Week-5

SVM Recap and Project Implementation

V Semester - ML and AIUP, Aug-Oct 2024

Session Date and Time: 20th Oct 2024 10:30 AM IST – 12:00 Noon IST

Venkateswar Reddy Melachervu
Visiting Faculty and CTO, Brillium Technologies
Department of Computer Science Engineering – AI and ML (CSM)
Email: venkat.reddy.gf@gprec.ac.in

G.Pulla Reddy Engineering College (Autonomous)
G.Pulla Reddy Nagar, Nandyal Road, Kurnool, AP 518007, India
Website: https://www.gprec.ac.in

Disclaimer and Confidentiality Notice

The content of this guest lecture, including all discussions, materials, and demonstrations, code is intended for educational purposes only and reflects the views and opinions of the speaker. While every effort has been made to ensure the accuracy and relevance of the information presented, it should not be considered as legal, financial, or professional advice.

Brillium Technologies retains unrestricted ownership of all information shared during this session. Participants must not record, reproduce, distribute, or disclose any part of the lecture or materials without prior written permission from Brillium Technologies. Unauthorized use or distribution of the content may result in legal action.

Additionally, all trademarks, service marks, and intellectual properties referenced or used in this presentation are the property of their respective owners. No ownership or rights over such third-party content are claimed or implied by the author or Brillium Technologies or by GPREC.

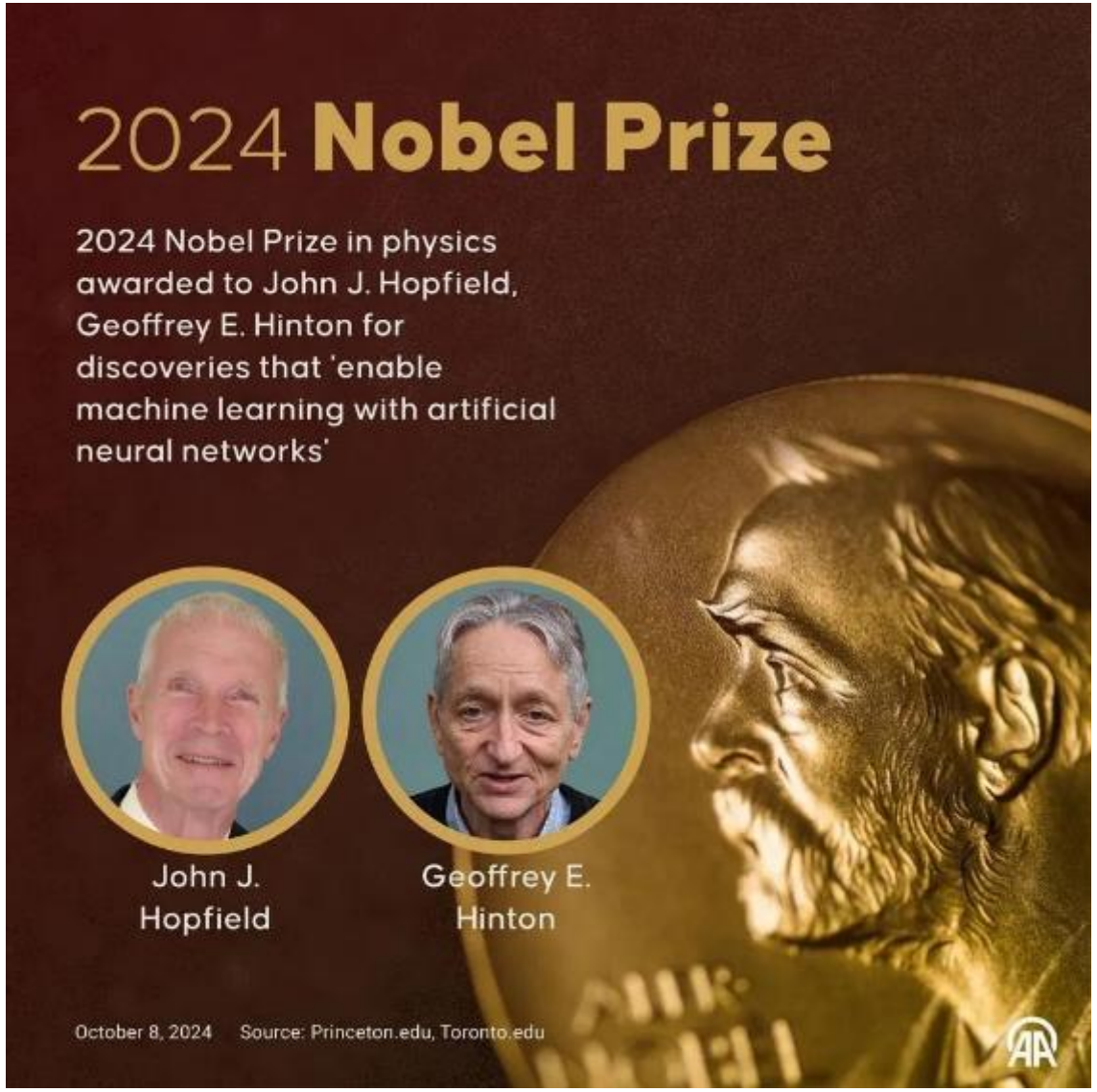
By attending this lecture, you agree to respect the confidentiality of the information shared and refrain from using it in any unauthorized manner. Failure to comply with these terms may result in legal action.

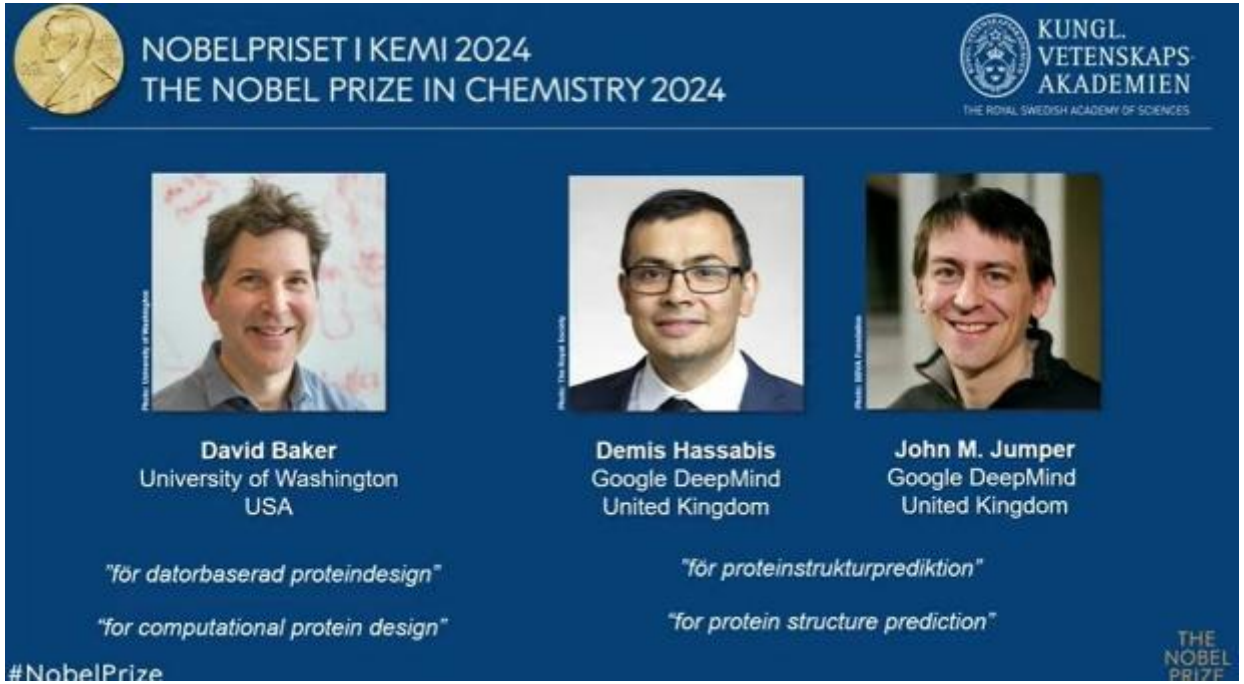
Thank you for your understanding and cooperation.

Lecture Outline

- Classification
- Real-world Classification Use Cases
- Classification Algorithms and Comparison
- Linear Classifiers
- Maximum Margin Classifier
- Constrained Optimisation – Lagrangian Duality
- Primal and Dual SVM
- Non-linear Data and Higher Dimensional Classification
- Kernel Tricks and Functions
- Common Kernel Functions
- Kernel SVM and Decision Boundaries
- Kernel Tricks Vs PCA
- SVM with Slack Variables
- Project Demo
- Q&A

AI Steals The Spotlight at 2024 Nobel Prizes

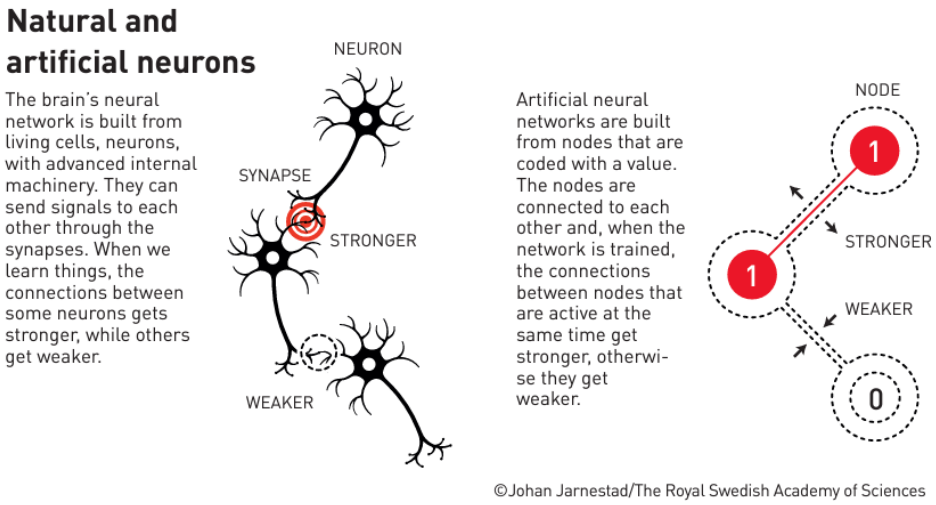


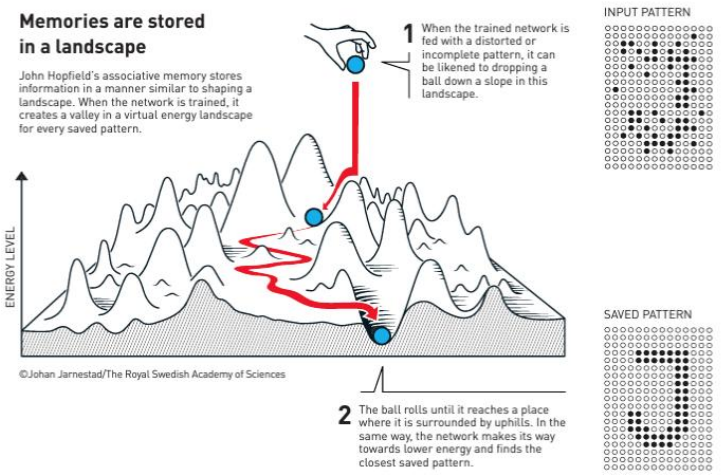


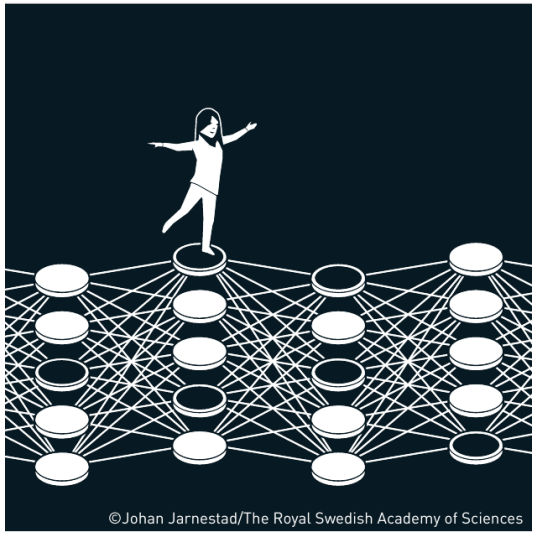
David Baker
University of Washington, Seattle, WA, USA
Howard Hughes Medical Institute, USA
"for computational protein design"
and the other half jointly to
Demis Hassabis
Google DeepMind, London, UK
John M. Jumper
Google DeepMind, London, UK
"for protein structure prediction"

