5.2.2 Mexican Hat Net

In 1989, Kohonen developed the Mexican hat network which is a more generalized contrast enhancement network compared to the earlier Maxnet. There exist several "cooperative neighbors" (neurons in close proximity) to which every other neuron is connected by excitatory links. Also each neuron is connected over inhibitory weights to a number of "competitive neighbors" (neurons present farther away). There are several other farther neurons to which the connections between the neurons are not established. Here, in addition to the connections within a particular layer of neural net, the neurons also receive some other external signals. This interconnection pattern is repeated for several other neurons in the layer.

5.2.2.1 Architecture

The architecture of Mexican hat is shown in Figure 5-2, with the interconnection pattern for node X_i. The neurons here are arranged in linear order; having positive connections between X_i and near neighboring units, and negative connections between X_i and farther away neighboring units. The positive connection region is called region of cooperation and the negative connection region is called region of competition. The size of these regions depends on the relative magnitudes existing between the positive and negative weights and also on the topology of regions such as linear, rectangular, hexagonal grids, etc. In Mexican Hat, there exist two symmetric regions around each individual neuron.

The individual neuron in Figure 5-2 is denoted by X_i. This neuron is surrounded by other neurons X_{i+1}, X_{i-1}, X_{i+2}, X_{i-2}, ... The nearest neighbors to the individual neuron X_i are X_{i+1}, X_{i-1}, X_{i+2}, and X_{i-2}. Hence, the weights associated with these are considered to be positive and are denoted by w_1 and w_2. The farthest neighbors to the individual neuron X_i are taken as X_{i+3} and X_{i-3}, the weights associated with these are negative and are denoted by w_3. It can be seen that X_{i+4} and X_{i-4} are not connected to the individual neuron X_i, and therefore no weighted interconnections exist between these connections. To make it easier, the units present within a radius of 2 [query for unit] to the unit X_i are connected with positive weights, the units within radius 3 are connected with negative weights and the units present farther away from radius 3 are not connected in any manner to the neuron X_i.

5.2.2.2 Flowchart

The flowchart for Mexican hat is shown in Figure 5-3. This clearly depicts the flow of the process performed in Mexican hat network.

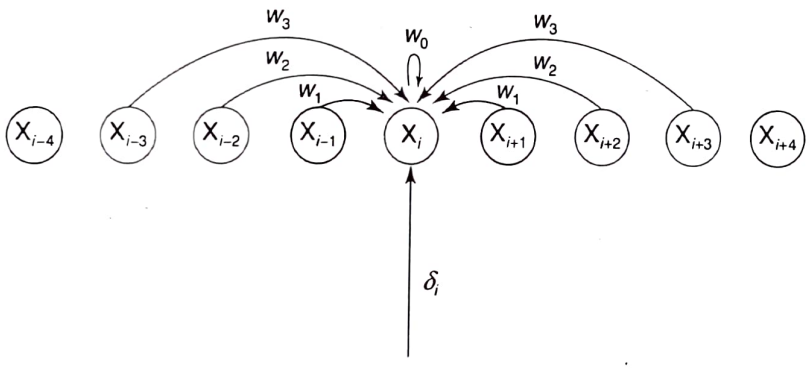

Figure 5-2 Structure of Mexican hat.

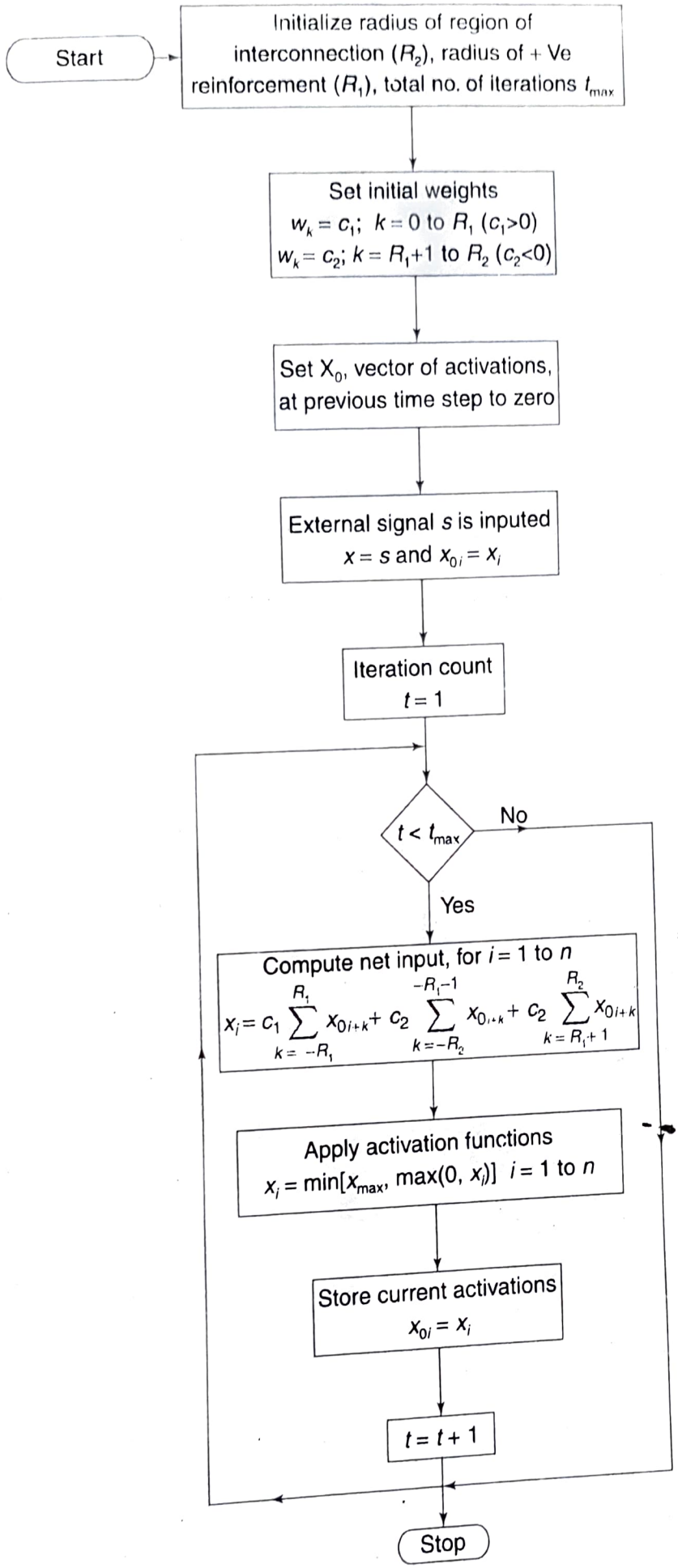

Figure 5-3 Flowchart of Mexican hat.

5.2.2.3 Algorithm

The various parameters used in the training algorithm are as sh wn below.

R_2 = radius of regions of interconnections

X_{i+k} and X_{i-k} are connected to the individual units X_i for $k = 1$ to R_2.

R_1 = radius of region with positive reinforcement ($R_1 < R_2$)

W_k = weight between X_i and the units X_{i+k} and X_{i-k}

$$0 \leqslant k \leqslant R_1, \qquad w_k = \text{positive}$$
$$R_1 \leqslant k \leqslant R_2, \qquad w_k = \text{negative}$$

s = external input signal

x = vector of activation

x_0 = vector of activations at previous time step

$t_{\max}$ = total number of iterations of contrast enhancement.

Here the iteration is started only with the incoming of the external signal presented to the network.

Step 0: The parameters R_1, R_2, $t_{\max}$ are initialized accordingly. Initialize weights as

$$w_k = c_1 \quad \text{for } k = 0, \ldots, R_1 \quad (\text{where } c_1 > 0)$$
$$w_k = c_2 \quad \text{for } k = R_1 + 1, \ldots, R_2 \quad (\text{where } c_2 < 0)$$

Initialize $x_0 = 0$.

Step 1: Input the external signal s:

$$x = s$$

The activations occurring are saved in array x_0. For $i = 1$ to n,

$$x_{0i} = x_i$$

Once activations are stored, set iteration counter $t = 1$.

Step 2: When t is less than $t_{\max}$, perform Steps 3–7.

