

OVER
40
YEARS
OF ACADEMIC
WISDOM



PRESIDENCY UNIVERSITY

CSE3006 – NEURAL NETWORK **& FUZZY LOGIC**



Department of Computer Science Engineering
School of Engineering

Basics of Computing

$y = f(x)$, f is a mapping function
 f is also called a formal method or
an algorithm to solve a problem

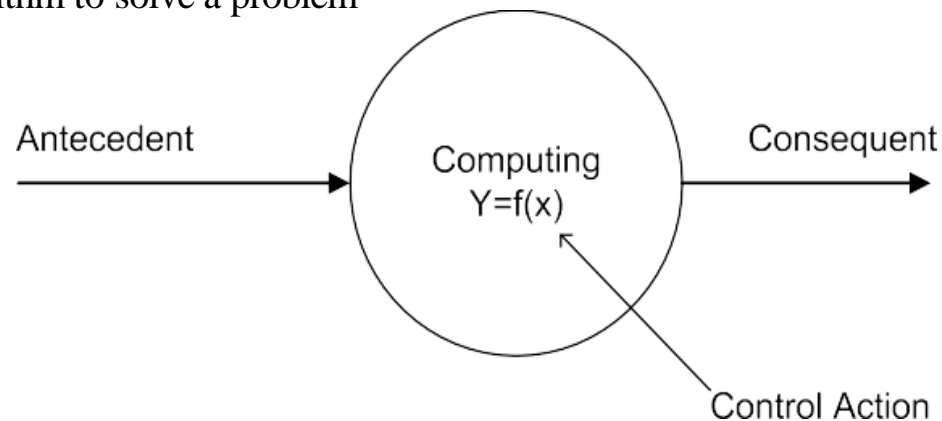
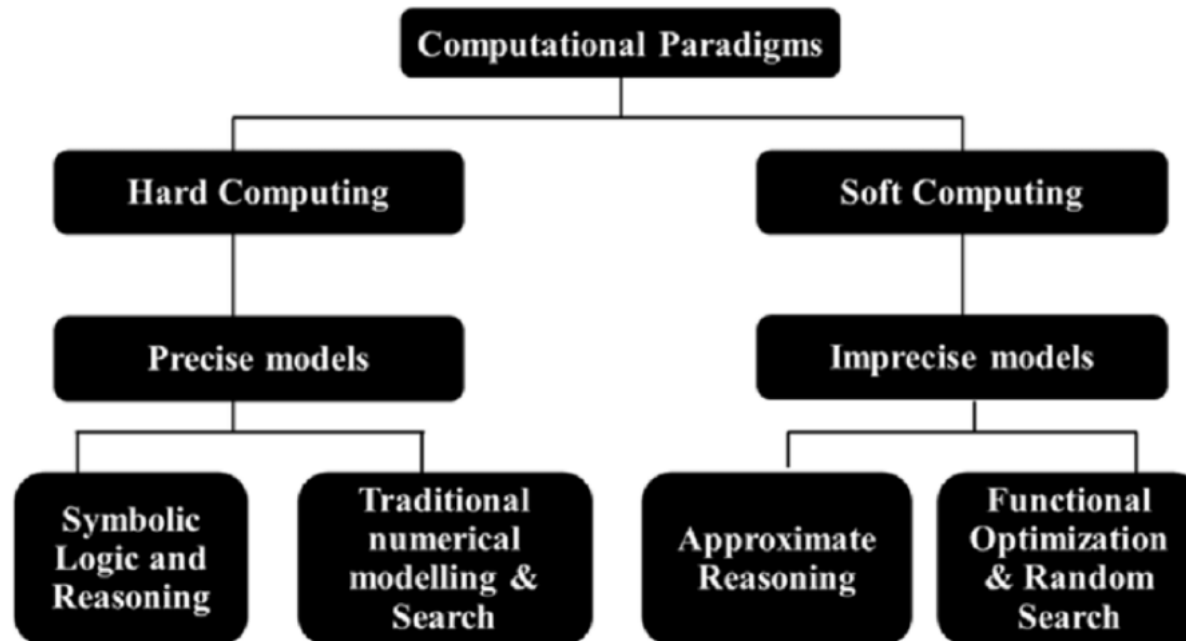


Figure 1: Basic of computing

Computational Paradigms



Computational Paradigms

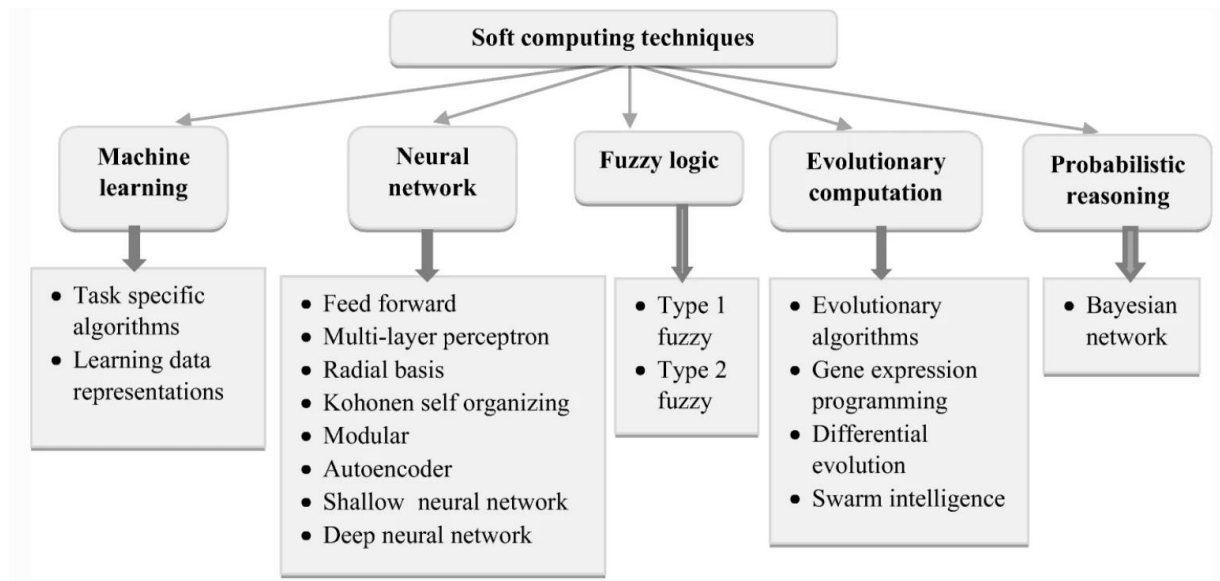
Attribute	Hard Computing	Soft Computing
Accuracy vs. Robustness	Accuracy mandatory	Robustness has priority
Logic	Binary logic	Multi-valued logic
Input data	Exact data required	Can tolerate imprecise data
Computation mode	Mostly Sequential	Supports parallelism
Precision of results	Precise answers	Approximate answers
Determinism	Deterministic	Non-deterministic



-
- Deals with the soft meaning of concepts.
 - According to Prof. Zadeh, soft computing is “an emerging approach to computing, which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision”.
 - Soft computing technologies are :
 - robust by design
 - operate by trading off precision for tractability
 - Since they can handle uncertainty with ease, they conform better to real-world situations and provide lower cost solutions.



Soft Computing



Neural Networks

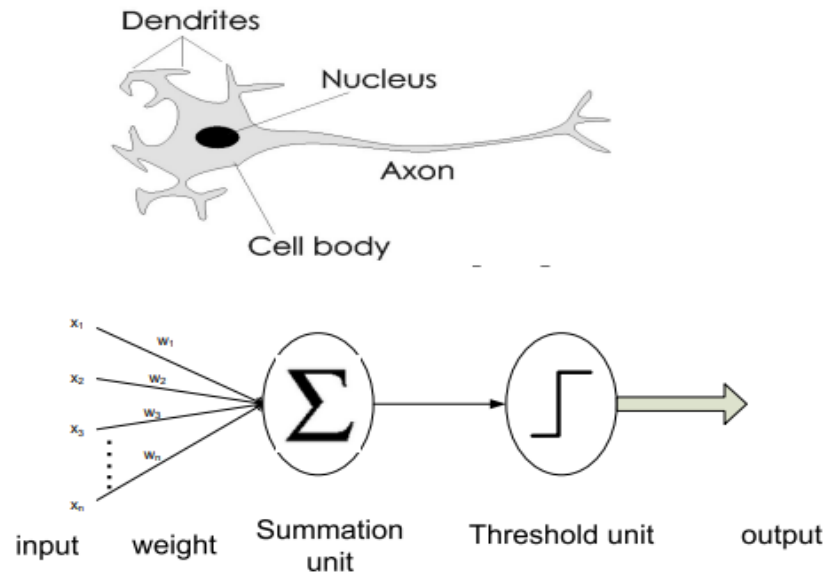
- The human brain is a highly complex structure viewed as a massive, highly interconnected network of simple processing elements called neurons.
- Artificial neural networks (ANNs) or simply we refer it as neural network (NNs), are simplified models (i.e. imitations) of the biological nervous system, and obviously, therefore, have been motivated by the kind of computing performed by the human brain.
- The behavior of a biological neural network can be captured by a simple model called artificial neural network.

