NEURAL NETWORK

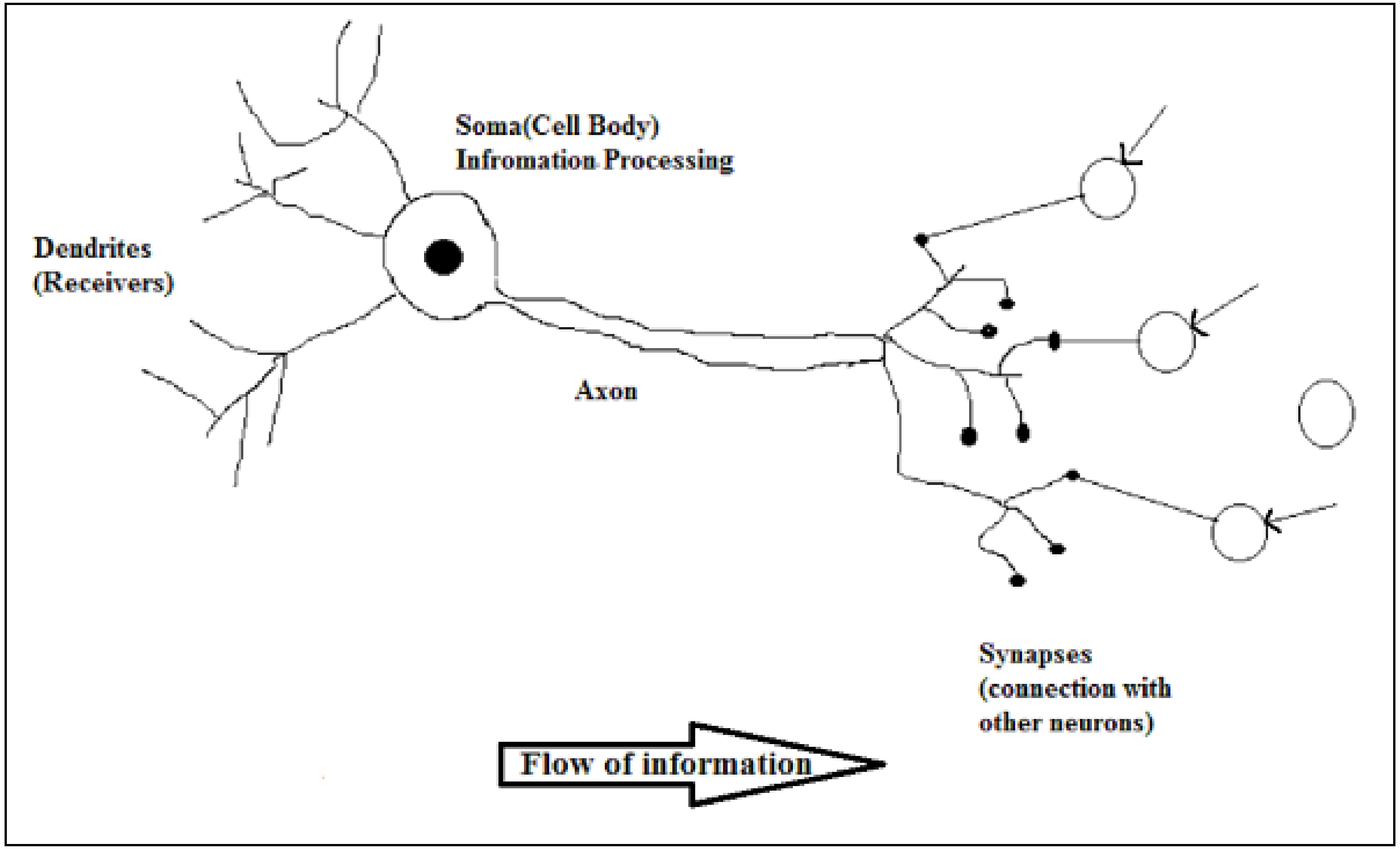
Dr. Devender Kumar

System and Architecture

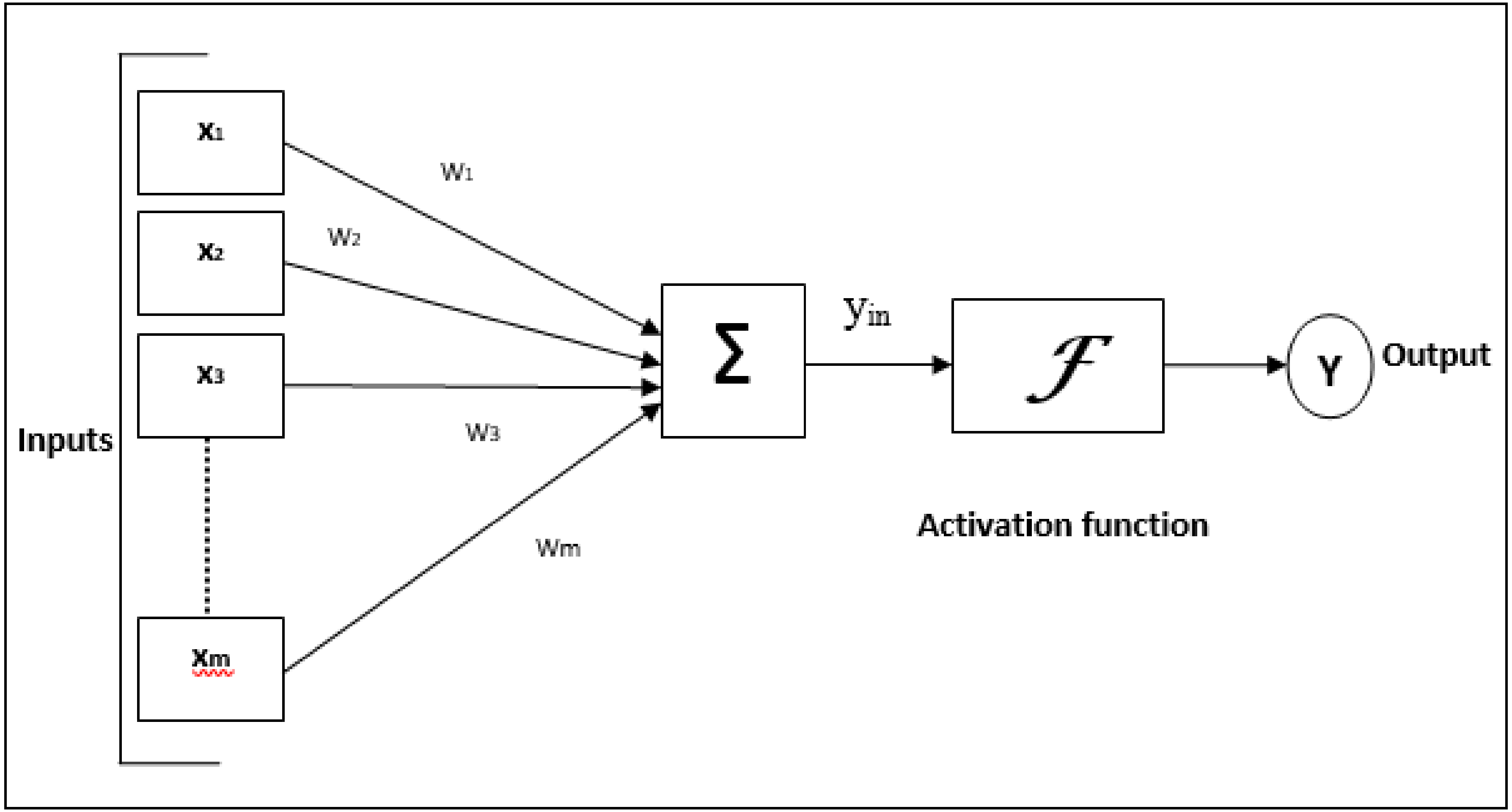
ARTIFICIAL NEURAL NETWORK ANN

- Computing system based on the analogy of biological neural networks.
- ANN acquires a large collection of units that are interconnected in some pattern to allow communication between the units.
- These units, also referred to as nodes or neurons, are simple processors which operate in parallel.
- Every neuron is connected with another neuron through a connection link.
- Each connection link is associated with a weight that has information about the input signal.
- This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated.
- Each neuron has an internal state, which is called an activation signal.
- Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

