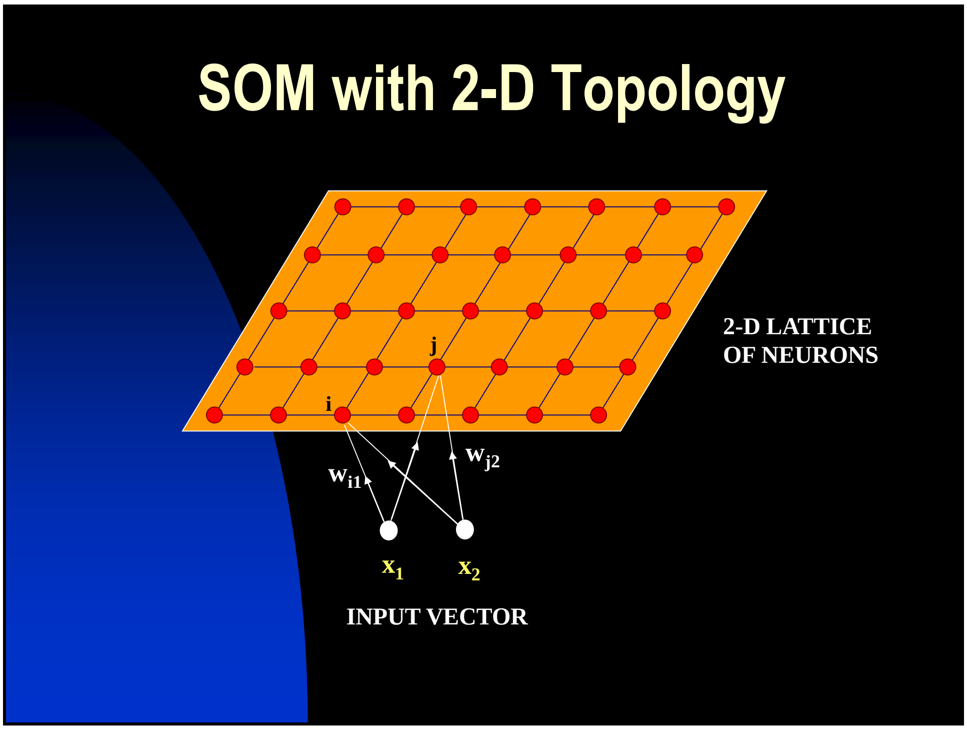
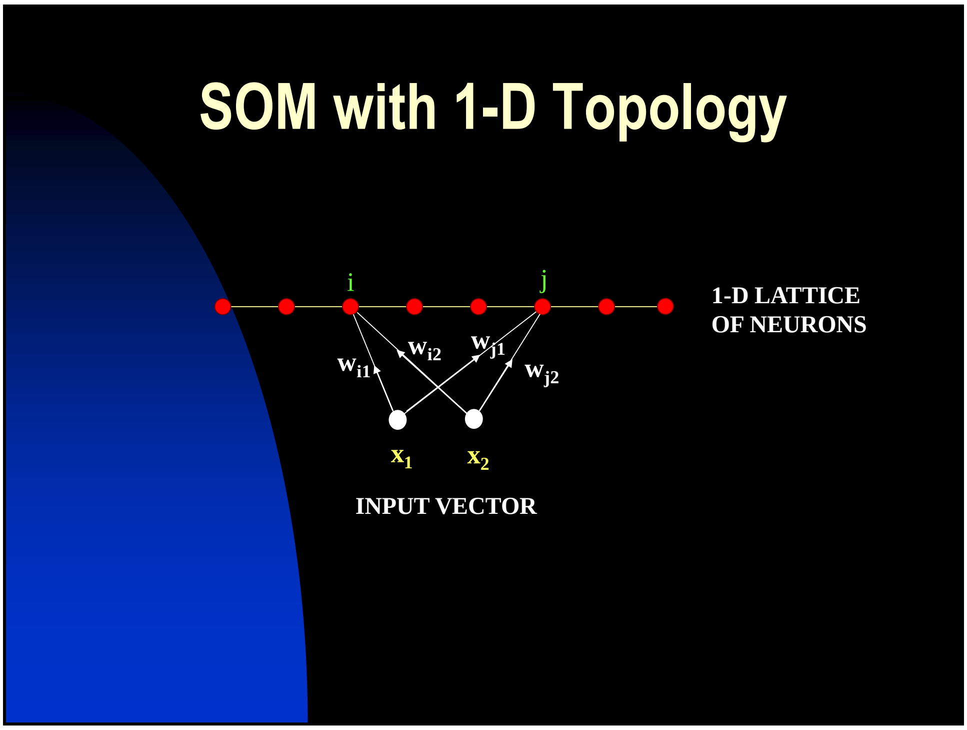
SELF-ORGANIZING MAPS

Self-Organizing Maps (SOMs)

- They are *single-layered* neural networks
- Neurons usually arranged in a *1-D* or *2-D topological structure* through lateral connections
- SOMs use *competitive learning* to adapt neurons
- They belong to the *unsupervised learning* category of neural networks
- These networks are used in *vector quantization* and *clustering* applications


SOM with 2-D Topology
j
i
2-D LATTICE
OF NEURONS
w_{i1}
w_{j2}
x_1
x_2
INPUT VECTOR


SOM with 1-D Topology
i
j
1-D LATTICE
OF NEURONS
w_{i1}
w_{i2}
w_{j1}
w_{j2}
x_1
x_2
INPUT VECTOR

Winner Selection

- Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ be the input vector in R^n
- To each neuron $\mathbf{i} = (i_1, i_2)$ $\exists$ an associated weight vector $\mathbf{w_i} = (w_{i1}, w_{i2}, \ldots, w_{in})^T \in R^n$
- Neuron $\mathbf{c} = (c_1, c_2)$ whose weight vector $\mathbf{w_c}$ is closest to the input (e.g. in Euclidean distance sense) is termed the *winner* of the competition:

$$\mathbf{c} = \arg\min_{\mathbf{i}} \{\|\mathbf{x} - \mathbf{w_i}\|\}$$

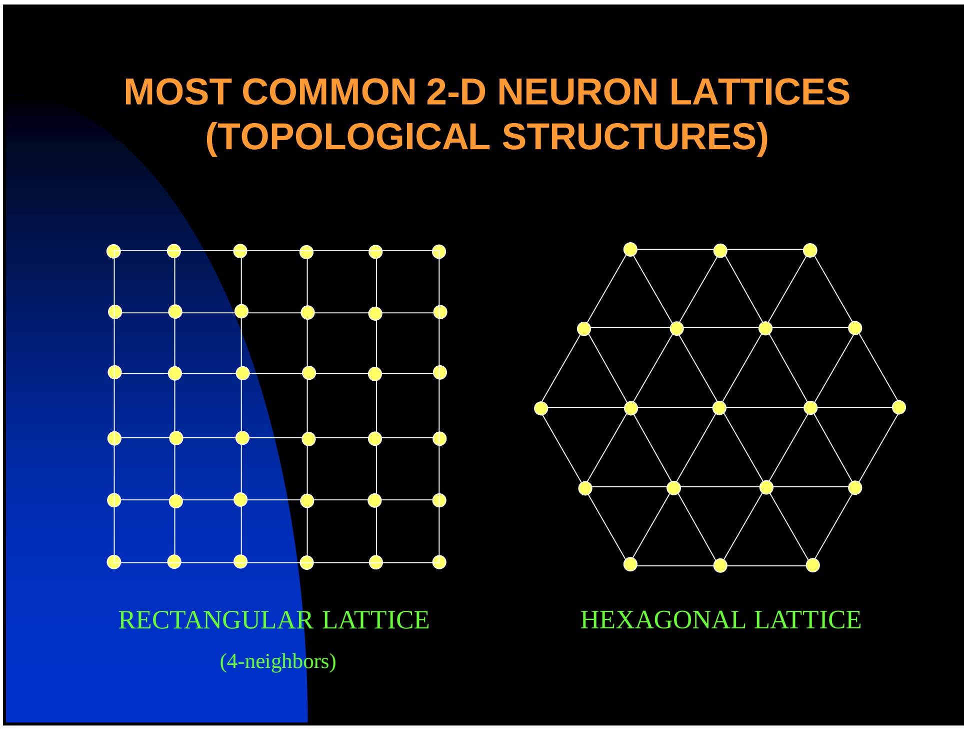
Weight Adaptation

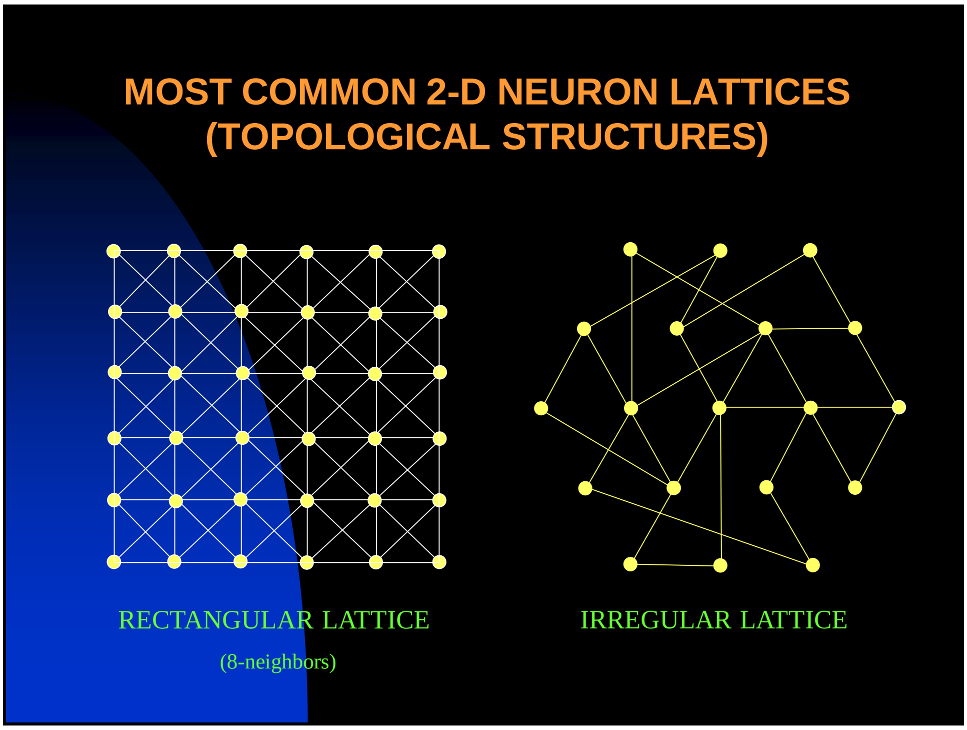
- The weights of the winner and of neurons in its topological neighborhood $N_c(t)$ are adapted according to the following equation:

$$\mathbf{w_i}(t+1) = \mathbf{w_i}(t) + \alpha(t)\,\Lambda(i, N_c(t))\,[\mathbf{x}(t) - \mathbf{w_i}(t)] \qquad \forall \mathbf{i} \in N_\mathbf{c}(t)$$

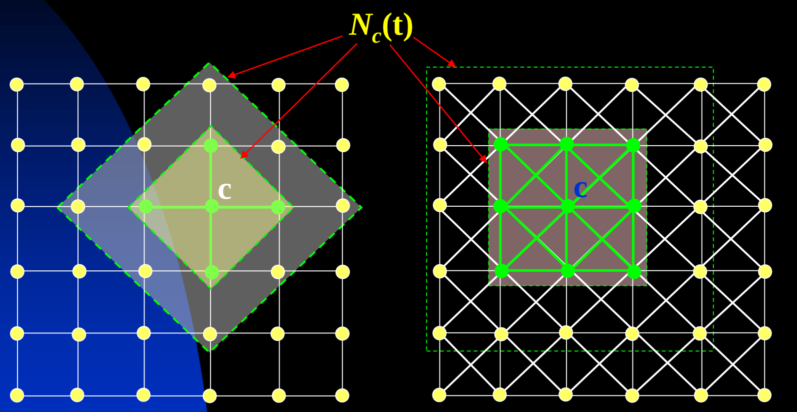
with

$$\mathbf{w_i}(t+1) = \mathbf{w_i}(t) \qquad \forall \mathbf{i} \notin N_\mathbf{c}(t)$$


MOST COMMON 2-D NEURON LATTICES
(TOPOLOGICAL STRUCTURES)
RECTANGULAR LATTICE
(4-neighbors)
HEXAGONAL LATTICE


MOST COMMON 2-D NEURON LATTICES
(TOPOLOGICAL STRUCTURES)
RECTANGULAR LATTICE
(8-neighbors)
IRREGULAR LATTICE

Topological Neighborhoods

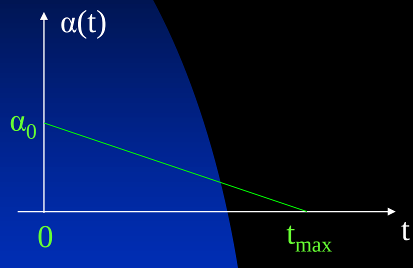


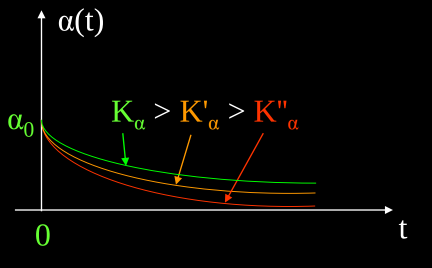
ROMBIC NEIGHBORHOODS

SQUARE NEIGHBORHOODS

Adaptation Gain $\alpha(t)$

The adaptation gain (or learning rate) is chosen to decay with time. Two popular functions for α(t) are:





$\alpha(t) = \alpha_0\,(1 - t/t_{max})$

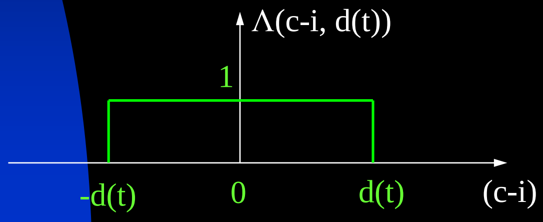
$\alpha(t) = \alpha_0 / (1 + K_\alpha\, t)$

$0 < \alpha_0 < 1$; typically: $0 < K_\alpha < 10^{-2}$

Neighborhood Function Λ(·)

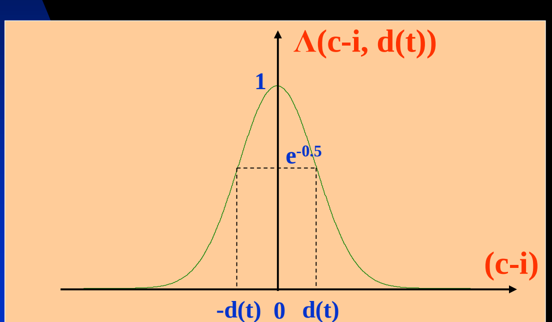
- 1-D maps – rectangular neighborhood function

$$\Lambda(c-i, d(t)) = \begin{cases} 1 & \text{if } |c-i| \le d(t) \text{ i.e. } i \in N_c(t) \\ 0 & \text{otherwise} \end{cases}$$



