A Brief Overview of Neural Networks

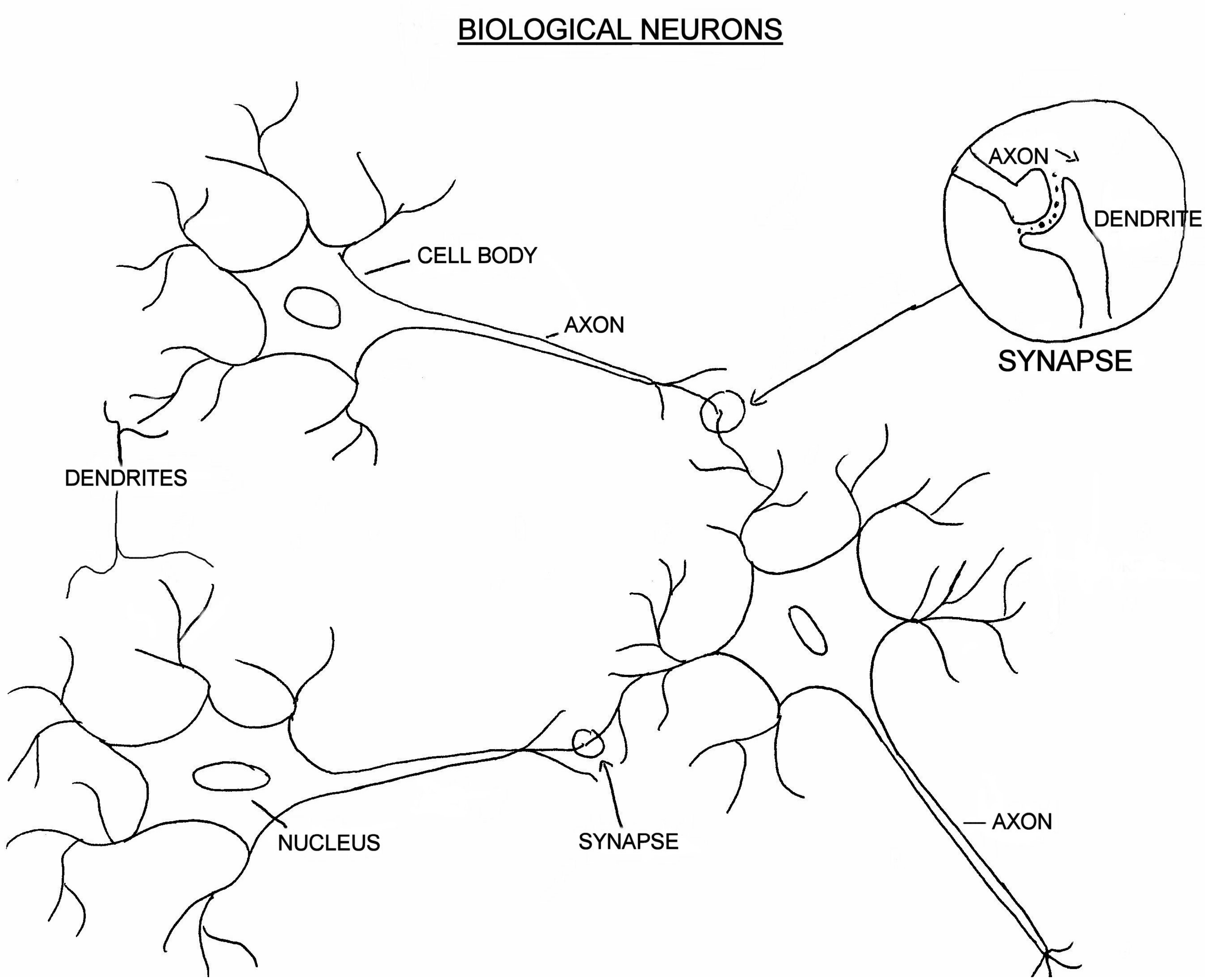
By

Rohit Dua, Samuel A. Mulder, Steve E. Watkins, and Donald C. Wunsch

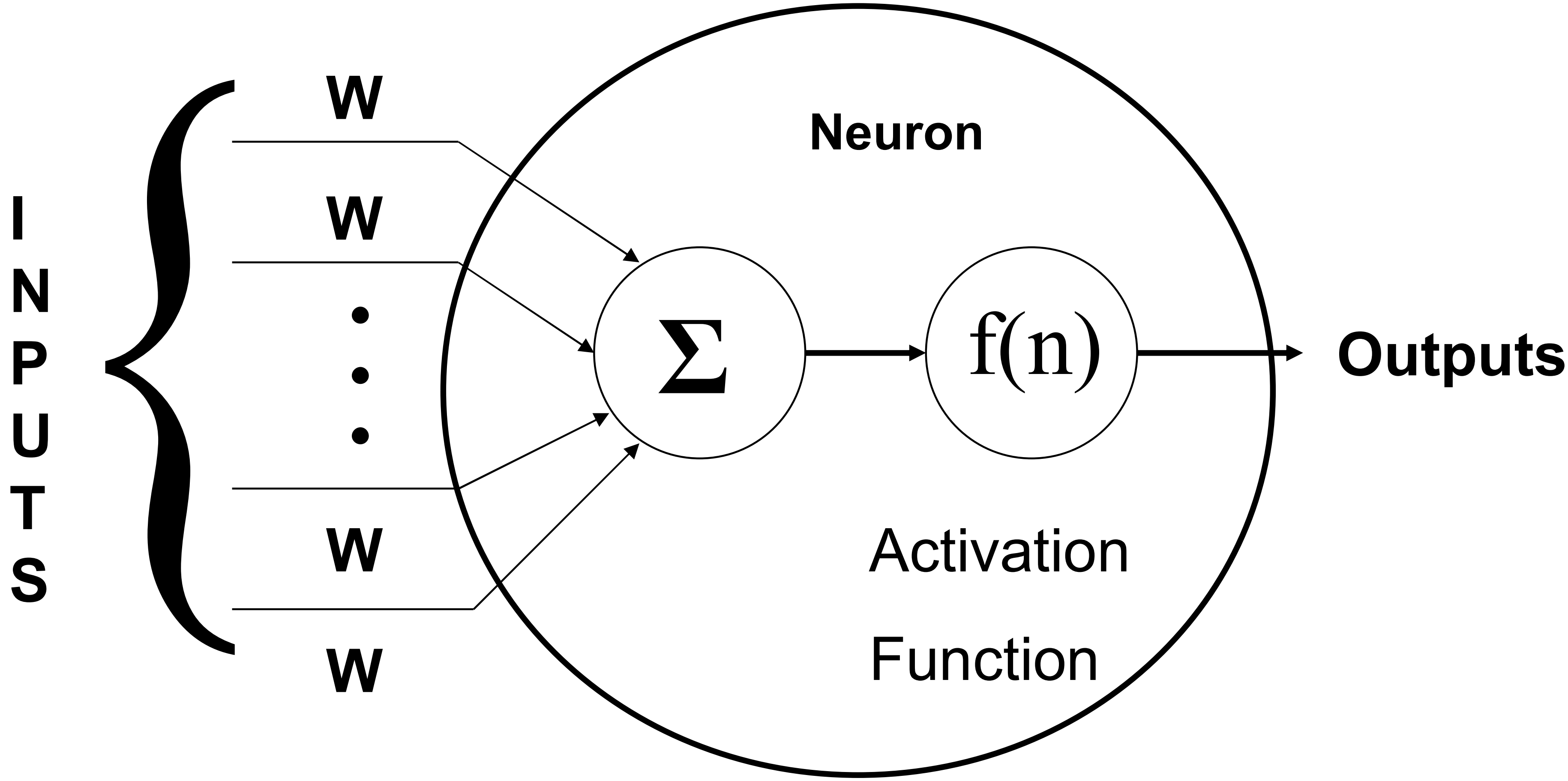
Overview

- Relation to Biological Brain: Biological Neural Network
- The Artificial Neuron
- Types of Networks and Learning Techniques
- Supervised Learning & Backpropagation Training Algorithm
- Learning by Example
- Applications
- Questions