Neural Networks

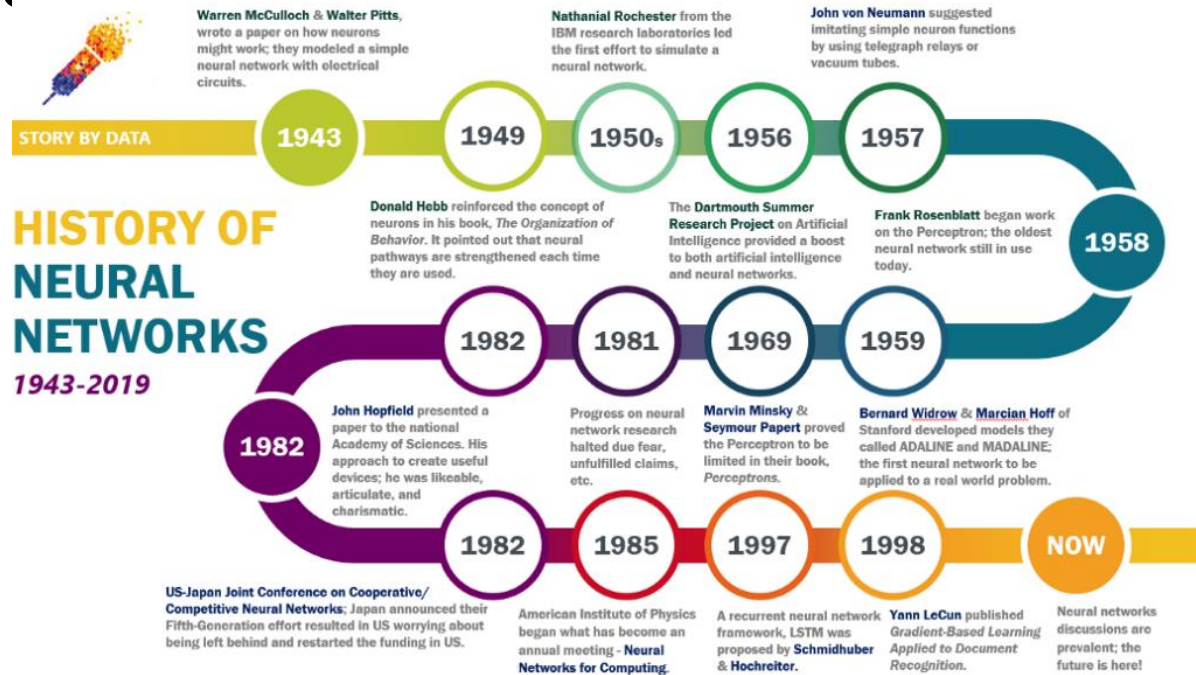
- A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two aspects:
- Knowledge is acquired by the network from its environment through a learning process.
- Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.



Analogy Between Neuron and an Artificial Neuron



History of Neural Networks



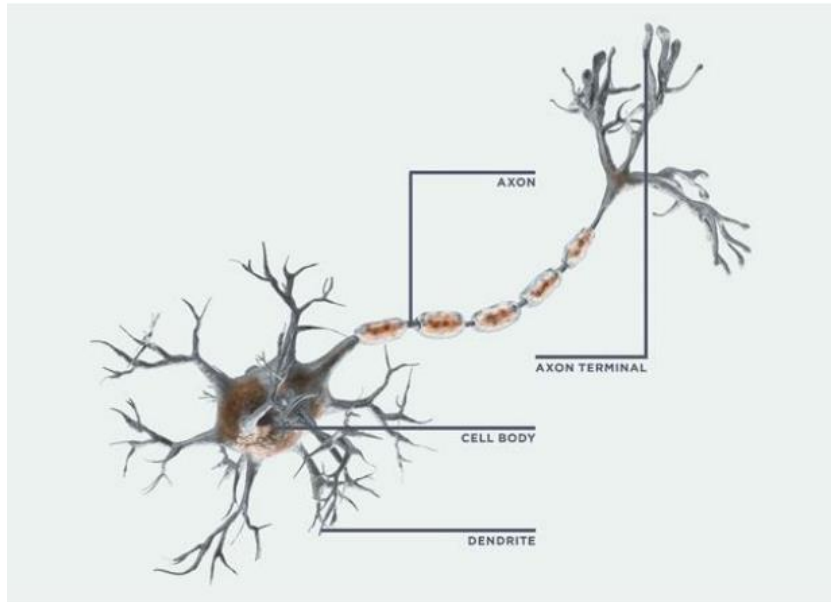
Benefits of Neural Networks

- Nonlinearity
- Input-Output Mapping
- Adaptivity
- Evidential Response
- Contextual Information
- Fault Tolerance
- VLSI Implementability
- Uniformity of Analysis and Design
- Neurobiological Analogy

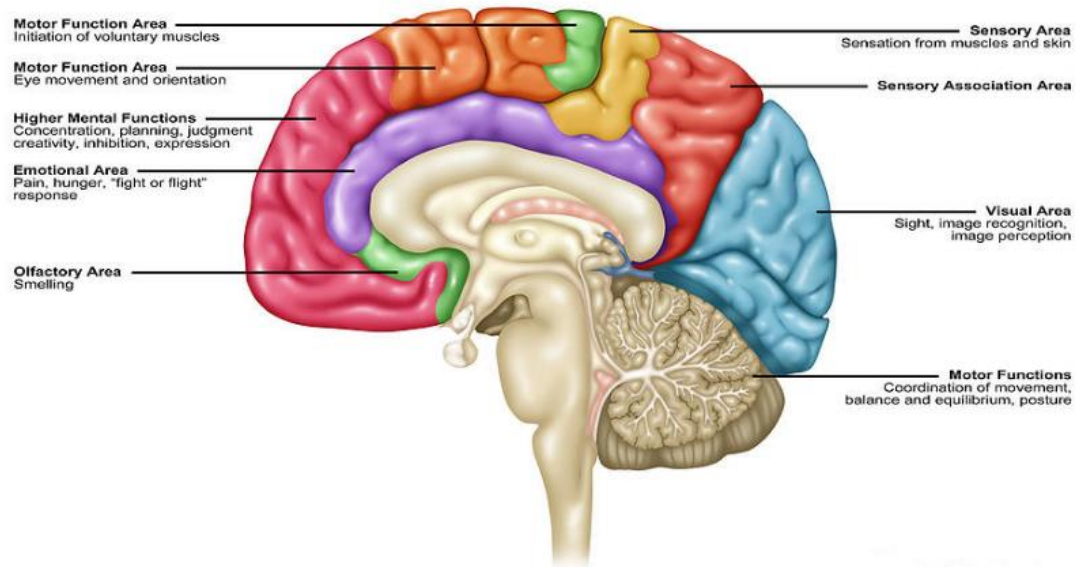


Biological Neurons

- Neurons are unique cells in the brain.
- They carry information throughout the body.
- Each component of the neuron is crucial.
- Dendrites, the "hairs" that surround the cell body, transmit data.
- The cell body integrates data from several dendrites.
- The message is carried by the long, tail-like axon.
- From the axon terminals, information "jump" to the next neuron.



Biological Neurons



Functional Areas of the Cerebral Cortex



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013





Characteristics of Biological Neural Network

- Many signals are received by the processing element.
- Weight at the receiving synapse can change the signal.
- The weighted inputs are added in the processing element.
- The neuron transmits a single output when the conditions are right (enough input).
- A neuron's output may be distributed to a large number of additional neurons (the axon branches).



Comparison between Human Brain and Neural Net

An Analogy of BNN and ANN

	
Biological Neurons	Silicon Transistors
200 Billion Neurons	Billion Bytes RAM
32 Trillion interconnections in Neurons	Trillions of Bytes on Disk
Neuron Size is 10^{-6} m	Single Transistor size is 10^{-9} m
Energy consumption is 6^{-10} joules per operation per second	Energy consumption is 10^{-16} joules per operation per second
Learning Capability	Programming Capability



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Comparison between Human Brain and Neural Net

Characteristics	Biological Neural Network	Artificial Neural Network
Speed	Processes information at a slower rate. Response time is measured in milliseconds.	Information is processed at a faster rate. The response time is measured in nanoseconds.
Processing	Massively parallel processing.	Serial processing.
Size & Complexity	An extremely intricate and dense network of linked neurons of the order of 10 ¹¹ neurons and 10 ¹⁵ interconnections.	Size and complexity are reduced. It is incapable of performing sophisticated pattern recognition tasks.
Storage	An extremely intricate and dense network of linked neurons with 10 ¹⁵ interconnections, including neurons on the order of 10 ¹¹ .	The term "replaceable information storage" refers to the practice of replacing fresh data with old data.
Fault tolerance	The fact that information storage is flexible means that new information may be added by altering the connectivity strengths without deleting existing information.	Intolerant of faults. In the event of a system failure, corrupt data cannot be recovered.
Control Mechanism	There is no unique control mechanism outside of the computational task.	Controlling computer activity is handled by a control unit.



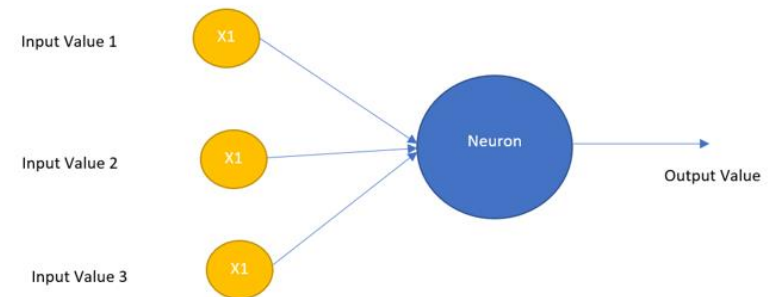
Artificial Neural Network

- A artificial neural network is an information-processing system.
- It has certain characteristics in common with biological neural networks.
- Its is a generalization of mathematical models of human cognition or neural biology, based on the following assumptions:
 - Information processing occurs with simple elements called neurons.
 - Signals are passed between neurons over connection links.
 - Each connection link has an associated weight, which in a typical neural net multiplies the signal transmitted.
 - Each neuron applies an activation function to its net input to determine its output signal.