- 1-D maps – Gaussian neighborhood function

$$\Lambda(c-i, d(t)) = \exp\left(-\frac{(c-i)^2}{2d^2(t)}\right)$$



- 2-D maps – rectangular neighborhood function

$$\Lambda(\|c-i\|_\infty, d(t)) = \begin{cases} 1 & \text{if } \|c-i\|_\infty \le d(t) \\ 0 & \text{otherwise} \end{cases}$$

where, $\|c-i\|_\infty = \max\{|c_1-i_1|, |c_2-i_2|\}$

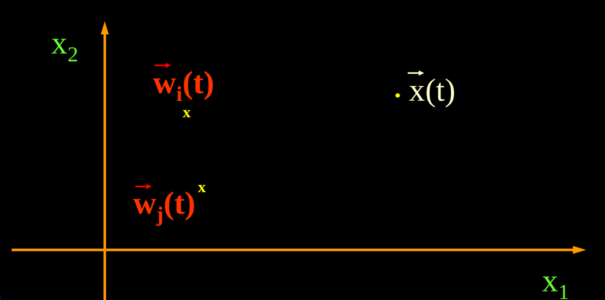
- 2-D maps – Gaussian neighborhood function

$$\Lambda(\|c-i\|_2, d(t)) = \exp\left(-\frac{\|c-i\|_2^2}{2d^2(t)}\right)$$

Graphical Interpretation of Adaptation

Assume that $\mathbf{i} \in N_c(t)$ and $\Lambda(i, N_c(t)) = 1$. In this case the weight adaptation equation becomes:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t)\,[\mathbf{x}(t) - \mathbf{w}_i(t)]$$

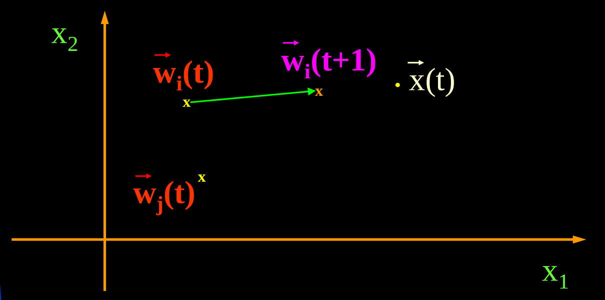


Input and Weight Space

Graphical Interpretation of Adaptation

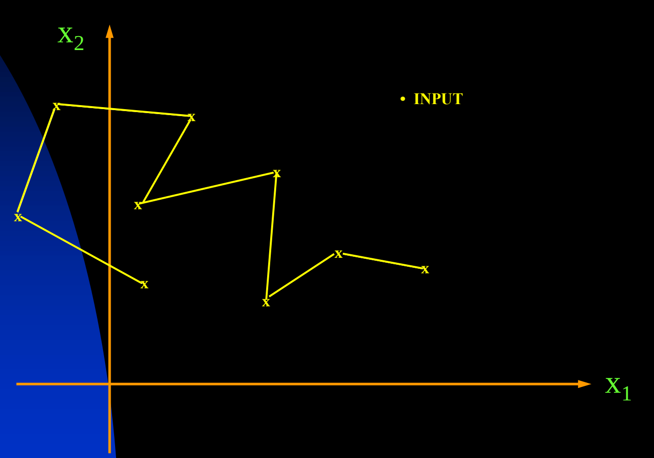
Assume that $\mathbf{i} \in N_c(t)$ and $\Lambda(i, N_c(t)) = 1$. In this case the weight adaptation equation becomes:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t)\,[\mathbf{x}(t) - \mathbf{w}_i(t)]$$

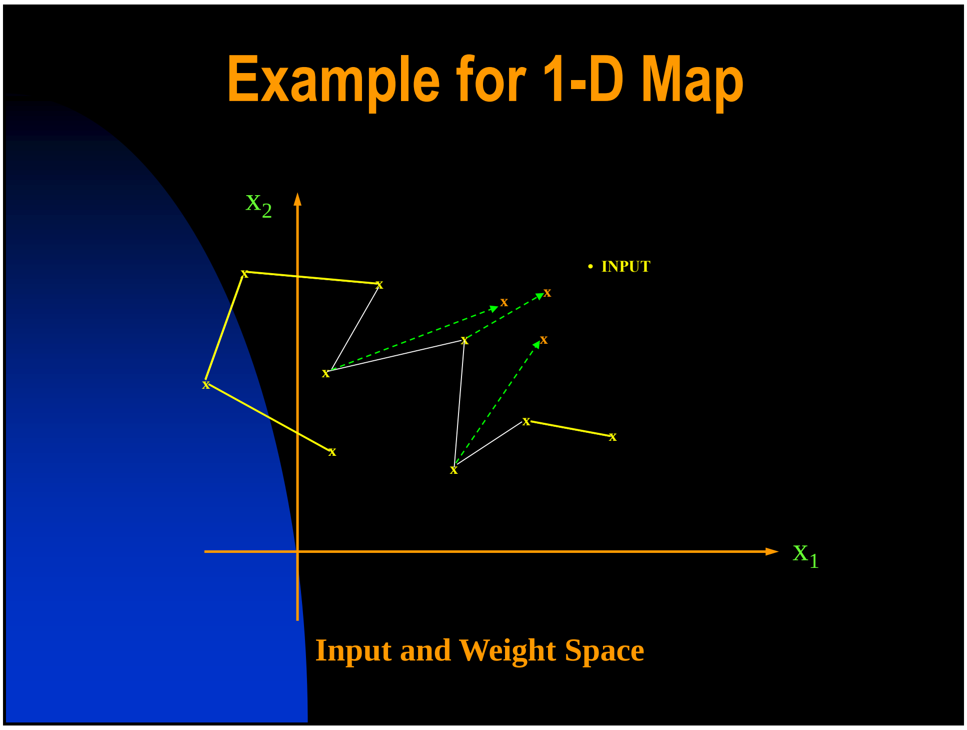


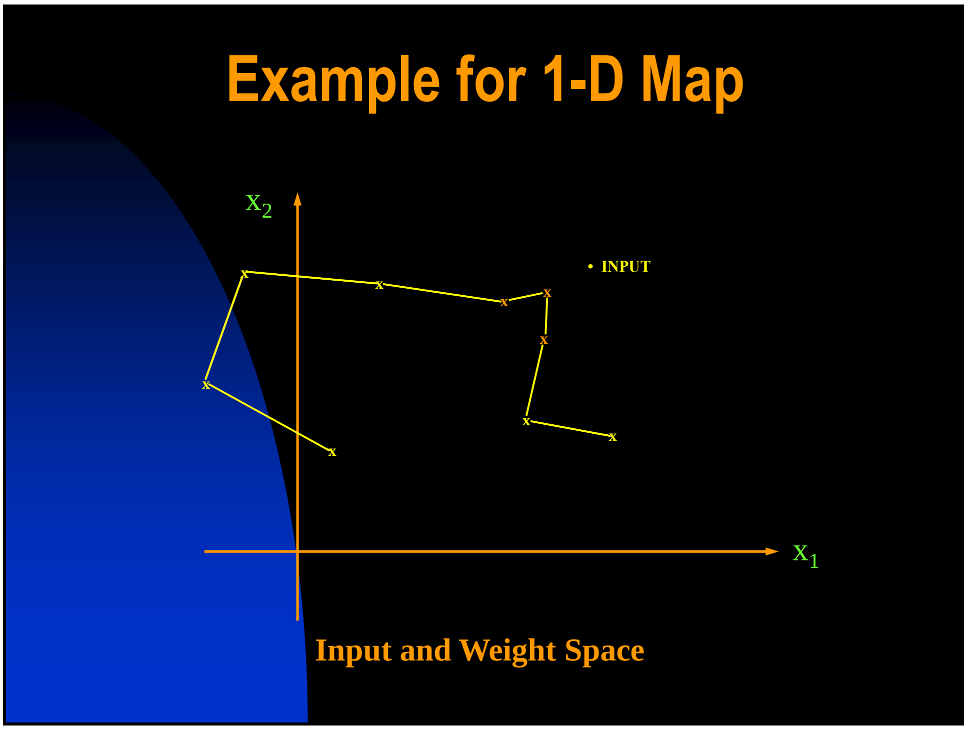
Input and Weight Space

Example for 1-D Map



Input and Weight Space


Example for 1-D Map
X2
• INPUT
X1
Input and Weight Space


Example for 1-D Map
X2
• INPUT
X1
Input and Weight Space

Loop of Training Algorithm: pass *k*

1. Pick an input vector $x(k)$ at random from the training set (a subset of the whole data set)
2. Compute the distances from all neurons to the input vector: $y_i(k) = ||x(k) - w_i(k)||_2$
3. Find the winning neuron, according to:

$$c(k) = \arg\min_i y_i(k)$$

4. Update the neurons in the neighborhood $N_c(k)$ of the winner:

$$w_i(k+1) = w_i(k) + \alpha(k)\ \Lambda(i, N_c(k))\ [x(k) - w_i(k)]$$

Phases of SOM Training

1. Self-organization phase.

   Even though the map may start from a random initial state, it self-organizes through time.

