UNIT II : INTRODUCTION TO DEEP LEARNING & FRAMEWORKS

Syllabus For Unit 2

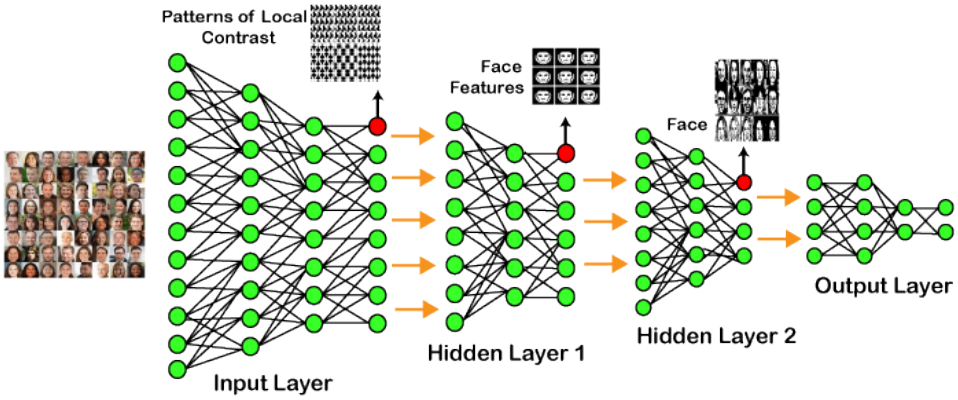
Deep Learning Basics: Intro, History, capabilities, the perceptron, Multi-Layer Perceptron, ANN architecture. Tensor Flow, Creating and Manipulating Tensor Flow Variables, Tensor Flow Operations, Placeholder Tensors, Managing Models over the CPU and GPU, Specifying the Logistic Regression Model in Tensor Flow, Logging and Training the Logistic Regression, Introduction to Keras, PyTorch.

❖ Introduction to Deep Learning

➢ What is Deep Learning?

- Deep learning is a subfield of machine learning that focuses on learning and improving on its own by examining computer algorithm.
- While machine learning uses simpler concepts, deep learning focuses on artificial neural networks with multiple layers, known as deep neural networks.
- These networks are designed to model and solve complex tasks by learning hierarchical representations of data.
- The term "deep" refers to the presence of multiple layers in these networks.
- Deep learning aided image classification, language translation, speech recognition. It can be used to solve any pattern recognition problem and without human intervention.

➢ How Does Deep Learning Works?

- Example of Deep Learning :
- 

- In the example given above, we provide the raw data of images to the first layer of the input layer.
- After then, these input layer will determine the patterns of local contrast that means it will differentiate on the basis of colours, luminosity, etc.
- Then the 1st hidden layer will determine the face feature, i.e., it will fixate on eyes, nose, and lips, etc. And then, it will fixate those face features on the correct face template.
- So, in the 2nd hidden layer, it will actually determine the correct face here as it can be seen in the above image, after which it will be sent to the output layer.
- Likewise, more hidden layers can be added to solve more complex problems, for example, if you want to find out a particular kind of face having large or light complexions.
- So, as and when the hidden layers increase, we are able to solve complex problems.

➢ Importance of Deep Learning

- Machine learning works only with sets of structured and semi-structured data, while deep learning works with both structured and unstructured data.
- Deep learning algorithms can perform complex operations efficiently, while machine learning algorithms cannot.
- Machine learning algorithms use labelled sample data to extract patterns, while deep learning accepts large volumes of data as input and analyses the input data to extract features out of an object.
- The performance of machine learning algorithms decreases as the number of data increases so to maintain the

performance of the model, we need a deep learning.

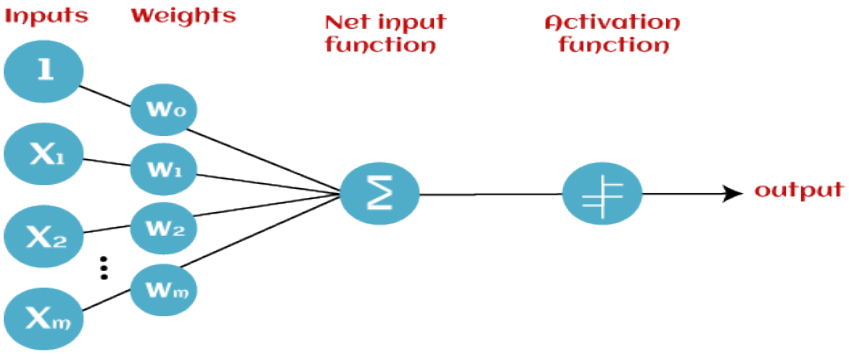
❖ The Perceptron

➢ What is The Perceptron Model in Machine Learning?

- Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks.
- Further, Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.
- Perceptron model is also treated as one of the best and simplest type of neural network, consisting of a single artificial neuron.
- However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., input values, weights and Bias, net sum, and an activation function.

➢ Basic Components of Perceptron

- Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:
-



◇ **Input Nodes or Input Layer:**

- This is the primary component of Perceptron which accepts the initial data into the system for further processing.
- Each input node contains a real numerical value. These are the values that the perceptron takes as input.

◇ **Wight and Bias:**

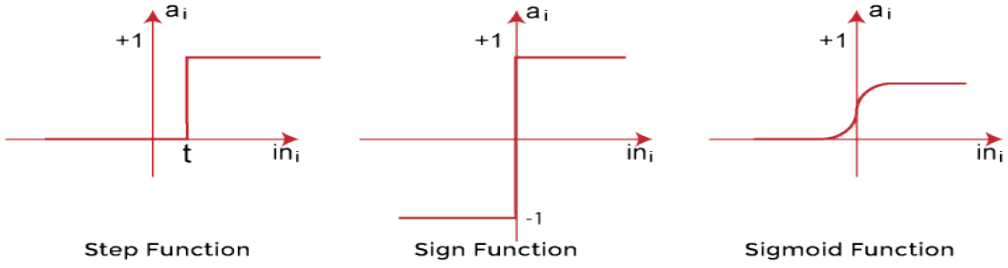
- Each input feature has an associated weight, which determines the strength of its influence on the perceptron's output. Weights can be positive, negative, or zero.
- Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

◇ **Weighted Sum/Net Sum:**

- The weighted sum is computed by multiplying each input feature by its corresponding weight and then summing up these products, along with a bias term (similar to the concept of a bias in deep networks).
- The bias is an additional parameter that allows the perceptron to shift its decision boundary.
- Weighted_Sum = (w1 * input1) + (w2 * input2) + ... + (wn * inputn) + bias

◇ **Activation Function:**

- The weighted sum of the inputs is passed through an activation function. The activation function introduces non-linearity into the perceptron's output.
- The activation function decides whether the perceptron should "fire" (output a 1) or "not fire" (output a 0) based on the weighted sum. The choice of the activation function depends on the problem at hand.
- Types of Activation Functions:
-



- **Sign Function:**
  - The "sign function" as an activation function is a simple binary activation function used in some neural network architectures.
  - In this context, the sign function, denoted as "sign(x)," is used to produce binary output values. It typically has the following characteristics:
    If 'x' is greater than or equal to 0, sign(x) outputs 1.
    If 'x' is less than 0, sign(x) outputs -1.
  - Mathematically, it can be represented as follows:
    sign(x) = 1, if x >= 0
    -1, if x < 0
- **Step Function:**
  - The step function is one of the simplest activation functions. It outputs 0 for inputs below a certain threshold and 1 for inputs above the threshold.
  - Mathematically, it can be defined as:
    f(x) = 0, if x < threshold
    f(x) = 1, if x >= threshold
- **Sigmoid Function:**
  - The sigmoid function is a smooth, S-shaped curve that maps its input to an output between 0 and 1. It is commonly used in the output layer of binary classification models.
  - The sigmoid function is defined as:
    f(x) = 1 / (1 + e^(-x))
- **Hyperbolic Tangent (tanh) Function:**
  - The hyperbolic tangent function is similar to the sigmoid but maps its input to an output between -1 and 1. It is often used in hidden layers of neural networks.
  - The tanh function is defined as:
    f(x) = (e^x - e^(-x)) / (e^x + e^(-x))
- **Rectified Linear Unit (ReLU):**
  - The ReLU activation function is one of the most widely used. It outputs zero for negative inputs and passes positive inputs as-is. This simple and computationally efficient function has helped fuel the success of deep learning.
  - The ReLU function is defined as:
    f(x) = max(0, x)
- **Leaky Rectified Linear Unit (Leaky ReLU):**
  - Leaky ReLU is a variation of the ReLU function that allows a small gradient for negative inputs, addressing the "dying ReLU" problem, where neurons can get stuck during training.
  - The Leaky ReLU function is defined as:
    f(x) = x, if x >= 0f(x) = αx, if x < 0 (where α is a small positive constant)
- **Parametric Rectified Linear Unit (PReLU):**
  - PReLU is an extension of Leaky ReLU where the slope of the negative part is learned during training. This makes it more flexible in modeling data.
  - The PReLU function is defined similarly to Leaky ReLU but with the slope parameter α being trainable.
- **Exponential Linear Unit (ELU):**
  - ELU is another alternative to ReLU that mitigates the vanishing gradient problem. It introduces a slight curvature for negative inputs while still being computationally efficient.
  - The ELU function is defined as:
    f(x) = x, if x >= 0f(x) = α * (e^x - 1), if x < 0 (where α is a positive constant)
- **Scaled Exponential Linear Unit (SELU):**
  - SELU is a self-normalizing activation function that can help neural networks converge faster and avoid the vanishing/exploding gradient problem.
  - It is designed for feedforward neural networks and requires specific initialization and dropout conditions.

