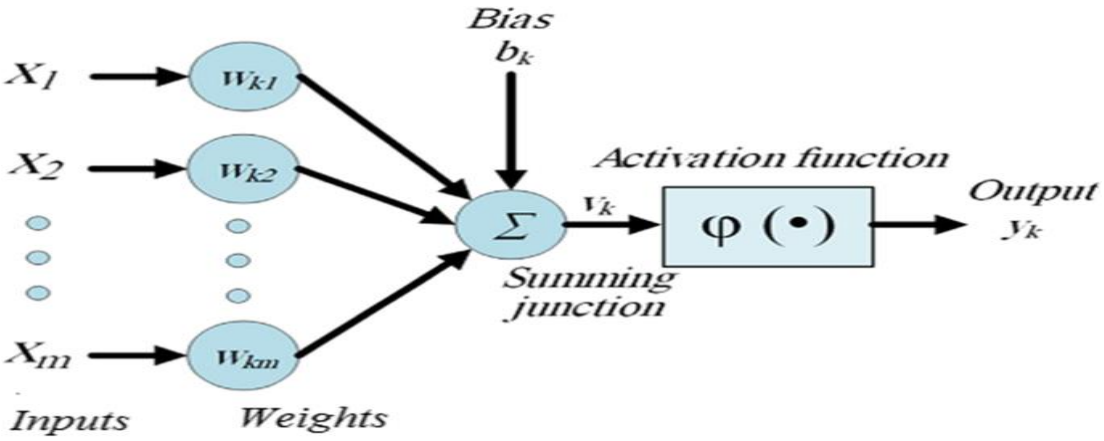
Soft computing

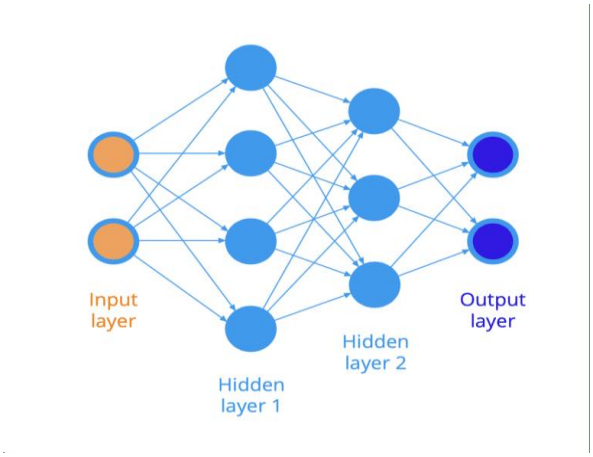
Unit 2

NEURAL NETWORK

- Neural networks reflect the behavior of the human brain, allowing computer programs to recognize patterns and solve common problems in the fields of AI, machine learning, and deep learning.
- A neural network is a network or circuit of biological neurons, or, in a modern sense, an artificial neural network composed of artificial neurons or nodes. Thus, a neural network is either a biological neural network, made up of biological neurons, or an artificial neural network, used for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled in artificial neural networks as weights between nodes.



- Artificial Neural Network which is an architecture of a large number of interconnected elements called neurons.
- These neurons process the input received to give the desired output. The nodes or neurons are linked by inputs, connection weights, and activation functions.
- The main characteristic of a neural network is its ability to learn. The neural networks train themselves with known examples. Once the network gets trained, it can be used for solving the unknown values of the problem.



- An ANN consists of 3 parts i.e. input, hidden layer, and output layer. There is a single input layer and output layer while there may be no hidden layer or 1 or more hidden layers that may be present in the network. Based on this structure the ANN is classified into a single layer, multilayer, feed-forward, or recurrent networks.

Important ANN Terminology

- 1) Weights: In an ANN, each neuron is connected to the other neurons through connection links. These links carry a weight. The weight has information about the input signal to the neuron. The weights and input signal are used to get an output. The weights can be denoted in a matrix form that is also called a Connection matrix.
- Each neuron is connected to every other neuron of the next layer through connection weights. Hence, if there are "n" nodes and each node has "m" weights, then the weight matrix will be:

$$\text{Weight matrix= W} = \begin{bmatrix} W1 \\ W2 \\ W3 \\ . \\ . \\ . \\ Wn \end{bmatrix} = \begin{bmatrix} w11 & w12 & \dots & w1m \\ w21 & w22 & \dots & w2m \\ w31 & w32 & \dots & w3m \\ . & . & \dots & .. \\ . & . & \dots & .... \\ . & . & \dots & .. \\ wn1 & wn2 & \dots & wnm \end{bmatrix}$$

W1 represents the weight vector starting from node 1. W11 represents the weight vector from the 1st node of the preceding layer to the 1st node of the next layer

2) Bias: The bias is added to the network by adding an input element x (b) = 1 into the input vector. The bias also carries a weight denoted by w (b).

- The bias plays an important role in calculating the output of the neuron. The bias can either be positive or negative. A positive bias increases the net input weight while the negative bias reduces the net input.

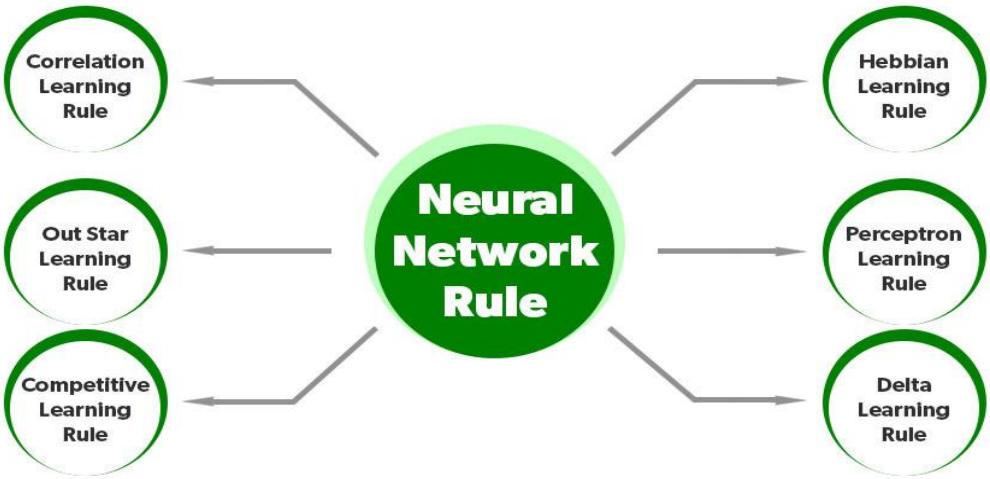
The new input will become:
X= {1, X1, X2, X3... Xj ....Xn}
The output neuron Y is calculated as:
$Y = \sum x * w = x(0)*w(0)(j) + x(1)*w(1)(j) + x(2)*w(2)(j) + \ldots\ldots + x(n)*w(n)(j)$
$= w(0)(j) + \sum x * w$
$Y = w(b) + \sum x * w$

3) Threshold: A threshold value is used in the activation function. The net input is compared with the threshold to get the output. In NN, the activation function is defined based on the threshold value and output is calculated.

4)Learning Rate: It is denoted by alpha. The learning rate ranges from 0 to 1. It is used for weight adjustment during the learning process of NN.

LEARNING RULE

- Learning rule enhances the Artificial Neural Network's performance by applying these rules over the network. Thus learning rule updates the weights and bias levels of a network when certain conditions are met in the training process. it is a crucial part of the development of the Neural Network.



1. Hebbian Learning Rule

- Donald Hebb developed it in 1949 as an unsupervised learning algorithm in the neural network. We can use it to improve the weights of nodes of a network. The following phenomenon occurs when
- If two neighbor neurons are operating in the same phase at the same period of time, then the weight between these neurons should increase.
- For neurons operating in the opposite phase, the weight between them should decrease.
- If there is no signal correlation, the weight does not change, the sign of the weight between two nodes depends on the sign of the input between those nodes
- When inputs of both the nodes are either positive or negative, it results in a strong positive weight.
- If the input of one node is positive and negative for the other, a strong negative weight is present.
- Mathematical Formulation:
- $\delta w = \alpha x_i y$ where δw=change in weight, α is the learning rate. x_i the input vector,y the output.

2.Perceptron Learning Rule

- It was introduced by Rosenblatt. It is an error-correcting rule of a single-layer feedforward network. it is supervised in nature and calculates the error between the desired and actual output and if the output is present then only adjustments of weight are done.
- Computed as follows:

Assume (x1,x2,x3.....xn) –>set of input vectors

and (w1,w2,w3.....wn) –>set of weights

y=actual output

wo=initial weight

wnew=new weight

δw=change in weight

α=learning rate

actual output(y)=wixi

learning signal(ej)=ti-y (difference between desired and actual output)

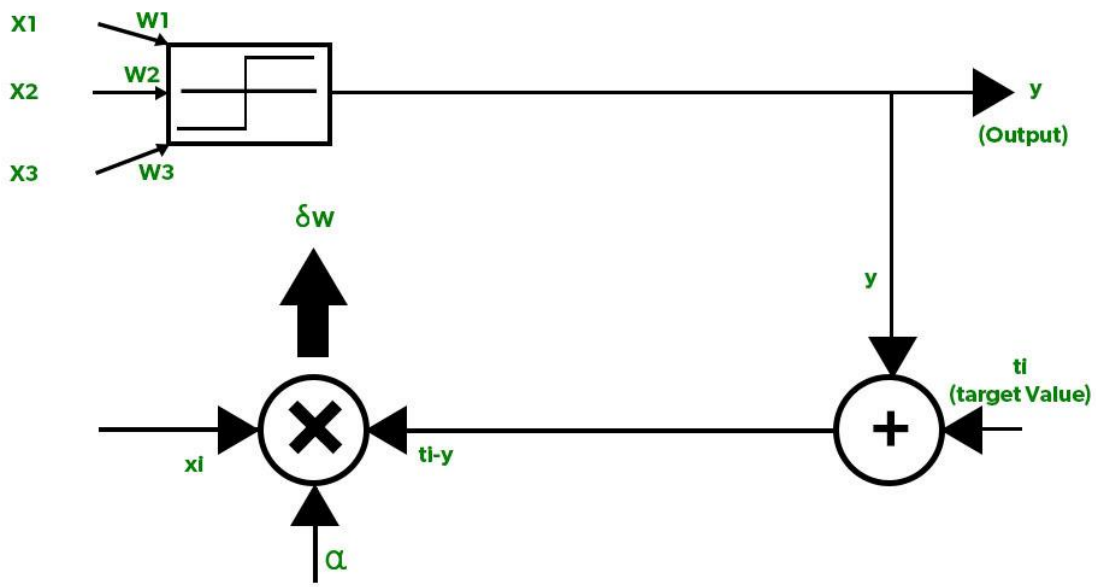
δw=αxiej

wnew=wo+δw

Now, the output can be calculated on the basis of the input and the activation function applied over the net input and can be expressed as:

y=1, if net input>=θ

y=0, if net input<θ



2. Delta Learning Rule

- It was developed by Bernard Widrow and Marcian Hoff and It depends on supervised learning and has a continuous activation function. It is also known as the Least Mean Square method and it minimizes error over all the training patterns
- It is based on a gradient descent approach which continues forever. It states that the modification in the weight of a node is equal to the product of the error and the input where the error is the difference between desired and actual output.
- Delta rule refers to the comparison of actual output with a target output, the technology tries to discover the match, and the program makes changes. The actual execution of the Delta rule will fluctuate as per the network and its composition

Computed as follows:

Assume (x1,x2,x3.....xn) –>set of input vectors

and (w1,w2,w3.....wn) –>set of weights

y=actual output

wo=initial weight

wnew=new weight

δw=change in weight

Error= ti-y

Learning signal(ej)=(ti-y)y'

y=f(net input)= ∫wixi

δw=αxiej=αxi(ti-y)y'

wnew=wo+δw

The updating of weights can only be done if there is a difference between the target and actual output(i.e., error) present:

---

case I: when t=y

then there is no change in weight

case II: else

wnew=wo+δw

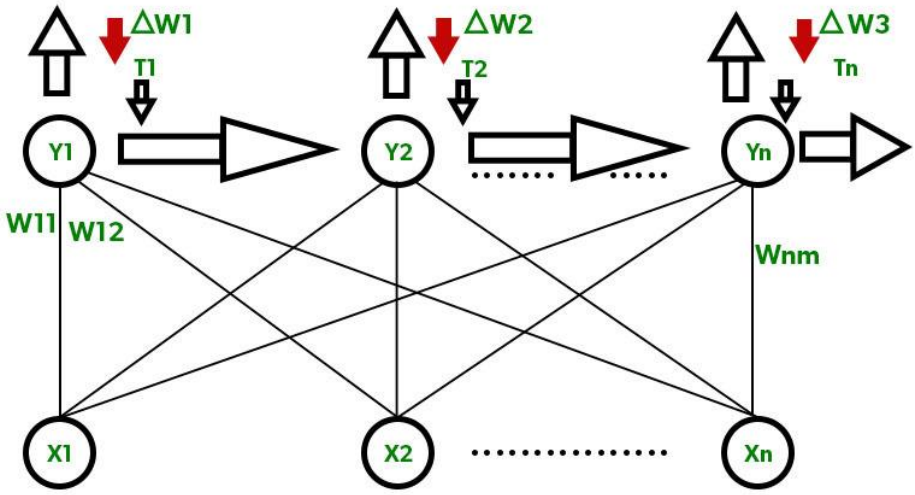
3. Correlation Learning Rule
    - The correlation learning rule follows the same similar principle as the Hebbian learning rule,i.e., If two neighbor neurons are operating in the same phase at the same period of time, then the weight between these neurons should be more positive. For neurons operating in the opposite phase, the weight between them should be more negative but unlike the Hebbian rule, the correlation rule is supervised in nature here, the targeted response is used for the calculation of the change in weight.

$\delta w = \alpha x i t j$

where δw=change in weight,α=learning rate,xi=set of the input vector, and tj=target value

5. Out Star Learning Rule

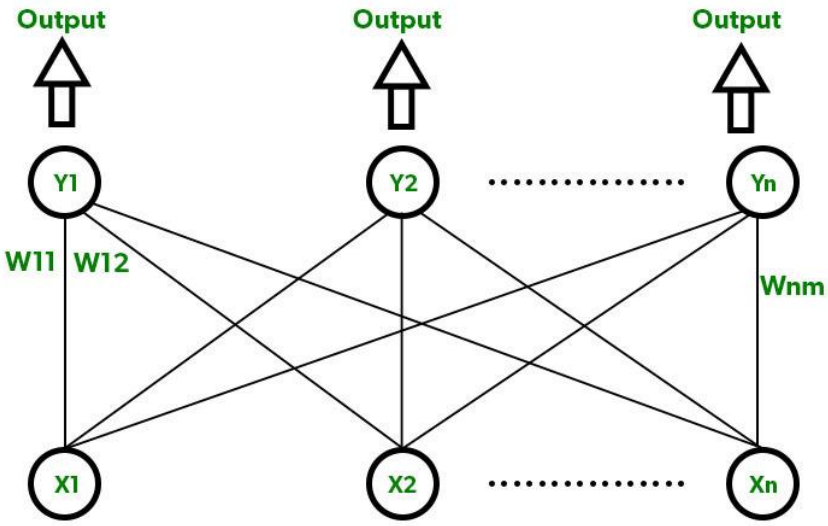
It was introduced by Grossberg and is a supervised training procedure.



Out Star Learning Rule is implemented when nodes in a network are arranged in a layer. Here the weights linked to a particular node should be equal to the targeted outputs for the nodes connected through those same weights. Weight change is thus calculated as=$\delta w = \alpha(t-y)$

Where α=learning rate, y=actual output, and t=desired output for n layer nodes.

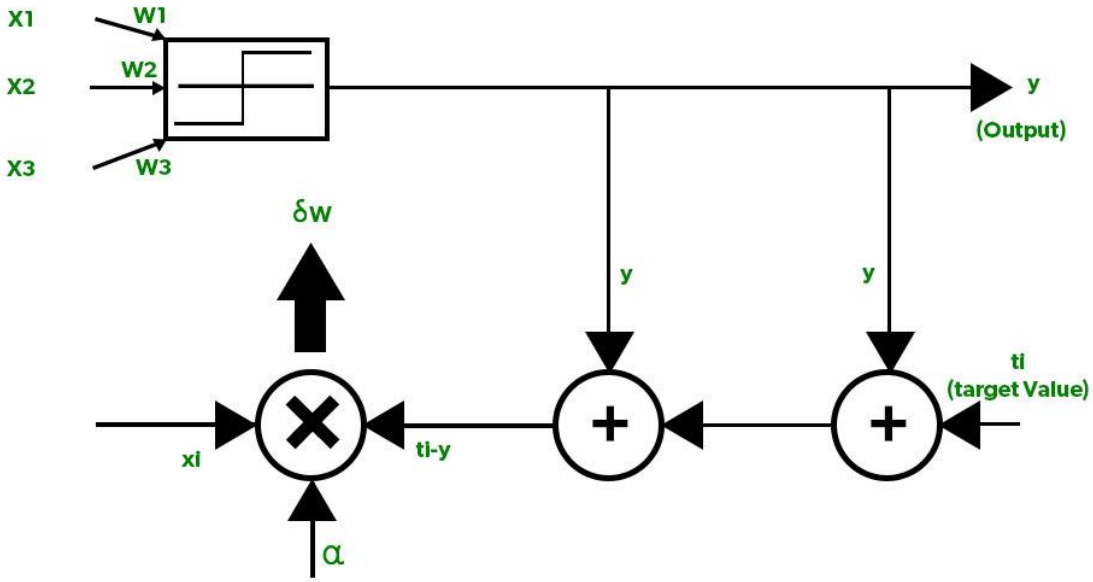
6. Competitive Learning Rule



It is also known as the Winner-takes-All rule and is unsupervised in nature. Here all the output nodes try to compete with each other to represent the input pattern and the winner is declared according to the node having the most outputs and is given the output 1 while the rest are given 0.

There are a set of neurons with arbitrarily distributed weights and the activation function is applied to a subset of neurons. Only one neuron is active at a time. Only the winner has updated weights, the rest remain unchanged.

Last Updated : 26 Oct, 2022



ACTIVATION FUNCTION

In artificial neural networks, an activation function is one that outputs a smaller value for tiny inputs and a higher value if its inputs are greater than a threshold. An activation function "fires" if the inputs are big enough; otherwise, nothing happens. An activation function, then, is a gate that verifies how an incoming value is higher than a threshold value.

Because they introduce non-linearities in neural networks and enable the neural networks can learn powerful operations, activation functions are helpful. A feedforward neural network might be refactored into a straightforward linear function or matrix transformation on to its input if indeed the activation functions were taken out.

By generating a weighted total and then including bias with it, the activation function determines whether a neuron should be turned on. The activation function seeks to boost a neuron's output's nonlinearity.

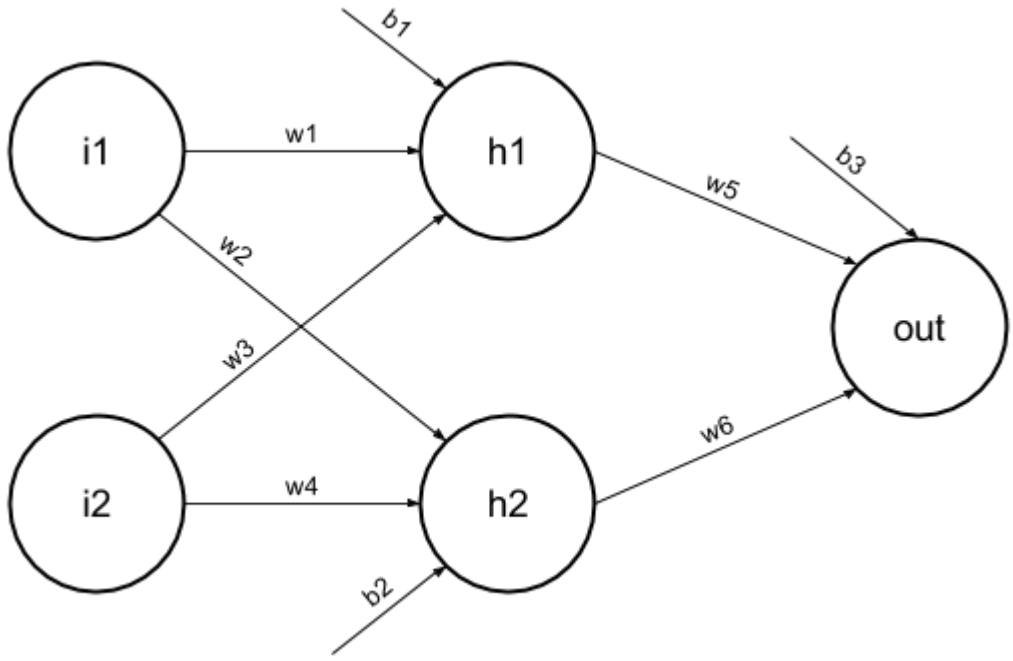
Explanation: As we are aware, neurons in neural networks operate in accordance with weight, bias, and their corresponding activation functions. Based on the mistake, the values of the neurons inside a neural network would be modified. This process is known as back-propagation. Back-propagation is made possible by activation functions since they provide the gradients and error required to change the biases and weights.

Why do we need Non-linear activation function?

