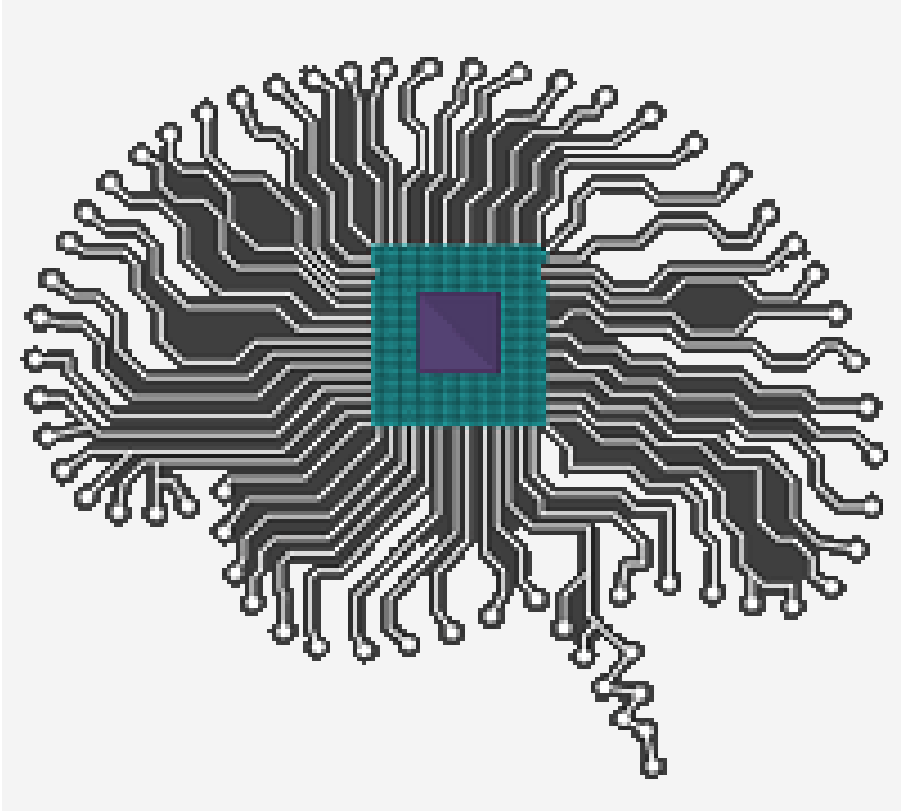


Artificial Neural Networks

By: Sajad Ahmadian

Kermanshah University of Technology

Unsupervised Learning

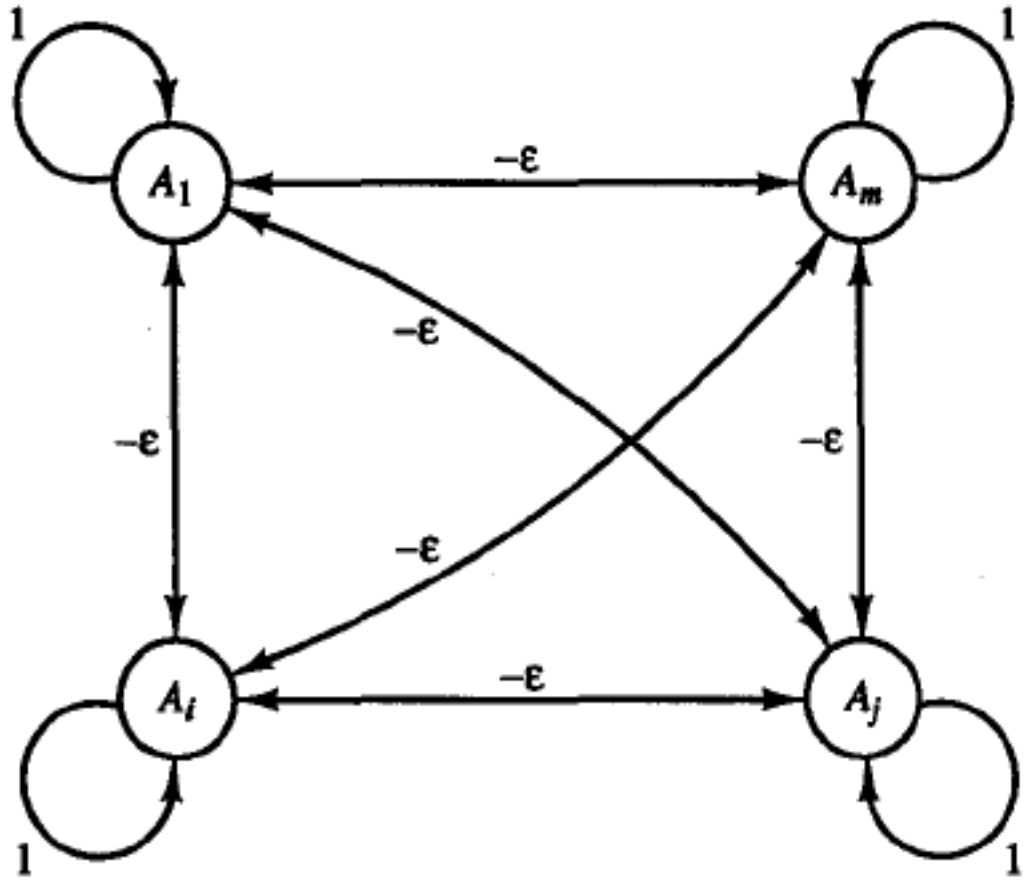
- This type of learning is done without the supervision of a teacher. This learning process is independent.
- During the training of ANN under unsupervised learning:
  - The input vectors of similar type are combined to form clusters.
  - When a new input pattern is applied, then the neural network gives an output response indicating the class to which input pattern belongs.
  - There would be no feedback from the environment as to what should be the desired output and whether it is correct or incorrect.
  - In this type of learning the network itself must discover the patterns, features from the input data and the relation for the input data over the output.

