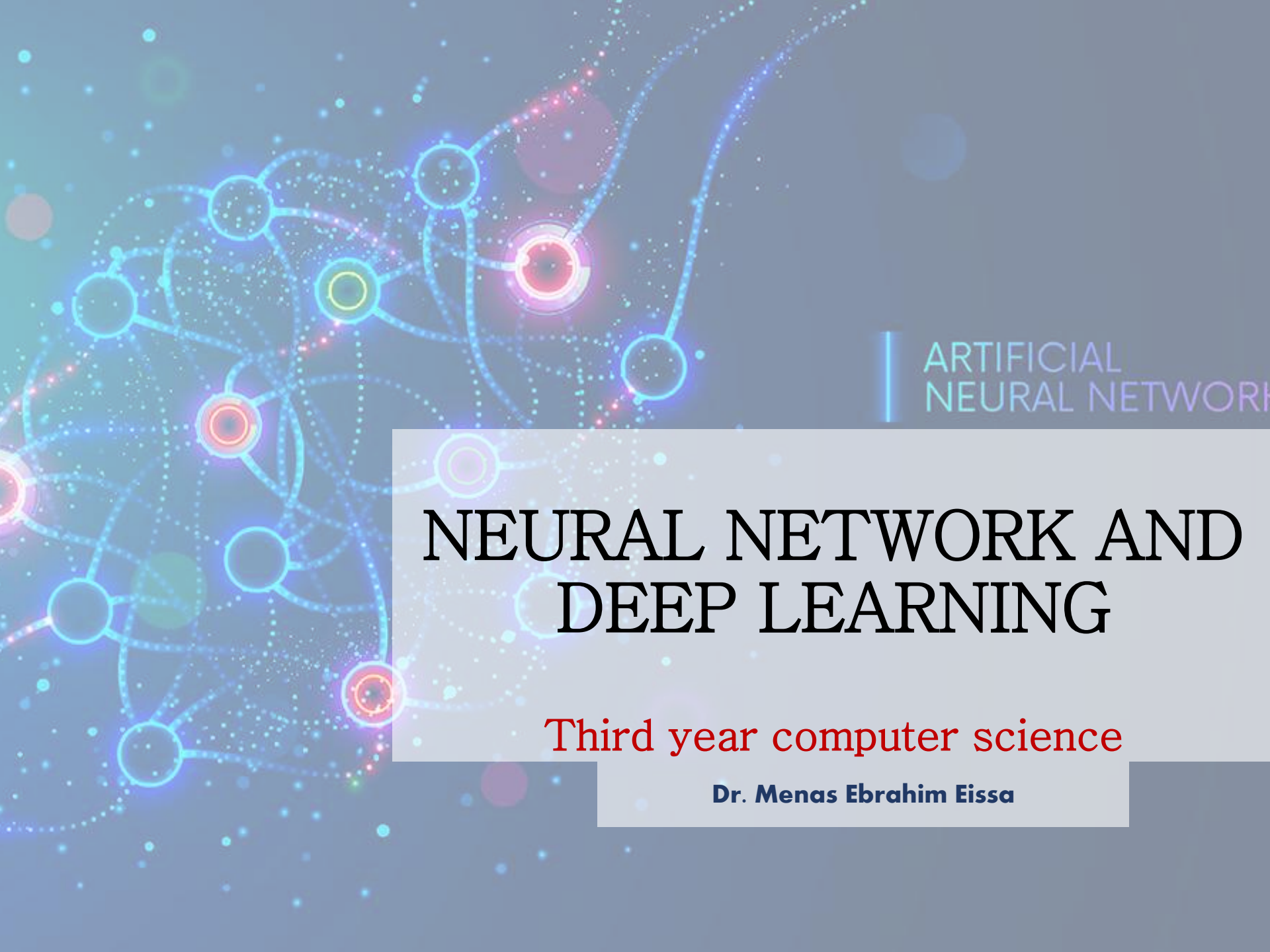


NEURAL NETWORK AND DEEP LEARNING

Third year computer science

Dr. Menas Ebrahim Eissa

What is Artificial Neural Network?

Definition:

- The term "**Artificial neural network**" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. (it is the heart of deep NN)
- An artificial neuron network (neural network) is a computational model **that mimics the way nerve cells work in the human brain**.

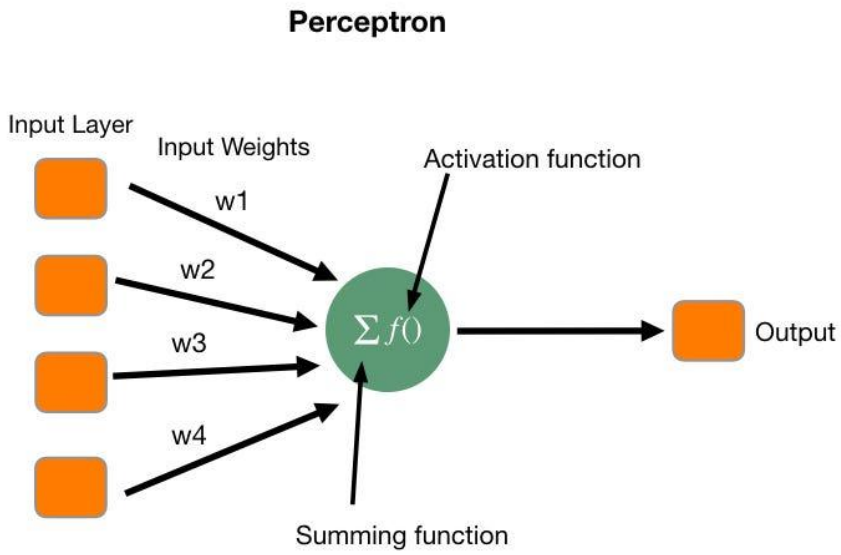
Artificial Neurons

- Artificial neuron also known as perceptron is the basic unit of the neural network.
- In simple terms, it is a mathematical function based on a model of biological neurons. It can also be seen as a simple logic gate with binary outputs.

Basic Parts of simple NN

- The perceptron(neuron) consists of 4 parts:

1. **Input layer**
   - We pass input values to a neuron using this layer. It might be something as simple as a collection of array values. It is similar to a dendrite in biological neurons.
2. **Weights and Bias**
   - Weights are a collection of array values which are multiplied to the respective input values. We then take a sum of all these multiplied values which is called a weighted sum. Next, we add a bias value to the weighted sum to get final value for prediction by our neuron.

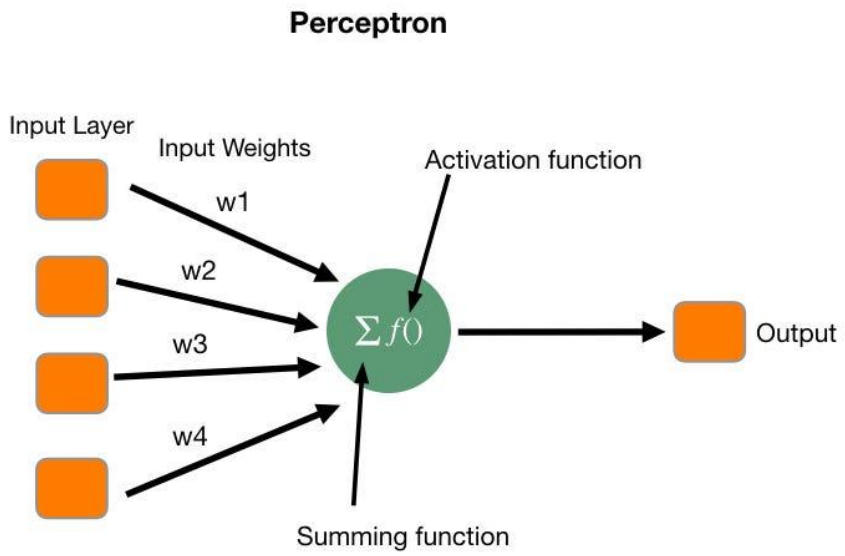


Each artificial neuron has the following main functions:

