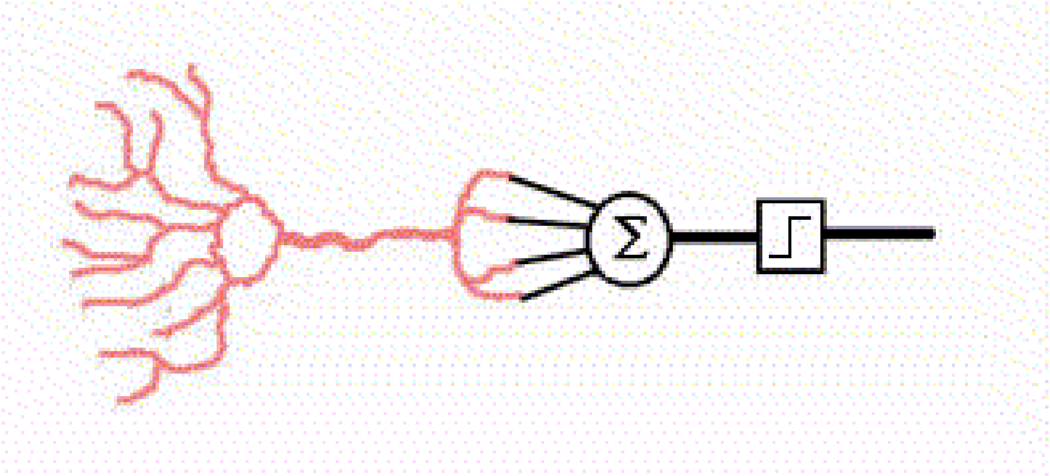


NEURAL NETWORKS (CS010 805G02)

Mod 4 -- Competitive networks

Shiney Thomas
AP,CSE,AJCE

Neural Networks Based on Competition

- Introduction
- Fixed weight competitive nets
  - Maxnet
  - Mexican Hat
  - Hamming Net
- Kohonen Self-Organizing Maps (SOM)
- Counterpropagation
- Adaptive Resonance Theory (ART)

Hamming Net

- A Hamming net [Lippmann, 1987; DARPA, 1988] is a maximum likelihood classifier net that can be used to determine which of several exemplar vectors is most similar to an input vector (an n-tuple).
- The exemplar vectors determine the weights of the net.
- The measure of similarity between the input vector and the stored exemplar vectors is n minus the Hamming distance between the vectors.
- The Hamming distance between two vectors is the number of components in which the vectors differ.

Hamming Net

- For bipolar vectors x and y,

$$\mathbf{x.y} = a - d,$$

- where a is the number of components in which the vectors agree and d is the number of components in which the vectors differ, i.e., the Hamming distance.
- If n is the number of components in the vectors, then, $d = n - a$

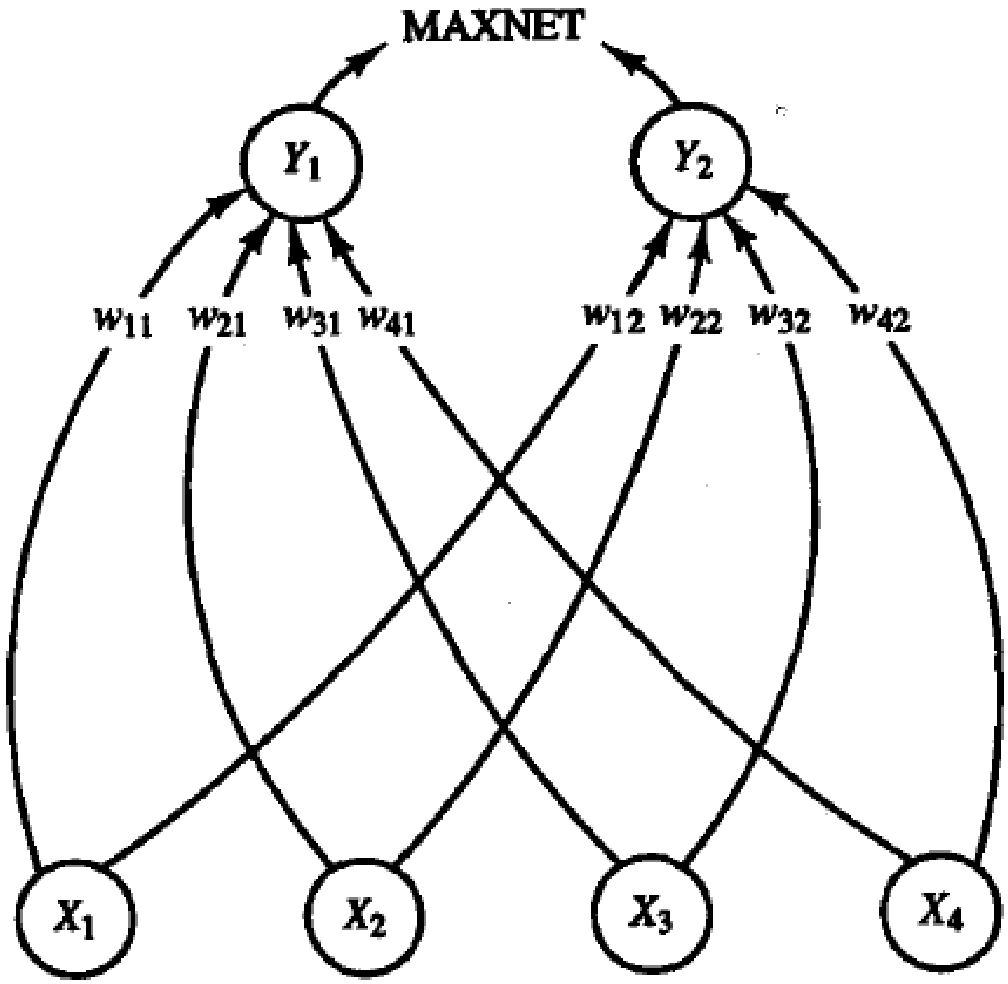
  and

$$\mathbf{x.y} = 2a - n,$$

  or

$$2a = \mathbf{x.y} + n.$$

Hamming Net



The sample architecture shown assumes input vectors are 4- tuples, to be categorized as belonging to one of 2 classes.

HAMMING NET

- By setting the weights to be one-half the exemplar vector and setting the value of the bias to $n/2$, the net will find the unit with the closest exemplar simply by finding the unit with the largest net input.
- The Hamming net uses MAXNET as a subnet to find the unit with the largest net input.
- The lower net consists of n input nodes, each connected to m output nodes (where m is the number of exemplar vectors stored in the net).
- The output nodes of the lower net feed into an upper net (MAXNET) that calculates the best exemplar match to the input vector.
- The input and exemplar vectors are bipolar.

Hamming Net Algorithm

- Given a set of *m* bipolar exemplar vectors

  e(1), e(2), …, e(m),

the Hamming net can be used to find the exemplar that is closest to the bipolar i/p vector **x**.

- The net i/p y_in_j to unit Y_j gives the number of components in which the i/p vector and the exemplar vector for unit $\mathbf{Y_j e(j)}$ agree (n minus the Hamming distance between these vectors).

- Nomenclature:

  n number of input nodes, number of components of any I/p vector

  m number of o/p nodes, number of exemplar vectors

  e(*j*) the j^{th} exemplar vector:

$$\mathbf{e}(j) = \left(e_1(j), \ldots\ldots, e_i(j), \ldots\ldots\ldots, e_n(j)\right)$$

Hamming Net Algorithm

Step 0. To store the m exemplar vectors, initialize the weights:

$$w_{ij} = \frac{e_i(j)}{2}, \; (i = 1, \ldots, n; j = 1, \ldots, m).$$

And initialize the biases:

$$b_j = \frac{n}{2}, \; (j = 1, \ldots, m).$$

Step 1. For each vector $\mathbf{x}$, do Steps 2–4.

Step 2. Compute the net input to each unit Y_j:

$$y\_in_j = b_j + \sum_i x_i w_{ij}, \; (j = 1, \ldots, m).$$

Step 3. Initialize activations for MAXNET:

$$y_j(0) = y\_in_j, \; (j = 1, \ldots, m).$$

Step 4. MAXNET iterates to find the best match exemplar.

Hamming Net Example-

A Hamming net to cluster four vectors

Given the exemplar vectors

$$\mathbf{e}(1) = (1, -1, -1, -1)$$

and

$$\mathbf{e}(2) = (-1, -1, -1, 1),$$

the Hamming net can be used to find the exemplar that is closest to each of the bipolar input patterns, $(1, 1, -1, -1)$, $(1, -1, -1, -1)$, $(-1, -1, -1, 1)$, and $(-1, -1, 1, 1)$.

Step 0. Store the m exemplar vectors in the weights:

$$\mathbf{W} = \begin{bmatrix} .5 & -.5 \\ -.5 & -.5 \\ -.5 & -.5 \\ -.5 & .5 \end{bmatrix}.$$

Initialize the biases:

$$b_1 = b_2 = 2.$$

Hamming Net Example-

A Hamming net to cluster four vectors

Step 1. For the vector $\mathbf{x} = (1, 1, -1, -1)$, do Steps **2–4.**

Step 2.
$$y\_in_1 = b_1 + \sum_i x_i w_{i1}$$
$$= 2 + 1 = 3;$$
$$y\_in_2 = b_2 + \sum_i x_i w_{i2}$$
$$= 2 - 1 = 1.$$

These values represent the Hamming similarity because **(1,1,-1,-1) agrees with e(1)=(1,-1,-1,-1) in the first,** third, and fourth components and because **(1,** 1, -1, **-1)** agrees with **e(2)** = $(-1, -1, -1, 1)$ in only the third component.

Step 3.
$$y_1(0) = 3;$$
$$y_2(0) = 1;$$

Step 4. Since $y_1(0) > y_2(0)$, **MAXNET** will find that unit Y_1 has the best match exemplar for input vector $\mathbf{x} = (1, 1, -1, -1)$.

Hamming Net Example-

A Hamming net to cluster four vectors

Step 1. For the vector $\mathbf{x} = (1, -1, -1, -1)$, do Steps 2–4.

Step 2.

$$y\_in_1 = b_1 + \sum_i x_i w_{i1}$$

$$= 2 + 2 = 4;$$

$$y\_in_2 = b_2 + \sum_i x_i w_{i2}$$

$$= 2 + 0 = 2.$$

Note that the input vector agrees with $\mathbf{e}(1)$ in all four components and agrees with $\mathbf{e}(2)$ in the second and third components.

Step 3.

$$y_1(0) = 4;$$

$$y_2(0) = 2.$$

Step 4. Since $y_1(0) > y_2(0)$, MAXNET will find that unit Y_1 has the best match exemplar for input vector $\mathbf{x} = (1, -1, -1, -1)$.

Hamming Net Example-
A Hamming net to cluster four vectors

Step 1. For the vector $\mathbf{x} = (-1, -1, 1, 1)$, do Steps 2–4.

Step 2.
$$y\_in_1 = b_1 + \sum_i x_i w_{i1}$$
$$= 2 - 1 = 1;$$
$$y\_in_2 = b_2 + \sum_i x_i w_{i2}$$
$$= 2 + 1 = 3.$$

The input vector agrees with **e(1)** in the second component and agrees with e(2) in the first, second, and fourth components.

Step 3.
$$y_1(0) = 1;$$
$$y_2(0) = 3.$$

Step 4. Since $y_2(0) > y_1(0)$, MAXNET will find that unit Y_2 has the best match exemplar for input vector $\mathbf{x} = (-1, -1, 1, 1)$.

