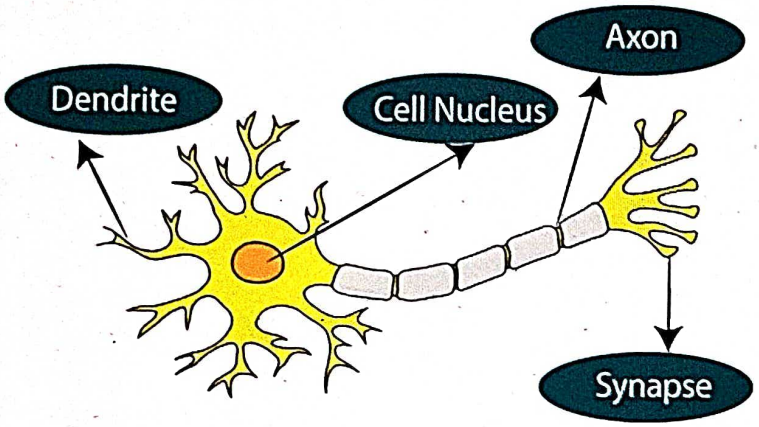
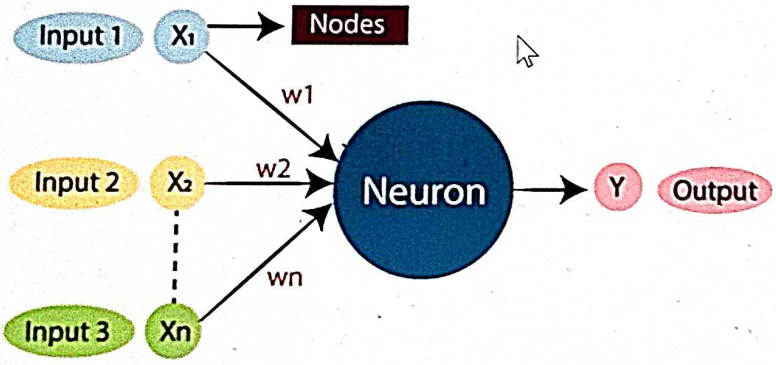
What is Artificial Neural Network? 1

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure illustrates the typical diagram of Biological Neural Network.

The typical Artificial Neural Network looks something like the given figure.



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

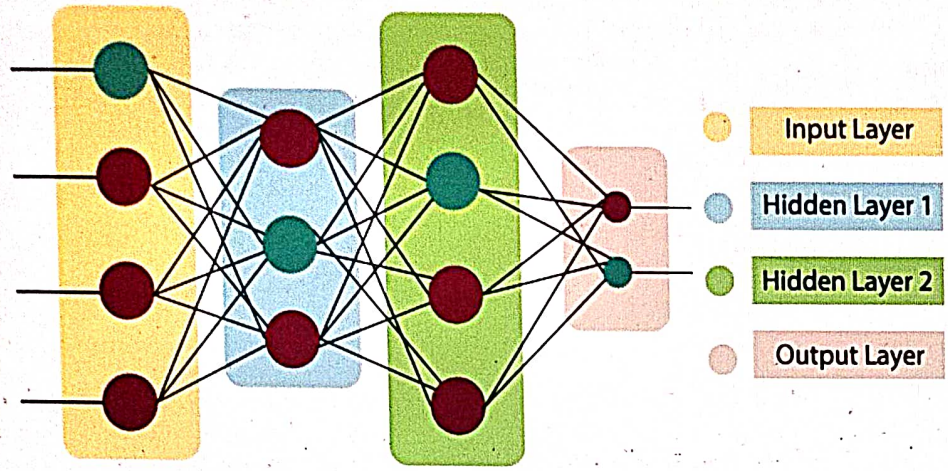
We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of. In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Lets us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:

Artificial Neural Network primarily consists of three layers:



Input Layer:

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

Hidden Layer:

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

Output Layer:

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

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

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

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

2 **Q) Define activation function. Explain different types of activation functions.**

- Activation Functions are extremely important feature of the Artificial Neural Network. They basically decide whether a neuron should be activated or not. It limits the output signal to a finite value.
- **Activation Function does the non-linear transformation** to the input making it capable to learn more complex relation between input and output. It make the network capable of learning more complex pattern.
- Without an activation function, the neural network is just a linear regression model as it performs only summation of product of input and weights.

Eg. In the below image 2 requires a complex relation which is curve unlike a simple linear relation in image 1.

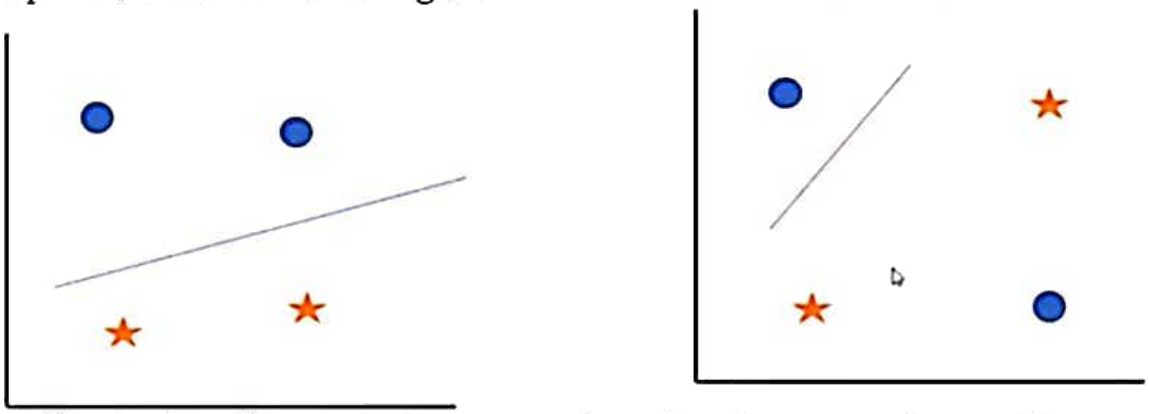


Fig. Illustrating the need of Activation Function for a complex problem.

