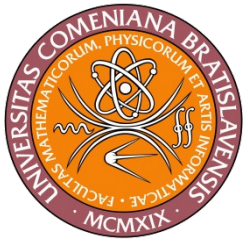
Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava



Neural Networks

Lecture 6

Unsupervised models

Igor Farkaš

2024

Self-organization

"Global order can arise from local interactions." (Turing, 1952)

Architectures of self-organizing NNs:

- single-layer with feedforward (ffwd) connections
- single-layer with ffwd and lateral connections
- multi-layer with ffwd connections b/w layers

Principles of self-organization (von der Malsburg, 1990):

1. Self-amplification – weight modifications tend to self-amplify
2. Competition – among neurons, due to limited resources
3. Cooperation – among neighboring neurons
4. Structural information (in input data) – is acquired as knowledge.

Applications of self-organizing NNs

- Self-organizing NNs are better at dealing with the scaling problem, because their learning rules are simpler and are mostly local
- Types of tasks:
  - Feature extraction – principal components (PCA)
  - Data coding with max. information preservation (PCA)
  - Data clustering (vector quantization) → SOM
  - Data visualisation (feature mapping) → SOM
  - Links to: (a) manifold learning, (b) growing/dynamic models (graphs)

Principal Component Analysis

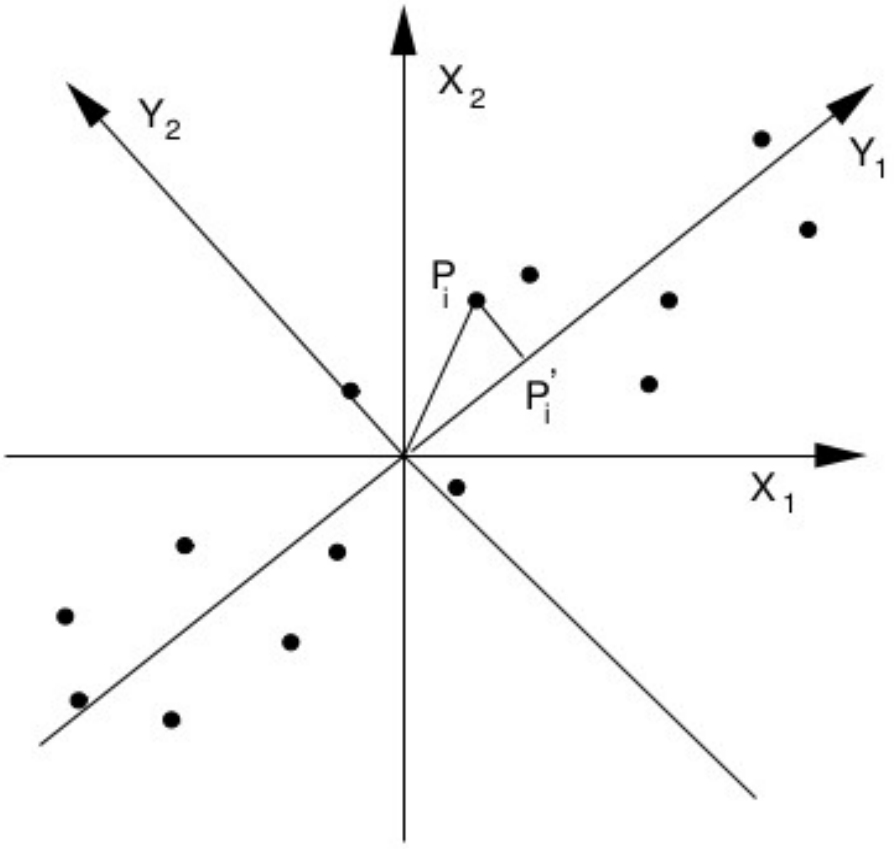
- PCA (Pearson, 1901) – linear transformation into feature space – typically with lower dimensionality
- Assume n-dim. (column) vectors $\boldsymbol{x}$, with zero means, $\langle \boldsymbol{x} \rangle = \mathbf{0}$.
- Let $\mathbf{R} = \langle \boldsymbol{x}\boldsymbol{x}^T \rangle$ be the covariance matrix of input data.
- Then PCA finds orthogonal directions (projections) that maximize variance of original data; $\mathbf{R}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$, $i = 1,2,\ldots, n$
- λ_i are eigenvalues, associated with eigenvectors $\boldsymbol{u}_i$ of $\mathbf{R}$.
- Eigenvectors are mutually orthogonal
- Eigenvalues (real values here) are sorted $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$

PCA example

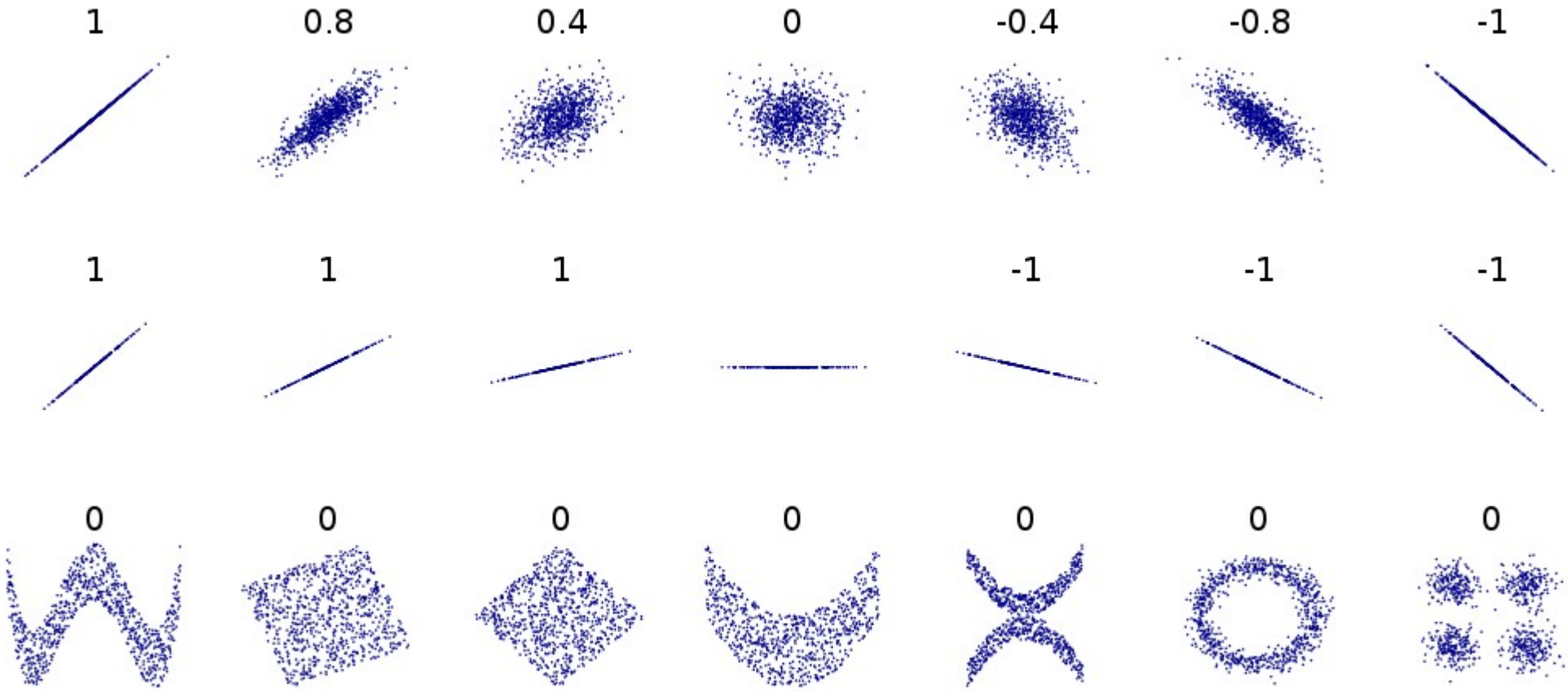
X = original space (2D)
Y = projection space (2D)

