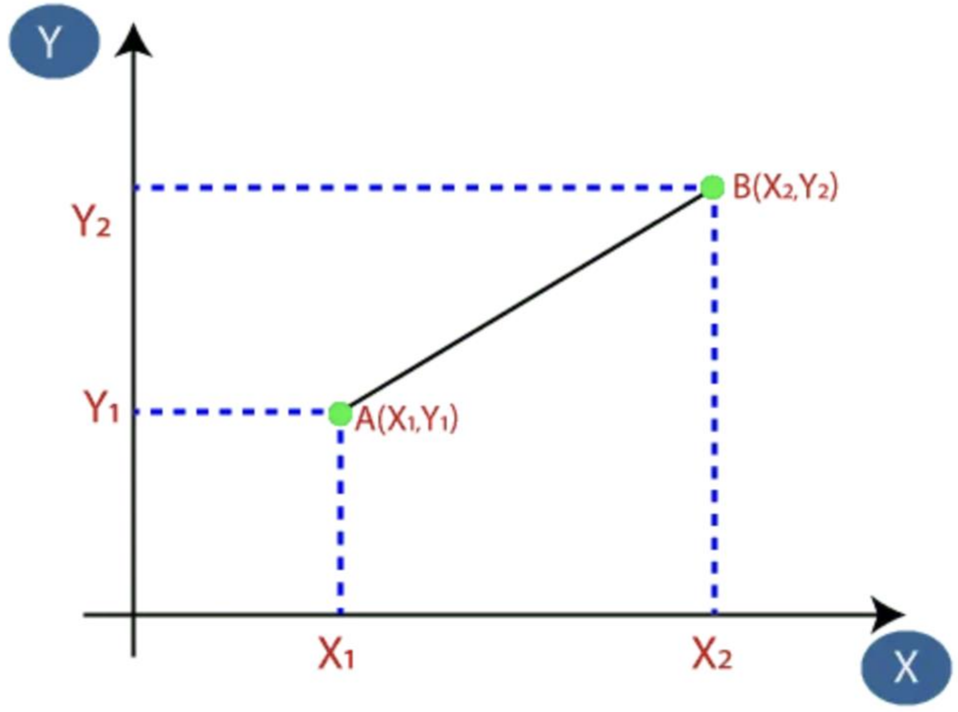
https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4#:~:text=The%20KNN%20algorithm%20calculates%20the,'K'%20selected%20training%20points.

KNN Procedure: an Example

- Step 2: we will calculate the Euclidean distance between the data points

- The Euclidean distance is the shortest/direct distance between points A and B

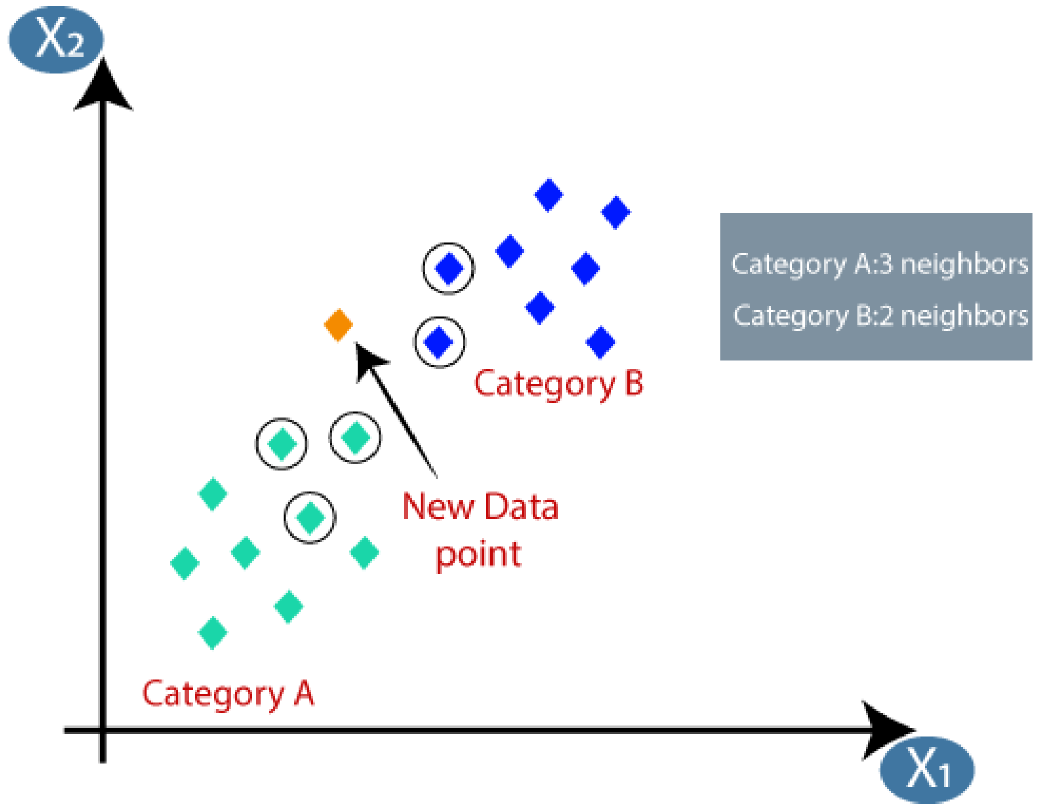
$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4#:~:text=The%20KNN%20algorithm%20calculates%20the,'K'%20selected%20training%20points.

KNN Procedure: an Example

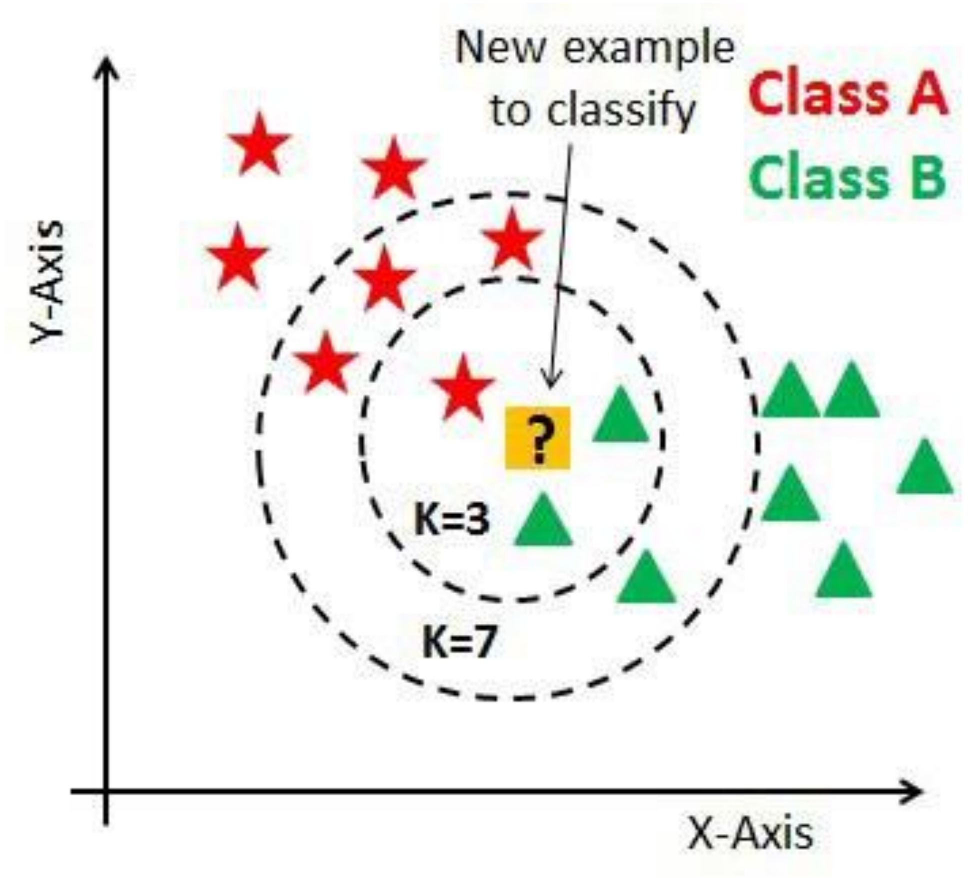
- Step 3: By calculating the Euclidean distance, we got the 5 nearest neighbors.
- Step 4: Three nearest neighbors in category *A* and two nearest neighbors in category *B*.
- Step 5: As majority of the points belong to category *A*, the new data point will be included / affiliated to category *A*.



https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4#:~:text=The%20KNN%20algorithm%20calculates%20the,'K'%20selected%20training%20points.

How to choose the value of K?

- If we proceed with *K*=3, then we predict that test input belongs to class *B*.
- But, if we continue with *K*=7, then we predict that test input belongs to class *A*.
- Which one is correct?
- How to find the optimal value of *K*?

