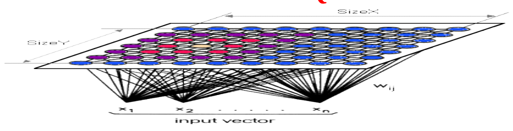
Dr. Qadri Hamarsheh



Neural Networks and Fuzzy Logic (630514)
Lecture 15

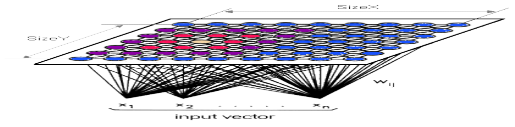
Self-Organizing Maps (Kohonen Maps)

Competitive learning:

- In competitive learning, neurons compete among themselves to be activated.
- While in Hebbian learning, several output neurons can be activated simultaneously, in competitive learning, only a single output neuron is active at any time.
- The output neuron that wins the "**competition**" is called the **winner-takes-all neuron**.

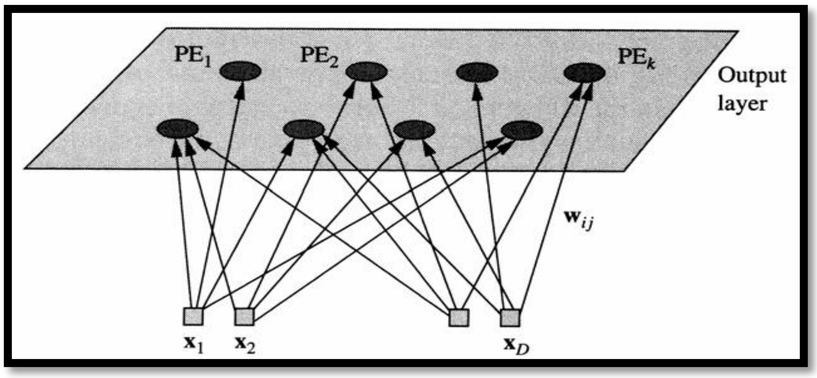
Self-organizing feature maps

- In the late 1980s, Teuvo Kohonen introduced a special class of artificial neural networks called **self-organising feature maps**. These maps are based on competitive learning.
- A Self-Organizing Feature Map (**SOM**) is a type of artificial neural network that is trained using **unsupervised** learning to produce a two-dimensional discretized representation of the input space of the training samples, called a **map**. These maps are useful for **classification** and **visualizing** low-dimensional views of high-dimensional data.
- **Self-Organizing Maps** (SOMs) is particularly similar to biological systems. In the human cortex, multi-dimensional sensory input spaces (e.g., visual input, tactile input) are represented by two-dimensional maps. The projection from sensory inputs onto such maps is **topology conserving**. This means that neighboring areas in these maps represent neighboring areas in the sensory input space. For example, neighboring areas in the sensory cortex are responsible for the arm and hand regions. Such topology-conserving mapping can be achieved by SOMs.
- The cortex is a self-organizing computational map in the human brain.
- Typically, **SOMs** have – like our **brain** – the task to map a high-dimensional input (N dimensions) onto areas in a low-dimensional grid of cells (G dimensions) to draw a map of the high-dimensional space. SOM is a visualization method to represent higher dimensional data in an usually **1-D**, **2-D** or **3-D** manner.
- Kohonen's SOM is called a **topology-preserving map** because there is a topological structure imposed on the nodes in the network. A topological map is simply a mapping that preserves neighborhood relations. The Kohonen map performs a mapping from a continuous input space to a discrete output space, preserving the topological properties of the input.

