1. **Describe the structure of an artificial neuron. How is it similar to a biological neuron? What are its main components?**

**Ans: -**

**Structure of Artificial Neural Networks**

The Artificial Neural Networks (ANNs) are computational models that are inspired from human brain. In another words, it is the modelling of human brain work logic mathematically. The main goal is providing*a result(or output)* that in line with our purpose after passing some processes. Just as the human brain has billions of neurons, ANNs also has hundreds or thousands of *artificial neurons*.

ANNs are used for regression or classification problems and they consists of two basic architecture:

1. Single-Layer Artificial Neural Networks
2. Multi-Layer Artifical Neural Networks

**Single-Layer Artificial Neural Networks**

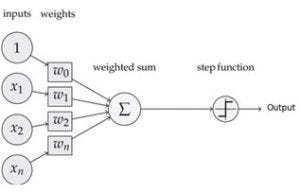
The Single-Layer Artificial Neural Networks are also called as **Perceptrons**. The Perceptron, is the basic component of ANNs. It is actually binary classification algorithm that is invented by Frank Rosenblatt in 1957. That is, it is an algorithm that tries to decide which output class an input belongs to.

**How does the Perceptron work?**

The perceptron consists of five components:

1. **Inputs:** These are the independent variables (x) that we have.
2. **Weigths:**Weight parameters (w) control the strength of the connection between inputs and neurons. It can also be said to represent the effect of an independent variable on the result.
3. **Bias value(b):**It is a constant value that allows to control the output value. Also, when all inputs are zero, it ensures that the process can still continue.
4. **Activation Functions:**The activation function (f) defines the output of the neuron according to certain conditions.
5. **Output:**The dependent variable (y) is the result we want to find. In perceptrons, the result is divided into two classes, classes 1 and 0.

If we formulate the process, we can show it like this:**y=f(x×w+b)**



The Perceptron

**The process proceeds as follows:**

* The weighted sum is calculated by first multiplication of weights and inputs, then addition of them. The bias value is included in this value.(bias=b=*x0×w0*)

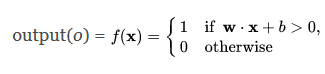
https://miro.medium.com/v2/resize:fit:252/1*XgkifN5mbeGoNpD6xkCBuw.png

The weighted sum

* The activation function is applied to the weighted sum(z) and the result is found. The perceptrons use *step function* as a activation function. According to step funciton, if the weigted sum;

**→ z>0, then result is 1,**

**→ z≤0, then result is 0.**



The output formula with activation function

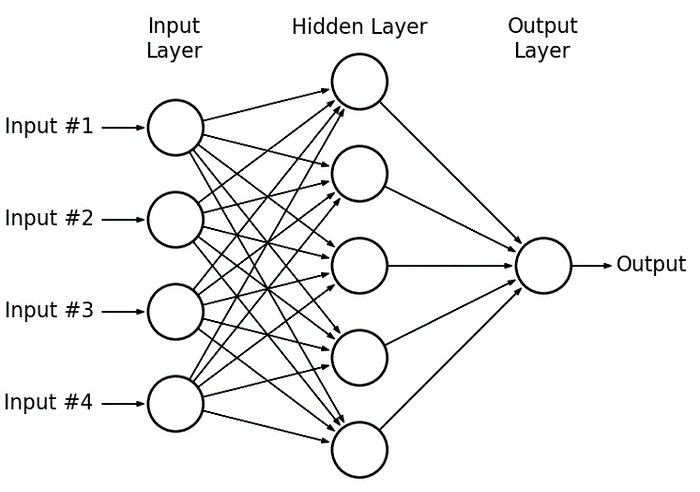
The applied area of perceptrons is limited. Beacuse perceptrons are usually used for simple binary classification problems that are linearly separable.

**Multi-Layer Neural Networks**

As the name suggests, the multi-layer neural networks, or the multi-layer perceptrons(MLPs), consist layers more than one. Beside of the perceptrons, they can be used for non-linearly separable problems. They achieve this with the activation functions they use in their layers. The activation functions make the output of neurons nonlinear. In this way, it enables to solve more complex problems. (Without activation function, ANNs actually become a linear regression model.)

The basic layers of MLPs are:

* **1 input layer**→ it includes **1 neuron per input x**.
* **Hidden layers(one or more)**→ The number of neurons it consists depends on the problem.
* **1 output layer** →The number of neurons it consists depends on the problem.



*The Multi-Layer Perceptron*

In the first step , **for every neurons of hidden layers**, the same process in the perceptron is applied:

1. The weighted sum(z) is calculated.
2. It is transmitted to related hidden neuron, then the activation function present in the neuron(ReLU or SELU) is applied.

In the next step, the outputs of hidden layers are transmitted to ***output layer***. As said before, the **number of neurons** depends on the problem in here:

***Regression:****consists of****1 neuron****,*

***Binary Classification:****consists of****1 neuron,***

***Multi-label Classification:****consists of****1 neuron per label****,*

***Multi-class Classification:****consists of****1 neuron per class****in the output layer.*

The activation functions in neurons of output layer also depends on the task:

***Regression:****None or ReLU/Softplus(if positive outputs) or Logistic/tanh( if bounded outputs),*

***Binary Classification:****Logistic(sigmoid) function,*

***Multi-label Classification:****Logistic(sigmoid) function,*

***Multi-class Classification:****Softmax function.*

The main goal is to enable ANN to learn the most accurate weight values (so achiving most accurate result) with correct hidden layer and neuron numbers. We can do this by applying certain processes in our artificial neural network and optimizing it.

We have heard of the latest advancements in the field of deep learning due to the usage of different neural networks. Most of these achievements are simply astonishing and I find myself amazed after reading every new article on the advancements in this field almost every week. At the most basic level, all such neural networks are made up of artificial neurons that try to mimic the working of biological neurons. I had a curiosity about understanding how these artificial neurons compare to the structure of biological neurons in our brains and if possibly this could lead to a way to improve neural networks further. So if you are curious about this topic too, then let’s embark on a short 5-minute journey to understand this topic in detail…

First, let’s understand how biological neurons work inside our brains…

**Biological Neurons**

Neurons are the basic functional units of the nervous system, and they generate electrical signals called **action potentials**, which allows them to quickly transmit information over long distances.  
Almost all the neurons have three basic functions essential for the normal functioning of all the cells in the body.

These are to:  
1. Receive signals (or information) from outside.  
2. Process the incoming signals and determine whether or not the information should be passed along.  
3. Communicate signals to target cells which might be other neurons or muscles or glands.

Now let us understand the ***basic parts*** of a neuron to get a deeper insight into how they actually work…

A biological neuron is mainly composed of 3 main parts and an external part called synapse:-

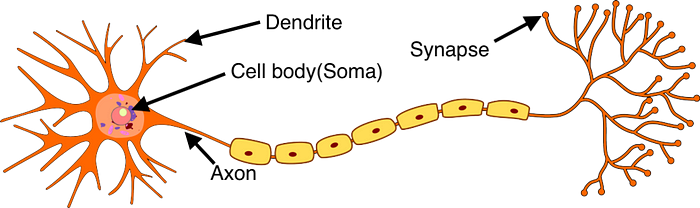


Image source: Wikimedia Commons

1. **Dendrite**

Dendrites are responsible for getting incoming signals from outside

**2. Soma**

Soma is the cell body responsible for the processing of input signals and deciding whether a neuron should fire an output signal

**3. Axon**

Axon is responsible for getting processed signals from neuron to relevant cells

**4. Synapse**

Synapse is the connection between an axon and other neuron dendrites

**Working of the parts**

The task of receiving the incoming information is done by dendrites, and processing generally takes place in the cell body. Incoming signals can be either **excitatory** — which means they tend to make the neuron **fire** (generate an electrical impulse) — or **inhibitory** — which means that they tend to keep the neuron from firing.

Most neurons receive many input signals throughout their dendritic trees. A single neuron may have more than one set of dendrites and may receive many thousands of input signals. Whether or not a neuron is excited into firing an impulse depends on the sum of all of the excitatory and inhibitory signals it receives. The processing of this information happens in **soma**which is neuron cell body. If the neuron does end up firing, the nerve impulse, or **action potential**, is conducted down the axon.

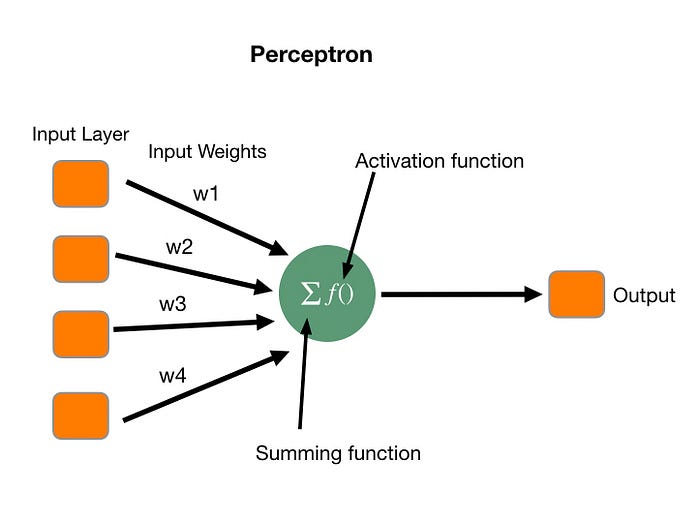
Towards its end, the axon splits up into many branches and develops bulbous swellings known as **axon terminals** (or **nerve terminals**). These axon terminals make connections on target cells.

**Artificial Neurons**

Artificial neuron also known as perceptron is the basic unit of the neural network. In simple terms, it is a mathematical function based on a model of biological neurons. It can also be seen as a simple logic gate with binary outputs. They are sometimes also called **perceptrons.**

Each artificial neuron has the following main functions:

1. Takes inputs from the input layer
2. Weighs them separately and sums them up
3. Pass this sum through a nonlinear function to produce output.



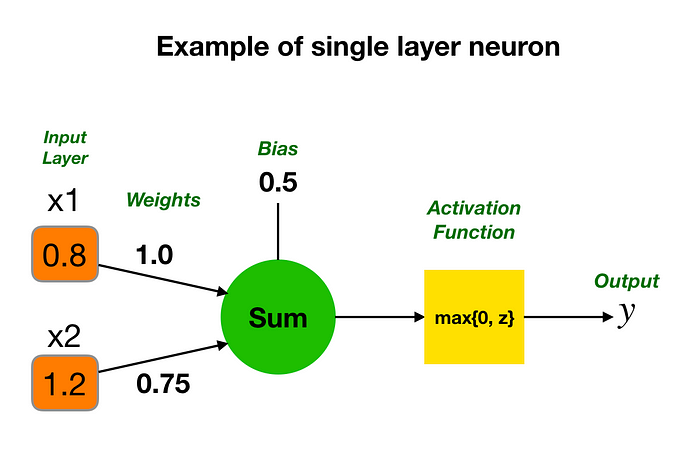
Basic perceptron diagram

The perceptron(neuron) consists of 4 parts:

1. **Input values or One input layer**  
   We pass input values to a neuron using this layer. It might be something as simple as a collection of array values. It is similar to a dendrite in biological neurons.
2. **Weights and Bias**Weights are a collection of array values which are multiplied to the respective input values. We then take a sum of all these multiplied values which is called a weighted sum. Next, we add a bias value to the weighted sum to get final value for prediction by our neuron.
3. **Activation Function**  
   Activation Function decides whether or not a neuron is fired. It decides which of the two output values should be generated by the neuron.
4. **Output Layer**  
   Output layer gives the final output of a neuron which can then be passed to other neurons in the network or taken as the final output value.

Now, all the above concepts might seem like too much theoretical knowledge without any practical insights, so let’s understand the working of an artificial neuron with an example.

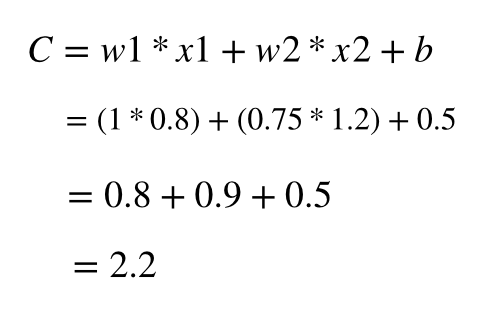
Consider a neuron with two inputs (x1,x2) as shown below:



Single-layer neuron example

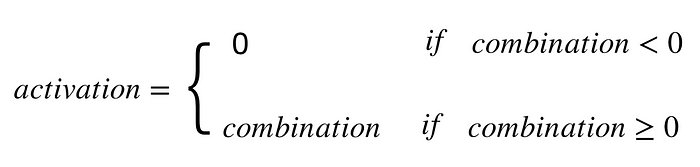
1. The values of the two inputs(x1,x2) are 0.8 and 1.2
2. We have a set of weights (1.0,0.75) corresponding to the two inputs
3. Then we have a bias with value 0.5 which needs to be added to the sum

The input to activation function is then calculated using the formula:-



Calculation of `C` value

Now the combination(C) can be fed to the activation function. Let us first understand the logic of Rectified linear (ReLU) activation function which we are currently using in our example:



ReLU activation function

In our case, the combination value we got was 2.2 which is greater than 0 so the output value of our activation function will be 2.2.

This will be the final output value of our single layer neuron

# Components of ANN

Neural networks are designed in a way to enact like an actual human brain. The neural network tries to simulate the functions of interconnected neurons by passing an input layer which can be seen as the sensory organ used to receive the information. The information thus received is provided to the neurons in the hidden layer. Each neuron gives importance to certain input nodes based on the weights and finally, an output is produced based on the information built by the neurons. In an artificial neural network, various components are tuned to improve the ANN model. Each component has its influence giving a better result that may be at the cost of an increase in model training time. So what are the components and where do we need to tune them? Let’s find out.

### ****Input layer****

The input layer is composed of nodes that brings in the initial data after pre-processing. The data could be on any subject matter depending upon our classification problem but the values are always numerical. If they are not then we have to convert them into numerical using pre-processing techniques.

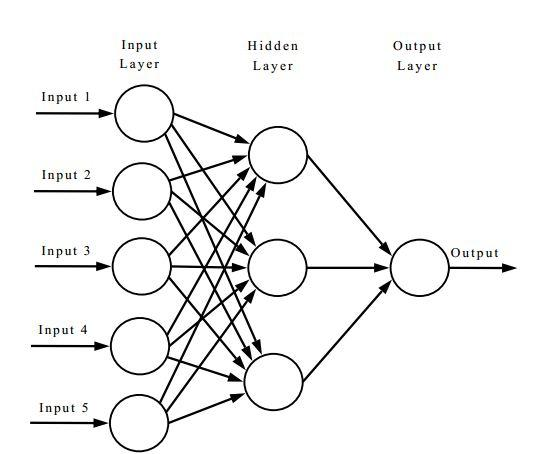
Input nodes are nothing but the features of the data we have. Let's say if it's 'Salary of employees' dataset then the features could be employee name, gender, salary, age, and experience. We have to keep the important feature and drop the irrelevant ones. The number of features is equivalent to the number of input nodes.

### ****Hidden layer****

The hidden layer is the ones that reside between the input layer and the output layer. It takes the weighted nodes as the input and produces an output with the help of an activation function. This is the layer where the actual learning takes place. The hidden layer works as a biological neuron.

### ****Output node****

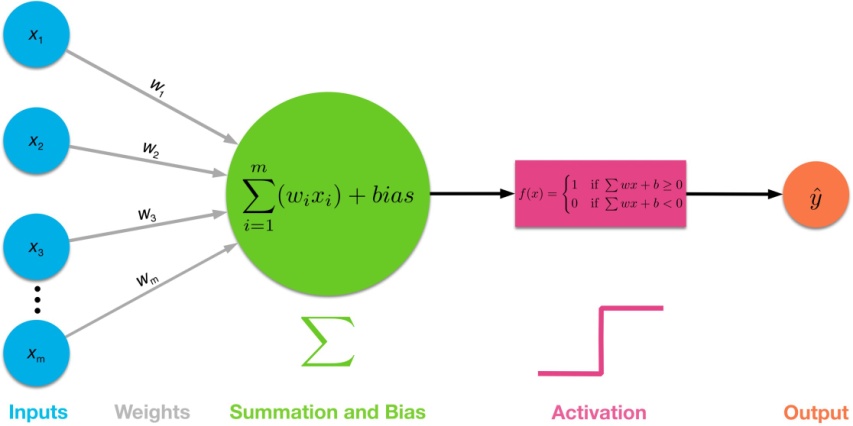
It is the last layer of a neural network. There can be a single node or multiple nodes in the output layer depending upon the classification problem.



### ****Activation function****

The activation function defines the output of a node based on the input provided. Neurons in neural networks support two functions which are summation and activation.

* A **summation** is the matrix product of weights and input.
* **Activation** is the transformation of the values after the summation. After the activation is performed, the resultant is considered as the output.

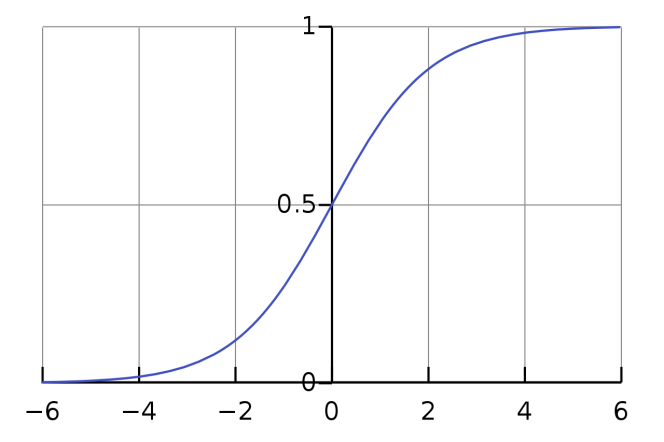


The activation function to use depends upon the problem. In the case of classification, we use the sigmoid activation function, in the case of multiclass classification we'll use softmax function and in case of regression, we use the ReLU activation function. Now let's discuss these functions.

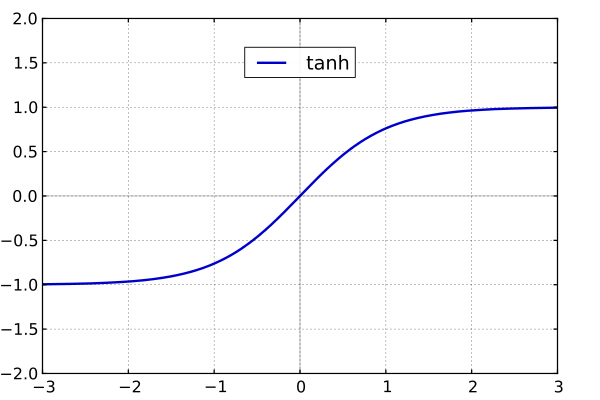
**Linear activation-** It represents a linear change from input to output. It is rarely used activation because it has a constant gradient due to which you can't do gradient descent. When you'll calculate the partial derivative in linear activation then you'll get 0 and hence you can't improve it.

**Non-linear activation-**This type of activation change the input in a non-linear fashion. It is widely used in deep learning models. Different types of activations are used in different cases. Its types are:

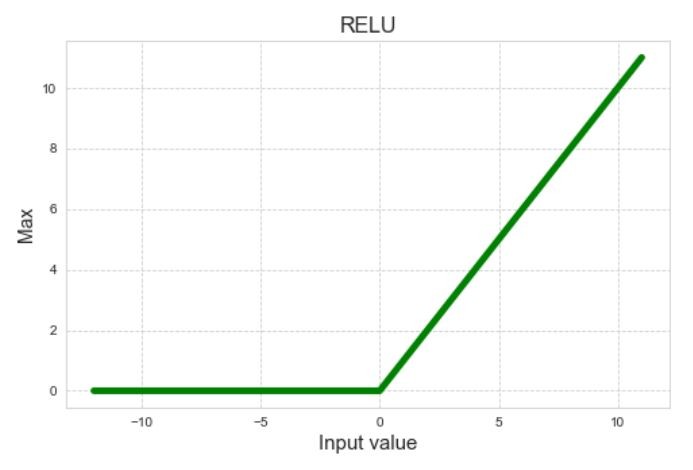
* **Sigmoid function-** It is also known as the logistic function and it converts the input value(x) in a range from 0 to 1, irrespective of how large or small the input value is. It is usually used for binary classification.



* **Tanh function-** Hyper tan function is quite similar to the sigmoid function, the only difference is that it converts the input values from -1 to 1 rather than 0 to 1. It gives more dispersed values.



**ReLU function-** It stands for 'Rectified Linear Unit'. It is calculated by max(0,x) where x is the positive input value. Any value below 0 is considered 0 and the value above is taken as it is. ReLU is very flexible when dealing with non-linear data.



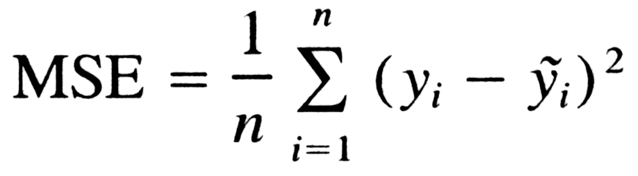
* **Softmax activation-** It is similar to a sigmoid function in terms of classification. We usually use it for multiclass classification. The difference between sigmoid and softmax is that in softmax the probabilities of class sum up to 1.

## ****Loss Metrics****

The loss metrics is a numerical measure of how wrong our predictions are. A bad prediction means greater loss and vice verse. Mathematically, it is the difference between the actual output and the predicted output. So let’s discuss some of the loss metrics used by neural network:

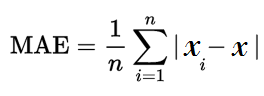
**Mean Squared Error (MSE)**

This function calculates the mean of the square of all the errors values i.e the difference between true and predicted values



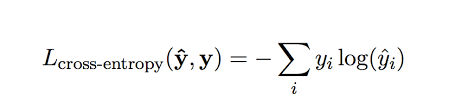
**Mean Absolute Error (MAE)**

This function is quite similar to MSE, the only difference is that in mean absolute error we take the mode of the error values and not the square. It works well even with the outliers. It is not widely used because it generates a large gradient even for small values.



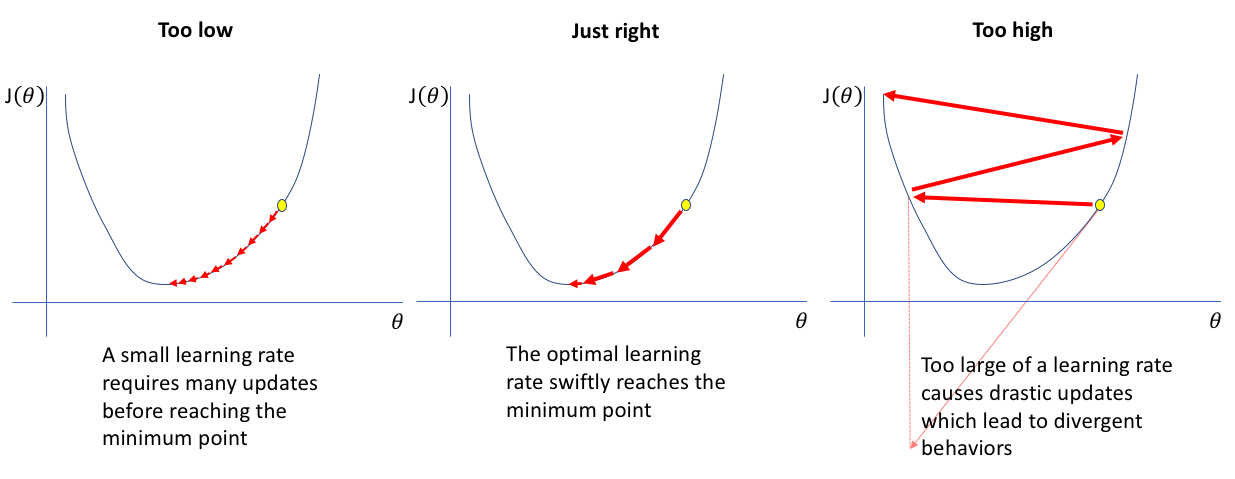
**Cross-Entropy Loss (log loss)**

This function measures the performance of a classification model whose output is a probabilityvalue between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. A perfect model would have a log loss of 0.



## ****Learning rate****

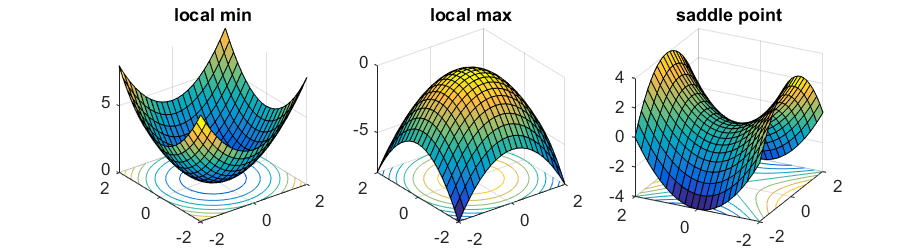
Deep neural network uses a stochastic gradient descent algorithm to train. It is an optimization algorithm that estimates the error gradient. The weights are updated using backpropagation. The amount by which the weight is updated during training is known as step size or **Learning rate**. The learning rate hyperparameter controls the rate or speed at which the model learns.



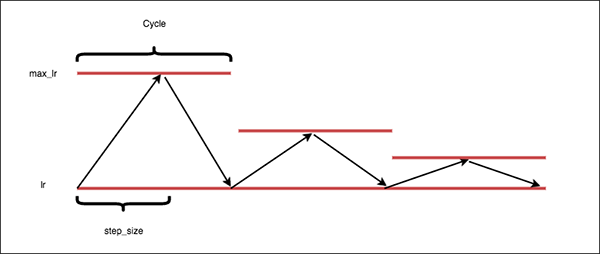
**Effects of the learning rate**

* A large learning rate allows the model to learn faster, at the cost of reaching the local minima. A smaller learning rate allows the model to learn more optimal solutions (reach the global minima) but it takes longer for the model to train. So basically it’s a trade-off between accuracy and the time taken for the model to train.
* Usually, the learning rate is set below 1 so that as the weights update, they don’t shoot and hence avoid the divergence problem. We can use the learning rate schedule in which we vary the learning rate rather than keeping a fixed value.

**Problem with fixed Learning rate-**With a fixed learning rate we can encounter a saddle point when considering multiple dimensions. A saddle point is a point which is maxima along one dimension but minima along the other.



As a solution, there is a concept of the cyclic learning rate, developed by Leslie N. Smith which states "Instead of monotonically decreasing the learning rate, this method lets the learning rate cyclically vary between reasonable boundary values. Training with cyclical learning rates instead of fixed values achieves improved classification accuracy without a need to tune and often in fewer iterations".



## ****Optimizer****

During the training process, we make changes in the parameters (weights) of our model to try and minimize that loss function in order to make our predictions as correct as possible. But how exactly do you do that? How do you change the parameters of your model, by how much, and when?

Optimizers shape the model in a way that produces an accurate form possible by tuning the weights. The loss function is the guide to the terrain, telling the optimizer when it’s moving in the right or wrong direction. It cannot know where to start from so it starts with random values. As the loss starts to decrease it move in that direction and as it starts increasing, the optimizer tunes it back accordingly. Some of the optimizers are:

* Stochastic Gradient descent
* Adagrad
* Adadelta
* RMSProp
* Adam
* Nadam

**Adam** is the most widely used optimizer as it has fewer hyperparameters to work on and the convergence is more accurate.

These are the primary components of an artificial neural network. we can adjust them accordingly depending upon the data we are working on and the model requirements. There are a lot more elements which can be tweaked and tuned to improve the ann model.

1. **What are the different types of activation functions popularly used? Explain each of them.**

**Ans:-**

## Types of Activation Functions

**The different kinds of activation functions include:**

### Linear Activation Functions

A linear function is also known as a straight-line function where the activation is proportional to the input i.e. the weighted sum from neurons. It has a simple function with the equation:

*f(x) = ax + c*

The problem with this activation is that it cannot be defined in a specific range. Applying this function in all the nodes makes the activation function work like linear regression. The final layer of the Neural Network will be working as a linear function of the first layer. Another issue is the gradient descent when differentiation is done, it has a constant output which is not good because during backpropagation the rate of change of error is constant that can ruin the output and the logic of backpropagation.

### Non-Linear Activation Functions

The non-linear functions are known to be the most used activation functions. It makes it easy for a neural network model to adapt with a variety of data and to differentiate between the outcomes.

**These functions are mainly divided basis on their range or curves:**

#### a) Sigmoid Activation Functions

Sigmoid takes a real value as the input and outputs another value between 0 and 1. The sigmoid activation function translates the input ranged in (-∞,∞) to the range in (0,1)

#### b) Tanh Activation Functions

The tanh function is just another possible function that can be used as a non-linear activation function between layers of a neural network. It shares a few things in common with the sigmoid activation function. Unlike a sigmoid function that will map input values between 0 and 1, the Tanh will map values between -1 and 1. Similar to the sigmoid function, one of the interesting properties of the tanh function is that the derivative of tanh can be expressed in terms of the function itself.

#### c) ReLU Activation Functions

The formula is deceptively simple: *max(0,z)*. Despite its name, Rectified Linear Units, it’s not linear and provides the same benefits as Sigmoid but with better [performance](https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/).

##### **Leaky Relu**

Leaky Relu is a variant of ReLU. Instead of being 0 when z<0, a leaky ReLU allows a small, non-zero, constant gradient α (normally, α=0.01). However, the consistency of the benefit across tasks is presently unclear. Leaky ReLUs attempt to fix the “dying ReLU” problem.

##### **Parametric Relu**

PReLU gives the neurons the ability to choose what slope is best in the negative region. They can become ReLU or leaky ReLU with certain values of α.

#### d) Maxout

The Maxout activation is a generalization of the ReLU and the leaky ReLU functions. It is a piecewise linear function that returns the maximum of inputs, designed to be used in conjunction with the dropout regularization technique. Both ReLU and leaky ReLU are special cases of Maxout. The Maxout neuron, therefore, enjoys all the benefits of a ReLU unit and does not have any drawbacks like dying ReLU. However, it doubles the total number of parameters for each neuron, and hence, a higher total number of parameters need to be trained.

#### e) ELU

The Exponential Linear Unit or ELU is a function that tends to converge faster and produce more accurate results. Unlike other activation functions, ELU has an extra alpha constant which should be a positive number. ELU is very similar to ReLU except for negative inputs. They are both in the identity function form for non-negative inputs. On the other hand, ELU becomes smooth slowly until its output equal to -α whereas ReLU sharply smoothes.

#### f) Softmax Activation Functions

Softmax function calculates the probabilities distribution of the event over ‘n’ different events. In a general way, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will help determine the target class for the given inputs.

## When to use which Activation Function in a Neural Network?

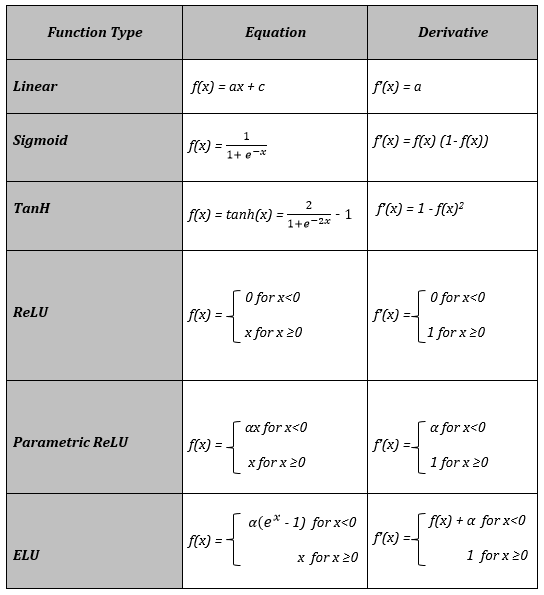
Specifically, it depends on the problem type and the value range of the expected [output](https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/). For example, to predict values that are larger than 1, tanh or sigmoid are not suitable to be used in the output layer, instead, ReLU can be used. On the other hand, if the output values have to be in the range (0,1) or (-1, 1) then ReLU is not a good choice, and sigmoid or tanh can be used here. While performing a classification task and using the neural network to predict a probability distribution over the mutually exclusive class labels, the softmax activation function should be used in the last layer. However, regarding the hidden layers, as a rule of thumb, use ReLU as an activation for these layers.

In the case of a binary classifier, the Sigmoid activation [function](https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/) should be used. The sigmoid activation function and the tanh activation function work terribly for the hidden layer. For hidden layers, ReLU or its better version leaky ReLU should be used. For a multiclass classifier, Softmax is the best-used activation function. Though there are more activation functions known, these are known to be the most used activation functions.

### ****Elements of Neural Networks****

* **Input Layer:**This is the first layer of the neural network, and it takes in the data that the network is going to learn from. For example, if you are training a neural network to recognize handwritten digits, the input layer would receive an image of a handwritten digit.
* **Output Layer:** This is the last layer of the neural network, and it produces the output of the network. For example, if you are training a neural network to recognize handwritten digits, the output layer would produce a prediction of which digit is in the image.
* **Hidden Layers:**These layers are in between the input layer and the output layer, and they do all the hard work of learning the patterns in the data. Neural networks with more hidden layers are better at learning complex patterns, like the ones that tell a cat from a dog.
* **Neurons:**Neural networks are made up of tiny processing units called neurons. They are connected to each other by weights, and they process the data that is passed to them from other neurons. The weights determine how much influence one neuron has on another neuron.