Winner-Takes-All Networks

- These kinds of networks are based on the competitive learning rule and will use the strategy where it chooses the neuron with the greatest total inputs as a winner.
- The connections between the output neurons show the competition between them and one of them would be ‘ON’ which means it would be the winner and others would be ‘OFF’.

Max Net

- This is also a fixed weight network, which serves as a subnet for selecting the node having the largest input. All the nodes are fully interconnected and there exists symmetrical weights in all these weighted interconnections.



Max Net

- It uses the mechanism which is an iterative process and each node receives inhibitory inputs from all other nodes through connections.
- The single node whose value is maximum would be active or winner and the activations of all other nodes would be inactive. Max Net uses identity activation function with:

$$f(x) = \begin{cases} x & if\, x > 0 \\ 0 & if\, x \leq 0 \end{cases}$$

- The task of this net is accomplished by the self-excitation weight of +1 and mutual inhibition magnitude, which is set like $[0 < \varepsilon < \frac{1}{m}]$ where “m” is the total number of the nodes.

Max Net

Step 0. Initialize activations and weights (set $0 < \epsilon < \frac{1}{m}$):

$a_j(0)$ input to node A_j,

$$w_{ij} = \begin{cases} 1 & \text{if } i = j; \\ -\epsilon & \text{if } i \neq j. \end{cases}$$

Step 1. While stopping condition is false, do Steps 2–4.

Step 2. Update the activation of each node: For $j = 1, \ldots, m$,

$$a_j(\text{new}) = f[a_j(\text{old}) - \epsilon \sum_{k \neq j} a_k(\text{old})].$$

Step 3. Save activations for use in next iteration:

$$a_j(\text{old}) = a_j(\text{new}), j = 1, \ldots, m.$$

Step 4. Test stopping condition:
If more than one node has a nonzero activation, continue; otherwise, stop.

Note that in Step 2, the input to the function f is simply the total input to node A_j from all nodes, including itself. Some precautions should be incorporated to handle the situation in which two or more units have the same, maximal, input.

Max Net (Example)

Consider the action of a MAXNET with four neurons and inhibitory weights $\epsilon = 0.2$ when given the initial activations (input signals)

