Self-organizing maps

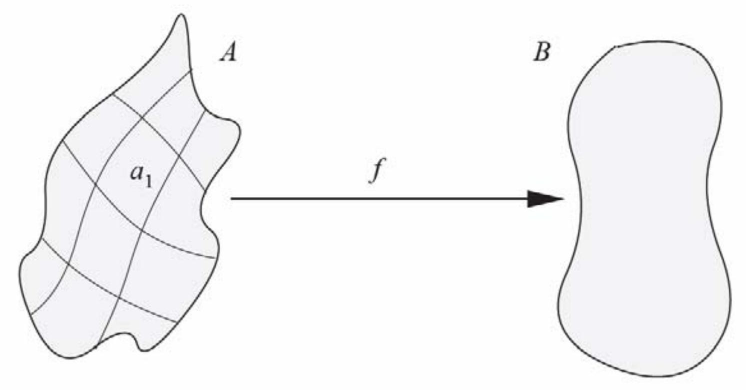
Part II

Kohonen networks

Marten Jäger

Self-organizing networks - introduction

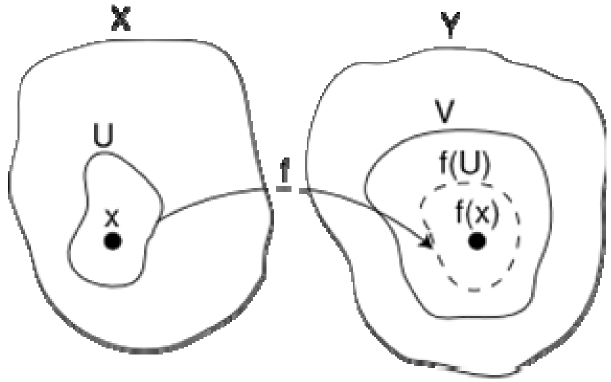
- Most popular self-organizing network: Kohonen maps (*by Teuvo Kohonen*)
  - → Topology preserving maps
  - ❑ Computes a function f defined from an inputspace A to an outputspace B
    - with dimension(A) ≥ dimension(B)



A function $f : A \rightarrow B$

Neural Networks - A Systematic Introduction *by Raul Rojas fig.15.1*

Continuous function (topology)



http://upload.wikimedia.org/wikipedia/commons/thumb/a/a7/Continuity_topology.svg/300px-Continuity_topology.svg.png

- is a function *f* where a set of points near *f(x)* always contain the image of a set of points near *x*

  or

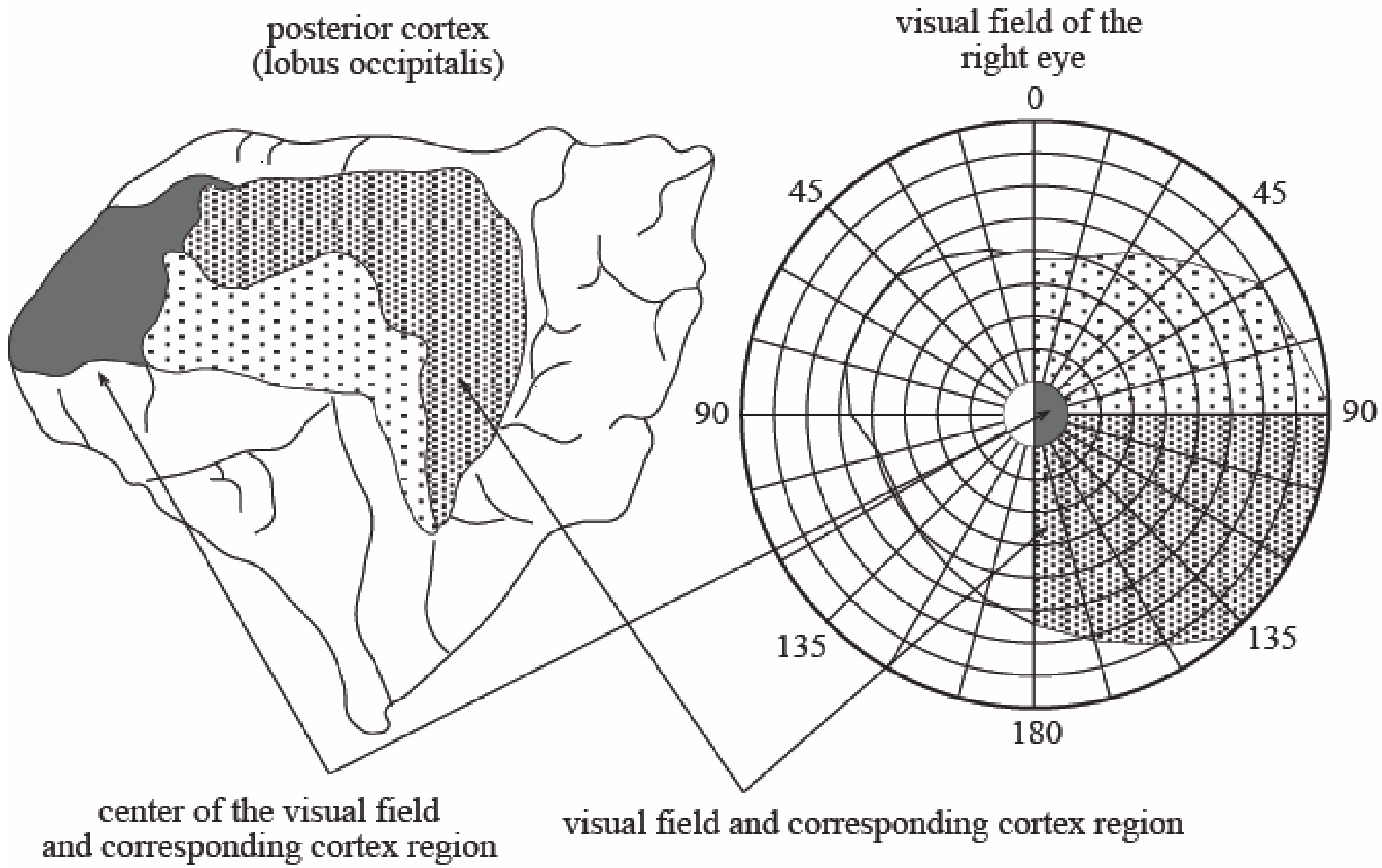
- a neighbourhood of *f(x)* always contains the image of a neighbourhood of *x*

