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 $\mathbf{x} = (-1, -1, 1, 1)$.

4.2 KOHONEN SELF-ORGANIZING MAPS

The self-organizing neural networks described in this section, also called *topology-preserving maps*, assume a topological structure among the cluster units. This property is observed in the brain, but is not found in other artificial neural networks. There are m cluster units, arranged in a one- or two-dimensional array; the input signals are n-tuples [Kohonen, 1989a].

The weight vector for a cluster unit serves as an exemplar of the input patterns associated with that cluster. During the self-organization process, the cluster unit whose weight vector matches the input pattern most closely (typically, the square of the minimum Euclidean distance) is chosen as the winner. The winning unit and its neighboring units (in terms of the topology of the cluster units) update their weights. The weight vectors of neighboring units are not, in general, close to the input pattern. For example, for a linear array of cluster units, the neighborhood of radius R around cluster unit J consists of all units j such that $\max(1, J - R) \leq j \leq \min(J + R, m)$.

The architecture and algorithm that follow for the net can be used to cluster a set of p continuous-valued vectors $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$ into m clusters. Note that the connection weights do not multiply the signal sent from the input units to the cluster units (unless the dot product measure of similarity is being used).

4.2.1 Architecture

The architecture of the Kohonen self-organizing map is shown in Figure 4.5.

Neighborhoods of the unit designated by # of radii $R = 2$, 1, and 0 in a one-dimensional topology (with 10 cluster units) are shown in Figure 4.6.

The neighborhoods of radii $R = 2$, 1 and 0 are shown in Figure 4.7 for a rectangular grid and in Figure 4.8 for a hexagonal grid (each with 49 units). In each illustration, the winning unit is indicated by the symbol "#" and the other units are denoted by "*."

Note that each unit has eight nearest neighbors in the rectangular grid, but only six in the hexagonal grid. Winning units that are close to the edge of the grid will have some neighborhoods that have fewer units than that shown in the respective figure. (Neighborhoods do not "wrap around" from one side of the grid to the other; "missing" units are simply ignored.)

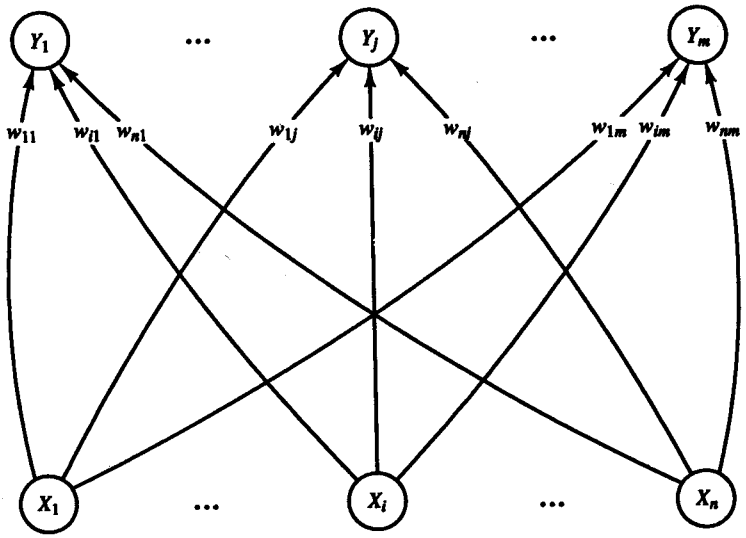

Figure 4.5 Kohonen self-organizing map.

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

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

Figure 4.6 Linear array of cluster units.

4.2.2 Algorithm

Step 0. Initialize weights w_{ij}. (Possible choices are discussed below.)
Set topological neighborhood parameters.
Set learning rate parameters.

Step 1. While stopping condition is false, do Steps 2–8.

Step 2. For each input vector **x**, do Steps 3–5.

Step 3. For each j, compute:

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

Step 4. Find index J such that $D(J)$ is a minimum.