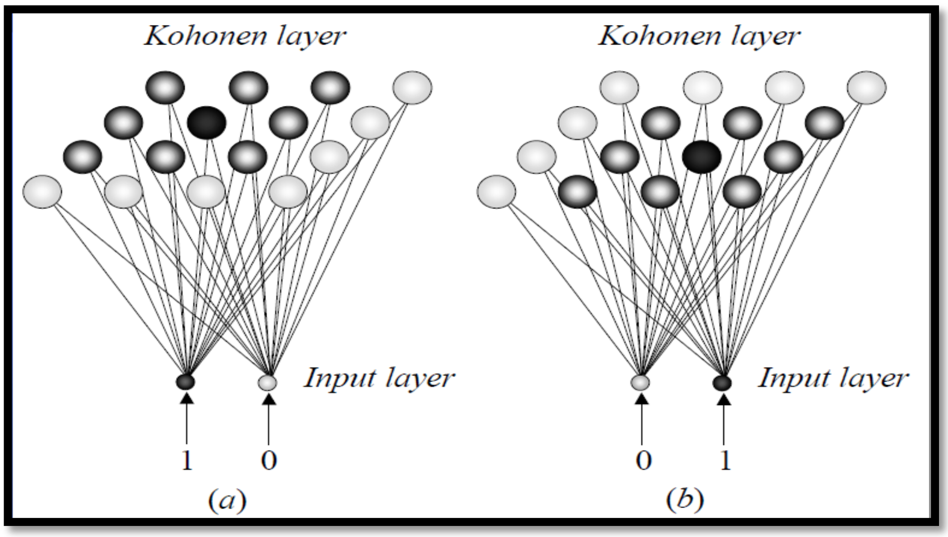


This means that points close to each other in the input space are mapped to the same neighboring neurons in the output space.

- SOMs have two phases:
  - **Learning phase**: **map is built**; network organizes using a competitive process using training set.
  - **Prediction phase**: new vectors are quickly given a location on the converged map, easily classifying or categorizing the new data.

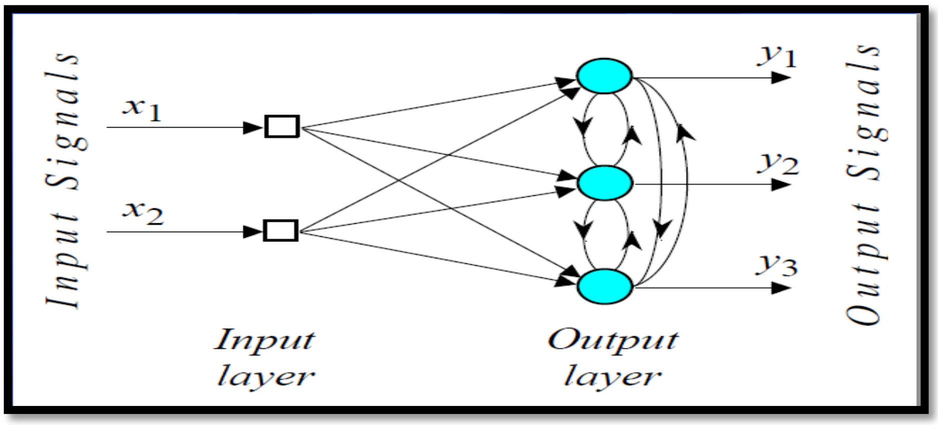


Architecture of a SOM with a 2-D output with n dimensional input vector

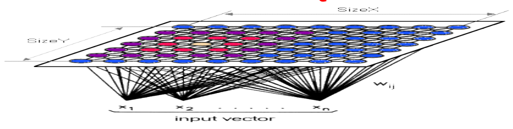


Feature-mapping Kohonen model

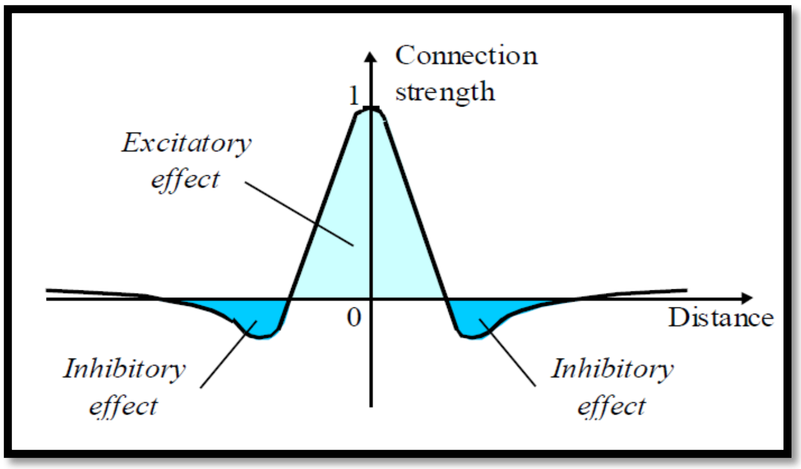
Architecture of the Kohonen Network:



- Two layers: input layer and output (map) layer
- Input and output layers are completely connected.



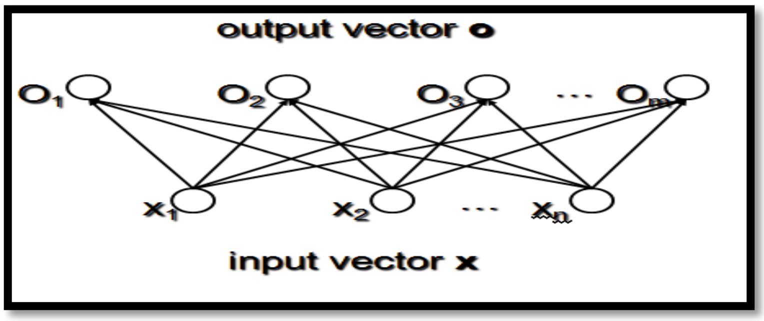
- Output neurons are interconnected within a defined neighborhood -Intra-layer ("**lateral**") connections:
  - Within output layer.
  - Defined according to some topology.
  - No weight between these connections, but used in algorithm for updating weights.
  - The lateral connections are used to create a competition between neurons.
  - The lateral feedback connections produce excitatory or inhibitory effects, depending on the distance from the winning neuron.
  - This is achieved by the use of a **Mexican hat function** which describes synaptic weights between neurons in the Kohonen layer.



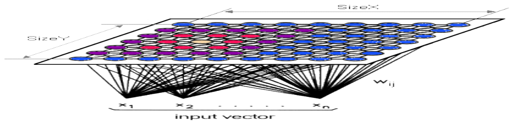
The Mexican hat function of lateral connection

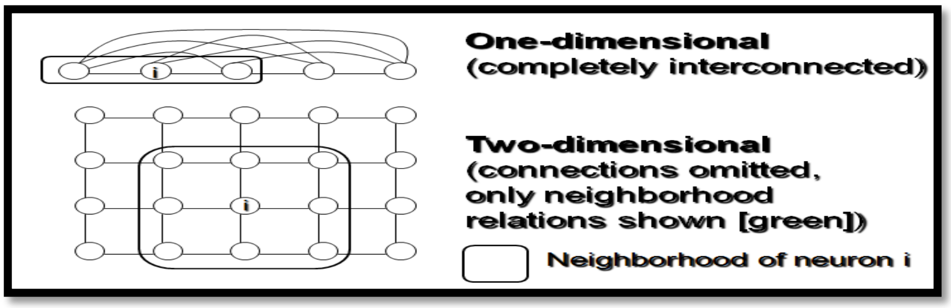
- A topology (neighborhood relation) is defined on the output layer.

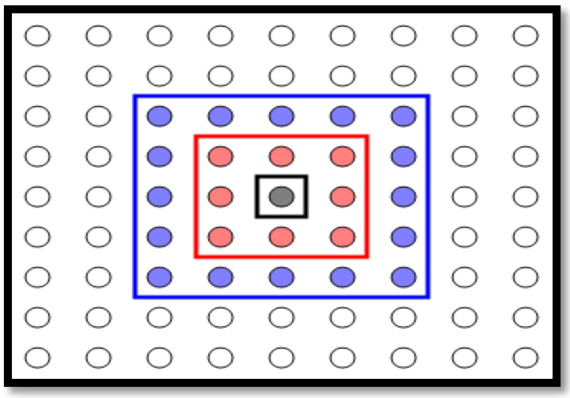
Network structure:



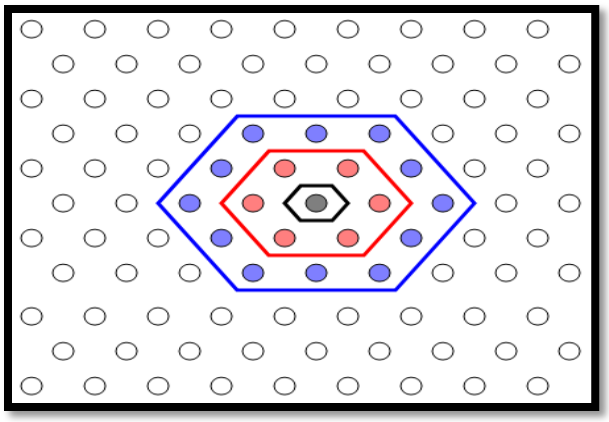
- Common output-layer structures (**neighborhood types**):
  - You can use a one-dimensional arrangement, two or more dimensions. For a one-dimensional SOM, a neuron has only two neighbors within a radius of 1 (or a single neighbor if the neuron is at the end of the line).
  - You can also define distance in different ways, for instance, by using rectangular and hexagonal arrangements of neurons and neighborhoods.




i
i
One-dimensional (completely interconnected)
Two-dimensional (connections omitted, only neighborhood relations shown [green])
Neighborhood of neuron i

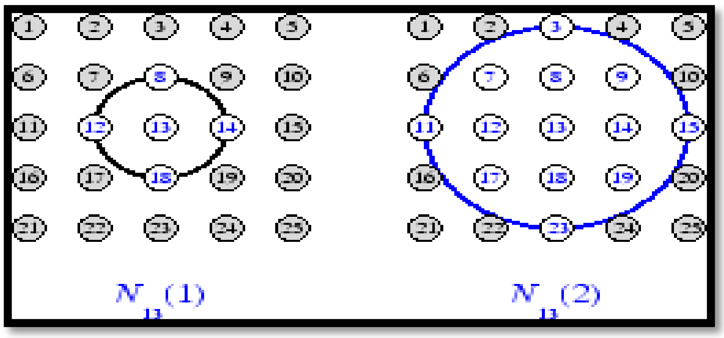


Neighborhoods (R) for a rectangular matrix of cluster units: R = 0 in black brackets, R = 1 in red, and R = 2 in blue.



Neighborhoods (R) for a hexagonal matrix of cluster units: R = 0 in black brackets, R = 1 in red, and R = 2 in blue.

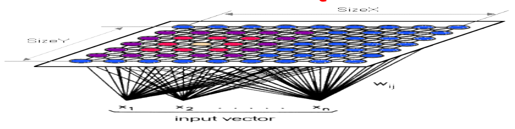
- To illustrate the concept of neighborhoods, consider the figure below. The left diagram shows a two-dimensional neighborhood of radius **d = 1** around neuron **13**. The right diagram shows a neighborhood of radius **d = 2**.



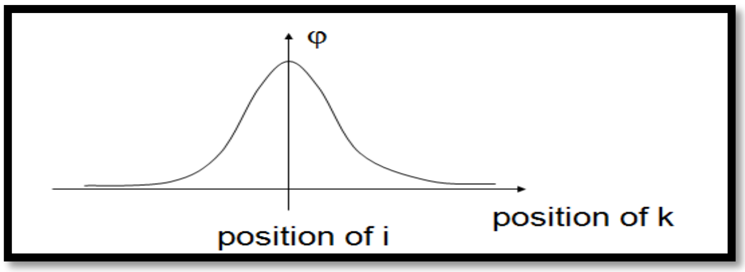
  - These neighborhoods could be written as

