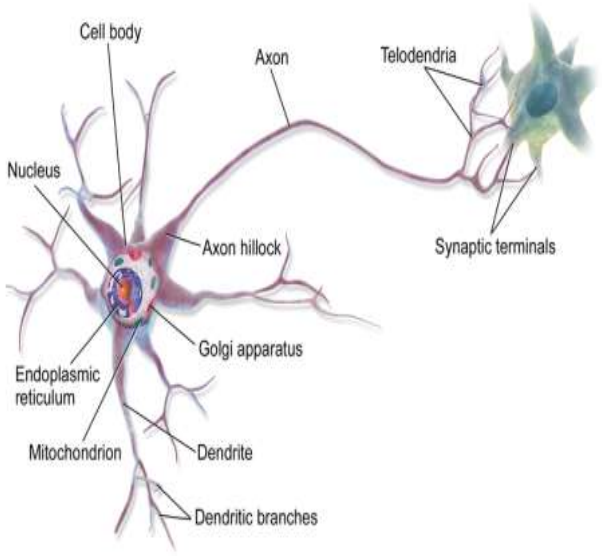
GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU
NAAC A++ ACCREDITED

ARTIFICIAL NEURAL NETWORKS(CSEN3011)

UNIT-I

Introduction to Neural Networks



By

Dr.Satyabrata Dash
Assistant Professor
Department of Computer Science and Engineering, GITAM Deemed to be University

GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU
NAAC A++ ACCREDITED

Course Educational Objectives(ANN)

1. To understand the architecture, learning algorithm and issues of various neural networks.
2. Analyze ANN learning, Error correction learning, Memory-based learning, Competitive learning and Boltzmann learning
3. To adopt gradient - descent techniques in real time applications
4. Provide knowledge on Generalization and function approximation and various
5. architectures of building an ANN
6. Implement and learn the applications of Self-organization Map

SYLLABUS

GITAM (DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU
NAAC A++ ACCREDITED

UNIT 1 Introduction to Neural Networks 9 hours, P - 6 hours

Introduction, The Basic Architecture of Neural Networks, Training a Neural Network with Backpropagation, Practical Issues in Neural Network Training, Common Neural Architectures

UNIT 2 Shallow Neural Networks 9 hours, P - 6 hours

Neural Architectures for Binary Classification Models, Neural Architectures for Multiclass Models, Autoencoder: Basic Principles, Neural embedding with continuous bag of words, Simple neural architectures for graph embedding

SYLLABUS

GITAM (DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU
NAAC A++ ACCREDITED

UNIT 3 Deep Neural Networks 9 hours, P - 6 hours

Introduction, Backpropagation, Setup and Initialization Issues, Gradient-Descent strategies, the bias-variance trade-off, Generalization Issues in Model Tuning and Evaluation, Ensemble Methods

UNIT 4 Attractor Neural Networks 9 hours, P - 6 hours

Associative Learning, Attractor Associative Memory, Linear Associative memory, Hopfield Network, application of Hopfield Network, Brain State in a Box neural Network, Simulated Annealing, Boltzmann Machine, Bidirectional Associative Memory.

SYLLABUS

GITAM (DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU
NAAC A++ ACCREDITED

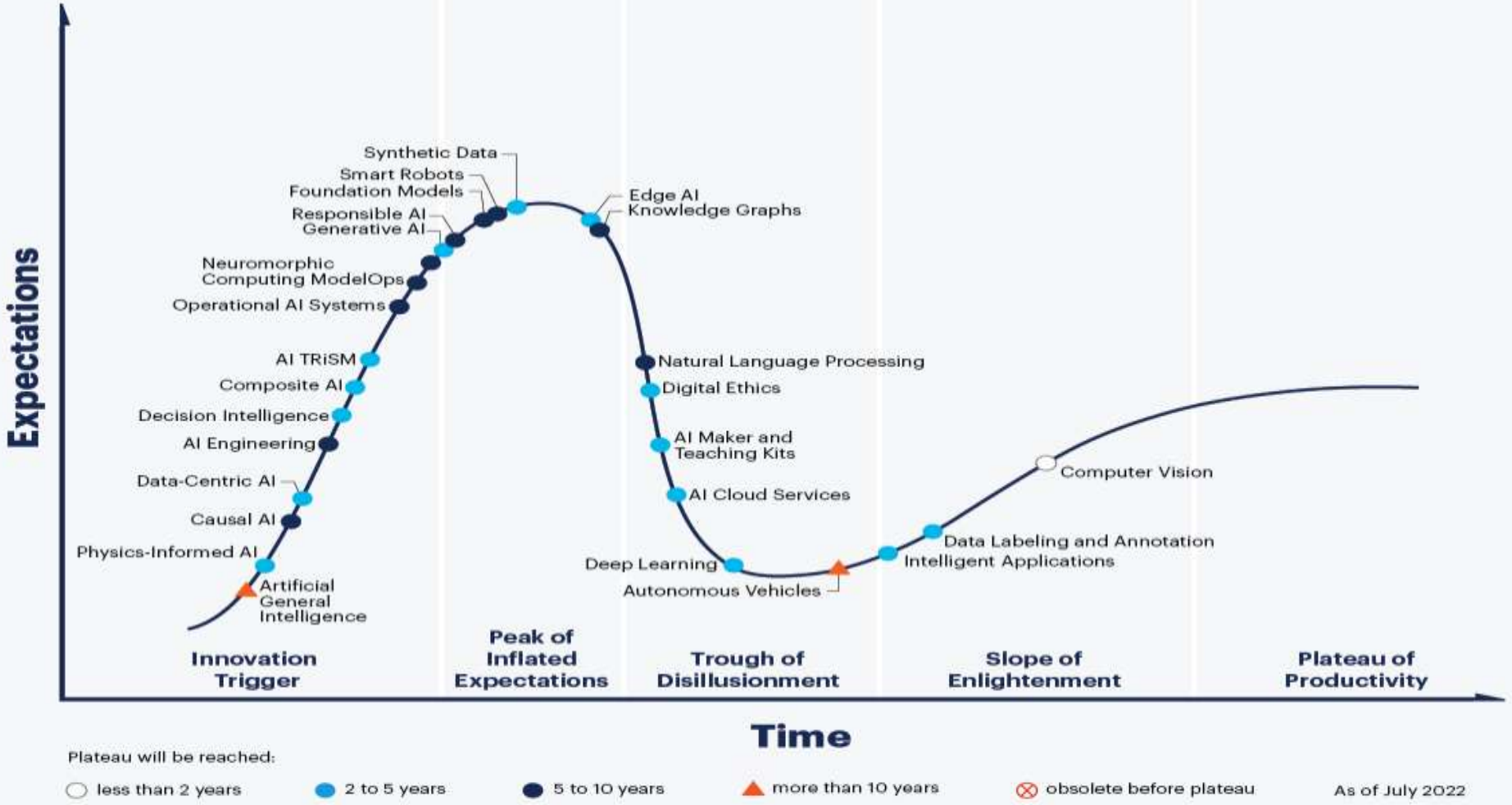
UNIT 5 Self-organization Feature Map 9 hours, P - 6 hours

Maximal Eigenvector Filtering, Extracting Principal Components, Generalized Learning Laws, VectorQuantization, Self-organization Feature Maps, Application of SOM.

Textbooks: 1. Neural Networks and Deep Learning - Charu C. Aggarwal, Springer International Publishing AG, part of Springer Nature 2018 **(Chapters 1, 2, 3)**

2. Neural Networks A Classroom Approach– Satish Kumar, McGraw Hill Education (India) Pvt. Ltd, Second Edition. **(Chapters 4, 5)**

Hype Cycle for Artificial Intelligence, 2022



Plateau will be reached:

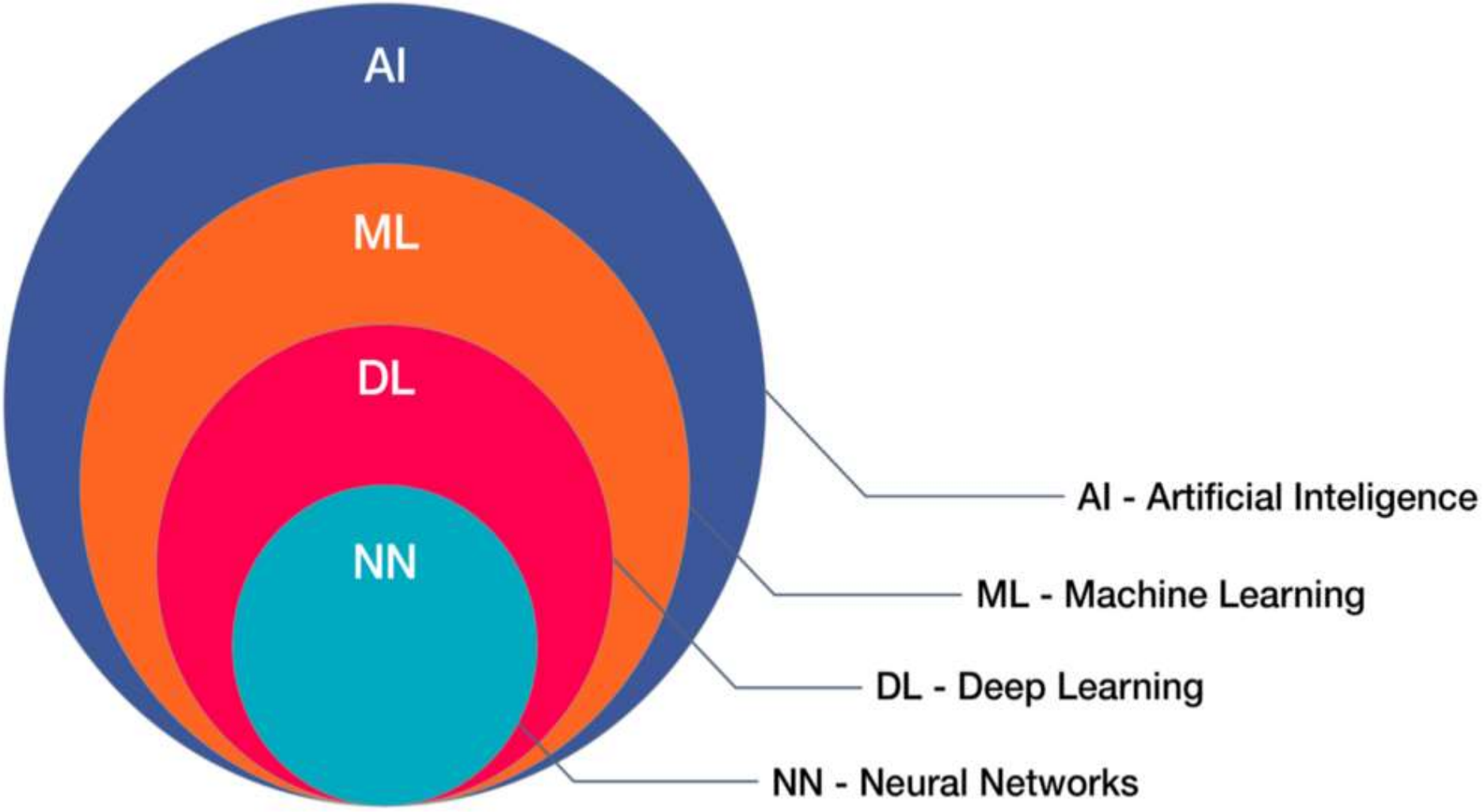
○ less than 2 years ● 2 to 5 years ● 5 to 10 years ▲ more than 10 years ⊗ obsolete before plateau