Counterpropagation Networks

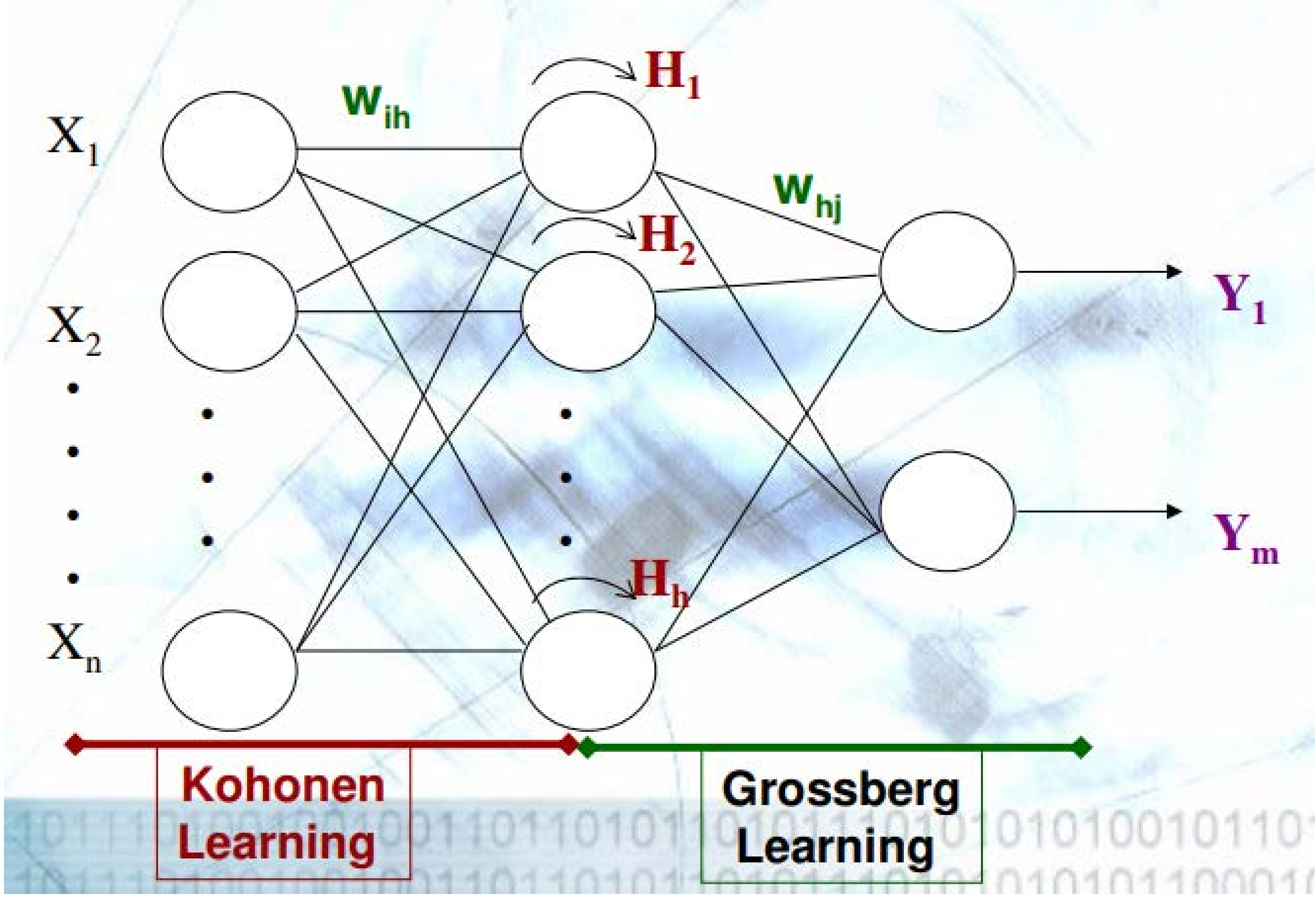
- Robert Hecht-Nielsen
- As compared to backpropagation, reduce training time by a hundred fold
- An example of a hybrid network which combine the features of two or more basic network designs.
- The hidden layer is a SOM Kohonen network with unsupervised learning and the output layer is a Grossberg (outstar) layer fully connected to the hidden layer.
- CP network functions as a look-up table capable of generalization.

Counterpropagation Networks

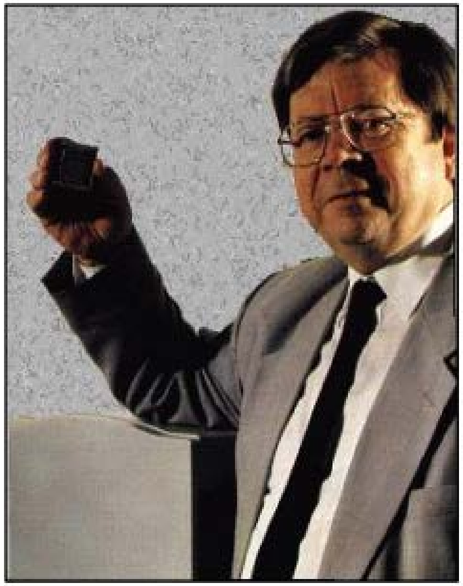
- Training process associates I/p vector to O/p vectors
- The generalization capability of the network allows it to produce a correct O/p even when it is given an input vector that is partially imcomplete or partially incorrect
  - Useful for - pattern- recognition,pattern-completion,signal-enhancement applications

Counterpropagation Networks

- Feedforward CPN



Kohonen's Self Organizing maps



Teuvo Kohonen

- Are a type of neural network.
- They were developed in 1982 by Tuevo Kohonen.
- "Self-Organizing" is because no supervision is required
- SOMs learn on their own through unsupervised competitive learning.
- "Maps" is because they attempt to map their weights to conform to the given input data.
- Primarily used for organization and visualization of complex data.

SOM

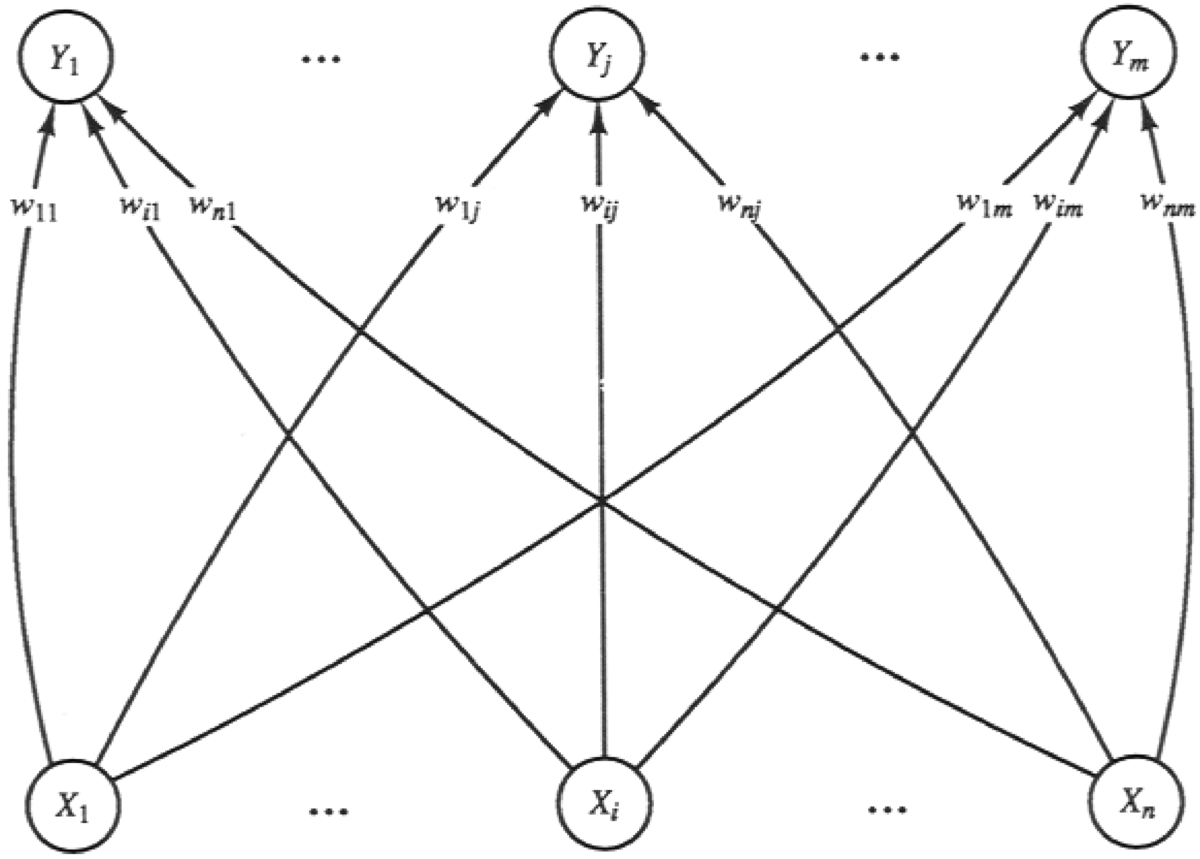
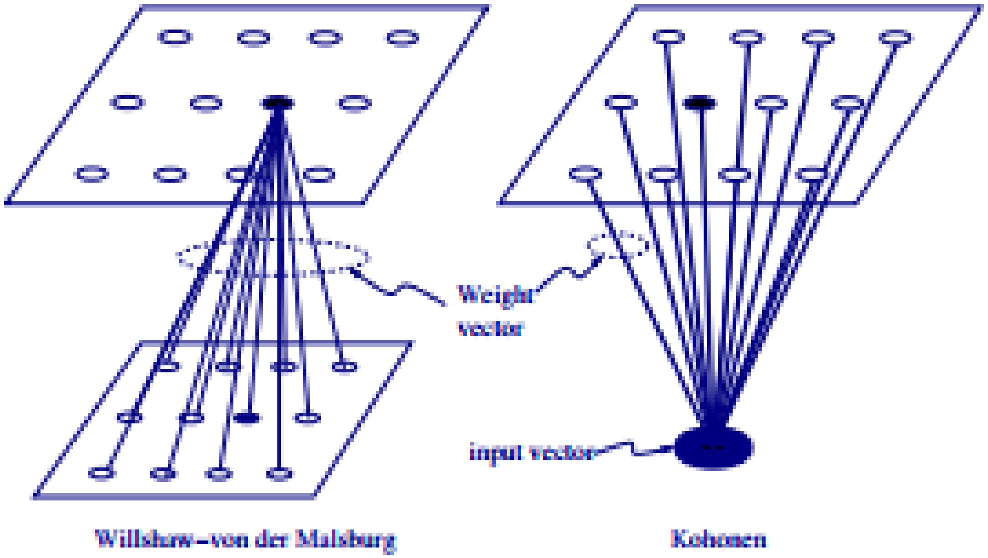


Figure 4.5 Kohonen self-organizing map.