Step 5. For all units j within a specified neighborhood of J, and for all i:

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

Step 6. Update learning rate.

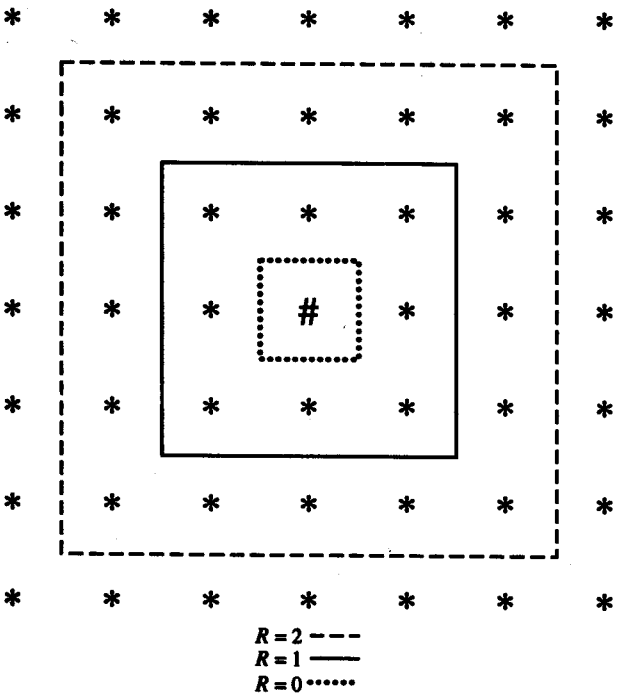

Figure 4.7 Neighborhoods for rectangular grid.

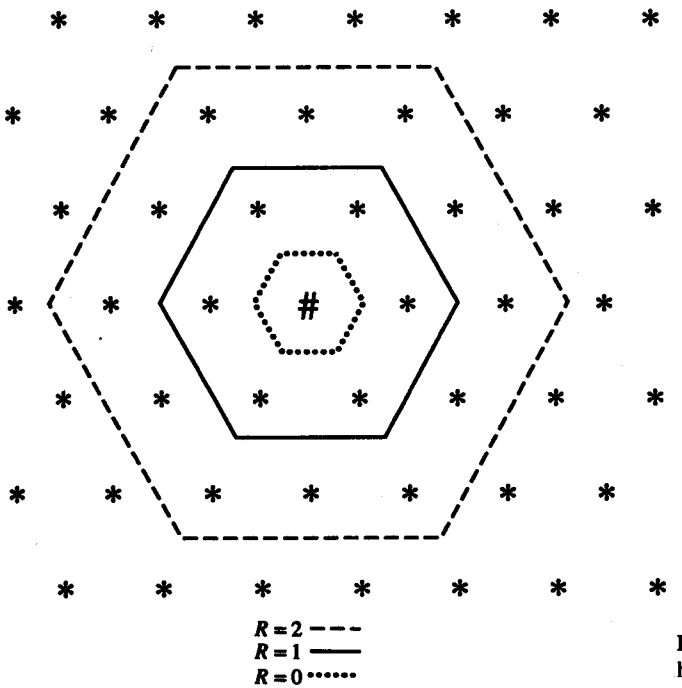

Figure 4.8 Neighborhoods for hexagonal grid.

Step 7. Reduce radius of topological neighborhood at specified times.

Step 8. Test stopping condition.

Alternative structures are possible for reducing R and α.

The learning rate α is a slowly decreasing function of time (or training epochs). Kohonen (1989a, p. 133) indicates that a linearly decreasing function is satisfactory for practical computations; a geometric decrease would produce similar results.

The radius of the neighborhood around a cluster unit also decreases as the clustering process progresses.

The formation of a map occurs in two phases: the initial formation of the correct order and the final convergence. The second phase takes much longer than the first and requires a small value for the learning rate. Many iterations through the training set may be necessary, at least in some applications [Kohonen, 1989a].

