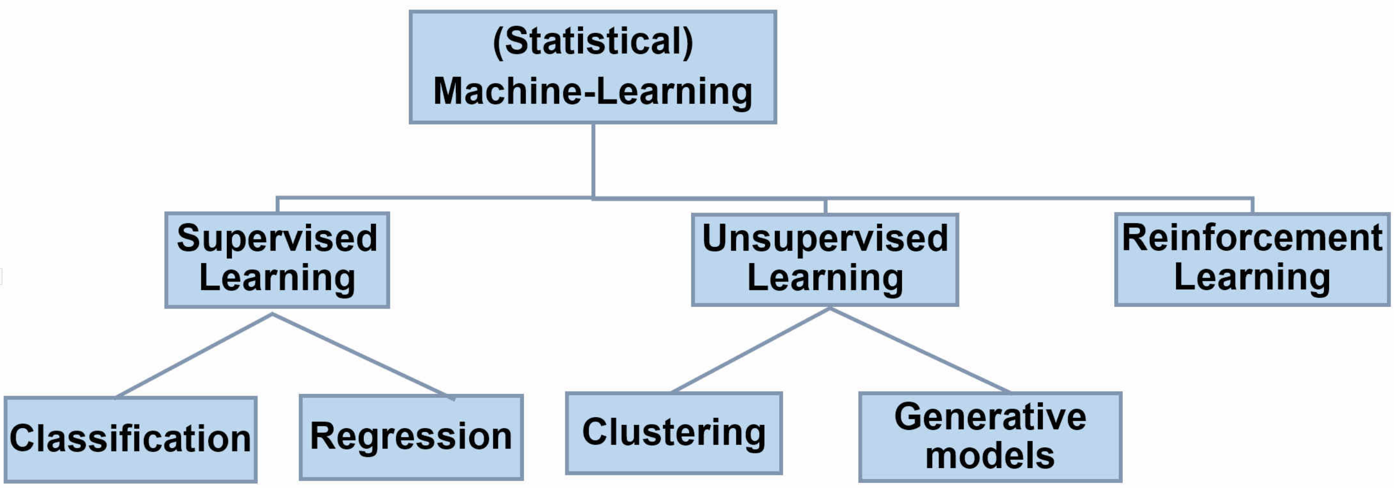
UNSUPERVISED MACHINE-LEARNING

Pr. Fabien MOUTARDE
Center for Robotics
MINES ParisTech
PSL Université Paris

Fabien.Moutarde@mines-paristech.fr

http://people.mines-paristech.fr/fabien.moutarde

Machine-Learning *TYPOLOGY*



Supervised vs UNsupervised learning

Learning is called ***"supervised"*** when <u>there are "target" values</u> for every example in training dataset:

examples = (input-output) = $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$

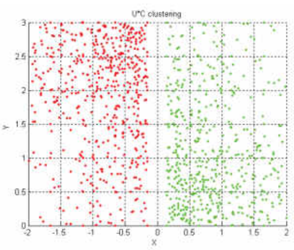
The goal is to build a (generally non-linear) approximate model for interpolation, in order to be able to GENERALIZE to input values other than those in training set

"Unsupervised" = when there are <u>NO target values</u>:

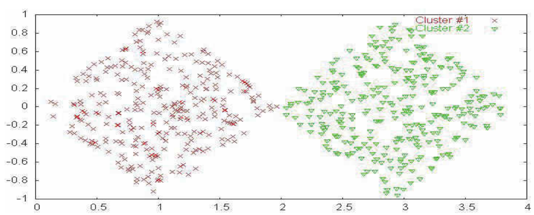
dataset = $\{x_1, x_2, \ldots, x_n\}$

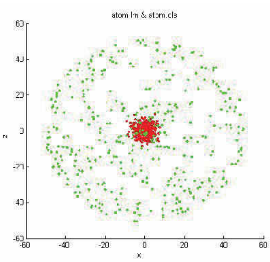
The goal is typically either to do datamining (unveil structure in the distribution of examples in input space), or to find an output maximizing a given evaluation function

