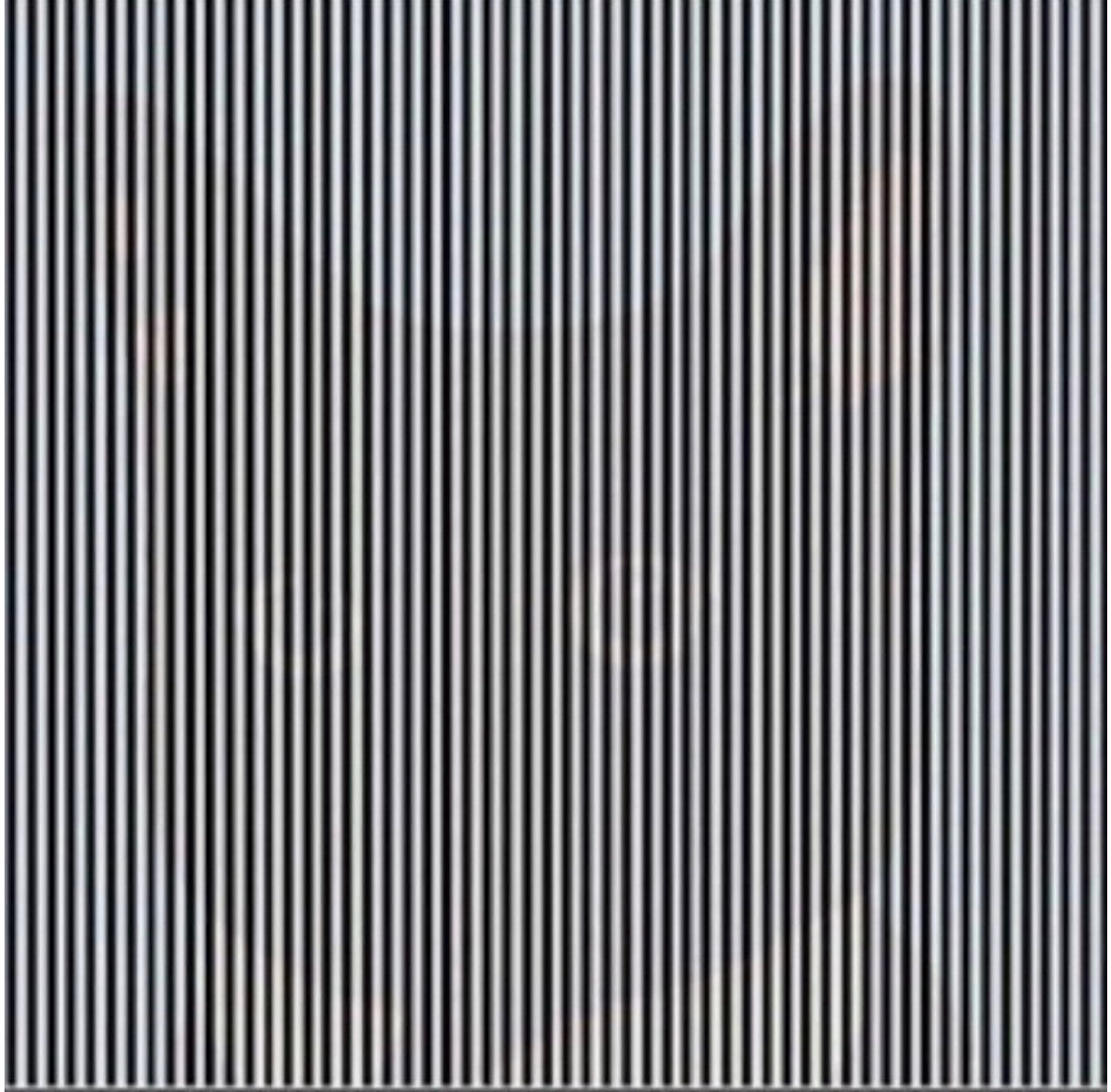
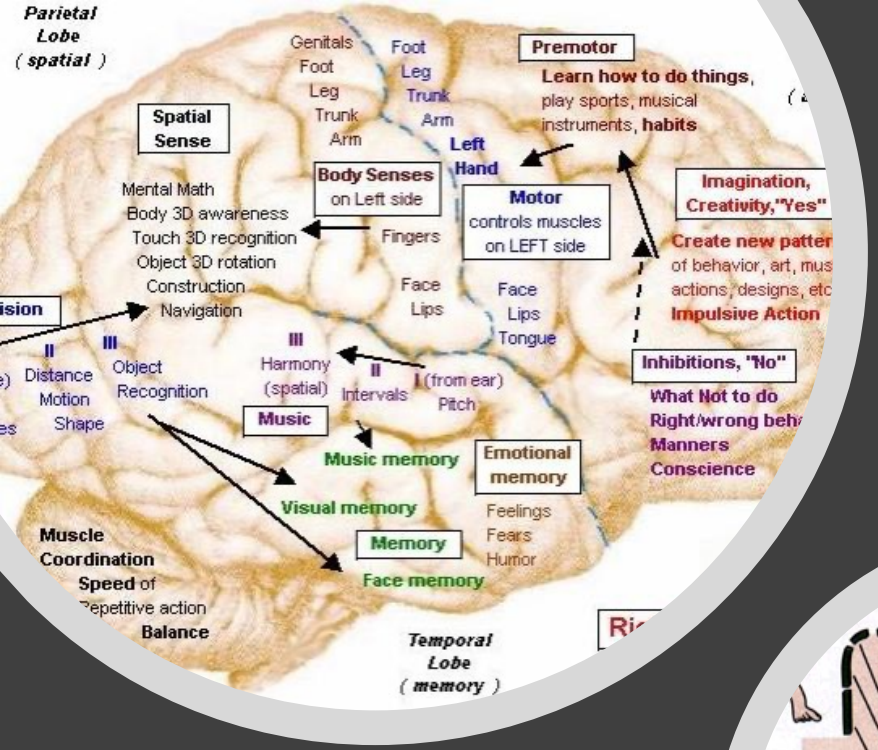
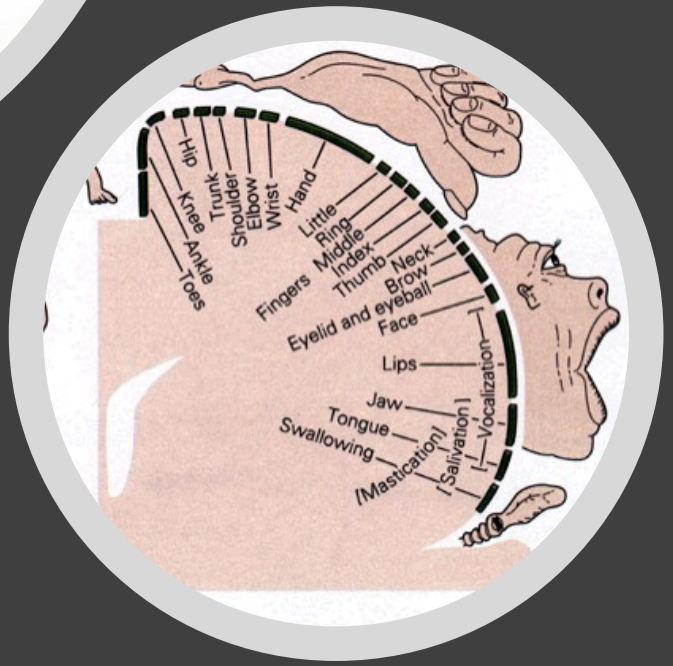
Self-Organized Learning

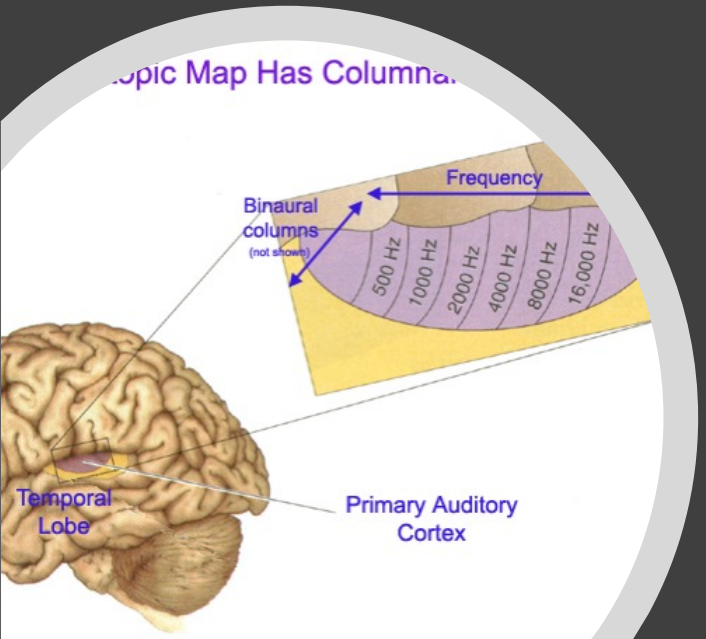
Self-Organized Maps

Simple Competitive Learning





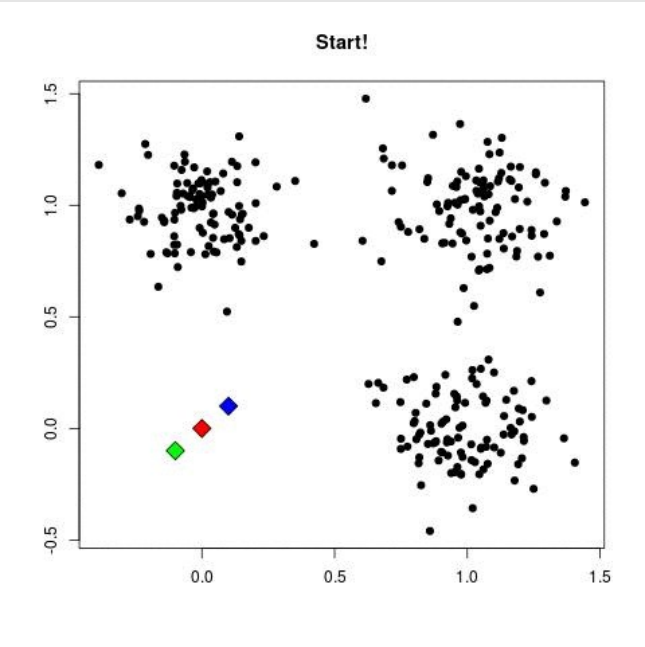


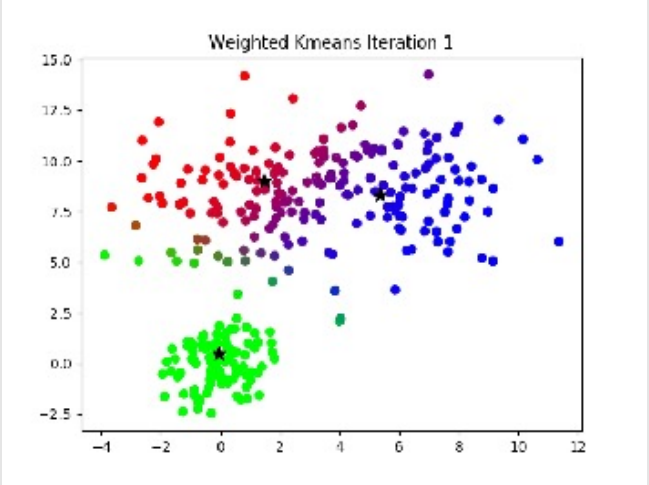


Self-Organized Learning

- Self-Organized Learning is term used to describe algorithms that organize their resources (prototype vectors) to describe the data
- Various areas of the brain are organized spatially according to different sensory modalities
- Data organized spatially according frequency

Self-Organized Learning





- "Self-organized learning", generally means one of two paradigms developed by Teuvo Kohonen
  - Self-Organizing Maps
  - Learning Vector Quantization
- One may consider other methods as Self-Organized Learning
  - K-Means, Soft K-Means, K-Mediods Expectation Maximization

Self-Organized Learning

- *Competitive learning* neural networks
  - Each PE(node, neuron) receives the same information
  - PEs “compete” and a “winner” is determined
  - Different inputs result in different winners

