Data Mining
(Mining Knowledge from Data)

Self Organizing Maps (SOM)

Marcel Jiřina, Pavel Kordík

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FIT

SOM

- SOM = Self Organizing Maps,
- Prof. Teuvo Kohonen, Finland,
- TU Helsinki, 1981, since that time several thousand scientific literature references are recorded.

Competitive learning

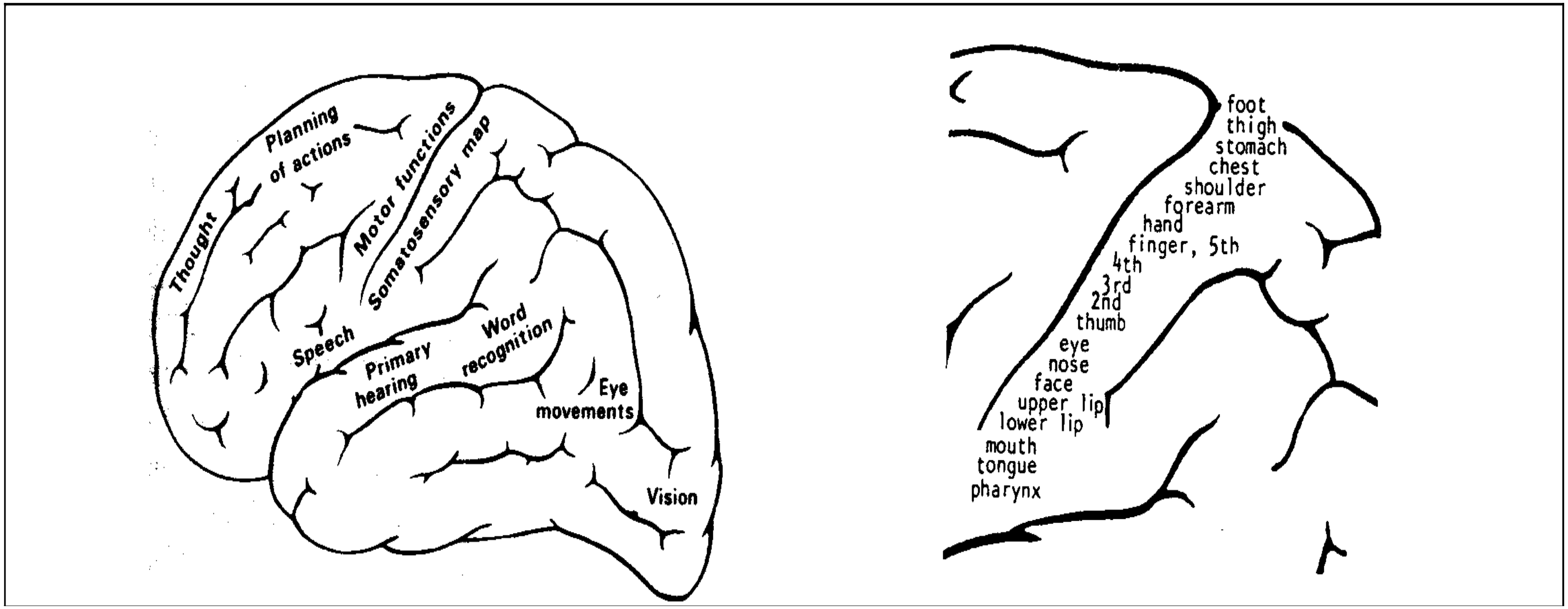
- Individuals (elements neurons) are competing
- Examples - rats and containers
  - I keep in mind where it was a good booty
  - Wins the one who comes first
  - I need to be close or someone else will overtake it
  - When I learn about a new container, and I have a chance to choose it, I need to move closer to it
  - Who will not learn it, will starve
  - Leads to a territorial settlement, reflecting the placement of containers and their usage

Competitive learning

- Inspired by nature
- I do not need any arbiter that would still say the individuals, where to go - **unsupervised learning**
- Individuals learn from examples
- The system organizes itself over time – **self-organizing**

- We will applied it to cluster analysis

SOM inspiration



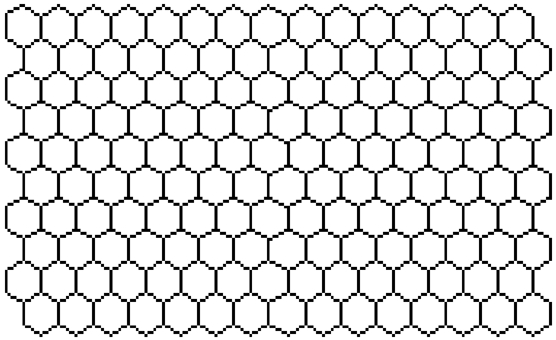
Control centers of related organs are close to each other.

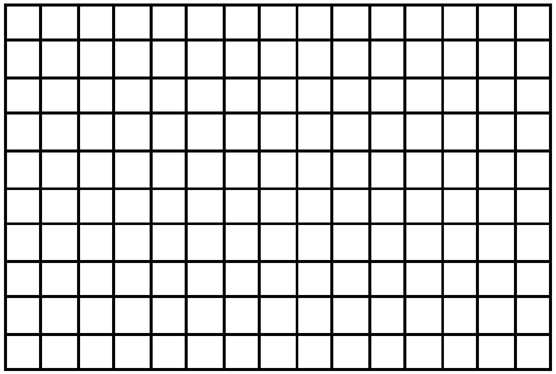
Goal

- SOM is used to approximate the complex multi-dimensional data using a small number of representatives
  - The representatives must be displayed in 2D or 3D space
  - Transformation of n-dimensional data into 2D or 3D space of representatives
  - Instances that are close to each other in the original space should be close to each other in the new space as well