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

Mathematical proof

Suppose we have a Neural net like this :-



Elements of the diagram are as follows:

Hidden layer i.e. layer 1:

$$z(1) = W(1)X + b(1)\ a(1)$$

Here,

- *z(1) is the vectorized output of layer 1*
- *W(1) be the vectorized weights assigned to neurons of hidden layer i.e. w1, w2, w3 and w4*
- *X be the vectorized input features i.e. i1 and i2*
- *b is the vectorized bias assigned to neurons in hidden layer i.e. b1 and b2*
- *a(1) is the vectorized form of any linear function.*

*(**Note:** We are not considering activation function here)*

Layer 2 i.e. output layer :-

***Note :** Input for layer 2 is output from layer 1*

$$z(2) = W(2)a(1) + b(2)$$

$$a(2) = z(2)$$

Calculation at Output layer

$$z(2) = (W(2) * [W(1)X + b(1)]) + b(2)$$

$$z(2) = [W(2) * W(1)] * X + [W(2)*b(1) + b(2)]$$

Let,

$$[W(2) * W(1)] = W$$

$$[W(2)*b(1) + b(2)] = b$$

Final output : $z(2) = W*X + b$

which is again a linear function

This observation results again in a linear function even after applying a hidden layer, hence we can conclude that, doesn't matter how many hidden layer we attach in neural net, all layers will behave same way because ***the composition of two linear function is a linear function itself***. Neuron can not learn with just a linear function

attached to it. A non-linear activation function will let it learn as per the difference w.r.t error. **Hence we need an activation function.**

The Activation Functions can be basically divided into 2 types-

1. Linear Activation Function
2. Non-linear Activation Functions

Linear or Identity Activation Function

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

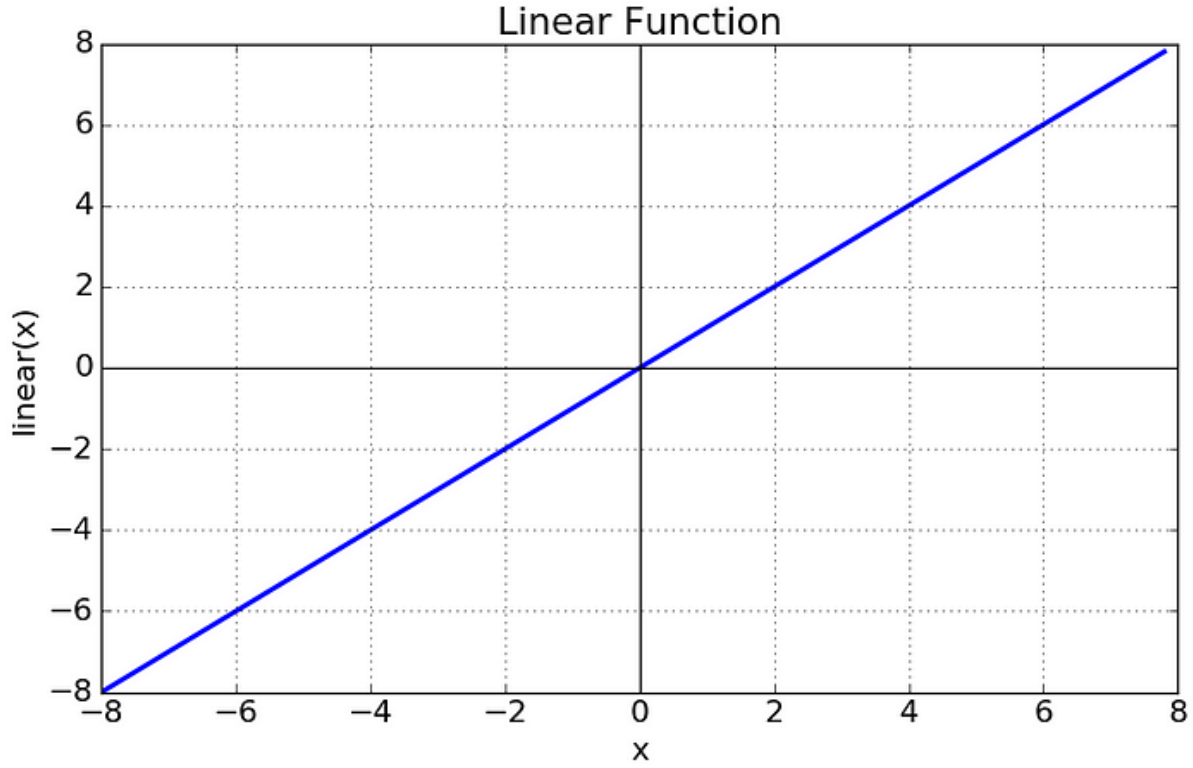


Fig: Linear Activation Function

Equation : f(x) = x

Range : (-infinity to infinity)

It doesn’t help with the complexity or various parameters of usual data that is fed to the neural networks.

Non-linear Activation Function

1. Sigmoid or Logistic Activation Function

The Sigmoid Function curve looks like a S-shape.

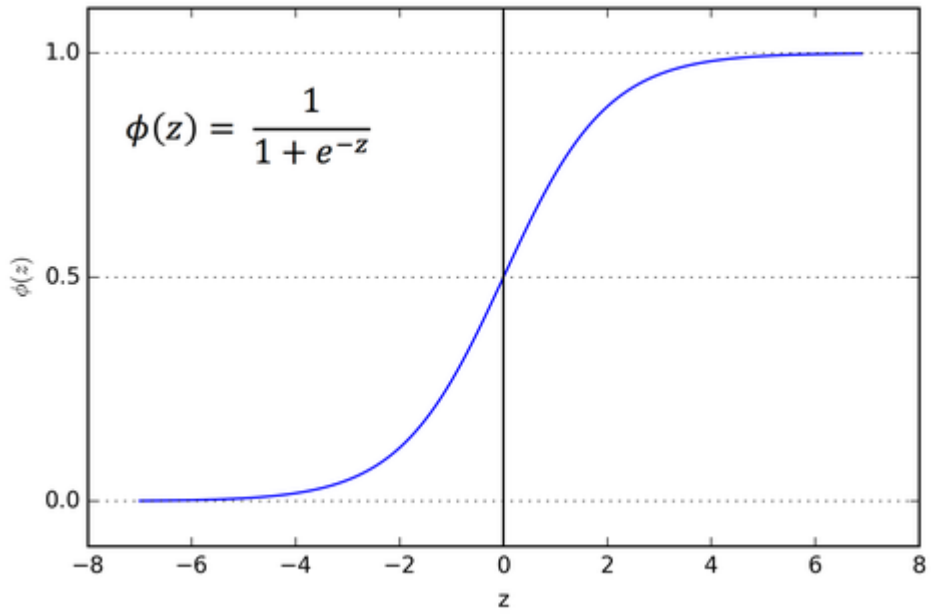


Fig: Sigmoid Function

The main reason why we use sigmoid function is because it exists between **(0 to 1).** Therefore, it is especially used for models where we have to **predict the probability** as an output.Since probability of anything exists only between the range of **0 and 1,** sigmoid is the right choice.

The function is **differentiable**.That means, we can find the slope of the sigmoid curve at any two points.

The function is **monotonic** but function’s derivative is not.

The logistic sigmoid function can cause a neural network to get stuck at the training time.

The **softmax function** is a more generalized logistic activation function which is used for multiclass classification.

2. Tanh or hyperbolic tangent Activation Function

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped).

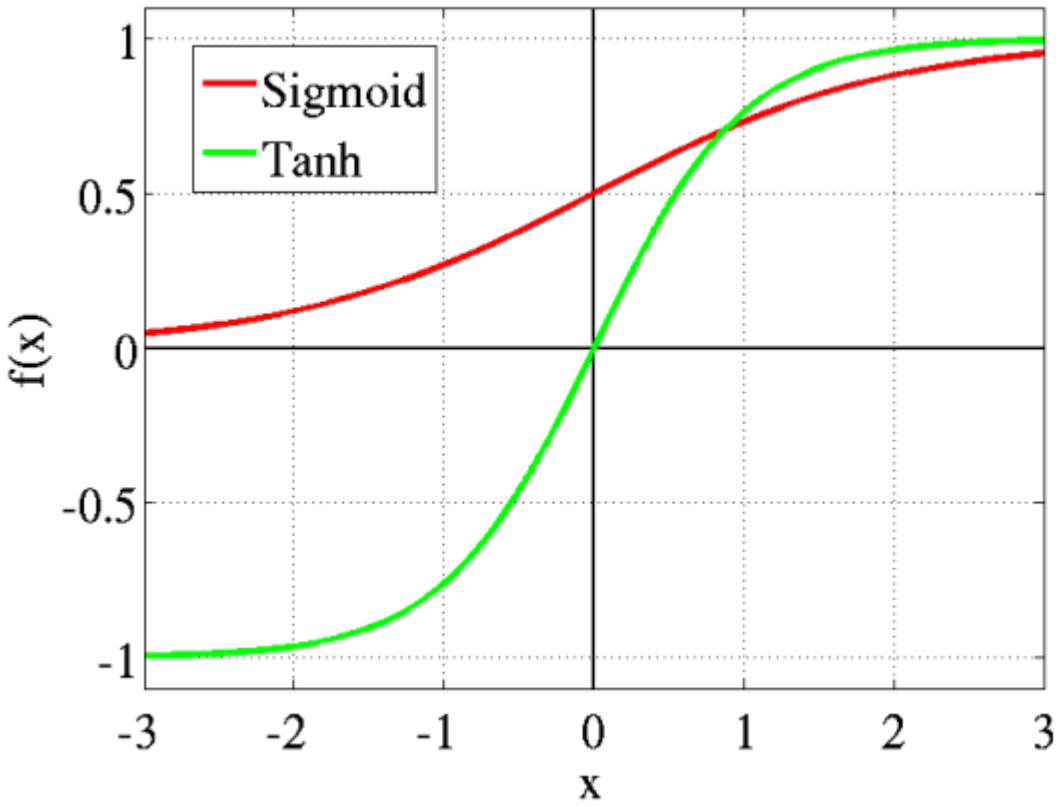


Fig: tanh v/s Logistic Sigmoid

The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

The function is **differentiable**.

The function is **monotonic** while its **derivative is not monotonic**.

The tanh function is mainly used classification between two classes.

Both tanh and logistic sigmoid activation functions are used in feed-forward nets.

3. ReLU (Rectified Linear Unit) Activation Function

The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning.

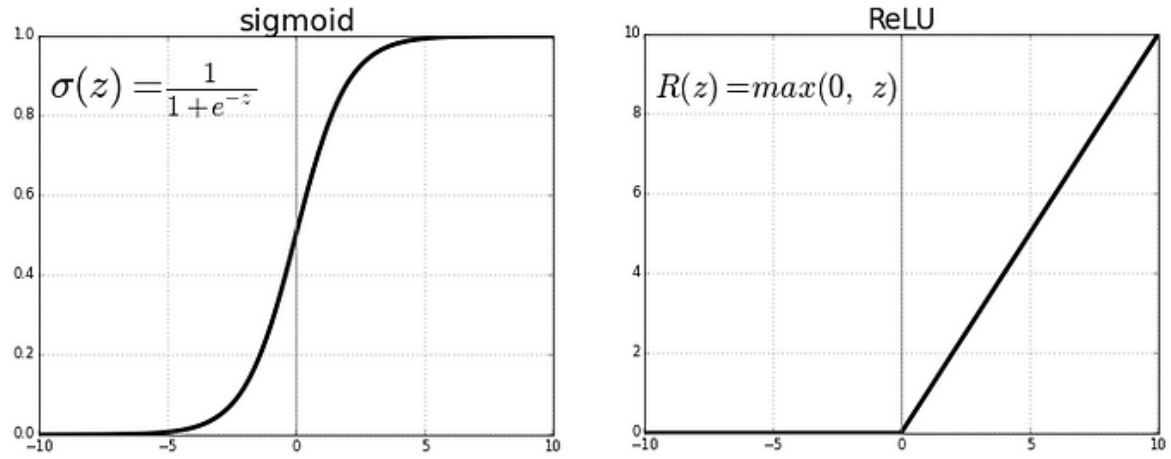


Fig: ReLU v/s Logistic Sigmoid

As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

Range: [0 to infinity)

The function and its derivative both are monotonic.

But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly.

That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

4. Leaky ReLU

It is an attempt to solve the dying ReLU problem

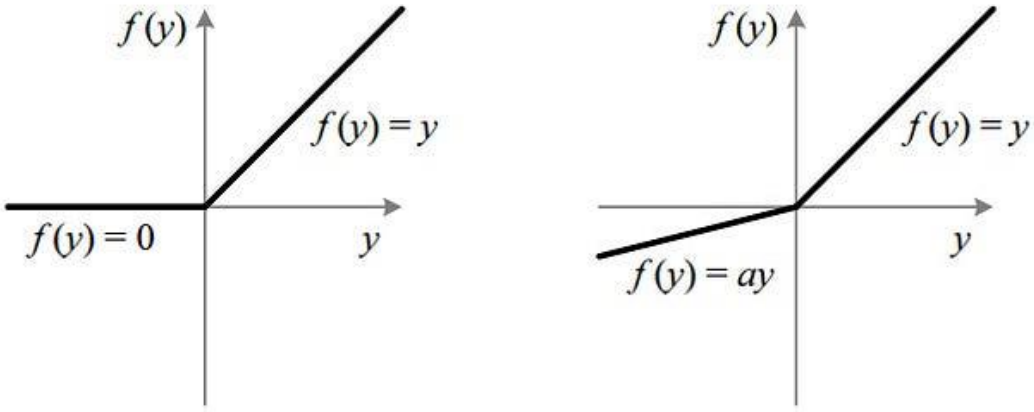


Fig : ReLU v/s Leaky ReLU

Can you see the Leak? 😆

The leak helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so.

