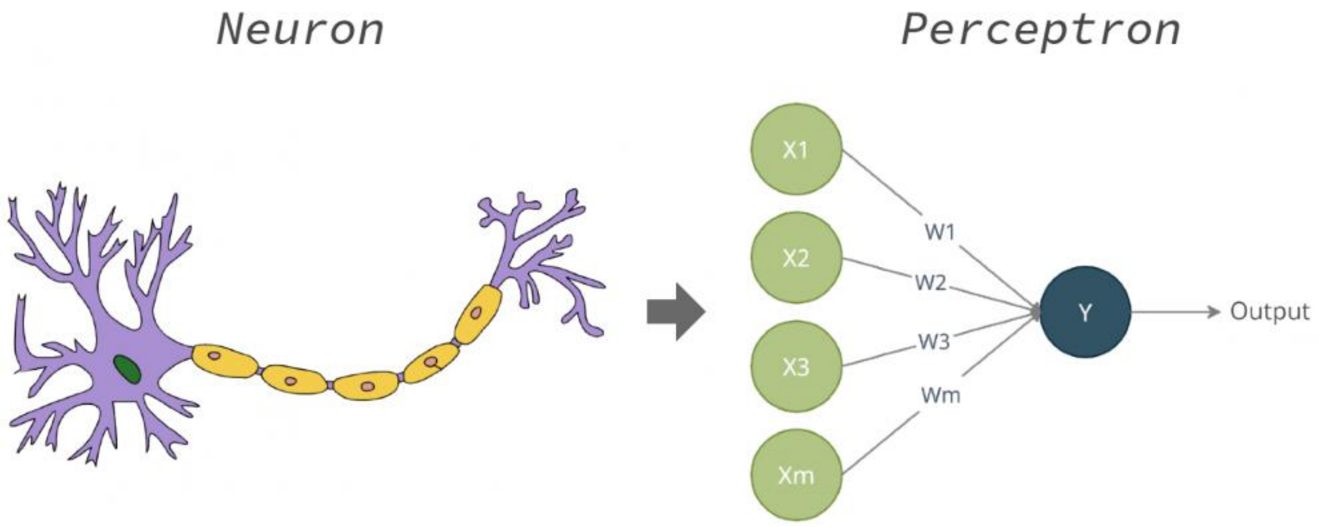
Introduction to Deep Learning

MUKESH KUMAR

- Perceptron structure
- Weights and Bias
- Activation Functions
- NN Structure
- Error and Loss Functions
- Gradient Descent
- Forward propagation
- Backward Propagation
- Learning Rate
- Early Stopping