◇ **Learning in Perceptron:**

– Perceptrons learn by adjusting their weights. The learning process involves presenting the perceptron with labelled data and then updating the weights to minimize the error between the predicted output and the actual target values.
– The weights are typically adjusted using a process known as the perceptron learning rule. This rule involves updating the weights based on the difference between the predicted output and the true target value.
– The learning process continues iteratively, gradually improving the perceptron's ability to make correct

predictions.

◇ **Limitations of Perceptron:**

– Perceptrons have limitations. They can only model linearly separable functions, which means they are not suitable for problems that require capturing complex, non-linear patterns. They cannot solve problems like XOR, which is not linearly separable.

– To overcome these limitations, more complex neural network architectures, like multi-layer perceptrons (MLPs) with multiple layers and non-linear activation functions, were developed.

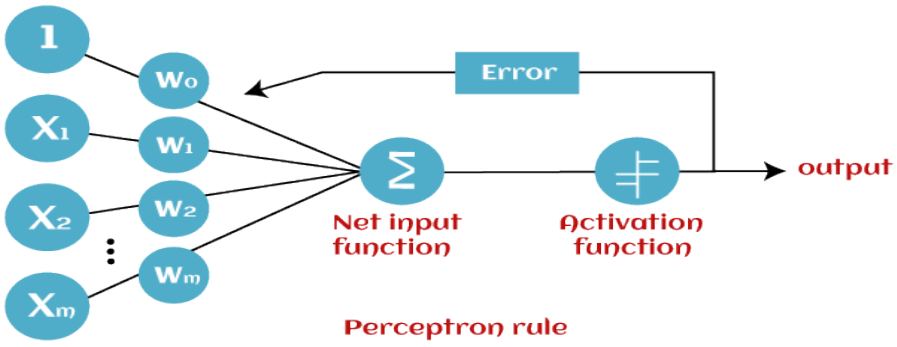
➢ How Does Perceptron Work?

– In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function.

– The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum.

– Then this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the **step function** and is represented by **'f'**.

–



– This step function or Activation function plays a vital role in ensuring that output is mapped between required values (0,1) or (-1,1).

– It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

– Perceptron model works in two important steps as follows:

(a) Step-1

- In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum.
- Mathematically, we can calculate the weighted sum as follows:

  $\sum wi*xi = x1*w1 + x2*w2 + \ldots wn*xn$
- Add a special term called bias 'b' to this weighted sum to improve the model's performance.

  $\sum wi*xi + b$

(b) Step-2

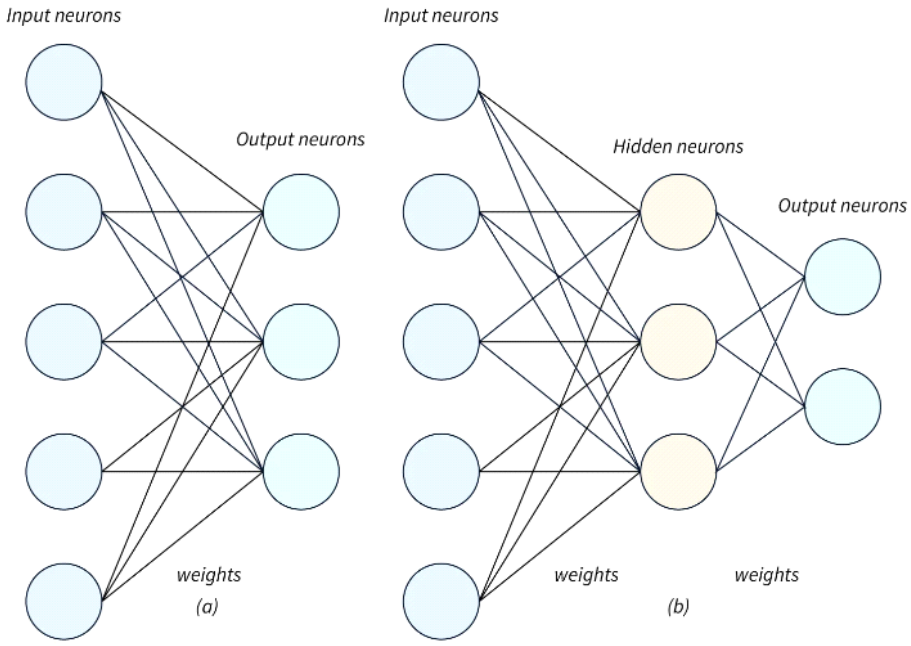
- In the second step, an activation function is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

