Data-driven Intelligent Systems

Lecture 20
Self-organizing Networks

KNOWLEDGE
TECHNOLOGY

http://www.informatik.uni-hamburg.de/WTM/

Self-organizing Maps – Overview

- Topological Maps
- SOM Cost Function and Learning Algorithm
- Visualizing SOMs
- SOM Applications
- Growing Neural Gas & Growing When Required networks

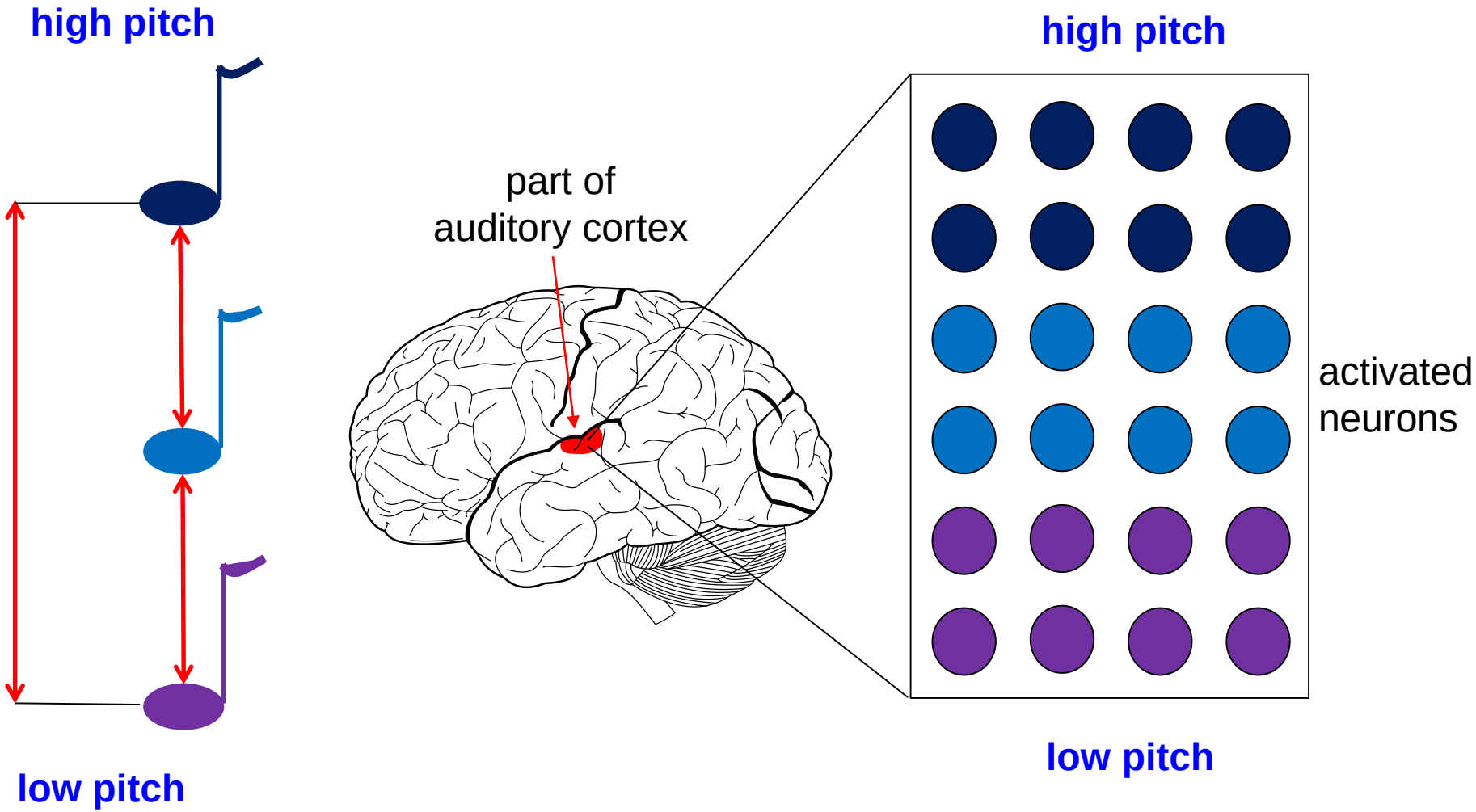
Model-Based Clustering

- What is model-based clustering?
  - Assumption: a cluster is generated by a model such as a probability distribution
  - A model (e.g., Gaussian distribution) is determined by a set of parameters
  - Cluster assignments might be soft, to represent uncertainties
  - Task: optimize the fit between the given data and some mathematical model by learning the parameters of the model
- Typical statistical methods
  - Gaussian Mixture Models
  - EM (Expectation maximization)

Neural Network Approaches

- Neural network approaches
  - Represent each cluster with an exemplar (a neuron), acting as a "prototype" of the cluster
  - New objects are assigned to the cluster whose exemplar is the most similar according to some distance measure
  - Neuron interactions can represent additional features of the data, such as topology
- Typical methods
  - SOM (Self-Organizing feature Map)
  - Competitive learning
    - Neurons compete in a "winner-takes-all" fashion for the currently presented object

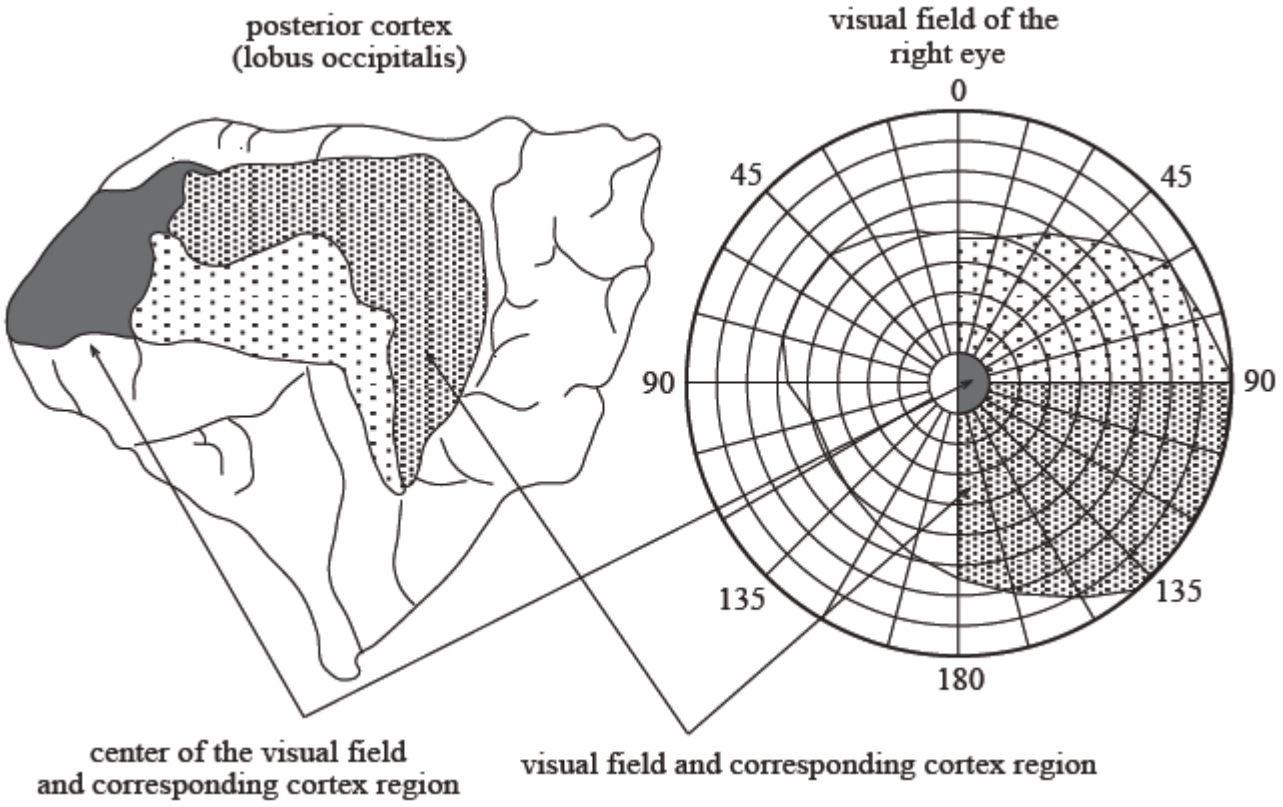
Feature Maps: Topological Mapping of Sound



tonotopy

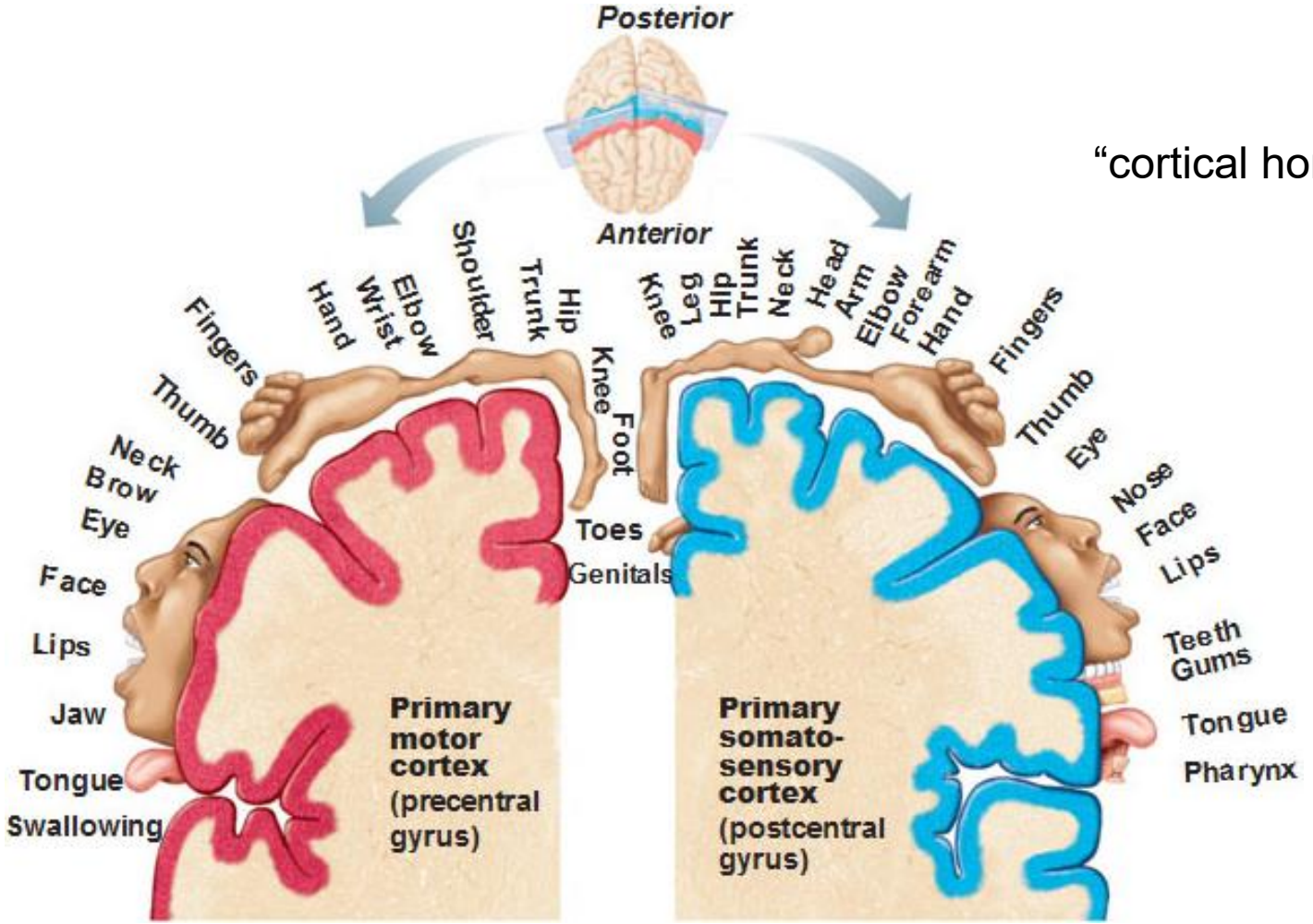
Feature Maps: Topological Mapping in Vision

- Nearby visual stimuli processed nearby in lower visual cortex



retinotopy

Localized Mapping of Sensors and Effectors



"cortical homunculus"

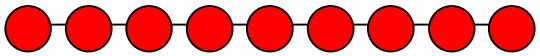
somatotopy

Feature Maps

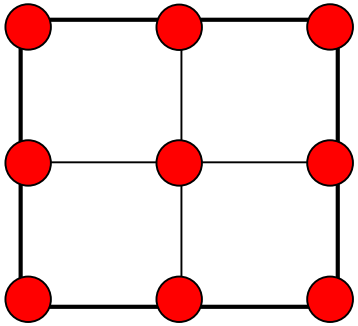
- The mapping preserves the ordering of the inputs:
  - Sensory stimuli that are similar (‘nearby in input space’) excite neurons that are near to each other (in `output space’)
  - Sensory stimuli that are very different excite neurons that are far from each other
- This map property is known as ***topology preservation***
  - “tonotopy” in case of sound
  - “visuotopy” for vision
  - “somatotopy” for touch
- Another example with topology
  - Convolutional neural networks, on their lower layers
- Not all neural projections or representations are topological!