Neuron Vs Perceptron

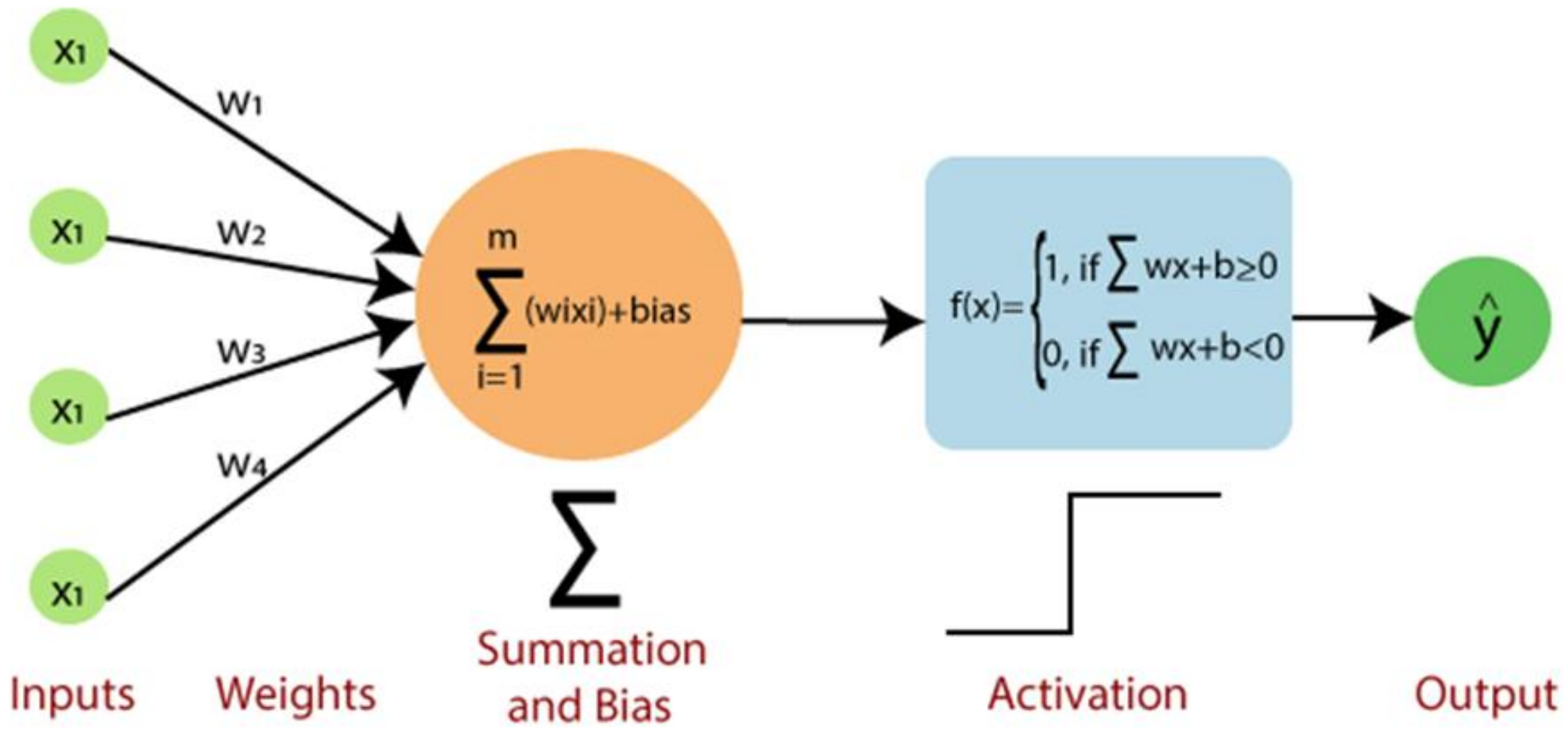
- Neurons are interconnected nerve cells that process and transmits electrochemical signals in the brain



Perceptron?

- A perceptron is an artificial neuron.
- Instead of electrochemical signals, data is represented as numerical values.
- input values, normally labeled as X are multiplied by weights (W) and then added to represent the input values' total strength.
- If this weighted sum exceeds the threshold, the perceptron will trigger, sending a signal.

