Matching and Self-Organizing Networks

Berlin Chen, 2002

Introduction

- Networks to be Discussed
  - Network Learning Mode
    - **Competitive learning**, instead of the correlation rule or gradient descent techniques, is used
  - No feedback or No Teacher
    - Discover for patterns, features, regularities, or categories of input pattern **without a teacher**
    - **Self-organization**: extract statistical or frequency properties of (**redundant**) input patterns

Introduction

- Measure of Similarity for Learning
  - The **scalar product** of the network weights (class prototype) and the input pattern vector
  - The topological neighborhood or distance between the responding neurons arranged in regular geometrical array

- Network architectures covered here
  - Hamming network, MAXNET, Kohonen layer, Grossberg outstar learning layer, a counter-propagation network, self-organizing feature mapping networks, and adaptive resonance networks

Hamming Network and MAXNET

- Hamming network
  - Match the input vector with the stored vectors
  - Implement the optimum minimum bit error classification for binary bipolar pattern inputs
  - The class prototype vector is encoded into respective weights of the neuron being the class indicator for the specific prototype

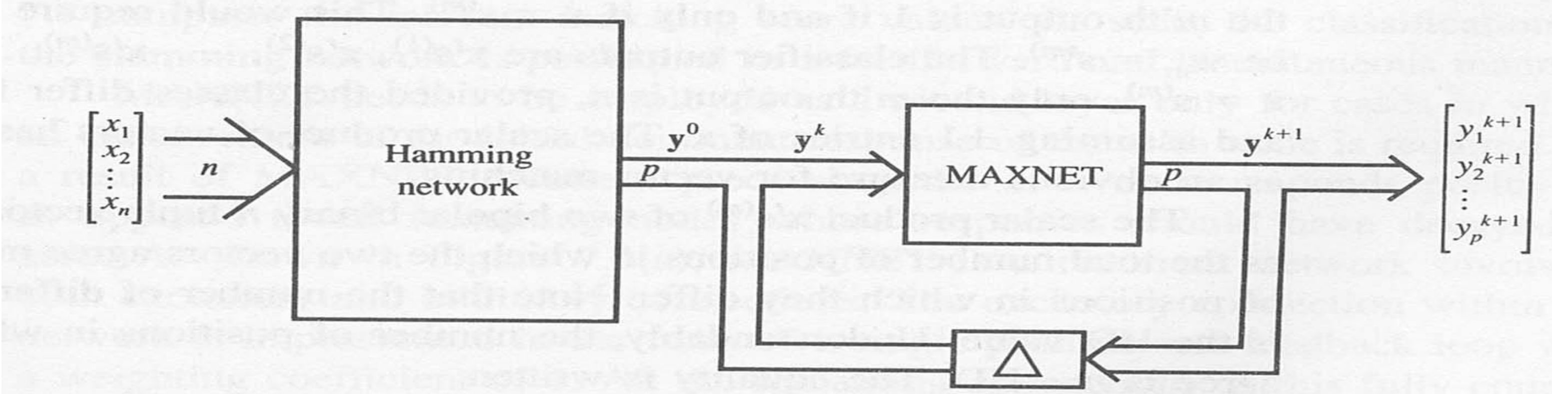


Figure 7.1 Block diagram of the minimum HD classifier.

Hamming Network and MAXNET

- MAXNET
  - A recurrent network involving both excitatory and inhibitory connections
    - Positive self-feedbacks and negative cross-feedbacks
  - After a number of recurrences, the only unsupervised node will be the one with the largest initializing entry from the Hamming network output vector

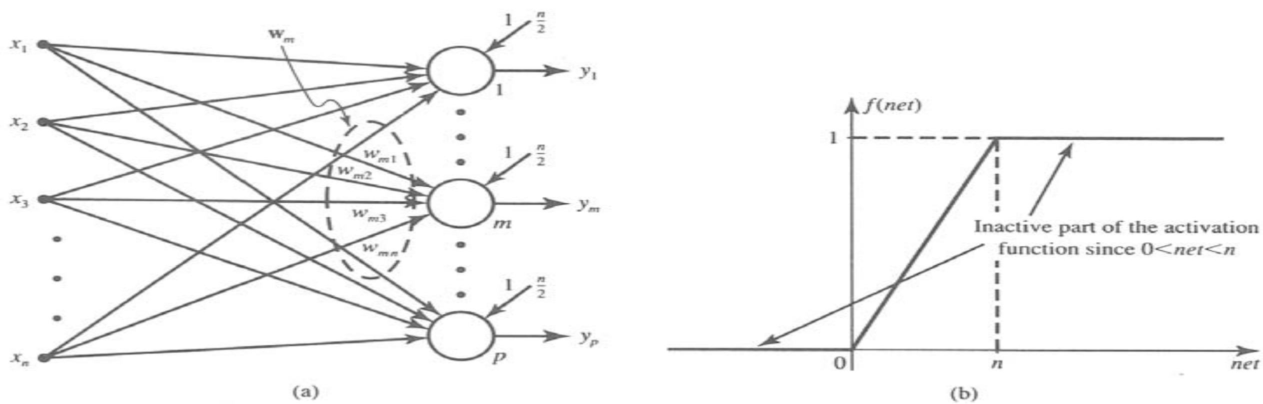


Figure 7.2 Hamming network for n-bit bipolar binary vectors representing p classes: (a) classifier network and (b) neurons' activation function.

Hamming Network Component

- $\mathrm{HD}\left(\boldsymbol{x}, \boldsymbol{s}^{(m)}\right)$, Hamming distance (HD), the number of different bit positions between two bipolar binary *n*-dimensional vectors, $\boldsymbol{x}^t$, $\boldsymbol{s}^{(m)}$
- The inner (scalar) product can be expressed as

$$\boldsymbol{x}^t \boldsymbol{s}^{(m)} = \underbrace{\left(n - \mathrm{HD}\left(\boldsymbol{x}, \boldsymbol{s}^{(m)}\right)\right)}_{\text{Number of bits agreed}} - \underbrace{\mathrm{HD}\left(\boldsymbol{x}, \boldsymbol{s}^{(m)}\right)}_{\text{Number of bits differed}}$$

$$\frac{1}{2}(\cdot) + \frac{n}{2}$$

- The neuron input

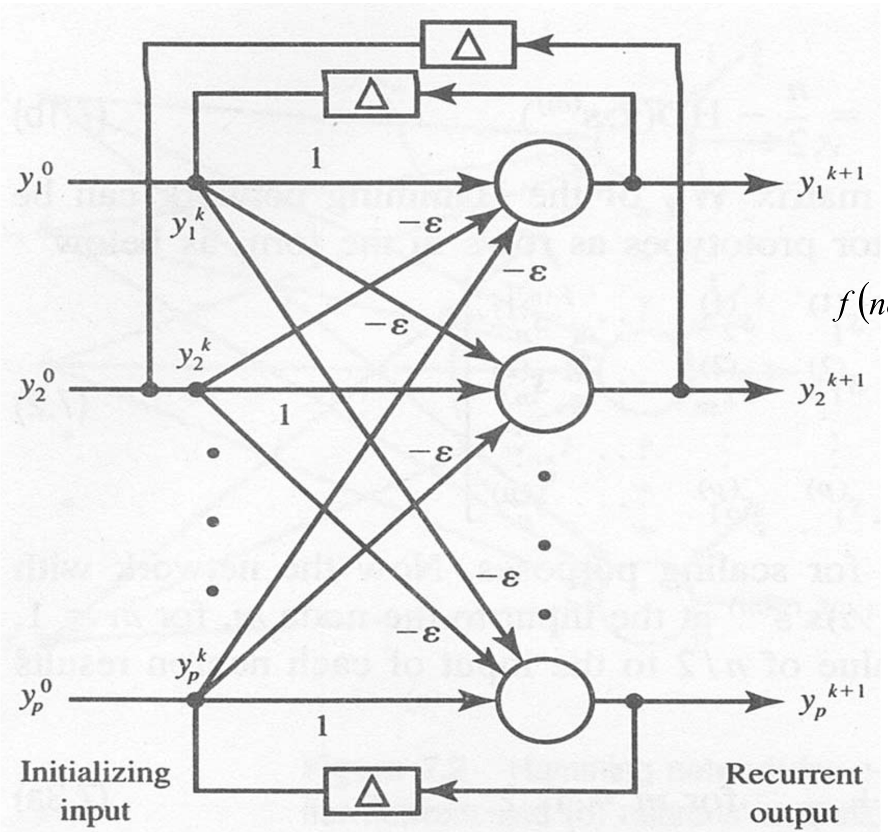
$$net_m = \frac{1}{2}\boldsymbol{x}^t \boldsymbol{s}^{(m)} + \frac{n}{2} = \boxed{n - \mathrm{HD}\left(\boldsymbol{x}, \boldsymbol{s}^{(m)}\right)}$$

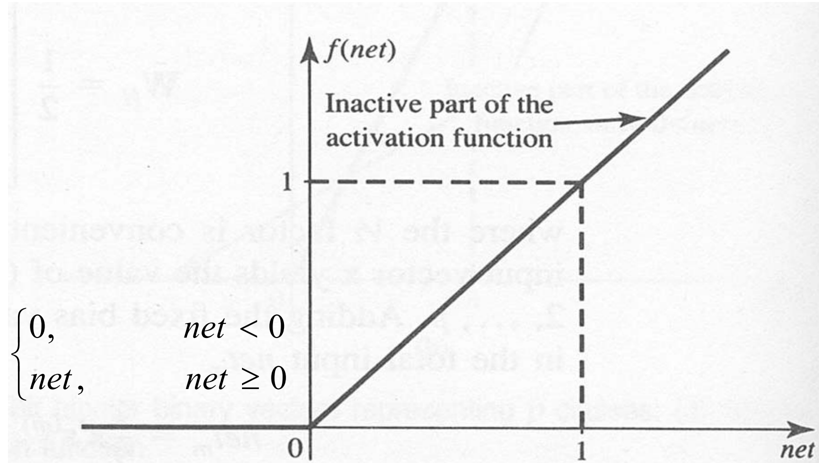
- The activation function $f\left(net_m\right) = \frac{1}{n} net_m$
  - The output of each neuron is scaled down to between 0 and 1
  - A prefect match of input to a specific class results in $f\left(net_m\right) = 1$

MAXNET Component

- Suppress values at MAXNET output neurons other than the initially maximum output neuron of the Hamming network
  - The excitatory connection implemented with a positive self-feedback loop with a weighting coefficient of *1*
  - The remaining inhibitory connections represent *p-1* cross-feedbacks with coefficients $-\varepsilon$ from each output
  - Recurrently update the outputs until all value except for one become zeros

