What Is Learning in ANN?

Basically, learning means to do and adapt the change in itself as and when there is a change in environment. ANN is a complex system or more precisely we can say that it is a complex adaptive system, which can change its internal structure based on the information passing through it.

Why Is It important?

Being a complex adaptive system, learning in ANN implies that a processing unit is capable of changing its input/output behavior due to the change in environment. The importance of learning in ANN increases because of the fixed activation function as well as the input/output vector, when a particular network is constructed. Now to change the input/output behavior, we need to adjust the weights.

Classification

It may be defined as the process of learning to distinguish the data of samples into different classes by finding common features between the samples of the same classes. For example, to perform training of ANN, we have some training samples with unique features, and to perform its testing we have some testing samples with other unique features. Classification is an example of supervised learning.

Neural Network Learning Rules

We know that, during ANN learning, to change the input/output behavior, we need to adjust the weights. Hence, a method is required with the help of which the weights can be modified. These methods are called Learning rules, which are simply algorithms or equations. Following are some learning rules for the neural network −

Hebbian Learning Rule

This rule, one of the oldest and simplest, was introduced by Donald Hebb in his book *The Organization of Behavior* in 1949. It is a kind of feed-forward, unsupervised learning.

Basic Concept − This rule is based on a proposal given by Hebb, who wrote −

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

From the above postulate, we can conclude that the connections between two neurons might be strengthened if the neurons fire at the same time and might weaken if they fire at different times.

Mathematical Formulation − According to Hebbian learning rule, following is the formula to increase the weight of connection at every time step.

$$\Delta w_{ji}(t)=\alpha x_i(t).y_j(t)\Delta wji(t)=\alpha xi(t).yj(t)$$

Here, $\Delta w_{ji}(t)\Delta wji(t)$ = increment by which the weight of connection increases at time step **t**

$\alpha\alpha$ = the positive and constant learning rate

$x_i(t)$xi(t) = the input value from pre-synaptic neuron at time step **t**

$y_i(t)$yi(t) = the output of pre-synaptic neuron at same time step **t**

Perceptron Learning Rule

This rule is an error correcting the supervised learning algorithm of single layer feedforward networks with linear activation function, introduced by Rosenblatt.

Basic Concept − As being supervised in nature, to calculate the error, there would be a comparison between the desired/target output and the actual output. If there is any difference found, then a change must be made to the weights of connection.

Mathematical Formulation − To explain its mathematical formulation, suppose we have 'n' number of finite input vectors, xnn, along with its desired/target output vector tnn, where n = 1 to N.

Now the output 'y' can be calculated, as explained earlier on the basis of the net input, and activation function being applied over that net input can be expressed as follows −

$$y=f(y_{in})=\{1,0,y_{in}>\theta y_{in}\leqslant\theta y=f(yin)=\{1,yin>\theta 0,yin\leqslant\theta$$

Where **θ** is threshold.

The updating of weight can be done in the following two cases −

Case I − when **t ≠ y**, then

$$w(new)=w(old)+txw(new)=w(old)+tx$$

Case II − when **t = y**, then

No change in weight

Delta Learning Rule Widrow−HoffRuleWidrow−HoffRule

It is introduced by Bernard Widrow and Marcian Hoff, also called Least Mean Square LMSLMS method, to minimize the error over all training patterns. It is kind of supervised learning algorithm with having continuous activation function.

Basic Concept − The base of this rule is gradient-descent approach, which continues forever. Delta rule updates the synaptic weights so as to minimize the net input to the output unit and the target value.

Mathematical Formulation − To update the synaptic weights, delta rule is given by

$$\Delta w_i=\alpha.x_i.e_j\Delta wi=\alpha.xi.ej$$

Here $\Delta w_i\Delta wi$ = weight change for i^{th} pattern;
$\alpha\alpha$ = the positive and constant learning rate;
x_ixi = the input value from pre-synaptic neuron;
e_jej = $(t-y_{in})(t-yin)$, the difference between the desired/target output and the actual output $y_{in}yin$

The above delta rule is for a single output unit only.

The updating of weight can be done in the following two cases −

Case-I − when **t ≠ y**, then

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

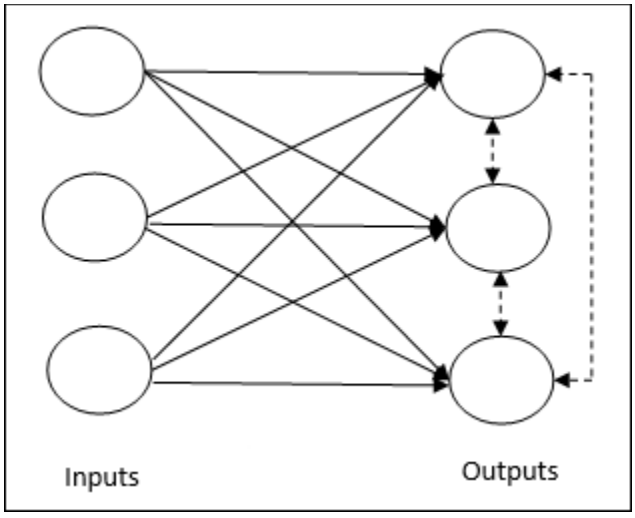
Case-II − when **t = y**, then

No change in weight

Competitive Learning Rule Winner−takes−allWinner−takes−all

It is concerned with unsupervised training in which the output nodes try to compete with each other to represent the input pattern. To understand this learning rule, we must understand the competitive network which is given as follows −

