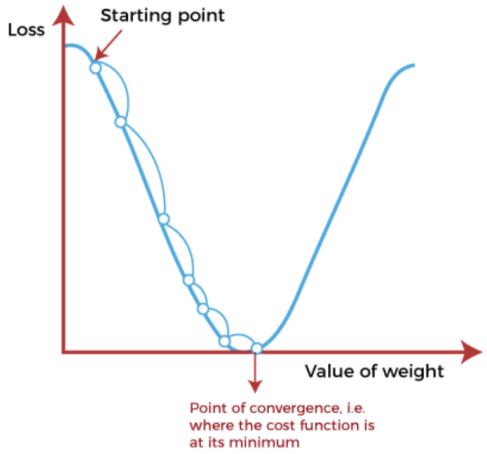
NOTE: PLEASE REFER TO THIS AT YOUR OWN RISK

1. Explain the term Gradient Descent in Machine Learning.

Ans) Gradient descent is an optimization algorithm which is commonly-used to train machine learning models and neural networks.

The goal of gradient descent is to minimize the cost function, or the error between predicted and actual y. In order to do this, it requires two data points—a direction and a learning rate. These factors determine the partial derivative calculations of future iterations, allowing it to gradually arrive at the local or global minimum (i.e. point of convergence).

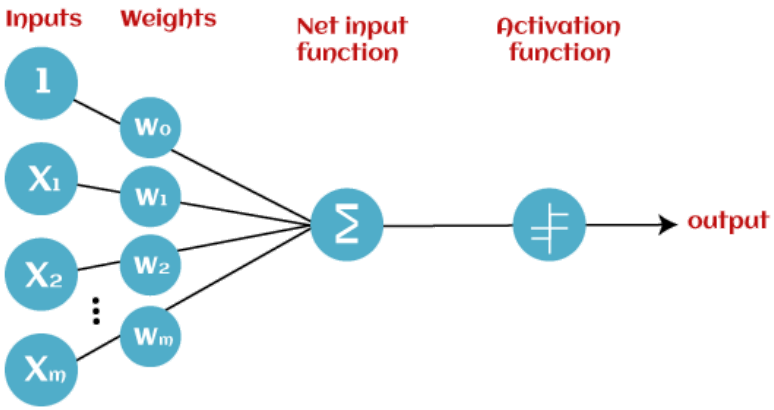


Learning rate (also referred to as step size or the alpha) is the size of the steps that are taken to reach the minimum. This is typically a small value, and it is evaluated and updated based on the behavior of the cost function. High learning rates result in larger steps but risks overshooting the minimum. Conversely, a low learning rate has small step sizes. While it has the advantage of more precision, the number of iterations compromises overall efficiency as this takes more time and computations to reach the minimum.

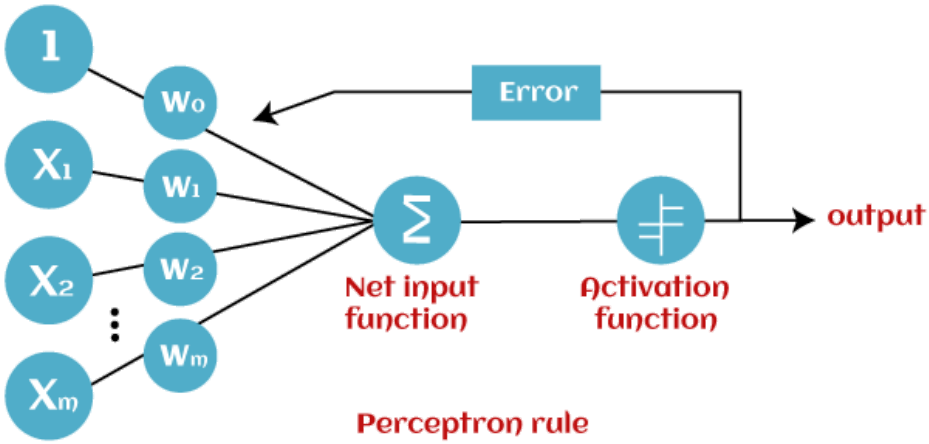
The cost (or loss) function measures the difference, or error, between actual y and predicted y at its current position. This improves the machine learning model's efficiency by providing feedback to the model so that it can adjust the parameters to minimize the error and find the local or global minimum. It continuously iterates, moving along the direction of steepest descent (or the negative gradient) until the cost function is close to or at zero. At this point, the model will stop learning. A loss function refers to the error of one training example, while a cost function calculates the average error across an entire training set.

2. Explain the concept of perceptron.

Ans) Perceptron is a 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.



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.



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).

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:

$\sum wi*xi = x1*w1 + x2*w2 + \ldots wn*xn$

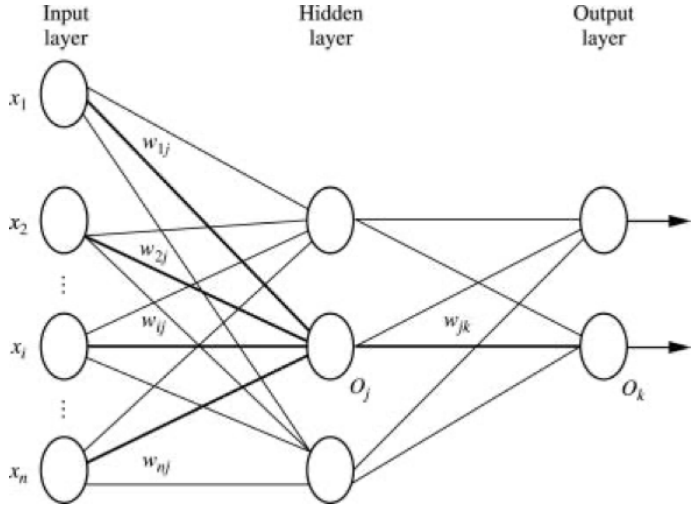
Add a special term called bias 'b' to this weighted sum to improve the model's performance.

$\sum wi*xi + b$

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)

3. Explain the term backpropagation.
Ans) Backpropagation, or backward propagation of errors, is an algorithm that is designed to test for errors working back from output nodes to input nodes. It is an important mathematical tool for improving the accuracy of predictions in data mining and machine learning.
Artificial neural networks use backpropagation as a learning algorithm to compute a gradient descent with respect to weight values for the various inputs. By comparing desired outputs to achieved system outputs, the systems are tuned by adjusting connection weights to narrow the difference between the two as much as possible.
They efficiently compute the gradient of the loss function with respect to the network weights. This approach eliminates the inefficient process of directly computing the gradient with respect to each individual weight. It enables the use of gradient methods, like gradient descent or stochastic gradient descent, to train multilayer networks and update weights to minimize loss.

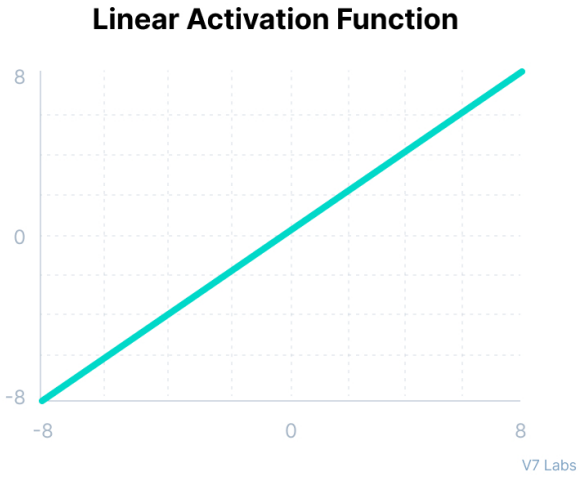


WORKING:
Step 1: Inputs X, arrive through the preconnected path.
Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.
Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.
Step 4: Calculate the error in the outputs
Backpropagation Error= Actual Output – Desired Output
Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.
Step 6: Repeat the process until the desired output is achieved.

4. Explain the activation functions used in neural networks.
Ans) The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

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.
The two main categories of activation functions are:

- **Linear Activation Function**
  The linear activation function, also known as "no activation," or "identity function" (multiplied x1.0), is where the activation is proportional to the input.
  The function doesn't do anything to the weighted sum of the input, it simply splits out the value it was given.



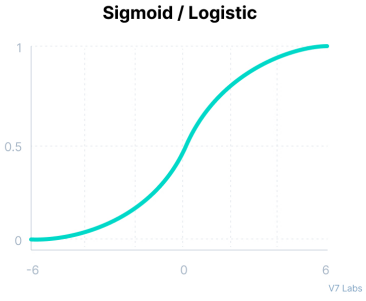
Mathematically it can be represented as:

$$Linear$$

$$f(x) = x$$

- **Non-linear Activation Functions**

  Sigmoid / Logistic Activation Function
  This function takes any real value as input and outputs values in the range of 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0, as shown below.



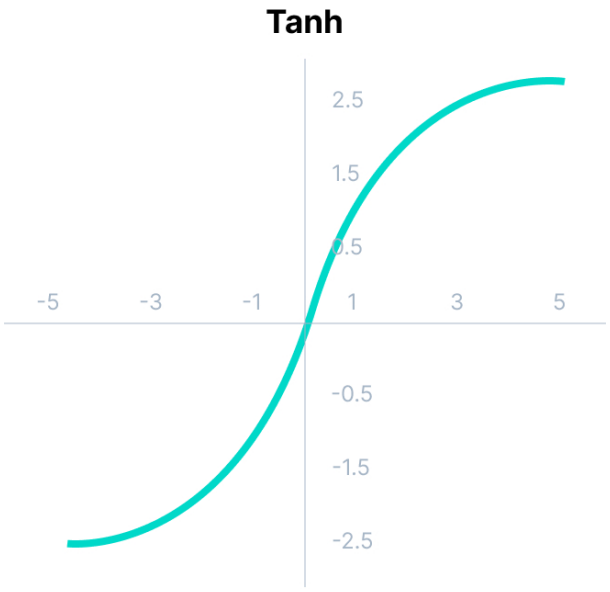
Mathematically it can be represented as:

$$Sigmoid\ /\ Logistic$$

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

Tanh Function (Hyperbolic Tangent)

Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.



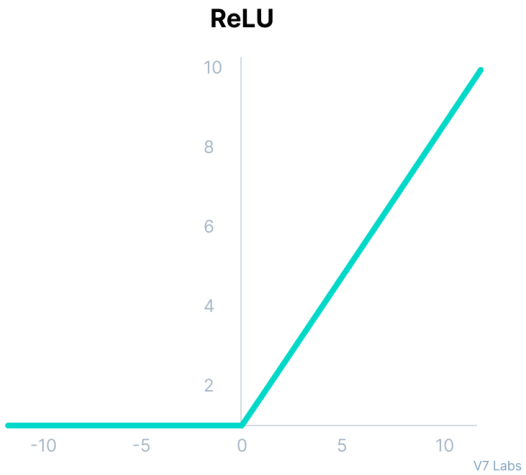
Mathematically it can be represented as:

$$Tanh$$

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

ReLU Function

ReLU stands for Rectified Linear Unit.
Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient.
The main catch here is that the ReLU function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0.



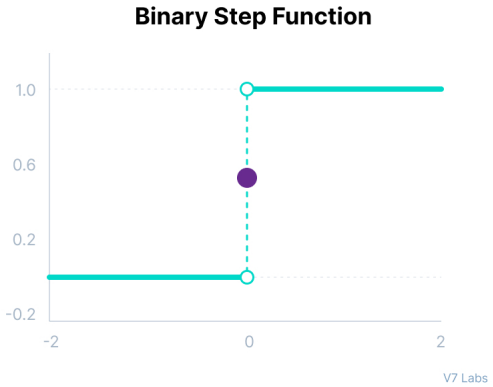
Mathematically it can be represented as:

$$ReLU$$

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

Binary Step Function

Binary step function depends on a threshold value that decides whether a neuron should be activated or not.
The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.