MAXNET Component





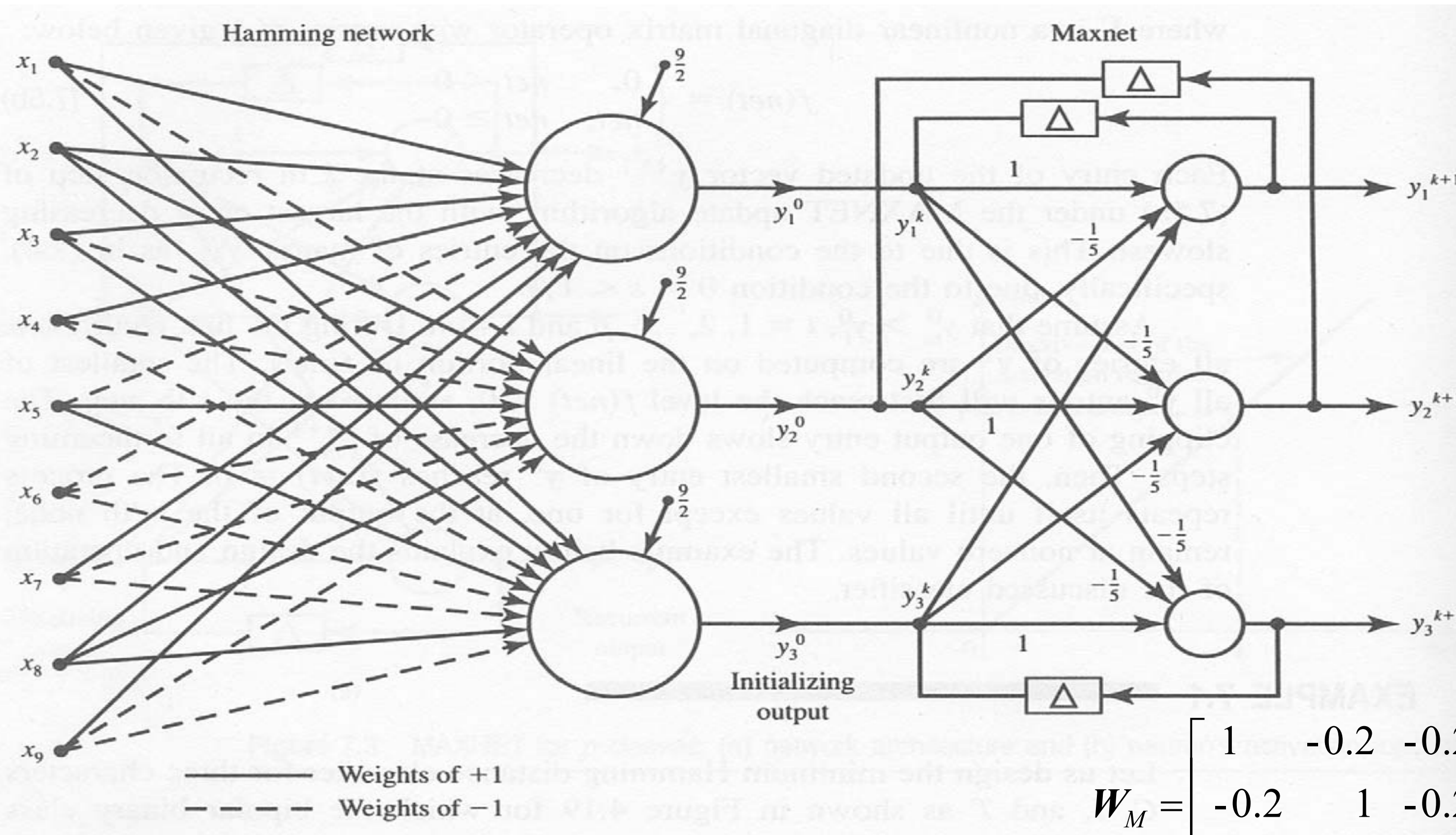
$$f(net)=\begin{cases}0, & net<0\\ net, & net\geq 0\end{cases}$$

$$W_M=\begin{bmatrix}1 & -\varepsilon & . & . & -\varepsilon\\ -\varepsilon & 1 & . & . & -\varepsilon\\ . & & & & \\ . & & & & \\ -\varepsilon & -\varepsilon & . & . & 1\end{bmatrix}, \quad 0<\varepsilon<\frac{1}{p}$$

$$\mathbf{y}^{k+1}=\Gamma\left[\mathbf{W}_M\,\mathbf{y}^{k}\right]$$

Hamming network and MAXNET

- **Example 7.1**

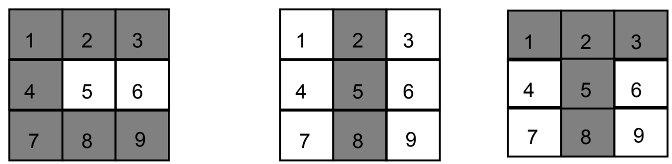


$$W_M = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix}$$

Figure 7.4 Hamming network and MAXNET for Example 7.1.

Hamming network and MAXNET

- **Example 7.1**
  - Three prototype characters *C*, *I*, and *T*



$$
\begin{aligned}
\boldsymbol{s}^{(1)} &= \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \end{bmatrix} \\
\boldsymbol{s}^{(2)} &= \begin{bmatrix} -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix} \\
\boldsymbol{s}^{(3)} &= \begin{bmatrix} 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}
\end{aligned}
\quad \Rightarrow \quad
\boldsymbol{W}_H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}
$$

prototype class vectors **weight matrix of Hamming network**

$$
\boldsymbol{net}_H = \frac{1}{2}\boldsymbol{W}_H\boldsymbol{x} + \begin{bmatrix} 9/2 \\ 9/2 \\ 9/2 \end{bmatrix}
$$

$$
\boldsymbol{net}_H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 9/2 \\ 9/2 \\ 9/2 \end{bmatrix}
$$

Hamming network and MAXNET

- **Example 7.1**
  - Input vector $x = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

$$\boldsymbol{net}_H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \end{bmatrix}\boldsymbol{x} + \begin{bmatrix} 9/2 \\ 9/2 \\ 9/2 \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \\ 5 \end{bmatrix} \xrightarrow{f(net_m) = \frac{1}{n} net_m} \Gamma\left[\boldsymbol{net}_H\right] = \begin{bmatrix} 7/9 \\ 3/9 \\ 5/9 \end{bmatrix}$$

$$\boldsymbol{net}_M^0 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix}\begin{bmatrix} 7/9 \\ 3/9 \\ 5/9 \end{bmatrix} = \begin{bmatrix} 0.599 \\ 0.067 \\ 0.333 \end{bmatrix} \xrightarrow{\mathbf{y}_1 = \Gamma\left[\mathbf{net}_M^0\right]} \boldsymbol{y}_1 = \Gamma\left[\boldsymbol{net}_H\right] = \begin{bmatrix} 0.599 \\ 0.067 \\ 0.333 \end{bmatrix}$$

$$\boldsymbol{net}_M^1 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix}\begin{bmatrix} 0.599 \\ 0.067 \\ 0.333 \end{bmatrix} = \begin{bmatrix} 0.520 \\ -0.120 \\ 0.120 \end{bmatrix} \xrightarrow{\mathbf{y}_2 = \Gamma\left[\mathbf{net}_M^1\right]} \boldsymbol{y}_2 = \begin{bmatrix} 0.520 \\ 0 \\ 0.120 \end{bmatrix}$$

$$\mathbf{net}_M^2 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix}\begin{bmatrix} 0.520 \\ 0 \\ 0.120 \end{bmatrix} = \begin{bmatrix} 0.480 \\ -0.14 \\ 0.096 \end{bmatrix} \xrightarrow{\mathbf{y}_3 = \Gamma\left[\mathbf{net}_M^2\right]} \boldsymbol{y}_3 = \begin{bmatrix} 0.480 \\ 0 \\ 0.096 \end{bmatrix}$$

$$\boldsymbol{net}_M^3 = \begin{bmatrix} 1 & -0.2 & -0.2 \\ -0.2 & 1 & -0.2 \\ -0.2 & -0.2 & 1 \end{bmatrix}\begin{bmatrix} 0.480 \\ 0 \\ 0.096 \end{bmatrix} = \begin{bmatrix} 0.460 \\ -0.115 \\ 0 \end{bmatrix} \xrightarrow{\mathbf{y}_4 = \Gamma\left[\mathbf{net}_M^3\right]} \mathbf{y}_4 = \begin{bmatrix} 0.461 \\ 0 \\ 0 \end{bmatrix}$$

Slow down the rate of decrease of nonzero outputs

Unsupervised Learning of Clusters

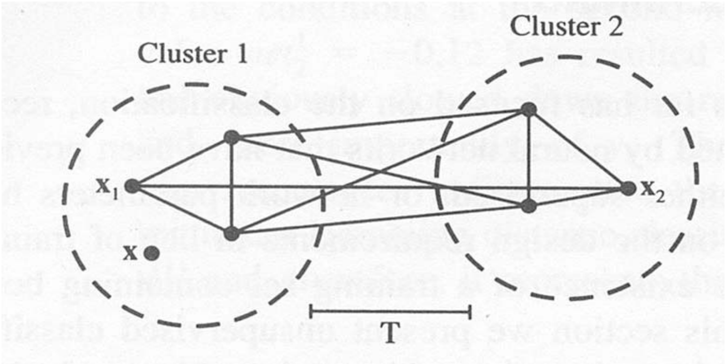
- Clustering or Unsupervised Classification
  - No a priori knowledge is assumed to be available regarding an input's membership in a particular class
  - Classify input vectors into one of the **specified number of *p* categories** during a self-organization process

- Clustering should be followed by labeling clusters with appropriate category names or numbers
  - The process of providing the category of objects with a label is termed as ***calibration***

Unsupervised Learning of Clusters

- **Similarity Criteria** for Clustering of Input Patterns
  1. Euclidean distance between two vectors

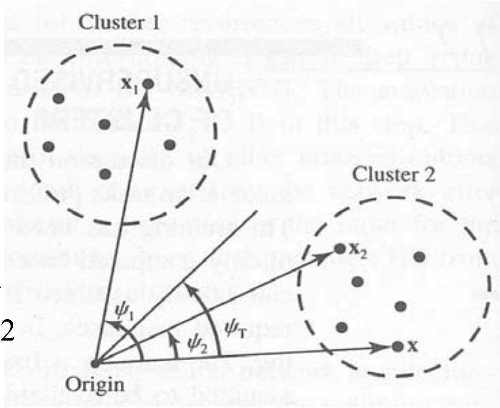
$$\left\|\boldsymbol{x}-\boldsymbol{x}_i\right\| = \sqrt{\left(\boldsymbol{x}-\boldsymbol{x}_i\right)^t\left(\boldsymbol{x}-\boldsymbol{x}_i\right)}$$



  2. Cosine of the angle between two vectors

$$\cos\psi = \frac{\boldsymbol{x}^t\boldsymbol{x}_i}{\left\|\boldsymbol{x}\right\|\left\|\boldsymbol{x}_i\right\|}$$

$$\psi_1 < \psi_T < \psi_2$$



Review:

Competitive (Winner-Take-All) Learning

- **Unsupervised learning**, and applicable for an ensemble of neurons (e.g. a layer of *p* neurons), not for a single neuron
- Adapt the neuron *m* which has the maximum response due to input **x**

$$w_m^t x = \max_{i=1,..,p} \left(w_i^t x \right)$$

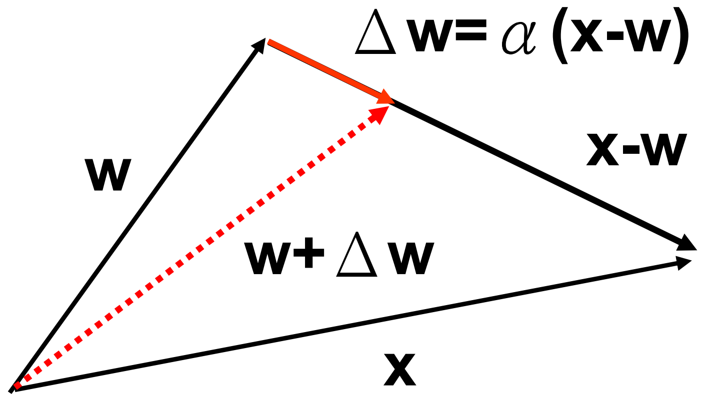
$$\Delta \mathbf{w}_i = \begin{cases} \alpha \left(\mathbf{x} - \mathbf{w}_i \right) & \text{if } i = m \\ \mathbf{0} & \text{if } i \neq m \end{cases}$$

Finding the weight vector closet to the input **x**

- Typically, it is used for learning the statistical properties of input patterns
  - Implemented with redundant input data

Review:
Competitive (Winner-Take-All) Learning

- Weights are typically initializing at random values and their lengths are normalized during learning
- The winner neighborhood is sometimes extended to beyond the single neuron winner to include the neighboring neurons