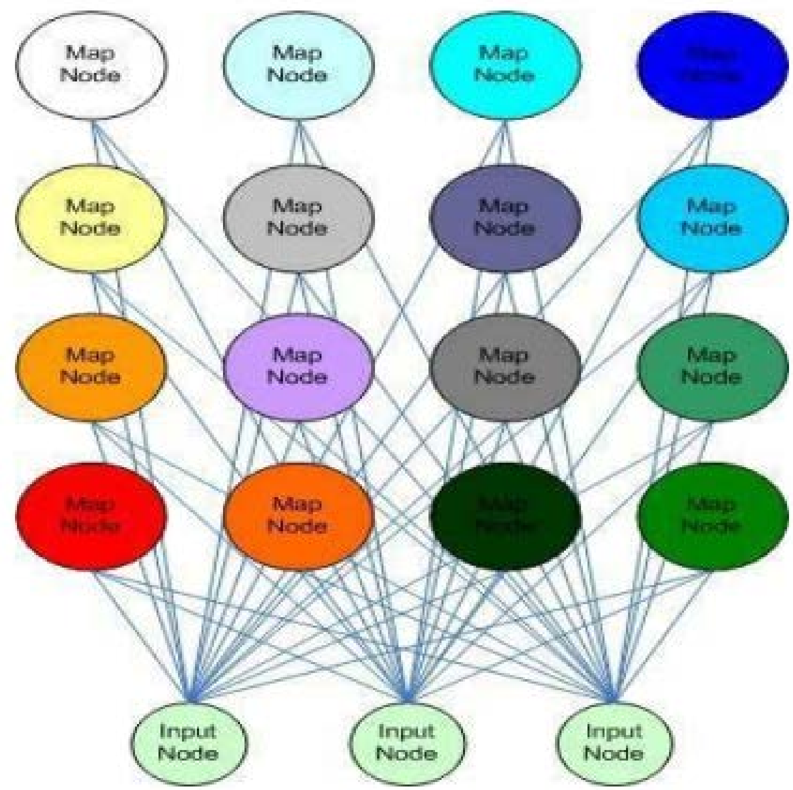
Two Models



- **Willshaw-von der Malsburg model**: input neurons arranged in 2D lattice, output in 2D lattice. Lateral excitation/inhibition (Mexican hat) gives rise to soft competition. Normalized Hebbian learning. Biological motivation.
- **Kohonen model**: input of any dimension, output neurons in 1D, 2D, or 3D lattice. Relaxed winner-takes-all (neighborhood). Competetive learning rule. Computational motivation.

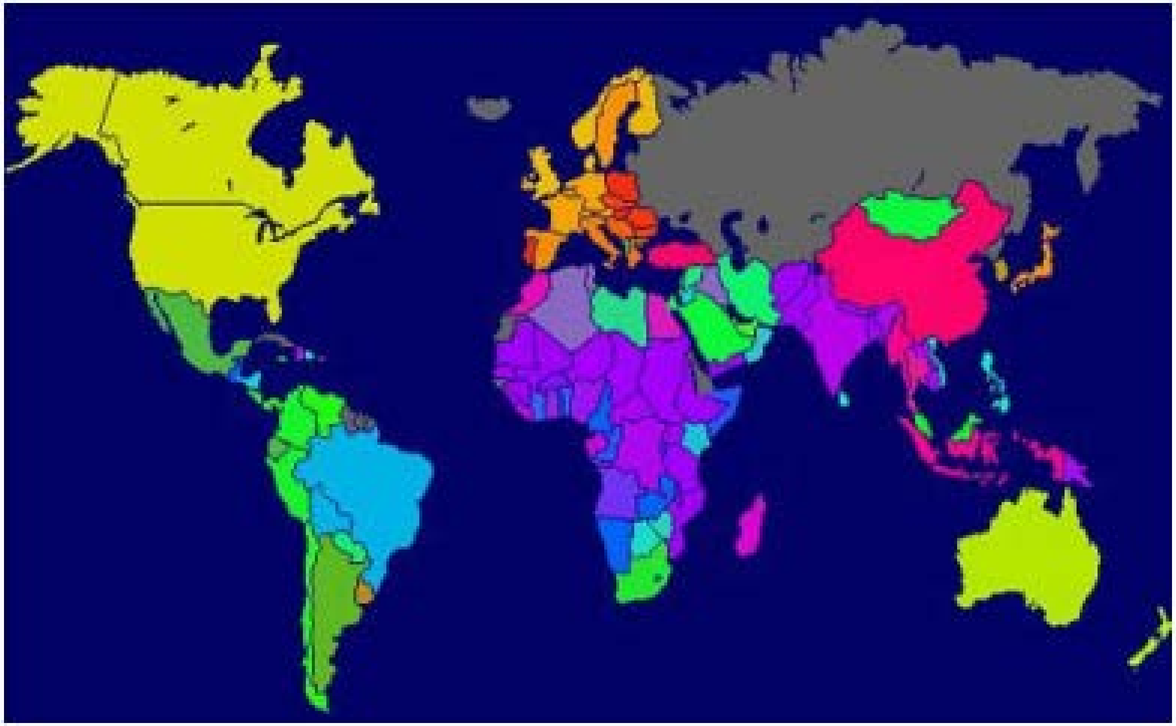
SOM

- In SOM, neurons are placed on a **lattice**, on which a meaningful coordinate system for different features is created (feature map).
- The lattice thus forms a **topographic map** where the spatial location on the lattice is indicative of the input features.
- Eg 4 X 4 SOM nodes



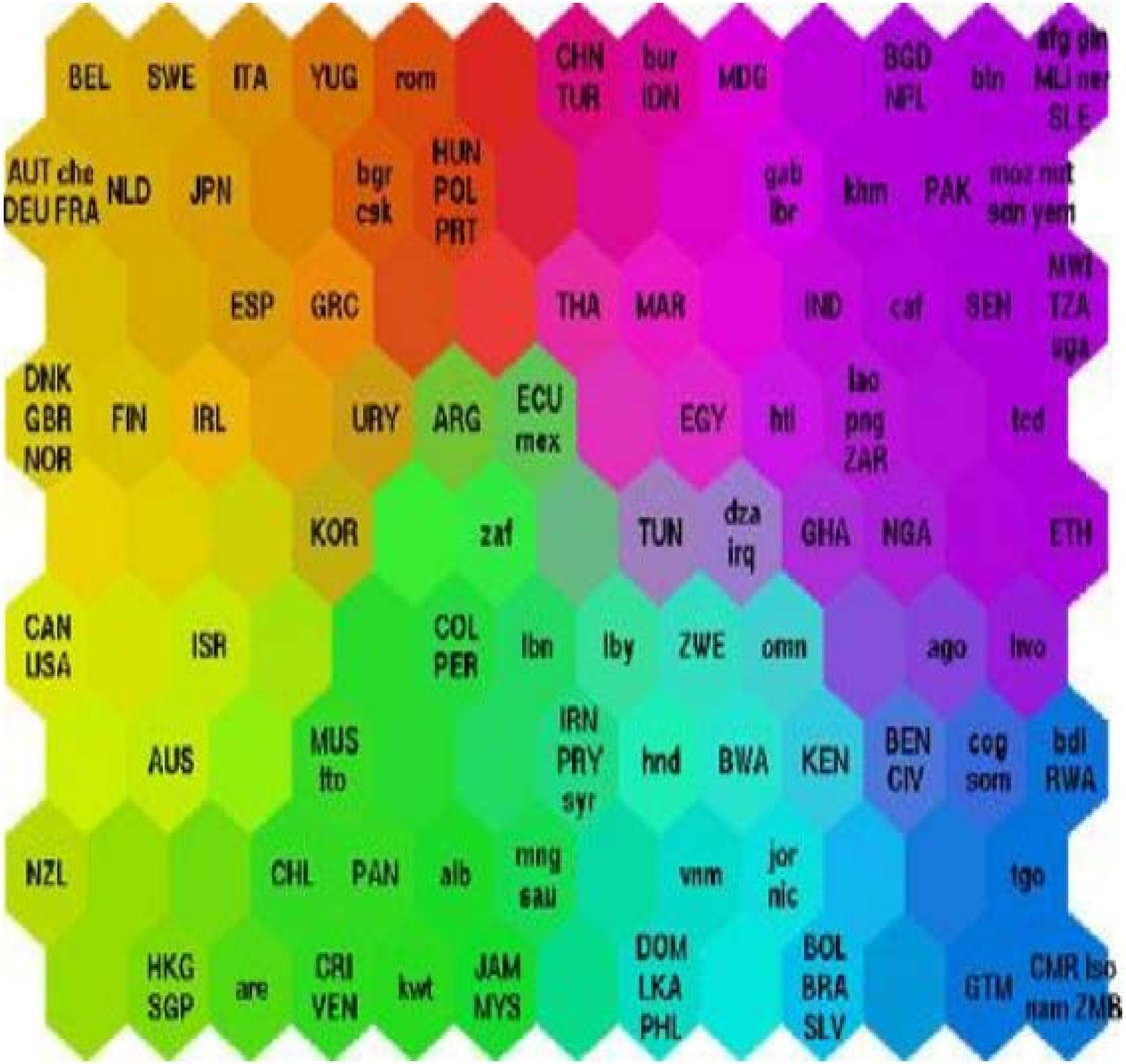
Example

- Yellows and oranges represent wealthy nations, while purples and blues are the poorer nations.



EXAMPLE

- □ After represented by a SOM as shown.
- This is a hexagonal grid. Each hexagon represents a node in the neural network.