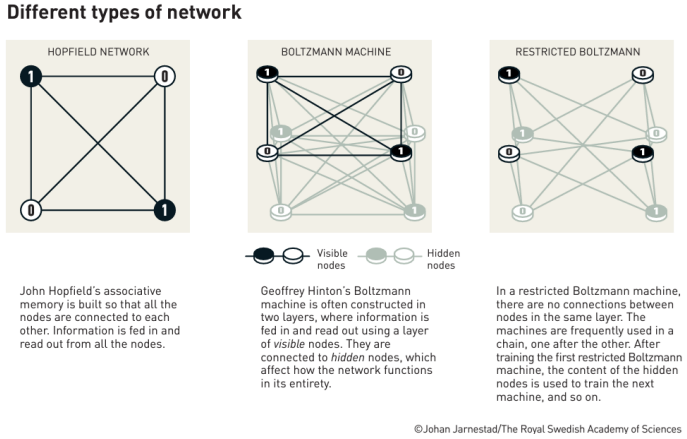
Source : Press release: 8th Oct 2024 - The Nobel Prize in Physics 2024 - NobelPrize.org

AI Steals The Spotlight at 2024 Nobel Prizes



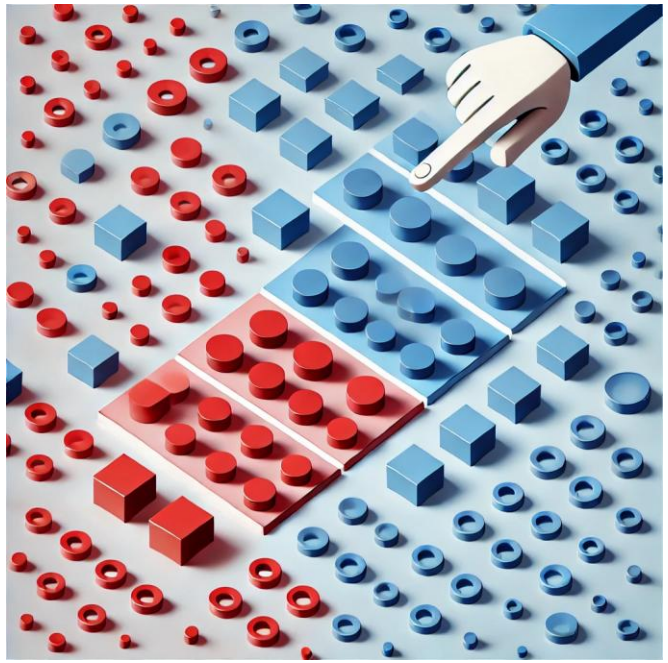






Classification

Image is generated by ChatGPT 4o for this Lecture



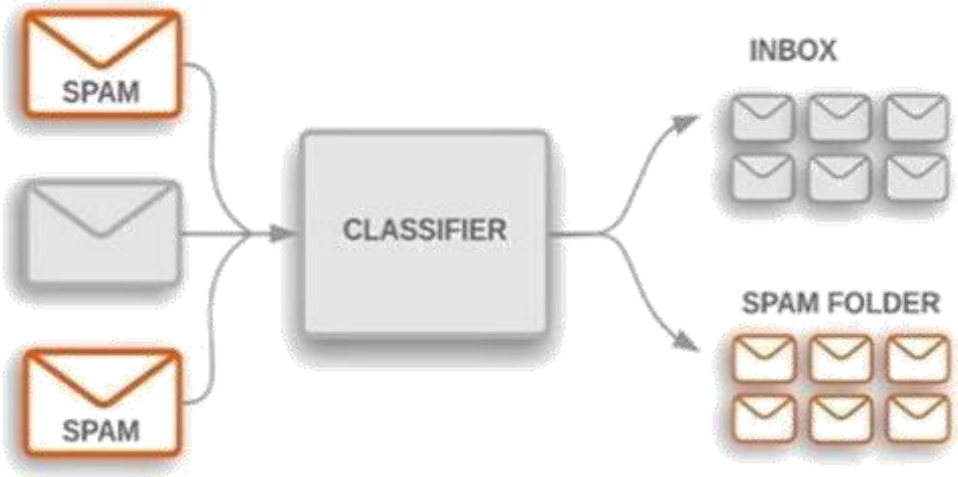


Image Courtesy : (1) Classification Algorithms in Machine Learning. | LinkedIn

- Classification is the process of assigning data into classes or categories - predefined or otherwise, based on its features
- Determines to which class an observed data belongs to
- **The decision boundary** separates two distinct groups, each representing a different class
- In the image on to the left, a classifier distinguished red circles and squares belonging to one class from blue circles and squares belonging to another
- **Support Vector Machines - SVM -** and other classification algorithms learn to create such boundaries for accurate **data class predictions**
- The goal of classification is to make decisions based on these separated groups
  - Identifying a received email as spam
  - Marking a credit card transaction as fraud
  - Defining customer categories

Real-World Use Cases of Classification

Decision Making and Prediction

- Allows us to predict categorical outcomes and make informed decisions based on historical data
- Example : Healthcare - Predict whether a patient has a disease or not based on medical data

Tasks Automation

- Automation of sorting and categorizing data which otherwise would be laborious and error-prone when done by humans
- Example : Spam classifier/detector in email server to automatically sort and filter each received emails into span and genuine emails

Understanding Patterns and Trends

- Helps us uncover hidden patterns in data that would otherwise not be immediately obvious through manual inspection
- Example : Online retail companies use classification to understand customer behaviour to identify whether a customer is likely to purchase a product based on their browsing history

Real-World Use Cases of Classification

Improved Decision Making in Business

- Companies use classification to segment customers, predict customer churn, and to target marketing efforts more effectively for growing business
- Example : A bank can use classification to determine which customers are likely to default on a loan, allowing them to mitigate risks in loan repayment

Handling Complex High-Dimensional Data

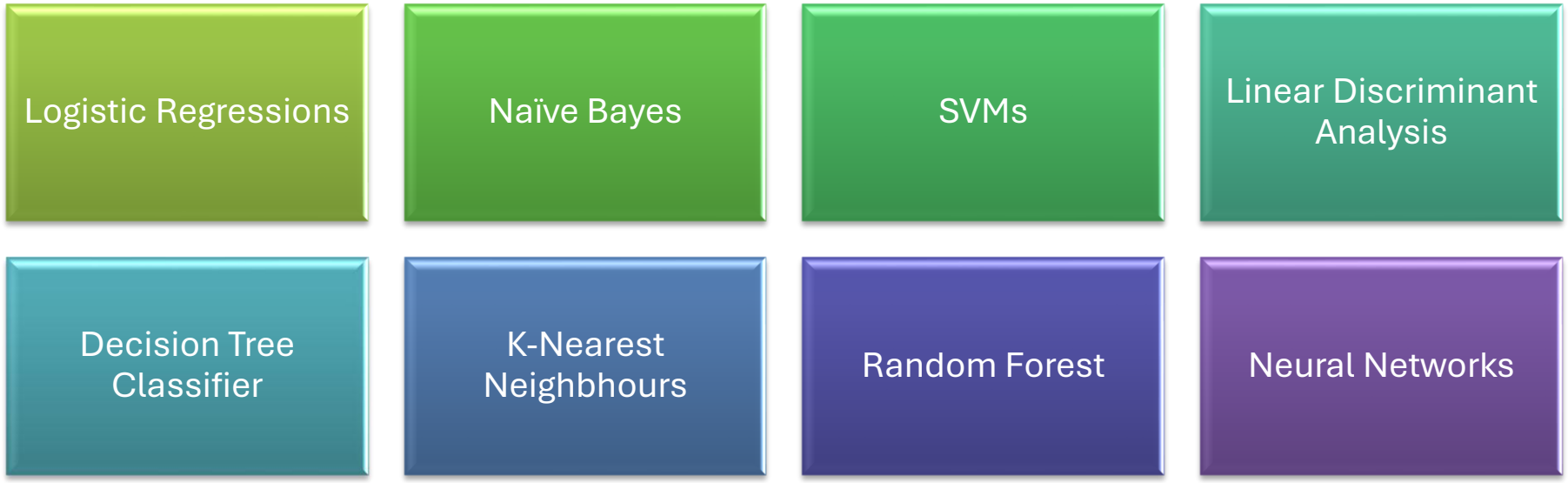
- Advanced classification algorithms like SVM or deep learning, can find patterns in complex, high-dimensional datasets that humans, otherwise, would struggle to analyze
- Example : In facial recognition or autonomous vehicle driving, classification algorithms help identify objects or people in images or in live video

NLP

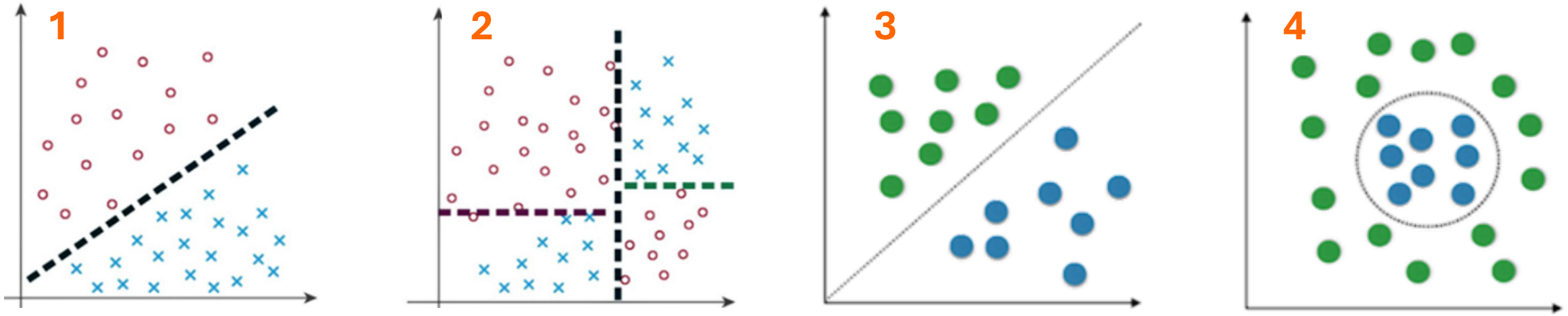
- Sentiment analysis and text classification