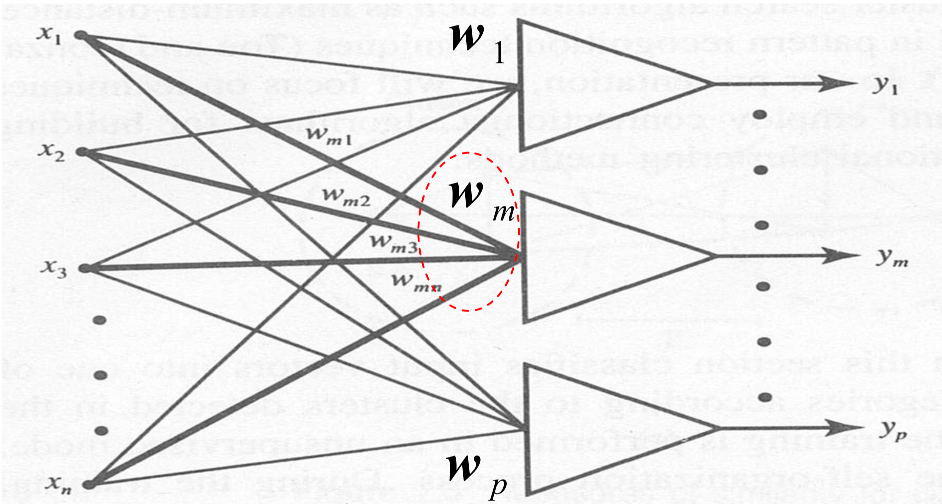


Unsupervised Learning of Clusters

Kohonen Network

(kohonen 1988)

- Suppose p categories are specified



An One-Layer Network

There are a set of p vectors needed to be learned

1. Normalize all weight vectors before learning:

$$\hat{\boldsymbol{w}}_m = \frac{\boldsymbol{w}_m}{\left\|\boldsymbol{w}_m\right\|}$$

Competitive Phase

2. Criterion for selection of candidate for weight adjustment

$$\left\|\boldsymbol{x}-\hat{\boldsymbol{w}}_m\right\| = \min_{i=1,2,\ldots,p}\left\{\left\|\boldsymbol{x}-\hat{\boldsymbol{w}}_i\right\|\right\} \Longrightarrow \hat{\boldsymbol{w}}_m^t\boldsymbol{x} = \max_{i=1,2,\ldots,p}\hat{\boldsymbol{w}}_i^t\boldsymbol{x}$$

$$\left\|\boldsymbol{x}-\hat{\boldsymbol{w}}_i\right\| = \left(\boldsymbol{x}^t\boldsymbol{x}-2\boldsymbol{x}^t\hat{\boldsymbol{w}}_i+1\right)^{1/2}$$

Unsupervised Learning of Clusters

Kohonen Network

Reward Phase

3. Weight adjustment for the winner neuron in the **negative gradient direction**

Gradient ⟹ $\nabla_{\hat{\boldsymbol{w}}_m} \left\| \boldsymbol{x} - \hat{\boldsymbol{w}}_m \right\|^2 = -2\left(\boldsymbol{x} - \hat{\boldsymbol{w}}_m\right)$

Minimal distance ⟹ $\Delta \hat{\boldsymbol{w}}_m = \alpha\left(\boldsymbol{x} - \hat{\boldsymbol{w}}_m\right)$ usually $0.1 \leq \alpha \leq 0.7$

Updated weights in the *k+1*th iteration

Winner-Take-All Learning

$$\begin{cases} \hat{\boldsymbol{w}}_m^{k+1} = \hat{\boldsymbol{w}}_m^{k} + \alpha^k\left(\boldsymbol{x} - \hat{\boldsymbol{w}}_m^{k}\right) \\ \hat{\boldsymbol{w}}_i^{k+1} = \hat{\boldsymbol{w}}_i^{k}, \quad \text{for}\, i \neq m \end{cases}$$

α is monotonically reduced in iterations

$$\left(\hat{\boldsymbol{w}}_m + \Delta \hat{\boldsymbol{w}}_m^t\right)\boldsymbol{x} \geq \hat{\boldsymbol{w}}_m^t \boldsymbol{x} \quad ?$$

$$\Delta \hat{\boldsymbol{w}}_m^t \boldsymbol{x} \geq 0 \Rightarrow \left(\boldsymbol{x} - \hat{\boldsymbol{w}}_m\right)\boldsymbol{x} \geq 0$$

$$\left\|\boldsymbol{x}\right\|^t \left\|\boldsymbol{x}\right\| \cos 0 - \left\|\hat{\boldsymbol{w}}_m\right\|^t \left\|\boldsymbol{x}\right\| \cos \psi \geq 0$$

$$1 - \cos \psi \geq 0$$

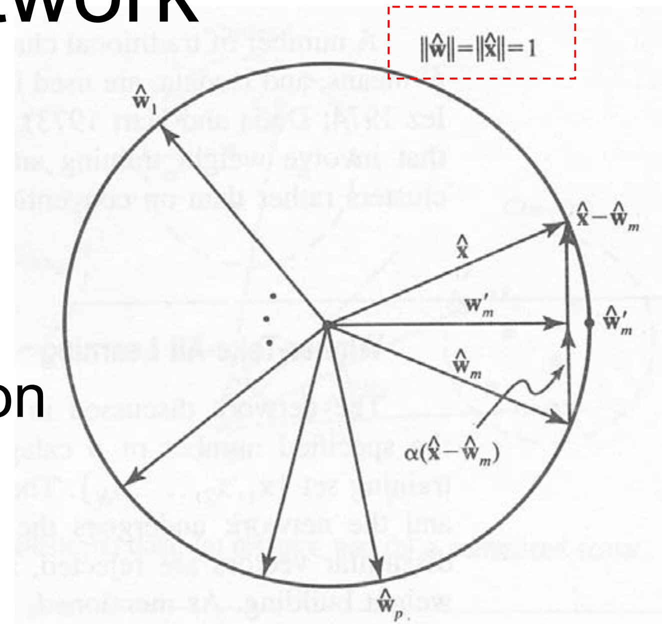
What about the idea of excitatory/inhibitory connections of neurons from MAXNET?

(Assume that $\boldsymbol{x}$ is a normalized vector)

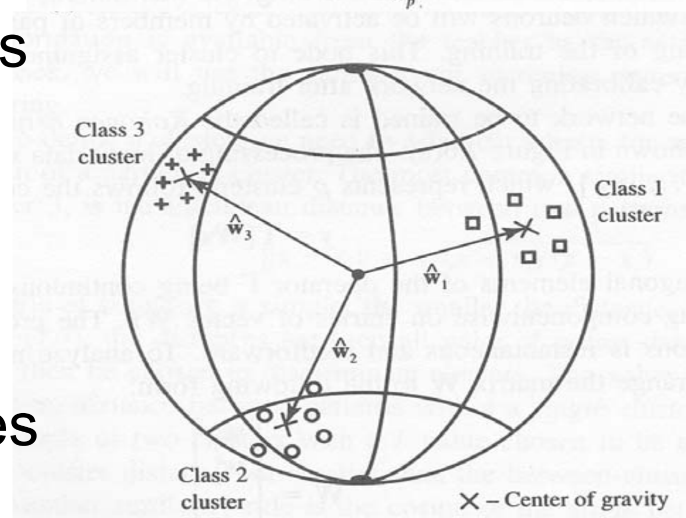
Kohonen Network

Reward Phase

- After the learning, each $\hat{w}_m$ represents the centroid of an *i*-th decision region
- The neuron's activation function is irrelevant to this learning



- Variation/Extension
  - Proper class for some patterns is known *a priori*
  - **Leaky competitive learning**: both the winners' or losers' weights are adjusted in proportion to their responses



Unsupervised Learning of Clusters

Kohonen Network

Recall Phase

- Forward recall at all *p* neuron outputs

$$y_m = \max\left(y_1, y_2, \dots, y_p\right)$$

- Supervised calibration is needed for one-to-one vector-to-cluster mapping
  - Calibration to physical neurons depends on the sequence of training data set, parameters, and initial weights

Kohonen Network

- **Initialization of weights**
  - Initial weights should be uniformly distributed on the unity hyper-sphere
- **Limitation of Kohonen Network**
  - Can't efficiently handle linearly nonseparable patterns because of its single-layer structure
  - May not always be successful even for linearly separable patterns because of getting stuck in isolated regions without forming adequate clusters

Solutions
    - Fine tuning of α
    - Supplement of an excessive number of neurons
- Multiple-winner unsupervised learning

Kohonen Network

- Example 7.2:Two-Cluster Case

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5\} = \left\{ \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}, \begin{bmatrix} 0.1736 \\ 0.9848 \end{bmatrix}, \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}, \begin{bmatrix} 0.342 \\ -0.9397 \end{bmatrix}, \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \right\}$$

$$w_1^0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad w_2^0 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

winner

$$\boldsymbol{w}^{0t} x_1 = \begin{bmatrix} 1 & 0 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.8 \end{bmatrix}$$

$$\alpha\left(\boldsymbol{x} - \boldsymbol{w}_1^0\right) = \frac{1}{2}\left(\begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} - \begin{bmatrix} 1.0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} -0.1 \\ 0.3 \end{bmatrix}$$

$$w_1^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -0.1 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.3 \end{bmatrix}$$

normalize

$$\hat{w}_1^1 = \begin{bmatrix} 0.948 \\ 0.316 \end{bmatrix}$$

$$\hat{w}_2^1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

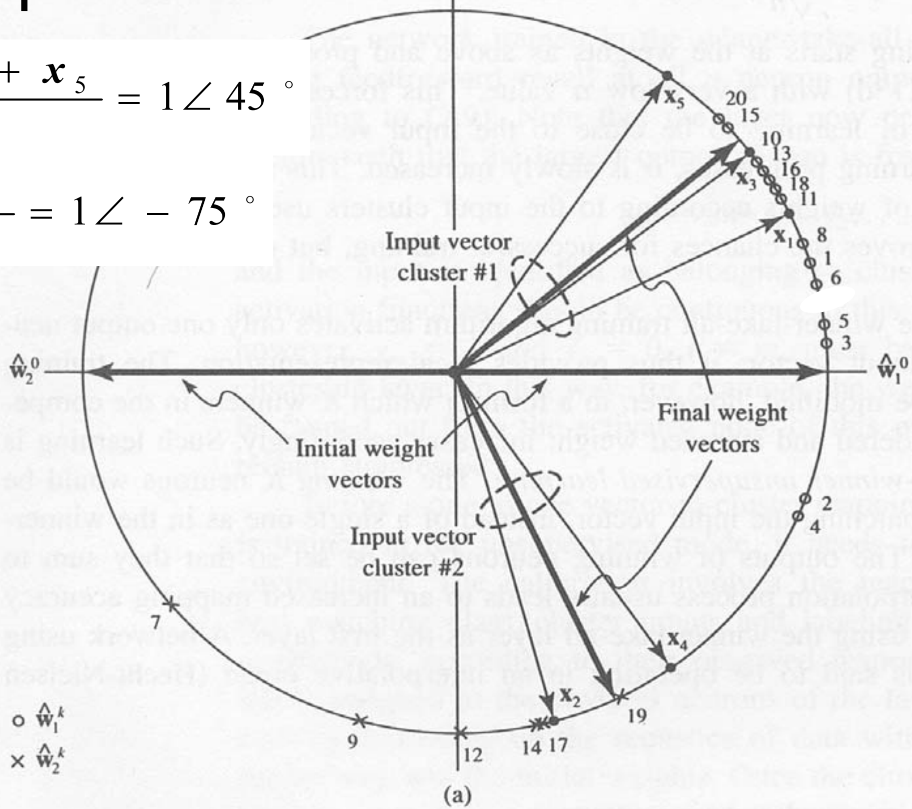
Unsupervised Learning of Clusters

Kohonen Network

- Example 7.2

$$\frac{x_1 + x_3 + x_5}{3} = 1\angle 45^\circ$$

$$\frac{x_2 + x_4}{2} = 1\angle -75^\circ$$



	cluster 1	cluster 2
Step k	$\hat{w}_1^k$ ∡ deg	$\hat{w}_2^k$ ∡ deg
1	18.46	−180.00°
2	−30.77	—
3	7.11	—
4	−31.45	—
5	10.85	—
6	23.86	—
7	—	−130.22
8	34.43	—
9	—	−100.01
10	43.78	—
11	40.33	—
12	—	−90.00
13	42.67	—
14	—	−80.02
15	47.90	—
16	42.39	—
17	—	−80.01
18	43.69	—
19	—	−75.01
20	48.42	—

(weight vectors of unity length)
(— means no change)

(b)

Final Weights

Figure 7.7 Competitive learning network of Example 7.2: (a) training patterns and weight assignments and (b) weight learning, Steps 1 through 20.