Direction Y1 captures maximum variance in original 2D data.

Y2 captures the remaining variance.



Types of correlations in data



Correlation coefficient only captures linear relationships

Data representations

- For n different eigenvectors $\boldsymbol{u}_i$ we get projections (*analysis*) that we call principal components:

$$a_j = \boldsymbol{u}_j^{\mathrm{T}} \boldsymbol{x} = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{u}_j \qquad j = 1,2,\ldots,n$$

- During reconstruction of $\boldsymbol{x}$ from projections a_j we combine these into the vector $\boldsymbol{a} = [a_1, a_2, \ldots, a_n]^{\mathrm{T}}$ and then

$$\boldsymbol{a} = \left[\boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{x}, \boldsymbol{u}_2^{\mathrm{T}} \boldsymbol{x}, \ldots, \boldsymbol{u}_n^{\mathrm{T}} \boldsymbol{x}\right]^{\mathrm{T}} = \boldsymbol{U}^{\mathrm{T}} \boldsymbol{x}$$

- Reconstruction of $\boldsymbol{x}$ (synthesis) $\qquad \boldsymbol{x} = \mathbf{U}\, \boldsymbol{a} = \sum_{j=1}^{n} a_j \boldsymbol{u}_j$
- PCA is a coordinate transf. of $\boldsymbol{x}$ into feature space (point $\boldsymbol{a}$).
- for *p<n:*

$$\boldsymbol{x}' = \mathbf{U}'\, \boldsymbol{a} = \sum_{j=1}^{p} a_j \boldsymbol{u}_j \qquad \boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}' = \sum_{j=p+1}^{n} a_j \boldsymbol{u}_j$$

Hebbian learning – single neuron

- Canadian psychologist Donald Hebb (1949) postulated:

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

- Let us assume one linear neuron with n inputs:

$$y = \sum_{j=1}^{n} w_j x_j = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}$$

- According to Hebb's postulate:

$$w_j(t+1) = w_j(t) + \alpha\, y\, x_j \quad \text{for} \quad j = 1,2,\ldots,n$$

where α is the learning rate.

- Oja's rule: $w_j(t+1) = w_j(t) + \alpha\, y\, x_i - \alpha\, y^2 w_j(t)$

Hebbian term

stabilizing term

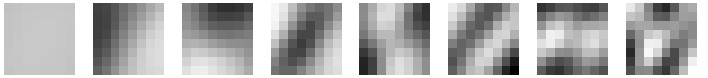
Single-unit behavior

Properties of linear neuron trained by Oja's rule with $\langle \mathbf{x} \rangle = \mathbf{0}$:

- weight vector $\boldsymbol{w}$ converges to match the eigenvector of $\mathbf{R} = \langle \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} \rangle$
- after convergence, $\boldsymbol{w}$ maximizes $\langle y^2 \rangle$.
- after convergence, $\|\boldsymbol{w}\| = 1$.

- Hence, the linear Hebbian neuron projects input vectors to the direction that maximizes the discrimination capability (maximum information preservation).
- If more units are used, they can perform PCA (e.g. using GHA algorithm)

Example of PCA application

- Image of 256x256 pixels, converted to gray image (0...255)
- NN trained by GHA on sub-images 8x8 (input size)
- 8 largest eigenvectors (weight vectors): 

Generalized Hebbian Algorithm

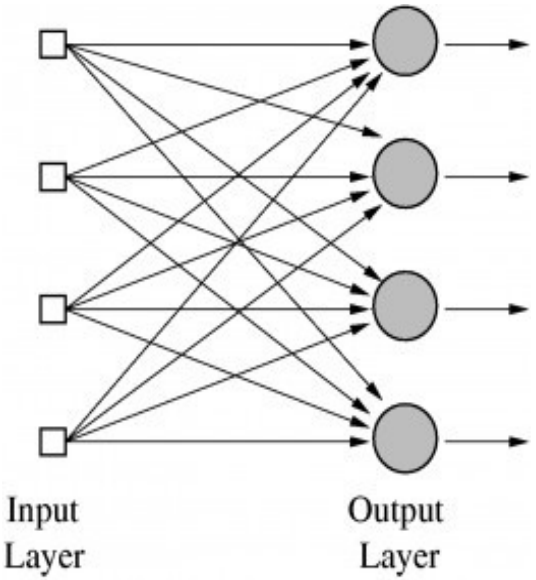
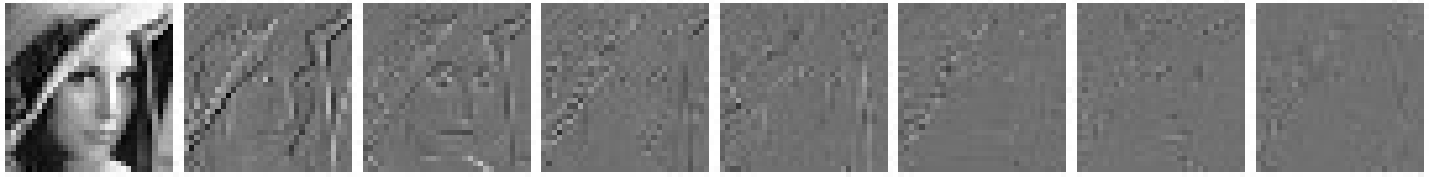