Examples of UNSUPERVISED Machine-Learning



Datamining (clustering)





Generative Learning

Generated fake faces

UNSUPERVISED learning *from data*

Set of "input-only" (i.e. unlabeled) examples : $X=\{x_1, x_2, \dots, x_n\}$ ($x_i \in \Re^d$, often with « large d»)

H family of mathematical models [each $h \in H \rightarrow y=h(x)$]

Hyper-parameters for training algorithm

LEARNING ALGORITHM

$h \in H$ so that criterion $J(h,X)$ is verified or optimised

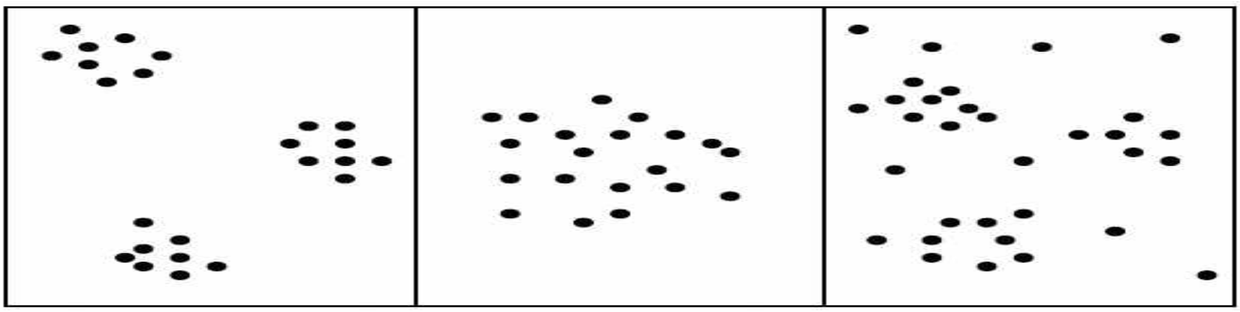
Typical example: "clustering"

- $h(x) \in C=\{1,2, \dots, K\}$ [each i ←→ a "cluster"]
- $J(h,X)$: $dist(x_i,x_j)$ smaller for x_i,x_j with $h(x_i)=h(x_j)$ than for x_i,x_j with $h(x_i) \neq h(x_j)$

Clustering (en français, regroupement ou partitionnement)

Goal = identify structure in data distribution

- Group together examples that are close/similar
- Pb: groups not always well-defined/delimited, can have arbitrary shape, and fuzzy borders



Similarity and Distances

Similarity

- The larger a similarity measure, the more similar the points are
- ≈ inverse of distance

How to measure distance between 2 points $d(x_1; x_2)$?

- **Euclidian** distance:
  $d^2(x_1;x_2) = \Sigma_i (x_{1i}-x_{2i})^2 = (x_1-x_2).^t(x_1-x_2)$ *[L_2 norm]*
- ***Manhattan*** distance:
  $d(x_1;x_2) = \Sigma_i |x_{1i}-x_{2i}|$ *[L_1 norm]*
- **Sebestyen** distance:
  $d^2(x_1;x_2) = (x_1-x_2).W.^t(x_1-x_2)$ *[with W=diagonal matrix]*
- ***Mahalanobis*** distance:
  $d^2(x_1;x_2) = (x_1-x_2).C.^t(x_1-x_2)$ *[with C=Covariance matrix]*

Typology of clustering techniques

- By ***agglomeration***
  - Agglomerative Hierarchical Clustering, AHC [en français, Regroupement Hiérarchique Ascendant]
- By ***partitioning***
  - Partitionnement Hiérarchique Descendant
  - Spectral partitioning (separation in the space of vecteurs propres of adjacency matrix)
  - K-means
- By ***modelling***
  - Mixture of Gaussians (GMM)
  - Self-Organizing (Kohohen) Maps, SOM (Cartes de Kohonen)
- Based on data density

Agglomerative Hierarchical Clustering (AHC)

Principle: recursively, each point or cluster is absorbed by the nearest cluster

Algorithm

- Initialization:
  - Each example is a cluster with only one point
  - Compute the matrix M of similarities for each pair of clusters
- Repeat:
  - Selection in M of the 2 most mutually similar clusters C_i and C_j
  - Fusion of C_i and C_j in a more general cluster C_g
  - Update of M matrix, by computing similarities between C_g and all pre-existing clusters

Until fusion of the 2 last clusters

Distance between 2 clusters??

