Artificial Neural Networks

UNIT 2

Learning Algorithms

Learning

- It's a process by which a NN adapts itself to a stimulus by making proper parameter adjustments, resulting in the production of desired response
- Two kinds of learning
  - Parameter learning:- connection weights are updated
  - Structure Learning:- change in network structure

Training

- The process of modifying the weights in the connections between network layers with the objective of achieving the expected output is called training a network.

- This is achieved through
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

Classification of learning

- Supervised learning:-
  - Learn to predict an output when given an input vector.
- Unsupervised learning
  - Discover a good internal representation of the input.
- Reinforcement learning
  - Learn to select an action to maximize payoff.

Supervised Learning

- Each input vector requires a corresponding target vector.
- Training pair=[input vector, target vector]

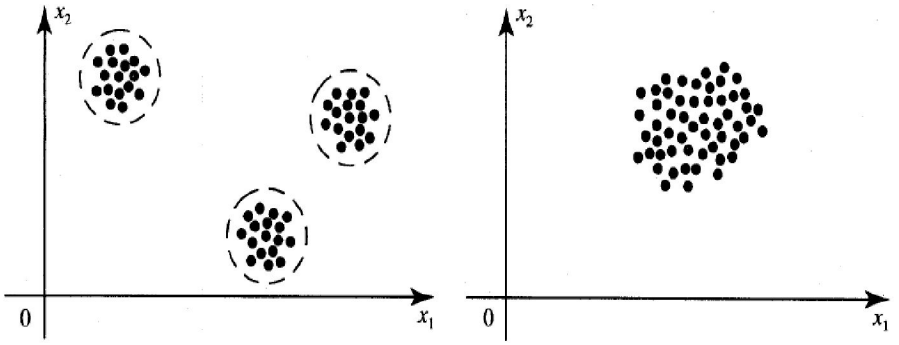
Two types of supervised learning

- Each training case consists of an input vector x and a target output t.

- Regression: The target output is a real number or a whole vector of real numbers.
  - The price of a stock in 6 months time.
  - The temperature at noon tomorrow.

- Classification: The target output is a class label.
  - The simplest case is a choice between 1 and 0.
  - We can also have multiple alternative labels.

Unsupervised Learning

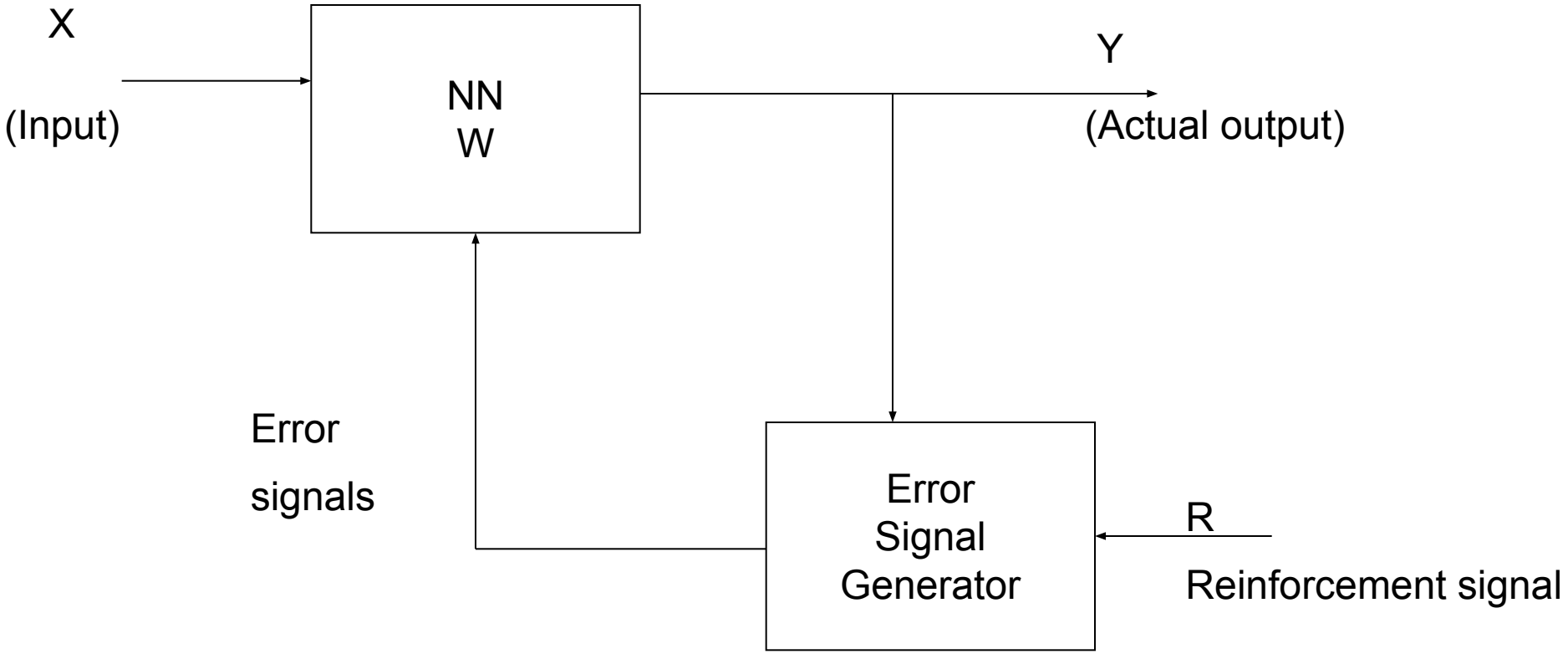


- How a fish or tadpole learns
- All similar input patterns are grouped together as clusters.
- If a matching input pattern is not found a new cluster is formed
- One major aim is to create an internal representation of the input that is useful for subsequent supervised or reinforcement learning.
- It provides a compact, low-dimensional representation of the input.

Self-organizing

- In unsupervised learning there is no feedback
- Network must discover patterns, regularities, features for the input data over the output
- While doing so the network might change in parameters
- This process is called self-organizing

Reinforcement Learning



When Reinforcement learning is used?

