FOUNDATION OF NEURAL NETWORK

PREPARED BY:

DHAVAL R. GANDHI

LECT IN IT

DR. S & S. S. GHANDHY COLLEGE OF ENGINEERING & TECHNOLOGY,SURAT

Learning Outcomes:-

- **Introduction to Neural Networks:**
- Understanding the Biological Neuron, Exploring the Artificial Neuron
- **Types of Activation Functions:**
- ReLU, Sigmoid, Hyperbolic Tangent Function
- **Architectures of Neural Network:**
- Single-layer feed forward network, Multi-layer feed forward ANNs, Recurrent network
- **Learning Process in ANN:**
- Number of layers, Direction of signal flow, Number of nodes in layers, Weight of interconnection between
- **Back Propagation**

Fundamental of Neural Network:

- ❑A neural net is an artificial representation of the human brain that tries to simulate its learning process.
- ❑An artificial neural network(ANN) is often called a "Neural Network" or simply Neural Net (NN).
- ❑Traditionally, the word neural network is referred to a network of biological neurons in the nervous system that process and transmit information.
- ❑Artificial neural network is an interconnected group of **artificial neurons** that uses **a mathematical model** or computational model for information processing based on a connectionist approach to computation.
- ❑The artificial neural networks are made of **interconnecting artificial neurons** which may share some properties of biological neural networks.

Fundamental of Neural Network:

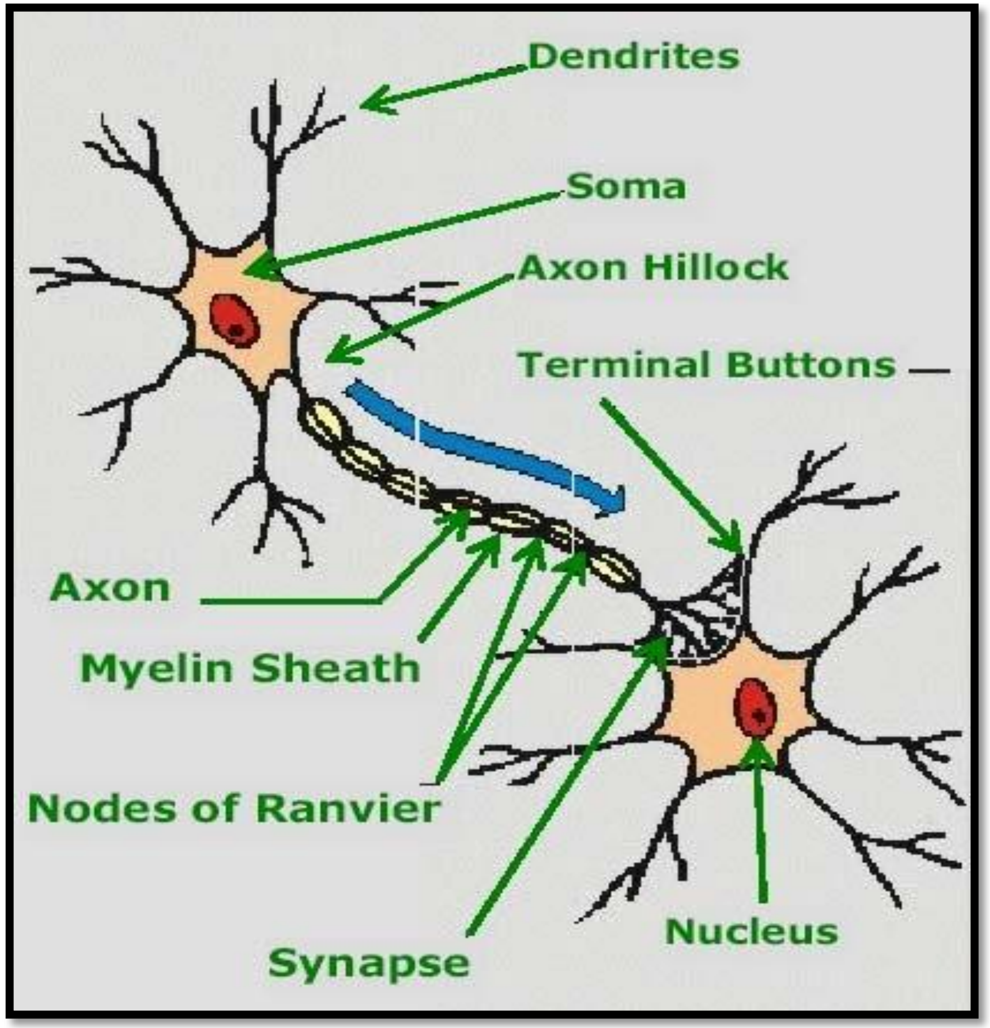
- Neural Computing is an **information processing paradigm**, inspired by biological system, composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems.
- Artificial Neural Networks (ANNs), like people, learn by example.
- An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.
- Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

Understanding a Biological Neuron:

- ❑**Neuron** is a special biological cell that processes information.
- ❑A biological neuron is a connection point in a Biological neural network.
- ❑Nodes are elementary unit in an ANN.
- ❑ Neurons are like the superheroes of our brain and nervous system! They are special cells.
- ❑Biological neural networks are like the brain's structure. We call it BNN for short The biological neuron is the basic building block of the nervous system.
- ❑The human brain consists of a large number, more than a billion of neural cells that process information.
- ❑Each cell works like a simple processor. The massive interaction between all cells and their parallel processing only makes the brain's abilities possible

Understanding a Biological Neuron:

- Biological neuron consists of five parts.
- **Neurons**
- **Soma or cell body**
- **Dendrites**
- **axon**
- **synapses**



Understanding a Biological Neuron:

Neurons:

- A Biological Neural Network (BNN) is like a team of tiny workers called neurons, and there are more than 100 billion of them in our brain! Each neuron has friends it talks to, and they send messages to the main part of the neuron called the cell body.
- They take in information from the world around us using our senses like seeing, hearing, and touching. Then, they tell our body to what do next like moving our arms and legs.
- Neurons also send messages to each other using electrical signals, like passing notes in a relay race.
- Neurons as the messengers and controllers of our body. They make sure everything works just like how a teacher guides a student.
- All the information from our senses is sent to our They process it, and then our brain responds to help us make decisions

Understanding a Biological Neuron:

Dendrites:

- They are tree-like branches, responsible for receiving the information from other neurons it is connected to.
- In other sense, we can say that they are like the ears of neuron.
- Dendrites are branching fibers that extend from the cell body or soma.
- Signals from outside move quickly through this network and are sent to other neurons through synapses.
- They work together to solve different tasks.

Understanding a Biological Neuron:

Soma or cell body:

- ❑It is the cell body of the neuron and is responsible for processing of information, they have received from dendrites.
- ❑Soma or cell body of a neuron contains the nucleus and other structures, support chemical processing and production of neurotransmitters.
- ❑Inside it, there is cell nucleus that get messages from other neurons through dendrites.
- ❑The cell body adds up all the messages to decide what to do and sends a signals through axons to other neurons.

Understanding a Biological Neuron:

Axon:

- It is just like a cable through which neurons send the information.
- Neurons also talk through pathways are called Axon.
- Axon is a singular fiber carries information away from the soma to the synaptic sites of other neurons (dendrites and somas), muscles, or glands.
- **Axon hillock** is the site of summation for incoming information.
- At any moment, the collective influence of all neurons that conduct impulses to a given neuron will determine whether or not an action potential will be initiated at the axon hillock and propagated along the axon.

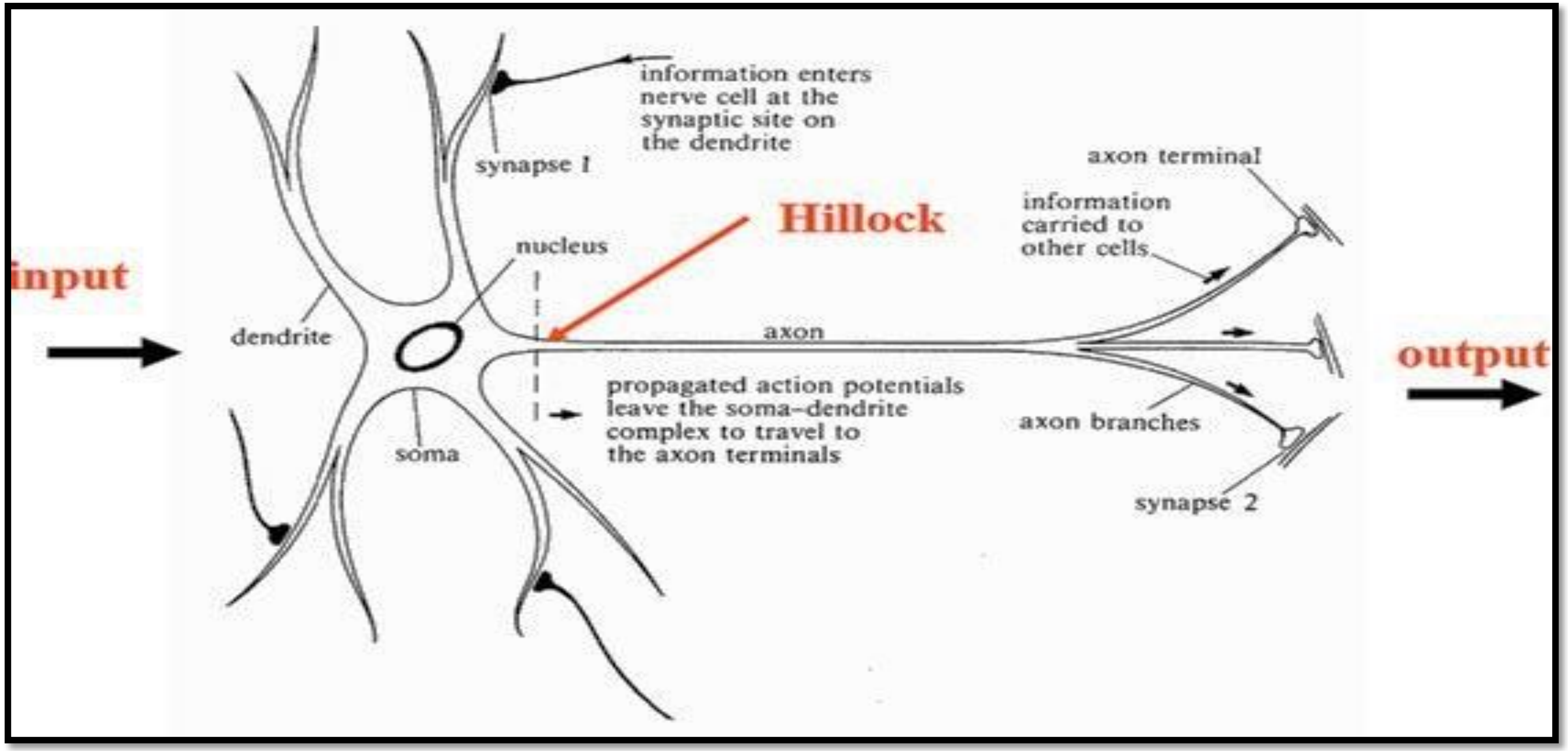
Understanding a Biological Neuron:

Synapses:

- ❑It is the connection between the axon and other neuron dendrites.
- ❑Synapse is the point of connection between two neurons or a neuron and a muscle or a gland.
- ❑Electrochemical communication between neurons takes place at these junctions.
- ❑They are like junctions and when they learn new thing, the connection change, just like when we learn new things.

Information flow in Biological Neuron:

The input /output and the propagation of information are shown below.


information enters nerve cell at the synaptic site on the dendrite
synapse 1
axon terminal
Hillock
information carried to other cells
nucleus
input
dendrite
axon
output
propagated action potentials leave the soma-dendrite complex to travel to the axon terminals
axon branches
soma
synapse 2

Understanding a Biological Neuron:

- **Dendrites** receive activation from other neurons.
- **Soma** processes the incoming activations and converts them into output activations.
- **Axons** act as transmission lines to send activation to other neurons.
- **Synapses** the junctions allow signal transmission between the axons and dendrites.
- The process of transmission is by diffusion of chemicals called neuro-transmitters.