Activation function must be efficient and it should **reduce the computation** time because the neural network sometimes trained on millions of data points.

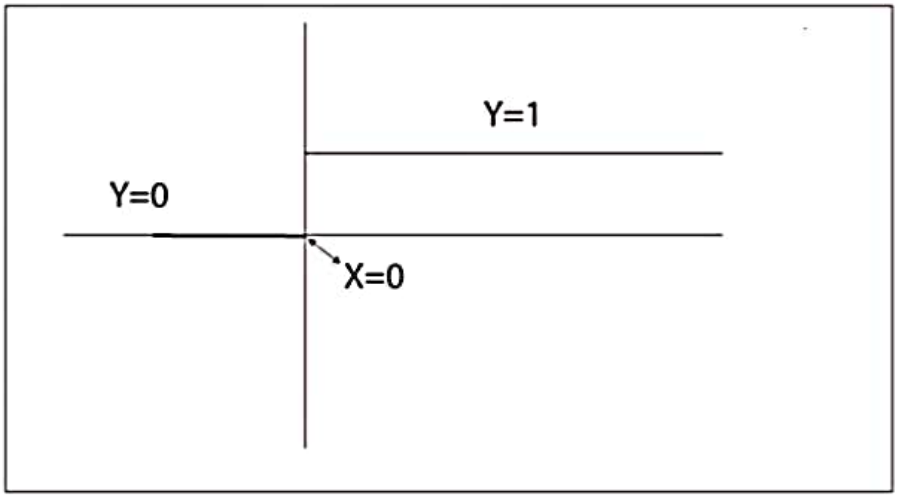
Types of AF:
The Activation Functions can be basically divided into 3 types-

1. Binary step Activation Function
2. Linear Activation Function
3. Non-linear Activation Functions

1. Binary Step Function

A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold, the neuron is activated and

sends exactly the same signal to the next layer.We decide some threshold value to decide output that neuron should be activated or deactivated.It is very simple and useful to classify binary problems or classifier.
Eg.f(x) = 1 if x > 0 else 0 if x <= 0



2. **Linear or Identity Activation Function**

As you can see the function is a line or linear. Therefore, the output of the functions will not be confined between any range.

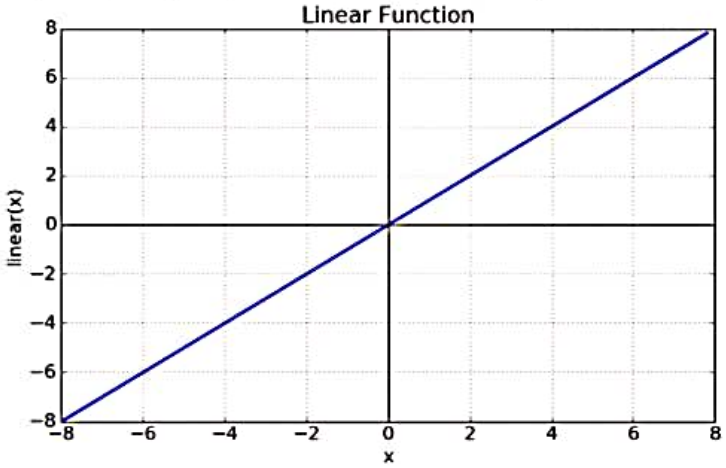


Fig: Linear Activation Function

Equation: f(x) = x
Range : (-infinity to infinity)
It doesn't help with the complexity or various parameters of usual data that is fed to the neural networks

3. **Non-linear Activation Function**

The Nonlinear Activation Functions are the most used activation functions. Nonlinearity helps to makes the graph look something like this.

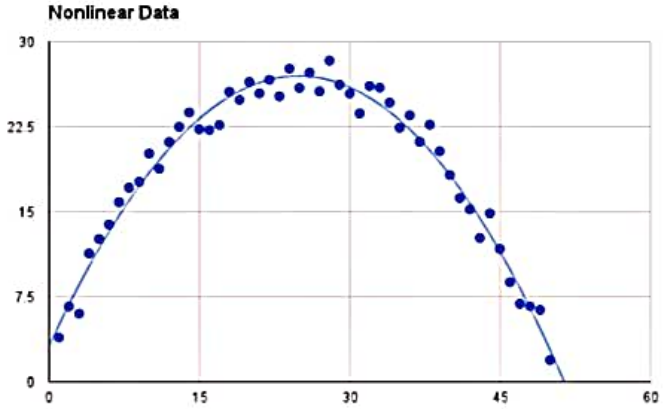


Fig: Non-linear Activation Function

The main terminologies needed to understand for nonlinear functions are:
Derivative or Differential: Change in y-axis w.r.t. change in x-axis.It is also known as slope.
Monotonic function: A function which is either entirely non-increasing or non-decreasing.

