Self-Organizing Maps

References:

https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4

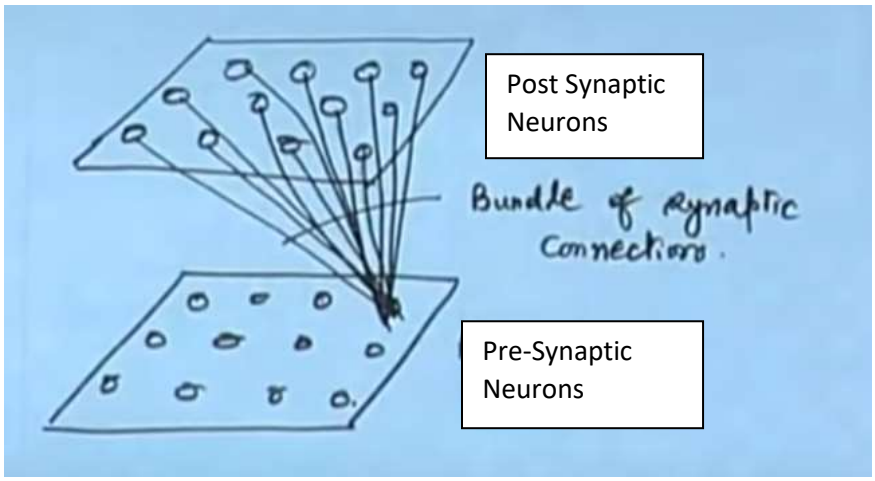
Nptel Lectures By S Sengupta on course: Neural Networks and Applications (Lec 35,36,37)

[https://nptel.ac.in/courses/117/105/117105084/]

=> Research Paper [T. Kohonen, "The self-organizing map," in Proceedings of the IEEE, vol. 78, no. 9, pp. 1464-1480, Sept. 1990, doi: 10.1109/5.58325.]

=> Wikipedia

- **SOM** was introduced by Finnish professor Teuvo Kohonen in the 1980s is sometimes called a **Kohonen map.**
- SOM is a type of ANN trained using unsupervised learning.
- It's a method for dimensionality reduction method producing low dimensional and discretized representation of input creating a map like structure.
- It's different from other ANNs as they apply error reduction techniques (like backpropagation with gradient descent) while SOM applies competitive learning technique where a neighborhood function is used to preserve the topological properties of the input space.
- SOM is similar to constrained K-Means Clustering.
- Output neurons acts in a competitive manner, though this is a biologically inspired fact that the neurons that are close to each other, they tend to excite the neighboring neurons and inhibit the father neurons. (Short Range Excitation and Long Range Inhibition).
- It's Neurobiologically inspired that when we, humans, have different inputs, visual, acoustics etc. they are sent to different parts of cerebral cortex in brain to be processed. It was also believed that there is a topological ordering there as well which inspired SOM.
- There are 2 popular model of SOM : Willshaw Von-der Malsburg model, Kohonen Model.
- Willshaw Von-Der Malsburg Model :

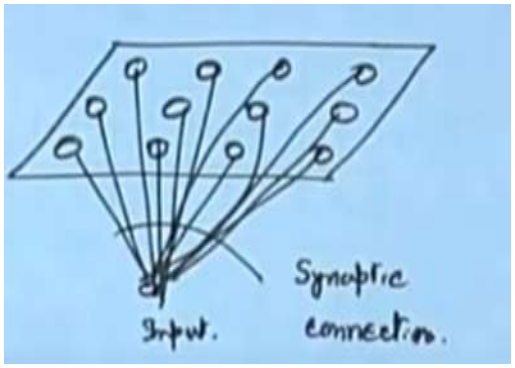


Input dimensions is same as output dimensions.

This model was used to explain the retino-optic mapping from

Electrical Signals of the presynaptic neurons are based on geometric proximities.

- Kohonen Model: this is more generalized model where only output layer is organized, the inputs can be unorganized.



Based on vector coding algorithm

Optimally places fixed number of vectors (codewords) into a higher dimensional input space.

This is used in data compression.