Biological Neuron

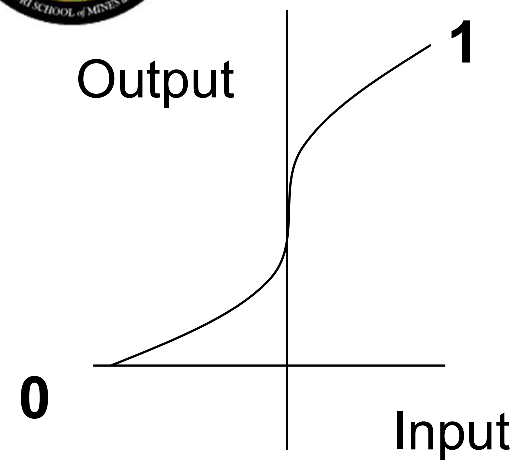


Artificial Neuron

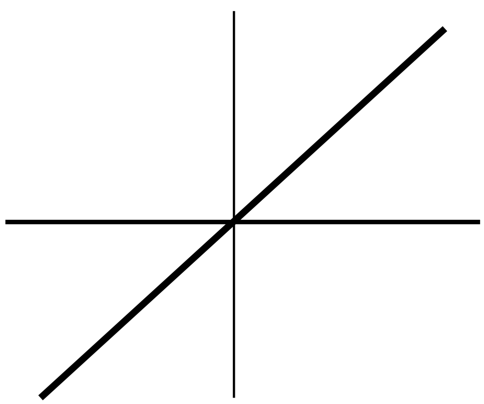


W=Weight

Transfer Functions

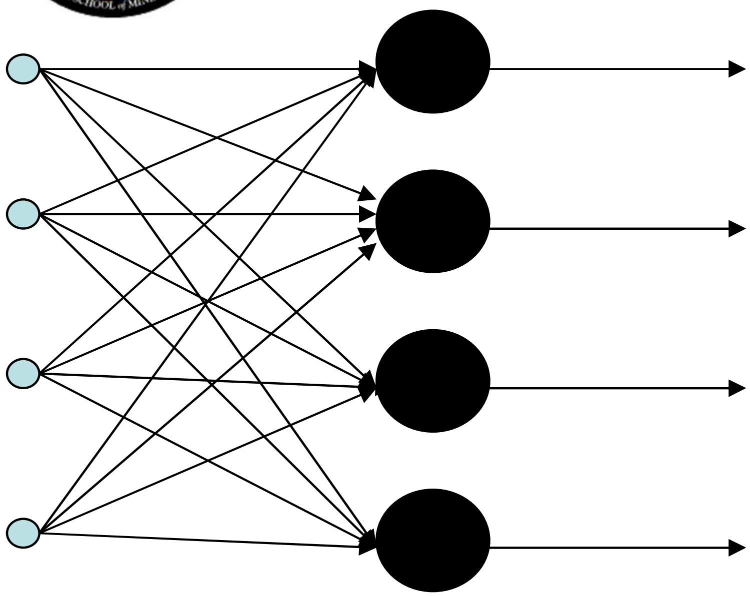


$$SIGMOID: f(n) = \frac{1}{1+e^{-n}}$$

