Machine Learning and Pervasive Computing

Stephan Sigg

Georg-August-University Goettingen, Computer Networks

13.07.2015

Overview and Structure

Outline

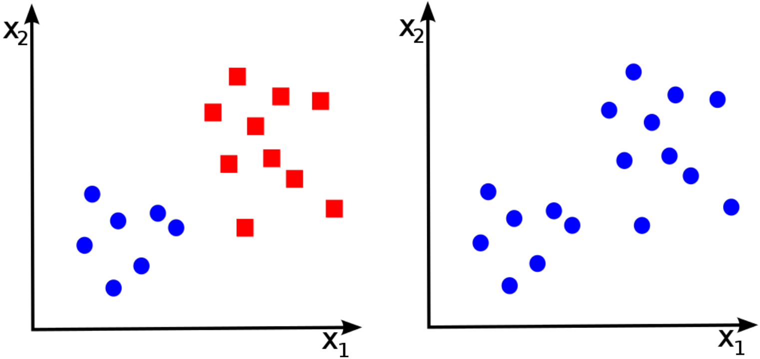
Unsupervised learning

Self Organizing Maps

- Introduction
- Definition
- Example

mvote.ugoe.de/282A

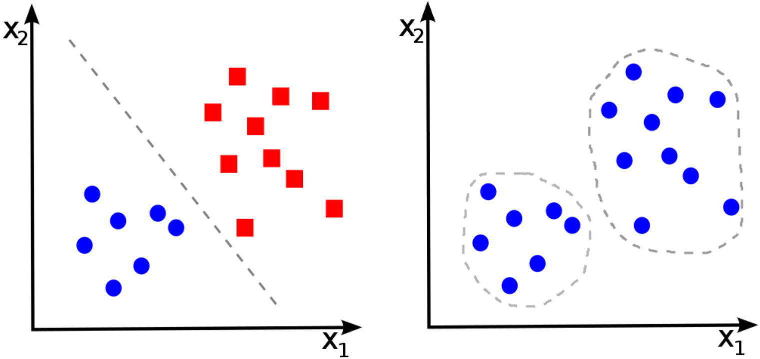
Unsupervised learning



Supervised:
$\{(x_{1,1}, x_{1,2}) \rightarrow y_1, (x_{2,1}, x_{2,2}) \rightarrow y_2, \ldots, (x_{n,1}, x_{n,2}) \rightarrow y_n\}$
Unsupervised: $\{(x_{1,1}, x_{1,2}), (x_{2,1}, x_{2,2}), \ldots, (x_{n,1}, x_{n,2})\}$

mvote.ugoe.de/282A

Unsupervised learning



Supervised:

$\{(x_{1,1}, x_{1,2}) \rightarrow y_1, (x_{2,1}, x_{2,2}) \rightarrow y_2, \ldots, (x_{n,1}, x_{n,2}) \rightarrow y_n\}$

Unsupervised: $\{(x_{1,1}, x_{1,2}), (x_{2,1}, x_{2,2}), \ldots, (x_{n,1}, x_{n,2})\}$

mvote.ugoe.de/282A

Unsupervised learning

k-means algorithm

Iteratively find k clusters in the data

Init Randomly choose k points as initial cluster centroids

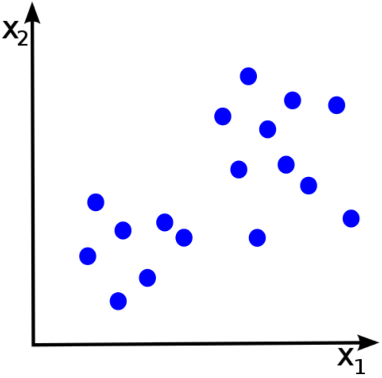
Repeat :