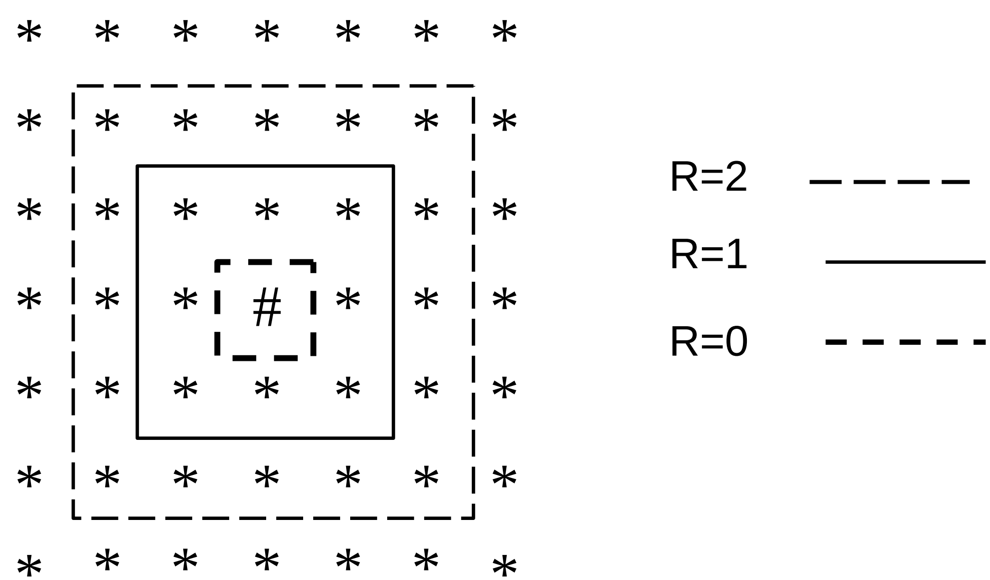
KOHONEN'S SOM

*   *   {*   (*   [#]   *)   *}   *   *

{ } R=2   ( ) R=1   [ ] R=0

Linear array of cluster units

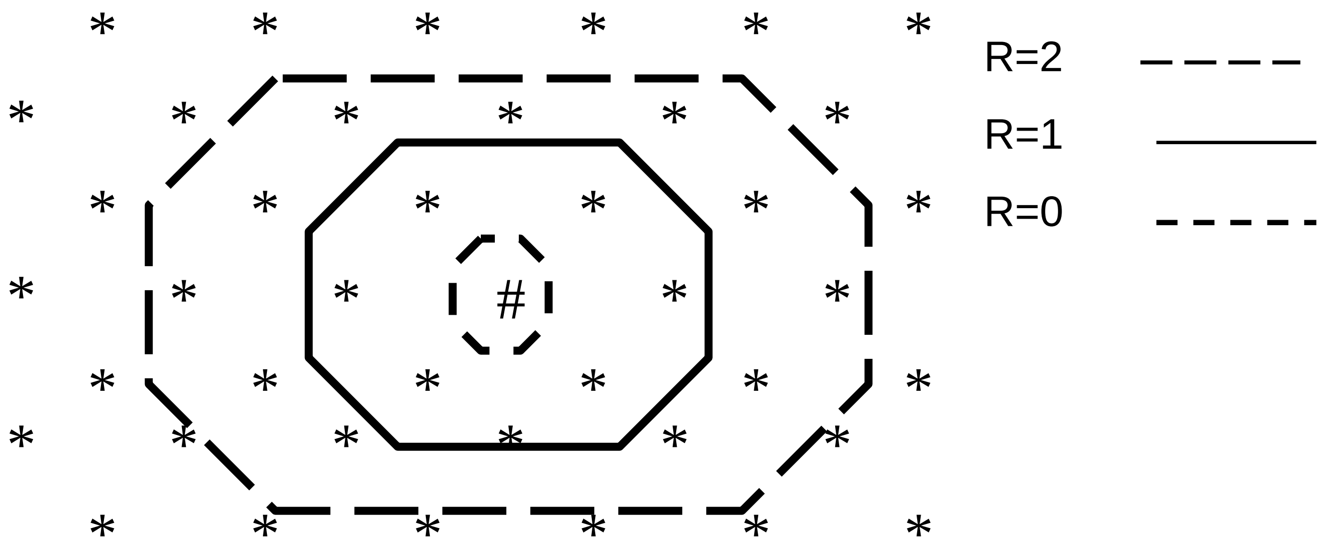
This above figure represents the neighborhood of the unit designated by # of radii R =2,1,0 in a 1D topology (with 10 cluster units).



Neighbourhoods for rectangular grid (Each unit has 8 neighbours)

Kohonen's SOM

Architecture:



Neighbourhoods for hexagonal grid (Each unit has 6 neighbours).

In each of these illustrations, the "winning unit" is indicated by the symbol # and the other units are denoted by *

NOTE: What if the winning unit is on the edge of the grid?

•Winning units that are close to the **edge of the grid** will have some neighbourhoods that have fewer units than those shown in the respective figures

•Neighbourhoods do not "wrap around" from one side of the grid to the other; "missing" units are simply ignored.

Kohonen SOM Training Algorithm

1. Apply an input vector **X**.
2. Calculate the distance D_j (in n dimensional space) between **X** and the weight vectors $\mathbf{W}_j$ of each neuron. In Euclidean space, this is calculated as follows:

$$D_j = \sqrt{\left[\sum_i (x_i - w_{ij})^2\right]}$$

where

x_i = component i of input vector **X**

w_{ij} = the weight from input i to neuron j

KOHONEN SOM TRAINING ALGORITHM

3. The neuron that has the weight vector closest to **X** is declared the winner. This weight vector, called $\mathbf{W}_c$, becomes the center of a group of weight vectors that lie within a distance D from $\mathbf{W}_c$.
4. Train this group of nearby weight vectors according to the formula that follows:

$$\mathbf{W}_j(t+1) = \mathbf{W}_j(t) + \alpha[\mathbf{X} - \mathbf{W}_j(t)] \text{ for all weight vectors within a distance } D \text{ of } \mathbf{W}_c$$

5. Perform steps 1 through 4, cycling through each input vector.

Kohonen SOM Algorithm

- As the network trains, gradually reduce the values of D and α.
- Kohonen recommends α should start near 1 and go down to 0.1, where as D can start out as large as the greatest distance between weight vectors, and end up so small that only one neuron is trained
- After training, classification is performed by applying an arbitrary vector, calculating the excitation produced for each neuron, and then selecting the neuron with the highest excitation as the indicator of correct classification.

APPLICATION

- Natural language processing
  - Document clustering
  - Document retrieval
  - Automatic query
- Image segmentation
- Data mining
- Fuzzy partitioning
- Image color quantization refers to the task of reducing the total amount of distinct colors used to reproduce a digital image
  - Two 24 bit truecolor images chosen as illustrative sample inputs, a picture of Selene and a picture of Lord Vader, Lando Calrissian and Boba Fett as shown in figure.
  - For each picture, the number of colors has been reduced from 16.8M to 16

APPLICATION

- Image color quantization

Figure 21: Selene
16.8M colors

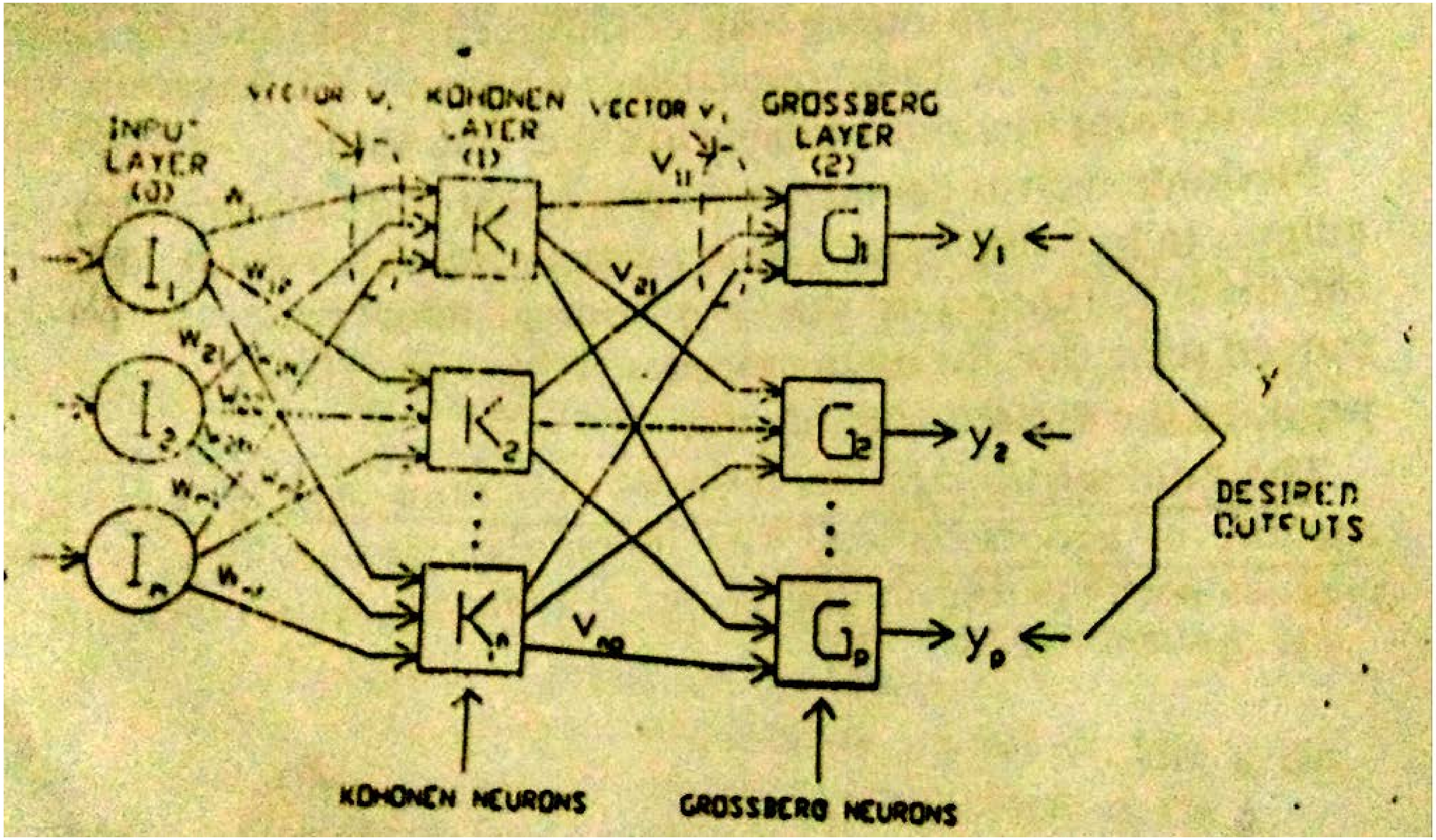
Figure 22: Lord Vader et al.
16.8M colors

Figure 23: Selene
16 colors (SOM)

Figure 24: Lord Vader et al.
16 colors (SOM)

Counterpropagation Network Structure

- Feedforward CPN



Counterpropagation Network Structure

- Feedforward CPN
  - Layer 0- performs no computation
  - Each layer 0 neuron connects to layer 1 (Kohonen layer) neuron with a weight w_{mni} , or weight vector **W**
  - Each neuron in layer 1 (Kohonen) connects to layer 2 (Grossberg layer) by weight v_{npi} , or weight vector **V**
- Two modes of operation
  - Normal mode
  - Training mode

Normal Operation

KOHONEN LAYER

- Functions as "Winner takes all"
  - i.e, for a given input vector, one and only one outputs a logical one; all others output a zero.
  - NET output simply summation of weighted inputs; NET output of Kohonen neuron *j* is

$$NET_j = w_{1j}\, x_1 + w_{2j}\, x_2 + w_{3j}\, x_3 + \ldots\ldots + w_{mj}\, x_m$$

or

$$NET_j = \Sigma_i\, x_i\, w_{ij}$$

or

$$N = XW \quad \text{(vectors)}$$

NORMAL OPERATION

GROSSBERG LAYER

- Functions in a similar manner
- NET output is weighted sum of Kohonen layer outputs $k_1, k_2, \ldots k_n$ forming vector **K**
- Connecting weight vector **V** ,consists of weights $v_{11}, v_{21}, \ldots\ldots v_{np}$
- NET output of Grossberg neuron *j* is

$$NET_j = \Sigma_i \, x_i \, v_{ij}$$

or

$$Y = KV \quad \text{(vectors)}$$

Y= the Grossberg layer output vector

K= the Kohonen layer output vector

V= the Grossberg layer weight matrix

Normal Operation

GROSSBERG LAYER

- Kohonen layer only one element of output vector K is nonzero, so the role of each neuron in the Grossberg layer is to output the value of the weight that connects it to the single non zero Kohonen neuron

Training The Kohonen Layer

- Kohonen layer classifies the input vectors into groups that are similar
  - By adjusting the weights so that similar input vectors activate the same Kohonen neuron

Preprocessing the Input vectors

- Highly desirable to **normalize** all input vectors before applying them to n/w
- Divide each component of an input by that vector's length
  - Length = square root of the sum of squares of all other vector's component
  - $x_i` = x_i / (x_1^2 + x_2^2 + ...+ x_n^2)^{1/2}$