Step 3: Calculate net input. For $i = 1$ to n,

$$x_i = c_1 \sum_{k=-R_1}^{R_1} x_{0_{i+k}} + c_2 \sum_{k=-R_2}^{-R_1-1} x_{0_{i+k}} + c_2 \sum_{k=R_1+1}^{R_2} x_{0_{i+k}}$$

Step 4: Apply the activation function. For $i = 1$ to n,

$$x_i = \min[x_{\max}, \max(0, x_i)]$$

Step 5: Save the current activations in x_0, i.e., for $i = 1$ to n,

$$x_{0i} = x_i$$

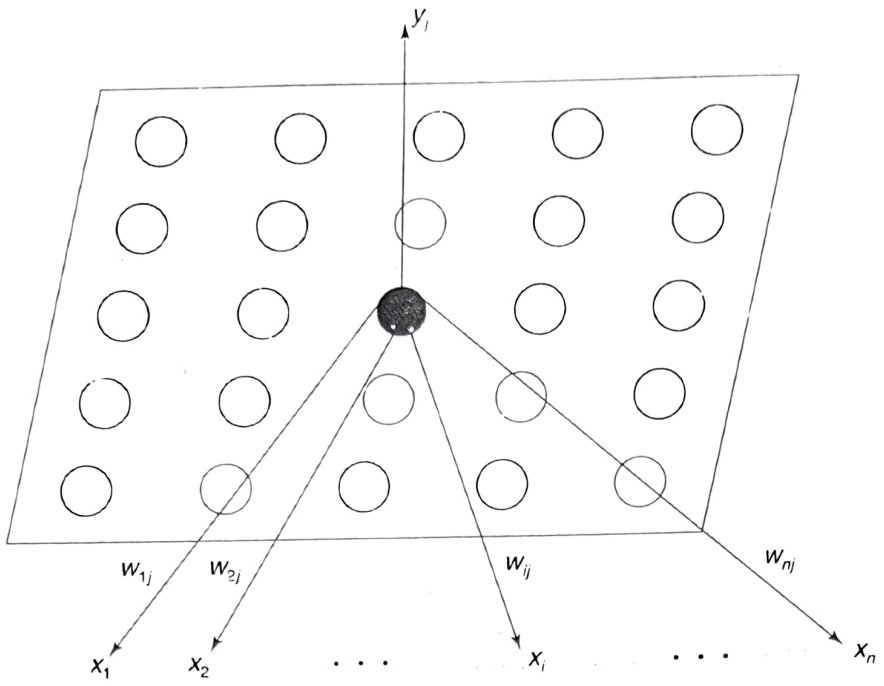

Figure 5-5 One-dimensional feature mapping network.

topology preserving map. For obtaining such feature maps, it is required to find a self-organizing neural array which consists of neurons arranged in a one-dimensional array or a two-dimensional array. To depict this, a typical network structure where each component of the input vector x is connected to each of the nodes is shown in Figure 5-5.

On the other hand, if the input vector is two-dimensional, the inputs, say $x(a, b)$, can arrange themselves in a two-dimensional array defining the input space (a, b) as in Figure 5-6. Here, the two layers are fully connected.

The topological preserving property is observed in the brain, but not found in any other artificial neural network. Here, there are m output cluster units arranged in a one- or two-dimensional array and the input signals are n-tuples. The cluster (output) units' weight vector serves as an exemplar of the input pattern that is associated with that cluster. At the time of self-organization, the weight vector of the cluster unit which matches the input pattern very closely is chosen as the winner unit. The closeness of weight vector of cluster unit to the input pattern may be based on the square of the minimum Euclidean distance. The weights are updated for the winning unit and its neighboring units. It should be noted that the weight vectors of the neighboring units are not close to the input pattern and the connective weights do not multiply the signal sent from the input units to the cluster units until dot product measure of similarity is being used.

5.3.2 Architecture

Consider a linear array of cluster units as in Figure 5-7. The neighborhoods of the units designated by "o" of radii $N_i(k_1)$, $N_i(k_2)$ and $N_i(k_3)$, $k_1 > k_2 > k_3$, where $k_1 = 2, k_2 = 1, k_3 = 0$.

For a rectangular grid, a neighborhood (N_i) of radii k_1, k_2 and k_3 is shown in Figure 5-8 and for a hexagonal grid the neighborhood is shown in Figure 5-9. In all the three cases (Figures 5-7–5-9), the unit with

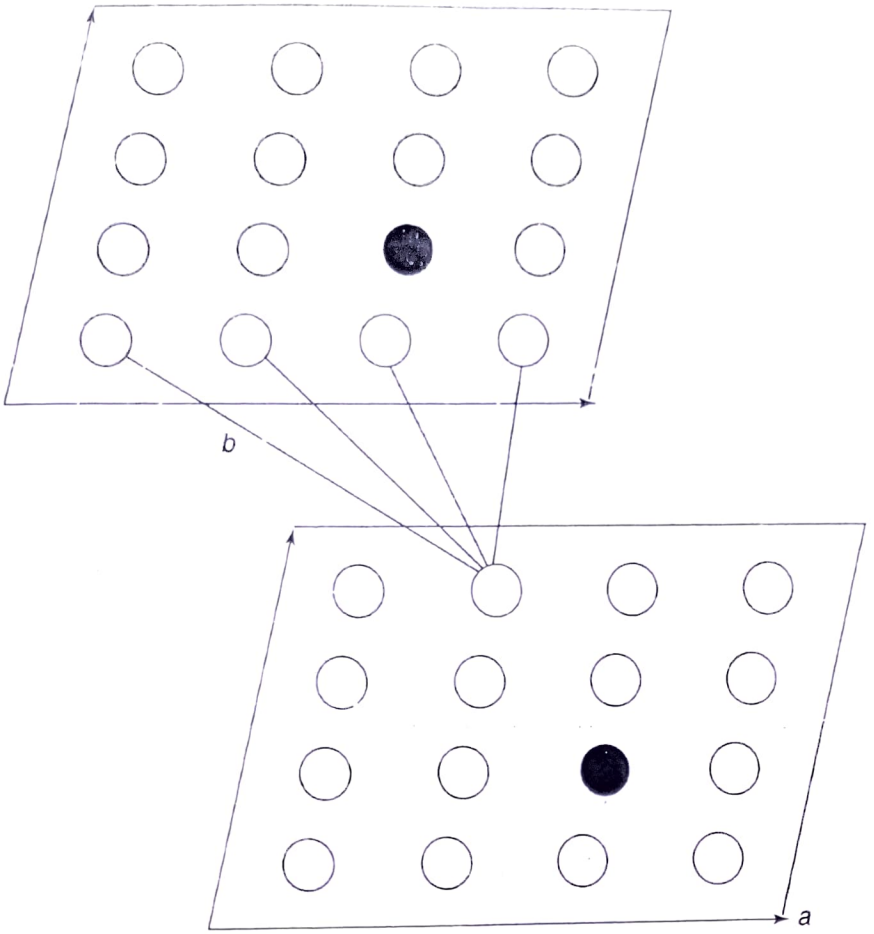

Figure 5-6 Two-dimensional feature mapping network.

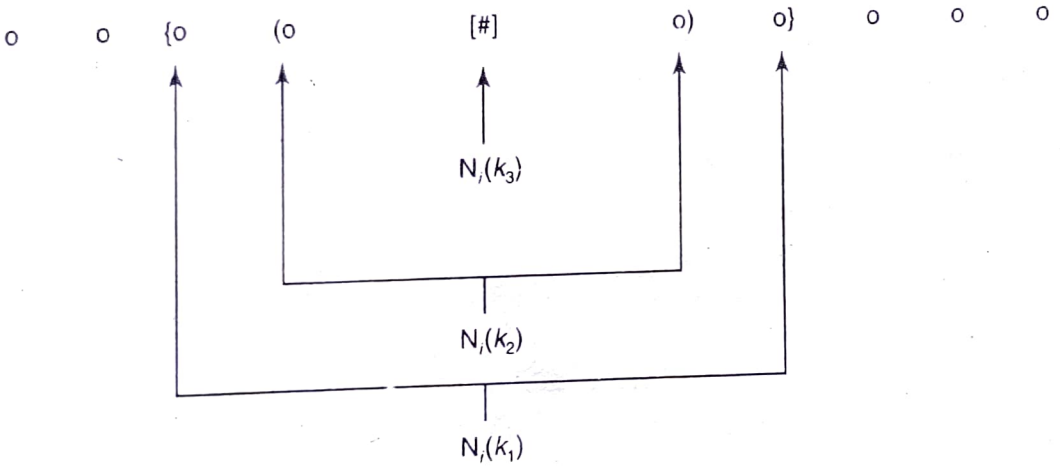

Figure 5-7 Linear array of cluster units.

"#" symbol is the winning unit and the other units are indicated by "o." In both rectangular and hexagonal grids, $k_1 > k_2 > k_3$, where $k_1 = 2, k_2 = 1, k_3 = 0$.

For rectangular grid, each unit has eight nearest neighbors but there are only six neighbors for each unit in the case of a hexagonal grid. Missing neighborhoods may just be ignored. A typical architecture of Kohonen self-organizing feature map (KSOFM) is shown in Figure 5-10.

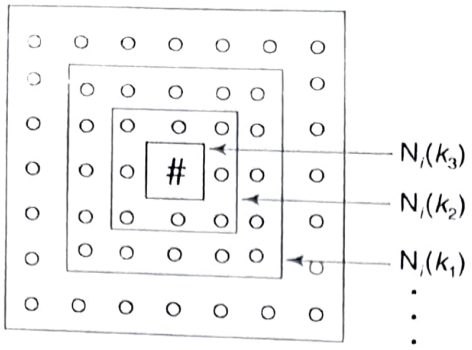

Figure 5-8 Rectangular grid.