$$a_1(0) = 0.2 \quad a_2(0) = 0.4 \quad a_3(0) = 0.6 \quad a_4(0) = 0.8$$

The activations found as the net iterates are

$$a_1(1) = 0.0 \quad a_2(1) = 0.08 \quad a_3(1) = 0.32 \quad a_4(1) = 0.56$$

$$a_1(2) = 0.0 \quad a_2(2) = 0.0 \quad a_3(2) = 0.192 \quad a_4(2) = 0.48$$

$$a_1(3) = 0.0 \quad a_2(3) = 0.0 \quad a_3(3) = 0.096 \quad a_4(3) = 0.442$$

$$a_1(4) = 0.0 \quad a_2(4) = 0.0 \quad a_3(4) = 0.008 \quad a_4(4) = 0.422$$

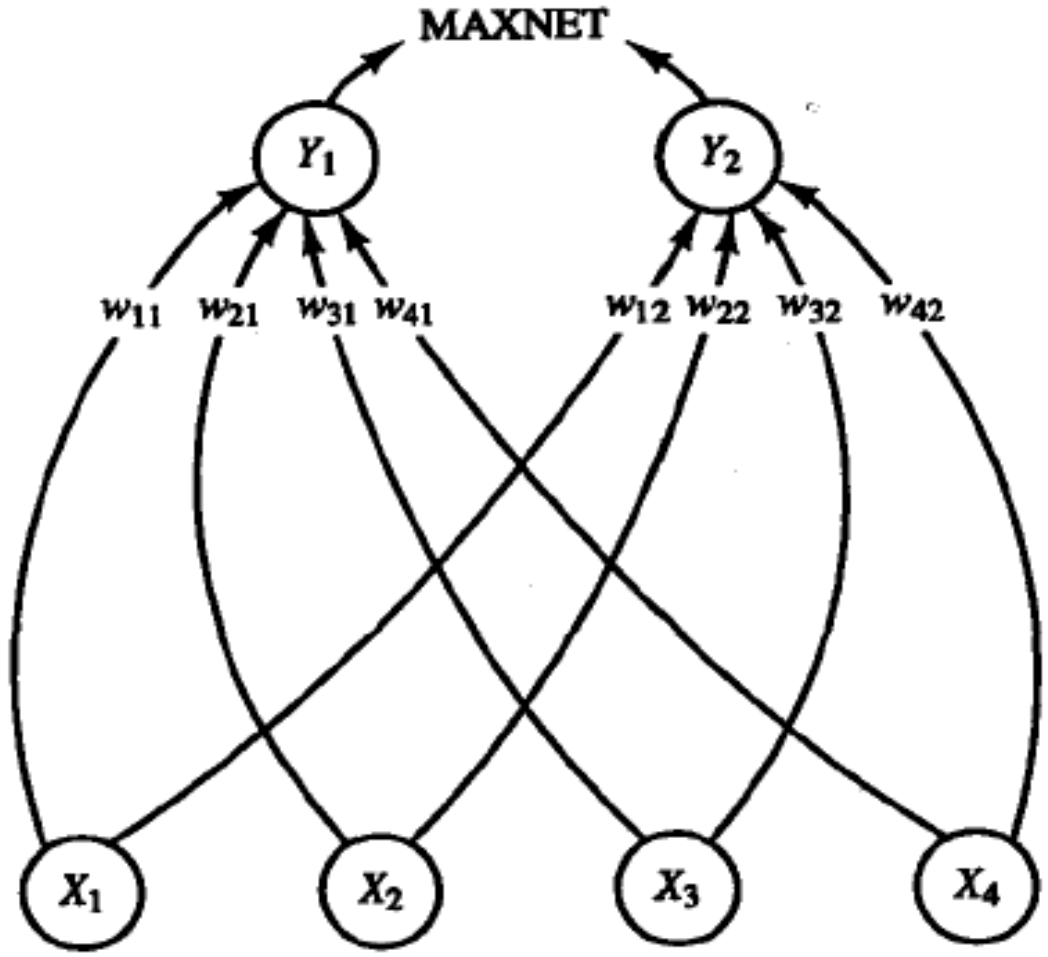
$$a_1(5) = 0.0 \quad a_2(5) = 0.0 \quad a_3(5) = 0.0 \quad a_4(5) = 0.421$$

Hamming Network

- In most of the neural networks using unsupervised learning, it is essential to compute the distance and perform comparisons.
- This kind of network is Hamming network, where for every given input vectors, it would be clustered into different groups.
  - It is a single layer network.
  - The inputs can be either binary {0, 1} of bipolar {-1, 1}.
  - The weights of the net are calculated by the exemplar vectors.
  - It is a fixed weight network which means the weights would remain the same even during training.

Hamming Network

- A sample architecture for 4-tuples input vectors to be categorized as belonging to one of two classes.



Hamming Network

n number of input nodes, number of components of any input vector;
m number of output nodes, number of exemplar vectors;
$\mathbf{e}(j)$ the jth exemplar vector:

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

The application procedure for the Hamming net is:

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

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

And initialize the biases:

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

Hamming Network

Step 1. **For each vector x, do Steps 2–4.**

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

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

Step 3. Initialize activations for **MAXNET**:

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

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

Hamming Network

Example: A Hamming net to cluster four vectors

Given the exemplar vectors

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

and

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

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

Hamming Network

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

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

Initialize the biases:

$$b_1 = b_2 = 2.$$

Hamming Network

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

Step 2. $y\_in_1 = b_1 + \sum_i x_i w_{i1}$

$= 2 + 1 = 3;$

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

$= 2 - 1 = 1.$

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

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

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

Hamming Network

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

Step 2. $y\_in_1 = b_1 + \sum_i x_i w_{i1}$

$= 2 + 2 = 4;$

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

$= 2 + 0 = 2.$

Note that the input vector agrees with **e(1)** in all four components and agrees with **e(2)** in the second and third components.

Step 3. $y_1(0) = 4;$

$y_2(0) = 2.$

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

Hamming Network

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

Step 2. $y\_in_1 = b_1 + \sum_i x_i w_{i1}$

$= 2 + 0 = 2;$

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

$= 2 + 2 = 4.$

The input vector agrees with **e(1)** in the second and third components and agrees with **e(2)** in all four components.