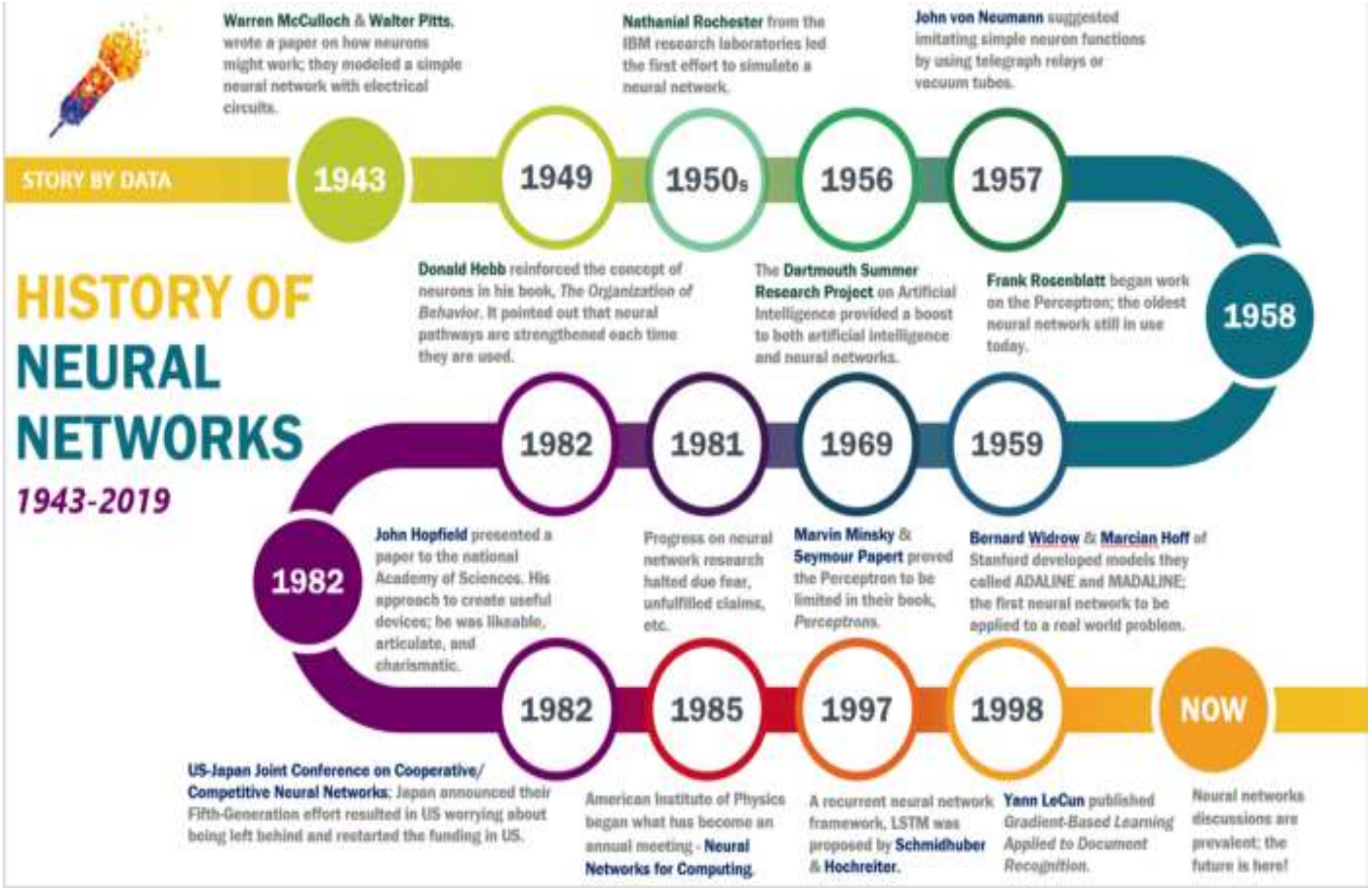
As of July 2022

gartner.com

Gartner®

AI,ML,DL,ANN


AI
ML
DL
NN
AI - Artificial Inteligence
ML - Machine Learning
DL - Deep Learning
NN - Neural Networks


HISTORY OF NEURAL NETWORKS
1943-2019
STORY BY DATA
1943
Warren McCulloch & Walter Pitts, wrote a paper on how neurons might work; they modeled a simple neural network with electrical circuits.
1949
Donald Hebb reinforced the concept of neurons in his book, The Organization of Behavior. It pointed out that neural pathways are strengthened each time they are used.
1950s
Nathanial Rochester from the IBM research laboratories led the first effort to simulate a neural network.
1956
The Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks.
1957
John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes.
1958
Frank Rosenblatt began work on the Perceptron; the oldest neural network still in use today.
1959
Bernard Widrow & Marcian Hoff of Stanford developed models they called ADALINE and MADALINE; the first neural network to be applied to a real world problem.
1969
Marvin Minsky & Seymour Papert proved the Perceptron to be limited in their book, Perceptrons.
1981
Progress on neural network research halted due fear, unfulfilled claims, etc.
1982
John Hopfield presented a paper to the national Academy of Sciences. His approach to create useful devices; he was likeable, articulate, and charismatic.
1982
US-Japan Joint Conference on Cooperative/ Competitive Neural Networks: Japan announced their Fifth-Generation effort resulted in US worrying about being left behind and restarted the funding in US.
1985
American Institute of Physics began what has become an annual meeting - Neural Networks for Computing.
1997
A recurrent neural network framework, LSTM was proposed by Schmidhuber & Hochreiter.
1998
Yann LeCun published Gradient-Based Learning Applied to Document Recognition.
NOW
Neural networks discussions are prevalent; the future is here!

INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS (ANN)

About the Subject:		
Artificial:	**Neural:**	**Networks:**
⌗ Artificial is Made or produced by human beings rather than occurring naturally, especially as a copy of something natural.	⌗ Neural is Relating to, a nerve or the nervous system of Brain. ⌗ Neurons are the information processing units of the brain	⌗ A network is a group of two or more computers or other electronic devices that are interconnected for the purpose of exchanging data and sharing resources.

INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS (ANN)

Objective of NEURAL NETWORKS

1. NN are constructed and implemented to model the human brain.
2. Performs various tasks such as pattern-matching, classification, optimization function, approximation, vector quantization and data clustering.
3. The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons in brains.
4. The human brain contains a densely interconnected network of approximately 1011-1012 neurons, each connected neuron, on average connected, to l04-105 other neurons. So on average human brain takes approximately 10^{-1} to make surprisingly complex decisions.

INTRODUCTION TO ARTIFICIAL NEURAL NETWORKS (ANN)

ANN Definitions

Definition :1

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the brain. ANNs, like people, learn by examples.

Definition :2

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

Definition :3

Neural networks reflect the behavior of the human brain, allowing computer programs to recognize patterns and solve common problems in the fields of AI, machine learning, and deep learning.

Definition :4

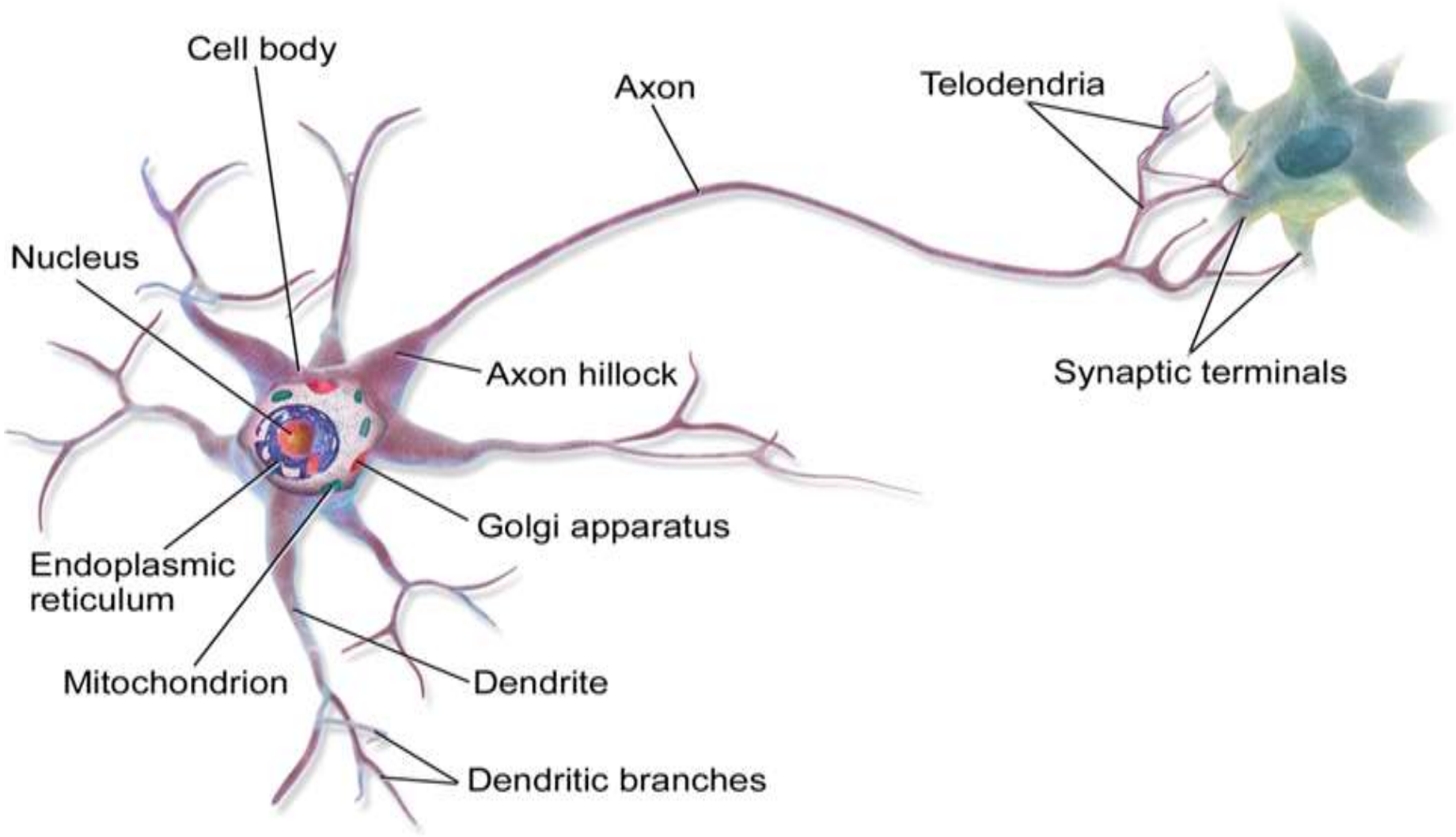
An artificial neural network is an attempt to simulate the network of neurons that make up a human brain so that the computer will be able to learn things and make decisions in a humanlike manner