Self-Organized Learning

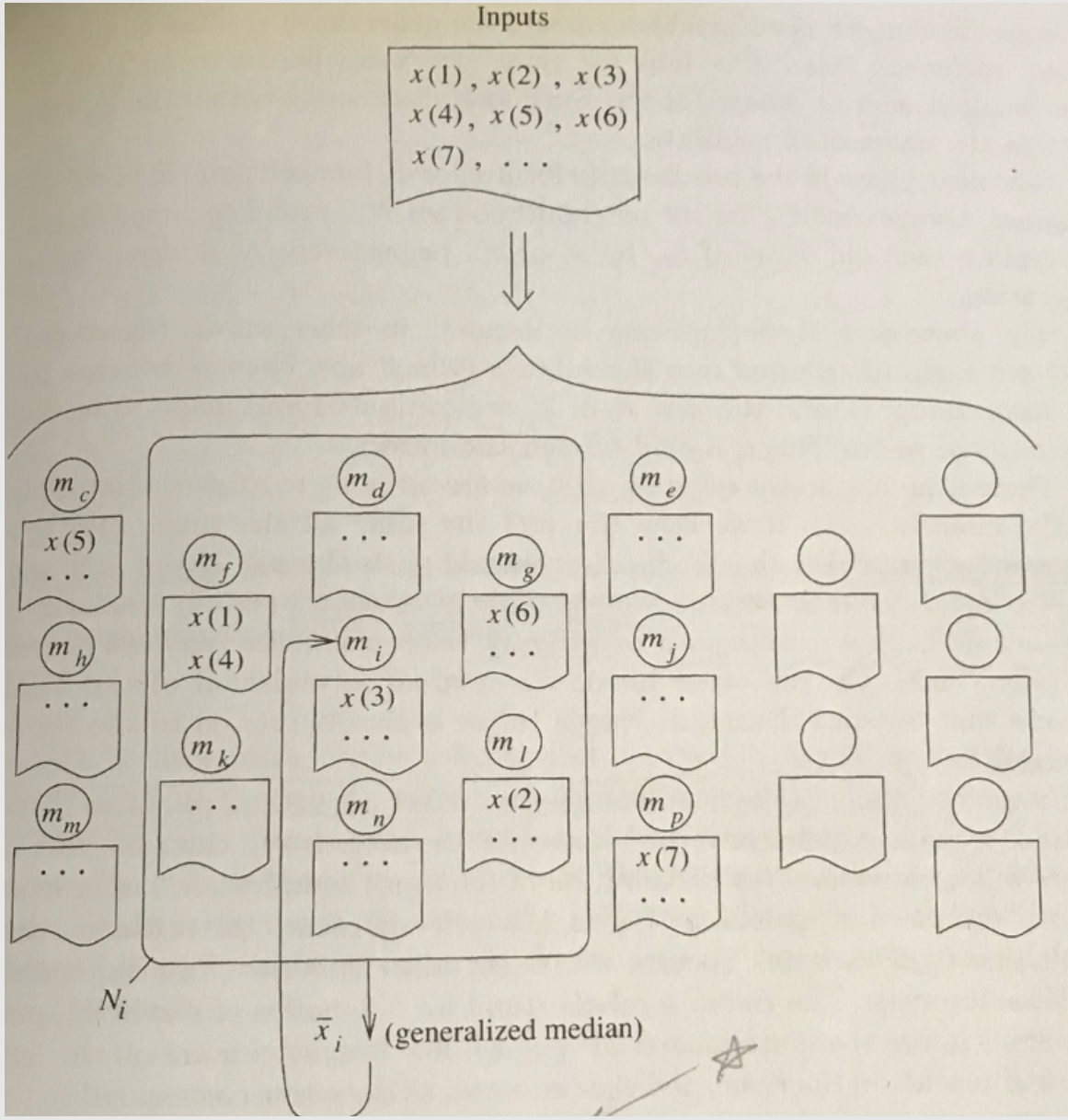
- Tool for visualizing High dimensional data
- Clustering algorithm
- Can become a classifier
- Produces a similarity graph of input data
- Great for:
  - Process analysis
  - Machine perception
  - Control
  - Communication

Self-Organized Learning

- Formal definition:
  - A nonlinear, ordered, smooth mapping of high-dimensional input data manifolds onto the elements of a regular, low-dimensional array.
  - Resembles vector quantization
- The Basics:
  - Input variables $\{\xi_j\}$ Where $x = [\xi_1, \xi_2, \xi_3, \dots, \xi_n]^T \in \Re^n$
  - Each element $m_i = [\mu_{i1}, \mu_{i2}, \dots, \mu_{in}]^T \in \Re^n$ called a model
  - and $d(x, m_i)$
  - $m_c \;\; ; \; c = arg \min_i \{d(x, m_i)\}$

Self-Organized Learning

- THE OPERATION:



Self-Organized Learning

- The description of operation from the previous slide is iterated on again with all input data
- Question: Does this ever converge?

Self Organized Learning

Self-Organizing Map

- Output Neurons
  - Compete to be activated (fired)

There Can Be Only ONE.

- Overview:

neurons are placed at nodes of a lattice, As a neurons win, they become ordered with respect to each other.

Self-Organizing Map

- Computational maps offer 4 properties
  1. Neurons act in parallel
     - Process information similar in nature but that come from different regions in the input space
  2. Each sample is kept in proper context
  3. Neighboring neurons associate with like samples and interact with short synaptic connections
  4. Contextual maps are decision-reduced maps from higher dimensional spaces

Self-Organizing Map

- 2 Basic Feature Mapping Models
  - Willshaw-von der Malsburg's
  - Kohonen model

Self-Organizing Map

- Goal:

  Build artificial topographic maps that learn on their own.

- Principal of Topographic map formation:

"The spatial location of an output neuron in a topographic map corresponds to a particular domain of feature of data drawn from the input space."

Self-Organizing Map

- Purpose:
  - Transform a sample of any dimension into a one or two dimensional discrete map in a topologically ordered method.

Self-Organizing Map

- 4 principals of Self Organization
  - Competition and Cooperation
    - Adhere to two of the 4 principals
  - Principal 2:
    - The limitation of available resources in one form or another, leads to competition among the synapses of a single neuron or an assembly of neurons, with the result that most vigorously growing synapses or neurons, respectively, are selected at the expense of others.

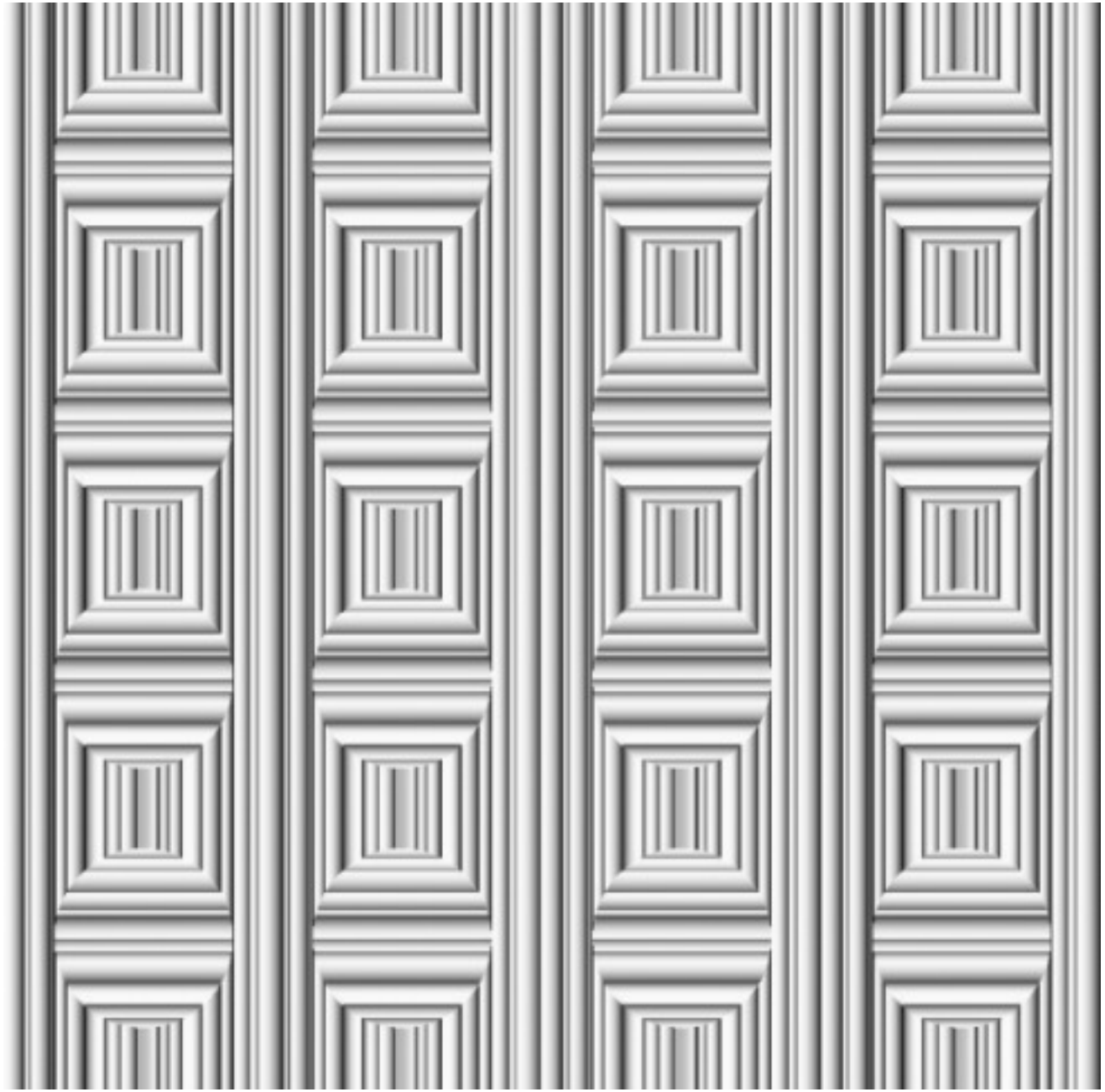
Self-Organizing Map

- Principal 3:
  - Modifications in synaptic weights at the neural level and in neurons at the network level tend to cooperate with each other.

