Neural Network I

Overview

Lecture No.	Content	Duration	Self-Study
1	What is Neural Network, Learning rules.	1	1
2	Various activation functions	1	1
3	Single layer Perceptron	1	1
4	Back Propagation networks	1	1
5	Character Recognition Application	1	1
6	Unsupervised Learning Network	1	1
7	Counter propagation network	1	1
8	Adaptive reasoning theory	1	1
9	Associative Memory.	1	1

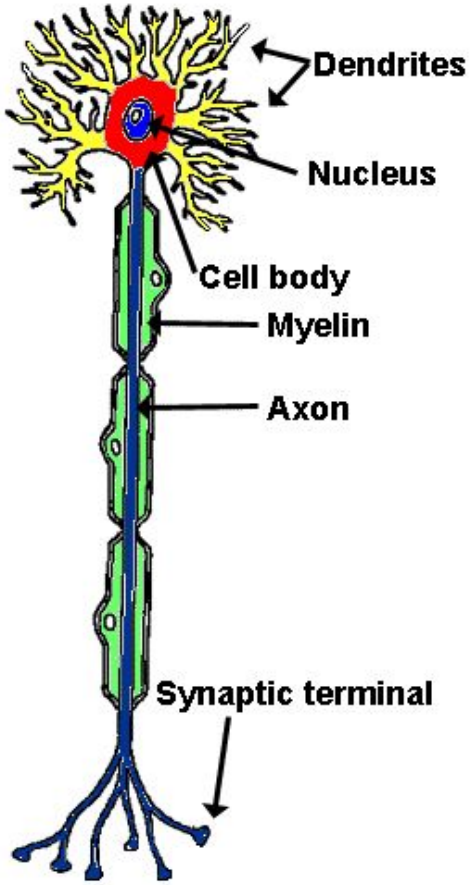
Introduction

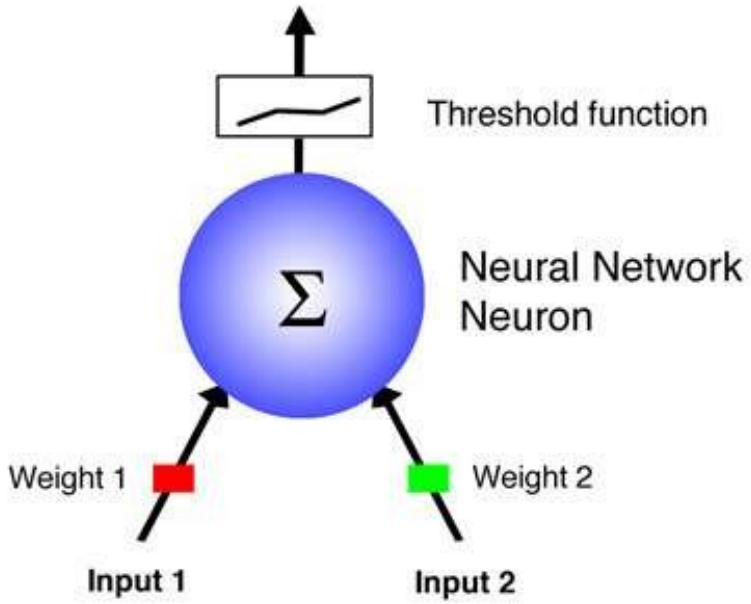
- **What is Neural Network??**
- A method of computing, based on the interaction of multiple connected processing elements.
- A powerful technique to **solve many real world problems**.
- The ability to **learn from experience** in order to improve their performance.
- At the core of a neural network is **a mathematical model** that is used to make predictions or decisions based on input data**.**
- The neurons in a neural network are **connected by weighted links** that allow them to communicate with one another.
- There are several types of neural networks, including **feedforward neural networks, convolutional neural networks, and recurrent neural networks**.

Basics of Neural Network

- A neuron is a cell that carries **electrical impulses and are the basic units** of the nervous system.
- Every neuron is made of a cell body (also called a soma), dendrites and an axon. Dendrites and axons are nerve fibers. There are about **86 billion neurons in the human brain**, which **comprises roughly 10% of all brain cells**.
- Neurons are connected to one another and tissues. They do not touch and instead form tiny gaps called **synapses**. These gaps can be chemical synapses or electrical synapses and pass the signal from one neuron to the next.
- **Dendrite** — It receives signals from other neurons.
- **Soma (cell body)** — It sums all the incoming signals to generate input.
- **Axon** — When the sum reaches a threshold value, neuron fires and the signal travels down the axon to the other neurons.
- **Synapses** — The point of interconnection of one neuron with other neurons. The amount of signal transmitted depend upon the strength (synaptic weights) of the connections.

Neurons

- 
Dendrites
Nucleus
Cell body
Myelin
Axon
Synaptic terminal


Threshold function
Σ
Neural Network
Neuron
Weight 1
Weight 2
Input 1
Input 2

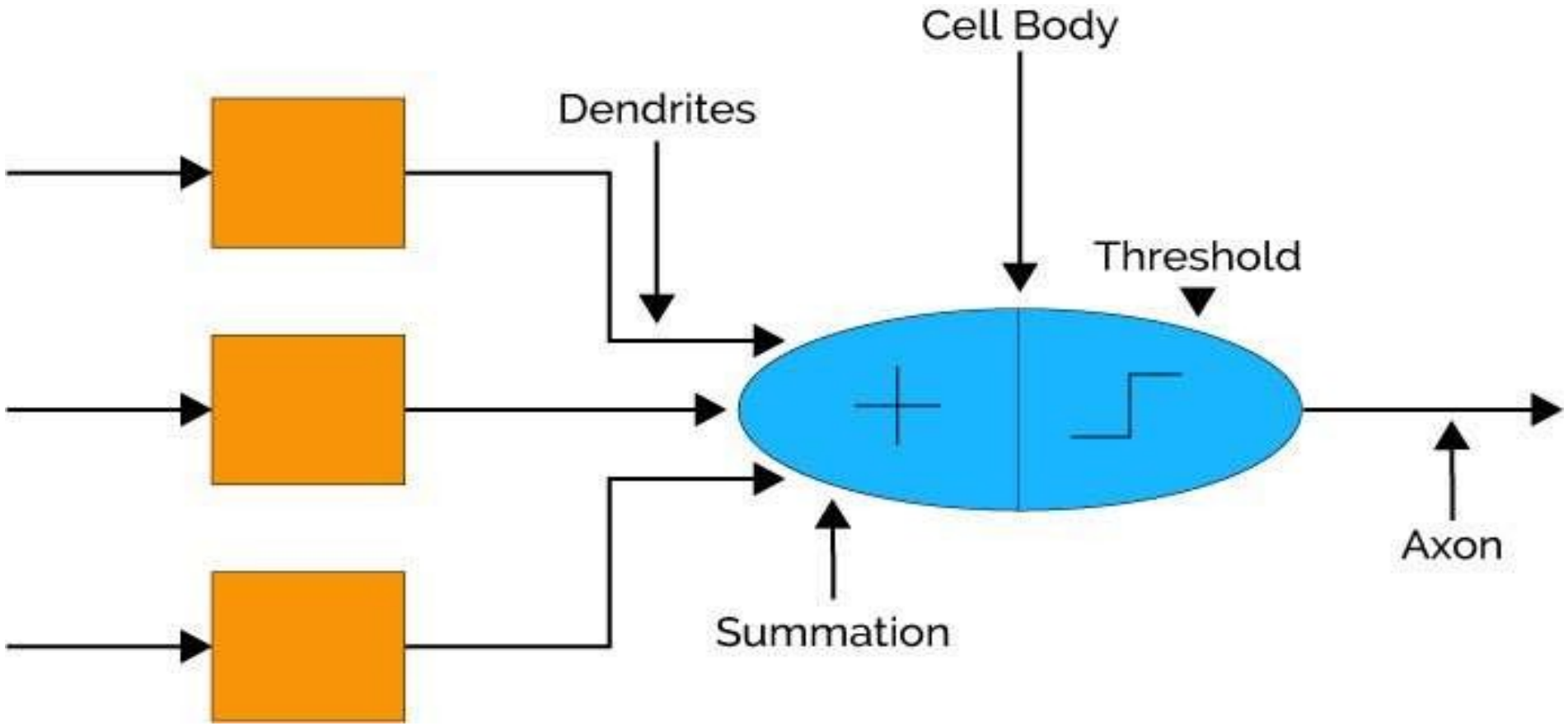
Comparing ANN and BNN

- As this concept borrowed from ANN there are lot of similarities though there are differences too.
- Similarities are in the following table

Biological Neural Network	Artificial Neural Network
Soma	Node
Dendrites	Input
Synapse	Weights or Interconnections
Axon	Output

Criteria	BNN	ANN
Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
Size	1011 neurons and 1015 interconnections	102 to 104 nodes (mainly depends on the type of application and network designer)
Learning	They can tolerate ambiguity	Very precise, structured and formatted data is required to tolerate ambiguity
Fault tolerance	Performance degrades with even partial damage	It is capable of robust performance, hence has the potential to be fault tolerant
Storage capacity	Stores the information in the synapse	Stores the information in continuous memory locations

Analogy of ANN with BNN



Learning

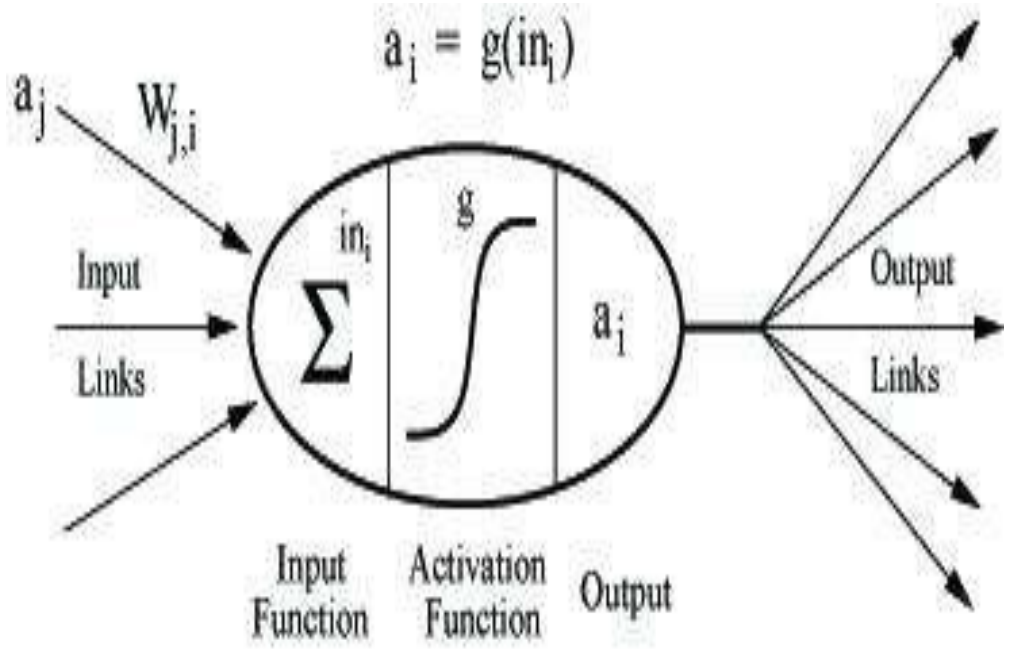
- Learning = learning by adaptation
- The objective of learning in biological organisms is to improve their survival and reproductive success by adapting to changing environmental conditions and developing new strategies for survival.
- Learning in biological organisms allows them to:
    1. Respond to environmental changes
    2. Improve their performance
    3. Develop new behaviors
    4. Enhance communication

Types of Learning in Neural Network

- **Supervised Learning** — Supervised learning is a type of machine learning where the algorithm **is trained on labeled data**, which means that the data is already categorized into specific classes or categories.
- **Unsupervised Learning** — Unsupervised learning is a type of machine learning where **the algorithm is trained on unlabeled data**, which means that the data is not categorized into specific classes or categories. The goal of unsupervised learning is to find patterns and relationships in the data without any prior knowledge of what the data represent.
- **Reinforcement Learning** — Reinforcement learning is a type of machine learning where an **agent learns to make decisions in an environment by receiving feedback in the form of rewards or penalties**.