STRUCTURE AND FUNCTIONS OF BIOLOGICAL NEURON

- The human nervous system contains cells, which are called neurons.
- The neurons are associated with each other with the utilization of axons and dendrites, and the interfacing districts among axons and dendrites are called as synapses

1. **Dendrite**: Receives signals from other neurons
2. **Soma:** Processes the information
3. **Axon:** Transmits the output of this neuron
4. **Synapse:** Point of connection to other neurons

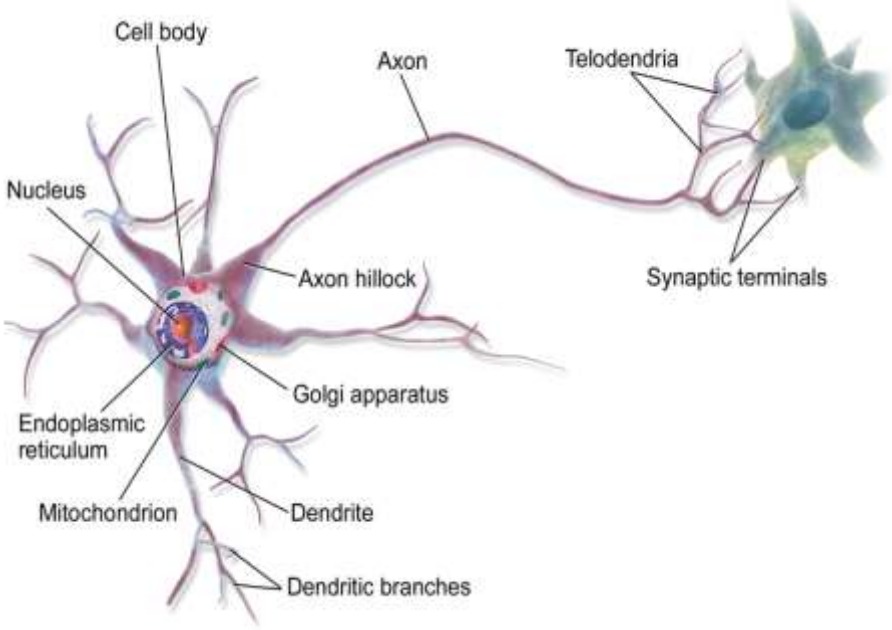
STRUCTURE AND FUNCTIONS OF BIOLOGICAL NEURON

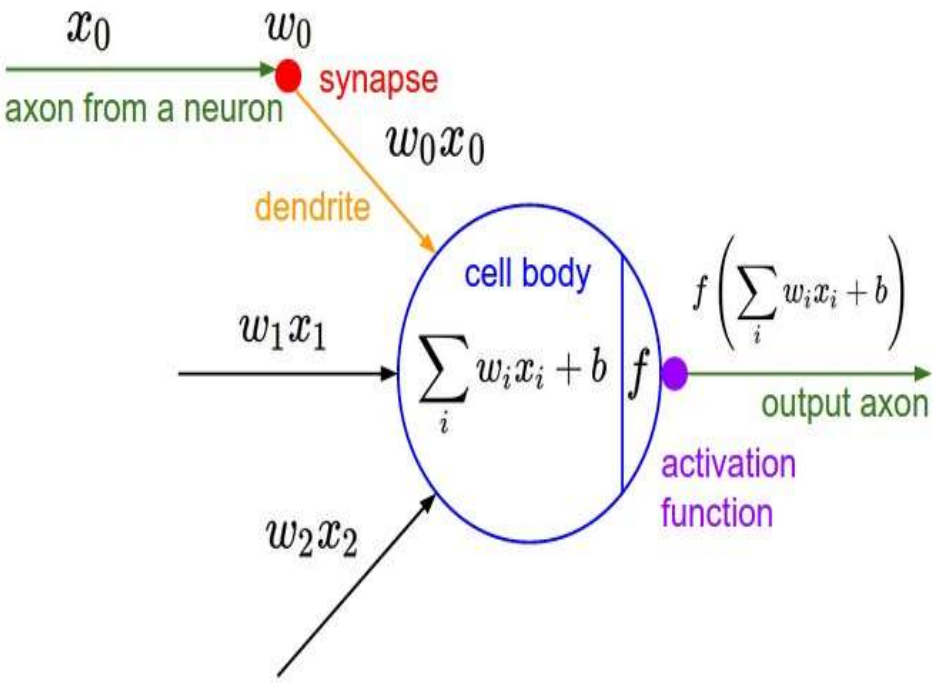


STRUCTURE AND FUNCTIONS OF ARTIFICIAL NEURON

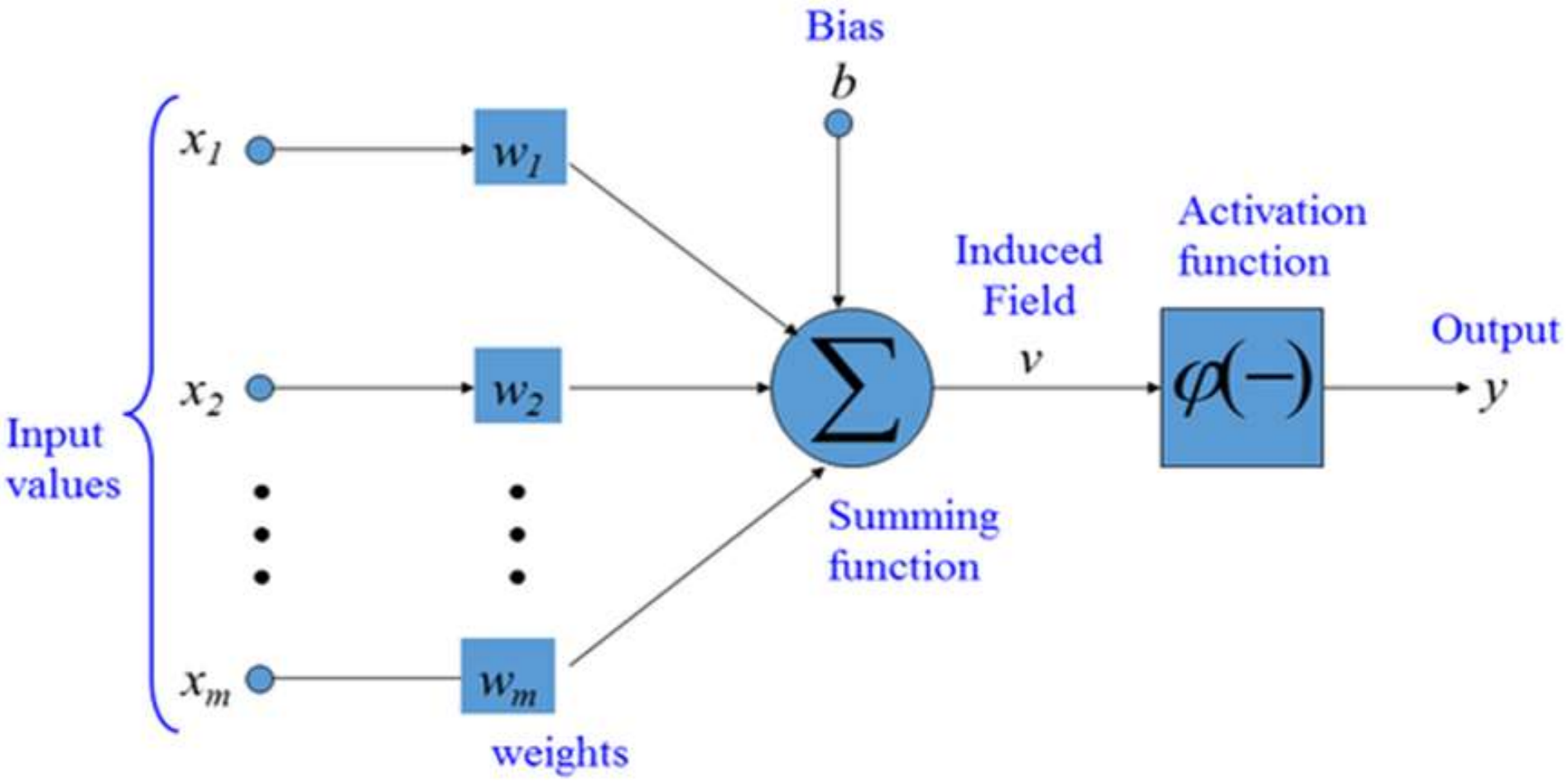
1. Artificial neural systems are well-known machine learning techniques that mimic the composition of learning in biological organisms (natural life forms).
2. ANN poses a large number of processing elements called **nodes/neurons** which operate in parallel.
3. Neurons are connected with others by **connection link**.
4. Each link is associated with weights which contain information about the input signal.
5. Each neuron has an internal state of its own which is a function of the inputs that neuron receives- **Activation level**
6. Bias - The role of bias is to shift the value produced by the activation function. Its role is similar to the role of a constant in a linear function.