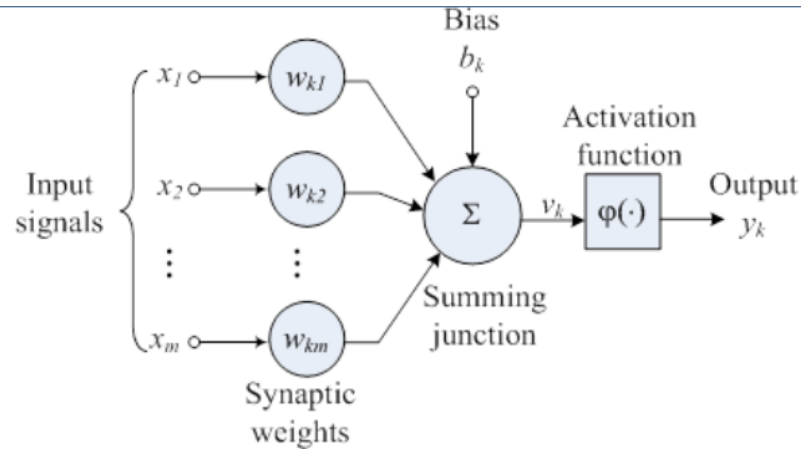


Artificial Neural Network

- The Perceptron/ Rosenblatt's Perceptron in the example contains three inputs, x_1 , x_2 , and x_3 .
- It might have more or fewer inputs in general.
- To compute the outcome, Rosenblatt developed a simple rule.
- He created weights, w_1, w_2, \dots , which are actual values that reflect the significance of the various inputs to the output.
- The neuron's output, which is either 0 or 1, is decided by whether the weighted sum $\sum w_j x_j$ is less than or larger than some threshold number.
- A threshold, like weights, is a real number that serves as a neuronal parameter.



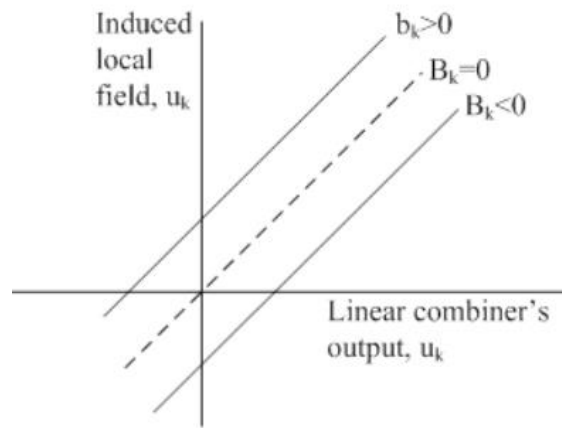
Nonlinear Model of Neuron



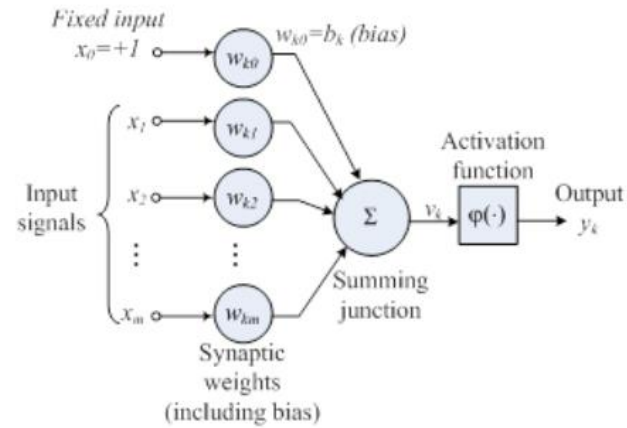
$$u_k = \sum_{j=1}^m w_{kj} x_j \quad v_k = u_k + b_k \quad y_k = \phi(v_k) = \phi(u_k + b_k)$$

$$\text{Let } b_k = w_{k0} \text{ and } x_0 = +1 \quad v_k = \sum_{j=0}^m w_{kj} x_j \quad \text{and} \quad y_k = \phi(v_k)$$

Nonlinear Model of Neuron contd...



Affine transformation produced by the presence of a bias



Another Nonlinear model of a neuron

Idealized neurons

- To model things we have to idealize them (e.g. atoms)
 - Idealization removes complicated details that are not essential for understanding the main principles.
 - It allows us to apply mathematics and to make analogies to other, familiar systems.
 - Once we understand the basic principles, its easy to add complexity to make the model more faithful.



Characteristics of ANN

- With characteristics of Neural Networks, the capacity to solve problems in a human-like manner and to apply that ability to large datasets neural networks have the following characteristics –
 - Adaptive Learning: Neural networks, like humans, represent non-linear and complicated interactions and build on prior knowledge through adaptive learning. Software that teaches arithmetic and language arts, for example, employs adaptive learning.
 - Fault Tolerance: Neural networks can fill in the gaps when important portions of a network are lost or absent. This skill is particularly valuable in space travel, where electronic gadget failure is a constant risk.
 - Prognosis: The capacity of neural networks to forecast based on models has several uses, including weather and transportation.
 - Real-Time Response: As with self-driving vehicles and drone navigation, neural networks may give real-time responses.
 - Self-Organization: Neural networks are especially suited for organising the complex visual issues posed by medical image analysis due to their capacity to cluster and categorise large volumes of data.



Features of ANN

- Features of Neural Network Neural Networks are used to accomplish a variety of tasks –
 - Associating: Pattern recognition may be taught to neural networks. When you present an unknown version of a pattern, the network correlates it with the most similar version in its memory and switches to that version.
 - Classification: Neural Networks classify patterns or information into predetermined categories.
 - Clustering: They categorize the data by identifying a unique characteristic without having any prior knowledge of the data.
 - Prediction: They get the anticipated result from the provided input



Limitations of ANN

- To function, the neural network requires training.
- A neural network's architecture differs from that of a microprocessor. Emulation is so required.
- Large neural networks need a lot of processing time.



Neural Network Architectures

- An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.
- Learning largely involves adjustments to the synaptic connections that exist between the neurons.
- The model of an artificial neural network can be specified by three entities:
- **Activation functions**
- **Interconnections**
- **Learning rules**



Activation Functions

- Activation functions are functions used in a neural network to compute the weighted sum of inputs and biases.
- It is 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*.



Benefit of Activation Functions

- Converts the linear input signals and models into non-linear output signals.
- This helps to learn high-order polynomials for deeper networks.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Types of Activation Functions

- Linear Activation Function
- Non-Linear Activation Function



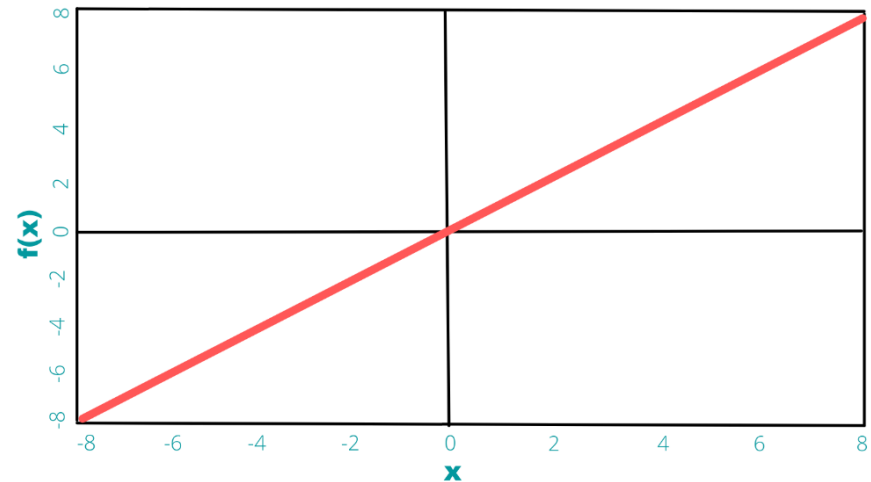
**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Linear Activation Function

- The linear activation function, also known as "no activation," or "identity function" (multiplied $\times 1.0$), is where the activation is proportional to the input.
- The function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given.
- It has the equation:
$$f(x) = kx, \text{ where } k \text{ is a constant.}$$



Linear Activation Function

Disadvantage of Linear Activation Function

- It's not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x .
- All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.



Non-Linear Activation Function

- Most modern neural network uses the non-linear function as their activation function to fire the neuron.
- They allow the model to create complex mappings between the network's inputs and outputs, which are essential for learning and modelling complex data
- They are helpful in datasets which are non-linear or have high dimensionality such as images, video, audio, etc.



Advantage of Non-linear function over the Linear function

- Differential is possible in all the non-linear functions.
- Stacking of networks is possible, which helps us in creating deep neural nets.
- It makes it easy for the model to generalize or adapt with a variety of data and to differentiate between the output.
- They allow backpropagation because now the derivative function would be related to the input, and it's possible to go back and understand which weights in the input neurons can provide a better prediction.
- They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

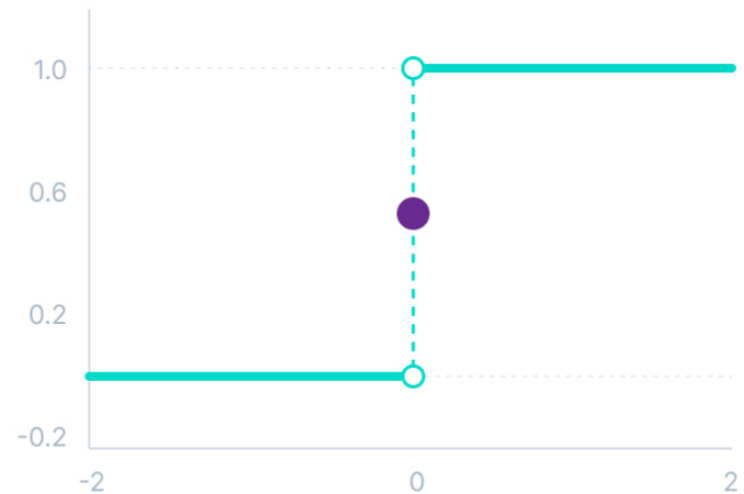
Important terminologies for Non-linear Activation function

- *Derivative or Differential:* Change in y-axis w.r.t. change in the x-axis. It is also known as a slope.
- *Monotonic function:* A function that is either entirely non-increasing or non-decreasing.