Topology Preservation in Non-matching Dimensions

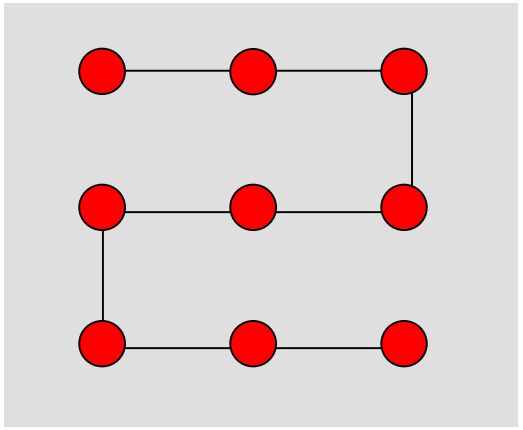
1D network topology

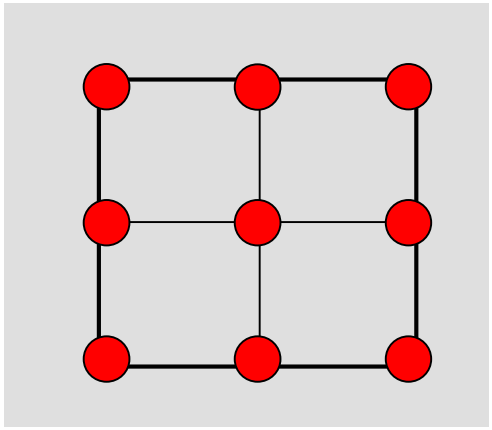


2D network topology

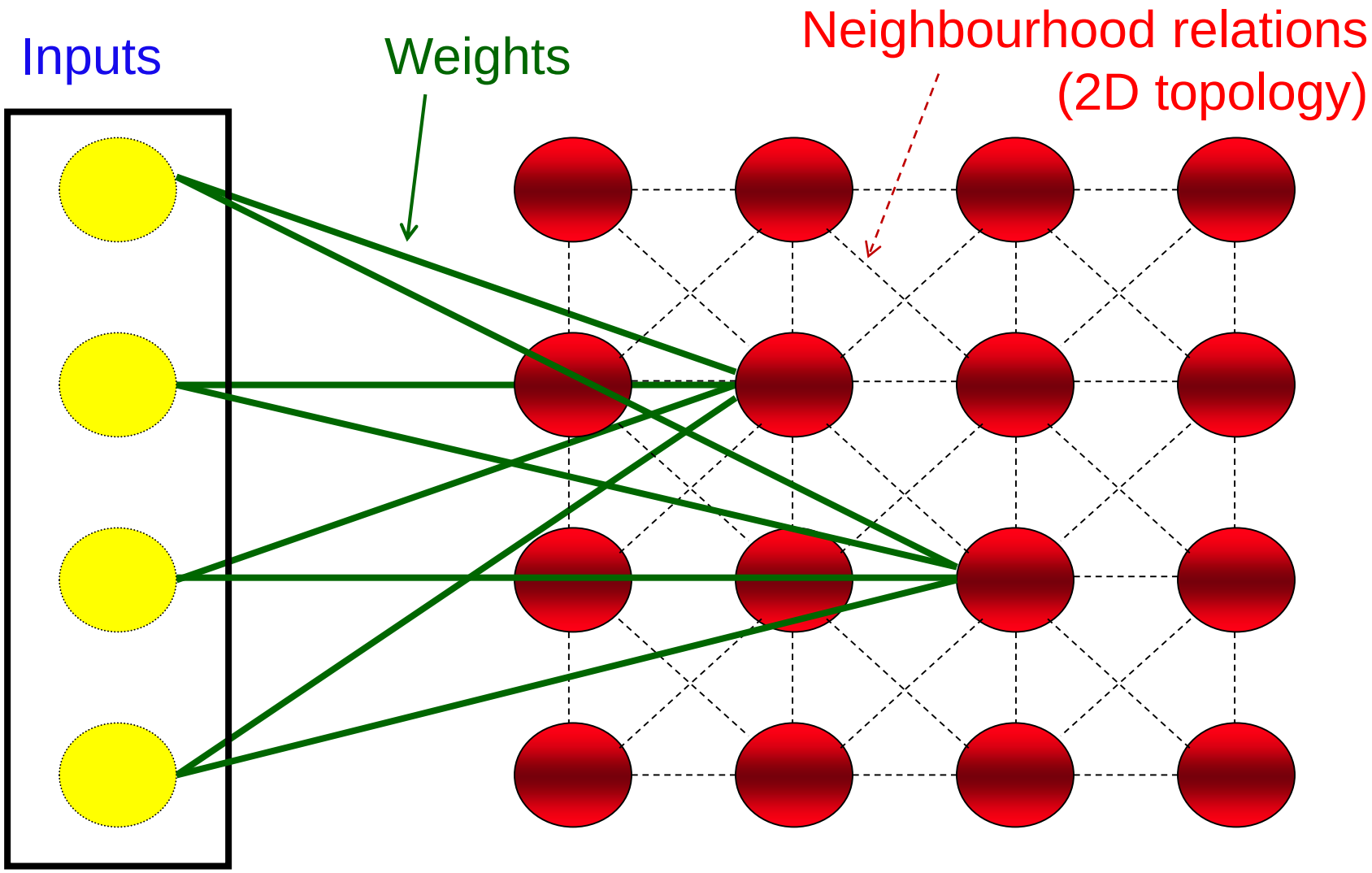


Network representation of 2D input space





The Self-Organising Map



Competitive Neuron Interactions in one Layer

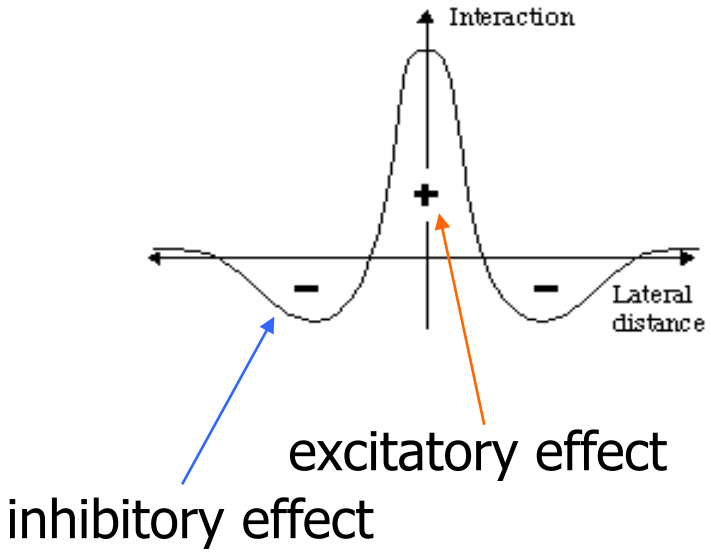
The ***Mexican hat function***,
or *Difference of Gaussians,*
represents the strength of
interactions *within* the neural layer
(while not modelling actual connections)

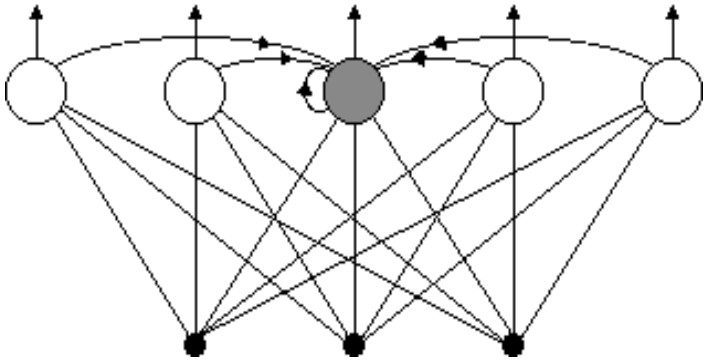
- ***Remote*** neighborhood –
  negative, ***inhibitory*** effect

  → "***winner-take-all***" behavior
  (WTA, competitive network!)

- ***Near*** neighborhood – short
  range lateral ***excitation*** area has
  strong positive effect

  → **topology preservation**





Neuron Connections?

- We can implement WTA behavior directly

  → we do not actually need the inhibitory connections

  → just use a neighborhood of positive interactions: $h(i,j)$

  neurons i,j

  neighbourhood strength

- How large should this neighborhood be?
  - At the beginning of learning, the network is unordered
    - Large neighborhood: each input vector excites many neurons

      → Nearby neurons will learn similarly

      → Their weights will be ordered in input space
  - Later on, fine-tuning
    - Small neighborhood: an input excites only neurons closeby

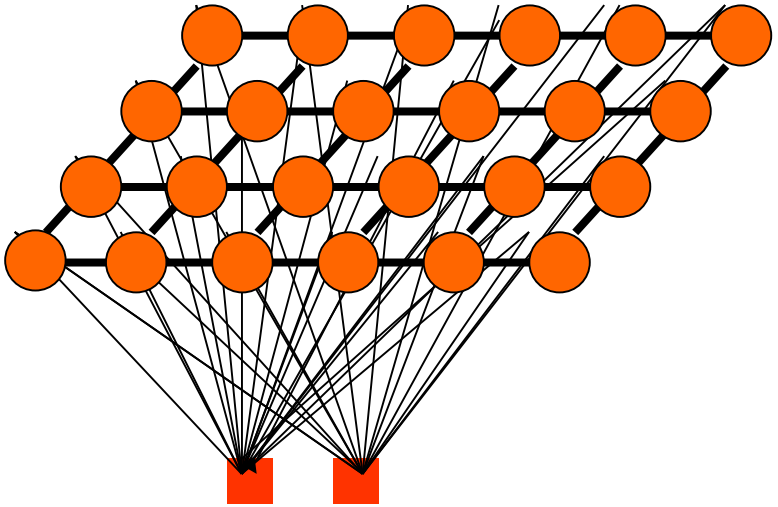
      → Neurons specialize on small input regions

Self-Organizing Map (SOM)

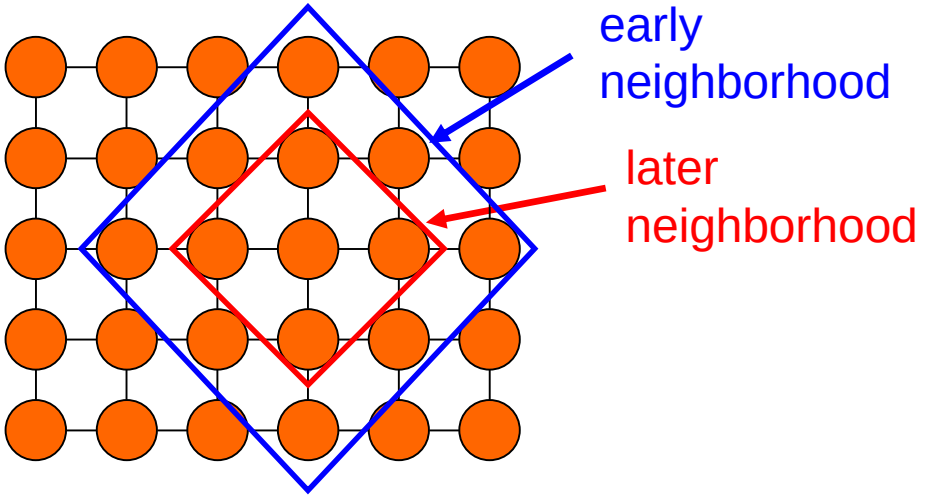
- Also called Kohonen Self-Organizing Feature Map
- It maps the points in a high-dimensional source space into a 1D or 2D target space (rarely, 3D or other)
- distances and proximity relationships (i.e., topology) are preserved as much as possible ("topological ordered map")
- Clustering is performed by having several units competing for the current object
  - The unit whose weight vector is closest to the current object wins
  - The winner and its neighbors learn by having their weights adjusted
- SOMs mimic some aspects of competitive processing in the brain
- Useful for visualizing high-dimensional data

Self organizing maps

- The activation of the neuron is spread in its direct neighborhood
  - neighbors become sensitive to the same input patterns

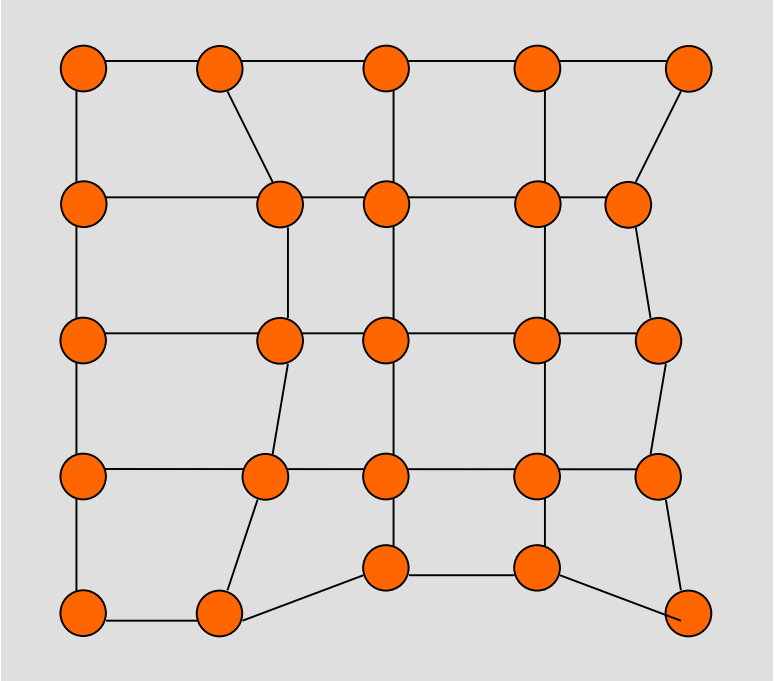


- The size of the neighborhood is initially large but reduced over time during training as the network neurons become more specialized