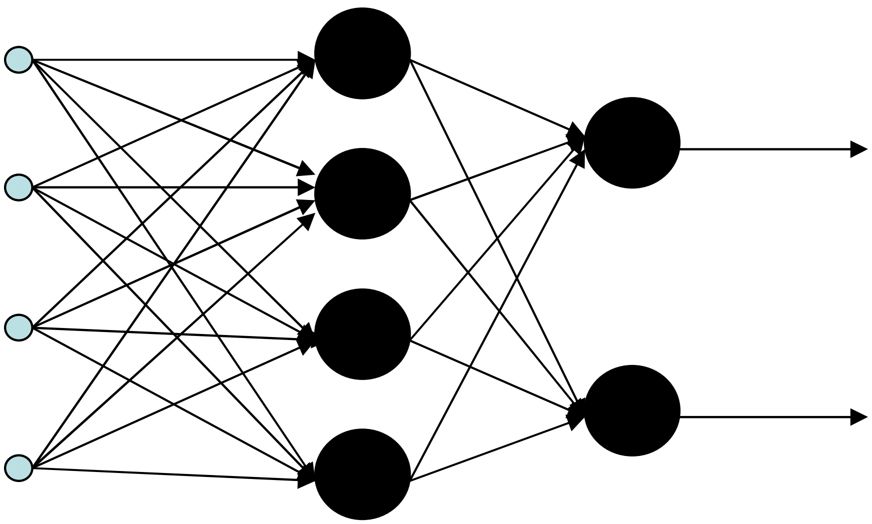


$$LINEAR: f(n) = n$$

Types of networks

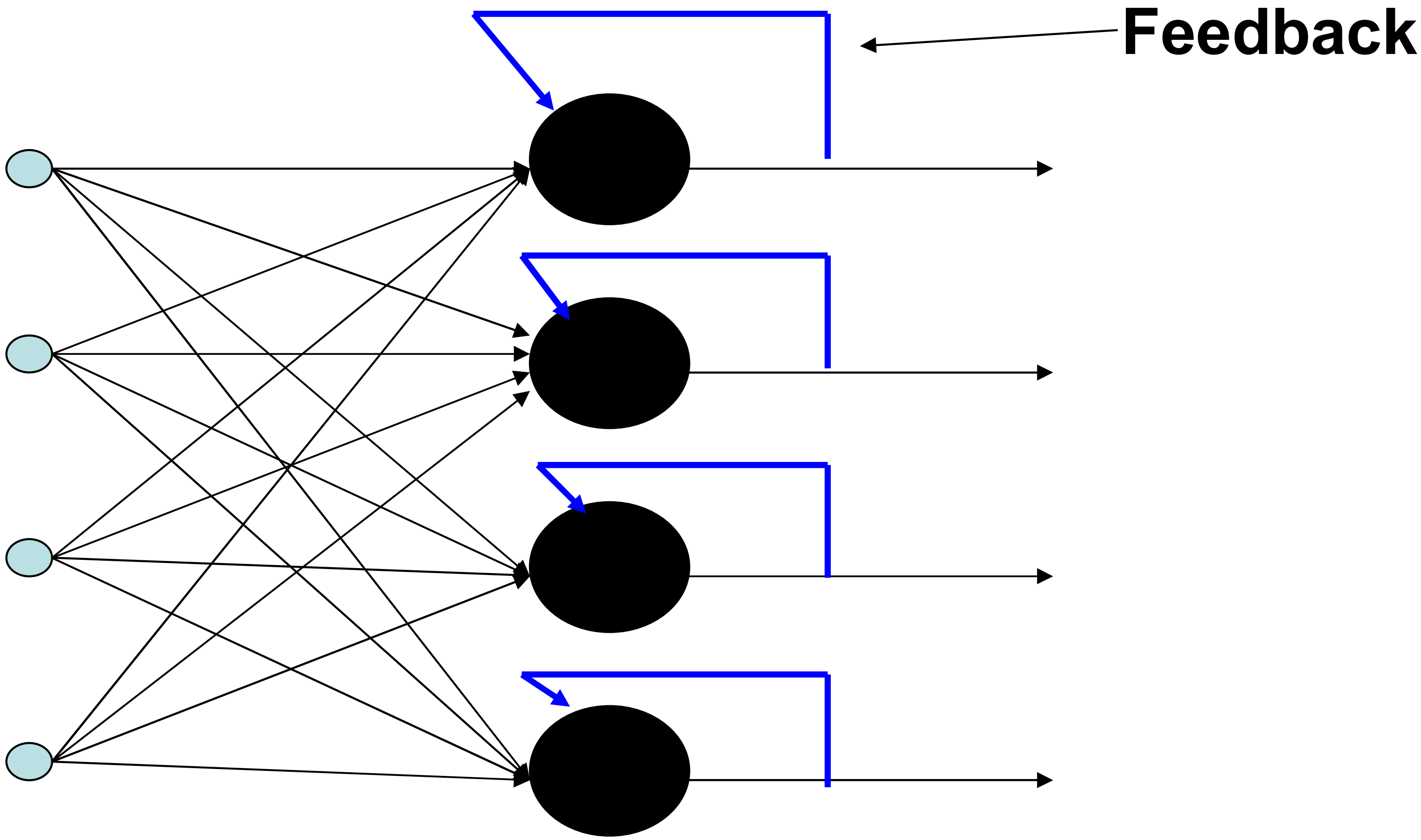


Multiple Inputs and Single Layer



Multiple Inputs and layers

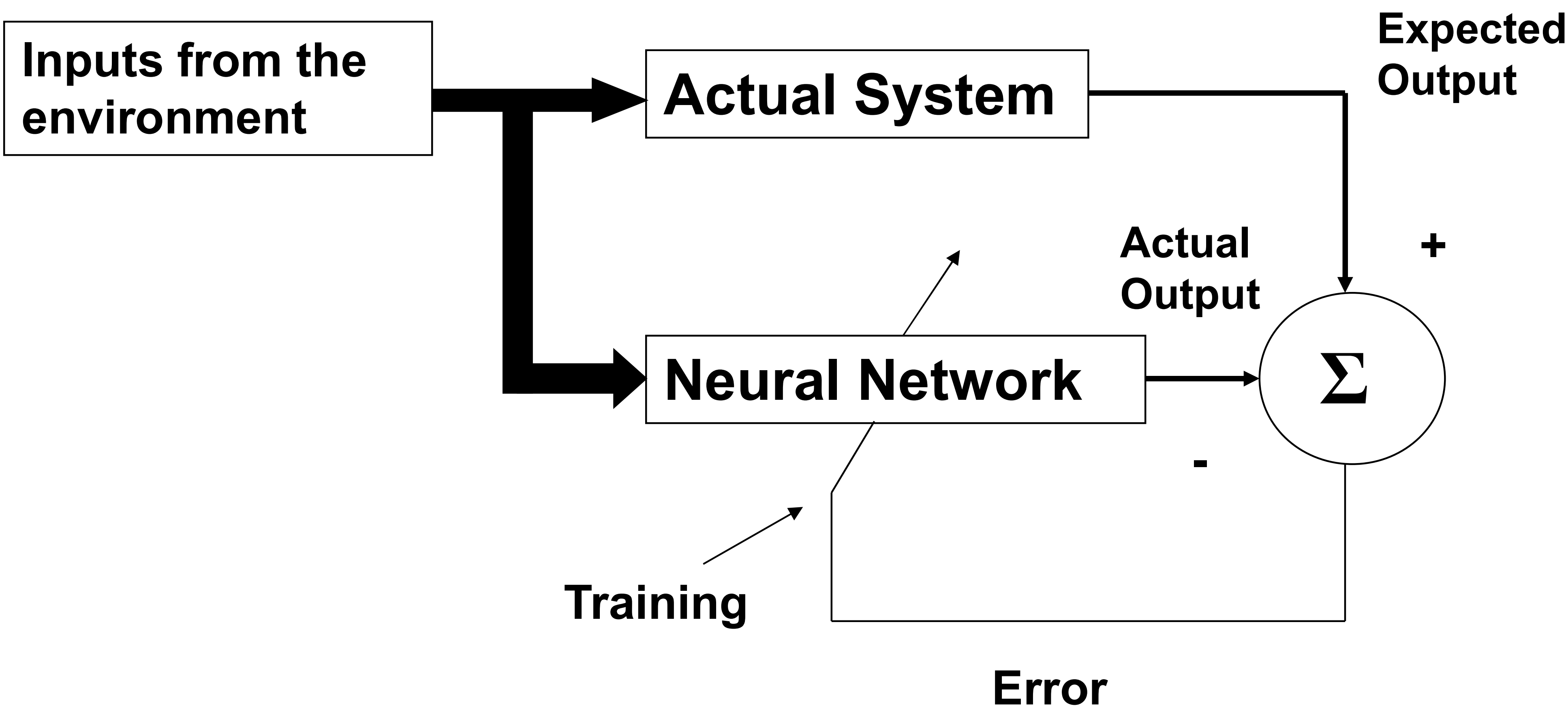
Types of Networks – Contd.