## Activation Functions and their Derivatives



## Implementation using Python

Having learned the types and significance of each activation function, it is also essential to implement some basic (non-linear) activation  
functions using python code and observe the output for more clear understanding of the concepts:

### Sigmoid Activation Function

import matplotlib.pyplot as plt

import numpy as np

def sigmoid(x):

s=1/(1+np.exp(-x))

ds=s\*(1-s)

return s,ds

x=np.arange(-6,6,0.01)

sigmoid(x)

fig, ax = plt.subplots(figsize=(9, 5))

ax.spines['left'].set\_position('center')

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

ax.xaxis.set\_ticks\_position('bottom')

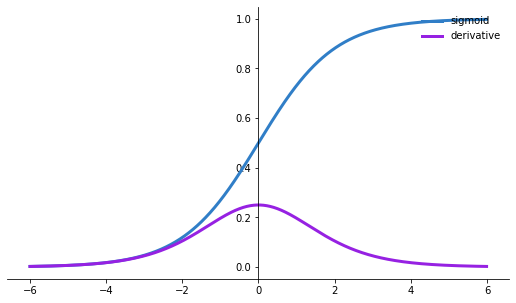
ax.yaxis.set\_ticks\_position('left')

ax.plot(x,sigmoid(x)[0], color="#307EC7", linewidth=3, label="sigmoid")

ax.plot(x,sigmoid(x)[1], color="#9621E2", linewidth=3, label="derivative")

ax.legend(loc="upper right", frameon=False)

fig.show()



#### Observations:

* The sigmoid function has values between 0 to 1.
* The output is not zero-centered.
* Sigmoids saturate and kill gradients.
* At the top and bottom level of sigmoid functions, the curve changes slowly, the derivative curve above shows that the slope or gradient it is zero.

### Tanh Activation Function

import matplotlib.pyplot as plt

import numpy as np

def tanh(x):

t=(np.exp(x)-np.exp(-x))/(np.exp(x)+np.exp(-x))

dt=1-t\*\*2

return t,dt

z=np.arange(-4,4,0.01)

tanh(z)[0].size,tanh(z)[1].size

fig, ax = plt.subplots(figsize=(9, 5))

ax.spines['left'].set\_position('center')

ax.spines['bottom'].set\_position('center')

ax.spines['right'].set\_color('none')

ax.spines['top'].set\_color('none')

ax.xaxis.set\_ticks\_position('bottom')

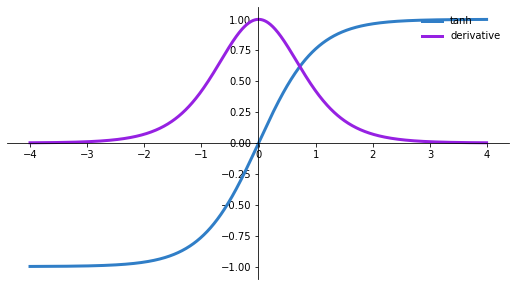
ax.yaxis.set\_ticks\_position('left')

ax.plot(z,tanh(z)[0], color="#307EC7", linewidth=3, label="tanh")

ax.plot(z,tanh(z)[1], color="#9621E2", linewidth=3, label="derivative")

ax.legend(loc="upper right", frameon=False)

fig.show()



#### Observations:

* Its output is zero-centered because its range is between -1 to 1. i.e. -1 < output < 1.
* Optimization is easier in this method hence in practice it is always preferred over the Sigmoid function.

## Pros and Cons of Activation Functions

| **Type of Function** | **Pros** | **Cons** |
| --- | --- | --- |
| Linear | – Provides a range of activations, not binary. – Can connect multiple neurons and make decisions based on max activation. – Constant gradient for stable descent. – Changes are constant for error correction. | – Limited modeling capacity due to linearity. |
| Sigmoid | – Nonlinear, allowing complex combinations. – Produces analog activation, not binary. | – Suffers from the “vanishing gradients” problem, making it slow to learn. |
| Tanh | – Stronger gradient compared to sigmoid. – Addresses the vanishing gradient issue to some extent. | – Still prone to vanishing gradient problems. |
| ReLU | – Solves the vanishing gradient problem. – Computationally efficient. | – Can lead to “Dead Neurons” due to fragile gradients. Should be used only in hidden layers. |
| Leaky ReLU | – Mitigates the “dying ReLU” problem with a small negative slope. | – Lacks complexity for some classification tasks. |
| ELU | – Can produce negative outputs for x>0. | – May lead to unbounded activations, resulting in a wide output range. |

1. **Explain, in details, Rosenblatt’s perceptron model. How can a set of data be classified using a simple perceptron?**

**ANS:-**

Rosenblatt’s perceptron model

**Rosenblatt perceptron** is a binary single neuron model. The inputs integration is implemented through the addition of the weighted inputs that have fixed weights obtained during the training stage. If the result of this addition is larger than a given threshold θ the neuron fires. When the neuron fires its output is set to 1, otherwise it’s set to 0.

This model implements the functioning of a single neuron that can solve linear classification problems through very simple learning algorithms. **Rosenblatt Perceptrons** are considered as the first generation of**neural networks** (the network is only compound of one neuron  ). This simple single neuron model has the main limitation of not being able to solve non-linear separable problems. In my next post I will describe how this advantage was overcome and what happens when we have a layer of various perceptrons or try different neuron activation functions.

**The Rosenblatt’s Perceptron (1957)**

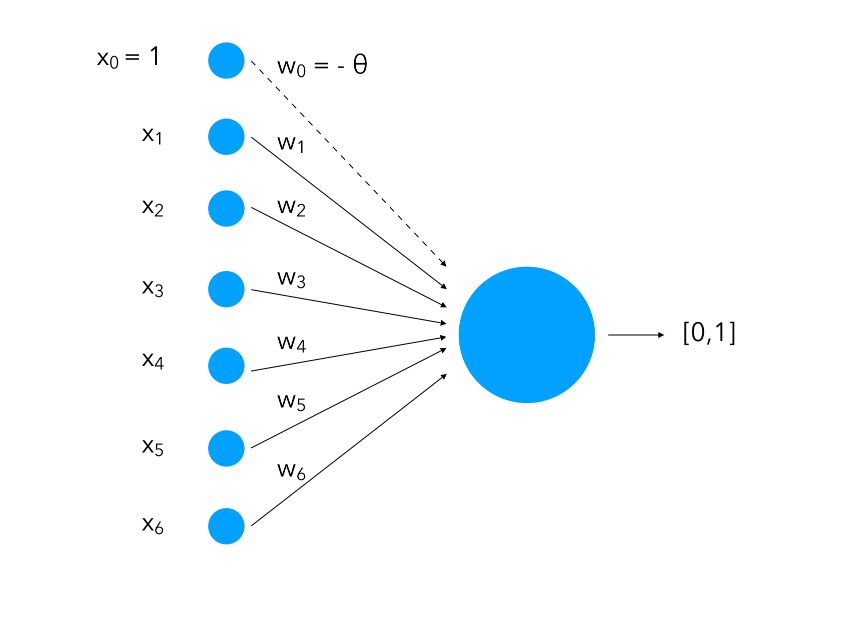
**The classic model**

The Rosenblatt’s Perceptron was designed to overcome most issues of the McCulloch-Pitts neuron :

* it can process non-boolean inputs
* and it can assign different weights to each input automatically
* the threshold θ𝜃 is computed automatically

A perceptron is a single layer Neural Network. A perceptron can simply be seen as a set of inputs, that are weighted and to which we apply an activation function. This produces sort of a weighted sum of inputs, resulting in an output. This is typically used for classification problems, but can also be used for regression problems.

The perceptron was first introduced in 1957 by Franck Rosenblatt. Since then, it has been the core of Deep Learning. We can represent schematically a perceptron as :



We attach to each input a weight ( wi𝑤𝑖) and notice how we add an input of value 1 with a weight of −θ−𝜃. This is called bias. What we are doing is instead of having only the inputs and the weight and compare them to a threshold, we also learn the threshold as a weight for a standard input of value 1.

The inputs can be seen as neurons and will be called the **input layer**. Altogether, these neurons and the function (which we’ll cover in a minute) form a **perceptron**.

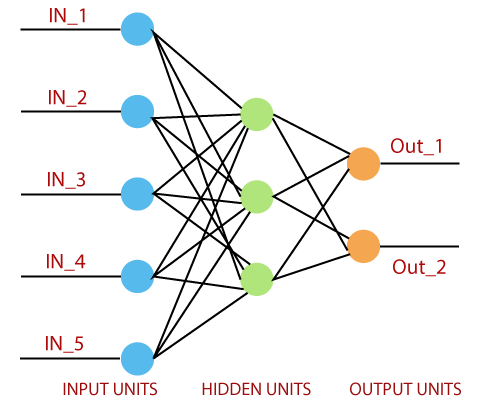
How do we make classification using a perceptron then?

y=1𝑦=1 if ∑iwixi≥0∑𝑖𝑤𝑖𝑥𝑖≥0, else y=0𝑦=0

One limitation remains: the inputs need to be linearly separable since we split the input space into two halves.

The perceptron is a single processing unit of any neural network. **Frank Rosenblatt** first proposed in **1958** is a simple neuron which is used to classify its input into one or two categories. Perceptron is a linear classifier, and is used in supervised learning. It helps to organize the given input data.

A perceptron is a neural network unit that does a precise computation to detect features in the input data. Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as **Linear Binary Classifier**.



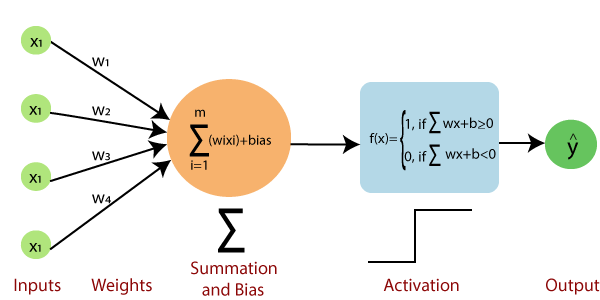
Perceptron uses the step function that returns +1 if the weighted sum of its input 0 and -1.

The activation function is used to map the input between the required value like (0, 1) or (-1, 1).

Backward Skip 10sPlay VideoForward Skip 10s

ADVERTISEMENT

A regular neural network looks like this:

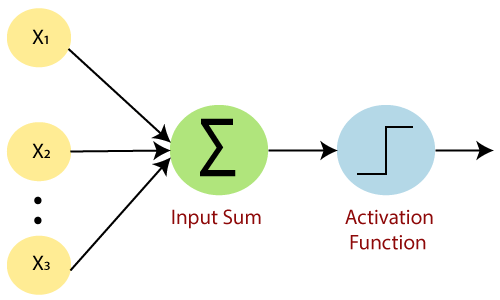


### The perceptron consists of 4 parts.

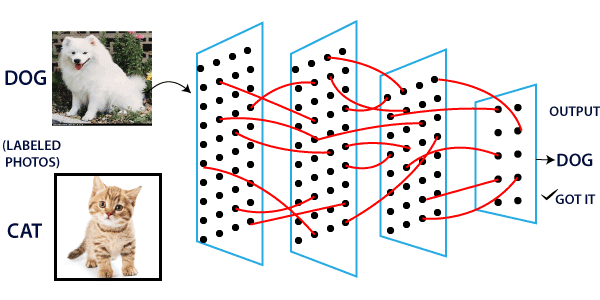
ADVERTISEMENT

ADVERTISEMENT

* **Input value or One input layer:** The input layer of the perceptron is made of artificial input neurons and takes the initial data into the system for further processing.
* **Weights and Bias:**  
  **Weight:** It represents the dimension or strength of the connection between units. If the weight to node 1 to node 2 has a higher quantity, then neuron 1 has a more considerable influence on the neuron.  
  **Bias:** It is the same as the intercept added in a linear equation. It is an additional parameter which task is to modify the output along with the weighted sum of the input to the other neuron.
* **Net sum:** It calculates the total sum.
* **Activation Function:** A neuron can be activated or not, is determined by an activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.



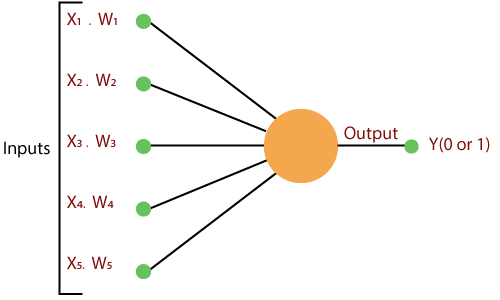
A standard neural network looks like the below diagram.



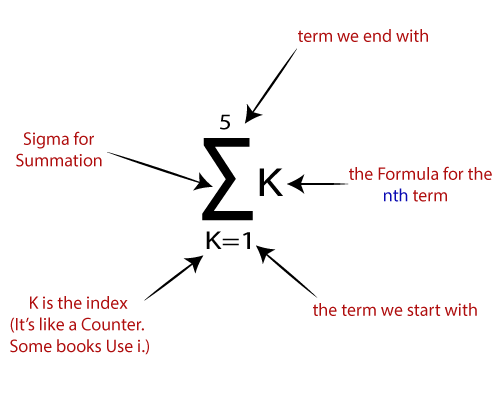
### How does it work?

The perceptron works on these simple steps which are given below:

**a.** In the first step, all the inputs x are multiplied with their weights **w**.



**b.** In this step, add all the increased values and call them the **Weighted sum**.

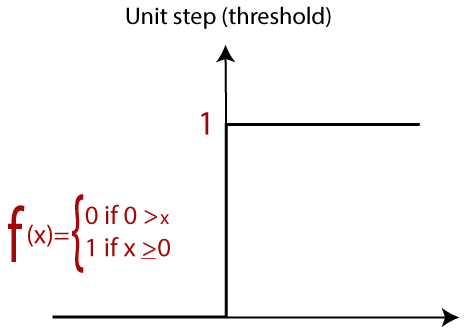


**c.** In our last step, apply the weighted sum to a correct **Activation Function**.

ADVERTISEMENT

**For Example:**

A Unit Step Activation Function



There are two types of architecture. These types focus on the functionality of artificial neural networks as follows-

* Single Layer Perceptron
* Multi-Layer Perceptron

## Single Layer Perceptron

The single-layer perceptron was the first neural network model, proposed in 1958 by Frank Rosenbluth. It is one of the earliest models for learning. Our goal is to find a linear decision function measured by the weight vector w and the bias parameter b.

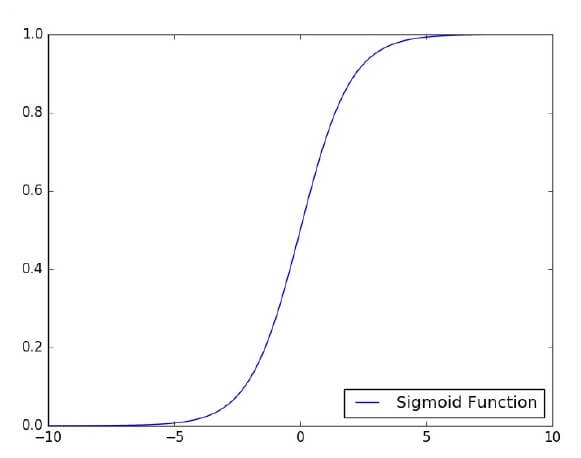
To understand the perceptron layer, it is necessary to comprehend artificial neural networks (ANNs).

The artificial neural network (ANN) is an information processing system, whose mechanism is inspired by the functionality of biological neural circuits. An artificial neural network consists of several processing units that are interconnected.

This is the first proposal when the neural model is built. The content of the neuron's local memory contains a vector of weight.

The single vector perceptron is calculated by calculating the sum of the input vector multiplied by the corresponding element of the vector, with each increasing the amount of the corresponding component of the vector by weight. The value that is displayed in the output is the input of an activation function.

Let us focus on the implementation of a single-layer perceptron for an image classification problem using TensorFlow. The best example of drawing a single-layer perceptron is through the representation of "**logistic regression**."



Now, We have to do the following necessary steps of training logistic regression-

* The weights are initialized with the random values at the origination of each training.
* For each element of the training set, the error is calculated with the difference between the desired output and the actual output. The calculated error is used to adjust the weight.
* The process is repeated until the fault made on the entire training set is less than the specified limit until the maximum number of iterations has been reached.

**Complete code of Single layer perceptron**

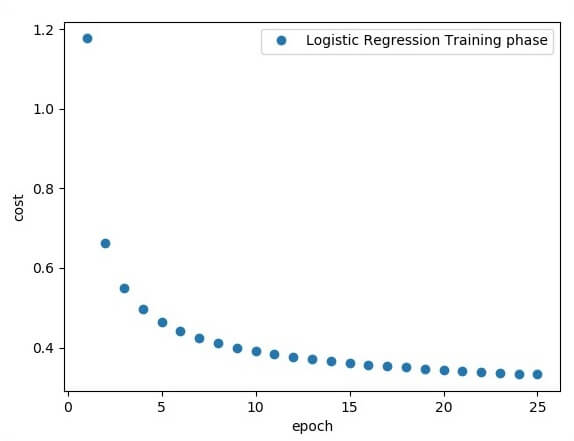
1. # Import the MINST dataset
2. from tensorflow.examples.tutorials.mnist **import** input\_data
3. mnist = input\_data.read\_data\_ ("/tmp/data/", one\_hot=True)
5. **import** tensorflow as tf
6. **import** matplotlib.pyplot as plt
7. # Parameters
8. learning\_rate = 0.01
9. training\_epochs = 25
10. batch\_size = 100
11. display\_step = 1
13. # tf Graph Input
14. x = tf.placeholder("float", [none, 784]) # MNIST data image of shape 28\*28 = 784
15. y = tf.placeholder("float", [none, 10]) # 0-9 digits recognition => 10 classes
16. # Create model
17. # Set model weights
18. W = tf.Variable(tf.zeros([784, 10]))
19. b = tf.Variable(tf.zeros([10]))
20. # Constructing the model
21. activation=tf.nn.softmaxx(tf.matmul (x, W)+b) # Softmax
22. of function
23. # Minimizing error using cross entropy
24. cross\_entropy = y\*tf.log(activation)
25. cost = tf.reduce\_mean\ (-tf.reduce\_sum\ (cross\_entropy, reduction\_indice = 1))
26. optimizer = tf.train.GradientDescentOptimizer(learning\_rate).minimize(cost)
27. #Plot settings
28. avg\_set = []
29. epoch\_set = []
30. # Initializing the variables where init = tf.initialize\_all\_variables()
31. # Launching the graph
32. with tf.Session() as sess:
33. sess.run(init)
35. # Training of the cycle in  the dataset
36. **for** epoch in range(training\_epochs):
37. avg\_cost = 0.
38. total\_batch = **int**(mnist.train.num\_example/batch\_size)
40. # Creating loops at all the batches in the code
41. **for** i in range(total\_batch):
42. batch\_xs, batch\_ys = mnist.train.next\_batch(batch\_size)
43. # Fitting the training by the batch data sess.run(optimizr,  feed\_dict = {
44. x: batch\_xs, y: batch\_ys})
45. # Compute all the average of loss avg\_cost += sess.run(cost, \ feed\_dict = {
46. x: batch\_xs, \ y: batch\_ys}) //total batch
47. # Display the logs at each epoch steps
48. **if** epoch % display\_step==0:
49. print("Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format (avg\_cost))
50. avg\_set.append(avg\_cost) epoch\_set.append(epoch+1)
51. print ("Training phase finished")
53. plt.plot(epoch\_set,avg\_set, 'o', label = 'Logistics Regression Training')
54. plt.ylabel('cost')
55. plt.xlabel('epoch')
56. plt.legend()
57. plt.show()
59. # Test the model
60. correct\_prediction = tf.equal (tf.argmax (activation, 1),tf.argmax(y,1))
62. # Calculating the accuracy of dataset
63. accuracy = tf.reduce\_mean(tf.cast (correct\_prediction, "float")) print
64. ("Model accuracy:", accuracy.eval({x:mnist.test.images, y: mnist.test.labels}))

**The output of the Code:**



ADVERTISEMENT

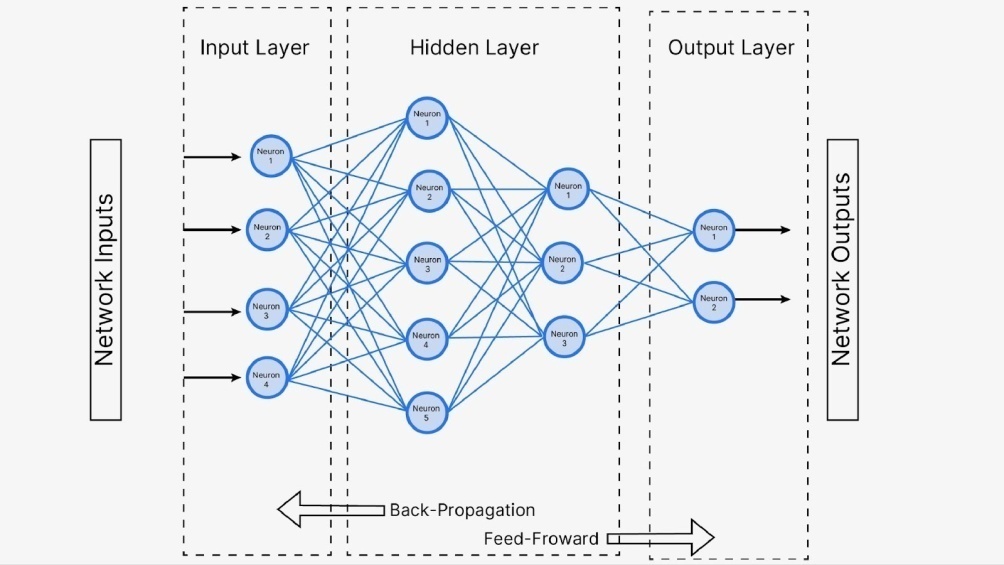
The logistic regression is considered as predictive analysis. Logistic regression is mainly used to describe data and use to explain the relationship between the dependent binary variable and one or many nominal or independent variables.



1. **Use a simple perceptron with weights *w*0, *w*1, and *w*2 as −1, 2, and 1, respectively, to classify data points (3, 4); (5, 2); (1, −3); (−8, −3); (−3, 0).**
2. **Explain the basic structure of a multi-layer perceptron. Explain how it can solve the XOR problem.**

## ANS:- Solving the XOR Problem with Neural Networks

To solve the XOR problem, we need to introduce multi-layer perceptrons (MLPs) and the backpropagation algorithm. MLPs are neural networks with one or more hidden layers between the input and output layers. These hidden layers allow the network to learn non-linear relationships between the inputs and outputs.

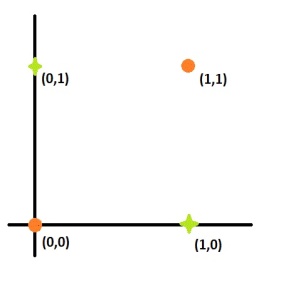


The backpropagation algorithm is a learning algorithm that adjusts the weights of the neurons in the network based on the error between the predicted output and the actual output. It works by propagating the error backwards through the network and updating the weights using gradient descent.

In addition to MLPs and the backpropagation algorithm, the choice of activation functions also plays a crucial role in solving the XOR problem. Activation functions introduce non-linearity into the network, allowing it to learn complex patterns. Popular activation functions for solving the XOR problem include the sigmoid function and the hyperbolic tangent function.

### The XOr problem

The XOr problem is that we need to build a Neural Network (a perceptron in our case) to produce the truth table related to the XOr logical operator. This is a binary classification problem. Hence, **supervised learning** is a better way to solve it. In this case, we will be using perceptrons. Uni layered perceptrons can only work with linearly separable data. But in the following diagram drawn in accordance with the truth table of the XOr loical operator, we can see that the data is **NOT** linearly separable.

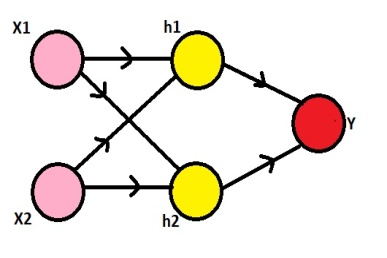


### The Solution

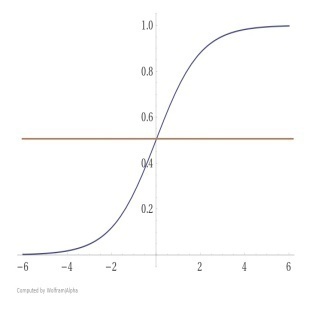
To solve this problem, we add an extra layer to our vanilla perceptron, i.e., we create a **Multi Layered Perceptron** (or **MLP**). We call this extra layer as the **Hidden layer**. To build a perceptron, we first need to understand that the XOr gate can be written as a combination of AND gates, NOT gates and OR gates in the following way:

a **XOr** b = (a **AND NOT** b)**OR**(b**AND NOT**a)

The following is a plan for the perceptron.



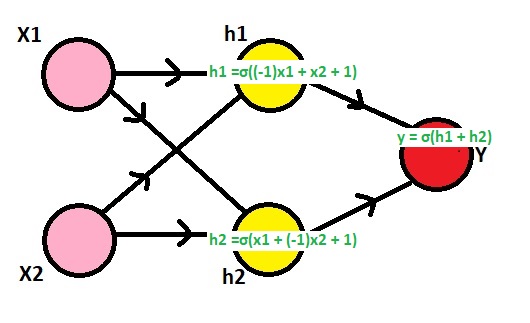
Here, we need to observe that our inputs are 0s and 1s. To make it a XOr gate, we will make the h1 node to perform the (x2 **AND NOT** x1) operation, the h2 node to perform (x1 **AND NOT** x2) operation and the y node to perform (h1 **OR** h2) operation. The NOT gate can be produced for an input a by writing (1-a), the AND gate can be produced for inputs a and b by writing (a.b) and the OR gate can be produced for inputs a and b by writing (a+b). Also, we'll use the sigmoid function as our activation function σ, i.e., σ(x) = 1/(1+e^(-x)) and the threshold for classification would be 0.5, i.e., any x with σ(x)>0.5 will be classified as 1 and others will be classified as 0.



Now, since we have all the information, we can go on to define h1, h2 and y. Using the formulae for AND, NOT and OR gates, we get:

1. h1 = σ((1-x1) + x2) = σ((-1)x1 + x2 + 1)
2. h2 = σ(x1 + (1-x2)) = σ(x1 + (-1)x2 + 1)
3. y = σ(h1 + h2) = σ(h1 + h2 + 0)

Hence, we have built a multi layered perceptron with the following weights and it predicts the output of a XOr logical operator.



**Explanation**

The truth table for a two-input XOR-Gate is given below,

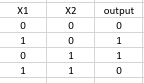


Fig 1.1 : XOR-Gate Truth Table

We want to get outputs as shown in the above truth table. For this purpose, we have made an MLP (Multilayer Perceptron) architecture shown below.

Here, the circles are neurons(O1, N1, N2, X1, X2), and the orange and blue lines with numbers are the representation of input direction with weights. The numbers on the arrows represent weights.B1 and B2 represent biases.

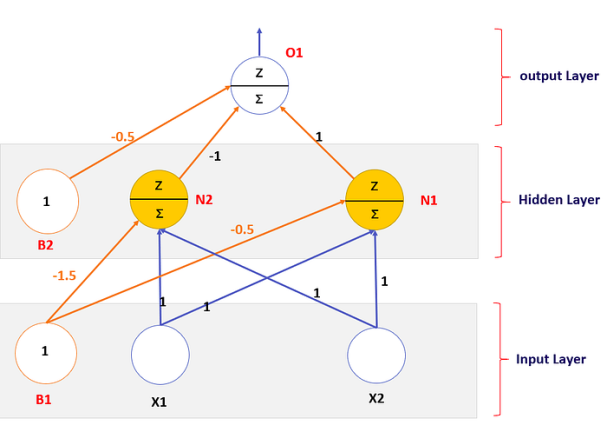
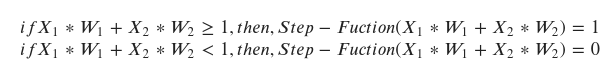


Fig 1.2:XOR-Gate representation using perceptrons.

**Step function:**

The step function (z) ,triggers only if the weighted sum is 1 or greater than 1. That is to say ,



Equation 1.1 :Defining step function

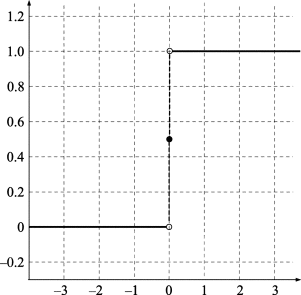


Fig 1.3 :Step function in graph

**Calculation of XOR gate output**

Recall that if we get a value of 1 or greater than 1 for the weighted sum, we will get a value of 1 as an output of the step function otherwise we will get a value of 0 in the output.

**Row 1 ,Truth table Fig(1.1),**

The XOR gate truth table says, if X1 = 0 and X2 =0 ,the output should be 0 .

*For hidden layer neuron N1 (Fig1.2),*

https://miro.medium.com/v2/resize:fit:302/1*vCJoyJaFwf-WBZ10RyfWYQ.png

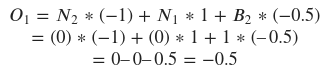
So ,step\_function(-0.5) = 0 ,output of **N1 = 0**

*For the hidden layer neuron N2 (fig1.2),*

https://miro.medium.com/v2/resize:fit:367/1*2brybAG7_M11TbIUxMyJ3g.png

So , step\_function(-1.5) = 0 ,output of **N2 = 0**

*For the output neuron O1 (fig1.2),*



So, step\_function(-0.5) = 0 ,output of **O1 = 0**

Matched with the Fig 1.1 ,XOR truth table first row.

**Row 2,Truth table Fig(1.1),**

The XOR gate truth table says, if X1 = 1 and X2 =0 ,the output should be 1 .

*For the hidden layer neuron N1 (fig1.2),*

https://miro.medium.com/v2/resize:fit:582/1*OcSyMVT1KDRd0WTdacNwrQ.png

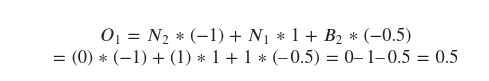
So, step\_function(0.5) = 1 ,output of **N1 = 1**

*For the hidden layer neuron N2 (fig1.2),*

https://miro.medium.com/v2/resize:fit:594/1*k4W0Wo4nnp1uOiOWYsYMtg.png

So, step\_function(-0.5) = 0 ,output of **N2 = 0**

*For the output neuron O1 (fig1.2),*



So, step\_function(0.5) = 1 ,output of **O1 = 1**

Matched with the Fig 1.1 ,XOR truth table second row.

**Row 3,Truth table Fig(1.1),**

The XOR gate truth table says, if X1 = 0 and X2 =1 ,the output should be 1.

*For the hidden layer neuron N1 (fig1.2),*

https://miro.medium.com/v2/resize:fit:622/1*i2wyJqE-1PlvwmSHGEEq4A.png

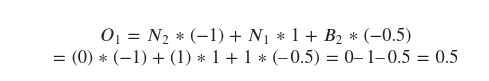
So, step\_function(0.5) = 1 ,output of **N1 = 1**

*For the hidden layer neuron N2 (fig1.2),*

https://miro.medium.com/v2/resize:fit:601/1*CCrPb1HFPKJCv_SD4N88HA.png

So, step\_function(-0.5) = 0 ,output of **N2 = 0**

*For the output neuron O1 (fig1.2),*



So, step\_function(0.5) = 1 ,output of **O1 = 1**

Matched with the Fig 1.1 ,XOR truth table third row.

**Row 4,Truth table Fig(1.1),**

The XOR gate truth table says, if X1 = 1 and X2 =1, the output should be 0.

*For the hidden layer neuron N1 (fig1.2),*

https://miro.medium.com/v2/resize:fit:609/1*0iw45CB0zu90VzDhFQSUcA.png

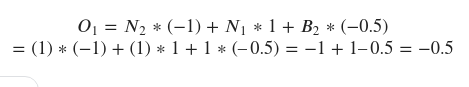
So, step\_function(1.5) = 1 ,output of **N1 = 1**

*For the hidden layer neuron N2 (fig1.2),*

https://miro.medium.com/v2/resize:fit:606/1*M4EKRU7aIQiKNlIdF2HWVQ.png

So, step\_function(0.5) = 1 ,output of **N2 = 1**

*For the output neuron O1 (fig1.2),*

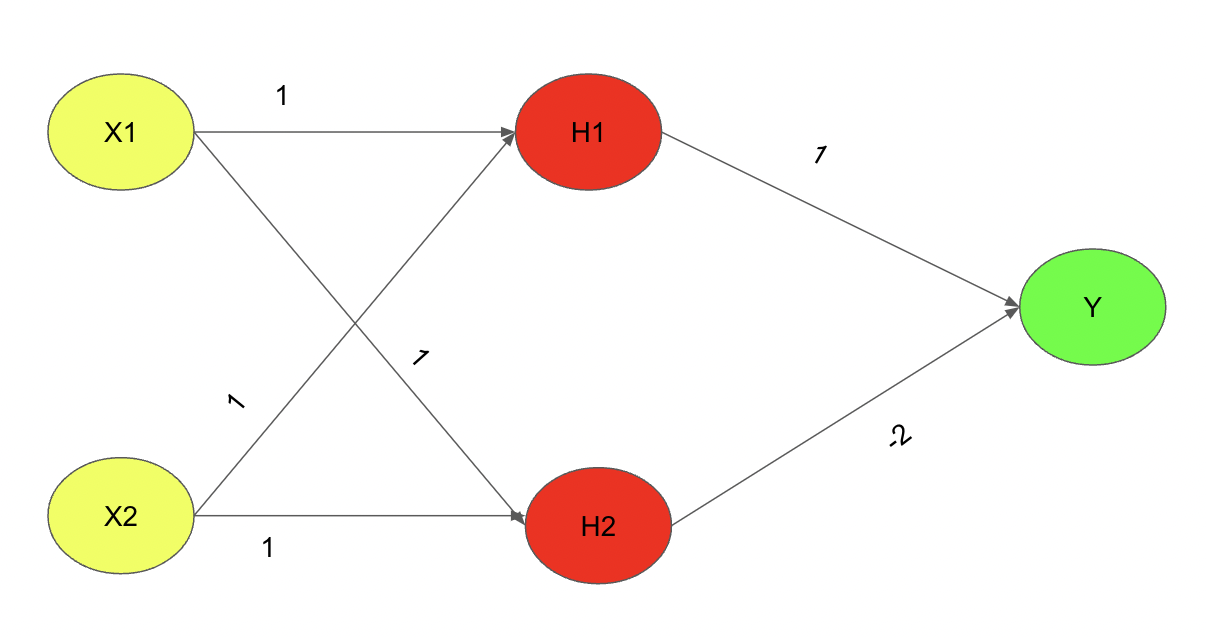


So, step\_function(-0.5) = 1 ,output of **O1 = 0**

Matched with the Fig 1.1 ,XOR truth table fourth row.

