Artificial Neural Networks

Objectives

After completing this module, you should be able to:

Understand Artificial Neural Networks

Understand the need of Activation Functions

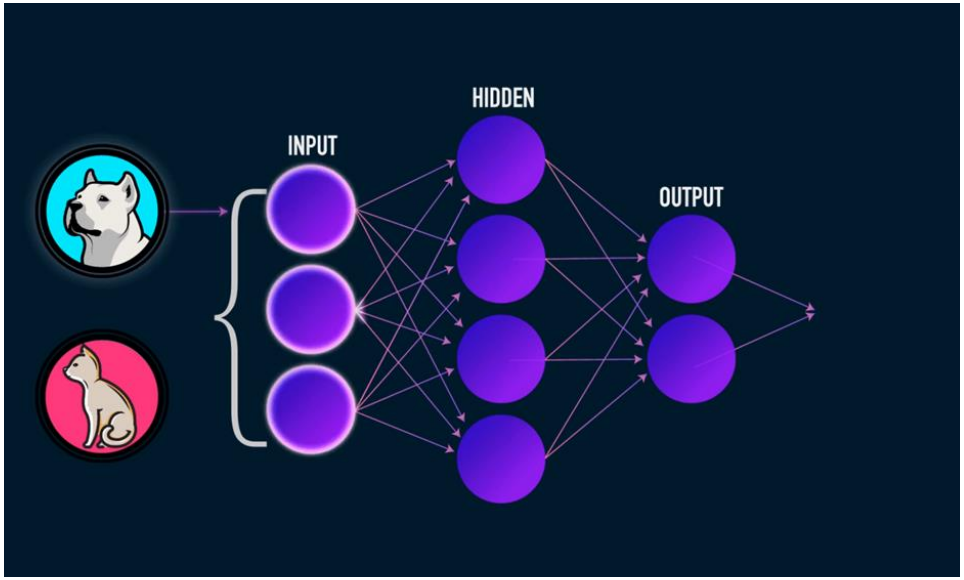
Neural Network Architecture

Types of Neural Networks

Applications, Advantages and Limitations

Artificial Neural Networks

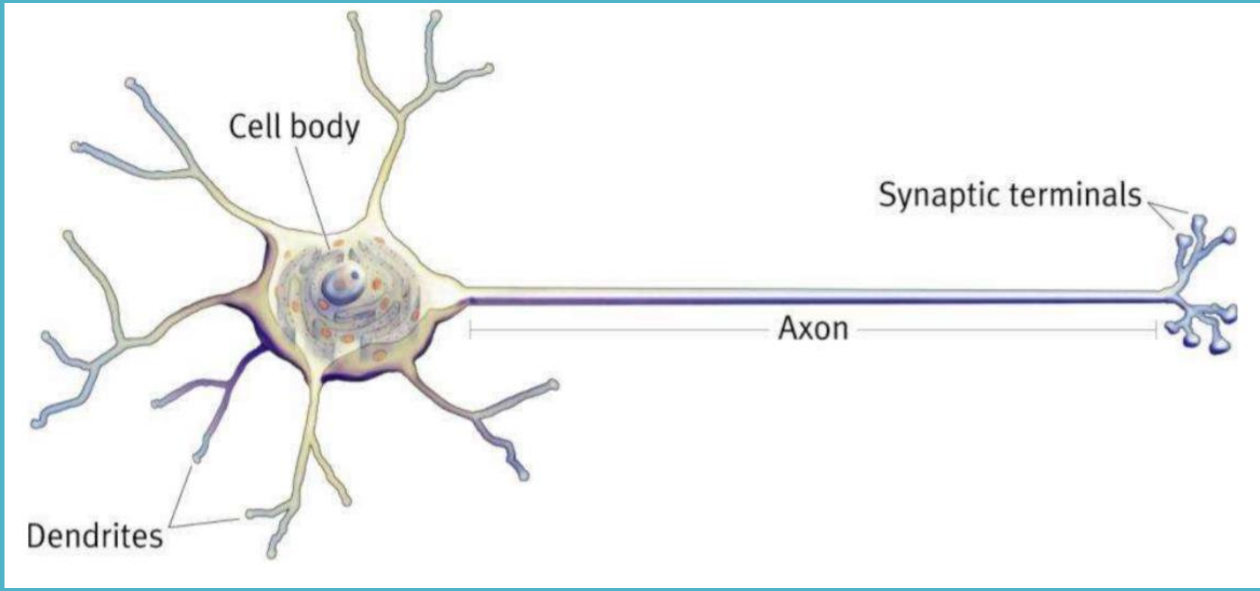
Artificial Neural Networks, in general — is a biologically inspired network of artificial neurons configured to perform specific tasks.



3

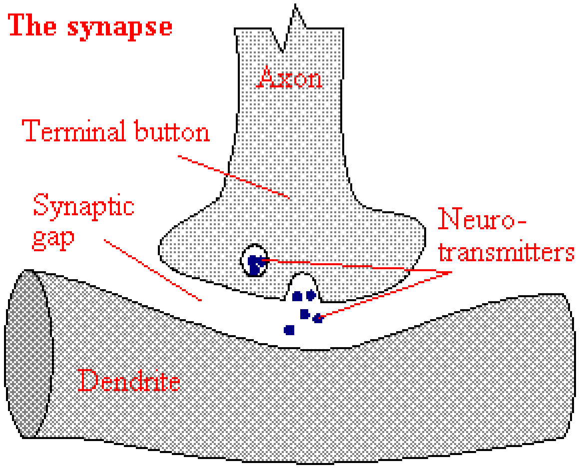
Neural networks learn from examples

- No requirement of an explicit description of the problem.
- No need for a programmer.
- The neural computer adapts itself during a training period, based on examples of similar problems even without a desired solution to each problem. After sufficient training the neural computer is able to relate the problem data to the solutions, inputs to outputs, and it is then able to offer a viable solution to a brand new problem.
- Able to generalize or to handle incomplete data.