Step 3. $y_1(0) = 2;$

$y_2(0) = 4.$

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

Hamming Network

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

Step 2. $y\_in_1 = b_1 + \sum_i x_i w_{i1}$

$= 2 - 1 = 1;$

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

$= 2 + 1 = 3.$

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

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

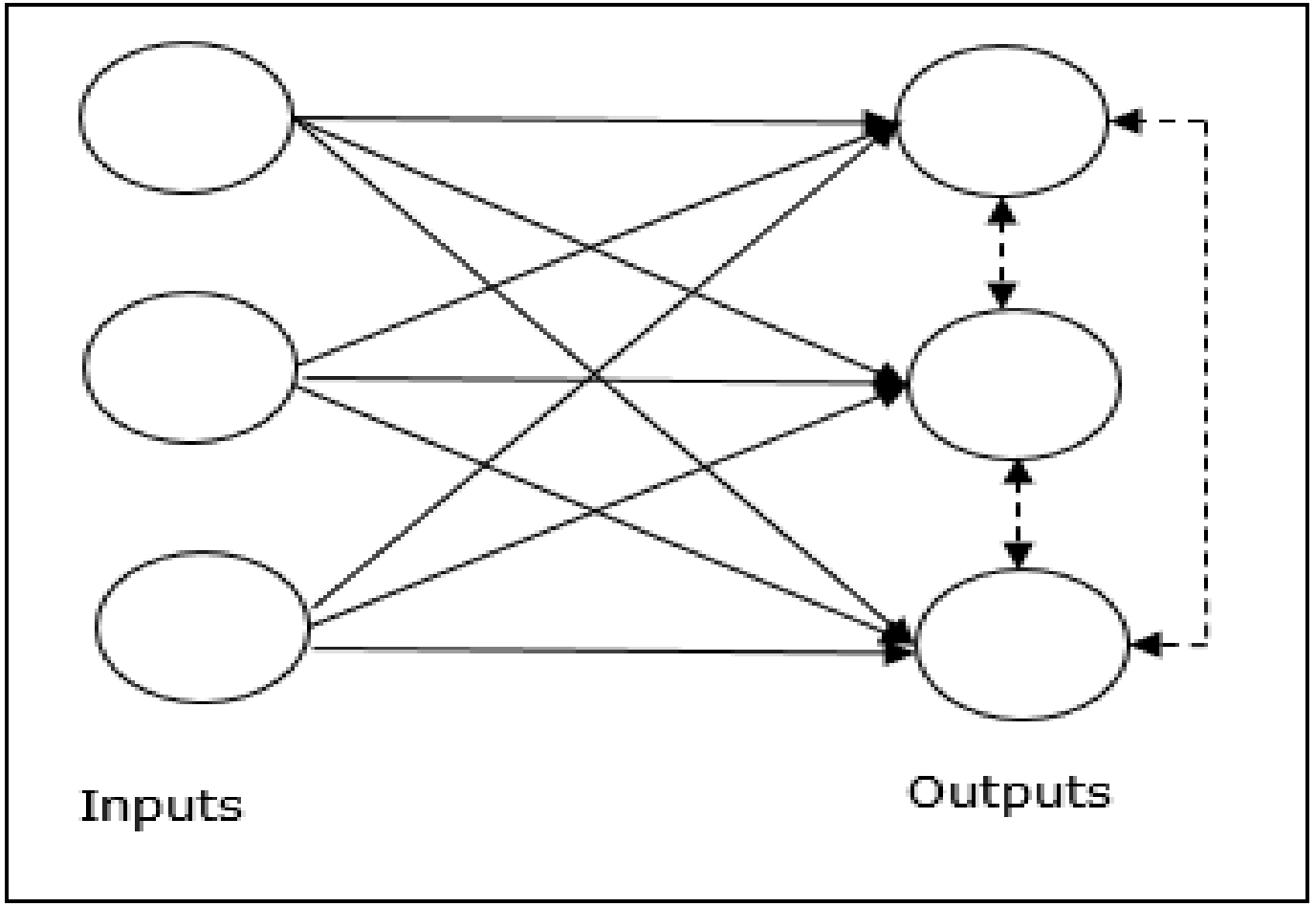
$y_2(0) = 3.$

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

Competitive Learning in ANN

- It is concerned with unsupervised training in which the output nodes try to compete with each other to represent the input pattern.
- Basic Concept of Competitive Network
  - This network is just like a single layer feed-forward network having feedback connection between the outputs.
  - The connections between the outputs are inhibitory type, which is shown by dotted lines, which means the competitors never support themselves.

Competitive Learning in ANN



Basic Concept of Competitive Learning Rule

- There would be competition among the output nodes so the main concept is - during training, the output unit that has the highest activation to a given input pattern, will be declared the winner.
- This rule is also called Winner-takes-all because only the winning neuron is updated and the rest of the neurons are left unchanged.

Basic Concept of Competitive Learning Rule

- Condition to be a winner

  Suppose if a neuron y_k wants to be the winner, then there would be the following condition

$$y_k = \begin{cases} 1 & if\ v_k > v_j\ for\ all\ j,\ j \neq k \\ 0 & otherwise \end{cases}$$

It means that if any neuron, say, y_k wants to win, then its induced local field $the output of the summation unit$, say v_k, must be the largest among all the other neurons in the network.

Basic Concept of Competitive Learning Rule

- Condition of the sum total of weight

  Another constraint over the competitive learning rule is the sum total of weights to a particular output neuron is going to be 1. For example, if we consider neuron **k** then

$$\sum_{k} w_{kj} = 1 \quad for\ all\ k$$

Basic Concept of Competitive Learning Rule

- Change of weight for the winner

If a neuron does not respond to the input pattern, then no learning takes place in that neuron. However, if a particular neuron wins, then the corresponding weights are adjusted as follows −

$$\Delta w_{kj} = \begin{cases} -\alpha(x_j - w_{kj}), & if\ neuron\ k\ wins \\ 0 & if\ neuron\ k\ losses \end{cases}$$

Here α is the learning rate.

