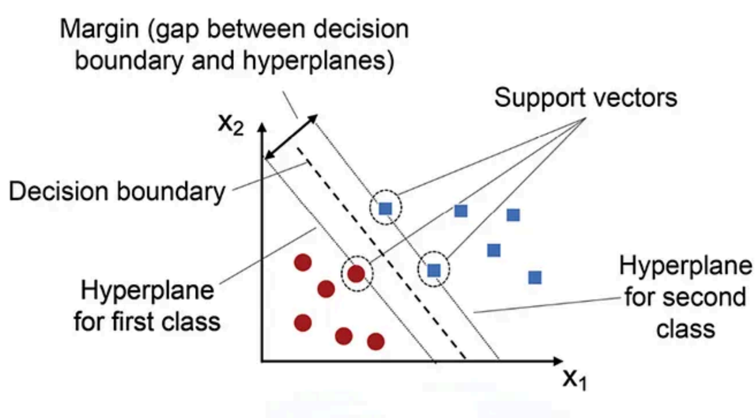
Unit - II Supervised and Unsupervised Learning Algorithms

2.1 Supervised Learning : Support Vector Machines- Working, Types and Implementation of SVM

Support Vector Machines:

- Used for classification and regression tasks.
- Find boundaries as - hyperplane - that separates classes in the data.
- Useful when - binary classification like spam vs. not spam or cat vs. dog.

Working

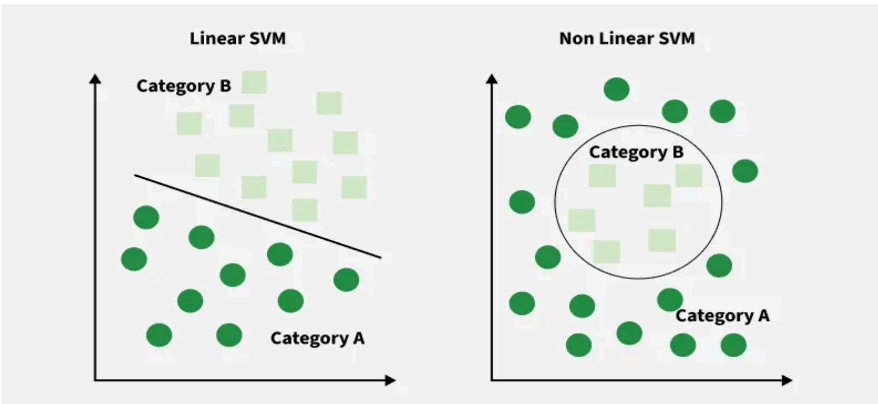


- It looks for the line or plane that gives the maximum margin — the largest distance between the boundary and the nearest data points from each class.
- These closest points are called support vectors (they “support” the boundary).
- When data isn’t linearly separable, SVM uses kernels (like RBF, polynomial) to map data into a higher dimension where it becomes separable.

Types

Linear SVM: Uses a straight line/hyperplane to separate data that is linearly separable by maximizing the margin.

Non-linear SVM: Uses kernel functions to map data into a higher dimension so a linear boundary can separate data that isn't linearly separable



Feature	**Linear SVM**	**Non-linear SVM**
Decision Boundary	Straight line / hyperplane	Curved or complex boundary
Data Type	Linearly separable data	Non-linearly separable data
Use of Kernels	Not required	Required (e.g., RBF, polynomial)
Computation	Faster, simpler	More complex, slower
Feature Space	Original space	Transformed to higher-dimensional space
Example Use Cases	Text classification, large sparse data	Image data, complex patterns

Implementation of SVM

1. **Import libraries** (e.g., sklearn.svm).
2. **Load and preprocess data** (scaling often needed).

   Choose SVM type (Linear or with kernel like RBF).
3. **Train the model** using fit().

4. **Make predictions** using predict().
5. **Evaluate performance** with accuracy or other metrics.

2.2 Unsupervised Learning : K-Mediod Algorithm- working and implementation

- K-Medoids, also known as Partitioning Around Medoids (PAM).
- It is similar to K-Means, but instead of using the mean of points as a cluster center, it uses an actual data point called a medoid.

