Data Mining

Ciro Donalek
donalek@astro.caltech.edu

Can I speak in Italian?

NO!

Latin?

NO!

C++ ?

NO!

OK, let's try in English...

Ciro, Yes you Can!

Summary

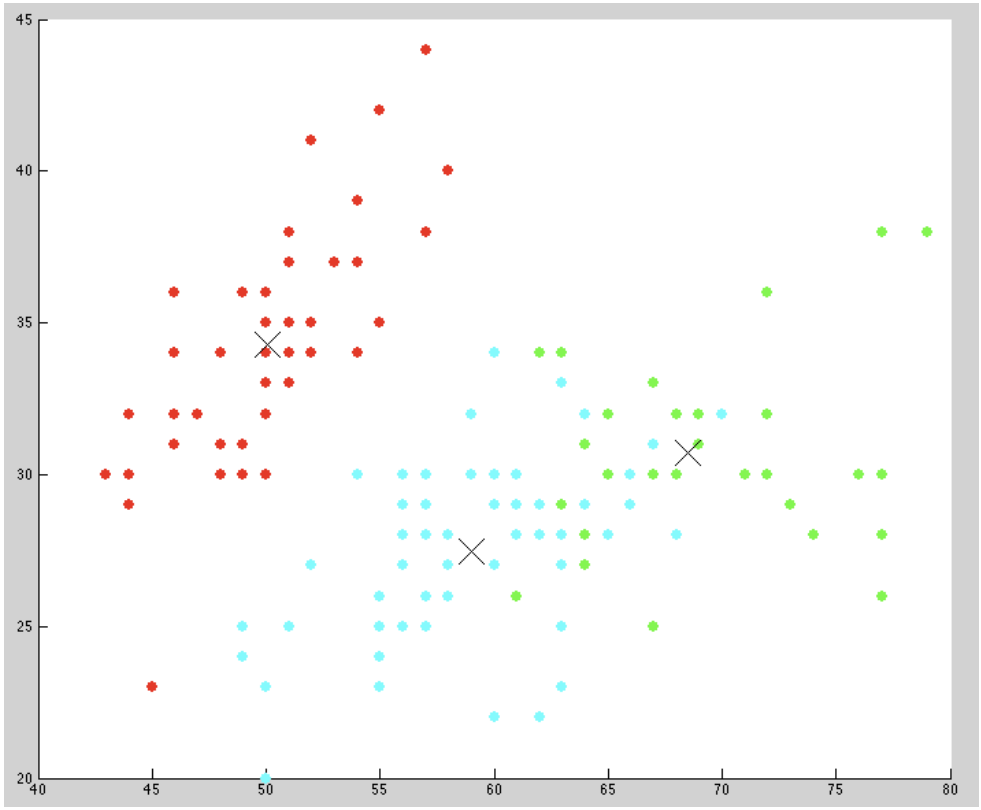
Today : unsupervised methods

Definitions
Models
Examples
Software

Tuesday: supervised methods

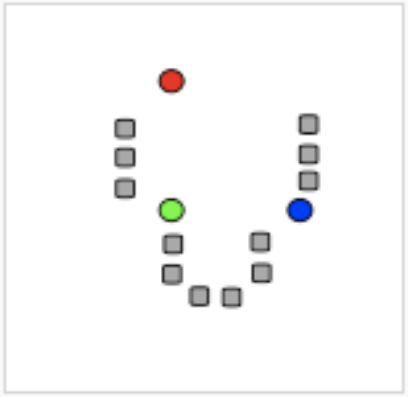
K-Means

- Unsupervised clustering method
- Partition the data into k clusters, based on their features
- Each cluster is represented by its centroid, defined as the center of the points in the cluster
- Each point is assigned to the cluster whose center is nearest
- Goal: minimize intra-cluster variance or the sum of squares of distances between data and the corresponding cluster centroid.

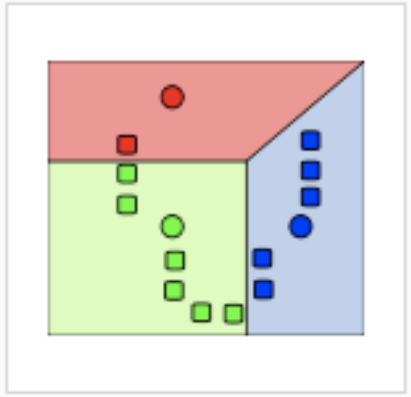


K-Means: Lloyd's Algorithm

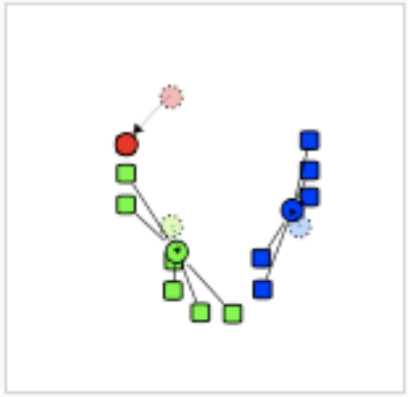
- Partition the input points into k initial random generates clusters
- Build a new partition by associating each point with the closest centroid
- Computes the mean point, or centroid, of each set
- Repeat these two steps until convergence



Shows the initial randomized point and a number of points.