This clearly shows that we are favoring the winning neuron by adjusting its weight and if a neuron is lost, then we need not bother to re-adjust its weight.

K-means Clustering Algorithm

- K-means is one of the most popular clustering algorithm in which we use the concept of partition procedure.
- We start with an initial partition and repeatedly move patterns from one cluster to another, until we get a satisfactory result.

K-means Clustering Algorithm

Algorithm

Step 1 – Select **k** points as the initial centroids. Initialize **k** prototypes **(w_1,…,w_k)**, for example we can identifying them with randomly chosen input vectors –

$$W_j = i_p, \quad where\, j \in \{1, \ldots, k\}\, and\, p \in \{1, \ldots, n\}$$

Each cluster C_j is associated with prototype w_j.

Step 2 – Repeat step 3-5 until E no longer decreases, or the cluster membership no longer changes.

Step 3 – For each input vector i_p where $p \in \{1,\ldots,n\}$, put i_p in the cluster C_{j*} with the nearest prototype w_{j*} having the following relation

$$|i_p - w_{j*}| \leq |i_p - w_j|,\ j \in \{1, \ldots, k\}$$

K-means Clustering Algorithm

Step 4 – For each cluster $\mathbf{C_j}$, where $\mathbf{j \in \{1,\ldots,k\}}$, update the prototype $\mathbf{w_j}$ to be the centroid of all samples currently in $\mathbf{C_j}$, so that