Model of Artificial Neural Network

- Receives n-inputs
- Multiplies each input by its weight
- Applies activation function to the sum of results
- Outputs result

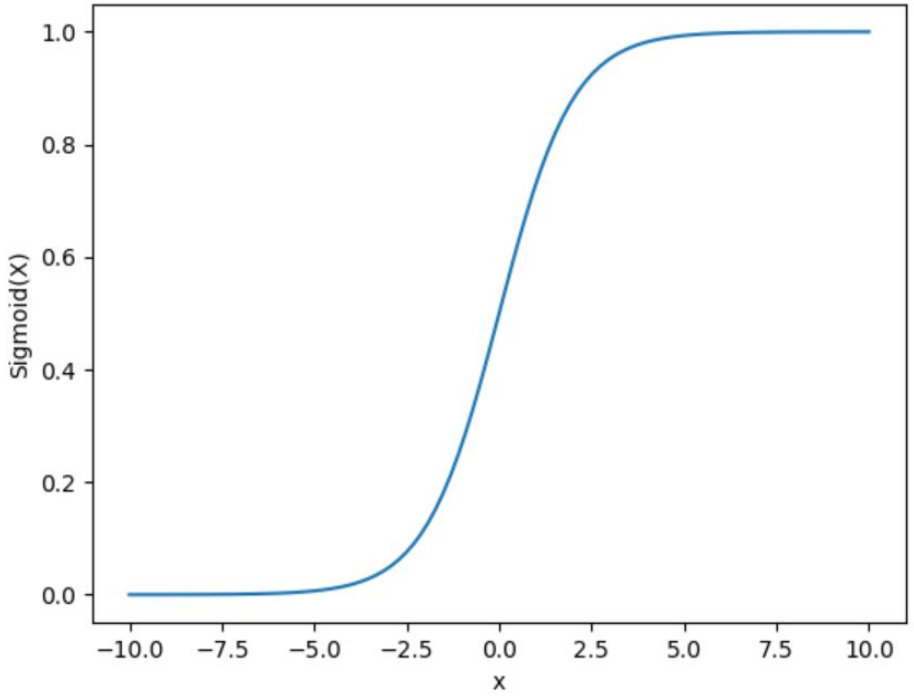


Activation Functions

- Activation functions in a neural network (NN) are mathematical functions that are applied to the output of a neuron in the network.
- The activation function introduces non-linearity into the network and helps to produce a non-linear decision boundary that can be used to model complex relationships in the input data.
- Some commonly used activation functions in NNs include:
- **Sigmoid function:** The sigmoid function is an S-shaped curve that maps any input value to a value between 0 and 1. It is commonly used as the activation function in the output layer of binary classification problems.
- **ReLU (Rectified Linear Unit) function**: The ReLU function maps any input value to 0 if it is negative, and to the input value if it is positive. It is commonly used as the activation function in the hidden layers of deep neural networks.
- **Tanh (Hyperbolic tangent) function**: The Tanh function is similar to the sigmoid function, but it maps any input value to a value between -1 and 1. It is also commonly used as an activation function in the hidden layers of neural networks.
- **Softmax function**: The softmax function is used in the output layer of multi-class classification problems. It maps the output values of each neuron to a probability distribution over the classes.

Sigmoid Function

- It is a function which is plotted as **'S'** shaped graph.
- **Equation :** $A = 1/(1 + e^{-x})$
- **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
- **Value Range :** 0 to 1
- **Uses :** Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

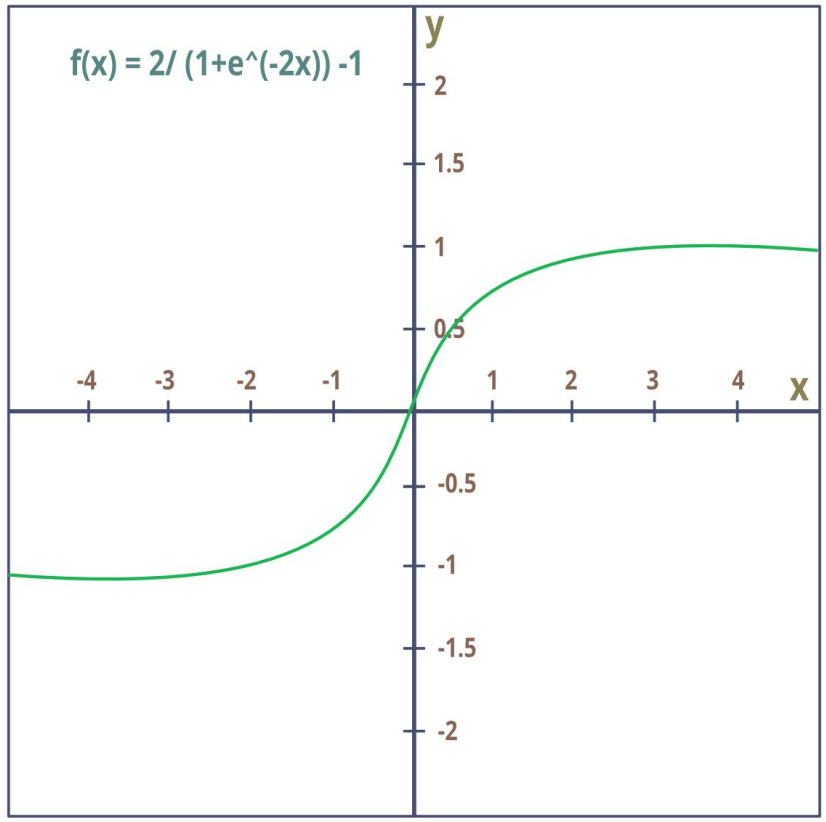


Tanh Function

- The activation that works almost always better than sigmoid function is Tanh function also known as **Tangent Hyperbolic function**. It's actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.
- **Equation :**

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

- **Value Range :-** -1 to +1
- **Nature :-** non-linear
- **Uses :-** Usually used in hidden layers of a neural network as it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.



RELU Function

•It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.

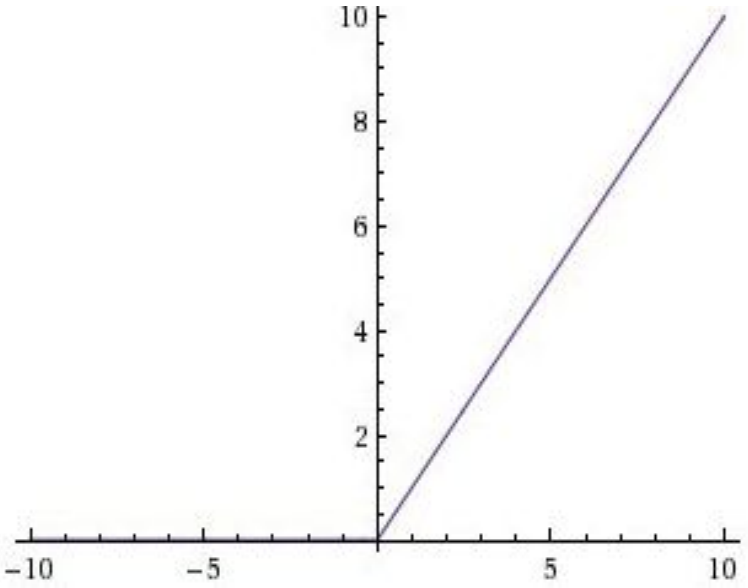
•**Equation :-** ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.

•**Value Range :-** [0, inf)

•**Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.

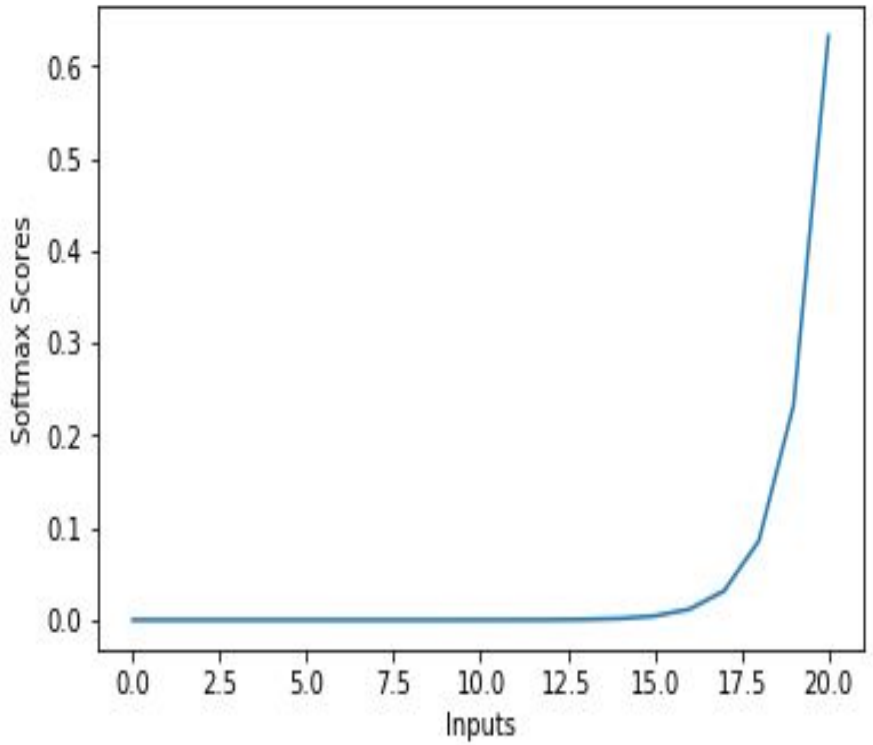
•**Uses :-** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

In simple words, RELU learns *much faster* than sigmoid and Tanh function.



Softmax Function

- The softmax activation function is commonly used in the output layer of a neural network when performing multiclass classification.
- **Nature :-** non-linear
- **Uses :-** Usually used when trying to handle multiple classes. The softmax function was commonly found in the output layer of image classification problems.
- The softmax function is particularly useful in multiclass classification tasks, where the goal is to predict the probability of each possible class for a given input.



Activation function

- The basic rule of thumb is if you really don't know what activation function to use, then simply use *RELU* as it is a general activation function in hidden layers and is used in most cases these days.
- If your output is for binary classification then, *sigmoid function* is very natural choice for output layer.
- If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each classes.