Self-organizing networks - introduction

- Kohonen model has mathematical & biological background
  - many stuctures in brain have linear or planar topology (1. or 2. Dimensions)
  - on the other hand sensory experience is multidimesional

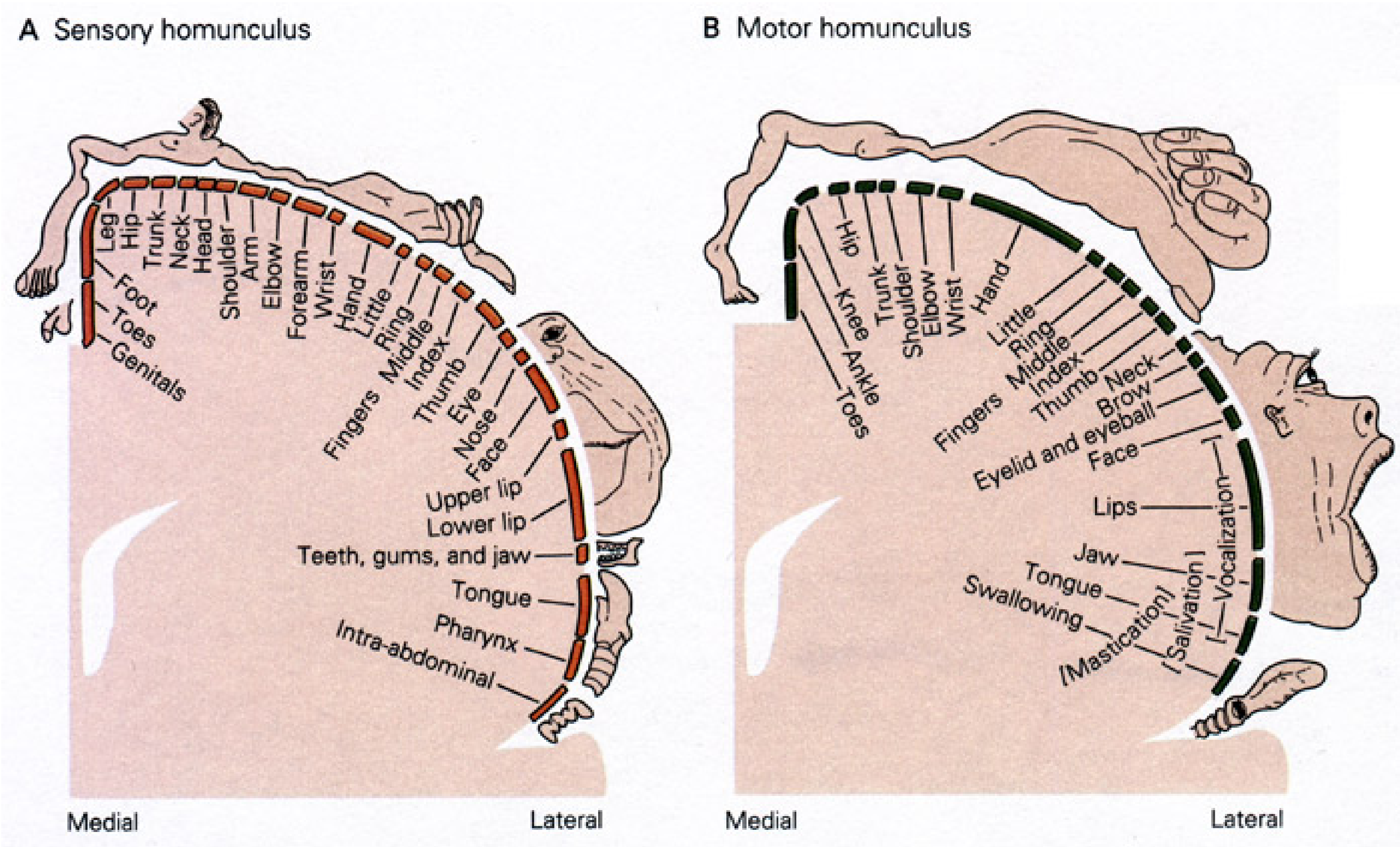
- Question:
  - how is the multidimensional input projected to the 2 dimensional neuronal structure