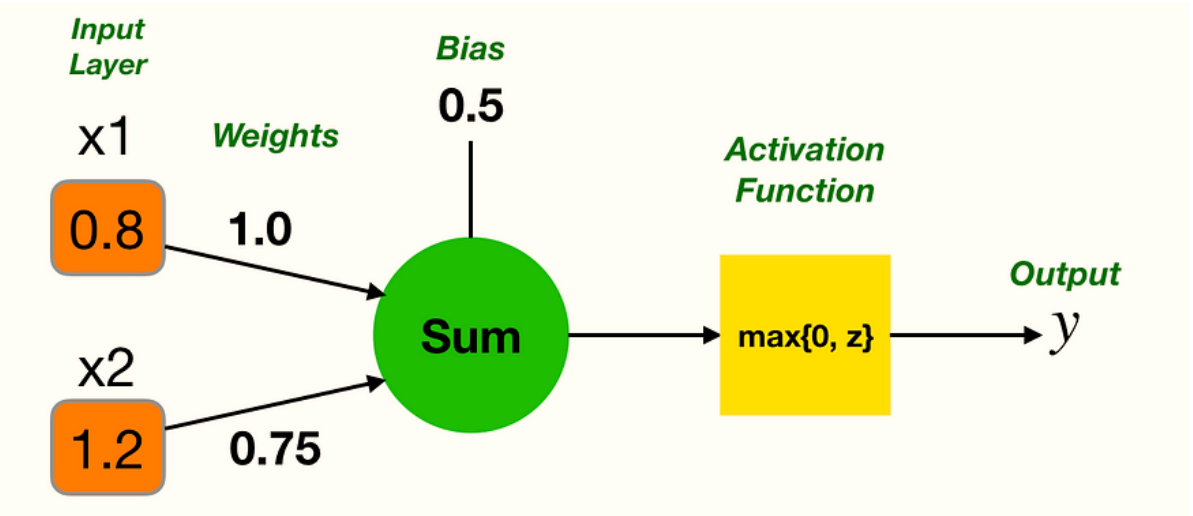
1. Takes inputs from the input layer
2. Weighs them separately and sums them up
3. Pass this sum through a nonlinear function to produce output

Basic Parts of a Artificial Neuron

3. **Activation Function**
   - Activation Function decides whether or not a neuron is fired. It decides which of the two output values should be generated by the neuron.
4. **Output Layer**
   - Output layer gives the final output of a neuron which can then be passed to other neurons in the network or taken as the final output value.



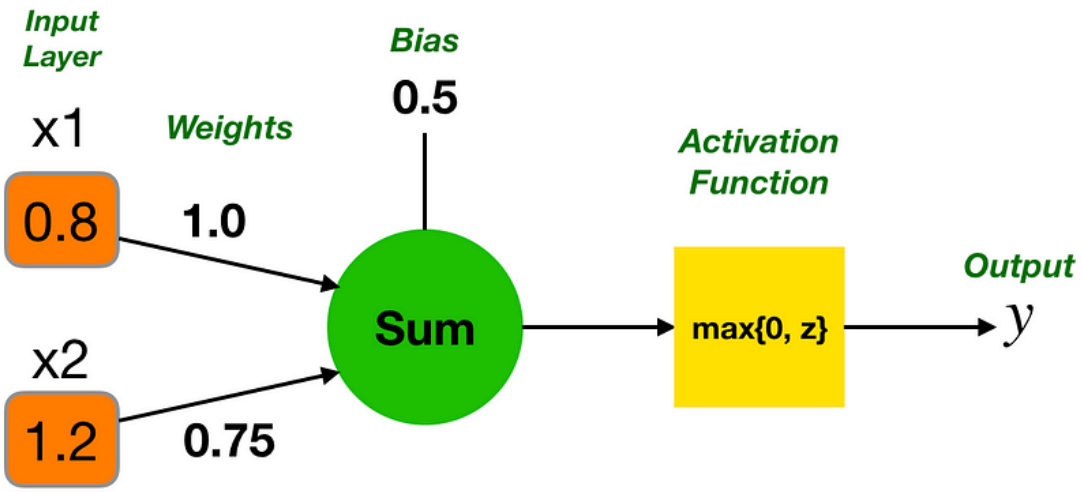
Practical Example



Consider a neuron with two inputs (x1,x2) as shown below

1. The values of the two inputs(x1,x2) are 0.8 and 1.2
2. We have a set of weights (1.0,0.75) corresponding to the two inputs
3. Then we have a bias with value 0.5 which needs to be added to the sum

Practical Example



- The input to activation function is then calculated using the formula:-

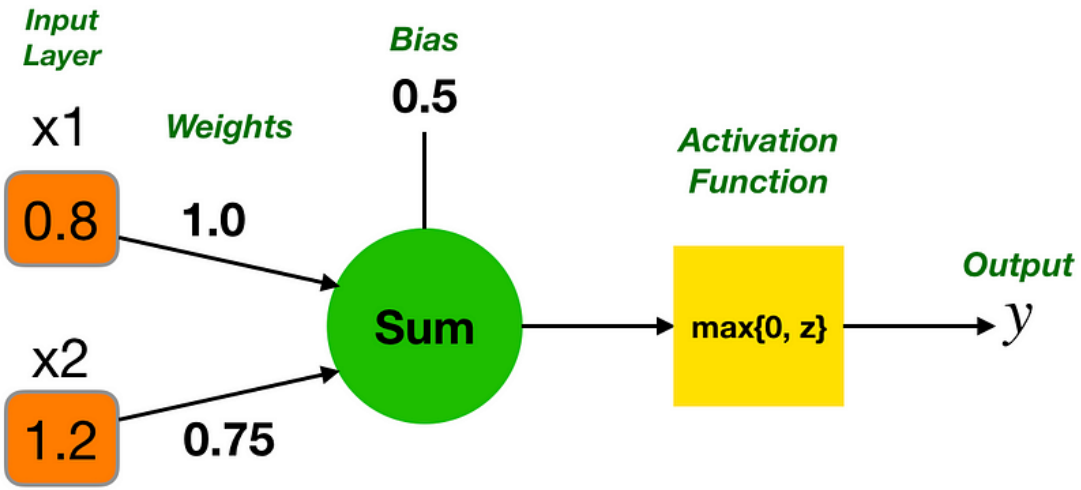
$$C = w1 * x1 + w2 * x2 + b$$

$$= (1 * 0.8) + (0.75 * 1.2) + 0.5$$

$$= 0.8 + 0.9 + 0.5$$

$$= 2.2$$

Practical Example



Note:
The purpose of **activation function** is to ensure that neuron response is bounded, i.e. actual response of neuron is damped or conditioned as a result of small or large stimuli and thus controllable

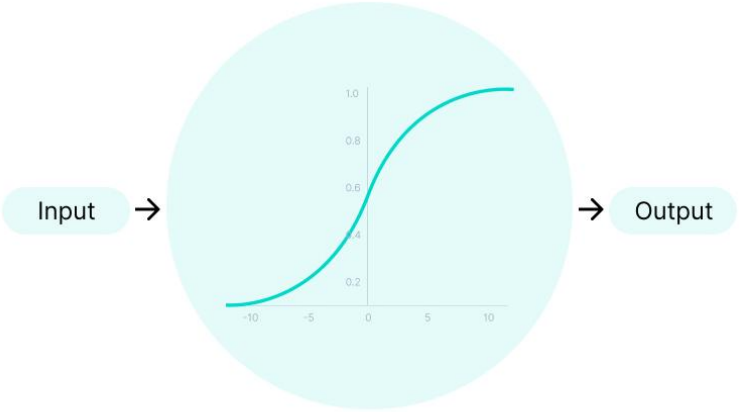
Now the combination(C) can be fed to the activation function. Let us first understand the logic of Rectified linear (ReLU) activation function which we are currently using in our example

$$activation = \begin{cases} 0 & if \quad combination < 0 \\ combination & if \quad combination \geq 0 \end{cases}$$

In our case, the combination value we got was 2.2 which is greater than 0 so the output value of our activation function will be 2.2.