Understanding a Biological Neuron:

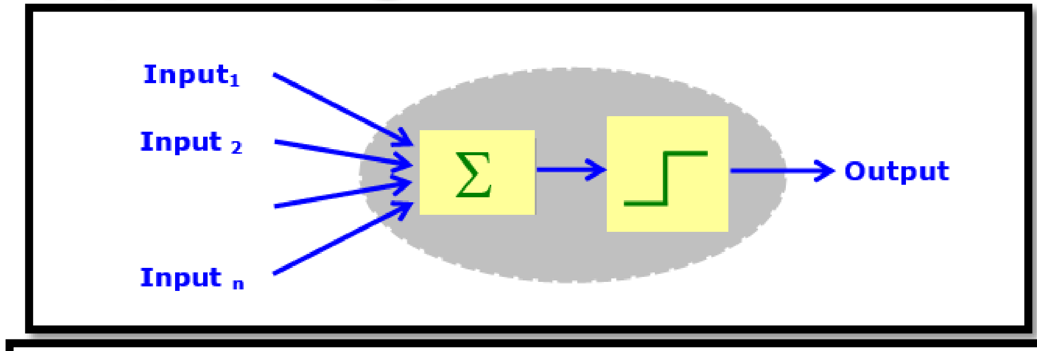
Threshold value:

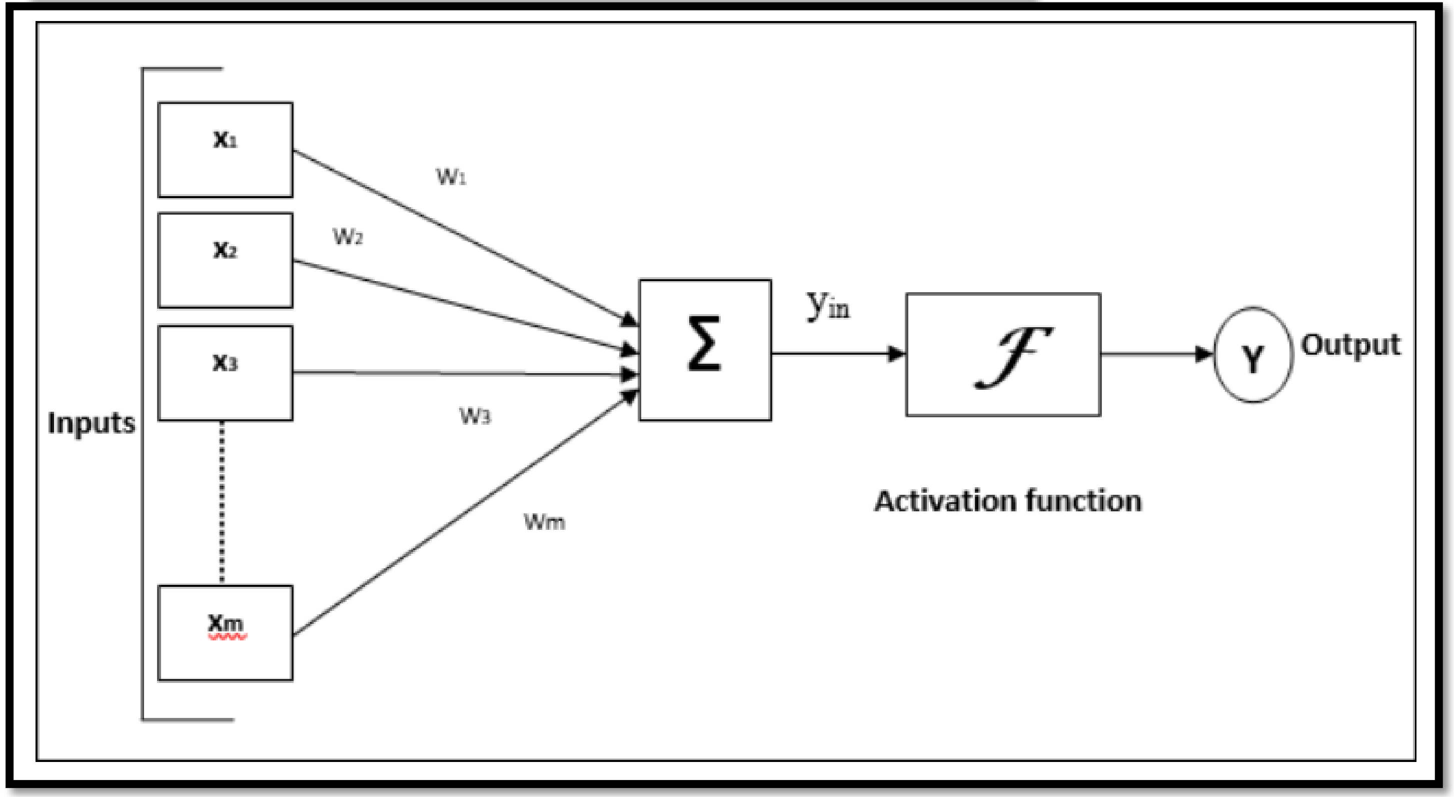
- ❑In the brain’s cells (neurons), a threshold value determines whether they should become active and transmit a message or remain inactive based on the signals they receive.
- ❑Ex: When you say this person has diabetes, it means then when you exceeds a certain blood sugar level, that limits is refereed to as threshold.

Exploring Artificial Neuron:

- ❑An artificial neuron is a connection point in an artificial neural network Artificial neural networks are like the human body's biological neural network.
- ❑An Artificial Neural Network (ANN) is a **computational model** inspired by the human brain's neural structure.
- ❑An Artificial Neural Network consists of **interconnected nodes** (same as biological neurons) organized into layers.
- ❑A Neural Network is also known as neural networks or neural nets.
- ❑An artificial neuron is a copy of a biological neuron.
- ❑An artificial neuron is also called a node. Node contains one or more inputs are separately weighted.
- ❑Every node has an activation function which is calculated on its state, and this helps determine its output signal.

Exploring Artificial Neuron:





Exploring Artificial Neuron:

- X1, X2. Xn are signals that come from different nodes or may be original input.
- In an artificial neural network (ANN) structure, there are multiple nodes (neurons), and the output of one node can become the input of another node.
- Each signal is associated with a weight W1, W2, in the image above. Wn are the respective weights
- A node has two main parts the summation and the activation function.

Exploring Artificial Neuron:

Summation:-

- The summation calculates a weighted sum like this:
- Every input multiplies by its corresponding weight X1*W1 +X2*W2 +Xn*Wn etc.
- (X1*W1 +X2*W2 +Xn*Wn) and then sends it to activation function.
- Input (X1,X2,..Xn) can be the original one or output of any other neurons.

$$v = \sum_{j=0}^{m} w_j x_j$$

Exploring Artificial Neuron:

Activation Function:-

- Input of activation function in weighted sum of Input signal to give one output signal.
- The activation function generates the output for a given node based on the input some Bias value.
- In other words, we can say it is used to convert the input signal on node of ANN to an output signal.

$$Y = F(y_{in})$$

Output = function $netinputcalculated$

Activation Function :

Activation Function:-

- ❑Activation functions are functions used in a neural network to compute the weighted sum of inputs and biases, which is in turn used to decide whether a neuron can be activated or not.
- ❑It manipulates the presented data and produces an output for the neural network that contains the parameters in the data.
- ❑The activation functions are also referred to as transfer functions in some literature.
- ❑depending on the function it represents and is used to control the output of neural networks across different domains.
- ❑For a linear model, a linear mapping of an input function to output is performed in the hidden layers before the final prediction for each label is given.
- ❑The input vector x transformation is given by
- ❑**$f(x) = wT . x + b$**
- ❑where, x = input, w = weight, and b = bias.

Activation Function :

Activation Function:-

- ❑Linear results are produced from the mappings of the above equation and the need for the activation function arises here, first to convert these linear outputs into non-linear output for further computation, and then to learn the patterns in the data.
- ❑The output of these models are given by

y = (w1 x1 + w2 x2 + … + wn xn + b)

- ❑These outputs of each layer are fed into the next subsequent layer for multilayered networks until the final output is obtained, but they are linear by default.
- ❑The expected output is said to determine the type of activation function that has to be deployed in a given network.
- ❑**Bias allows you to shift the activation function by adding a constant (i.e. the given bias) to the input.**

Activation Function :

Types of Activation Function:-

- A proper choice has to be made in choosing the activation function to improve the results in neural network computing.
- All activation functions must be monotonic, differentiable, and quickly converging with respect to the weights for optimization purposes.
- **Linear Activation Function**
- **Non Linear Activation Function**

Activation Function :

- **Linear Activation Function**
- A linear function is also known as a straight-line function where the activation is proportional to the input i.e. the weighted sum from neurons. It has a simple function with the equation:
- **$f(x) = ax + c$**
- The problem with this activation is that it cannot be defined in a specific range.
- Applying this function in all the nodes makes the activation function work like linear regression.
- The final layer of the Neural Network will be working as a linear function of the first layer.
- A good example of a regression problem is determining the cost of a house. We can use linear activation at the output layer since the price of a house may have any huge or little value.
- The neural network's hidden layers must perform some sort of non-linear function even in this circumstance.

Activation Function :

- **Non Linear Activation Function**
- The non-linear functions are known to be the most used activation functions.
- It makes it easy for a neural network model to adapt with a variety of data and to differentiate between the outcomes.
- The normal data input to neural networks is unaffected by the complexity or other factors.

Activation Function :

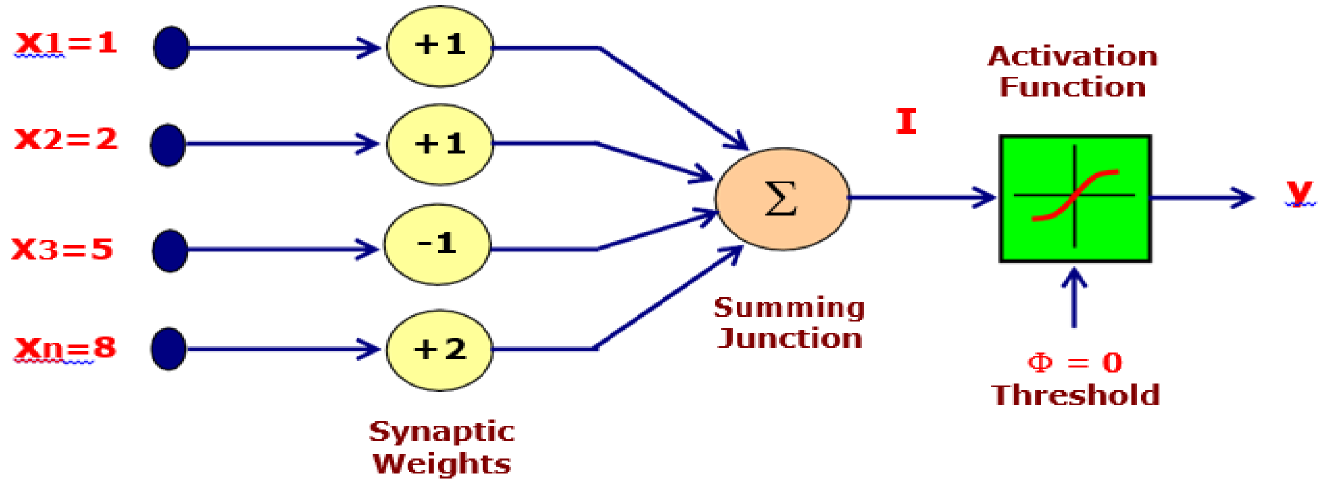


Fig Neuron Structure of Example

The output **I** of the network, prior to the activation function stage, is

$$I = X^T . W = \begin{bmatrix} 1 & 2 & 5 & 8 \end{bmatrix} \bullet \begin{pmatrix} +1 \\ +1 \\ -1 \\ +2 \end{pmatrix} = 14$$

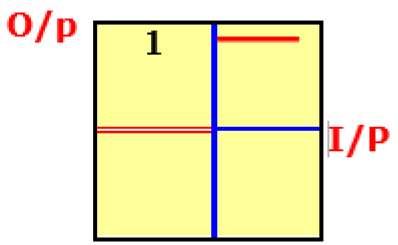
$$= (1 \times 1) + (2 \times 1) + (5 \times -1) + (8 \times 2) = 14$$

With a binary activation function the outputs of the neuron is:

$$y \text{ (threshold) } = 1;$$

Activation Function :

- **Binary Activation Function:**
- The activation function compares the input value to a threshold value.
- If the input value is greater than the threshold value, the neuron is activated.
- It's disabled if the input value is less than the threshold value, which means its output isn't sent on to the next or hidden layer.



1 if the weighted sum of the inputs is +ve,

0 if the weighted sum of the inputs is -ve.

