**FSDS MAY BATCH 2022(DL Assignment -1)**

**Submitted by: Shubham Tiwari.**

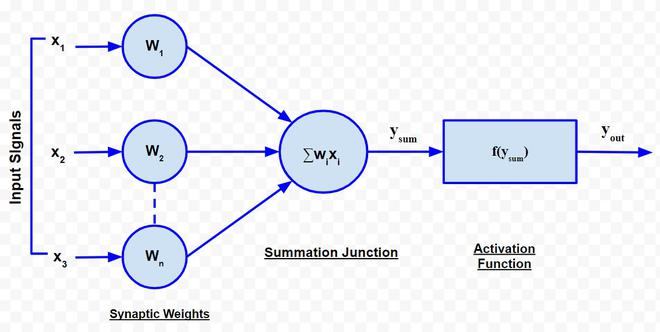
Q1: What is the function of a summation junction of a neuron? What is threshold activation function ?

Ans: Each neuron consists of three major components:

1. A set of **‘i’ synapses having weight wi.**A signal xi forms the input to the i-th synapse having weight wi. The value of any weight may be positive or negative. A positive weight has an extraordinary effect, while a negative weight has an inhibitory effect on the output of the summation junction.
2. A **summation junction** for the input signals is weighted by the respective synaptic weight. Because it is a linear combiner or adder of the weighted input signals, the output of the summation junction can be expressed as follows:

**ysum =**

1. A threshold**activation function** (or simply **the activation function,**also known as **squashing function**) results in an output signal only when an input signal exceeding a specific threshold value comes as an input. It is similar in behaviour to the biological neuron which transmits the signal only when the total input signal meets the firing threshold.



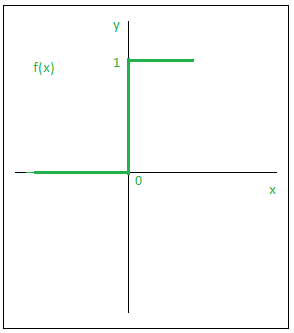
Q2: What is a step function? What is the difference of step function with threshold function?

Ans: **Step Function** is one of the simplest kind of activation functions. In this, we consider a threshold value and if the value of net input say y is greater than the threshold then the neuron is activated.

Mathematically,

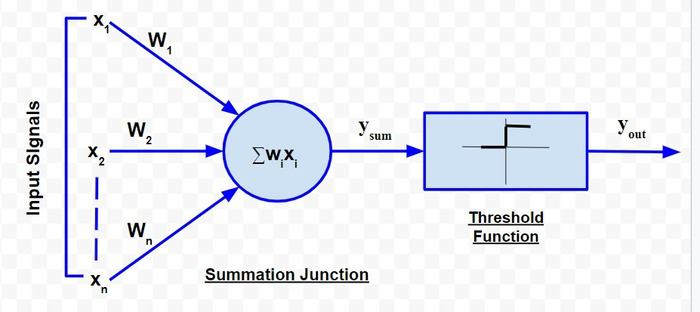
f(x)=1,if x >=0

f(x)=0,if x<0



Q3: Explain the McCulloch–Pitts model of neuron.

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 yout is 1 if the input ysum is greater than or equal to a given threshold value, else 0. The diagrammatic representation of the model is as follows:



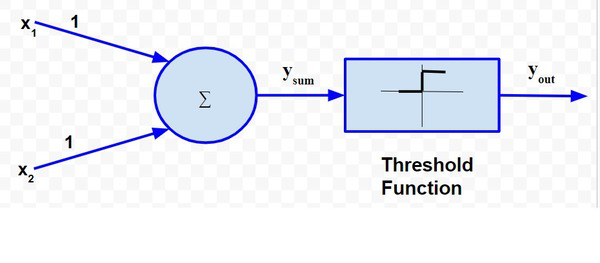
Simple McCulloch-Pitts neurons can be used to design logical operations. For that purpose, the connection weights need to be correctly decided along with the threshold function (rather than the threshold value of the activation function). For better understanding purpose, let me consider an example:

* First scenario: It is not raining, nor it is sunny
* Second scenario: It is not raining, but it is sunny
* Third scenario: It is raining, and it is not sunny
* Fourth scenario: It is raining as well as it is sunny

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

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

So, the value of both scenarios can be either 0 or 1. We can use the value of both weights X1 and X2as 1 and a threshold function as 1. So, the neural network model will look like:



**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, I can say that,





The truth table built with respect to the problem is depicted above. From the truth table, I can conclude that in the situations where the value of *yout* is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.