$N_{13}(1) = \{8, 12, 13, 14, 18\}$

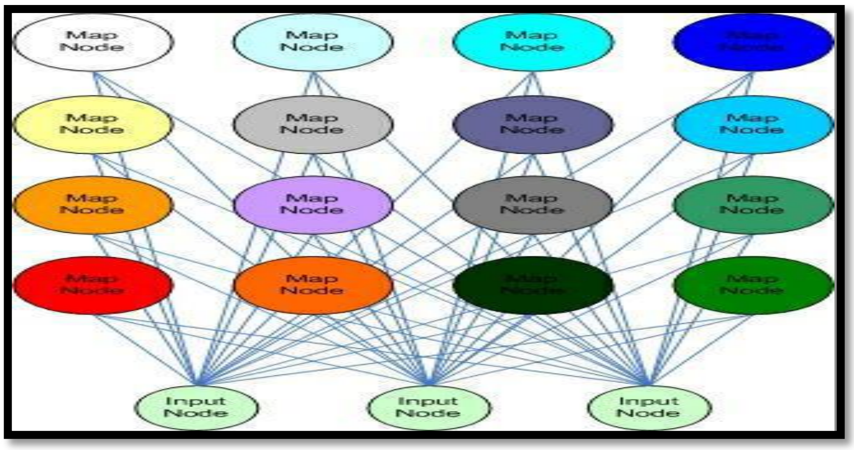
$N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\}$.



- A neighborhood function $\phi(i,k)$ indicates how closely neurons **i** and **k** in the output layer are connected to each other. Usually, a **Gaussian function** on the distance between the two neurons in the layer is used:



- **Example of SOM (3 input vectors and 2-d matrix with 4x4 nodes (clusters)):**



- In the Kohonen network, a neuron learns by **shifting** its weights from inactive connections to active ones. Only the **winning neuron and its neighborhood** are allowed to learn. If a neuron does not respond to a given input pattern, then learning cannot occur in that particular neuron.
- The **competitive learning rule** defines the change Δw_{ij} applied to synaptic weight w_{ij} as:

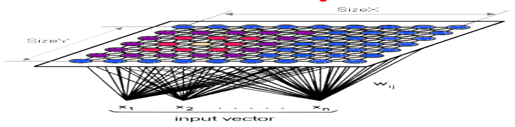
$$\Delta w_{ij} = \begin{cases} \alpha\,(x_i - w_{ij}), & \text{if neuron } j \text{ wins the competition} \\ 0, & \text{if neuron } j \text{ loses the competition} \end{cases}$$

where x_i is the input signal and α is the learning rate parameter.

- The overall effect of the competitive learning rule resides in moving the synaptic weight vector **Wj** of the winning neuron **j** towards the input pattern **X**. The matching criterion is equivalent to the **minimum Euclidean distance** between vectors.
- The **Euclidean distance** between a pair of **n-by-1** vectors **X** and **Wj** is defined by

$$d = \left\| \mathbf{X} - \mathbf{W}_j \right\| = \left[\sum_{i=1}^{n} (x_i - w_{ij})^2 \right]^{1/2}$$

where $\mathbf{x_i}$ and $\mathbf{w_{ij}}$ are the ith elements of the vectors **X** and **Wj**, respectively.



- To identify the winning neuron, $\mathbf{j_X}$, that best matches the input vector $\mathbf{X}$, we may apply the following condition:

$$\boxed{j_{\mathbf{X}} = \min_{j} \left\| \mathbf{X} - \mathbf{W}_j \right\|, \qquad j = 1, 2, \ldots, m}$$

where **m** is the number of neurons in the Kohonen layer.

The SOM Algorithm

The Self-Organizing Map algorithm can be broken up into 9 steps.

Step 0:

- Initialize synaptic weights $\mathbf{w_{ij}}$ to small random values, say in an interval [0, 1].
- Set topological neighborhood parameters.
- Set learning rate parameters (small positive value).

Step 1: While stopping condition is false, do **Steps 2-8**.

Step 2: For each input vector **x** chosen at random from the set of training data and **presented** to the network, do Steps 3-5.

