ARTIFICIAL NEURAL NETWORKS: AN INTRODUCTION

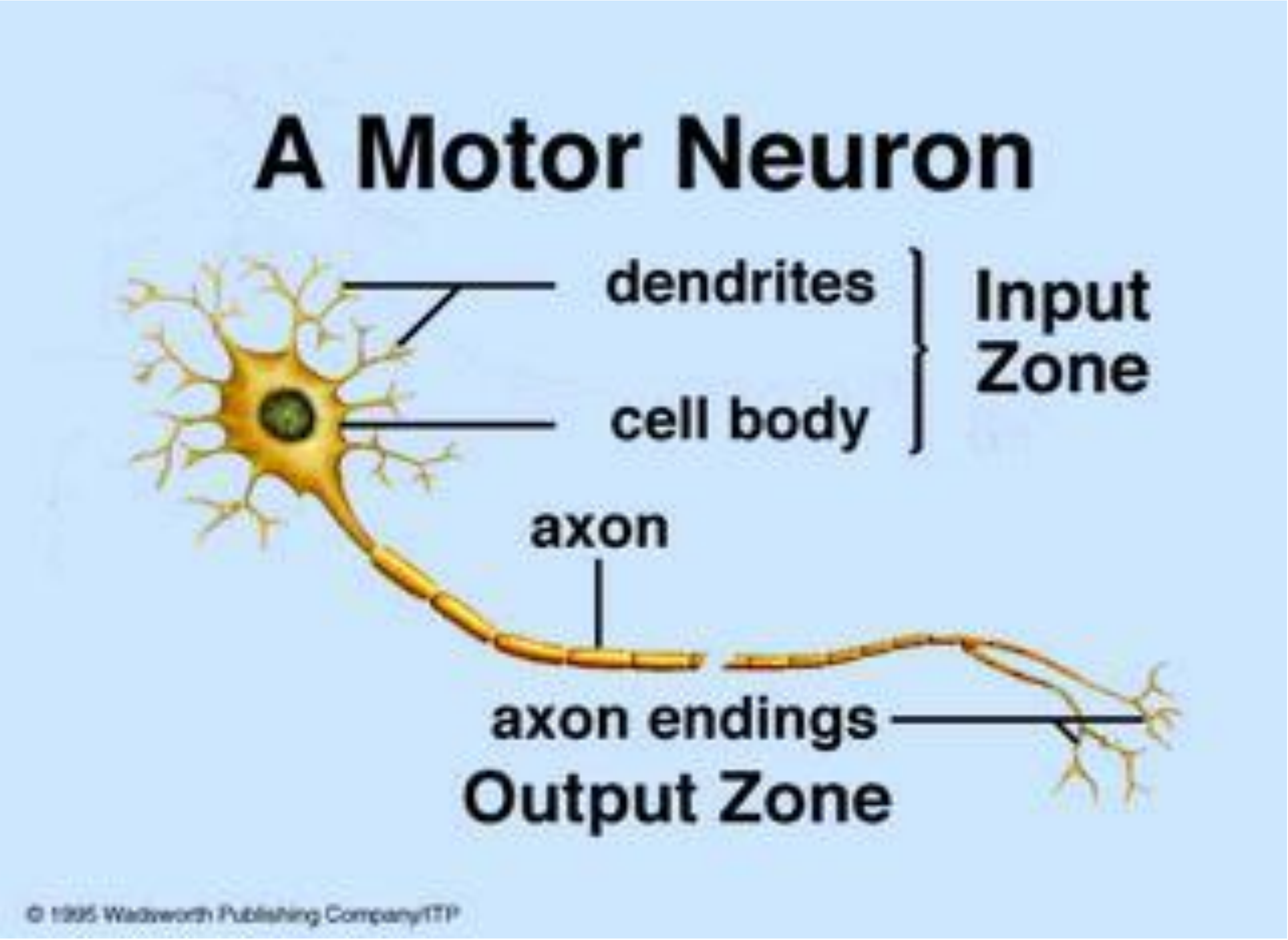
NEURAL NETWORKS

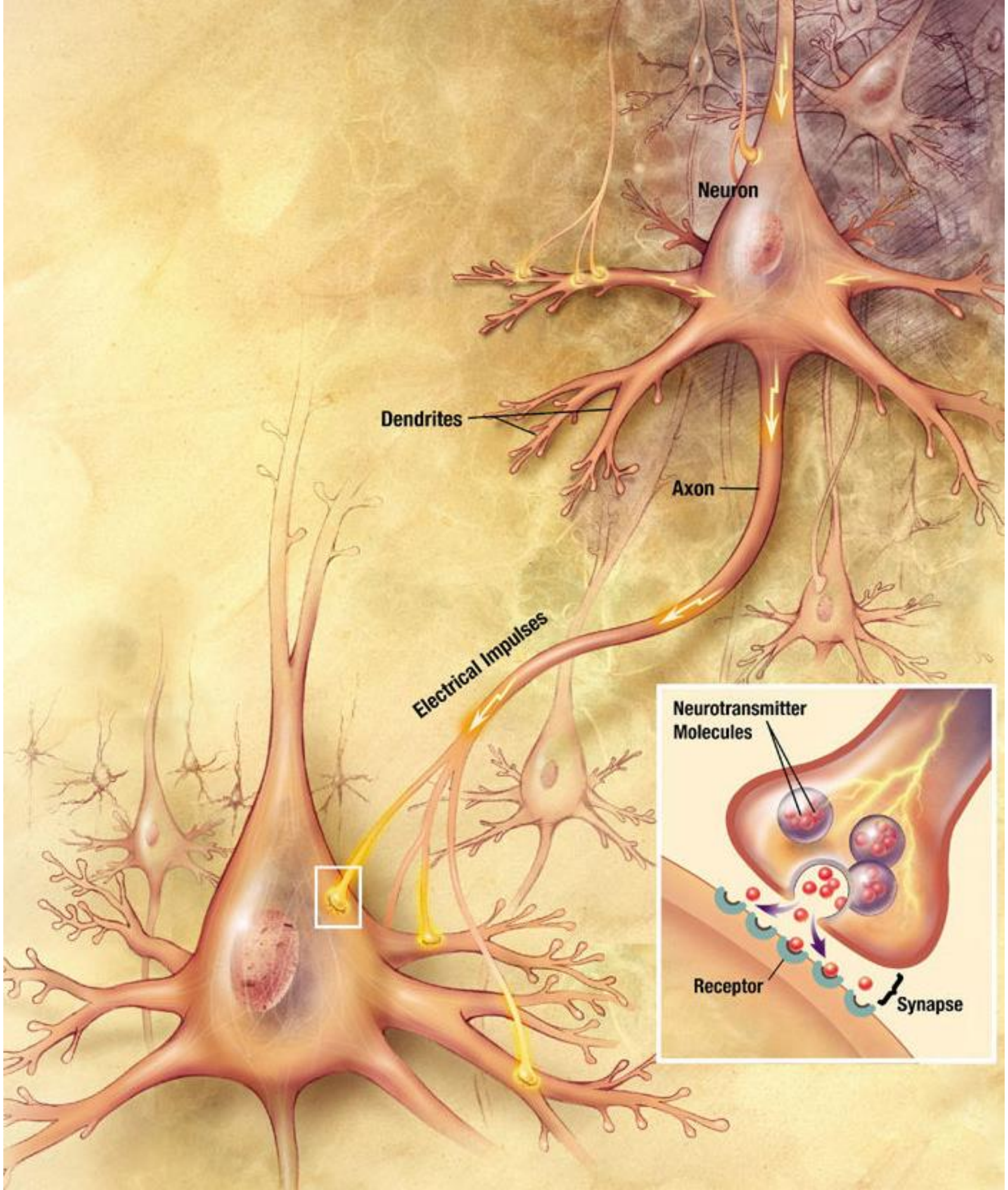
NN model human brain

ANN tasks – pattern-matching, classification, optimization function, approximation, vector quantization, data clustering