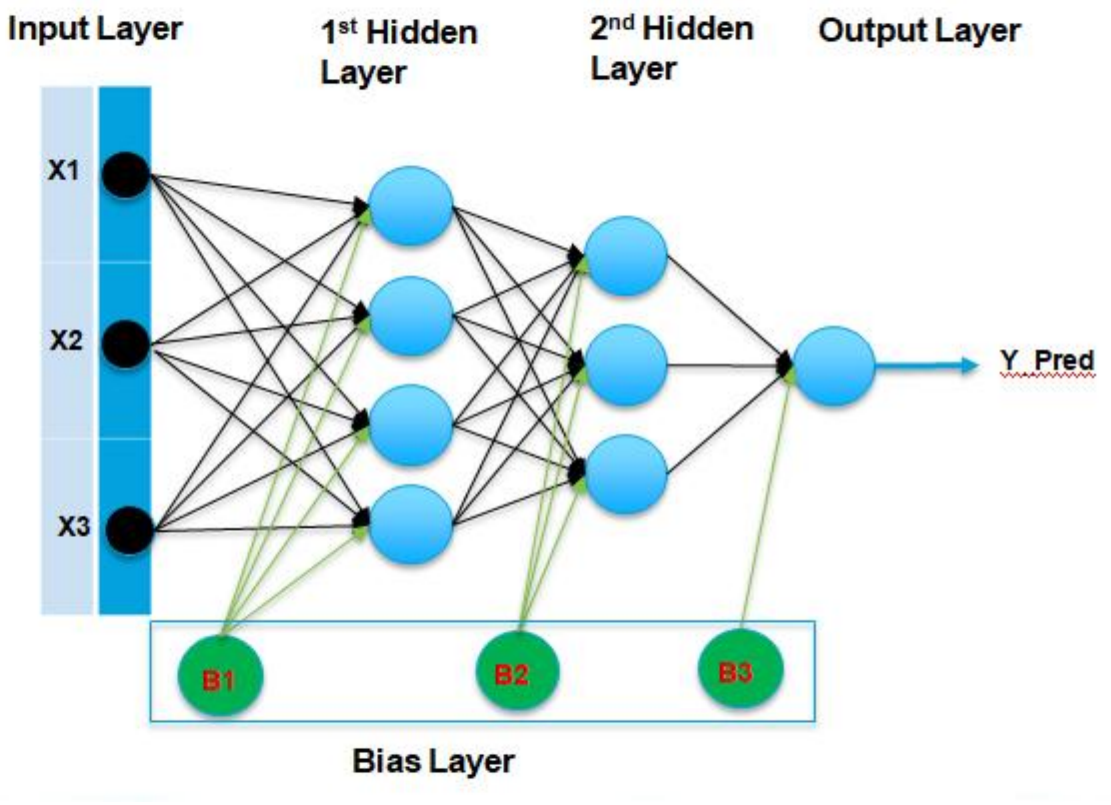
Perceptron



x_1
x_1
x_1
x_1
w_1
w_2
w_3
w_4
$\sum_{i=1}^{m}(w_ix_i)+\text{bias}$
$f(x)=\begin{cases}1, \text{ if } \sum wx+b\geq 0\\ 0, \text{ if } \sum wx+b<0\end{cases}$
$\hat{y}$
$\sum$
Inputs
Weights
Summation and Bias
Activation
Output