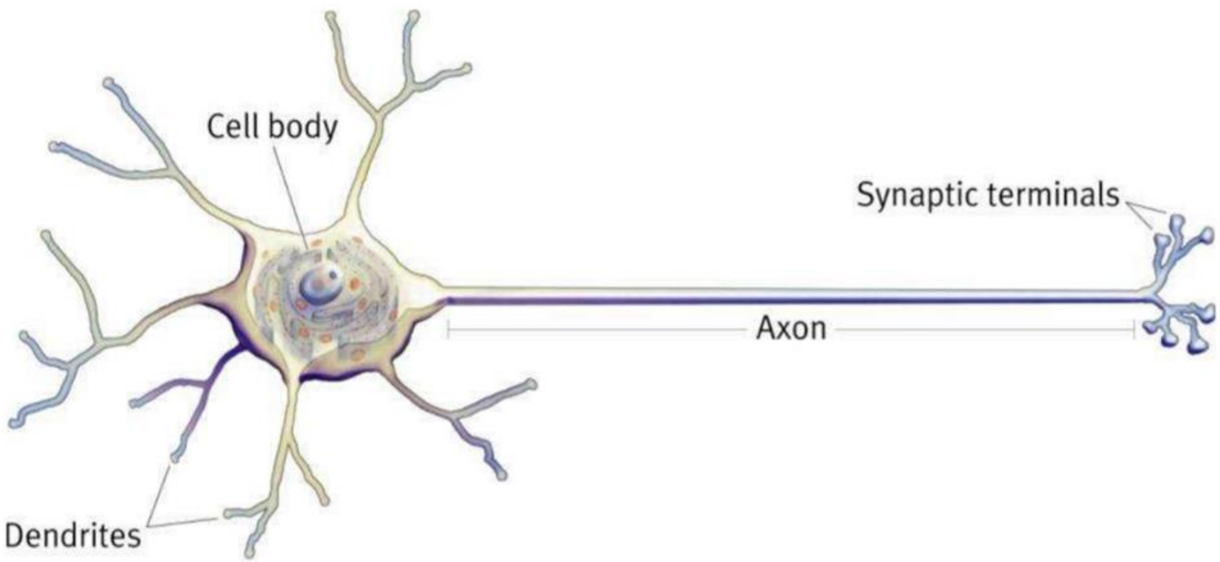
INTRODUCTION TO NEURON

Information exchange b/w Neurons



Neurons

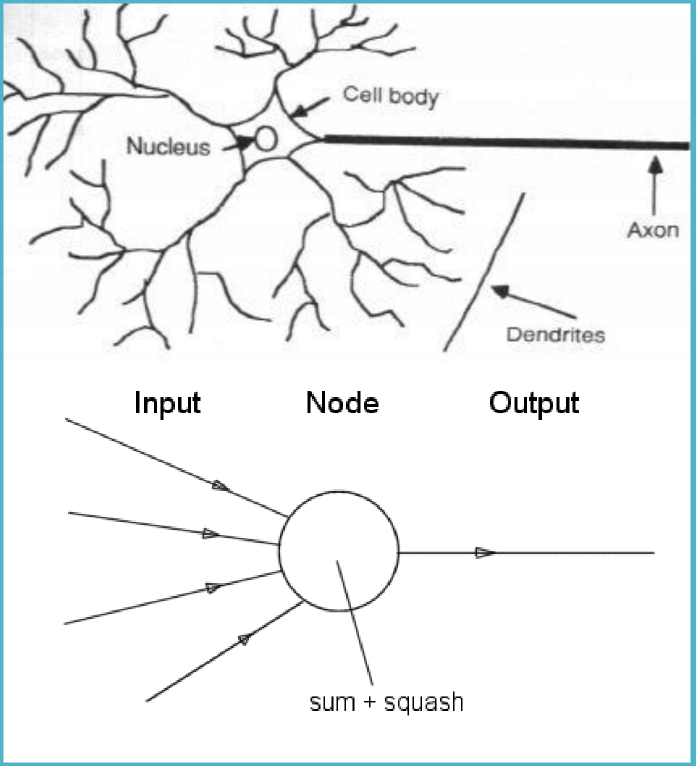
Neuron



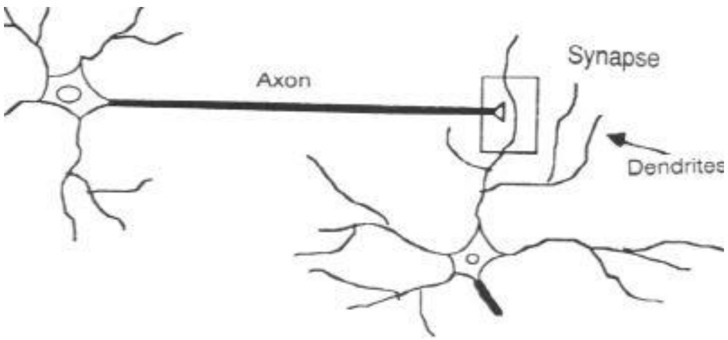
Neuron in Brain

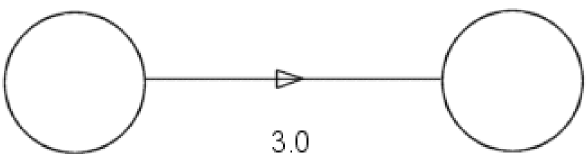
A Neuron/Node in Artificial Neural Network

