Machine Learning
ICT-4261

By-
Dr. Jesmin Akhter
Professor
Institute of Information Technology
Jahangirnagar University

Contents

The course will mainly cover the following topics:

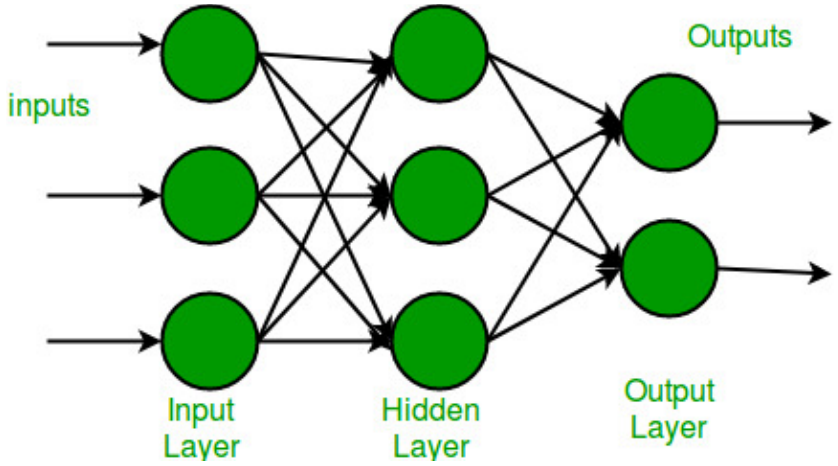
- ✔ A Gentle Introduction to Machine Learning
- ✔ Important Elements in Machine Learning
- ✔ Linear Regression
- ✔ Logistic Regression
- ✔ Naive Bayes
- ✔ Support Vector Machines
- ✔ Decision Trees and Ensemble Learning
- ✔ Neural Networks and Deep Learning
- ✔ Unsupervised Learning

Outline

- ✔ Neural Networks

Neural Networks

- ✔ Neural Networks are inspired by the most complex object in the universe – the human brain.
- ✔ Let us understand how the brain works first.
  - The human brain is made up of something called Neurons. A neuron is the most basic computational unit of any neural network, including the brain. Neurons take input, process it, and pass it on to other neurons present in the multiple hidden layers of the network, till the processed output reaches the Output Layer.
- ✔ In its simplest form, an Artificial Neural Network (ANN) has only three layers – the input layer, the output layer, and a hidden layer.

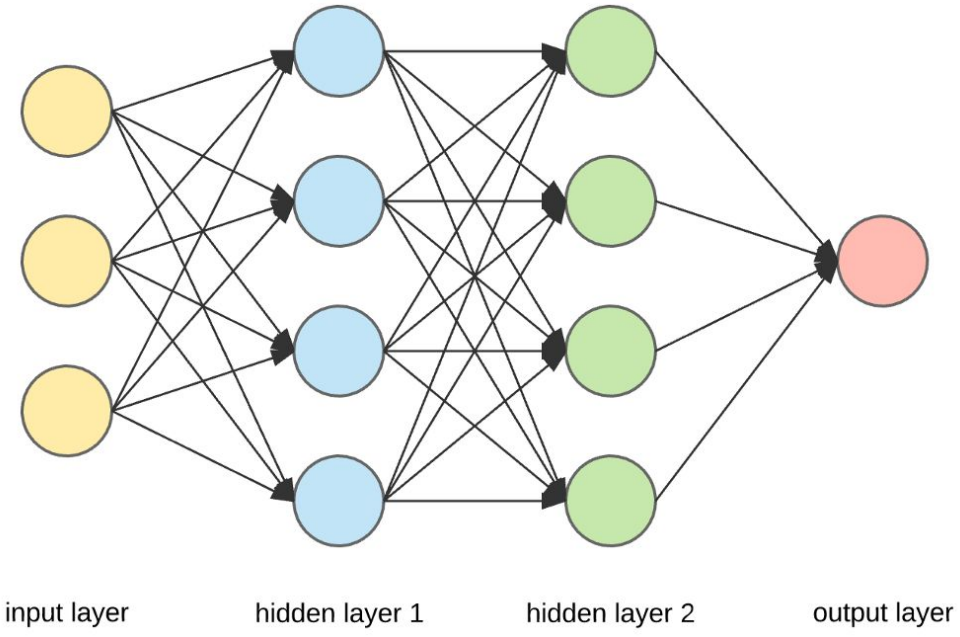


Key Components of the Neural Network Architecture

- ✔ **Input -** It is the set of features that are fed into the model for the learning and training process. For example, the input in object detection can be an array of pixel values pertaining to an image.
- ✔ **Weight -** Its main function is to give importance to those features that contribute more towards the learning. It does so by introducing scalar multiplication between the input value and the weight matrix. For example, a negative word would impact the decision of the sentiment analysis model more than a pair of neutral words.
- ✔ **Transfer function -** The job of the transfer function is to combine multiple inputs into one output value so that the activation function can be applied. It is done by a simple summation of all the inputs to the transfer function.
- ✔ **Activation Function**—The role of the activation function is to decide whether or not a specific neuron should be activated. This decision is based on whether or not the neuron's input will be important to the prediction process
- ✔ **Bias -** The role of bias is to shift the value produced by the activation function. Its role is similar to the role of a constant in a linear function.

Key Components of the Neural Network Architecture

- ✔ When multiple neurons are stacked together in a row, they constitute a layer, and multiple layers piled next to each other are called a multi-layer neural network.



Multi-layer neural network

Key Components of the Neural Network Architecture

- ✔ **Input Layer**
- ✔ The data that we feed to the model is loaded into the input layer from external sources like a CSV file or a web service. It is the only visible layer in the complete Neural Network architecture that passes the complete information from the outside world without any computation.
- ✔ **Hidden Layers**
- ✔ The hidden layers are what makes deep learning what it is today. They are intermediate layers that do all the computations and extract the features from the data.
- ✔ There can be multiple interconnected hidden layers that account for searching different hidden features in the data.
  - For example, in image processing, the first hidden layers are responsible for higher-level features like edges, shapes, or boundaries. On the other hand, the later hidden layers perform more complicated tasks like identifying complete objects (a car, a building, a person).

Key Components of the Neural Network Architecture

- ✔ **Output Layer**
- ✔ The output layer takes input from preceding hidden layers and comes to a final prediction based on the model's learnings. It is the most important layer where we get the final result.
- ✔ In the case of classification/regression models, the output layer generally has a single node. However, it is completely problem-specific and dependent on the way the model was built.

Categories of Neural Networks

- ✔ Neural networks are majorly divided into two categories:
  - Deep neural networks and Shallow neural networks.
  - Their distinction is based on the number of hidden layers in their architecture. As the names suggest, deep neural networks have a greater number of layers than their shallow counterparts.

Shallow Neural Networks

- ✔ These refer to a class of artificial neural networks that typically consist of an input layer and an output layer, with at most one hidden layer in between. Hence, they have a simpler structure and are used for less complex tasks as they are simpler to understand and implement.
- ✔ Their simpler architecture also results in the need for less computational power and resources, making them quicker to train. However, it also results in their limitations as they struggle with complex modeling problems. Moreover, they possess the risk of overfitting.
- ✔ Despite these limitations, shallow neural networks are a useful resource. Some of their practical applications in the world of data analysis include:
- ✔ **Linear Regression**: Shallow neural networks are often used in linear regression problems where the relationship between the input and output variables is relatively simple
- ✔ **Classification Tasks**: They are effective for simple classification tasks where the input data can be linearly separable or nearly so

Deep Neural Networks

- ✔ **Deep Neural Networks** is defined by several layers of neurons between the input and output layers. This depth of architecture enables them to model complex patterns and relationships within large sets of data, making them highly effective for a wide range of artificial intelligence tasks.
- ✔ Thus, they offer enhanced accuracy when trained using sufficient amounts of data. Moreover, they have the ability to automatically detect and use relevant features from raw data, reducing the need for manual feature engineering. However, they also possess the risk of overfitting while also using extensive computational resources.
- ✔ Owing to these benefits and limitations, deep neural networks have made their mark in the industry for the following uses:
- ✔ **Image Recognition**: They are extensively used in image recognition tasks, enabling applications such as facial recognition and medical image analysis
- ✔ **Speech Recognition**: They are pivotal in developing systems that convert spoken language into text, used in virtual assistants and automated transcription services
- ✔ **Natural Language Processing (NLP)**: They help in understanding and generating human language, applicable in translation services, sentiment analysis, and chatbots