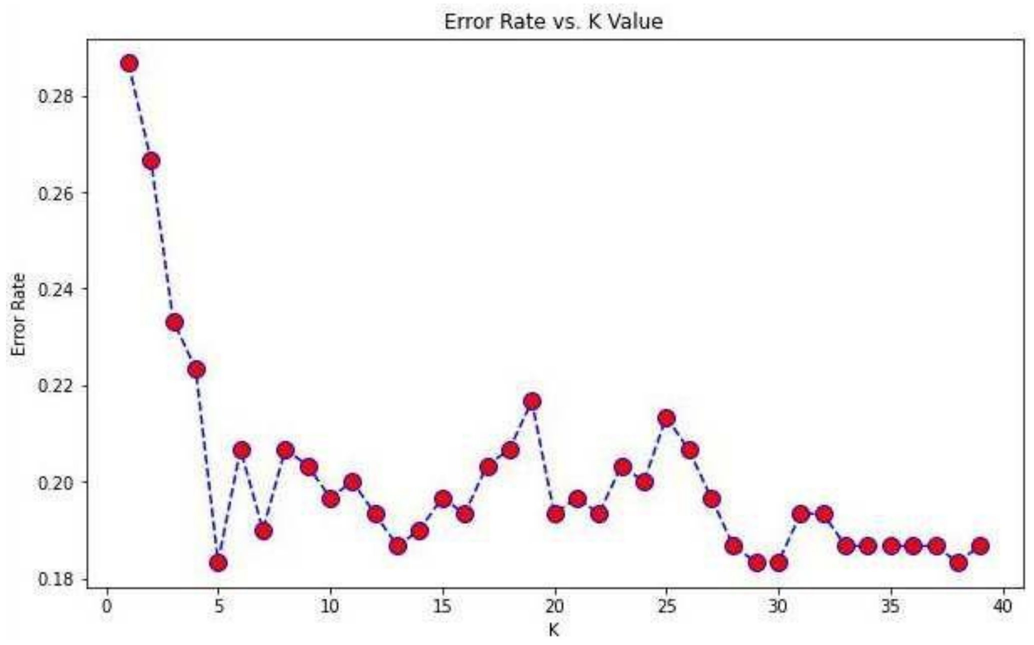


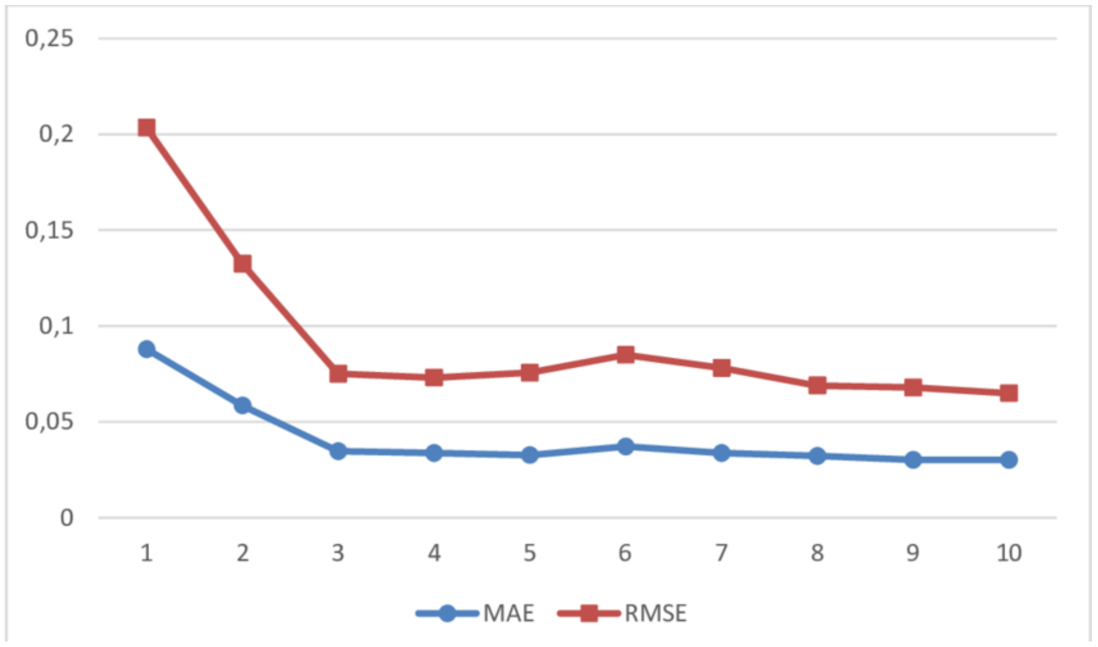
https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4#:~:text=The%20KNN%20algorithm%20calculates%20the,'K'%20selected%20training%20points.

Optimal *K* Value

- There are no pre-defined statistical methods to find the most favourable value of *K*.
- Initialize a random *K* value and start computing.
- Choosing a small value of *K* leads to unstable decision boundaries.
- The substantial *K* value is better for classification as it leads to smoothening the decision boundaries.
- Derive a plot between error rate and *K* denoting values in a defined range. Then, choose the *K* value as having a minimum error rate.

Optimal K Value – Sample Results





Source: https://www.researchgate.net/figure/Error-rate-vs-K-value-graph-for-KNN_fig3_357230455

Source: https://www.researchgate.net/figure/Error-rates-versus-neighbourliness-number-k-in-kNN-algorithm_fig1_343658672

Calculating Distance

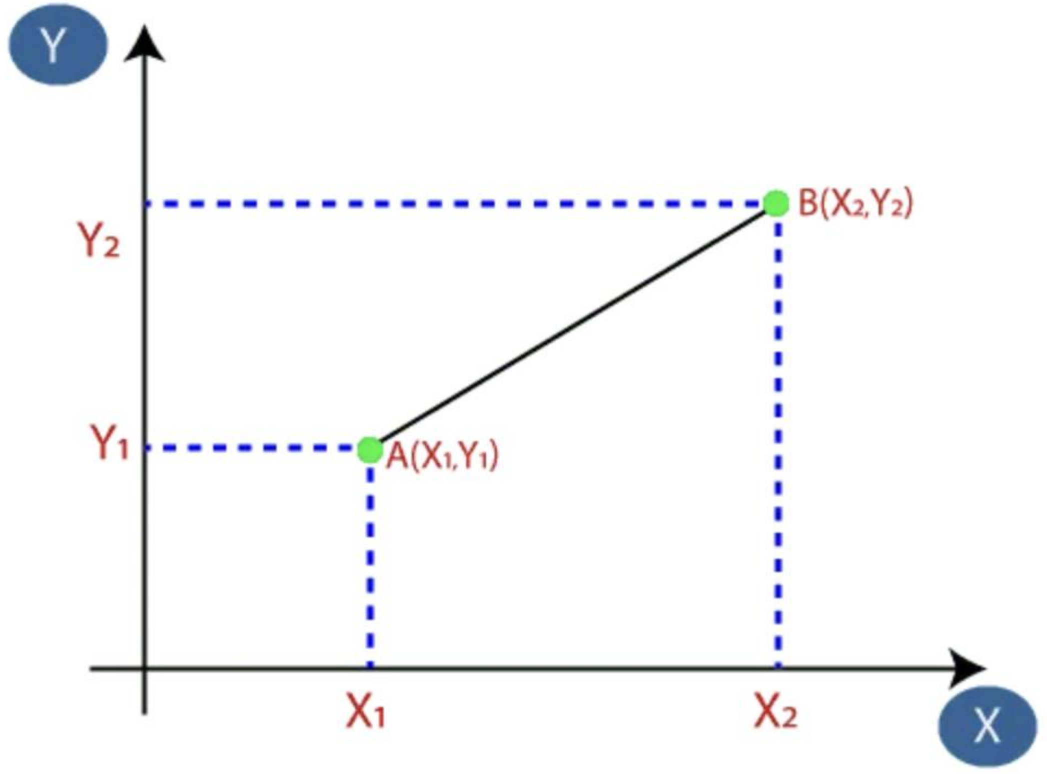
- There are various methods for calculating the distance, of which the most commonly known methods are:
  - Euclidian distance
  - Manhattan distance
  - Hamming distance
  - Minkowski distance

Euclidean Distance

- The Euclidean distance is defined as the distance between two points:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

