An introduction to

Graph Neural Network(GNN)

Part 1

Dr. Jamshaid Ul Rahman

Abdus Salam School of Mathematical Sciences,

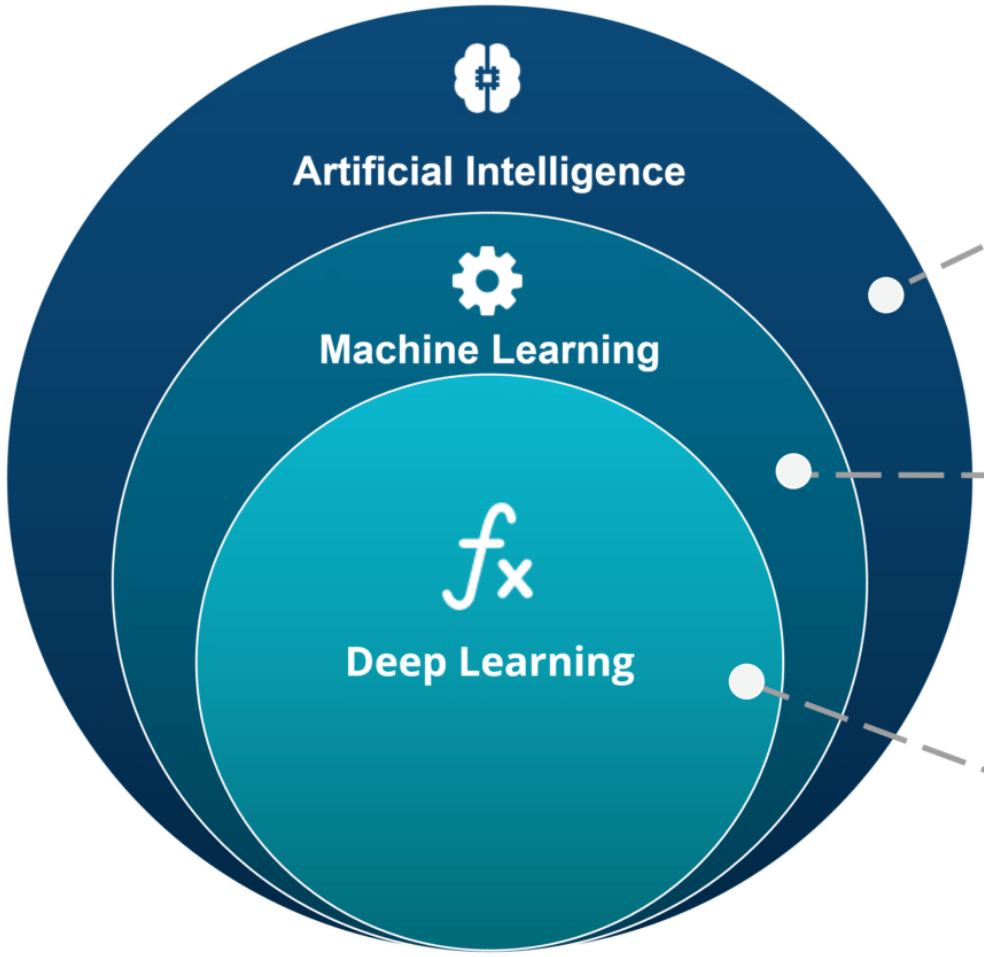
GC University, Lahore

Part 1 covers the basics of machine learning and graphs, setting the foundation for understanding GNNs.

Acknowledgment:

This presentation has been prepared by **Hasnain Ali**, an M.Phil. student at the Abdus Salam School of Mathematical Sciences, Govt. College University, Lahore. Special thanks to him and Areen Rasool, a Ph.D. student, for their valuable assistance.

AI vs ML vs DL



ARTIFICIAL INTELLIGENCE

A technique which enables machines to mimic human behaviour

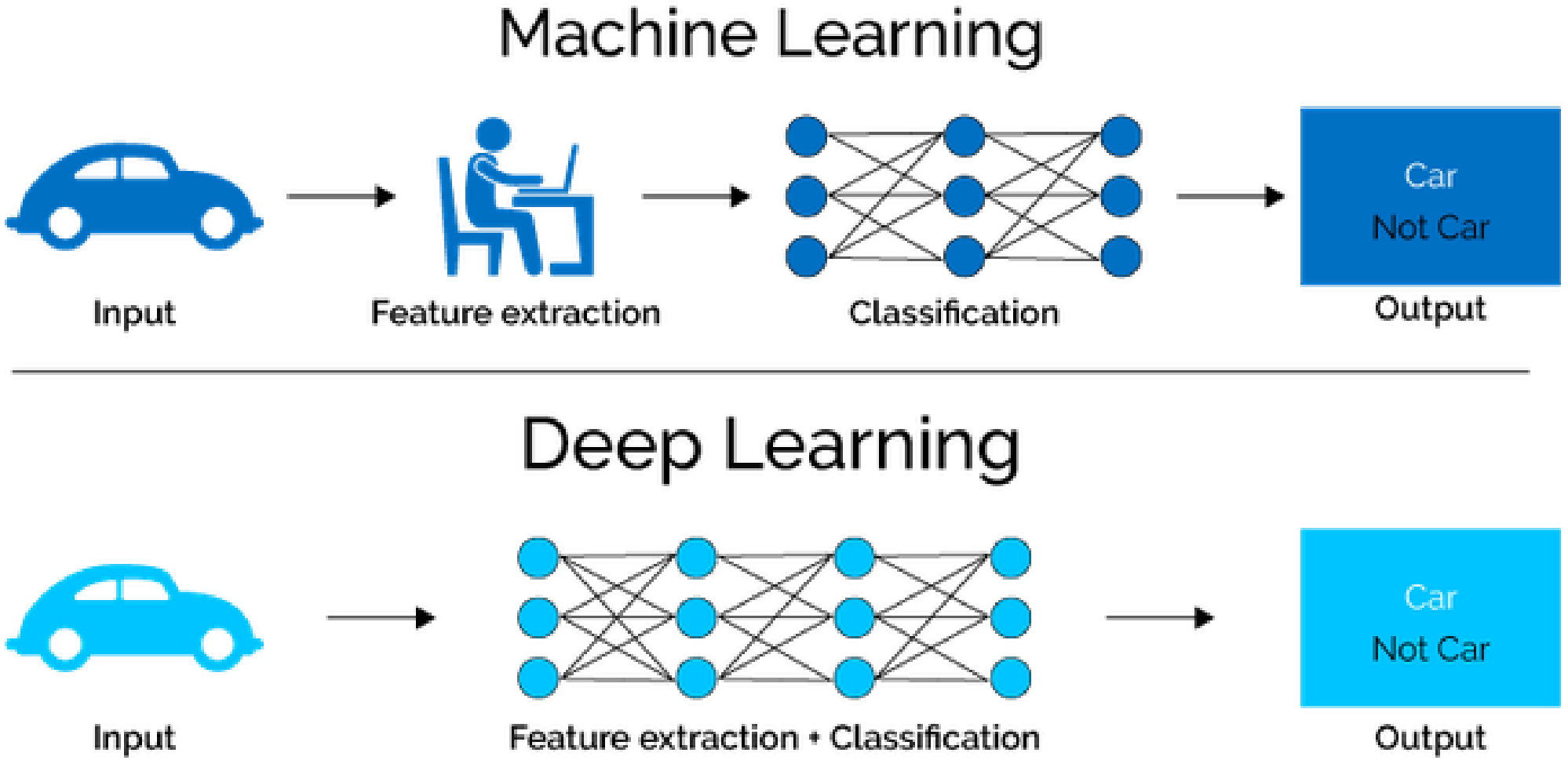
MACHINE LEARNING

Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING

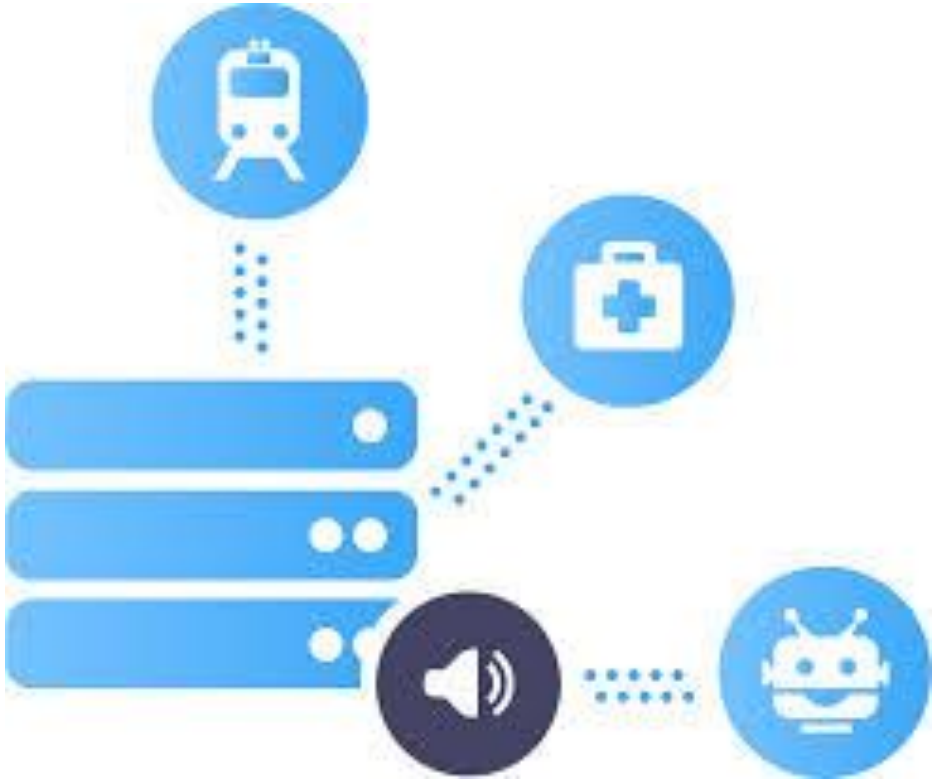
Subset of ML which make the computation of multi-layer neural network feasible