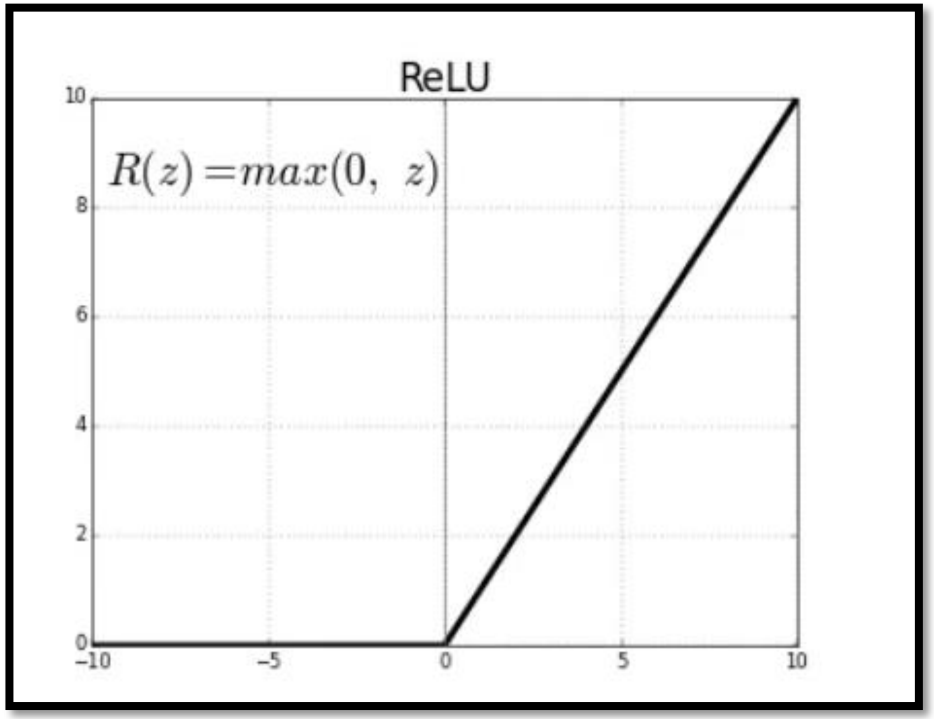
$$Y = f(I) = \begin{cases} 1 & \text{if } I \geq 0 \\ 0 & \text{if } I < 0 \end{cases}$$

Activation Function :

- **ReLU Function:**
- The **Rectified Linear Unit** is the most commonly used activation function in deep learning models.
- The function returns 0 if it receives any negative input, but for any positive value x it returns that value back.
- So it can be written as **$f(x)=\max(0,x)$**
- "x" represents the input to the activation function.
- "f(x)" is the output of the activation function.
- The ReLU activation function has a very simple behavior:
- If the input "x" is greater than or equal to zero, the output is equal to "x."
- If the input "x" is less than zero, the output is zero.

Activation Function :

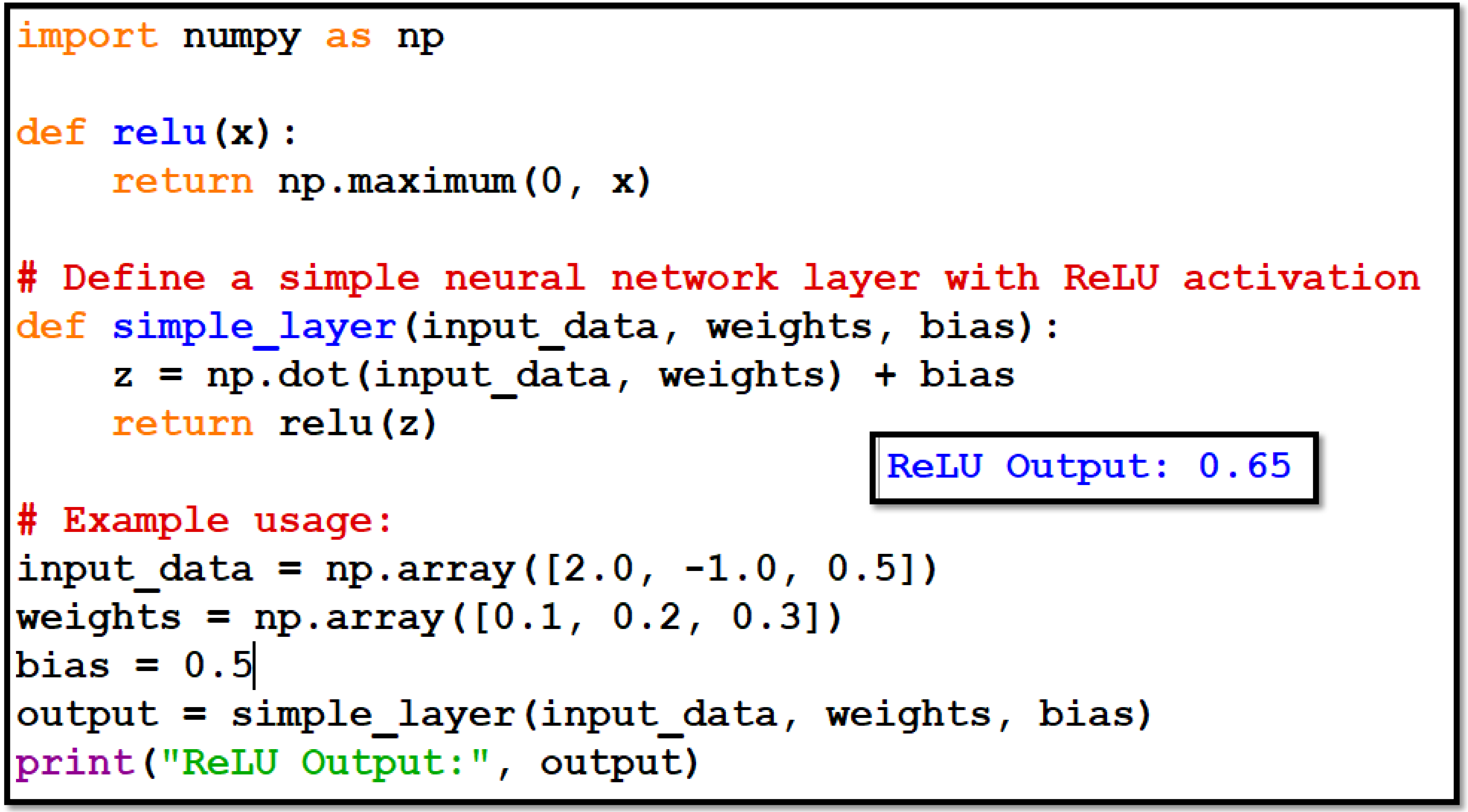
- **ReLU Function:**
- In other words, ReLU activation is linear for positive values of "x" and completely inactive (outputting zero) for negative values of "x."
- This sparsity of activation is one of the key advantages of ReLU because it encourages sparse representations and can help with the vanishing gradient problem during training.



Activation Function :

- **Advantages ReLU Function:**
- **Simplicity:** The ReLU function is computationally efficient and easy to implement, which makes it suitable for deep neural networks.
- **Mitigation of vanishing gradients**: ReLU helps address the vanishing gradient problem that can occur when training deep networks with sigmoid or tanh activation functions. The gradient for ReLU is either zero or one, which allows gradients to flow more freely during backpropagation.
- **Faster convergence:** ReLU networks often converge faster during training compared to networks using other activation functions, like sigmoid or tanh.
- However, ReLU does have a limitation known as the "dying ReLU" problem, where neurons can get stuck during training and never activate (output zero) for any input. This happens when the input to a ReLU neuron is always negative, and the neuron never updates its weights during training. Several variants of ReLU, such as Leaky ReLU and Parametric ReLU, have been introduced to address this issue by allowing a small, non-zero gradient for negative inputs.

Activation Function :



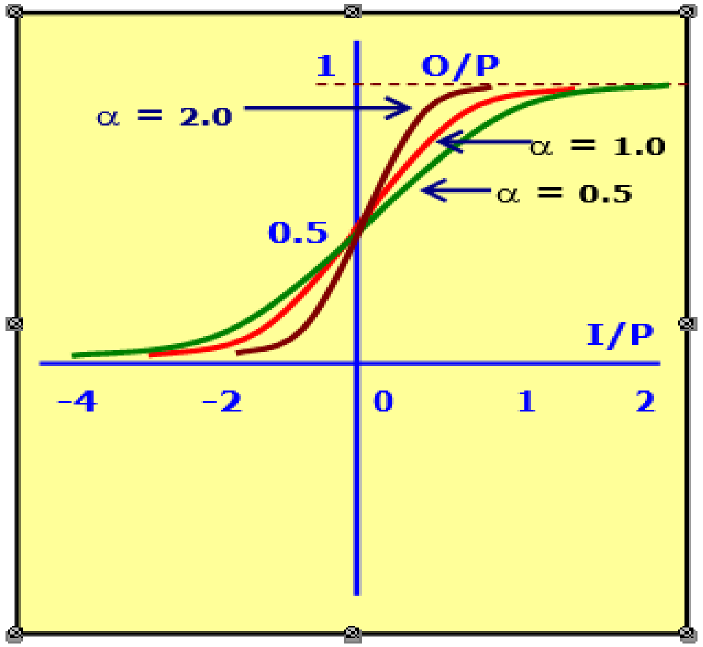
```python
import numpy as np

def relu(x):
    return np.maximum(0, x)

# Define a simple neural network layer with ReLU activation
def simple_layer(input_data, weights, bias):
    z = np.dot(input_data, weights) + bias
    return relu(z)

# Example usage:
input_data = np.array([2.0, -1.0, 0.5])
weights = np.array([0.1, 0.2, 0.3])
bias = 0.5
output = simple_layer(input_data, weights, bias)
print("ReLU Output:", output)
```

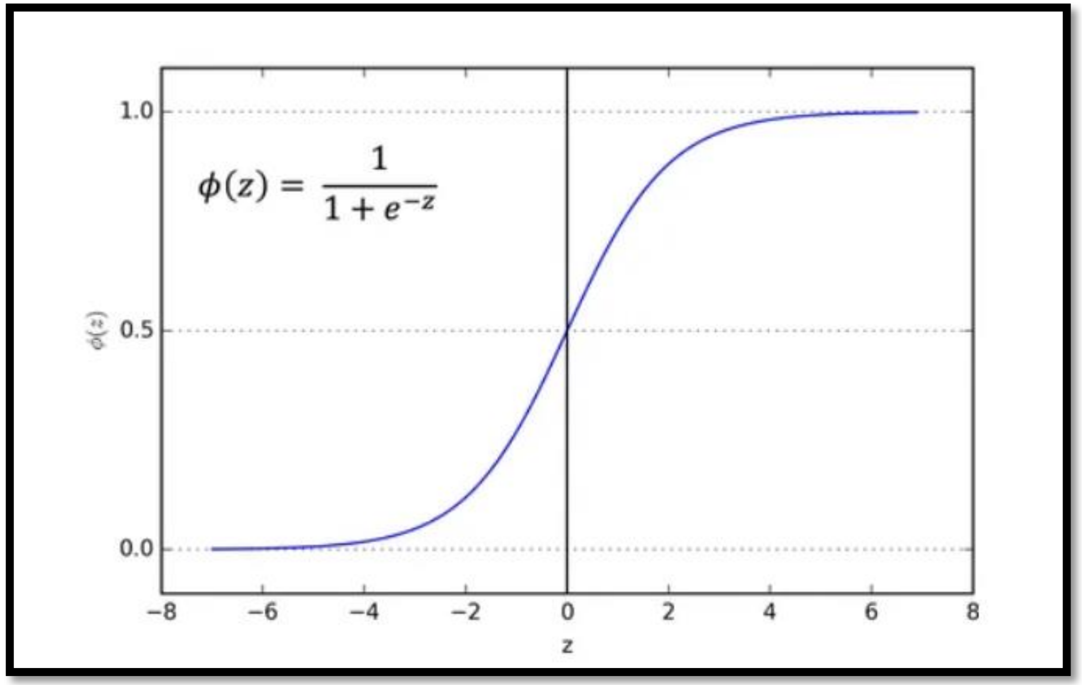
```
ReLU Output: 0.65
```

Activation Function :

- **Sigmoid Function:**
- The nonlinear curved S-shape function is called the sigmoid function.
- This is most common type of activation used to construct the neural networks.
- It is mathematically well behaved, differentiable and strictly increasing function.

Sigmoidal function





Activation Function :

- **Sigmoid Function:**
- The sigmoid activation function, also known as the logistic function, is a common activation function used in artificial neural networks.
- It maps the input to an output between 0 and 1, which can be interpreted as a probability.
- The sigmoid activation function translates the input ranged in $(-\infty,\infty)$ to the range in $(0,1)$
- **f(x) = 1 / (1 + e^(-x))**
- In this equation:
- "x" represents the input to the activation function.
- "f(x)" is the output of the activation function.
- "e" is the base of the natural logarithm (approximately 2.71828).

Activation Function :

- **Sigmoid Function:**
- Key properties and characteristics of the sigmoid activation function:
- **Range:** The output of the sigmoid function is always in the range (0, 1). As "x" becomes very negative, the output approaches 0, and as "x" becomes very positive, the output approaches 1.
- **S-shape:** The sigmoid function has an "S"-shaped curve. This shape introduces non-linearity into the neural network, allowing it to model complex relationships in the data.
- **Smoothness:** The sigmoid function is smooth and continuously differentiable, which is advantageous for gradient-based optimization algorithms like gradient descent during the training of neural networks.
- **Probabilistic interpretation:** Because the output of the sigmoid function can be interpreted as a probability, it is often used in binary classification problems where the goal is to estimate the probability of an input belonging to one of two classes (e.g., 0 or 1).