Neuron vs. Node



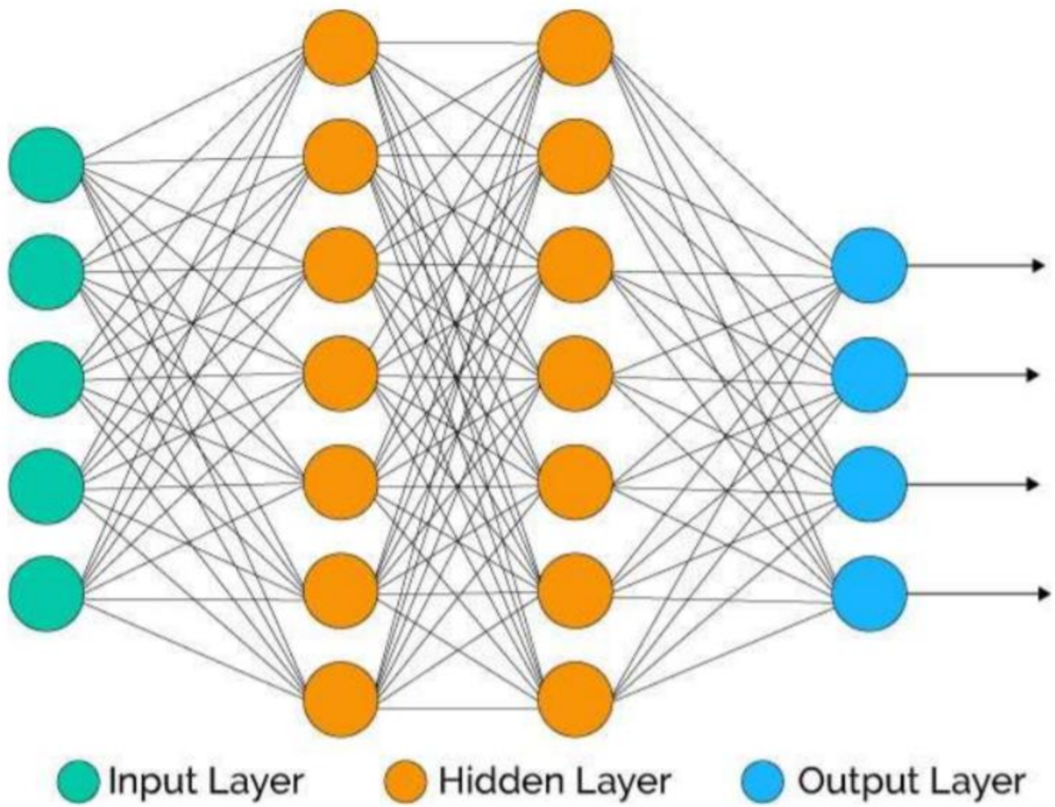
Synapse vs. weight





Architecture of Neural Network

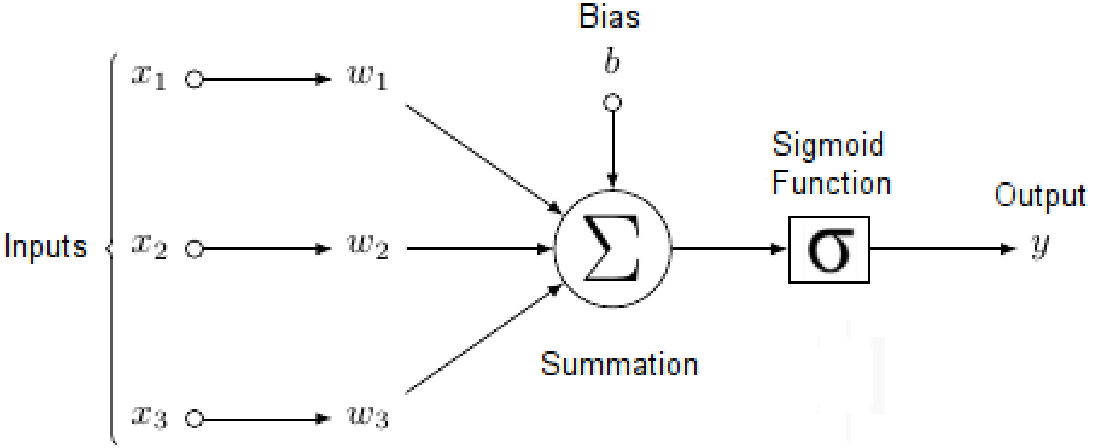
A typical neural network contains a large number of artificial neurons called units arranged in a series of layers.



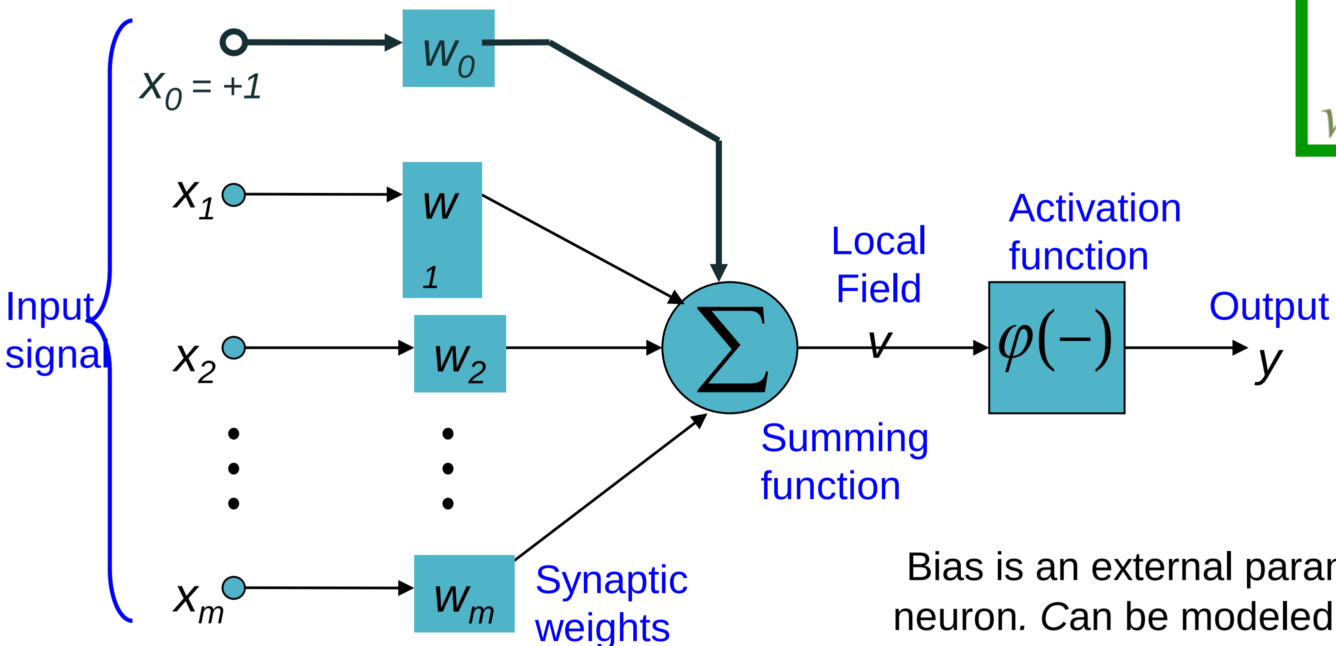
Elements of Neural Networks

1. No Computation on Input layer Neurons

2. Two process on every computational Neuron.
- Weighted Sum
- Activation Function



Bias as extra input



$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$

Bias is an external parameter of the neuron. Can be modeled by adding an extra input.

Activation Functions

What are Activation Functions?

Activation Functions *introduce non-linear properties to Neural Network.*

There are 2 types of activation functions –

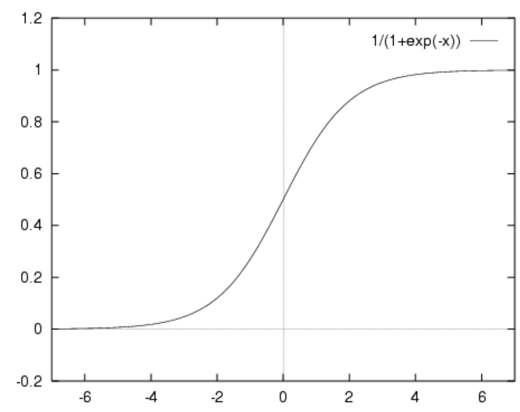
1. Linear Activation Functions

2. Non Linear Activation Functions

Sigmoid Function

$$\boldsymbol{\sigma(z) = \frac{1}{1 + e^{-z}}}$$

- The function is differentiable. That means, we can find the slope of the sigmoid curve at any two points.
- The function is monotonic but function's derivative is not.
- The logistic sigmoid function can cause a neural network to get stuck at the training time.
- The softmax function is a more generalized logistic activation function which is used for multiclass classification.