Image reconstructions using 1 to 8 dimensions

$$\Delta w_{ij} = \alpha y_i \left(x_j - \sum_{k=1}^{i} y_k w_{kj} \right)$$

Activations from 8 output neurons



PCA summary

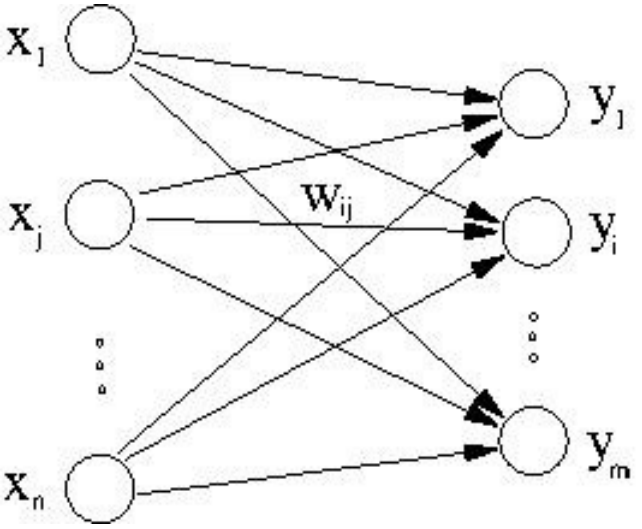
- Principal component analysis – standard method for linear reduction of data dimensionality
- Suitable for data with Gaussian distribution (i.e. leading to minimal reconstruction error)
- PCA projects onto orthogonal directions with maximal variance
- A single-layer neural net trained by Hebbian-like learning rule can implement PCA
- No need to calculate covariance matrix of input data
- Generalization works (with GHA)

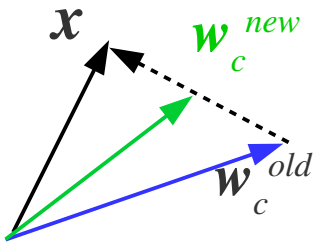
Simple competitive learning

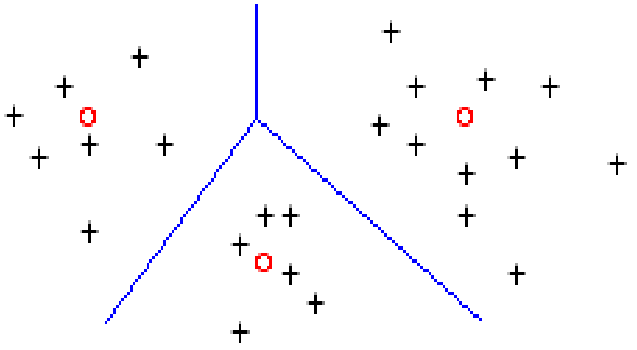
- an example of unsupervised learning
- Features:
  - linear neurons
  - winner: $y_c = \max_i \{\boldsymbol{w}_i^T . \boldsymbol{x}\}$
    - i.e. best matching unit *c*
  - winner-take-all adaptation:

    $\Delta \boldsymbol{w}_c = \alpha(\boldsymbol{x} - \boldsymbol{w}_c) \quad \alpha \in (0,1)$

    $\|\boldsymbol{w}_c\| \leftarrow 1$
  - risk of “dead” neurons
- algorithm: in each iteration
  - find winner, adapt its weights
- useful for clustering

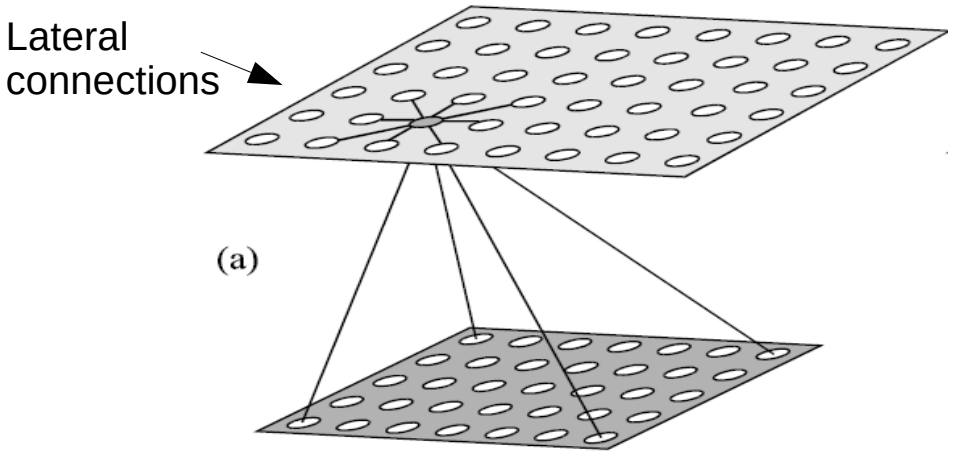




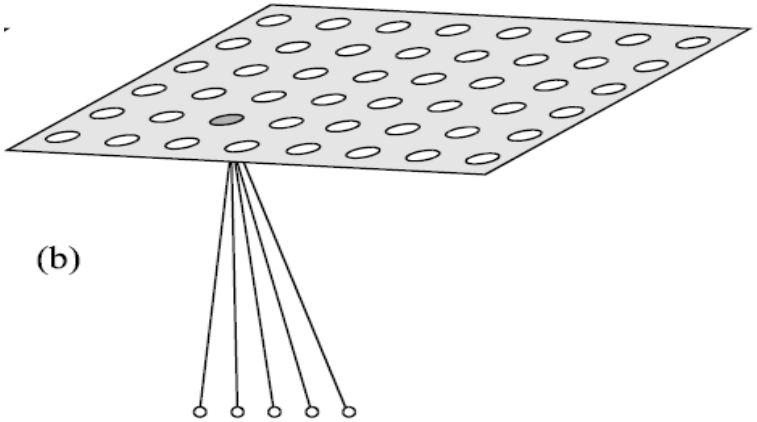


Feature mapping

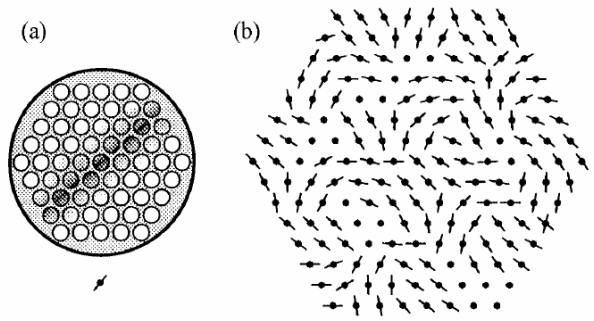
biologically motivated models

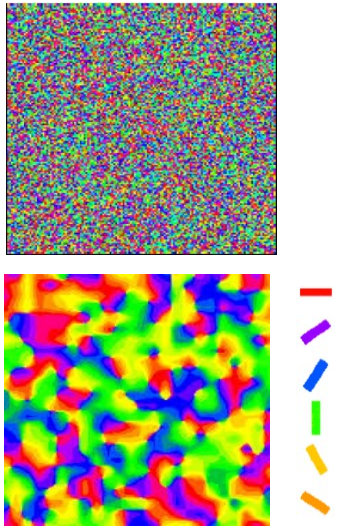


self-organizing map (SOM)