Points are associated with the nearest initial randomized point.



Now the initial randomized points are moved to the center of their respective clusters(the centroids).

Steps 2 & 3 are repeated until a suitable level of convergence has been reached.

K-Means: Pro and Cons

Pro

- Simple
- Fast

Cons

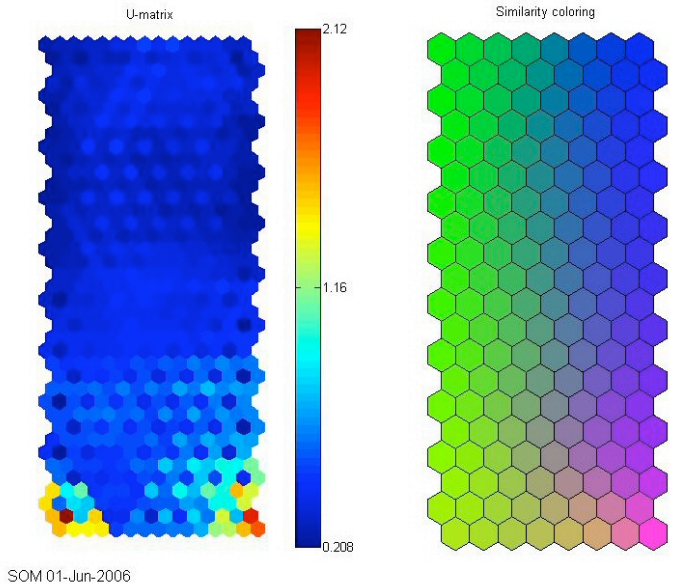
- does not yield to the same result with each run
- need to choose a priori the number of clusters
- it is not guaranteed to find the optimal configuration
  - Trick: place the first centroid on a data point, place the second centroid on a point that is far away as possible from the first one and so on...

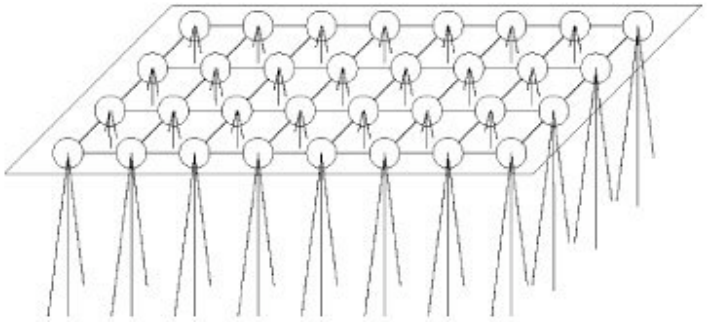
Self Organizing Maps - SOM

Self Organizing Maps learn to recognize groups of similar input vectors in such a way that neurons physically near each other in the neuron layer respond to similar input vectors.

They project high dimensional data into a low dimensional output space.

Used for clustering (or classification), data visualization, modeling, probability density estimation.

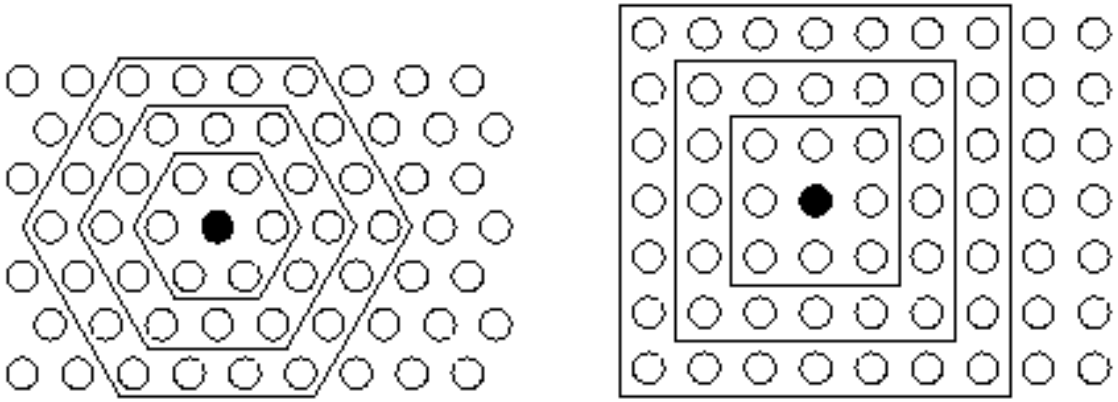




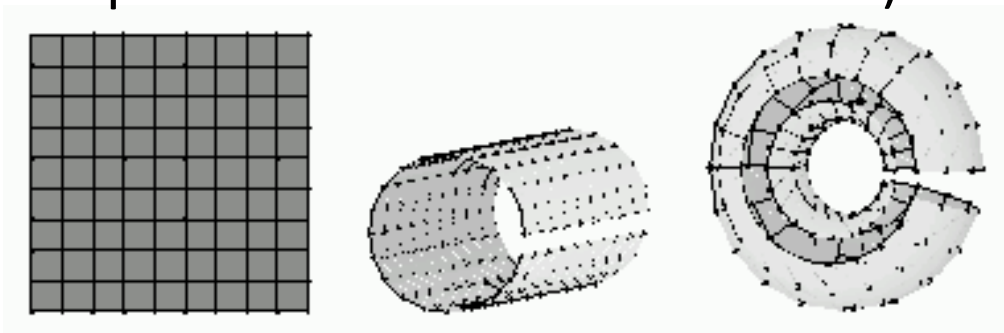
The architecture of a SOM with a two dimensional grid architecture and three inputs fed to all the neurons.