Why do Neural Networks Need an Activation Function?

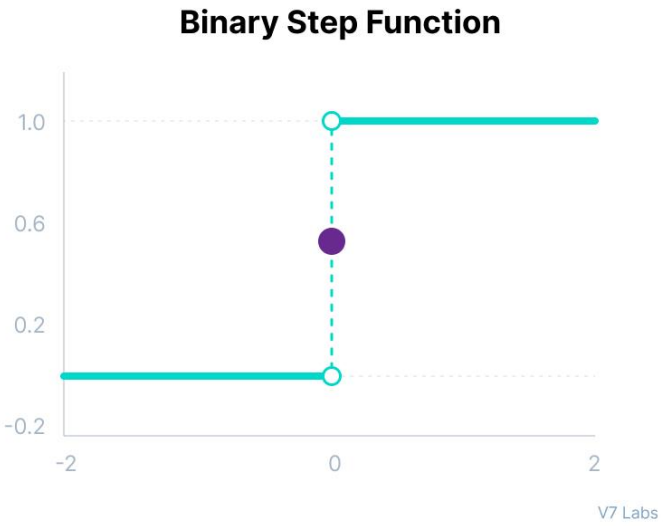
- Well, the purpose of an activation function is to add non-linearity to the neural network.
- Activation functions introduce an additional step at each layer during the forward propagation, but its computation is worth it. Here is why—
- Let's suppose we have a neural network working *without* the activation functions.
- In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases. It's because it doesn't matter how many hidden layers we attach in the neural network; all layers will behave in the same way because the composition of two linear functions is a linear function itself.
- Although the neural network becomes simpler, learning any complex task is impossible, and our model would be just a linear regression model



Activation Function

Types of Neural Networks Activation Functions

Binary Step Function
(Hard limit)



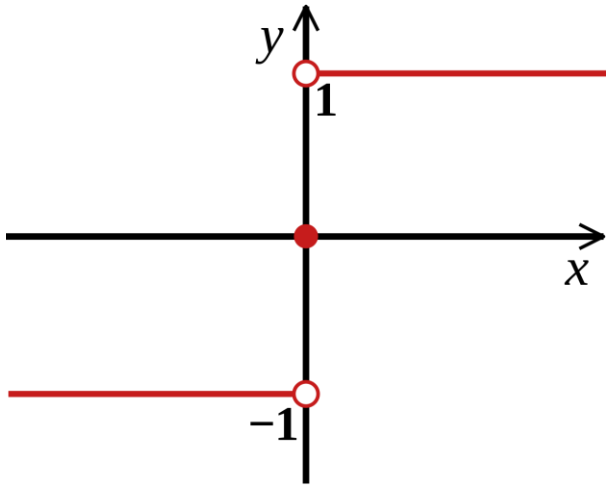
Binary step

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

- The limitations of binary step function:
  - It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.

Types of Neural Networks Activation Functions

Sign Function
(symmetric Hard limit)

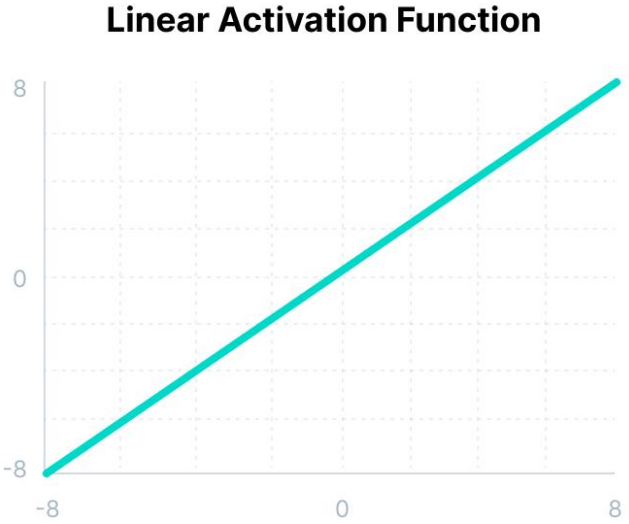


$$\operatorname{sgn} x := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

- It can provide only 3 multi-value output.

Types of Neural Networks Activation Functions

Linear Activation Function



Linear

$$f(x) = x$$

- However, a linear activation function has two major problems :
  - 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. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially**, a linear activation function turns the neural network into just one layer**.

Types of Neural Networks Activation Functions

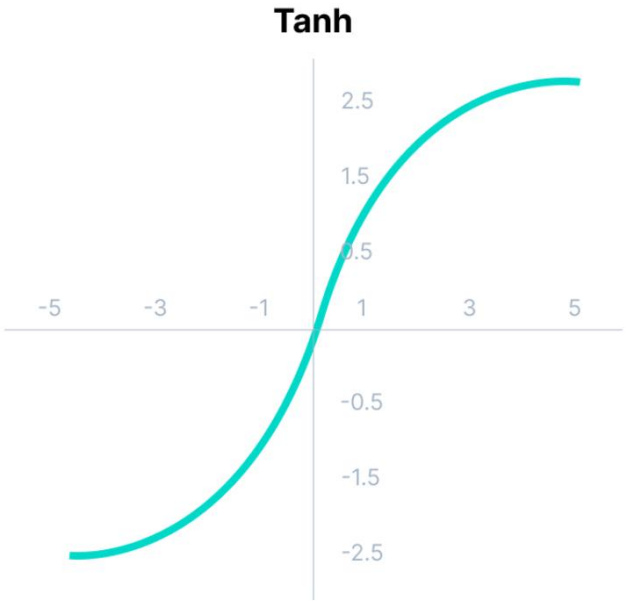
Sigmoid / Logistic Activation Function

Sigmoid / Logistic

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

Types of Neural Networks Activation Functions

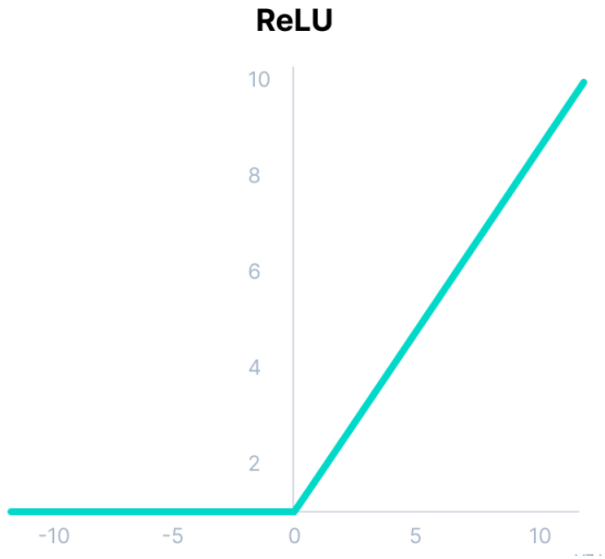
Tanh Function (Hyperbolic Tangent)



