Artificial Intelligence

Introduction to Neural Networks and Deep Learning Frameworks

Agenda

01 Topology of Neural Networks

02 Perceptrons

03 Activation Functions and Their Types

04 Perceptron Training Algorithm

05 Deep Learning Frameworks

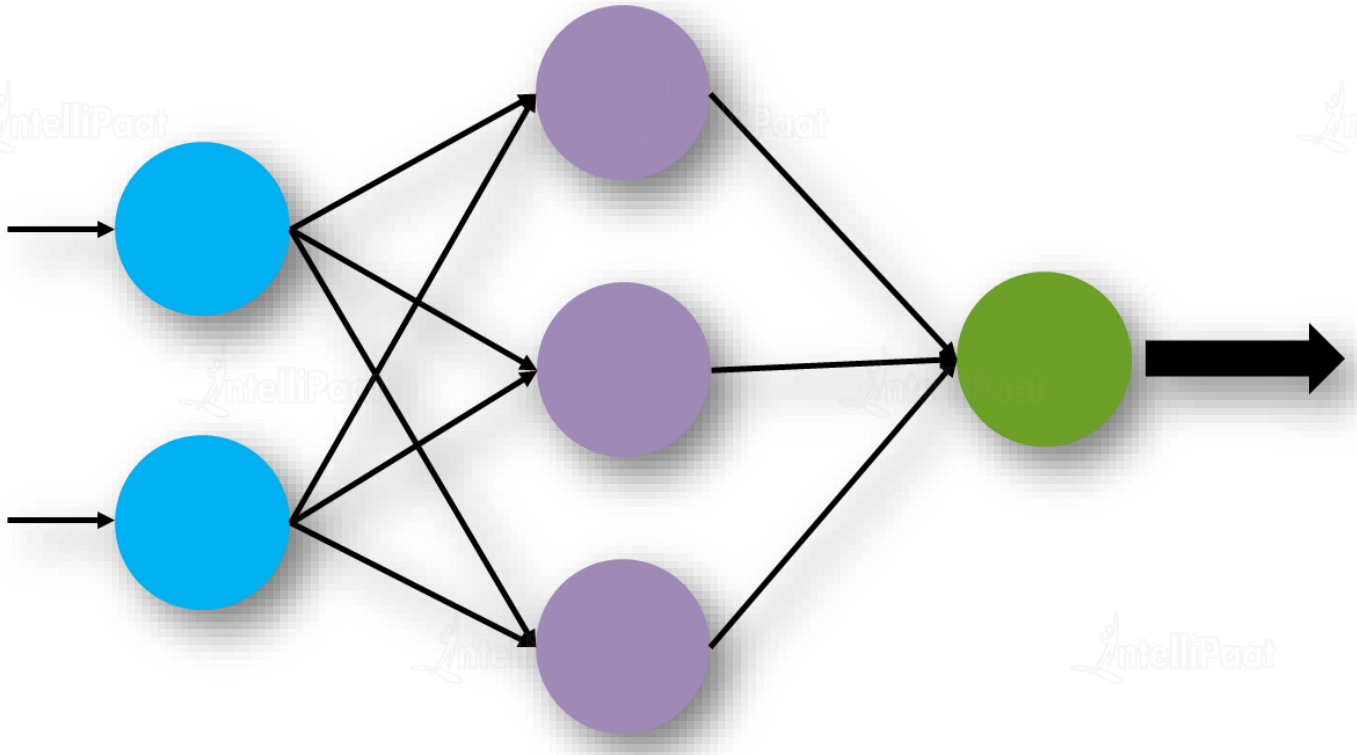
06 What Are Tensors?

07 Computational Graph

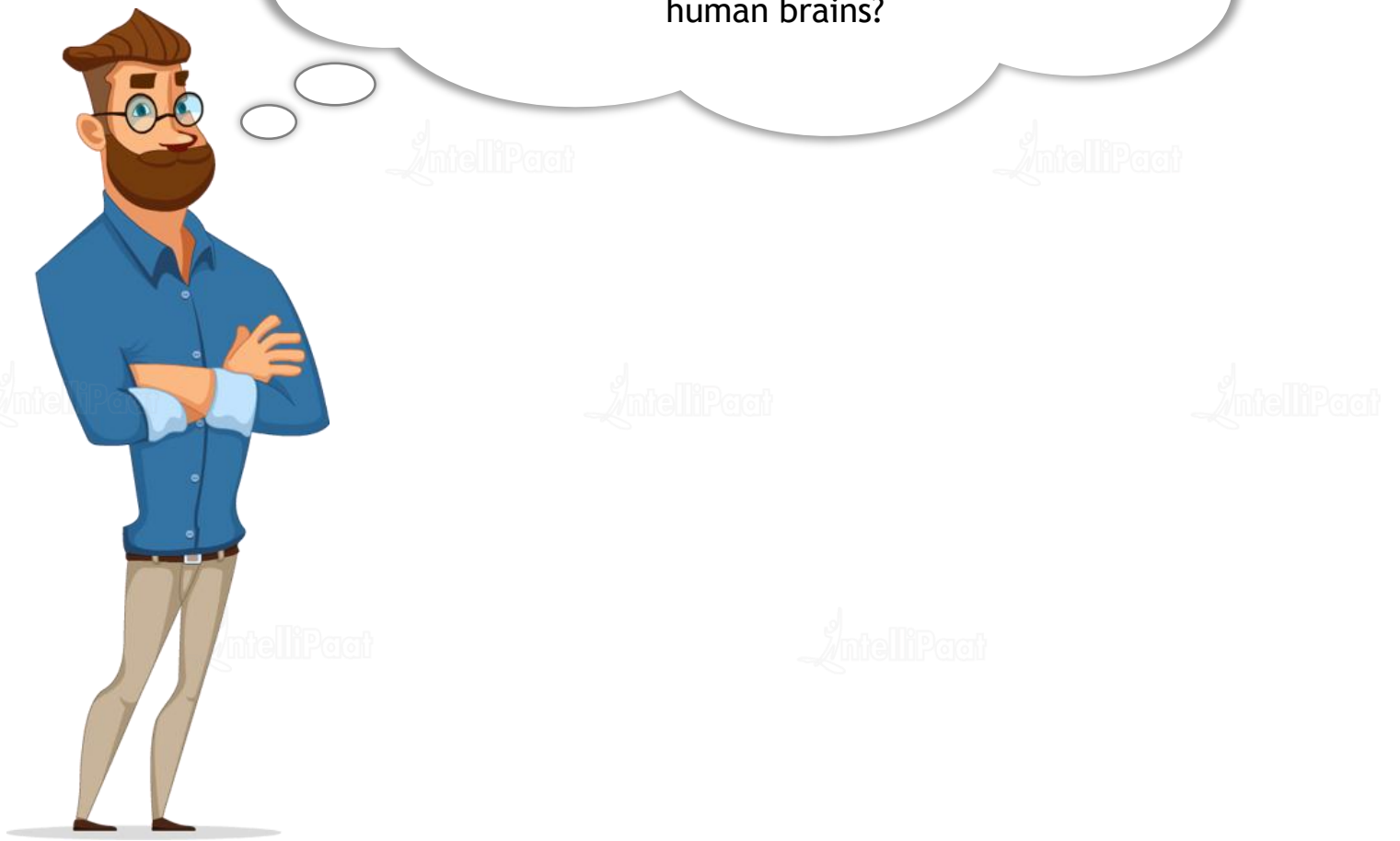
08 Program Elements in TensorFlow

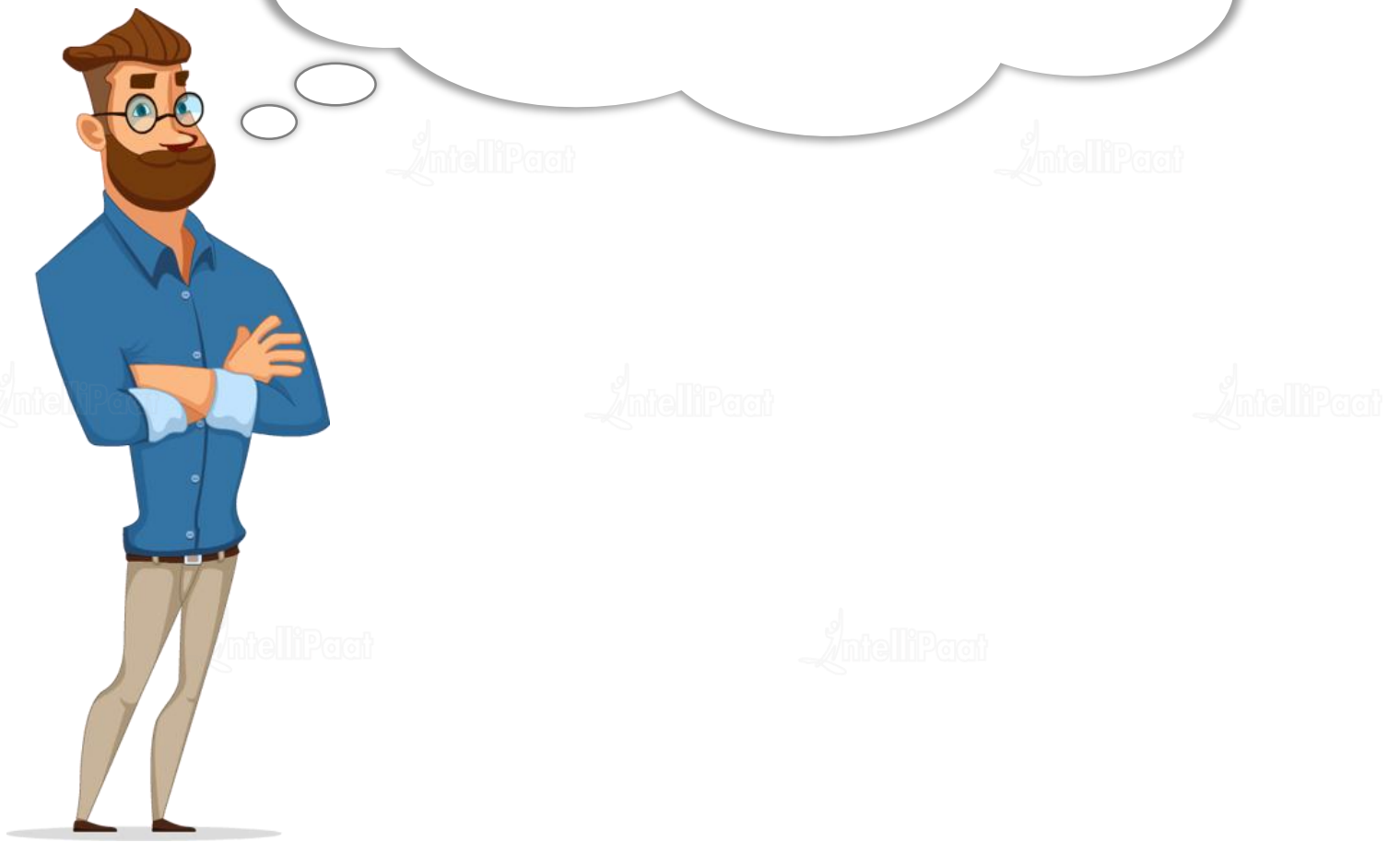
Topology of a Neural Network

Typically, artificial neural networks have a layered structure. The Input Layer picks up the input signals and passes them on to the next layer, also known as the 'Hidden' Layer (there may be more than one Hidden Layer in a neural network). Last comes the Output Layer that delivers the result



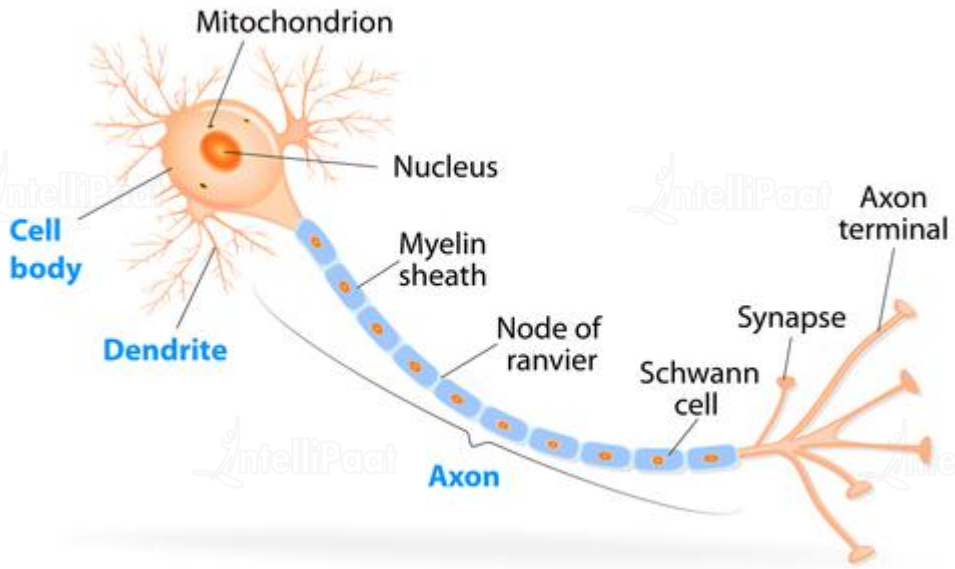
Input layer ***Hidden layer*** ***Output layer***





Neurons: How Do They Work?