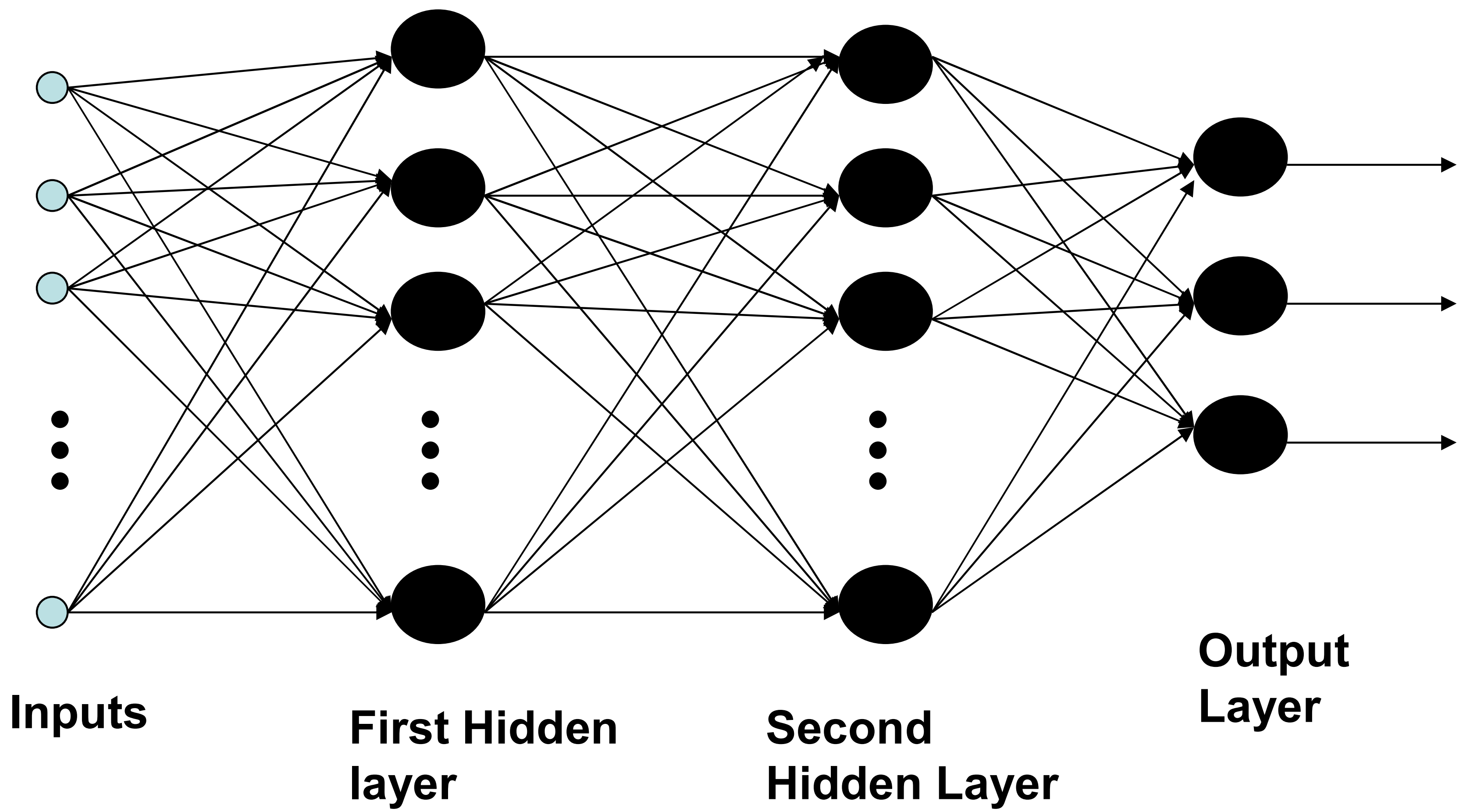
Recurrent Networks

Learning Techniques

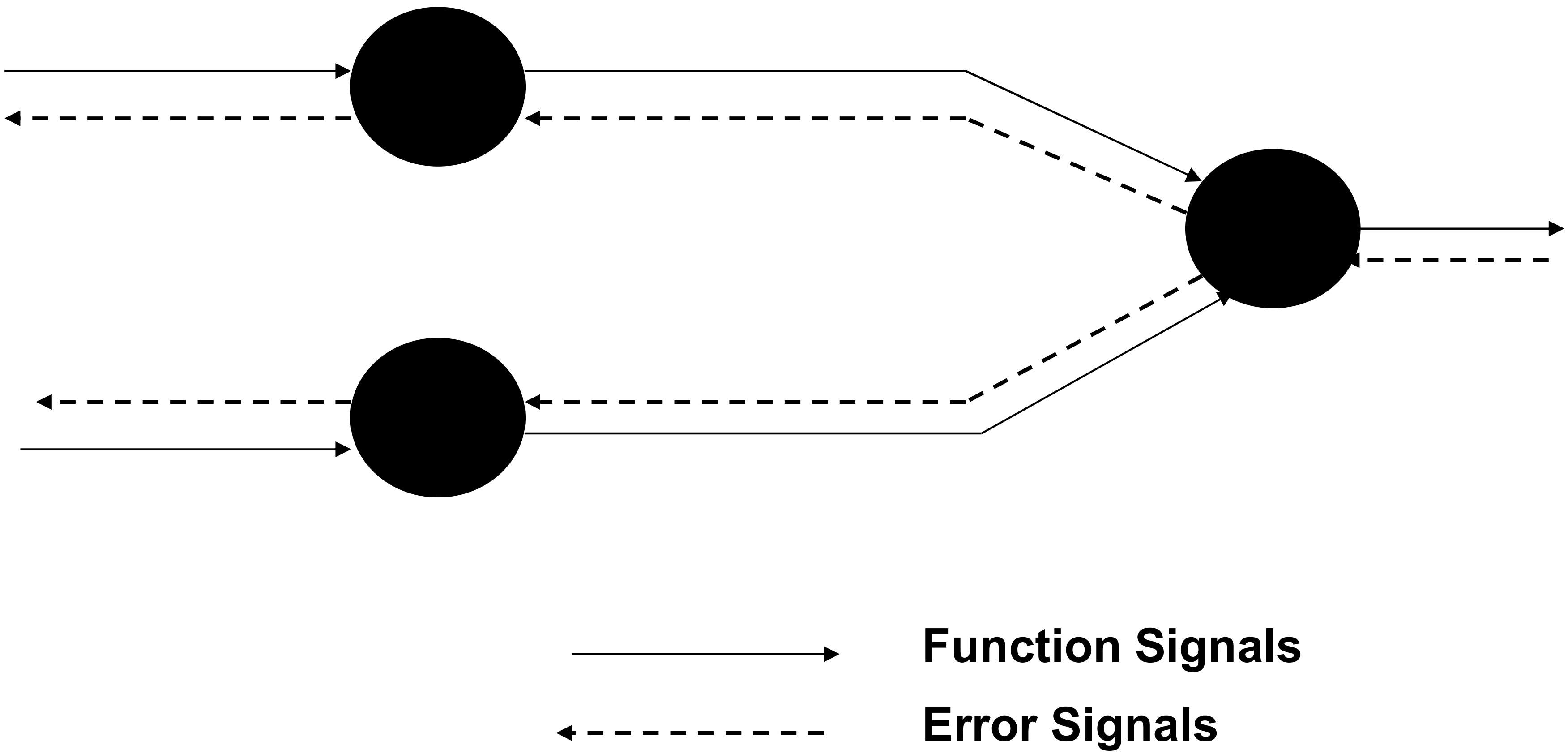
- Supervised Learning:



Multilayer Perceptron



Signal Flow
Backpropagation of Errors



Learning by Example

- Hidden layer transfer function: Sigmoid function = F(n)= 1/(1+exp(-n)), where n is the net input to the neuron.

  Derivative= F'(n) = (output of the neuron)(1-output of the neuron) : Slope of the transfer function.
