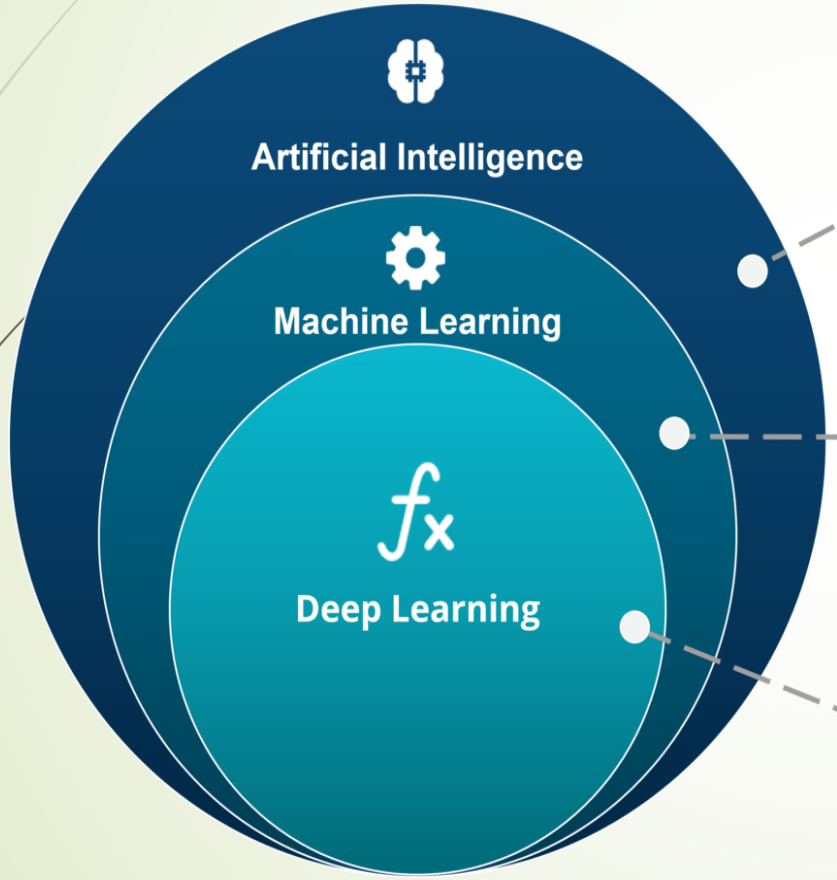
Introduction to Deep Learning

Lecture 1

What is Deep Learning?

- A subset of machine learning that uses neural networks with multiple layers to model complex patterns in data.
- Mimics the human brain's structure and function.

Umbrella Diagram



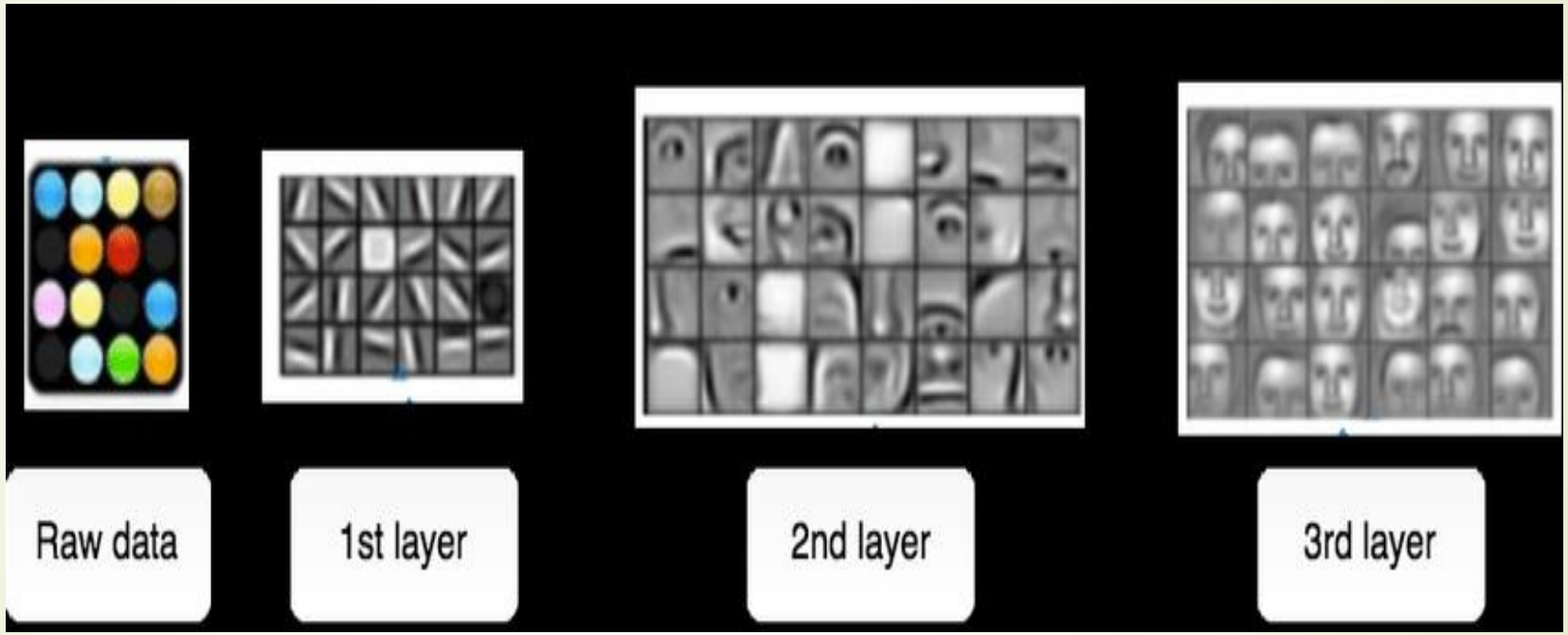
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

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

https://www.edureka.co/blog/ai-vs-machine-learning-vs-deep-learning/

Continue…..