Training The Kohonen Layer

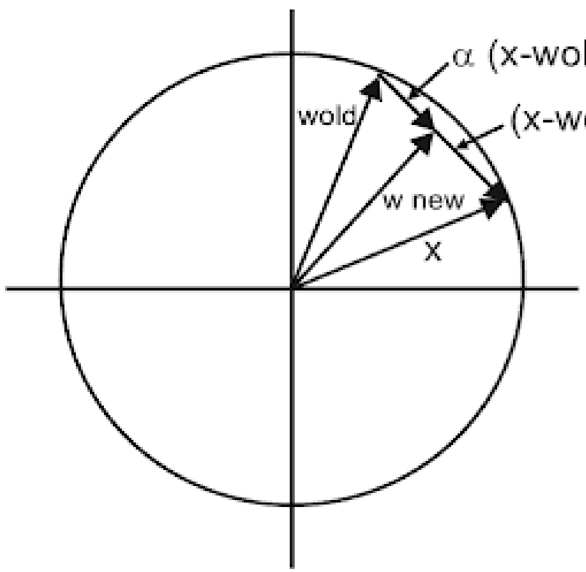
Preprocessing the Input vectors

- To train ,the input vector is applied,its dot product is calculated with the weight vector associated with each Kohonen neuron
- Highest dot product is declared "winner" and weights are adjusted
- Training equation

$$w_{new} = w_{old} + \alpha\ (x - w_{old})$$

α is < than 1,say 0.7

Rotating weight vector by training



Training The Kohonen Layer

<u>Initializing the Weight vectors</u>

- For Kohonen training, randomized weight vectors should be normalized.
- After training, the weight vectors must be equal to normalized input vectors
- Randomizing weight can cause problem
  - Several set of input vectors which are similar,yet must be separated into diff categories
  - Several input vectors are slight variations of same pattern, and should be lumped together
- Solution- Distribute the weight vectors according to density of input vectors that must be separated(place more weight vector in vicinity of large number input vector)

Training The Grossberg Layer

- Relatively simple to learn; Supervised training
- Input vector → Kohonen output → Grossberg output are calculated
- Weight is adjusted only if it connects to a Kohonen neuron having a nonzero output
- Weight adjustment is proportional to the difference between the weight and the desired output of the Grossberg neuron to which it connects

$$v_{ij\ new} = v_{ij\ old} + \beta\,(y_j - v_{ij\ old})k_i$$

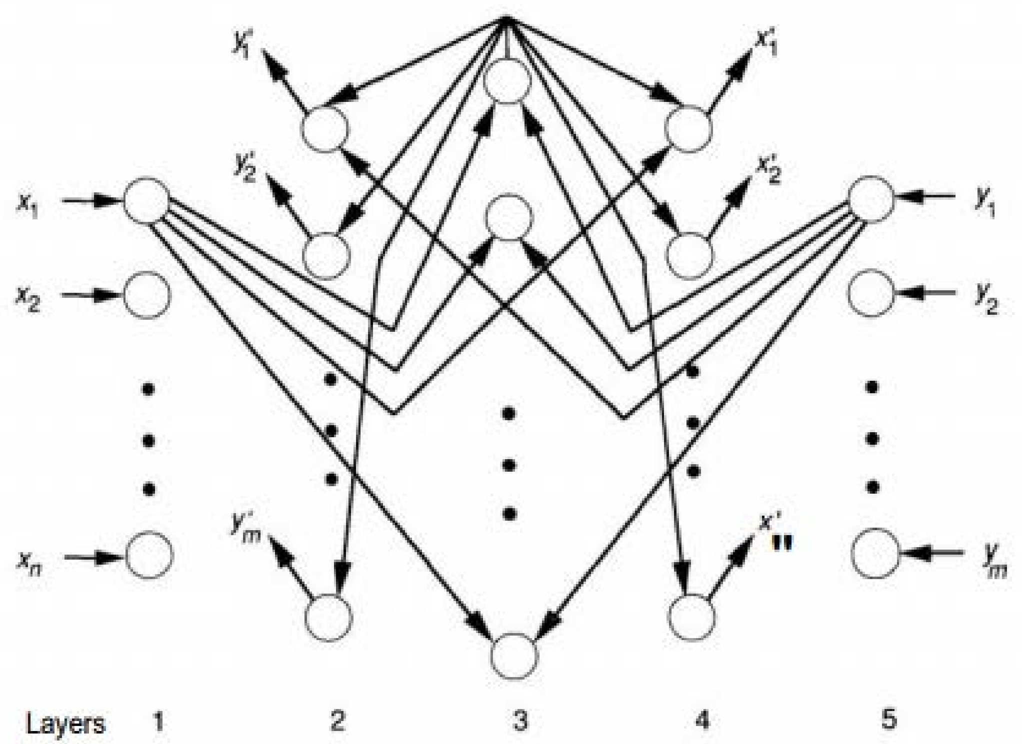
  - k_i = o/p of Kohonen neuron *i*
  - y_j = component *j* of the vector of desired outputs
  - Initially β =0.1,gradually reduced during training

Training The Grossberg Layer

- Weights of Grossberg layer converges to the average values of the desired outputs
  - Where as weights of Kohonen layer converges to the average values of the inputs

Full Counterpropagation Network

- This spiderlike diagram of the CPN architecture has five layers: two input layers (1 and 5), one hidden layer (3), and two output layers (2 and 4).
- The CPN gets its name from the fact that the input vectors on layers 1 and 2 appear to propagate through the network in opposite directions



FULL COUNTERPROPAGATION NETWORK

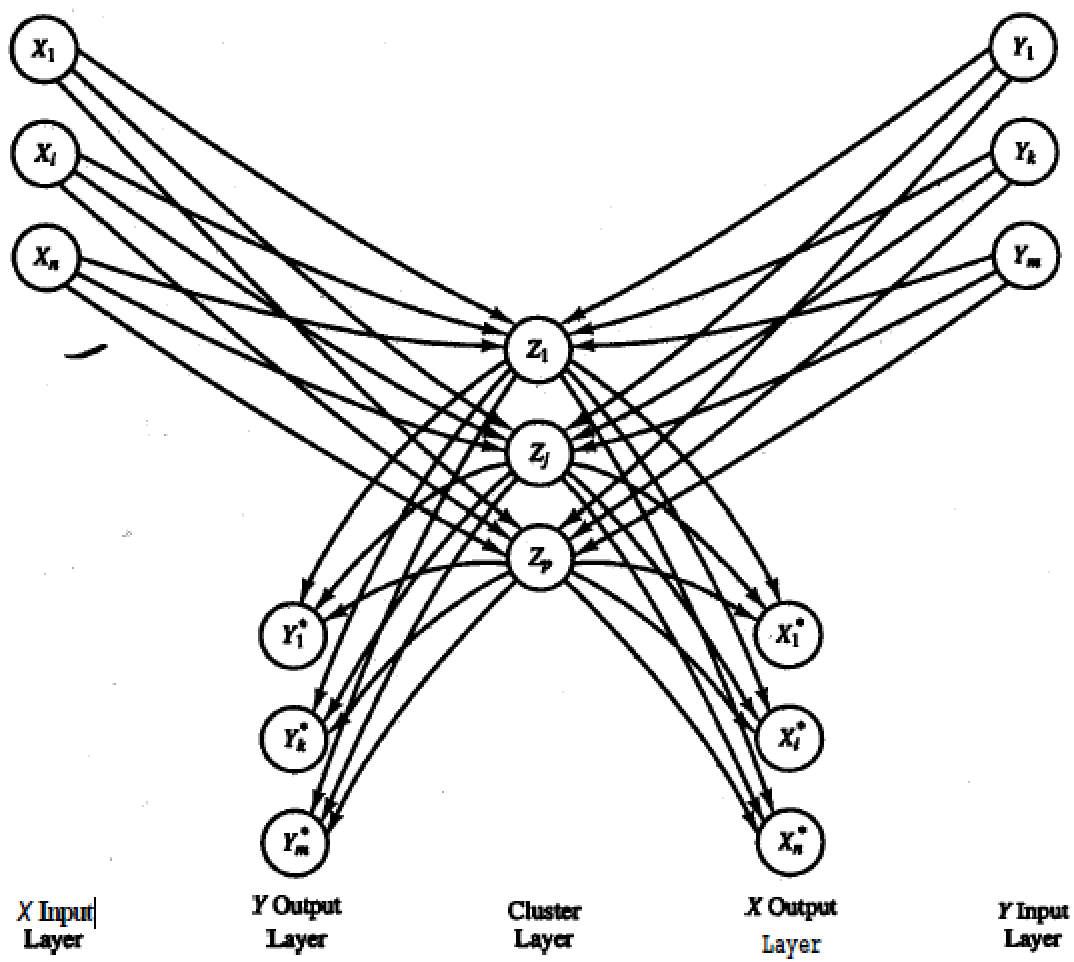
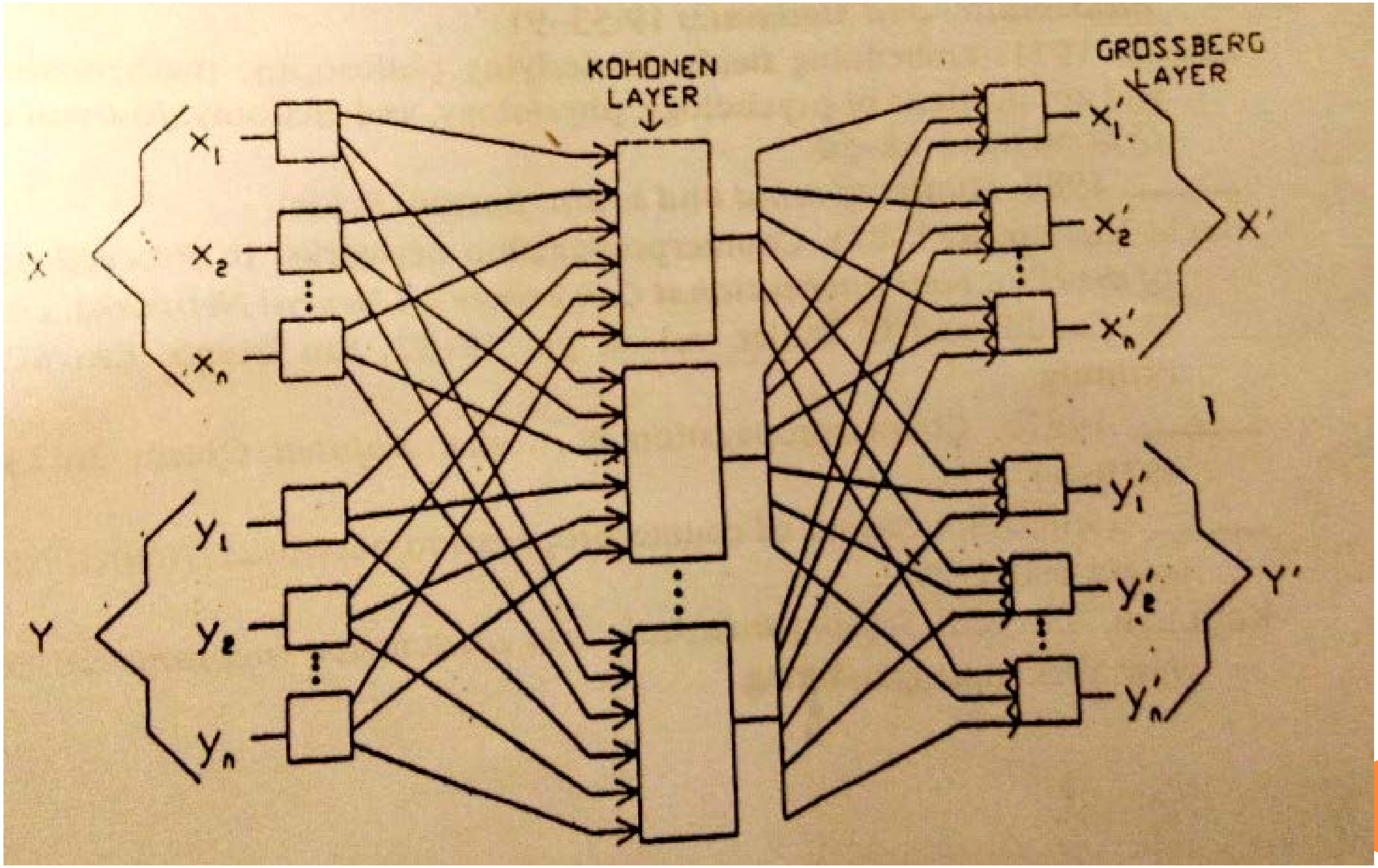


Figure 4.41 Architecture of full counterpropagation network.