e.g. mapping from retina to cortex
➔ orientation map



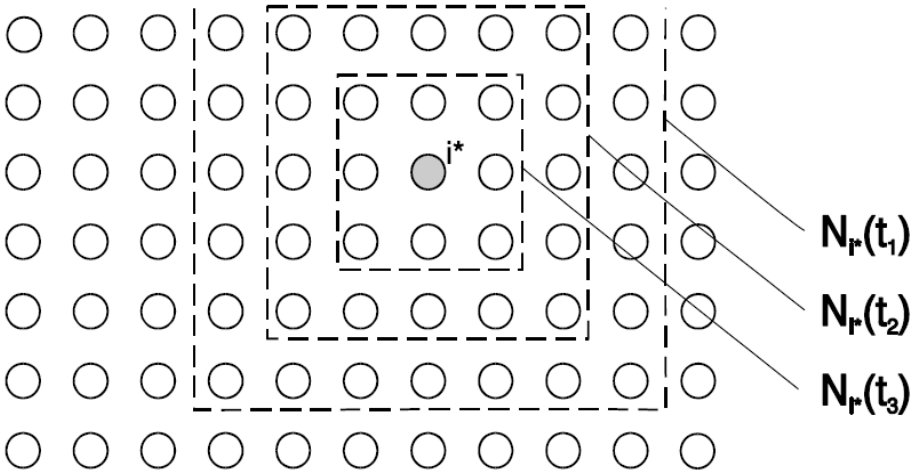


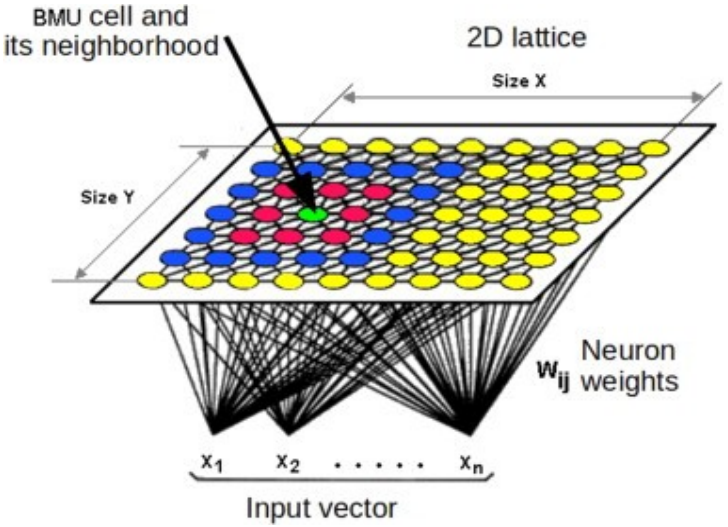
- introduced topology of neurons in the map =>
- winner-take-most due to neuron cooperation
- lateral interactions between units avoided

Neighborhood function in SOM

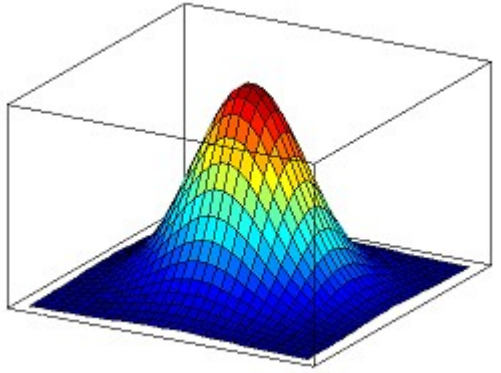
- computationally efficient substitute for lateral interactions
- neurons adapt only within the winner neighborhood
- neighborhood radius decreases in time

- rectangular neighborhood:





- alternative: Gaussian neighborhood



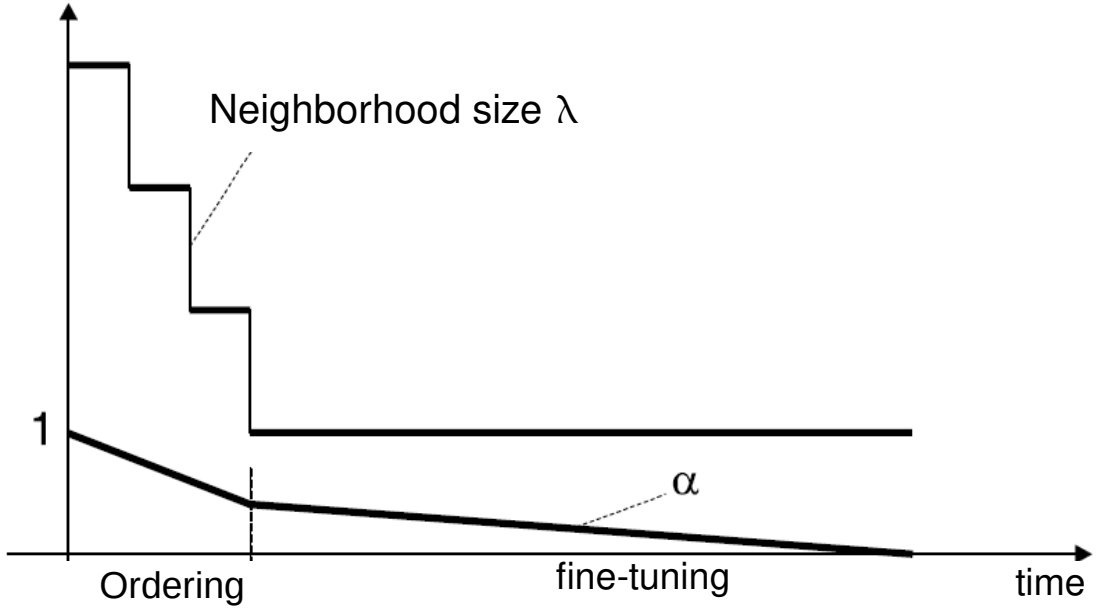
SOM algorithm

(Kohonen, 1982)

- randomly choose an input $\boldsymbol{x}$ from the training set
- find winner i^* for $\boldsymbol{x}$: $i^* = \arg\min_i \|\boldsymbol{x} - \boldsymbol{w}_i\|$
- adapt weights within the neighborhood

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + \alpha(t)\, h(i^*, i)[\boldsymbol{x}(t) - \boldsymbol{w}_i(t)]$$

- update SOM parameters (neighborhood, learning rate)
- repeat until stopping criterion is met