## ****How to solve the XOR problem with neural networks?****

The XOR problem with neural networks can be solved by using Multi-Layer Perceptrons or a neural network architecture with an input layer, hidden layer, and output layer. So during the forward [propagation](https://analyticsindiamag.com/how-to-visualize-backpropagation-in-neural-networks/) through the neural networks, the weights get updated to the corresponding layers and the XOR logic gets executed. The Neural network architecture to solve the XOR problem will be as shown below.



So with this overall architecture and certain weight parameters between each layer, the XOR logic output can be yielded through forward propagation. The overall neural network architecture uses the Relu activation function to ensure the weights updated in each of the processes to be 1 or 0 accordingly where for the positive set of weights the output at the particular neuron will be 1 and for a negative weight updation at the particular neuron will be 0 respectively. So let us understand one output for the first input state

**Example**:  For X1=0 and X2=0 we should get an input of 0. Let us solve it.

Solution: Considering X1=0 and X2=0

H1=RELU(0.1+0.1+0) = 0

H2=RELU(0.1+0.1+0)=0

So now we have obtained the weights that were propagated from the input layer to the hidden layer. So now let us propagate from the hidden layer to the output layer

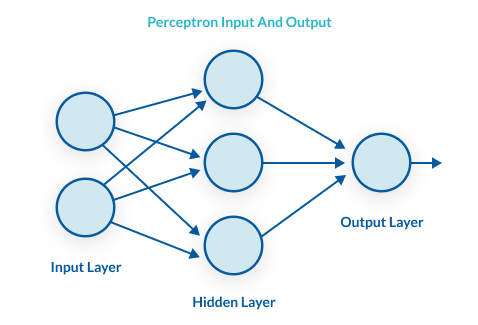
Y=RELU(0.1+0.(-2))=0

This is how multi-layer neural networks or also known as Multi-Layer perceptrons (MLP) are used to solve the XOR problem and for all other input sets the architecture provided above can be verified and the right outcome for XOR logic can be yielded.

**How is the XOR problem solved?**

The solution to the XOR problem lies in multidimensional analysis. We plug in numerous inputs in various layers of interpretation and processing, to generate the optimum outputs.

The inner layers for deeper processing of the inputs are known as hidden layers. The hidden layers are not dependent on any other layers. This architecture is known as Multilayer Perceptron (MLP).



The layers in a perceptron

The number of layers in MLP is not fixed and thus can have any number of hidden layers for processing. In the case of MLP, the weights are defined for each hidden layer, which transfers the signal to the next proceeding layer.

Using the MLP approach lets us dive into more than two dimensions, which in turn lets us separate the outputs of XOR using multidimensional equations.

Each hidden unit invokes an activation function, to range down their output values to 0 or 1.

The MLP approach also lies in the class of feed-forward Artificial Neural Network, and thus can only communicate in one direction. MLP solves the XOR problem efficiently by visualizing the data points in multi-dimensions and thus constructing an n-variable equation to fit in the output values.

1. **What is artificial neural network (ANN)? Explain some of the salient highlights in the different architectural options for ANN.**

**Ans :-**

An [Artificial Neural Network (ANN)](https://www.geeksforgeeks.org/implementing-ann-training-process-in-python/) is an information processing paradigm that is inspired by the brain. ANNs, like people, learn by examples. 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.

Artificial Neural Networks (ANNs) are a type of machine learning model that are inspired by the structure and function of the human brain. They consist of layers of interconnected “neurons” that process and transmit information.

There are several different architectures for ANNs, each with their own strengths and weaknesses. Some of the most common architectures include:

Feedforward Neural Networks: This is the simplest type of ANN architecture, where the information flows in one direction from input to output. The layers are fully connected, meaning each neuron in a layer is connected to all the neurons in the next layer.

Recurrent Neural Networks (RNNs): These networks have a “memory” component, where information can flow in cycles through the network. This allows the network to process sequences of data, such as time series or speech.

Convolutional Neural Networks (CNNs): These networks are designed to process data with a grid-like topology, such as images. The layers consist of convolutional layers, which learn to detect specific features in the data, and pooling layers, which reduce the spatial dimensions of the data.

Autoencoders: These are neural networks that are used for unsupervised learning. They consist of an encoder that maps the input data to a lower-dimensional representation and a decoder that maps the representation back to the original data.

Generative Adversarial Networks (GANs): These are neural networks that are used for generative modeling. They consist of two parts: a generator that learns to generate new data samples, and a discriminator that learns to distinguish between real and generated data.

The model of an artificial neural network can be specified by three entities: 

* **Interconnections**
* [**Activation functions**](https://www.geeksforgeeks.org/activation-functions-neural-networks/)
* **Learning rules**

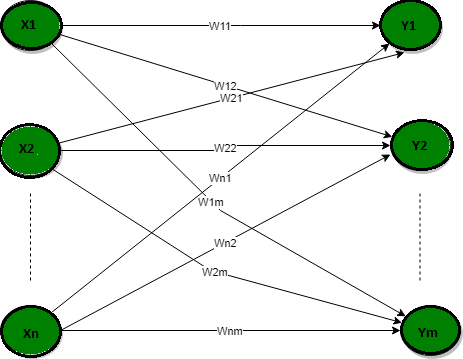
### Interconnections:

Interconnection can be defined as the way processing elements (Neuron) in ANN are connected to each other. Hence, the arrangements of these processing elements and geometry of interconnections are very essential in ANN.   
These arrangements always have two layers that are common to all network architectures, the Input layer and output layer where the input layer buffers the input signal, and 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.

There exist five basic types of neuron connection architecture :

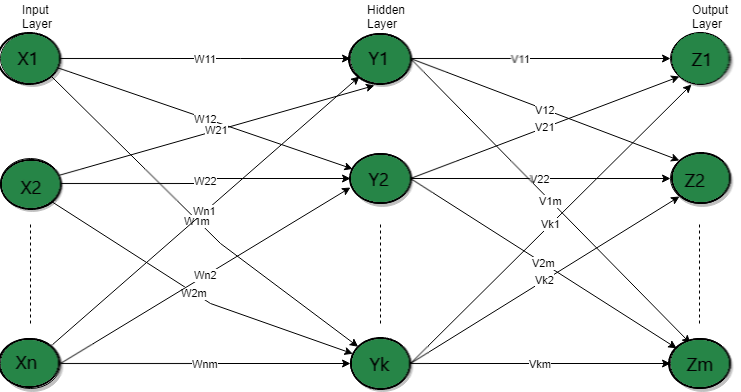
1. Single-layer feed-forward network
2. Multilayer feed-forward network
3. Single node with its own feedback
4. Single-layer recurrent network
5. Multilayer recurrent network

**1.** **Single-layer feed-forward network**



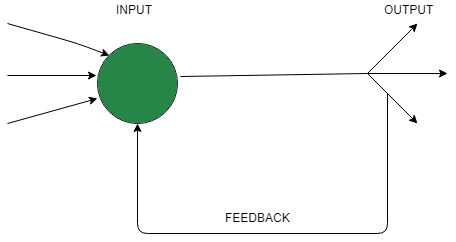
In this type of network, we have only two layers input layer and the output layer but the input layer does not count because no computation is performed in this layer. The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken. After this,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

**2.** **Multilayer feed-forward network**



This layer also has a hidden layer that is internal to the network and has no direct contact with the external layer. The existence of one or more hidden layers enables the network to be computationally stronger, a feed-forward network because of information flow through the input function, and the intermediate computations used to determine the output Z. There are no feedback connections in which outputs of the model are fed back into itself.

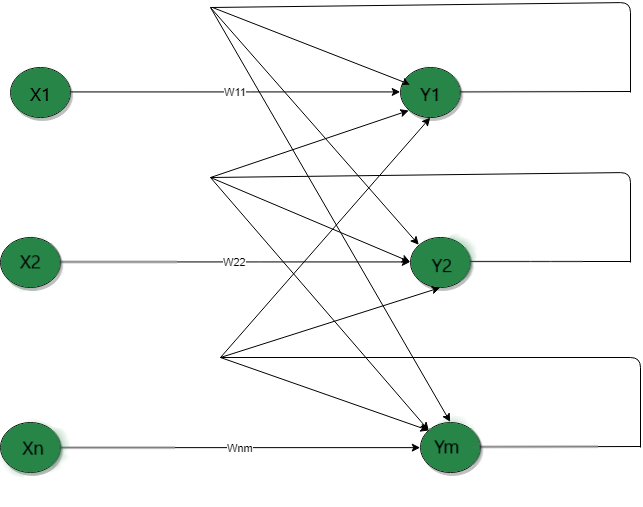
**3.** **Single node with its own feedback** 



*Single Node with own Feedback*

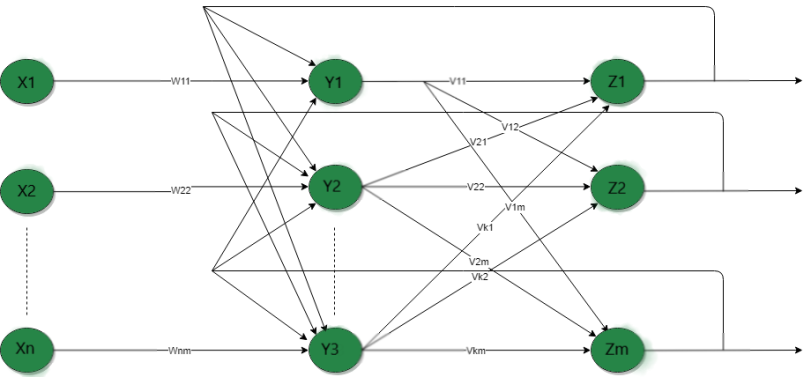
When outputs can be directed back as inputs to the same layer or preceding layer nodes, then it results in feedback networks. Recurrent networks are feedback networks with closed loops. The above figure shows a single recurrent network having a single neuron with feedback to itself.

**4.** **Single-layer recurrent network**



The above 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 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.

**5.** **Multilayer recurrent network** 



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. 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.

# Artificial Neural Network Tutorial

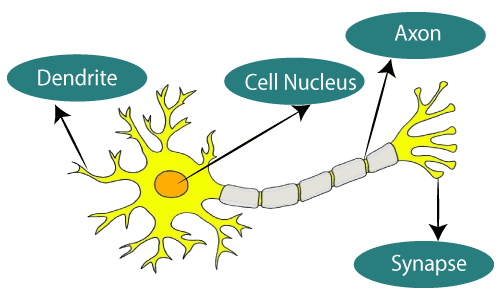
Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions.

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

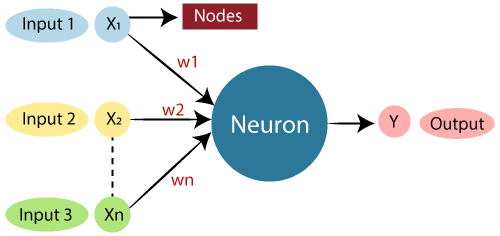
## What is Artificial Neural Network?

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.

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.

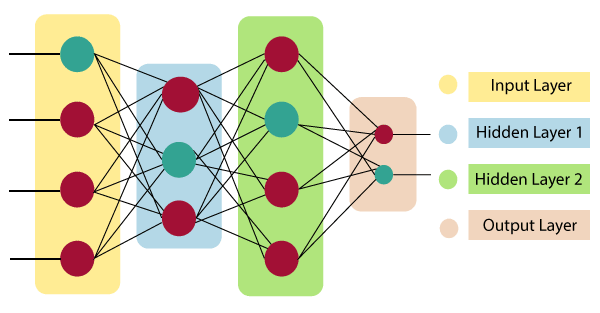
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly.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



**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.

What is Artificial Neural Network

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.

## Advantages of Artificial Neural Network (ANN)

**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.

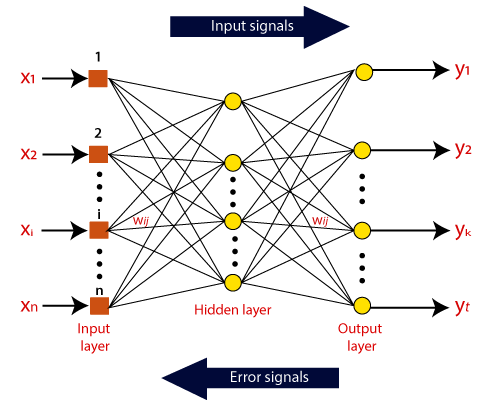
**The duration of the network is unknown:**

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

Science artificial neural networks that have steeped into the world in the mid-20th century are exponentially developing. In the present time, we have investigated the pros of artificial neural networks and the issues encountered in the course of their utilization. It should not be overlooked that the cons of ANN networks, which are a flourishing science branch, are eliminated individually, and their pros are increasing day by day. It means that artificial neural networks will turn into an irreplaceable part of our lives progressively important.

## 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.

## Types of Artificial Neural Network:

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification.

### Feedback ANN:

In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. As per the **University of Massachusetts**, Lowell Centre for Atmospheric Research. The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

### Feed-Forward ANN:

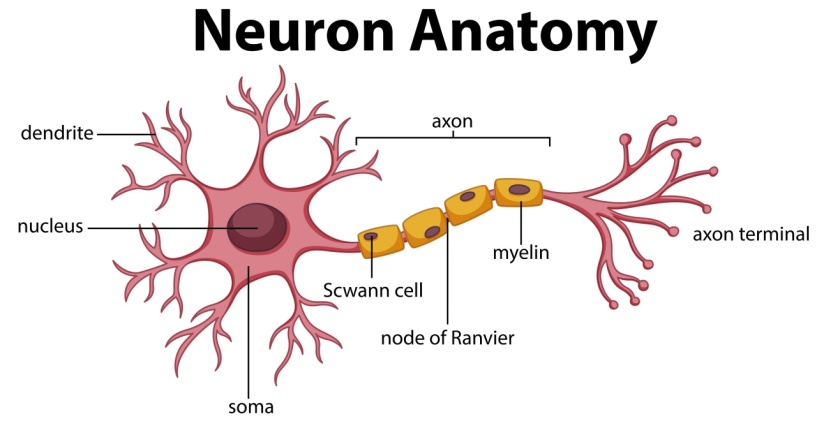
A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

**ANN – General Introduction:**

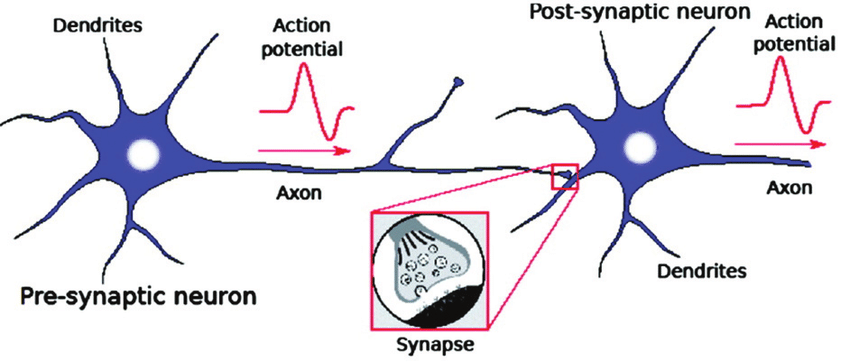
Artificial Neural Networks (ANN)are the basic algorithms and also simplified methods used in Deep Learning (DL) approach. We have come across more complicated and high-end models in the DL approach. However, ANN is a vital element of the progressive procedure and is the first stage in the DL algorithm.

Before wetting our hands over ANN, we have to comprehend the importance or existence of DL. ANN is established by the human brain activity, therefore our neural system can be understood by ANN as a way of transmitting our information via neurons to our brain. Thus, through some significant layman terms, we shall learn about ANN in forthcoming topics.

**Neuron – Concept Introduction to form as ANN:**

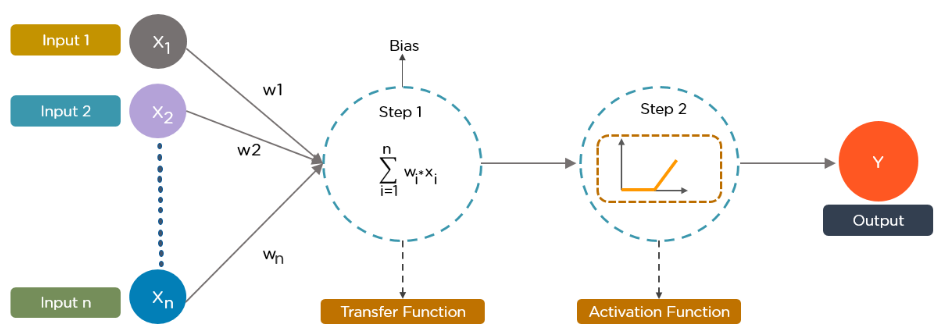


In our school time, most of them find this diagram. In simple terms, the fundamental job of the neuron is to receive and transmit impulses/information in various parts of our own body. In a simple, neuron which forges building blocks of our human nervous system. By the way, there will be no single neuron, the building block generates millions and millions of neurons. Just think of the circumstance as a car accident happened in front of our flat, what our brain will think immediately, go for help if feasible to seek assistance, if not contacting the ambulance.



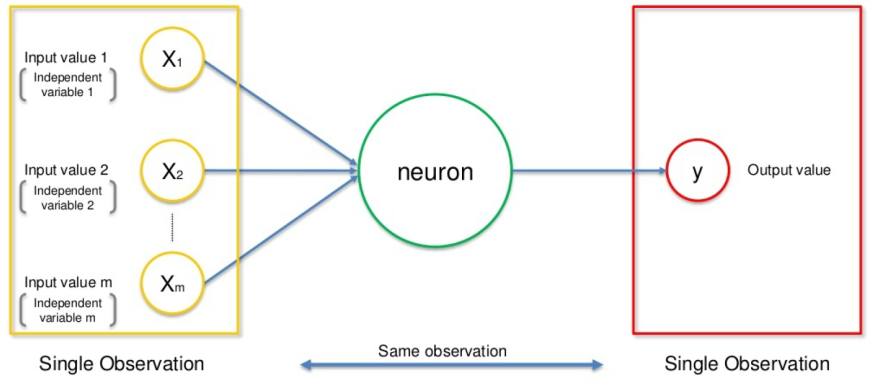
Here, while we have seen so many neuronal elements, Dendrites are regarded as a signal receiver and Axon as a signal transmitter. In this figure, to better understand, it shows a signal from a neuron that passes along its axon and links or transmits to dendrites of another neuron (Axon is not technically touched dendrites of another neuron, the process is called Synapse for receiving and transmitting the signals). What if the concepts mentioned above are artificial? What if? Artificial Neural Network’s name comes there!

**Neural Network – Basic foundation to understand:**



Now biological notions, as demonstrated in the previous subject, become a technological approach in this graphical representation above. So let’s assume, x1, x2…xn is considered as multiple neurons (presumed to be multiple sensory) transmitting signal to a signal neuron. Let us assume a scenario for the above diagram, you are going to watch a horror movie with your friends in a theatre, so there is a particular scene where the whole theatre gets erupted for a horror scene and you are having horror phobia (fear of watching horror scene), so for that scene what you do? assume x1 – eyes, x2 – ears, and x3 – mouth are some of the inputs, then what about your outputs be like? You will automatically close your eyes, close your ears and scream more (as like the voice in the movie is not audible) as per the scenario. In short, it can be picturized as,

icturized as,



We have terms like w1, w2, w3 which means weights: Weights are crucial for ANN function because weights are how neural networks (NN) learn, by modifying the weights, the NN decides which signal is significant in each case and which signal is not relevant for neurons, or which signal is passed or which signal is not passed or to which intensity signal is passed, they are the things that can be adapted through the learning method, like when you are training ANN, you are basically adjusting all the weights, and that’s where Gradient  
Descent (GD) and Back Propagation (BP) comes into play, so for the same scenario, you will be having the priority right which part has the highest weightage to process – here by default we close our eyes right. Finally, the dotted line (blue lines) here in the above diagram means Synapse, communication  
between neurons.

To summarize in short, how neural network takes place

1. Signals from multiple neurons like x1, x2, x3… xn, along with associated weights w1, w2, w3… wn are transmitted to the successive neurons. (Input layers)

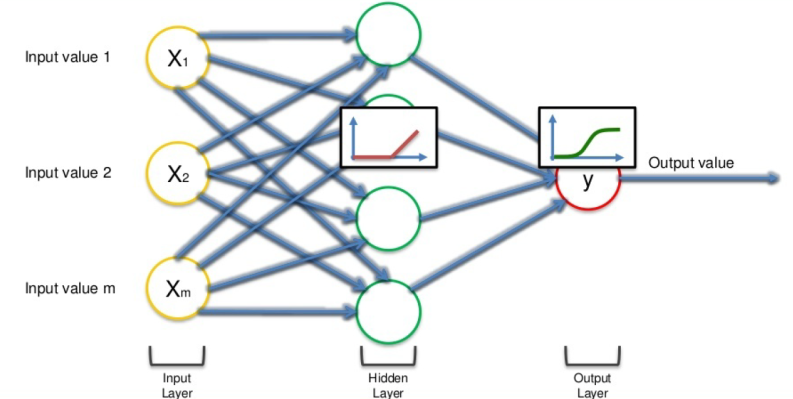
2. So after we are receiving multiple signals (here consider it as neurons), with weights associated with it, we will be summing up all the signals with weights, and that’s our step 1 as shown in the above figure (Hidden layers).

3. Then it applies the activation function as step 2 as shown in the above figure (Hidden layers).

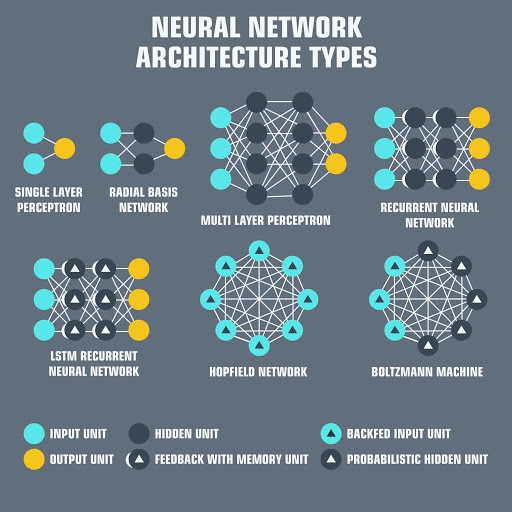
4. From the above step neuron understand if it needs to pass the signal or not.

5. If the neuron passes the signal (that’s to another neuron), that’s our output (Output layers).

6. The above process keeps on repeating until the last neuron or the part of our body here.



Finally, NN is a sequence of algorithms that imitate human brain functions to recognize connections between huge volumes of data. This is the basis for the DL approach, and NN can be categorized as stated below based on the architecture,



## What is the Activation Function? – Superpower for ANN

Image Source: towardsdatascience.com

So what actually Activation Function (AF) do? AF is a function that is added into an ANN to help the network learn complex patterns in the data. When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell. It’s a non-linear transformation that we do for inputs that we receive before sending them to the next layer or neuron. AF is also known as Transfer Function.

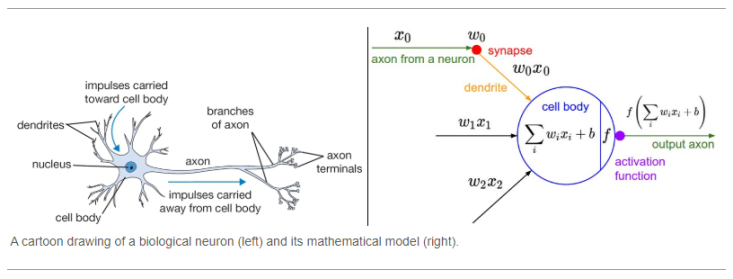
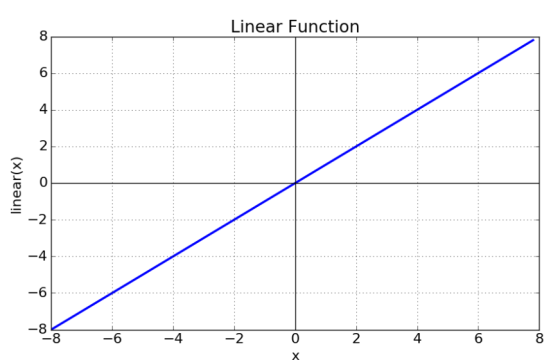
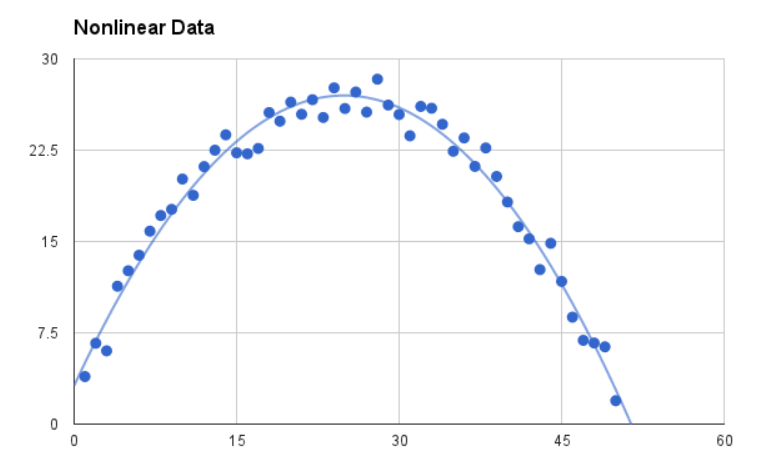


Image Source: images.google.com

So basically AF can be classified into linear AF and non-linear AF. In simple, without AF, the output signal will be a linear function (like polynomial of one degree or as a linear regression model) – they have less power to learn the complex functional mapping from data. AF captures a non-linear relationship between the inputs and also converts the inputs to useful output.



But our brain functions non-linearly for complex calculation, robustness, and high parallelism, ability to handle imprecise and fuzzy information. So for this to mimic we need AF, to work in the complicated, high dimensional, and non-linear large dataset that has complicated architecture (having many hidden layers between input and output layers).



To understand the non-linear Af, we need to understand two terminologies like,

1. **Differential function:** Change in Y-axis w.r.t change in X-axis (slope!)

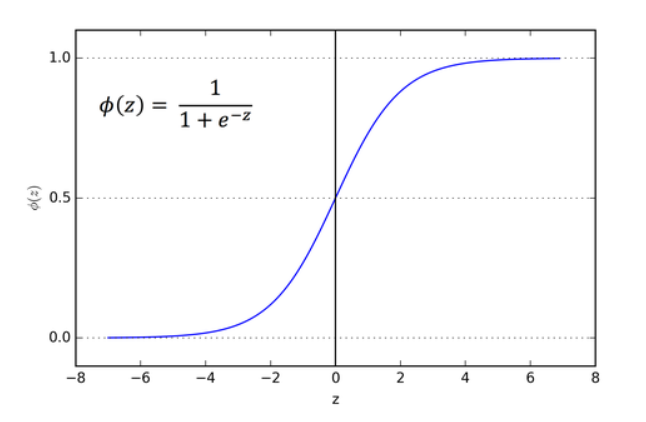
2. **Monotonic function:** A function which either entirely non decreasing or non increasing. (Non-linear!)

Some important key features for AF,

1. Vanishing Gradient problem
2. Zero-Centered
3. Computational inexpensive
4. Differentiable

There are some important types of the non-linear activation function (classified based on curves),

### 1. Sigmoid function



Bullet points for sigmoid function,

1. Zero is the threshold function here, if it is less than zero, then it passes to zero, if it is equal to zero or greater than zero then, it passes to one, as simple.

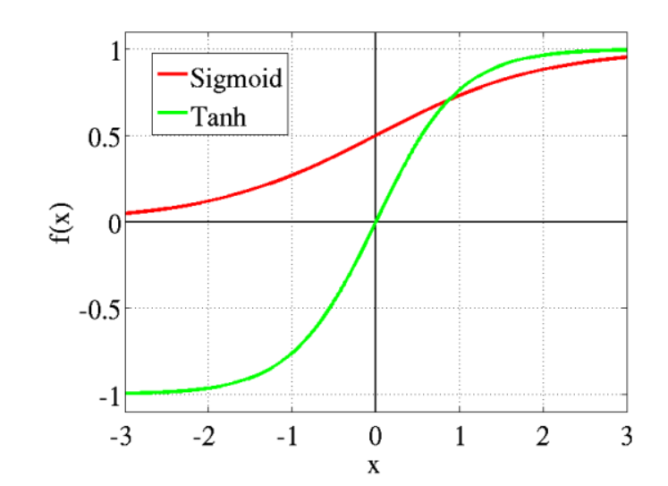
2. Application: Predominately used for binary classification

3. Its differentiable means we can find the slope at any two points

4. It is a monotonic function

5. Mostly won’t be used in real-world examples, because of computationally inexpensive, causes vanishing gradient problems, and not zero-centered.

### 2. Hyperbolic Tangent function (Tanh)



Bullet points for Hyperbolic Tangent function,

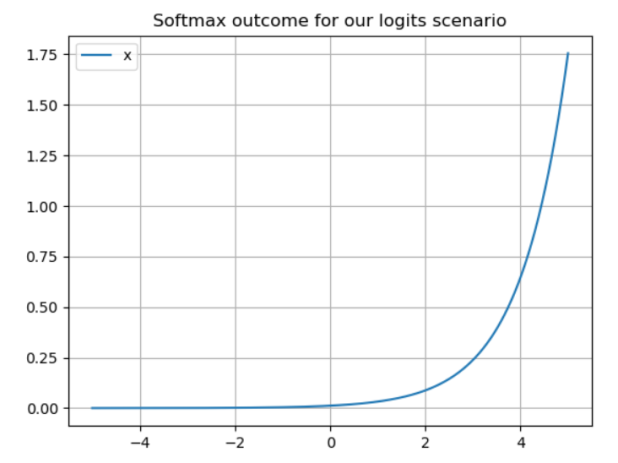
1. Better version of Sigmoid function (range: -1 to +1), goes below zero.

2. It is a differentiable function

3. It is a monotonic function

4. Application: Used for two-class problems, Used in feedforward nets.

### 3. Softmax function



Bullet points for softmax function,

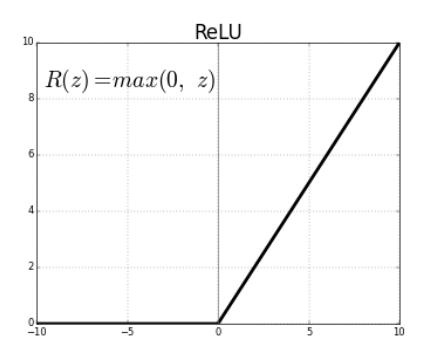
1. This particular AF works in the output layer

2. Need to apply a standard exponential function to all the values in the output layer, then do the normalization by dividing the sum of all exponentials. (Sum of all makes to 1)

3. Application: Multi-class classification problems (confidence score for each class)

4. It is a differentiable function

### 4. Rectified Linear Unit (ReLU) function



Bullet points for Rectifier Linear Unit (ReLU) function,

1. Most used in CNN and many DL approaches because does not cause vanishing gradient problem.

2. If the value is less than zero, then the function becomes zero, if the value is equal to zero or greater than the function becomes the number. – dying ReLU (which causes some nodes to completely die and not learn anything)

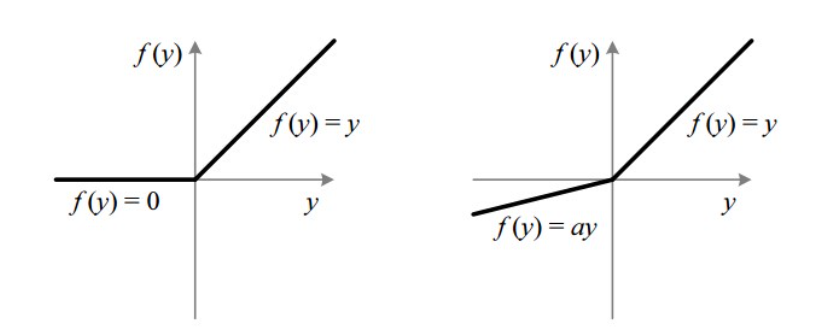
3. It is a differentiable function

4. It is not a monotonic function

5. Range of this AF is o to infinity

6. Disadvantage: If any value is less than zero then automatically function makes it to zero, which makes the negative value not mapped properly except but zero and also leads to the unstable node.

