UNSUPERVISED LEARNING NETWORKS

- No help from the outside.
- No training data, no information available on the desired output.
- Learning by doing.
- Used to pick out structure in the input:
  - Clustering,
  - Reduction of dimensionality → compression.
- Example: Kohonen's Learning Law.

There exists several networks under this category, such as

- Max Net,
- Mexican Hat,
- Kohonen Self-organizing Feature Maps,
- Learning Vector Quantization,
- Counterpropagation Networks,
- Hamming Network,
- Adaptive Resonance Theory.

COMPETITIVE LEARNING

- Output units compete, so that eventually only one neuron (the one with the most input) is active in response to each output pattern.

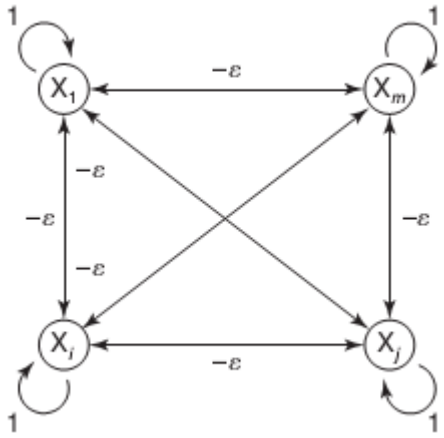
- The total weight from the input layer to each output neuron is limited. If some connections are strengthened, others must be weakened.

- A consequence is that the winner is the output neuron whose weights best match the activation pattern.

MAX NET

- Max Net is a fixed weight competitive net.

- Max Net serves as a subnet for picking the node whose input is larger. All the nodes present in this subnet are fully interconnected and there exist symmetrical weights in all these weighted interconnections.

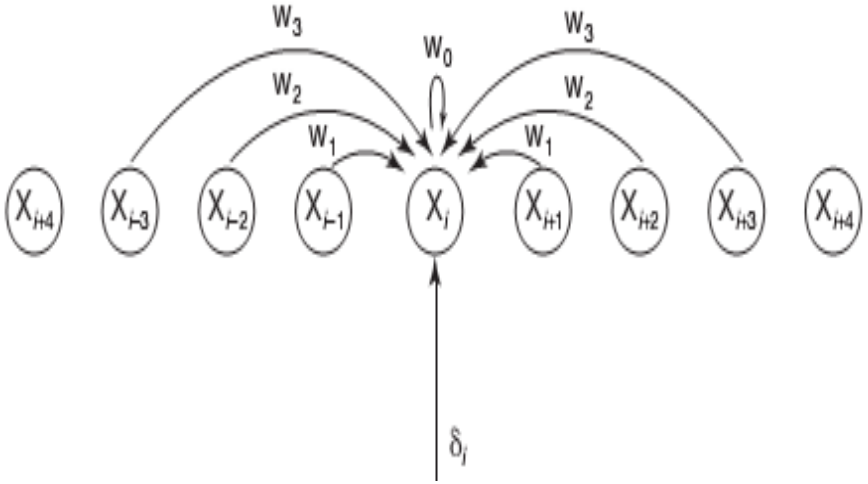


- The weights between the neurons are inhibitory and fixed.

- The architecture of this net is

MEXICAN HAT NETWORK

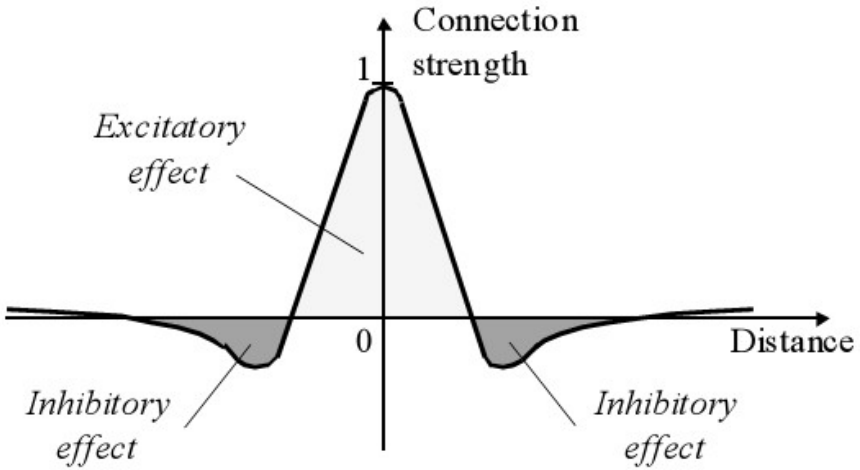
- Kohonen developed the Mexican hat network which is a more generalized contrast enhancement network compared to the earlier Max Net.
- Here, in addition to the connections within a particular layer of neural net, the neurons also receive some other external signals. This interconnection pattern is repeated for several other neurons in the layer.The architecture for the network is as shown below:



MEXICAN HAT NETWORK

- The lateral connections are used to create a competition between neurons. The neuron with the largest activation level among all neurons in the output layer becomes the winner. This neuron is the only neuron that produces an output signal. The activity of all other neurons is suppressed in the competition.

- 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 ne



MEXICAN HAT FUNCTION OF LATERAL CONNECTION

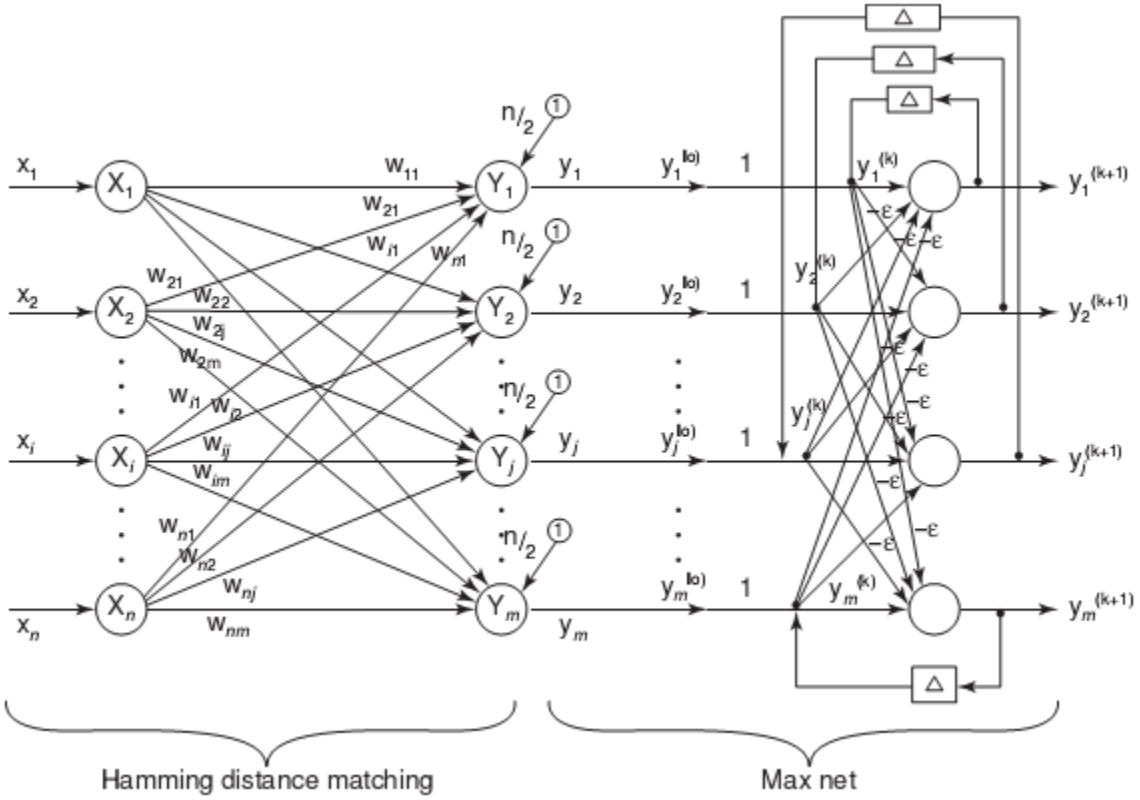
HAMMING NETWORK

- The Hamming network selects stored classes, which are at a maximum Hamming distance (H) from the noisy vector presented at the input.

- The Hamming distance between the two vectors is the number of components in which the vectors differ.

