Deep Learning

BCSE-332L

Module 1:

Introduction to Neural and Deep Neural Networks

Dr . Saurabh Agrawal

Faculty Id: 20165

School of Computer Science and Engineering

VIT, Vellore-632014

Tamil Nadu, India

Outline

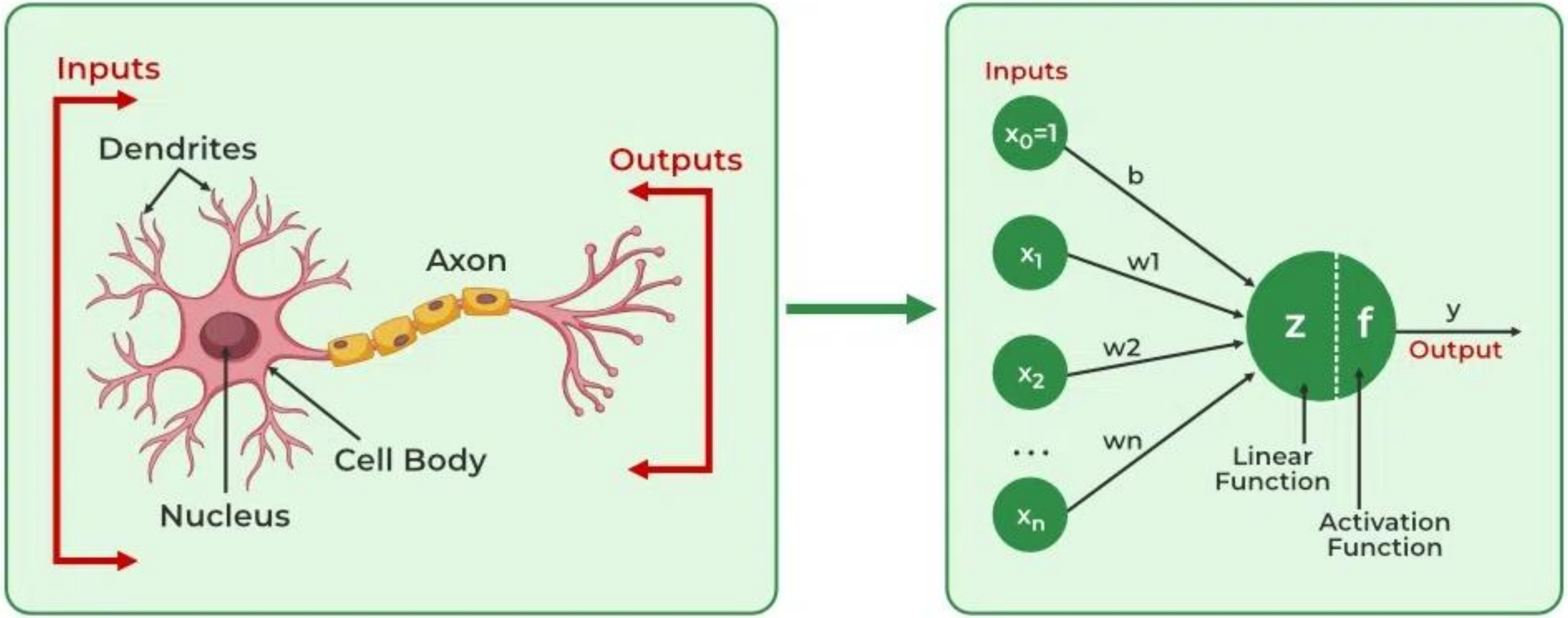
- ❑Neural Networks Basics
- ❑Functions in Neural networks:
- ❑Activation function
- ❑Loss function
- ❑Function approximation
- ❑Classification and Clustering problems
- ❑Deep networks basics
- ❑Shallow neural networks
- ❑Activation Functions
- ❑Gradient Descent
- ❑Back Propagation
- ❑Deep Neural Networks
- ❑Forward and Back Propagation
- ❑Parameters
- ❑Hyperparameters

Neural Networks Basics

- Neural Networks are computational models that mimic the complex functions of the human brain.
- The neural networks consist of interconnected nodes or neurons that process and learn from data, enabling tasks such as pattern recognition and decision making in machine learning.
- Neural networks extract identifying features from data, lacking pre-programmed understanding.
- Network components include neurons, connections, weights, biases, propagation functions, and a learning rule.
- Neurons receive inputs, governed by thresholds and activation functions.
- Connections involve weights and biases regulating information transfer.
- Learning, adjusting weights and biases, occurs in three stages: **input computation**, **output generation**, and **iterative refinement** enhancing the network's proficiency in diverse tasks.