The Perceptron

Based on the layers, Perceptron models are divided into two types. These are as follows:

- ✔ Single-layer Perceptron Model
- ✔ Multi-layer Perceptron model

Single Layer Perceptron Model:

- ✔ It is a shallow neural network with the simplest model consisting of just one hidden layer. Hence, it is also considered a fundamental building block for neural networks. It can perform certain computations to detect features or business intelligence in the input data.
- ✔ it acts as a single-layer neural network with four main parameters, i.e., **input values, weights and Bias, net sum, and an activation function.**
- ✔ Perceptrons take multiple numerical inputs and assign a weight to each to determine their influence on the output. The weighted inputs are then summed together and the value is passed through an activation function to determine the output.
- ✔ This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

The Perceptron

Multi-Layered Perceptron Model:

- ✔ Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.
- ✔ Typically consists of fully connected layers. Each neuron in one layer connects to every neuron in the next layer. MLPs can have multiple hidden layers, but they are still considered "shallow" compared to more complex architectures.
- ✔ The multi-layer perceptron model is also known as the **Backpropagation algorithm**, which executes in two stages as follows:
- ✔ **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.
- ✔ **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.
- ✔ Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.
- ✔ A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.

What is the difference between Perceptron and Multi-layer Perceptron?

The Perceptron is a single-layer neural network used for binary classification, learning linearly separable patterns. In contrast, a Multi-layer Perceptron (MLP) has multiple layers, enabling it to learn complex, non-linear relationships. MLPs have input, hidden, and output layers, allowing them to handle more intricate tasks compared to the simpler Perceptron.

Feed Forward Neural Networks (FNNs)

- Also known as feedforward networks, they are a type of shallow neural network where connections between the nodes do not form a cycle. They form a multi-layered network where information moves in only one direction—from input to output, through one or more layers of nodes, without any cycles or loops.

Deep Neural Networks

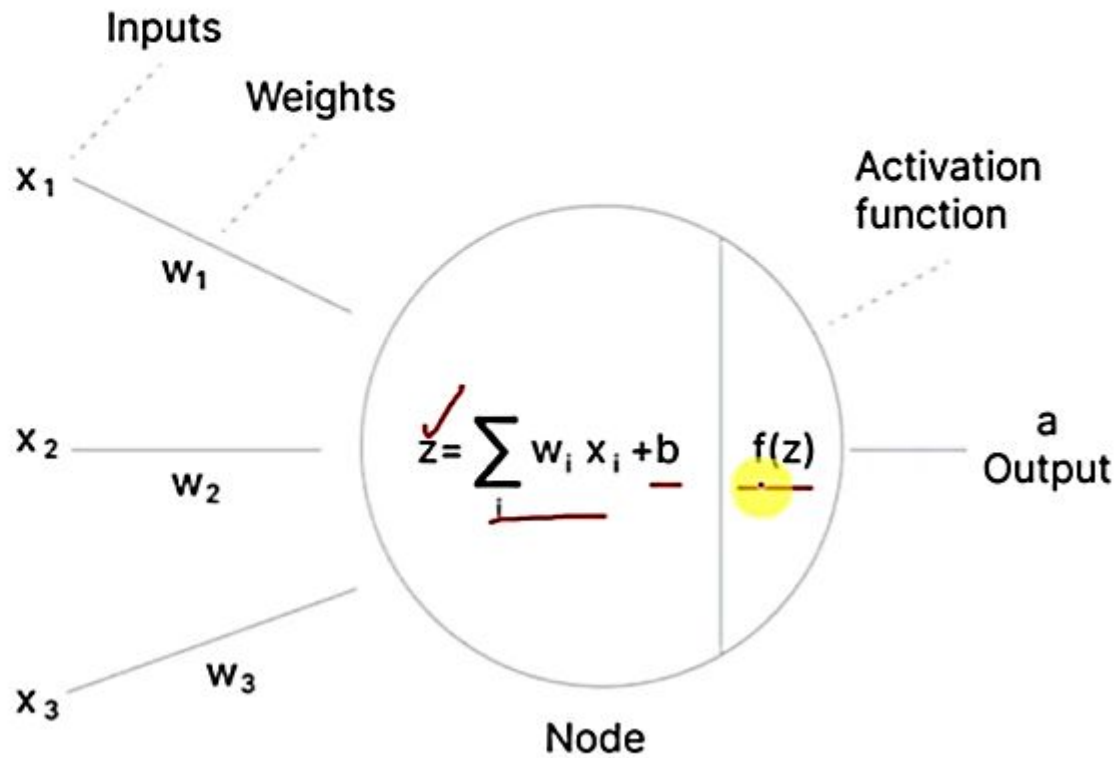
- ✔ **Residual Networks (ResNet)**
  - The greater the number of hidden layers, the better is the learning process.
  - Very deep Neural Networks are extremely difficult to train due to vanishing and exploding gradient problems.
  - ResNets provide an alternate pathway for data to flow to make the training process much faster and easier.
  - This is different from the feed-forward approach of earlier Neural Networks architectures.
  - The core idea behind ResNet is that a deeper network can be made from a shallow network by copying weight from the shallow counterparts using identity mapping.
- ✔ **Recurrent Neural Networks** have the power to remember what it has learned in the past and apply it in future predictions
- ✔ **The Long Short Term Memory Network (LSTM):** We want Recurrent Neural Networks to be able to remember what happened many timestamps ago. To achieve this, we need **to add extra structures called gates** to the artificial neural network structure.
  - **Cell state (c_t):** It corresponds to the long-term memory content of the network.
  - **Forget Gate:** Some information in the cell state is no longer needed and is erased. The gate receives two inputs, x_t (current timestamp input) and h_t-1 (previous cell state), multiplied with the relevant weight matrices before bias is added. The result is sent into an activation function, which outputs a binary value that decides whether the information is retained or forgotten.
  - **Input gate:** It decides what piece of new information is to be added to the cell state. It is similar to the forget gate using the current timestamp input and previous cell state with the only difference of multiplying with a different set of weights.
  - **Output gate:** The output gate's job is to extract meaningful information from the current cell state and provide it as an output.

Standard Neural Networks

- ✔ **Echo State Networks (ESN)**
- ✔ **Convolutional Neural Networks (CNNs)**
  - Convolutional Neural Networks is a type of Feed-Forward Neural Networks used in tasks like image analysis, natural language processing, and other complex image classification problems.
  - A CNN has hidden layers of convolutional layers that form the base of ConvNets. *Features* refer to minute details in the image data like edges, borders, shapes, textures, objects, circles, etc.
  - At a higher level, convolutional layers detect these patterns in the image data with the help of filters. The higher-level details are taken care of by the first few convolutional layers.
  - The deeper the network goes, the more sophisticated the pattern searching becomes.
  - For example, in later layers rather than edges and simple shapes, filters may detect specific objects like eyes or ears, and eventually a cat, a dog, and what not.
- ✔ **Deconvolutional Neural Networks** are CNNs that work in a reverse manner

What are Activation Functions…?

- **An Activation Function** decides whether a neuron should be activated or not.
- This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.