$$w_j = \sum_{i_p \in C_j} \frac{i_p}{|C_j|}$$

Step 5 – Compute the total quantization error as follows –

$$E = \sum_{j=1}^{k} \sum_{i_p \in w_j} |i_p - w_j|^2$$

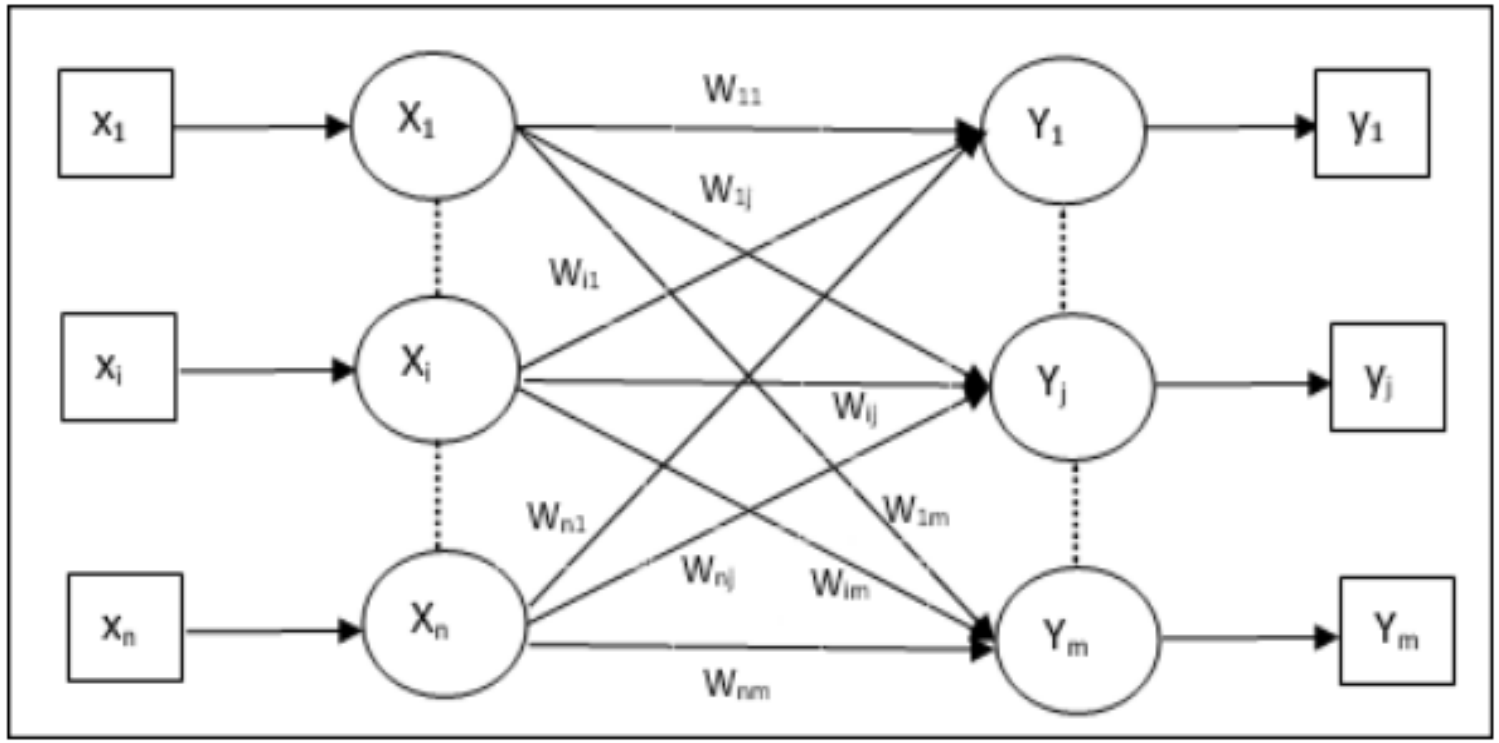
Learning Vector Quantization

- Learning Vector Quantization (LVQ) basically is a competitive network which uses supervised learning.
- We may define it as a process of classifying the patterns where each output unit represents a class.
- As it uses supervised learning, the network will be given a set of training patterns with known classification along with an initial distribution of the output class.
- After completing the training process, LVQ will classify an input vector by assigning it to the same class as that of the output unit.

Learning Vector Quantization

Architecture

Following figure shows the architecture of LVQ which is quite similar to the architecture of KSOM. As we can see, there are **"n"** number of input units and **"m"** number of output units. The layers are fully interconnected with having weights on them.



Learning Vector Quantization

Parameters Used

Following are the parameters used in LVQ training process as well as in the flowchart

- **x** = training vector $(x_1,...,x_i,...,x_n)$
- **T** = class for training vector **x**
- $\mathbf{w_j}$ = weight vector for $\mathbf{j^{th}}$ output unit
- $\mathbf{C_j}$ = class associated with the $\mathbf{j^{th}}$ output unit

Learning Vector Quantization

Training Algorithm

Step 1 – Initialize reference vectors, which can be done as follows –

- **Step 1** a – From the given set of training vectors, take the first "**m**" $numberofclusters$ training vectors and use them as weight vectors. The remaining vectors can be used for training.
- **Step 1** b – Assign the initial weight and classification randomly.
- **Step 1** c – Apply K-means clustering method.

Learning Vector Quantization

Step 2 – Initialize learning rate α

Step 3 – Continue with steps 4-9, if the condition for stopping this algorithm is not met.

Step 4 – Follow steps 5-6 for every training input vector **x**.

Step 5 – Calculate Square of Euclidean Distance for **j = 1 to m** and **i = 1 to n**

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

Learning Vector Quantization

Step 6 – Obtain the winning unit **J** where **D** j is minimum.

Step 7 – Calculate the new weight of the winning unit by the following relation –

if **T = C$_j$** then $w_j(new) = w_j(old) + \alpha[x - w_j(old)]$

if **T ≠ C$_j$** then $w_j(new) = w_j(old) - \alpha[x - w_j(old)]$

Step 8 – Reduce the learning rate α .

Step 9 – Test for the stopping condition. It may be as follows –

- Maximum number of epochs reached.
- Learning rate reduced to a negligible value.

Kohonen Self-Organizing Maps (SOM)

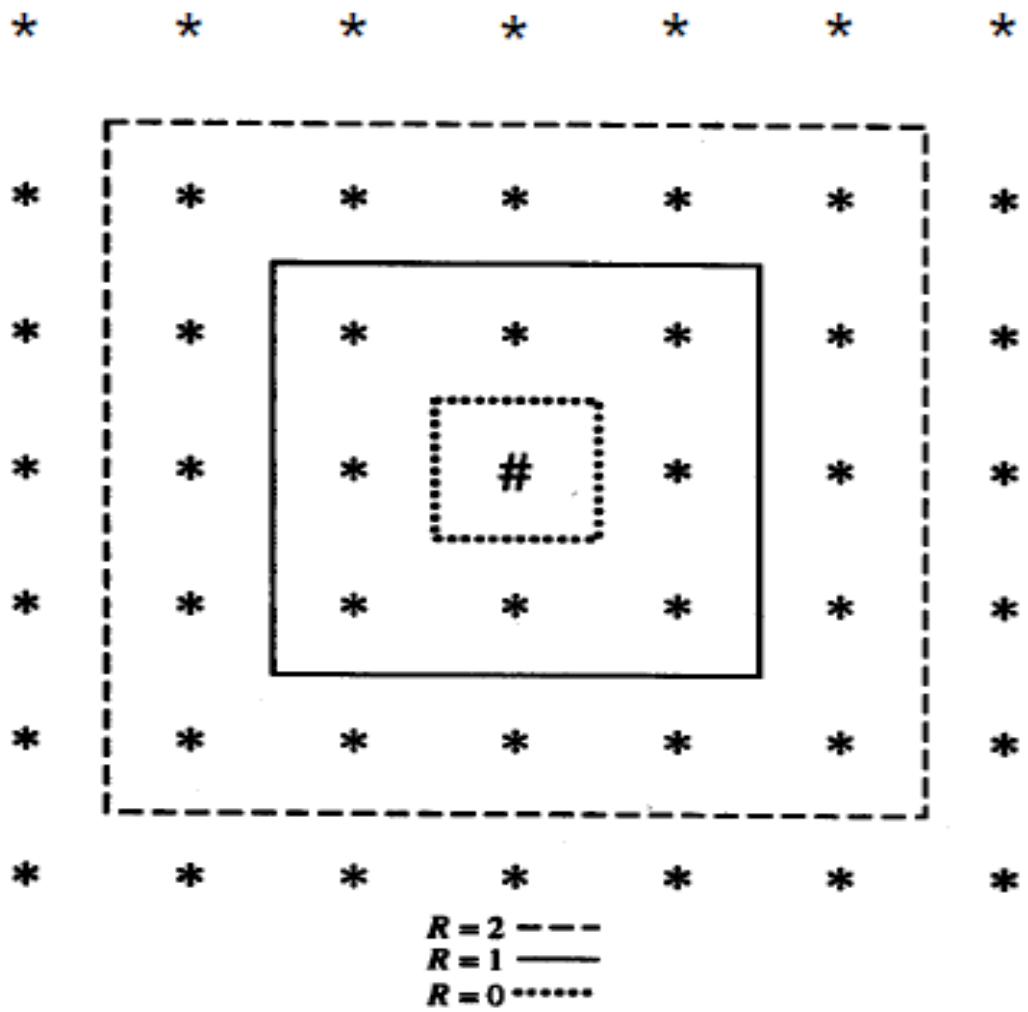
- The SOM neural networks also called topology preserving maps, assume a topological structure among the cluster units.
- The weight vector for a cluster unit serves as an exemplar of the input patterns associated with that cluster.
- During the self-organization process, the cluster unit whose weight 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.