ML vs DL

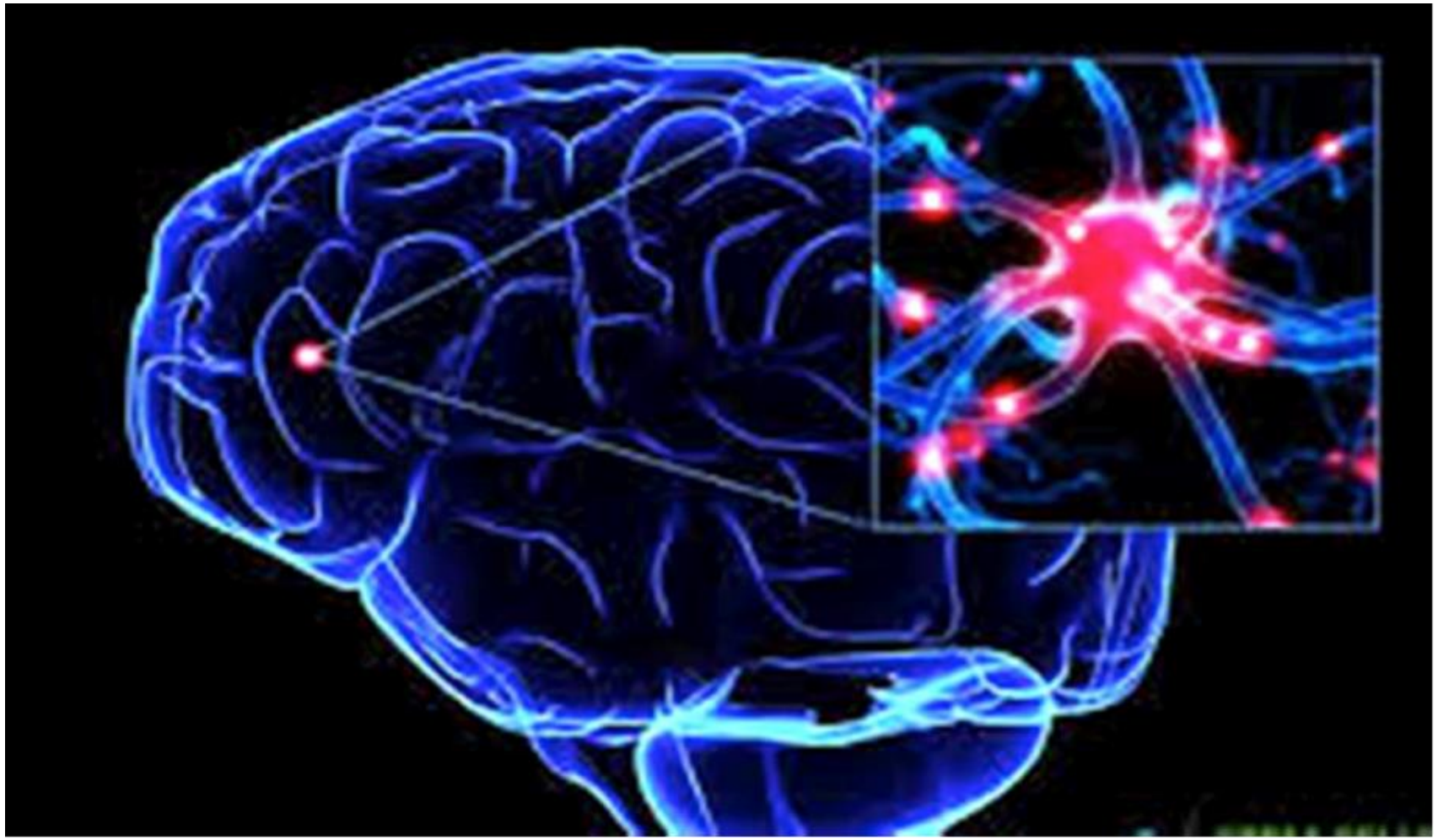

Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction • Classification
Output

So, What do you think, what is Deep Learning

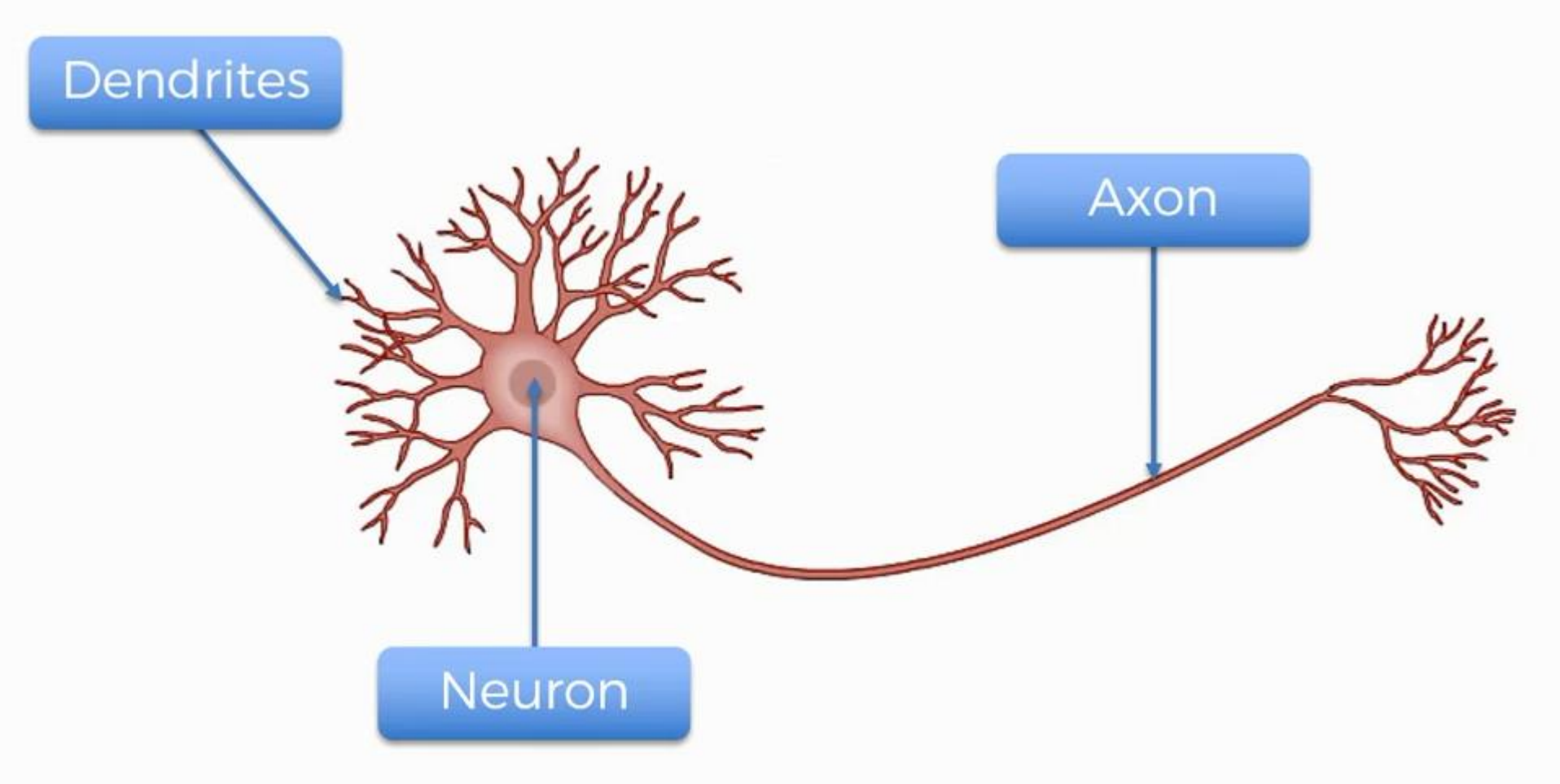
Deep learning is a machine learning technique that learns features and task directly from the data, where data may by images, graph, text or sound!



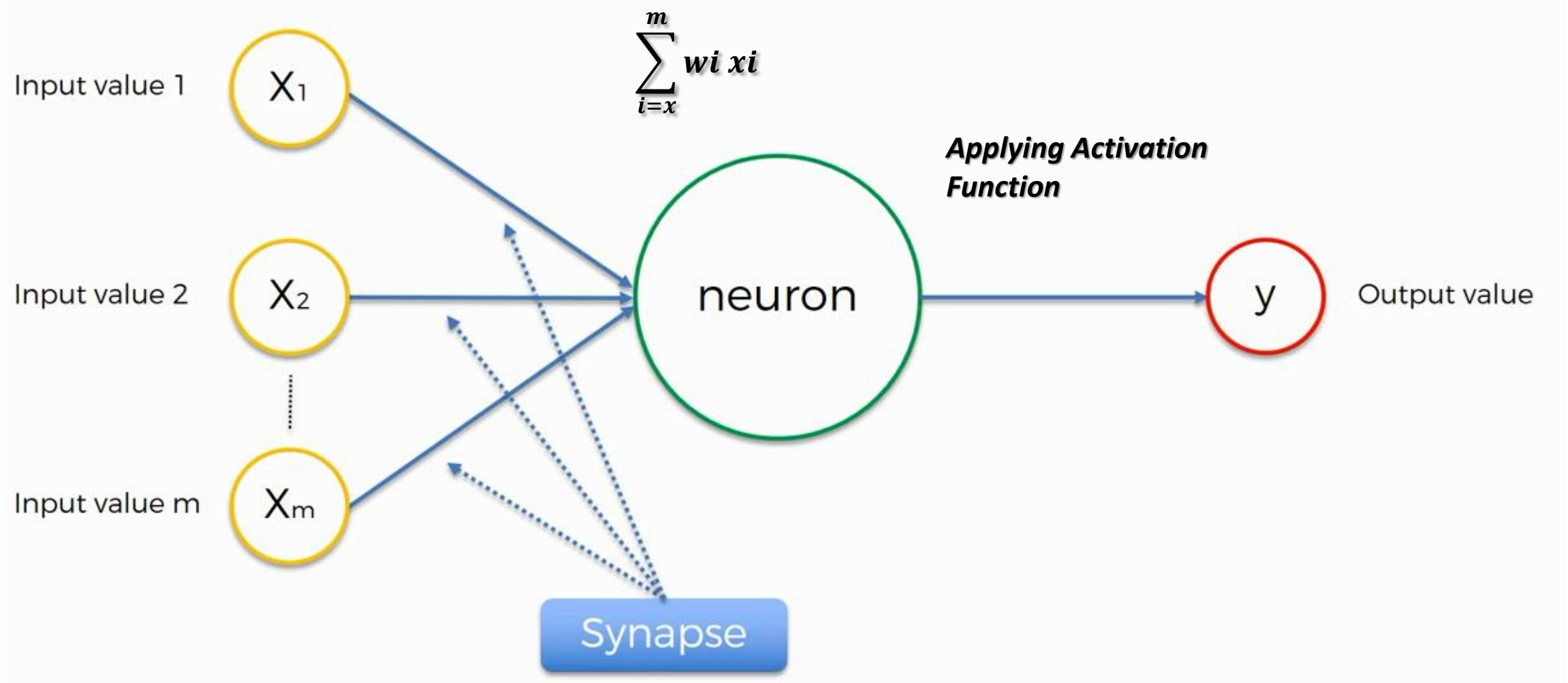
What is Neural



Neural Working



Artificial Neural Network



What is an Activation Function?

- **Activation functions** are crucial in neural networks as they determine whether a neuron should be activated or not by calculating the weighted sum and adding bias. They help introduce non-linearity into the network, enabling the model to learn complex patterns.

$$Y = \text{Activation}(\Sigma(weight * input) + bias)$$

- The activation function is the non linear transformation that we do over the input signal. This transformed output is then seen to the next layer of neurons as input.

