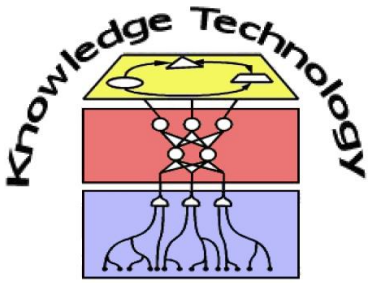
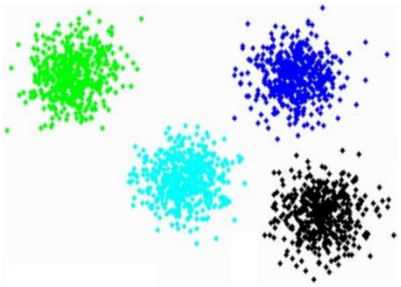
Data Mining

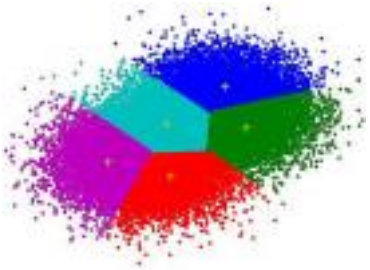
Lecture 8
Clustering and Selforganizing Networks



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



What is Cluster Analysis?

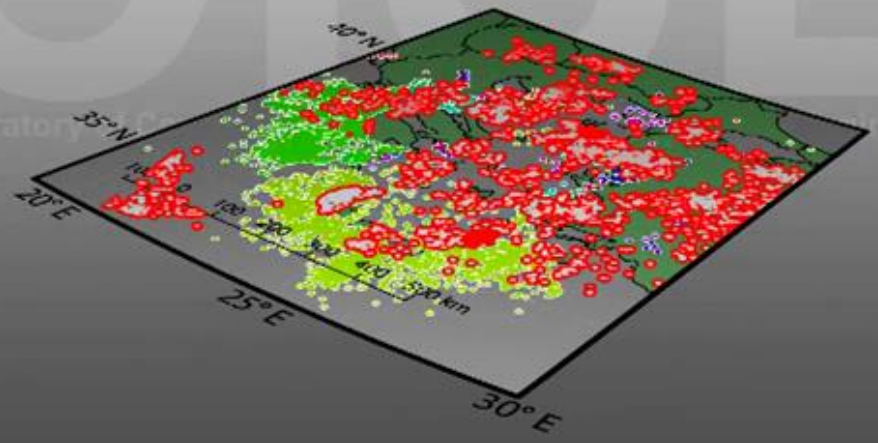


- Cluster: A group of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis
  - Finding similarities between data according to their characteristics and grouping similar data objects into clusters
- ***Unsupervised learning***: no predefined classes
- Typical applications
  - As a ***stand-alone tool*** to get insight into data distribution
  - As a ***preprocessing step*** for other algorithms

Clustering for Data Understanding and Applications

- Biology: taxonomy of living things: class, family, genus and species
- Information retrieval: document clustering
- Marketing: help marketers discover distinct customer groups, and develop targeted marketing programs
- Land use: similar land use in an earth observation database
- City-planning: identifying groups of houses according to their house type, value (and geographical location)
- Climate: understanding earth climate, find patterns of atmospheric similiarities
- Earth-quake studies: observed earth quake epicenters should be clustered along continent faults

Spatiotemporal Clustering of Earthquakes in Greece



http://www.teicrete.gr

Quality: What Is Good Clustering?

- High quality clusters
  - high ***intra-class*** similarity: ***cohesive*** within clusters
  - low ***inter-class*** similarity: ***distinctive*** between clusters
- The **quality** of a clustering method depends on
  - the ***similarity measure*** used by the method
  - its implementation, and
  - its ability to discover the ***hidden patterns***

Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - ***Similarity*** is expressed in terms of a ***distance function***, e.g. a ***metric*** $d(i, j)$
  - The definitions of distance functions are usually different for interval-scaled, boolean, categorical, or vector variables
  - Different variables may be assigned different weights based on applications and data semantics
- Quality functions ("Clustering Indices")
  - The "goodness" of a clustering process can be quantified, e.g. by relating intra-class vs. inter-class similarities
  - It is hard to set thresholds for "similar enough" for data points or "good enough" for the clustering

Typical Requirements

- Scalability
- Incremental clustering and insensitivity to input order
- High dimensionality
- Ability to deal with different types of attributes
- Ability to deal with noisy data
- Discovery of clusters with arbitrary shape
- Constraint-based clustering
- Domain knowledge to determine input parameters
- Interpretability and usability

Major Clustering Approaches (1)

- ***Partitioning approach***
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of squared errors
  - Typical methods: k-means, k-medoids, CLARANS
- ***Hierarchical approach***
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- ***Density-based approach***
  - Based on connectivity and density functions above threshold
  - Typical methods: DBSCAN, OPTICS, DenClue

Major Clustering Approaches (2)

- ***Grid-based approach***
  - multiple-level granularity structure, finite number of cells
  - Typical methods: STING, WaveCluster, CLIQUE
- ***Model-based***
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical: Gaussian Mixture Models, EM, SOM, COBWEB
- ***Frequent pattern-based***
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- ***Instance-based***
  - Typical: K-nearest neighbors (kNN) — *classify* a data point by the majority vote of its K closest neighbour points

Data Matrix and Dissimilarity Matrix

- ***Data matrix***
  - n data points with p dimensions

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

- ***Dissimilarity matrix***
  - Registers the distances between the n data points
  - A triangular matrix

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Reminder: Numeric Data: Minkowski Distance

- ***Minkowski distance***: A popular distance measure

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where

$i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects,

h = order (the distance so defined is also called L-h norm)

- Properties
  - $d(i, j) > 0$ if $i \neq j$, and $d(i, i) = 0$ (Positive definiteness)
  - $d(i, j) = d(j, i)$ (Symmetry)
  - $d(i, j) \leq d(i, k) + d(k, j)$ (Triangle Inequality)
- A distance that satisfies these properties is a **metric**

Reminder: Special Cases of Minkowski Distance

- *h* = 1: (L_1 norm) ***Manhattan distance***

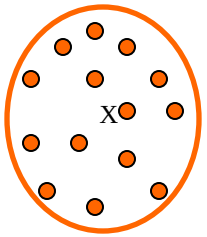
$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + ... + |x_{ip} - x_{jp}|$$

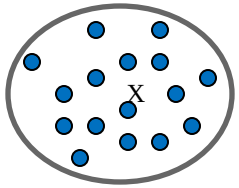
  - E.g., the *Hamming distance* between two *binary vectors*: the number of bits that are different

- *h* = 2: (L_2 norm) ***Euclidean distance***

$$d(i, j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{ip} - x_{jp}|^2}$$

Distance between Clusters





- Single link: smallest distance between any element (p) in one cluster (i) and any element (q) in the other (j), i.e., $dist(K_i, K_j) = min_{p,q}\ d(x_{ip}, x_{jq})$
- Complete link: largest distance between any element in one cluster and any element in the other, i.e., $dist(K_i, K_j) = max_{p,q}\ d(x_{ip}, x_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = avg_{p,q}\ d(x_{ip}, x_{jq})$
- Centroid: distance between the centroids of two clusters, i.e., $dist(K_i, K_j) = d(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e., $dist(K_i, K_j) = d(M_i, M_j)$
  - Medoid: a chosen, centrally located ***object*** in the cluster (whose dissimilarity to all other objects in the cluster is minimal)

Centroid, Radius and Diameter of a Cluster (for numerical data sets)

- Centroid: the "center of mass" of a cluster (N_i = # points in the cluster i)
  ← has minimal average Euclidean distance

$$C_i = \frac{\sum_{p=1}^{N_i} x_{ip}}{N_i}$$

- Radius: ~ average Euclidean distance from any point of the cluster to its centroid

$$R_i = \sqrt{\frac{\sum_{p=1}^{N_i} (x_{ip} - C_i)^2}{N_i}}$$

- Diameter: average Euclidean distance between all *pairs* of points in the cluster

$$D_i = \sqrt{\frac{\sum_{p=1}^{N_i} \sum_{q=1}^{N_i} (x_{ip} - x_{iq})^2}{N_i (N_i - 1)}}$$

Partitioning Algorithms: Basic Concept

- ***Partitioning method***: partition a database *D* of objects x_p into a set of *k* clusters, minimising sum of squared distances to means m_i

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (x_p - m_i)^2$$

points assigned to cluster *i*

sum over clusters

each point *p* is assigned to exactly one cluster *i*

- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Find global optimum: exhaustively enumerate all possible partitions
  - Find a local optimum by heuristic methods:
    - ***k-means*** (MacQueen'67): Each cluster is represented by its center
    - ***k-medoids*** or Partition around medoids (PAM, Kaufman&Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is as:
  1. Partition objects into *k* non-empty subsets
  2. Compute the centroids (center of mass / mean) of the clusters of the current partitioning
  3. Assign each object to the cluster with the nearest centroid
  4. Go to Step 2; Stop when none of the assignments changed

The *k*-Means Algorithm

- Initialization
  - Set a value for *k*
  - Place the k cluster centres μ_j at random positions in input space
- Learning: Repeat …
  - For each data point x_p
    - Compute distance to each cluster centre
    - Assign data point to nearest cluster centre with distance

      $d_p = \min_j d(x_p, \mu_j)$
  - For each cluster centre
    - Move position of centre to mean of points in cluster

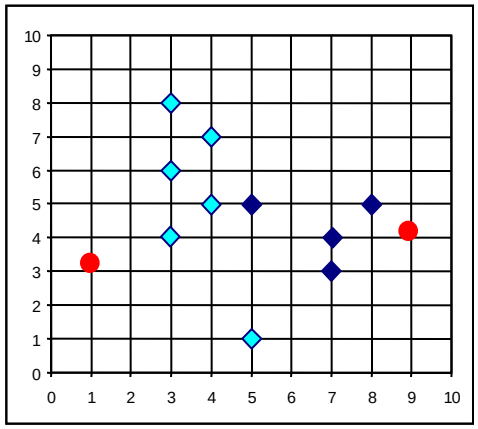
$$\mu_j = \frac{1}{N_j} \sum_{i=1}^{Nj} x_i$$

N_j = number of points in cluster *j*

- … until the assignments don't change (then, cluster centres stop moving)
- On-line version: move cluster center only a bit for each data point

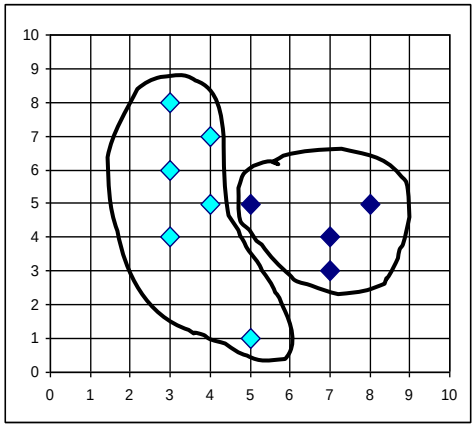
The *K-Means* Clustering Method

- Example

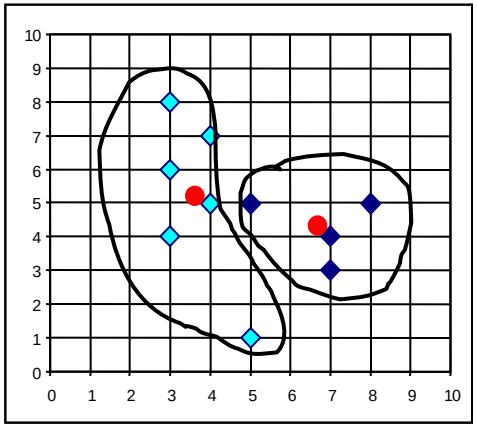


K=2

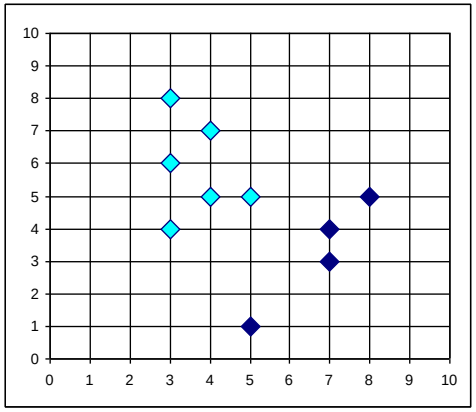
Arbitrarily choose K objects, or K locations, as initial cluster centers



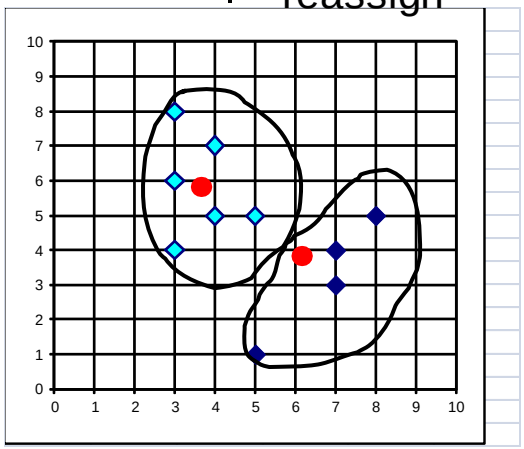
Assign each object to most similar center