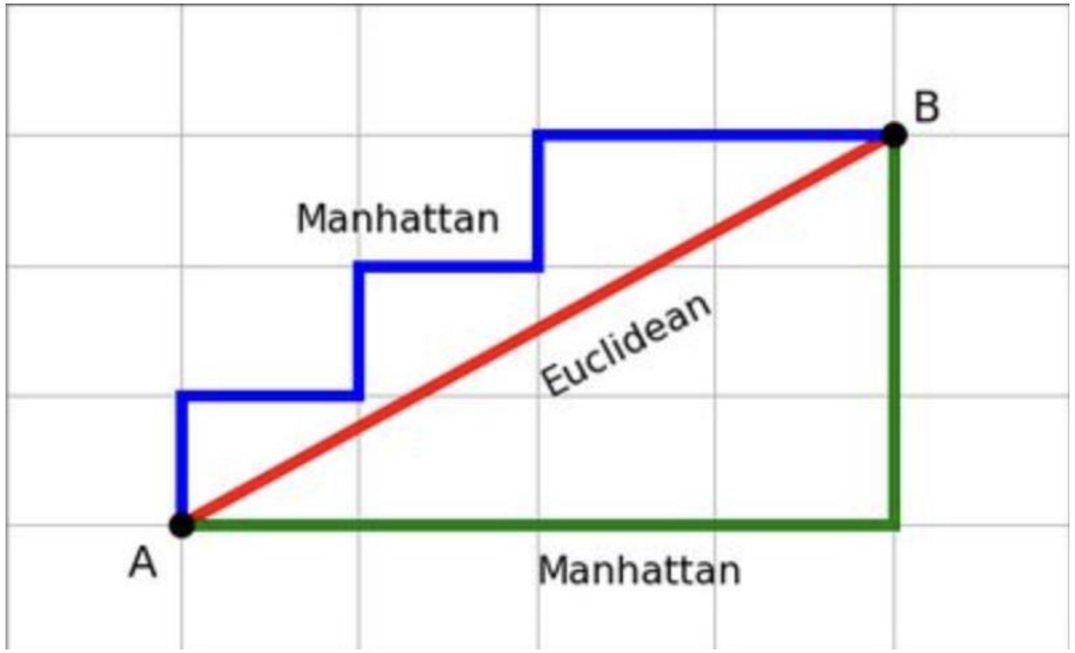
$$d = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$



Manhattan Distance

- This is the distance between real vectors using the sum of their absolute difference.
- If there two vectors A and B in an n-dimensional space, the Manhattan Distance can be calculated as:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$



Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

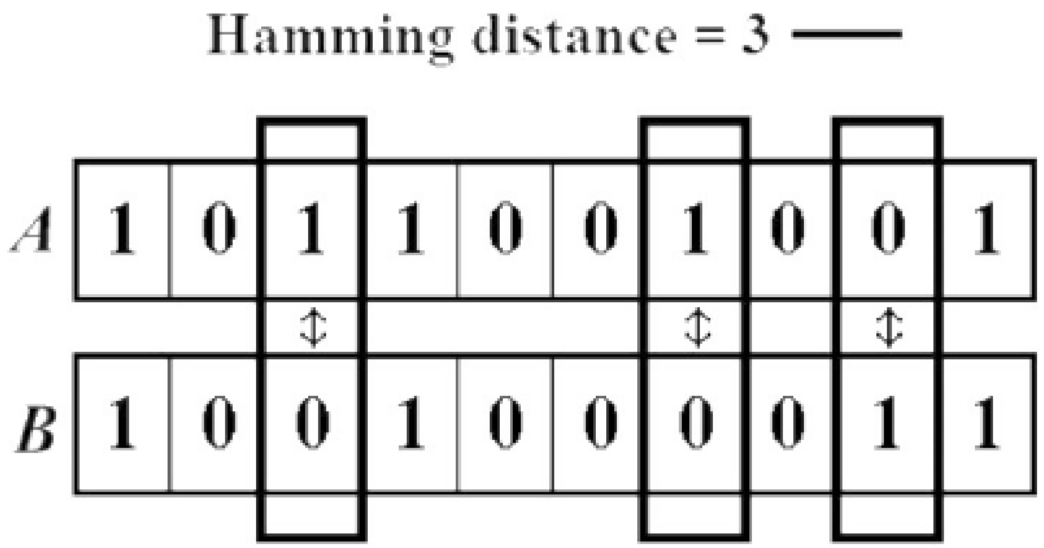
Hamming Distance

- Hamming Distance is used for categorical variables.
- If the value (x) and the value (y) are the same, the distance D will be equal to 0.
- Otherwise, D = 1

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$



Minkowski Distance

- Minkowski Distance is a metric used to measure the similarity or dissimilarity between two time series by computing the sum of the absolute differences raised to a certain power.

The Minkowski distance of order p (where p is an integer) between two points

$$X = (x_1, x_2, \ldots, x_n) \text{ and } Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$$

is defined as:

$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

Applications of KNN

- The k-NN algorithm has been utilized within a variety of applications, mainly within classification. Some of these use cases include:
- Data pre-processing: Datasets frequently have missing values, but the KNN algorithm can estimate those values in a process known as missing data imputation.
- Recommendation Engines: Using clickstream data from websites, the KNN algorithm has been used to recommend additional content to users automatically. This kind of research shows that a user is assigned to a particular group, and based on that group's user behaviour, they are given a recommendation. However, given the scaling issues with KNN, this approach may not be optimal for larger datasets.

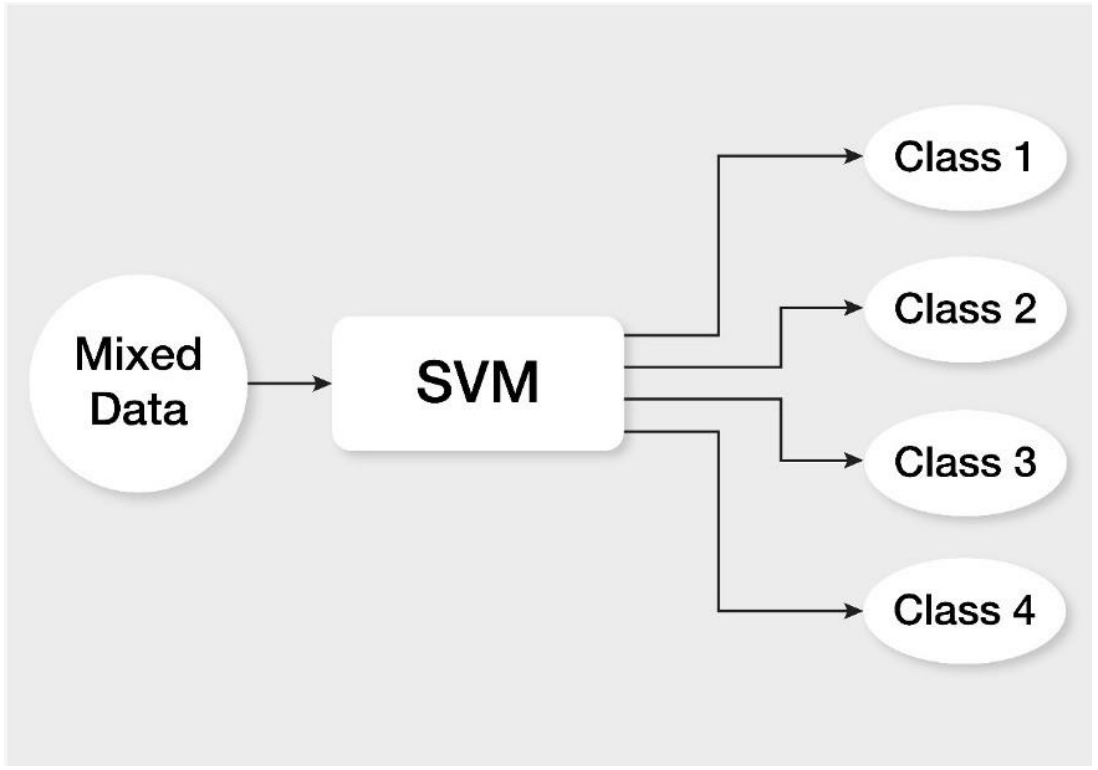
Applications of KNN

- Finance: KNN has also been used in a variety of finance and economic use cases. For example, one research shows how using KNN on credit data can help banks assess risk of a loan to an organization or individual. It is used to determine the credit-worthiness of a loan applicant.
- Healthcare: KNN has also had application within the healthcare industry, making predictions on the risk of heart attacks and prostate cancer. The algorithm works by calculating the most likely gene expressions.
- Pattern Recognition: KNN has also assisted in identifying patterns, such as in text and digital classification. This has been particularly helpful in identifying handwritten numbers that you might find on forms or mailing envelopes.