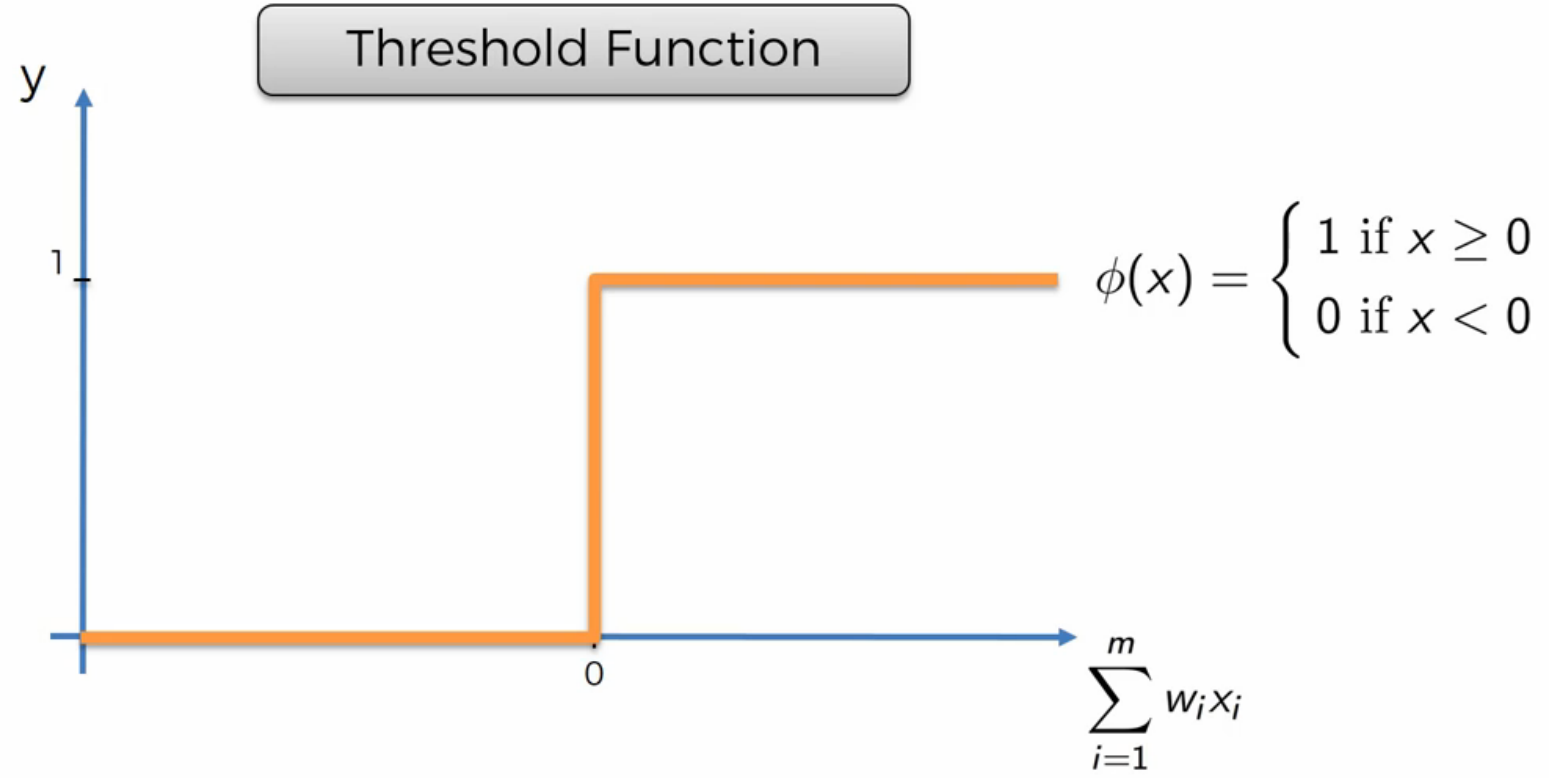
Activation Function

- Here are some common activation functions:

1. **Threshold**
2. **Sigmoid**
3. **Tanh**
4. **ReLU**
5. **Leaky ReLU**
6. **Softmax**

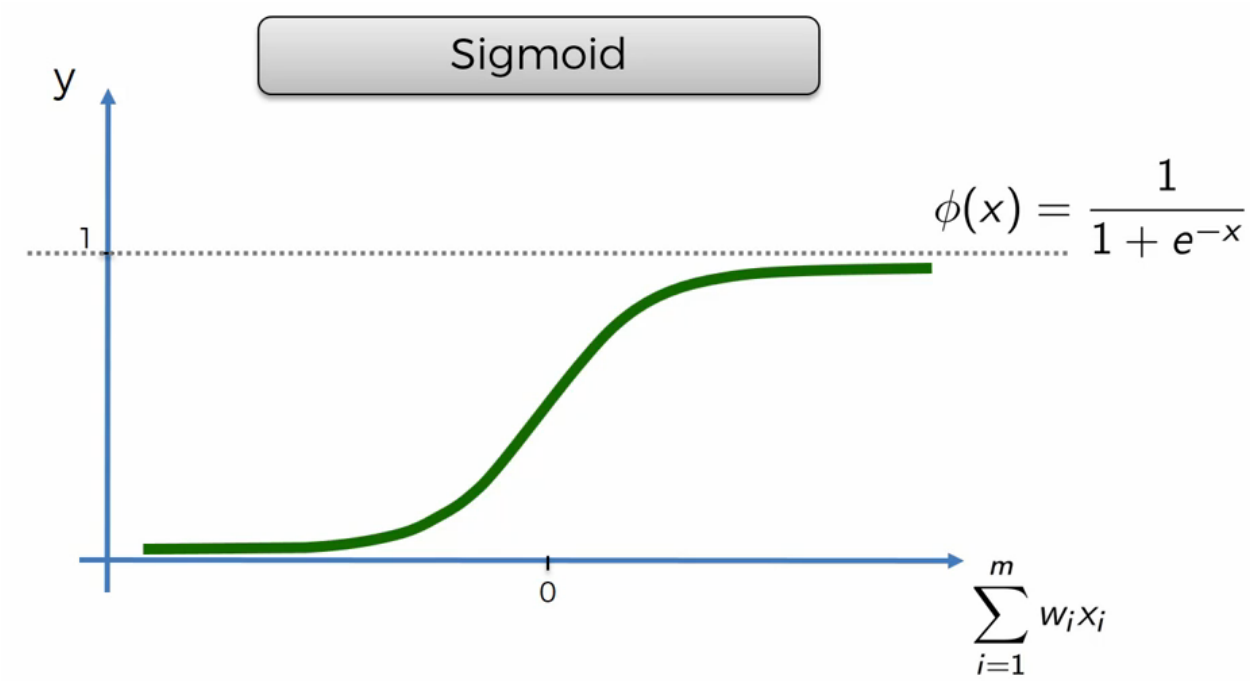
Threshold Function

- A threshold function, often used in the context of neural networks and perceptrons, is a type of activation function that is used to make a binary decision.



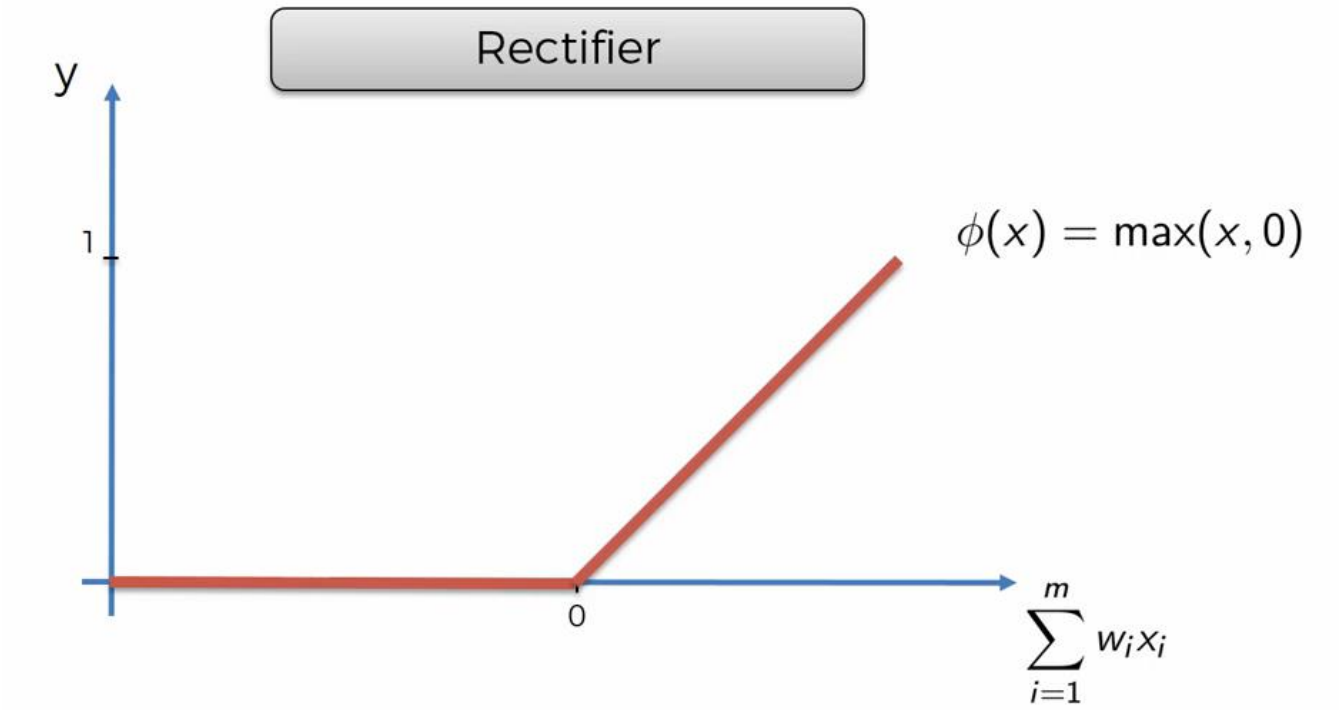
Sigmoid Function

- The Sigmoid Function curve looks like a S-shape
- This function reduces extreme values or outliers in data without removing them.
- It converts independent variables of near infinite range into simple probabilities between 0 and 1, and most of its output will be very close to 0 or 1.



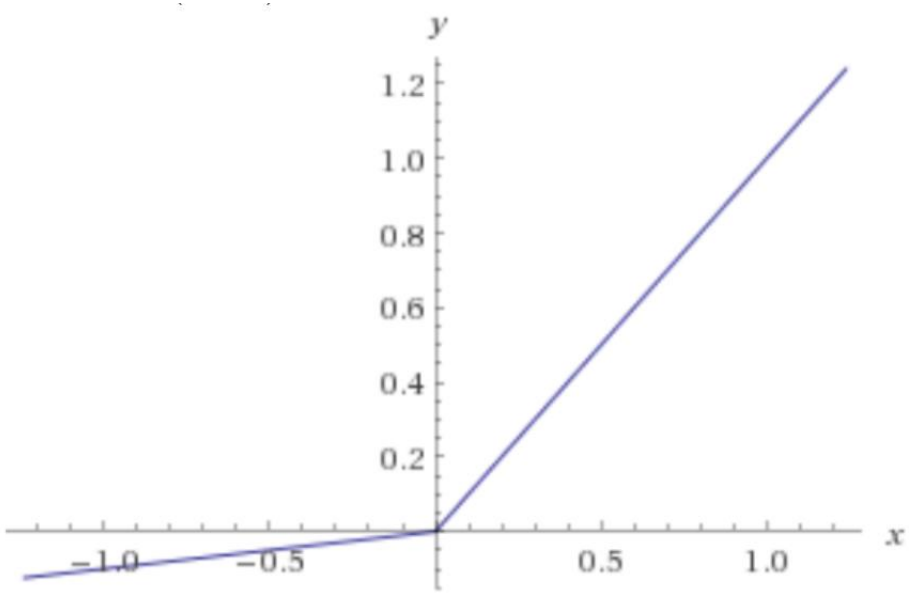
Rectifier (Relu) Function

- ReLU is the most widely used activation function while designing networks today. First things, the ReLU function is non linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.