Unsupervised Learning of Clusters

- Separability Limitation

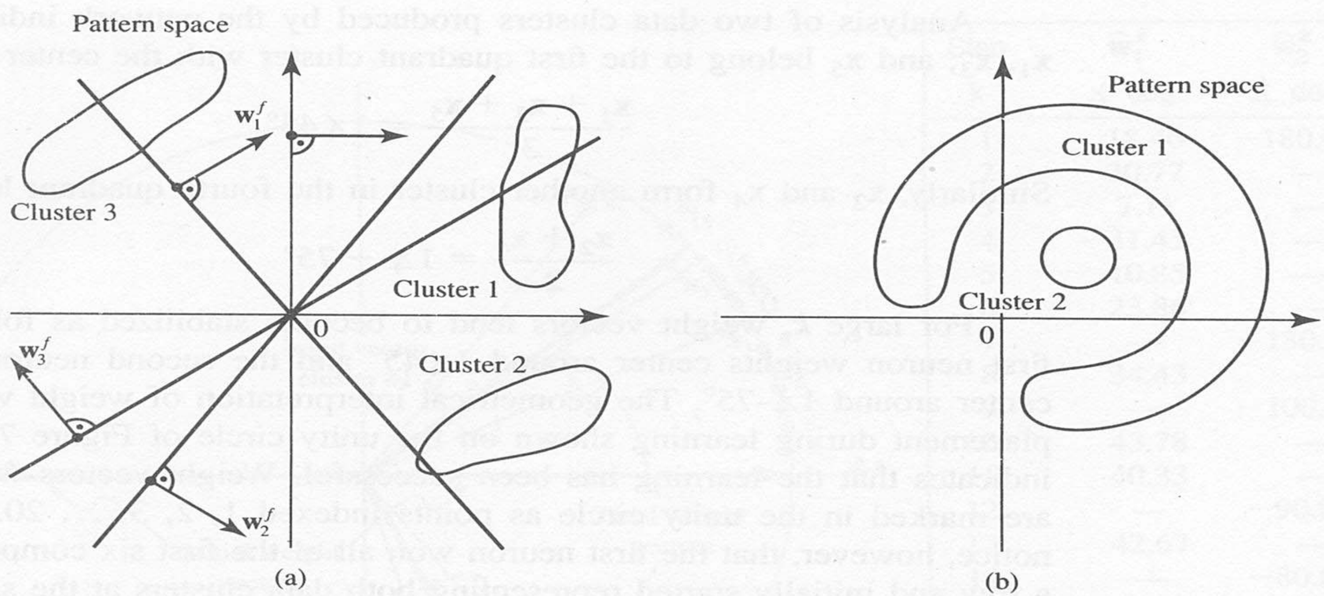


Figure 7.8 Separability illustration for a winner-take-all learning network ($\mathbf{w}_f$ denotes the final weight vectors): (a) possible classification and (b) impossible classification.

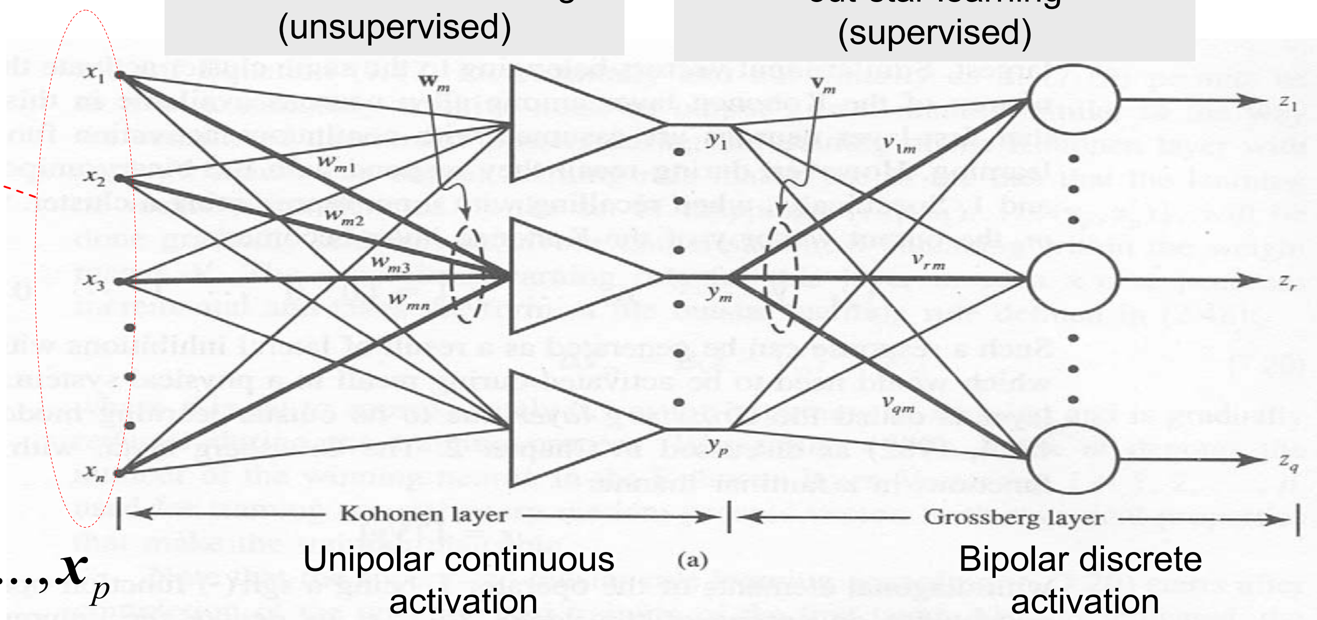
Counterpropagation Network

Hecht-Nielsen 1987,1988

- Vector-to-Vector Mapping (heteroassociation)
  - Two-layer, feedforward
  - No feedback and delay during recall

winner-take-all learning (unsupervised)

out-star learning (supervised)



Bipolar binary vector

$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_p$

Unipolar continuous activation

Bipolar discrete activation

$\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_p$

Counterpropagation Network

- The Simple feedforward Version
- First layer: Kohonen layer
  - Trained in the unsupervised winner-take-all mode
  - Each of the neurons represents an input/pattern cluster
  - The adjustment of weight vectors is in proportional to the probability of the occurrence and distribution of winner events
  - **During recall**, neurons respond with **binary unipolar** value 0 an 1, e.g.

$$\left[y_1 \, y_2 \, .. \, y_m \, .. \, y_p\right] = \left[0 \; 0 \; 0 \;\; ... \; 1 \; ... \;\; 0\right]$$

Counterpropagation Network

- Second layer: Grossberg layer
  - The weights of the second layer tend to converge to the average values of the desired output vectors associated with the input

$$z = \Gamma[\boldsymbol{V}\boldsymbol{y}] = \Gamma[\boldsymbol{v}_m]$$

$$z_i = 1 \quad \text{if} \quad v_{im} > 0$$
$$z_i = -1 \quad \text{if} \quad v_{im} < 0$$

$$\boldsymbol{V}\boldsymbol{y} = \begin{bmatrix} v_{11} & v_{21} \cdots & \boxed{v_{m1}} & \cdots v_{p1} \\ v_{12} & v_{22} \cdots & v_{m2} & \cdots v_{p2} \\ \cdots\cdots & \cdots & \cdots\cdots & \cdots \\ v_{1q} & v_{2q} \cdots & v_{mq} & \cdots v_{pq} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ y_q \end{bmatrix} \cong \begin{bmatrix} v_{m1} \\ v_{m2} \\ \cdot \\ v_{mq} \end{bmatrix}$$

$$\boldsymbol{v}_m$$

Counterpropagation Network

- Second layer: Grossberg layer
  - Make use of the learning pairs of vectors $\{(\boldsymbol{x}_1, \boldsymbol{z}_1), \ldots, (\boldsymbol{x}_p, \boldsymbol{z}_p)\}$
  - Out-star learning rule for learning the statistical properties

Oversimplified adaptation

Incremental adaptation

$$\boxed{\boldsymbol{v}_m = \boldsymbol{z}}$$

or

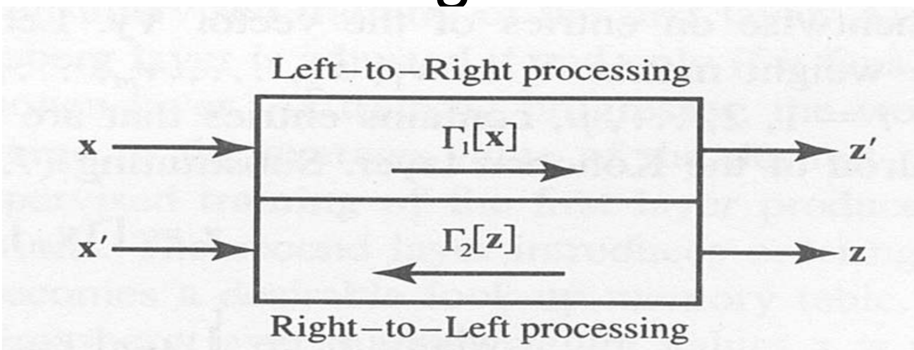
$$\Delta \boldsymbol{v}_m = \beta(\boldsymbol{z} - \boldsymbol{v}_m), \quad \beta \approx 0.1, \text{gradually reduced}$$

$$\boldsymbol{v}_m = \boldsymbol{v}_m + \Delta \boldsymbol{v}_m$$

  - Only the weights fan out from the winner neuron in the first layer are adjusted
  - The weights of the this layer tend to coverage to the average of the desired output

Counterpropagation Network

- Counterflow of Signals



$$\begin{bmatrix} z' \\ x' \end{bmatrix} = \begin{bmatrix} \Gamma_1[x] \\ \Gamma_2[z] \end{bmatrix}$$

Bidirectional Table Lookup

Feature Mapping

- Learn feature mapping **without supervision** from the input space into feature space
  - Two aspects of mapping features
    - Reduce the dimensionality of vector in pattern space
    - Facilitate perception, provide as natural a structure of feature as possible

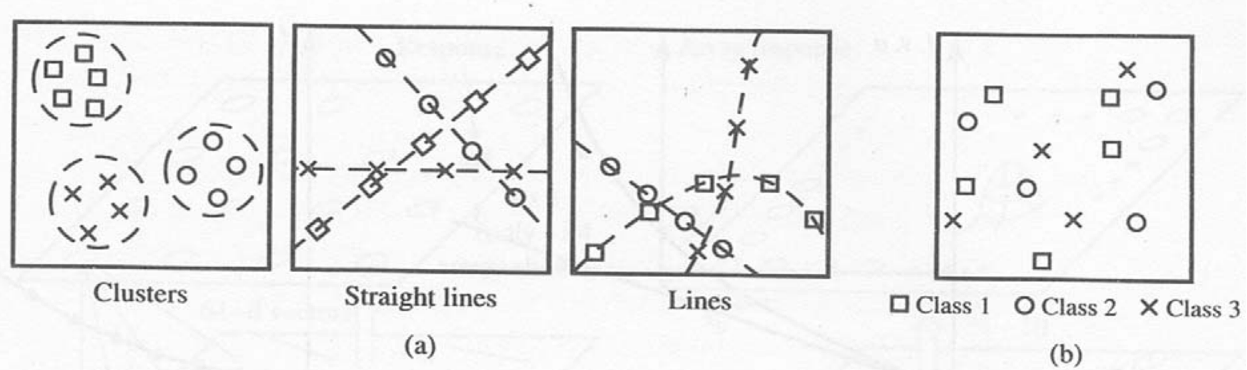


Figure 7.10 Pattern structure: (a) natural similarity and (b) no natural similarity.