### 5. Leaky ReLU



Bullet points for Leaky Rectifier Linear Unit function,

1. Advance function of ReLU, and it solves negative impact of ReLU function

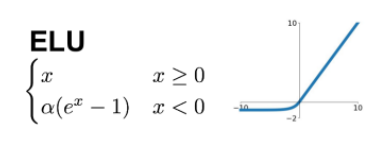
2. Usually the “a” value in the above diagram is around 0.01 only, if it exceeds means it became Randomized ReLU (mostly we stick with ReLU and sometimes with Leaky ReLU AF only)

3. Range of Leaky ReLU is – infinity to + infinity

4. It is a differentiable function

5. It is a monotonic function

### 6. Exponential Linear Units (ELUs) function



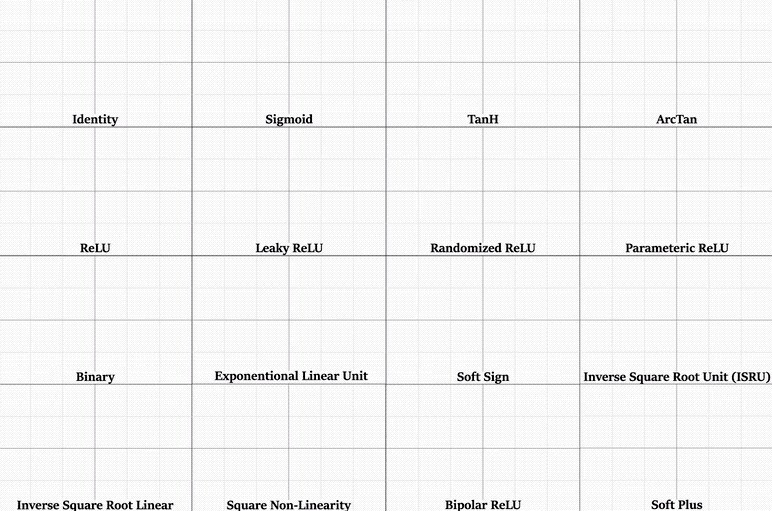
Bullet points for Exponential Linear Unit function,

1. Advance to ReLU and Leaky ReLU AF

2. Because of the negative impact of the above-mentioned AFs, it causes mean unit activation greater than zero, hence it rectifies this, ELU AF comes to the picture and makes mean activation closer to zero, hence it makes the model faster learning and convergence

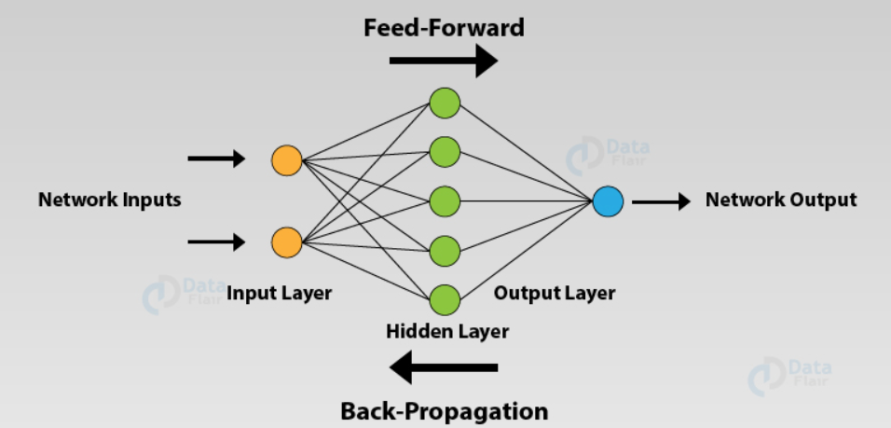
3. It is a differentiable function

and some other AF are picturized below,



**Note:**Among all, Tanh and Sigmoid AF can’t be used more in real-time because of vanishing gradient problems.

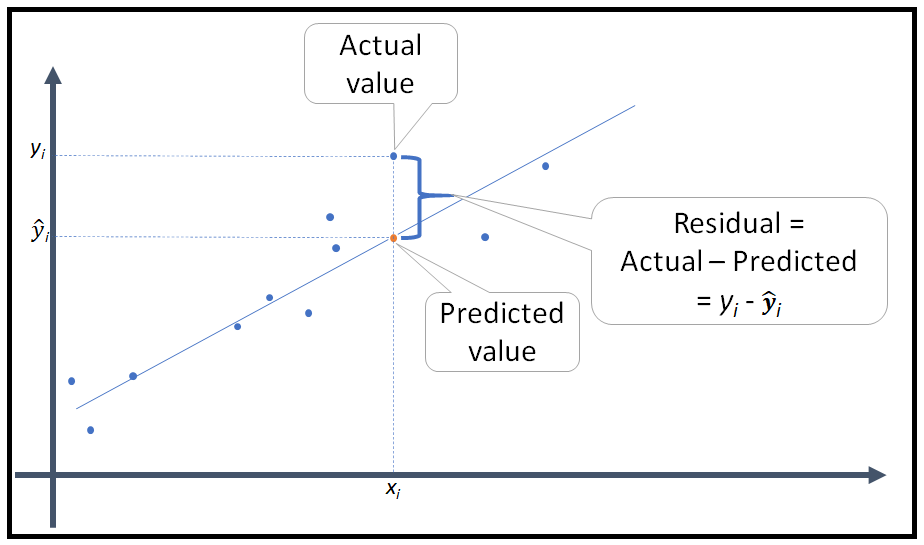
## How NN learn? – Some interesting topics ALERT!



Before getting into the topic, we must know about two basic terminologies like Feed-forward and Back Propagation network. In simple term,

1. Feed Forward network (FFN) – If we achieved output what we expected from the input without turning back or fine-tuning.

2. Back Propagation Network (BPN) – Opposite to FFN, if the output that we got is not as expected and we need to turn back for fine-tuning it to the expected output. (learning and adjusting!)

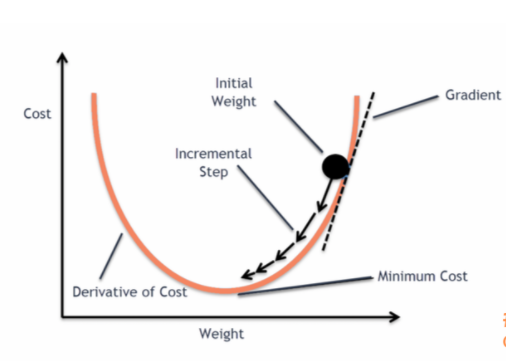


So here in this picture, don’t see the residual concept, consider only actual and predicted values alone, here there will be two scenarios,

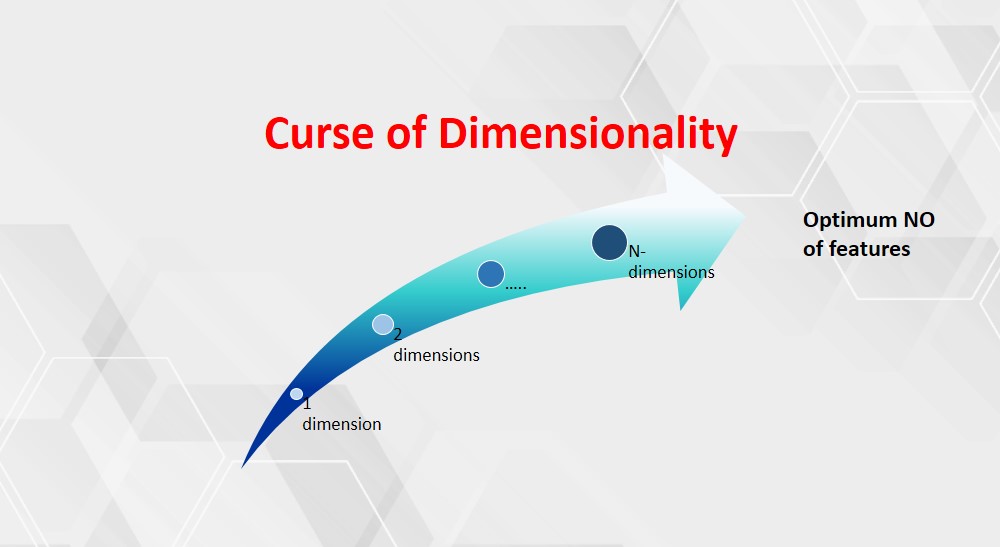
1. If the predicted output equals actual output then no worries.

2. If the predicted output varies with actual output, then we need to find the difference between the outputs, which we call as COST (error) function, our main goal is to reduce it. So we go for BPN (weights get updated and then proceed until cost function is reduced but not zero!)

### Gradient Descent:



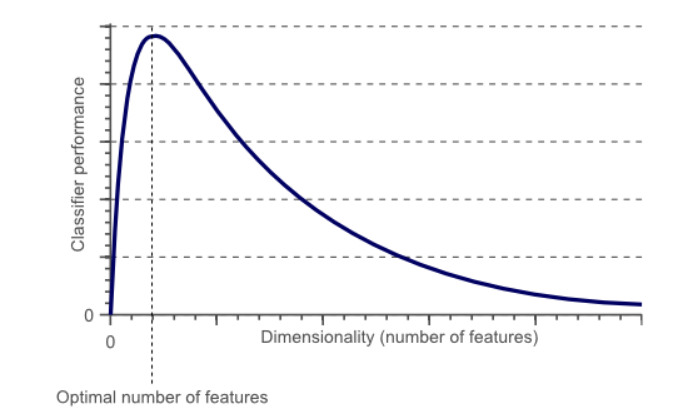
Let’s assume that we have a data set with fewer characteristics (independent variables), it is very obvious from the figure above that when we generate series of cost values, and we can find the least-cost value in a  
simple model while updating different weights. If our dataset has more characteristics or separate variables, we must suffer a curse of dimensionality (CoD).



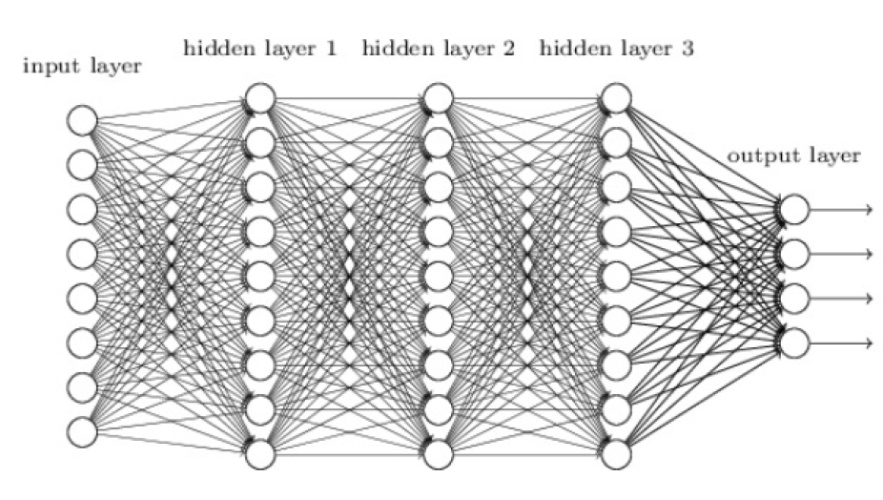
As the dimentionality of the features space increases, the number of configurations can grow exponentially and thus the number of configurations covered by an observation increases.

The above picture, which explains the layman concept about CoD.

So according to Huges Phenomenon (below diagram), the curse of dimensionality, if the features are added, the accuracy of the classification model is increased until the ideal number of features are achieved and the accuracy is further diminished



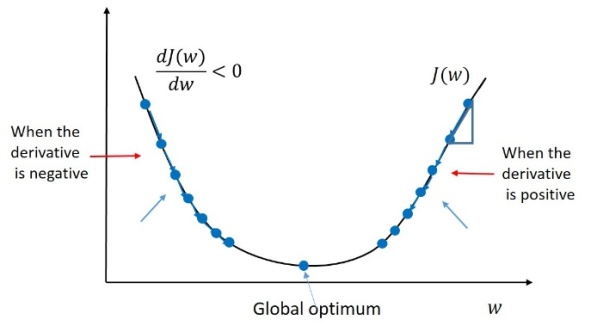
So let us consider a neural network model with more number hidden layers and independent features as shown below or even greater than that, obviously, we will face CoD. So for this problem, one of the solutions is Gradient Descent



So let’s presume how GD works? Consider this is the cost function of our model and blue dot which implies different iterations (here different weights adjustment), we will get different cost functions. We start from the left side first, for the top value let’s take the slope value, the value may be positive or negative irrespective of the values (numbers), if our value is negative then we need to move towards the downside, the same way in the right side top value again if we calculate the slope, if we get the negative value again move downwards or else upwards, so likewise we need to find out for all the values. Finally when we find the minimum value (best weights probably!) among all others is our minimum cost value function.

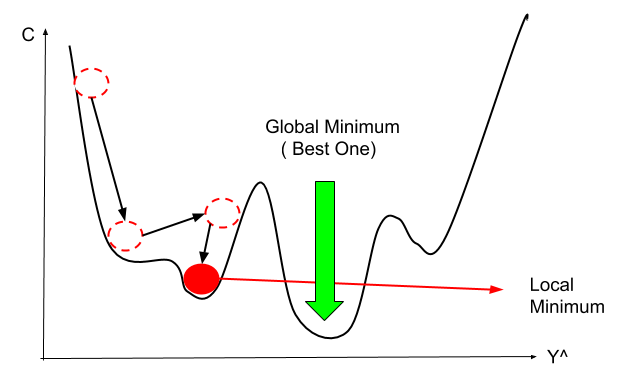
**Limitation:** Convex cost function (depends on the shape from cost values)

**Gradient Descent = Descending into Minimum of COST function**

Image Source: images.google.com

### Stochastic Gradient Descent:

The main drawback of GD is limited to convex function (shape). Suppose for our dataset, while calculating cost function and then the shape of the cost function assume as like below picture, in that scenario can we proceed with same GD, the simple answer is no!, because if you note in the same figure, we have multiple minimum values (local minimum values), rather than the optimal value (global minimum), for this situation we opt for stochastic gradient descent optimization technique (SGD).

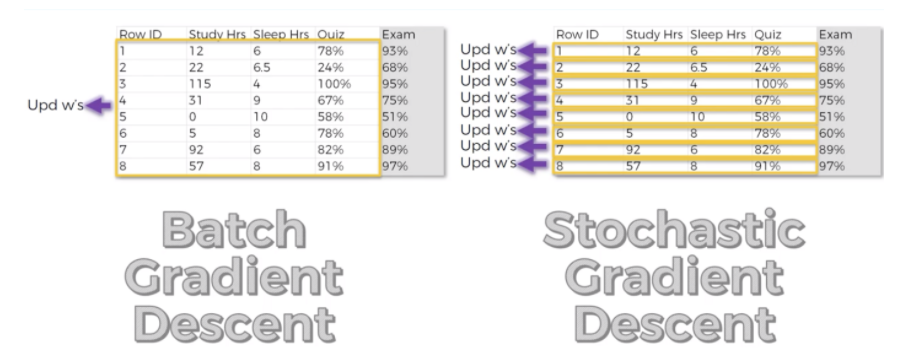
Image Source: images.google.com

The only difference between GD and SGD are,

1. In GD, we will take all rows (here samples we call), plug them into our network model, then find the output value and then correlate with actual value, find the cost function, then revert (by backpropagation network) adjust the weights and proceed further for different values among them, we will find the minimum values of the cost function.

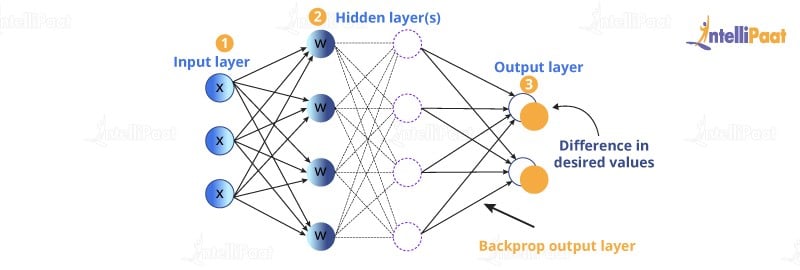
2. In SGD, we follow the same steps as GD only one difference is instead of applying all samples we go for each row separately and update the weights.

For more clarification between GD and SGD, find out the below picture,



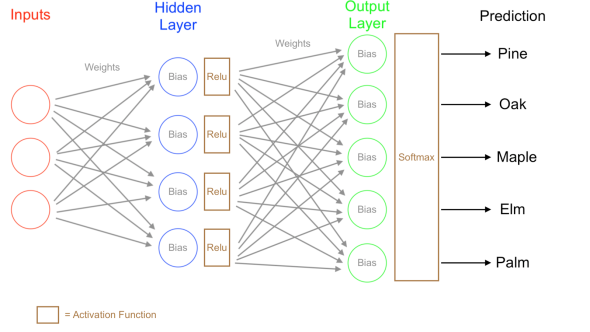
**Mini Batch Gradient Descent = Gradient Descent + Stochastic Gradient Descent**

### Back Propagation:



As we know, BPN means after finding the output value, we are going to compare with the actual value, then we will find the errors, after that to minimize the error, we are backpropagated then do some adjustments with weights then again we proceed until we get the minimum error. BPN is an advanced algorithm, which has more complex mathematics for adjusting weights simultaneously. The main advantage of BPN is the way the algorithm is structured because we can adjust the weights simultaneously and also basically we know which part of the NN which is part of the weight is responsible for the error.

## How network works? – Overall Summary!



Bullet points on how NN works in general,

1. the First layer is the input layer as we all know.

2. the Last layer is the output layer that’s by default.

3. Assume here we don’t have a hidden layer, we need to calculate the price of the house, and let’s consider the inputs are area, no. of bedrooms, distance from the city, amenities available, age of the property, hospitals and schools for kids, so the price will be simply the summation of the product between input’s and weightage priority and with AF we will calculate the price. So it’s almost like a Machine learning model.

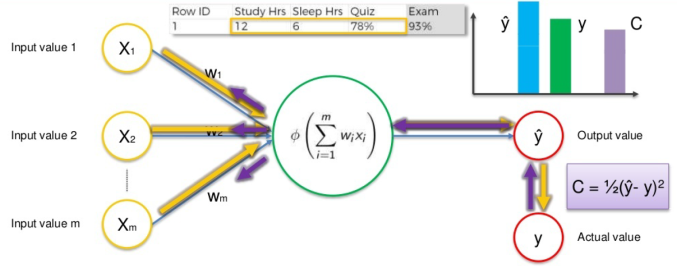
4. So what makes DL approach something better than the Machine Learning model is the addition of a hidden layer, it gives some flexibility and power, to increase accuracy.

5. Let us consider, for the same example above all the inputs are connected to all the neurons in the hidden layer, which means it won’t have the same weightage from the input layer, which means depending on the priorities the weightage values changes (Higher priority have high values!) Ex: We all know the flat rate is more in the city rather than the flat rate in outer of the city for the same square feet likewise it changes with other inputs as we mentioned. Each neuron in the hidden layer has some priorities and weightage for application-related like how our brain works?

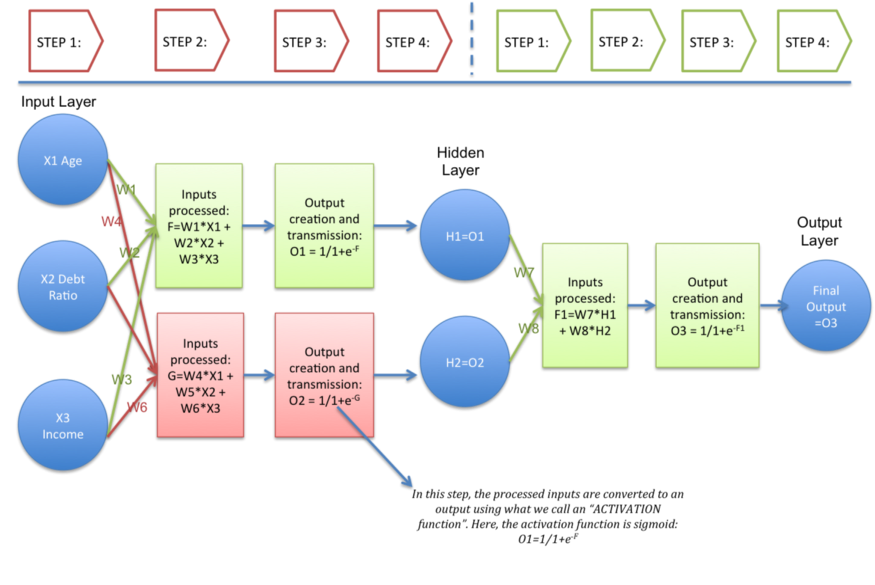
6. If we keep on increasing the hidden layers for some importance or weightage or for solving complex calculations, the accuracy gets increased, it’s like fine-tuning the network to get the optimal output.

7. We will be using two AF for a simpler NN, i.e, 1st between input and hidden layer, 2nd between hidden and output layer.

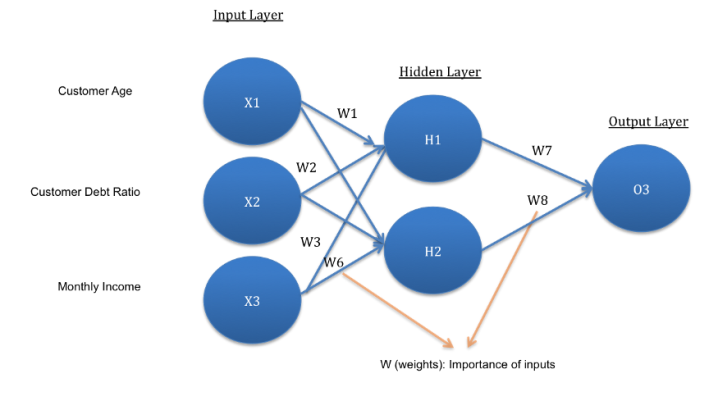
The below picture summarizes the whole concept,  here let us assume the inputs x1, x2 be the study hours, sleeping hours, and quizzes, so with weightage given it is passed to the hidden layer and it performs the summation and by applying AF, we get the output values. then it is compared with the actual value, if the error value is more, that we call it a cost function then again it was backpropagated then adjusted the weight, then again process continued until we get minimal cost value, and the optimization process is called gradient descent, if the whole samples (rows) processed and updating the weights or Stochastic gradient descent if each row is processed and updates it’s the weight (mostly we will be using this because of it’s robust nature)



The below steps which summarize all,



Overall, ANN can be picturized in general as,



**Advantage:**

1. Ability to work with incomplete knowledge

2. Fault tolerance

3. Parallel processing capability

**Disadvantage:**

1. Hardware dependence

2. Duration of the network is unknown

1. **Explain the learning process of an ANN. Explain, with example, the challenge in assigning synaptic weights for the interconnection between neurons? How can this challenge be addressed?**

**ANS:-**

**Architecture and Learning process in neural network**

In order to learn about Backpropagation**,**we first have to understand the architecture of the neural network and then the learning process in ANN. So, let’s start about knowing the various architectures of the ANN:

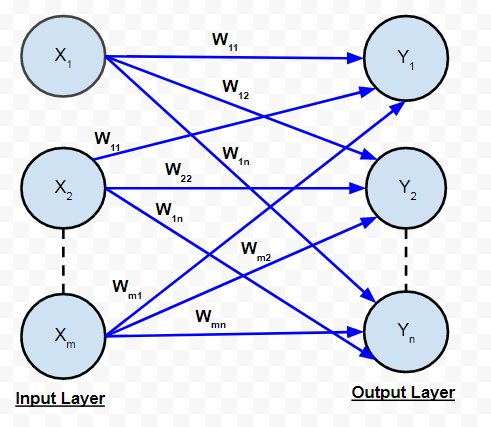
#### ****Architectures of Neural Network:****

ANN is a computational system consisting of many interconnected units called **artificial neurons**. The connection between artificial neurons can transmit a signal from one neuron to another. So, there are multiple possibilities for connecting the neurons based on which the **architecture** we are going to adopt for a specific solution. Some permutations and combinations are as follows:

* There may be just two layers of neuron in the network – the input and output layer.
* There can be one or more intermediate **‘hidden’** layers of a neuron.
* The neurons may be connected with all neurons in the next layer and so on …..

So let’s start talking about the various possible architectures:

**A. Single-layer Feed Forward Network:**

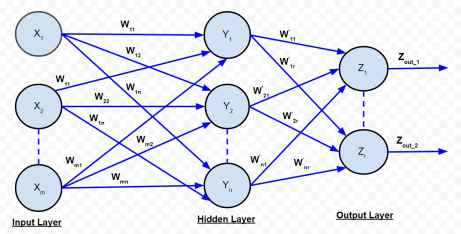


It is the simplest and most basic architecture of ANN’s. It consists of only two layers- the input layer and the output layer. **The input layer** consists of ‘m’ input neurons connected to each of the ‘n’ output neurons. The connections carry weights w11 and so on. The input layer of the neurons doesn’t conduct any processing – they pass the i/p signals to the o/p neurons. The computations are performed in the output layer. So, though it has 2 layers of neurons, only one layer is performing the computation. This is the reason why **the network is known as SINGLE** layer. Also, the signals always flow from the input layer to the output layer. Hence, the **network is known as FEED FORWARD.**

The net signal input to the output neurons is given by:

The signal output from each output neuron will depend on the activation function used.

**B. Multi-layer Feed Forward Network:**



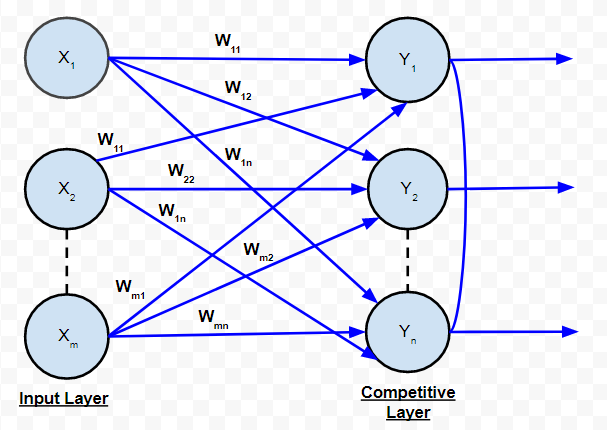
***Multi-Layer Feed Forward Network***

The multi-layer feed-forward network is quite similar to the single-layer feed-forward network, except for the fact that there are one or more intermediate layers of neurons between the input and output layer. Hence, the **network is termed as multi-layer.**Each of the layers may have a varying number of neurons. For example, the one shown in the above diagram has ‘m’ neurons in the input layer and ‘r’ neurons in the output layer and there is only one hidden layer with ‘n’ neurons.

for the kth hidden layer neuron. The net signal input to the neuron in the output layer is given by:

**C. Competitive Network:**

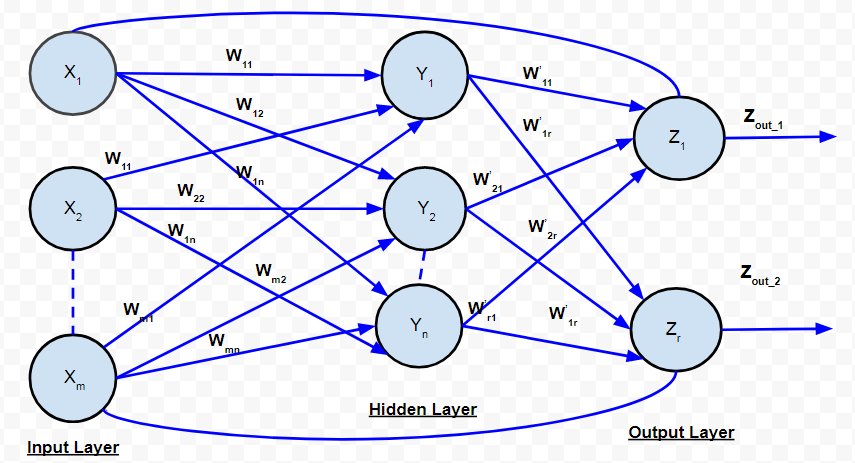
It is as same as the single-layer feed-forward network in structure. The only difference is that**the output neurons are connected with each other (either partially or fully)**. Below is the diagram for this type of network.



***Competitive Network***

According to the diagram, it is clear that few of the output neurons are interconnected to each other. For a given input, the output neurons compete against themselves to represent the input. It represents a form of an unsupervised learning algorithm in ANN that is suitable to find the clusters in a data set.

**D. Recurrent Network:**



***Recurrent Network***

In feed-forward networks, the signal always flows from the input layer towards the output layer (in one direction only). In the case of recurrent neural networks, there is **a feedback loop** (from the neurons in the output layer to the input layer neurons). There can be self-loops too.

#### ****Learning Process In ANN:****

Learning process in ANN mainly depends on four factors, they are:

1. **The number of layers in the network (Single-layered or multi-layered)**
2. **Direction of signal flow (Feedforward or recurrent)**
3. **Number of nodes in layers:**The number of node in the input layer is equal to the number of features of the input data set. The number of output nodes will depend on possible outcomes i.e. the number of classes in case of supervised learning. But the number of layers in the hidden layer is to be chosen by the user. A larger number of nodes in the hidden layer, higher the performance but too many nodes may result in overfitting as well as increased computational expense.
4. **Weight of Interconnected Nodes:**Deciding the value of weights attached with each interconnection between each neuron so that a specific learning problem can be solved correctly is quite a difficult problem by itself. Take an example to understand the problem. Take the example of a**Multi-layered Feed-Forward Network,**we have to train an ANN model using some data, so that it can classify a new data set, sayp\_5(3,-2). Say we have deduced that p\_1=(5,2)   and  p\_2 = (-1,12)   belonging to class C1 while p\_3=(3,-5)   and p\_4 = (-2,-1)  belonging to class C2. We assume the values of synaptic weights w\_0,w\_1,w\_2 as -2, 1/2 and 1/4 respectively. But we will NOT get these weight values for every learning problem. For solving a learning problem with ANN, we can start with a set of values for synaptic weights and keep changing those in multiple iterations. The stopping criterion may be the **rate of misclassification < 1% or the maximum numbers of iterations should be less than 25(a threshold value).** There may be another problem that, the rate of misclassification may not reduce progressively.

So, we can summarize the learning process in ANN as the combination of – **deciding the number of hidden layers,**the **number of nodes in each of the hidden layers,**the **direction of signal flow, deciding the connection weight.**

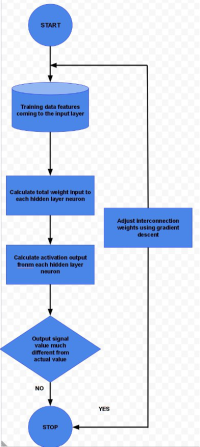
**Multi-layer feed network**is a commonly used architecture. It has been observed that a neural network with even one hidden layer can be used to reasonably approximate any continuous function. The learning methodology adopted to train a multi-layer feed-forward network is **Backpropagation**.

#### ****Backpropagation:****

In the above section, we get to know that the most critical activities of training an ANN are to assign the inter-neuron connection weights. In 1986, an efficient way of training an ANN was introduced. In this method, the **difference in output values of the output layer and the expected values, are**propagated**back from the output layer to the**preceding**layers.** Hence, the algorithm implementing this method is known as BACK PROPAGATION**i.e. propagating the errors back to the**preceding**layers.**

The backpropagation algorithm is applicable for multi-layer feed-forward network. It is a supervised learning algorithm which continues adjusting the weights of the connected neurons with an objective to reduce the deviation of the output signal from the target output. This algorithm consists of multiple iterations, **known as epochs.** Each epoch consists of two phases:

* **Forward Phase:**Signal flow from neurons in the input layer to the neurons in the output layer through the hidden layers. The weights of the interconnections and activation functions are used during the flow. In the output layer, the output signals are generated.
* **Backward Phase:**Signal is compared with the expected value. The computed errors are propagated backwards from the output to the preceding layer. The error propagated back are used to adjust the interconnection weights between the layers.

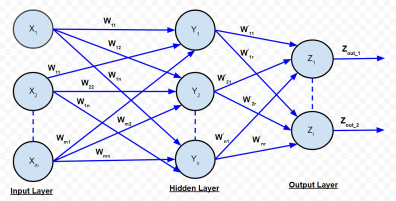


***BACKPROPAGATION***

The above diagram depicts a reasonably simplified version of the back propagation algorithm.

One main part of the algorithm is adjusting the interconnection weights. This is done using a technique termed as **Gradient Descent**. In simple words, the algorithm calculates the partial derivative of the activation function by each interconnection weight to identify the ‘gradient’ or extent of change of the weight required to minimize the cost function.

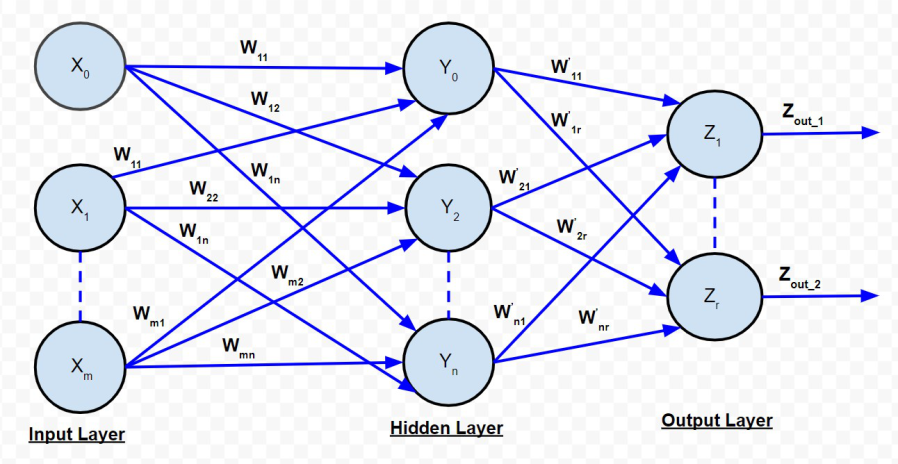
In order to understand the back propagation algorithm in detail, let us consider the **Multi-layer Feed Forward Network.**



**The net signal input to the hidden layer neurons is given by:**

If  is the activation function of the hidden layer, then

**The net signal input to the output layer neurons is given by:**



***BACKPROPAGATION NET***

Note that the signals  and  are assumed to be 1. If  is the activation function of the hidden layer, then

**If is the target of the k-th output neuron, then the cost function defined as the squared error of the output layer is given by:**

**According to the descent algorithm**, partial derivative of cost function E has to be taken with respect to interconnection weights. Mathematically it can be represented as:

{Above expression is for the interconnection weights between the **j-th neuron in the hidden layer and the k-th neuron in the output layer**.} **This expression can be reduced to**

**where,  or**

**If we assume** as a component of the weight adjustment needed for weight  corresponding to the k-th output neuron, then :

On the basis of this, the weights and bias need to be updated as follows:

* **For weights:**
* **Hence,**
* **For bias:**
* **Hence,**

In the above expressions, alpha is the learning rate of the neural network.Learning rate is a user parameter which decreases or increases the speed with which the interconnection weights of a neural network is to be adjusted. If the learning rate is too high, the adjustment done as a part of the gradient descent process may diverge the data set rather than converging it. On the other hand, if the learning rate is too low, the optimization may consume more time because of the small steps towards the minima.

