Unit 5

Fundamentals of Machine Learning

5.1 Regression

Regression is a statistical technique used to model and analyze the relationship between one dependent variable (Y) and one or more independent variables $X_1, X_2, \dots, X_n$). Its primary goals are:

1. Prediction: Estimating the value of the dependent variable based on the independent variables.
2. Understanding Relationships: Determining how and to what extent the independent variables affect the dependent variable.

Types of Regression:

- Simple Linear Regression: Involves one independent variable and a linear relationship.
- Multiple Regression: Involves multiple independent variables.
- Non-linear Regression: Models relationships that cannot be represented as straight lines.
- Logistic Regression: Used for binary or categorical outcomes.

Simple Linear Regression is a statistical method used to model the linear relationship between a dependent variable (Y) and a single independent variable (X). It assumes that the relationship can be expressed as a straight line:

$$Y = aX + b$$

Key Components:

1. **Dependent Variable (Y)**: The outcome or response variable being predicted or explained.
2. **Independent Variable (X)**: The predictor or explanatory variable.
3. **Intercept(b)**: The value of Y when $X = 0$. It represents the intercept on Y axis.
4. **Slope** (a): The change in Y for a one-unit change in X. It measures the strength and direction of the relationship.

Least Squares Method for Regression

The **Least Squares Method** is a mathematical approach to determine the best-fitting line (or curve) by minimizing the sum of the squared differences (residuals) between the observed values Y and the predicted values $\hat{Y} = a + bX$.

Residuals:
The difference between the observed value Y and the predicted value $\hat{Y} = a + bX$

$$\text{Residual } (R) = Y - \hat{Y}$$

Objective: To minimize the sum of square of residuals, that is, minimize

$$E = \sum R^2$$

Let the observed values are $(x_1, y_1),\ (x_2, y_2), \ldots., (x_n, y_n)$, then

$$E = \sum_{i=1}^{n} (y_i - a - bx_i)^2$$

To minimize E, we use principal of maxima and minima and find the values a and b.

The normal equations are

$$\sum y = na + b \sum x$$

$$\sum xy = a \sum x + b \sum x^2$$

We solve the above two equations to find the values of a and b, and putting these values in $y = a + bx$, we get the straight line of best fit.

Using least square method, we find

$$b = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Since line is passing through $(\bar{x}, \bar{y})$, to find a, we can write

$$\bar{y} = a + b\bar{x}$$

Example 1. Predict a student's final exam score based on the number of hours they studied. Fit a straight line to the following data.

x	y
2	65
4	70
6	75
8	85
10	95

Solution:

x	y	$x_i - \overline{x}$	$y_i - \overline{y}$	$(x_i - \overline{x})(y_i - \overline{y})$	$(x_i - \overline{x})^2$
2	65	-4	-13	52	16
4	70	-2	-8	16	4
6	75	0	-3	0	0
8	85	2	7	14	4
10	95	4	17	68	16
30	**390**	**0**	**0**	**150**	**40**

$$b = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2} = \frac{150}{40} = 3.75$$

$$\bar{y} = a + b\bar{x}$$

$$\bar{x} = \frac{30}{5} = 6, \bar{y} = \frac{390}{5} = 78$$

$$a = 78 - 3.75 * 6 = 55.5$$

Thus, the equation of best fit is: $y = 3.75x + 55.5$.

5.2 Bayesian Classification

Bayesian Classification is a probabilistic approach to classification based on Bayes' Theorem. In this approach, we use prior knowledge (prior probabilities) about classes along with observed data to estimate the probability that a given data point belongs to each class. The classifier assigns a class label to each data point based on which class has the highest posterior probability.

Bayes' Theorem

Bayes' theorem provides the probability of an event given another event which has already occurred. Let A and B are two events, then

$$P(A/B) = \frac{P(A)P(B/A)}{P(B)}, \quad P(B) \neq 0 \qquad (1)$$

where,

$P(A/B)$: Posterior probability, probability of A given B

$P(B/A)$: Likelihood, the probability of B given A

$P(A)$: Prior Probability

$P(B)$: Marginal Probability: The overall probability of B across all possibilities of A

Naïve Bayes' Classification

Naïve Bayes' classification algorithm is based on Bayes' theorem. Naïve Bayes' classifier works on the assumptions that the

- **Feature independence:** The features of the data are conditionally independent of each other, given the class label. In other words, given the class label, the presence or absence of one feature does not affect the presence or absence of another. This is called the "naive" assumption and simplifies the computation, but it may not hold for all real-world data.
- **Equal importance of all features:** Each feature is assumed to contribute independently to the probability of a particular class. This means that all features are considered relevant, though some might have a stronger influence on classification outcomes than others. This relevance is weighted by each feature's probability in the context of the class.