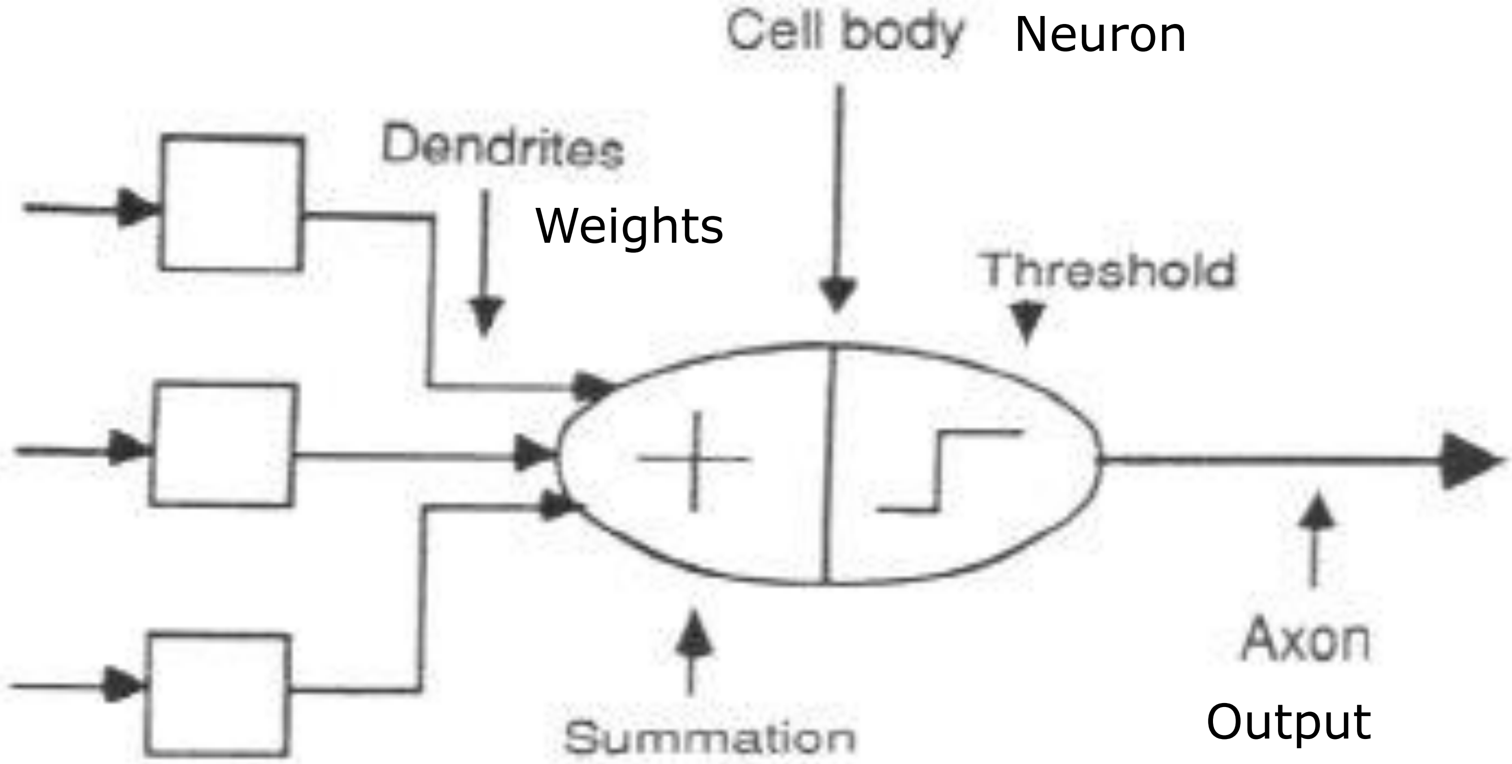
ANN is an efficient processing system which resembles in characteristics with biological neural network.

BIOLOGICAL (MOTOR) NEURON




Neuron
Dendrites
Axon
Electrical Impulses
Neurotransmitter
Molecules
Receptor
Synapse

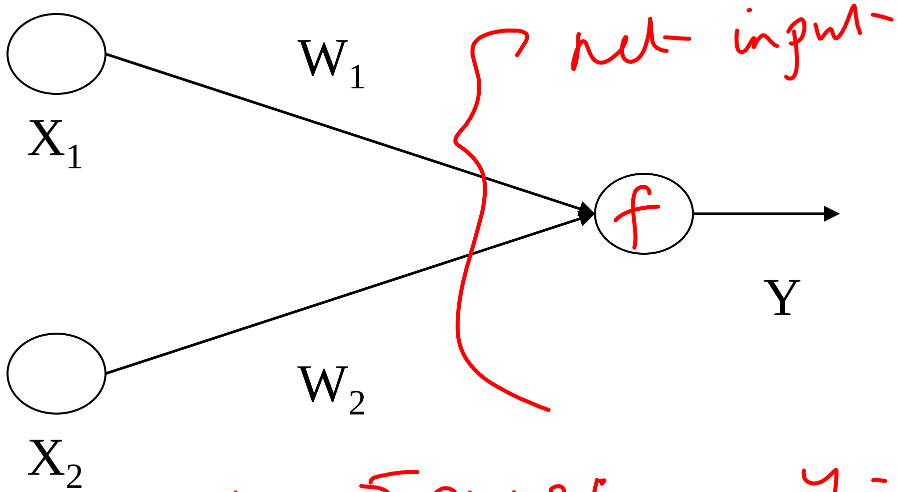
ASSOCIATION OF BIOLOGICAL NET WITH ARTIFICIAL NET



ARTIFICIAL NEURAL NET

- Information-processing system.
- Neurons process the information.
- The signals are transmitted by means of connection links.
- The links possess an associated weight.
- The output signal is obtained by applying activations to the net input.

ARTIFICIAL NEURAL NET



$net = \sum x_i w_i$ $\quad y = f(net)$

The figure shows a simple artificial neural net with two input neurons (X_1, X_2) and one output neuron (Y). The inter connected weights are given by W_1 and W_2.

PROCESSING OF AN ARTIFICIAL NET

The neuron is the basic information processing unit of a NN. It consists of:

1. A set of links, describing the neuron inputs, with weights W_1, W_2, …, W_m.

2. An adder function (linear combiner) for computing the weighted sum of the inputs (real numbers):

$$\mathbf{u} = \sum_{j=1}^{m} \mathbf{W}_j \mathbf{X}_j$$

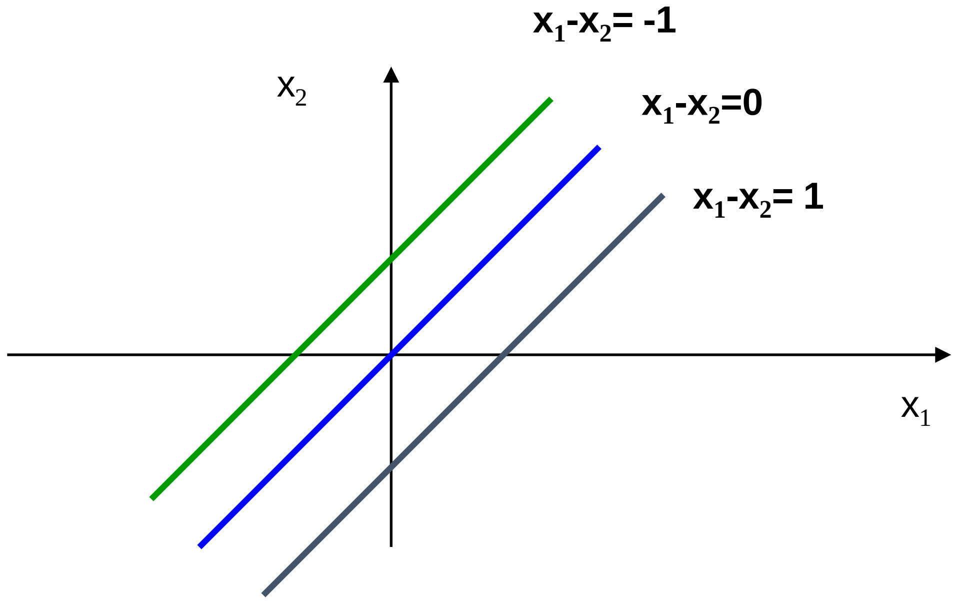
3. Activation function for limiting the amplitude of the neuron output.

$$\mathrm{y} = \varphi(\mathrm{u} + b)$$

BIAS OF AN ARTIFICIAL NEURON

The bias value is added to the weighted sum $\sum w_i x_i$ so that we can transform it from the origin.

$$Y_{in} = \sum w_i x_i + b, \text{ where } b \text{ is the bias}$$



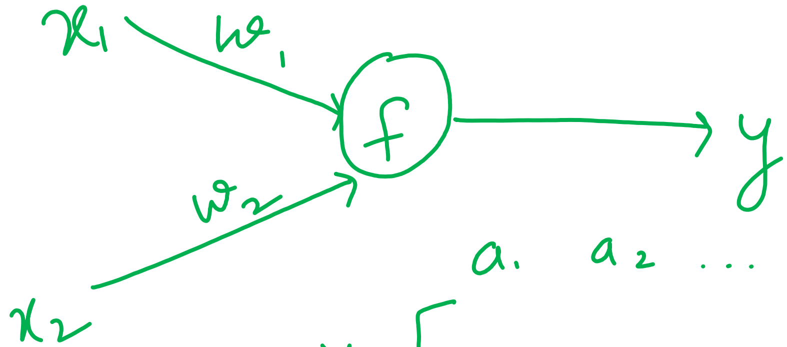
MULTI LAYER ARTIFICIAL NEURAL NET

INPUT: records without class attribute with normalized attributes values.

INPUT VECTOR: $X = \{ x_1, x_2, \dots, x_n \}$ where n is the number of (non-class) attributes.

INPUT LAYER: there are as many nodes as non-class attributes, i.e. as the length of the input vector.