Binary Step Function or Heaveside Function or Unit Step Function

- Binary step 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,
- Its output is not passed on to the next hidden layer.



Binary Step Function

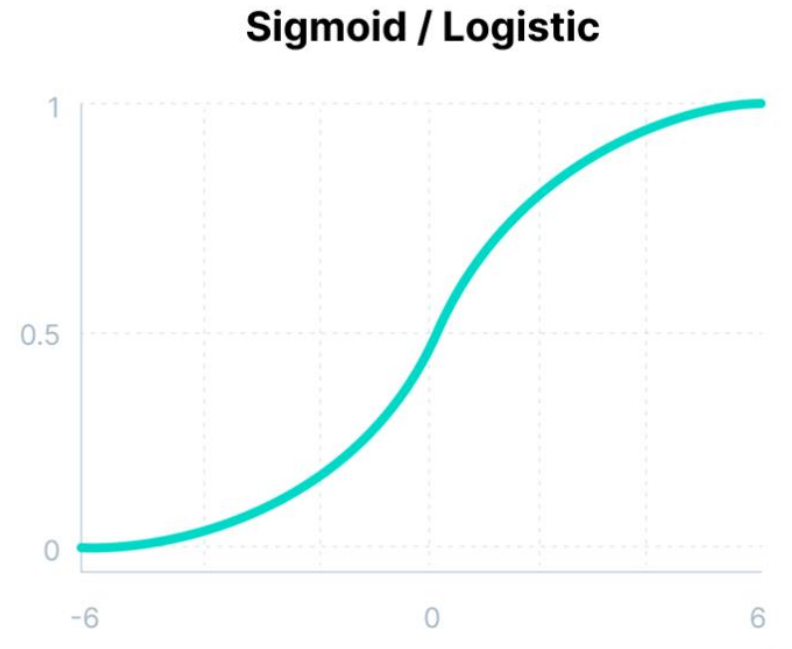
- Mathematically it can be represented using the following equation:

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

- Limitations:
- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

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 Activation Function

- Mathematically it can be represented as follows:

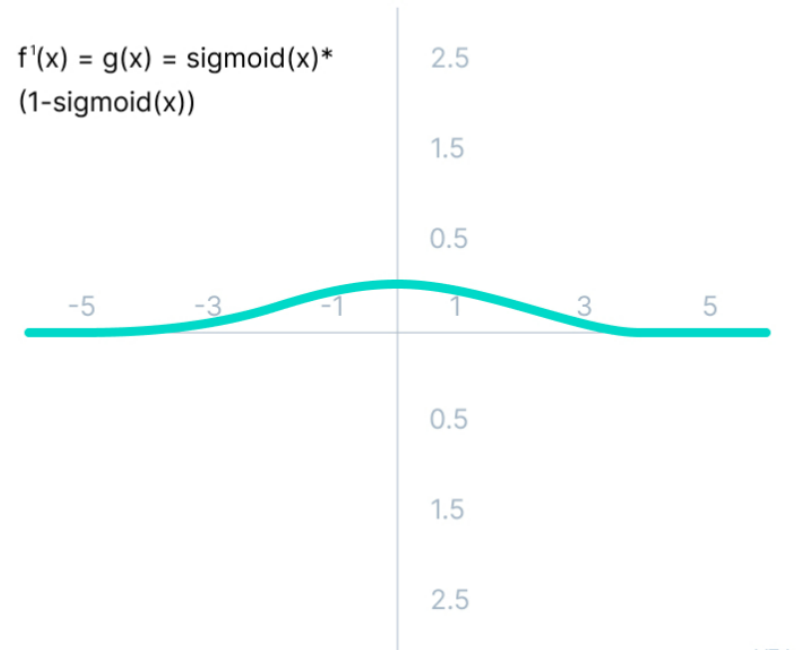
$$f(x) = \frac{1}{1 + e^{-x}}$$

- Advantages of Sigmoid Activation Function:
- It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.



Sigmoid/ Logistic Activation Function

- Limitations of Sigmoid Activation Function:
- The derivative of the function is $f'(x) = \text{sigmoid}(x) * (1 - \text{sigmoid}(x))$.



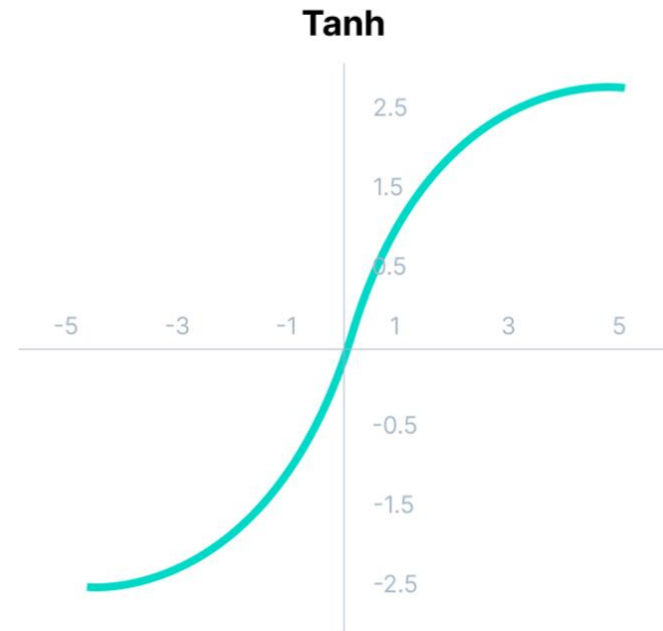
Sigmoid/ Logistic Activation Function

- The gradient values are only significant for the range -3 to 3, and the graph gets much flatter in other regions.
- It implies that for values greater than 3 or less than -3, the function will have very small gradients.
- As the gradient value approaches zero, the network ceases to learn and suffers from the *Vanishing gradient* problem.
- The output of the logistic function is not symmetric around zero.
- So the output of all the neurons will be of the same sign.
- This makes the training of the neural network more difficult and unstable.



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 the 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 Function (Hyperbolic Tangent)

- Mathematically it can be represented as follows:

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

- Advantages :
- The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive.
- Usually used in hidden layers of a neural network as its values lie between -1 to; therefore, the mean for the hidden layer comes out to be 0 or very close to it. It helps in centering the data and makes learning for the next layer much easier.

Limitations of tanh Functions

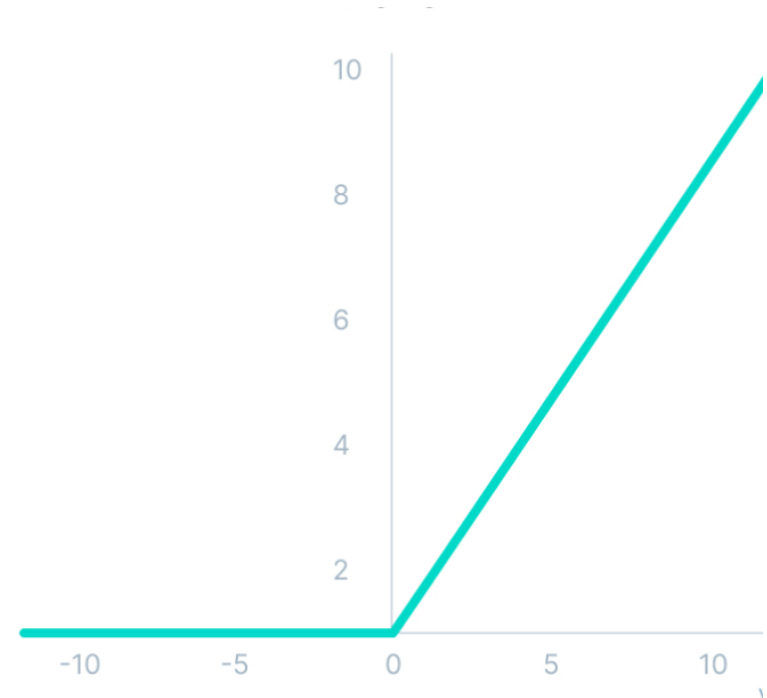
- Tanh also faces the problem of vanishing gradients similar to the sigmoid activation function.
- The gradient of the tanh function is much steeper as compared to the sigmoid function.



Fig: Gradient of the tanh activation function

ReLU Functions

- ReLU stands for Rectified Linear Unit.
- Although it gives an impression of a linear function, ReLU is a derivative function.
- It allows backpropagation and makes the process computationally efficient.



ReLU activation function

ReLU Functions

- ReLU function does not activate all the neurons at the same time.
- The neurons are only be deactivated if the output of the linear transformation is less than 0.
- Mathematically it can be represented as follows:

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

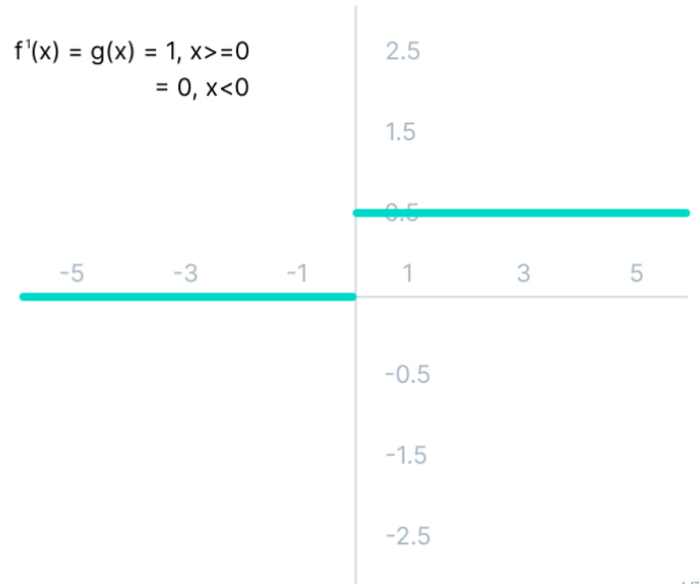
ReLU Functions

- The advantages of using ReLU as an activation function are as follows:
- As a certain number of neurons are activated, it is far more computationally efficient when compared to the sigmoid and tanh functions.
- ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear, non-saturating property.