Mathematically it can be represented as:

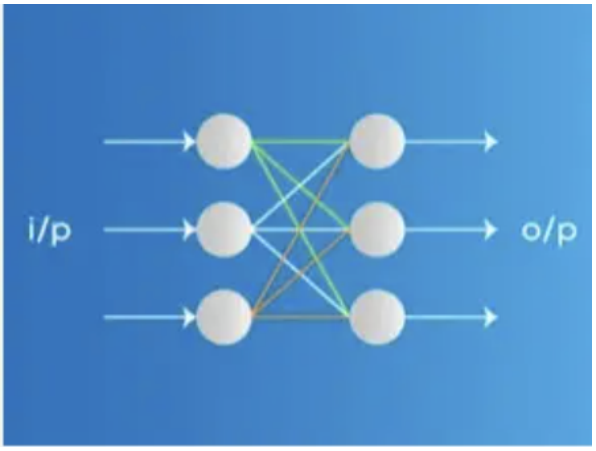
$$Binary\ step$$

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

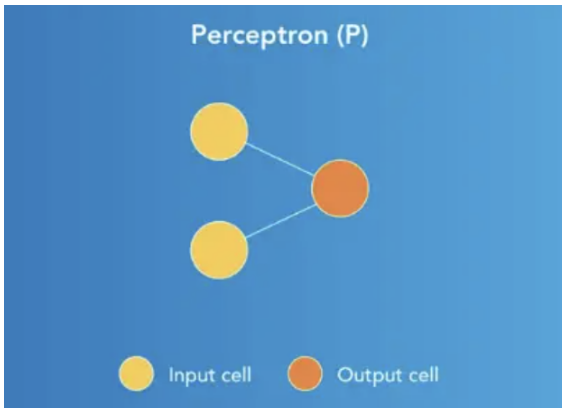
5. Explain the working of back propagation neural networks with neat architecture and diagrams.
Ans) Same as 3.

6. What are the different types of neural networks explained in brief.
Ans) Neural networks, a subset of machine learning and at the core of deep learning algorithms. Their structure and nomenclature are modeled after the human brain, mirroring the communication between organic neurons. Computers can use this to build an adaptive system that helps them continuously improve by learning from their failure.
Types of neural networks are as follows:

1. **Feed Forward Neural Network**
   Feedforward neural networks are among the most basic types of neural networks. Information is passed through several input nodes in one direction until it reaches the output node. The network may or may not include hidden node layers, which helps to explain how it functions.

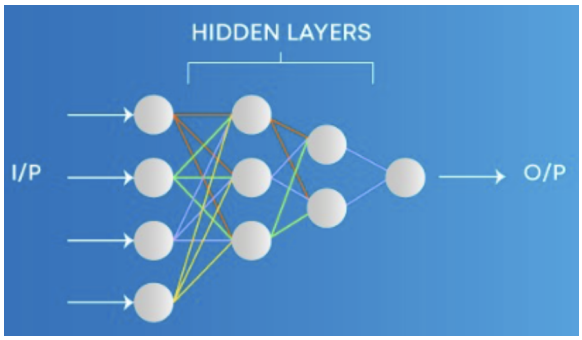


2. **Single-layer Perceptron Network:**
   Layers of connected nodes make up a neural network. Every node is a perceptron, which resembles a multiple linear regression. The signal obtained by multiple linear regression is fed into a non-linear activation function via the perceptron.



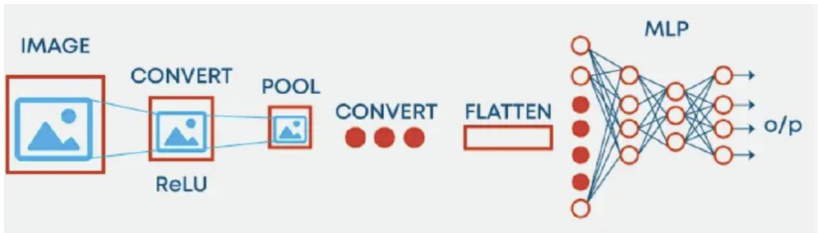
3. **Multilayer Perceptron Network:**

A multilayer perceptron is a fully convolutional network that creates a collection of outputs from a set of inputs. A directed graph connecting the input and output layers of an MLP is made up of multiple layers of input nodes.



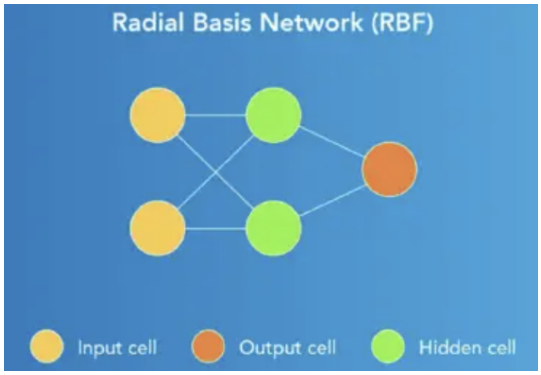
4. **Convolutional Neural Network**

The neurons in a convolution neural network are arranged in three dimensions rather than the typical two-dimensional array. The convolutional layer refers to the top layer. Each neuron in the convolutional layer processes only a small portion of the visual field. Like a filter, input features are gathered in batches.



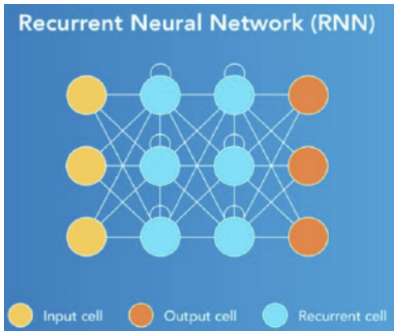
5. **Radial Basis Functional Neural Network**

A Radial Basis Function Network comprises an input vector, an output layer with one node for each category, and a layer of RBF neurons. The classification process involves comparing the input to examples from the training set, where each neuron has a prototype stored.



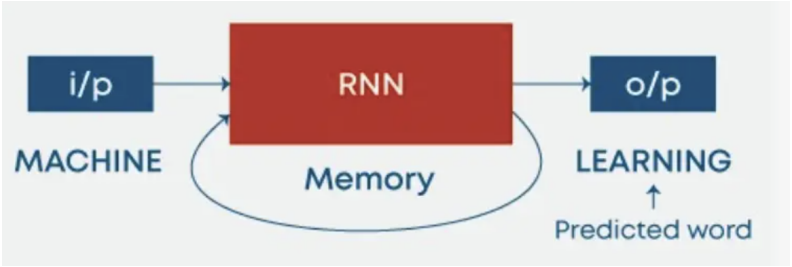
6. Recurrent Neural Network

Recurrent neural networks are constructed to comprehend temporal or sequential data. RNNs improve their predictions by using additional data points in a sequence. To modify the output, they take in input and reuse the activations of earlier or later nodes in the sequence.



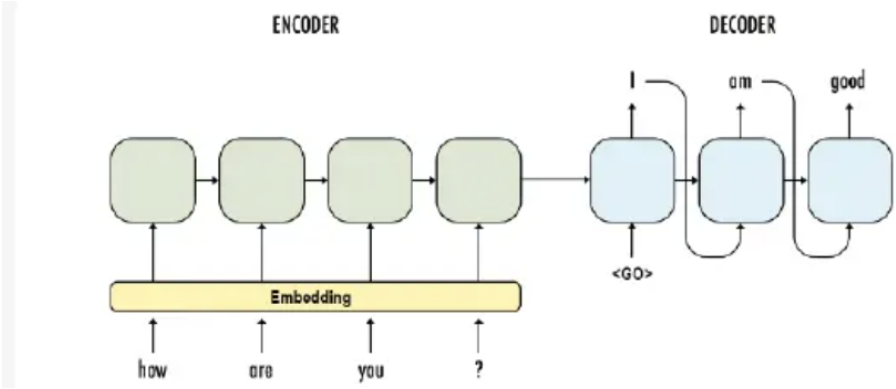
7. LSTM: Long Short-Term Memory

LSTM networks introduce a memory cell. They can handle data that has memory gaps. The time delay is a factor that may be taken into account when using RNNs. However, LSTMs should be used if our RNN fails when we have a lot of relevant data and want to extract important information from it.



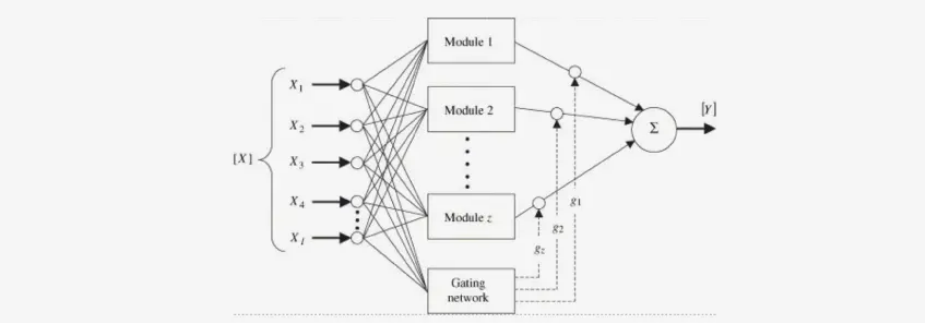
8. Sequence to Sequence Models

Two Recurrent Neural Networks create a sequence-to-sequence model. In this case, a decoder processes the output while an encoder processes the input. Working simultaneously, the encoder and decoder can use the same parameter or a different one.

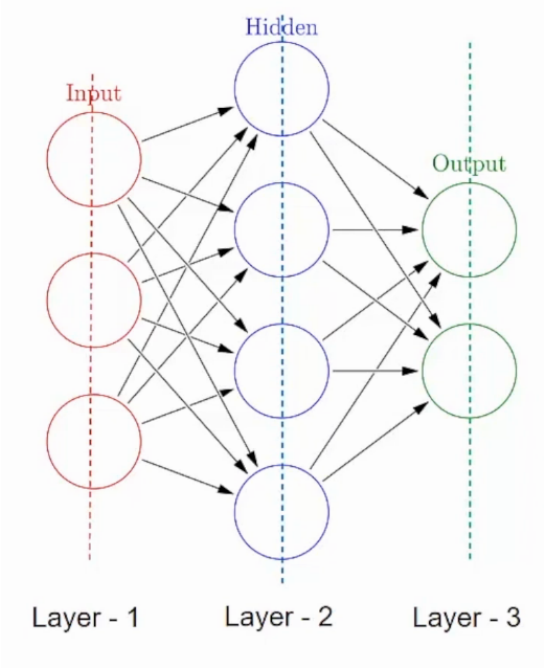


9. Modular Neural Network

A modular neural network consists of several distinct networks that each carry out a specific task. Throughout the calculation process, there isn't much communication or interaction between the various networks. They each contribute separately to the outcome.



7. Explain different layers in the neural network.
Ans)



A neural network is made up of vertically stacked components called Layers. Each
The dotted line in the image represents a layer. There are three types of layers in a NNInput Layer–

First is the input layer- This layer will accept the data and pass it to the rest of the network.

Hidden Layer– The second type of layer is called the hidden layer. Hidden layers are either one or more in number for a neural network. In the above case, the number is 1. Hidden layers are the ones that are actually responsible for the excellent performance and complexity of neural networks. They perform multiple functions at the same time such as data transformation, automatic feature creation, etc.

Output layer– The last type of layer is the output layer. The output layer holds the result or the output of the problem. Raw images get passed to the input layer and we receive output in the output layer.

8. Describe how PCA is carried out to reduce the dimensionality of the dataset.
Ans) Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data points much easier and faster for machine learning algorithms without extraneous variables to process.
PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are image processing, movie recommendation systems, and optimizing the power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable.
Steps for PCA algorithm

- Getting the dataset
- Representing data into a structure
- Standardizing the data
- Calculating the Covariance of Z
- Calculating the Eigen Values and Eigen Vectors
- Sorting the Eigen Vectors
- Calculating the new features Or Principal Components
- Remove fewer or unimportant features from the new dataset.