Hyperbolic Tangent function

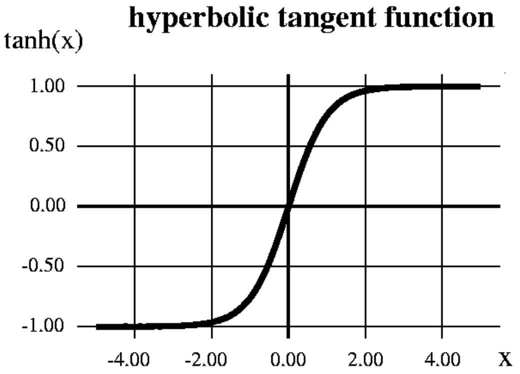
$$\boldsymbol{tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}}$$

The function is **differentiable**.

The function is **monotonic** while its **derivative is not**.

The tanh function is mainly used classification between two classes.

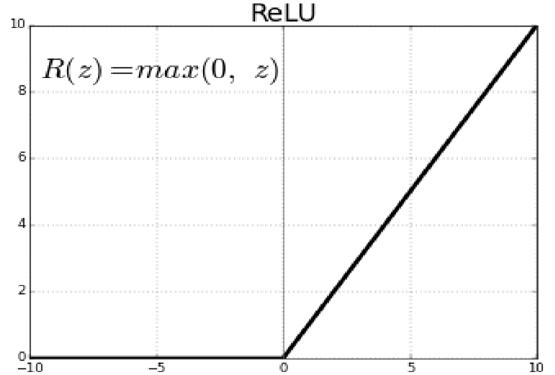
output is zero centered because its range in between -1 to 1 i.e -1 < output < 1 . Hence optimization is *easier* in this method hence in practice it is always preferred over Sigmoid function .



ReLu- Rectified Linear units

$$f(x) = \max(x, 0)$$

The function and its derivative **both are monotonic**.



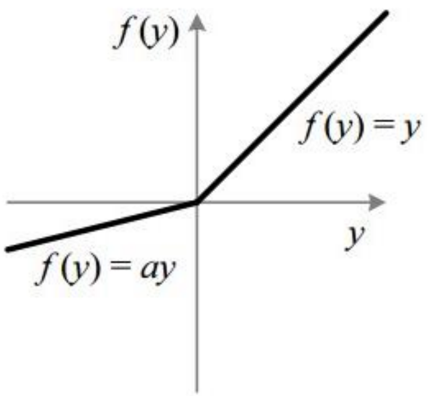
It should only be used within Hidden layers of a Neural Network Model.

Hence for output layers we should use a **Softmax** function for a Classification problem to compute the probabilites for the classes , and for a regression problem it should simply use a **linear** function.

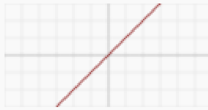
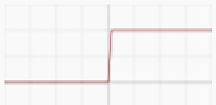
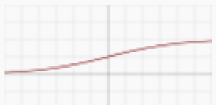
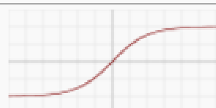
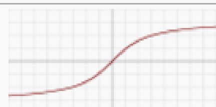
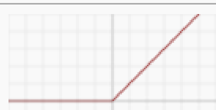
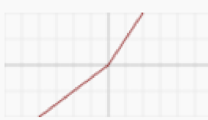
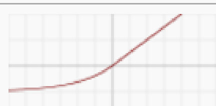
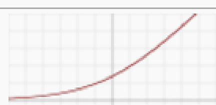
Leaky ReLu

$$f(x) = \begin{Bmatrix} x & if\ x < 0 \\ 0.01x & otherwise \end{Bmatrix}$$

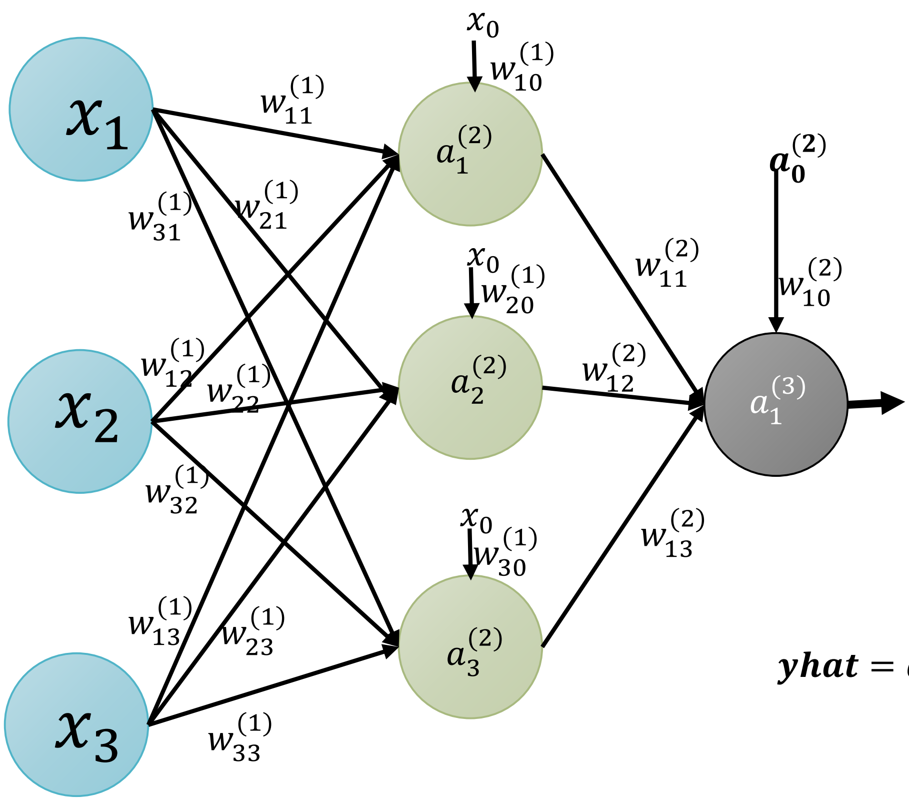
- The problem with ReLu is that some gradients can be fragile during training and can die. It can cause a weight update which will makes it never activate on any data point again. Simply saying that ReLu could result in Dead Neurons.