- The Hamming network consists of two layers.
  - The first layer computes the difference between the total number of components and Hamming distance between the input vector x and the stored pattern of vectors in the feed forward path.
  - The second layer of the Hamming network is composed of Max Net (used as a subnet) or a winner-take-all network which is a recurrent network.

ARCHITECTURE OF HAMMING NET



SELF-ORGANIZATION

- Network Organization is fundamental to the brain
  - Functional structure.
  - Layered structure.
  - Both parallel processing and serial processing require organization of the brain.

SELF-ORGANIZING FEATURE MAP

Our brain is dominated by the cerebral cortex, a very complex structure of billions of neurons and hundreds of billions of synapses. The cortex includes areas that are responsible for different human activities (motor, visual, auditory, etc.) and associated with different sensory inputs. One can say that each sensory input is mapped into a corresponding area of the cerebral cortex. ***The cortex is a self-organizing computational map in the human brain.***

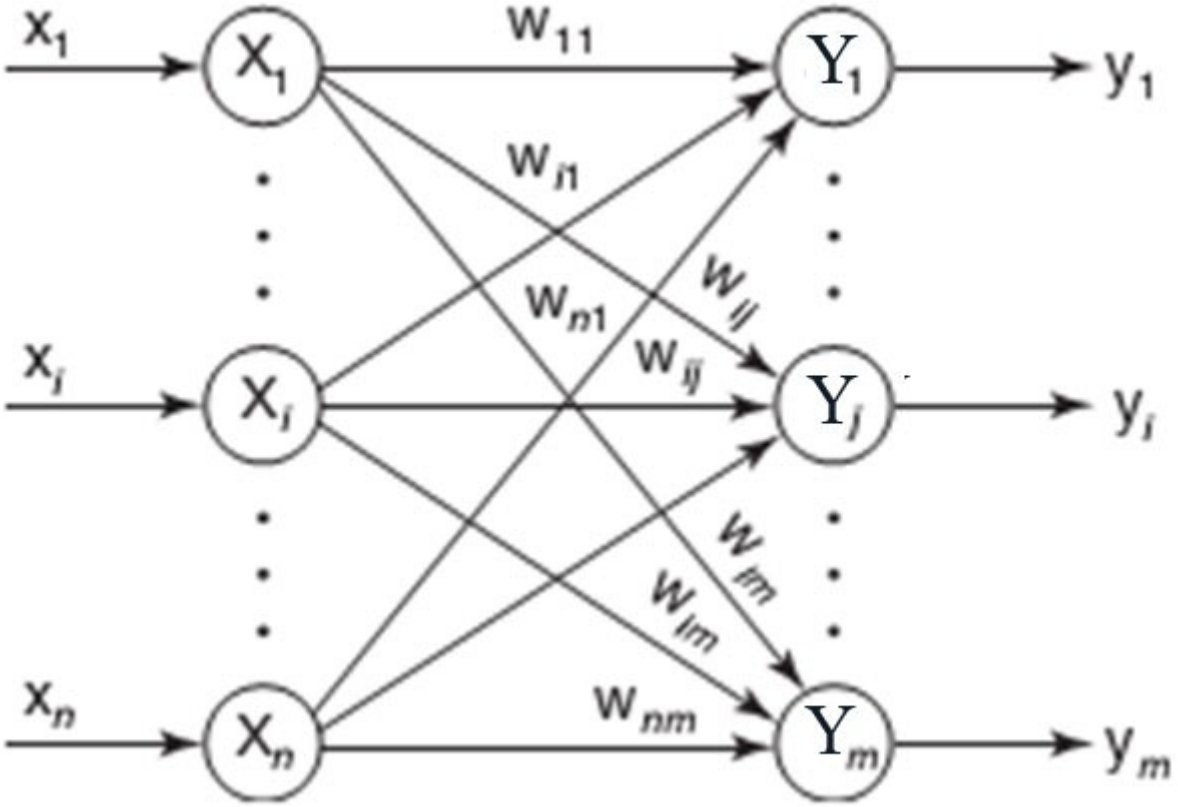
SELF-ORGANIZING NETWORKS

- Discover significant patterns or features in the input data.
- Discovery is done without a teacher.
- Synaptic weights are changed according to local rules.
- The changes affect a neuron's immediate environment until a final configuration develops.