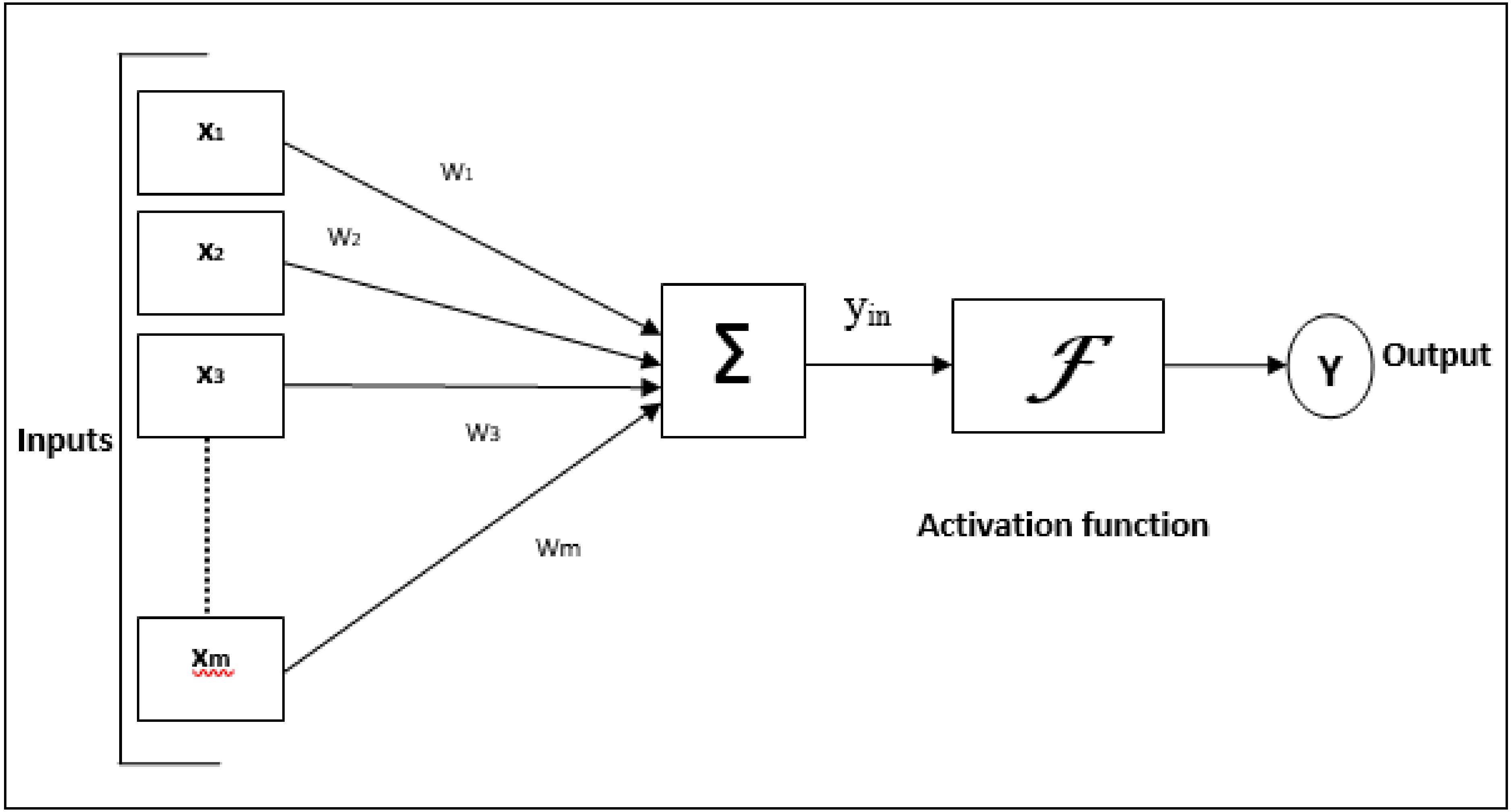
Neural Networks Basics

1. The neural network is simulated by a new environment.
2. Then the free parameters of the neural network are changed as a result of this simulation.
3. The neural network then responds in a new way to the environment because of the changes in its free parameters.



Neural Networks Basics

- The following diagram represents the general model of ANN followed by its processing.



Neural Networks Basics

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

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

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 $netinputcalculated$

Neural Networks Basics

❑Importance of Neural Networks

- ❑The ability of neural networks to identify patterns, solve intricate puzzles, and adjust to changing surroundings is essential.
- ❑Their capacity to learn from data has far-reaching effects, ranging from revolutionizing technology like natural language processing and self-driving automobiles to automating decision-making processes and increasing efficiency in numerous industries.
- ❑The development of artificial intelligence is largely dependent on neural networks, which also drive innovation and influence the direction of technology.

Neural Networks Basics

- **How does Neural Networks work?**
- Consider a neural network for email classification.
- The input layer takes features like email content, sender information, and subject.
- These inputs, multiplied by adjusted weights, pass through hidden layers.
- The network, through training, learns to recognize patterns indicating whether an email is spam or not.
- The output layer, with a binary activation function, predicts whether the email is spam (1) or not (0).
- As the network iteratively refines its weights through backpropagation, it becomes adept at distinguishing between spam and legitimate emails, showcasing the practicality of neural networks in real-world applications like email filtering.

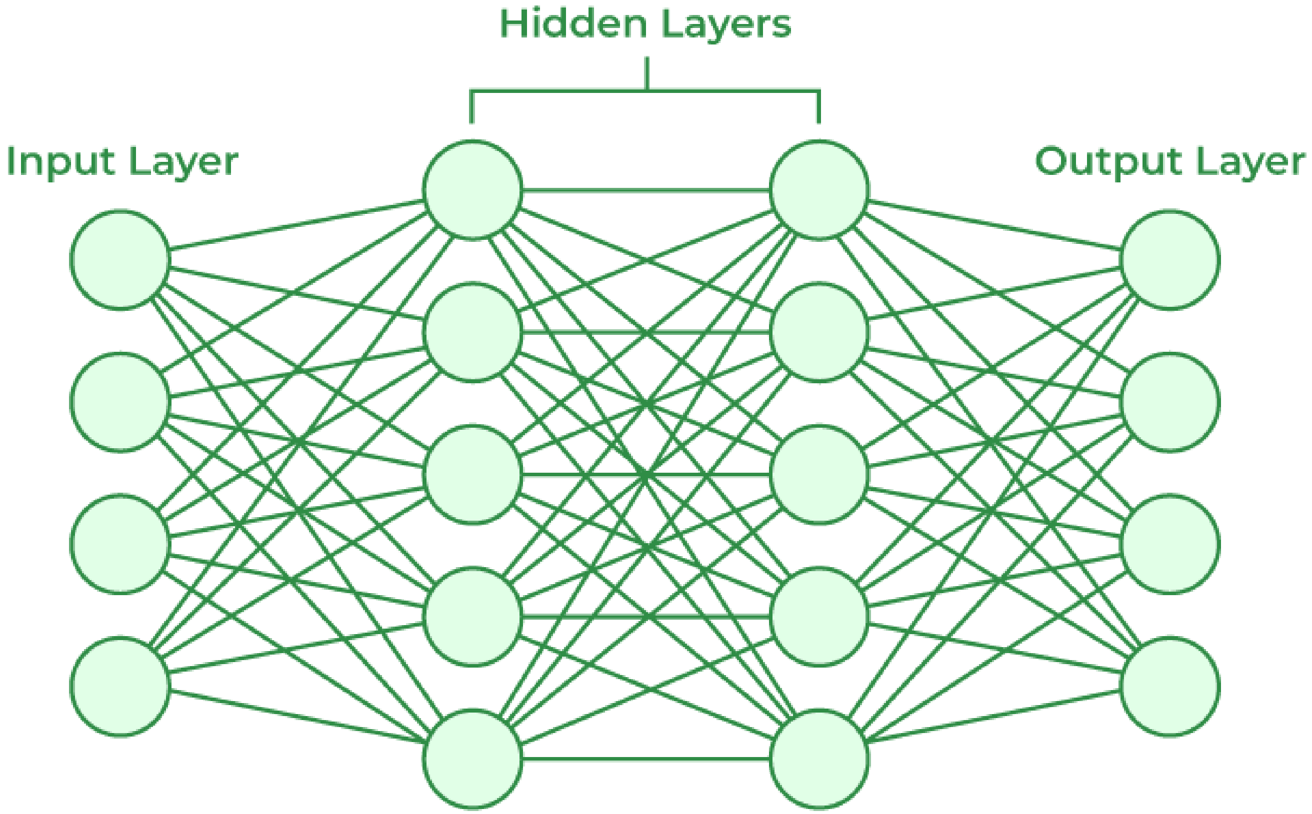
Neural Networks Basics

❑How does Neural Networks work?