Single Layer Perceptron

A single-layer perceptron is a type of artificial neural network that consists of only one layer of artificial neurons.

It is the simplest type of neural network and was proposed by Frank Rosenblatt in 1958.

Single layer perceptron has been used in various applications, including: Pattern Recognition, Binary Classification, , Control Systems, Medical Diagnosis, Financial Forecasting

The perceptron consists of 4 parts:

Input value or One input layer: The input layer of the perceptron is made of artificial input neurons and takes the initial data into the system for further processing.

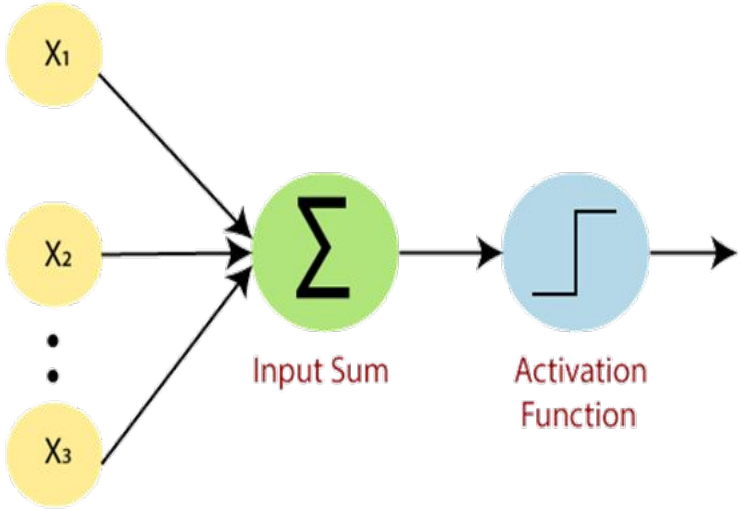
Weights and Bias:

Weight: It represents the dimension or strength of the connection between units.

Bias: It is the same as the intercept added in a linear equation. bias is a tunable parameter in neural networks that can help improve the accuracy and flexibility of the model by allowing it to learn more complex decision boundaries.

Net sum: It calculates the total sum.

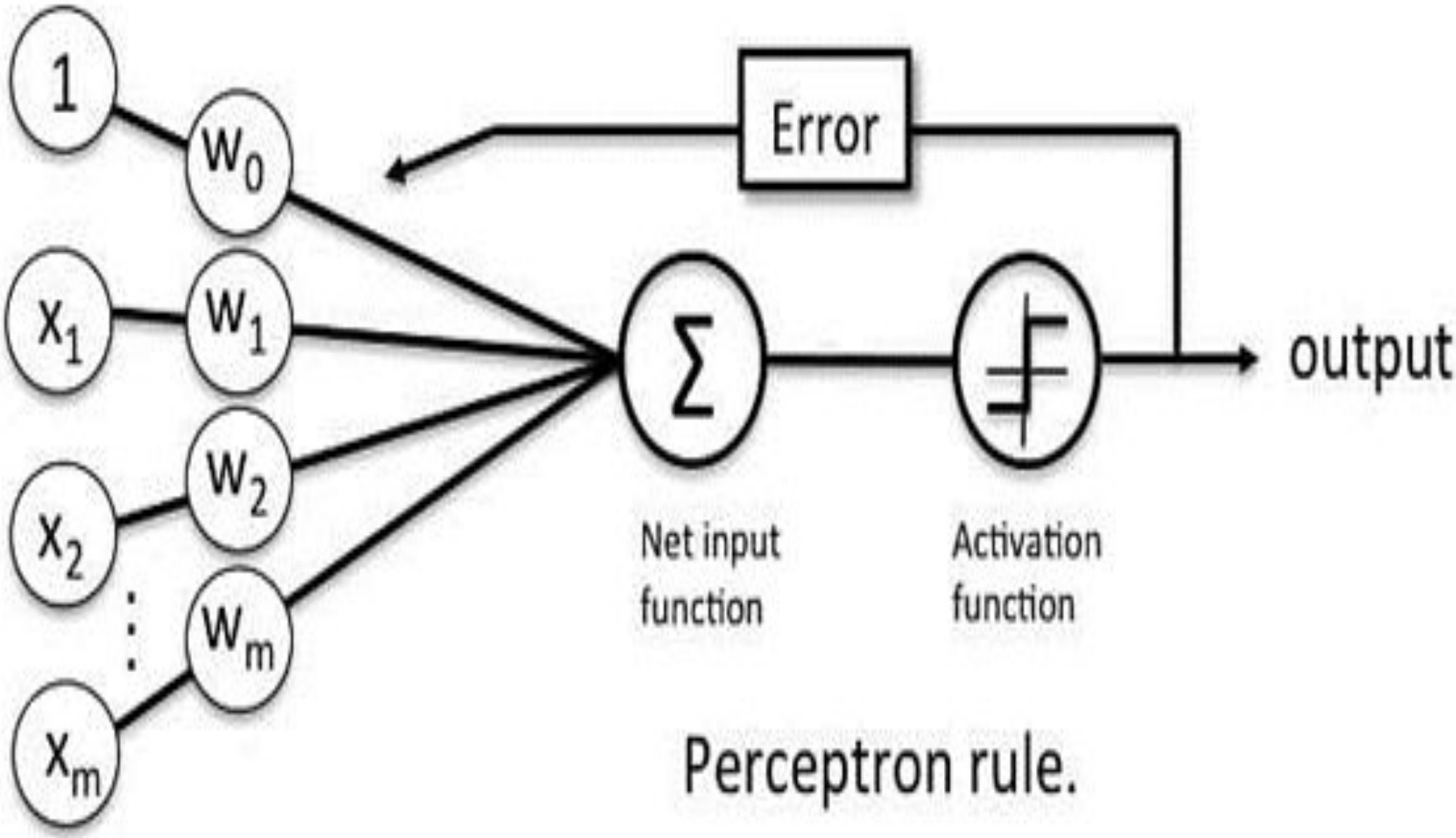
Activation Function: A neuron can be activated or not, is determined by an activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.



The Perceptron Learning Rule

1. Initialize the weights: Start with random weights for each input.
2. Input the training data: Input the features into the perceptron and calculate the output.
3. Calculate the error: Compare the predicted output with the desired output to calculate the error.
4. Update the weights: Adjust the weights of the inputs based on the error. If the predicted output is less than the desired output, increase the weights of the inputs. If the predicted output is greater than the desired output, decrease the weights of the inputs. The magnitude of the weight adjustment is proportional to the error and the input value.

Repeat: Repeat steps 2 to 4 until the error is minimized or a maximum number of iterations is reached



Perceptron rule.

Perceptron Function

Perceptron is a function that maps its input "x," which is multiplied with the learned weight coefficient; an output value" f(x)"is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the equation given above:

"w" = vector of real-valued weights

"b" = bias (an element that adjusts the boundary away from origin without any dependence on the input value)

"x" = vector of input x values

$$\sum_{i=1}^{m} w_i x_i$$

"m" = number of inputs to the Perceptron

The output can be represented as "1" or "0." It can also be represented as "1" or "-1" depending on which activation function is used

Activation Functions of Perceptron

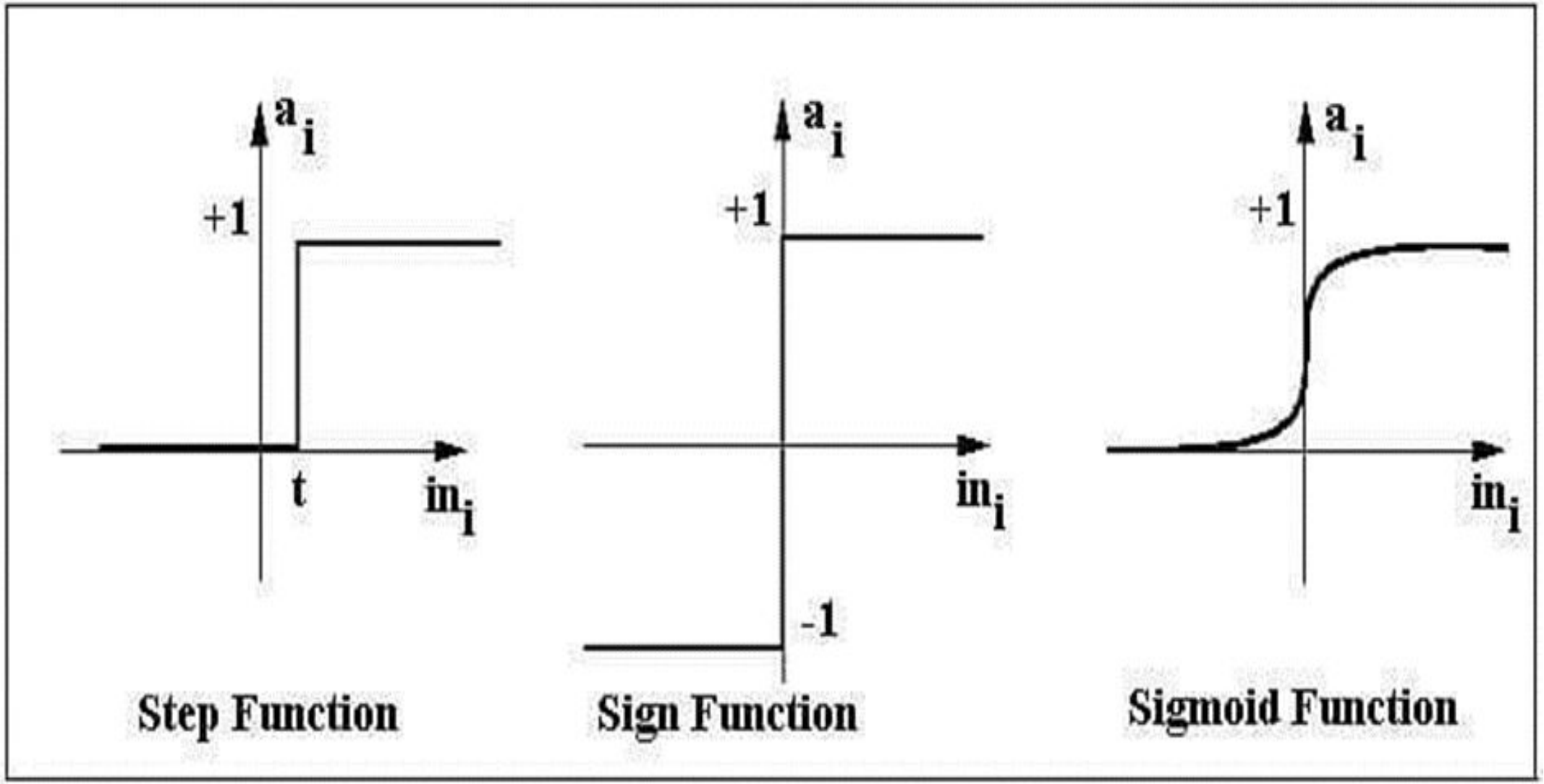
The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.