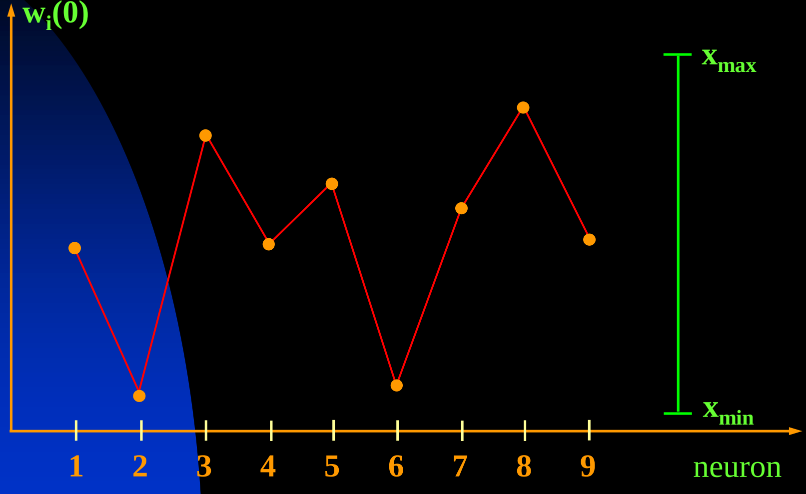
   The map preserves the topology from the output neuron array to the input space in the sense that nearby units respond to nearby stimuli.

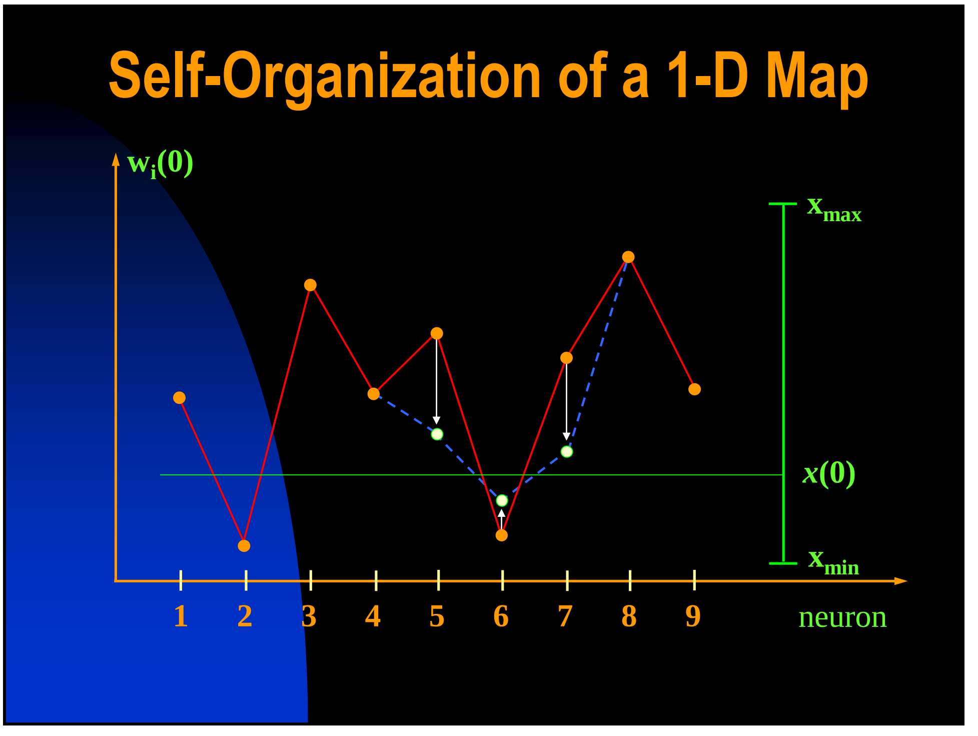
2. Convergence phase.

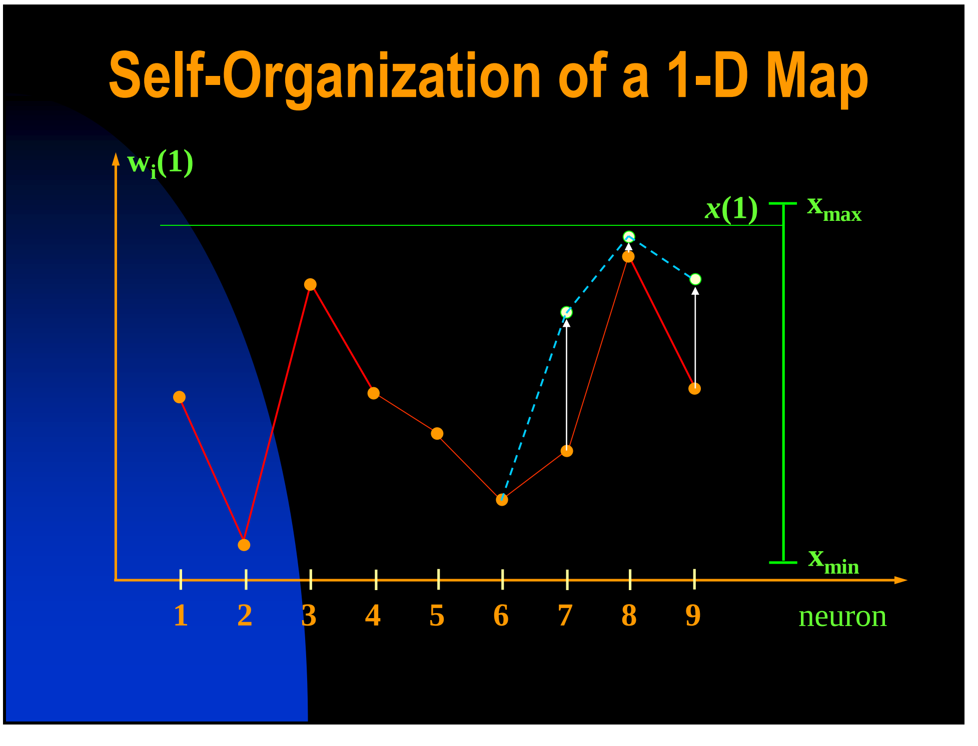
During this phase the weight vectors tend towards their asymptotic values and reproduce the input probability distribution as closely as possible.