SOM – Map Grid

A SOM consists of neurons organized on a regular low dimension grid.
Higher dimensional grids are possible but harder to visualize.
Neurons can be organized on a rectangular or hexagonal lattice.



The global map shape can be rectangular, cylinder, toroid (the latter two if the sides of the map are connected to each other).

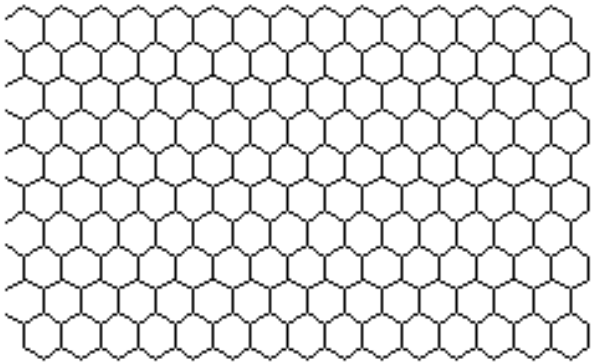


SOM – Prototypes

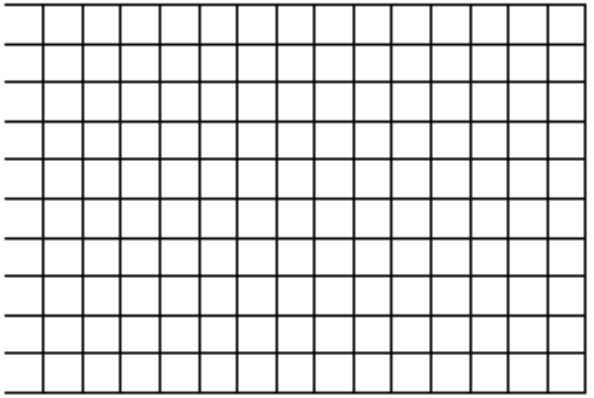
Each neuron is represented by a d-dimensional weight vector:

$$\mathbf{m}_i=[m_{i1}, \ldots , m_{id}] \quad \text{where } d=\dim(\text{input\_vector})$$

Hexagonal SOM grid



Rectangular SOM grid



Each map unit can be thought as having two sets of coordinates:
- in the input space: the prototype vectors;
- in the output space: the position on the map.

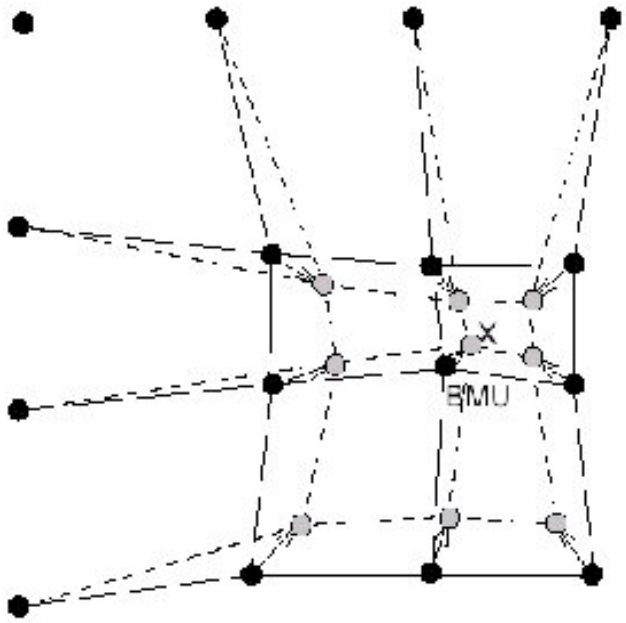
SOM - Training

In each training step, one sample **x** from the input data set is chosen;
the distances between **x** and all the weight vectors of the SOM are computed;
the neuron whose weight vector is closest to the input vector is called the **Best Matching Unit** (BMU, $\mathbf{m}_c$):

$$\|x - m_c\| = \min_i \|x - m_i\|$$

After finding the BMU, the weight vectors are updated so that the BMU is moved closer to the input vector in the input space.

Also the topological neighbors of the BMU are treated similarly.