Introduction of SVM

Support Vector Machine

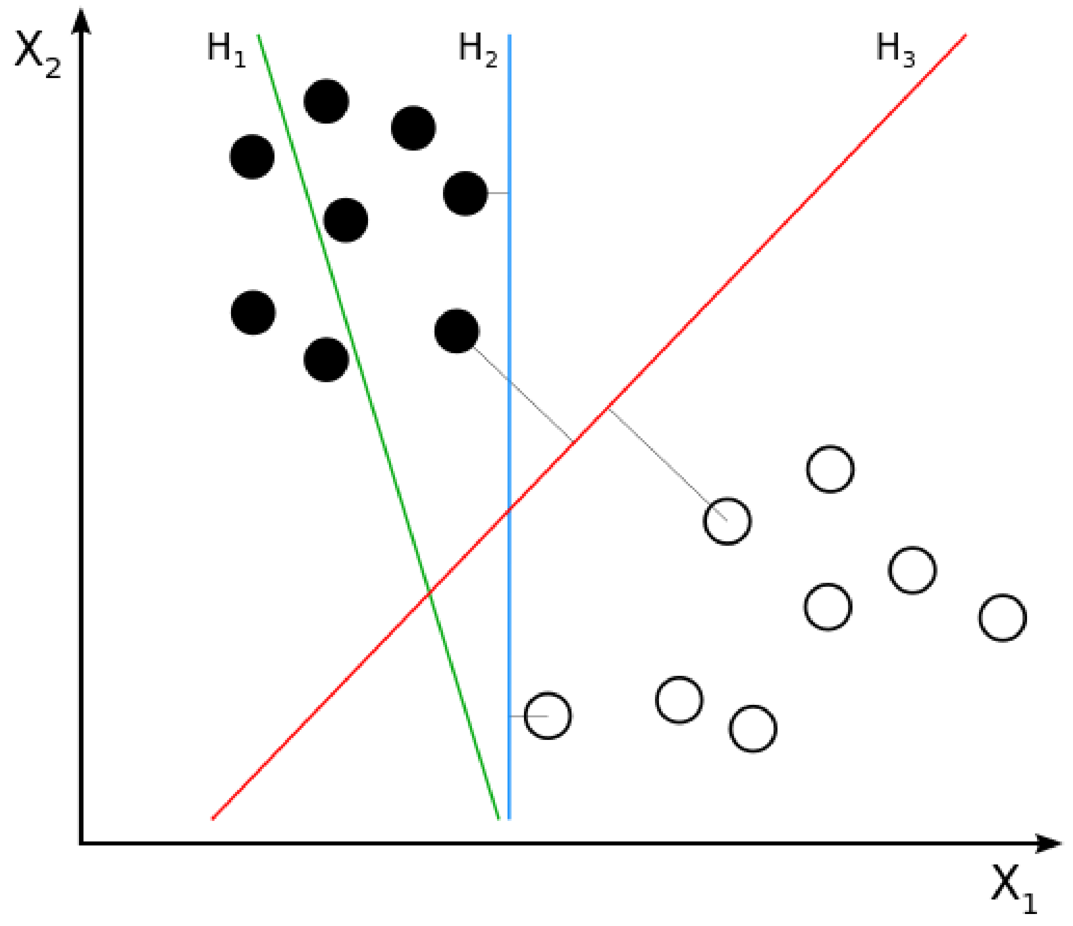
- Support Vector Machine (SVM) is a linear model for classification and regression problems.
- The SVM algorithm creates a line or a hyperplane which separates the data into classes.



Intuition

- At first approximation what SVMs do is to find a separating line (or hyperplane) between data of two classes.

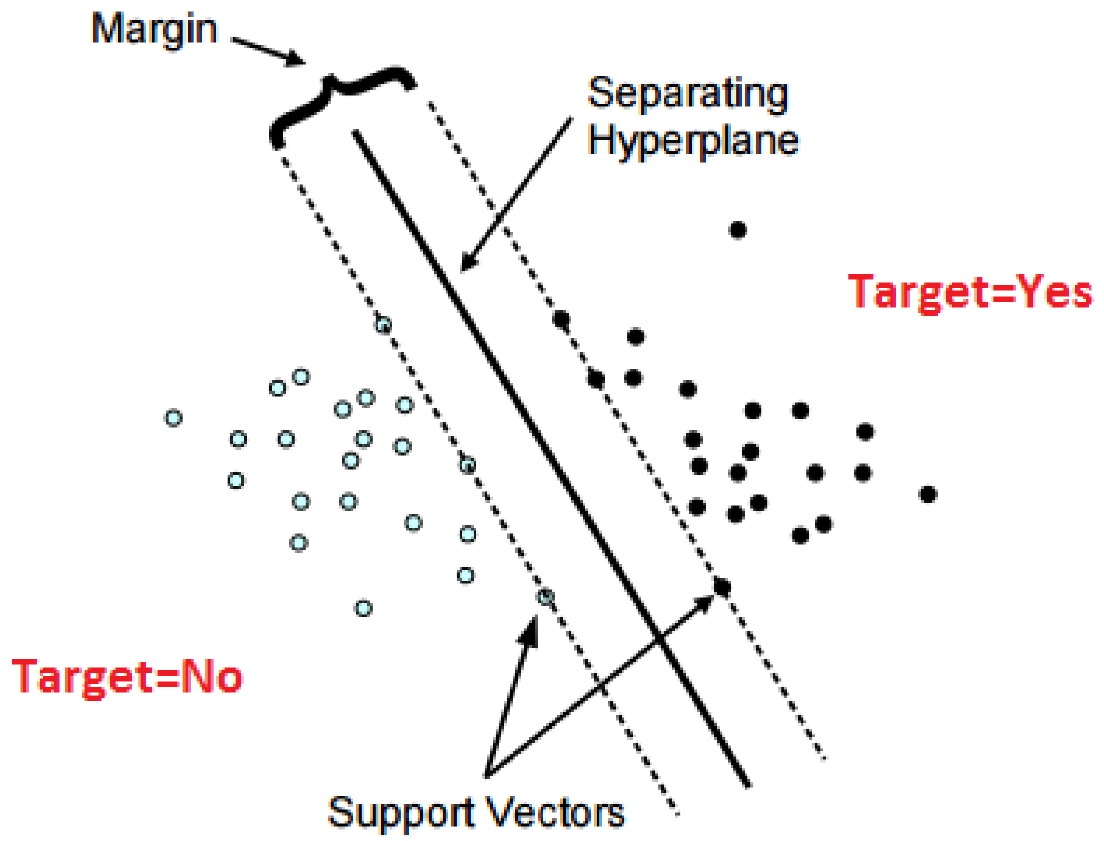
- SVM is an algorithm that takes the data as an input and outputs a line that separates those classes if possible.



Hyperplane

- Hyperplane is a decision boundary which separates between given set of data points having different class labels.

- The SVM classifier separates data points using a hyperplane with the maximum amount of margin.