KOHONEN SELF-ORGANIZING FEATURE MAP (KSOFM)

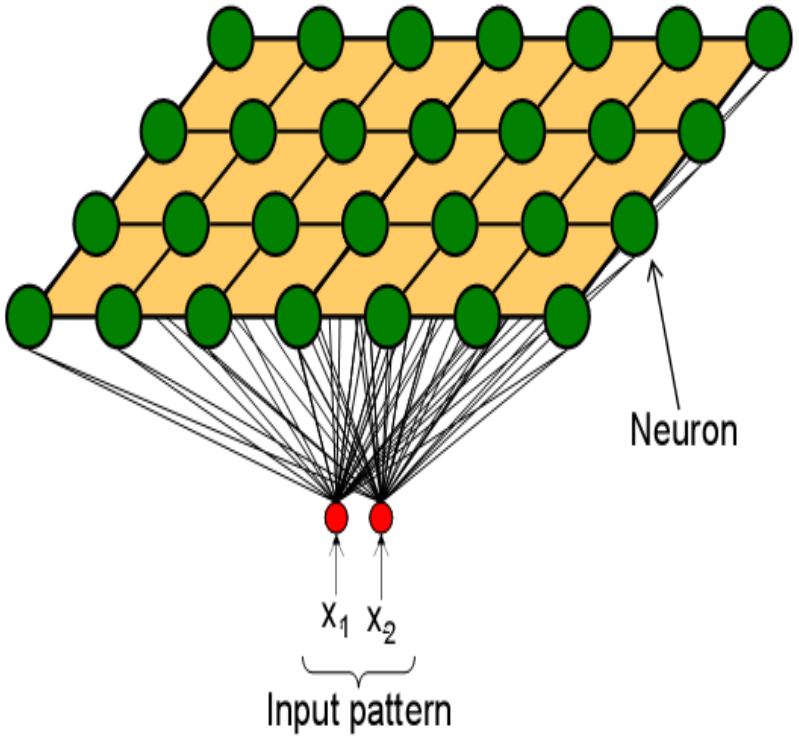
- The Kohonen model provides a topological mapping.
- It places a fixed number of input patterns from the input layer into a higher dimensional output or Kohonen layer.
- Training in the Kohonen network begins with the winner's neighborhood of a fairly large size. Then, as training proceeds, the neighborhood size gradually decreases.
- Kohonen SOMs result from the synergy of three basic processes
  - Competition,
  - Cooperation,
  - Adaptation.

ARCHITECTURE OF KSOFM



COMPETITION OF KSOFM

- Each neuron in an SOM is assigned a weight vector with the same dimensionality N as the input space.
- Any given input pattern is compared to the weight vector of each neuron and the closest neuron is declared the winner.
- The Euclidean norm is commonly used to measure distance.



CO-OPERATION OF KSOFM

- The activation of the winning neuron is spread to neurons in its immediate neighborhood.
- This allows topologically close neurons to become sensitive to similar patterns.

- The winner's neighborhood is determined on the lattice topology.
  - Distance in the lattice is a function of the number of lateral connections to the winner.

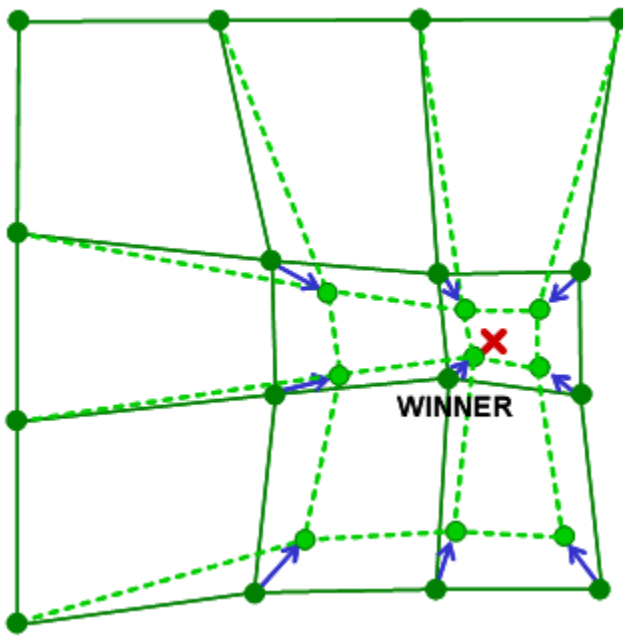
- The size of the neighborhood is initially large, but shrinks over time.
  - An initially large neighborhood promotes a topology-preserving mapping.
  - Smaller neighborhoods allow neurons to specialize in the latter stages of training.