Classification Algorithms

- Based on Output
  - Binary Classifiers – 2 output classes
    - Logistic Regression, Simple Naïve Bayes, etc.
  - Multinomial Classifiers – More than 2 output classes
    - Multi-monomial Logistic Regression/ Softmax, Non-linear Naïve Bayes, KNN, Decision Tree Classifier, etc.
- Based on Data Characteristics
  - Linear Classifier - Classify data using a linear decision boundary - a straight line or hyperplane
  - Non-Linear Classifier - Classifying using complex, non-linear decision boundaries



Classification Algorithms – Linear and Non-linear Classifiers



Linear Classifiers

- Examples : Logistic Regression, SVM, Linear Discriminant Analysis, etc.
- **Best for**
  - Data that can be separated by a straight line (linearly separable data).
- **Advantages**
  - Simpler and faster to compute
  - Easier to interpret
  - Work well for high-dimensional, sparse data (e.g., text classification)
- **Limitations**
  - Cannot handle complex, non-linear patterns in data

Non-linear Classifiers

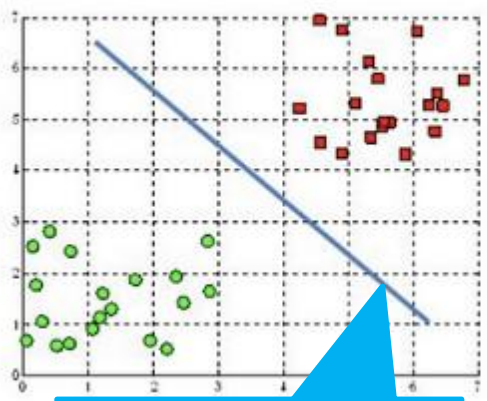
- Example : Kernel SVM, DTC, Random Forest, KNN, Neural Networks, etc.
- **Best for**
  - Data with complex, non-linear relationships
- **Advantages**
  - Can model complex and intricate patterns
  - More flexible, able to handle diverse types of data
- **Limitations**
  - Often computationally more expensive
  - May require more data to train
  - More challenging to interpret

Key Comparison

Decision Boundary	• Linear classifiers use straight lines or hyperplanes, while non-linear classifiers can create curves or complex boundaries
Performance	• Non-linear classifiers generally perform better when data isn't linearly separable
Computational Cost	• Linear classifiers are faster and simpler • Non-linear ones are more computationally expensive but powerful for complex tasks

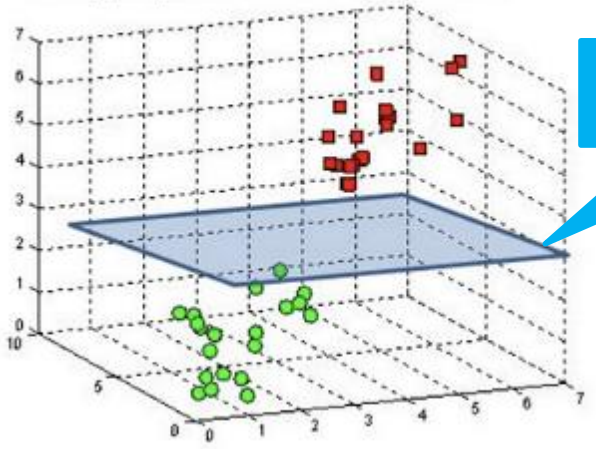
Classifier Decision Boundaries

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



$ax = b$

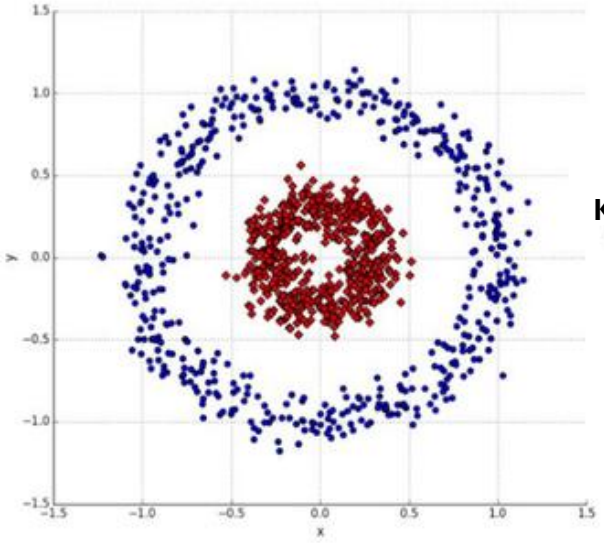
A hyperplane in $\mathbb{R}^3$ is a plane



$\bar{a}^T \bar{x} = b$

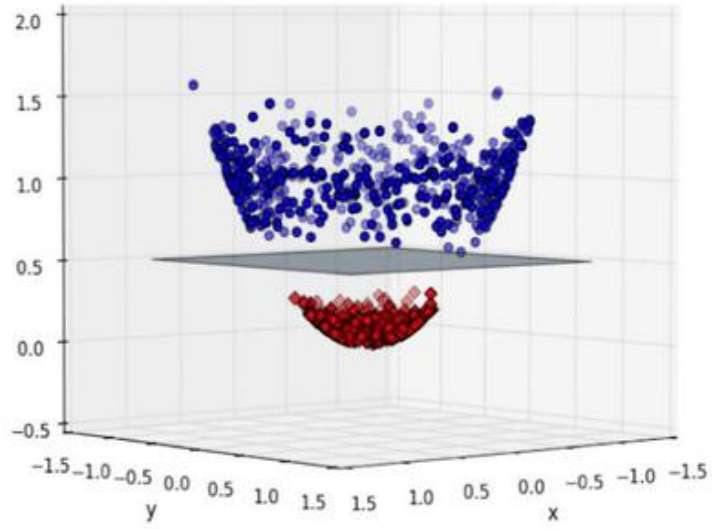
$$\text{2-D} : a_1x_1 + a_2x_2 = b$$
$$\text{3-D} : a_1x_1 + a_2x_2 + a_3x_3 = b$$
$$\text{n-D} : a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

2D

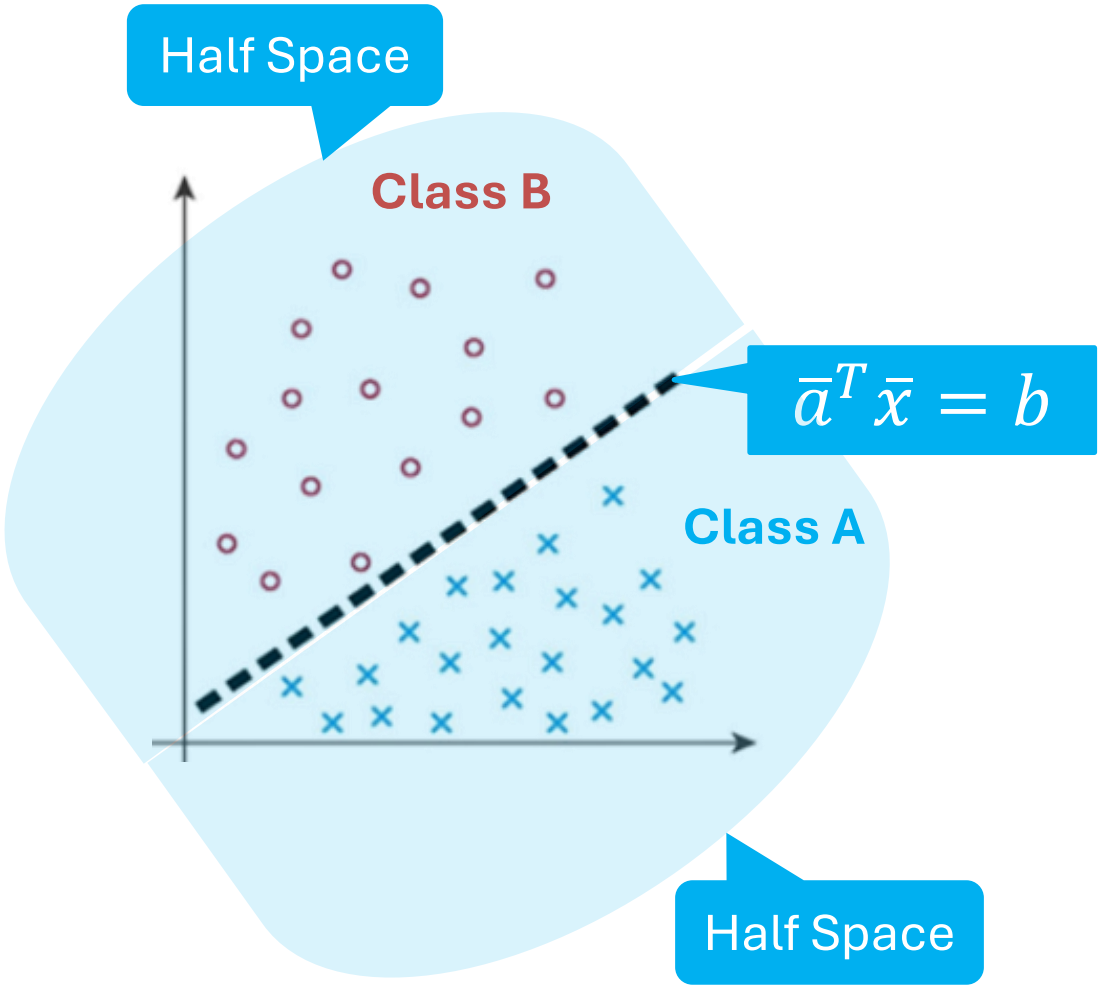


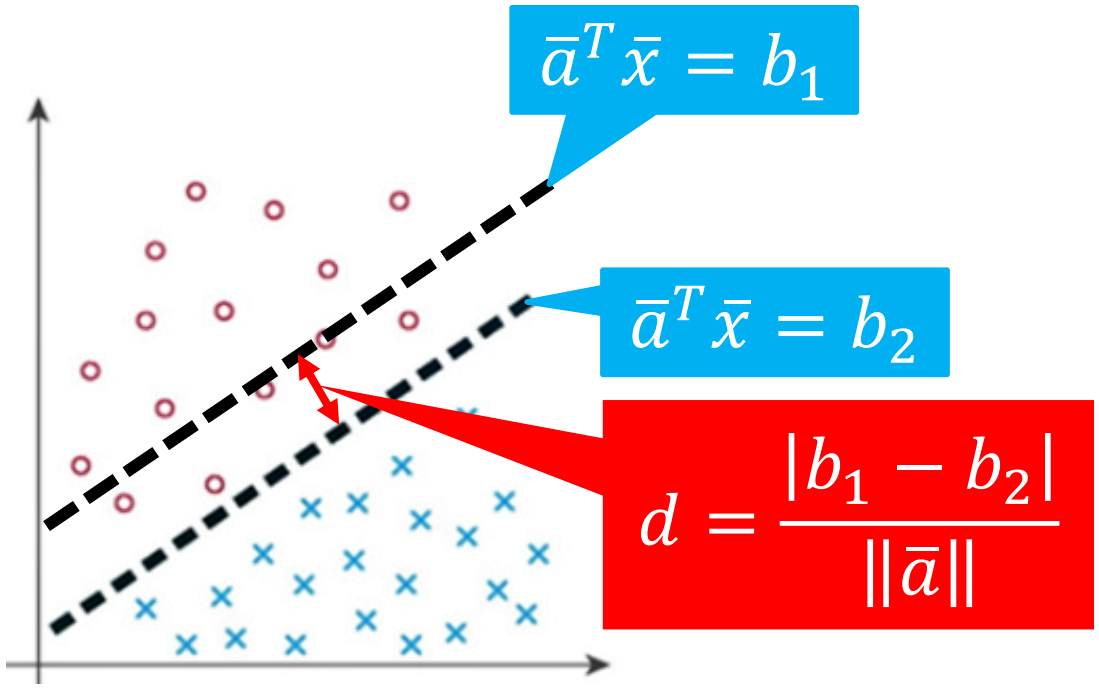
SVM – Kernel Trick

3D



Linear Classifier





Two Parallel Lines/Planes

General Structure of a Linear Classifier:

Class A: $\mathcal{C}_A$: $\bar{a}^T\bar{x} \leq b$

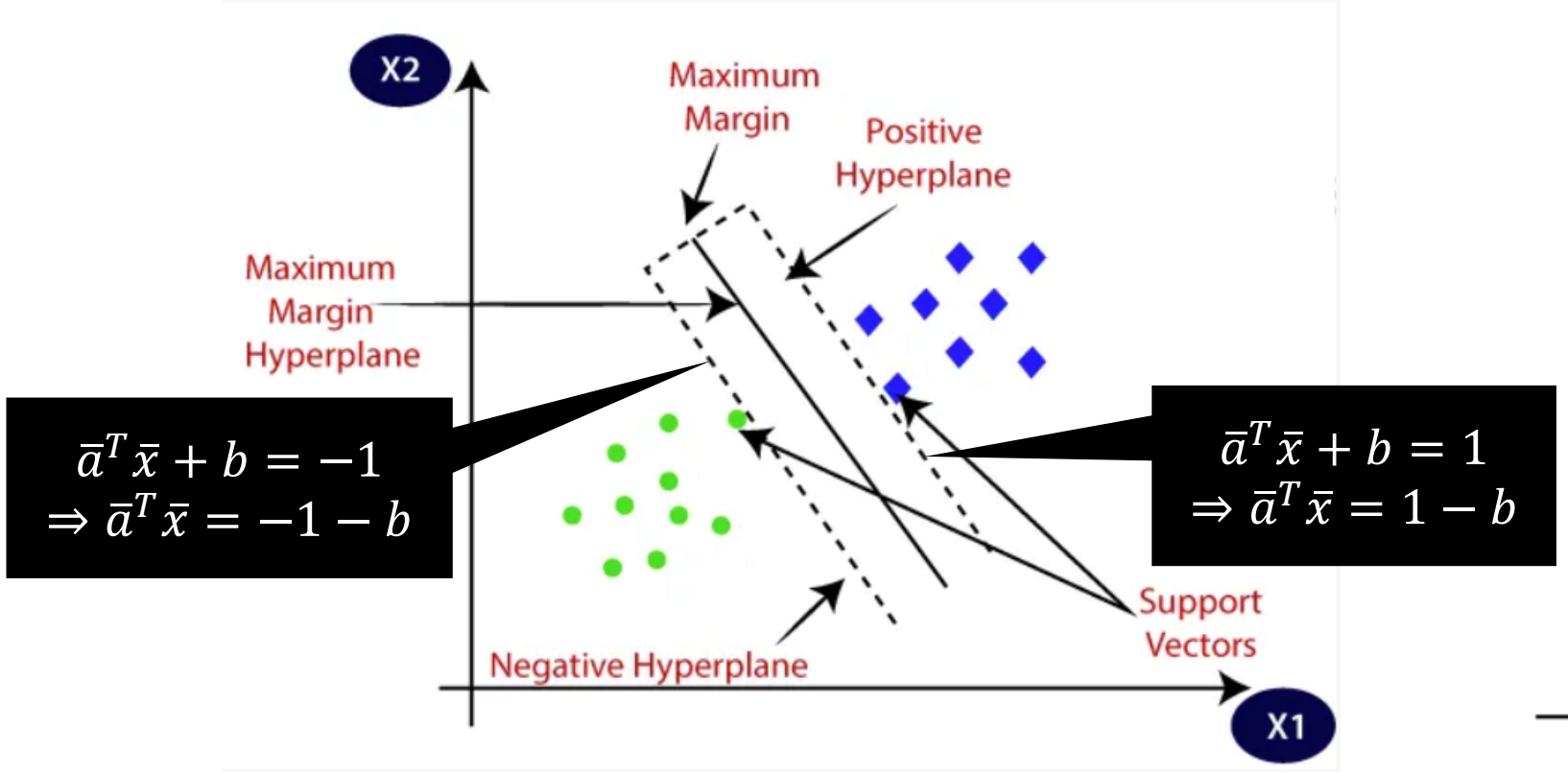
Class B: $\mathcal{C}_B$: $\bar{a}^T\bar{x} > b$

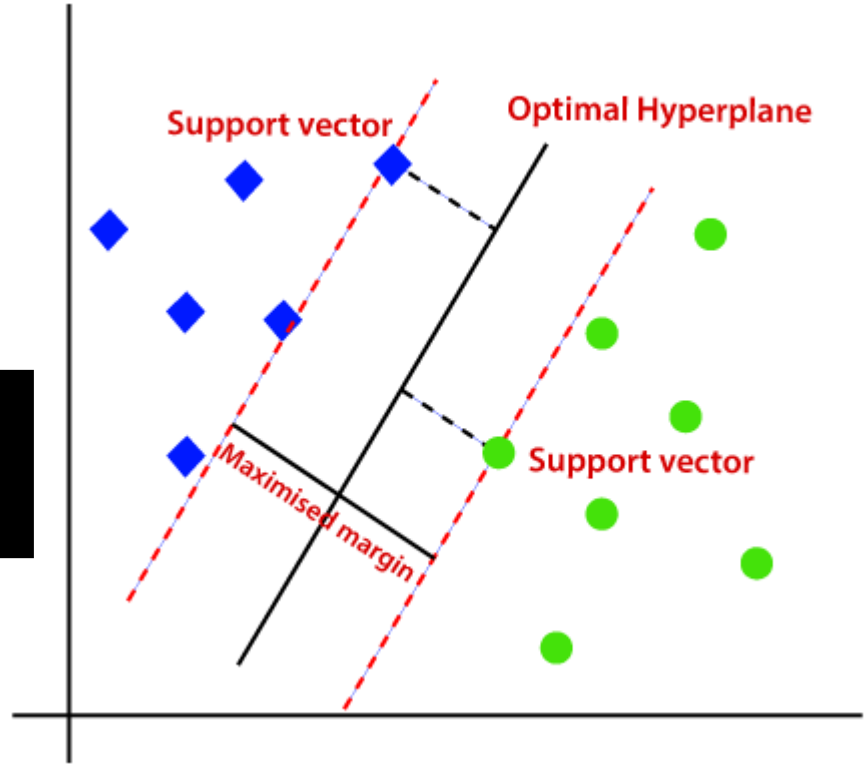
Distance Between Two Parallel Lines:

$$d = \frac{|b_1 - b_2|}{\sqrt{a_1^2 + a_2^2 + \cdots a_n^2}}$$

Support Vector Machines – Maximum Margin Classifier

- Objective of SVM algorithm is to find **an optimal hyperplane/decision boundary** in an n-dimensional data space that separates classes with the **maximum margin**





Margin – Distance between the positive and negative hyperplanes:

$$\boldsymbol{d} = \frac{|\mathbf{1} - \boldsymbol{b} - (-\mathbf{1} - \boldsymbol{b})|}{\|\bar{\boldsymbol{a}}\|} = \frac{|\mathbf{2}|}{\|\bar{\boldsymbol{a}}\|}$$

Linear Maximum Margin Classifier – SVM Classifier

- Maximize the margin/distance (d) between the positive and negative hyperplanes for all data points in training set $1 \dots 2M$

Convex Function

$$\boldsymbol{max}\ \frac{\boldsymbol{2}}{\|\bar{\boldsymbol{a}}\|} = \boldsymbol{min}\|\bar{\boldsymbol{a}}\| = \boldsymbol{min}\ \|\bar{\boldsymbol{a}}\|_{\boldsymbol{2}} \approx \boldsymbol{min}\ \frac{\boldsymbol{1}}{\boldsymbol{2}}\|\bar{\boldsymbol{a}}\|^{\boldsymbol{2}}$$

Objective Function

Constrained Optimisation Problem

Subject to:

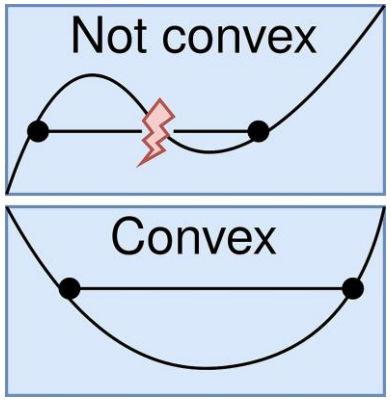
$$\bar{\boldsymbol{a}}^{\boldsymbol{T}}\bar{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{b} \geq \boldsymbol{1}, \boldsymbol{1} \leq \boldsymbol{i} \leq \boldsymbol{M}$$

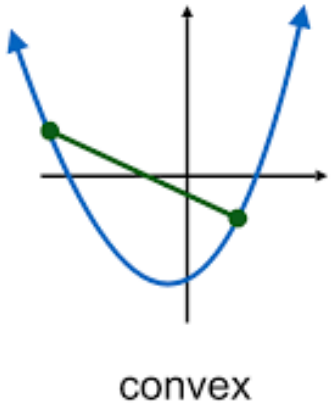
$$\bar{\boldsymbol{a}}^{\boldsymbol{T}}\bar{\boldsymbol{x}}_{\boldsymbol{i}} + \boldsymbol{b} \leq \boldsymbol{-1}, \boldsymbol{M+1} \leq \boldsymbol{i} < \boldsymbol{2M}$$