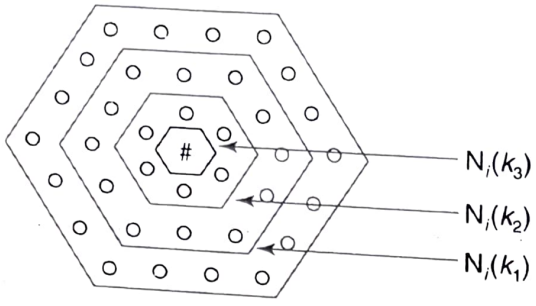

Figure 5-9 Hexagonal grid.

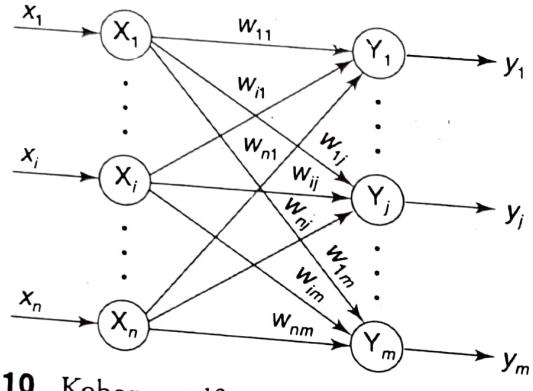

Figure 5-10 Kohonen self-organizing feature map architecture.

5.3.3 Flowchart

The flowchart for KSOFM is shown in Figure 5-11, which indicates the flow of training process. The process is continued for particular number of epochs or till the learning rate reduces to a very small rate.

The architecture consists of two layers: input layer and output layer (cluster). There are "n" units in the input layer and "m" units in the output layer. Basically, here the winner unit is identified by using either dot product or Euclidean distance method and the weight updation using Kohonen learning rules is performed over the winning cluster unit.

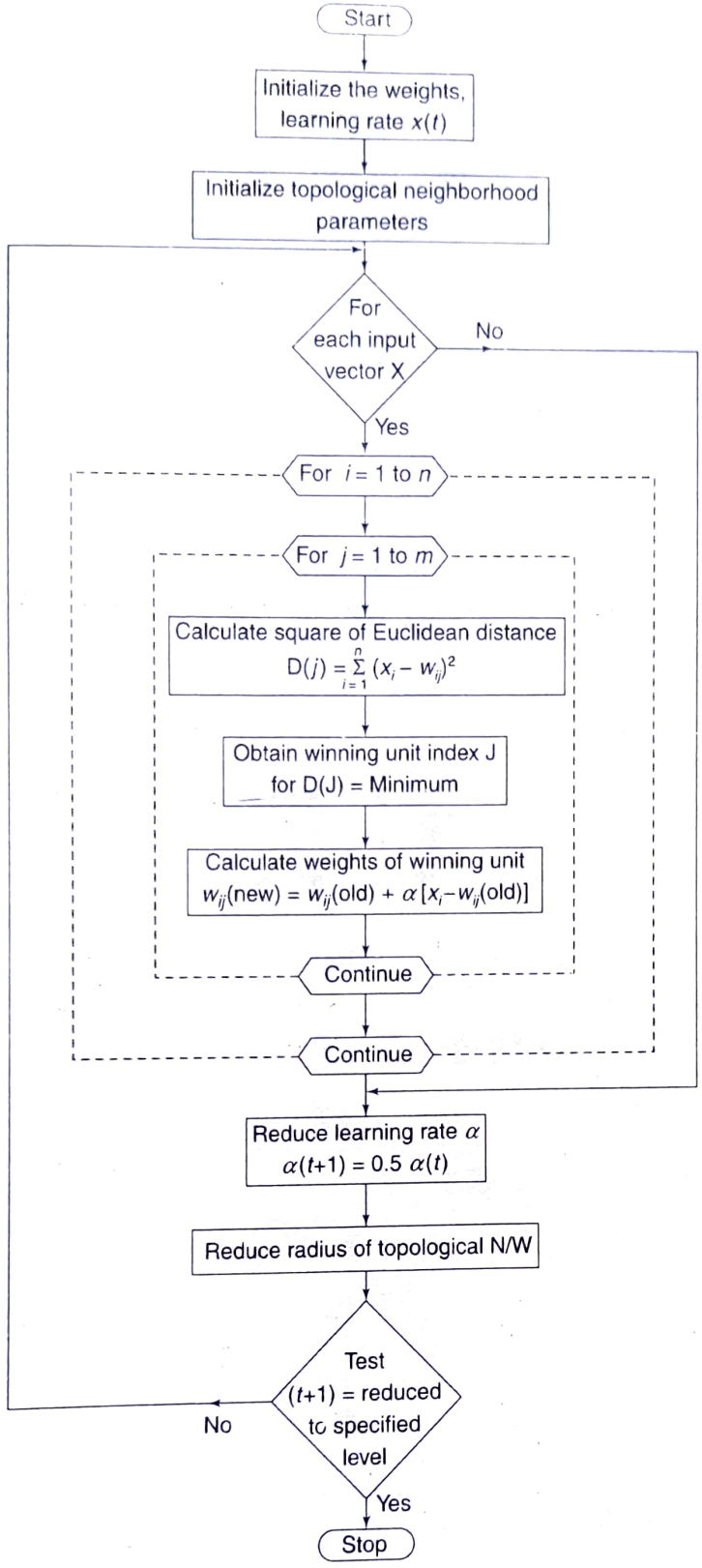

Figure 5-11 Flowchart for training process of KSOFM.

5.3.4 Training Algorithm

The steps involved in the training algorithm are as shown below.

Step 0:
- Initialize the weights w_{ij}: Random values may be assumed. They can be chosen as the same range of values as the components of the input vector. If information related to distribution of clusters is known, the initial weights can be taken to reflect that prior knowledge.
- Set topological neighborhood parameters: As clustering progresses, the radius of the neighborhood decreases.
- Initialize the learning rate α: It should be a slowly decreasing function of time.

Step 1: Perform Steps 2–8 when stopping condition is false.

Step 2: Perform Steps 3–5 for each input vector x.

Step 3: Compute the square of the Euclidean distance, i.e., for each $j = 1$ to m,

$$D(j) = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_i - w_{ij})^2$$

Step 4: Find the winning unit index J, so that D(J) is minimum. (In Steps 3 and 4, dot product method can also be used to find the winner, which is basically the calculation of net input, and the winner will be the one with the largest dot product.)

Step 5: For all units j within a specific neighborhood of J and for all i, calculate the new weights:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha\,[x_i - w_{ij}(\text{old})]$$

or

$$w_{ij}(\text{new}) = (1-\alpha)\,w_{ij}(\text{old}) + \alpha\, x_i$$

Step 6: Update the learning rate α using the formula $\alpha\,(t+1) = 0.5\alpha\,(t)$.

Step 7: Reduce radius of topological neighborhood at specified time intervals.

Step 8: Test for stopping condition of the network.

Thus using this training algorithm, an efficient training can be performed for an unsupervised learning network.

5.4 Learning Vector Quantization

5.4.1 Theory

Learning vector quantization (LVQ) is a process of classifying the patterns, wherein each output unit represents a particular class. Here, for each class several units should be used. The output unit weight vector is called the reference vector or code book vector for the class which the unit represents. This is a special case of competitive net, which uses supervised learning methodology. During training, the output units are found to be positioned to approximate the decision surfaces of the existing Bayesian classifier. Here, the set of training patterns with known classifications is given to the network, along with an initial distribution of the reference vectors. When the training process is complete, an LVQ net is found to classify an input vector by assigning it to the same class as that of the output unit, which has its weight vector very close to the input vector. Thus, LVQ is a classifier paradigm that adjusts the boundaries between categories to minimize existing misclassification. LVQ is used for optical character recognition, converting speech into phonemes and other applications as well. LVQ net may resemble KSOFM net. Unlike LVQ, KSOFM output nodes do not correspond to the known classes but rather correspond to unknown clusters that the KSOFM finds in the data autonomously.

5.4.2 Architecture

Figure 5-13 shows the architecture of LVQ, which is almost the same as that of KSOFM, with the difference being that in the case of LVQ, the topological structure at the output unit is not being considered. Here, each output unit has knowledge about what a known class represents.

From Figure 5-13 it can be noticed that there exist input layer with "n" units and output layer with "m" units. The layers are found to be fully interconnected with weighted linkage acting over the links.

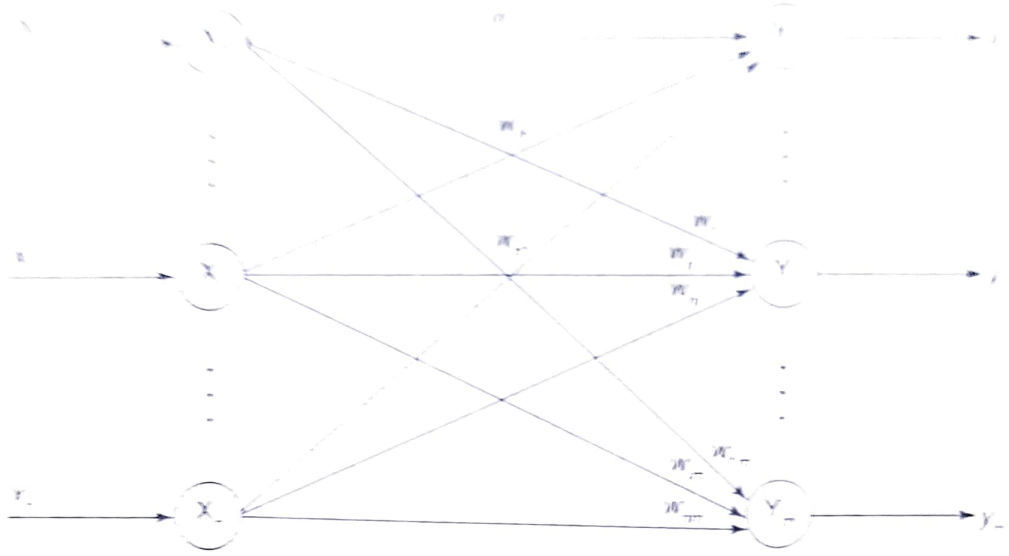

Figure 5-13 Architecture of LVQ.