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

Types of Neural Networks Activation Functions

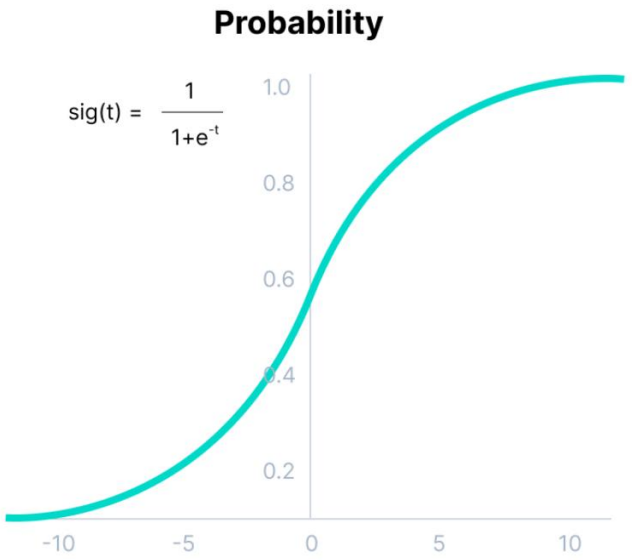
ReLU Function (Positive linear)



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

Types of Neural Networks Activation Functions

Softmax Function

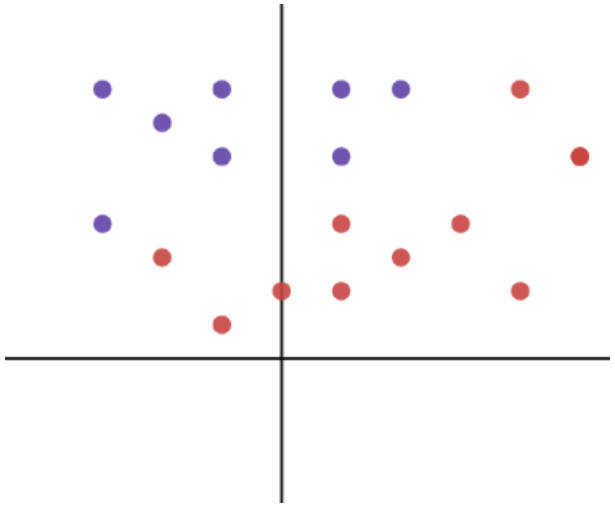


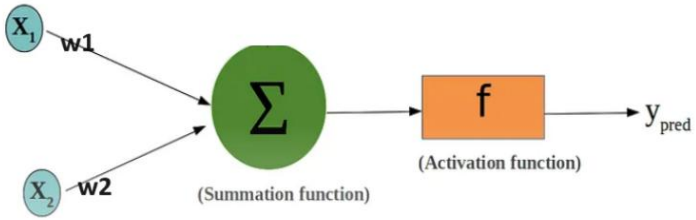
$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

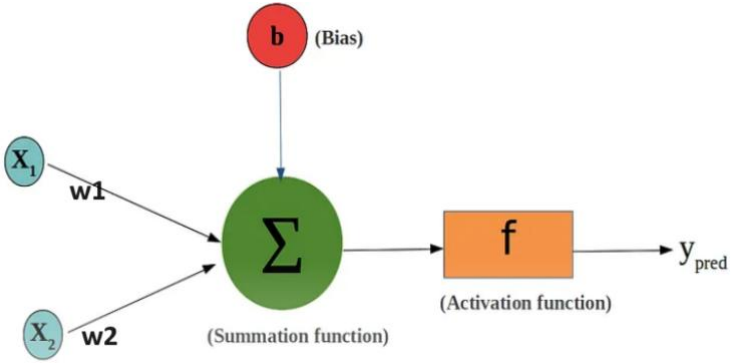
- When we need to classifying data into multiple categories.
- The Softmax function takes the output values from the last layer and converts them into probabilities.
- Sum of this probabilities is equal to 1
- If the model outputs [1.8, 0.9, 0.68], applying Softmax gives [0.58, 0.23, 0.19].
- The highest is 0.58 (index 0), so the model classifies the sample as the first category.

The bias in neural networks

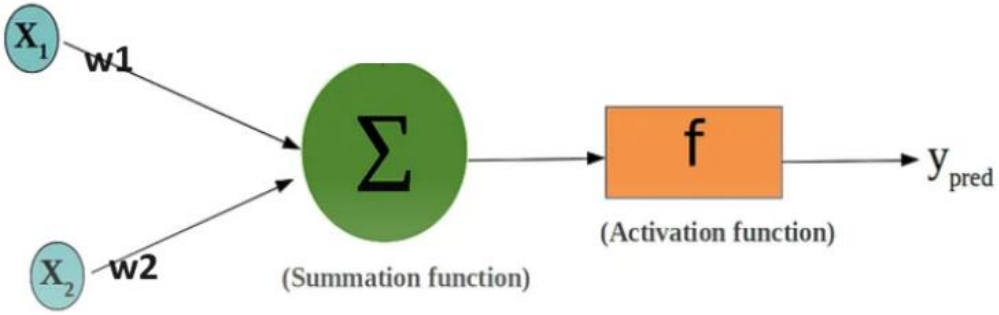
- **Bias** in neural networks is a value that shifts the line of the perceptron's equation to make the perceptron more flexible under real-world scenarios.