Artificial Neural Network

- ANN models are based on the observed behaviour of neural nets in our brains



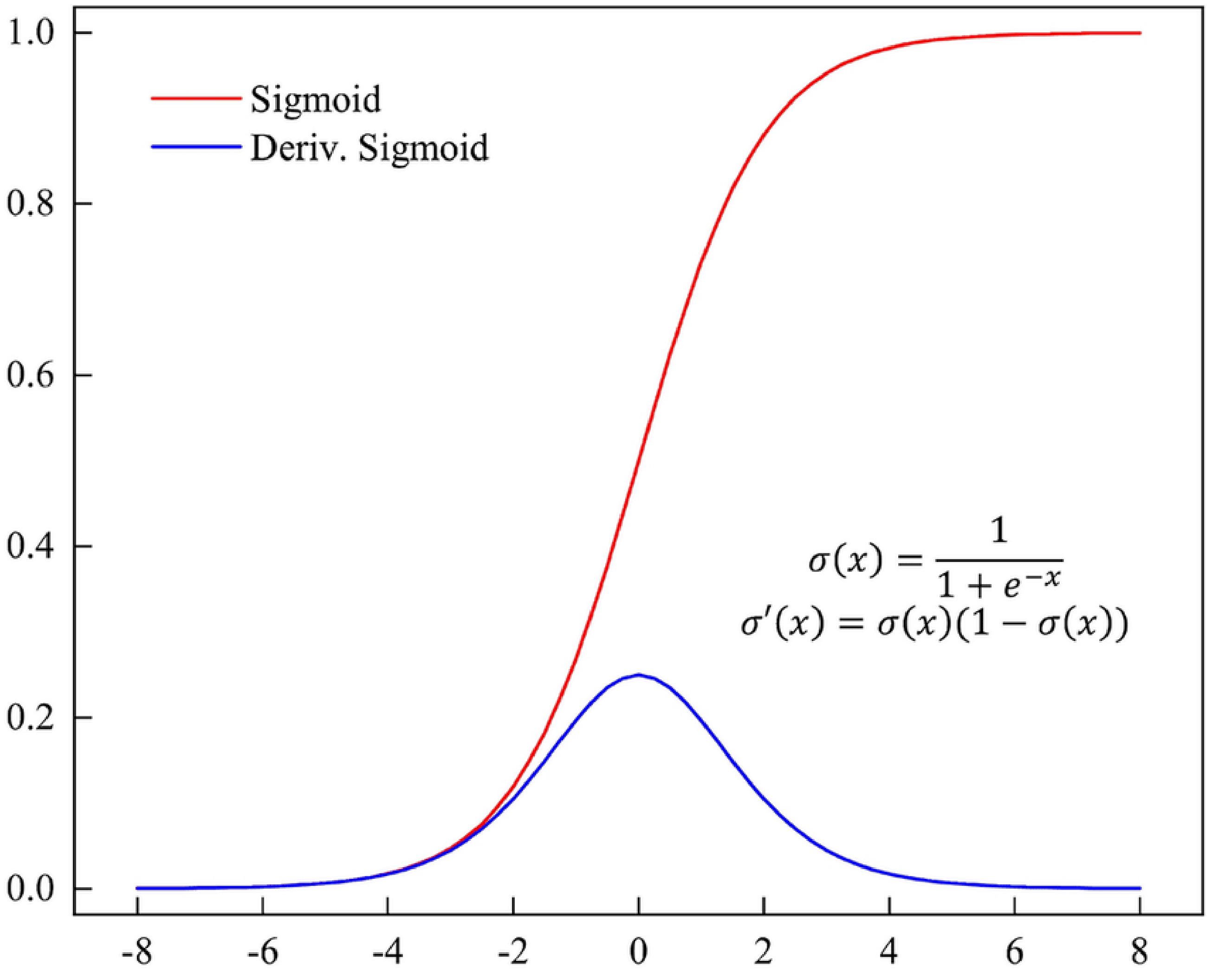
What is Activation Function?

- In artificial neural network, each neuron forms a weighted sum of its inputs and passes the resulting scalar value through a function referred to as an activation function if a neuron has n inputs, then the output or activation of a neuron is.
- ***Output = Activation Function (Weighted Sum of Inputs + Bias)***

A Good Activation Function

- It should be non-linear (sigmoid)
- It can capture non-linear patterns in dataset
- It should be differentiable otherwise model will not train
- Should be fast , computational inexpensive so that loss can be calculated quickly
- Should be zero centered (tanh).
- Non saturating else it will cause vanishing gradient
  - Example of saturating: sigmoid, tanh
  - Non saturating example: Relu