Solid and dashed lines: situation before and after updating

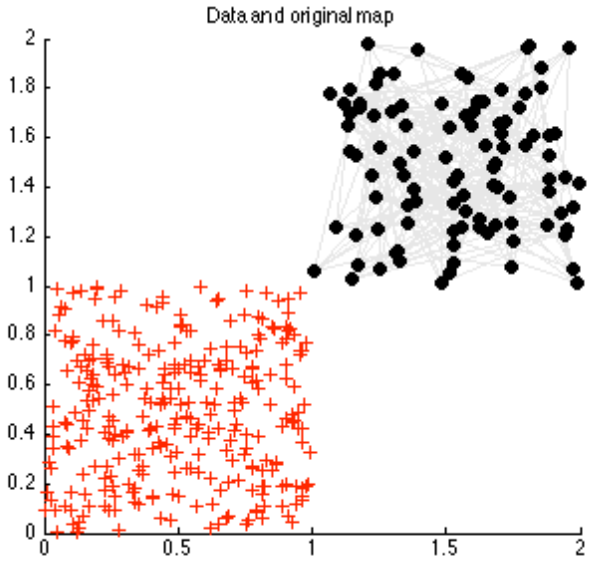
SOM: Competitive and Cooperative Train

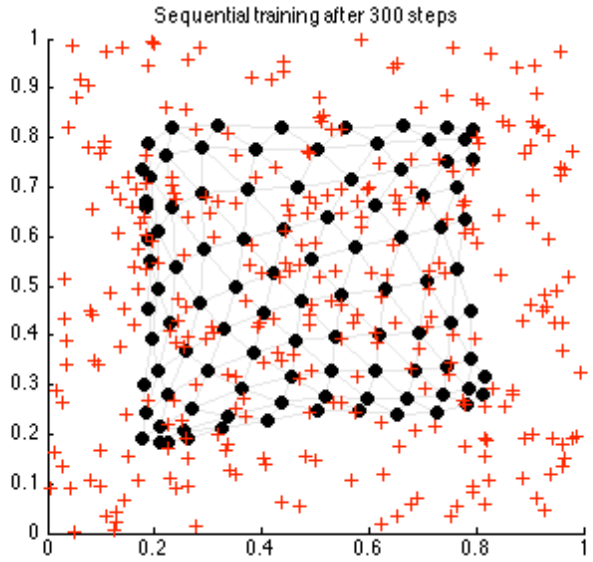
Competitive learning

The prototype vector most similar to a data vector is modified so that it is even more similar to it.

Cooperative learning

Not only the most similar prototype vector, but also its neighbors on the map are moved towards the data vector.





SOM – Update Rule

The SOM update rule for the weight vector of unit *i* is:

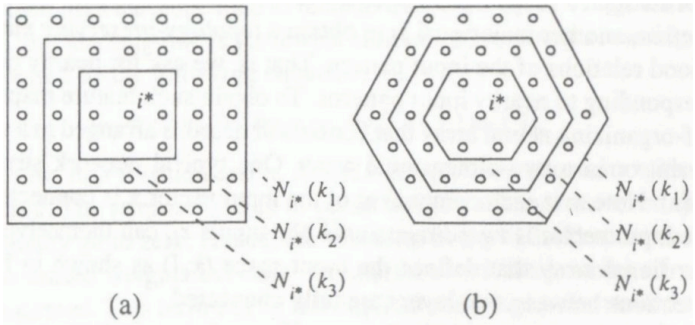
$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t)[x(t) - m_i(t)]$$

where:

- x(t) input vector chosen at time t;
- $h_{ci}(t)$ is the neighborhood function that define the kernel around the winner unit (BMU) c:

$$h_{ci}(t) = \exp(-\|r_c - r_i\|^2 / 2\sigma^2(t))$$

- σ(t) is the neighborhood radius at time t



where $k_1 < k_2 < k_3$

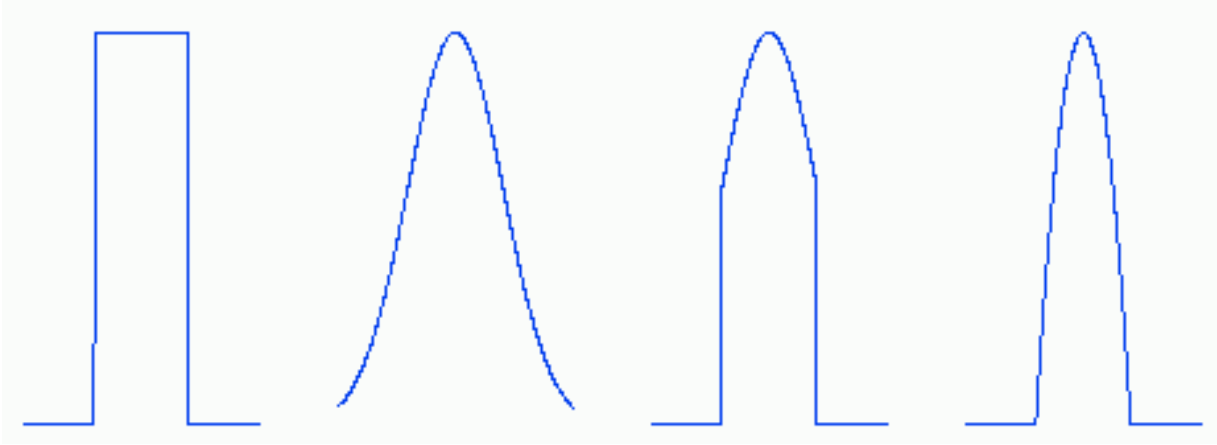
- α(t) is the learning rate at time t
  - linear .α(t)= $\alpha_0(1-t/T)$ where α_0=initial learning rate; T=training length
  - inversely proportional to time α(t)= A/(t+B) with A,B suitable constants

SOM – Neighborhood Function

The neighborhood function determines how strongly the neurons are connected to each other.