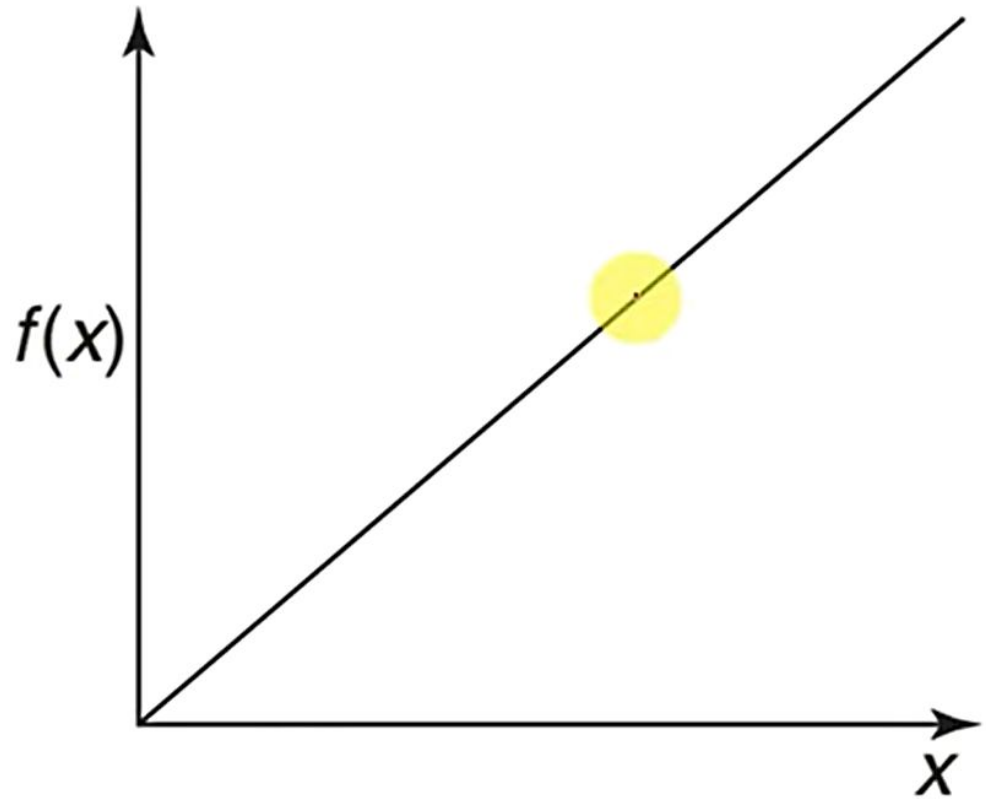


Activation Functions – Identity Function

- **Identity function:** It is a linear function and can be defined as

$$f(x) = x \qquad for\ all\ x$$

- The output here remains the same as input.
- The input layer uses the identity activation function.

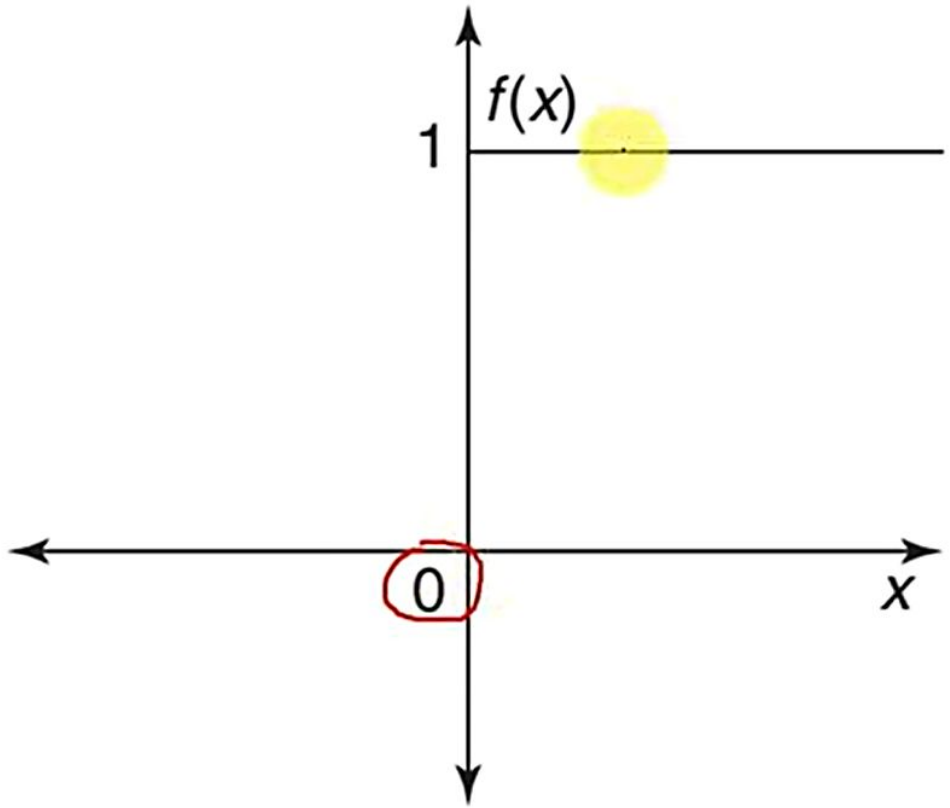


Activation Functions – Binary Step Function

- **Binary step function:** This function can be defined as

$$f(x) = \begin{cases} 1 & \text{if} \quad x \geq \theta \\ 0 & \text{if} \quad x < \theta \end{cases}$$

- where θ represents the threshold value.
- This function is most widely used in single-layer nets to convert the net input to an output that is a binary (1 or 0).



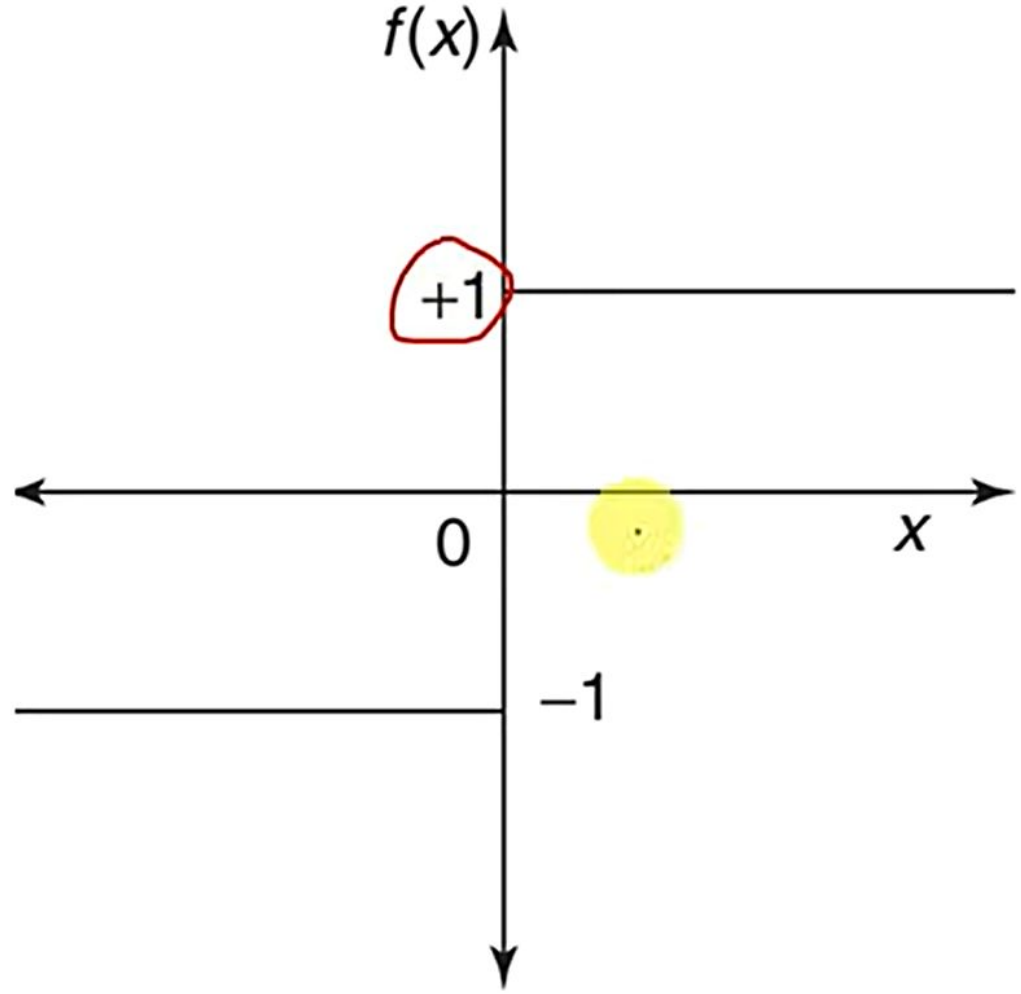
Activation Functions - Binary Step Function

- Here are some of the limitations of binary step function:
- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

Activation Functions – Bipolar Step Function

- **Bipolar step function:** This function can be defined as

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ -1 & \text{if } x < \theta \end{cases}$$



- where θ represents the threshold value.
- This function is most widely used in single-layer nets to convert the net input to an output that is a bipolar (-1 or +1).

Activation Functions – Non-Linear Activation Function

- The linear activation function shown above is simply a linear regression model.
- Because of its limited power, this does not allow the model to create complex mappings between the network's inputs and outputs.

