Artificial Neural Network

Unit-1 - Introduction to Neural Networks

History of neural network

- 1940s: Theoretical Foundations
- 1943: Warren McCulloch & Walter Pitts proposed a mathematical model of how neurons work using electrical circuits — the first artificial neural network model.
- 1949: Donald Hebb published The Organization of Behavior, introducing Hebbian Learning — “neurons that fire together, wire together.”

1950s: Early Experiments

- 1950s: Computers became advanced enough to simulate neural networks.
- Nathanial Rochester (IBM) attempted the first simulation — failed.
- 1959: Bernard Widrow & Marcian Hoff (Stanford) developed:
- ADALINE: for binary pattern recognition.
- MADALINE: first neural net used in real-world application (eliminating phone line echoes).

1960s: Learning Rules & Setbacks

- 1962: Widrow & Hoff developed a learning rule to adjust weights by distributing errors across neurons.
- Despite early promise, the von Neumann architecture took over computing.
- A paper incorrectly claimed multi-layer neural nets weren’t feasible, and flawed learning functions (non-differentiable) led to a funding winter for neural networks.
- Hype led to disappointment; fears about "thinking machines" added to public skepticism.

1970s: Slow Progress

- 1972: Kohonen and Anderson independently developed similar matrix-based analog networks.

- 1975: First unsupervised multilayer neural network created — marked a turning point.

1980s: Revival of Neural Network Research

- 1982:
- John Hopfield introduced recurrent networks with bidirectional connections.
- Reilly & Cooper developed hybrid networks with multiple learning strategies.
- US-Japan conference sparked competitive interest; Japan launched its Fifth Generation AI initiative.
- 1986:
- Backpropagation rediscovered independently by multiple researchers (notably David Rumelhart), allowing error to be propagated backward through multiple layers.
- Enabled deep learning, but networks were slow learners, requiring many iterations.

Future of Neural Networks

- Performance now heavily depends on hardware acceleration (e.g., GPUs, TPUs).

Emerging tech includes:

- Silicon compilers for neural-specific chips.
- Analog and optical chips (mimicking biological neurons more closely than digital systems).
- Analog signals, despite being "old tech," better reflect how biological neurons function (gradual signals vs binary).

Biological Neurons

The Brain's Neural Network

The human brain contains:

~10 billion neurons

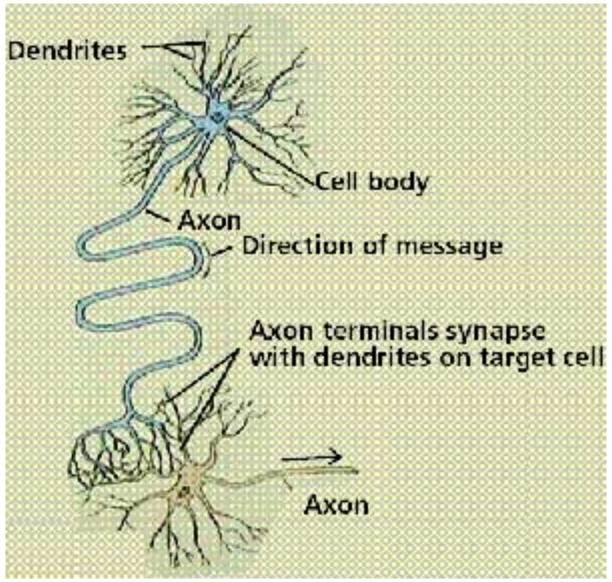
Each connected to ~10,000 other neurons

<u>Neuron structure:</u>

Cell body (soma): the central processing unit

Dendrites: receive inputs

Axons: send outputs to other neurons



How a Neuron Works

- Neurons receive electrochemical signals through dendrites
- If total input exceeds a threshold, the neuron fires
- Signal travels through the axon to other neurons

Neuron firing is binary:

- Fires (on) or does not fire (off)
- No partial or graded firing

Simple Units, Complex Behavior

Brain functions arise from simple units acting together

Each neuron does a weighted sum of inputs

Fires if input exceeds threshold

Despite simplicity, neurons perform complex tasks in unison

Inspiration for Artificial Neural Networks

Artificial neural networks (ANNs) model this behavior:

- Use weighted inputs
- Apply an activation function
- Output is binary or continuous

ANNs are far less complex than the brain but:

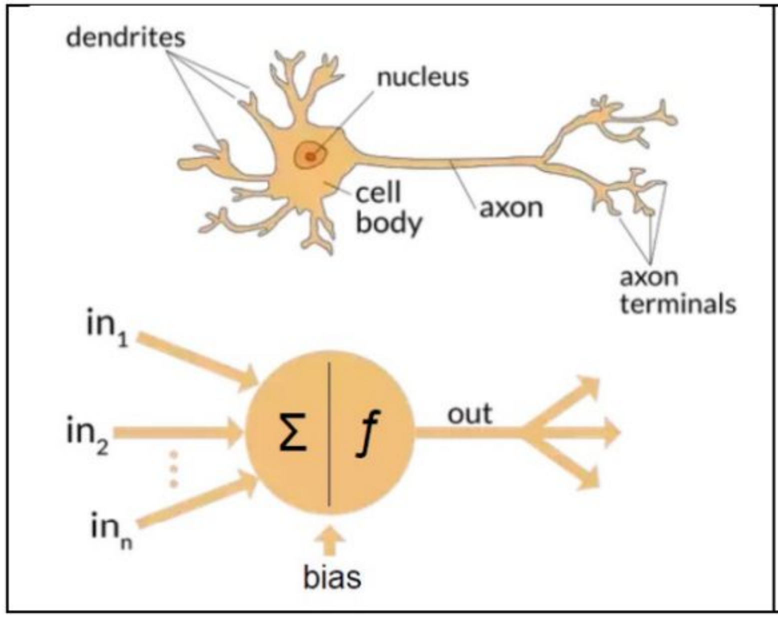
- Good at tasks difficult for traditional computers
- Examples: image recognition, pattern prediction

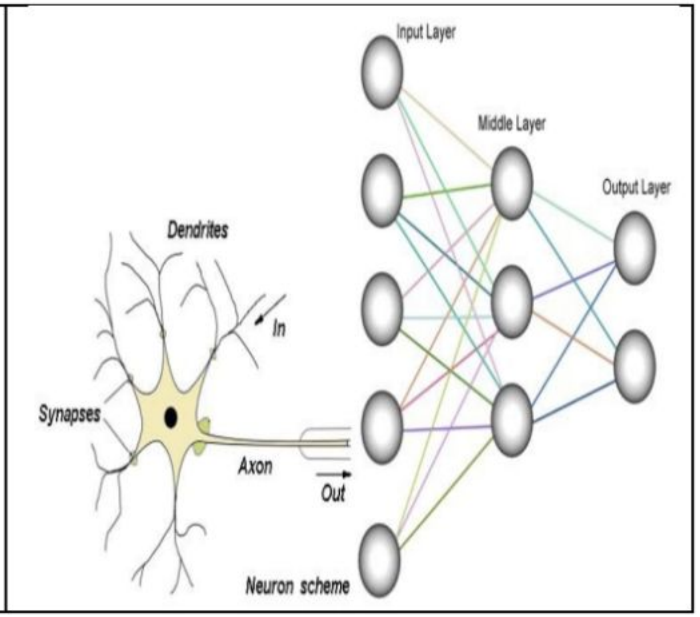
Artificial Neurons

- An artificial neuron is the building block of an artificial neural network.
- Artificial neural networks, like the biological neural network in the human body, have a layered architecture, and each network node (connection point) can process input and forward output to other nodes in the network.
- ANNs are composed of artificial neurons arranged in input, hidden, and output layers, employing a connectionist approach to computation.
- They use nonlinear statistical data modelling techniques to uncover intricate relationships between inputs and outputs. This enables the prediction or classification of complex problems