A neural network is a computer simulation of the way biological neurons work within a human brain



Dendrites: These branch-like structures extending away from the cell body receive messages from other neurons and allow them travel to the cell body

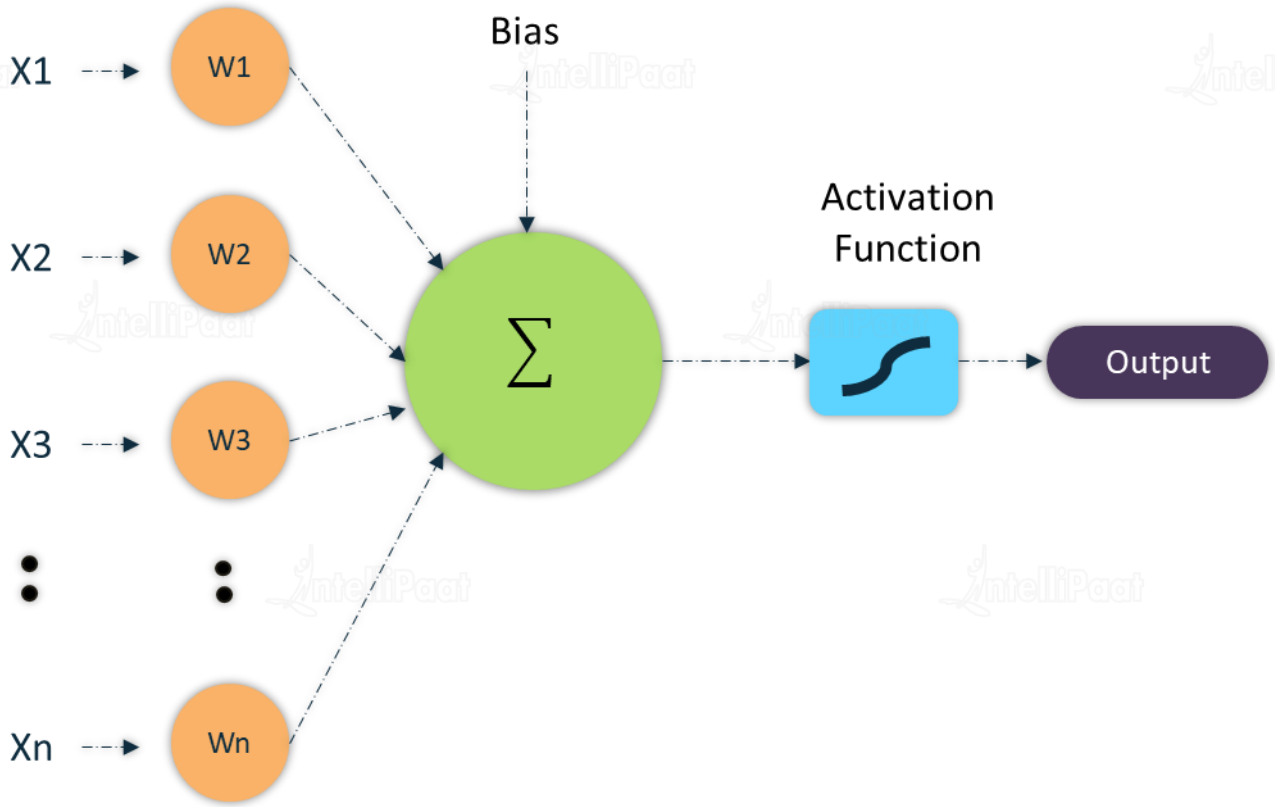
Cell Body: It contains a nucleus, smooth and rough endoplasmic reticulum, Golgi apparatus, mitochondria, and other cellular components

Axon: An axon carries an electrical impulse from the cell body to another neuron

Now, let us understand about artificial neurons in detail!

Artificial Neurons

- The most fundamental unit of a deep neural network is called as an artificial neuron
- It takes an input, processes it, passes it through an activation function, and returns the output
- Such type of artificial neurons are called as *perceptrons*
- A perceptron is a linear model used for binary classification



Schematic Representation of a Neuron in a Neural Network

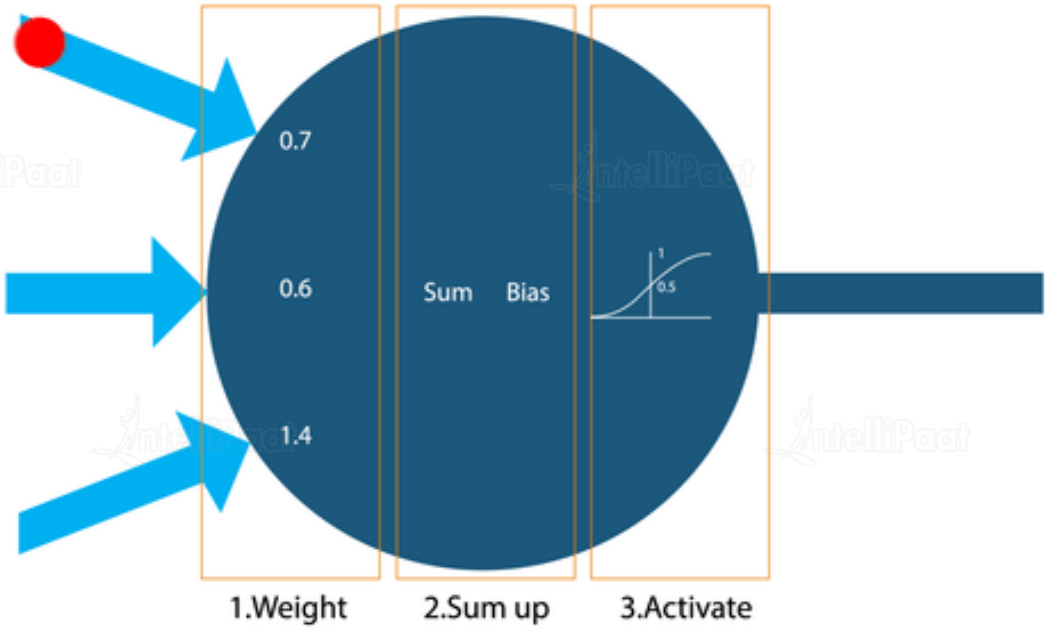
Perceptron: How Does It Work?

- The three arrows correspond to the three inputs coming into the network
- Values [0.7, 0.6, and 1.4] are weights assigned to the corresponding input
- Inputs get multiplied with their respective weights and their sum is taken
- Consider the three inputs as x1, x2, and x3
- Let the three weights be w1, w2, and w3

```
Sum = x1w1 + x2w2+x3w3
Sum=x1(0.7) + x2(0.6) + x3(1.4)
```

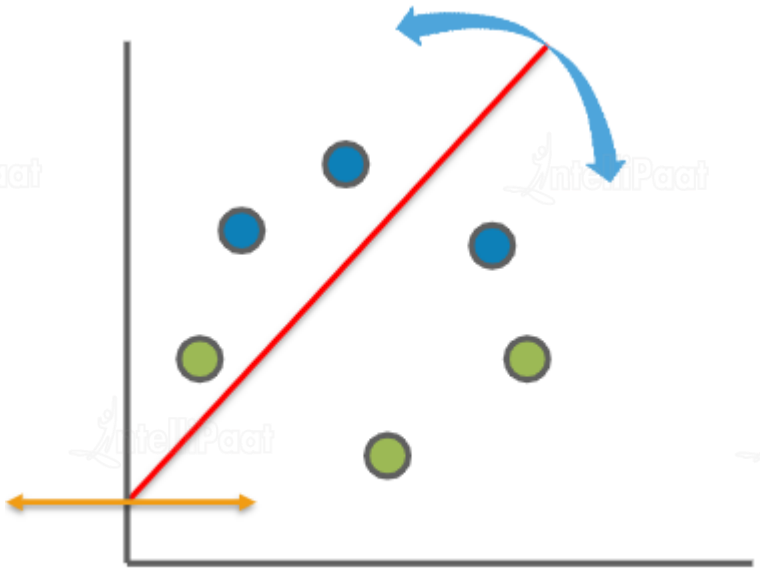
- An offset is added to this sum. This offset is called ***Bias***
- It is just a constant number, say 1, which is added for scaling purposes

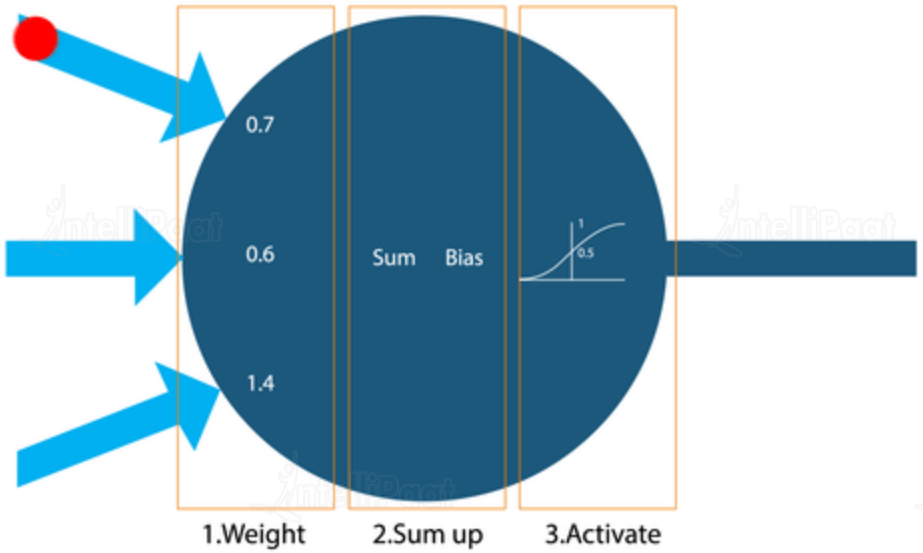
```
New_Sum = x1(0.7) + x2(0.6) + x3(1.4) + bias
```