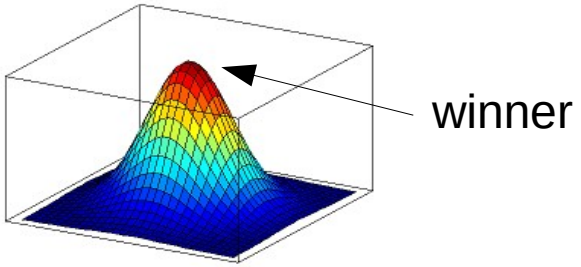


Input-output mapping:

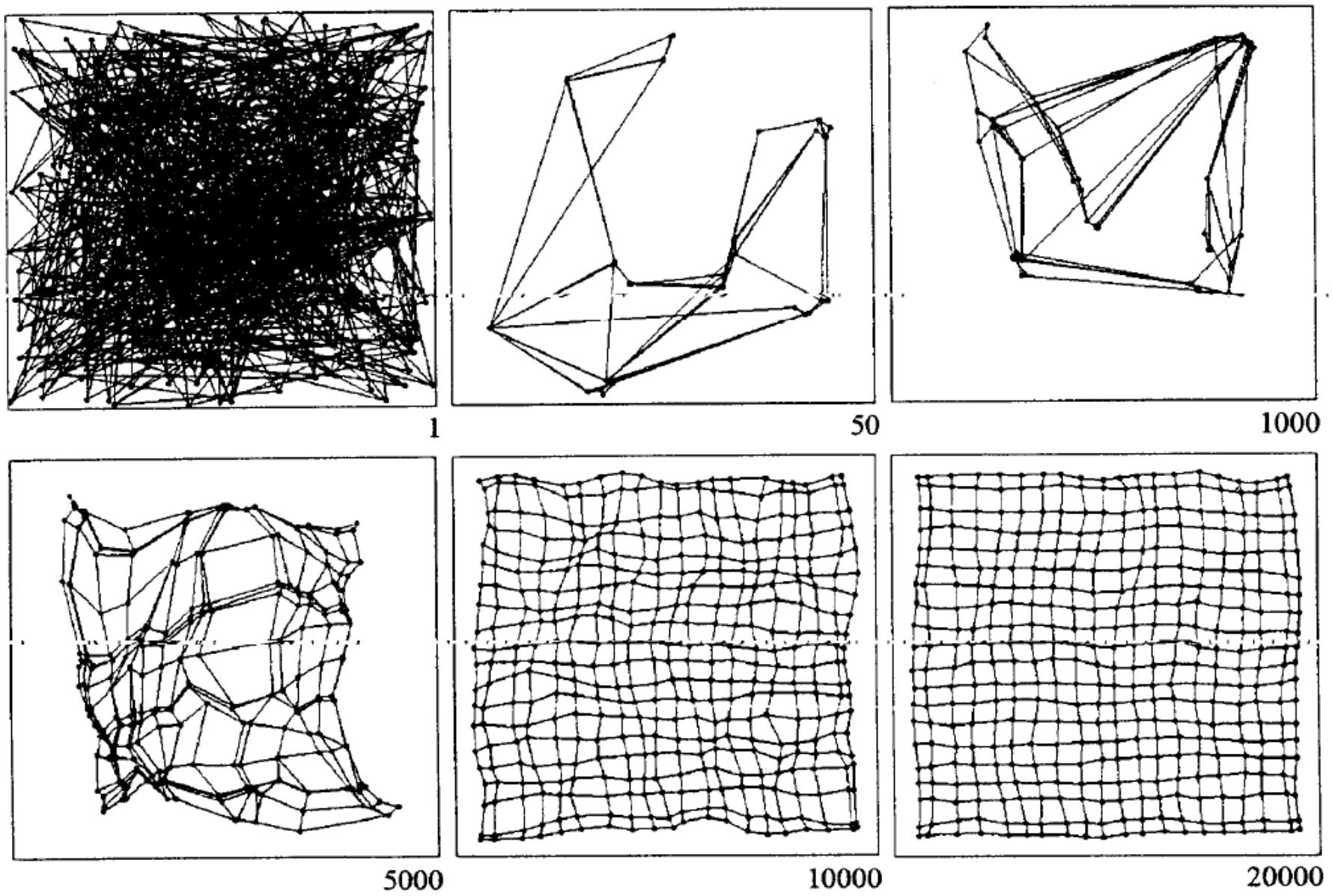
$\mathbf{X} \rightarrow \{1,2,...,m\}$ or

$\mathbf{X} \rightarrow \mathbf{Y}$ $\quad \boldsymbol{y} = [y_1, y_2, ..., y_m]$

where e.g. $y_i = \exp(-\|\boldsymbol{x}\text{-}\boldsymbol{w}_i\|)$



Example: 2D inputs, 20x20 neurons

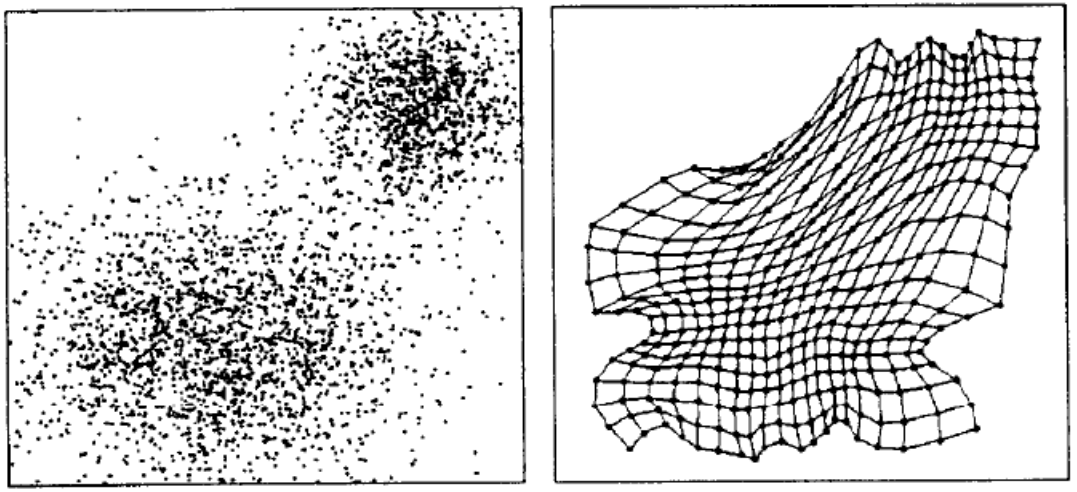


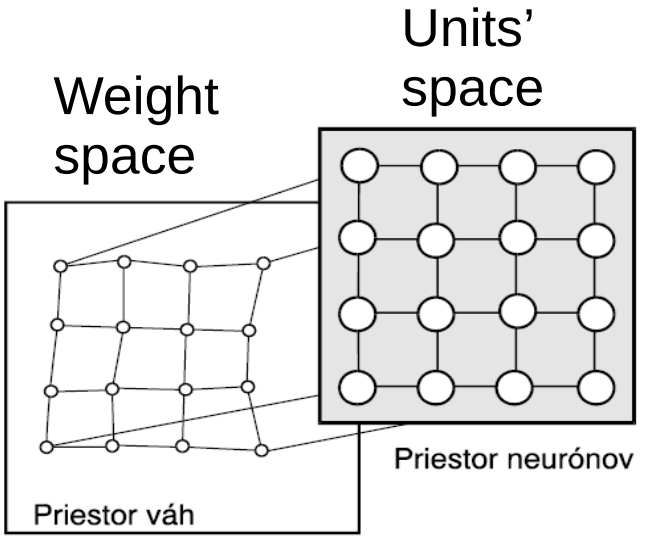
Weight vectors (knots) become topographically ordered at the end of training