- Output layer transfer function: Linear function= F(n)=n; Output=Input to the neuron

  Derivative= F'(n)= 1

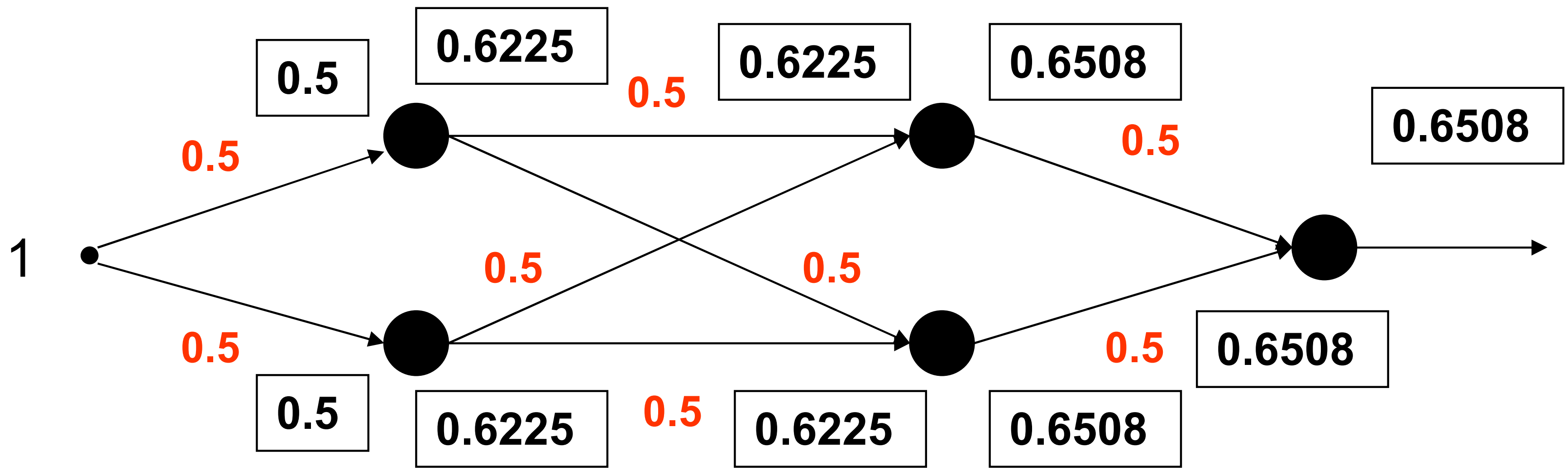
Learning by Example

- Training Algorithm: backpropagation of errors using gradient descent training.
- Colors:
  - Red: Current weights
  - Orange: Updated weights
  - Black boxes: Inputs and outputs to a neuron
  - Blue: Sensitivities at each layer