- To fix the problem of dying neurons another modification was introduced called Leaky ReLu. It introduces a small slope to keep the updates alive.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)[2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)[3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

Mathematical Modelling of Neural Network



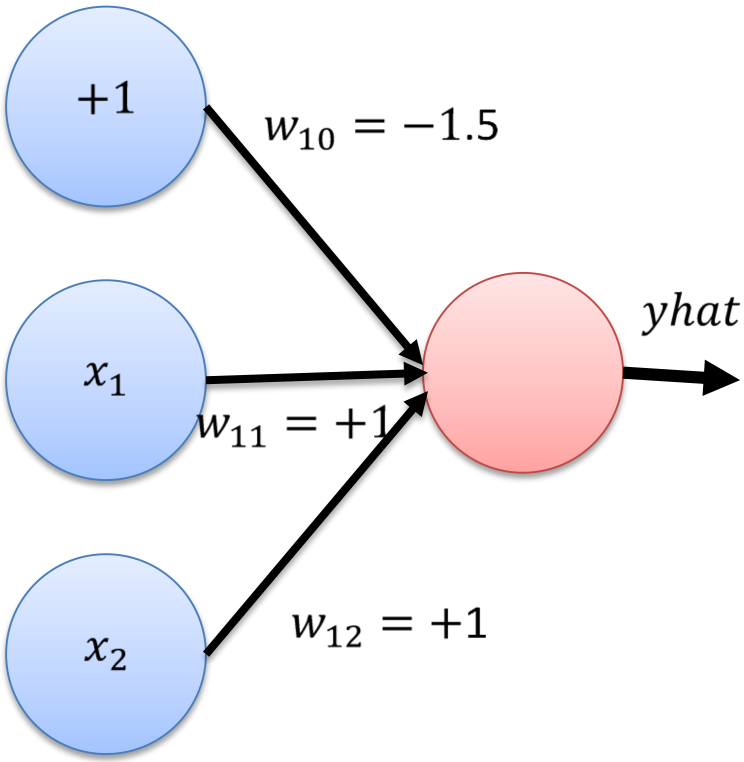
$$a_1^{(2)} = g\{w_{10}^{(1)}x_0 + w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3\}$$

$$a_2^{(2)} = g\{w_{20}^{(1)}x_0 + w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3\}$$

$$a_3^{(2)} = g\{w_{30}^{(1)}x_0 + w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3\}$$

$$\boldsymbol{yhat = a_1^{(3)} = g\{w_{10}^{(2)}a_0^{(2)} + w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)}\}}$$

AND FUNCTION

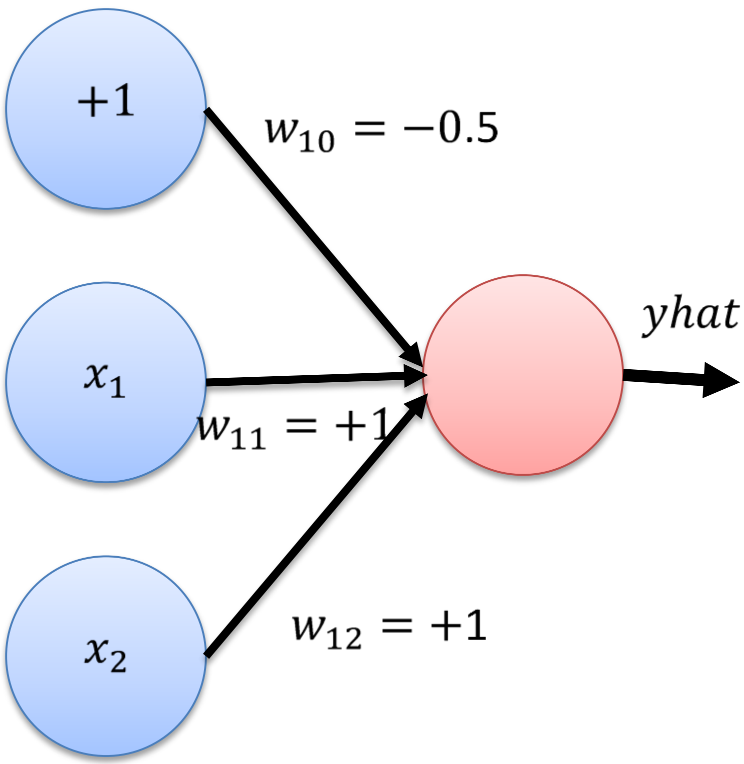


AND FUNCTION

x_1	x_2	**Yhat=F(x,w)**
0	0	yhat=g{1*-1.5+0*1+0*1}=g{-1.5}=0
0	1	yhat=g{1*-1.5+0*1+1*1}=g{-0.5}=0
1	0	yhat=g{1*-1.5+1*1+0*1}=g{-0.5}=0
1	1	yhat=g{1*-1.5+1*1+1*1}=g{+0.5}=1

22

OR FUNCTION



23

OR FUNCTION

x_1	x_2	**Yhat=F(x,w)**
0	0	yhat=g{1*-0.5+0*1+0*1}=g{-0.5}=0
0	1	yhat=g{1*-0.5+0*1+1*1}=g{+0.5}=1
1	0	yhat=g{1*-0.5+1*1+0*1}=g{+0.5}=1
1	1	yhat=g{1*-0.5+1*1+1*1}=g{+1.5}=1

Neural Network Architecture

Network Architecture

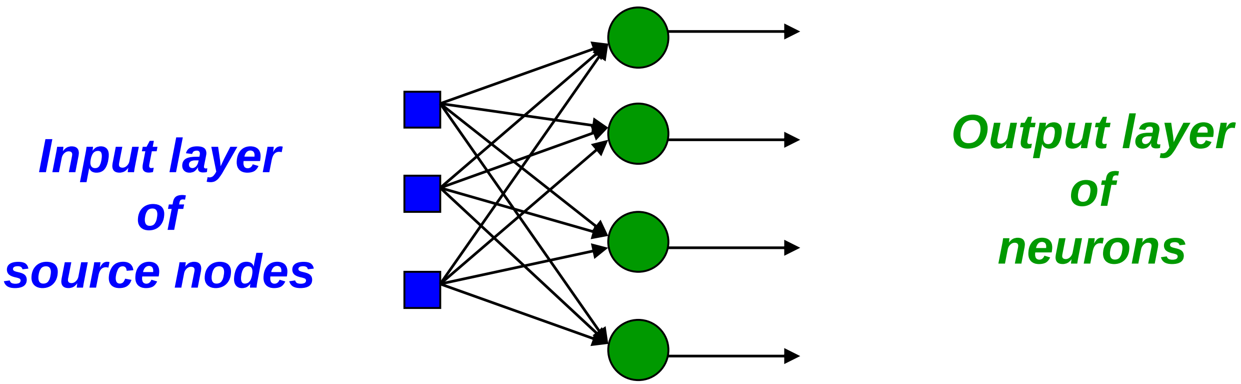
Three basic different classes of network architectures