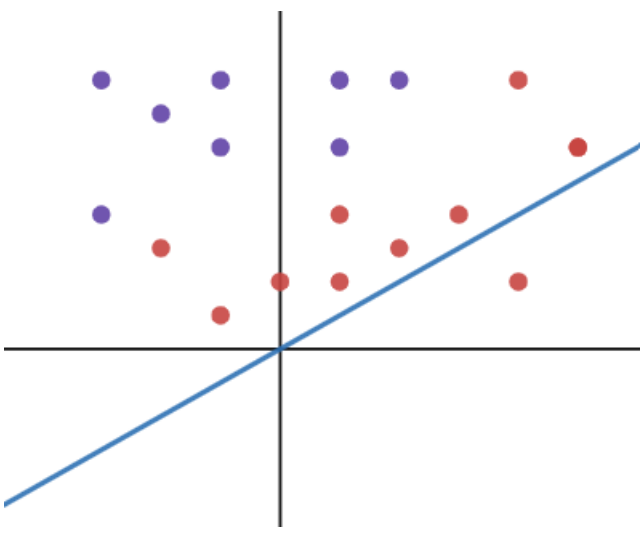




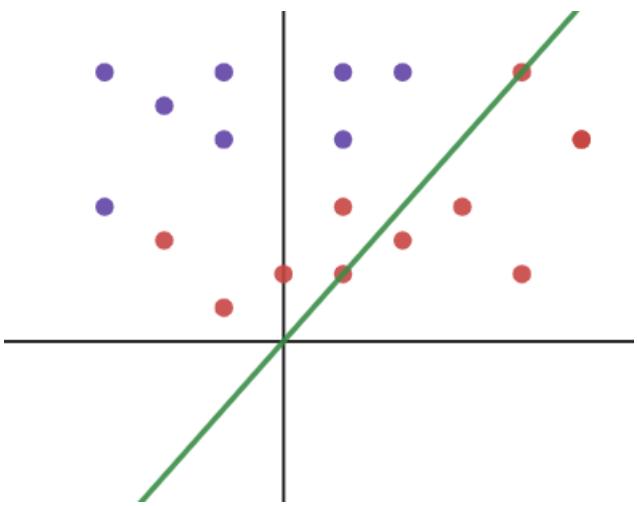


Bias of a Neuron

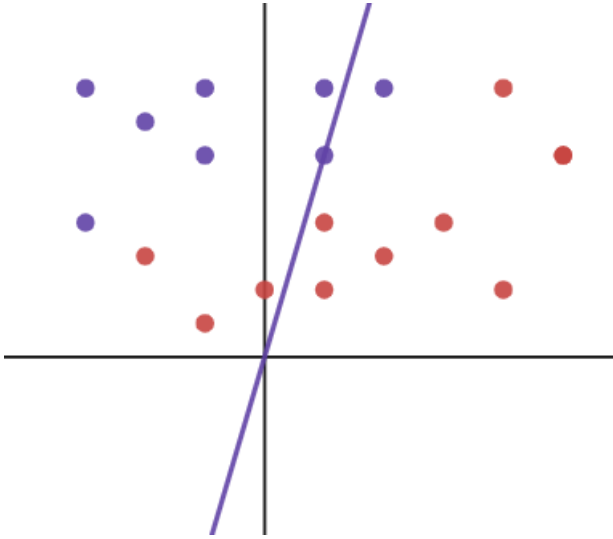




$X_1 - X_2 = 0$

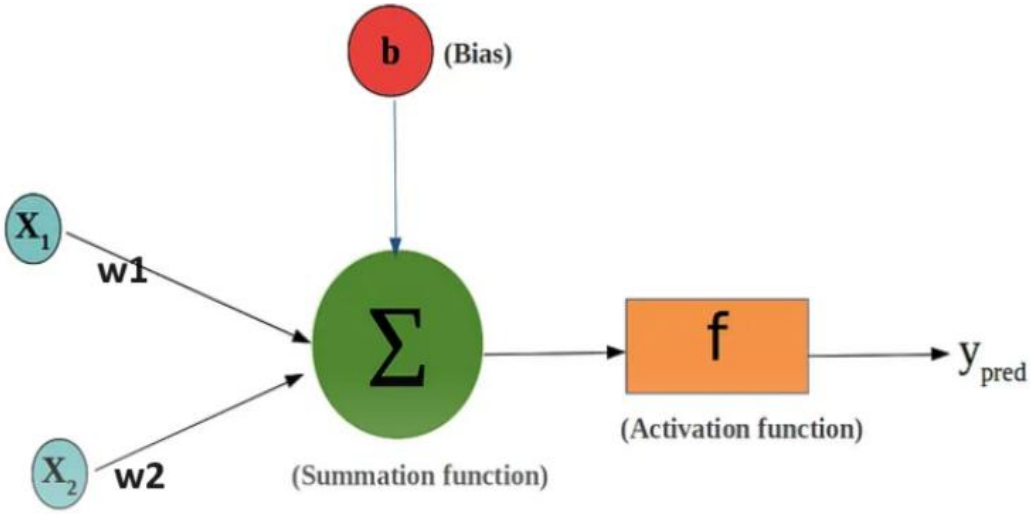


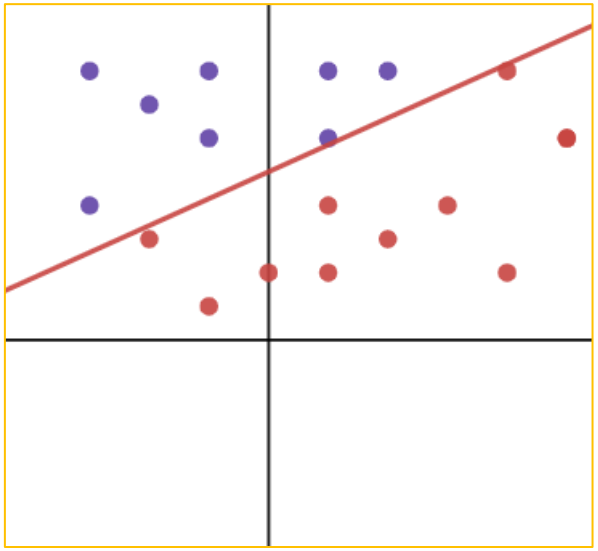
$2X_1 - X_2 = 0$



$3X_1 - 0.5X_2 = 0$

Bias of a Neuron





$0.8X_1 - X_2 = -5$

Why we need bias?

- **Prevents Strict Linearity**

If a neuron had only weighted inputs, it would be forced to always pass through the origin (in the case of linear activation). Bias allows flexibility by shifting the decision boundary.

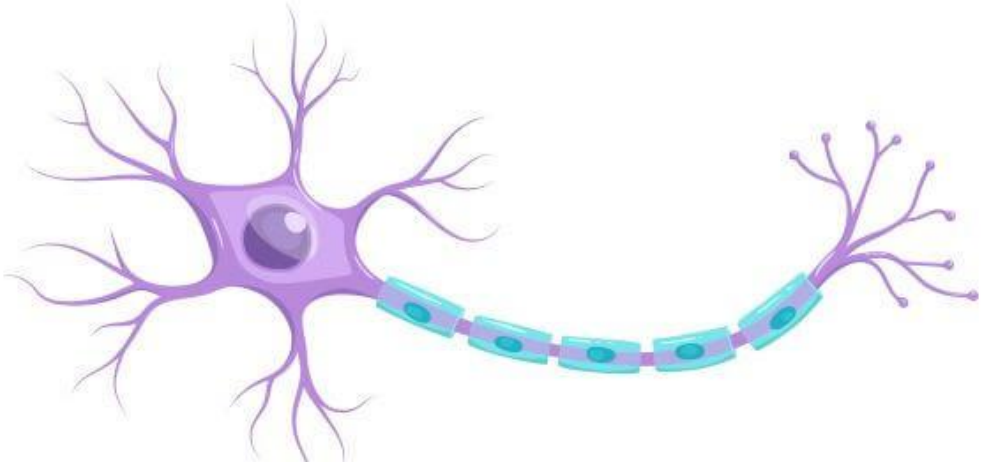
- **Adjusting Activation Threshold**

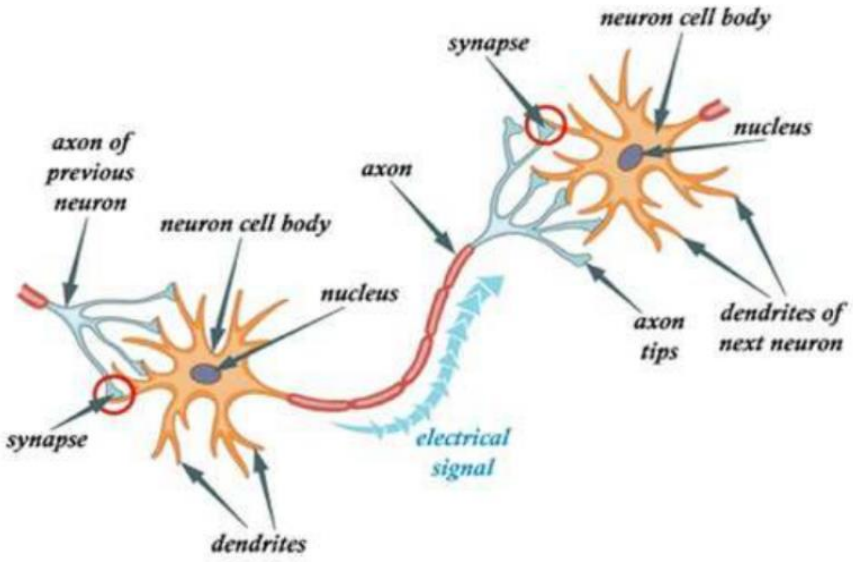
Without bias, the activation of a neuron depends only on the weighted sum of inputs. Bias allows shifting the activation function to better fit the data.

- **Enabling Learning of Complex Patterns**

Without a bias term, the network may struggle to learn patterns effectively, especially when inputs are zero.