First Pass

G1= (0.6225)(1-0.6225)(0.0397)(0.5)(2)=0.0093

G2= (0.6508)(1-0.6508)(0.3492)(0.5)=0.0397



G3=(1)(0.3492)=0.3492

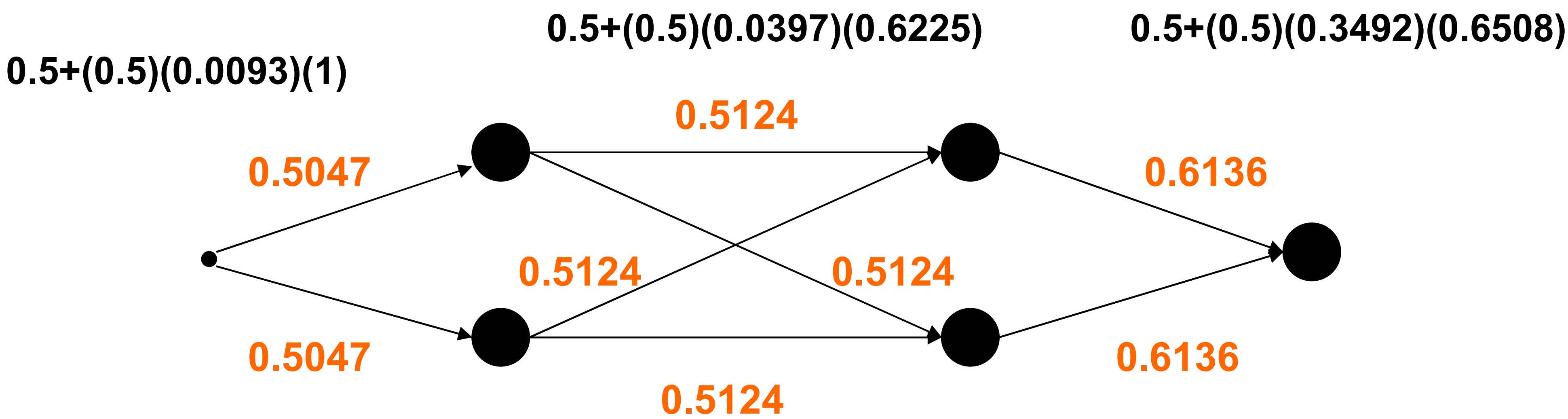
Gradient of the neuron= **G** =slope of the transfer function×[Σ{(weight of the neuron to the next neuron) × (output of the neuron)}]

Gradient of the output neuron = slope of the transfer function × error

Error=1-0.6508=0.3492

Weight Update 1

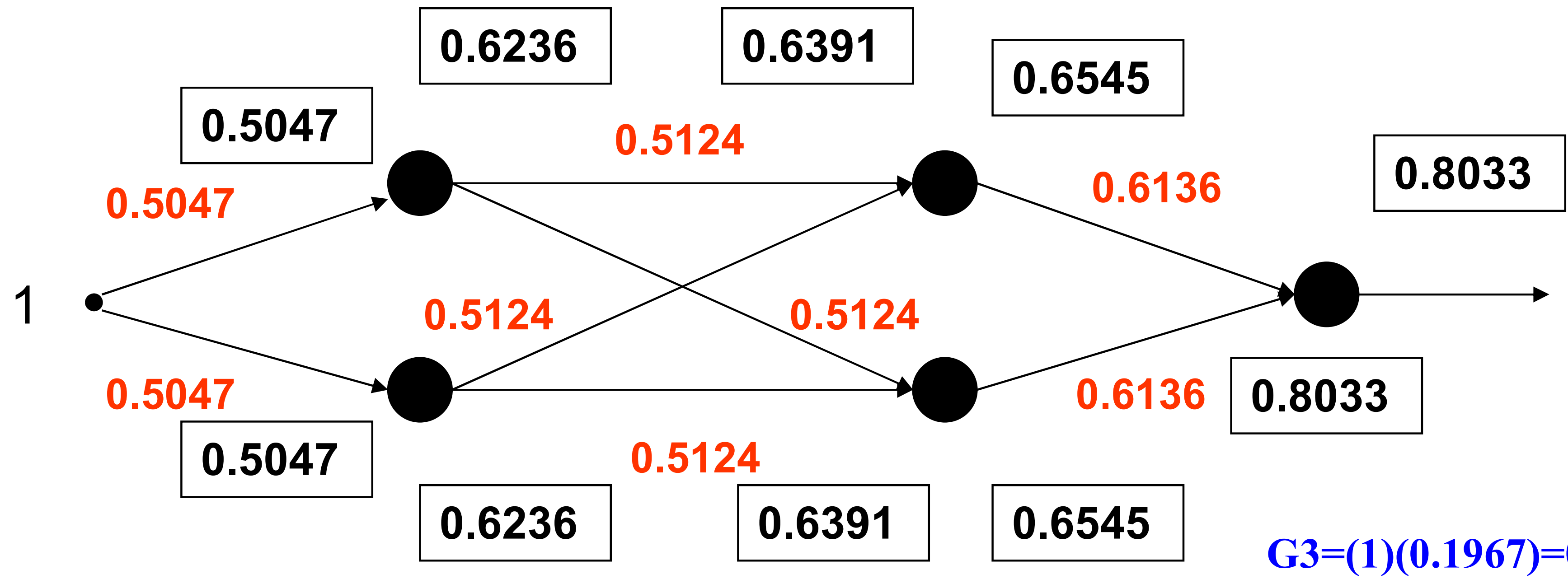
New Weight=Old Weight + {(learning rate)(gradient)(prior output)}