Leaky Relu Function

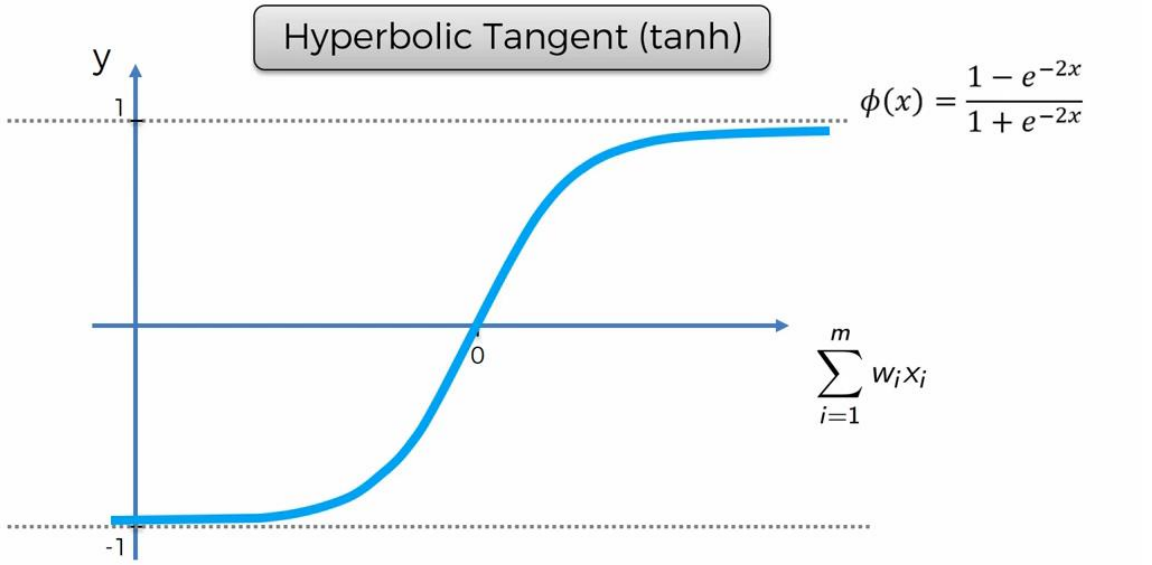
- Leaky ReLU function is nothing but an improved version of the ReLU function. As we saw that for the ReLU function, the gradient is 0 for x<0, which made the neurons die for activations in that region. Leaky ReLU is defined to address this problem. Instead of defining the Relu function as 0 for x less than 0, we define it as a small linear component of x.



- What we have done here is that we have simply replaced the horizontal line with a non-zero, non-horizontal line. Here a is a small value like 0.01 or so.

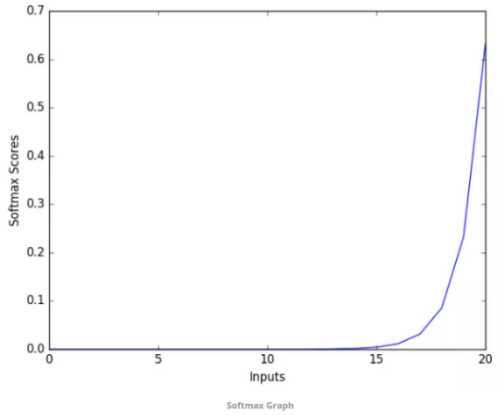
Hyperbolic Tangent Function

- tanh is a hyperbolic trigonometric function, tanh represents the ratio of the hyperbolic sine to the hyperbolic cosine: tanh(x) = sinh(x) / cosh(x)
- Unlike the Sigmoid function, the normalized range of tanh is –1 to 1 The advantage of tanh is that it can deal more easily with negative numbers.
- The hyperbolic tangent (tanh) activation function is primarily used in problems where the data needs to be centered around zero, which helps in faster and more efficient training of neural networks.



Softmax Function (for Multiple Classification)

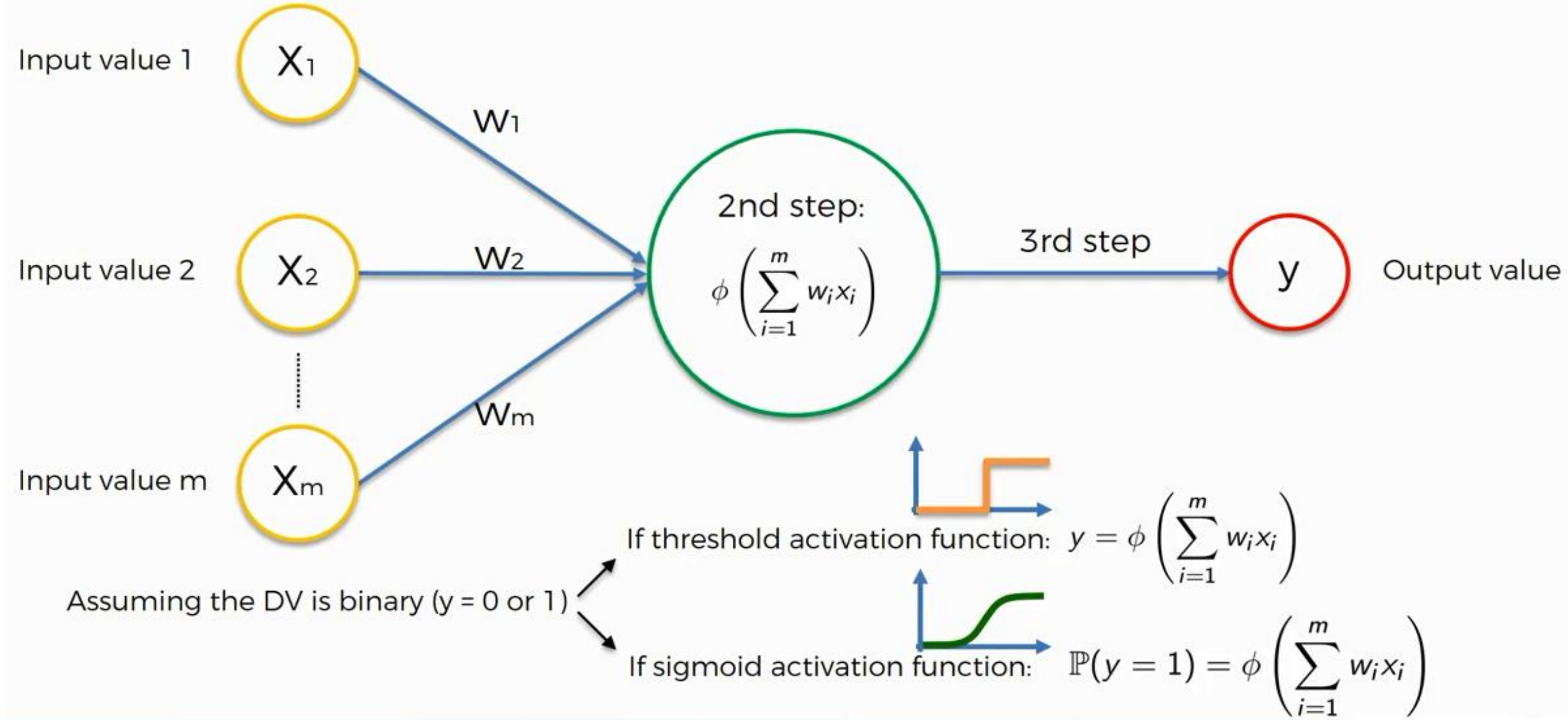
- The **Softmax function** is a type of activation function commonly used in the output layer of neural networks, especially for multi-class classification problems.
- It converts the raw prediction scores (logits) into probabilities, which sum to 1, providing a meaningful probabilistic interpretation of the outputs.



