... function:

22/03

Neural Networks & Deep Learning

SK MUSKAN
32112651OL07
3/4 BTech CSE-B

ASSIGNMENT-1

1. Define Artificial Neural Network. Explain briefly the operation of biological neural network and its characteristics.

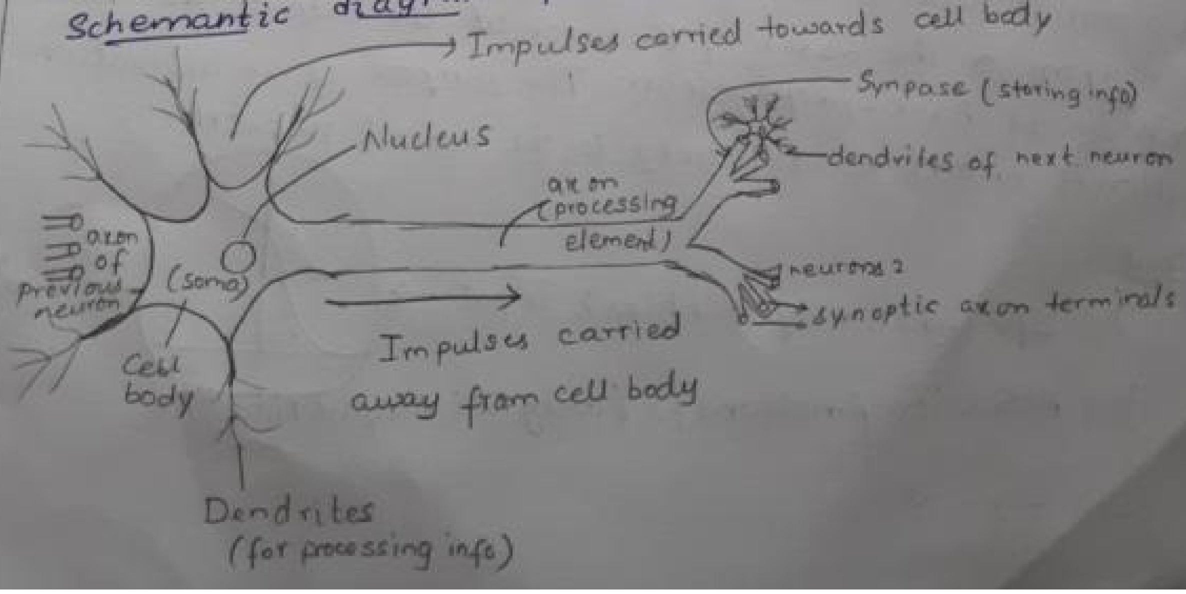
Artificial Neural Network is derived from Biological neural networks that develop the structure of a human brain. Artificial Neural Networks have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

Characteristics of Neural network

1. Robustness and Fault Tolerance: The decay of nerve cell doesnot effect the performance
2. Flexibility: The n/w will adapt the new environments without using the pre-programmed instructions.
3. Ability to adapt different data situations: The n/w can deal with different information such as fuzzy, noisy & inconsistent
4. Collective Computation: The n/w performs many operations in parallel and the given task in a distributed manner

Biological Neural Network

Schemantic diagram for a biological Neuron:



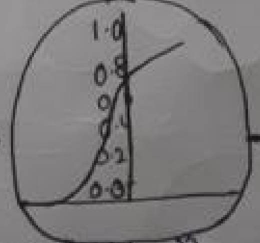
- These dendrites are tree like nerve fi[illegible]
incoming signals toward cell body.
- Fundamental unit of Network – Neuron/Nerve cell.
- Axon is single long fibre extending from cell body to process the elements
- Neuron consists of cellbody /soma where nucleus is located.
- The signals generated in soma are transmitted to other neurons is called axon/nerve fibre.
- Tree like nerve fibres are called dendrite which receives the incoming signals from other neurons.
- A neuron can drive upto 10^3 to 10^4 synoptic junctions.

2. Identify the need for activation function. Explain different types of activation functions.

The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it.

Explanation

We know that neural network has neurons that work in correspondance with weight, bias and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as back-propagation. The purpose of an activation function is to add non-linearity to the neural network.



Output

$Y = \Sigma(\text{weights} * \text{input} + \text{bias})$ Input → → Output

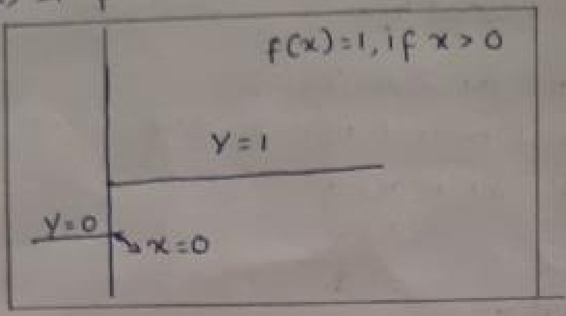
$Y = \text{Activation functions}(\Sigma(\text{weights} * \text{input} + \text{bias}))$

$-\infty \leq Y \leq \infty$

There are 3 types of Activation function:

1. Binary step function: It is a threshold-based activation function. If the input-value is above or below a certain threshold, the neuron is activated sends exactly the same signal to next layer. and the value to decide output that neuron should be activated or deactivated.

$f(x) = 1$ if $x > 0$ else 0 if $x < 0$

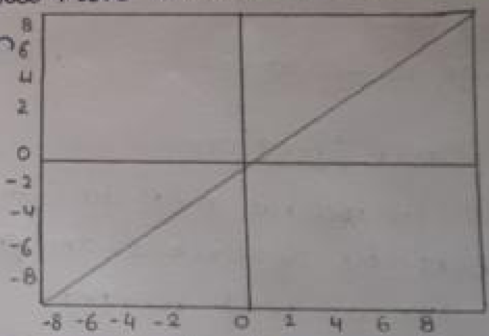


2. Linear function: Simple straight line activation function where our function directly proportional to the weighted sum of neuron

$y = mz$

Better in giving a wide range of activations and of positive slope may increase the firing input rate increases. A linear activation function takes the form

$A = cx$



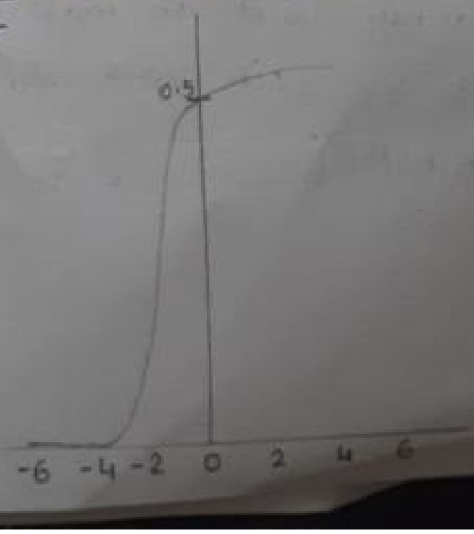
→ takes inputs & creates output
→ range is from $-\infty$ to $+\infty$

3. Non-Linear function:

Modern neural n/w models use non-linear activation functions almost any process imaginable can be represented as a functional computation in a neural n/w, provided activation function is non-linear.

→ Sigmoid or logistic Activation function which appears in o/p layer of deep learning models & it is used for predicting probability-based outputs sigmoid function is represented as

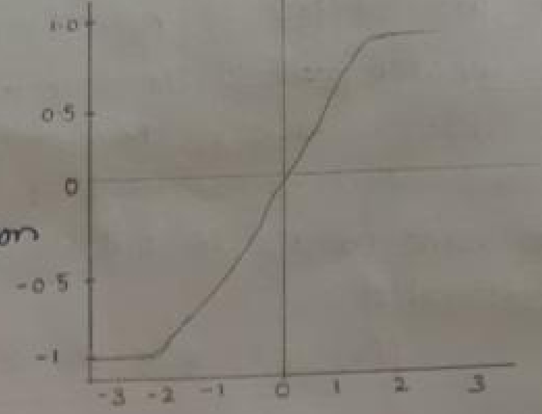
$$A = \frac{1}{1+e^{-x}}$$



- It is non-linear in nature
- It is s-shaped

→ Hyperbolic Tangent function

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



- Similar to sigmoid function
- also called as scaled sigmoid function

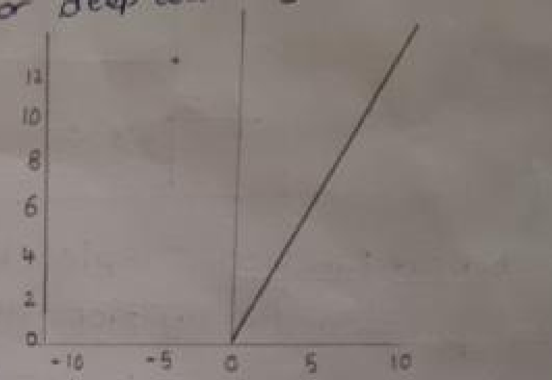
$$\tanh(x) = 2\,\mathrm{sigmoid}(2x) - 1$$

- range -1 to +1
- used in feed forward neural n/w

→ Rectified Linear Unit function (ReLU)

used in convolutional neural n/w or deep learning

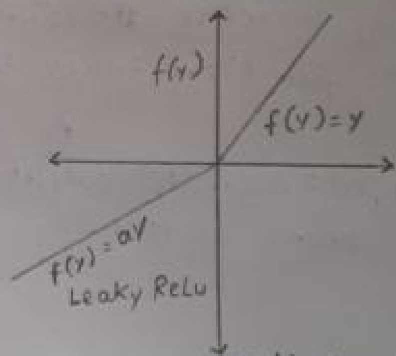
$$f(x) = 0 \quad \text{when } x < 0$$
$$= x \quad x \geq 0$$



- range 0 to ∞
- It is non linear in nature
- It is s-shaped

→ Leaky ReLU: It helps to increase range of ReLU function. value of a is 0.01 or 0.5. when a is not 0.01 it is called randomized Leaky ReLU

- range is $-\infty$ to $+\infty$



→ Softmax function:

Used in neural n/w to compute probability distribution from a vector of real numbers

o/p range from 0 & 1 with sum of probabilities = 1

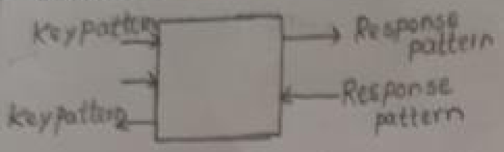
$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

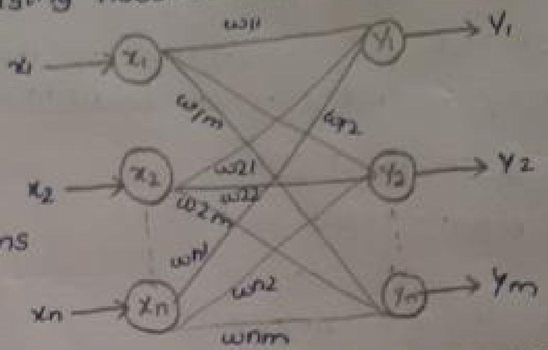
Mainly used in multi-class models where it returns prob of each class, with the target class having highesht probability.

3. Explain the architecture for about Bidirectional Associative memory model

Bidirectional Associative Memory (BAM):

- developed by kosko in the year 1988.
- performs backward + forward search.
- Encodes binary/bipolar pattern using hebbian learning rules.





- consists of two layers of neurons connected by directed weights
- network iterates by sending the signals back & forth between two layers until all the Neurons reach equilibrium.
- Weights are bidirection.
- Consists of 'n' units in x layer & m units in Y layer.

Training Algorithm:

Obtain Weight Matrix $W = \{W_{ij}\}$ by

$$W_j = \sum_{P=1}^{P} [2S_i(P)-1][2t_j(P)-1] \quad \text{[for binary I/P vectors]}$$

$$W_j = \sum_{P=1}^{P} S_i(P)-t_j(P) \quad \text{[for bipolar I/P vector]}$$

$$S(P) = (S_1(P), --- S_i(P) -- S_n(P))$$

$$t(P) = (t_1(P), ---- t_i(P) --- t_n(P))$$

Testing Algorithm:

Step-1: Initialize weights to store p patterns.

Step-2: Update the activations of units in Y layer.

Calculate the net input

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Apply Activation function

$$y_{in} = f(y_{inj})$$

Send this signal to X layer

Step-3: Update activation of units in xlayer.

net input $x_{in\,i} = \sum_{y=1}^{m} y_i w_{ij}$

apply activation over net input

$x_i = f(x_{in\,i})$

Send this signal to ylayer.

Step-4 Test for convergence of the net convergence occurs of activation vectory x and y reach equilibrium. If this occurs then stop.

Activation functions:

1. With binary input vectors is

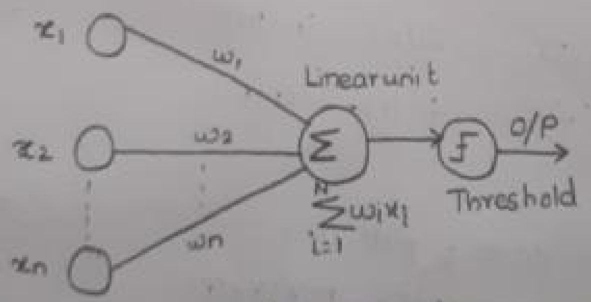
$$y_j = \begin{cases} 1 & y_{inj} > 0 \\ 0 & y_{inj} < 0 \\ y_j & y_{inj} = 0 \end{cases}$$

2. With bipolar I/P vectors is

$$y_j = \begin{cases} 1 & y_{inj} > 0 \\ y_j & y_{inj} = 0 \\ -1 & y_{inj} < 0 \end{cases}$$

4 Illustrate how pattern classification takes place using perception

Pattern classification N/w perceptron is mainly used for building artificial neural network system



O/P = 1 if $\sum w_i x_i \geq T$
= 0 otherwise

Perceptron takes a vector of real valued I/P values & computes the linear combination of weights & I/P's & results 1

If O/P is greater than or equal to threshold value otherwise 0

$$W_i \leftarrow w_i + \Delta w_i$$

$$\Delta W_i = \eta(t - o)x_i$$

where $t \rightarrow$ target O/P

$O \rightarrow$ actual O/P

$\eta \rightarrow$ learning parameter

Perceptron training Rule (x, η) algorithm:

initialize w_i to random values

repeat

for all instances of $(x, t_x) \in X$

compute O_x = activation, summation$(\omega \cdot x)$)

if $(t_x \neq O_x)$

for all weight w_i

$w_i \leftarrow w_i + \Delta w_i$

$\Delta w_i = \eta(t-o)x_i$

end for

end if

end for

until all O/P's are classified correctly

return new updated as learned parameter.

if data is linearly seperable then the perceptron converges.
if data is not linearly seperable then the perceptron not converges
mostly used algorithm for convergence rule we use gradient descent
algorithm called as delta learning rule.

Perceptron learning rule for AND Gate:

A	B	A∧B
0	0	0
0	1	0
1	0	0
1	1	1

Let $w_1 = 1.2$, $w_2 = 0.6$, $\eta = 0.5$, $T = 1$

Case-1: $0(1.2) + 0(0.6) = 0 \geq T$ ✗ actual O/P = 0
actual O/P = target O/P = 0. No need to update weights

case-2: $0(1.2) + 1(0.6) = 0.6 \geq T$ ✗ actual O/P = 0

actual O/P = target O/P = 0, No need to update weights.

Case-3: $1(1.2) + 0(0.6) = 1.2 \geq T$ ✓

actual O/P = 1, target O/P = 0, A O/P $\neq$ T O/P

So, update the weights

$\Delta w_i = \eta(t-o)x_i$

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

$w_2 = w_2 + \Delta w_2 \Rightarrow 0.6 + (0.5(0-1)0) = 0.6$

$w1 = 0.7, w_2 = 0.6$

After updation,

case1: $0(0.7) + 0(0.6) = 0 \geq T$ ×

actual O/P = target O/P = 0, no need to update weights

case2: $0(0.7) + 1(0.6) = 0.6 \geq T$ ×

actual O/P = target O/P = 0, no weights updates

case3: $0(0.6) + 1(0.7) = 0.7 \geq T$ ×

actual O/P = target O/P = 0, no updated weights

case4: $1(0.7) + 1(0.6) \geq T$ ✓

actual O/P = target O/P = 1