Update the cluster means



reassign



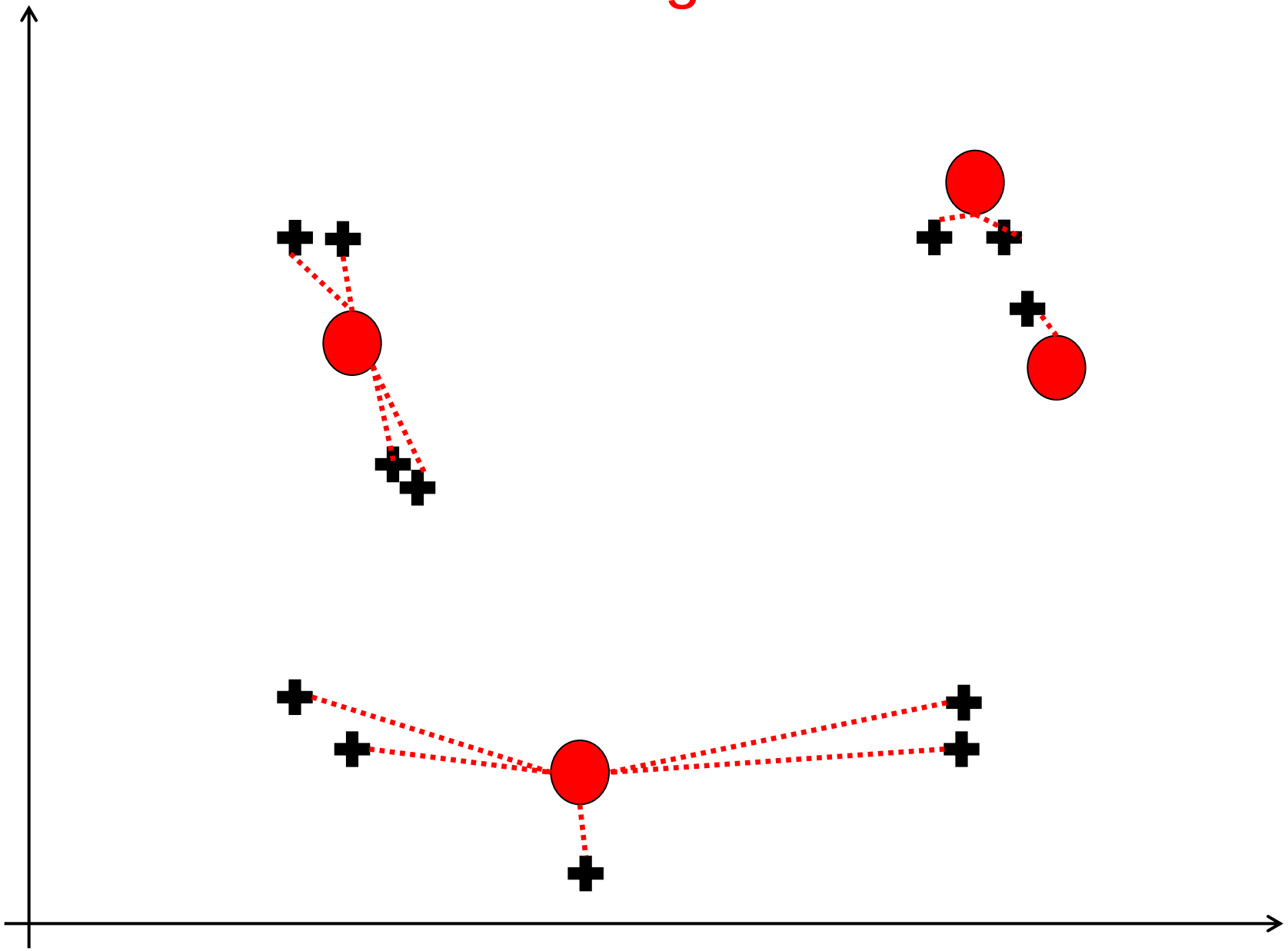
Update the cluster means

reassign

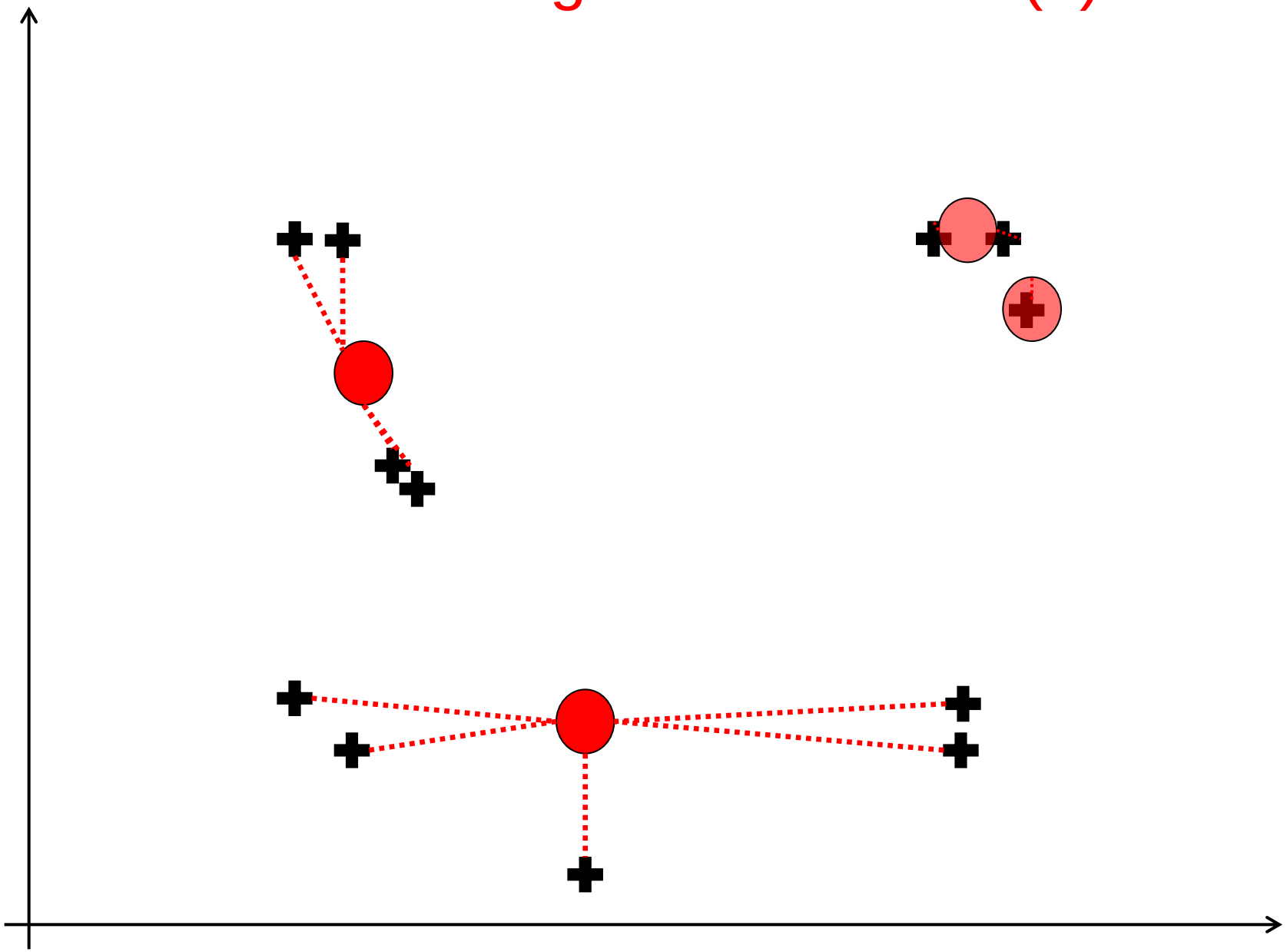
Why k-Means Converges

- Change of assignments reduces the sum squared distances of the datapoints to their assigned cluster centers.
- Moving a cluster center reduces the sum squared distances of the datapoints to their assigned cluster centers.
- If the assignments do not change in the assignment step, we have converged.

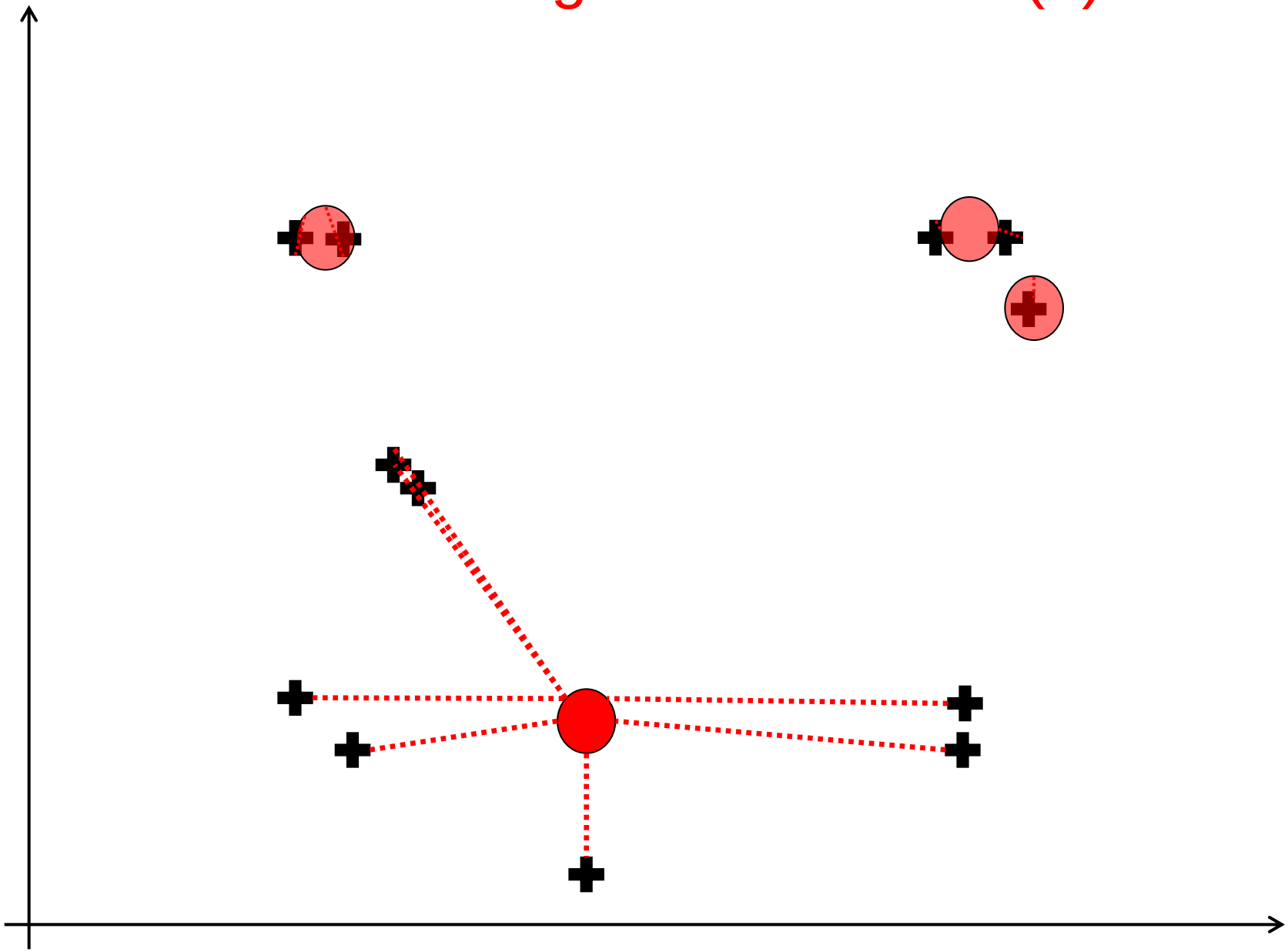
Clustering: 4-means



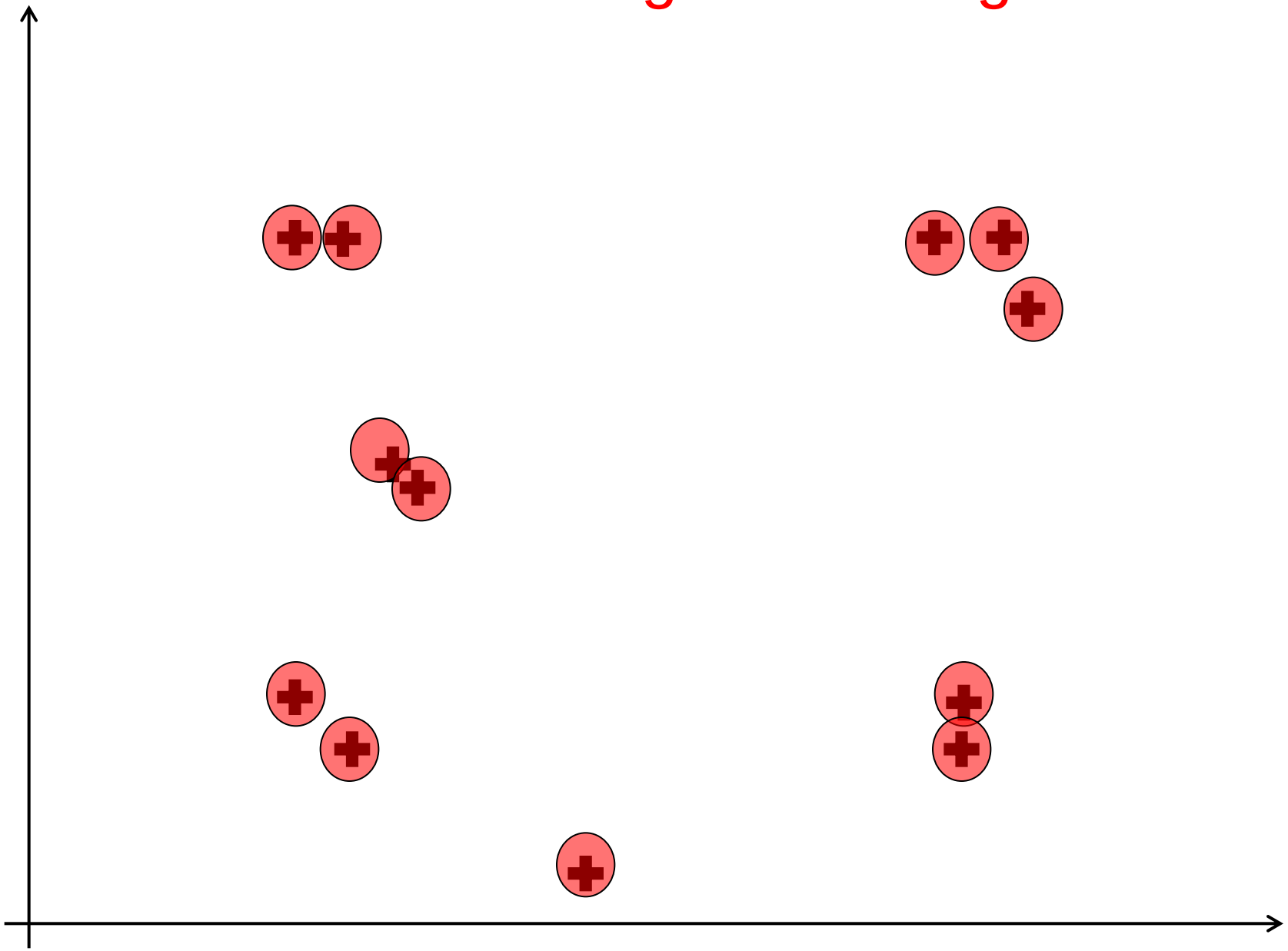
Clustering: Local Minima (1)



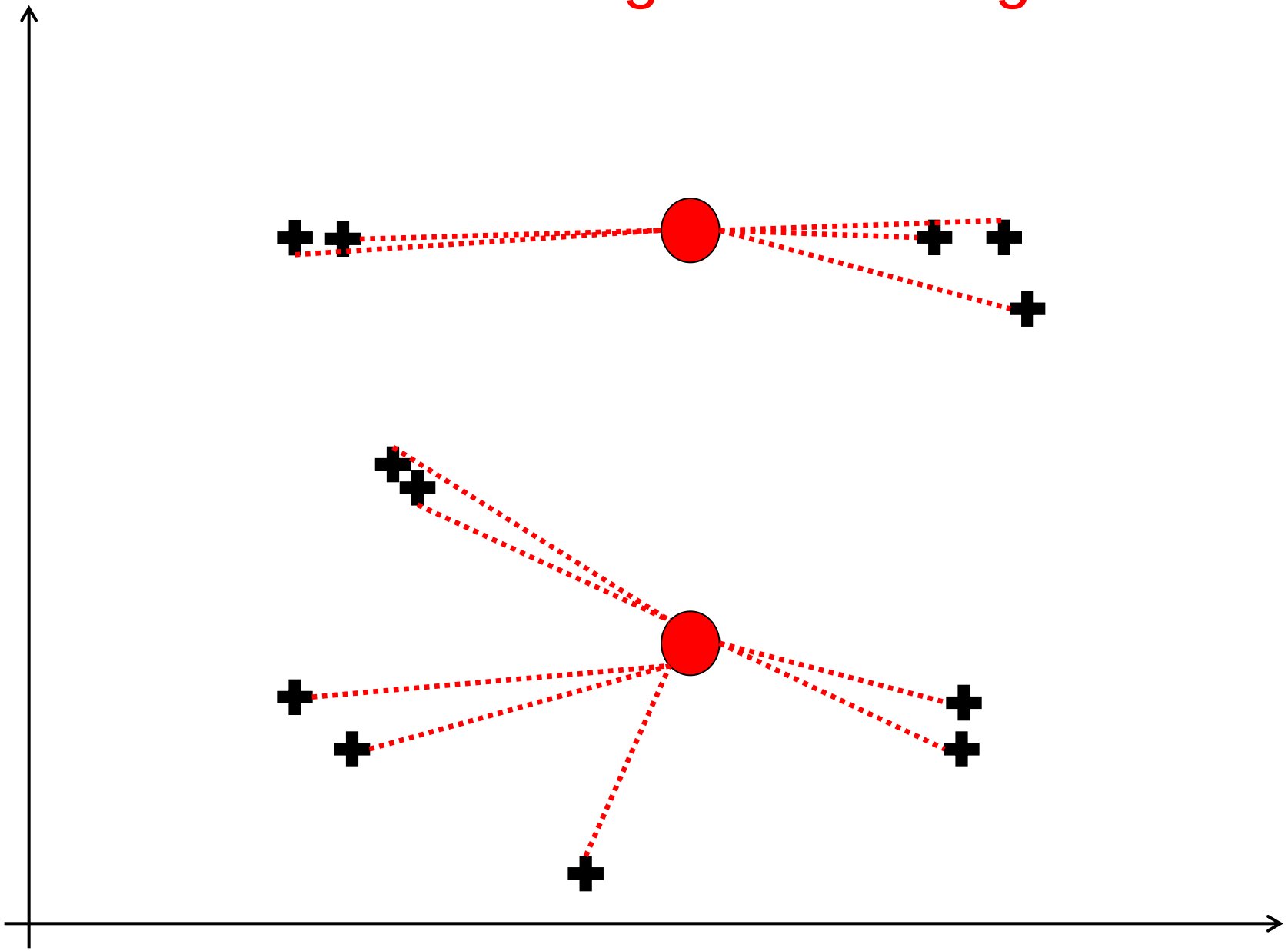
Clustering: Local Minima (2)



Clustering: Overfitting



Clustering: Underfitting



Comments on the *K-Means* Method

- **Strength**: *Relatively efficient*: $O(tkn)$. Typically, k, $t << n$. (n = #objects, k = #clusters, t = #iterations)
- **Weaknesses**
  - Need to specify *k,* the *number* of clusters, in advance
  - Sensitive to noisy data and *outliers*
  - Terminates at a *local* optimum
  - Not suitable to discover clusters with *non-convex shapes*
  - Applicable only when *mean* is defined ─ not for categorical data

Comments on the *K-Means* Method

k-means: choose the result that has minimal error E

Example: Object Hypotheses in Natural Scenes using k-Means

- In a stereo image pair of a scene, pixels can be clustered based on position, disparity, hue and saturation.
- For object segmentation, if two objects are in close proximity, they are likely to be encapsulated by the same segment.
- If we give the information that a segment covers two objects, k-means (k=2) can find a likely split of that segment.
- Then the object modeling loop is resumed with the new hypotheses.

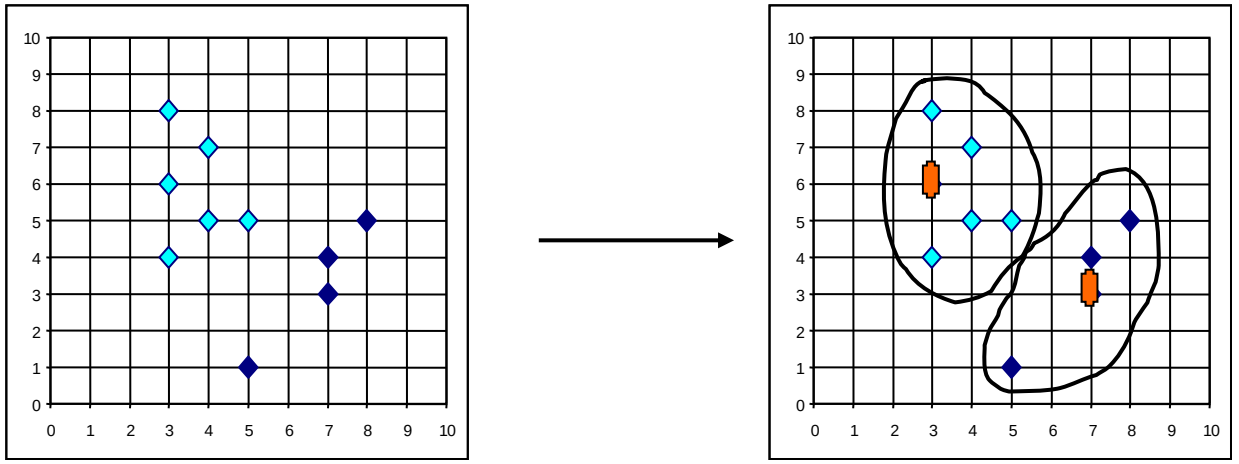
Object Hypotheses Example

Generating Object Hypotheses in Natural Scenes through Human-Robot Interaction

Niklas Bergström, Mårten Björkman, Danica Kragic
CSC/KTH Stockholm, Sweden
IROS '11

Handling Outliers: the K-Medoids Method

- The k-means algorithm is sensitive to outliers!
  - Since an object with an extremely large value may substantially distort the distribution of the data.
- ***K-Medoids***: Instead of taking the *mean* value of the objects in a cluster as a reference point, *medoids* are used, which is the ***most centrally located object*** in a cluster.

