Neural Networks with Unsupervised Learning

- ❑ Particularities of unsupervised learning
- ❑ Data clustering with neural networks (ART networks)
- ❑ Vectorial quantization
- ❑ Topological mapping (self-organizing maps or Kohonen networks)
- ❑ Principal components analysis

Particularities of unsupervised learning

Supervised learning

- The training set contains both inputs and correct answers
- Example: classification in predefined classes for which examples of labeled data are known
- It is similar with the optimization of an error function which measures the difference between the true answers and the answers given by the network

Unsupervised learning:

- The training set contains just input data
- Example: grouping data in categories based on the similarities between them
- Relies on the statistical properties of data when tries to extract models from data
- Does not use an error concept but a model quality concept which should be maximized

Data clustering

Data clustering = identifying natural groups (clusters) in the data set such that

- Data in the same cluster are highly similar
- Data belonging to different clusters are dissimilar enough

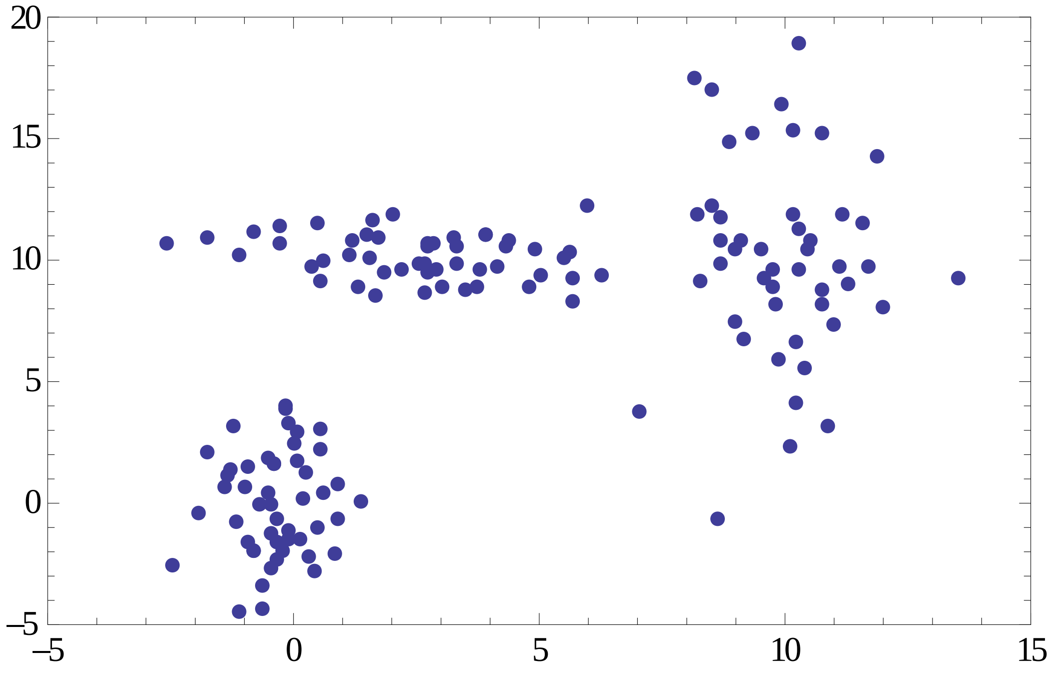
Rmk: It does not exist an apriori labeling of the data (unlike supervised classification) and sometimes even the number of clusters is unknown

Applications:

- Identification of user profiles in the case of e-commerce systems, e-learning systems etc.
- Identification of homogeneous regions in digital images (image segmentation)
- Electronic document categorization
- Biological data analysis (analysis of microarray data)

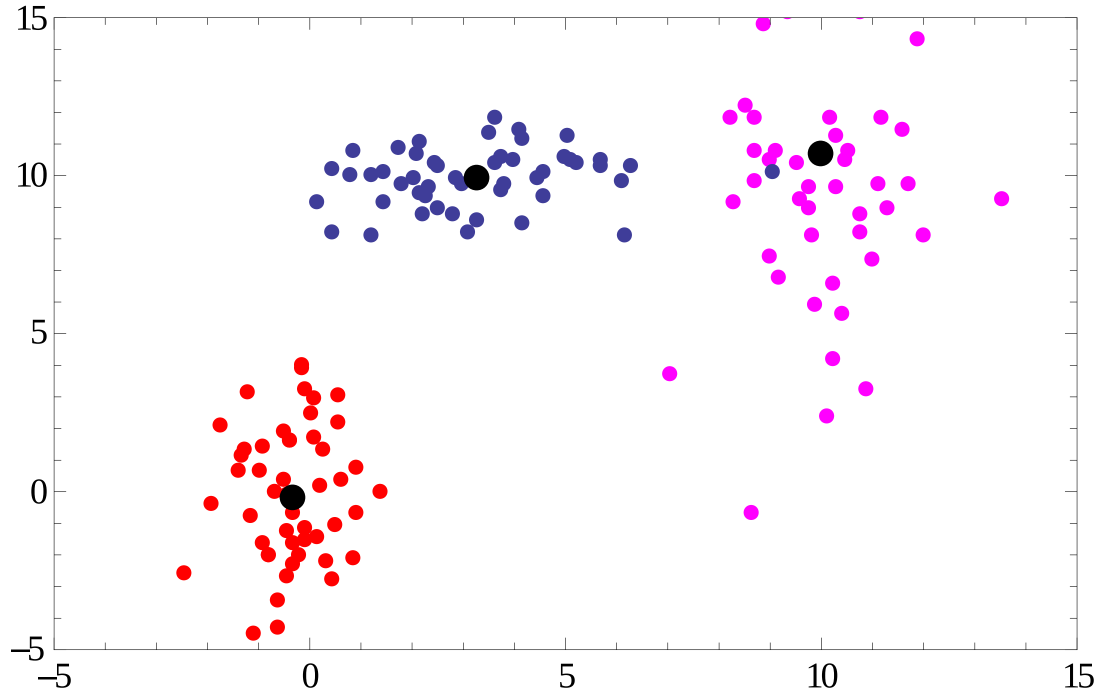
Data clustering

- Example: synthetic bidimensional data (three sources of random data based on the normal distribution)
- The real problems are usually characterized by high-dimensional data



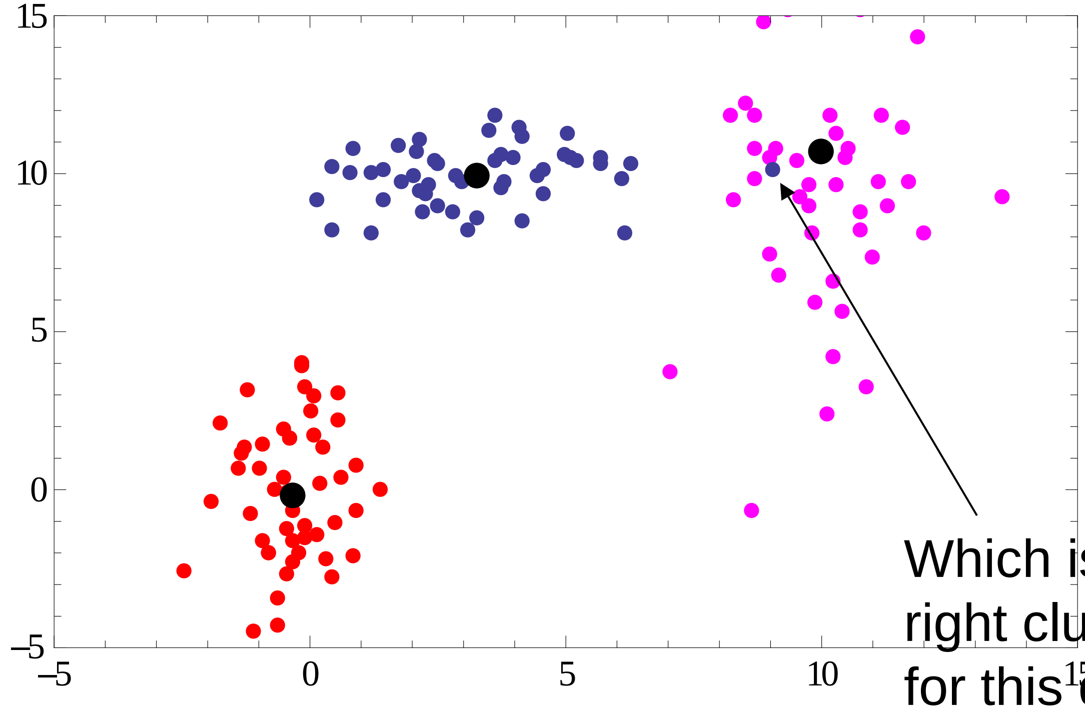
Data clustering

- Example: the real distribution of data according to their generation
- Outliers may exist
- Identifying the “right” cluster is not easy
- The clusters can be represented by their means