For example:

If $\sum$ wixi> 0 => then final output “o” = 1 (issue bank loan)

Else, final output “o” = -1 (deny bank loan)

Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.


a_i
+1
t
in_i
Step Function
a_i
+1
in_i
-1
Sign Function
a_i
+1
in_i
Sigmoid Function

A Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron (MLP) is a type of neural network that consists of multiple layers of artificial neurons.

MLPs are also known as feedforward neural networks.

The architecture of an MLP consists of an input layer, one or more hidden layers, and an output layer.

Each layer is composed of multiple artificial neurons that compute a weighted sum of the input signals and apply an activation function to produce an output signal.

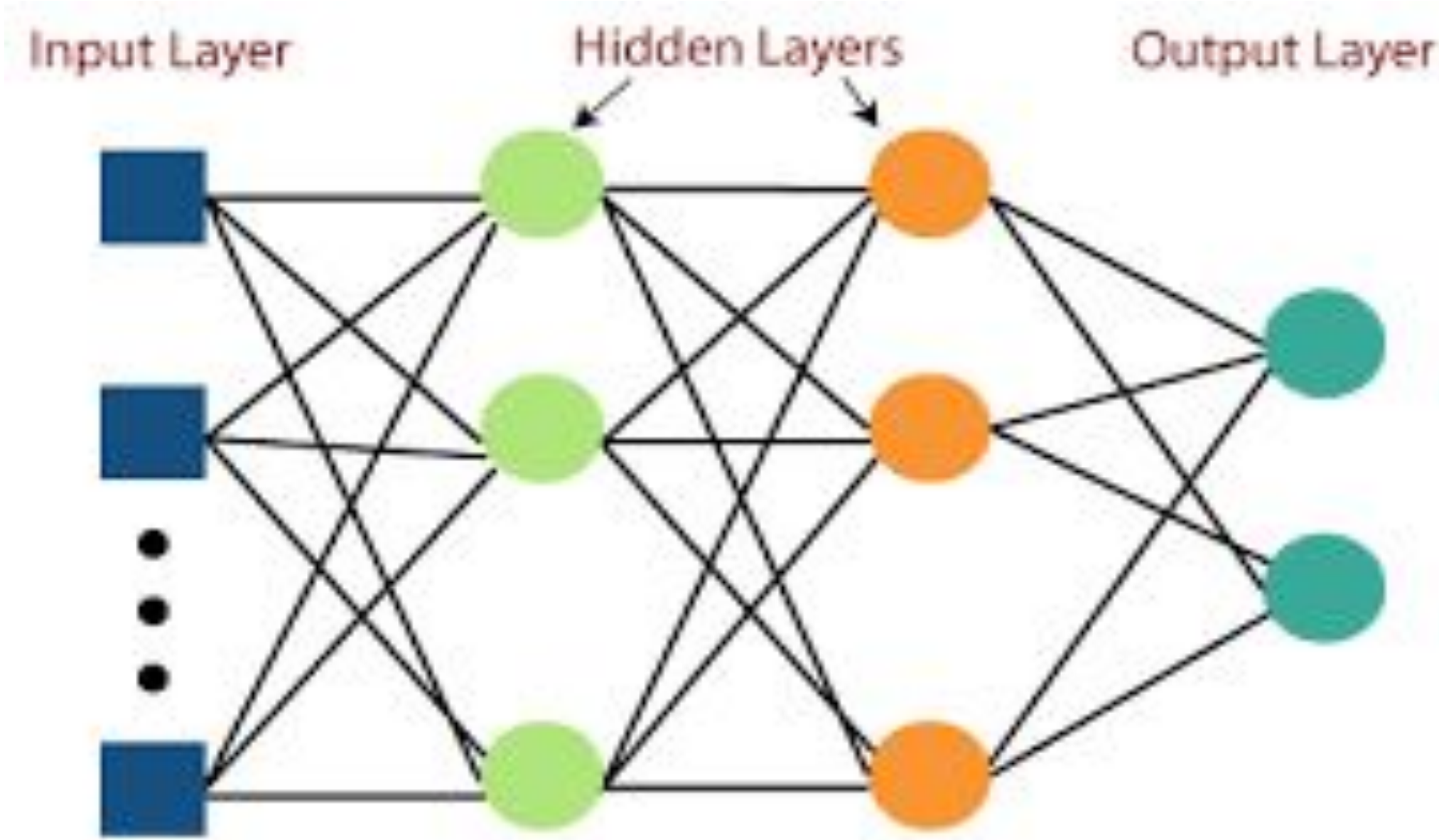
A Multi-Layer Perceptron (MLP)

The hidden layers in an MLP are responsible for extracting features from the input data and transforming them into a format that is suitable for the output layer.

The output layer produces the final output of the network, which can be binary or continuous.

The learning process of an MLP involves adjusting the weights of the input signals using backpropagation.

Backpropagation allows the MLP to learn from the training data and improve its performance over time.


Input Layer
Hidden Layers
Output Layer

Compare single layer and multilayer perceptron model

Architecture:

Single-layer perceptrons have only one layer of neurons that directly connects to the input data, whereas multilayer perceptrons consist of multiple layers of neurons, including one or more hidden layers that lie between the input and output layers.

Capabilities:

Single-layer perceptrons are limited to linearly separable problems, meaning they can only learn and classify data that can be separated by a single straight line.
In contrast, multilayer perceptrons can learn and classify non-linearly separable problems by using hidden layers to transform the input data into a more complex feature space that can be separated by the output layer.

Compare single layer and multilayer perceptron model

Training:

Single-layer perceptrons use a simple learning rule called the Perceptron Learning Algorithm, which adjusts the weights of the input signals to minimize the error between the predicted and actual output. In contrast, multilayer perceptrons use a more complex learning algorithm called backpropagation, which iteratively adjusts the weights of all the neurons in the network to minimize the error between the predicted and actual output.

Applications:

Single-layer perceptrons are typically used for simple binary classification problems, such as predicting whether an email is spam or not. Multilayer perceptrons are more powerful and can be used for a wide range of applications, including image and speech recognition, natural language processing, and financial forecasting.

Delta Learning Rule

The Delta learning rule is a supervised learning algorithm used for updating the weights of a neural network during the training process.

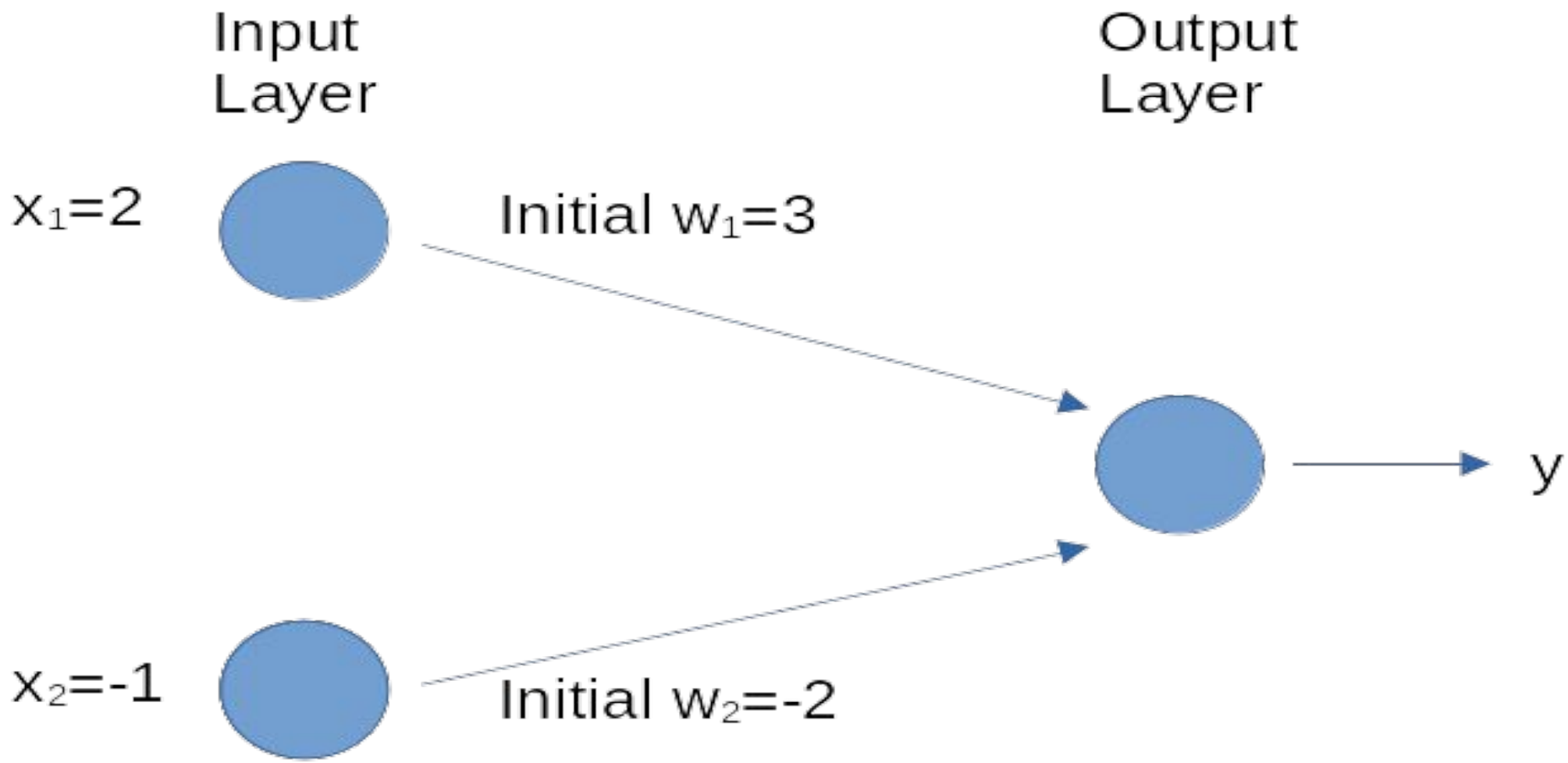
The Delta learning rule is used to minimize the error between the predicted output of the network and the actual output. It does this by adjusting the weights of the network in the direction that reduces the error.

The Delta learning rule works as follows:

1. The network receives an input vector, which is multiplied by the weights to produce the output.
2. The output is compared to the actual output to calculate the error.
3. The weights are adjusted based on the error and the learning rate. The learning rate is a parameter that determines the size of the weight update.
4. The process is repeated for each input vector in the training set.

In Mathematical form the delta rule is as follows:

$$\Delta w = \eta (t - y) x_i$$


Input Layer
Output Layer
$x_1=2$
Initial $w_1=3$
$x_2=-1$
Initial $w_2=-2$
y

Back Propagation networks

Back Propagation are supervised learning algorithms used for training neural networks.

The basic structure of a backpropagation network consists of an input layer, one or more hidden layers, and an output layer.

Each layer is composed of one or more neurons, which receive inputs, process them, and pass the outputs to the next layer.