HIDDEN LAYER: the number of nodes in the hidden layer and the number of hidden layers depends on implementation.



attributes / variables or features / dimensions

$$
\begin{array}{c|c}
 & a_1 \quad a_2 \quad \dots \quad a_n \\
\hline
X = \begin{array}{c} X_1 \\ X_2 \\ \vdots \\ X_n \end{array} & \begin{bmatrix} \text{———————} \\ \\ \\ \\ \end{bmatrix}
\end{array}
$$

instance / sample / data points or data objects

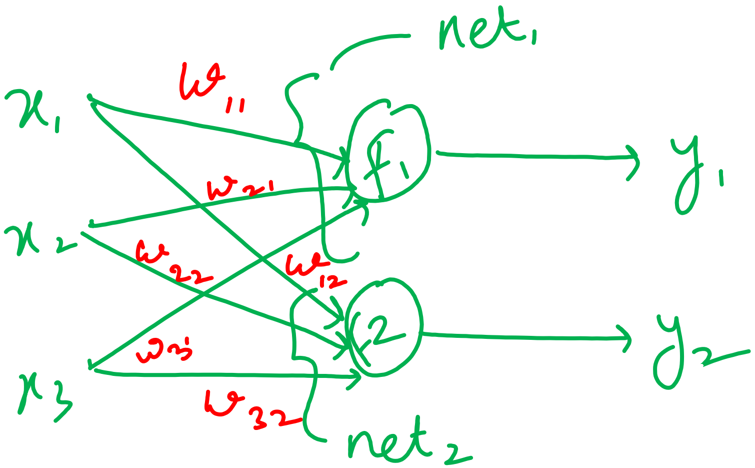
iris

$$
X = \begin{array}{c} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_{150} \end{array}
\begin{bmatrix}
\underline{x_{11}} & x_{12} & \dots & x_{14} \\
x_{21} & x_{22} & \dots & x_{24} \\
 & & & \\
 & & & \\
 & & &
\end{bmatrix}
$$

$$W = [w_1 \quad w_2]$$

100

95-98



$X = [\quad]$

W_1 1 $\quad W_2$ 2x3

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

2

3x2

1

$$net_1 = x_1 w_{11} + x_2 w_{21} + x_3 w_{31}$$

$$y_1 = f_1(net_1)$$

$$net_2 = x_1 w_{12} + x_2 w_{22} + x_3 w_{32}$$

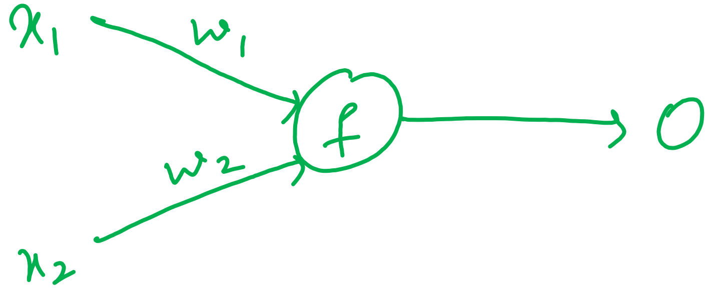
$$y_2 = f_2(net_2)$$

$$net_1 = X W_1$$

10 x 3

3 x 2

10 x 2



Training | Epoch | Iteration

$$X = \begin{array}{c c} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{array}{c} X_1 \\ X_2 \\ \vdots \\ X_{10} \end{array} & \left[\begin{array}{cc} \; & \; \\ \; & \; \\ \; & \; \\ \; & \; \end{array} \right]_{10\times 2} \end{array} \qquad D = \left[\begin{array}{c} y_1 \\ y_2 \\ \; \\ \; \end{array} \right]_{10\times 1}$$

$X_1 \overset{W_1}{-} net_1 - O_1 \equiv y_1$

$X_2 \overset{w_2}{-} net_2 - O_2 \equiv y_2$ $\qquad W_2 \leftarrow W_1$ $\qquad W_1 = [w_1 \quad w_2]$

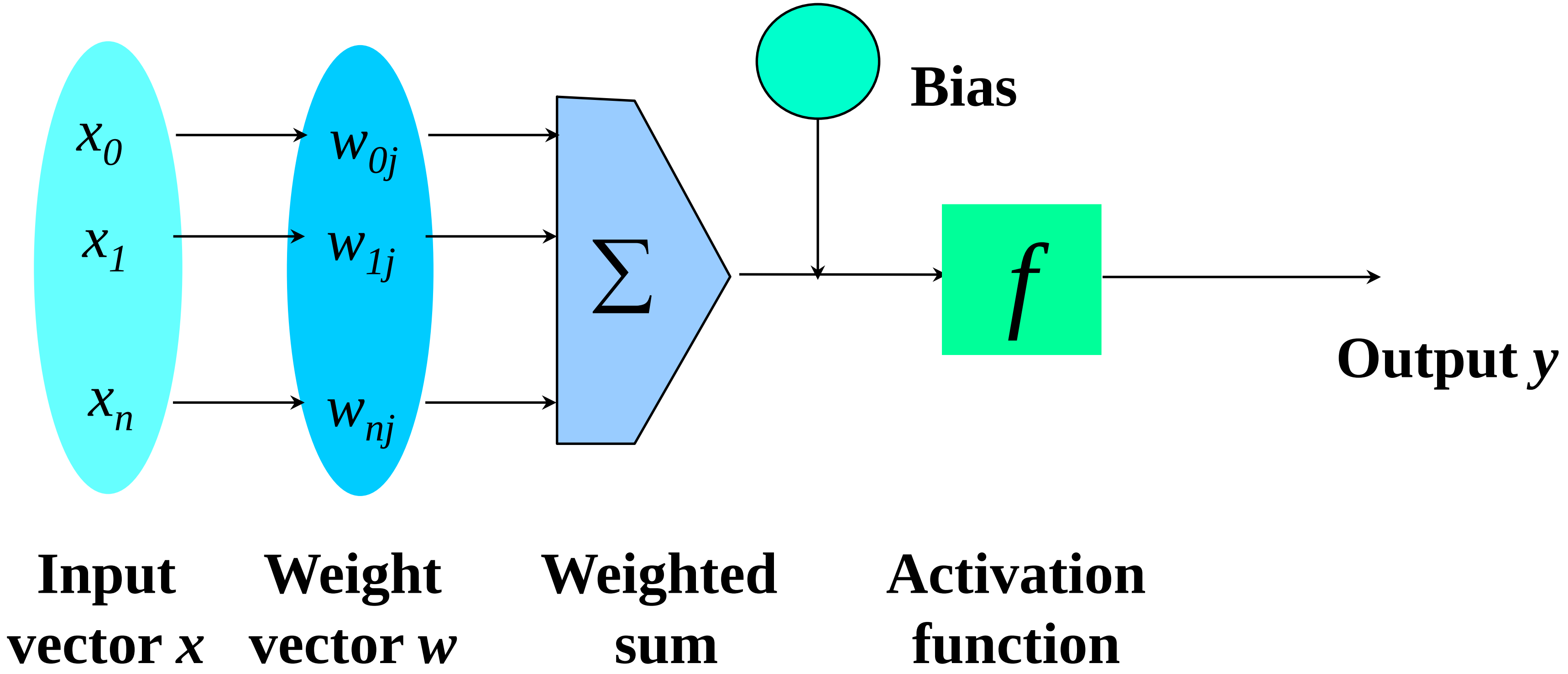
$D =$

$X_3 \overset{w_3}{-} net_3 - O_3 \equiv y_3$ $\qquad W_3 \leftarrow W_2$

$X_{10} \overset{w_{10}}{-} net_{10} - O_{10} \equiv y_{10}$

$W_{11} \leftarrow W_{10}$

OPERATION OF A NEURAL NET



WEIGHT AND BIAS UPDATION

Per Sample Updating

- updating weights and biases after the presentation of each sample.

Per Training Set Updating (Epoch or Iteration)

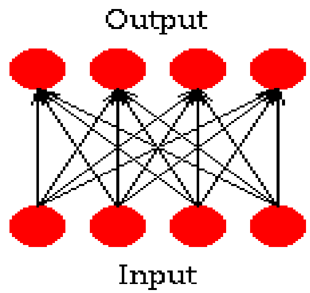
- weight and bias increments could be accumulated in variables and the weights and biases updated after all the samples of the training set have been presented.

STOPPING CONDITION

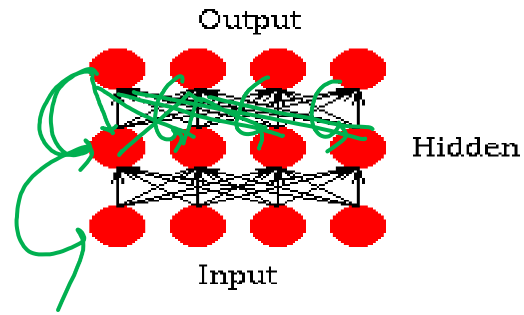
- All change in weights (Δwij) in the previous epoch are below some threshold, or
- The percentage of samples misclassified in the previous epoch is below some threshold, or
- A pre-specified number of epochs has expired.
- In practice, several hundreds of thousands of epochs may be required before the weights will converge.

BUILDING BLOCKS OF ARTIFICIAL NEURAL NET

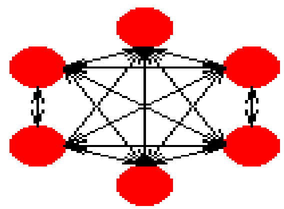
- Network Architecture (Connection between Neurons)
- Setting the Weights (Training)
- Activation Function