Second Pass

G1= (0.6236)(1-0.6236)(0.5124)(0.0273)(2)=0.0066

G2= (0.6545)(1-0.6545)(0.1967)(0.6136)=0.0273

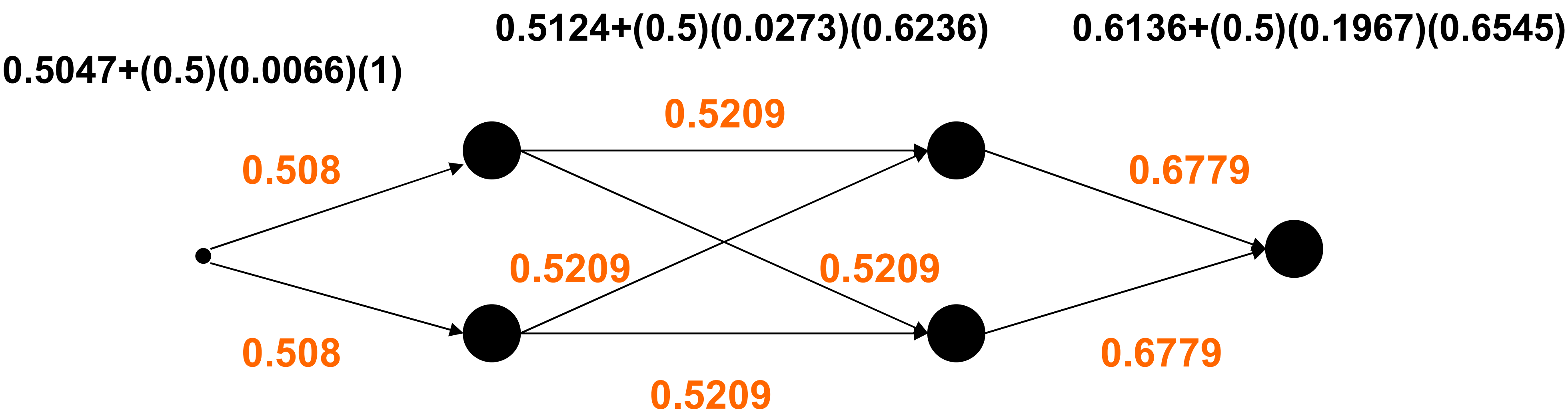


G3=(1)(0.1967)=0.1967

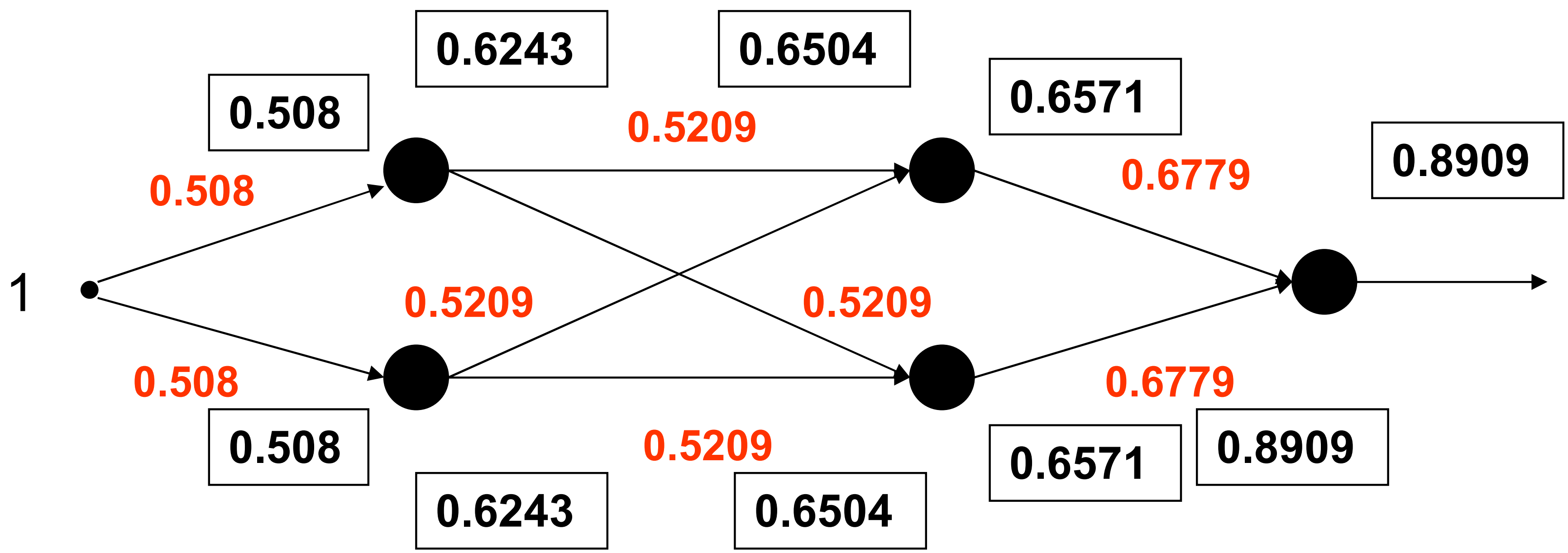
Error=1-0.8033=0.1967

Weight Update 2

New Weight=Old Weight + {(learning rate)(gradient)(prior output)}



Third Pass


0.6243
0.6504
0.508
0.6571
0.5209
0.6779
0.8909
0.508
1
0.5209
0.5209
0.508
0.6779
0.508
0.5209
0.6571
0.8909
0.6243
0.6504

Weight Update Summary

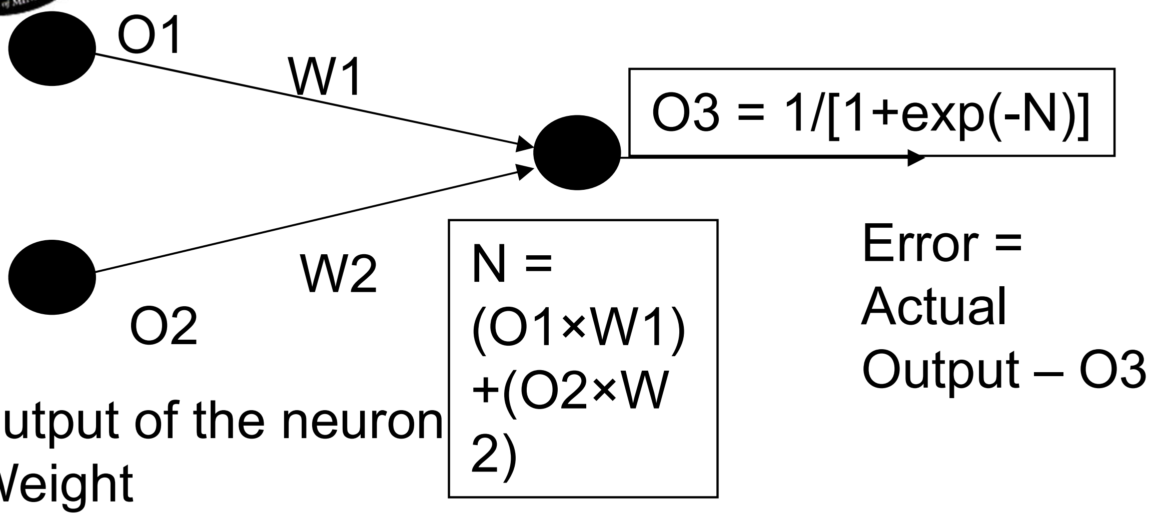
	Weights			Output	Expected (	Error
	w1	w2	w3			
Initial conditions	0.5	0.5	0.5	0.6508	1	0.3492
Pass 1 Update	0.5047	0.5124	0.6136	0.8033	1	0.1967
Pass 2 Update	0.508	0.5209	0.6779	0.8909	1	0.1091