When **a is not 0.01** then it is called **Randomized ReLU**.

Therefore the **range** of the Leaky ReLU is (-infinity to infinity).

Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

Single layer perceptron

freely.

PERCEPTRON N/W :-

Researchers find out :-

① McCulloch (1943) introduce the idea of NN.

② Hebb (1944), introduce first rule for self-organized learning (unsupervised learning).

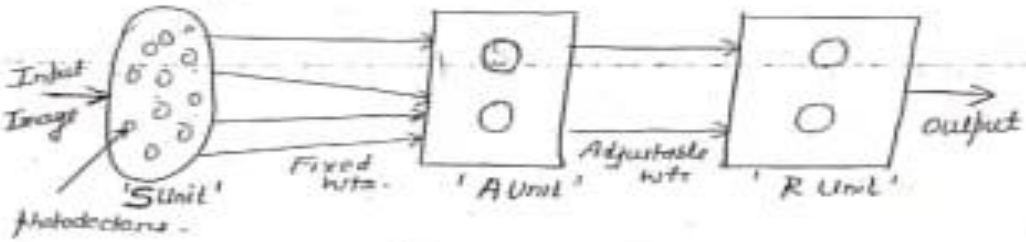
③ Rosenblatt (1958), introduce first model for learning with a teacher (supervised learning) called perceptron model.

Frank Rosenblatt in 1962, developed large class of ANN called perceptrons. The perceptron learning rule uses an iterative weight adjustment that is more powerful than Hebb Rule.

The iterative learning of this n/w converges to correct weights : i.e the weight that produce the exact output value for the training input pattern.

Perceptron Model have 3 units:-

- Sensory Unit (S)
- Association Unit (A)
- Response Unit (R).



This perceptron is a computational model of the retina of the eye.

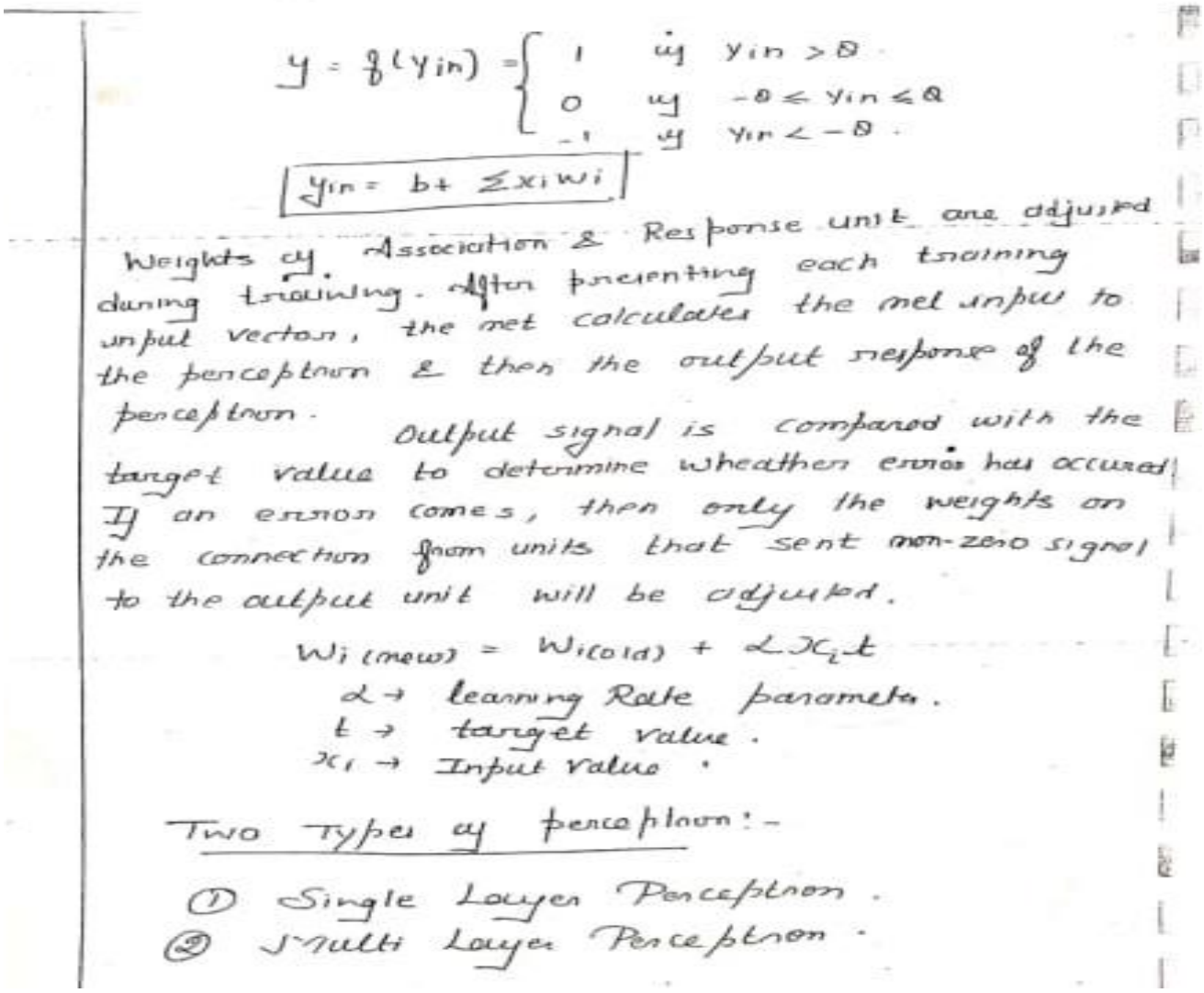
Sensory Unit have 400 photodectors. receives input images & provides 0 or 1 as output If input signal exceed a threshold value 'θ' then photodector output is 1 else 0. Photodectors are randomly connected with Association Unit.

Association Unit have predicates, examine the output of S unit for specific feature of the image.

Response Unit is a pattern recognizer or perceptrons, which receives the result of the predicate, in binary form.

Weight of Sensory Unit & Association Unit are fixed, Weight of Response Unit are adjustable

The output of R unit could be such that if the wt sum of its inputs is less than or equal to 0, then output is 0 else 1.



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.

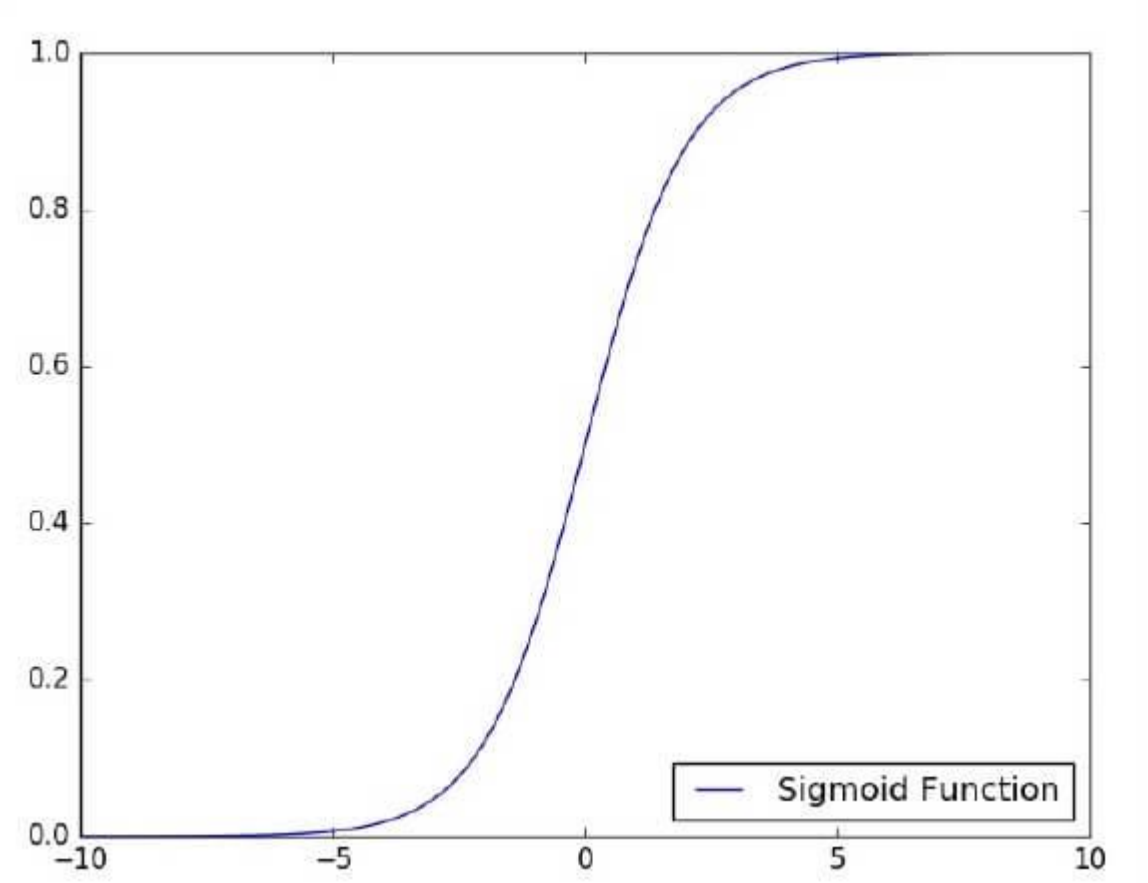
To understand the perceptron layer, it is necessary to comprehend artificial neural networks (ANNs).

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.

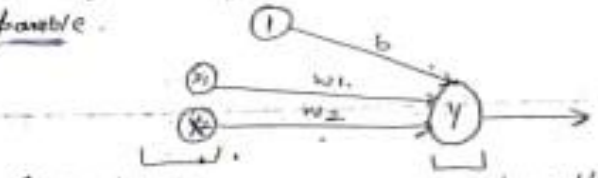
Let us focus on the implementation of a single-layer perceptron for an image classification problem using TensorFlow. The best example of drawing a single-layer perceptron is through the representation of "**logistic regression**."



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.

SINGLE LAYER :- It is a simple model of NN, for classification of patterns that are linearly separable.



Perceptron has sensory, association & Response units. The input to the Response unit will be the output from the association unit, which is binary vector.

In Single layer perceptron Sensory unit is hidden, ∴ wts between the Sensory & Res Associa. unit is fixed while wts between Association & Response unit are adjusted during training / learning.

In this Input layer consist of input neuron. x_1, x_2 & Bias b.

Input neurons are connected to the output neurons through the weighted interconnection.

It is a single layer, because it has only one layer of interconnections between the input & output neurons.

To start the training process, initially the wts & bias set to 0.

The intal wts of the n/w can be formulated from other techniques like Fuzzy system, Genetic Algo.

Also set learning rate parameter α which range 0 to 1.