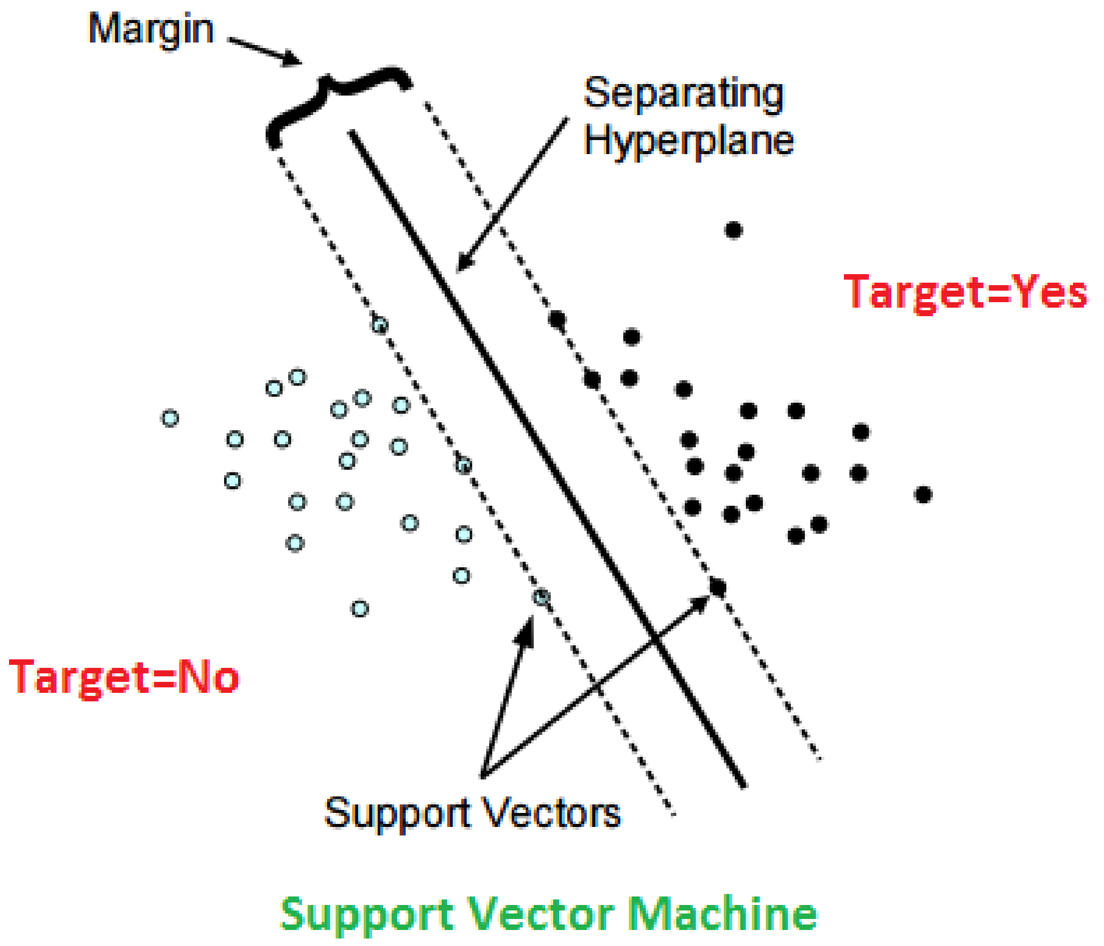


Support Vector Machine

Support Vector

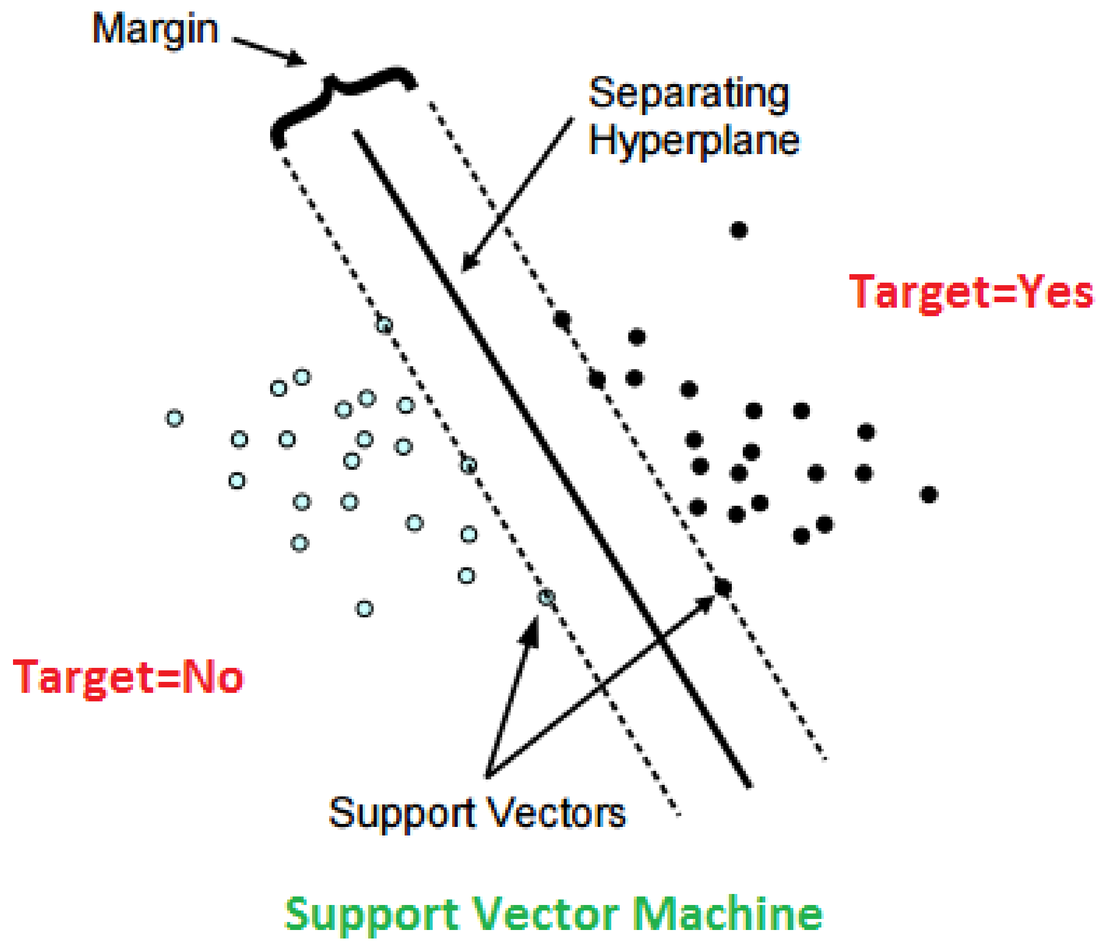
- Support vectors are the sample data points, which are closest to the hyperplane.

- These data points will define the separating line or hyperplane better by calculating margins.

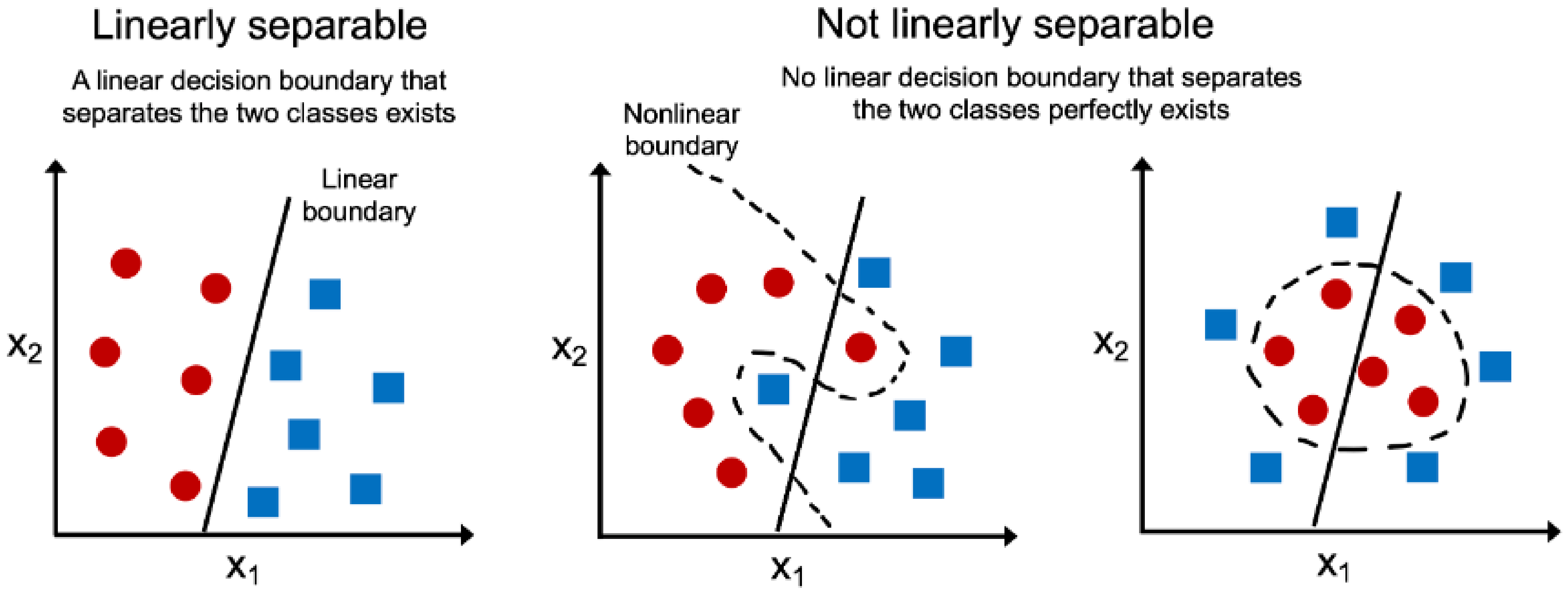


Margin

- A margin is a separation gap between the two lines on the closest data points.
- In SVMs, we try to maximize this separation gap so that we get maximum margin.



Types of SVM


Linearly separable
A linear decision boundary that separates the two classes exists
Linear boundary
x_2
x_1
Not linearly separable
No linear decision boundary that separates the two classes perfectly exists
Nonlinear boundary
x_2
x_1
x_2
x_1

Types of SVM

- **Linear Separable SVM**

  When the data is perfectly linearly separable only then we can use Linear SVM.

- **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) .