Magnification factor

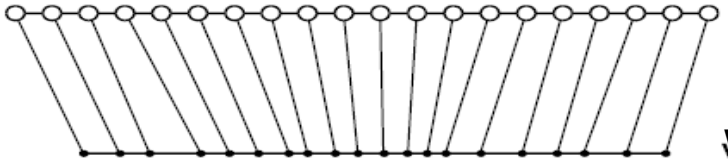
- SOM tends to approximate input distribution

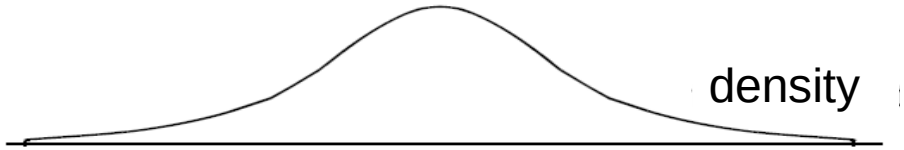
2D:

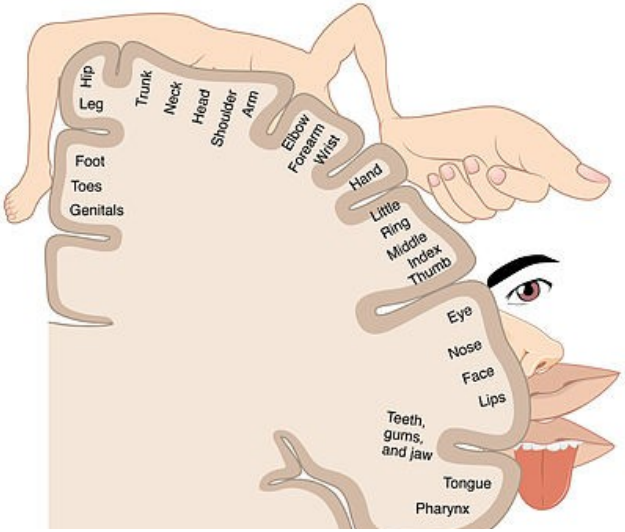




1D:





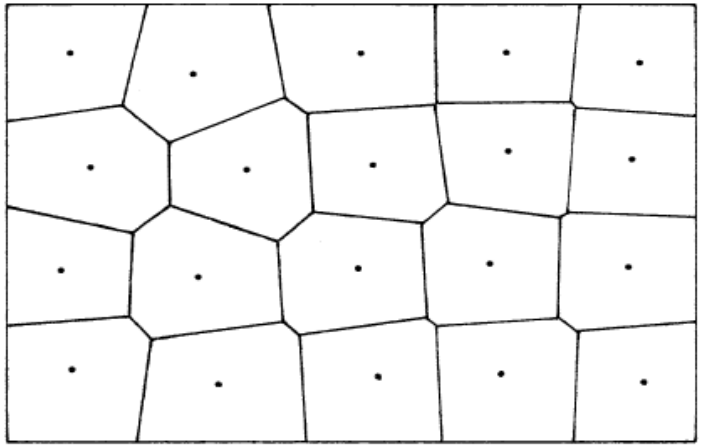


SOM performs 2 tasks simultaneously

Vector quantization

Voronoi compartments:

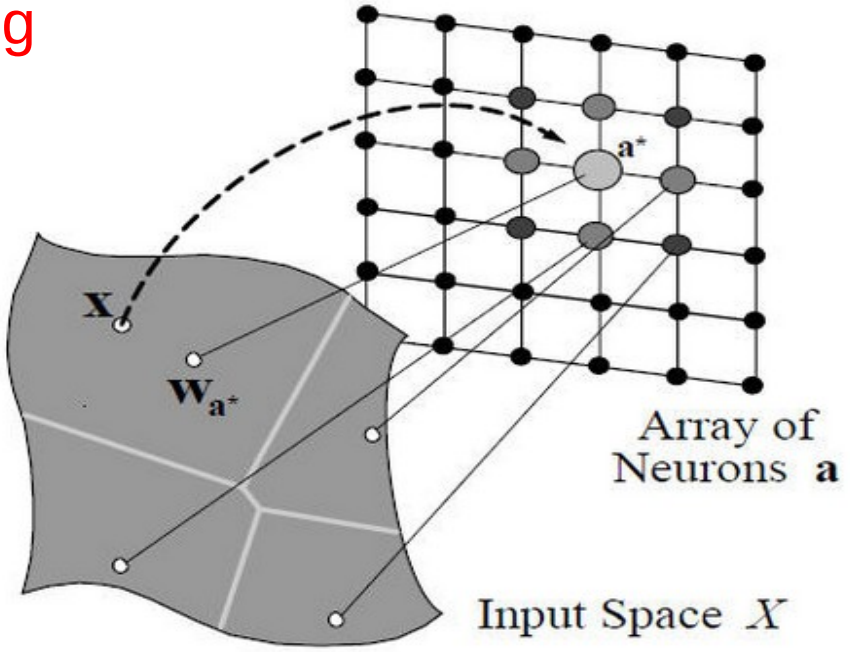
$$V_i = \{x \mid \|x - w_i\| < \|x - w_j\|, \ \forall j \neq i$$