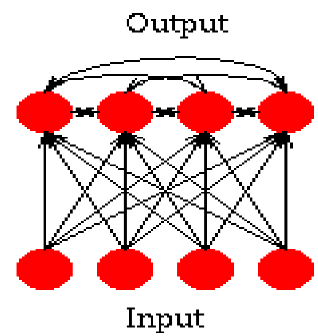
Single Layer Feedforward



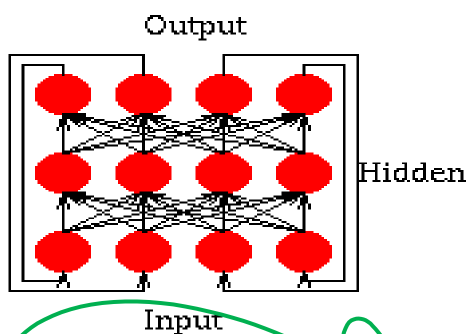
Multi Layer Feedforward



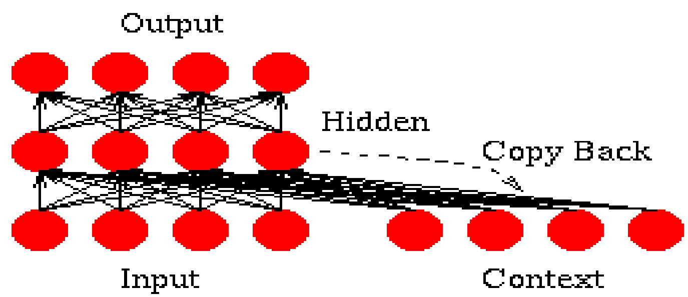
Fully Recurrent Network



Competitive Network



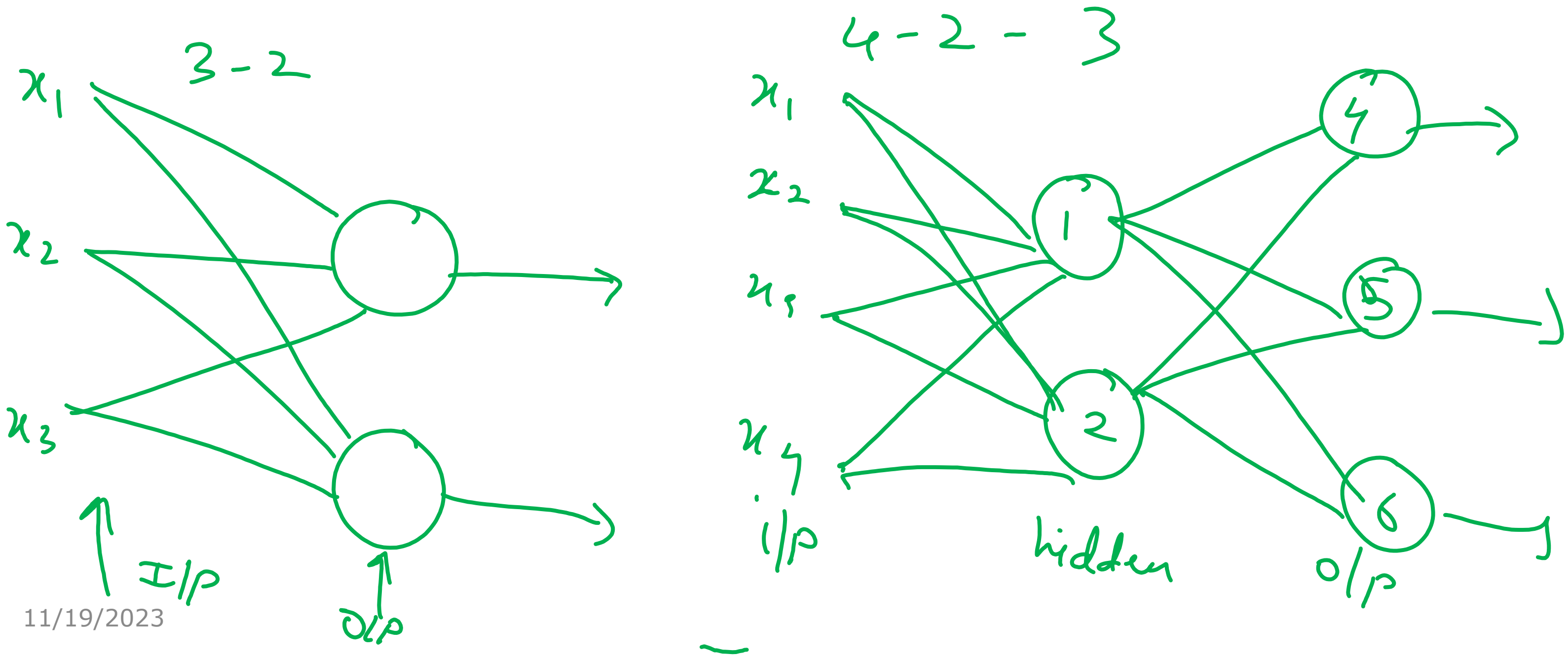
Jordan Network



Simple Recurrent Network

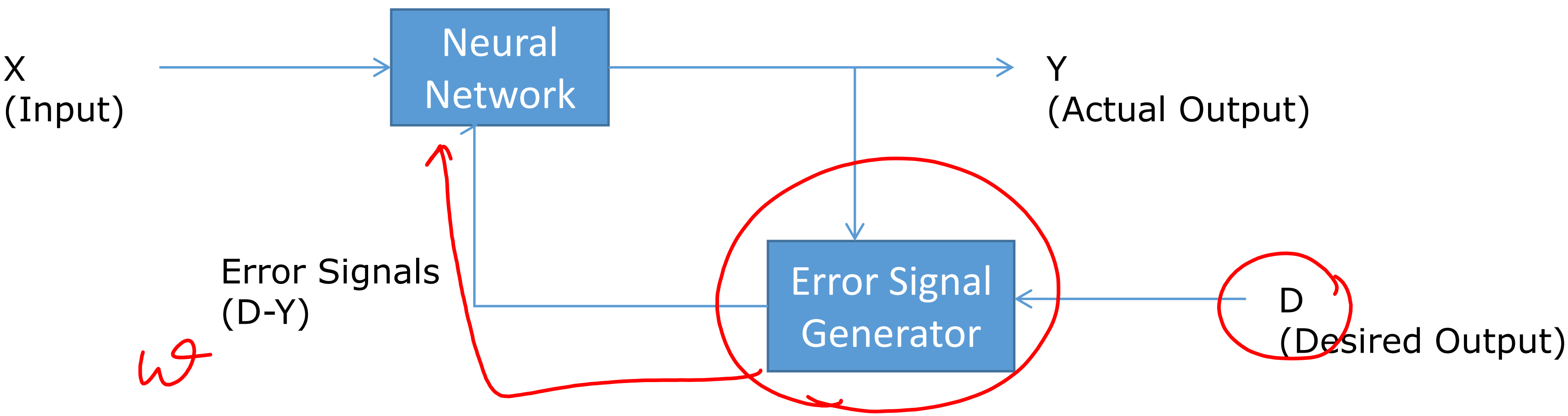
LAYER PROPERTIES

- **Input Layer:** Each input unit may be designated by an attribute value possessed by the instance.
- **Hidden Layer:** Not directly observable, provides nonlinearities for the network.
- **Output Layer:** Encodes possible values.



TRAINING PROCESS

- **Supervised Training** - Providing the network with a series of sample inputs and comparing the output with the expected responses.



w

Teacher - adapts

I/p ⟷ o/p maps — what is expected

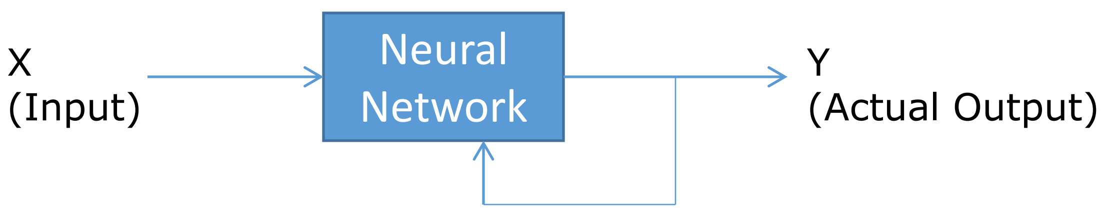
X D

X_1 x_{11} x_{12} Y_1 y_{11} y_{12}

X_2 x_{21}, x_{22} Y_2 y_{21} y_{22}

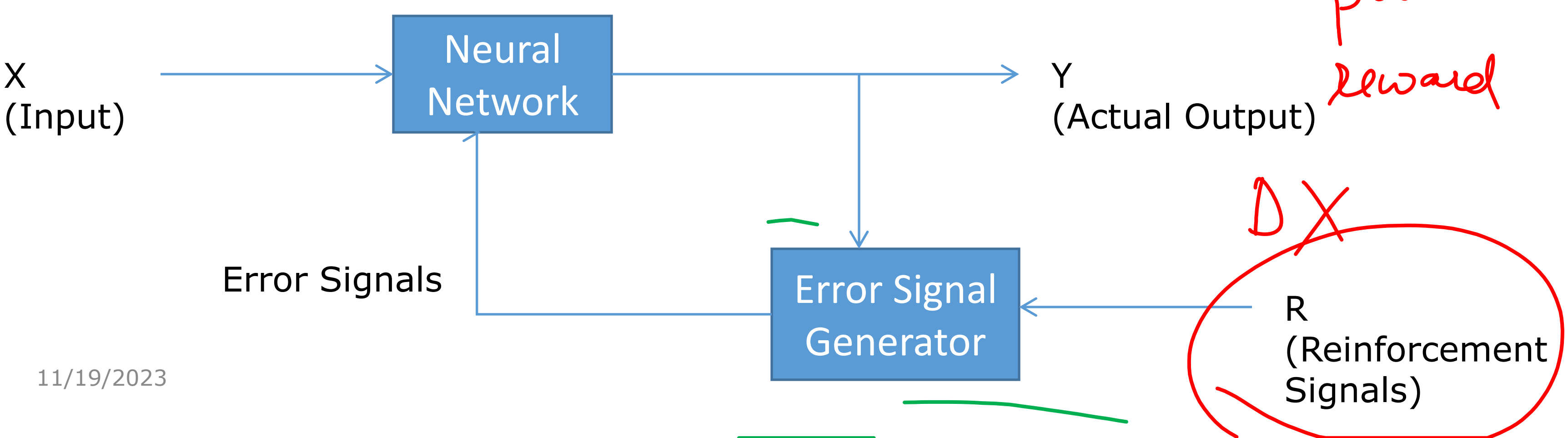
TRAINING PROCESS

- **Unsupervised Training** - Most similar input vector is assigned to the same output unit.



- **Reinforcement Training** - Right answer is not provided but indication of whether 'right' or 'wrong' is provided.