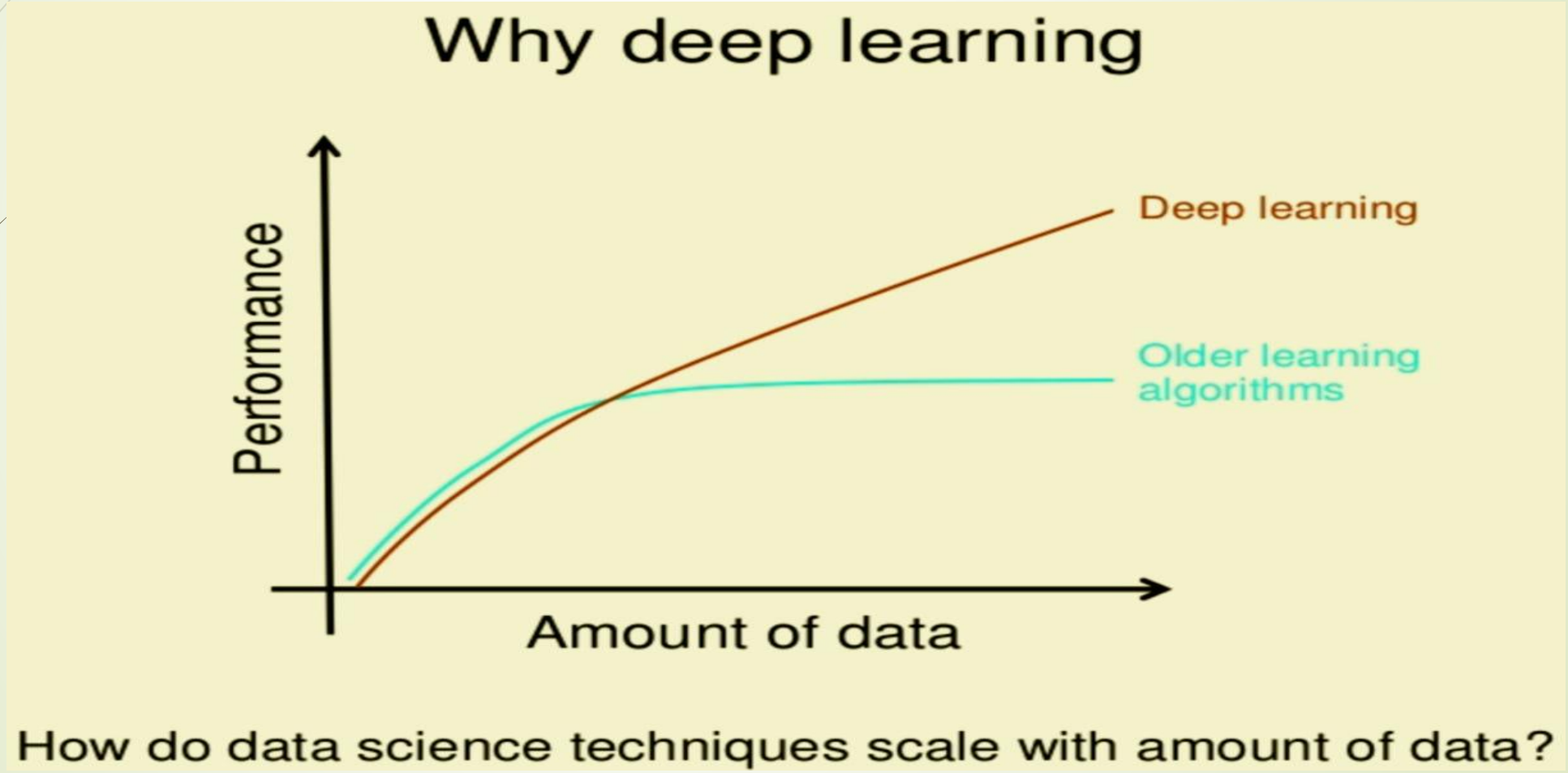


https://www.researchgate.net/publication/326412238_Deep_generative_neural_networks_for_novelty_generation_a_foundational_framework_metrics_and_experiments/figures?lo=1&utm_source=google&utm_medium=organic

Why Deep Learning

1. **Handles Complex Data**
2. **Automates Feature Engineering**
3. **State-of-the-Art Performance**
4. **Scalability with Big Data**
5. **Continuous Innovation**