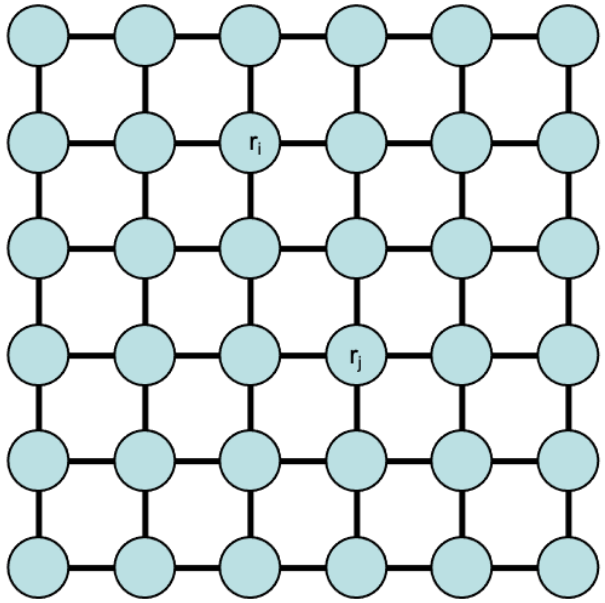
Self-Organized Learning

Self-Organized Maps

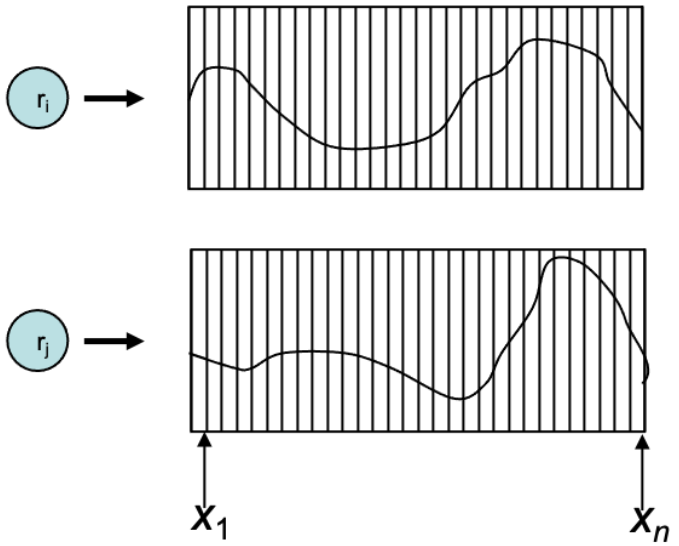
Simple Competitive Learning



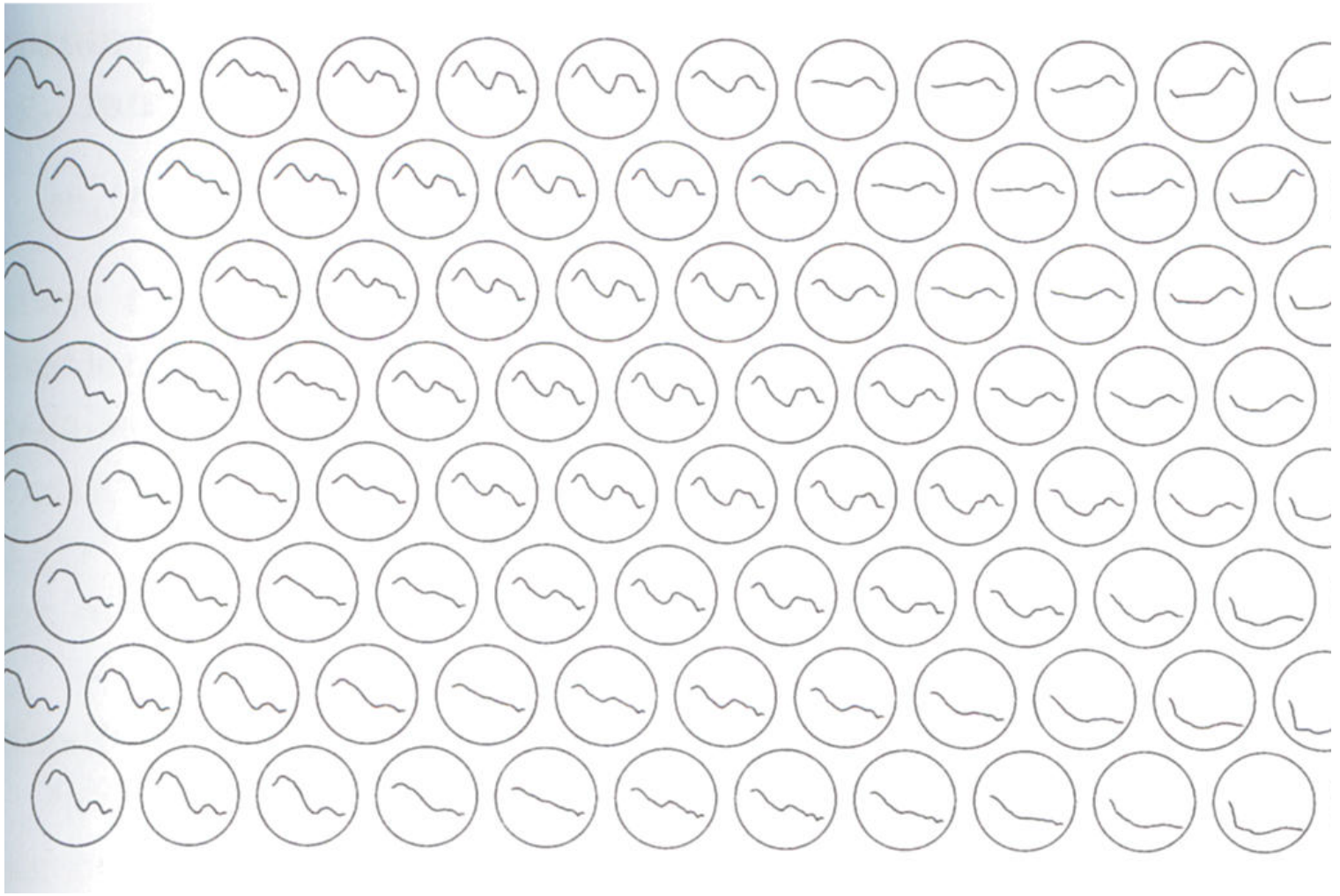
Self-Organizing Map



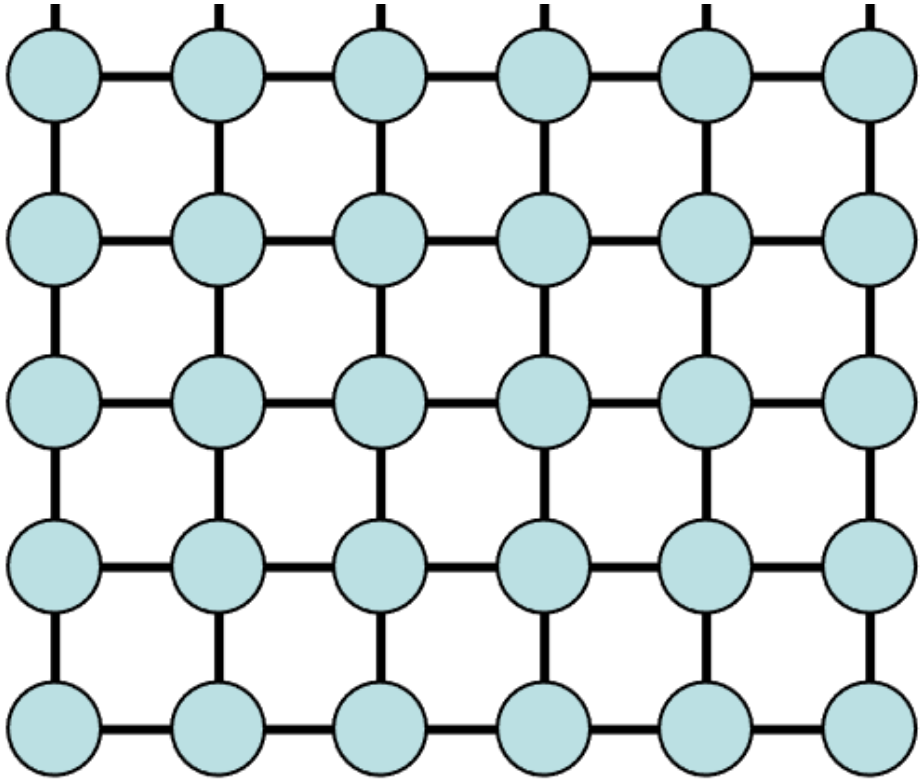
Fixed Lattice of Neuron
(Often 2D, but can be 1D,
rarely is larger such as 3D)



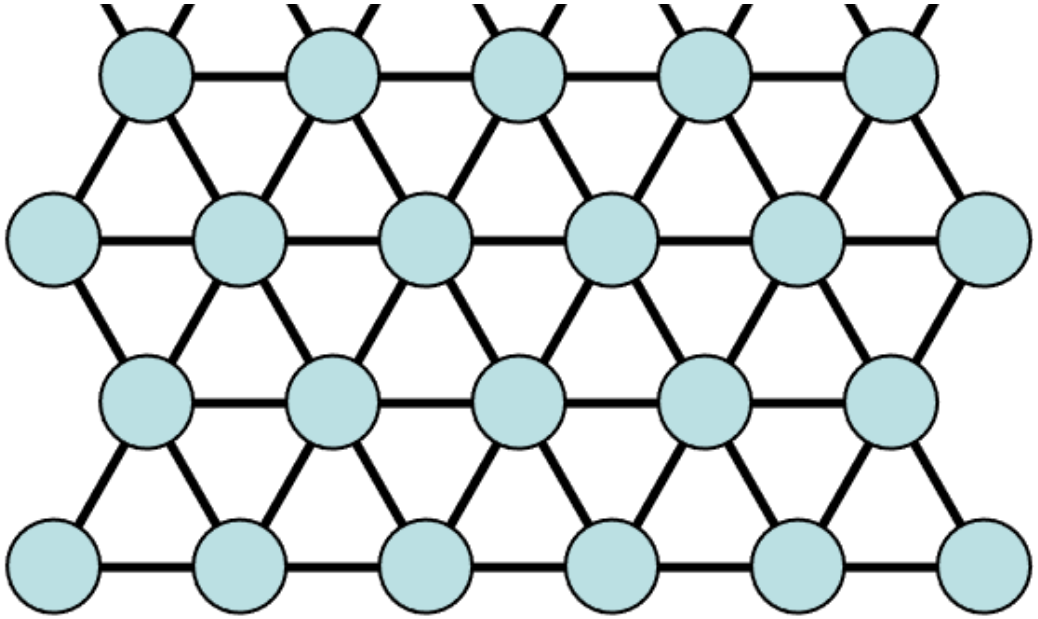
Associated Weight Vector
(*n*-dimensional, same as data)



Self-Organizing Map



Square Lattice



Hexagonal Lattice

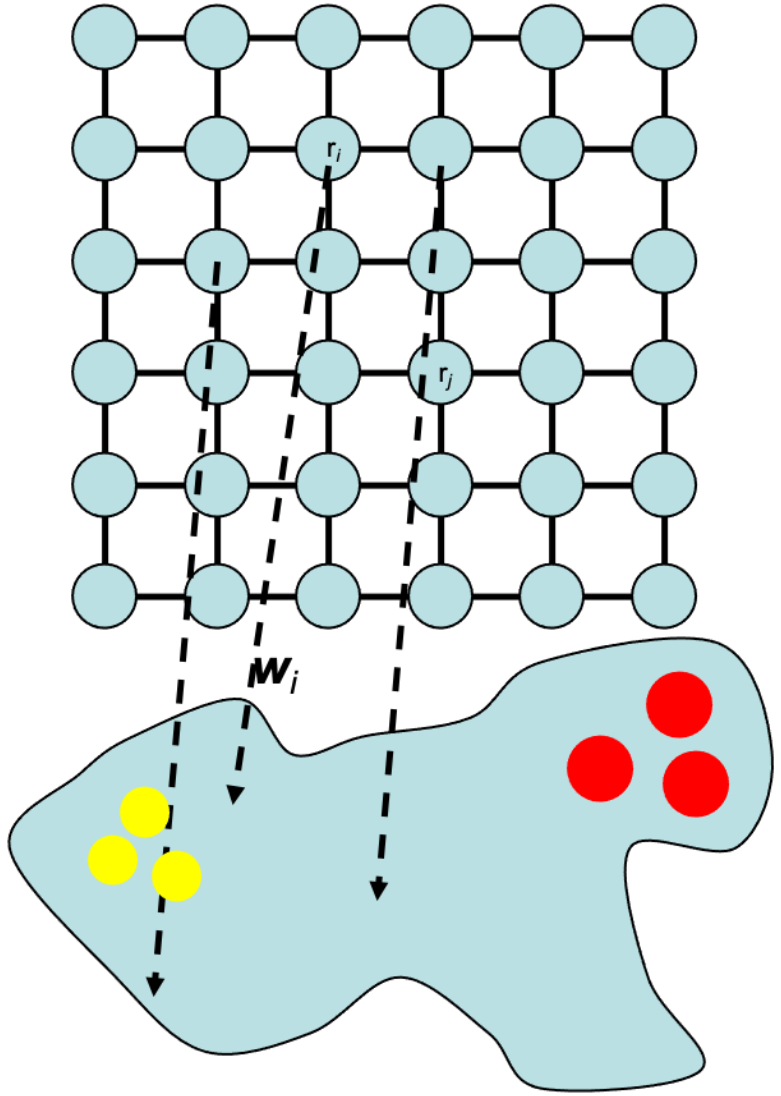
Self-Organizing Map

Lattice Variations

Self-Organizing Map

- Three Phases – Pictorial View

Initially: neuron weights point to some random location in *data space*

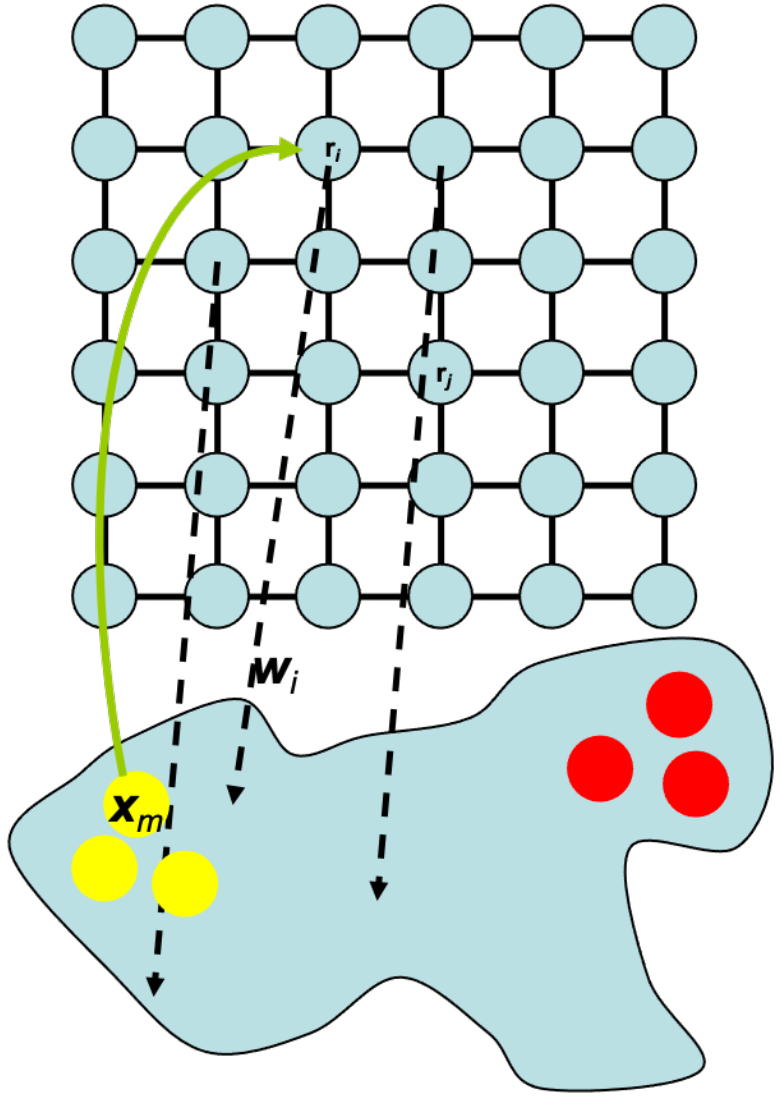


Self-Organizing Map

- Three Phases – Pictorial View

Initially: neuron weights point to some random location in *data space*

Competition: Select a pattern at random and find the winning neuron



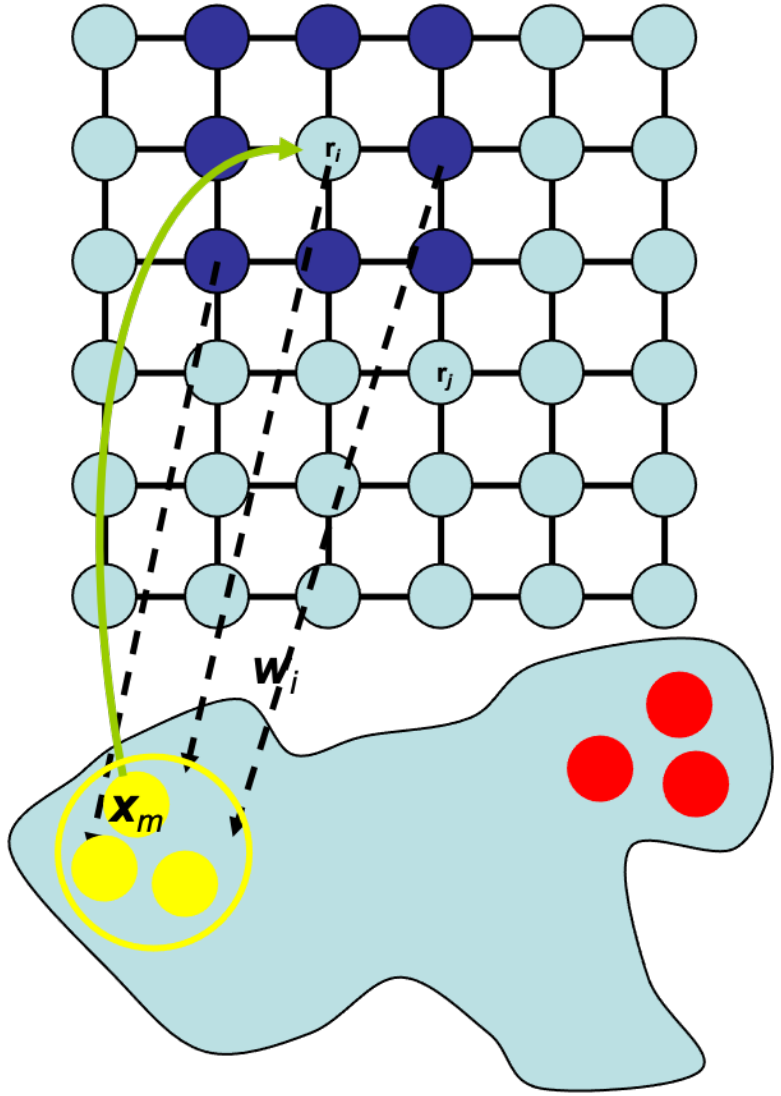
Self-Organizing Map

- Three Phases – Pictorial View

Initially: neuron weights point to some random location in *data space*

Competition: Select a pattern at random and find the winning neuron

Cooperation: Winning neuron activates neighboring neurons according the neighborhood function