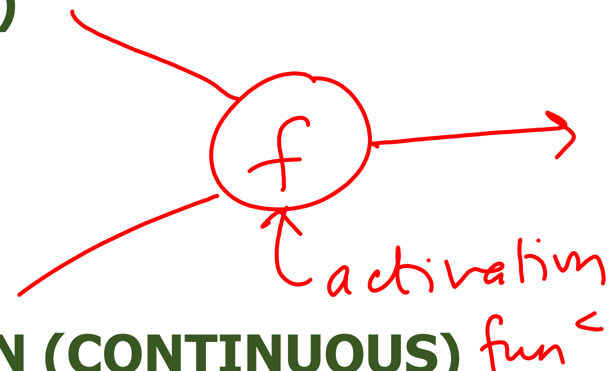
punish
reward



ACTIVATION FUNCTION

- **ACTIVATION LEVEL – DISCRETE OR CONTINUOUS**

- **HARD LIMIT FUCNTION (DISCRETE)**
  - Binary Activation function
  - Bipolar activation function
  - Identity function

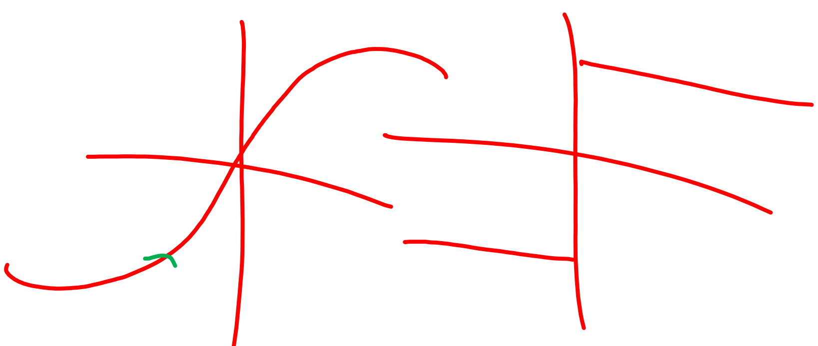


Soft limit.

- **SIGMOIDAL ACTIVATION FUNCTION (CONTINUOUS)**
  - Binary Sigmoidal activation function
  - Bipolar Sigmoidal activation function

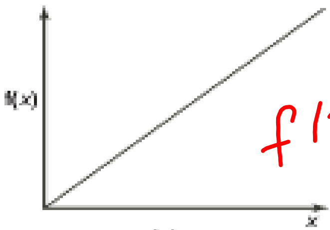
0 or 1

-1 or 1



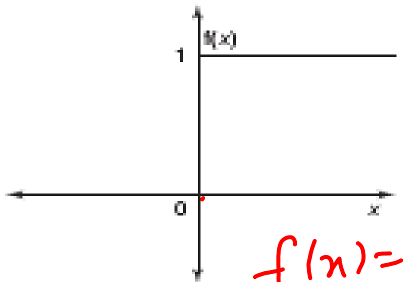
ACTIVATION FUNCTION

$f(net) = sgn(net)$



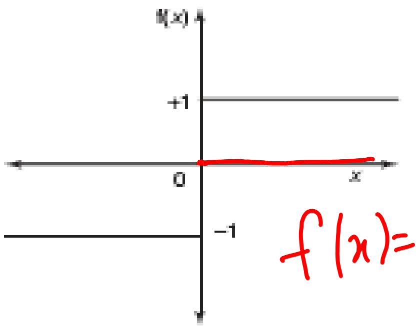
(a)

$f(x) = x$



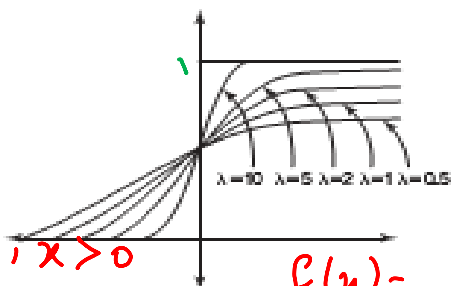
(b)

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



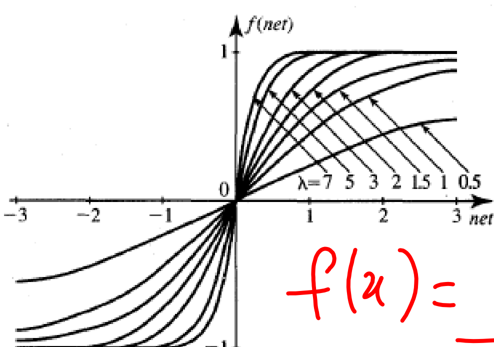
(c)

$f(x) = \begin{cases} 1, & x > 0 \\ -1, & x \leq 0 \end{cases}$

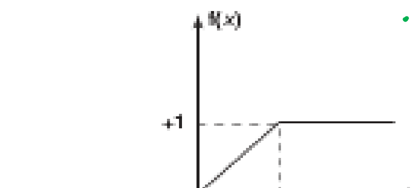


(d)

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



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



(f)

Activation functions:

(A) Identity $f(x) = x$

(B) Binary step
(unipolar)

(C) Bipolar step
$sgn(x)$

(D) Binary sigmoidal
unipolar sigmoidal

(E) Bipolar sigmoidal

(F) Ramp

$f(x) = \begin{cases} x, & x > 0 \text{ and } x \leq 1 \\ 1, & x > 1 \end{cases}$

CONSTRUCTING ANN

- Determine the network properties:
  - Network topology architecture
  - Types of connectivity
  - Order of connections fully connected
  - Weight range
- Determine the node properties:
  - Activation range
- Determine the system dynamics
  - Weight initialization scheme
  - Activation – calculating formula
  - Learning rule

PROBLEM SOLVING

- Select a suitable NN model based on the nature of the problem.
- Construct a NN according to the characteristics of the application domain.
- Train the neural network with the learning procedure of the selected model.
- Use the trained network for making inference or solving problems.

NEURAL NETWORKS

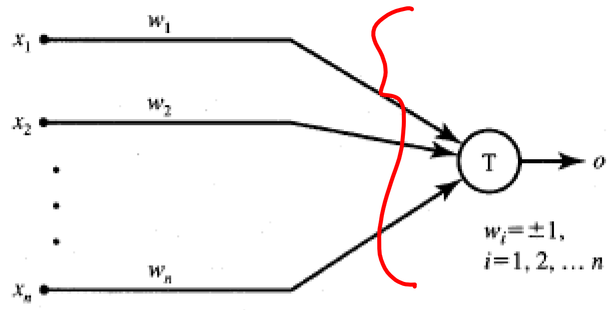
- **Neural Network** learns by adjusting the weights so as to be able to correctly classify the training data and hence, after testing phase, to classify unknown data.
- **Neural Network** needs long time for training.
- **Neural Network** has a high tolerance to noisy and incomplete data.

SALIENT FEATURES OF ANN

- Adaptive learning
- Self-organization
- Real-time operation
- Fault tolerance via redundant information coding
- Massive parallelism
- Learning and generalizing ability
- Distributed representation

McCULLOCH–PITTS NEURON (MP neuron model)

➢ First formal synthetic neuron model based on the highly simplified biological neuron



(a)

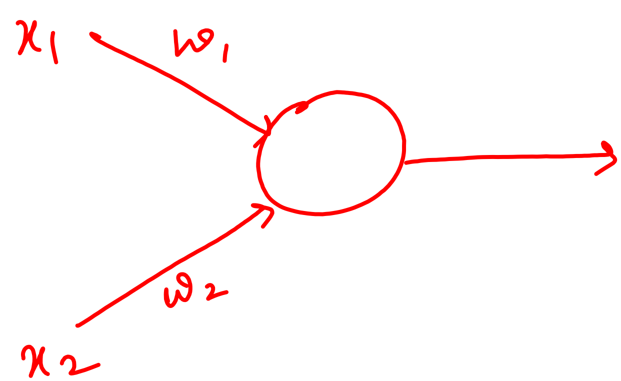
➢ The inputs are 0 or 1
➢ Outputs o is defined as

$$o^{k+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i^k < T \end{cases}$$

➢ Though simplistic the model has sufficient computing potential
➢ It can perform the basic logic operations NOT, OR, and AND, provided its weights and thresholds are appropriately selected

Design an OR gate using MP neuron model

x_1	x_2	OR
0	0	0
0	1	1
1	0	1
1	1	1



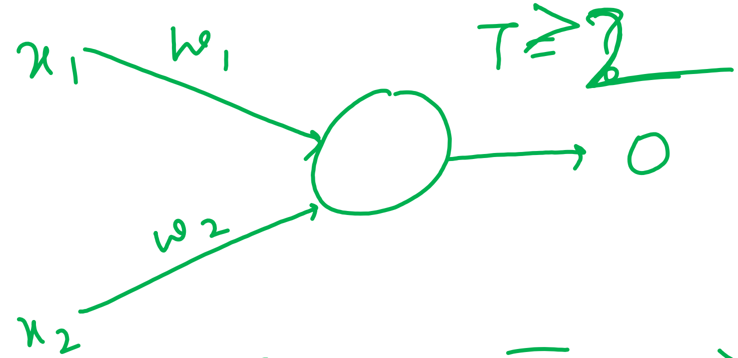
$$f\left(\sum x_i w_i\right) = \begin{cases} 0 \; ; \sum x_i w_i < \\ 1 \; ; \sum x_i w_i \geq 1 \end{cases}$$

T

Case 1: $w_1 = 1$, $w_2 = 1$

x_1	x_2	$x_1 w_1 + x_2 w_2$	D
0	0	0	0
0	1	1	0
1	0	1	D
1	1	2	1

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



$$O = \begin{cases} 1, & \sum x_i w_i \geq 2 \\ 0, & \sum x_i w_i < 2 \end{cases}$$