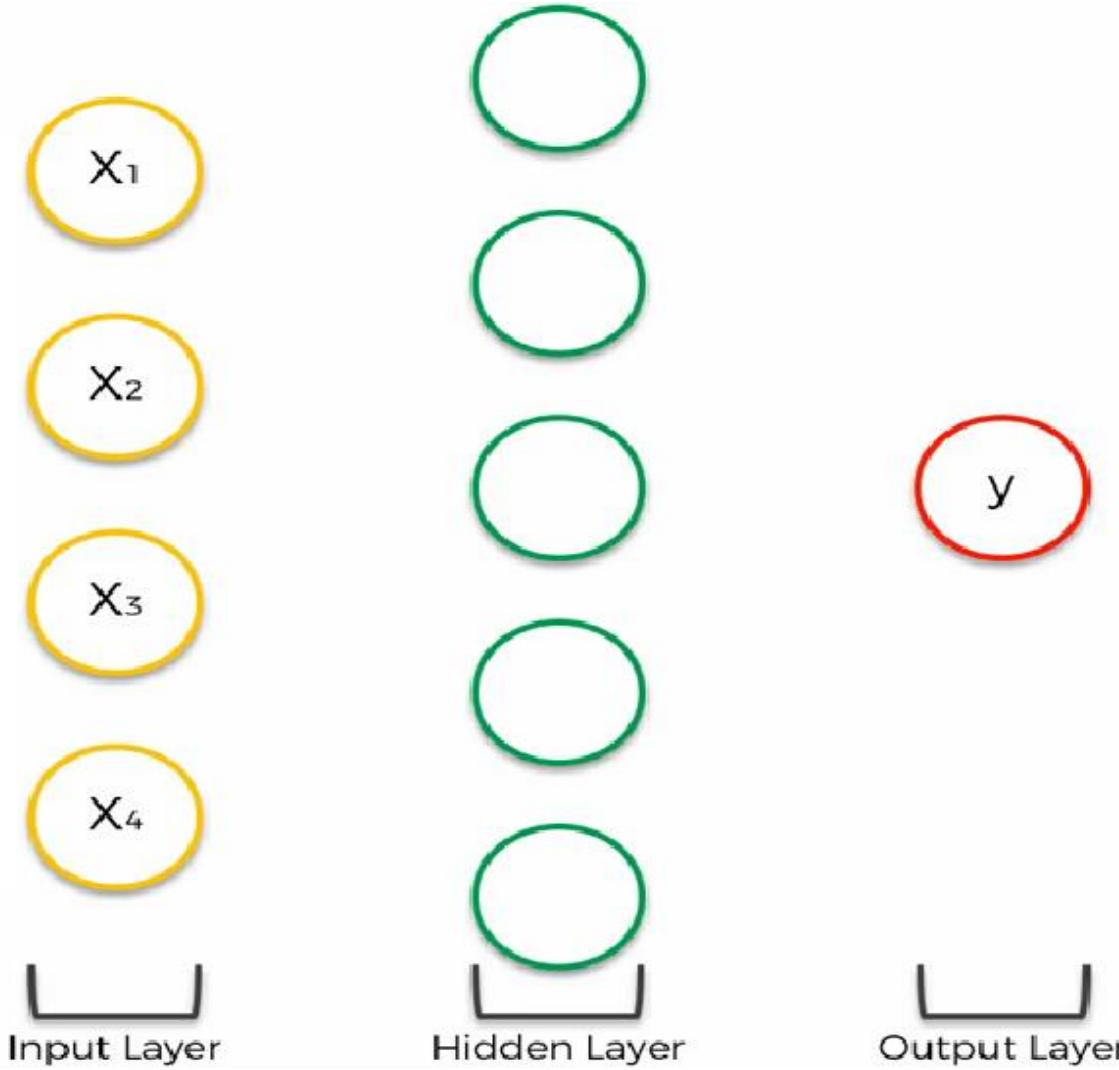
Softmax Graph

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Activation Function Example



How Neural Network Work with many neurons

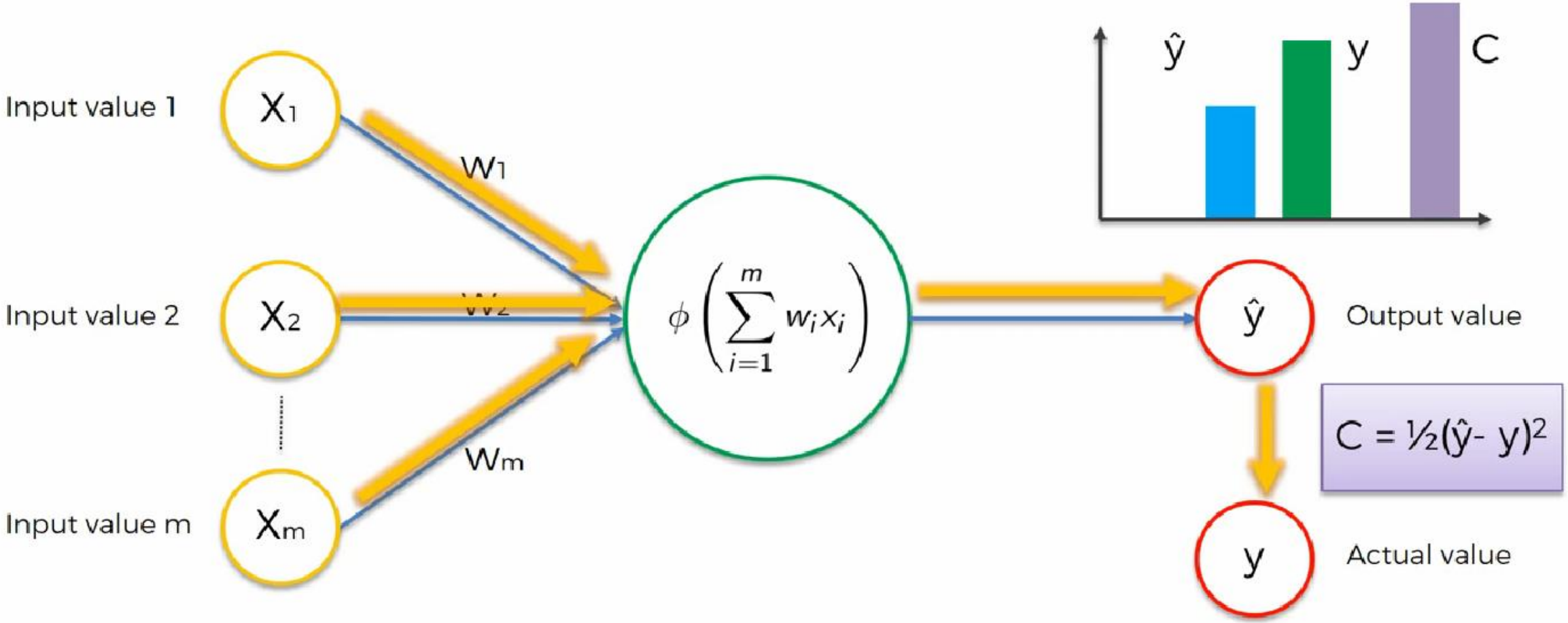


Back Propagation in Deep Learning

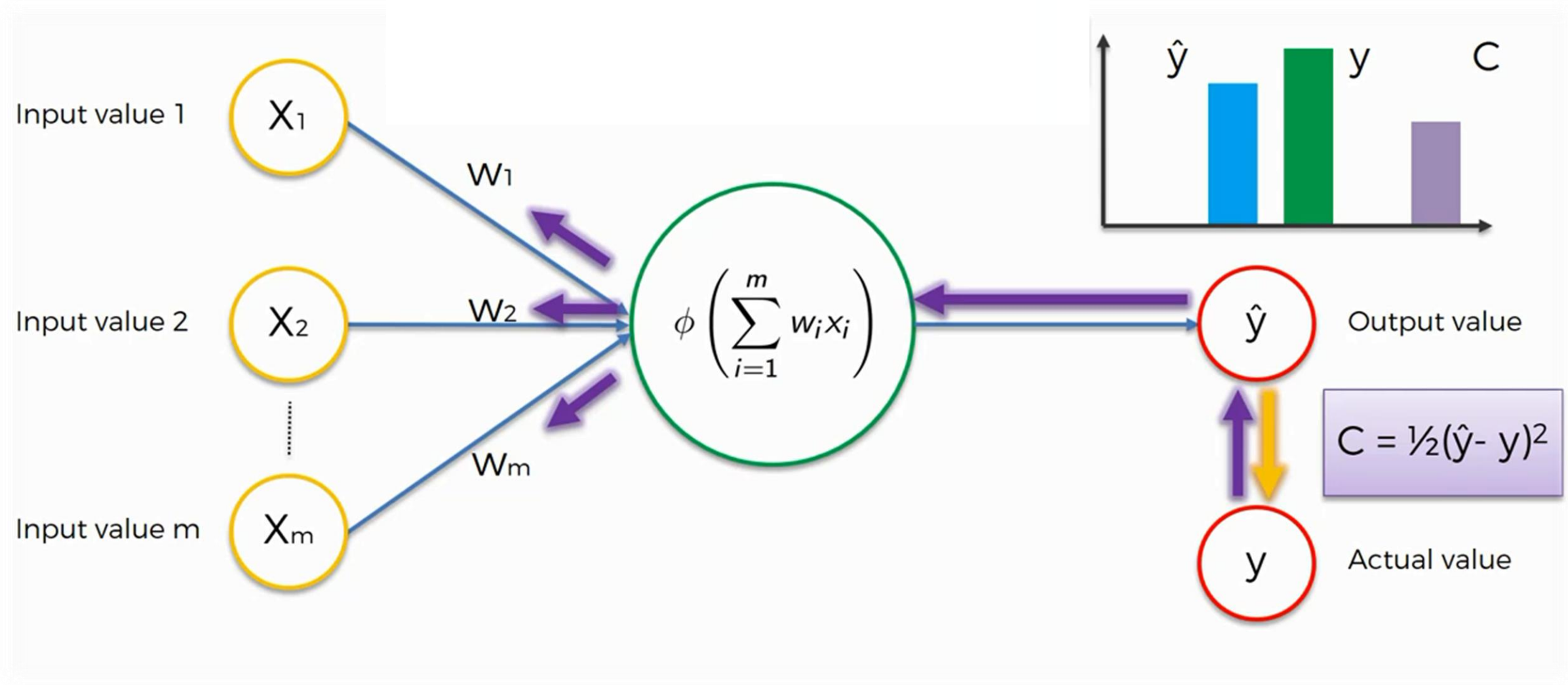
- Back-propagation is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization.

- Backpropagation is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

Back Propagation in Deep Learning



Back Propagation in Deep Learning



What is Bias

- **Bias** is a parameter added to the input of each neuron, along with the weights, to adjust the output along with the weighted sum of the inputs.
- It helps the model to make accurate predictions even when all the input features are zero.
- Essentially, bias allows the activation function to be shifted to the left or right, which enables the neural network to fit the data better. This shift can significantly improve the model's ability to learn patterns and generalize to new data.

output = sum (weights * inputs) + bias

- A simpler way to understand bias is through a constant c of a linear function

y =mx + c

Different Types of Neural Network

- Perceptron (Multilayer Perceptron) & ANN
- Feedforward Neural Network – Artificial Neuron
- Radial Basis Function Neural Network
- Convolutional Neural Network
- Recurrent Neural Network(RNN)
- Graph Neural Network (GNN)

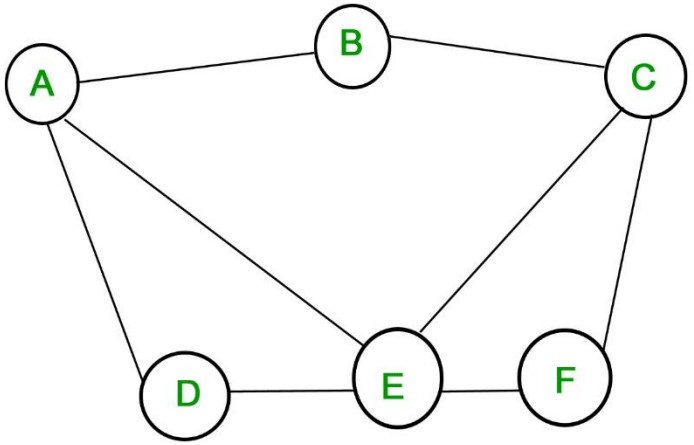
Graph Neural Network (GNN)

- A **Graph Neural Network (GNN)** is a type of artificial neural network specifically designed to handle data that is structured as a graph.
- **GNNs** are used to analyze and make predictions on this relational data, making them particularly useful in applications such as social network analysis, recommendation systems, and biological network modeling

Introduction to Graphs and Their Terminologies

Graph

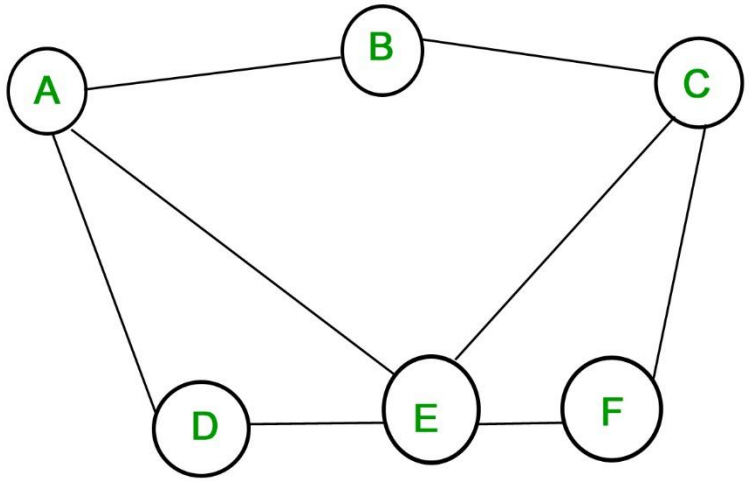
- A **graph** in mathematics and computer science is a structure made up of nodes (also called vertices) and edges (also called links or arcs).



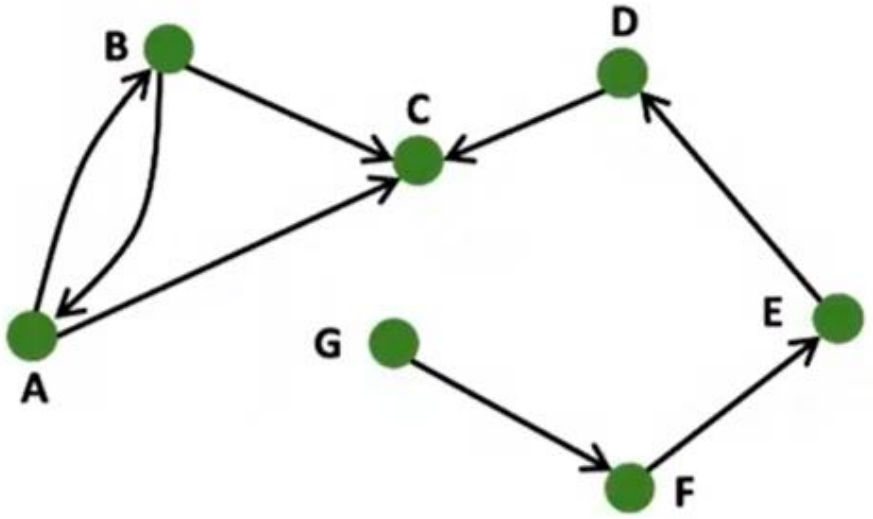
- **Nodes (Vertices):** The fundamental units or points in a graph. They can represent entities like people, cities, or other objects.
- **Edges (Links/Arcs):** The connections between nodes. They represent relationships or pathways between the nodes.