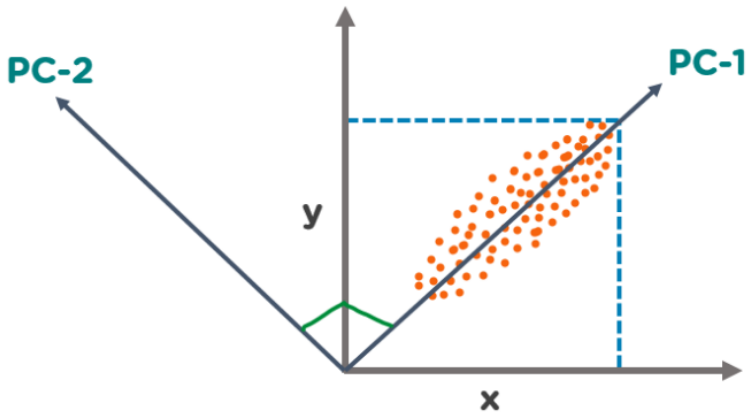
9. What are the applications of PCA?
Ans)

1. **Image Compression:** PCA can be used to compress high-resolution images into smaller sizes without losing much information. It achieves this by reducing the number of dimensions in the image data.
2. **Genetics:** PCA is used in genetics to analyze and visualize genomic data. It can help identify patterns of genetic variation and structure in large datasets.
3. **Finance:** In finance, PCA can be used to identify the underlying factors that contribute to the risk and return of a portfolio of assets. It can also help in the analysis of financial time series data.
4. **Marketing:** PCA can be used in market research to identify the key factors that influence customer behavior. This can help in segmenting customers based on their preferences and needs.

5. **Signal Processing:** PCA is used in signal processing to extract features from complex signals. It can help in identifying patterns in speech, audio, and other signals.
6. **Computer Vision:** PCA can be used in computer vision to reduce the dimensionality of image data. This can help in the recognition and classification of objects in images.
7. **Climate Science:** PCA is used in climate science to analyze and visualize climate data. It can help in identifying patterns of temperature and rainfall variation over time.
8. **Social Sciences:** PCA is used in social sciences to analyze survey data. It can help in identifying the underlying factors that contribute to attitudes and opinions of individuals.

10. Explain PCA in brief.

Ans) The Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data. It increases interpretability yet, at the same time, it minimizes information loss. It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D. PCA helps in finding a sequence of linear combinations of variables.



In the above figure, we have several points plotted on a 2-D plane. There are two principal components. PC1 is the primary principal component that explains the maximum variance in the data. PC2 is another principal component that is orthogonal to PC1.

PCA works by considering the variance of each attribute because the high attribute shows the good split between the classes, and hence it reduces the dimensionality. Some real-world applications of PCA are image processing, movie recommendation systems, and optimizing the power allocation in various communication channels. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

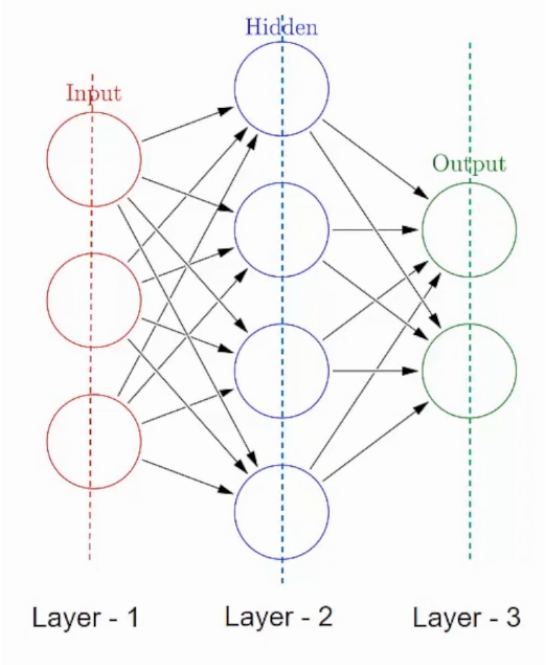
Steps for PCA algorithm

- Getting the dataset
- Representing data into a structure
- Standardizing the data
- Calculating the Covariance of Z
- Calculating the Eigen Values and Eigen Vectors
- Sorting the Eigen Vectors
- Calculating the new features Or Principal Components

- Remove fewer or unimportant features from the new dataset.

11. Explain Neural Network architecture.

Ans) Neural Networks are complex structures made of artificial neurons that can take in multiple inputs to produce a single output.The architecture of neural networks is made up of input, output, and hidden layers. Neural networks themselves, or artificial neural networks (ANNs), are a subset of machine learning designed to mimic the processing power of a human brain.



Each dotted line in the image represents a layer. There are three types of layers in a NNInput Layer–

First is the input layer- This layer will accept the data and pass it to the rest of the network.

Hidden Layer– The second type of layer is called the hidden layer. Hidden layers are either one or more in number for a neural network. In the above case, the number is 1. Hidden layers are the ones that are actually responsible for the excellent performance and complexity of neural networks. They perform multiple functions at the same time such as data transformation, automatic feature creation, etc.

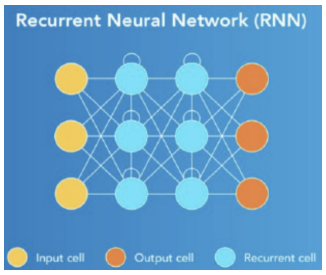
Output layer– The last type of layer is the output layer. The output layer holds the result or the output of the problem. Raw images get passed to the input layer and we receive output in the output layer.

Types of Neural Network Architectures

Recurrent neural networks

Recurrent neural networks are constructed to comprehend temporal or sequential data. RNNs improve their predictions by using additional data points in a sequence. To modify

the output, they take in input and reuse the activations of earlier or later nodes in the sequence. - FARHAT

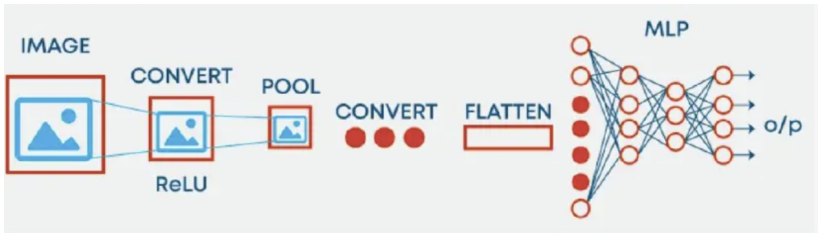


Recurrent neural networks (RNNs) remember previously learned predictions to help make future predictions with accuracy.

- Long short term memory network (LSTM) - LSTM adds extra structures, or gates, to an RNN to improve memory capabilities.
- Echo state network (ESN) - A type of RNN hidden layers that are sparsely connected.

Convolutional neural networks

The neurons in a convolution neural network are arranged in three dimensions rather than the typical two-dimensional array. The convolutional layer refers to the top layer. Each neuron in the convolutional layer processes only a small portion of the visual field. Like a filter, input features are gathered in batches. - FARHAT



Convolutional neural networks (CNNs) are a type of feed-forward network that are used for image analysis and language processing. There are hidden convolutional layers that form ConvNets and detect patterns. CNNs use features such as edges, shapes, and textures to detect patterns. Examples of CNNs include:

- AlexNet - Contains multiple convolutional layers designed for image recognition.
- Visual geometry group (VGG) - VGG is similar to AlexNet, but has more layers of narrow convolutions.
- Capsule networks - Contain nested capsules (groups of neurons) to create a more powerful CNN.

Generative adversarial networks

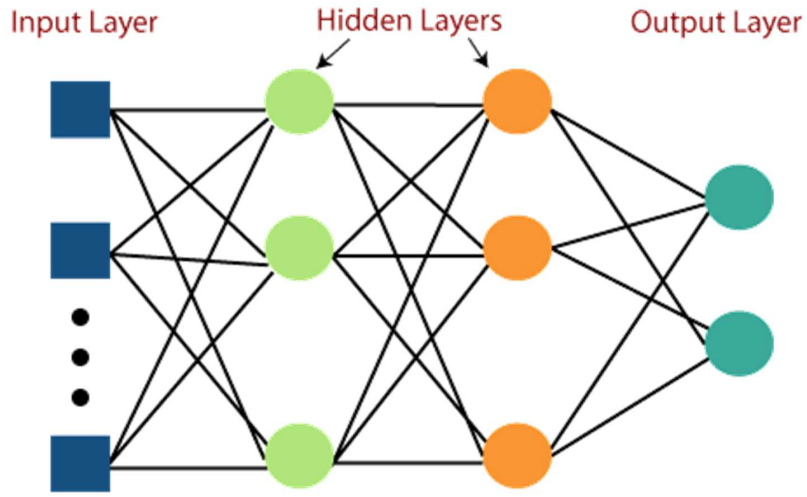
Generative adversarial networks (GAN) are a type of unsupervised learning where data is generated from patterns that were discovered from the input data.
GANs are used to help predict what the next frame in a video might be, text-to-image generation, or image-to-image translation.

Transformer neural networks

Unlike RNNs, transformer neural networks do not have a concept of timestamps. This enables them to pass through multiple inputs at once, making them a more efficient way to process data.

12. Explain a multilayer neural network.

Ans) A multi-layer neural network contains more than one layer of artificial neurons or nodes. They differ widely in design. It is important to note that while single-layer neural networks were useful early in the evolution of AI, the vast majority of networks used today have a multi-layer model.



Multi-layer neural networks can be set up in numerous ways. Typically, they have at least one input layer, which sends weighted inputs to a series of hidden layers, and an output layer at the end. These more sophisticated setups are also associated with nonlinear builds using sigmoids and other functions to direct the firing or activation of artificial neurons.
Convolutional neural networks (CNNs), so useful for image processing and computer vision, as well as recurrent neural networks, deep networks and deep belief systems are all examples of multi-layer neural networks. CNNs, for example, can have dozens of layers that work sequentially on an image. A typical learning algorithm for Multi-Layer networks is also called back propagation's algorithm.

13. What is dimensionality reduction?

Ans) In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

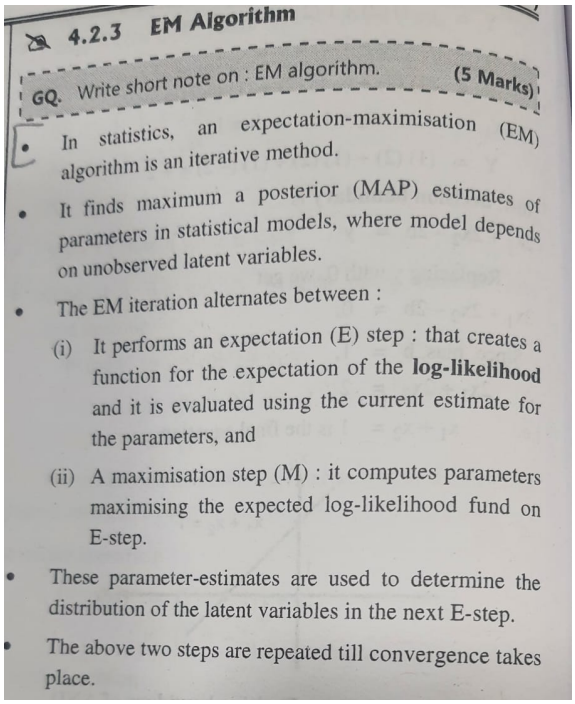
An intuitive example of dimensionality reduction can be discussed through a simple email classification problem, where we need to classify whether the email is spam or not. This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the email uses a template, etc. However, some of these features may overlap.

In another condition, a classification problem that relies on both humidity and rainfall can be collapsed into just one underlying feature, since both of the aforementioned are correlated to a high degree. Hence, we can reduce the number of features in such problems. There are two components of dimensionality reduction:

- Feature selection: In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:
    1. Filter
    2. Wrapper
    3. Embedded
- Feature extraction: This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

14. Explain EM algorithm.

Ans)



4.2.3 EM Algorithm

GQ. Write short note on : EM algorithm. (5 Marks)

- In statistics, an expectation-maximisation (EM) algorithm is an iterative method.
- It finds maximum a posterior (MAP) estimates of parameters in statistical models, where model depends on unobserved latent variables.
- The EM iteration alternates between :
    (i) It performs an expectation (E) step : that creates a function for the expectation of the **log-likelihood** and it is evaluated using the current estimate for the parameters, and
    (ii) A maximisation step (M) : it computes parameters maximising the expected log-likelihood fund on E-step.
- These parameter-estimates are used to determine the distribution of the latent variables in the next E-step.
- The above two steps are repeated till convergence takes place.