- If less information is available about the target output values (critic information)
- Learning based on this critic information is called reinforcement learning and the feedback sent is called reinforcement signal
- Feedback in this case is only evaluative and not instructive

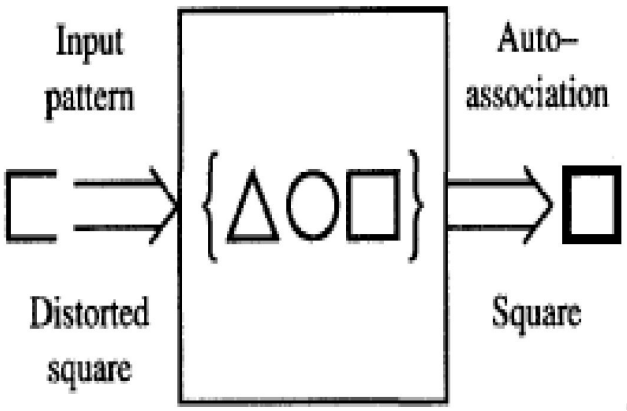
Some learning algorithms we will learn are

- Supervised:
  - Adaline, Madaline
  - Perceptron
- Unsupervised
  - Competitive Learning
  - Kohenen self organizing map
  - Learning vector quantization
  - Hebbian learning

Neural processing

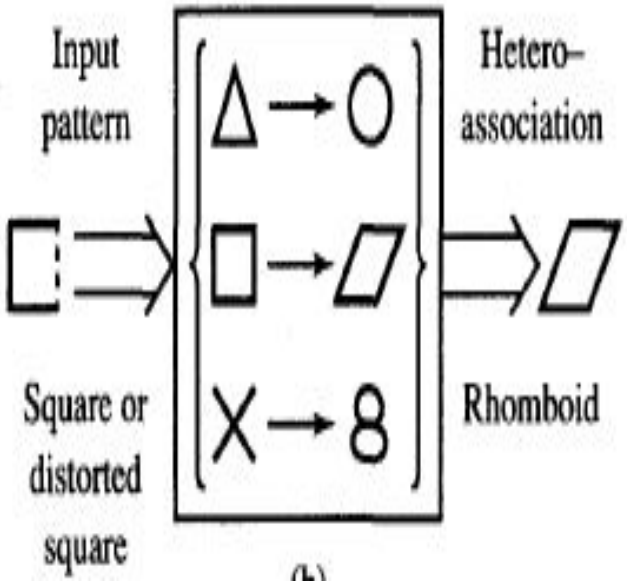
- **Recall:-** processing phase for a NN and its objective is to retrieve the information. The process of computing **o** for a given **x**
- **Basic forms of neural information processing**
  - Auto association
  - Hetero association
  - Classification

Neural processing-Autoassociation



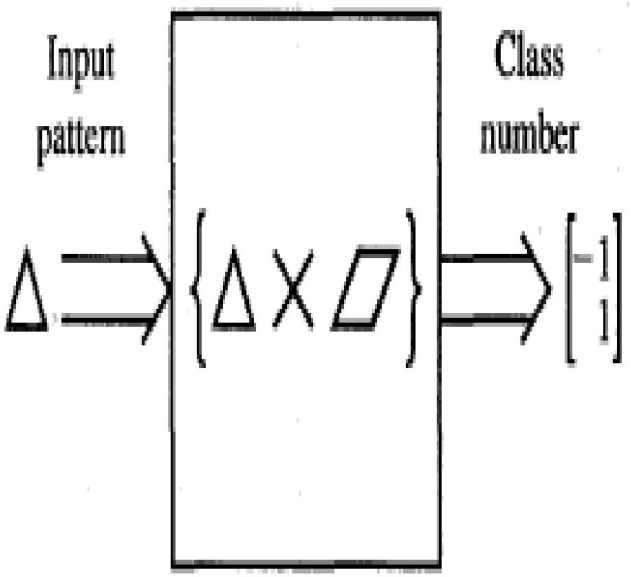
- Set of patterns can be stored in the network
- If a pattern similar to a member of the stored set is presented, an association with the input of closest stored pattern is made

Neural Processing- Heteroassociation



- Associations between pairs of patterns are stored
- Distorted input pattern may cause correct heteroassociation at the output

Neural processing-Classification



- Set of input patterns is divided into a number of classes or categories
- In response to an input pattern from the set, the classifier is supposed to recall the information regarding class membership of the input pattern.

Important terminologies of ANNs

- Weights
- Bias
- Threshold
- Learning rate

Weights

- Each neuron is connected to every other neuron by means of directed links
- Links are associated with weights
- Weights contain information about the input signal and is represented as a matrix
- Weight matrix also called connection matrix

Weight matrix

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ w_3^T \\ . \\ . \\ . \\ . \\ w_n^T \end{bmatrix} = \begin{bmatrix} w_{11} w_{12} w_{13} \cdots w_{1m} \\ w_{21} w_{22} w_{23} \cdots w_{2m} \\ \cdots\cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots\cdots \\ w_{n1} w_{n2} w_{n3} \cdots w_{nm} \end{bmatrix}$$

Weights contd…

- w_{ij} _ is the weight from processing element *"i"* (source node) to processing element *"j"* (destination node)

$$y_{inj} = \sum_{i=0}^{n} x_i w_{ij}$$

$$= x_0 w_{0j} + x_1 w_{1j} + x_2 w_{2j} + ... + x_n w_{nj}$$

$$= w_{0j} + \sum_{i=1}^{n} x_i w_{ij}$$

$$y_{inj} = b_j + \sum_{i=1}^{n} x_i w_{ij}$$

Activation Functions

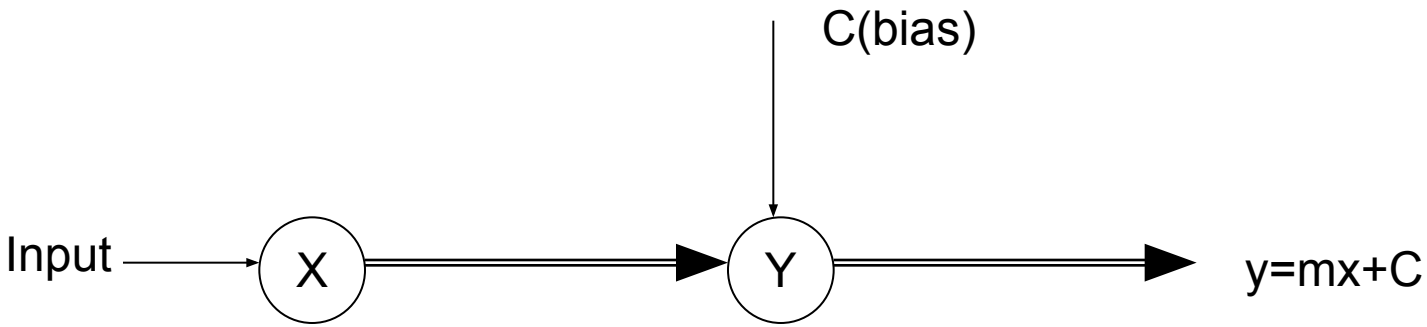
- Used to calculate the output response of a neuron.
- Sum of the weighted input signal is applied with an activation to obtain the response.
- Activation functions can be linear or non linear
- Already dealt
  - Identity function
  - Single/binary step function
  - Discrete/continuous sigmoidal function.