Feature Mapping

- Each component in the input vector is connected to each of the nodes

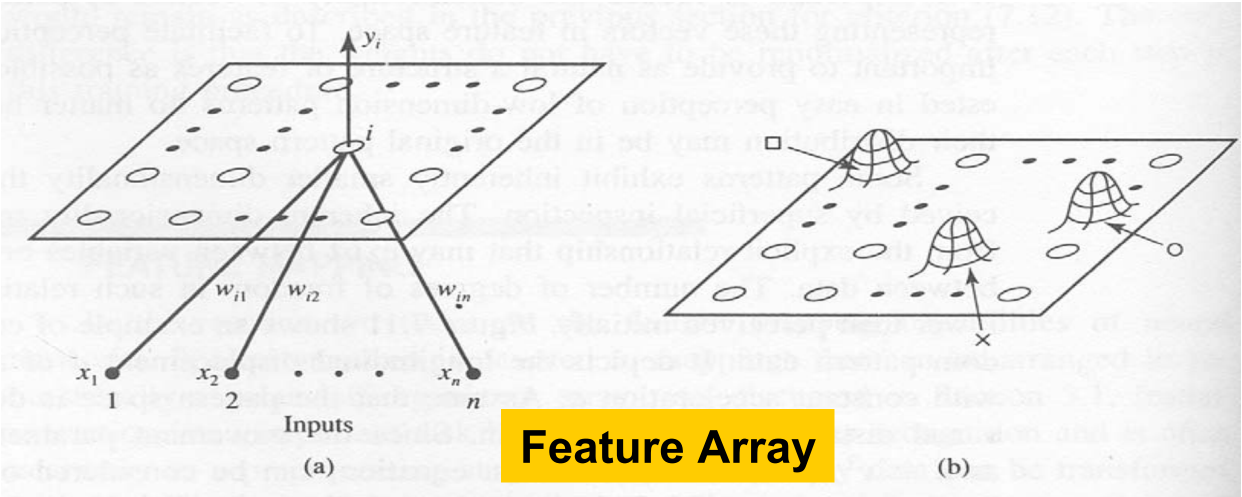


Figure 7.12 Mapping features of input **x** into a rectangular array of neurons: (a) general diagram and (b) desirable response peaks for Figure 7.10(b).

Dimension Reduction
Natural Structure

Similarity Metric

$$y_i = f\left(S\left(\boldsymbol{x}, \boldsymbol{w}_i\right)\right)$$

Feature Mapping

- A Linear Array of Neurons
  - Feature Map with Lateral Feedbacks

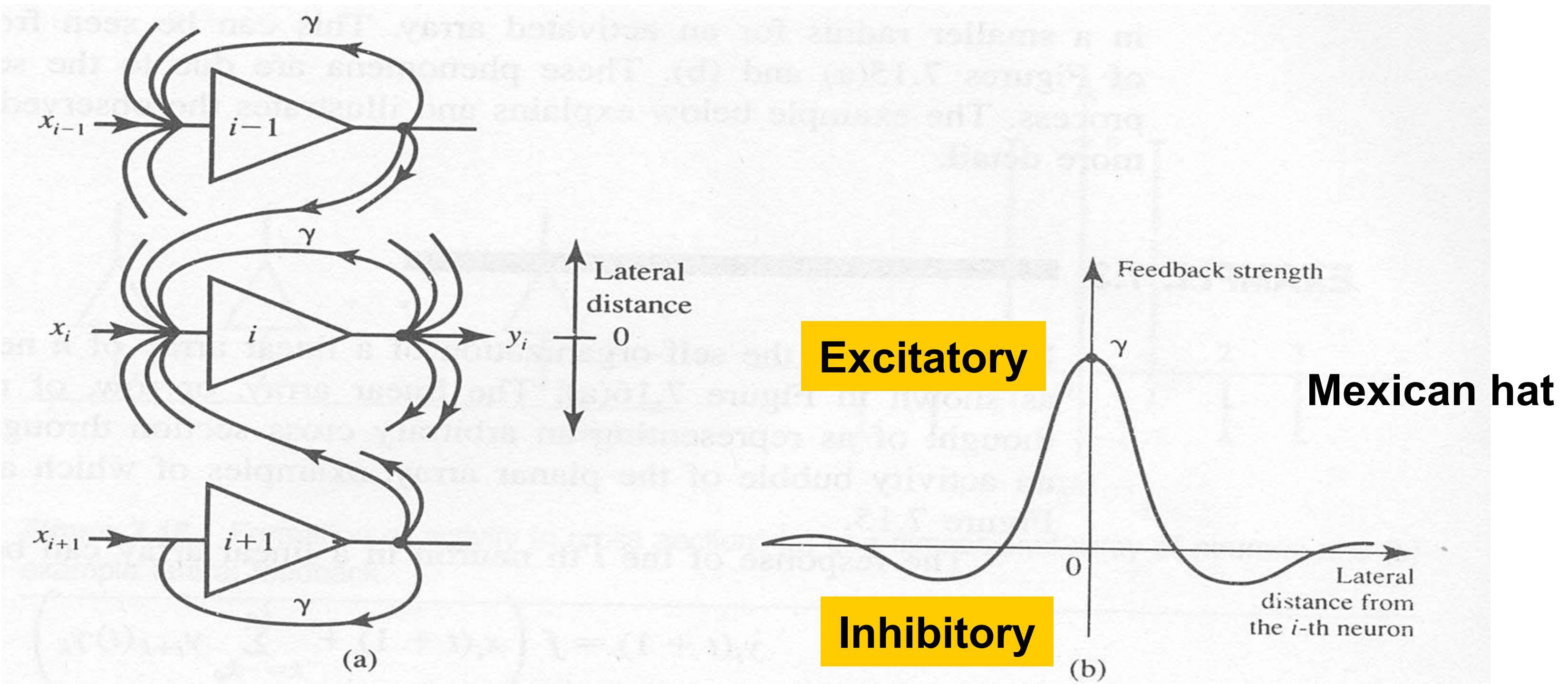


Figure 7.14 Lateral connections for the clustering demonstration: (a) interconnections of neurons and (b) lateral- and self-feedback strength.

Feature Mapping

- A Planar Array: A Two-Dimensional Layer of Neurons
  - Feature Map with Lateral Feedbacks

Activity Bubble

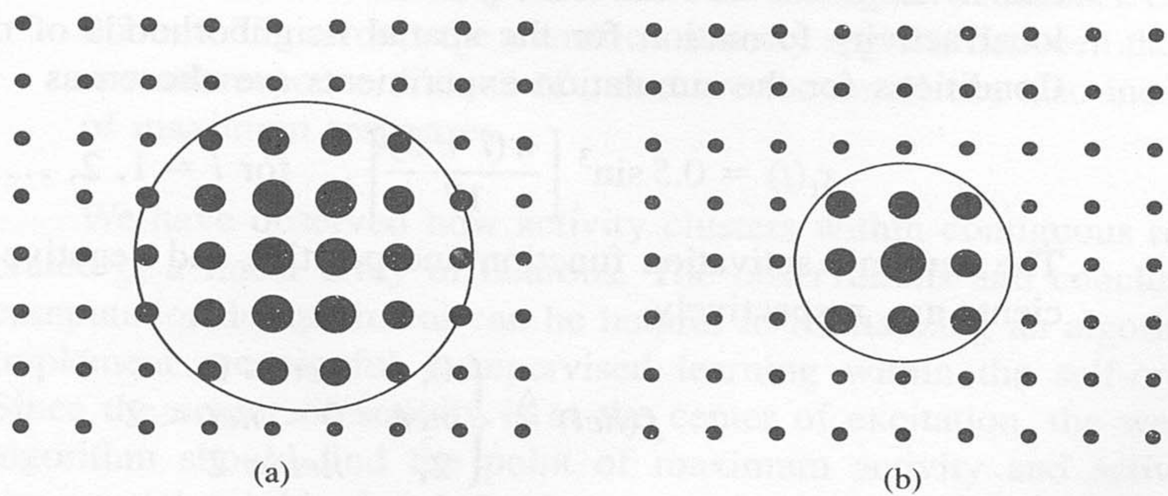


Figure 7.15 Planar activity formation for various strengths of lateral interaction: (a) strong positive feedback and (b) weak positive feedback. [Adapted from Kohonen (1984). © Springer Verlag; with permission.]

Feature Mapping

- **Example 7.3:** A Linear Array of Neurons

$$x_i(t) = 0.5\sin^3\left[\frac{\pi(i+3)}{17}\right], \quad \text{for } i = 1,2,\ldots,10$$

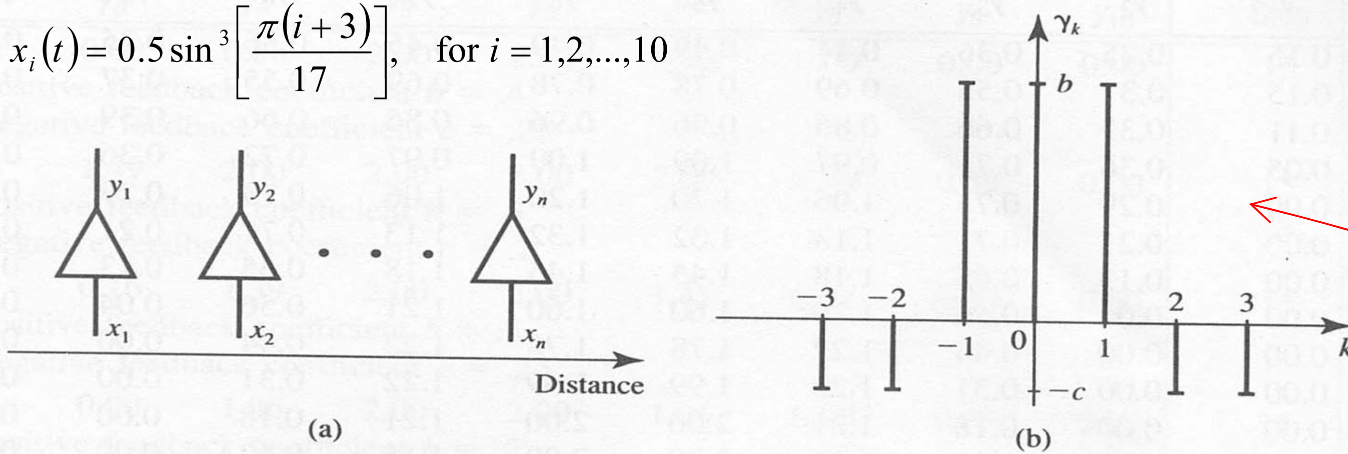


Figure 7.16 Formation of activity in cross section: (a) one-dimensional array of neurons and (b) example lateral feedback.

$$f(net) \overset{\Delta}{=} \begin{cases} 0, & net \le 0 \\ net, & 0 < net < 2 \\ 2, & net > 0 \end{cases}$$