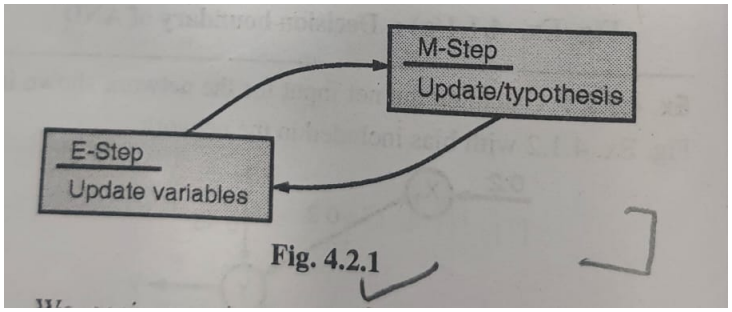


Fig. 4.2.1

In the E-step, the algorithm computes the expected value of the missing data, given the current estimates of the model parameters. This step involves computing the posterior probabilities of the missing data, given the observed data and the current parameter estimates.

In the M-step, the algorithm updates the model parameters to maximize the likelihood of the observed data, based on the expected values computed in the E-step. This step involves finding the values of the parameters that maximize the log-likelihood of the observed data, given the expected values of the missing data.

The algorithm iterates between the E-step and the M-step until convergence, that is, until the change in the estimates of the parameters is below a certain threshold.

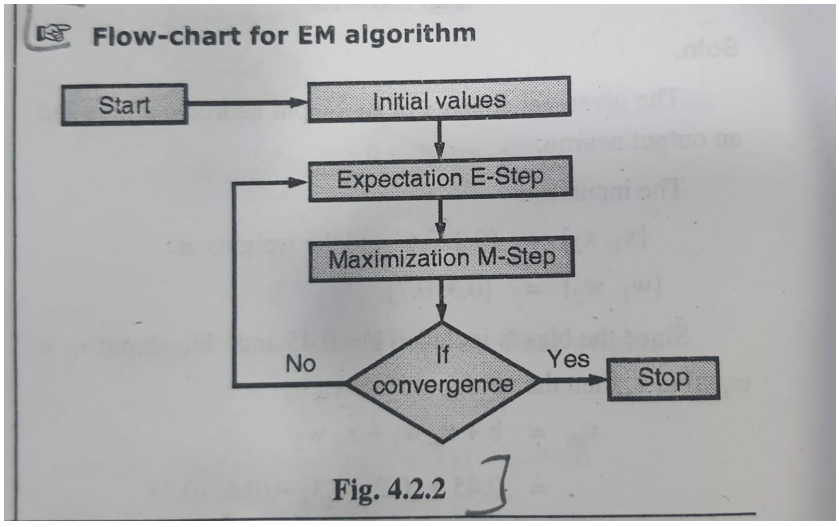


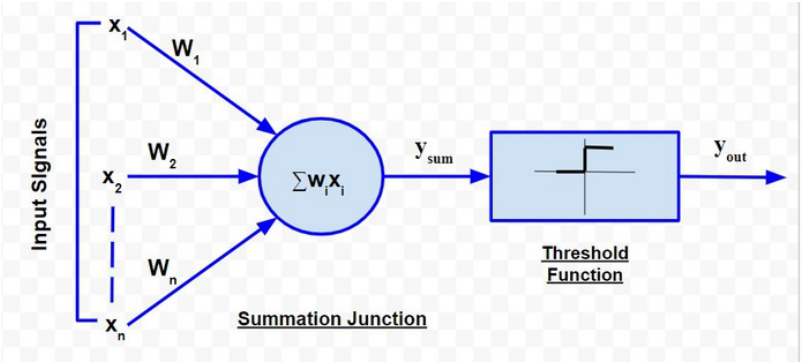
Fig. 4.2.2

15. Explain MC-culloch pitts model.

Ans) The McCulloch-Pitts neural model, which was the earliest ANN model, has only two types of inputs — Excitatory and Inhibitory.

The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch-Pitts neuron could be either 0 or 1. It has a threshold function as an activation function.

So, the output signal y_{out} is 1 if the input y_{sum} is greater than or equal to a given threshold value, else 0. The diagrammatic representation of the model is as follows:



Simple McCulloch-Pitts neurons can be used to design logical operations. For that purpose, the connection weights need to be correctly decided along with the threshold function (rather than the threshold value of the activation function). For better understanding purpose, let me consider an example:
John carries an umbrella if it is sunny or if it is raining. There are four given situations. I need to decide when John will carry the umbrella. The situations are as follows:

- First scenario: It is not raining, nor it is sunny (0, 0)
- Second scenario: It is not raining, but it is sunny (0, 1)
- Third scenario: It is raining, and it is not sunny (1, 0)
- Fourth scenario: It is raining as well as it is sunny (1, 1)

To analyze the situations using the McCulloch-Pitts neural model, I can consider the input signals as follows:

- X1: Is it raining?
- X2 : Is it sunny?

Truth Table for this case will be:

Situation	x1	x2	ysum	yout
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

So, the value of both scenarios can be either 0 or 1. We can use the value of both weights X1 and X2 as 1 and a threshold function as 1.
The truth table built with respect to the problem is depicted above. From the truth table, I can conclude that in situations where the value of y_{out} is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.

16. What are the advantages and disadvantages of EM algorithms.

Ans)

4.2.4 Advantages and Disadvantages of EM Algorithm

Advantages

(i) It is guaranteed that likelihood will increase with each iteration.

(ii) The E-step and M-step are quite easy to implement for many problems.

(iii) Solutions to the M-steps exist in the closed form.

Disadvantages

(i) It has slow convergence

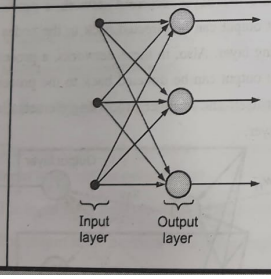
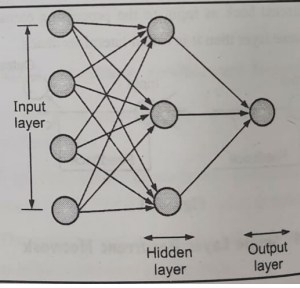
(ii) It makes convergence to the local optima only.

(iii) It requires both the probabilities, forward and backward (note that, numerical optimisation requires only forward probability).

17. Difference between single layer neural network and multi-layer feed forward neural network.

Ans)

Forward Neural Network

Sr. No.	Single layer feed forward Neural Network	Multi layer feed forward Neural Network
1.	Formation of layer is done by **processing** and **combining** elements with other processing elements.	Multi layer is formed by **interconnection** of **several** layers.
2.	There is **linking** between **input** and **output** layer.	Between input and output layers, there are **multiple layers**, which are known as **hidden layers.**
3.	Inputs are connected to the **processing** nodes with various **weights** resulting series of output **one for node**.	Input layers receive input and stores input signal and output layer generates output.
4.	There is **no hidden layer.**	There are **several hidden layers** in a network.
5.	In certain applications, it is **not efficient.**	More hidden layers create **complexity** of network, but the output is **efficient.**
		

18. Explain features of big data.

Ans) Big data is a term used to describe the massive amounts of data that are being generated at an unprecedented rate. The features of big data can be summarized by the "three Vs" - volume, velocity, and variety. Let's take a closer look at each of these features:

Volume: The volume of data generated by businesses, individuals, and organizations is enormous. With the proliferation of connected devices, social media, and online transactions, the amount of data being generated is growing exponentially. Traditional databases and data processing tools are not equipped to handle this volume of data.

Velocity: Big data is being generated at an incredibly high velocity. Data is being produced in real-time, and it needs to be processed and analyzed quickly. For example, social media feeds generate large volumes of data every second, and businesses need to analyze this data quickly to stay ahead of their competitors.

Variety: Big data is not just about the sheer volume of data but also about the variety of data. Data can come in different forms, such as structured, unstructured, and semistructured. It can include text, images, audio, video, and other types of data. The variety of data makes it difficult to store and process using traditional methods.

Other features of big data include:

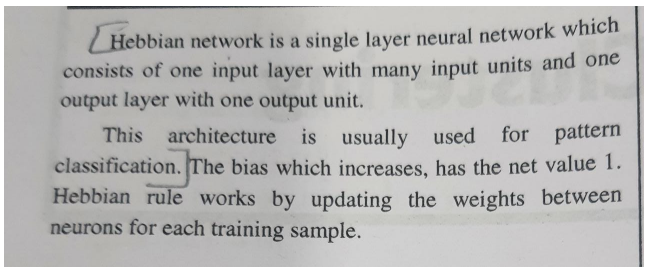
Veracity: The quality of big data can vary greatly. Data can be inaccurate, incomplete, or unreliable. It is important to ensure that the data used for analysis is accurate and reliable.

Value: The ultimate goal of big data is to derive insights and value from the data. By analyzing large volumes of data, businesses can identify patterns and trends that can help them make informed decisions and gain a competitive edge.

Variability: The structure and format of big data can be highly variable, which makes it challenging to process and analyze. For example, social media data can be highly unstructured, making it difficult to analyze using traditional data processing methods.

19. State Hebbian Learning rule.

Ans)



Hebbian network is a single layer neural network which consists of one input layer with many input units and one output layer with one output unit.

This architecture is usually used for pattern classification. The bias which increases, has the net value 1. Hebbian rule works by updating the weights between neurons for each training sample.

4.1.4 Hebbian Learning is Unsupervised

Hebbian learning requires no other information than the activates, such as labels or error signals. It is an unsupervised learning method.

Hebbian learning is not a concrete learning rule, it is a postulate on the fundamental principle of biological learning.

4.1.1 Hebbian Learning Rule Algorithm

(1) Set all weights to zero, $w_i = 0$ for $i = 1$ to n, and bias to zero.

(2) For each input vector, S (input vector) t (target output pair), repeat steps 3-5.

(3) Set activations for input units with the input vector $X_i = S_i$ for $i = 1$ to n.

(4) Set the corresponding output value to the output neuron, i.e. $y = t$.

(5) Update weight and bias by applying Hebb rule for $i = 1$ to n :

w_i (new) = w_i (old) + x_i y

b (new) = b (old) + y

20. What are the applications of EM algorithm.
Ans)

4.2.5 Applications of EM-Algorithm

The latent variable model has several real-life applications in Machine-learning :

(i) Use to calculate the Gaussian density of a function.

(ii) It helps to fill in the missing data during a sample.

(iii) In domains like Natural Language Processing (NLP), computer Vision etc. It finds plenty of applications.

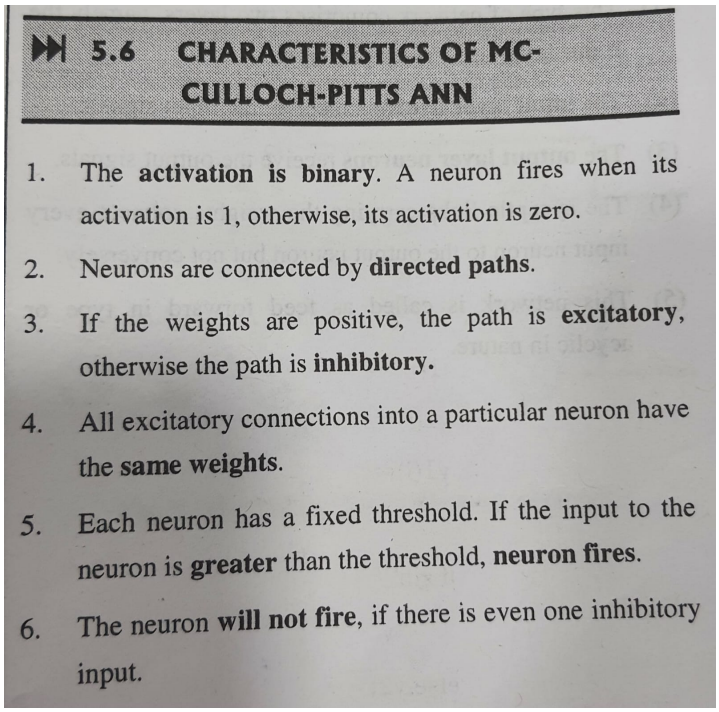
(iv) It is also helpful in image reconstruction in the field of Medicine, and structural Engineering.