Bias

- Bias is like another weight. Its included by adding a component $x_0=1$ to the input vector X.
- $X=(1,X_1,X_2 \ldots X_i, \ldots X_n)$
- Bias is of two types
  - Positive bias: increase the net input
  - Negative bias: decrease the net input

Why Bias is required?

- The relationship between input and output given by the equation of straight line y=mx+c



Threshold

- Set value based upon which the final output of the network may be calculated
- Used in activation function
- The activation function using threshold can be defined as

$$f(net) = \begin{cases} 1 if net \geq \theta \\ -1 if net < \theta \end{cases}$$

Learning rate

- Denoted by α.
- Used to control the amount of weight adjustment at each step of training
- Learning rate ranging from 0 to 1 determines the rate of learning in each time step

Bidirectional associative memory (BAM)

- The Hopfield network represents an auto associative type of memory - it can retrieve a corrupted or incomplete memory but cannot associate this memory with another different memory.
- Human memory is essentially associative. One thing may remind us of another, and that of another, and so on. We use a chain of mental associations to recover a lost memory. If we forget where we left an umbrella, we try to recall where we last had it, what we were doing, and who we were talking to. We attempt to establish a chain of associations, and thereby to restore a lost memory.

- To associate one memory with another, we need a recurrent neural network capable of accepting an input pattern on one set of neurons and producing a related, but different, output pattern on another set of neurons.
- Bidirectional associative memory (BAM), first proposed by Bart Kosko, is a heteroassociative network. It associates patterns from one set, set A, to patterns from another set, set B, and vice versa. Like a Hopfield network, the BAM can generalise and also produce correct outputs despite corrupted or incomplete inputs.

The basic idea behind the BAM is to store pattern pairs so that when n-dimensional vector X from set A is presented as input, the BAM recalls m-dimensional vector Y from set B, but when Y is presented as input, the BAM recalls X.

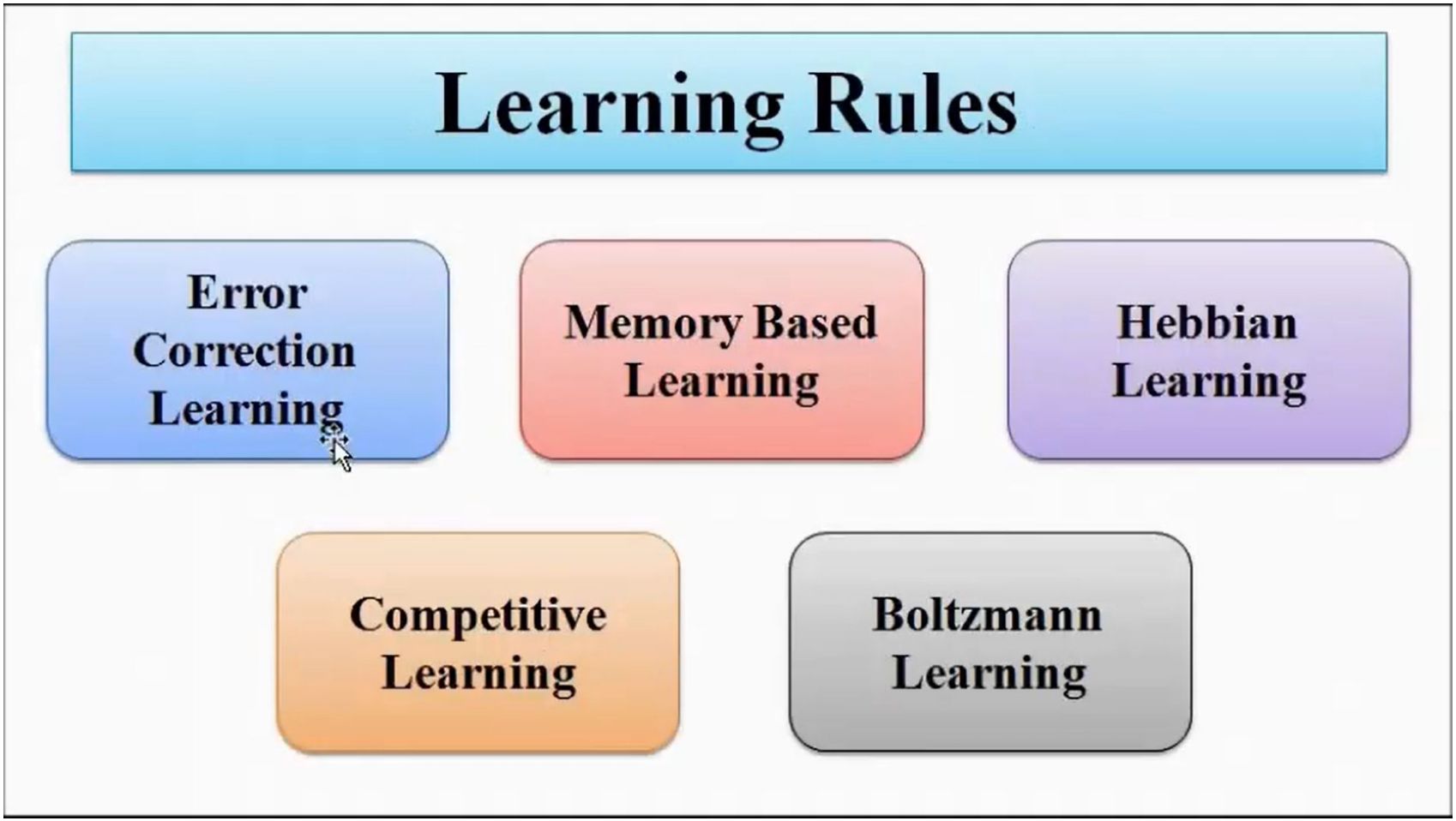
- To develop the BAM, we need to create a correlation matrix for each pattern pair we want to store. The correlation matrix is the matrix product of the input vector X, and the transpose of the output vector Y^T . The BAM weight matrix is the sum of all correlation matrices, that is,

$$\mathbf{W} = \sum_{p=1}^{P} \mathbf{X}_p \ \mathbf{Y}_p^T$$

where p is the number of pattern pairs to be stored in the BAM.