Directed vs Undirected Graphs

An **undirected graph** is a graph where the edges have no direction.

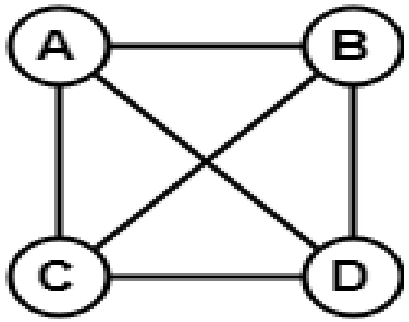


A **directed graph** is a graph where the edges have a direction.

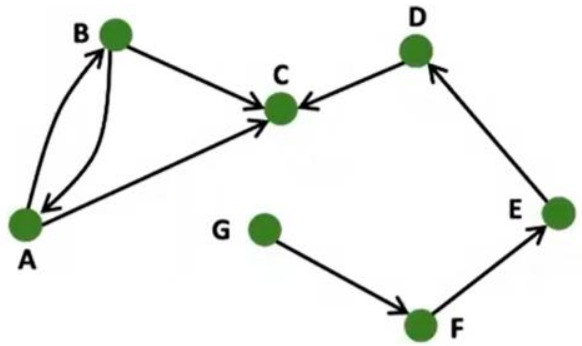


Node Degrees(K_i)

- In **undirected network** The number of edges adjacent to node i.
- $K_A = K_B = K_C = K_D = 3$

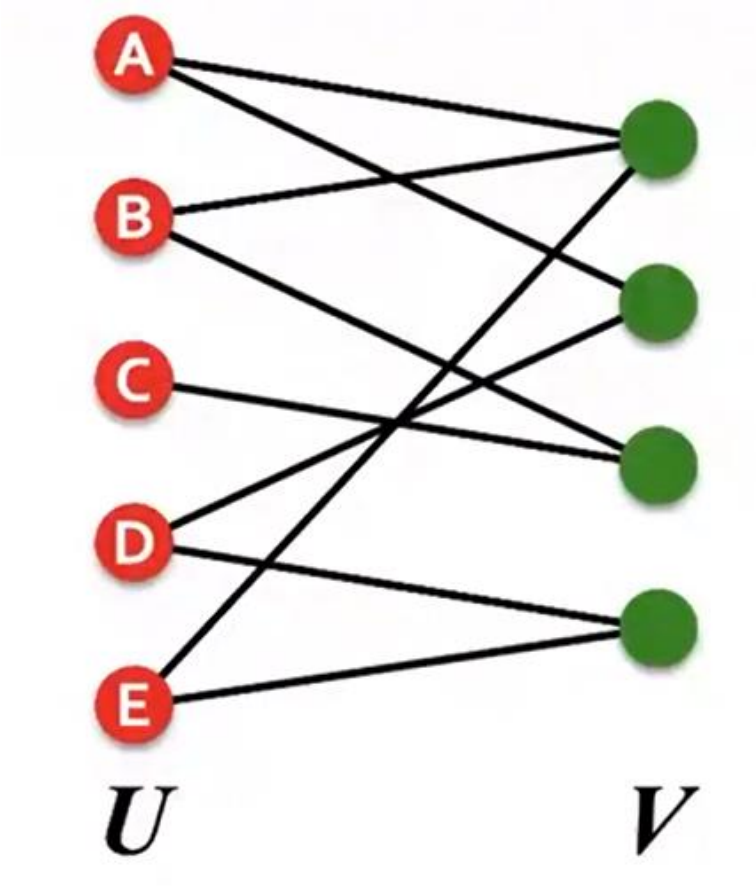


- In **directed network** we define the in-degree and out-degree .
- The total degree of a node is the sum of in and out degrees.
- $K_C^{in} = 3, \quad K_c^{out} = 0, \quad K_c = 3$



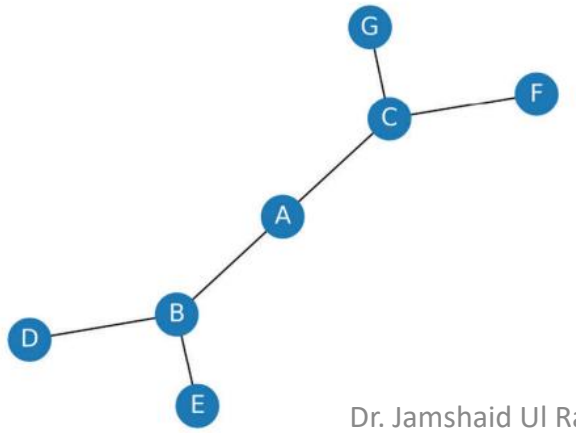
Bipartite Graph

- **Biparite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every link connect a node in U to one in V; that is U and V are independent sets



Adjacency Matrix

- An **adjacency matrix** is a matrix that represents the edges in a graph, where each cell indicates whether there is an edge between two nodes.
- The matrix is a square matrix of size $n * n$, where n is the number of nodes in the graph.
- A value of 1 in the cell (i, j) indicates that there is an edge between node i and j node , while a value of 0 indicates that there is no edge.

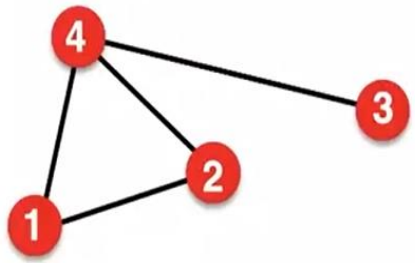


$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Adjacency Matrix

For an undirected graph, the matrix is symmetric, while for directed graph, the matrix is not necessarily symmetric.

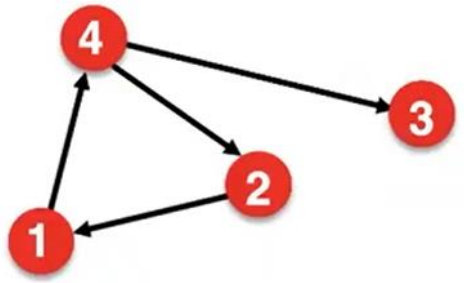
- ***Undirected:***



$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

$$A_{ij} = A_{ji}$$
$$A_{ii} = 0$$

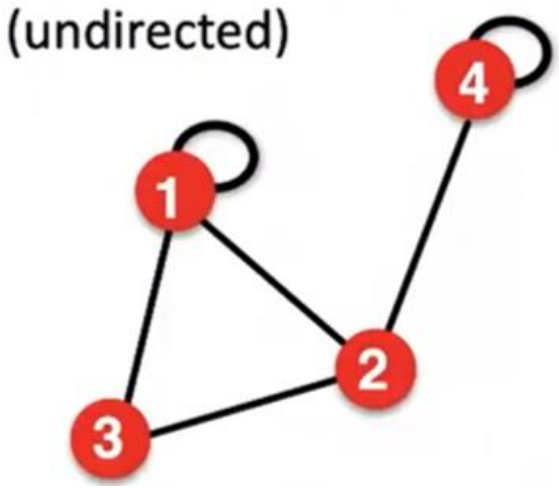
- ***Directed:***



$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

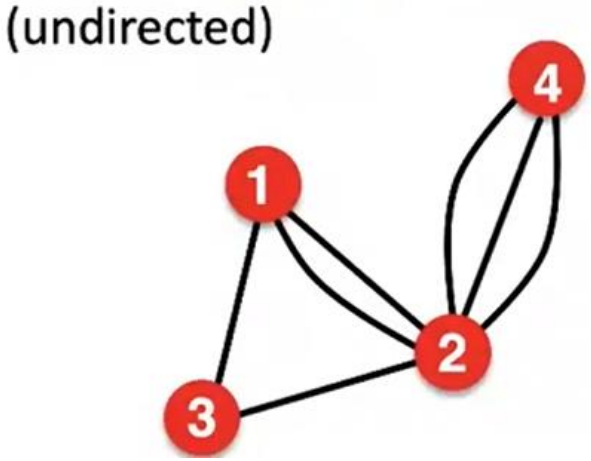
$$A_{ij} \neq A_{ji}$$
$$A_{ii} = 0$$

- A **self-loop** in a graph is an edge that connects a node to itself.



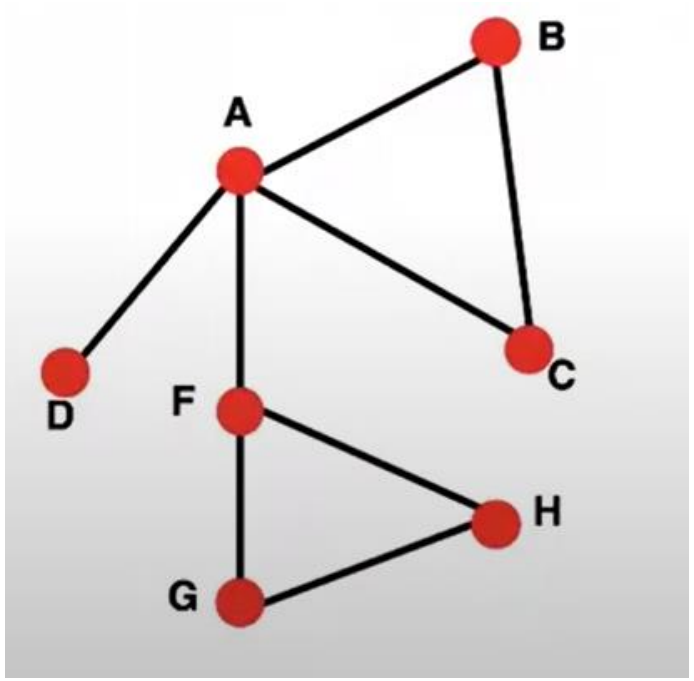
- $A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$

- A **multigraph** is a type of graph in which multiple edges (also known as parallel edges) between the same pair of nodes are allowed.

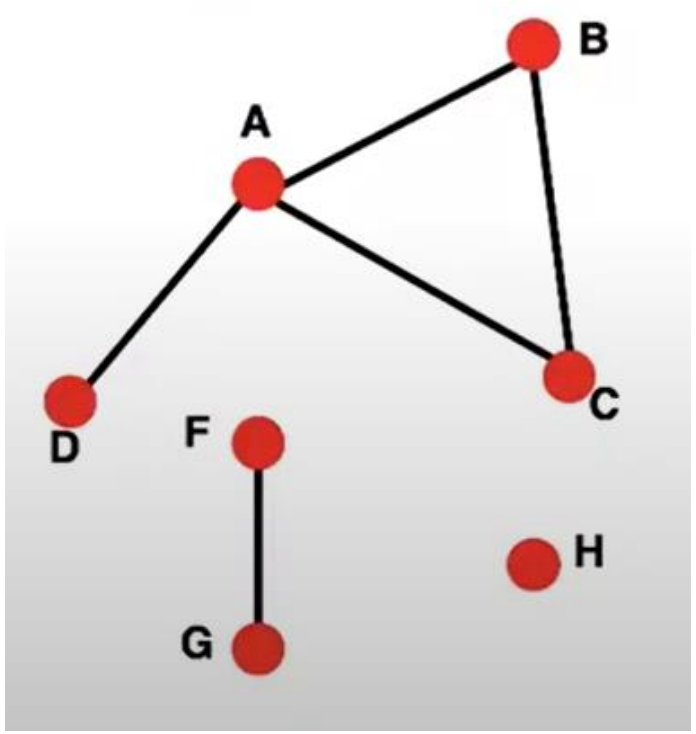


- $\mathrm{B} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$

- A **connected graph** is a graph in which there is a path between every pair of nodes.