(v) Used to estimate the prameters of mixed models like gaussian Mixture Models.

21. Comparison between biological neural network and Artificial neural network.
Ans)

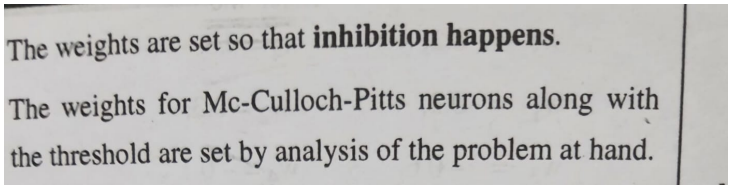
Sr. No.	Characteristics	Artificial Neural Network	Biological (Real) Neural Network
1.	Speed	**Faster** in processing information. Response time is in nanoseconds.	**Slower** in processing information. The response time is in milliseconds.
2.	Processing	**Serial** processing	Massively **parallel** processing.
3.	Size and complexity	**Less** size and complexity. It does not perform complex pattern recognition tasks.	A **highly complex** and dense network of interconnected neurons containing neurons of the order of 1011 with 1015 of interconnections.
4.	Storage	New data with old data can be replaced, i.e. information storage is **replaceable.**	New information or data is added by adjusting the interconnection strength **without destroying old information.**
5.	Anism	There is a **control unit** for controlling computing activities.	**No specific control** mechanism external to the computing task.

22. What are the characteristics of MC-Culloch Pitts ANN.
Ans)



⏭ 5.6 CHARACTERISTICS OF MC-CULLOCH-PITTS ANN

1. The **activation is binary**. A neuron fires when its activation is 1, otherwise, its activation is zero.
2. Neurons are connected by **directed paths**.
3. If the weights are positive, the path is **excitatory**, otherwise the path is **inhibitory.**
4. All excitatory connections into a particular neuron have the **same weights**.
5. Each neuron has a fixed threshold. If the input to the neuron is **greater** than the threshold, **neuron fires**.
6. The neuron **will not fire**, if there is even one inhibitory input.



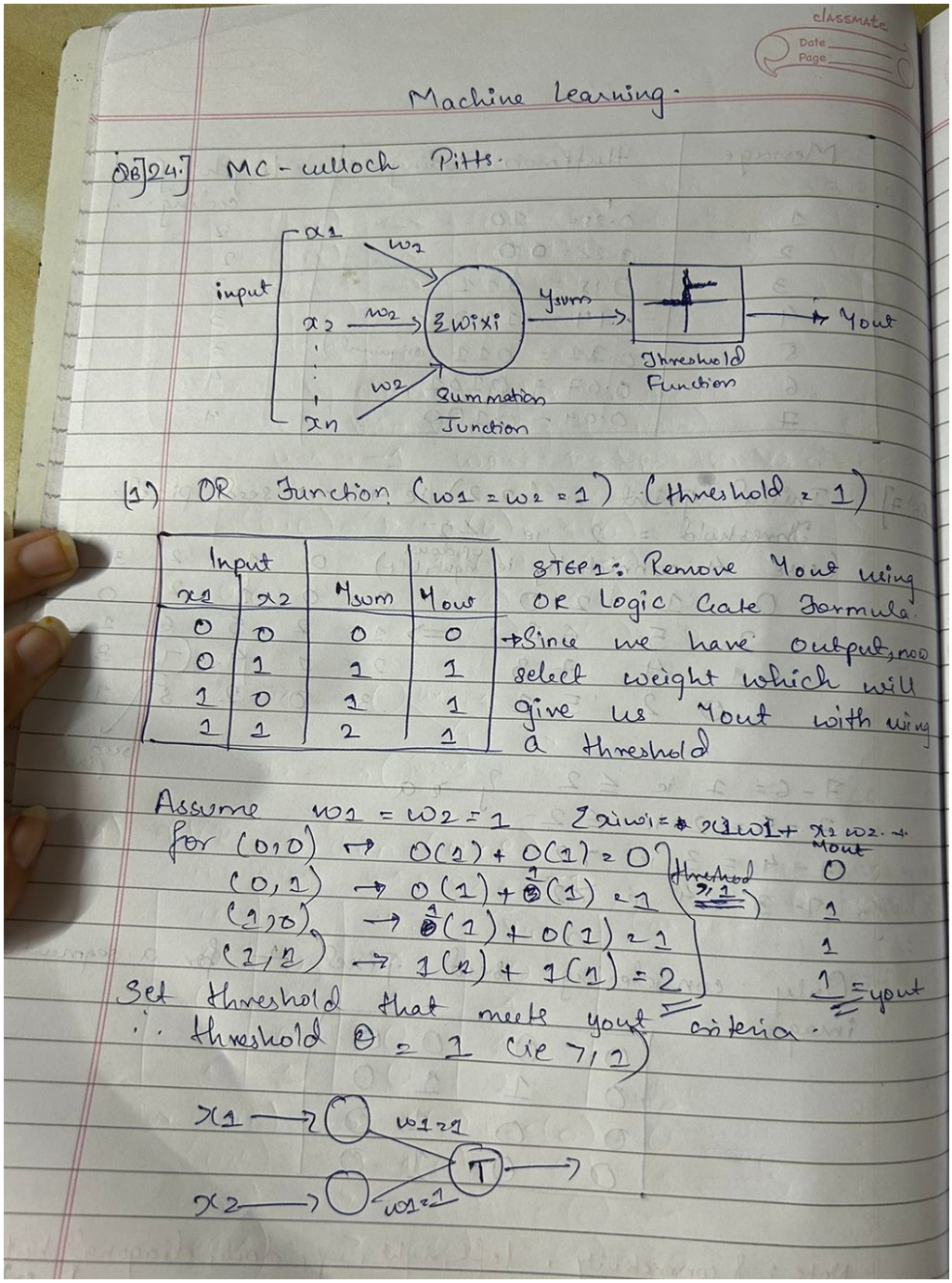
The weights are set so that **inhibition happens**.

The weights for Mc-Culloch-Pitts neurons along with the threshold are set by analysis of the problem at hand.

23. Why dimensionality reduction is an important step in Machine Learning.
Ans) same as 13.

24. Apply Mc-Culloch Pitts model to implement OR function, AND function, NAND, NOR, XOR (use binary data)

Ans)



Machine Learning.

[24.] MC-culloch Pitts.

(1) OR Function ($w_1 = w_2 = 1$) (Threshold = 1)

Input			
x_1	x_2	Ysum	Yout
0	0	0	0
0	1	1	1
1	0	1	1
1	1	2	1

STEP 1: Remove Yout using OR Logic Gate formula.

→ Since we have output, now select weight which will give us Yout with using a threshold

Assume $w_1 = w_2 = 1$ $\quad \Sigma x_i w_i = x_1 w_1 + x_2 w_2 \ldots$

for (0,0) → 0(1) + 0(1) = 0 → 0

(0,1) → 0(1) + 1(1) = 1 → 1

(1,0) → 1(1) + 0(1) = 1 → 1

(1,1) → 1(1) + 1(1) = 2 → 1 = Yout

Threshold ≥ 1, 1

Set threshold that meets Yout criteria.

∴ Threshold = 1 (ie 1, 1)

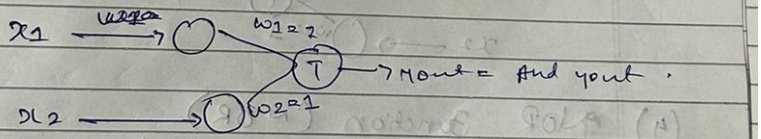
(2) AND function (similarly)

x1	x2	Ysum	Yout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

∴ Considering $w_1 = w_2 = 1$
since AND function produces binary output that ~~it~~ is 1 if both inputs are 1, so we will be considering threshold = 2 (ie 1, 2)

∴ for (0,0) → 0(1) + 0(1) = 0
(0,1) → 0(1) + 1(1) = 1
(1,0) → 1(1) + 1(0) = 1
(1,1) → 1(1) + 1(1) = 2

Ysum	Yout
threshold. ≥ 2	0
	0 = Yout
	0
	1



(3) NAND function (~AND)
$w_1 = w_2 = 1$

x1	x2	Ysum	Yout
0	0	0	1
0	1	1	1
1	0	1	1
1	1	2	0

For $w_1 = w_2 = 1$

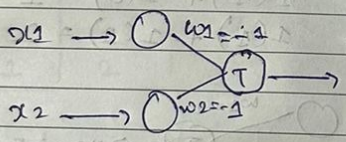
		Ysum	Yout
for (0,0)	→ 1(0) + 1(0) = 0	threshold ≥ 0	1
(0,1)	→ 1(0) + 1(1) = 1	⟹	1
(1,0)	→ 1(1) + 1(0) = 1		1
(1,1)	→ 1(1) + 1(1) = 2		1

does not give you output

Considering other weights as it does not satisfy that criteria. Also changing threshold.

∴ $w_1 = -1$ $w_2 = -1$ and threshold $= -1$

	Ysum	Yout
for (0,0) →	-1(0) + -1(0) = 0 threshold ≥ -1	1
	-1(0) + -1(1) = -1 ⟹	1
	-1(1) + -1(0) = -1	1
	-1(1) + -1(1) = -2	0 = yout

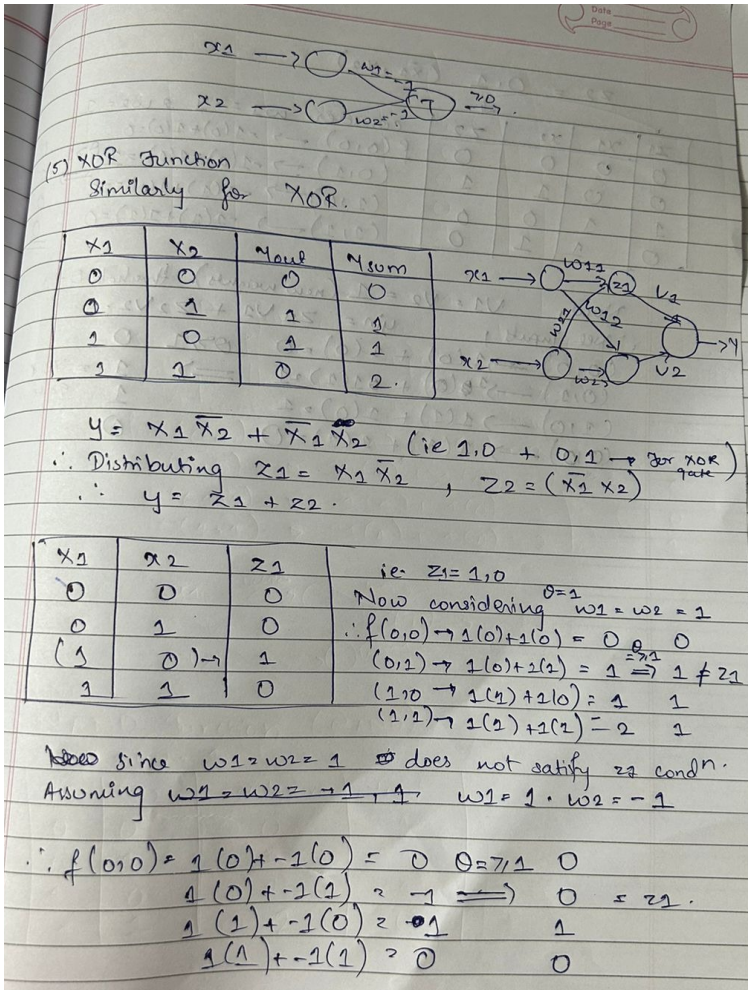


(4) NOR function (~OR)

Considering $w_1 = w_2 = -1$ & threshold $\theta = 0$ (as $w_1 = w_2 = -1$ will not satisfy)

x1	x2	Yout	Ysum
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	2

	Ysum	Yout
for (0,0) →	-1(0) + -1(0) = 0 threshold ≥ 0	1
(0,1) →	-1(0) + -1(1) = -1 ⟹	0
(1,0) →	-1(1) + -1(0) = -1	0
(1,1) →	-1(1) + -1(1) = -2	0



x1	x2	Yout	Ysum
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	2

$y = x_1\bar{x}_2 + \bar{x}_1x_2$ (ie 1,0 + 0,1 → for XOR gate)

∴ Distributing $z_1 = x_1\bar{x}_2$, $z_2 = (\bar{x}_1 x_2)$

∴ $y = z_1 + z_2$.

x1	x2	z1
0	0	0
0	1	0
1	0	1
1	1	0

ie. $z_1 = 1,0$

Now considering $\theta = 1$ $w_1 = w_2 = 1$

∴ f(0,0) → 1(0)+1(0) = 0 θ 0

(0,1) → 1(0)+1(1) = 1 ⟹ 1 ≠ z1

(1,0 → 1(1)+1(0) = 1 1

(1,1) → 1(1)+1(1) = 2 1

Since $w_1 = w_2 = 1$ does not satisfy z1 condn.