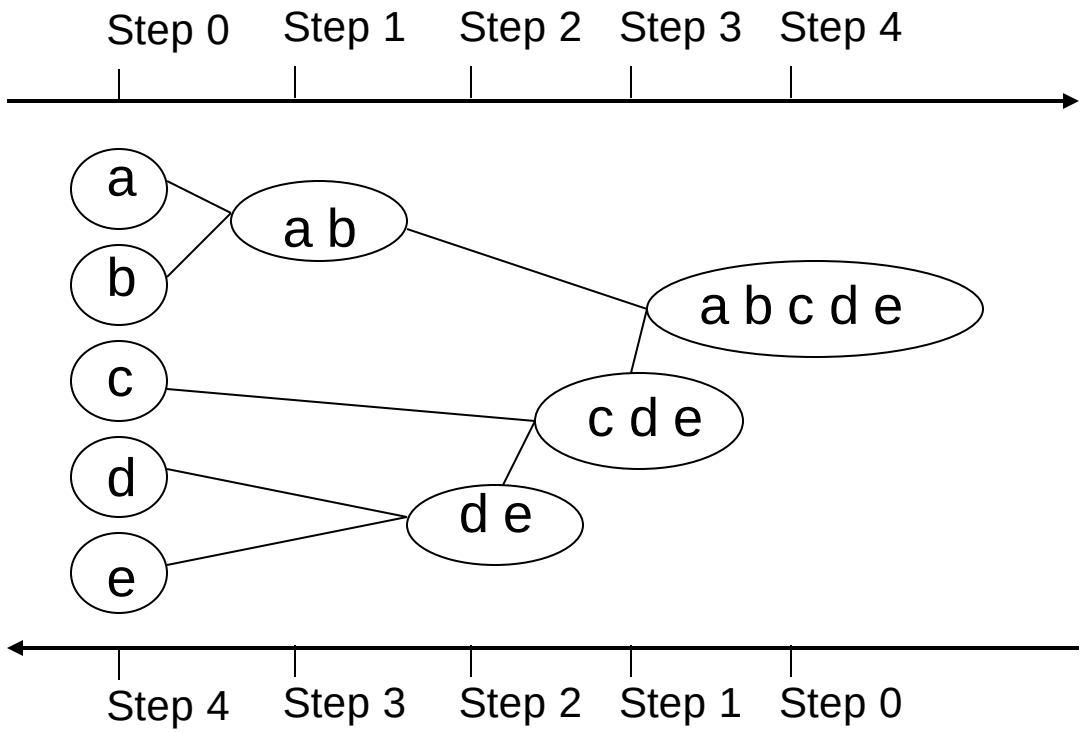


Other Typical Partitioning Clustering Methods

- *K-Medoids* finds *representative* objects (*medoids*) in clusters
  - *PAM* (Partitioning Around Medoids, 1987)
    - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    - does not scale well for large data sets (computational complexity)
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized sampling

Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition

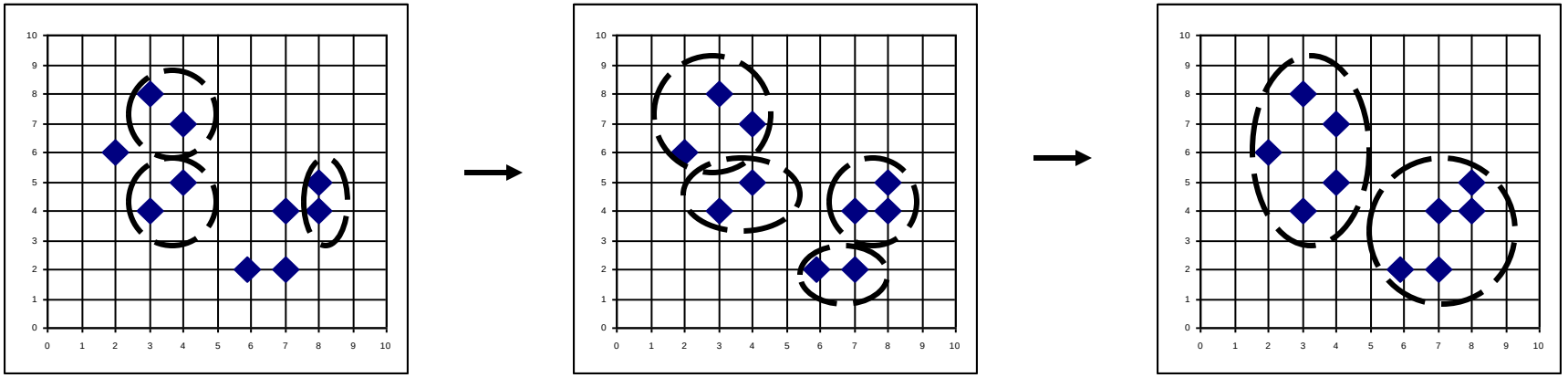


agglomerative
(AGNES)
Bottom up

divisive
(DIANA)
Top down

AGNES (Agglomerative Nesting)

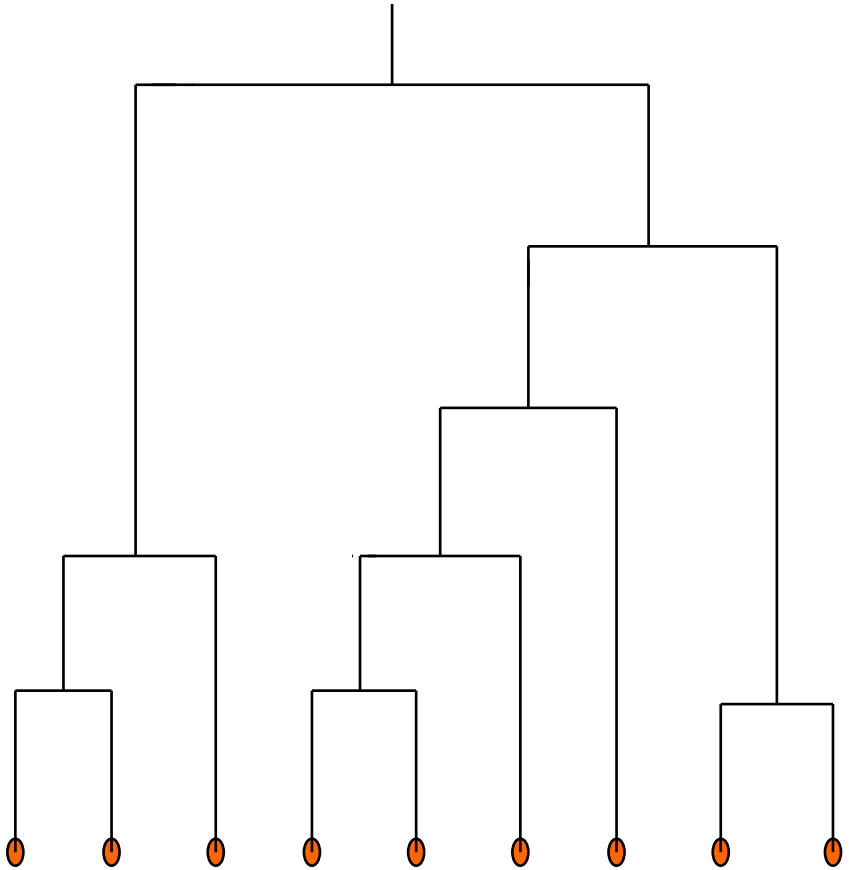
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the ***single-link*** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Dendrogram Shows how Clusters are Merged

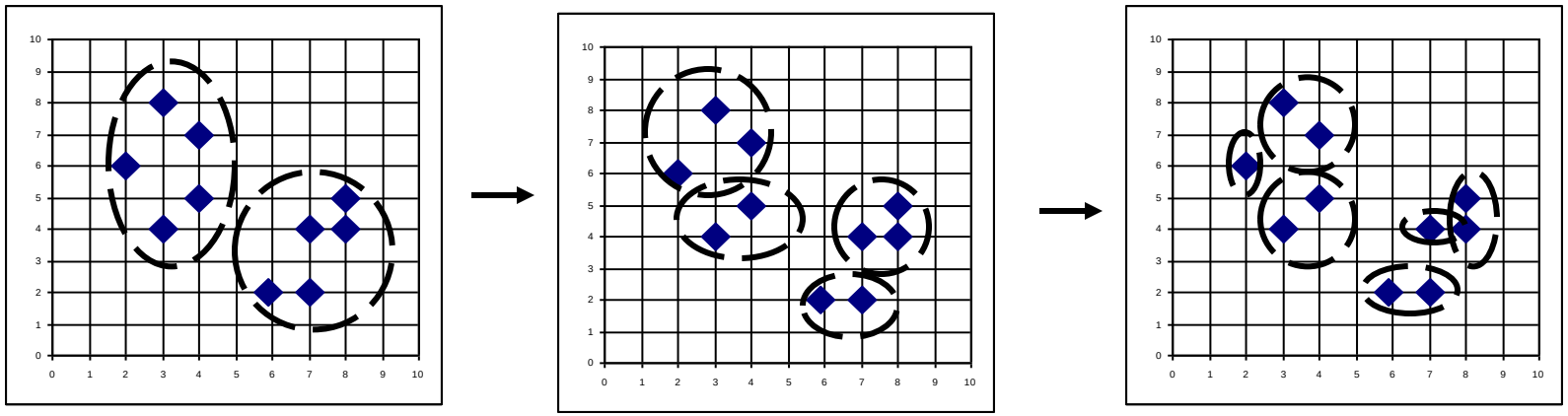
- Decompose data objects into several levels of nested partitioning (***tree*** of clusters), called a ***dendrogram***.

- A ***clustering*** of the data objects is obtained by ***cutting*** the dendrogram at the desired level, then each ***connected component*** forms a cluster.



DIANA (Divisive Analysis)

- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical analysis packages, e.g., Splus
- Inverse order of AGNES: top down
- Eventually each node forms a cluster on its own



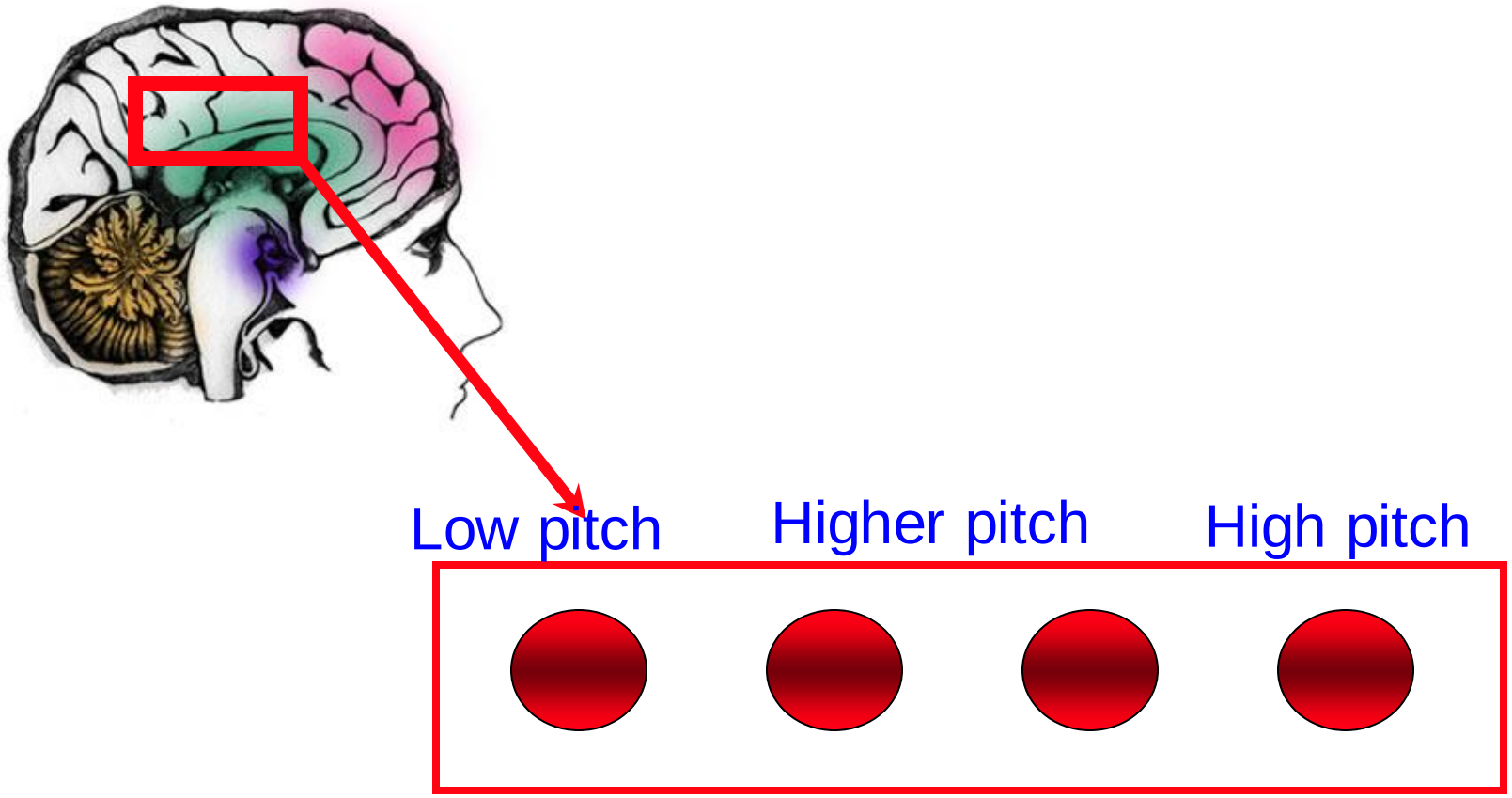
Model-Based Clustering

- 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
  - 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), AutoClass
- Typical neural network methods
  - SOM (Self-Organizing Feature Map)

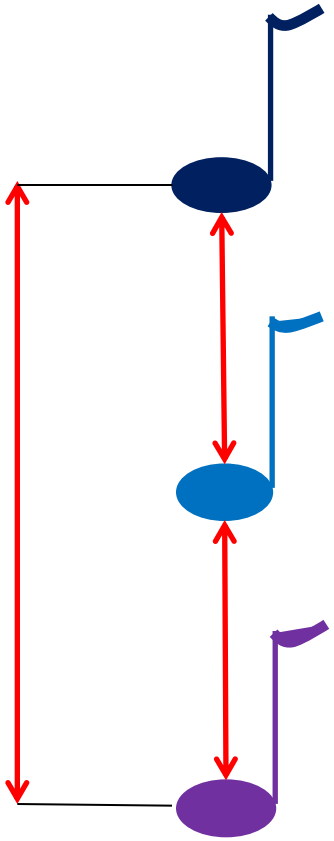
Neural Network Approaches

- 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
- Typical methods
  - SOM (Self-Organizing feature Map)
  - Competitive learning
    - Neurons compete in a "winner-takes-all" fashion for the object currently being presented

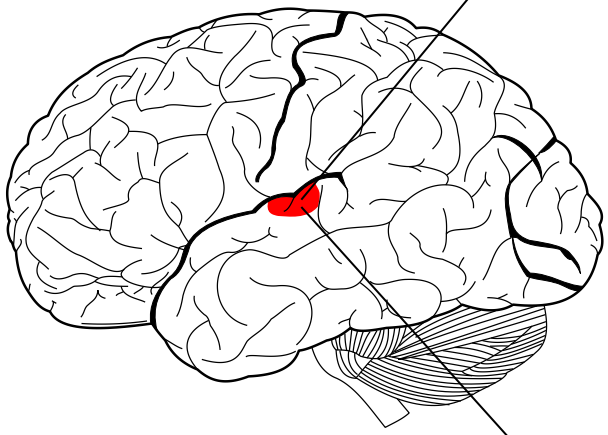
Feature Maps



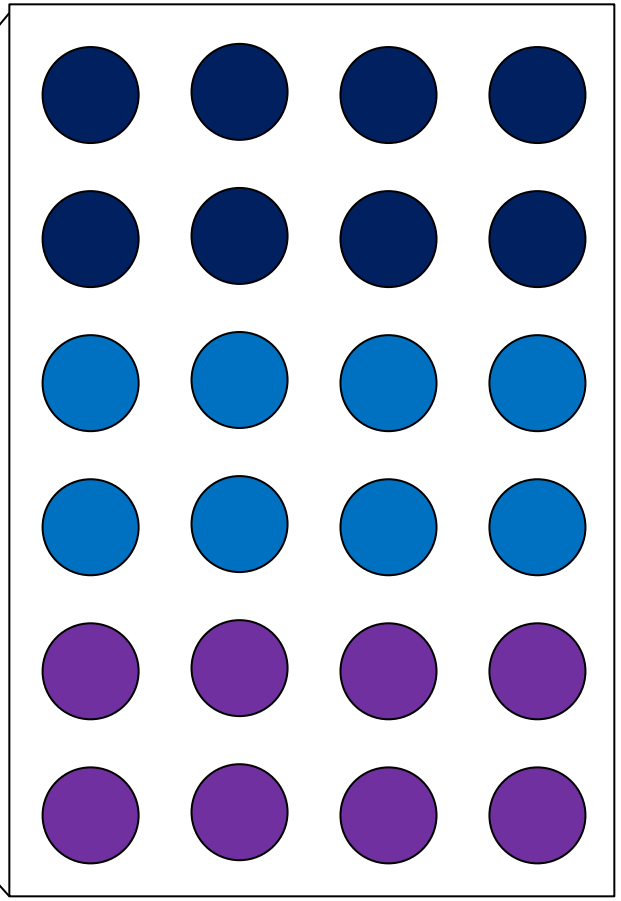
Feature Maps: How are Signals Mapped into the Auditory Cortex?