Let us consider a dataset with a set of attributes $X = \{X_1, X_2, \dots, X_n\}$ and each of the element in the dataset with a set attribute values corresponds to one of the k classes from the set $C = \{C_1, C_2, \dots, C_k\}$ of classes. Here, the problem is given a set of attribute values $X = \{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n\}$ of an item X, classify X in one of the classes, that is, $P(C/X)$.

On applying Bayes' theorem, we get

$$P(C_i/X) = \frac{P(C_i)P(X/C_i)}{P(X)} \quad (1 \le i \le k) \tag{2}$$

We can calculate the probability $P(C_i/X)$ for all classes. The denominator $P(X)$ will remain constant for each class, thus for classification purpose, we need to calculate $P(C_i)P(X/C_i)$ part only.

Now, $$P(X/C_i) = P(\{X_1 = x_1, X_2 = x_2, \dots, X_n = x_n\}/C_i) \tag{3}$$

Using the naïve bayes assumption that each attribute is independent, we have

$$P(X/C_i) = P(X_1 = x_1/C_i).P(\,X_2 = x_2/C_i).\dots P(\,X_n = x_n/C_i) \tag{4}$$

The prior and likelihood probabilities $P(C_i)$ and $P\left(x_j/C_i\right)$ $(1 \leq j \leq n, 1 \leq i \leq k)$ can be calculated with the help of data. The class for which the probability $P(C_i/X)$ is highest will be assigned to the given set of attribute values.

Example 2: The following data describes the attributes of leaves of certain plant with their class. Using the following data and Naïve Bayes classification, classify the item with colour = Green, length = Average and Width = Wide

Color	**Length**	**Width**	**Class**
Green	Long	Wide	Healthy
Yellow	Short	Wide	Healthy
Brown	Long	Wide	Healthy
Brown	Short	Medium	Non-Healthy
Green	Average	Short	Healthy
Yellow	Long	Short	Healthy
Yellow	Short	Short	Non-Healthy
Green	Long	Wide	Non-Healthy
Brown	Average	Short	Non-Healthy
Green	Average	Medium	Healthy

Solution: Here we have three attributes $X = \{X_1 =$ 'Colour', $X_2 =$ 'Length' and $X_3 =$ ' Width'} with two classes, $C_1 =$ 'Healthy' (H), and $C_2 =$ 'Non-Healthy' (NH). We have to find the probability:

$$P(H/X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'})$$

and

$$P(NH/X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'})$$

Using eq(2) and ignoring the denominator,

$$P(C_1/X) = P(C_i)P(X/C_i)$$

$$\Rightarrow P(H/X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'}) =$$

$$P(H).P(X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'}/H)$$

Using equation (4)

$$P(H/X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'})$$

$$= P(H).P(X_1 = \text{'Green'}/H).P(X_2 = \text{'Average'}/H).\ P(X_3 = \text{' Wide'}/H) \quad (5)$$

Similarly,

$$P(NH\ /\ X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'})$$

$$= P(NH).P(X_1 = \text{'Green'}/NH).P(X_2 = \text{'Average'}/NH).\ P(X_3 = \text{'Wide'}/NH) \qquad (6)$$

From the given data, we observe that there are 10 items out of which 6 are healthy and 4 are non healthy. Thus,

$$P(H) = \frac{6}{10} = \frac{3}{5} \text{ and } P(NH) = \frac{4}{10} = \frac{2}{5}.$$

We will create bi-variate distributions for calculation of conditional probabilities:

Bi-variate Distribution for colour and class:

Colour	**Class**	
	Healthy	**Non-Healthy**
Green	3	1
Yellow	2	1
Brown	1	2
Total	**6**	**4**

Thus, $P(X_1 = \text{'Green'}/H) = \frac{3}{6} = \frac{1}{2}$ and $P(X_1 = \text{'Green'}/NH) = \frac{1}{4}$