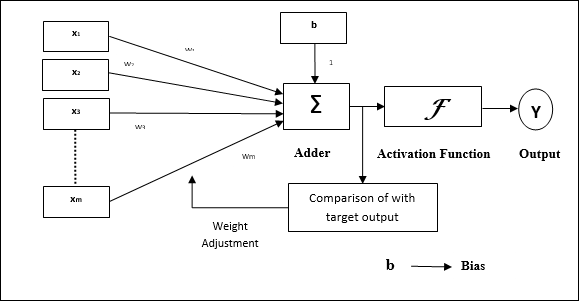
Q4: Explain the ADALINE network model.

Ans:ADALINE basically stands for Adaptive Linear Neuron, is a network having a single linear unit. Some important points about Adaline are as follows :

* It uses bipolar activation function.
* It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.
* The weights and the bias are adjustable.

**Architecture:**

The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired/target output. After comparison on the basis of training algorithm, the weights and bias will be updated.



Training Algorithm

**Step 1** − Initialize the following to start the training −

* Weights
* Bias
* Learning rate α.

For easy calculation and simplicity, weights and bias must be set equal to 0 and the learning rate must be set equal to 1.

**Step 2** − Continue step 3-8 when the stopping condition is not true.

**Step 3** − Continue step 4-6 for every bipolar training pair **s:t**.

**Step 4** − Activate each input unit as follows −

xi=si(i=1to n)

**Step 5** − Obtain the net input with the following relation −

yin=b+

Here **‘b’** is bias and **‘n’** is the total number of input neurons.

**Step 6** − Apply the following activation function to obtain the final output

**Step 7** − Adjust the weight and bias as follows :

f(yin)={1 ifyin⩾0

-1 ifyin<0

**Case 1** − if **y ≠ t** then,

wi(new)=wi(old)+α(t−yin)xi

b(new)=b(old)+α(t−yin)

**Case 2** − if **y = t** then,

wi(new)=wi(old)

b(new)=b(old)

Here **‘y’** is the actual output and **‘t’** is the desired/target output.

(t−yin) is the computed error.

**Step 8** − Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.

Q5: What is the constraint of a simple perceptron? Why it may fail with a real-world data set?

Ans: The most glaring limitation of the perceptron is the fact that it is only capable of solving classification problems that are linearly separable. This implies separation by a line in two-dimensional space, a plane in three-dimensional space, and a hyperplane in p-dimensional space.It fail because :

**A perceptron model has limitations as follows:**

* The output of a perceptron can only be a binary number (0 or 1) due to the hard limit transfer function.
* Perceptron can only be used to classify the linearly separable sets of input vectors. If input vectors are non-linear, it is not easy to classify them properly.

Q6: What is linearly inseparable problem? What is the role of the hidden layer?

Ans: **Linear separability problem** means that if there are two classes then there will be a point, line, plane, or hyperplane that splits the input features in such a way that all points of one class are in one-half space and the second class is in the other half-space.

**Hidden layers**: They are “hidden” because the true values of their nodes are unknown in the training dataset. In fact, we only know the input and output. Each neural network has at least one hidden layer. Otherwise, it is not a neural network. Networks with multiple hidden layers are called deep neural networks. The most common type of hidden layer is the fully-connected layer.

Q7: Explain XOR problem in case of a simple perceptron.

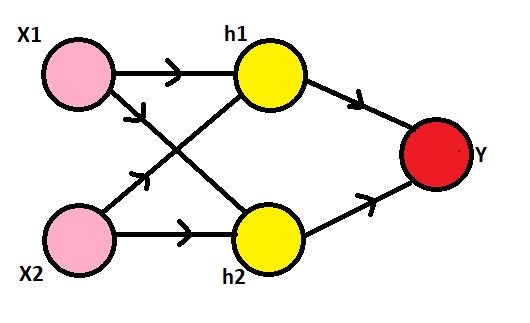
Ans: The XOR classification problem cannot be solved using a single perceptron. A perceptron can only converge on linearly separable data. Therefore, it isn't capable of imitating the XOR function.

Q8: Design a multi-layer perceptron to implement A XOR B.