Mapping of the visual field on the cortex



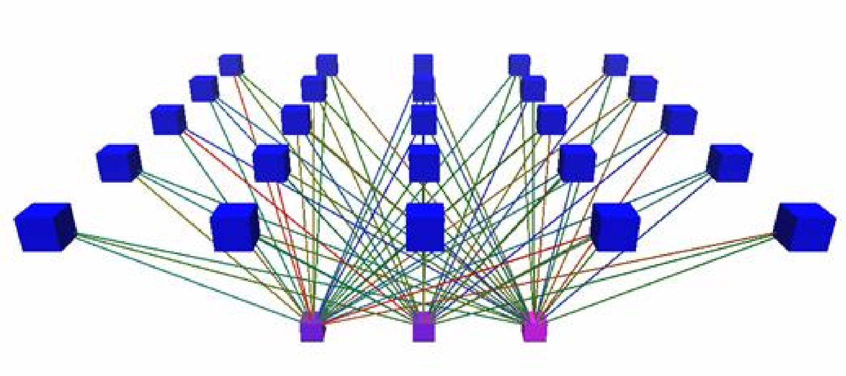
→ The visual cortex is a kind of map of the visual field

The somatosensory and motor cortex



http://pharyngula.org/~pzmyers/neuro/chap9/homunculi.jpg

Kohonen - idea



http://johannes.lampel.net/bll137-Dateien/image044.jpg

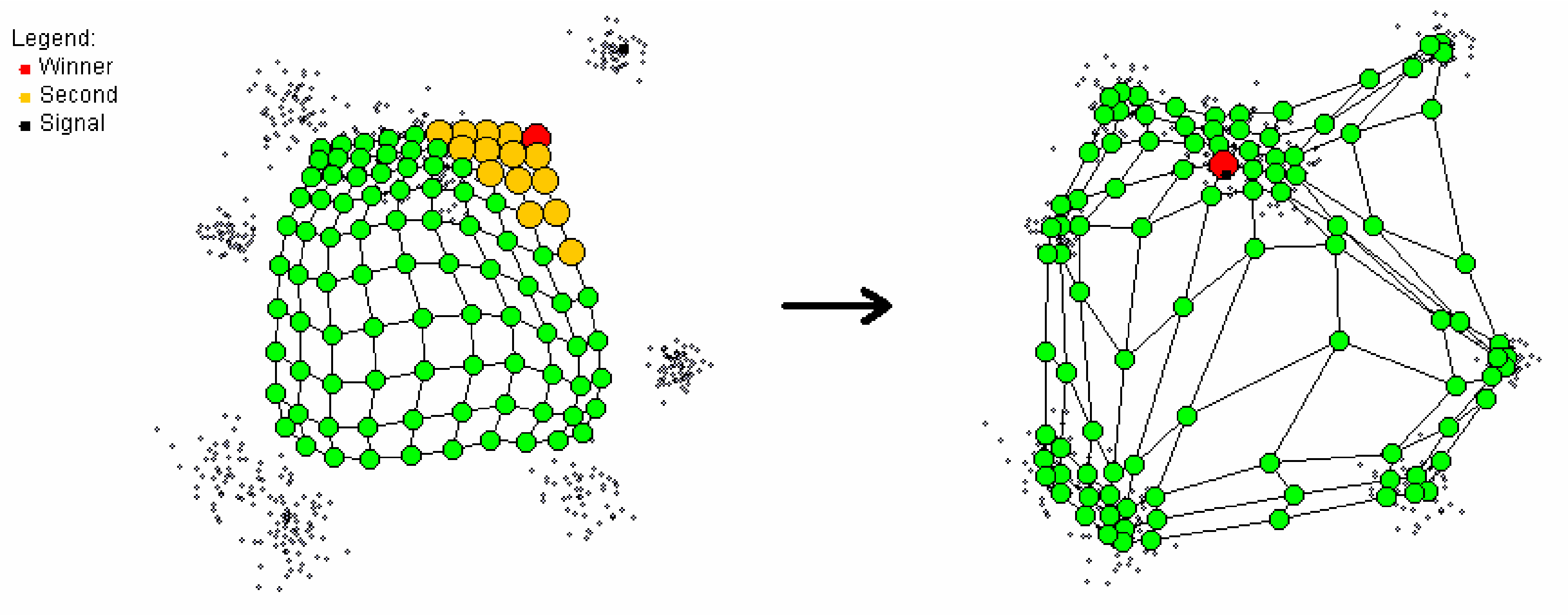
There is a give training set M with $M = \{m_j = (x_j) \mid x_j \in X \subseteq \Re^n, j = 1, \ldots, \mu_M\}$

and

There is a given set of neurons N with $N = \{n_i = (w_i, k_i) \mid w_i \in X \subseteq \Re^n, k_i \in K^2, i = 1, \ldots, \mu_N\}$