❑Neural networks are complex systems that mimic some features of the functioning of the human brain.

❑It is composed of an input layer, one or more hidden layers, and an output layer made up of layers of artificial neurons that are coupled.

❑The two stages of the basic process are called backpropagation and forward propagation.



Neural Networks Basics

❑How does Neural Networks work?: Forward Propagation

1. **Input Layer:** Each feature in the input layer is represented by a node on the network, which receives input data.
2. **Weights and Connections:** The weight of each neuronal connection indicates how strong the connection is. Throughout training, these weights are changed.
3. **Hidden Layers:** Each hidden layer neuron processes inputs by multiplying them by weights, adding them up, and then passing them through an activation function. By doing this, non-linearity is introduced, enabling the network to recognize intricate patterns.
4. **Output:** The final result is produced by repeating the process until the output layer is reached.

Neural Networks Basics

❑How does Neural Networks work?: Backpropagation

1. **Loss Calculation:** The network's output is evaluated against the real goal values, and a loss function is used to compute the difference. For a regression problem, the Mean Squared Error (MSE) is commonly used as the cost function.

   Loss Function: $MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$

2. **Gradient Descent:** Gradient descent is then used by the network to reduce the loss. To lower the inaccuracy, weights are changed based on the derivative of the loss with respect to each weight.
3. **Adjusting weights:** The weights are adjusted at each connection by applying this iterative process, or backpropagation, backward across the network.
4. **Training:** During training with different data samples, the entire process of forward propagation, loss calculation, and backpropagation is done iteratively, enabling the network to adapt and learn patterns from the data.
5. **Activation Functions:** Model non-linearity is introduced by activation functions like the rectified linear unit (ReLU) or sigmoid. Their decision on whether to "fire" a neuron is based on the whole weighted input.

Neural Networks Basics

❑ Types of Neural Networks:

1. **Feedforward Networks:** is a simple artificial neural network architecture in which data moves from input to output in a single direction. It has input, hidden, and output layers; feedback loops are absent. Its straightforward architecture makes it appropriate for a number of applications, such as regression and pattern recognition.
2. **Multilayer Perceptron (MLP):** is a type of feedforward neural network with three or more layers, including an input layer, one or more hidden layers, and an output layer. It uses nonlinear activation functions.
3. **Convolutional Neural Network (CNN):** is a specialized artificial neural network designed for image processing. It employs convolutional layers to automatically learn hierarchical features from input images, enabling effective image recognition and classification. CNNs have revolutionized computer vision and are pivotal in tasks like object detection and image analysis.

Neural Networks Basics

❑Types of Neural Networks:

4. **Recurrent Neural Network (RNN):** An artificial neural network type intended for sequential data processing is called a Recurrent Neural Network (RNN). It is appropriate for applications where contextual dependencies are critical, such as time series prediction and natural language processing, since it makes use of feedback loops, which enable information to survive within the network.
5. **Long Short-Term Memory (LSTM):** LSTM is a type of RNN that is designed to overcome the vanishing gradient problem in training RNNs. It uses memory cells and gates to selectively read, write, and erase information.

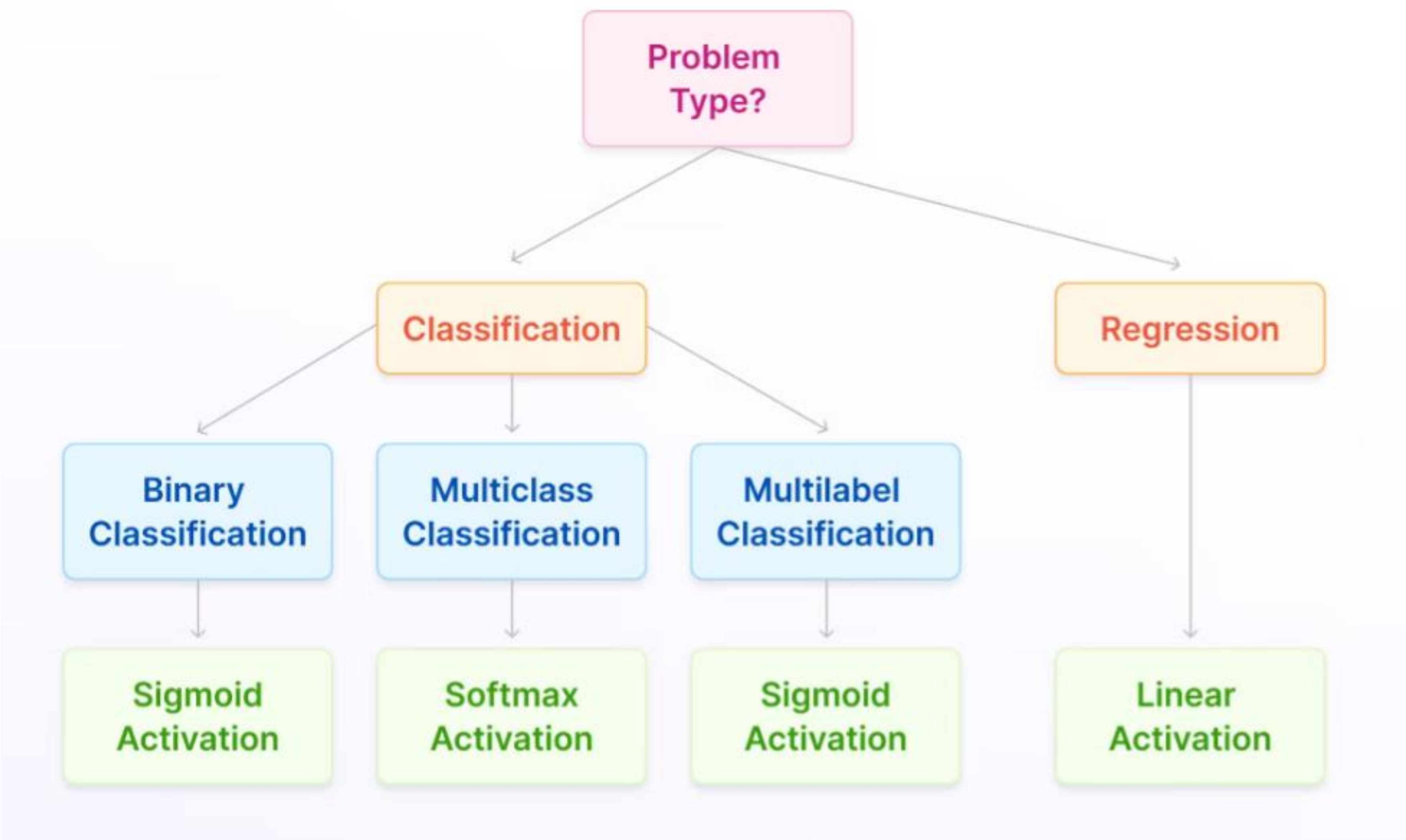
Functions in Neural Networks: Activation Function