- → Assign samples to these cluster centroids conditioned on distance
- → Move cluster centroids to the center weight of the points associated to them

mvote.ugoe.de/282A

Unsupervised learning

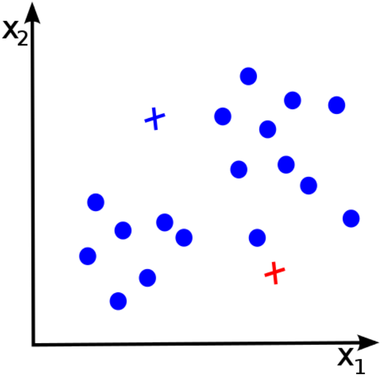
k-means algorithm



Unsupervised learning

k-means algorithm

Init: k cluster centroids μ_i chosen randomly



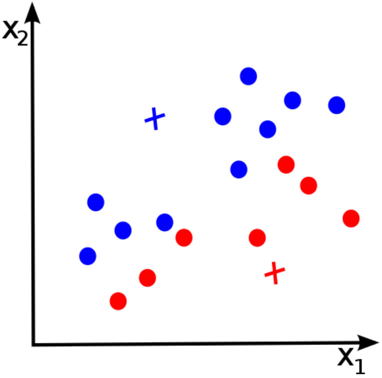
Unsupervised learning

k-means algorithm

Init: k cluster centroids μ_i chosen randomly

Repeat:

1: assign samples to centroids conditioned on distance



Unsupervised learning

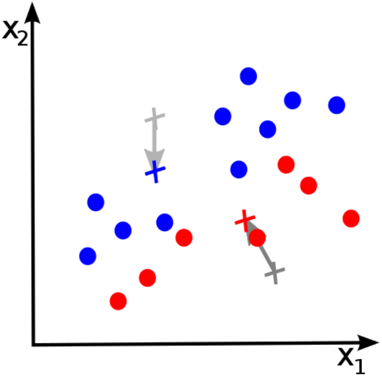
k-means algorithm

Init: k cluster centroids μ_i chosen randomly

Repeat:

1: assign samples to centroids conditioned on distance

2: $\mu_j(t+1) = \frac{1}{C_j} \sum_{i=1}^{C_j} x^{(i)}$



Unsupervised learning

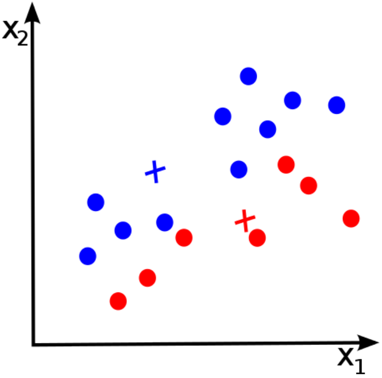
k-means algorithm

Init: k cluster centroids μ_i chosen randomly

Repeat:

1: assign samples to centroids conditioned on distance

2: $\mu_j(t+1) = \frac{1}{C_j} \sum_{i=1}^{C_j} x^{(i)}$



mvote.ugoe.de/282A

Unsupervised learning

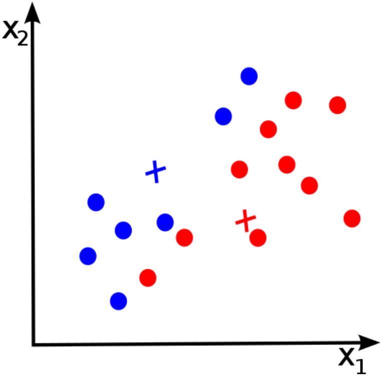
k-means algorithm

Init: k cluster centroids μ_i chosen randomly

Repeat:

1: assign samples to centroids conditioned on distance

2: $\mu_j(t+1) = \frac{1}{C_j}\sum_{i=1}^{C_j} x^{(i)}$



mvote.ugoe.de/282A

Unsupervised learning

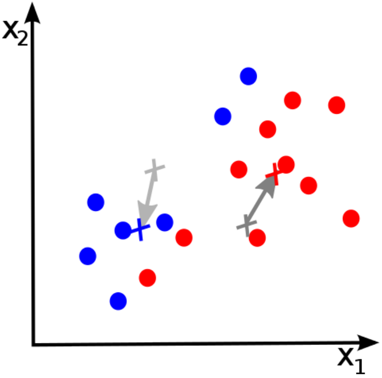
k-means algorithm

Init: k cluster centroids μ_i chosen randomly

Repeat:

1: assign samples to centroids conditioned on distance

2: $\mu_j(t+1) = \frac{1}{C_j}\sum_{i=1}^{C_j} x^{(i)}$



mvote.ugoe.de/282A

Unsupervised learning

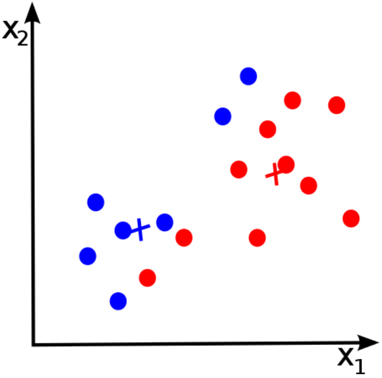
k-means algorithm

Init: k cluster centroids μ_i chosen randomly

Repeat:

1: assign samples to centroids conditioned on distance

2: $\mu_j(t+1) = \frac{1}{C_j} \sum_{i=1}^{C_j} x^{(i)}$



Unsupervised learning

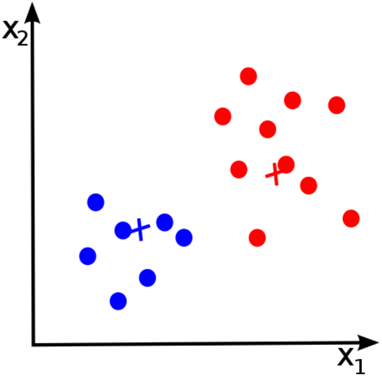
k-means algorithm

Init: k cluster centroids μ_i chosen randomly

Repeat:

1: assign samples to centroids conditioned on distance

2: $\mu_j(t+1) = \frac{1}{C_j} \sum_{i=1}^{C_j} x^{(i)}$



mvote.ugoe.de/282A

Unsupervised learning

k-means algorithm

How to randomly initialise the k-means algorithm

The k-means algorithms may compute different solutions for different initial choice of cluster centroids

With respect to the overall distance of the samples to their cluster centroids, k-means might run into local optima

Unsupervised learning

k-means algorithm

How to randomly initialise the k-means algorithm

The k-means algorithms may compute different solutions for different initial choice of cluster centroids

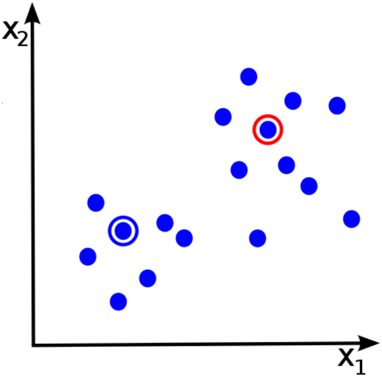
With respect to the overall distance of the samples to their cluster centroids, k-means might run into local optima