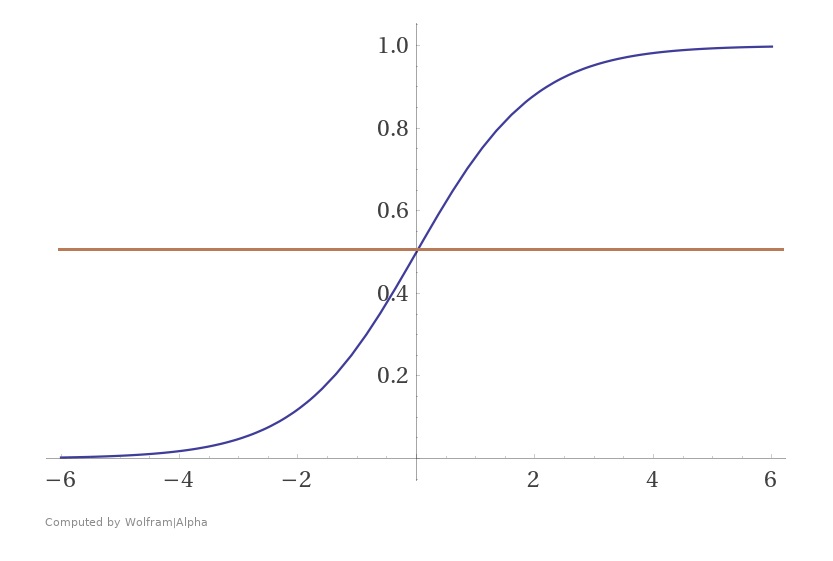
Ans: To solve this problem, we add an extra layer to our vanilla perceptron, i.e., we create a **Multi Layered Perceptron** (or **MLP**). We call this extra layer as the **Hidden layer**. To build a perceptron, we first need to understand that the XOr gate can be written as a combination of AND gates,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

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

The following is a plan for the perceptron.



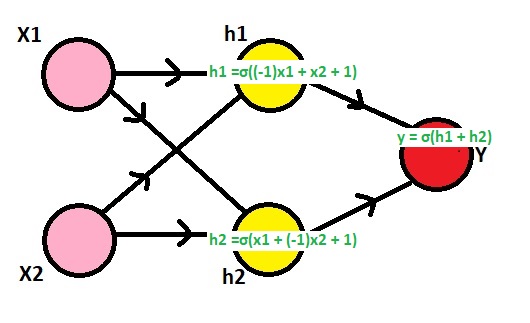
Here, we need to observe that our inputs are 0s and 1s. To make it a XOr gate, we will make the h1 node to perform the (x2 AND NOT x1) operation, the h2 node to perform (x1 AND NOT x2) operation and the y node to perform (h1 OR h2) operation. The NOT gate can be produced for an input a by writing (1-a),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



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

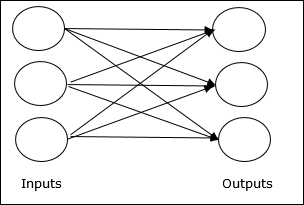
1. h1 = σ((1-x1) + x2) = σ((-1)x1 + x2 + 1)
2. h2 = σ(x1 + (1-x2)) = σ(x1 + (-1)x2 + 1)
3. y = σ(h1 + h2) = σ(h1 + h2 + 0)

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



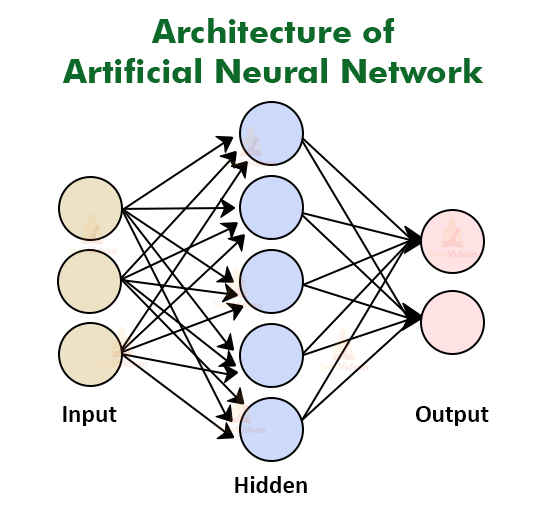
Q9: Explain the single-layer feed forward architecture of ANN.