Random values may be assigned for the initial weights. If some information is available concerning the distribution of clusters that might be appropriate for a particular problem, the initial weights can be taken to reflect that prior knowledge. In Examples 4.4–4.9, the weights are initialized to random values (chosen from the same range of values as the components of the input vectors).

4.2.3 Application

Neural networks developed by Kohonen have been applied to an interesting variety of problems. One recent development of his is a neural network approach to computer-generated music [Kohonen, 1989b]. Angeniol, Vaubois, and Le Texier (1988) have applied Kohonen self-organizing maps to the solution of the well-known traveling salesman problem. These applications are discussed briefly later in this section. A more common neural network approach to the traveling salesman problem is discussed in Chapter 7.

Simple example

Example 4.4 A Kohonen self-organizing map (SOM) to cluster four vectors

Let the vectors to be clustered be

$$(1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, 0); (0, 0, 1, 1).$$

The maximum number of clusters to be formed is

$$m = 2.$$

Suppose the learning rate (geometric decrease) is

$$\alpha(0) = .6,$$

$$\alpha(t + 1) = .5\,\alpha(t).$$

With only two clusters available, the neighborhood of node J (Step 4) is set so that only one cluster updates its weights at each step (i.e., $R = 0$).

Step 0. Initial weight matrix:

$$\begin{bmatrix} .2 & .8 \\ .6 & .4 \\ .5 & .7 \\ .9 & .3 \end{bmatrix}.$$

Initial radius:

$$R = 0.$$

Initial learning rate:

$$\alpha(0) = 0.6.$$

Step 1. Begin training.

Step 2. For the first vector, (1, 1, 0, 0), do Steps 3–5.

Step 3.
$$D(1) = (.2 - 1)^2 + (.6 - 1)^2 + (.5 - 0)^2 + (.9 - 0)^2 = 1.86;$$
$$D(2) = (.8 - 1)^2 + (.4 - 1)^2 + (.7 - 0)^2 + (.3 - 0)^2 = 0.98.$$

Step 4. The input vector is closest to output node 2, so

$$J = 2.$$

Step 5. The weights on the winning unit are updated:

$$w_{i2}(\text{new}) = w_{i2}(\text{old}) + .6\,[x_i - w_{i2}(\text{old})]$$
$$= .4\, w_{i2}(\text{old}) + .6\, x_i.$$

This gives the weight matrix

$$\begin{bmatrix} .2 & .92 \\ .6 & .76 \\ .5 & .28 \\ .9 & .12 \end{bmatrix}.$$

Step 2. For the second vector, (0, 0, 0, 1), do Steps 3–5.

Step 3.
$$D(1) = (.2 - 0)^2 + (.6 - 0)^2 + (.5 - 0)^2 + (.9 - 1)^2 = 0.66;$$
$$D(2) = (.92 - 0)^2 + (.76 - 0)^2 + (.28 - 0)^2 + (.12 - 1)^2 = 2.2768.$$

Step 4. The input vector is closest to output node 1, so

$$J = 1.$$

Step 5. Update the first column of the weight matrix:

$$\begin{bmatrix} .08 & .92 \\ .24 & .76 \\ .20 & .28 \\ .96 & .12 \end{bmatrix}.$$

Step 2. For the third vector, (1, 0, 0, 0), do Steps 3–5.

Step 3.

$$D(1) = (.08 - 1)^2 + (.24 - 0)^2 + (.2 - 0)^2 + (.96 - 0)^2 = 1.8656;$$

$$D(2) = (.92 - 1)^2 + (.76 - 0)^2 + (.28 - 0)^2 + (.12 - 0)^2 = 0.6768.$$

Step 4. The input vector is closest to output node 2, so

$$J = 2.$$

Step 5. Update the second column of the weight matrix:

$$\begin{bmatrix} .08 & .968 \\ .24 & .304 \\ .20 & .112 \\ .96 & .048 \end{bmatrix}.$$

Step 2. For the fourth vector, (0, 0, 1, 1), do Steps 3–5.

