Self-organizing map

Data visualization, feature reduction and cluster analysis

Ricco RAKOTOMALALA
Université Lumière Lyon 2

Outline

1. SOM – Kohonen map – Kohonen network
2. Learning algorithm
3. Data visualization
4. Assigning a new instance to a node
5. Tools – Case study (R, Tanagra)
6. Cluster analysis from SOM
7. Supervised self-organizing map
8. Conclusion
9. References

Self-organizing map

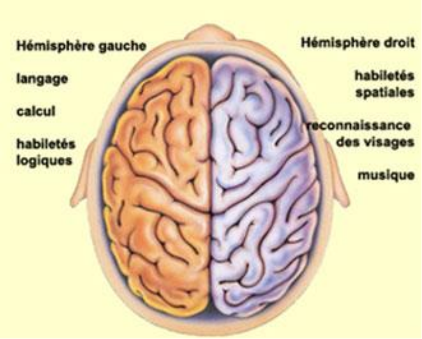
Kohonen network

Self-organizing map

Kohonen map, Kohonen network

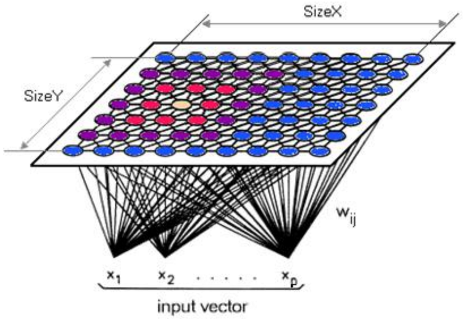
Biological metaphor

Our brain is subdivided into specialized areas, they specifically respond to certain stimuli i.e. stimuli of the same kind activate a particular region of the brain.



Kohonen map

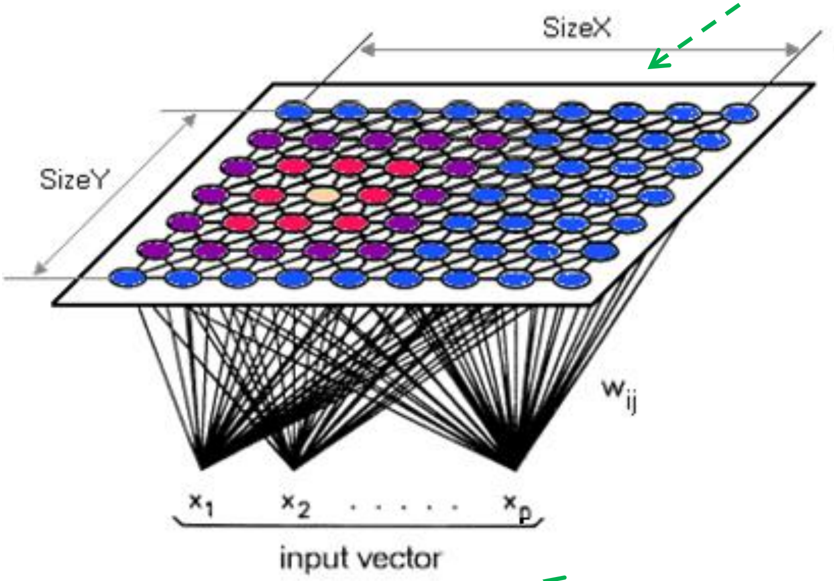
The idea is transposed to a competitive unsupervised learning system where the input space is "mapped" in a small (often rectangular) space with the following principle: similar individuals in the initial space will be projected into the same neuron or, at least, in neighboring neurons in the output space (preservation of proximity).



⇨ **SOM serves both to the dimensionality reduction, data visualization and cluster analysis (clustering).**

SOM - Architecture

- To each neuron (node) corresponds a set of instances from the dataset.
- To each neuron (node) is associated a vector of weights (**codebook**) which describes the typical profile of the neuron.
- The positions of the neurons in the map are important i.e. (1) two neighboring neurons have similar codebook; (2) a set of contiguous neurons correspond to a particular pattern in the data.

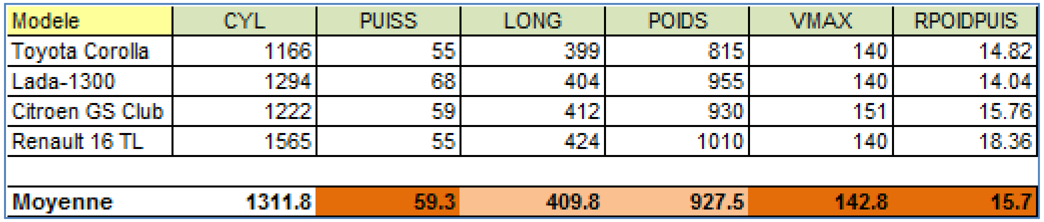


The connections between the input and output layers indicate the relationships between the input and output vectors.

Input space, description of the dataset into the original representation space (vector with p values [for p variables]).

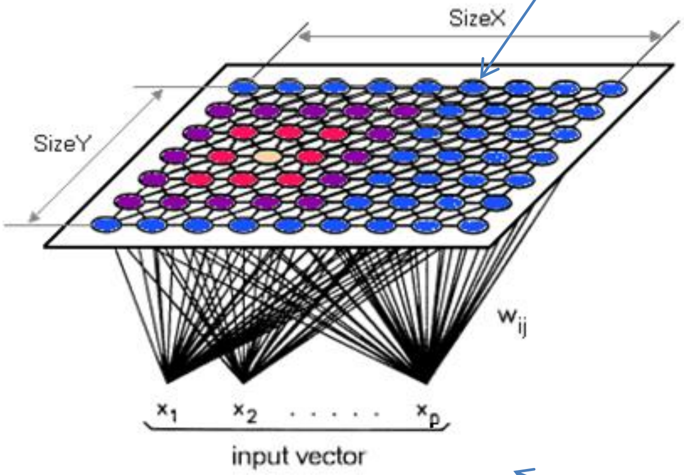
SOM – Example (1)

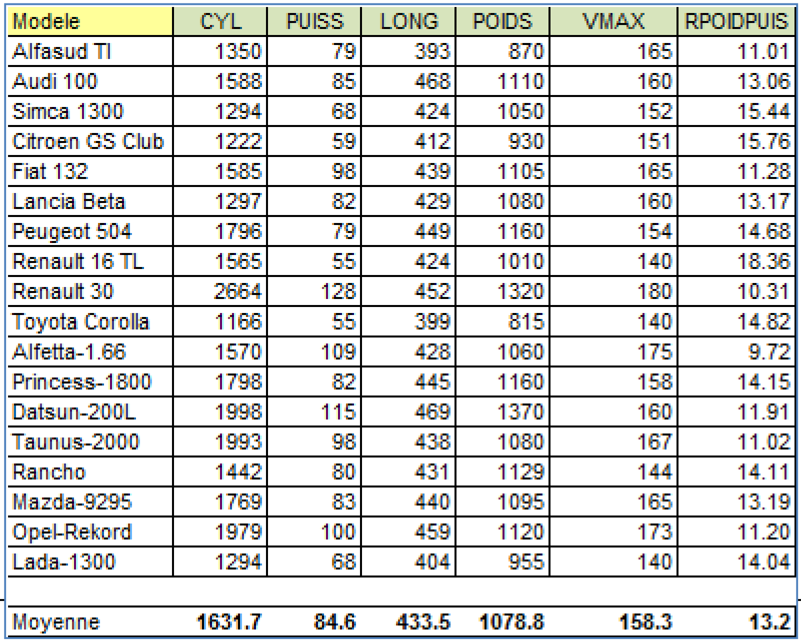
“Weights” (codebook) of the node



Modele	CYL	PUISS	LONG	POIDS	VMAX	RPOIDPUIS
Toyota Corolla	1166	55	399	815	140	14.82
Lada-1300	1294	68	404	955	140	14.04
Citroen GS Club	1222	59	412	930	151	15.76
Renault 16 TL	1565	55	424	1010	140	18.36
Moyenne	**1311.8**	**59.3**	**409.8**	**927.5**	**142.8**	**15.7**

A neuron a "small" cars (4 cars above), poorly performing, with bad power-to-weight ratio.