Input is presented, the net input is calculated by multiplying the wt's with the inputs & adding the result with the bias

$$y_{in} = b + \sum x_i w_i$$

Once, net input is calculated, then by applying Activation Function, output of n/w also obtained. Output is compared with the target, if they are not equal, then we update wt's & bias by perceptron Learning Rule:

Activation Function:-

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Threshold fixed

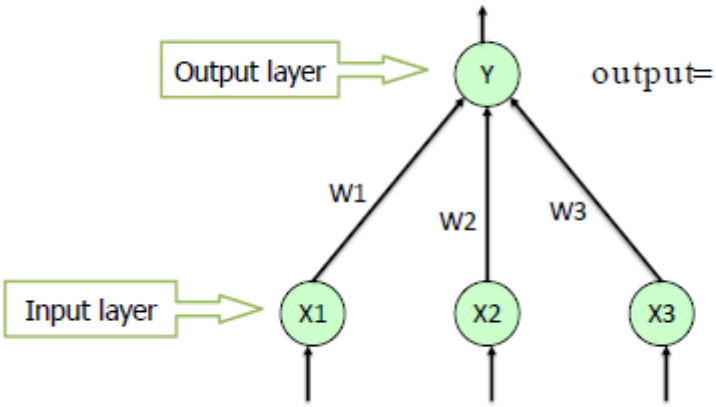
ALGO:-

1. Initialize weights & bias to 0.
   Set learning Rate parameter α to 1.
   Threshold value θ to 0.
2. While stopping condition is false do steps 3-7
3. For each training pair s:t do step 4-6
4. set activation of input units.
   $x_i = s_i \quad \forall \; i = 1 \text{ to } n$
5. Calculate the Output Unit Response.
   $y_{in} = b + \sum x_i w_i$

   Activation Function

$$y = f(y_{in}) \quad \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Single Layer Perceptron



19.

⑥ Wts & bias are updated
if the target is not equal to output response.

If $t \neq y$ $\quad w_{new} = w_{old} + \alpha t x$.

$b_{new} = b_{old} + \alpha t$

else $(t = y)$ $\quad w_{new} = w_{old}$

$b_{new} = b_{old}$

⑦ Test for stopping condition.
The stopping condition may be wt changes.

BACK PROPAGATION NETWROK

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.

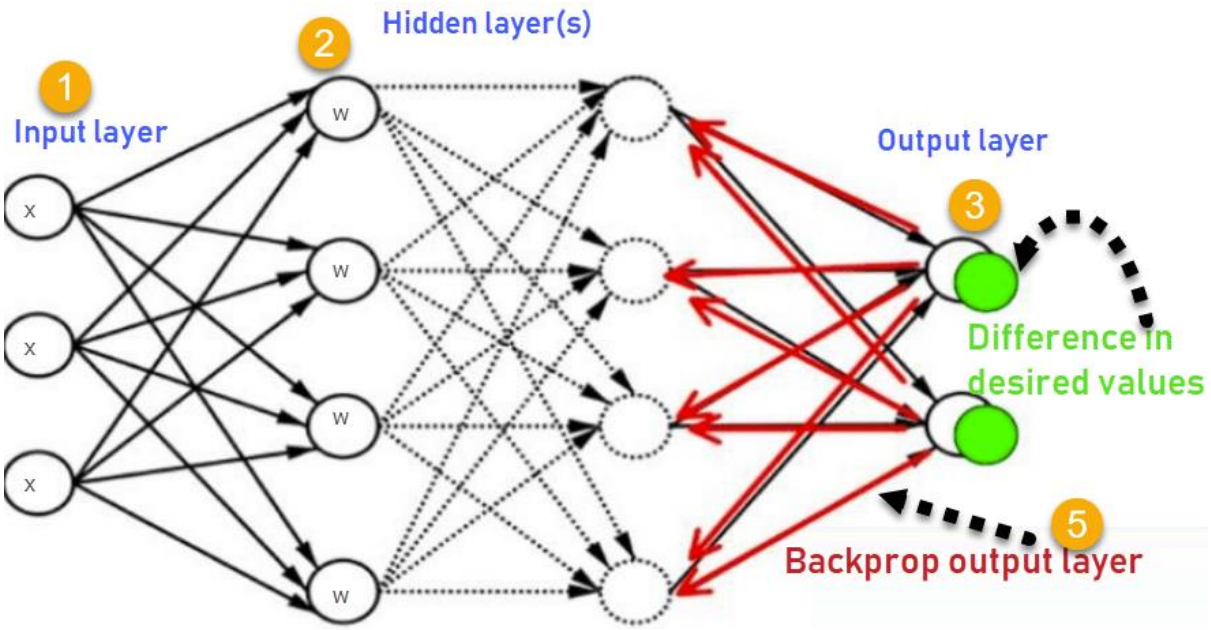
Features of Backpropagation:

1. it is the gradient descent 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 of input training pattern
    - the calculation and backpropagation of the error
    - updation of the weight

How Backpropagation Algorithm Works

The Back propagation algorithm in neural network computes the gradient of the loss function for a single weight by the chain rule. It efficiently computes one layer at a time, unlike a native direct computation. It computes the gradient, but it does not define how the gradient is used. It generalizes the computation in the delta rule.

Consider the following Back propagation neural network example diagram to understand:



1. Inputs X, arrive through the preconnected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

$$Error_B = Actual\ Output - Desired\ Output$$

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Keep repeating the process until the desired output is achieved

Why We Need Backpropagation?

Most prominent advantages of Backpropagation are:

- Backpropagation is fast, simple and easy to program
- It has no parameters to tune apart from the numbers of input
- It is a flexible method as it does not require prior knowledge about the network
- It is a standard method that generally works well
- It does not need any special mention of the features of the function to be learned.

Types of Backpropagation Networks

Two Types of Backpropagation Networks are:

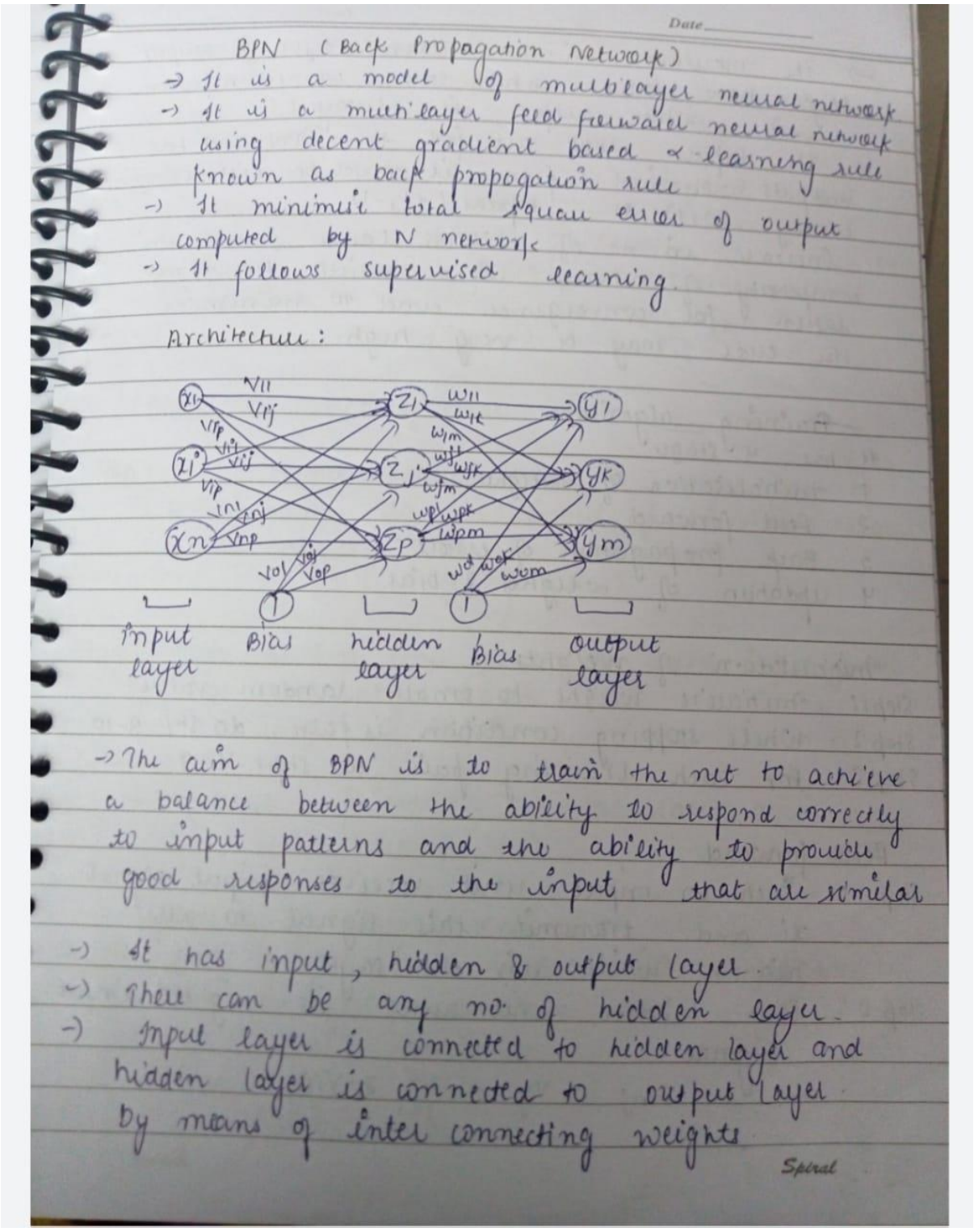
- Static Back-propagation
- Recurrent Backpropagation

Static back-propagation:
It is one kind of backpropagation network which produces a mapping of a static input for static output. It is useful to solve static classification issues like optical character recognition.
Recurrent Backpropagation:
Recurrent Back propagation in data mining is fed forward until a fixed value is achieved. After that, the error is computed and propagated backward.
The main difference between both of these methods is: that the mapping is rapid in static back-propagation while it is nonstatic in recurrent backpropagation.

Disadvantages of using Backpropagation

- The actual performance of backpropagation on a specific problem is dependent on the input data.
- Back propagation algorithm in data mining can be quite sensitive to noisy data
- You need to use the matrix-based approach for backpropagation instead of mini-batch.

Architecture of Backpropagation network



Date

BPN (Back Propagation Network)

→ It is a model of multilayer neural network
→ It is a multilayer feed forward neural network using decent gradient based learning rule known as back propagation rule
→ It minimise total squar error of output computed by N network
→ It follows supervised learning.

Architecture :

→ The aim of BPN is to train the net to achieve a balance between the ability to respond correctly to input patterns and the ability to provide good responses to the input that are similar

→ It has input, hidden & output layer.
→ There can be any no. of hidden layer.
→ Input layer is connected to hidden layer and hidden layer is connected to output layer by means of inter connecting weights

Spiral

Date

→ The neurons in the hidden layer & output layer have biases which are connection from units whose output is always 1

→ The inputs that are fed to BPN and the outputs obtained from it could be either binary (0,1) or bipolar (+1, -1)

→ Increase in no. of hidden layer result in complexity of network. As a result the time taken for convergence and to minimize the error may be very high.

Training algorithm

It has 4 stages

1. Initialisation of weights
2. Feed forward
3. Back propagation of errors
4. Updation of weights & bias

Initialisation of weights

Step 1: Initialize weight to small random value.

Step 2: While stopping condition is false, do step 3-10

Step 3: For each training pair do steps 4-9

Feed forward

Step 4: Each input unit receives input signal x_i and transmit this signal to all hidden units in the layer.