BIOLOGICAL NEURON

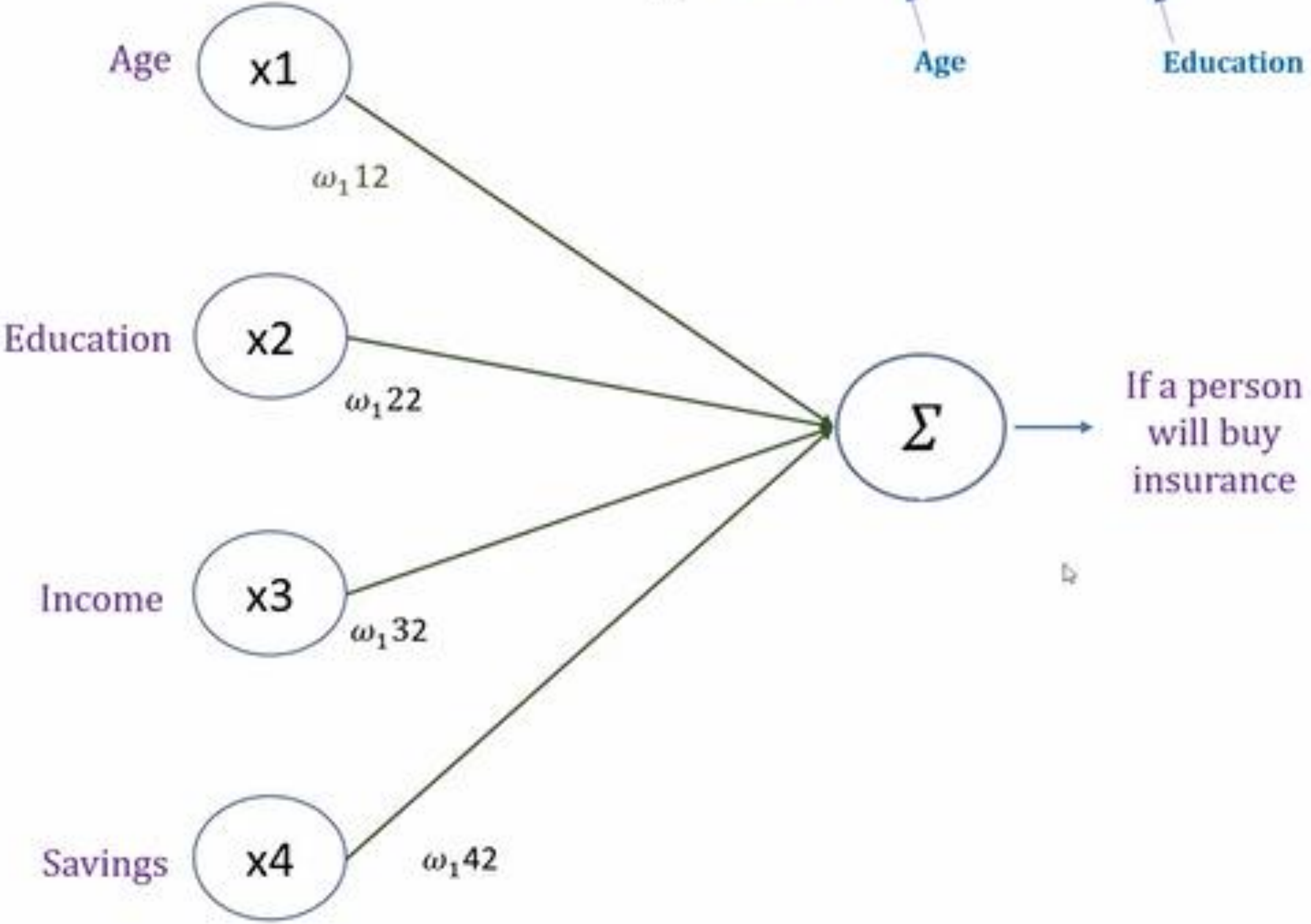


MODEL OF ARTIFICIAL NEURAL NETWORK



$$y = 0.042 * x1 + 0.008 * x2 + 0.2 * x3 + 0.02 * x4 - 1.53$$

Age Education Income Savings



MODEL OF ARTIFICIAL NEURAL NETWORK

