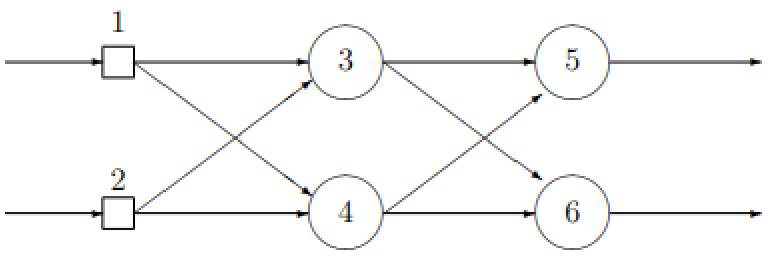
USN | | | | | | | | | | |

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A++ GRADE BY NAAC

Sub:	**Introduction to AI & ML**				Sub Code:	**21CS752**	Branch:	**EEE**	
Date:	**19/11/24**	Duration:	**90 minutes**	Max Marks:	**50**	Sem/Sec:	**VII**		**OBE**

		<u>Answer any FIVE FULL Questions</u>	**MARKS**	**CO**	**RBT**
1	a	Describe the terms Artificial Neural Network, Perceptron and Self organizing maps.	[10]	04	L1
2	a	Differentiate between Deep Learning and Machine Learning.	[5]	04	L2
	b	Illustrate the architecture of Radial Basis Function Neural Networks.	[5]	04	L1
3	a	Describe the stepwise working of Multilayer perceptron.	[6]	04	L1
	b	Elucidate the merits and demerits of Back Propagation Neural Networks	[4]	04	L1
4	a		[10]	04	L3



A weight on connection between nodes i and j is given by w_{ij}. The following table lists all the weights in the network.

$w_{13} = -2$	$w_{35} = 1$
$w_{23} = 3$	$w_{45} = -1$
$w_{14} = 4$	$w_{36} = -1$
$w_{24} = -1$	$w_{46} = 1$

Each of the nodes 3, 4, 5 uses the following activation function. Where ϑ denotes the weighted sum of a node. Each of the input nodes can only receive binary values either 0 or 1.

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Calculate the output of the network y_5 and y_6 for each of the input patterns.

Pattern:	P_1	P_2	P_3	P_4
Node 1:	0	1	0	1
Node 2:	0	0	1	1

5	a	What are the advantages and disadvantages of self organizing feature maps. Discuss.	[10]	04	L1
6	a	Explain sigmoid activation function in detail.	[4]	04	L1
	b	Explain the different activation functions used in neural networks, giving an example of each.	[6]	04	L2

Faculty CCI HOD

SOLUTION

Ans 1(a) **Artificial Neural Network (ANN)**

An Artificial Neural Network (ANN) is a computational model inspired by the structure and functioning of biological neural networks in the human brain. It consists of interconnected nodes (neurons) organized in layers, working together to process data and learn patterns**.**

- **Structure:**
  - **Input Layer: Takes in the data features.**
  - **Hidden Layers: Perform computations and extract patterns through interconnected neurons.**
  - **Output Layer: Produces the final prediction or classification.**
- **Learning Process: ANN learns by adjusting the weights of the connections between neurons using algorithms like gradient descent to minimize errors based on a defined loss function.**
- **Applications: Used in tasks like image recognition, natural language processing, and predictive analytics.**

Neuron

A neuron is the basic computational unit in an ANN. It mimics the behavior of a biological neuron by receiving input, processing it, and passing an output.