- An activation function in the context of neural networks is a mathematical function applied to the output of a neuron.
- The purpose of an activation function is to introduce non-linearity into the model, allowing the network to learn and represent complex patterns in the data.
- Without non-linearity, a neural network would essentially behave like a linear regression model, regardless of the number of layers it has.
- The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it.
- The purpose of the activation function is to introduce non-linearity into the output of a neuron.

Functions in Neural Networks: Activation Function

- Activation functions can generally be classified into three main categories: binary step, linear, and non-linear, with numerous subcategories, derivatives, variations, and other calculations now being used in neural networks.
- **Binary step** is the simplest type of activation function, where the output is binary based on whether the input is above or below a certain threshold.
- **Linear** functions are also relatively simple, where the output is proportional to the input.
- **Non-linear** functions are more complex and introduce non-linearity into the model, such as Sigmoid and Tanh.
- In every case, the activation function is picked based on the specific problem and challenge that needs solving.
- It isn't always obvious which one data scientists and machine learning engineers need to use, so sometimes it's a case of trial and error.
- But that's always the starting point for choosing the right activation function for a neural network or any other kind of complicated algorithmic-based model that requires activation functions.

Functions in Neural Networks: Activation Function

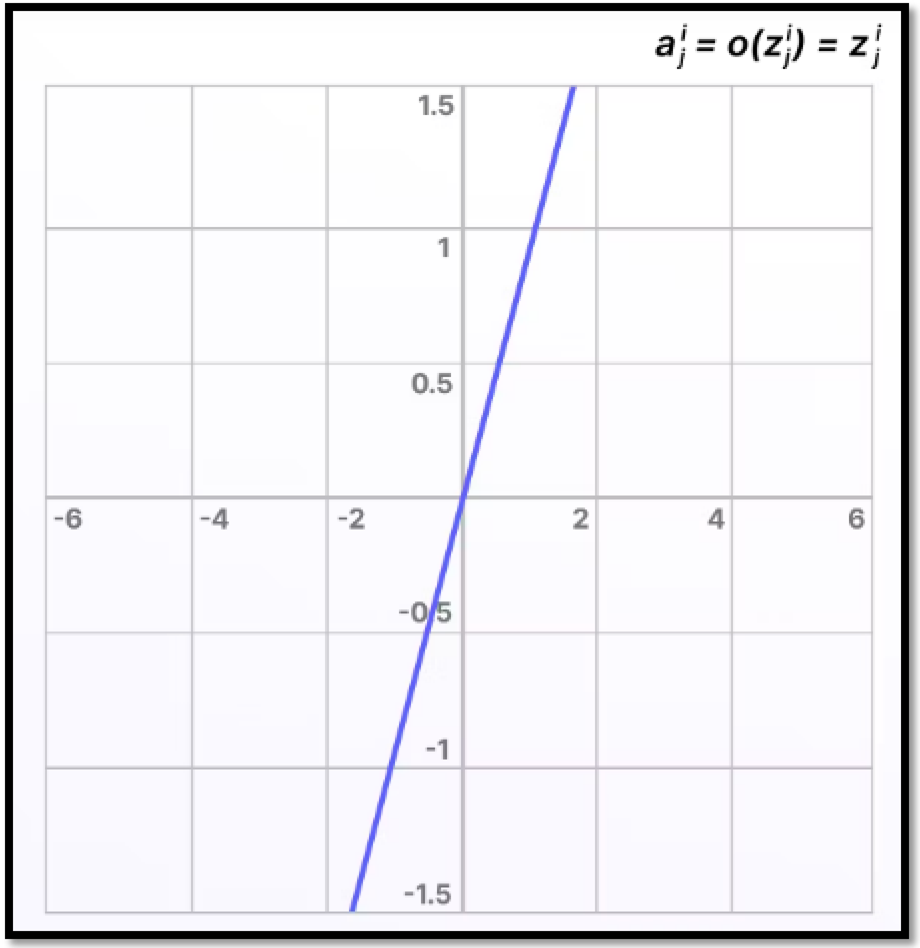


Functions in Neural Networks: Activation Function

- **Linear Activation Function (Identity):** In deep learning, data scientists use linear activation functions, also known as identity functions, when they want the output to be the same as the input signal.
- Identity is differentiable, and like a train passing through a station without stopping, this activation function doesn't change the signal in any way, so it's not used within internal layers of a DL network.
- Although, in most cases, this might not sound very useful, it is when you want the outputs of your neural network to be continuous rather than modified or discrete. There is no convergence of data, and nothing decreases either.
- If you use this activation function for every layer, then it would collapse the layers in a neural network into one.
- So, not very useful unless that's exactly what you need or there are different activation functions in the subsequent hidden layers.

