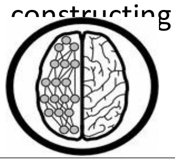
Artificial Neural Network

- An artificial neural network consists of a **pool** of simple processing units which communicate by sending signals to each other over **a large number of weighted connections**.
- An artificial neural network (ANN) is either a hardware implementation or a computer program which strives to simulate the information processing capabilities of its biological exemplar. **ANNs are typically composed of a great number of interconnected artificial neurons**. The artificial neurons are simplified models of their biological counterparts.
- ANN is a technique for solving problems by constructing software that **works like our brains**.

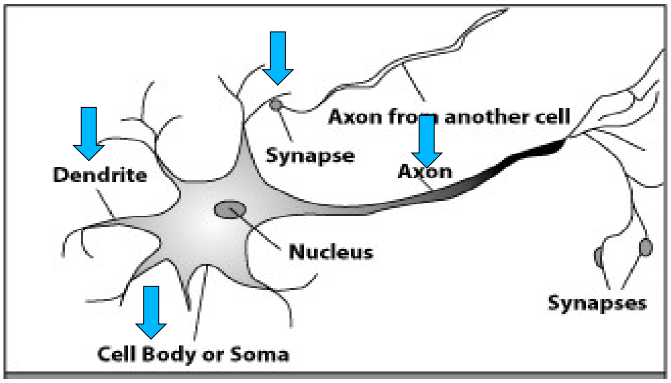


How do our brains work?

- The Brain is a massively **parallel** information processing system.
- Our brains are a huge network of processing elements. A typical brain contains a network of **10 billion neurons**.

How do our brains work?

- A processing element



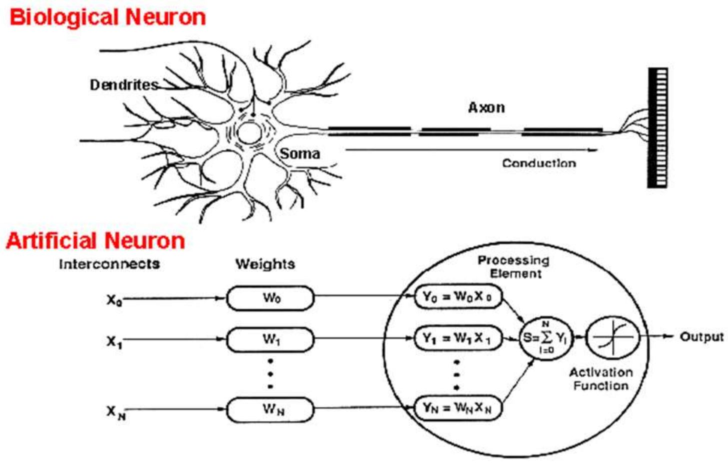
Dendrites: Input

Cell body: Processor

Synaptic: Link

Axon: Output

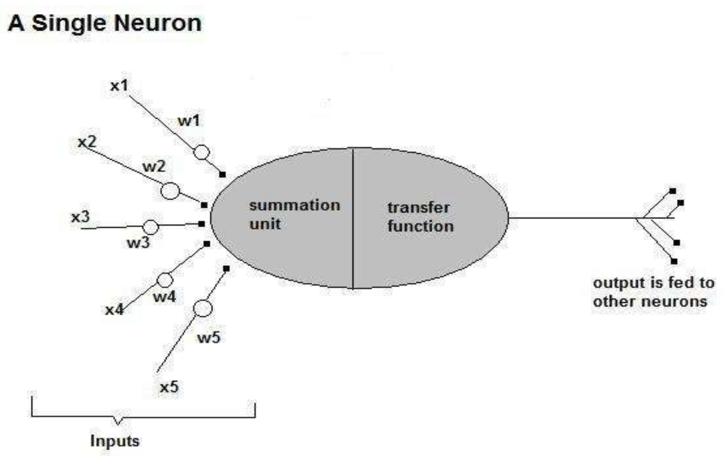
How do ANNs work?



An artificial neuron is an **imitation** of a human neuron

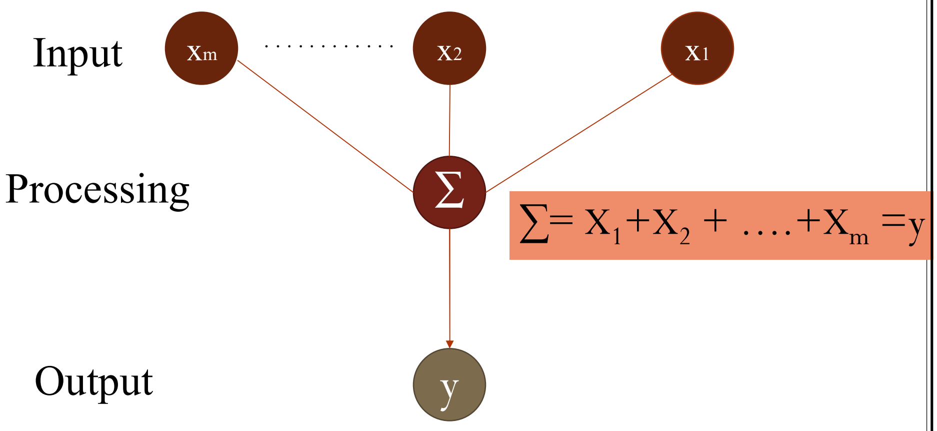
How do ANNs work?

- Now, let us have a look at the model of an artificial neuron.

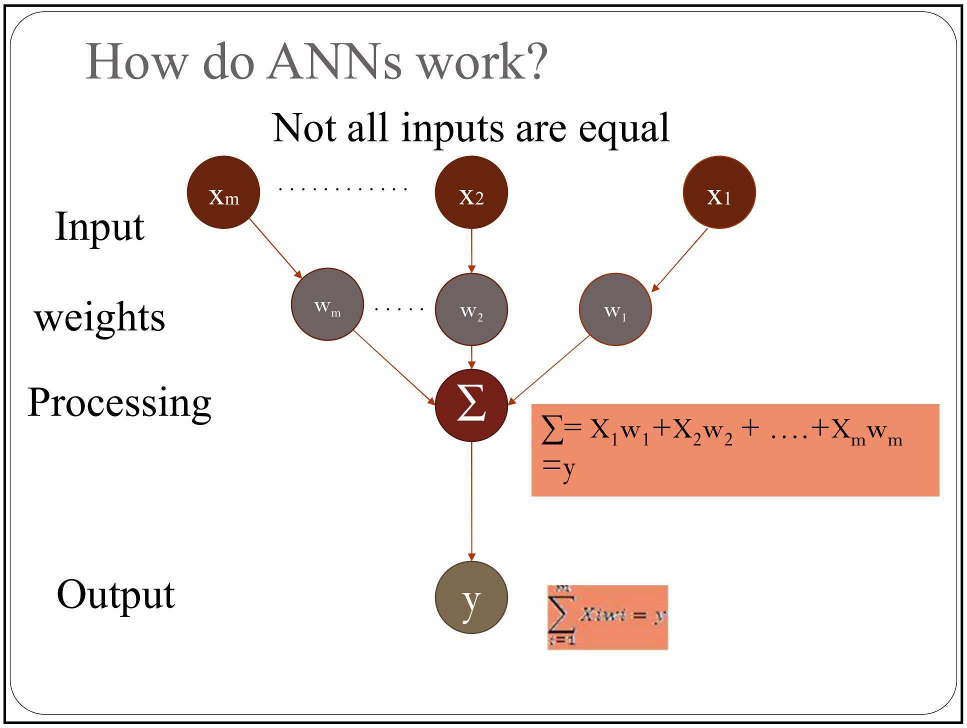


A **Perceptron** is the simplest type of artificial neural network and is the fundamental building block of deep learning.

How do ANNs work?



$$\sum = X_1 + X_2 + \ldots. + X_m = y$$



How do ANNs work?

Not all inputs are equal

Input

weights

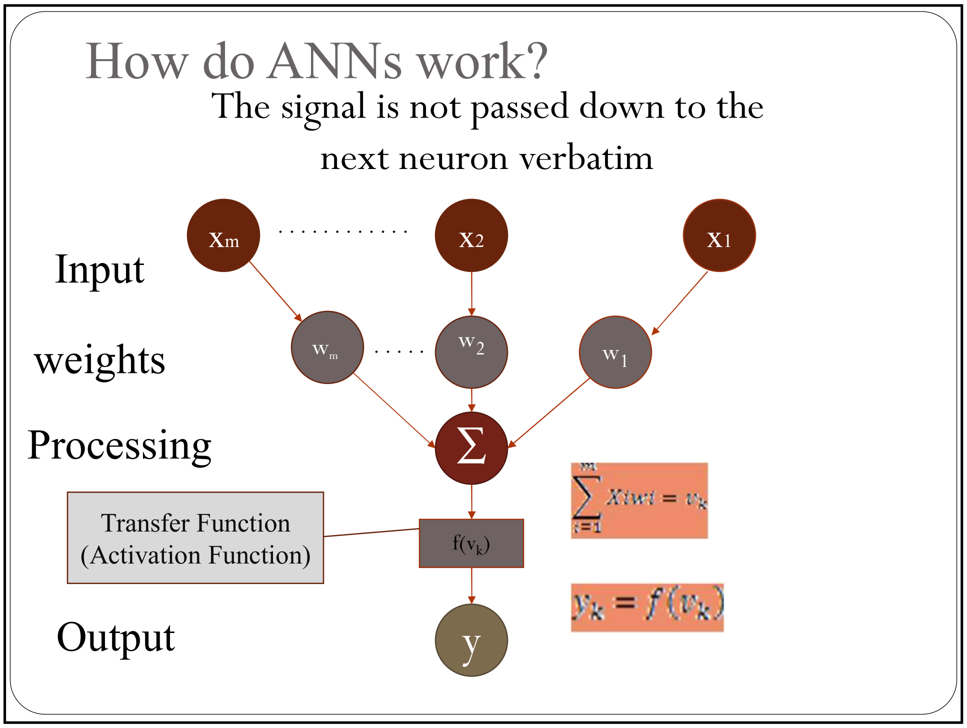
Processing

Output

$\sum = X_1 w_1 + X_2 w_2 + \ldots . + X_m w_m = y$

$\sum_{i=1}^{m} Xiwi = y$

7



How do ANNs work?

The signal is not passed down to the next neuron verbatim

Input

weights

Processing

Transfer Function (Activation Function)

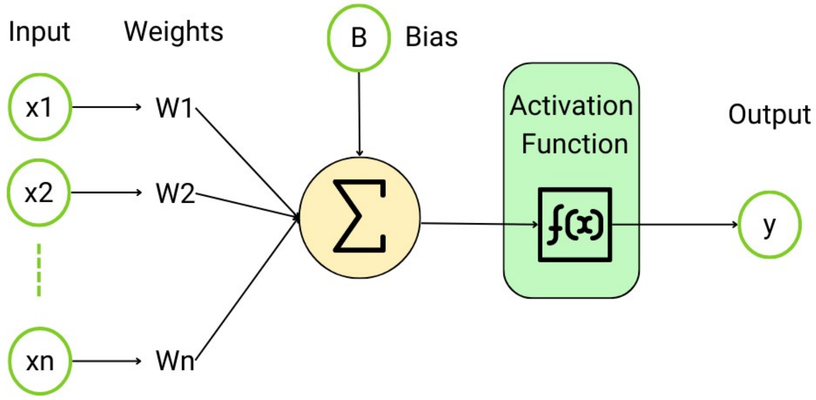
Output

$\sum_{i=1}^{m} Xiwi = v_k$

$y_k = f(v_k)$

8

The output is a function of the input, that is affected by the weights, and the transfer functions



<u>Basic Structure & Components:</u>

1. **Neuron (Node, Perceptron)** – The fundamental unit of an ANN that processes inputs and applies an activation function.
2. **Input Layer** – The first layer that receives data and passes it to hidden layers.
3. **Hidden Layer(s)** – Intermediate layers **between input and output** layers that perform complex computations.
4. **Output Layer** – The **final layer** that provides the ANN’s result or prediction.
5. **Weights** – Parameters that define the strength of connections between neurons.
6. **Bias** – An additional parameter added to neurons to shift the activation function and improve flexibility.

Neuron (Node, Perceptron) in an Artificial Neural Network (ANN)

A **neuron**, also known as a **node** or **perceptron**, is the fundamental unit of an artificial neural network (ANN). It is inspired by biological neurons in the human brain and is responsible for processing input data and passing signals through the network.

Structure of a Neuron in ANN: Inputs, Weights, Summation Function, Bias, Activation Function, and Output.

Types of Neurons

1. **Perceptron (Single-layer neuron)**
    1. The simplest form of a neuron that uses a step activation function (threshold-based).
    2. Used in **binary classification** problems.
    3. Cannot handle **non-linearly separable** data.
2. **Multi-layer Neurons (Used in Deep Learning)**
    1. Have multiple layers of neurons (hidden layers).
    2. Use complex activation functions to model non-linear relationships.

Mathematical Operations & Learning:

1. **Activation Function –** A function applied to neuron inputs to introduce non-linearity (e.g., ReLU, Sigmoid, Tanh).
2. **Forward Propagation –** The process of passing input through the network to generate an output.
3. **Loss Function (Cost Function) –** A mathematical function that measures the error between predicted and actual output.
4. **Backpropagation –** The algorithm used to adjust weights by minimizing errors through gradient descent.
5. **Gradient Descent –** An optimization technique to adjust weights and minimize loss iteratively.
6. **Learning Rate –** A hyperparameter that determines how much weights update in each iteration.
7. **Epoch –** One complete cycle of training over the entire dataset.
8. **Batch Size –** The number of samples processed before updating weights during training.

Types & Variations:

1. **Feedforward Neural Network (FNN)** – A network where information moves in one direction, from input to output.
2. **Convolutional Neural Network (CNN)** – A network designed for image processing with convolutional layers.
3. **Recurrent Neural Network (RNN)** – A network with connections that allow information to persist, used for sequential data.
4. **Long Short-Term Memory (LSTM)** – A special type of RNN that overcomes vanishing gradient problems.
5. **Autoencoder** – A type of ANN used for unsupervised learning and feature extraction.

Learning by trial-and-error

Continuous process of:

- Trial:

  Processing an input to produce an output (In terms of ANN: Compute the output function of a given input)

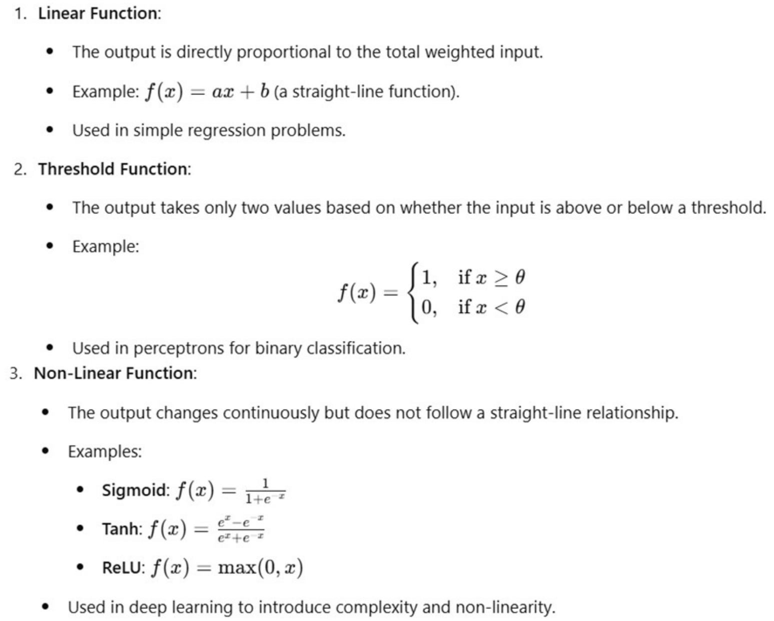
- Evaluate:

  Evaluating this output by comparing the actual output with the expected output.

- Adjust:

  Adjust the weights.

Transfer Functions



1. **Linear Function**:
   - The output is directly proportional to the total weighted input.
   - Example: $f(x) = ax + b$ (a straight-line function).
   - Used in simple regression problems.
2. **Threshold Function**:
   - The output takes only two values based on whether the input is above or below a threshold.
   - Example:

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

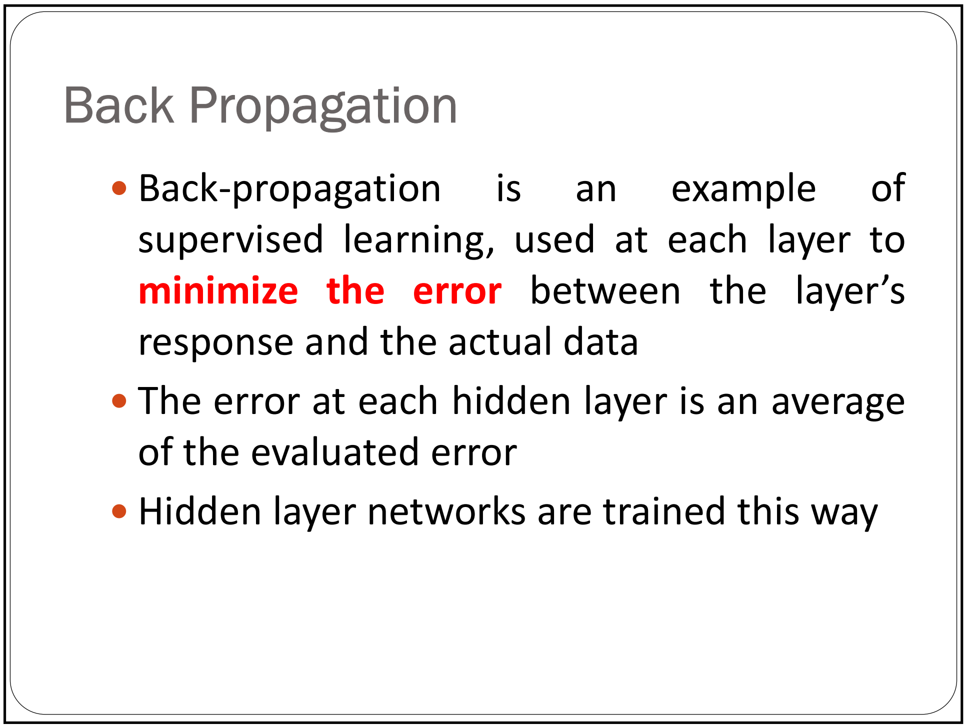
   - Used in perceptrons for binary classification.
3. **Non-Linear Function**:
   - The output changes continuously but does not follow a straight-line relationship.
   - Examples:
     - **Sigmoid**: $f(x) = \frac{1}{1+e^{-x}}$
     - **Tanh**: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
     - **ReLU**: $f(x) = \max(0, x)$
   - Used in deep learning to introduce complexity and non-linearity.

Error Estimation

- The **root mean square error (RMSE)** is a frequently-used measure of the differences between values predicted by a model or an **estimator** and the values actually observed from the thing being modeled or estimated

Weights Adjusting

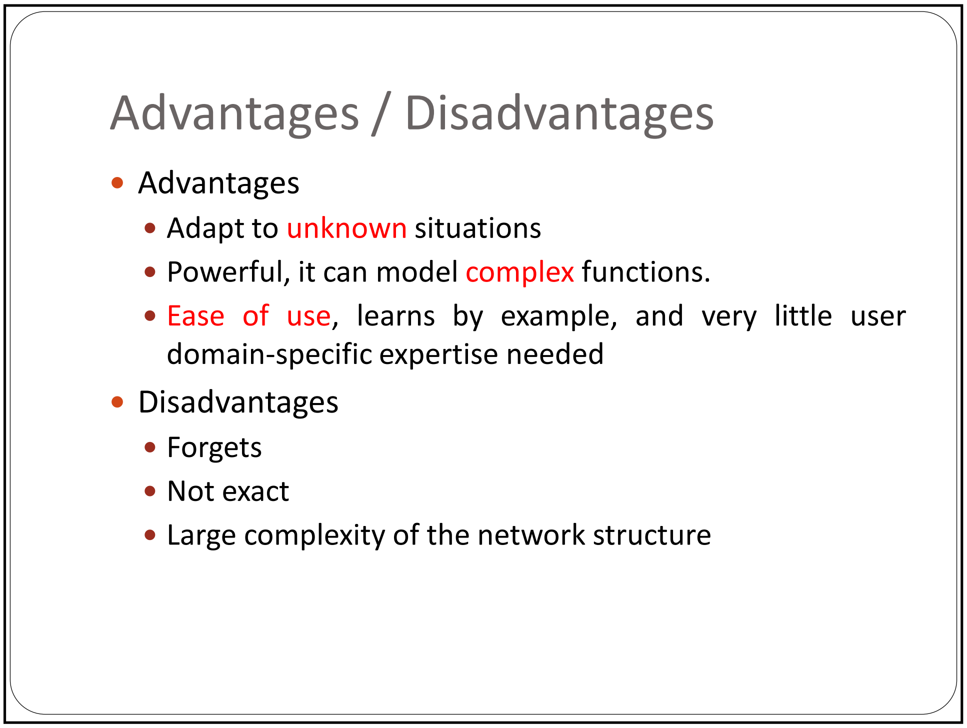
- After each iteration, weights should be **adjusted** to **minimize** the error.
  - – All possible weights
  - – Back propagation



Back Propagation

- Back-propagation is an example of supervised learning, used at each layer to **minimize the error** between the layer's response and the actual data
- The error at each hidden layer is an average of the evaluated error
- Hidden layer networks are trained this way

17



Advantages / Disadvantages

- Advantages
  - Adapt to unknown situations
  - Powerful, it can model complex functions.
  - Ease of use, learns by example, and very little user domain-specific expertise needed
- Disadvantages
  - Forgets
  - Not exact
  - Large complexity of the network structure

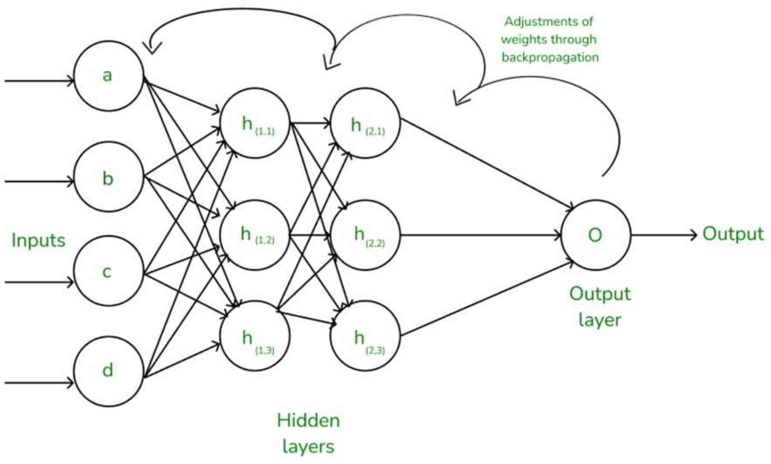
18

Conclusion

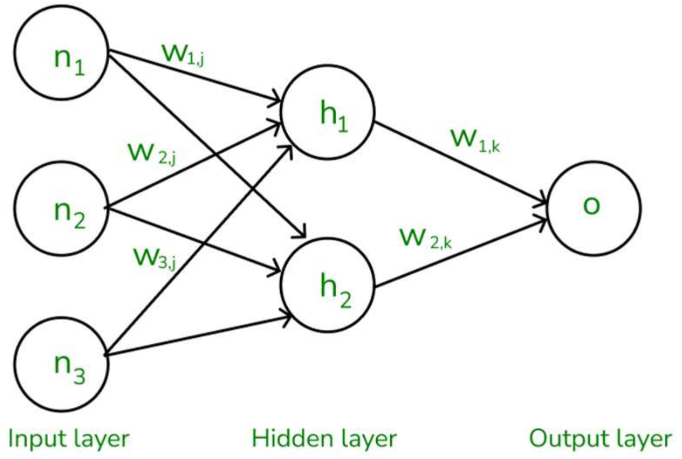
- Artificial Neural Networks are an **imitation** of the **biological neural networks**, but much simpler ones.
- The computing would have a lot to gain from neural networks. Their ability to learn by example makes them **very flexible and powerful**. Furthermore, there is a need to devise an algorithm to perform a specific task.
- Neural networks also contribute to the area of research such as **neurology and psychology**. They are regularly used to model parts of living organizations and to investigate the **internal mechanisms of the brain**.
- Many factors affect the **performance** of ANNs, such as the transfer functions, size of training sample, network topology, weights adjusting algorithm, …

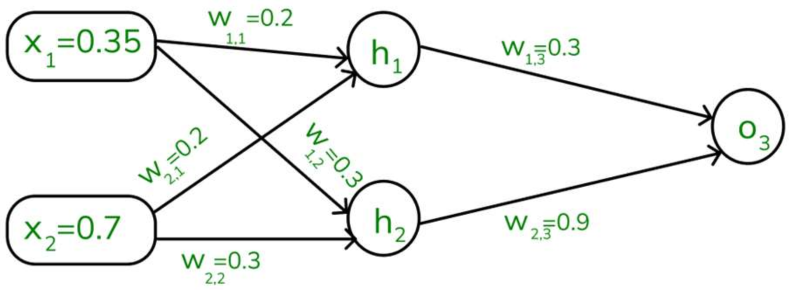
What is Backpropagation?

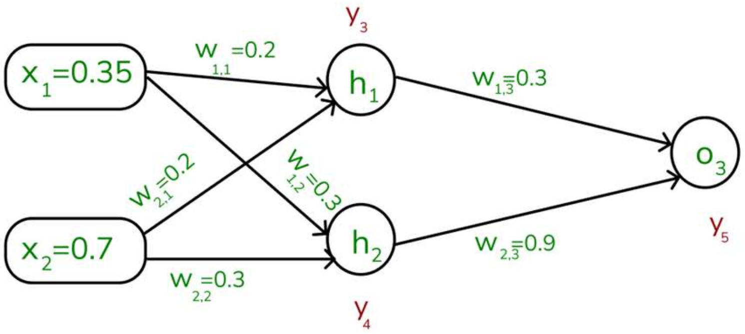
Backpropagation is a technique used in deep learning to train artificial neural networks particularly feed-forward networks. It works iteratively to adjust weights and bias to minimize the cost function. It often uses optimization algorithms like gradient descent or stochastic gradient descent.

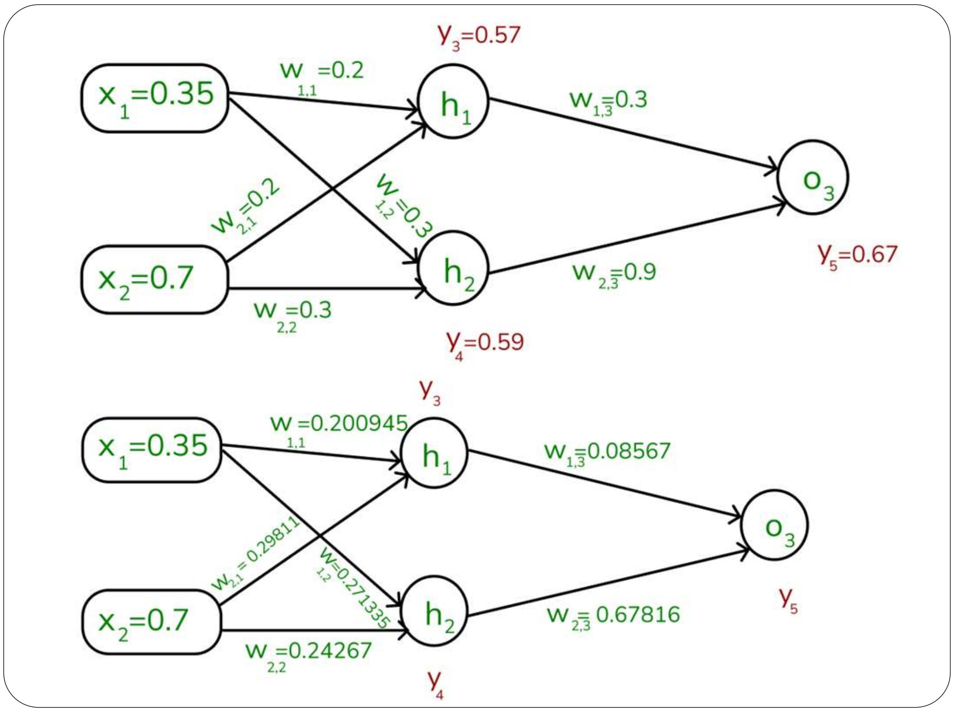


Example


n_1
n_2
n_3
$w_{1,j}$
$w_{2,j}$
$w_{3,j}$
h_1
h_2
$w_{1,k}$
$w_{2,k}$
o
Input layer
Hidden layer
Output layer


$x_1=0.35$
$x_2=0.7$
$w_{1,1}=0.2$
$w_{2,1}=0.2$
$w_{1,2}=0.3$
$w_{2,2}=0.3$
h_1
h_2
$w_{1,3}=0.3$
$w_{2,3}=0.9$
o_3


y_3
$x_1=0.35$
$x_2=0.7$
$w_{1,1}=0.2$
$w_{2,1}=0.2$
$w_{1,2}=0.3$
$w_{2,2}=0.3$
h_1
h_2
$w_{1,3}=0.3$
$w_{2,3}=0.9$
o_3
y_5
y_4



23

Thank You

24