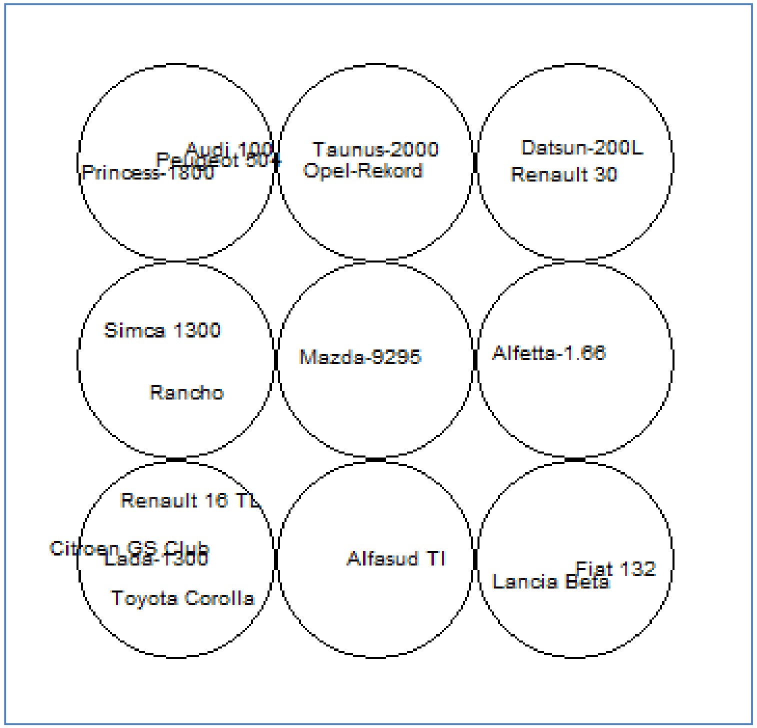
Modele	CYL	PUISS	LONG	POIDS	VMAX	RPOIDPUIS
Alfasud TI	1350	79	393	870	165	11.01
Audi 100	1588	85	468	1110	160	13.06
Simca 1300	1294	68	424	1050	152	15.44
Citroen GS Club	1222	59	412	930	151	15.76
Fiat 132	1585	98	439	1105	165	11.28
Lancia Beta	1297	82	429	1080	160	13.17
Peugeot 504	1796	79	449	1160	154	14.68
Renault 16 TL	1565	55	424	1010	140	18.36
Renault 30	2664	128	452	1320	180	10.31
Toyota Corolla	1166	55	399	815	140	14.82
Alfetta-1.66	1570	109	428	1060	175	9.72
Princess-1800	1798	82	445	1160	158	14.15
Datsun-200L	1998	115	469	1370	160	11.91
Taunus-2000	1993	98	438	1080	167	11.02
Rancho	1442	80	431	1129	144	14.11
Mazda-9295	1769	83	440	1095	165	13.19
Opel-Rekord	1979	100	459	1120	173	11.20
Lada-1300	1294	68	404	955	140	14.04
Moyenne	**1631.7**	**84.6**	**433.5**	**1078.8**	**158.3**	**13.2**

Cars dataset.

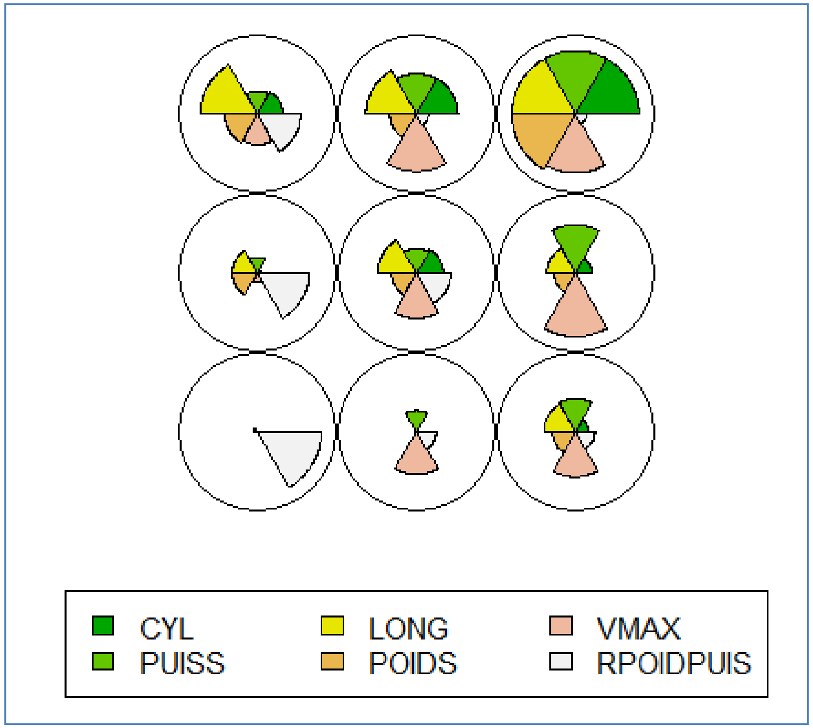
p = 6 variables.

SOM – Example (2)

A **rectangular grid** with (3 x 3) neurons



Mapping plot

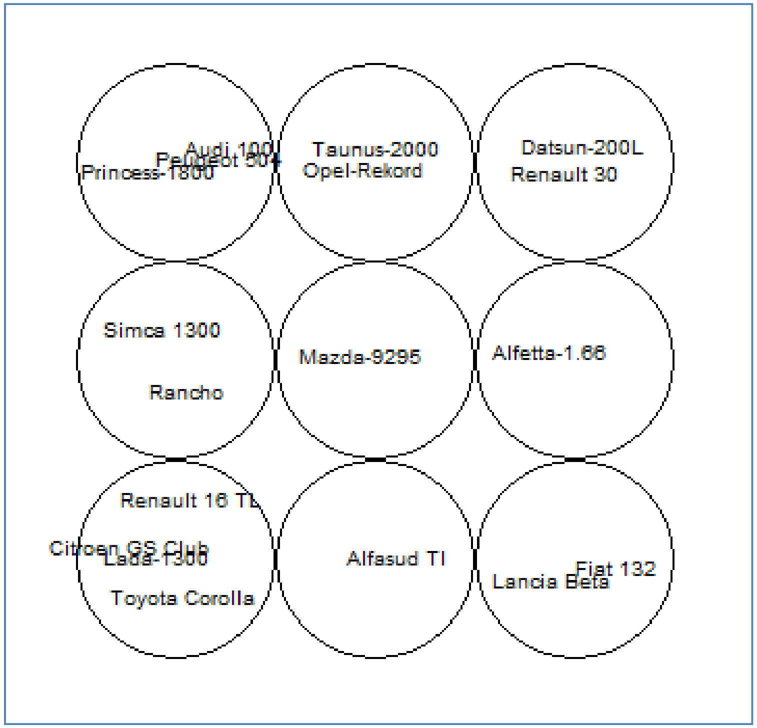


Codebooks plot

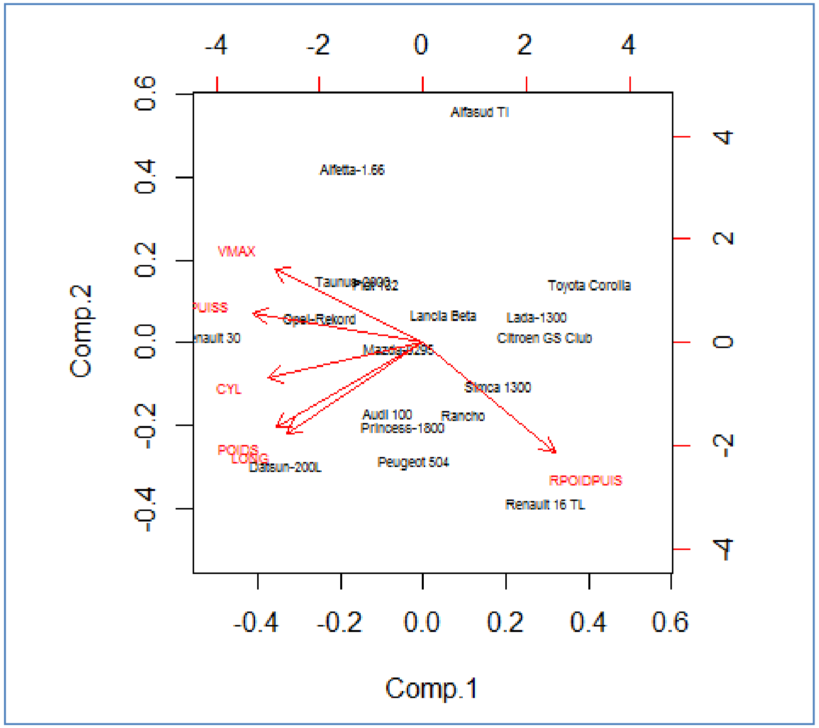
We have both a visualization tool (the proximity between the neurons has meaning) and clustering tool (we have a first organization of the data in groups).

SOM and PCA (principal component analysis)

PCA is a popular statistical method for dimensionality reduction and data visualization.