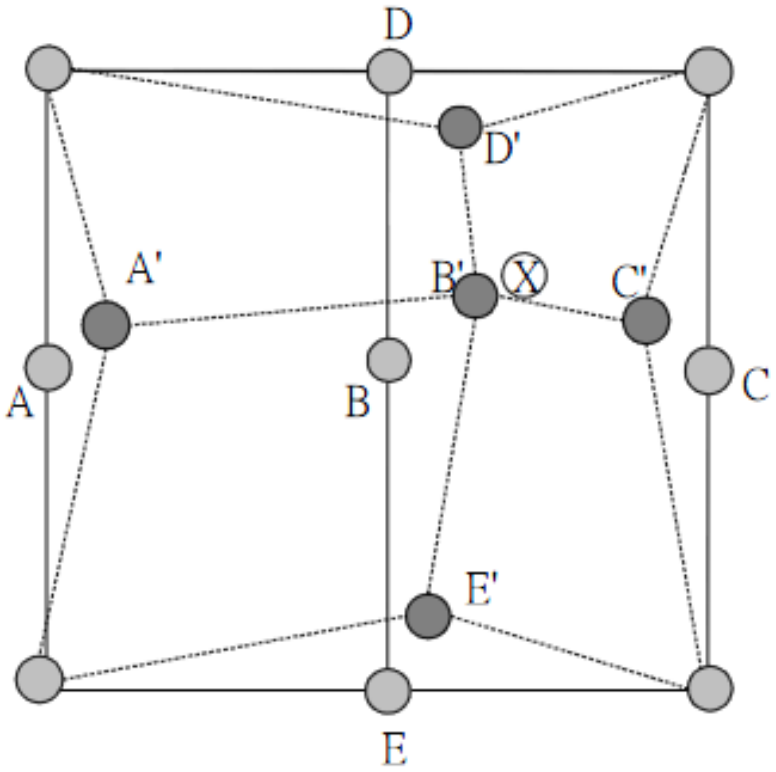


Adaptation

- During training, the “winner” neuron and its neighborhood adapt to make their weight vectors more similar to the input pattern that caused the activation
- The neurons` weight vectors are moved closer to the input pattern
- The magnitude of the adaptation is controlled via a learning parameter, which may decay over time



Influence by Pre-Defined Topology



- The winner B and its neighbors such as A, C, D and E move towards the input vector X
- Modified units are shown as dark circles

Self-organizing Maps – Overview

- Topological Maps
- SOM Cost Function and Learning Algorithm
- Visualizing SOMs
- SOM Applications
- Growing Neural Gas & Growing When Required networks

SOM `Cost Function´

- Recall k-means: $$E = \sum_{i=1}^{k} \sum_{p \in C_i} (x_p - \mu_i)^2$$

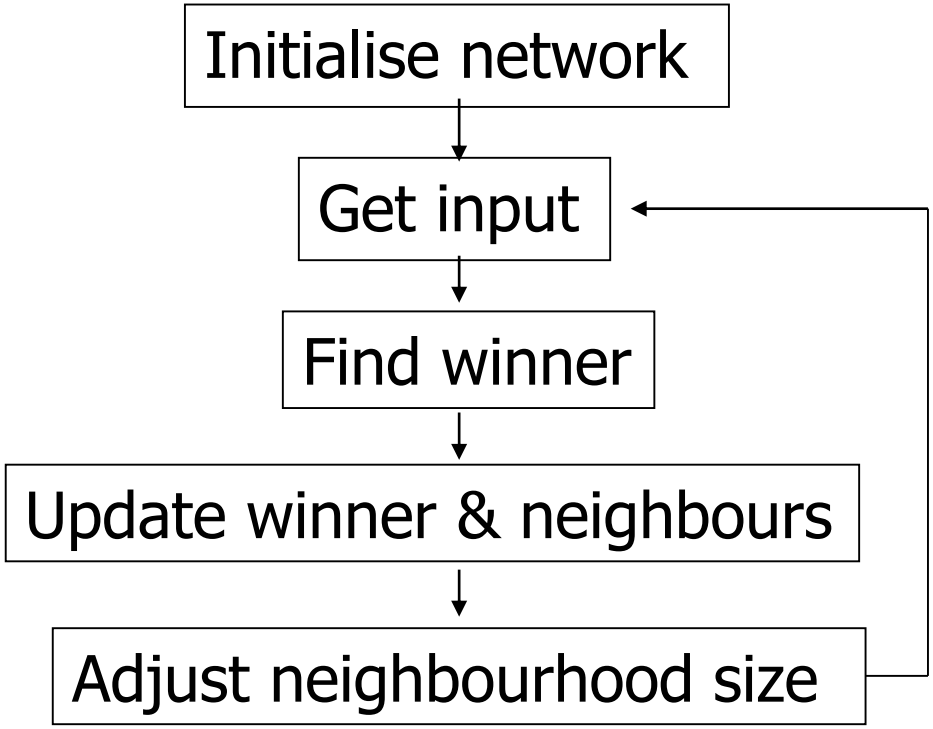
each data point is assigned to ***one*** winning cluster C_i
(cluster = neuron)

- SOM: $$E = \sum_{i=1}^{k} \sum_{p \in C_i} \sum_{j}^{k} h(|i-j|)\,(x_p - \mu_j)^2$$

data point ***assigned to several*** neurons *j*
around winning neuron *i*

neighbourhood activation function *h*
signifies the degree of assignment

SOM Algorithm in Short



The Self-Organising Map Algorithm

- Initialization
  - Determine shape and size of the map (e.g. 2D with neurons *j*)
  - Randomly initialise the weight vectors w_j
- Learning: Repeat …
  - Present input vector x to the network
  - Determine ***best matching neuron*** n_b with the minimal Euclidean distance between input x and weight vector w

$$n_b = \min_j \left\| x - w_j^T \right\|$$

  - The winning node has weight vector moved closer to the input (with learning rate $\eta(t)$)

$$w_j^T \leftarrow w_j^T + \eta(t) \cdot \left(x - w_j^T \right)$$

- … so far, it's k-means (online version) …

The Self-Organising Map Algorithm

- Initialization
  - Determine shape and size of the map (e.g. 2D with neurons *j*)
  - Randomly initialise the weight vectors w_j
- Learning: Repeat …
  - Present input vector x to the network
  - Determine ***best matching neuron*** n_b with the minimal Euclidean distance between input x and weight vector w

$$n_b = \min_j \left\| x - w_j^T \right\|$$

  - The winning node and neighbours have weight vector moved closer to the input (with learning rate $\eta(t)$)

$$w_j^T \leftarrow w_j^T + \eta(t) \cdot h(n_b, t) \cdot \left(x - w_j^T \right)$$

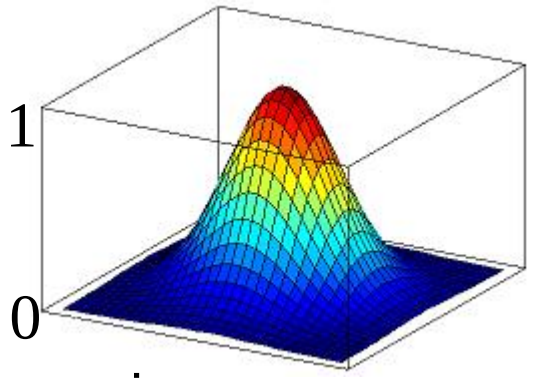
- Over time, the network *self-organises* so that the input topology is preserved

Neighborhood Function Preserves Topology

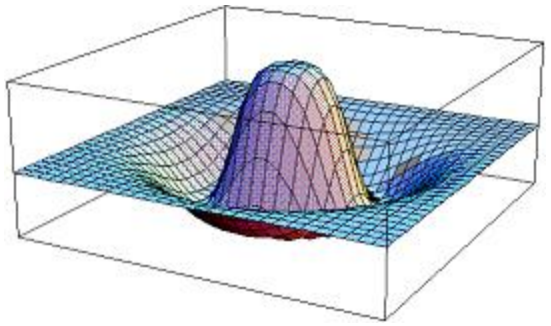
- The neighborhood function $h(n_b,t)$ determines the degree of weight vector change of the neighbors

$$w_j^T \leftarrow w_j^T + \eta(t) \cdot h(n_b,t) \cdot \left(x - w_j^T\right)$$

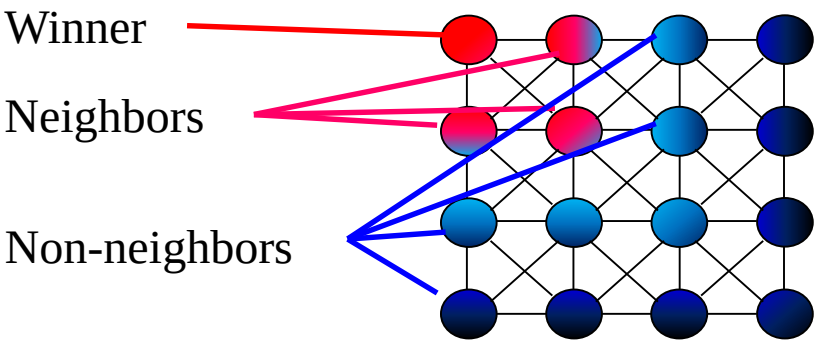
  - Mostly: Gaussian function;
    rarely: Mexican Hat function
  - Width decreases during training
    (→ implicit decrease of learning rate)
  - *may* decrease to zero (→ k-means)



Gaussian
(not normalized)

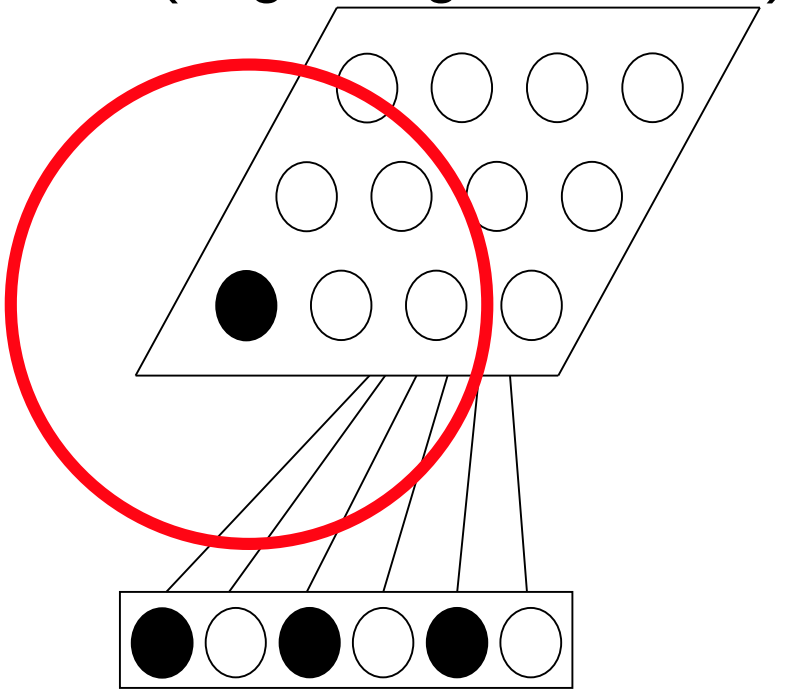


Mexican Hat
(Difference of Gaussian)

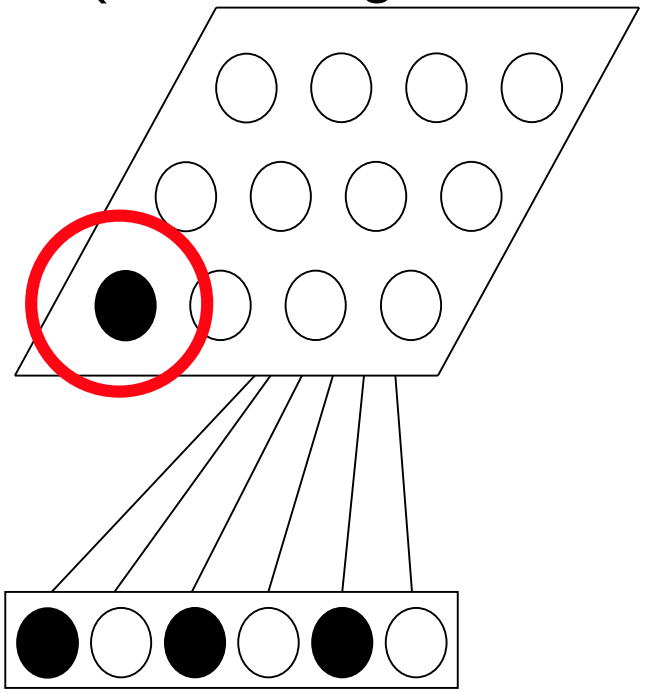


The Self-Organizing Map

Begin of training
(large neighbourhood)