Common choice of the initial k cluster centroids

Choose the initial k cluster centroids randomly from the set of training samples

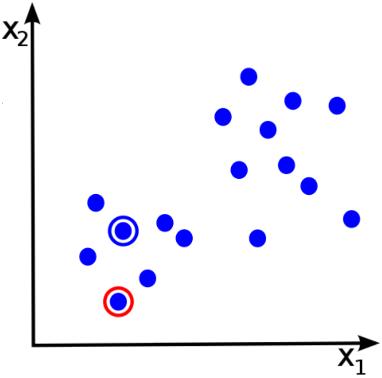
Unsupervised learning

k-means algorithm



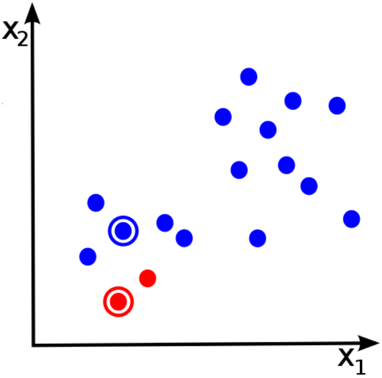
Unsupervised learning

k-means algorithm



Unsupervised learning

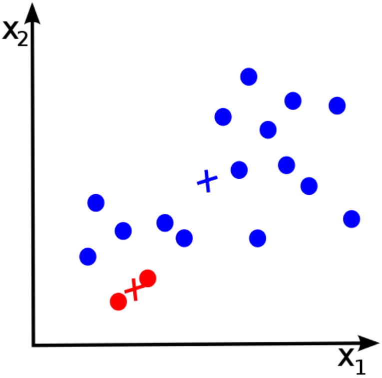
k-means algorithm



mvote.ugoe.de/282A

Unsupervised learning

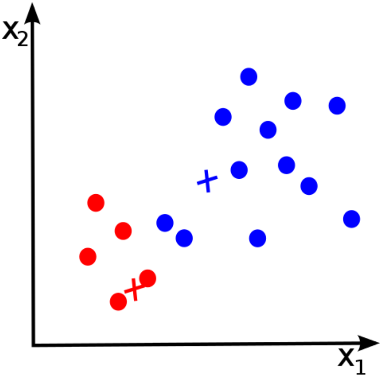
k-means algorithm



mvote.ugoe.de/282A

Unsupervised learning

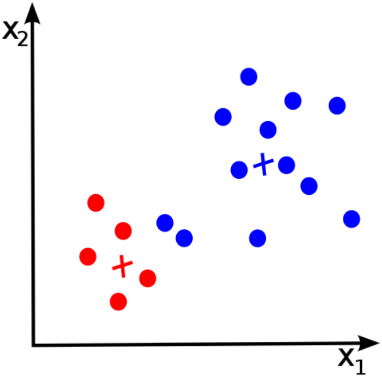
k-means algorithm



mvote.ugoe.de/282A

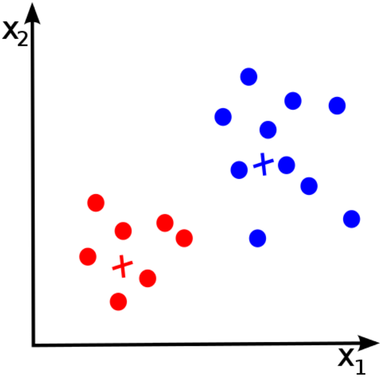
Unsupervised learning

k-means algorithm



Unsupervised learning

k-means algorithm



mvote.ugoe.de/282B

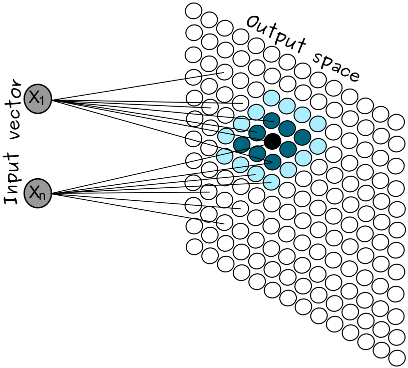
Outline

Introduction to self organising maps (SOM)

Proposed by Teuvo Kohonen[1]

→ Model of the self-organisation of neural connections

→ Maps high dimensional input to low dimensional (e.g. 2D) output



[1] Teuvo Kohonen, *Self-Organizing Maps*, Springer, 2001.

mvote.ugoe.de/282B

Introduction to self organising maps (SOM)

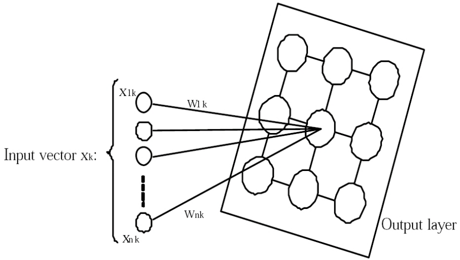
Relation to Neural Networks

Similarity

- Weighted inputs mapped to vector of outputs

Difference

- Considers neighbourhood relation and ordering of output layer
- Unsupervised
- Alternative learning and updating



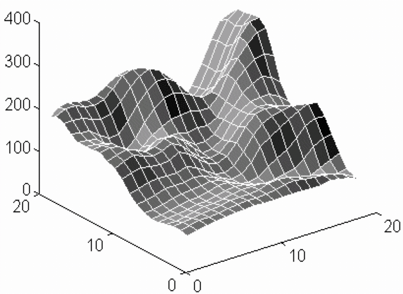
mvote.ugoe.de/282B

Introduction to self organising maps

Represent all points in a source space by points in a target space

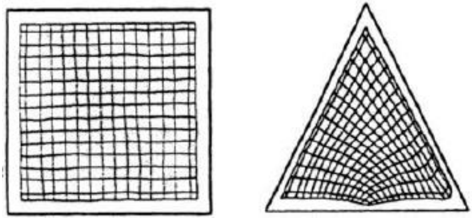
Given a sequence of points in a sample space,