Functions in Neural Networks: Activation Function

❑Linear Activation Function (Identity):

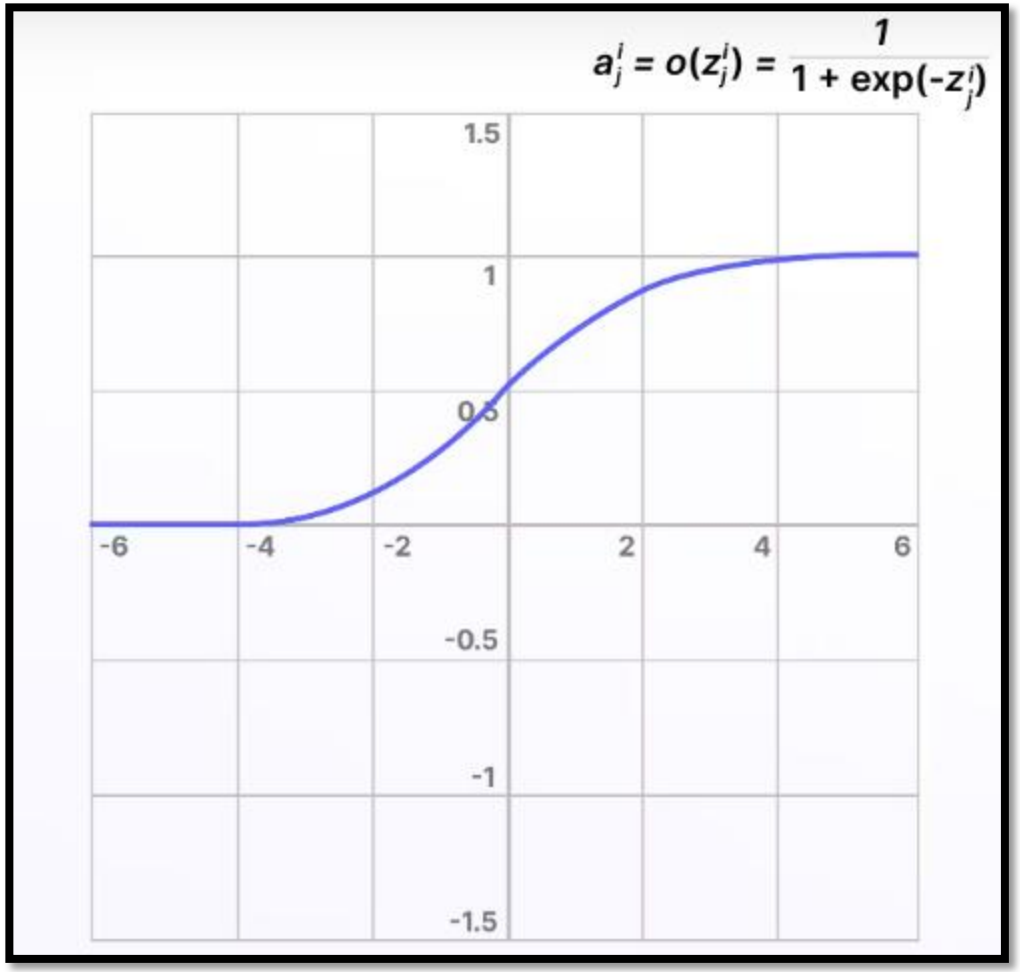


Functions in Neural Networks: Activation Function

- **Non-Linear Activation Functions: Sigmoid, Logistic Activation Functions**
- The Sigmoid activation function, also known as the logistic activation function, takes inputs and turns them into outputs ranging between 0 and 1.
- For this reason, sigmoid is referred to as the “squashing function” and is differentiable.
- Larger, more positive inputs should produce output values close to 1.0, with smaller, more negative inputs producing outputs closer to 0.0.
- It’s especially useful for classification or probability prediction tasks so that it can be implemented into the training of computer vision and deep learning networks.
- However, vanishing gradients can make these problematic when used in hidden layers, and this can cause issues when training a model.

Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Sigmoid, Logistic Activation Functions



Functions in Neural Networks: Activation Function

❑**Non-Linear Activation Functions: Tanh Function (Hyperbolic Tangent)**

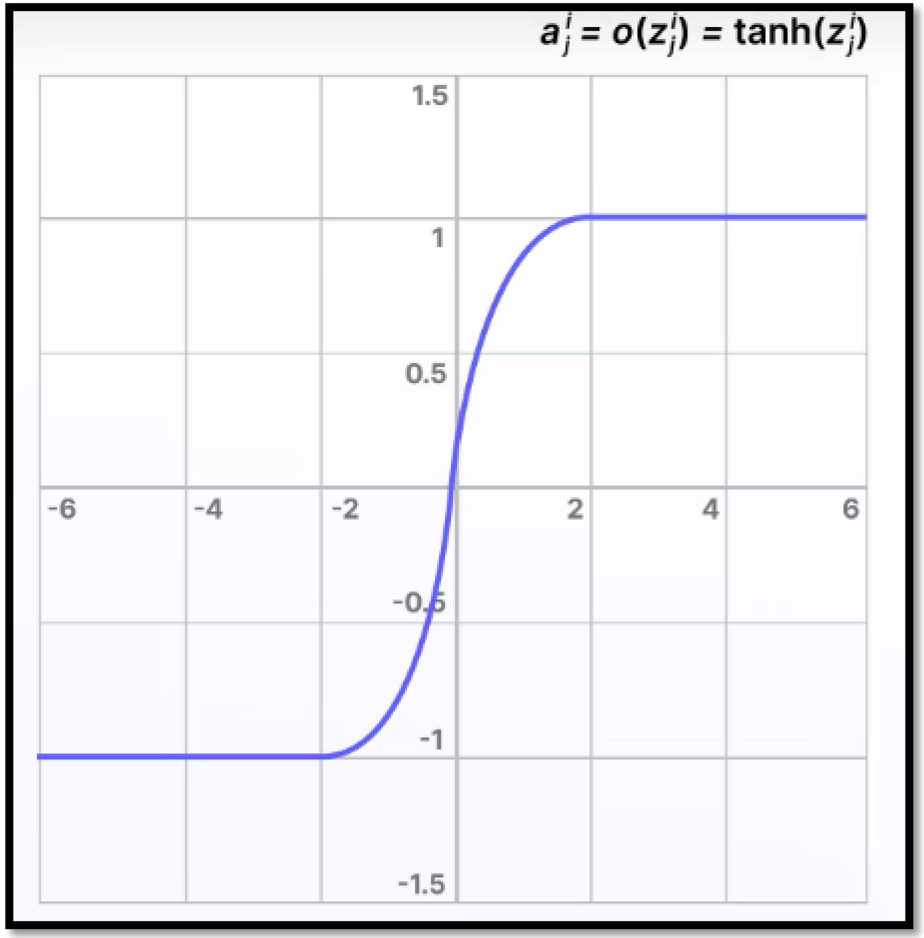
❑Tanh (or TanH), also known as the hyperbolic tangent activation function, is similar to sigmoid/logistic, even down to the S shape curve, and it is differentiable.

❑Except, in this case, the output range is -1 to 1 (instead of 0 to 1). It is a steeper gradient and also encounters the same vanishing gradient challenge as sigmoid/logistic.