- Single-layer feed-forward Neural Networks
- Multi-layer feed-forward Neural Networks
- Recurrent Neural Networks

The architecture of a neural network is linked with the learning algorithm used to train

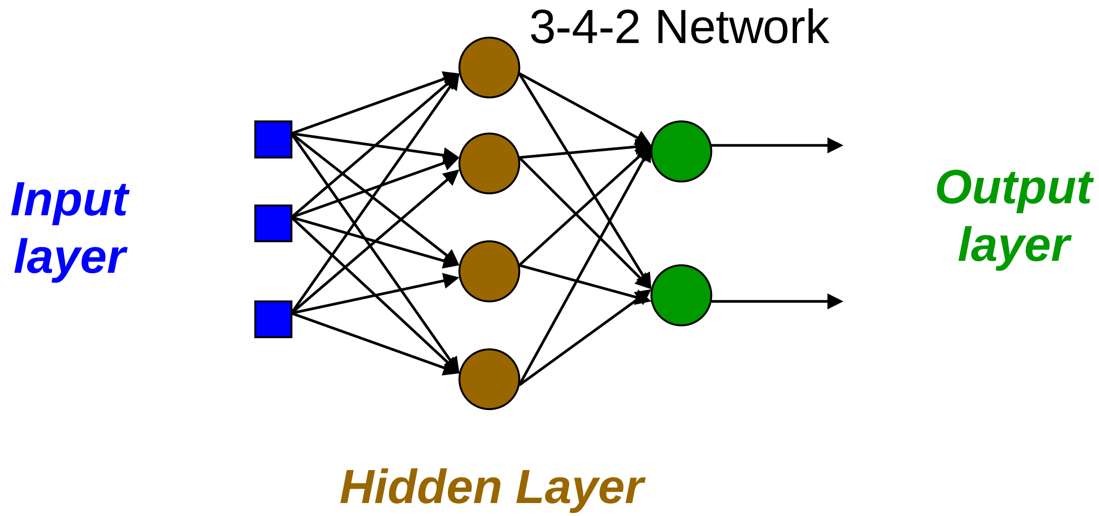
Single Layer Feed-forward

It is simplest kind of neural network. It consists only single layer of output nodes. The inputs are fed directly via series of weights.



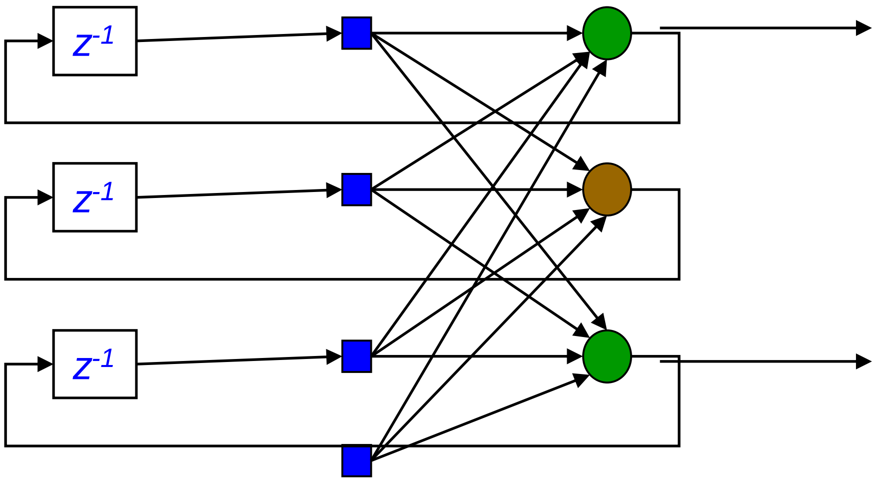
Multi Layer Feed-forward

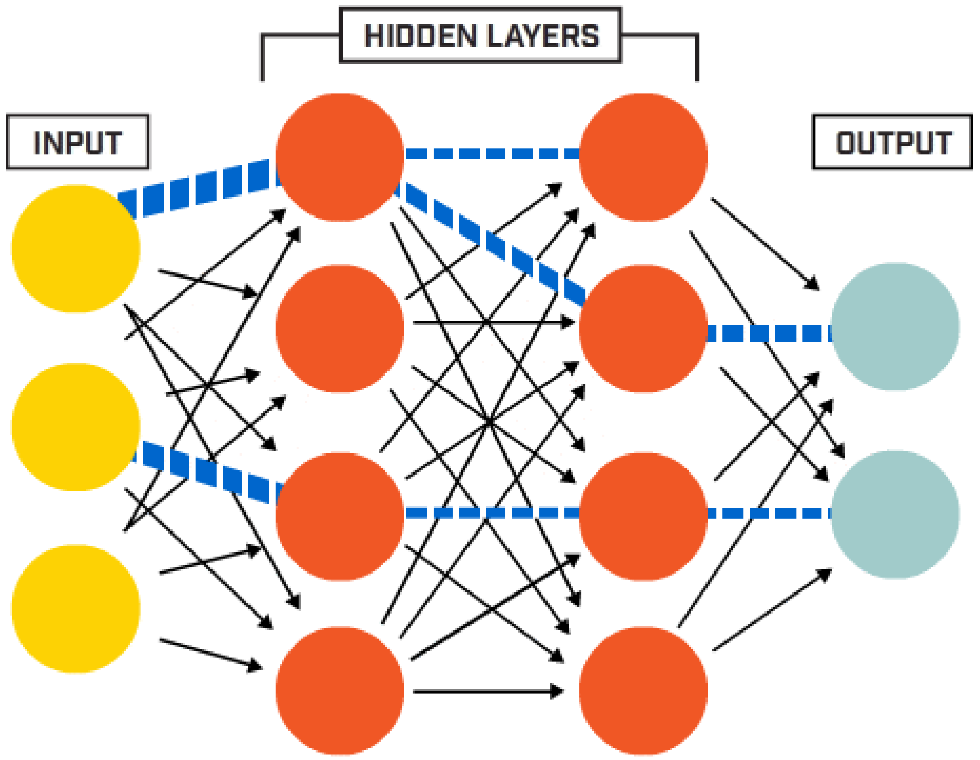
A multilayer feed forward neural network is an interconnection of perceptron in which data and calculation flow in a single direction from input to the output.
In given diagram we can see there are three input nodes connected with one hidden layer (4 hidden nodes) and then hidden layer nodes connected to output nodes