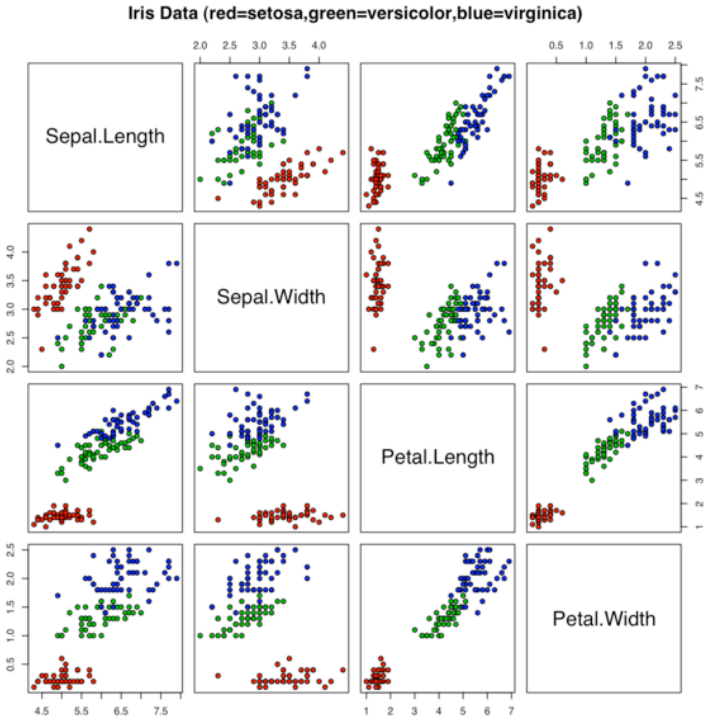
Neighborhood function and the number of neurons determine the granularity of the resulting mapping.

The larger the area where neighborhood function has high values, the more rigid is the map. The larger the map, the more flexible it can become. This interplay determines the accuracy and generalization capability of the SOM.



Datasets

- IRIS (Bi)
  - consists of 3 classes, 50 instances each and 4 numerical attributes (sepal and petal lengths and widths in cm);
  - each class refers to a type of Iris plant (Setosa, Versicolor, Verginica);
  - the first class is linearly separable from others while that latter are not linearly separable;



SOM - Preprocessing

Since SOM algorithm is based on Euclidian distances, the scale of the variables is very important in determining what the map will be like.

For this reason, the components of the data set are usually normalized.

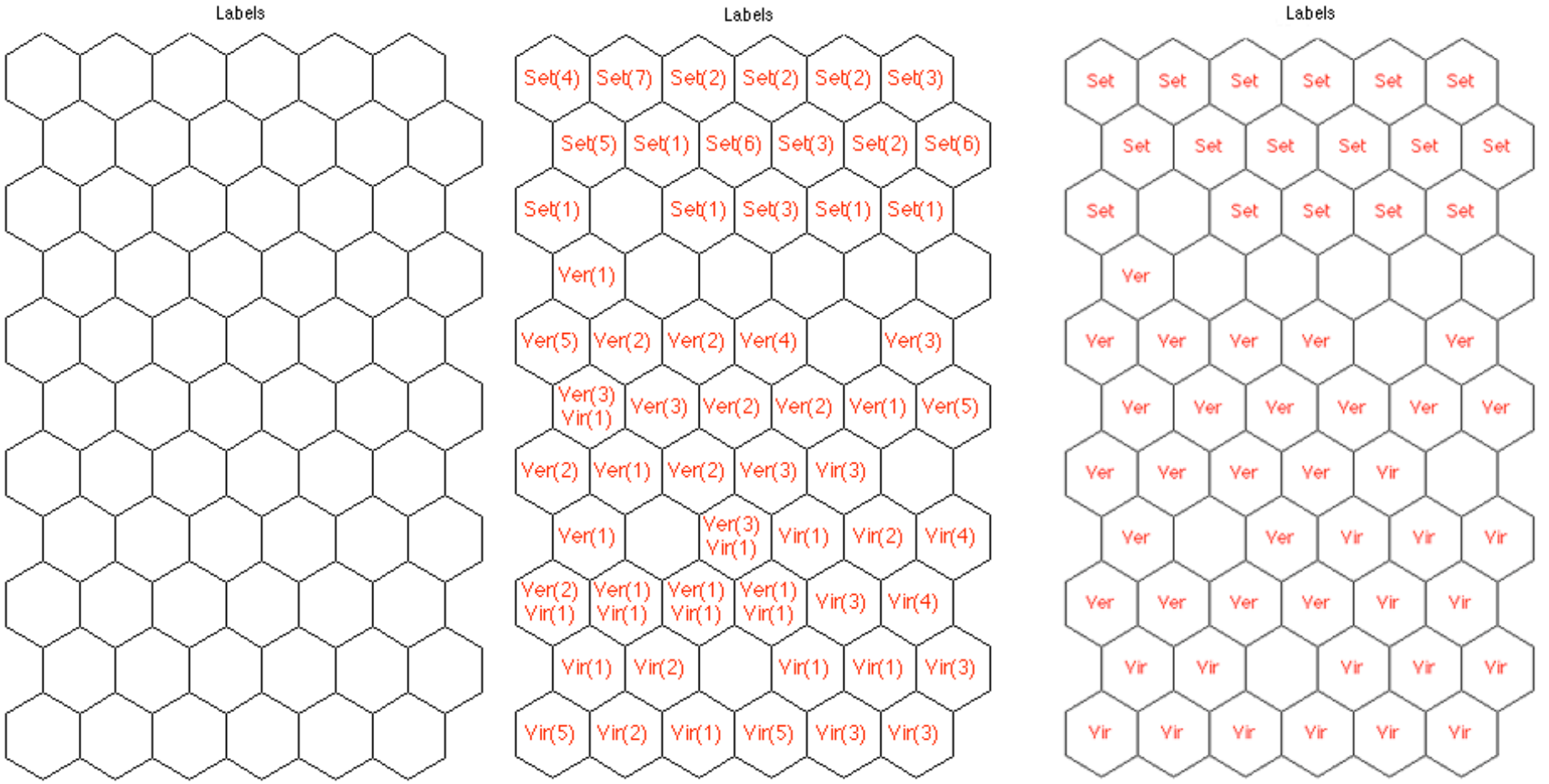
Var	$x' = \frac{x - \bar{x}}{\sigma_x}$
Range [0,1]	$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$
Log	$x' = \ln(x - \min(x) + 1)$
Softmax	$\hat{x} = \frac{x - \bar{x}}{\sigma_x} \qquad x' = \frac{1}{1 + e^{-\hat{x}}}$