5.4.3 Flowchart

The parameters used for the training process of a LVQ include the following:

x = training vector $(x_1, \ldots, x_i, \ldots, x_n)$

T = category or class for the training vector x

w_j = weight vector for jth output unit $(w_{1j}, \ldots, w_{ij}, \ldots, w_{nj})$

c_j = cluster or class or category associated with jth output unit.

The Euclidean distance of jth output unit is $D(j) = \sum (x_i - w_{ij})^2$. The flowchart indicating the flow of training process is shown in Figure 5-14.

5.4.4 Training Algorithm

In case of training, a set of training input vectors with a known classification is provided with some initial distribution of reference vector. Here, each output unit will have a known class. The objective of the algorithm is to find the output unit that is closest to the input vector.

Step 0: Initialize the reference vectors. This can be done using the following steps.

- From the given set of training vectors, take the first "m" (number of clusters) training vectors and use them as weight vectors, the remaining vectors can be used for training.
- Assign the initial weights and classifications randomly.
- K-means clustering method.

Set initial learning rate α.

Step 1: Perform Steps 2–6 if the stopping condition is false.

Step 2: Perform Steps 3–4 for each training input vector x.

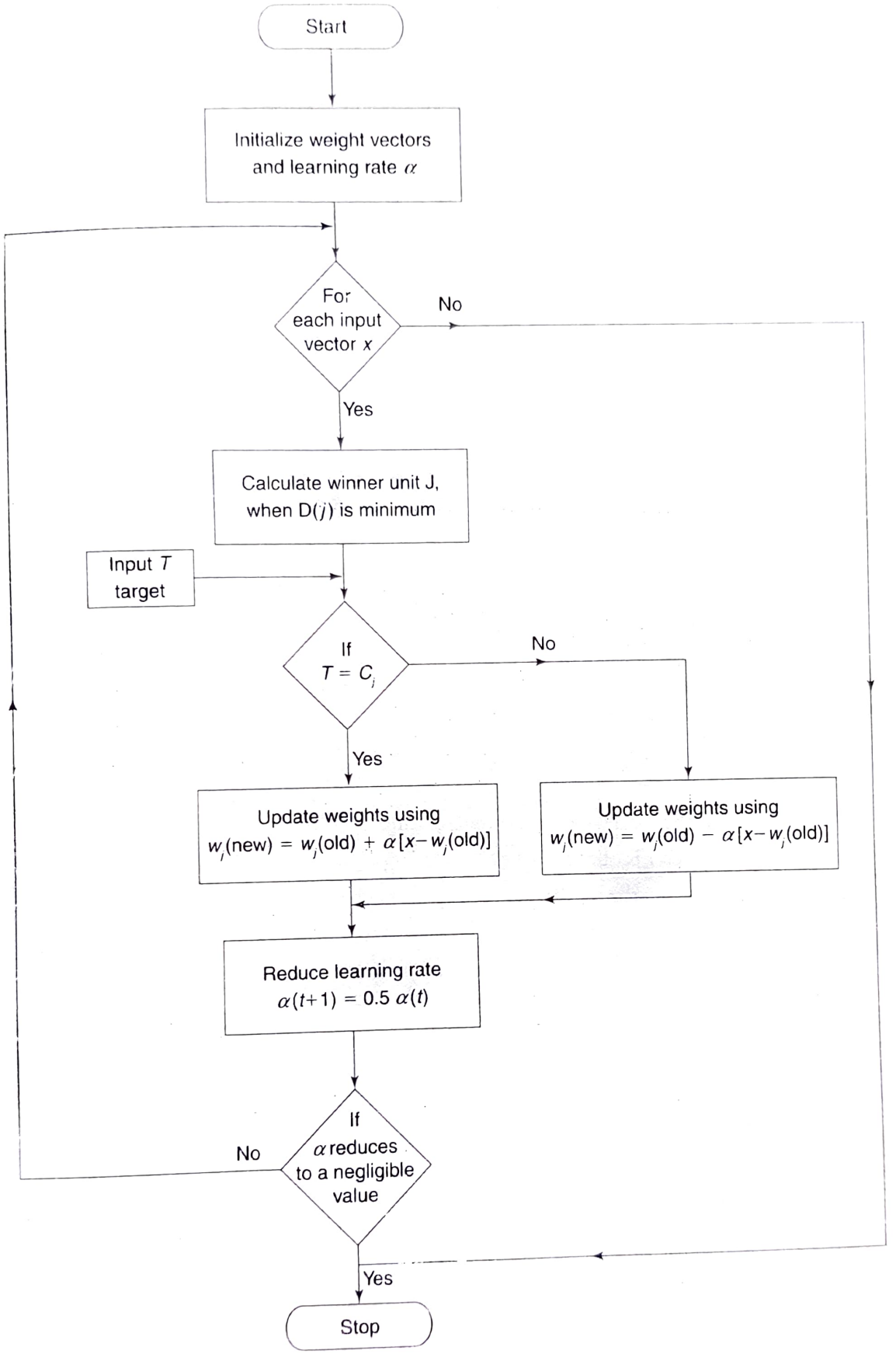

Figure 5-14 Flowchart for LVQ.

Step 3: Calculate the Euclidean distance; for $i = 1$ to n, $j = 1$ to m,

$$D(j) = \sum_{i=1}^{n} \sum_{j=1}^{m} (x_i - w_{ij})^2$$

Find the winning unit index J, when D(J) is minimum.

Step 4: Update the weights on the winning unit, w_J using the following conditions.

$$\text{If } T = c_J, \text{then } w_J(\text{new}) = w_J(\text{old}) + \alpha [x - w_J(\text{old})]$$
$$\text{If } T \neq c_J, \text{then } w_J(\text{new}) = w_J(\text{old}) - \alpha [x - w_J(\text{old})]$$

Step 5: Reduce the learning rate α.

Step 6: Test for the stopping condition of the training process. (The stopping conditions may be fixe number of epochs or if learning rate has reduced to a negligible value.)

3. Construct a Kohonen self-organizing map to cluster the four given vectors, [0 0 1 1], [1 0 0 0], [0 1 1 0] and [0 0 0 1]. The number of clusters to

be formed is two. Assume an initial learning rate of 0.5.

Solution: The number of input vectors is four and number of clusters to be formed is two. Thus, $n = 4$ and $m = 2$. The architecture of the Kohonen self-organizing feature map is given by Figure 2.

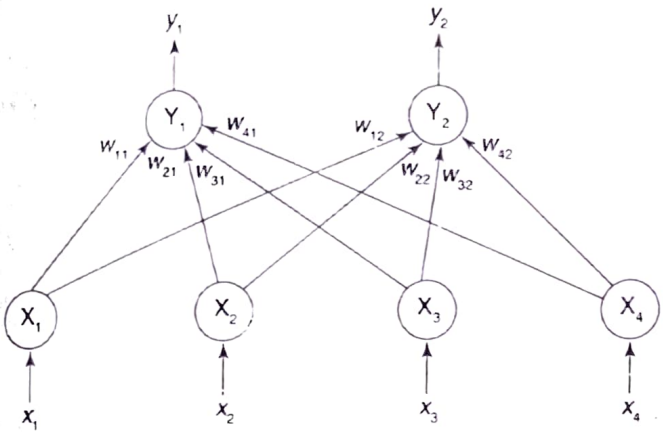

Figure 2 Architecture of KSOFM.

Step 0: Initialize the weights randomly between 0 and 1.

$$w_{ij} = \begin{bmatrix} 0.2 & 0.9 \\ 0.4 & 0.7 \\ 0.6 & 0.5 \\ 0.8 & 0.3 \end{bmatrix}_{4\times 2} \quad ; \quad R = 0;\ \alpha(0) = 0.5$$

First input vector:

Step 1: For $x = [0\ 0\ 1\ 1]$, perform Steps 2–4.