Biological Neurons





Biological vs. Artificial Neurons

Neural networks are inspired by the human brain but differ significantly:

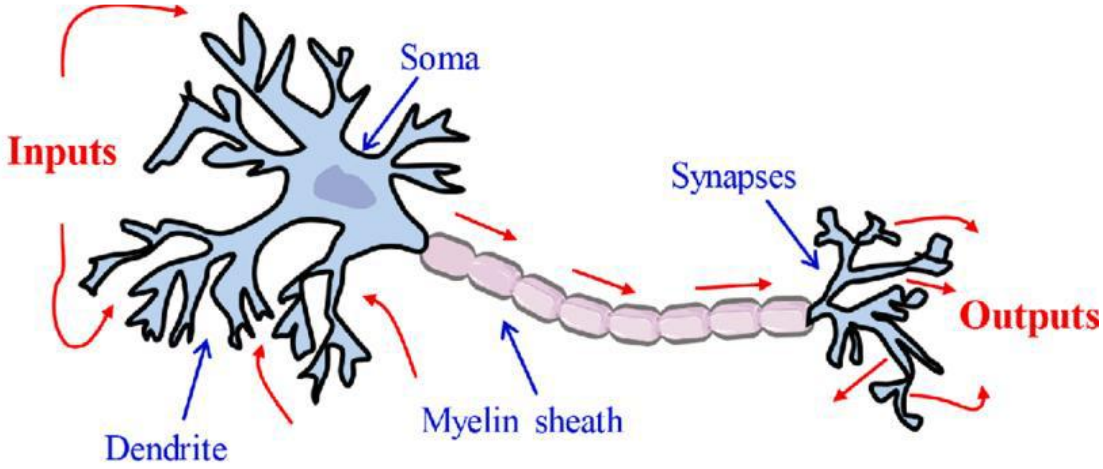
- **Biological Neurons:** Composed of dendrites (inputs), a soma (processing unit), and axons (outputs). They communicate via electrical and chemical signals.
- **Artificial Neurons:** Modeled as mathematical functions that process weighted inputs and apply an activation function to produce an output.

Biological vs. Artificial Neurons

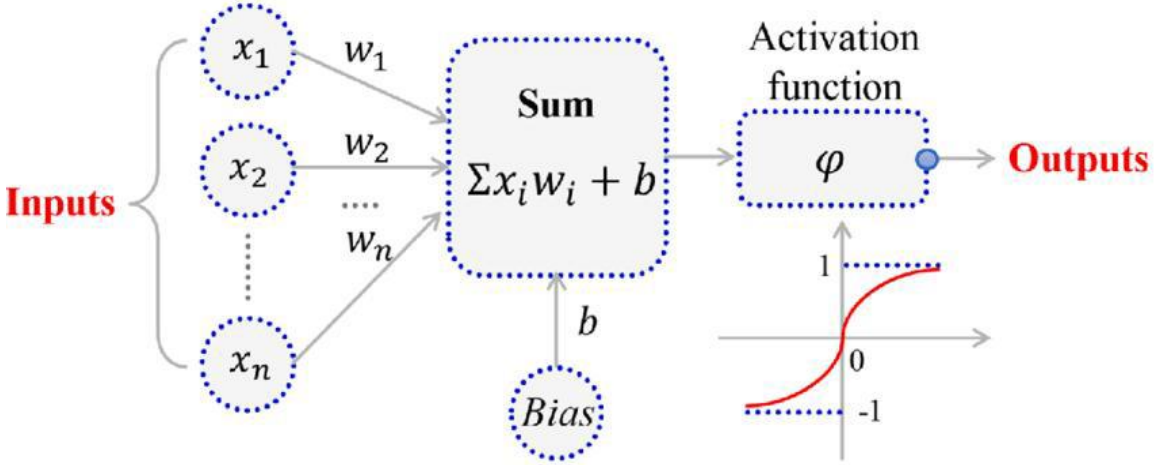
Key differences:

- The human brain has approximately 86 billion neurons, whereas artificial neural networks typically use thousands to millions of neurons.
- Biological neurons communicate in parallel with complex synaptic structures, while artificial neurons use simpler mathematical operations.
- Learning in the brain involves synaptic plasticity, whereas artificial networks use gradient-based optimization.

Biological vs. Artificial Neurons



(a) Biological neuron



(b) Artificial neuron

ANN and matrices

- The equation of the weighted sum can be written with matrices as

$$v = w * x + b$$

- where w and x are defined as:

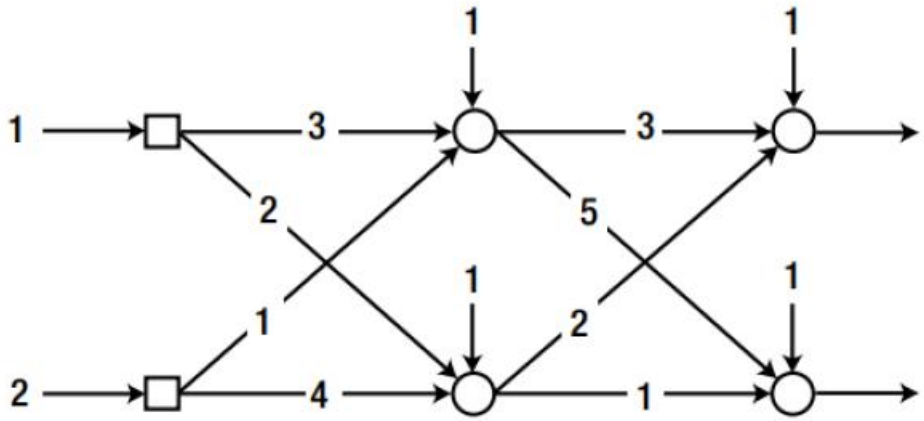
$$w = [w_1 \;\; w_2 \;\; w_3] \;\; x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

- Finally, the node enters the weighted sum into the activation function and yields its output. The activation function determines the behavior of the node.

$$y = \varphi(v)$$

- $\varphi(v)$ of this equation is the activation function. Many types of activation functions are available in the neural network. We will elaborate on them later.

Example



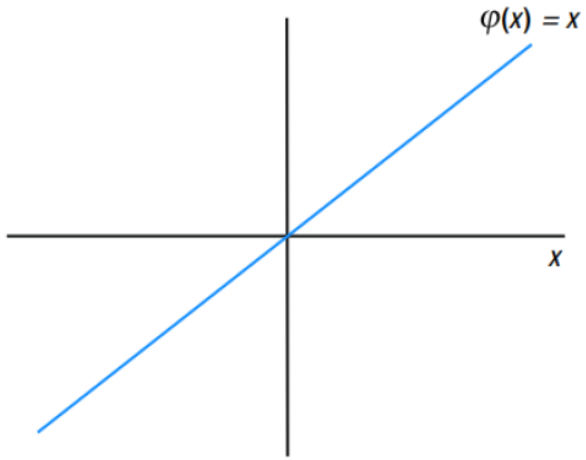
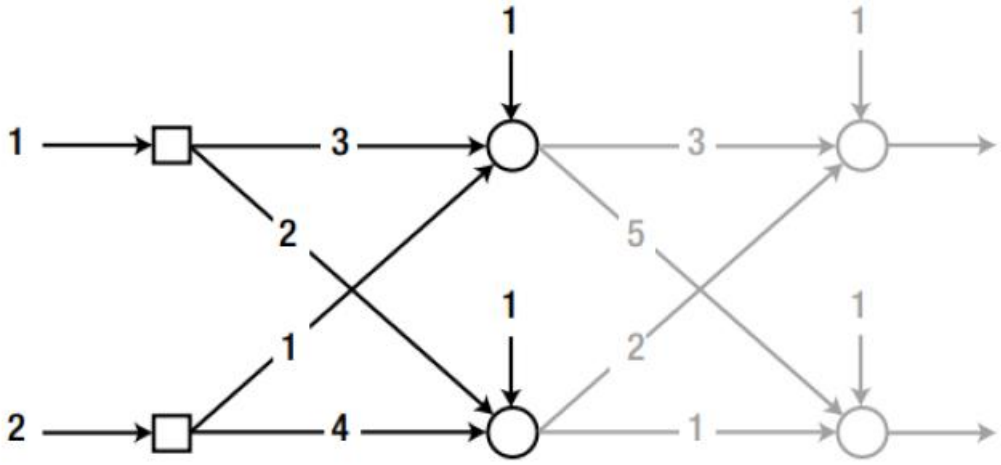