End of training
(small neighbourhood)



If neighbourhood reduced to zero size:
→ behavior like k-means

K-means vs. SOM

- "Clusters"
- Cluster movements independent of others
  - cluster indices can be exchanged
- # clusters typically small
- Used for clustering
- Learning typically in batch mode
  - convergence after few batches
  - on-line mode possible

- "Neurons"
- Neurons co-excited by neighbourhood function
  - Neurons arranged on a map in 1D, 2D, …
- # neurons typically large
- Used for mapping
- Learning in online mode
  - approximate convergence
  - no batch mode

Self-Organization

- Global ordering from local interactions
  - Each neuron sees its neighbors
  - The whole network becomes ordered

- Understanding self-organization is part of ***complexity science***

Network Size

We have to predetermine the network size

- Small network
  - Too much generalization
    → no differentiation
- Large network
  - Each neuron can represent exact features
    → “little generalization”
- Neighbourhood interaction ties parameters to each other
  - if **not** letting neighbourhood interaction size decrease to zero:

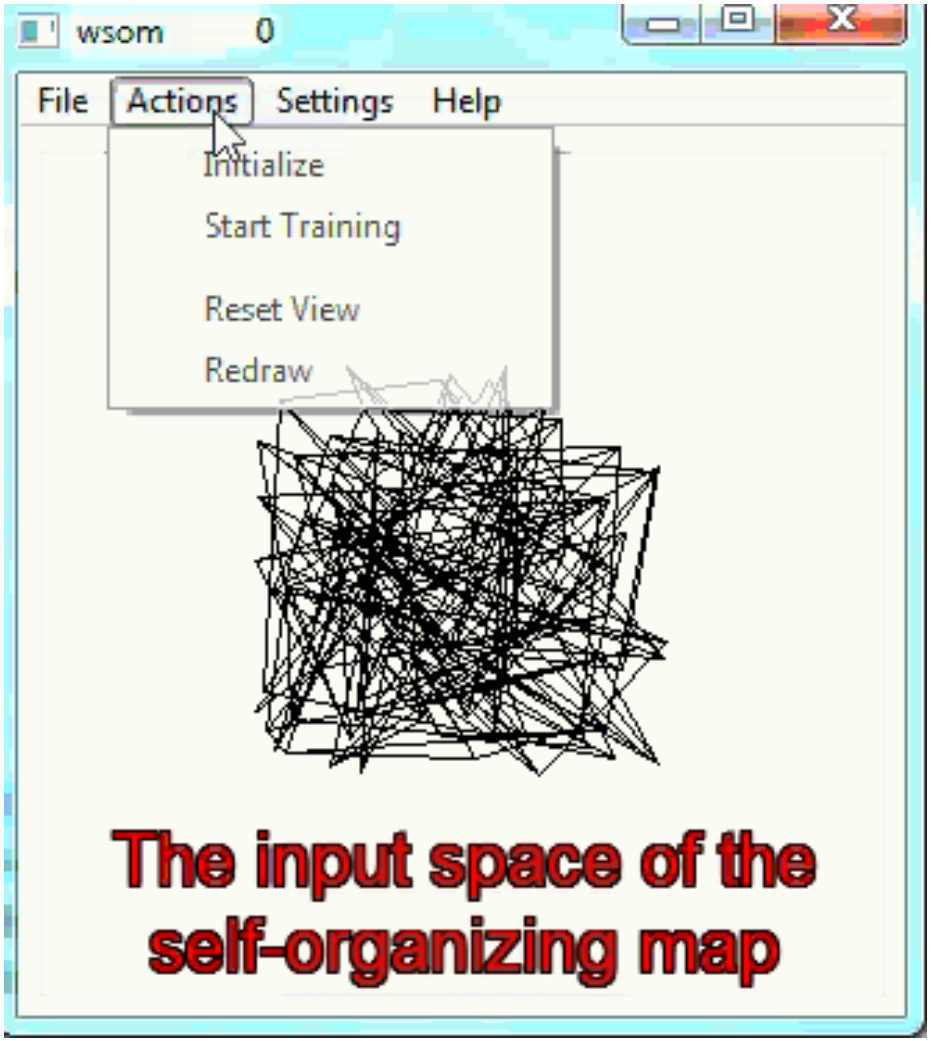
    → small effective network size

    → this “regularization” allows SOMs
    to be kept large

Self-organizing Maps – Overview

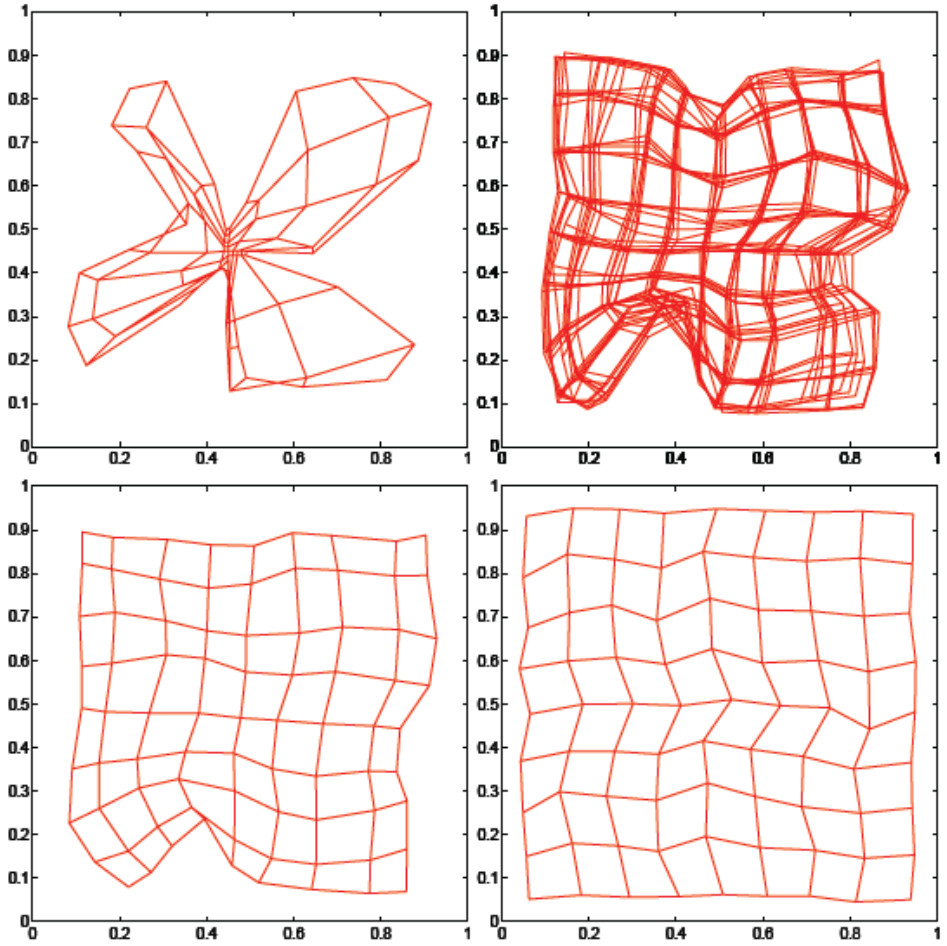
- Topological Maps
- SOM Cost Function and Learning Algorithm
- Visualizing SOMs
- SOM Applications
- Growing Neural Gas & Growing When Required networks

SOM Demo – 2D lattice in 2D / 3D input space



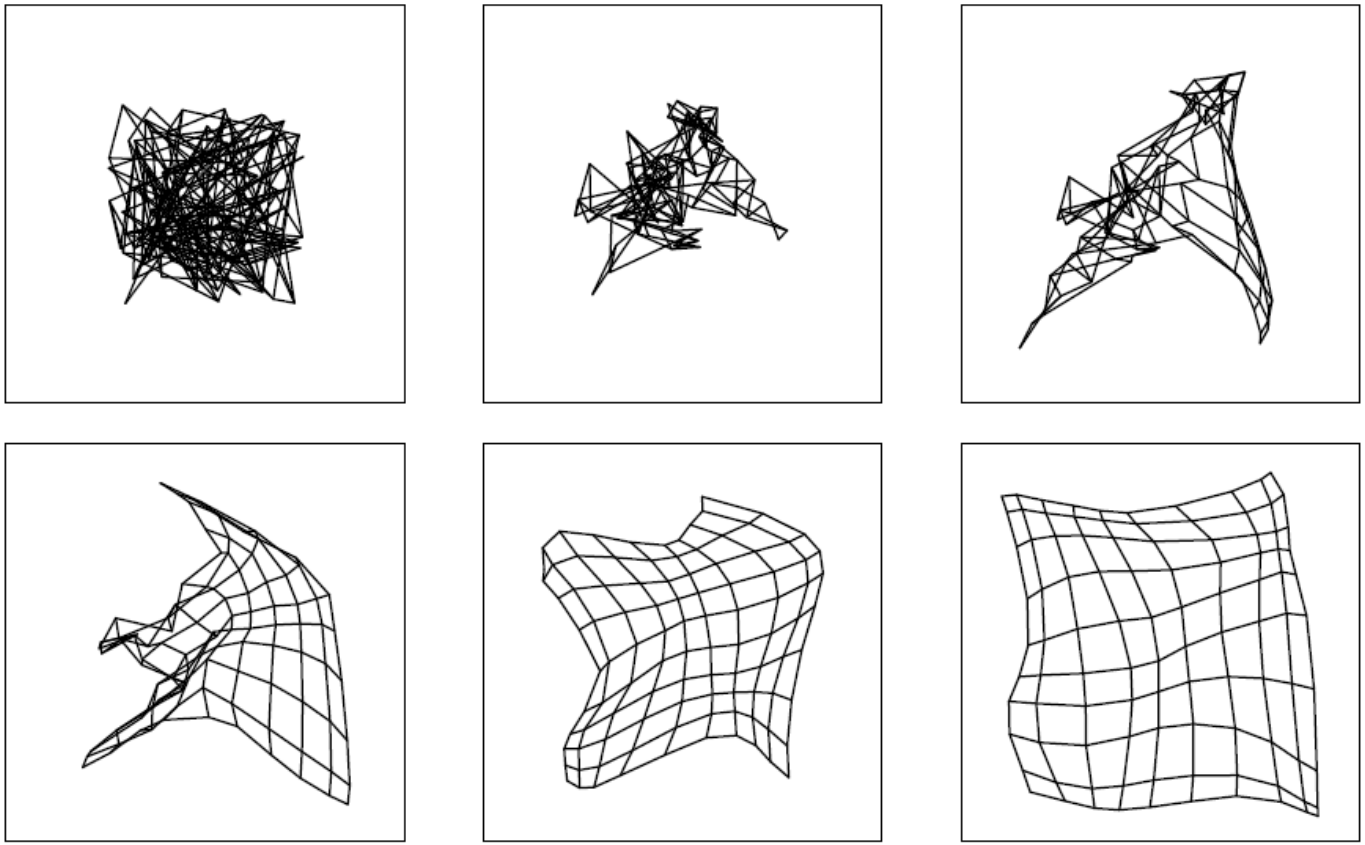
[http://www.borgelt.net/somd.html]