FULL COUNTERPROPAGATION NETWORK



Full Counterpropagation Network

- Normal mode:
  - Input vectors X and Y are applied and the trained network produces output vectors X' and Y',which are approximations of X and Y, resp.
  - X and Y are normalized unit vectors, hence output also normalized.
- Training mode:
  - X and Y are applied both as inputs to network and as desired output
  - X is used to train X' output ,Y is used to train Y' output of the grossberg layer
  - Training process is same as feedforward CPN

CPN APPLICATION

- Data Compression (Refer from text-Philip D Wasserman)

Adaptive Resonance Theory

Adaptive Resonance Theory

- Human Memorization
- New memories are stored such that existing ones are not forgotten and modified
- Creates a Dilemma
  - How can brain remain plastic, able to record new memories as they arrive and yet retain the stability needed to ensure that existing memories are not erased or corrupted in the process
- **Stability – Plasticity Dilemma**
- More generally, the Stability – Plasticity Dilemma asks:
  - How can a system retain its previously learned knowledge while incorporating new information?

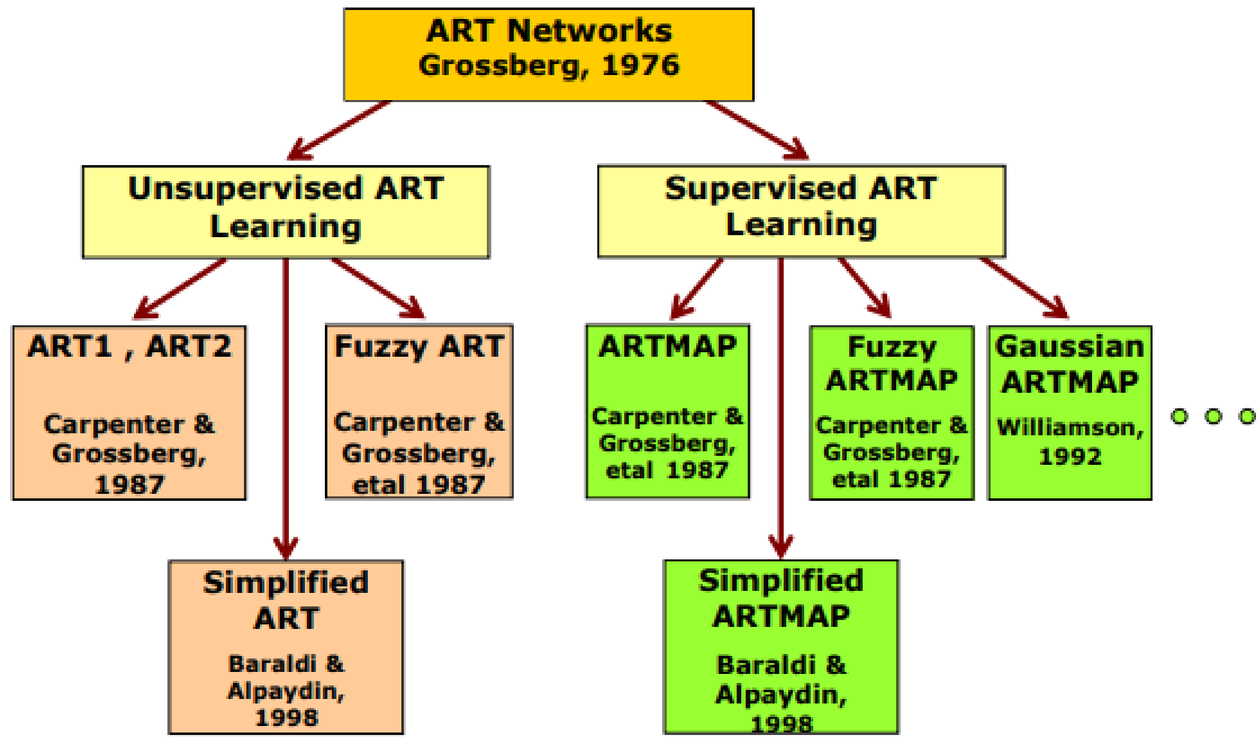
Adaptive Resonance Theory

- **Stability – Plasticity Dilemma**
- A system must be able to learn to adapt to changing environment (must be plastic) but constant change can make a system unstable, because the system may learn new information only by forgetting every thing it has learned so far.

Adaptive Resonance Theory

- Gail Carpenter and Stephen Grossberg (1987,Boston University) developed the Adaptive Resonance learning model to answer this question.
- Essentially, ART (Adaptive Resonance Theory) models incorporate new data by checking for similarity between this new data and data already learned; "memory".
  - If there is a close enough match, the new data is learned.
  - Otherwise, this new data is stored as a "new memory".
- Unsupervised self organizing network

ART Types

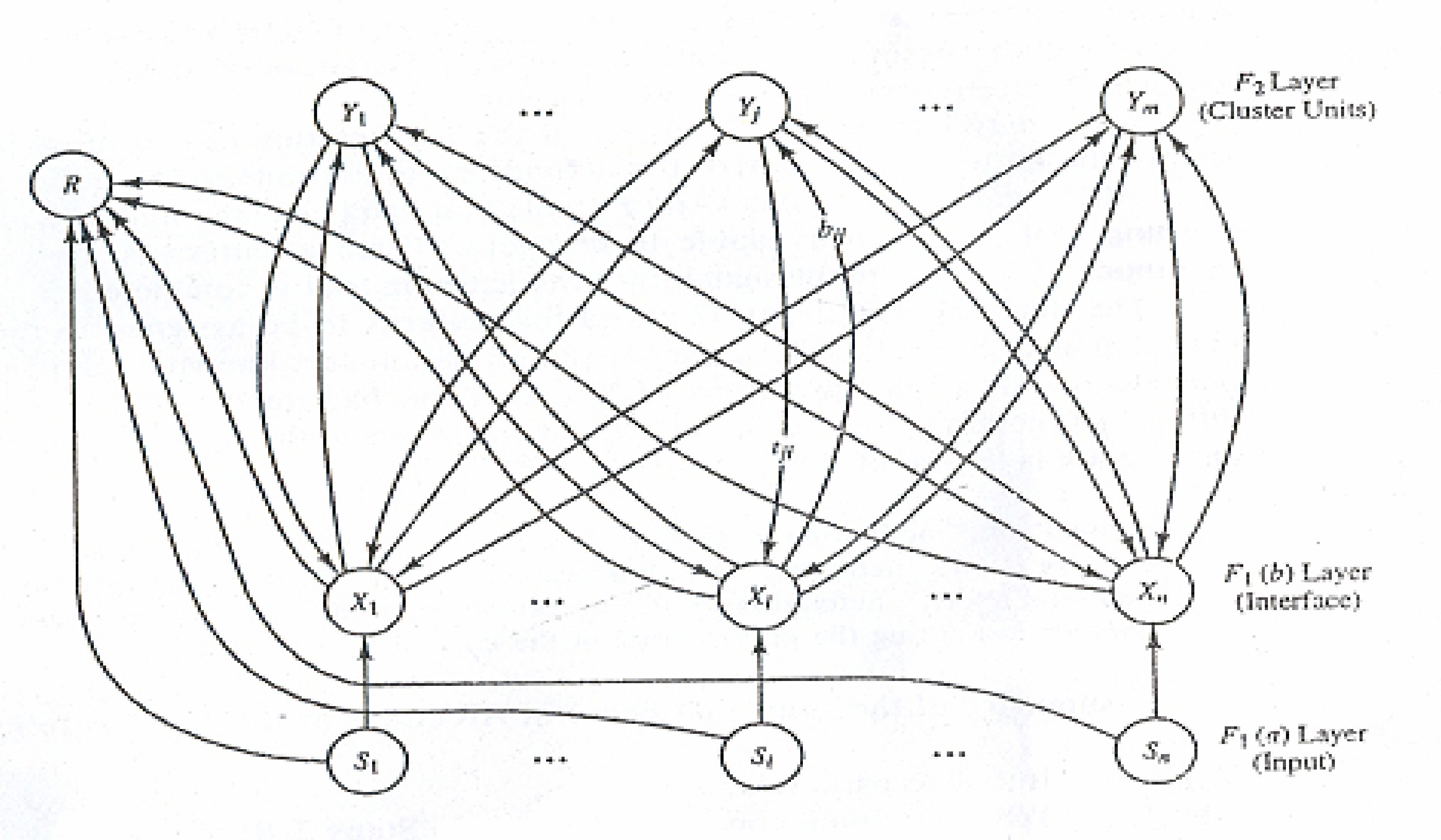


- Introduced- Gr
- Unsupervised A
- Supervised AR
  1991
- **ART1**: Unsupe
- **ART2**: Unsupe
  valued input v
- ART3: Incorporates "chemical transmitters" to control the search process in a hierarchical ART structure.
- **ARTMAP**: Supervised version of ART that can learn arbitrary mappings of binary patterns.
- Fuzzy ART: Synthesis of ART and fuzzy logic.
- Fuzzy ARTMAP: Supervised fuzzy ART
- dART and dARTMAP: Distributed code representations in the F2 layer (extension of winner take all approach).
- Gaussian ARTMAP

ART

- ART network is a vector classifier
- It accepts an input vector and classifies it into one of a no. of categories depending upon which of a no. of stored patterns it most resembles.
- If the input vector does not match any stored pattern, a new category is created by storing a pattern that is like the input vector.
- Once a stored pattern is found that matches the input vector within a specified tolerance(the vigilance),that pattern is adjusted to make it more like the input vector
  - No stored pattern is ever modified if it does not match the input pattern within the vigilance tolerance