  $Y = f(\sum wi*xi + b)$

➢ Types of Perceptron Model

– Based on layers, Perceptron models are divided into two types. These are as follows:

- Single Layer Perceptron Model
- Multi-Layer Perceptron Model



(a) Architecture of a single layer perceptron. The architecture consists of a layer on input neurons fully connected to a single layer of output neurons.

(b) Extension to a multi-layer perceptron including more than one layer of trainable weights. In this example, the network includes 3 layers: input, hidden and output layer. Each connection between two neurons is given by a certain weight.

–

◇ **Single Layer Perceptron Model**

– This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model.
– The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.
– In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters.
– Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.
– If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change.
– However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

◇ **Multi-Layer Perceptron Model**

– Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.
– The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:
  - **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
  - **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.
– Multi-layer perceptron is also known as MLP. It is fully connected dense layer transform any input dimension to the desired dimension.
– To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.
– A multi-layer perceptron has one input layer and for each input, there is one neuron, it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes.
– In the multi-layer perceptron diagram above, we can see that there are five input nodes and the hidden layer

has three nodes and output layer has two nodes.

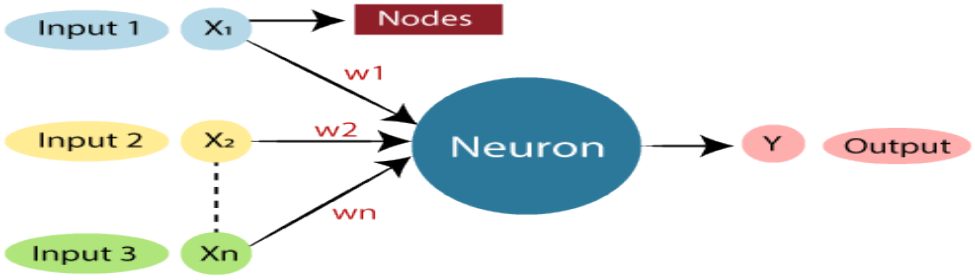
- The nodes in the input layer take input and forward their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.
- Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model.
- Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.
- A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.
- **Advantages of Multi-Layer Perceptron:**
  - A multi-layered perceptron model can be used to solve complex non-linear problems.
  - It works well with both small and large input data.
  - It helps us to obtain quick predictions after the training.
  - It helps to obtain the same accuracy ratio with large as well as small data.
- **Disadvantages of Multi-Layer Perceptron:**
  - In Multi-layer perceptron, computations are difficult and time-consuming.
  - In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
  - The model functioning depends on the quality of the training.

➢ Characteristics of Perceptron

- Perceptron is a machine learning algorithm for supervised learning of binary classifiers.
- In Perceptron, the weight coefficient is automatically learned.
- Initially, weights are multiplied with input features, and the decision is made whether the neuron is fired or not.
- The activation function applies a step rule to check whether the weight function is greater than zero.
- The linear decision boundary is drawn, enabling the distinction between the two linearly separable classes +1 and -1.
- If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown.