Step 3: For each **j,** compute:

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

Euclidean distance is a measurement of **similarity** between two sets of data. (Every node in the network is examined to calculate which ones' weights are most like the input vector).

Step 4: Find index **J** such that **D(J)** is a minimum (The winning node is commonly known as the **Best Matching Unit** (**BMU**)).

$$\boxed{j_{\mathbf{X}}(p) = \min_{j} \left\| \mathbf{X} - \mathbf{W}_j(p) \right\| = \left\{ \sum_{i=1}^{n} [x_i - w_{ij}(p)]^2 \right\}^{1/2}, \quad j = 1, 2, \ldots, m}$$

Where: **n** is the number of neurons in the input layer,
m is the number of neurons in the Kohonen layer.

Step 5: Learning (Update the synaptic weights):

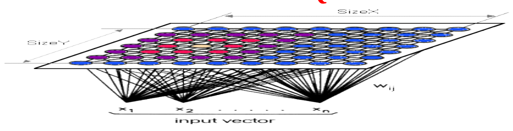
For all units **j** within a specified neighborhood of **J**, and for all **i**:

$$\boxed{w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)}$$

Where $\Delta \mathbf{w_{ij}}$ (p) is the **weight correction** at iteration **p**.

The weight correction is determined by the competitive learning rule:

$$\boxed{\Delta w_{ij}(p) = \begin{cases} \alpha\left[x_i - w_{ij}(p)\right], & j \in \Lambda_j(p) \\ 0, & j \notin \Lambda_j(p) \end{cases}}$$



Where α is the learning rate parameter, and $\Lambda_j(p)$ is the neighborhood function centered around the winner-takes-all neuron j_X at iteration p.

Any nodes found within the radius of the **BMU** are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its' weights are altered.

Step 6: Update learning rate (α is a slowly decreasing function of time).

Step 7: Reduce radius of topological neighborhood at specified times. The **radius** of the neighborhood of the BMU is calculated. This value starts large (typically it is set to be the radius of the network). The radius of the neighborhood around a cluster unit also decreases as the clustering process progresses.

Step 8: Test stopping condition.

Example: Suppose, for instance, that the **2-dimensional** input vector **X** is presented to the **three-neuron** Kohonen network:

$$\mathbf{X} = \begin{bmatrix} 0.52 \\ 0.12 \end{bmatrix}$$

- The initial weight vectors, **Wj**, are given by

$$\mathbf{W}_1 = \begin{bmatrix} 0.27 \\ 0.81 \end{bmatrix} \qquad \mathbf{W}_2 = \begin{bmatrix} 0.42 \\ 0.70 \end{bmatrix} \qquad \mathbf{W}_3 = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix}$$

- We find the winning (**best-matching**) neuron j_X using the minimum-distance Euclidean criterion:

$$d_1 = \sqrt{(x_1 - w_{11})^2 + (x_2 - w_{21})^2} = \sqrt{(0.52-0.27)^2 + (0.12-0.81)^2} = 0.73$$

$$d_2 = \sqrt{(x_1 - w_{12})^2 + (x_2 - w_{22})^2} = \sqrt{(0.52-0.42)^2 + (0.12-0.70)^2} = 0.59$$

$$d_3 = \sqrt{(x_1 - w_{13})^2 + (x_2 - w_{23})^2} = \sqrt{(0.52-0.43)^2 + (0.12-0.21)^2} = 0.13$$

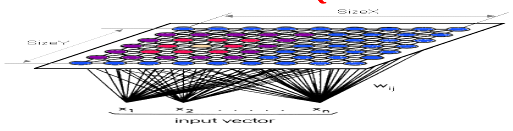
- Neuron **3** is the winner and its weight vector $\mathbf{W}_3$ is updated according to the competitive learning rule.