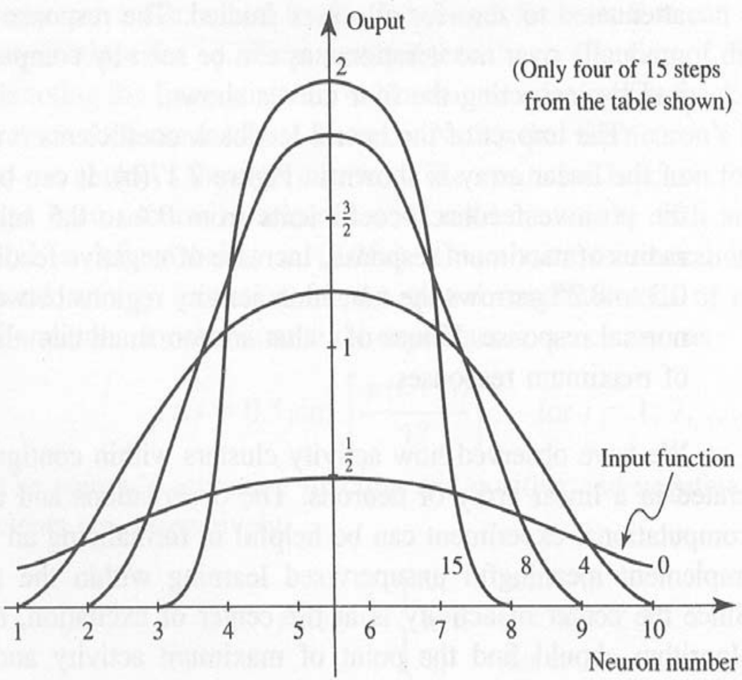
$$b = 0.4, \quad c = 0.2$$

$$y_i(t+1) = f\left(x_i(t+1) + \sum_{k=-k_0}^{k_0} y_{i+k}(t)\gamma_k\right)$$

γ_k : a function of interneuronal distance

Feature Mapping

- **Example 7.3:** results at different iterations



y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	Step
0.15	0.25	0.36	0.44	0.49	0.49	0.45	0.36	0.25	0.15	0
0.15	0.37	0.55	0.69	0.78	0.78	0.69	0.55	0.37	0.16	1
0.11	0.39	0.66	0.86	0.96	0.96	0.86	0.66	0.39	0.11	2
0.05	0.36	0.71	0.97	1.09	1.09	0.97	0.72	0.36	0.05	3
0.00	0.29	0.73	1.06	1.20	1.20	1.06	0.73	0.29	0.00	4
0.00	0.21	0.71	1.13	1.32	1.32	1.13	0.71	0.21	0.00	5
0.00	0.13	0.65	1.18	1.45	1.45	1.18	0.65	0.13	0.00	6
0.00	0.04	0.56	1.20	1.60	1.60	1.21	0.56	0.04	0.00	7
0.00	0.00	0.44	1.22	1.78	1.78	1.22	0.44	0.00	0.00	8
0.00	0.00	0.31	1.22	1.99	1.99	1.22	0.31	0.00	0.00	9
0.00	0.00	0.18	1.21	2.00	2.00	1.21	0.18	0.00	0.00	10
0.00	0.00	0.11	1.16	2.00	2.00	1.16	0.12	0.00	0.00	11
0.00	0.00	0.07	1.12	2.00	2.00	1.12	0.07	0.00	0.00	12
0.00	0.00	0.03	1.09	2.00	2.00	1.10	0.04	0.00	0.00	13
0.00	0.00	0.01	1.08	2.00	2.00	1.08	0.01	0.00	0.00	14
0.00	0.00	0.00	1.06	2.00	2.00	1.07	0.00	0.00	0.00	15

Positive feedback coefficient $b = .4$
Negative feedback coefficient $c = .2$

Feature Mapping

- **Example 7.3:** with different *b* and *c*

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	Step
0.00	0.00	0.00	0.88	2.00	2.00	0.88	0.00	0.00	0.00	15
Positive feedback coefficient $b = .4$										
Negative feedback coefficient $c = .25$										
0.00	0.38	1.49	2.00	2.00	2.00	2.00	1.49	0.39	0.00	15
Positive feedback coefficient $b = .5$										
Negative feedback coefficient $c = .2$										
0.00	0.00	0.16	1.39	2.00	2.00	1.41	0.17	0.00	0.00	15
Positive feedback coefficient $b = .5$										
Negative feedback coefficient $c = .25$										
0.00	0.00	0.00	1.06	2.00	2.00	1.06	0.00	0.00	0.00	15
Positive feedback coefficient $b = .5$										
Negative feedback coefficient $c = .3$										
0.00	0.00	0.00	0.88	2.00	2.00	0.88	0.00	0.00	0.00	15
Positive feedback coefficient $b = .5$										
Negative feedback coefficient $c = .35$										

  - Increase of positive feedback coefficients widens the radius of maximum responses
  - Increase of negative feedback coefficients narrows the transition activity regions

Feature Mapping

- The center of activity is the center of excitation, the weight adaptation algorithm should find the point of maximum activity and activate its respective spatial neighborhood

Self-Organization Feature Maps

- Self-Organization Feature Maps
  - Find the best neurons cells which also activate their spatial neighbors to react the same input

1. find the best matching neuron

$$\|\boldsymbol{x}-\boldsymbol{w}_m\| = \min_i \{\|\boldsymbol{x}-\boldsymbol{w}_i\|\}$$

2. weight updating

$$\Delta \boldsymbol{w}_i(t) = \alpha(N_i, t)[\boldsymbol{x}-\boldsymbol{w}_m] \quad \text{for } i \in N_m(t)$$

$$0 < \alpha(N_i, t) < 1$$

$N_m(t)$: the current spatial neighborhood

$$\alpha(N_i, t) = \alpha(t)\exp\left[\frac{-\|r_i - r_m\|}{\sigma^2(t)}\right]$$

r_m and r_i are position vectors of the winner neuron and its neighborhood neuron

$\alpha(t)$ and $\sigma(t)$ are decreasing in iterations

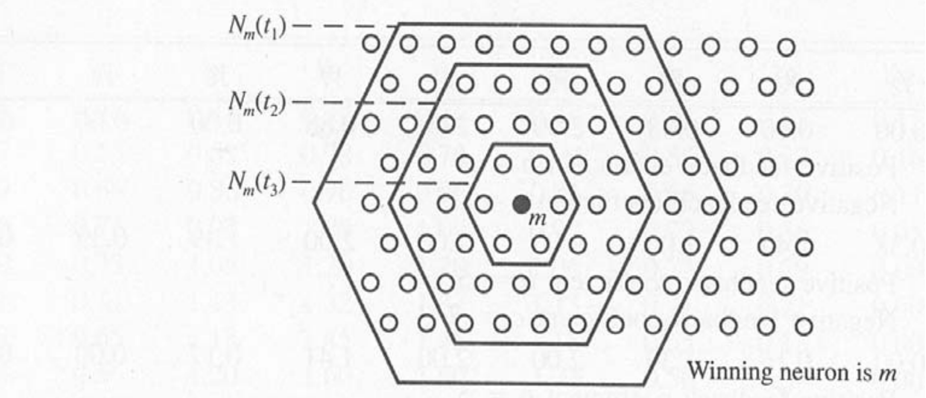


Figure 7.18 Topological neighborhood definition, $t_1 < t_2 < t_3 \ldots$

Self-Organization Feature Maps

- Main Considerations
  - The neurons are exposed to a sufficient number of inputs
  - Only the weights leading to an excited neighborhood are affected
  - The adjustment is in proportion to the activation received by each neuron within the neighborhood

Self-Organization Feature Maps

- Self-Organization Mapping for the Alphabet

Kohonen 1984

Attribute	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6
x_1	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
x_2	0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
x_3	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	3	6	6	6	6	6	6	6	6	6	6
x_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2	2	2
x_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6

(a)

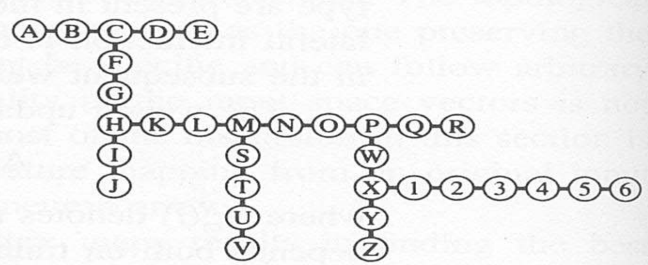
After 10K training steps and calibration

10

7

B	C	D	E	*	Q	R	*	Y	Z
A	*	*	*	*	P	*	*	X	*
*	F	*	N	O	*	W	*	*	1
*	G	*	M	*	*	*	*	2	*
H	K	L	*	T	U	*	3	*	*
*	I	*	*	*	*	*	*	4	*
*	J	*	S	*	*	V	*	5	6

(b)

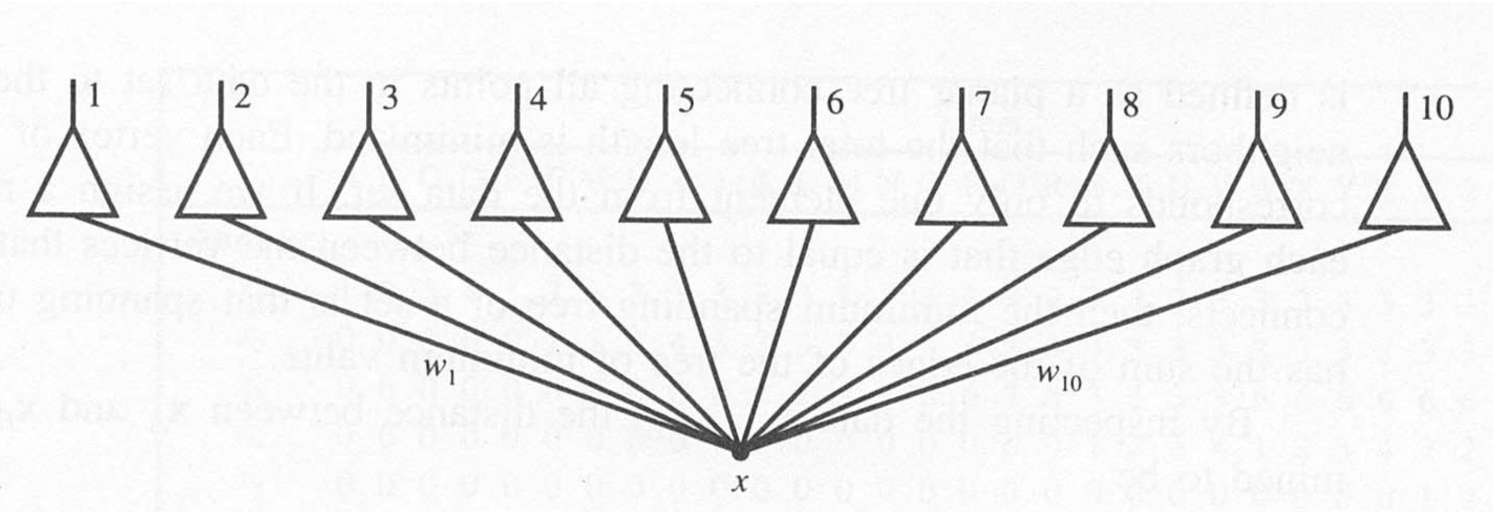


(c)

Figure 7.19 Self-organizing feature mapping example: (a) list of patterns, (b) feature map produced after training, and (c) minimum spanning tree. [from Kohonen (1984). © Springer Verlag; reprinted with permission.]

Self-Organization Feature Maps

- **Example 7.4**
  - A Linear Array with 10 neurons
  - Inputs are one-dimensional random variable uniformly distributed between 0 and 1



Self-Organization Feature Maps

- **Example 7.4**

10 experiments started at different initial values and trained with different random sequence

Weights w_1 - w_{10} (unordered) after 1000 training steps

0.859	0.961	0.753	0.052	0.181	0.297	0.541	0.398	0.469	0.637
0.041	0.124	0.429	0.315	0.223	0.646	0.736	0.544	0.845	0.952
0.528	0.778	0.707	0.623	0.323	0.872	0.963	0.433	0.069	0.210
0.169	0.064	0.690	0.481	0.879	0.960	0.790	0.275	0.385	0.583
0.049	0.152	0.590	0.651	0.721	0.954	0.833	0.498	0.257	0.372
0.061	0.182	0.265	0.707	0.521	0.596	0.357	0.450	0.814	0.937
0.073	0.202	0.436	0.933	0.798	0.669	0.497	0.552	0.378	0.302
0.920	0.975	0.837	0.610	0.724	0.501	0.315	0.402	0.202	0.067
0.052	0.165	0.684	0.471	0.366	0.274	0.752	0.942	0.835	0.583
0.061	0.265	0.163	0.782	0.853	0.949	0.617	0.707	0.497	0.387

Weights of neurons after 1000 training steps

1. Initial weights are random and centered around 0.5 with 0.05 radius

2. Winner neighborhood reduced to zero after 300 training steps

Self-Organization Feature Maps

- **Example 7.4**

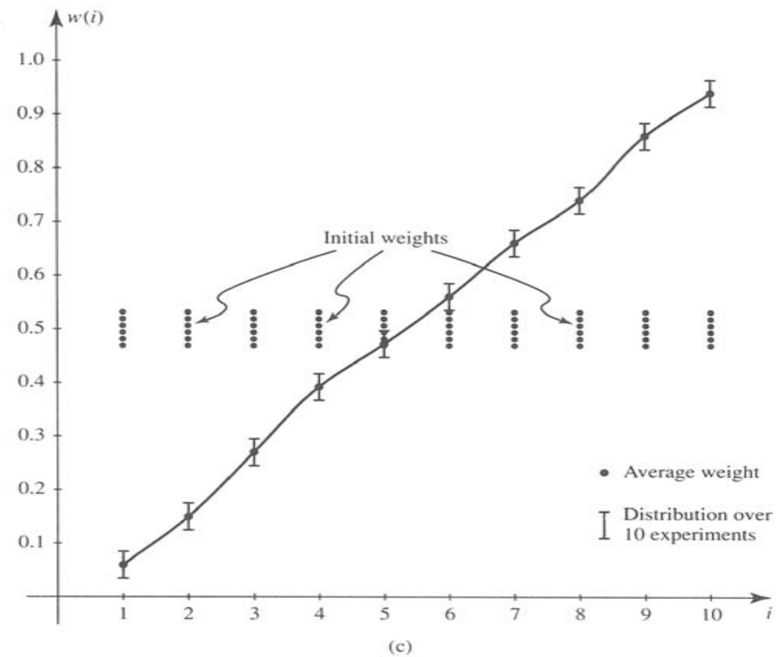


Figure 7.20c Linear array for Example 7.4 *(continued):* (c) weight values for an ordered linear array.