Why Deep Learning



Why not Deep Learning

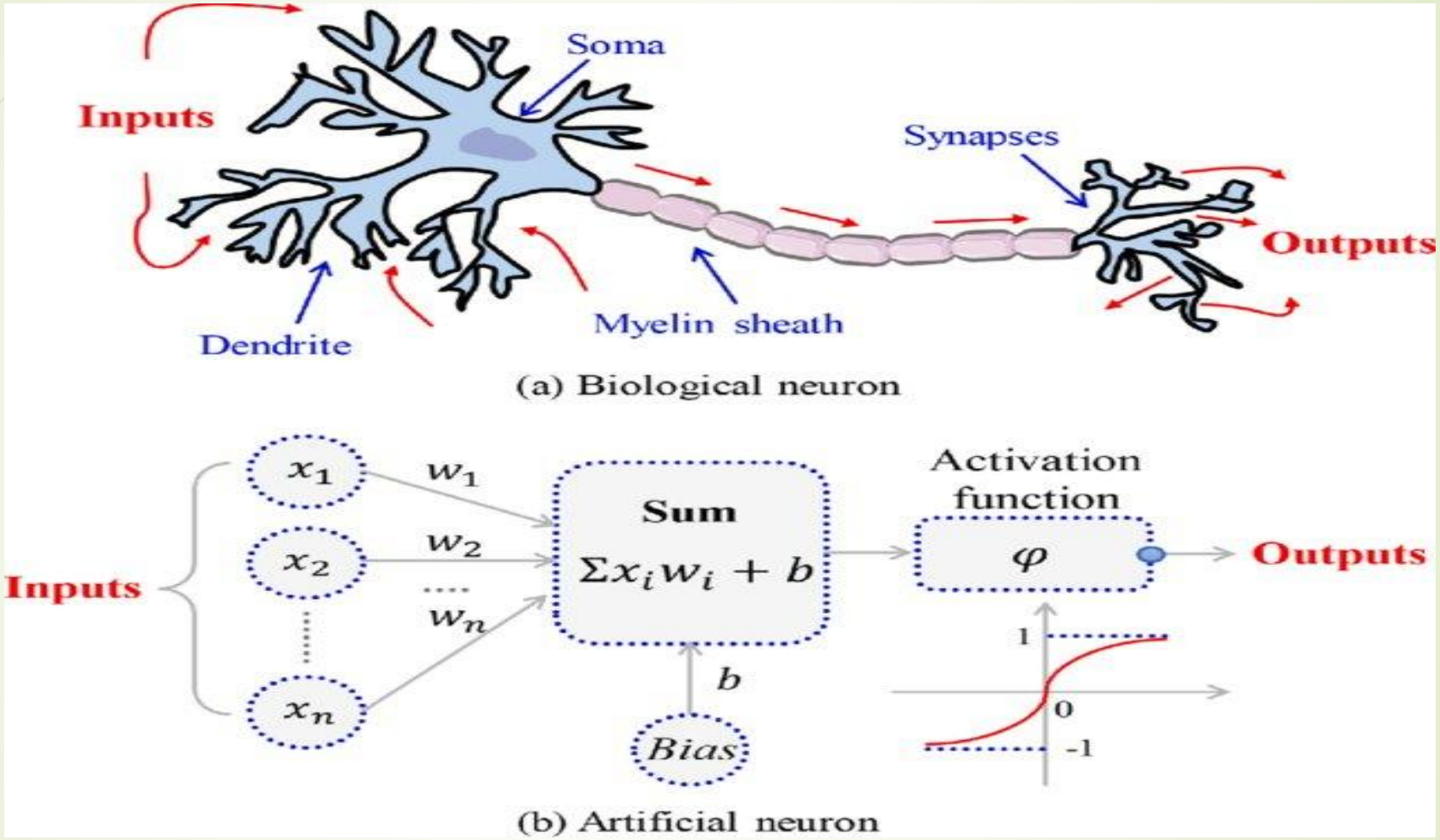
1. **Small datasets (deep learning requires large amounts of data).**
2. **Simple problems**
3. **Limited computational resources.**
4. **Need for interpretability (deep learning models are often "black boxes").**

Future of Deep Learning

- More efficient and interpretable models
- Deploying deep learning on edge devices (e.g., smartphones, IoT).
- Solving global challenges like climate change and healthcare.
- Combining with quantum computing, robotics, and blockchain.