Step 2: Calculate the Euclidean distance:

$$D(j) = \sum_i (w_{ij} - x_i)^2$$

$$D(1) = \sum_{i=1}^{4} (w_{i1} - x_i)^2$$

$$= (0.2 - 0)^2 + (0.4 - 0)^2 + (0.6 - 1)^2 + (0.8 - 1)^2$$

$$= 0.04 + 0.16 + 0.16 + 0.04$$

$$= 0.4$$

$$D(2) = \sum_{i=1}^{4} (w_{i2} - x_i)^2$$

$$= (0.9 - 0)^2 + (0.7 - 0)^2 + (0.5 - 1)^2 + (0.3 - 1)^2$$

$$= 0.81 + 0.49 + 0.25 + 0.49$$

$$= 2.04$$

Step 3: Since $D(1) < D(2)$, therefore $D(1)$ is minimum. Hence the winning cluster unit is Y_1, i.e., $J = 1$.

Step 4: Update the weights on the winning cluster unit $J = 1$.

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$

$$w_{i1}(\text{new}) = w_{i1}(\text{old}) + 0.5\,[x_i - w_{i1}(\text{old})]$$

$$w_{11}(n) = w_{11}(0) + 0.5\,[x_1 - w_{11}(0)]$$

$$= 0.2 + 0.5(0 - 0.2) = 0.1$$

$$w_{21}(n) = w_{21}(0) + 0.5\,[x_2 - w_{21}(0)]$$

$$= 0.4 + 0.5(0 - 0.4) = 0.2$$

$$w_{31}(n) = w_{31}(0) + 0.5\,[x_3 - w_{31}(0)]$$

$$= 0.6 + 0.5(1 - 0.6) = 0.8$$

$$w_{41}(n) = w_{41}(0) + 0.5\,[x_4 - w_{41}(0)]$$

$$= 0.8 + 0.5(1 - 0.8) = 0.9$$

The updated weight matrix after presentation of first input pattern is

$$w_{ij} = \begin{bmatrix} 0.1 & 0.9 \\ 0.2 & 0.7 \\ 0.8 & 0.5 \\ 0.9 & 0.3 \end{bmatrix}$$

Second input vector:

Step 1: For $x = [1\ 0\ 0\ 0]$, perform Steps 2–4.

Step 2: Calculate the Euclidean distance:

$$D(j) = \sum_i (w_{ij} - x_i)^2$$

$$D(1) = \sum_{i=1}^{4} (w_{i1} - x_i)^2$$

$$= (0.1 - 1)^2 + (0.2 - 0)^2 + (0.8 - 0)^2 + (0.9 - 0)^2$$
$$= 0.81 + 0.04 + 0.64 + 0.81$$
$$= 2.3$$

$$D(2) = \sum_{i=1}^{4} (w_{i2} - x_i)^2$$
$$= (0.9 - 1)^2 + (0.7 - 0)^2 + (0.5 - 0)^2 + (0.3 - 0)^2$$
$$= 0.01 + 0.49 + 0.25 + 0.09$$
$$= 0.84$$

Step 3: Since $D(2) < D(1)$, therefore $D(2)$ is minimum. Hence the winning cluster unit is Y_2, i.e., $J = 2$.

Step 4: Update the weights on the winning cluster unit $J = 2$:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$
$$w_{i2}(\text{new}) = w_{i2}(\text{old}) + 0.5\,[x_i - w_{i2}(\text{old})]$$
$$w_{12}(n) = w_{12}(0) + 0.5\,[x_1 - w_{12}(0)]$$
$$= 0.9 + 0.5(1 - 0.9) = 0.95$$
$$w_{22}(n) = w_{22}(0) + 0.5\,[x_2 - w_{22}(0)]$$
$$= 0.7 + 0.5(0 - 0.7) = 0.35$$
$$w_{32}(n) = w_{32}(0) + 0.5\,[x_3 - w_{32}(0)]$$
$$= 0.5 + 0.5(0 - 0.5) = 0.25$$
$$w_{42}(n) = w_{42}(0) + 0.5\,[x_4 - w_{42}(0)]$$
$$= 0.3 + 0.5(0 - 0.3) = 0.15$$

The updated weight matrix after presentation of second input pattern is

$$w_{ij} = \begin{bmatrix} 0.1 & 0.95 \\ 0.2 & 0.35 \\ 0.8 & 0.25 \\ 0.9 & 0.15 \end{bmatrix}$$

Third input vector:

Step 1: For $x = [0\ 1\ 1\ 0]$, perform Steps 2–4.

Step 2: Calculate the Euclidean distance:

$$D(j) = \sum_i (w_{ij} - x_i)^2$$
$$D(1) = \sum_{i=1}^{4} (w_{i1} - x_i)^2$$
$$= (0.1 - 0)^2 + (0.2 - 1)^2 + (0.8 - 1)^2 + (0.9 - 0)^2$$
$$= 0.01 + 0.64 + 0.04 + 0.81 = 1.5$$
$$D(2) = \sum_{i=1}^{4} (w_{i2} - x_i)^2$$
$$= (0.95 - 0)^2 + (0.35 - 1)^2 + (0.25 - 1)^2 + (0.15 - 0)^2$$
$$= (0.9025) + (0.4225) + (0.5625) + (0.0225) = 1.91$$

Step 3: Since $D(1) < D(2)$, therefore $D(1)$ is minimum. Hence the winning cluster unit is Y_1, i.e., $J = 1$.

Step 4: Update the weights on the winning cluster unit $J = 1$:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$
$$w_{i1}(\text{new}) = w_{i1}(\text{old}) + 0.5\,[x_i - w_{i1}(\text{old})]$$
$$w_{11}(n) = w_{11}(0) + 0.5\,[x_1 - w_{11}(0)]$$
$$= 0.1 + 0.5(0 - 0.1) = 0.05$$
$$w_{21}(n) = w_{21}(0) + 0.5\,[x_2 - w_{21}(0)]$$
$$= 0.2 + 0.5(1 - 0.2) = 0.6$$
$$w_{31}(n) = w_{31}(0) + 0.5\,[x_3 - w_{31}(0)]$$
$$= 0.8 + 0.5(1 - 0.8) = 0.9$$
$$w_{41}(n) = w_{41}(0) + 0.5\,[x_4 - w_{41}(0)]$$
$$= 0.9 + 0.5(0 - 0.9) = 0.45$$

The weight update after presentation of third input pattern is

$$w_{ij} = \begin{bmatrix} 0.05 & 0.95 \\ 0.6 & 0.35 \\ 0.9 & 0.25 \\ 0.45 & 0.15 \end{bmatrix}$$

Fourth input vector:

Step 1: For $x = [0\ 0\ 0\ 1]$, perform Steps 2–4.

Step 2: Compute the Euclidean distance:

$$D(j) = \sum_{i=1}^{4} (w_{ij} - x_i)^2$$

$$D(1) = \sum_{i=1}^{4} (w_{i1} - x_i)^2$$

$$= (0.05 - 0)^2 + (0.6 - 0)^2 + (0.9 - 0)^2 + (0.45 - 1)^2$$

$$= 0.0025 + 0.36 + 0.81 + 0.3025$$

$$= 1.475$$

$$D(2) = \sum_{i=1}^{4} (w_{i2} - x_i)^2$$

$$= (0.95 - 0)^2 + (0.35 - 0)^2 + (0.25 - 0)^2 + (0.15 - 1)^2$$

$$= (0.9025) + (0.1225) + (0.0625) + (0.7225)$$

$$= 1.81$$

Step 3: Since $D(1) < D(2)$, therefore $D(1)$ is minimum. Hence the winning cluster unit is Y_1, i.e., $J = 1$.

Step 4: Update the weights on the winning cluster unit $J = 1$:

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$

$$w_{i1}(\text{new}) = w_{i1}(\text{old}) + 0.5\,[x_i - w_{i1}(\text{old})]$$

$$w_{11}(n) = w_{11}(0) + 0.5\,[x_1 - w_{11}(0)]$$

$$= 0.05 + 0.5(0 - 0.05) = 0.025$$

$$w_{21}(n) = w_{21}(0) + 0.5\,[x_2 - w_{21}(0)]$$

$$= 0.6 + 0.5(0 - 0.6) = 0.3$$

$$w_{31}(n) = w_{31}(0) + 0.5\,[x_3 - w_{31}(0)]$$

$$= 0.9 + 0.5(0 - 0.9) = 0.45$$

$$w_{41}(n) = w_{41}(0) + 0.5\,[x_4 - w_{41}(0)]$$

$$= 0.45 + 0.5(1 - 0.95) = 0.475$$

The final weight obtained after the presentation of fourth input pattern is

$$w_{ij} = \begin{bmatrix} 0.025 & 0.95 \\ 0.3 & 0.35 \\ 0.45 & 0.25 \\ 0.475 & 0.15 \end{bmatrix}$$

Since all the four given input patterns are presented, this is end of first iteration or 1-epoch. Now the learning rate can be updated as

$$\alpha(t+1) = 0.5\,\alpha(t)$$

$$\alpha(1) = 0.5\,\alpha(0) = 0.5 \times 0.5 = 0.25$$

With this learning rate you can proceed further up to 100 iterations or till radius becomes zero or the weight matrix reduces to a very negligible value. The net with updated weights is shown by Figure 3.