The self-organized result after calibration

Self-Organization Feature Maps

- Planner Visualization of Complex Multidimensional Speech Spectra

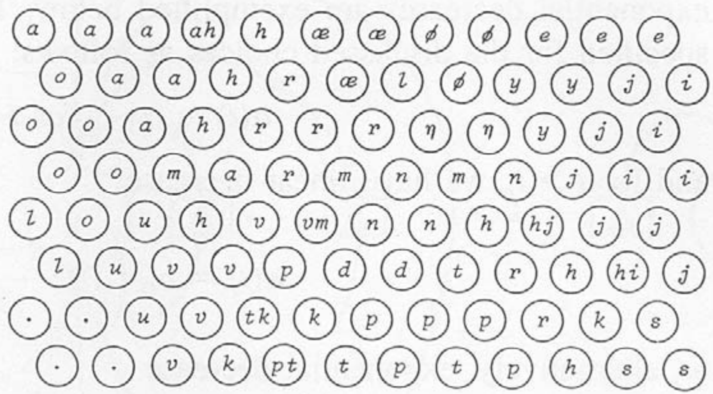


Figure 7.22 Speech phoneme map after training. [from Kohonen (1990). © IEEE; reprinted with permission.]

Finnish Speech

Self-Organization Feature Maps

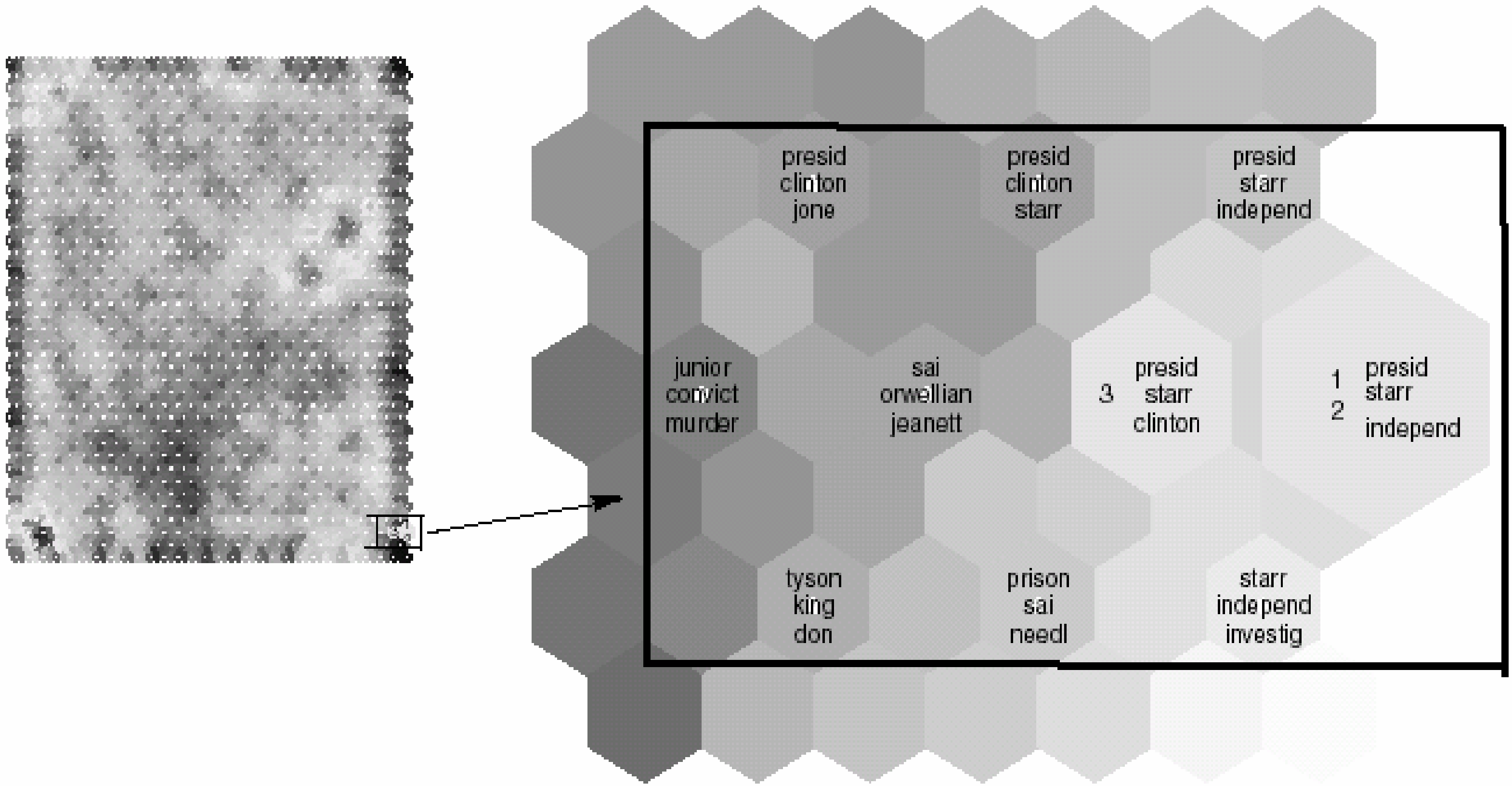


Fig. 2. Displaying the latent document topics in a 2D map of hexagons. The original query was "Lewensky" (sic.). The closest map cells for the three best documents are shown with magnified hexagons. The topic labeling used in Fig. 1 is here extended to the three best index terms.

Self-Organization Feature Maps

- More about learning function $\alpha(t)$
  - Adaptively decreased

$$\alpha(t) = \alpha_0, \text{ for } \quad t < t_p$$

$$\alpha(t) = \alpha_0 \left(1 - \frac{t - t_p}{t_q} \right), \quad \text{for} \quad t \geq t_p$$

or $$\alpha(t) = \alpha_0 \exp\left(- \frac{t - t_p}{t_q} \right), \quad \text{for} \quad t \geq t_p$$

Adaptive Resonance Theory 1 (ART1)

調適共振理論

- ART1’s Characteristics
  - Input patterns are unipolar binary vectors
  - A single Kohonen layer with competitive learning neurons
  - Remain Stability and Plasticity
  - Controlled discovery of clusters (cluster number not predefined) without a prior information about the possible number and type of clusters
  - Accommodate new clusters without affecting the storage or recall capabilities for clusters already learned

- ART2 is for continuous input patterns

Adaptive Resonance Theory 1 (ART1)

1. Originate the first cluster after receiving the first input pattern
2. Then create the another new cluster if the distance of the following input pattern to the existed clusters exceeds a certain threshold

- The process of pattern inspection followed by
  - Either new cluster to be originated
  - Or the input to be accepted to the previous encoded (old) cluster

Adaptive Resonance Theory 1 (ART1)

Two Tests for Clustering:
1. Down-to-Up Test
Long-term memory

2. Up-to-Down Test
Short-term memory

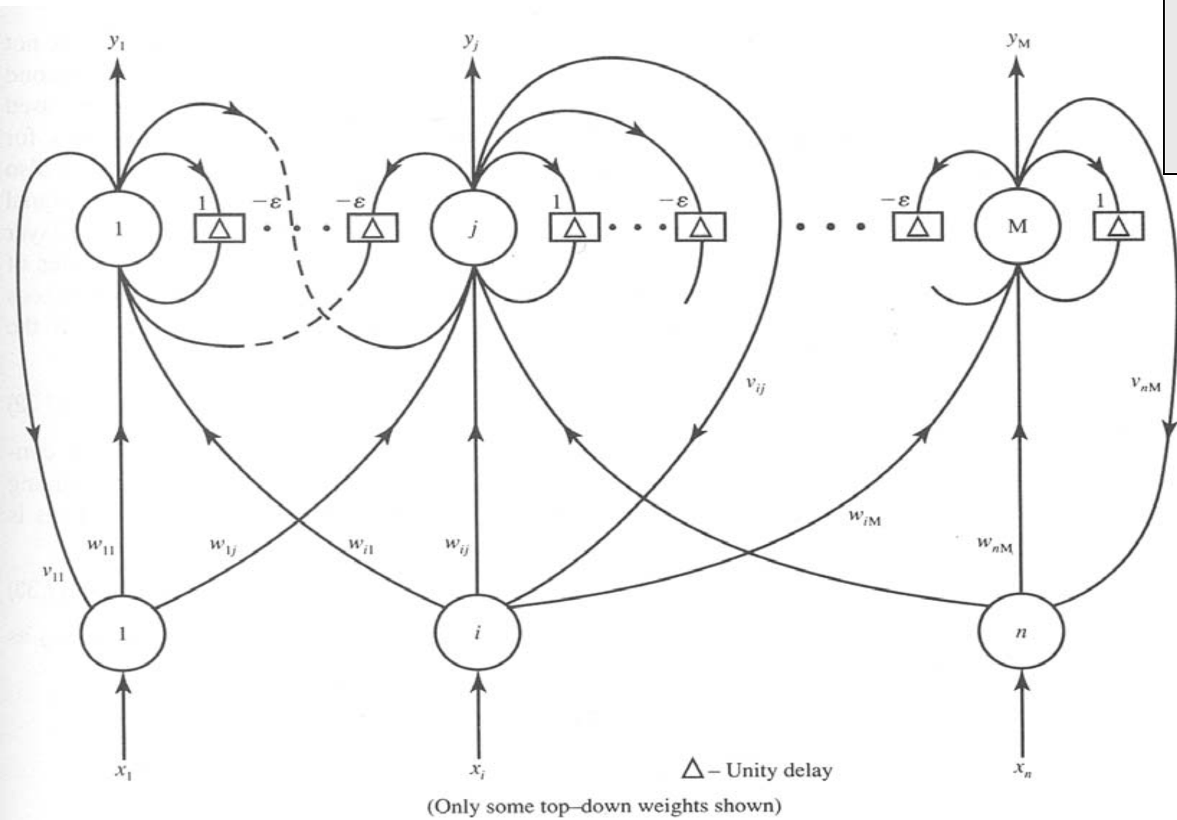


Figure 7.23 Network for discovering clusters (elements computing norms for the vigilance test and elements performing the vigilance test and disabling y_j are not shown).

Initialization :

1.

All elements in $\boldsymbol{W}$ are set to $\frac{1}{1+n}$

All elements in $\boldsymbol{V}$ are set to 1

The vigilance threshold is set to be $0 < \rho < 1$

2.

