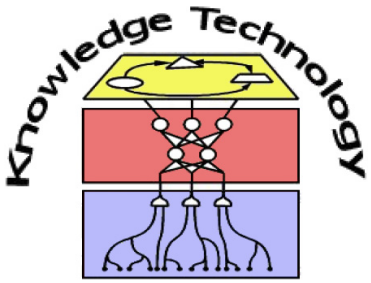
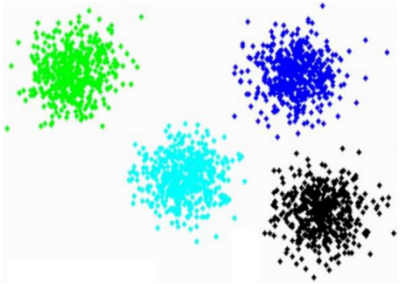
Knowledge Management and Intelligent Assistive Systems

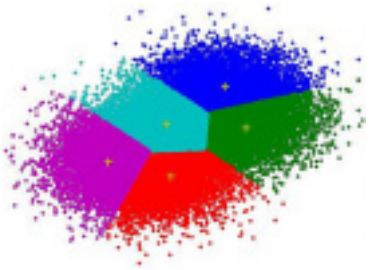
Lecture 8
Clustering and Selforganizing Networks



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



What is Cluster Analysis?

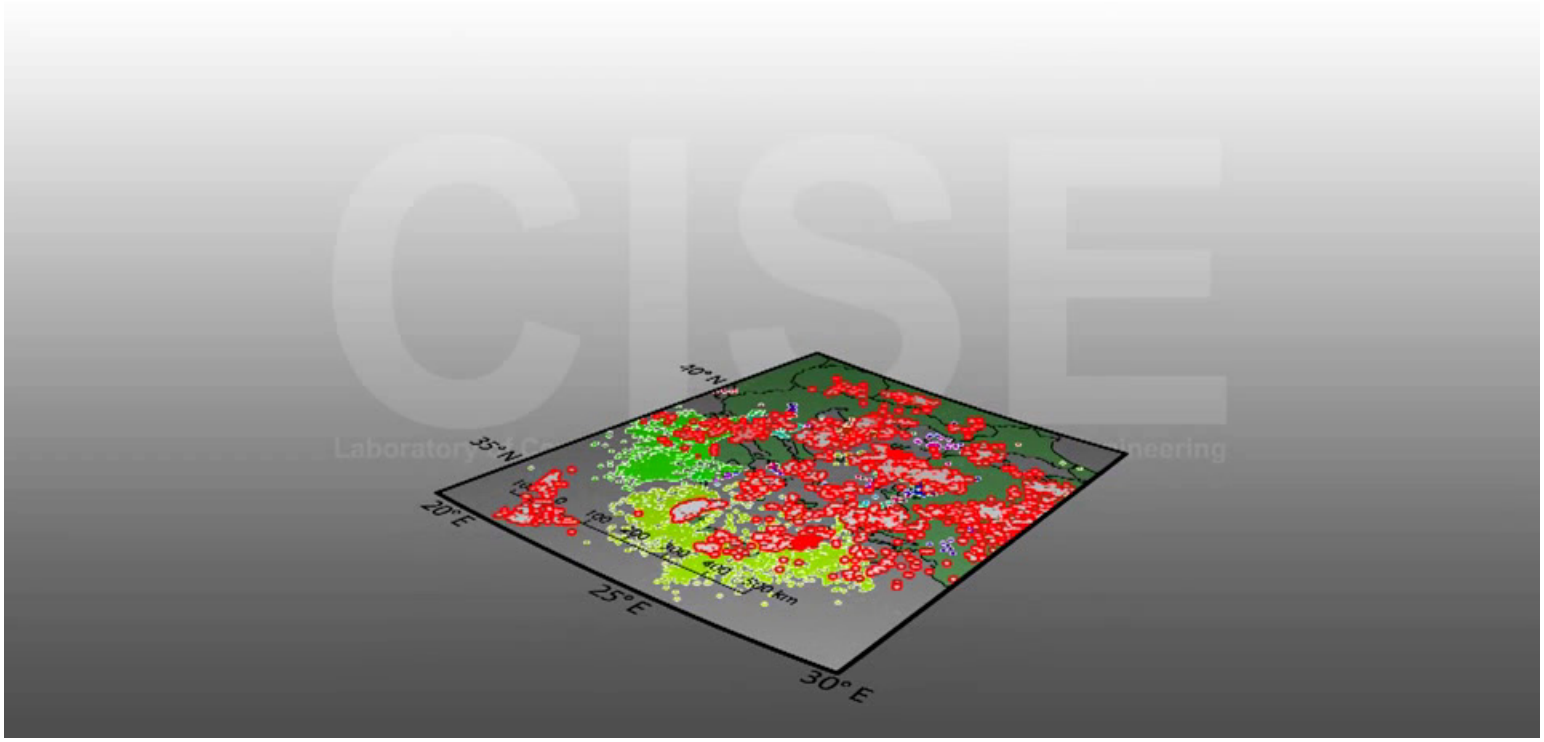


- Cluster: A group of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis
  - Finding similarities between data according to their characteristics and grouping similar data objects into clusters
- ***Unsupervised 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?

- A **good clustering** method will produce 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. metric: $d(i, j)$
  - The definitions of ***distance functions*** are usually different for interval-scaled, boolean, categorical, or vector variables
  - Weights may be associated with different variables based on applications and data semantics
- Quality of clustering:
  - A (separate) "quality" function measures the "goodness" of a clustering process
  - 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 method: K-nearest neighbors – 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

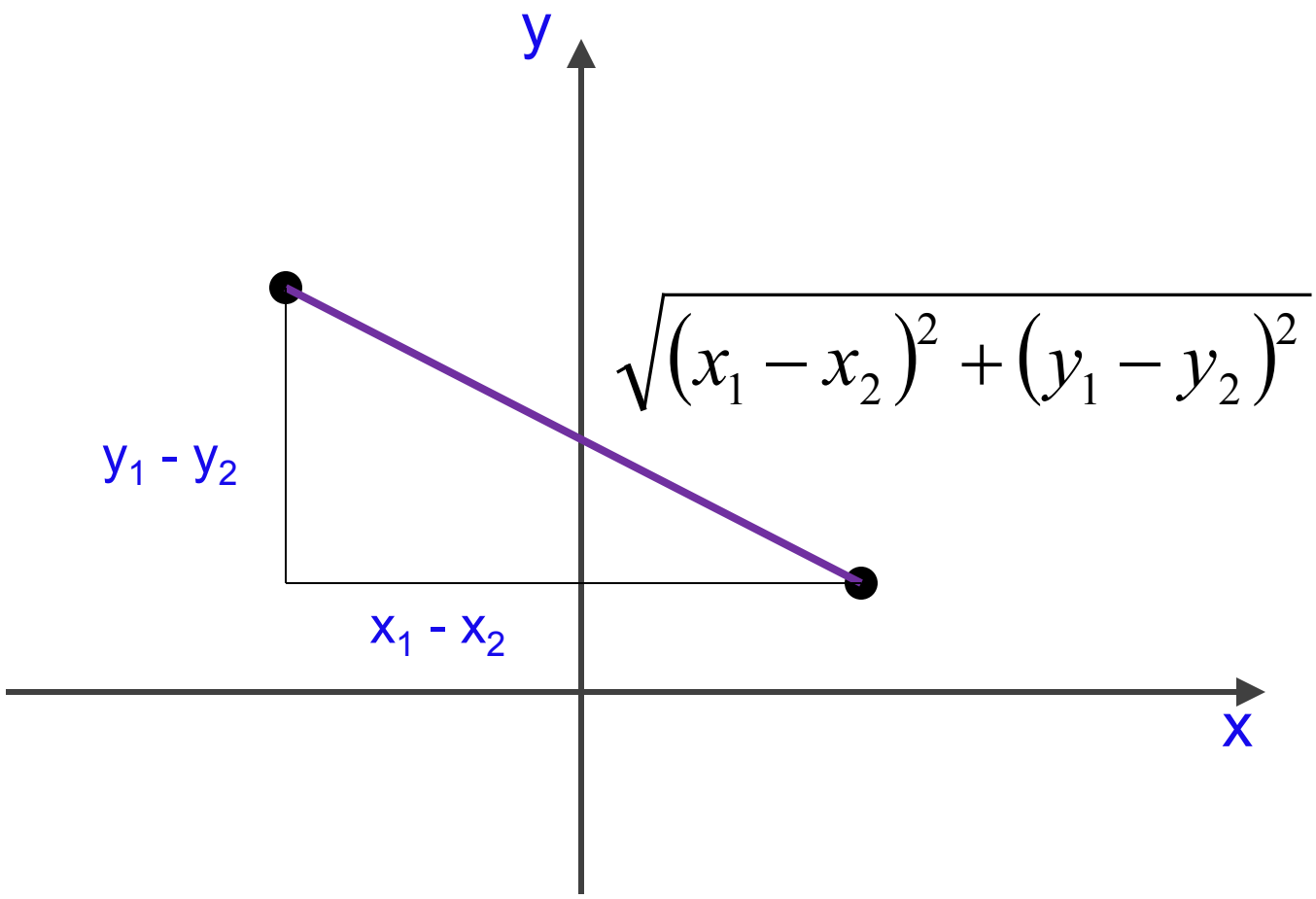
- *h* = 1: (L_1 norm) ***Manhattan distance***
  - E.g., the *Hamming distance* between two *binary vectors*: the number of bits that are different

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

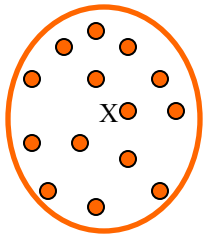
- *h* = 2: (L_2 norm) ***Euclidean distance***

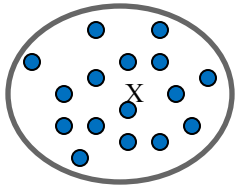
$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2}$$

Euclidean Distance



Distance between Clusters





- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = min(t_{ip}, t_{jq})$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = max(t_{ip}, t_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = avg(t_{ip}, t_{jq})$
- Centroid: distance between the centroids of two clusters, i.e., $dist(K_i, K_j) = dist(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e., $dist(K_i, K_j) = dist(M_i, M_j)$
  - 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 (has minimal average Euclidean distance)

$$C_m = \frac{\sum_{i=1}^{N}(t_{ip})}{N}$$

- Radius: (proportional to) average Euclidean distance from any point of the cluster to its centroid

$$R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{ip} - c_m)^2}{N}}$$

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

$$D_m = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(t_{ip} - t_{jp})^2}{N(N-1)}}$$

Partitioning Algorithms: Basic Concept

- ***Partitioning method***: partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, minimising sum of squared distance to means m_i

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

- 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* nonempty subsets
  2. Compute the centroids (center of mass / ***mean point***) 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_i
    - Compute distance to each cluster centre
    - Assign data point to nearest cluster centre with distance

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

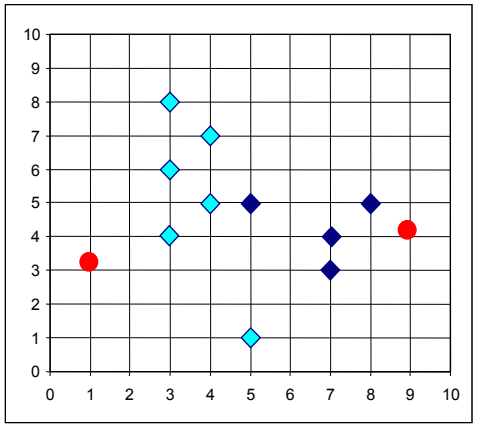
$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{i=1}^{Nj} \boldsymbol{x}_i$$

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

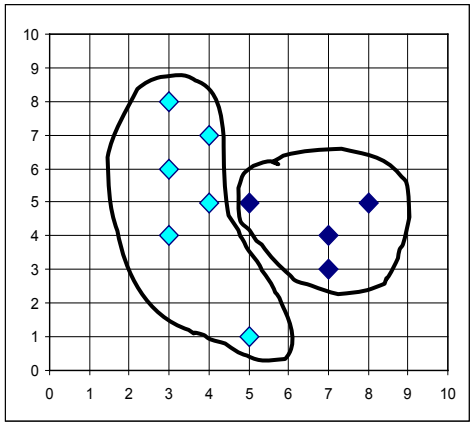
- Example



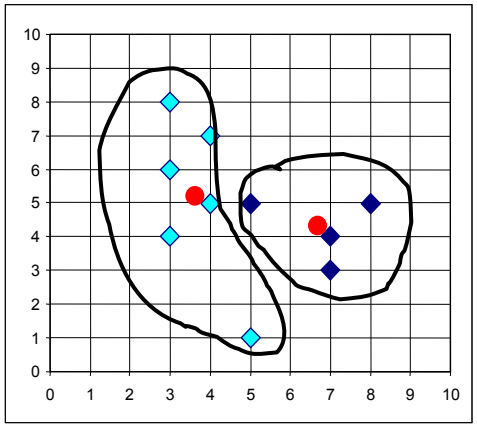
K=2

Arbitrarily choose K objects as initial cluster center

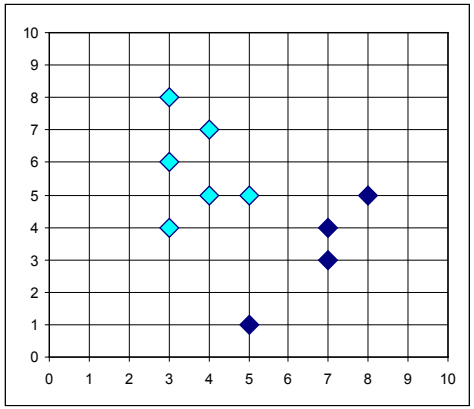
Assign each object to most similar center



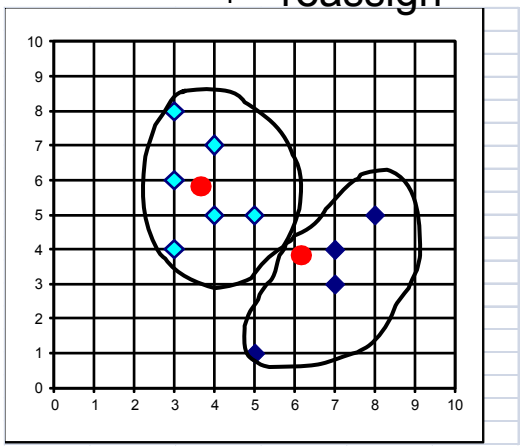
Update the cluster means



reassign



Update the cluster means



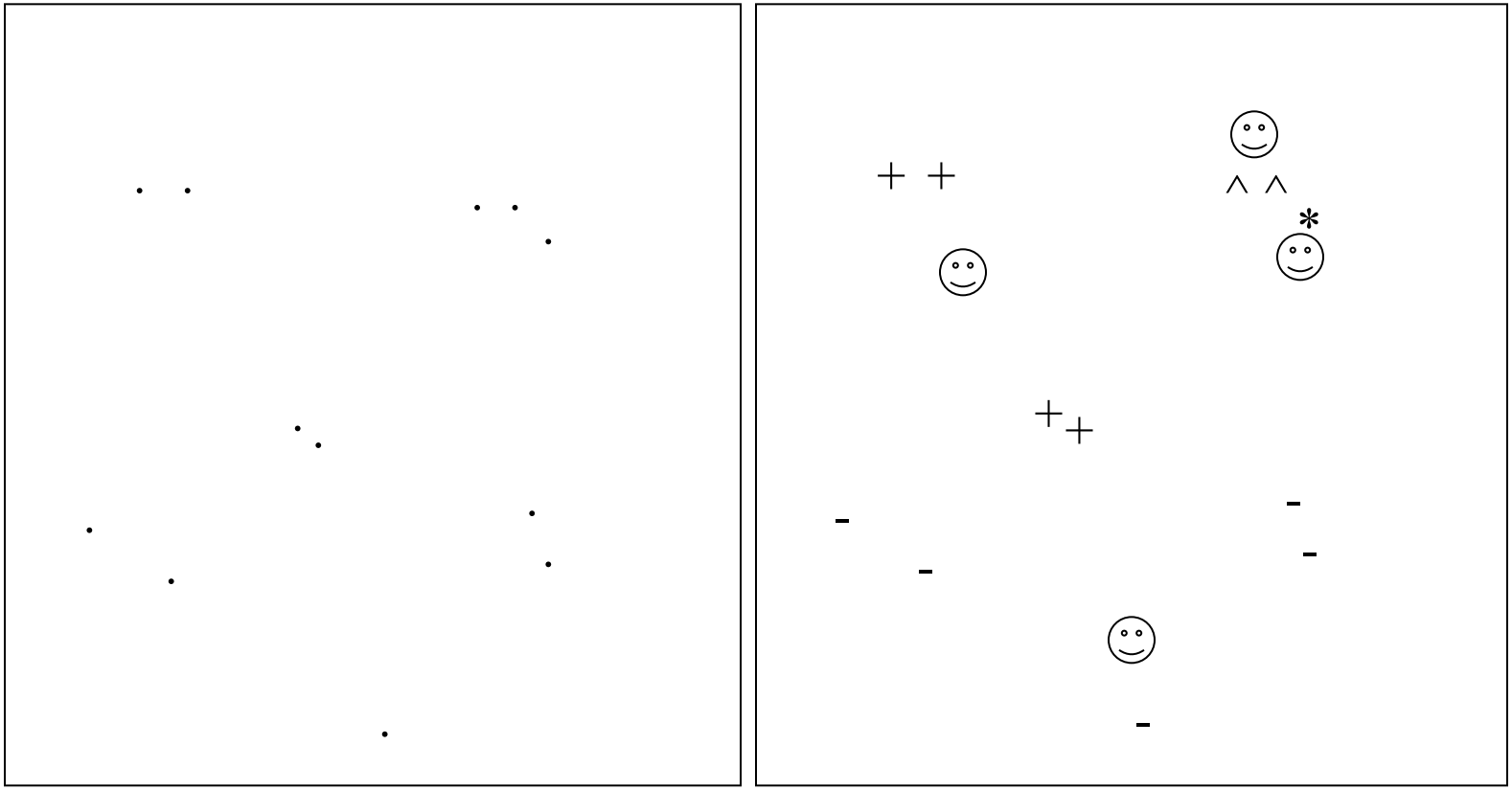
reassign

Why k-means converges

- Whenever an assignment is changed, the sum squared distances of datapoints from their assigned cluster centers is reduced.
- Whenever a cluster center is moved, the sum squared distances of the datapoints from their currently assigned cluster centers is reduced.
- If the assignments do not change in the assignment step, we have converged.

Visualisation of the Clustering

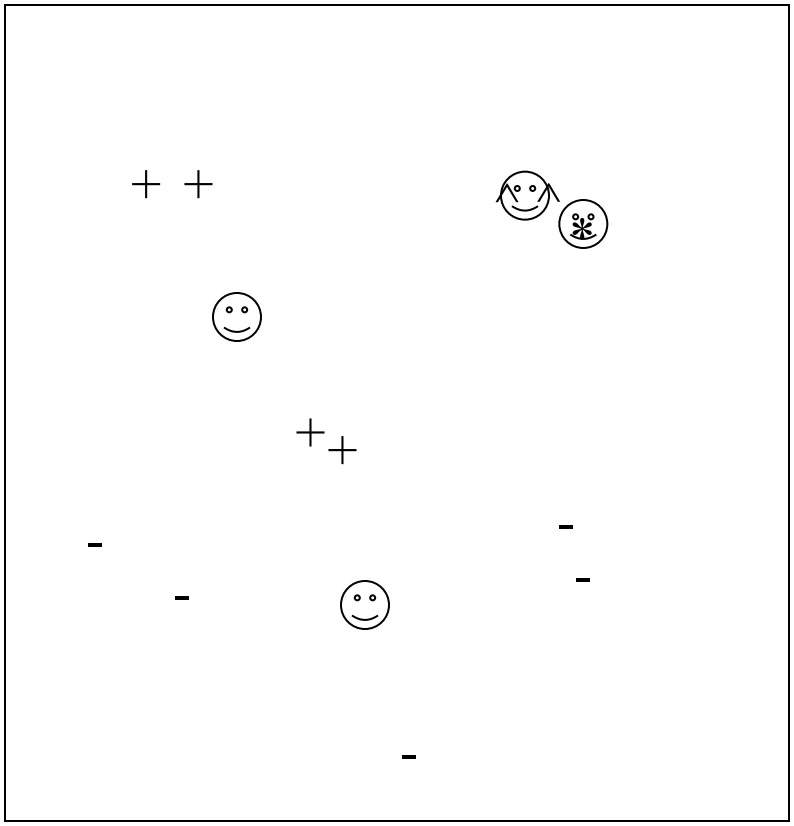
4 means

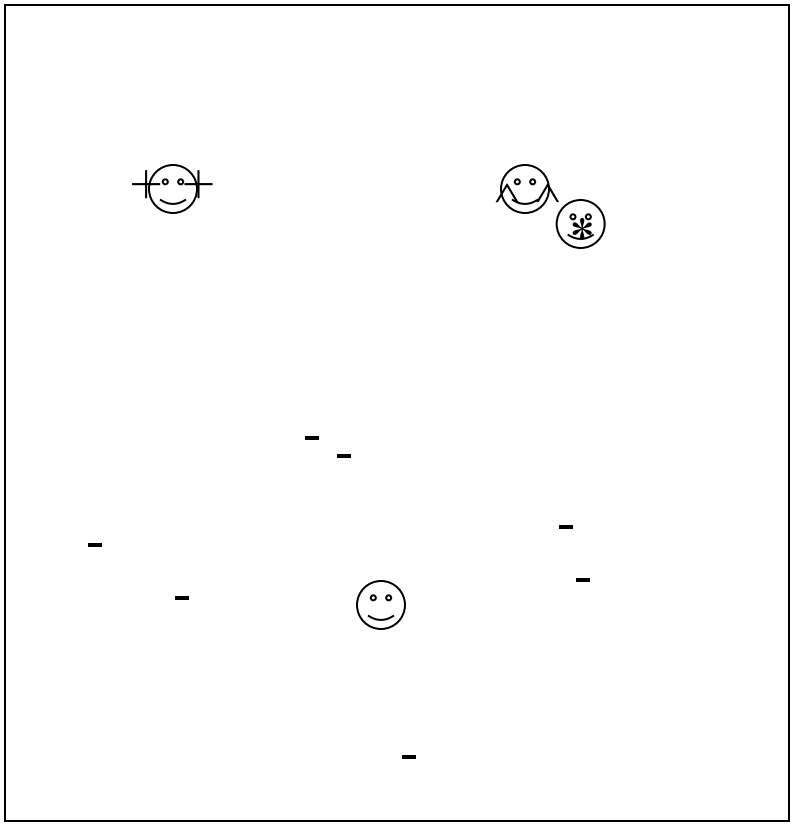


A two dimensional dataset; different ways to position the centres, one is shown to the right

Visualisation of the Clustering

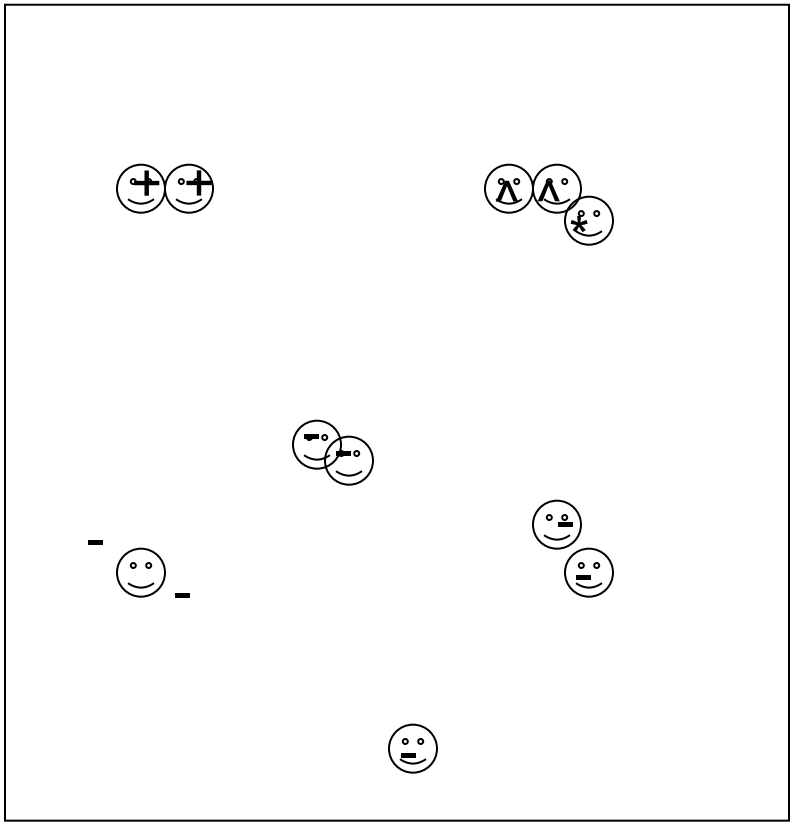
These are local minima solutions



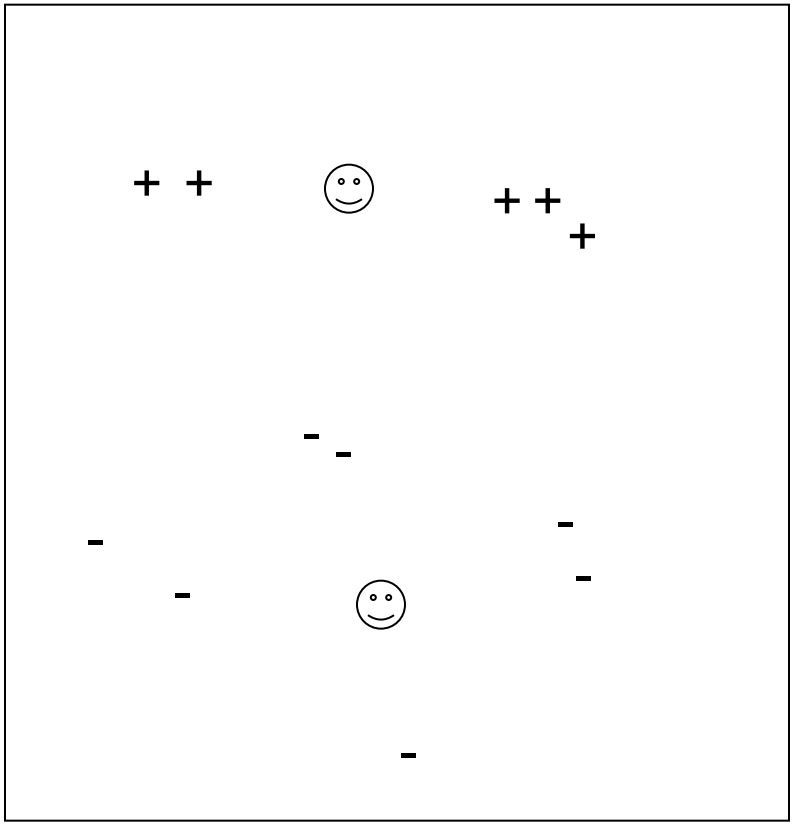


Visualisation of the Clustering

Overfitting



Underfitting

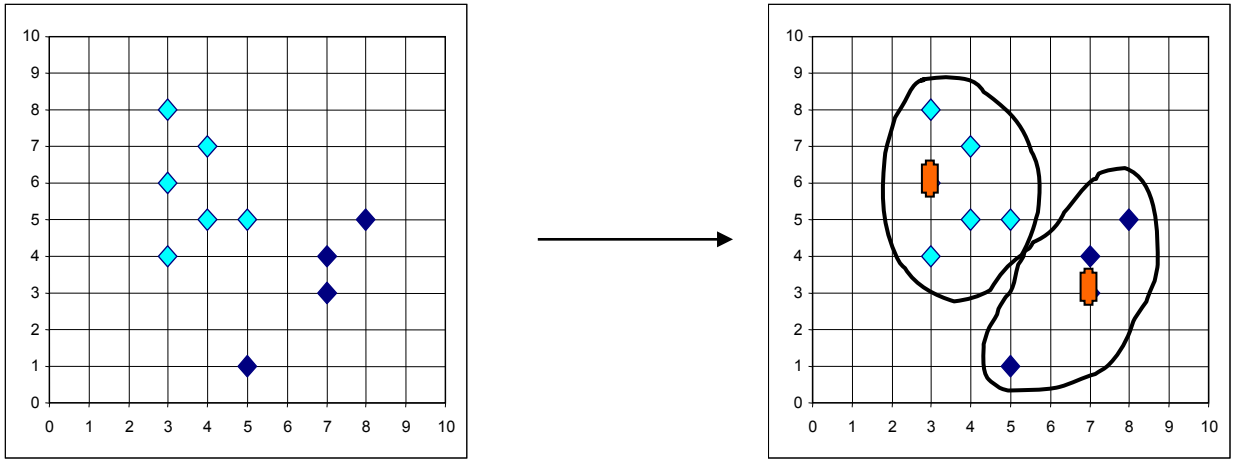


Comments on the *K-Means* Method

- **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

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*** can be used, which is the ***most centrally located object*** in a cluster.



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

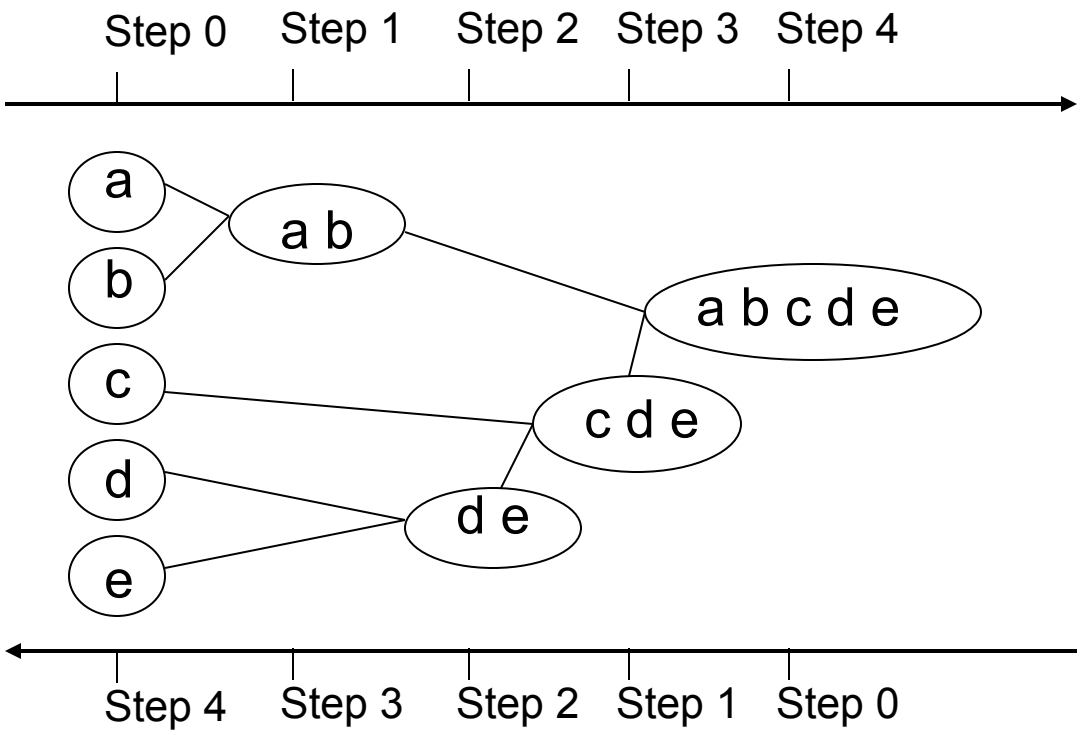
Niklas Bergström, Mårten Björkman, Danica Kragic
CSC/KTH Stockholm, Sweden
IROS '11

Other Typical Partitioning Clustering Methods

- *K-Medoids* Clustering: Find *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
    - *PAM* works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity)
  - *CLARA* (Kaufmann & Rousseeuw, 1990): PAM on samples
  - *CLARANS* (Ng & Han, 1994): Randomized sampling
- *K-Medians* Clustering: Find k medians instead of means (minimizes error w.r.t. the 1-norm distance, e.g. Manhatten)

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

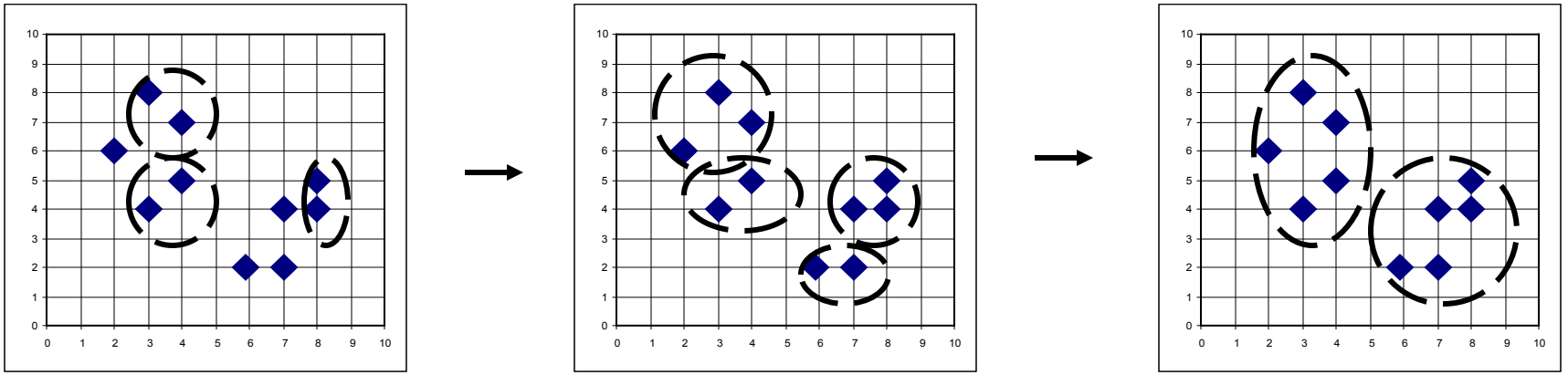


agglomerative
(AGNES)
Bottom up

divisive
(DIANA)
Top down

AGNES (Agglomerative Nesting)

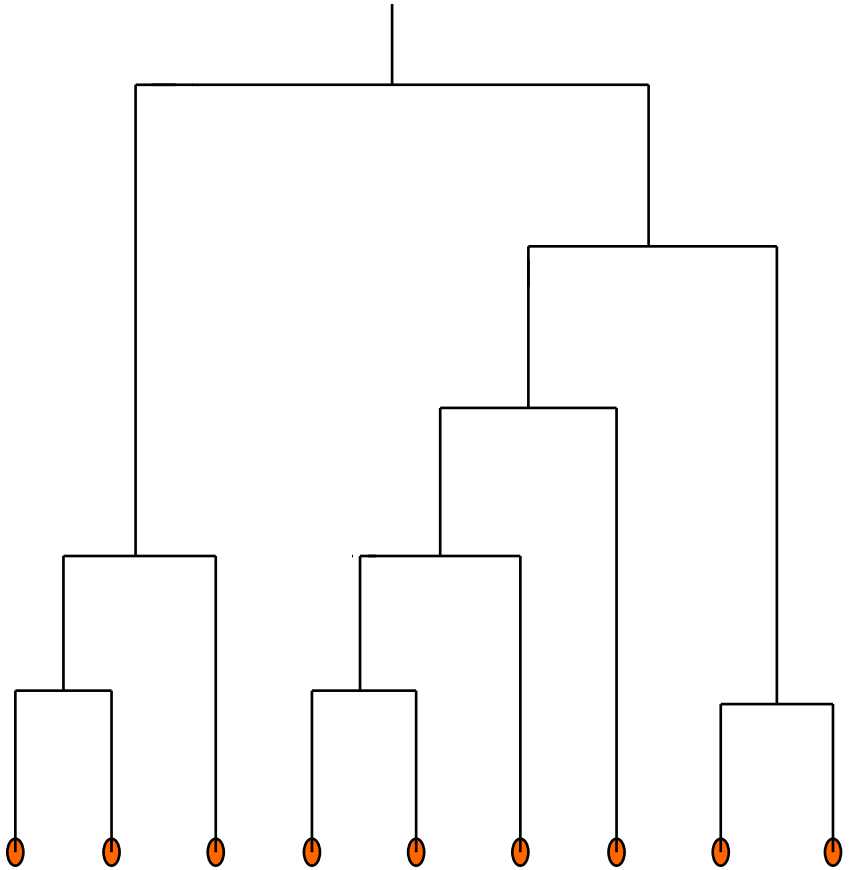
- Introduced in Kaufmann and Rousseeuw (1990)
- Implemented in statistical packages, e.g., Splus
- Use the ***single-link*** method and the dissimilarity matrix
- Merge nodes that have the least dissimilarity
- Go on in a non-descending fashion
- Eventually all nodes belong to the same cluster



Dendrogram shows how clusters are merged

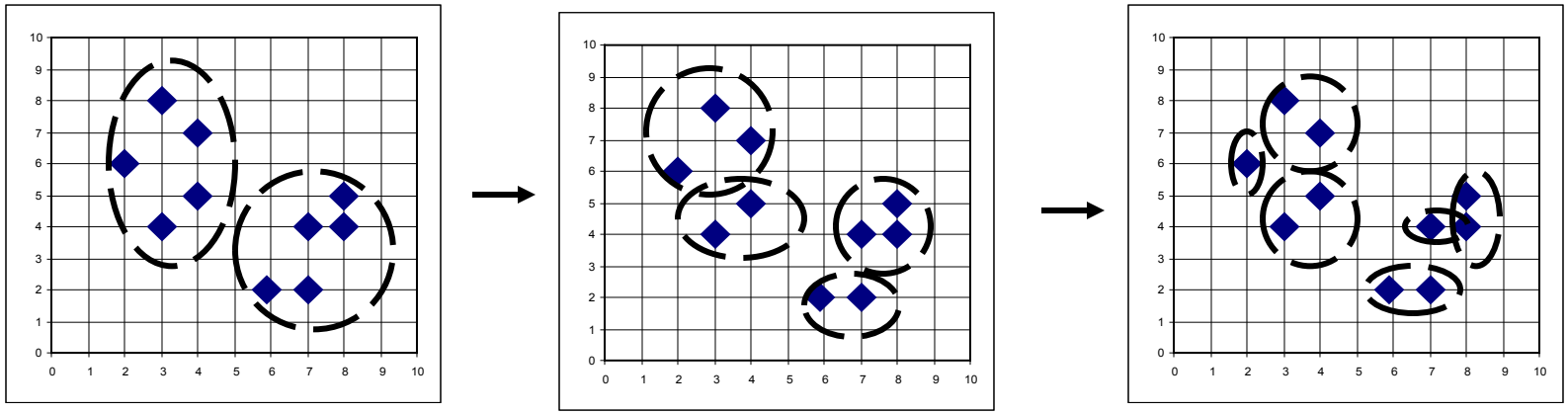
- Decompose data objects into several levels of nested partitioning (***tree*** of clusters), called a ***dendrogram***.

- 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



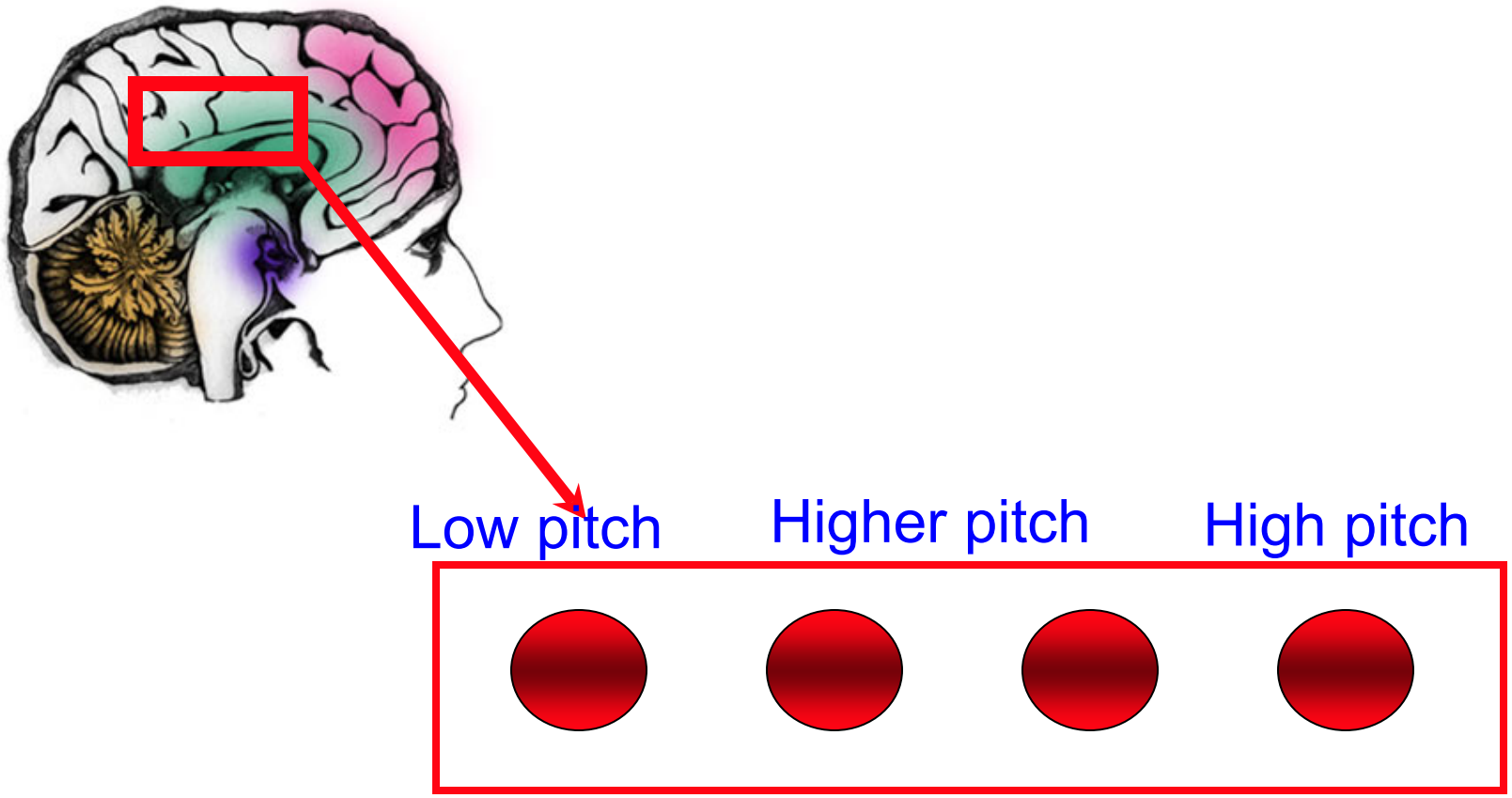
Model-Based Clustering

- 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)

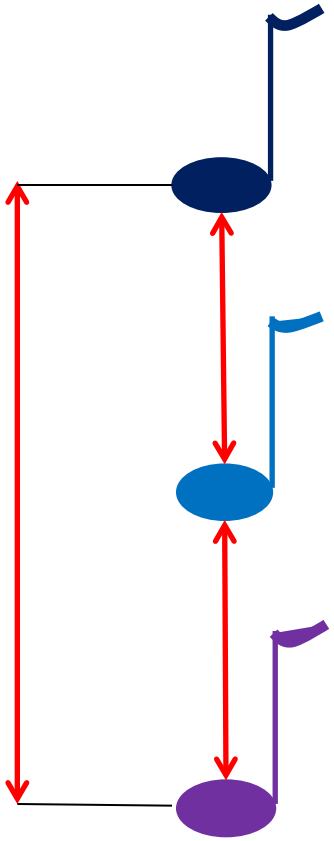
Neural Network Approaches

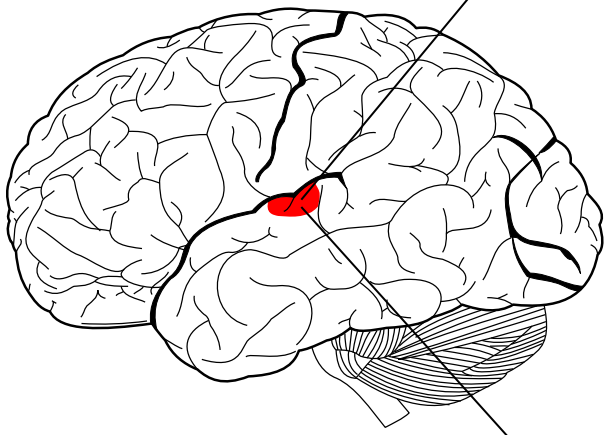
- Neural network approaches
  - Represent each cluster with an exemplar (a neuron), acting as a "prototype" of the cluster
  - New objects are assigned to the cluster whose exemplar is the most similar according to some distance measure
- Typical methods
  - SOM (Self-Organizing feature Map)
  - Competitive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

Feature Maps

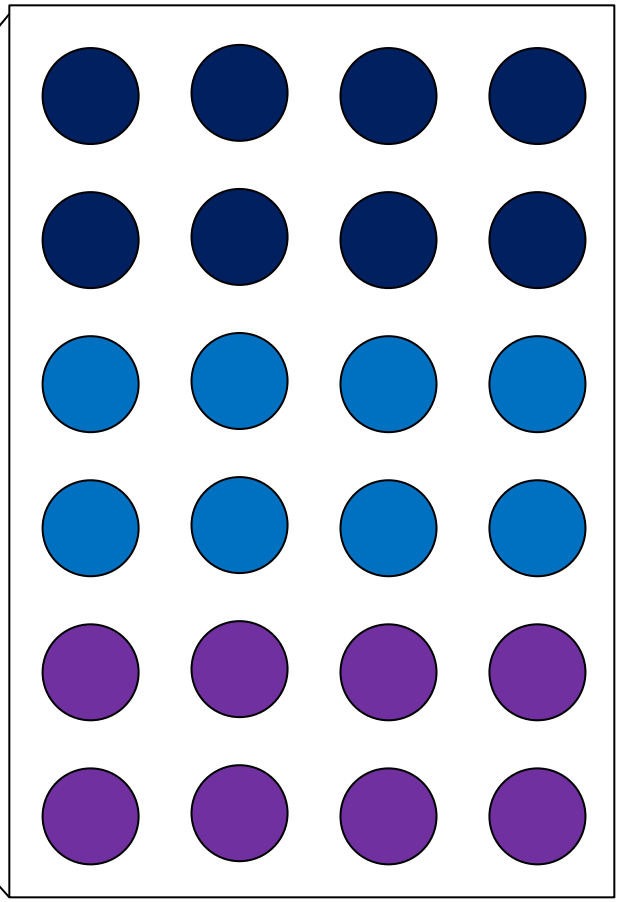


Feature Maps: How are Signals Mapped into the Auditory Cortex?





Auditory Cortex



Different sound signals

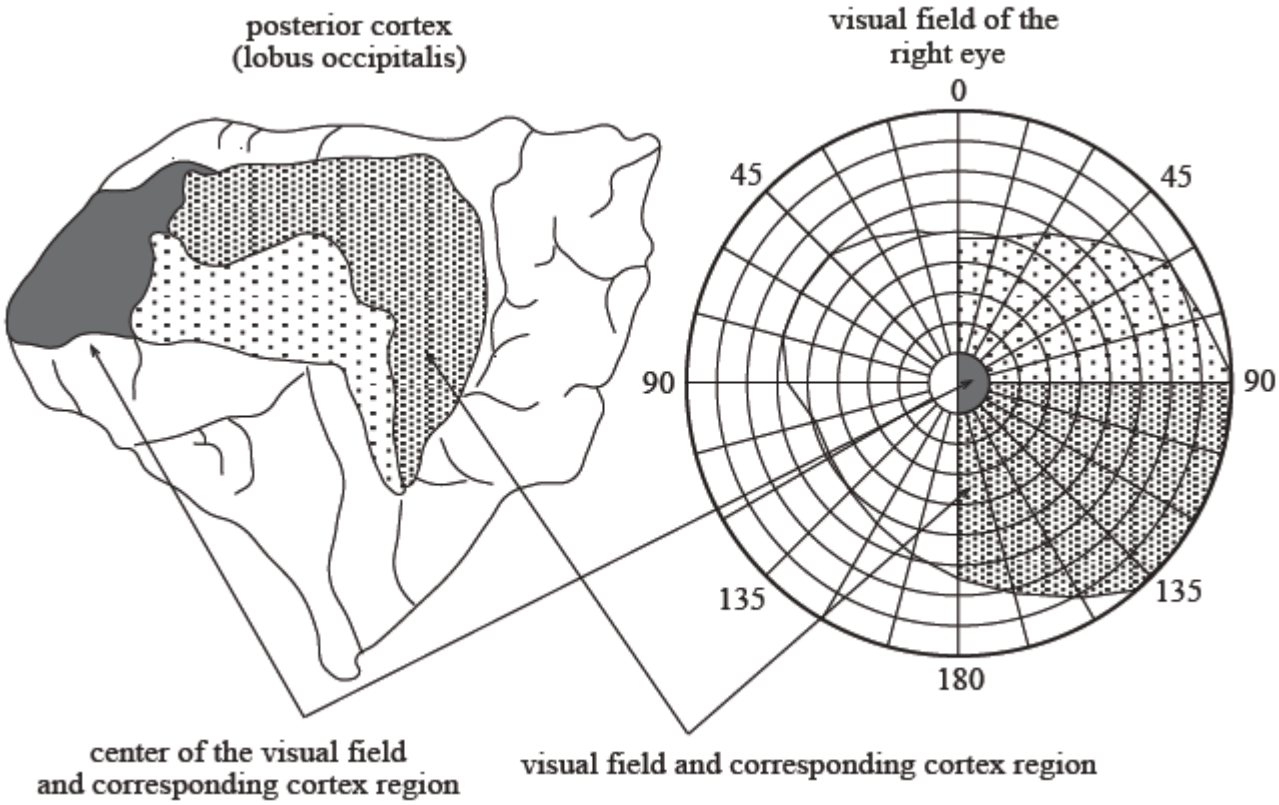
Activated Neurons

Feature Maps

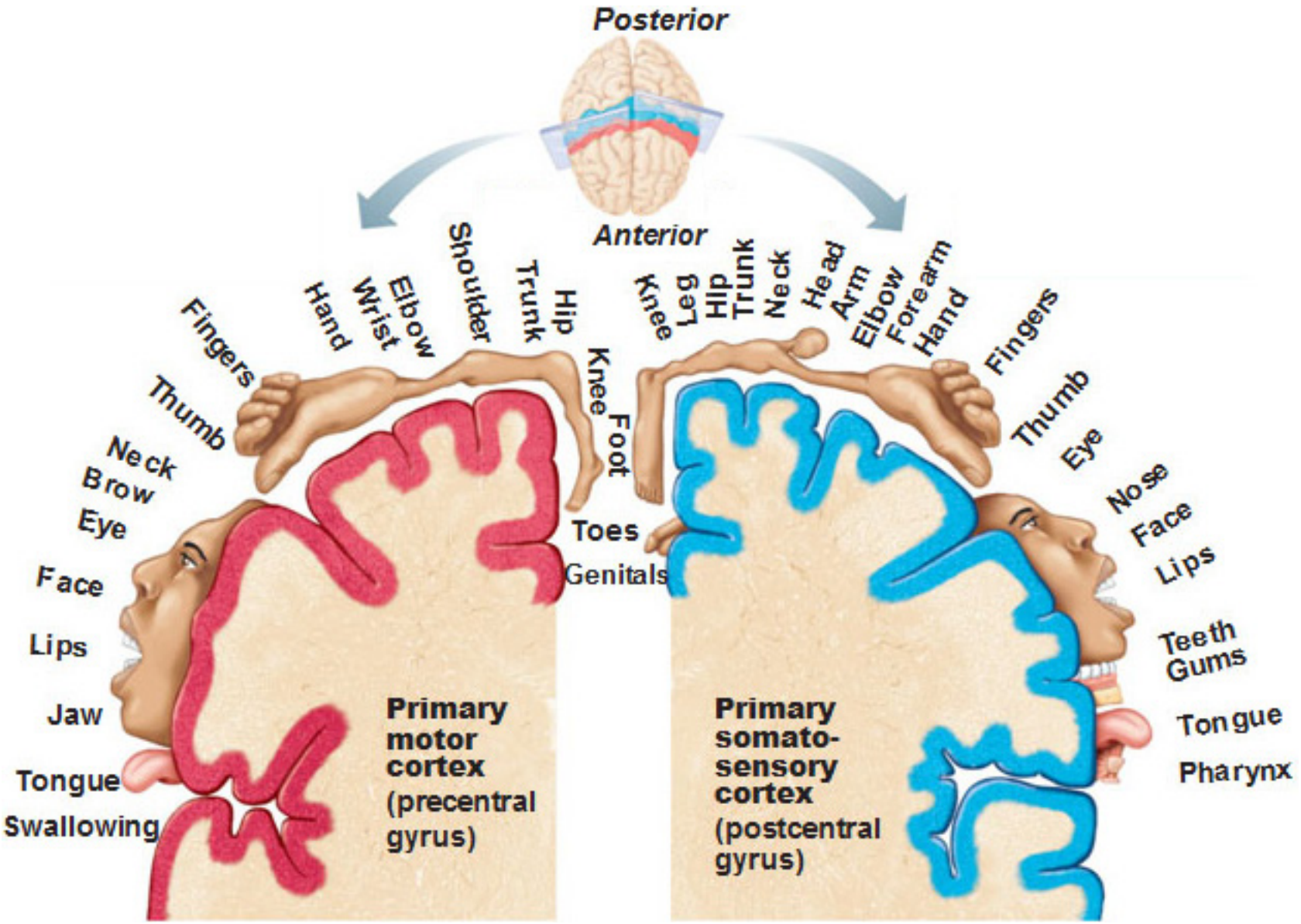
- Sounds that are similar (‘nearby in input space’) excite neurons that are near to each other (in `output space’)
- Sounds that are very different excite neurons that are a long way off
- This is known as ***topology preservation***
- The ordering of the inputs is preserved
  - If possible (perfectly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

Mapping of visual Field to Cortex

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

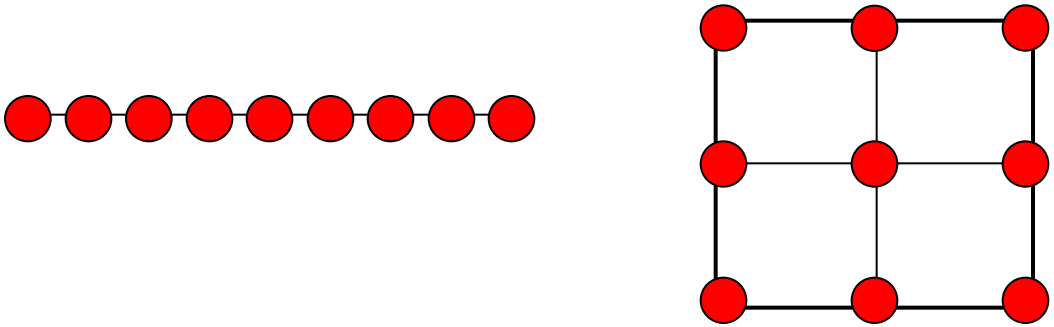


Mapping of effectors and sensors to Cortex

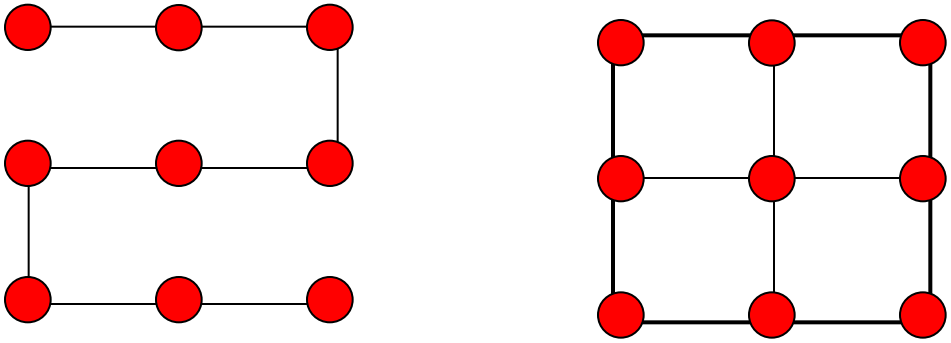


Topology Preservation in 2D

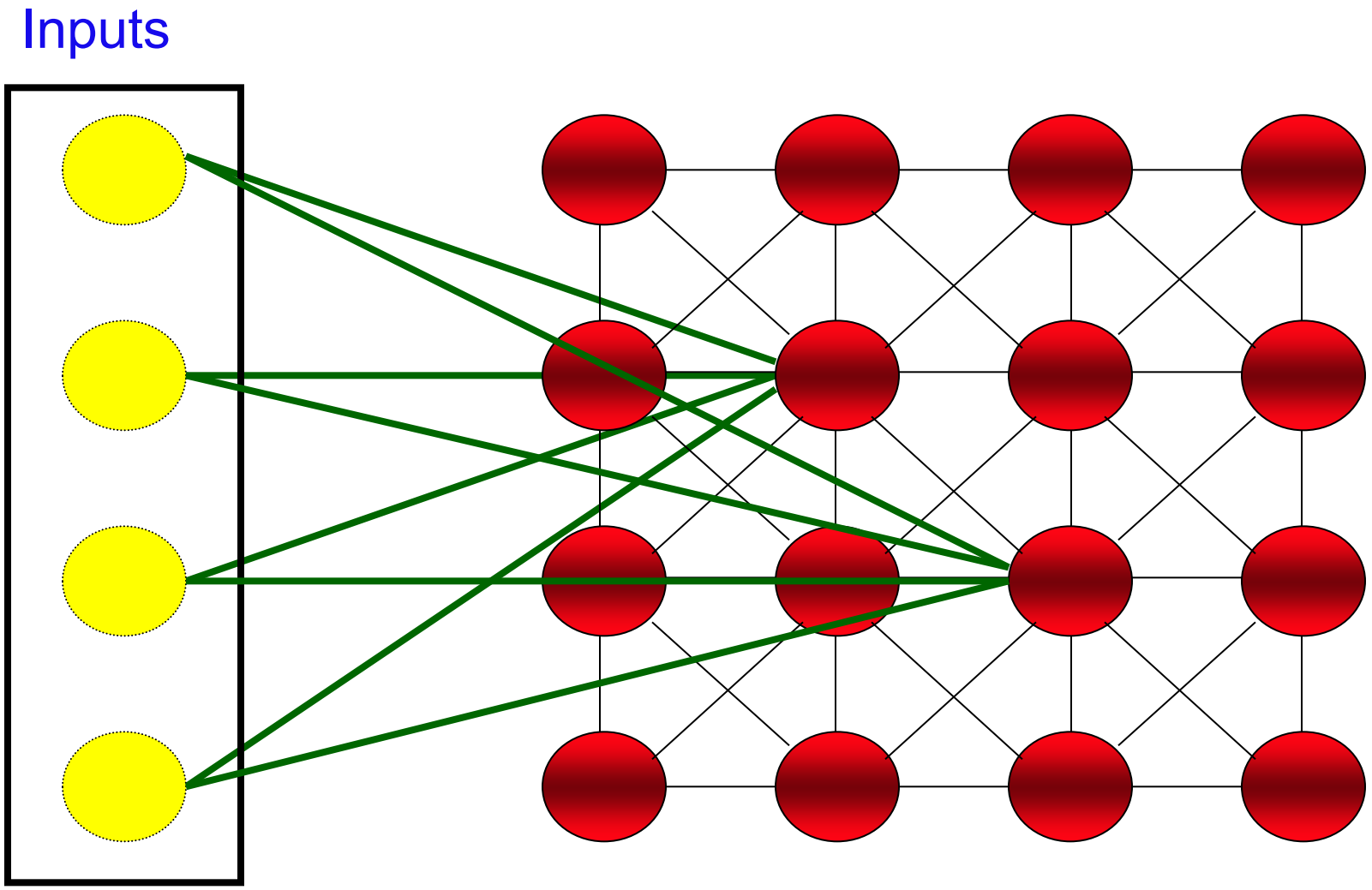
Network Topology



Network Representation of Input Space

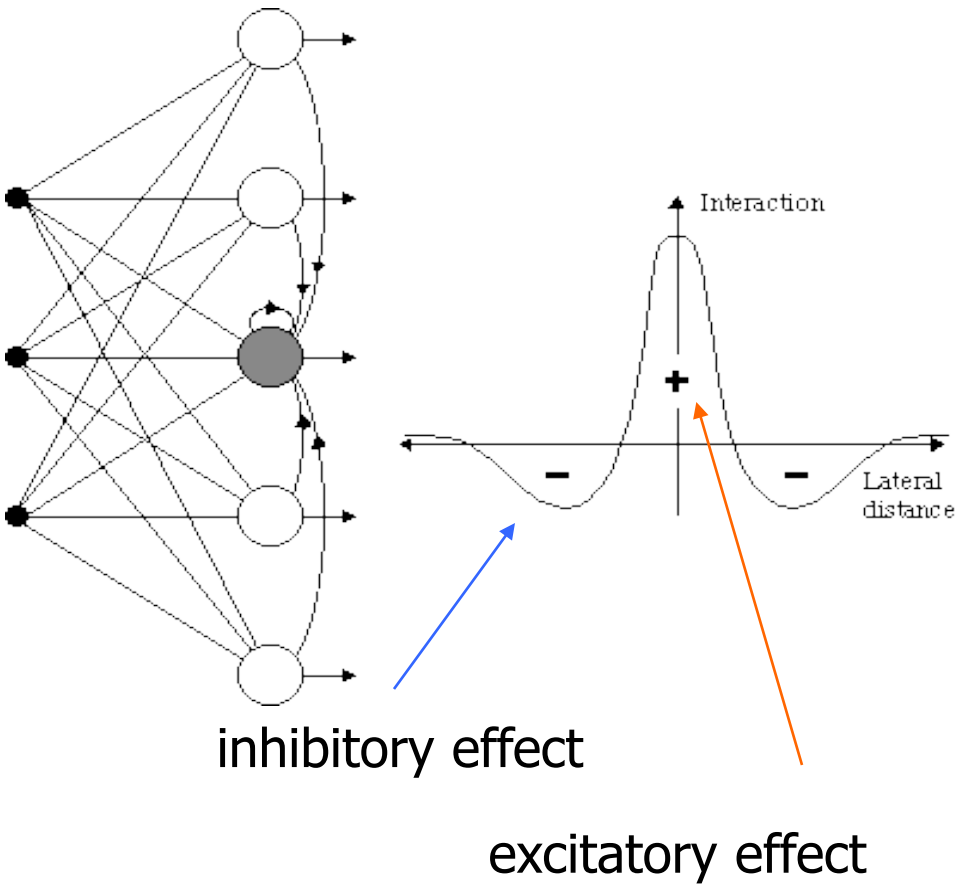


The Self-Organising Map



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
- → leads to WTA 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?
  - Early in learning, network is unordered
    - Big neighbourhood: similar input vectors excite neurons far apart
    - They will also learn similarly and form clusters in input space
  - Later on, just fine-tuning network
    - Small neighbourhood: similar input vectors excite neurons closer together

Everybody Needs Good Neighbours

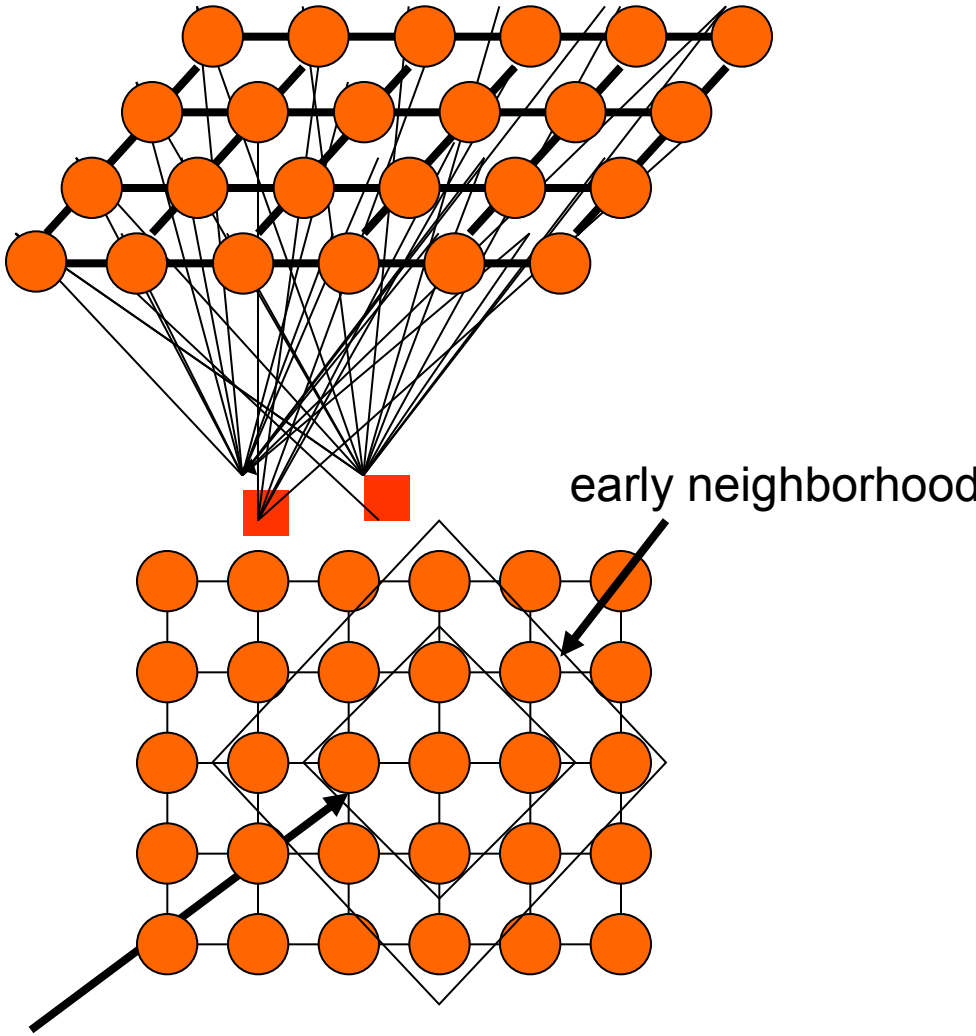
- Generate a neighbourhood size function
- Pick the nodes that are in the neighbourhood
- Decrease the neighbourhood each epoch
- Do the same for the learning rate

Self-Organizing Feature Map (SOM)

- SOMs, also called topological ordered maps, or Kohonen Self-Organizing Feature Map (KSOMs)
- 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

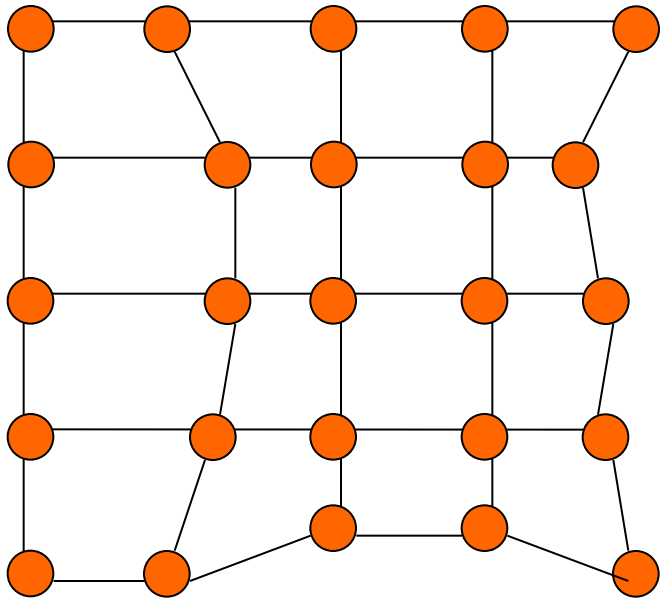
Self organizing maps

- 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

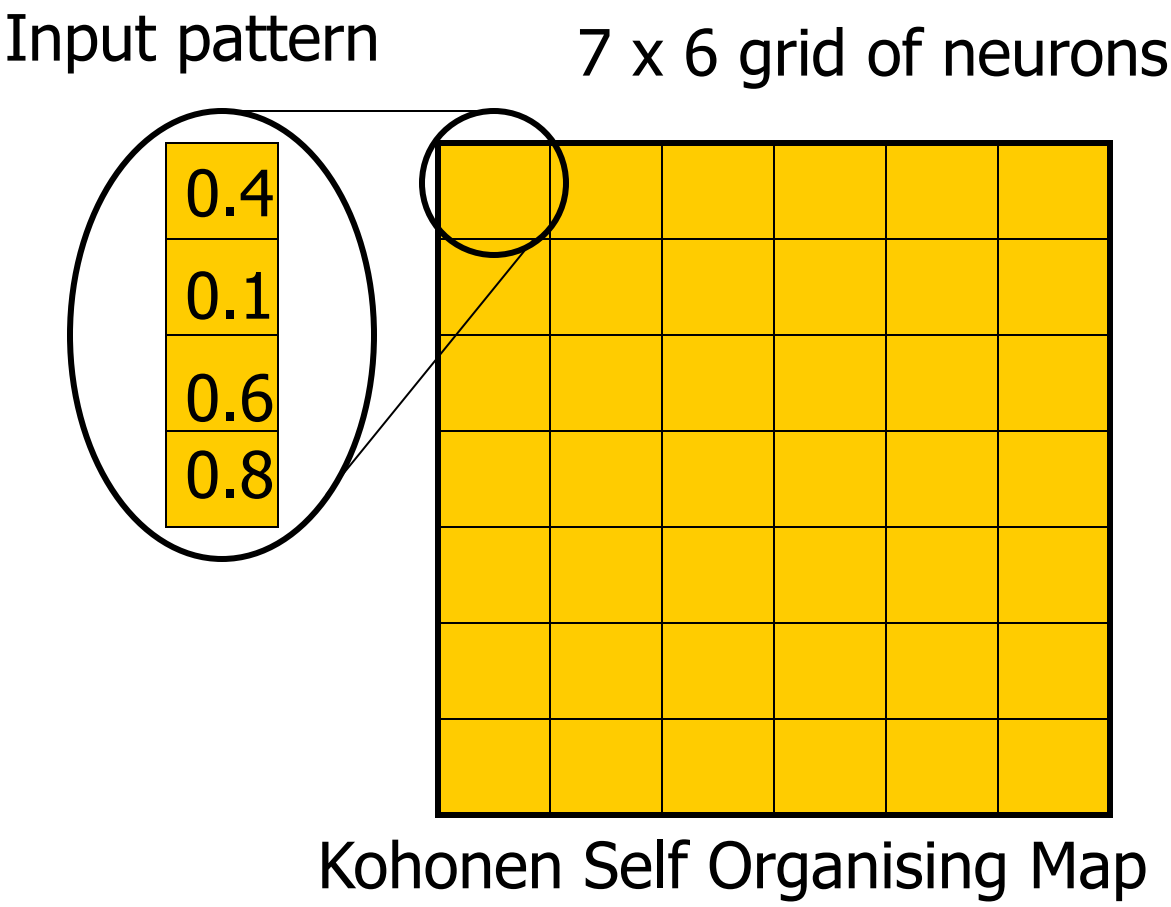


Adaptation

- 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 Architecture Overview



SOM `Cost Function´

- K-means:

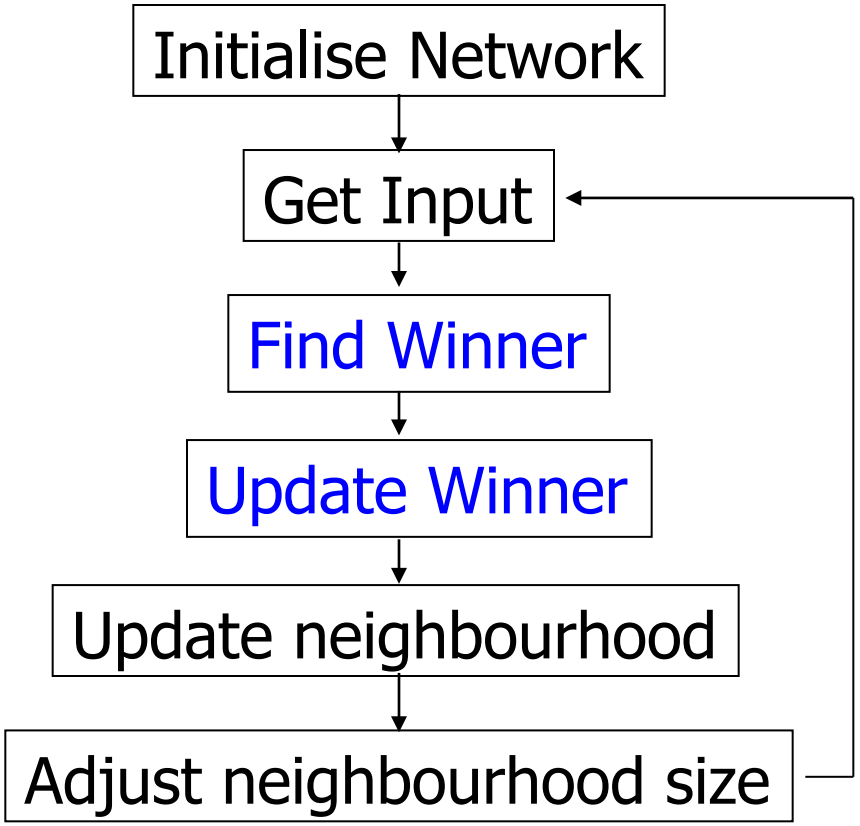
$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2$$

- SOM:

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

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 $\eta(t)$)

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

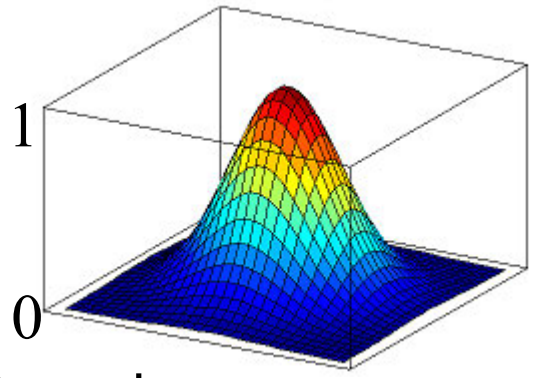
- Over time, the network *self-organises* so that the input topology is preserved

Neighborhood Function Preserves Topology

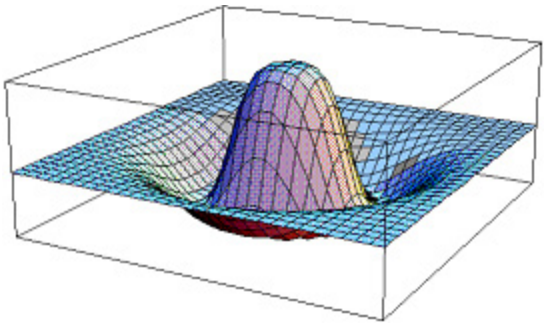
- The neighborhood function $h(n_b,t)$ determines the degree of weight vector change of the neighbors

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

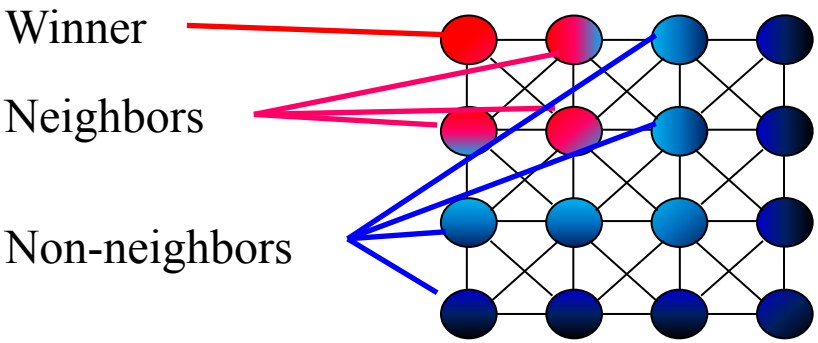
  - Mostly: Gaussian function
    rarely: Mexican Hat function
  - Width decreases during training
    (→ implicit decrease of learning rate)
  - *May* decrease to zero (→ k-means)



Gaussian
(not normalized)



Mexican Hat
(Difference of Gaussian)



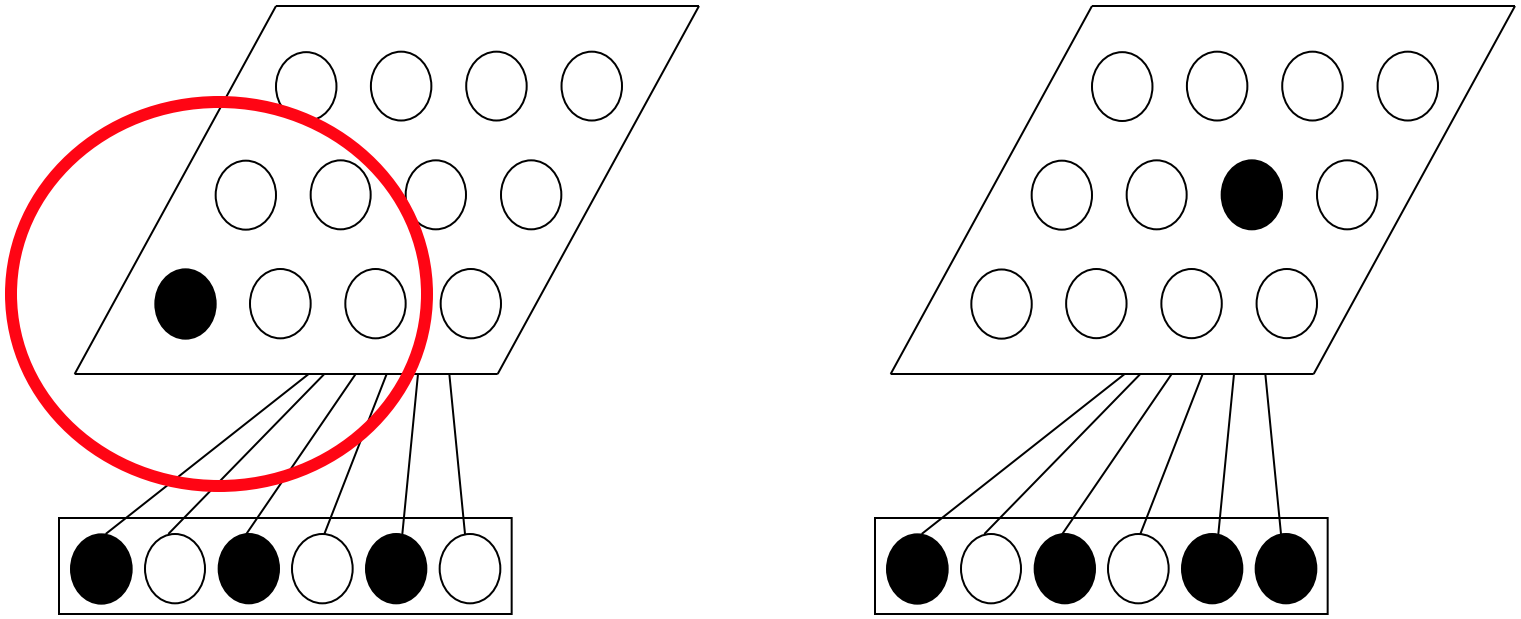
Self-Organization

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

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

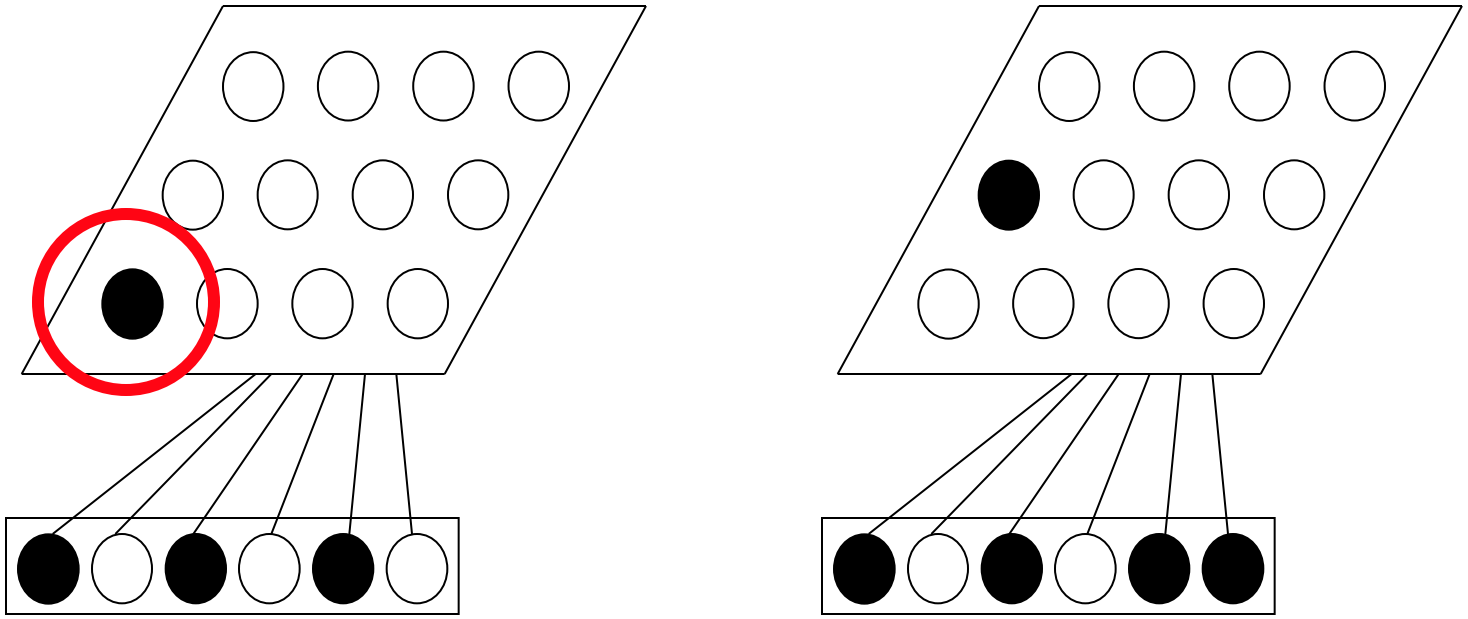
The Self-Organizing Map

Before training (large neighbourhood)

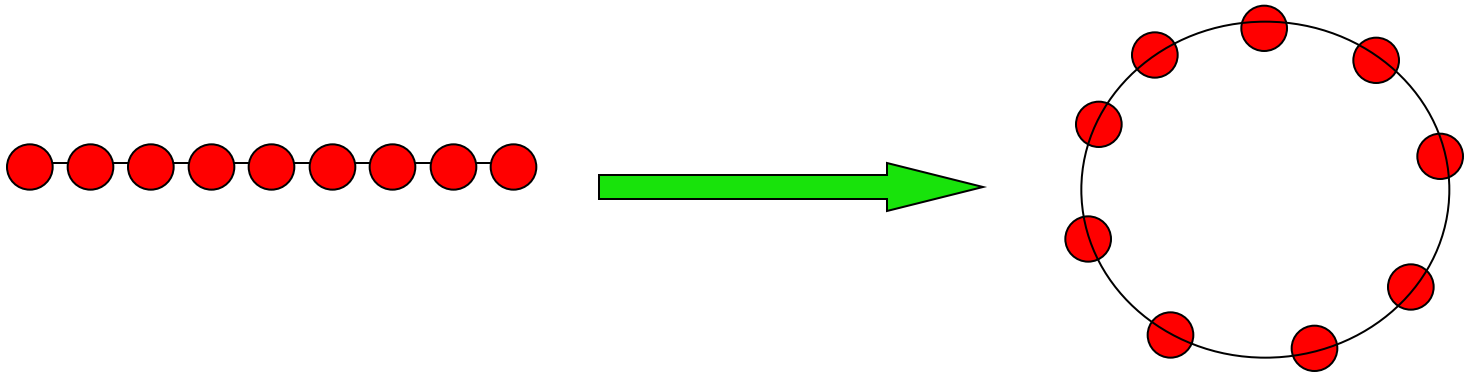


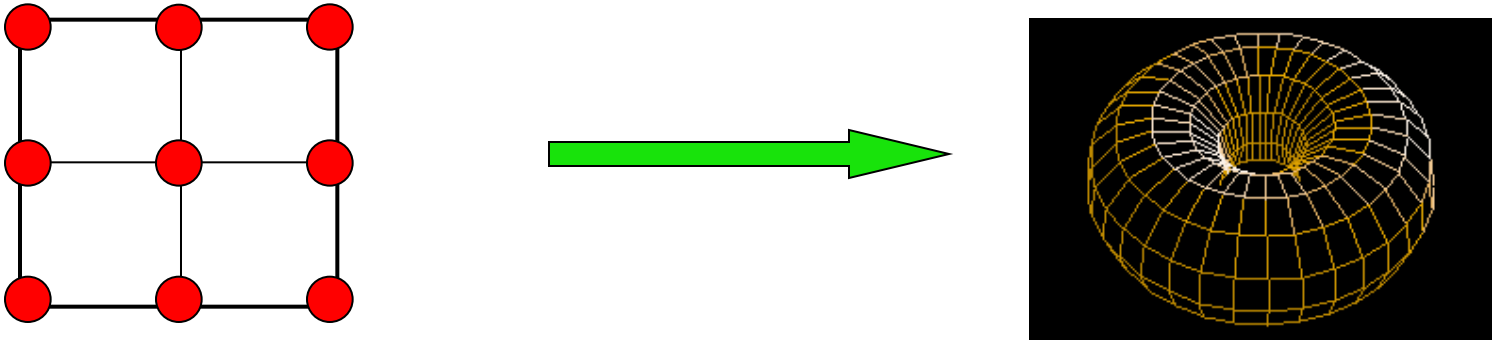
The Self-Organizing Map

After training (small neighbourhood)

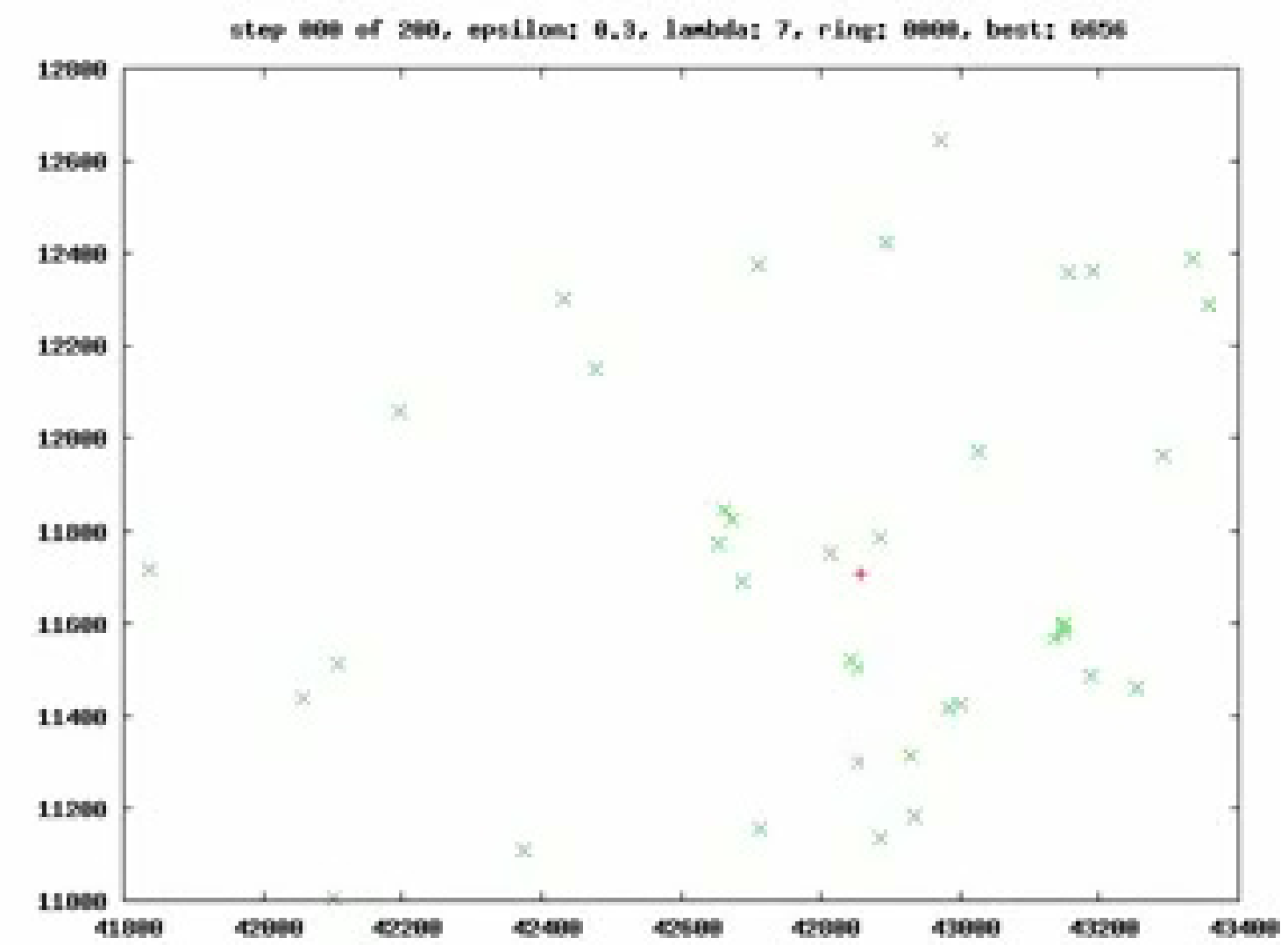


Boundary Conditions:
No Neurons at the End of the Map





1D Ring-Form SOM for the Travelling Salesman Problem



Network Size

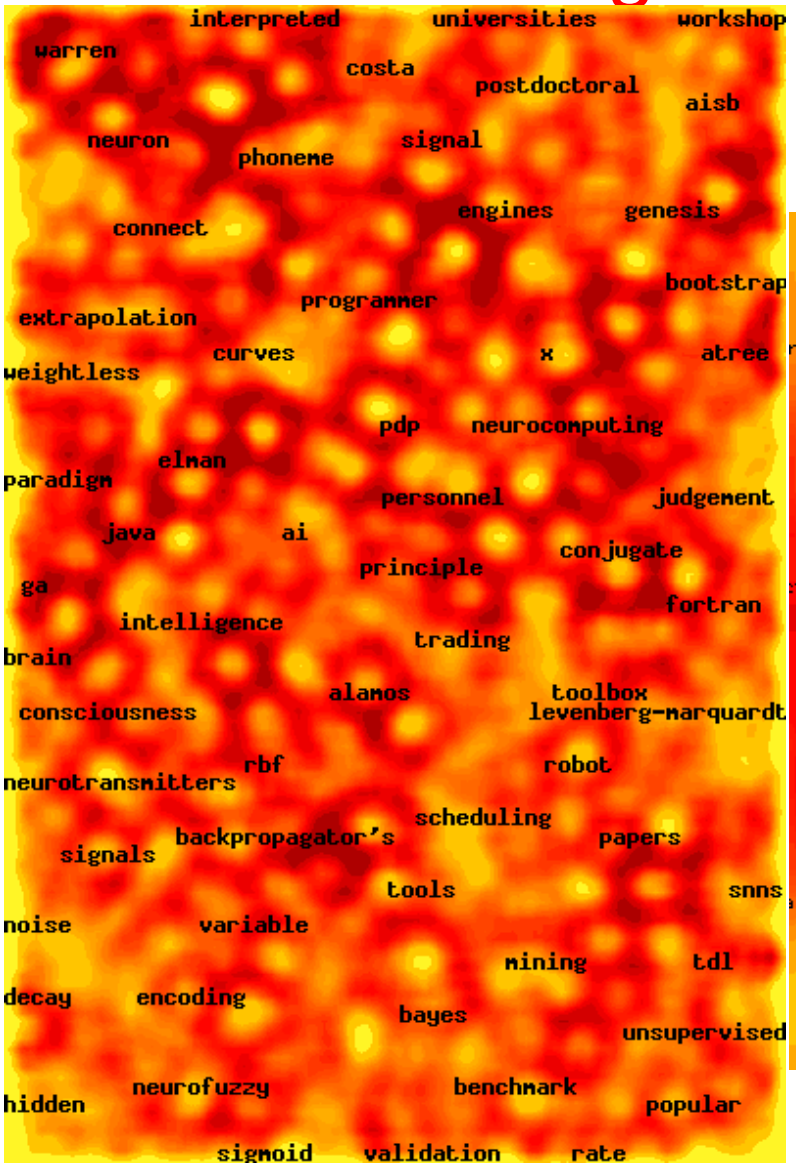
- We have to predetermine the network size
- Big network
  - Each neuron can in principle represent exact features
    → not much generalisation
  - Large neighbourhood interaction keeps network `small'
- Small network
  - Too much generalization
    → no differentiation
- Experiments to identify most suitable size

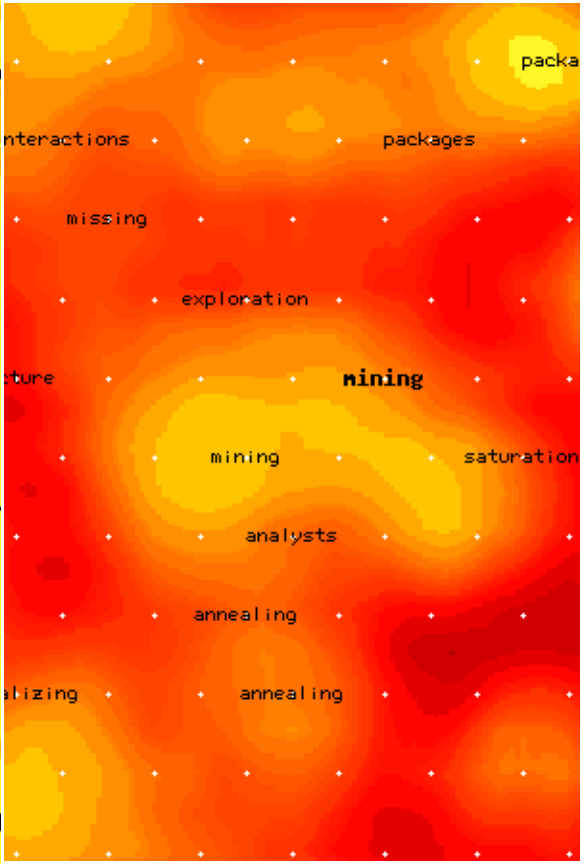
Batch Learning

- Need all the data to start with
  - Run for many iterations
- Can therefore order neurons to begin with
  - Principal components of data
- For on-line learning there are other algorithms
  - Fritzke's 'Growing Neural Gas'
  - Marsland's 'Growing When Required' network

Web Document Clustering Using SOM

- The result of SOM clustering of 12088 Web articles
- The picture on the right: drilling down on the keyword “mining”

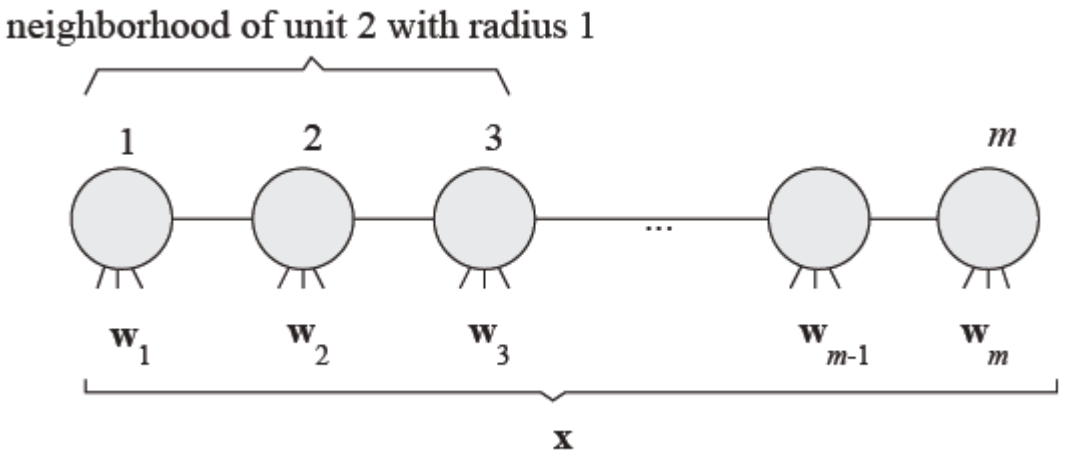




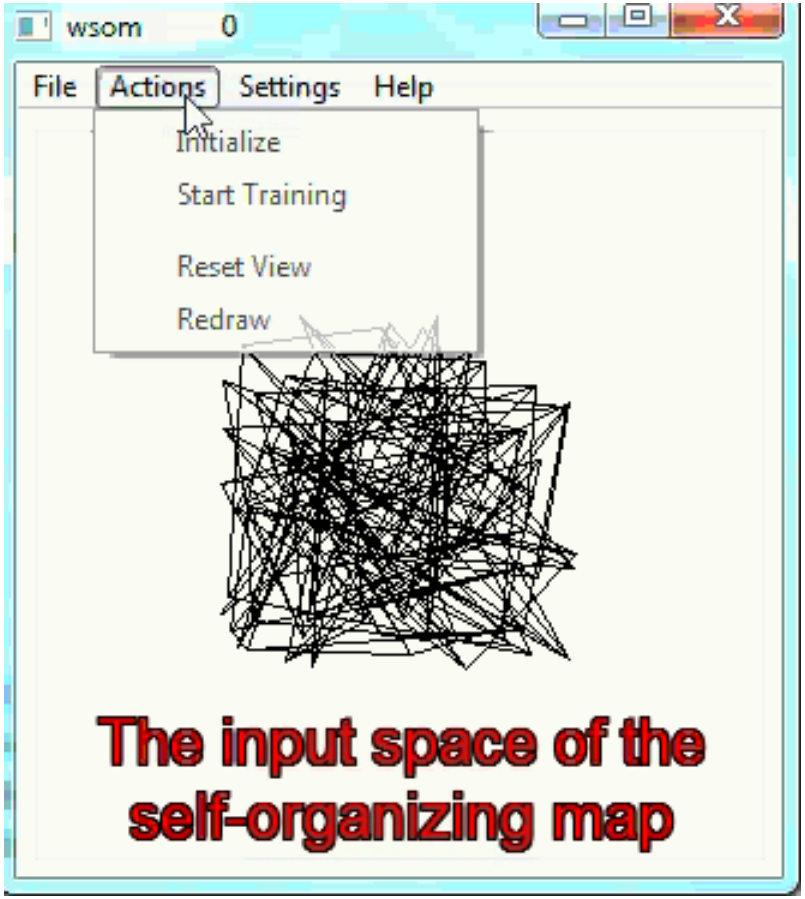
Examples: One-dimensional Lattice of Kohonen Elements

- Units are arranged in sequence

- Each unit learns to specialize
  for different regions of input space [Rojas]



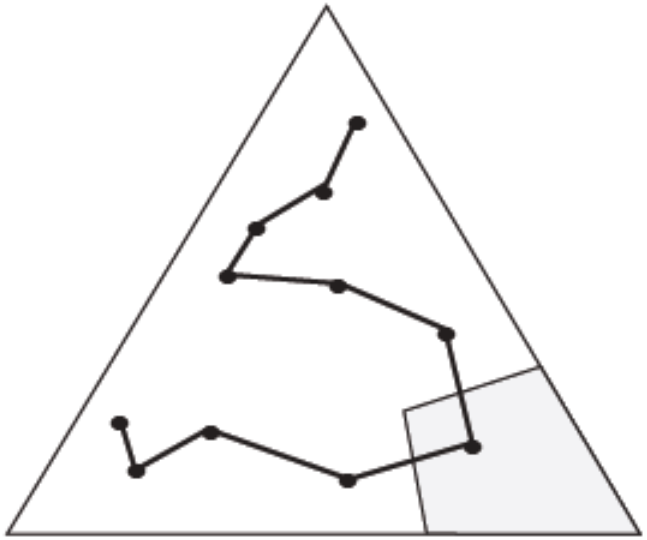
SOM Demo



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

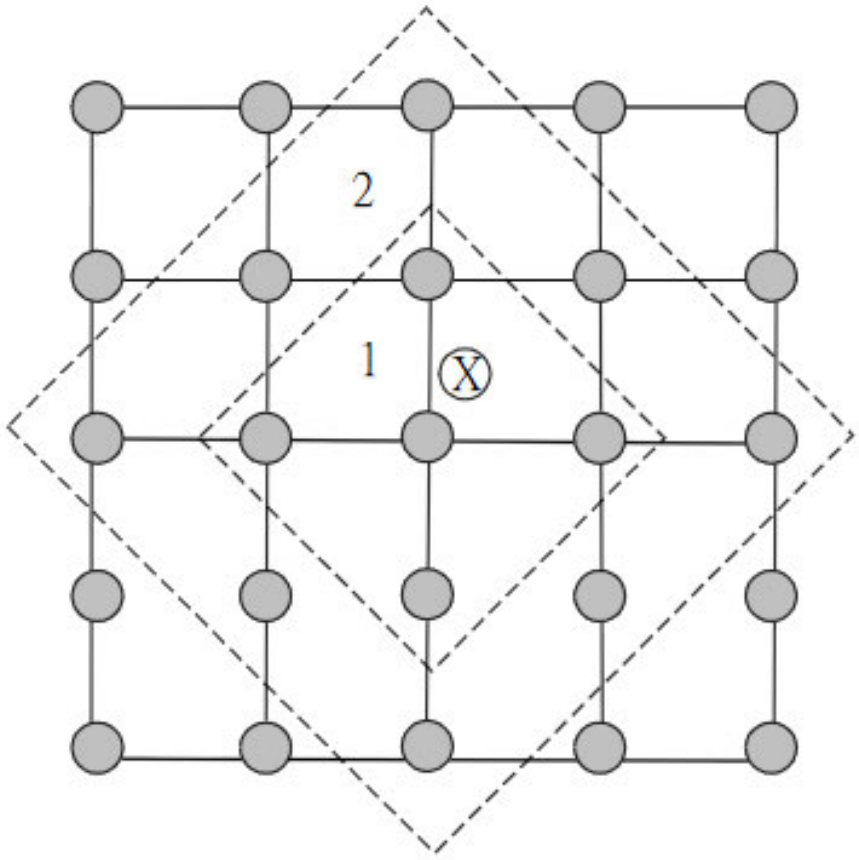
Map of a Triangular Region

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



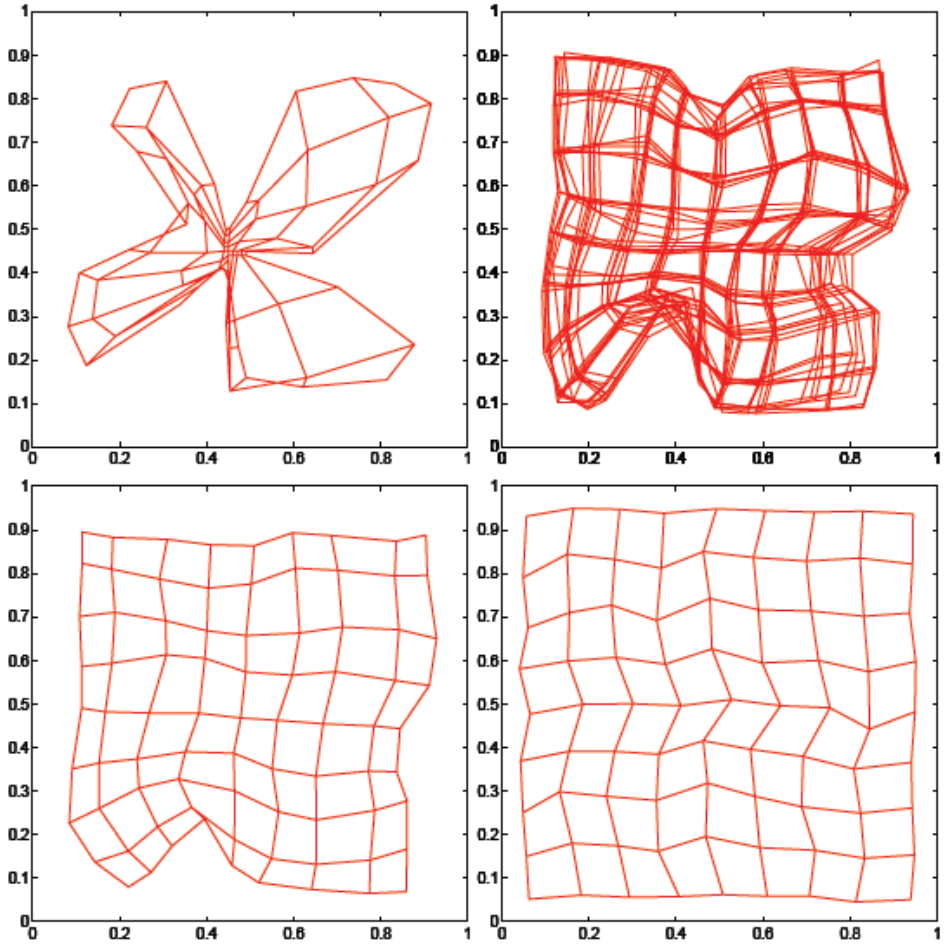
Unit with strongest excitation for shaded region

Example of SOM Neighborhood



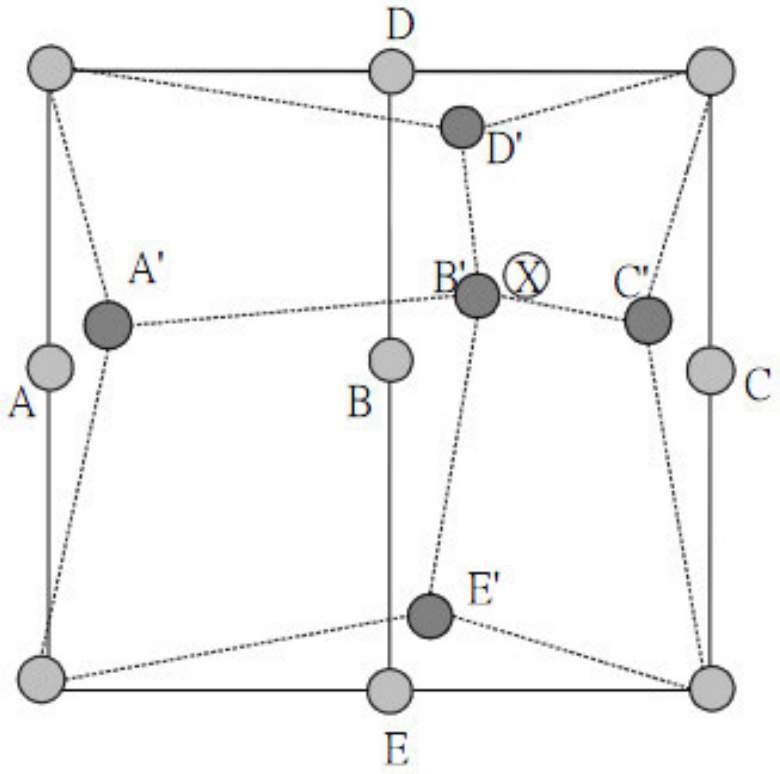
The input vector is represented as X.
Units in neighbourhood 1 are more active than those in neighbourhood 2.

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



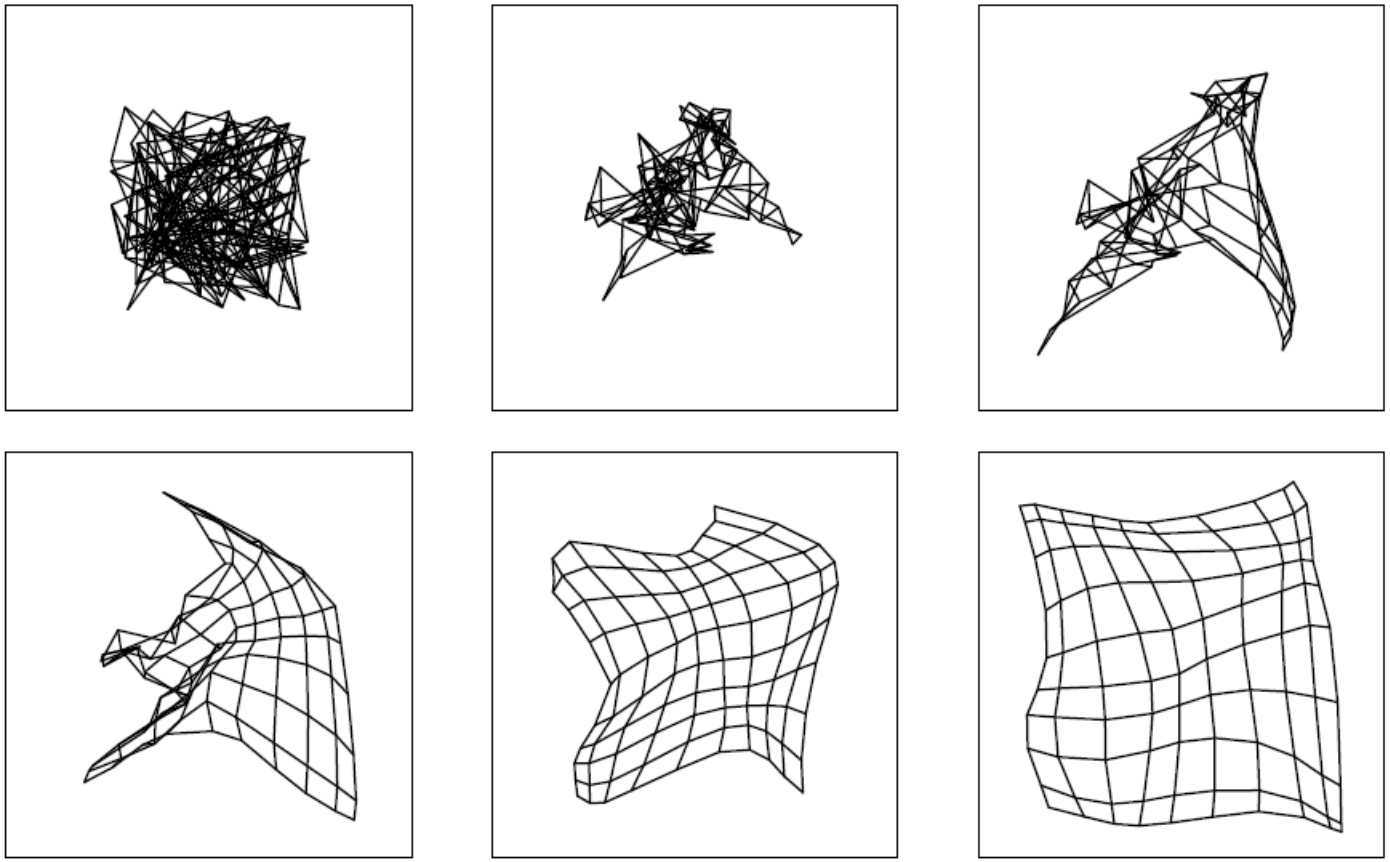
- Upper right with some overlapped learning iterations
- Results for 100, 1000, 5000, 10000 iterations (Kohonen 1984)

Influence by Pre-Defined Topology

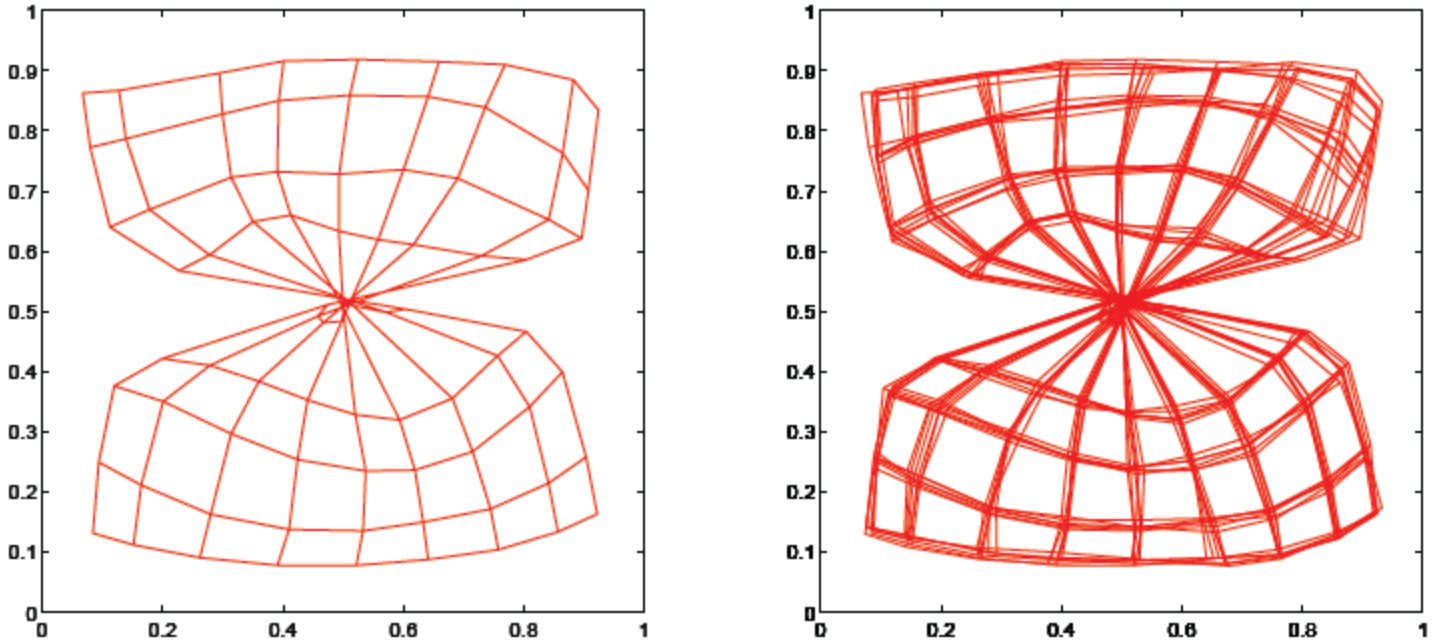


- The winner B and its neighbors such as A, C, D and E move towards the input vector X
- Modified units are shown as dark circles

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

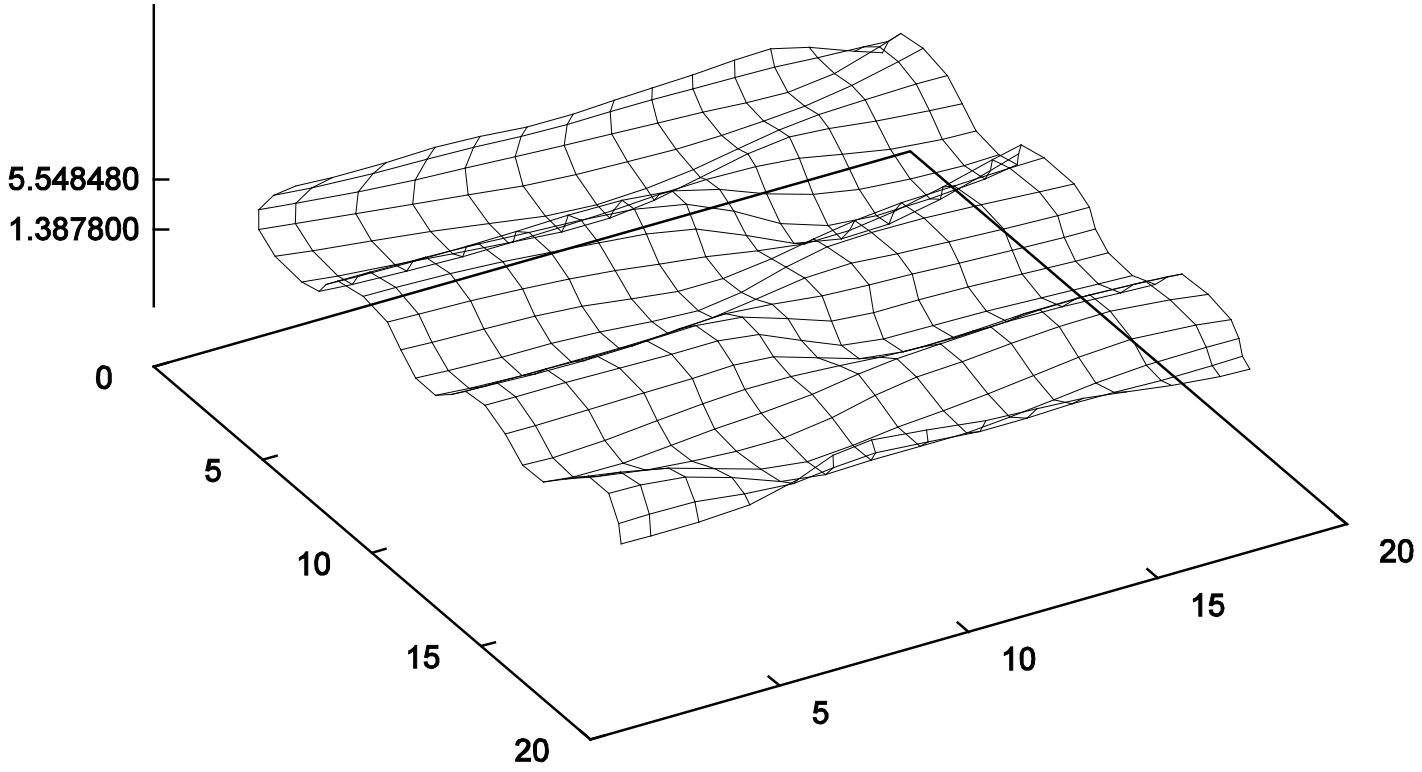


Planar Network with a Knot



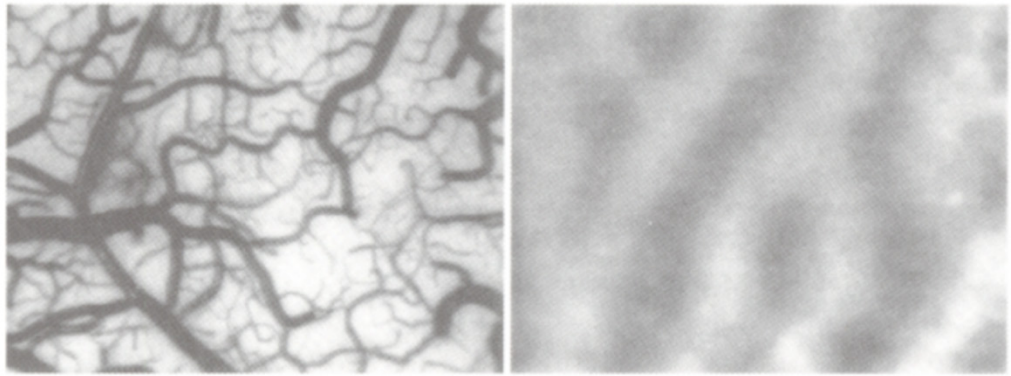
State difficult to correct

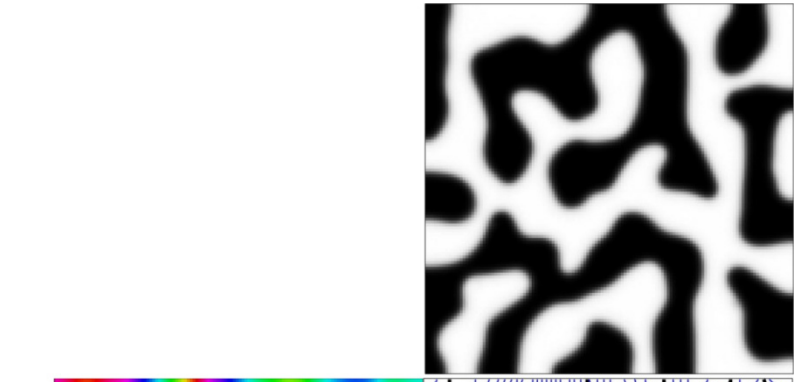
Unfolding of a 2-D Map in a 3-D Data Space

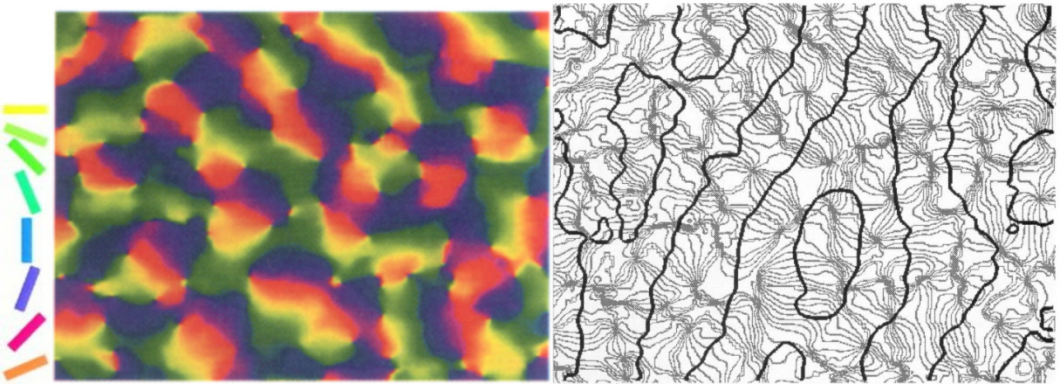


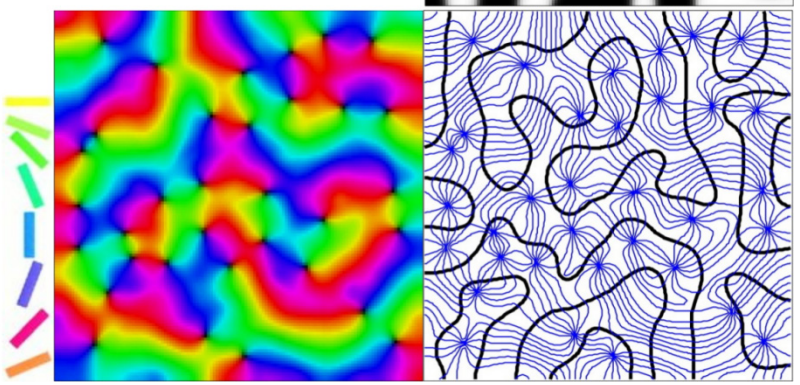
Example Application: V1 Maps

ocular dominance









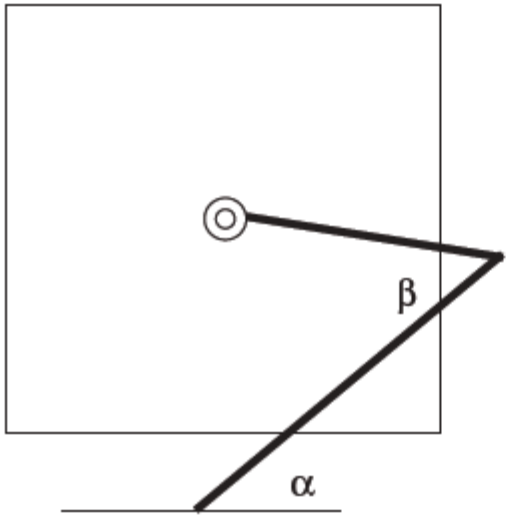
orientation preference

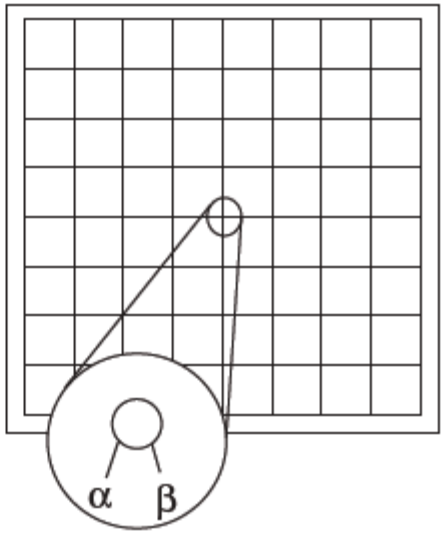
Obermayer, Blasdel. .. Orientation and Ocular Dominance Columns in Monkey .. Jneurosci, 1993

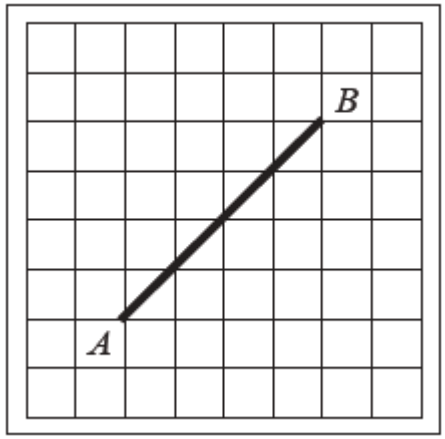
Goodhill . Theoretical Modelling to .. Neural Map Development. Neuron, 2007

Learning Simple Inverse Kinematics

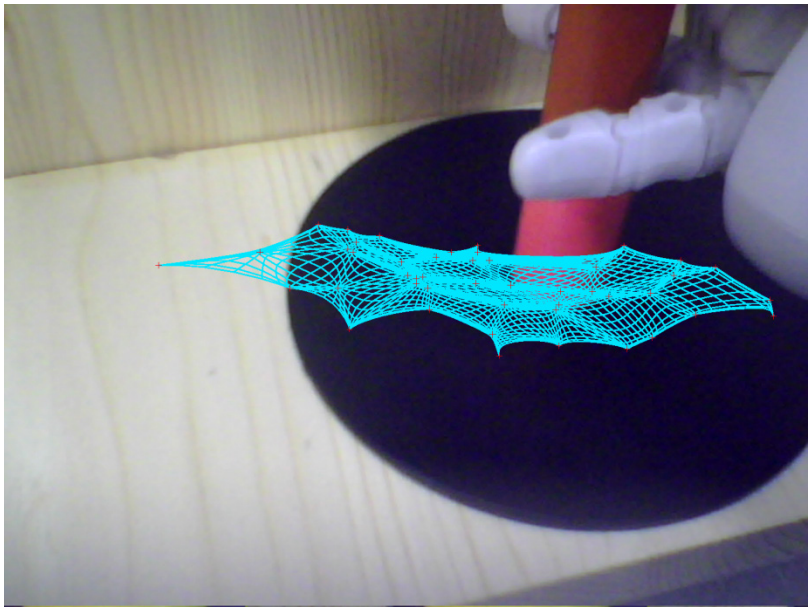
- Mapping the configuration space of a robot arm using 2-dimensional network
- For one point only one parameter combination; many paths from A to B

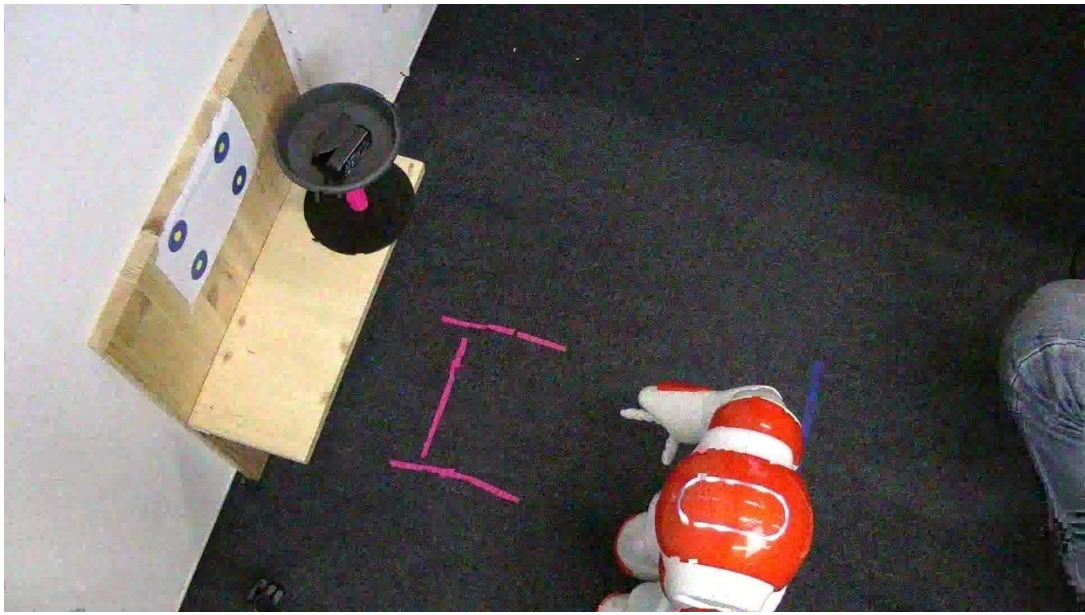






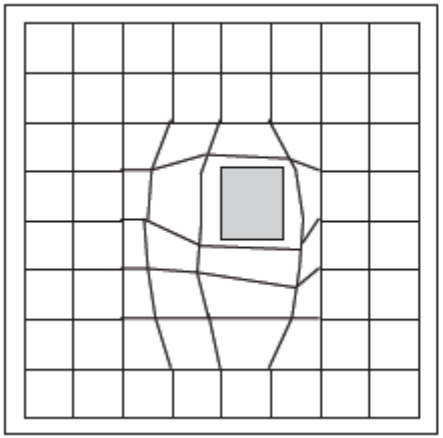
Learning Simple Inverse Kinematics

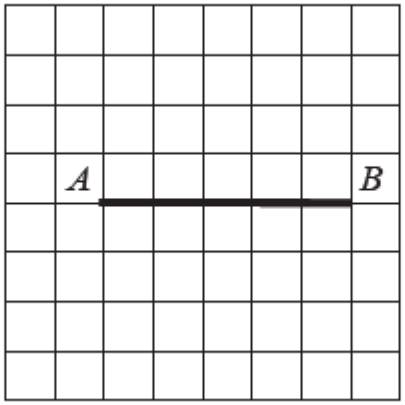


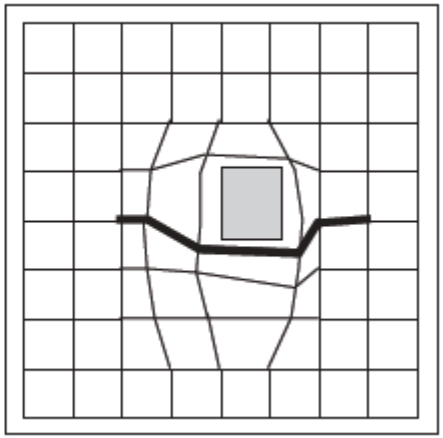


Learning Obstacles in Work Area

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





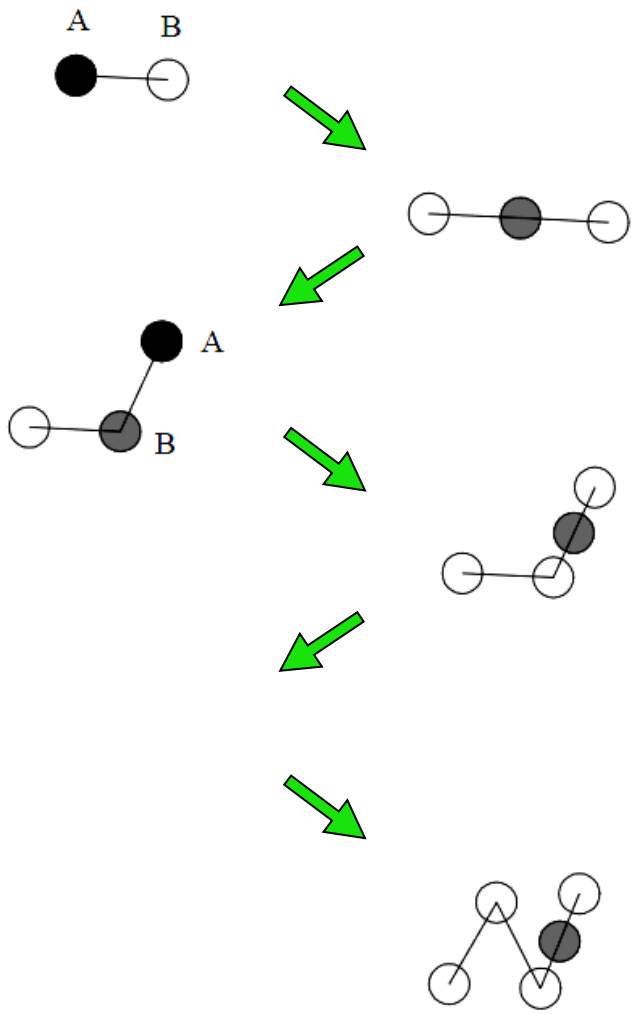


Existing Neural Clustering models

- ***Static Models*** (fixed number of units): Competitive Learning (CL), Self-Organizing Map (SOM), Neural Gas (NG)….
- ***Dynamic Models*** (variable number of units): Growing Grid (GG), Growing Cell Structure (GCS), Growing Neural Gas (GNG), Grow When Required (GWR), etc.
- ***Hierarchical Models***: Multilayered Self-Organising Feature Maps (M-SOM), Growing Hierarchical Self-Organizing Map (GHSOM), etc.

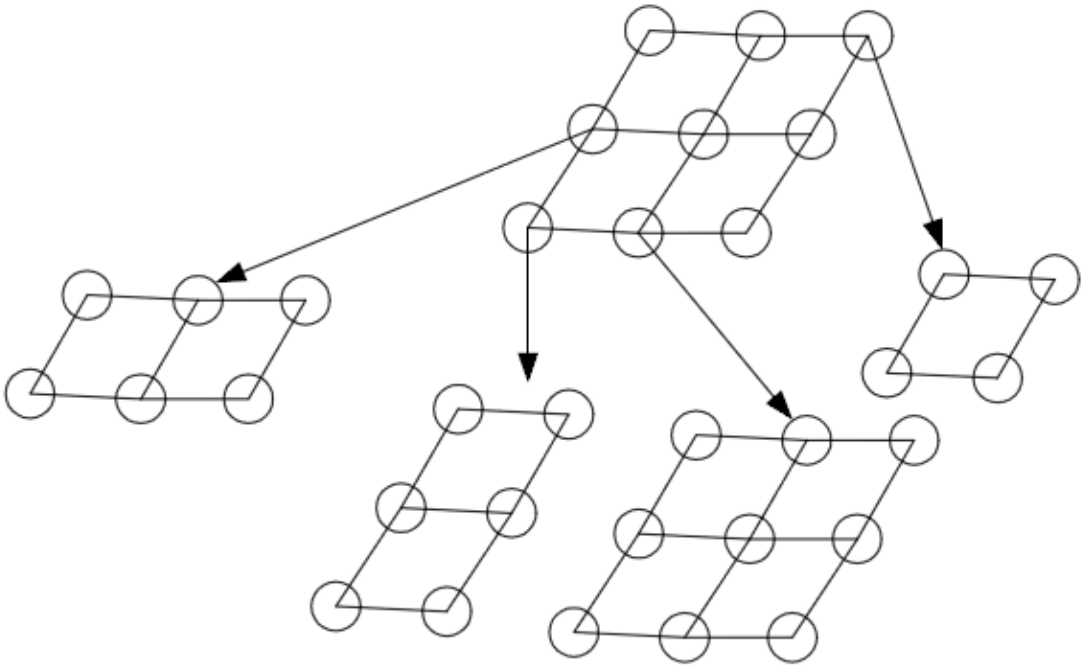
But there are several shortcomings for them in a non-stationary environment.

Growing Processes for DASH Model (Growing Neural Gas – GNG Fritzke)



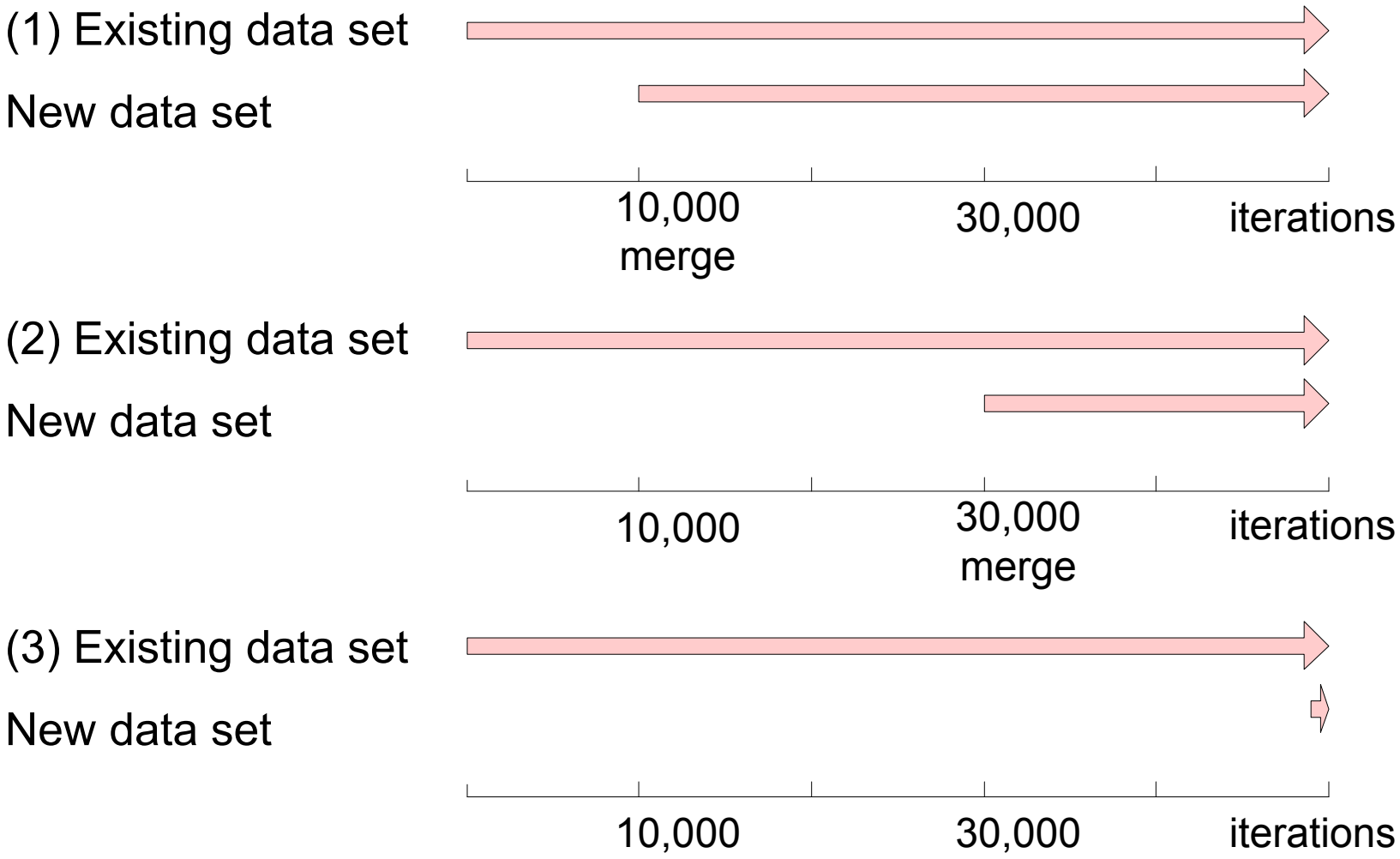
- GNG adds unit after every pre-defined period.
- Units are represented as circles.
- Circle A indicates ***unit with biggest error***
- Circle B indicates ***neighbor with biggest error*** for Circle *A.*
- The grey circle is the ***new unit*** at each stage.

Growing Hierarchical SOM
(Rauber et al)



A sub-map ***grows*** from a unit whose error is greater than a pre-defined proportion of the expected unit error

Dynamical SOM for Dynamical Knowledge Acquisition Over Time

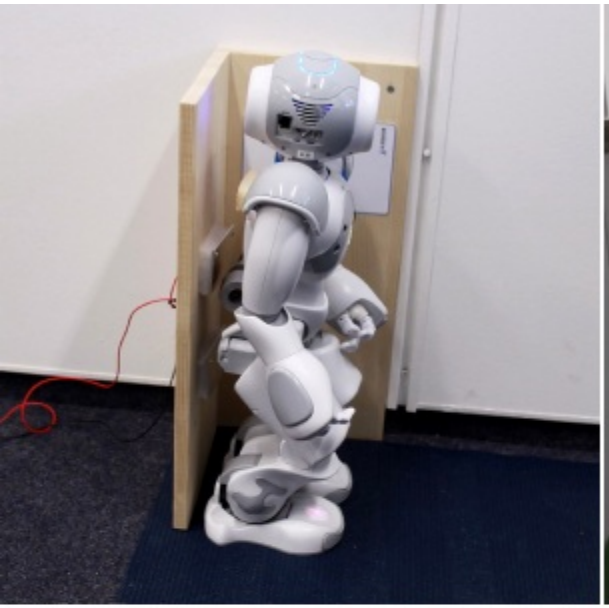


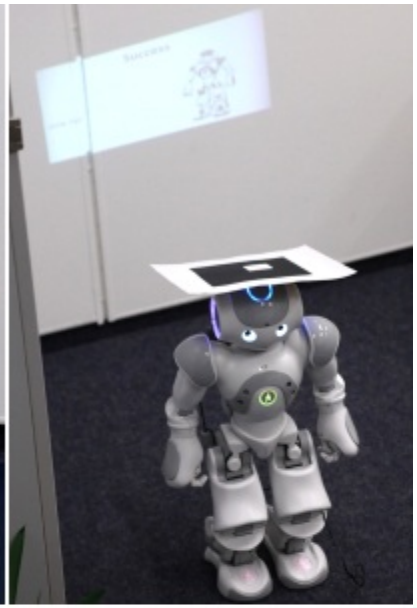
Shortcomings of static Models in a non-stationary Environment

- 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)

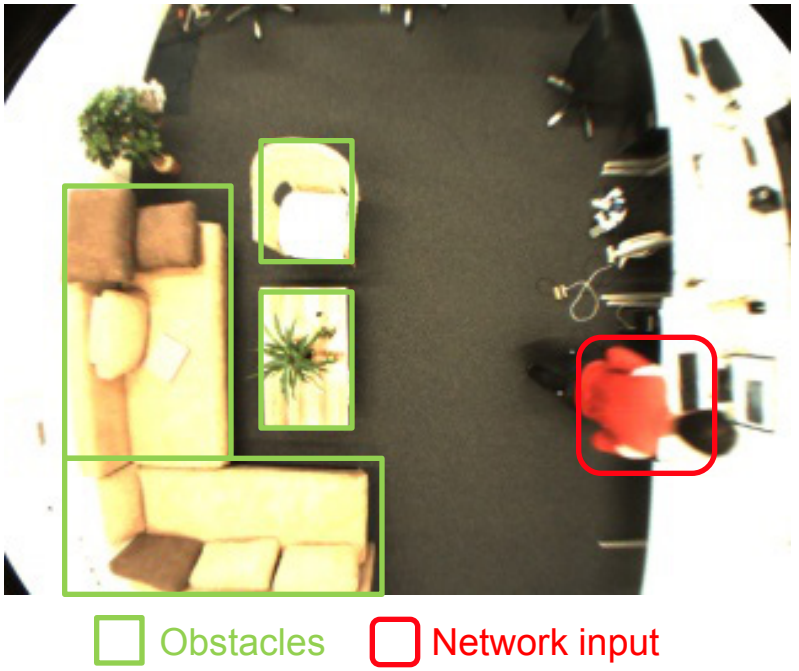




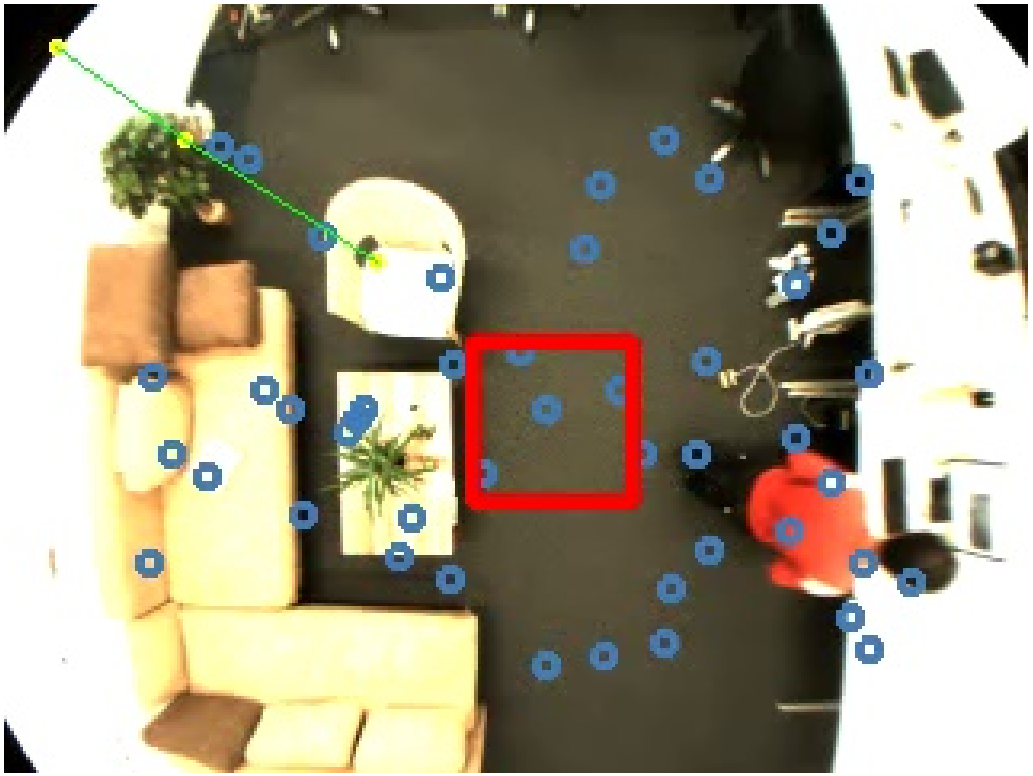
Room Mapping via GNG (KT Lab)

- 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

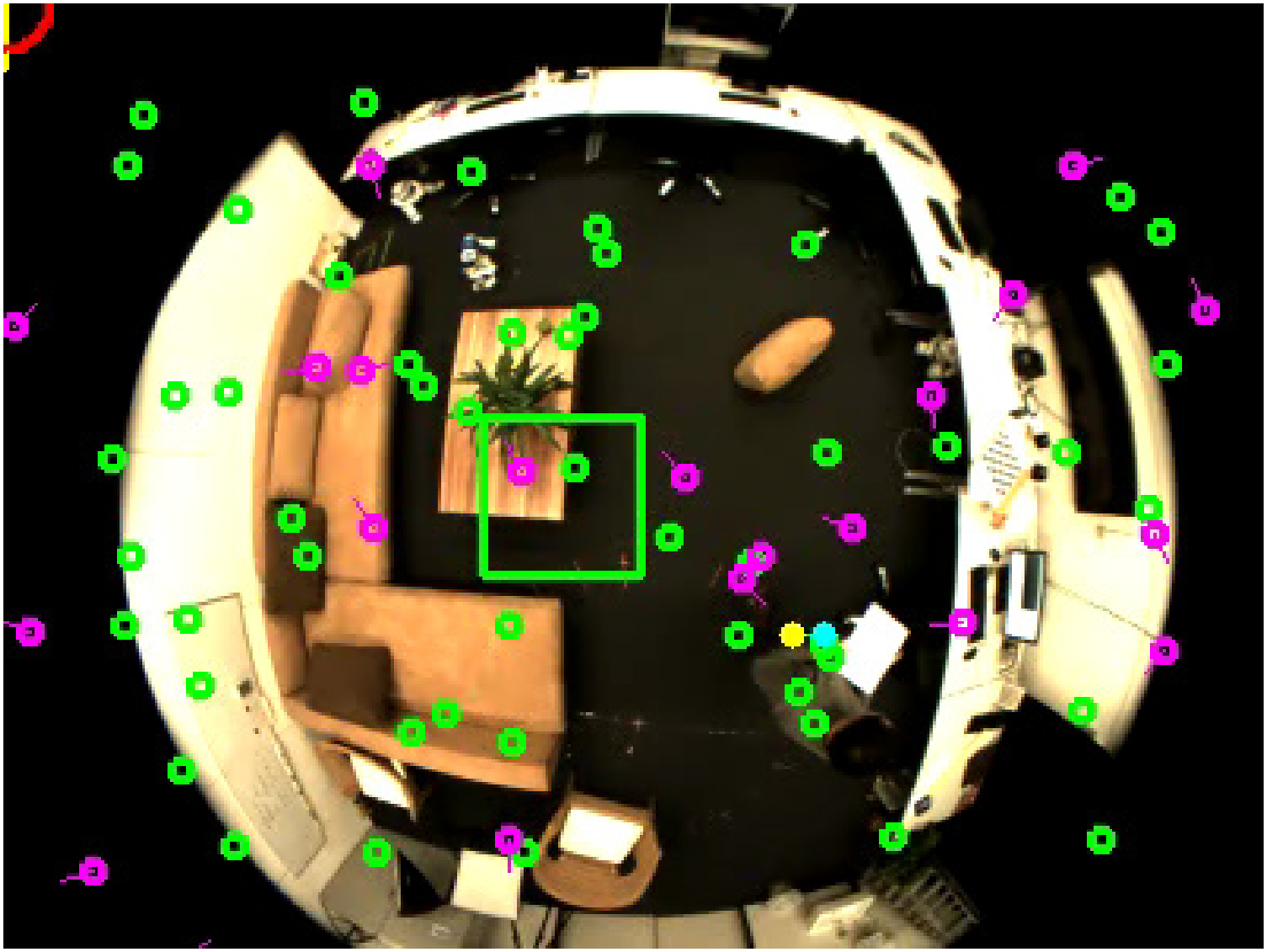


Room Mapping via GNG (KT Lab)



- 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

Room Mapping via GNG (KT Lab)



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