Bi-variate Distribution for Length and class:

Length	**Class**	
	Healthy	**Non-Healthy**
Long	3	1
Average	2	1
Short	1	2
Total	**6**	**4**

Thus, $P(X_2 = \text{'Average'}/H) = \frac{2}{6} = \frac{1}{3}$ and $P(X_2 = \text{'Average'}/\text{N}H) = \frac{1}{4}$

Bi-variate Distribution for Width and class:

Width	**Class**	
	Healthy	**Non-Healthy**
Wide	3	1
Medium	1	1
Short	2	2
Total	**6**	**4**

Thus, $P(X_3 = \text{'Wide'}/H) = \frac{3}{6} = \frac{1}{2}$ and $P(X_3 = \text{'Wide'}/NH) = \frac{1}{4}$

From eq(4) and eq(5), we get

$$P(H).P(H/X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{'Wide'}) = \frac{3}{5}.\frac{1}{2}.\frac{1}{3}.\frac{1}{2} = \frac{1}{20}$$

$$P(NH).P(NH/X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'}) = \frac{2}{5}.\frac{1}{4}.\frac{1}{4}.\frac{1}{4} = \frac{1}{160}$$

Since $P(H/X_1 = \text{'Green'}, X_2 = \text{'Average' and } X_3 = \text{' Wide'})$ is maximum. Hence the predicted class of the item is : 'Healthy'

5.3 K-Nearest Neighbors (KNN) algorithm

The K-Nearest Neighbors (KNN) algorithm is a supervised machine learning algorithm used for classification.

KNN Works (Step-by-Step)

1. **Choose K**:
   - Decide how many neighbors (**K**) to consider (e.g., K = 3 or K = 5).
2. **Calculate Distance**:
   - For a new input, compute the **distance** between this point and all points in the training data.
   - Common distance metrics:
     - Euclidean Distance
     - Manhattan Distance
     - Minkowski Distance
3. **Find Nearest Neighbors**:
   - Identify the **K closest data points** from the training set.
4. **Vote for Class (Classification)**:
   - For classification, assign the class that is most frequent among the K neighbors.

Example 3 Classify the point (3,4) using KNN algorithm and the following data:

x1	x2	Class
2	4	C1
3	5	C1
4	4	C2
6	5	C2
6	7	C2

Solution: K = 3, we calculate distance of (x2 = 3, y2 = 4) from each of the point (x1, x2) using Euclidean distance $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

Sr. No	x1	x2	Class	D
1.	2	4	C1	1.00
2.	3	5	C1	1.00
3.	4	4	C2	1.00
4.	6	5	C2	3.16
5.	6	7	C2	4.24

Since K = 3, the three nearest neighbors are 1, 2 and 3 data items.

5.4 Decision tree-based algorithm

A **decision tree-based algorithm** is a type of supervised machine learning technique used for **classification** and **regression** tasks. It works by learning simple decision rules from data features and organizing them in a tree-like structure.

A **decision tree** has the following aspects

- **Each internal node** represents a decision based on a feature.
- **Each branch** represents the outcome of that decision (Yes or No).
- **Each leaf node** represents a final prediction or classification.

Key points:

1. **Root Node**: The starting point of the tree; it splits the data based on the most important feature.
2. **Splitting**: Dividing data into subsets based on a condition.
3. **Leaf/Terminal Node**: The final output (class or value).
4. **Information Gain** or **Gini Impurity**: Metrics used to decide the best feature to split on.
5. **Pruning**: Cutting down branches that have little importance to avoid overfitting.

Working of Decision tree Algorithms

1. **Choose the Best Feature to Split**: Based on metrics like Gini Impurity, Entropy, or Information Gain.
2. **Split the Data**: According to the chosen feature's values.
3. **Repeat the Process**: On each branch recursively until a stopping condition is met (like max depth or minimum samples).
4. **Make Predictions**: When a new sample is input, it follows the tree's path to a leaf node, and the prediction is made based on that leaf.