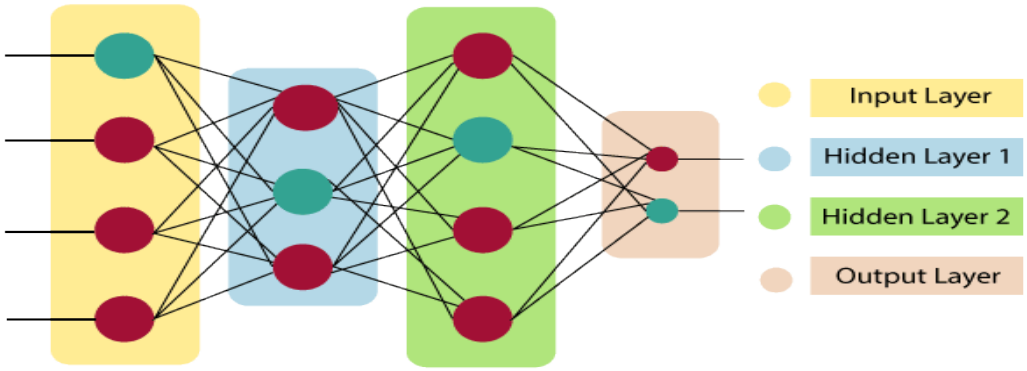
❖ Artificial Neural Network (ANN)

- Artificial Neural Networks (ANNs) are a class of machine learning models inspired by the structure and function of the human brain. They consist of interconnected artificial neurons (nodes) organized into layers.
-



➢ Architecture of Artificial Neural Network

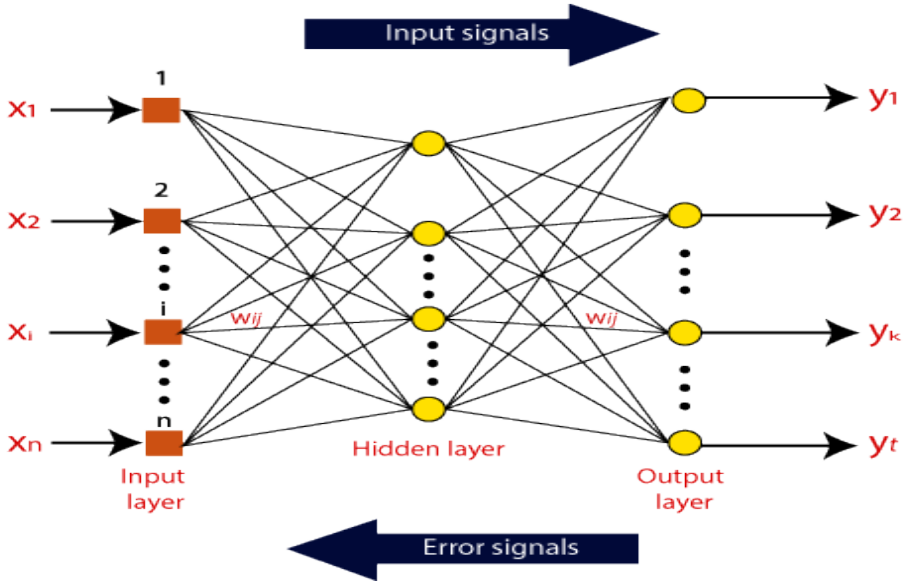
- The architecture of an ANN plays a crucial role in determining its capabilities and the types of problems it can solve. To understand the architecture of artificial neural network we must understand what a neural network consists of.
- In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers.



◇ **Input Layer:**
– The input layer is the first layer of the neural network.
– Neurons in the input layer represent the features or variables from the dataset that are used as inputs to the network.
– The number of neurons in the input layer is determined by the dimensionality of the input data.
◇ **Hidden Layers:**
– Between the input layer and the output layer, an ANN can have one or more hidden layers.
– The term "hidden" refers to the fact that these layers do not directly interact with the external data; they serve as intermediate layers for feature transformation.
– Each neuron in a hidden layer takes the outputs of neurons from the previous layer, computes a weighted sum, and passes the result through an activation function.
– Deep neural networks are characterized by having multiple hidden layers. The depth of the network plays a role in its ability to model complex functions.
◇ **Weights and Biases:**
– Each connection between neurons has an associated weight. These weights determine the strength of the connection.
– Additionally, each neuron has a bias term. Biases allow neurons to have some level of activation even when all inputs are zero.
– During training, the weights and biases are adjusted to minimize the error between the network's predictions and the actual target values. This is done through optimization techniques such as gradient descent.
◇ **Activation Functions:**
– Activation functions introduce non-linearity into the network. They are applied to the weighted sum of inputs before being passed to the next layer.
– Common activation functions include:
  - Sigmoid: S-shaped curve, output between 0 and 1.
  - ReLU (Rectified Linear Unit): Output is 0 if the input is negative, and the input itself if positive.
  - Tanh (Hyperbolic Tangent): S-shaped curve, output between -1 and 1.
– The choice of activation function depends on the specific task and the desired properties of the network.
◇ **Output Layer:**
– The output layer is the final layer of the network and produces the network's predictions or results.
– The number of neurons in the output layer depends on the nature of the task:
  - For binary classification, a single neuron with an appropriate activation function (e.g., sigmoid) is used.
  - For multi-class classification, there is one neuron per class, often with a softmax activation function.
  - For regression tasks, the output layer can have a single neuron for continuous predictions.
◇ **Loss Function:**
– The loss function, also known as the objective function, measures the error between the network's predictions and the actual target values.
– The choice of the loss function depends on the type of problem being solved. For example, mean squared error (MSE) is commonly used for regression, while cross-entropy loss is used for classification.