Create a mapping of these points into a target space that respects the neighbourhood relation in the sample space



mvote.ugoe.de/282B

Introduction to self organising maps



SOM is a topology preserving lattice of predefined number of nodes

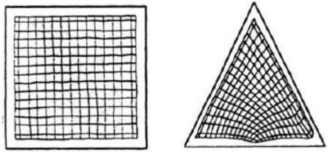
Represents topology of elements in input space.

Algorithm inherits self-organisation property

- Able to produce organisation starting from total disorder.
- Defines and preserves neighbourhood structure between nodes

Learning by two layer neural network

Introduction to self organising maps



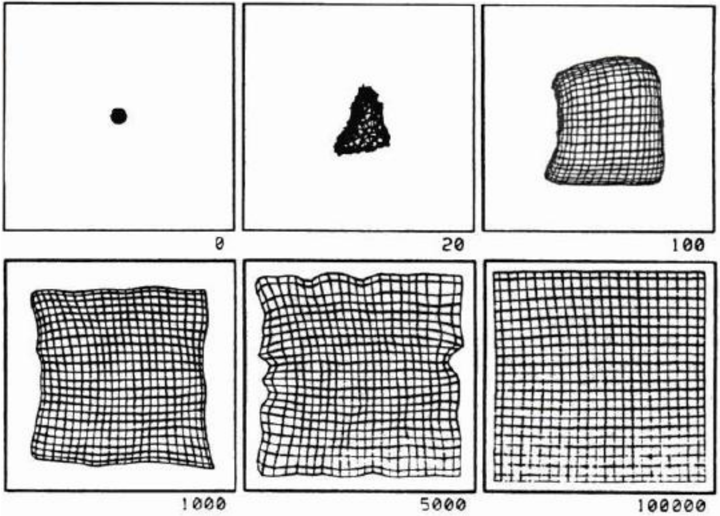
When a pattern $\overrightarrow{\phi_i}$ is presented, each node (represented by outer neurons) in the target space computes its activation $\overrightarrow{\phi}_i^t \overrightarrow{w}$.

Most activated node y^* and weights to its neighbours are updated according to a learning rate $\rho(t)$

$$w_{ki}(t+1) = w_{ki}(t) + \rho(t)\Lambda(|y - y^*|)(\overrightarrow{\phi_i} - w_{ki}(t))$$

$\Lambda(\cdot)$ defines a non-increasing neighbourhood function and $|y - y^*|$ describes the distance of nodes in the neighbourhood

SOM – Self organisation



mvote.ugoe.de/282B

SOM – Definition

Condensed definition of SOM from Cottrell et al.[2]

Self organising maps

- Let $I = \{\overrightarrow{\eta_1}, \ldots, \overrightarrow{\eta_{|S|}}\}$ be a set of km-dimensional vectors that are associated with nodes in a lattice.
- Neighbourhood structure provided by symmetrical neighbourhood function $d : I \times I \to \mathbb{R}$ which depends on the distance between two nodes $\overrightarrow{\eta_i}$ and $\overrightarrow{\eta_j} \in I$.
- State of the map at time t given by

$$\eta(t) = \left(\overrightarrow{\eta_1(t)}, \overrightarrow{\eta_2(t)}, \ldots, \overrightarrow{\eta_{|S|}(t)}\right),$$

[2] M. Cottrell, J.C. Fort and G. Pages, *Theoretical aspects of the SOM algorithm*, Neurocomputing, pp. 119-138, vol 21, 1998.