Data clustering

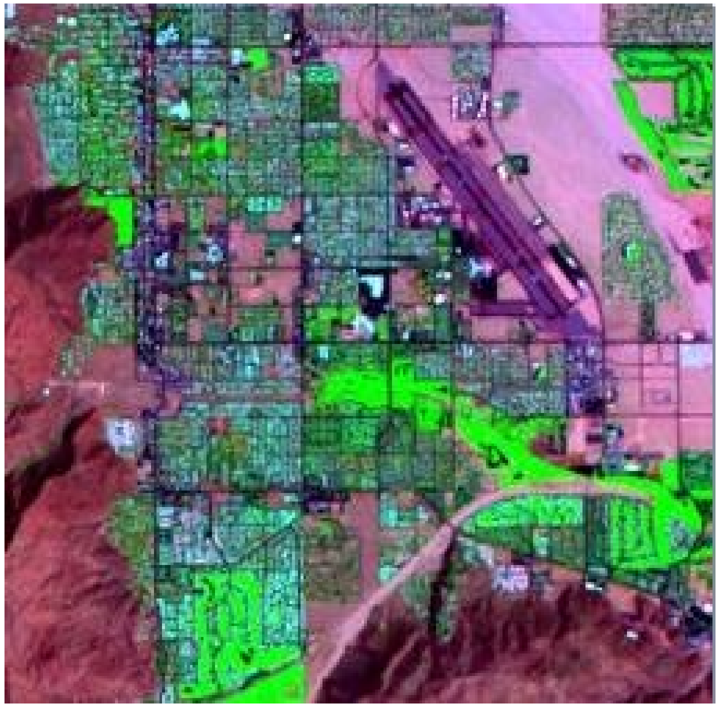
- Example: the real distribution of data according to their generation
- Outliers may exist
- Identifying the “right” cluster is not easy
- The clusters can be represented by their means

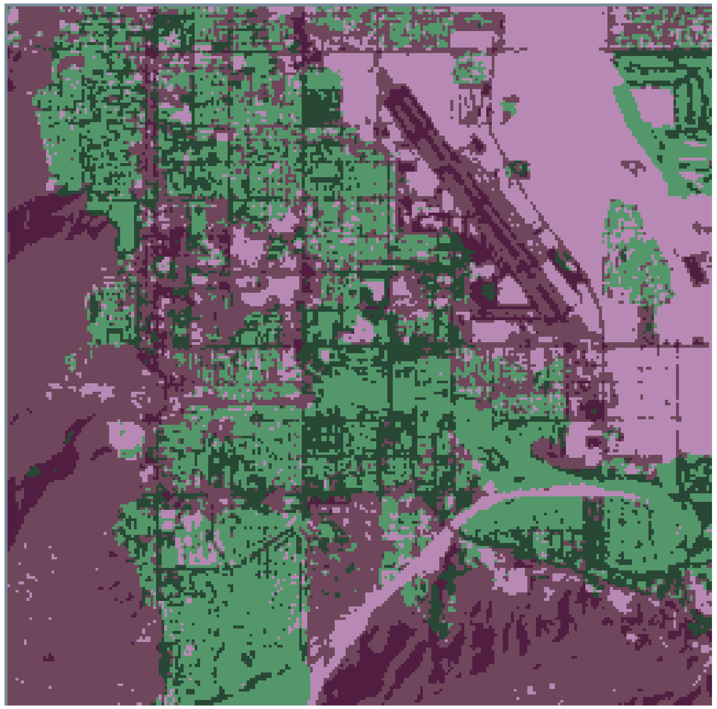


Data clustering

Example: image segmentation = identification the homogeneous regions in the image = reduction of the number of labels (colours) in the image in order to help the image analysis

Rmk: satellite image obtained by combining three spectral bands (false color)





Data clustering

A key element in data clustering is the similarity/dissimilarity measure

The choice of an appropriate similarity/dissimilarity measure is influenced by the nature of attributes

- Numerical attributes
- Categorical attributes
- Mixed attributes

Remark: there are several methods to transform categorical attributes in numerical ones, e.g. binarization:

Categorical attribute	Binary coding
Low	1 0 0
Medium	0 1 0
High	0 0 1

Data clustering

- Measures which are appropriate for numerical data

Similarity measure (cos):

$$S(X,Y) = \frac{X^T Y}{\|X\|\|Y\|}$$

Dissimilarity measure (Euclidean distance):

$$d(X,Y) = \|X-Y\|$$

For normalized vectors:

$$d^2(X,Y) = 2(1 - S(X,Y))$$

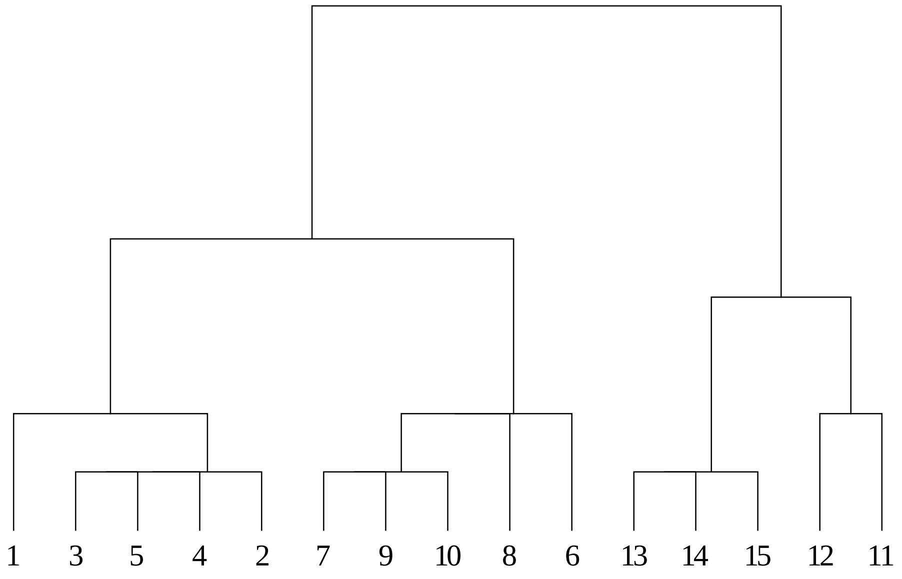
Data clustering

Clustering methods:

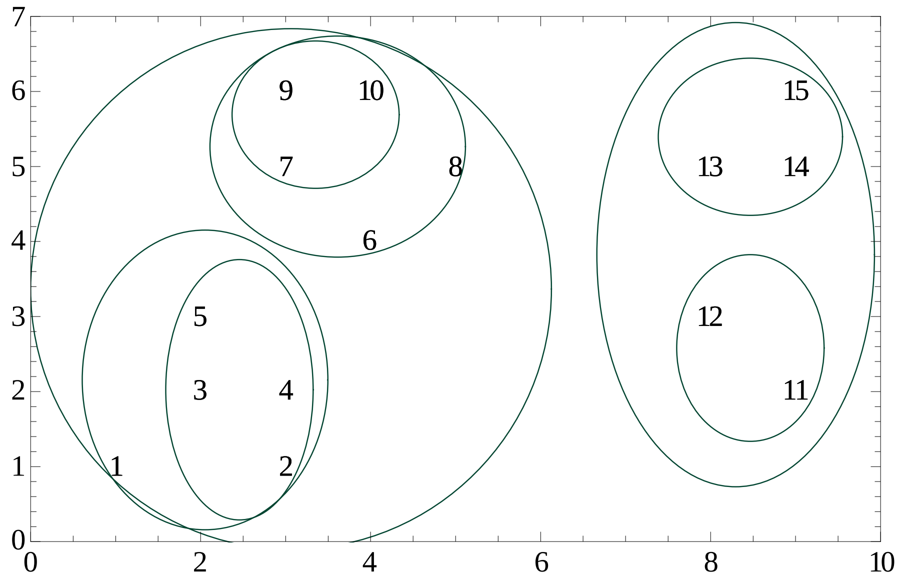
- Partitional methods:
  - Lead to a data partitioning in disjoint or partially overlapping clusters
  - Each cluster is characterized by one or multiple prototypes
- Hierarchical methods:
  - Lead to a hierarchy of partitions which can be visualized as a tree-like structure called dendrogram.

