UNSUPERVISED LEARNING
SELF ORGANIZING MAPS
MODULE 6

Dr. R.B.Ghongade,
VIIT, Pune-411048

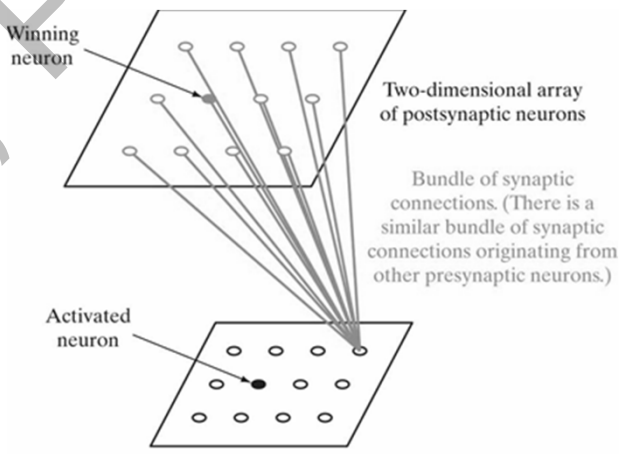
Agenda

- Introduction
- Concept
- Motivation
- Models
  - Willshaw von der Marlsburg model
  - Kohonen model
- Architecture of SOM
- Essential Processes in SOM
  - Competition
  - Co-operation
  - Synaptic Adaptation
- Mathematical model of the processes
- Phases of weight adaptation
- Concept of Topology Organization
- Algorithm
- MATLAB Demo

Introduction

- Self organizing maps(SOM) are also known as Self Organizing Feature maps(SOFM)
- SOMs are topology preserving nets
- They employ unsupervised learning
- SOMs work on the principle of competitive learning
- There is a spatial organization of the distribution of neurons(lattice of output neurons, 1-D ,2-D or even higher D)
- Most practical problems make use of 1-D or 2-D lattices, higher D lattices NOT preferred due to complexity
- Neurons are arranged in a structure manner and if we present input patterns , they act as a stimuli to these neurons

Structured Output Neurons



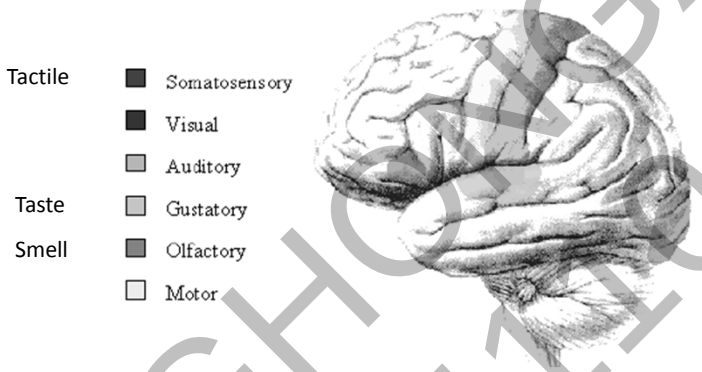
Concept

- Out of the different neurons in the output lattice, one of them will be the winner and synaptic weights from input to output will be adjusted so that the Euclidean distance between the synaptic weights and the input vector is minimized
- Minimization of Euclidean distance means maximization of $W^T \cdot X$
- The inputs are randomly distributed in the input space, but if we start with a regular structure, depending on the input statistics, **the ultimate organization of the lattice that results would be indicative of the statistics of input patterns** that we are applying as stimuli
- Training is the process where the lattice structure is disturbed and it will move the winning neuron physically towards the input pattern

- Neurons at the output act in a competitive manner: they inhibit the responses of other output neurons
- But in the vicinity of the winning neuron an excitatory response is created while an inhibitory response is created for other output neurons which are far apart
- Such networks exhibit:
  - Short range excitation
  - Long range inhibition

Motivation

- Neuro-biological phenomena in the brain actually exhibits topologically ordered computation
- Various sensory inputs like visual, acoustic and tactile inputs are processed in different regions of the cerebral cortex