Neural Networks

Biological Vs Artificial Neuron

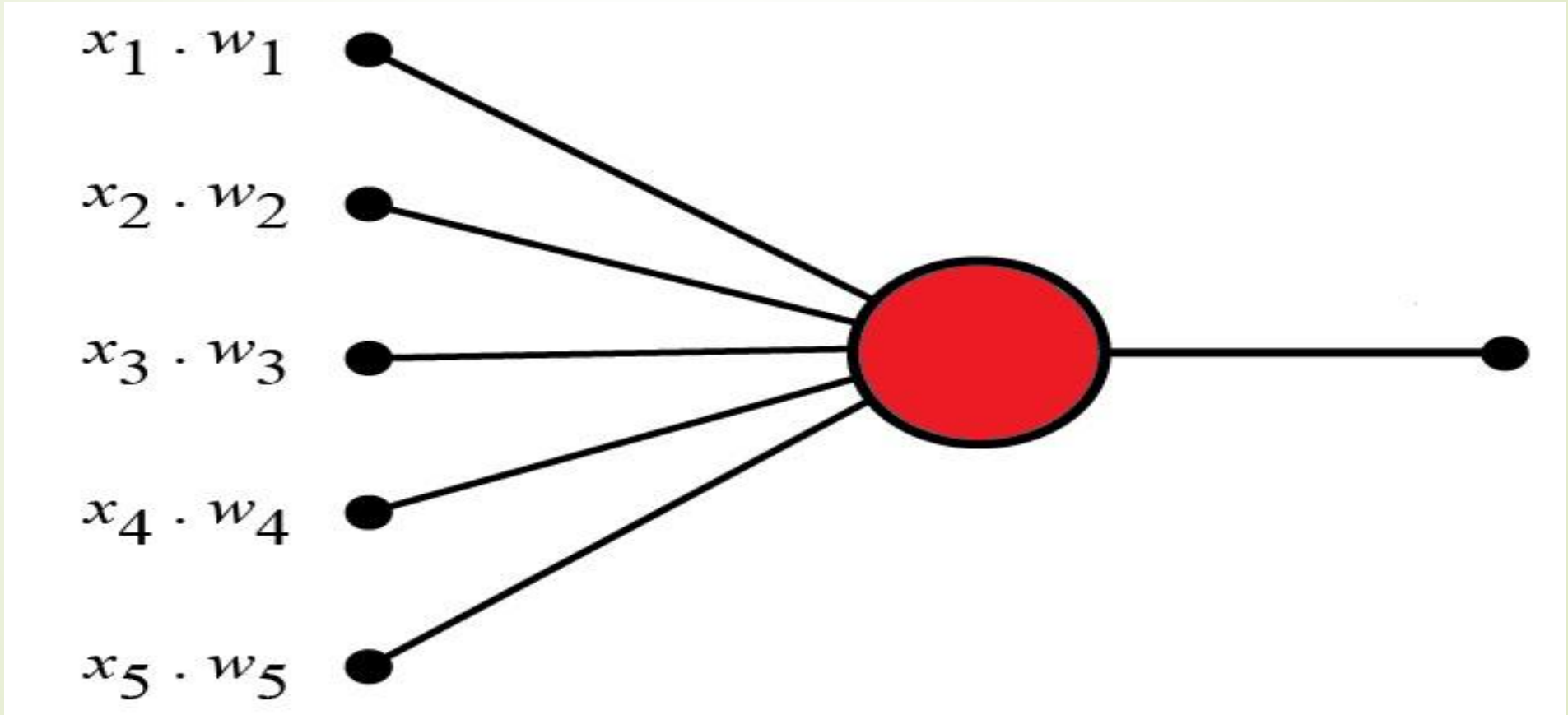


https://www.researchgate.net/publication/351372032_Bond_strength_prediction_of_concrete-encased_steel_structures_using_hybrid_machine_learning_method/figures

Perceptron

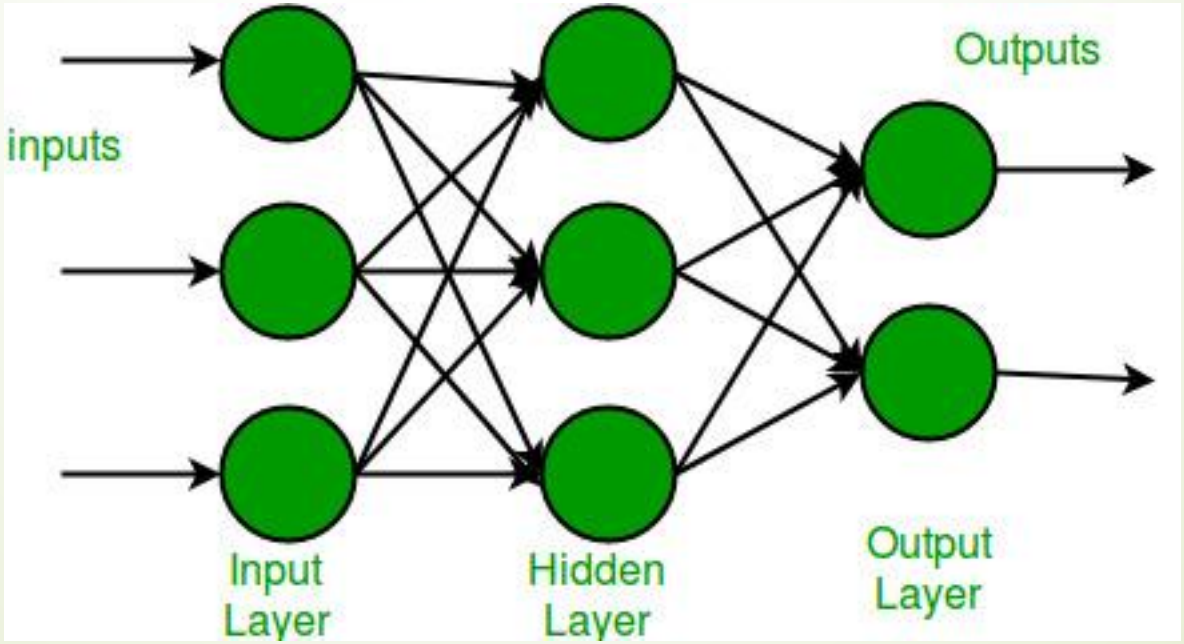
- **Perceptron was introduced by Frank Rosenblatt in 1957.**
- **A single-layer neural network is used for binary classification tasks.**
- **Input Layer: Receives input features (e.g., x1,x2,...,xn$x1,x2,...,xn$).**
- **Weights: Each input is multiplied by a weight (w1,w2,...,wn$w1,w2,...,wn$).**
- **Activation Function: A step function (e.g., Sigmoid function) to produce the output (0 or 1).**