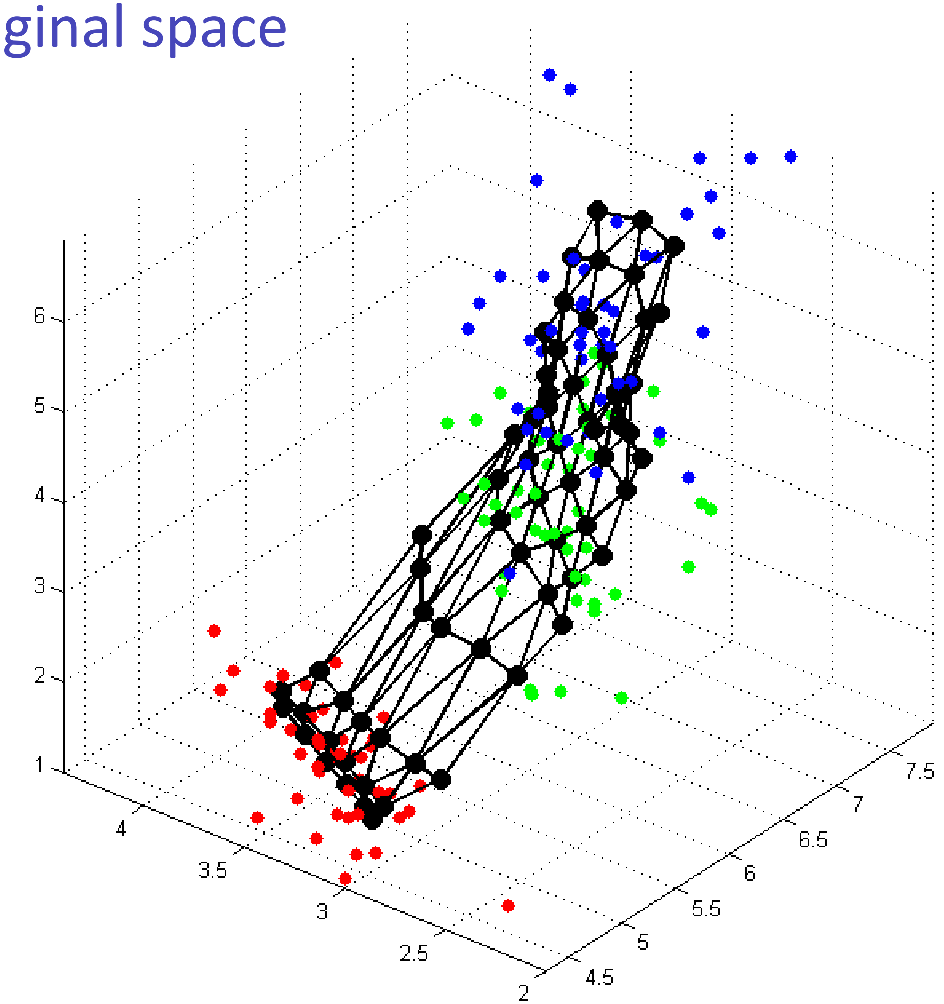
Space of the representatives

- Mostly 2D (sometimes 1D or 3D) rectangular lattice
  - Can be squared but is most hexagonal





- 2D network is adjusted so to cover data in the multi-dimensional space
  - Neighbors in the lattice of representatives are close to each other as in the original space



Learning the SOM

- Each representative has its coordinates in the original space (= space of weights) and the lattice
- Learning the SOM = setting the coordinates of representatives in the multidimensional space so that they are as close as possible to the training data
  - The neighbors in the lattice has to stay close to each other as in the original space

Learning the SOM

- Iterative algorithm
  1. Randomly initialize the weights
  2. One randomly selected instance data is always introduced to the network
  3. Find the BMU – the representative that is closest to the instance
  4. Update weights so that the BMU is moved toward the instance
     - Move the neighbors of the BMU towards the instance as well (not so significantly)
  5. Repeat step 2 until the criterion for stopping is fulfilled

- The formula for adjusting the weights

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\underbrace{[x_i(t) - w_{ij}(t)]}$$

The neighborhood function, it depends on the distance to the BMU

Vector from the current vector of weights to the instance

Example

$$\mathbf{X} = \begin{bmatrix} 0.52 \\ 0.12 \end{bmatrix}$$

$$\mathbf{W}_1 = \begin{bmatrix} 0.27 \\ 0.81 \end{bmatrix} \qquad \mathbf{W}_2 = \begin{bmatrix} 0.42 \\ 0.70 \end{bmatrix} \qquad \mathbf{W}_3 = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix}$$

$$d_1 = \sqrt{(x_1 - w_{11})^2 + (x_2 - w_{21})^2} = \sqrt{(0.52 - 0.27)^2 + (0.12 - 0.81)^2} = 0.73$$

$$d_2 = \sqrt{(x_1 - w_{12})^2 + (x_2 - w_{22})^2} = \sqrt{(0.52 - 0.42)^2 + (0.12 - 0.70)^2} = 0.59$$

$$d_3 = \sqrt{(x_1 - w_{13})^2 + (x_2 - w_{23})^2} = \sqrt{(0.52 - 0.43)^2 + (0.12 - 0.21)^2} = 0.13$$

The third node is the winner – it is the closest node to the instance X

Example ...