Data clustering

Clustering methods:



Hierarchical (dendrogram)



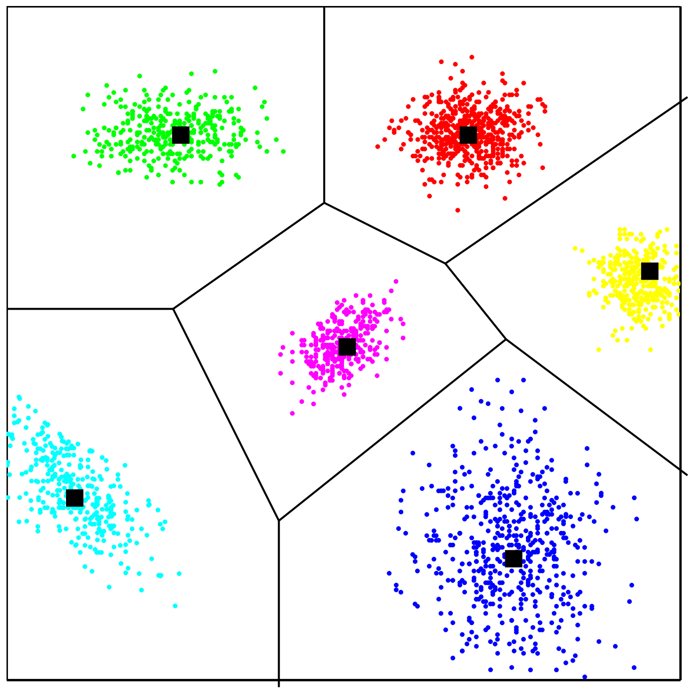
Partitions (different section levels in the dendrogram)

Data clustering

The prototypes can be:

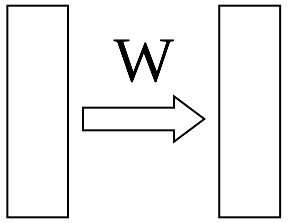
- The average of data in the class
- The median of data in the class (more robust to outliers)

The data are assigned to clusters based on the nearest prototype (the nearest neighbor principle is frequently applied for classification tasks)



Neural networks for clustering

Problem: unsupervised classification of N-dimensional data in K clusters



Architecture:

- One input layer with N units
- One linear layer with K output units
- Full connectivity between the input and the output layers (the weights matrix, W, contains on row k the prototype corresponding to class k)

Functioning:

For an input data X compute the distances between X and all prototypes and find the closest one. This corresponds to the cluster to which X belongs.

The unit having the closest weight vector to X is called winning unit

Neural networks for clustering

Training:

Training set: $\{X^1, X^2, \ldots, X^L\}$

Algorithms:

- the number of clusters (output units) is known
  - "Winner Takes All" algorithm (WTA)

- the number of clusters (output units) is unknown
  - "Adaptive Resonance Theory" (ART algorithms)

Particularities of these algorithms:

- Only the weights corresponding to the winning unit are adjusted
- The learning rate is decreasing

Neural networks for clustering

Examplu: WTA algorithm

```
Initialization
   $W^k$ := randomly selected element from the training set
   $t := 1$
Prototypes adjustment
  REPEAT
   FOR l := 1, L  DO
      find $k^*$ such that $d(W^{k^*}, X^l) \le d(W^k, X^l)$ for all $k = 1..K$
      $W^{k^*} := W^{k^*} + \eta(t)(X^l - W^{k^*})$
   ENDFOR
   $t := t + 1$
   UNTIL  $\eta(t) < \varepsilon$
```

Neural networks for clustering

Remarks:

- Decreasing learning rate (it decreases to 0 but not too fast)
- Corresponding mathematical properties:

$$\lim_{t\to\infty} \eta(t) = 0, \quad \sum_{t=1}^{\infty} \eta(t) = \infty, \quad \sum_{t=1}^{\infty} \eta^2(t) < \infty$$

Examples:

$$\eta(t) = \eta(0)/t^{\alpha}, \alpha \in (0.5, 1]$$

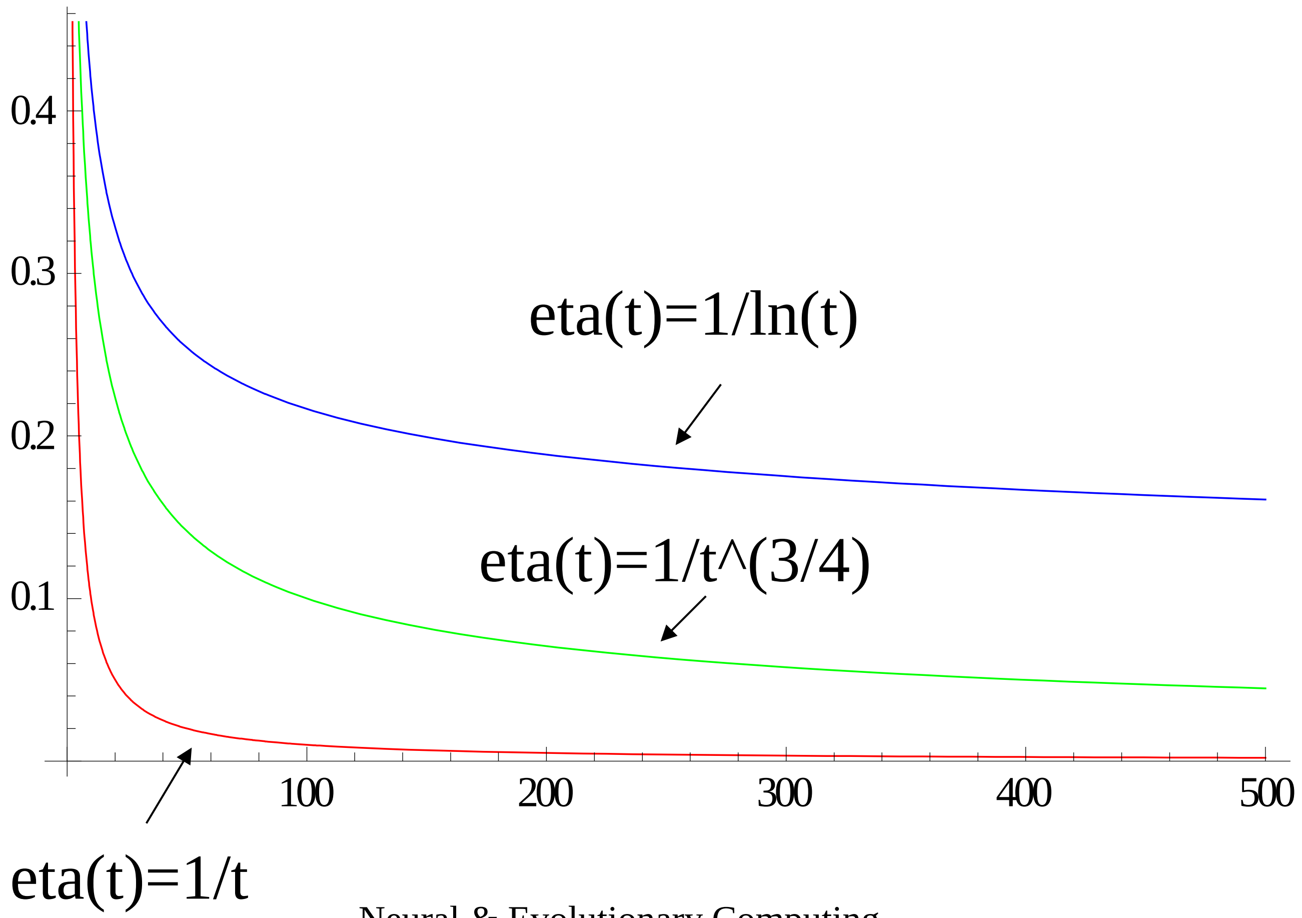
$$\eta(t) = \eta(0)/\ln(t+1) \ \text{ (it doesn't satisfy the second condition}$$

$$\text{but it is used in practice)}$$

Neural networks for clustering

Remarks:

- Decreasing learning rate



Neural networks for clustering

Remarks:

- “Dead” units: units which are never winners

Cause: inappropriate initialization of prototypes

Solutions:
- Using vectors from the training set as initial prototypes

- Adjusting not only the winning units but also the other units (using a much smaller learning rate)

- Penalizing the units which frequently become winners

Neural networks for clustering

- Penalizing the units which frequently become winners: change the criterion to establish if a unit is a winner or not

$$d(X, W^{k^*}) - \theta_{k*} \leq d(X, W^{k}) - \theta_k$$

θ_k – threshold which decreases as the number of situations when the unit k is a winner increases

Neural networks for clustering

It is useful to normalize both the input data and the weights (prototypes):

- The normalization of data from the training set is realized before the training
- The weights are normalized during the training:

$$W^{k^*}(t+1) = \frac{W^{k^*}(t) + \eta(t)(X - W^{k^*}(t))}{\left\| W^{k^*}(t) + \eta(t)(X - W^{k^*}(t)) \right\|}$$

Neural networks for clustering

Adaptive Resonance Theory: gives solutions to the following problems arising in the design of unsupervised classification systems:

- Adaptability (plasticity)
  - Refers to the capacity of the system to assimilate new data and to identify new clusters (this usually means a variable number of clusters)
- Stability
  - Refers to the capacity of the system to conserve the clusters' structures such that during the adaptation process the system does not radically change its output for a given input data

Neural networks for clustering

Example: ART2 algorithm

```
Initialization
  Choose the initial number of prototypes (K < L)
  $W^k :=$ randomly selected element from the training set
  $t := 1$
Prototypes adjustment
  REPEAT
  FOR l := 1, L DO
     find $k^*$ such that $d(W^{k^*}, X^l) \leq d(W^k, X^l)$ for all $k = 1..K$
     IF $d(W^{k^*}, X^l) < \rho$ OR $K > K_{max}$
        THEN $W^{k^*} := (X^l + W^{k^*}\mathrm{card}(\omega_{k^*}))/(1+\mathrm{card}(\omega_{k^*}))$
        ELSE $K := K+1;\ W^K := X^l$
     ENDIF
  ENDFOR
  $t := t+1$
  UNTIL $t > t_{max}$
```

ω_{k^*} = cluster corresponding to prototype W^{k^*}

Neural networks for clustering

Remarks:

- The value of ρ influences the number of output units (clusters)
  - A small value of ρ leads to a large number of clusters
  - A large value of ρ leads to a small number of clusters
- Main drawback: the presentation order of the training data influences the training process
- The main difference between this algorithm and that used to find the centers of a RBF network is represented just by the adjustment equations
- There are also versions for binary data (alg ART1)

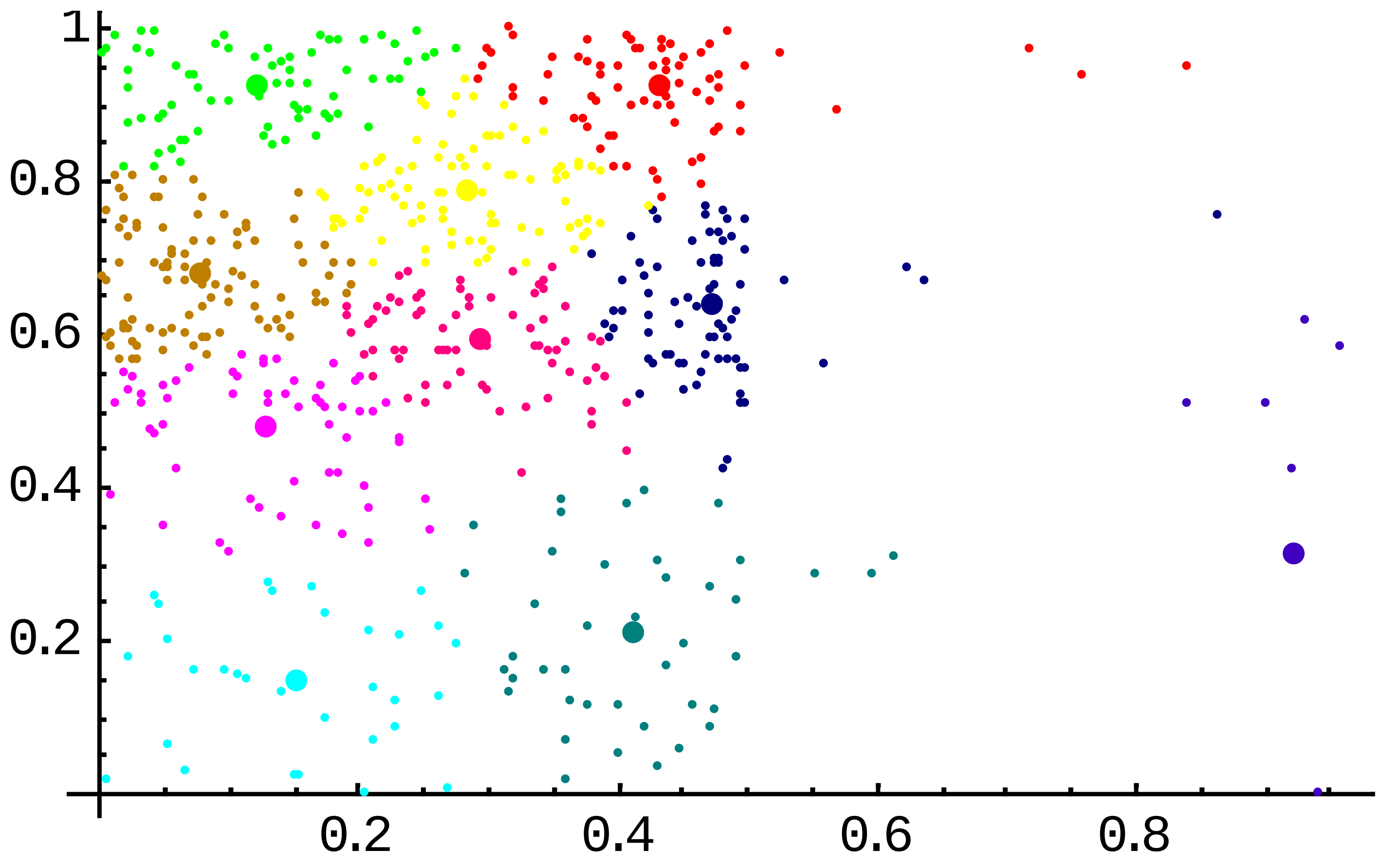
Vectorial quantization

Aim of vectorial quantization:

- Mapping a region of R^N to a finite set of prototypes
- Allows the partitioning of a N-dimensional region in a finite number of subregions such that each subregion is identified by a prototype
- The cuantization process allows to replace a N-dimensional vector with the index of the region which contains it leading to a very simple compression method, but with loss of information. The aim is to minimize this loss of information
- The number of prototypes should be large in the dense regions and small in less dense regions

Vectorial quantization

Example



Vectorial quantization

- If the number of regions is predefined then one can use the WTA algorithm

- There is also a supervised variant of vectorial quantization (LVQ - Learning Vector Quantization)

  Training set: $\{(X^1,d_1),\ldots,(X^L,d_L)\}$

LVQ algorithm:

1. Initialize the prototypes by applying a WTA algorithm to the set $\{X^1,\ldots,X^L\}$
2. Identify the clusters based on the nearest neighbour criterion
3. Establish the label for each class by using the labels from the training set: for each class is assigned the most frequent label from $d_1,\ldots d_L$

Vectorial quantization

- Iteratively adjust the prototypes by applying an algorithm similar to that of perceptron (one layer neural networks). At each iteration one applies

```
FOR $l := 1, L$ DO
    find k* such that: $d(X^l, W^{k^*}) \le d(X^l, W^k)$
    IF $c(k^*) = d_l$ THEN $W^{k^*} := W^{k^*} + \eta(t)(X^l - W^{k^*})$
    ELSE $W^{k^*} := W^{k^*} - \eta(t)(X^l - W^{k^*})$
    ENDIF
ENDFOR
```

Topological mapping

- It is a variant of vector quantization which ensures the conservation of the neighborhood relations between input data
  - Similar input data will either belong to the same class or to "neighbor" classes.
  - In order to ensure this we need to define an order relationship between prototypes and between the network's output units.
  - The architecture of the networks which realize topological mapping is characterized by the existence of a geometrical structure of the output level; this correspond to a one, two or three-dimensional grid.
  - The networks with such an architecture are called Kohonen networks or self-organizing maps (SOMs)

Self-organizing maps (SOMs)

They were designed in the beginning in order to model the so-called cortical maps (regions on the brain surface which are sensitive to some inputs):

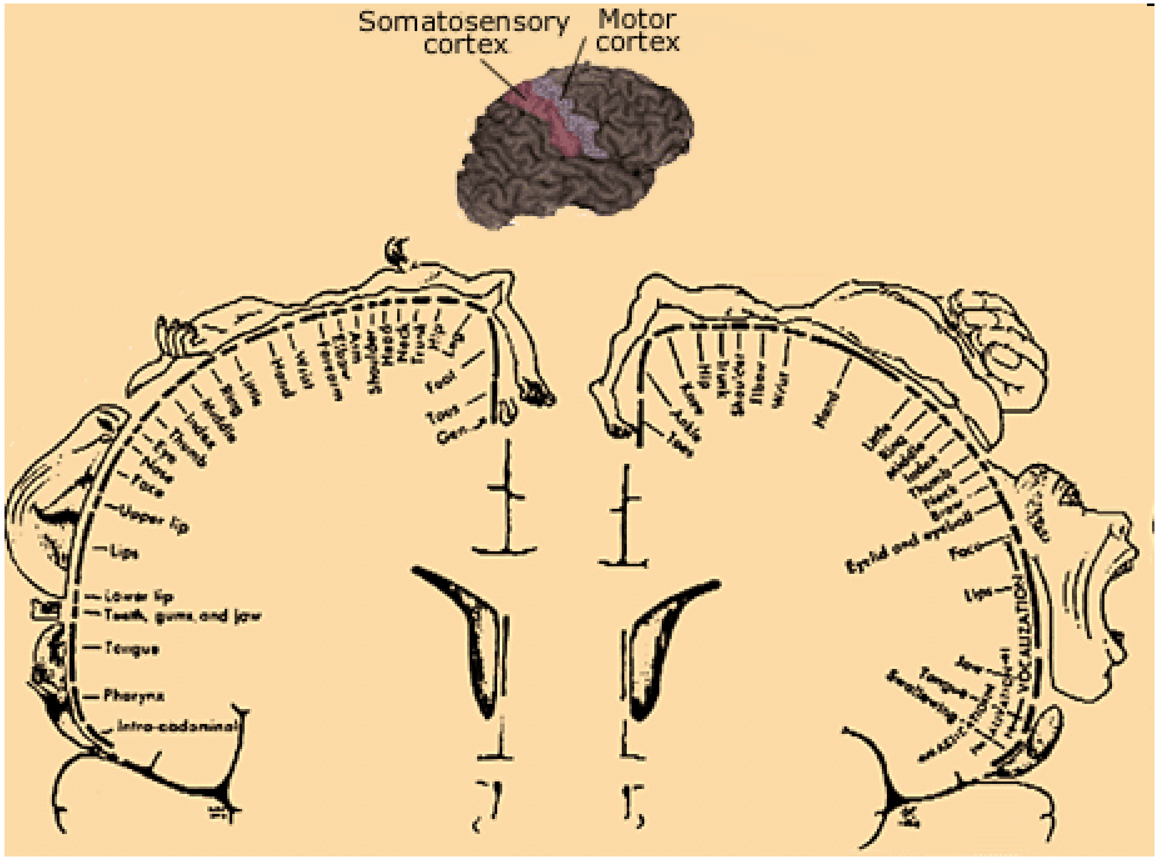
- Topographical maps (visual system)
- Tonotopic maps (auditory system)
- Sensorial maps (associated with the skin surface and its receptors)