Good Fit of dimensions: Mapping a Square with a 2-dimensional Lattice

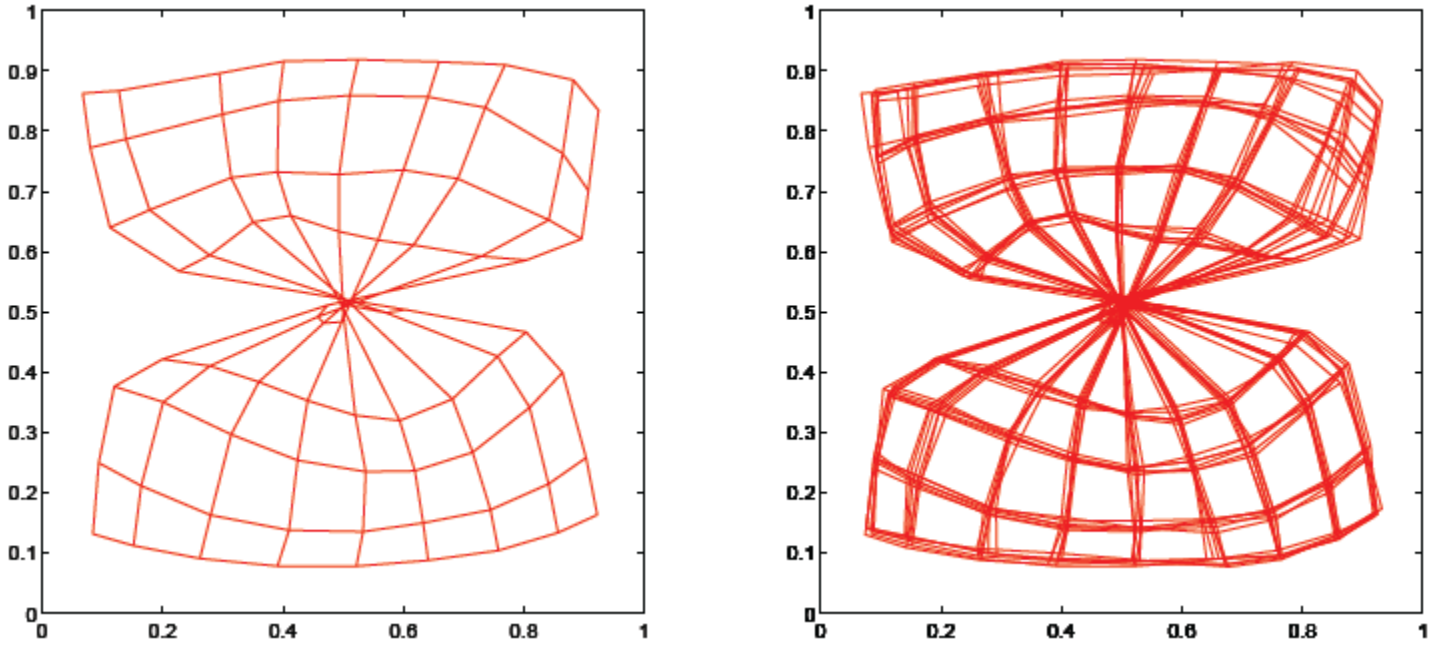


- Upper right with some overlapped learning iterations
- Results for 100, 1000, 5000, 10000 iterations (Kohonen 1984)

Unfolding of a 2-D Map in a 2-D Data Space



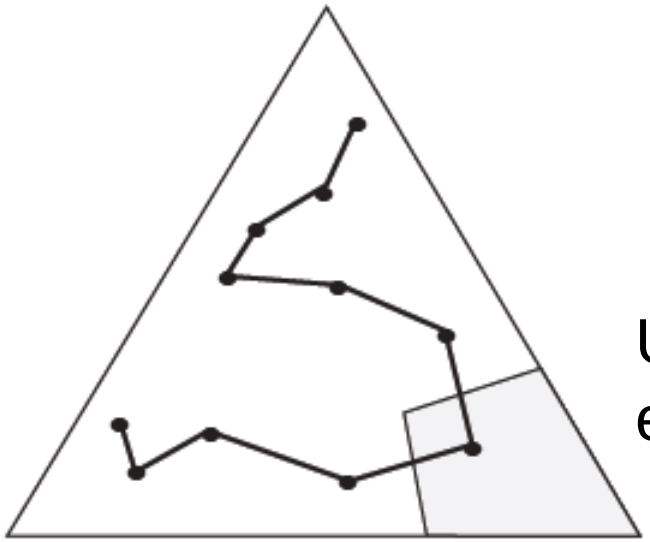
Planar Network with a Knot



State difficult to correct

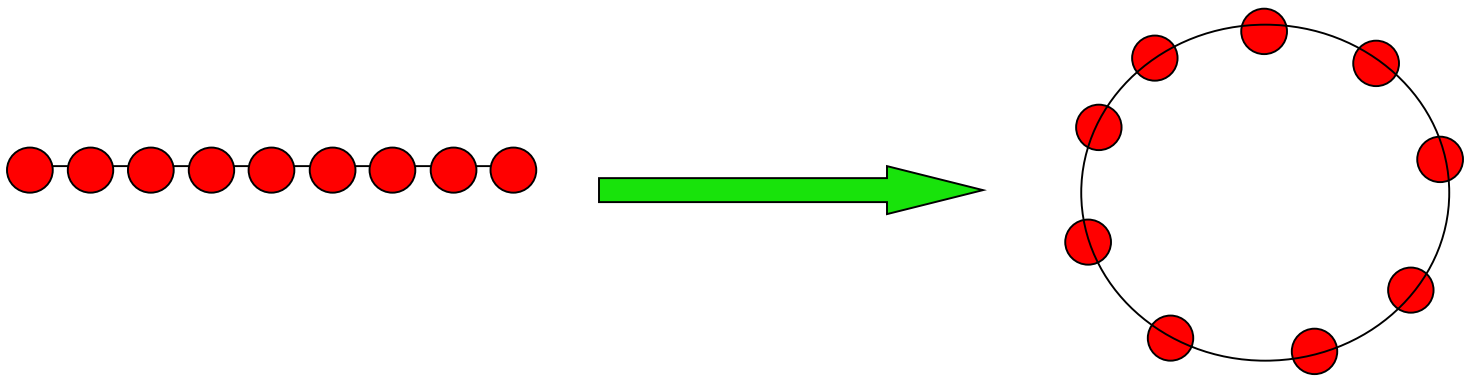
1D Lattice Mapping a 2D Triangular Region

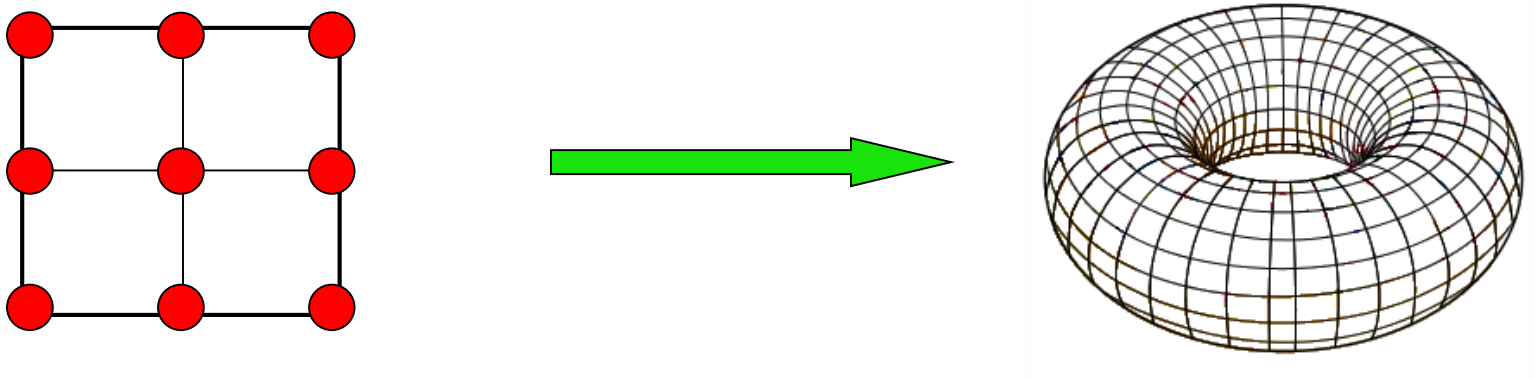
- Triangular input domain is mapped to a small number of one-dimensional representative units



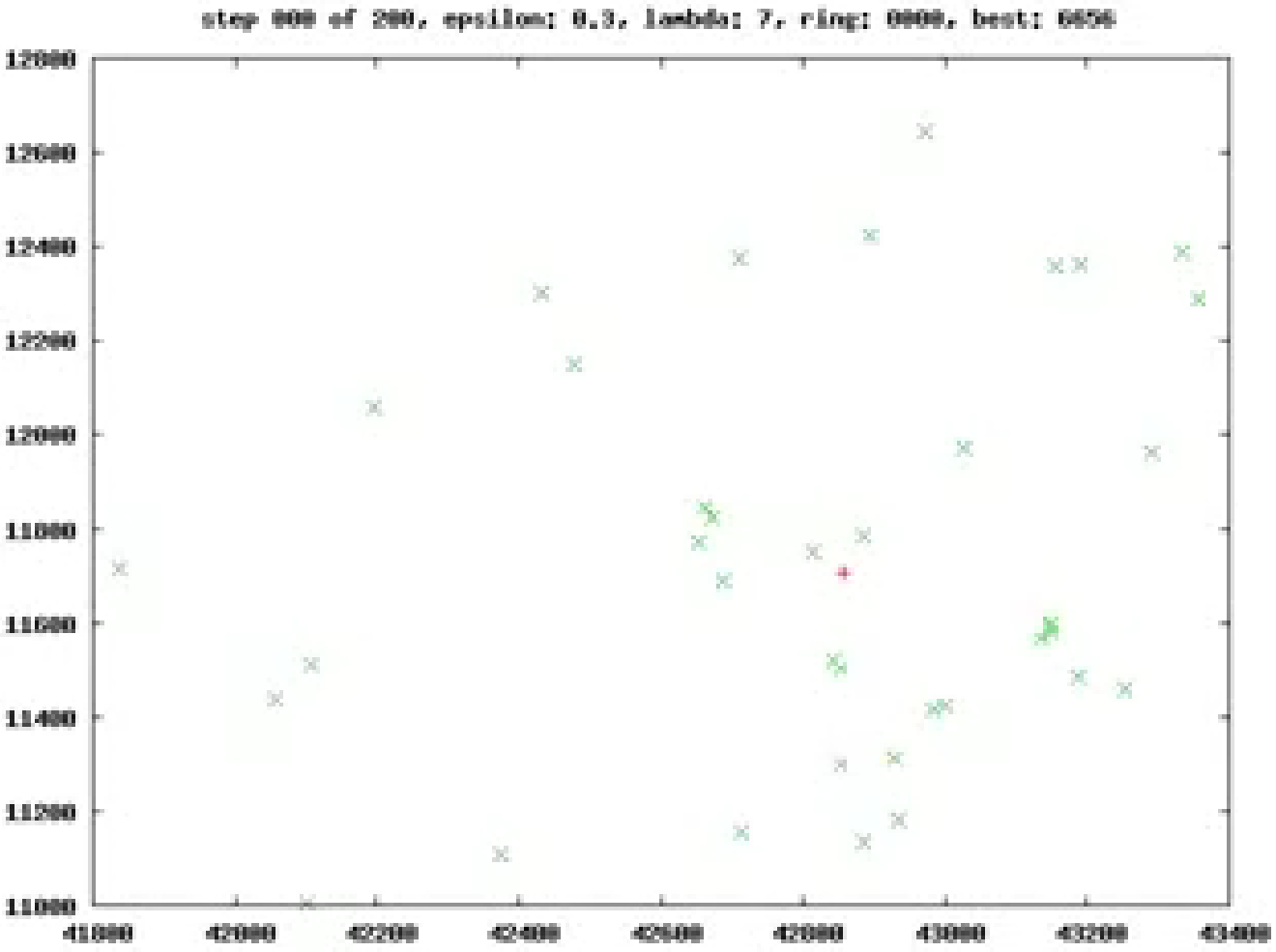
Unit with strongest excitation for shaded region

Boundary Conditions:
No Neurons at the End of the Map



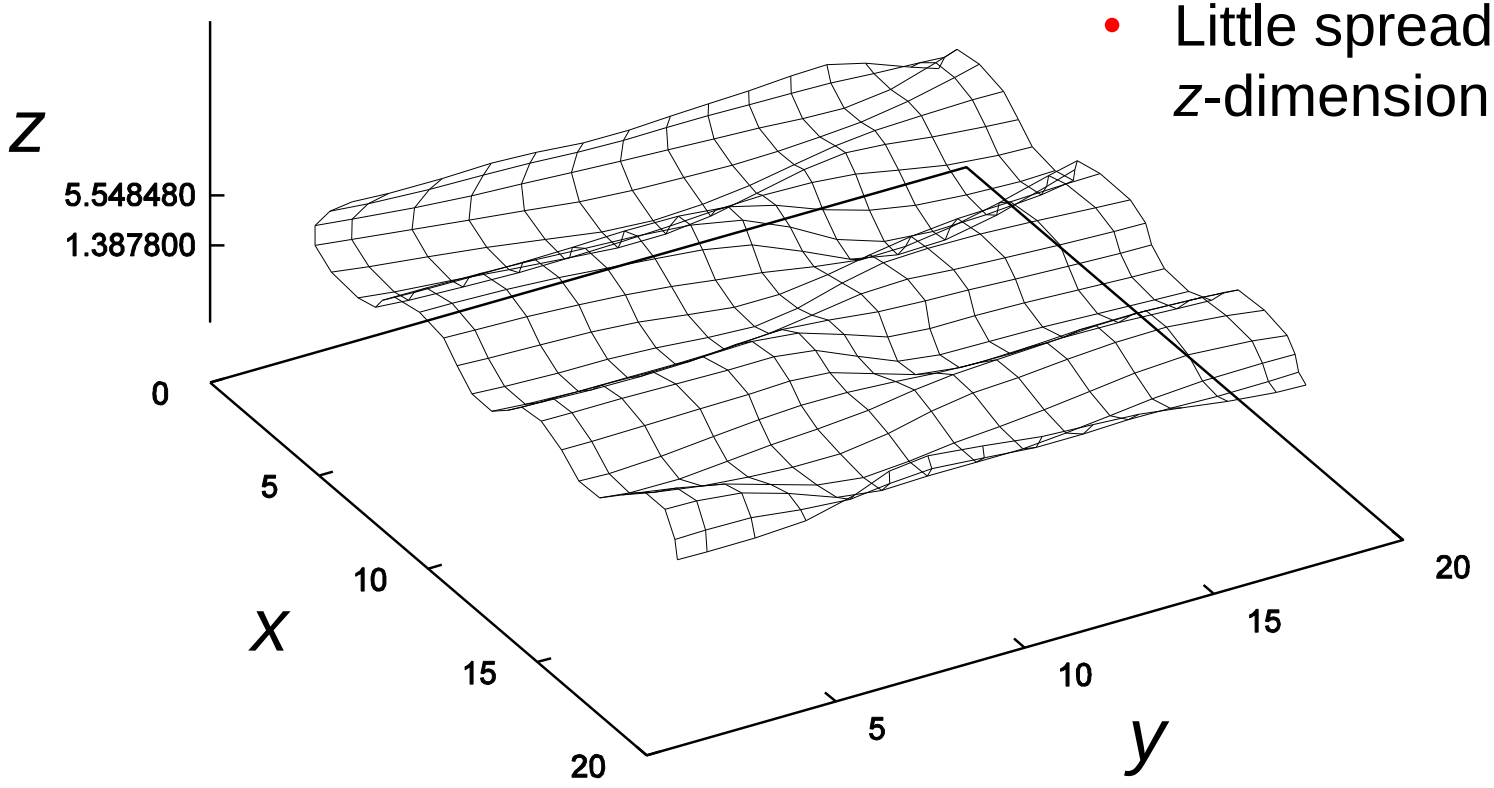


1D Ring-Form SOM for the Travelling Salesman Problem



Unfolding of a 2-D Map in a 3-D Data Space

- Large spread of data in *x*- and *y*-dimensions
- Little spread of data in *z*-dimension