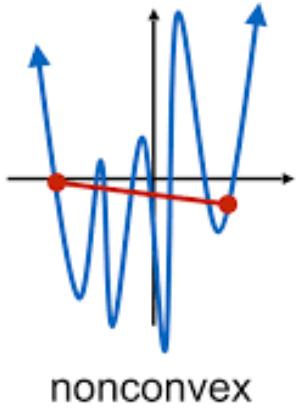
Constraints

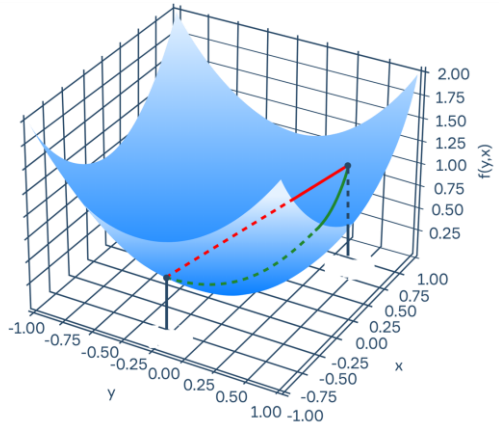
Convex Optimisation Problem

- Convex Function
  - The line segment connecting any two points on the function graph, lies above or on the graph of the function
  - $f(x)$ is convex, if for any two points $\bar{x}_1$ and $\bar{x}_2$ in its domain and for any $\lambda \in [0,1]$, the inequality $f(\lambda\bar{x}_1 + (1-\lambda)\bar{x}_2) \leq \lambda f(\bar{x}_1) + (1-\lambda)f(\bar{x}_2)$
    - $\forall\ \bar{x}_1, \bar{x}_2 \in \mathcal{D}\ and\ \lambda \in [0,1] \Rightarrow f(\lambda\bar{x}_1 + (1-\lambda)\bar{x}_2) \leq \lambda f(\bar{x}_1) + (1-\lambda)f(\bar{x}_2)$
  - In simpler terms, a convex function curves **upward** or is **bowl-shaped**









Constrained Optimisation – Lagrangian Duality

- Standard optimisation problem form - aka **Primal Problem**

  $p^* = min\, f(x)$

  Subject to (Constraints):

  $g_i(x) \leq 0, i = 1,2, \ldots m$ – Inequality constraints

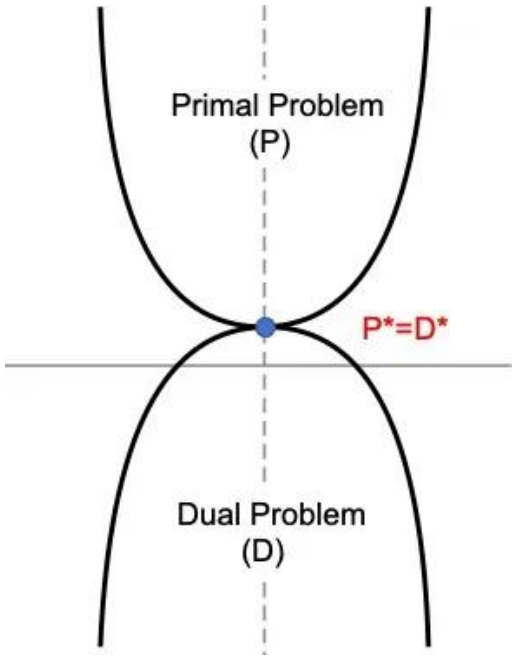
  $h_j(x) \geq 0, j = 1,2, \ldots p$ – Inequality constraints

  $c_k(x) = 0, k = 1,2, \ldots t$ – Equality constraints

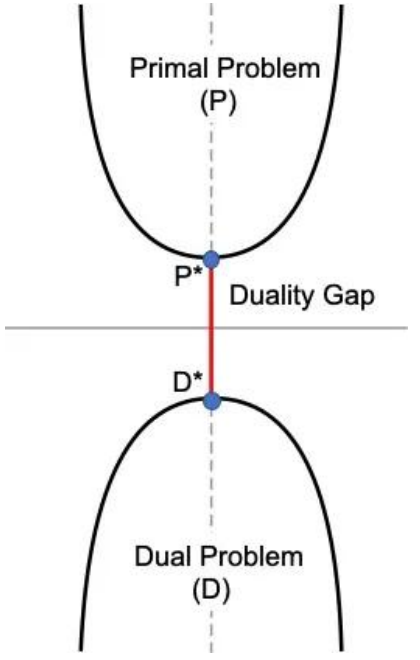
  $x \in \mathbb{R}^n$, domain $\mathcal{D}$, optimal value p^*
- Lagrangian Duality aka **Dual Optimisation Problem**
  - A simplified **Lagrange Function** aka **Lagrangian** is formulated:
    - By combining **objective function** and **constraints** using **Lagrange multipliers** (λ, μ, ν)
    - This mathematical formulation reflects the **impact of constraints** in the solution
    - $\mathcal{L}(x, \lambda, \mu, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) - \sum_{j=1}^{p} \mu_j h_j(x) + \sum_{k=1}^{t} \nu_k c_k(x)$ where $\lambda_i \geq 0, \mu_j \leq 0, \nu_k$ - no constraints
  - Minimize the **Lagrangian** with respect to the original/primal variable x while treating the Lagrange multipliers as fixed
    - $\phi(x, \lambda, \mu, \nu) = inf/\min_{x} \mathcal{L}(x, \lambda, \mu, \nu)$
  - Result is **dual function $\boldsymbol{\phi(\lambda, \mu, \nu)}$** and depends only on Lagrange multipliers λ, μ, ν
  - A **dual problem** is constructed to **maximise** the **dual function** w.r.t to Lagrange multipliers subject to the feasibility conditions on Lagrange multipliers
    - $\max_{\lambda \geq 0, \mu \leq 0, \nu} \phi(\lambda, \mu, \nu)$
- When the **maximum margin classifier** is solved using **Lagrangian Duality ⇒ Dual SVM**

Primal SVM and Dual SVM

- Primal SVM
  - Directly solves for the hyperplane/decision boundary in feature space
  - Suitable for low-dimensional problems
  - The **maximum margin classifier** is solved using **primal optimisation problem**
- Dual SVM
  - Preferred when the number of features is larger than the number of samples
  - Instead of solving for the weights $\vec{a}$ and b of the hyperplane directly from **primal optimisation problem**, dual SVM solves for the **Lagrange multipliers** making it more flexible for non-linear decision boundaries
  - Reduced size of the problem - from a high-dimensional feature space to a lower-dimensional dual space
  - The m**aximum margin classifier** is solved using **dual optimisation problem**



Strong Duality



Weak Duality

Dual SVM Solution

- Solve $\boldsymbol{min}\frac{1}{2}\|\bar{a}\|^2$

  Subject to:

  $\bar{a}^T\bar{x}_i + b \geq 1, 1 \leq i \leq M$

  $\bar{a}^T\bar{x}_i + b \leq -1, M+1 \leq i < 2M$

- Two constraints can be combined into one

  $y_i(\bar{a}^T\bar{x}_i + b) \geq 1, \forall i$

  Where:

$$y_i = \begin{cases} +1, & i = 1,2,\dots M \\ -1, & i = M+1,\dots 2M \end{cases}$$

- Lagrangian

$$\mathcal{L}(\bar{a}, b, \lambda) = \frac{1}{2}\|\bar{a}\|^2 - \sum_{i=1}^{2M}\lambda_i[y_i(\bar{a}^T\bar{x}_i + b) - 1] = \frac{1}{2}\bar{a}^T\bar{a} - \sum_{i=1}^{2M}\lambda_i[y_i(\bar{a}^T\bar{x}_i + b) - 1]$$

- Solving for $\bar{a}, b$ - minimising $\mathcal{L}(\bar{a}, b, \lambda)$
  - $\phi(\lambda, \bar{x}) = \underset{x}{inf/min}\,\mathcal{L}(\lambda)$
  - With respect to a
    - $\frac{\partial\mathcal{L}}{\partial a} = \nabla_{\bar{a}}(\mathcal{L}(\bar{a}, b, \lambda) = \bar{a} - \sum_{i=1}^{2M}\lambda_i y_i x_i = 0 \Rightarrow \boldsymbol{\bar{a} = \sum_{i=1}^{2M}\lambda_i y_i x_i}$
    - The points for which $\boldsymbol{\lambda_i \neq 0}$ are called **Support Vectors**
  - With respect to b
    - $\frac{\partial\mathcal{L}}{\partial b} = \nabla_b(\mathcal{L}(\bar{a}, b, \lambda) = -\sum_{i=1}^{2M}\lambda_i y_i = 0 \Rightarrow \boldsymbol{\sum_{i=1}^{2M}\lambda_i y_i = 0}$
  - Substituting $\bar{a}$ and b in Lagrangian and simplifying for **dual function** $\boldsymbol{\phi(\bar{x}, \lambda) = \sum_{i=1}^{2M}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{2M}\lambda_i\lambda_j y_i y_j \bar{x}_i^T\bar{x}_j}$

Dual SVM Solution contd...

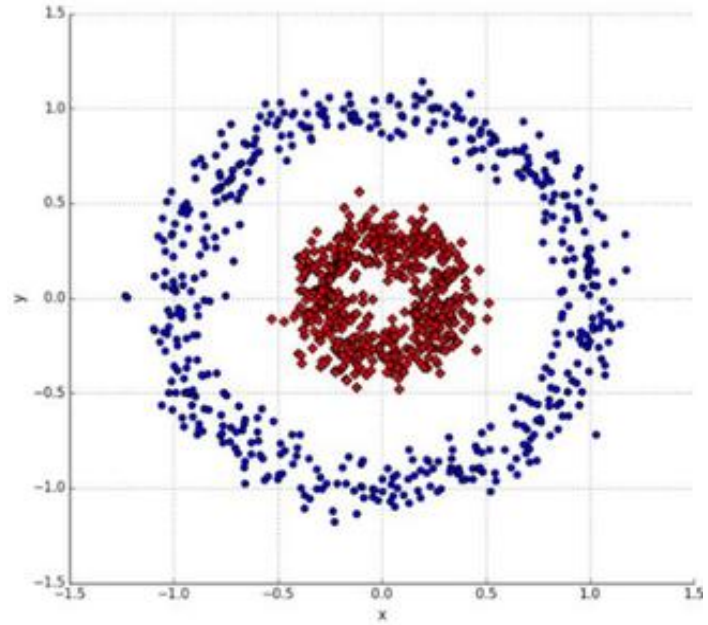
- Therefore **dual problem** - $\max_{\lambda \geq 0} \phi(\lambda, \bar{x})$ can be formulated as:
  - $max \left[\sum_{i=1}^{2M} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{2M} \lambda_i \lambda_j y_i y_j \bar{x}_i^T \bar{x}_j \right]$

  Subject to:
  $\lambda_i \geq 0, \forall i$
  $\sum_{i=1}^{2M} \lambda_i = 0$
- Dual problem is a quadratic programming problem – objective is quadratic and constraints are linear
- Solved for λ_i using standard quadratic solvers
  - Python and SVM solvers – cvxopt, libsvm, sklearn.svm etc.
  - $\bar{a}$ and b are computed as below
    - $\bar{a} = \sum_{i=1}^{2M} \lambda_i y_i \bar{x}_i$
    - b is found by using any one support vector ($\lambda_i > 0$) and solving $y_i(\bar{a}^T \bar{x}_i + b) = 1$
- New data point classification
  - $f(\bar{x}, b) = \bar{a}^T \bar{x} + b > 0 \Rightarrow$ data point is classified as $y = 1$ – **Class 1**
  - $f(\bar{x}, b) = \bar{a}^T \bar{x} + b < 0 \Rightarrow$ data point is classified as $y = -1$ – **Class 0**
- In its simplest form, SVM works well for linearly separable data
- In real-world, data is often **not linearly separable** in the original input space

How can we achieve linear separability for a non-linearly separable real world data?