The Nonlinear Activation Functions are mainly divided on the basis of their range or curves-

Advantage of Non-linear function over the Linear function :
Differential is possible in all the non -linear function.
Stacking of network is possible, which helps us in creating deep neural nets.
It makes it easy for the model to generalize

3.1 Sigmoid(Logistic AF)(σ):
The main reason why we use sigmoid function is it exists between **0 to 1.**
It is especially used for models where we have to predict the probability as output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

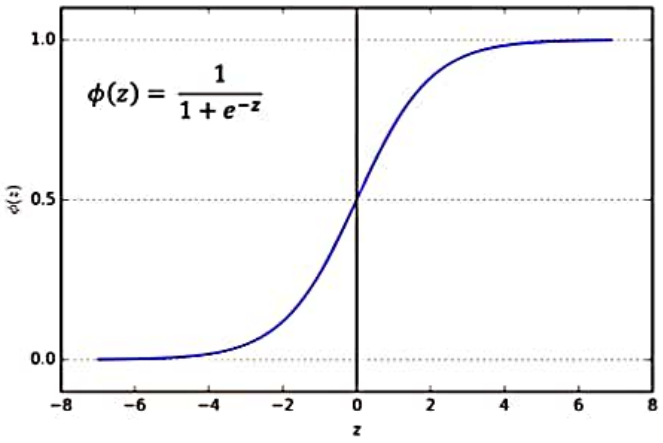


Fig: Sigmoid Function (S-shaped Curve)

The function is **differentiable and monotonic**. But function derivative is not monotonic.
The logistic sigmoid function can cause a neural network to get stuck at the training time.

Advantages

1. Easy to understand and apply
2. Easy to train on small dataset
3. Smooth gradient, preventing "jumps" in output values.
4. Output values bound between 0 and 1, normalizing the output of each neuron.

Disadvantages:

- Vanishing gradient—for very high or very low values of X, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction.
- Outputs not zero centered.
- Computationally expensive

3.2 **TanH(Hyperbolic Tangent AF):**

TanH is also like logistic sigmoid but in better way. The range of the TanHfunction is from **-1 to +1.**

TanH is often preferred over the sigmoid neuron because it is zero centred. The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in tanh graph.

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

tanh(x) = 2 * sigmoid(2x) - 1

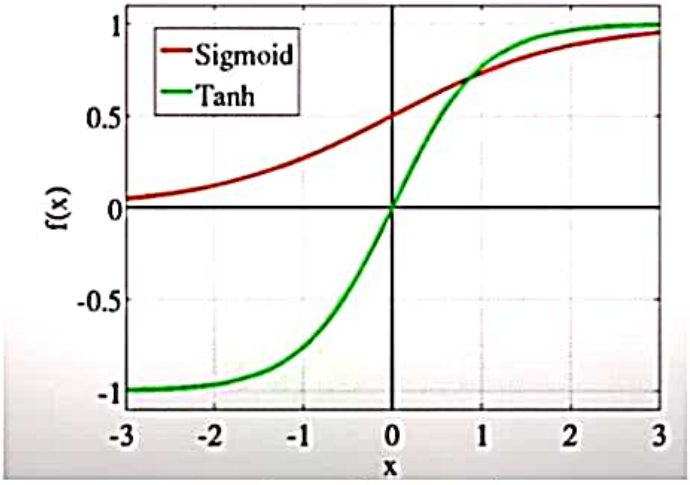


Fig. Sigmoid Vs Tanh

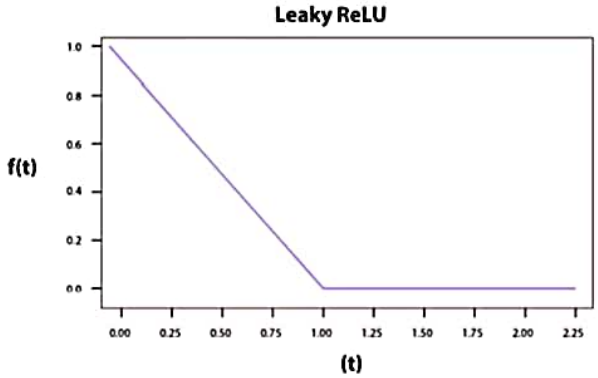
The function is **differentiable** and **monotonic**. But function derivative is not monotonic.

Advantages

- **Zero centered**—making it easier to model inputs that have strongly negative, neutral, and strongly positive values.

Leaky ReLU Activation Function

We needed the **Leaky ReLU activation** function to solve the '*Dying ReLU*' problem.
Leaky ReLU we do not make all negative inputs to zero but to a value near to zero which solves the major issue of ReLU activation function.



R(z) = max(0.1*z,z)

Advantages

- **Prevents dying ReLU problem**—this variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values
- Otherwise like ReLU

Disadvantages

- **Results not consistent**—leaky ReLU does not provide consistent predictions for negative input values.

3.4 **Softmax:**

- Sigmoid able to handle more than two cases(class label).
- Softmax can handle multiple cases. Softmax function squeeze the output for each class between 0 and 1 with sum of them is 1.
- It is ideally used in the final output layer of the classifier, where we are actually trying to attain the probabilities.
- Softmax produces multiple outputs for an input array. For this reason, we can build neural network models that can classify more than 2 classes instead of binary class solution.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

sigma	=	softmax
z_i	=	input vector
e^{z_i}	=	standard exponential function for input vector
K	=	number of classes in the multi-class classifier
e^{z_i}	=	standard exponential function for output vector
e^{z_j}	=	standard exponential function for output vector

Advantages

Able to handle multiple classes only one class in other activation functions—normalizes the outputs for each class between 0 and 1 with sum of the probabilities been equal to 1, and divides by their sum, giving probability of the input value being in a specific class.