Case 1 $w_1 = 1$ $w_2 = 1$

x_1	x_2	y	$x_1 w_1 + x_2 w_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	2

OR

$T \geq 1$

y
0
1
1
1

XOR

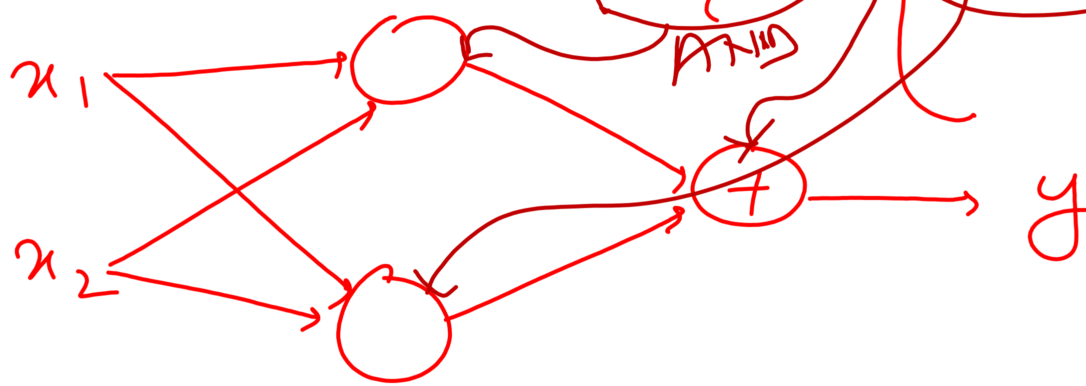
NOT

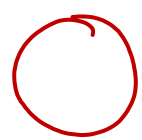
OR

$$y = \underbrace{x_1 \cdot \bar{x}_2}_{1} + \underbrace{\bar{x}_1 \cdot x_2}_{1}$$

AND

Assignment

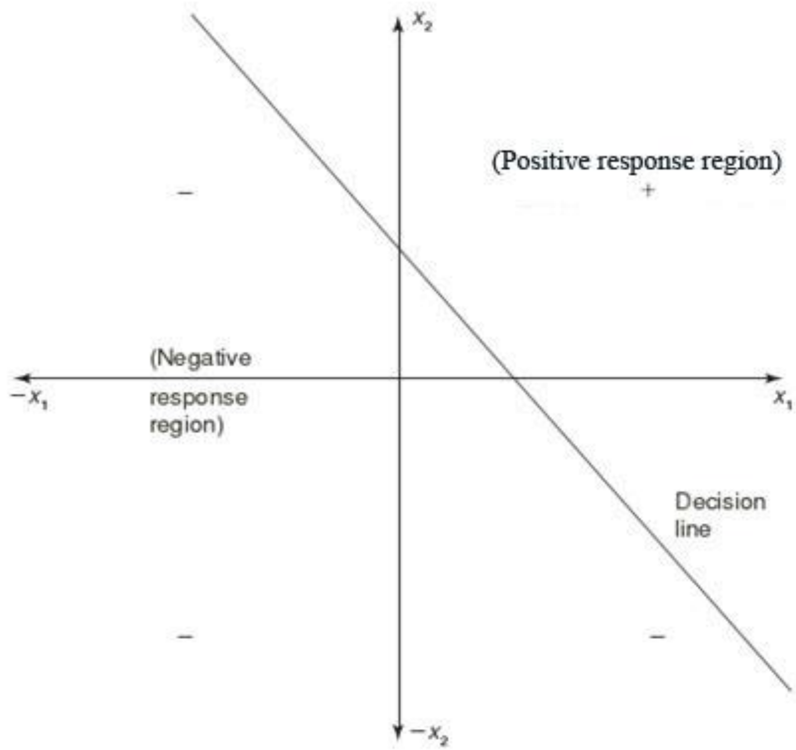




LINEAR SEPARABILITY

- Linear separability is the concept wherein the separation of the input space into regions is based on whether the network response is positive or negative.

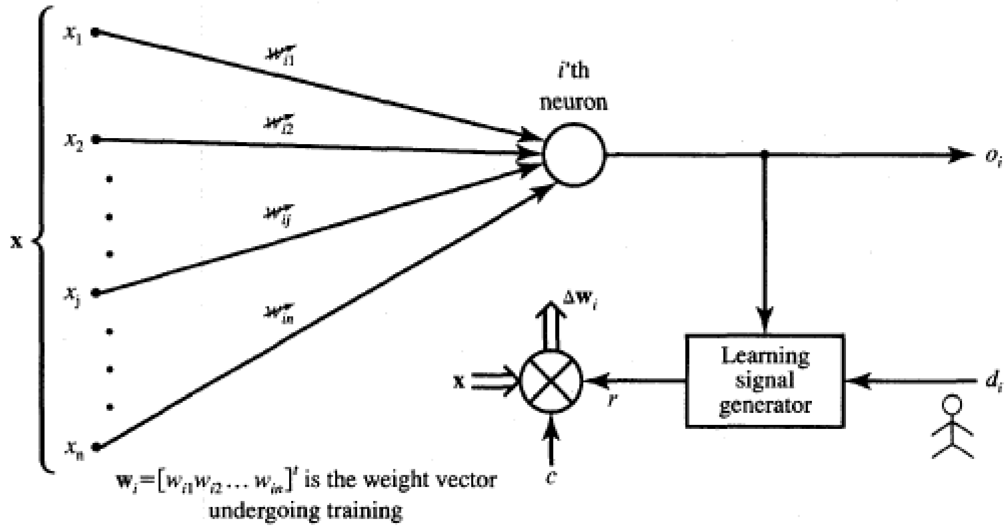
- Consider a network having positive response in the first quadrant and negative response in all other quadrants (AND function) with either binary or bipolar data, then the decision line is drawn separating the positive response region from the negative response region.



FEW APPLICATIONS OF NEURAL NETWORKS

- Aerospace
- Automotive
- Banking
- Credit Card Activity Checking
- Defense
- Electronics
- Entertainment
- Financial
- Industrial
- Insurance
- Insurance
- Manufacturing
- Medical
- Oil and Gas
- Robotics
- Speech
- Securities
- Telecommunications
- Transportation

LEARNING RULES



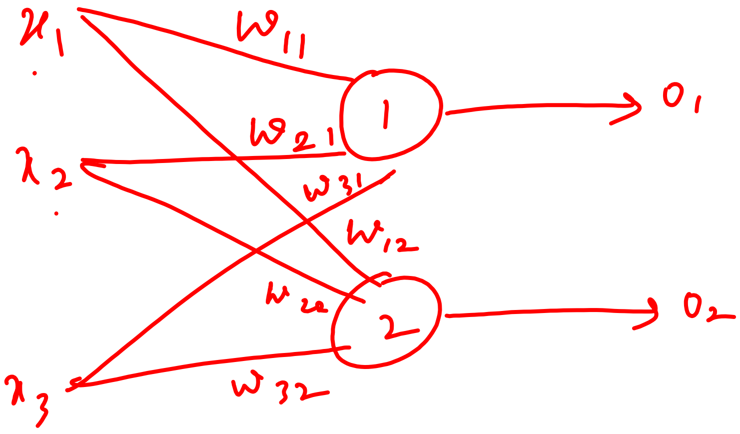
$$r = r(\mathbf{w}_i, \mathbf{x}, d_i)$$

- The learning step produces the weight vector

$$\Delta \mathbf{w}_i(t) = cr\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right] \mathbf{x}(t)$$

- The weight vector incremented at the next iteration

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + cr\left[\mathbf{w}_i(t), \mathbf{x}(t), d_i(t)\right] \mathbf{x}(t)$$



$$X = \begin{array}{c} X_1 \\ X_2 \\ \\ X_3 \end{array} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ \hline x_{21} & x_{22} & x_{23} \\ \vdots & & \\ x_{n1} & x_{n2} & x_{n3} \end{bmatrix}$$

$$X_1 - W_1 - O_1 \equiv D_1$$

$$\Delta W_1$$

$$W_2 = W_1 + \Delta W_1$$

$$d = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \hline \phi_{21} & \phi_{22} \\ \vdots & \\ \phi_{n1} & \phi_{n2} \end{bmatrix}$$

$$W_1 = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

Hebbian	$co_i x_j$ $j = 1, 2, \ldots, n$	0	U	Any	Neuron
Perceptron	$c\left[d_i - \mathrm{sgn}\,(\mathbf{w}_i^t\mathbf{x})\right] x_j$ $j = 1, 2, \ldots, n$	Any	S	Binary bipolar, or Binary unipolar*	Neuron
Delta	$c(d_i - o_i)f'(net_i)x_j$ $j = 1, 2, \ldots, n$	Any	S	Continuous	Neuron
Widrow-Hoff	$c(d_i - \mathbf{w}_i^t\mathbf{x})x_j$ $j = 1, 2, \ldots, n$	Any	S	Any	Neuron
Correlation	$cd_i x_j$ $j = 1, 2, \ldots, n$	0	S	Any	Neuron
Winner-take-all	$\Delta w_{mj} = \alpha(x_j - w_{mj})$ m-winning neuron number $j = 1, 2, \ldots, n$	Random Normalized	U	Continuous	Layer of p neurons
Outstar	$\beta(d_i - w_{ij})$ $i = 1, 2, \ldots, p$	0	S	Continuous	Layer of p neurons

c, α, β are positive learning constants
S — supervised learning, U — unsupervised learning
* — Δw_{ij} not shown

Learning Rules

1.} Hebbian learning rule

Type of learning — Unsupervised

Neuron characteristics — Any (discrete or continuous)