Ans**: Single layer feedforward network** − The concept is of feedforward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer.



Q10: Explain the competitive network architecture of ANN.

Ans: An [Artificial Neural Network (ANN)](https://www.geeksforgeeks.org/implementing-ann-training-process-in-python/) is an information processing paradigm that is inspired by the brain. ANNs, like people, learn by examples. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning largely involves adjustments to the synaptic connections that exist between the neurons.



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

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

**Interconnections:**

Interconnection can be defined as the way processing elements (Neuron) in ANN are connected to each other. Hence, the arrangements of these processing elements and geometry of interconnections are very essential in ANN.

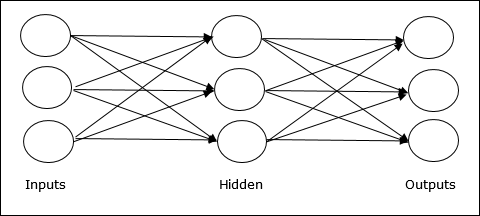
These arrangements always have two layers that are common to all network architectures, the Input layer and output layer where the input layer buffers the input signal, and the output layer generates the output of the network. The third layer is the Hidden layer, in which neurons are neither kept in the input layer nor in the output layer. These neurons are hidden from the people who are interfacing with the system and act as a black box to them. By increasing the hidden layers with neurons, the system’s computational and processing power can be increased but the training phenomena of the system get more complex at the same time.

**Working of ANN:**

Firstly, the information is feed into the input layer. Which then transfers it to the hidden layers, and interconnection between these two layers assign weights to each input randomly at the initial point. Then bias is add to each input neuron and after this, the weight sum which is a combination of weights and bias is pass through the activation function. Activation Function has the responsibility of which node to fire for feature extraction and finally output is calculate. Therefore this whole process is known as Forward Propagation. After getting the output model to compare it with the original output and the error is known and finally, weights are updates in backward propagation to reduce the error and this process continues for a certain number of epochs (iteration). Finally, model weights get updates and prediction is done.

Q11: Consider a multi-layer feed forward neural network. Enumerate and explain steps in the backpropagation algorithm used to train the network.

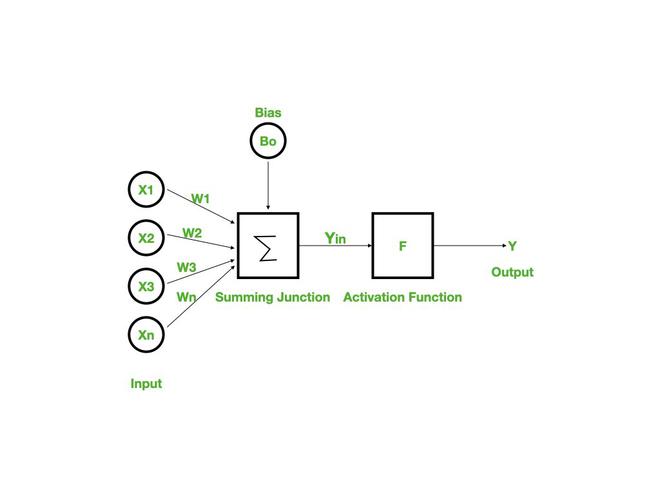
Ans: **Multilayer feedforward network** − The concept is of feedforward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.



**Backpropagation algorithm:**

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.



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

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

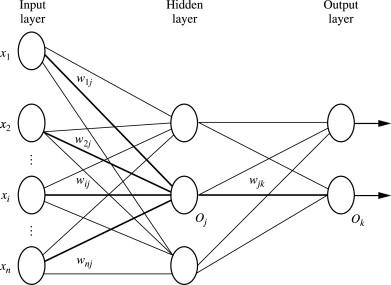
**Step 3**: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.

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

Backpropagation Error= Actual Output – Desired Output

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

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



**Parameters :**

x = inputs training vector x=(x1,x2,…………xn).

t = target vector t=(t1,t2……………tn).

δk = error at output unit.

δj = error at hidden layer.

α = learning rate.

V0j = bias of hidden unit j.

* **Training Algorithm** :

Step 1: Initialize weight to small random values.