ID3 (Iterative Dichotomiser 3)

The ID3 (Iterative Dichotomiser 3) algorithm is a classic decision tree algorithm used in supervised learning for classification tasks. The ID3 algorithm works by building a decision tree, which is a hierarchical structure that classifies data points into different categories and splits the dataset into smaller subsets based on the values of the features in the dataset. The ID3 algorithm then selects the feature that provides the most information about the target variable. The decision tree is built top-down, starting with the root node, which represents the entire dataset. At each node, the ID3 algorithm selects the attribute that provides the most information gain about the target variable. The attribute with the highest information gain is the one that best separates the data points into different categories.

Key Concepts

Entropy (H): Entropy measures the uncertainty in a dataset D, it is defined as

$$H(D) = -\sum_{i=1}^{n} p_i \, log_2 \, p_i$$

where, n is the number of classes in a dataset and p_i is the proportion or probability of i^{th} class.

Information gain: Information gain assesses how much valuable information an attribute can provide. We select the attribute with the highest information gain, which signifies its potential to contribute the most to understanding the data. If information gain is high, it implies that the attribute offers a significant insight. ID3 acts like an investigator, making choices that maximize the information gain in each step. This approach aims to minimize uncertainty and make well-informed decisions, which can be further enhanced by preprocessing the data.

$$IG(A) = H(D) - \sum_{v \in values \ in \ A} \frac{|D_v|}{|D|} H(D_v)$$

where, A is an attribute and D_v is the subset of D, where attribute value is $A = v$.

The attribute with the **highest information gain** is chosen for the split.

Example 4. Apply ID 3 to make a decision tree for the following dataset.

Day	**Weather**	**Temperature**	**Play**
1	Sunny	Hot	No
2	Sunny	Hot	No
3	Cloudy	Hot	Yes
4	Rainy	Mild	Yes
5	Rainy	Cool	Yes
6	Rainy	Cool	No
7	Cloudy	Cool	Yes
8	Sunny	Mild	No
9	Sunny	Cool	Yes
10	Rainy	Mild	Yes

Solution:

Steps:

1. Calculate entropy for "Play"
2. Compute information gain for "Wheather" and "Humidity".
3. Choose the one with the highest gain to split
4. Repeat on subsets

Step 1. Calculation of overall Entropy for the dataset.

Here, two classes are there. Yes = 6, No = 4

$p_1 = \frac{6}{10} = \frac{3}{5}$, and $p_2 = \frac{4}{10} = \frac{2}{5}$.

$$H(D) = -\left[\frac{3}{5}\ log_2\frac{3}{5} + \frac{2}{5}\ log_2\frac{2}{5}\right]$$

$$= -[0.6 \times (-0.737) + 0.4 \times (-1.322)]$$

$$= 0.442 + 0.529$$

$$= 0.971$$

Step 2: Calculate Information Gain for "Weather"

There are three unique values: Sunny, Cloudy, and Rainy

For ‘Sunny’

Weather	**Play**
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes

Yes = 1, No = 3

$p_1 = \frac{1}{4}$, and $p_2 = \frac{3}{4}$.

$$H(D_{sunny}) = -\left[\frac{1}{4}\ log_2\frac{1}{4} + \frac{3}{4}\ log_2\frac{3}{4}\right]$$

$$= -[0.25 \times (-2) + 0.75 \times (-0.415)]$$

$$= 0.811$$

For ‘Cloudy’

Weather	**Play**
Cloudy	Yes
Cloudy	Yes

Yes = 2, No = 0

$p_1 = 1$, and $p_2 = 0$

$$H\left(D_{cloudy}\right) = -[1 *\ log_2 1 + 0]$$

$$= 0$$

For ‘Rainy’

Weather	**Play**
Rainy	Yes

Weather	**Play**
Rainy	Yes
Rainy	No
Rainy	Yes

Yes = 3, No = 1

$p_1 = \frac{1}{4}$, and $p_2 = \frac{3}{4}$.

$$H(D_{rainy}) = -\left[\frac{3}{4}\, log_2 \frac{3}{4} + \frac{1}{4}\, log_2 \frac{1}{4}\right]$$

$$= 0.811$$

Weighted Avg. Entropy:

$$H_{Weather} = \frac{4}{10}(0.811) + \frac{2}{10}(0) + \frac{4}{10}(0.811) = 0.6488$$

Information Gain for weather:

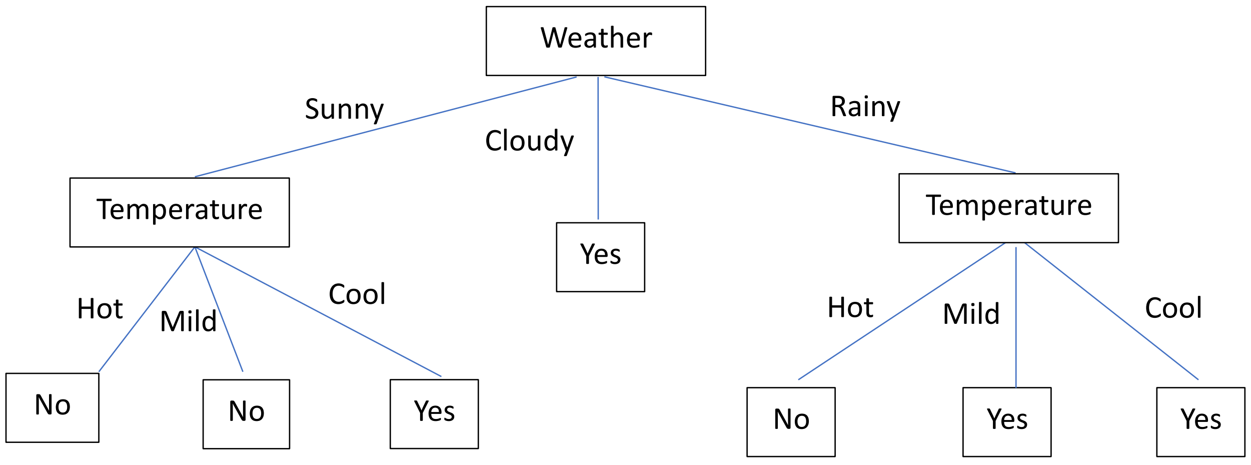
IG(Weather) = 0.971 − 0.6488 = 0.322

Similarly, Information Gain for "Temperature"

IG (Temperature) = 0.090

Since information gain is maximum for 'Weather'. Thus, first splitting node is 'Weather'. Since only two attributes are there, the second node is 'Temperature'.

The decision tree is



5.5 Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for binary classification problems. It predicts the probability that a given input point belongs to a particular class.

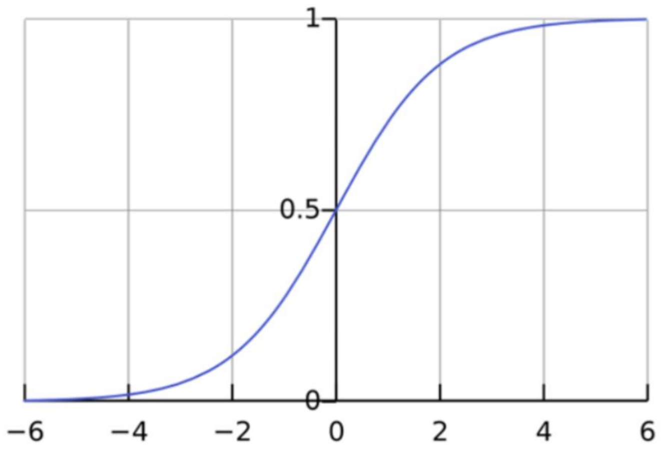
Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.

Mathematical Foundation

Logistic regression uses **sigmoid function** which maps predicted values to probabilities between 0 and 1. The sigmoid function is defined as:

$$f(x) = \frac{1}{1+e^{-x}}$$

It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the **Sigmoid function** or the logistic function. In logistic regression, we use the concept of the threshold value (generally 0.5), which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0. The graph of the function is as follows:



Let there are n $(X_1, X_2, \dots, X_n)$ independent variables with one dependent variable Y. First, we use multiple regression on these variables, we get

$$Z = a_0 + a_1X_1 + a_2\,X_2 + \cdots + a_nX_n$$

Then we use sigmoid function to predict the probabilities, that is,

$$f(Z) = \frac{1}{1 + e^{-Z}}$$

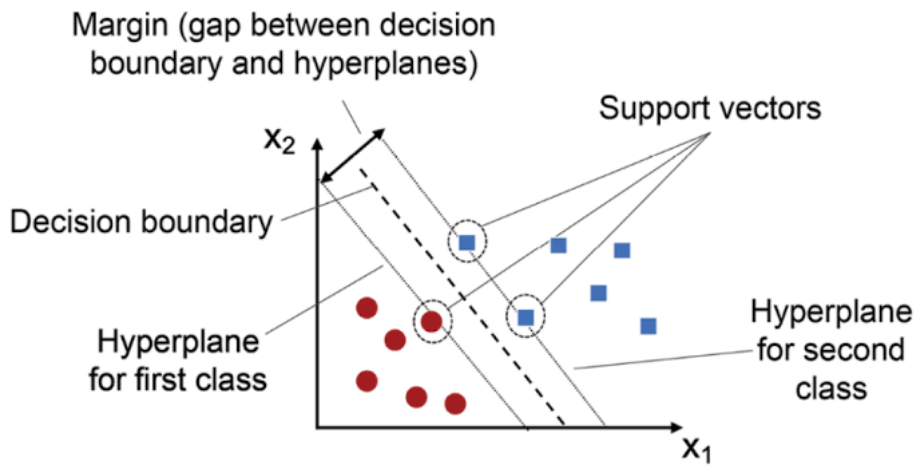
The predicted probability can be converted into a binary outcome using a threshold (e.g., if $P \geq 0.5$, predict 1; otherwise, predict 0).

For example, we have two classes Class 0 and Class 1 if the value of the logistic function for an input is greater than 0.5 (threshold value) then it belongs to Class 1 otherwise it belongs to Class 0. It's referred to as regression because it is the extension of linear regression but is mainly used for classification problems.

5.6 Support Vector Machine (SVM)

A **Support Vector Machine (SVM)** is a supervised machine learning algorithm used primarily for **classification**, but also for **regression** tasks. The core idea of SVM is to find the **optimal hyperplane** that best separates the data points of different classes in a high-dimensional space.

Key Concepts:

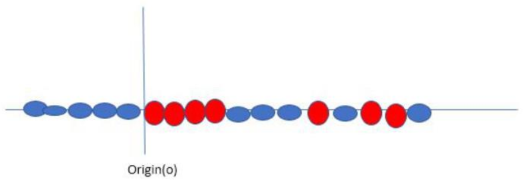

Margin (gap between decision boundary and hyperplanes)
Support vectors
X_2
Decision boundary
Hyperplane for first class
Hyperplane for second class
X_1

1. **Hyperplane**: A decision boundary that separates data points into classes. In 2D, it's a line; in 3D, a plane; and in higher dimensions, a hyperplane.
2. **Support Vectors**: The data points that are closest to the hyperplane. These are the most critical elements of the dataset because they directly affect the position and orientation of the hyperplane.

3. **Margin**: The distance between the hyperplane and the nearest support vectors. SVM aims to maximize this margin to achieve better generalization.

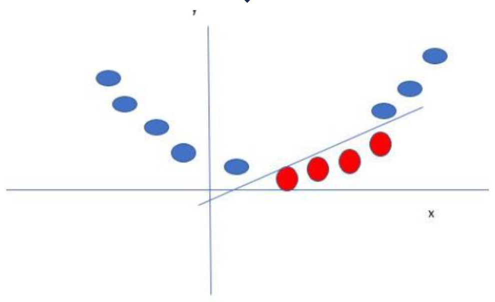
4. **Kernel**: A **kernel** is a function that maps data points into a higher-dimensional space without explicitly computing the coordinates in that space. This allows SVM to work efficiently with non-linear data by implicitly performing the mapping. For example, consider data points that are not linearly separable. By applying a kernel function, SVM transforms the data points into a higher-dimensional space where they become linearly separable.

   a) **Linear Kernel**: For linear separability.
   b) **Polynomial Kernel**: Maps data into a polynomial space.
   c) **Radial Basis Function (RBF) Kernel**: Transforms data into a space based on distances between data points.



One-dimensional data

After applying Kernel



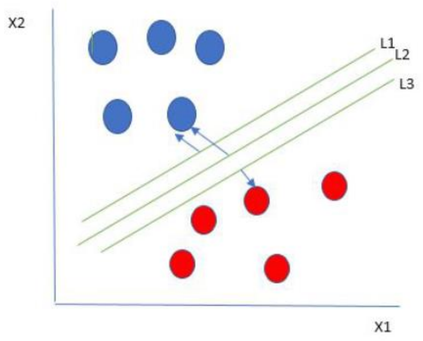
Two-dimensional transformed data

5. **Hard Margin**: A maximum-margin hyperplane that perfectly separates the data without misclassifications.