Basic Concept of Competitive Network − This network is just like a single layer feedforward network with feedback connection between outputs. The connections between outputs are inhibitory type, shown by dotted lines, which means the competitors never support themselves.



Basic Concept of Competitive Learning Rule − As said earlier, there will be a competition among the output nodes. Hence, the main concept is that during training, the output unit with the highest activation to a given input pattern, will be declared the winner. This rule is also called Winner-takes-all because only the winning neuron is updated and the rest of the neurons are left unchanged.

Mathematical formulation − Following are the three important factors for mathematical formulation of this learning rule −

- **Condition to be a winner** − Suppose if a neuron $y_k yk$ wants to be the winner then there would be the following condition −

$$y_k=\{10if v_k>v_j forall j,j\neq kotherwiseyk=\{1ifvk>vjforallj,j\neq k0otherwise$$

It means that if any neuron, say $y_k yk$, wants to win, then its induced local field $theoutputofsummationunittheoutputofsummationunit$, say $v_k vk$, must be the largest among all the other neurons in the network.

- **Condition of sum total of weight** − Another constraint over the competitive learning rule is, the sum total of weights to a particular output neuron is going to be 1. For example, if we consider neuron **k** then −

$$\sum_j w_{kj}=1forallk\sum jwkj=1forallk$$

- **Change of weight for winner** − If a neuron does not respond to the input pattern, then no learning takes place in that neuron. However, if a particular neuron wins, then the corresponding weights are adjusted as follows

$$\Delta w_{kj}=\{-\alpha(x_j-w_{kj}),0,\text{ifneuronkwinsifneuronklosses}\Delta wkj=\{-\alpha(xj-wkj),\text{ifneuronkwins}0,\text{ifneuronklosses}$$

Here αα is the learning rate.

This clearly shows that we are favoring the winning neuron by adjusting its weight and if there is a neuron loss, then we need not bother to re-adjust its weight.

Outstar Learning Rule

This rule, introduced by Grossberg, is concerned with supervised learning because the desired outputs are known. It is also called Grossberg learning.

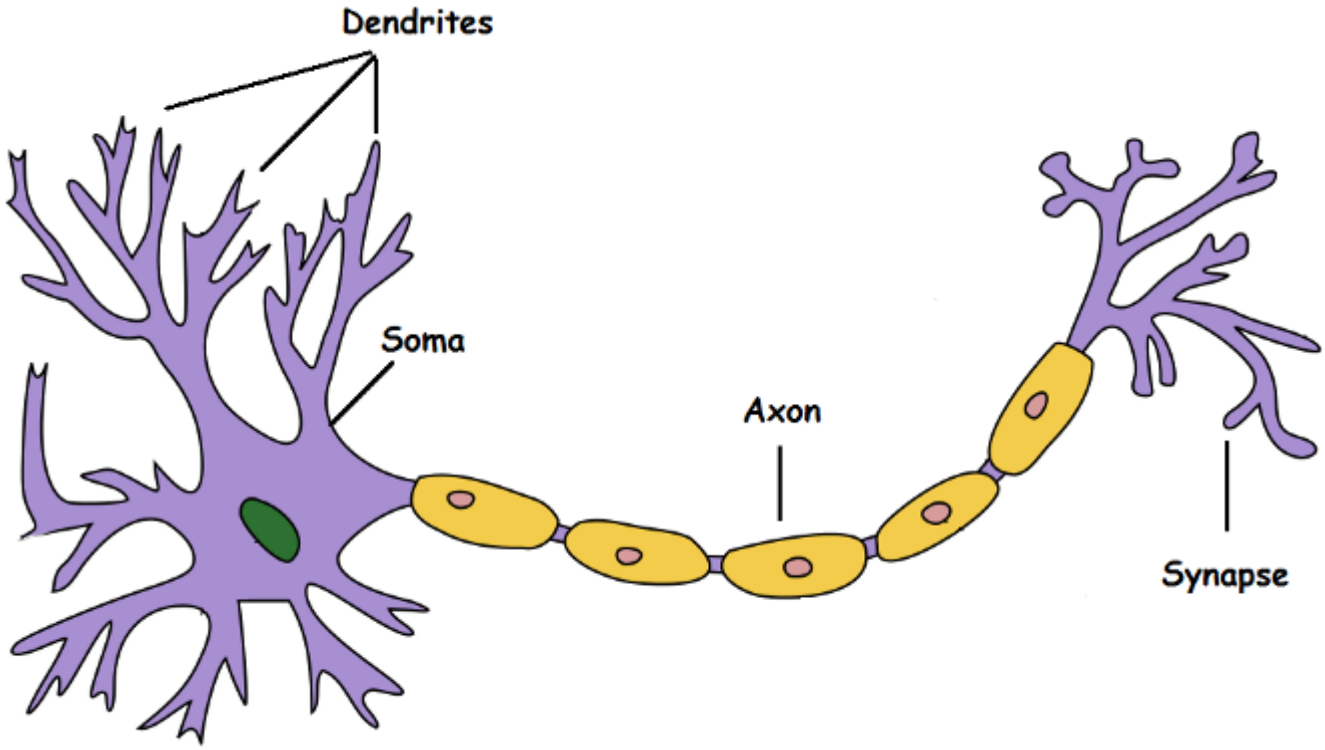
Basic Concept − This rule is applied over the neurons arranged in a layer. It is specially designed to produce a desired output **d** of the layer of **p** neurons.