Mapping plot



Biplot

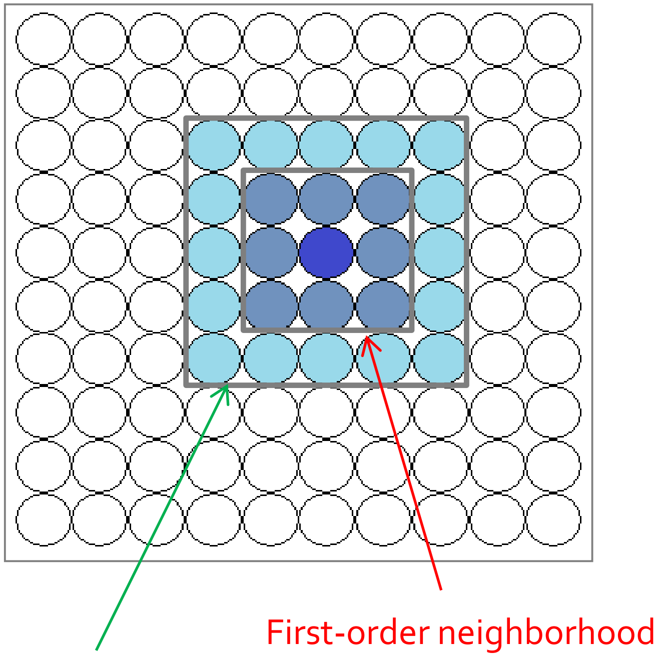
We can see roughly the same proximities. But there is a linear constraint in the PCA (the components are linear combinations of the initial variables) that does not exist in SOM. This constraint, as well as the orthogonality between the factors, can be a drawback for the handling of nonlinear problems (see the example at Wikipedia). Unlike PCA, the output of SOM is in 2D space (very often).

SOM

Architecture and neighborhood

The notion of neighborhood is essential in SOM, especially for the updating of weights and their propagation during the learning process.

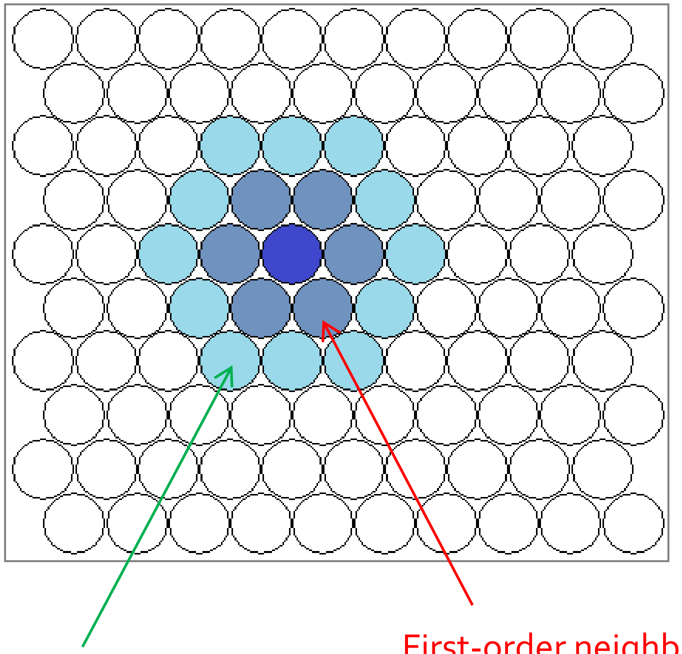
Rectangular grid - Rectangular neighborhood



First-order neighborhood

Second-order neighborhood

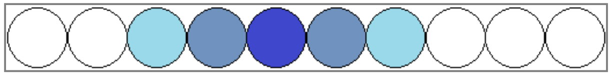
Hexagonal grid - Circular neighborhood



First-order neighborhood

Second-order neighborhood

Note: an unidimensional map (vector) is possible



SOM – Learning algorithm

Initialization, competition, cooperation, adaptation

SOM – Algorithm (rough description)

Scale the variables to avoid the over influence of the variables with high variance

```
Input: dataset, size and topology of the map
Output: the codebook for each output node

1. Randomize the map's nodes weight
2. Select randomly one instance
3. Find the closest node: best matching unit
4. The codebook of this node is updated
5. The codebooks of the adjacent nodes are
   also updated, by not to the same degree
6. Reduce the intensity of the update
   progressively
7. Repeat 1 to 6 for Tmax iterations
```

(1) Initialization phase

(2) We process all instances from the data. One instance may be processed several times.

(3) Competition phase. A distance measure must be defined between the codebook and the description of the instances.

(4) Update the weights (codebook).

(5) Cooperation phase. This is what ensures the similarity of weights between contiguous nodes. The size of the neighborhood to be considered is reduced progressively. Node: if we ignore the neighborhood structure, the algorithm corresponds to the K-means approach.

(6) Adaptation. At first, high learning rate, move quickly to the solution; at the end, small learning rate, to avoid oscillations.

SOM – Algorithm – Details (1)

σ_o, ε_o and Tmax are parameters of the algorithm

```
Input: dataset, size and topology
of the map
Output: the codebook for each
output node

1.  Randomize the map's nodes
    weight
2.  Select randomly one instance
3.  Find the closest node: best
    matching unit
4.  The codebook of this node is
    updated
5.  The codebooks of the adjacent
    nodes are also updated, by not
    to the same degree
6.  Reduce the intensity of the
    update progressively
7.  Repeat 1 to 6 for Tmax
    iterations
```

Weight update rule for a node j, knowing that j* is the winning node

$$w_{t+1}(j) = w_t(j) + \varepsilon_t \times h_t(j, j^*) \times \left(w_t(j) - x\right)$$

(a) h() is a neighborhood function. Its amplitude (spatial width of the kernel) decreases according to the step index (t)

$$h_t(j, j^*) = \exp\left(-\frac{d^2(j, j^*)}{2\sigma^2(t)}\right)$$

Où

d(j,j*) is the distance between the nodes j and j* on the map

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\text{Tmax}}\right)$$

(b) ε is the learning rate. Its value decreases according the step index (t)

$$\varepsilon_t = \varepsilon_0 \exp\left(-\frac{t}{\text{Tmax}}\right)$$

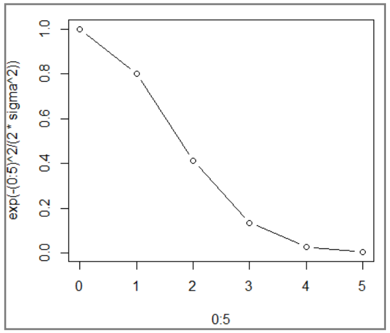
Implementations differ from one software to another, but the guiding ideas are there.
Gradual reduction: of the size of the neighborhood to consider, of the learning rate.

SOM – Algorithm – Details (2)

$\sigma_0 = 1.5$, Tmax = 20

Influence of the neighborhood distance [d(j,j*) = 0, …, 5 (**t = 0**)] on the neighborhood function

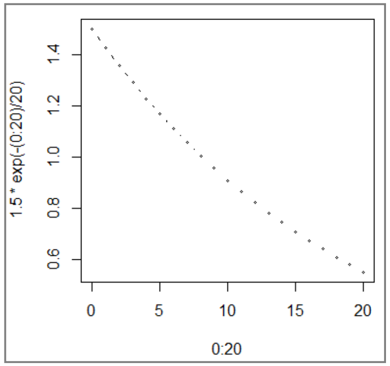
$$h_0(j, j^*) = \exp\left(-\frac{d^2(j, j^*)}{2\sigma^2(0)}\right)$$



First order neighborhood: learning rate= 0.8 ; second order : 0.41 ; etc.