Figure 2-7 The activation function of each node is a linear function



The first node of the hidden layer calculates the output as:

$$\text{Weighted sum: } v = (3\times1) + (1\times2) + 1 = 6$$

$$\text{Output: } y = \varphi(v) = v = 6$$

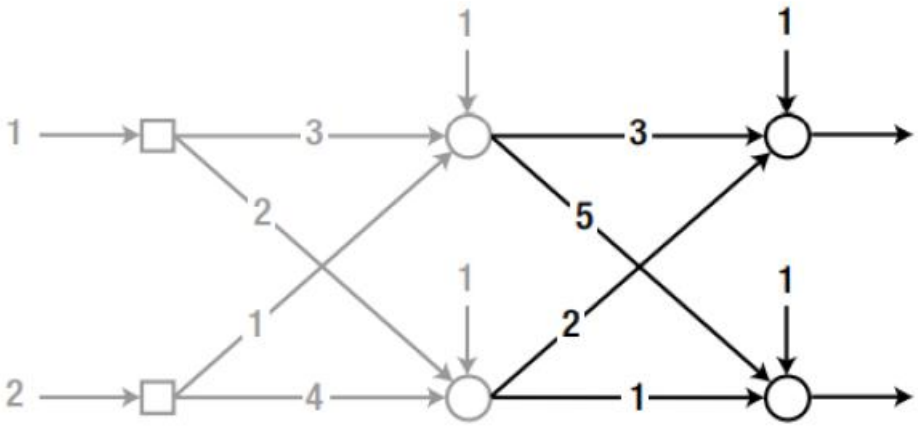
In a similar manner, the second node of the hidden layer calculates the output as:

$$\text{Weighted sum: } v = (2\times1) + (4\times2) + 1 = 11$$

$$\text{Output: } y = \varphi(v) = v = 11$$

The weighted sum calculations can be combined in a matrix equation as follows:

$$v = \begin{bmatrix} 3\times1+1\times2+1 \\ 2\times1+4\times2+1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 11 \end{bmatrix}$$



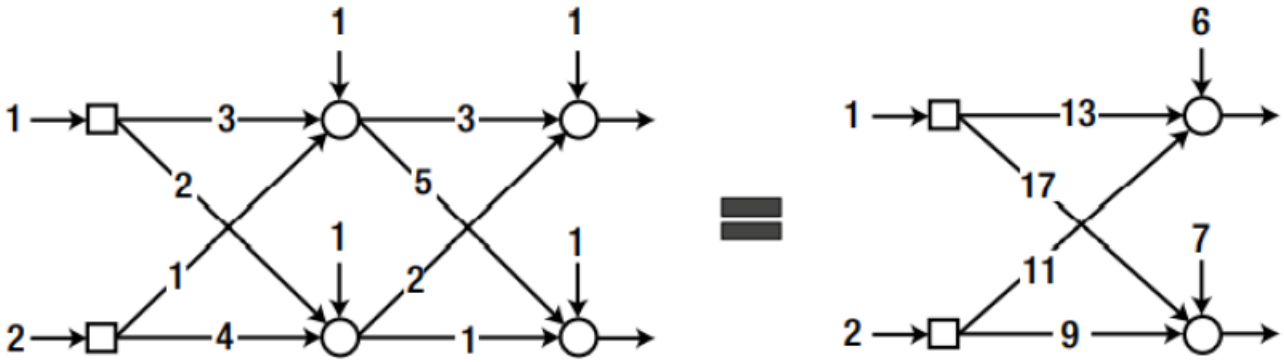
Weighted sum: $v = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 11 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 41 \\ 42 \end{bmatrix}$

Output: $y = \varphi(v) = v = \begin{bmatrix} 41 \\ 42 \end{bmatrix}$

We used a **linear equation for the activation of the hidden nodes**, just for convenience. This is **not practically correct**. The use of a linear function for the nodes negates the effect of adding a layer. In this case, the model is mathematically identical to a single-layer neural network, which does not have hidden layers. Let's see what really happens. Substituting the equation of weighted sum of the hidden layer into the equation of weighted sum of the output layer yields the following equation:

$$
\begin{aligned}
v &= \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 11 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \left(\begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 13 & 11 \\ 17 & 9 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 6 \\ 7 \end{bmatrix}
\end{aligned}
$$

This matrix equation indicates that this example neural network is equivalent to a single layer neural network as shown in the Figure

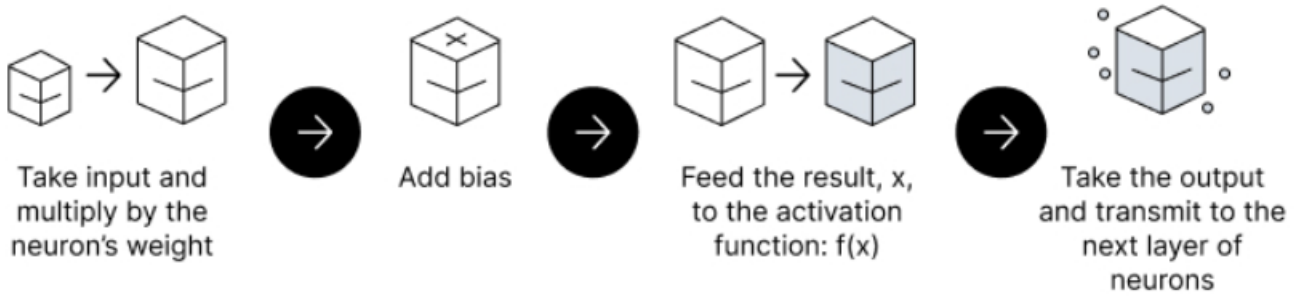


Feedforward vs. Backpropagation

When learning about neural networks, you will come across two essential terms describing the movement of information—feedforward and backpropagation.

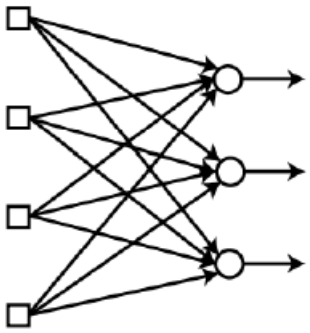
💡 **Feedforward Propagation** - the flow of information occurs in the forward direction. The input is used to calculate some intermediate function in the hidden layer, which is then used to calculate an output.

In the feedforward propagation, the Activation Function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.

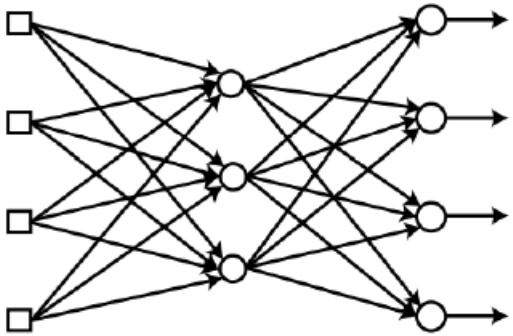


Layers in Neural Network

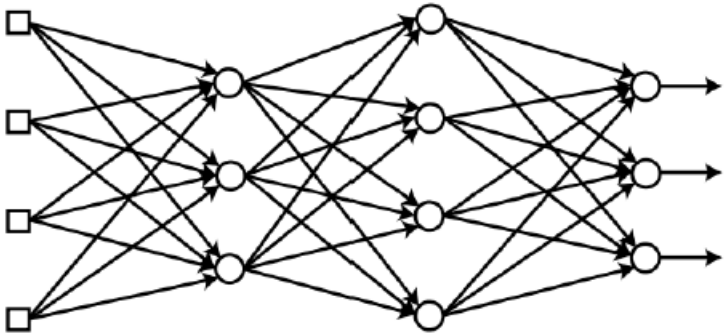
Single layer NN		Input layer - output layer
Multilayer NN	Shallow NN	Input layer - Hidden layer - output layer
	Deep NN	Input layer - Hidden layers - output layer