Useful for output neurons—typically Softmax is used only for the output layer, for neural networks that need to classify inputs into multiple categories.

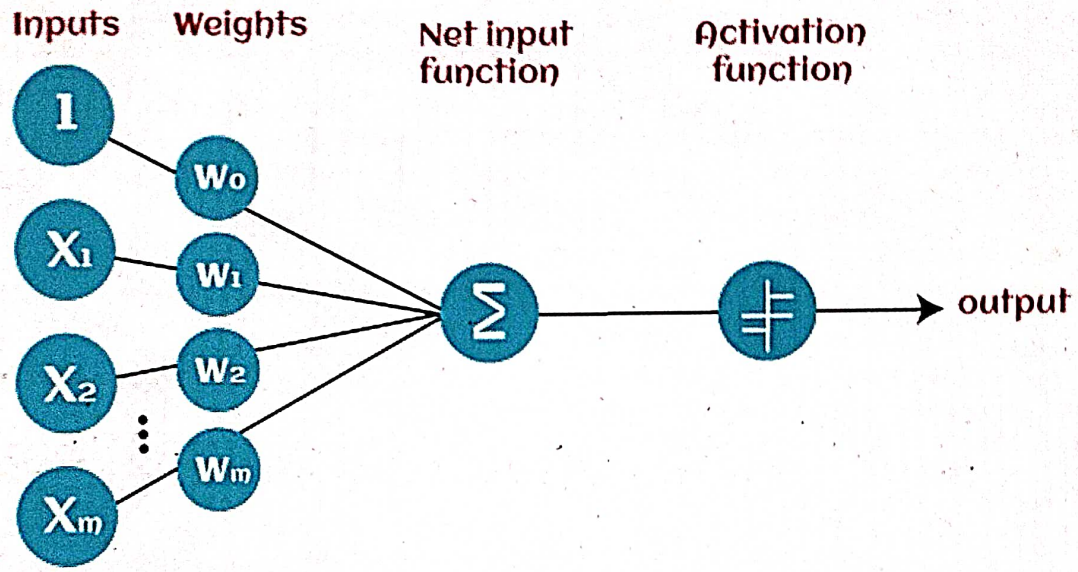
3b What is the Perceptron model in Machine Learning?

Perceptron is Machine Learning algorithm for supervised learning of various binary classification tasks. Further, ***Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.***

Perceptron model is also treated as one of the best and simplest types of Artificial Neural networks. However, it is a supervised learning algorithm of binary classifiers. Hence, we can consider it as a single-layer neural network with four main parameters, i.e., **input values, weights and Bias, net sum, and an activation function.**

Basic Components of Perceptron

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



- **Input Nodes or Input Layer:**

This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.

- **Wight and Bias:**

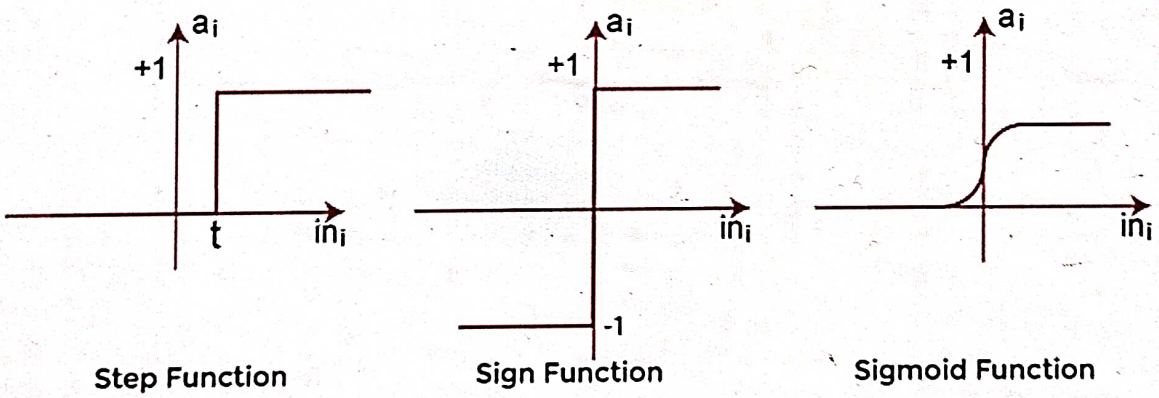
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

- **Activation Function:**

These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

Types of Activation functions:

- Sign function
- Step function, and
- Sigmoid function



Step Function Sign Function Sigmoid Function

The data scientist uses the activation function to take a subjective decision based on various problem statements and forms the desired outputs. Activation function may differ (e.g., Sign, Step, and Sigmoid) in perceptron models by checking whether the learning process is slow or has vanishing or exploding gradients.

Future of Perceptron

The future of the Perceptron model is much bright and significant as it helps to interpret data by building intuitive patterns and applying them in the future. Machine learning is a rapidly growing technology of Artificial Intelligence that is continuously evolving and in the developing phase; hence the future of perceptron technology will continue to support and facilitate analytical behavior in machines that will, in turn, add to the efficiency of computers.

The perceptron model is continuously becoming more advanced and working efficiently on complex problems with the help of artificial neurons.