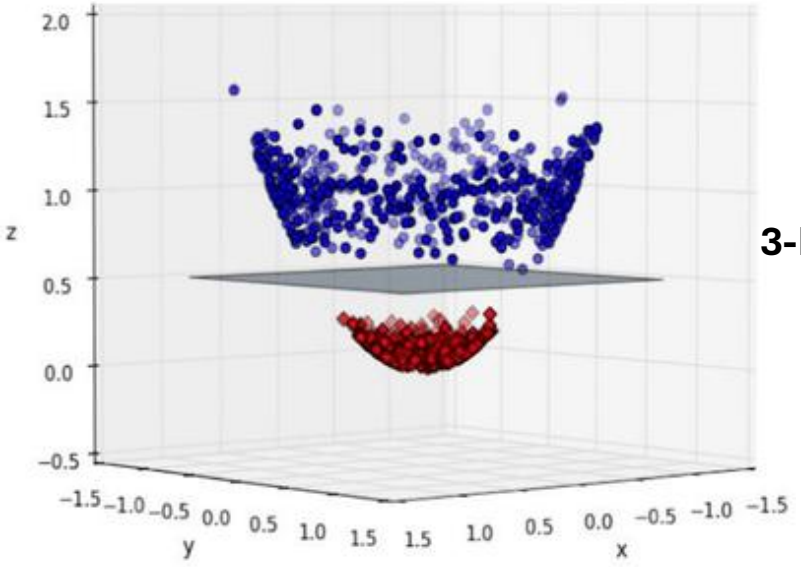
Non-linear Data and Higher Dimensional Classification

- Dual problem
  - $\boldsymbol{max}\left[\sum_{i=1}^{2M}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{2M}\lambda_i\lambda_j y_i y_j \bar{x}_i^T \bar{x}_j\right]$ Subject to: $\lambda_i \geq 0, \forall i$ and $\sum_{i=1}^{2M}\lambda_i = 0$
- Data points $\bar{x}_i, \bar{x}_j$ appear in the form of inner dot product - $\bar{x}_i{}^T \bar{x}_j$ dot/inner product - $\langle \bar{x}_i, \bar{x}_j \rangle$
- A **non-linear data set** in its **current dimensional space χ** might become **linearly separable** if **transformed** into a **higher-dimensional space ν**
  - Set of 2-D points forming concentric circles – non-linear data set
- The input dataset is transformed into a higher dimensional space using **mapping function $\phi(\bar{x})$**
  - Example : 2-D point (x_1, x_2) is transformed into 3-D space using $\phi(\bar{x}) = \left(x_1^2, \sqrt{2}\, x_1 x_2, x_2^2\right)$
  - Possible to separate classes linearly with a hyperplane in 3-D
- Explicit computation of mapped vectors and their inner product $\phi(\bar{x}_i).\phi(\bar{x}_j)$ in higher dimensional feature space becomes computationally expensive



2-D

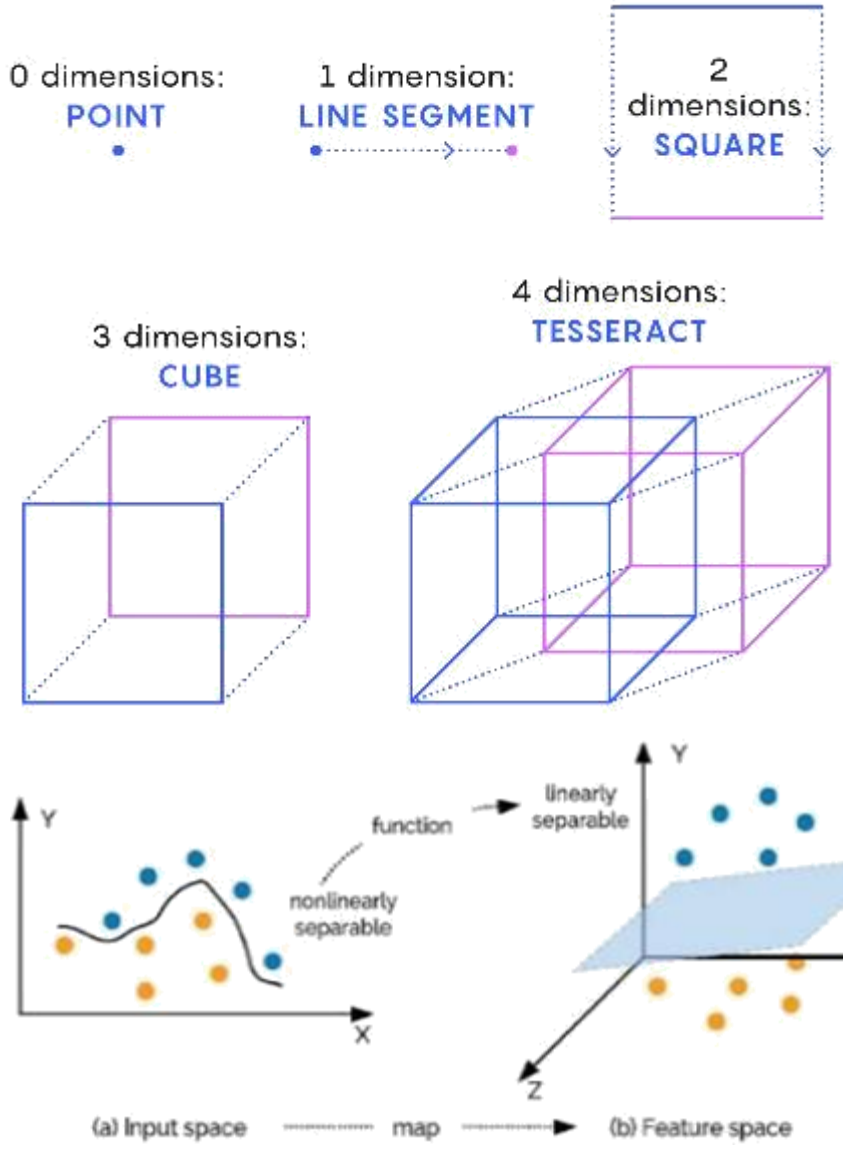
$\phi(\bar{x}) = (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$



3-D

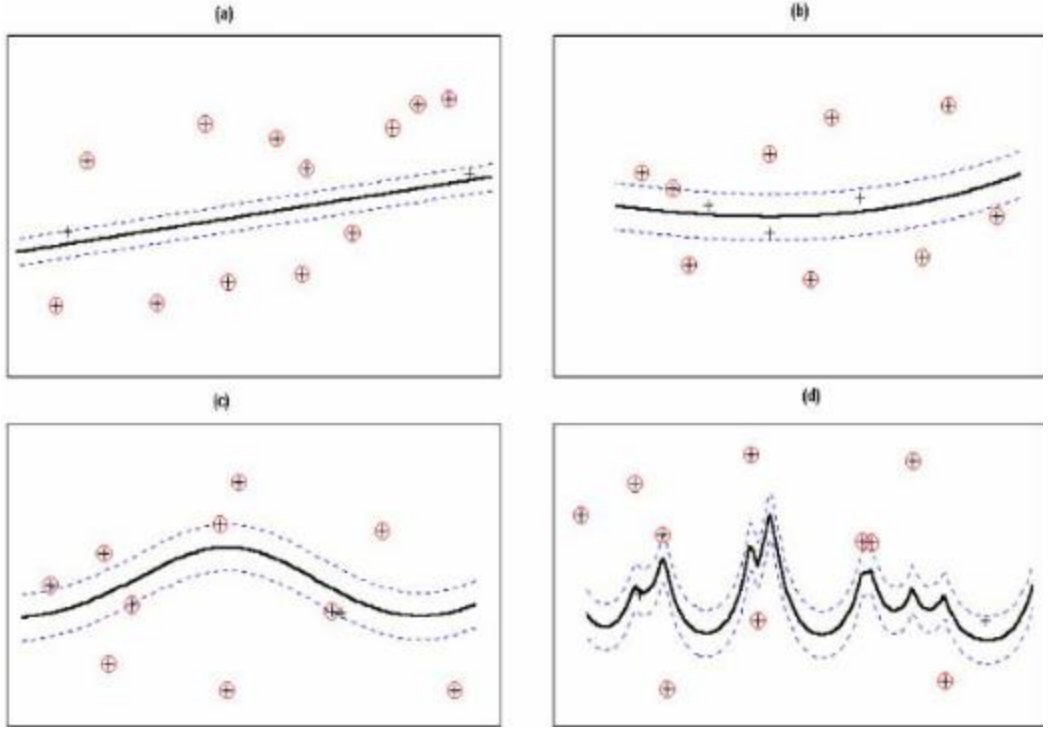
Kernel Tricks/Methods

- Kernel Tricks or Methods owe their name to Kernel Functions
- These kernel functions enable to operate in a high-dimensional, implicit feature space without needing to compute the higher dimensional space coordinates but simply computing the inner products between the images of all pairs of data in the feature space
- For all $\bar{x}_i, \bar{x}_j$ in the input space $\boldsymbol{\chi}$, certain functions $K(\bar{x}_i, \bar{x}_j)$ can be expressed as an inner product in another/higher dimensional space $\boldsymbol{\nu}$
  - $K: \chi \times \chi \rightarrow \mathbb{R}$ is often referred to as a Kernel of Kernel function
- The dot product computation in the higher dimensional space $\boldsymbol{\nu}$ is made much simpler using
  - $\phi: \chi \rightarrow \nu$ satifying $K(\bar{x}_i, \bar{x}_j) = \langle \phi(\bar{x}_i), \phi(\bar{x}_j) \rangle_\nu$
  - An explicit representation for ϕ is NOT necessary as long as ν is an inner product
- **Kernel Function** $\boldsymbol{K(\bar{x}_i, \bar{x}_j)}$ that produces equivalent dot product in transformed higher dimensional space - $\boldsymbol{\phi(\bar{x}_i)^T . \phi(\bar{x}_j)}$
- This trick helps avoid the explicit computation of mapped vectors and their inner product
- Kernel function $K(\bar{x}_i, \bar{x}_j)$ is a generalisation of the dot product in higher feature space $\phi(\bar{x})$



