Machine Learning

Lecture 6

Unsupervised learning. Clustering.

Self-organizing maps of Kohonen

Unsupervised learning

- Without teacher
  - Example (instance) is only input data
- It is basics of learning in nature (animals) (for mammals and human beings essentially in early time of life)
- Based on definition of order in data
- Closely connecting with data mining
- Closely connecting with concept of clustering

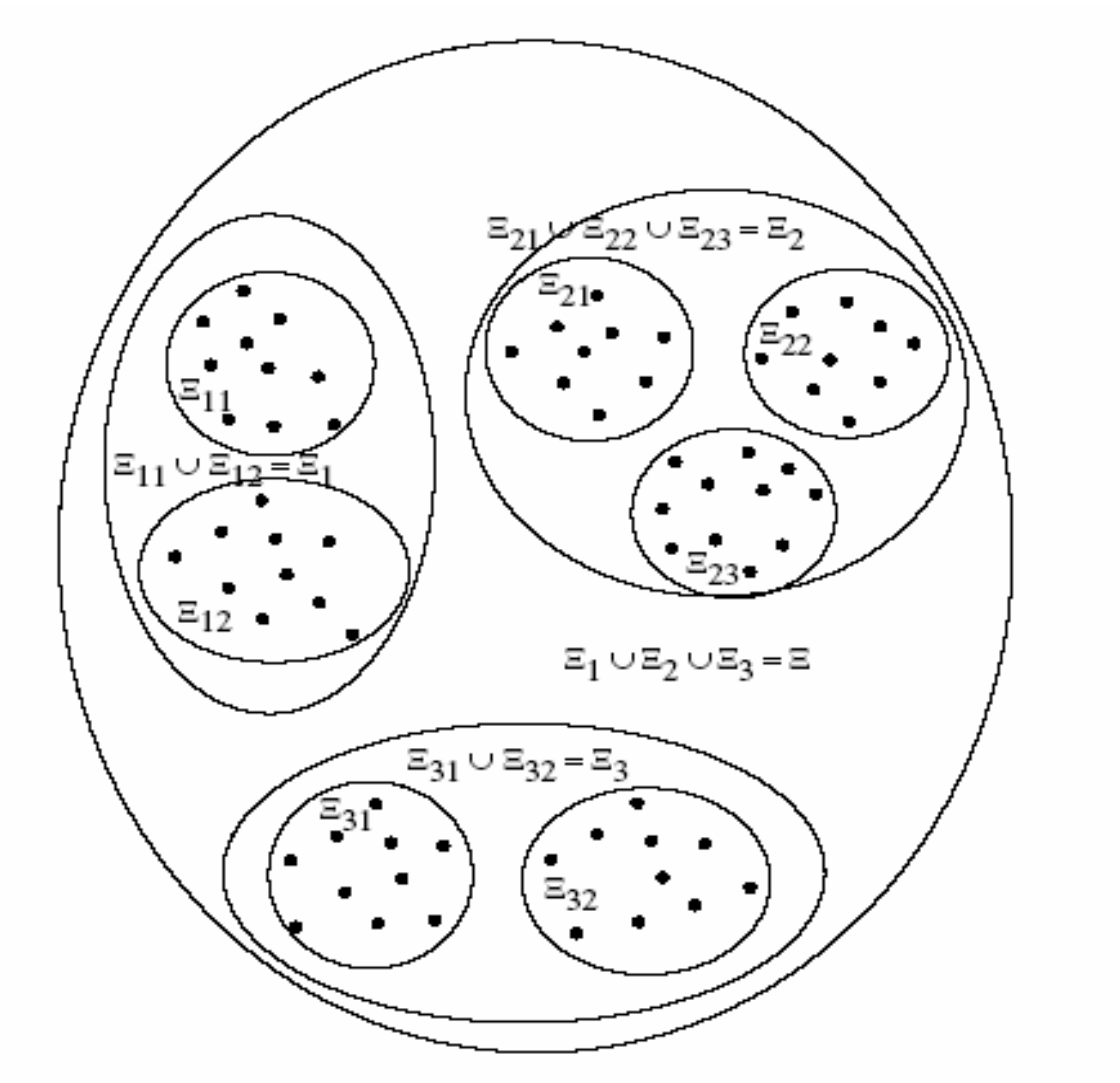
Clustering

- Clustering is the process of partitioning a set of objects into subsets based on some measure of similarity (or dissimilarity) between pairs of the objects
- Partition unlabeled examples into disjoint subsets of *clusters*, such that:
  - Examples within a cluster are very similar
  - Examples in different clusters are very different