Mathematical Formulation − The weight adjustments in this rule are computed as follows

$$\Delta w_j=\alpha(d-w_j)\Delta wj=\alpha(d-wj)$$

Here **d** is the desired neuron output and αα is the learning rate.

Biological Neurons: An Overly Simplified Illustration



A Biological Neuron

Dendrite: Receives signals from other neurons

Soma: Processes the information

Axon: Transmits the output of this neuron

Synapse: Point of connection to other neurons

Basically, a neuron takes an input signal (dendrite), processes it like the CPU (soma), passes the output through a cable like structure to other connected neurons (axon to synapse to other neuron's dendrite). Now, this might be biologically inaccurate as there is a lot more going on out there but on a higher level, this is what is going on with a neuron in our brain — takes an input, processes it, throws out an output.

Our sense organs interact with the outer world and send the visual and sound information to the neurons. Let's say you are watching Friends. Now the information your brain receives is taken in by the "laugh or not" set of neurons that will help you make a decision on whether to laugh or not. Each neuron gets fired/activated only when its respective criteria (more on this later) is met

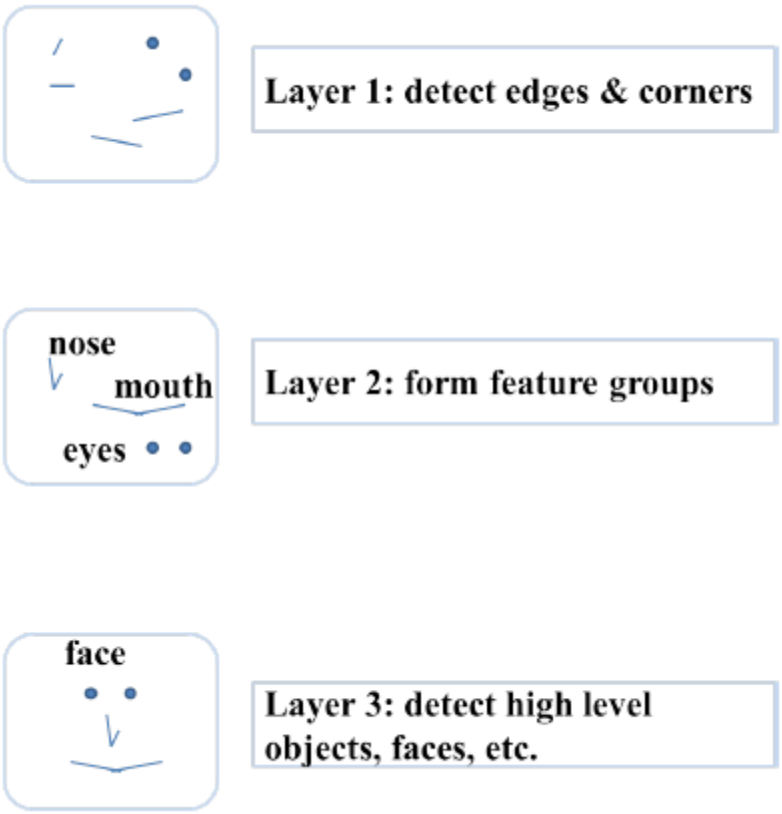
In reality, it is not just a couple of neurons which would do the decision making. There is a massively parallel interconnected network of 10^{11} neurons (100 billion) in our brain and their connections are not as simple as I showed you above.

Now the sense organs pass the information to the first/lowest layer of neurons to process it. And the output of the processes is passed on to the next layers in a hierarchical manner, some of the neurons will fire and some won't and this process goes on until it results in a final response — in this case, laughter.

This massively parallel network also ensures that there is a division of work. Each neuron only fires when its intended criteria is met i.e., a neuron may perform a certain role to a certain stimulus, as shown below.