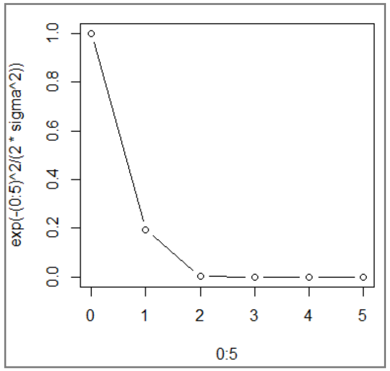
Decreasing of the influence of the neighborhood according the step index (t = 0, …, 20)

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\text{Tmax}}\right)$$



Influence of the neighborhood distance [d(j,j*) = 0, …, 5 (**t = 20**)] on the neighborhood function

$$h_{20}(j, j^*) = \exp\left(-\frac{d^2(j, j^*)}{2\sigma^2(20)}\right)$$

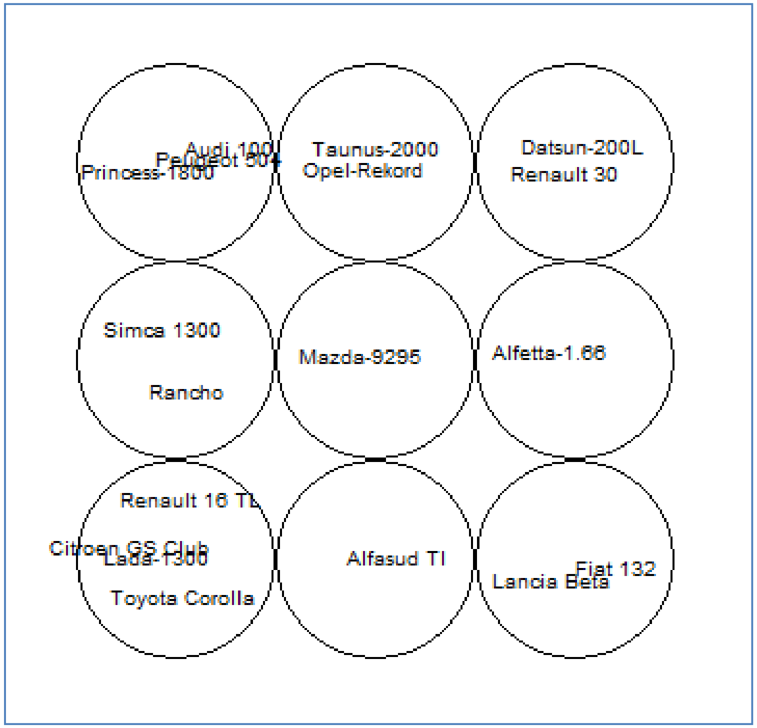


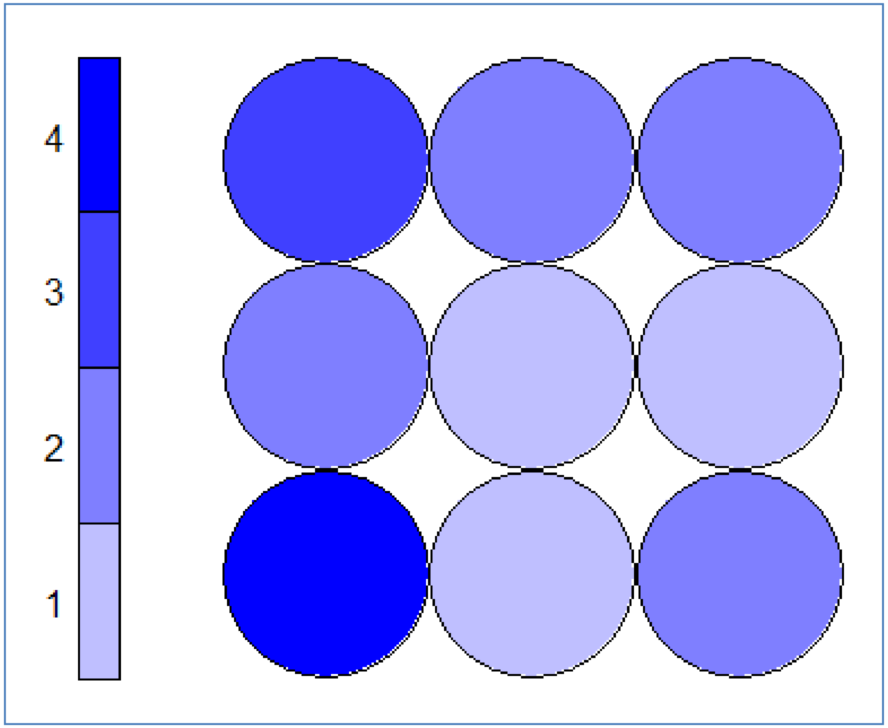
The influence on the neighbors decreases over the iterations. For t = 20, only the first order neighborhood is affected by the update process.

Visualisations

SOM provides various very interesting data visualization scenarios

Visualization – Sample size, list of individuals



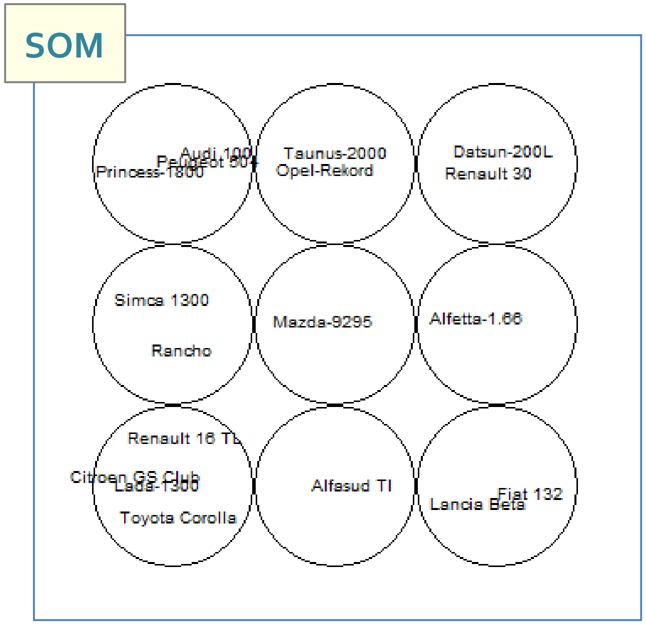


Labels of individuals, impracticable when we deal with large dataset (number of instances).

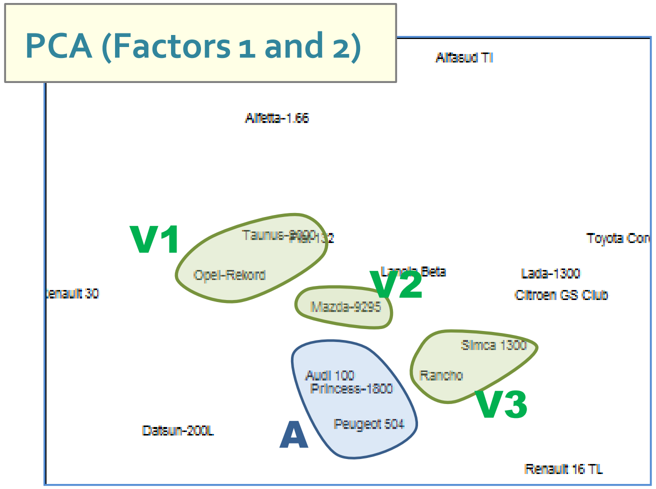
Color code. It enables to identify areas with a high density of individuals. Useful on large databases.

Visualization – Distance between nodes (U-matrix)

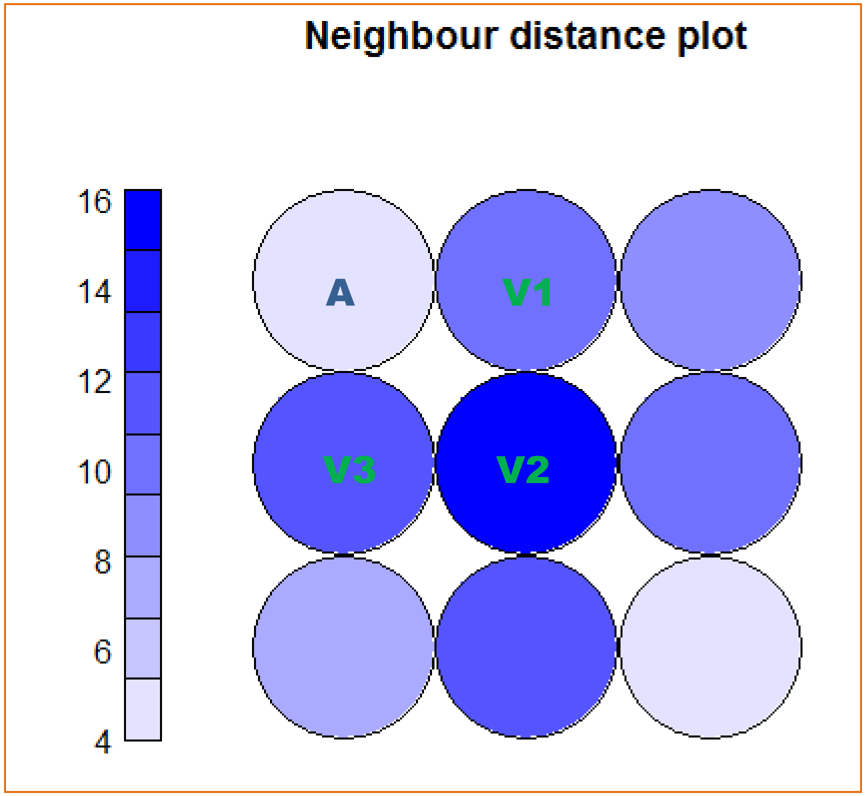
SOM



PCA (Factors 1 and 2)



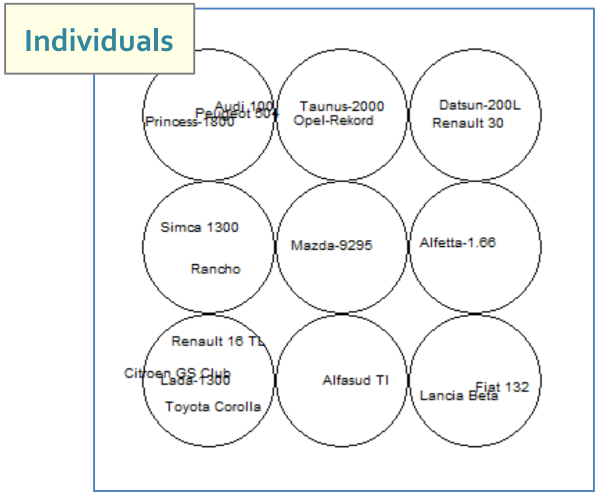
Euclidean distance between codebooks of neighboring nodes.