❑Because the outputs of tanh are zero-centric, the values can be more easily mapped on a scale between strongly negative, neutral, or positive.

Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Tanh Function (Hyperbolic Tangent)

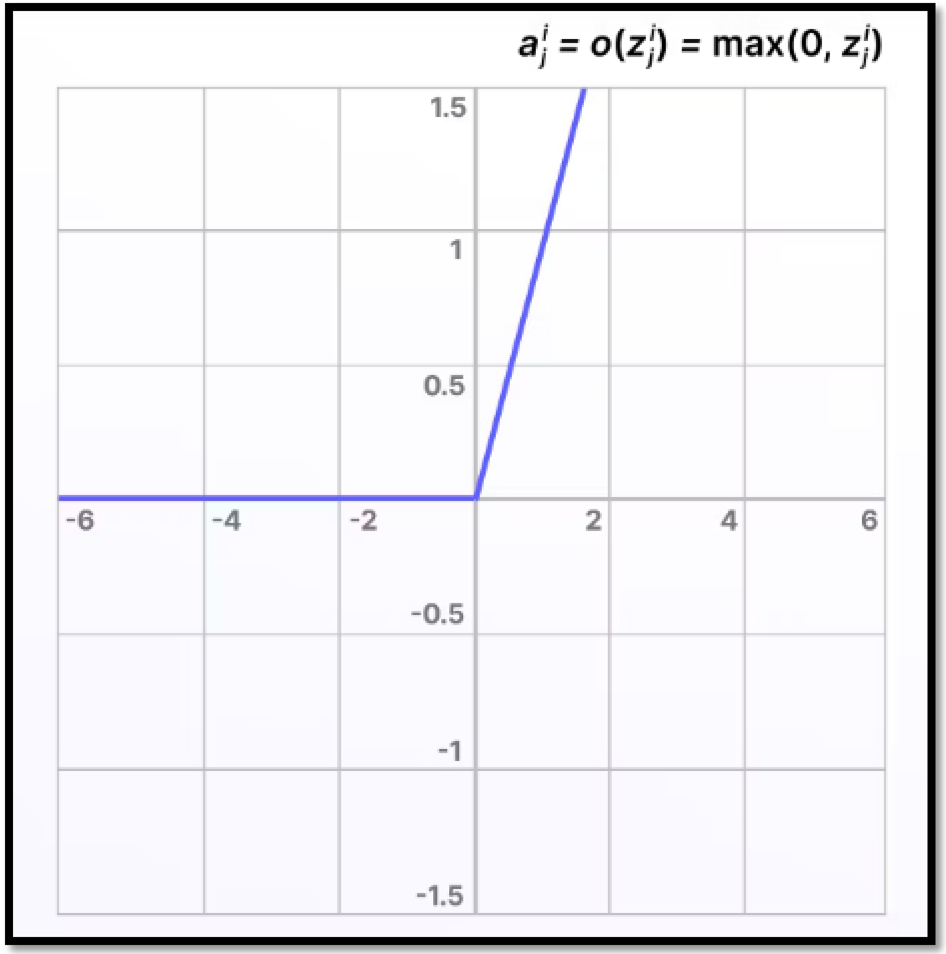


Functions in Neural Networks: Activation Function

- **Non-Linear Activation Functions: Rectified Linear Unit (ReLU)**
- Compared to linear functions, the rectified linear unit (ReLU) is more computationally efficient For many years, researchers and data scientists mainly used Sigmoid or Tanh, and then when ReLU came along, training performance increased significantly.
- ReLU isn’t differentiable, but this isn’t a problem because derivatives can be generated for ReLU.
- ReLU doesn’t activate every neuron in sequence at the same time, making it more efficient than the tanh or sigmoid/logistic activation functions.
- Unfortunately, the downside of this is that some weights and biases for neurons in the network might not get updated or activated.
- This is known as the “dying ReLU” problem, and it can be solved in a number of ways, such as using variations on this formula, including the exponential ReLU or parametric ReLU function.

Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Rectified Linear Unit (ReLU)



Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Leaky ReLU Function

❑One solution to the "dying ReLU" problem is a variation on this known as the Leaky ReLU activation function.

❑With the Leaky ReLU, instead of being 0 when $z<0$, a leaky ReLU allows a small, non-zero, constant gradient α (Normally, $\alpha=0.01$).

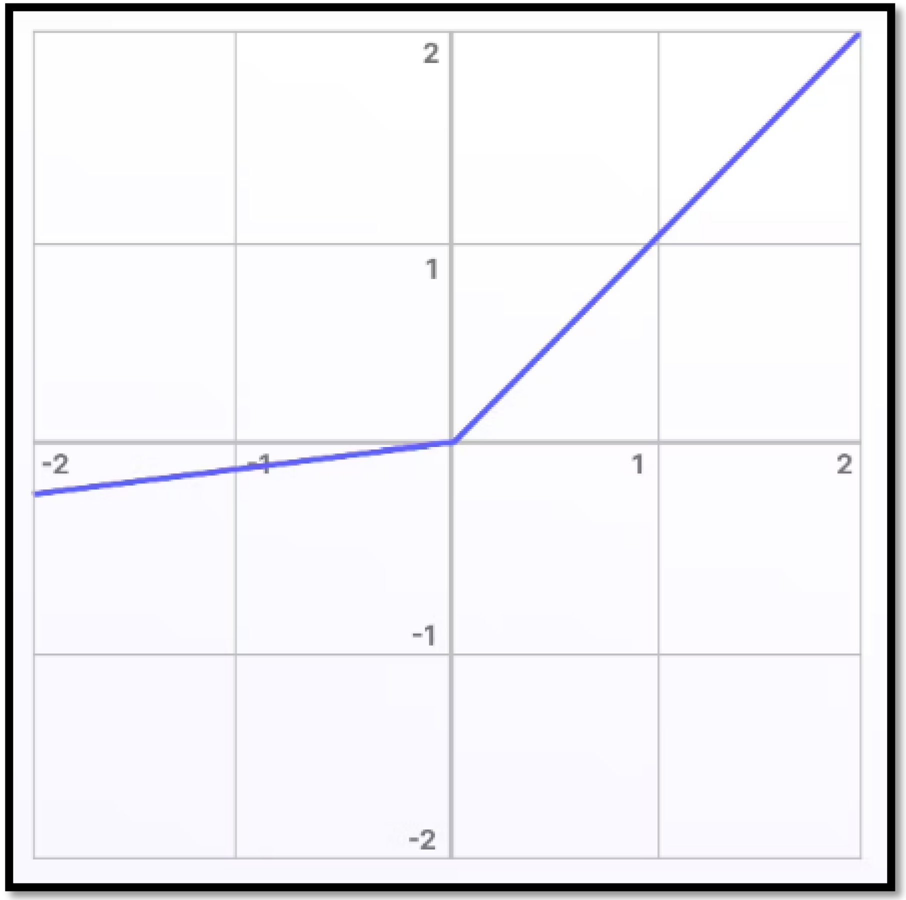
❑Leaky ReLU has been shown to perform better than the traditional ReLU activation function.