ReLU Functions

- The limitations faced by this function are:
- The Dying ReLU problem: The negative side of the graph makes the gradient value zero. Due to this reason, during the backpropagation process, the weights and biases for some neurons are not updated. This can create dead neurons which never get activated.
- All the negative input values become zero immediately, which decreases the model's ability to fit or train from the data properly.

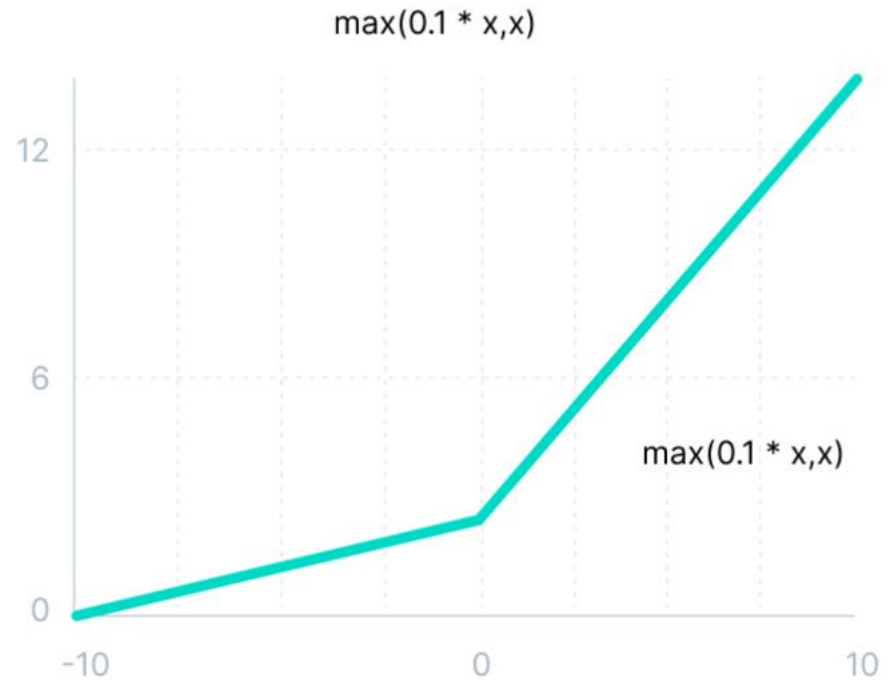
$$f'(x) = g(x) = 1, x \geq 0 \\ = 0, x < 0$$



v7

Leaky ReLU Functions

- Leaky ReLU is an improved version of ReLU function to solve the Dying ReLU problem as it has a small positive slope in the negative area.



Leaky ReLU Functions

- Mathematically it can be represented as follows:

$$f(x) = \max(0.1x, x)$$

- The advantages of Leaky ReLU are same as that of ReLU.
- Additionally it enables backpropagation, even for negative input values.

By making this minor modification for negative input values, the gradient of the left side of the graph comes out to be a non-zero value. Therefore, dead neuron problems is resolved for negative inputs.

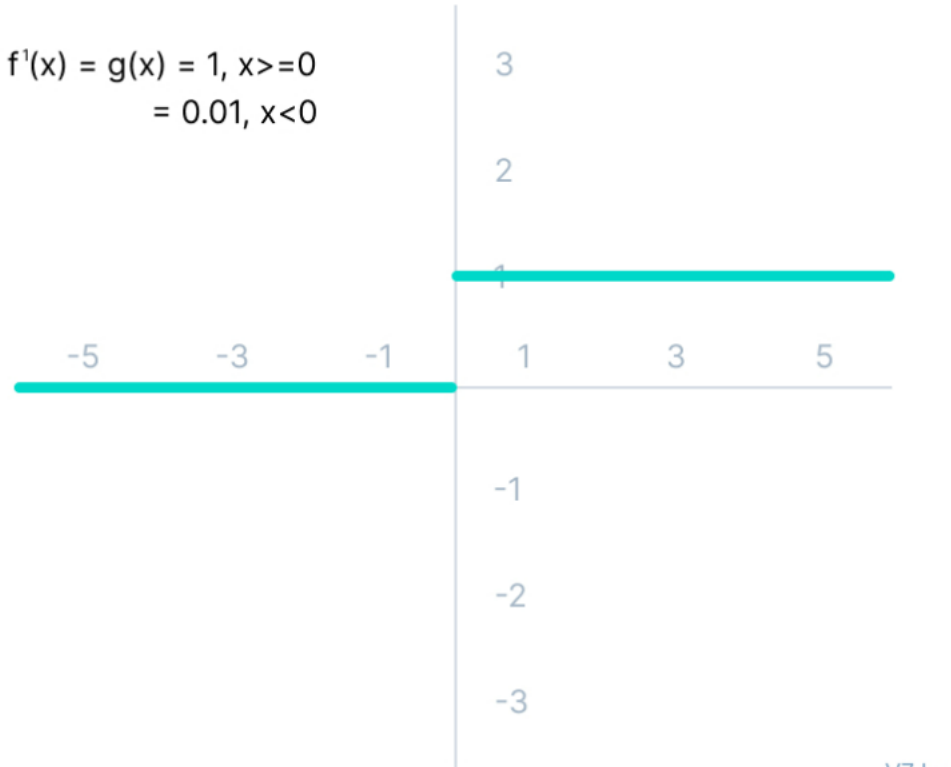
Leaky ReLU Functions

The limitations that this function faces include:

- The predictions may not be consistent for negative input values.
- The gradient for negative values is a small value that makes the learning of model parameters time-consuming.

Leaky ReLU (derivative)

$$f'(x) = g(x) = 1, x \geq 0 \\ = 0.01, x < 0$$



Interconnections in Neural Network

- Interconnection is the way neurons are connected to each other.
- The arrangements of these processing elements and geometry of interconnections are very essential in ANN.
- They have two common layers in all network architectures, viz., the Input layer and output layer.
- The input layer buffers the input signal.
- The output layer generates the output of the network.
- The third layer is the Hidden layer, in which neurons are neither kept in the input layer nor in the output layer.
- These neurons are hidden from the people who are interfacing with the system and act as a black box to them.
- By increasing the hidden layers with neurons, the system's computational and processing power can be increased but the training phenomena of the system get more complex at the same time.



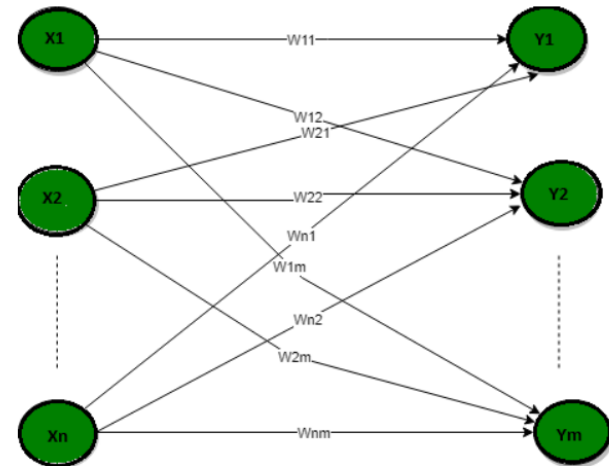
Neural Network Architecture

- There are five basic types of neuron connection/architectures :
 - Single-layer feed-forward network
 - Multilayer feed-forward network
 - Single node with its own feedback
 - Single-layer recurrent network
 - Multilayer recurrent network



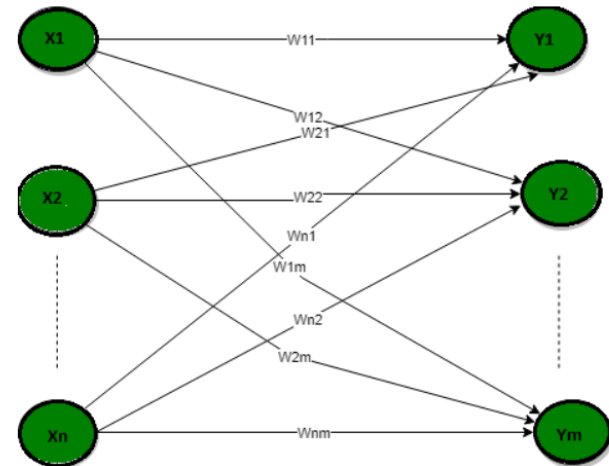
Single-layer feed-forward network

- In this type of network, only two layers input layer and the output layer are present.
- The input layer does not count because no computation is performed in this layer.
- The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken.



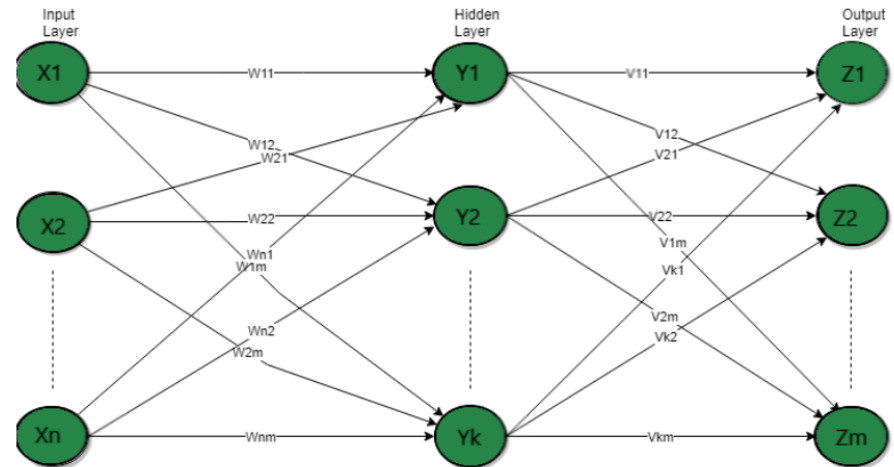
Single-layer feed-forward network

- The neurons collectively give the output layer to compute the output signals.
- In other words, this network is strictly of a feedforward type.
- It is called a single-layer network, with the designation “single-layer” referring to the output layer of computation nodes (neurons)



Multilayer feed-forward network

- This feedforward neural network has one or more *hidden layers*.
- These computation nodes are correspondingly called *hidden neurons* or *hidden units*.
- The term “hidden” refers to the fact that this part of the neural network is not seen directly from either the input or output of the network.
- The function of hidden neurons is to intervene between the external input and the network output.