Division of work

It is believed that neurons are arranged in a hierarchical fashion (however, many credible alternatives with experimental support are proposed by the scientists) and each layer has its own role and responsibility. To detect a face, the brain could be relying on the entire network and not on a single layer.



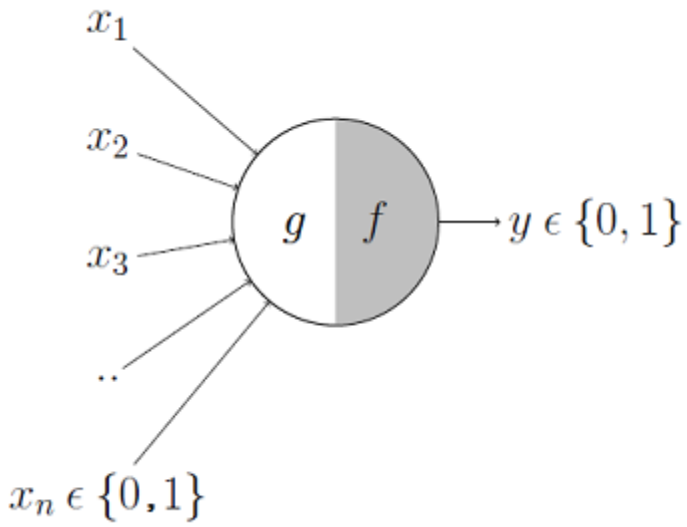
Sample illustration of hierarchical processing. Credits: Mitesh M. Khapra's lecture slides

Now that we have established how a biological neuron works, lets look at what McCulloch and Pitts had to offer.

Note: My understanding of how the brain works is very very very limited. The above illustrations are overly simplified.

McCulloch-Pitts Neuron

The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.



This is where it all began..

It may be divided into 2 parts. The first part, ***g*** takes an input (ahem dendrite ahem), performs an aggregation and based on the aggregated value the second part, ***f*** makes a decision.

Lets suppose that I want to predict my own decision, whether to watch a random football game or not on TV. The inputs are all boolean i.e., {0,1} and my output variable is also boolean {0: Will watch it, 1: Won't watch it}.

- So, ***x_1*** could be *isPremierLeagueOn* (I like Premier League more)
- ***x_2*** could be *isItAFriendlyGame* (I tend to care less about the friendlies)
- ***x_3*** could be *isNotHome* (Can't watch it when I'm running errands. Can I?)
- ***x_4*** could be *isManUnitedPlaying* (I am a big Man United fan. GGMU!) and so on.

These inputs can either be *excitatory* or *inhibitory*. Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs i.e., if ***x_3*** is 1 (not home) then my output will always be 0 i.e., the neuron will never fire, so ***x_3*** is an inhibitory input. Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together. Formally, this is what is going on:

$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

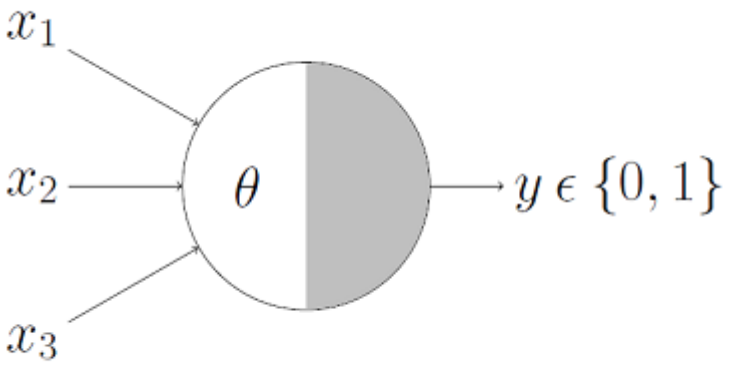
$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

We can see that ***g*(x)** is just doing a sum of the inputs — a simple aggregation. And ***theta*** here is called thresholding parameter. For example, if I always watch the game when the sum turns out to be 2 or more, the ***theta*** is 2 here. This is called the Thresholding Logic.

Boolean Functions Using M-P Neuron

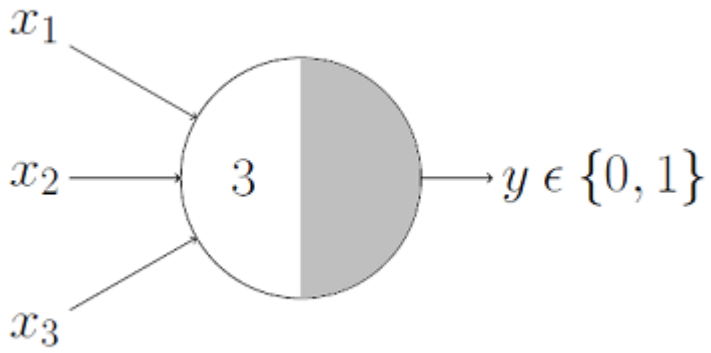
So far we have seen how the M-P neuron works. Now lets look at how this very neuron can be used to represent a few boolean functions. Mind you that our inputs are all boolean and the output is also boolean so essentially, the neuron is just trying to learn a boolean function. A lot of boolean decision problems can be cast into this, based on appropriate input variables— like whether to continue reading this post, whether to watch Friends after reading this post etc. can be represented by the M-P neuron.