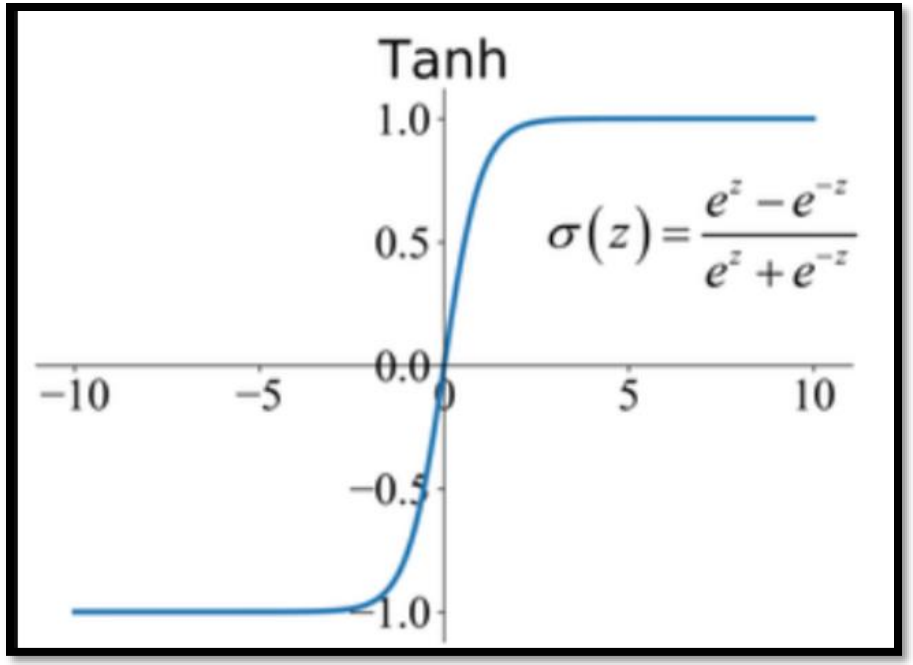
Activation Function :

```python
import numpy as np

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Define a simple neural network layer with Sigmoid activation
def simple_layer(input_data, weights, bias):
    z = np.dot(input_data, weights) + bias
    return sigmoid(z)

# Example usage:
input_data = np.array([2.0, -1.0, 0.5])
weights = np.array([0.1, 0.2, 0.3])
bias = 0.5
output = simple_layer(input_data, weights, bias)
print("Sigmoid Output:", output)
```

```
Sigmoid Output: 0.6570104626734988
```

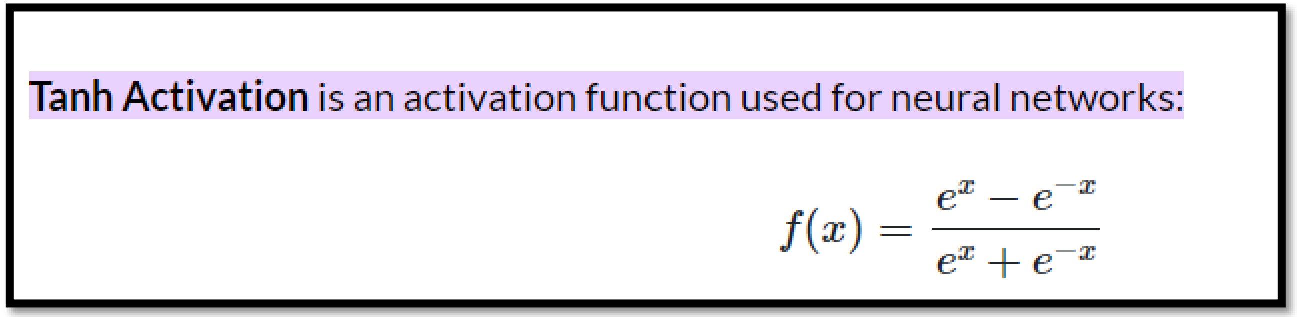
Activation Function :

- ❑**Hyperbolic Tangent Function:**
- ❑The tanh function is just another possible function that can be used as a non-linear activation function between layers of a neural network.
- ❑It shares a few things in common with the sigmoid activation function. Unlike a sigmoid function that will map input values between 0 and 1, the Tanh will map values between -1 and 1.



Activation Function :

- **Hyperbolic Tangent Function:**
- Similar to the sigmoid function, one of the interesting properties of the tanh function is that the derivative of tanh can be expressed in terms of the function itself.



Tanh Activation is an activation function used for neural networks:

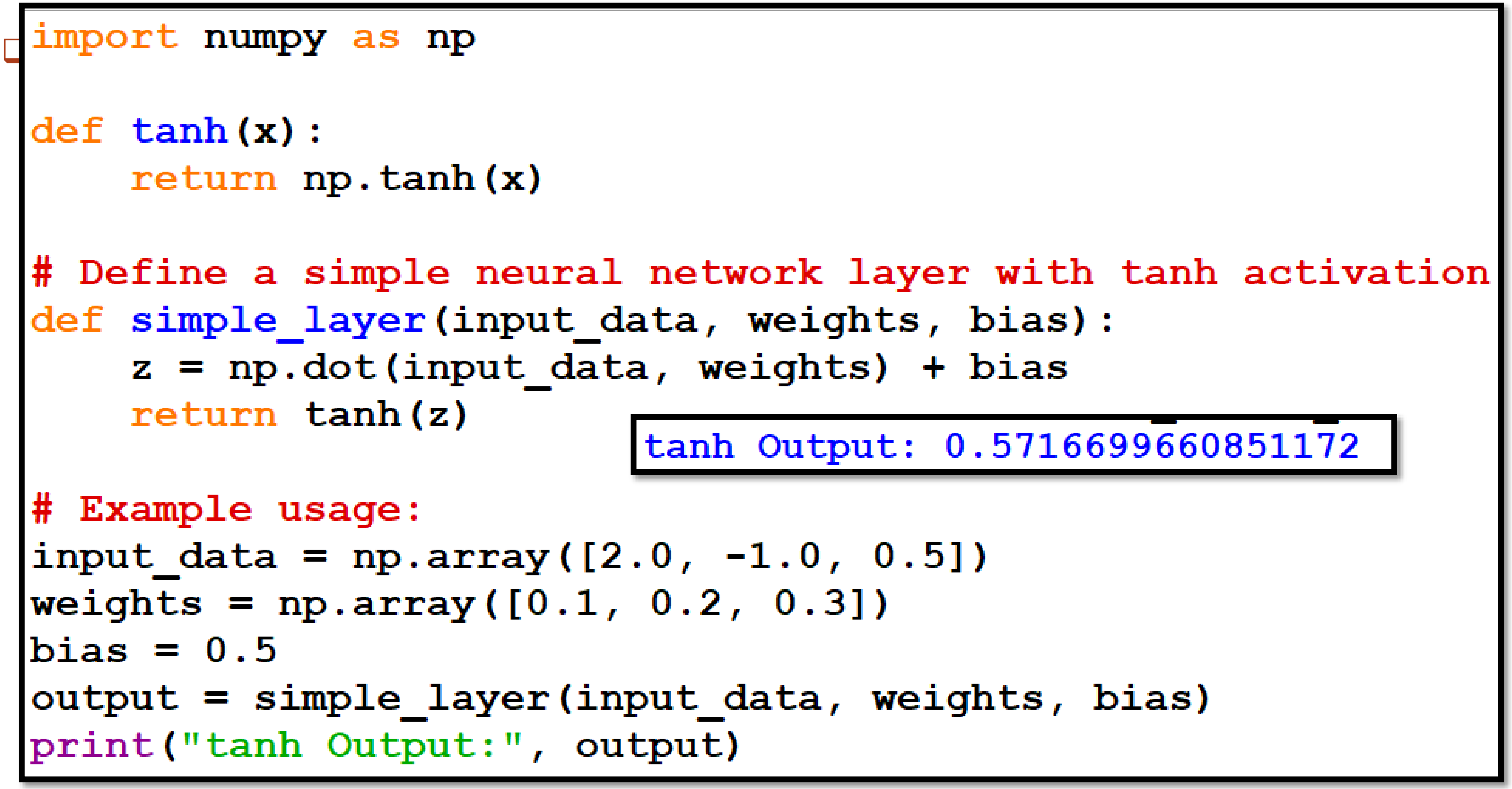
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- f(x) = (e^(x) - e^(-x)) / (e^(x) + e^(-x))
- In this equation:
- "x" represents the input to the activation function.
- "f(x)" is the output of the activation function.
- "e" is the base of the natural logarithm (approximately 2.71828).

Activation Function :

- **Hyperbolic Tangent Function:**
- Key properties and characteristics of the tanh activation function:
- **Range:** The output of the tanh function is in the range (-1, 1). As "x" becomes very negative, the output approaches -1, and as "x" becomes very positive, the output approaches 1.
- **S-shaped curve:** Similar to the sigmoid function, the tanh function has an "S"-shaped curve, which introduces non-linearity into neural networks, allowing them to capture complex relationships in the data.
- **Smoothness:** The tanh function is smooth and continuously differentiable, making it suitable for gradient-based optimization algorithms during the training of neural networks.
- **Zero-centered**: One advantage of the tanh function over the sigmoid function is that it is zero-centered, meaning that its output is centered around zero. This can help mitigate the vanishing gradient problem to some extent and may make it easier for the network to learn.

Activation Function :



```
import numpy as np

def tanh(x):
    return np.tanh(x)

# Define a simple neural network layer with tanh activation
def simple_layer(input_data, weights, bias):
    z = np.dot(input_data, weights) + bias
    return tanh(z)

# Example usage:
input_data = np.array([2.0, -1.0, 0.5])
weights = np.array([0.1, 0.2, 0.3])
bias = 0.5
output = simple_layer(input_data, weights, bias)
print("tanh Output:", output)
```

```
tanh Output: 0.5716699660851172
```

Architectures of Neural Network:

- ❑An artificial neural network enables computers to learn and make decisions in a human-like manner.
- ❑Like Biological Neurons, Artificial Neurons have the capability to process input and forward output to other nodes in the network. Here, connections are characterized by synaptic weights, which represent the significance of the connection.
- ❑Like BNN, neurons talk to each other using synapses, which have special values and weights.
- ❑As new data is received and processed, the synaptic weights change, and this is how learning occurs.
- ❑The decision of whether to send information on is called bias, and it's determined by an activation function built into the system.

Architectures of Neural Network:

- Relationship between Biological neural network and artificial neural network.

Biological Neural Network BNN	Artificial Neural network ANN
Dendrites	Inputs
Soma or Cell Body	Nodes
Synapse	Weights or Interconnections
Axon	Output

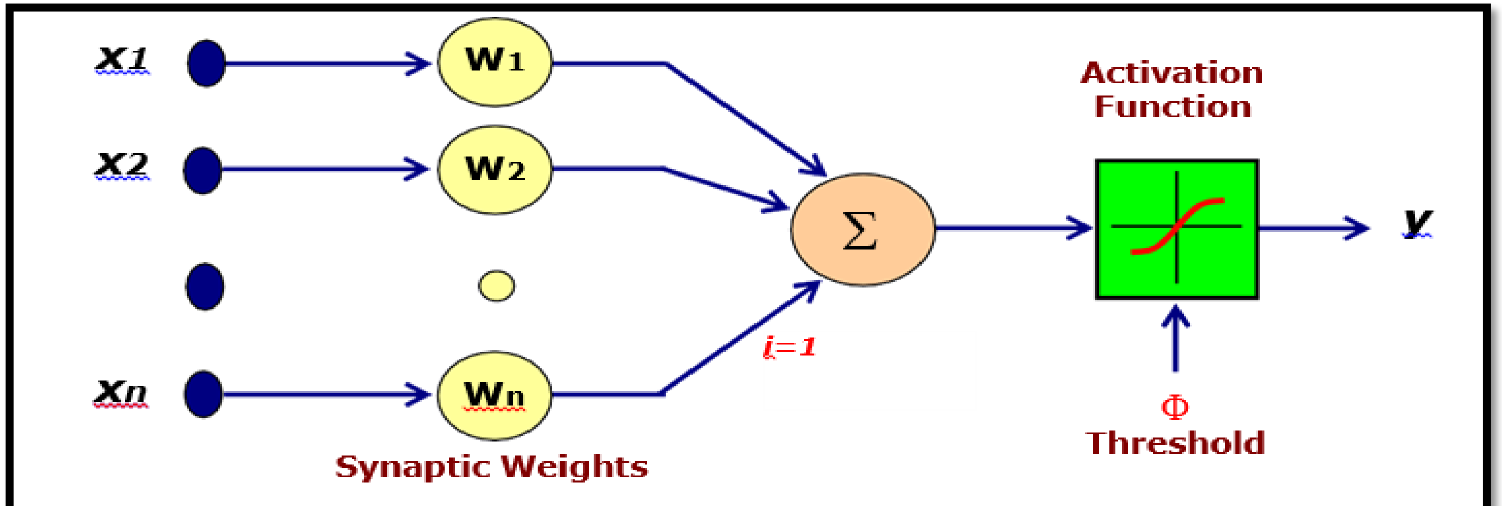
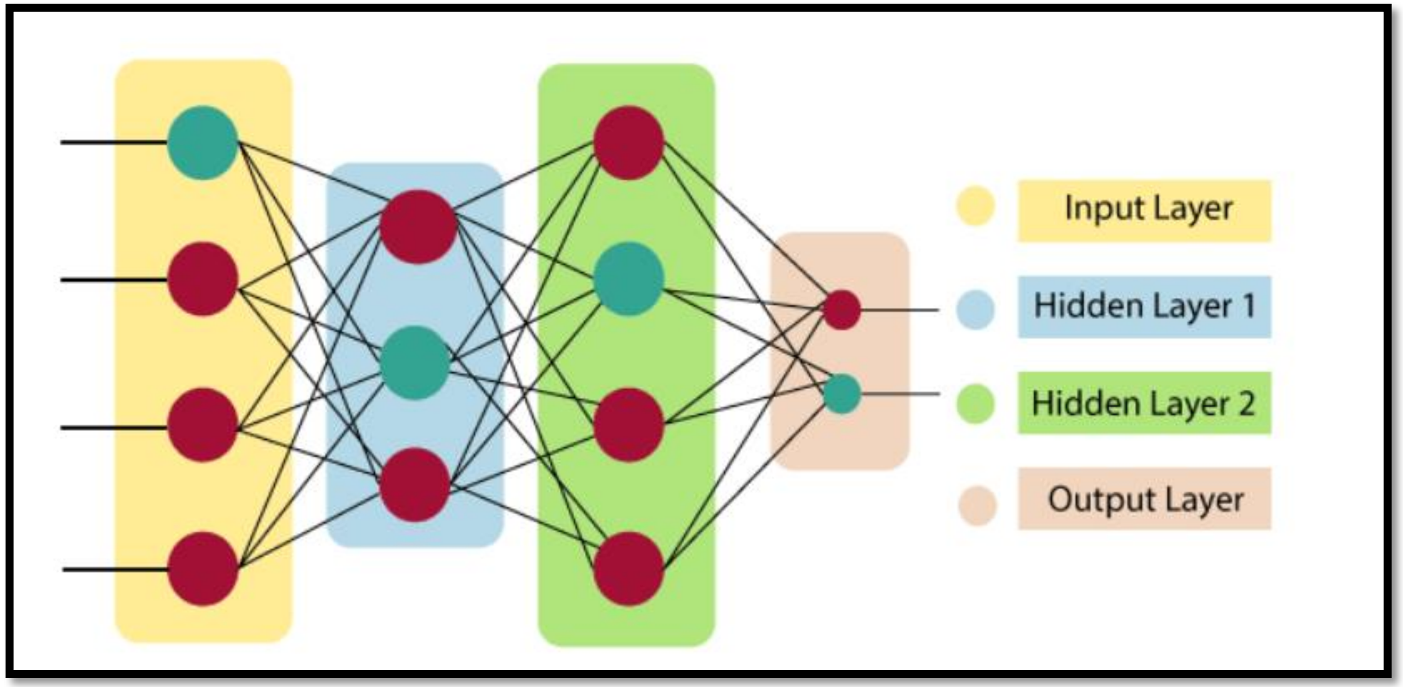


Fig Basic Elements of an Artificial Linear Neuron

Architectures of Neural Network:

- To understand the concept of the architecture of an artificial neural network, we have to 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.
- Lets us look at various types of layers available in an artificial neural network.



Architectures of Neural Network:

- Artificial Neural Network primarily consists of three layers:
- **Input Layer**
- **Hidden Layer**
- **Output Layer**

Architectures of Neural Network:

Input Layer:

- ❑As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer:

- ❑The hidden layer presents in-between input and output layers.
- ❑It performs all the calculations to find hidden features and patterns.

Output Layer:

- ❑The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

Architectures of Neural Network:

Output Layer:

- The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\boxed{\sum_{i=1}^{n} Wi * Xi + b}$$

- It determines weighted total is passed as an input to an activation function to produce the output.
- Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer.
- There are distinctive activation functions available that can be applied upon the sort of task we are performing.

Architectures of Neural Network:

Types of ANN:-

Feed-forward neural networks:

- In this ANN, the data or the input provided travels in a single direction.
- It enters into the ANN through the input layer and exits through the output layer while hidden layers may or may not exist.

Recurrent neural networks (RNNs):

- The Recurrent Neural Network saves the output of a layer and feeds this output back to the input to better predict the outcome of the layer.

Convolutional neural networks (CNNs):-

Modular neural networks:-

Architectures of Neural Network:

Single layer Feed Forward Network:-

- In this type of network, we have only two layers input layer and the output layer.
- The input layer receives the input signals .It does not do any processing.
- It does not count because no computation is performed in this layer, it simply forwards the data to the output layer.
- Information flow from the input layer to the output layer without any loops and feedback connections, so this is the simplest type of ANN.
- Only the output layer is performing computation and providing output and the flow of data passes through the Input to the Output layer only in one direction.
- That’s why this architecture is known as single layer feed forward network.

Architectures of Neural Network:

Single layer Feed Forward Network:-

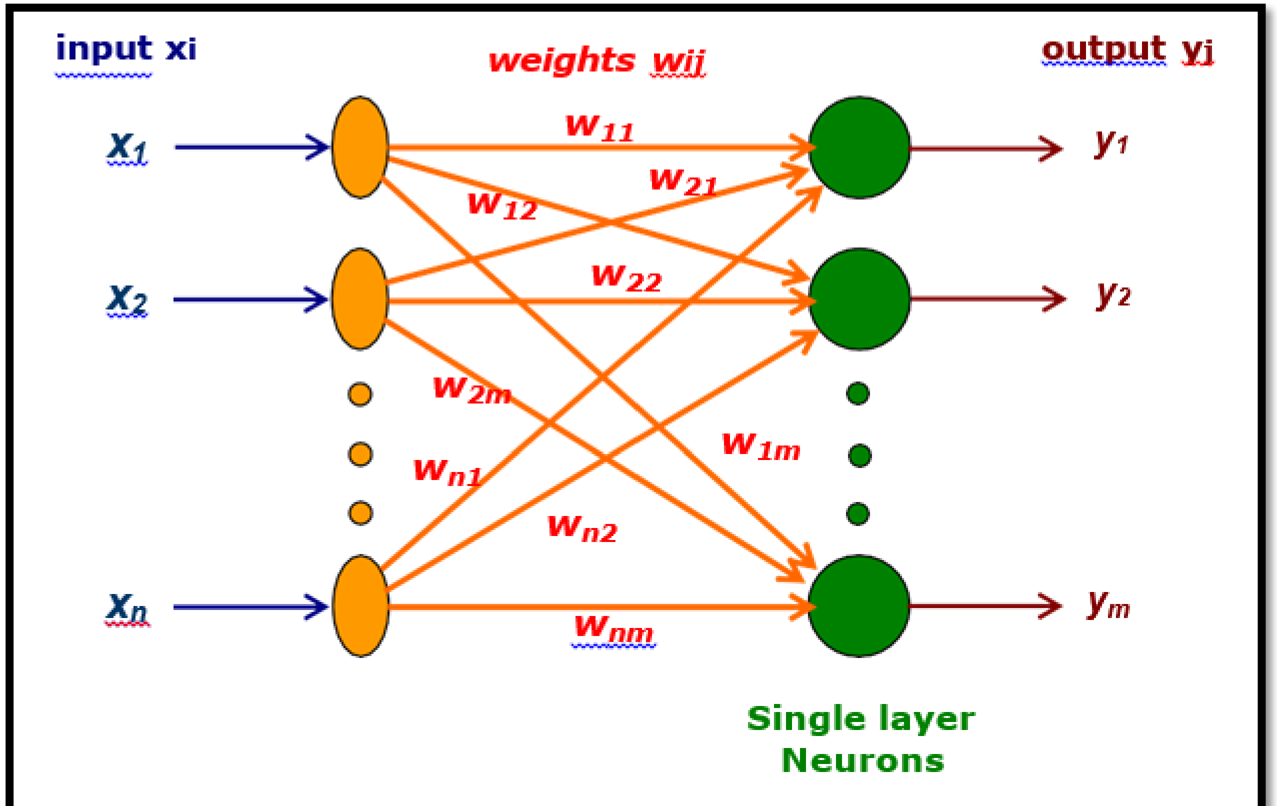


Fig. Single Layer Feed-forward Network

Architectures of Neural Network:

Single layer Feed Forward Network:-

- ❑The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken.
- ❑After this, the neurons collectively give the output layer to compute the output signals.
- ❑There is only one computational layer, which is output layer, making it single layer.

Advantages:-

- ❑Less complex, easy to design & maintain
- ❑Fast and speedy [One-way propagation]

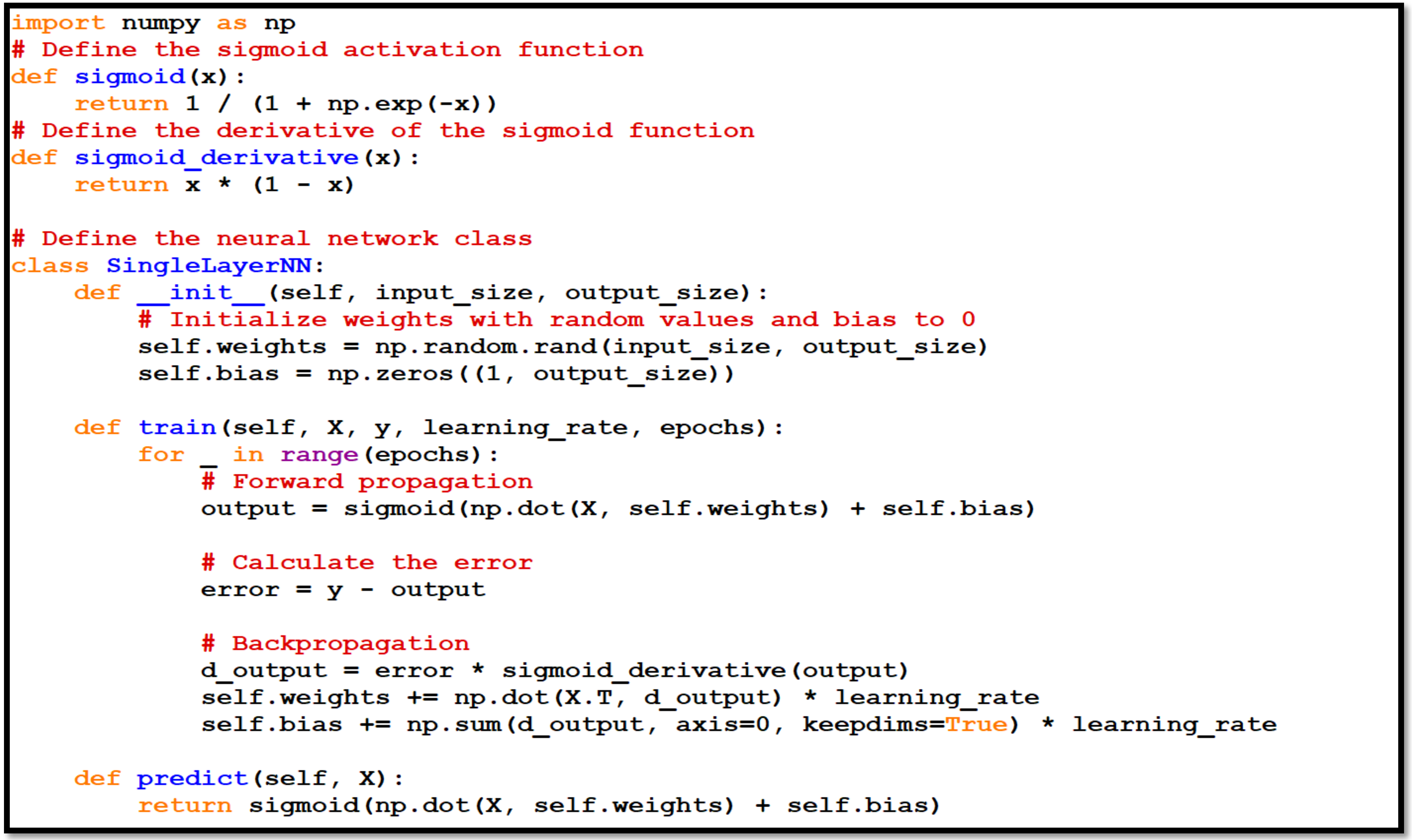
Disadvantages :

Cannot be used for deep learning [due to absence of dense layers]

Architectures of Neural Network:

- **Input Layer:** This layer receives the input data, which could be raw features or preprocessed data. Each input is connected to the nodes in the next layer.
- **Weighted Sum Calculation:** Each input is multiplied by a corresponding weight, and these products are summed up. A bias term may also be added to the weighted sum.
- **Activation Function:** The weighted sum is then passed through an activation function, typically a step function or a sigmoid function. The purpose of the activation function is to introduce non-linearity into the model.
- **Output Layer:** The result of the activation function is the output of the network. For binary classification tasks, the output could be a binary value (0 or 1). For multiclass classification, the output could be a vector of probabilities representing the likelihood of each class.
- **Training Algorithm:** The training of a single-layer feed forward network typically involves adjusting the weights and biases using a supervised learning algorithm such as the perceptron learning rule or the delta rule. These rules aim to minimize the error between the predicted output and the target output.

Architectures of Neural Network:



```python
import numpy as np
# Define the sigmoid activation function
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
# Define the derivative of the sigmoid function
def sigmoid_derivative(x):
    return x * (1 - x)

# Define the neural network class
class SingleLayerNN:
    def __init__(self, input_size, output_size):
        # Initialize weights with random values and bias to 0
        self.weights = np.random.rand(input_size, output_size)
        self.bias = np.zeros((1, output_size))

    def train(self, X, y, learning_rate, epochs):
        for _ in range(epochs):
            # Forward propagation
            output = sigmoid(np.dot(X, self.weights) + self.bias)

            # Calculate the error
            error = y - output

            # Backpropagation
            d_output = error * sigmoid_derivative(output)
            self.weights += np.dot(X.T, d_output) * learning_rate
            self.bias += np.sum(d_output, axis=0, keepdims=True) * learning_rate

    def predict(self, X):
        return sigmoid(np.dot(X, self.weights) + self.bias)
```

Architectures of Neural Network:

```python
# Example data
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [0]])

# Create and train the single-layer neural network
input_size = 2
output_size = 1
learning_rate = 0.1
epochs = 10000

model = SingleLayerNN(input_size, output_size)
model.train(X, y, learning_rate, epochs)

# Test the model
test_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
predictions = model.predict(test_data)

for i in range(len(test_data)):
    print(f"Input: {test_data[i]} - Predicted Output: {predictions[i]}")
```

```
Input: [0 0] - Predicted Output: [0.5]
Input: [0 1] - Predicted Output: [0.5]
Input: [1 0] - Predicted Output: [0.5]
Input: [1 1] - Predicted Output: [0.5]
```

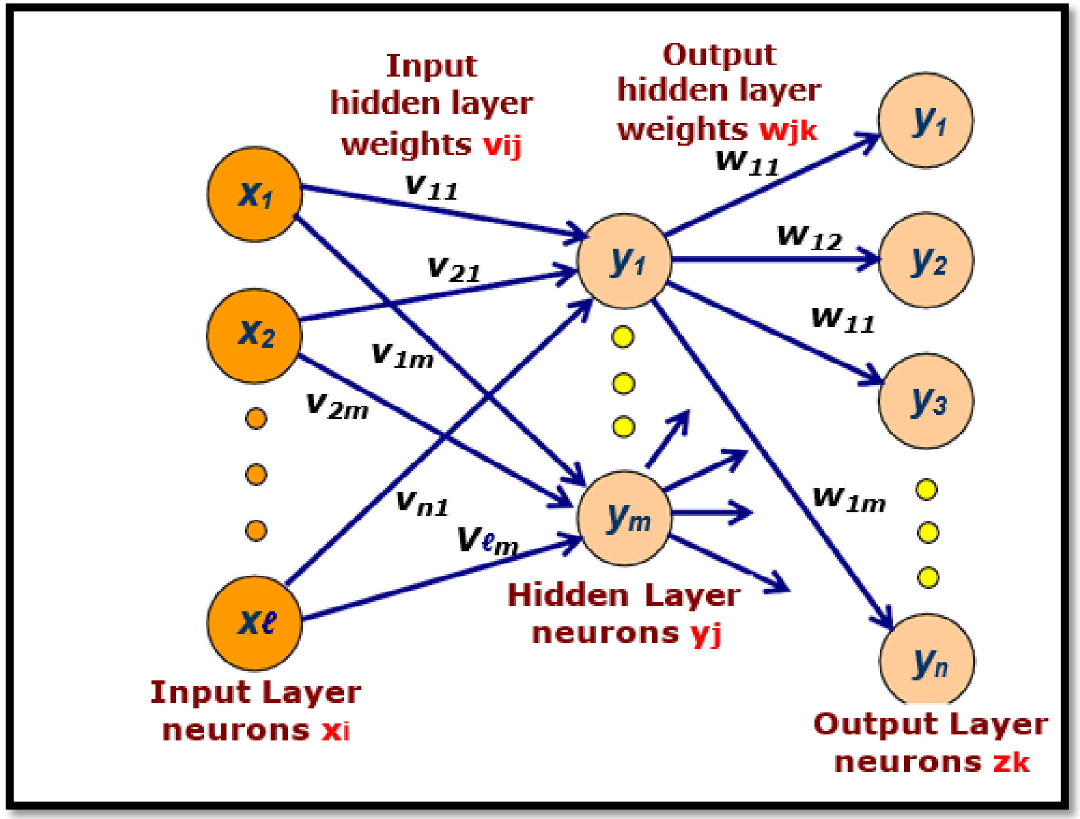
Architectures of Neural Network:

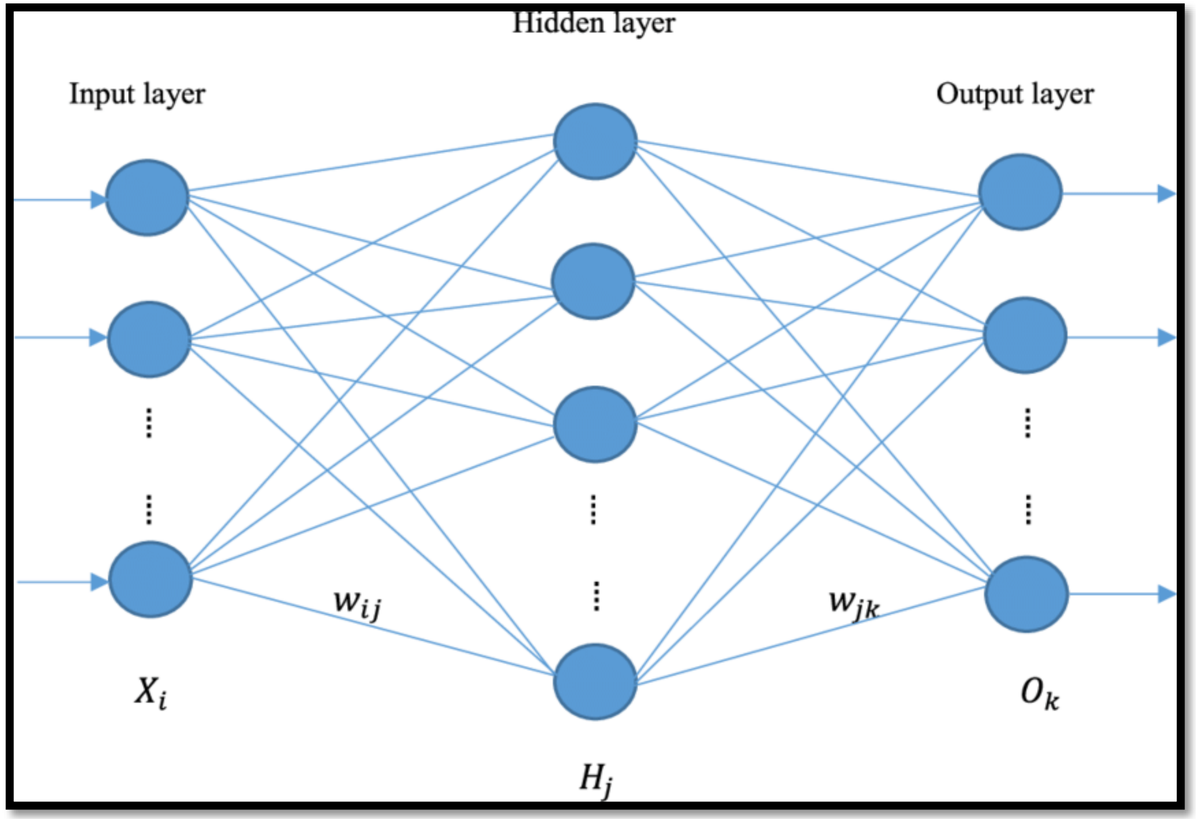
Multi layer Feed Forward ANNs:-

- The name suggests, it consists of multiple layers.
- There are three layers in ANNs
- Input Layer
- Hidden Layer
- Output Layer
- There is a hidden layer computationally stronger than single layer.
- It consists of an input layer, one or more hidden layers, and an output layer.
- Each layer is fully connected to the next one.
- The layers between the input and output layers are referred to as "hidden" layers because their values are not observed in the training data.

Architectures of Neural Network:

Multi layer Feed Forward ANNs:-





Multi layer Feed Forward ANNs:-

- **Input Layer:-**
- This layer receives the input data, which could be raw features or transformed data from previous layers and release it to hidden layers.
- **Hidden Layer:-**
- It can be single or multiple.
- The hidden layer is a mediator between the input and output layers.
- It transforms the input into something meaningful that the output layer can use in some way.
- In this layer, neurons receive input from the previous layer, compute the weighted sum, and send it to the neurons in the next layer.
- All the computations are done in this layer.
- These layers perform successive transformations of the input data. Each hidden layer usually has its own weights and biases, and the number of hidden layers can vary depending on the complexity of the task at hand.
- These layers are responsible for the non-linear mapping of inputs to the outputs.

Multi layer Feed Forward ANNs:-

- **Output Layer:-**
- The final layer of the network that produces the output.
- We get this result through different computations done by the hidden layers.
- The number of nodes in this layer depends on the specific task.
- For regression tasks, there is usually a single output node, while for classification tasks, there are as many output nodes as there are classes.
- In each layer, there are several nodes. All neurons interconnect with each other through the different layers.
- The nodes in each layer use the outputs of all nodes in the previous layer as inputs.
- Each neuron is typically assigned a weight that is adjusted during the learning process.
- The weight can be decreased or increased and it affects the strength of neuron’s signal.

Multi layer Feed Forward ANNs:-

- **Activation Function:-**
- Each node in the network applies an activation function to the weighted sum of its inputs. Commonly used activation functions include the sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU).
- **Back propagation Algorithm:**
- This is the primary method used for training the neural network.
- It adjusts the weights and biases of the network to minimize the difference between the predicted output and the actual output.

Multi layer Feed Forward ANNs:-

- Training a multi-layer feed forward ANN involves the following steps:
- **Forward propagation:** The input data is fed forward through the network, producing an output.
- **Error calculation:** The difference between the predicted output and the actual output is computed using a suitable error or loss function.
- **Backward propagation:** The error is then back-propagated through the network, and the weights and biases are adjusted using an optimization algorithm such as gradient descent.
- This process is repeated for multiple iterations until the network converges to a satisfactory solution.

Multi layer Feed Forward ANNs:-

Advantages:

- Universal Approximation
- Non-linear Mapping
- Back propagation Training
- Generalization

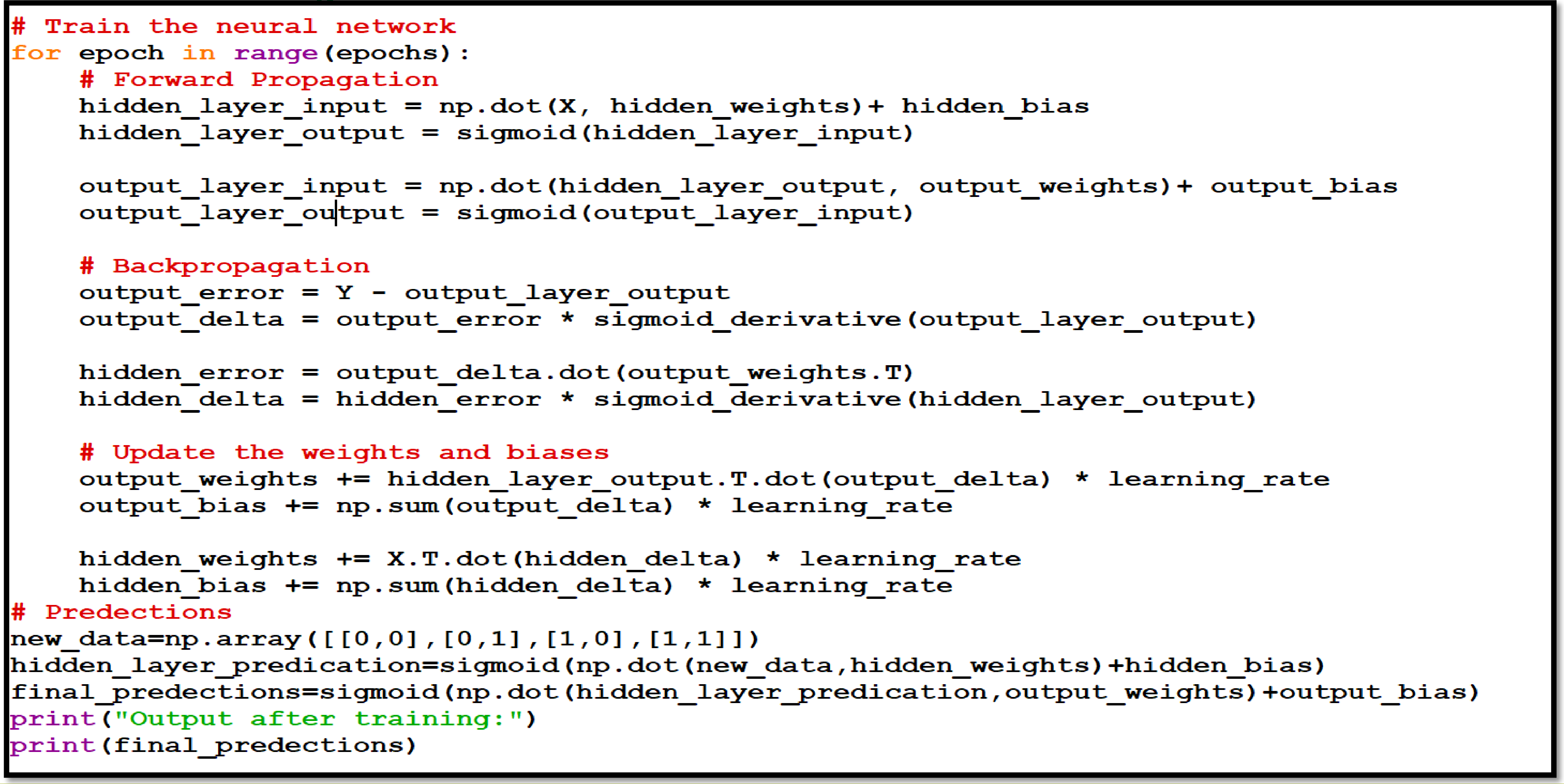
Disadvantages:-

- Black Box Nature
- Over fitting
- Computational Complexity
- Hyperparameter Sensitivity