SOM - Training

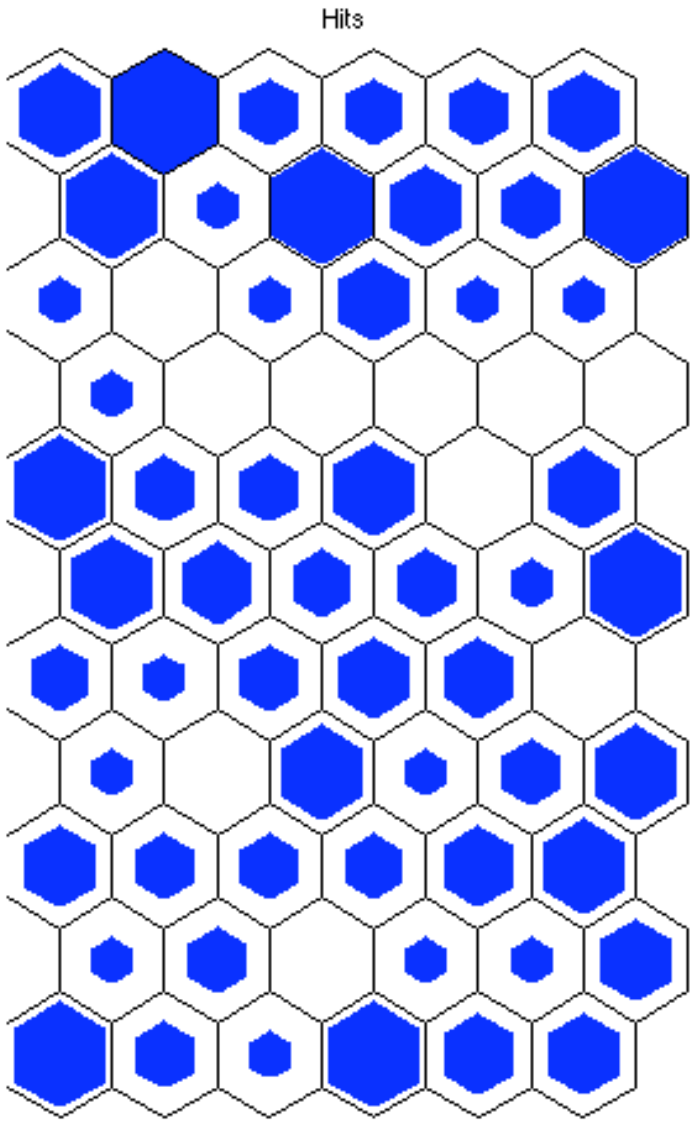
- Map size and Topology
  - Default: 5*sqrt(n) where n = # of samples
- Weights initialization
  - random: with small random values
  - sample: with random samples drawn from the input
  - linear: along the linear subspace spanned by the two principal eigenvectors of the data set
- Batch and Sequential training
  - sequential: sample are presented to the map one a time and the algorithm gradually moves the weight vectors toward them
  - batch: the data set is presented as a whole and the new weight vectors are weighted averages of the data vectors
- Learning rate, neighborhood function, radius.

SOM – Labeling

The BMU of each sample is found from the map, and the species label can be given to each map unit.