Recurrent Network

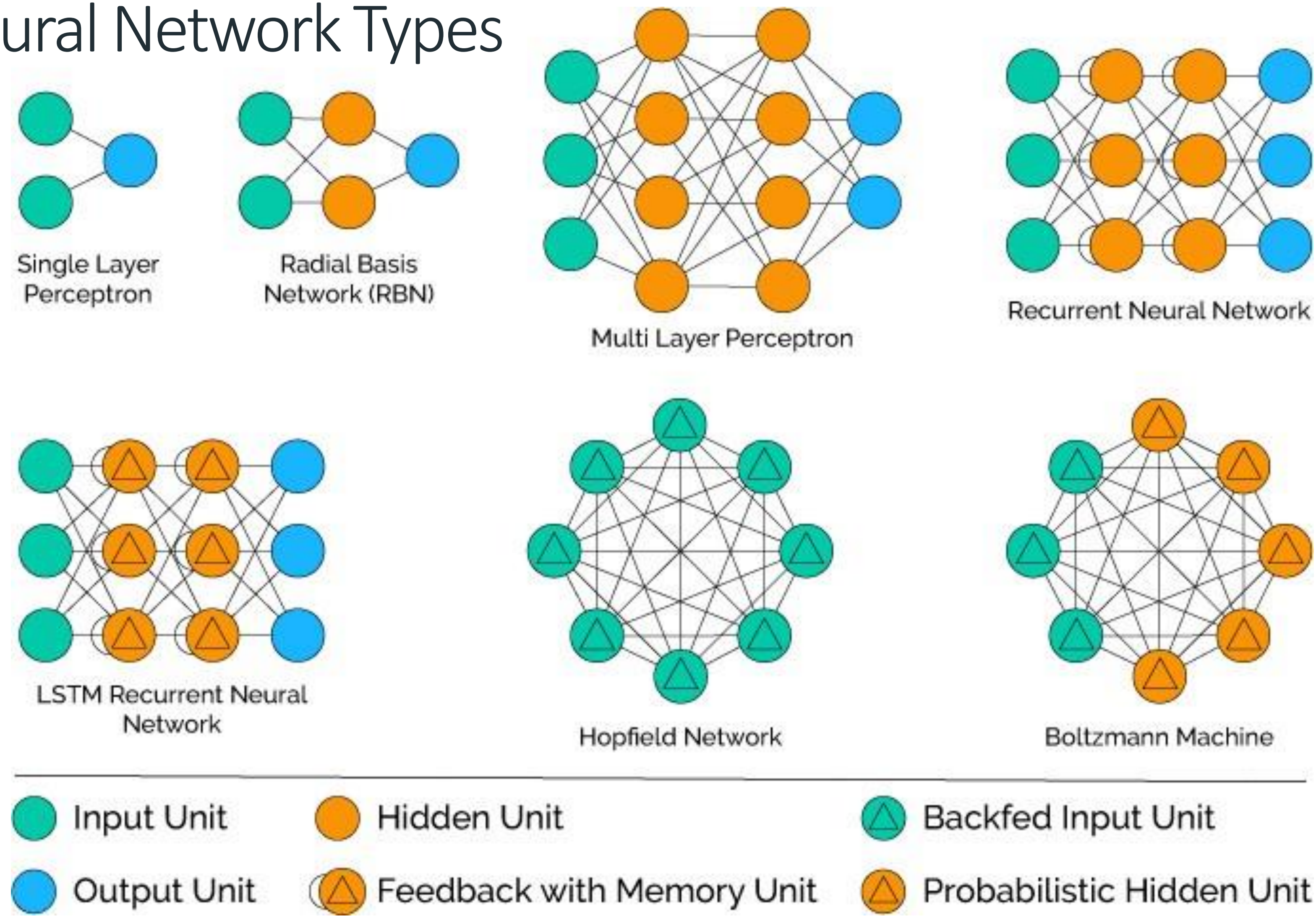
Recurrent networks are distinguished from feed forward network by that feedback loops connected to their past decision, ingesting their own output moment after moment as input.





Network Architecture

Neural Network Types



Neural Network for Machine Learning

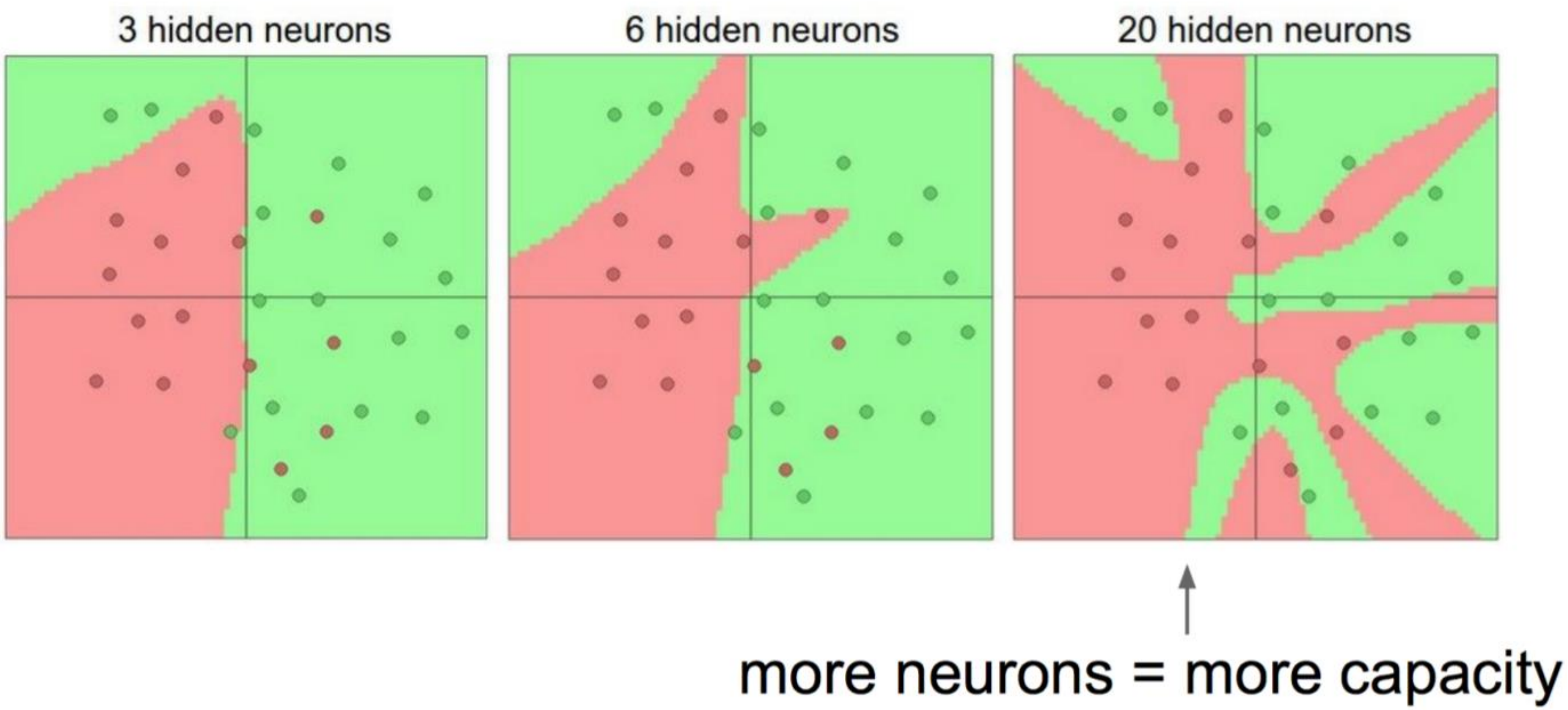
- Multilayer Perceptron (supervised classification)
- Back Propagation Network (supervised classification)
- Hopfield Network (for pattern association)
- Deep Neural Networks (unsupervised clustering)