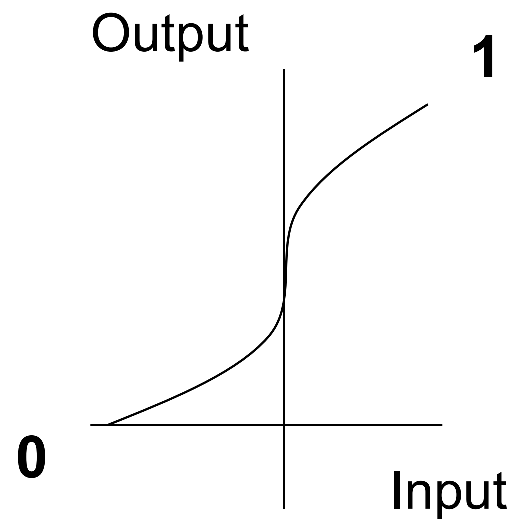
W1: Weights from the input to the input layer
W2: Weights from the input layer to the hidden layer
W3: Weights from the hidden layer to the output layer

Training Algorithm

- The process of feedforward and backpropagation continues until the required mean squared error has been reached.
- Typical mse: 1e-5
- Other complicated backpropagation training algorithms also available.

Why Gradient?





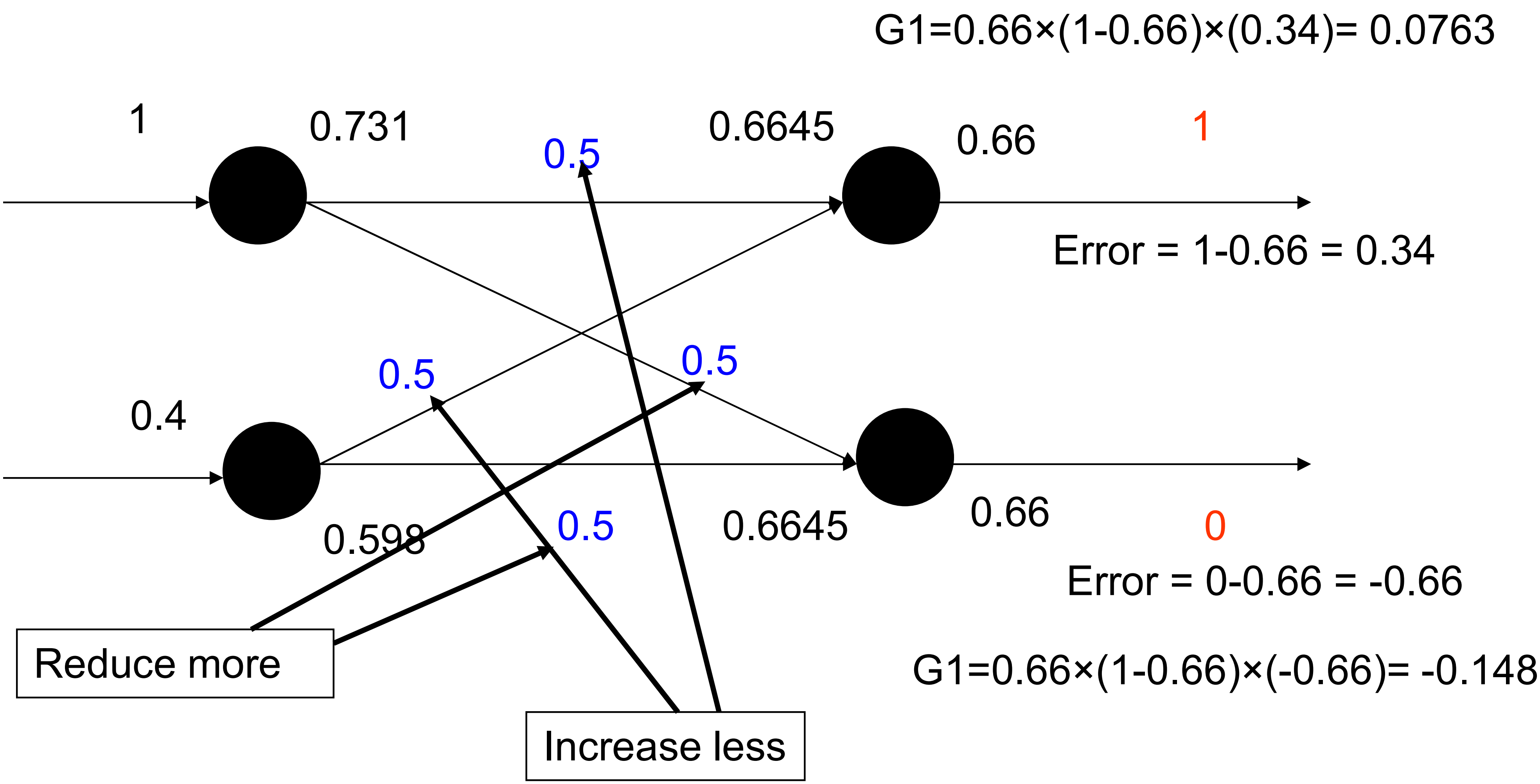
O = Output of the neuron
W = Weight
N = Net input to the neuron

- To reduce error: Change in weights:
    - o Learning rate
    - o Rate of change of error w.r.t rate of change of weight
        - ▪ Gradient: rate of change of error w.r.t rate of change of ‘N’
        - ▪ Prior output (O1 and O2)

Gradient in Detail

- Gradient : Rate of change of error w.r.t rate of change in net input to neuron
  - o For output neurons
    - Slope of the transfer function × error
  - o For hidden neurons : A bit complicated ! : error fed back in terms of gradient of successive neurons
    - Slope of the transfer function × [Σ (gradient of next neuron × weight connecting the neuron to the next neuron)]
    - Why summation? Share the responsibility!!
  - o ***Therefore: Credit Assignment Problem***

An Example



G1=0.66×(1-0.66)×(0.34)= 0.0763
1
0.731
0.5
0.6645
0.66
1
Error = 1-0.66 = 0.34
0.5
0.4
0.5
0.5
0.598
0.6645
0.66
0
Error = 0-0.66 = -0.66
G1=0.66×(1-0.66)×(-0.66)= -0.148
Reduce more
Increase less

Improving performance

- Changing the number of layers and number of neurons in each layer.
- Variation in Transfer functions.
- Changing the learning rate.
- Training for longer times.
- Type of pre-processing and post-processing.

Applications

- Used in complex function approximations, feature extraction & classification, and optimization & control problems
- Applicability in all areas of science and technology.