Sigmoid

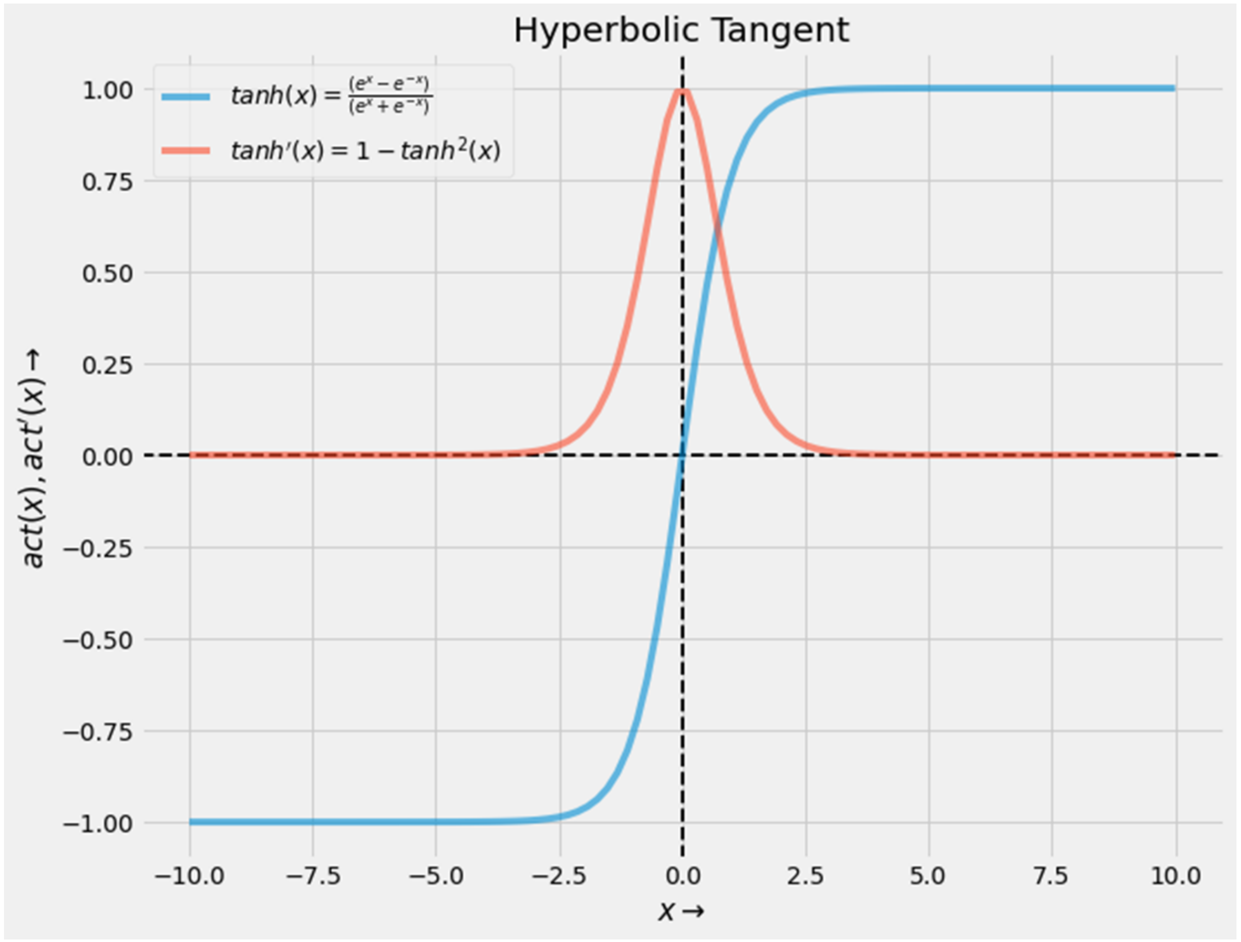


$$\sigma(x) = \frac{1}{1+e^{-x}}$$
$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

Sigmoid

- Advantages:
  - Can be used for probabilities for binary classification
  - Non-linear function
  - Differentiable
- Disadvantage:
  - Saturating : causes vanishing gradients as for bigger values derivative close to 0.