Self-organizing maps (SOMs)

- Sensorial map (Wilder Penfield)

Left part: somatosensory cortex
- receives sensations
- sensitive areas, e.g. fingers, mouth, take up most space of the map

Right part: motor cortex
- controls the movements



Self-organizing maps (SOMs)

Applications of SOMs:

- low dimensional views of high-dimensional data
- data clustering

Specific applications (http://www.cis.hut.fi/research/som-research/)

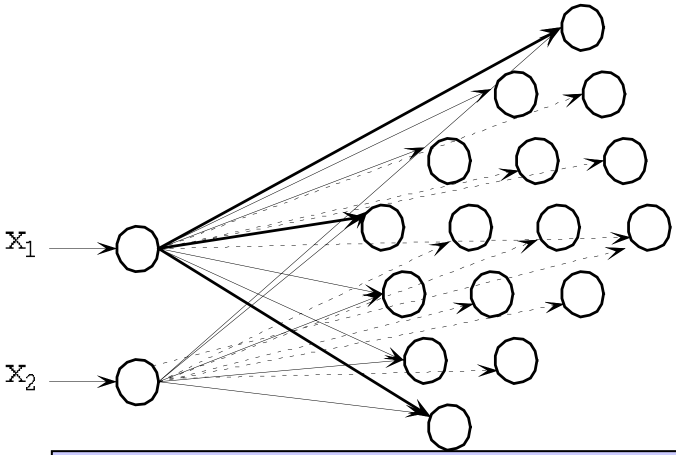
- Automatic speech recognition
- Clinical voice analysis
- Monitoring of the condition of industrial plants and processes
- Cloud classification from satellite images
- Analysis of electrical signals from the brain
- Organization of and retrieval from large document collections (WebSOM)
- Analysis and visualization of large collections of statistical data (macroeconomic date)

Kohonen networks

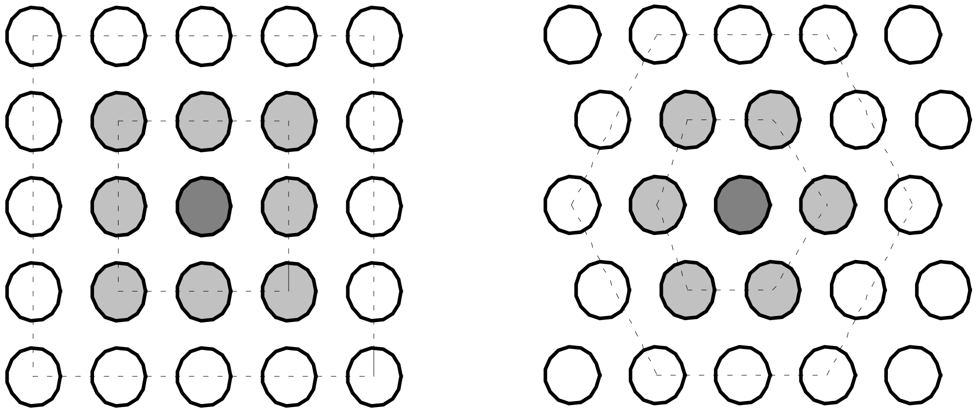
Architecture:

- One input layer

- One layer of output units placed on a grid (this allows defining distances between units and defining neighboring units)



Input Output



Rectangular Hexagonal

- Grids:

Wrt the size:

- One-dimensional
- Two-dimensional
- Three-dimensional

Wrt the structure:

- Rectangular
- Hexagonal
- Arbitrary (planar graph)

Kohonen networks

- Defining neighbors for the output units
  - Each functional unit (p) has a position vector (r_p)
  - For n-dimensional grids the position vector will have n components
  - Choose a distance on the space of position vectors

$$d_1(r_p, r_q) = \sqrt{\sum_{i=1}^{n} (r_p^i - r_q^i)^2} \quad \text{(Euclidean distance)}$$

$$d_2(r_p, r_q) = \sum_{i=1}^{n} | r_p^i - r_q^i | \quad \text{(Manhattan distance)}$$

$$d_3(r_p, r_q) = \max_{i=\overline{1,n}} | r_p^i - r_q^i |$$

Kohonen networks

- A neighborhood of order (radius) s of the unit p::

$$V_s(p) = \{q \mid d(r_p, r_q) \le s\}$$

- Example: for a two dimensional grid the first order neighborhoods of p having r_p=(i,j) are (for different types of distances):

$$V_1^{(1)}(i,j) = V_1^{(2)}(i,j) = \{(i-1,j),(i+1,j),(i,j-1),(i,j+1)\}$$

$$V_1^{(3)}(i,j) = \{(i-1,j),(i+1,j),(i,j-1),(i,j+1),(i-1,j-1),$$
$$(i-1,j+1),(i+1,j-1),(i+1,j+1)\}$$

Kohonen networks

- Functioning:
  - For an input vector, X, we find the winning unit based on the nearest neighbor criterion (the unit having the weights vector closest to X)
  - The result can be the position vector of the winning unit or the corresponding weights vector (the prototype associated to the input data)
- Learning:
  - Unsupervised
  - Training set: $\{X^1,\ldots,X^L\}$
  - Particularities: similar with WTA learning but besides the weights of the winning unit also the weights of some neighboring units are adjusted.

Kohonen networks

- Learning algorithm

```
Initialize $W, t, \eta(t), s(t)$
Repeat
  For $l := 1, L$ do
    find $p^*$ such that $d(X^l, W^{p^*}) \leq d(X^l, W^p)$, for all $p$
    $W^p := W^p + \eta(t)(X^l - W^p)$, for all $p \in N_{s(t)}(p^*)$
  Endfor
  $t := t + 1$
  compute $s(t), \eta(t)$
Until $t > t_{\max}$
```

Kohonen networks

- Learning algorithm
  - By adjusting the units in the neighbourhood of the winning one we ensure the preservation of the topological relation between data (similar data will correspond to neighboring units)
  - Both the learning rate and the neighborhood size are decreasing in time
  - The decreasing rule for the learning rate is similar to that from WTA (e.g. eta(t+1)=0.99*eta(t); eta(0)=1)
  - The initial size of the neighbor should be large enough (in the first learning steps all weights should be adjusted). Example: s(0)=m/2 (m is the number of units for a 1D network or the size of the grid for a 2D network), s(t+T)=s(t)/2 (s is halved at each T iterations)

Kohonen networks

- There are two main stages in the learning process
  - Ordering stage: it corresponds to the first iterations when the neighbourhood size is large enough; its role is to ensure the ordering of the weights such that similar input data are in correspondence with neighboring units.
  - Refining stage: it corresponds to the last iterations, when the neighborhood size is small (even just one unit – this is similar to a WTA algorithm); its role is to refine the weights such that the weight vectors are representative prototypes for the input data.

Rmk: in order to differently adjust the winning unit and the units in the neighbourhood one can use the concept of neighborhood function.

Kohonen networks

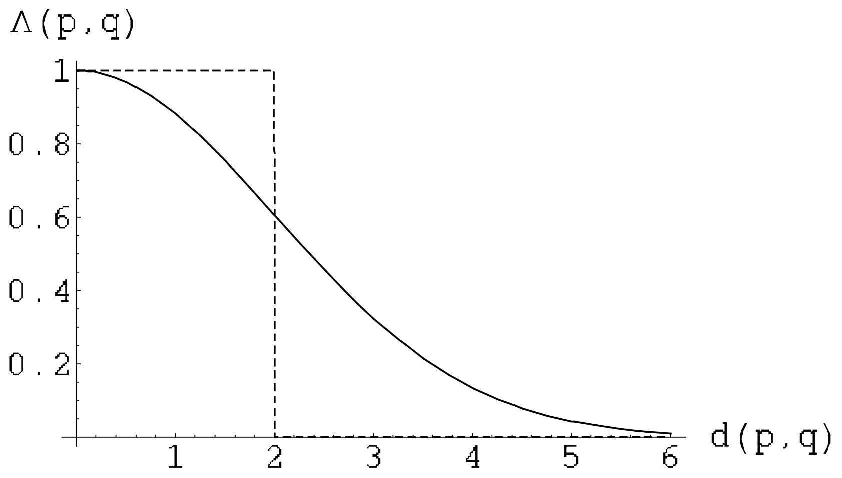
- Using a neighborhood function:

$$W^p = W^p + \eta(t)\Lambda(p^*, p)(X - W^p).$$

- Examples:

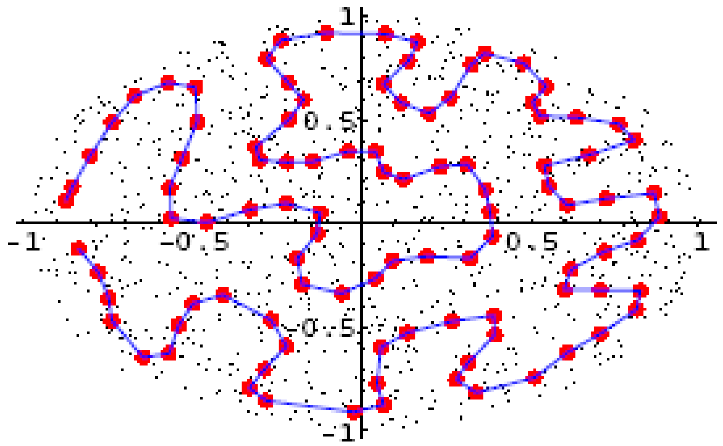
$$\Lambda(p,q) = \begin{cases} 1 & \text{if} \quad q \in V_s(p) \\ 0 & \text{otherwise} \end{cases}$$

$$\Lambda(p,q) = \exp\left(-\frac{d^2(r_p, r_q)}{2s^2}\right)$$

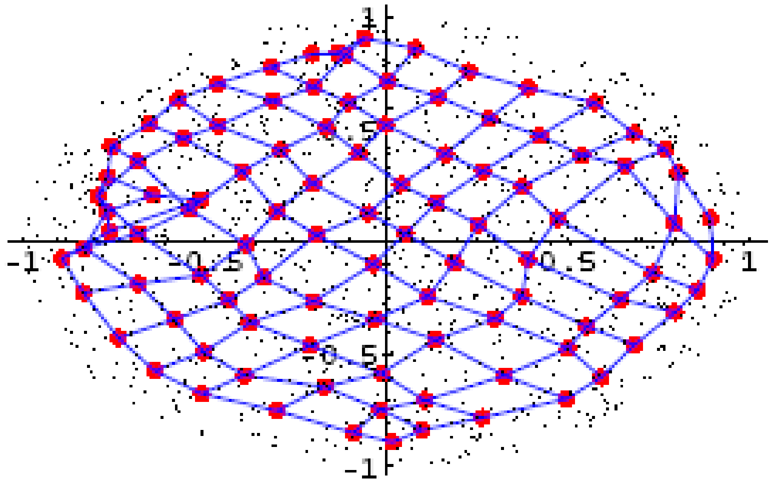


Kohonen networks

- Illustration of topological mapping
  - visualize the points corresponding to the weights vectors attached to the units.
  - Connect the points corresponding to neighboring units (depending on the grid one point can be connected with 1,2,3,4 other points)



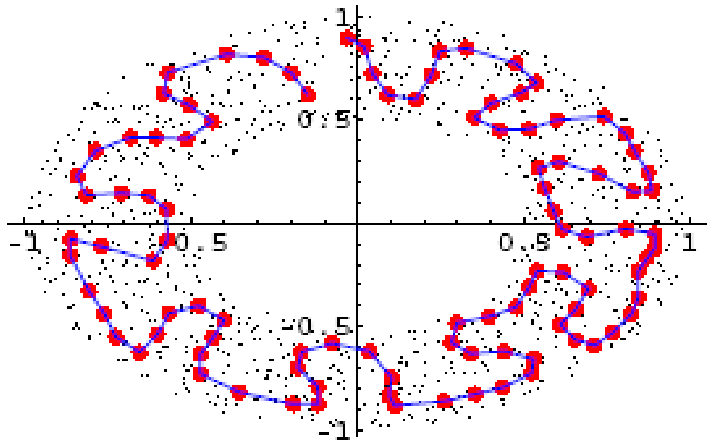
One dimensional grid

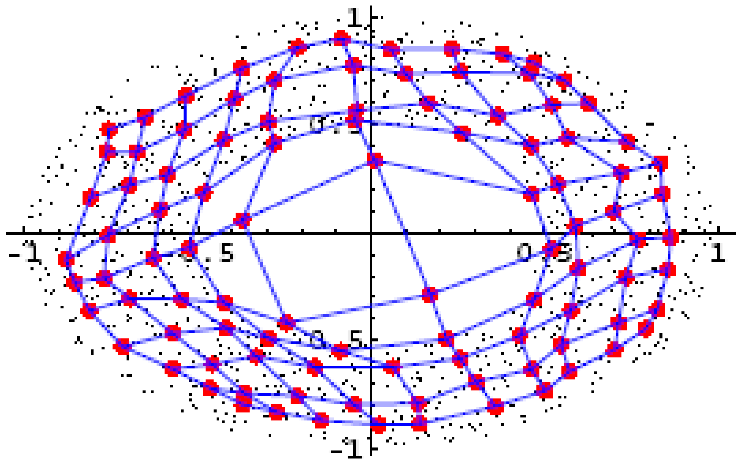


Two dimensional grid

Kohonen networks

- Illustration of topological mapping
  - Two dimensional input data randomly generated inside a circular ring
  - The functional units are concentrated in the regions where are data



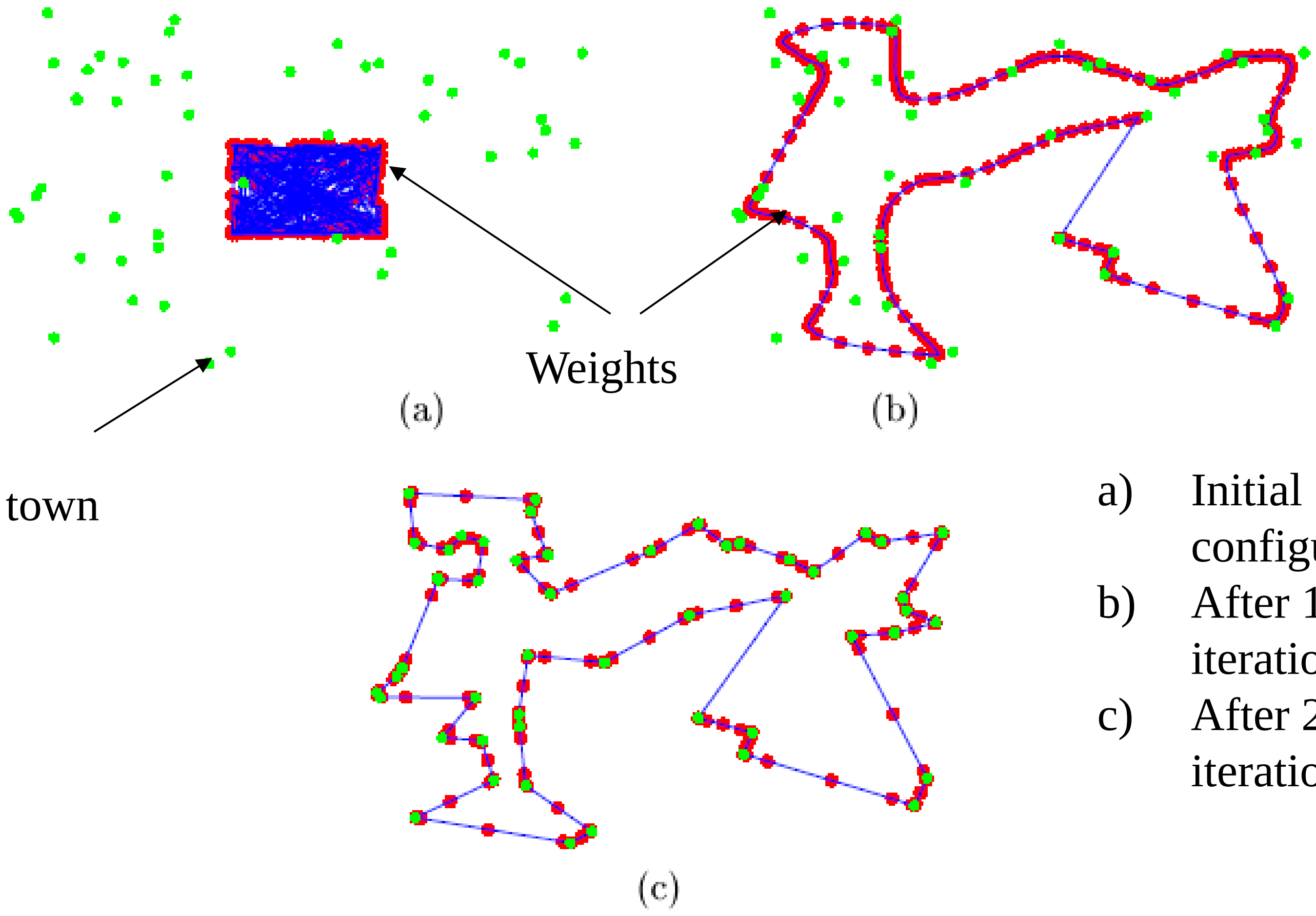


Kohonen networks

- Traveling salesman problem:
  - Find a route of minimal length which visits only once each town (the tour length is the sum of euclidean distances between the towns visited at consecutive time moments)
  - We use a network having two input units and n output units placed on a circular one-dimensional grids (unit n and unit 1 are neighbours). Such a network is called elastic net
  - The input data are the coordinates of the towns
  - During the learning process the weights of the units converges toward the positions of towns and the neighborhood relationship on the iunits set illustrates the order in which the towns should be visited.
  - Since more than one unit can approach one town the network should have more units than towns (twice or even three times)