- Essential Processes in the formation of self-organizing map:
  - Competition: Each neuron computes discriminant function and the neuron with largest discriminant wins.
  - Cooperation: The winning neuron will determine the spatial location of topological neighborhood of the exited neurons.
  - Synaptic Adaptation: Enables the excited neurons to increase their individual values of discriminant function in relation to the input pattern.

- Mathematical Modelling of these 3 processes:

STEP 1 : Competition:

Competitive process

m-dimensional input

$$\vec{x} = [x_1 \; x_2 \; \ldots\ldots \; x_m]^T$$

$$\vec{w_j} = [w_{j1} \; w_{j2} \; \ldots\ldots \; w_{jm}]^T \qquad j = 1, 2, \ldots\ldots, l.$$

where l is the total number of output neurons in the network.

Best match between $\vec{x}$ and w_j

- Compute $\vec{w_j}^T \vec{x}$ for $j = 1, 2, \ldots\ldots, l$ and select the largest among this.

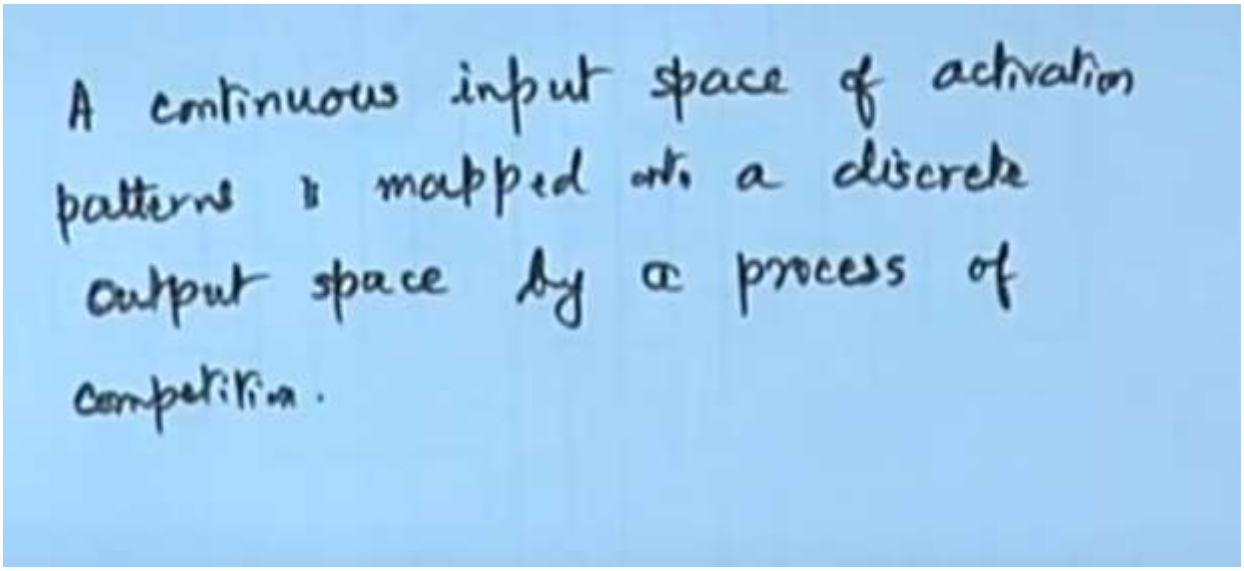
Maximizing $\vec{w_j}^T \vec{x}$

→ minimizing $\| \vec{x} - \vec{w_j} \|$.