Linear SVM

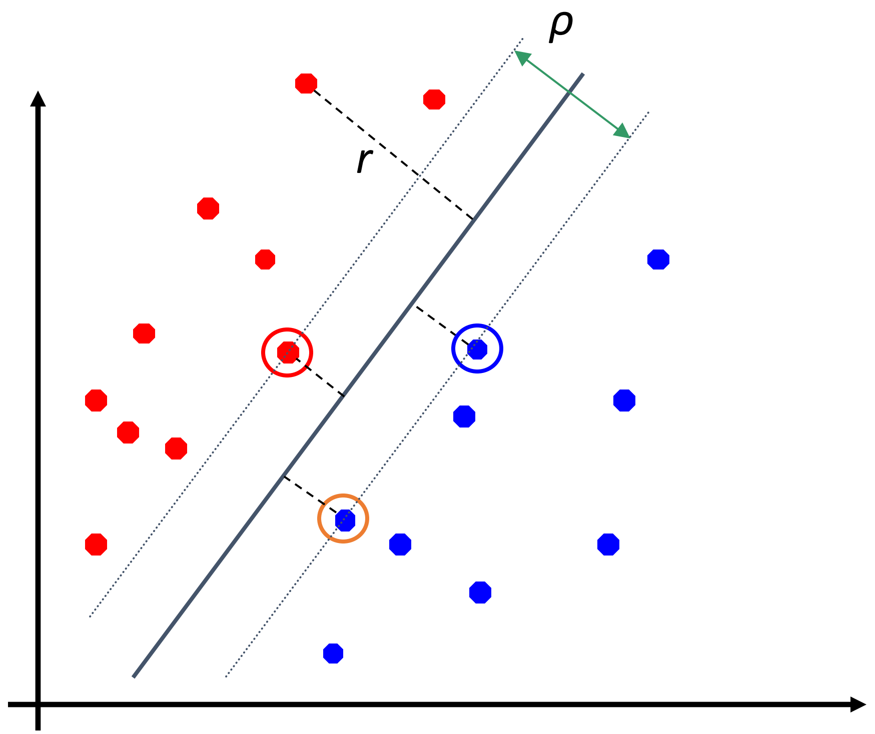
Classification Margin

- Distance from example $\mathbf{x}_i$ to the separator is

$$r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$$

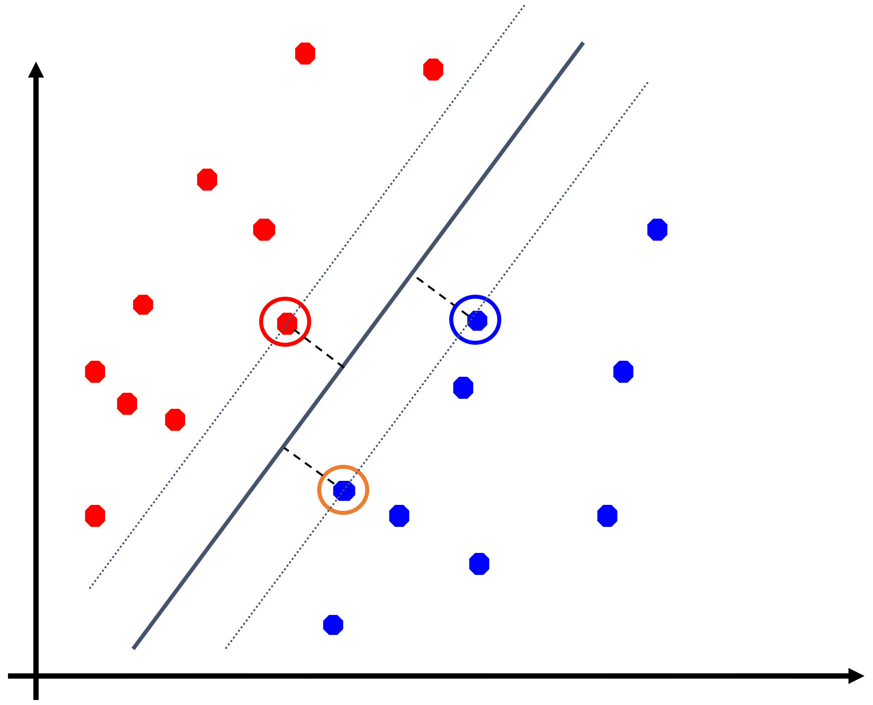
- Examples closest to the hyperplane are *support vectors*.

- *Margin ρ* of the separator is the distance between support vectors



Maximum Margin Classification

- Maximizing the margin is good, according to intuition.
- Only support vectors matter; other training examples are ignorable.



Linear SVM Mathematically

- Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{array}{ll} \mathbf{w}^\mathbf{T}\mathbf{x}_i + b \leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^\mathbf{T}\mathbf{x}_i + b \geq \rho/2 & \text{if } y_i = 1 \end{array} \quad \Leftrightarrow \quad y_i(\mathbf{w}^\mathbf{T}\mathbf{x}_i + b) \geq \rho/2$$

- For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that the distance between each $\mathbf{x}_s$ and the hyperplane is

$$r = \frac{\mathrm{y}_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

- Then, the margin can be expressed through (rescaled) $\mathbf{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVM Mathematically

- Then we can formulate the *quadratic optimization problem:*

Find **w** and *b* such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- Which can be reformulated as:

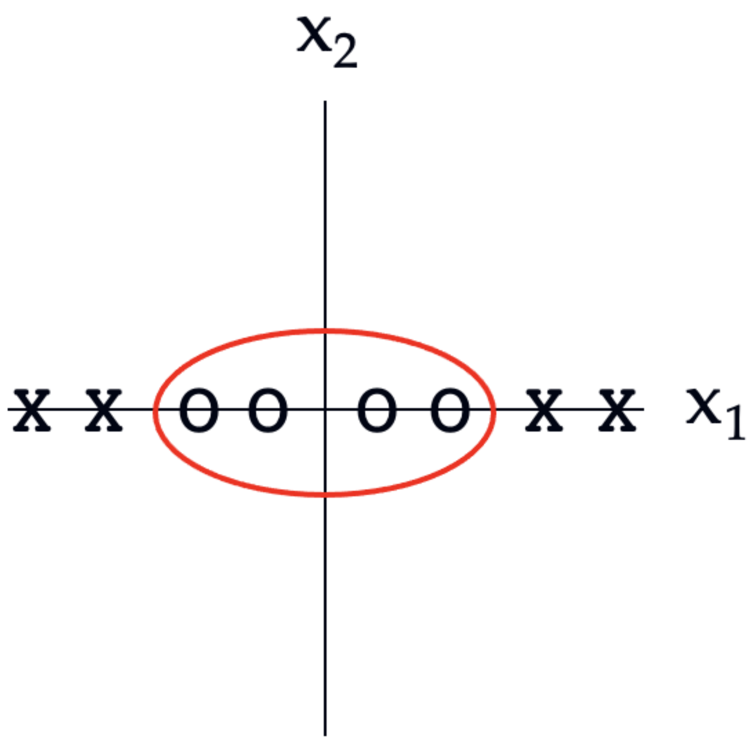
Find **w** and *b* such that

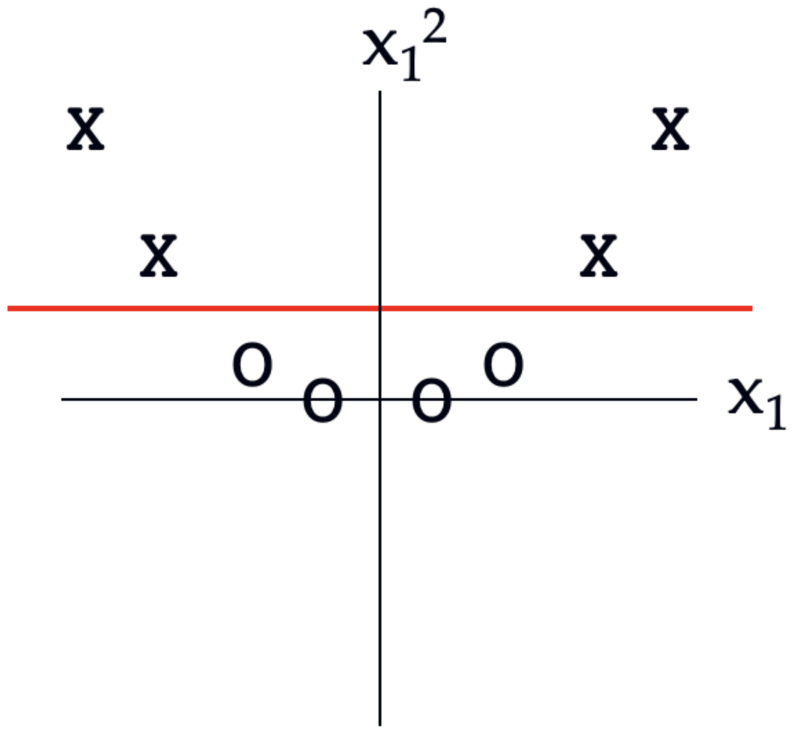
$\Phi(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Non-linear SVM