ADAPTATION OF KSOFM

During training, the winner neuron and its topological neighbors are adapted to make their weight vectors more similar to the input pattern that caused the activation.

Neurons that are closer to the winner will adapt more heavily than neurons that are further away.

The magnitude of the adaptation is controlled with a learning rate, which decays over time to ensure convergence of the SOM.



KSOFM ALGORITHM

Step 1: *Initialization*

Set initial synaptic weights to small random values, say in an interval [0, 1], and assign a small positive value to the learning rate parameter α.

Step 2: *Activation and Similarity Matching.*

Activate the Kohonen network by applying the input vector **X**, and find the winner-takes-all (best matching) neuron $j_{\mathbf{X}}$ at iteration p, using the minimum-distance Euclidean criterion

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

where n is the number of neurons in the input layer, and m is the number of neurons in the Kohonen layer.

Step 3: ***Learning.***

Update the synaptic weights

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

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

The weight correction is determined by the competitive learning rule:

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

where α is the *learning rate* parameter, and $\Lambda_j(p)$ is the neighbourhood function centred around the winner-takes-all neuron j_X at iteration p.

Step 4: ***Iteration.***

Increase iteration p by one, go back to Step 2 and continue until the minimum-distance Euclidean criterion is satisfied, or no noticeable changes occur in the feature map.

EXAMPLE OF KSOFM

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, $\mathbf{W}_j$, are given by

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

Find the winning neuron using the Euclidean distance:

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

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

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

Neuron 3 is the winner and its weight vector 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 W3 at iteration (p+1) is determined as:

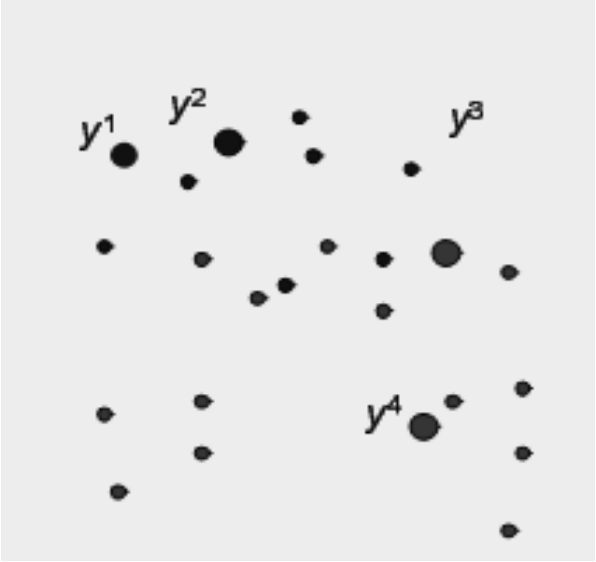
$$\mathbf{W}_3(p+1) = \mathbf{W}_3(p) + \Delta \mathbf{W}_3(p) = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} = \begin{bmatrix} 0.44 \\ 0.20 \end{bmatrix}$$

The weight vector W3 of the winning neuron 3 becomes closer to the input vector X with each iteration.

LEARNING VECTOR QUANTIZATION (LVQ)

- This is a supervised version of vector quantization. Classes are predefined and we have a set of labeled data.

- The goal is to determine a set of prototypes that best represent each class.



BASIC SCHEME OF LVQ

Step 1: Initialize prototype vectors for different classes.

Step 2: Present a single input.

Step 3: Identify the closest prototype, i.e., the so-called winner.

Step 4: Move the winner

- closer toward the data (same class),
- away from the data (different class).

VARIANTS OF LVQ

- LVQ 1
- LVQ 2
- LVQ 2.1
- LVQ 3