$$\Delta w_{13} = \alpha\,(x_1 - w_{13}) = 0.1\,(0.52 - 0.43) = 0.01$$

$$\Delta w_{23} = \alpha\,(x_2 - w_{23}) = 0.1\,(0.12 - 0.21) = -0.01$$

- The updated weight vector $\mathbf{W}_3$ at iteration **(p + 1)** is determined as:

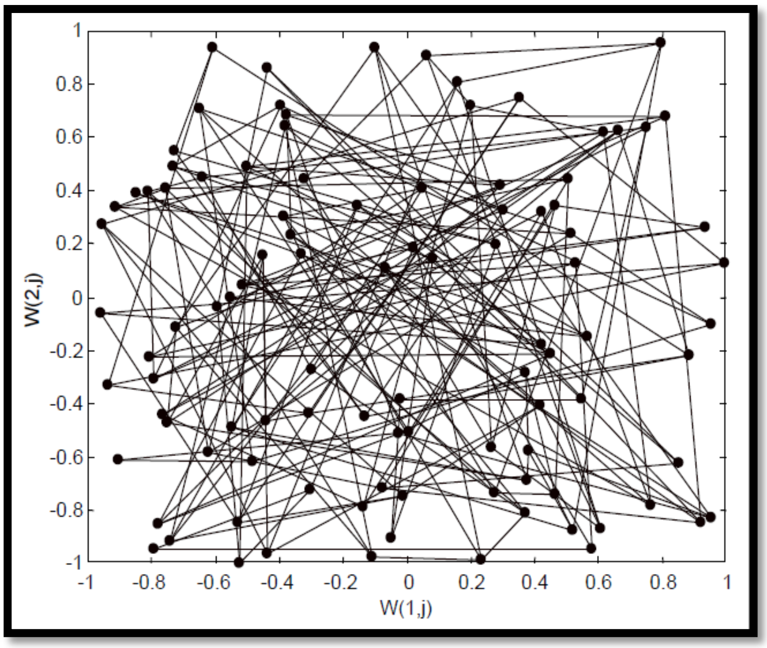
$$\mathbf{W}_3(p+1) = \mathbf{W}_3(p) + \Delta\mathbf{W}_3(p) = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} = \begin{bmatrix} 0.44 \\ 0.20 \end{bmatrix}$$



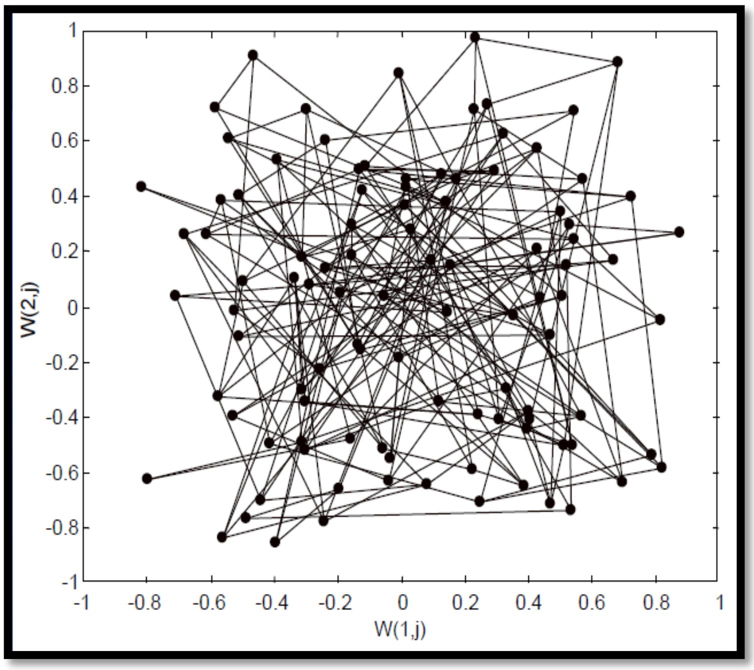
- The weight vector $\mathbf{W_3}$ of the wining neuron **3** becomes closer to the input vector $\mathbf{X}$ with each iteration.