STRUCTURE OF ARTIFICIAL NEURON





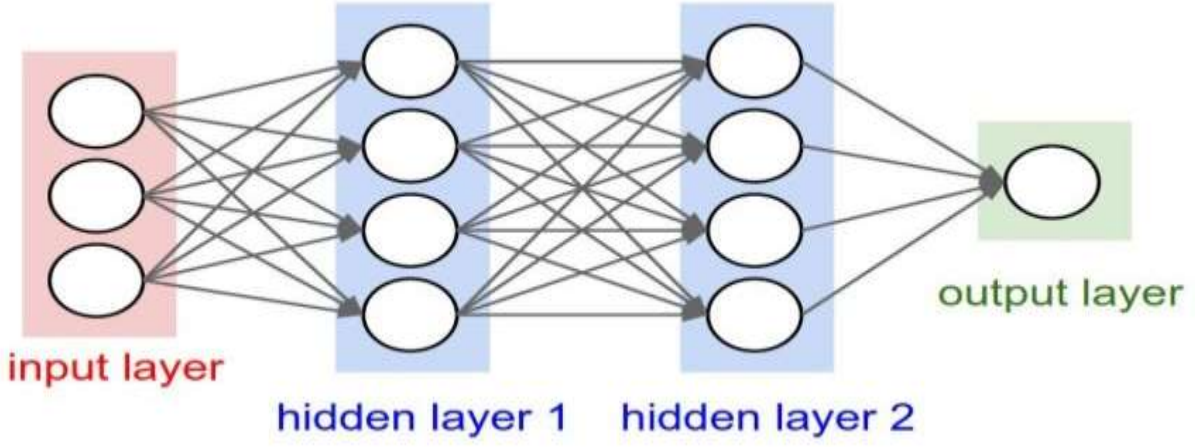
STRUCTURE AND FUNCTIONS OF ARTIFICIAL NEURON



STRUCTURE AND FUNCTIONS OF ARTIFICIAL NEURON

Elements of a Neural Network :-

1. ***Input Layer :-*** This layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.
2. ***Hidden Layer :-*** Nodes of this layer are not exposed to the outer world, they are the part of the abstraction provided by any neural network. Hidden layer performs all sort of computation on the features entered through the input layer and transfer the result to the output layer.
3. ***Output Layer :-*** This layer bring up the information learned by the network to the outer world.

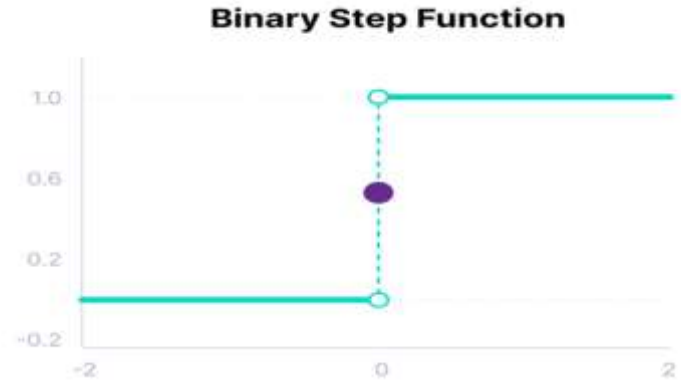


ANN Activation Functions.

ANNs use activation functions (AFs) to perform complex computations in the hidden layers and then transfer the result to the output layer. The primary purpose of AFs is to introduce non-linear properties in the neural network. Some popular neural networks activation functions.

1.Binary Step Function

Binary step function/Threshold function depends on a threshold value that decides whether a neuron should be activated or not. The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.



$$\textit{Binary step}$$

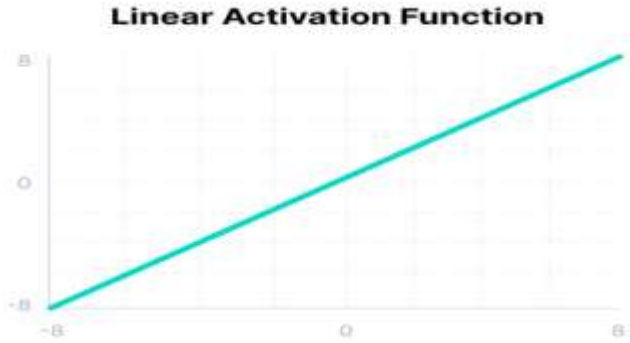
$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

2. Linear Activation Function

The linear activation function is also known as Identity Function where the activation is proportional to the input. Mathematically it can be represented as:

$$Linear$$

$$f(x) = x$$



3. Sigmoid / Logistic Activation Function

This function takes any real value as input and outputs values in the range of 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0, as shown below.

$$Sigmoid\ /\ Logistic$$

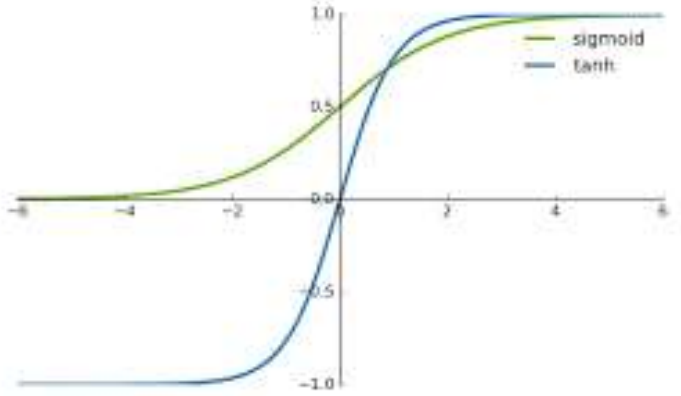
$$f(x) = \frac{1}{1 + e^{-x}}$$

4.Tanh Function (Hyperbolic Tangent)

Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

$$Tanh$$

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

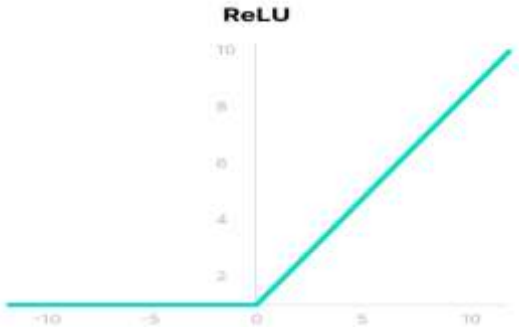


5. ReLU Function

ReLU stands for Rectified Linear Unit. Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. The main catch here is that the ReLU function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0.

$$ReLU$$

$$f(x) = max\,(0, x)$$



Example-1: **Design neural network represents basic Boolean function AND with two input variables and bias, using the threshold function as the output activation function.**

Solutions:

Let **x1, x2** be the two input variables, **w0** the weight of the bias and **y** the output of the N.N.

Since the output of the AND gate is 1 only when x1 and x2 are 1's, this will lead to the following:

$$w1x1 + w2x2 + w0 \leq 0 , \quad \text{for } y = 0$$

$$w1x1 + w2x2 + w0 > 0 , \quad \text{for } y = 1$$

$$w1x1 + w2x2 + w0 \leq 0 , \qquad \text{for } y = 0$$

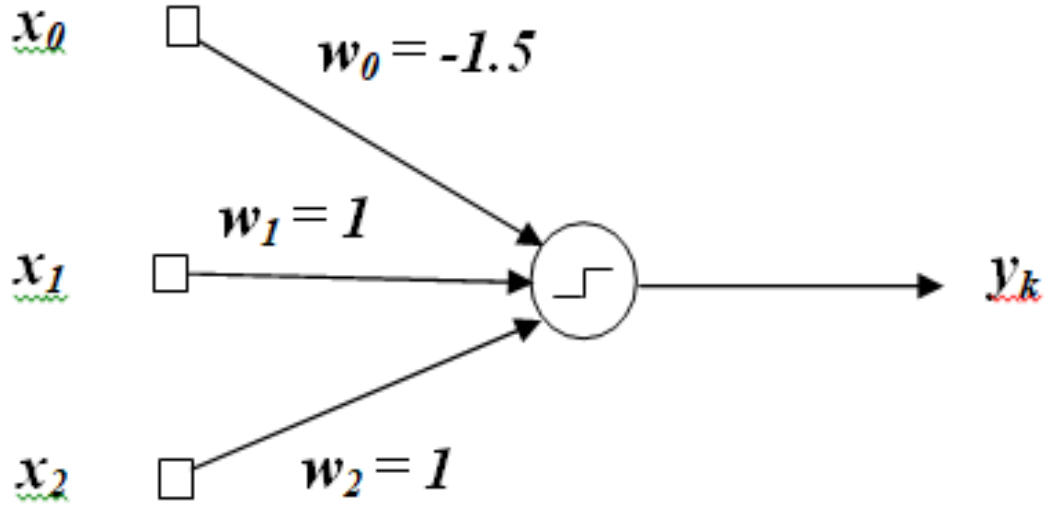
$$w1x1 + w2x2 + w0 > 0 , \qquad \text{for } y = 1$$

x_1	x_2	$w_1x_1 + w_2x_2$	y	
0	0	0	0	$w_0 \leq 0$
1	0	w_1	0	$(w_1 + w_0) \leq 0$
0	1	w_2	0	$(w_2 + w_0) \leq 0$
1	1	$w_1 + w_2$	1	$(w_1 + w_2 + w_0) > 0$

From the table we can recognize that w0 is negative value, and (w1 +w2 + w0) should be positive, thus,

w1 = w2 = 1, and w0 = -1.5 will satisfy the required conditions.

w1 = w2 = 1, and w0 = -1.5 will satisfy the required conditions.



Learning and Types of Learning rules in ANN

Learning rule is a method or a mathematical logic.

- It helps a Neural Network to learn from the existing conditions and improve its performance.
- Applying learning rule is an iterative process.

There are 5 different learning rules in the Neural network:

1	**Hebbian learning rule**	It identifies how to modify the weights of nodes of a network.
2	**Perceptron learning rule**	Network starts its learning by assigning a random value to each weight
3	**Delta learning rule**	Modification in sympatric weight of a node is equal to the multiplication of error and the input.
4	**Correlation learning rule**	The correlation rule is the supervised learning
5	**Outstar learning rule**	We can use it when it assumes that nodes or neurons in a network arranged in a layer

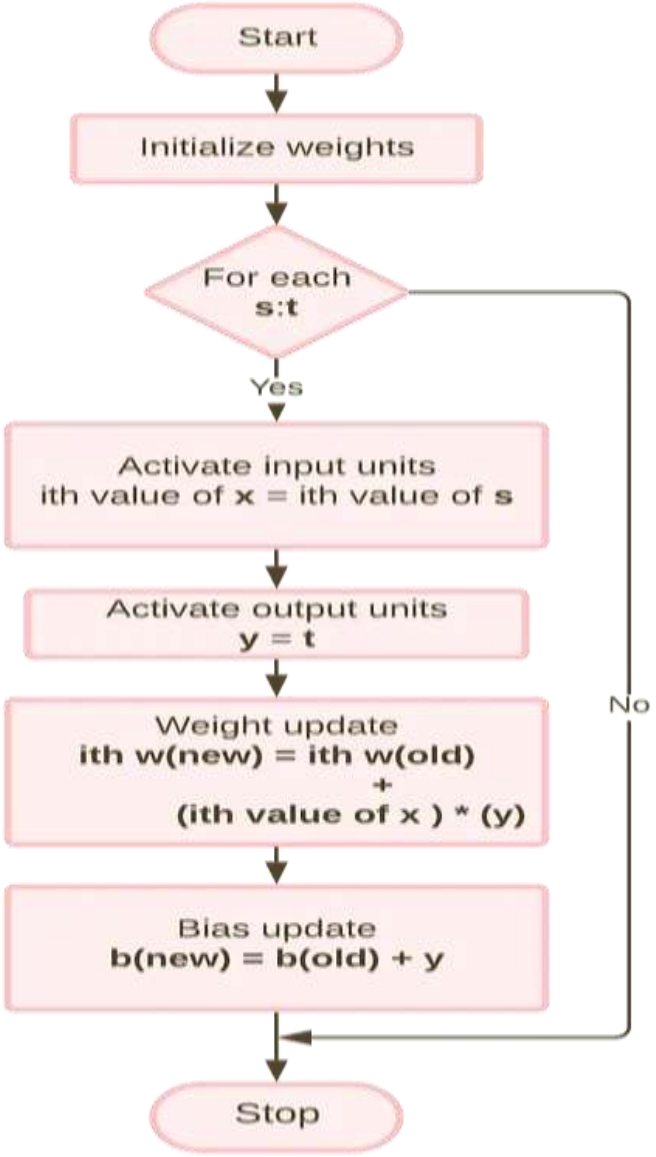
Hebbian learning Rule with Example

- It is one of the first and also easiest learning rules in the neural network.
- It is used for **pattern classification**.
- It is a **single layer neural network**, i.e. it has one input layer and one output layer.
- The input layer can have many units, say n. The output layer only has one unit.
- Hebbian rule works by **updating the weights between neurons** in the neural network for each training sample.