- For the above general model of the artificial neural network, the net input can be calculated as follows –

$$y_{in} = x_1.w_1 + x_2.w_2 + x_3.w_3 \dots x_m.w_m$$

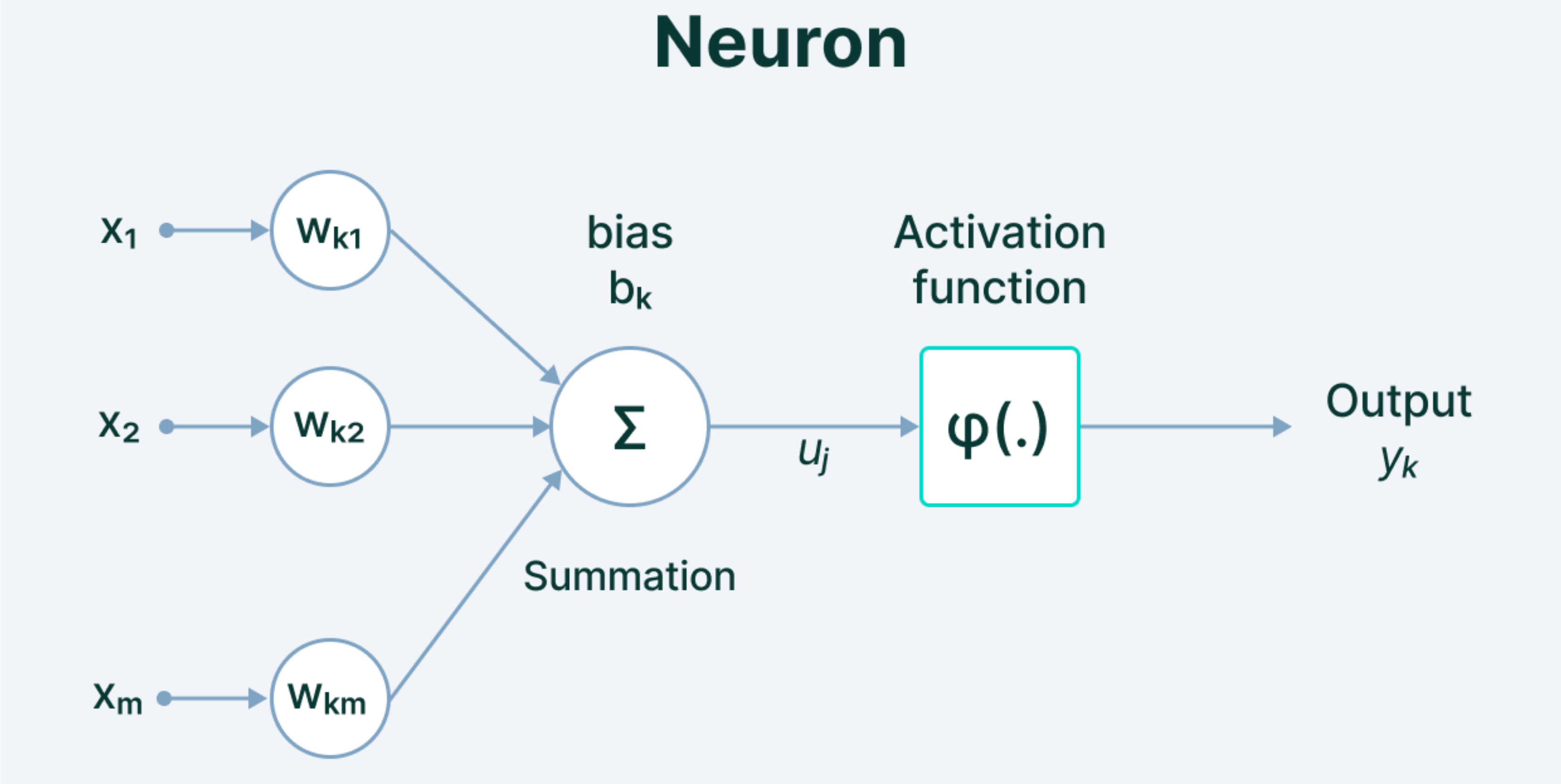
i.e., Net input $y_{in} = \sum_i^m x_i.w_i$