Move it closer to the instance

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left[x_i(t) - w_{ij}(t)\right]$$

$$\Delta w_{13} = \eta(\mathrm{t})(x_1 - w_{13}) = 0.1(0.52 - 0.43) = 0.01$$

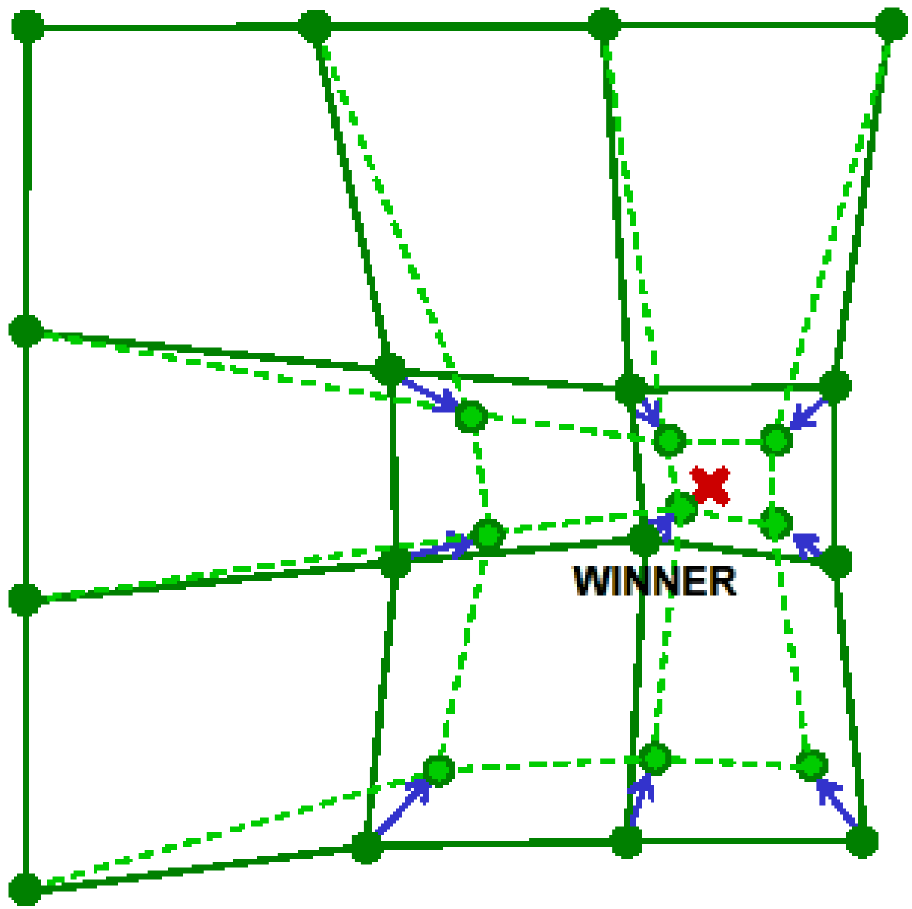
$$\Delta w_{23} = \eta(\mathrm{t})(x_2 - w_{23}) = 0.1(0.12 - 0.21) = -0.01$$

$$\mathbf{W}_3(p+1) = \mathbf{W}_3(p) + \Delta\mathbf{W}_3(p) = \begin{bmatrix} 0.43 \\ 0.21 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.01 \end{bmatrix} = \begin{bmatrix} 0.44 \\ 0.20 \end{bmatrix}$$

