Computational Intelligence: Methods and Applications

Lecture 9
Self-Organized Mappings

Włodzisław Duch
SCE, NTU, Singapore
Google: Duch

Brain maps

Tactile, motor, and olfactory data are most basic.

Such data is analyzed by animal brains using topographical organization of the brain cortex.

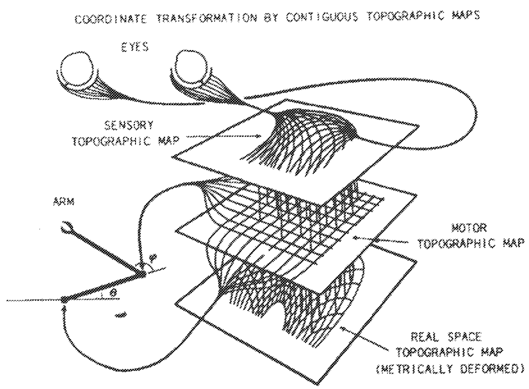
- Somatosensory maps for tactile, temperature, pain, itching, and vibration signals.
- Motor maps in frontal neocortex and cerebellum cortex.
- Auditory tonotopic maps in temporal cortex.
- Visual orientation maps in primary visual cortex.
- Multimodal orientation maps (superior colliculus)

Senso-motoric map

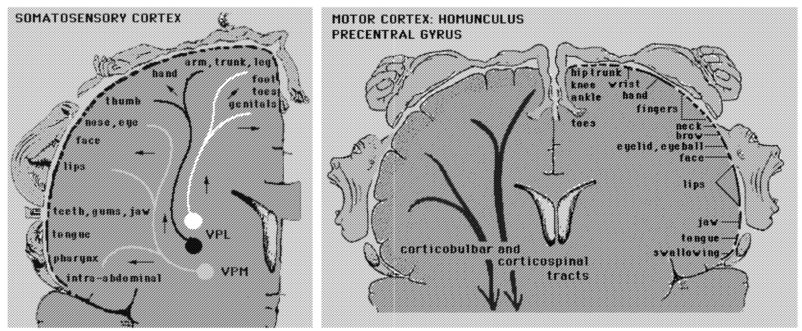
Visual signals are analyzed by maps coupled with motor maps and providing senso-motoric responses.

Figure from:

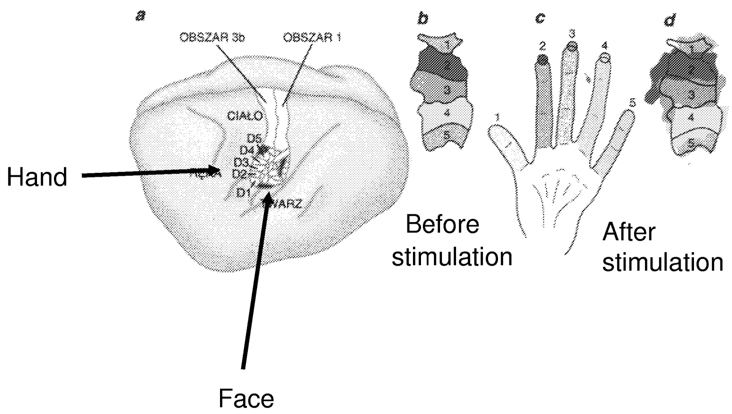
P.S. Churchland, T.J. Sejnowski,
The computational brain.
MIT Press, 1992



Somatosensoric and motor maps



Representation of fingers



Models of self-organization

SOM or SOFM (Self-Organized Feature Mapping) – self-organizing feature map, one of the simplest models.

How can such maps develop spontaneously?

Local neural connections: neurons interact strongly with those nearby, but weakly with those that are far (in addition inhibiting some intermediate neurons).

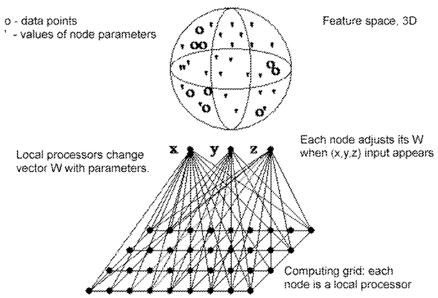
History:

von der Malsburg and Willshaw (1976), competitive learning, Hebb mechanisms, „Mexican hat” interactions, models of visual systems.

Amari (1980) – models of continuous neural tissue.

Kohonen (1981) - simplification, no inhibition; leaving two essential factors: competition and cooperation.

Self-Organized Map: idea



Data: vectors $\mathbf{X}^T = (X_1, \ldots X_d)$ from d-dimensional space.

Grid of nodes, with local processor (called neuron) in each node.

Local processor # *j* has *d* adaptive parameters $\mathbf{W}^{(j)}$.

Goal: change $\mathbf{W}^{(j)}$ parameters to recover data clusters in **X** space.

SOM algorithm: competition

Nodes should calculate similarity of input data to their parameters.
Input vector **X** is compared to node parameters **W.**
Similar = minimal distance or maximal scalar product.