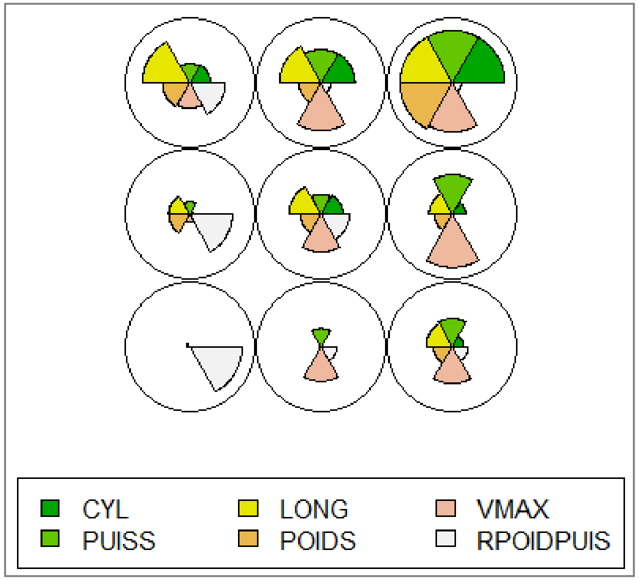
Distance to the immediate neighbors of each node. E.g. the node including « Audi 100 », « Peugeot 504 » and « Princess 1800 » is close to its immediate neighbors (V1, V2 and V3). See the position of the groups in the representation space defined by the two first components of the PCA.

Visualization – Characterization by variables

Objective: Understand what characterizes the regions of the topological map



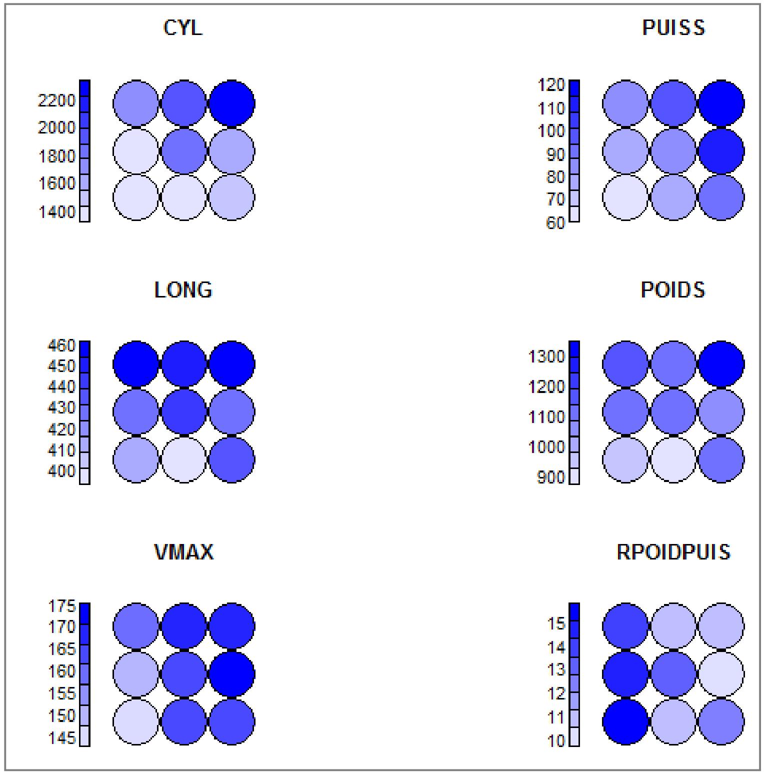
Codebooks



Impracticable when the number of variables is high.

Heatmaps

Comparing the conditional averages



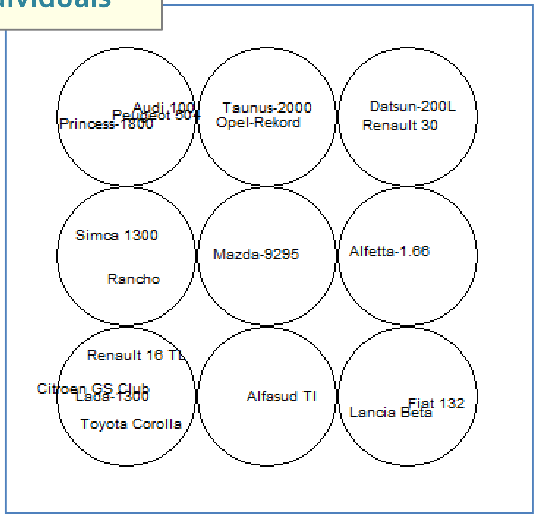
We can have a global view. But the larger number of the graphs does not make things easy. *The correlation ratio can be used to determine the overall contrast for each variable on the output nodes*.

Visualization – Characterization by variables

It can be extended to illustrative variables

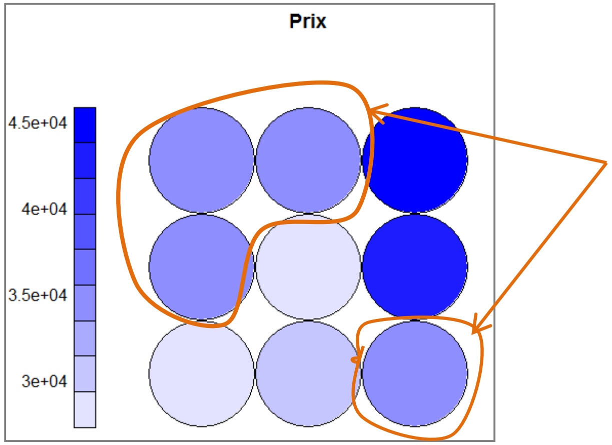
Illustrative variables are not used on the construction of the network, but used to help the interpretation of the results.

Individuals



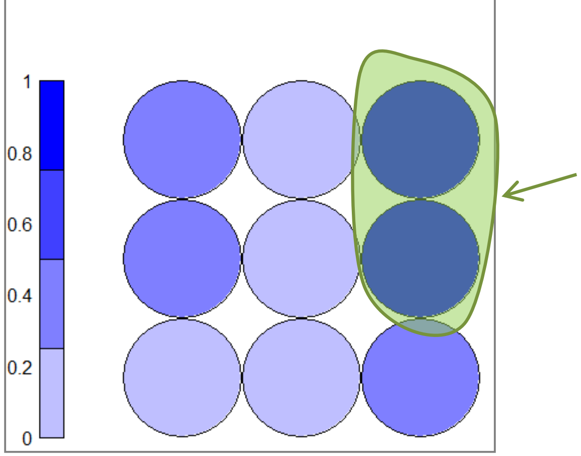
Modele	FINITION	PRIX
Alfasud TI	B	30570
Audi 100	TB	39990
Simca 1300	M	29600
Citroen GS Club	M	28250
Fiat 132	B	34900
Lancia Beta	TB	35480
Peugeot 504	B	32300
Renault 16 TL	B	32000
Renault 30	TB	47700
Toyota Corolla	M	26540
Alfetta-1.66	TB	42395
Princess-1800	B	33990
Datsun-200L	TB	43980
Taunus-2000	B	35010
Rancho	TB	39450
Mazda-9295	M	27900
Opel-Rekord	B	32700
Lada-1300	M	22100

Average price according to the output cells.