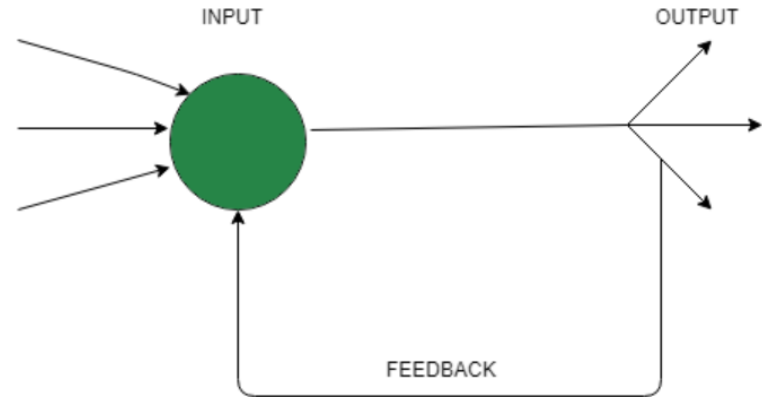


Multilayer feed-forward network

- By adding one or more hidden layers, the network is enabled to extract higher-order statistics from its input.
- The network acquires a *global* perspective despite its local connectivity, due to the extra set of synaptic connections and the extra dimension of neural interactions.
- The source nodes in the input layer of the network supply respective elements of the activation pattern (input vector), which constitute the input signals applied to the neurons (computation nodes) in the second layer (i.e., the first hidden layer).
- The output signals of the second layer are used as inputs to the third layer, and so on for the rest of the network.
- Typically, the neurons in each layer of the network have as their inputs the output signals of the preceding layer only.
- The set of output signals of the neurons in the output (final) layer of the network constitutes the overall response of the network to the activation pattern supplied by the source nodes in the input (first) layer.

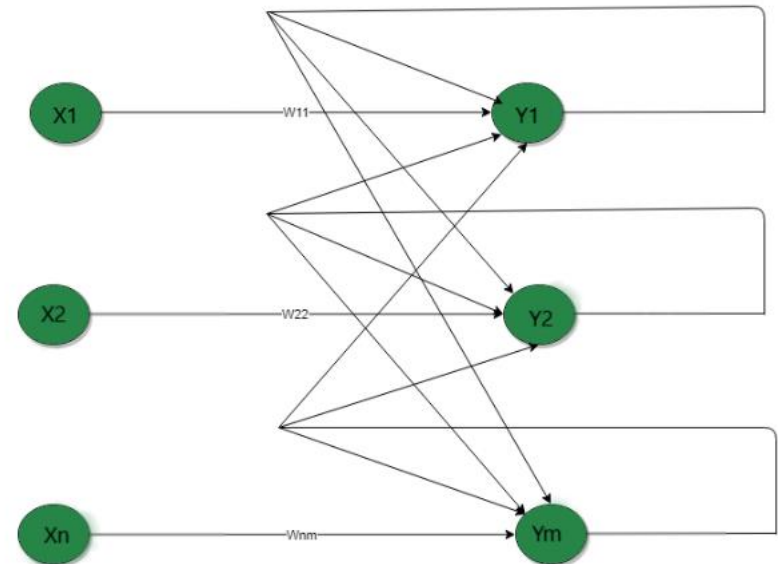
Single node with its own feedback

- The networks where the output layer output is sent back as an input to the input layer or the other hidden layers is called Feedback Networks.
- In single-node feedback systems, there is a single input layer where the output is redirected back as feedback.



Single-layer recurrent network

- In a single-layer recurrent network, the feedback network forms a closed loop.
- A single neuron receives feedback to itself or the other neurons in the network or both.
- The feedback network forms a closed loop.



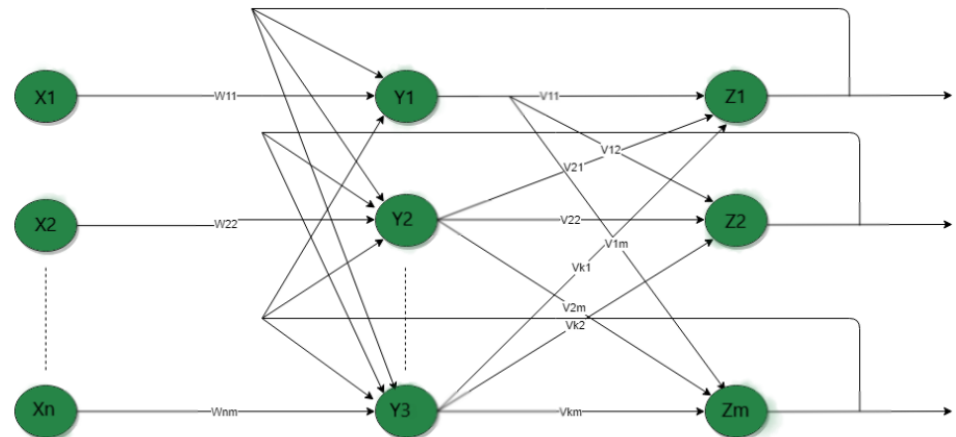
Single-layer recurrent network

- Here, the processing element's output can be directed back to itself or to another processing element or both.
- A recurrent neural network is a class of artificial neural networks where connections between nodes form a directed graph along a sequence.
- This allows it to exhibit dynamic temporal behavior for a time sequence.
- Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.



Multilayer recurrent network

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



Knowledge

- Knowledge refers to stored information or models used by a person or machine to interpret, predict, and appropriately respond to the outside world.
- The primary characteristics of *knowledge representation* are :
 - what information is actually made explicit
 - how the information is physically encoded for subsequent use.
- Knowledge representation is goal directed.
- In real-world applications of “intelligent” machines, it can be said that a good solution depends on a good representation of knowledge.
- We find that the possible forms of representation from the inputs to internal network parameters are highly diverse, which tends to make the development of a satisfactory solution by means of a neural network a real design challenge.



Knowledge

- Knowledge of the world consists of two kinds of information:
 - The known world state, represented by facts about what is and what has been known, this form of knowledge is referred to as *prior information*.
 - Observations (measurements) of the world, obtained by means of sensors designed to probe the environment, in which the neural network is supposed to operate.



Representing Information

- **Object:** All the facts about objects in our world domain. E.g., Guitars contains strings, trumpets are brass instruments.
- **Events:** Events are the actions which occur in our world.
- **Performance:** It describe behavior which involves knowledge about how to do things.
- **Meta-knowledge:** It is knowledge about what we know.
- **Facts:** Facts are the truths about the real world and what we represent.
- **Knowledge-Base:** The central component of the knowledge-based agents is the knowledge base. It is represented as KB. The Knowledge base is a group of the Sentences (Here, sentences are used as a technical term and not identical with the English language).

Types of Knowledge



Types of Knowledge

1. Declarative Knowledge:

- Declarative knowledge is to know about something.
- It includes concepts, facts, and objects.
- It is also called descriptive knowledge and expressed in declarative sentences.
- It is simpler than procedural language.

2. Procedural Knowledge

- It is also known as imperative knowledge.
- Procedural knowledge is a type of knowledge which is responsible for knowing how to do something.
- It can be directly applied to any task.
- It includes rules, strategies, procedures, agendas, etc.
- Procedural knowledge depends on the task on which it can be applied.



Types of Knowledge

3. Meta-knowledge:

Knowledge about the other types of knowledge is called Meta-knowledge.

4. Heuristic knowledge:

Heuristic knowledge is representing knowledge of some experts in a filed or subject.

Heuristic knowledge is rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

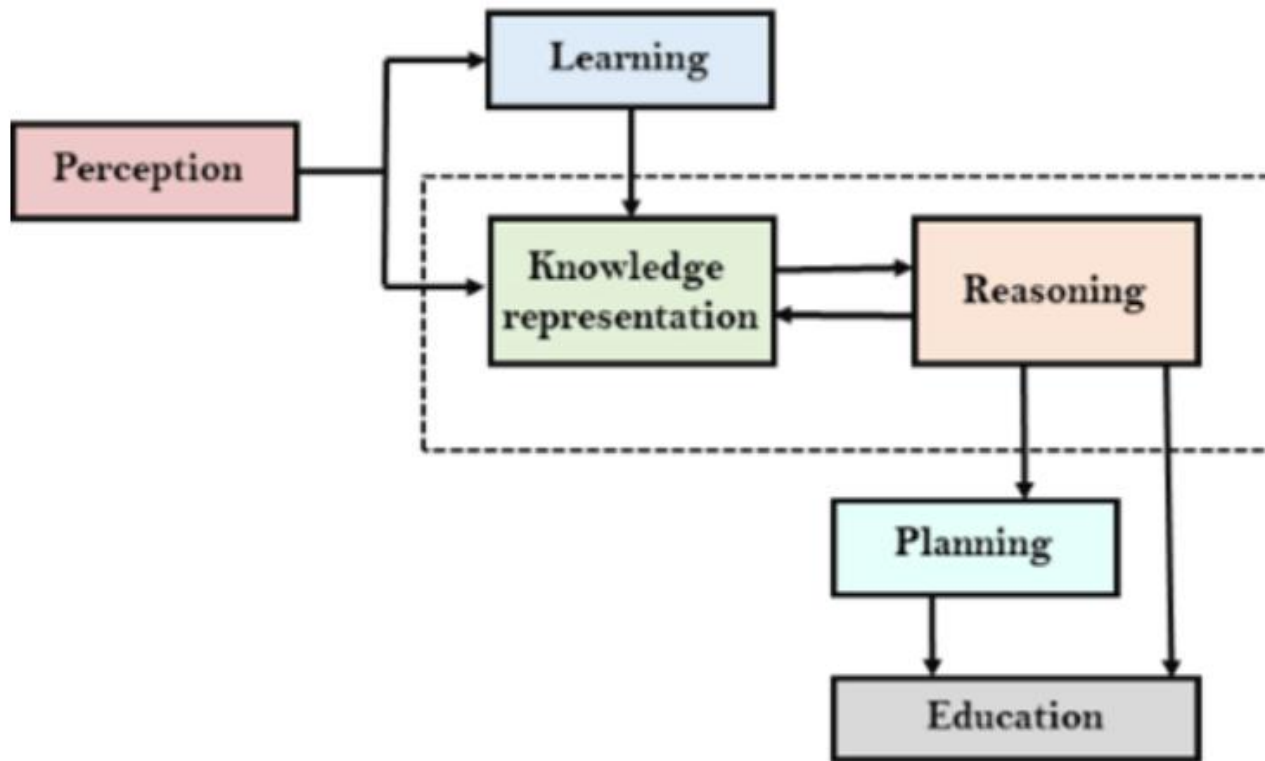
5. Structural knowledge:

Structural knowledge is basic knowledge to problem-solving.

It describes relationships between various concepts such as kind of, part of, and grouping of something.

It describes the relationship that exists between concepts or objects.

AI Knowledge Cycle



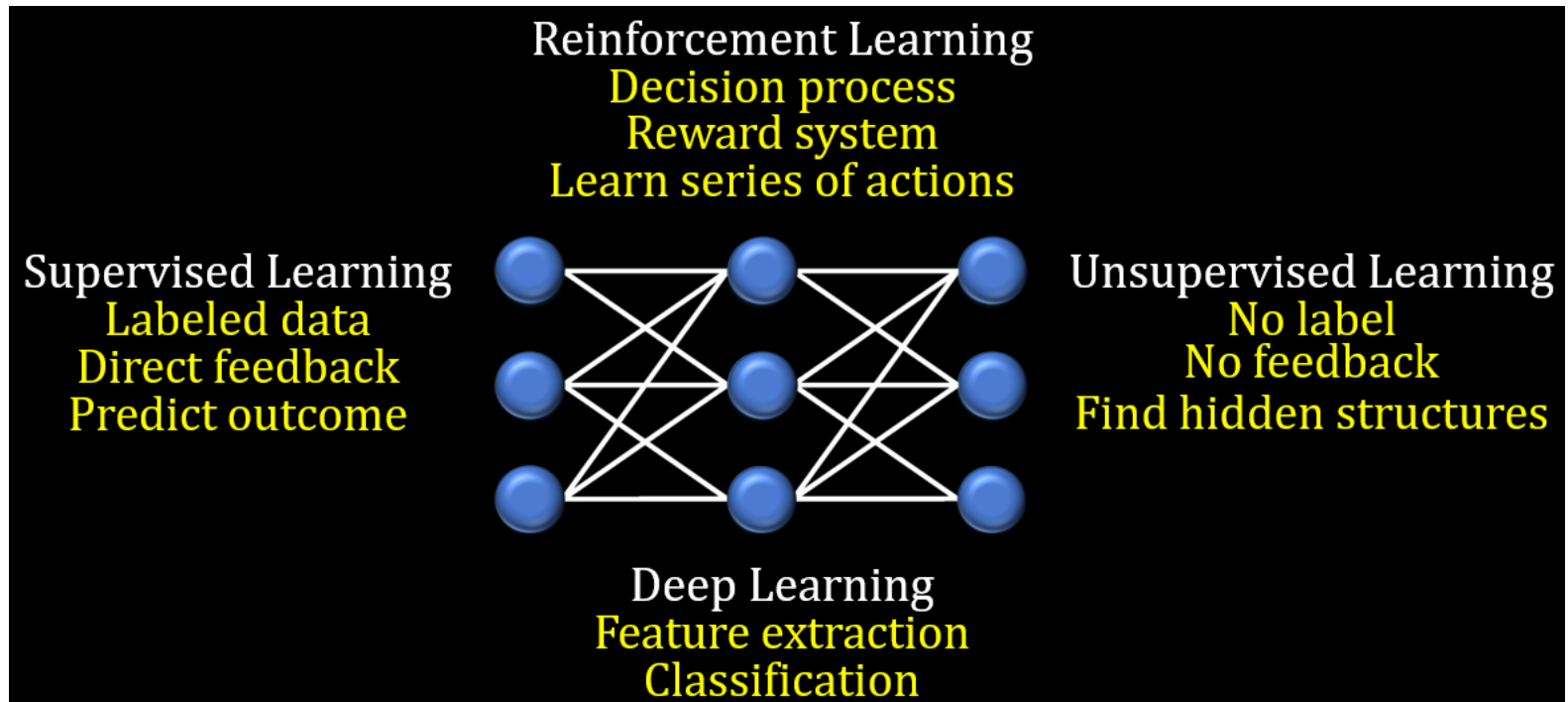
Rules of Knowledge Representation

- **Rule 1:** Similar inputs (i.e., patterns drawn) from similar classes should usually produce similar representations inside the network, and should therefore be classified as belonging to the same class.
- **Rule 2:** Items to be categorized as separate classes should be given widely different representations in the network.
- **Rule 3:** If a particular feature is important, then there should be a large number of neurons involved in the representation of that item in the network.
- **Rule 4:** Prior information and invariances should be built into the design of a neural network whenever they are available, so as to simplify the network design by its not having to learn them.

Learning in Neural Networks

- Neural networks learn from its environment and improve its performance.
- A neural network learns its environment through an interactive process of adjustments applied to its synaptic weights and bias.
- We define learning in the context of neural networks as:
Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded.
- The type of learning is determined by how the parameter changes take place.

Types of Learning



Learning Algorithm

- Since the single layer perceptron is a supervised neural network architecture, it requires learning of some a priori knowledge base for its operation.
- This learning paradigm for a single layer perceptron can be summarized as follows:
 - Set the weights and thresholds randomly.
 - Calculate the actual outputs by taking the thresholded value of the weighted sum of the inputs.
 - Alter the weights to reinforce correct decisions and discourage incorrect decisions, i.e. reduce the error.



Learning Algorithm

Algorithm 1 Learning algorithm of a single layer perceptron [6]

- 1 Begin
 - 2 Define $w_i(t)$, ($0 \leq i \leq n$), to be the weight from input i at time t
 - 3 Set w_0 to be k , the bias in the output node
 - 4 Set input $x_0 = 1$
 - 5 Set $w_i(0)$ to small random values
Remark: Initialize weights and threshold
 - 6 Present inputs $x_0, x_1, x_2, x_3, \dots, x_n$
 - 7 Present desired output $d(t)$
Remark: Present input and desired output
 - 8 $y(t) = f(\sum_{i=1}^n w_i(t)x_i(t))$
Remark: Calculate the actual output
 - 9 $w_i(t+1) = w_i(t) \pm x_i(t)$
Remark: Adapt weights
 - 10 End
-

Learning Algorithm

- The weights are unchanged if the net makes the correct decision.
- The weights are also not adjusted on input lines which do not contribute to the incorrect response, since each weight is adjusted by the value of the input on that line, x_i , which would be zero.
- In order to predict the expected outputs, a loss (also called objective or error) function E can be defined over the model parameters to ascertain the error in the prediction process.
- A popular choice for E is the sum-squared error, given by

$$E = \sum_p (y_p - d_p)^2$$

- It is the sum over all points i in the data set of the squared difference between the target value d_i and the perceptron's prediction y_i , calculated from the input value x_i .
- For a linear model, the sum-squared error is a quadratic function of the model parameters.

Learning Algorithm

- The loss function E provides an objective measure of predictive error for a specific choice of perceptron model parameters.
- Minimizing the loss function would yield more accurate predicted outputs by the perceptron.