M-P Neuron: A Concise Representation



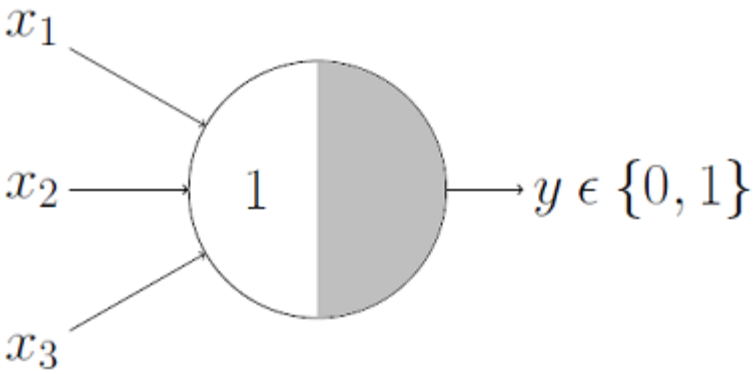
This representation just denotes that, for the boolean inputs ***x_1***, ***x_2*** and ***x_3*** if the ***g*(x)** i.e., **sum ≥ theta**, the neuron will fire otherwise, it won't.

AND Function



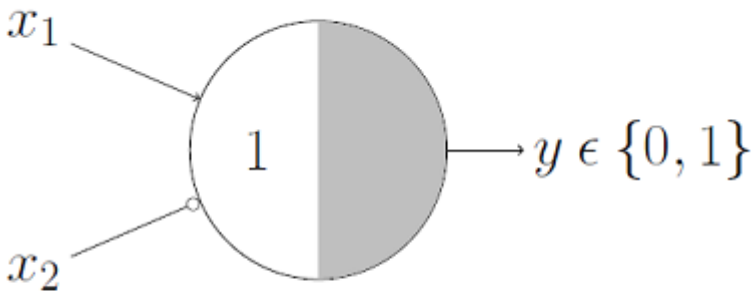
An AND function neuron would only fire when ALL the inputs are ON i.e., ***g*(x)** ≥ 3 here.

OR Function



I believe this is self explanatory as we know that an OR function neuron would fire if ANY of the inputs is ON i.e., ***g*(x)** ≥ 1 here.

A Function With An Inhibitory Input



$x_1\ AND\ !x_2$*

Now this might look like a tricky one but it's really not. Here, we have an inhibitory input i.e., ***x_2*** so whenever ***x_2*** is 1, the output will be 0. Keeping that in mind, we know that ***x_1 AND !x_2*** would output 1 only when ***x_1*** is 1 and ***x_2*** is 0 so it is obvious that the threshold parameter should be 1.

Lets verify that, the ***g(x)*** i.e., ***x_1*** + ***x_2*** would be ≥ 1 in only 3 cases:

Case 1: when ***x_1*** is 1 and ***x_2*** is 0
Case 2: when ***x_1*** is 1 and ***x_2*** is 1
Case 3: when ***x_1*** is 0 and ***x_2*** is 1

But in both Case 2 and Case 3, we know that the output will be 0 because ***x_2*** is 1 in both of them, thanks to the inhibition. And we also know that ***x_1 AND !x_2*** would output 1 for Case 1 (above) so our thresholding parameter holds good for the given function.

Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is a collection of connected units (nodes). These connected units are known as artificial neurons. These units closely resemble the original neurons of a human brain. Every node is built with a set of inputs, weights, and a bias value. Weights of the neural network are held within the hidden layers.

Weights and biases are learning parameters of machine learning models, they are modified for training the neural networks.

Applications of Neural Networks

Neural Networks are regulating some key sectors including finance, healthcare, and automotive. As these artificial neurons function in a way similar to the human brain. They can be used for image recognition, character recognition and stock market predictions. *Let's understand the diverse applications of neural networks*

1. Facial Recognition

Facial Recognition Systems are serving as robust systems of surveillance. Recognition Systems matches the human face and compares it with the digital images. They are used in offices for selective entries. The systems thus authenticate a human face and match it up with the list of IDs that are present in its database.

Convolutional Neural Networks (CNN) are used for **facial recognition and image processing.** Large number of pictures are fed into the database for training a neural network. The collected images are further processed for training.

Sampling layers in CNN are used for proper evaluations. Models are optimized for accurate recognition results.