Stability and storage capacity of the BAM

- The BAM is unconditionally stable. This means that any set of associations can be learned without risk of instability.
- The maximum number of associations to be stored in the BAM should not exceed the number of neurons in the smaller layer.
- The more serious problem with the BAM is incorrect convergence. The BAM may not always produce the closest association. In fact, a stable association may be only slightly related to the initial input vector.

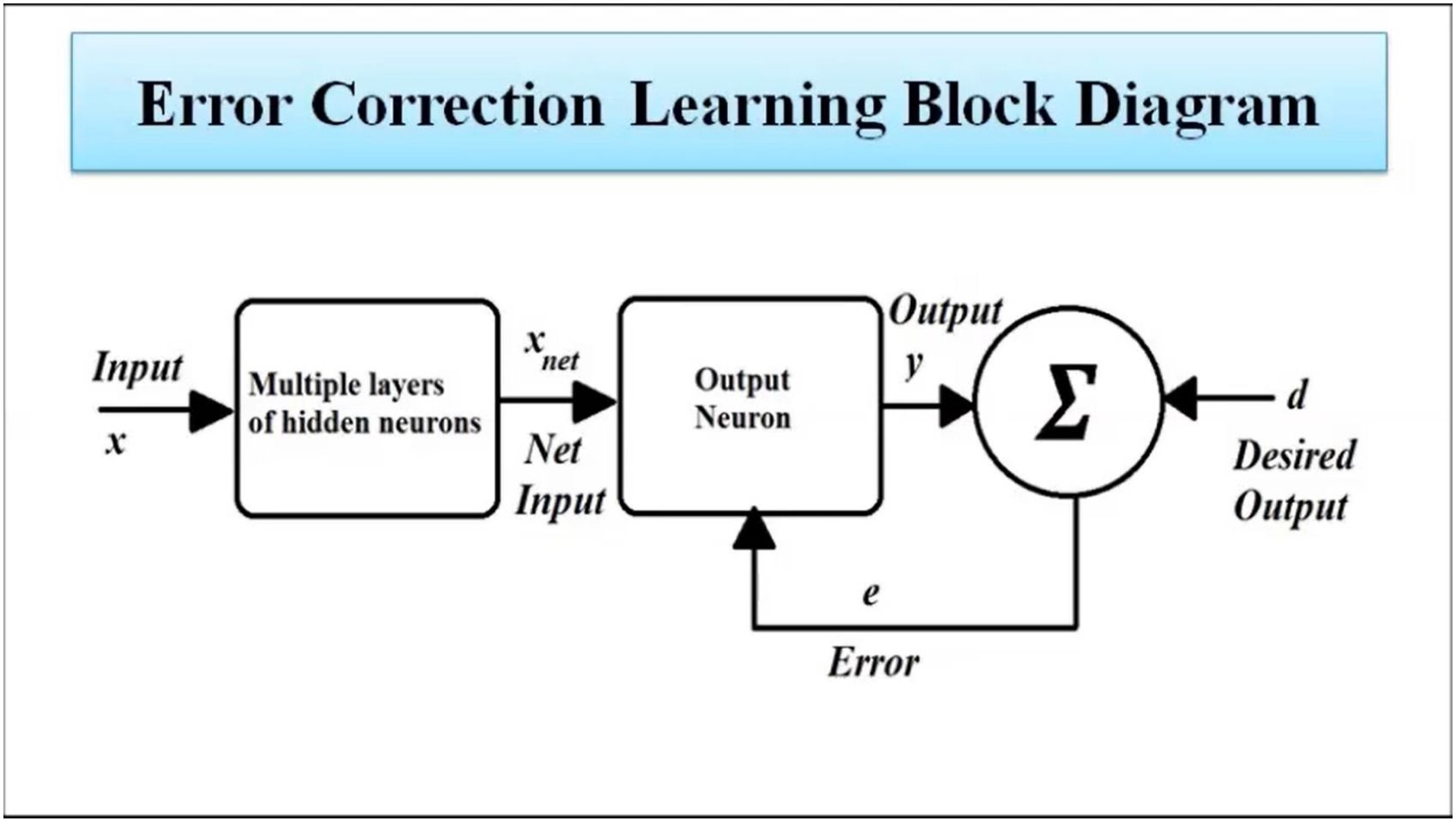

Learning Rules
Error Correction Learning
Memory Based Learning
Hebbian Learning
Competitive Learning
Boltzmann Learning