Neural Network for Deep Learning

- Recurrent neural network
- Multi-layer perceptrons (MLP)
- Convolutional neural networks
- Recursive neural networks

- Deep belief networks
- Convolutional deep belief networks
- Self-Organizing Maps
- Deep Boltzmann machines
- Stacked de-noising auto-encoders

How many hidden Neurons?



Decision Boundaries in Two Dimensions

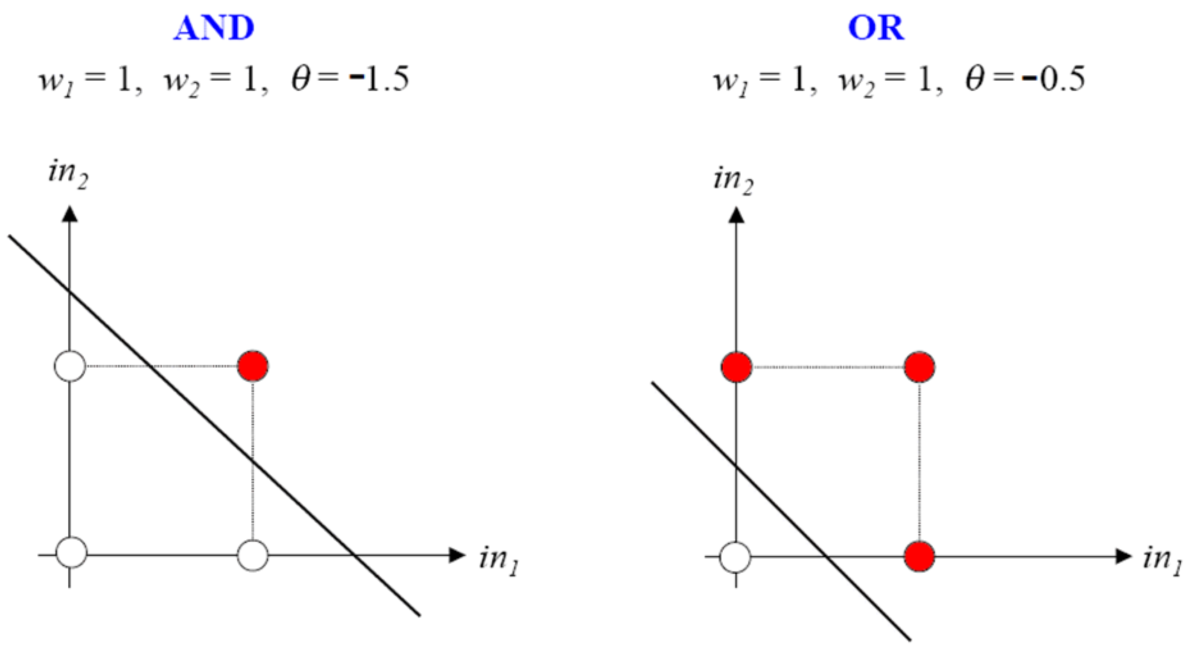
For simple logic gate problems, it is easy to visualize what the neural network is doing. It is forming ***decision boundaries*** between classes. Remember, the network output is:

The decision boundary (between *out* = 0 and *out* = 1) is at

$$w_1x_1 + w_2x_2 - b = 0$$

The **decision boundary** is the line that separates the area where y = 0 and where y = 1. It is created by our current model.

Decision Boundaries in Two Dimensions



36

Decision Hyperplanes and Linear Separability

If we have two inputs, then the weights define a decision boundary that is a one dimensional straight line in the two dimensional ***input space*** of possible input values

If we have n inputs, the weights define a decision boundary that is an n-1 dimensional ***hyperplane*** in the n dimensional input space:

$$w_1 in_1 + w_2 in_2 + \dots + w_n in_n - \theta = 0$$

Advantages of Neural Networks

- A neural network can perform tasks that a linear program can not.
- When an element of the neural network fails, it can continue without any problem by their parallel nature.
- A neural network learns and does not need to be reprogrammed.
- It can be implemented in any application.
- It can be performed without any problem.

Limitations of Neural Networks

- The neural network needs the training to operate.
- The architecture of a neural network is different from the architecture of microprocessors, therefore, needs to be emulated.
- Requires high processing time for large neural networks.

Applications of Neural Networks

Application	Architecture / Algorithm	Activation Function
Process modeling and control	Radial Basis Network	Radial Basis
Machine Diagnostics	Multilayer Perceptron	Tan- Sigmoid Function
Portfolio Management	Classification Supervised Algorithm	Tan- Sigmoid Function
Target Recognition	Modular Neural Network	Tan- Sigmoid Function
Medical Diagnosis	Multilayer Perceptron	Tan- Sigmoid Function
Credit Rating	Logistic Discriminant Analysis with ANN, Support Vector Machine	Logistic function

Summary

This module covered the following topics:

Artificial Neural Networks

Introduction to Neurons

Single Neuron Model

Activation Functions

Neural Network Architecture

Types of Neural Networks

Applications, Advantages and Limitations

Thank you