Hebbian Learning Rule Algorithm :

1. Set all weights to zero, **wi = 0** for i=1 to n, and bias to zero.
2. For each input vector, **S(input vector) : t(target output pair),** repeat steps 3-5.
3. Set activations for input units with the input vector **Xi = Si** for i = 1 to n.
4. Set the corresponding output value to the output neuron, i.e. **y = t**.
5. Update weight and bias by applying Hebb rule for all i = 1 to n:

Hebbian learning Rule(Flow Chart)



$$w_i(new) = w_i(old) + x_i y$$

$$b(new) = b(old) + y$$

Hebbian learning Rule with Example

Example: Designing a Hebb network to implement AND function using bipolar input

Input & Target Values

Inputs			Target
x1	x2	b	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Here we have used '-1' instead of '0' this is because the Hebb network uses **bipolar data** and not binary data because the product item in the above equations would give the output as 0 which leads to a wrong calculation.

Hebbian learning Rule with Example

Starting with setp1 which is inializing the weights and bias to ‘0’, so we get w1=w2=b=0

A) First input **[x1,x2,b]=[1,1,1]** and target/**y = 1.** Now using the initial weights as old weight and applying the Hebb rule
B) (ith value of w(new) = ith value of w(old) + (ith value of x * y)) as follow;

w1(new) = w1(old) + (x1*y) = 0+1 * 1 = 1
w2(new) = w2(old) + (x2*y) = 0+1 * 1 = 1
b(new) = b(old) + y = 0+1 =1

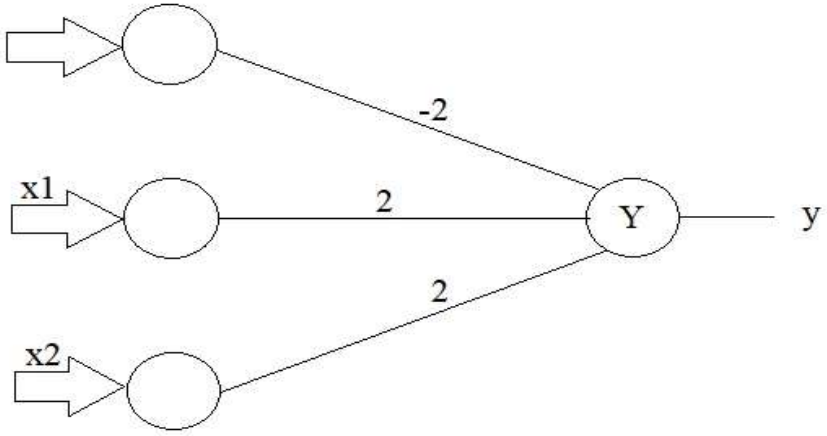
Δith value of w = ith value of x * y
hence weight changes relating to the first input are;
Δw1= x1y = 1*1=1
Δw2 = x2y = 1*1=1
Δb = y = 1

Hebbian learning Rule with Example

So using the same process process for third and fourth row we get a new table as follows

Inputs				Weight Changes			Weights		
x1	x2	b	y	Δw1	Δw2	Δb	w1 (0	w2 0	b 0)
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

Here the final weights we get are w1=2, w2=2, b=-2



Final Hebb network with AND function

Artificial Neural Network Architecture

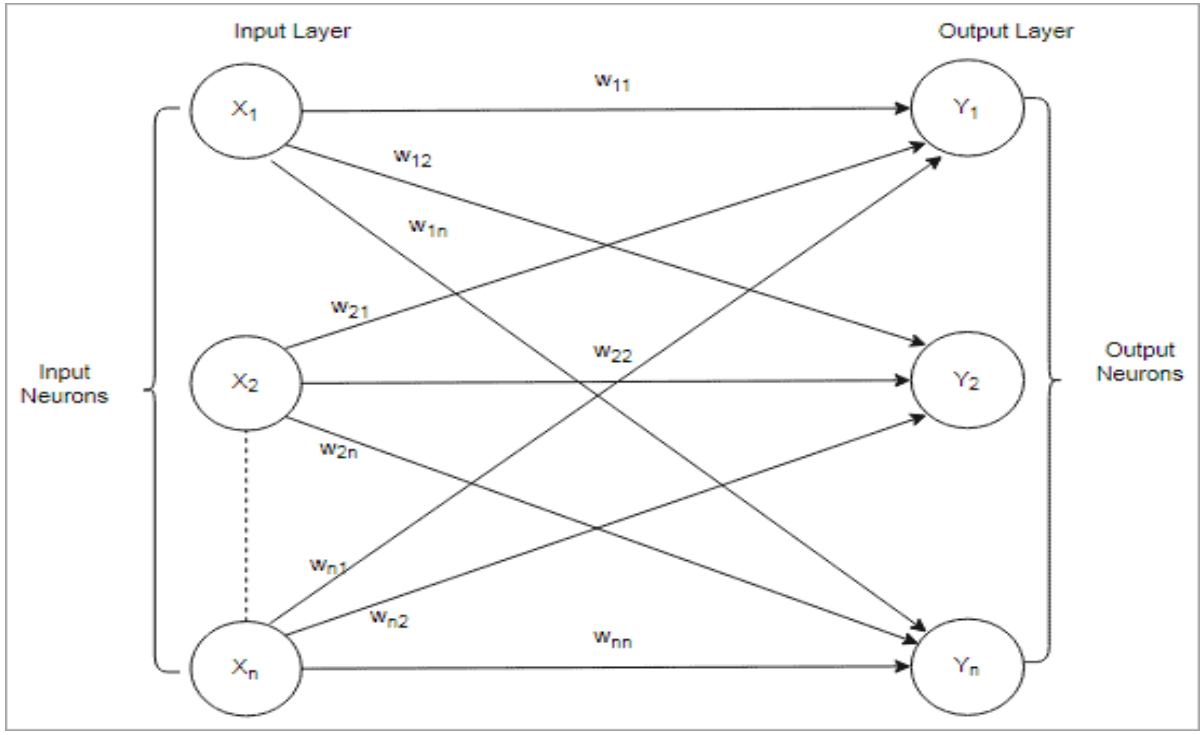
There exist five basic types of neuron connection architecture :

- Single-layer feed-forward network
- Multilayer feed-forward network
- Single node with its own feedback
- Single-layer recurrent network
- Multilayer recurrent network

Artificial Neural Network Architecture

Single Layer Feed Forward Network

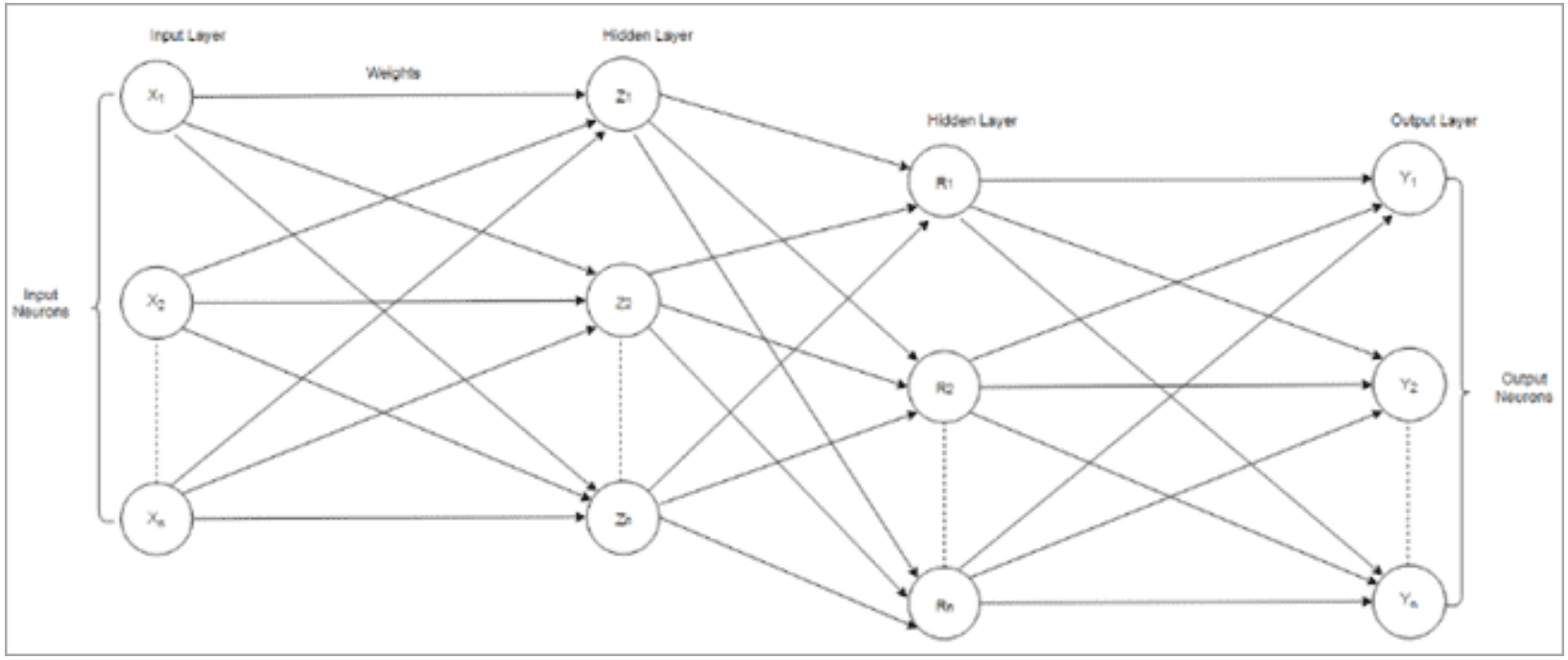
1. In this type of network, we have only two layers, i.e. input layer and output layer but the input layer does not count because no computation is performed in this layer.
2. Output Layer is formed when different weights are applied on input nodes and the cumulative effect per node is taken.
3. After this, the neurons collectively give the output layer to compute the output signals.