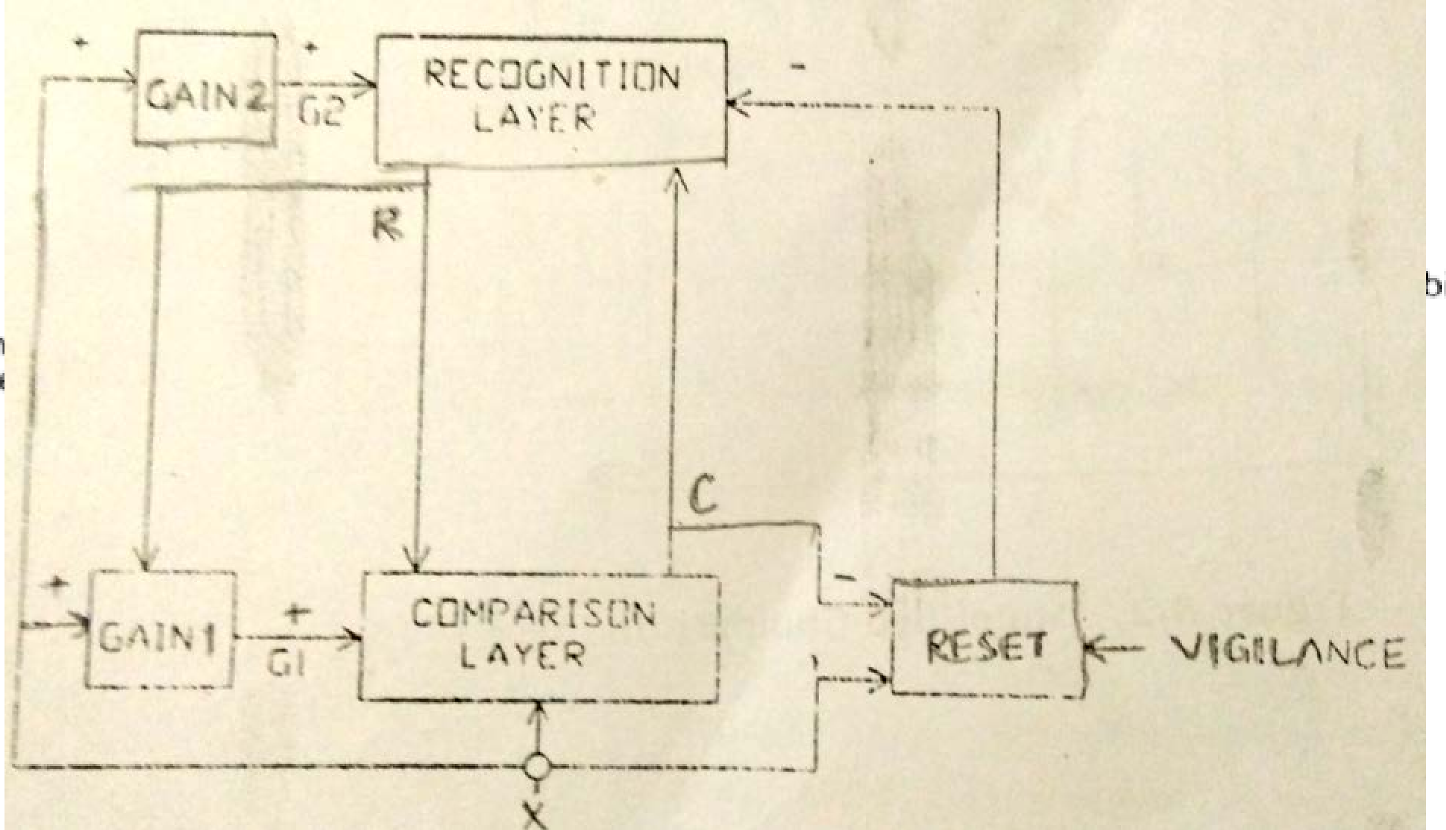
ART1 Architecture



From Text Laurene Fausett

ART1 ARCHITECTURE

- Two Layers - Comparison, Recognition
- Control functions for training and classification- Gain 1,Gain 2, Reset

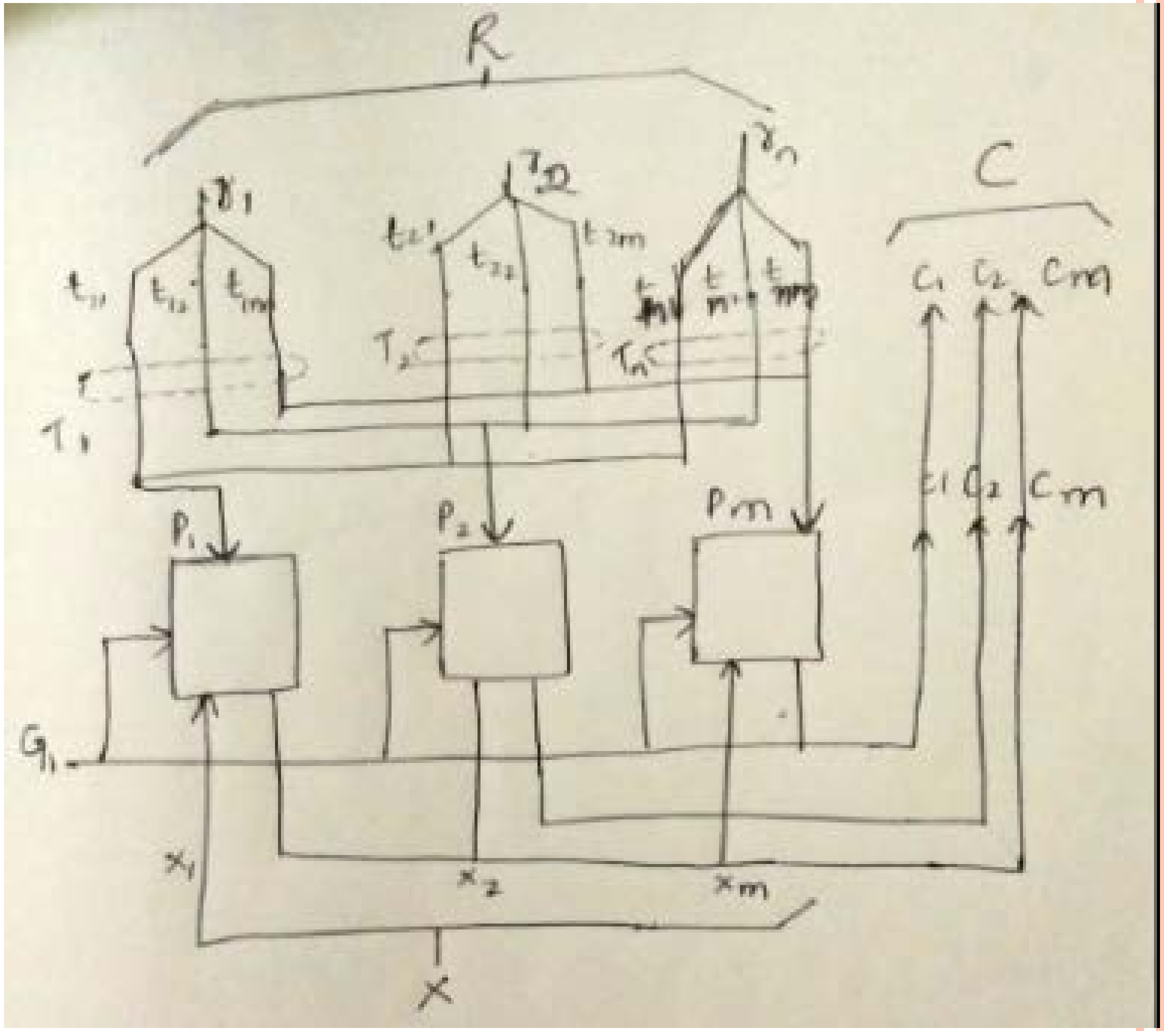


In
ve

bition

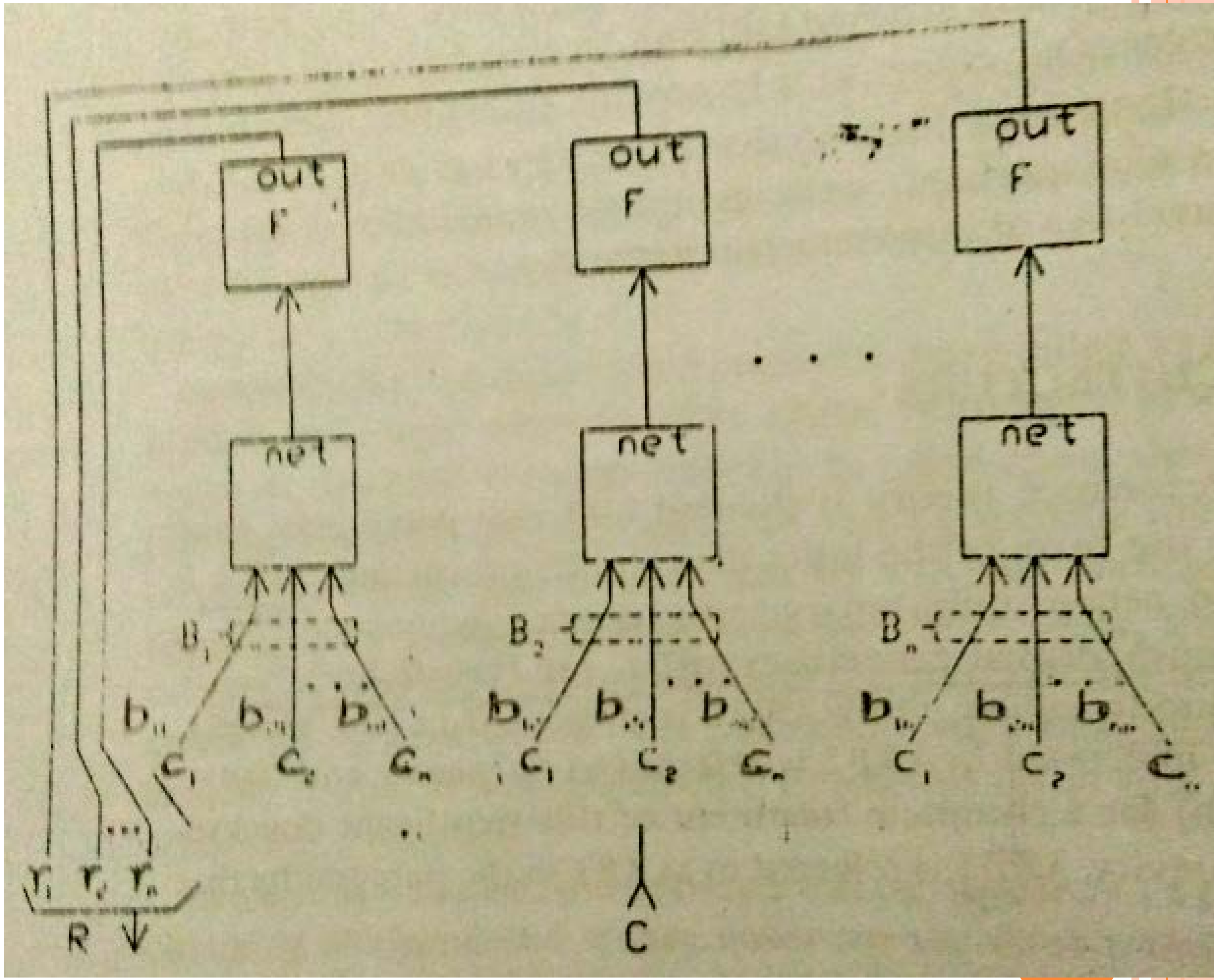
COMPARISON LAYER

- Input vector X
- Unchanged vector C
- 3 inputs to comp layer
  - x_i from X
  - Feedback P_j
  - I/p from Gain 1
- To o/p “1”,at least two neurons must be “1”;otherwise 0
- **TWO-THIRDS rule**



Recognition layer

- Classify I/p vector
- Assoc. weight vector B_j
- Only the neuron with a weight vector best matching with the input vector "fires", all other are inhibited



Recognition layer

- Each neuron computes a dot product between its weight and incoming vector C
- The neuron that has largest output(weight most like vector C) will win the competition while inhibiting all other neurons in the layer

GAIN 2

- G2 ,the output of Gain 2, is one if Input vector X has any component that is one
- Logical "or" of components of X

GAIN 1

- Like G2,o/p of G1 is one if any component of binary i/p vector X is one
- However if any component of R is one,G1 is forced to Zero.