Same price range, but not for the same reasons

Proportion of very good finishes



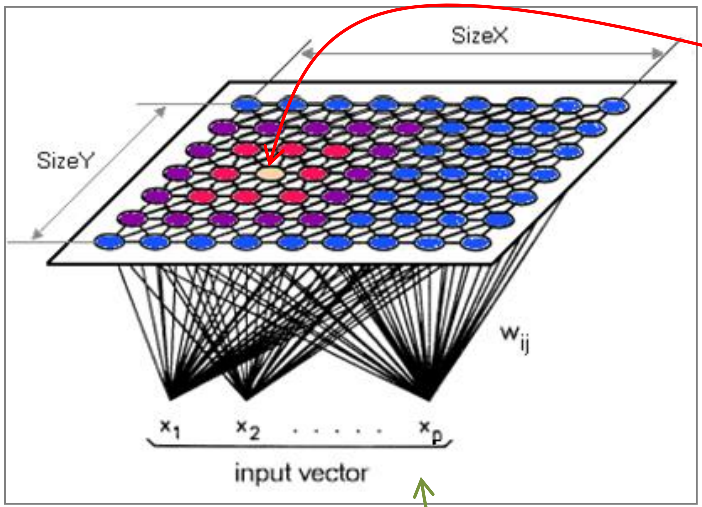
The most expensive cars are the ones with the best finishes.

Assigning a new instance to a node

Assignment to the most closest node

Predicting the node membership of a new instance

Predicting the node membership of a new instance. This operation will be really essential when we use the SOM network for the cluster analysis.



Identify the output neuron (winning neuron) in the sense of the smallest distance to the codebook of the neurons (e.g. Euclidean distance).

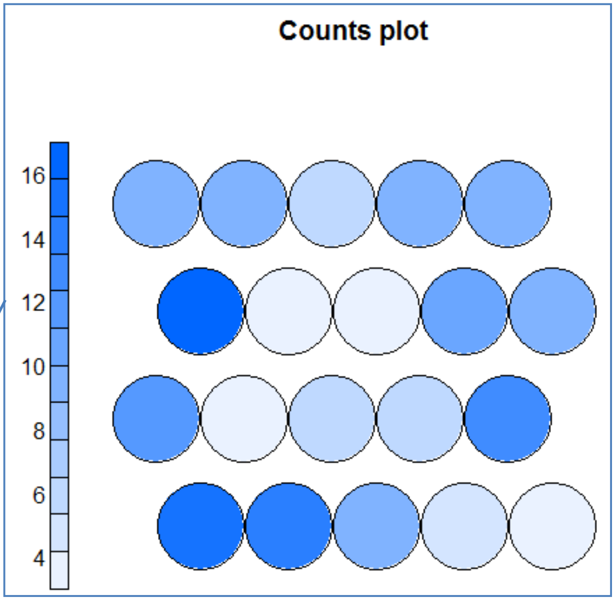
Submit the new instant to the input layer, with possibly the data transformation used during the learning phase (standardization,…).

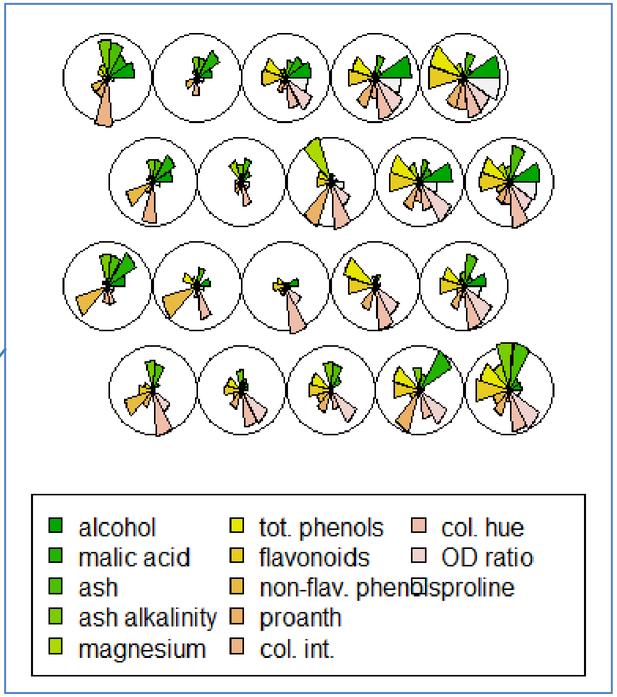
Data Mining Tools

R (Kohonen package), Tanagra

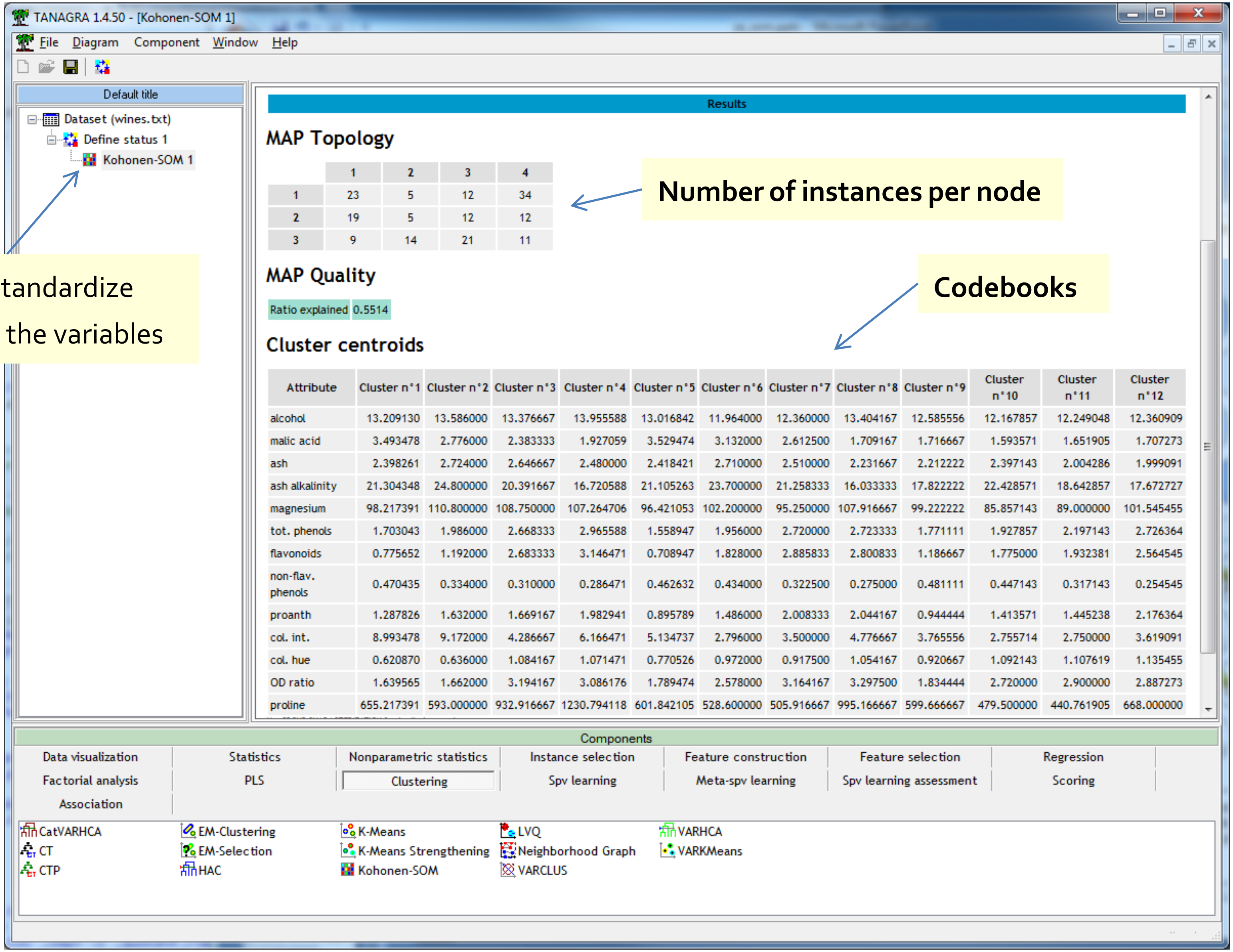
R – « kohonen » package

```
#package kohonen
library(kohonen)

#wines dataset, included in the package (n = 177, p = 13)
data(wines)
print(summary(wines))

#Z - standardization of the variables
Z <- scale(wines,center=T,scale=T)

#learning phase - hexagonal grid
grille <- som(Z,grid=somgrid(5,4,"hexagonal"))

#shades of blue colors
degrade.bleu <- function(n){
  return(rgb(0,0.4,1,alpha=seq(1/n,1,1/n)))
}

#plot number of instances per node
plot(grille,type="count",palette.name=degrade.bleu)

#plot the codebook
plot(grille,type="codes",codeRendering = "segments")
```