Self-Organizing Map

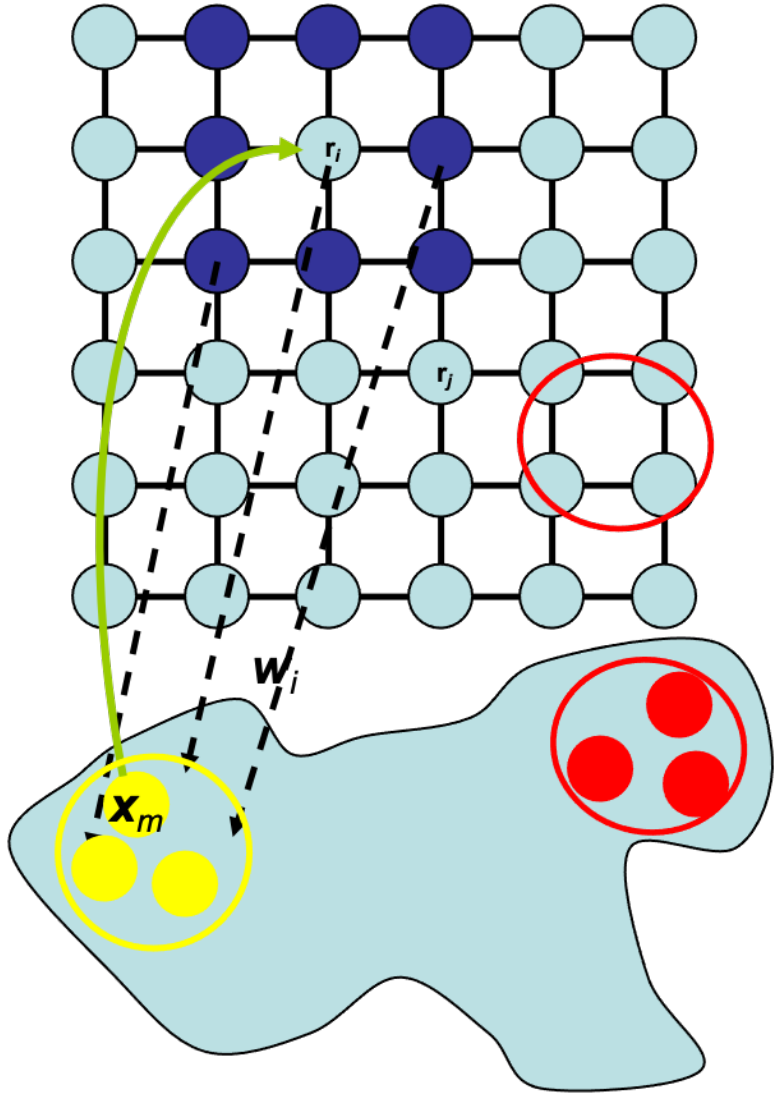
- Three Phases – Pictorial View

Initially: neuron weights point to some random location in *data space*

Competition: Select a pattern at random and find the winning neuron

Cooperation: Winning neuron activates neighboring neurons according the neighborhood function

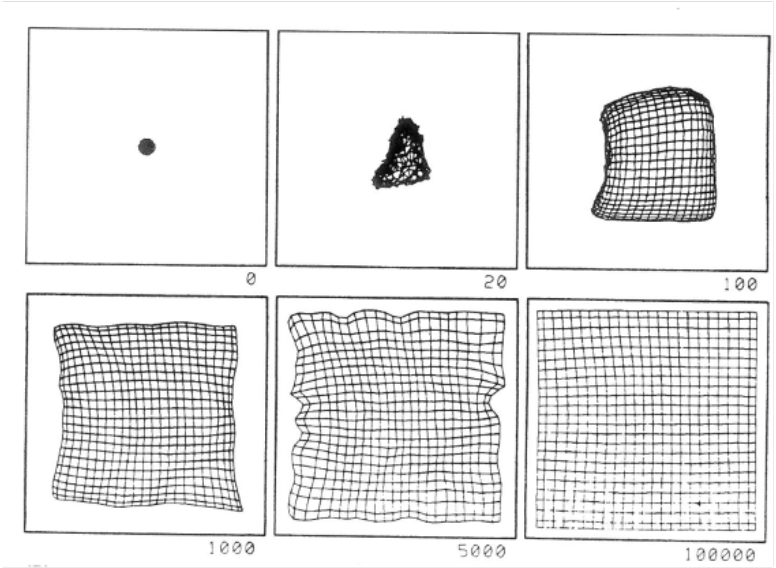
Synaptic Adaptation: Weights of the winning $neuron_i$ and each neighboring neuron are updated (in data space)



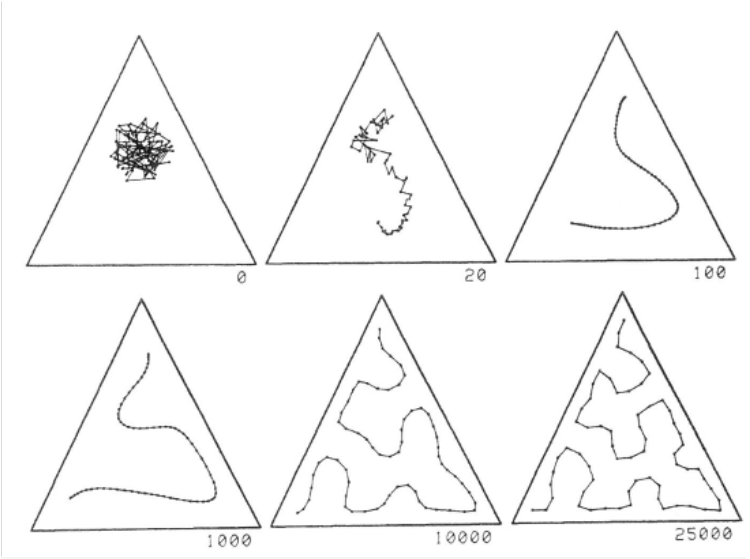
Self-Organized Learning

Self-Organized Maps

Simple Competitive Learning



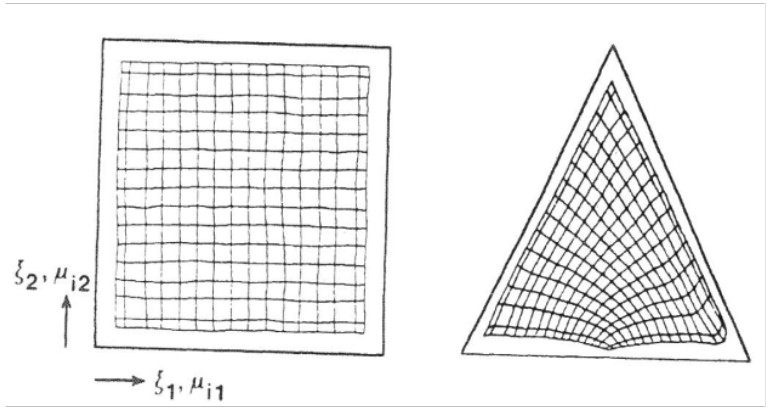
Kononen 01, Fig 3.6 Pg 114



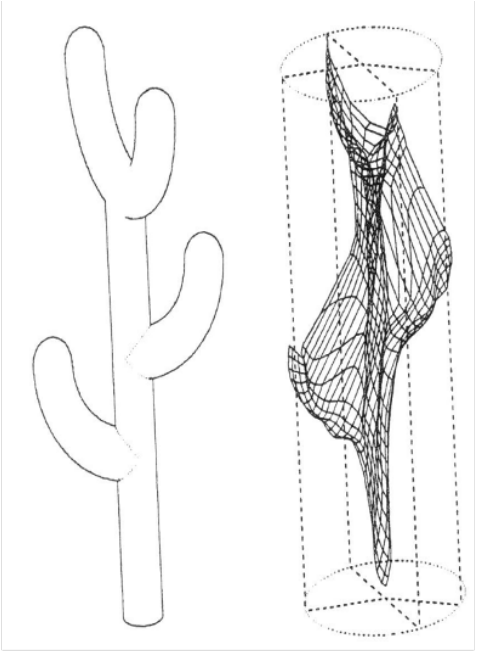
Kononen 01, Fig 3.7 Pg 115

Self-Organizing Map

Examples:



Kononen 01, Fig 3.5, Pg 113



Kononen 01, Fig 3.9, Pg 118

Self-Organizing Map

Examples:

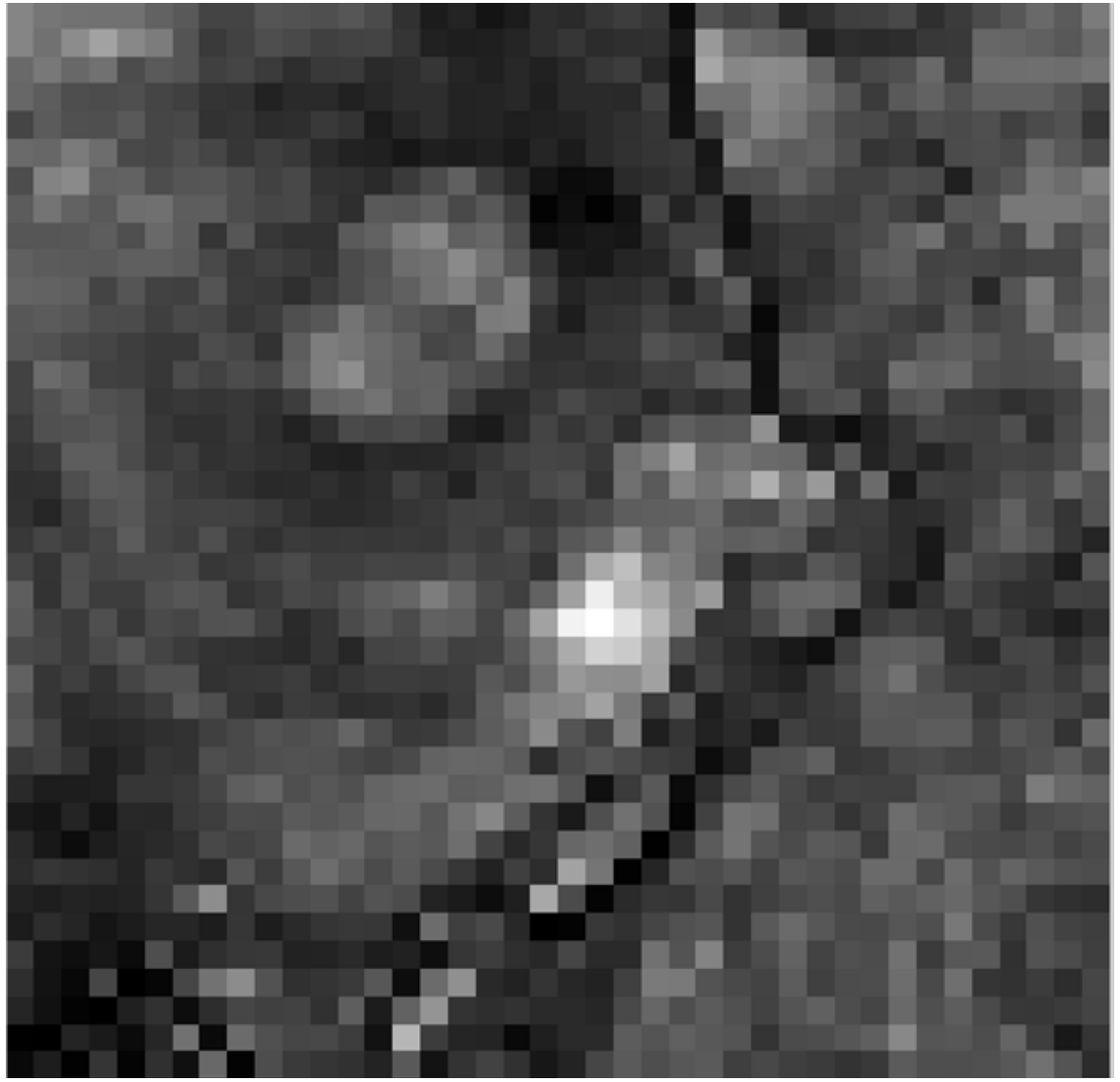
Self-Organized Learning

Self-Organized Maps

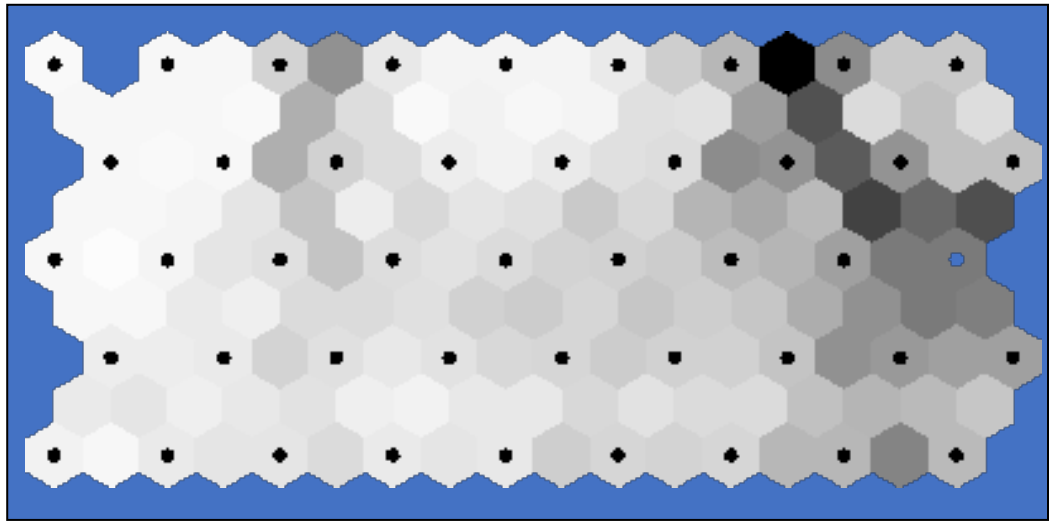
Simple Competitive Learning

Self-Organizing Map

- Density
  - One can keep a count of the number of samples assigned to each Neuron
  - This is a density representation of the data
  - Brighter Neuron’s (each square represents a neuron) are proportional to the number of samples assigned to it.