"or" of X comp.	"or of R comp.	G 1
0	0	0
1	0	1
1	1	0
0	1	0

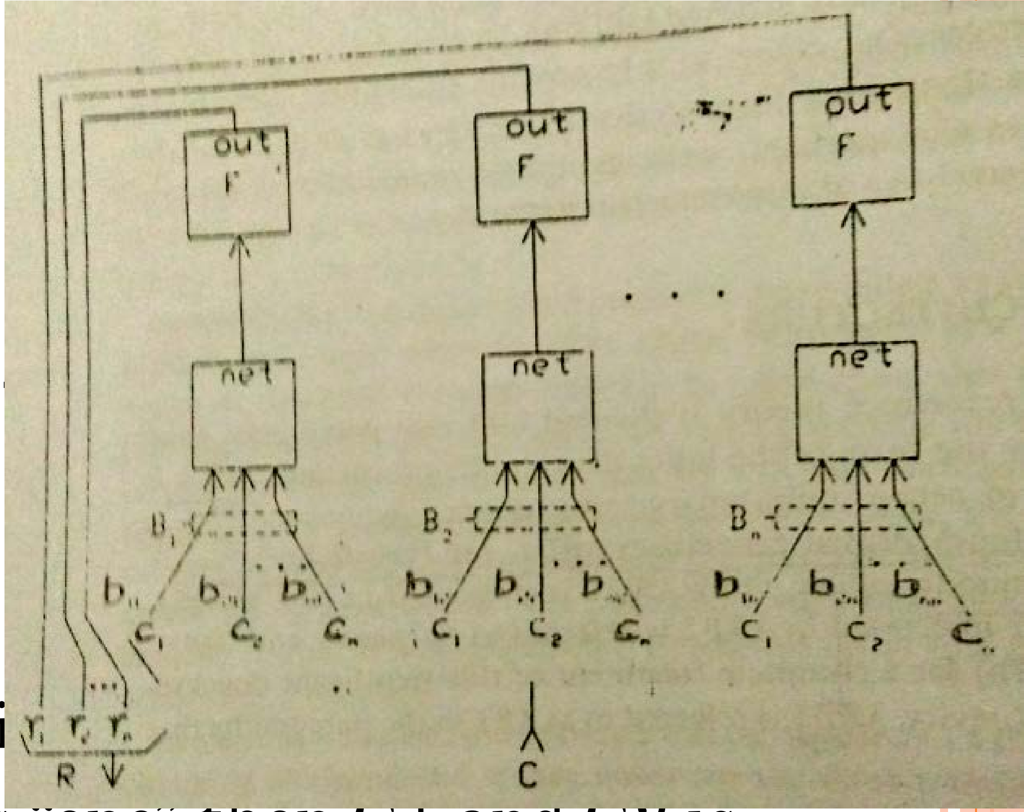
RESET

- Reset module measures the similarity between vector X and C
- If they differ by more than vigilance parameter, a reset signal is sent to disable the firing neuron in Recognition layer
- It calculates similarity as the ratio of no. of ones between vectors X and C
- If this ratio is below the vigilance parameter ρ, the reset signal is issued

ART Classification Operation

- Three phases
  - Recognition Phase
  - Comparison Phase
  - Search Phase

ART Classification
Recognition Phase

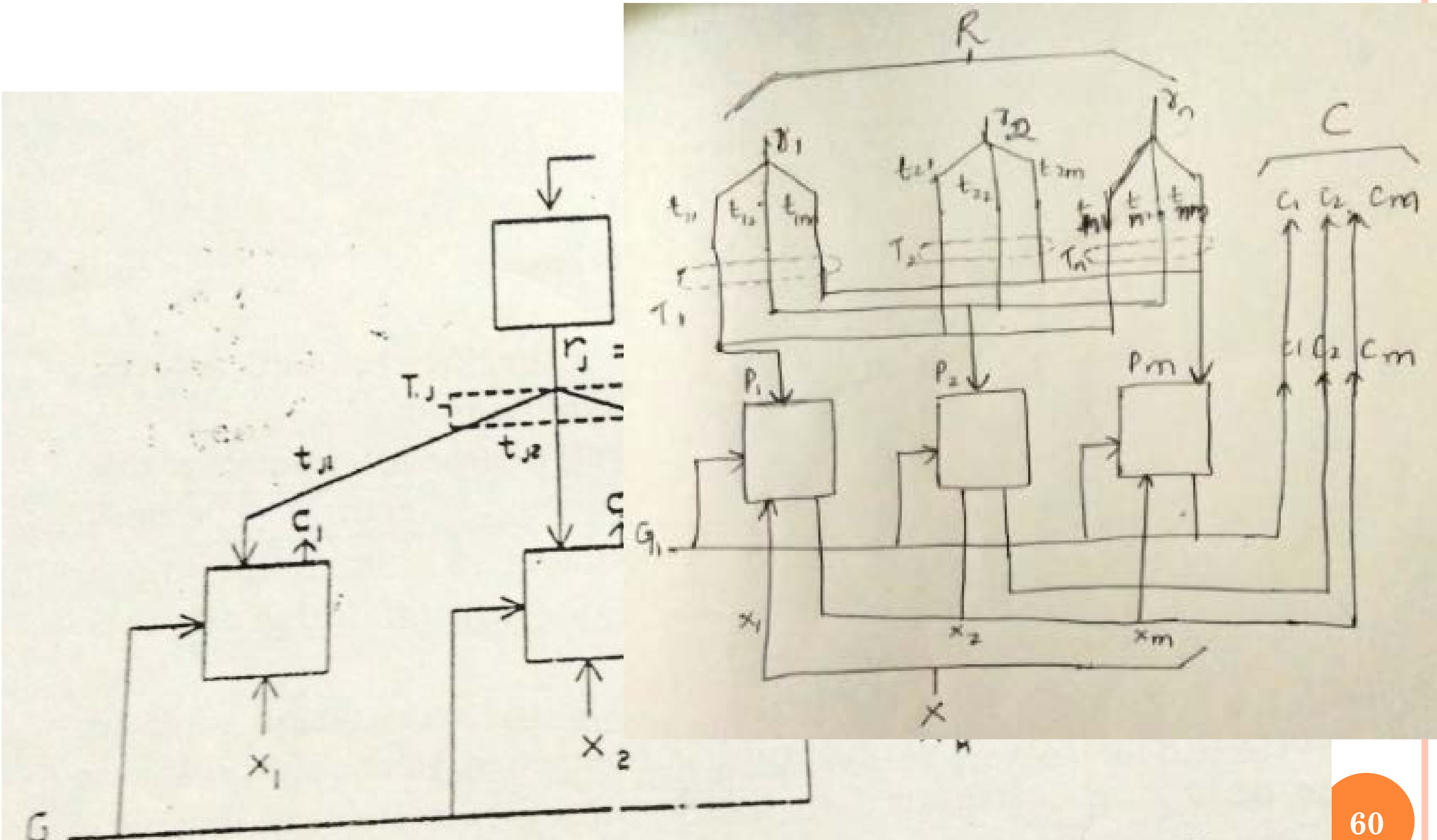


- Initially,no input vector
  X are zero
- So G2 is set to zero→ all
  p their o/p also zero
- Now vector X to be classi
- One or more components "one" then G1 and G2 is set to one
- By two-thirds rule ,if X input vector is one, a neuron will fire; thus vector C = vector X
- For each neuron in RL, a dot product is formed between associated weight B_j and vector C

ART Classification Operation-Recognition Phase

- Neuron with largest dot product has weights that best match the input vector
- It wins the competition and fire
  - Component r_j of vector R = 1,rest all 0

ART Classification Operation- Comparison Phase



ART Classification Operation-Search Phase

- No reset signal, then match is adequate, classification done
- Otherwise, stored pattern must be searched for better match
- Inhibition of firing neuron in RL causes components of R to zero → G1 goes to one → Input pattern X appears at C
- A different neuron wins the RL and a different stored pattern P is fed back to CL
- Again see if P and X matches

ART Classification Operation- Search Phase

- This process repeats until:
  1. A stored pattern is found that matches X above the vigilance parameter, that is $S > \rho$. Network enters **a training cycle** that modifies the weight of T_j and B_j
  2. All stored patterns have been tried, found to mismatch the input vector, and all recognition layer neurons are inhibited. Then a previously **unallocated** neuron in the RL is assigned to this pattern and its weight vector T_j and B_j are set to match the pattern

ART Implementation

- ART operation
  - Initialization
  - Recognition
  - Comparison
  - Search
  - Training

Initialization

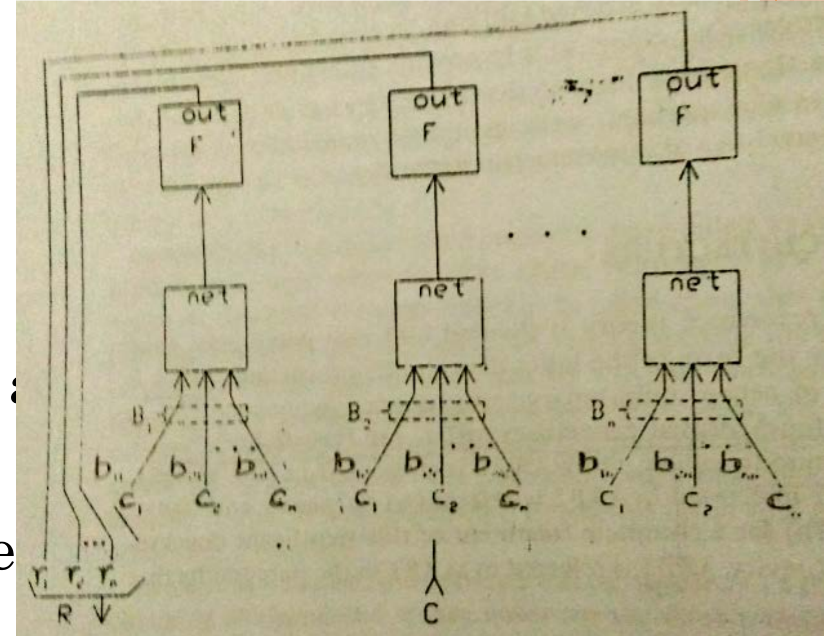
- Set T_j and B_j and ρ to initial values
- Weight of bottom up vectors B_j are all initialized to low values

$$\mathbf{b_{ij} < L / (L\text{-}1+m) \text{ for all } i,j}$$

m= no. of components of input vector

L= a constant >1 (typically, L=2)

ART IMPLEMENTATION



- Weight of top down vectors T_j are a

$$\mathbf{t_{ij} = 1 \quad \text{for all } i,j}$$

- Vigilance parameter ρ set in range

<u>Recognition</u>

- Recognition performed as Dot product for each neuron

$$NET_j = (B_j \cdot C)$$

where B_j = the wght vector associated with RL layer neuron

C = the o/p vector of CL neuron

NET_j = excitation of neuron j in RL

- F is the threshold fn

$$OUT_j = 1 \text{ , if } NET_j > T \qquad (T=\text{threshold})$$
$$0 \text{ , otherwise}$$

ART Implementation

Comparison

- Compare C and X, producing a **reset output** whenever their **similarity S is below the vigilance value**
- Compute similarity as S = N/D

  where D is the no of 1s in X vector

  and N is the no of 1s in C vector

Search

Training (*Refer text-Philip D Wasserman)

ART Example