- The output can be calculated by applying the activation function over the net input.

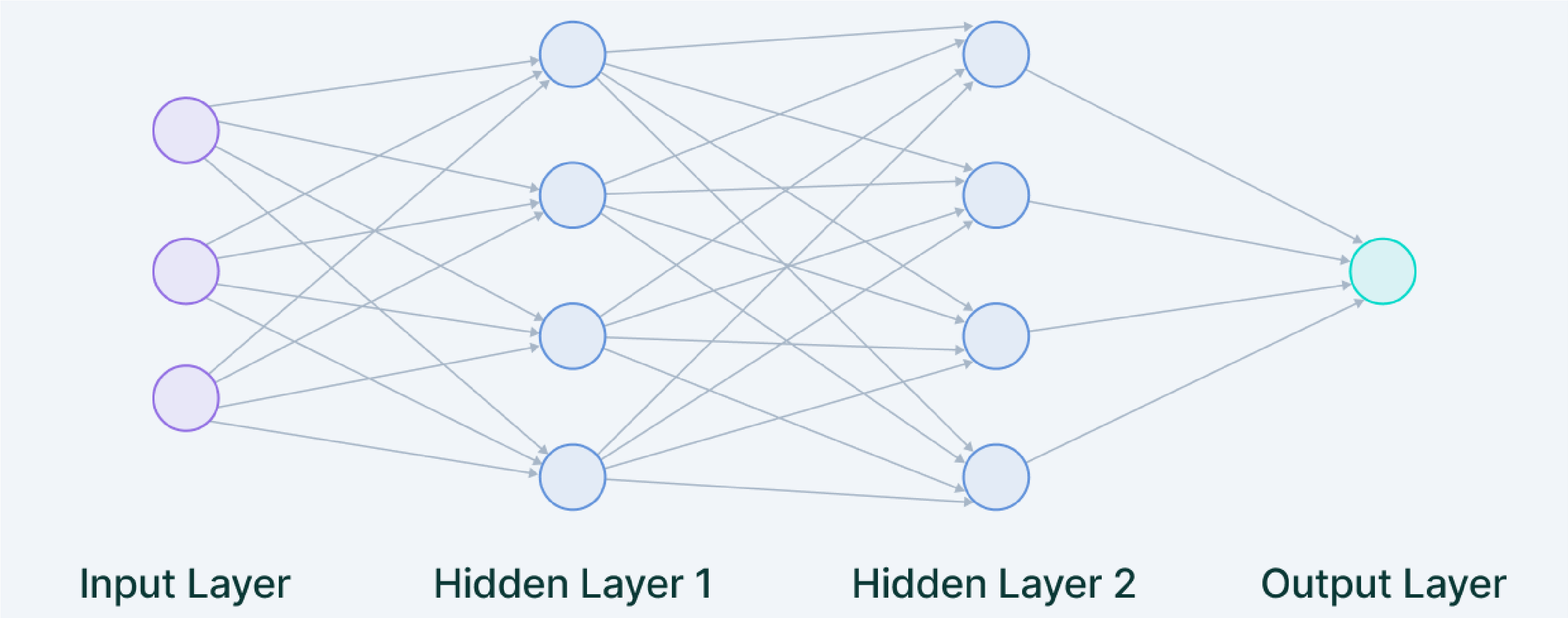
$$Y = F(y_{in})$$

Output = function $netinputcalculated$

KEY COMPONENTS OF THE NEURAL NETWORK ARCHITECTURE



KEY COMPONENTS OF THE NEURAL NETWORK ARCHITECTURE



FEED-FORWARD NETWORKS

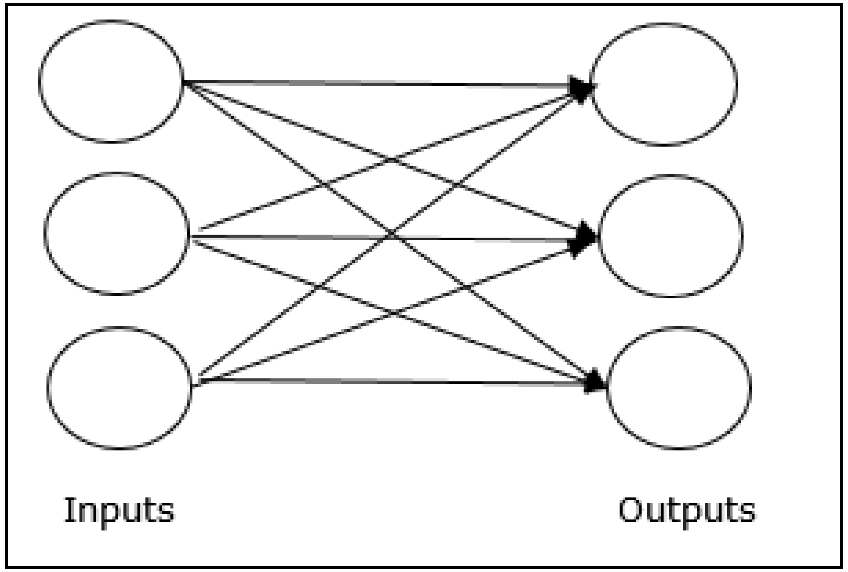
- Perceptron represents how a single neuron works.
- What about a series of perceptrons stacked in a row and piled in different layers? How does the model learn then?
- It is a multi-layer Neural Network, and, as the name suggests, the information is passed in the forward direction—from left to right.