Artificial Neurons

Biological Neuron	**Artificial Neuron**
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output



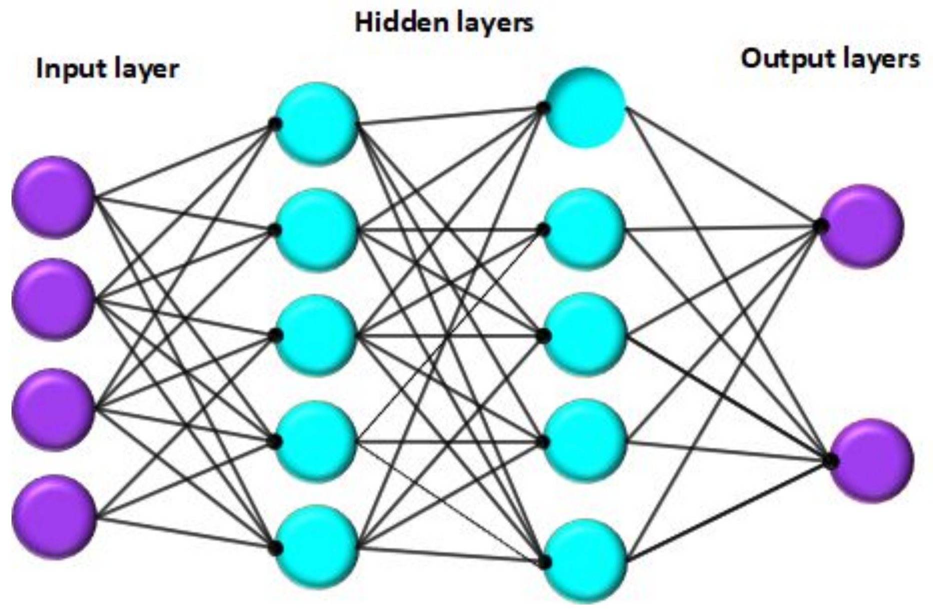


Key Components of Artificial Neurons

- Interconnecting model of neurons: The neuron is the elementary component connected with other neurons to form the network.
- Learning algorithm: This is to train the network, as various learning algorithms are available in the literature to train the model.
- Each layer consists of neurons, and these neurons are connected to other layers.
- Weight is also assigned to each layer, and these weights are changed at each iteration for training purposes

ANN Structure

- An Artificial Neural Network (ANN) is a computational model inspired by the way biological neural networks in the human brain process information.
- ANNs are a key technology in machine learning and artificial intelligence, used for a variety of applications such as classification, regression, clustering, and more



Core Components

Neurons (Nodes)

- Basic units of an ANN.
- Receive input, process it, and pass the output to the next layer.
- Layers
- Input Layer: The layer that receives the input data.
- Hidden Layers: Intermediate layers where the processing is done. There can be one or more hidden layers.
- Output Layer: Produces the final output of the networ

Core Components

2. Weights

Parameters that are adjusted during training.

Represent the strength of the connection between neurons.

3. Bias

An additional parameter in each neuron that allows the model to fit the data better.

4. Activation Function

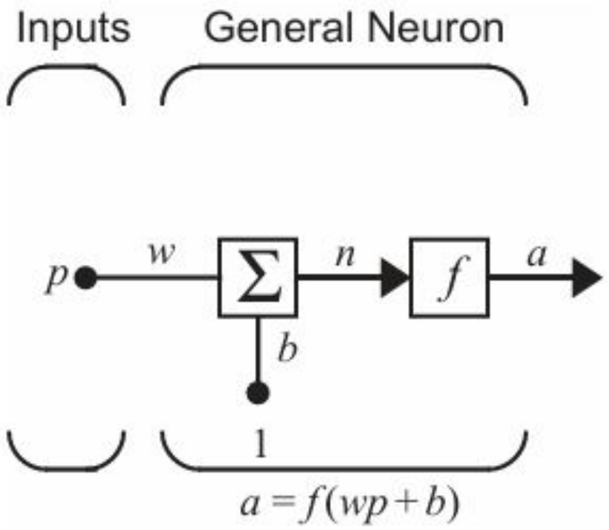
A mathematical function is applied to the output of each neuron.