Competition: find node $j=c$ with **W** most similar to **X**.

$$\left\|\mathbf{X}-\mathbf{W}^{(j)}\right\| = \sqrt{\sum_i \left(X_i - W_i^{(j)}\right)^2}$$

$$c = \arg\min_j \left\|\mathbf{X}-\mathbf{W}^{(j)}\right\|$$

Node number c is most similar to the input vector **X**
It is a winner, and it will learn to be more similar to **X**, hence this is a “competitive learning” procedure.

Brain: those neurons that react to some signals pick it up and learn.

SOM algorithm: cooperation

Cooperation: nodes on a grid close to the winner *c* should behave similarly. Define the "neighborhood function" *O*(*c*):

$$h(r, r_c, t) = h_0(t) \exp\left(-\|r - r_c\|^2 / \sigma_c^{\,2}(t)\right)$$

t – iteration number (or time);

$\boldsymbol{r_c}$ – position of the winning node *c* (in physical space, usually 2D).

$\|\boldsymbol{r}-\boldsymbol{r_c}\|$ – distance from the winning node, scaled by $\sigma_c(t)$.

$h_0(t)$ – slowly decreasing multiplicative factor

The neighborhood function determines how strongly the parameters of the winning node and nodes in its neighborhood will be changed, making them more similar to data **X**

SOM algorithm: dynamics

Adaptation rule: take the winner node *c*, and those in its neighborhood $O(r_c)$, change their parameters making them more similar to the data **X**

$$\text{For } \forall i \in O(c)$$

$$\mathbf{W}^{(i)}(t+1) = \mathbf{W}^{(i)}(t) + h(r_i, r_{c,} t)\left[\mathbf{X}(t) - \mathbf{W}^{(i)}(t)\right]$$

Select randomly new sample vector X, and repeat.

Decrease $h_0(t)$ slowly until there will be no changes.

Result:

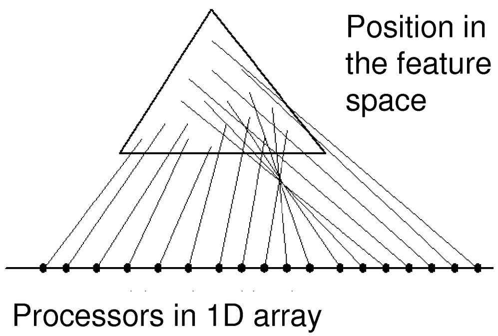
- $W^{(i)} \approx$ the center of local clusters in the X feature space
- Nodes in the neighborhood point to adjacent areas in X space

SOM algorithm

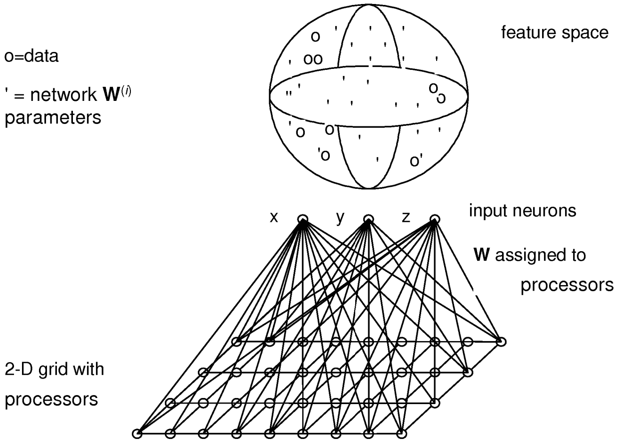
$\mathbf{X}^T=(X_1, X_2 \,..\, X_d)$, samples from feature space.
Create a grid with nodes $i = 1 \,..\, K$ in 1D, 2D or 3D,
each node with d-dimensional vector $\mathbf{W}^{(i)T} = (W_1^{(i)}\ W_2^{(i)} \,..\, W_d^{(i)})$,
$\mathbf{W}^{(i)} = \mathbf{W}^{(i)}(t)$, changing with t – discrete time.

1. Initialize: random small $\mathbf{W}^{(i)}(0)$ for all $i=1...K$.
   Define parameters of neighborhood function $h(|r_i - r_c|/\sigma(t), t)$
2. Iterate: select randomly input vector $\mathbf{X}$
3. Calculate distances $d(\mathbf{X}, \mathbf{W}^{(i)})$, find the winner node $\mathbf{W}^{(c)}$ most similar (closest to) $\mathbf{X}$
4. Update weights of all neurons in the neighborhood $O(r_c)$
5. Decrease the influence $h_0(t)$ and shrink neighborhood $\sigma(t)$.
6. If in the last T steps all $\mathbf{W}^{(i)}$ changed less than ε then stop.