Voronoi tessellation

Topographic mapping

i.e. similar stimuli in input space are mapped close to each other in the SOM



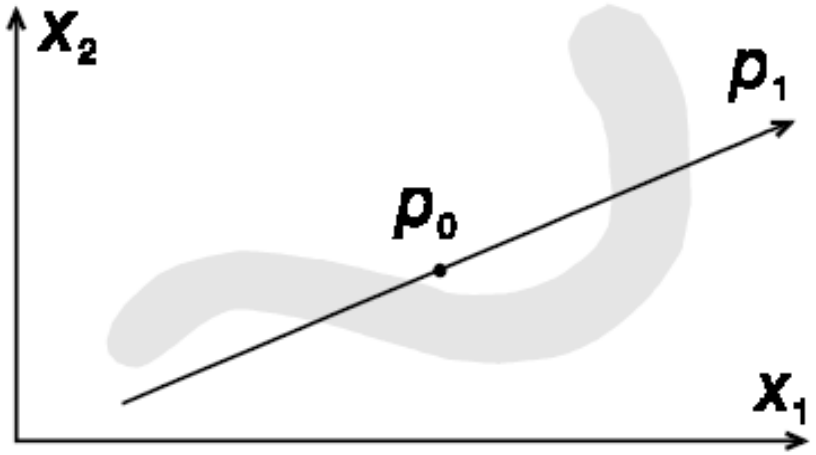
Main properties of SOM

- Approximation of the input space (input data) by the grid of neurons → Vector quantization theory
- Topological ordering (preservation of similarities between input and output spaces)
- Density matching – reflecting the variations in the statistics of input distribution
- Feature selection – via nonlinear mapping → principal curves or surfaces (Hastie and Stuetzle, 1989) →
- SOM as a nonlinear generalization of PCA

Comparison of SOM to PCA

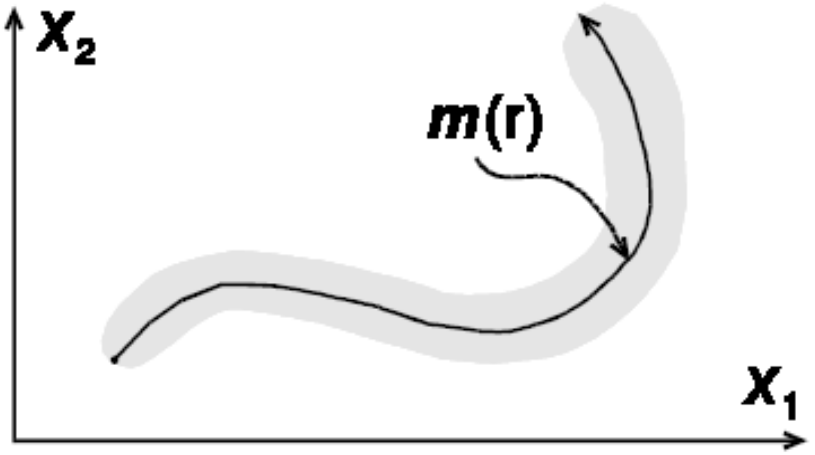
- feature extraction and mapping, difference in feature representation

PCA



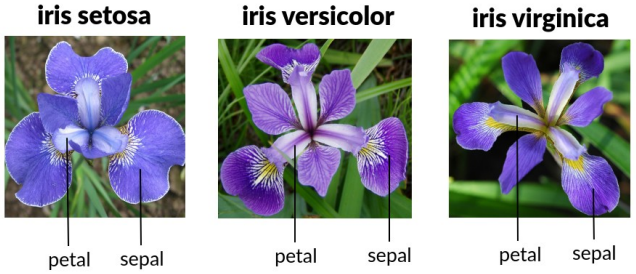
(linear) principal components
One unit represents 1 dimension

SOM

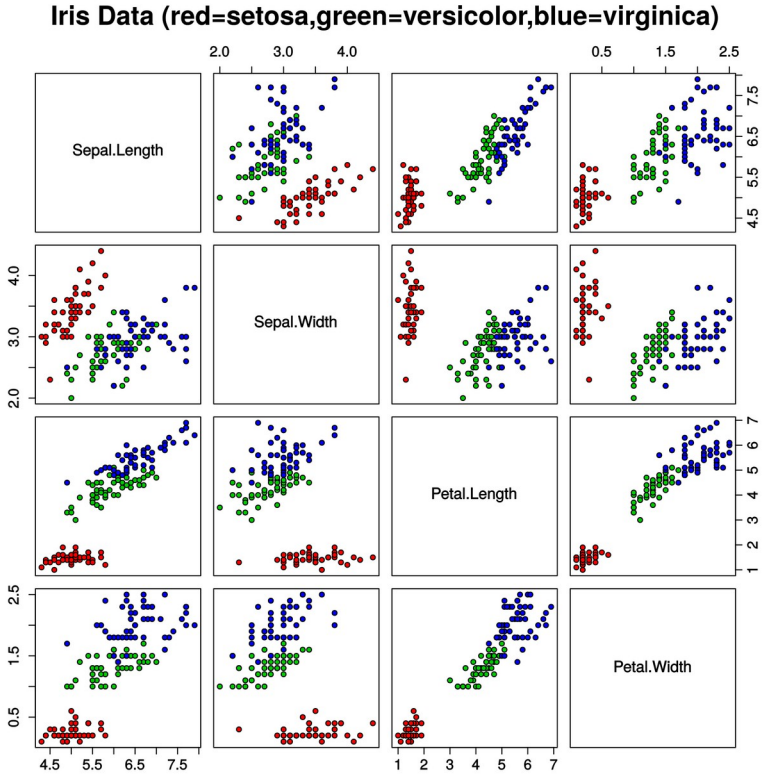


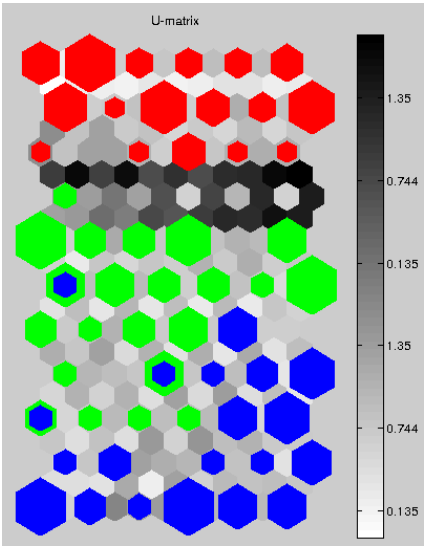
(nonlinear) principal manifold
More units represent 1 dimension