5 **Q) Explain indetail about CNN model.**

MLP's use one perceptron for each input (e.g. pixel in an image, multiplied by 3 in RGB case). The amount of weights rapidly becomes unmanageable for large images. For a 224 x 224 pixel image with 3 color channels there are around 150,528 weights that must be trained! As a result, difficulties arise whilst training and overfitting can occur.

A **Convolutional neural network (CNN)** is a neural network that has one or more convolutional layers and is used mainly for image processing, classification, segmentation.

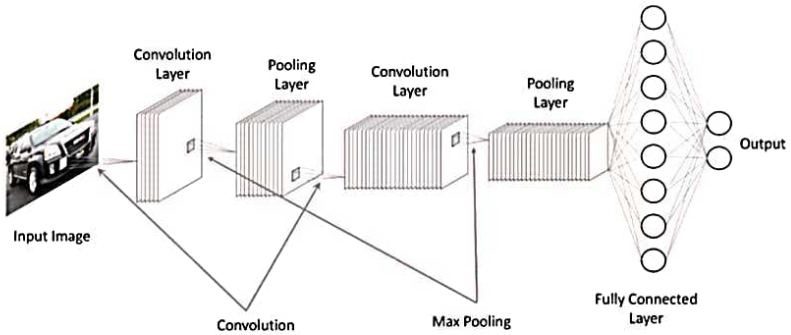


Fig. CNN Architecture

Input layer:
The input to a cnn, is mostly an image(nxmx1-gray scale image and nxmx3-colored image)

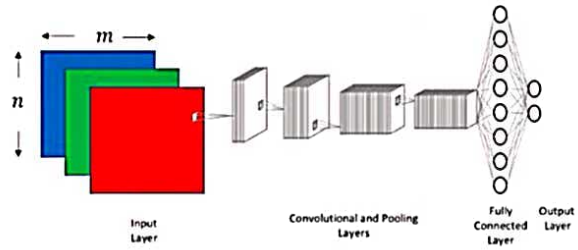


Fig. RGB image as input

Convolution layer:
Here, we basically define filters and we compute the convolution between the defined filters and each of the 3 images.

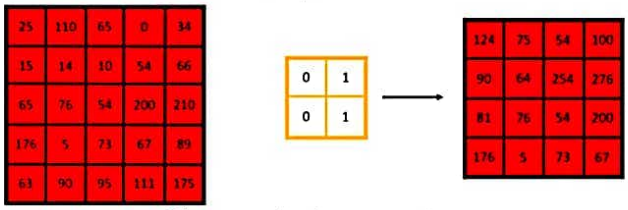


Fig. convolution operation

In the same way we apply to remaining (above is for red image, then we do same for green and blue) images. We can apply more than one filter. More filters we use, we can preserve spatial dimensions better.

We use convolution instead of considering flatten image as input as we will end up with a massive number of parameters that will need to be optimized and computationally expensive.
Eg. We require 25 weights if we take 5x5x1 image with out convolution.
We require 16 weights**(n-f+1 x n-f+1)** if we take 5x5x1 image with 2x2 convolution filter.

By using convolution we can prevent overfitting of the model.

It is worth to have ReLU activation function in convolution layer which passed only positive values and make negative values to zeros.

Pooling layer:
Pooling layer objective is to reduce the spatial dimensions of the data propagating through the network.

It is worth to have ReLU activation function in convolution layer which passed only positive values and make negative values to zeros.

Pooling layer:
Pooling layer objective is to reduce the spatial dimensions of the data propagating through the network.

1. Max Pooling is the most common, for each section of the image we scan and keep the highest value.

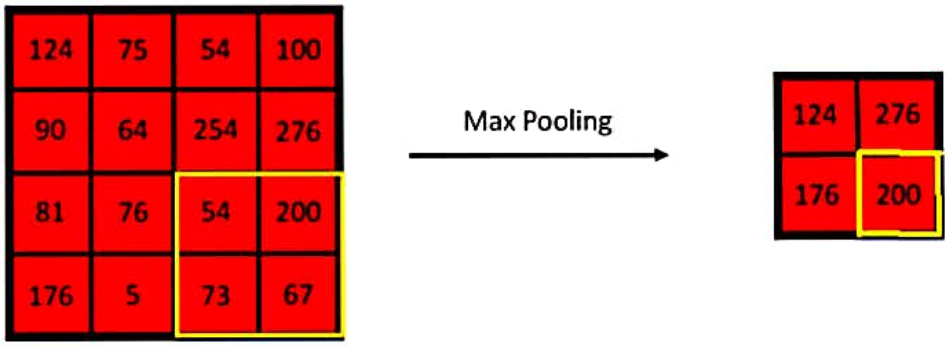


Fig. Max Pooling with stride = 2

Max. pooling provides spatial variance which enables the neural network to recognize objects in an image even if the object does not exactly resemble the original object.

2 Average Pooling: Here, we take average of area we scan.

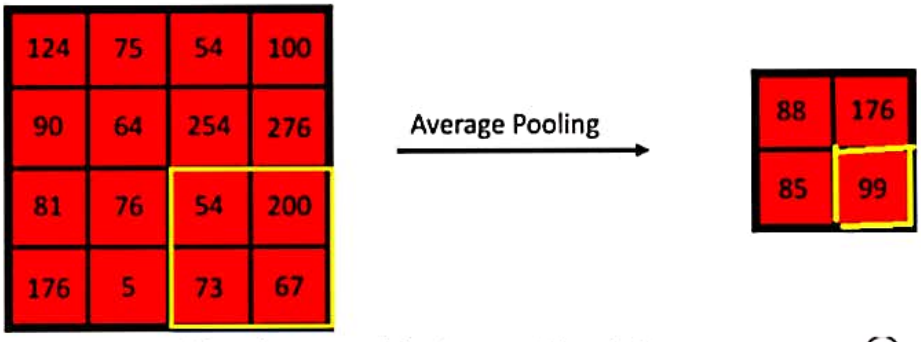


Fig. Average Pooling with stride = 2

Fully Connected Layer:
Here, we flatten the output of the last convolutional layer and connect every node of the current layer with every other node of the next layer.