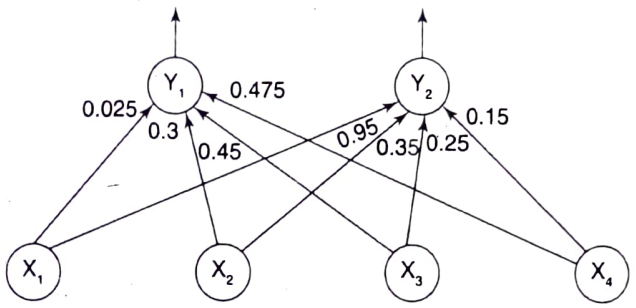

Figure 3 Net for problem 3.

4. For a given Kohonen self-organizing feature map with weights shown in Figure 4: (a) Use the square of the Euclidean distance to find the cluster unit Y_J closest to the input vector (0.2, 0.4). Using a learning rate of 0.2, find the new weights for unit Y_J. (b) For the input vector (0.6, 0.6) with learning rate 0.1, find the winning cluster unit and its new weights.

Solution: (a) For the input vector $(0.2, 0.4) = (x_1, x_2)$ and $\alpha = 0.2$, the weight vector W is given by

$$W = \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

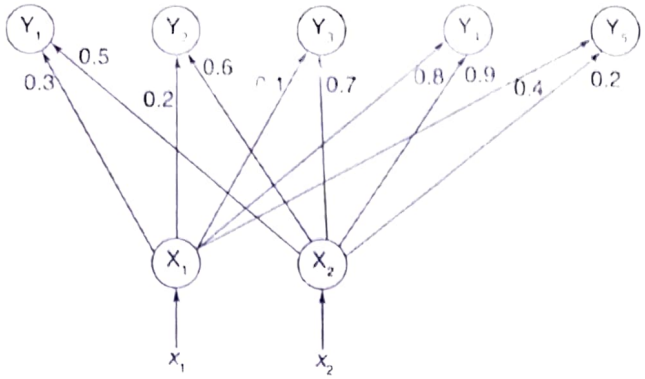

Figure 4 KSOFM net for problem 4.

Now we find the winner unit using square of Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^{2} (w_{ij} - x_i)^2 = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2$$

For $j = 1$ to 5

$$\begin{aligned} D(1) &= (0.3 - 0.2)^2 + (0.5 - 0.4)^2 \\ &= 0.01 + 0.01 = 0.02 \\ D(2) &= (0.2 - 0.2)^2 + (0.6 - 0.4)^2 \\ &= 0 + 0.4 = 0.04 \\ D(3) &= (0.1 - 0.2)^2 + (0.7 - 0.4)^2 \\ &= 0.01 + 0.09 = 0.01 \\ D(4) &= (0.8 - 0.2)^2 + (0.9 - 0.4)^2 \\ &= 0.36 + 0.25 = 0.61 \\ D(5) &= (0.4 - 0.2)^2 + (0.2 - 0.4)^2 \\ &= 0.04 + 0.04 = 0.08 \end{aligned}$$

Since $D(1) = 0.2$ is the minimum value, the winner unit is $J = 1$. We now update the weights on the winner unit $J = 1$. The weight updation formula is given by

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$

Substituting $J = 1$ in the equation above, we obtain

$$w_{i1}(\text{new}) = w_{i1}(\text{old}) + \alpha[x_i - w_{i1}(\text{old})]$$

For $i = 1$ to 2,

$$\begin{aligned} w_{11}(n) &= w_{11}(0) + \alpha[x_1 - w_{11}(0)] \\ &= 0.3 + 0.2(0.2 - 0.3) = 0.28 \\ w_{21}(n) &= w_{21}(0) + \alpha[x_2 - w_{21}(0)] \\ &= 0.5 + 0.2(0.4 - 0.5) = 0.48 \end{aligned}$$

The updated weight matrix is given by

$$W = \begin{bmatrix} 0.28 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.48 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

- For the input vector $(x_1, x_2) = (0.6, 0.6)$ and $\alpha = 0.1$, the weight matrix is initialized from Figure 4 as

$$W = \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

Now we find the winner unit using square of Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^{2} (w_{ij} - x_i)^2 = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2$$

For $j = 1$ to 5

$$\begin{aligned} D(1) &= (0.3 - 0.6)^2 + (0.5 - 0.6)^2 \\ &= 0.09 + 0.01 = 0.1 \\ D(2) &= (0.2 - 0.6)^2 + (0.6 - 0.6)^2 \\ &= 0.08 + 0 = 0.08 \\ D(3) &= (0.1 - 0.6)^2 + (0.7 - 0.6)^2 \\ &= 0.25 + 0.01 = 0.26 \\ D(4) &= (0.8 - 0.6)^2 + (0.9 - 0.6)^2 \\ &= 0.04 + 0.09 = 0.13 \\ D(5) &= (0.4 - 0.6)^2 + (0.2 - 0.6)^2 \\ &= 0.04 + 0.16 = 0.2 \end{aligned}$$

Since $D(2) = 0.08$ is the minimum value, the winner unit is $J = 2$. We now update the weights on the winner unit with $\alpha = 0.1$. The weight updation formula is given by

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$

Substituting $J = 2$ in the equation above, we obtain

$$w_{i2}(\text{new}) = w_{i2}(\text{old}) + \alpha[x_i - w_{i2}(\text{old})]$$

For $i = 1$ to 2,

$$w_{12}(n) = w_{12}(0) + \alpha[x_1 - w_{12}(0)]$$
$$= 0.2 + 0.1(0.6 - 0.2) = 0.24$$
$$w_{22}(n) = w_{22}(0) + \alpha[x_2 - w_{22}(0)]$$
$$= 0.6 + 0.1(0.6 - 0.6) = 0.6$$

The new weight matrix is given by

$$W = \begin{bmatrix} 0.3 & 0.24 & 0.1 & 0.8 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.9 & 0.2 \end{bmatrix}$$

5. Consider a Kohonen self-organizing net with two cluster units and five input units. The weight vectors for the cluster units are given by

$$w_1 = [1.0\ 0.9\ 0.7\ 0.5\ 0.3]$$
$$w_2 = [0.3\ 0.5\ 0.7\ 0.9\ 1.0]$$

Use the square of the Euclidean distance to find the winning cluster unit for the input pattern $x = [0.0\ 0.5\ 1.0\ 0.5\ 0.0]$. Using a learning rate of 0.25, find the new weights for the winning unit.

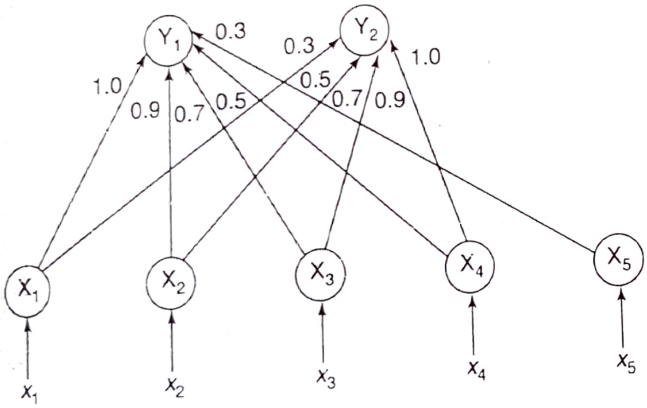

Figure 5 KSOFM net.

Solution: The net can be formed as shown in Figure 5. For the input vector $x = [0.0\ 0.5\ 1.0\ 0.5\ 0.0]$ and the learning rate $\alpha = 0.25$, the weight vector W is given by

$$W = \begin{bmatrix} 1.0 & 0.3 \\ 0.9 & 0.5 \\ 0.7 & 0.7 \\ 0.5 & 0.9 \\ 0.3 & 1.0 \end{bmatrix}$$

Now we find the winner unit using square of Euclidean distance, i.e.,

$$D(j) = \sum_i (w_{ij} - x_i)^2$$

For $i = 1$ to 5 and $j = 1$ to 2,

$$D(1) = (1 - 0)^2 + (0.9 - 0.5)^2 + (0.7 - 1)^2 + (0.5 - 0.5)^2 + (0.3 - 0)^2$$
$$= 1 + 0.16 + 0.09 + 0 + 0.09 = 1.34$$
$$D(2) = (0.3 - 0)^2 + (0.5 - 0.5)^2 + (0.7 - 1)^2 + (0.9 - 0.5)^2 + (1 - 0)^2$$
$$= 0.09 + 0 + 0.09 + 0.16 + 1 = 1.34$$

As we can see, in this case $D(1) = D(2)$, so the winner unit is the one with the smallest index. Thus, winner unit is Y_1, i.e., $J = 1$. We now update the weights on the winner unit with $\alpha = 0.25$. The weight updation formula is given by

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha[x_i - w_{ij}(\text{old})]$$

Substituting $J = 1$ in the equation above, we obtain

$$w_{i1}(\text{new}) = w_{i1}(\text{old}) + \alpha[x_i - w_{i1}(\text{old})]$$

For $i = 1$ to 5,

$$w_{11}(n) = w_{11}(0) + \alpha\,[x_1 - w_{11}(0)]$$
$$= 1 + 0.25(0 - 1) = 0.75$$
$$w_{21}(n) = w_{21}(0) + \alpha\,[x_2 - w_{21}(0)]$$
$$= 0.9 + 0.25(0.5 - 0.9) = 0.8$$
$$w_{31}(n) = w_{31}(0) + \alpha\,[x_3 - w_{31}(0)]$$
$$= 0.7 + 0.25(1 - 0.7) = 0.775$$
$$w_{41}(n) = w_{41}(0) + \alpha\,[x_4 - w_{41}(0)]$$
$$= 0.5 + 0.25(0.5 - 0.5) = 0.5$$
$$w_{51}(n) = w_{51}(0) + \alpha\,[x_5 - w_{51}(0)]$$
$$= 0.3 + 0.25(0 - 0.3) = 0.225$$

The updated weight matrix for the winning unit given by

$$W = \begin{bmatrix} 0.75 & 0.3 \\ 0.8 & 0.5 \\ 0.775 & 0.7 \\ 0.5 & 0.9 \\ 0.225 & 1.0 \end{bmatrix}$$

6. Construct and test an LVQ net with five vectors assigned to two classes. The given vectors along with the classes are as shown in Table 1.