Introduces non-linearity to the model.

Common activation functions include Sigmoid, Tanh, ReLU (Rectified Linear Unit), and Softmax.

Neuron Model – Single Input

Single Input Neuron



The neuron output is calculated as

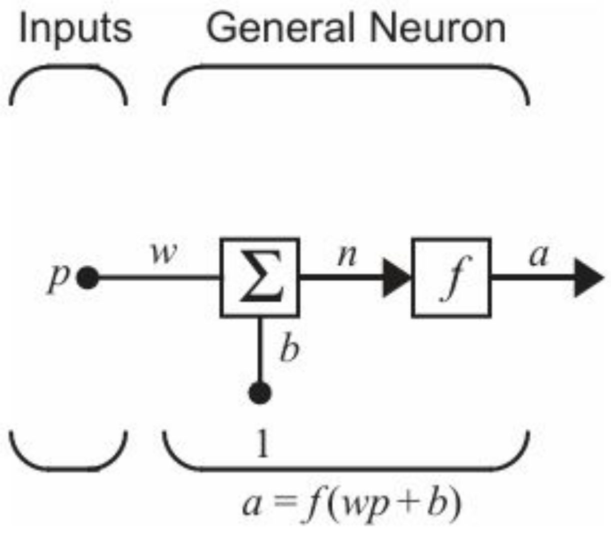
$$a = f(wp + b).$$

If, for instance, $w = 3$, $p = 2$ and $b = -1.5$, then

$$a = f(3(2) - 1.5) = f(4.5)$$

Neuron Model – Single Input

Single Input Neuron



p – scalar input
w- weight
b – bias
n – summer output
f – transfer function or Activation unit – which produces the output a