Different sound signals



Auditory Cortex



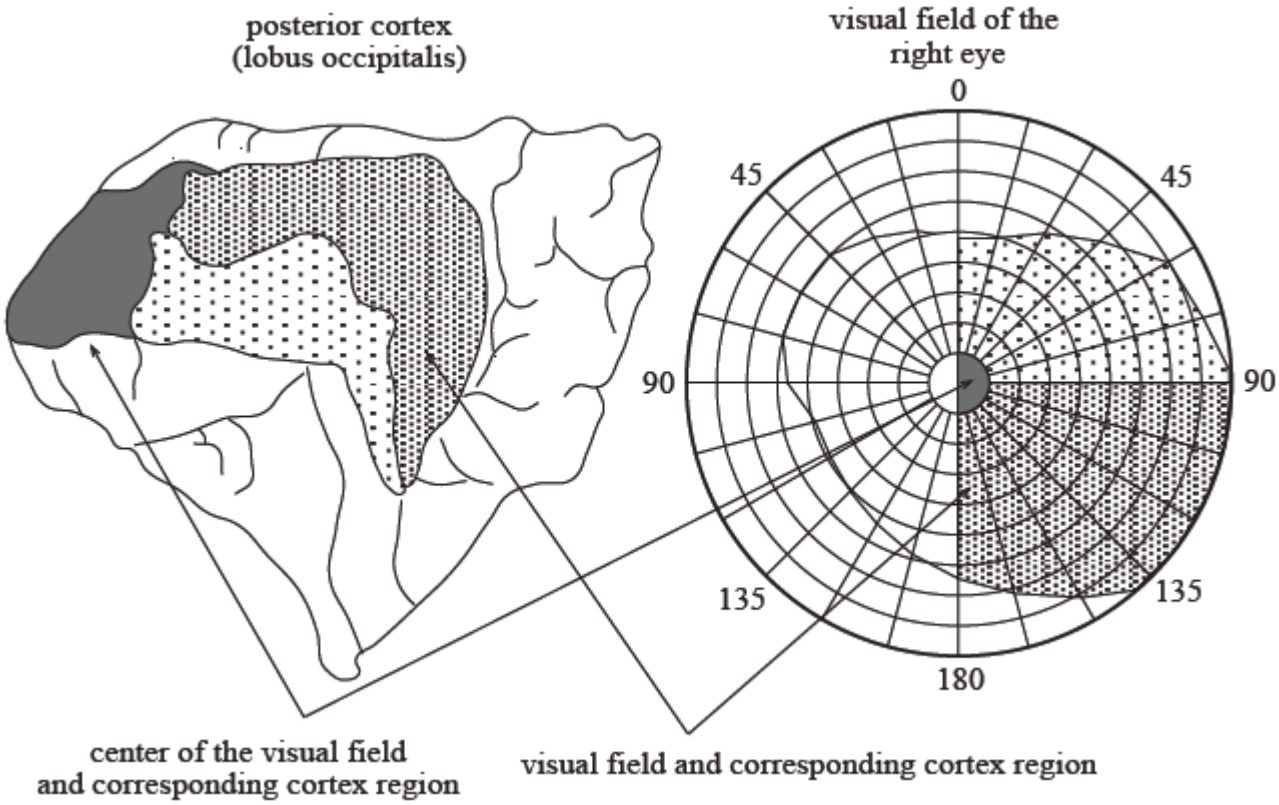
Activated Neurons

Feature Maps

- Sounds that are similar (‘nearby in input space’) excite neurons that are near to each other (in `output space’)
- Sounds that are very different excite neurons that are a long way off
- This is known as ***topology preservation***
- The ordering of the inputs is preserved
  - If possible (perfectly topology-preserving)

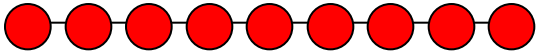
Mapping of visual Field to Cortex

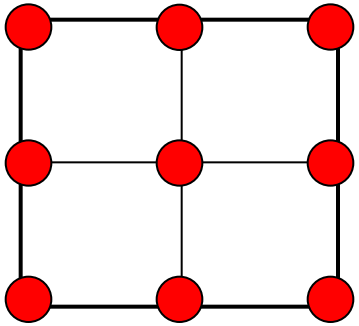
- Neighboring visual fields processed by neighboring cortex regions; fovea is large



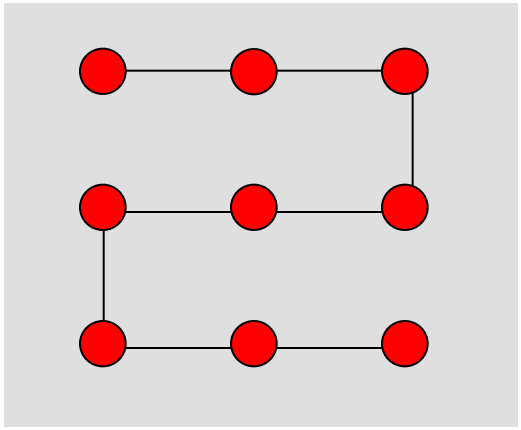
Topology Preservation in 2D

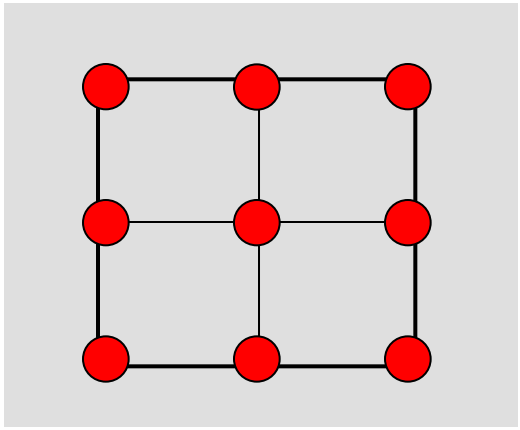
Network topology



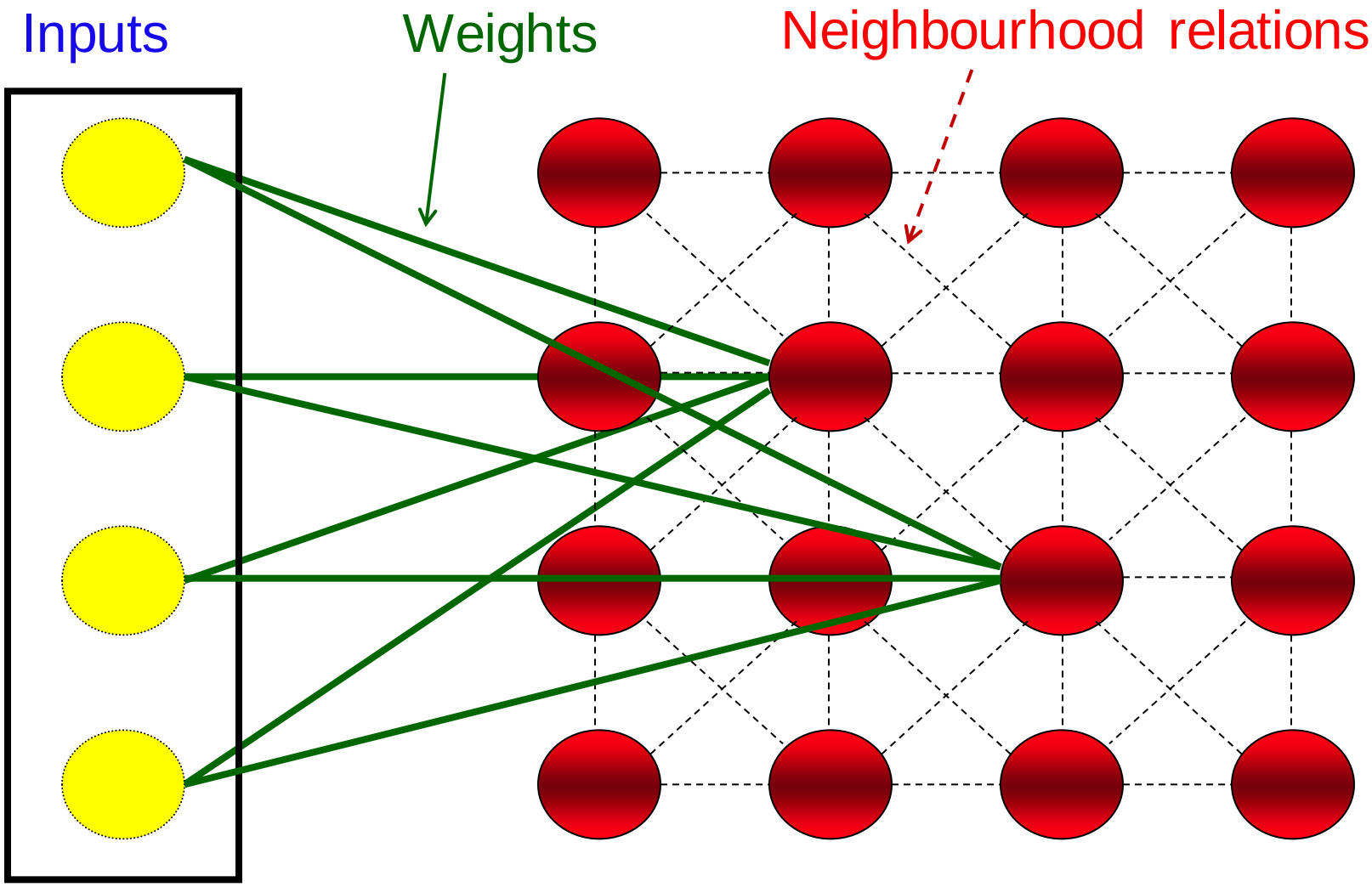


Network representation of input space



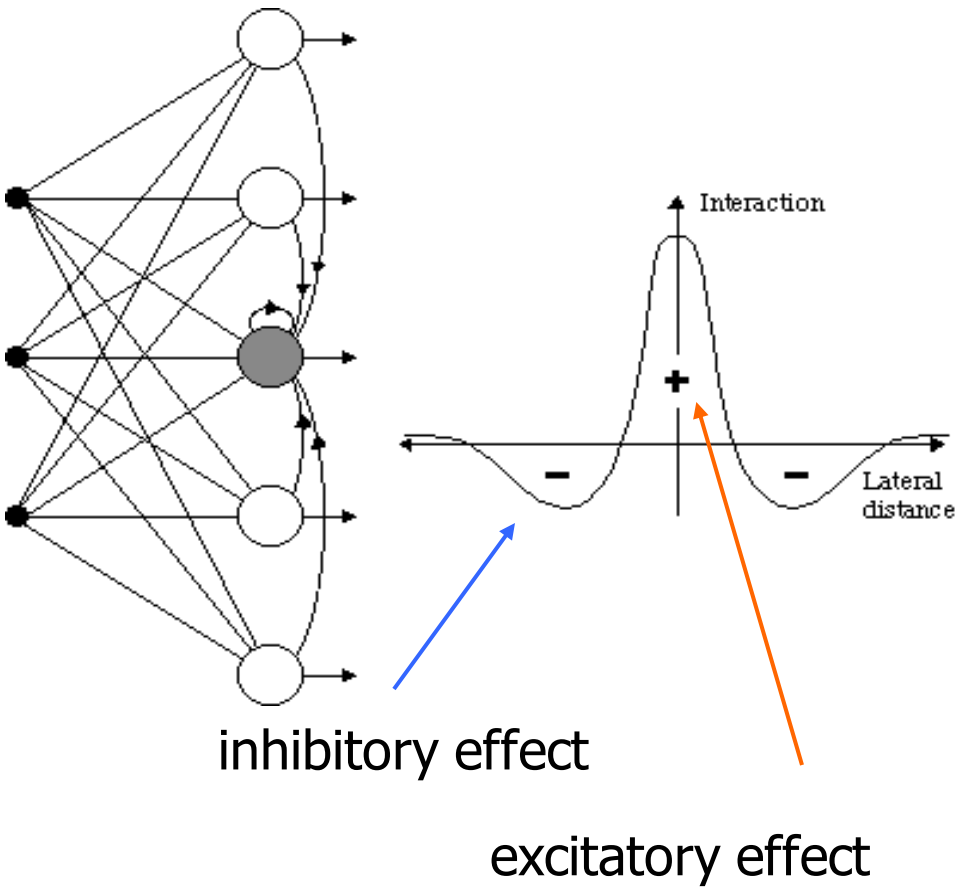


The Self-Organising Map



Mexican hat function

- The Mexican hat function represents the relationship between the distance from the winning neuron and the strength of “connections” within the Kohonen layer
- Near neighbourhood – short range lateral excitation area has strong positive effect
- Remote neighbourhood – has a weak negative inhibitory effect
- → WTA-like behaviour (competitive network!)



Neuron Connections?

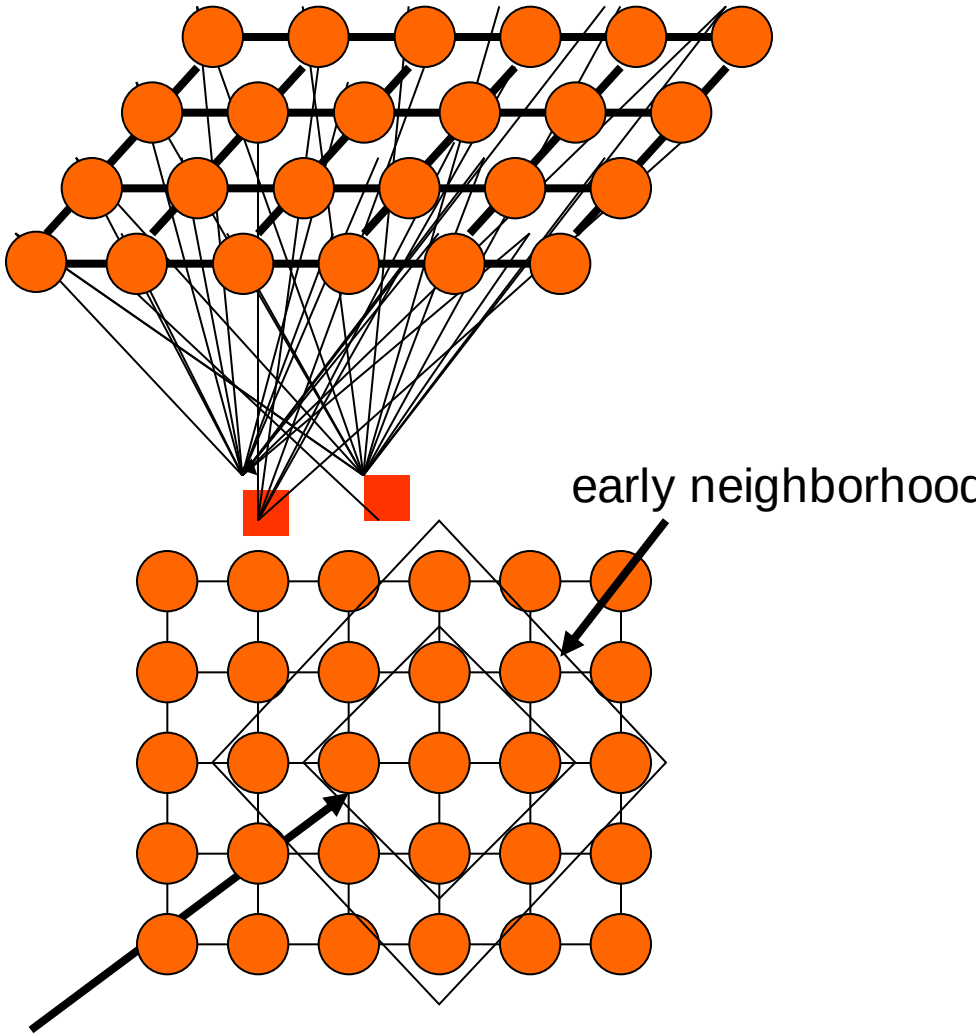
- We do not actually need the inhibitory connections
  - Just use a neighbourhood of positive connections
- How large should this neighbourhood be?
  - At the beginning of learning, the network is unordered
    - Large neighbourhood: each input vector excites many neurons
    - Nearby neurons will learn similarly
    - Their weights will be ordered in input space!
  - Later on, fine-tuning
    - Small neighbourhood: an input vector excites only neurons closeby
    - Neurons can specialize on small regions in input space

Self-Organizing Map (SOM)

- SOMs, also called topological ordered maps, or Kohonen Self-Organizing Feature Map
- It maps the points in a high-dimensional source space into a 1D, 2D (most typical) or 3D target space; distances and proximity relationships (i.e., topology) are preserved as much as possible
- Similar to k-means: cluster centers tend to lie in a low-dimensional manifold in the feature space
- 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