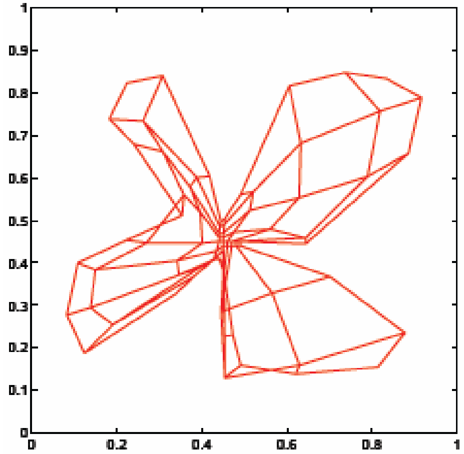
Sample

- SOM applet from the „Universität Bochum“
  - http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/GNG.html



Algorithm – start

Start: The n-dimensional weight vectors $w_1 \ldots w_{\mu_N}$ are selected at random. An initial radius δ, a learning constant ε, and a neighbourhood function d_X are selected.



There is a given set of neurons N with $N = \{n_i = (w_i, k_i) \mid w_i \in X \subseteq \Re^n, k_i \in K^2, i = 1, \ldots, \mu_N\}$

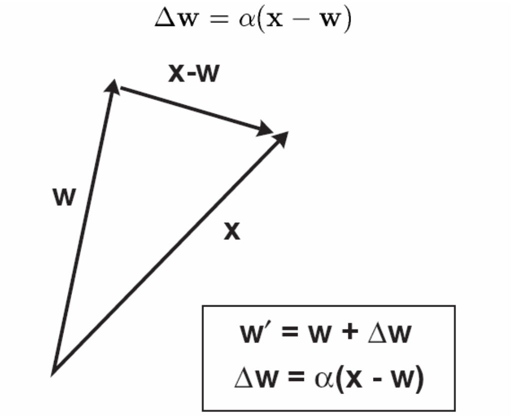
Algorithm – step 1

Step 1: Select an input vector x using the desired probability distribution over the input space.
Or use the training set.

There is a give training set M with $M = \{m_j = (x_j) \mid x_j \in X \subseteq \Re^n, j = 1, \ldots, \mu_M\}$

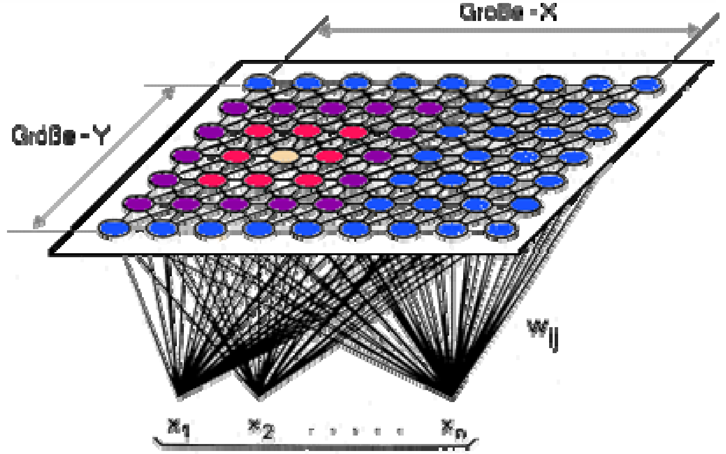
Algorithm – step 2

Step 2: The unit n_s^t with the maximum excitation is selected (that is, for which the distance between w_i and x is minimal) and the set of neurons $N^{+t} \subset N^t$ which lay in between the radius δ of n_s^t.



vector triangle - weights and inputs

ftp://ftp.shef.ac.uk/pub/uni/academic/N-Q/psology/gurney/NeuralNets/l7.ps



http://www.statistics4u.info/fundstat_germ/img/hl_kohonen1.png

Winner neuron: $d_X\left(x_j^t, w_s^t\right) = \min\left\{d_X\left(x_j^t, w_i^t\right) \mid i = 1,\ldots,\mu_N\right\}$