6. **Soft Margin**: Allows some misclassifications by introducing slack variables, balancing margin maximization and misclassification penalties when data is not perfectly separable.

7. **C**: A regularization term balancing margin maximization and misclassification penalties. A higher C value enforces a stricter penalty for misclassifications.
8. **Hinge Loss**: A loss function penalizing misclassified points or margin violations, combined with regularization in SVM.

The key idea behind the SVM algorithm is to find the hyperplane that best separates two classes by maximizing the margin between them. This margin is the distance from the hyperplane to the nearest data points (**support vectors**) on each side.



Multiple hyperplanes separating the data from two classes

The best hyperplane, also known as the **"hard margin,"** is the one that maximizes the distance between the hyperplane and the nearest data points from both classes. This ensures a clear separation between the classes. So, from the above figure, we choose L2 as hard margin.

Mathematical Computation: SVM

Consider a binary classification problem with two classes, labeled as +1 and -1. We have a training dataset consisting of input feature vectors X and their corresponding class labels Y.

The equation for the linear hyperplane can be written as:

$$w\,x + b = 0$$

where:

w is the normal vector to the hyperplane (the direction perpendicular to it).

b is the offset or bias term, representing the distance of the hyperplane from the origin along the normal vector w.

Distance from a Data Point to the Hyperplane

The distance between a data point x and the decision boundary can be calculated as:

$$d = \frac{w\,x + b}{\|w\|}$$

where $\|w\|$ represents the Euclidean norm of the vector w.

Linear SVM Classifier

Distance from a Data Point to the Hyperplane:

$$\hat{y} = \begin{cases} 1 & if : wx + b \geq 0 \\ 0 & if : wx + b < 0 \end{cases}$$

where, $\hat{y}$ is the predicted label of a data point.

Optimization Problem for SVM

$$z = minimize\ \frac{1}{2}\|w\|^2$$

Subject to the constraint

$$y_i(wx_i + b) \geq 1 \text{ for } i = 1,2,\dots,m$$

where, y_i is the class level,

x_i is the feature vector

m is the total number of training ith training instance.

5.7 CN2 Algorithm (Covering Algorithm for Classification)

The CN2 algorithm is a classic rule-based machine learning algorithm used for classification. It was introduced by Peter Clark and Timothy Niblett in 1989. CN2 is especially effective when you want to extract comprehensible rules from your data.

The goal is to learn a set of if-then rules that can classify new data based on existing examples.

Basic Working Principle:

1. Input: A labeled dataset (i.e., examples with known class labels).
2. Rule Induction:
   - CN2 generates candidate rules using a beam search strategy.
   - It evaluates the quality of each rule using a heuristic (like entropy or accuracy).

3. Rule Selection:
   - The best rule (that covers many correct examples and few wrong ones) is selected.
4. Covering Loop:
   - Remove examples covered by the rule from the dataset.
   - Repeat the process on the remaining examples until all are covered or no good rules are left.
5. Output: A set of if-then classification rules.

Example 5. Using the example of decision tree algorithm, the rules may be

Rule 1:

IF Weather = Sunny AND Temperature = Hot THEN Play = No

Rule 2:

IF Weather = Cloudy THEN Play = Yes

In decision tree, branches are exclusive, however in CN2 rules may be overlapping.

CN2 uses an entropy-based measure or Laplace accuracy to evaluate rule quality, such as:

Laplace accuracy $=\frac{p+1}{p+n+c}$

where:

- p: positive examples covered by the rule
- n: negative examples
- c: number of classes

5.8 Artificial Neural Networks

An **Artificial Neural Network (ANN)** is a **computational model** inspired by the structure and functioning of the human brain. It is used in **machine learning and artificial intelligence** to recognize patterns, model complex relationships, and solve problems such as classification, prediction, and decision-making.