The first input pattern is arbitrarily assigned to a neuron (cluster), numbered 1, in default

Adaptive Resonance Theory 1 (ART1)

- **Down-to-Up Test**
  - For each input pattern ***x***, select a winner node *j* of the top layer

$$1.\quad y_m^0 = \boldsymbol{w}_m^t \boldsymbol{x},\quad \text{for}\quad m = 1,2,\ldots,M$$

$$\text{or}\quad \boldsymbol{y}^0 = \boldsymbol{W}\boldsymbol{x},\quad \boldsymbol{W} = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{n1} \\ w_{12} & w_{22} & \cdots & w_{n2} \\ & & & \\ w_{1M} & w_{2M} & \cdots & w_{nM} \end{bmatrix}$$

$$2.\quad \boldsymbol{y}^{k+1} = \Gamma\left[\boldsymbol{W}_M \boldsymbol{y}^k\right],\quad \boldsymbol{W}_M = \begin{bmatrix} 1 & -\varepsilon & -\varepsilon \ldots & -\varepsilon \\ -\varepsilon & 1 & -\varepsilon \ldots & -\varepsilon \\ -\varepsilon & \ldots\ldots & & \ldots \\ -\varepsilon & -\varepsilon & -\varepsilon \ldots & 1 \end{bmatrix}$$

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

It's equivalent to the search for winning node of the top layer.

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

$$= \max_{m=1,2,\ldots,M} y_m$$

$$= \max_{m=1,2,\ldots,M} \left(\sum_{i=1}^{n} w_{im} x_i \right)$$

After a number of recurrences, only one single nonzero output of a specific neuron *j* will be produced.

Adaptive Resonance Theory 1 (ART1)

- **Up-to-Down Test**
  - A similarity test for the winner neuron (cluster) *j*

$$\frac{1}{\|x_i\|}\sum_{i=1}^{n} v_{ij}\,x_i > \rho \quad \text{(vigilance threshold)}$$

$$\text{where } \|x_i\| \overset{\Delta}{=} \sum_{i=1}^{n} |x_i|$$

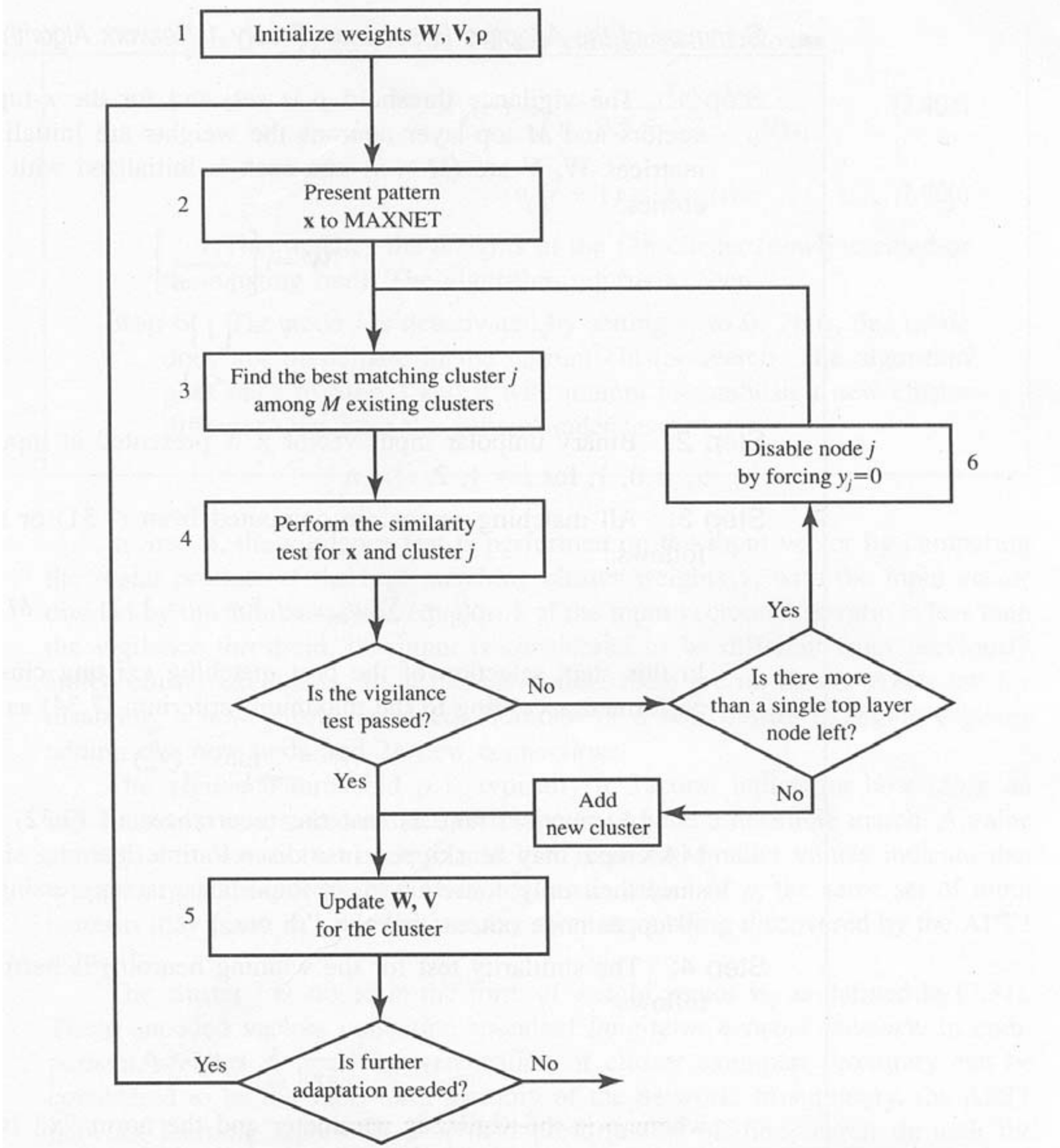
  - If test is passed, the input belong to the winner cluster *j*
    - Update the weights connected to the winner

$$\text{for } i = 1,2,\ldots,M \qquad w_{ij}(t+1) = \begin{cases} \dfrac{v_{ij}(t)x_i}{0.5 + \sum_{i=1}^{n} x_i} & \text{if } x_i = 1 \\ \text{unchanged} & \text{if } x_i = 0 \end{cases}$$

$$\text{for } i = 1,2,\ldots,M \qquad v_{ij}(t+1) = v_{ij}(t)x_i$$

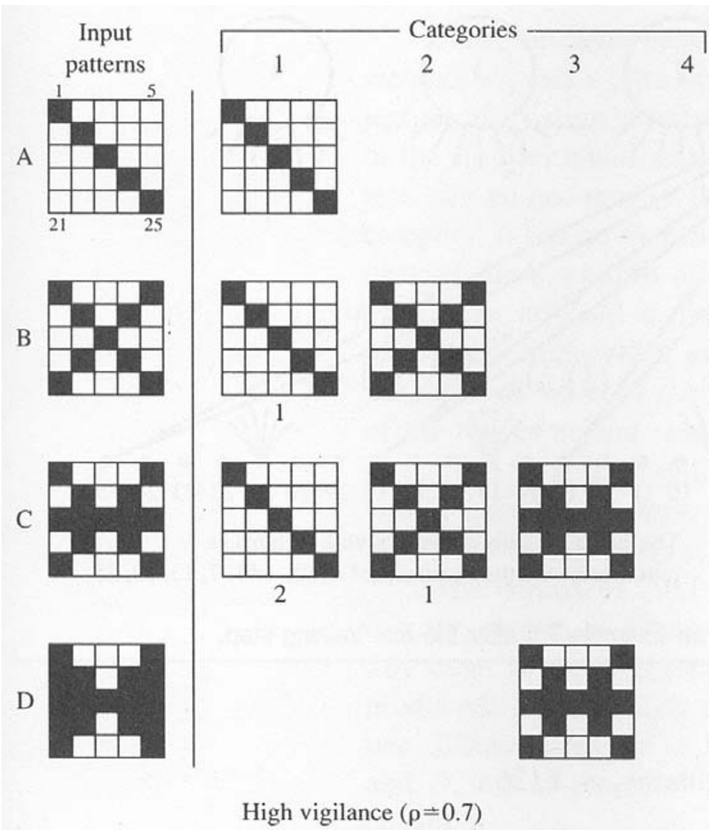
    - If not passed, set y_j to ZERO and select Another cluster has the highest y value and do these two test again
  - If no cluster pass the test, create new one!

Adaptive Resonance Theory 1 (ART1)



Adaptive Resonance Theory 1 (ART1)

- **Example 7.5**



Initial: $w_{ij} = \frac{1}{26},\ v_{ij} = 1$

Input pattern A: the neuron 1 is the default cluster

$$w_{1,1} = w_{7,1} = w_{13,1} = w_{19,1} = w_{25,1} = \frac{1}{0.5+5} = \frac{2}{11}$$

The remaining weights unchanged as initialized, $w_{ij} = \frac{1}{26}$

$$v_{1,1} = v_{7,1} = v_{13,1} = v_{19,1} = v_{25,1} = 1$$

The remaining v_{ij} is set to 0

Input pattern B: Cluster (neuron) 1 the winner

$$\text{Vigilance test} = \frac{1}{\|\boldsymbol{x}\|}\sum_{i=1}^{n} v_{i1}x_i = \frac{1}{9}(5) < 0.7$$

$$w_{i,2} = \begin{cases} \frac{1}{0.5+9} = \frac{2}{19} & \text{if } x_i = 1 \\ \frac{1}{26} & \text{if } x_i = 0 \end{cases} \qquad v_{i,2} = \begin{cases} 1 & \text{if } x_i = 1 \\ 0 & \text{if } x_i = 0 \end{cases}$$

Input pattern C:

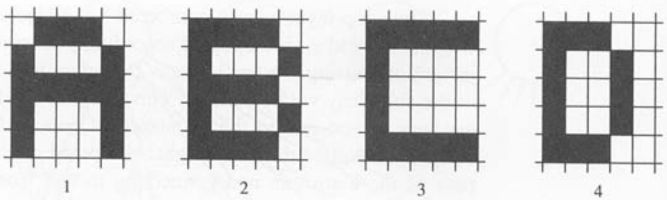
$$y_1^0 = 5\left(\frac{2}{11}\right) + 8\left(\frac{1}{26}\right) = 1.217 \qquad \frac{1}{\|\boldsymbol{x}\|}\sum_{i=1}^{n} v_{i1}x_i = \frac{5}{13} < 0.7$$

$$y_1^0 = 9\left(\frac{2}{11}\right) + 4\left(\frac{1}{26}\right) = 1.101 \qquad \frac{1}{\|\boldsymbol{x}\|}\sum_{i=1}^{n} v_{i2}x_i = \frac{9}{13} < 0.7$$

Cluster 3 Originated!

Adaptive Resonance Theory 1 (ART1)

- **Example 7.6:** ART1 under noisy conditions

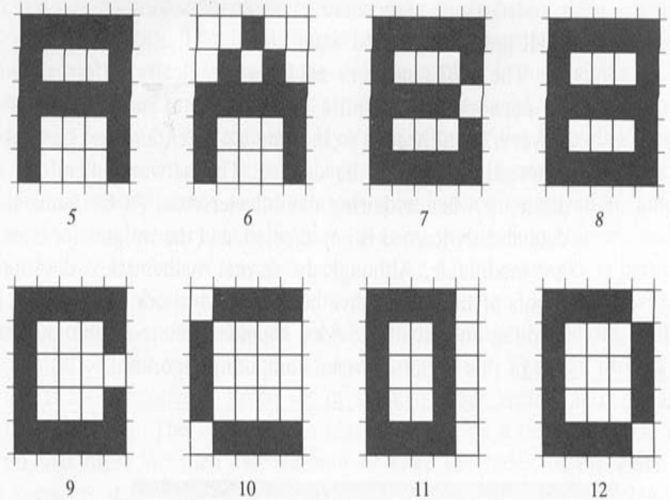


(a)

Input pattern number	Clustering Vigilance level 0.95	0.90	0.85	Desired resonance cluster
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	1	1
6	6	6	5	1
7	7	7	2	2
8	8	7	6	2
9	7	7	3	3
10	3	3	3	3
11	9	8	4	4
12	8	7	4	4

(b)

Figure 7.27a,b Clustering of patterns in Example 7.6: (a) noise-free prototypes, (b) cluster generation by the ART1 network.



(c)

Figure 7.27c Clustering of patterns in Example 7.6 *(continued)*: (c) noisy patterns.

Adaptive Resonance Theory 1 (ART1)

- **Example 7.5**