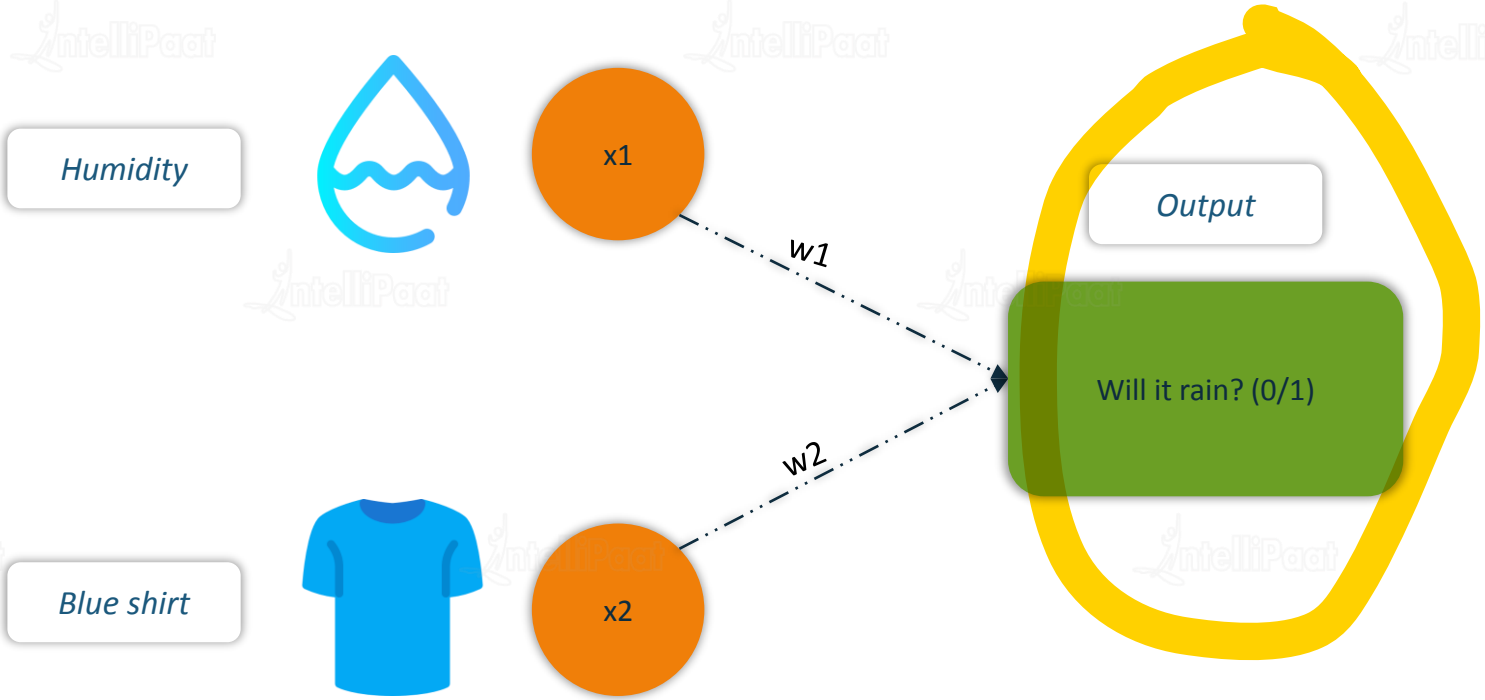
Why Do We Need Weights?

- Statistically, weights determine the relative importance of input
- Mathematically, they are just the slope of the line



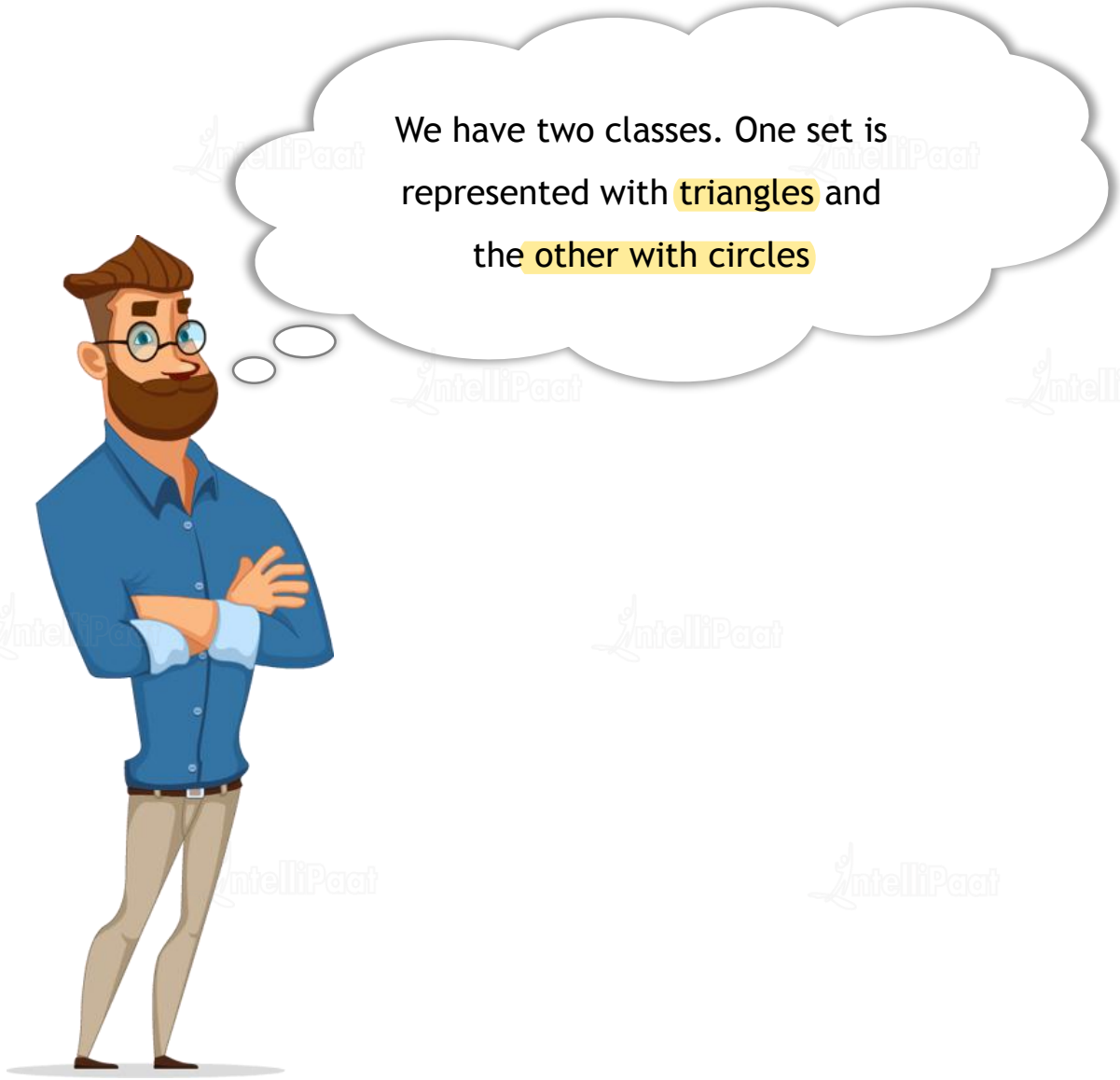


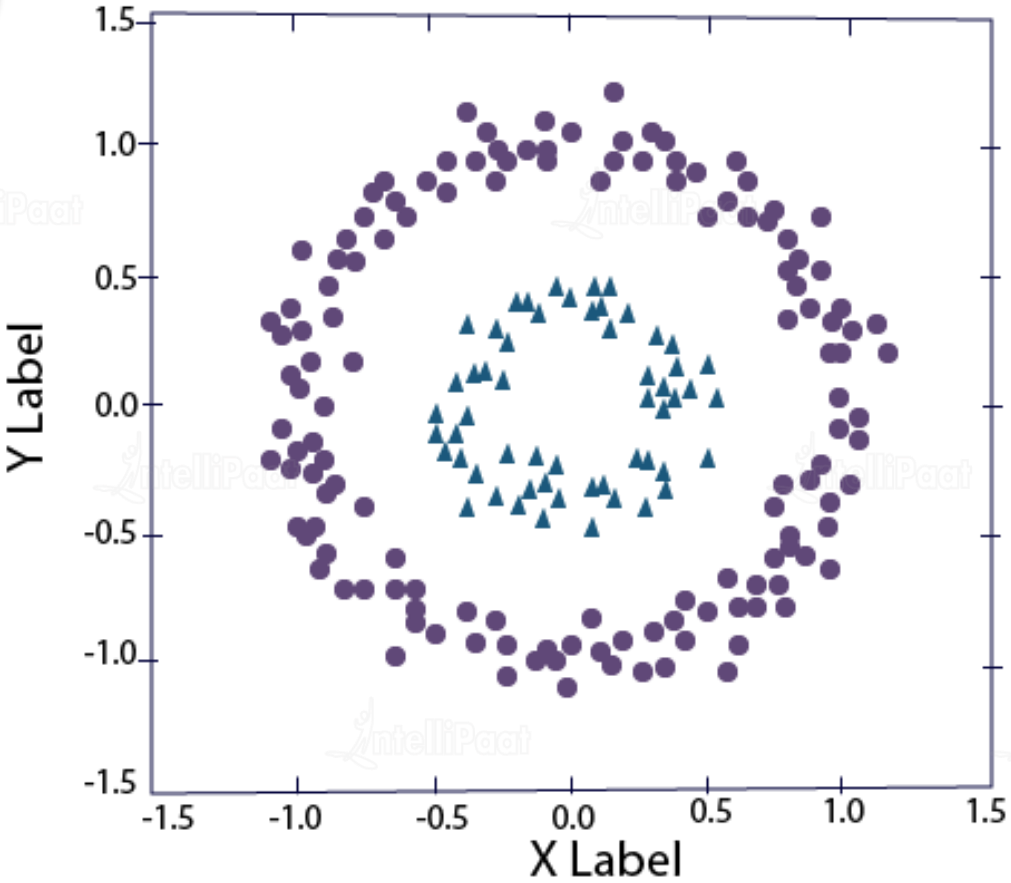
Why Do We Need Weights?