The neuron output is calculated as

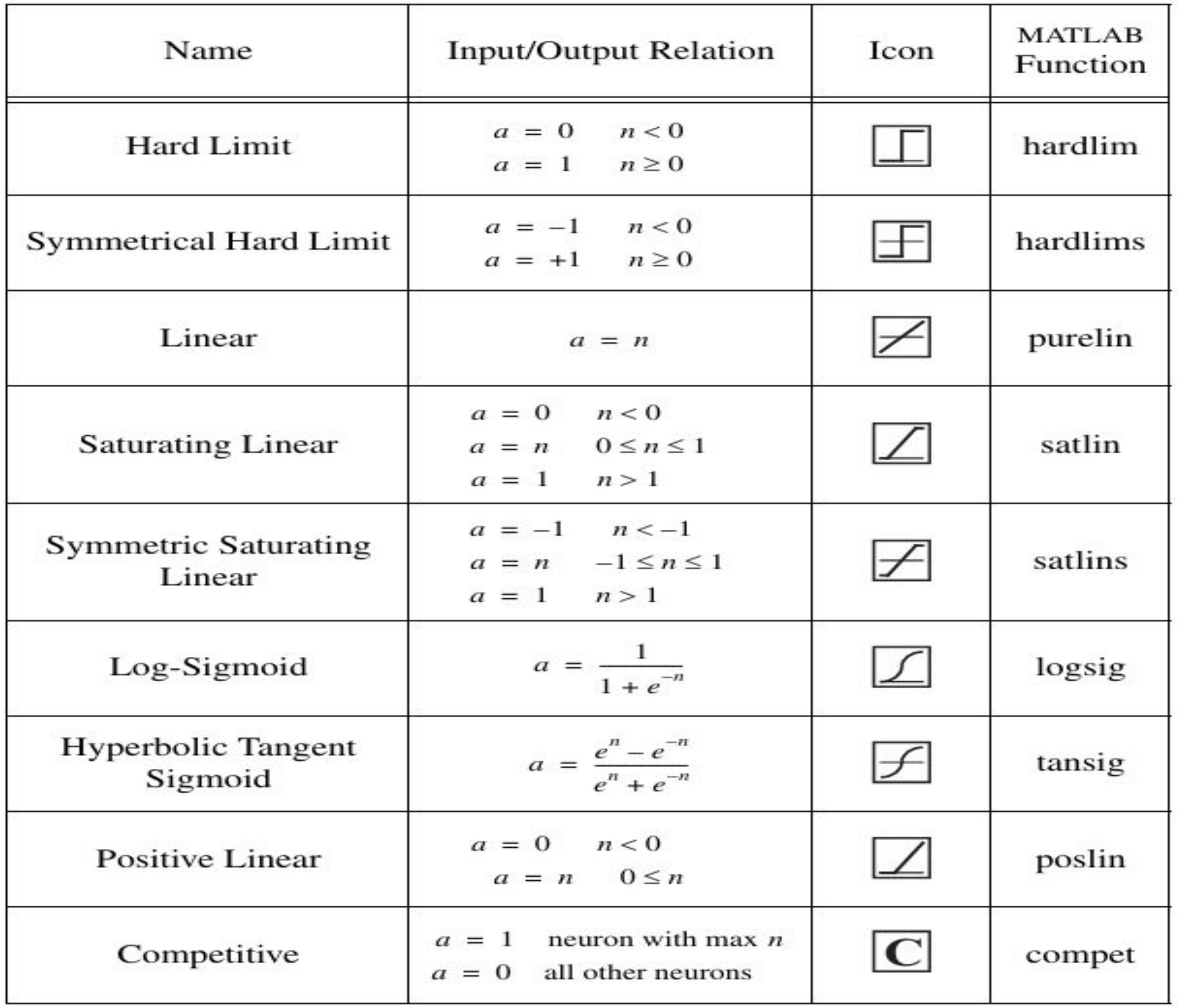
$$a = f(wp + b).$$

If, for instance, $w = 3$, $p = 2$ and $b = -1.5$, then

$$a = f(3(2) - 1.5) = f(4.5)$$

Transfer function – choosen by the designer
w&b – adjusted by some learning rule

Transfer Functions

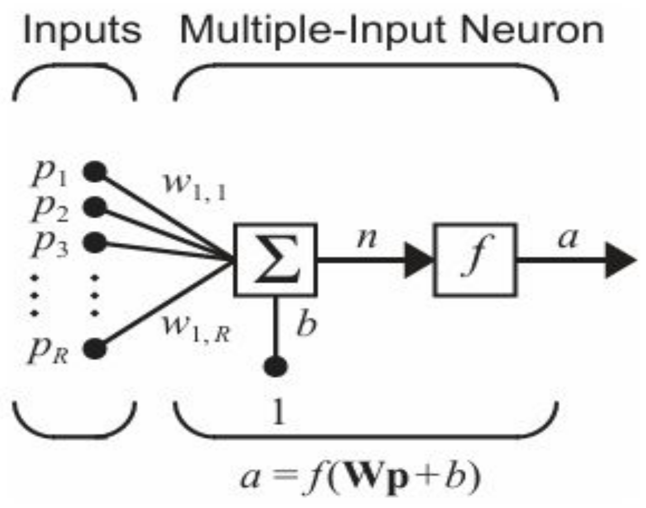


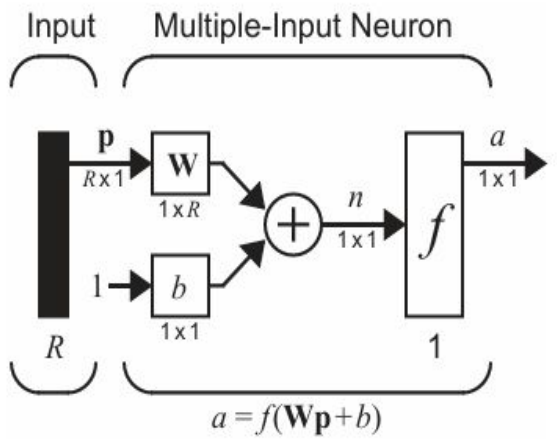
Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons	C	compet