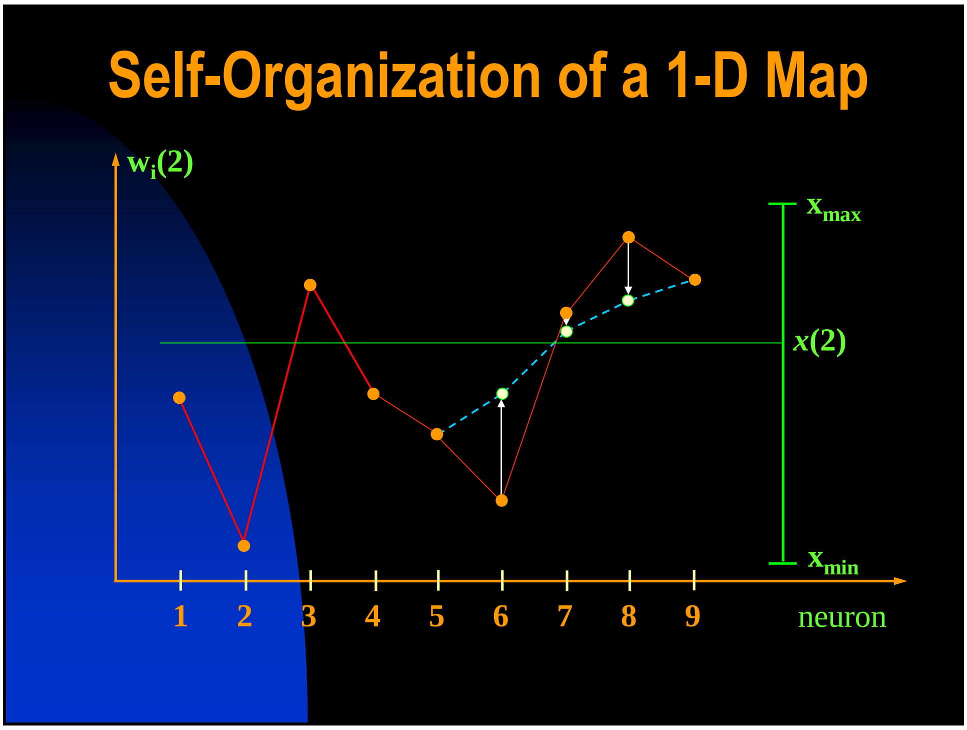
This phase has a much longer duration than self-organization.