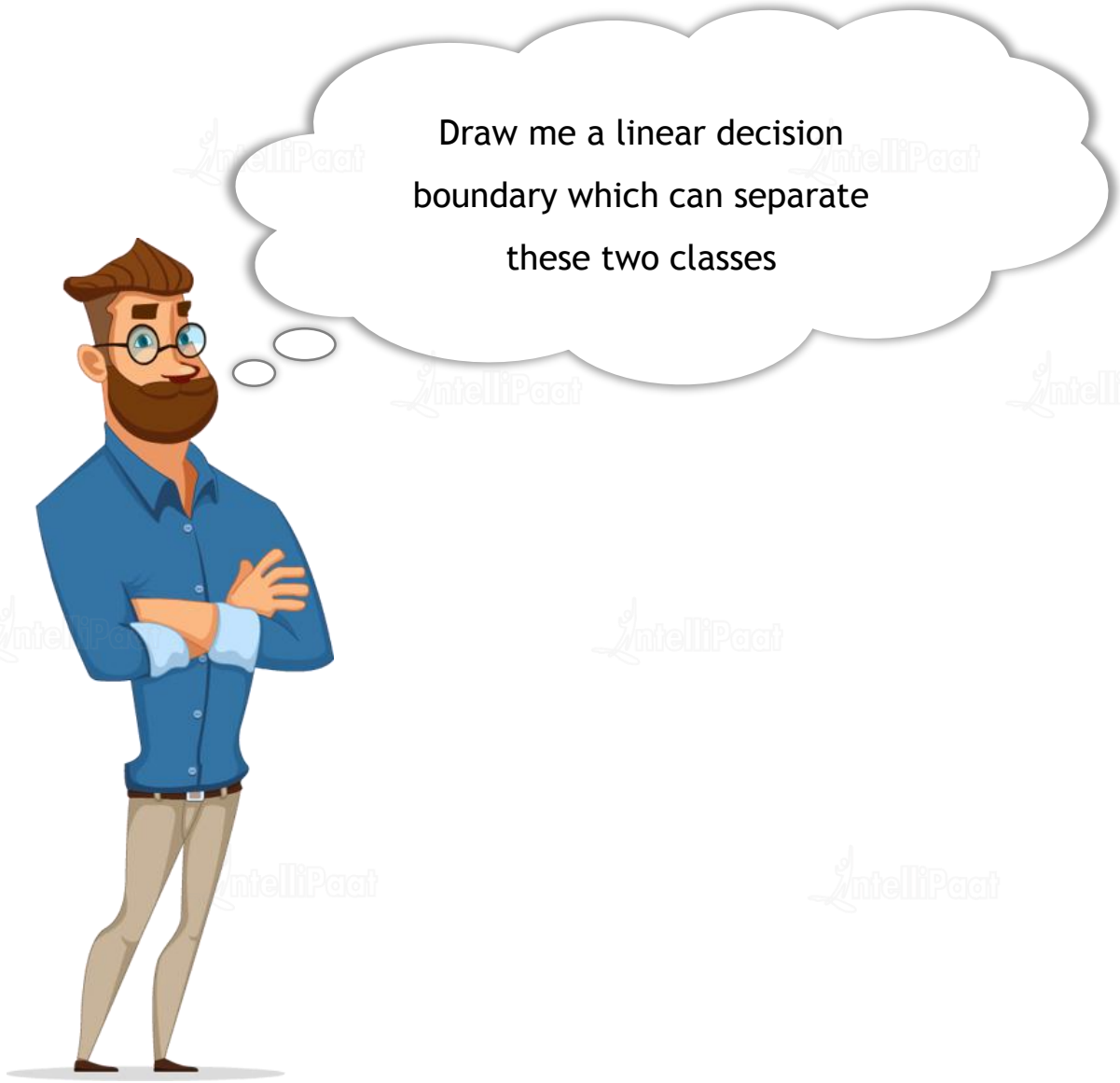
w2 is assigned a lower value because significance of the input 'blue shirt' is less than 'humidity'

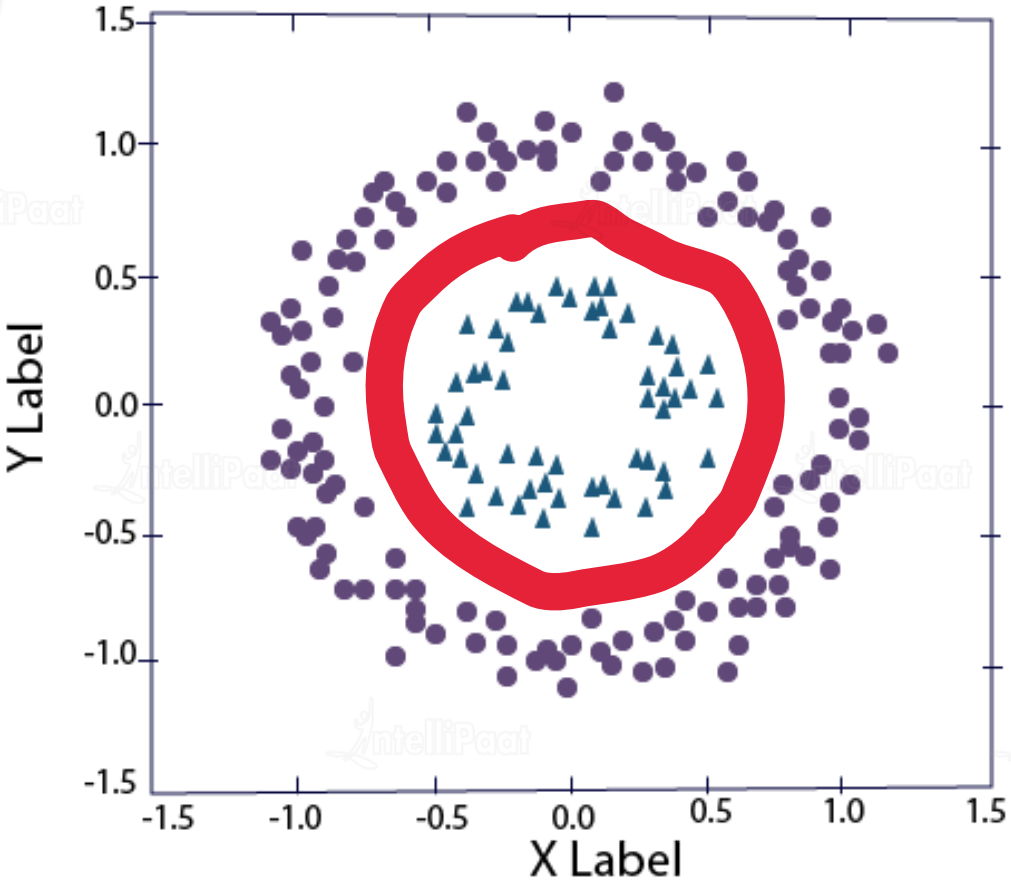
Why Do We Need Activation Functions?



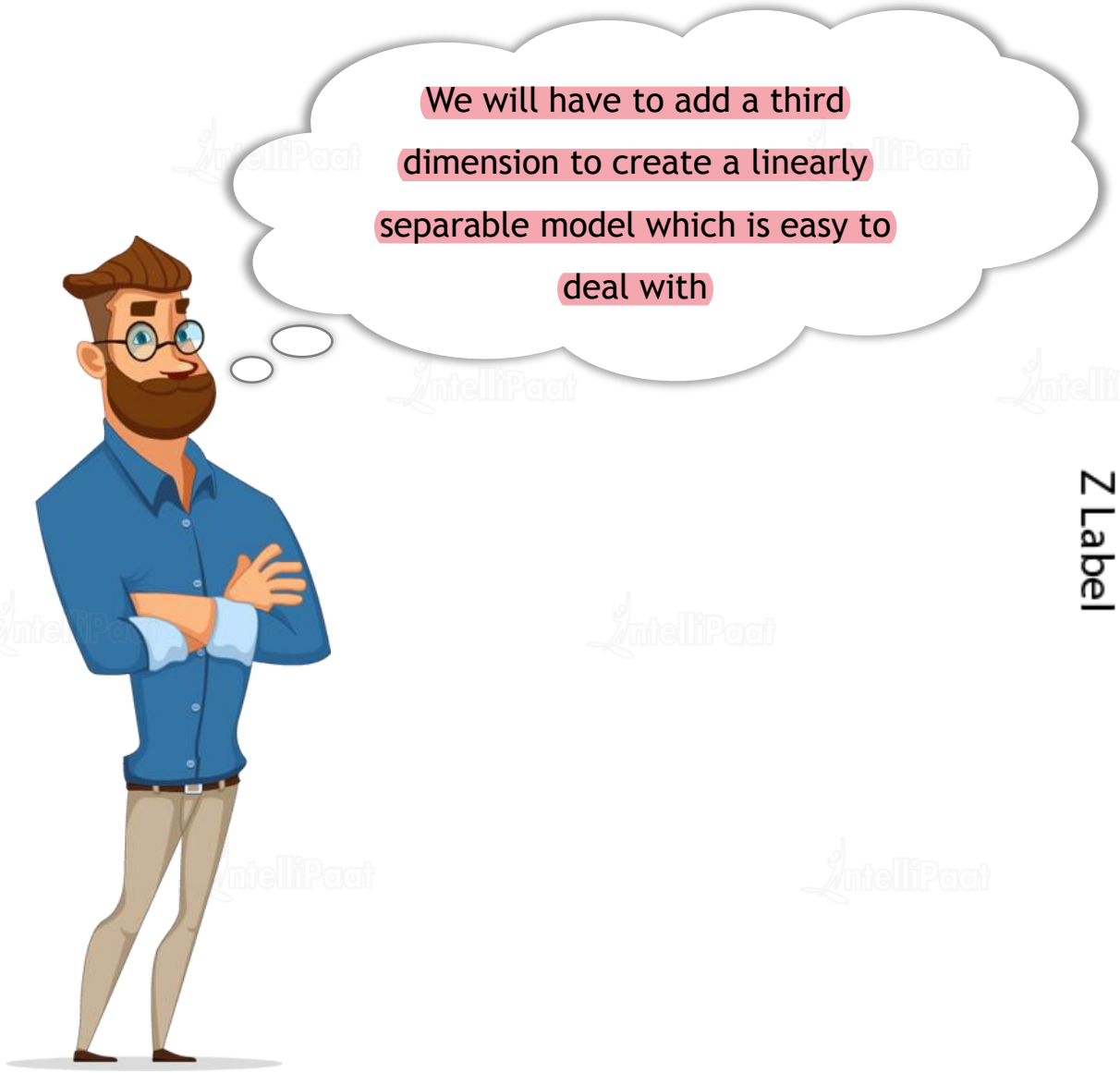


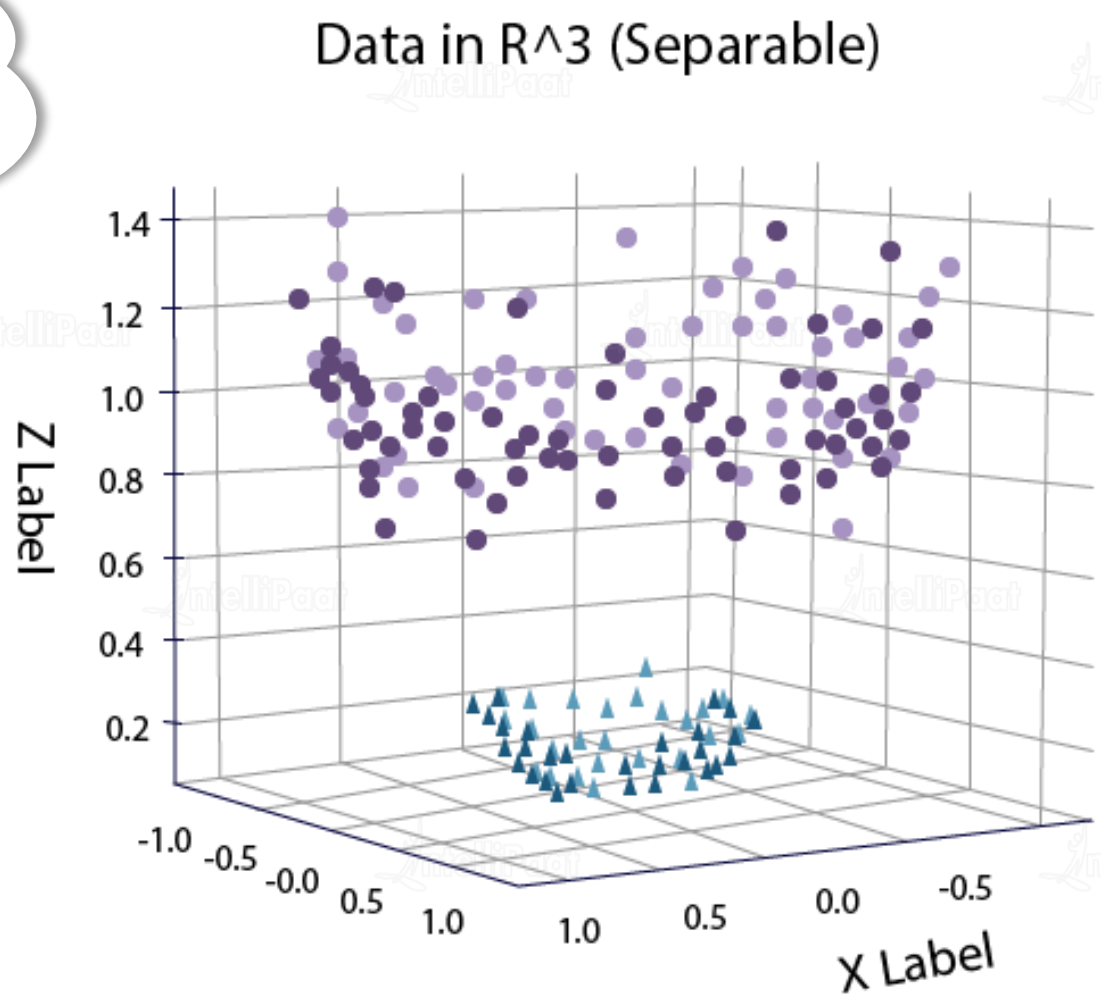
Why Do We Need Activation Functions?