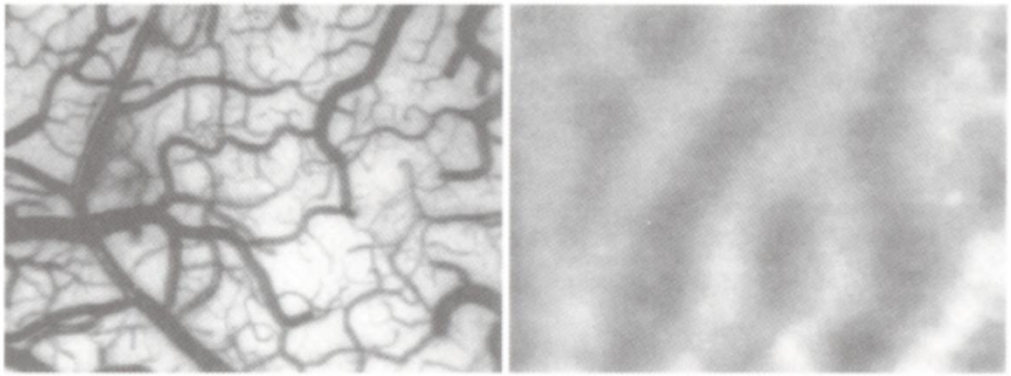
Self-organizing Maps – Overview

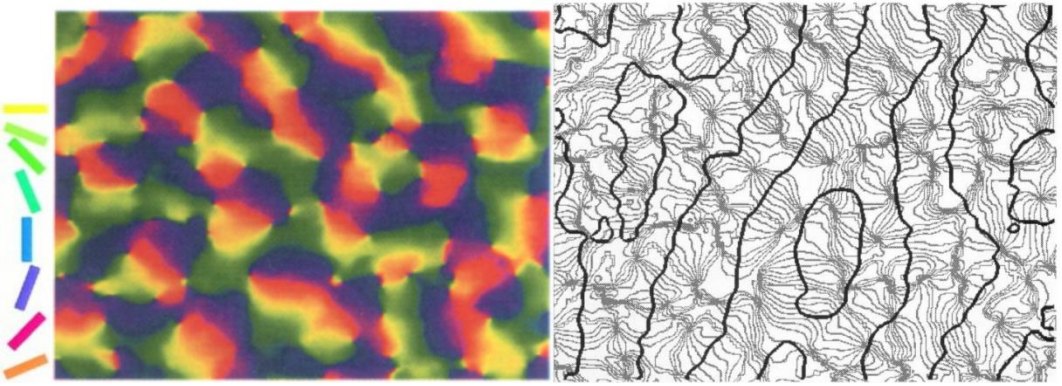
- Topological Maps
- SOM Cost Function and Learning Algorithm
- Visualizing SOMs
- SOM Applications
- Growing Neural Gas & Growing When Required networks

Feature Maps in Visual Cortex V1

Biology

ocular dominance (OD)

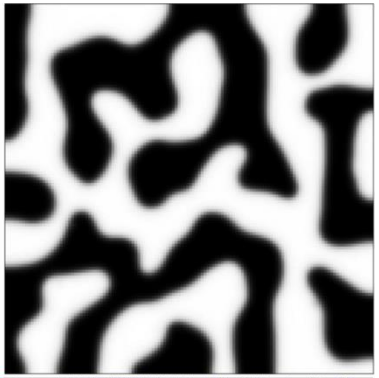


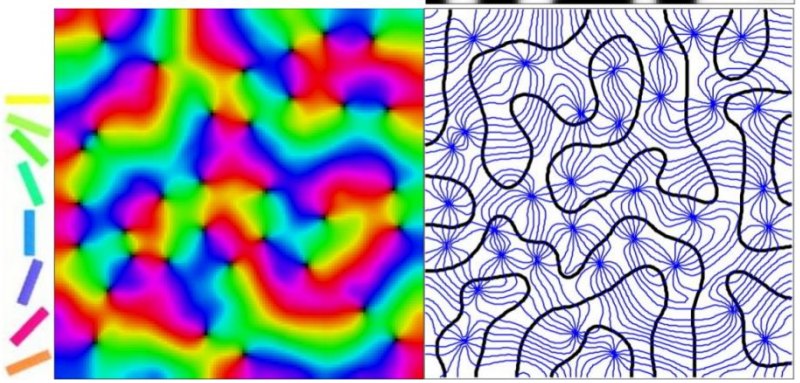


orientation preference (ori) iso-contours

SOM

4D inputs: pos_x, pos_y, OD, ori



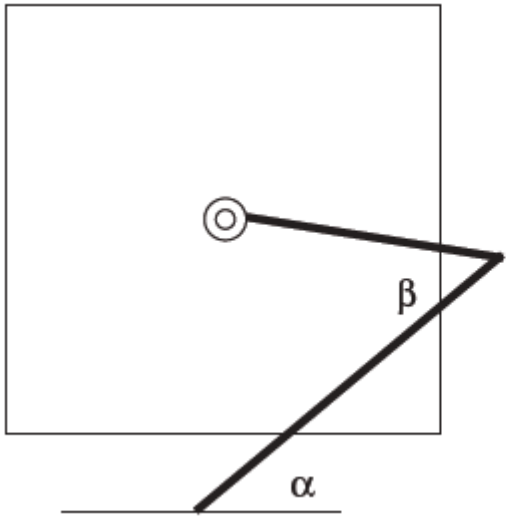


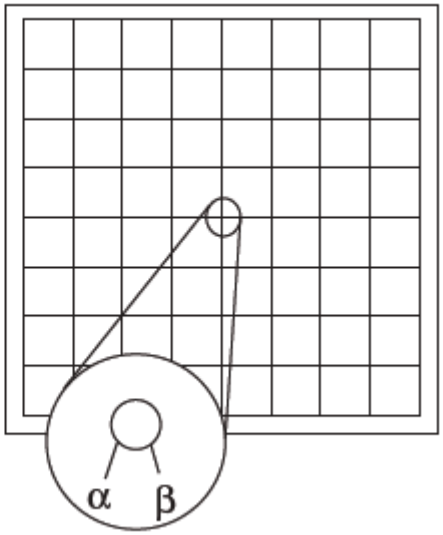
Obermayer, Blasdel. .. Orientation and Ocular Dominance Columns in Monkey .. Jneurosci, 1993

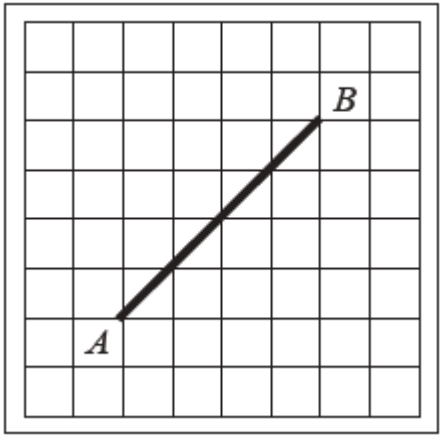
Goodhill. .. Theoretical Modeling .. Neural Map Development. Neuron, 2007

Learning Simple Inverse Kinematics (IK)

- Map configuration space of a robot arm using 2D network
- For one point only one parameter combination
  - Pair input *(x,y)* coordinates with "output" coordinates *(α,β)*
    → *supervised learning*



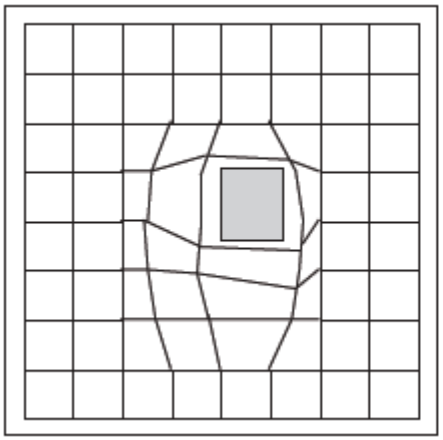


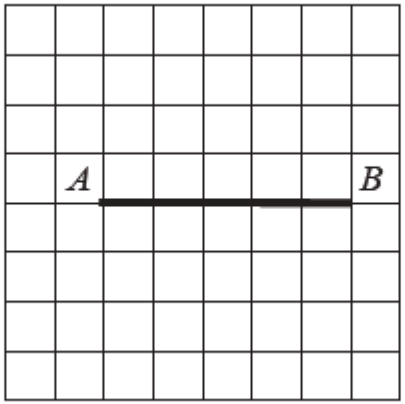


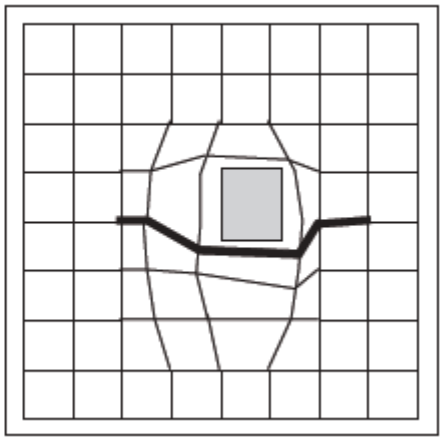
→ IK: Given input *(x,y)*, retrieve matching *(α,β)*

Learning Obstacles in Work Area

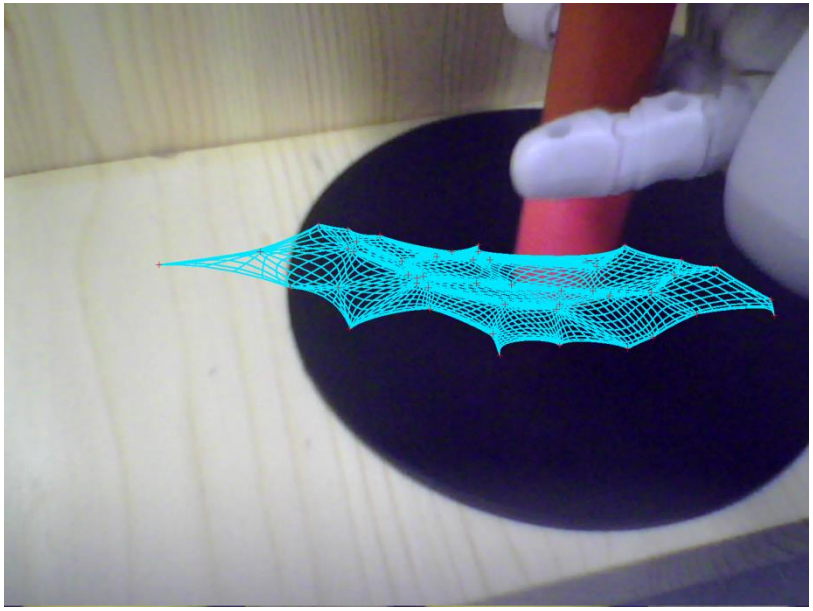
- SOM charts configuration space avoiding the obstacles
- Moving the arm from A to B avoids the obstacle