Only weights of the winning BMU are adjusted

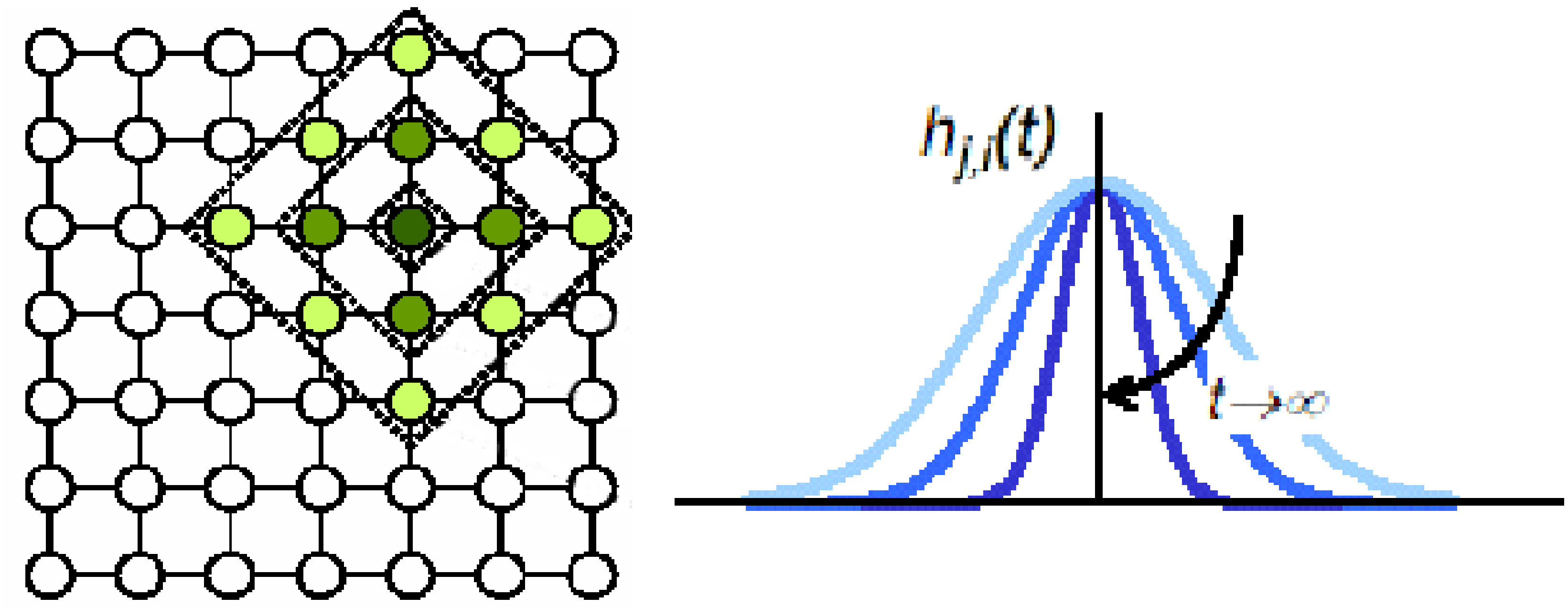
Winner takes all

Updating BMU and the surrounding representatives



Slide by Johan Everts

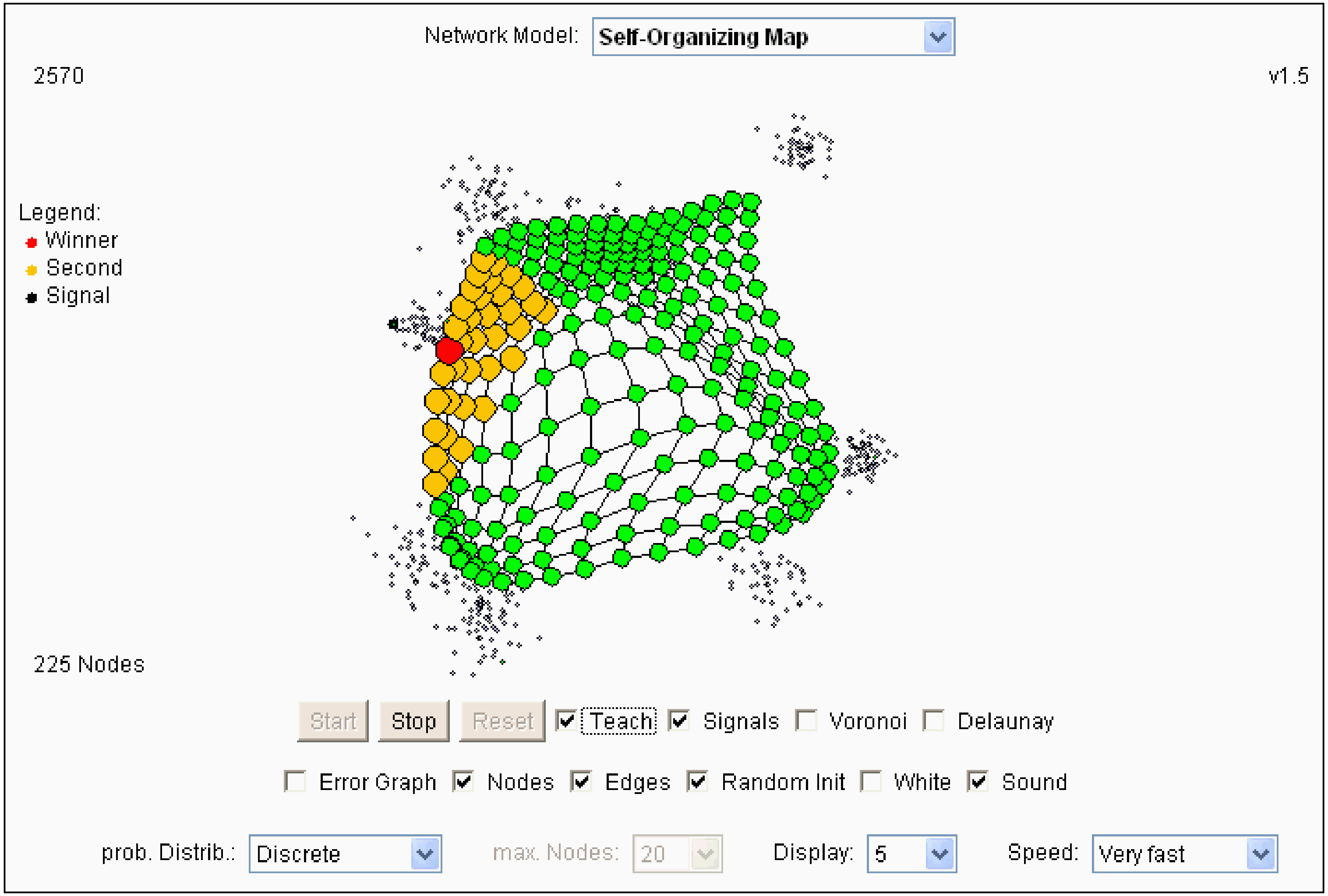
Example of the neighborhood function: Bell curve



- Representatives close to BMU (in the lattice) are moved more than distant BMUs (they are moved little or not all all)

Slide by Johan Everts

Applet



- http://www.sund.de/netze/applets/gng/full/GNG.html

Visualization: classic SOM

- The problem is how to view locations of neurons (representatives)
- Dimension of weights = dimension of the input vector

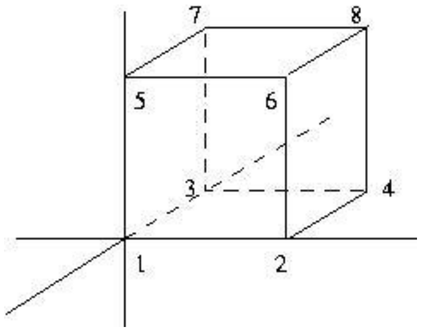
- I need to display it in 2D. How?
  - U-matrix
  - Principal Component Analysis (PCA)
  - Sammon's nonlinear projection

U-matrix (Unified distance)