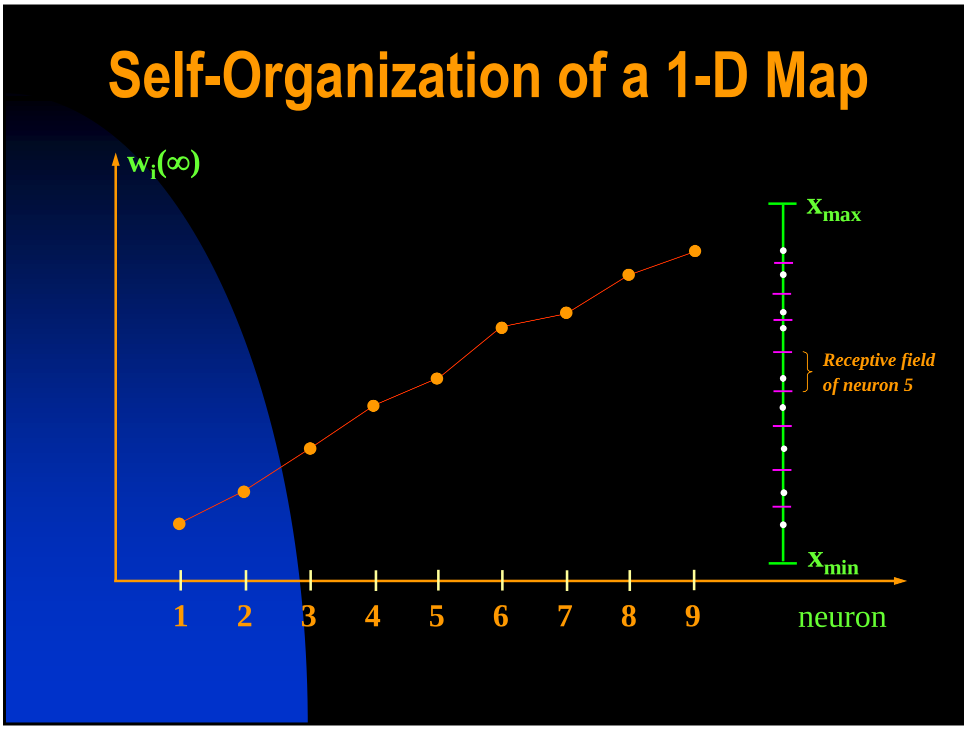
Self-Organization of a 1-D Map

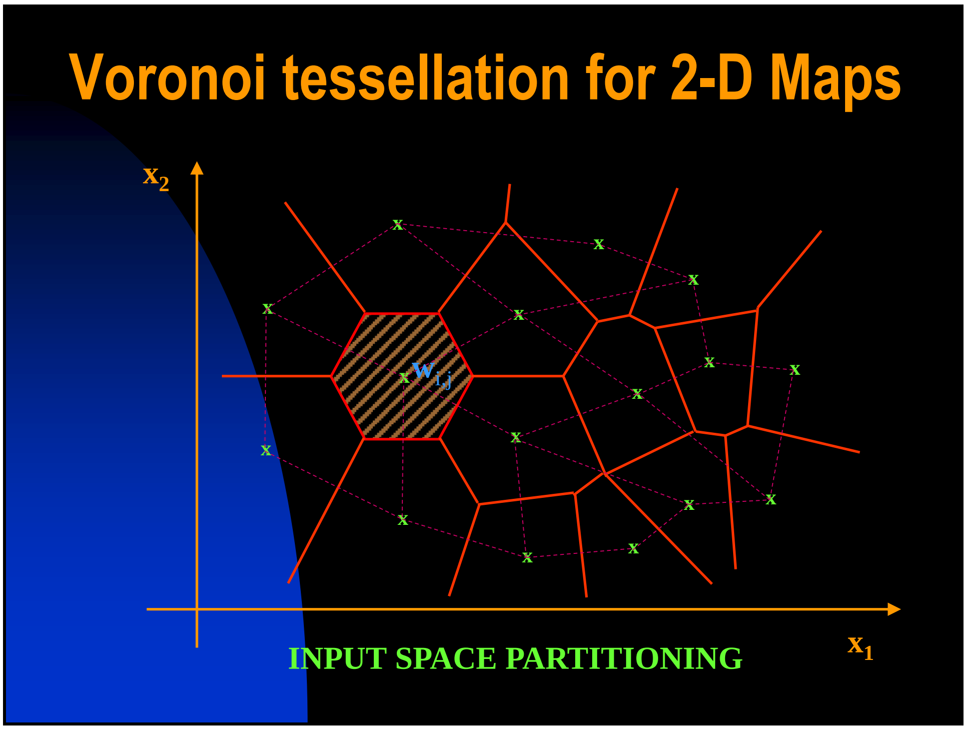


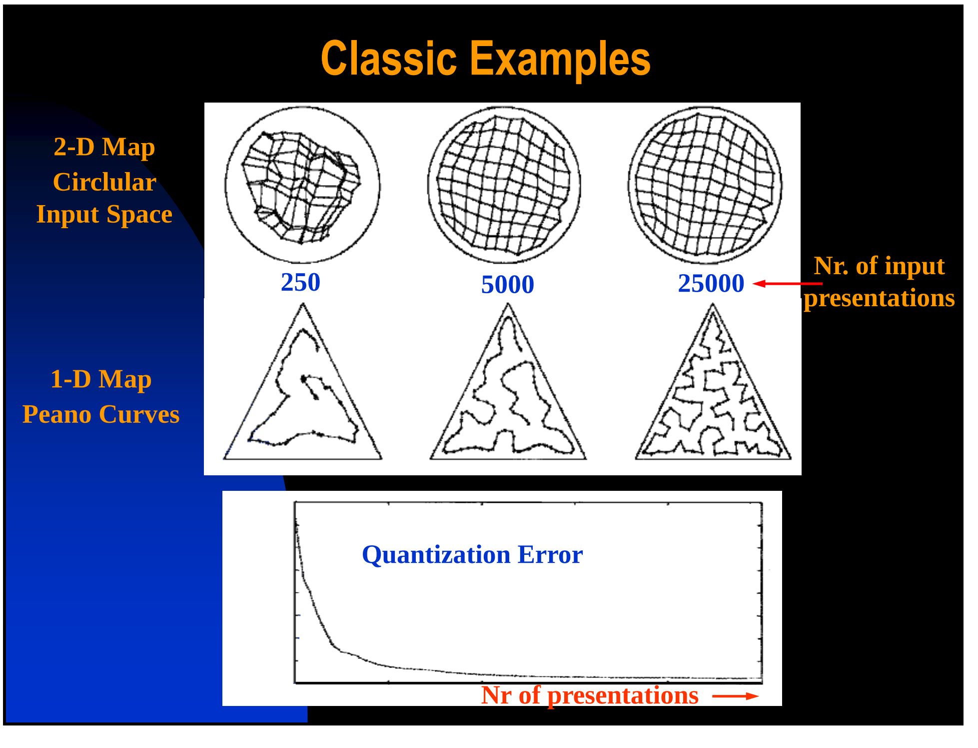

Self-Organization of a 1-D Map
wi(0)
xmax
x(0)
xmin
1 2 3 4 5 6 7 8 9
neuron


Self-Organization of a 1-D Map
wi(1)
x(1)
xmax
xmin
1 2 3 4 5 6 7 8 9
neuron

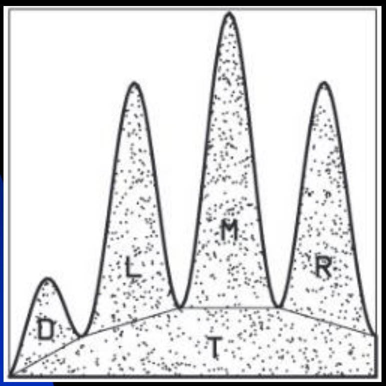

Self-Organization of a 1-D Map
wi(2)
xmax
x(2)
xmin
1 2 3 4 5 6 7 8 9
neuron


Self-Organization of a 1-D Map
wi(∞)
xmax
Receptive field
of neuron 5
xmin
1 2 3 4 5 6 7 8 9
neuron


Voronoi tessellation for 2-D Maps
x_2
$w_{i,j}$
x_1
INPUT SPACE PARTITIONING

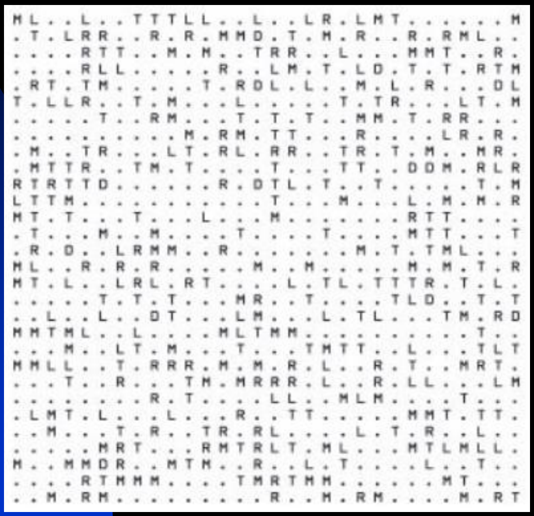

Classic Examples
2-D Map
Circlular
Input Space
1-D Map
Peano Curves
250
5000
25000
Nr. of input presentations
Quantization Error
Nr of presentations

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



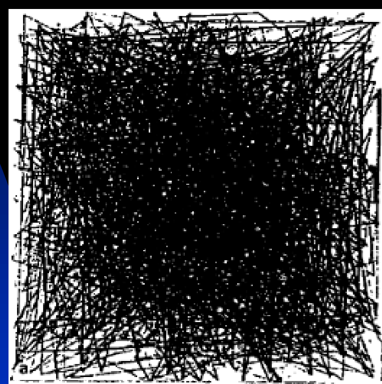
Model hand surface with touch receptors. D, L, M, R, T denote the five subregions. The dots mark the locations of 1,200 touch receptors distributed at random over the hand surface.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



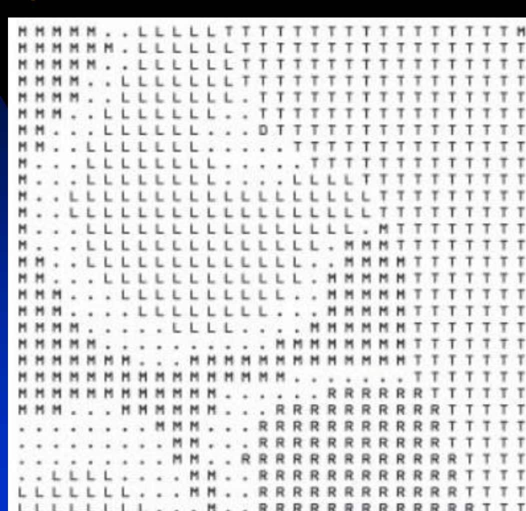
Initial assignment of the 30x30 lattice neurons.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



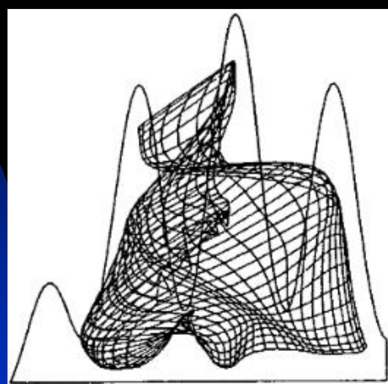
Initial weight configuration.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



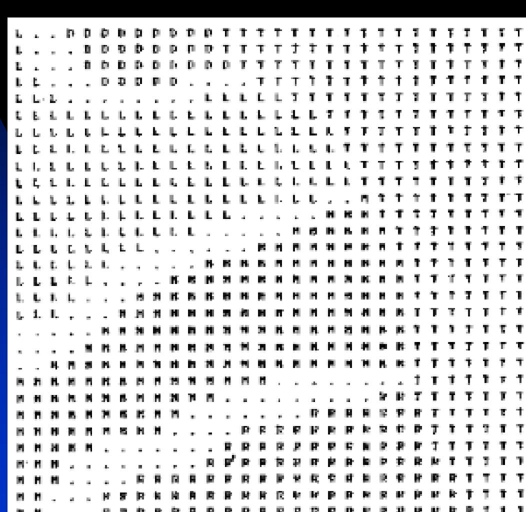
Coarsely ordered assignment after 500 "touch stimuli".

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



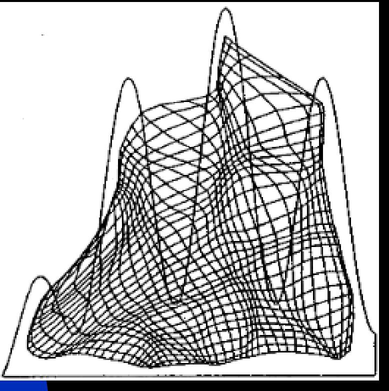
Weight configuration after 500 input presentations.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



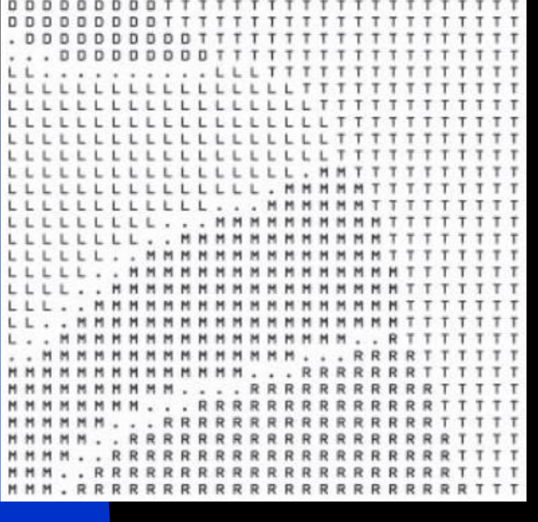
Assignment after 3000 "touch stimuli".