Tanagra – « Kohonen – SOM » component



Results

MAP Topology

	1	2	3	4
1	23	5	12	34
2	19	5	12	12
3	9	14	21	11

Number of instances per node

MAP Quality

Ratio explained	0.5514

Cluster centroids

Codebooks

Attribute	Cluster n°1	Cluster n°2	Cluster n°3	Cluster n°4	Cluster n°5	Cluster n°6	Cluster n°7	Cluster n°8	Cluster n°9	Cluster n°10	Cluster n°11	Cluster n°12
alcohol	13.209130	13.586000	13.376667	13.955588	13.016842	11.964000	12.360000	13.404167	12.585556	12.167857	12.249048	12.360909
malic acid	3.493478	2.776000	2.383333	1.927059	3.529474	3.132000	2.612500	1.709167	1.716667	1.593571	1.651905	1.707273
ash	2.398261	2.724000	2.646667	2.480000	2.418421	2.710000	2.510000	2.231667	2.212222	2.397143	2.004286	1.999091
ash alkalinity	21.304348	24.800000	20.391667	16.720588	21.105263	23.700000	21.258333	16.033333	17.822222	22.428571	18.642857	17.672727
magnesium	98.217391	110.800000	108.750000	107.264706	96.421053	102.200000	95.250000	107.916667	99.222222	85.857143	89.000000	101.545455
tot. phenols	1.703043	1.986000	2.668333	2.965588	1.558947	1.956000	2.720000	2.723333	1.771111	1.927857	2.197143	2.726364
flavonoids	0.775652	1.192000	2.683333	3.146471	0.708947	1.828000	2.885833	2.800833	1.186667	1.775000	1.932381	2.564545
non-flav. phenols	0.470435	0.334000	0.310000	0.286471	0.462632	0.434000	0.322500	0.275000	0.481111	0.447143	0.317143	0.254545
proanth	1.287826	1.632000	1.669167	1.982941	0.895789	1.486000	2.008333	2.044167	0.944444	1.413571	1.445238	2.176364
col. int.	8.993478	9.172000	4.286667	6.166471	5.134737	2.796000	3.500000	4.776667	3.765556	2.755714	2.750000	3.619091
col. hue	0.620870	0.636000	1.084167	1.071471	0.770526	0.972000	0.917500	1.054167	0.920667	1.092143	1.107619	1.135455
OD ratio	1.639565	1.662000	3.194167	3.086176	1.789474	2.578000	3.164167	3.297500	1.834444	2.720000	2.900000	2.887273
proline	655.217391	593.000000	932.916667	1230.794118	601.842105	528.600000	505.916667	995.166667	599.666667	479.500000	440.761905	668.000000

The tool can standardize automatically the variables

Self-organizing map for cluster analysis

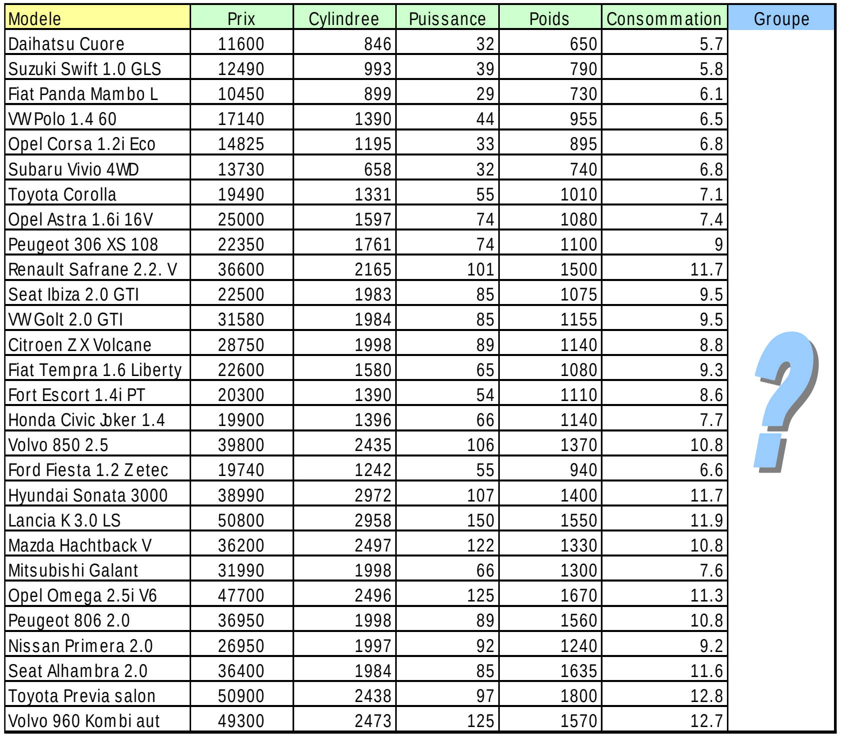
Two step clustering – Large dataset processing

Cluster analysis

Also called: clustering, unsupervised learning, numerical taxonomy, typological analysis

Input X (all continuous)

No target attribute



Modele	Prix	Cylindree	Puissance	Poids	Consommation	Groupe
Daihatsu Cuore	11600	846	32	650	5.7	
Suzuki Swift 1.0 GLS	12490	993	39	790	5.8	
Fiat Panda Mambo L	10450	899	29	730	6.1	
VW Polo 1.4 60	17140	1390	44	955	6.5	
Opel Corsa 1.2i Eco	14825	1195	33	895	6.8	
Subaru Vivio 4WD	13730	658	32	740	6.8	
Toyota Corolla	19490	1331	55	1010	7.1	
Opel Astra 1.6i 16V	25000	1597	74	1080	7.4	
Peugeot 306 XS 108	22350	1761	74	1100	9	
Renault Safrane 2.2. V	36600	2165	101	1500	11.7	
Seat Ibiza 2.0 GTI	22500	1983	85	1075	9.5	
VW Golt 2.0 GTI	31580	1984	85	1155	9.5	
Citroen ZX Volcane	28750	1998	89	1140	8.8	
Fiat Tempra 1.6 Liberty	22600	1580	65	1080	9.3	?
Fort Escort 1.4i PT	20300	1390	54	1110	8.6	
Honda Civic Joker 1.4	19900	1396	66	1140	7.7	
Volvo 850 2.5	39800	2435	106	1370	10.8	
Ford Fiesta 1.2 Zetec	19740	1242	55	940	6.6	
Hyundai Sonata 3000	38990	2972	107	1400	11.7	
Lancia K 3.0 LS	50800	2958	150	1550	11.9	
Mazda Hachtback V	36200	2497	122	1330	10.8	
Mitsubishi Galant	31990	1998	66	1300	7.6	
Opel Omega 2.5i V6	47700	2496	125	1670	11.3	
Peugeot 806 2.0	36950	1998	89	1560	10.8	
Nissan Primera 2.0	26950	1997	92	1240	9.2	
Seat Alhambra 2.0	36400	1984	85	1635	11.6	
Toyota Previa salon	50900	2438	97	1800	12.8	
Volvo 960 Kombi aut	49300	2473	125	1570	12.7	

Goal: Identifying the set of objects with similar characteristics

We want that:

(1) The objects in the same group are more similar to each other

(2) Thant to those in other groups

For what purpose?

→ Identify underlying structures in the data

→ Summarize behaviors or characteristics

→ Assign new individuals to groups

→ Identify totally atypical objects

The aim is to detect the set of “similar” objects, called groups or clusters. “Similar” should be understood as “which have close characteristics”.

SOM: we can perform directly a clustering by limiting the number of output nodes

But nothing really distinguishes the approach from the K-means method in this case.

!

Two-step clustering - Principle

Issue

The HAC (Hierarchical Agglomerative Clustering) requires the calculation of distances between each pair of individuals (distance matrix). It also requires to access to this matrix at each aggregation. This is too time consuming on large datasets (in number of observations).

Approach

The idea is to perform a pre-clustering using the SOM method which can process very large database, and start the HAC from these pre-clusters. Often (attention, not always), the adjacent nodes of the topological map belong to the same final cluster. The interpretation will be easier (interpretation of the map helps to better understand the groups obtained from the clustering process).