mvote.ugoe.de/282B

SOM – Definition

Self organising map algorithm

The SOM algorithm is recursively defined by

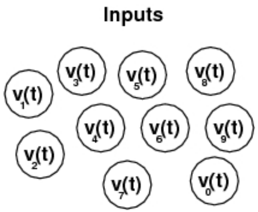
$$\begin{aligned} i_c\left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)}\right) &= argmin\left\{\left\|\overrightarrow{v(t+1)} - \overrightarrow{\eta_i(t)}\right\|, \overrightarrow{\eta_i(t)} \in \eta(t)\right\}, \\ \overrightarrow{\eta_i(t+1)} &= \overrightarrow{\eta_i(t)} - \varepsilon_t d\left[i_c\left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)}\right), \overrightarrow{\eta_i}\right] \\ &\cdot \left(\overrightarrow{\eta_i(t)} - \overrightarrow{v(t+1)}\right), \forall \overrightarrow{\eta_i} \in I. \end{aligned}$$

In this formula, $i_c\left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)}\right)$ corresponds to the node in the network that is closest to the input vector.

Parameter ε_t controls the adaptability.

SOM – Operational principle

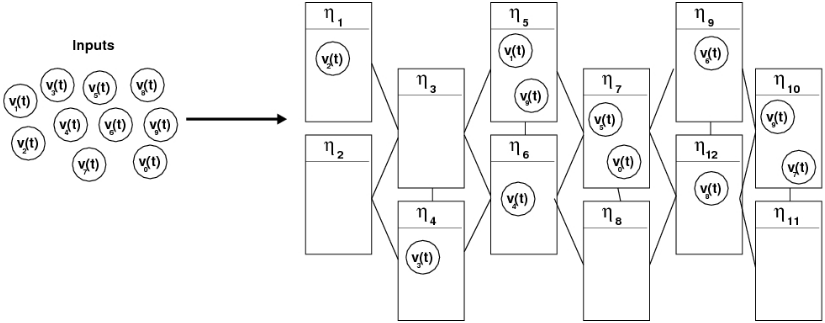
mvote.ugoe.de/282B



Input values $v_i(t)$ are to be mapped onto the target space

mvote.ugoe.de/282B

SOM – Operational principle

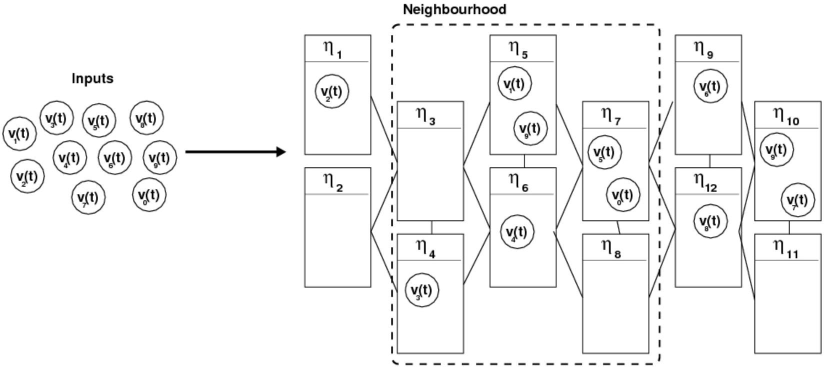


Node with the lowest distance is associated with the input value:

$$i_c\left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)}\right) = argmin\left\{\left\|\overrightarrow{v(t+1)} - \overrightarrow{\eta_i(t)}\right\|, \overrightarrow{\eta_i(t)} \in \eta(t)\right\}$$

mvote.ugoe.de/282B

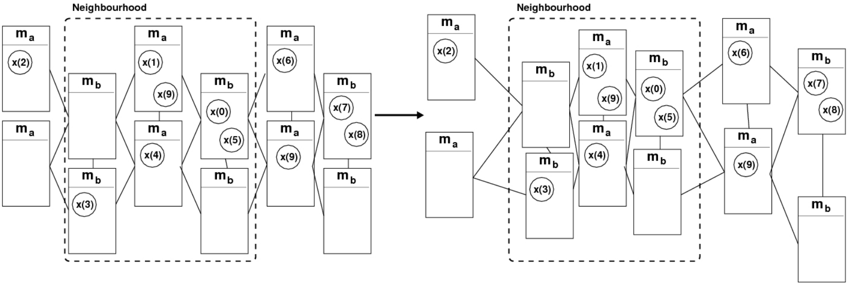
SOM – Operational principle



Nodes in the neighbourhood of the associated node are moved closer to the input value