Learning Simple Inverse Kinematics



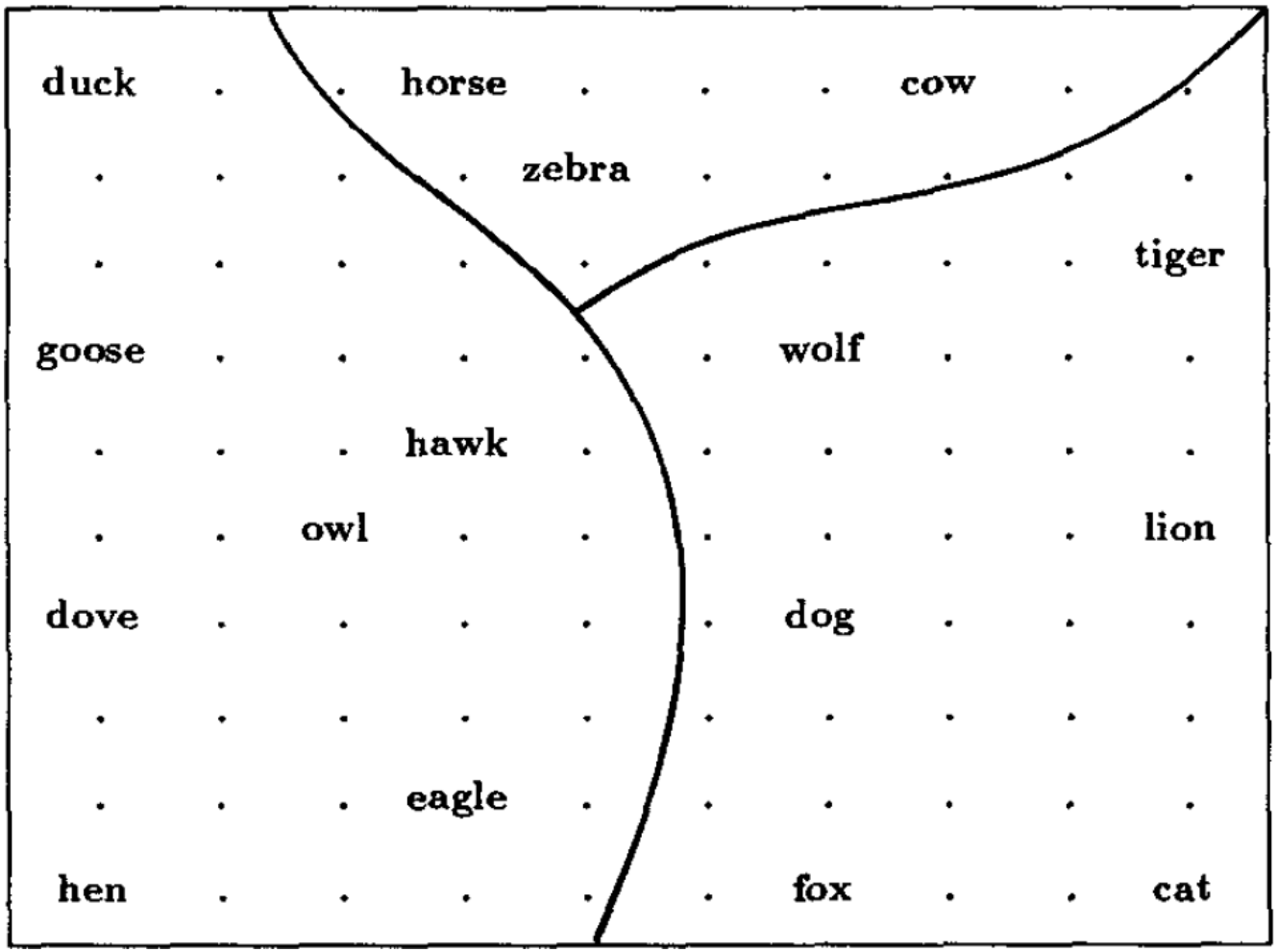
Semantic Map – Input data

attributes

data points

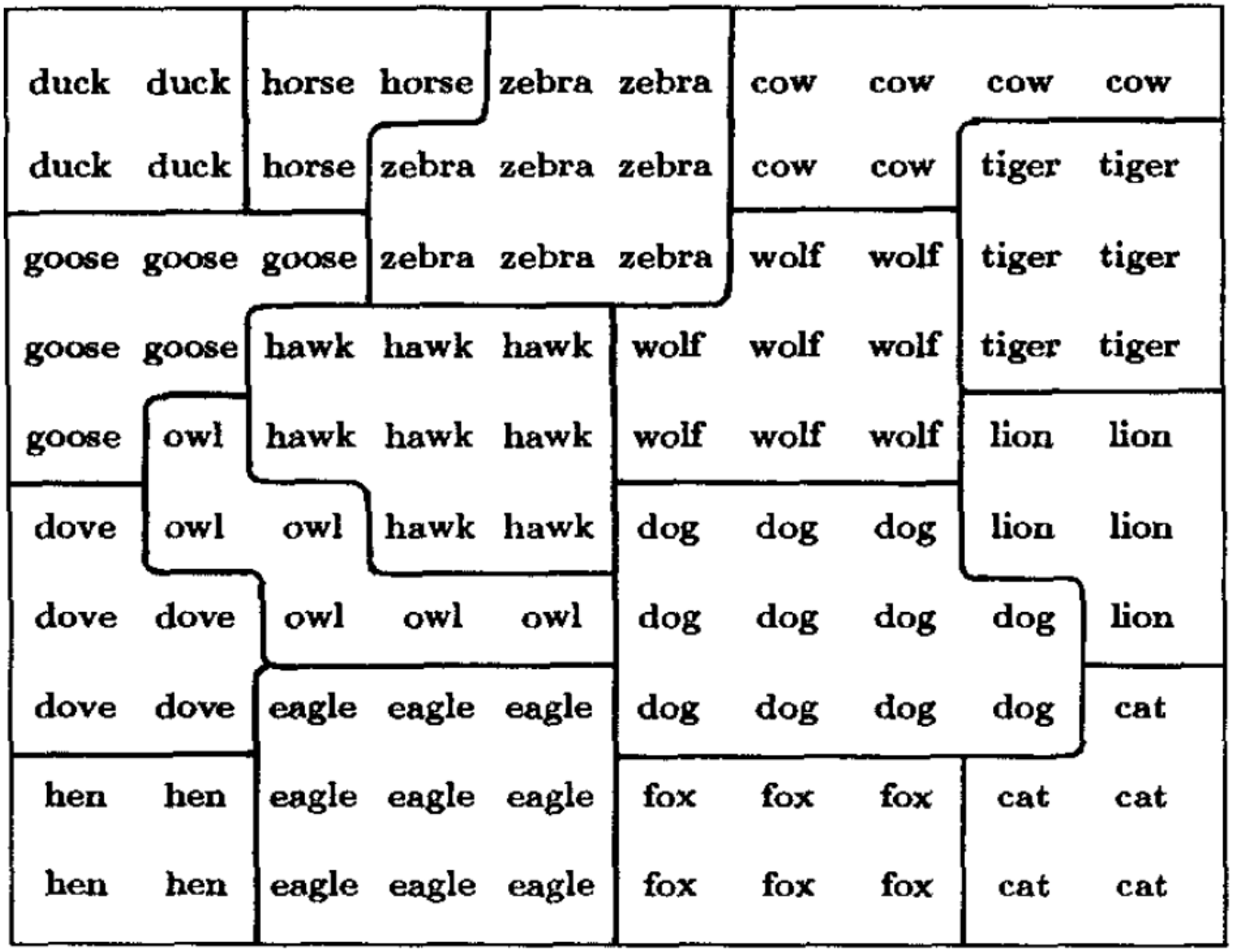
		dove	hen	duck	goose	owl	hawk	eagle	fox	dog	wolf	cat	tiger	lion	horse	zebra	cow
is	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
has	2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
likes to	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Semantic Map - Results



winning units are labeled

Semantic Map - Results



each unit is labeled according to its nearest data point

Self-organizing Maps – Overview

- Topological Maps
- SOM Cost Function and Learning Algorithm
- Visualizing SOMs
- SOM Applications
- Growing Neural Gas & Growing When Required networks

Existing Neural Clustering models

- ***Static Models*** (fixed number of units): Competitive Learning (CL), Self-Organizing Map (SOM), Neural Gas (NG), ….
  - *Hierarchical*: Multilayered Self-Organising Feature Maps (M-SOM), etc.

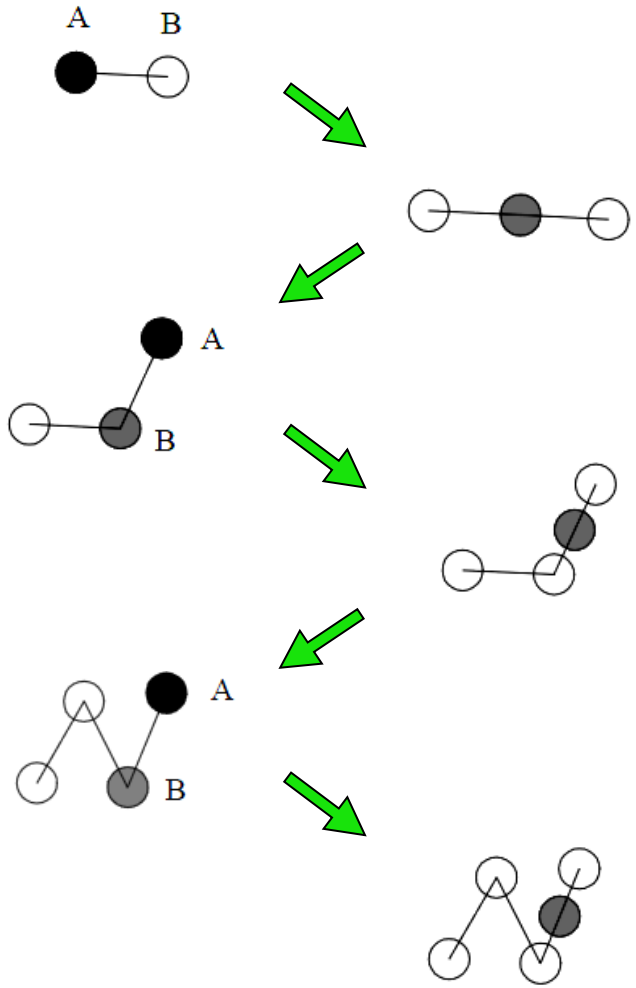
There are shortcomings for static models in a non-stationary environment.

- ***Dynamic Models*** (variable number of units): Growing Grid (GG), Growing Cell Structure (GCS), Growing Neural Gas (GNG), Grow When Required (GWR), etc.
  - *Hierarchical:* Growing Hierarchical Self-Organizing Map (GHSOM), etc.

Shortcomings of Static Models in a Non-stationary Environment

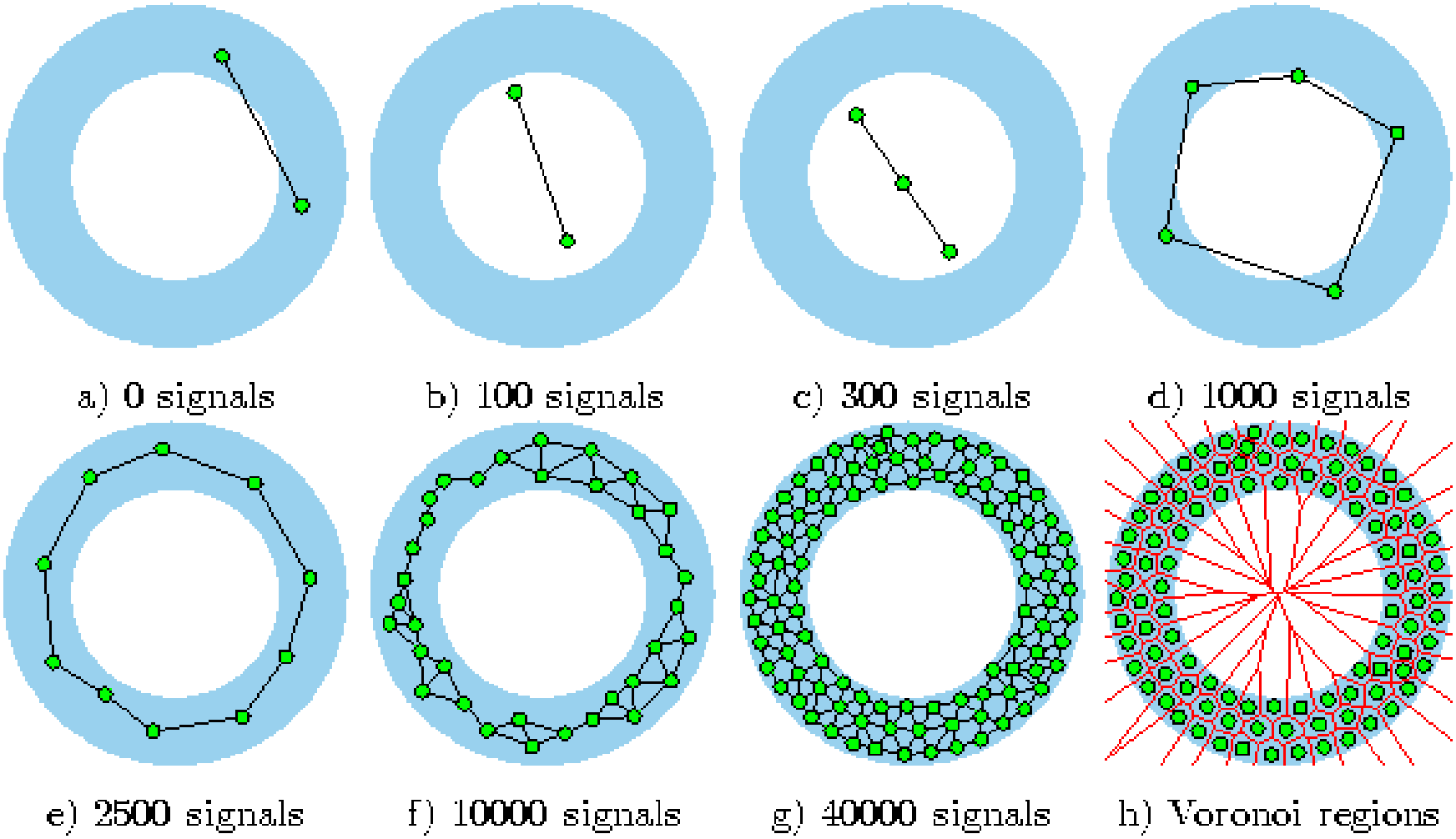
- Pre-defined number of units
- Pre-defined topology (mostly a grid)
- Pre-defined training length
- A decaying learning rate (e.g. SOM, CL, NG, GG, GSOM, Snet-SOM, M-SOM, GHSOM)