Activation Functions – Non-Linear Activation Function

- Non-linear activation functions solve the following limitations of linear activation functions:
- They allow backpropagation because now the derivative function would be related to the input, and it's possible to go back and understand which weights in the input neurons can provide a better prediction.
- They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers.
- Any output can be represented as a functional computation in a neural network.

Activation Functions – Non-Linear Activation Function

10 Non-Linear Neural Networks Activation Functions

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU
- Parametric ReLU
- ELU
- Softmax
- Swish
- GELU
- SELU

Activation Functions – Sigmoidal Function

- **Sigmoidal functions:** The sigmoidal functions are widely used in back-propagation neural networks.

1. Binary sigmoid function
2. Bipolar sigmoid function

Activation Functions – Binary Sigmoidal Function

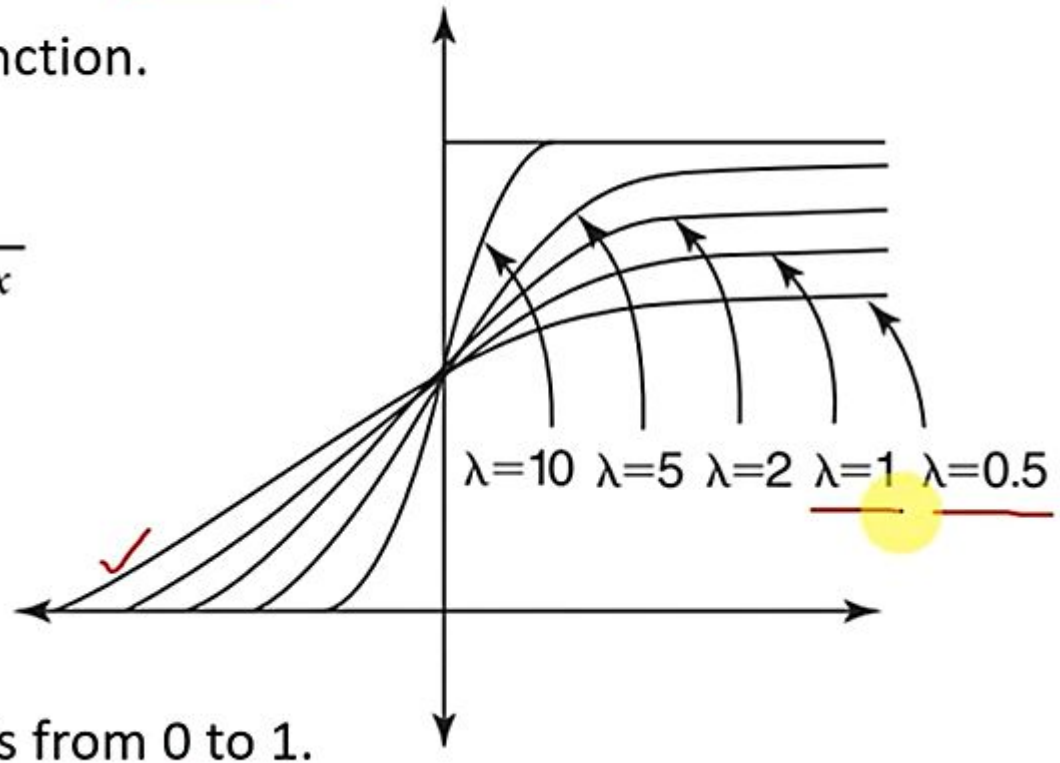
- **Binary Sigmoid function:** It is also known as logistic sigmoid function or unipolar sigmoid function.
- It can be defined as

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

- where λ is the steepness parameter.
- The derivative of this function is

$$f'(x) = \lambda f(x)[1-f(x)]$$

- Here the range of the sigmoid function is from 0 to 1.



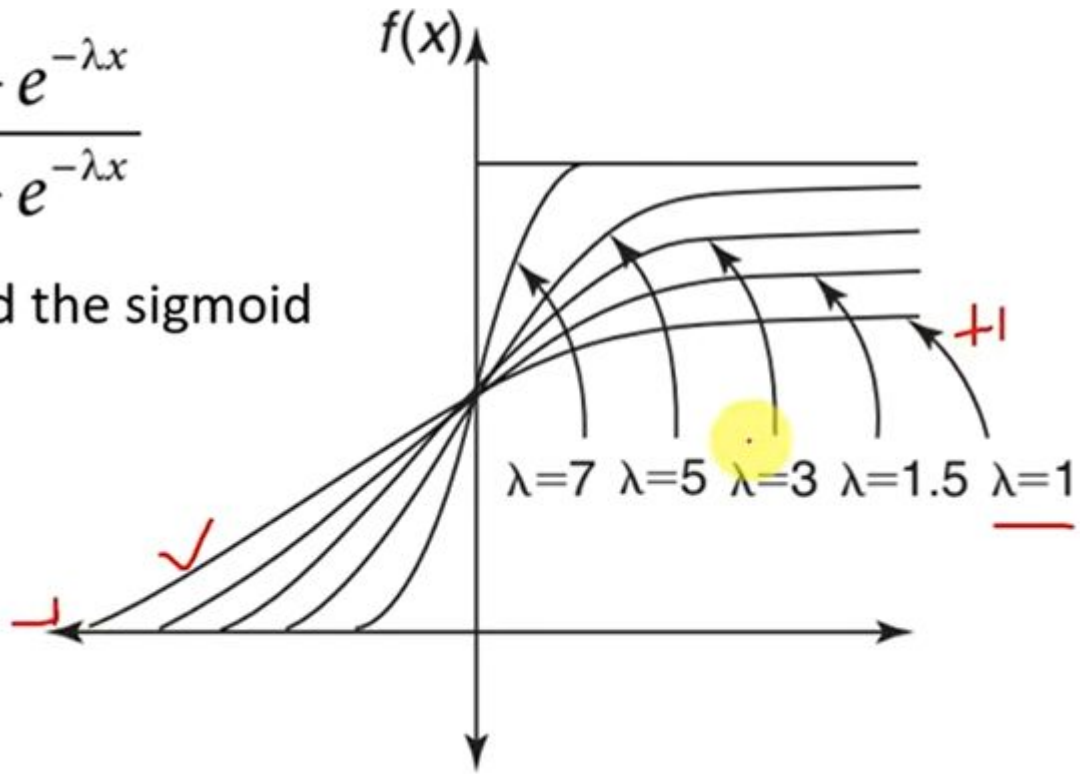
Activation Functions – Bipolar Sigmoidal Function

- **Bipolar Sigmoid function:** This function is defined as

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

- where λ is the steepness parameter and the sigmoid function range is between –1 and +1.
- The derivative of this function is

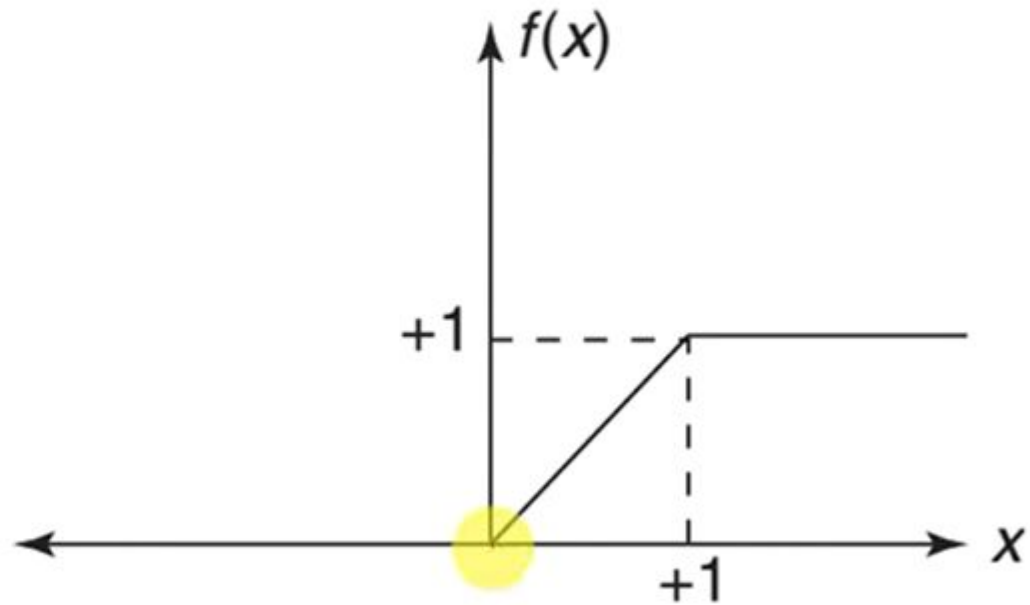
$$f'(x) = \frac{\lambda}{2}[1+f(x)][1-f(x)]$$



Activation Functions – Ramp Function

- **Ramp function:**
- The ramp function is defined as

$$f(x) = \begin{cases} 1 & \text{if} \quad x > 1 \\ x & \text{if} \quad 0 \le x \le 1 \\ 0 & \text{if} \quad x < 0 \end{cases}$$



Radial Basis Function (RBF) Neural Networks

- ✔ RBF network in its simplest form is a three-layer feedforward neural network. The first layer corresponds to the inputs of the network, the second is a hidden layer consisting of a number of RBF non-linear activation units, and the last one corresponds to the final output of the network. Activation functions in RBFNs are conventionally implemented as Gaussian functions.
- ✔ A Radial Basis Function (RBF), also known as kernel function, is applied to the distance to calculate every neuron's weight (influence). The name of the Radial Basis Function comes from the radius distance, which is the argument to the function. Weight = RBF[distance)The greater the distance of a neuron from the point being evaluated, the less influence (weight) it has.