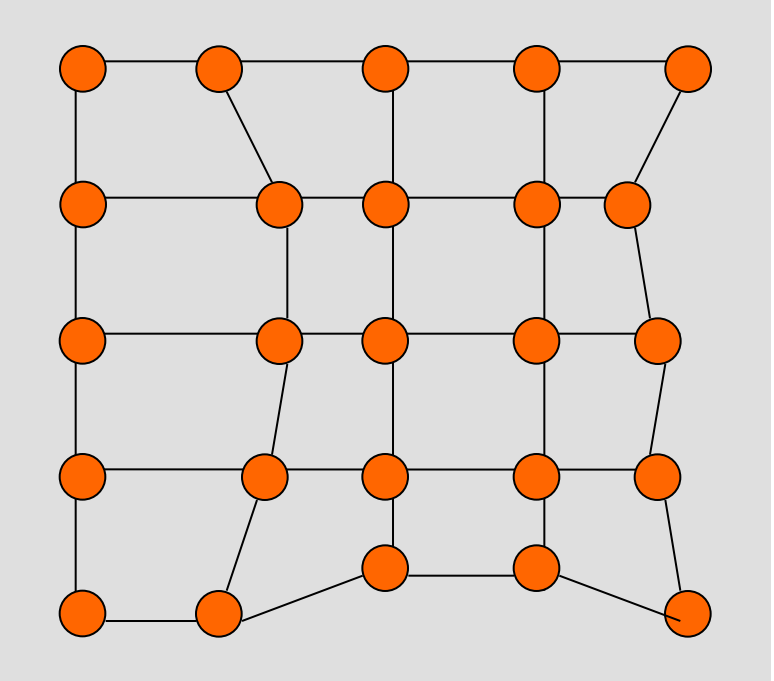
Self organizing maps

- 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 adapts to make their weight vector more similar to the input pattern that caused the activation
- The neurons are moved closer to the input pattern
- The magnitude of the adaptation is controlled via a learning parameter which decays over time



SOM `Cost Function´

- K-means:

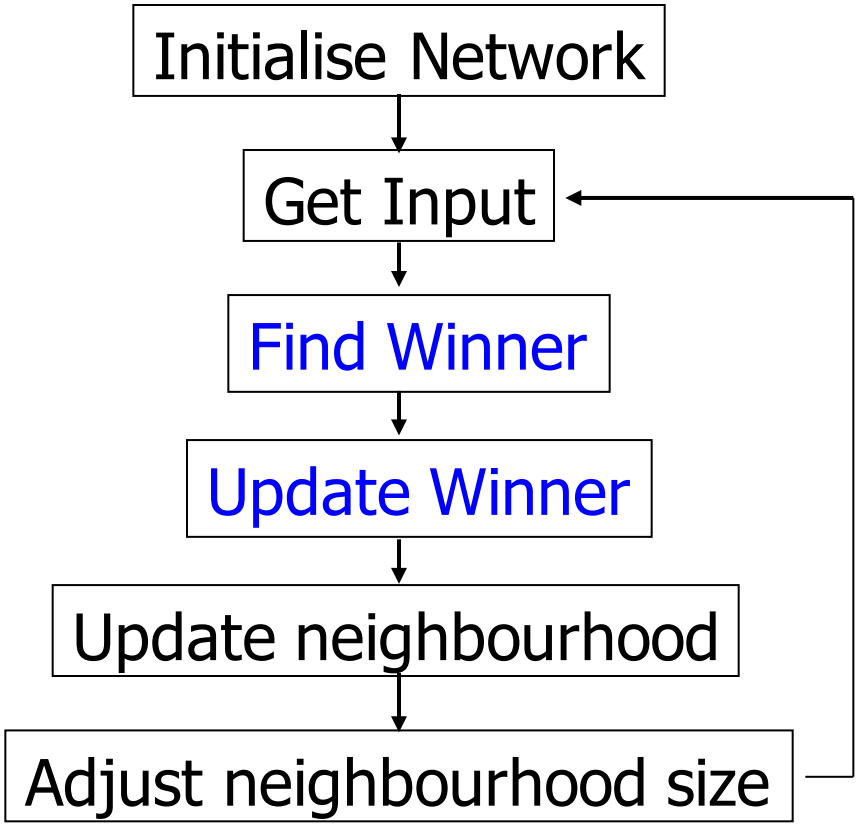
$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (x_p - m_i)^2$$

- SOM:

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

neighbourhood activation function h

SOM Algorithm



The Self-Organising Map Algorithm

- The weight vectors are randomly initialised
- Input vectors are presented 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 $\eta(t)$)

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

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

The Self-Organising Map Algorithm

- The weight vectors are randomly initialised
- Input vectors are presented 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 $\eta(t)$)

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

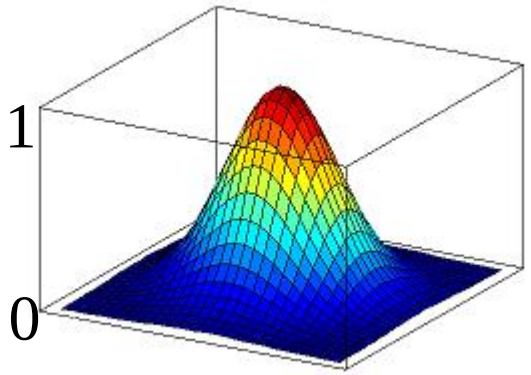
- Over time, the network *self-organises* so that the input topology is preserved

Neighborhood Function Preserves Topology

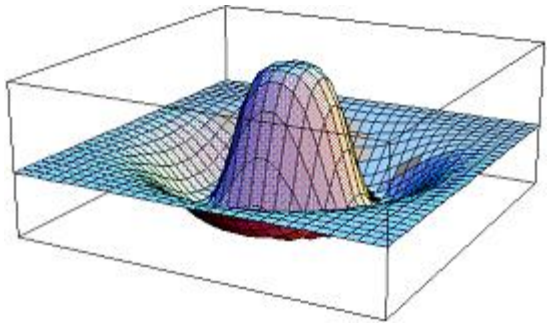
- The neighborhood function $h(n_b,t)$ determines the degree of weight vector change of the neighbors

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

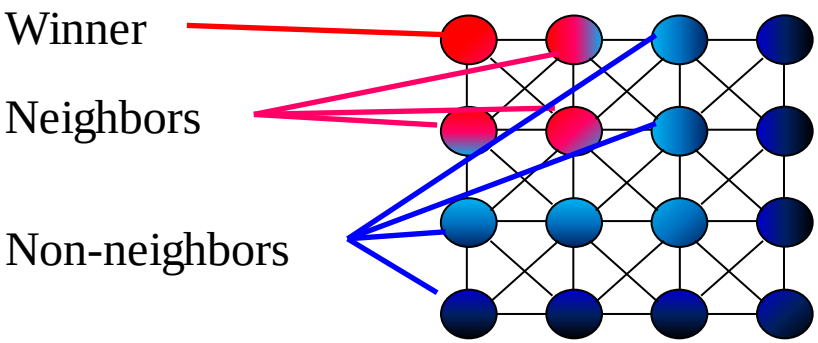
  - Mostly: Gaussian function; rarely: Mexican Hat function
  - Width decreases during training (→ implicit decrease of learning rate)
  - *May* decrease to zero (→ k-means)



Gaussian
(not normalized)



Mexican Hat
(Difference of Gaussian)



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***

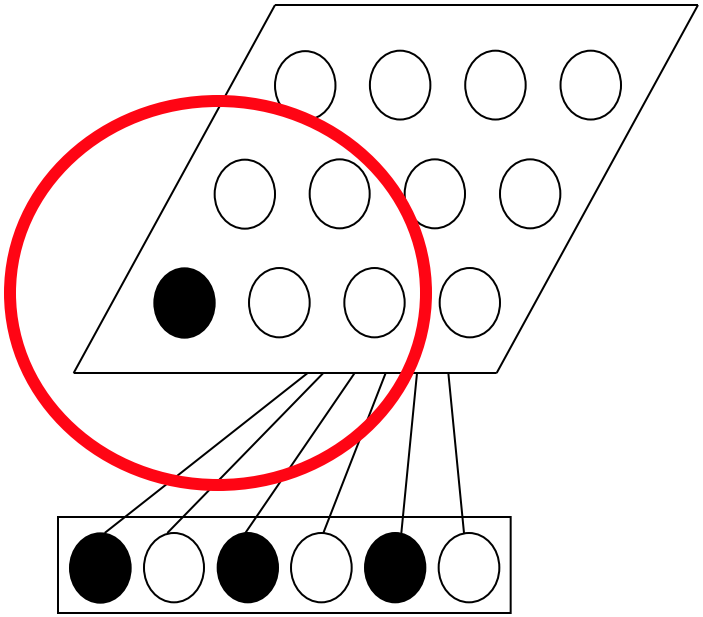
K-means vs. SOM

- "Clusters"
- Clusters independent of each other
  - Clusters indices can be exchanged
- # clusters typically small
- Used for clustering

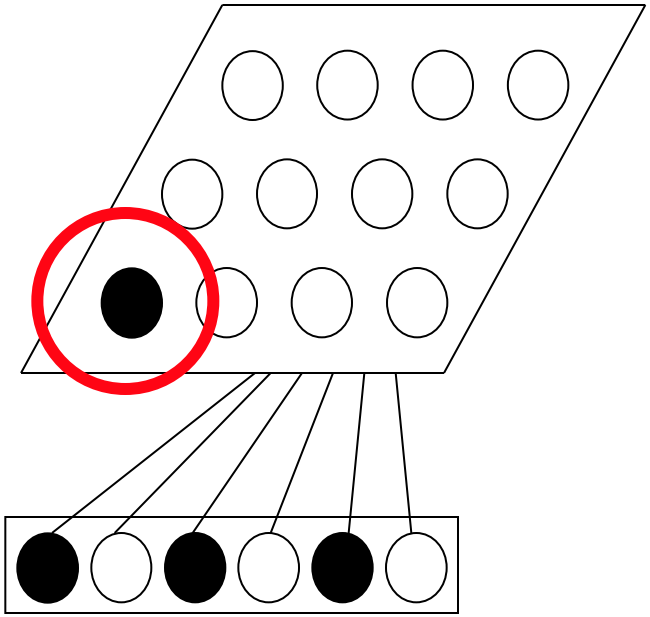
- "Neurons"
- Neurons co-excited by neighbourhood function
  - Neurons arranged on a map in 1D, 2D, …
- # neurons typically large
- Used for mapping

The Self-Organizing Map

Begin of training
(large neighbourhood)

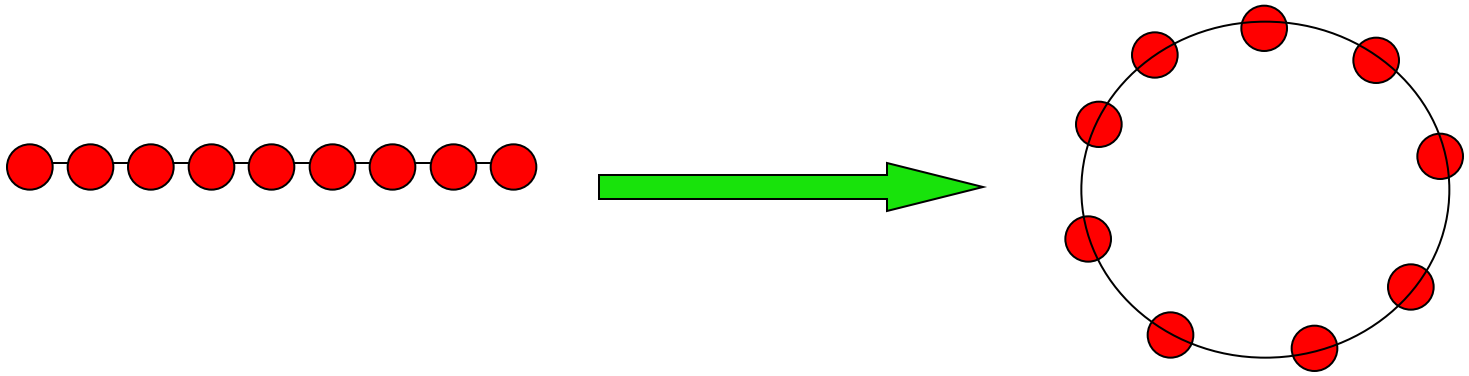


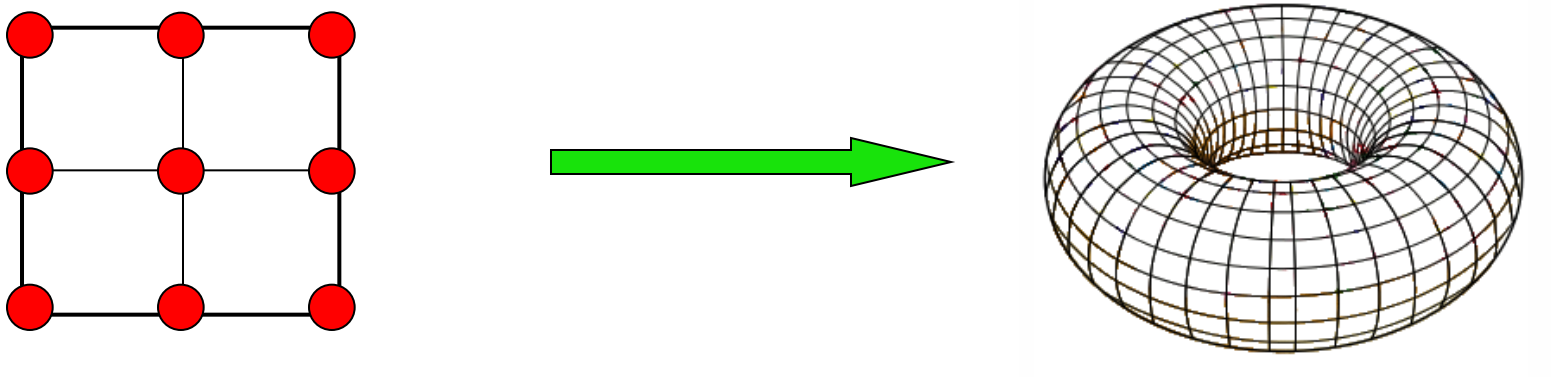
End of training
(small neighbourhood)



If neighbourhood reduced to zero size: → k-means

Boundary Conditions:
No Neurons at the End of the Map



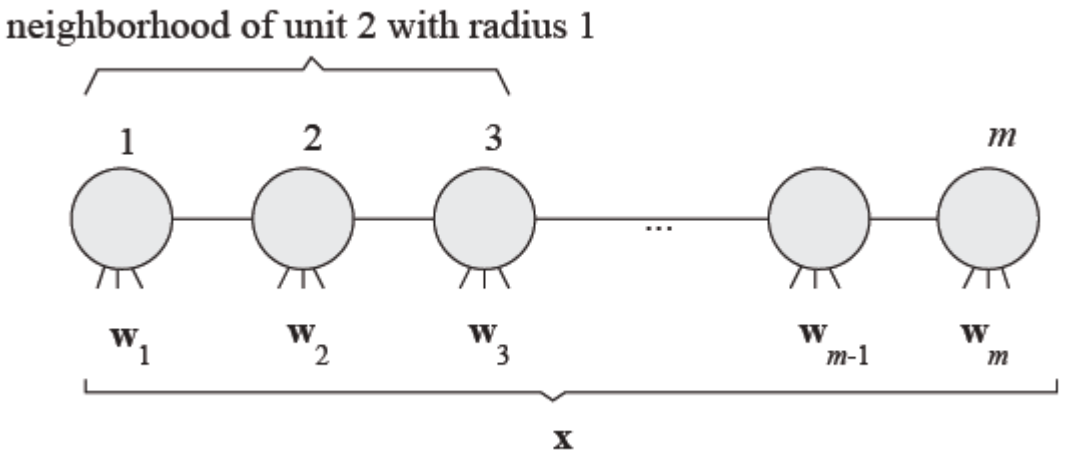


Network Size

- We have to predetermine the network size
- Small network
  - Too much generalization
    → no differentiation
- Large network
  - Each neuron can represent exact features
    → little generalisation
  - Large neighbourhood interaction keeps network “small”, since the parameters are tied to each other
    → this “regularization” allows SOMs to be kept large

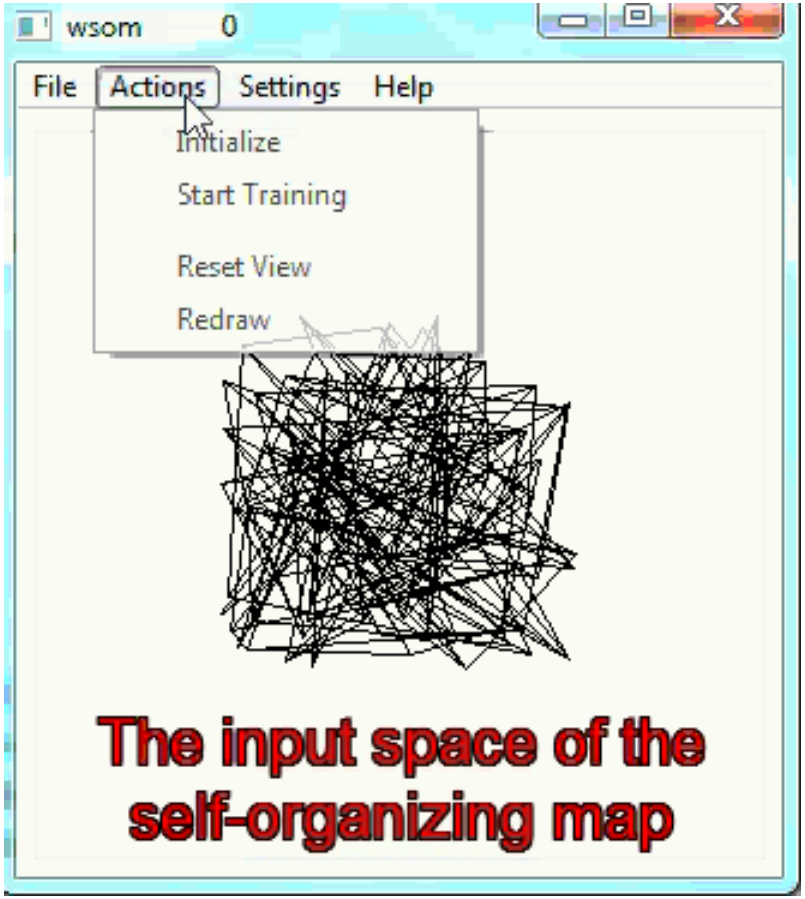
Examples: One-dimensional Lattice of Kohonen Elements