When Linear Separators Fail

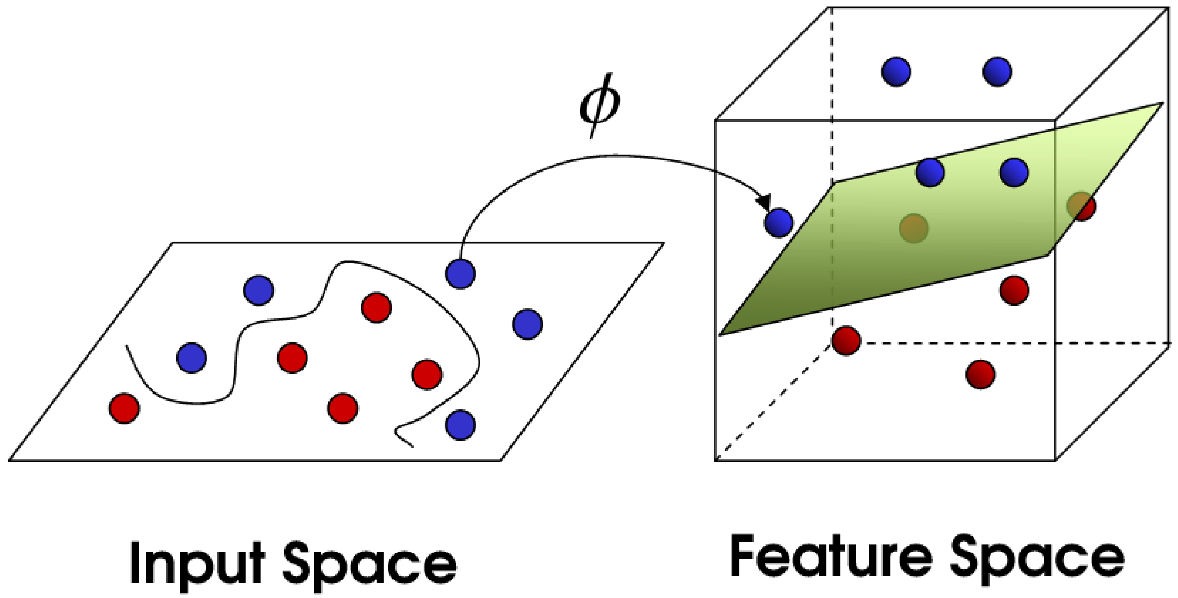




Mapping into a New Feature Space

$$\Phi : x \rightarrow X = \Phi(x)$$

$$\Phi(x_1,x_2) = (x_1,x_2,x_1^2,x_2^2,x_1x_2)$$



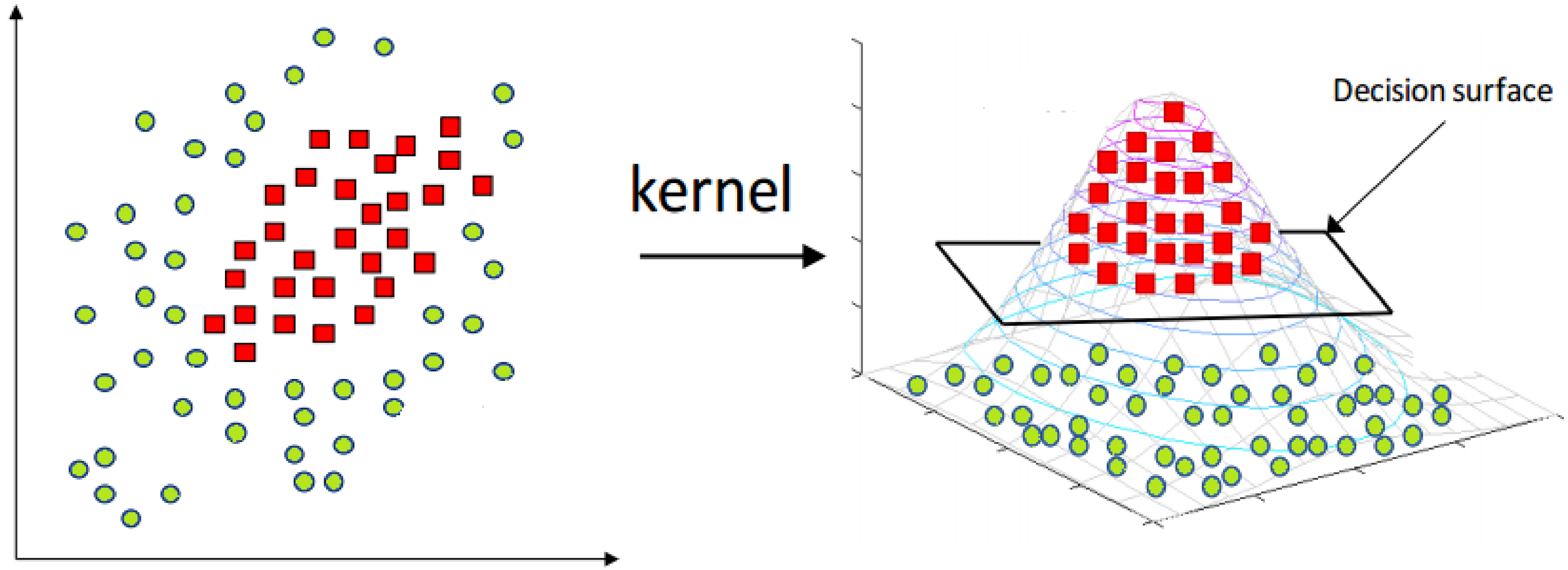
General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable.

Kernel function: mathematical Definition

$$K(x, y) = \phi(x) \cdot \phi(y)$$

- *K* is the Kernel function, *x* and *y* are *n*-dimensional inputs, and ϕ is a mapping function from *n*-dimension to *m*-dimension space.
- $\phi(x) \cdot \phi(y)$ denotes the dot product.
- Usually, *m* is much larger than *n.*

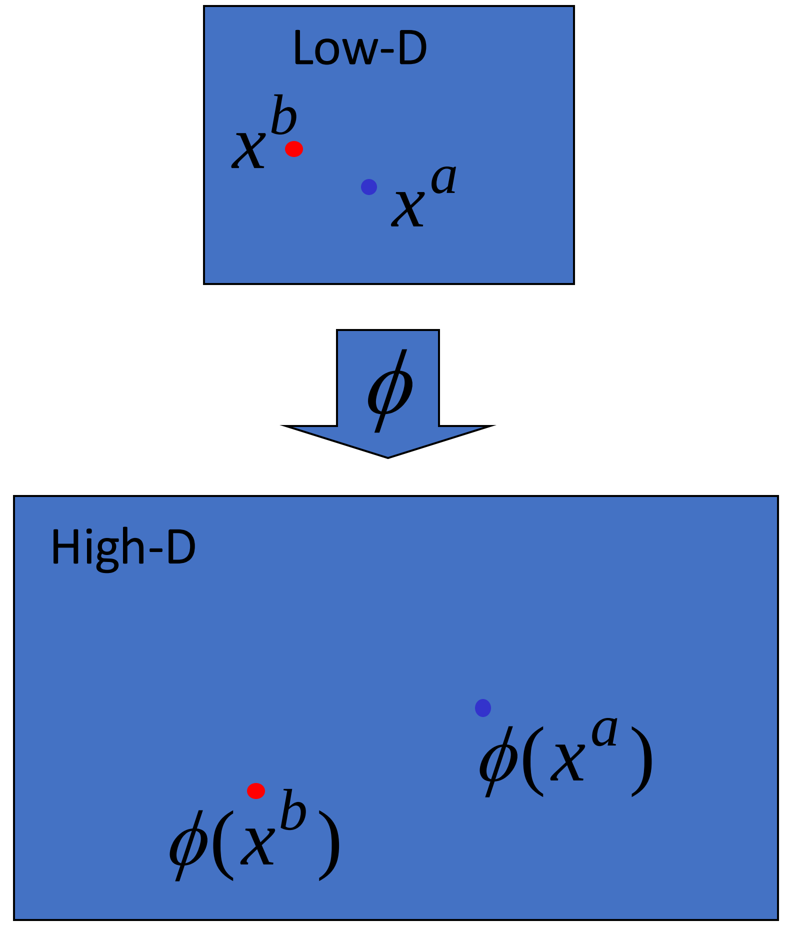
Kernel Function



The Kernel Trick

- For many mappings from a low-D space to a high-D space, there is a simple operation on two vectors in the low-D space that can be used to compute the scalar product of their two images in the high-D space.

$$\underset{\substack{\uparrow \\ \text{Letting the kernel do the work}}}{K(x^a, x^b)} = \underset{\substack{\uparrow \\ \text{doing the scalar product in the obvious way}}}{\phi(x^a) . \phi(x^b)}$$



What the Kernel trick Achieves

- All of the computations that we need to do to find the maximum-margin separator can be expressed in terms of scalar products between pairs of data points (in the high-dimensional feature space)
- These scalar products are the only part of the computation that depends on the dimensionality of the high-dimensional space.
  - So, if we had a fast way to do the scalar products, we would not have to pay a price for solving the learning problem in the high-D space.
- The kernel trick is just a magic way of doing scalar products much faster than is usually possible.
  - It relies on choosing a way of mapping to the high-dimensional feature space that allows fast scalar products

Some Commonly used Kernels

Polynomial: $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}.\mathbf{y} + 1)^p$

Gaussian radial basis function: $K(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}$

Parameters that the user must choose

Neural net: $K(\mathbf{x},\mathbf{y}) = \tanh(k\,\mathbf{x}.\mathbf{y} - \delta)$

For the neural network kernel, there is one "hidden unit" per support vector, so the process of fitting the maximum margin hyperplane decides how many hidden units to use. Also, it may violate Mercer's condition.

Multi-class SVM

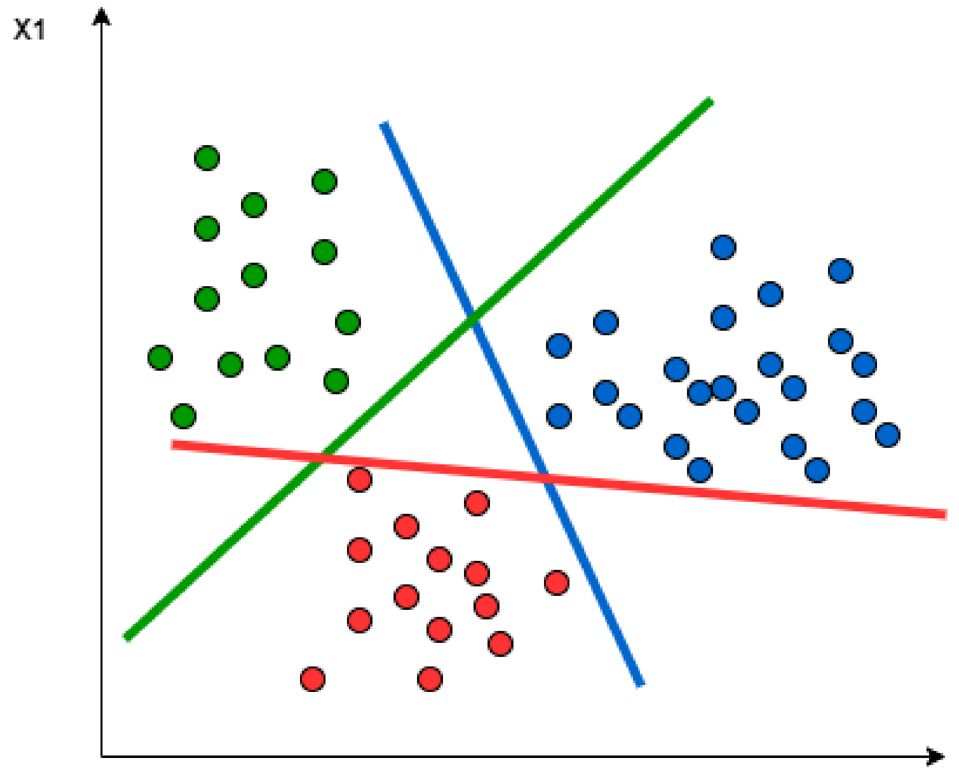
Multi-class Classification

- In its most simple type, SVM doesn’t support multiclass classification natively.
- SVM supports binary classification and separating data points into two classes.
- For multiclass classification, the same principle is utilized after breaking down the multiclassification problem into multiple binary classification problems.