Artificial Neural Network Architecture

Multilayer Feed Forward Network

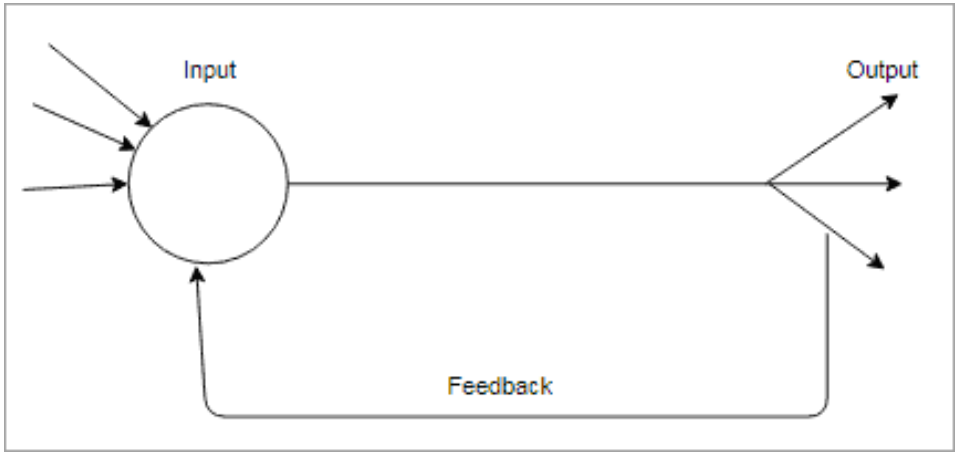
1. This network has **a hidden layer** that is internal to the network and has no direct contact with the external layer.
2. The existence of one or more hidden layers enables the network to be computationally stronger.
3. There are **no feedback connections** in which outputs of the model are fed-back into itself.



Artificial Neural Network Architecture

Single node with its own feedback

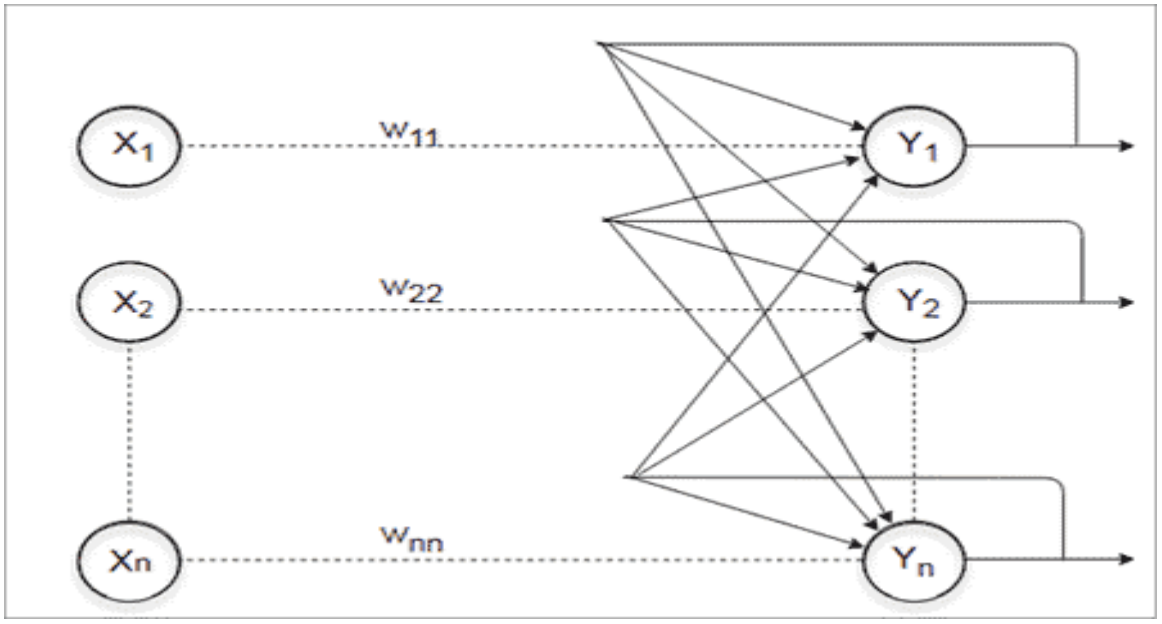
1. When outputs can be directed back as inputs to the same layer or proceeding layer nodes, then it results in feedback networks.
2. Recurrent networks are feedback networks with closed loop.
3. The figure below shows a single recurrent having a single neuron with feedback to itself.



Artificial Neural Network Architecture

Single Layer Recurrent Network

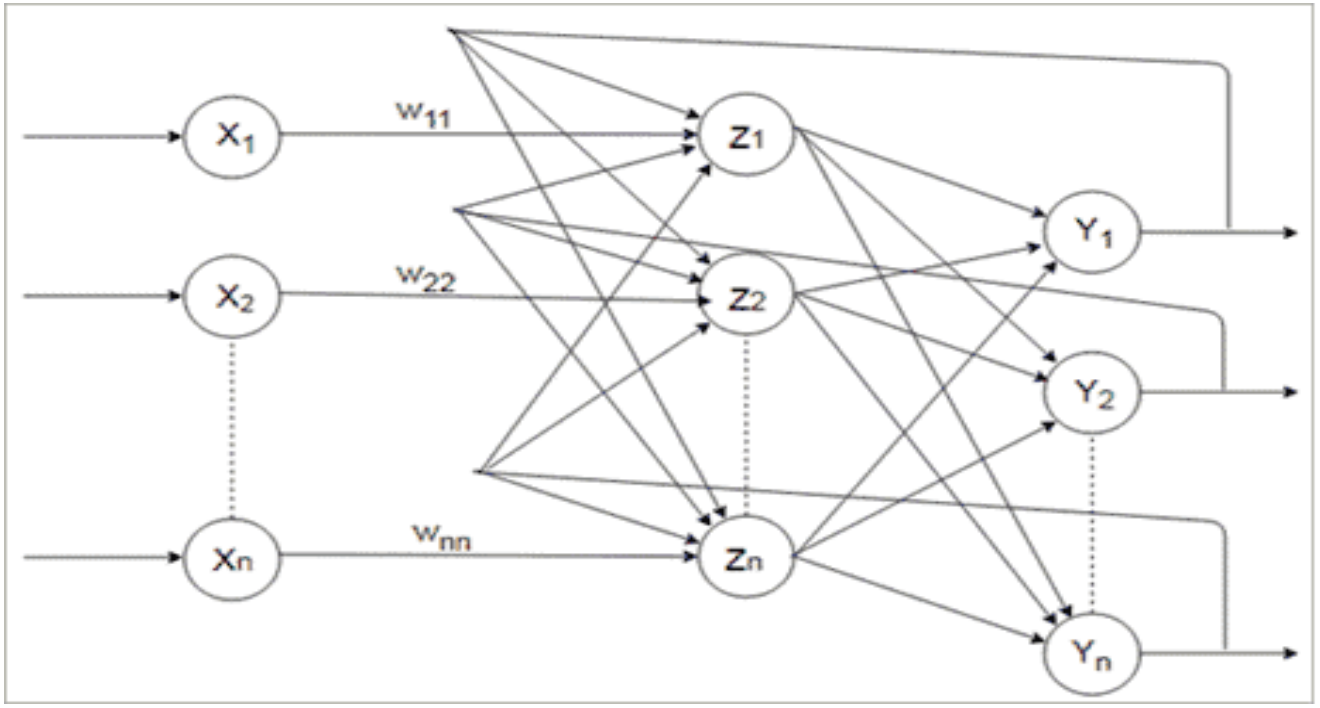
1. This network is a single-layer network with a feedback connection in which the processing element's output can be directed back to itself or to other processing elements or both.
2. A recurrent neural network is a class of artificial neural network where the connection between nodes forms a directed graph along a sequence.
3. This allows is it to exhibit dynamic temporal behavior for a time sequence. Unlike feed-forward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.