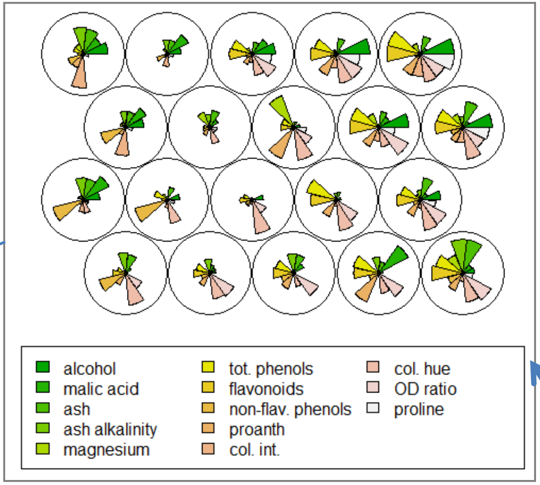
Advantage

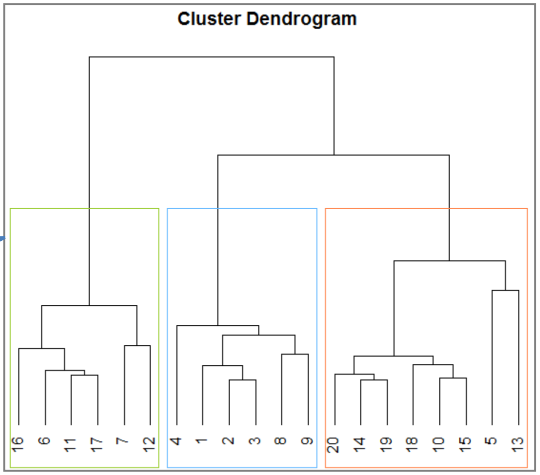
The approach allows to handle very large bases, while benefiting from the advantages of HAC (hierarchy of nested partitions, dendrogram for understanding and identification of clusters).

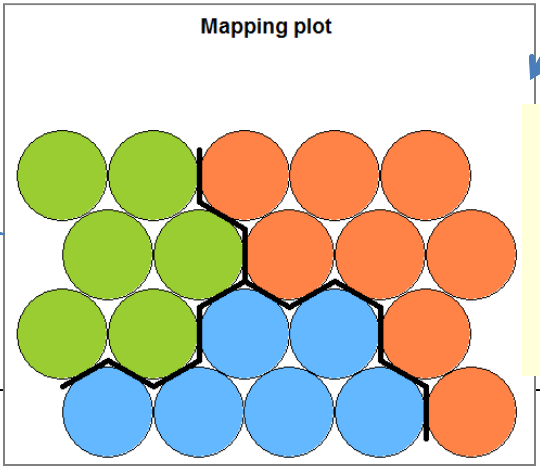
Two step clustering

Example under R (« wines » dataset)

```
#codebooks of the output nodes
plot(grille,type="codes",codeRendering = "segments")

#distance between nodes using the codebook
d <- dist(grille$codes)

#HAC – ward approach
#the number of instances in each node is ignored here
cah <- hclust(d,method="ward.D")
plot(cah,hang=-1)

#subdivision in 3 clusters
groupes <- cutree(cah,k=3)

#highlight the clusters into the dendrogram
rect.hclust(cah,k=3,border=c("yellowgreen","steelblue1","sienna1"))

#highlight the clusters into the Kohonen map
plot(grille,type="mapping",bgcol=c("steelblue1","sienna1","yellowgreen")[groupes])
add.cluster.boundaries(grille,clustering=groupes)
```



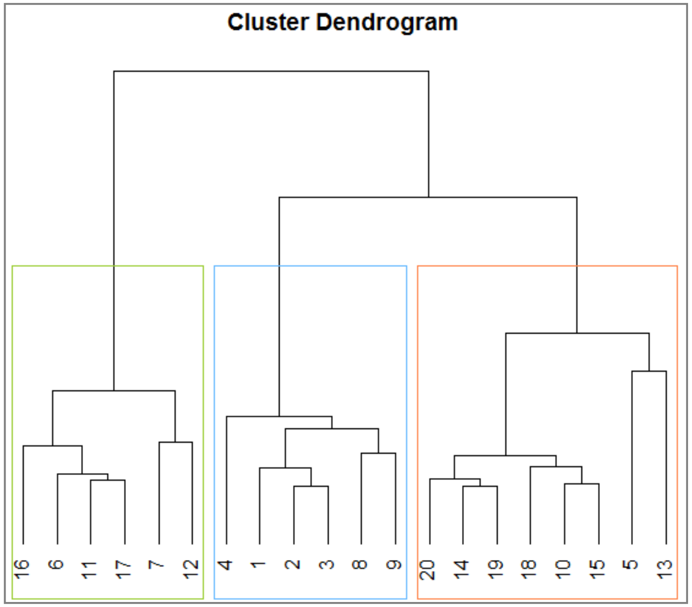


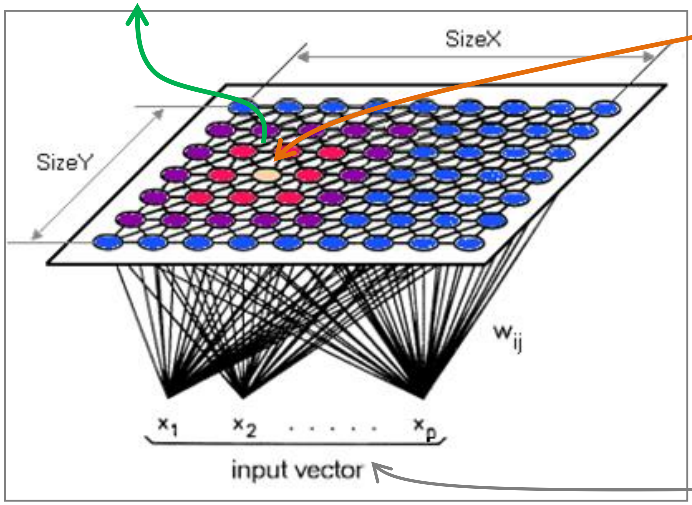


The codebooks enables to better understand the nature of the groups.

Assign a new instance to an existing cluster

Proceed in two steps: identify the node of the topological map associated to the new individual (See *Predicting the node membership of a new instance*), and then the cluster associated with this node.





Identify the cluster (group) associated with the output-layer neuron. The instance is assigned to this group.

Identify the output neuron (winning neuron) in the sense of the smallest distance to the codebook of the neurons (e.g. Euclidean distance).

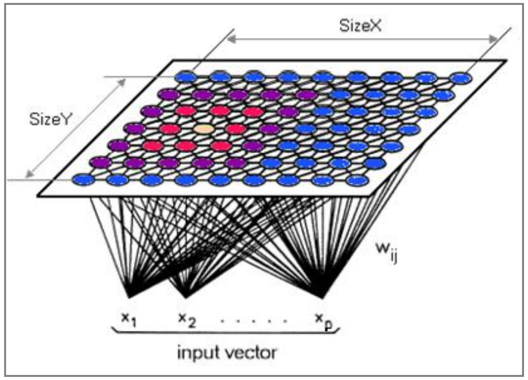
Submit the new instant to the input layer, with possibly the data transformation used during the learning phase (standardization,...).

Supervised self-organizing map

Extension of SOM to the supervised learning task

$$Y = f(X1, X2, \ldots ; \alpha)$$

Supervised SOM



Solution 1. Construct the map in the (usual) unsupervised fashion then, calculate the best prediction on each node (the most common value of Y in the classification context, average y in the regression context).

Solution 2. Add the information about the target attribute into the codebooks. Calculate D_X, distance to codebooks defined on the input attributes; and D_Y distance defined on the target attribute. Normalize D_X and D_Y to balance their influences (i.e. define each D in [0..1]), then calculate an overall distance that we can parameterize

$$D = \alpha.D_X + (1 - \alpha).D_Y$$

We vary α *according the relative importance that we attach to X and Y*

Conclusion

Pros

SOM serves both to the dimensionality reduction, data visualization and cluster analysis (clustering).

The two-step approach for clustering is especially attractive.

This is a nonlinear approach for dimensionality reduction (vs. PCA for instance)

Numerous visualization possibilities

The method is simple, easy to explain … and understand

Ability to handle large bases (linear complexity regarding the number of observations and variables)

Cons

But … the processing time may be long on very large bases (we need to pass several times the individuals)

The visualization and the interpretation of codebooks becomes difficult when the number of variables is very high

References

State-of-the art book

Kohonen T., « Self-Organizing Maps », 3rd Edition, Springer Series in Information Sciences, Vol. 30, 2001.

Course materials and tutorials

Ahn J.W., Syn S.Y., « Self Organizing Maps », 2005.

Bullinaria J.A., « Self Organizing Maps : Fundamentals », 2004.

Lynn Shane, « Self Organizing Maps for Customer Segmentation using R », R-bloggers, 2014.

Minsky M., « Kohonen's Self Organizing Features Maps ».

Tanagra tutorial, « Self-organizing map (SOM) », July 2009.

Wikibooks – Data Mining Algorithms in R, « Clustering / Self Organizing Maps (SOM) ».

Scholarpedia, "Kohonen network".