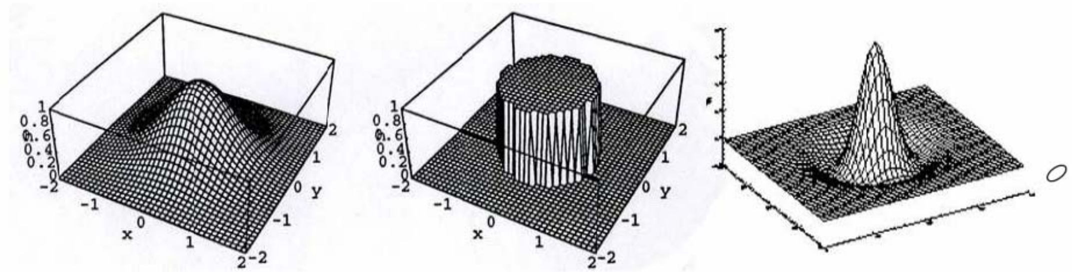
Neighborhood: $N^{+t} = \left\{n_i = \left(w_i, k_i\right) \mid d_A\left(k_s, k_i\right) \le \delta^t\right\}$

Algorithm – step 3

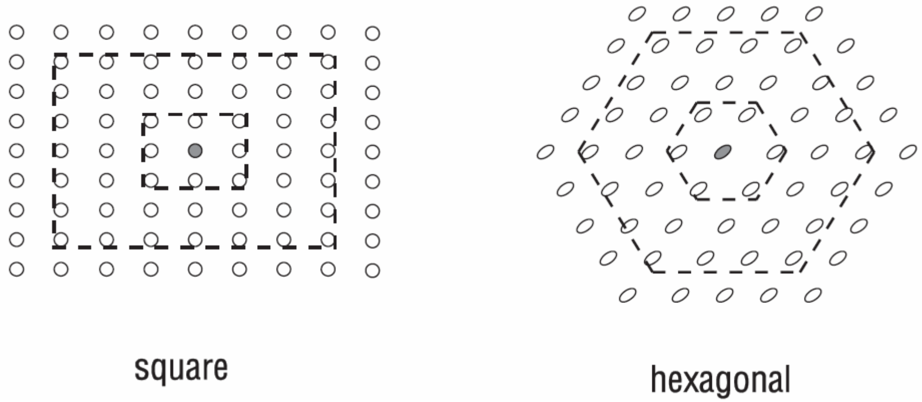
Step 3: The weight vectors of N^{+t} are updated using the neighbourhood function and the update rule:

$$w_i^{t+1} = w_i^t + \varepsilon^t \cdot h_{si}^t \cdot d_X\left(x_j^t, w_i^t\right)$$

$$h_{si}^t = \varepsilon^{\frac{-d_A\left(k_s, k_i\right)^2}{2\cdot{\delta^t}^2}}$$



http://www.informatik.uni-ulm.de/ni/Lehre/WS04/ProSemNN/pdf/SOM.pdf



ftp://ftp.shef.ac.uk/pub/uni/academic/N-Q/psology/gurney/NeuralNets/l7.ps

Algorithm – step 4

Step 4: Stop if the maximum number of iterations has been reached;
Otherwise recalculate the radius δ and the learning constant ε.

$$\varepsilon^t = \varepsilon_{start} \cdot \left(\frac{\varepsilon_{end}}{\varepsilon_{start}} \right)^{\frac{t}{t_{\max}}} \qquad \delta^t = \delta_{start} \cdot \left(\frac{\delta_{end}}{\delta_{start}} \right)^{\frac{t}{t_{\max}}}$$

Algorithm

Start: The n-dimensional weight vectors $w_1 \ldots w_{\mu_N}$ are selected at random. An initial radius δ, a learning constant ε, and a neighbourhood function d_X are selected.