Use index $i(\vec{x})$

$$i(\vec{x}) = \arg \min_j \| \vec{x} - \vec{w_j} \|.$$

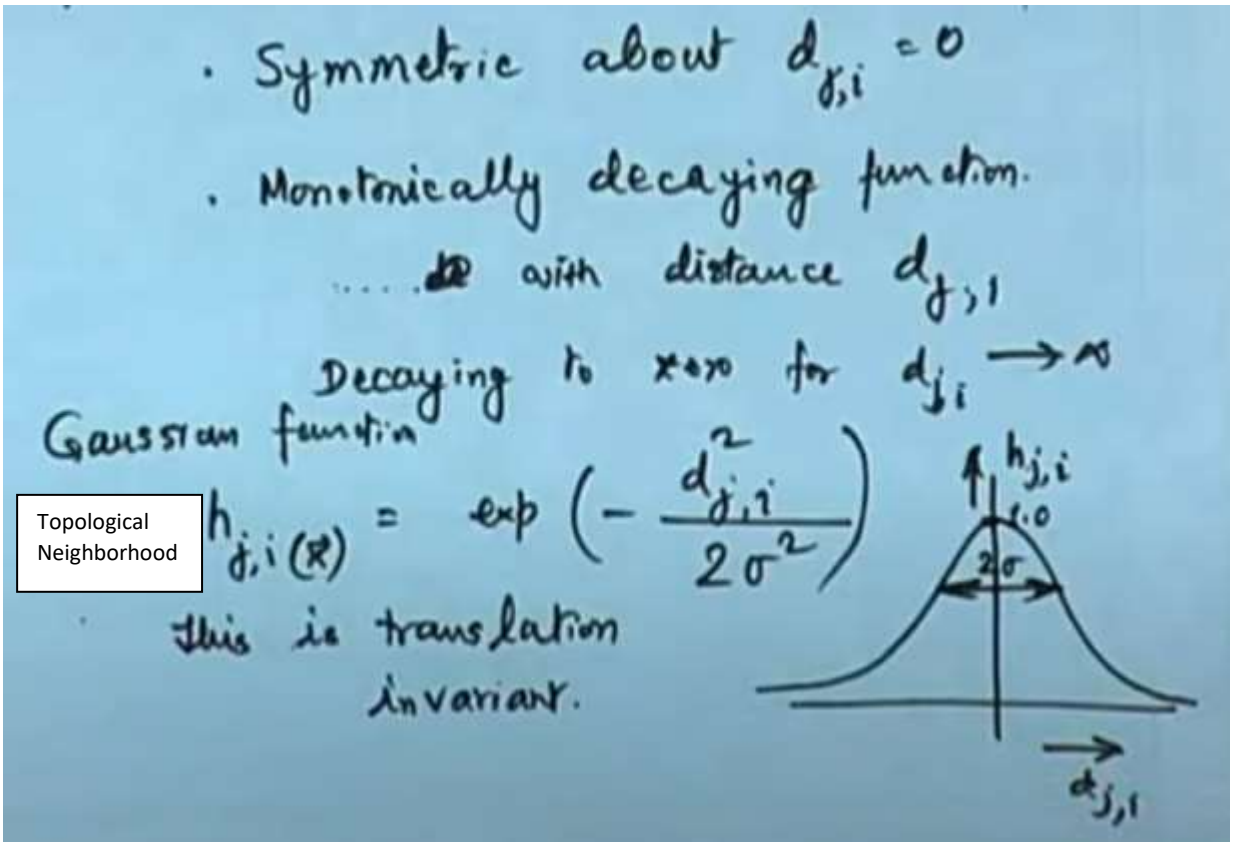
and the corresponding weight vector is $i(\vec{x})$ is the closest weight vector.



Winner of this competition will be the one with max. value of W*X (w* is W transpose) or min. distance between X and Wj vector.