SOM – Localizing data

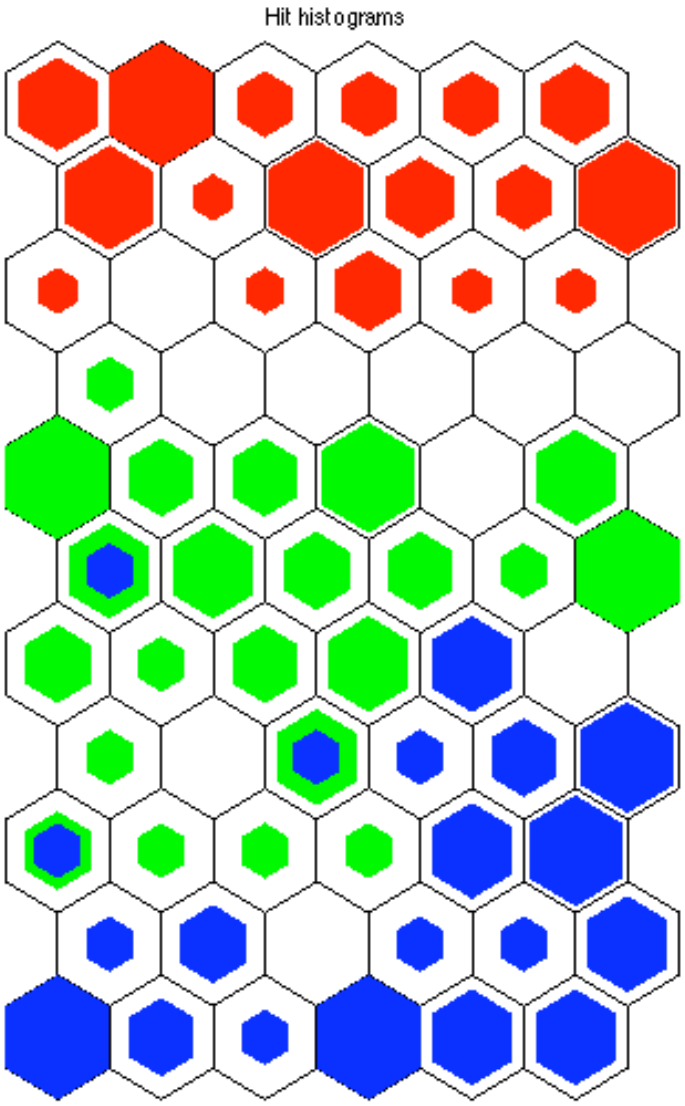


An important tool in data analysis using SOM are the so called **hit histogram**.

The hit histogram shows the distribution of the data set on the map.

They are formed by taking a data set, finding the BMU of each data sample from the map, and increasing a counter in a map unit each time it is the BMU.

SOM – Localizing data



Red color is for 'Setosa'
Green color is for 'Versicolor'
Blue color is for 'Virginica'.

One can see that the three species are pretty well separated, although 'Versicolor' and 'Virginica' are slightly mixed up.

SOM – Questions...

How to find clusters?

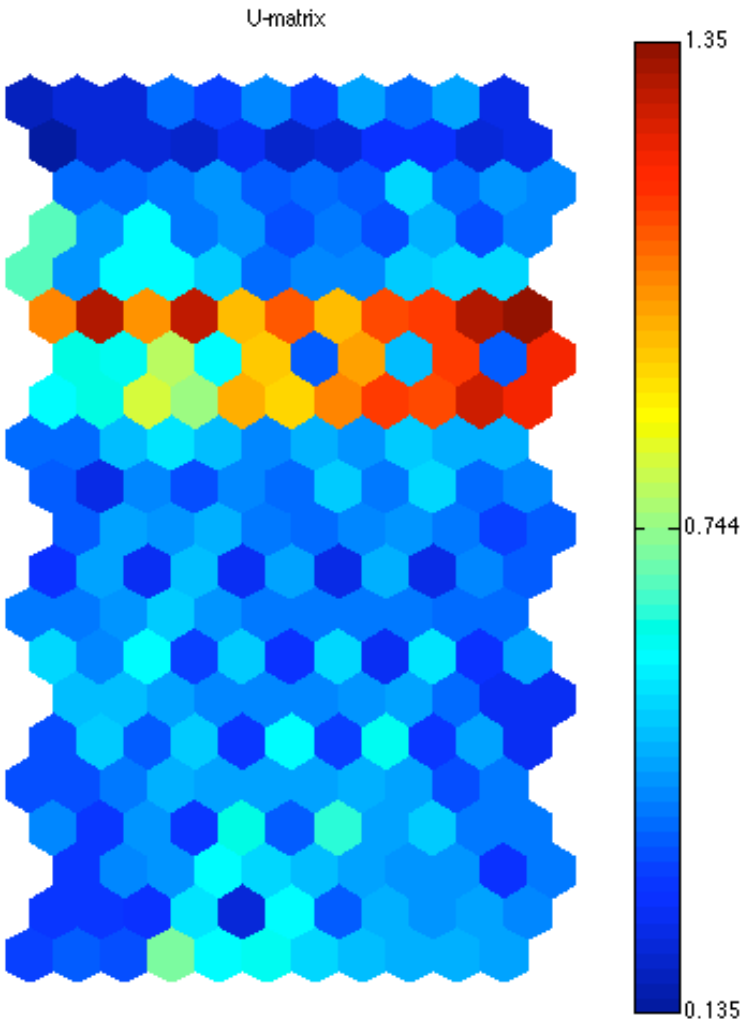
Which components are the most important in discriminating among the classes?