- Units are arranged in sequence
- Each unit learns to specialize for a different region in input space



[Rojas]

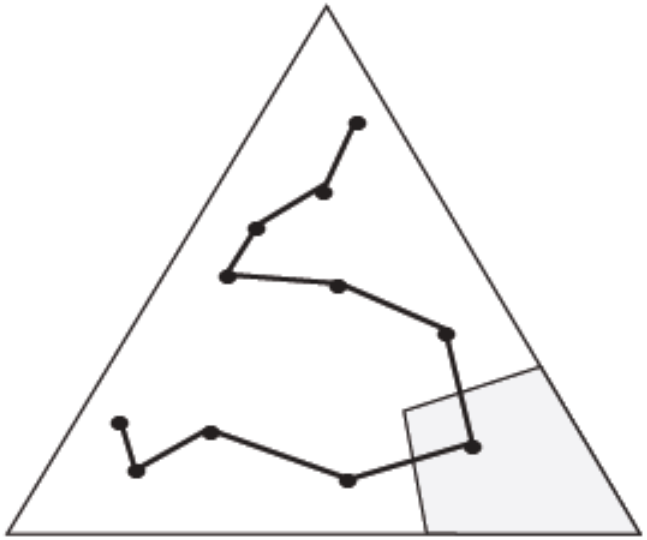
SOM Demo – 2D lattice in 2D / 3D input space



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

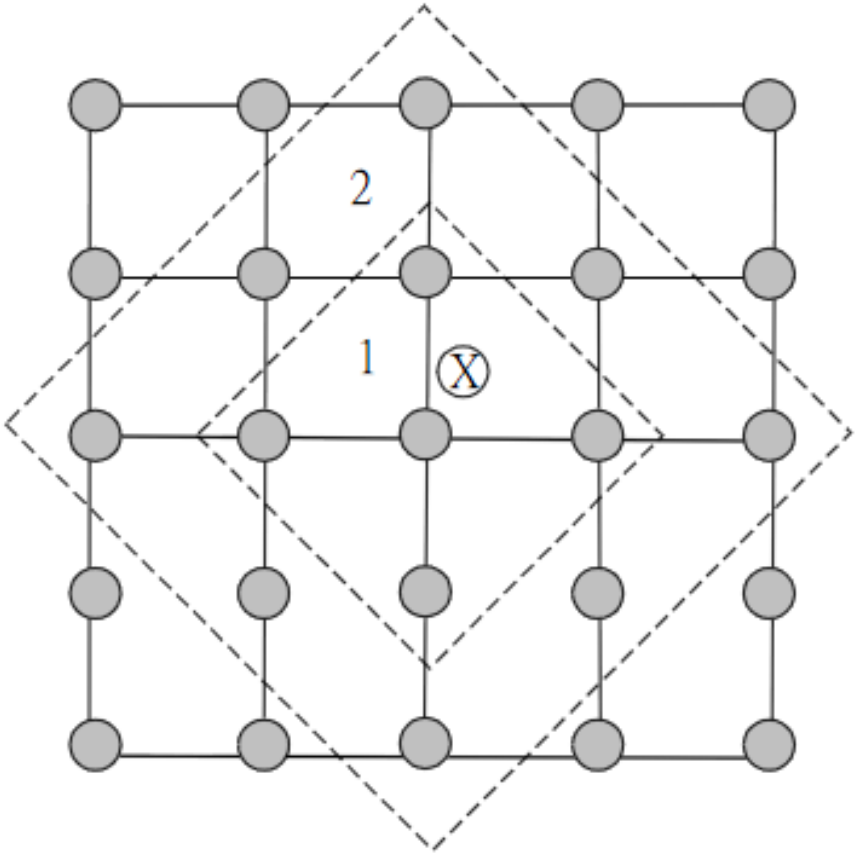
1D Lattice Mapping a 2D Triangular Region

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



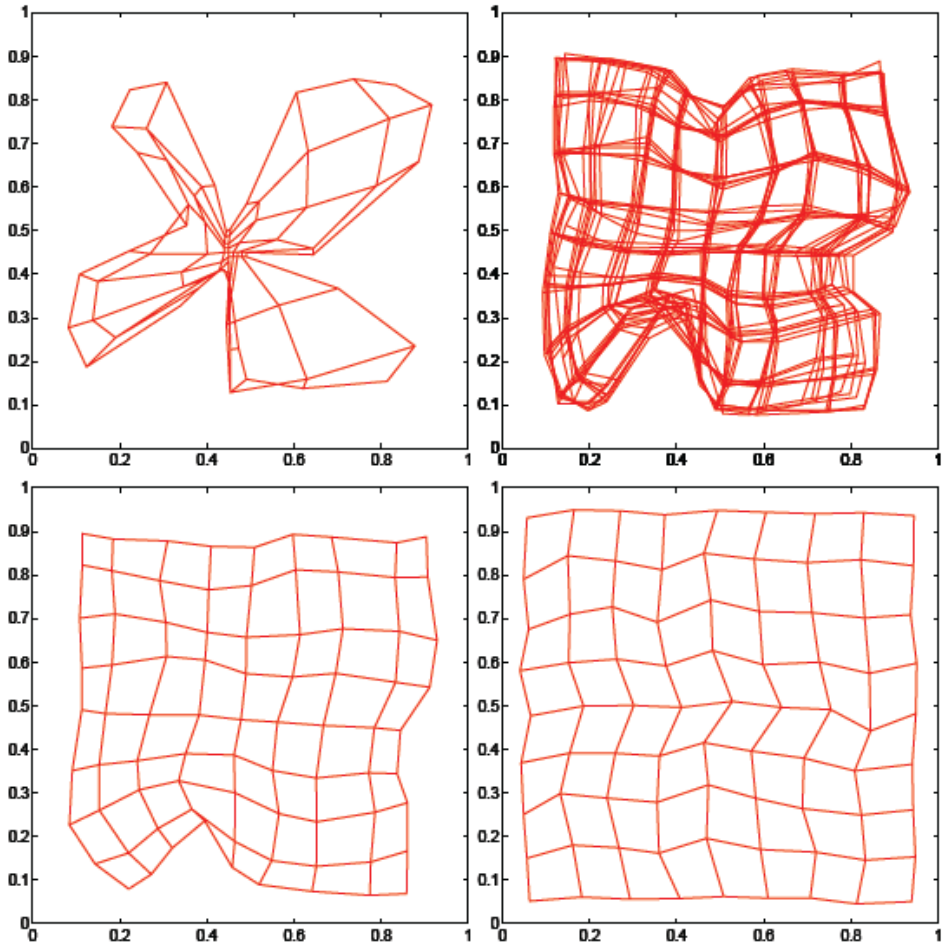
Unit with strongest excitation for shaded region

Example of SOM Neighborhood



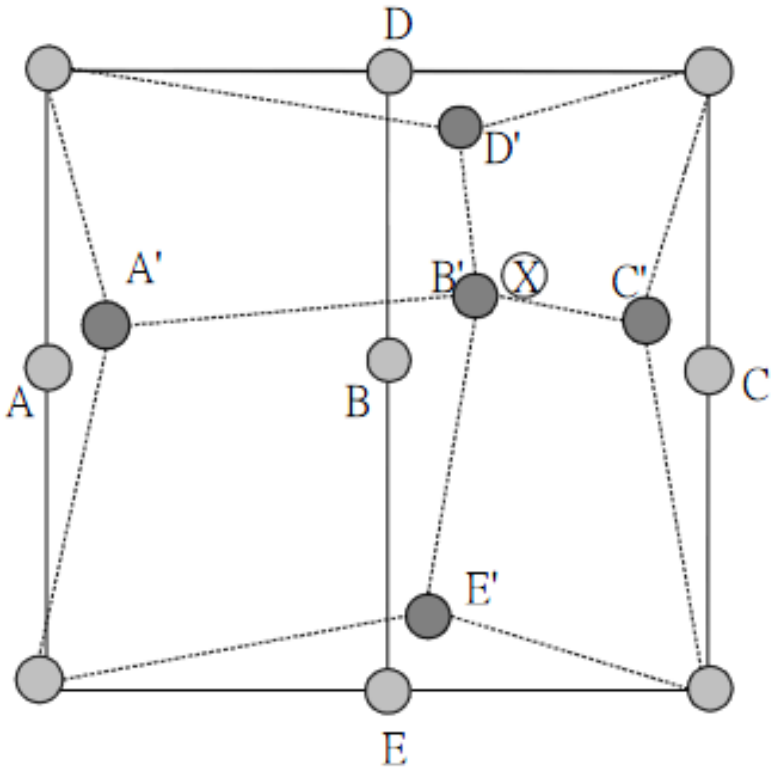
The input vector is represented as X. Units in neighbourhood 1 are more active than those in neighbourhood 2.

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



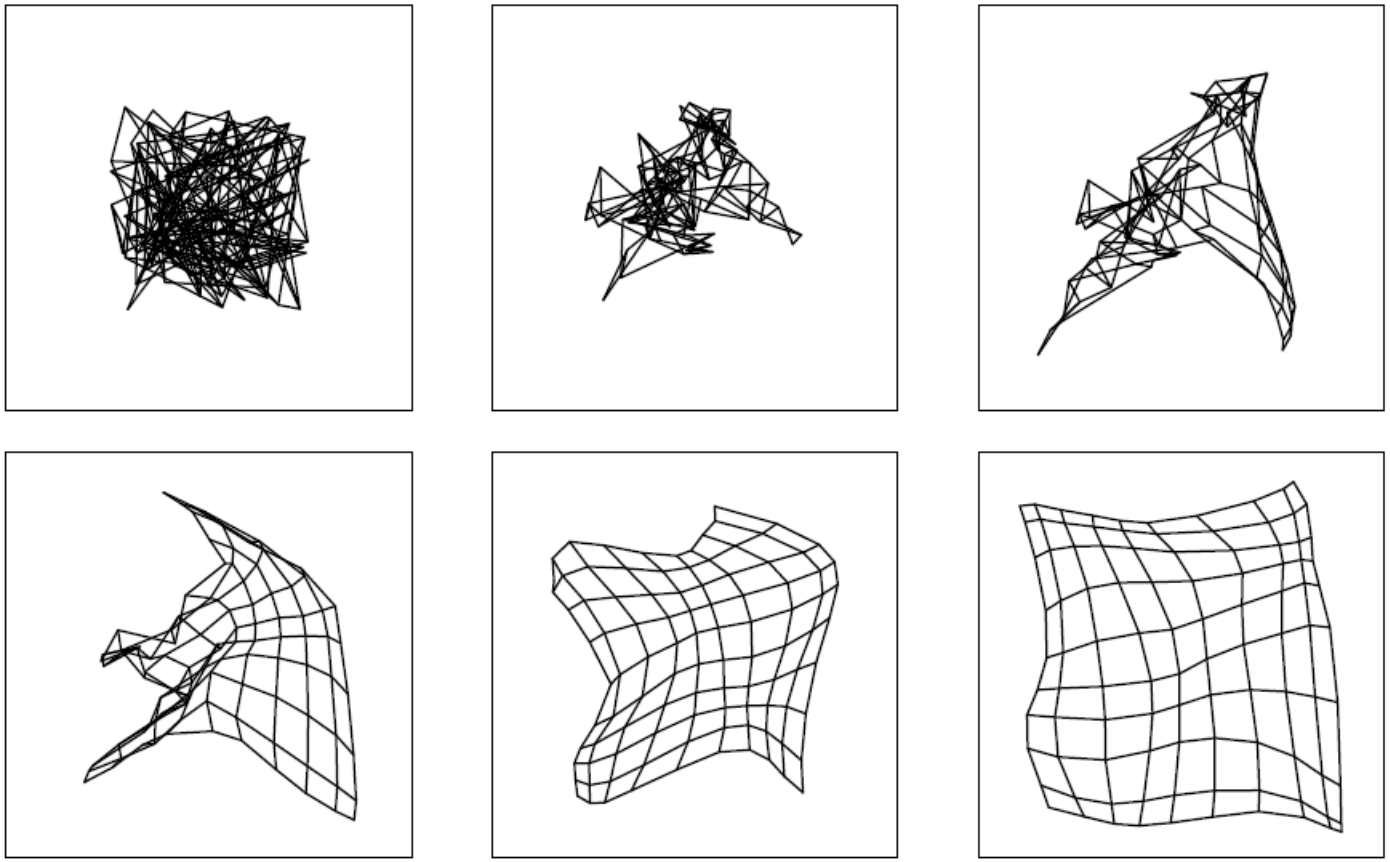
- Upper right with some overlapped learning iterations
- Results for 100, 1000, 5000, 10000 iterations (Kohonen 1984)

Influence by Pre-Defined Topology

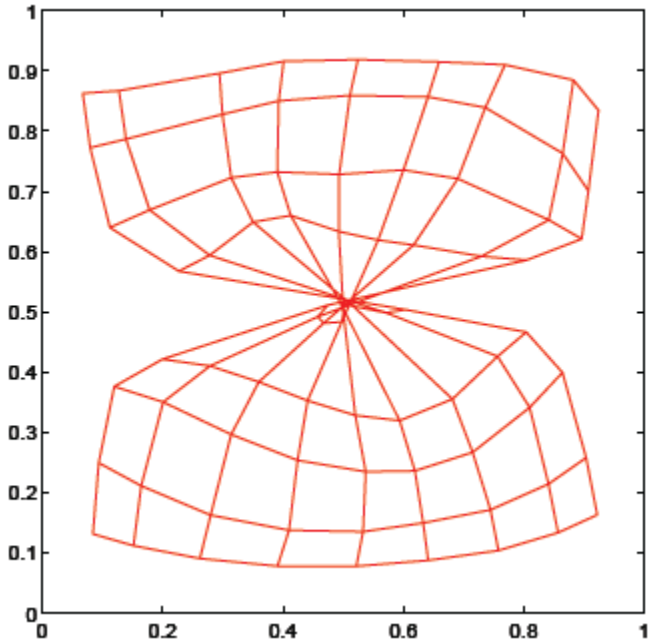


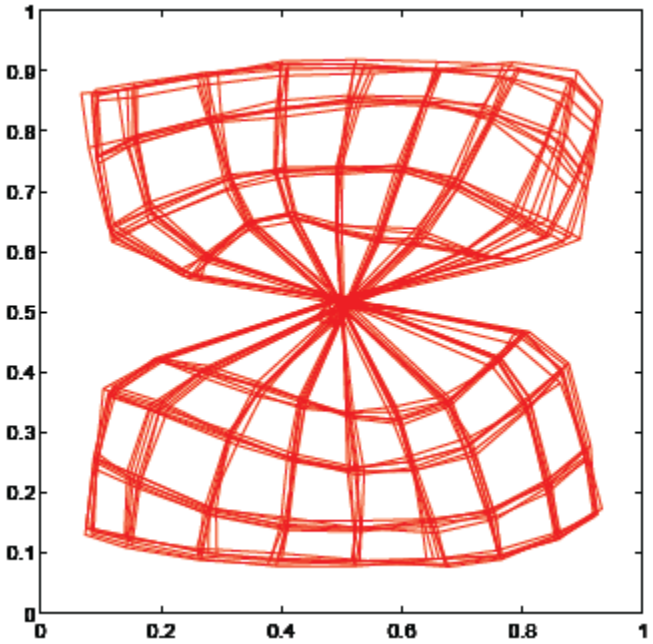
- The winner B and its neighbors such as A, C, D and E move towards the input vector X
- Modified units are shown as dark circles