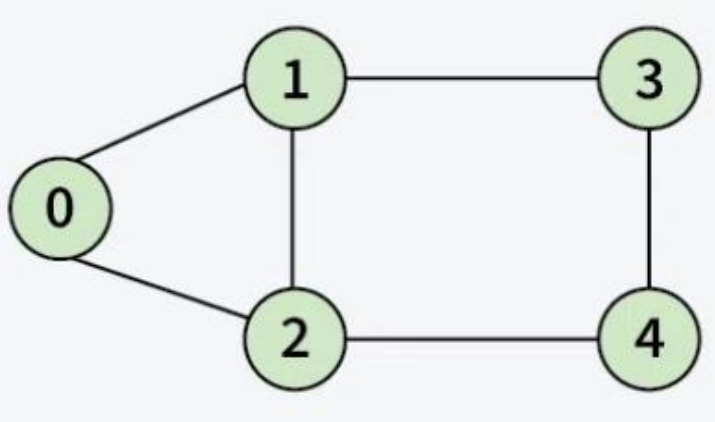


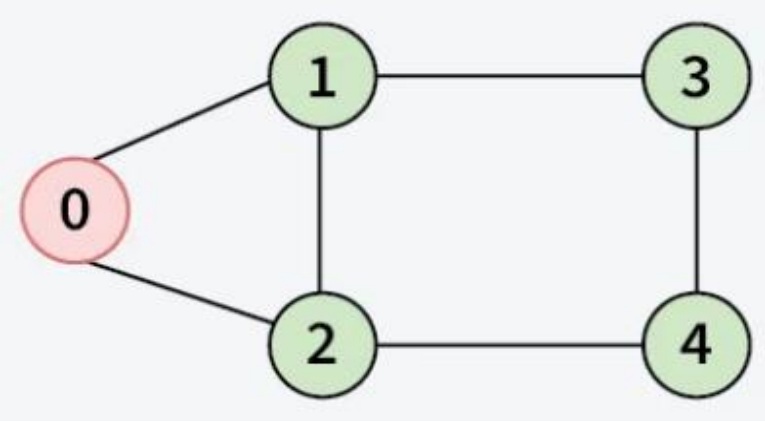
- A **disconnected graph** is a graph that is not connected. In other words, there are at least two nodes in the graph that do not have a path connecting them.

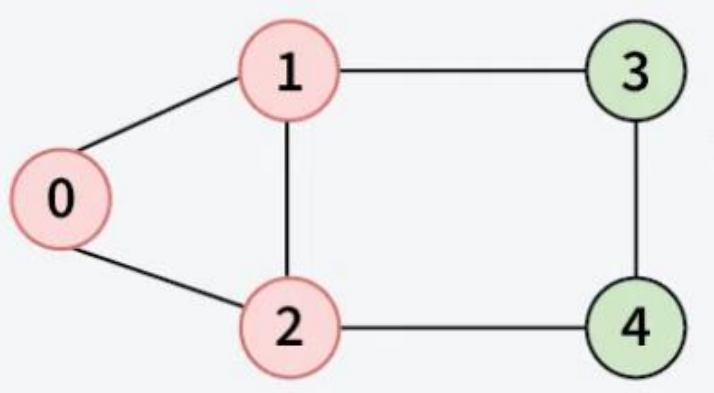


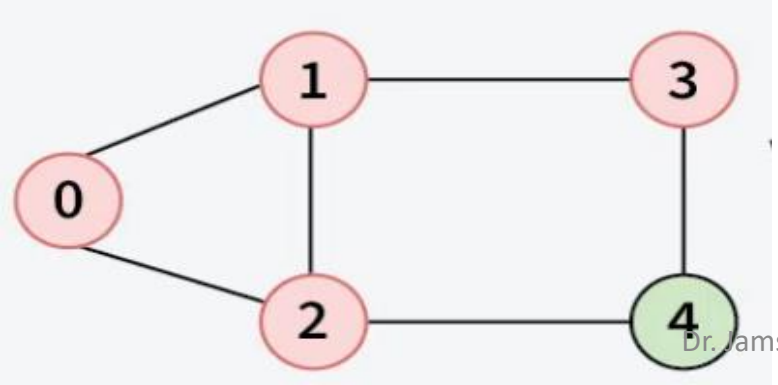
Breadth-First Search(BFS)

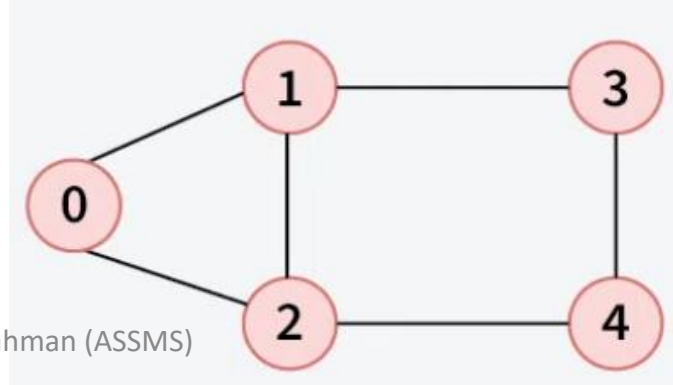
- **Breadth First Search (BFS)** is a fundamental **graph traversal algorithm.** It begins with a node, then first traverses all its adjacent. Once all adjacent are visited, then their adjacent are traversed.







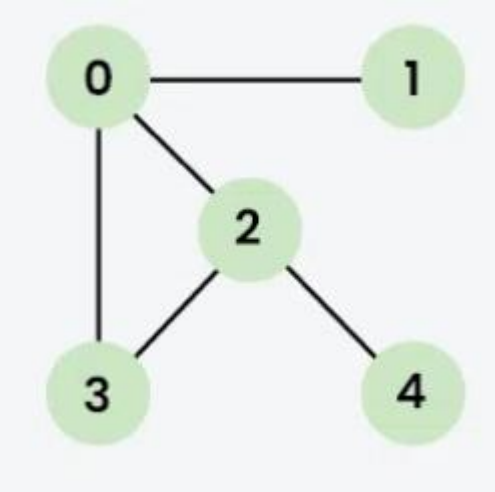


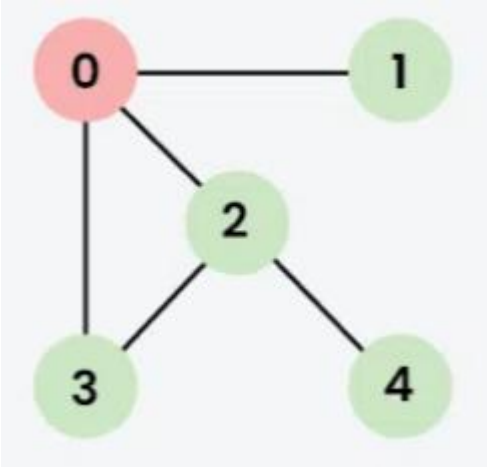


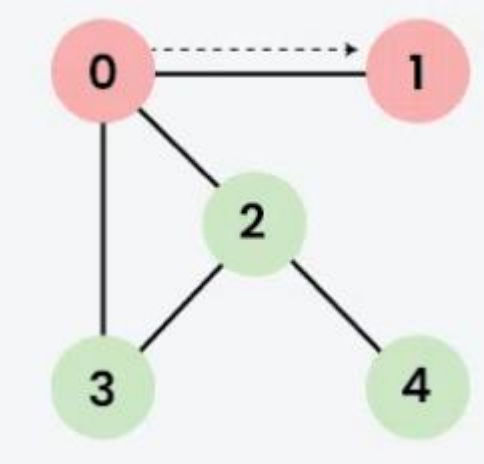
Depth-First Search(DFS)

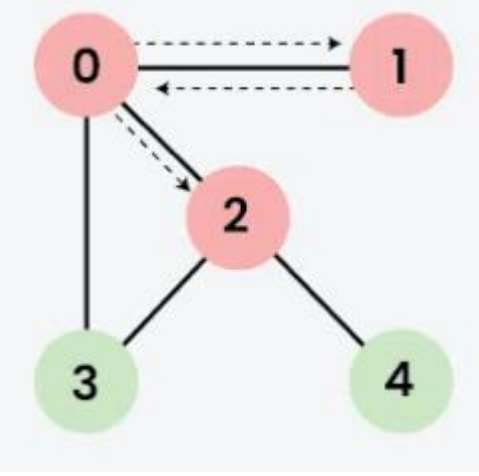
- **Depth First Search (or DFS)** for a graph we traverse all adjacent vertices one by one.
- When we traverse an adjacent vertex, we completely finish the traversal of all vertices reachable through that adjacent vertex.
- After we finish traversing one adjacent vertex and its reachable vertices, we move to the next adjacent vertex and repeat the process.

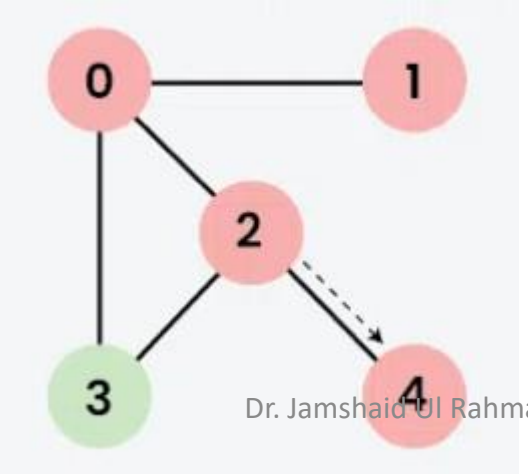
Depth-First Search(DFS)

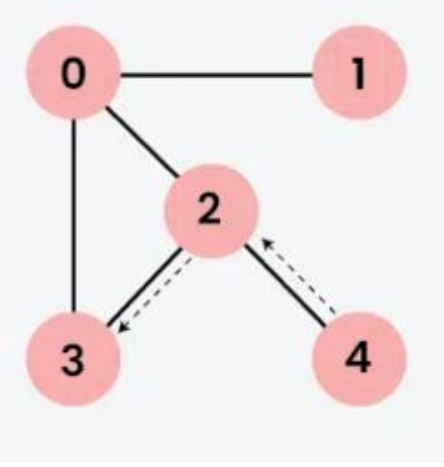












Graphs : Machine Learning

Traditional ML With Graph Pipeline

- Given input graph
- Extract nodes/links/graph level features
- Train an ML model
  1) Random forest
  2) Support Vector Machines (SVM)
  3) Neural Networks
- Apply the model: given a new node/link/graph, obtain its features and make a prediction.