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



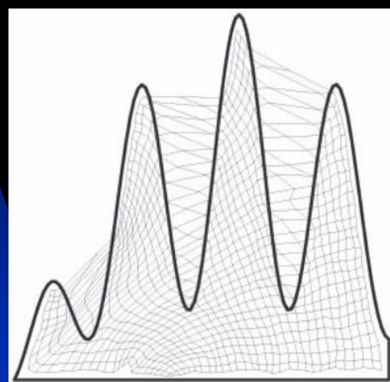
Weight configuration after 3000 input presentations.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



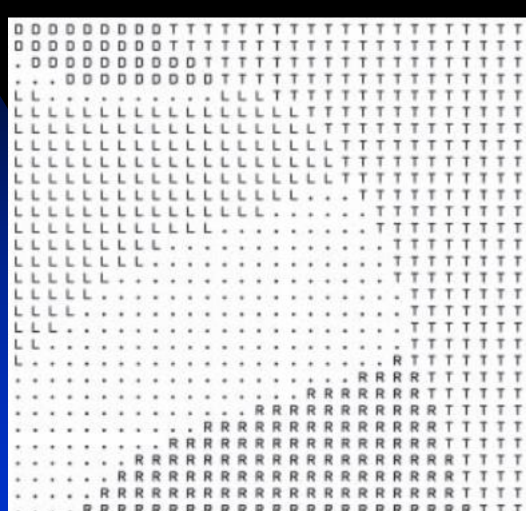
Final assignment after 20,000 "touch stimuli".

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



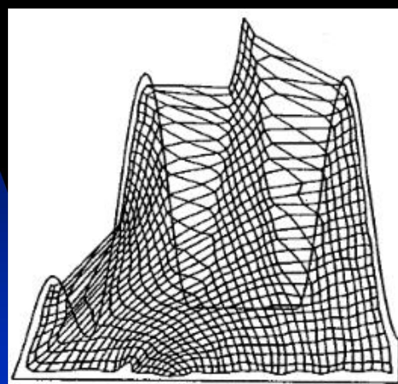
Final neuron lattice (after 20,000 presentations).

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



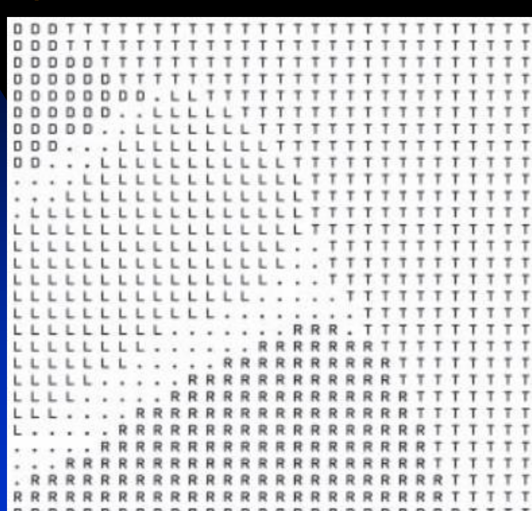
Map after middle finger (M) amputation.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



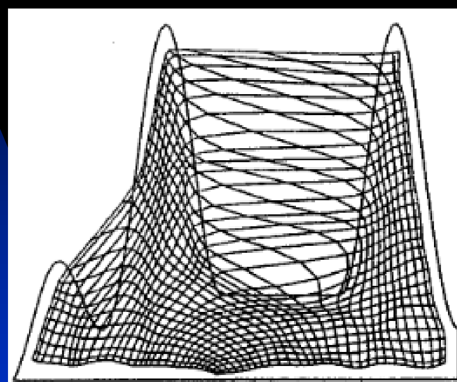
Input space after middle finger (M) amputation.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



Readapted map after another 50,000 iterations.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)



Final weight configuration after another 50,000 input presentations.

Modeling of the somatosensory map of the hand (Ritter, Martinetz & Schulten, 1992)

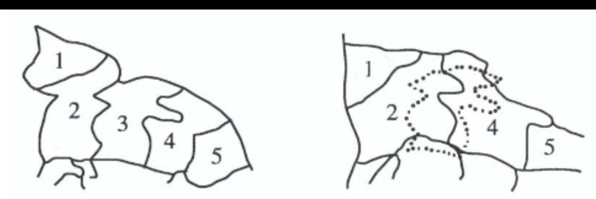
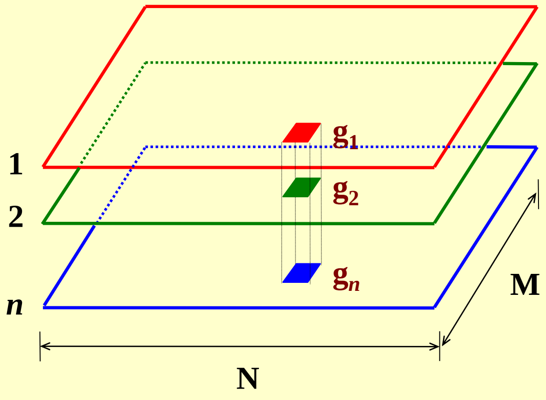


Abb. 7.8: Readaptation of the somatosensory map of the hand region of an adult nocturnal ape due to the amputation of one finger. (a) (left) Before the operation, each finger in the map is represented as one of the regions 1-5. (b) (right) Several weeks after amputation of the middle finger, the assigned region 3 has disappeared, and the adjacent regions have correspondingly spread out (after Fox, 1984).

SOM Application to Remotely-Sensed Data

LANDSAT SATELLITE IMAGE OVER LESVOS ISLAND IN GREECE: Bands TM3, TM2, TM1 -- 256 gray levels/band

Representation of multispectral satellite images of *MxN* pixels and *n* bands.

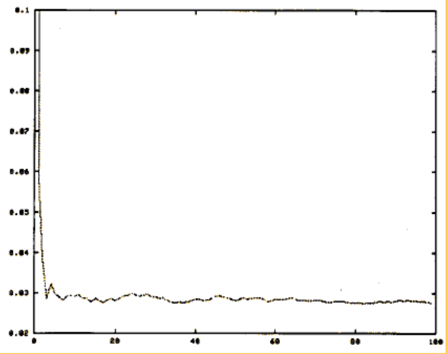


Spectral signature (or stacked vector)

$$\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \in \Re^n$$

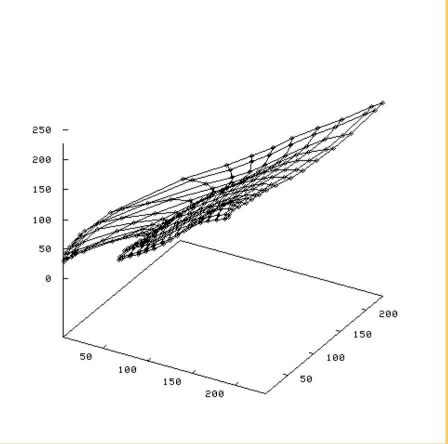
Input to SOM

SOM TRAINING - FINAL 16x16 MAP



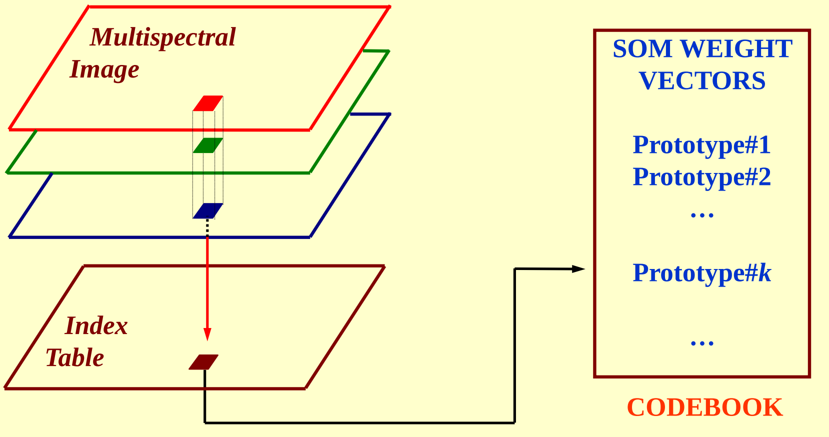
iterations (x1000)