Common Kernel Functions

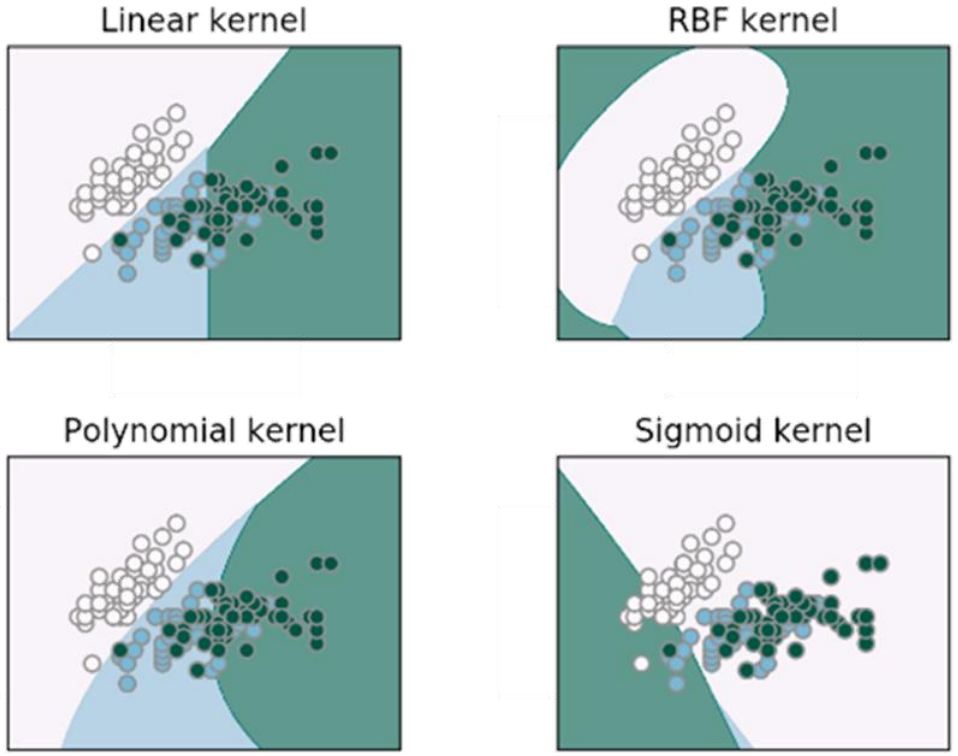
- **Linear Kernel**
  - $K(\bar{x}_i, \bar{x}_j) = \bar{x}_i \,.\, \bar{x}_j$
  - Equivalent of no transformation – works for linearly separable data
- **Polynomial Kernel**
  - $K(\bar{x}_i, \bar{x}_j) = (\bar{x}_i . \bar{x}_j + c)^d$
  - Maps data into polynomial feature space to model polynomial decision boundary in original space
- **Radial Basis Function aka Gaussian Kernel**
  - $K(\bar{x}_i, \bar{x}_j) = e^{\left(-\frac{\|\bar{x}_i - \bar{x}_j\|^2}{2\sigma^2}\right)}$
  - Maps data into an infinite-dimensional feature space enabling highly flexible decision boundary
- A right kernel function is chosen based on data complexity and distribution



(a) Linear, (b) Polynomial, (c) Gaussian RBF and (d) Exponential RBF

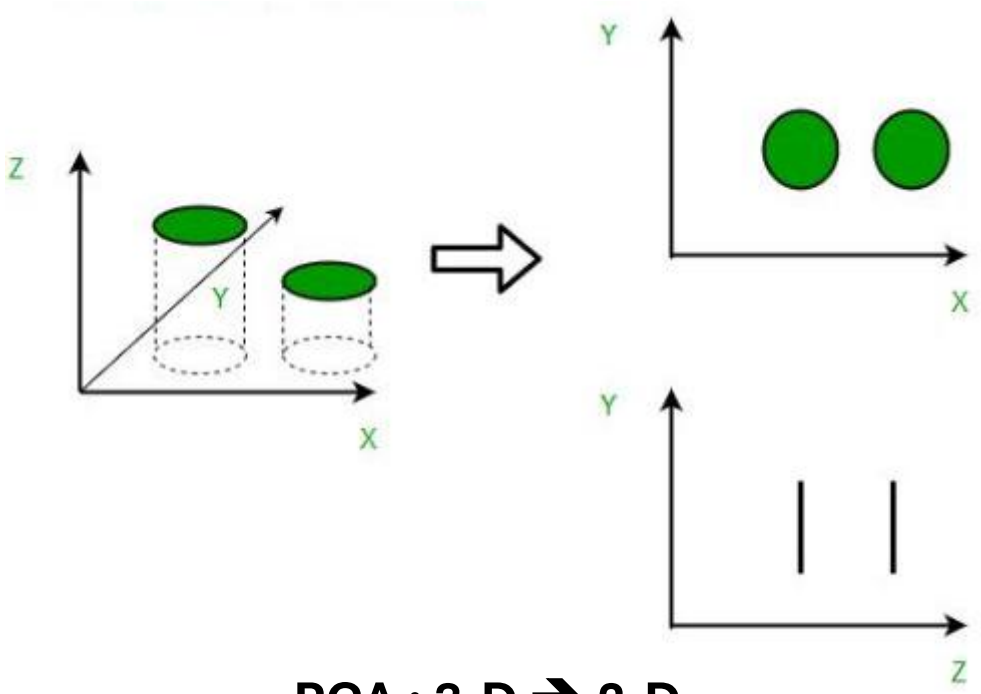
Kernel SVM and Decision Boundaries

- Substituting the **Kernel Function** into the dual problem
  - $\boldsymbol{max}\left[\sum_{i=1}^{2M}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{2M}\lambda_i\lambda_j y_i y_j \bar{x}_i^T \bar{x}_j\right] \Rightarrow \boldsymbol{max}\left[\sum_{i=1}^{2M}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{2M}\lambda_i\lambda_j y_i y_j \boldsymbol{K}(\bar{x}_i, \bar{x}_j)\right]$
  - Subject to: $\boldsymbol{\lambda_i \geq 0}, \forall \boldsymbol{i}$ and $\sum_{i=1}^{2M}\boldsymbol{\lambda_i}$**=0**
- The decision boundary after incorporating the Kernel Function
  - $\boldsymbol{f}(\bar{x}, \boldsymbol{b}) = \boldsymbol{sign}\left(\sum_i \boldsymbol{\lambda_i y_i K}(\bar{x}_i, \bar{x}_j) + \boldsymbol{b}\right)$

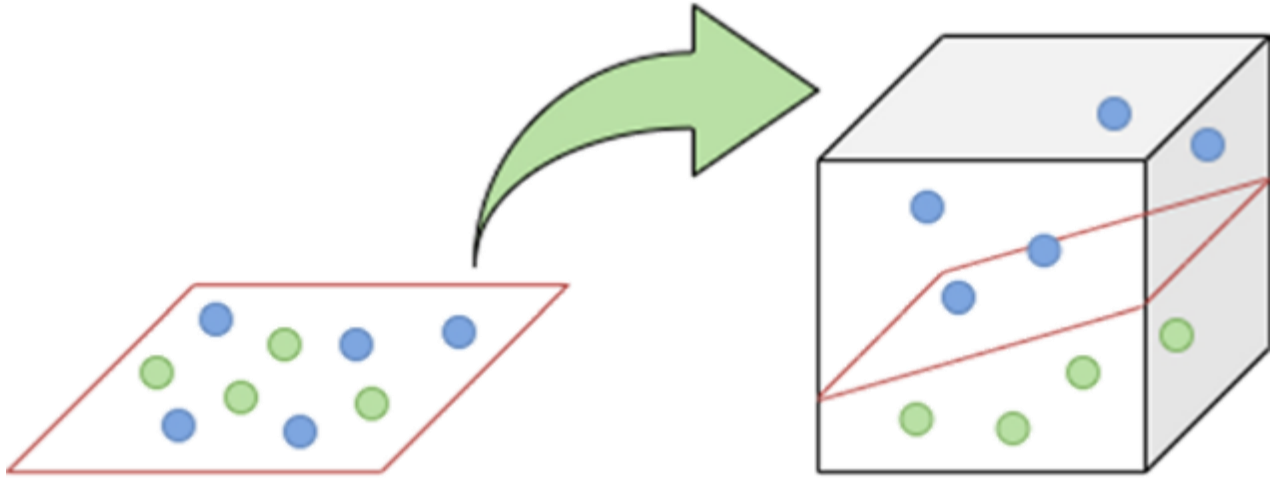


Kernel Tricks Vs PCA

- PCA **reduces dimensionality**, Kernel Trick effectively **increases dimensionality**
- **PCA** performs **linear** operations on the original feature space to find principal/lower dimensional components
- **Kernel Trick** applies a **non-linear transformation** via kernel functions to implicitly increase dimensionality
- **PCA simplifies**, while **Kernel Trick enables richer separation**



PCA : 3-D ➔ 2-D



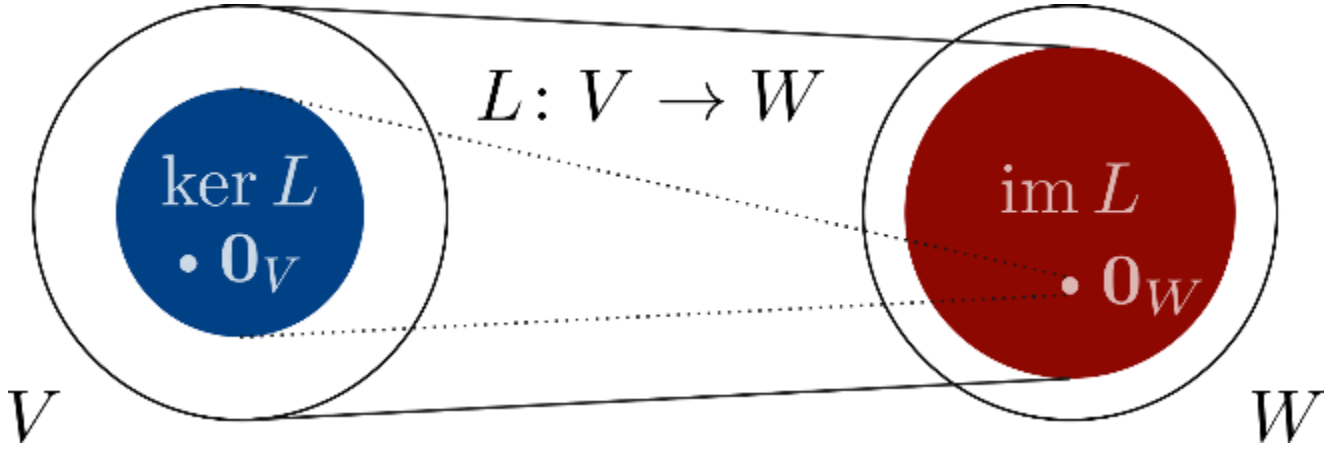
Kernel Trick: 2-D ➔ 3-D

Sidebar Topic – Kernels in Linear Algebra

- Kernel of a linear map/matrix – aka **null space**
- Linear map $L: V \rightarrow W$ - V, W are two vector spaces
- $ker(L) = \{v \in V \mid L(v) = 0\} = L^{-1}(0)$
- L is a linear sub-space of domain V
- Two elements of V have the same image in W **iff** their difference lies in the kernel of L
  - $v_1, v_2 \in V, L(v_1) = L(v_2) \Leftrightarrow (v_1 - v_2) \in V$
  - $\Rightarrow\ L(v_1) - L(v_2) = L(v_1 - v_2) = 0$
- **Rank-nullity theorem** (V is finite dimensional)
  - $dim(ker\ L) + dim(image\ L) = dim(V)$