)

2. Stock Market Prediction

Investments are subject to market risks. It is nearly impossible to predict the upcoming changes in the highly volatile stock market. The forever changing bullish and bearish phases were unpredictable before the advent of neural networks. But well what changed it all? Neural Networks of course…

To make a successful stock prediction in real time a **Multilayer Perceptron MLP** *(class of feedforward artificial intelligence algorithm)* is employed. MLP comprises multiple layers of nodes, each of these layers is fully connected to the succeeding nodes. Stock's past performances, annual returns, and non profit ratios are considered for building the MLP model.

3. Social Media

No matter how cliche it may sound, social media has altered the normal boring course of life. Artificial Neural Networks are used to study the behaviours of social media users. Data shared everyday via virtual conversations is tacked up and analyzed for competitive analysis.

Neural networks duplicate the behaviours of social media users. Post analysis of individuals' behaviours via social media networks the data can be linked to people's spending habits. **Multilayer Perceptron ANN** is used to mine data from social media applications.

MLP forecasts social media trends, it uses different training methods like **Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Squared Error (MSE).** MLP takes into consideration several factors like user's favourite instagram pages, bookmarked choices etc. These factors are considered as inputs for training the MLP model.

In the ever changing dynamics of social media applications, artificial neural networks can definitely work as the best fit model for user data analysis.

4. Aerospace

Aerospace Engineering is an expansive term that covers developments in spacecraft and aircraft. Fault diagnosis, high performance auto piloting, securing the aircraft control systems, and modeling key dynamic simulations are some of the key areas that neural networks have taken over. Time delay Neural networks can be employed for modelling non linear time dynamic systems.

Time Delay Neural Networks are used for **position independent feature recognition.** The algorithm thus built based on time delay neural networks can recognize patterns. *(Recognizing patterns are automatically built by neural networks by copying the original data from feature units).*

Other than this TNN are also used to provide stronger dynamics to the NN models. As passenger safety is of utmost importance inside an aircraft, algorithms built using the neural network systems ensures the accuracy in the autopilot system. As most of the autopilot functions are automated, it is important to ensure a way that maximizes the security.

Applications of neural networks

5. Defence

Defence is the backbone of every country. Every country's state in the international domain is assessed by its military operations. Neural Networks also shape the defence operations of technologically advanced countries. The United States of America, Britain, and Japan are some countries that use artificial neural networks for developing an active defence strategy.

Neural networks are used in logistics, armed attack analysis, and for object location. They are also used in air patrols, maritime patrol, and for controlling automated drones. The defence sector is getting the much needed kick of artificial intelligence to scale up its technologies.

Convolutional Neural Networks(CNN), are employed for determining the presence of underwater mines. Underwater mines are the underpass that serve as an illegal commute route between two countries. **Unmanned Airborne Vehicle (UAV),** and **Unmanned Undersea Vehicle (UUV)** these autonomous sea vehicles use convolutional neural networks for the image processing.

Convolutional layers form the basis of Convolutional Neural Networks. These layers use different filters for differentiating between images. Layers also have bigger filters that filter channels for image extraction.

6. Healthcare

The age old saying goes like "Health is Wealth". Modern day individuals are leveraging the advantages of technology in the healthcare sector. **Convolutional Neural Networks** are actively employed in the healthcare industry for **X ray detection, CT Scan** and **ultrasound.**

As CNN is used in image processing, the medical imaging data retrieved from aforementioned tests is analyzed and assessed based on neural network models. **Recurrent Neural Network (RNN)** is also being employed for the development of voice recognition systems.

Voice recognition systems are used these days to keep track of the patient's data. Researchers are also employing **Generative Neural Networks** for drug discovery. Matching different categories of drugs is a hefty task, but generative neural networks have broken down the hefty task of drug discovery. They can be used for combining different elements which forms the basis of drug discovery.

7. Signature Verification and Handwriting Analysis

Signature Verification , as the self explanatory term goes, is used for verifying an individual's signature. Banks, and other financial institutions use signature verification to cross check the identity of an individual.

Usually a signature verification software is used to examine the signatures. As cases of forgery are pretty common in financial institutions, signature verification is an important factor that seeks to closely examine the authenticity of signed documents.

Artificial Neural Networks are used for **verifying the signatures.** ANN are trained to recognize the difference between real and forged signatures. ANNs can be used for the verification of both offline and online signatures.

For training an ANN model, varied datasets are fed in the database. The data thus fed help the ANN model to differentiate. **ANN model employs image processing** for extraction of features.

Handwriting analysis plays an integral role in forensics. The analysis is further used to evaluate the variations in two handwritten documents. The process of spilling words on a blank sheet is also used for behavioural analysis. **Convolutional Neural Networks (CNN)** are used for handwriting analysis and handwriting verification.

8. Weather Forecasting

The forecasts done by the meteorological department were never accurate before artificial intelligence came into force. Weather Forecasting is primarily undertaken to anticipate the upcoming weather conditions beforehand. In the modern era, weather forecasts are even used to predict the possibilities of natural disasters.