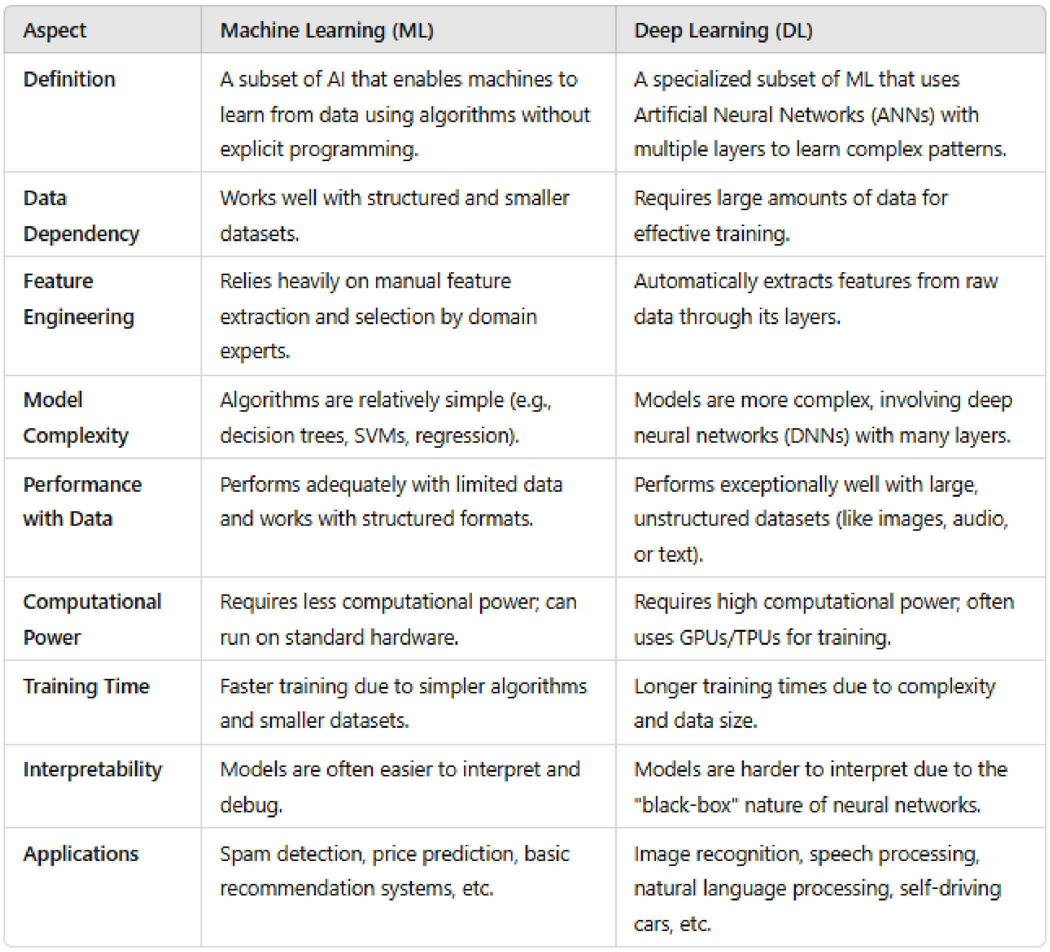
- **Components:**
  - Input: Signals or data (numerical values) fed to the neuron.
  - Weights: Parameters that scale the input signals.
  - Bias: A constant added to the weighted input to influence the activation.
  - Activation Function: Determines whether the neuron fires (produces output).
- **Processing:**
    - Each input is multiplied by its weight.
    - The weighted inputs are summed up, along with the bias.
    - The sum is passed through the activation function to produce the output**.**

Activation Function

An activation function decides whether a neuron should be activated by transforming the weighted sum of inputs into an output. It introduces non-linearity, enabling the ANN to model complex patterns and relationships.