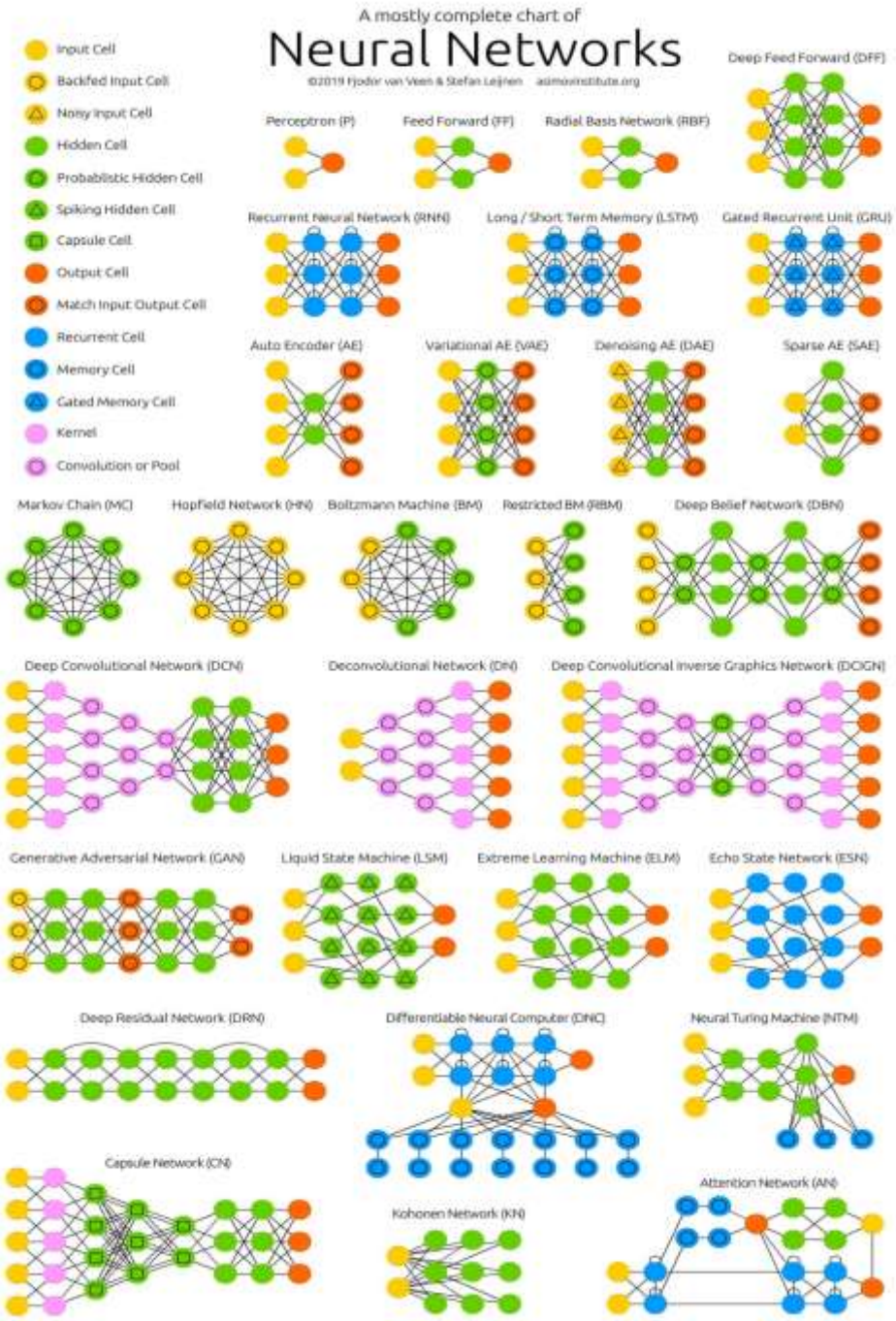


Artificial Neural Network Architecture

Multilayer Recurrent Network

1. In this type of network, processing element output can be directed to the processing element in the same layer and in the preceding layer forming a multilayer recurrent network.
2. They perform the same task for every element of the sequence, with the output being dependent on the previous computations. Inputs are not needed at each time step.
3. The main feature of a multilayer recurrent network is its hidden state, which captures information about a sequence.





10 most popular deep learning algorithms:

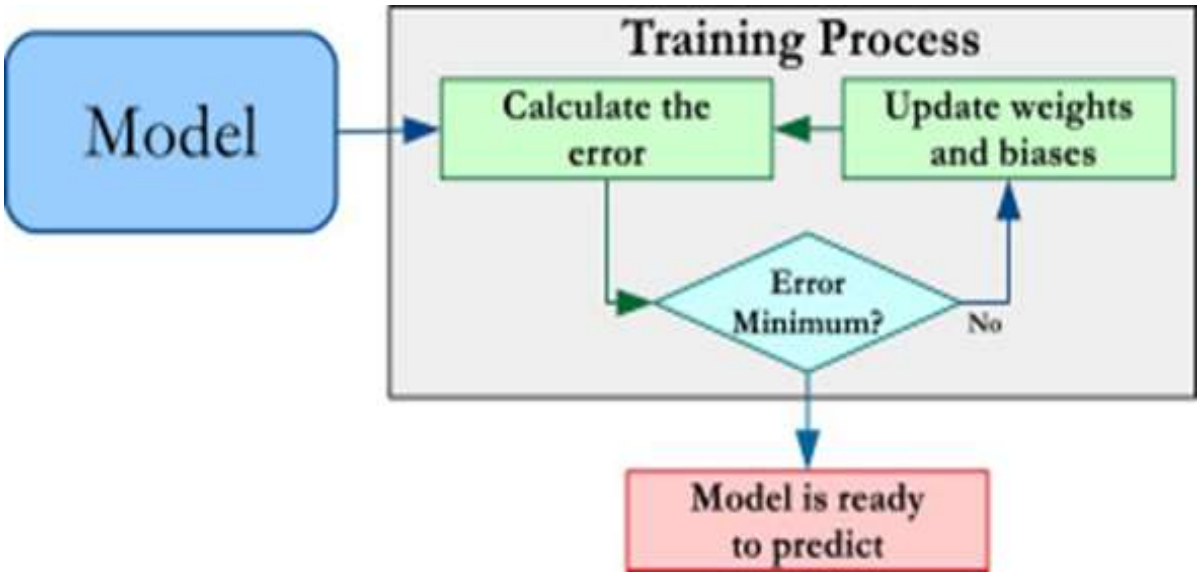
1. Convolutional Neural Networks (CNNs)
2. Long Short Term Memory Networks (LSTMs)
3. Recurrent Neural Networks (RNNs)
4. Generative Adversarial Networks (GANs)
5. Radial Basis Function Networks (RBFNs)
6. Multilayer Perceptrons (MLPs)
7. Self Organizing Maps (SOMs)
8. Deep Belief Networks (DBNs)
9. Restricted Boltzmann Machines(RBMs)
10. Autoencoders

Training a Neural Network with Backpropagation,

Back Propagation Learning

1. Backpropagation is a supervised learning algorithm, for training Multi-layer Perceptrons (Artificial Neural Networks).
2. Backpropagation (backward propagation) is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning.
3. The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the **delta rule or gradient descent**. The weights that minimize the error function is then considered to be a solution to the learning problem.
4. One way to train our model is called as **Backpropagation.** Consider the diagram below:

Backpropagation Process & Steps



Steps:

1. **Calculate the error** – How far is your model output from the actual output.
2. **Minimum Error** – Check whether the error is minimized or not.
3. **Update the parameters** – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
4. **Model is ready to make a prediction** – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output

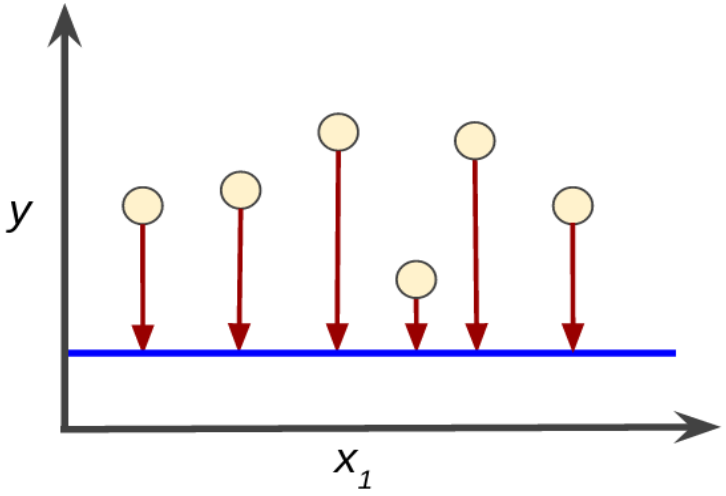
Errors,Training and Test Loss

- **Training and Test** Loss is a number indicating how **bad(un-accurate)** the model's prediction was on a single example. If the model's prediction is perfect, the **loss is zero; otherwise, the loss is greater.**
- Means the model is not learning; probably there's something wrong with either the model or the optimization process
- The goal of training a model is to find a set of weights and biases that have *low* loss, on average,
- Computationally, the training loss is calculated by **taking the sum of errors for each example in the training set**.

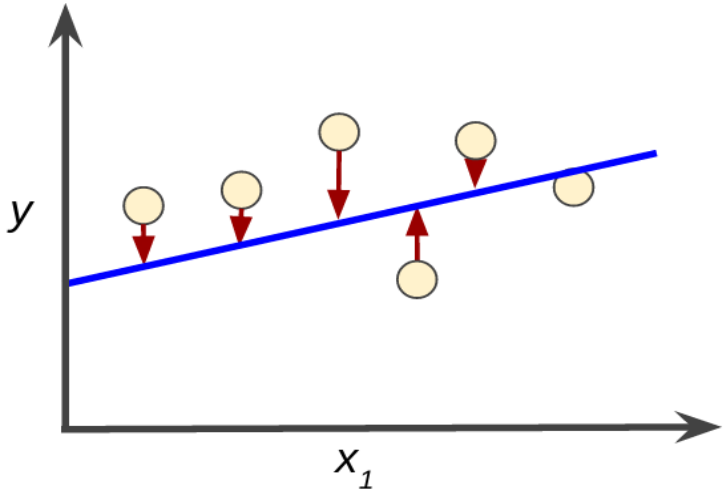
Errors, Training and Test Loss

For example,
lets check a high **loss model (A)**on the left and a **low loss model(B)** on the right.
Where

- The arrows represent loss.
- The blue lines represent predictions.
- Notice that the arrows in the left plot are much longer than their counterparts in the right plot. Clearly, the line in the right plot is a much better predictive model than the line in the left plot.



(A)



(B)

Training a Neural Network with Backpropagation,

Example:
Training a neural network using the backpropagation algorithm.

- In this example, we'll build and train a feedforward neural network to solve a basic binary classification problem. The network will have one hidden layer and will use the sigmoid activation function. We'll use the mean squared error (MSE) loss function and gradient descent as the optimization algorithm.

- **Lets take the following training dataset:**

Input 1	Input 2	Target
0	0	0
0	1	1
1	0	1
1	1	0

Training a Neural Network with Backpropagation,

Step 1: Initialize the neural network parameters

- Let's start by initializing the neural network parameters: weights and biases for the input-to-hidden and hidden-to-output layers
- .

Let's assume:

- **The hidden layer has 2 neurons.**
- **Weights and biases are initialized randomly.**

Step 2: Forward Pass

Next, perform a forward pass through the network to compute the predicted outputs.

- **Input layer:** The input layer consists of two neurons, one for each input feature.
- **Hidden layer:** The hidden layer consists of two neurons. Each neuron will receive inputs from the input layer, and we'll apply the sigmoid activation function to the weighted sum of inputs.
- **Output layer:** The output layer consists of one neuron. It will receive inputs from the hidden layer and also apply the sigmoid activation function.

Training a Neural Network with Backpropagation,

The formulas for forward pass are as follows:

Weighted sum at hidden layer:

$$\text{Hidden}_{\text{weighted}} = \text{Input}_1 \times \text{Weight}_{\text{in}_1} + \text{Input}_2 \times \text{Weight}_{\text{in}_2} + \text{Bias}_{\text{hidden}}$$

Output of hidden layer (after applying activation function):

$$\text{Hidden}_{\text{output}} = \text{sigmoid}(\text{Hidden}_{\text{weighted}})$$

Weighted sum at output layer:

$$\text{Output}_{\text{weighted}} = \text{Hidden}_{\text{output}} \times \text{Weight}_{\text{out}} + \text{Bias}_{\text{output}}$$

Final output (after applying activation function):

$$\text{Output}_{\text{final}} = \text{sigmoid}(\text{Output}_{\text{weighted}})$$

Training a Neural Network with Backpropagation,

Step 3: Compute Loss

Now, we'll compute the mean squared error loss using the predicted outputs and the target values from the dataset.

Mean Squared Error (MSE) Loss:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N}(\text{Target}_i - \text{Output}_{\text{final}_i})^2$$

where N is the number of samples in the dataset.