Step 5: Each hidden unit sums its weighted input signal

$$z_{inj} = v_{0j} + \sum_{i=1}^{n} x_i \cdot v_{ij}$$

Spiral

Date

apply activation function

$z_j = f(z_{inj})$

send this signal to all above layers

Step 6: Each output unit sums its weighted input signal

$$y_{ink} = w_{ok} + \sum_{j=1}^{n} z_j w_{jk}$$

apply activation function to calculate output signal

$$y_k = f(y_{ink})$$

Back propagation of error

Step 7: Each output unit receives a target pattern corresponding to an input pattern and error information term is calculated

$$\delta_k = (t_k - y_k) f(y_{ink})$$

Step 8: Each hidden unit sums its delta input from units in above layer.

$$\delta_{inj} = \sum_{k=1} \delta_j w_{jk}$$

The error information is calculated as

$$\Delta = \delta_{inj} f(z_{inj})$$

Updation of weights and bias

Step 9: Each output unit updates its weight and bias

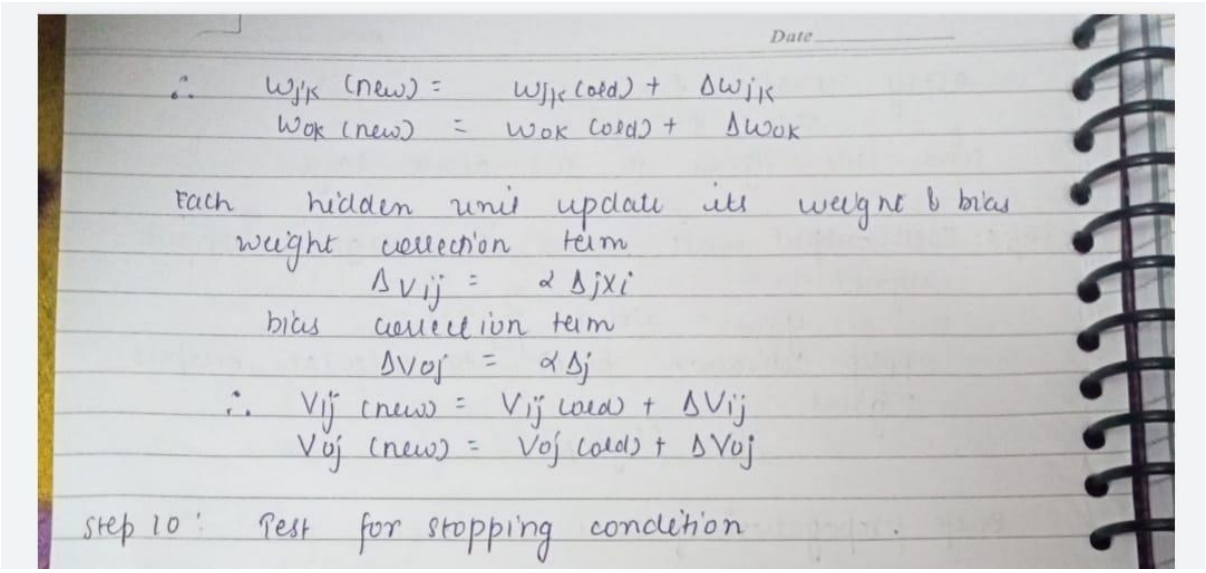
The weight correction term

$$\Delta w_{jk} = \alpha \delta_k z_j$$

Bias correction term

$$\Delta w_{ok} = \alpha \delta_k$$

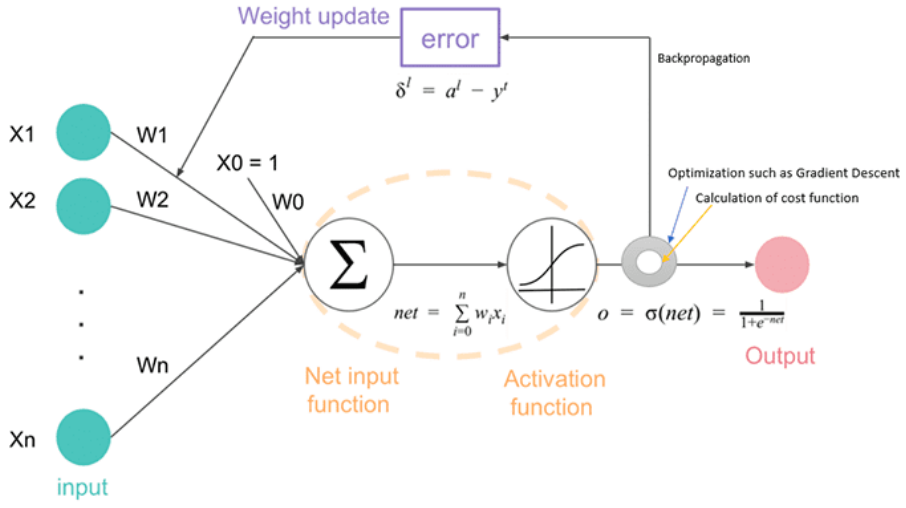
Spiral



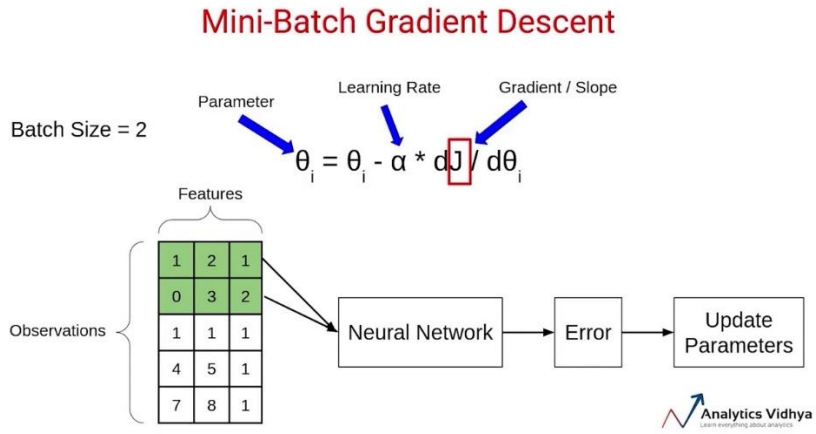
Variation of standard back propagation

There are several variations and enhancements to the standard backpropagation algorithm that have been proposed to improve training efficiency, address certain limitations, or achieve better performance in specific scenarios. Here are a few notable variations of standard backpropagation:

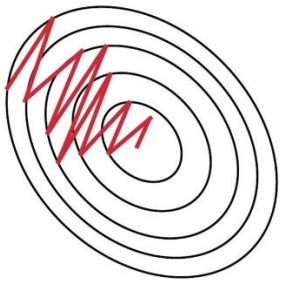
1. Stochastic Gradient Descent (SGD): In standard backpropagation, the weights and biases are updated after computing the gradients over the entire training dataset. In stochastic gradient descent, the updates are made after computing the gradients on a subset of the training data, typically one training example or a small batch of examples. This approach can converge faster and is more computationally efficient, especially for large datasets.



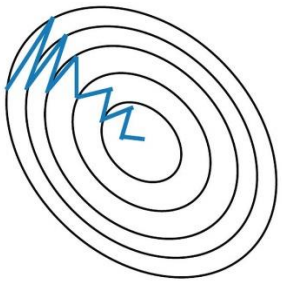
2. Mini-batch Gradient Descent: This variation is similar to stochastic gradient descent, but instead of updating the weights after processing a single training example, updates are made after processing a small batch of training examples. This strikes a balance between the efficiency of stochastic gradient descent and the stability of batch gradient descent (where updates are made after processing the entire training dataset). Mini-batch gradient descent is commonly used in practice.



3. Momentum: Momentum is a technique that introduces an additional term in the weight update process to accelerate learning and help overcome local minima. It adds a fraction of the previous weight update to the current update, which helps the optimization process to keep moving consistently in the same direction, especially in flat or shallow regions of the error surface.



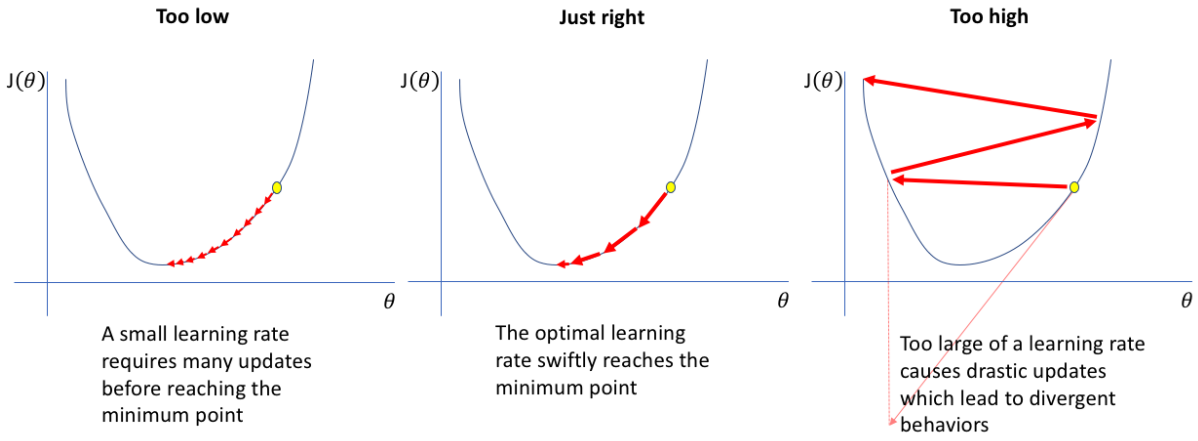
Stochastic Gradient Descent **without** Momentum



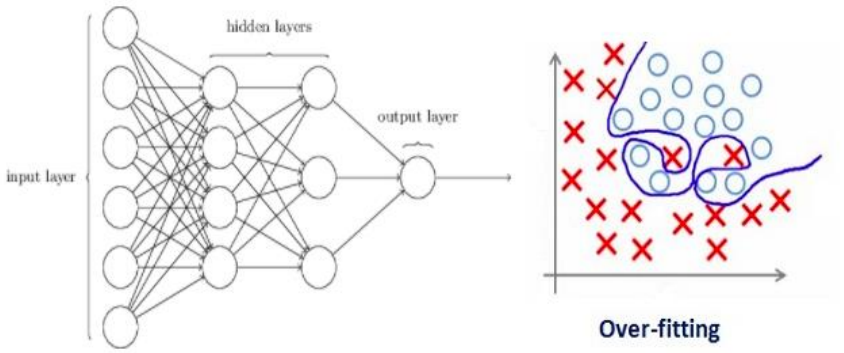
Stochastic Gradient Descent **with** Momentum

4. Learning Rate Schedules: The learning rate determines the step size in the weight update process. Instead of using a fixed learning rate, variations of backpropagation often employ learning rate schedules. These schedules dynamically adjust the learning rate during training. For example, a common

approach is to start with a relatively high learning rate and gradually decrease it over time to allow for faster progress initially and more fine-tuning later.



5. Regularization Techniques: Regularization is used to prevent overfitting, which occurs when a model becomes too complex and performs well on the training data but fails to generalize to unseen data. Variations of backpropagation often incorporate regularization techniques such as L1 or L2 regularization (weight decay), dropout, or batch normalization to improve generalization and prevent overfitting.