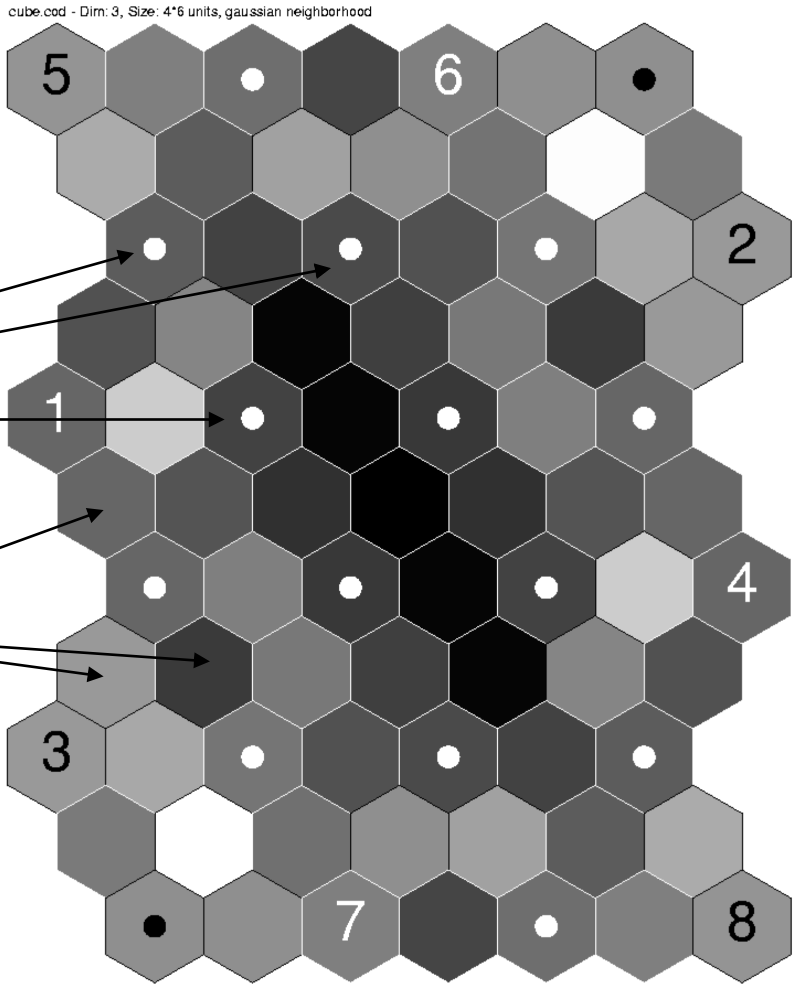
- Distance matrix among weighting vectors of individual neurons, typically it is visualized, distance is expressed by color - light color = small distance.
- Shows the structure of distances in the data space.
- Location of BMU reflects the topology of the data.
- The color of a neuron is a distance of its weight vector from all other weight vectors (neurons)
- Dark weight vectors are faraway from other data vectors in the input space.
- Bright weight vectors are surrounded by weight vectors of close neurons in the input space.
- The hills separate the clusters (valleys).

Example of U-matrix

- Data:



- Neurons
- Distances among neighboring neurons



P-matrix (Pareto density estimation)

- Shows the number of data vectors from input space belonging to a sphere around its weight vector (with radius set by the Pareto rule).
- Reflects data density.
- Neurons with high values are placed in dense regions of the input space.
- Neurons with low values are "lonesome" in the input space.
- "Valleys" separate clusters ("plateau").
- Completes the information obtained from the U-matrix.

U*-Matrix

- The combination of U-Matrix and P-Matrix
- It is U-matrix corrected by values in the P-matrix.
- The distance between neighboring neurons (neurons *a* and *b* in the lattice) are computed from the U-matrix and are weighted by the density of vectors around neuron *a*.

Disadvantages of U-Matrix, P-Matrix, …

- Shows only distances among neighbors
- After re-training of the network on the same data the matrices can be different (e.g. they can be rotated about 90 degrees)
- They are not intuitively interpretable, if you do not know what is exactly coded by the colors.

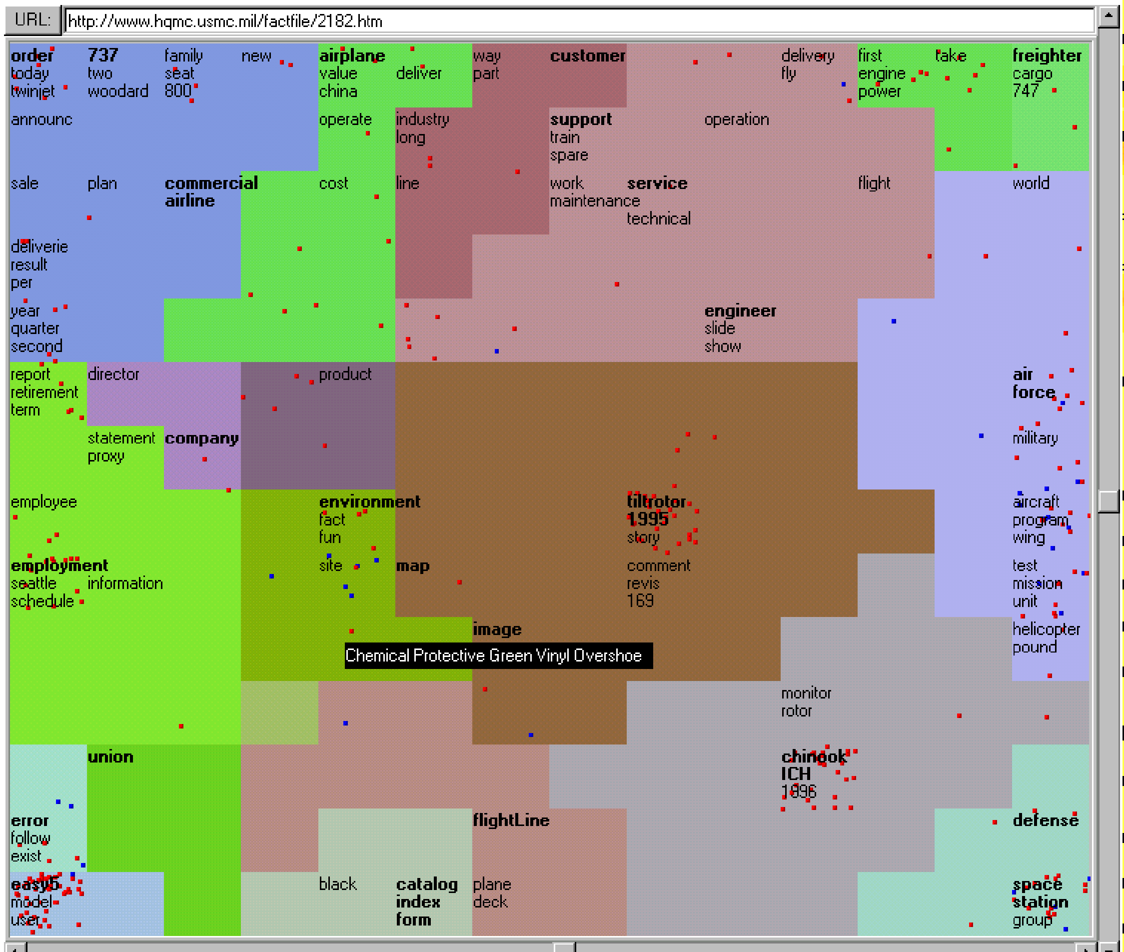
- But how to view n-dimensional data in 2D, to maintain the original distance if possible?