This layer basically takes output of the preceding layer, whether it is a convolutional layer, ReLU or Pooling layer and outputs an n-dimensional vector, where n is number of classes pertaining to the problem.

Fig. Fully Connected Layer

6 Backpropagation is a fundamental algorithm used for training artificial neural networks, including feedforward neural networks and deep learning models like multilayer perceptrons. It's a supervised learning algorithm that's used to adjust the model's weights and biases in order to minimize the error between the predicted output and the actual target output. The name "backpropagation" is short for "backward propagation of errors."

Here's an overview of the backpropagation algorithm:

1. **Forward Pass:**
   - Start by feeding an input sample through the neural network.
   - Calculate the output by propagating the input through the network's layers, applying weights and biases, and applying activation functions at each layer.
2. **Error Calculation:**
   - Calculate the error (also known as the loss) between the predicted output and the actual target output. The choice of the error metric depends on the specific problem (e.g., mean squared error for regression, cross-entropy for classification).
3. **Backward Pass:**
   - The key step in backpropagation involves propagating the error backward through the network to update the model's parameters (weights and biases). This is done using the gradient of the error with respect to the parameters.
4. **Gradient Calculation:**
   - Calculate the gradient of the error with respect to the weights and biases at each layer of the network. This gradient represents how much the error would change with respect to a small change in the weights and biases.
5. **Weight and Bias Updates:**
   - Adjust the weights and biases in the direction that reduces the error. The magnitude of the adjustment is determined by the learning rate, a hyperparameter chosen in advance.
   - Update the weights and biases at each layer by subtracting the learning rate times the gradient.
6. **Repeat:**
   - Repeat the forward pass, error calculation, and backward pass for a specified number of iterations (epochs) or until the error converges to a satisfactory level.

7a The sigmoid function is a common activation function used in artificial neural networks. It has a characteristic S-shaped curve and maps input values to a range between 0 and 1. The formula for the sigmoid function is:

```scss
σ(x) = 1 / (1 + e^(-x))
```

Copy code

In this formula:

- σ(x) represents the output of the sigmoid function for input x.
- "e" is the base of the natural logarithm (approximately 2.71828).

You can use this formula to calculate the sigmoid activation for any real-numbered input "x." The output will be a value between 0 and 1, which is often interpreted as the probability or confidence level of an event occurring, especially in binary classification problems.

Here's an example of applying the sigmoid function to an input value:

Let's say you want to compute σ(2). Using the formula:

```scss
σ(2) = 1 / (1 + e^(-2))
```

Copy code

Calculating this:

```scss
σ(2) = 1 / (1 + e^(-2)) ≈ 0.8808
```

Copy code

So, the sigmoid activation for an input of 2 is approximately 0.8808.

- **Medical Diagnosis (3 marks):** Sigmoid functions are used in medical diagnosis systems to predict the likelihood of a disease based on patient data. For instance, determining the probability of a patient having a particular medical condition based on symptoms and test results.
- **Spam Detection (3 marks):** Sigmoid functions are applied in spam email classification, where they assess the likelihood that an email is spam or not. Messages with a high probability of being spam are filtered out.
- **Financial Risk Assessment (3 marks):** Sigmoid functions are used in credit scoring and risk assessment. They determine the probability of a borrower defaulting on a loan based on their financial history and other factors.
- **Neural Machine Translation (3 marks):** In machine translation models, sigmoid functions are applied to predict the probability of each word in the target language vocabulary, allowing the system to select the most probable word for translation.

No change in weight

Delta learning rule: 7b

The delta rule in an artificial neural network is a specific kind of backpropagation that assists in refining the machine learning/artificial intelligence network, making associations among input and outputs with different layers of artificial neurons. The Delta rule is also called the Delta learning rule.

Generally, backpropagation has to do with recalculating input weights for artificial neurons utilizing a gradient technique. Delta learning does this by using the difference between a target activation and an obtained activation. By using a linear activation function, network connections are balanced. Another approach to explain the Delta rule is that it uses an error function to perform gradient descent learning.

Delta rule refers to the comparison of actual output with a target output, the technology tries to discover the match, and the program makes changes. The actual execution of the Delta rule will fluctuate as per the network and its composition. Still, by applying a linear activation function, the delta rule can be useful in refining a few sorts of neural networks with specific kinds of backpropagation.

Delta rule is introduced by **Widrow and Hoff**, which is the most significant learning rule that depends on supervised learning.

This rule states that the change in the weight of a node is equivalent to the product of error and the input.

Mathematical equation:

The given equation gives the mathematical equation for delta learning rule:

$\Delta w = \mu.x.z$

$\Delta w = \mu(t-y)x$

Here,

Δw = weight change.

μ = the constant and positive learning rate.

X = the input value from pre-synaptic neuron.

z= (t-y) is the difference between the desired input t and the actual output y. The above mentioned mathematical rule cab be used only for a single output unit.