Step 3.

$$D(1) = (.08 - 0)^2 + (.24 - 0)^2 + (.2 - 1)^2 + (.96 - 1)^2 = 0.7056;$$

$$D(2) = (.968 - 0)^2 + (.304 - 0)^2 + (.112 - 1)^2 + (.048 - 1)^2 = 2.724.$$

Step 4.

$$J = 1.$$

Step 5. Update the first column of the weight matrix:

$$\begin{bmatrix} .032 & .968 \\ .096 & .304 \\ .680 & .112 \\ .984 & .048 \end{bmatrix}.$$

Step 6. Reduce the learning rate:

$$\alpha = .5\ (0.6) = .3$$

The weight update equations are now

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + .3\ [x_i - w_{ij}(\text{old})]$$

$$= .7w_{ij}(\text{old}) + .3x_i.$$

The weight matrix after the second epoch of training is

$$\begin{bmatrix} .016 & .980 \\ .047 & .360 \\ .630 & .055 \\ .999 & .024 \end{bmatrix}$$

Modifying the adjustment procedure for the learning rate so that it decreases geometrically from .6 to .01 over 100 iterations (epochs) gives the following results:

Iteration 0: Weight matrix: $\begin{bmatrix} .2 & .8 \\ .6 & .4 \\ .5 & .7 \\ .9 & .3 \end{bmatrix}$

Iteration 1: Weight matrix: $\begin{bmatrix} .032 & .970 \\ .096 & .300 \\ .680 & .110 \\ .980 & .048 \end{bmatrix}$

Iteration 2: Weight matrix: $\begin{bmatrix} .0053 & .9900 \\ -.1700 & .3000 \\ .7000 & .0200 \\ 1.0000 & .0086 \end{bmatrix}$

Iteration 10: Weight matrix: $\begin{bmatrix} 1.5e\text{-}7 & 1.0000 \\ 4.6e\text{-}7 & .3700 \\ .6300 & 5.4e\text{-}7 \\ 1.0000 & 2.3e\text{-}7 \end{bmatrix}$

Iteration 50: Weight matrix: $\begin{bmatrix} 1.9e\text{-}19 & 1.0000 \\ 5.7e\text{-}15 & .4700 \\ .5300 & 6.6e\text{-}15 \\ 1.0000 & 2.8e\text{-}15 \end{bmatrix}$

Iteration 100: Weight matrix: $\begin{bmatrix} 6.7e\text{-}17 & 1.0000 \\ 2.0e\text{-}16 & .4900 \\ .5100 & 2.3e\text{-}16 \\ 1.0000 & 1.0e\text{-}16 \end{bmatrix}$

These weight matrices appear to be converging to the matrix

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{bmatrix},$$

the first column of which is the average of the two vectors placed in cluster 1 and the second column of which is the average of the two vectors placed in cluster 2.

Character Recognition

Examples 4.5–4.7 show typical results from using a Kohonen self-organizing map to cluster input patterns representing letters in three different fonts. The input patterns for fonts 1, 2, and 3 are given in Figure 4.9. In each of the examples, 25 cluster units are available, which means that a maximum of 25 clusters may be formed. Results are shown only for the units that are actually the winning unit for some input pattern after training. The effect of the topological structure is seen in the contrast between Example 4.5 (in which there is no structure), Example 4.6 (in which there is a linear structure as described before), and Example 4.7 (in which a rectangular structure is used). In each example, the learning rate is reduced linearly from an initial value of .6 to a final value of .01.