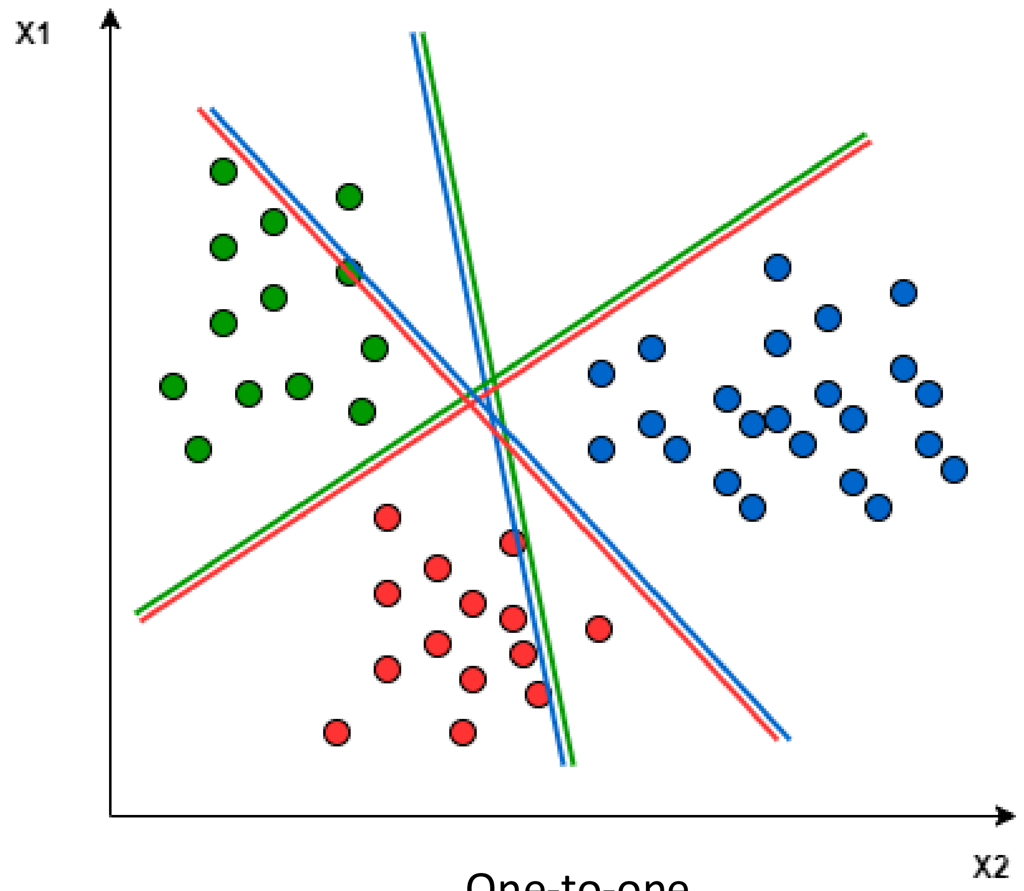
Multi-class Classification

- **One-to-One approach**, which breaks down the multiclass problem into multiple binary classification problems.
- **One-to-Rest**. In that approach, the breakdown is set to a binary classifier per each class.

Multi-class classification



One-to-all



One-to-one

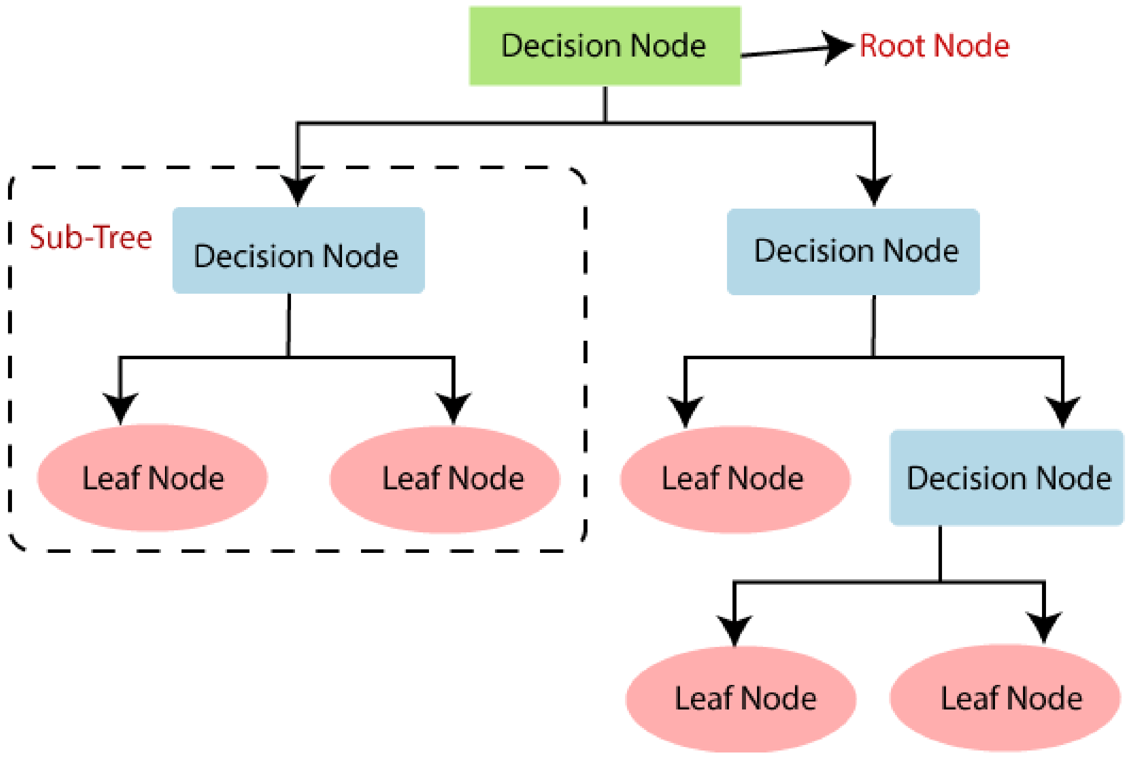
Decision Tree

Introduction

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving classification problems.
- It is a tree-structured classifier, where:
  - internal nodes represent the features of a dataset,
  - branches represent the decision rules and,
  - each leaf node represents the outcome.

Structure

- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node.
- Decision nodes are used to make any decision and have multiple branches,
- Leaf nodes are the output of those decisions and do not contain any further branches.



Note: A decision tree can contain categorical data (YES/NO) as well as numeric data.

Terminologies

- Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- Branch/Sub Tree: A tree formed by splitting the tree.
- Pruning: Pruning is the process of removing the unwanted branches from the tree.
- Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

Decision Principle

- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- Similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, the CART algorithm is used which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.

Working Principal

- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree.
- It compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.
- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further.
- It continues the process until it reaches the leaf node of the tree.

Algorithm Steps

- **Step-1:** Begin the tree with the root node, says *S*, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- **Step-3:** Divide the *S* into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

How to Measures Attribute Selection?

- While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes.
- To solve such problems there is a technique which is called as Attribute selection measure (ASM).
- By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:
  - Information Gain
  - Gini Index

Information Gain

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.
- It can be calculated using the below formula:

Information Gain = Entropy(S) - [(Weighted Avg) * Entropy(each feature)]

Entropy

- Entropy is a metric to measure the impurity in a given attribute.
- It specifies randomness in data.
- Entropy can be calculated as:

$$\text{Entropy}(s) = -P(\text{yes}) \log 2\ P(\text{yes}) - P(\text{no}) \log 2\ P(\text{no})$$

where,

- S = Total number of samples
- P(yes) = probability of yes
- P(no) = probability of no

Gini Index

- Gini index is a measure of impurity or purity used while creating a decision tree in the Classification and Regression Tree (CART) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$\text{Gini Index} = 1 - \sum_j P_j^2$$