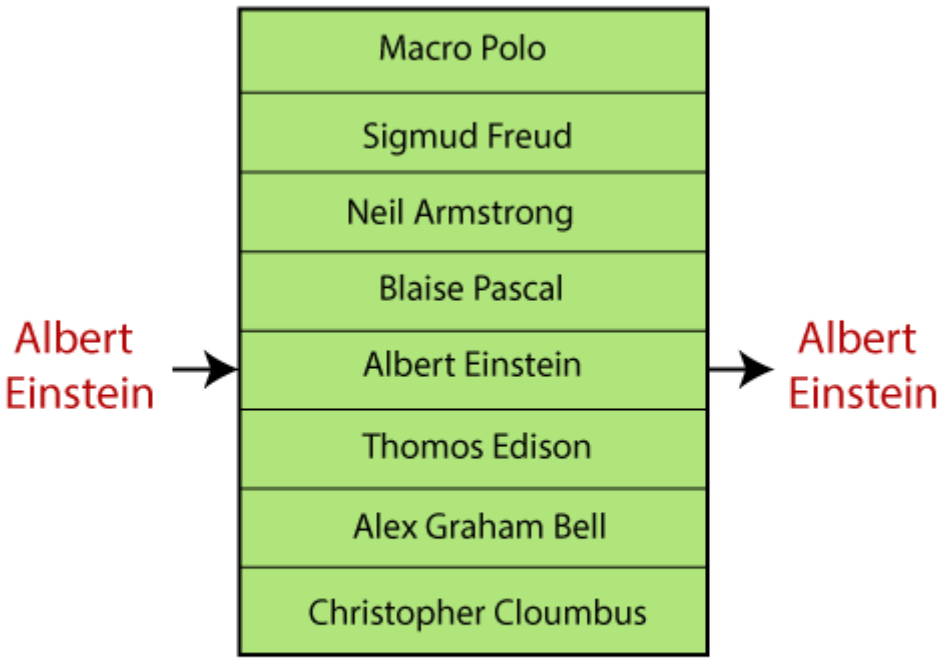
These are just a few examples of the many variations and enhancements to the standard backpropagation algorithm. Different variations may be more suitable for specific problem domains or network architectures, and researchers continue to explore new techniques to further enhance the training of neural networks.

INTRODUCTION TO ASSOCIATIVE MEMORY

- An associative memory is a content addressable structure that maps a set of input patterns to a set out patterns
- A content addressable structure is a type of a memory that allow the recall of database on the degree of similarity between input pattern and the pattern stored in the memory .
- There are two type of associative memory

  -Auto-associative memory

  -Hetro associative memory
- An auto-associative memory retrieves a previously stored pattern that most closely resembles the current pattern.
- In hetro-associative memory retrieve the pattern is in general ,different from the input pattern not only in the content but possibly in the type and format.
- Neural network use these associative memory models called NAM (neural associative model)
- EXAMPLE-

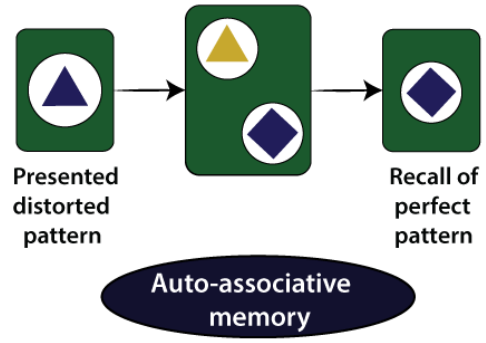


Content-addressable memory, Input and Output

- The figure given below illustrates a memory containing the names of various people. If the given memory is content addressable, the incorrect string "Albert Einstein" as a key is sufficient to recover the correct name "Albert Einstein."
- In this condition, this type of memory is robust and fault-tolerant because of this type of memory model, and some form of error-correction capability.

Auto-associative memory:

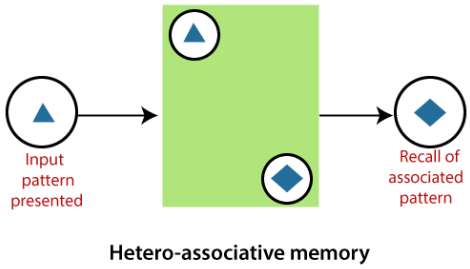
An auto-associative memory recovers a previously stored pattern that most closely relates to the current pattern. It is also known as an auto-associative correlator.



Consider x[1], x[2], x[3],..... x[M], be the number of stored pattern vectors, and let x[m] be the element of these vectors, showing characteristics obtained from the patterns. The auto-associative memory will result in a pattern vector x[m] when putting a noisy or incomplete version of x[m].

Hetero-associative memory:

In a hetero-associate memory, the recovered pattern is generally different from the input pattern not only in type and format but also in content. It is also known as a hetero-associative correlator.



Hetero-associative memory

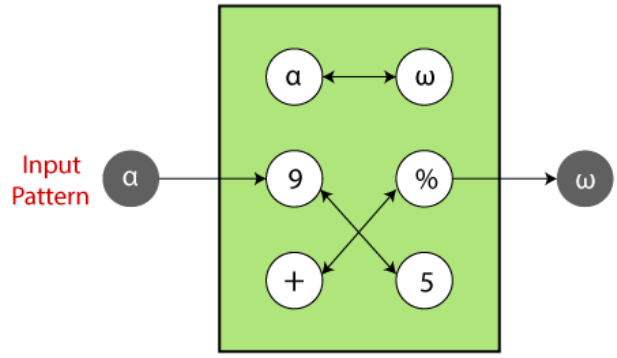
Consider we have a number of key response pairs {a(1), x(1)}, {a(2),x(2)},.....,{a(M), x(M)}. The hetero-associative memory will give a pattern vector x(m) when a noisy or incomplete version of the a(m) is given.

Neural networks are usually used to implement these associative memory models called neural associative memory (NAM). The linear associate is the easiest artificial neural associative memory.

These models follow distinct neural network architecture to memorize data.

Working of Associative Memory:

Associative memory is a depository of associated pattern which in some form. If the depository is triggered with a pattern, the associated pattern pair appear at the output. The input could be an exact or partial representation of a stored pattern.



Working of an associated memory

If the memory is produced with an input pattern, may say α, the associated pattern ω is recovered automatically.

These are the terms which are related to the Associative memory network:

Encoding or memorization:

Encoding or memorization refers to building an associative memory. It implies constructing an association weight matrix w such that when an input pattern is given, the stored pattern connected with the input pattern is recovered.

$(W_{ij})_k = (p_i)_k (q_j)_k$

Where,

$(P_i)_k$ represents the i^{th} component of pattern p_k, and

$(q_j)_k$ represents the j^{th} component of pattern q_k

Where,

strong>i = 1,2, …,m and **j** = 1,2,…,n.

Constructing the association weight matrix **w** is accomplished by adding the individual correlation matrices w_k , i.e.,

$$W = \alpha \sum_{k=1}^{p} wk$$

Where α = Constructing constant.

Errors and noise:

The input pattern may hold errors and noise or may contain an incomplete version of some previously encoded pattern. If a corrupted input pattern is presented, the network will recover the stored Pattern that is adjacent to the actual input pattern. The existence of noise or errors results only in an absolute decrease rather than total degradation in the efficiency of the network. Thus, associative memories are robust and error-free because of many processing units performing highly parallel and distributed computations.

Performance Measures:

The measures taken for the associative memory performance to correct recovery are memory capacity and content addressability. **Memory capacity** can be defined as the maximum number of associated pattern pairs that can be stored and correctly recovered. **Content-addressability** refers to the ability of the network to recover the correct stored pattern.

If input patterns are mutually orthogonal, perfect recovery is possible. If stored input patterns are not mutually orthogonal, non-perfect recovery can happen due to intersection among the patterns.

Associative memory models:

Linear associator is the simplest and most widely used associative memory models. It is a collection of simple processing units which have a quite complex collective computational capability and behavior. The Hopfield model computes its output that returns in time until the system becomes stable. Hopfield networks are constructed using bipolar units and a learning process. The **Hopfield model** is an **auto-associative memory** suggested by **John Hopfield** in **1982**. **Bidirectional Associative Memory (BAM)** and the Hopfield model are some other popular artificial neural network models used as associative memories.

Network architectures of Associate Memory Models:

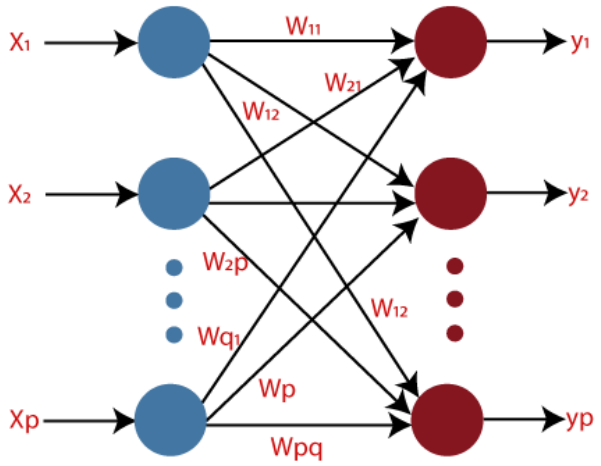
The neural associative memory models pursue various neural network architectures to memorize data. The network comprises either a single layer or two layers. The linear associator model refers to a feed-forward type network, comprises of two layers of different processing

units- The first layer serving as the input layer while the other layer as an output layer. The Hopfield model refers to a single layer of processing elements where each unit is associated with every other unit in the given network. The **bidirectional associative memory (BAM)** model is the same as the linear associator, but the associations are bidirectional.

The neural network architectures of these given models and the structure of the corresponding association weight matrix **w** of the associative memory are depicted.

Linear Associator model (two layers):

The linear associator model is a feed-forward type network where produced output is in the form of single feed-forward computation. The model comprises of two layers of processing units, one work as an input layer while the other work as an output layer. The input is directly associated with the outputs, through a series of weights. The connections carrying weights link each input to every output. The addition of the products of the weights and the input is determined in each neuron node. The architecture of the linear associator is given below.



Linear associator model

All p inputs units are associated to all q output units via associated weight matrix

W = [w_{ij}]p * q where w_{ij} describes the strength of the unidirectional association of the i^{th} input unit to the j^{th} output unit.

The connection weight matrix stores the **z** different associated pattern pairs **{(X_k,Y_k); k= 1,2,3,...,z}**. Constructing an associative memory is building the connection weight matrix **w** such that if an input pattern is presented, the stored pattern associated with the input pattern is recovered.