{All the above calculations are for the interconnection weight between neurons in the hidden layer and neurons in the output layer}

Like the above expressions, we can deduce the expressions for “Interconnection weights between the input and hidden layers:

* **For weights:**
* **Hence,**
* **For bias:**
* **Hence,**

So, in this way, we can use the Backpropagation algorithm to solve various Artificial Neural Networks.

Artificial Neural Network Tutorial

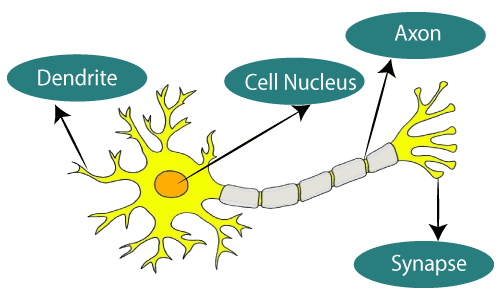
Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions.

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

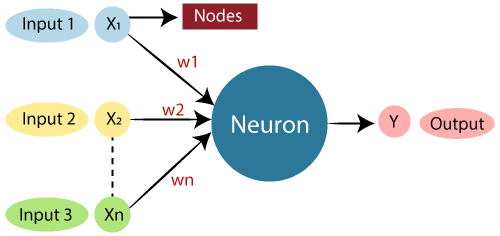
What is Artificial Neural Network?

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.

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.

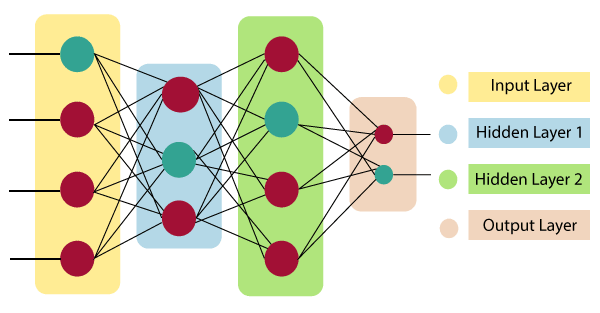
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly.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



**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.

ADVERTISEMENT

What is Artificial Neural Network

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.

Advantages of Artificial Neural Network (ANN)

**Parallel processing capability:**

ADVERTISEMENT

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.

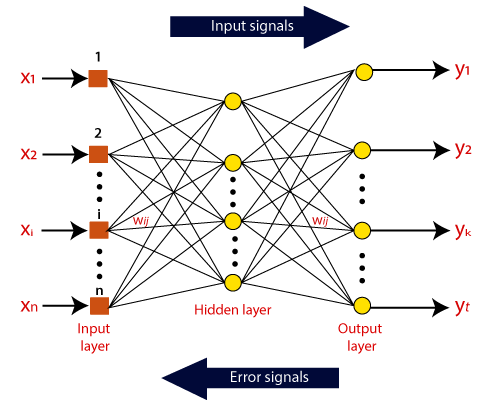
**The duration of the network is unknown:**

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

*Science artificial neural networks that have steeped into the world in the mid-20th century are exponentially developing. In the present time, we have investigated the pros of artificial neural networks and the issues encountered in the course of their utilization. It should not be overlooked that the cons of ANN networks, which are a flourishing science branch, are eliminated individually, and their pros are increasing day by day. It means that artificial neural networks will turn into an irreplaceable part of our lives progressively important.*

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.

Types of Artificial Neural Network:

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification.

## What is an Artificial Neural Network?

These are computational models inspired by the human brain. Many of the recent advancements have been made in the field of Artificial intelligence, including Voice Recognition, Image recognition, and Robotics using it. They are the biologically inspired simulations performed on the computer to perform certain specific tasks like

1. Classification

2. Pattern Recognition

**In general -**  It is a biologically inspired network of artificial neurons configured to perform specific tasks. These biological methods of computing are known as the next major advancement in the Computing Industry.

### What is a Neural Network?

The term ‘Neural’ has origin from the human (animal) nervous system’s basic functional unit ‘neuron’ or nerve cells present in the brain and other parts of the human (animal) body. A neural network is a group of algorithms that certify the underlying relationship in a set of data similar to the human brain. The neural network helps to change the input so that the network gives the best result without redesigning the output procedure. You can also learn more about [ONNX](https://www.xenonstack.com/blog/onnx/) in this insight.

## Advantages and disadvantages of  Artificial Neural Network (ANN)

**The advantages are listed below**

1. A neural network can perform tasks that a linear program can not.  
2. When an element of the neural network fails, its parallel nature can continue without any problem.  
3. A neural network learns, and reprogramming is not necessary.  
4. It can be implemented in any application.  
5. It can be performed without any problem.

**The disadvantages are described below**

1. The neural network needs training to operate.  
2. The architecture of a neural network is different from the architecture of microprocessors. Therefore, emulation is necessary.  
3. Requires high processing time for large neural networks.

A combination of neurons whose performance vector signifies the creation of real instance parameters of a particular type of an object or it's part. Click to explore about our, [Capsule Networks Benefits](https://www.xenonstack.com/insights/capsule-networks)

### What are the parts of Neurons and their Functions?

The typical nerve cell of the human brain comprises four parts:

#### 1. Function of Dendrite

It receives signals from other neurons.

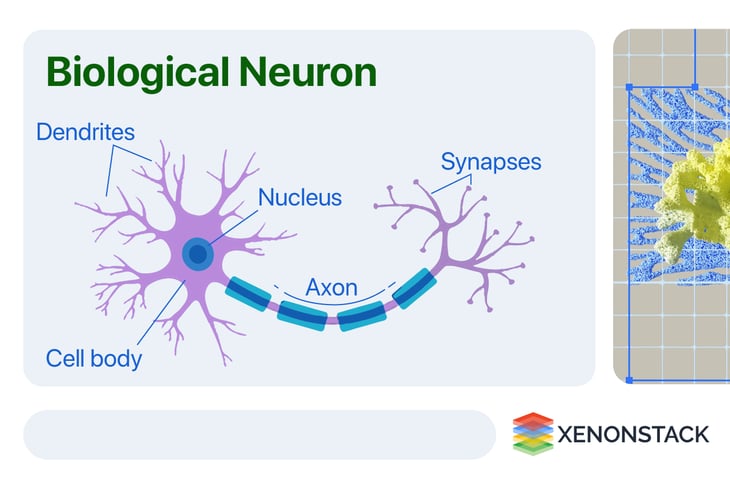
#### 2. Soma (cell body)

It sums all the incoming signals to generate input.

#### 3. Axon Structure

When the sum reaches a threshold value, the neuron fires and the signal travels down the axon to the other neurons.

#### 4. Synapses Working

The point of interconnection of one neuron with other neurons. The amount of signal transmitted depends upon the strength (synaptic weights) of the connections.The connections can be inhibitory (decreasing strength) or excitatory (increasing strength) in nature. So, a neural network, in general, has a connected network of billions of neurons with a trillion of interconnections between them.

## What is the difference between a Brain and a Computer?

### difference-between-brain-and-computer

## Unveiling the Distinctions: Artificial Neural Networks (ANN) vs Biological Neural Networks (BNN)

|  |  |  |
| --- | --- | --- |
| **Characteristics** | **Artificial Neural Network (ANN)** | **Biological(Real) Neural Network (BNN)** |
| **Speed** | Faster in processing information. Response time is in nanoseconds. | Slower in processing information. The response time is in milliseconds. |
| **Processing** | Serial processing. | Massively parallel processing. |
| **Size & Complexity** | Less size & complexity. It does not perform complex pattern recognition tasks. | A highly complex and dense network of interconnected neurons containing neurons of the order of 1011 with 1015 of interconnections.<strong |
| **Storage** | Information storage is replaceable means replacing new data with an old one. | A highly complex and dense network of interconnected neurons containing neurons of the order of 1011 with 1015 of interconnections. |
| **Fault tolerance** | Fault intolerant. Corrupt information cannot retrieve in case of failure of the system. | Information storage is adaptable means new information is added by adjusting the interconnection strengths without destroying old information. |
| **Control Mechanism** | There is a control unit for controlling computing activities | No specific control mechanism external to the computing task |

**Artificial Neural Networks with Biological Neural Network**

Neural Networks resemble the human brain in the following two ways -

1. A neural network acquires knowledge through learning.  
2. A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

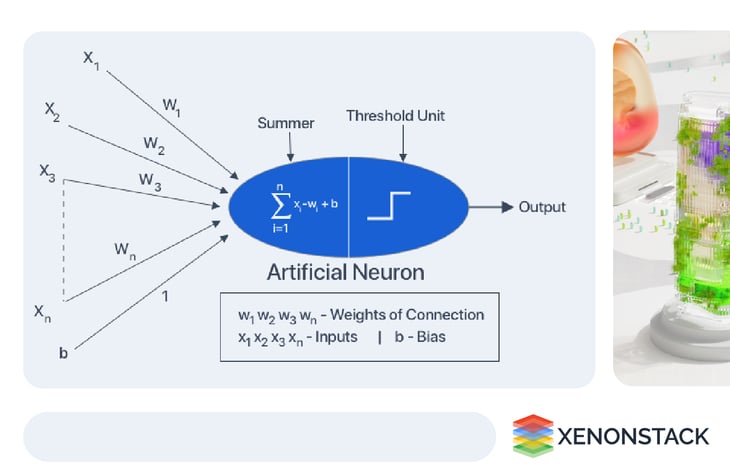
|  |  |
| --- | --- |
| Von Neumann architecture-based computing | Ann-Based Computing |
| Serial processing: processing instruction and problem rule one at the time (sequential) | Parallel processing - several processors perform simultaneously (multitasking) |
| Function logically with a set of if & else rules - rule-based approach | Function by learning pattern from a given input (image, text or video, etc.) |
| Programmable by higher-level languages such as C, [Java](https://www.xenonstack.com/blog/serverless-architecture/), C++, etc. | ANN is, in essence, the program itself. |
| Requires either big or error-prone parallel processors | Use of application-specific multi-chips. |

### Artificial Neural Network (ANN) vs. biological Neural Network (BNN)

### neural-network-pattern-recognition

1. The Biological Neural Network's dendrites are analogous to the weighted inputs based on their synaptic interconnection in it.  
2. The cell body is comparable to the artificial neuron unit in it, comprising summation and threshold unit.  
3. Axon carries output that is analogous to the output unit in the case of it. So, it is model using the working of basic biological neurons.

**How does Artificial Neural Networks work?**

**1.** It can be viewed as weighted directed graphs in which artificial neurons are nodes, and directed edges with weights are connections between neuron outputs and neuron inputs.

**2.**The Artificial Neural Network receives information from the external world in pattern and image in vector form. These inputs are designated by the notation x(n) for n number of inputs.

**3.** Every input is multiplied by its specific weights, which serve as crucial information for the neural network to solve problems. These weights essentially represent the strength of the connections between neurons within the neural network.

**4.**The weighted inputs are all summed up inside the computing unit (artificial neuron). In case the weighted sum is zero, bias is added to make the output not- zero or to scale up the system response. Bias has the weight and input always equal to ‘1'.

**5.** The sum corresponds to any numerical value ranging from 0 to infinity. To limit the response to arrive at the desired value, the threshold value is set up. For this, the sum is forward through an activation function.

**6.** The activation function is set to the transfer function to get the desired output. There are linear as well as the nonlinear activation function.

### What are the commonly used activation functions?

Some of the commonly used activation function is - binary, sigmoidal (linear) and tan hyperbolic sigmoidal functions(nonlinear).

**1. Binary**

The output has only two values, either 0 and 1. For this, the threshold value is set up. If the net weighted input is greater than 1, the output is assumed as one otherwise zero.  
**2. Sigmoidal Hyperbolic**

This function has an ‘S’ shaped curve. Here the tan hyperbolic function is used to approximate output from net input. The function is defined as - f (x) = (1/1+ exp(-????x)) where ???? - steepness parameter.

## What are the various types of Artificial Neural Networks?

|  |  |  |
| --- | --- | --- |
| ****Parameter**** | ****Types**** | ****Description**** |
| Based on the connection pattern | FeedForward, Recurrent | **Feedforward** - In which graphs have no loops. **Recurrent** - Loops occur because of feedback. |
| Based on the number of hidden layers | Single-layer, Multi-Layer | **Single Layer** - Having one secret layer. E.g., Single Perceptron **Multilayer** - Having multiple secret layers. Multilayer Perceptron |
| Based on the nature of weights | Fixed, Adaptive | **Fixed** - Weights are a fixed priority and not changed at all. **Adaptive** - Updates the weights and changes during training. |
| Based on the Memory unit | Static, Dynamic | **Static** - Memoryless unit. The current output depends on the current input. E.g., Feedforward network. **Dynamic** - Memory unit - The output depends upon the current input as well as the current output. E.g., Recurrent Neural Network |

### Neural Network Architecture Types

1. Perceptron Model in Neural Networks  
2. Radial Basis Function Neural Network  
3. Multilayer Perceptron Neural Network  
4. Recurrent Neural Network  
5. Long Short-Term Memory Neural Network (LSTM)  
6. Hopfield Network  
7. Boltzmann Machine Neural Network  
8. Convolutional Neural Network  
9. Modular Neural Network  
10. Physical Neural Network

### neural-network-architecture-type

#### 1. Perceptron Model

Neural Network is having two input units and one output unit with no hidden layers. These are also known as ‘single-layer perceptrons.'

#### 2. Radial Basis Function

These networks are similar to the feed-forward Neural Network, except the radial basis function is used as these neurons' activation function.

#### 3. Multilayer Perceptron

Unlike single-layer perceptron, these networks use more than one hidden layer of neurons. These are also known as Deep Feedforward Neural Networks.

#### 4. Recurrent

Type of Neural Network in which hidden layer neurons have self-connections. It possesses memory. At any instance, the hidden layer neuron receives activation from the lower layer and its previous activation value.

### 5. Long Short-Term Memory Neural Network (LSTM)

The type of Neural Network in which memory cell is incorporated into hidden layer neurons is called an LSTM network.

#### 6. Hopfield Network

A fully interconnected network of neurons in which each neuron is connected to every other neuron. The network is trained with input patterns by setting a value of neurons to the desired pattern. Then its weights are computed. The weights are not changed. Once trained for one or more patterns, the network will converge to the learned patterns. It is different from other Neural Networks.

#### 7. Boltzmann Machine Neural Network

These networks are similar to the Hopfield network, except some neurons are input, while others are hidden in nature. The weights are initialized randomly and learn through the backpropagation algorithm.

#### 8. Convolutional Neural Network

Get a complete overview of it through our blog Log Analytics with Machine Learning and Deep Learning.

#### 9. Modular Neural Network

It is the combined structure of different types of it like multilayer perceptron, Hopfield Networks, Recurrent Neural Networks, etc., which are incorporated as a single module into the network to perform independent subtasks of whole complete.

#### 10. Physical Neural Network

In this type of Artificial Neural Network, electrically adjustable resistance material is used to emulate synapses instead of software simulations performed in the neural network.

Artificial Intelligence collects and analyze data using smart sensors or machine learning algorithms and automatically route service requests to reduce the human workload. Click to explore about our, [Artificial Intelligence Applications](https://www.xenonstack.com/blog/artificial-intelligence)

## Hardware Architecture for Neural Networks

Two types of methods are used to implement hardware for it.

1. Software simulation in conventional computer  
2. A special hardware solution for decreasing execution time.  
When Neural Networks are used with fewer processing units and weights, software simulation is performed on the computer directly. E.g., voice recognition, etc. When Neural Network Algorithms developed to the point where useful things can be done with 1000's neurons and 10000's of synapses, high-performance Neural network hardware will become essential for practical operation. E.g., GPU ( [Graphical processing unit](https://www.intel.in/content/www/in/en/products/docs/processors/what-is-a-gpu.html" \t "_blank)) in the case of Deep Learning algorithms in object recognition, image classification, etc. The implementation's performance is measured by connection per the second number (cps), i.e., the number of the data chunk is transported through the neural network's edges. While the performance of the learning algorithm is measured in the connection updates per second (cups)

### Learning Techniques

The neural network learns by adjusting its weights and bias (threshold) iteratively to yield the desired output. These are also called free parameters. For learning to take place, the Neural Network is trained first. The training is performed using a defined set of rules, also known as the learning algorithm.

### Training Algorithms

**i. Gradient Descent Algorithm**  
This is the simplest training algorithm used in the case of a supervised training model. In case the actual output is different from the target output, the difference or error is find out. The gradient descent algorithm changes the weights of the network in such a manner as to minimize this mistake.

**ii. Back Propagation Algorithm**  
It is an extension of the gradient-based delta learning rule. Here, after finding an error (the difference between desired and target), the error is propagated backward from the output layer to the input layer via the hidden layer. It is used in the case of Multi-layer Neural Networks.

#### Learning Data Sets

**1. Training Data Set:** A set of examples used for learning is to fit the parameters [i.e., weights] of the network. One approach comprises one full training cycle on the training set.

**2. Validation Set Approach:** A set of examples used to tune the parameters [i.e., architecture] of the network. For example, to choose the number of hidden units in a Neural Network.

**3. Making Test Set:** A set of examples is used only to assess the performance [generalization] of a fully specified network or apply successfully to predict output whose input is known.

### What are the Five Algorithms to Train a Neural Network?

1. Hebbian Learning Rule  
2. Self-Organizing Kohonen Rule  
3. Hopfield Network Law  
4. LMS algorithm (Least Mean Square)  
5. Competitive Learning

## What is the Architecture of Artificial Neural Networks?

A typical Neural Network contains many artificial neurons called units arranged in layers. A typical Artificial Neural Network comprises different layers -

### ****1. Input layer****

It contains those units (Artificial Neurons) that receive input from the outside world on which the network will learn, recognize, or otherwise process.

### ****2. Output layer****

It contains units that respond to the information about how it learn any task.

### ****3. Hidden layer****

These units are in between the input and output layers. The hidden layer's job is to transform the input into something the output unit can use.

Connect Neural Networks, which means say, each hidden neuron links completely to every neuron in its previous layer(input) and the next layer (output) layer.

### What are the Learning Techniques in Neural Networks?

Here is a list of Learning Techniques

i. Supervised Learning  
ii. Unsupervised Learning  
iii. Reinforcement Learning  
iv. Offline Learning  
v. Online Learning

**Let's Discuss each one of them in length**

**i.** Supervised Learning  
In this learning, the training data is input to the network, and the desired output is known weights are adjusted until production yields the desired value.

**ii.** **Unsupervised Learning**  
Use the input data to train the network whose output is known. The network classifies the input data and adjusts the weight by feature extraction in input data.

**iii. Reinforcement Learning**  
Here, the output value is unknown, but the network provides feedback on whether the output is right or wrong. It is Semi-Supervised Learning.

**iv. Offline Learning**  
The weight vector and threshold adjustments are made only after the training set is shown to the network. It is also called Batch Learning.

**v. Online Learning**

The adjustment of the weight and threshold is made after presenting each training sample to the network.

### Learning and Development in Neural Networks

Learning occurs when the weights inside the network get updated after many iterations. For example - Suppose we have inputs in the form of patterns for two different classes of patterns - I & 0 as shown and b -bias and y as the desired output.

We want to classify input patterns into either pattern ‘I’ & ‘O.' The following are the steps performed:

**1.** Nine inputs from x1 - x9 and bias b (input having weight value 1) are fed to the network for the first pattern.

**2.** Initially, weights are initialized to zero.  
**3.** Then weights are updated for each neuron using the formulae: Δ wi = xi y for i = 1 to 9 (Hebb’s Rule)  
**4.** Finally, new weights are found using the formulas:  
**5.** wi(new) = wi(old) + Δwi  
**6.** Wi(new) = [111-11-1 1111]  
**7.** The second pattern is input to the network. This time, weights are not initialized to zero. The initial weights used here are obtained after presenting the first pattern. By doing so, the network.  
**8.** The steps from 1 - 4 are repeated for second inputs.  
**9.** The new weights are Wi(new) = [0 0 0 -2 -2 -2 000]  
So, these weights correspond to the learning ability of the network to classify the input patterns successfully.

## What are the Use Cases of Artificial Neural Networks?

There are four broad use cases of Neural Networks

i. Classification Neural Network  
ii. Prediction Neural Network  
iii. Clustering Neural Network  
iv. Association Neural Network

### Classification Neural Network

A neural network can be trained to classify a given pattern or dataset into a predefined class. It uses feedforward networks.

### Prediction Neural Network

A Neural Network can be trained to produce expected outputs from a given input. E.g., - Stock market prediction.

### ****Clustering Neural Network****

The neural network can identify a unique feature of the data and classify them into different categories without any prior knowledge of the data. The following networks are used for clustering -

1. Competitive networks  
2. Adaptive Resonance Theory Networks  
3. Kohonen Self-Organizing Maps.

### Association Neural Network

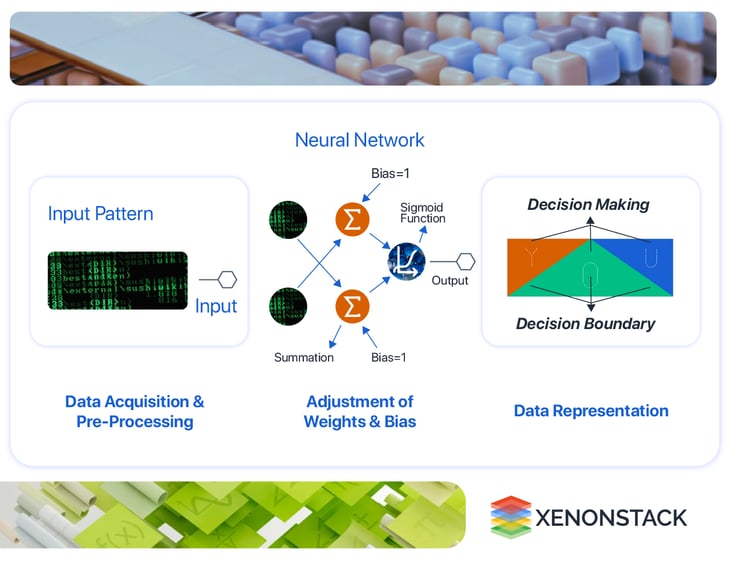
Train the Neural Network to remember the particular pattern. When the noise pattern is presented to the network, the network associates it with the memory's closest one or discards it. E.g., Hopfield Networks, which performs recognition, classification, clustering, etc.

## What are the Applications of Neural Networks?

Top applications of neural networks:

i. Neural Network for Machine Learning  
Face Recognition using it  
ii. Neuro-Fuzzy Model and its Applications   
iii. Neural Networks for data-intensive applications

### Neural Networks for Pattern Recognition

**Pattern recognition** is the study of how machines can observe the environment, learn to distinguish patterns of interest from their background and make sound and reasonable decisions about the patterns' categories. Some examples of the pattern are - fingerprint images, a handwritten word, a human face, or a speech signal. Given an input pattern, its recognition involves the following task –

**1. Supervised classification** - Given the input pattern is known as the member of a predefined class.  
**2. Unsupervised classification** - Assign pattern is to a hitherto unknown class.

So, the recognition problem here is essentially a classification or categorized task. The design of pattern recognition systems usually involves the following three aspects-

i. Data acquisition and preprocessing

ii. Data representation  
iii. Decision Making

#### Approaches For Pattern Recognition

1. Template Matching  
2. Statistical  
3. Syntactic Matching  
Following Neural Network architectures used for Pattern Recognition -

1. Multilayer Perceptron  
2. Kohonen SOM (Self-Organizing Map)  
3. Radial Basis Function Network (RBF)

A division of unsupervised learning which makes it more handful because it can also handle unsupervised learning which is itself a big plus. Click to explore about our, [Generative Adversarial Networks Applications](https://www.xenonstack.com/insights/generative-adversarial-networks)

### Neuro-Fuzzy Model and its Applications

#### What is Fuzzy logic?

Fuzzy logic refers to the logic developed to express the degree of truthiness by assigning values between 0 and 1 unlike traditional boolean logic representing 0 and 1.

#### What is Fuzzy logic role in Neural networks?

Fuzzy logic and it have one thing in common. They can be used to solve pattern recognition problems and others that do not involve any mathematical model.

#### What are the applications of Neuro-Fuzzy Model?

Systems combining both fuzzy logic and neural networks are neuro-fuzzy systems. These systems (Hybrid) can combine the advantages of both it and fuzzy logic to perform in a better way. Fuzzy logic and it have been integrated for use in the following applications -

**1.**Automotive engineering  
**2.**Applicant screening of jobs  
**3.**Control of the crane  
**4.**Monitoring of glaucoma  
In a hybrid (neuro-fuzzy) model, Neural Networks Learning Algorithms are fused with the fuzzy reasoning of fuzzy logic. It determines the values of parameters, while if-then rules are controlled by fuzzy logic.

### Neural Network for Machine Learning

i. Multilayer Perceptron (supervised classification)  
ii. Back Propagation Network (supervised classification)  
iii. Hopfield Network (for pattern association)  
iv. Deep Neural Networks (unsupervised clustering)

### Neural Networks for data-intensive applications

It has been successfully applied to the broad spectrum of data-intensive applications, such as:

|  |  |  |
| --- | --- | --- |
| **Application** | **Architecture / Algorithm** | **Activation Function** |
| Process modeling and control | Radial Basis Network | Radial Basis |
| Machine Diagnostics | Multilayer Perceptron | Tan- Sigmoid Function |
| Portfolio Management | Classification Supervised Algorithm | Tan- Sigmoid Function |
| Target Recognition | Modular Neural Network | Tan- Sigmoid Function |
| Medical Diagnosis | Multilayer Perceptron | Tan- Sigmoid Function |
| Credit Rating | Logistic Discriminant Analysis with ANN, Support Vector Machine | Logistic function |
| Targeted Marketing | Back Propagation Algorithm | Logistic function |
| Voice recognition | Multilayer Perceptron, Deep Neural Networks( Convolutional Neural Networks) | Logistic function |
| Financial Forecasting | Backpropagation Algorithm | Logistic function |
| Intelligent searching | Deep Neural Network | Logistic function |
| Fraud detection | Gradient - Descent Algorithm and Least Mean Square (LMS) algorithm. | Logistic function |

### Face Recognition Using Artificial Neural Networks

Face recognition entails comparing an image with a database of saved faces to identify the person in that input picture. It is a mechanism that involves dividing images into two parts; one containing targets (faces) and one providing the background. The associated assignment of face detection has direct relevance to the fact that images need to be analysed and faces identified earlier than they can be recognized.

A subclass of deep learning techniques specifically built to deal with graph data and make inferences from it. Click to explore about our, [Graph Neural Network on AWS](https://www.xenonstack.com/blog/graph-neural-network-on-aws)

### What is learning rule in neural network?

The learning rule is a type of mathematical logic. It encourages to gain from the present conditions and upgrade its efficiency and performance. The learning procedure of the brain modifies its neural structure. The expanding or diminishing quality of its synaptic associations relies upon their activity. Learning rules in the Neural network:

**1. Hebbian learning rule:** It determines how to customize the weights of nodes of a system.  
**2. Perceptron learning rule:** The network starts its learning by assigning a random value to each load.  
**3. Delta learning rule:** Modification in a node's sympatric weight is equal to the multiplication of the error and the input.  
**Correlation learning rule:** It is similar to supervised learning.

1. **Explain, in details, the backpropagation algorithm. What are the limitations of this algorithm?**

**ANS :-**

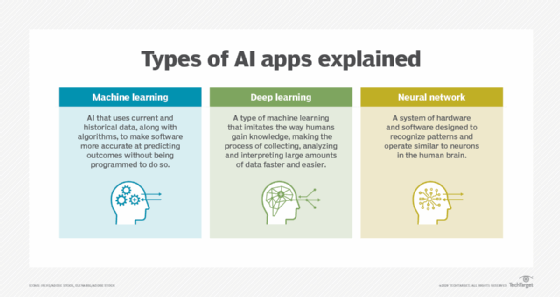
## What is a backpropagation algorithm?

Backpropagation, or backward propagation of errors, is an [algorithm](https://www.techtarget.com/whatis/definition/algorithm) that is designed to test for errors working back from output nodes to input nodes. It's an important mathematical tool for improving the accuracy of predictions in [data mining](https://www.techtarget.com/searchbusinessanalytics/definition/data-mining) and [machine learning](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML). Essentially, backpropagation is an algorithm used to quickly calculate derivatives in a [neural network](https://www.techtarget.com/searchenterpriseai/definition/neural-network), which are the changes in output because of tuning and adjustments.

There are two leading types of backpropagation networks:

* **Static backpropagation.** Static backpropagation is a network developed to map static inputs for static outputs. Static networks can solve static classification problems, such as optical character recognition ([OCR](https://www.techtarget.com/searchcontentmanagement/definition/OCR-optical-character-recognition)).
* **Recurrent backpropagation.** The recurrent backpropagation network is used for fixed-point learning. This means that during neural network training, the weights are numerical values that determine how much nodes -- also referred to as neurons -- influence output values. They're adjusted so that the network can achieve stability by reaching a fixed value.

The key difference here is that static backpropagation offers instant mapping, while recurrent backpropagation does not.

Find out how machine learning, deep learning and neural networks compare.

## What is a backpropagation algorithm in a neural network?

Artificial neural networks (ANNs) and deep neural networks use backpropagation as a learning algorithm to compute a gradient descent, which is an optimization algorithm that guides the user to the maximum or minimum of a function.

In a machine learning context, the gradient descent helps the system minimize the gap between desired outputs and achieved system outputs. The algorithm tunes the system by adjusting the weight values for various inputs to narrow the difference between outputs. This is also known as the error between the two.

More specifically, a gradient descent algorithm uses a gradual process to provide information on how a network's parameters need to be adjusted to reduce the disparity between the desired and achieved outputs. An evaluation metric called a cost function guides this process. The cost function is a mathematical function that measures this error. The algorithm's goal is to determine how the parameters must be adjusted to reduce the cost function and improve overall accuracy.

In backpropagation, this error is propagated backward from the output layer or output neuron through the hidden layers toward the input layer so that neurons can adjust themselves along the way if they played a role in producing the error. Activation functions [activate neurons to learn new complex patterns](https://www.techtarget.com/searchenterpriseai/feature/How-neural-network-training-methods-are-modeled-after-the-human-brain), information and whatever else they need to adjust their weights and biases, and mitigate this error to improve the network.

The algorithm gets its descent gradient name because the weights are updated backward, from output to input.

## What is the objective of a backpropagation algorithm?

Backpropagation algorithms are used extensively to train feedforward neural networks, such as [convolutional neural networks](https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network), in areas such as [deep learning](https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network). A backpropagation algorithm is pragmatic because it computes the gradient needed to adjust a network's weights more efficiently than computing the gradient based on each individual weight. It enables the use of gradient methods, such as gradient descent and stochastic gradient descent, to train multilayer networks and update weights to minimize errors.

It's not easy to understand exactly how changing weights and biases affect the overall behavior of an ANN. That was one factor that held back more comprehensive use of neural network applications until the early 2000s, when computers provided the necessary insight.

Today, backpropagation algorithms have practical applications in many areas of artificial intelligence, including OCR, [natural language processing](https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP) and image processing.

## Advantages and disadvantages of backpropagation algorithms

There are several advantages to using a backpropagation algorithm, but there are also challenges.

### Advantages of backpropagation algorithms

* They don't have any parameters to tune except for the number of inputs.
* They're highly adaptable and efficient, and don't require prior knowledge about the network.
* They use a standard process that usually works well.
* They're user-friendly, fast and easy to program.
* Users don't need to learn any special functions.

### Disadvantages of backpropagation algorithms

* They prefer a matrix-based approach over a mini-batch approach.
* Data mining is sensitive to [noisy data](https://www.techtarget.com/searchbusinessanalytics/definition/noisy-data) and other irregularities. Unclean data can affect the backpropagation algorithm when training a neural network used for data mining.
* Performance is highly dependent on input data.
* Training is time- and resource-intensive.

## What is a backpropagation algorithm in machine learning?

Backpropagation is a type of [supervised learning](https://www.techtarget.com/searchenterpriseai/definition/supervised-learning) since it requires a known, desired output for each input value to calculate the loss function gradient, which is how desired output values differ from actual output. Supervised learning, the most common training approach in machine learning, uses a training data set that has clearly labeled data and specified desired outputs.

Along with classifier algorithms such as naive Bayesian filters, K-nearest neighbors and support vector machines, the backpropagation training algorithm has emerged as an important part of machine learning applications that involve [predictive analytics](https://www.techtarget.com/searchbusinessanalytics/definition/predictive-analytics). While backpropagation techniques are mainly applied to neural networks, they can also be applied to both classification and regression problems in machine learning. In real-world applications, developers and machine learning experts implement backpropagation algorithms for neural networks using programming languages such as Python.

## What is the time complexity of a backpropagation algorithm?

The time complexity of each iteration -- how long it takes to execute each statement in an algorithm -- depends on the network's structure. In the early days of deep learning, a multilayer [perceptron](https://www.techtarget.com/whatis/definition/perceptron) was a basic form of neural network consisting of an input layer, hidden units and an output unit. The time complexity was low compared with today's networks, which can have exponentially more parameters. Therefore, the sheer size of a neural network is the primary factor affecting time complexity, but there are other factors, such as the [size of training data sets](https://www.techtarget.com/searchenterpriseai/feature/Using-small-data-sets-for-machine-learning-models-sees-growth) or the amount of data used to train networks.

Essentially, the number of neurons and parameters directly affects how backpropagation works. During a forward pass, in which input data moves forward from the input layer to the next layer and so on, the time complexity is larger when there are more neurons involved. During the subsequent backward pass, where parameters are adjusted to rectify an error, more parameters also mean more of a time complexity.

## What is a backpropagation momentum algorithm?

Using gradient descent optimization algorithms for tuning weights to reduce an error can be time-consuming. That's why the concept of momentum in backpropagation is used to speed up this process. It states that previous weight changes must influence the present direction of movement in weight space. Simply put, an aggregate of past weight changes is used to influence a current one.

During optimization, it's possible for gradients to change direction, which would appear to complicate the overall process. That is why this momentum technique is used to ensure optimization continues moving in the right direction and the performance of the neural network improves.