- In the forward pass, the information comes inside the model through the input layer, passes through the series of hidden layers, and finally goes to the output layer.
- This Neural network architecture is forward in nature—the information does not loop with two hidden layers.
- The later layers give no feedback to the previous layers. The basic learning process of Feed-Forward Networks remains the same as the perceptron.

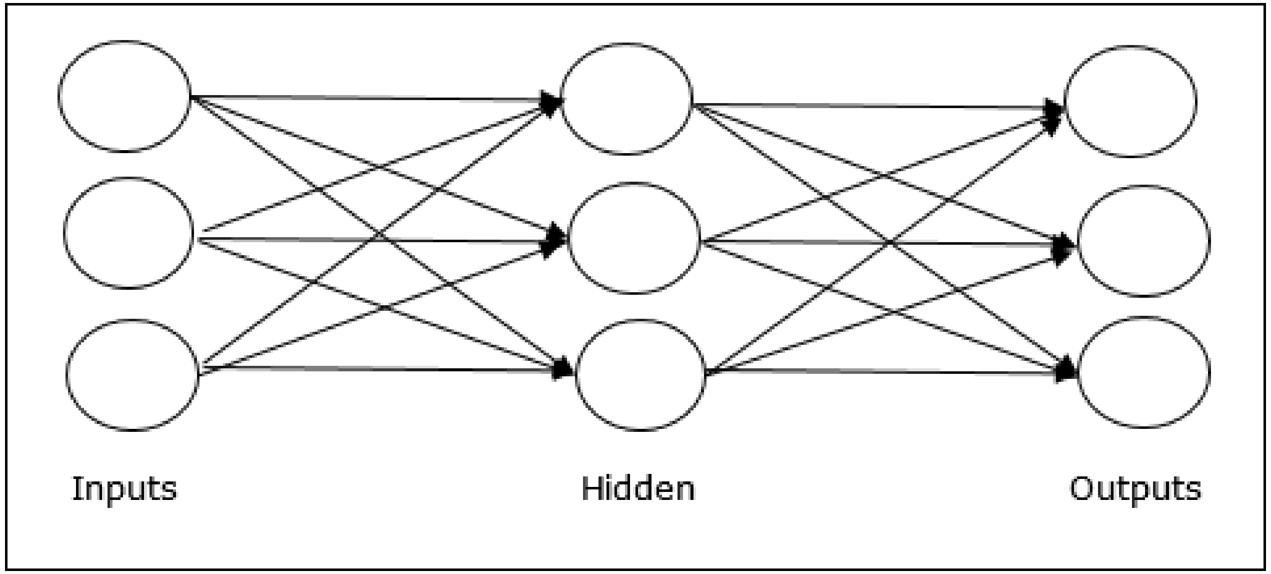
BUILDING BLOCKS

- Network Topology
- Adjustments of Weights or Learning
- Activation Functions

NETWORK TOPOLOGY - FEEDFORWARD NETWORK

- A network topology is the arrangement of a network along with its nodes and connecting lines.