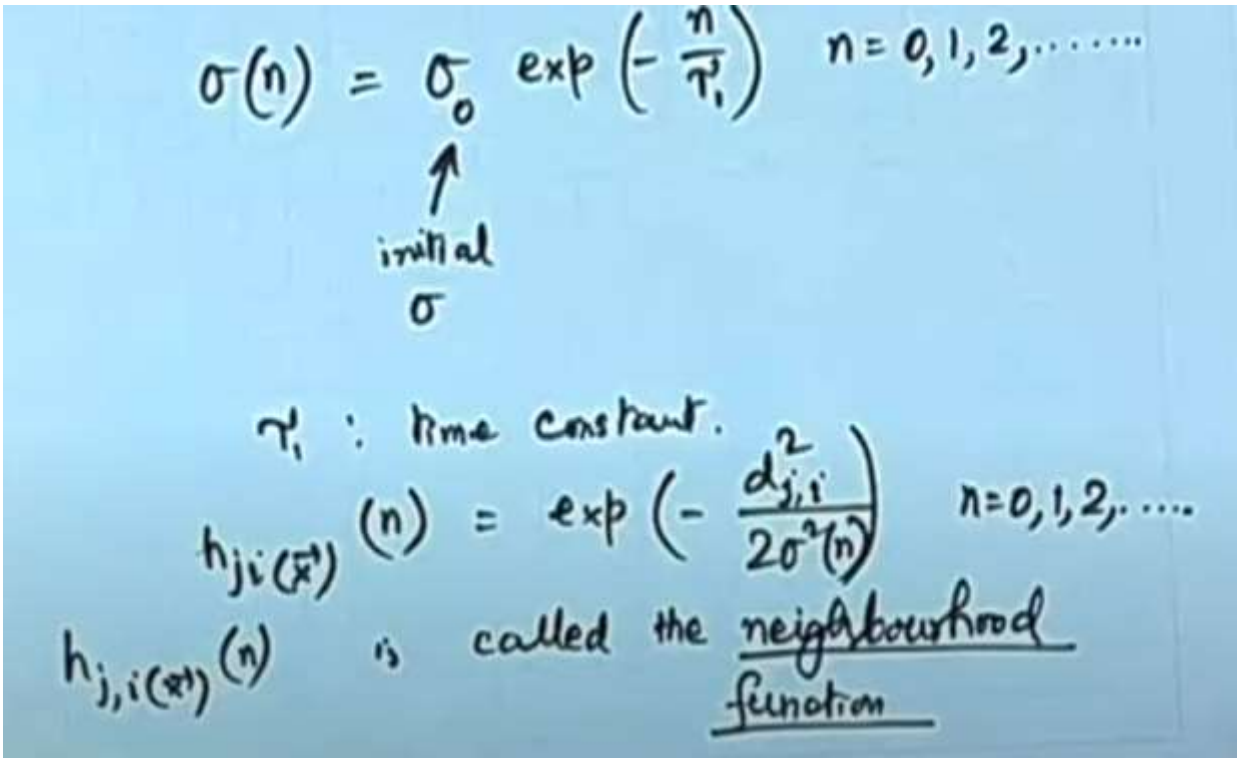
STEP 2: Cooperative Process:

When a neuron is fired then it also excites the neurons in its surrounding. Now remember we talked about neighborhood function, it has to be maximum when its near to winning neuron and it has to be vanishing when going away from the winning neuron.
Hence it should be monotonically decreasing.



Topological Neighborhood

σ shrinks with iterations so as to eliminate neighbors slowly. And with sigma shrinking, hji will also decrease.



STEP 3: Adaptation:

Learning Mech. Employed mostly for SOM is Hebbian learning. Hebbian Learning Rule, also known as Hebb Learning Rule, was proposed by Donald O Hebb. It is one of the first and also easiest learning rules in the neural network. It is used for pattern classification. It is a single layer neural network, i.e. it has one input layer and one output layer. The input layer can have many units, say n. The output layer only has one unit. Hebbian rule works by updating the weights between neurons in the neural network for each training sample.
Now there is a limitation of Hebbian learning which is : If we keep feeding same training pattern then weights will saturate(causing overlearning) and to prevent this we need to introduce a forgetting term, given by g(yj).

$g(y_j)\,\vec{w_j}$: forgetting term in Hebbian hypothesis.

↑ positive scalar function

To simplify,

$$g(y_j) = y_j.$$

$$y_j = h_{j,i}(\vec{x})$$

$$\Delta \vec{w}_j = \eta y_j \vec{x} - g(y_j)\vec{w}_j \quad \ldots\ldots (1)$$

η : learning rate parameter

$$g(y_j) = \eta y_j$$

Then eq. (1) can be rewritten as.