Architecture of Perceptron

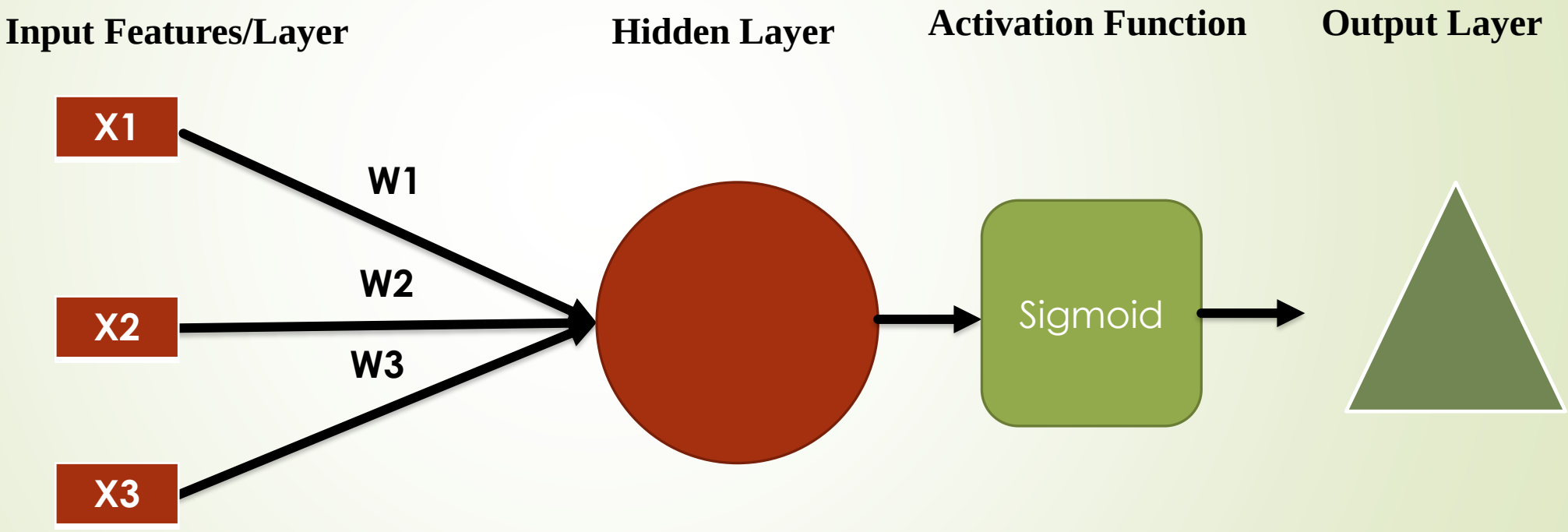


Multi-Layer Perceptron

- **Feedforward artificial neural network (ANN) with one or more hidden layers between the input and output layers.**
- **Extends the single-layer perceptron.**



How ANN Works



Y=w1x1+w2x2+w3x3….wnxn+bias

Example

- Let's say:
- x1 = 2, x2 = 3, x3 = 1
- w1 = 0.5, w2 = -0.2, w3 = 0.8
- b = 0.1
- Activation function: Sigmoid

1. **Weighted Sum:** sum = (2 * 0.5) + (3 * -0.2) + (1 * 0.8) + 0.1 = 1 - 0.6 + 0.8 + 0.1 = 1.3
2. **Activation:** y = 1 / (1 + exp(-1.3)) ≈ 0.785

- So, the neuron's output for these inputs would be approximately 0.785.