Radial Basis Function (RBF) Neural Networks
Solved Example

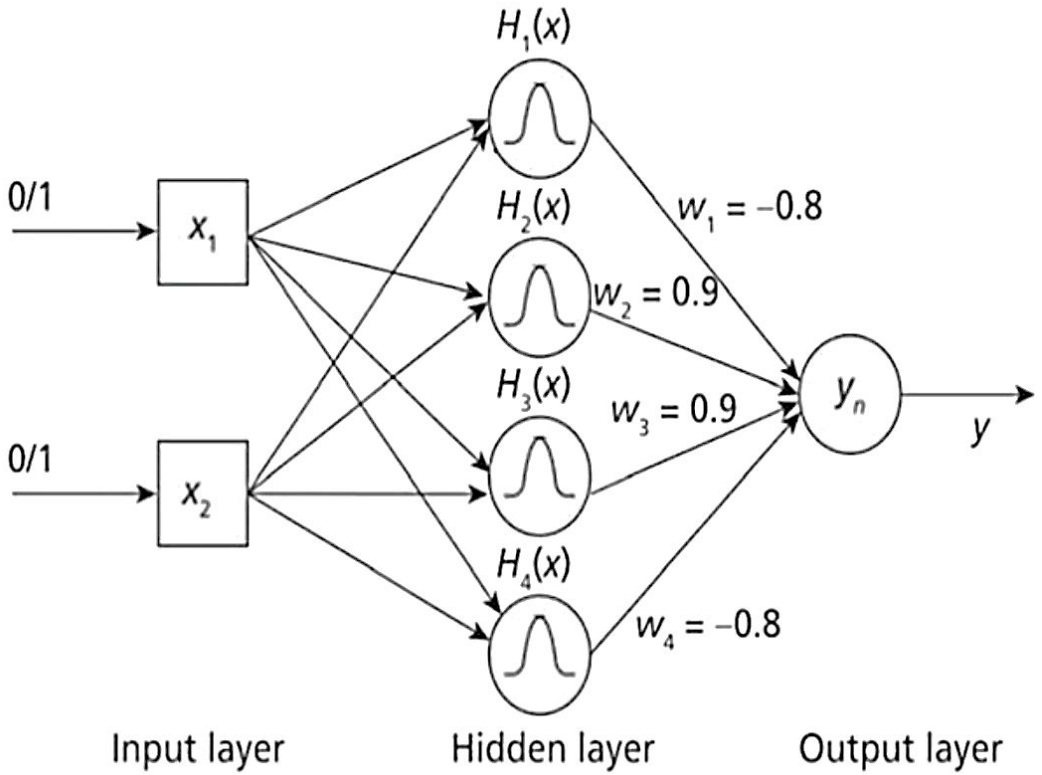
Consider the XOR Boolean function that has patterns (0, 0) (0, 1) ((1, 0) and (1, 1) in a 2-D input space. Radius is r=1.414. Construct a Radial Basis Function Neural Network as shown in the Figure that classifies the input pattern:

(0, 0) ☐ 0

(0, 1) ☐1

(1, 0) ☐1

(1, 1) ☐0



Radial Basis Function
Solved Example

Define 4 hidden layer neurons with Gaussian RBF:

$$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} \; ; c1 = (0, 0)$$

$$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} \; ; c2 = (0, 1)$$

$$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} \; ; c3 = (1, 0)$$

$$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} \; ; c4 = (1, 1)$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

For input pattern (0, 0)

Distance squared of x from $c1 = (0, 0)$

$= (0-0)^2 + (0-0)^2 = 0$

$$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} = e^{-\frac{0}{2}} = 1.0$$

Distance squared of x from $c2 = (0, 1)$

$= (0-0)^2 + (0-1)^2 = 1$

$$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$$

Distance squared of x from $c3 = (1, 0)$

$= (0-1)^2 + (0-0)^2 = 1$

$$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$$

Distance squared of x from $c4 = (1, 1)$

$= (0-1)^2 + (0-1)^2 = 2$

$$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$$

$\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 1.0 + 0.9 \times 0.6 + 0.9 \times 0.6 +$

$-0.8 \times 0.4 = -0.04$

Radial Basis Function
Solved Example

Define 4 hidden layer neurons with Gaussian RBF:

$$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} \; ; c1 = (0, 0)$$

$$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} \; ; c2 = (0, 1)$$

$$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} \; ; c3 = (1, 0)$$

$$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} \; ; c4 = (1, 1)$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

For input pattern (0, 1)

Distance squared of x from $c1 = (0, 0)$

$= (0 - 0)^2 + (1 - 0)^2 = 1$

$$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$$

Distance squared of x from $c2 = (0, 1)$

$= (0 - 0)^2 + (1 - 1)^2 = 0$

$$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{0}{2}} = 1.0$$

Distance squared of x from $c3 = (1, 0)$

$= (0 - 1)^2 + (1 - 0)^2 = 2$

$$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$$

Distance squared of x from $c4 = (1, 1)$

$= (0 - 1)^2 + (1 - 1)^2 = 1$

$$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$$

$\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.6 + 0.9 \times 1.0 + 0.9 \times 0.4 +$
$-0.8 \times 0.6 = 0.3$

Radial Basis Function
Solved Example

Define 4 hidden layer neurons with Gaussian RBF:

$$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} \; ; c1 = (0, 0)$$

$$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} \; ; c2 = (0, 1)$$

$$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} \; ; c3 = (1, 0)$$

$$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} \; ; c4 = (1, 1)$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

For input pattern (1, 0)

Distance squared of x from $c1 = (0, 0)$

$= (1 - 0)^2 + (0 - 0)^2 = 1$

$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ ✓

Distance squared of x from $c2 = (0, 1)$

$= (1 - 0)^2 + (0 - 1)^2 = 2$

$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$ ✓

Distance squared of x from $c3 = (1, 0)$

$= (1 - 1)^2 + (0 - 0)^2 = 0$

$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{1}{2}} = 1.0$ ✓

Distance squared of x from $c4 = (1, 1)$

$= (1 - 1)^2 + (0 - 1)^2 = 1$

$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ ✓

$\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.6 + 0.9 \times 0.4 + 0.9 \times 1.0 +$

$-0.8 \times 0.6 = 0.3$

Radial Basis Function
Solved Example

Define 4 hidden layer neurons with Gaussian RBF:

$$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} ; c1 = (0, 0)$$

$$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} ; c2 = (0, 1)$$

$$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} ; c3 = (1, 0)$$

$$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} ; c4 = (1, 1)$$

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

For input pattern (1, 1)

Distance squared of x from $c1 = (0, 0)$

$= (1 - 0)^2 + (1 - 0)^2 = 2$

$H_1(x) = e^{\frac{-(x-c1)^2}{r^2}} = e^{-\frac{2}{2}} = 0.4$ ✓

Distance squared of x from $c2 = (0, 1)$

$= (1 - 0)^2 + (1 - 1)^2 = 1$

$H_2(x) = e^{\frac{-(x-c2)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ ✓

Distance squared of x from $c3 = (1, 0)$

$= (1 - 1)^2 + (1 - 0)^2 = 1$

$H_3(x) = e^{\frac{-(x-c3)^2}{r^2}} = e^{-\frac{1}{2}} = 0.6$ ✓

Distance squared of x from $c4 = (1, 1)$

$= (1 - 1)^2 + (1 - 1)^2 = 0$

$H_4(x) = e^{\frac{-(x-c4)^2}{r^2}} = e^{-\frac{0}{2}} = 1.0$

$\sum_{j=1}^{m} w_j H_j(x) = -0.8 \times 0.4 + 0.9 \times 0.6 + 0.9 \times 0.6 +$

$-0.8 \times 1.0 = -0.04$

Radial Basis Function

Solved Example

Forward Phase Calculation

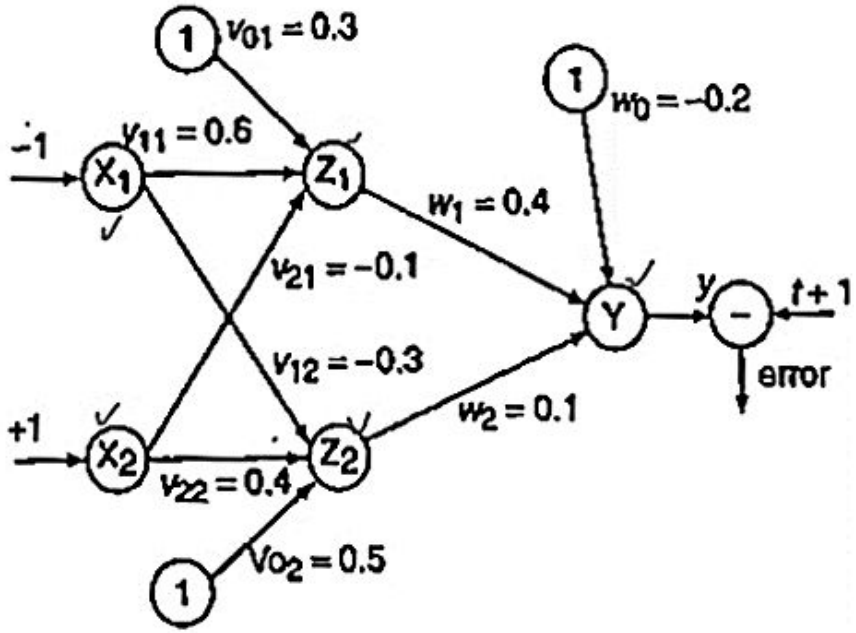
Input		$H_1(x)$	$H_2(x)$	$H_3(x)$	$H_4(x)$	$\sum_{j=1}^{m} w_j H_j(x)$	Output
0	0	1.0	0.6	0.6	0.4	–0.04	0
0	1	0.6	1.0	0.4	0.6	0.3	1
1	0	0.6	0.4	1.0	0.6	0.3	1
1	1	0.4	0.6	0.6	1.0	–0.04	0
		$w_1 = -0.8$	$w_2 = 0.9$	$w_3 = 0.9$	$w_4 = -0.8$		

How does the network learn?

- ✔ The training samples are passed through the network and the output obtained from the network is compared with the actual output. This error is used to change the weights of the neurons such that the error decreases gradually. This is done using the **Backpropagation** algorithm, also called backprop. Iteratively passing batches of data through the network and updating the weights, so that the error is decreased, is known as **Stochastic Gradient Descent (SGD)**. The amount by which the weights are changed is determined by a parameter called **Learning rate**.

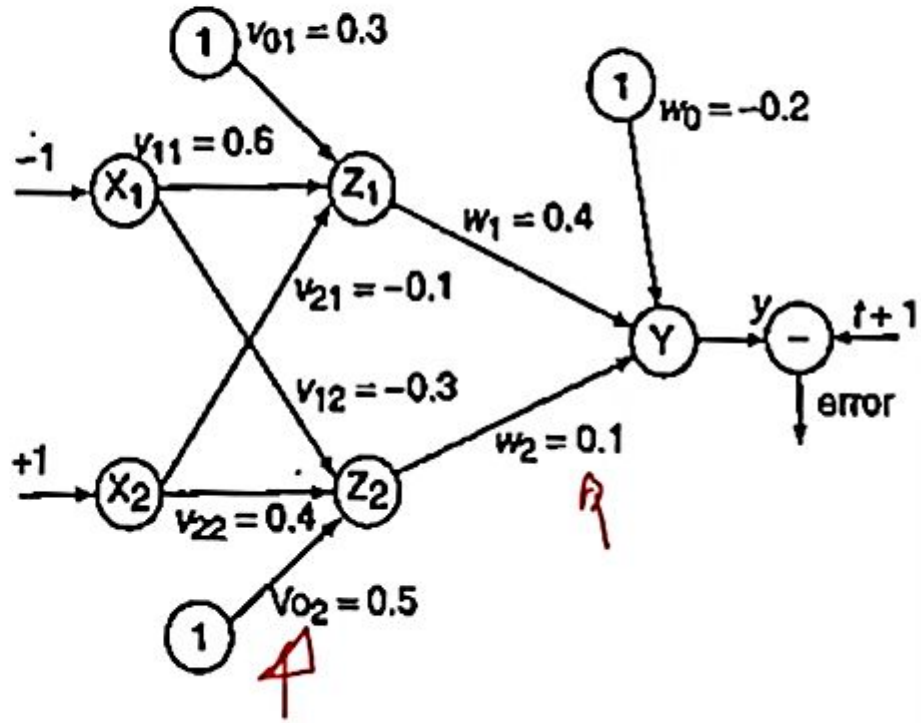
How to Find New Weights – Back Propagation Algorithm

✔ Using back-propagation network, find the new weights for the figure shown. It is presented with the input pattern [-1, 1] and the target output is 1.Use a learning rate α = 0.25.and bipolar sigmoidal activation function.



How to Find New Weights – Back Propagation Algorithm

- The new weights are calculated based on the back propagation training algorithm.
- The initial weights are
- $[v_{11}, v_{21}, v_{01}] = [0.6, -0.1, 0.3]$
- $[v_{12}, v_{22}, v_{02}] = [-0.3, 0.4, 0.5]$
- $[w_1, w_2, w_0] = [0.4, 0.1, -0.2]$
- and the learning' rate is $\alpha = 0.25$



How to Find New Weights – Back Propagation Algorithm

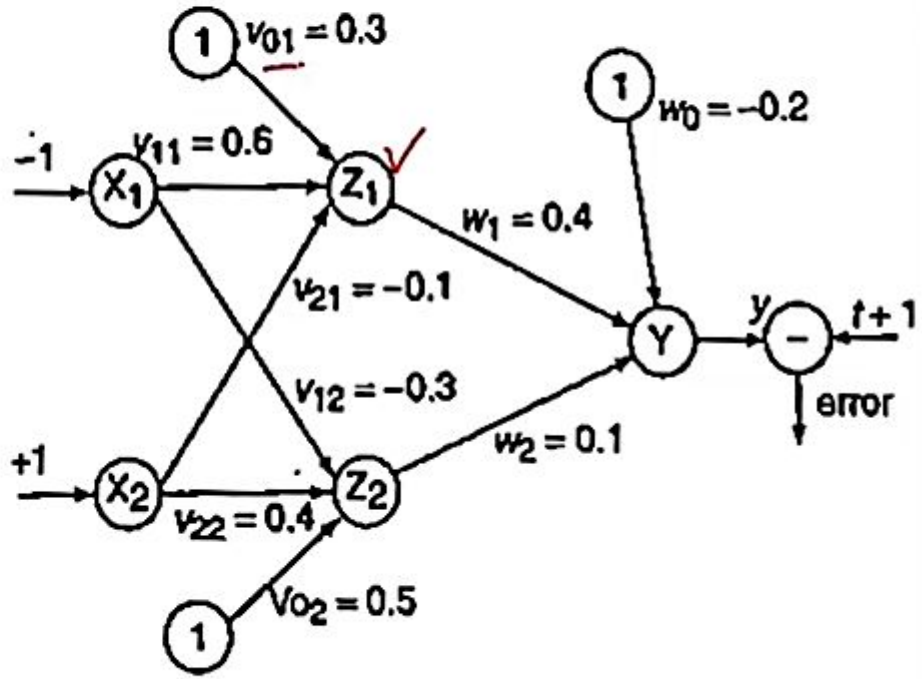
Activation function used is binary sigmoidal activation function and is given by

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

Given the input sample $[x_1, x_2] = [-1, 1]$ and target $t = 1$:

- Calculate the net input: For z_1 layer

$$z_{in1} = v_{01} + x_1 v_{11} + x_2 v_{21}$$
$$= 0.3 + (-1) \times 0.6 + 1 \times -0.1 = -0.4$$



How to Find New Weights – Back Propagation Algorithm

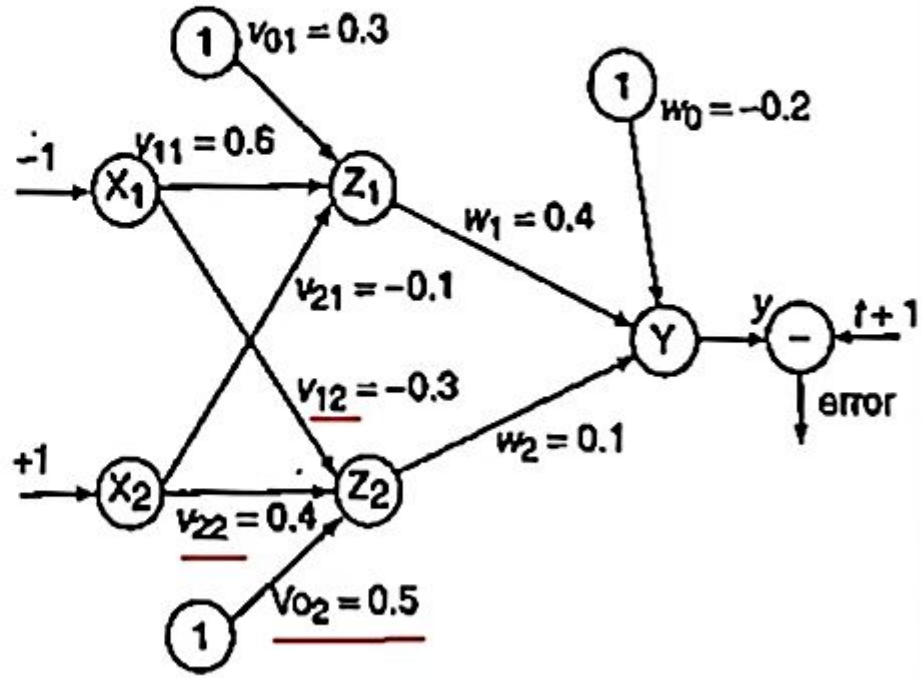
For z_2 layer

$$z_{in2} = v_{02} + x_1 v_{12} + x_2 v_{22}$$
$$= 0.5 + (-1) \times -0.3 + 1 \times 0.4 = 1.2$$

Applying activation to calculate the output, we obtain

$$z_1 = f(z_{in1}) = \frac{1 - e^{-z_{in1}}}{1 + e^{-z_{in1}}} = \frac{1 - e^{0.4}}{1 + e^{0.4}} = -0.1974$$

$$z_2 = f(z_{in2}) = \frac{1 - e^{-z_{in1}}}{1 + e^{-z_{in2}}} = \frac{1 - e^{-1.2}}{1 + e^{-1.2}} = 0.537$$



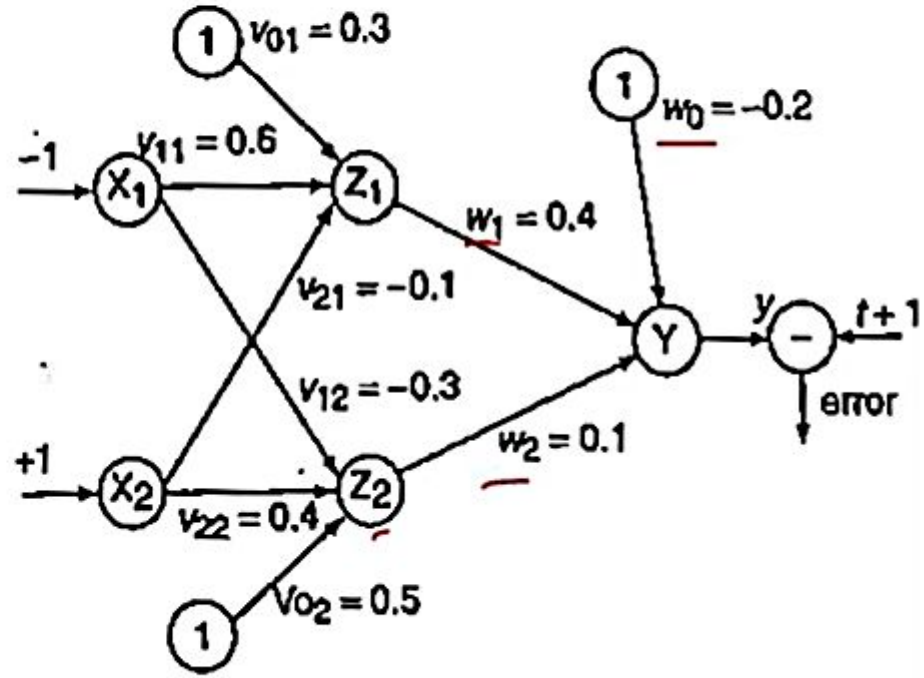
How to Find New Weights – Back Propagation Algorithm

Calculate the net input entering the output layer.
For y layer

$$y_{in} = w_0 + z_1 w_1 + z_2 w_2$$
$$= -0.2 + (-0.1974) \times 0.4 + 0.537 \times 0.1$$
$$= -0.22526$$

Applying activations to calculate the output, we obtain

$$y = f(y_{in}) = \frac{1 - e^{-y_{in}}}{1 + e^{-y_{in}}} = \frac{1 - e^{0.22526}}{1 + e^{0.22526}} = -0.1122$$

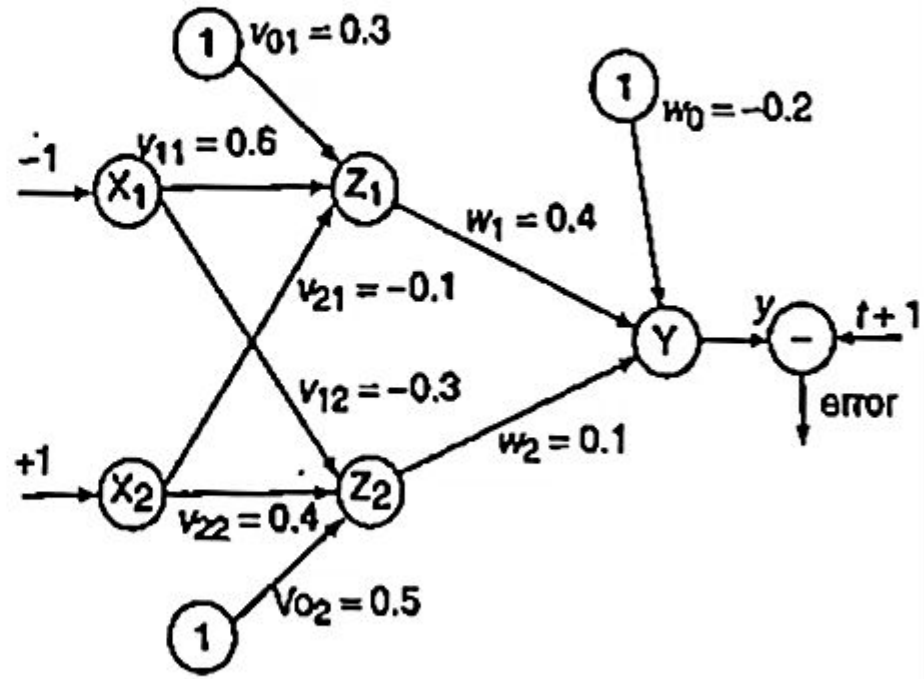


How to Find New Weights – Back Propagation Algorithm

Compute the error portion δ_k:

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

$$f'(x) = \frac{\lambda}{2}[1 + f(x)][1 - f(x)]$$



Now

$$f'(y_{in}) = 0.5[1 + f(y_{in})]\,[1 - f(y_{in})]$$
$$= 0.5[1 - 0.1122][1 + 0.1122] = 0.4937$$