Assuming $w_1 = w_2 = -1, 1$ $w_1 = 1 \cdot w_2 = -1$

∴ f(0,0) = 1(0)+-1(0) = 0 $\theta = 1$ 0

1(0)+-1(1) = -1 ⟹ 0 = z1.

1(1)+-1(0) = 1 1

1(1)+-1(1) = 0 0

$z_2 = 0, 1$ $(\bar{x}_1, x_2)$

z1	x1	x2	z2
0	0	0	0
0	0	1	1
1	1	0	0
0	1	1	0

consider $w_1 = -1$ & $w_2 = 1$

f(0,0) → -1(0)+1(0)=0

(0,1) → -1(0)+1(1) = 1 $\theta = 1$ 0

(1,0) → -1(1)+1(0)=-1 ⟹ 1

(1,1) → -1(1)+1(1)=0 0

0

Now, $v_1 = v_2 = 1$ (new weights) threshold $\theta = 1$

∴ for inputs, $y_{in} = z_1v_1 + z_2v_2$.

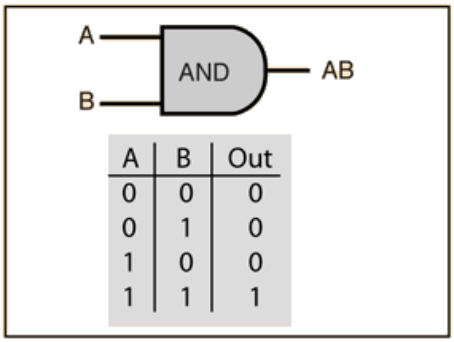
(0,0) → 1(0) + 1(0) = 0 0?1 0

(0,1) → 1(0) + 1(1) = 1 ⟹ 1 = XOR

(1,0) → 1(1) + 1(0) = 1 1 gate.

(1,0) → 1(1) + 1(1) = 0 0

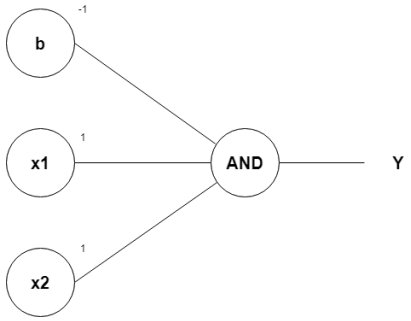
25. Use perceptron to implement AND, OR, NAND, NOR, XOR logic function.
Ans) AND Gate
From our knowledge of logic gates, we know that an AND logic table is given by the diagram below



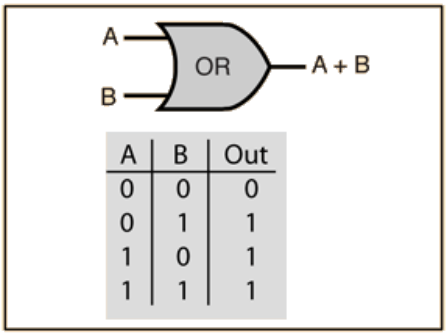
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- From w1*x1+w2*x2+b, initializing w1, w2, as 1 and b as –1, we get;

x1(1)+x2(1)–1
1 row:0+0–1 = –1
2 row :0+1–1 = 0
3 row: 1+0-1=0
4 row:1+1–1 = 1
From the Perceptron rule, if Wx+b≤0, then y`=0 else, y`=1. The rows are correct, as the output is 0 for 1st to the 3rd row & 1 for the 4th row respectively for the AND gate.
Therefore, we can conclude that the model to achieve an AND gate, using the Perceptron algorithm is;



OR Gate

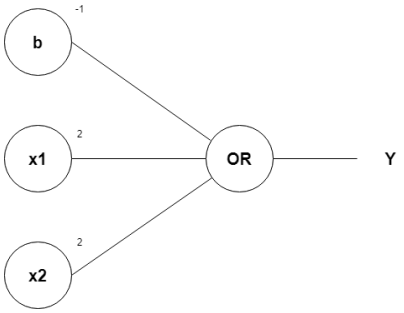


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

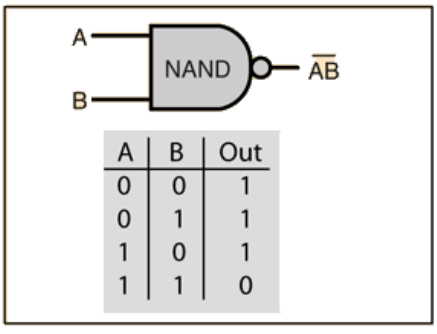
- From w1x1+w2x2+b, initializing w1, w2, as 1 and b as –1, we get;

1 row:0+0–1 = –1
From the Perceptron rule, if Wx+b≤0, then y`=0. Therefore, this row is correct.
2 row :0+1–1 = 0
From the Perceptron rule, if Wx+b <= 0, then y`=0. Therefore, this row is incorrect.
So we want values that will make inputs x1=0 and x2=1 give y` a value of 1. If we change w2 to 2, we have;
Therefore, 2 row: 0+2–1 = 1
3 row: 1+0-1=0
From the Perceptron rule, if Wx+b <= 0, then y`=0. Therefore, this row is incorrect.
So we want values that will make inputs x1=0 and x2=1 give y` a value of 1. If we change w1 to 2, we have;
Therefore, 3 row: 2+0-1
now, since w1=2 and w2=2 from above changes,
4 row:2+2–1 = 3
From the Perceptron rule, if Wx+b≤0, then y`=0 else, y`=1. The rows are correct, as the output is 0 for 1st to the 3rd row & 1 for the 4th row respectively for the OR gate.

Therefore, we can conclude that the model to achieve an OR gate, using the Perceptron algorithm is;



NAND Gate

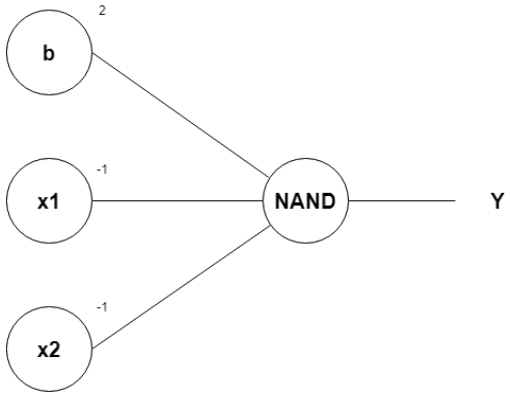


A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

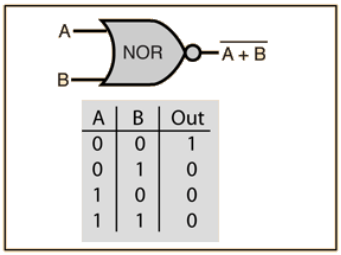
- From w1x1+w2x2+b, initializing w1, w2, as -1 and b as 2, we get;

1 row: 0-0+2 = 2
2 row :0-1+2 = 1
3 row: -1-0+2=1
4 row:-1-1+2 = 0
From the Perceptron rule, if Wx+b≤0, then y`=0 else, y`=1. The rows are correct, as the output is 1 for 1st to the 3rd row & 0 for the 4th row respectively for the NAND gate.

Therefore, we can conclude that the model to achieve an OR gate, using the Perceptron algorithm is;



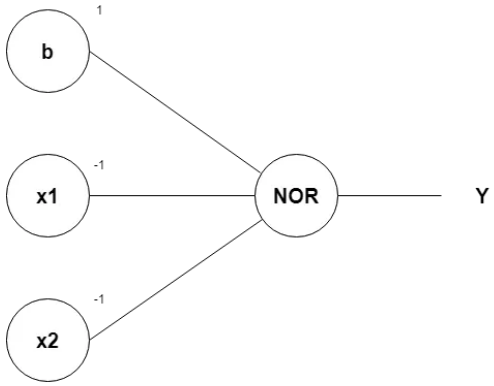
NOR GATE:



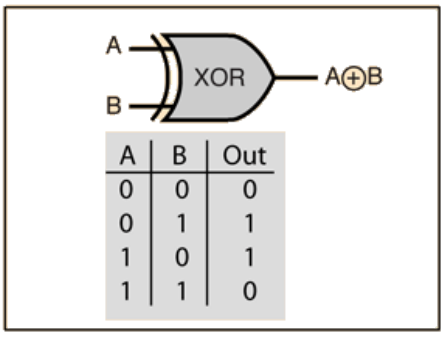
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

- From w1x1+w2x2+b, initializing w1 and w2 as -1, and b as 1, we get;

1 row: 0-0+1 = 2
2 row :0-1+1 = 1
3 row: -1+0+1=0
4 row:-1-1+1 = -1



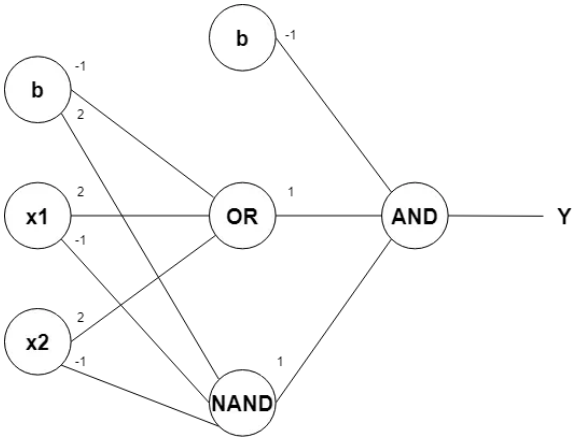
XOR Gate



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

XOR Gate
The boolean representation of an XOR gate is;
x1x`2 + x`1x2
We first simplify the boolean expression
x`1x2 + x1x`2 + x`1x1 + x`2x2
x1(x`1 + x`2) + x2(x`1 + x`2)
(x1 + x2)(x`1 + x`2)
(x1 + x2)(x1x2)`
From the simplified expression, we can say that the XOR gate consists of an OR gate (x1 + x2), a NAND gate (-x1-x2+2) and an AND gate (x1+x2–1).
This means we will have to combine 2 perceptrons:

- OR (2x1+2x2–1)
- NAND (-x1-x2+2)
- AND (x1+x2–1)



26. Calculate the output of the neuron Y for the network given below. Use binary and bipolar sigmoid function.

	Bias	X1	X2
Input	1	0.7	0.8
Weights	0.9	0.2	0.3

Ans)

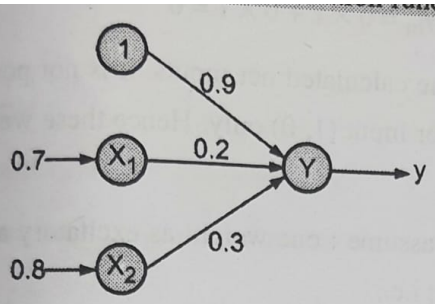


Fig. Ex. 5.5.3

☑ **Soln. :**

▸ **Step (I) :**

(i) The given network has two input neurons with bias and one output neuron.