➢ How Do Artificial Neural Network Works?

– The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



- Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem).
- In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.
- If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1.
- Here the total of weighted inputs can be in the range of 0 to positive infinity. To keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.
- The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions.
- Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions.

◇ **Binary:**

- In binary activation function, the output is either a one or a 0. Here, to accomplish this, there is a threshold value set up.
- If the net weighted input of neurons is more than 1, then the final output of the activation function is returned as one or else the output is returned as 0.

◇ **Sigmoidal Hyperbolic:**

The Sigmoidal Hyperbola function is generally seen as an "S" shaped curve. Here the tan hyperbolic function is used to approximate output from the actual net input.

The function is defined as:

F(x) = (1/1 + exp(-????x))

➢ Types Of Artificial Neural Network (ANN)

◇ **Feedback ANN**

- A Feedback Artificial Neural Network, often referred to as a Recurrent Neural Network (RNN), is a type of neural network architecture designed to process sequences of data with a form of memory.
- Unlike feedforward neural networks, which have a strictly forward flow of information from input to output, RNNs incorporate feedback loops that allow them to maintain and update internal states as they process sequential data.

• **Recurrent Connections:**

- The defining characteristic of RNNs is the presence of recurrent connections, which introduce loops in the network's architecture. These loops enable information to be passed from one time step to the next.
- At each time step 't,' an RNN receives an input, produces an output, and updates its internal state or memory.

• **Memory and Hidden State:**

- RNNs have a hidden state that serves as their memory. This hidden state is a vector that encodes information from previous time steps.

- The hidden state at each time step is updated based on the input at that time step, the previous hidden state, and learned parameters (weights and biases).
- The hidden state effectively stores information about previous inputs, allowing the network to maintain context over time.

• **Processing Sequences:**

- RNNs are well-suited for processing sequences of data, where each element in the sequence is associated with a specific time step.
- In natural language processing, for example, each word in a sentence can be associated with a time step, allowing the RNN to consider the context of previous words when processing the current word.

◇ **Feed-Forward ANN**

- A Feed-Forward Artificial Neural Network (ANN), also known as a feedforward neural network or a multilayer perceptron (MLP), is one of the most common types of neural network architectures.
- It is designed to process data in a strictly forward direction, moving from input to output without any feedback loops or recurrent connections. Let's explore the structure and workings of a feed-forward ANN in detail:

• **Architecture:**

- A feed-forward ANN consists of three main types of layers:

(a) **Input Layer:** The input layer is the initial layer that receives the data or features. Each neuron in the input layer corresponds to a feature in the dataset. The number of neurons in the input layer depends on the dimensionality of the input data.

(b) **Hidden Layers:** Between the input and output layers, one or more hidden layers can be included. These hidden layers are responsible for performing computations on the input data, transforming it through a series of weighted connections and activation functions. The number of neurons in each hidden layer and the number of hidden layers are design choices that can vary based on the complexity of the problem.

(c) **Output Layer:** The output layer is the final layer of the network, and it produces the network's predictions or outputs. The number of neurons in the output layer depends on the nature of the task. For example:

i. For binary classification, a single neuron with an appropriate activation function (e.g., sigmoid) may be used.
ii. For multi-class classification, there may be one neuron per class, typically with a softmax activation function.
iii. For regression tasks, the output layer may have a single neuron for continuous predictions.

• **Feedforward Process:**

- The main operation in a feed-forward ANN is the feedforward process, where data or input features are passed through the network to generate a prediction or output.
- The feedforward process consists of the following steps:

i. Each input feature is associated with a weight, and the network computes the weighted sum of inputs for each neuron in the first hidden layer.
ii. An activation function is applied to the weighted sum, introducing non-linearity.
iii. The resulting values from the first hidden layer serve as inputs to the next layer, and the process is repeated for subsequent layers.
iv. The final output is produced by the neurons in the output layer.