Example 4.5 A SOM to cluster letters from different fonts: no topological structure

If no structure is assumed for the cluster units, i.e., if only the winning unit is allowed to learn the pattern presented, the 21 patterns form 5 clusters:

UNIT	PATTERNS
3	C1, C2, C3
13	B1, B3, D1, D3, E1, K1, K3, E3
16	A1, A2, A3
18	J1, J2, J3
24	B2, D2, E2, K2

Example 4.6 A SOM to cluster letters from different fonts: linear structure

A linear structure (with $R = 1$) gives a better distribution of the patterns onto the available cluster units. The winning node J and its topological neighbors ($J + 1$ and $J - 1$) are allowed to learn on each iteration. Note that in general, the neighboring nodes that learn do not initially have weight vectors that are particularly close to the input pattern.

UNIT	PATTERNS	UNIT	PATTERNS
6	K2	20	C1, C2, C3
10	J1, J2, J3	22	D2
14	E1, E3	23	B2, E2
16	K1, K3	25	A1, A2, A3
18	B1, B3, D1, D3		

Note also that in many cases there are unused units between a pair of units that have clusters of patterns associated with them. This suggests that units which are being pulled in opposite directions during training do not learn any pattern very well. (In other words, in most cases, these input patterns form very distinct classes.)

Example 4.7 A SOM to cluster letters from different fonts: diamond structure

In this example, a simple two-dimensional topology is assumed for the cluster units, so that each cluster unit is indexed by two subscripts. If unit X_{IJ} is the winning unit, the units $X_{I+1,J}$, $X_{I-1,J}$, $X_{I,J+1}$, and $X_{I,J-1}$ also learn. This gives a diamond to-