- The Transfer function may be linear or non linear functions
- A particular transfer function is chosen to satisfy some specification of the problem that the neuron is attempting to solve.

Neuron Model – Multiple Input

Multiple- Input Neuron





The neuron has a bias b, which is summed with the weighted inputs to form the net input n:

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \cdots + w_{1,R}p_R + b\,. \tag{2.3}$$

This expression can be written in matrix form:

$$n = \mathbf{Wp} + b\,, \tag{2.4}$$

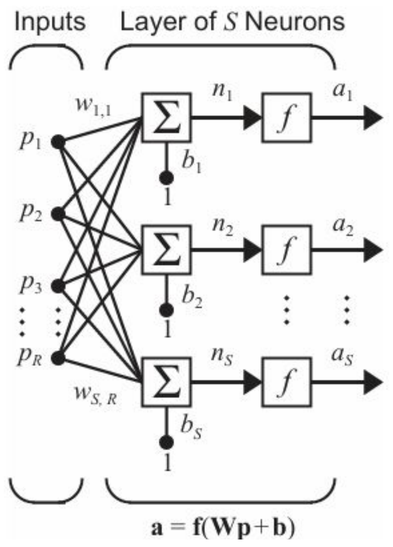
where the matrix $\mathbf{W}$ for the single neuron case has only one row.