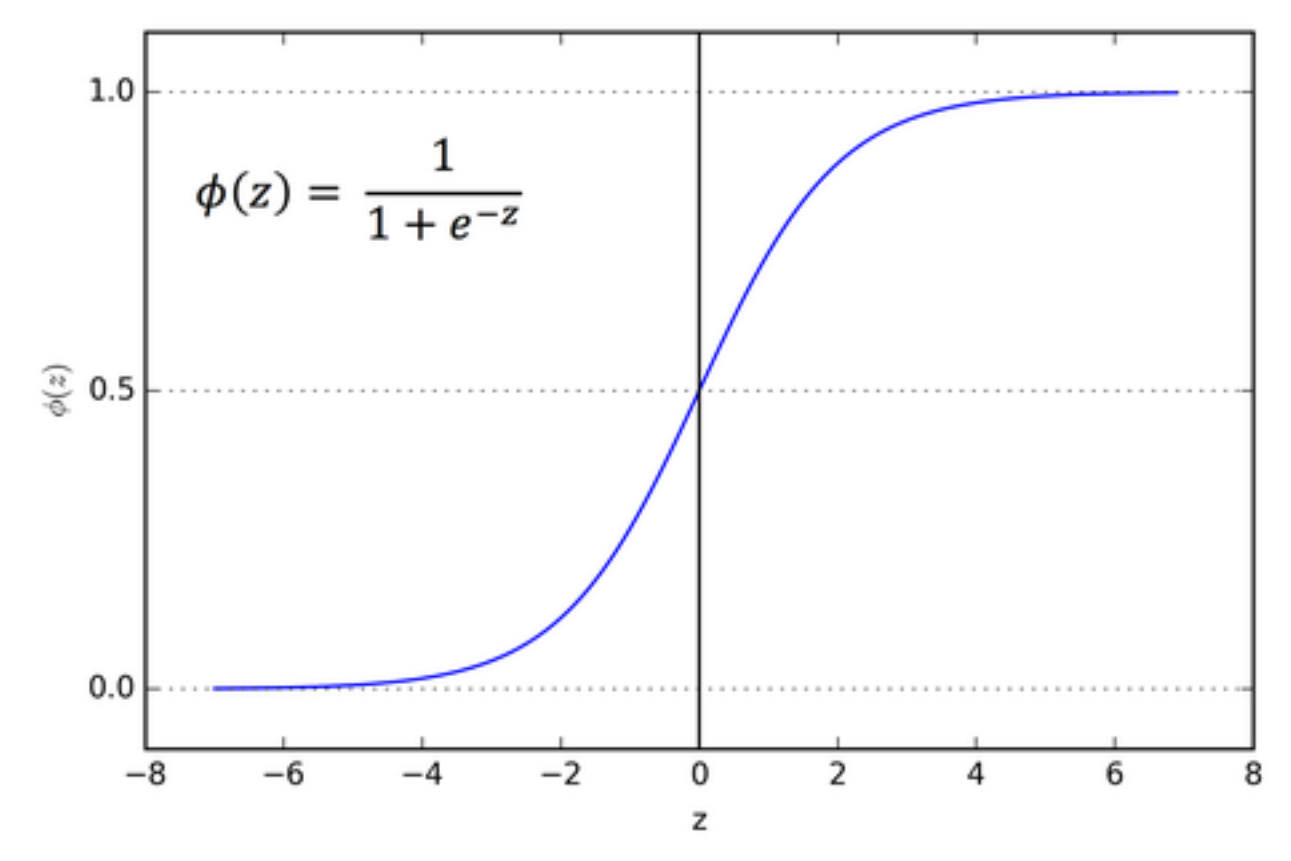
Activation Functions

1. Sigmoid
2. Relu

Activation Functions (Sigmoid)

- **Formula: f(x) = 1 / (1 + exp(-x))**
- **Output: Between 0 and 1.**
- **Use cases: Historically popular, but less used in deep learning now due to vanishing gradient issues. Still sometimes used in output layers for binary classification.**

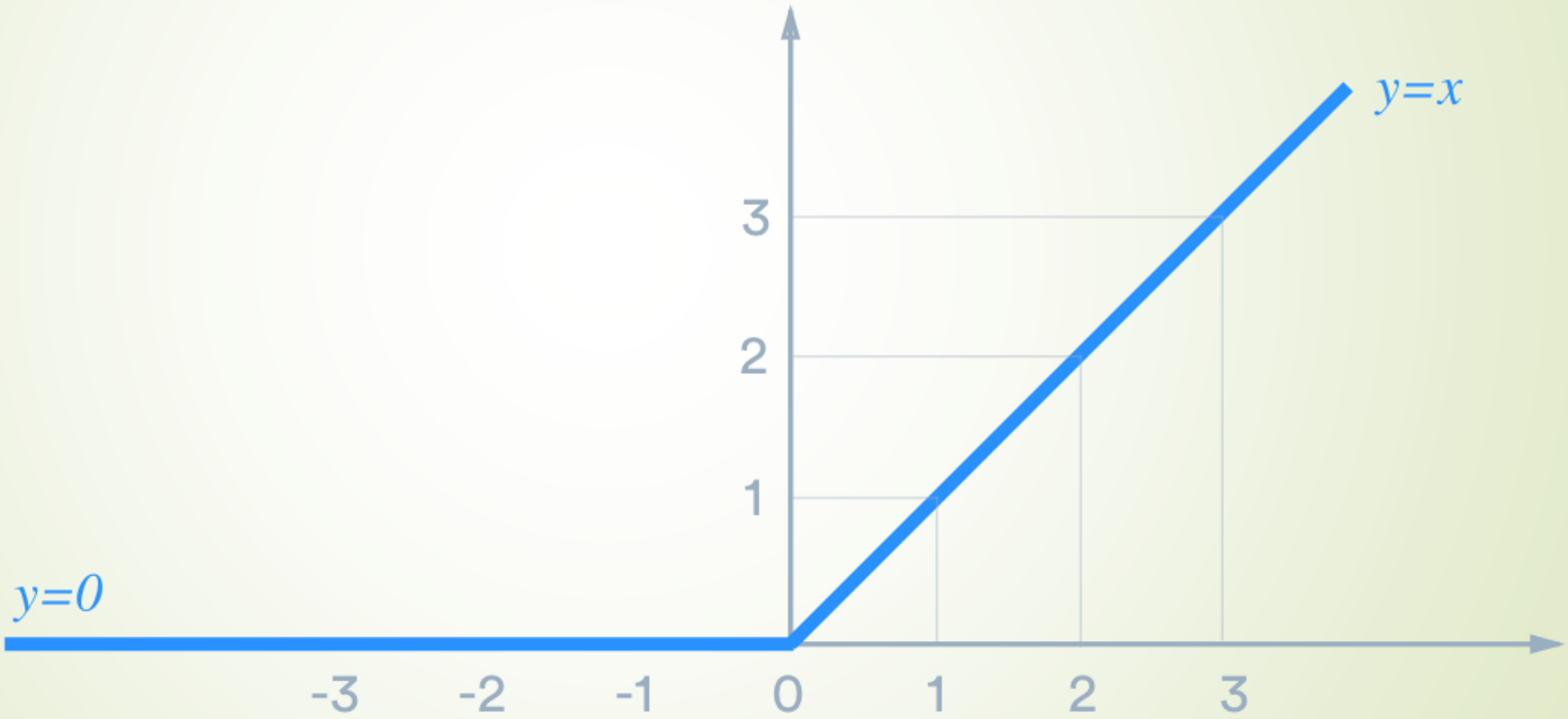
Activation Functions (Sigmoid)