Linear Map/Transform

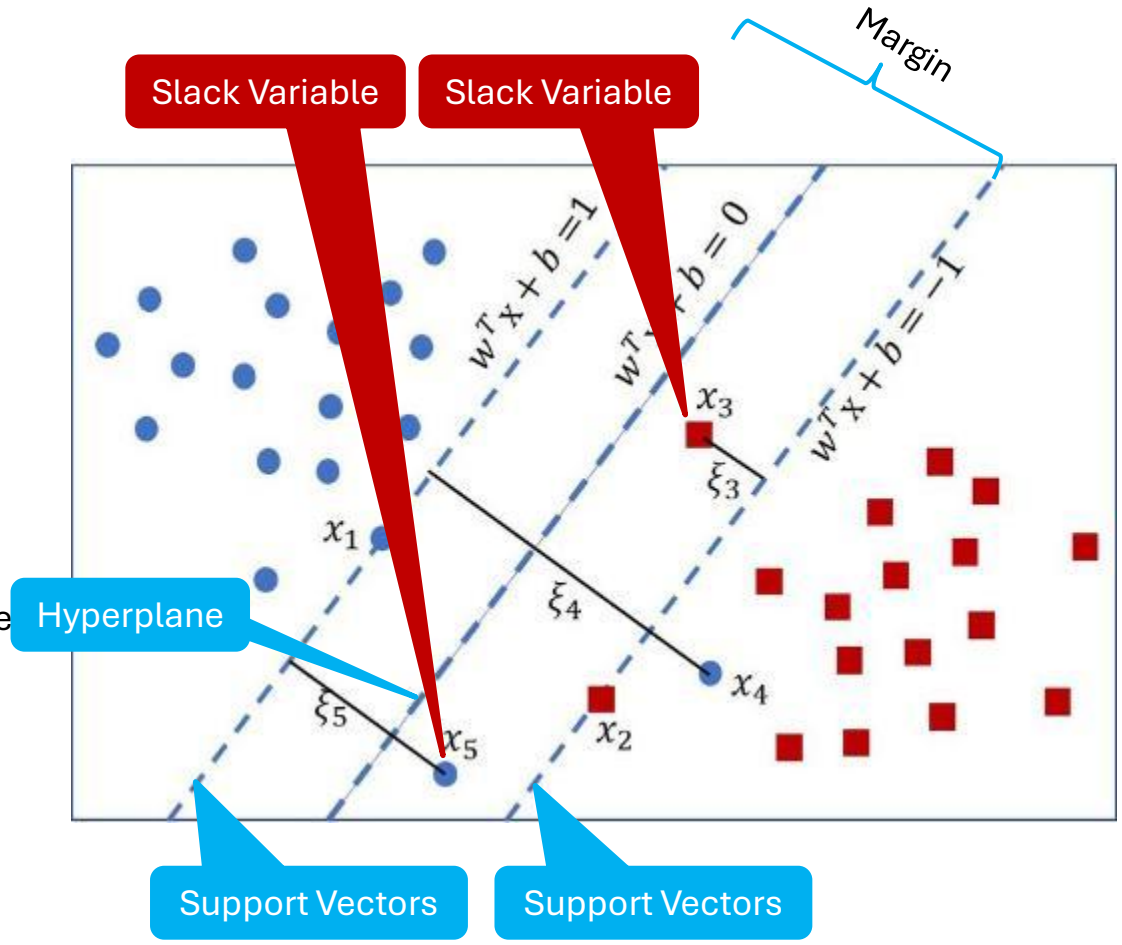
$$\boldsymbol{Lx} \Leftrightarrow \begin{bmatrix} \boldsymbol{l_{11}x_1 + l_{12}x_2 + \cdots l_{1n}x_n} \\ \boldsymbol{l_{21}x_1 + l_{22}x_2 + \cdots l_{2n}x_n} \\ \boldsymbol{\vdots} \\ \boldsymbol{l_{m1}x_1 + l_{m2}x_2 + \cdots l_{mn}x_n} \end{bmatrix}_{m \times n}$$



SVM With Slack Variables

- Primal SVM Classifier aka **Hard Margin SVM** works well when the data is perfectly linearly separable
- Real-world data often contains **Noise** and/or **overlaps** between classes
- One way to handle **noise** and **overlaps** is using **slack variables** (ξ_i)to allow some data points to lie inside the margin or on the wrong side ➔ **Soft Margin SVM**
- Optimisation problem with slack variables
  - $\min_{\bar{a},b,\xi} \left[\frac{1}{2}\|\bar{a}\|^2 + C\sum_{i=1}^{2M}\xi_i\right]$

  Subject to:
  - $y_i(\bar{a}^T\bar{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0\ \forall\ i$
  - $\xi_i \geq 0$ – the point is either inside the margin or mis-classified
  - C – Regularisation parameter to control the trade-off between maximising the margin and minimizing the slack
- Purpose of Slack Variables
  - **Classification flexibility** – works for both **linearly separable** and **non-separable** data
  - **Trade-off Control**: C allows balancing a larger margin and fewer violations
- Key Points
  - Slack Variables ξ_i allow some points to be inside the margin or misclassified
  - Soft Margin SVM balances maximizing margin and minimizing classification errors using C
  - SVM handle noisy, overlapping, or non-linearly separable data with slack variables
  - Soft Margin SVM is more flexible, generalizable model



Accuracy and Efficiency

- Accuracy
  - Accuracy represents the proportion of correctly classified samples (both true positives and true negatives) out of the total number of samples.
  - $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- Precision
  - Precision measures how many of the samples predicted as positive are actually positive. It focuses on the quality of the positive predictions.
  - $Precision = \frac{TP}{TP+FP}$
- Recall aka Sensitivity aka TPR
  - Recall measures how many of the actual positive samples were correctly classified. It focuses on the ability to find all positive instances
  - $Recall = \frac{TP}{TP+FN}$
- F1-Score
  - The F1-score is the harmonic mean of precision and recall. It is useful when the dataset has an uneven class distribution.
  - $F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$
- Confusion Matrix
  - A confusion matrix is a table used to describe the performance of a classification model. It provides insight into the following
    - True Positives (TP): Correctly predicted positive instances
    - True Negatives (TN): Correctly predicted negative instances
    - False Positives (FP) (Type I Error): Incorrectly predicted positive instances
    - False Negatives (FN) (Type II Error): Incorrectly predicted negative instances
- RoC Curve
  - The ROC curve is a graphical representation of a classification model's performance across all classification thresholds. It plots the True Positive Rate (Recall) against the False Positive Rate.
- Feature Importance
  - Relative contribution of each feature to the prediction made by a model. It provides insight into which features have the most influence on the target prediction.

Key Takeaways

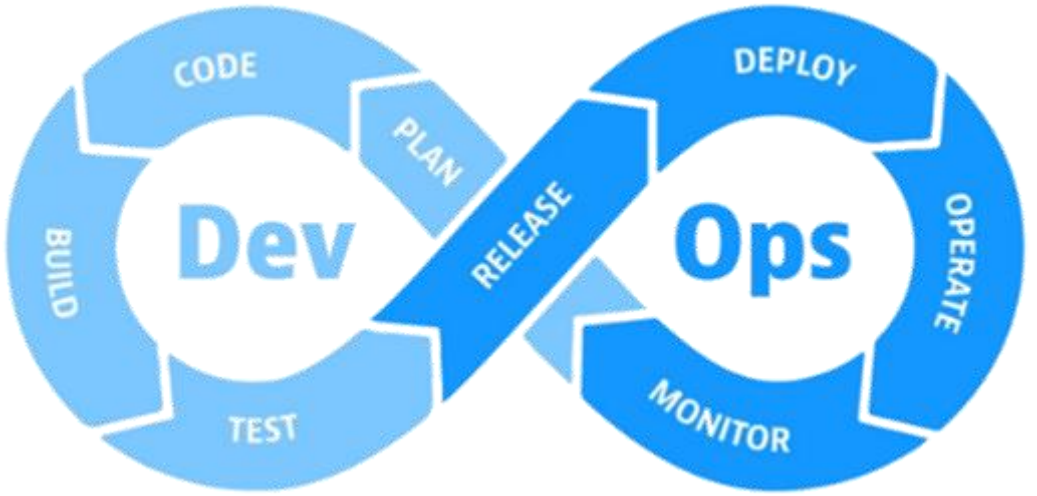
- SVM is powerful for classification
- Works well in high-dimensional spaces and for non-linearly separable data with the kernel trick
- Effective for imbalanced data with hyperparameter tuning

Project Overview

- **Project Title**
  - Diagnosing breast cancer (malignant or benign tumor) based on the diagnostic data
- **Project Objective**
  - This project implements a Support Vector Machine (SVM) classifier to predict the class of breast cancer (breast cancer detected or not detected)) from the provided dataset.
  - The dataset is loaded from sklearn
  - The breast cancer dataset consists of 569 samples, each representing a patient with a set of features.
  - The dataset is used for training a SVM classifier and for evaluating performance metrics such as accuracy, precision, recall, F1-score, confusion matric etc.
- **Dataset Description**
  - Implements a Support Vector Machine (SVM) – linear and RBF – classifiers to predict the class of breast tumor (breast cancer) from the provided dataset – malignant or benign.
  - The dataset needs to be loaded from scikit-learn
  - The dataset should be used for training a SVM classifier and evaluate its performance using various metrics such as accuracy, precision, recall, F1-score
- **Dataset Details**
  - The breast cancer dataset consists of 569 samples, each representing a patient with a set of features
- **Data Source/Published By**
  - Breast Cancer Wisconsin (Diagnostic) - UCI Machine Learning Repository
  - Nuclear feature extraction for breast tumor diagnosis | Semantic Scholar
  - Supported by - CS Department, University of Wisconsin-Madison
- **Features & Data Download Link**
  - Breast Cancer Wisconsin (Diagnostic) - UCI Machine Learning Repository

Breast Cancer Detection Project Implementation Steps

- Define settings and configurations
- Define a function to add AAM-IPL watermark to all plots of this project
- Load the breast cancer data set from scikit-learn
- Print the description/meta data of the breast cancer data set
- Scale the data using standard scaler
- Display column/feature names in the breast cancer dataset
- Count the data samples for each target value
- Print the counts separately
- Print the feature names
- Display the first row of the data set
- Split the data set into train and test, train the model for Linear SVM and SVM with RBF Kernel Trick, and predict the disease presence for the test data
- Plot the graphs for accuracy, confusion matrix, RoC curve, Precision-Recall curve, feature importance and F1 score
- Print the formulae for precision, recall, F1 score
- Plot relative performance of model for the above metrics separately – one graph per metric
- Plot decision boundaries for Linear SVM and SVM with RBF Kernel Trick



Project Implementation Demo

Interested in building a Gen AI application?

Reach out to venkat@brillium.in

AAML-IPL Brought You in Partnership with:

Brillium Technologies

Sector 7, HSR Layout, Bengaluru 560102, Karnataka, India
Website: www.brillium.in | Email: connect@brillium.in