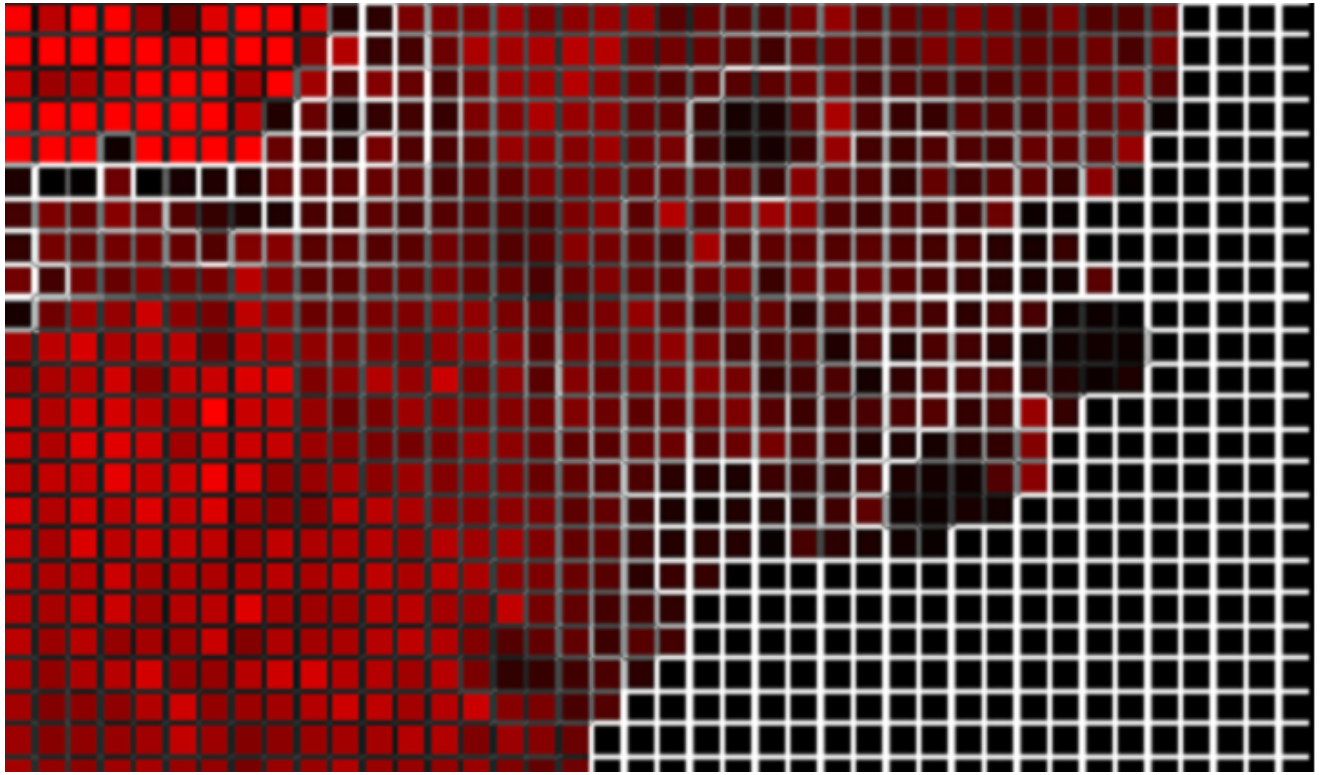


Self-Organizing Map



- **U-Matrix**
  - Its called the Unified Matrix
  - Represents the distance (Euclidean) between neighboring Neurons

Self-Organizing Map



- Alternative U-Matrix representation
  - Fence representation
  - A line between each square Neuron where the brightness is proportional to the Euclidean distance between neurons
- Dual representation
  - Density + Fence

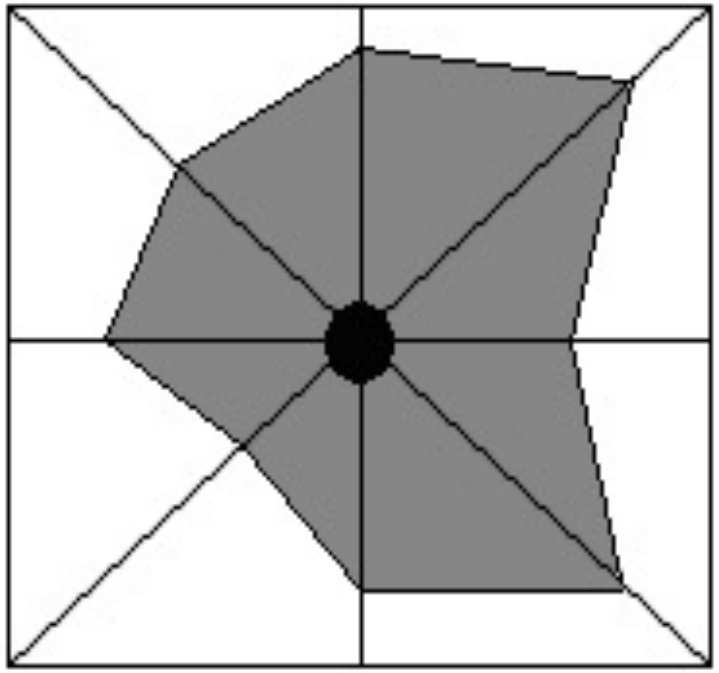
Self-Organizing Map

Visualization Schemes - Octagonal Erosion

- To visualize all 8 neighboring distances
- Add intensity for density

$$length = length_{cell-size} * (1 - \frac{\|w_i - w_{ij}\|}{d_{\max}})$$

$$d_{\max} = \max_{i,j,neighbors} \|w_i - w_j\|$$



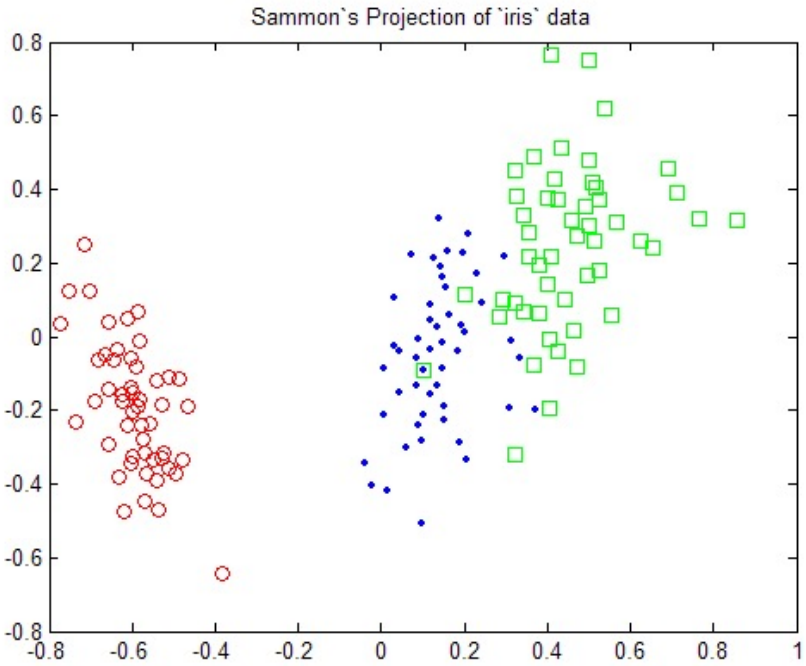
Distance visualization for one PE

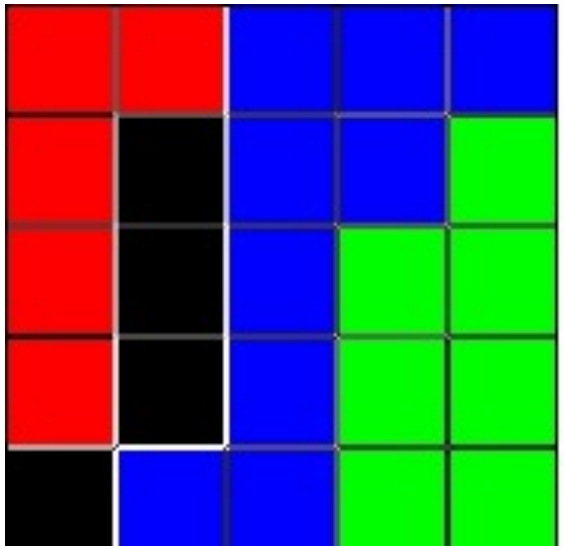
Produced by Dr. Kadim Tasdemir (Rice University, 2004)

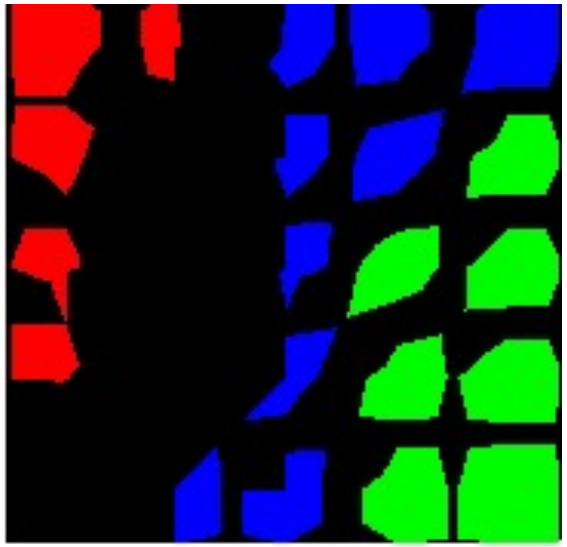
Self-Organizing Map

Example - Octagonal Erosion

- 4-D 3-class ‘iris’ data:
- 150 random samples of flowers from the iris species *setosa*, *versicolor*, and *virginica*.
- 50 observations for sepal length, sepal width, petal length, and petal width



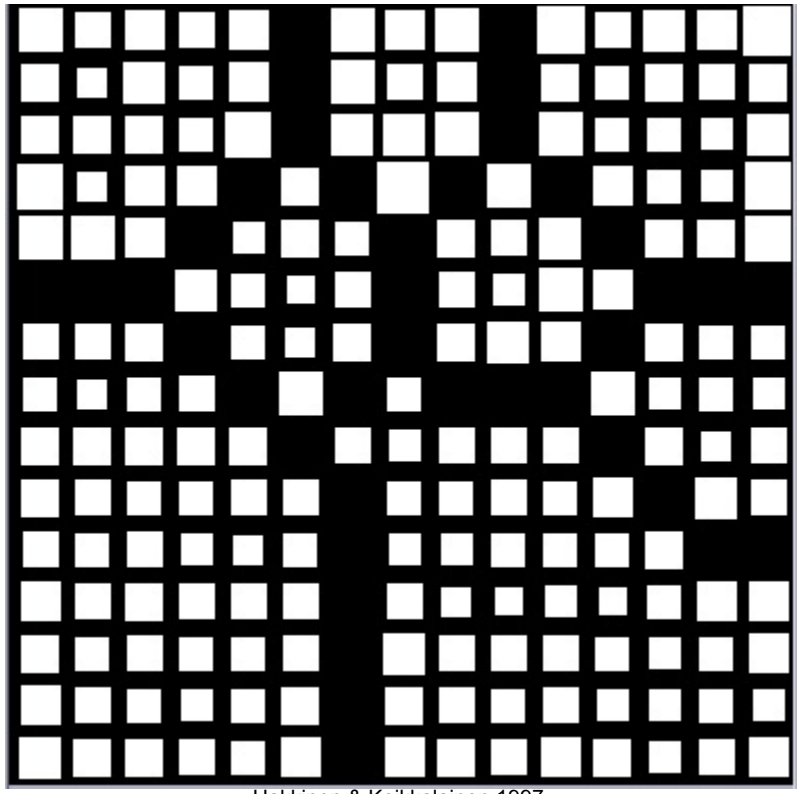




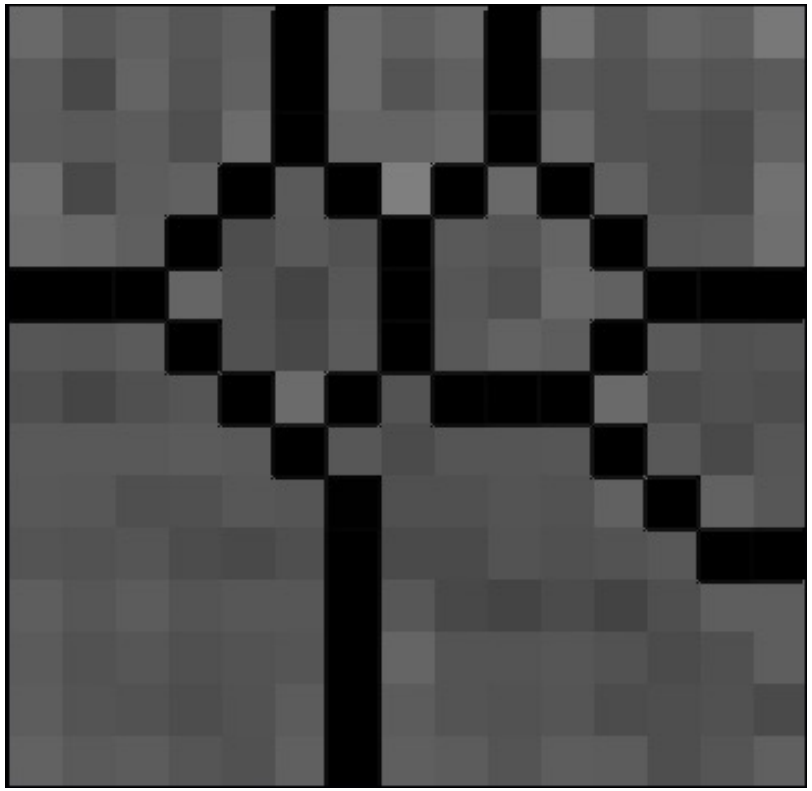
Self-Organizing Map

Visualization Schemes – Cell Size

- Cell Size (left) and Screen Intensity (right)
  - Screen intensity = density representation