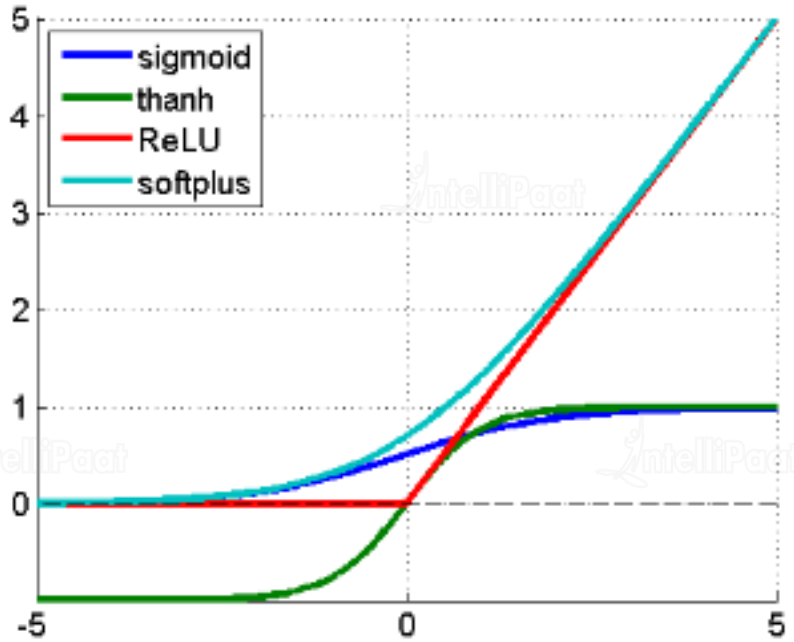
Why Do We Need Activation Functions?





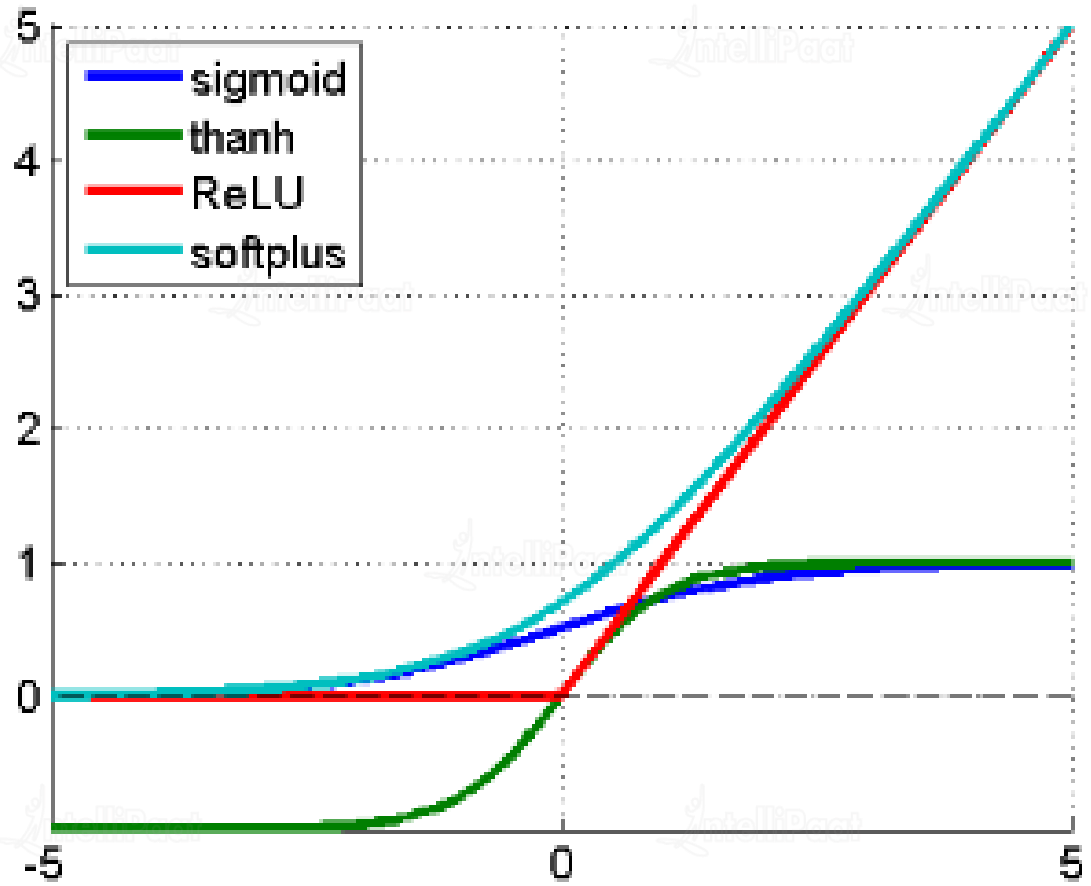
Activation Functions

- They are used to *convert an input signal* of a node in an artificial neural network to an *output signal*
- That output signal now is used as an input in the next layer in the stack
- Activation functions introduce *non-linear properties* to our network
- A neural network without an activation function is essentially just a *linear regression model*
- The activation function does non-linear transformation to the input making it capable to *learn and perform more complex tasks*



Types of Activation Functions

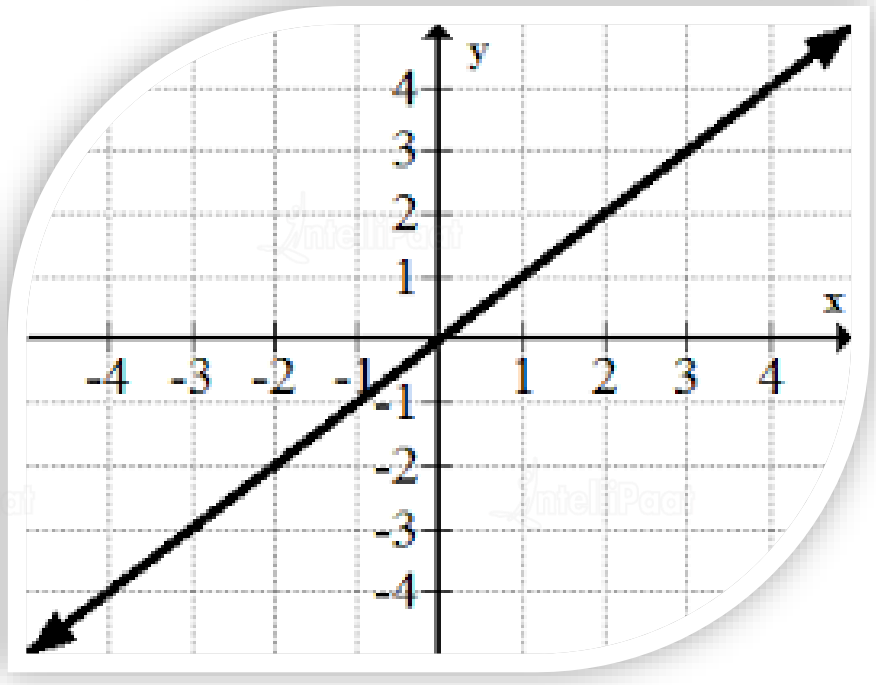
- *Identity*
- *Binary Step*
- *Sigmoid*
- *Tanh*
- *ReLU*
- *Leaky ReLU*
- *Softmax*



Identity Function

$$f(x) = x$$

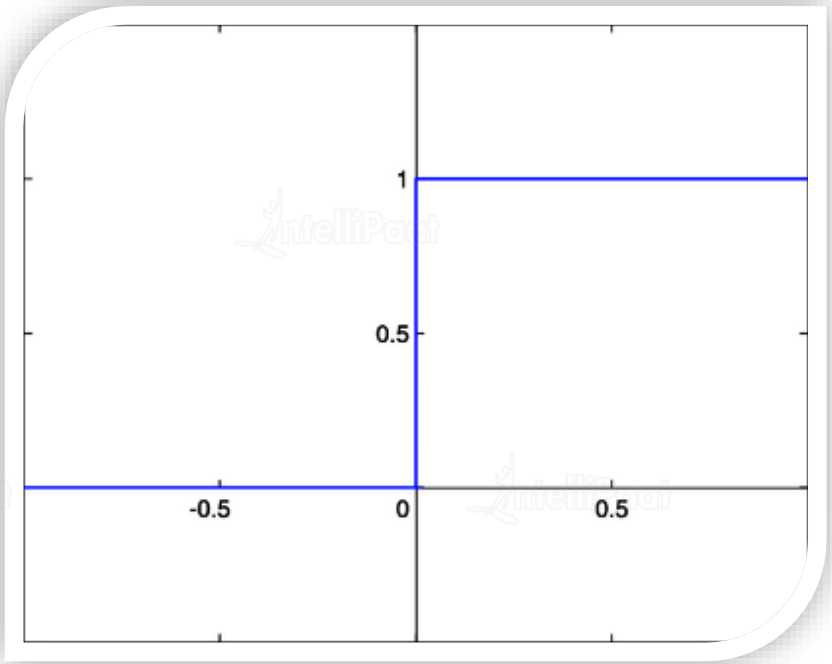
- A straight line function where activation is proportional to input
- No matter how many layers we have, if all of them are linear in nature, the final activation function of the last layer will be nothing but just a linear function of the input of the first layer
- We use a linear function to solve a linear regression problem
- Range: $(-\infty, \infty)$



Binary Step Function

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

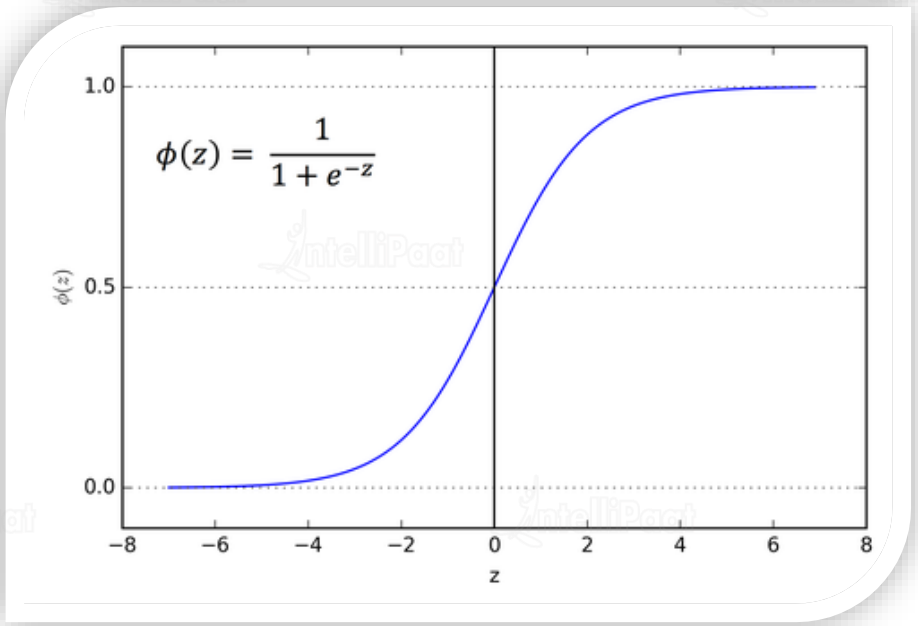
- It is also known as the Heaviside step function, or the unit step function, which is usually denoted by H or θ, is a discontinuous function
- Its value is 0 for the negative argument and 1 for the positive argument
- It depends on the threshold value we define
- We use the binary step function to solve a binary classification problem
- Range: (0,1)