Kohonen Self-Organizing Maps (SOM)

- 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:

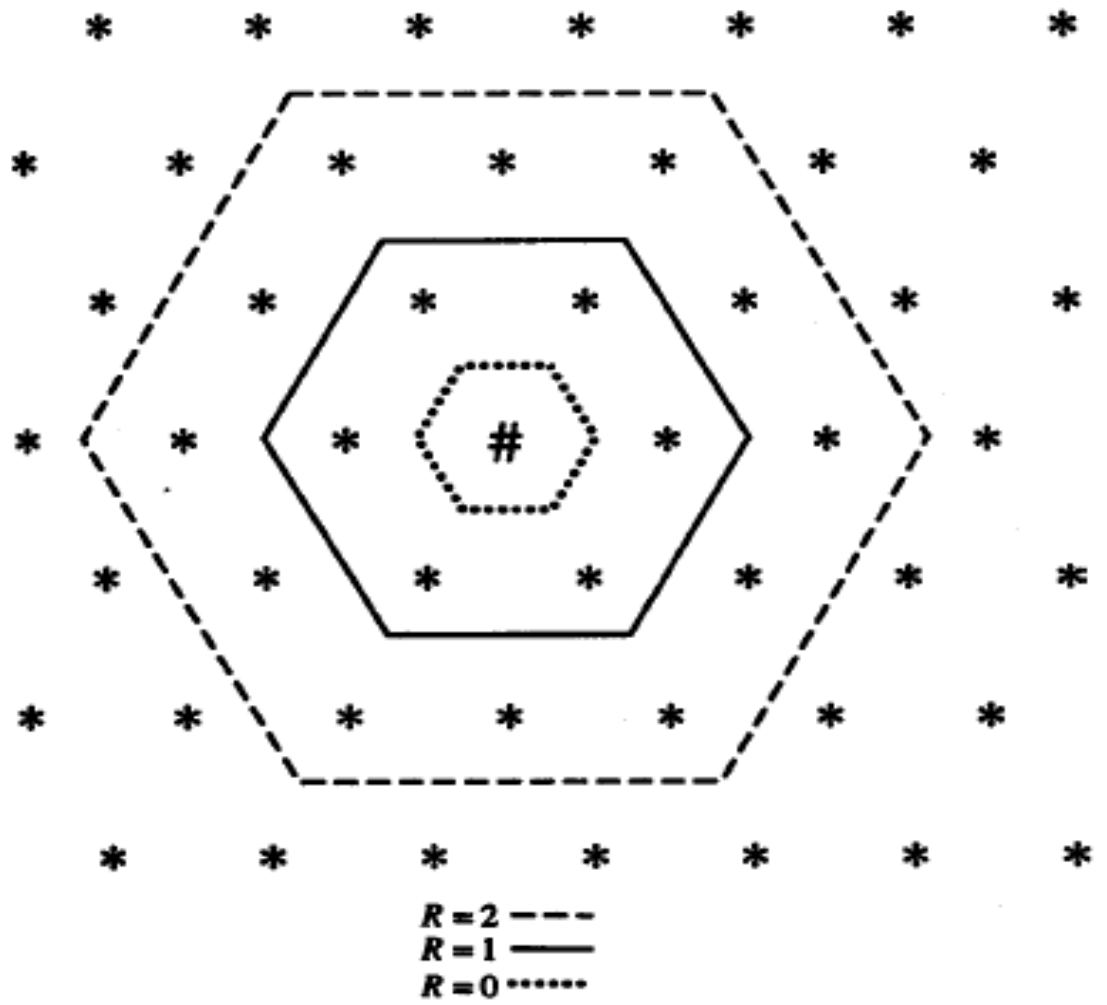
$$\max(1, J - R) \leq j \leq \min(J + R, m))$$