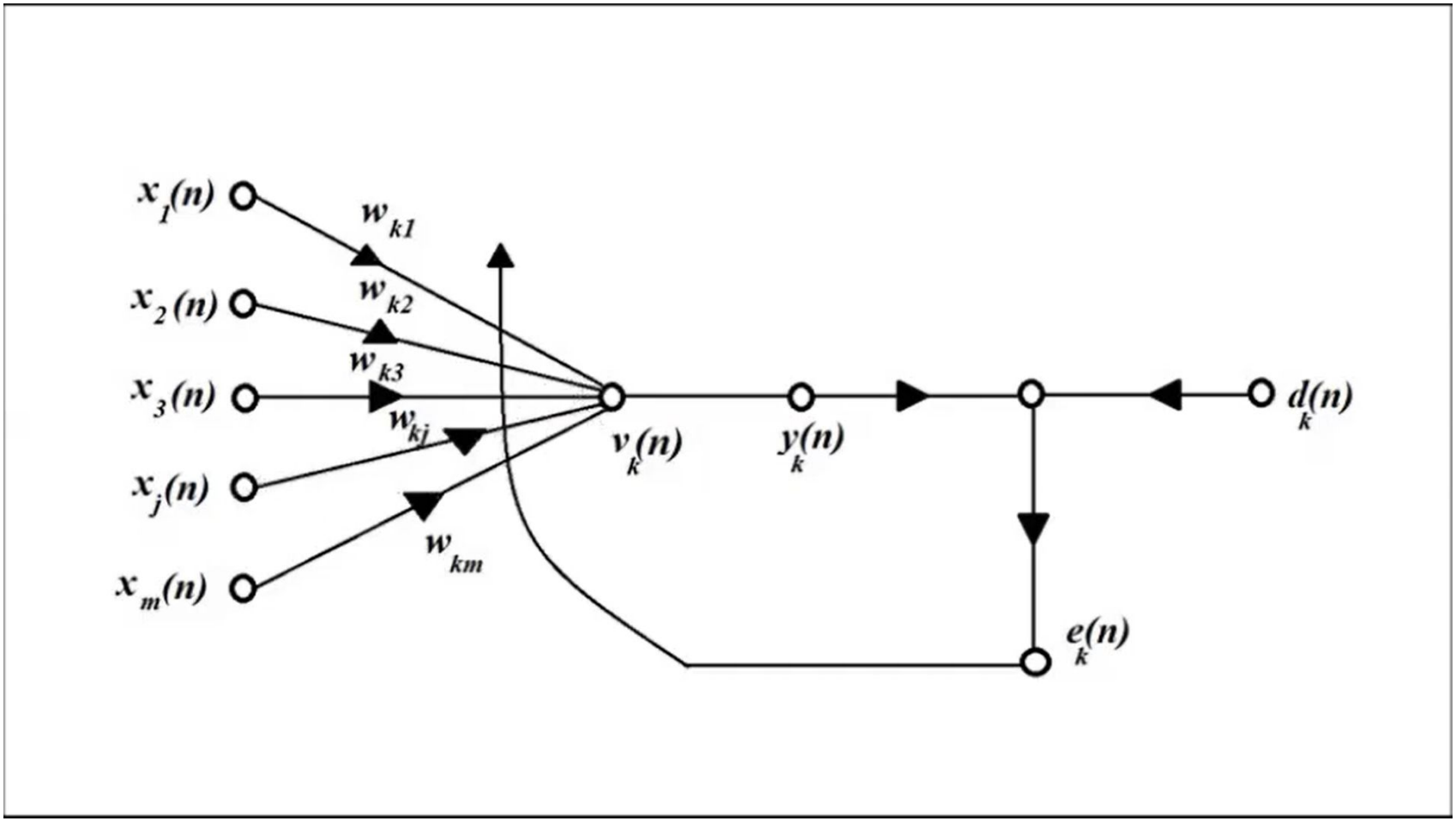
Error Correction Learning

- *Error correction learning* consists of :
  - *Input and output layers of neurons*
  - *Input signal*
  - *Comparator*
  - *Desired or target output*

- It is a *parameter based learning.*

Error Correction Learning




$x_1(n)$
$x_2(n)$
$x_3(n)$
$x_j(n)$
$x_m(n)$
w_{k1}
w_{k2}
w_{k3}
w_{kj}
w_{km}
$v_k(n)$
$y_k(n)$
$d_k(n)$
$e_k(n)$

Important Terms and Parameters associated with Neural Network...

Error Signal

- Depending on the output and desired or target input the error signal is given by,

$$e_k(n) = d_k(n) - y_k(n)$$

Basic Control Mechanism

- The ***error signal***, $e_k(n)$ activates a ***control mechanism*** by which ***the synaptic weights associated with the output neurons are changed***.

- The ***change in synaptic weights*** is given by,

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

where,

$$\eta = rate\ of\ learning$$
$$(a\ positive\ constant)$$

$$x_j = input\ signal\ of\ j^{th}\ neuron$$

$$e_k = error\ signal$$

$$\Delta w_{kj} = change\ in\ synaptic\ weight$$

Hebbian Learning

Hebb's Postulate of Learning

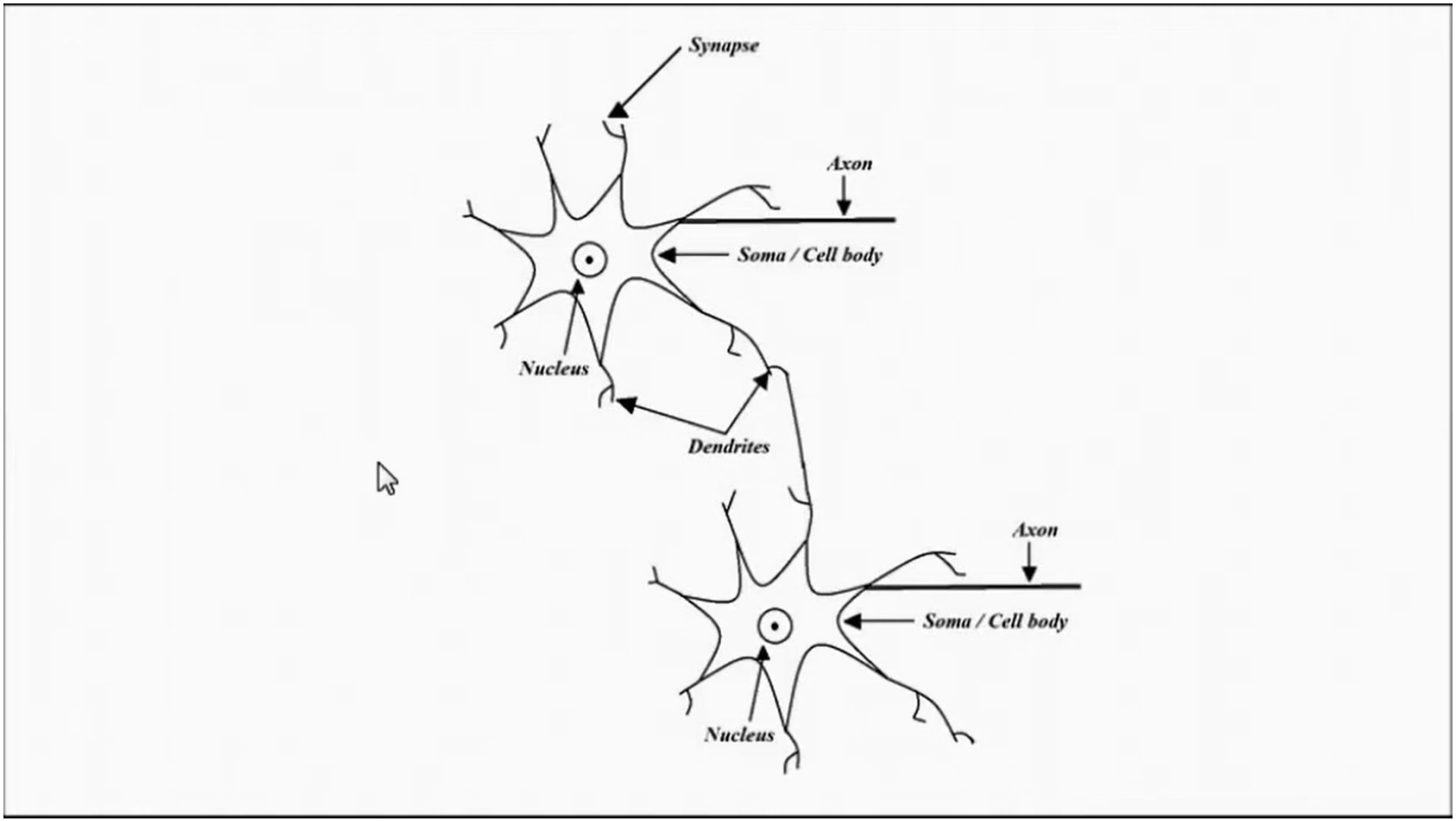
- ***Hebbian learning rule*** is one of the earliest and the simplest learning rules for the neural networks. It was proposed by ***Donald Hebb***.