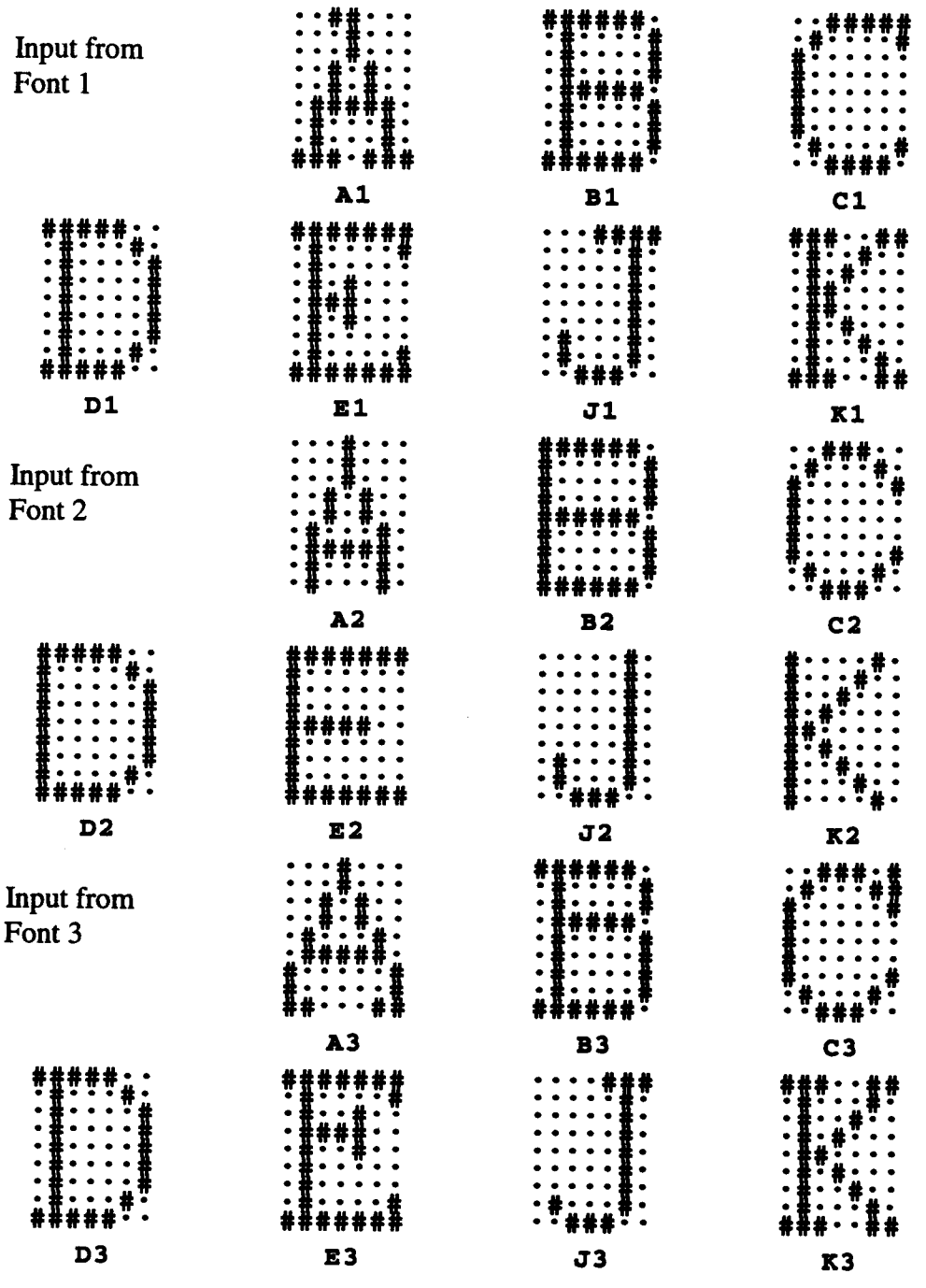

Figure 4.9 Training input patterns for character recognition examples.

i\j	1	2	3	4	5
1		J1, J2, J3		D2	
2	C1, C2, C3		D1, D3		B2, E2
3		B1		K2	
4			E1, E3, B3		A3
5		K1, K3		A1, A2	

Figure 4.10 Character recognition with rectangular grid.

pology, rather than the entire rectangle illustrated in Figure 4.7. The results are shown in Figure 4.10.

Spanning tree

Example 4.8 Using a SOM: Spanning Tree Data

The 32 vectors [Kohonen, 1989a] shown in Figure 4.11 were presented in random order to a Kohonen self-organizing map with a rectangular topology on its cluster units. There were 70 cluster units arranged in a 10 × 7 array. The pattern names are for ease of identification of the results. The relationships between the patterns can be displayed graphically, as in Figure 4.12 [Kohonen, 1989a]; patterns that are adjacent to each other in the diagram differ by exactly 1 bit.

The net was used with random initial weights. In this example, the initial radius, $R = 3$, was reduced by 1 after each set of 75 iterations. During these 75 iterations, the learning rate was reduced linearly from .6 to .01. If unit $X_{I,J}$ is the winning unit, the units $X_{i,j}$ for all i and j such that $I - R \leq i \leq I + R$ and $J - R \leq j \leq J + R$ also learn (unless the value of i or j falls outside the permissible range for the topology and number of cluster units chosen). Note that when $R = 3$, as many as 49 units will learn (see Figure 4.7). When the Kohonen net is used with $R = 0$, only the winning cluster node is allowed to learn.

Figures 4.13–4.16 show the evolution of the solution, as R is decreased, for the data in Figure 4.11, using a rectangular array for the cluster units. The structure of the data is shown in Figure 4.16 to indicate how the positioning of the patterns on the cluster units reflects the spanning tree relationships among the patterns.

A hexagonal grid can also be used for a two-dimensional topology. The final results obtained using such a grid are shown in Figure 4.17. As in Figure 4.16, the structure of the data is also indicated to show how the position of the patterns on the cluster units reflects the original spanning tree. The same iteration scheme was used as before, i.e., 75 iterations at each radius, starting with $R = 3$ and decreasing to $R = 0$.

Other examples

Example 4.9 Using a SOM: A Geometric Example

The cluster units in a Kohonen self-organizing map can viewed as having a position (given by their weight vector). For input patterns with two components, this position is easy to represent graphically. The topological relationships between cluster units