Single-layer Neural Network

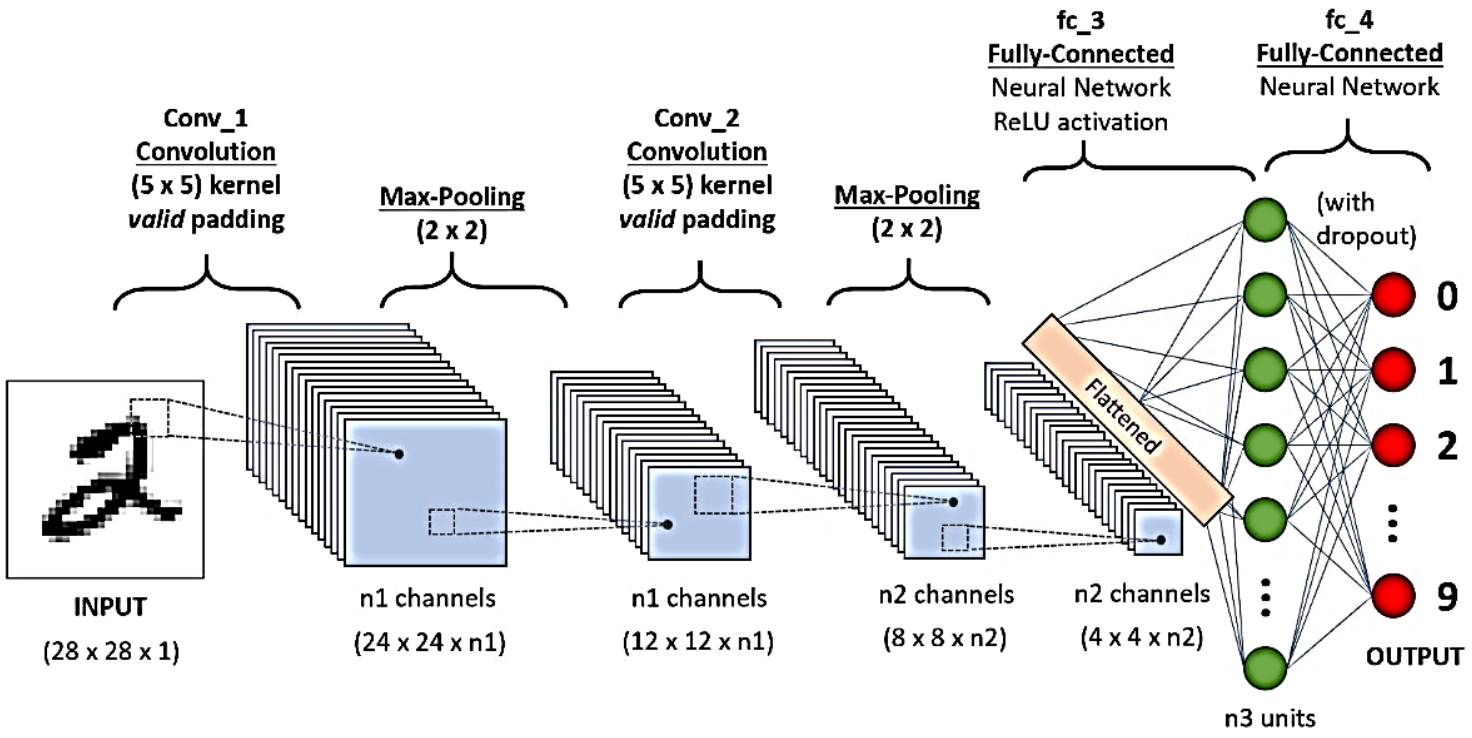


(Shallow) Multi-layer Neural Network



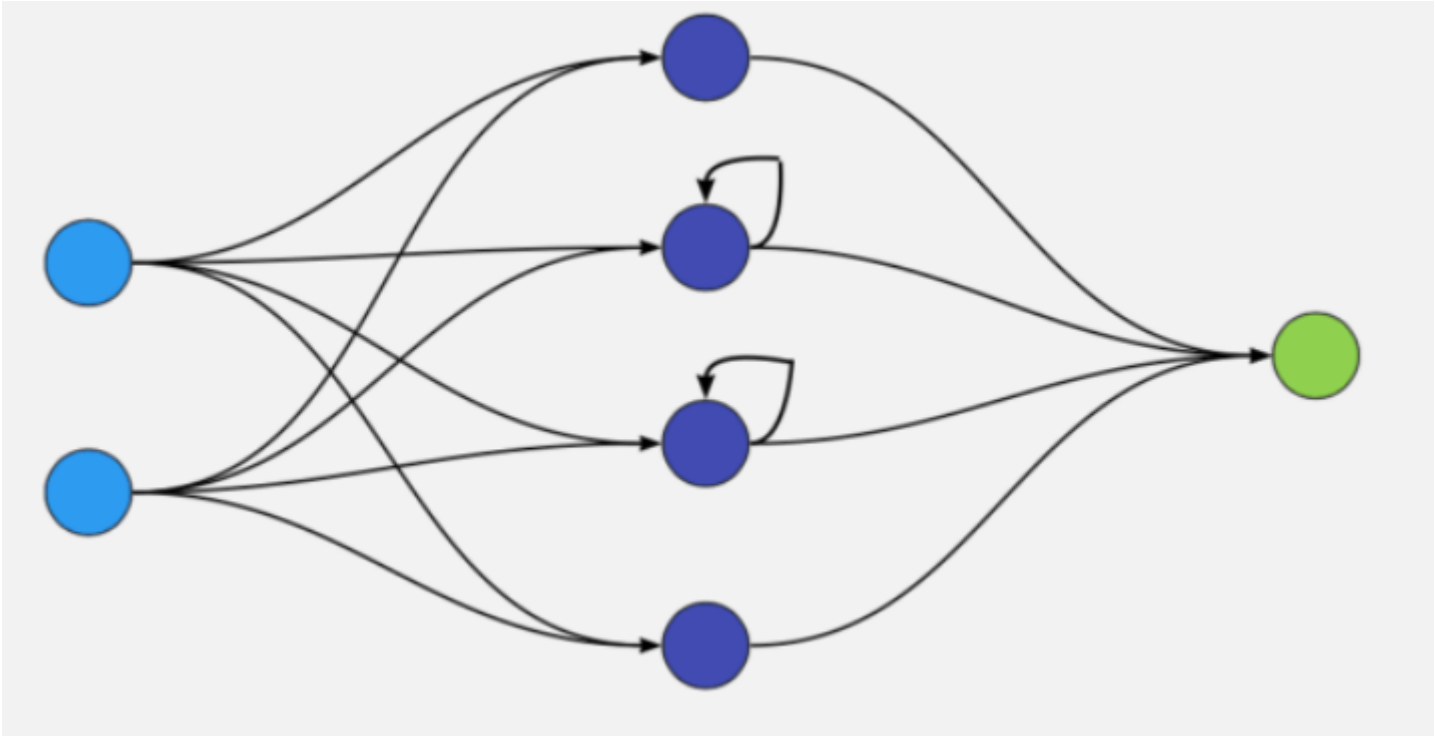
Deep Neural Network

convolutional neural networks (CNNs)



You can use CNN architecture when you process images and videos, as it can handle varying inputs in dimension and size.

recurrent neural networks (RNNs).



RNNs excel at natural language functions like language modeling, speech recognition, and sentiment analysis

	Deep learning systems	Simple neural networks
Architecture	Consists of several hidden layers arranged for convolution or recurrence.	Neural networks consist of an input, hidden, and output layer. They mimic the human brain in structure.
Complexity	Depending on its function, a deep learning network is highly complicated and has structures like long short-term memory (LSTM) and autoencoders.	Neutral networks are less complicated, as they have only a few layers.
Performance	A deep learning algorithm can solve complex issues across large data volumes.	Neural networks perform well when solving simple problems.
Training	It costs a lot of money and resources to train a deep learning algorithm.	The simplicity of a neural network means it costs less to train.