❑However, because it possesses linearity it can't be used for more complex classification tasks and lags behind more advanced activation functions such as Sigmoid and Tanh.

$$R(z) = \left\{ \begin{array}{ll} z & z > 0 \\ \alpha z & z <= 0 \end{array} \right\}$$

Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Leaky ReLU Function



Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Exponential Linear Units (ELUs) Function

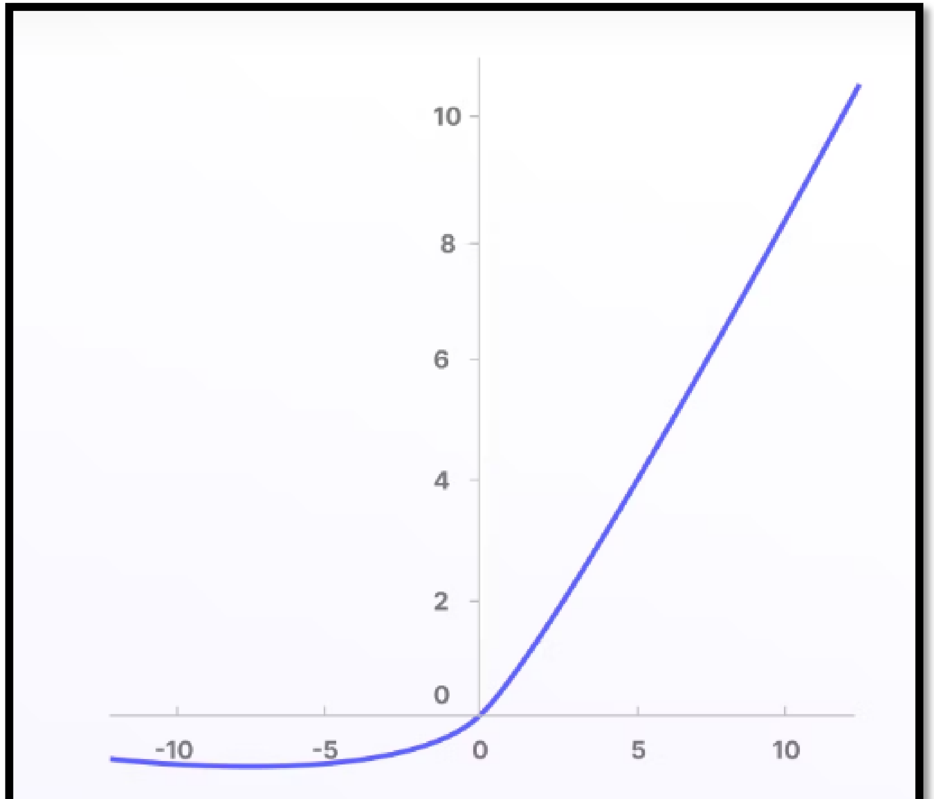
❑The exponential linear units (ELUs) function is another iteration on the original ReLU, another way to overcome the "dying ReLU" problem, and it's also not differentiable.

❑ELUs use a log curve for negative values instead of a straight line, with it becoming smooth slowly until it reaches -α.

The exponential linear unit (ELU) with $0 < \alpha$ is:

$$f(x) = x \text{ if } x > 0$$

$$\alpha(\exp(x) - 1) \text{ if } x \leq 0$$



(b) ELU

Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Scaled Exponential Linear Units (SELUs)

❑Scaled exponential linear units (SELUs)is similar to ELUs, the scaled version of this is also attempting to overcome the same challenges of ReLUs.

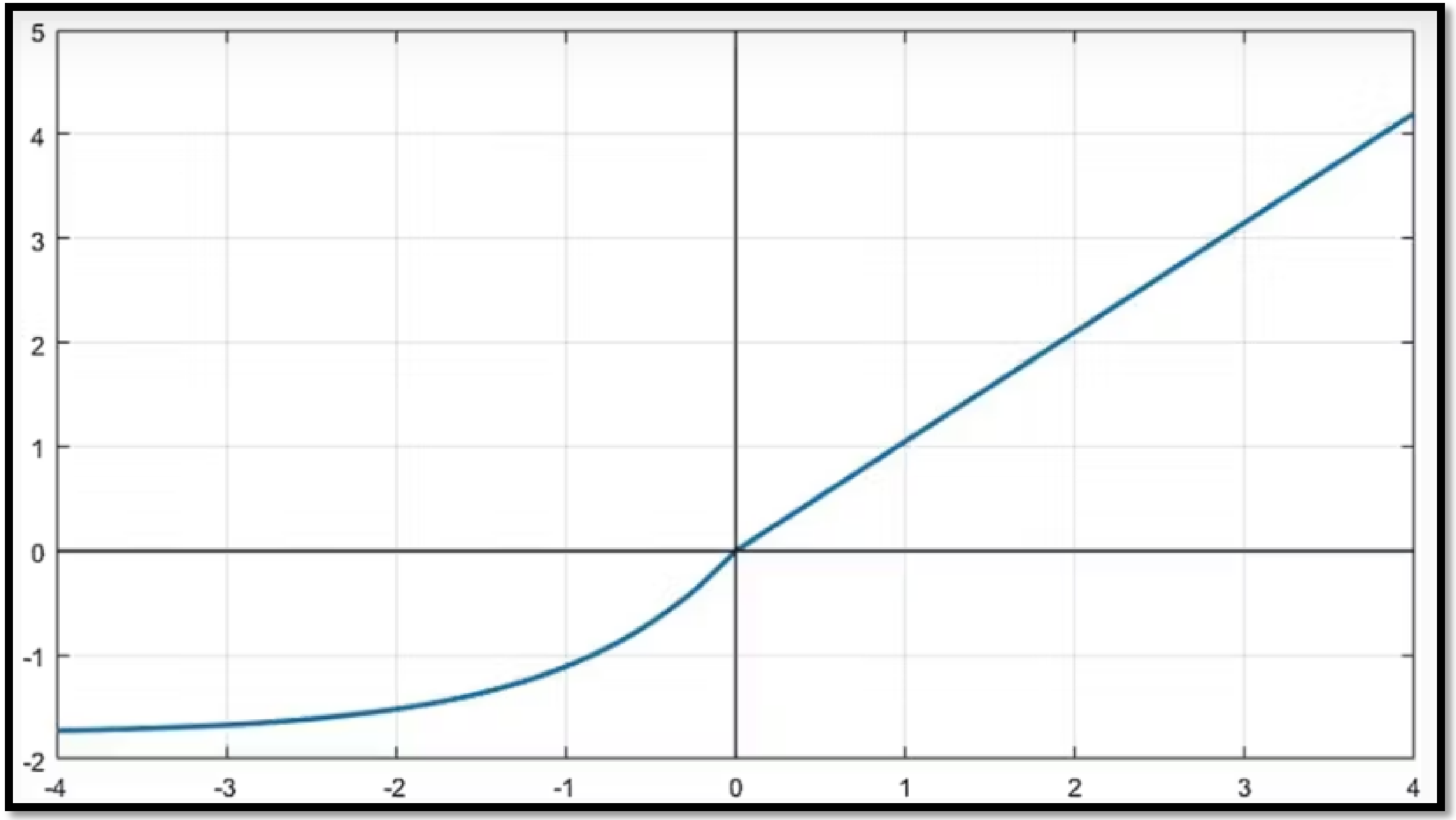
❑**SELUs** control the gradient more effectively and scale the normalization concept, and that is scales with a lambda parameter.