- Hebb proposed that -

- ***If two interconnected neurons on either side of a synapse are both on or fired or activated at the same time (synchronously), then the synaptic weight between them should be increased.***

- ***If two neurons on either side of a synapse are on or fired or activated at different times (asynchronously), then the weight of that synaptic connection is decreased.***

- Such a synapse is called ***Hebbian Synapse***.


Synapse
Axon
Soma / Cell body
Nucleus
Dendrites
Axon
Soma / Cell body
Nucleus

Mathematical Form of Hebbian Learning

- *Hebbian learning* can be mathematically expressed as,

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

where,

$$x_j = pre - synaptic\ signal$$

$$y_k = post - synaptic\ sigmal$$

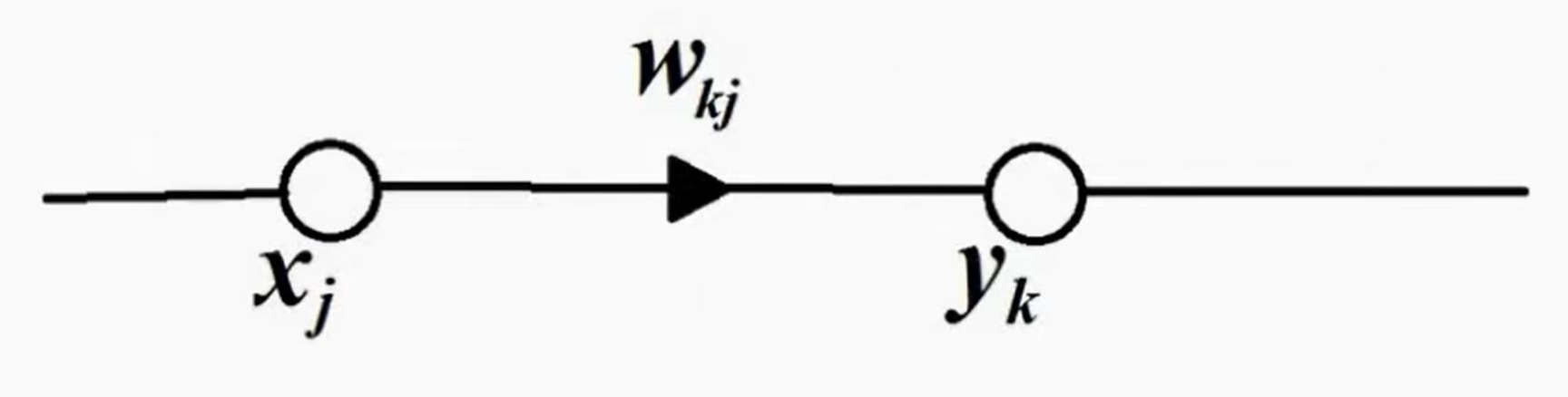
where,

$$\eta = rate\ of\ learning$$
$$(a\ positive\ constant)$$

$$x_j = input\ signal\ of\ j^{th}\ neuron$$

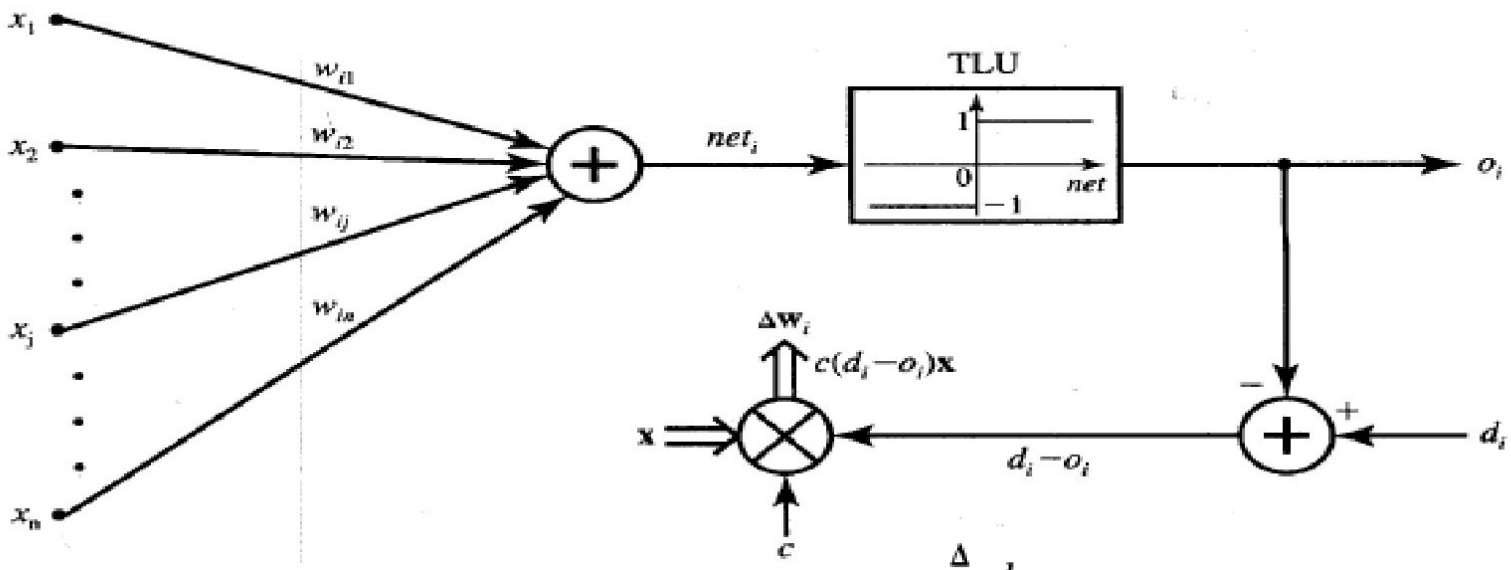
$$e_k = error\ signal$$

$$\Delta w_{kj} = change\ in\ synaptic\ weight$$


w_{kj}
x_j
y_k

Perceptron Learning rule

- Learning signal is the difference between the desired and actual neuron's response
- Learning is supervised



$$r \stackrel{\Delta}{=} d_i - o_i$$

$$o_i = \text{sgn}\,(\mathbf{w}_i^t \mathbf{x})$$