Table 1

Vector	Class
[0 0 1 1]	1
[1 0 0 0]	2
[0 0 0 1]	2
[1 1 0 0]	1
[0 1 1 0]	1

Solution: In the given five vectors, first two vectors are used as initial weight vectors and the remaining three vectors are used as input vectors. Based on this, LVQ net is shown in Figure 6, along with initial weights.

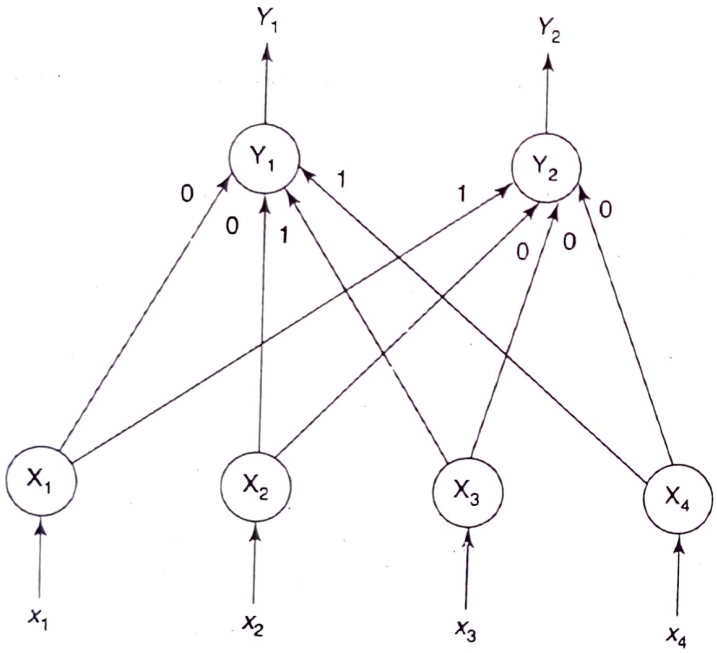

Figure 6 LVQ net.

Initialize the reference weight vectors as

$$w_1 = [0 \quad 0 \quad 1 \quad 1]; \quad w_2 = [1 \quad 0 \quad 0 \quad 0]$$

Let the learning rate be $\alpha = 0.1$.

First input vector

For [0 0 0 1] with $T = 2$, calculate the square of the Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^{4} (w_{ij} - x_i)^2$$

For $j = 1$ to 2,

$$D(1) = (0-0)^2 + (0-0)^2 + (1-0)^2 + (1-1)^2 = 1$$

$$D(2) = (1-0)^2 + (0-0)^2 + (0-0)^2 + (0-1)^2 = 2$$

Since $D(1) < D(2)$, $D(1)$ is minimum; hence winner unit index is $J = 1$. Now that $T \neq J$, weight updation is performed as

$$w_J(\text{new}) = w_J(\text{old}) - \alpha[x - w_J(\text{old})]$$

$$w_{11}(n) = w_{11}(0) - \alpha[x_1 - w_{11}(0)]$$
$$= 0 - 0.1(0 - 0) = 0$$

$$w_{21}(n) = w_{21}(0) - \alpha[x_2 - w_{21}(0)]$$
$$= 0 - 0.1(0 - 0) = 0$$

$$w_{31}(n) = w_{31}(0) - \alpha[x_3 - w_{31}(0)]$$
$$= 1 - 0.1(0 - 1) = 1.1$$

$$w_{41}(n) = w_{41}(0) - \alpha[x_4 - w_{41}(0)]$$
$$= 1 - 0.1(1 - 1) = 1$$

After the presentation of first input pattern, weight matrix becomes

$$W = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1.1 & 0 \\ 1 & 0 \end{bmatrix}$$

Second input vector

For [1 1 0 0] with $T = 1$, calculate the square of th Euclidean distance, i.e.,

$$D(j) = \sum_{i=1}^{4} (w_{ij} - x_i)^2$$

For $j = 1$ to 2,

$$D(1) = (0-1)^2 + (0-1)^2 + (1.1-0)^2 + (1-0)^2 = 4.21$$

$$D(2) = (1-1)^2 + (0-1)^2 + (0-0)^2 + (0-0)^2 = 1$$

Since $D(2) < D(1), D(2)$ is minimum; hence the winner unit index is $J = 2$. Again since $T \neq J$, the weight updation is performed as

$$w_J(\text{new}) = w_J(\text{old}) - \alpha[x - w_J(\text{old})]$$

$$w_{12}(n) = w_{12}(0) - \alpha[x_1 - w_{12}(0)]$$
$$= 1 - 0.1(1 - 1) = 1$$

$$w_{22}(n) = w_{22}(0) - \alpha[x_2 - w_{22}(0)]$$
$$= 0 - 0.1(1 - 0) = -0.1$$

$$w_{32}(n) = w_{32}(0) - \alpha[x_3 - w_{32}(0)]$$
$$= 0 - 0.1(0 - 0) = 0$$

$$w_{42}(n) = w_{42}(0) - \alpha[x_4 - w_{42}(0)]$$
$$= 0 - 0.1(0 - 0) = 0$$

After the presentation of second input pattern, the weight matrix becomes

$$\mathrm{W} = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \\ 1.1 & 0 \\ 1 & 0 \end{bmatrix}$$

Third input vector

For [0 1 1 0] with $T = 1$, calculate the square of the Euclidean distance as

$$D(j) = \sum_{i=1}^{4} (w_{ij} - x_i)^2$$

For $j = 1$ to 2,

$$D(1) = (0 - 0)^2 + (0 - 1)^2 + (1.1 - 1)^2 + (1 - 0)^2 = 2.01$$

$$D(2) = (1 - 0)^2 + (-0.1 - 1)^2 + (0 - 1)^2 + (0 - 0)^2 = 3.21$$

Since $D(1) < D(2), D(1)$ is minimum; hence the winner unit index is $J = 1$. Now that $T = J$, the weight updation is performed as

$$w_J(\text{new}) = w_J(\text{old}) + \alpha[x - w_J(\text{old})]$$

Updating the weights on the winner unit, we obtain

$$w_{11}(n) = w_{11}(0) + \alpha[x_1 - w_{11}(0)]$$
$$= 0 + 0.1(0 - 0) = 0$$

$$w_{21}(n) = w_{21}(0) + \alpha[x_2 - w_{21}(0)]$$
$$= 0 + 0.1(1 - 0) = 0.1$$

$$w_{31}(n) = w_{31}(0) + \alpha[x_3 - w_{31}(0)]$$
$$= 1.1 + 0.1(1 - 1.1) = 1.09$$

$$w_{41}(n) = w_{41}(0) + \alpha[x_4 - w_{41}(0)]$$
$$= 1 + 0.1(0 - 1) = 0.9$$

After the presentation of third input pattern, the weight matrix becomes

$$\mathrm{W} = \begin{bmatrix} 0 & 1 \\ 0.1 & -0.1 \\ 1.09 & 0 \\ 0.9 & 0 \end{bmatrix}$$

Thus the first epoch of the training has been completed. It is noted that if correct class is obtained for first and second input patterns, further epochs can be performed until all the winner units become equal to all the classes, i.e., all $T = J$.

7. Consider an LVQ net with two input units and four target classes: c_1, c_2, c_3 and c_4. There exist 16 classification units, with weight vectors indicated by the coordinates on the following chart, read in row–column order. For example, the unit with weight vector (0.2, 0.2), (0.2, 0.6) is assigned to represent class 1 and the classification units for class 2 have initial weight vectors of (0.4, 0.2), (0.4, 0.6), (0.8, 0.4) and (0.8, 0.8). The chart is given in Table 2.

Table 2

x_2						
1.0						
0.8	c_3	c_4	c_1	c_2		
0.6	c_1	c_2	c_3	c_4		
0.4	c_3	c_4	c_1	c_2		
0.2	c_1	c_2	c_3	c_4		
0.0						
0.0	0.2	0.4	0.6	0.8	1.0	x_1

Use square of Euclidean distance to measure the changes occurring.

- Given an input vector of (0.25, 0.25) representing class 1 and using a learning rate

of $\alpha = 0.25$, show which classification unit moves where (i.e., determine its new weight vector).