Multilayer Perceptron (MLP), Convolutional Neural Network (CNN) and Recurrent Neural Networks (RNN) are used for weather forecasting. Traditional ANN multilayer models can also be used to predict climatic conditions 15 days in advance. A combination of different types of neural network architecture can be used to predict air temperatures.

Various inputs like air temperature, relative humidity, wind speed and solar radiations were considered for training neural network based models. **Combination models (MLP+CNN), (CNN+RNN)** usually works better in the case of weather forecasting.

Summing Up

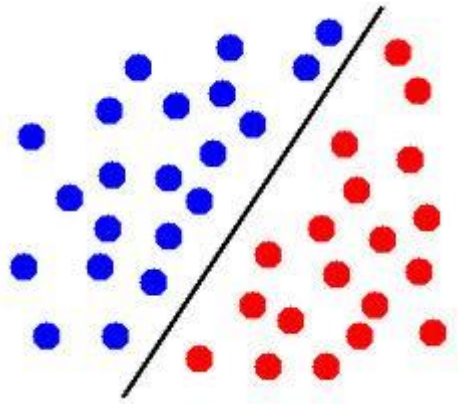
Neural Networks have a myriad of applications, from facial recognition to weather forecasting the interconnected layers (human brain's replica), can do a lot of things with some simple inputs. ANN algorithms have simplified the assessment and modified the traditional algorithms. With humanoid robots like Grace on its way the world can expect some sci- fi movies turning into reality pretty soon!

LINEAR SEPARABLITY

This means that there is a hyperplane (which splits your input space into two half-spaces) such that all points of the first class are in one half-space and those of the second class are in the other half-space.

In two dimensions, that means that there is a line which separates points of one class from points of the other class.

EDIT: for example, in this image, if blue circles represent points from one class and red circles represent points from the other class, then these points are linearly separable.



In three dimensions, it means that there is a plane which separates points of one class from points of the other class.

In higher dimensions, it's similar: there must exist a hyperplane which separates the two sets of points.

What is the Perceptron model in Machine Learning?

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

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

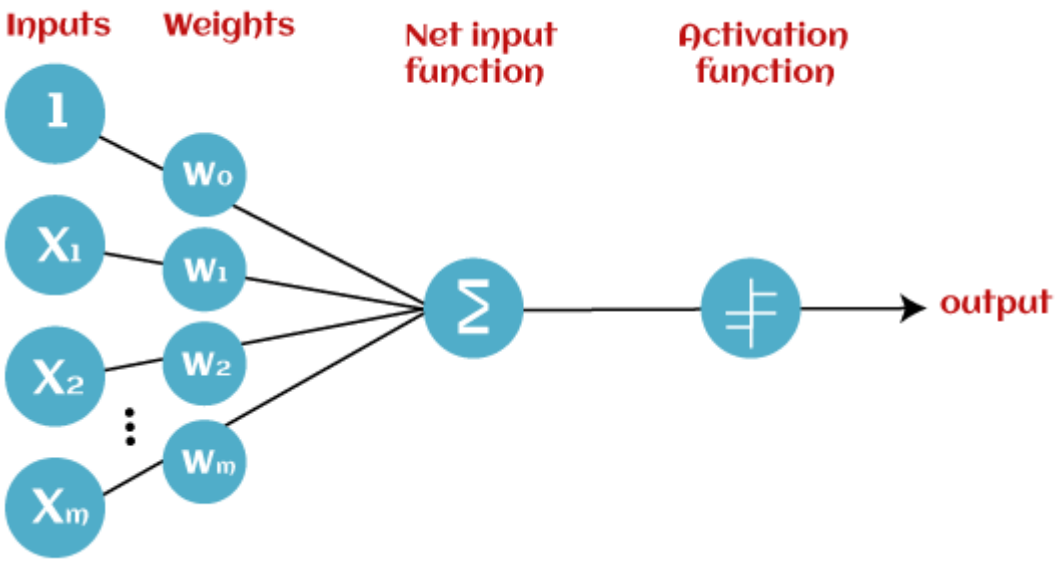
What is Binary classifier in Machine Learning?

In Machine Learning, binary classifiers are defined as the function that helps in deciding whether input data can be represented as vectors of numbers and belongs to some specific class.

Binary classifiers can be considered as linear classifiers. In simple words, we can understand it as a ***classification algorithm that can predict linear predictor function in terms of weight and feature vectors.***