Neuron / layer — Neuron or layer

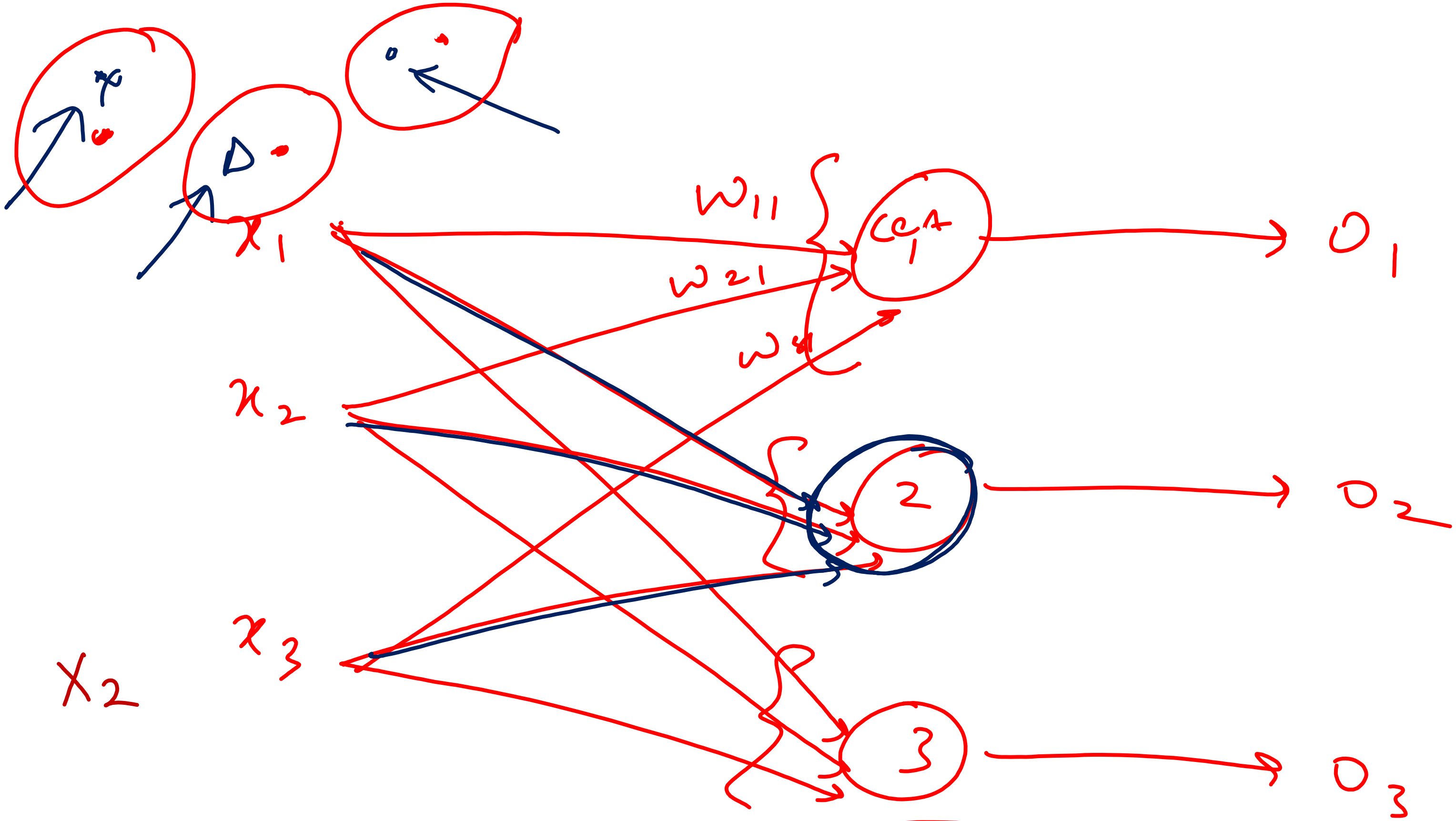
Initial weights — 0

Weight change formula — $c\,o\,x$

input

learning rate constant

actual output



$$X_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$W = \begin{array}{c} 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \\ w_{13} & w_{23} & w_{33} \end{bmatrix} \begin{array}{c} W_1 \\ W_2 \\ W_3 \end{array}$$

3×3

$$\left\{ \begin{array}{l} X_1 - W_1 \\ X_1 - W_2 \\ X_1 - W_3 \end{array} \right\} \min$$

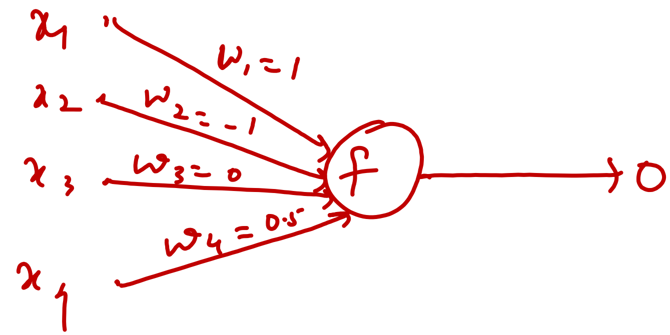
$$X_1 = \begin{bmatrix}1\\-2\\1.5\\0\end{bmatrix} \quad X_2 = \begin{bmatrix}1\\-0.5\\-2\\-1.5\end{bmatrix} \quad X_3 = \begin{bmatrix}0\\1\\-1\\1.5\end{bmatrix} \quad W_1 = \begin{bmatrix}1\\-1\\0\\0.5\end{bmatrix} \quad c = 1$$

$$f(net) = \begin{cases} 1 & net_i \geq 1 \\ -1 & \end{cases} \qquad f(net) = \frac{2}{1+e^{(-\lambda net)}} - 1$$

$\underline{f(net) = sgn(net)}$ be the activation fun^c

Calculate the weight vector ~~after~~ upto first iteration

Soln) Step 1 : $X_1 = \begin{bmatrix}1\\-2\\1.5\\0\end{bmatrix}_{4\times1}$



$W_1 = \begin{bmatrix}1\\-1\\0\\0.5\end{bmatrix}_{4\times1}$

$1\times4 \quad 4\times1 \quad - \quad 1$

$4\times1 \quad 1\times4 \quad - \quad 4$

$$net_1 = X_1^T \cdot W_1 = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4$$
$$= 1 + 2 + 0 + 0 = 3$$
$$O_1 = f(net_1) = f(3) = 1$$

$$\Delta W_1 = c\, O_1 X_1$$

$$= 1 \cdot 1 \cdot \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}$$

$$W_2 = W_1 + \Delta W_1$$

$$= \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix}$$

Step 2: $X_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix}$ $\quad W_2 = \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix}$

$$net_2 = X_2^T \cdot W_2 = 1 \times 2 + (-0.5 \times -3) + (-2 \times 1.5) + (-1.5 \times 0.5)$$

$$= 2 + 1.5 - 3 - 0.75 = \cancel{8} - 0.25$$

$$O_2 = f(net_2) = f(-0.25) = -1$$

$$\Delta W_2 = c\, O_2\, X_2 = 1 * (-1) * X_2 = -X_2 =$$

$$W_3 = W_2 + \Delta W_2 = W_2 - X_2$$

$$= \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -2.5 \\ 3.5 \\ 2 \end{bmatrix}$$

Step 3: $X_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$ $\quad W_3 = \begin{bmatrix} 1 \\ -2.5 \\ 3.5 \\ 2 \end{bmatrix}$

$$net_3 = X_3^T W_3 = \begin{bmatrix} 0 & 1 & -1 & 1.5 \end{bmatrix} \begin{bmatrix} 1 \\ -2.5 \\ 3.5 \\ 2 \end{bmatrix}$$

$$= 0 - 2.5 - 3.5 + 3 = -3$$

$$O_3 = f(net_3) = f(-3) = -1$$

$$W_4 = W_3 + \Delta W_3 = W_3 - X_3 = \begin{bmatrix} 1 \\ -3.5 \\ 4.5 \\ 0.5 \end{bmatrix}$$

	2] Perceptron Learning Rule:	3] Delta LR
Type of learning	– Supervised	Supervised
Neuron characteristics	– discrete	continuous
Neuron / layer	– Neuron or layer	Neuron
Initial weights	– Any	Any
weight change formula	– $c(d-o)X$	$c(d-o)f'(net)X$

d – desired o/p

o – actual o/p

$\pm 2cX$

bipolar – $f(net) = sgn(net)$

1 or −1

sgn(net) **Learning Rules** Zurada

, $O_i = -1$

$+2cX$

$:= -1 \quad O_i = 1$

1.] Perception L R

Type of learning — Supervised

Type of neuron — Discrete (Unipolar / Bipolar)

Neuron / layer — neuron

Weight change — $\Delta W_i = c[d_i - O_i]X$

$\Delta w_{ij} = c[d_i - O_i]x_j$

Initial weights — any

$\pm 2cX$

Let $X_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$, $X_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}$ $X_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$, $W_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$

$d_1 = -1$, $d_2 = -1$, $d_3 = 1$

$c = 0.1$

Using Perceptron LR find weights after one epoch (iteration).