The different weights can be determined with respect to these two cases.

Case 1 - When $t \neq k$, then

$w(new) = w(old) + \Delta w$

Case 2 - When $t = k$, then

No change in weight

For a given input vector, we need to compare the output vector, and the final output vector would be the correct answer. If the difference is zero, then no learning takes place, so we need to adjust the weight to reduce the difference. If the set of input patterns is taken from an independent set, then it uses learn arbitrary connections using the delta learning rule. It has examined for networks with linear activation function with no hidden units. The error squared versus the weight graph is a paraboloid shape in n-space. The proportionality constant is negative, so the graph of such a function is concave upward with the least value. The vertex of the paraboloid represents the point where it decreases the error. The weight vector is comparing this point with the ideal weight vector. We can utilize the delta learning rule with both single output units and numerous output units. When we are applying the delta learning rule is to diminish the difference between the actual and probable output, we find an error.

8 **Biological Neurons and Biological Neural Networks:**

The ***biological neural network*** is also composed of several processing pieces known as ***neurons*** that are linked together via ***synapses***. These neurons accept either external input or the results of other neurons. The generated output from the individual neurons propagates its effect on the entire network to the last layer, where the results can be displayed to the outside world.

Every synapse has a processing value and weight recognized during network training. The performance and potency of the network fully depend on the neuron numbers in the network, how they are connected to each other (i.e., topology), and the weights assigned to every synapse.

Advantages and Disadvantages of Biological Neural Network

There are various ***advantages*** and ***disadvantages*** of the biological neural network. Some advantages and disadvantages of the biological neural network are as follows:

Advantages

1. It can handle extremely complex parallel inputs.
2. The input processing element is the synapses.

Disadvantages

1. As it is complex, the processing speed is slow.
2. There is no controlling mechanism in this network.

9a notable applications of ANNs:

1. **Image Recognition and Computer Vision:**
   - Convolutional Neural Networks (CNNs) are used for image classification, object detection, facial recognition, and image segmentation. They power applications like self-driving cars, medical image analysis, and photo tagging.
2. **Natural Language Processing (NLP):**
   - Recurrent Neural Networks (RNNs) and Transformers are used for tasks such as text classification, sentiment analysis, machine translation, and chatbots.
3. **Speech Recognition:**
   - ANNs are used to convert spoken language into text, enabling voice assistants like Siri and automatic transcription services.
4. **Recommendation Systems:**
   - Collaborative filtering and deep learning models are used for personalized content recommendations on platforms like Netflix, Amazon, and YouTube.
5. **Autonomous Vehicles:**
   - ANNs are crucial for self-driving cars, helping them perceive their environment, make decisions, and navigate safely.
6. **Healthcare:**
   - ANNs assist in disease diagnosis, medical image analysis, drug discovery, and predicting patient outcomes.
7. **Finance:**
   - ANNs are used for fraud detection, algorithmic trading, credit scoring, and risk assessment.
8. **Predictive Maintenance:**
   - ANNs can predict when machinery and equipment need maintenance, reducing downtime and costs in industries like manufacturing.
9. **Game Playing:**
   - Deep reinforcement learning techniques, such as Deep Q-Networks, have been used to achieve superhuman performance in games like Go, chess, and video games.
10. **Anomaly Detection:**
    - ANNs can detect unusual patterns in data, making them valuable for identifying fraud in financial transactions, network security breaches, and industrial equipment failures.

11. **Customer Churn Prediction:**
    - ANNs are used to predict when customers are likely to churn (cancel a service or subscription), allowing businesses to take proactive retention measures.
12. **Environmental Monitoring:**
    - ANNs can analyze data from sensors and satellites to monitor and predict natural disasters, weather patterns, and environmental changes.
13. **Drug Discovery:**
    - ANNs help in virtual screening and predicting the interactions between drugs and biological molecules, accelerating drug development.
14. **Agriculture:**
    - ANNs are used for crop yield prediction, disease detection in plants, and optimizing farming practices.
15. **Quality Control and Inspection:**
    - ANNs are used in manufacturing to inspect products for defects, ensuring quality control.
16. **Content Generation:**
    - ANNs, such as GANs (Generative Adversarial Networks), can generate art, music, and text content.
17. **Energy Management:**
    - ANNs optimize energy consumption and grid management, improving energy efficiency.
18. **Human Pose Estimation:**
    - ANNs can estimate the positions of body joints in images or videos, with applications in motion capture and fitness tracking.
19. **Emotion Recognition:**
    - ANNs are used to detect emotions from facial expressions or voice, which is applied in human-computer interaction and market research.
20. **Fraud Detection:**
    - ANNs help identify fraudulent activities in banking, e-commerce, and insurance.

challenges in ANNs:

1. **Overfitting:**
   - ANNs are prone to overfitting, where the model performs well on the training data but poorly on unseen data. This occurs when the network captures noise in the data rather than true patterns. Regularization techniques are often used to mitigate this issue.
2. **Data Quality and Quantity:**
   - ANNs require large amounts of high-quality labeled data for effective training. In many real-world applications, obtaining such data can be challenging and costly.
3. **Model Complexity:**
   - Deep neural networks can be extremely complex with a large number of parameters. This complexity can make training and optimization more difficult.
4. **Choice of Architecture:**
   - Selecting the right architecture (e.g., number of layers, neurons, activation functions) for a specific task can be challenging. There's no one-size-fits-all solution, and architecture design often relies on trial and error.
5. **Vanishing and Exploding Gradients:**
   - In deep networks, gradients can become too small (vanishing gradients) or too large (exploding gradients) during training. This makes optimization challenging. Techniques like careful weight initialization and activation functions can help alleviate this issue.
6. **Hardware and Computational Resources:**
   - Training deep networks can be computationally intensive and may require specialized hardware (e.g., GPUs or TPUs) for efficient training. Access to such resources can be a limitation.
7. **Interpretability:**
   - ANNs, especially deep neural networks, are often considered as "black boxes." Understanding the reasons behind their predictions and decision-making can be challenging, making them less interpretable compared to traditional machine learning models.
8. **Training Time:**
   - Training deep networks can take a long time, especially for large datasets and complex architectures. Faster convergence and training techniques are areas of ongoing research.