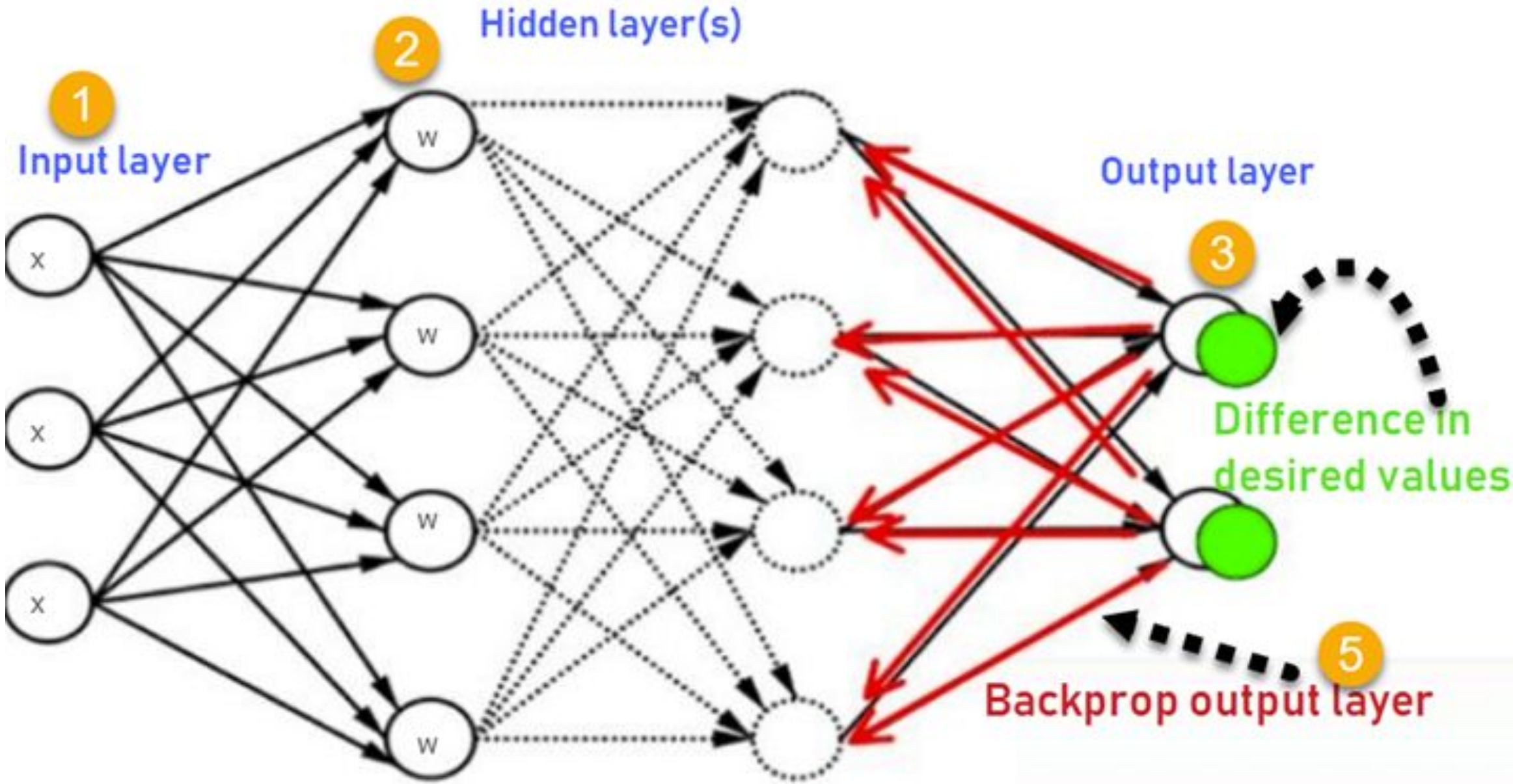
The connections between the neurons are weighted, and these weights are adjusted during training to improve the accuracy of the network's predictions.

Back Propagation networks

During the training process, the network is fed a set of input-output pairs, and the output of the network is compared to the desired output.

The error between the actual output and the desired output is then back propagated through the network, and the weights are adjusted to reduce the error.

This process is repeated many times, with the hope that the network will eventually converge to a set of weights that produces accurate predictions for new input data.


1
Input layer
2
Hidden layer(s)
Output layer
3
x
x
x
w
w
w
w
Difference in
desired values
5
Backprop output layer

How Backpropagation Algorithm Works:

1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

$Error_B$ = Actual Output – Desired Output

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Why We Need Backpropagation?

- Backpropagation is **fast, simple and easy to program**
- It has **no parameters to tune** apart from the numbers of input
- It is a flexible method as it **does not require prior knowledge about the network**
- It is a standard method that generally works well
- It **does not need any special mention of the features** of the function to be learned.

Character Recognition Application

Character recognition is a common application of neural networks, and **can be achieved using various types of neural networks**, including Feedforward neural networks, convolutional neural networks, and recurrent neural networks

The network must be **trained on a dataset of labeled character** images in order to learn to recognize characters.

During training, **the network adjusts its weights based on the difference between its predicted output and the true label of the input image**.

Once the network is trained, it can be used to make predictions on new, unlabeled character images.

OCR (Optical Character Recognition)

OCR is a technology that **analyzes the text of a page and turns the letters into code** that may be used to process information.

OCR is a technique for **detecting printed or handwritten text characters** inside digital images of paper files, such as scanning paper records

OCR systems are hardware and software systems that **turn physical documents into machine-readable text.**

These digital versions can be highly beneficial to children and young adults who struggle to read.

The essential application of OCR is to convert hard copy legal or historical documents into PDFs.

How OCR works?

1. Image Pre-Processing

Pre-processing covers all those functions of feature extraction to produce a original image.

The steps in pre-processing involves

Size normalization: Bicubic interpolation is used for standard sized image.

Binarization: it is process of converting a gray scale image into binary image by thresholding

Smoothing: the erosion and dilation smooth the Boundaries of objects.

How OCR works?

- **Text recognition**
- The two main types of OCR algorithms or software processes that an OCR software uses for text recognition are called pattern matching and feature extraction.
- **Pattern matching**
- This method works well with scanned images of documents that have been typed in a known font.
- **Feature extraction**
- Feature extraction of character.

How OCR works?

Postprocessing

After analysis, the system converts the extracted text data into a computerized file. Some OCR systems can create annotated PDF files that include both the before and after versions of the scanned document.

Kohonen's self-organizing map

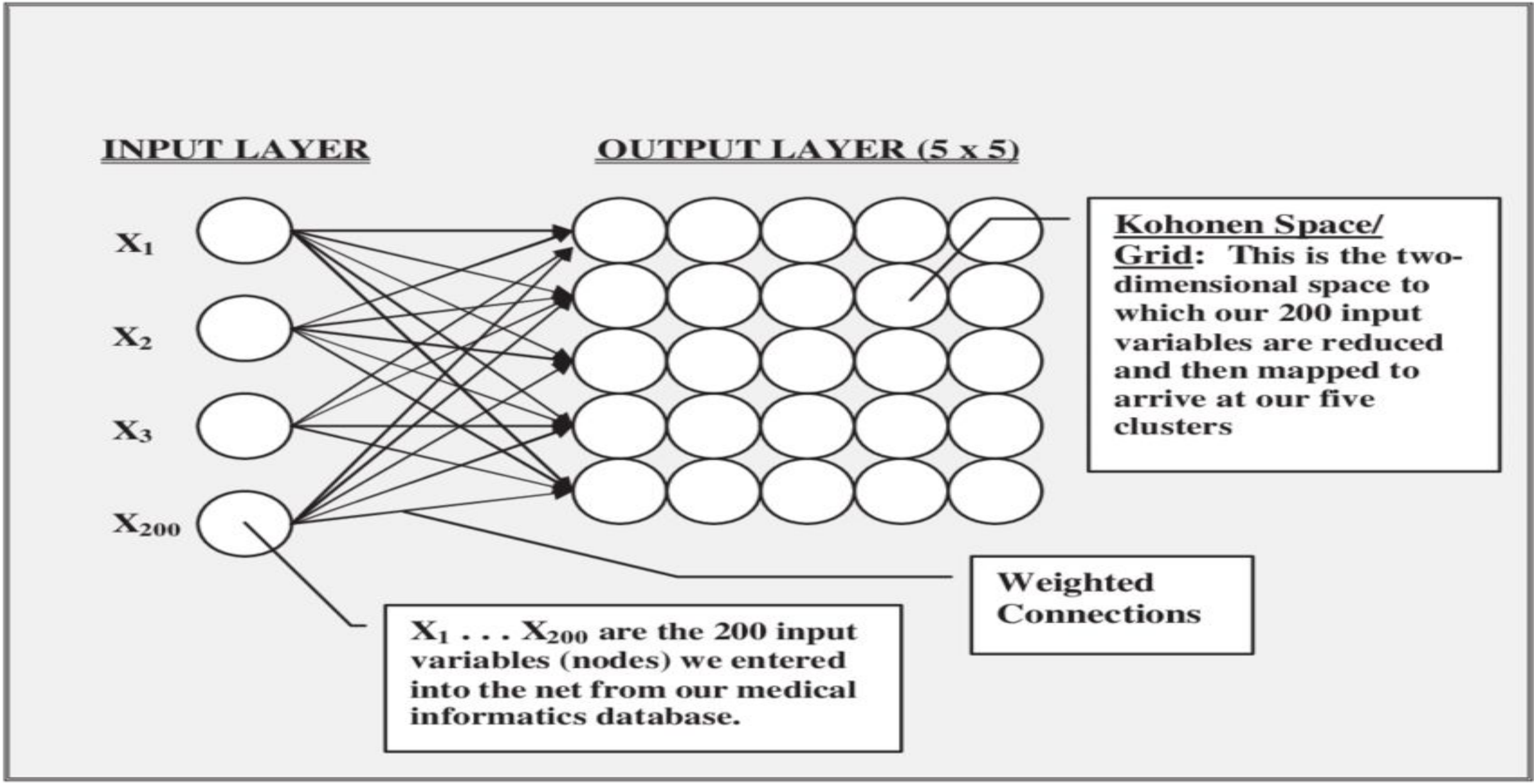
1. It is a type of unsupervised neural network that is used for clustering and visualization of high-dimensional data.
2. Self Organising Map: A self organising map (SOM) **is an unsupervised neural network** that reduces the input dimensionality in order to represent its distribution as a map.
3. SOMs can be used for various applications, such as data visualization, feature extraction, and clustering.
4. They have been applied in fields such as image processing, speech recognition, and bioinformatics.

Kohonen's self-organizing map

Self Organising map has two layers one is input layer and the output layer.

Unlike other ANN types, SOM doesn’t have activation function in neurons, we directly pass weights to output layer without doing anything.

A self organising map (SOM) is a types of ANN that is tarined using unsupervised learning


INPUT LAYER
OUTPUT LAYER (5 x 5)
X1
X2
X3
X200
Kohonen Space/ Grid: This is the two-dimensional space to which our 200 input variables are reduced and then mapped to arrive at our five clusters
Weighted Connections
X1 . . . X200 are the 200 input variables (nodes) we entered into the net from our medical informatics database.

Kohonen's Algorithm

Step:1

Initialize the weight w_ij initialize to a random value.

Step:2

Calculate the Euclidean distance between weight vector w_{ij} and the input vector x(t) connected with the first node, where t, i, j =0.

$D(j) = \Sigma (wij - xi)^2$ where i=1 to n and j=1 to m

Kohonen's Algorithm

Step:3

Find the winning index “j” of the distance calculated.