$$\Delta \mathbf{w}_i = c\,[d_i - \text{sgn}\,(\mathbf{w}_i^t \mathbf{x})]\,\mathbf{x}$$

Example

This example illustrates the perceptron learning rule of the network shown in Figure 2.23. The set of input training vectors is as follows:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

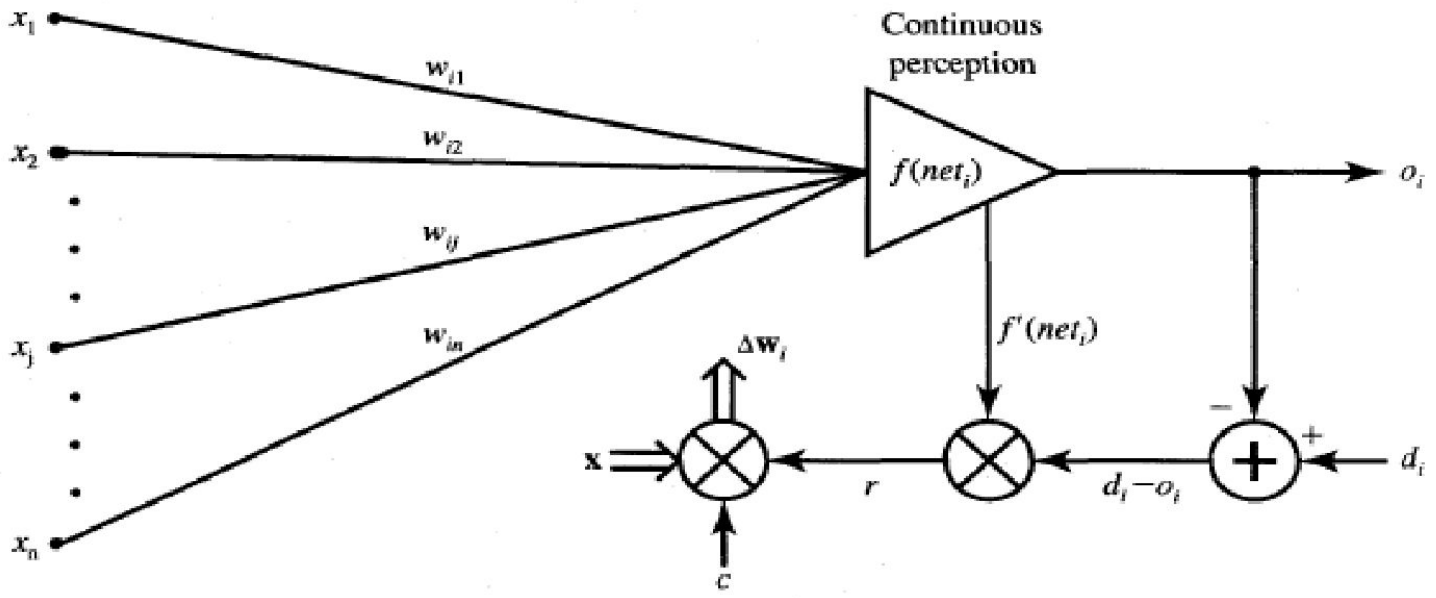
$$\mathbf{w}^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

and the initial weight vector $\mathbf{w}^1$ is assumed identical as in Example [illegible] learning constant is assumed to be $c = 0.1$. The teacher's desired responses for $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ are $d_1 = -1$, $d_2 = -1$, and $d_3 = 1$, respectively. The learning according to the perceptron learning rule progresses as follows.

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

Delta Learning Rule

- Only valid for continuous activation function
- Used in supervised training mode
- Learning signal for this rule is called delta
- The aim of the delta rule is to minimize the error over all training patterns



Delta Learning Rule Contd.

$$r \triangleq [d_i - f(\mathbf{w}_i^t \mathbf{x})] f'(\mathbf{w}_i^t \mathbf{x})$$

Learning rule is derived from the condition of least squared error.