How to Find New Weights – Back Propagation Algorithm

This implies

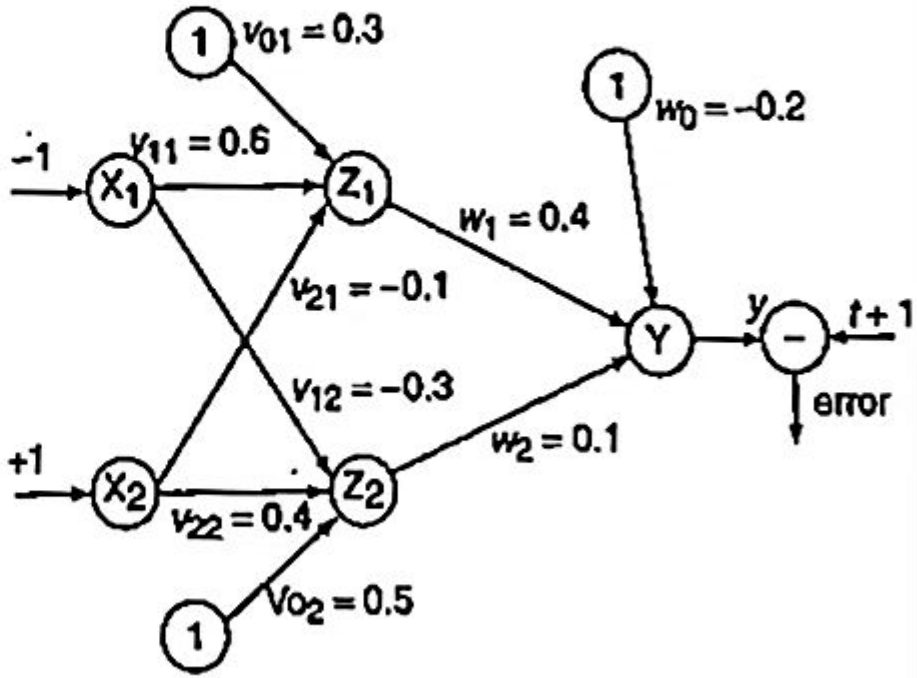
$$\delta_1 = (1 + 0.1122)\,(0.4937) = 0.5491$$

Find the changes in weights between hidden and output layer:

$$\Delta w_1 = \alpha \delta_1 z_1 = 0.25 \times 0.5491 \times -0.1974$$
$$= -0.0271$$

$$\Delta w_2 = \alpha \delta_1 z_2 = 0.25 \times 0.5491 \times 0.537 = 0.0737$$

$$\Delta w_0 = \alpha \delta_1 = 0.25 \times 0.5491 = 0.1373$$



How to Find New Weights – Back Propagation Algorithm

Compute the error portion δ_j between input and hidden layer ($j = 1$ to 2):

$$\delta_j = \delta_{inj} f'(z_{inj})$$

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

$\delta_{inj} = \delta_1 w_{j1}$ [$\because$ only one output neuron]

$\Rightarrow \delta_{in1} = \delta_1 w_{11} = 0.5491 \times 0.4 = 0.21964$

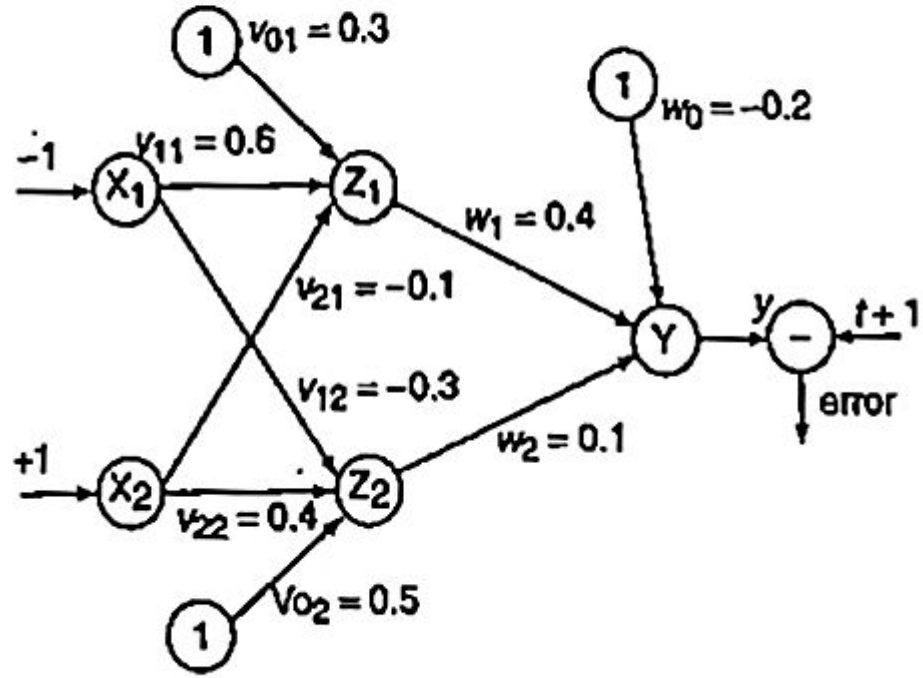
$\Rightarrow \delta_{in2} = \delta_1 w_{21} = 0.5491 \times 0.1 = 0.05491$

Error, $\delta_1 = \delta_{in1} f'(z_{in1}) = 0.21964 \times 0.5$

$\times (1 + 0.1974)(1 - 0.1974) = 0.1056$

Error, $\delta_2 = \delta_{in2} f'(z_{in2}) = 0.05491 \times 0.5$

$\times (1 - 0.537)(1 + 0.537) = 0.0195$



How to Find New Weights – Back Propagation Algorithm

Now find the changes in weights between input and hidden layer:

$$\Delta v_{11} = \alpha \delta_1 x_1 = 0.25 \times 0.1056 \times -1 = -0.0264$$

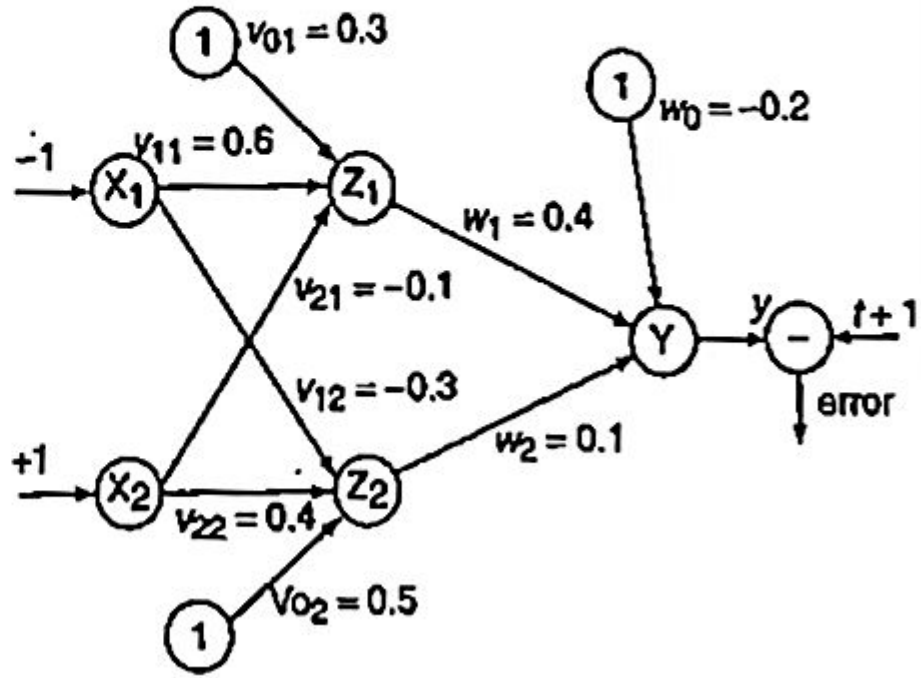
$$\Delta v_{21} = \alpha \delta_1 x_2 = 0.25 \times 0.1056 \times 1 = 0.0264$$

$$\Delta v_{01} = \alpha \delta_1 = 0.25 \times 0.1056 = 0.0264$$

$$\Delta v_{12} = \alpha \delta_2 x_1 = 0.25 \times 0.0195 \times -1 = -0.0049$$

$$\Delta v_{22} = \alpha \delta_2 x_2 = 0.25 \times 0.0195 \times 1 = 0.0049$$

$$\Delta v_{02} = \alpha \delta_2 = 0.25 \times 0.0195 = 0.0049$$



Thank You