These form a single-layer network.

(ii) The inputs are : $x_1 = 0.7$; $x_2 = 0.8$ and weights are : $w_1 = 0.2$; $w_2 = 0.3$ and bias = b = 0.9 (its input is always 1)

▸ **Step (II) :** Now the net input to the output neuron is

$$y_{in} = b + \sum_{i=1}^{2} x_i w_i ;$$

(∵ only 2 input neurons are mentioned)

$$= b + (x_1 w_1 + x_2 w_2)$$

$$= 0.9 + [(0.7)(0.2) + (0.8)(0.3)]$$

$$= 0.9 + 0.14 + 0.24 = 1.28$$

Step (III) : (i) Binary sigmoidal function is given by y

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

$$= \frac{1}{1 + e^{-1.28}} = \frac{1}{1 + 0.278} = 0.7824$$

(ii) Bipolar sigmoidal activation function is,

$$y = f(y_{in}) = \left[\frac{2}{1 + e^{-y_{in}}}\right] - 1$$

$$= \left[\frac{2}{1 + e^{-1.28}}\right] - 1 = 0.5649$$

27. Calculate the output of the neuron Y for the network given below. Use binary and bipolar sigmoid function.

	Bias	X1	X2	X3
Input	1	0.8	0.6	0.4
Weights	0.35	0.1	0.3	-0.2

Soln. :

Step (I) : (i) The given network has three input neurons with bias and one output neuron.

These form a single – layer network.

(ii) The inputs are : $x_1 = 0.8$; $x_2 = 0.6$; $x_3 = 0.4$

and weights are $w_1 = 0.1$; $w_2 = 0.3$; $w_3 = -0.2$

and bias is b = 0.35 (its input is always 1)

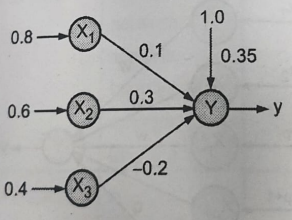


Fig. Ex. 5.5.2

Step (II) : Now the net input to the output neuron is

$$y_{in} = b + \sum_{i=1}^{3} x_i w_i;$$

(∵ only 3 input neurons are given)

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$= 0.35 + (0.8)(0.1) + (0.6)(0.3) + (0.4)(-0.2)$$

$$= 0.35 + 0.08 + 0.18 - 0.08 = 0.53$$

Step (III) : (i) Binary sigmoidal activation function given by,

$$y = f(y_{in}) = \left(\frac{1}{1 + e^{-y_{in}}}\right) = \frac{1}{1 + e^{-0.53}} = 0.625$$

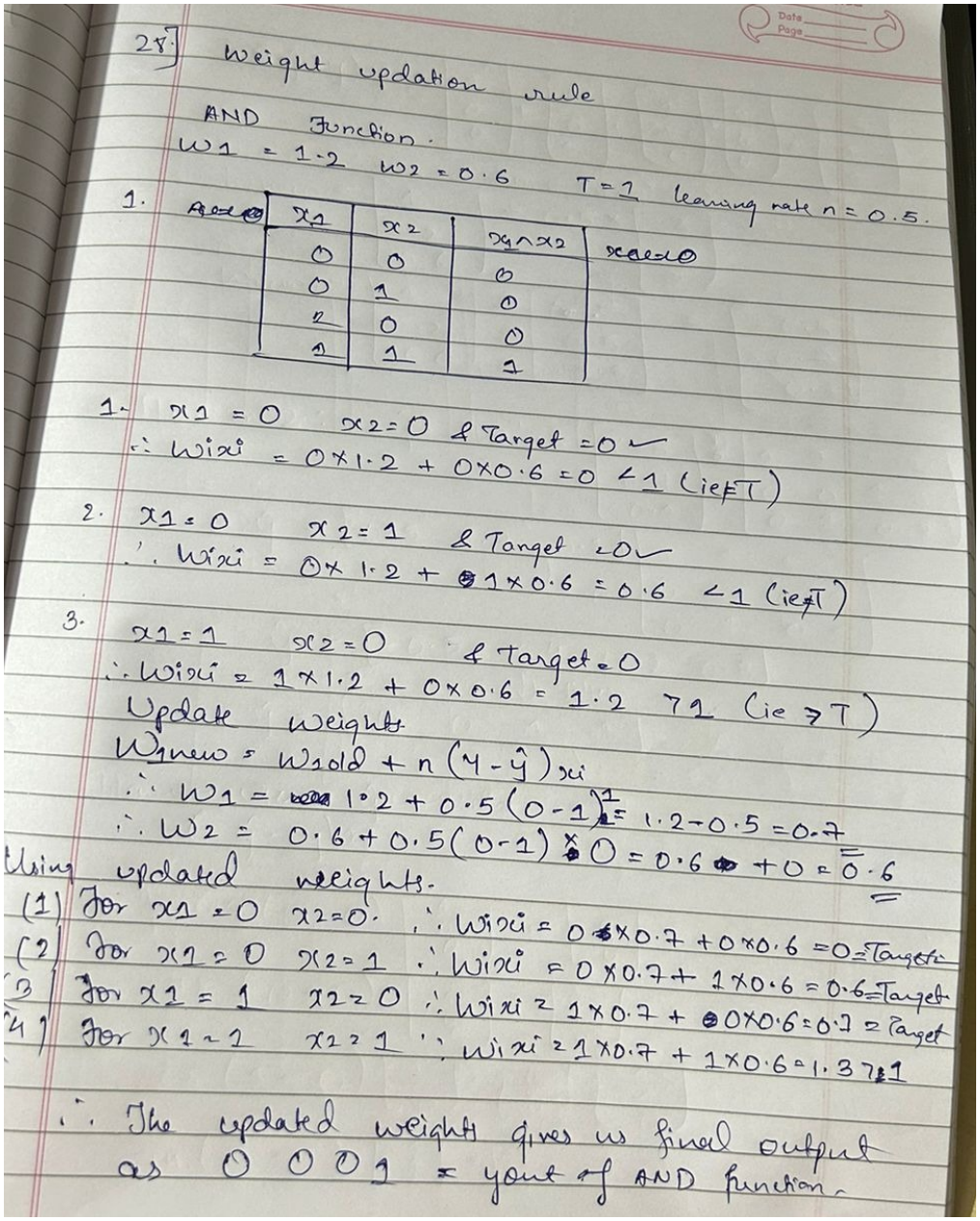
(ii) bipolar sigmoidal activation functions is given by,

$$y = f(y_{in}) = \left(\frac{2}{1 + e^{-y_{in}}}\right) - 1$$

$$= \left(\frac{2}{1 + e^{-0.53}}\right) - 1 = 0.259$$

28. Apply weight updating rule to calculate new weight for AND function w1=1.2, w2= 0.6 and threshold =1, learning rate n=0.5

Ans)



28] Weight updation rule

AND function.

$w_1 = 1.2$ $w_2 = 0.6$ $T = 1$ learning rate $n = 0.5$.

1.

x_1	x_2	$x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

1. $x_1 = 0$ $x_2 = 0$ & Target = 0 ✓
   $\therefore w_i x_i = 0 \times 1.2 + 0 \times 0.6 = 0 < 1$ (ie $\neq T$)
2. $x_1 = 0$ $x_2 = 1$ & Target = 0 ✓
   $\therefore w_i x_i = 0 \times 1.2 + 1 \times 0.6 = 0.6 < 1$ (ie $\neq T$)
3. $x_1 = 1$ $x_2 = 0$ & Target = 0
   $\therefore w_i x_i = 1 \times 1.2 + 0 \times 0.6 = 1.2 \geq 1$ (ie $\geq T$)

Update weights

$W_{new} = W_{old} + n(y - \hat{y}) x_i$

$\therefore w_1 = 1.2 + 0.5(0-1)1 = 1.2 - 0.5 = 0.7$

$\therefore w_2 = 0.6 + 0.5(0-1) \times 0 = 0.6 + 0 = 0.6$

Using updated weights.

(1) For $x_1 = 0$ $x_2 = 0$ $\therefore w_i x_i = 0 \times 0.7 + 0 \times 0.6 = 0 =$ Target

(2) For $x_1 = 0$ $x_2 = 1$ $\therefore w_i x_i = 0 \times 0.7 + 1 \times 0.6 = 0.6 =$ Target

(3) For $x_1 = 1$ $x_2 = 0$ $\therefore w_i x_i = 1 \times 0.7 + 0 \times 0.6 = 0.7 =$ Target

(4) For $x_1 = 1$ $x_2 = 1$ $\therefore w_i x_i = 1 \times 0.7 + 1 \times 0.6 = 1.3 \geq 1$

$\therefore$ The updated weights gives us final output as 0 0 0 1 = output of AND function.

29. Use Adaline network to train NAND function with bipolar inputs and targets w1=0.2, w2= 0.2 and b=0.2, learning rate n=0.2

Ans) SIMILAR SUM DIFFERENT GATE REFER TO THE TABLE:

▶ **Step (I) :** We prepare truth-table for AND NOT function with bipolar inputs and targets :

For convenience, we choose random values say 0.2 for weights and bias.

Let $w_1 = w_2 = b = 0.2$ and let learning rate $\alpha = 0.2$

▶ **Step (II) :**

Case (1) : Let $x_1 = 1$, $x_2 = 1$, $t = -1$;

$w_1 = w_2 = b = 0.2$, $\alpha = 0.2$

for NAND

Truth-table

x_1	x_2	t
1	1	−1
1	−1	1
−1	1	1
−1	−1	1

(i) The net input is,

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$= 0.2 + 1\,(0.2) + 1(0.2) = 0.6$

$\therefore \quad t - y_{in} = -1 - 0.6 = -1.6$

(ii) We update the weights,

$$w_i\,(new) = w_i\,(old) + \alpha\,(t - y_{in})x_i$$

for i = 1, 2 $\quad w_1\,(new) = w_1\,(old) + \alpha\,(t - y_{in})\,x_1$

$= 0.2 + 0.2\,(-1.6)(1) = -0.12$;