Multi layer Feed Forward ANNs:-

```
import numpy as np

# Define the sigmoid activation function
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# Define the derivative of the sigmoid function
def sigmoid_derivative(x):
    return x * (1 - x)

# Define the dataset
X = np.array([[0, 0],[0, 1],[1, 0],[1, 1]])
Y = np.array([[0], [1], [1], [0]])

# Initialize weights and biases with random values
input_size, hidden_size, output_size = 2, 2, 1

hidden_weights = np.random.rand(input_size,hidden_size)
hidden_bias = np.random.rand(hidden_size)
output_weights = np.random.rand(hidden_size,output_size)
output_bias = np.random.rand(output_size)

# Set the learning rate
learning_rate = 0.1
epochs=10000
```

Multi layer Feed Forward ANNs:-



```
# Train the neural network
for epoch in range(epochs):
    # Forward Propagation
    hidden_layer_input = np.dot(X, hidden_weights)+ hidden_bias
    hidden_layer_output = sigmoid(hidden_layer_input)

    output_layer_input = np.dot(hidden_layer_output, output_weights)+ output_bias
    output_layer_output = sigmoid(output_layer_input)

    # Backpropagation
    output_error = Y - output_layer_output
    output_delta = output_error * sigmoid_derivative(output_layer_output)

    hidden_error = output_delta.dot(output_weights.T)
    hidden_delta = hidden_error * sigmoid_derivative(hidden_layer_output)

    # Update the weights and biases
    output_weights += hidden_layer_output.T.dot(output_delta) * learning_rate
    output_bias += np.sum(output_delta) * learning_rate

    hidden_weights += X.T.dot(hidden_delta) * learning_rate
    hidden_bias += np.sum(hidden_delta) * learning_rate
# Predections
new_data=np.array([[0,0],[0,1],[1,0],[1,1]])
hidden_layer_predication=sigmoid(np.dot(new_data,hidden_weights)+hidden_bias)
final_predections=sigmoid(np.dot(hidden_layer_predication,output_weights)+output_bias)
print("Output after training:")
print(final_predections)
```

Recurrent Network:-

- ❑Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step.
- ❑In traditional neural networks, all the inputs and outputs are independent of each other, but in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.
- ❑Thus RNN came into existence, which solved this issue with the help of a Hidden Layer.
- ❑The main and most important feature of RNN is its **Hidden state**, which remembers some information about a sequence.
- ❑The state is also referred to as Memory State since it remembers the previous input to the network.
- ❑It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output.
- ❑This reduces the complexity of parameters, unlike other neural networks.

Recurrent Network:-

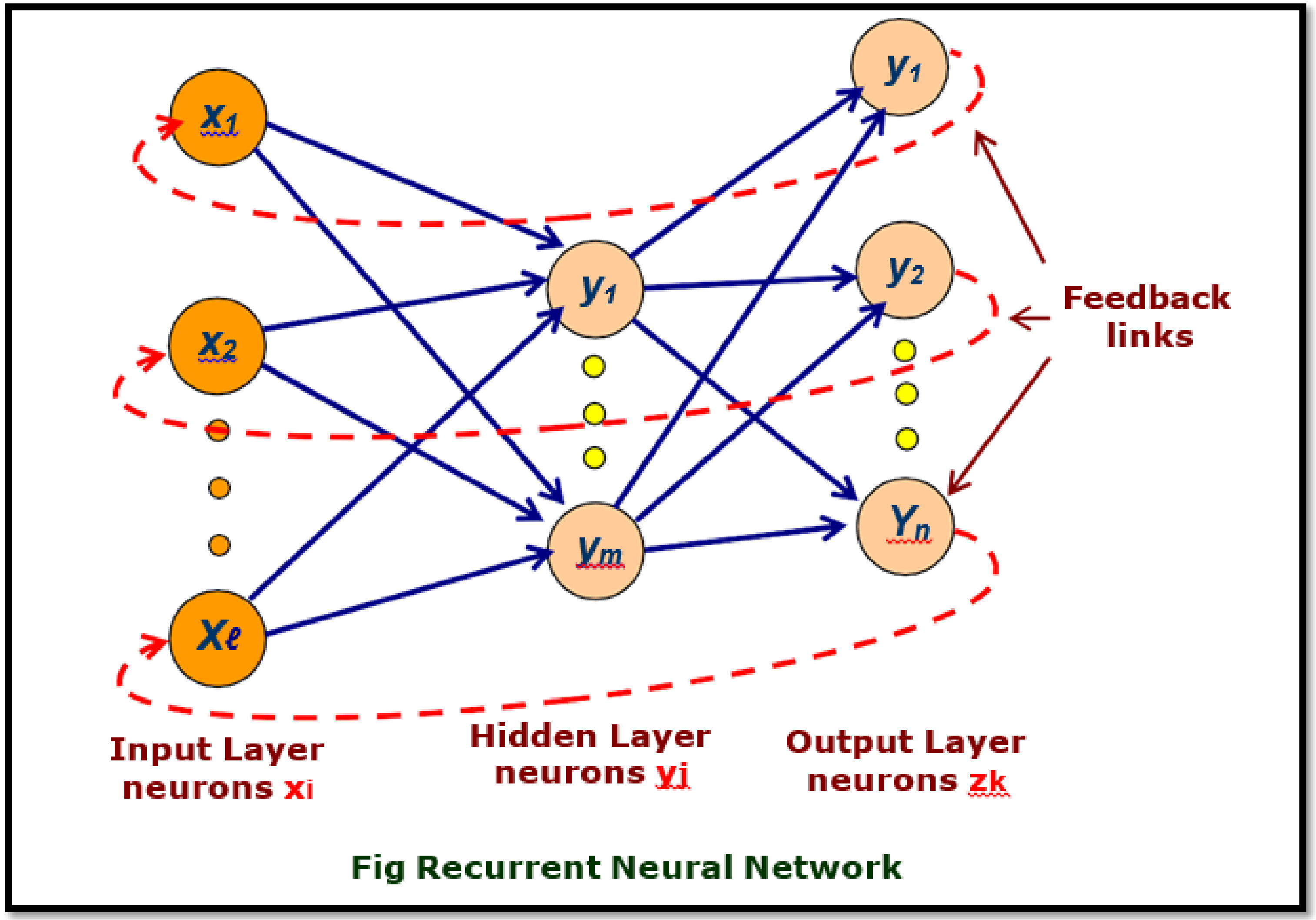


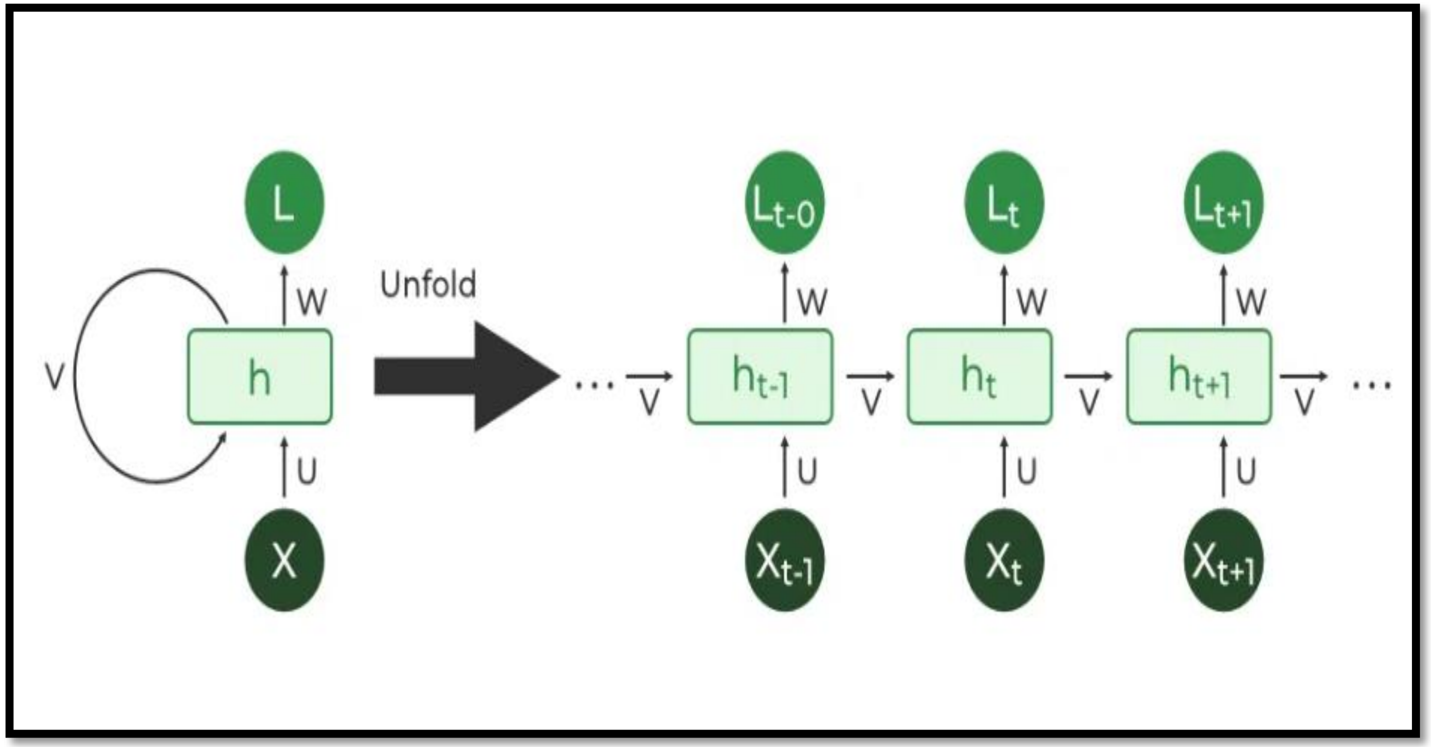
Fig Recurrent Neural Network

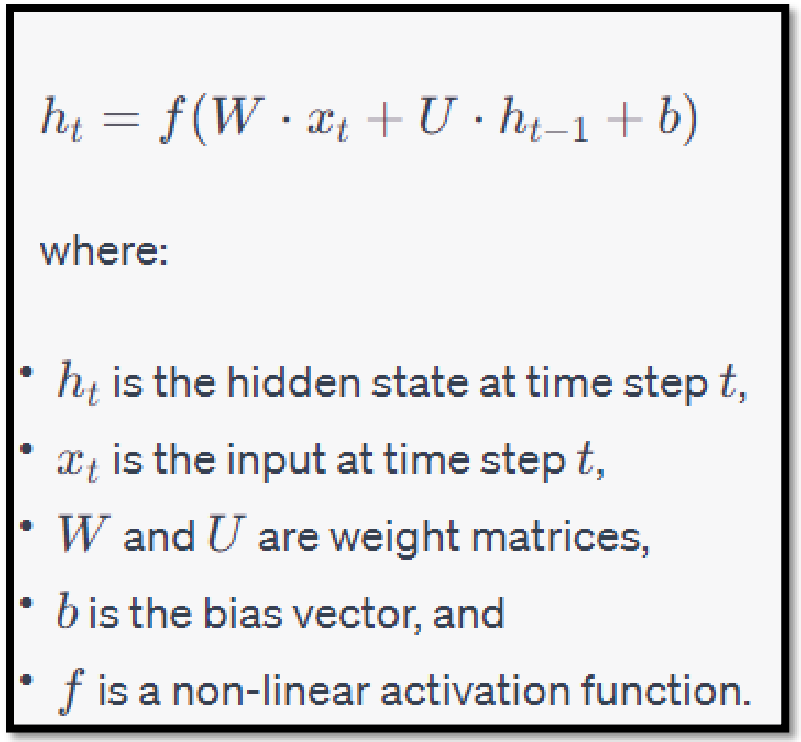
Recurrent Network:-

- The key components of a Recurrent Neural Network (RNN) include:
- **Input Layer:**
- This layer receives the input data at each time step in the sequence.
- **Hidden State:**
- The hidden state is the memory of the RNN that captures information about the sequence it has seen so far.
- It is updated at each time step based on the current input and the previous hidden state.
- The hidden state enables the network to retain information over time and learn dependencies within sequential data.
- This hidden state is updated at every time step to signify the change in the knowledge of the network about the past.
- **Recurrent Connection:**
- This connection allows the hidden state to be passed from one time step to the next, enabling the network to consider previous information while processing the current input.
- This recurrent connection is what gives the network its temporal dynamic behavior.