SOM training error (on 1% of image pixels)



Final Map of 16x16 neurons

Representing Multispectral Images by Index Table and SOM prototypes



COMPRESSION

- **Store only the index and the SOM prototypes.**
- **For a 16x16 map we need 8-bit indices. Hence, for *n* data bands the compression ratio will approximately be:**

$$CR \approx n : 1$$

- **Further compression can be achieved using variable-length encoding on the index itself (e.g. Huffman coding, LZW compression, e.t.c.).**

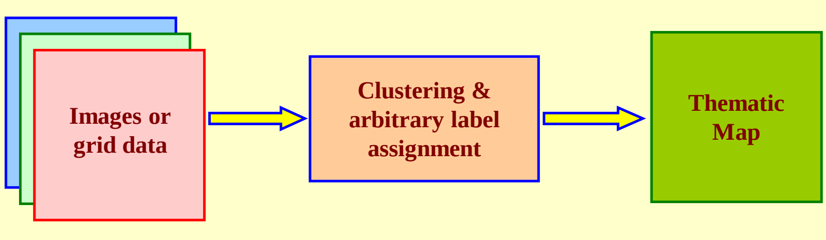
Original and Quantized Images

Original 512x512 Landsat image
(3 Bands, 256 Gray Levels)

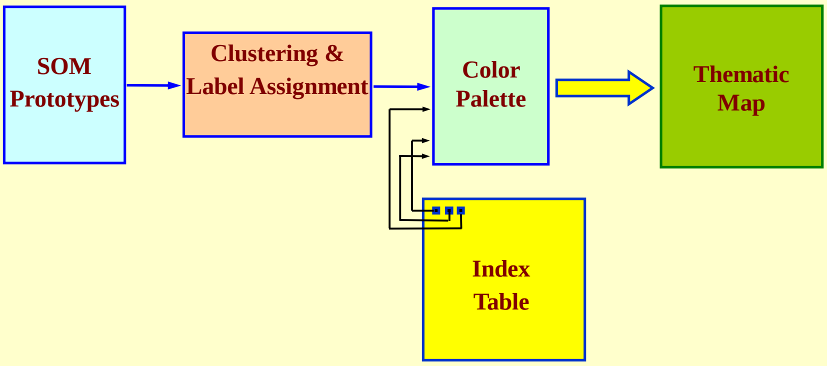
Vector quantized image
(16x16 Self-Organizing Map)

Efficient Automatic Classification
(through Data Clustering)

- **Classic Approach:** cluster the original data and assign arbitrary labels (colors) to each cluster. Then, colorize each pixel with the color of the cluster it belongs to produce the final classification result (thematic map).

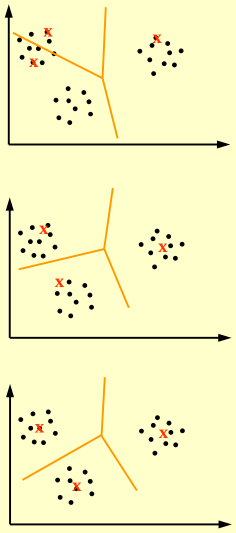


- **Proposed Approach:** apply the clustering algorithm to the codebook (SOM prototypes) rather than to the original data and assign arbitrary colors to each cluster to obtain a color palette. Then, use the index to produce the final classification result.

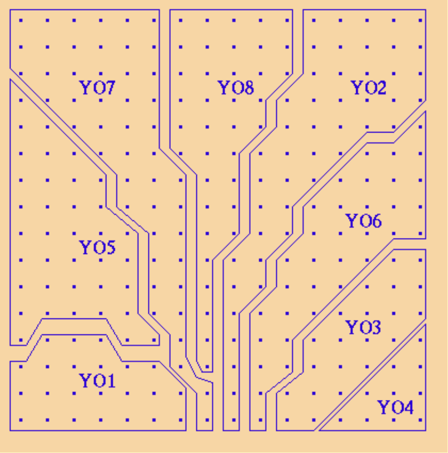


The *k*-Means Clustering Algorithm

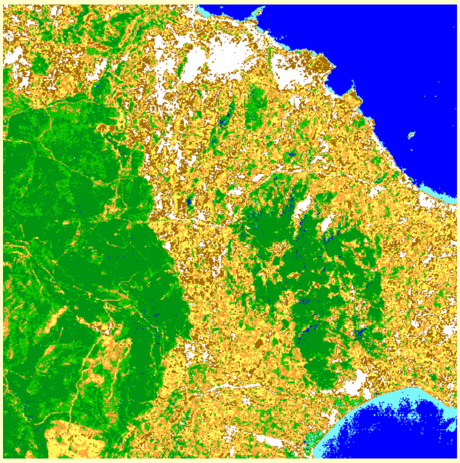
1. Choose desired number of clusters *k* and *k* initial prototypes (usually from the available data).
2. Classify data points to the category of their closest prototype adding them to the list of their closest prototype.
3. Compute new prototypes as the mean values of data samples in their lists.
4. If no prototype has changed or the termination condition is satisfied, stop. Otherwise, go back to step 2.



Automatic Classification using 16x16 SOM

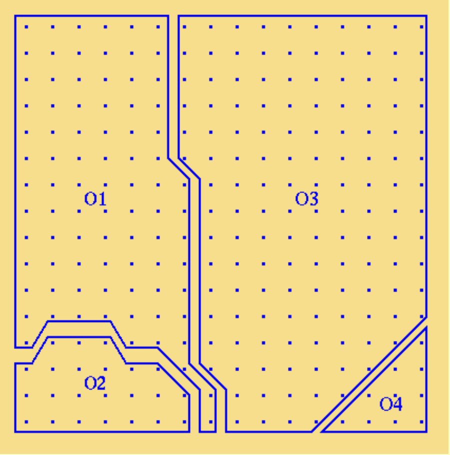


k-means: 8 neuron clusters

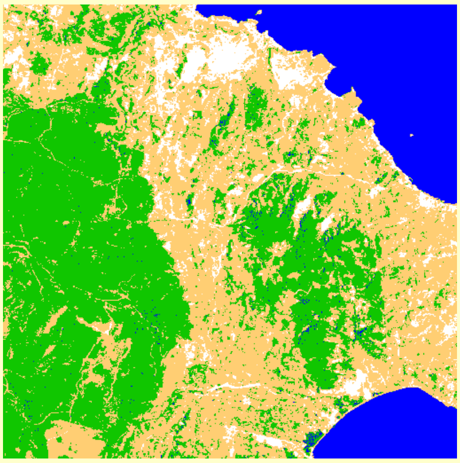


Thematic Map (8 categories)

Merging clusters to reduce to 4 classes



Manual cluster merging



Thematic Map (4 categories)

Clustering Time Comparisons

Table I: Computational times and clustering gain (per iteration) for a map of 16x16 neurons. Experiments run on a SUN ULTRA II Enterprise workstation (64MB, 167MHz).

CLUSTERING ALGORITHM	CLASSICAL METHOD	PROPOSED METHOD	SPEEDUP
k-MEANS	2.74sec	2.67msec	1026
FUZZY ISODATA	10.84sec	9.67msec	1121
HIERARCHICAL CLUSTERING	∞	11.90sec	∞