Relu (Rectified Linear Unit)

- **Formula: f(x) = max(0, x)**
- **Output: 0 for negative input, x for positive input.**
- **Use cases: Very popular in deep learning. It helps to solve the vanishing gradient problem and often leads to faster training.**

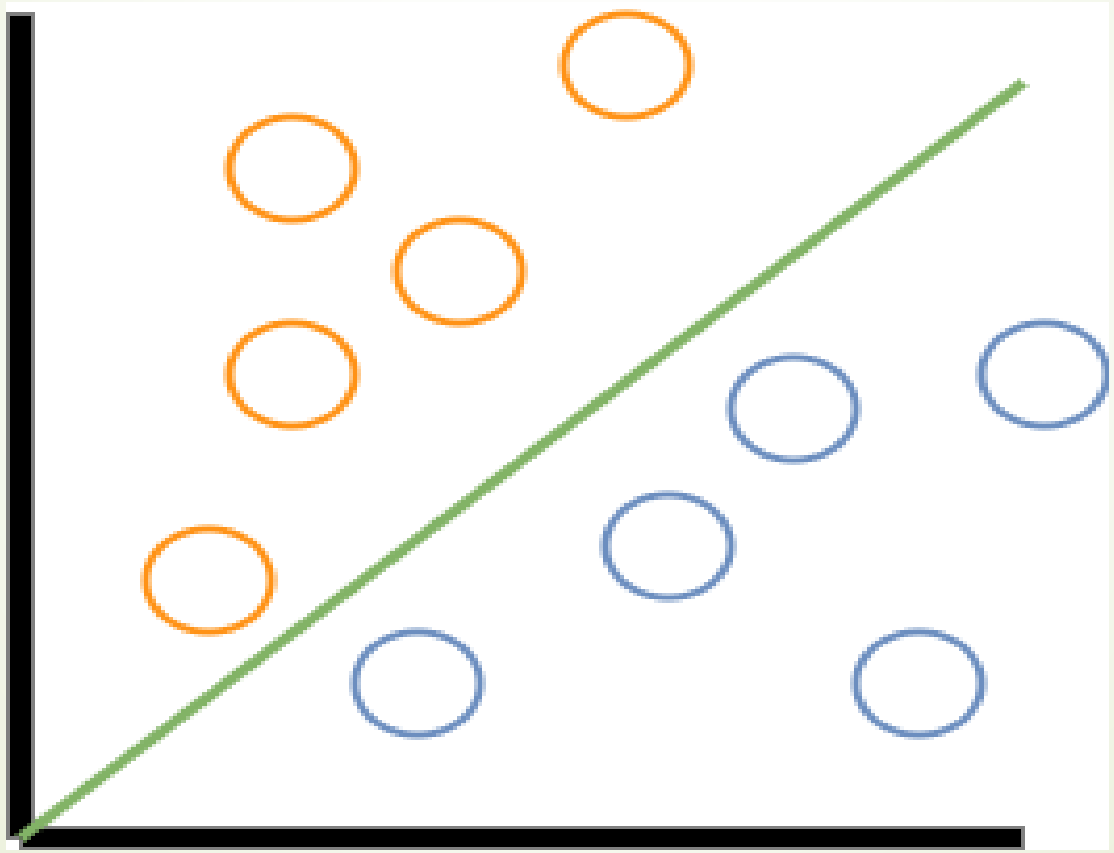
Relu (Rectified Linear Unit)