Now the neuron output can be written as

$$a = f(\mathbf{Wp} + b)\,. \tag{2.5}$$

Network Architecture

A Layer of Neurons

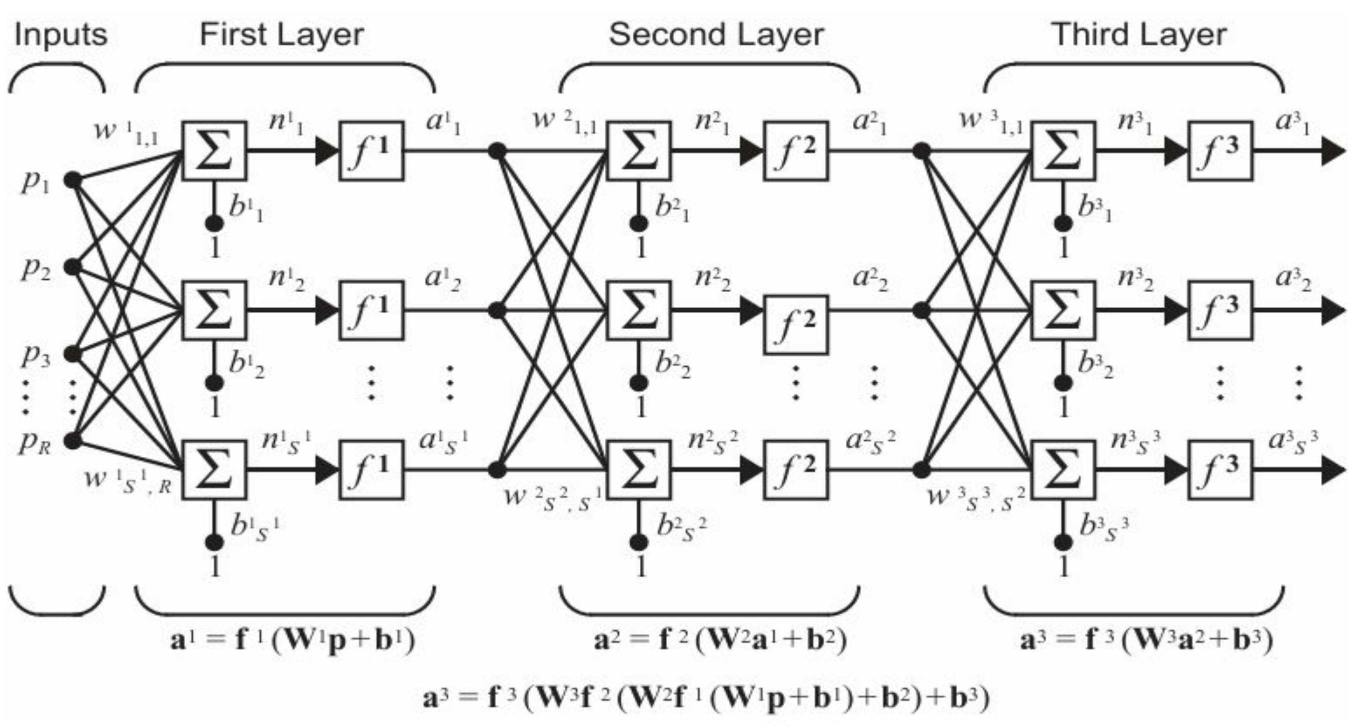


$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}.$$

Row indices – indicate the destination neuron associated with that weight
Coloumn indices – indicate the source of the input for that weight
W3,2 – indicate the connection to the third neuron from the second source.

Multiple Layer of Neurons

Three-Layer Network



A layer whose output is the network output is called an output layer other layers are called hidden layers

Recurrent Networks

Recurrent Network

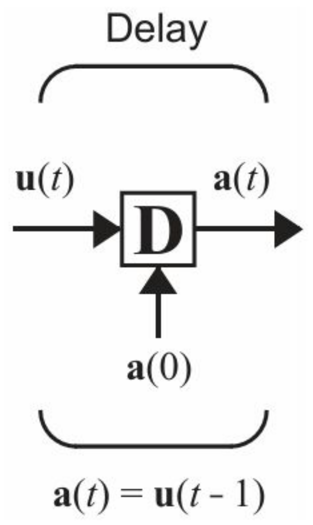
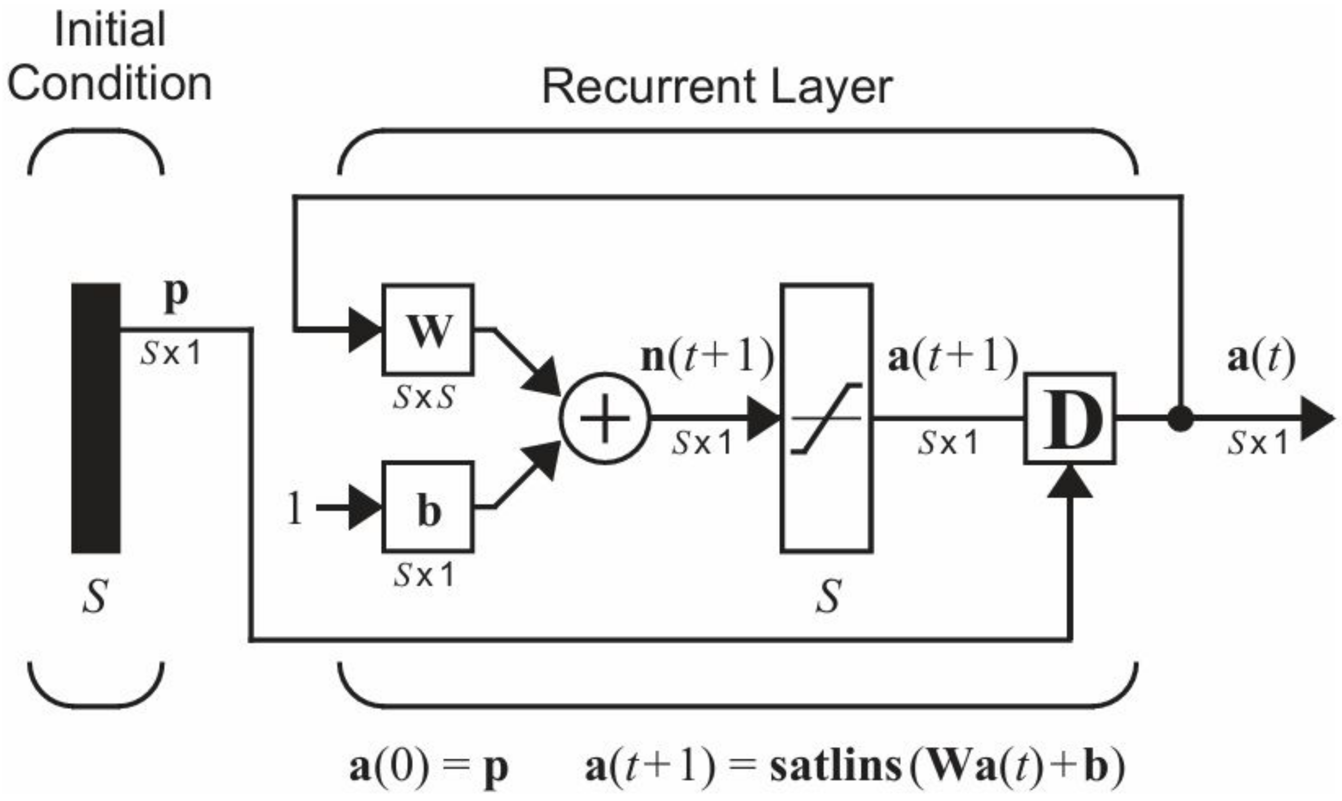