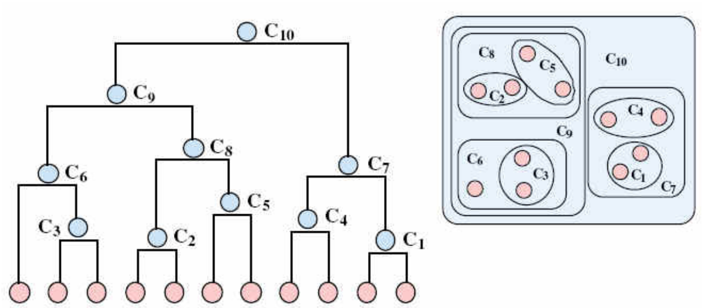
- ***Min*** distance (between closest points):
  `min(d(i,j) i∈C1 & j∈C2)` — $\min(d(i,j)\ \ i \in C_1\ \&\ j \in C_2)$
- ***Max*** distance: $\max(d(i,j)\ \ i \in C_1\ \&\ j \in C_2)$
- ***Average*** distance:
  $(\Sigma_{i \in C1 \& j \in C2}\, d(i,j)) / (card(C_1) \times card(C_2))$
- distance ***between the 2 centroïds:*** $d(b_1;b_2)$
- ***Ward*** distance:
  $sqrt(n_1 n_2 / (n_1+n_2)) \times d(b_1;b_2)$ *[où n_i=card(C_i)]*

Each type of clusters inter-distance

➔ specific variant ≠ of AHC

- distMin (ppV) → *single-linkage*
- distMax → *complete-linkage*

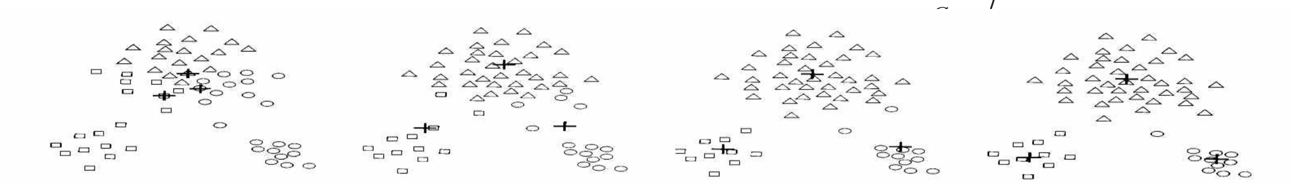
AHC output = dendrogram



- **dendrogram = representation of the full hierarchy of successively grouped clusters**
- **Height from a cluster to its sub-clusters ≈ distance between the 2 merged clusters**

Clustering by partitionning: K-means algorithm

- **Each cluster C_k defined by its « centroïd » c_k, which is a « prototype » (a vector template in input space);**
- **Each training example x is « assigned » to cluster $C_{k(x)}$ which has centroïd nearest to x :**
  `k(x)=ArgMin`$_k$`(dist(x,`c_k`))`
- **ALGO :**
  - Initialization = *__randomly__* choose K *__distinct__* points $c_1,\dots,c_K$ among training examples $\{x_1,\dots, x_n\}$
  - REPEAT until stabilization » of all c_k:
    - Assign each x_i to cluster $C_{k(i)}$ which has min $dist(x_i,c_{k(i)})$
    - Recompute centroïds c_k of clusters: $c_k = \sum x \,/\, card(C_k)$