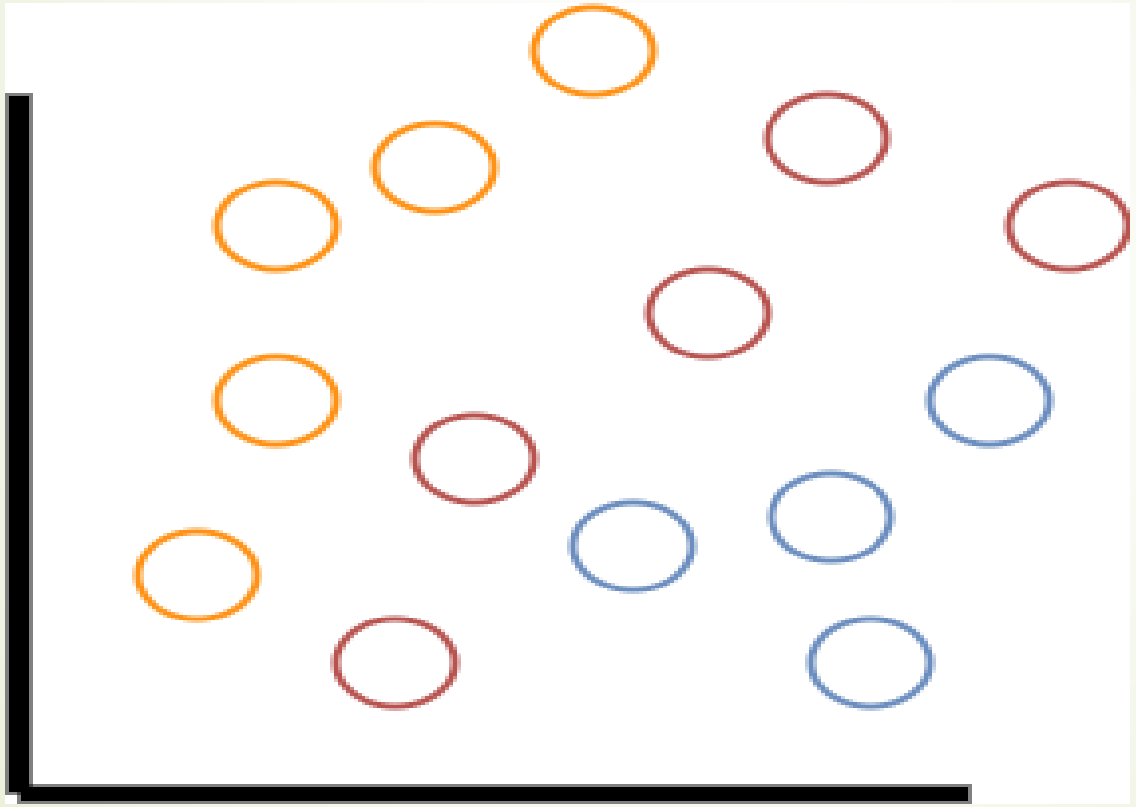
Why Activation Functions

- Real-world data is rarely linearly separable
- Introduce Non-linearity
- No matter how many layers you have, a network without activation functions is equivalent to a single-layer perceptron, severely limiting its capacity
- They decide whether the neuron should be "activated" or not.

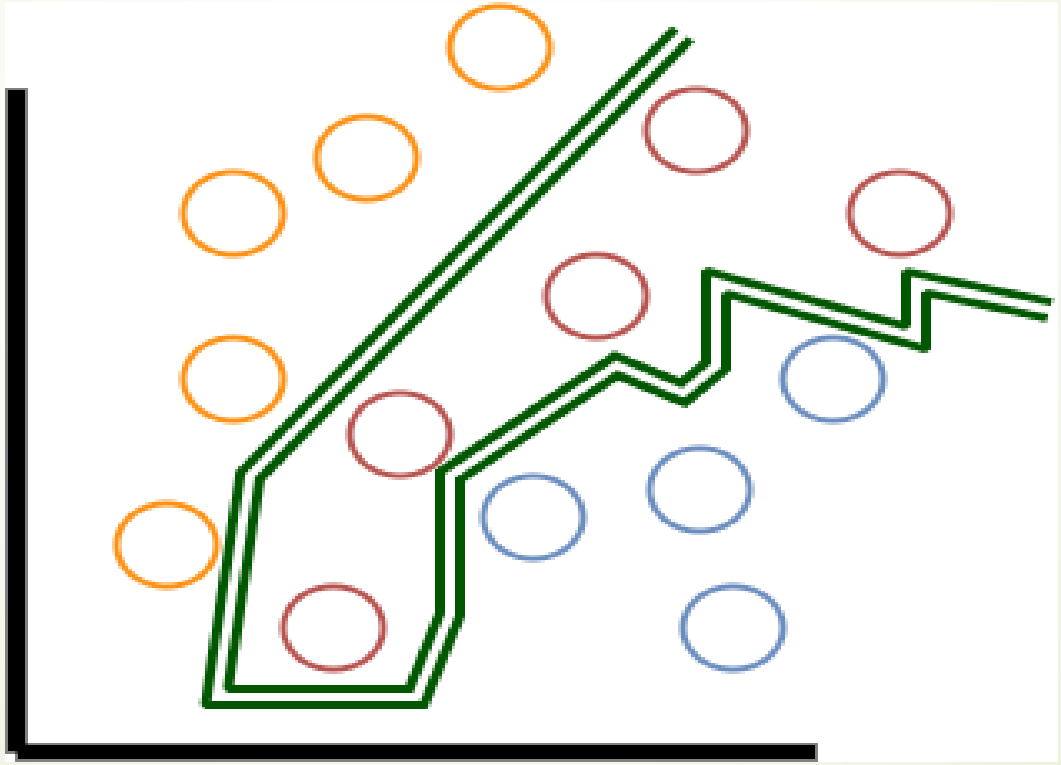
Why Activation Functions



Why Activation Functions



Why Activation Functions



Class Participation

Determine whether the neuron will activate or not. If the weighted sum is greater than or equal to 0.60, then output "Neuron activated" or 1; otherwise, output "Not activated" or 0.

- x1 = 5, x2 = 2, x3 = 7
- w1 = 0.25, w2 = 0.10, w3 = 0.50
- b = 0.15

➨ **Note: Use the Sigmoid activation function.**