Step:4

For all units “j” within a specific neighborhood of j and for all i calculate new weights

$$wij(new)=wij(old) + \alpha[xi - wij(old)]$$

Kohonen's Algorithm

Step:5

Update the learning rule by using :

$$\alpha(t+1) = 0.5 * t$$

Step:6

Test the Stopping Condition.

Counter propagation network

Counter propagation Network (CPN) is an artificial neural network architecture that **combines both supervised and unsupervised learning**.

It was proposed by Professor Teuvo Kohonen in the 1980s.

The working of CPN network consists of two layers of neurons: **a competitive layer and a linear output layer**.

During the training phase, **the competitive layer learns to recognize and classify the patterns in the input data**, while the l**inear output layer learns to map these patterns to a desired output**.

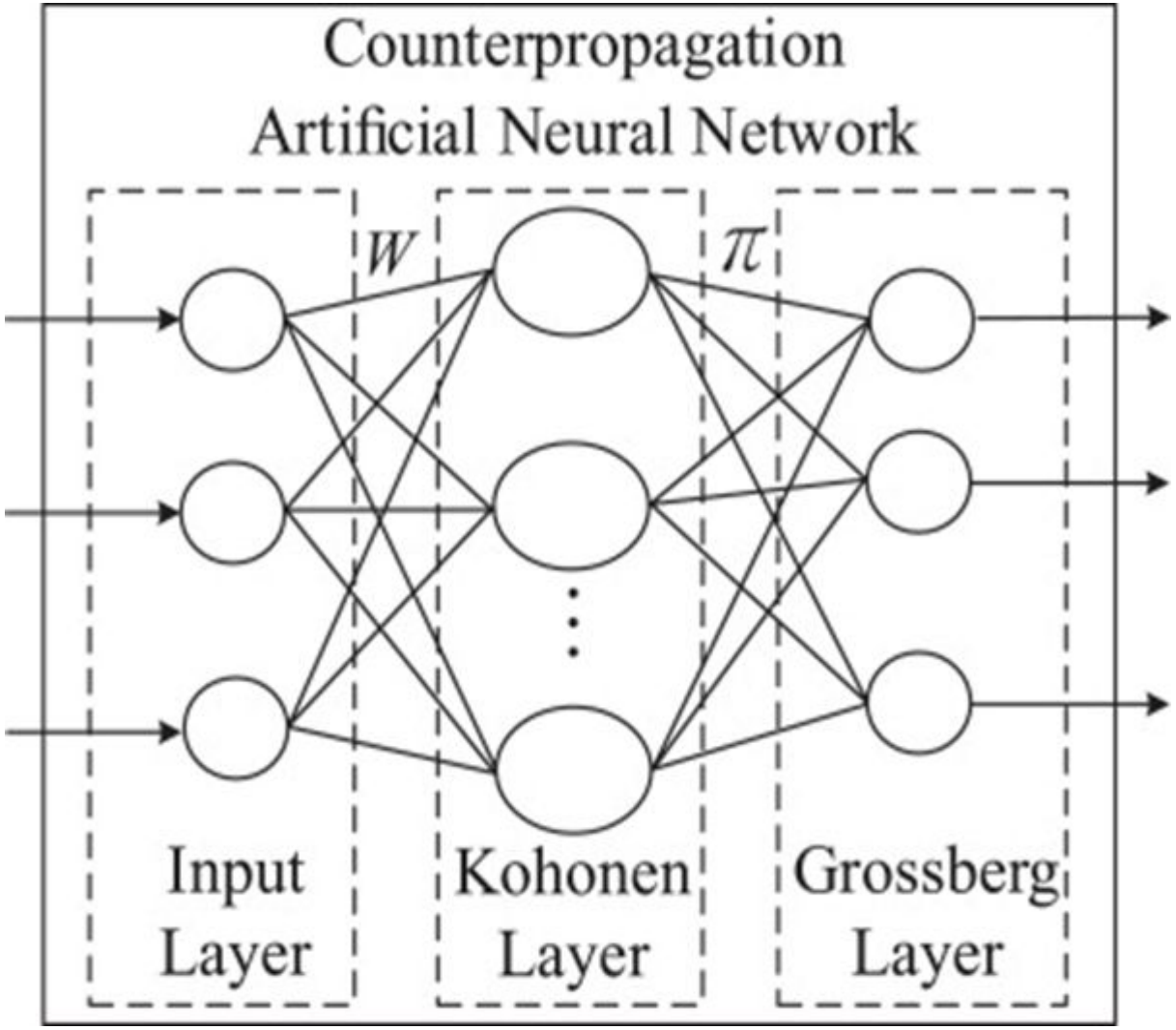
The learning process in CPN consists of **two phases**. In the first phase, called the **unsupervised phase**, the competitive layer **learns to recognize and cluster the patterns in the input data** without any supervision.

In the second phase, called the **supervised phase**, the linear output layer is trained using supervised learning to map the input patterns to a desired output.

Counter propagation network

When a new input pattern is presented to the network, it is f**irst mapped to the competitive layer,** which identifies the closest neuron to the input pattern. **The output layer then maps the identified neuron to the desired output value**.

CPN has been used in a variety of applications, including **pattern recognition, signal processing, and data compression**.

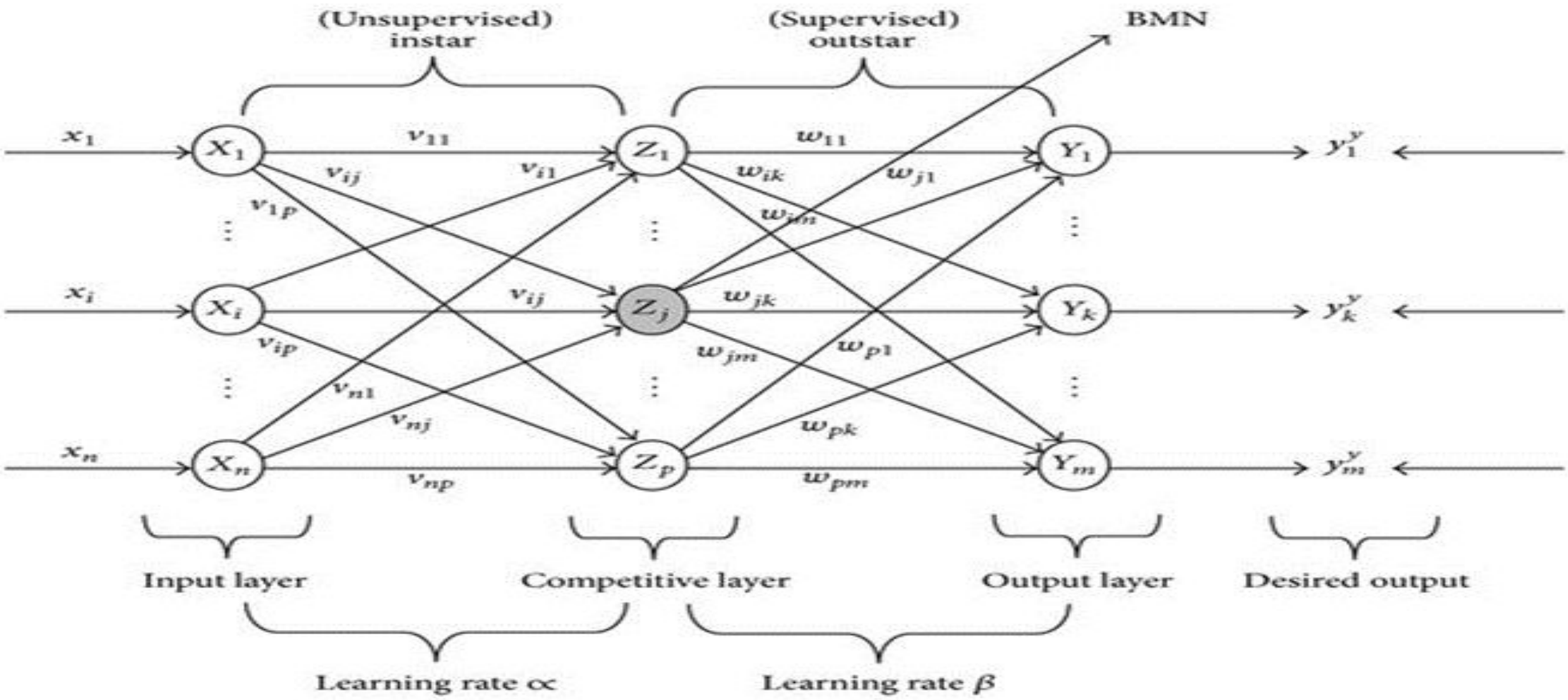


Types of Counter propagation network

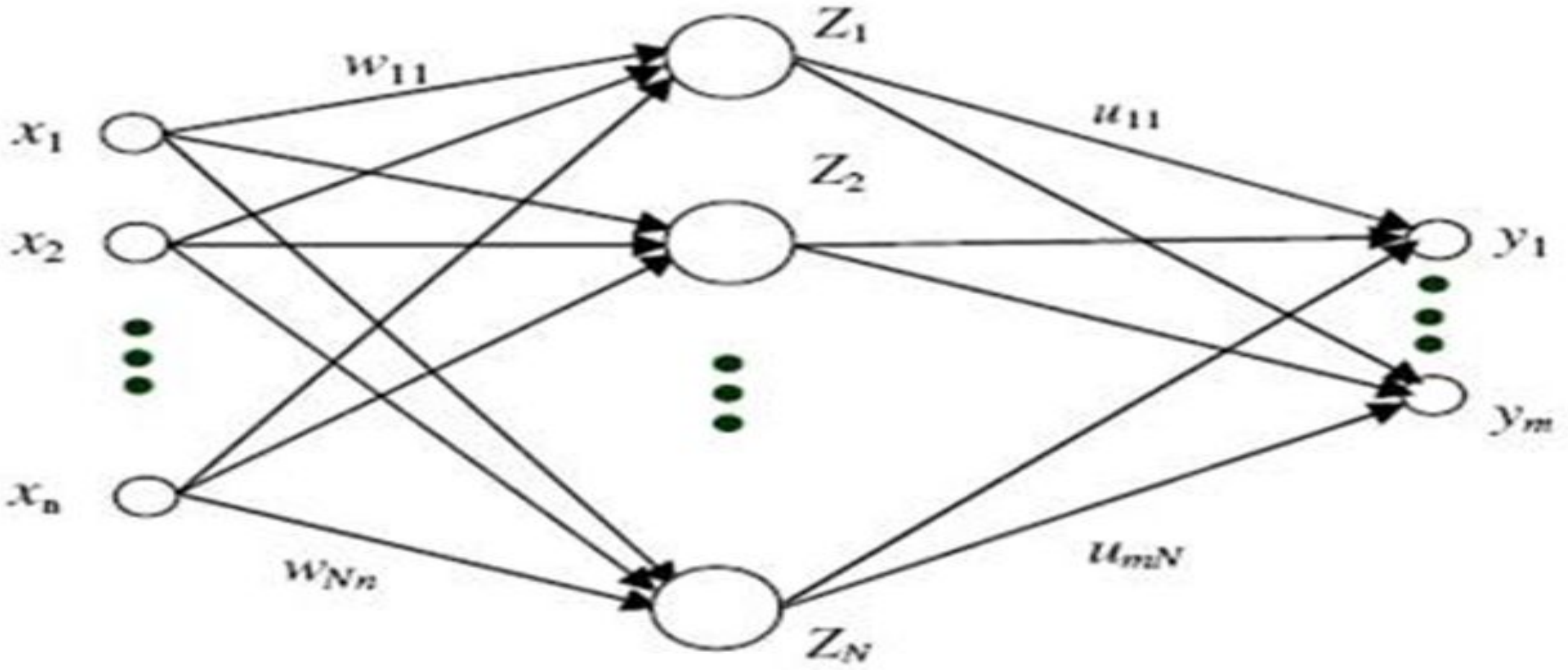
Full counterpropagation network: It is composed of two layers: a Kohonen layer (also known as a self-organizing layer) and a Grossberg layer.