- Neighborhoods for rectangular grid



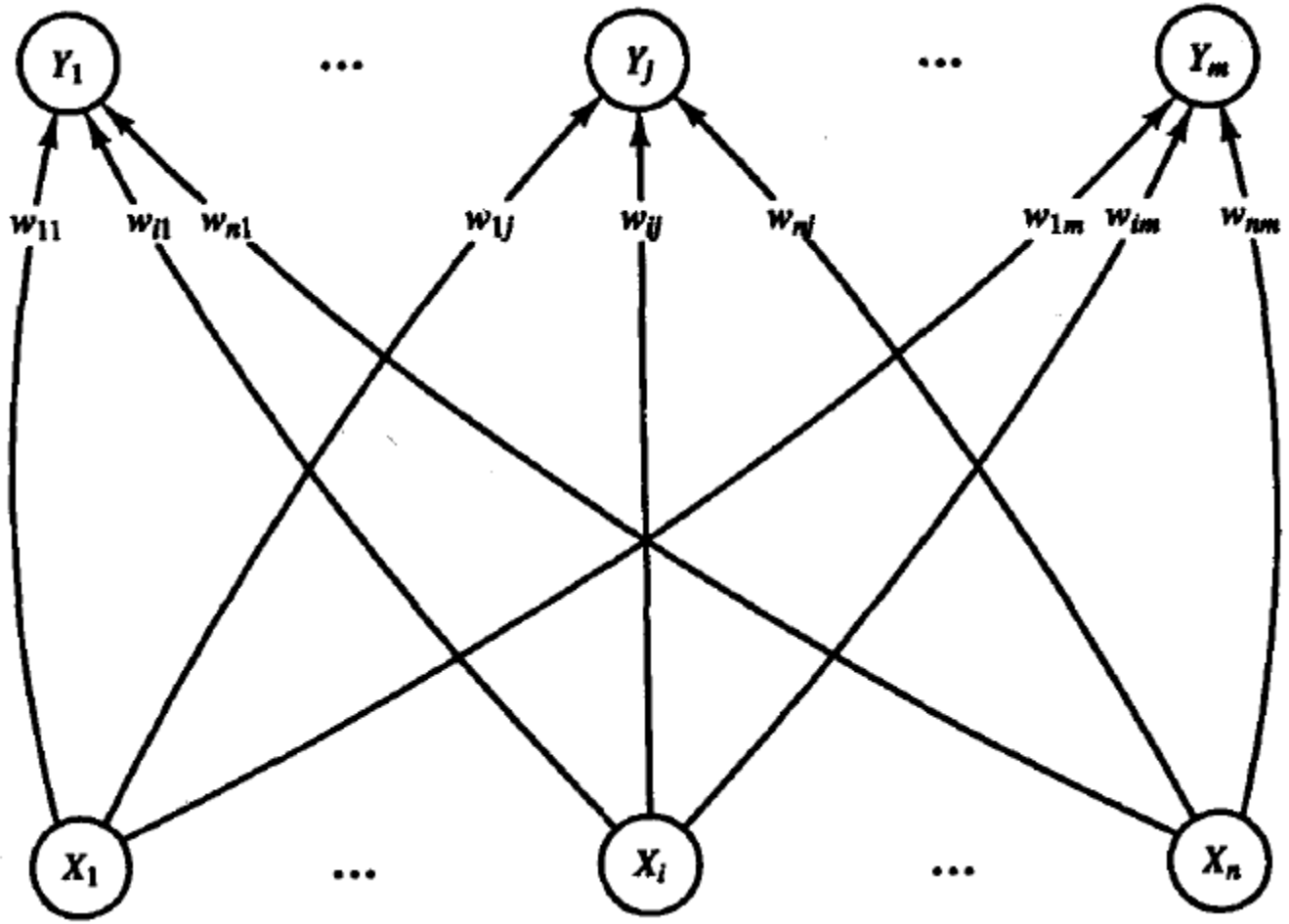
Kohonen Self-Organizing Maps (SOM)

- Neighborhoods for hexagonal grid



Kohonen Self-Organizing Maps (SOM)

- **Architecture**



Kohonen Self-Organizing Maps (SOM)

- **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.

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

Step 8. Test stopping condition.

Kohonen Self-Organizing Maps (SOM)

- **Example: 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)*.

Kohonen Self-Organizing Maps (SOM)

Step 0. Initial weight matrix:

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

Initial radius:

$$\mathbf{R} = \mathbf{0}.$$

Initial learning rate:

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

Kohonen Self-Organizing Maps (SOM)

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

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

Kohonen Self-Organizing Maps (SOM)

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

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

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

Kohonen Self-Organizing Maps (SOM)

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:

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

Kohonen Self-Organizing Maps (SOM)

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:

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

Kohonen Self-Organizing Maps (SOM)

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

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