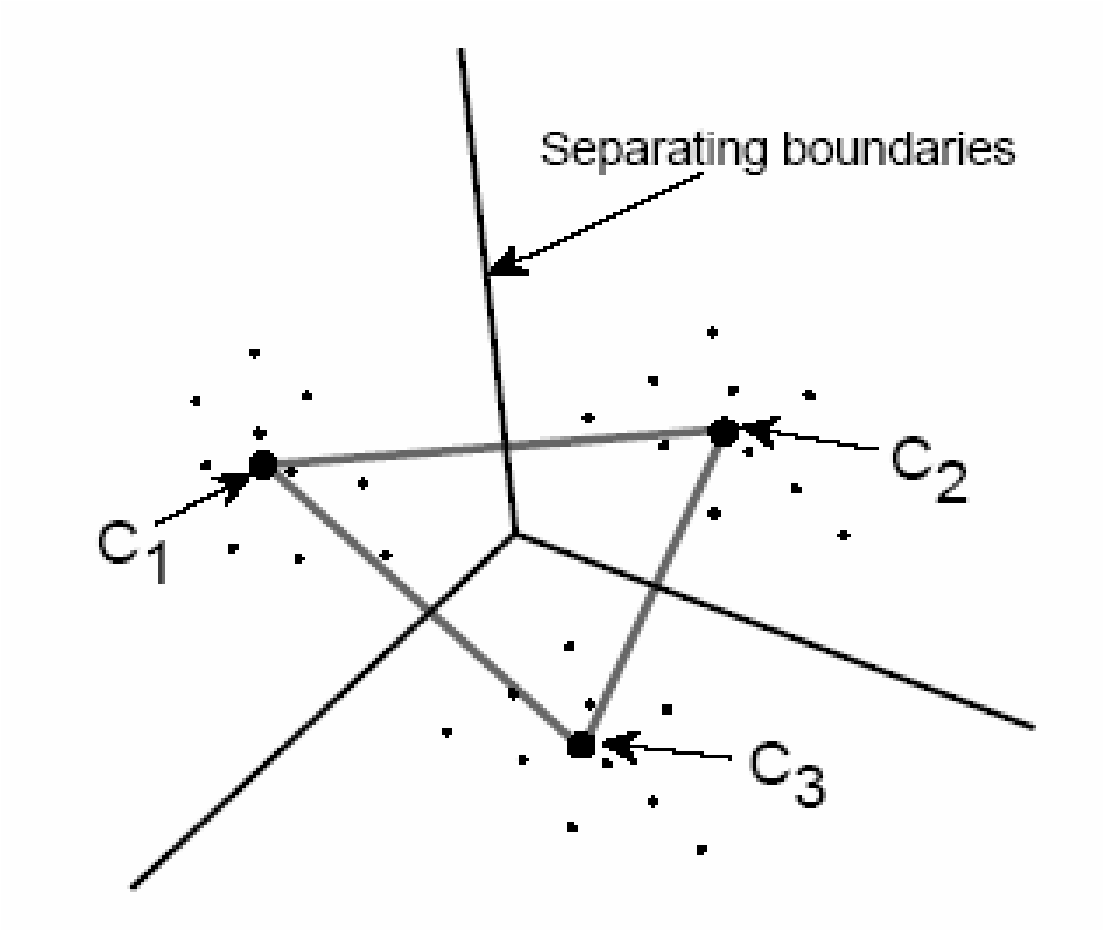
Cluster analysis

- Goals
  - Organize information about data so that relatively homogeneous groups (clusters) are formed and describe their unknown properties.
  - Find useful and interesting groupings of samples.
  - Find representatives for homogeneous groups.
- Two components of cluster analysis.
  - The (dis)similarity measure between two data samples.
  - The clustering algorithm.