Forward-only Counter propagation network: The FOCP network consists of two layers: a Kohonen layer and a Grossberg layer, similar to the full CPN. However, in the FOCP network, the weights in both layers are updated only during the forward pass of the network, using a simple and computationally efficient algorithm

Full counterpropagation network



Forward-only Counter propagation network



Adaptive Resonance Theory

The term **“adaptive”** and **“resonance”** used in this suggests that they are open to new learning (i.e. adaptive) without discarding the previous or the old information (i.e. resonance).

The ART networks are known to solve the **stability-plasticity dilemma** i.e., stability refers to their nature of memorizing the learning and plasticity refers to the fact that they are flexible to gain new information.

Input is presented to the network and the a**lgorithm checks whether it fits into one of the already stored clusters**.

If it fits then the **input is added to the cluster that matches the most** else a new cluster is formed.

Types of Adaptive Resonance Theory

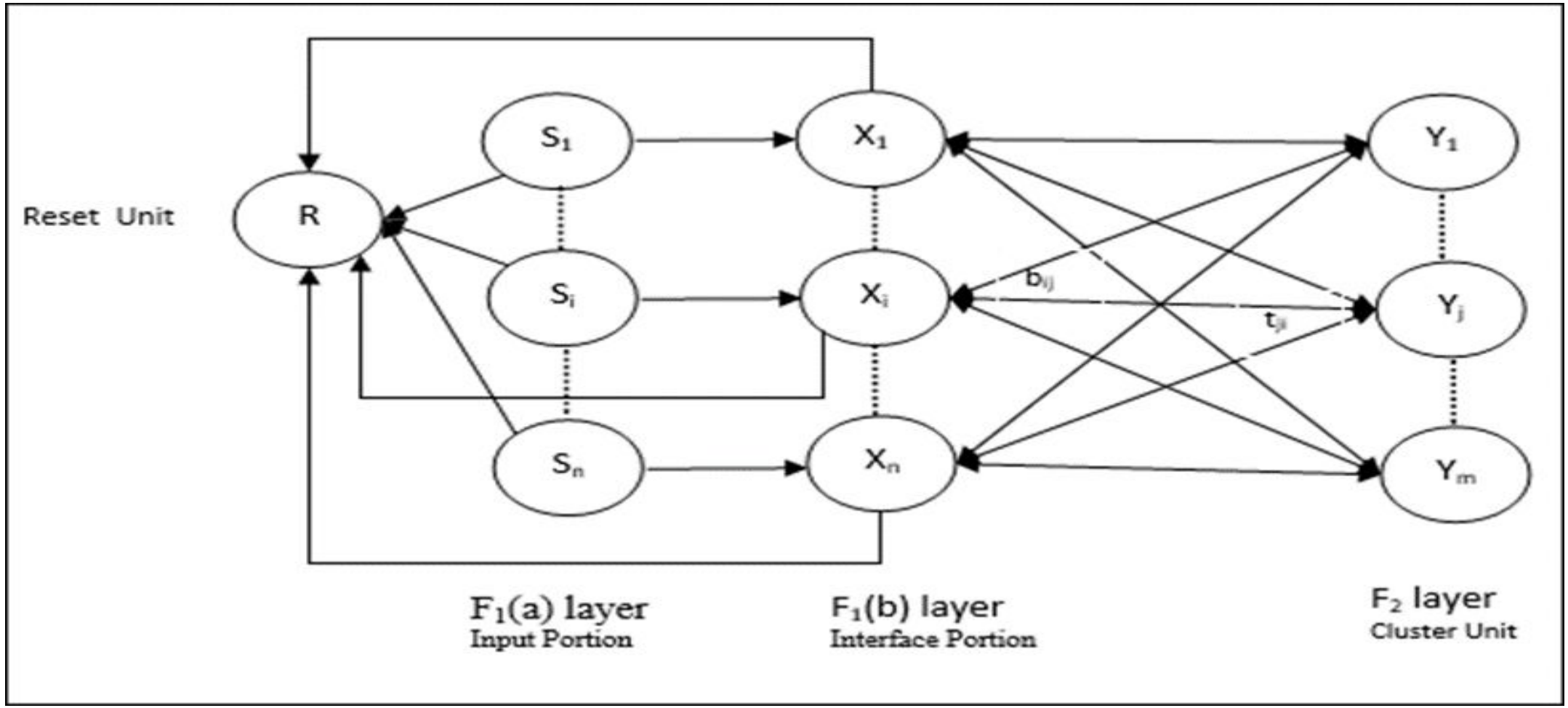
ART1 – It is the simplest and the basic ART architecture. It is capable of **clustering binary input values**.

ART2 – It is extension of ART1 that is capable of c**lustering continuous-valued input data**.

Fuzzy ART – It is the augmentation of **fuzzy logic and ART**.

ARTMAP – It is a supervised form of ART learning where **one ART learns based on the previous ART module**. It is also known as predictive ART.

How Adaptive Resonance Theory (ART) Works



Associative Memory Network

An associate memory network refers to a **content addressable memory structure** that associates a relationship between the set of input patterns and output patterns.

A content addressable memory structure is a kind of memory structure that **enables the recollection of data based on the intensity of similarity between the input pattern and the patterns stored in the memory**.

Following are the two types of associative memories we can observe:

- Auto Associative Memory
- Hetero Associative memory

Auto Associative Memory

Autoassociative memory, also known as auto-association memory or an autoassociative network, is any type of memory that is **able to retrieve a piece of data from only a tiny sample of itself.**

Autoassociative memories are capable of **retrieving a piece of data upon presentation of only partial information from that piece of data**.

For example, the sentence fragments presented below are sufficient for most English-speaking adult humans to recall the missing information.

"To be or not to be, that is _____."

"I came, I saw, _____."

Many readers will realize the missing information is in fact:

"To be or not to be, that is the question."

"I came, I saw, I conquered."

Hetero Associative memory

Heteroassociative memories, on the other hand, **can recall an associated piece of data from one category upon presentation of data from another category**.

For example: It is possible that the associative recall is a transformation from the pattern “banana” to the different pattern “monkey.”

A neural network model that performs such a transformation from a pattern to a different pattern is referred to as heteroassociative memory

Auto Associative Memory

Has **'n' number of input training vectors and similar 'n' number of output target vectors**

Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as $\mathbf{w_{ij} = 0}$ i=1 to n, j=1 to n

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows −

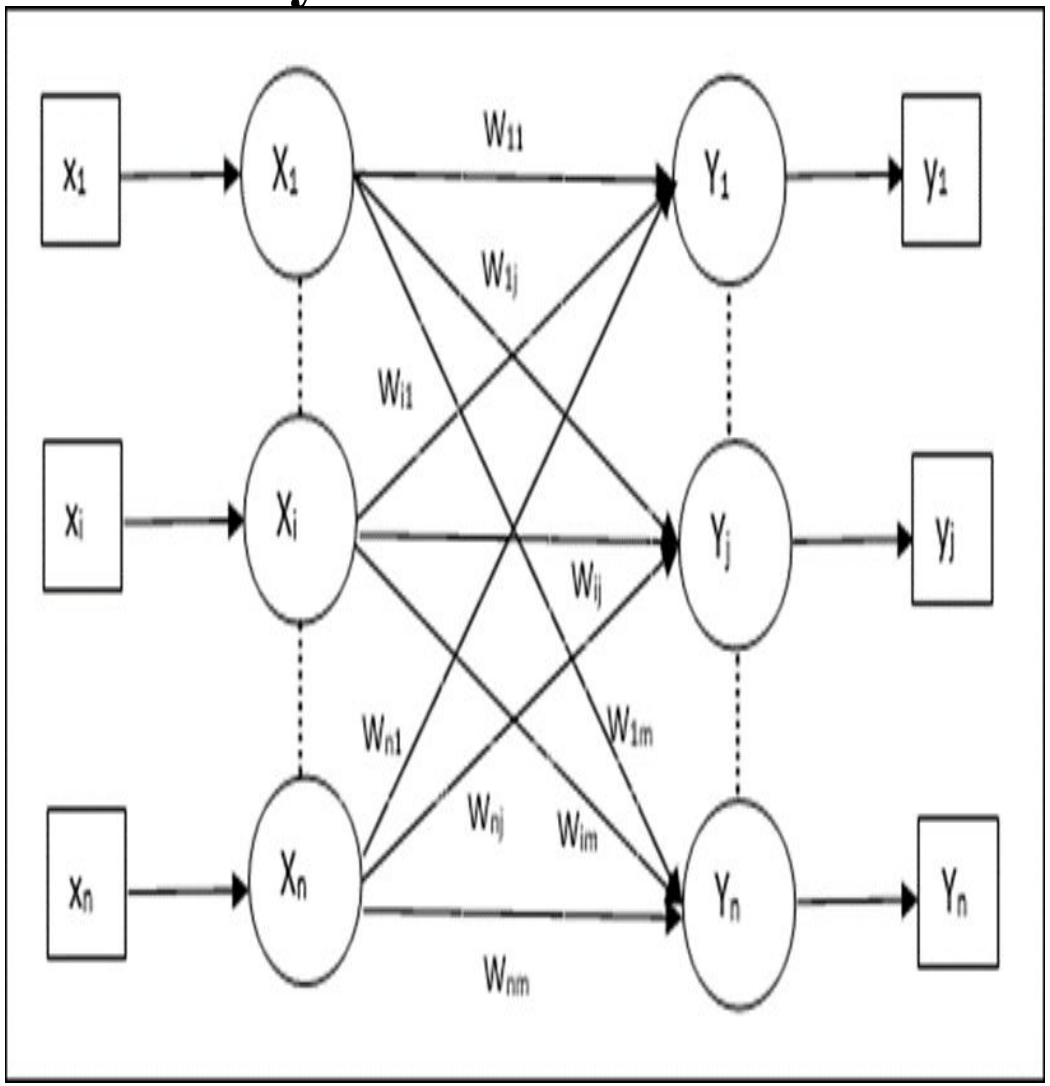
$$Xi = si \text{ (i=1 to n)}$$

Step 4 − Activate each output unit as follows −

$$Yj = sj \text{ (j=1 to n)}$$

Step 5 − Adjust the weights as follows −

$$wij(new) = wij(old) + xiyj$$



Auto Associative Memory

Testing Algorithm

Step 1 − Set the weights obtained during training for Hebb's rule.

Step 2 − Perform steps 3-5 for each input vector.