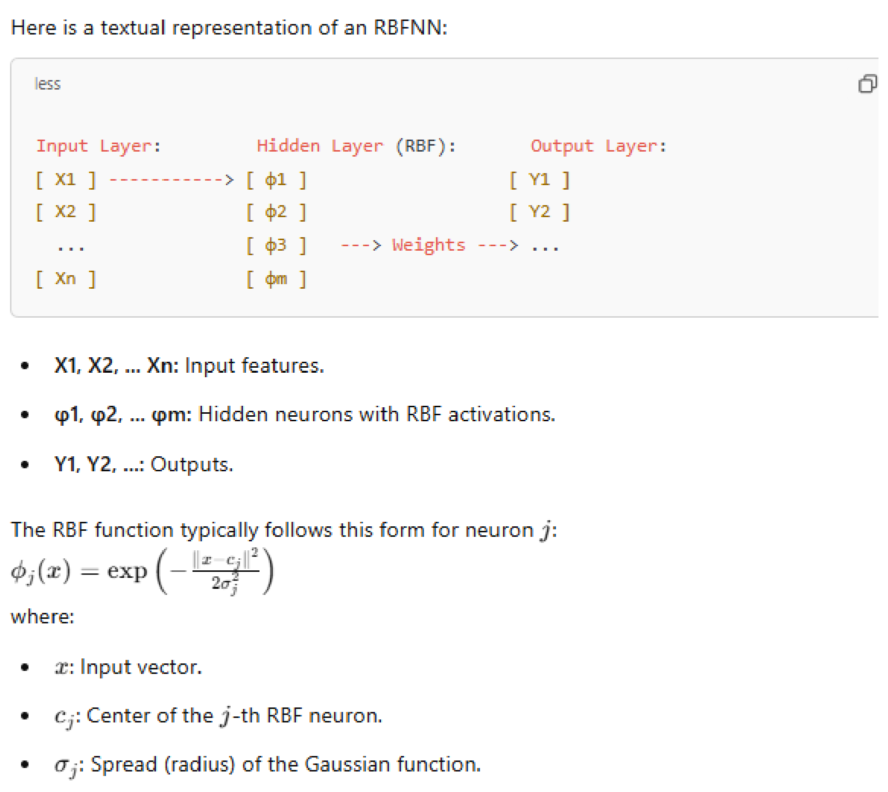
- Types of Activation Functions:
  1. Linear Activation: Directly passes the input (rarely used in modern ANNs).
  2. Non-linear Activations:
     - Sigmoid: Compresses values between 0 and 1. σ(z)=11+e−z\sigma(z) = \frac{1}{1 + e^{-z}}σ(z)=1+e−z1
     - ReLU (Rectified Linear Unit): Outputs zero for negative inputs and the input itself for positive values. f(z)=max(0,z)f(z) = \max(0, z)f(z)=max(0,z)
     - Tanh (Hyperbolic Tangent): Compresses values between -1 and 1. f(z)=tanh(z)f(z) = \tanh(z)f(z)=tanh(z)
     - Softmax: Used in multi-class classification to normalize outputs into probabilities.
- Role: Allows the network to handle complex data patterns and learn effectively during training.

Ans- 2 (a)



Aspect	Machine Learning (ML)	Deep Learning (DL)
Definition	A subset of AI that enables machines to learn from data using algorithms without explicit programming.	A specialized subset of ML that uses Artificial Neural Networks (ANNs) with multiple layers to learn complex patterns.
Data Dependency	Works well with structured and smaller datasets.	Requires large amounts of data for effective training.
Feature Engineering	Relies heavily on manual feature extraction and selection by domain experts.	Automatically extracts features from raw data through its layers.
Model Complexity	Algorithms are relatively simple (e.g., decision trees, SVMs, regression).	Models are more complex, involving deep neural networks (DNNs) with many layers.
Performance with Data	Performs adequately with limited data and works with structured formats.	Performs exceptionally well with large, unstructured datasets (like images, audio, or text).
Computational Power	Requires less computational power; can run on standard hardware.	Requires high computational power; often uses GPUs/TPUs for training.
Training Time	Faster training due to simpler algorithms and smaller datasets.	Longer training times due to complexity and data size.
Interpretability	Models are often easier to interpret and debug.	Models are harder to interpret due to the "black-box" nature of neural networks.
Applications	Spam detection, price prediction, basic recommendation systems, etc.	Image recognition, speech processing, natural language processing, self-driving cars, etc.

Ans 2(b)



Here is a textual representation of an RBFNN:

```
Input Layer:        Hidden Layer (RBF):       Output Layer:
[ X1 ] -----------> [ ϕ1 ]                  [ Y1 ]
[ X2 ]              [ ϕ2 ]                  [ Y2 ]
  ...               [ ϕ3 ]   ---> Weights ---> ...
[ Xn ]              [ ϕm ]
```

- **X1, X2, ... Xn:** Input features.
- **φ1, φ2, ... φm:** Hidden neurons with RBF activations.
- **Y1, Y2, ...:** Outputs.

The RBF function typically follows this form for neuron j:

$$\phi_j(x) = \exp\left(-\frac{\|x - c_j\|^2}{2\sigma_j^2}\right)$$

where:

- x: Input vector.
- c_j: Center of the j-th RBF neuron.
- σ_j: Spread (radius) of the Gaussian function.

Key Features of RBFNN:

1. **Localization:** Hidden neurons respond only to inputs within a localized region.
2. **Flexibility:** Works well for regression and classification tasks.
3. **Training:** Requires determining centers, spreads, and output weights.

Ans -3(a) The **Multilayer Perceptron (MLP)** is a type of artificial neural network composed of an input layer, one or more hidden layers, and an output layer. Here is a stepwise explanation of how it works:

Initialization

- **Architecture Design:** Define the number of layers, neurons per layer, and activation functions.
- **Weight and Bias Initialization:** Randomly initialize weights and biases for each connection in the network.

---

2. Forward Propagation

The input data flows through the network, layer by layer, producing an output prediction. The steps are:

Step 1: Input Layer

- Accepts the feature set x=[x1,x2,…,xn]\mathbf{x} = [x_1, x_2, \ldots, x_n]x=[x1,x2,…,xn] from the dataset.
- Passes the inputs to the first hidden layer without modifications.

Step 2: Hidden Layer(s)

where:

- w_{ij}: Weight between the i-th input and j-th neuron.
- b_j: Bias term for the j-th neuron.
- z_j: The pre-activation value of the j-th neuron.

- Applies an **activation function** (e.g., ReLU, sigmoid, or tanh) to introduce non-linearity:

$$a_j = f(z_j)$$

where a_j is the output of the j-th neuron.

Step 3: Output Layer

- The outputs from the last hidden layer are transformed into final predictions using the same process:

$$y_k = f\left(\sum_{j=1}^{m} w_{jk} \cdot a_j + b_k\right)$$

Loss Calculation

- Compare the predicted output with the actual target value using a **loss function** (e.g., mean squared error for regression, cross-entropy for classification):

4. **Backpropagation**

Backpropagation adjusts the weights and biases to minimize the loss function by propagating the error backward through the network. The steps are:

Step 1: Compute Gradients

- Calculate the derivative of the loss function with respect to weights and biases using the **chain rule** of calculus.

$$L = \mathrm{Loss}(y_{\text{pred}}, y_{\text{true}})$$

5. Weight Update (Optimization)

Update the weights and biases using an optimization algorithm like **Stochastic Gradient Descent (SGD)** or **Adam**:

$$w_{ij} \leftarrow w_{ij} - \eta \cdot \frac{\partial L}{\partial w_{ij}}$$

where η is the learning rate.

Ans-3 (b) Merits of Back Propagation Neural Networks (BPNN)

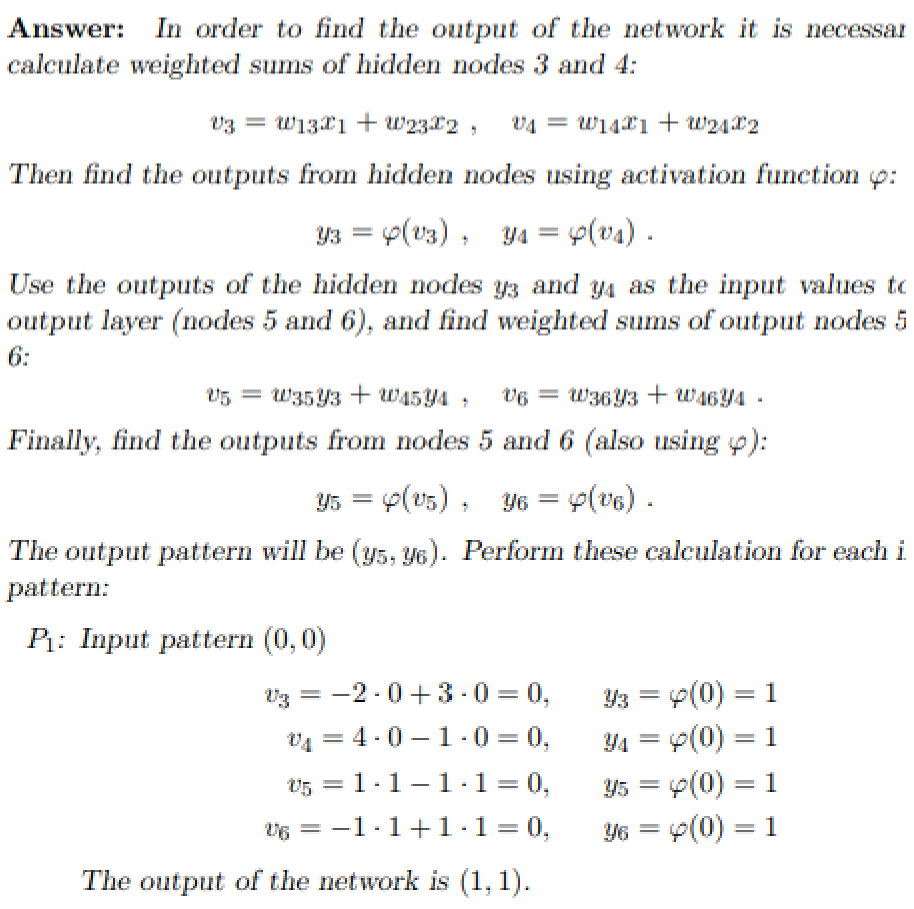
1. **Capability to Learn Nonlinear Relationships:**
   - BPNNs can model complex, nonlinear relationships between inputs and outputs, making them suitable for diverse applications like image recognition, speech processing, and predictive modeling.
2. **Versatile Application:**
   - They are applicable across domains such as classification, regression, and time-series prediction, offering wide usability.
3. **Automatic Feature Learning:**
   - Unlike traditional models that rely heavily on manual feature engineering, BPNNs learn features directly from the data.
4. **Adaptability:**
   - BPNNs can be extended and adapted to new problems by modifying the architecture or retraining on new data.
5. **Parallel Processing Capability:**
   - With multiple neurons and layers, BPNNs process information in parallel, making them efficient for large-scale problems.
6. **High Accuracy with Sufficient Data:**
   - Given enough data and computational power, BPNNs can achieve high accuracy levels in prediction and classification tasks.

Demerits of Back Propagation Neural Networks (BPNN)

1. **Computational Complexity:**
   - Training BPNNs requires significant computational resources, especially for large datasets and deep networks.
2. **Slow Convergence:**
   - Training can be slow due to the iterative nature of backpropagation, particularly when the learning rate is not optimized.

3. **Overfitting:**
   - BPNNs can overfit to training data, especially if the model is too complex or the training dataset is small.
4. **Dependence on Data Quality:**
   - Performance heavily relies on the quality and quantity of data. Poor or insufficient data can lead to suboptimal results.
5. **Difficulty in Choosing Hyperparameters:**
   - Selecting appropriate hyperparameters (e.g., learning rate, number of layers, neurons per layer) requires experience and often trial-and-error.
6. **Black Box Nature:**
   - The internal workings of BPNNs are difficult to interpret, making it challenging to understand how decisions are made.
7. **Vanishing Gradient Problem:**
   - In deeper networks, gradients may become very small during backpropagation, slowing learning or halting it entirely. This issue has been mitigated in modern architectures using techniques like ReLU activation and batch normalization.
8. **Sensitivity to Initial Weights:**
   - Poor initialization of weights can lead to slow or ineffective training.

Ans 4(a)



Answer: *In order to find the output of the network it is necessar calculate weighted sums of hidden nodes 3 and 4:*

$$v_3 = w_{13}x_1 + w_{23}x_2 \ , \quad v_4 = w_{14}x_1 + w_{24}x_2$$

Then find the outputs from hidden nodes using activation function φ:

$$y_3 = \varphi(v_3) \ , \quad y_4 = \varphi(v_4) \ .$$

Use the outputs of the hidden nodes y_3 and y_4 as the input values tc output layer (nodes 5 and 6), and find weighted sums of output nodes 5 6:

$$v_5 = w_{35}y_3 + w_{45}y_4 \ , \quad v_6 = w_{36}y_3 + w_{46}y_4 \ .$$

Finally, find the outputs from nodes 5 and 6 (also using φ):

$$y_5 = \varphi(v_5) \ , \quad y_6 = \varphi(v_6) \ .$$

The output pattern will be (y_5, y_6). Perform these calculation for each i pattern:

P_1: Input pattern $(0, 0)$

$$\begin{aligned} v_3 &= -2 \cdot 0 + 3 \cdot 0 = 0, & y_3 &= \varphi(0) = 1 \\ v_4 &= 4 \cdot 0 - 1 \cdot 0 = 0, & y_4 &= \varphi(0) = 1 \\ v_5 &= 1 \cdot 1 - 1 \cdot 1 = 0, & y_5 &= \varphi(0) = 1 \\ v_6 &= -1 \cdot 1 + 1 \cdot 1 = 0, & y_6 &= \varphi(0) = 1 \end{aligned}$$

The output of the network is $(1, 1)$.

P_2: Input pattern $(1, 0)$

$$\begin{aligned} v_3 &= -2 \cdot 1 + 3 \cdot 0 = -2, & y_3 &= \varphi(-2) = 0 \\ v_4 &= 4 \cdot 1 - 1 \cdot 0 = 4, & y_4 &= \varphi(4) = 1 \\ v_5 &= 1 \cdot 0 - 1 \cdot 1 = -1, & y_5 &= \varphi(-1) = 0 \\ v_6 &= -1 \cdot 0 + 1 \cdot 1 = 1, & y_6 &= \varphi(1) = 1 \end{aligned}$$

The output of the network is $(0, 1)$.

P_3: Input pattern $(0, 1)$

$$\begin{aligned} v_3 &= -2 \cdot 0 + 3 \cdot 1 = 3, & y_3 &= \varphi(3) = 1 \\ v_4 &= 4 \cdot 0 - 1 \cdot 1 = -1, & y_4 &= \varphi(-1) = 0 \\ v_5 &= 1 \cdot 1 - 1 \cdot 0 = 1, & y_5 &= \varphi(1) = 1 \\ v_6 &= -1 \cdot 1 + 1 \cdot 0 = -1, & y_6 &= \varphi(-1) = 0 \end{aligned}$$

The output of the network is $(1, 0)$.

P_4: Input pattern $(1, 1)$

$$\begin{aligned} v_3 &= -2 \cdot 1 + 3 \cdot 1 = 1, & y_3 &= \varphi(1) = 1 \\ v_4 &= 4 \cdot 1 - 1 \cdot 1 = 3, & y_4 &= \varphi(3) = 1 \\ v_5 &= 1 \cdot 1 - 1 \cdot 1 = 0, & y_5 &= \varphi(0) = 1 \\ v_6 &= -1 \cdot 1 + 1 \cdot 1 = 0, & y_6 &= \varphi(0) = 1 \end{aligned}$$

The output of the network is $(1, 1)$.

Ans-5 (a) **Advantages of Self-Organizing Feature Maps (SOFM or SOM)**

1. **Dimensionality Reduction:**
   - SOMs reduce high-dimensional data to a 2D or 3D representation while preserving the topological structure of the input data, aiding in visualization and interpretation.
2. **Clustering Capability:**
   - They group similar data points into clusters based on inherent patterns without requiring predefined labels or categories.
3. **Unsupervised Learning:**
   - SOMs do not require labeled data, making them useful for tasks where labeled data is unavailable or expensive to obtain.
4. **Topology Preservation:**
   - The network retains the relative positioning of data points, meaning similar data points are mapped close to each other in the output space.
5. **Noise Resistance:**
   - SOMs are relatively robust to noisy data and can identify patterns even in datasets with minor inconsistencies.
6. **Intuitive Visualization:**
   - By representing data in a grid, SOMs enable intuitive visualizations of complex relationships, often used in exploratory data analysis.
7. **Adaptive Learning:**
   - SOMs adjust dynamically to input data distributions, ensuring that the model adapts as new data patterns emerge.
8. **Versatility:**
   - Applicable in various domains like data mining, image processing, speech recognition, and genomics for tasks like clustering, anomaly detection, and pattern recognition.

Disadvantages of Self-Organizing Feature Maps

1. **Parameter Sensitivity:**
   - Performance depends on hyperparameters like learning rate, neighborhood radius, and grid size, which often require manual tuning.
2. **High Computational Cost:**
   - Training SOMs can be computationally intensive, especially for large datasets or high-dimensional input spaces.
3. **Limited Scalability:**
   - As the size of the dataset or the number of features increases, the training time and memory requirements grow significantly.
4. **Fixed Grid Size:**
   - The output grid size must be predefined, which might lead to underutilization or overcrowding of neurons if not chosen appropriately.
5. **No Probabilistic Framework:**
   - Unlike some clustering methods (e.g., Gaussian Mixture Models), SOMs do not provide probabilistic outputs, limiting their interpretability in some contexts.
6. **Difficulty in Defining Clusters:**
   - Determining the boundaries between clusters on the map can be subjective and requires additional interpretation.
7. **Not Suitable for Dynamic Data:**
   - SOMs are static models and need retraining to incorporate new data, which can be computationally expensive.
8. **Overfitting Risk in Small Data:**
   - With small datasets, SOMs might adapt too closely to the training data, losing their generalization capability.

Ans- 6(a) **Sigmoid Activation Function**

The sigmoid activation function is a mathematical function used in neural networks to map input values to an output range of $(0, 1)$.

Mathematical Definition

The sigmoid function is defined as:

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

where:

- xxx is the input to the function (e.g., the weighted sum of inputs and biases in a neuron).
- eee is the base of the natural logarithm

2. Hyperbolic Tangent (Tanh) Activation Function

- $$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
- **Range:** (−1,1)(-1, 1)(−1,1)
- **Key Features:**
- Similar to sigmoid but zero-centered.
- Preferred over sigmoid in hidden layers to facilitate faster convergence.

Also suffers from the vanishing gradient problem.

- **Example:**
  In hidden layers of neural networks for tasks like time-series prediction.

3. Rectified Linear Unit (ReLU) Activation Function

  - **Definition:** f(x)=max⁡(0,x)f(x) = \max(0, x)

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

4. **Leaky ReLU Activation Function**

- **Definition**

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

Ans- 5(b) Explain sigmoid activation function in detail.

Sigmoid Activation Function

The **sigmoid function** is one of the earliest activation functions used in neural networks. It is defined as:

σ(x)=11+e−x$\sigma(x)=1+e-x1$

Advantages:

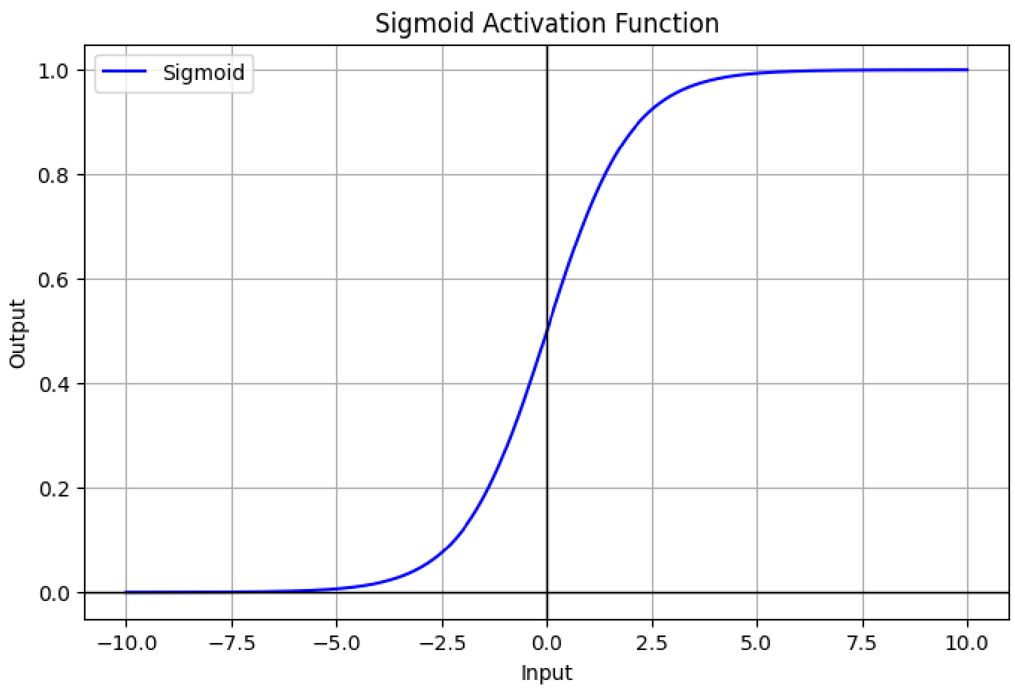
- Sigmoid is differentiable and smooth, making it suitable for backpropagation.
- Its output is in the range (0, 1), which is especially useful for binary classification problems.
- It works well in shallow networks with few layers.

Limitations:

- **Vanishing Gradient Problem**: For large positive or negative inputs, the gradient of the sigmoid function becomes very small, slowing down the learning process.
- **Non-zero centered output**: Since the output is between 0 and 1, this can cause issues during optimization by leading to slower convergence.