Hierarchy of clusters



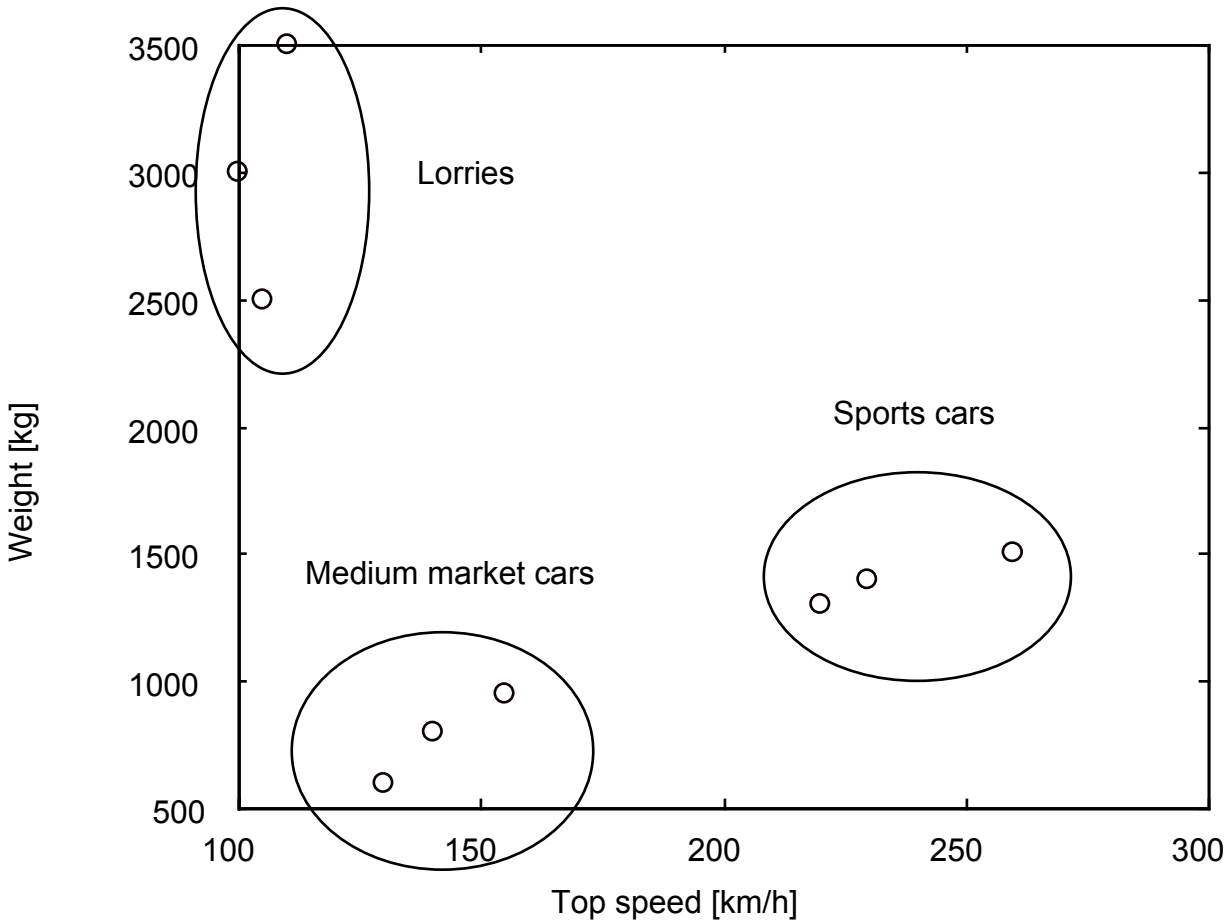
Minimum-Distance Clustering



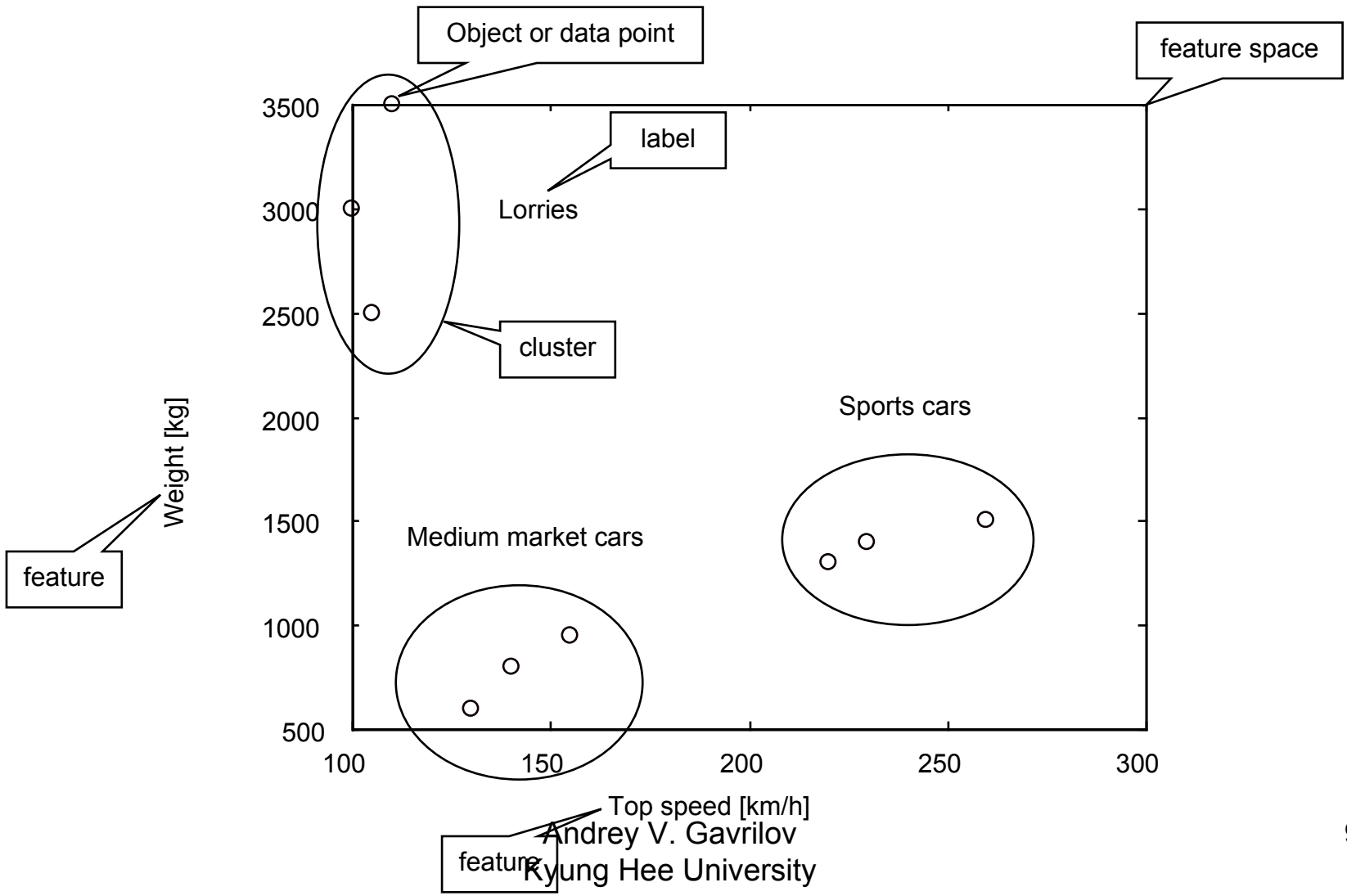
Vehicle Example

Vehicle	Top speed km/h	Colour	Air resistance	Weight Kg
V1	220	red	0.30	1300
V2	230	black	0.32	1400
V3	260	red	0.29	1500
V4	140	gray	0.35	800
V5	155	blue	0.33	950
V6	130	white	0.40	600
V7	100	black	0.50	3000
V8	105	red	0.60	2500
V9	110	gray	0.55	3500

Vehicle Clusters



Terminology



Distance/Similarity Measures

- **Minkowski** metric

$$dist(\mathbf{x}_i, \mathbf{x}_k) = \left(\sum_{j=1}^{d} \left| x_{ij} - x_{kj} \right|^r \right)^{1/r} \qquad \mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$$

1. $\forall i,\ dist(i,i) = 0$
2. $\forall (i,k)\quad dist(i,k) \geq 0$
3. $\forall (i,k)\quad dist(i,k) = dist(k,i)$
4. $\forall (i,k,m)\quad dist(i,k) \leq dist(i,m) + dist(m,k)$

Distance/Similarity Measures (2)