Hakkinen & Koikkalainen 1997



Kraaijveld '95

SOM's in Data Exploration

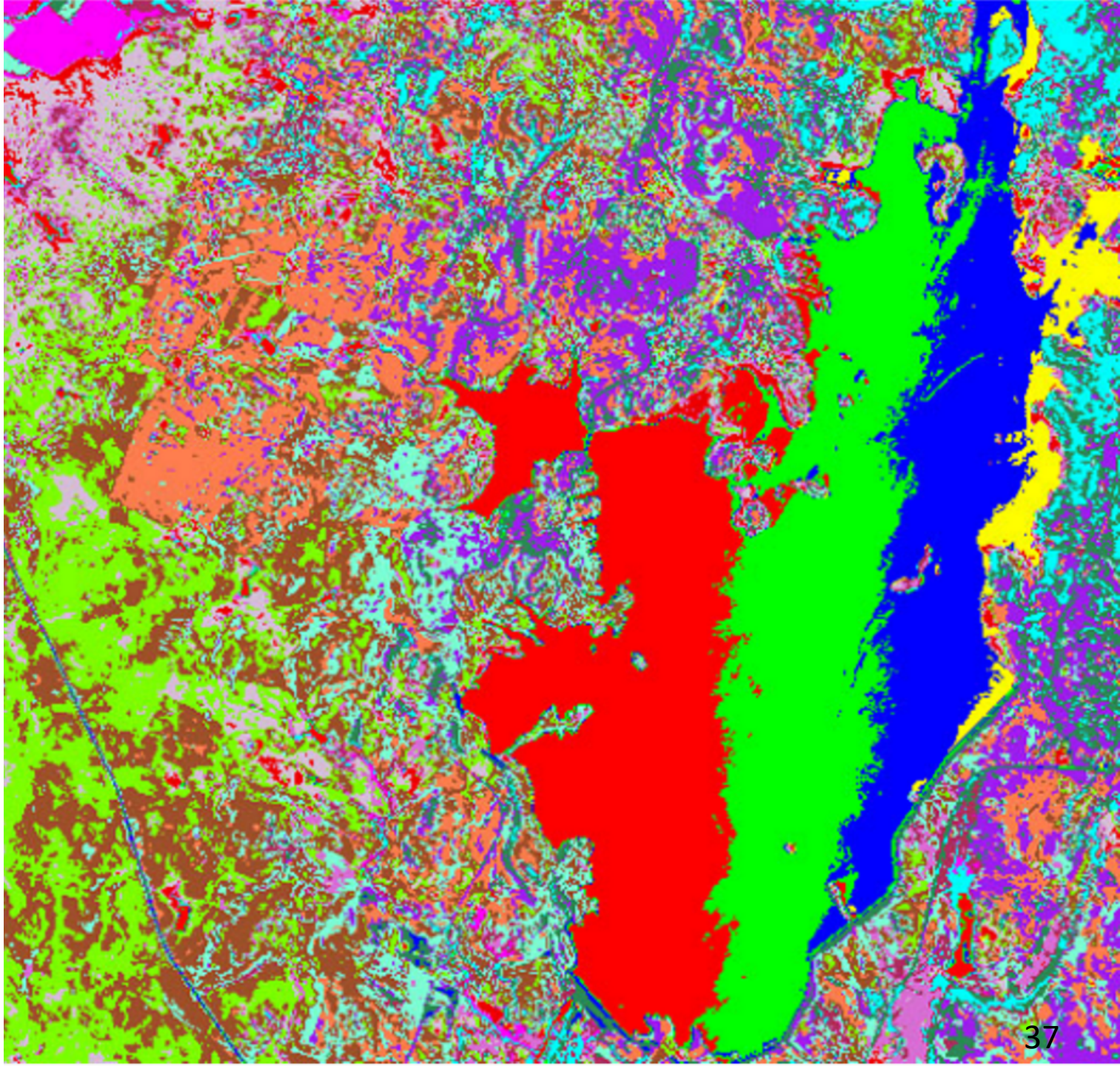
Data Clustering

- Used for a wide variety of problems
- Strengths
  - Clustering of high-dimensional data
  - Works well w/noisy & partially missing/ incomplete data

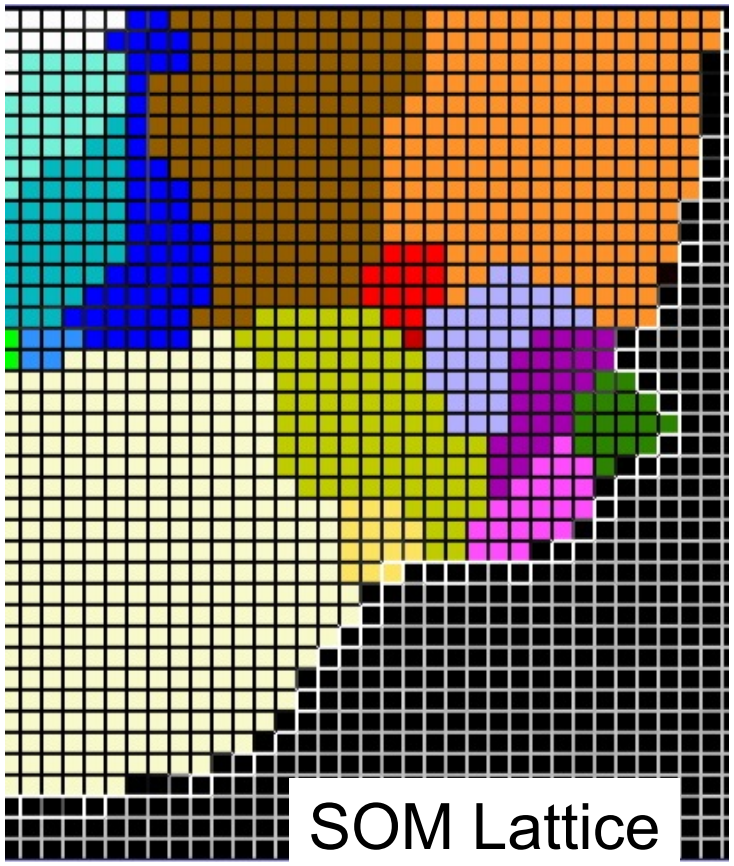
"Motherload" Image from NASA/JPL AVIRIS Imager

K-Means Clustering

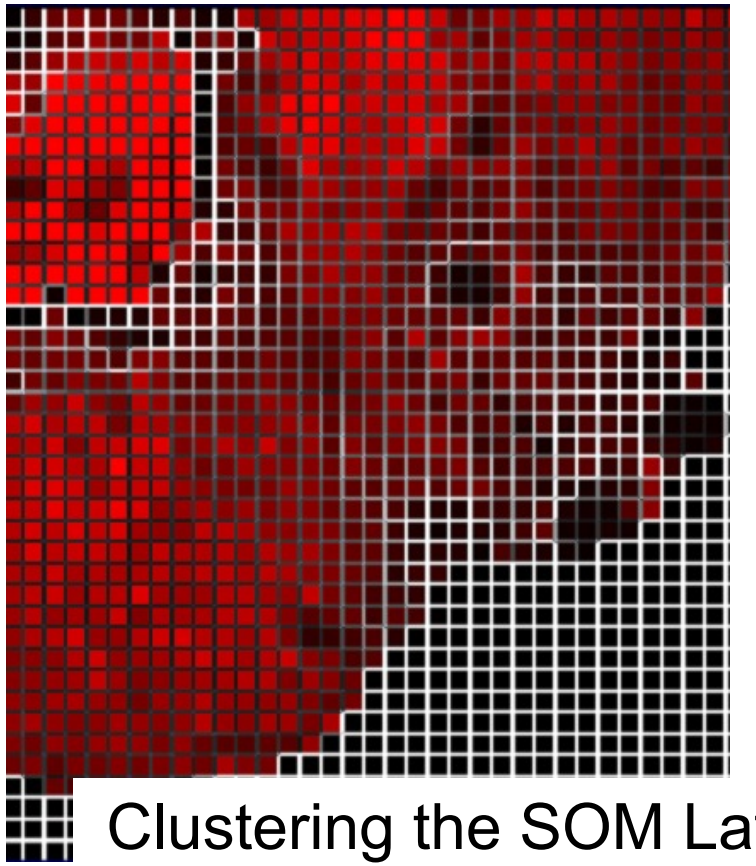
- K-means clustering using ENVI
- 16 Clusters
  10 Iterations
- User does not have any control over which clusters are drawn out



SOM Clustering



SOM Lattice

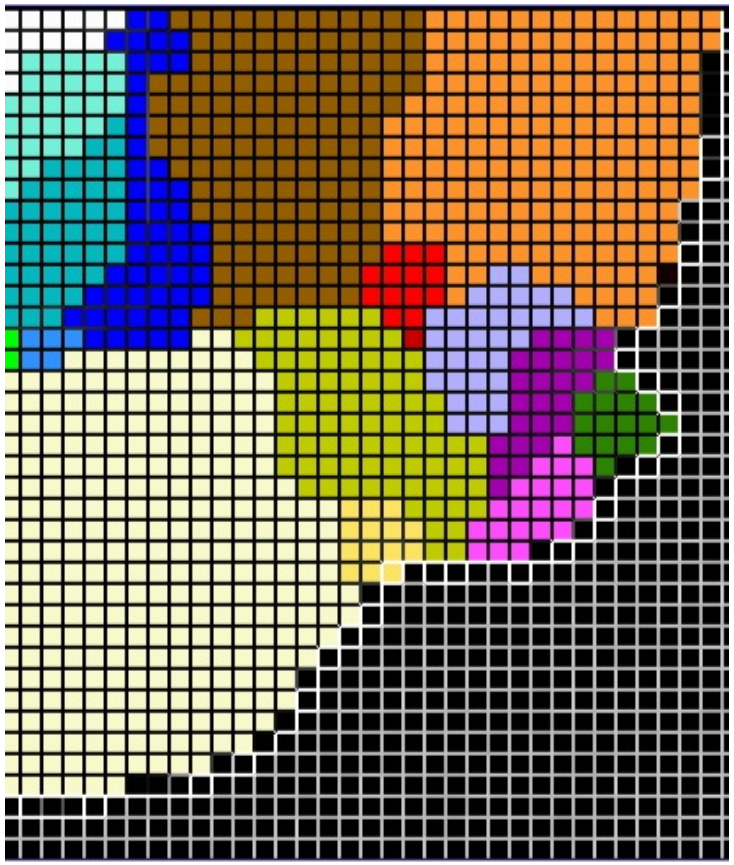


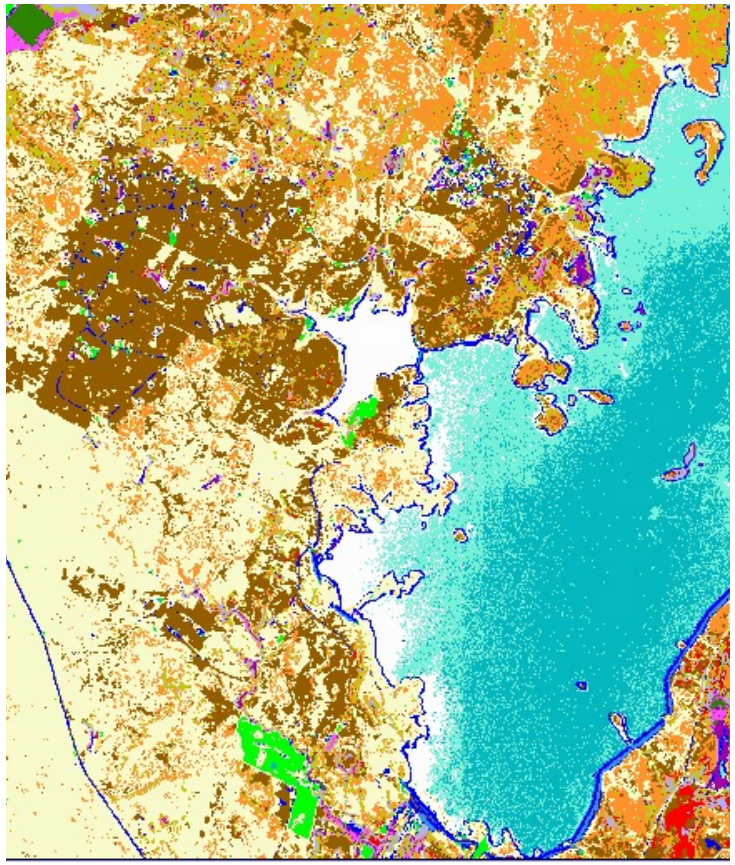
Clustering the SOM Lattice

Clustering the SOM lattice is a manual process (Current research topic)