Recurrent Network:-

❑Hidden state formula:-





$$h_t = f(W \cdot x_t + U \cdot h_{t-1} + b)$$

where:

- h_t is the hidden state at time step t,
- x_t is the input at time step t,
- W and U are weight matrices,
- b is the bias vector, and
- f is a non-linear activation function.

Recurrent Network:-

- **Activation Function**:
- An activation function is applied to the combined input at each time step, transforming it into an output or hidden state representation. Common choices for the activation function include the hyperbolic tangent function (tanh) or the Rectified Linear Unit (ReLU) function.
- **Output Layer:**
- The output layer produces the final output or prediction based on the information processed through the hidden states.
- The specific design of the output layer depends on the task the RNN is being used for, such as classification, regression, or sequence generation.
- **Loss Function:**
- A loss function measures the difference between the predicted output and the actual target output.
- The choice of the loss function depends on the specific task, with common options including mean squared error for regression tasks and categorical cross-entropy for classification tasks.

Recurrent Network:-

Advantages :-

- An RNN remembers each and every piece of information through time.
- It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
- Recurrent neural networks are even used with convolutional layers to extend the effective pixel neighborhood.

Disadvantages:-

- Gradient vanishing and exploding problems.
- Training an RNN is a very difficult task.
- It cannot process very long sequences if using tanh or relu as an activation function.

Learning Process in ANN:-

- ❑Number of Layers
- ❑Direction of signal flow
- ❑Number of nodes in layers
- ❑Weight of interconnection between nodes

Learning Process in ANN:-

- **Number of Layers:-**
- In a neural network, the number of layers refers to how many different levels or layers of nodes there are in the neural network.
- Each layer can have multiple nodes.
- There are three layers in ANNs
- **Input Layer**
- **Hidden Layer**
- **Output Layer**

Learning Process in ANN:-

- **Direction of signal flow:-**
- In a neural network, the direction of signal flow can be unidirectional or bidirectional.
- It can go from the start to the end which is known as feed forward.
- It can go back and forth which is known as recurrent feedback.
- There are two types of signal flow in neural network.
- **Feed Forward direction**
- **Recurrent Feedback direction**

Learning Process in ANN:-

- **Direction of signal flow:-**
- **Feed Forward direction:-**
- When signal flows straight from the beginning to the end without turning back is known as feed forward direction.
- It passes through different layers and reaches the output layer.
- **Recurrent Feedback direction:-**
- In some networks, the signal can go back and forth like a loop is known as recurrent feedback direction.
- In Recurrent neural networks, there are feedback loops, allowing the signal to flow back from the output to the input or through hidden layer.

Learning Process in ANN:-

- **Number of Nodes in a layer:-**
- The number of nodes in a layer of an Artificial Neural Network (ANN) is determined by the specific architecture and design of the network, which is influenced by the complexity of the task and the nature of the data.
- The number of nodes in a layer affects the network's capacity to learn and represent complex relationships within the data.
- The number of nodes in each layer determine how much information the neural network can process.

Learning Process in ANN:-

- **Weight of interconnection between nodes:-**
- In an Artificial Neural Network (ANN), the interconnections between nodes are represented by weights.
- These weights play a crucial role in determining the output of the network and are adjusted during the training process to minimize the error between the predicted output and the target output.
- The weights essentially represent the strength of the connections between nodes and control the flow of information throughout the network.
- It's crucial to initialize the weights properly and update them effectively during training to ensure that the network can learn the underlying patterns and relationships in the data.

Back Propagation:-

- ❑Back propagation is an algorithm that back propagates the errors from the output nodes to the input nodes.
- ❑Therefore, it is simply referred to as the backward propagation of errors. It uses in the vast applications of neural networks in data mining like Character recognition, Signature verification, etc.
- ❑The primary objective of backpropagation is to adjust the weights of the connections in the network to minimize the difference between the actual output and the desired output.
- ❑Back propagation is a widely used algorithm for training feed forward neural networks.
- ❑It computes the gradient of the loss function with respect to the network weights. It is very efficient, rather than naively directly computing the gradient concerning each weight.
- ❑This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

Back Propagation:-

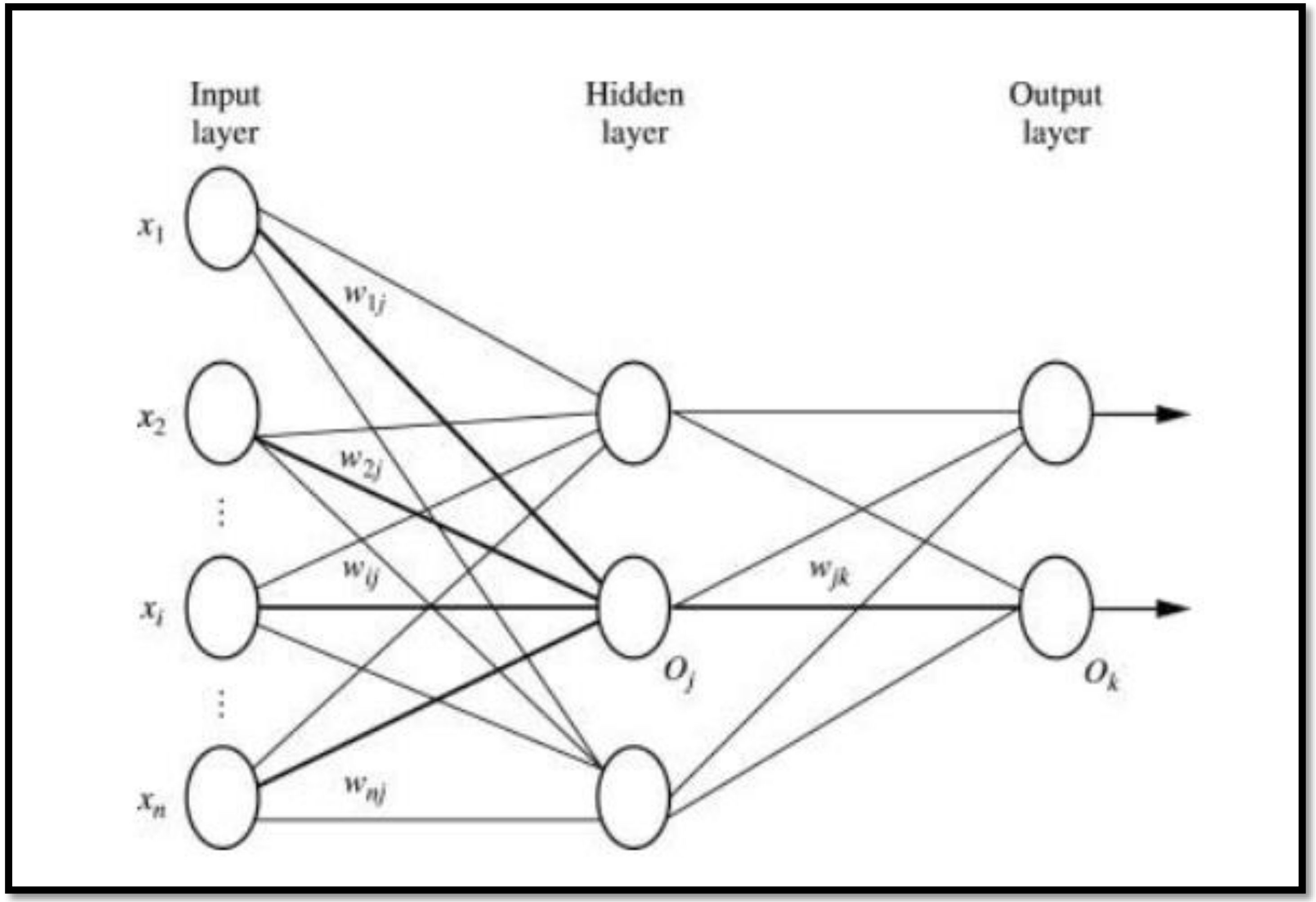
Features of Back propagation:-

- it is different from other networks in respect to the process by which the weights are calculated during the learning period of the network.
- training is done in the three stages :
- The feed-forward of input training pattern
- The calculation and back propagation of the error
- Updation of the weight and bias

Working of Back propagation:-

- Neural networks use supervised learning to generate output vectors from input vectors that the network operates on.
- It Compares generated output to the desired output and generates an error report if the result does not match the generated output vector.
- Then it adjusts the weights according to the bug report to get your desired output.

Back Propagation:-



Back Propagation:-

Back propagation Algorithm:-

- ❑**Step 1:** Inputs X, arrive through the preconnected path.
- ❑**Step 2:** The input is modeled using true weights W. Weights are usually chosen randomly.
- ❑**Step 3:** Calculate the output of each neuron from the input layer to the hidden layer to the output layer.
- ❑**Step 4:** Calculate the error in the outputs

  Back propagation Error= Actual Output – Desired Output

- ❑**Step 5**: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.
- ❑**Step 6:** Repeat the process until the desired output is achieved.

Back Propagation:-

- There are two phases of Back Propagation.
- **Forward Pass:**
- The algorithm begins by performing a forward pass through the network, where the input data is fed into the network, and the output is computed.
- Each neuron in the network calculates its output based on the weighted sum of its inputs and an activation function.
- **Calculate Error:**
- The output of the neural network is compared to the desired output, and the error (or loss) is calculated using a suitable loss function such as mean squared error or cross-entropy.
- **Backward Pass:**
- In the backward pass, the error is propagated backward through the network.
- The gradient of the loss function with respect to the weights of the network is calculated using the chain rule of calculus.

Back Propagation:-

Need for Back propagation:-

- ❑Back propagation is “back propagation of errors” and is very useful for training neural networks.
- ❑It’s fast, easy to implement, and simple.
- ❑Back propagation does not require any parameters to be set, except the number of inputs.
- ❑Back propagation is a flexible method because no prior knowledge of the network is required.

Back Propagation:-

Types of Back propagation:-

- ❑There are two types of back propagation networks.

Static back propagation:

- ❑Static back propagation is a network designed to map static inputs for static outputs.
- ❑These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).

Recurrent back propagation:

- ❑Recursive back propagation is another network used for fixed-point learning.
- ❑Activation in recurrent back propagation is feed-forward until a fixed value is reached.
- ❑Static back propagation provides an instant mapping, while recurrent backpropagation does not provide an instant mapping.

Back Propagation:-

Advantages:-

- ❑It is simple, fast, and easy to program.
- ❑Only numbers of the input are tuned, not any other parameter.
- ❑It is Flexible and efficient.
- ❑No need for users to learn any special functions.

Disadvantages:-

- ❑It is sensitive to noisy data and irregularities. Noisy data can lead to inaccurate results.
- ❑Performance is highly dependent on input data.
- ❑Spending too much time training.
- ❑The matrix-based approach is preferred over a mini-batch.

Advantages of ANNs:-

- ❑Artificial neural networks have the ability to provide the data to be processed in parallel, which means they can handle more than one task at the same time.
- ❑Artificial neural networks have been in resistance. This means that the loss of one or more cells, or neural networks, influences the performance of Artificial Neural networks.
- ❑Artificial neural networks are used to store information on the network so that, even in the absence of a data pair, it does not mean that the network is not generating results.
- ❑Artificial neural networks are gradually being broken down, which means that they will not suddenly stop working and these networks are gradually being broken down.
- ❑We are able to train ANN’s that these networks learn from past events and make decisions.

Disadvantages of ANNs:-

- ❑As we mentioned before, with ANN arms hanging along with the execution of parallel processing, and so they need processors that support parallel processing, so the ANNs are dependent on the hardware.
- ❑Since it's similar to the functionality of the human brain, we may not be able to determine what is the proper network structure of an Artificial Neural network.
- ❑Not only do artificial neural networks, but also the statistical models can be trained with only numeric data, so it makes it very difficult for ANN to understand the problem statement.
- ❑When an artificial neural network that provides a solution to the problem statements that we really don't know on what basis it will give the solution, and this time, ANN is not a reliable.

Applications of ANNs:-

- ❑Image Processing
- ❑Pattern Recognition
- ❑Forecasting
- ❑Stock market prediction
- ❑Social media
- ❑Face Recognition
- ❑Text translation
- ❑Voice recognition