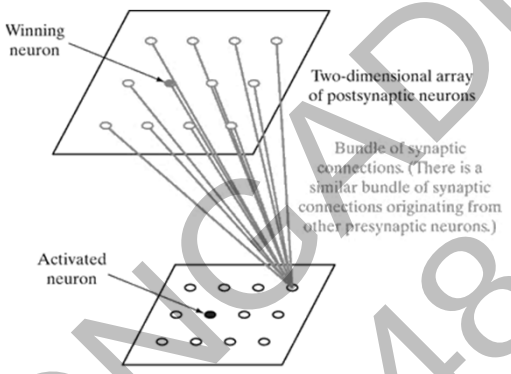


Models of SOM

1. Willshaw von der Marlsburg model
2. Kohonen model

Willshaw von der Marlsburg model

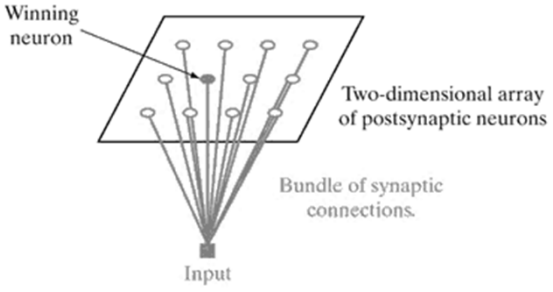
- Two arrays (lattices) of pre-synaptic and post-synaptic neurons
- Was used to explain the retino-optic mapping from retina to visual cortex, where retina will form the pre-synaptic layer and the visual cortex is the post-synaptic layer
- Dimensions of input and output are **same**
- Electric signals of pre-synaptic neurons are based on geometric proximities



- The neurons near the excited neuron have highly correlated electrical responses
- If signal for A is strong , the signal for a geometrically close neuron B, will also be strong and it will enhance spatially those output neurons which are there in similar spatial locations
- There is a spatial correlation of activities
- Hence pre-synaptic layer activities are mapped onto similar neuron activities in the post synaptic layer

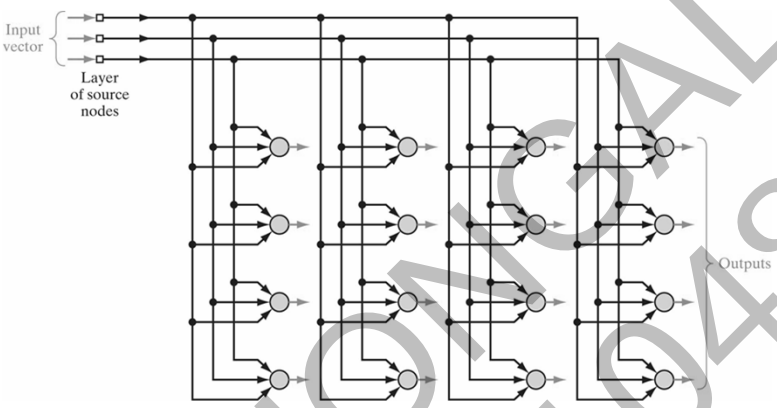
Kohonen model

- Based on vector coding algorithm which optimally places a fixed number of vectors (called code words) into a higher dimensional input space
- Same as data compression technique like entropy coding



- If we have an input of length l , it can be compressed into a code word of length m , which is much less than l
- This technique exploits the entropy in the data eg: Huffmann coding
- A fixed number of codewords form a vector
- Kononen model is more popularly used because it offers dimensionality reduction and is more general
- Hence SOMs are also referred to as Kohonen self organizing maps

Architecture of SOM



Essential Processes in SOM

1. Competition:
   - Long range inhibition
   - For each input pattern the neurons in the output layer determine the value of a function
   - Called as the discriminant function
   - This function provides the basis of competition
   - A particular neuron with the largest discriminant function value is the winner
   - Responses of other neurons (at longer geometric distances) is set to zero

Essential Processes in SOM

2. Co-operation:
   - Short range excitation
   - Winning neuron determines the topological neighbourhood of excitation for other output neurons
   - Neuron that wins , excites the neighbouring neurons surrounding it
   - Excitation is co-operative and always follows competition
   - In other words, the winning neuron determines the spatial location of topological neighbourhood of excited neurons

Essential Processes in SOM

3. Synaptic Adaptation:
   - Enables the excited neurons to increase their individual values of discriminant function in relation to the input pattern
   - This adaptation is only for excited neurons
   - When similar input patterns are fed repeatedly , then we have the increasing winning neuron response each time

Competition

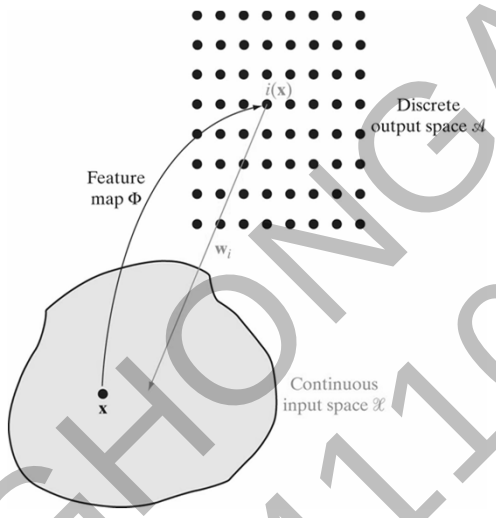
- Consider an m dimensional input $\vec{X} = [x_1 x_2 \cdots x_m]$
- Then $\overrightarrow{W_j} = [w_{j1}\ w_{j2} \cdots w_{jm}]^{\mathrm{T}}$ where $j = 1,2, \cdots l$ (l=total number of output neurons)
- Find the best match between $\vec{X}$ and $\overrightarrow{W_j}$
- There will be competition between l neurons and whichever $\overrightarrow{W_j}$ is having the best match with $\vec{X}$, that j^{th} neuron will emerge as the winner
- Some applications require only the index j of the winning neuron while some may require the actual winning vector
- We compute $W^T \cdot X$ for $j = 1,2, \cdots l$ and select the largest amongst these
- Maximizing $W^T \cdot X$ ->minimizing $\left\|\vec{X} - \overrightarrow{W_j}\right\|$, the Euclidean distance

- Using index $i(\vec{X})$ where i is the index based on input vector $\vec{X}$
- Then

$$i(\vec{X}) = \arg min \left\|\vec{X} - \overrightarrow{W_j}\right\|$$

- Corresponding weight vector $\overrightarrow{W_j}$ Is closest to input pattern $\vec{X}$
- Input $\vec{X}$ is a continuous m-dimensional space and we are mapping it onto a discrete space of l outputs
- *Hence a continuous input space of activation patterns is mapped onto a discrete output space of neurons by a process of competition*

Illustration of the relationship between feature map Φ and weight vector $\overrightarrow{W_j}$ of winning neuron *i*.



Cooperation

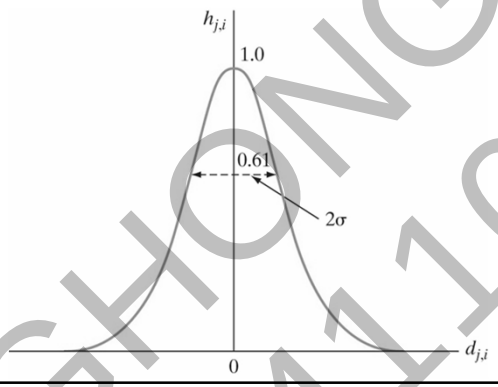
- When a neuron fires , it also excites the neighbouring neurons
- If we go farther away from the winning neuron then its neighbourhood function should gradually decrease
- This function should be monotonically decreasing function
- Topological neighbourhood is defined as

  $h_{i,j(\vec{X})}$:Topological neighbourhood centered around i encompassing neuron j using the measure

  $d_{j,i}$:Lateral distance between the winning neuron i and the excited neuron j
- Weights of this excited neurons have to be updated and up to what extent is determined by the neighbourhood

- This ***topological neighbourhood*** $h_{i,j}$ should satisfy :
  - It should be symmetric about $d_{j,i}$
  - It should be monotonically decaying function, decreasing with distance $d_{j,i}$ i.e;as $d_{j,i} \to \infty$, $h_{i,j} = 0$
- A typical choice is the Gaussian function

$$h_{i,j} = exp\left(\frac{-d_{j,i}^{\ 2}}{2\sigma^2}\right)$$



- This function is independent of the position of the winning neuron , hence it is translation invariant
- In case of 1-D lattice $d_{j,i} = |j - i|$
- For 2-D lattice $d_{j,i} = \left\|\vec{r_j} - \vec{r_i}\right\|$

Where $\vec{r_j}$:position vector of excited neuron

$\vec{r_i}$: position vector of winning neuron

- Another important feature of SOM is that σ is not constant with iterations
- As iterations increase σ decreases means the neighbourhood shrinks with iterations

$$\sigma(n) = \sigma_0 exp\left(\frac{-n}{\tau_1}\right) \ldots\ldots n = 0,1,2,\ldots$$

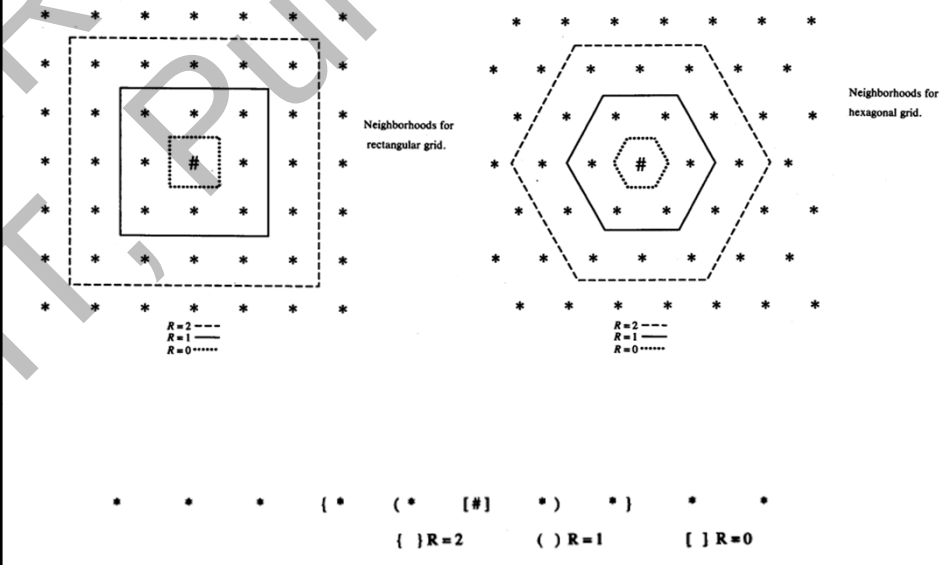
Where $\tau_1$is a time constant

- Consequently

$$h_{j,i(\overrightarrow{X})}(n) = exp\left(\frac{-{d_{j,i}}^2}{2\sigma(n)^2}\right) \ldots n = 0,1,2 \ldots$$

- $h_{j,i(\overrightarrow{X})}(n)$ is called as the **neighbourhood function**
- Significance of co-operation:
  - Initially a large number of neurons participate in the co-operative process because σ is large
  - We update the winning neurons and its neighbours
  - If say $\overrightarrow{X_1}$is fed to the network and output neuron i is the winner, next if $\overrightarrow{X_2}$ is now fed to the network and now neuron j is the new winner.
  - If $\overrightarrow{X_1}$ and $\overrightarrow{X_2}$ are close then neuron i also participates in weight updation
  - If we feed a large number of vectors close to $\overrightarrow{X_1}$ i.e,; cluster of patterns, the winners are different but every output neuron in the neighbourhood gets its share of weight update
  - Ultimately the topology of the network will be adjusted to the clusters

Alternative neighbourhood Functions



Weight Adaptation

- This the learning phase
- We can use the hebbian learning but it is severely limited by weight saturation if presented with same input pattern repeatedly
- Hence we modify it by introducing the *forgetting term:*

$$g(y_j) \cdot \overrightarrow{W_j}$$

Where $g(\cdot)$ is a positive scalar function

- To simplify let

$$g(y_j) = \eta y_j \dots\dots\dots \eta = learning\ rate$$

- We can set y_j as:

$$y_j = h_{j,i(\overrightarrow{X})}$$

- The modified Hebbian learning is

$$\Delta\overrightarrow{W_j} = \eta y_j \vec{X} - \eta g(y_j)\overrightarrow{W_j}$$

- The weight update rule becomes:

$$\Delta\overrightarrow{W_j} = \eta y_j \vec{X} - \eta h_{j,i(\overrightarrow{X})}\overrightarrow{W_j}$$

Or

$$\Delta\overrightarrow{W_j} = \eta h_{j,i(\overrightarrow{X})} \vec{X} - \eta h_{j,i(\overrightarrow{X})}\overrightarrow{W_j}$$

Finally can be written as:

$$\boldsymbol{\Delta\overrightarrow{W_j} = \eta h_{j,i(\overrightarrow{X})}(\vec{X} - \overrightarrow{W_j})}$$

- We essentially move the weight vector $\overrightarrow{W_j}$ to align with input vector $\vec{X}$
- Using discrete time formulation we express weight update as:

$$\boldsymbol{\overrightarrow{W_j}(n+1) = \overrightarrow{W_j}(n) + \eta(n) h_{j,i(\overrightarrow{X})}(n)(\vec{X} - \overrightarrow{W_j})}$$

- $\eta(n)$ is the variable learning rate
- This leads to topological ordering

- The two important heuristics involved here are:

$$\boldsymbol{\eta(n)}$$

and

$$\boldsymbol{h_{j,i(\overrightarrow{X})}(n)}$$

- We use

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

Where τ_2 is another time constant

- Recall that

$$h_{j,i(\overrightarrow{X})}(n) = exp\left(\frac{-d_{j,i}^2}{2\sigma(n)^2}\right) \ldots n = 0,1,2 \ldots$$

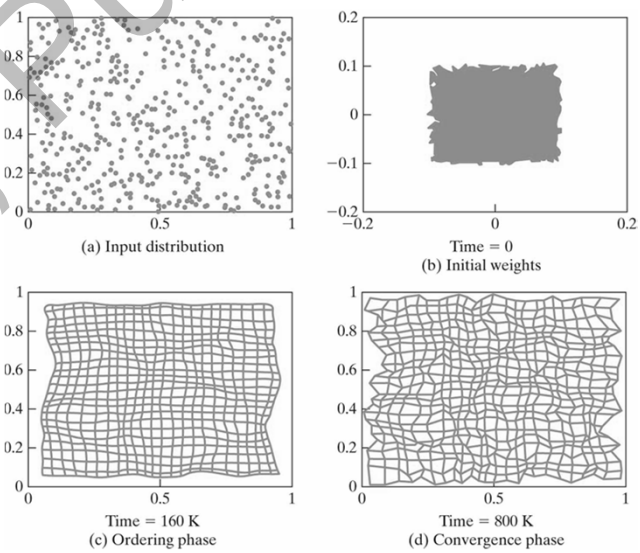
Two phases of weight adaptation

1. Self organizing(ordering)
   - where topology is arranged
   - It is a fast phase requires less iterations
   - $\eta(n)$ to starts with, say, 0.1 and should decrease to 0.01 (τ_1=1000)
   - It takes generally more than 1000 iterations
   - $h_{j,i(\overrightarrow{X})}(n)$ to start with should include a large number of neurons and later be restricted to a small neighbourhood

Two phases of weight adaptation

2. Convergence
   - All obtained positions are fine tuned
   - Requires iterations , at least 500 times the number of output neurons
   - $\eta(n)$ remains constant in this phase but $h_{j,i(\overrightarrow{X})}(n)$ may continue to decrease

Concept of topology organization



Algorithm(simplified)

STEP 0	Initialize weights $w_{i,j}$ 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 $\vec{X}$, do Steps 3-5.	
		STEP 3	For each j, compute: $D(j) = \sum_i (x_i - w_{i,j})^2$
		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: Compute $y_j = w_{i,j} \cdot x_i$ $w_{i,j}(new) = w_{i,j}(old) + \eta y_j [x_i - w_{i,j}(old)]$

STEP 1	**STEP 2**	
	STEP 6	Update learning rate
	STEP 7	Reduce radius of topological neighborhood at specified times.
	STEP 8	Test stopping condition

MATLAB Demo