❑SELUs remove the problem of vanishing gradients, can't die (unlike ReLUs), and learn faster and better than other more limited activation functions.

$$\operatorname{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$

Functions in Neural Networks: Activation Function

❑Non-Linear Activation Functions: Scaled Exponential Linear Units (SELUs)



Functions in Neural Networks: Activation Function

❑Softmax

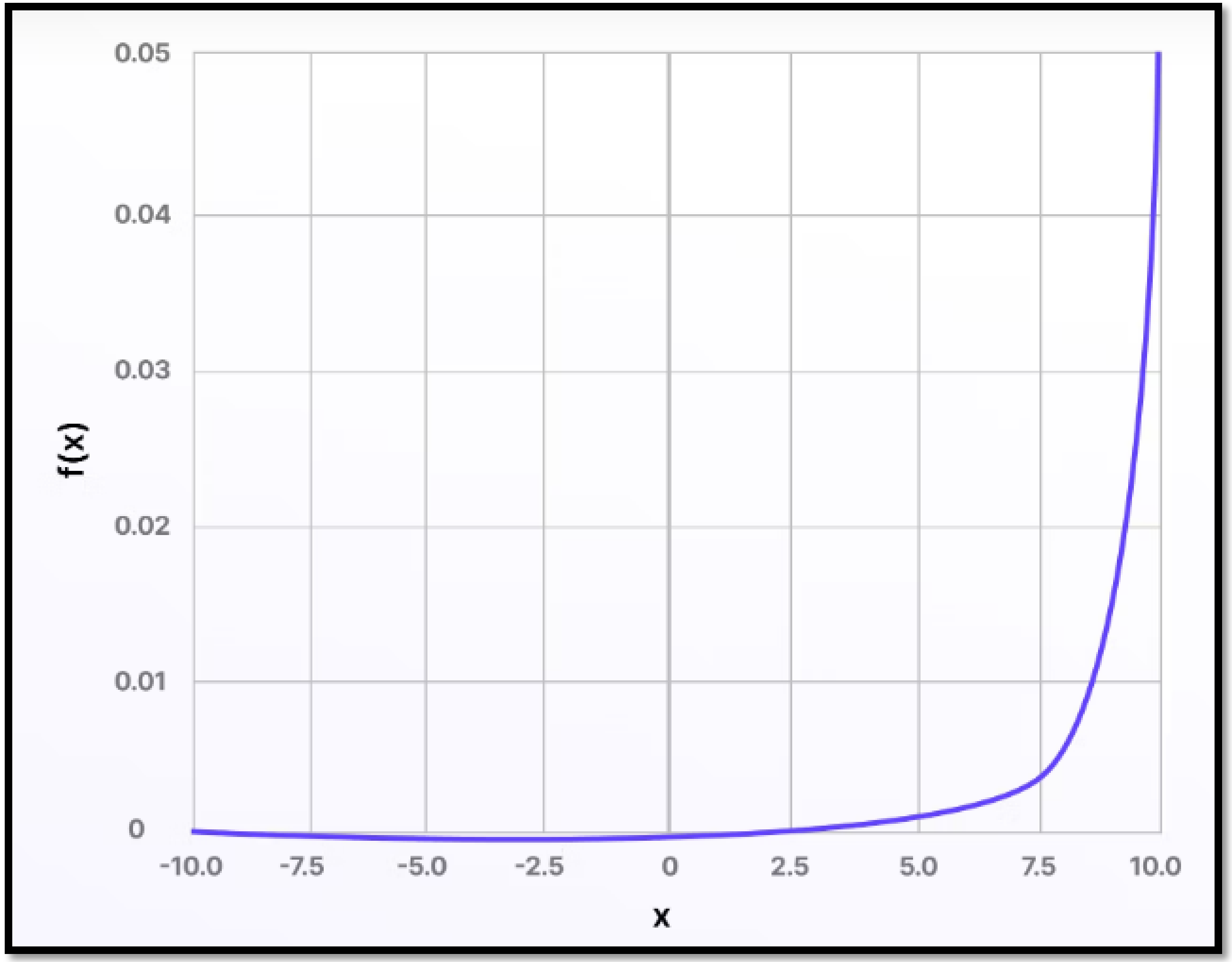
❑The softmax function, also known as the softargmax function and the multi-class logistic regression, is one of the most popular and well-used differentiable layer activation functions.

❑Softmax turns input values that are positive, negative, zero, or greater than one into values between 0 and 1.

❑By doing this, it turns input scores into a normalized probability distribution, making softmax a useful activation function in the final layer of deep learning and artificial neural networks.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

❑Softmax



Note for Students

- ❑ **This power point presentation is for lecture, therefore it is suggested that also utilize the text books and lecture notes.**