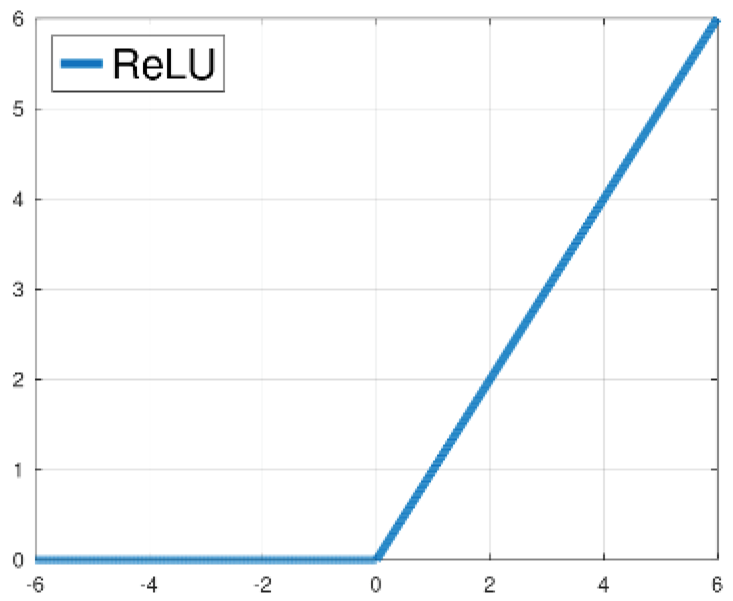
TANH

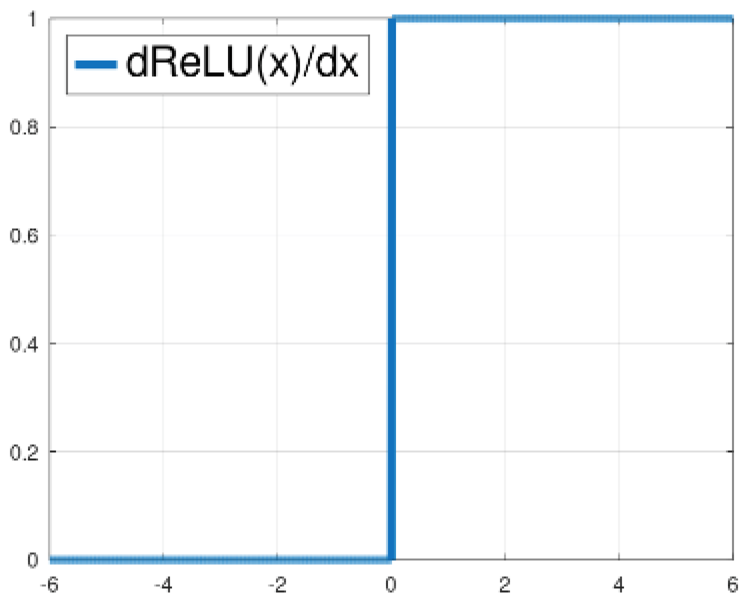

Hyperbolic Tangent
$tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$
$tanh'(x) = 1 - tanh^2(x)$
$act(x), act'(x) \rightarrow$
$x \rightarrow$
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

TANH

- Similar to sigmoid except its centered around zero so convergence is fast

RELU





RELU

- **Advantages:**
  - **Sparsity and Efficiency**: ReLU is computationally efficient to compute since it simply replaces negative values with zero. This sparsity can lead to faster convergence during training
  - **Avoids Vanishing Gradient Problem**: ReLU helps mitigate the vanishing gradient problem, which is common in deep networks trained with sigmoid or tanh activations. By allowing non-zero gradients for positive inputs.
  - **Non-saturating**: ReLU does not saturate for positive inputs, unlike sigmoid and tanh functions, which become saturated at large positive or negative values. This property can help prevent the model from getting stuck during training.