Growing Neural Gas – GNG
(Fritzke, 1994)



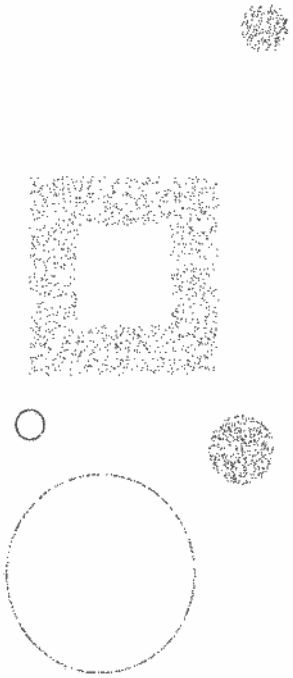
- GNG adds unit after every pre-defined period.
- Units are represented as circles.
- Circle A indicates ***unit with maximum accumulated error***
- Circle B indicates ***neighbor with largest error*** for Circle *A*.
- The grey circle is the ***new unit*** at each stage.

Growing Neural Gas – GNG
(Fritzke, 1994)

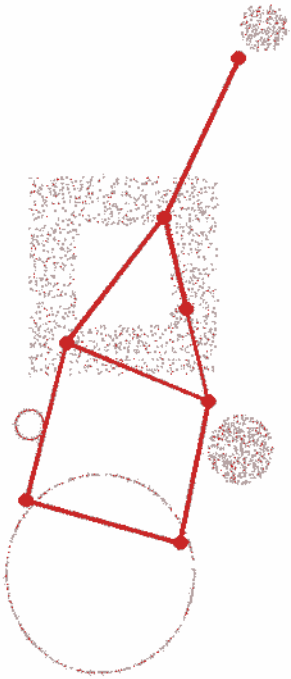


Growing Neural Gas – GNG
(Fritzke, 1994)

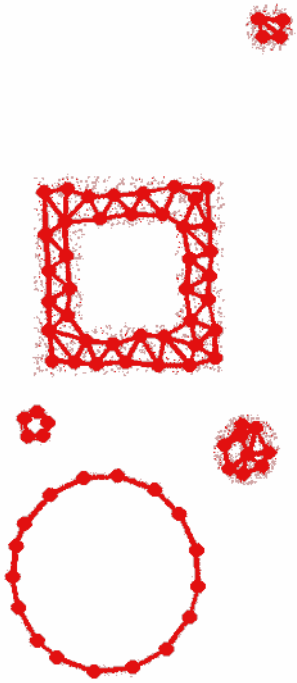
data



early GNG



late GNG



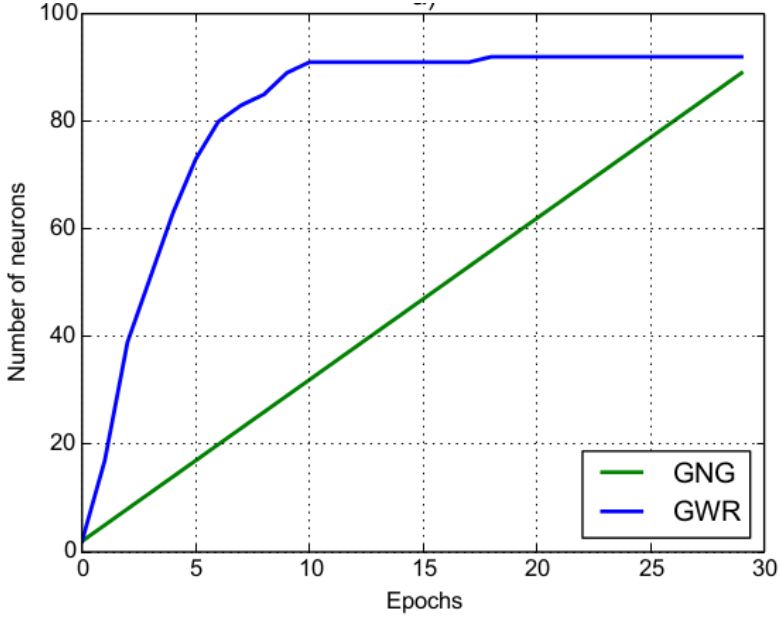
Growing When Required – GWR
(Marsland et al., 2002)

GNG: network grows with *time*

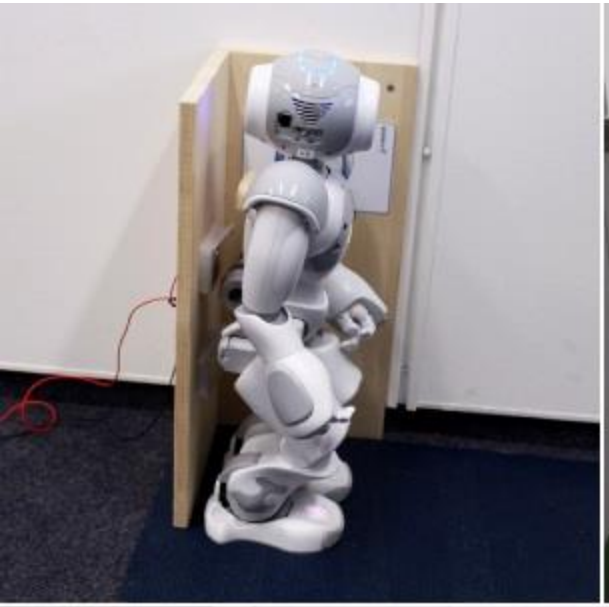
GWR: network grows *when required*

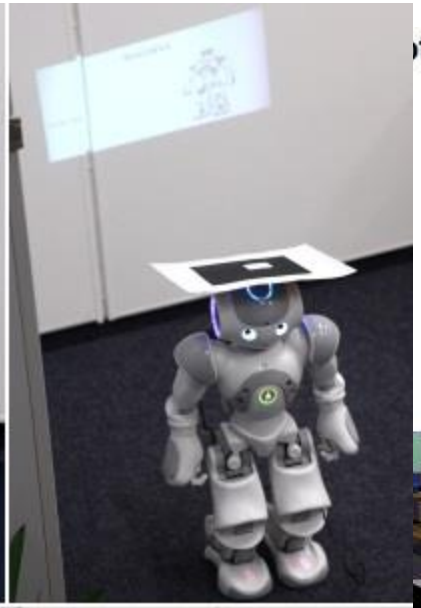
- Each neuron has a *firing counter* (how often is it the winner)
  - Many counts → lower learning rate to help convergence ("habituation")
  - Few counts → novelty detected
- Each edge has an *age* (refreshed when linking winner and 2nd winner)
  - Old edge → cut it (and remove units when isolated)

→ strong neurogenesis at early stages



Room Mapping via GNG (KT Lab)

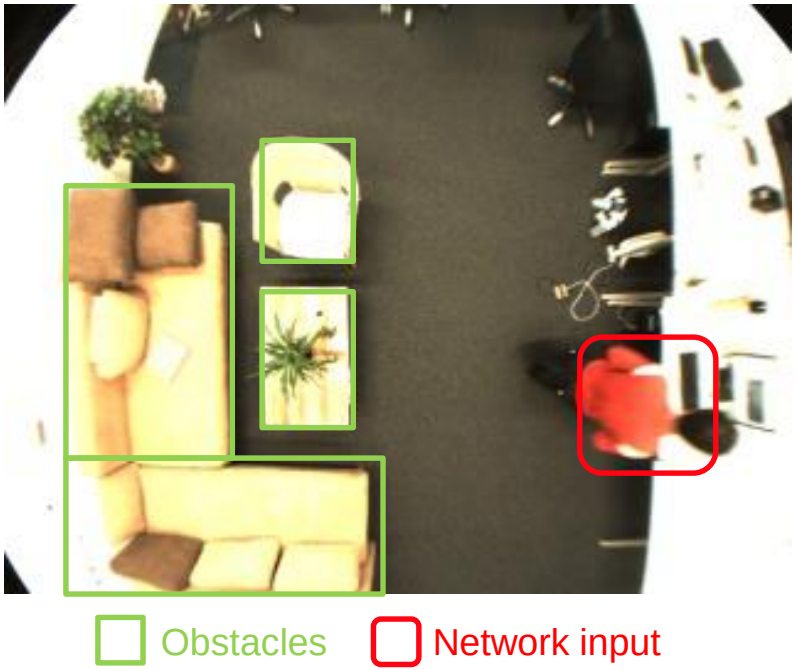




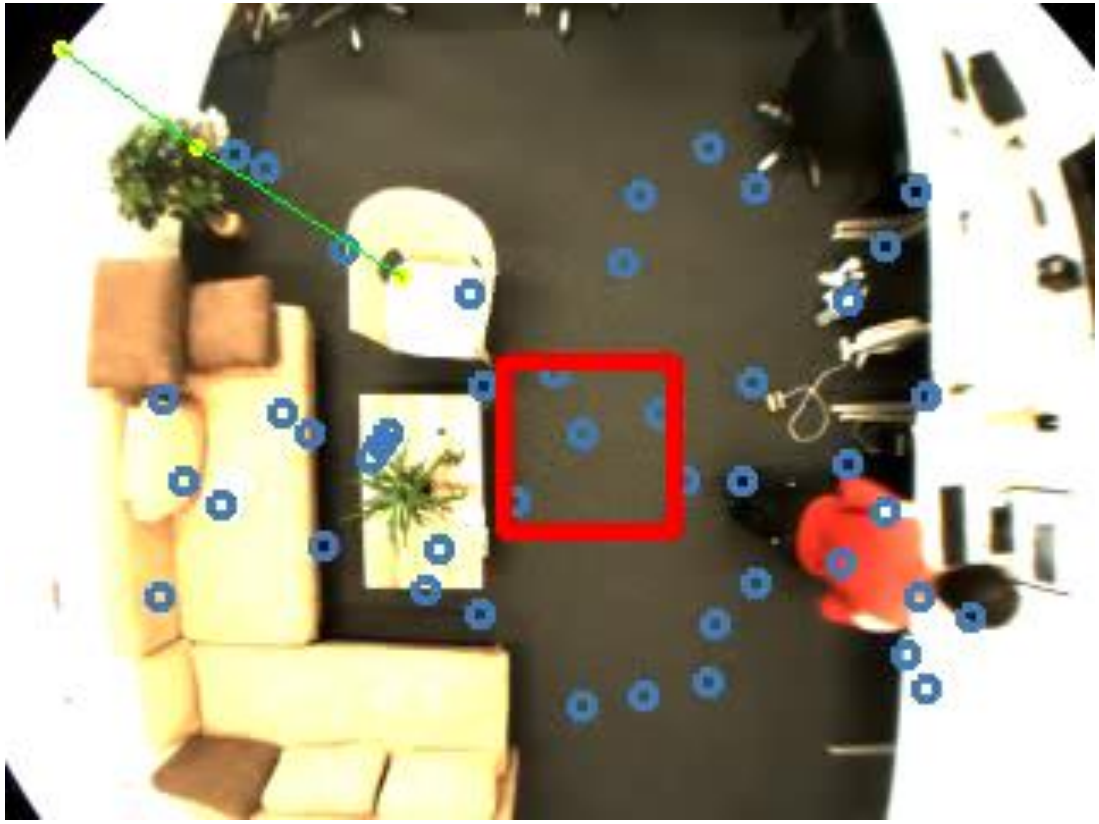
Room Mapping via GNG (KT Lab)

- Map the topological structure of a room using growing neural gas for robot navigation

- Since the person's movement can show the free space in the room, we use the detected position of a person as the network input



Room Mapping via GNG (KT Lab)



- Blue dots: particles for person detection
- Red bounding box: estimated position of target person
- Yellow dots: the neurons of the growing neural gas and the green lines are the connections

Self-organizing Maps – Summary

- WTA behavior of Kohonen's algorithm resembles k-means
  - unlike the distributed code on most (deep) neural networks
- Topology preservation is a feature of Kohonen's SOM
  - brain-inspired, implemented by near-surround excitation
  - leads to good spread of neurons over the data space
  - grid of 2D topology ideal for visualization
- Growing architectures
  - do not use grid to define neighbor interactions
  - use flexibly linked neighbours