Sigmoid Function

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

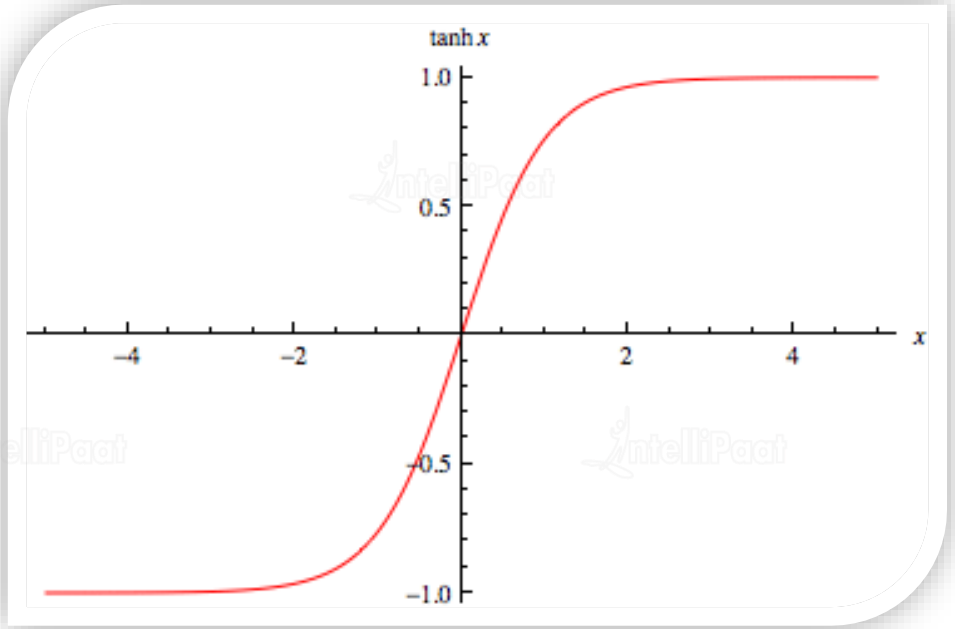
- The sigmoid function is an activation function where it scales values between 0 and 1 by applying a threshold
- When we apply the weighted sum in the place of *x*, the values are scaled in between 0 and 1
- Large negative numbers are scaled toward 0, and large positive numbers are scaled toward 1
- Range: (0,1)



Tanh Function

$$f(x): tan\,h(x) = \frac{2}{1+e^{-2x}} - 1$$

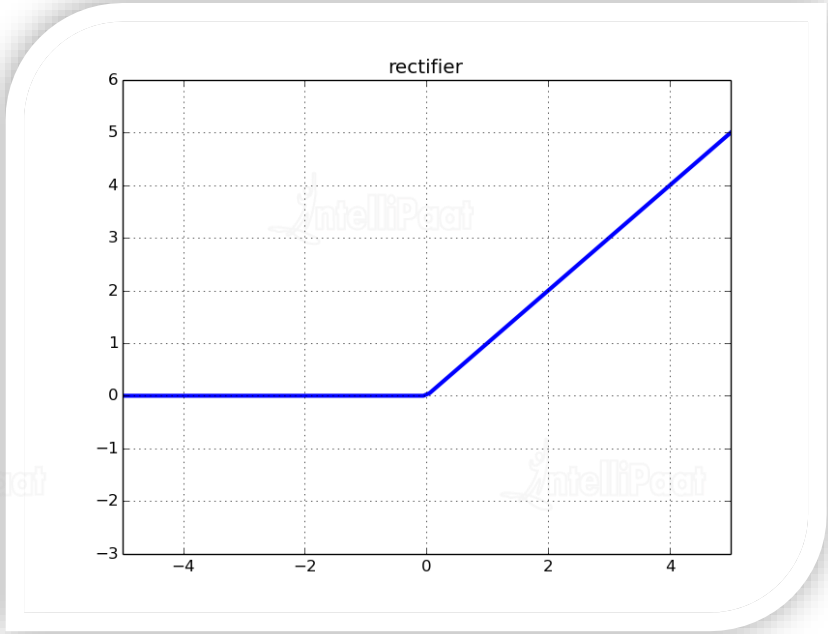
- It is a hyperbolic trigonometric function
- The Tanh activation works almost always better than sigmoid functions as optimization is easier in this method
- The advantage of Tanh is that it can deal more easily with negative numbers
- It is actually a mathematically shifted version of the sigmoid function
- Range: (−1,1)



ReLU Function

$$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x => 0 \end{cases}$$

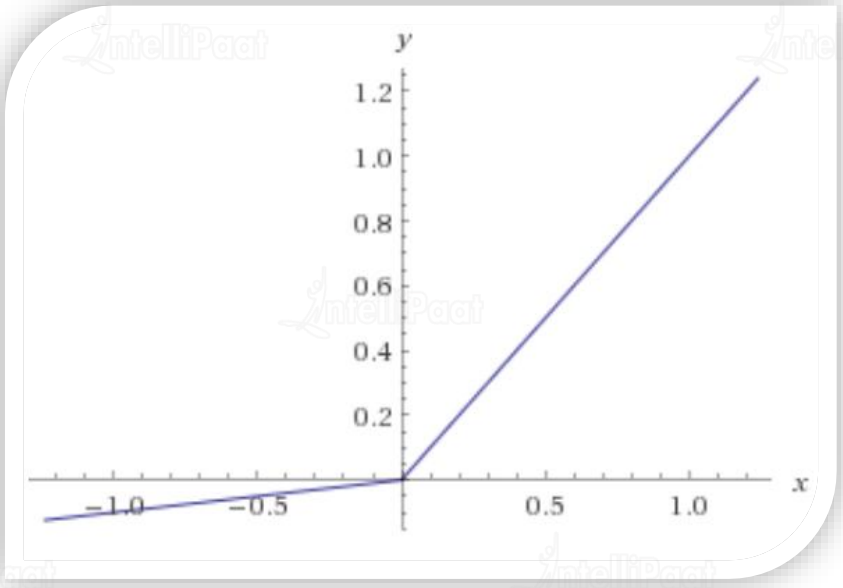
- ReLU stands for rectified linear unit
- It is the most widely used activation function
- It is primarily implemented in Hidden Layers of the neural network
- This function allows only the maximum values to pass during the front propagation as shown in the graph below
- Range: (0,∞)



Leaky ReLU Function

$$f(x) = \begin{cases} 0.01x \ for \ x < 0 \\ \\ x \ for \ x => 0 \end{cases}$$

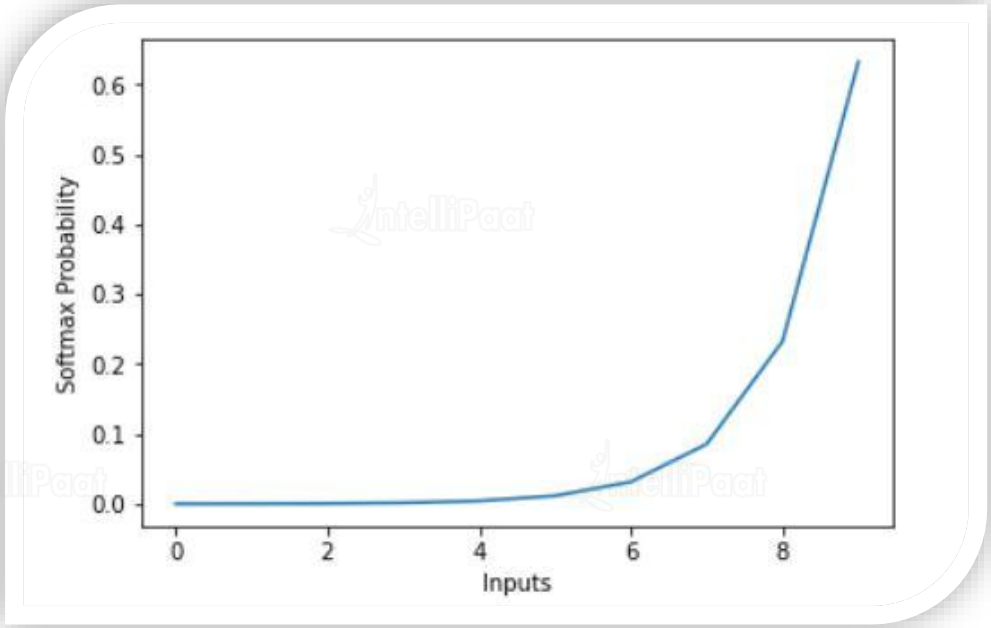
- Leaky ReLU allows a small negative value to pass during the back propagation if we have a dead ReLU problem
- This eventually activates the neuron and brings it down
- Range: (−∞,∞)



Softmax Function

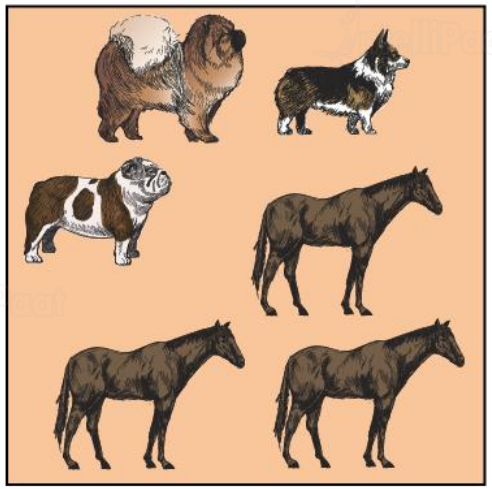
$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, j = 1, 2, .. K$$

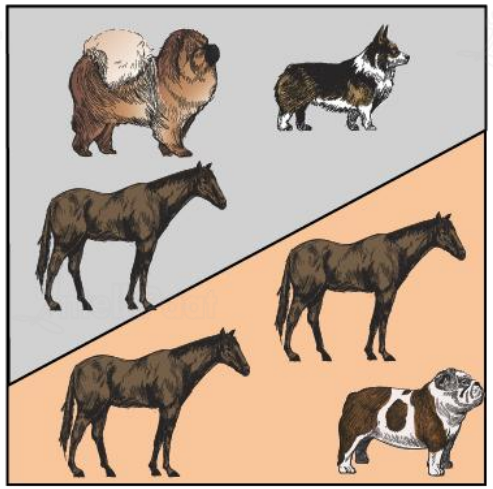
- The Softmax function is used when we have multiple classes
- It is useful for finding out the class which has the max. probability
- The Softmax function is ideally used in the Output Layer of the classifier where we are actually trying to attain the probabilities to define the class of each input
- Range: (0,1)



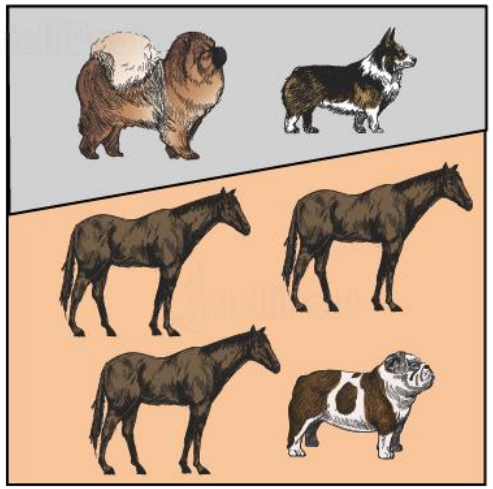
Training a Perceptron

By training a perceptron, we try to find a line, plane, or some hyperplane which can accurately separate two classes by adjusting weights and biases