mvote.ugoe.de/282B

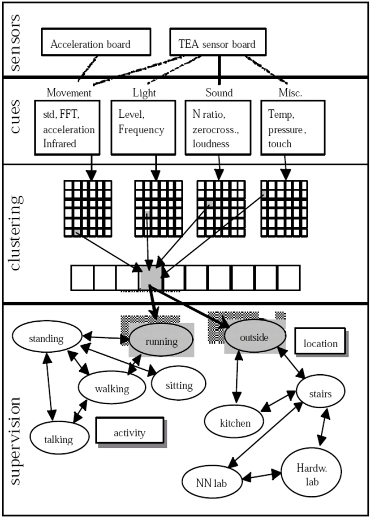
SOM – Operational principle



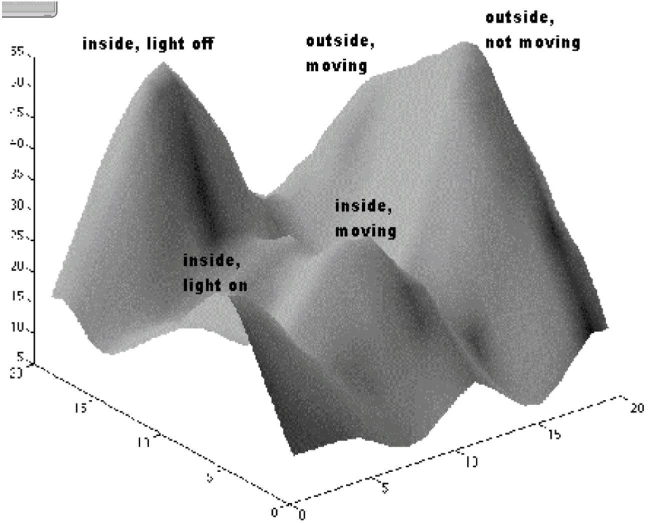
Nodes in the neighbourhood of the associated node are moved to the input value

$$\overrightarrow{\eta_i(t+1)} = \overrightarrow{\eta_i(t)} - \varepsilon_t d \left[i_c \left(\overrightarrow{v(t+1)}, \overrightarrow{\eta(t)} \right), \overrightarrow{\eta_i} \right] \cdot \left(\overrightarrow{\eta_i(t)} - \overrightarrow{v(t+1)} \right), \forall \overrightarrow{\eta_i} \in I.$$

SOM – Example application: TEA



SOM – Example application: TEA



mvote.ugoe.de/282B

SOM – Remarks

SOM algorithm always converges[3]

Normalisation of input vectors might improve numerical accuracy

Not guaranteed that self-optimisation will always occur
(Dependent on choice of parameters)

Difficult to set parameters of the model since SOM is not optimising any well-defined function[4]

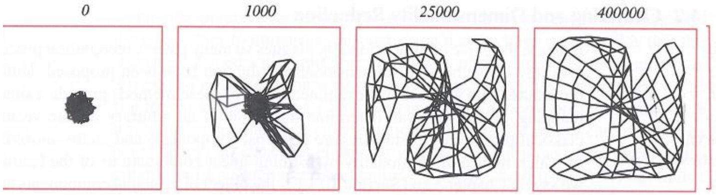
If neighbourhood is chosen to be too small, the map will not be ordered globally

[3] Y. Cheng, *Neural Computation*, 9(8), 1997.

[4] E. Erwin, K. Obermayer, K. Schulten: Self-organising maps: Ordering, convergence properties and energy functions. Biological Cybernetics, 67, 47-55, 1992

mvote.ugoe.de/282B

Problems of SOMs



Map created as target space might have several orientations

One part of the map might follow one orientation, while other parts are following other orientations

Questions?

Stephan Sigg
stephan.sigg@cs.uni-goettingen.de

Literature

- C.M. Bishop: Pattern recognition and machine learning, Springer, 2007.
- R.O. Duda, P.E. Hart, D.G. Stork: Pattern Classification, Wiley, 2001.