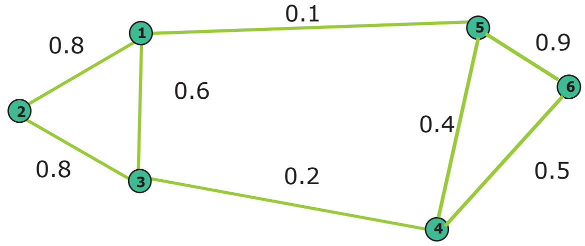
(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

[This minimizes $D = \sum_{k=1}^{K}\sum_{x\in C_k} dist(c_k,x)^2$ ***]***

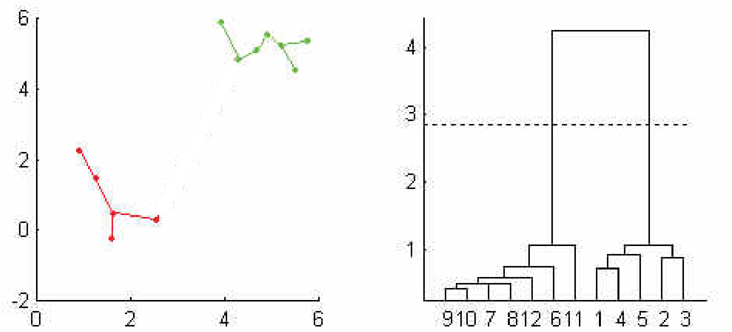
Other partitioning method: SPECTRAL clustering

- **Principle = use adjacency *graph***

Nodes = input examples
Edge values = similarities
(in [0;1], so 1 ←→ same point)



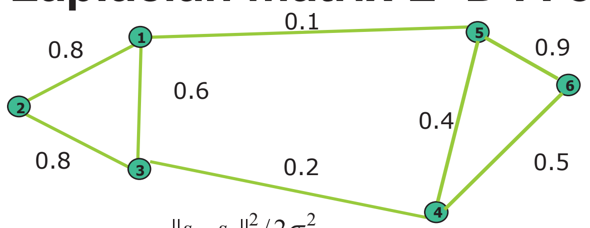
➔ Graph partitioning algos (min-cut, etc…) allow to recursively split graph in several connex componants



Ex: Minimal Spanning Tree + remove edges from smallest to bigger values
→ single-linkage clusters

Spectral clustering algo

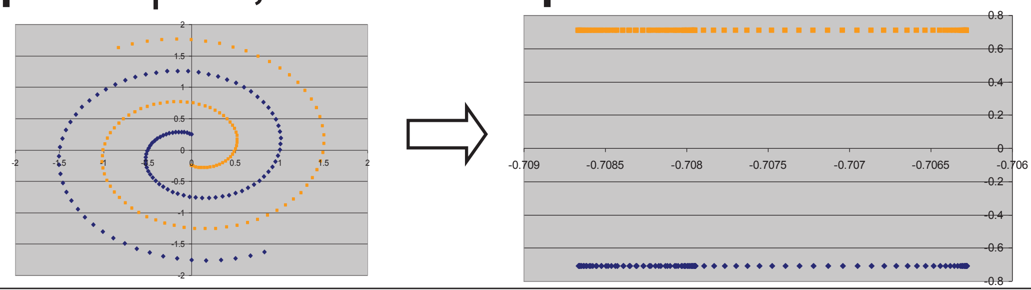
- **Compute Laplacian matrix L=D-A of the adjacency graph**



$$A_{ij} = e^{-\|s_i - s_j\|^2 / 2\sigma^2}$$

	x_1	x_2	X_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.1	-0.4	-0.5
x_5	-0.1	0	0	-0.4	1.4	-0.9
x_6	0	0	0	-0.5	-0.9	1.4

- **L is symmetric ➔ it has real and positive eigenvalues (and ⊥ eigen vectors)**
- **Compute and sort the eigenvalues, then project examples $x_i \in \Re^d$ on the k eigen vectors of highest eigen values → new input space $s_i \in \Re^k$, in which separation in clusters will be easier**