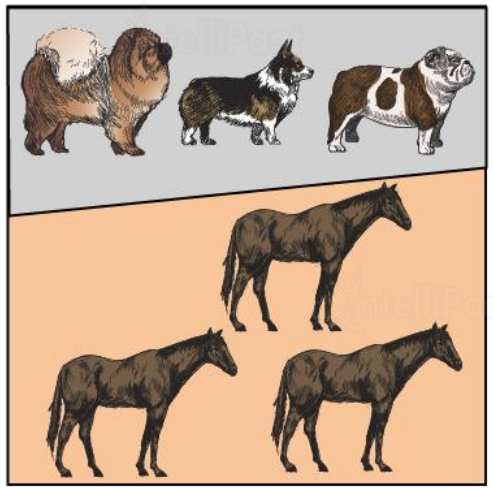




Error = 2

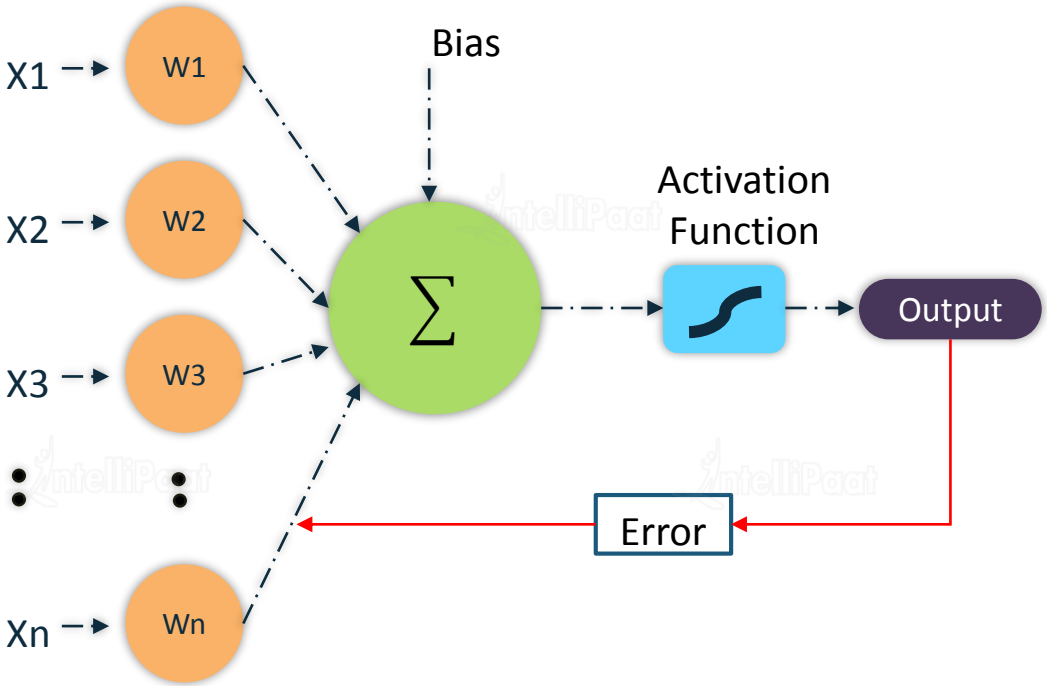


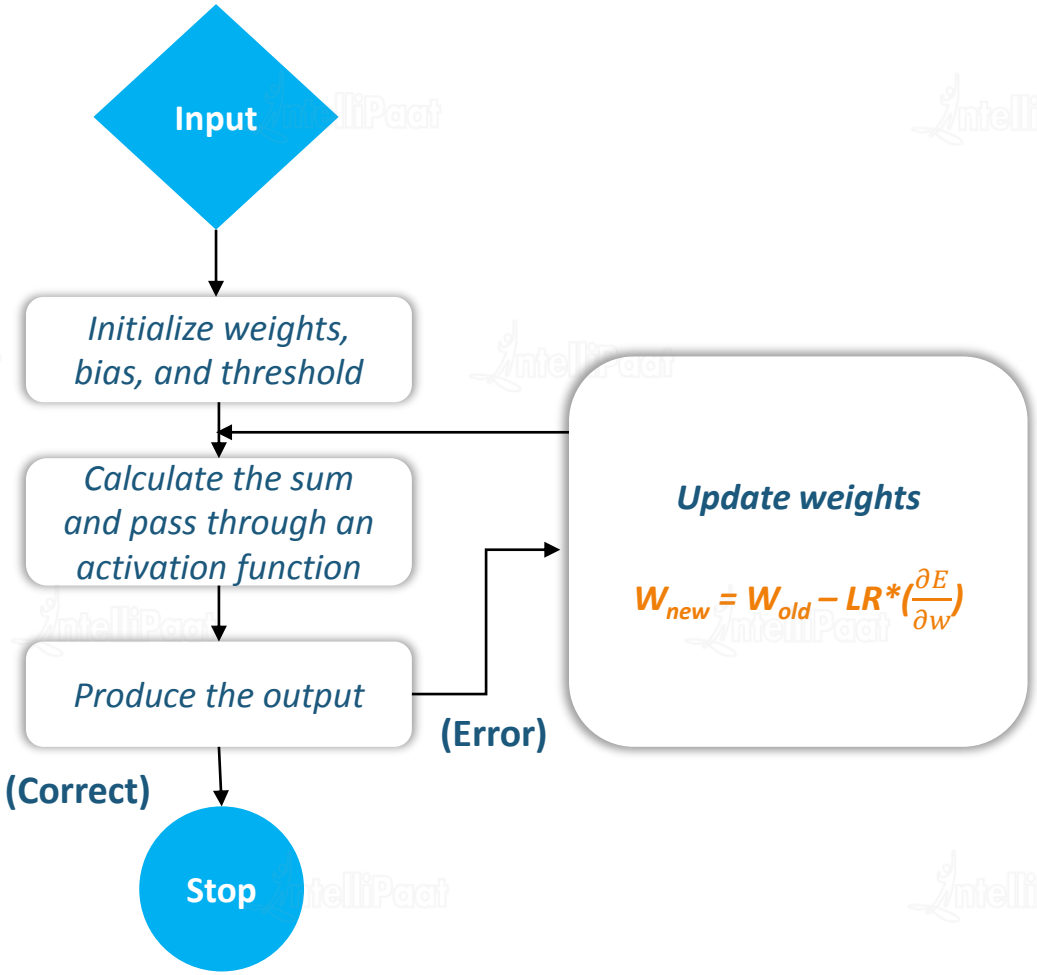
Error = 1



Error = 0

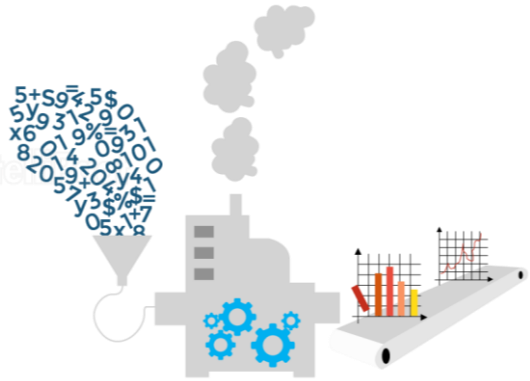
Perceptron Training Algorithm





Benefits of Using Artificial Neural Networks

Organic Learning



Non-linear Data Processing

Fault Tolerance

Self-repairing

Deep Learning Frameworks

These Deep Learning libraries help in implementing artificial neural networks

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

TensorFlow is an open-source software library for high-performance numerical computations

Developed by Google

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

Google Translate

- Natural language processing
- Forecasting
- Text classification
- Tagging

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

Tensor Board	Used for visualizing TensorFlow computations and graphs
TensorFlow Serving	Used for rapid deployment of new algorithms/experiments while retaining the same server architecture and APIs

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

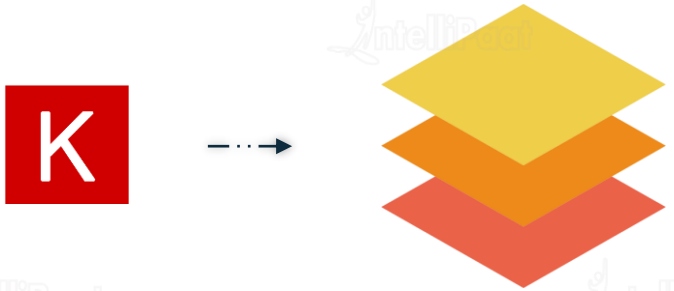
A high-Level API which can run on top of TensorFlow, Theano, or CNTK

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

A recurrent neural network

A convolutional neural network

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet



Stacks layers on top

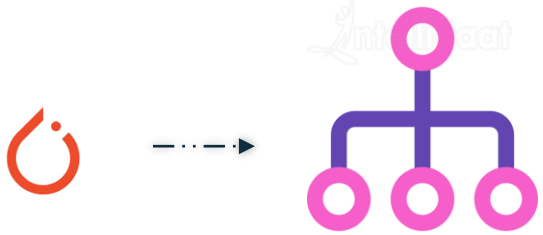
- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

A scientific computing framework developed by Facebook

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

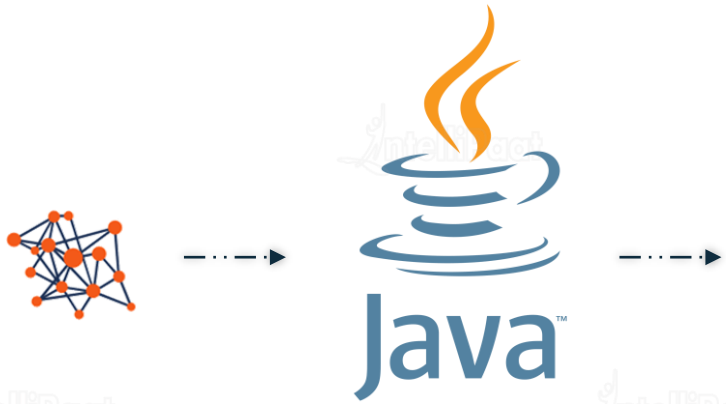
‘Pythonic’ in nature

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet



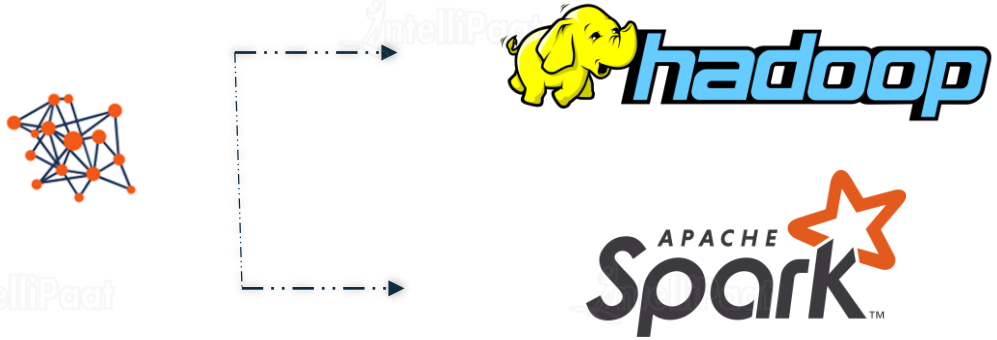
Offers dynamic computational graphs

- TensorFlow
- Keras
- PyTorch
- **DL4J**
- MXNet



A Deep Learning programming library written for Java

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet



TensorFlow

Keras

PyTorch

DL4J

MXNet

Image recognition

Fraud detection

Text mining

Parts of speech tagging

Natural language processing

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

Developed by Apache Software Foundation

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

- TensorFlow
- Keras
- PyTorch
- DL4J
- MXNet

- Imaging
- Speech recognition
- Forecasting
- NLP

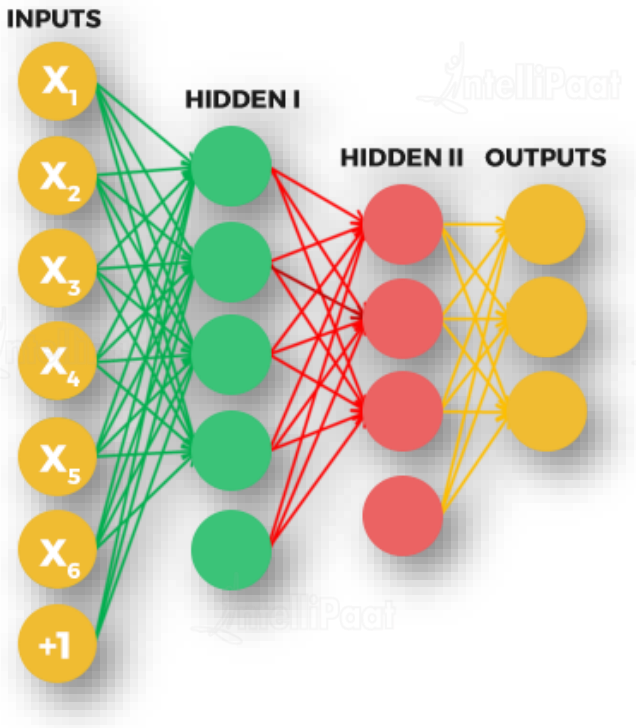
What Are Tensors?