- Common Minkowski metrics
  - Euclidean distance: $r = 2$

$$dist(i,k) = \left[\sum_{j=1}^{d} (x_{ij} - x_{kj})^2 \right]^{1/2}$$

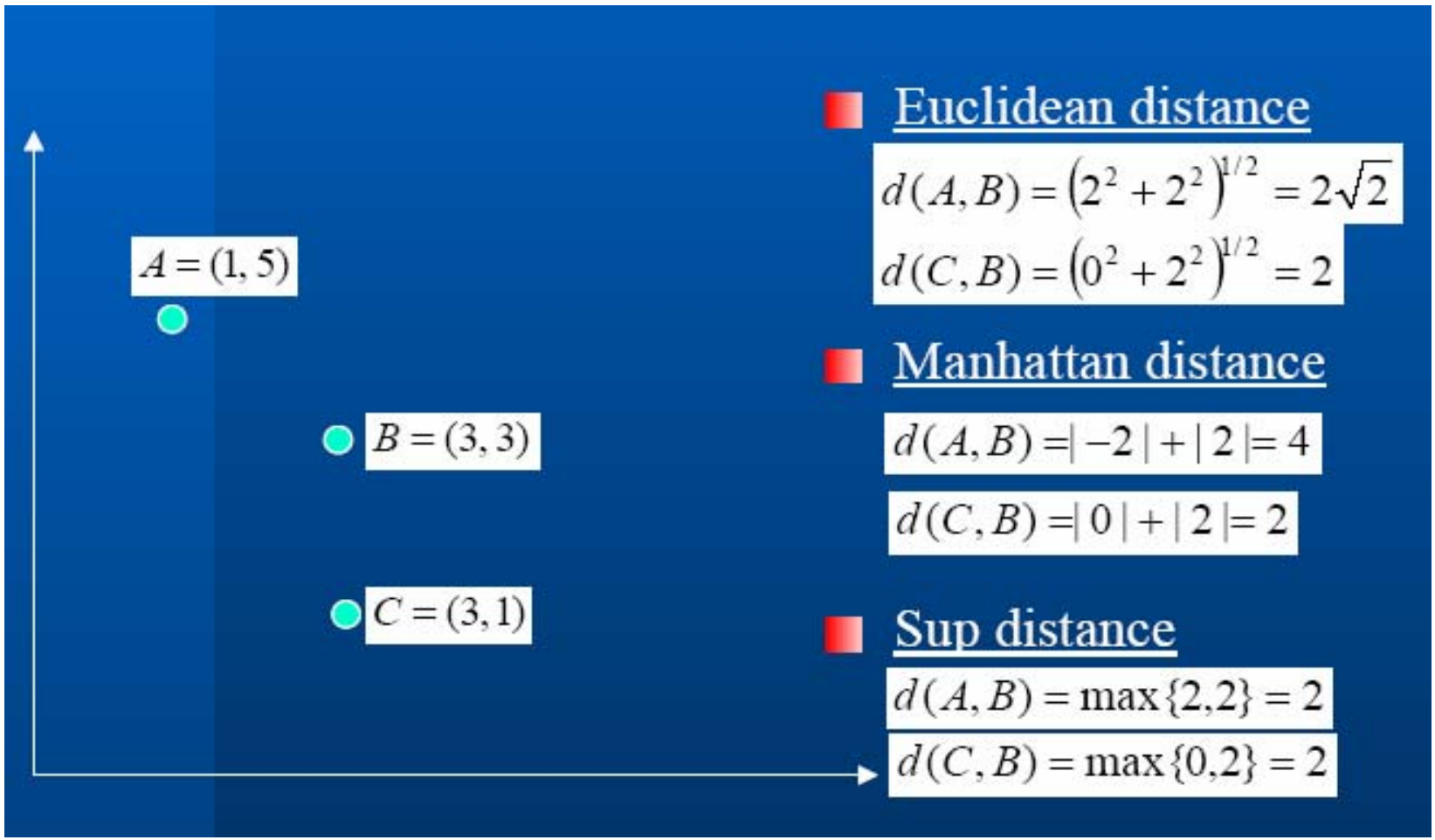
  - Manhattan distance: $r = 1$

$$dist(i,k) = \sum_{j=1}^{d} \left| x_{ij} - x_{kj} \right|$$

    - If all the features are binary, Hamming distance.
  - Sup distance: $r \to \infty$

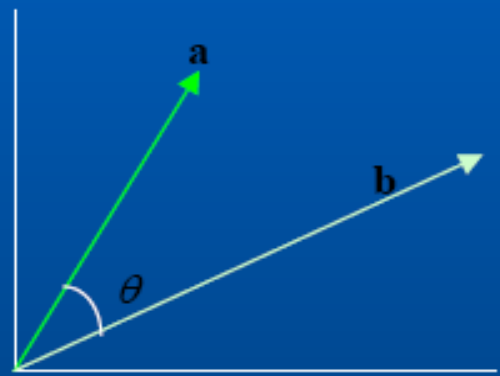
$$dist(i,k) = \max_{1 \le j \le d} \left| x_{ij} - x_{kj} \right|$$

Distance/Similarity Measures (3)



Distance/Similarity Measures (4)

- Cosine similarity
  - Estimate the angle between two data items represented in vector space.



$$SIM(\mathbf{a},\mathbf{b}) = \cos\theta = \frac{\mathbf{a} \bullet \mathbf{b}}{|\mathbf{a}| \times |\mathbf{b}|}$$

- Pearson correlation coefficient
  - Represent linear relationship between two data items.

$$SIM(\mathbf{a},\mathbf{b}) = \frac{\sum_{i=1}^{d} \mathbf{a}_i \mathbf{b}_i - \frac{1}{d}\sum_{i=1}^{d} \mathbf{a}_i \sum_{j=1}^{d} \mathbf{b}_j}{\sqrt{\sum_{i=1}^{d} \mathbf{a}_i^2 - \frac{1}{d}\left(\sum_{i=1}^{d} \mathbf{a}_i\right)^2}\sqrt{\sum_{i=1}^{d} \mathbf{b}_i^2 - \frac{1}{d}\left(\sum_{i=1}^{d} \mathbf{b}_i\right)^2}}$$

Clustering Algorithms

- Hierarchy
  - ♦Hierarchical clustering
  - ♦Non-hierarchical (flat) clustering
- Underlying model assumption
  - ♦Parametric clustering
  - ♦Non-parametric clustering
- Strictness
  - ♦Hard clustering
  - ♦Soft clustering

Neural networks for clustering

ART (Adaptive resonance Theory)

SOM (Self-organizing Maps)

The phrase "Kohonen network" most often refers to one of the following three types of networks:

- VQ: Vector Quantization--competitive networks that can be viewed as unsupervised density estimators or autoassociators (Kohonen, 1995/1997; Hecht-Nielsen 1990), closely related to k-means cluster analysis (MacQueen, 1967; Anderberg, 1973).
- LVQ: Learning Vector Quantization--competitive networks for supervised classification (Kohonen, 1988, 1995; Ripley, 1996). Each codebook vector is assigned to one of the target classes. Each class may have one or more codebook vectors. A case is classified by finding the nearest codebook vector and assigning the case to the class corresponding to the codebook vector. Hence LVQ is a kind of nearest-neighbor rule.
- SOM: Self-Organizing Map--competitive networks that provide a "topological" mapping from the input space to the clusters (Kohonen, 1995). The SOM was inspired by the way in which various human sensory impressions are neurologically mapped into the brain such that spatial or other relations among stimuli correspond to spatial relations among the neurons.

VQ

- Each competitive unit corresponds to a cluster, the center of which is called a "codebook vector". Kohonen's learning law is an on-line algorithm that finds the codebook vector closest to each training case and moves the "winning" codebook vector closer to the training case. The codebook vector is moved a certain proportion of the distance between it and the training case, the proportion being specified by the learning rate, that is:

  new_codebook = old_codebook * (1-learning_rate) + data * learning_rate

VQ (2)

- MacQueen's on-line k-means algorithm is essentially the same as Kohonen's learning law except that the learning rate is the reciprocal of the number of cases that have been assigned to the winning cluster. Suppose that when processing a given training case, N cases have been previously assigned to the winning codebook vector. Then the codebook vector is updated as:

  new_codebook = old_codebook * N/(N+1) + data * 1/(N+1)

VQ (3)

- This reduction of the learning rate makes each codebook vector the mean of all cases assigned to its cluster and guarantees convergence of the algorithm to an optimum value of the error function (the sum of squared Euclidean distances between cases and codebook vectors) as the number of training cases goes to infinity.
- Kohonen's learning law with a fixed learning rate does not converge. As is well known from stochastic approximation theory, convergence requires the sum of the infinite sequence of learning rates to be infinite, while the sum of squared learning rates must be finite (Kohonen, 1995, p. 34).
- These requirements are satisfied by MacQueen's k-means algorithm.

LVQ

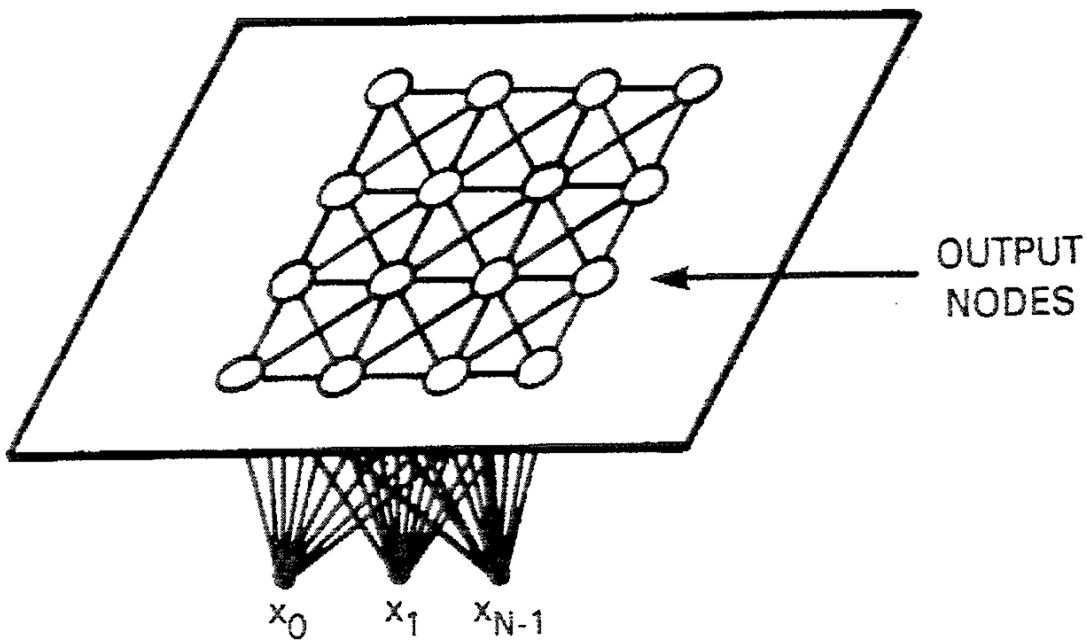
- Ordinary VQ methods, such as Kohonen's VQ or k-means, can easily be used for supervised classification. Simply count the number of training cases from each class assigned to each cluster, and divide by the total number of cases in the cluster to get the posterior probability. For a given case, output the class with the greatest posterior probability--i.e. the class that forms a majority in the nearest cluster. Such methods can provide universally consistent classifiers (Devroye et al., 1996) even when the codebook vectors are obtained by unsupervised algorithms.
- LVQ tries to improve on this approach by adapting the codebook vectors in a supervised way. There are several variants of LVQ--called LVQ1, OLVQ1, LVQ2, and LVQ3--based on heuristics.

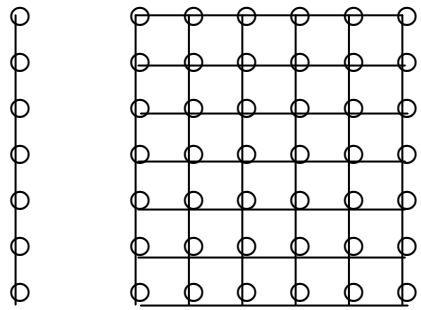
Self Organizing Maps (SOM)

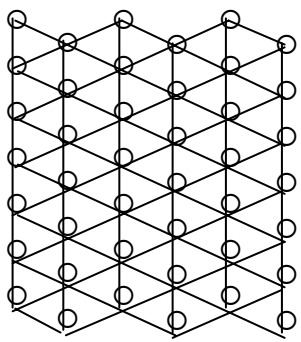
- Based on competitive learning (Unsupervised)
  - Only one output neuron activated at any one time
  - Winner-takes-all neuron or winning neuron
- In a Self-Organizing Map
  - Neurons placed at the nodes of a lattice
    - one or two dimensional (rarely more)
  - Neurons selectively tuned to input patterns
    - by a competitive learning process
  - Locations of neurons so tuned to be ordered
    - formation of topographic map of input patterns
  - Spatial locations of the neurons in the lattice -> intrinsic statistical features contained in the input patterns

Self Organizing Maps

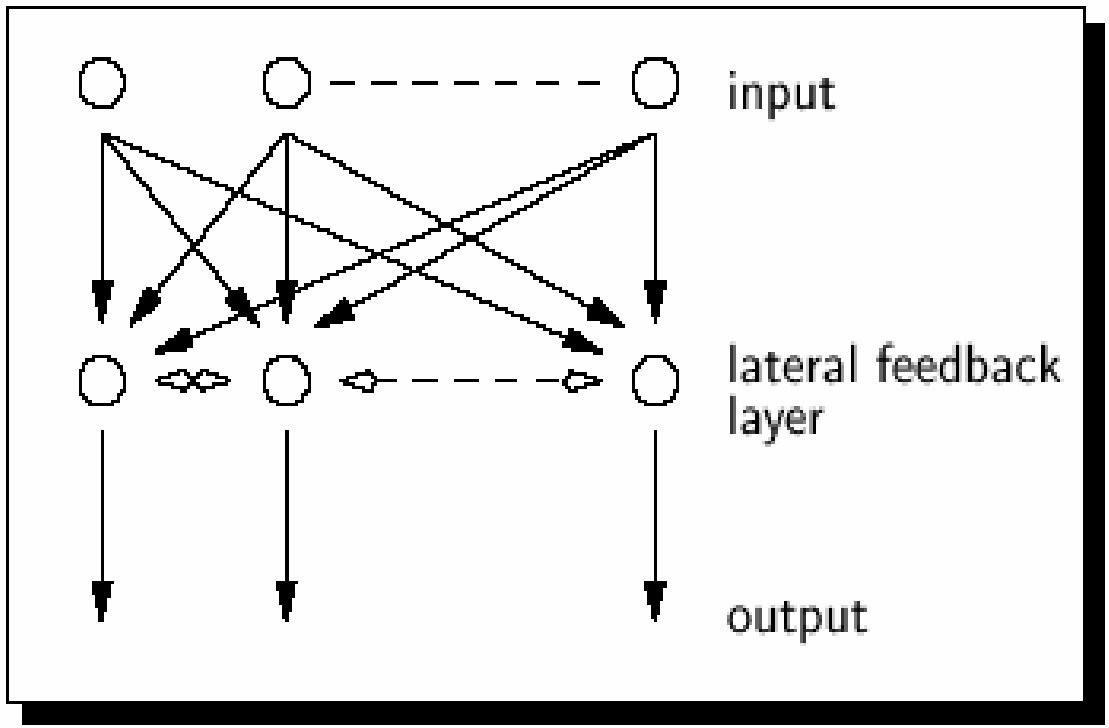
- Topology-preserving transformation







Structure of Kohonen's maps



Formation Process of SOM

- After initialization for synaptic weights, there are three essential processes
  - Competition
    - Largest value of discriminant function selected
    - Winner of competition
  - Cooperation
    - Spatial neighbors of winning neuron is selected
  - Synaptic adaptation
    - Excited neurons adjust synaptic weights

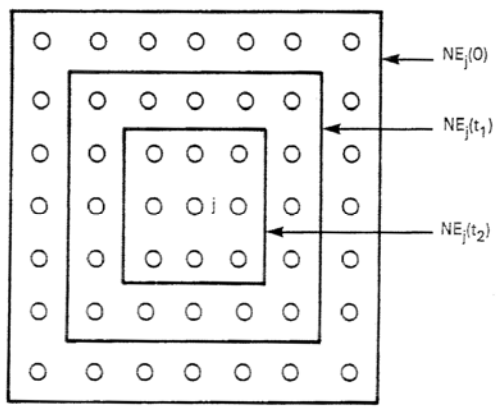
Competitive Process

- Input vector, synaptic weight vector
  - $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$
  - $\boldsymbol{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T$, $j = 1, 2, 3, l$
- Best matching, winning neuron
  - $i(\boldsymbol{x}) = \arg\min \|\boldsymbol{x} - \boldsymbol{w}_j\|$, $j = 1, 2, 3, .., l$
- Determine the location where the topological neighborhood of excited neurons is to be centered

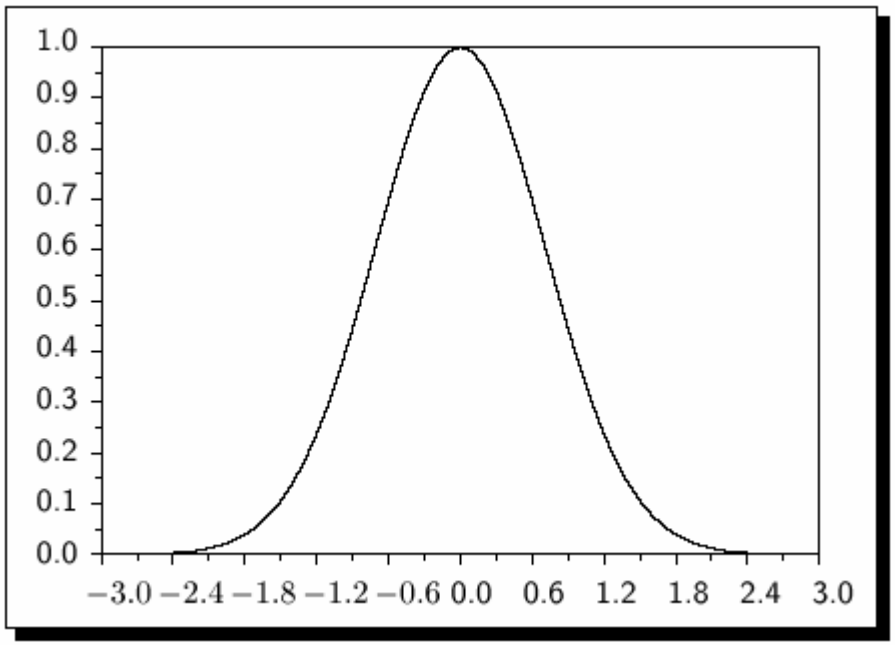
Cooperative Process

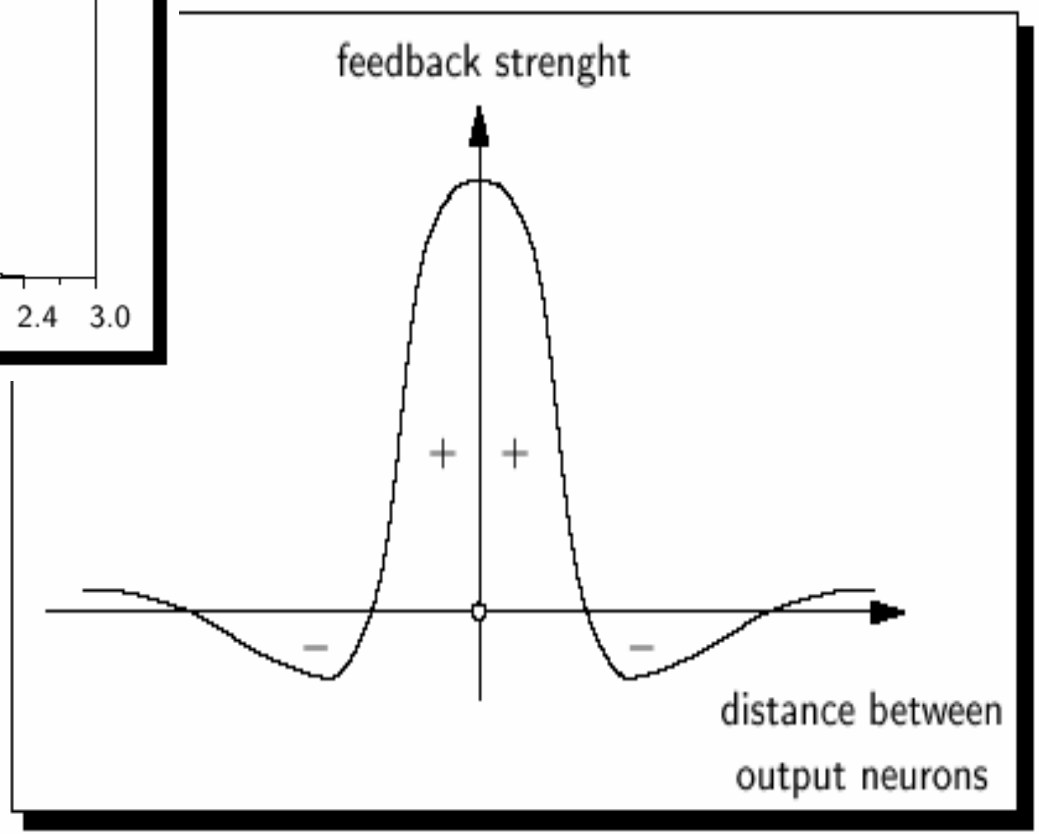
- For a winning neuron, the neurons in its immediate neighborhood excite more than those farther away
- topological neighborhood decay smoothly with lateral distance
  - Symmetric about maximum point defined by $d_{ij} = 0$
  - Monotonically decreasing to zero for $d_{ij} \rightarrow \infty$
  - Neighborhood function: Gaussian case $h_{j,i(x)} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right)$
- Size of neighborhood shrinks with time

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right), \quad n = 0,1,2,3$$



Examples of neighborhood function





Adaptive process

- Synaptic weight vector is changed in relation with input vector

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta(n)\, h_{j,i(x)}(n)\, (\boldsymbol{x} - \boldsymbol{w}_j(n))$$

- Applied to all neurons inside the neighborhood of winning neuron *i*
- Upon repeated presentation of the training data, weight tend to follow the distribution
- Learning rate $\eta(n)$: decay with time
- May decompose two phases
  - Self-organizing or ordering phase : topological ordering of the weight vectors
  - Convergence phase : after ordering, for accurate statistical quantification of the input space

Summary of SOM

- Continuous input space of activation patterns that are generated in accordance with a certain probability distribution
- Topology of the network in the form of a lattice of neurons, which defines a discrete output space
- Time-varying neighborhood function defined around winning neuron
- Learning rate decrease gradually with time, but never go to zero

Summary of SOM(2)

- Learning Algorithm
  - 1. Initialize *w*'s
  - 2. Present input vector
  - 3. Find nearest cell

$$i(\underline{x}) = argmin_j \| \underline{x}(n) - \underline{w}_j(n) \|$$

  - 4. Update weights of neighbors

$$\underline{w}_j(n+1) = \underline{w}_j(n) + \eta(n)\, h_{j,i(x)}(n) [\, \underline{x}(n) - \underline{w}_j(n) \,]$$

  - 5. Reduce neighbors and η
  - 6. Go to 2

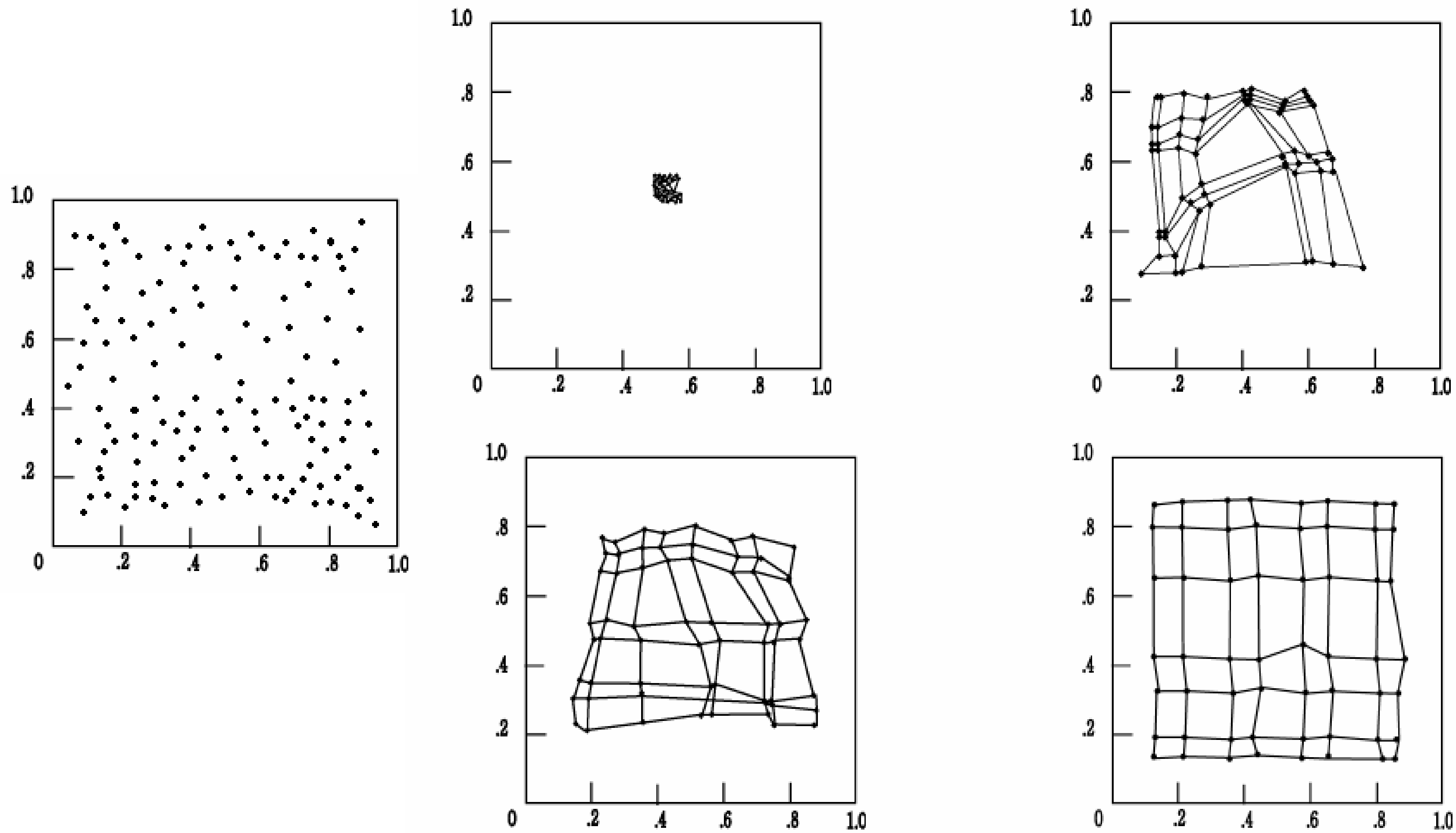
$h_{j,i(x)}(n)$ - the neighborhood function that has value 1 when i=k and falls off with the distance $| r_{i(x)} - r_i |$ between units j and i(x) in the output array.

Example of the neighborhood function is:

$$\aleph(i, k) = e^{\left(-|r_k - r_i|^2\right)/(2\sigma^2)}$$

where σ^2 is the width parameter that can gradually be decreased over time.

SOFM Example
2-D Lattice by 2-D distribution



Example of implementation

```
typedef struct
{ /* A LAYER OF A NET: */
 INT Units; /* - number of units in this layer */
 REAL* Output; /* - output of ith unit */
 REAL** Weight; /* - connection weights to ith unit */
 REAL* StepSize; /* - size of search steps of ith unit */
 REAL* dScoreMean; /* - mean score delta of ith unit */ }
 LAYER;
 typedef struct
 { /* A NET: */
 LAYER* InputLayer;
 /* - input layer */
 LAYER* KohonenLayer; /* - Kohonen layer */
 LAYER* OutputLayer; /* - output layer */
 INT Winner; /* - last winner in Kohonen layer */
 REAL Alpha; /* - learning rate for Kohonen layer */
 REAL Alpha_; /* - learning rate for output layer */
 REAL Alpha__; /* - learning rate for step sizes */
 REAL Gamma; /* - smoothing factor for score deltas */
 REAL Sigma; /* - parameter for width of neighborhood */ } NET;
```

```
void PropagateToKohonen(NET* Net)
 { INT i,j;
 REAL Out, Weight, Sum, MinOut;
 for (i=0; i<Net->KohonenLayer->Units; i++)
 { Sum = 0;
 for (j=0; j<Net->InputLayer->Units; j++)
 { Out = Net->InputLayer->Output[j];
 Weight = Net->KohonenLayer->Weight[i][j];
 Sum += sqr(Out - Weight);
 }
 Net->KohonenLayer->Output[i] = sqrt(Sum);
 }
 MinOut = MAX_REAL;
 for (i=0; i<Net->KohonenLayer->Units; i++)
 { if (Net->KohonenLayer->Output[i] < MinOut) MinOut
= Net->KohonenLayer->Output[Net->Winner = i];
 } }
 void PropagateToOutput(NET* Net)
 { INT i; for (i=0; i<Net->OutputLayer->Units; i++)
 { Net->OutputLayer->Output[i] = Net->OutputLayer->Weight[i][Net->Winner];
 } }
 void PropagateNet(NET* Net)
 { PropagateToKohonen(Net);
 PropagateToOutput(Net); }
```

```
REAL Neighborhood(NET* Net, INT i)
 { INT iRow, iCol, jRow, jCol;
 REAL Distance;
 iRow = i / COLS;
 jRow = Net->Winner / COLS;
 iCol = i % COLS;
 jCol = Net->Winner % COLS;
 Distance = sqrt(sqr(iRow-jRow) + sqr(iCol-jCol));
 return exp(-sqr(Distance) / (2*sqr(Net->Sigma)));
 }

  void TrainKohonen(NET* Net, REAL* Input)
   { INT i,j;
   REAL Out, Weight, Lambda, StepSize;
   for (i=0; i<Net->KohonenLayer->Units; i++)
   { for (j=0; j<Net->InputLayer->Units; j++)
   { Out = Input[j];
   Weight = Net->KohonenLayer->Weight[i][j];
   Lambda = Neighborhood(Net, i);
   Net->KohonenLayer->Weight[i][j] += Net->Alpha * Lambda * (Out - Weight);
   }
   StepSize = Net->KohonenLayer->StepSize[i];
   Net->KohonenLayer->StepSize[i] += Net->Alpha__ * Lambda * -StepSize;
   } }
```

```
void TrainOutput(NET* Net, REAL* Output)
 { INT i,j;
 REAL Out, Weight, Lambda;
 for (i=0; i<Net->OutputLayer->Units; i++)
 { for (j=0;
 j<Net->KohonenLayer->Units; j++)
 { Out = Output[i];
 Weight = Net->OutputLayer->Weight[i][j];
 Lambda = Neighborhood(Net, j);
 Net->OutputLayer->Weight[i][j] += Net->Alpha_ * Lambda * (Out - Weight);
 } } }


  void TrainUnits(NET* Net, REAL* Input, REAL* Output)
  { TrainKohonen(Net, Input);
    TrainOutput(Net, Output);
    }
```

Application of SOM in business

- Industrial process control
  - quality control and classification
  - process tracking and analysis
- Customer data analysis
  - segmentation of current customers
  - tailored products
  - targeted information distribution
  - better customer profitability
  - profiling of lost customers
- Identifying new customer groups
  - groups, that resemble your best customers
  - profitable groups, that are not currently being served
  - profitable groups, that have not been identified without eSom
- Business data analysis
  - grouping companies by financial key figures
  - identifying profitable and non-profitable companies
  - discovering possible growth opportunities
  - predicting bad credit and bankruptcies

Other applications

- Classification of documents for searching of its by content
- Compression of data
- Visualization of data in decision support systems
- Camera-robot coordination in robot-manipulator with vision