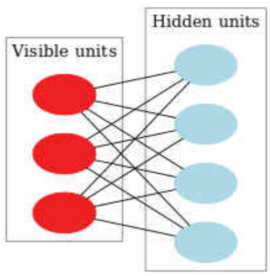


Other UNsupervised algos

- Learn the PROBABILITY DISTRIBUTION:
  - *Restricted Boltzmann Machine (RBM)*
  - *etc…*
- Learn a kind of « PROJECTION » into a LOWER DIMENSION SPACE (« Manifold Learning ») :
  - *<u>Non-linear</u> Principle Componant Analysis (PCA), (e.g. kernel-based)*
  - *Auto-encoders*
  - *Kohonen topological Self-Organizing Maps (SOM)*
  - *…*

Restricted Boltzmann Machine

- Proposed by Smolensky (1986) + *<u>Hinton (2005)</u>*
- Learns the probability distribution of examples
- Two-layers Neural Networks with BINARY neurons and *<u>bidirectional</u>* connections



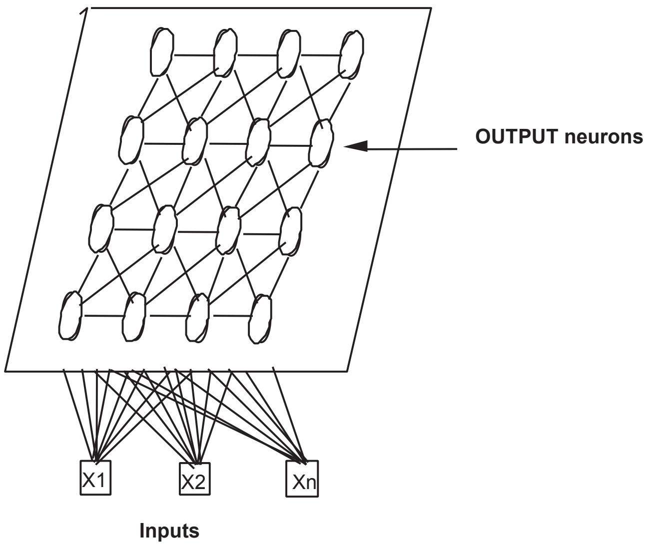
- Use: $P(v,h) = \frac{1}{Z} e^{-E(v,h)}$ $P(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}$

  where $E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i w_{i,j} h_j$ = energy
- Training: maximize product of probabilities $\Pi_i P(v_i)$ by gradient descent with *Contrastive Divergence*

  $\Delta w_{i,j} = \epsilon(vh^{\mathsf{T}} - v'h'^{\mathsf{T}})$ v' = reconstruction from h and h' deduced from v'

Kohonen Self-Organizing Maps (SOM)

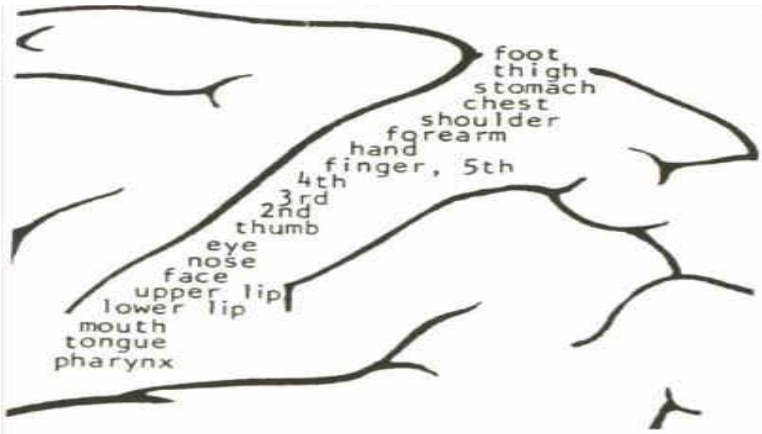
Another *specific* type of Neural Network



…with a self-organizing training algorithm which generates a MAPPING from input space to the Map THAT RESPECTS THE TOPOLOGY OF DATA

Inspiration and use of SOM

Biological inspiration: self-organization of regions in perception parts of brain.