FEEDBACK NETWORK

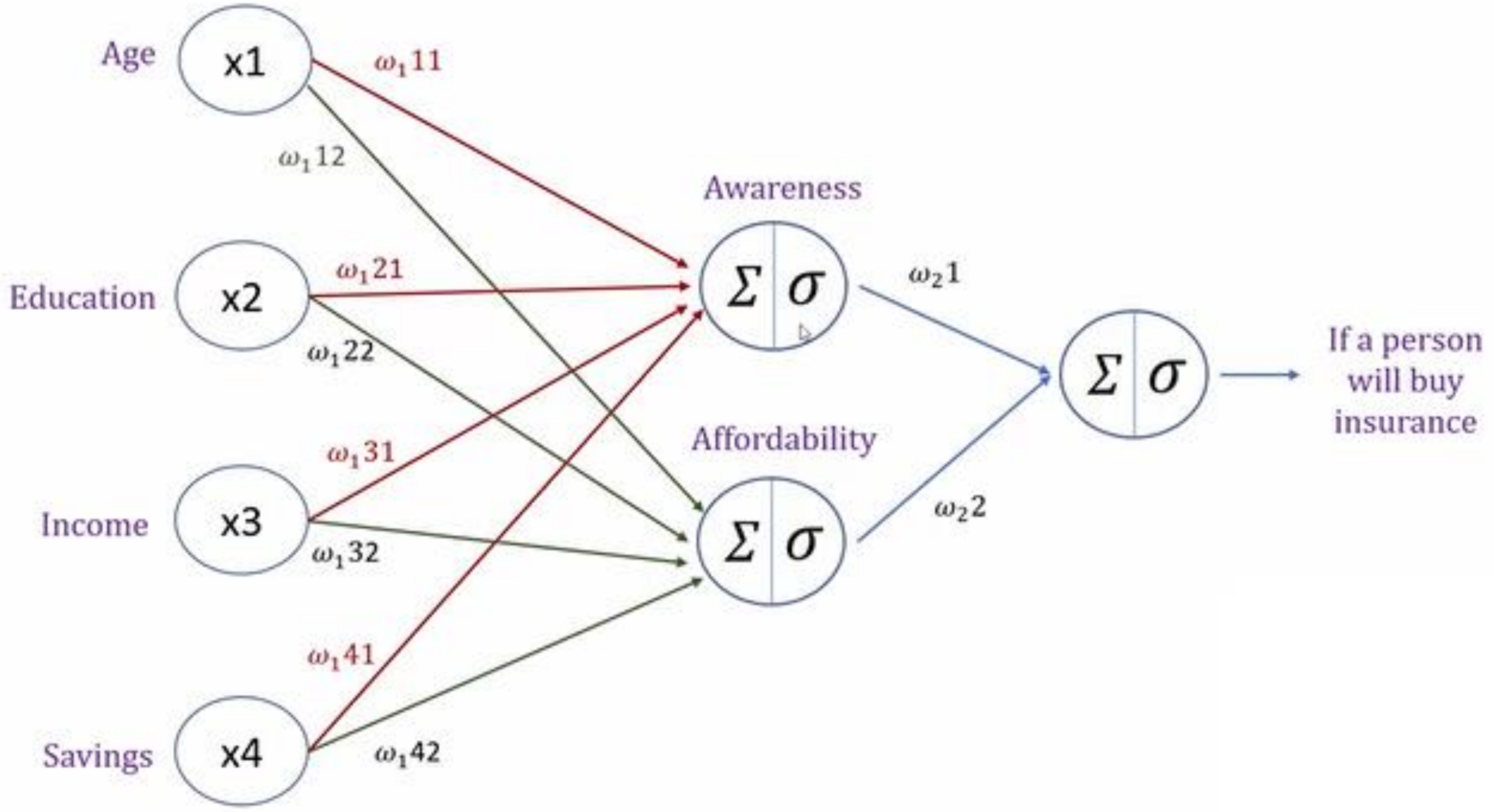
- A feedback network has feedback paths, which means the signal can flow in both directions using loops.
- This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium.
- **Recurrent networks** – They are feedback networks with closed loops. Following are the two types of recurrent networks.
- **Fully recurrent network** – It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.
- **Jordan network** – It is a closed loop network in which the output will go to the input again as feedback as shown in the following diagram.

ADJUSTMENTS OF WEIGHTS OR LEARNING

Supervised Learning

Unsupervised Learning

Reinforcement Learning



Age
x1
Education
x2
Income
x3
Savings
x4
$\omega_1 11$
$\omega_1 12$
$\omega_1 21$
$\omega_1 22$
$\omega_1 31$
$\omega_1 32$
$\omega_1 41$
$\omega_1 42$
Awareness
Σ σ
Affordability
Σ σ
$\omega_2 1$
$\omega_2 2$
Σ σ
If a person will buy insurance