Kohonen networks

- Traveling salesmen problem:



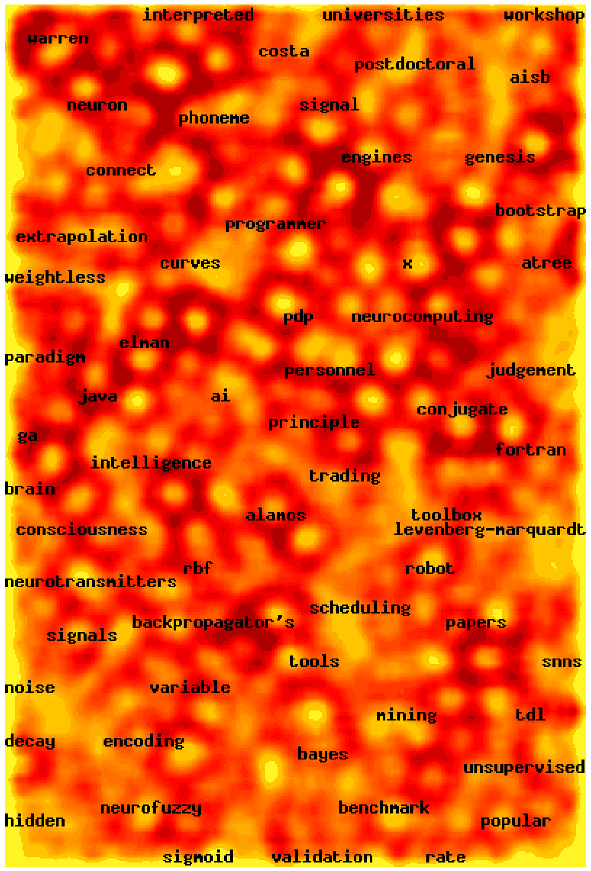
a) Initial configuration
b) After 1000 iterations
c) After 2000 iterations

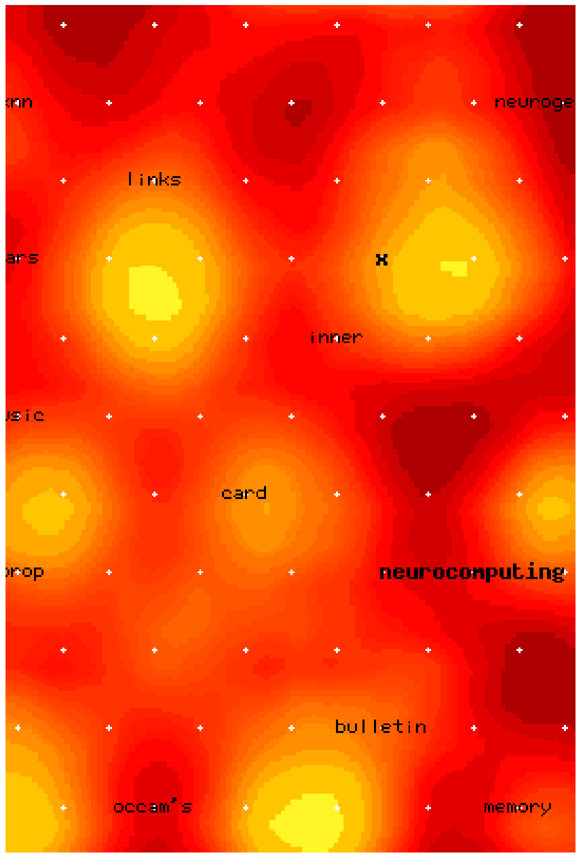
Kohonen networks

- Other applications:
  - Autonomous robots control: the robot is trained with input data which belong to the regions where there are not obstacles (thus the robot will learn the map of the region where he can move)
  - Categorization of electronic documents: WebSOM
    - WEBSOM is a method for automatically organizing collections of text documents and for preparing visual maps of them to facilitate the mining and retrieval of information.

Kohonen networks

- WebSOM (http://websom.hut.fi/websom/)



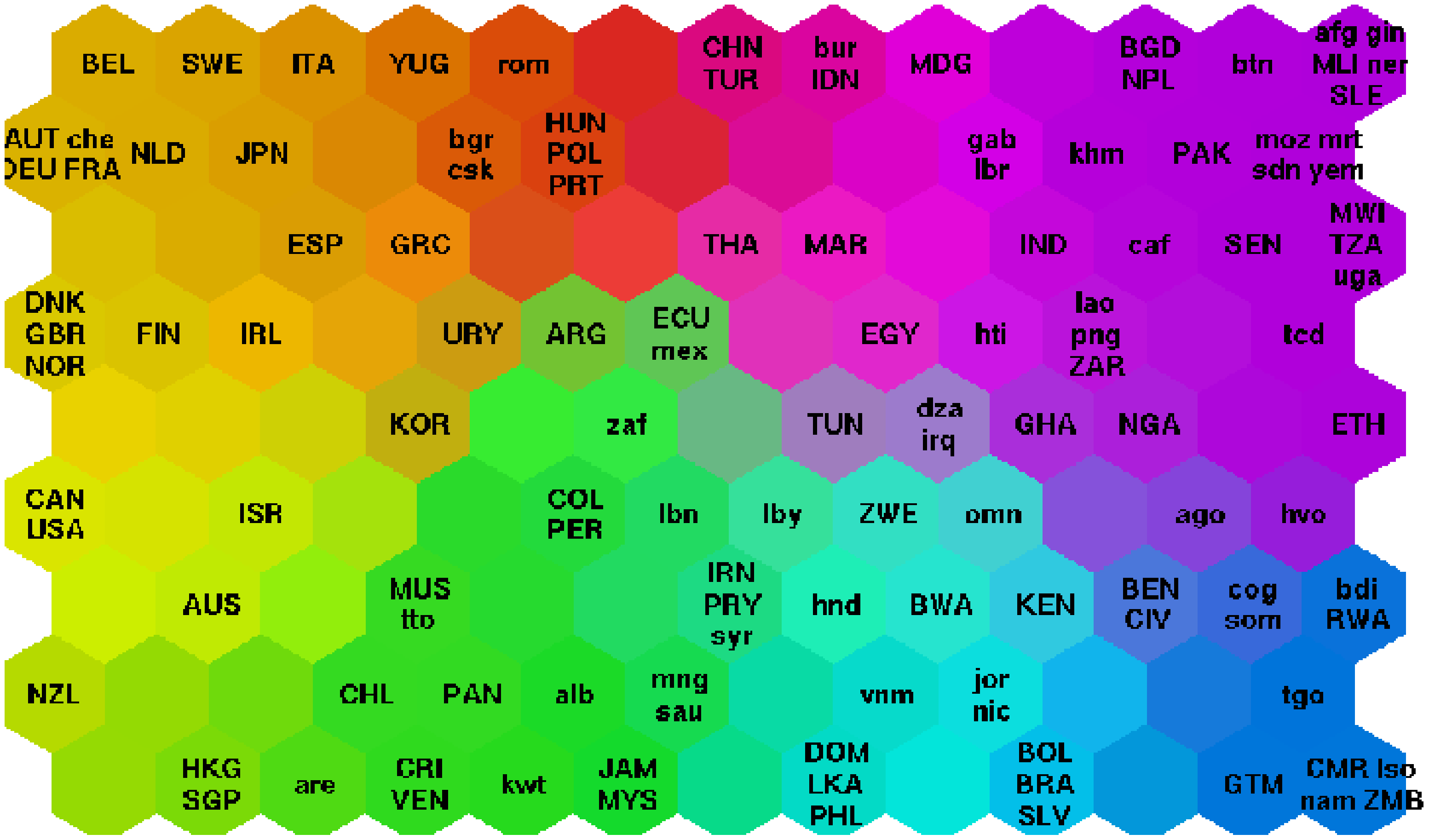


The labels represents keywords of the core vocabulary of the area in question.