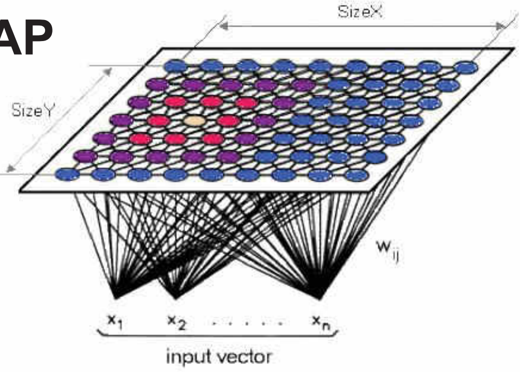


USE IN DATA ANALYSIS

- **VISUALIZE (generally in 2D) the distribution of data with a topology-preserving "projection" (2 points close in input space should be projected on close cells on the SOM)**
- **CLUSTERING**

The Kohonen Neural Network

- only ONE layer of neurons = output MAP
- type of neurons = ***DISTANCE neurons***
- use some defined "neighbourhood" on the output map



- each neuron should be seen as a vector in input space (corresponding to its vector of weights)
- USE after training: for each input vector X (in $\Re^d$), each neuron k on the Map compute its output = $d(W_k,X)$
➔ input X associated to the « winner » neuron
= the one with smallest output

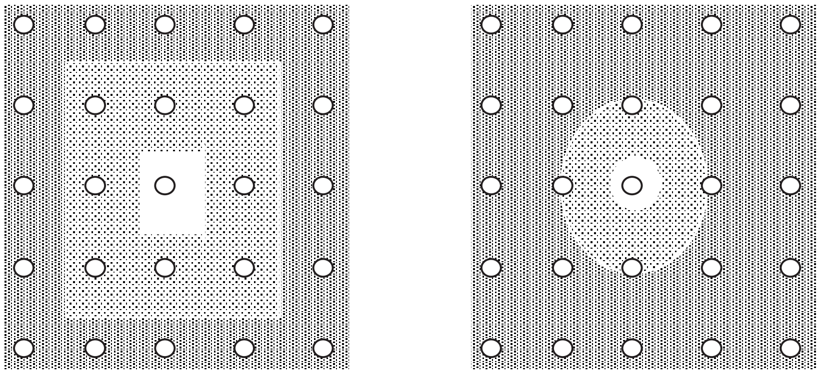
$\Rightarrow$ non-linear projection $\Re^d \rightarrow$ map
+ possible use for clustering

Training principle of Kohonen SOM

- The output of neuron i with weight vector $W_i = (w_{i1}, \dots, w_{in})$ when input is $X = (x_1, \dots, x_n)$ is the Euclidian distance $d(X,W_i)$
- <u>TRAINING principle:</u>
  - Determine most active neuron (= closest)
  - <u>Modify its weight vector to make it even closer to input + MOVE ALSO NEIGHBORING NEURONS TOWARDS INPUT</u>
- 2 parameters: shape and size of neighbourhood (on Map)
Weight modification step $\alpha(t)$
THEY BOTH DECREASE ALONG ITERATIONS

Neighborhoods on Kohonen Map

One possible type: finite-size with given shape



$V_i(t)$ size normally ***decreases when iteration t grow***

Another often used neighborhood type: « infinite » neigborhood with Gaussian width decreasing with iterations