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



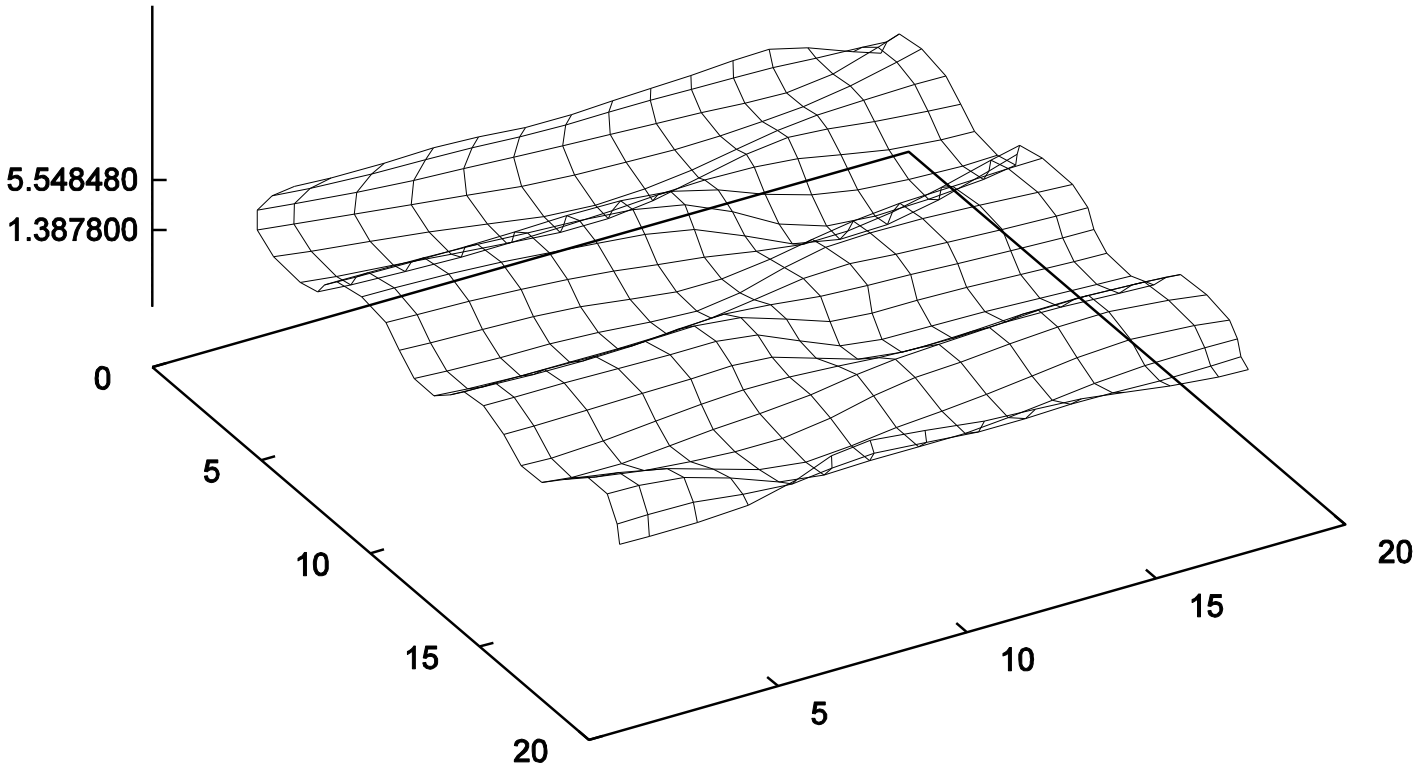
Planar Network with a Knot





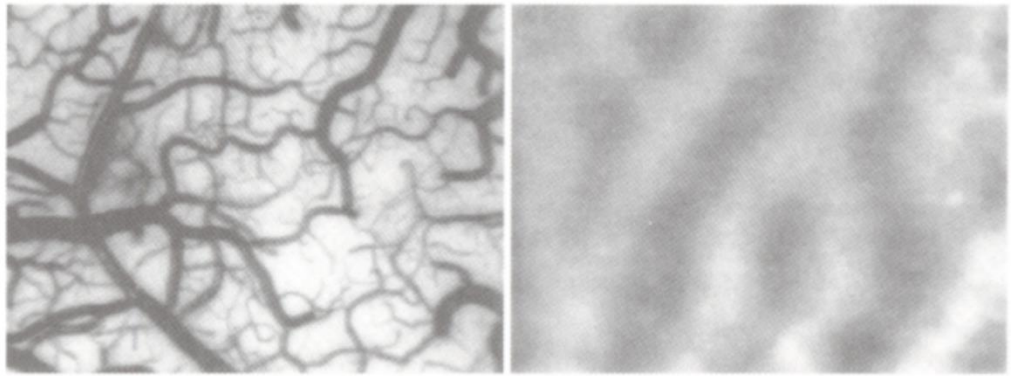
State difficult to correct

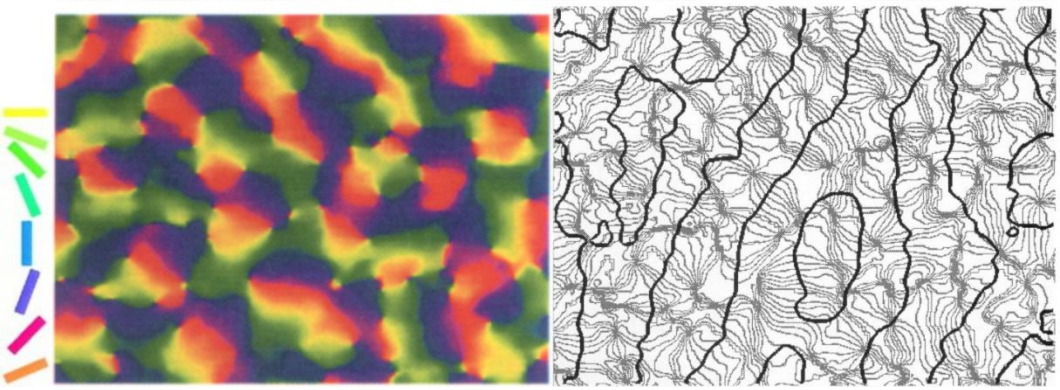
Unfolding of a 2-D Map in a 3-D Data Space

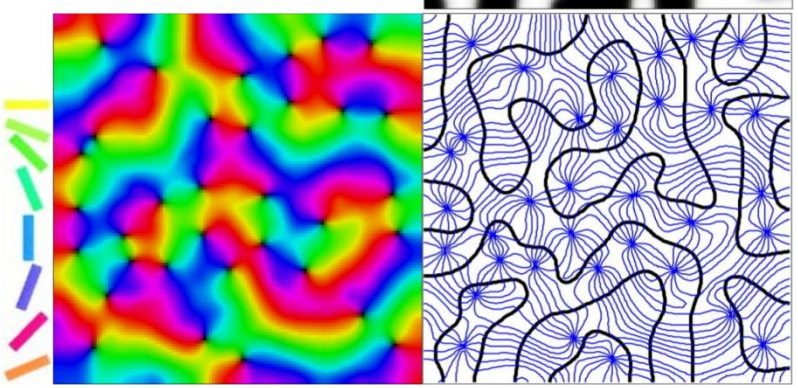


Example Application: V1 Maps

ocular dominance







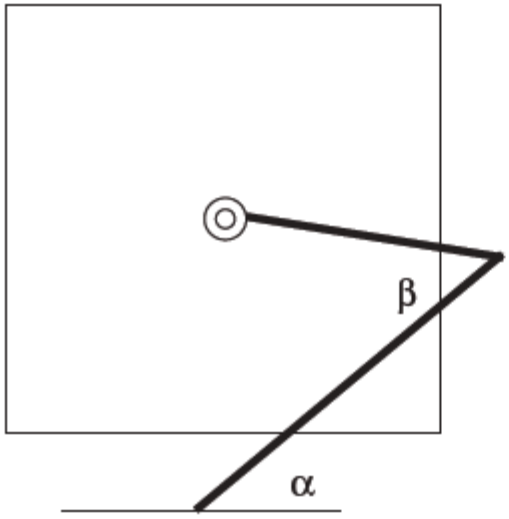
orientation preference

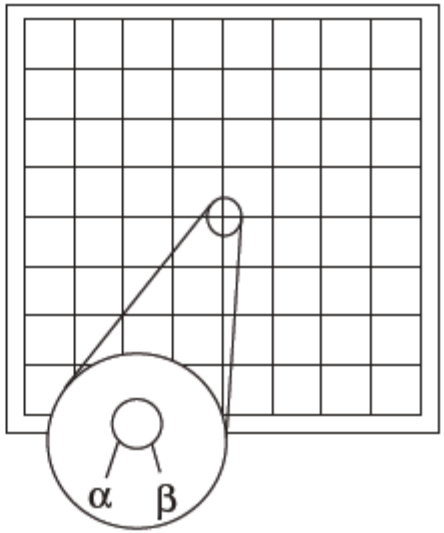
Obermayer, Blasdel. .. Orientation and Ocular Dominance Columns in Monkey .. Jneurosci, 1993

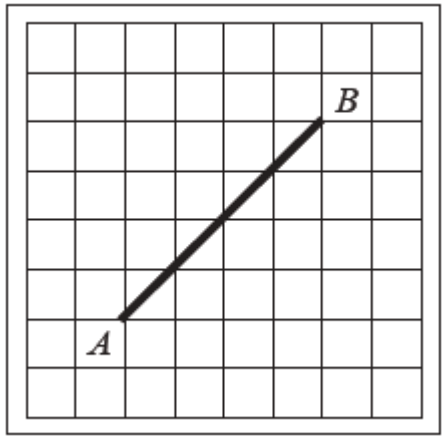
Goodhill . Theoretical Modelling to .. Neural Map Development. Neuron, 2007

Learning Simple Inverse Kinematics

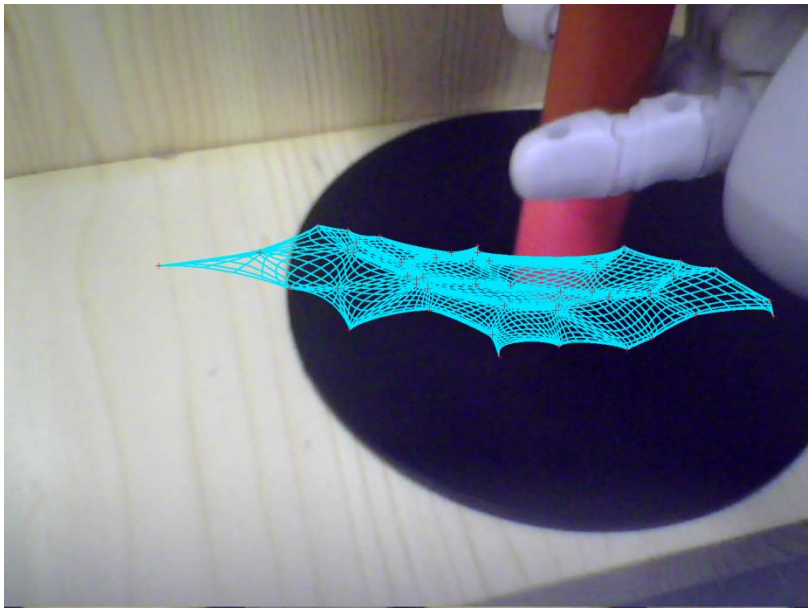
- Mapping the configuration space of a robot arm using 2-dimensional network
- For one point only one parameter combination; many paths from A to B