Advantages of Artificial Neural Network (ANN)

9b

Parallel processing capability:

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

Storing data on the entire network:

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

Capability to work with incomplete knowledge:

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

Having a memory distribution:

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

Having fault tolerance:

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of Artificial Neural Network:

Assurance of proper network structure:

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

Unrecognized behavior of the network:

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

Hardware dependence:

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

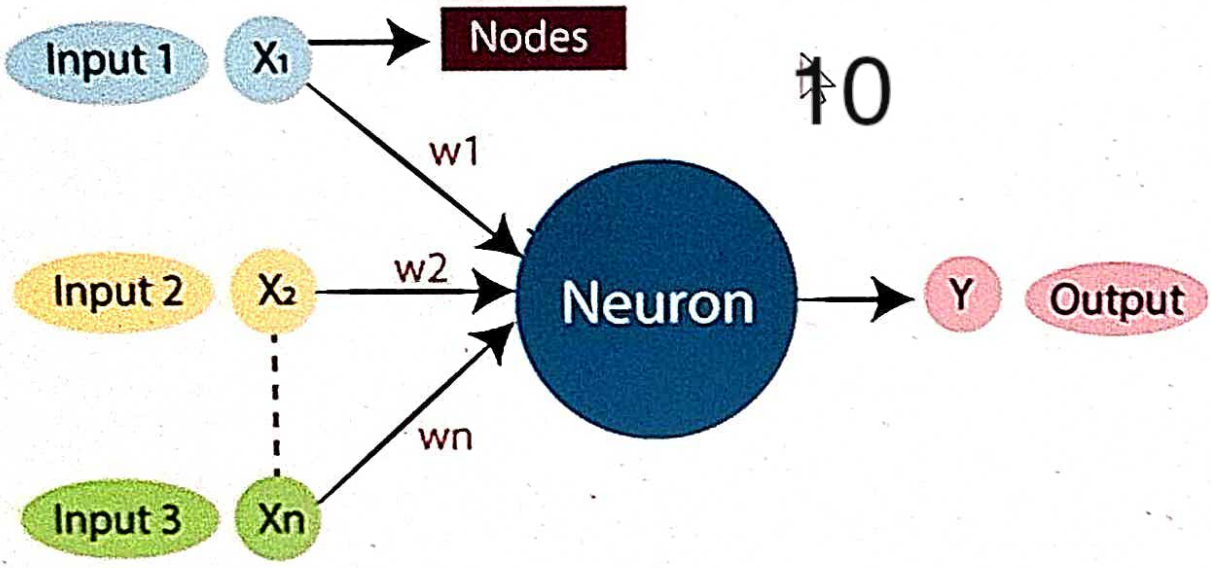
Difficulty of showing the issue to the network:

ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

The duration of the network is unknown:

The network is reduced to a specific value of the error, and this value does not give us optimum results.

The typical Artificial Neural Network looks something like the given figure.



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

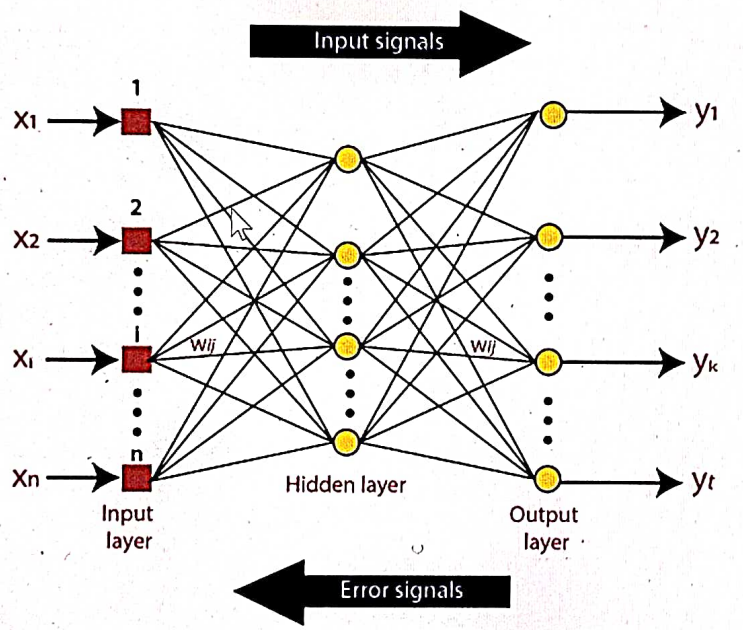
An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

How do artificial neural networks work?

Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.

If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions. Let us take a look at each of them in details:

Binary:

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

Sigmoidal Hyperbolic:

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

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

Where ???? is considered the Steepness parameter.