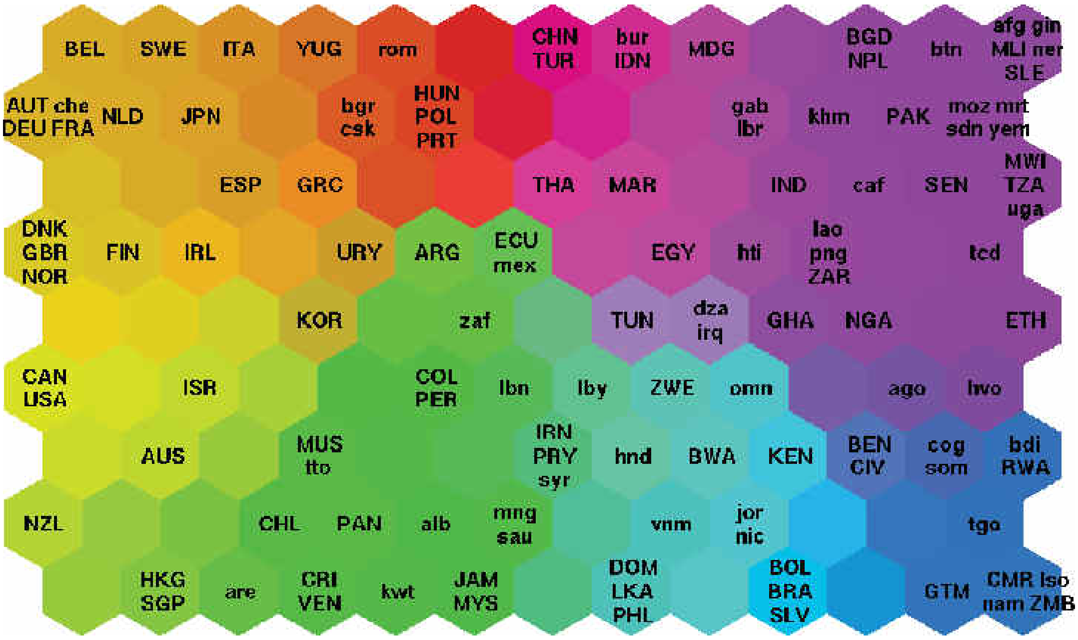
KOHONEN MAP TRAINING ALGORITHM

- **t=0, initialize weights (usually randomly)**
- **for each iteration t, present training example X and:**
  - **determine the <u>"winner" neuron g</u>**
    (higher output = weight vector most similar to X)
  - **determine learning step $\alpha(t)$ [and neighborhood V(t)]**
  - **modify weights:**
    $W_i(t+1) = W_i(t) + \alpha(t)\,(X-W_i(t))\,\beta(i,g,t)$
    with $\beta(i,g,t)=1$ if $i \in V(t)$, or otherwise 0 [IF finite-size V(t)]
    or $\beta(i,g,t)=\exp(-\mathrm{dist}(i,g)^2/\sigma(t)^2)$ [Gaussian V(t)]
- **t = t+1**

- **Training can be proved to converge under condition on $\alpha(t)$**
  (e.g. $\alpha(t) \propto 1/t$ is OK, and often used)

[See demo applet?]

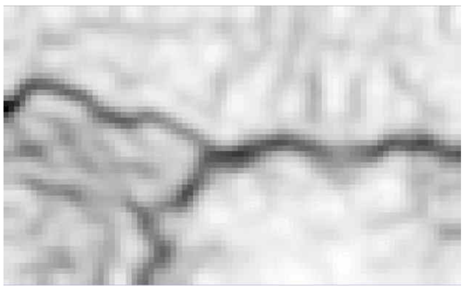
Example applications of SOM



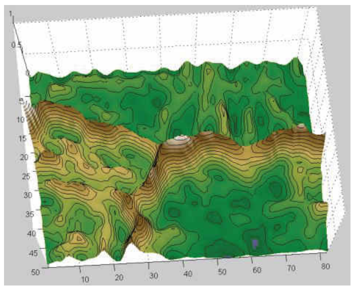
Result of a training on a dataset in which each example is a country represented by a vector of 39 indicators of quality of life (health, life duration expectation, nutrition, education services, etc…

Use of SOM for clustering

Analysis of distances between neurons of the SOM (U-matrix)