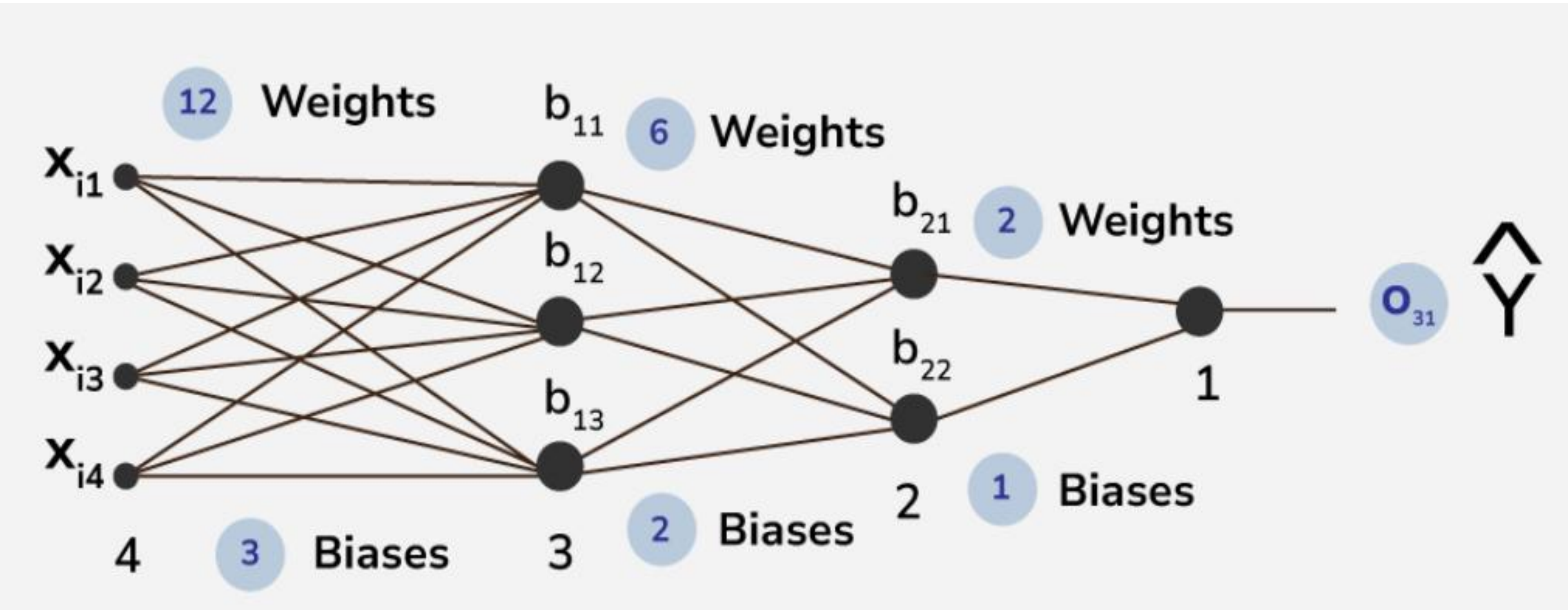
RELU

- **Disadvantages:**
- **Dying ReLU Problem**: Neurons using ReLU can sometimes become "dead" during training, meaning they always output zero. This occurs when the input to the ReLU is consistently negative, causing the neuron to never activate and update its weights. Once a neuron is in this state, it may never recover, leading to wasted computational resources and reduced model capacity.
- **Unbounded Activation**: ReLU is unbounded on the positive side, which can lead to exploding activations during training. This can sometimes cause instability during training if not appropriately addressed with techniques like weight regularization or gradient clipping.
- **Not Centered Around Zero**: Unlike activation functions such as tanh or sigmoid, ReLU is not centered around zero. This can lead to issues when dealing with data centered around zero, as the output of ReLU will be shifted towards the positive side.

Forward Propagation

- Forward propagation is the process in a neural network where the input data is passed through the network's layers to generate an output.

Forward Propagation Cont..



Forward Propagation Cont..