Step 1: $X_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$ $W_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$

$$net_1 = X^T \cdot W$$

$$= \begin{bmatrix} 1 & -2 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} = 2.5$$

$$O_1 = f(net_1) = sgn(net_1) = 1$$

$$\Delta W_1 = c(d_1 - o_1) X_1$$

$$= 0.1(-1-1) X_1$$

$$= \begin{bmatrix} -0.2 \\ 0.4 \\ 0 \\ 0.2 \end{bmatrix}$$

$$W_2 = W_1 + \Delta W_1 = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix}$$

Step 2: $X_2 =$ $\qquad$ $W_2 =$

$$net_2 = -1.6$$

$$O_2 = -1$$

Now, $d_2 = -1$ $\therefore$ No correction/change in weights

$\therefore W_3 = W_2$

Step 3: $X_3 =$ $\qquad$ $W_3 =$

$$net_3 = -2.1$$

$$O_3 = -1$$

$$\Delta W_3 = \begin{bmatrix} -0.2 \\ 0.2 \\ 0.1 \\ -0.2 \end{bmatrix}$$

$$W_4 = W_3 + \Delta W_3$$

$$= \begin{bmatrix} 0.6 \\ -0.4 \\ 0.1 \\ 0.5 \end{bmatrix}$$

Delta LR

weight change

$$\Delta w_i = c\,(d_i - o_i)\,f'(net_i)\,X_i$$

To calculate &

~~The~~

$$E \triangleq \frac{1}{2}(d_i - o_i)^2$$

$\frac{1}{2}(-2)(d - o)$
$f'(wx) \cdot X$

$\frac{\partial E}{\partial W}$ i.e.

$$E = \frac{1}{2}\left[d_i - f(w_i X)\right]^2$$

we obtain the error gradient vector value

$$\nabla E = -(d_i - o_i) \cdot f'(w_i X) \cdot X$$

$$\Delta w_i = -\eta \nabla E \qquad -\eta \nabla E$$

where η is a positive constant

i.e. $\Delta w_i = \eta (d_i - o_i) f'(w_i x) \cdot x$

$$\Delta w_i = \eta (d_i - o_i) f'(net_i) \cdot \cancel{x}$$

For Unipolar sigmoidal activation fun^c

$$f(net) = \frac{1}{1 + e^{(-\lambda net)}} = o \quad \text{--- (1)}$$

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

lets $\cancel{\lambda} = 1$

$$f'(net) = \frac{e^{(-net)}}{\left[1 + e^{(-net)}\right]^2}$$

$$f'(net) = \frac{1}{(1+e^{-net})^2} \cdot e^{-net}$$

$$f'(net) = \left[\frac{1}{1 + e^{(-net)}}\right] \cdot \left[\frac{1 + e^{(-net)} - 1}{1 + e^{(-net)}}\right]$$

$o(1-o)$

$$f'(net) = o(1-o)$$ — unipolar sigmoidal

For bipolar sigmoidal activation fun^n

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

Let $\lambda = 1$

$$\therefore \quad f(net) = \frac{2}{1+e^{(-net)}} - 1 = o \quad \text{— (1)}$$

Taking partial derivative w.r.t net

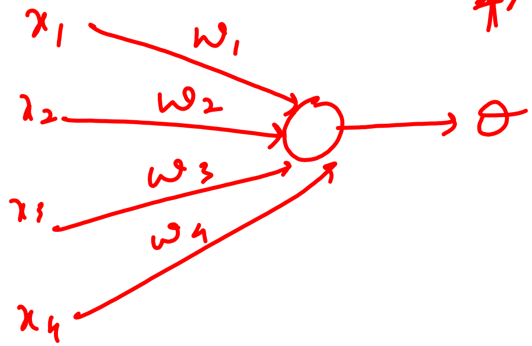
$$f'(net) = \frac{2e^{(-net)}}{\left[1+e^{(-net)}\right]^2}$$

$$f'(net) = \frac{1}{2}\left[1-o^2\right]$$

$$X_1 = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}_{4\times 1} \quad X_2 = \begin{bmatrix} 1.0 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} \quad X_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix} \quad W_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}_{4\times 4}$$

$c = 1$, $f(net) = sgn(net)$

Step 1 : Consider input X_1 & W_1



$$net_1 = \sum x_i w_i = X_1 W_1$$

$$= 1\times 1 + (-2\times -1) + (1.5\times 0) + (0\times 0.5)$$

$$\left(\begin{bmatrix} 1 & -2 & 1.5 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} \right.$$

$$net_1 = 3$$

$$O_1 = f(net_1)$$

$$= 1$$

$$f(net) = sgn(net) = \begin{cases} 1, & net \geq 1 \\ -1, & net < 1 \end{cases}$$

$$\Delta W_1 = c\, O_1 X_1$$

$$= 1\times 1\times \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}$$

$$W_2 = W_1 + \Delta W_1$$

$$= \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix}$$

Step 2 : $X_2 = \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix}$

$$net_2 = X_2 W_2 = \begin{bmatrix} 1 & -0.5 & -2 & -1.5 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix}$$

$$= -0.25$$

$$O_2 = f(net_2) = -1$$

$$W_3 = W_2 + \Delta W_2 = W_2 + c O_2 X_2$$

$$= \begin{bmatrix} 2 \\ -3 \\ 1.5 \\ 0.5 \end{bmatrix} + (1) \times (-1) \times \begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix} = \begin{bmatrix} 1 \\ -2.5 \\ 3.5 \\ 2.0 \end{bmatrix}$$

Step 3: $X_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$

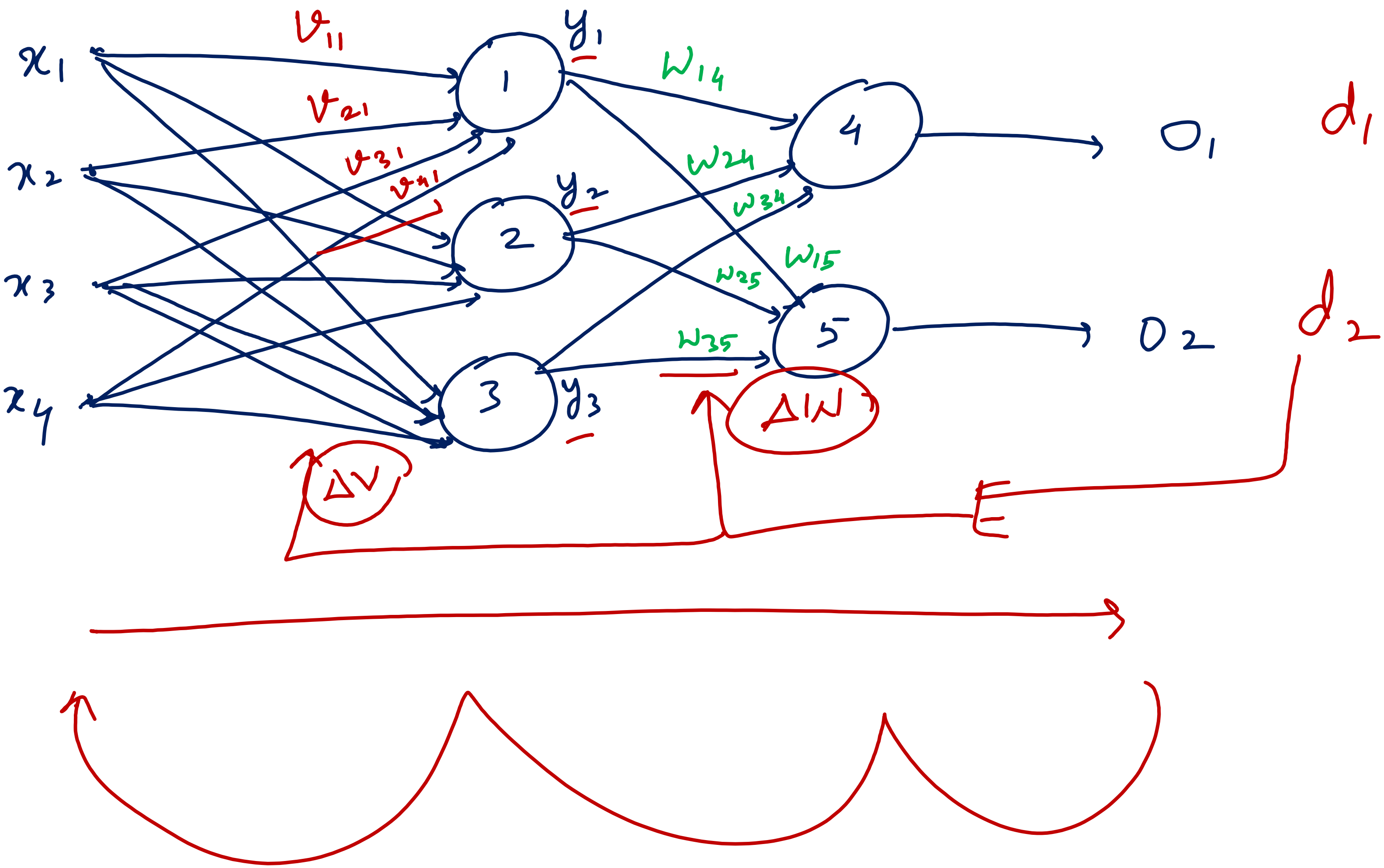
$$net_3 = X_3^T W_3 = -3$$

$$O_3 = f(net_3) = -1$$

$$W_4 = W_3 + \Delta W_3 = W_3 + c\, O_3 X_3$$

$$= \begin{bmatrix} 1 \\ -2.5 \\ 3.5 \\ 0 \end{bmatrix} + (1) \times (-1) \times \begin{bmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{bmatrix}$$

$$W_4 = \begin{bmatrix} 1 \\ -3.5 \\ +4.5 \\ 0.5 \end{bmatrix}$$



x_1
x_2
x_3
x_4
v_{11}
v_{21}
v_{31}
v_{41}
1
2
3
y_1
y_2
y_3
W_{14}
W_{24}
W_{34}
W_{15}
W_{25}
W_{35}
4
5
O_1
O_2
d_1
d_2
ΔW
ΔV
E