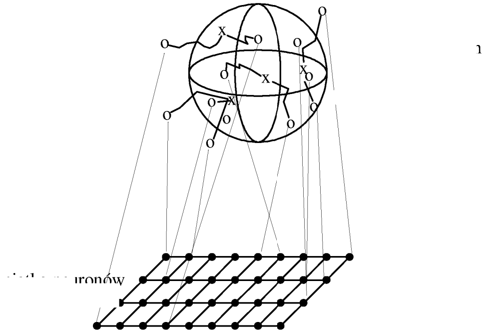
1D network, 2D data



2D network, 3D data

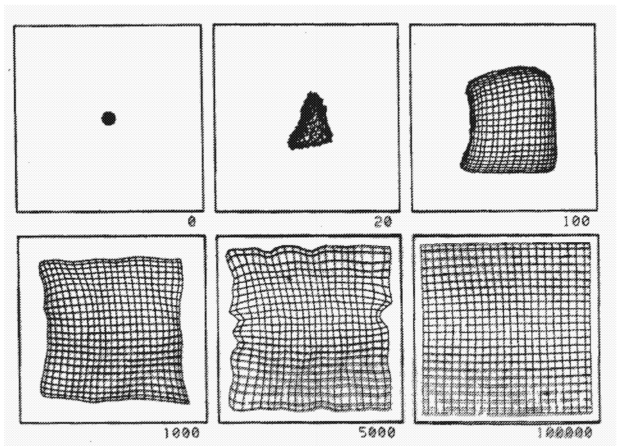


Training process



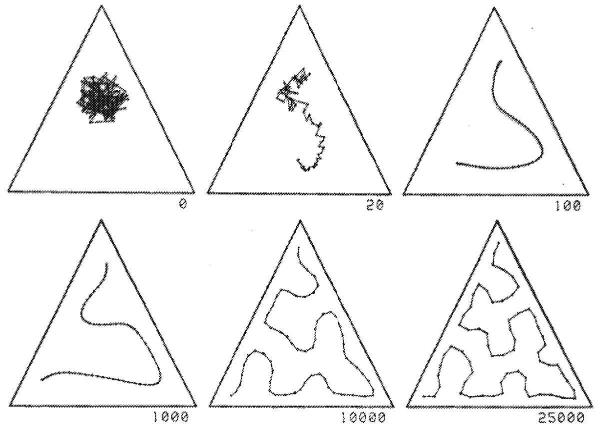
Java demos:
http://www.neuroinformatik.ruhr-uni-bochum.de/
ini/VDM/research/gsn/DemoGNG/GNG.html

2D => 2D, square



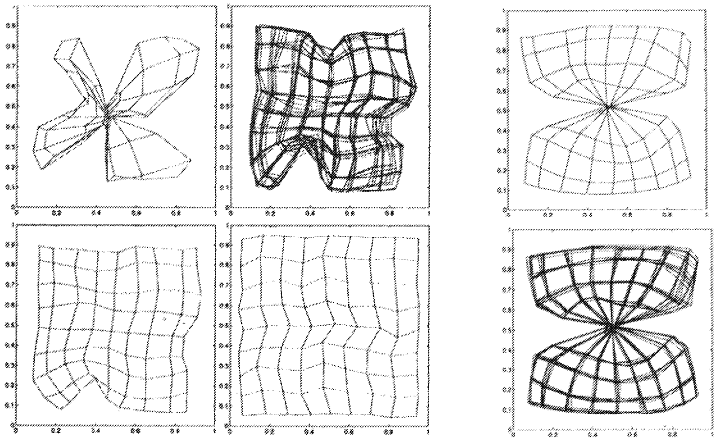
Initially all W≈0, pointing to the center of the 2D space, but over time they learn to point at adjacent positions with uniform distribution.

2D => 1D in a triangle



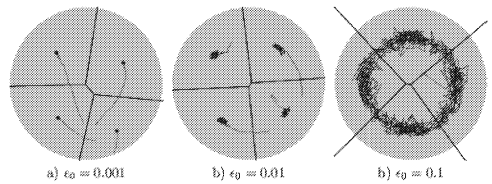
The line in the data space forms a Peano curve, an example of a fractal. Why?

Map distortions

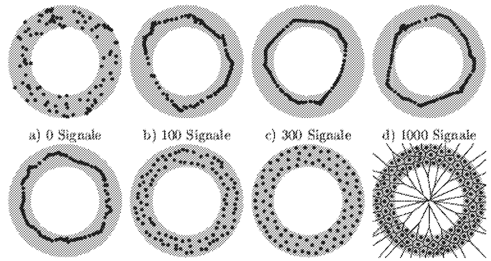


Initial distortions may slowly disappear or may get frozen ... giving the user a completely distorted view of reality.

Learning constant



Large learning constants: point on the map move constantly, slow stabilization.



Uniform distribution of data points within the torus lead to formation of maps that have uniform distribution of parameters (codebook vectors).

Demonstrations with GNG

Growing Self-Organizing Networks demo

Parameters in the SOM program:

t – iterations
$\varepsilon(t) = \varepsilon_i (\varepsilon_f / \varepsilon_i)^{t/tmax}$ to reduce the learning step
$\sigma(t) = \sigma_i (\sigma_f / \sigma_i)^{t/tmax}$ to reduce the neighborhood size

$$h(r, r_c, t, \varepsilon, \sigma) = \varepsilon(t) \exp\left(-\|r - r_c\|^2 / \sigma^2(t)\right)$$

Try some 1x30 maps to see forming of Peano curves.