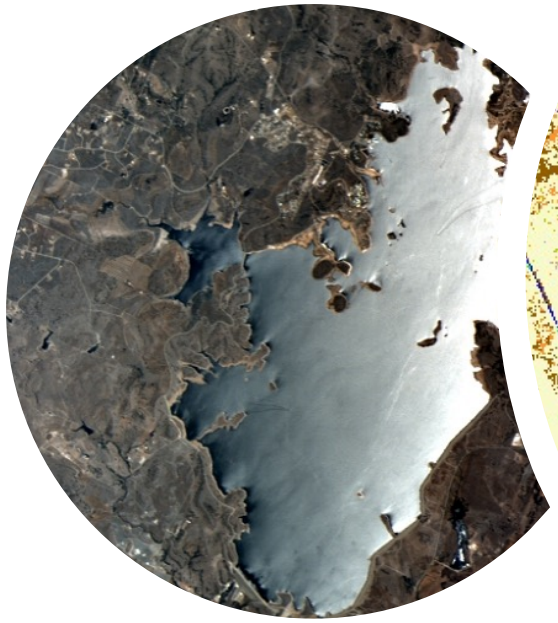
We can draw more “meaningful” clusters than, say, K-means

Clustering of the Image

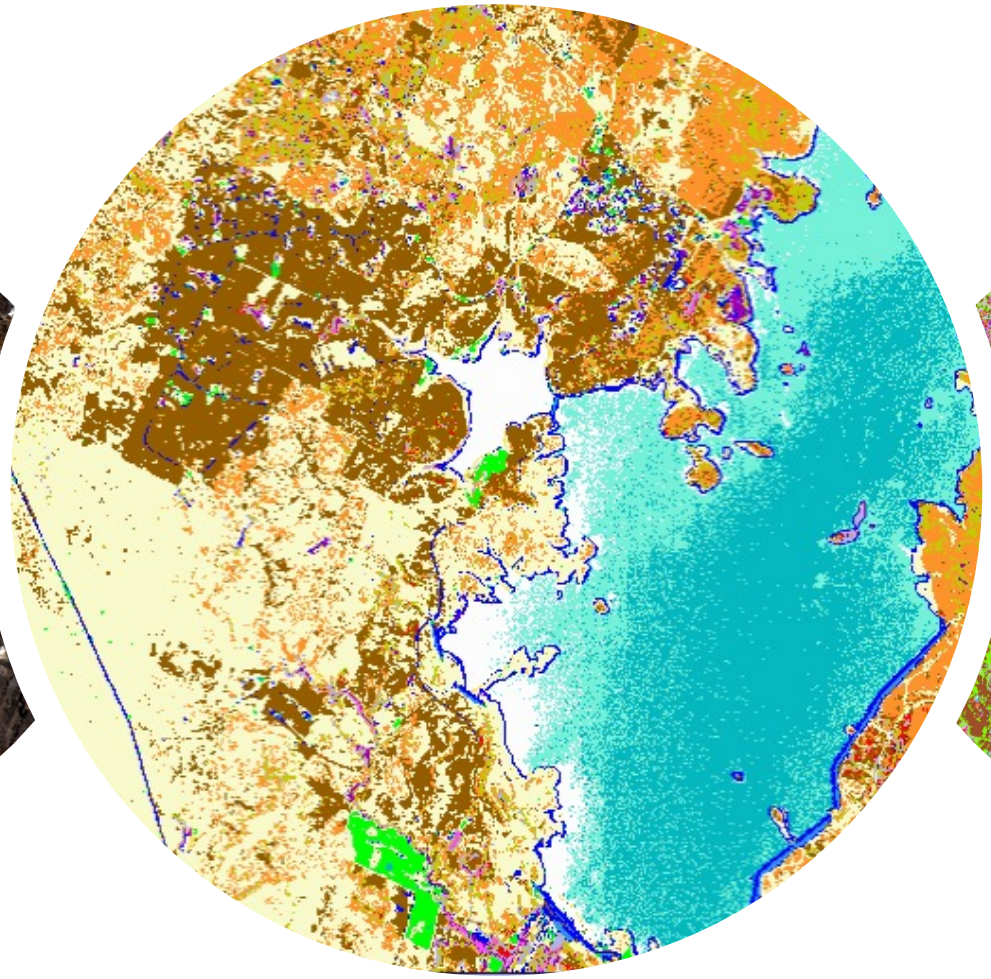




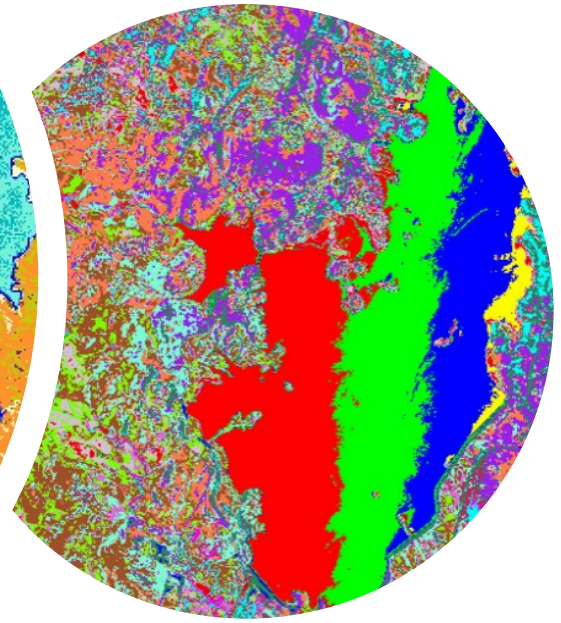
- Identifying the 16 SOM clusters in the original Image
- Clustering is more meaningful than K-means



Original image



SOM Clustered



K Means

Self-Organizing Map

The Starving Neuron Problem

- Frequency Selective Competitive Learning
  - D. DeSieno. Adding a conscience to competitive learning. In *Proceedings of the IEEE International Conference on Neural Networks* I, pages 117-124, New York, Jul 1988.
  - S.C. Ahalt, A.K. Krishnamurthy, P. Chen, and D.E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3:277-290, 1990.

The Starving Neuron

- What is a starving neuron?
  - It's a neuron that never learns
  - This is common in prototype-based paradigms
    - SOMs, LVQs, K-Means, and others
- Problem
  - Most methods are not equiprobablistic learning
    - Equiprobablistic → each prototype has the same opportunity to learn some aspect of the data
    - Equal Probable implies maximum entropy…
  - Consider it an issue of resource allocation

The Starving Neuron

Max Entropy & Equal Probability

- Given M samples, and M_i samples "assigned" to PE_i (prototype *i*, or neuron *i*)
- Equal Probable
  - $P(i) = P(j) \ \forall\ i, j \Rightarrow M_i = M_j \ \forall\ i, j$
- Entropy
  - Information-theoretic quantity that measures uncertainty

    $$H = -\sum_i P(i) \log_2 P(i)$$

  - *H* is the entropy
  - $P(i)$ is the fraction of elements assigned to $neuron_i$

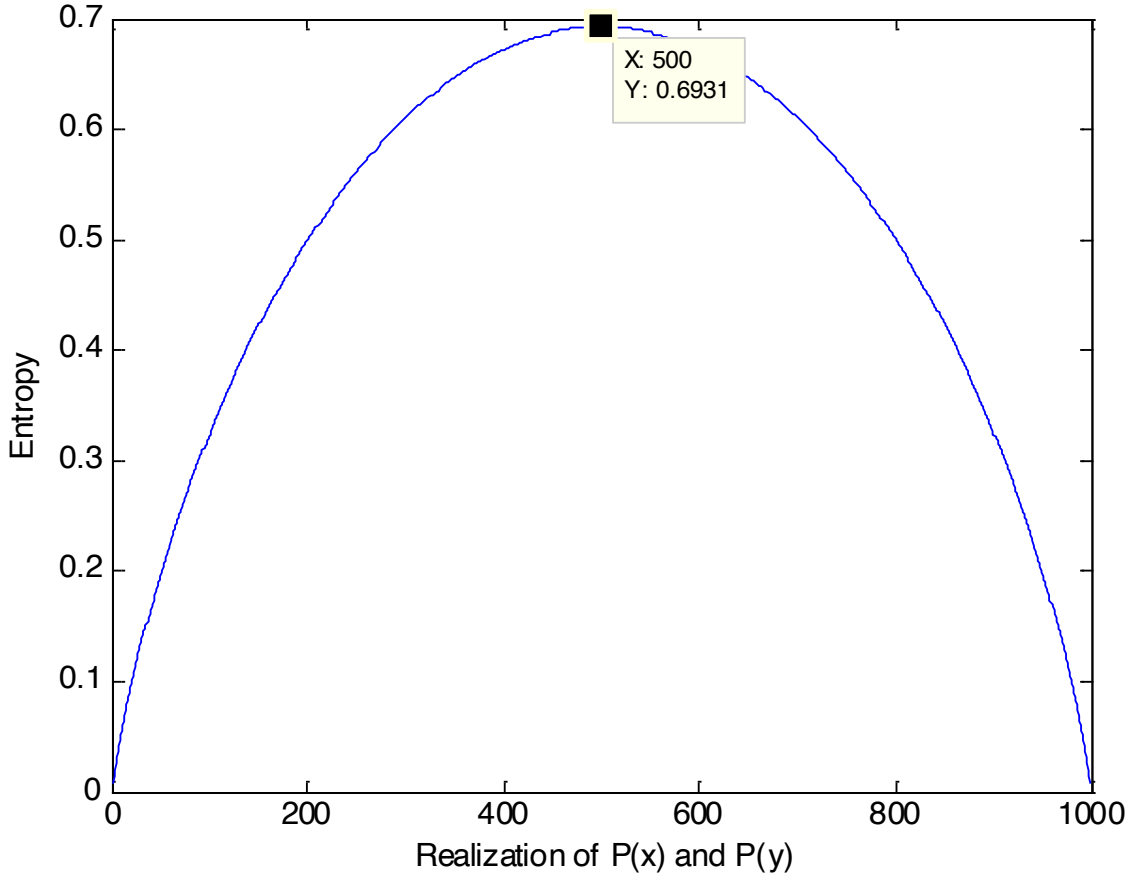
The Starving Neuron

Max Entropy & Equal Probability

- What happens to the Entropy *H* if $P(i)=P(j) \quad \forall i,j,\ i \neq j$
- Consider the following realizations for:
  - P(x)=[0.001 0.002 0.003 … 0.997 0.998 0.999]
  - P(y)=[0.999 0.998 0.997 … 0.003 0.002 0.001]
  - P(x)+P(y) = 1
- In Matlab
  - Plot (-P(x).*log(P(x)))+(-P(y).*log(P(y)))
  - What do you expect to see?

The Starving Neuron

Max Entropy & Equal Probability



Addressing the Dead Neuron

Conscience Learning Basics

- Idea
  - Encourage frequent winners to win less often
  - Encourage infrequent winners to win more often
- How?
  - Simple, just Bias the Euclidean distance!
- But…
  - Some bookkeeping is required
    - Each *neuron* has an associated winning frequency P_i
  - Slight computational burden

Addressing the Dead Neuron

Conscience Learning Basics

- In the *competition* stage
  - Winner selection was determined as:

$$i = \arg\min_k \left\| x - w_k \right\|, k = 1 \ldots K$$

- Modify the *competition* stage
  - To include a simple bias term

$$i = \arg\min_k \left(\left\| x - w_k \right\| - B_k \right), k = 1 \ldots K$$

Addressing the Dead Neuron

- The *Bias* term

$$B_k = \gamma\left(\frac{1}{K} - P_k\right)$$

- γ controls the amount of *Bias* applied to the current distance measure

- Also we notice that the *Bias* is a function of the *winning frequency* P_k

Addressing the Dead Neuron

The Winning Frequency

- *Frequency* updates occur to all neurons
- For the current winning neurons vector

$$P_k = P_k + b\left(1 - P_k\right)$$

  - Note we are *increasing* winning frequency
- Updates to the losing neurons

$$P_k = P_k + b\left(0 - P_k\right)$$

  - Note we are *decreasing* winning frequency
- b controls the amount of update to the frequencies for winning and losing

Addressing the Dead Neuron

Conscience Learning Process

- Initialize:
  - P(0) = 1/K, where K = total # of neurons in the lattice
  - Initialize neurons as described previously
- Competition

$$i = \arg\min_{k} \left(\left\| x - w_k \right\| - B_k \right), k = 1 \ldots K$$

- Cooperation – Remains unchanged
- Frequency Update

Addressing the Dead Neuron

Conscience Learning Process

- Weight Update
  - Bias is not used here, just for the *Competition*
  - The method described by DeSieno updates the winner only
  - In practice, we keep weight updates as described previously

Addressing the Dead Neuron

Conscience Learning: Example

- Example data from DeSieno 1988

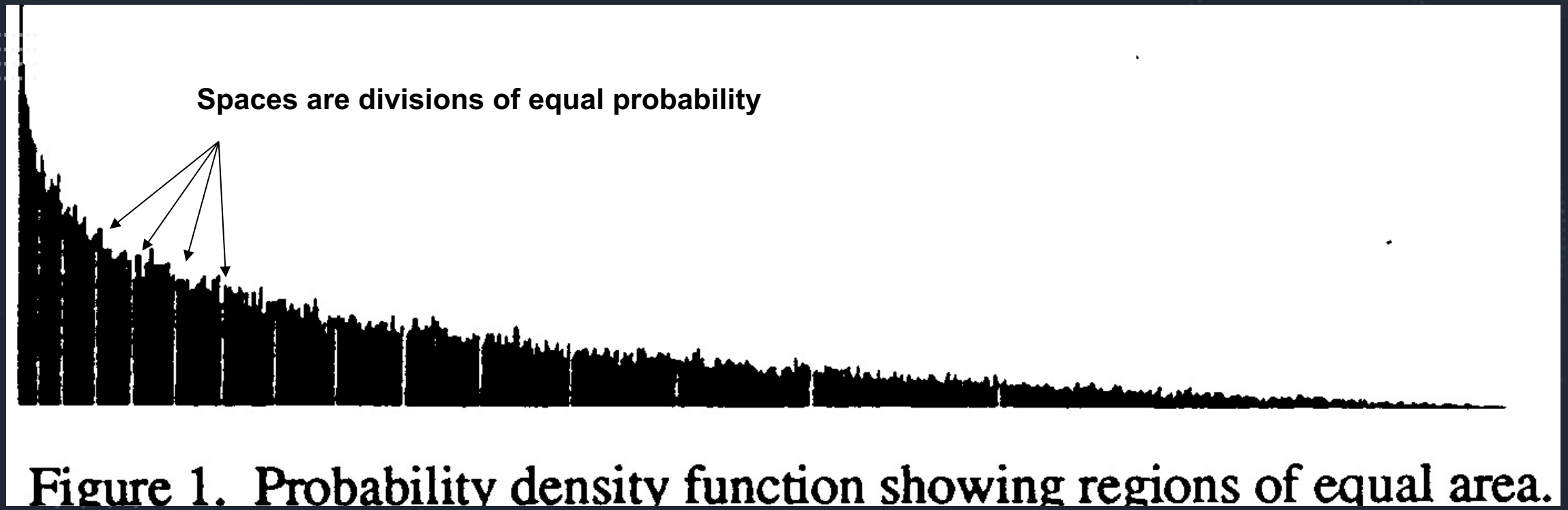


Figure 1. Probability density function showing regions of equal area.