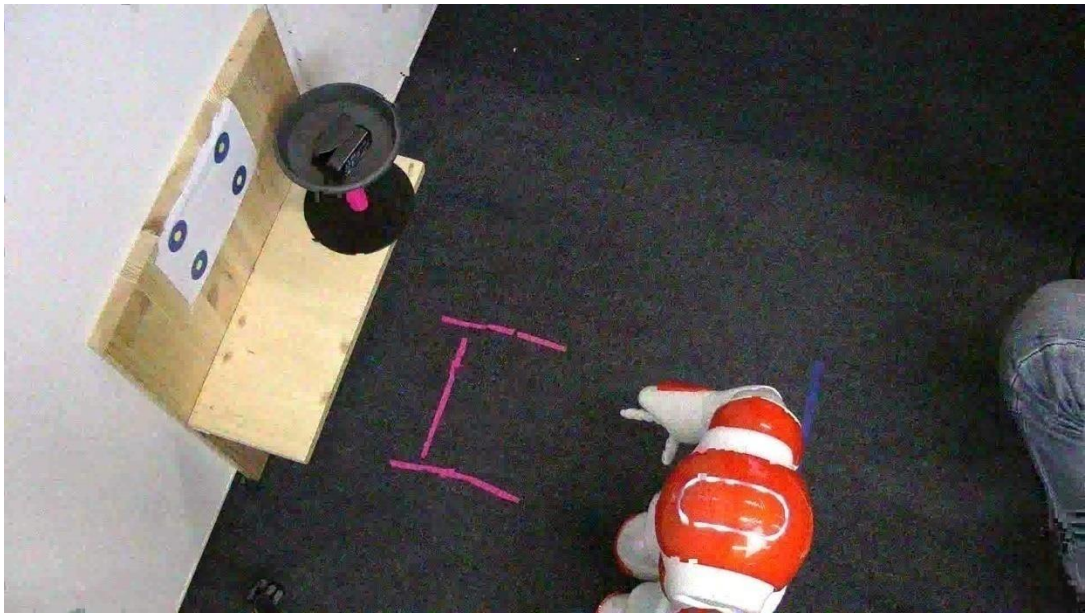






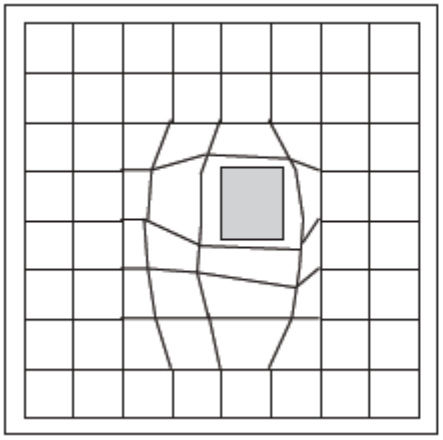
Learning Simple Inverse Kinematics

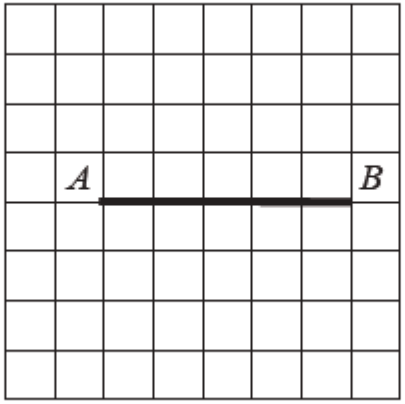


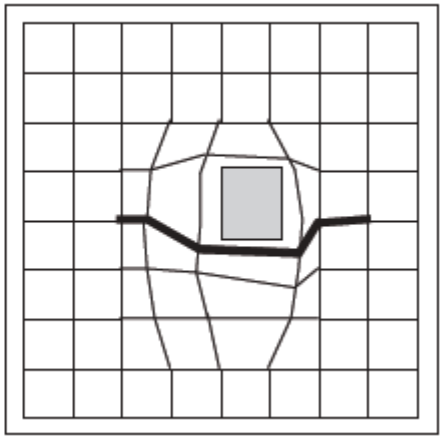


Learning Obstacles in Work Area

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





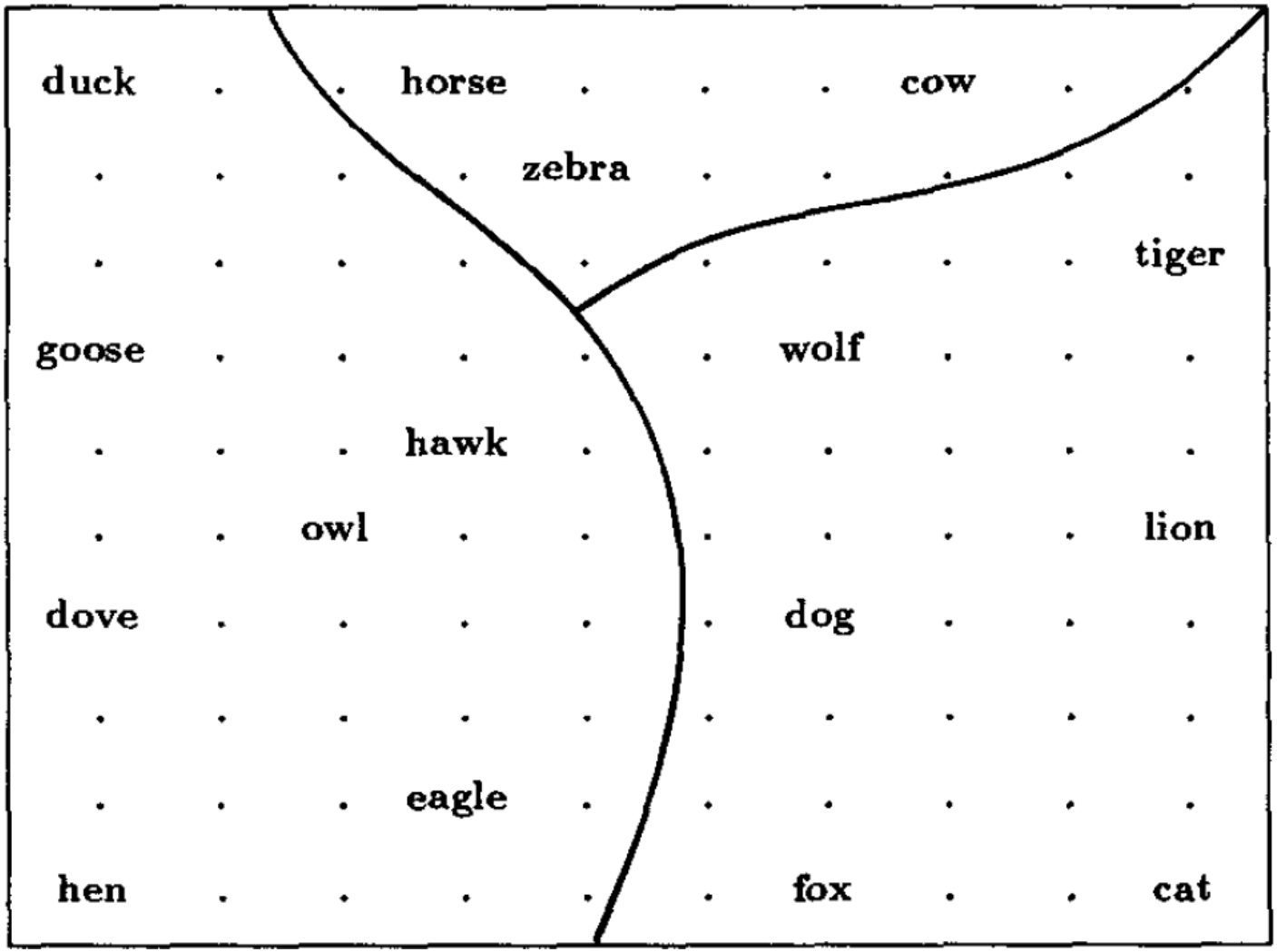


Semantic Map – Input data

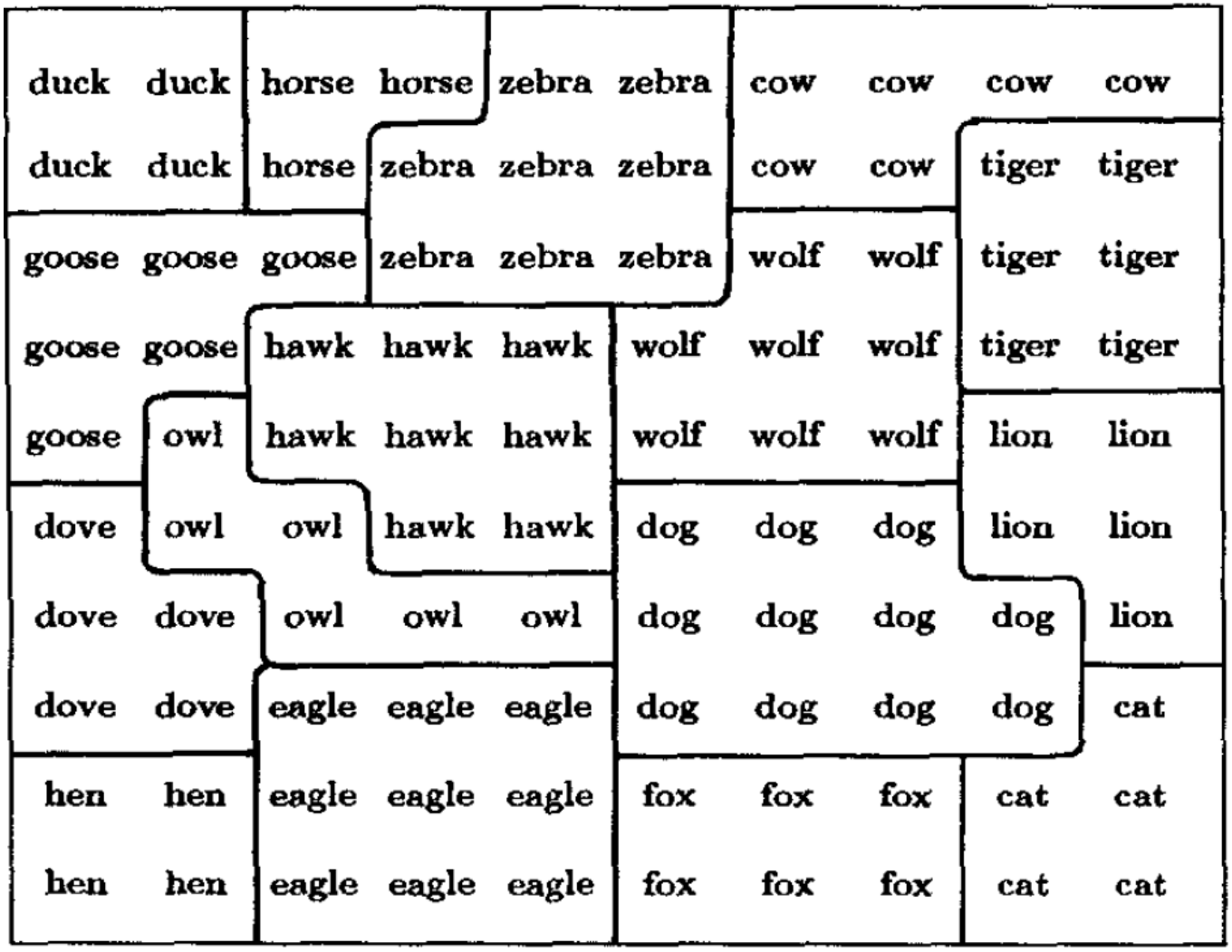
attributes of the inputs

		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



Semantic Map - Results



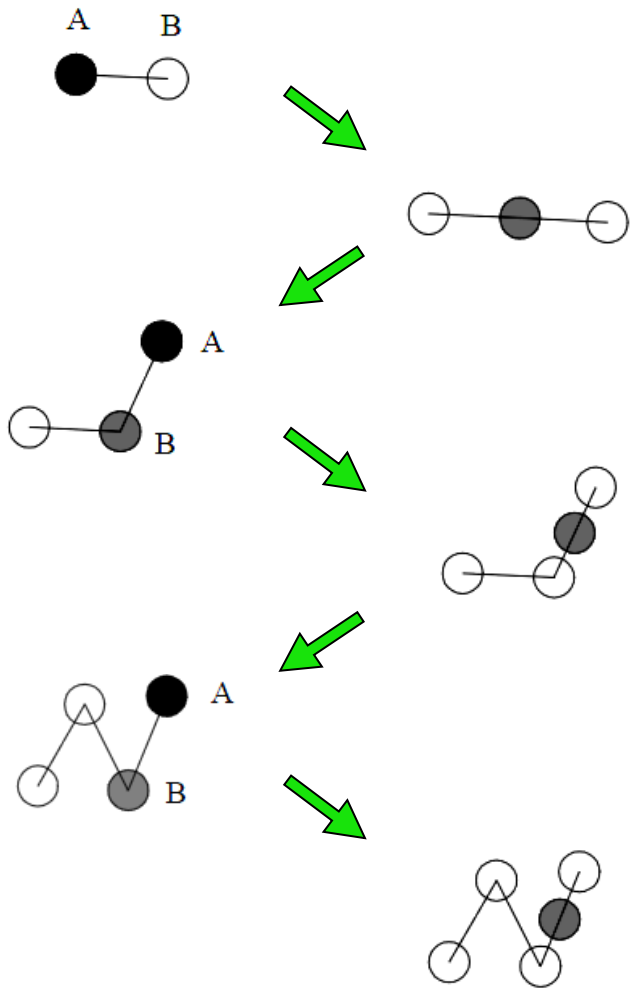
Existing Neural Clustering models

- ***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.

Growing Neural Gas – GNG
(Fritzke)



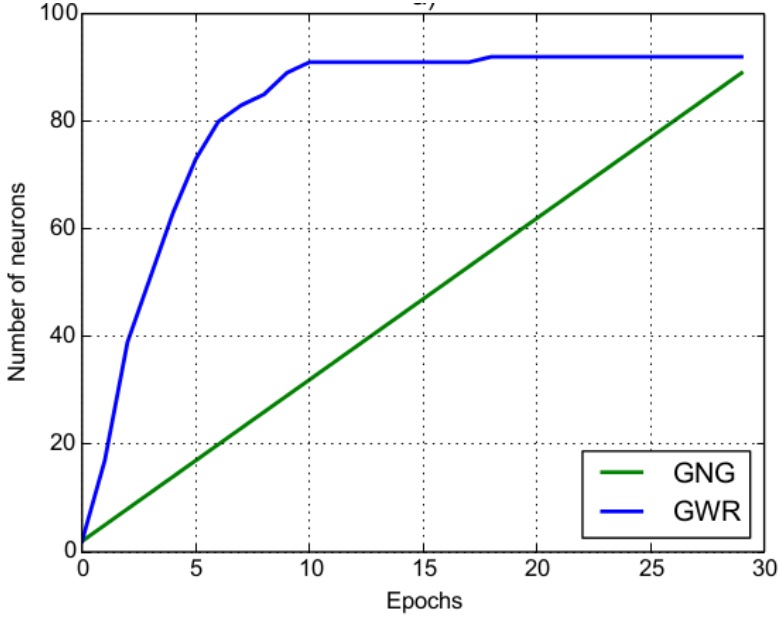
- GNG adds unit after every pre-defined period.
- Units are represented as circles.
- Circle A indicates ***unit with biggest error***
- Circle B indicates ***neighbor with biggest error*** for Circle *A.*
- The grey circle is the ***new unit*** at each stage.

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

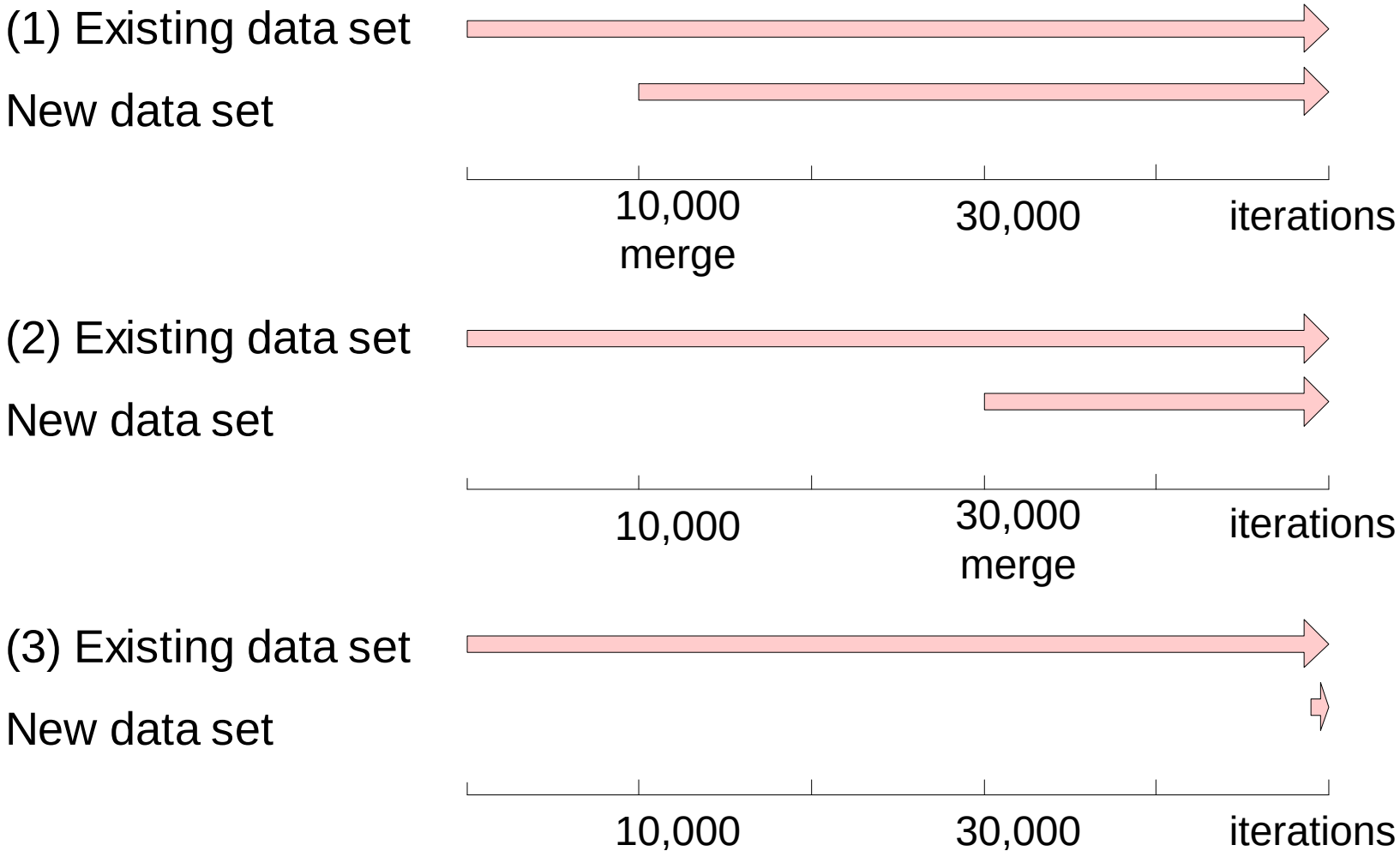
- **GNG**: network grows with *time*
- **GWR**: network grows *when required*

→ strong neurogenesis at early stages

- Each neuron has a *firing counter* (how often is it the winner)
  - Many counts → lower learning rate to help convergence
  - 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)



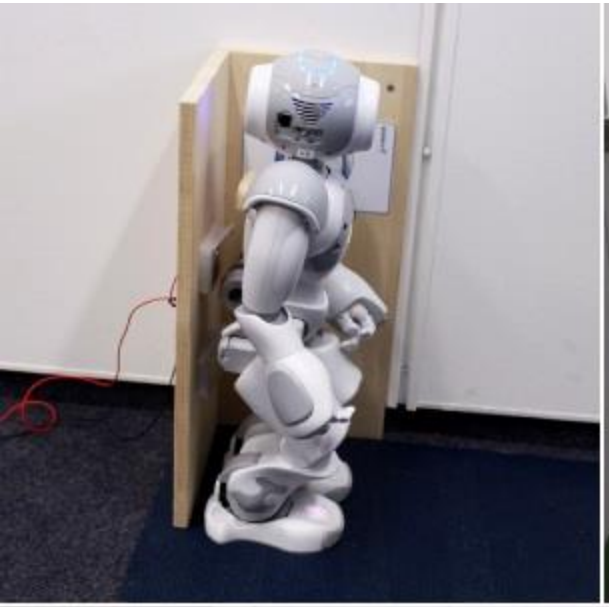
Dynamical SOM for Dynamical Knowledge Acquisition Over Time

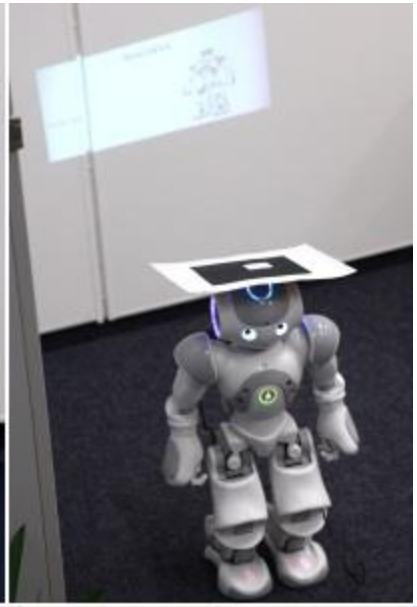


Shortcomings of Static Models in a Non-stationary Environment

- 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)

Room Mapping via GNG (KT Lab)

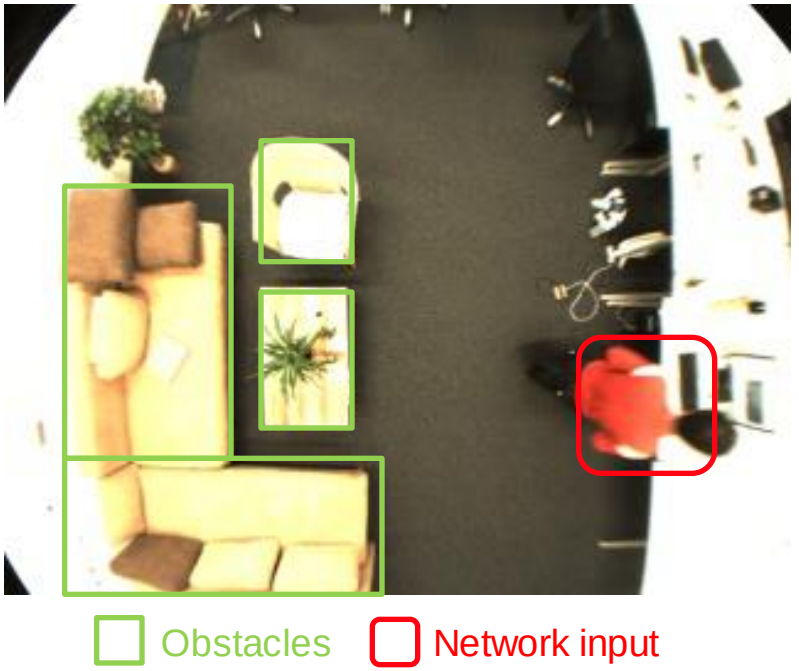




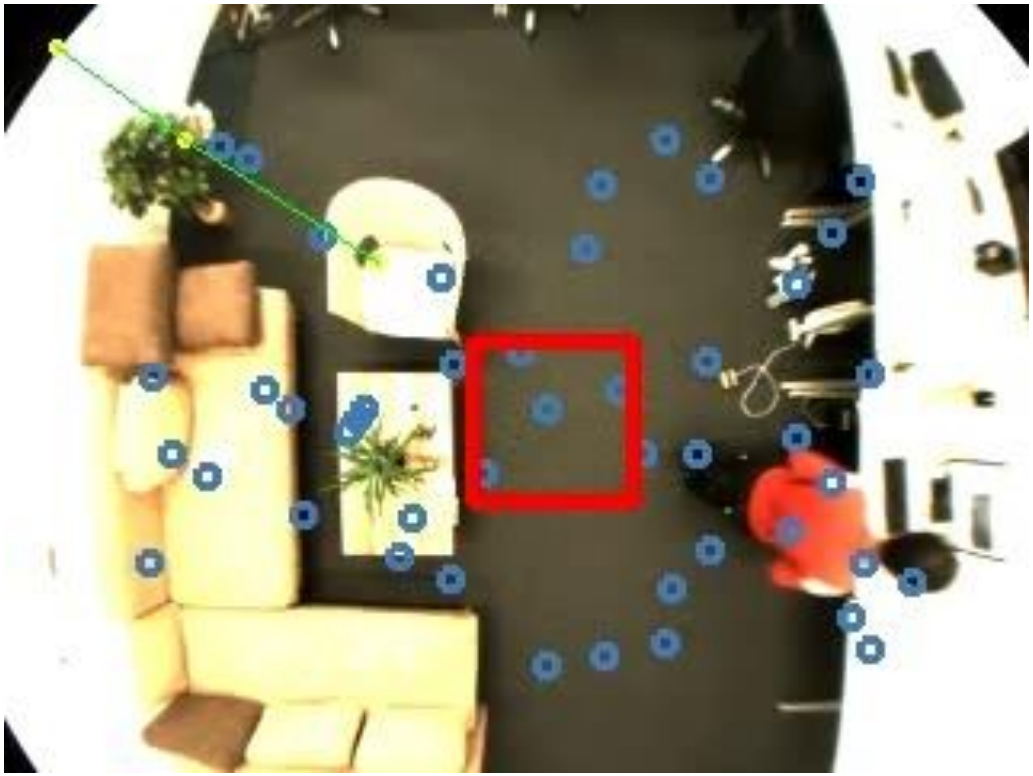
Room Mapping via GNG (KT Lab)

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

Summary

- ***Cluster analysis*** groups objects based on their ***similarity*** and has wide applications
- Measure of similarity can be computed for ***various types of data***
- Clustering algorithms can be ***categorized*** into partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods
- ***Outlier detection*** and analysis are very useful for fraud detection, etc. and can be performed by statistical, distance-based or deviation-based approaches
- There are still lots of research issues on cluster analysis