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

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

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

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

zinj = v0j + Σxivij ( i=1 to n)

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

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

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

           and applies its activation function to calculate the output signals.

                     yk= f(yink)

**Backpropagation Error :**

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

                   δk = ( tk – yk ) + yink

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

  δinj = Σ δj wjk

              The error information term is calculated as :

                  δj = δinj+ zinj

**Updation of weight and bias :**

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

                                        Δ wjk= α δkzj

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

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

w0k(new) = wok(old)+ Δ wok

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

                                 Δ vij= α δj xi

and the bias connection on term

                                 Δ v0j= α δj

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

v0j(new) = v0j(old)+  Δv0j

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

Q12: What are the advantages and disadvantages of neural networks?

Ans: Following are the advantages and disadvantages of neural networks:

**Advantages:**

1)**Ability to train machine**: Artificial neural networks learn events and make decisions by commenting on similar events.

2) **Parallel processing ability**: Artificial neural networks have numerical strength that can perform more than one job at the same time.

3) **It has fault tolerance**: Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault-tolerant.

4) **Storing information on the entire network:** Information such as in traditional programming is stored on the entire network, not on a database. The disappearance of a few pieces of information in one place does not restrict the network from functioning.

**Disadvantages:**

1)**Hardware dependence:** Artificial neural networks require processors with parallel processing power, by their structure. For this reason, the realization of the equipment is dependent.

2)**Incomplete results**: The second demerit of neural networks is that they can often create incomplete results or outputs. Since ANNs are trained to adapt to the changing applications of neural networks, they are often left untrained for the whole process.

3)**Data suitability**: Another one of the challenges of neural networks is that they are highly dependent on the data made available to them. This infers that the efficiency of any neural network is directly proportional to the amount of data it receives to process.

4)**Minimal control:** While artificial neural networks programs are pretty much advantageous when it comes to organizing unorganized data, they can be highly damaging too. This refers to the minimal control that the trainers have over the actual performance and overall functioning of the ANNs.

Q13: Write short notes on **any two** of the following:

1. Biological neuron.

Ans:xxxxxxxxxxxxxxx.

2. ReLU function

Ans: (i)The rectified linear activation function or ReLU is a non-linear function or piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

(ii)It is the most commonly used activation function in neural networks, especially in Convolutional Neural Networks (CNNs) & Multilayer perceptrons.

(iii)It is simple yet it is more effective than it's predecessors like sigmoid or tanh.

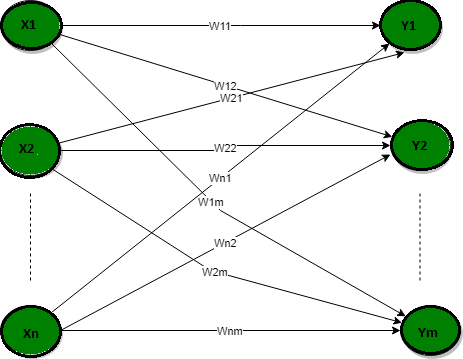
Mathematically it is expressed as :

**F(x)= max(0, x)**

(iv) It is not zero-centric.

(v) It is going to solve vanishing gradient problem but there is a issue that if the derivative is zero ,then it will lead to **Dead neuron.**

3. Single-layer feed forward ANN.

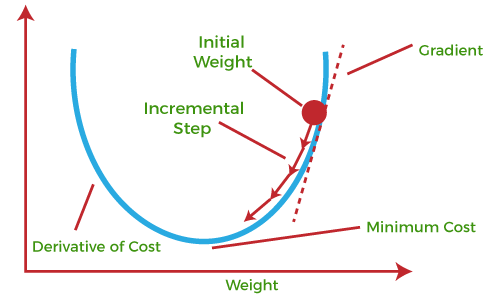


Ans: In this type of network, we have only two layers input layer and the output layer but the input layer does not count because no computation is performed in this layer. The output layer is formed when different weights are applied to input nodes and the cumulative effect per node is taken. After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

4. Gradient descent.

Ans**: The main objective of using a gradient descent algorithm is to minimize the cost function using iteration*.*** To achieve this goal, it performs two steps iteratively:

* Calculates the first-order derivative of the function to compute the gradient or slope of that function.
* Move away from the direction of the gradient, which means slope increased from the current point by alpha times, where Alpha is defined as Learning Rate. It is a tuning parameter in the optimization process which helps to decide the length of the steps.



5. Recurrent networks.

Ans: xxxxxxxxxx