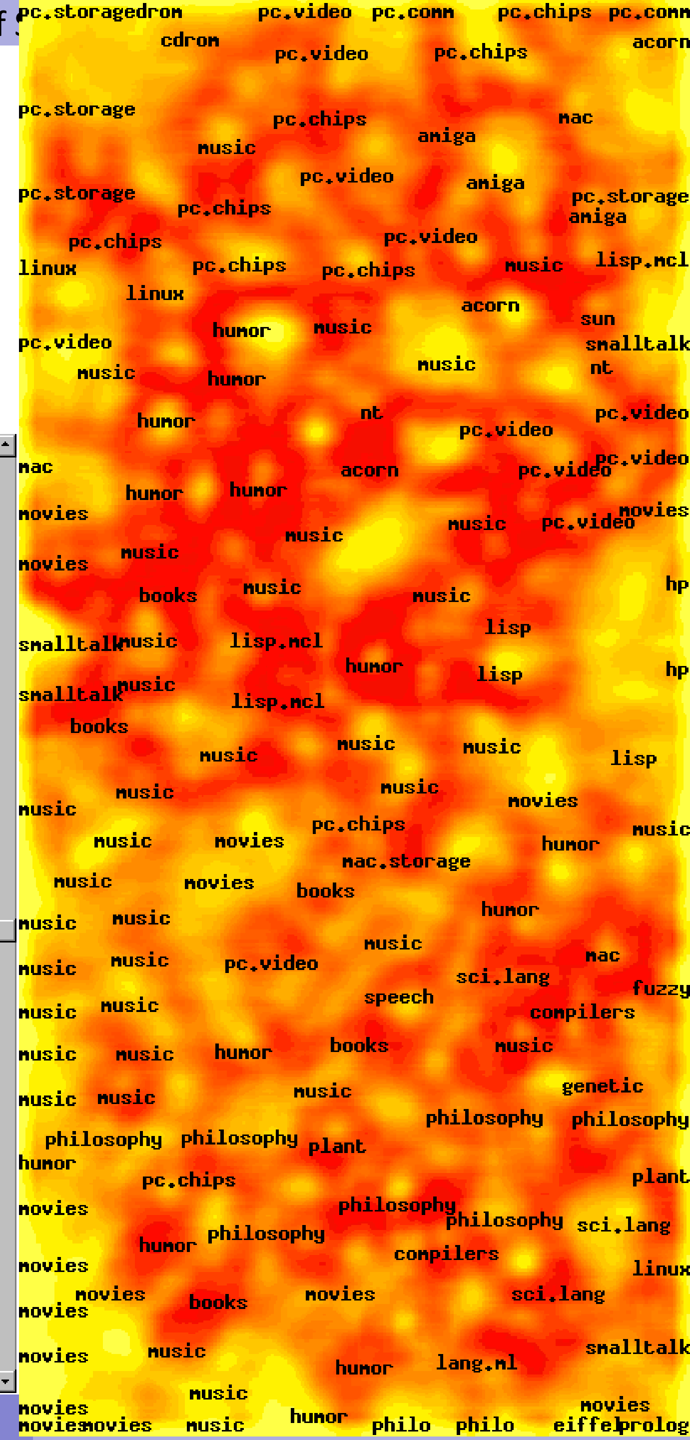
Applications of SOM

- http://www.generation5.org/content/2007/kohonenImage.asp

Websom

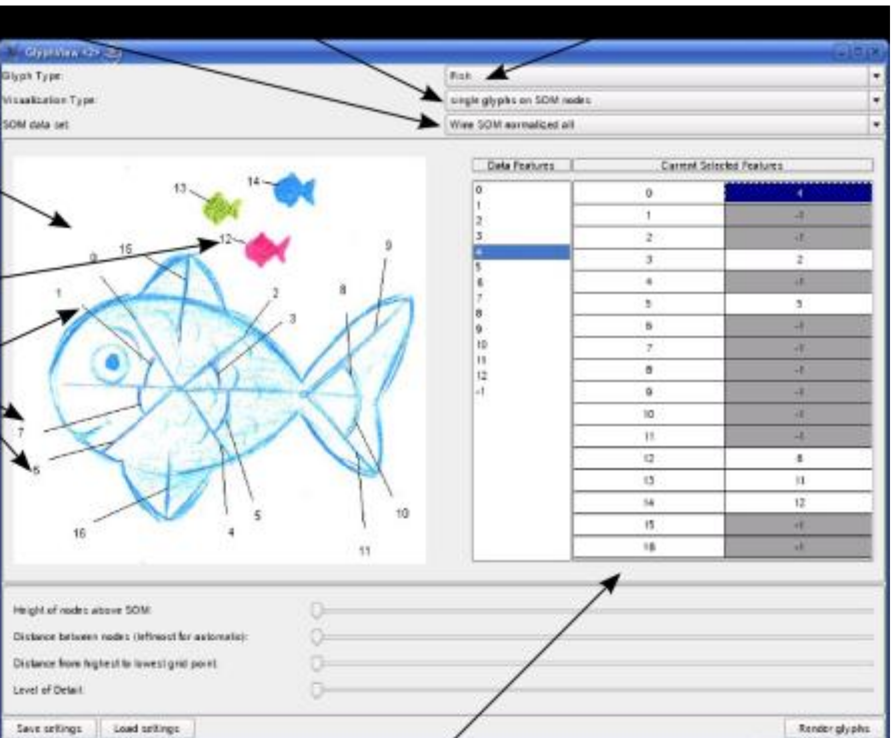
- Similarity of webages

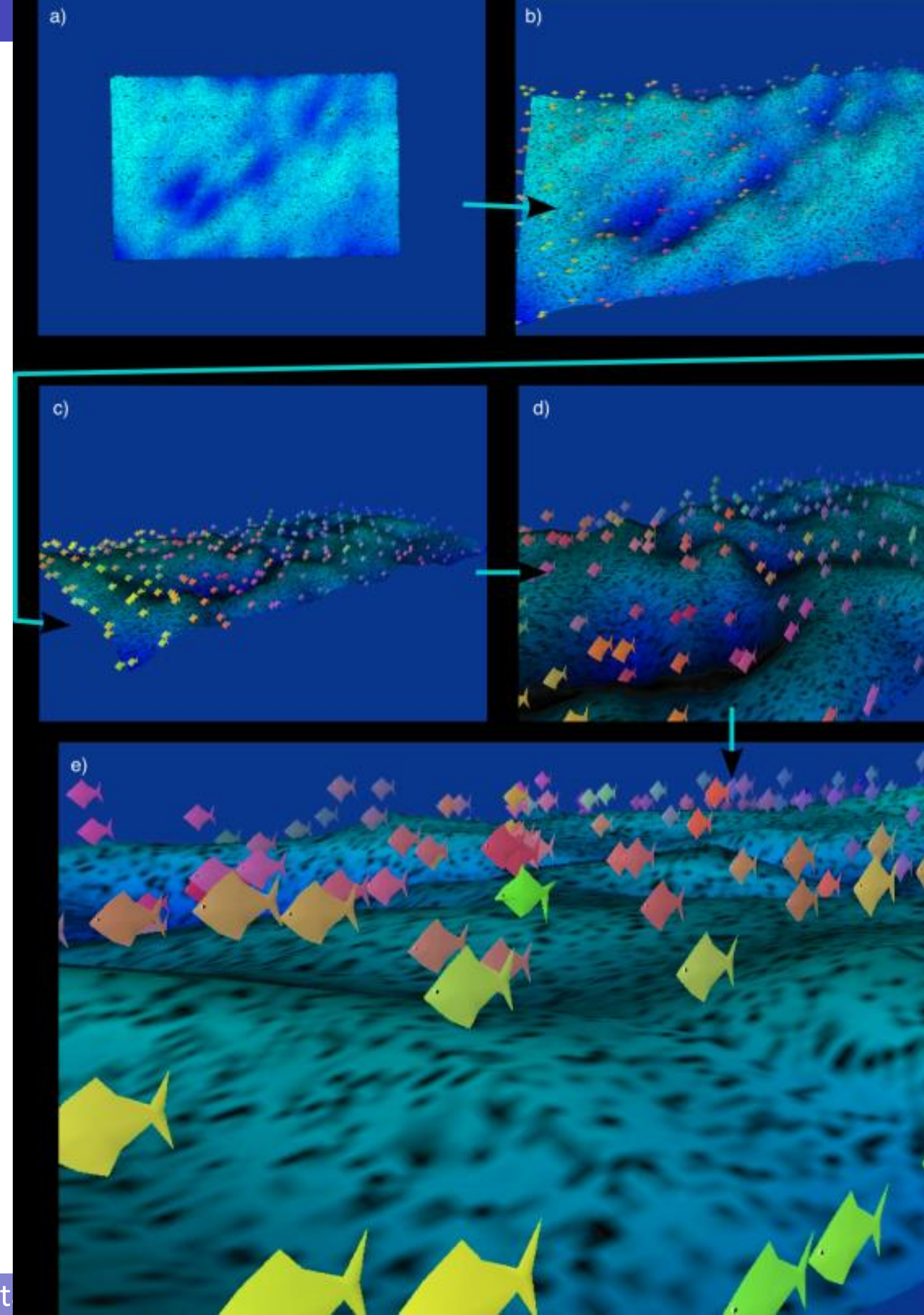




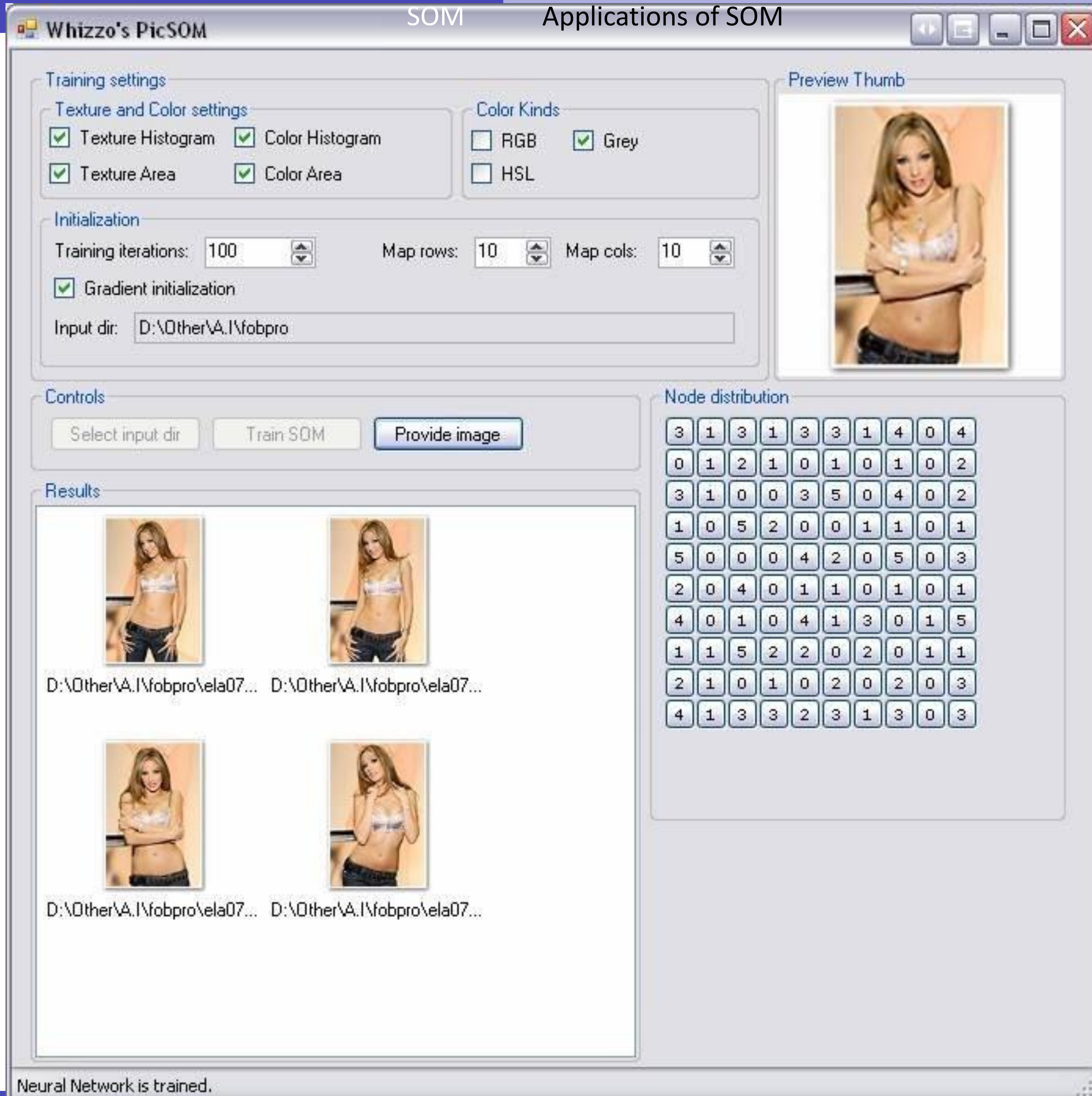
ReefSOM

- http://www.brains-minds-media.org/archive/305





at


Whizzo's PicSOM
Training settings
Texture and Color settings
Texture Histogram
Color Histogram
Texture Area
Color Area
Color Kinds
RGB
Grey
HSL
Initialization
Training iterations: 100
Map rows: 10
Map cols: 10
Gradient initialization
Input dir: D:\Other\A.I\fobpro
Preview Thumb
Controls
Select input dir
Train SOM
Provide image
Node distribution
3 1 3 1 3 3 1 4 0 4
0 1 2 1 0 1 0 1 0 2
3 1 0 0 3 5 0 4 0 2
1 0 5 2 0 0 1 1 0 1
5 0 0 0 4 2 0 5 0 3
2 0 4 0 1 1 0 1 0 1
4 0 1 0 4 1 3 0 1 5
1 1 5 2 2 0 2 0 1 1
2 1 0 1 0 2 0 2 0 3
4 1 3 3 2 3 1 3 0 3
Results
D:\Other\A.I\fobpro\ela07...