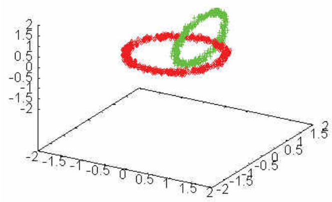


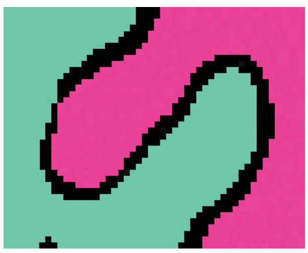
Grey level (darker = bigger distance)



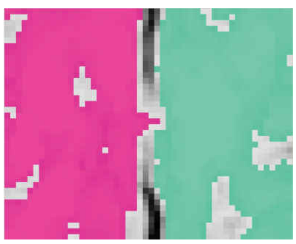
Idem in « 3D view » (courbes de niveau)

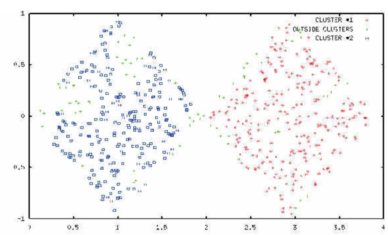
➔ **Possibility of automated segmentation, which produces a clustering *with no a priori on # and shapes of clusters***





« ChainLink » example



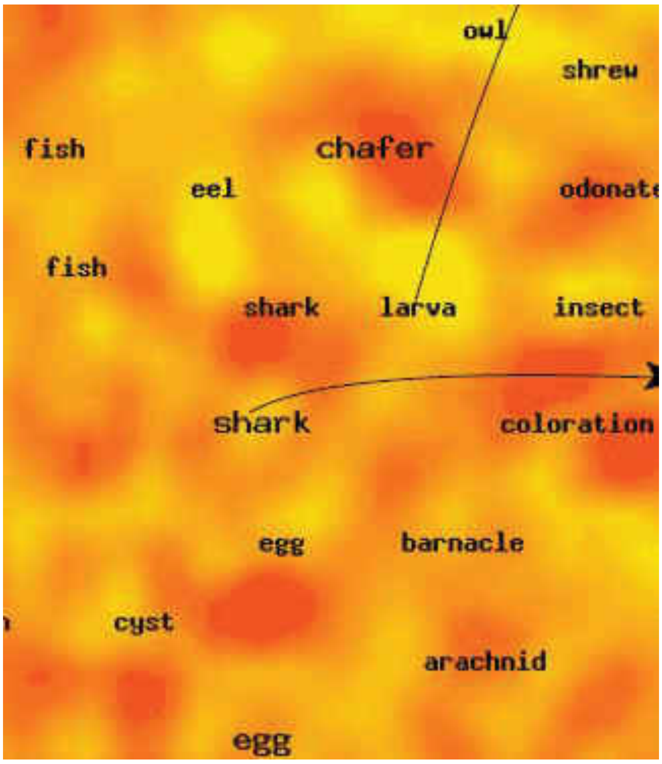


« TwoDiamonds » example

Application of Kohonen to « text-mining »

- Each document represented as a histogram of the words it contains

- On the right, extract of a Kohonen map obtained with articles from Encyclopedia Universalis…



WebSOM (see demo, etc… at http://websom.hut.fi/websom)

SOME REFERENCE TEXTBOOKS ON MACHINE-LEARNING

- ***The Elements of Statistical Learning*** *(2nd edition)*
  T. Hastier, R. Tibshirani & J. Friedman, Springer, 2009.
  http://statweb.stanford.edu/~tibs/ElemStatLearn/
- ***Deep Learning***
  I. Goodfellow, Y. Bengio & A. Courville, MIT press, 2016.
  http://www.deeplearningbook.org/
- ***Pattern recognition and Machine-Learning***
  C. M. Bishop, Springer, 2006.
- ***Introdution to Data Mining***
  P.N. Tan, M. Steinbach & V. Kumar, AddisonWeasley, 2006.
- ***Machine Learning***
  T. Mitchell, McGraw-Hill Science/Engineering/Math, 1997.
- ***Apprentissage artificiel : concepts et algorithmes***
  A. Cornuéjols, L. Miclet & Y. Kodratoff, Eyrolles, 2002.