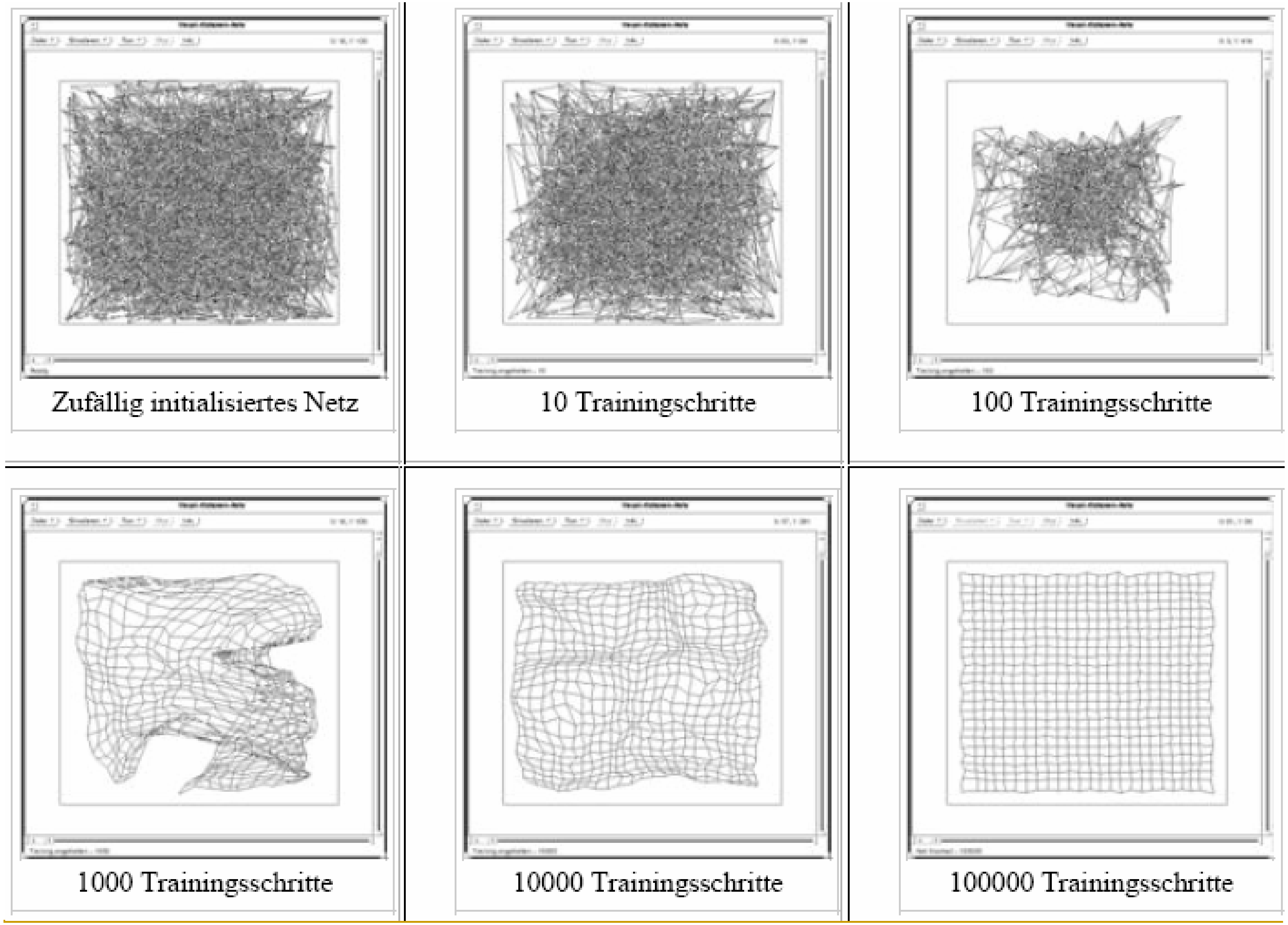
Step 1: Select an input vector x using the desired probability distribution over the input space. Or use the training set.

Step 2: The unit n_s^t with the maximum excitation is selected (that is, for which the distance between w_i and x is minimal) and the set of neurons $N^{+t} \subset N^t$ which lay in between the radius δ of n_s^t.

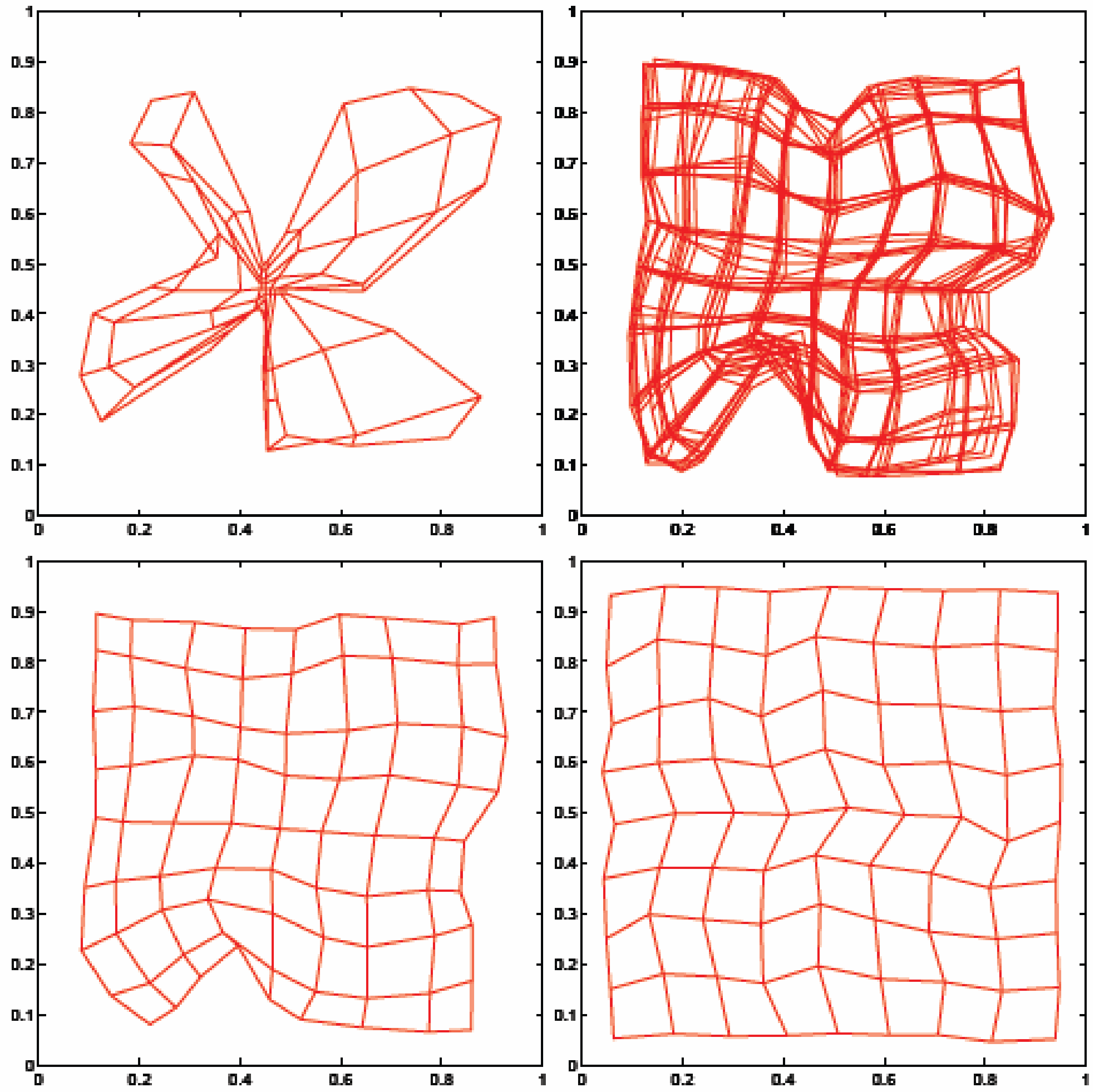
Step 3: The weight vectors of N^{+t} are updated using the neighbourhood function and the update rule:

$$w_i^{t+1} = w_i^t + \varepsilon^t \cdot h_{si}^t \cdot d_X\left(x_j^t, w_i^t\right)$$

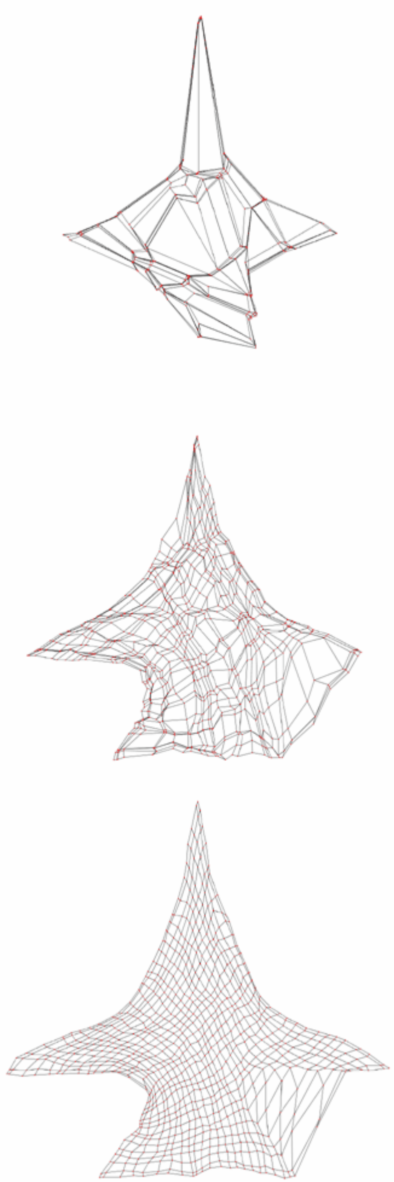
Step 4: Stop if the maximum number of iterations has been reached; Otherwise recalculate the radius δ and the learning constant ε.