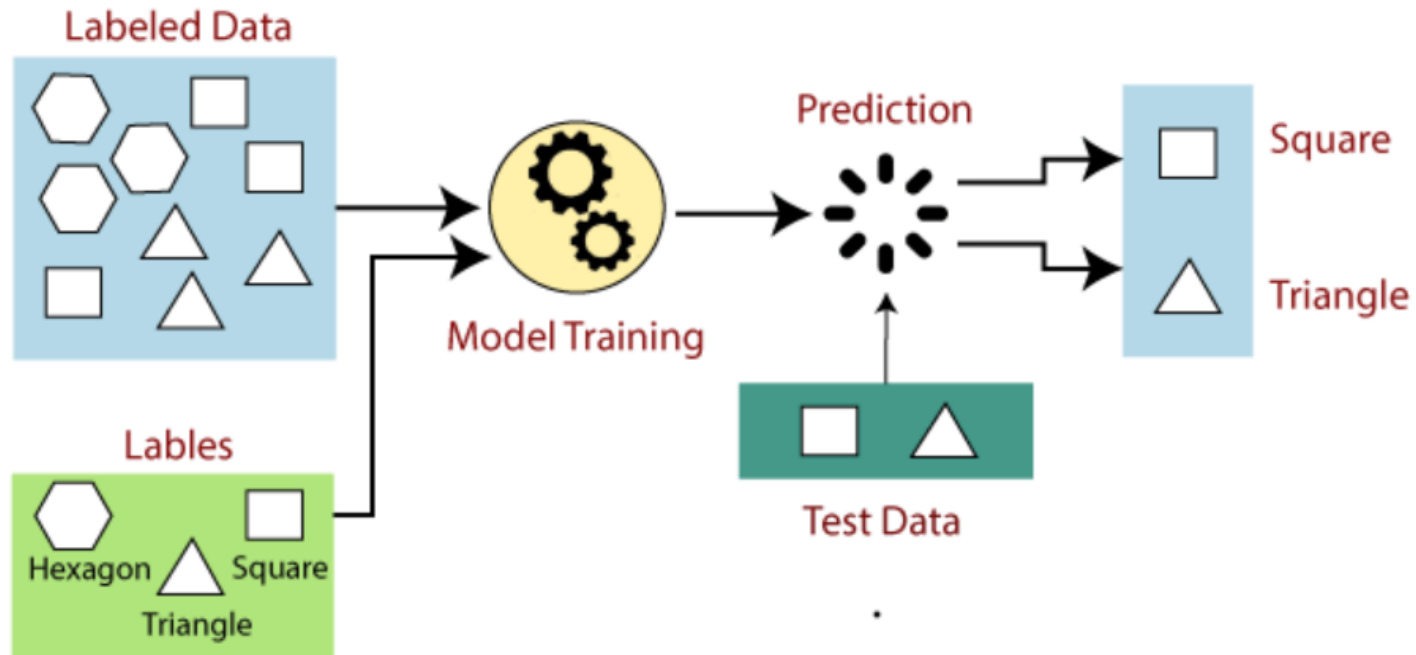


Supervised Learning

- A neural network is trained with the help of a set of patterns known as the training vectors. The outputs for these vectors might be, or might not be, known beforehand.
- When these are known, and that knowledge is employed in the training process, the training is termed as *supervised* learning.
- Some popular supervised learning methods are perceptron learning, delta learning, least-mean-square (LMS) learning, correlation learning, outstar learning etc.



Supervised Learning



Hebb Rule

- Hebb rule is one of the earliest learning rules for ANNs.
- According to this rule the weight adjustment is computed as

$$\Delta w_i = x_i \times t$$

- where t is the target activation.
- Important facts regarding Hebb learning rule:
 - Hebb rule cannot learn when the target is 0, because the weight adjustment becomes zero when $t = 0$, irrespective of the input value.
 - the Hebb rule results in better learning if the input / output both are in bipolar form.
 - Hebb rule does not guarantee to learn a classification instance even if the classes are linearly separable.



Hebb Rule- *Realizing the logical AND function through Hebb learning*

- To realize a two-input *AND* function we need a net with two input units and one output unit. A bias is also needed. Hence the structure of the required neural net should be as shown in Fig. 2. Moreover, the input and output signals must be in *bipolar* form, rather than the *binary* form, so that the net may be trained properly. Considering the truth table of *AND* operation, and the fact that the bias is permanently set to 1, we get the training set depicted in Table 1.

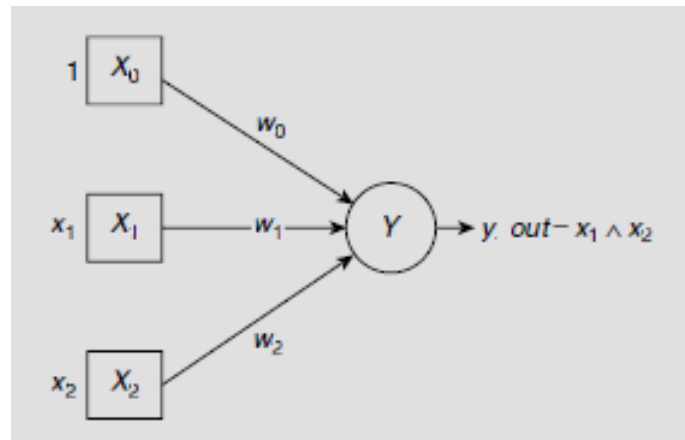


Fig 1.

Delta/ LMS (Least Mean Square) Rule

- Least Mean Square (*LMS*), also referred to as the Delta, or Widrow-Hoff Rule, is another widely used learning rule in ANN.
- Here the weight adjustment is computed as:

$$\Delta w_i = \eta \times (t - y_{in}) \times x_i$$

- In *LMS* learning, the identity function is used as the activation function during the training phase.
- The learning rule minimizes mean squared error between the activation and the target value.
- The output of *LMS* learning is in binary form.



Perceptron Learning Rule

- The perceptron learning rule is informally stated as :
- IF *the output is erroneous* THEN *adjust the interconnection weights* ELSE *leave the interconnection weights unchanged*.

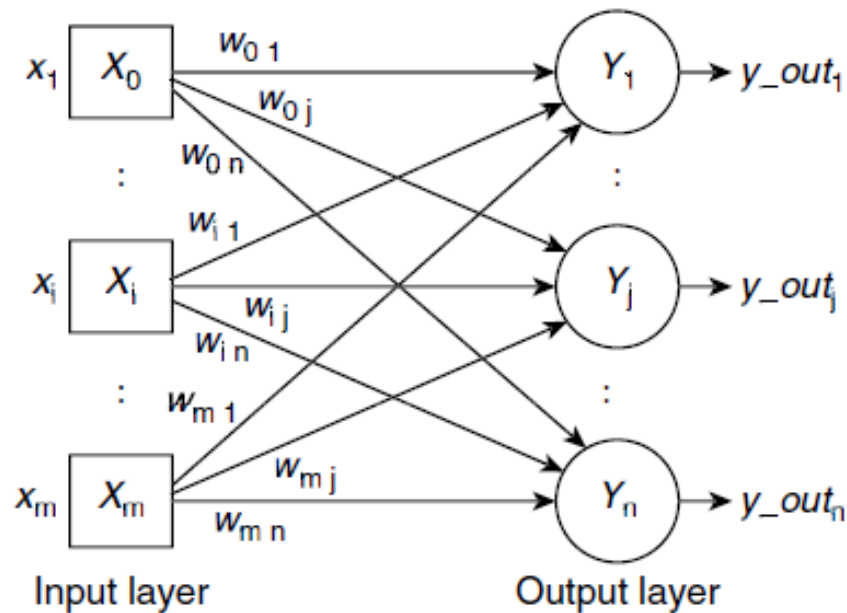
#	Condition	Action
1	The perceptron classifies the input pattern correctly ($y_{out} = t$)	No change in the current set of weights w_0, w_1, \dots, w_m .
2	The perceptron misclassifies the input pattern negatively ($y_{out} = -1$, but $t = +1$)	Increase each w_i by Δw_i , where Δw_i is proportional to x_i , for all $i = 0, 1, \dots, m$.
3	The perceptron misclassifies the input pattern positively ($y_{out} = +1$, but $t = -1$)	Decrease each w_i by Δw_i , where Δw_i is proportional to x_i , for all $i = 0, 1, \dots, m$.



Perceptron Learning Rule

- The perceptron learning rule is formulated as :

$$\Delta w_i = \eta \times (t - y_{out}) \times x_i, \text{ for } i = 0, 1, \dots, m$$



Perceptron Learning Rule

- The perceptron learning rule can be formulated as:

$$\Delta w_i = \eta \times (t - y_{out}) \times x_i, \text{ for } i = 0, 1, \dots, m$$

- Here η is a constant known as the *learning rate*.
- When a training vector is correctly classified then $y_{out} = t$, and the weight adjustment $\Delta w_i = 0$. When $y_{out} = -1$ but $t = +1$, i.e., the pattern is misclassified negatively, then $t - y_{out} = +2$ so that Δw_i is incremental and is proportional to x_i .
- If, however, the input pattern is misclassified positively, the adjustment is decremental.
- Using matrix notation, the perceptron learning rule may now be written as

$$\Delta W = \eta \times (t - y_{out}) \times X$$

- where, Δw_i and X are the vectors corresponding to the interconnection weights and the inputs.

Perceptron Learning Rule

$$\Delta W = [\Delta w_0, \dots, \Delta w_m], \text{ and}$$

$$X = [x_0, x, \dots, x_m].$$

- The net inputs to the perceptrons are

$$\begin{bmatrix} y_in_1 \\ \vdots \\ y_in_j \\ \vdots \\ y_in_n \end{bmatrix} = \begin{bmatrix} w_{01} & \cdots & w_{i1} & \cdots & w_{m1} \\ \vdots & & \vdots & & \vdots \\ w_{0j} & \cdots & w_{ij} & \cdots & w_{mj} \\ \vdots & & \vdots & & \vdots \\ w_{0n} & \cdots & w_{in} & \cdots & w_{mn} \end{bmatrix} \times \begin{bmatrix} x_0 \\ \vdots \\ x_i \\ \vdots \\ x_m \end{bmatrix}$$

or,

$$Y_in^T = W^T \times X^T$$

For such an ANN, the adjustment Δw_{ij} of the weight w_{ij} is given by

$$\Delta w_{ij} = \eta \times (t_j - y_out_j) \times x_i$$



Perceptron Learning Rule

Let us assume the following matrix notations :

$$\Delta W = \begin{bmatrix} \Delta w_{01} & \Delta w_{02} & \cdots & \Delta w_{0n} \\ \Delta w_{11} & \Delta w_{12} & \cdots & \Delta w_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta w_{m1} & \Delta w_{m2} & \cdots & \Delta w_{mn} \end{bmatrix}, T = [t_1 \quad \cdots \quad t_n],$$
$$X = [x_0, \quad \dots, \quad x_m], \text{ and } Y_out = [y_out_1, \quad \dots, \quad y_out_n].$$

Then the expression of the perceptron learning rule for the architecture shown in Fig. 6.34 becomes

$$[\Delta W]^T = \eta \times [T - Y_out]^T \times X \quad (6.4)$$



References

- <https://medium.com/analytics-Vidhya/brief-history-of-neural-networks-44c2bf72eec>
- Haykin, Simon. *“Neural networks and learning machines”*, 3/E. Pearson Education India, 2011.
- Bhattacharyya, Siddhartha & Maulik, Ujjwal, *“Soft Computing for Image and Multimedia Data Processing”*, Springer Berlin, Heidelberg, 2013.

Thank You!



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013