Fig. Delay Block

The delay output a(t) is computed from its input u(t)
a(t) = u(t-1)

Recurrent Networks

Recurrent Network



A recurrent network is a network with feedback; some of its outputs are connected to its inputs.
This is quite different from the networks that we have studied thus far,
which were strictly feedforward with no backward connections.

Recurrent Networks

Recurrent Network

- A recurrent network is a network with feedback; some of its outputs are connected to its inputs.
- This is quite different from the networks that we have studied thus far, which were strictly feedforward with no backward connections.
- In this particular network the vector supplies the initial conditions (i.e., a(0) =p) .
- Then future outputs of the network are computed from previous outputs:

$$\mathbf{a}(1) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(0) + \mathbf{b}),\ \mathbf{a}(2) = \mathbf{satlins}(\mathbf{W}\mathbf{a}(1) + \mathbf{b}),\ \ldots$$

How to Pick an Architecture

Problem specifications help define the network in the following ways:

1. Number of network inputs = number of problem inputs

2. Number of neurons in output layer = number of problem outputs

3. Output layer transfer function choice at least partly determined by problem specification of the output

Problems

P2.3 Given a two-input neuron with the following parameters: $b = 1.2$, $\mathbf{W} = \begin{bmatrix} 3 & 2 \end{bmatrix}$ and $\mathbf{p} = \begin{bmatrix} -5 & 6 \end{bmatrix}^T$, calculate the neuron output for the following transfer functions:

i. A symmetrical hard limit transfer function

ii. A saturating linear transfer function

iii. A hyperbolic tangent sigmoid (tansig) transfer function

First calculate the net input n:

$$n = \mathbf{W}\mathbf{p} + b = \begin{bmatrix} 3 & 2 \end{bmatrix} \begin{bmatrix} -5 \\ 6 \end{bmatrix} + (1.2) = -1.8 .$$

Now find the outputs for each of the transfer functions.

i. $a = hardlims(-1.8) = -1$

ii. $a = satlin(-1.8) = 0$

iii. $a = tansig(-1.8) = -0.9468$