Step 4: Backpropagation - Compute Gradients

- In the backpropagation step, we compute the gradients of the loss with respect to the network parameters. These gradients will tell us the direction in which to adjust the weights and biases to minimize the loss.
- We use the chain rule to calculate the gradients at each layer.

For the output layer:

$$\text{Gradient}_{\text{Output}} = -2 \times (\text{Target} - \text{Output}_{\text{final}}) \times \text{sigmoid}'(\text{Output}_{\text{weighted}})$$

For the hidden layer:

$$\text{Gradient}_{\text{Hidden}} = \text{Gradient}_{\text{Output}} \times \text{Weight}_{\text{out}} \times \text{sigmoid}'(\text{Hidden}_{\text{weighted}})$$

Training a Neural Network with Backpropagation,

Step 5: Update Weights and Biases

- With the gradients computed, we can now update the weights and biases using gradient descent.

- Learning rate (α) is a hyperparameter that controls the step size in the gradient descent update.

Update equations:

$$\text{Weight}_{\text{out}} = \text{Weight}_{\text{out}} - \alpha \times \text{Gradient}_{\text{Output}} \times \text{Hidden}_{\text{output}}$$

$$\text{Bias}_{\text{output}} = \text{Bias}_{\text{output}} - \alpha \times \text{Gradient}_{\text{Output}}$$

$$\text{Weight}_{\text{in}_1} = \text{Weight}_{\text{in}_1} - \alpha \times \text{Gradient}_{\text{Hidden}} \times \text{Input}_1$$

$$\text{Weight}_{\text{in}_2} = \text{Weight}_{\text{in}_2} - \alpha \times \text{Gradient}_{\text{Hidden}} \times \text{Input}_2$$

$$\text{Bias}_{\text{hidden}} = \text{Bias}_{\text{hidden}} - \alpha \times \text{Gradient}_{\text{Hidden}}$$

Training a Neural Network with Backpropagation,

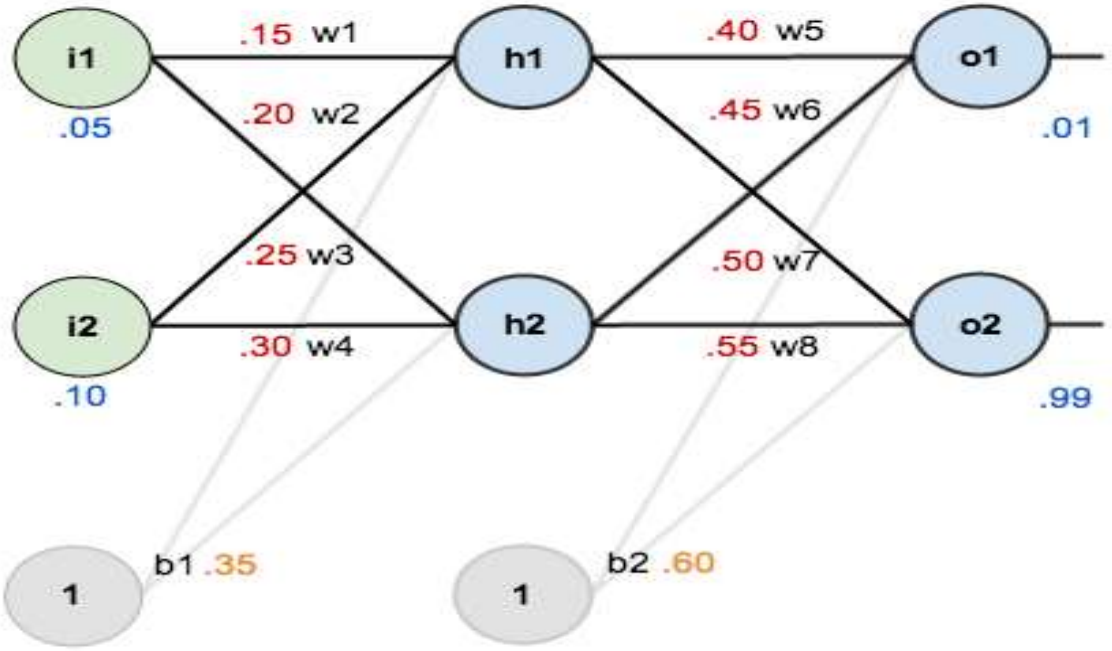
Step 6: Repeat Training

Now, you can repeat steps 2 to 5 (forward pass, compute loss, backpropagation, update weights and biases) for multiple iterations (epochs) until the loss converges to a minimum or reaches a satisfactory value.

Training a Neural Network with Backpropagation,

Question: 1

Optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs

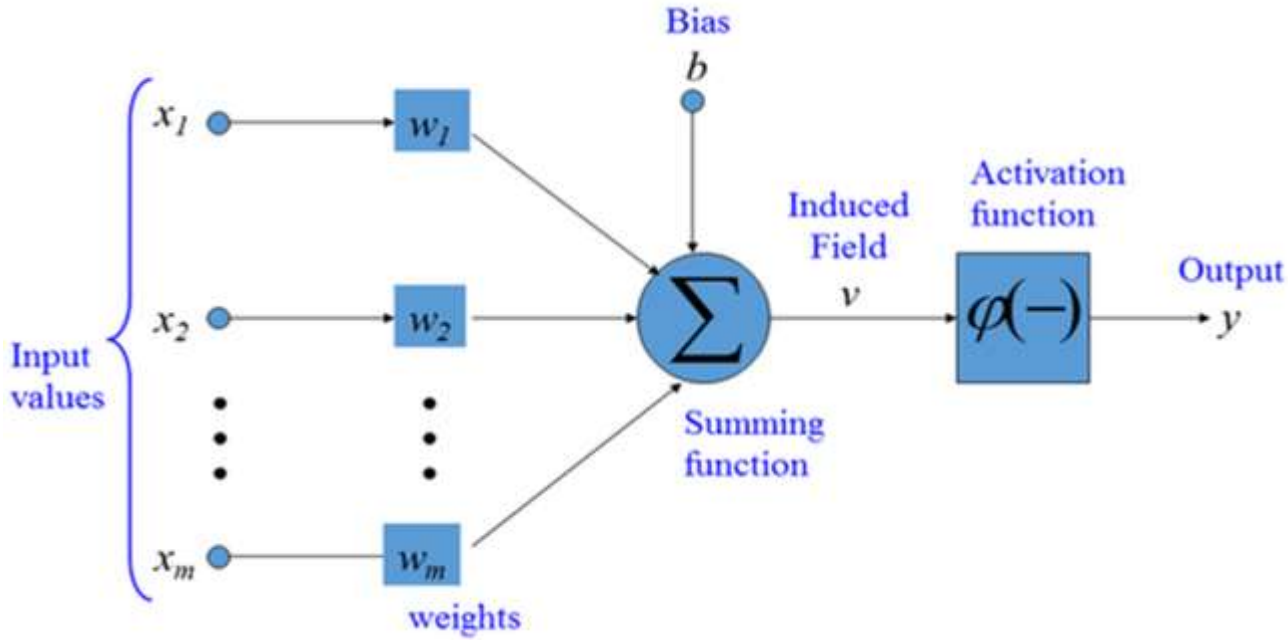


Reference solution *

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Knowledge Representation for ANN

1. Knowledge representation in artificial neural networks refers to the process of encoding information or knowledge within the network's architecture and parameters.
2. Neural networks are powerful machine learning models inspired by the structure and function of the human brain.
3. They are designed to learn patterns and relationships from data, enabling them to make predictions, classify inputs, and solve various tasks.



Knowledge Representation for ANN Steps

The key aspects of knowledge representation in artificial neural networks include:

- **Neurons (Nodes):** Neurons are fundamental units in a neural network. Each neuron receives input signals, processes them using an activation function, and produces an output. The activation function introduces non-linearity, allowing the network to learn complex relationships in the data.

- **Connections (Weights):** Neurons in a neural network are connected with each other via synapses, represented by connection weights. These weights determine the strength of the signal between neurons and are adjusted during the learning process to optimize the network's performance.

- **Layers:** Neurons are organized into layers, which are stacked one after the other. The typical neural network architecture consists of an input layer, one or more hidden layers, and an output layer. Deep learning models can have many hidden layers, hence the term "deep" neural networks.

Knowledge Representation for ANN Steps

- **Feedforward Propagation:** In a feedforward neural network, information flows in one direction, from the input layer through the hidden layers to the output layer. During this process, neurons in each layer perform computations based on the input data and learned weights, producing an output at the end.

- **Backpropagation:** Backpropagation is a key algorithm used to train neural networks. It involves calculating the error (difference between predicted output and actual output) and then propagating it backward through the network to adjust the connection weights. This process iterates multiple times during training until the network converges to a state where the error is minimized.

- **Activation Functions:** Activation functions introduce non-linearity to the neural network, allowing it to model complex relationships in the data. Common activation functions include ReLU (Rectified Linear Unit), sigmoid, tanh, and variants like Leaky ReLU and ELU.

Knowledge Representation for ANN Steps

- **Convolutional and Recurrent Architectures:** For specific tasks like image processing and natural language processing, specialized architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are used. CNNs excel at extracting spatial features from images, while RNNs are designed to handle sequential data with temporal dependencies.

- **Embeddings:** Embeddings are a way of representing categorical data or words as continuous vectors in a lower-dimensional space. Neural networks often learn these embeddings during the training process, which helps capture meaningful relationships between different categories or words.

Practical Issues in Neural Network Training

Practical Issues	Explanations
1. Data Quality and Quantity:	Neural networks require a significant amount of labeled training data to learn meaningful patterns and make accurate predictions.
2. Computational Resources:	Training deep neural networks can be computationally intensive and time-consuming, especially when dealing with large datasets and complex architectures. Training on powerful hardware, such as GPUs or TPUs, can help accelerate the process.
3. Choice of Architecture:	Selecting an appropriate neural network architecture for a specific task is critical. Different architectures have different strengths and weaknesses. Deciding on the number of layers, units per layer, and overall network structure requires domain knowledge and experimentation.

Practical Issues in Neural Network Training

Practical Issues	Explanations
4.Initialization of Weights:	Initializing the weights of a neural network can influence how effectively it converges during training. Poor weight initialization can lead to slow convergence or prevent the network from learning.
5.Hyperparameter Tuning:	Neural networks have various hyperparameters, such as learning rate, batch size, and architecture-specific parameters, that need to be carefully tuned for optimal performance. Finding the right combination of hyperparameters is often a trial-and-error process.
6.Overfitting:	Overfitting occurs when a neural network performs well on the training data but poorly on new, unseen data. This problem arises when the model becomes too complex or when the training data is limited. Regularization techniques and data augmentation are commonly used to mitigate overfitting.


GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU
NAAC A++ ACCREDITED

Thank You