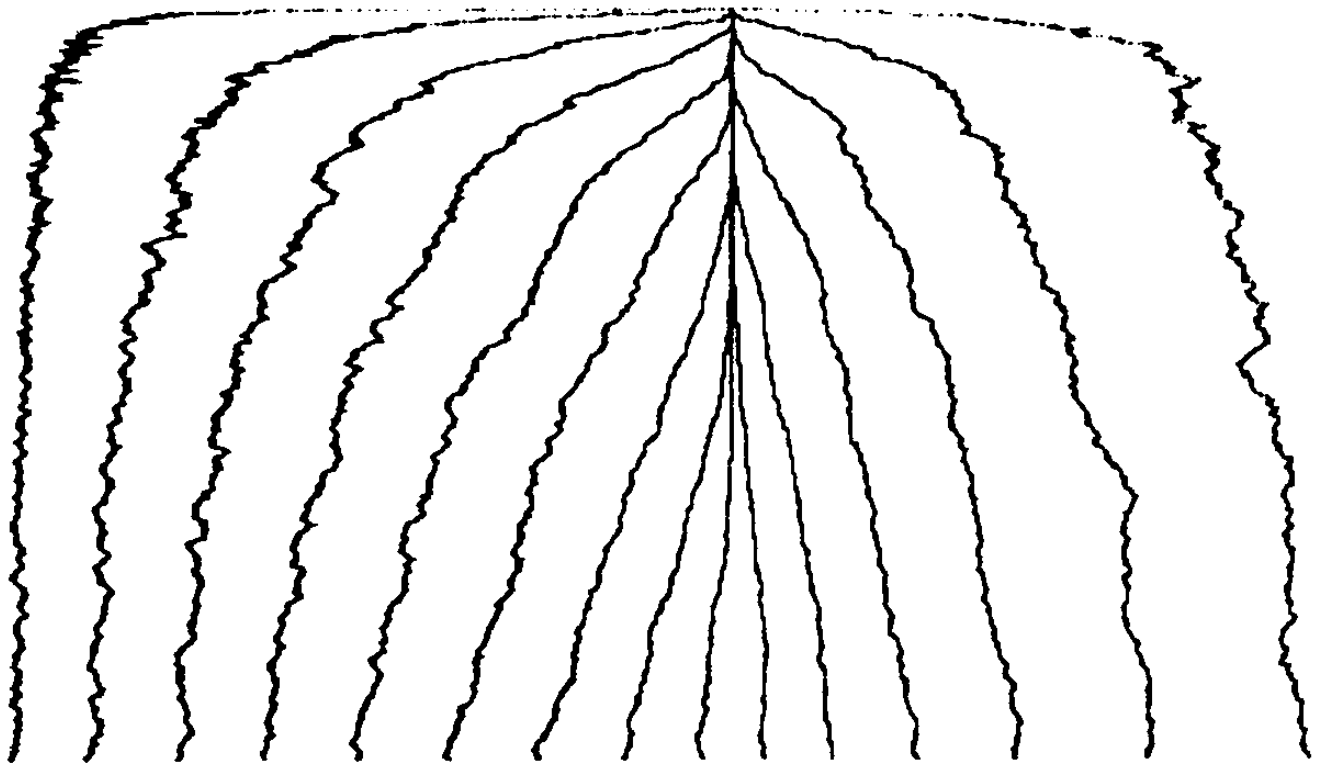


Figure 2. Kohonen learning. A = 0.03 for 32000 iterations.

Addressing the Dead Neuron

Conscience Learning Example

- Kohonen SOM (winner update only)
  - 15 neurons
    - x-axis : neuron weight Value
    - y-axis : Time
  - Should see the fingers at the same location as the white space in the previous image

Addressing the Dead Neuron

Figure 3. Conscience learning. A = 0.03 for 3200 iterations.

Conscience Learning Example

- SOM w/conscience (winner update only)

Addressing the Dead Neuron

Conscience Learning Example

- Learning rate = 0.1, 3200 Iterations…

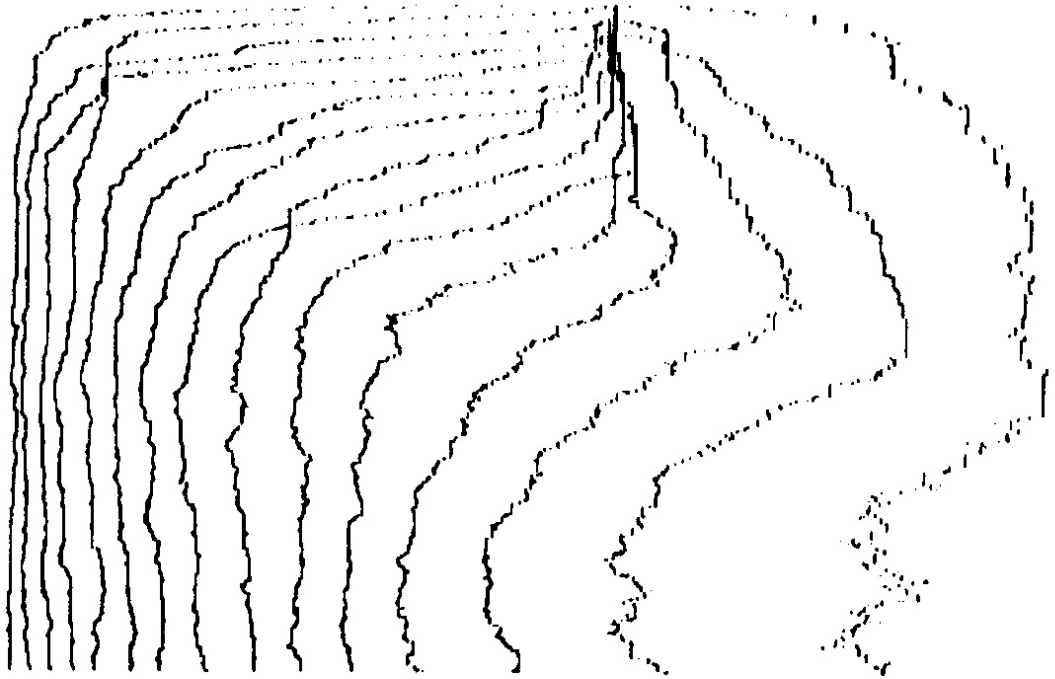


Figure 5. Conscience learning. A = 0.10 for 3200 iterations.

Kohonen SOM
Winner Update Only

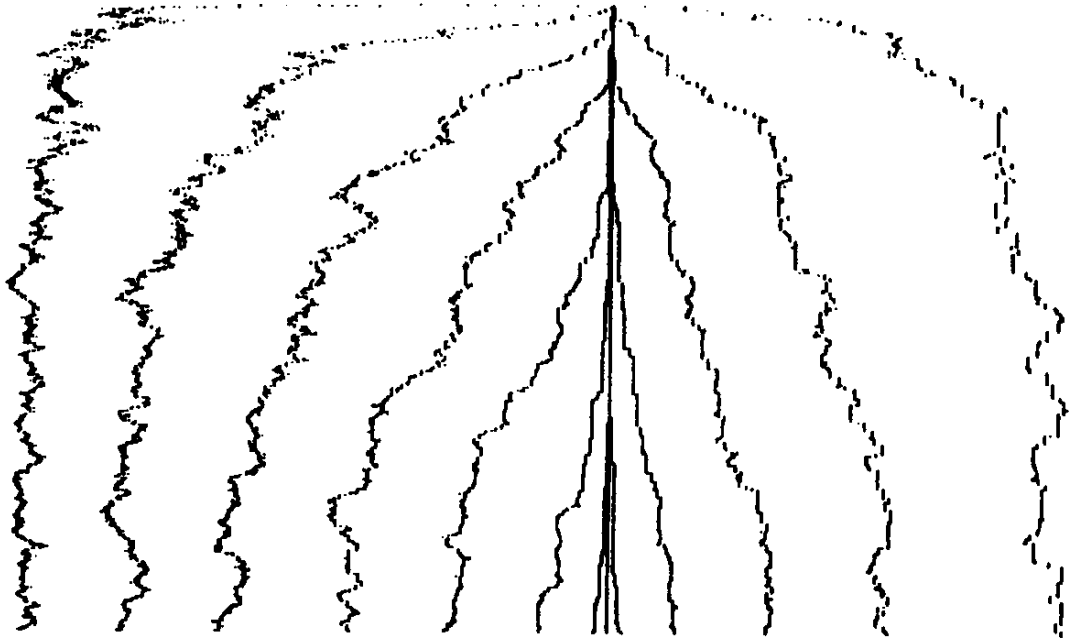


Figure 4. Kohonen learning. A = 0.10 for 3200 iterations.

Kohonen SOM
w/Conscience
Winner Update Only

Addressing the Dead Neuron

Conscience Learning Example

- Learning rate = 0.01, 9600 Iterations…

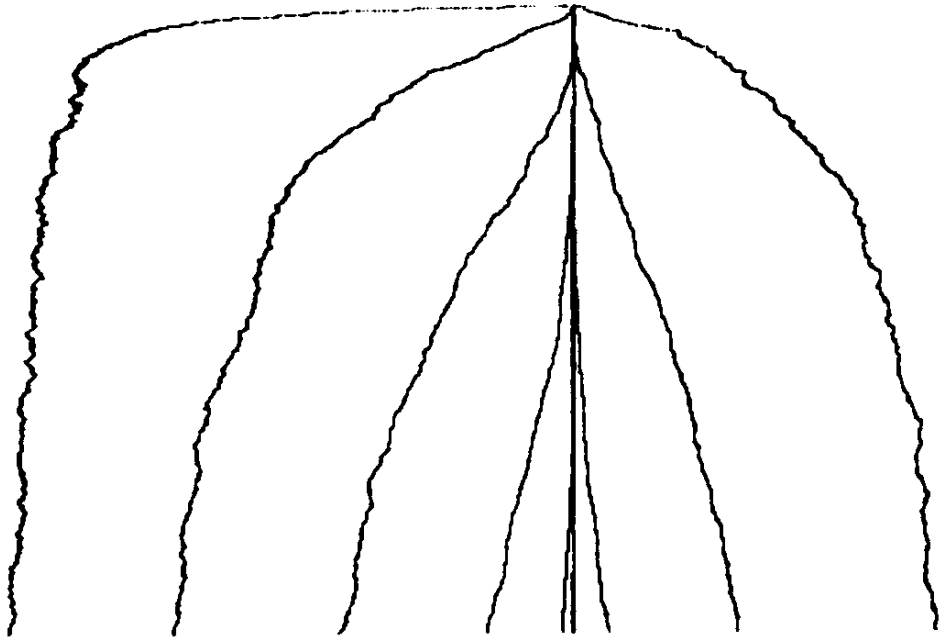


Figure 6. Kohonen learning. A = 0.01 for 9600 iterations.

Kohonen SOM
Winner Update Only

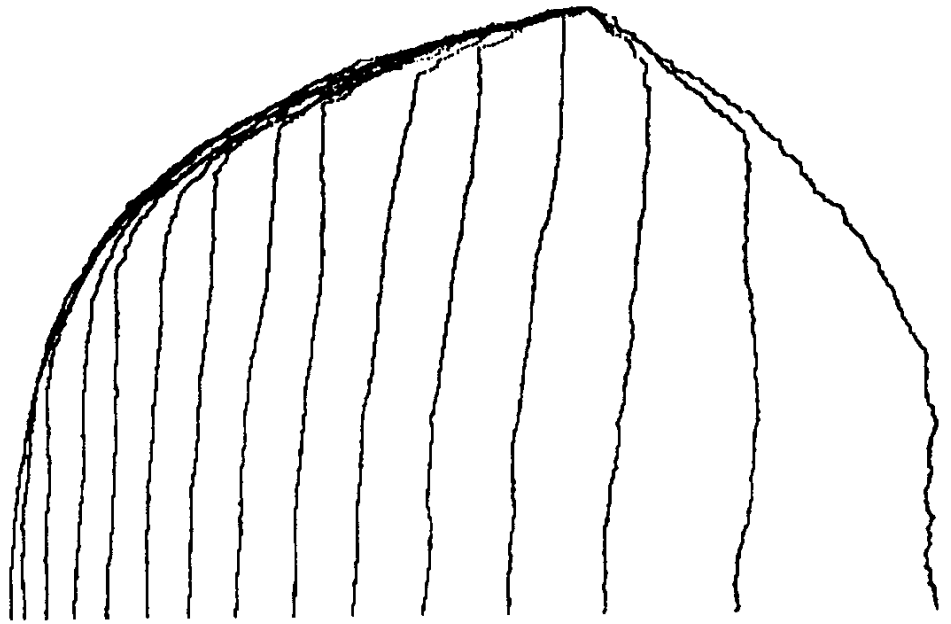


Figure 7. Conscience learning. A = 0.01 for 9600 iterations.

Kohonen SOM
w/Conscience
Winner Update Only