Also, $w_2\,(new) = w_2\,(old) + \alpha\,(t - y_{in})$

$x_2 = 0.2 + 0.2\,(-1.6)(1) = -0.12$

and $b\,(new) = b\,(old) + \alpha\,(t - y_{in})$

$= 0.2 + 0.2\,(-1.6) = -0.12$

$\therefore \quad (t - y_{in}) = -1.6$

$\therefore$ **Error** $= (t - y_{in})^2 = (-1.6)^2 = \mathbf{2.56}$

▶ **Step (III) :** Now, we have

$w_1 = -0.12, \quad w_2 = -0.12, \quad b = -0.12, \quad x_1 = 1,$

$x_2 = -1, t = 1$

$\therefore$ input $y_{in} = b + x_1 w_1 + x_2 w_2$

$= (-0.12) + (1)\,(-0.12) + (-1)\,(-0.12)$

$= -0.12$

$\therefore \quad t - y_{in} = (1) - (-0.12) = 1.12$

$\therefore$ Error $= (t - y_{in})^2 = (1.12)^2 = \mathbf{1.25}$

▶ **Step (IV) :** We update weights

We have, $x_1 = 1, x_2 = -1, \quad t = 1, \quad y_{in} = -0.12, b = -0.12$

Now, $w_i(new) = w_i\,(old) + \alpha\,(t - y_{in})x_i$

$w_1\,(new) = w_1(old) + \alpha\,(t - y_{in})\,x_1$

$= -0.12 + 0.2\,(1 + 0.12)(1)$

$= 0.10$

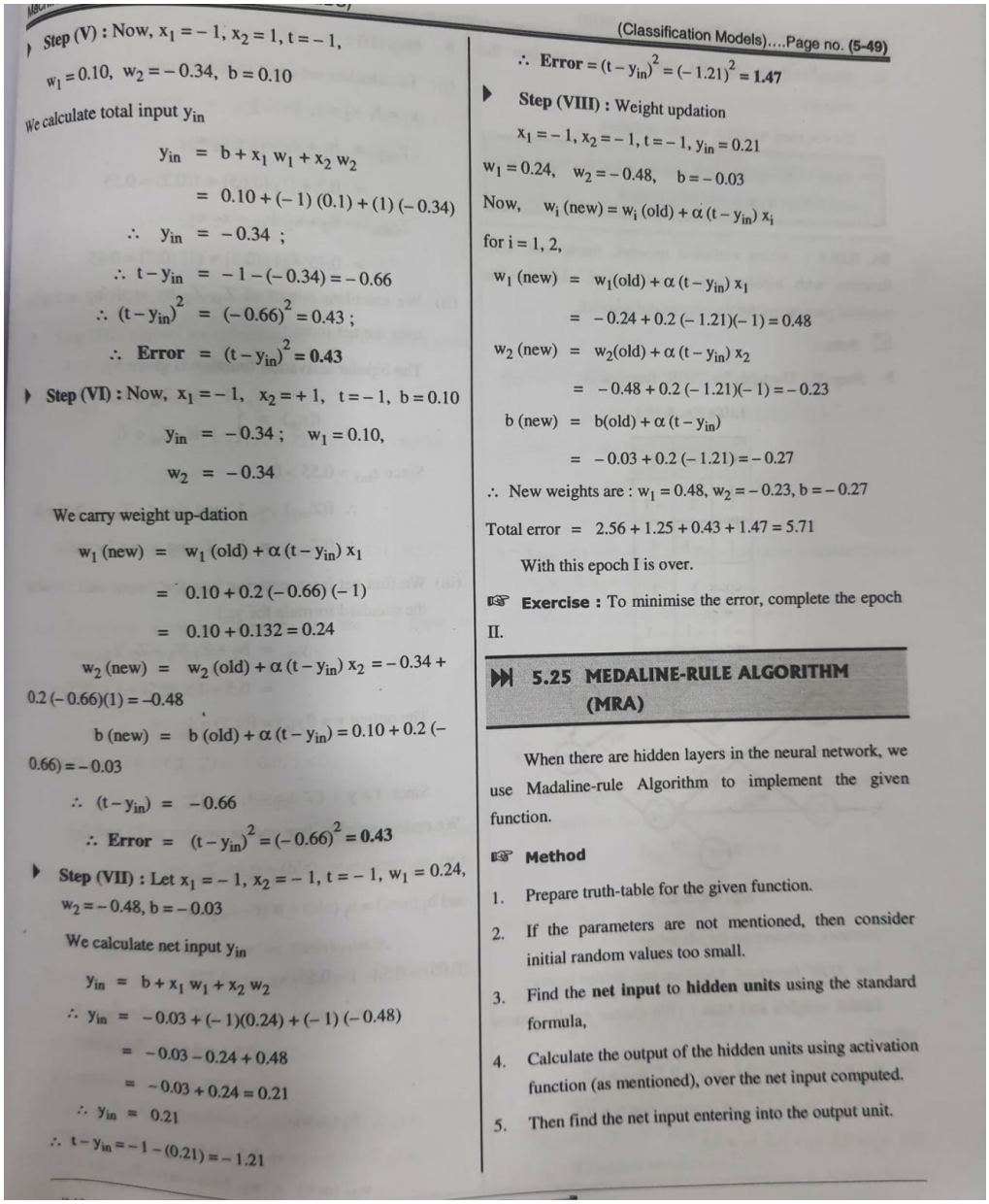
$w_2\,(new) = w_2\,(old) + \alpha\,(t - y_{in})x_2$

$= (-0.12) + 0.2\,(-1.12)(-1)$

$= -0.34$

and $b(new) = b(old) + \alpha\,(t - y_{in})$

$= -0.12 + 0.2\,(-1.12) = 0.10$



(Classification Models)....Page no. (5-49)

Step (V) : Now, $x_1 = -1$, $x_2 = 1$, $t = -1$,

$w_1 = 0.10$, $w_2 = -0.34$, $b = 0.10$

We calculate total input y_{in}

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$= 0.10 + (-1)(0.1) + (1)(-0.34)$$

$$\therefore \quad y_{in} = -0.34 \ ;$$

$$\therefore \quad t - y_{in} = -1 - (-0.34) = -0.66$$

$$\therefore \quad (t - y_{in})^2 = (-0.66)^2 = 0.43 \ ;$$

$$\therefore \quad \textbf{Error} = (t - y_{in})^2 = \mathbf{0.43}$$

Step (VI) : Now, $x_1 = -1$, $x_2 = +1$, $t = -1$, $b = 0.10$

$$y_{in} = -0.34 \ ; \quad w_1 = 0.10,$$

$$w_2 = -0.34$$

We carry weight up-dation

$$w_1 (\text{new}) = w_1 (\text{old}) + \alpha (t - y_{in}) x_1$$

$$= 0.10 + 0.2 (-0.66) (-1)$$

$$= 0.10 + 0.132 = 0.24$$

$w_2 (\text{new}) = w_2 (\text{old}) + \alpha (t - y_{in}) x_2 = -0.34 + 0.2 (-0.66)(1) = -0.48$

$b (\text{new}) = b (\text{old}) + \alpha (t - y_{in}) = 0.10 + 0.2 (-0.66) = -0.03$

$$\therefore \quad (t - y_{in}) = -0.66$$

$$\therefore \quad \textbf{Error} = (t - y_{in})^2 = (-0.66)^2 = \mathbf{0.43}$$

Step (VII) : Let $x_1 = -1$, $x_2 = -1$, $t = -1$, $w_1 = 0.24$, $w_2 = -0.48$, $b = -0.03$

We calculate net input y_{in}

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$\therefore \quad y_{in} = -0.03 + (-1)(0.24) + (-1)(-0.48)$$

$$= -0.03 - 0.24 + 0.48$$

$$= -0.03 + 0.24 = 0.21$$

$$\therefore \quad y_{in} = 0.21$$

$$\therefore \quad t - y_{in} = -1 - (0.21) = -1.21$$

$$\therefore \quad \textbf{Error} = (t - y_{in})^2 = (-1.21)^2 = \mathbf{1.47}$$

Step (VIII) : Weight updation

$x_1 = -1$, $x_2 = -1$, $t = -1$, $y_{in} = 0.21$

$w_1 = 0.24$, $w_2 = -0.48$, $b = -0.03$

Now, $w_i (\text{new}) = w_i (\text{old}) + \alpha (t - y_{in}) x_i$

for i = 1, 2,

$$w_1 (\text{new}) = w_1(\text{old}) + \alpha (t - y_{in}) x_1$$

$$= -0.24 + 0.2 (-1.21)(-1) = 0.48$$

$$w_2 (\text{new}) = w_2(\text{old}) + \alpha (t - y_{in}) x_2$$

$$= -0.48 + 0.2 (-1.21)(-1) = -0.23$$

$$b (\text{new}) = b(\text{old}) + \alpha (t - y_{in})$$

$$= -0.03 + 0.2 (-1.21) = -0.27$$

$\therefore$ New weights are : $w_1 = 0.48$, $w_2 = -0.23$, $b = -0.27$

Total error = $2.56 + 1.25 + 0.43 + 1.47 = 5.71$

With this epoch I is over.

☞ **Exercise :** To minimise the error, complete the epoch II.

5.25 MEDALINE-RULE ALGORITHM (MRA)

When there are hidden layers in the neural network, we use Madaline-rule Algorithm to implement the given function.

☞ **Method**

1. Prepare truth-table for the given function.
2. If the parameters are not mentioned, then consider initial random values too small.
3. Find the **net input** to **hidden units** using the standard formula,
4. Calculate the output of the hidden units using activation function (as mentioned), over the net input computed.
5. Then find the net input entering into the output unit.

30. Explain LMS weight update rule.

Ans) The Least Mean Squares (LMS) algorithm is an iterative method used for adapting the weights of the perceptron in order to minimize the mean square error (MSE) between the output of the perceptron and the target output. The LMS rule is used to adjust the weights of a linear regression model to minimize the sum of the squared residuals. It is commonly used in adaptive signal processing, machine learning, and control systems.

The LMS weight update rule works as follows:

1. Initialize the weights of the perceptron to small random values.
2. Present a training example to the perceptron and compute the output.
3. Compute the error between the output of the perceptron and the target output.

4. Adjust the weights of the perceptron based on the error and the learning rate using the following formula:
   w(t+1) = w(t) + n * error * x(t)

where w(t+1) is the updated weight, w(t) is the current weight, n is the learning rate, error is the difference between the target output and the perceptron output, and x(t) is the input vector.

5. Repeat steps 2 to 4 for all training examples until the MSE is minimized.

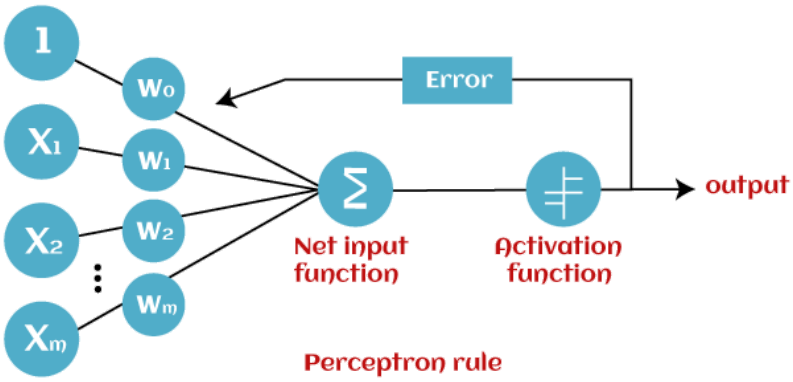
The LMS weight update rule can be used for both single-layer perceptrons and multi-layer perceptrons. It is a simple and efficient algorithm that can converge quickly to the optimal solution.

31. Discuss the Perceptron training rule.

Ans)

In the following algorithm the inputs are initially assigned and then the net input is calculated. Applying activation formula over the calculated net input, the output of the network is obtained.

We compare the calculated and desired output, and then carry out the weight updation. The perceptron algorithm is used for either binary or bipolar input vectors, having bipolar targets, with varying bias and fixed threshold. The entire network is based on stopping criterion.



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:

$\sum wi*xi = x1*w1 + x2*w2 + \ldots wn*xn$

Add a special term called bias 'b' to this weighted sum to improve the model's performance.
$\sum wi*xi + b$

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:
$Y = f(\sum wi*xi + b)$

32. What are the types of problems in which Artificial Neural Network can be applied.
Ans)

1. Classification: This type of problem involves predicting a categorical output. ANNs can be used for tasks such as image classification, sentiment analysis, and spam detection. For example, an ANN can be trained on a dataset of labeled images of cats and dogs to classify a new image as either a cat or a dog.
2. Regression: This type of problem involves predicting a continuous output. ANNs can be used for tasks such as stock price prediction, weather forecasting, and house price prediction. For example, an ANN can be trained on a dataset of housing prices based on features such as location, number of bedrooms, and square footage, and then be used to predict the price of a new house based on those same features.
3. Clustering: This type of problem involves grouping data points together based on their similarity. ANNs can be used for tasks such as customer segmentation, anomaly detection, and image segmentation. For example, an ANN can be trained on a dataset of unlabeled images to group them into different categories based on their visual features.
4. Reinforcement Learning: This type of problem involves training an agent to take actions in an environment to maximize a reward. ANNs can be used for tasks such as game playing, robotics, and autonomous driving. For example, an ANN can be used to train a robotic arm to grasp and manipulate objects based on feedback from its sensors and environment.