- Given the input vector of (0.4, 0.35) representing class 1, using initial weight vector and learning rate of $\alpha = 0.25$, note what happens?
- Given the input vector of (0.4, 0.45), determine the performance of the net. The input vector represents class 1.

Solution: The LVQ net for this problem with two input units and four cluster units is shown in Figure 7. The initial weight vectors for the respective classes are shown below.

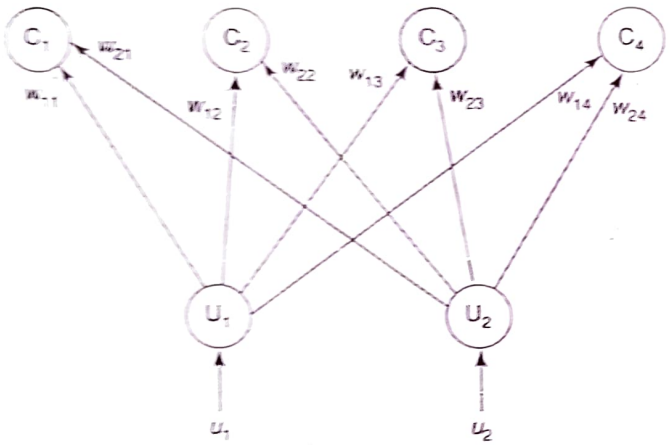

Figure 7 LVQ net (with two input units, four cluster units).

Class 1:

Initial weight vector

$$W_1 = \begin{bmatrix} 0.2 & 0.2 & 0.6 & 0.6 \\ 0.2 & 0.6 & 0.8 & 0.4 \end{bmatrix}$$

with target $t = 1$.

Class 2:

Initial weight vector

$$W_2 = \begin{bmatrix} 0.4 & 0.4 & 0.8 & 0.8 \\ 0.2 & 0.6 & 0.8 & 0.4 \end{bmatrix}$$

with target $t = 2$.

Class 3:

Initial weight vector

$$W_3 = \begin{bmatrix} 0.2 & 0.2 & 0.6 & 0.6 \\ 0.4 & 0.8 & 0.6 & 0.2 \end{bmatrix}$$

with target $t = 3$.

Class 4:

Initial weight vector

$$W_4 = \begin{bmatrix} 0.4 & 0.4 & 0.8 & 0.8 \\ 0.4 & 0.8 & 0.6 & 0.2 \end{bmatrix}$$

with target $t = 4$.

- For the given input vector $(u_1, u_2) = (0.25, 0.25)$ with $\alpha = 0.25$ and $t = 1$, we calculate the square of the Euclidean distance using the formula

$$D(j) = \sum_{i=1}^{2} (w_{ij} - x_i)^2 = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2$$

For $j = 1$ to 4,

$$D(1) = (0.2 - 0.25)^2 + (0.2 - 0.25)^2 = 0.005$$
$$D(2) = (0.2 - 0.25)^2 + (0.6 - 0.25)^2 = 0.125$$
$$D(3) = (0.6 - 0.25)^2 + (0.8 - 0.25)^2 = 0.425$$
$$D(4) = (0.6 - 0.25)^2 + (0.4 - 0.25)^2 = 0.145$$

As $D(1)$ is minimum, therefore the winner unit index is $J = 1$. Now we update the weights on the winner unit, since $t = J = 1, \alpha = 0.25$, using the weight updation formula

$$w_J(\text{new}) = w_J(\text{old}) + \alpha[x - w_J(\text{old})]$$

Updating the weights on the winner unit, we obtain

$$\begin{aligned} w_{11}(\text{new}) &= w_{11}(0) + \alpha[x_1 - w_{11}(\text{old})] \\ &= 0.2 + 0.25(0.25 - 0.2) = 0.2125 \\ w_{21}(\text{new}) &= w_{21}(0) + \alpha[x_2 - w_{21}(\text{old})] \\ &= 0.2 + 0.25(0.25 - 0.2) = 0.2125 \end{aligned}$$

Therefore, the new weight vector is

$$W_1 = \begin{bmatrix} 0.2125 & 0.2 & 0.6 & 0.6 \\ 0.2125 & 0.6 & 0.8 & 0.4 \end{bmatrix}$$

- For the given input vector $(u_1, u_2) = (0.4, 0.35)$ with $\alpha = 0.25$ and $t = 1$, we calculate the square of the Euclidean distance using the formula

$$D(j) = \sum_{i=1}^{2} (w_{ij} - x_i)^2 = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2$$

For $j = 1$ to 4,

$$D(1) = (0.2 - 0.4)^2 + (0.2 - 0.35)^2 = 0.0625$$
$$D(2) = (0.2 - 0.4)^2 + (0.6 - 0.35)^2 = 0.1025$$
$$D(3) = (0.6 - 0.4)^2 + (0.8 - 0.35)^2 = 0.2425$$
$$D(4) = (0.6 - 0.4)^2 + (0.4 - 0.35)^2 = 0.0425$$

As $D(4)$ is minimum, therefore the winner unit index is $J = 4$. Thus, fourth unit is the winner unit that is closest to the input vector. Since $t \neq J$, the weight updation formula used is

$$w_J(\text{new}) = w_J(\text{old}) - \alpha[x - w_J(\text{old})]$$

Updating the weights on the winner unit, we obtain

$$w_{14}(\text{new}) = w_{14}(0) - \alpha[x_1 - w_{14}(\text{old})]$$
$$= 0.6 - 0.25(0.4 - 0.6) = 0.65$$
$$w_{24}(\text{new}) = w_{24}(0) - \alpha[x_2 - w_{24}(\text{old})]$$
$$= 0.4 - 0.25(0.35 - 0.4) = 0.4125$$

Therefore, the new weight vector is

$$\mathrm{W}_1 = \begin{bmatrix} 0.2 & 0.2 & 0.6 & 0.65 \\ 0.2 & 0.6 & 0.8 & 0.4125 \end{bmatrix}$$

- For the given input vector $(u_1, u_2) = (0.4, 0.45)$ with $\alpha = 0.25$ and $t = 1$, we calculate the square of the Euclidean distance using the formula

$$D(j) = \sum_{i=1}^{2} (w_{ij} - x_i)^2 = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2$$

For $j = 1$ to 4,

$$D(1) = (0.2 - 0.4)^2 + (0.2 - 0.45)^2 = 0.1025$$
$$D(2) = (0.2 - 0.4)^2 + (0.6 - 0.45)^2 = 0.0625$$
$$D(3) = (0.6 - 0.4)^2 + (0.8 - 0.45)^2 = 0.1625$$
$$D(4) = (0.6 - 0.4)^2 + (0.4 - 0.45)^2 = 0.0425$$

As $D(4)$ is minimum, therefore in this case also the winner unit index is $J = 4$. Since $t \neq J$, the weight updation formula used is

$$w_J(\text{new}) = w_J(\text{old}) - \alpha[x - w_J(\text{old})]$$

Updating the weights on the winner unit, we obtain

$$w_{14}(\text{new}) = w_{14}(0) - \alpha[x_1 - w_{14}(\text{old})]$$
$$= 0.6 - 0.25(0.4 - 0.6) = 0.65$$
$$w_{24}(\text{new}) = w_{24}(0) - \alpha[x_2 - w_{24}(\text{old})]$$
$$= 0.4 - 0.25(0.45 - 0.4) = 0.3875$$

Therefore, the new weight vector is

$$\mathrm{W}_1 = \begin{bmatrix} 0.2 & 0.2 & 0.6 & 0.65 \\ 0.2 & 0.6 & 0.8 & 0.3875 \end{bmatrix}$$

8. Consider the following full CPN shown in Figure 8. Using the input pair $x = [1\ 0\ 0\ 0]$ and $y = [1\ 0]$, perform the phase I of training (one step only). Find the activation of the cluster layer units and update the weights using learning rates $\alpha = \beta = 0.2$.

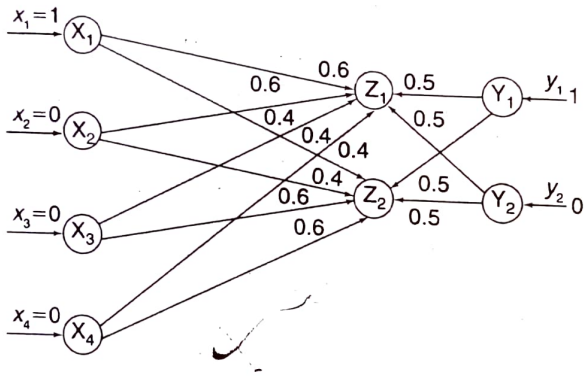

Figure 8 Instar model of CPN net.

Solution: The input pair is $x = [1\ 0\ 0\ 0]$ and $y = [1\ 0]$ and the learning rates are $\alpha = 0.2$ and $\beta = 0.2$.

Phase I of training: The initial weights are obtained from Figure 8 as

$$\mathrm{V} = \begin{bmatrix} 0.6 & 0.4 \\ 0.6 & 0.4 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix} \quad \text{and} \quad \mathrm{W} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$