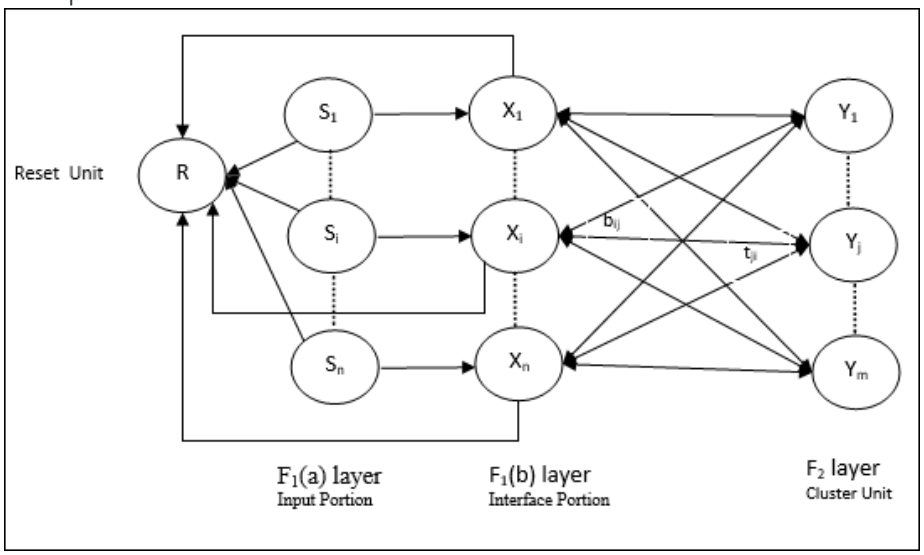
ADAPTIVE RESONANCE THEORY

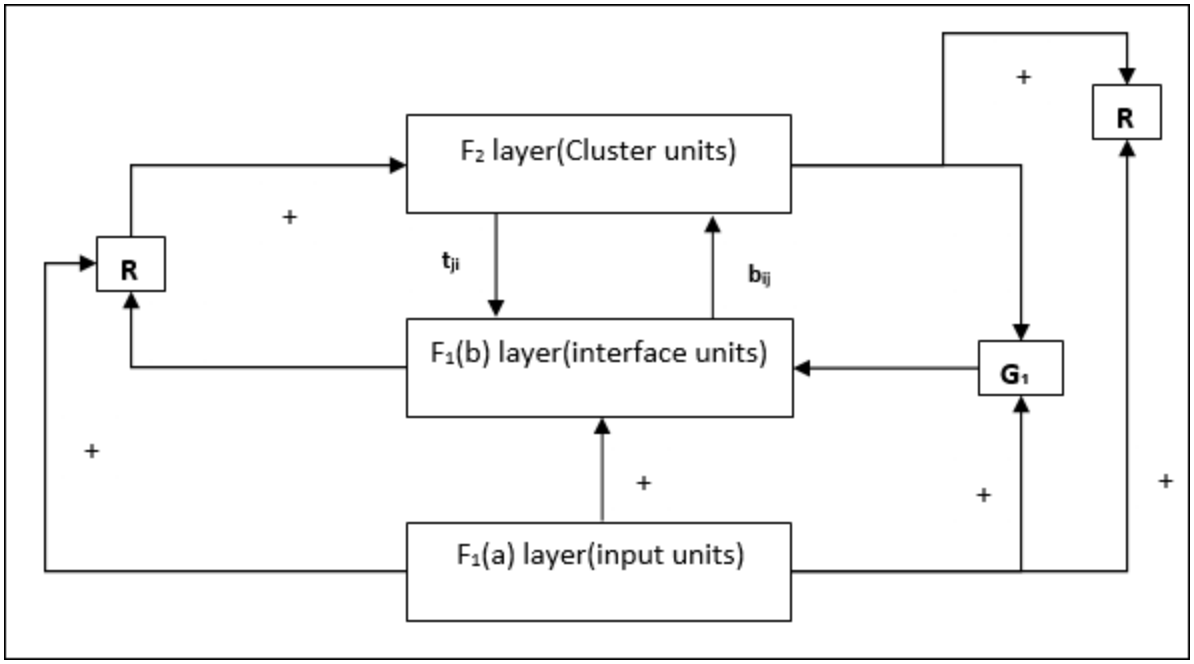
Adaptive Resonance Theory (ART) Adaptive resonance theory is a type of neural network technique developed by Stephen Grossberg and Gail Carpenter in 1987. The basic ART uses unsupervised learning technique. The term **"adaptive"** and **"resonance"** used in this suggests that they are open to new learning(i.e. adaptive) without discarding the previous or the old information(i.e. resonance). The ART networks are known to solve the stability-plasticity dilemma i.e., stability refers to their nature of memorizing the learning and plasticity refers to the fact that they are flexible to gain new information. Due to this the nature of ART they are always able to learn new input patterns without forgetting the past. ART networks implement a clustering algorithm. Input is presented to the network and the algorithm checks whether it fits into one of the already stored clusters. If it fits then the input is added to the cluster that matches the most else a new cluster is formed.

Types of Adaptive Resonance Theory(ART) Carpenter and Grossberg developed different ART architectures as a result of 20 years of research. The ARTs can be classified as follows:

- ART1 – It is the simplest and the basic ART architecture. It is capable of clustering binary input values.
- ART2 – It is extension of ART1 that is capable of clustering continuous-valued input data.
- Fuzzy ART – It is the augmentation of fuzzy logic and ART.
- ARTMAP – It is a supervised form of ART learning where one ART learns based on the previous ART module. It is also known as predictive ART.
- FARTMAP – This is a supervised ART architecture with Fuzzy logic included.

Basic of Adaptive Resonance Theory (ART) Architecture The adaptive resonant theory is a type of neural network that is self-organizing and competitive. It can be of both types, the unsupervised ones(ART1, ART2, ART3, etc) or the supervised ones(ARTMAP). Generally, the supervised algorithms are named with the suffix "MAP". But the basic ART model is unsupervised in nature and consists of :





- F1 layer or the comparison field(where the inputs are processed)
- F2 layer or the recognition field (which consists of the clustering units)
- The Reset Module (that acts as a control mechanism)
- Supplement Unit − Actually the issue with Reset mechanism is that the layer F_2 must have to be inhibited under certain conditions and must also be available when some learning happens. That is why two supplemental units namely, G_1 and G_2 is added along with reset unit, R. They are called gain control units. These units receive and send signals to the other units present in the network. '+' indicates an excitatory signal, while '−' indicates an inhibitory signal.

The F1 layer accepts the inputs and performs some processing and transfers it to the F2 layer that best matches with the classification factor. There exist two sets of weighted interconnection for controlling the degree of similarity between the units in the F1 and the F2 layer. The F2 layer is a competitive layer. The cluster unit with the large net input becomes the candidate to learn the input pattern first and the rest F2 units are ignored. The reset unit makes the decision whether or not the cluster unit is allowed to learn the input pattern depending on how similar its top-down weight vector is to the input vector and to the decision. This is called the vigilance test. Thus we can say that the vigilance parameter helps to incorporate new memories or new information. Higher vigilance produces more detailed memories, lower vigilance produces more general memories. Generally two types of learning exists,slow learning and fast learning. In fast learning, weight update during resonance occurs rapidly. It is used in ART1.In slow learning, the weight change occurs slowly relative to the duration of the learning trial. It is used in ART2.

Alogorithm of ART

Parameters Used

Following parameters are used −

- n − Number of components in the input vector
- m − Maximum number of clusters that can be formed
- b_{ij} − Weight from F_1 b to F_2 layer, i.e. bottom-up weights
- t_{ji} − Weight from F_2 to F_1 b layer, i.e. top-down weights

- ρ – Vigilance parameter
- ||x|| – Norm of vector x

Step 1 – Initialize the learning rate, the vigilance parameter, and the weights as follows –

$$\alpha > 1 \ and \ 0 < \rho \leq 1$$

$$0 < b_{ij}(0) < \frac{\alpha}{\alpha - 1 + n} \ and \ t_{ij}(0) = 1$$

Step 2 – Continue step 3-9, when the stopping condition is not true.

Step 3 – Continue step 4-6 for every training input.

Step 4 – Set activations of all F_1 a and F_1 units as follows
F_2 = 0 and F_1 a = input vectors

Step 5 – Input signal from F_1 a to F_1 b layer must be sent like

$$s_i = x_i$$

Step 6 – For every inhibited F_2 node

$$y_j = \sum_i b_{ij} x_i \quad \text{the condition is } y_j \neq -1$$

Step 7 – Perform step 8-10, when the reset is true.

Step 8 – Find J for $y_J \geq y_j$ for all nodes j

Step 9 – Again calculate the activation on F_1b as follows

$$x_i = sitJi$$

Step 10 – Now, after calculating the norm of vector **x** and vector **s**, we need to check the reset condition as follows –

If ||x||/ ||s|| < vigilance parameter ρ, then inhibit node J and go to step 7

Else If ||x||/ ||s|| ≥ vigilance parameter ρ, then proceed further.

Step 11 – Weight updating for node J can be done as follows –

$$b_{ij}(new) = \frac{\alpha x_i}{\alpha - 1 + ||x||}$$

$$t_{ij}(new) = x_i$$

Step 12 – The stopping condition for algorithm must be checked and it may be as follows –

- Do not have any change in weight.
- Reset is not performed for units.
- Maximum number of epochs reached

Advantage of Adaptive Resonance Theory (ART)

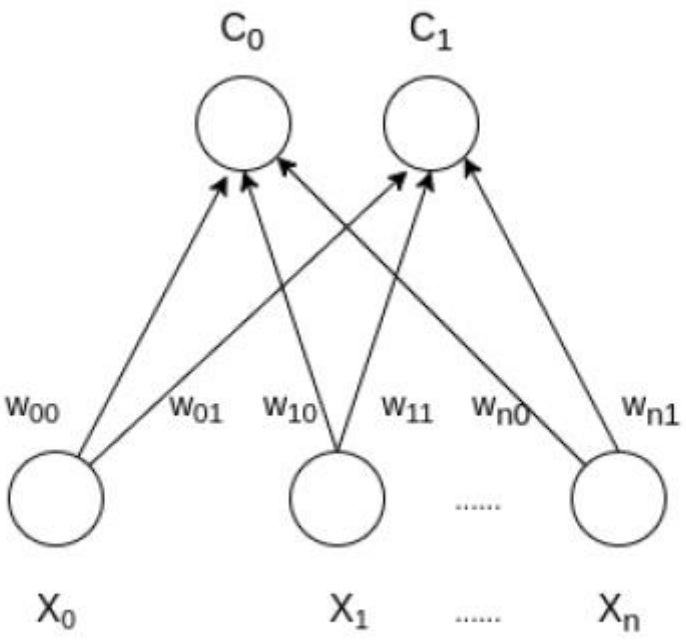
- It exhibits stability and is not disturbed by a wide variety of inputs provided to its network.
- It can be integrated and used with various other techniques to give more good results.
- It can be used for various fields such as mobile robot control, face recognition, land cover classification, target recognition, medical diagnosis, signature verification, clustering web users, etc.
- It has got advantages over competitive learning (like bpnn etc). The competitive learning lacks the capability to add new clusters when deemed necessary.
- It does not guarantee stability in forming clusters.

Limitations of Adaptive Resonance Theory Some ART networks are inconsistent (like the Fuzzy ART and ART1) as they depend upon the order of training data, or upon the learning rate.

SELF ORGANISING MAP

Self Organizing Map (or Kohonen Map or SOM) is a type of Artificial Neural Network which is also inspired by biological models of neural systems from the 1970s. It follows an unsupervised learning approach and trained its network through a competitive learning algorithm. SOM is used for clustering and mapping (or dimensionality reduction) techniques to map multidimensional data onto lower-dimensional which allows people to reduce complex problems for easy interpretation. SOM has two layers, one is the Input layer and the other one is the Output layer.

The architecture of the Self Organizing Map with two clusters and n input features of any sample is given below:



How do SOM works?
Let's say an input data of size (m, n) where m is the number of training examples and n is the number of features in each example. First, it initializes the weights of size (n, C) where C is the number of clusters. Then iterating over the input data, for each training example, it updates the winning vector (weight vector with the shortest distance (e.g Euclidean distance) from training example). Weight updation rule is given by :

$$w_{ij} = w_{ij}(old) + alpha(t) * (x_i^k - w_{ij}(old))$$

where alpha is a learning rate at time t, j denotes the winning vector, i denotes the i^{th} feature of training example and k denotes the k^{th} training example from the input data. After training the SOM network, trained weights are used for clustering new examples. A new example falls in the cluster of winning vectors.

Algorithm

Training:

Step 1: Initialize the weights w_{ij} random value may be assumed. Initialize the learning rate α.

Step 2: Calculate squared Euclidean distance.

$$D(j) = \Sigma (wij - xi)^2 \quad \text{where } i=1 \text{ to } n \text{ and } j=1 \text{ to } m$$

Step 3: Find index J, when D(j) is minimum that will be considered as winning index.

Step 4: For each j within a specific neighborhood of j and for all i, calculate the new weight.

$$wij(new)=wij(old) + \alpha[xi - wij(old)]$$

Step 5: Update the learning rule by using :

$$\alpha(t+1) = 0.5 * t$$

Step 6: Test the Stopping Condition.

APPLICATION OF NEURAL NETWORK

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.

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.

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.

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.

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.

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.