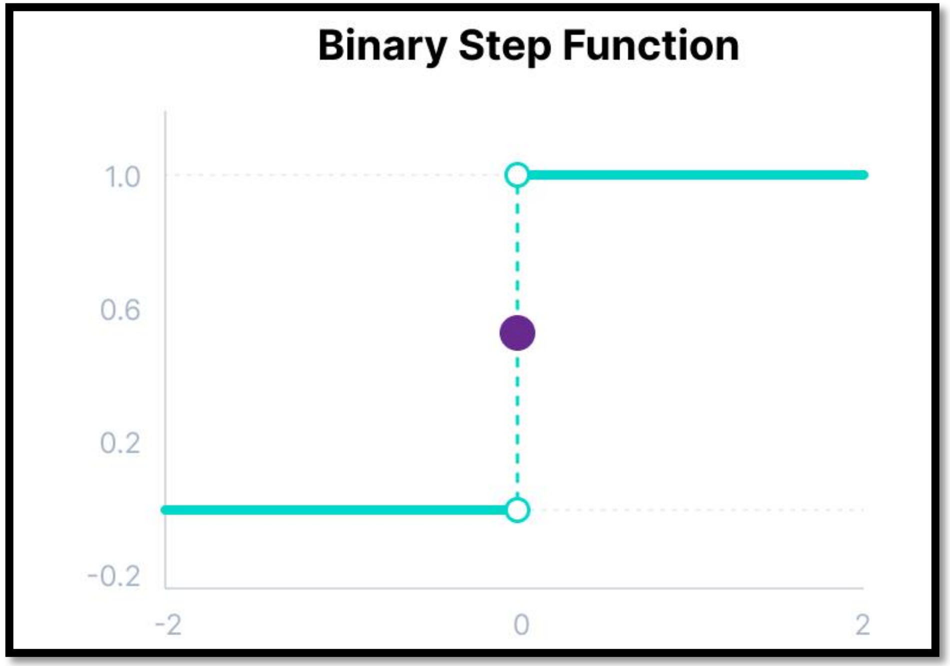
ACTIVATION FUNCTIONS

- An activation function is a mathematical function that determines the output of a neuron in a neural network.
- It takes the weighted sum of inputs and adds a bias term, then applies a non-linear transformation to produce the neuron's output.
- Activation functions introduce non-linearity into the neural network, enabling it to learn and represent complex patterns in the data.
- Without activation functions, neural networks would reduce to linear transformations, limiting their ability to learn and generalize from data.

BINARY STEP ACTIVATION FUNCTION/THRESHOLD FUNCTION

- The step function is the simplest activation function, where the output is binary: either 0 or 1, depending on whether the input is above or below a certain threshold.

Equation: $f(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$



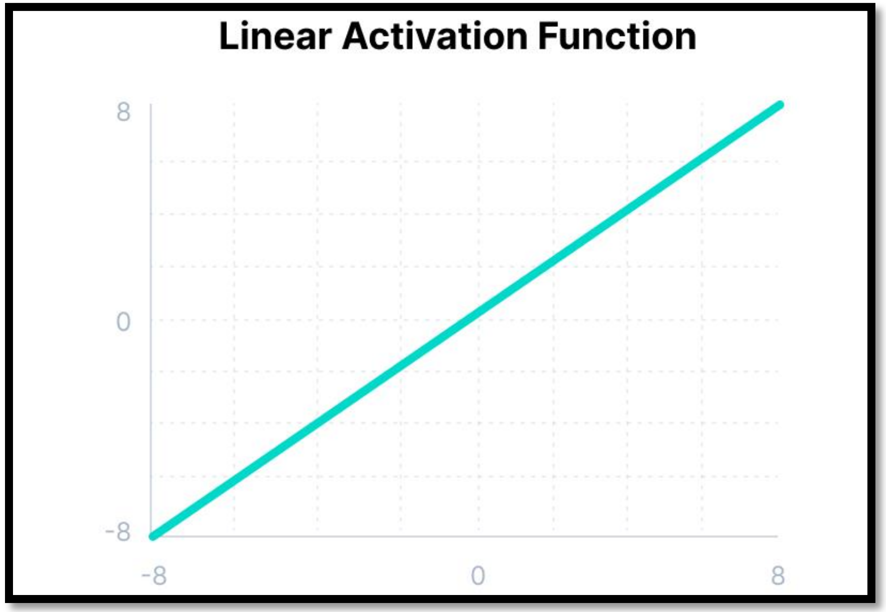
- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

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.

$$F(x) = x$$

- It's not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
- All layers of the neural network will collapse into one if a linear activation function is used.



SIGMOID ACTIVATION FUNCTION

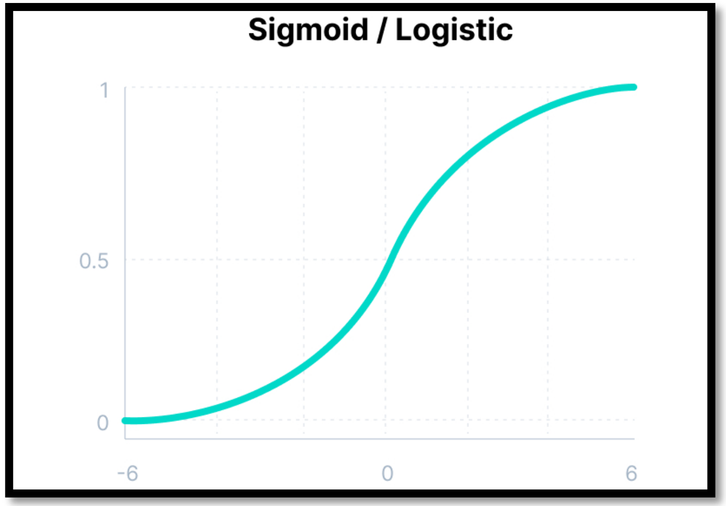
- The sigmoid function produces an S-shaped curve, squashing the input values between 0 and 1. It's commonly used in binary classification problems.
- Binary sigmoidal function

$$F(x) = sigm(x) = \frac{1}{1 + exp(-x)}$$

- Bipolar sigmoidal function

$$F(x) = sigm(x) = \frac{2}{1 + exp(-x)} - 1 = \frac{1 - exp(x)}{1 + exp(x)}$$

- As the gradient value approaches zero, the network ceases to learn and suffers from the Vanishing gradient problem.