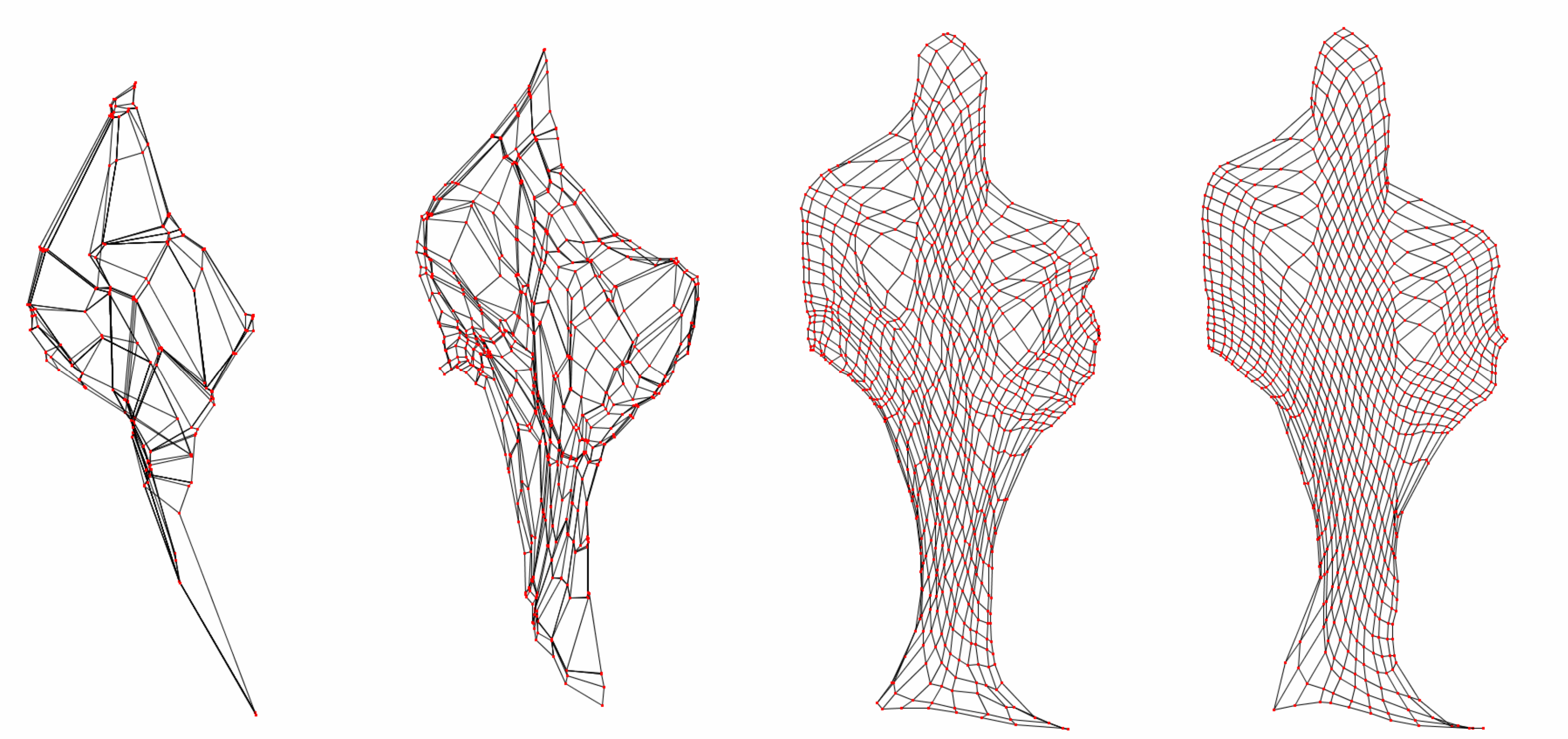


http://de.wikipedia.org/wiki/Self-Organizing_Maps



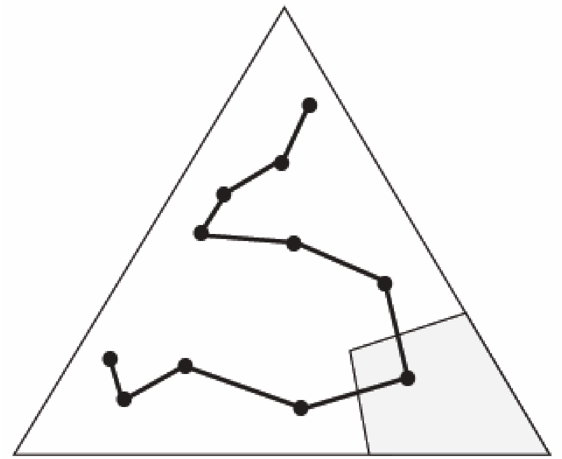
Neural Networks - A Systematic Introduction *by Raul Rojas fig.15.7*





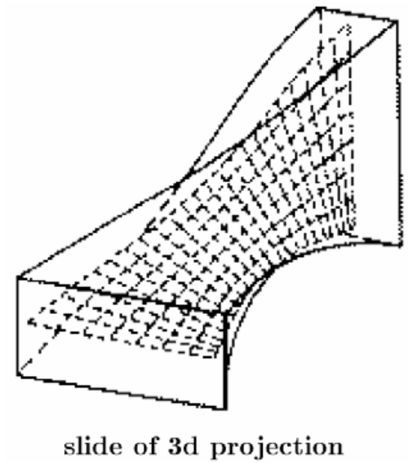
Kohonen maps – reduction of dimensions

- 1-dimensional Kohonen map spanning a 2-dimensional triangle



Neural Networks - A Systematic Introduction *by Raul Rojas fig.15.´5*

- 2-dimensional Kohonen map spanning a 3-dimensional figure



ftp://ftp.shef.ac.uk/pub/uni/academic/N-Q/psology/gurney/NeuralNets/l7.ps

Literature

- Neural Networks - A Systematic Introduction *a book by Raul Rojas* Springer-Verlag, Berlin, New-York, 1996 (502 p.,350 illustrations)
- Referenzen in Wikipedia-Eintrag Selbstorganisierende_Karte (http://de.wikipedia.org/wiki/Selbstorganisierende_Karte)
- Neural Nets by Kevin Gurney (http://www.shef.ac.uk/psychology/gurney/notes/)
- Willshaw D., Self-organisation in the nervous system.
- Obermayer K., Statistical-mechanical analysis of self-organization and pattern formation during the development of visual maps