D:\Other\A.I\fobpro\ela07...
D:\Other\A.I\fobpro\ela07...
D:\Other\A.I\fobpro\ela07...
Neural Network is trained.

SOM features

- VQ - vector quantization, several vectors are mapped into a single neuron (its weight vector). How exactly? -> **Quantization error**.
- A compression of a dimension of the input space.
- Data topology preservation - adjacent (in the input space) vectors are mapped to adjacent (in the lattice) neurons. How good? -> **Topographical error**.
- SOM has an energy function which minimizes -> **distortion**.

Quantization error of SOM

- The average distance between each data vector and its BMU.
- Determines the accuracy of the mapping (vector quantization) - we already know

$$E_d = \sum_{i}^{\mathcal{N}} \sum_{j}^{\mathcal{M}} h_{b_i j} \|\mathbf{c_i} - \mathbf{m_j}\|$$

- c_i is a weight vector of a neuron
- m_j is a data vector
- h_{bij} is a neighborhood function

Topographical error of SOM

- As a percentage of the number of samples which have no preserved topology.
- The number of input vectors for which the winning neuron and the second winning neuron are not neighbors in the lattice.

$$\epsilon_t = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} u(\mathbf{c}_i)$$

  - u(ci) equals 1, if there are no neighbors, 0 otherwise

Application areas of cluster analysis

- Searching for similarities in data
- Determining the significance of variables
- Detection of remote instances (outliers)
- Data reduction

Software:

- Interesting and useful SW SOM_PAK:
  - http://www.cis.hut.fi/research/som_pak
  - http://service.felk.cvut.cz/courses/36NAN
- **Matlab SOM toolbox**
  - http://www.cis.hut.fi/projects/somtoolbox/
- SOMPAK addon
  - http://neuron.felk.cvut.cz/~jurikm
- Zooming SOM
  - http://service.felk.cvut.cz/courses/36NAN
- TKM, RSOM
  - http://service.felk.cvut.cz/courses/36NAN