A tensor is a multi-dimensional array in which data is stored

Tensor is given as an input to a neural network

5	1	8
3	5	4
6	9	0
4	7	2
5	9	3

Tensor



Tensor Rank

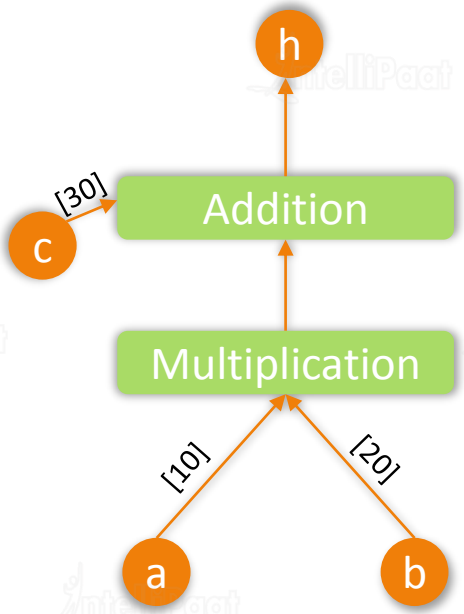
Tensor rank represents the dimension of the *n*-dimensional array

Rank	Math Entity	Example
0	Scalar (magnitude only)	s = 483
1	Vector (magnitude and direction)	v = [1.1, 2.2, 3.3]
2	Matrix (table of numbers)	m = [1, 2, 3], [4, 5, 6], [7, 8, 9]]
3	3-Tensor (cube of numbers)	t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]
n	*n*-Tensor	……

Computational Graph

Computation is done in the form of a graph

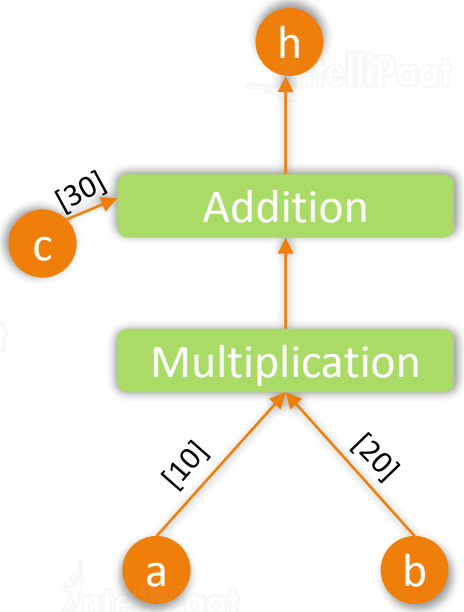
$$a = 10$$
$$b = 20$$
$$c = 30$$
$$h = (a * b) + c$$

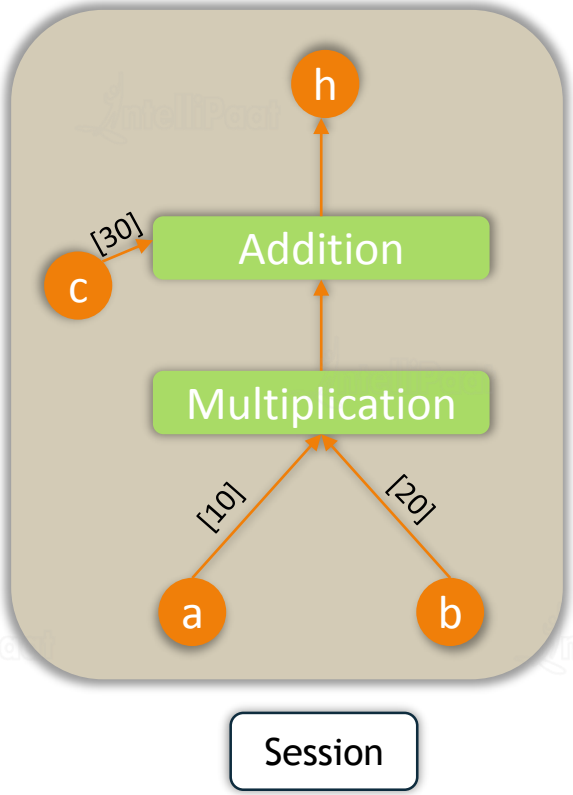


Computational Graph

The computational graph is executed inside a session

$$a = 10$$
$$b = 20$$
$$c = 30$$
$$h = (a * b) + c$$

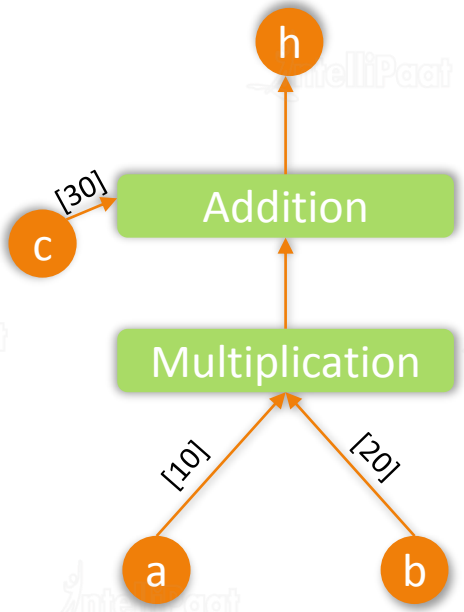




Computational Graph

The computational graph is executed inside a session

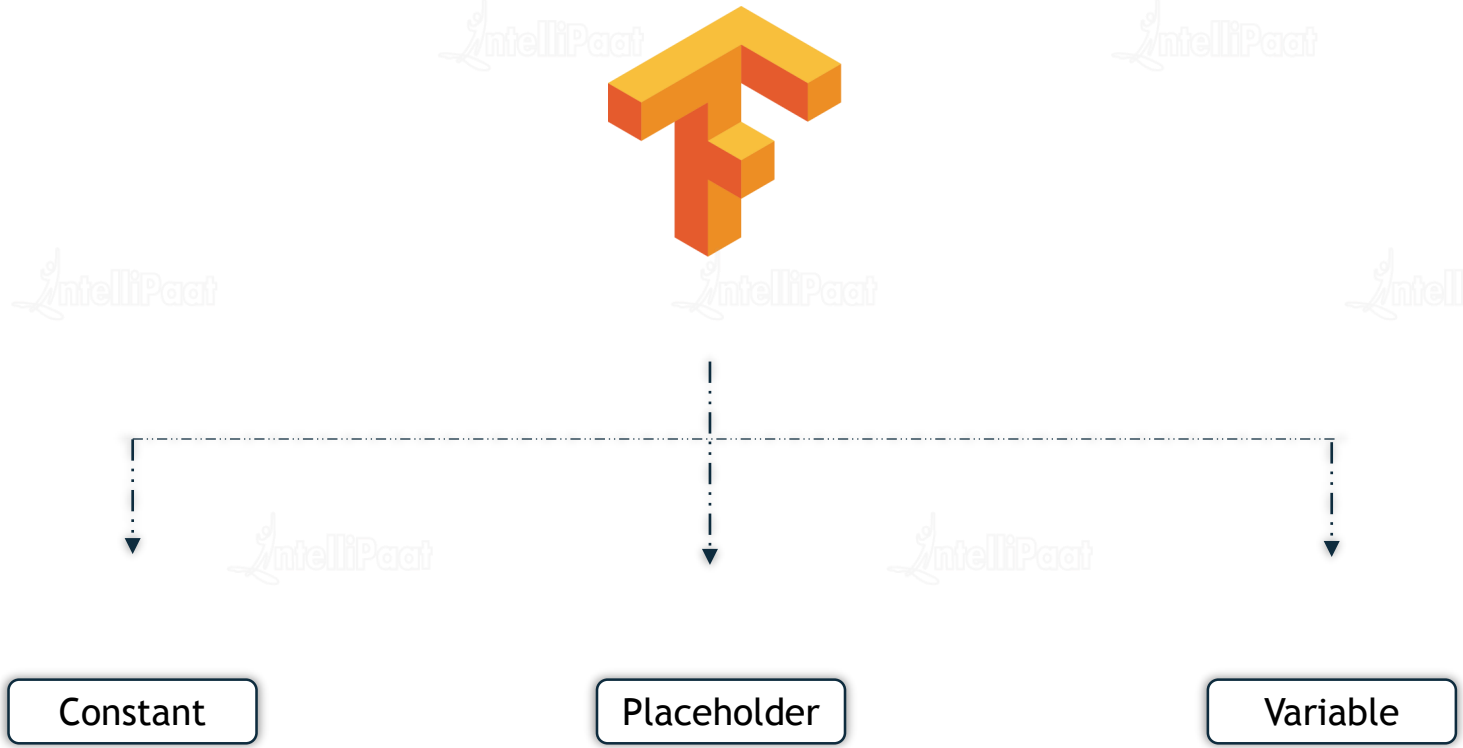
$$a = 10$$
$$b = 20$$
$$c = 30$$
$$h = (a * b) + c$$



Node -> Mathematical operation

Edge -> Tensor

Program Elements in TensorFlow



- Constant
- Placeholder
- Variable

Constants are program elements whose values do not change

```
a=tf.constant(10)
```

```
b=tf.constant(20)
```

Constant

Placeholder

Variable

A placeholder is a program element to which we can assign data at a later time

```
x=tf.placeholder(tf.float32)
```

```
y=tf.placeholder(tf.string)
```

Constant

Placeholder

Variable

A variable is a program element which allows us to add new trainable parameters to the graph

```
W=tf.Variable([3],tf.float32)
```

```
b=tf.Variable([0.4],tf.float32)
```

Quiz

Quiz 1

A tensor is a single-dimensional array in which data is stored

A	True
B	False

Answer 1

A tensor is a single-dimensional array in which data is stored

A	True
B	False

Quiz 2

How many layers does a standard Neural Network has?

A	1
B	2
C	3
D	4 or more

Answer 2

How many layers does a standard Neural Network has?

A	1
B	2
C	3
D	4 or more

Quiz 3

Perceptron is other name of Neurons

A	Yes
B	No

Answer 3

Perceptron is other name of Neurons

A	Yes
B	No

Thank you!

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor