

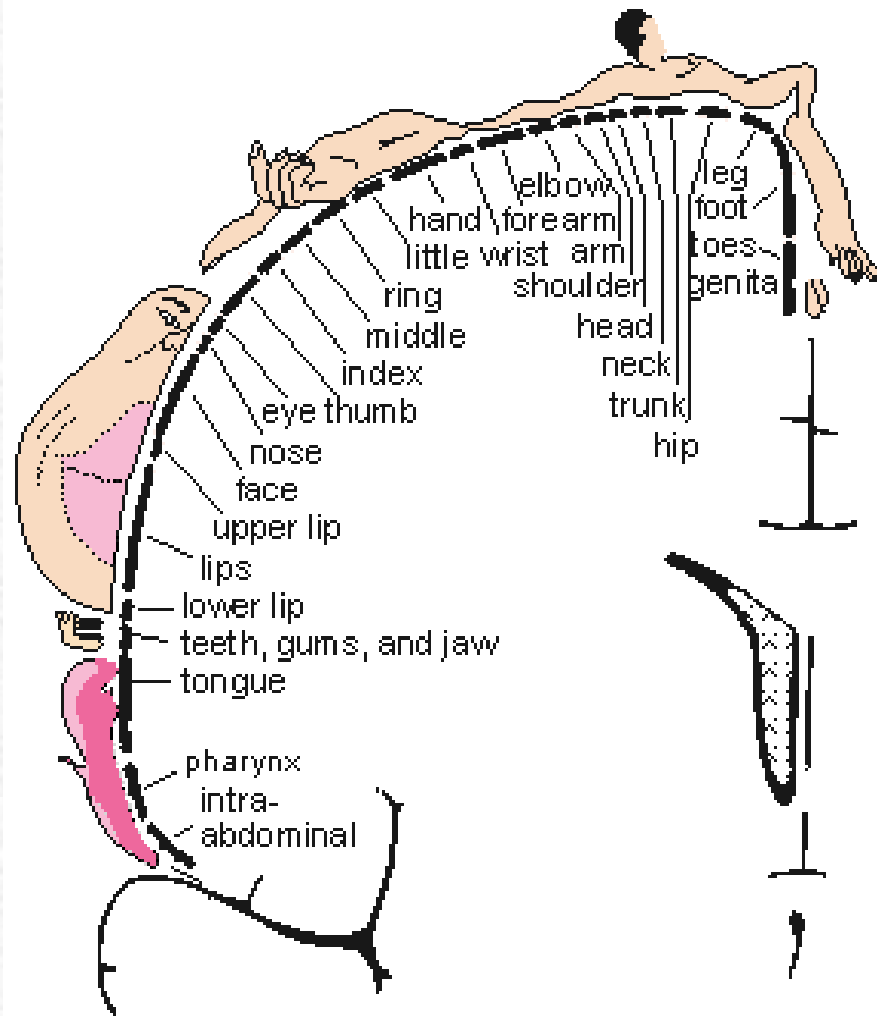
VI. SELF ORGANIZING NETWORKS

These networks are based on competitive learning in which the output neurons of the network compete among themselves

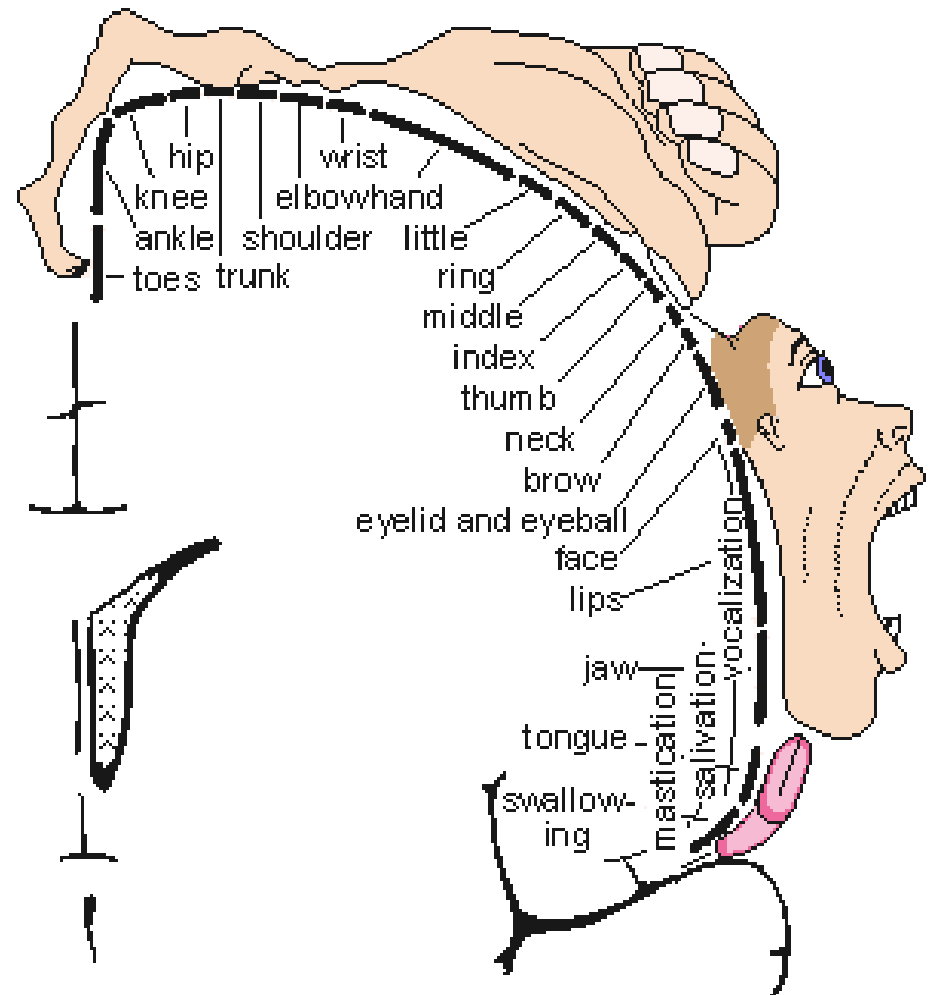
- ✓ Basic Feature-Mapping Model
- ✓ Self Organizing Map
 - ✓ Competition
 - ✓ Cooperation
 - ✓ Synaptic Adaptation
- ✓ Learning Vector Quantization

Self Organizing Process

- ***unsupervised training***, in which the networks learn to form their own classifications of the training data without external help.
- we have to assume that class membership is broadly defined by the input patterns sharing ***common features***, and that the network will be able to identify those features across the range of input patterns.





SENSORY CORTEX



MOTOR CORTEX


- One particularly interesting class of unsupervised system is based on ***competitive learning***– the output neurons compete amongst themselves to be activated, with the result that only one is activated at any one time. This activated neuron is called a ***winner-takes-all neuron*** or simply the ***winning neuron***. Such competition can be induced/implemented by having ***lateral inhibition connections*** (negative feedback paths) between the neurons. The result is that the neurons are forced to organize themselves. For obvious reasons, such a network is called a ***Self Organizing Map (SOM)***.

- 
- Neurobiological studies indicate that different sensory inputs (motor, visual, auditory, etc.) are mapped onto corresponding areas of the cerebral cortex in an ***orderly fashion***. This form of map, known as a ***topographic map***, has two important properties
 - 1. At each stage of representation, or processing, each piece of incoming information is kept in its proper context/neighbourhood.
 - 2. Neurons dealing with closely related pieces of information are kept close together so that they can interact via short synaptic connections.
- 



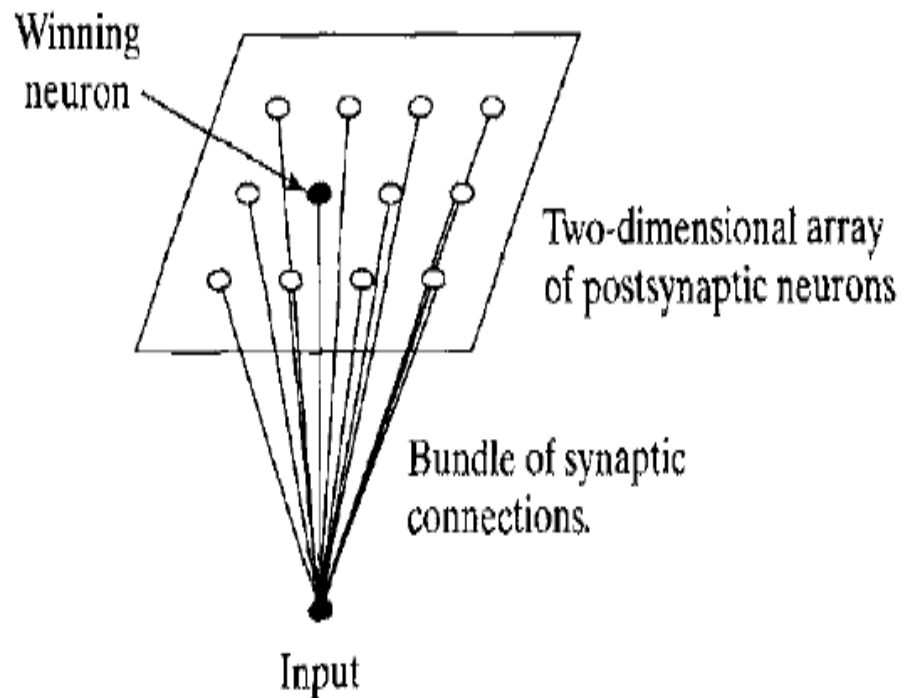
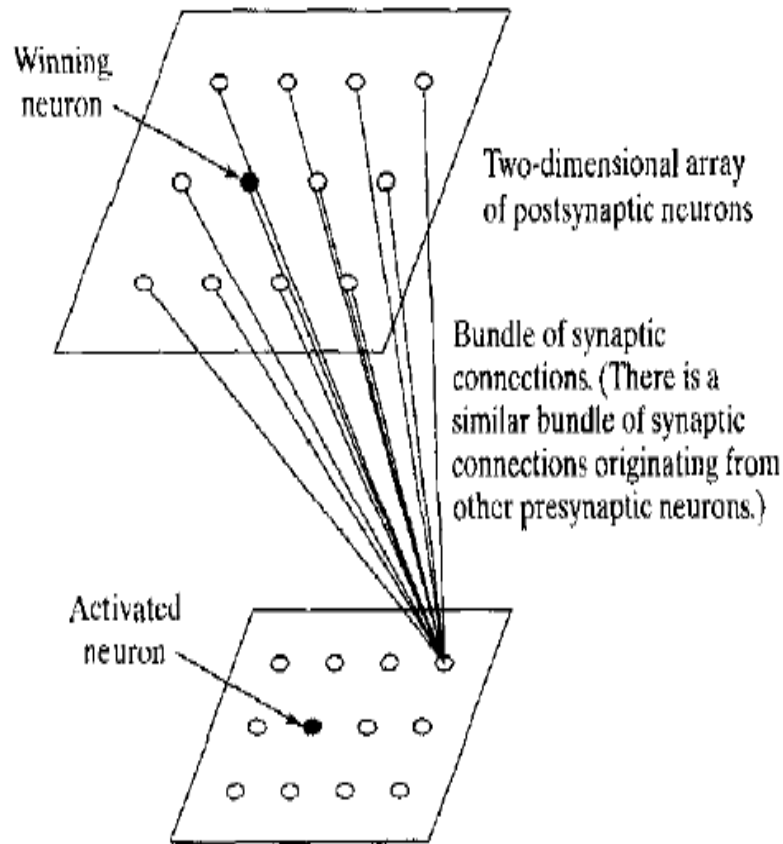
• *principle of topographic map*

formation: “The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature drawn from the input space”. Two basically different feature-mapping models.

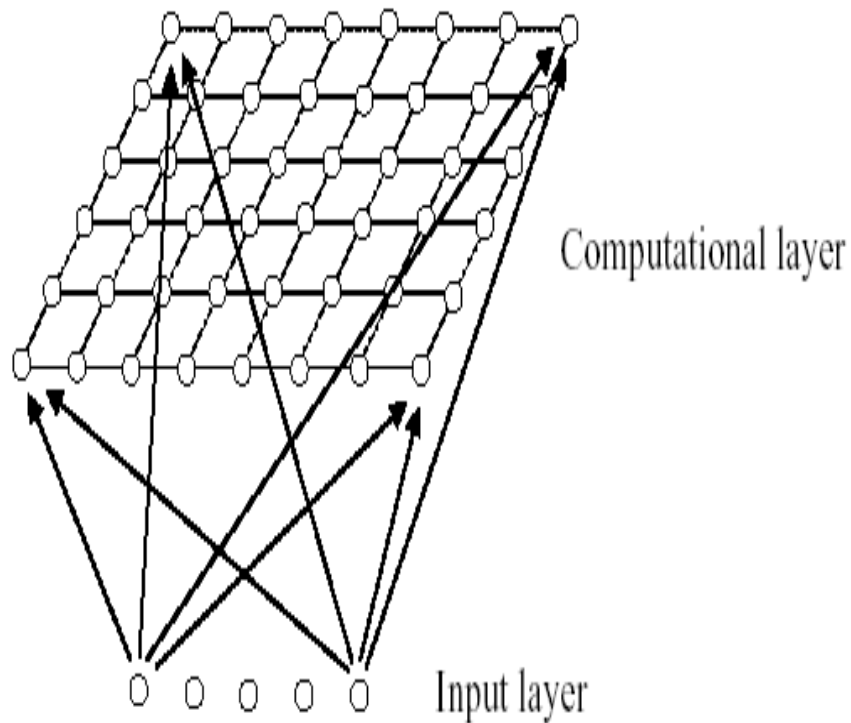
- Networks trained without a teacher ususally learn by matching certain explicit familiarity criteria.
- 

Two self-organizing feature maps:

- Willshaw-Van der Malsburg model
- Kohonen model



The Architecture a Self Organizing Map



- We shall concentrate on the SOM known as a ***Kohonen Network***. This SOM has a feedforward structure with a single computational layer arranged in rows and columns. Each neuron is fully connected to all the source nodes in the input layer:

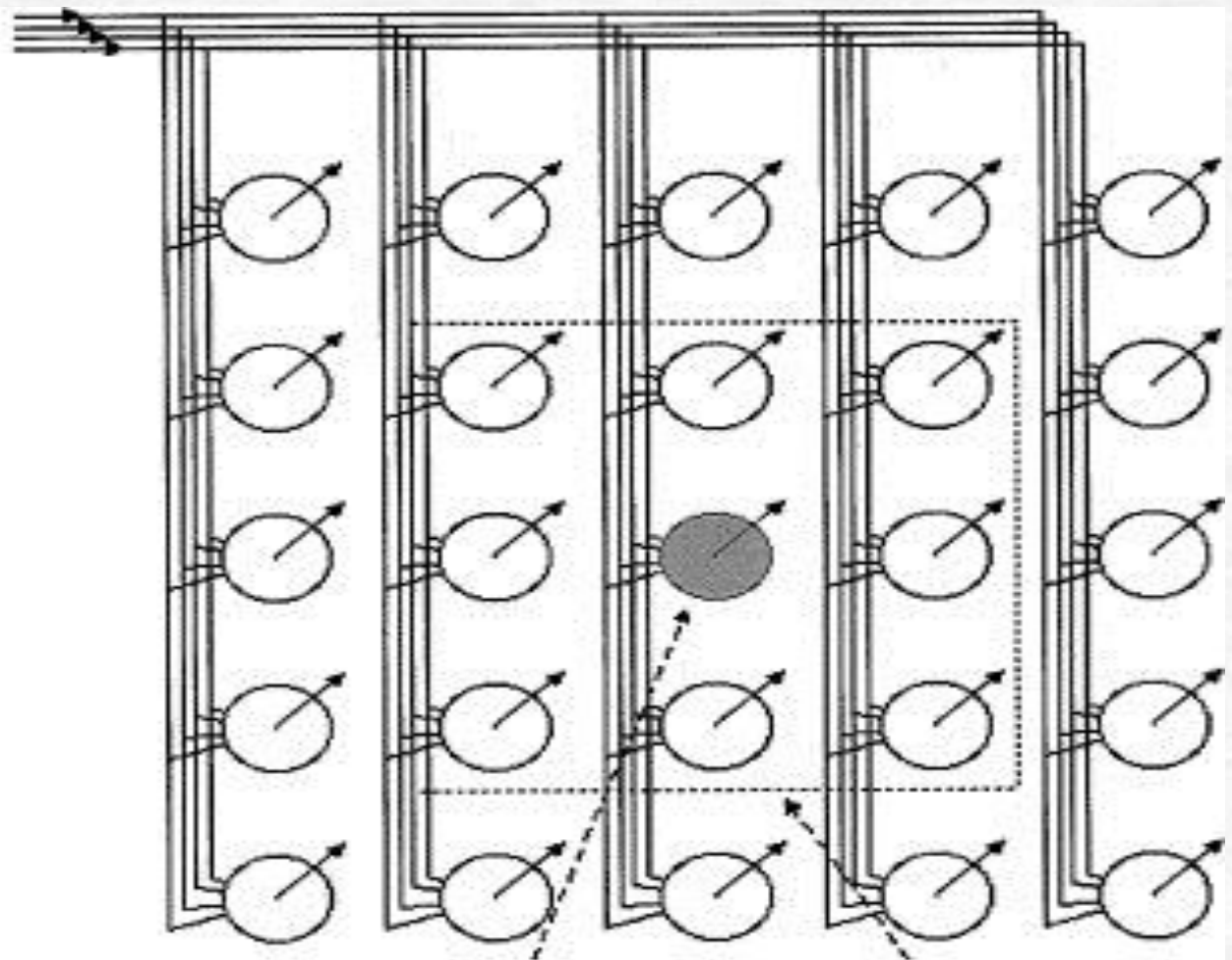
Components of Self Organization

- **Initialization:** Weights are initialized with small random values.
- **Competition:** The particular neuron with the smallest value of the discriminant function is declared the winner.
- **Cooperation:** The winning neuron is providing the basis for cooperation among neighbouring neurons.
- **Adaptation:** Through suitable adjustment to the associated connection weights, the response of the winning neuron to the subsequent application of a similar input pattern is enhanced.

7.2.SOM Structure

- Output (post-synaptic) neurons are organized in a spatial pattern (typically 1- d, 2-d or 3- d) corresponding to spatial characteristics of the problem domain. Each neuron in the lattice is fully connected to the all source nodes in the input layer.
- The learning process consists of 2 steps:
 - a specialization step where each output neuron is trained to a specific class of inputs
 - a re-organization step where the neurons of the output layer are “placed” so they correspond spatially to the problem domain

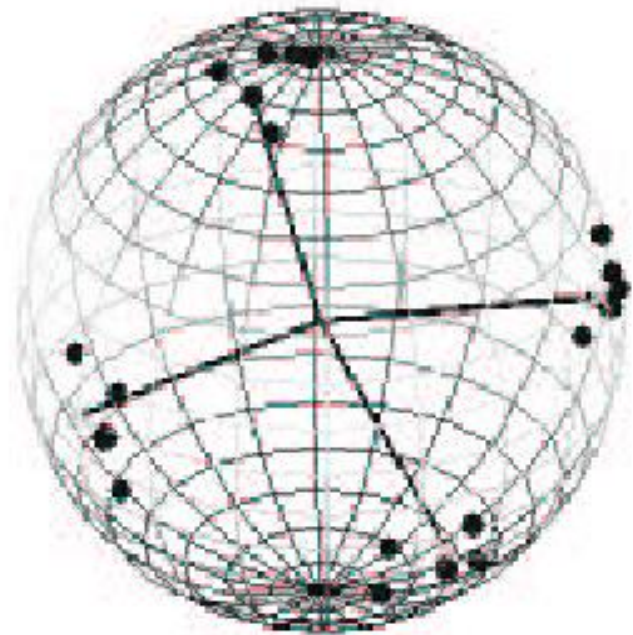
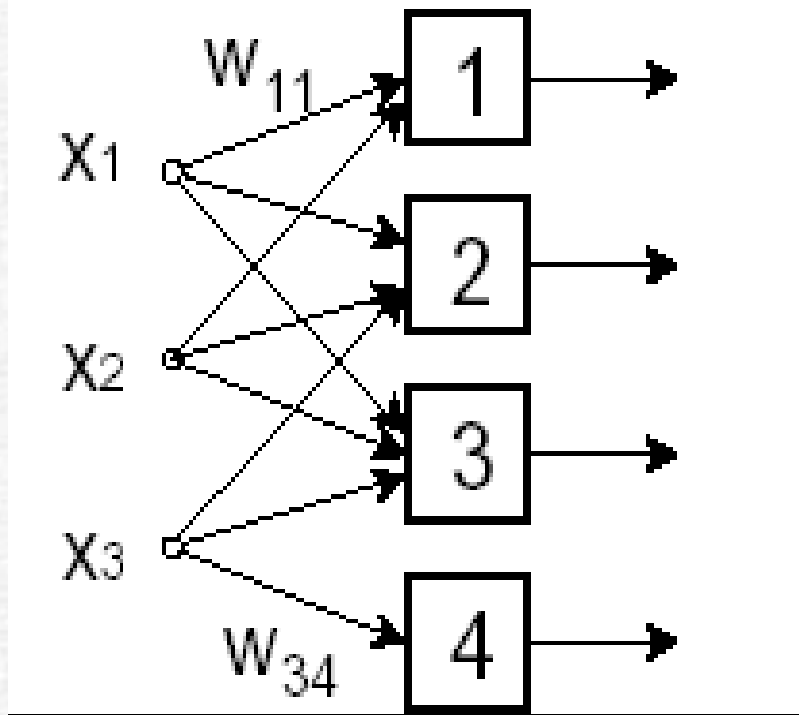
Input neurons



Most responsive neuron

Neighborhood

Winner takes all:
Weights of the unit with highest activation are updated
The weight vector rotates slowly towards the cluster centers



The Competitive Process

- If the input space is D dimensional (D input units) the input patterns is

$\mathbf{x} = \{x_i : i = 1, \dots, D\}$ and the connection weights between the input units i and the neurons j in the computation layer can be written

$\mathbf{w}_j = \{w_{ji} : j = 1, \dots, N; i = 1, \dots, D\}$ where N is the total number of neurons.

- Define the **discriminant function** as the squared Euclidean distance between the input vector \mathbf{x} and the weight vector \mathbf{w}_j for each neuron j

$$d_j(\mathbf{x}) = \sum_{i=1}^D (x_i - w_{ji})^2$$

Competition among the neurons

- The index of the winning neuron

Best matching condition:

$$i(\mathbf{x}) = \arg \min_j \|\mathbf{x} - \mathbf{w}_j\| \quad j=1, \dots, N$$

- The particular neuron i that satisfies this condition is called the best matching or winning neuron for the particular input \mathbf{x} .

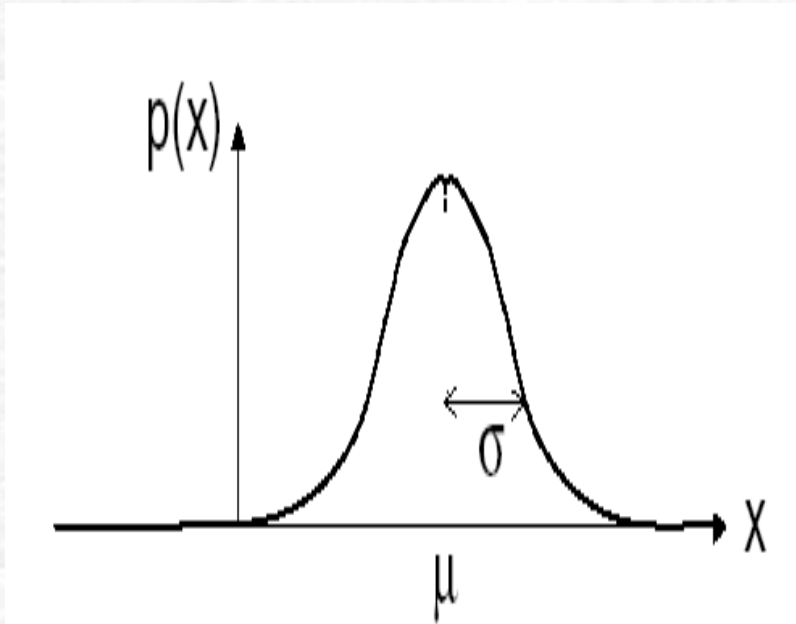
The Cooperative Process

- In neurobiological studies we find that there is ***lateral interaction*** within a set of excited neurons. When one neuron fires, its closest neighbours tend to get excited more than those further away.
- There is a ***topological neighbourhood*** that decays with distance. We also want to define a similar topological neighbourhood for the neurons in SOM topology.

Cooperative Step

- Define a topological neighborhood around the winning neuron. The winning neuron is at the center of a neighborhood of cooperating neurons,
- The topological neighborhood can be defined using any reasonable “closeness” measure,
- A Gaussian distribution is typically used for the proximity measure,
- The neighborhood reach should be diminished through the iterations of the learning process.

Gaussian (normal) distribution



- μ is called the 'mean' (the expected value)
- σ^2 is the 'variance' (it controls the spread)
- (σ is 'standard deviation')

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

Synaptic Adaptation

- For all neurons in the neighborhood of the winning neuron, modify the synaptic weights according to the rule:

$$\Delta w_j = \eta y_j - g(y_j) w_j$$

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

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

$$\sigma(n) = \sigma_0 \exp\left(\frac{-n}{\tau}\right)$$

Adaptive Process

$$\frac{d\mathbf{w}_j}{dt} = \eta y_j \mathbf{x} - g(y_j) \mathbf{w}_j$$

learning rate parameter

stabilizing term

$$y_j = \begin{cases} 1 & \text{if neuron } j \text{ is inside } \Lambda_{i(x)}^{(n)} \\ 0 & \text{otherwise} \end{cases}$$

winning neuron

Continuous-time Dynamics:

$$g(y_j) = \begin{cases} \alpha & \text{neuron } j \text{ is active} \\ 0 & \text{neuron is off} \end{cases}$$

Hebbian Term and Forgetting Term

$$\frac{d\mathbf{w}_j}{dt} = \left\{ \begin{array}{ll} \eta \mathbf{x} - \alpha \mathbf{w}_j & \text{if } j \in \Lambda_{i(x)}(n) \\ 0 & \text{otherwise} \end{array} \right\}$$

Choose $\alpha = \eta$; hence,

$$\begin{aligned} \frac{d\mathbf{w}_j}{dt} &= \eta(\mathbf{x} - \mathbf{w}_j) \\ \mathbf{w}_{j(n+1)} &= \begin{cases} \mathbf{w}_{j(n)} + \eta(\mathbf{x} - \mathbf{w}_{j(n)}) & \text{if } j \in \Lambda_i \\ \mathbf{w}_{j(n)} & \text{if } j \in \Lambda_i \end{cases} \end{aligned}$$

Choice of Parameters

Ordering phase

$\eta(n)$ close to 1, and then decreased slowly during first 1000 iterations; but remaining above 0.1.

Convergence phase (final tuning)

tens of thousands of iterations

$\eta(n)$ is maintained at a very small value 0.01

Λ_i is large at the beginning, then it is shrunk slowly with time during the ordering phase, until after 1000 iterations it is shrunk to a couple of neighboring neurons.

Then for the convergence phase, Λ_i is reduced to a neighborhood of 1 or 0 neuron.

Reformulation of SOM

d_{ji} = distance of neuron i from winning neuron j

σ = spread of the neighborhood function

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n)\pi_{ji}(n)(\mathbf{x}(n) - \mathbf{w}_j(n))$$

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

$$\sigma(n) = \sigma_o \exp\left(-\frac{n}{\tau_1}\right)$$

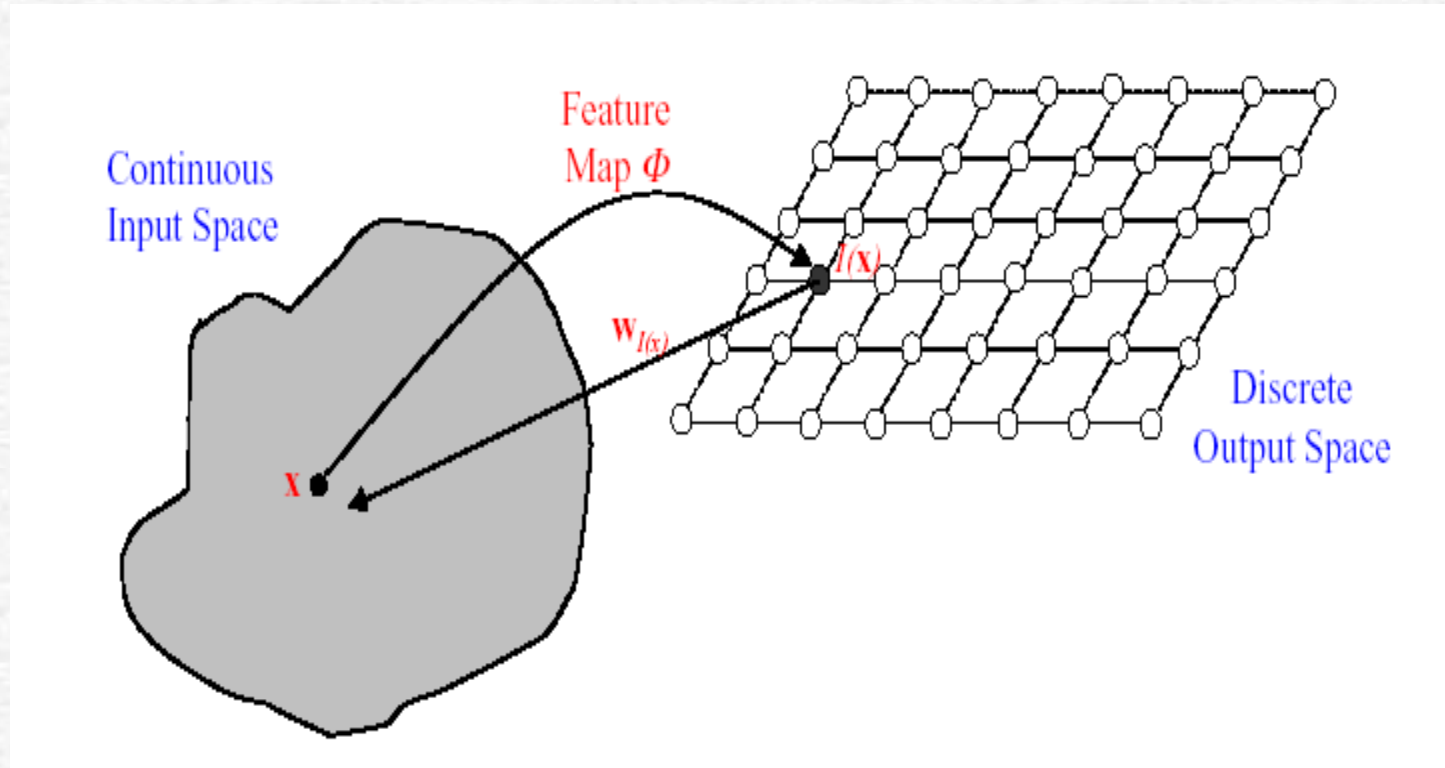
time constants

$$\eta(n) = \eta_o \exp\left(-\frac{n}{\tau_2}\right)$$

learning
rate



Feature Mapping

A mapping from a continuous input space to a discrete output space. Given an input vector \mathbf{x} , the feature map F provides a winning neuron $I(\mathbf{x})$ in the output space, and the weight vector $\mathbf{w}_{I(\mathbf{x})}$ provides the coordinates of the image of that neuron in the input space.



Properties of SOM

1. Approximation of input space: The theoretical basis of this idea is rooted in ***vector quantization theory***, the motivation of which is dimensionality reduction or data compression.
2. Topological ordering : spatial location of a neuron in the map corresponds to a particular feature in the input space forces the weight vector $\mathbf{w}_{I(\mathbf{x})}$ of the winning neuron $I(\mathbf{x})$ to move toward the input vector \mathbf{x} .

- 
3. **Density matching:** The feature map reflects variations in the statistics of the input distribution. Regions of the input space where the input vector \mathbf{x} occurs with high probability are mapped onto larger domains of the output space.
 4. **Feature Selection:** The SOM provides a discrete approximation of finding so-called ***principal curves*** or ***principal surfaces***, and may therefore be viewed as a non-linear generalization of PCA. SOM networks extract the principal features of the input space
- 

Summary of SOM algorithm

- Initialization
- Sampling: Pick a sample \mathbf{X} from input distribution with some probability

- Similarity matching:

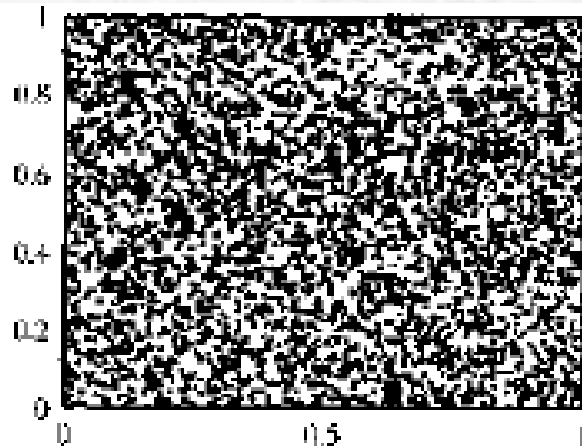
$$i(\mathbf{x}) = \arg \min_j \|\mathbf{x} - \mathbf{w}_j\|$$

- Updating

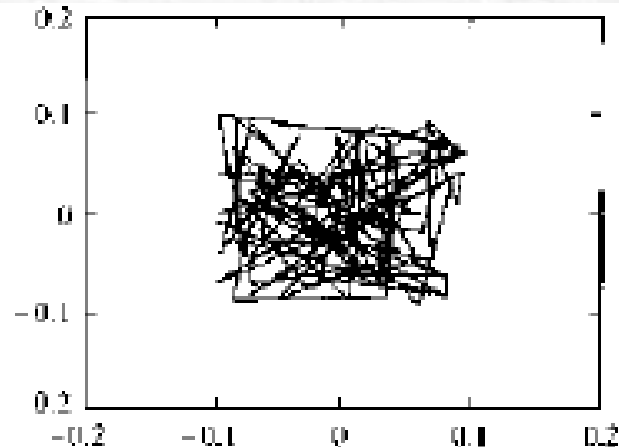
$$\mathbf{w}_j(n+1) = \begin{cases} \mathbf{w}_j(n) + \eta(\mathbf{x}(n) - \mathbf{w}_j) & \text{if } j \in \Lambda_j \\ \mathbf{w}_j(n), & \text{otherwise} \end{cases}$$

- Continue* by picking another sample until the map reaches *stable condition*

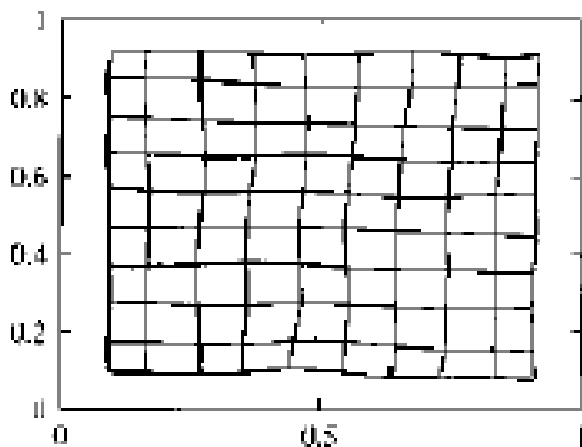
Example (No dimension reduction)



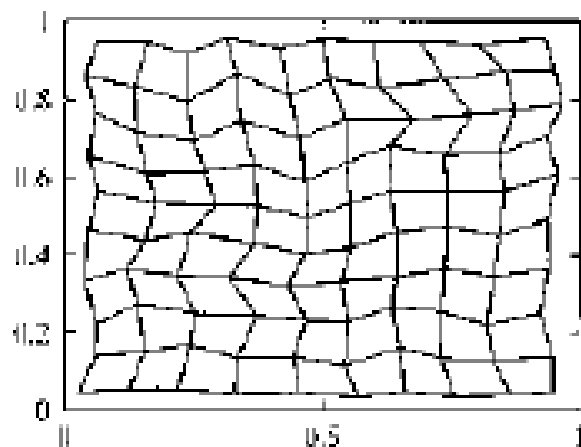
(a)



(b)

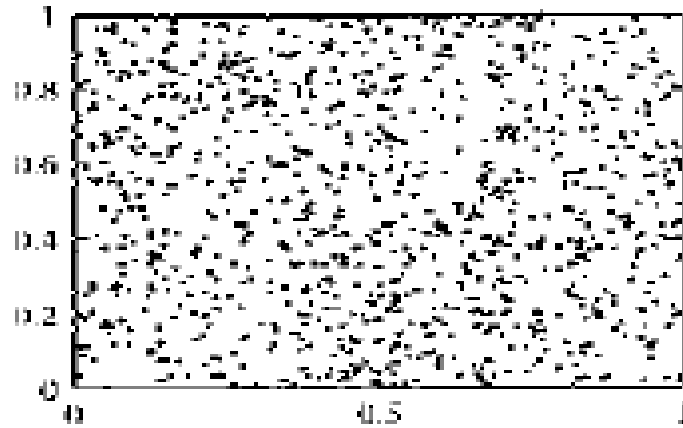


(c)

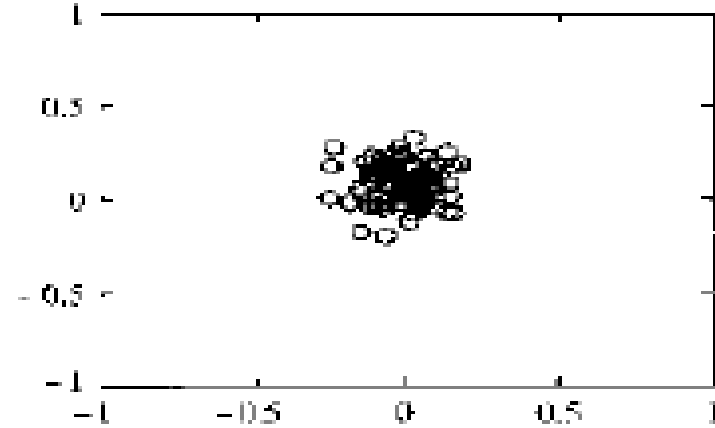


(d)

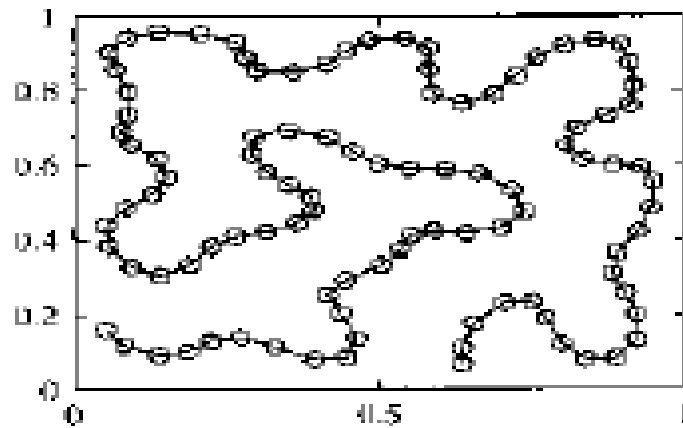
Example (with dimensionality reduction)



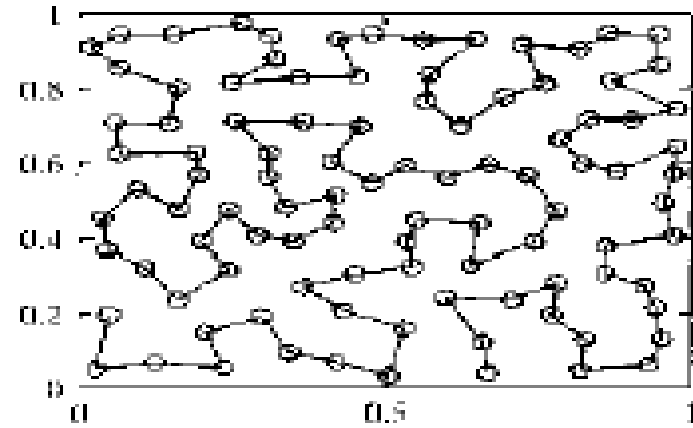
(a)



(b)



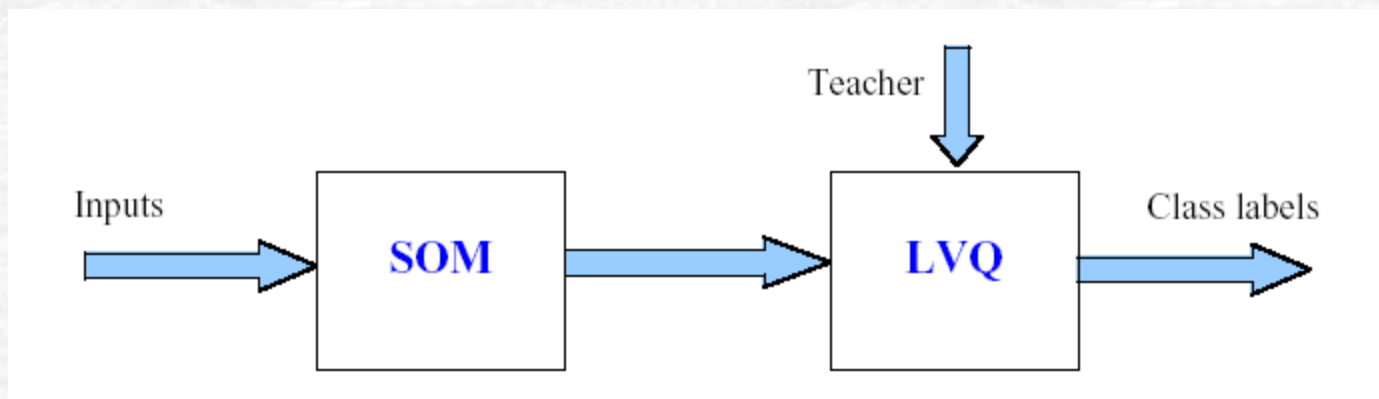
(c)



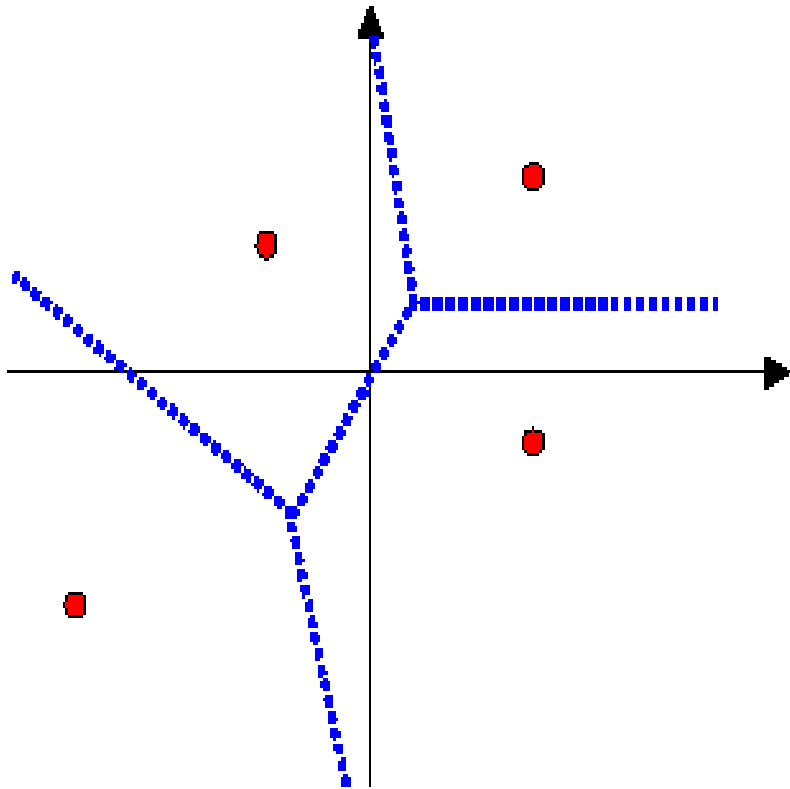
(d)

Learning Vector Quantization (LVQ)

- **Learning Vector Quantization** (LVQ) is a supervised version of vector quantization that can be used when we have labelled input data. This learning technique uses the class information to move the Veroni vectors slightly, so as to improve the quality of the classifier decision regions. It is a two stage process – a SOM followed by LVQ: The second step is the classification where the feature domains are assigned to individual classes.



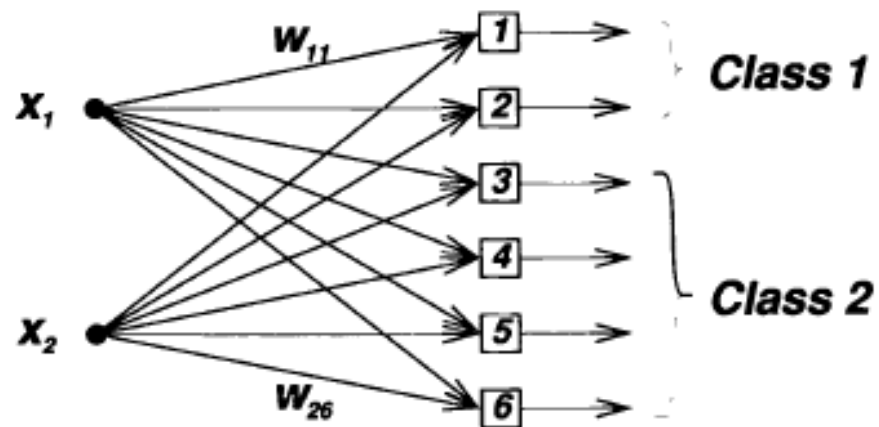
Voronoi Quantizer



A vector quantizer with minimum encoding distortion is often called a ***Voronoi quantizer*** or ***nearest-neighbour quantizer***. The input space is partitioned into a set of ***Voronoi or nearest neighbour cells*** each containing an associated ***Voronoi or reconstruction vector***:

LVQ Algorithm

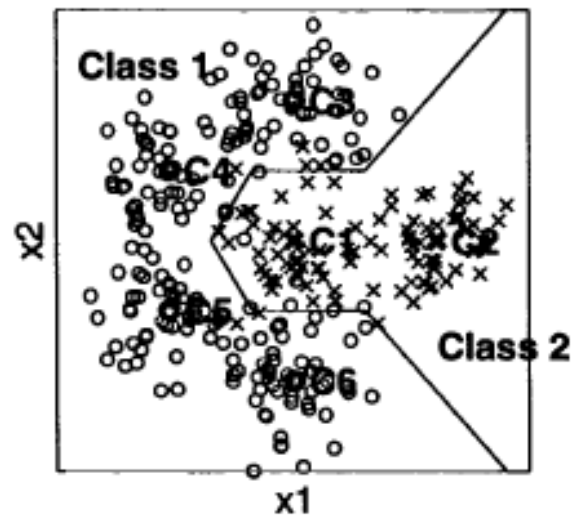
- It starts from a trained SOM with input vectors $\{\mathbf{x}\}$ and weights/Voronoi vectors $\{\mathbf{w}_j\}$. Use the classification labels of the inputs to find the best classification label for each \mathbf{w}_j , i.e. for each Voronoi cell. It is unlikely that these Voronoi cell boundaries will match the classification boundaries. The LVQ algorithm attempts to correct this by shifting the boundaries:



↑
Input Units

↑
Output Units

(a)



(b)

1. If the input \mathbf{x} and the associated Voronoi vector/weight $\mathbf{w}_{I(\mathbf{x})}$ (i.e. the weight of the winning output node $I(\mathbf{x})$) have the same class label, then move them closer together by as in the SOM algorithm.

$$\Delta \mathbf{w}_{I(\mathbf{x})}(t) = \beta(t)(\mathbf{x} - \mathbf{w}_{I(\mathbf{x})}(t))$$

2. If the input \mathbf{x} and associated Voronoi vector/weight $\mathbf{w}_{I(\mathbf{x})}$ have the different class labels, then move them apart by

$$\Delta \mathbf{w}_{I(\mathbf{x})}(t) = -\beta(t)(\mathbf{x} - \mathbf{w}_{I(\mathbf{x})}(t)).$$

3. Other input regions are left unchanged with $\Delta \mathbf{w}_j(t) = 0$. where $\beta(t)$ is a learning rate that decreases with the number of iterations/epochs of training.