$$\Delta \vec{w}_j = \eta y_j \vec{x} - \eta y_j \vec{w}_j$$

Considering $y_j = h_{j,i(x)}$

$$\Delta \vec{w}_j = \eta h_{j,i(x)} (\vec{x} - \vec{w}_j)$$

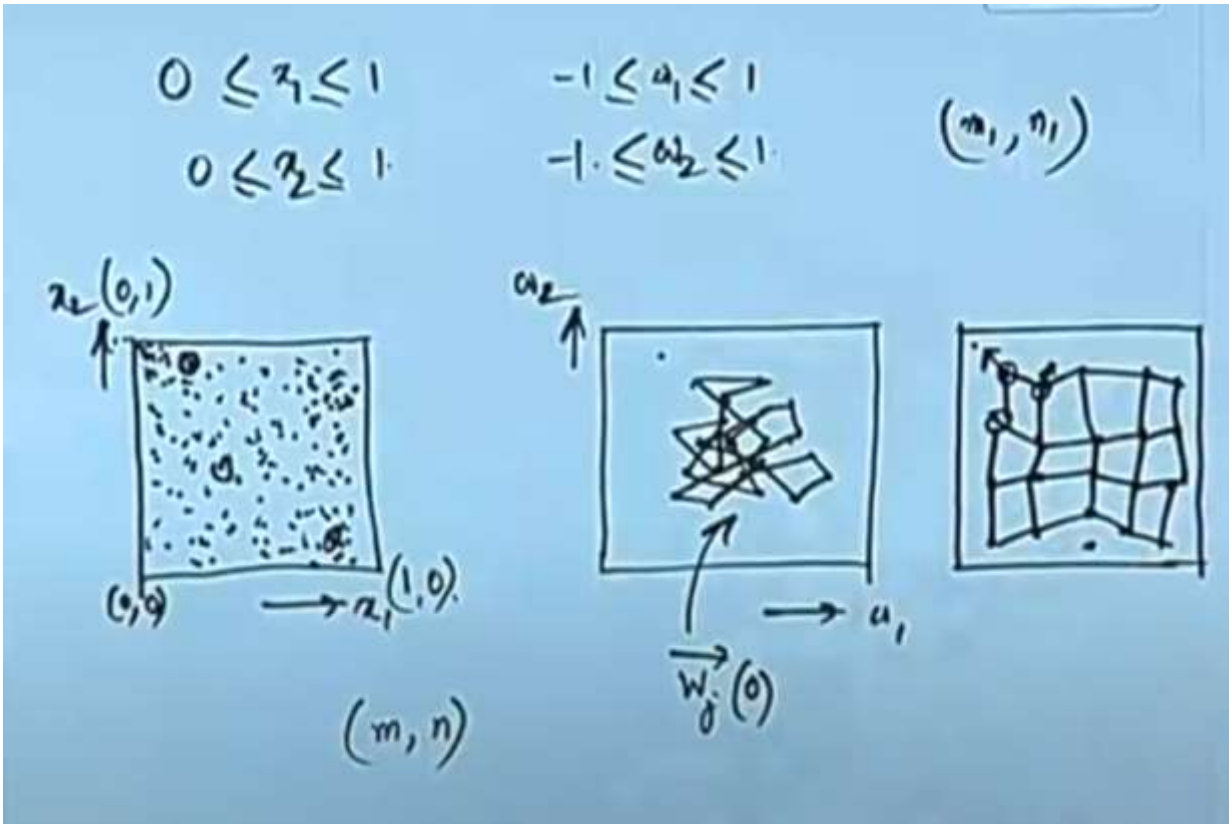
Using discrete-time formulation

$$\vec{w}_j(n+1) = \vec{w}_j(n) + \eta(n) h_{j,i(x)}(n) (\vec{x} - \vec{w}_j)$$

Topological ordering.

$$\eta(n) = \eta_0 \exp\left(-\frac{n}{\tau_2}\right) \quad n = 0, 1, 2, \ldots\ldots$$

τ_2 : another time constant.


$0 \le z_1 \le 1$
$0 \le z_2 \le 1$
$-1 \le w_1 \le 1$
$-1 \le w_2 \le 1$
(m_1, n_1)
z_2
$(0,1)$
$(0,0)$
z_1
$(1,0)$
(m, n)
w_2
w_1
$\vec{w}_{ij}(0)$