Working

1. **Initialize medoids** randomly from the dataset.
2. **Assign each data point** to the nearest medoid based on a distance metric (e.g., Manhattan/Euclidean).
3. **Update medoids:**
   For each cluster, choose the point that **minimizes the total distance** to all other points in that cluster → this becomes the new medoid.
4. **Repeat** the assignment and update steps until medoids no longer change or convergence is reached.

Implementation

- Choose the number of clusters (k) and select initial medoids.
- Assign points to the nearest medoid.
- Update medoids to minimize total distance in each cluster.
- Reassign points to new medoids.
- Repeat until the medoids don’t change.
- Output final clusters and medoids.

2.3 Dimensionality Reduction: Introduction, Subset Selection, Principal Component Analysis

- **Dimensionality Reduction** is the process of reducing the number of features (variables) in a dataset while retaining most of the important information.
- Benefits include:
  - Reduces computation time
  - Removes noise and redundant features
  - Helps in visualization and improves model performance

Subset Selection

- Selects a **subset of the original features** that are most relevant.
- Techniques include:
  - **Filter methods:** Use statistical measures (e.g., correlation, chi-square)
  - **Wrapper methods:** Use model performance to evaluate feature subsets
  - **Embedded methods:** Feature selection occurs during model training (e.g., LASSO)

Principal Component Analysis (PCA)

- **PCA** is a feature extraction technique that transforms data into a new set of **uncorrelated variables called principal components**.
- Working:
  1. Standardize the data.
  2. Compute the covariance matrix.

3. Compute eigenvectors and eigenvalues of the covariance matrix.
4. Select top principal components (based on variance explained).
5. Transform the original data into the new reduced feature space.

- PCA reduces dimensionality **without losing much information** and is widely used for visualization and noise reduction.

2.4 Association Rule Learning–Apriori Algorithm, Eclat Algorithm

Association Rule Learning

- **Purpose:** Discover interesting relationships (rules) between variables in large datasets.
- Common in market basket analysis: finding items that are frequently bought together.
- Key concepts:
  - **Support:** Frequency of an itemset in the dataset.
  - **Confidence:** Likelihood that item B is purchased when item A is purchased.
  - **Lift:** Strength of the rule compared to random chance.

Apriori Algorithm

- **Type:** Breadth-first search algorithm.
- **Working:**
  1. Identify all frequent 1-itemsets that meet minimum support.
  2. Generate candidate k-itemsets from frequent (k-1)-itemsets.
  3. Prune candidates that have infrequent subsets.
  4. Repeat until no more frequent itemsets can be generated.

5. Generate association rules from the frequent itemsets that meet minimum confidence.

- **Use:** Simple and widely used but can be slow for large datasets.

Eclat Algorithm

- **Type:** Depth-first search algorithm.
- **Working:**
    1. Uses **vertical data format** (item → list of transaction IDs containing the item).
    2. Intersect transaction ID lists to compute support for itemsets.
    3. Recursively generate frequent itemsets using intersections.
- **Use:** Faster than Apriori for dense datasets because it avoids generating many candidate sets.

2.5 Generative Models - Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs)

Generative Models

- **Purpose:** Learn the underlying data distribution to **generate new, similar data**.
- Applications: Image synthesis, text generation, data augmentation.
- Unlike discriminative models (which classify data), generative models **model how the data is generated**.

Generative Adversarial Networks (GANs)

- **Components:**
    - **Generator:** Creates fake data from random noise.
    - **Discriminator:** Distinguishes between real and generated data.

- **Working:**
    - Generator and discriminator compete in a **minimax game**.
    - Generator improves to produce realistic data; discriminator improves to detect fakes.
    - Training continues until generated data is indistinguishable from real data.
- **Use:** Image generation, video synthesis, style transfer.

Variational Autoencoders (VAEs)

- **Components:**
    - **Encoder:** Maps input data to a probability distribution in latent space.
    - **Decoder:** Generates data from sampled points in latent space.
- **Working:**
    - Input is compressed into latent variables (mean and variance).
    - Decoder reconstructs input from latent space.
    - Optimized using **reconstruction loss** to enforce distribution.
- **Use:** Image generation, anomaly detection, data compression.