## What is a backpropagation algorithm pseudocode?

The backpropagation algorithm [pseudocode](https://www.techtarget.com/whatis/definition/pseudocode) is a basic blueprint that developers and researchers can use to conduct the backpropagation process. It's a high-level overview with plain language instructions as well as the code snippets to perform the most essential tasks in the process.

While this overview covers the essentials, the actual implementation typically is far more complex. The pseudocode covers the steps that need to get done; it typically reads like a sequential series of actions, and within it are all the core components that the backpropagation process will involve. Each pseudocode instance is pertinent to a specific context, and any [common programming language](https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/) can be used to write it, such as Python and other object-oriented programming languages.

## What is the Levenberg-Marquardt backpropagation algorithm?

The Levenberg-Marquardt algorithm is another technique that helps adjust neural network weights and biases during training. However, within the context of training neural networks, it's not an alternative or replacement for a backpropagation algorithm, but rather an optimization technique used within backpropagation-based training.

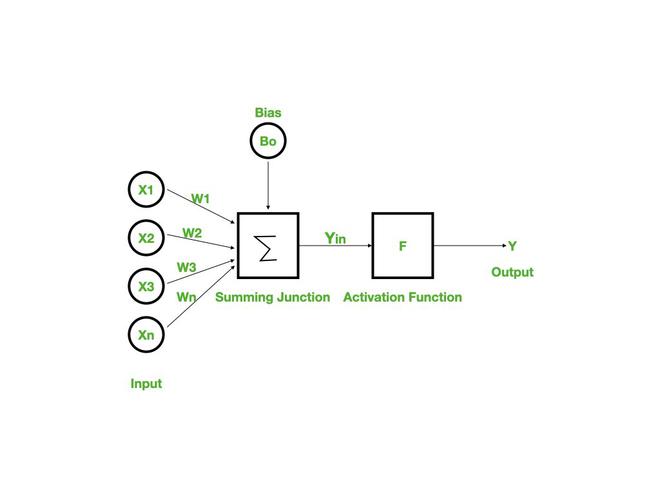
To reduce neural network errors, Levenberg-Marquardt blends gradient information from the gradient descent method with insights from what is called the Gauss-Newton algorithm -- where gradient information is represented in a curved format using mathematical matrices -- as a method of guiding updates and speeding up what would take a traditional gradient descent method a longer time to complete.

**Backpropagation**

Backpropagation is an algorithm that backpropagates the errors from the output nodes to the input nodes. Therefore, it is simply referred to as the backward propagation of errors. It uses in the vast applications of neural networks in data mining like Character recognition, Signature verification, etc.

### Neural Network:

Neural networks are an information processing paradigm inspired by the human nervous system. Just like in the human nervous system, we have biological neurons in the same way in neural networks we have artificial neurons, artificial neurons are mathematical functions derived from biological neurons. The human brain is estimated to have about 10 billion neurons, each connected to an average of 10,000 other neurons. Each neuron receives a signal through a synapse, which controls the effect of the signconcerning on the neuron.



### Backpropagation:

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights. It is very efficient, rather than naively directly computing the gradient concerning each weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

### Features of Backpropagation:

1. it is the [gradient descent](https://www.geeksforgeeks.org/gradient-descent-algorithm-and-its-variants/) method as used in the case of simple perceptron network with the differentiable unit.
2. it is different from other networks in respect to the process by which the weights are calculated during the learning period of the network.
3. training is done in the three stages :
   * the [feed-forward](https://www.geeksforgeeks.org/multilayer-feed-forward-neural-network-in-data-mining/) of input training pattern
   * the calculation and backpropagation of the error
   * updation of the weight

### ****Working of Backpropagation:****

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

### Backpropagation Algorithm:

**Step 1:** Inputs X, arrive through the preconnected path.

**Step 2:** The input is modeled using true weights W. Weights are usually chosen randomly.

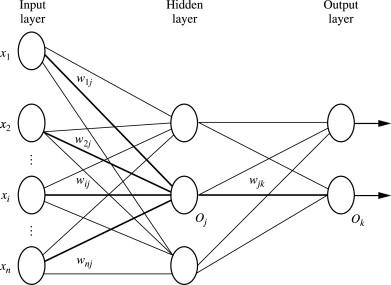
**Step 3:**Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

**Step 4:** Calculate the error in the outputs

Backpropagation Error= Actual Output – Desired Output

**Step 5:** From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

**Step 6:** Repeat the process until the desired output is achieved.



**Parameters :**

* x = inputs training vector x=(x1,x2,…………xn).
* t = target vector t=(t1,t2……………tn).
* δk= error at output unit.
* δj= error at hidden layer.
* α = learning rate.
* V0j= bias of hidden unit j.

**Training Algorithm :**

**Step 1:**Initialize weight to small random values.

**Step 2:**While the stepsstopping condition is to be false do step 3 to 10.

**Step 3:**For each training pair do step 4 to 9 (Feed-Forward).

**Step 4:**Each input unit receives the signal unit and transmitsthe signal xi signal to all the units.

**Step 5 :**Each hidden unit Zj (z=1 to a) sums its weighted input signal to calculate its net input

                     zinj = v0j + Σxivij   ( i=1 to n)

           Applying activation function zj = f(zinj) and sends this signals to all units in the layer about i.e output units

           For each output l=unit yk = (k=1 to m) sums its weighted input signals.

                     yink = w0k+ Σ ziwjk    (j=1 to a)

           and applies its activation function to calculate the output signals.

                     yk= f(yink)

**Backpropagation Error :**

**Step 6:**Each output unit yk (k=1 to n)  receives a target pattern corresponding to an input pattern then error is calculated as:

                   δk = ( tk – yk ) + yink

**Step 7:**Each hidden unit Zj (j=1 to a) sums its input from all units in the layer above

                  δinj = Σ δj wjk

              The error information term is calculated as :

                  δj = δinj+ zinj

**Updation of weight and bias :**

**Step 8:**Each output unit yk(k=1 to m) updates its bias and weight (j=1 to a). The weight correction term is given by :

                                        Δ wjk= α δkzj

and the bias correction term is given by  Δwk = α δk.

therefore    wjk(new)= wjk(old) + Δ wjk

w0k(new) = wok(old)+ Δ wok

for each hidden unit zj (j=1 to a) update its bias and weights (i=0 to n) the weight connection term

                                 Δ vij= α δj xi

and the bias connection on term

                                 Δ v0j= α δj

Therefore vij(new) = vij(old) +   Δvij

v0j(new) = v0j(old)+  Δv0j

**Step 9:**Test the stopping condition. The stopping condition can be the minimization of error, number of epochs.

### Need for Backpropagation:

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

### Types of Backpropagation

There are two types of backpropagation networks.

* **Static backpropagation:**Static backpropagation is a network designed to map static inputs for static outputs. These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).
* **Recurrent backpropagation:** Recursive backpropagation is another network used for fixed-point learning. Activation in recurrent backpropagation is feed-forward until a fixed value is reached. Static backpropagation provides an instant mapping, while recurrent backpropagation does not provide an instant mapping.

### Advantages:

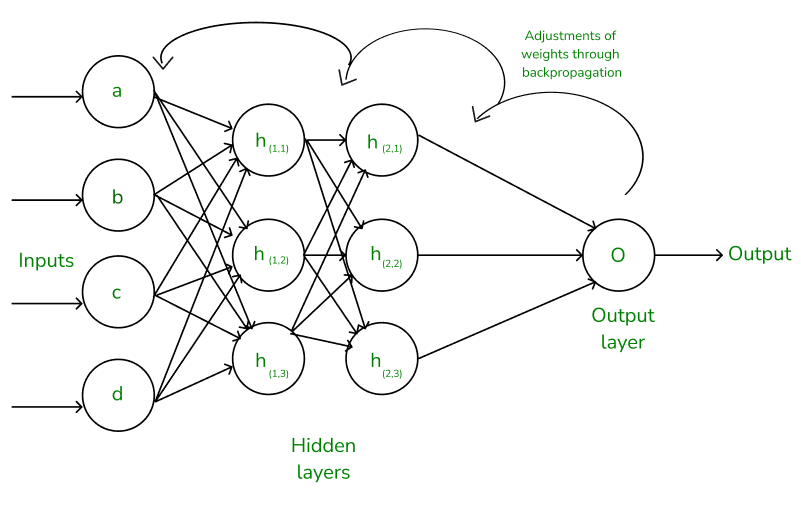
* It is simple, fast, and easy to program.
* Only numbers of the input are tuned, not any other parameter.
* It is Flexible and efficient.
* No need for users to learn any special functions.

### Disadvantages:

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

## What is backpropagation?

* In machine learning, backpropagation is an effective algorithm used to train artificial neural networks, especially in feed-forward neural networks.
* Backpropagation is an iterative algorithm, that helps to minimize the cost function by determining which weights and biases should be adjusted. During every epoch, the model learns by adapting the weights and biases to minimize the loss by moving down toward the gradient of the error. Thus, it involves the two most popular optimization algorithms, such as [gradient descent](https://www.geeksforgeeks.org/gradient-descent-algorithm-and-its-variants/) or [stochastic gradient descent](https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/).
* Computing the gradient in the backpropagation algorithm helps to minimize the [cost function](https://www.geeksforgeeks.org/what-is-cost-function/) and it can be implemented by using the mathematical rule called chain rule from calculus to navigate through complex layers of the neural network.



*fig(a) A simple illustration of how the backpropagation works by adjustments of weights*

## Advantages of Using the Backpropagation Algorithm in Neural Networks

Backpropagation, a fundamental algorithm in training neural networks, offers several advantages that make it a preferred choice for many machine learning tasks. Here, we discuss some key advantages of using the backpropagation algorithm:

1. **Ease of Implementation:** Backpropagation does not require prior knowledge of neural networks, making it accessible to beginners. Its straightforward nature simplifies the programming process, as it primarily involves adjusting weights based on error derivatives.
2. **Simplicity and Flexibility:** The algorithm’s simplicity allows it to be applied to a wide range of problems and network architectures. Its flexibility makes it suitable for various scenarios, from simple feedforward networks to complex recurrent or convolutional neural networks.
3. **Efficiency:** Backpropagation accelerates the learning process by directly updating weights based on the calculated error derivatives. This efficiency is particularly advantageous in training deep neural networks, where learning features of a function can be time-consuming.
4. **Generalization:** Backpropagation enables neural networks to generalize well to unseen data by iteratively adjusting weights during training. This generalization ability is crucial for developing models that can make accurate predictions on new, unseen examples.
5. **Scalability:** Backpropagation scales well with the size of the dataset and the complexity of the network. This scalability makes it suitable for large-scale machine learning tasks, where training data and network size are significant factors.

In conclusion, the backpropagation algorithm offers several advantages that contribute to its widespread use in training neural networks. Its ease of implementation, simplicity, efficiency, generalization ability, and scalability make it a valuable tool for developing and training neural network models for various machine learning applications.

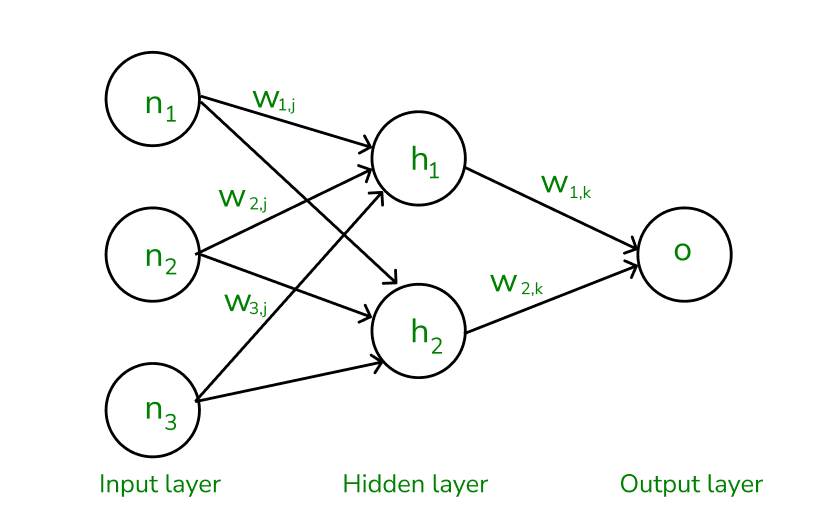
## Working of Backpropagation Algorithm

The Backpropagation algorithm works by two different passes, they are:

* Forward pass
* Backward pass

### How does Forward pass work?

* In forward pass, initially the input is fed into the input layer. Since the inputs are raw data, they can be used for training our neural network.
* The inputs and their corresponding weights are passed to the hidden layer. The hidden layer performs the computation on the data it receives. If there are two hidden layers in the neural network, for instance, consider the illustration fig(a), h1 and h2 are the two hidden layers, and the output of h1 can be used as an input of h2. Before applying it to the activation function, the bias is added.
* To the weighted sum of inputs, the activation function is applied in the hidden layer to each of its neurons. One such activation function that is commonly used is ReLU can also be used, which is responsible for returning the input if it is positive otherwise it returns zero. By doing this so, it introduces the non-linearity to our model, which enables the network to learn the complex relationships in the data. And finally, the weighted outputs from the last hidden layer are fed into the output to compute the final prediction, this layer can also use the activation function called the softmax function which is responsible for converting the weighted outputs into probabilities for each class.



*The forward pass using weights and biases*

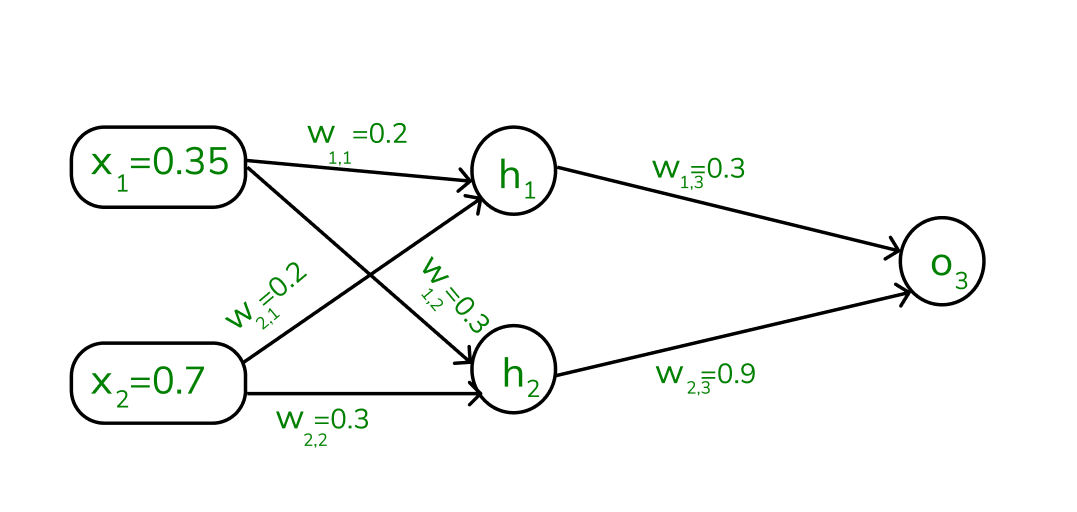
### How does backward pass work?

* In the backward pass process shows, the error is transmitted back to the network which helps the network, to improve its performance by learning and adjusting the internal weights.
* To find the error generated through the process of forward pass, we can use one of the most commonly used methods called mean squared error which calculates the difference between the predicted output and desired output. The formula for mean squared error is: 𝑀𝑒𝑎𝑛𝑠𝑞𝑢𝑎𝑟𝑒𝑑𝑒𝑟𝑟𝑜𝑟=(𝑝𝑟𝑒𝑑𝑖𝑐𝑡𝑒𝑑𝑜𝑢𝑡𝑝𝑢𝑡–𝑎𝑐𝑡𝑢𝑎𝑙𝑜𝑢𝑡𝑝𝑢𝑡)2*Meansquarederror*=(*predictedoutput*–*actualoutput*)2
* Once we have done the calculation at the output layer, we then propagate the error backward through the network, layer by layer.
* The key calculation during the backward pass is determining the gradients for each weight and bias in the network. This gradient is responsible for telling us how much each weight/bias should be adjusted to minimize the error in the next forward pass. The chain rule is used iteratively to calculate this gradient efficiently.
* In addition to gradient calculation, the activation function also plays a crucial role in backpropagation, it works by calculating the gradients with the help of the derivative of the activation function.

## Example of Backpropagation in Machine Learning

Let us now take an example to explain backpropagation in Machine Learning,

**Assume that the neurons have the sigmoid activation function to perform forward and backward pass on the network. And also assume that the actual output of y is 0.5 and the learning rate is 1. Now perform the backpropagation using backpropagation algorithm.**



*Example (1) of backpropagation sum*

### Implementing forward propagation:

**Step1:** Before proceeding to calculating forward propagation, we need to know the two formulae:

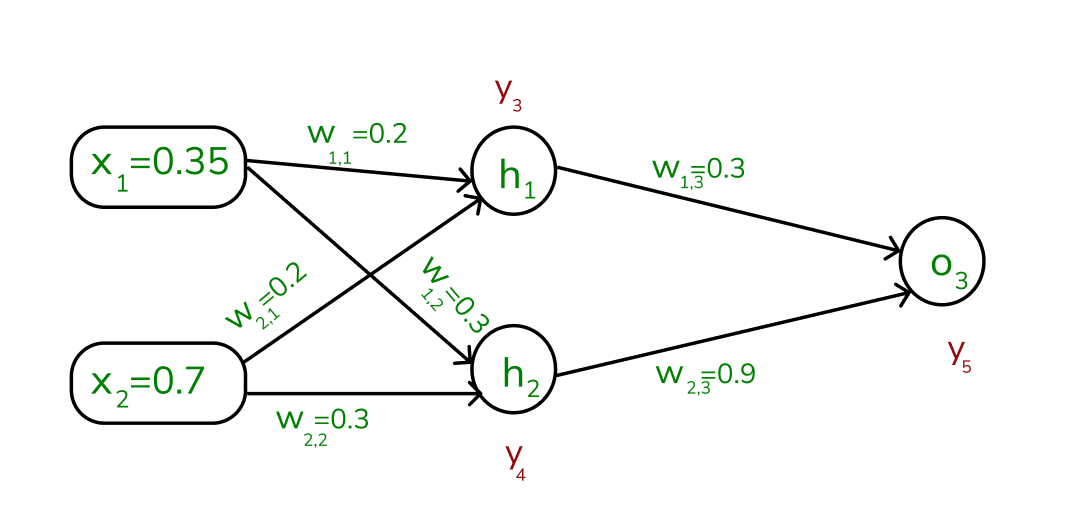
𝑎𝑗=∑(𝑤𝑖,𝑗∗𝑥𝑖)*aj*​=∑(*wi*​,*j*∗*xi*​)

Where,

* aj is the weighted sum of all the inputs and weights at each node,
* wi,j – represents the weights associated with the jth input to the ith neuron,
* xi – represents the value of the jth input,

𝑦𝑗=(𝑎𝑗)=11+𝑒−𝑎𝑗*yj*​=*F*(*aj*​)=1+*e*−*aj*1​, yi – is the output value, F denotes the activation function [sigmoid activation function is used here), which transforms the weighted sum into the output value.

**Step 2: To compute the forward pass, we need to compute the output for y3 , y4 , and y5.**



*To find the outputs of y3, y4 and y5*

We start by calculating the weights and inputs by using the formula:

𝑎𝑗=∑(𝑤𝑖,𝑗∗𝑥𝑖)*aj*​=∑(*wi*,*j*​∗*xi*​) To find y3 , we need to consider its incoming edges along with its weight and input. Here the incoming edges are from X1 and X2.

#### At h1 node,

𝑎1=(𝑤1,1𝑥1)+(𝑤2,1𝑥2)=(0.2∗0.35)+(0.3∗0.7)=0.28*a*1​​=(*w*1,1​*x*1​)+(*w*2,1​*x*2​)=(0.2∗0.35)+(0.3∗0.7)=0.28​

Once, we calculated the a1 value, we can now proceed to find the y3 value:

𝑦𝑗=(𝑎𝑗)=11+𝑒−𝑎𝑗*yj*​=*F*(*aj*​)=1+*e*−*aj*1​

𝑦3=(0.28)=11+𝑒−0.28*y*3​=*F*(0.28)=1+*e*−0.281​

𝑦3=0.57*y*3​=0.57

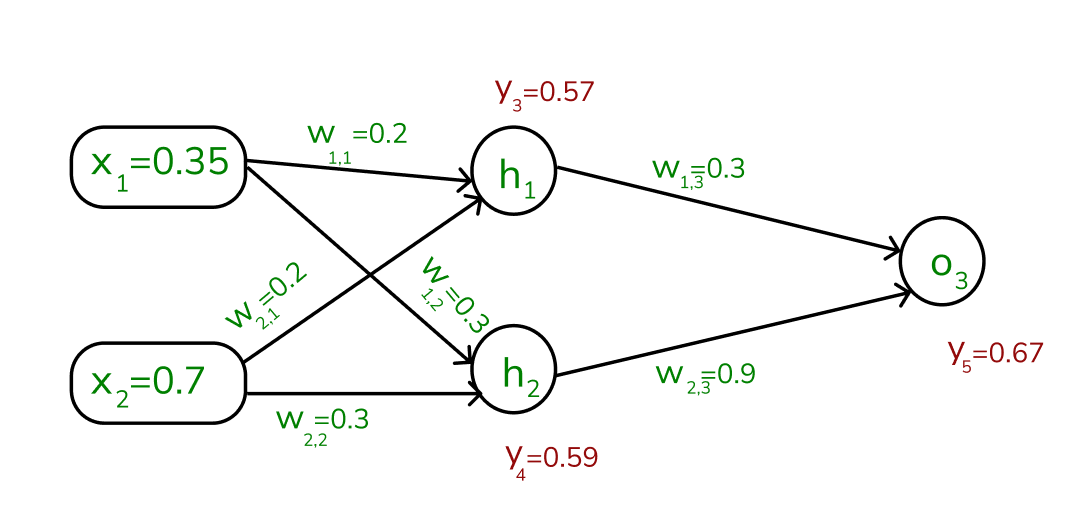
Similarly find the values of y4 at **h2**and y5 at O3 ,

𝑎2=(𝑤1,2∗𝑥1)+(𝑤2,2∗𝑥2)=(0.3∗0.35)+(0.3∗0.7)=0.315*a*2=(*w*1,2​∗*x*1​)+(*w*2,2​∗*x*2​)=(0.3∗0.35)+(0.3∗0.7)=0.315

𝑦4=(0.315)=11+𝑒−0.315*y*4​=*F*(0.315)=1+*e*−0.3151​

𝑎3=(𝑤1,3∗𝑦3)+(𝑤2,3∗𝑦4)=(0.3∗0.57)+(0.9∗0.59)=0.702*a*3=(*w*1,3​∗*y*3​)+(*w*2,3​∗*y*4​)=(0.3∗0.57)+(0.9∗0.59)=0.702

𝑦5=(0.702)=11+𝑒−0.702=0.67*y*5​=*F*(0.702)=1+*e*−0.7021​=0.67



*Values of y3, y4 and y5*

Note that, our actual output is 0.5 but we obtained 0.67. To calculate the error, we can use the below formula:

𝐸𝑟𝑟𝑜𝑟𝑗=𝑦𝑡𝑎𝑟𝑔𝑒𝑡–𝑦5*Errorj*​=*ytarget*​–*y*5​

Error = 0.5 – 0.67

= -0.17

Using this error value, we will be backpropagating.

### ****Implementing Backward Propagation****

Each weight in the network is changed by,

∇wij = η 𝛿j Oj

𝛿j = Oj (1-Oj)(tj - Oj) (if j is an output unit)

𝛿j = Oj (1-O)∑k 𝛿k wkj (if j is a hidden unit)

where ,

η is the constant which is considered as learning rate,

tj is the correct output for unit j

𝛿j is the error measure for unit j

**Step 3: To calculate the backpropagation, we need to start from the output unit:**

To compute the 𝛿5, we need to use the output of forward pass,

𝛿5 = y5(1-y5) (ytarget -y5)

= 0.67(1-0.67) (-0.17)

= -0.0376

**For hidden unit,**

To compute the hidden unit, we will take the value of 𝛿5

𝛿3 = y3(1-y3) (w1,3 \* 𝛿5)

=0.57(1-0.57) (0.3\*-0.0375)

=-0.0027

𝛿4 = y4 (1-y5) (w2,3 \* 𝛿5)

=0.59(1-0.59) (0.9\*-0.376)

=-0.0819

**Step 4: We need to update the weights, from output unit to hidden unit,**

∇ wj,i = η 𝛿j Oj

Note- Here our learning rate is 1

∇ w2,3 = η 𝛿5 O4

= 1 \* (-0.376) \* 0.59

= -0.22184

We will be updating the weights based on the old weight of the network,

w2,3(new) = ∇ w4,5 + w4,5 (old)

= -0.22184 + 0.9

= 0.67816

From hidden unit to input unit,

For an hidden to input node, we need to do calculations by the following;

∇ w1,1 = η 𝛿3 O4

= 1 \* (-0.0027) \* 0.35

= 0.000945

Similarly, we need to calculate the new weight value using the old one:

w1,1(new) = ∇ w1,1+ w1,1 (old)

= 0.000945 + 0.2

= 0.200945

**Similarly, we update the weights of the other neurons: The new weights are mentioned below**

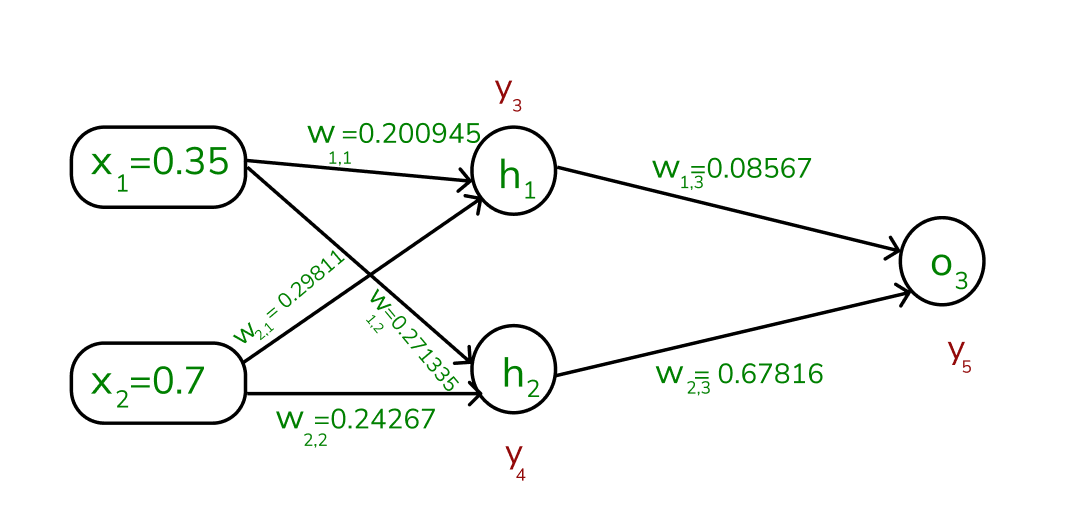
w1,2 (new) = 0.271335

w1,3 (new) = 0.08567

w2,1 (new) = 0.29811

w2,2 (new) = 0.24267

The updated weights are illustrated below,



*Through backward pass the weights are updated*

Once, the above process is done, we again perform the forward pass to find if we obtain the actual output as 0.5.

While performing the forward pass again, we obtain the following values:

y3 = 0.57

y4 = 0.56

y5 = 0.61

We can clearly see that our y5 value is 0.61 which is not an expected actual output, So again we need to find the error and backpropagate through the network by updating the weights until the actual output is obtained.

𝐸𝑟𝑟𝑜𝑟=𝑦𝑡𝑎𝑟𝑔𝑒𝑡–𝑦5*Error*=*ytarget*​–*y*5​

= 0.5 – 0.61

= -0.11

This is how the backpropagate works, it will be performing the forward pass first to see if we obtain the actual output, if not we will be finding the error rate and then backpropagating backwards through the layers in the network by adjusting the weights according to the error rate. This process is said to be continued until the actual output is gained by the neural network.

## Python program for backpropagation

Here’s a simple implementation of feedforward neural network with backpropagation in Python:

1. **Neural Network Initialization**: The NeuralNetwork class is initialized with parameters for the input size, hidden layer size, and output size. It also initializes the weights and biases with random values.
2. **Sigmoid Activation Function**: The sigmoid method implements the sigmoid activation function, which squashes the input to a value between 0 and 1.
3. **Sigmoid Derivative**: The sigmoid\_derivative method calculates the derivative of the sigmoid function. It computes the gradients of the loss function with respect to weights.
4. **Feedforward Pass**: The feedforward method calculates the activations of the hidden and output layers based on the input data and current weights and biases. It uses matrix multiplication to propagate the inputs through the network.
5. **Backpropagation**: The backward method performs the backpropagation algorithm. It calculates the error at the output layer and propagates it back through the network to update the weights and biases using gradient descent.
6. **Training the Neural Network**: The train method trains the neural network using the specified number of epochs and learning rate. It iterates through the training data, performs the feedforward and backward passes, and updates the weights and biases accordingly.
7. **XOR Dataset**: The XOR dataset (X) is defined, which contains input pairs that represent the XOR operation, where the output is 1 if exactly one of the inputs is 1, and 0 otherwise.
8. **Testing the Trained Model**: After training, the neural network is tested on the XOR dataset (X) to see how well it has learned the XOR function. The predicted outputs are printed to the console, showing the neural network’s predictions for each input pair.

|  |
| --- |
|  |

1. **Describe, in details, the process of adjusting the interconnection weights in a multi-layer neural network.**

**Ans:-**

In order to learn about Backpropagation**,**we first have to understand the architecture of the neural network and then the learning process in ANN. So, let’s start about knowing the various architectures of the ANN:

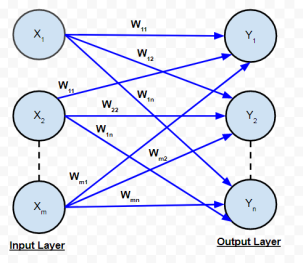
#### ****Architectures of Neural Network:****

ANN is a computational system consisting of many interconnected units called **artificial neurons**. The connection between artificial neurons can transmit a signal from one neuron to another. So, there are multiple possibilities for connecting the neurons based on which the **architecture** we are going to adopt for a specific solution. Some permutations and combinations are as follows:

* There may be just two layers of neuron in the network – the input and output layer.
* There can be one or more intermediate **‘hidden’** layers of a neuron.
* The neurons may be connected with all neurons in the next layer and so on …..

So let’s start talking about the various possible architectures:

**A. Single-layer Feed Forward Network:**

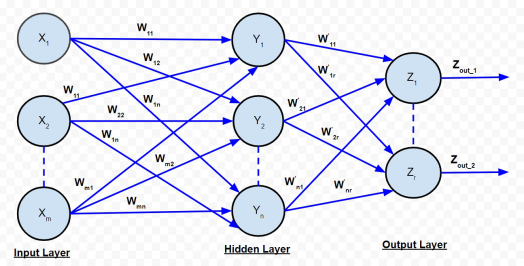


It is the simplest and most basic architecture of ANN’s. It consists of only two layers- the input layer and the output layer. **The input layer** consists of ‘m’ input neurons connected to each of the ‘n’ output neurons. The connections carry weights w11 and so on. The input layer of the neurons doesn’t conduct any processing – they pass the i/p signals to the o/p neurons. The computations are performed in the output layer. So, though it has 2 layers of neurons, only one layer is performing the computation. This is the reason why **the network is known as SINGLE** layer. Also, the signals always flow from the input layer to the output layer. Hence, the **network is known as FEED FORWARD.**

The net signal input to the output neurons is given by:

The signal output from each output neuron will depend on the activation function used.

**B. Multi-layer Feed Forward Network:**



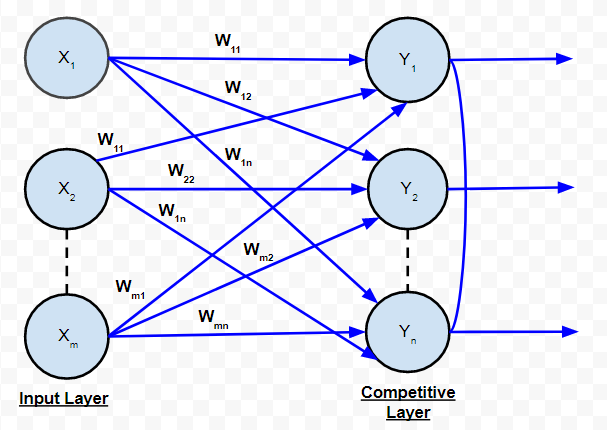
***Multi-Layer Feed Forward Network***

The multi-layer feed-forward network is quite similar to the single-layer feed-forward network, except for the fact that there are one or more intermediate layers of neurons between the input and output layer. Hence, the **network is termed as multi-layer.**Each of the layers may have a varying number of neurons. For example, the one shown in the above diagram has ‘m’ neurons in the input layer and ‘r’ neurons in the output layer and there is only one hidden layer with ‘n’ neurons.

for the kth hidden layer neuron. The net signal input to the neuron in the output layer is given by:

**C. Competitive Network:**

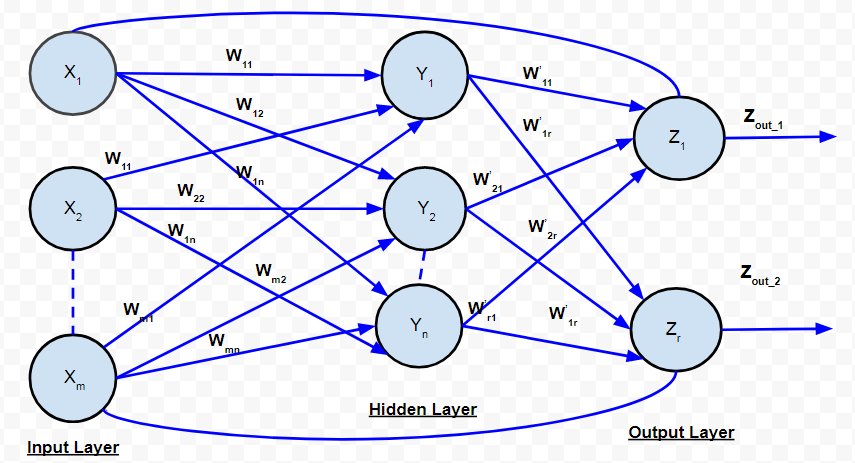
It is as same as the single-layer feed-forward network in structure. The only difference is that**the output neurons are connected with each other (either partially or fully)**. Below is the diagram for this type of network.