Basic Components of Perceptron

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



- **Input Nodes or Input Layer:**

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

- **Wight and Bias:**

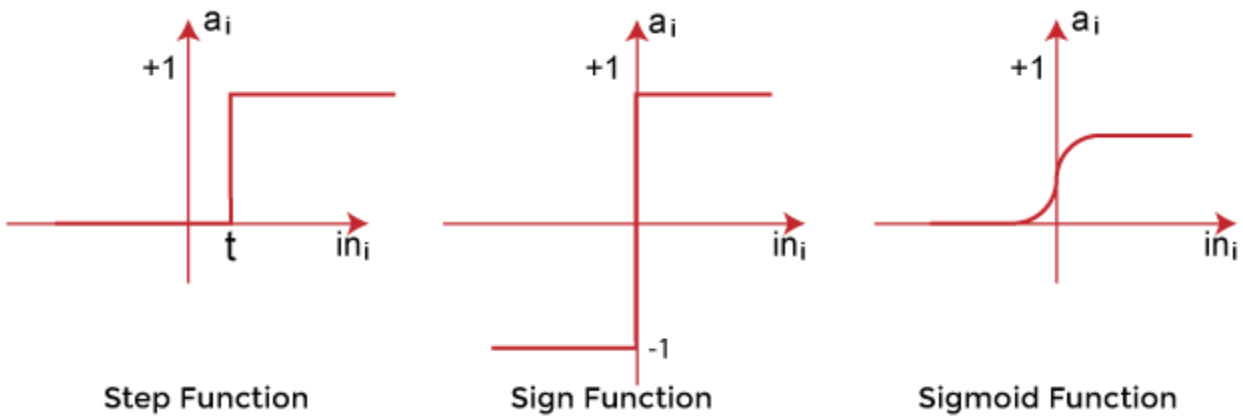
Weight parameter represents the strength of the connection between units. This is another most important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept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

- **Activation Function:**

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

Types of Activation functions:

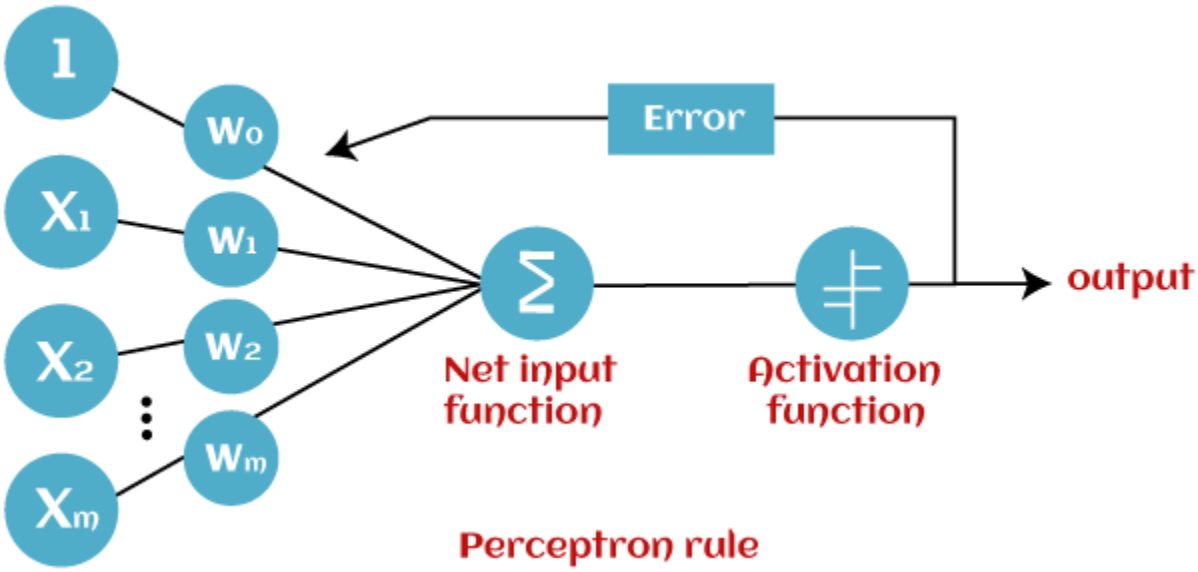
- Sign function
- Step function, and
- Sigmoid function



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

How does Perceptron work?

In Machine Learning, Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function. The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum. Then this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the **step function** and is represented by **'f'**.



This step function or Activation function plays a vital role in ensuring that output is mapped between required values (0,1) or (-1,1). It is important to note that the weight of input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

Perceptron model works in two important steps as follows:

Step-1

In the first step first, multiply all input values with corresponding weight values and then add them to determine the weighted sum. Mathematically, we can calculate the weighted sum as follows:

∑wi*xi = x1*w1 + x2*w2 +...wn*xn

Add a special term called **bias 'b'** to this weighted sum to improve the model's performance.

∑wi*xi + b

Step-2

In the second step, an activation function is applied with the above-mentioned weighted sum, which gives us output either in binary form or a continuous value as follows:

Y = f(∑wi*xi + b)

Types of Perceptron Models

Based on the layers, Perceptron models are divided into two types. These are as follows:

1. Single-layer Perceptron Model
2. Multi-layer Perceptron model

Single Layer Perceptron Model:

This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.

If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

"*Single-layer perceptron can learn only linearly separable patterns.*"

Multi-Layered Perceptron Model:

Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.

The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
- **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.

Advantages of Multi-Layer Perceptron:

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron:

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of the training.