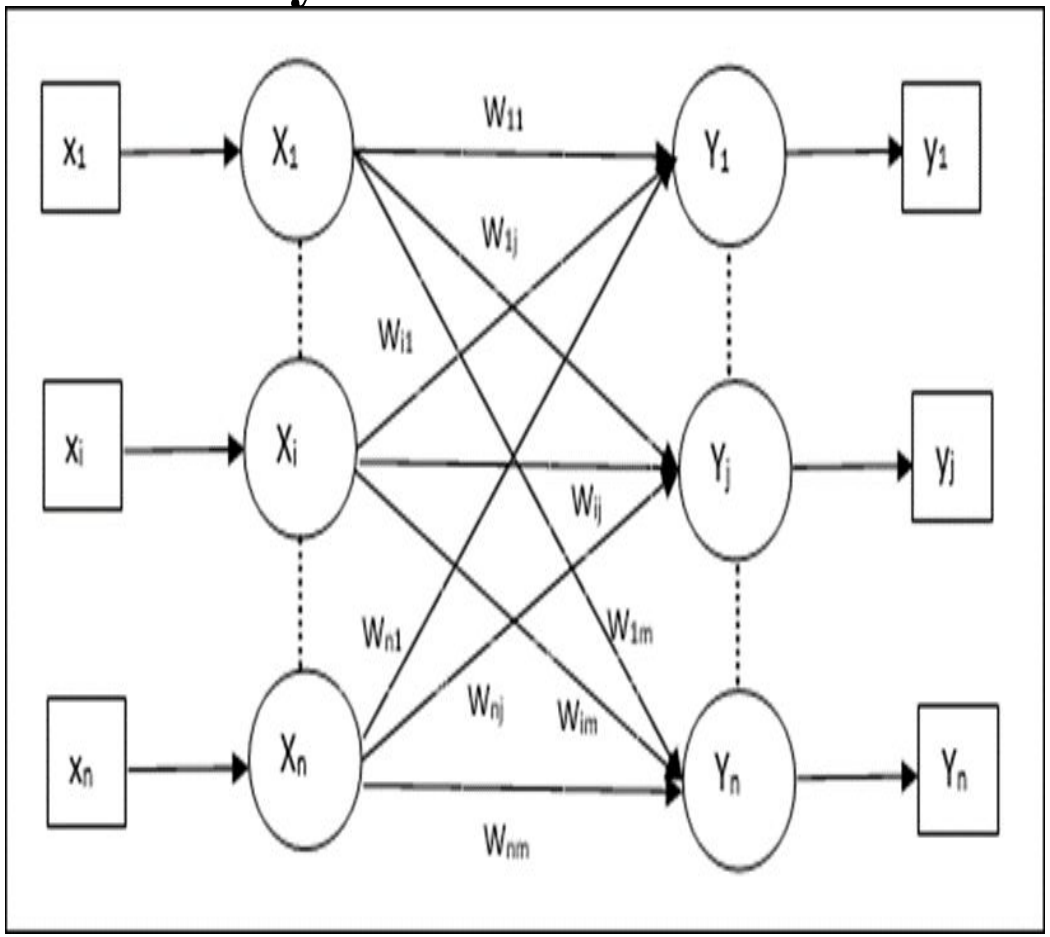
Step 3 − Set the activation of the input units equal to that of the input vector.

Step 4 − Calculate the net input to each output unit **j = 1 to n**

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 − Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 & if\ y_{inj} > 0 \\ -1 & if\ y_{inj} \leqslant 0 \end{cases}$$



Hetero Associative memory

Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as $\mathbf{w_{ij}} = \mathbf{0}$ i=1ton,j=1ton

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows −

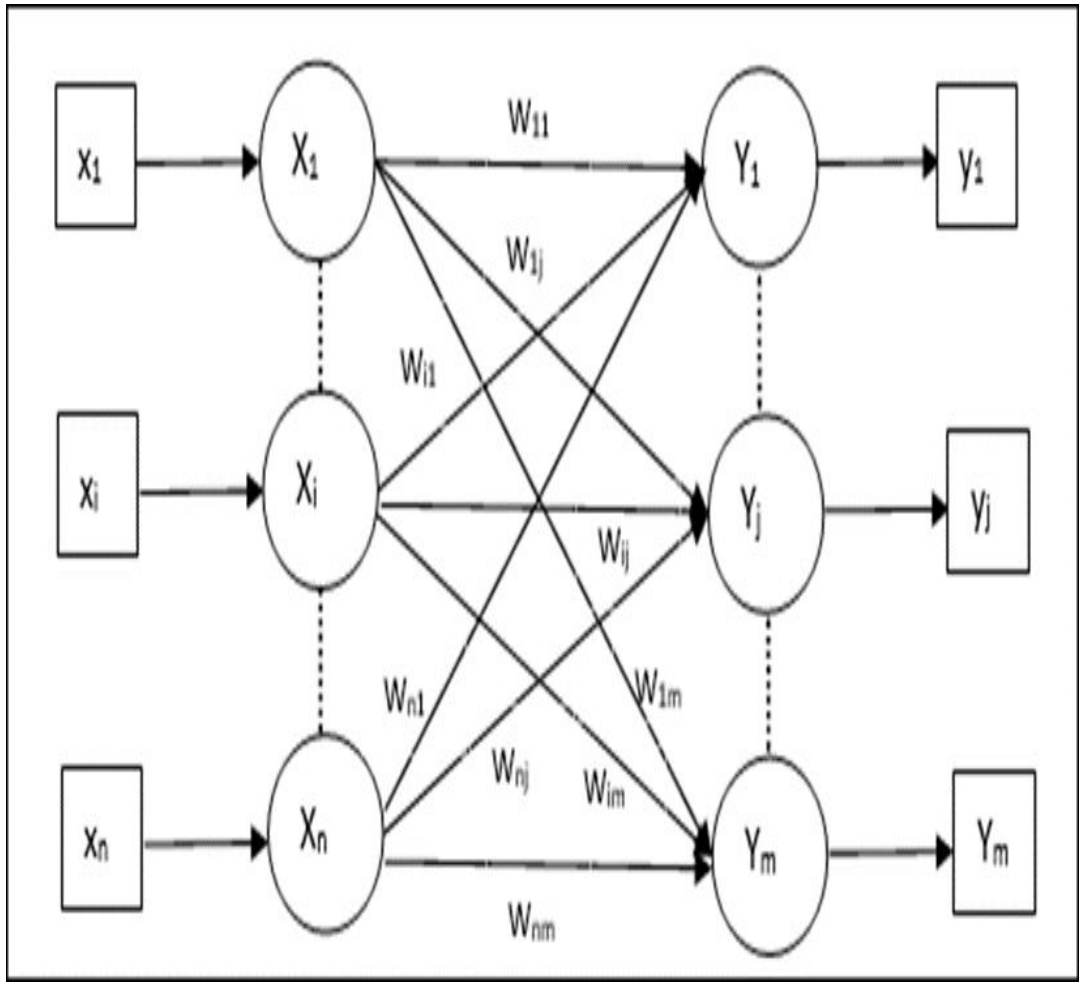
$$Xi = si \ (i=1 \text{ to } n)$$

Step 4 − Activate each output unit as follows −

$$Yj = sj \ (j=1 \text{ to } n)$$

Step 5 − Adjust the weights as follows −

$$wij(new) = wij(old) + xiyj$$



Hetero Associative memory

Testing Algorithm

Step 1 − Set the weights obtained during training for Hebb's rule.

Step 2 − Perform steps 3-5 for each input vector.

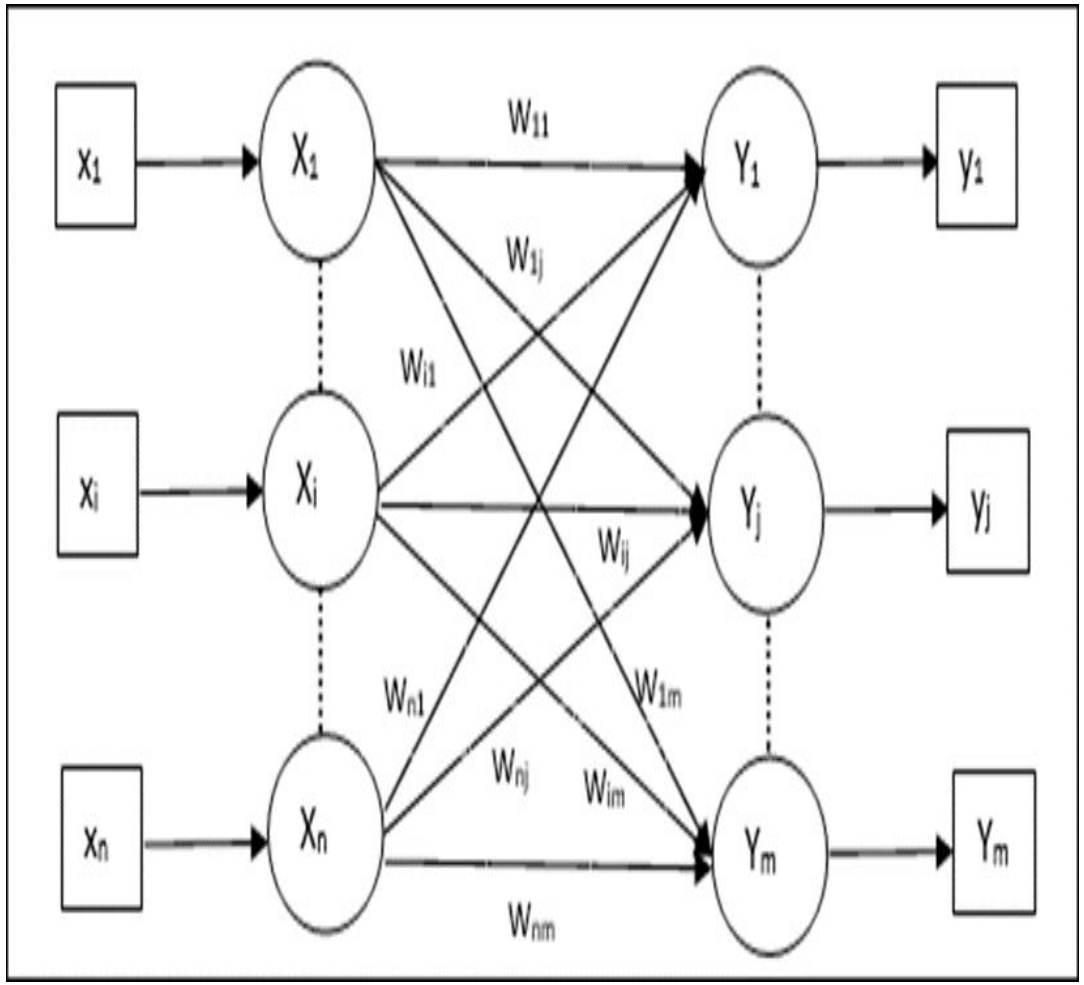
Step 3 − Set the activation of the input units equal to that of the input vector.

Step 4 − Calculate the net input to each output unit **j = 1 to m**

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 − Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 & if\ y_{inj} > 0 \\ 0 & if\ y_{inj} = 0 \\ -1 & if\ y_{inj} < 0 \end{cases}$$



z