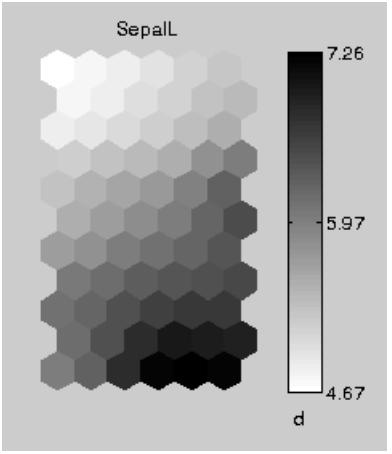
Application: Data visualization

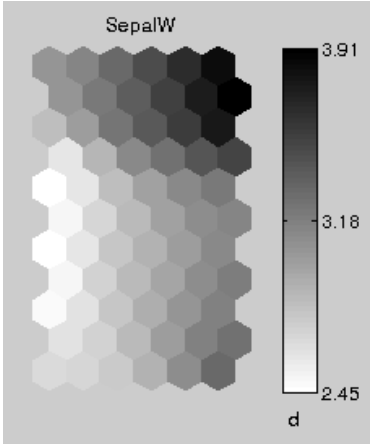


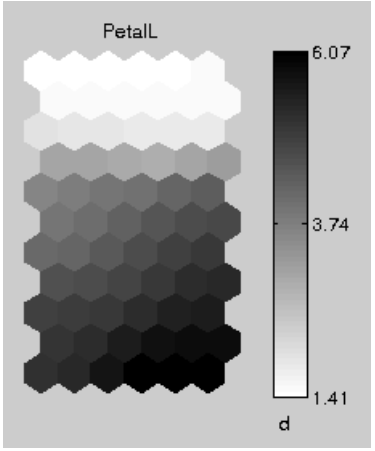
- 7x10 SOM with hexa grid
- trained on 4D Iris data
- 3 classes
- Neuron labels generated
- according to votes
- Plots of component weights reveal an order

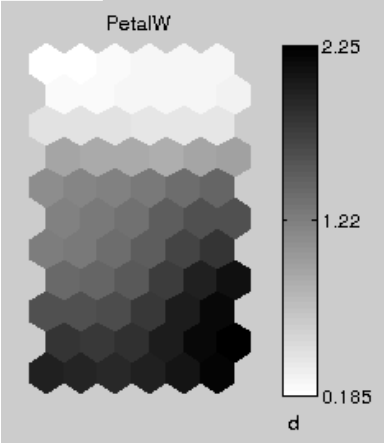






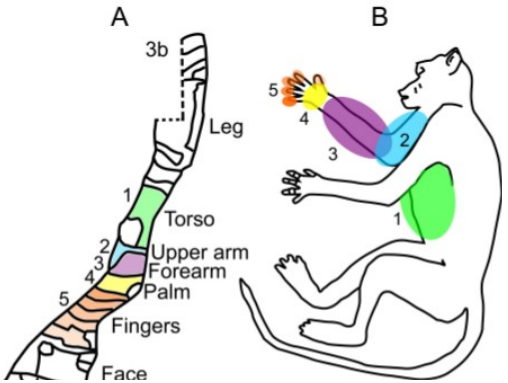


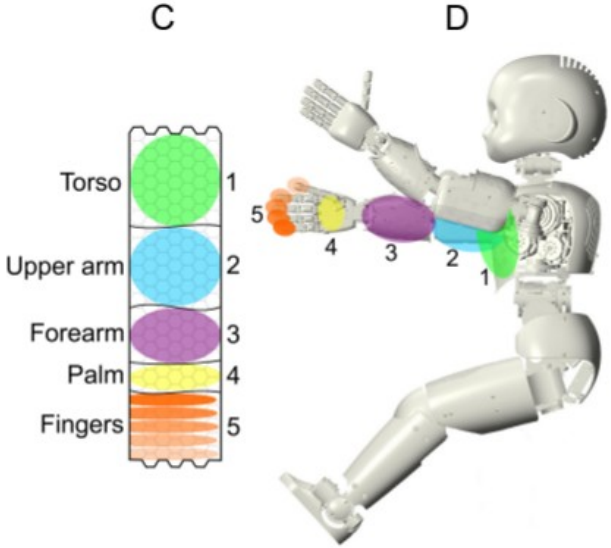


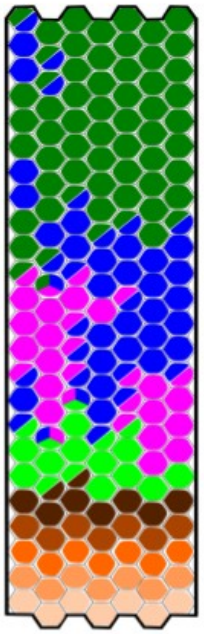


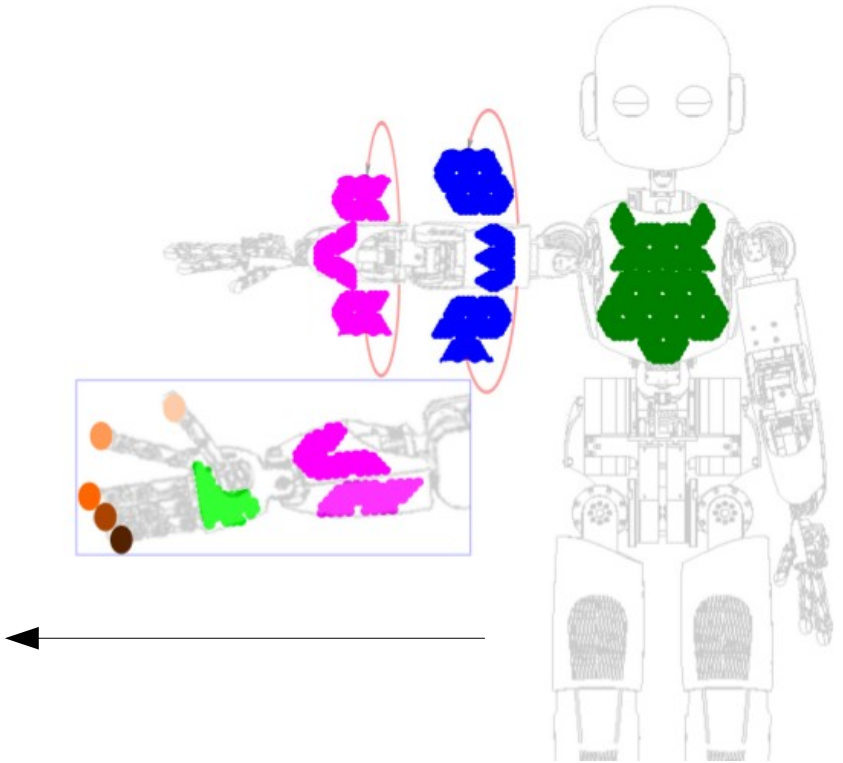
Application: Somatosensory homunculus

Self-organized mapping from the skin to the “brain”









Hoffmann, Straka, Farkaš, Vavrečka, Metta: Robotic homunculus: Learning of artificial skin representation in a humanoid robot motivated by primary somatosensory cortex. *IEEE Trans. on Cog. and Devel. Systems,* 2017.

SOM summary

- self-organizing map – a very popular algorithm
  - principles of competition and cooperation, unsupervised learning
  - performs vector quantization and topology-preserving mapping
- useful for data clustering and visualization
- theoretical analysis of SOM limited to simple cases
- various self-organizing algorithms developed
  - main purpose: data clustering
  - not all unsupervised algorithms implement dimensionality-reducing mapping (and hence visualization)