Calculating the gradient vector with respect to **wi**

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

$$\nabla E = -(d_i - o_i) f'(\mathbf{w}_i^t \mathbf{x}) \mathbf{x}$$

The components of the gradient vector are

$$\frac{\partial E}{\partial w_{ij}} = -(d_i - o_i) f'(\mathbf{w}_i^t \mathbf{x}) x_j, \quad \text{for } j = 1, 2, \ldots, n$$

Minimization of error requires the weight changes to be in the negative gradient direction

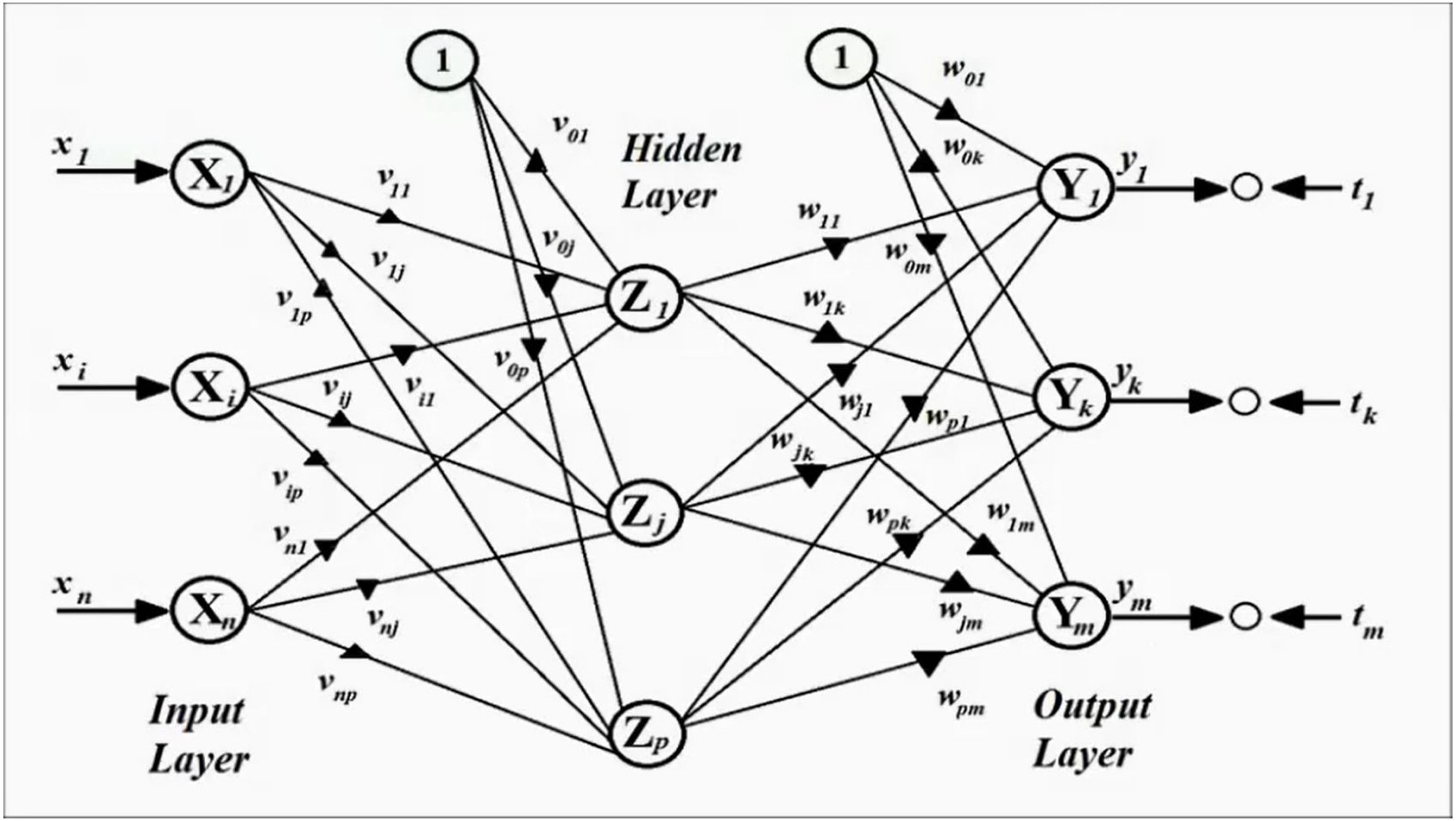
$$\Delta \mathbf{w}_i = -\eta \nabla E \qquad \Delta \mathbf{w}_i = \eta (d_i - o_i) f'(net_i) \mathbf{x}$$

Back - Propagation Network

- The *back-propagation learning algorithm is one of the most important developments in neural network.*

- It is applicable for *multilayer feedforward networks with neurons consisting of differentiable activation functions.*

- The ***main objective*** of ***back-propagation algorithm*** is to ***achieve a balance between memorization and generalization of data by the neural network.***


1
1
v_{01}
w_{01}
x_1
X_1
v_{11}
Hidden Layer
w_{0k}
Y_1
y_1
t_1
v_{1j}
v_{0j}
w_{11}
w_{0m}
v_{1p}
Z_1
w_{1k}
x_i
X_i
v_{0p}
v_{ij}
v_{i1}
w_{j1}
w_{p1}
Y_k
y_k
t_k
w_{jk}
v_{ip}
Z_j
v_{n1}
w_{pk}
w_{1m}
x_n
X_n
v_{nj}
w_{jm}
Y_m
y_m
t_m
v_{np}
w_{pm}
Input Layer
Z_p
Output Layer

- The output of from the network could either be ***binary (0,1)*** or ***bipolar (-1,+1)***.

- The training process involves three steps: ***feedforward of input***, ***back-propagation of error*** and ***weight updation***.

Training Process :
Step 1- Feedforward of Input

- *Net input to each hidden neuron is given by,*

$$z_{inj} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

- *The output of the hidden neuron is given by,*

$$z_j = f(z_{inj})$$

- *Net input to each output neuron is given by,*

$$y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

- The ***final output*** is given by,

$$y_k = f(y_{ink})$$

Step 2- Backpropagation of Error

- Each output neuron receives a ***target value, t.***
- The ***error correction term*** is given by,

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

- *Each hidden neuron sums the delta inputs from output neurons as,*

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

- Again, ***error term*** is calculated as,

$$\delta_j = \delta_{inj} f'(z_{inj})$$

- *The output of the hidden neuron is given by,*

$$z_j = f(z_{inj})$$

- *Net input to each output neuron is given by,*

$$y_{ink} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk}$$

Step 3 – Weight Updation

- The *change in weights and bias for the connections between hidden and output layer* is given by,

$$\Delta w_{jk} = \alpha \delta_k z_j$$

$$\Delta w_{0k} = \alpha \delta_k$$

- The ***change in weights and bias for the connections between input and hidden layer*** is given by,

$$\Delta v_{ij} = \alpha \delta_j x_i$$

$$\Delta v_{0j} = \alpha \delta_j$$

Thank You