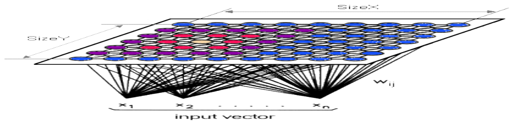
Competitive learning in the Kohonen network

- To illustrate competitive learning, consider the Kohonen network with 100 neurons arranged in the form of a two-dimensional lattice with 10 rows and 10 columns. The network is required to classify two-dimensional input vectors - each neuron in the network should respond only to the input vectors occurring in its region.
- The network is trained with 1000 two-dimensional input vectors generated randomly in a square region in the interval between –1 and +1. The learning rate parameter α is equal to 0.1.
- Initial random weights

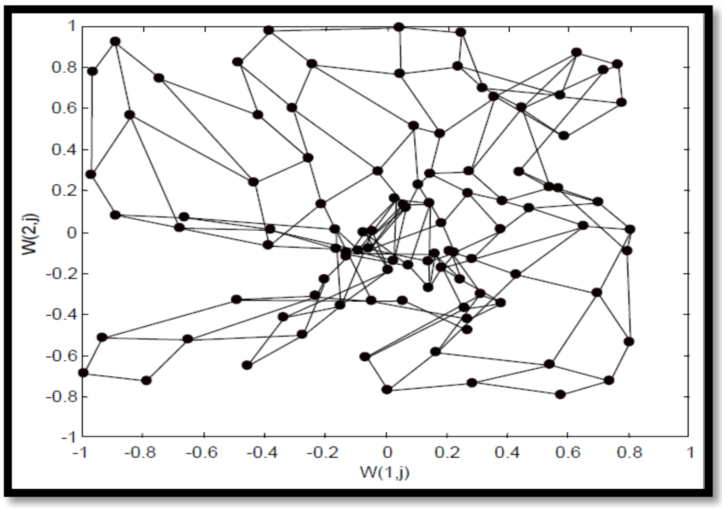


- Network after 100 iterations

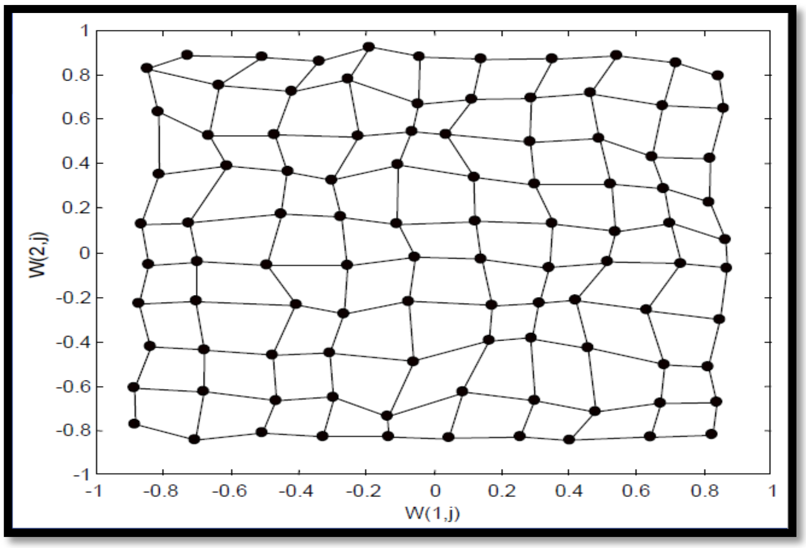




- Network after 1000 iterations



- Network after 10,000 iterations



- To see how neurons respond, apply the following input vectors:

$$X_1 = \begin{bmatrix} 0.2 \\ 0.9 \end{bmatrix}, X_2 = \begin{bmatrix} 0.6 \\ -0.2 \end{bmatrix}, X_3 = \begin{bmatrix} -0.7 \\ -0.8 \end{bmatrix},$$

  - Neuron **6** respond to the input vector X_1.
  - Neuron **69** respond to the input vector X_2.
  - Neuron **92** respond to the input vector X_3.