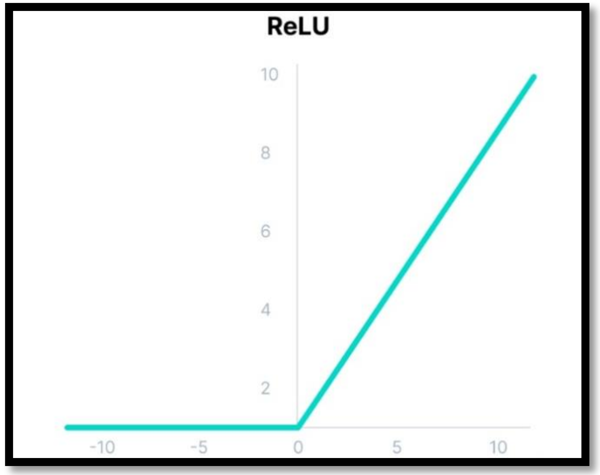


ReLU (Rectified Linear Unit) Activation Function

- The ReLU function returns the input value if it's positive, and zero otherwise. It's the most widely used activation function in deep learning due to its simplicity and effectiveness.
- The main catch here is that the ReLU function does not activate all the neurons at the same time.

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

- The Dying ReLU problem
- Can create dead neurons that never get activated.

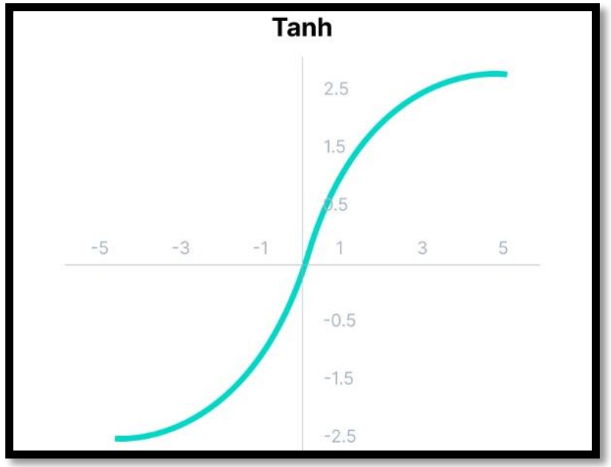


TANH FUNCTION

- The hyperbolic tangent (tanh) function is similar to the sigmoid function but symmetric around the origin.
- Tanh function is very similar to the sigmoid/logistic activation function and even has the same S-shape with the difference in the output range of -1 to 1.

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

- Usually used in hidden layers of a neural network as its values lie between -1 to.
- It helps in centering the data and makes learning for the next layer much easier.
- faces the problem of vanishing gradients



CHOOSING THE RIGHT ACTIVATION FUNCTION

Sigmoid:	ReLU:	Tanh:
• Suitable for binary classification tasks where the output needs to be between 0 and 1.	• Recommended for most hidden layers in deep neural networks due to its simplicity and effectiveness.	• Similar to sigmoid but centered at zero, making it suitable for outputs that need to range from -1 to 1.

FORWARD PROPAGATION IN MACHINE LEARNING

$x = a^{(1)}$ *Input layer*

$z^{(2)} = W^{(1)}x + b^{(1)}$ *neuron value at* $Hidden_1$ *layer*

$a^{(2)} = f(z^{(2)})$ *activation value at* $Hidden_1$ *layer*

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ *neuron value at* $Hidden_2$ *layer*

$a^{(3)} = f(z^{(3)})$ *activation value at* $Hidden_2$ *layer*

$s = W^{(3)}a^{(3)}$ *Output layer*

BACKPROPAGATION IN MACHINE LEARNING

- According to the paper from 1989, backpropagation:
- *"repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector."*
- *"the ability to create useful new features distinguishes back-propagation from earlier, simpler methods…"*
- Backpropagation aims to minimize the cost function by adjusting network's weights and biases.
- The level of adjustment is determined by the gradients of the cost function concerning those parameters.

GRADIENT

- The gradient is a measure of how much a function changes as you move in different directions. In other words, it tells you the direction and rate of change of a function at a particular point.
- In machine learning, the gradient is used to optimize functions, such as loss functions, to find the minimum or maximum values.
- It helps algorithms like gradient descent determine which way to adjust parameters to minimize errors or maximize performance.

BACKPROPAGATION IN MACHINE LEARNING

$$\frac{\partial C}{\partial w_{jk}^{l}} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \qquad \textit{chain rule}$$

$$z_j^l = \sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l \qquad \textit{by definition}$$

m – *number of neurons in* $l-1$ *layer*

$$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \qquad \textit{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial w_{jk}^{l}} = \frac{\partial C}{\partial z_j^l} a_k^{l-1} \qquad \textit{final value}$$

BACKPROPAGATION IN MACHINE LEARNING

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \qquad \textit{chain rule}$$

$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \qquad \textit{by differentiation (calculating derivative)}$$

$$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}1 \qquad \textit{final value}$$

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \qquad \textit{local gradient}$$

BACKPROPAGATION IN MACHINE LEARNING

$$while\ (termination\ condition\ not\ met)$$

$$w := w - \epsilon \frac{\partial C}{\partial w}$$

$$b := b - \epsilon \frac{\partial C}{\partial b}$$

$$end$$

BACKPROPAGATION IN MACHINE LEARNING