***Competitive Network***

According to the diagram, it is clear that few of the output neurons are interconnected to each other. For a given input, the output neurons compete against themselves to represent the input. It represents a form of an unsupervised learning algorithm in ANN that is suitable to find the clusters in a data set.

**D. Recurrent Network:**



***Recurrent Network***

In feed-forward networks, the signal always flows from the input layer towards the output layer (in one direction only). In the case of recurrent neural networks, there is **a feedback loop** (from the neurons in the output layer to the input layer neurons). There can be self-loops too.

#### ****Learning Process In ANN:****

Learning process in ANN mainly depends on four factors, they are:

1. **The number of layers in the network (Single-layered or multi-layered)**
2. **Direction of signal flow (Feedforward or recurrent)**
3. **Number of nodes in layers:**The number of node in the input layer is equal to the number of features of the input data set. The number of output nodes will depend on possible outcomes i.e. the number of classes in case of supervised learning. But the number of layers in the hidden layer is to be chosen by the user. A larger number of nodes in the hidden layer, higher the performance but too many nodes may result in overfitting as well as increased computational expense.
4. **Weight of Interconnected Nodes:**Deciding the value of weights attached with each interconnection between each neuron so that a specific learning problem can be solved correctly is quite a difficult problem by itself. Take an example to understand the problem. Take the example of a**Multi-layered Feed-Forward Network,**we have to train an ANN model using some data, so that it can classify a new data set, sayp\_5(3,-2). Say we have deduced that p\_1=(5,2)   and  p\_2 = (-1,12)   belonging to class C1 while p\_3=(3,-5)   and p\_4 = (-2,-1)  belonging to class C2. We assume the values of synaptic weights w\_0,w\_1,w\_2 as -2, 1/2 and 1/4 respectively. But we will NOT get these weight values for every learning problem. For solving a learning problem with ANN, we can start with a set of values for synaptic weights and keep changing those in multiple iterations. The stopping criterion may be the **rate of misclassification < 1% or the maximum numbers of iterations should be less than 25(a threshold value).** There may be another problem that, the rate of misclassification may not reduce progressively.

So, we can summarize the learning process in ANN as the combination of – **deciding the number of hidden layers,**the **number of nodes in each of the hidden layers,**the **direction of signal flow, deciding the connection weight.**

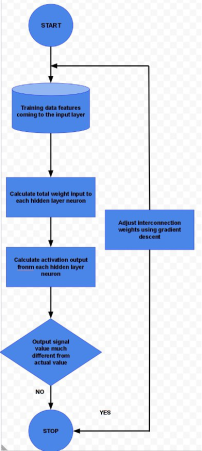
**Multi-layer feed network**is a commonly used architecture. It has been observed that a neural network with even one hidden layer can be used to reasonably approximate any continuous function. The learning methodology adopted to train a multi-layer feed-forward network is **Backpropagation**.

#### ****Backpropagation:****

In the above section, we get to know that the most critical activities of training an ANN are to assign the inter-neuron connection weights. In 1986, an efficient way of training an ANN was introduced. In this method, the **difference in output values of the output layer and the expected values, are**propagated**back from the output layer to the**preceding**layers.** Hence, the algorithm implementing this method is known as BACK PROPAGATION**i.e. propagating the errors back to the**preceding**layers.**

The backpropagation algorithm is applicable for multi-layer feed-forward network. It is a supervised learning algorithm which continues adjusting the weights of the connected neurons with an objective to reduce the deviation of the output signal from the target output. This algorithm consists of multiple iterations, **known as epochs.** Each epoch consists of two phases:

* **Forward Phase:**Signal flow from neurons in the input layer to the neurons in the output layer through the hidden layers. The weights of the interconnections and activation functions are used during the flow. In the output layer, the output signals are generated.
* **Backward Phase:**Signal is compared with the expected value. The computed errors are propagated backwards from the output to the preceding layer. The error propagated back are used to adjust the interconnection weights between the layers.

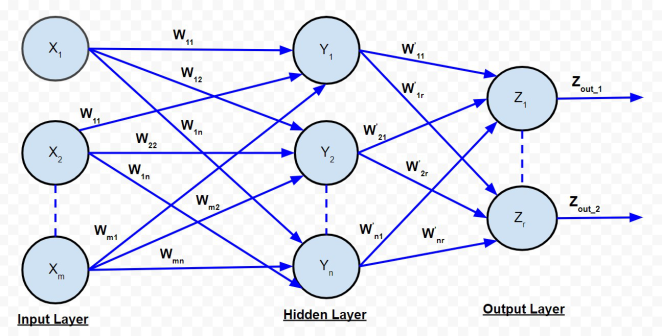


***BACKPROPAGATION***

The above diagram depicts a reasonably simplified version of the back propagation algorithm.

One main part of the algorithm is adjusting the interconnection weights. This is done using a technique termed as **Gradient Descent**. In simple words, the algorithm calculates the partial derivative of the activation function by each interconnection weight to identify the ‘gradient’ or extent of change of the weight required to minimize the cost function.

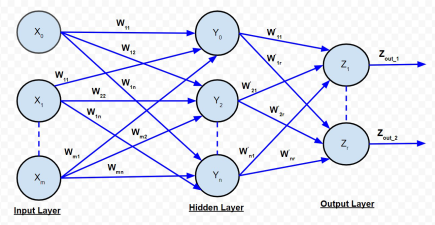
In order to understand the back propagation algorithm in detail, let us consider the **Multi-layer Feed Forward Network.**



**The net signal input to the hidden layer neurons is given by:**

If  is the activation function of the hidden layer, then

**The net signal input to the output layer neurons is given by:**



***BACKPROPAGATION NET***

Note that the signals  and  are assumed to be 1. If  is the activation function of the hidden layer, then

**If is the target of the k-th output neuron, then the cost function defined as the squared error of the output layer is given by:**

**According to the descent algorithm**, partial derivative of cost function E has to be taken with respect to interconnection weights. Mathematically it can be represented as:

{Above expression is for the interconnection weights between the **j-th neuron in the hidden layer and the k-th neuron in the output layer**.} **This expression can be reduced to**

**where,  or**

**If we assume** as a component of the weight adjustment needed for weight   corresponding to the k-th output neuron, then :

On the basis of this, the weights and bias need to be updated as follows:

* **For weights:**
* **Hence,**
* **For bias:**
* **Hence,**

In the above expressions, alpha is the learning rate of the neural network.Learning rate is a user parameter which decreases or increases the speed with which the interconnection weights of a neural network is to be adjusted. If the learning rate is too high, the adjustment done as a part of the gradient descent process may diverge the data set rather than converging it. On the other hand, if the learning rate is too low, the optimization may consume more time because of the small steps towards the minima.

{All the above calculations are for the interconnection weight between neurons in the hidden layer and neurons in the output layer}

Like the above expressions, we can deduce the expressions for “Interconnection weights between the input and hidden layers:

* **For weights:**
* **Hence,**
* **For bias:**
* **Hence,**

So, in this way, we can use the Backpropagation algorithm to solve various Artificial Neural Networks.

1. **What are the steps in the backpropagation algorithm? Why a multi-layer neural network is required?**

**ANS:-**

Backpropagation is a widely used algorithm for training neural networks. It is a supervised learning method that involves adjusting the weights of the connections between neurons in order to minimize the difference between the predicted output and the actual output. The backpropagation algorithm is essentially a way of calculating the gradient of the loss function with respect to the weights in the network. The steps of the algorithm are as follows:

1. **Initialization**: The first step is to initialize the weights in the network to small random values.
2. **Forward Pas**s: The next step is to perform a forward pass through the network. This involves applying the input to the first layer of neurons, which in turn produces an output that is passed on to the next layer, and so on until the output layer is reached. At each layer, the inputs are multiplied by the weights and then passed through an activation function.
3. **Calculate the Erro**r: Once the output of the network is obtained, we calculate the difference between the predicted output and the actual output. This difference is called the error or loss.
4. **Backward Pass**: In this step, we propagate the error backwards through the network to calculate the gradient of the loss function with respect to the weights. This involves computing the partial derivative of the loss with respect to each weight in the network, using the chain rule of calculus.
5. **Update the Weights**: Once the gradients have been calculated, we use them to update the weights in the network. This is done by subtracting a fraction of the gradient from the weight, which is called the learning rate. The learning rate determines the step size that is taken in the direction of the negative gradient.
6. **Repea**t: Steps 2-5 are repeated for each input in the training set until the weights have converged to a minimum value of the loss function.

The backpropagation algorithm is an iterative process that adjusts the weights in the network in order to minimize the difference between the predicted output and the actual output. The key idea behind backpropagation is to use the error signal to adjust the weights in a way that reduces the error. By repeatedly adjusting the weights in this way, the network gradually learns to produce more accurate predictions.

The backpropagation algorithm is a widely used method for training artificial neural networks. It is based on the gradient descent optimization algorithm and aims to minimize the error between the predicted output and the actual output of the network. The steps of the backpropagation algorithm are as follows:

1. Forward propagation: In this step, the input data is fed forward through the neural network, layer by layer, to produce the output. Each layer applies a set of weights to the inputs and applies an activation function to produce the output of that layer.
2. Calculate the error: Once the output is generated, the error between the predicted output and the actual output is calculated using a loss function. The most common loss function used in neural networks is the mean squared error.
3. Backpropagation: In this step, the error is propagated backward through the network to adjust the weights. The error at each output unit is computed as the difference between the predicted output and the actual output. The error is then propagated backward through the layers of the network using the chain rule of calculus.
4. Compute the gradient: The gradient of the error with respect to each weight is computed. This is done by multiplying the error at each unit by the derivative of the activation function at that unit.
5. Update the weights: The weights are then updated using the computed gradients and the learning rate. The learning rate determines how quickly the weights are updated and is typically set to a small value to avoid overshooting the minimum.
6. Repeat steps 1-5: The above steps are repeated for each input in the training data set, and the weights are adjusted accordingly. This process is repeated for a fixed number of iterations or until the error reaches a satisfactory level.

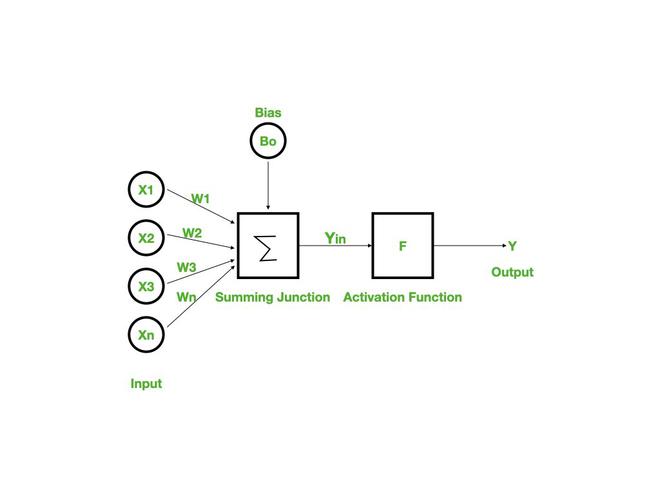
In summary, the backpropagation algorithm involves propagating the input forward through the network to produce the output, calculating the error between the predicted and actual output, propagating the error backward through the network to adjust the weights, and repeating the process until the error is minimized.

**Backpropagation**

Backpropagation is an algorithm that backpropagates the errors from the output nodes to the input nodes. Therefore, it is simply referred to as the backward propagation of errors. It uses in the vast applications of neural networks in data mining like Character recognition, Signature verification, etc.

### Neural Network:

Neural networks are an information processing paradigm inspired by the human nervous system. Just like in the human nervous system, we have biological neurons in the same way in neural networks we have artificial neurons, artificial neurons are mathematical functions derived from biological neurons. The human brain is estimated to have about 10 billion neurons, each connected to an average of 10,000 other neurons. Each neuron receives a signal through a synapse, which controls the effect of the signconcerning on the neuron.



### Backpropagation:

Backpropagation is a widely used algorithm for training feedforward neural networks. It computes the gradient of the loss function with respect to the network weights. It is very efficient, rather than naively directly computing the gradient concerning each weight. This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.

The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

### Features of Backpropagation:

1. it is the [gradient descent](https://www.geeksforgeeks.org/gradient-descent-algorithm-and-its-variants/) method as used in the case of simple perceptron network with the differentiable unit.
2. it is different from other networks in respect to the process by which the weights are calculated during the learning period of the network.
3. training is done in the three stages :
   * the [feed-forward](https://www.geeksforgeeks.org/multilayer-feed-forward-neural-network-in-data-mining/) of input training pattern
   * the calculation and backpropagation of the error
   * updation of the weight

### ****Working of Backpropagation:****

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

### Backpropagation Algorithm:

**Step 1:** Inputs X, arrive through the preconnected path.

**Step 2:** The input is modeled using true weights W. Weights are usually chosen randomly.

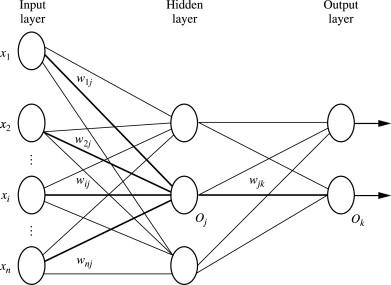
**Step 3:**Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

**Step 4:** Calculate the error in the outputs

Backpropagation Error= Actual Output – Desired Output

**Step 5:** From the output layer, go back to the hidden layer to adjust the weights to reduce the error.

**Step 6:** Repeat the process until the desired output is achieved.



**Parameters :**

* x = inputs training vector x=(x1,x2,…………xn).
* t = target vector t=(t1,t2……………tn).
* δk= error at output unit.
* δj= error at hidden layer.
* α = learning rate.
* V0j= bias of hidden unit j.

**Training Algorithm :**

**Step 1:**Initialize weight to small random values.

**Step 2:**While the stepsstopping condition is to be false do step 3 to 10.

**Step 3:**For each training pair do step 4 to 9 (Feed-Forward).

**Step 4:**Each input unit receives the signal unit and transmitsthe signal xi signal to all the units.

**Step 5 :**Each hidden unit Zj (z=1 to a) sums its weighted input signal to calculate its net input

                     zinj = v0j + Σxivij   ( i=1 to n)

           Applying activation function zj = f(zinj) and sends this signals to all units in the layer about i.e output units

           For each output l=unit yk = (k=1 to m) sums its weighted input signals.

                     yink = w0k+ Σ ziwjk    (j=1 to a)

           and applies its activation function to calculate the output signals.

                     yk= f(yink)

**Backpropagation Error :**

**Step 6:**Each output unit yk (k=1 to n)  receives a target pattern corresponding to an input pattern then error is calculated as:

                   δk = ( tk – yk ) + yink

**Step 7:**Each hidden unit Zj (j=1 to a) sums its input from all units in the layer above

                  δinj = Σ δj wjk

              The error information term is calculated as :

                  δj = δinj+ zinj

**Updation of weight and bias :**

**Step 8:**Each output unit yk(k=1 to m) updates its bias and weight (j=1 to a). The weight correction term is given by :

                                        Δ wjk= α δkzj

and the bias correction term is given by  Δwk = α δk.

therefore    wjk(new)= wjk(old) + Δ wjk

w0k(new) = wok(old)+ Δ wok

for each hidden unit zj (j=1 to a) update its bias and weights (i=0 to n) the weight connection term

                                 Δ vij= α δj xi

and the bias connection on term

                                 Δ v0j= α δj

Therefore vij(new) = vij(old) +   Δvij

v0j(new) = v0j(old)+  Δv0j

**Step 9:**Test the stopping condition. The stopping condition can be the minimization of error, number of epochs.

### Need for Backpropagation:

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

### Types of Backpropagation

There are two types of backpropagation networks.

* **Static backpropagation:**Static backpropagation is a network designed to map static inputs for static outputs. These types of networks are capable of solving static classification problems such as OCR (Optical Character Recognition).
* **Recurrent backpropagation:** Recursive backpropagation is another network used for fixed-point learning. Activation in recurrent backpropagation is feed-forward until a fixed value is reached. Static backpropagation provides an instant mapping, while recurrent backpropagation does not provide an instant mapping.

### Advantages:

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

### Disadvantages:

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

**Multi-layer feed forward networks and learning**

The backpropagation algorithm is quite useful in solving many real world problems including finding out right set of motivational strategies for employees1, shortlisting resumes from a given bulk, fingerprint recognition, voice recognition, face detection and so on. The back propagation algorithm is applied over multilayer, feed forward and complete neural networks usually. Most problems are designed and solved using those networks. We will look at how these layers are organized and back propagation algorithm is applied in this module. Back propagation algorithm is basically a classification learning algorithm. When the user knows how things are classified without applying any formal method, back propagation is an attractive option. For example we can classify faces as who is who but we do not know the formal method for the same. We can classify signatures but again we do not know how our mind does that. These are excellent examples of where can one use back propagation algorithm.

Many variants of the basic algorithm are used in practice with different types of networks butwe will confine our discussionto multilayer networks and the conventional backpropagation algorithm in this module. This discussion will be sufficient to initiate the learner in the process of using back propagation algorithm. Most real world problems can be solved with this combination. One can modify the network and algorithm for a typical case. The reader can proceed further using that knowledge.

The network used here is called multi-layer as it contains more than one layers. It is feed forward as the input activations are flowing in forward direction. Though activations are propagated forward, the errors are reported (flown) back to adjust weights. That is why the algorithm is called backpropagation;it propagates errors back. The network is also called complete as every input unit is connected to every hidden unit and every hidden unit in turn is connected to every output unit. In the sample figure 9.1 we only have 3-3-2 architecture (3 input, 3 hidden and 2 output) but the number of all the layers and number of neurons in each layer depends on what we are planning to learn.

**One student of the author of this module has done his Ph D on this topic**

A multi-layer feed forward network is shown in figure 9.1. The first layer is known as input layer and its job is to accept and distribute inputs in a way that every neuron in the hidden layer receives a copy of each of the input. Thus every unit of the hidden layer receives the information from all input neurons. The same thing is also true for hidden and output layer communication. The hidden layer comes next (though most networks have one hidden layer, some typical cases might have multiple hidden layers). The job of hidden layer to get features out of inputs. We will soon elaborate that. The layer is called ‘hidden’ as it is not seen from either side. The inputs interact with input layer while the output layer generates the output but hidden layer does not interact either with input or output and hence the name. The output layers job is to recognize the output. Let us take one example to understand how number of neurons at each layer are counted.

Assume a character recognition program is running. The input is a character which is in form of a 9\*9 matrix as shown in figures 9.2, 9.3 and 9.4. This is a very crude representation but good for our discussion. One can take an image instead with each pixel as one unit and taking may be 500 \* 500 pixels for making it far better. Even in that case our discussion does not change much.

The input layer accepts 9\*9 matrix with values 0 for a blank square while 1 for filled square. Thus the input to the network for figure 9.2 would be following. (For better viewing, the ones are boldfaced)

0000**1**0000

000**1**0**1**000

000**1**0**1**000

00**1**000**1**00

00**11111**00

0**1**00000**1**0

0**1**00000**1**0

Thus total 81 (9\*9) binary input units are needed at input layer; one for each binary value in the input. Some of them are 1 while all others are zero. How many output units are needed? Suppose if we wanted to recognize all uppercase letters only. That makes it 26. If we want digits also, it makes it 36. We also want upper and lowercase with digits, the total different characters we would like to recognize mounts to 62 (26+26+10). For first case we need a 0..32 as output while the last two require 0..64 as output. For having 0..32 output we need 5 binary neurons while for 0..64 we need 6 binary neurons2.

What would be the number of hidden units? Researchers found that the geometric mean of number of input units and output units is a good measure. That means number of hidden units = **√** **∗** where nI is number of input units and nO is number of output units. In our case the number of hidden units = **√81 ∗ 6**=**√486**= 22 (after rounding off). One may have a query, why we have to have hidden units? What if they are not present? In fact the first generation of neural networks haven’t had hidden layers, they have only input and output layers. They were popularly known as ***single layer perceptrons***. The single layer perceptron could solve quite a large number of problems. It has an excellent, fool-proof method known to make it learn anything it is capable of. Unfortunately the single layer perceptrons were found to be incapable of solving a typical class of problems called non-linearly separable problems which includes simple problems likeXOR. Including hidden layer removes that hindrance and makes them capable to learn any problem that they can be trained for; including non-linearly separable problems. Researcher had also found that a network with one hidden unit can learn whatever a network with multiple hidden unit can and so one does not really need to have multiple hidden layers.

The algorithms to train multilayer perceptrons are not fool-proof though, in the sense that we cannot guarantee that network will eventually learn anything it is capable to. Practically though, in most cases it is able to. Even when it is not able to learn, the only trouble the programmer has to take it to run the learning programonce again3.

*2 As total combinations represented by five neurons is 25which is 32. Same as the case with 64.*

*3 The search space of multi layer perceptron is highly convoluted, full of local maxima, thus the network might stop due to that reason. When the program starts all over again, it starts with another set of random values and effectively starting from a new place in the solution space and more likely to reach to a solution.*

Hidden layers help the network learn about features of the inputs. For example in the case of our character recognition problem of character A, the hidden layer learns about the features of the character input. You can easily see that different samples of each character has different input cells turned on and if the decision is made based on which units are active, it would be incorrect 4.

Here, the hidden layer comes to the rescue. In case of A, one hidden layer might learn to remain active when the input produces a slanting line rising from left to right. Another hidden unit may learn to remain active when a straight horizontal line is found. One more hidden unit learn to remain active when there is a slanting line coming down from left to right. When all these three hidden units are active the output unit combination which represents the character A which learn to be active. A may be represented by 000001 as an output, that means whenever these three hidden units are on (that means there are three lines, two slanting in different directions and one horizontal), first five output neurons learns to have 0 while the last output learn to have 1.

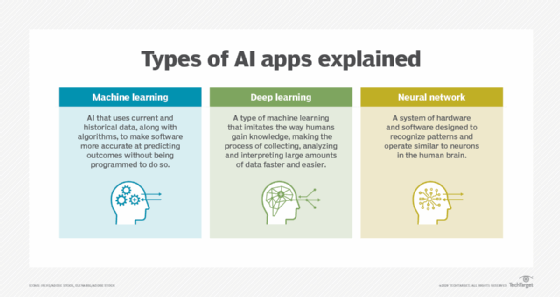
## What is a backpropagation algorithm?

Backpropagation, or backward propagation of errors, is an [algorithm](https://www.techtarget.com/whatis/definition/algorithm) that is designed to test for errors working back from output nodes to input nodes. It's an important mathematical tool for improving the accuracy of predictions in [data mining](https://www.techtarget.com/searchbusinessanalytics/definition/data-mining) and [machine learning](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML). Essentially, backpropagation is an algorithm used to quickly calculate derivatives in a [neural network](https://www.techtarget.com/searchenterpriseai/definition/neural-network), which are the changes in output because of tuning and adjustments.

There are two leading types of backpropagation networks:

* **Static backpropagation.** Static backpropagation is a network developed to map static inputs for static outputs. Static networks can solve static classification problems, such as optical character recognition ([OCR](https://www.techtarget.com/searchcontentmanagement/definition/OCR-optical-character-recognition)).
* **Recurrent backpropagation.** The recurrent backpropagation network is used for fixed-point learning. This means that during neural network training, the weights are numerical values that determine how much nodes -- also referred to as neurons -- influence output values. They're adjusted so that the network can achieve stability by reaching a fixed value.

The key difference here is that static backpropagation offers instant mapping, while recurrent backpropagation does not.

Find out how machine learning, deep learning and neural networks compare.

## What is a backpropagation algorithm in a neural network?

Artificial neural networks (ANNs) and deep neural networks use backpropagation as a learning algorithm to compute a gradient descent, which is an optimization algorithm that guides the user to the maximum or minimum of a function.

In a machine learning context, the gradient descent helps the system minimize the gap between desired outputs and achieved system outputs. The algorithm tunes the system by adjusting the weight values for various inputs to narrow the difference between outputs. This is also known as the error between the two.

More specifically, a gradient descent algorithm uses a gradual process to provide information on how a network's parameters need to be adjusted to reduce the disparity between the desired and achieved outputs. An evaluation metric called a cost function guides this process. The cost function is a mathematical function that measures this error. The algorithm's goal is to determine how the parameters must be adjusted to reduce the cost function and improve overall accuracy.

**THIS ARTICLE IS PART OF**

### [What is machine learning and how does it work? In-depth guide](https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML)

* Which also includes:
* [**4 types of learning in machine learning explained**](https://www.techtarget.com/searchenterpriseai/tip/Types-of-learning-in-machine-learning-explained)
* [**How to build a machine learning model in 7 steps**](https://www.techtarget.com/searchenterpriseai/feature/How-to-build-a-machine-learning-model-in-7-steps)
* [**CNN vs. RNN: How are they different?**](https://www.techtarget.com/searchenterpriseai/feature/CNN-vs-RNN-How-they-differ-and-where-they-overlap)

In backpropagation, this error is propagated backward from the output layer or output neuron through the hidden layers toward the input layer so that neurons can adjust themselves along the way if they played a role in producing the error. Activation functions [activate neurons to learn new complex patterns](https://www.techtarget.com/searchenterpriseai/feature/How-neural-network-training-methods-are-modeled-after-the-human-brain), information and whatever else they need to adjust their weights and biases, and mitigate this error to improve the network.

The algorithm gets its descent gradient name because the weights are updated backward, from output to input.

## What is the objective of a backpropagation algorithm?

Backpropagation algorithms are used extensively to train feedforward neural networks, such as [convolutional neural networks](https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network), in areas such as [deep learning](https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network). A backpropagation algorithm is pragmatic because it computes the gradient needed to adjust a network's weights more efficiently than computing the gradient based on each individual weight. It enables the use of gradient methods, such as gradient descent and stochastic gradient descent, to train multilayer networks and update weights to minimize errors.

It's not easy to understand exactly how changing weights and biases affect the overall behavior of an ANN. That was one factor that held back more comprehensive use of neural network applications until the early 2000s, when computers provided the necessary insight.

Today, backpropagation algorithms have practical applications in many areas of artificial intelligence, including OCR, [natural language processing](https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP) and image processing.

## Advantages and disadvantages of backpropagation algorithms

There are several advantages to using a backpropagation algorithm, but there are also challenges.

### Advantages of backpropagation algorithms

* They don't have any parameters to tune except for the number of inputs.
* They're highly adaptable and efficient, and don't require prior knowledge about the network.
* They use a standard process that usually works well.
* They're user-friendly, fast and easy to program.
* Users don't need to learn any special functions.

### Disadvantages of backpropagation algorithms

* They prefer a matrix-based approach over a mini-batch approach.
* Data mining is sensitive to [noisy data](https://www.techtarget.com/searchbusinessanalytics/definition/noisy-data) and other irregularities. Unclean data can affect the backpropagation algorithm when training a neural network used for data mining.
* Performance is highly dependent on input data.
* Training is time- and resource-intensive.

## What is a backpropagation algorithm in machine learning?

Backpropagation is a type of [supervised learning](https://www.techtarget.com/searchenterpriseai/definition/supervised-learning) since it requires a known, desired output for each input value to calculate the loss function gradient, which is how desired output values differ from actual output. Supervised learning, the most common training approach in machine learning, uses a training data set that has clearly labeled data and specified desired outputs.

Along with classifier algorithms such as naive Bayesian filters, K-nearest neighbors and support vector machines, the backpropagation training algorithm has emerged as an important part of machine learning applications that involve [predictive analytics](https://www.techtarget.com/searchbusinessanalytics/definition/predictive-analytics). While backpropagation techniques are mainly applied to neural networks, they can also be applied to both classification and regression problems in machine learning. In real-world applications, developers and machine learning experts implement backpropagation algorithms for neural networks using programming languages such as Python.

## What is the time complexity of a backpropagation algorithm?

The time complexity of each iteration -- how long it takes to execute each statement in an algorithm -- depends on the network's structure. In the early days of deep learning, a multilayer [perceptron](https://www.techtarget.com/whatis/definition/perceptron) was a basic form of neural network consisting of an input layer, hidden units and an output unit. The time complexity was low compared with today's networks, which can have exponentially more parameters. Therefore, the sheer size of a neural network is the primary factor affecting time complexity, but there are other factors, such as the [size of training data sets](https://www.techtarget.com/searchenterpriseai/feature/Using-small-data-sets-for-machine-learning-models-sees-growth) or the amount of data used to train networks.

Essentially, the number of neurons and parameters directly affects how backpropagation works. During a forward pass, in which input data moves forward from the input layer to the next layer and so on, the time complexity is larger when there are more neurons involved. During the subsequent backward pass, where parameters are adjusted to rectify an error, more parameters also mean more of a time complexity.

## What is a backpropagation momentum algorithm?

Using gradient descent optimization algorithms for tuning weights to reduce an error can be time-consuming. That's why the concept of momentum in backpropagation is used to speed up this process. It states that previous weight changes must influence the present direction of movement in weight space. Simply put, an aggregate of past weight changes is used to influence a current one.

During optimization, it's possible for gradients to change direction, which would appear to complicate the overall process. That is why this momentum technique is used to ensure optimization continues moving in the right direction and the performance of the neural network improves.

## What is a backpropagation algorithm pseudocode?

The backpropagation algorithm [pseudocode](https://www.techtarget.com/whatis/definition/pseudocode) is a basic blueprint that developers and researchers can use to conduct the backpropagation process. It's a high-level overview with plain language instructions as well as the code snippets to perform the most essential tasks in the process.

While this overview covers the essentials, the actual implementation typically is far more complex. The pseudocode covers the steps that need to get done; it typically reads like a sequential series of actions, and within it are all the core components that the backpropagation process will involve. Each pseudocode instance is pertinent to a specific context, and any [common programming language](https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/) can be used to write it, such as Python and other object-oriented programming languages.

## What is the Levenberg-Marquardt backpropagation algorithm?

The Levenberg-Marquardt algorithm is another technique that helps adjust neural network weights and biases during training. However, within the context of training neural networks, it's not an alternative or replacement for a backpropagation algorithm, but rather an optimization technique used within backpropagation-based training.

To reduce neural network errors, Levenberg-Marquardt blends gradient information from the gradient descent method with insights from what is called the Gauss-Newton algorithm -- where gradient information is represented in a curved format using mathematical matrices -- as a method of guiding updates and speeding up what would take a traditional gradient descent method a longer time to complete.

1. **Write short notes on:**
   * + 1. **Artificial neuron**

**ANS:-**

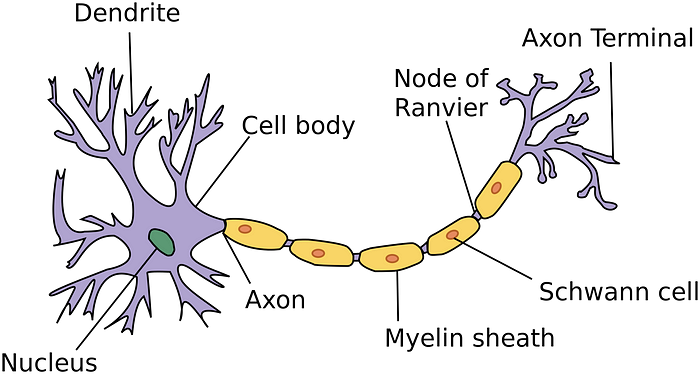
**What the heck are artificial neurons?**

Artificial neurons are **inspired**by biology and try to replicate how our brain works. Our brain has billions of neural cells that are helping us to interpret so called signals. You can imagine a signal as anything ranging from a ray of light that hits our eyes or our own thought that we want to move our arm. Neurons help us to interpret these signals in the right way. For example when a ray of light hits our eyes, a neuron will know that this ray of light can be intrepreted as blue, green or yellow. Now, artificial neurons are the approach to transfer this principle into a computer by **replicating**a biological neuron in form of code.

**How can we replicate biological neurons in code?**

You probably ask yourself how it is possible to recreate something that is living and put it into a code that is based on these ones and zeros that you know from the Matrix movies. Let me take you further down the rabbit hole and explain it to you step by step.

**How does a biological neuron function?**



“Anatomy and Physiology” by the US National Cancer Institute’s Surveillance, Epidemiology and End Results (SEER) Program, extracted from Wikipedia.com

Take a look at the illustration of a biological neuron, note all the different terminologies and then forget about them. You do not need to understand them in detail but what is important are their functions. A neuron is generally structured in three parts. The first part is the cell body that receives a signal and **processes**it. If the signal is relevant, the cell body will get **excited**and charges up electricity. If the electricity reaches a certain treshhold, the cell body will allow his excitement to be transferred to the next cell. This **transfer** is performed by the axon which **connects**to the next cell via the axon terminal. Each neuron has its own **unique**setting, meaning that the degree of excitement varies between the cells. While some cells get excited about everything, there are cells with no excitement, causing an **inhibitory**effect.