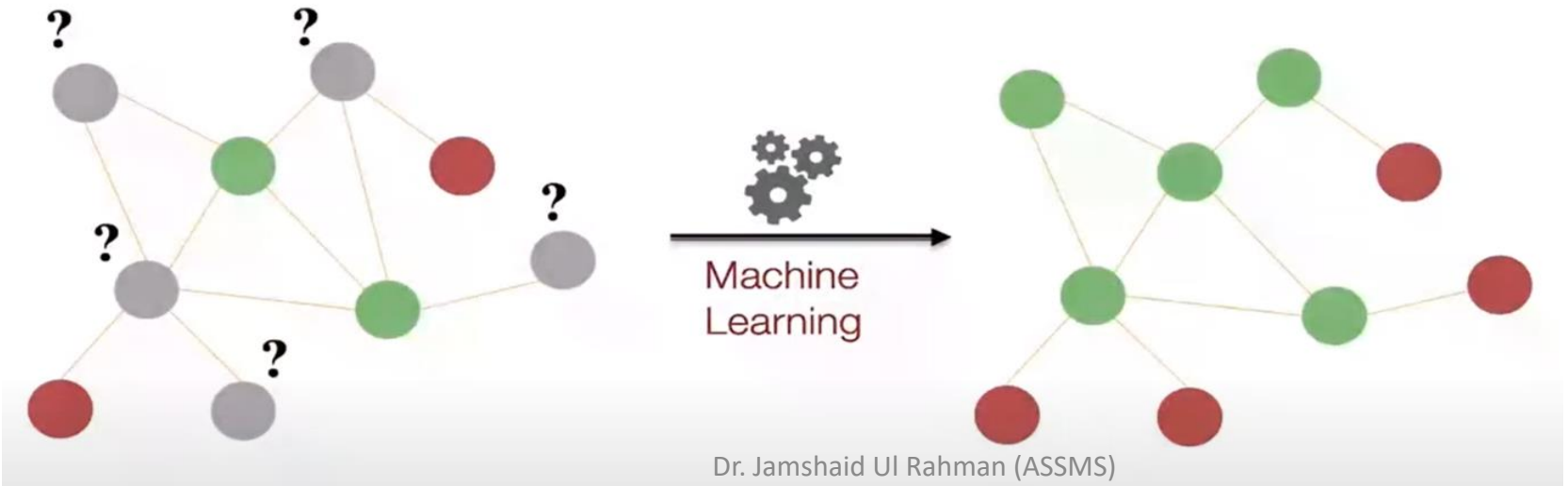
Graph Neural Network Usage

- **Node-Level Tasks**
- **Edge-Level Tasks**
- **Graph-Level Tasks**

Node Level Task and Features

Node-Level Tasks

- **Node Property Prediction:** Predicting properties or attributes of nodes, such as classifying nodes in a social network or predicting properties of molecules in a chemical graph.
- **Node Classification:** Assigning labels to nodes based on their features and connections.



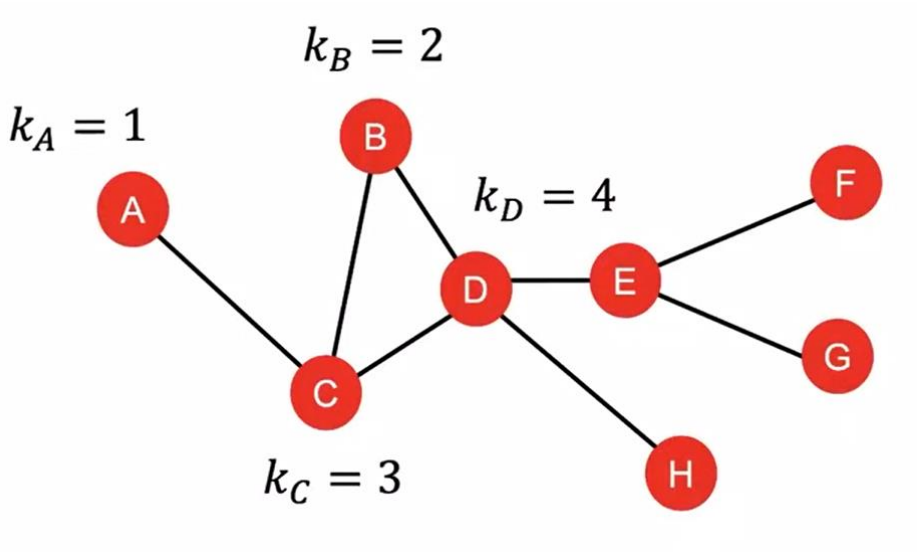
Node-Level Features

Characterize the structure and position of a node in the network:

1. **Node degree**
2. **Node centrality**
3. **Clustering Coefficient**
4. **Graphlets**

Node-Level Features

- **Node degree :**
- The degree k_v of node v is the number of edges the node has



Node-Level Features

- **Node centrality:**
- **Node centrality** measures are used to determine the importance or influence of a node within a graph.
- These measures help to identify key nodes that play a crucial role in the structure and function of the network.
- Different ways to model importance :

  i) **Betweenness centrality**

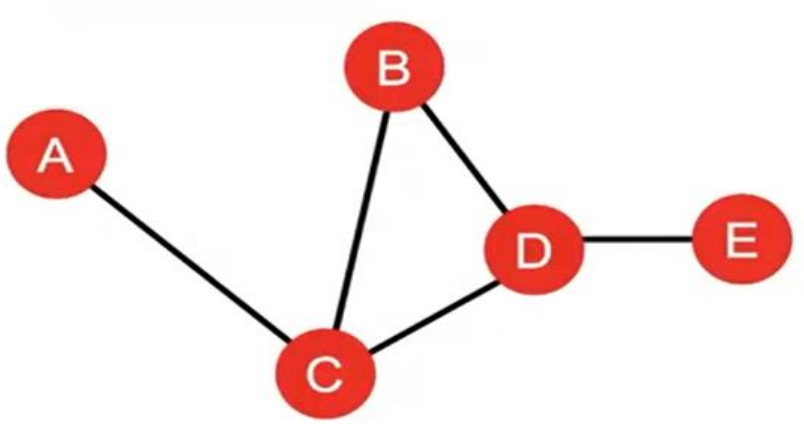
  ii) **Closeness centrality**

Node-Level Features

- **Betweenness centrality:**
- A node is important if it lies on many shortest paths between other nodes.
- $c_v = \sum_{s \neq v \neq t} \frac{shortest\ paths\ between\ s\ and\ t\ that\ contain\ v}{shortest\ paths\ between\ s\ and\ t}$

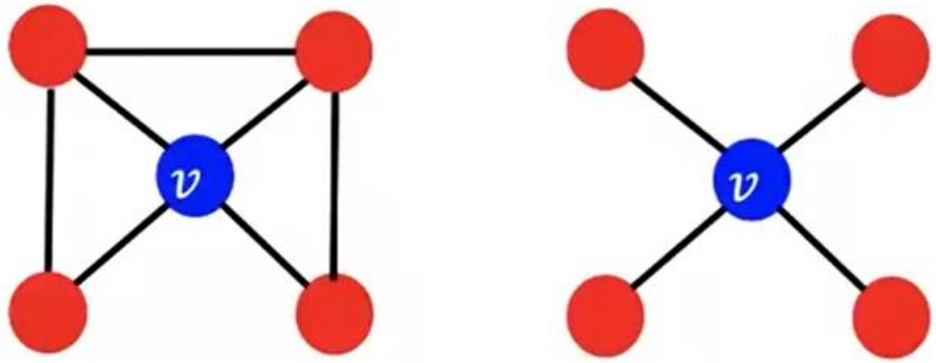
Node-Level Features

- **Closeness centrality:**
- A node is important if it has small shortest path length to all other nodes.
- $C_v = \frac{1}{\sum_{u \neq v} shortest\ path\ length\ between\ u\ and\ v}$
- Example:
- $(\boldsymbol{A-C-B}, \boldsymbol{A-C}, \boldsymbol{A-C-D}, \boldsymbol{A-C-D-E})$
- $C_A = \frac{1}{(2+1+2+3)} = 1/8$



Node-Level Features

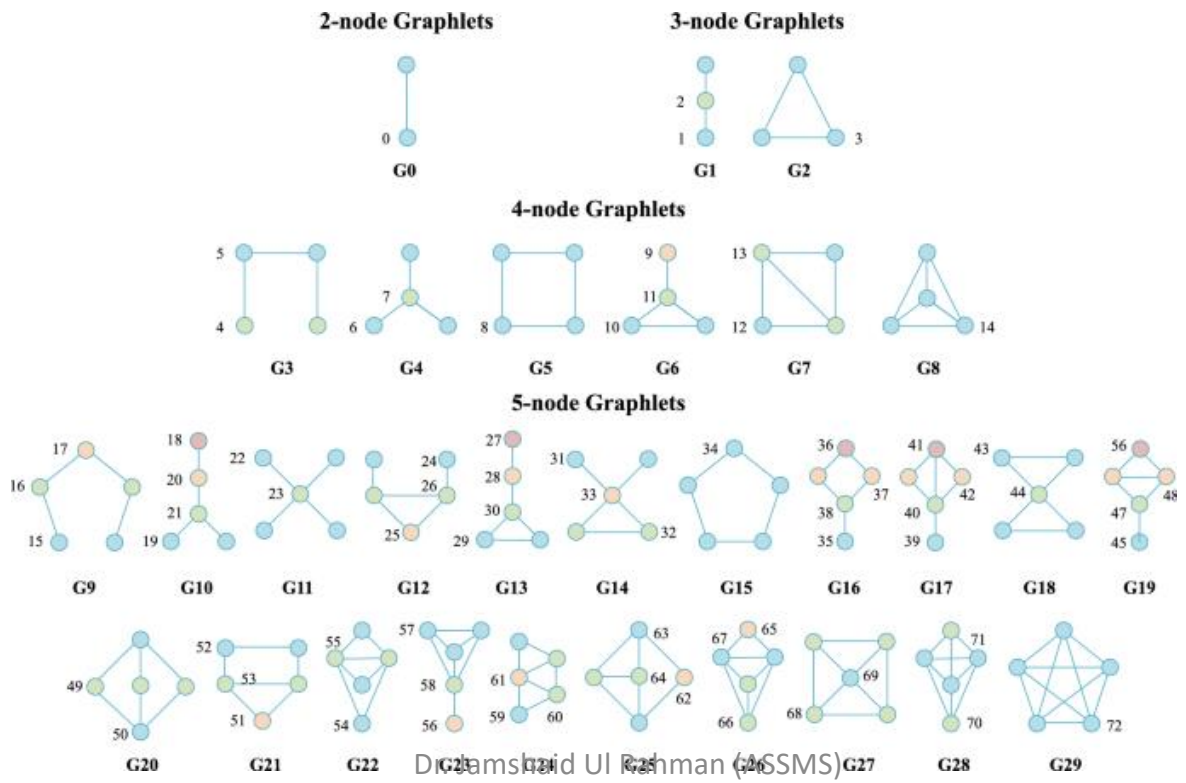
- The **clustering coefficient** of a node is the ratio of the number of edges between the nodes within its neighborhood to the number of possible edges between those nodes.
- $e_v = \frac{edges\ among\ neighboring\ nodes}{possible\ edges\ between\ thos\ nodes}$
- Example:



$e_v = 0.5$ $e_v = 0$

Node-Level Features

- **Graphlets** are small, induced subgraphs(connected non-isomorphic) of a larger network.
- Example:



Thank You!