When to Use Sigmoid: *Sigmoid is most effective in the output layer of binary classification tasks where the output needs to be in the range of 0 to 1.*

Sigmoid Activation Function Plot:



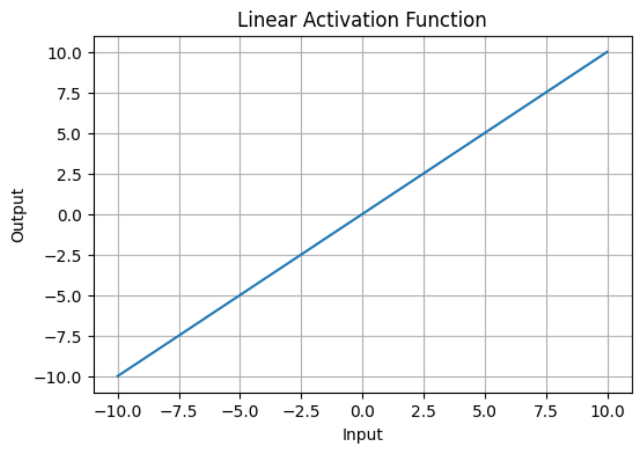
Ans-6 **What is an Activation Function?**
An activation function is a mathematical function applied to the output of a neuron. It introduces non-linearity into the model, allowing the network to learn and represent complex patterns in the data. Without this non-linearity feature, a neural network would behave like a linear regression model, no matter how many layers it has.
The activation function decides whether a neuron should be activated by calculating the weighted sum of inputs and adding a bias term. This helps the model make complex decisions and predictions by introducing non-linearities to the output of each neuron.
Why is Non-Linearity Important in Neural Networks?
Neural networks consist of neurons that operate using **weights**, **biases**, and **activation functions**.
In the learning process, these weights and biases are updated based on the error produced at the output—a process known as **backpropagation**. Activation functions enable backpropagation by providing gradients that are essential for updating the weights and biases.
Without non-linearity, even deep networks would be limited to solving only simple, linearly separable problems. Activation functions empower neural networks to model highly complex data distributions and solve advanced deep learning tasks. Adding non-linear activation functions introduce flexibility and enable the network to learn more complex and abstract patterns from data.
Mathematical Proof of Need of Non-Linearity in Neural Networks
To illustrate the need for non-linearity in neural networks with a specific example, let's consider a network with two input nodes (i1and i2)($i1$and $i2$), a single hidden layer containing one neuron (h1)($h1$), and an output neuron (out). We will use w1,w2$w1$,$w2$ as weights connecting the inputs to the hidden neuron, and w5$w5$ as the weight connecting the hidden neuron to the output.
Network Structure

1. **Input Layer**: Two inputs i1$i1$ and i2$i2$.
2. **Hidden Layer**: One neuron h1$h1$.
3. **Output Layer**: One output neuron.

Types of Activation Functions in Deep Learning
1. Linear Activation Function
Linear Activation Function resembles straight line define by y=x. No matter how many layers the neural network contains, if they all use linear activation functions, the output is a linear combination of the input.

- The range of the output spans from (−∞ to +∞)($-\infty$ to $+\infty$).
- **Linear activation function** is used at just one place i.e. output layer.
- Using linear activation across all layers makes the network's ability to learn complex patterns limited.

Linear activation functions are useful for specific tasks but must be combined with non-linear functions to enhance the neural network's learning and predictive capabilities.



Tanh Activation Function
Tanh function or hyperbolic tangent function, is a shifted version of the sigmoid, allowing it to stretch across the y-axis. It is defined as:

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

Alternatively, it can be expressed using the sigmoid function:

$$\tanh(x) = 2 \times \text{sigmoid}(2x) - 1$$

- **value Range**: Outputs values from -1 to +1.
- **Non-linear**: Enables modeling of complex data patterns.
- **Use in Hidden Layers**: Commonly used in hidden layers due to its zero-centered output, facilitating easier learning for subsequent layers.

4. ReLU (Rectified Linear Unit) Function

ReLU activation is defined by A(x)=max⁡(0,x)*A*(*x*)=max(0,*x*), this means that if the input x is positive, ReLU returns x, if the input is negative, it returns 0.

- **Value Range**: [0,∞)[0,∞), meaning the function only outputs non-negative values.
- **Nature**: It is a **non-linear** activation function, allowing neural networks to learn complex patterns and making backpropagation more efficient.
- **Advantage over other Activation:** ReLU is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.