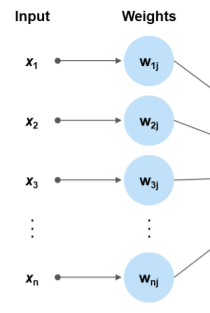
**How do we replicate this behavior in artificial neurons?**

Now, let us break these things down and try to formulate them in a way a computer can understand them. We begin by looking at our signals. Whether it is a ray of light or some kind of thought, we can express this information by assigning it some kind of numerical value. Yellow could be 1, blue could be 2, and green could be a 3. This way we make the information interpretable to a machine. In terms of computer science, we also call these signals our **input**.

https://miro.medium.com/v2/resize:fit:57/1*ESlD-T-zJXQBbUaxVuoxOw.png

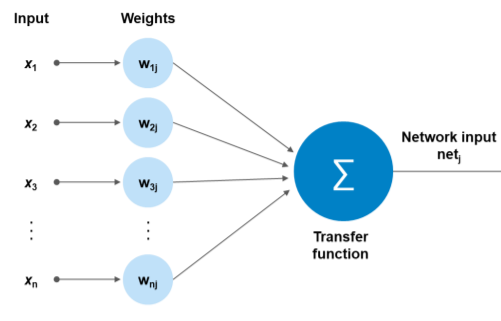
Representation of the input element of an artificial neuron. Geshenson 2003. Illustration by author.

Remember that a signal can make an actual neuron excited or not? An artificial neuron has this ability too! We can mimic the natural properties of an actual neuron by introducing so called **weights**. A weight is **multiplied**with the input and results in a **new**value. Coming back to our colour detection example, we could set the properties of an artificial neuron so that it always gets excited about the colour yellow but not so much over the colours green and blue. In a simplified mathematical way, the artificial neuron multiplies 1 with a weight of 10 but multiplies green and blue only with a weight of 1. In result, this particular artificial neuron will always get **more excited** about the colour yellow than any other colour.



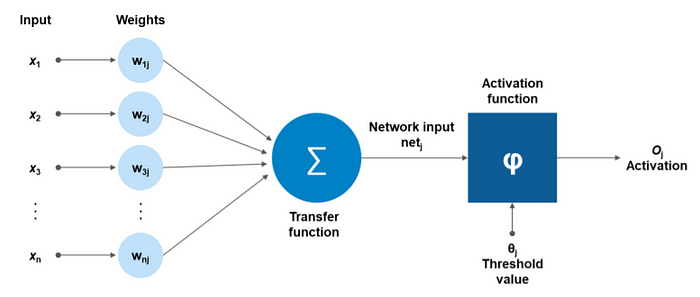
Representation of input and weights elements of an artificial neuron. Geshenson 2003. Illustration by author.

Now, a biological neuron accumulates all this information and charges itself with electricity. An artificial neuron implemented in code tries to copy this behaviour by summing up all the different weighted values for each input. We call this the **transfer function**. The sum of all newly calculated, weighted values is called the **network input.**



Representation of transfer function and other elements of an artificial neuron. Geshenson 2003. Illustration by author.

After a neuron is charged up, it releases the electricity to the next neuron, if a certain **threshold**is exceeded. We can implement this behavior by introducing a so called **activation function** to our artificial neuron. An activation function takes the network input and checks, if it **exceeds**its given threshhold. If it is larger than the threshold, the artificial neuron will decide to “fire” and pass his information to the next neuron. We referr to this state also as an artificial neuron being **activated**.



Representation of an artificial neuron and its elements. Geshenson 2003. Illustration by author.

# Artificial Neural Network

Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions.

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

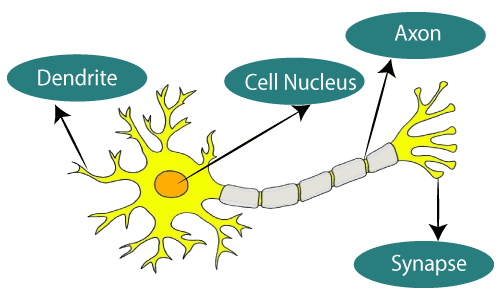
Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

## What is Artificial Neural Network?

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.

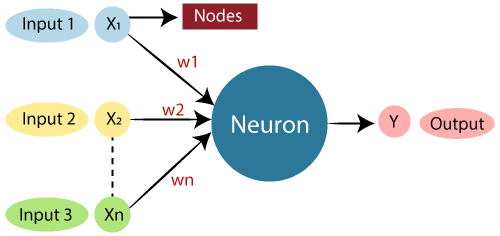
Backward Skip 10sPlay VideoForward Skip 10s

ADVERTISEMENT



**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.

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.

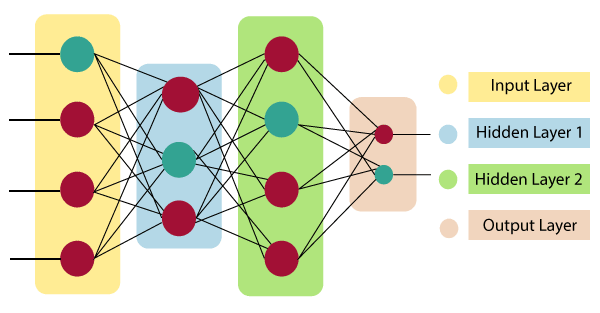
There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly.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



**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.

What is Artificial Neural Network

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.

## Advantages of Artificial Neural Network (ANN)

**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**

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.

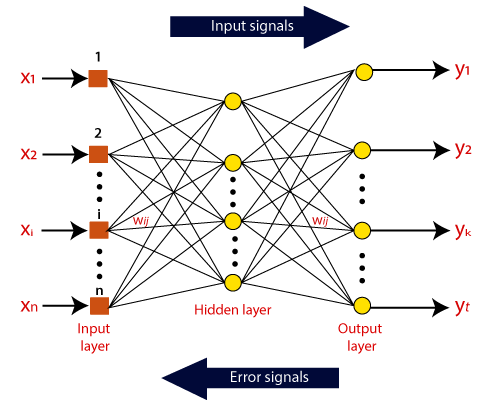
**The duration of the network is unknown:**

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

Science artificial neural networks that have steeped into the world in the mid-20th century are exponentially developing. In the present time, we have investigated the pros of artificial neural networks and the issues encountered in the course of their utilization. It should not be overlooked that the cons of ANN networks, which are a flourishing science branch, are eliminated individually, and their pros are increasing day by day. It means that artificial neural networks will turn into an irreplaceable part of our lives progressively important.

## 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.

## Types of Artificial Neural Network:

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks. For example, segmentation or classification.

### Feedback ANN:

In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. As per the **University of Massachusetts**, Lowell Centre for Atmospheric Research. The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

* + - 1. **Multi-layer perceptron**

**ANS:-**

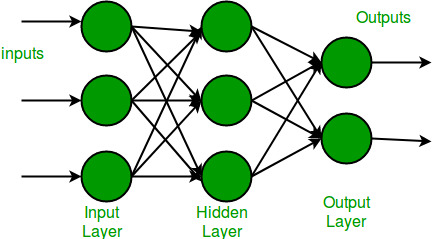
## Multi-layer Perceptron

Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

A gentle introduction to **neural networks and TensorFlow** can be found here:

* [Neural Networks](https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/)
* [Introduction to TensorFlow](https://www.geeksforgeeks.org/introduction-to-tensorflow/)

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.



In the multi-layer perceptron diagram above, we can see that there are three inputs and thus three input nodes and the hidden layer has three nodes. The output layer gives two outputs, therefore there are two output nodes. The nodes in the input layer take input and forward it for further process, in the diagram above the nodes in the input layer forwards their output to each of the three nodes in the hidden layer, and in the same way, the hidden layer processes the information and passes it to the output layer.

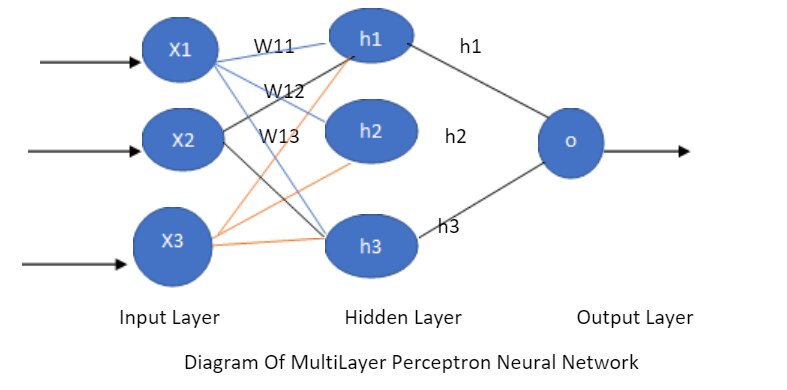
Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.

**What is a Multilayer Perceptron Neural Network?**

A multilayer perceptron (MLP) Neural network belongs to the feedforward neural network. It is an Artificial Neural Network in which all nodes are interconnected with nodes of different layers.

Frank Rosenblatt first defined the word Perceptron in his perceptron program. Perceptron is a basic unit of an artificial neural network that defines the artificial neuron in the neural network. It is a supervised learning algorithm containing nodes’ values, activation functions, inputs, and weights to calculate the output.

The Multilayer Perceptron (MLP) Neural Network works only in the forward direction. All nodes are fully connected to the network. Each node passes its value to the coming node only in the forward direction. The MLP neural network uses a Backpropagation algorithm to increase the accuracy of the training model.



**Must Read:**[**Deep Learning vs Machine Learning – Concepts, Applications, and Key Differences**](https://www.shiksha.com/online-courses/articles/deep-learning-vs-machine-learning-concepts-applications-and-key-differences/)

### ****Structure of MultiLayer Perceptron Neural Network****

This network has three main layers that combine to form a complete Artificial Neural Network. These layers are as follows:

#### ****Input Layer****

It is the initial or starting layer of the Multilayer perceptron. It takes input from the training data set and forwards it to the hidden layer. There are n input nodes in the input layer. The number of input nodes depends on the number of dataset features. Each input vector variable is distributed to each of the nodes of the hidden layer.

**Must Explore –**[**Data Science Courses**](https://www.shiksha.com/online-courses/data-science-courses-certification-training-ct123)

#### ****Hidden Layer****

It is the heart of all Artificial neural networks. This layer comprises all computations of the neural network. The edges of the hidden layer have weights multiplied by the node values. This layer uses the activation function.

There can be one or two hidden layers in the model.

Several hidden layer nodes should be accurate as few nodes in the hidden layer make the model unable to work efficiently with complex data. More nodes will result in an overfitting problem.

#### ****Output Layer****

This layer gives the estimated output of the Neural Network. The number of nodes in the output layer depends on the type of problem. For a single targeted variable, use one node. N classification problem, ANN uses N nodes in the output layer.

### ****Working of MultiLayer Perceptron Neural Network****

* The input node represents the feature of the dataset.
* Each input node passes the vector input value to the hidden layer.
* In the hidden layer, each edge has some weight multiplied by the input variable. All the production values from the hidden nodes are summed together. To generate the output
* The activation function is used in the hidden layer to identify the active nodes.
* The output is passed to the output layer.
* Calculate the difference between predicted and actual output at the output layer.
* The model uses backpropagation after calculating the predicted output.

### ****Difference Between Multilayer Perceptron Neural Network and Conventional Neural Network****

|  |  |  |
| --- | --- | --- |
|  | **MultiLayer Perceptron Neural Network** | **Convolutional Neural Network** |
| **Types of Input** | It takes vector inputs. | It takes both vectors and matrices as input. |
| **Network Type** | It is a fully connected Neural network | It is a spatially connected neural network. |
| **Focus Problem** | It can deal with non-linear problems. | Can only deal with linear problems. |
| **Application** | It is good for simple image classification. | It is mostly used for complex image classification. |

## ****Advantages of MultiLayer Perceptron Neural Network****

1. MultiLayer Perceptron Neural Networks can easily work with non-linear problems.
2. It can handle complex problems while dealing with large datasets.
3. Developers use this model to deal with the fitness problem of Neural Networks.
4. It has a higher accuracy rate and reduces prediction error by using backpropagation.
5. After training the model, the Multilayer Perceptron Neural Network quickly predicts the output.

## ****Disadvantages of MultiLayer Perceptron Neural Network****

1. This Neural Network consists of large computation, which sometimes increases the overall cost of the model.
2. The model will perform well only when it is trained perfectly.
3. Due to this model’s tight connections, the number of parameters and node redundancy increases.
   * + 1. **Deep learning**

**ANS:-**

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don’t need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain’s biological neurons, and they are designed to learn from large amounts of data.

1. Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. Neural networks are modeled after the structure and function of the human brain and consist of layers of interconnected nodes that process and transform data.
2. The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data. Deep Learning algorithms can automatically learn and improve from data without the need for manual feature engineering.
3. Deep Learning has achieved significant success in various fields, including image recognition, natural language processing, speech recognition, and recommendation systems. Some of the popular Deep Learning architectures include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Deep Belief Networks (DBNs).
4. Training deep neural networks typically requires a large amount of data and computational resources. However, the availability of cloud computing and the development of specialized hardware, such as Graphics Processing Units (GPUs), has made it easier to train deep neural networks.

In summary, Deep Learning is a subfield of Machine Learning that involves the use of deep neural networks to model and solve complex problems. Deep Learning has achieved significant success in various fields, and its use is expected to continue to grow as more data becomes available, and more powerful computing resources become available.

## ****What is Deep Learning?****

Deep learning is the branch of [machine learning](https://www.geeksforgeeks.org/machine-learning/) which is based on artificial neural network architecture. An artificial neural network or ANN uses layers of interconnected nodes called neurons that work together to process and learn from the input data.

In a fully connected Deep neural network, there is an input layer and one or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. The layers of the neural network transform the input data through a series of nonlinear transformations, allowing the network to learn complex representations of the input data.

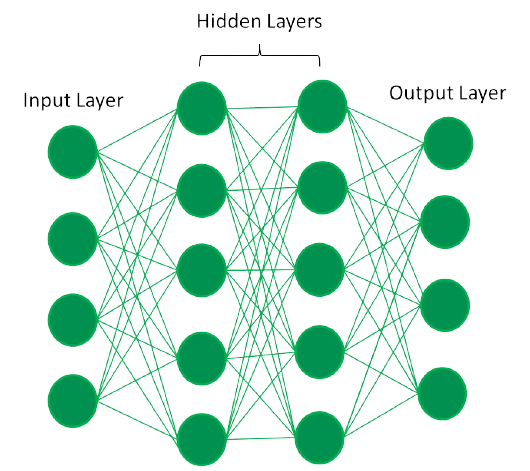
Today Deep learning has become one of the most popular and visible areas of machine learning, due to its success in a variety of applications, such as computer vision, natural language processing, and Reinforcement learning.

Deep learning can be used for supervised, unsupervised as well as reinforcement machine learning. it uses a variety of ways to process these.

* **Supervised Machine Learning:** [Supervised machine learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) is the [machine learning](https://www.geeksforgeeks.org/machine-learning/) technique in which the neural network learns to make predictions or classify data based on the labeled datasets. Here we input both input features along with the target variables. the neural network learns to make predictions based on the cost or error that comes from the difference between the predicted and the actual target, this process is known as backpropagation.  Deep learning algorithms like Convolutional neural networks, Recurrent neural networks are used for many supervised tasks like image classifications and recognization, sentiment analysis, language translations, etc.
* **Unsupervised Machine Learning:** [Unsupervised machine learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) is the [machine learning](https://www.geeksforgeeks.org/machine-learning/) technique in which the neural network learns to discover the patterns or to cluster the dataset based on unlabeled datasets. Here there are no target variables. while the machine has to self-determined the hidden patterns or relationships within the datasets. Deep learning algorithms like autoencoders and generative models are used for unsupervised tasks like clustering, dimensionality reduction, and anomaly detection.
* **Reinforcement  Machine Learning**: [Reinforcement  Machine Learning](https://www.geeksforgeeks.org/what-is-reinforcement-learning/) is the [machine learning](https://www.geeksforgeeks.org/machine-learning/) technique in which an agent learns to make decisions in an environment to maximize a reward signal. The agent interacts with the environment by taking action and observing the resulting rewards. Deep learning can be used to learn policies, or a set of actions, that maximizes the cumulative reward over time. Deep reinforcement learning algorithms like Deep Q networks and Deep Deterministic Policy Gradient (DDPG) are used to reinforce tasks like robotics and game playing etc.

### Artificial neural networks

[Artificial neural networks](https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/) are built on the principles of the structure and operation of human neurons. It is also known as neural networks or neural nets. An artificial neural network’s input layer, which is the first layer, receives input from external sources and passes it on to the hidden layer, which is the second layer. Each neuron in the hidden layer gets information from the neurons in the previous layer, computes the weighted total, and then transfers it to the neurons in the next layer. These connections are weighted, which means that the impacts of the inputs from the preceding layer are more or less optimized by giving each input a distinct weight. These weights are then adjusted during the training process to enhance the performance of the model.



*Fully Connected Artificial Neural Network*

Artificial neurons, also known as units, are found in artificial neural networks. The whole Artificial Neural Network is composed of these artificial neurons, which are arranged in a series of layers. The complexities of neural networks will depend on the complexities of the underlying patterns in the dataset whether a layer has a dozen units or millions of units.  Commonly, Artificial Neural Network has an input layer, an output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about.

In a fully connected artificial neural network, there is an input layer and one or more hidden layers connected one after the other. Each neuron receives input from the previous layer neurons or the input layer. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. Then, after passing through one or more hidden layers, this data is transformed into valuable data for the output layer. Finally, the output layer provides an output in the form of an artificial neural network’s response to the data that comes in.

Units are linked to one another from one layer to another in the bulk of neural networks. Each of these links has weights that control how much one unit influences another. The neural network learns more and more about the data as it moves from one unit to another, ultimately producing an output from the output layer.

### ****Difference between Machine Learning and Deep Learning :****

[machine learning](https://www.geeksforgeeks.org/machine-learning/) and deep learning both are subsets of artificial intelligence but there are many similarities and differences between them.

| **Machine Learning** | **Deep Learning** |
| --- | --- |
| Apply statistical algorithms to learn the hidden patterns and relationships in the dataset. | Uses artificial neural network architecture to learn the hidden patterns and relationships in the dataset. |
| Can work on the smaller amount of dataset | Requires the larger volume of dataset compared to machine learning |
| Better for the low-label task. | Better for complex task like image processing, natural language processing, etc. |
| Takes less time to train the model. | Takes more time to train the model. |
| A model is created by relevant features which are manually extracted from images to detect an object in the image. | Relevant features are automatically extracted from images. It is an end-to-end learning process. |
| Less complex and easy to interpret the result. | More complex, it works like the black box interpretations of the result are not easy. |
| It can work on the CPU or requires less computing power as compared to deep learning. | It requires a high-performance computer with GPU. |

### Types of neural networks

Deep Learning models are able to automatically learn features from the data, which makes them well-suited for tasks such as image recognition, speech recognition, and natural language processing. The most widely used architectures in deep learning are feedforward neural networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs).

[Feedforward neural networks (FNNs)](https://www.geeksforgeeks.org/understanding-multi-layer-feed-forward-networks/) are the simplest type of ANN, with a linear flow of information through the network. FNNs have been widely used for tasks such as image classification, speech recognition, and natural language processing.

[Convolutional Neural Networks (CNNs)](https://www.geeksforgeeks.org/introduction-convolution-neural-network/) are specifically for image and video recognition tasks. CNNs are able to automatically learn features from the images, which makes them well-suited for tasks such as image classification, object detection, and image segmentation.

[Recurrent Neural Networks (RNNs)](https://www.geeksforgeeks.org/recurrent-neural-networks-explanation/)are a type of neural network that is able to process sequential data, such as time series and natural language. RNNs are able to maintain an internal state that captures information about the previous inputs, which makes them well-suited for tasks such as speech recognition, natural language processing, and language translation.

## ****Applications of Deep Learning :****

The main applications of deep learning can be divided into computer vision, natural language processing (NLP), and reinforcement learning.

### [Computer vision](https://www.geeksforgeeks.org/applications-of-computer-vision/)

In [computer vision](https://www.geeksforgeeks.org/applications-of-computer-vision/), Deep learning models can enable machines to identify and understand visual data. Some of the main applications of deep learning in computer vision include:

* **Object detection and recognition:**Deep learning model can be used to identify and locate objects within images and videos, making it possible for machines to perform tasks such as self-driving cars, surveillance, and robotics.
* **Image classification:**Deep learning models can be used to classify images into categories such as animals, plants, and buildings. This is used in applications such as medical imaging, quality control, and image retrieval.
* **Image segmentation:**Deep learning models can be used for image segmentation into different regions, making it possible to identify specific features within images.

#### [Natural language processing (NLP)](https://www.geeksforgeeks.org/natural-language-processing-nlp-tutorial/):

In [NLP](https://www.geeksforgeeks.org/natural-language-processing-nlp-tutorial/), the Deep learning model can enable machines to understand and generate human language. Some of the main applications of deep learning in [NLP](https://www.geeksforgeeks.org/natural-language-processing-nlp-tutorial/) include:

* **Automatic Text Generation** – Deep learning model can learn the corpus of text and new text like summaries, essays can be automatically generated using these trained models.
* **Language translation:** Deep learning models can translate text from one language to another, making it possible to communicate with people from different linguistic backgrounds.
* **Sentiment analysis:**Deep learning models can analyze the sentiment of a piece of text, making it possible to determine whether the text is positive, negative, or neutral. This is used in applications such as customer service, social media monitoring, and political analysis.
* **Speech recognition:** Deep learning models can recognize and transcribe spoken words, making it possible to perform tasks such as speech-to-text conversion, voice search, and voice-controlled devices.

#### [Reinforcement learning:](https://www.geeksforgeeks.org/what-is-reinforcement-learning/)

In [reinforcement learning](https://www.geeksforgeeks.org/what-is-reinforcement-learning/), deep learning works as training agents to take action in an environment to maximize a reward. Some of the main applications of deep learning in reinforcement learning include:

* **Game playing:**Deep reinforcement learning models have been able to beat human experts at games such as Go, Chess, and Atari.
* **Robotics:**Deep reinforcement learning models can be used to train robots to perform complex tasks such as grasping objects, navigation, and manipulation.
* **Control systems:**Deep reinforcement learning models can be used to control complex systems such as power grids, traffic management, and supply chain optimization.

## ****Challenges in Deep Learning****

Deep learning has made significant advancements in various fields, but there are still some challenges that need to be addressed. Here are some of the main challenges in deep learning:

1. Data availability: It requires large amounts of data to learn from. For using deep learning it’s a big concern to gather as much data for training.
2. Computational Resources: For training the deep learning model, it is computationally expensive because it requires specialized hardware like GPUs and TPUs.
3. Time-consuming: While working on sequential data depending on the computational resource it can take very large even in days or months.
4. Interpretability: Deep learning models are complex, it works like a black box. it is very difficult to interpret the result.
5. Overfitting: when the model is trained again and again, it becomes too specialized for the training data, leading to overfitting and poor performance on new data.

### Advantages of Deep Learning:

1. High accuracy: Deep Learning algorithms can achieve state-of-the-art performance in various tasks, such as image recognition and natural language processing.
2. Automated feature engineering: Deep Learning algorithms can automatically discover and learn relevant features from data without the need for manual feature engineering.
3. Scalability: Deep Learning models can scale to handle large and complex datasets, and can learn from massive amounts of data.
4. Flexibility: Deep Learning models can be applied to a wide range of tasks and can handle various types of data, such as images, text, and speech.
5. Continual improvement: Deep Learning models can continually improve their performance as more data becomes available.

### Disadvantages of Deep Learning:

1. High computational requirements: Deep Learning models require large amounts of data and computational resources to train and optimize.
2. Requires large amounts of labeled data: Deep Learning models often require a large amount of labeled data for training, which can be expensive and time- consuming to acquire.
3. Interpretability: Deep Learning models can be challenging to interpret, making it difficult to understand how they make decisions.  
   Overfitting: Deep Learning models can sometimes overfit to the training data, resulting in poor performance on new and unseen data.
4. Black-box nature: Deep Learning models are often treated as black boxes, making it difficult to understand how they work and how they arrived at their predictions.  
   In summary, while Deep Learning offers many advantages, including high accuracy and scalability, it also has some disadvantages, such as high computational requirements, the need for large amounts of labeled data, and interpretability challenges. These limitations need to be carefully considered when deciding whether to use Deep Learning for a specific task.
   * + 1. **Learning rate**

**ANS:-**

# What is Learning Rate?

Learning rate (λ) is one such **hyper-parameter** that defines the **adjustment in the weights of our network with respect to the loss gradient descent**. It determines how fast or slow we will move towards the optimal weights

The Gradient Descent Algorithm estimates the weights of the model in many iterations by minimizing a cost function at every step.

Here is the algorithm:

Repeat until convergence {  
   
 Wj = Wj - λ θF(Wj)/θWj  
   
}

Where:

* **Wj** is the weight
* **θ** is the theta
* **F(Wj)** is the cost function respectively.

In order for Gradient Descent to work, we must set the learning rate to an appropriate value. This parameter **determines how fast or slow we will move towards the optimal weights**. If the learning rate is very large we will skip the optimal solution. If it is too small we will need too many iterations to converge to the best values. So using a good learning rate is crucial.

In simple language, we can define learning rate as how quickly our network abandons the concepts it has learned up until now for new ones.

# Learning rate explained through a child’s interaction

To understand this better let’s consider an example.

If a child sees **ten dogs** and all of them are black in color, he might believe that all dogs are black and would consider this as a feature when trying to identify a dog.

Imagine he’s shown a white dog, and his parents tell him that it’s a dog. With a **desirable learning rate**, he would quickly understand that black color is not an important feature of dogs and would look for another feature.

But with **a low learning rate**, he would consider the white dog as an outlier and would continue to believe that all dogs are black.

And if the **learning rate is too high**, he would instantly start to believe that all dogs are white even though he has seen more black dogs than white ones.

The point is it’s’ really important to **achieve a desirable learning rate** because:

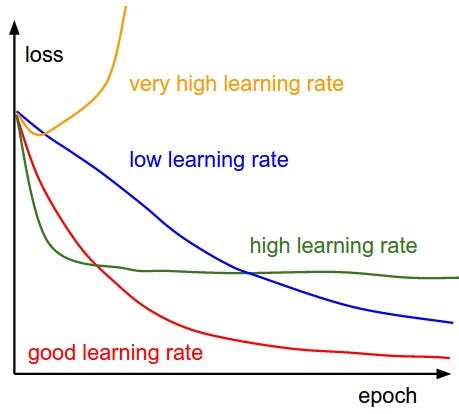
* both low and high learning rates results in wasted time and resources
* A lower learning rate means more training time
* more time results in increased cloud GPU costs
* a higher rate could result in a model that might not be able to predict anything accurately

A desirable learning rate is one that’s low enough so that the network converges to something useful but high enough so that it can be trained in a reasonable amount of time.

# Tuning the learning rate

The learning rate is the most important hyper-parameter for tuning neural networks. A good learning rate could be the difference between a model that doesn’t learn anything and a model that presents state-of-the-art results.

The below diagram demonstrates the different scenarios one can fall into when configuring the learning rate.



Effect of various learning rates on convergence (Img Credit: [cs231n](http://cs231n.github.io/neural-networks-3/))

The obvious way to find a desirable or optimal learning rate is through trial and error. To do this efficiently, there are a few ways that we should adhere to.

# Search from coarse to fine learning rate

A learning rate of 0.01 and 0.011 are unlikely to yield vastly different results. Even if they did, searching in such small increments is very costly and inefficient: what if both learning rates caused the model to diverge? The time spent training would be a waste.

A more efficient way is to try widely different learning rates to determine the range of learning rates you should explore and concentrate your efforts there.

For instance, whenever I am trying to tune the learning rate, I generally start off by searching across the learning rates 1e-7, 1e-6, 1e-5, … 0.01, 0.1, 1. In other words, I search across various orders of 10 to find an optimal range of learning rates. Then, I search in smaller increments. For instance, if I found the optimal range to be somewhere between 0.01 and 0.1, I would then start searching learning rates in that range such as 0.03. This is exactly similar to the idea behind binary search and is a widely applicable technique.

# Don’t start off by training on the entire dataset

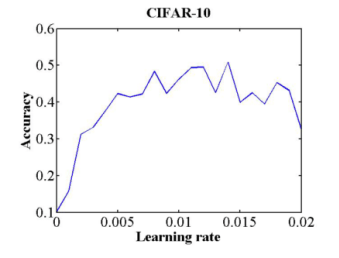
Starting off with the entire dataset is likely to be a waste of time. Chances are, some learning rates will cause your loss to diverge or fluctuate and some learning rates are likely to train very slowly, so these orders of learning rates can be removed from your search after a few iterations.

Although trial and error is a relatively fail-proof way of tuning the learning rate, a more efficient way which does minimal training to find the best learning rate to start from is **“Learning Rate Test”**.  
The basic idea is that you want to efficiently find the maximum learning rate you can use which will improve the loss. In order to find that value, you train slightly with multiple learning rates and see how the loss changes.  
The actual procedure is:

Choose a minimum and maximum learning rate to search through (e.g. 1e-7 and 0.1)  
Train the model for several epochs using SGD while linearly increasing the learning rate from the minimum to maximum learning rate.

At each iteration, record the accuracy (or loss).

Plot the test accuracy, and see where the loss/accuracy starts to improve, and when it starts to get worse/plateau/to become ragged. The latter learning rate is the maximum learning rate that converges and is a good value for your initial learning rate. The former learning rate, or 1/3–1/4 of the maximum learning rates is a good minimum learning rate that you can decrease if you are using learning rate decay.



If the test accuracy curve looks like the above diagram, a good learning rate to begin from would be 0.006, where the loss starts to become jagged.

# Learning rate should be adaptive

This method of improving the convergence rate of hyper-parameters reduces the need for the manual tuning of the initial learning rate. This method works by dynamically updating the learning rate during optimization using the gradient with respect to the learning rate of the update rule itself. Computing this “hyper-gradient” needs little additional computation, requires only one extra copy of the original gradient to be stored in memory, and relies upon nothing more than what is provided by reverse-mode automatic differentiation.

* 1. **Write the difference between:-**
  2. **Activation function vs threshold function**

**ANS:-**

The threshold function differs from other activation functions in the following ways -

1. **Output range:** The threshold function outputs a binary value (0 or 1) based on whether the input is greater than or equal to a threshold value, while other activation functions such as sigmoid or ReLU output continuous values between 0 and 1 or between 0 and the input value, respectively.
2. **Differentiability:** The threshold function is non-differentiable, which means it cannot be used in certain types of neural networks that require differentiable activation functions, while other activation functions such as sigmoid or ReLU are differentiable.
3. **Interpretability:** The threshold function is simple and interpretable, making it easy to understand and use in a variety of applications, while other activation functions such as softmax or hyperbolic tangent may be more difficult to interpret.
4. **Sensitivity to input:** The threshold function is highly sensitive to the choice of the threshold value, which can affect the accuracy of the classification, while other activation functions such as sigmoid or ReLU may be less sensitive to input values.
5. **Smoothness:** The threshold function is discontinuous and has a sharp edge at the threshold value, while other activation functions such as sigmoid or ReLU are smooth and have a continuous gradient.

**2.Step function vs sigmoid function**

**ANS:-**

### Step Activation Function

The step activation function is a simple but effective activation function. It takes a single real number as input and outputs a 0 or 1, depending on whether the input is greater than or equal to a threshold value. The equation for the step activation function is as follows:

f(x) = 1 if x >= threshold

f(x) = 0 if x < threshold

The step activation function is often used in binary classification problems, where the goal is to classify an input into one of two categories. For example, the step activation function could be used to classify a tumor as benign or malignant.

### Sigmoid Activation Function

The sigmoid activation function is a more complex activation function that produces a smooth, S-shaped curve. The equation for the sigmoid activation function is as follows:

f(x) = 1 / (1 + e^(-x))

The sigmoid activation function is often used in multi-class classification problems, where the goal is to classify an input into one of several categories. For example, the sigmoid activation function could be used to classify a flower as a rose, tulip, or daisy.

### Major Differences

The main difference between the step activation function and the sigmoid activation function is the range of outputs that they produce.

The step activation function produces only two outputs, 0 and 1, while the sigmoid activation function produces a continuous range of outputs between 0 and 1.

Another difference between the two activation functions is their use in different types of neural networks.

The step activation function is typically used in feedforward neural networks, while the sigmoid activation function is typically used in recurrent neural networks.

**Step Function:** Step Function is one of the simplest kind of activation functions. In this, we consider a threshold value and if the value of net input say **y** is greater than the threshold then the neuron is activated. Mathematically,

**Sigmoid Function:** Sigmoid function is a widely used activation function. It is defined as: This is a smooth function and is continuously differentiable. The biggest advantage that it has over step and linear function is that it is non-linear. This is an incredibly cool feature of the sigmoid function. This essentially means that when I have multiple neurons having sigmoid function as their activation function – the output is non linear as well. The function ranges from 0-1 having an S shape.

**3.Single layer vs multi-layer perceptron**

**ANS:-**

**Single-layer perceptron**

One of the main advantages of using a single-layer perceptron is its simplicity and efficiency. It is easy to implement, train, and understand. It has a clear geometric interpretation as a hyperplane that separates two classes of data. It can also perform well on problems that are linearly separable, such as logical operations, linear regression, and binary classification.

**Multi-layer perceptron**

One of the main advantages of using a multi-layer perceptron is its versatility and universality. It can learn to approximate any continuous function, given enough hidden layers and neurons. It can also perform well on problems that are non-linearly separable, such as image recognition, natural language processing, and complex classification. It has more hidden layers and activation functions that can introduce non-linearity and flexibility to the model.

A single-layer feedforward neural network is a neural network that has only one layer of neurons that directly connects the input to the output. This type of network is also known as a perceptron. It takes the input, processes it, and produces an output in one forward pass.

On the other hand, a multilayer feedforward neural network, also known as a "deep neural network," has more than one layer of neurons between the input and output layers. These intermediate layers are also known as hidden layers. The input data is processed through the hidden layers in a series of forward passes to produce an output.

The main difference between a single-layer and a multilayer feedforward neural network is the number of layers. The single-layer perceptron can only learn linearly separable patterns, whereas the multilayer feedforward neural network can learn non-linear patterns. This is because the hidden layers in a multilayer network allow for more complex transformations of the input data.

In summary, a single-layer feedforward neural network is a simple neural network that has only one layer of neurons connecting the input to the output, whereas a multilayer feedforward neural network is a more complex neural network that has multiple layers of neurons, including hidden layers, between the input and output layers.