- There are total 26 trainable parameters
- Here, considering we are using sigmoid function as the activation function
- **The output prediction function is:** $\sigma(w$Tx+$b)\sigma(w$Tx+$b)$re

Inside Layer 1

$$\sigma\left(\begin{bmatrix} W_{111} & W_{112} & W_{113} \\ W_{121} & W_{122} & W_{123} \\ W_{131} & W_{132} & W_{133} \\ W_{141} & W_{142} & W_{143} \end{bmatrix}^T \begin{bmatrix} X_{i1} \\ X_{i2} \\ X_{i3} \\ X_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}\right) =$$

$$\sigma\left(\begin{bmatrix} W_{111}X_{i1} + W_{121}X_{i2} + W_{131}X_{i3} + W_{141}X_{i4} \\ W_{112}X_{i1} + W_{122}X_{i2} + W_{132}X_{i3} + W_{142}X_{i4} \\ W_{113}X_{i1} + W_{123}X_{i2} + W_{133}X_{i3} + W_{143}X_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}\right) =$$

$$\sigma\left(\begin{bmatrix} W_{111}X_{i1} + W_{121}X_{i2} + W_{131}X_{i3} + W_{141}X_{i4} + b_{11} \\ W_{112}X_{i1} + W_{122}X_{i2} + W_{132}X_{i3} + W_{142}X_{i4} + b_{12} \\ W_{113}X_{i1} + W_{123}X_{i2} + W_{133}X_{i3} + W_{143}X_{i4} + b_{13} \end{bmatrix}\right) = \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix} = a^{[1]}$$

- After the completion of the matrix operations, the one-dimensional matrix has been formed as $[O^{11}O^{12}O^{13}][O^{11}O^{12}O^{13}]$which is continued as **a**$^{[1]}$ known as the output of the first hidden layer.

Inside Layer 2

$$
\begin{aligned}
&\sigma\left(\begin{bmatrix} W_{211} & W_{212} \\ W_{221} & W_{222} \\ W_{231} & W_{232} \end{bmatrix}^T \cdot \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} W_{211} \cdot O_{11} + W_{221} \cdot O_{12} + W_{231} \cdot O_{13} \\ W_{212} \cdot O_{11} + W_{222} \cdot O_{12} + W_{232} \cdot O_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}\right) \\
&= \sigma\left(\begin{bmatrix} W_{211} \cdot O_{11} + W_{221} \cdot O_{12} + W_{231} \cdot O_{13} + b_{21} \\ W_{212} \cdot O_{11} + W_{222} \cdot O_{12} + W_{232} \cdot O_{13} + b_{22} \end{bmatrix}\right) \\
&= \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} \\
&= a^{[2]}
\end{aligned}
$$

- The output of the first hidden layer and the weights of the second hidden layer will be computed under the previously stated prediction function which will be matrix manipulated with those biases to give us the output of the second hidden layer i.e. the matrix [O21O22][O21O22]

Output Layer

$$\sigma\left(\begin{bmatrix} W_{311} \\ W_{321} \end{bmatrix}^T \cdot \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} + [b_{31}]\right)$$
$$= \sigma\left([W_{311} \cdot O_{21} + W_{321} \cdot O_{22}] + [b_{31}]\right)$$
$$= \sigma\left(W_{311} \cdot O_{21} + W_{321} \cdot O_{22} + b_{31}\right)$$
$$= Y^i = [O_{31}]$$
$$= a^{[3]}$$

- Output is $a^{[3]}$ or Y^i (predicted value)

Final Equation

Now as for the complete equation , we can conclude as follows :

$$\sigma(a[0] * W[1] + b[1]) = a[1]\sigma(a[1] * W[2] + b[2]) = a[2]\sigma(a[2] * W[3] + b[3]) = a[3]$$

which concludes us to :

$$\sigma(\sigma((\sigma(a[0] * W[1] + b[1]) * W[2] + b[2]) * W[3] + b[3]) = a[3]$$

Backward Propagation

- Run these steps for n epocs:
  - Select a random row from dataset
  - Using Forward propagation predict
  - Calculate loss using loss function(MSE,Cross-Entropy)
  - Update Weights & Biases using Gradient Descent
  - Calculate the average loss