The colors express the homogeneity.
Light color: high similarity, Dark color: low similarity

Kohonen networks

- World Poverty Map (http://www.cis.hut.fi/research/som-research/worldmap.html /) - based on World Bank data from 1992



Principal components analysis

- Aim:
  - reduce the dimension of the vector data by preserving as much as possible from the information they contain.
  - It is useful in data mining where the data to be processed have a large number of attributes (e.g. multispectral satellite images, gene expression data)

- Usefulness: reduce the size of data in order to prepare them for other tasks (classification, clustering); allows the elimination of irrelevant or redundant components of the data

- Principle: realize a linear transformation of the data such that their size is reduced from N to M (M<N) and Y retains the most of the variability in the original data

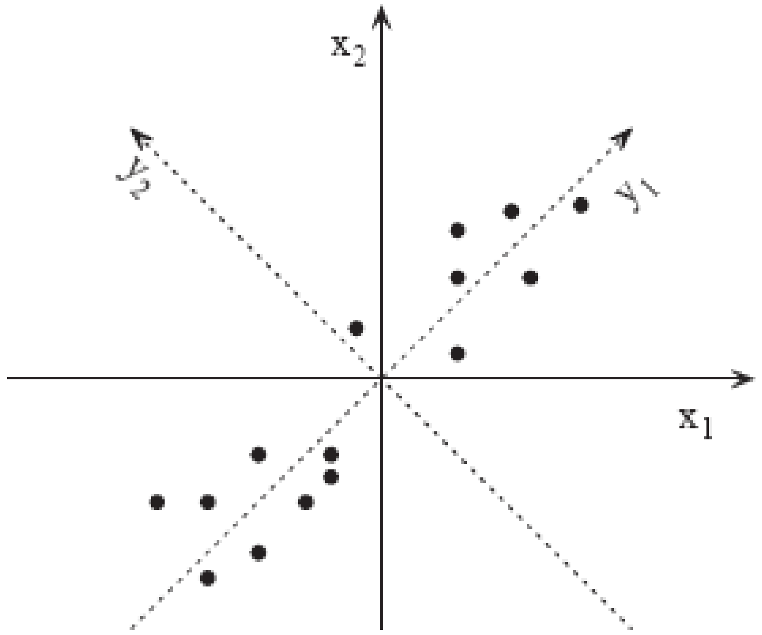
$$Y=WX$$

Principal components analysis

- Ilustration: N=2, M=1

The system of coordinates x_1Ox_2 is transformed into y_1Oy_2

Oy_1 - this is the direction corresponding to the largest variation in data; thus we can keep just component y1; it is enough to solve a further classification task



Principal components analysis

Formalization:

- Suppose that the data are sampled from a N-dimensional random vector characterized by a given distribution (usually of mean 0 – if the mean is not 0 the data can be transformed by subtracting the mean)

- We are looking for a pair of transformations

$$T:R^N \to R^M \quad \text{and} \quad S:R^M \to R^N$$

$$X \xrightarrow[T]{} Y \xrightarrow[S]{} X'$$

Which have the property that the reconstructed vector X'=S(T(X)) is as close as possible from X (the reconstruction error is small)

Principal components analysis

Formalization: the matrix W (M rows and N columns) which leads to the smallest reconstruction error contains on its rows the eigenvectors (corresponding to the largest M eigenvectors) of the covariance matrix of the input data distribution

Random vector with zero mean

$$X = (X_1,\dots,X_N),\ E(X_i) = 0$$

Covariance matrix : C(X)

$$c_{ij} = E((X_i - E(X_i))(X_j - E(X_j)) = E(X_i X_j)$$

$C(X)$ is symmetric and (semi)positive defined $\Rightarrow$

all eigenvalues are real and positive, thus they can be sorted

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{\mathrm{M}} \dots \geq \lambda_{\mathrm{N}} \geq 0$$

Principal components analysis

Constructing the transformation T (statistical method):

- Transform the data such that their mean is 0
- Construct the covariance matrix
  - Exact (when the data distribution is known)
  - Approximate (selection covariance matrix)
- Compute the eigenvalues and the eigenvectors of C
  - They can be approximated by using numerical methods
- Sort decreasingly the eigenvalues of C and select the eigenvectors corresponding to the M largest eigenvalues.

Principal components analysis

Drawbacks of the statistical method :

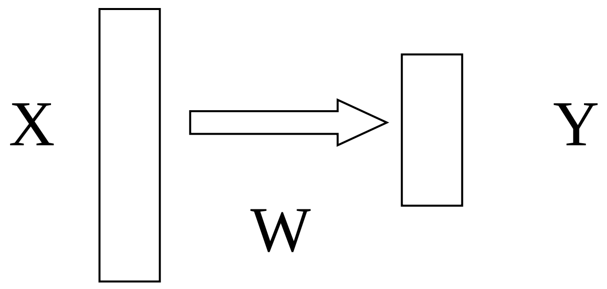
- High computational cost for large values of N
- It is not incremental
  - When a new data have to be taken into consideration the covariance matrix should be recomputed

Other variant: use a neural network with a simple architecture and an incremental learning algorithm

Neural networks for PCA

Architecture:

- N input units
- M linear output units
- Total connectivity between layers



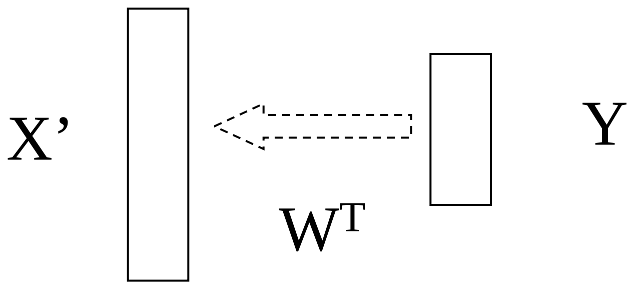
Functioning:

- Extracting the principal components

  $Y=WX$

- Reconstructing the initial data

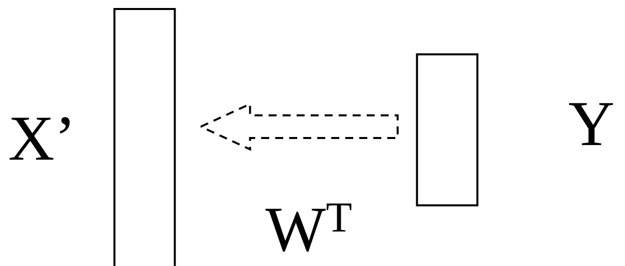
  $X'=W^TY$



Neural networks for PCA

Learning:

- Unsupervised
- Training set: {X1,X2,…} (the learning is incremental, the learning is adjusted as it receives new data)



Learning goal: reduce the reconstruction error (difference between X and X')

- It can be interpreted as a self-supervised learning

Neural networks for PCA

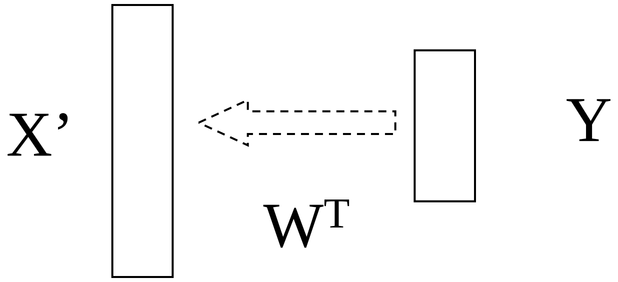
Self-supervised learning:

Training set:
{(X^1,X^1), (X^2,X^2),….}

Quadratic error for reconstruction (for one example):

$$E(W)=\frac{1}{2}\sum_{j=1}^{N}(x_j-x'_j)^2=$$

$$=\frac{1}{2}\sum_{j=1}^{N}(x_j-\sum_{i=1}^{M}w_{ij}y_i)^2$$



By applying the idea of gradient based minimization:

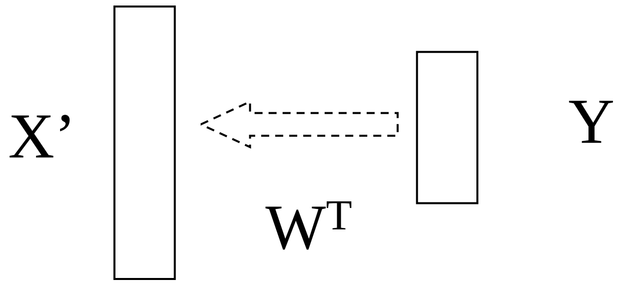
$$w_{ij}:=w_{ij}+\eta\cdot(x_j-x')y_i$$

$$w_{ij}:=w_{ij}+\eta\cdot(x_j-\sum_{k=1}^{M}w_{kj}y_k)y_i$$

Neural networks for PCA

Oja's algorithm:

Training set:
{(X^1,X^1), (X^2,X^2),….}



$$\text{Initialization} : w_{ij} := rand(-1,1);\ t := 1$$

REPEAT // for each example X

$$y_i = \sum_{j=1}^{N} w_{ij} x_j,\ i = 1..M$$

$$w_{ij} := w_{ij} + \eta \cdot (x_j - \sum_{k=1}^{M} w_{kj} y_k) y_i,\ i = 1..M,\ j = 1..N$$

$$t := t + 1$$

UNTIL $t > t_{\max}$

Rmks:
- the rows of W converges toward the eigenvectors of C which corresponds to the M largest eigenvalues
- It does not exist a direct correspondence between the unit position and the rank of the eigenvalue

Neural networks for PCA

Sanger's algorithm:

It is a variant of Oja's algorithm which ensures the fact that the row I of W converges to the eigenvector corresponding to the ith eigenvalue (in decreasing order)

```
Initialization: $w_{ij} := rand(-1,1);\ t := 1$
REPEAT // for each example X
    $y_i = \sum_{j=1}^{N} w_{ij} x_j,\ i = 1..M$
    $w_{ij} := w_{ij} + \eta \cdot (x_j - \sum_{k=1}^{i} w_{kj} y_k) y_i,\ i = 1..M, j = 1..N$
    $t := t + 1$
UNTIL $t > t_{\max}$
```

Particularity of the Sanger's algorithm