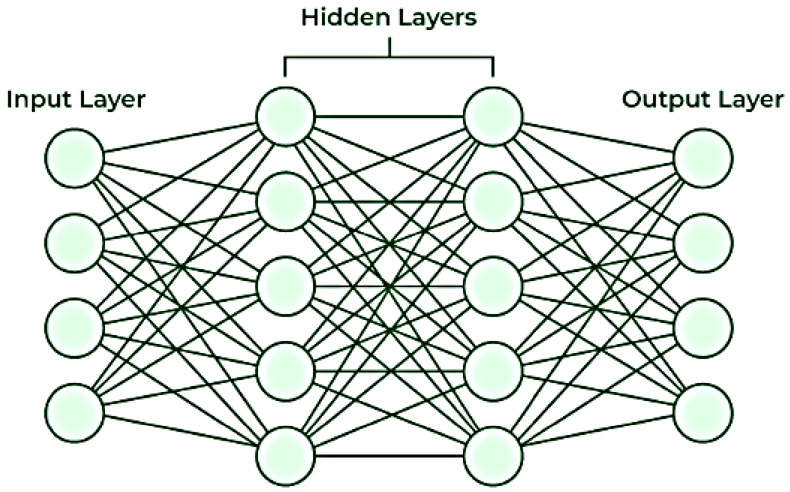
Artificial Neural Networks contain artificial neurons, which are called units. These units are arranged in a series of layers that together constitute the whole Artificial Neural Network in a system. A layer can have only a dozen units or millions of units, as this depends on how complex neural networks will be required to learn the hidden patterns in the dataset.

Basic Structure

An ANN consists of **three main layers**:

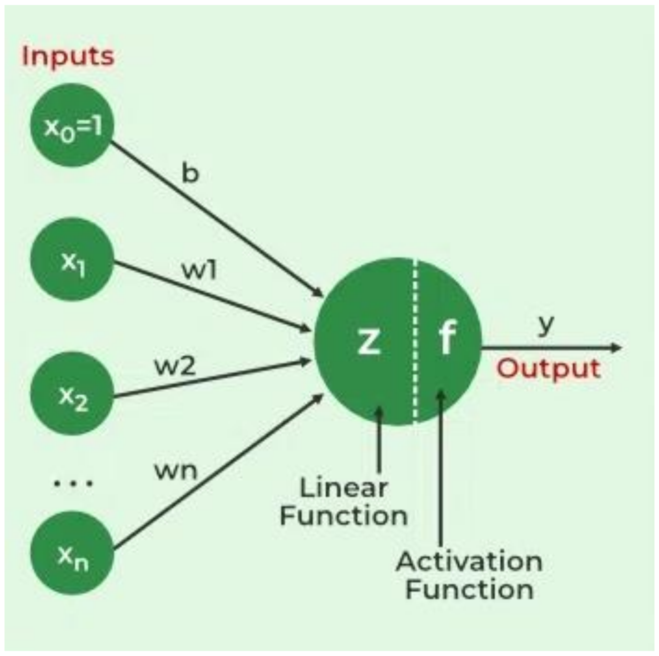
1. **Input Layer**:
   - The input layer receives data from the outside world, which the neural network needs to analyze or learn about.
   - Each neuron in this layer represents a feature of the input.
2. **Hidden Layer(s)**:
   - Processes input through weighted connections and activation functions.
   - There can be one or more hidden layers (deep learning uses many).
3. **Output Layer**:
   - Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to the input data provided. It produces the final prediction or decision (e.g., Yes/No, classification result).

In the majority of neural networks, units are interconnected from one layer to another. Each of these connections has weights that determine the influence of one unit on another unit. As the data transfers from one unit to another, the neural network learns more and more about the data, which eventually results in an output from the output layer.



Key Components

- Neuron (Node): Basic unit that receives inputs, processes them, and passes output.
- Weights: Parameters that determine the importance of inputs.
- Bias: Shifts the activation function to fit the data better.
- Activation Function: Introduces non-linearity (e.g., sigmoid, ReLU, tanh).



Working of ANN (Forward Propagation)

1. Input is multiplied by weights and added to bias.
2. The result is passed through an activation function.
3. This process is repeated layer by layer until the output is produced.

Learning Process (Backpropagation)

4. Compare output with the actual value (calculate error).
5. Propagate the error backward to adjust weights using optimization algorithms (e.g., gradient descent).
6. Repeat over many epochs to minimize the error (training).