How do the parameters relate to the clusters?

With the SOM we have several ways to visually (and not only) answer to the these question.

"Cool" visualizations looks good in the papers too!

SOM – Cluster Structure



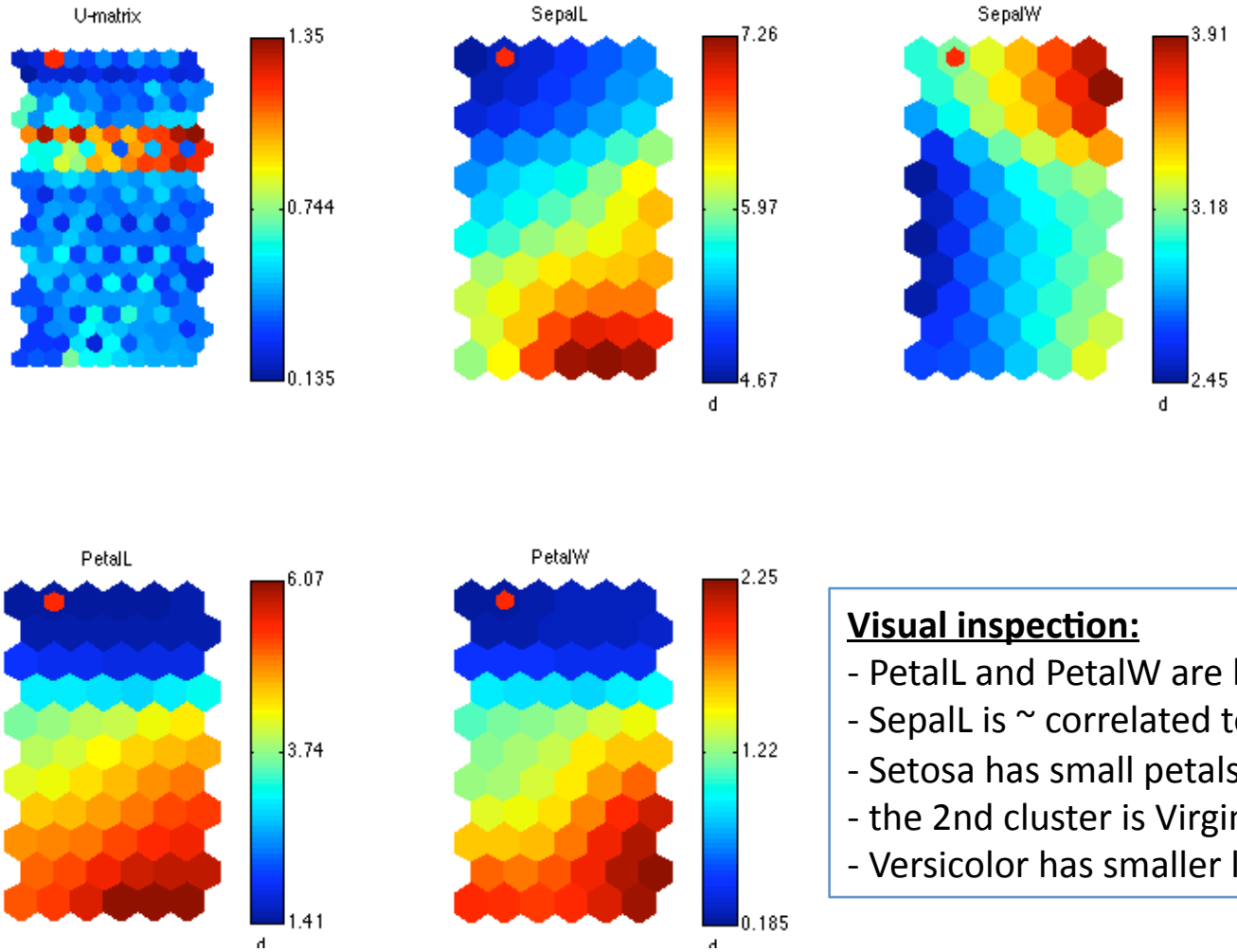
The **U-matrix** shows distances between neighboring units and thus visualizes the cluster structure of the map.

The U-matrix visualization has much more hexagons that the "real" map because distances between map units are shown, and not only the distance values at the map units.

High values on the U-matrix mean large distance between neighboring map units.

If we have five units 5x1: m(1), m(2), m(3), m(4), m(5)
U-Matrix is a 9x1 vector:
u(1), u(1,2), u(2), u(2,3), u(3), u(3,4), u(4), u(4,5), u(5)
Where:
u(i,j) is the distance between m(i) and m(j);
u(k) is the mean: u(k)=(u(k-1,k)+u(k,k+1))/2.

SOM – Component Planes

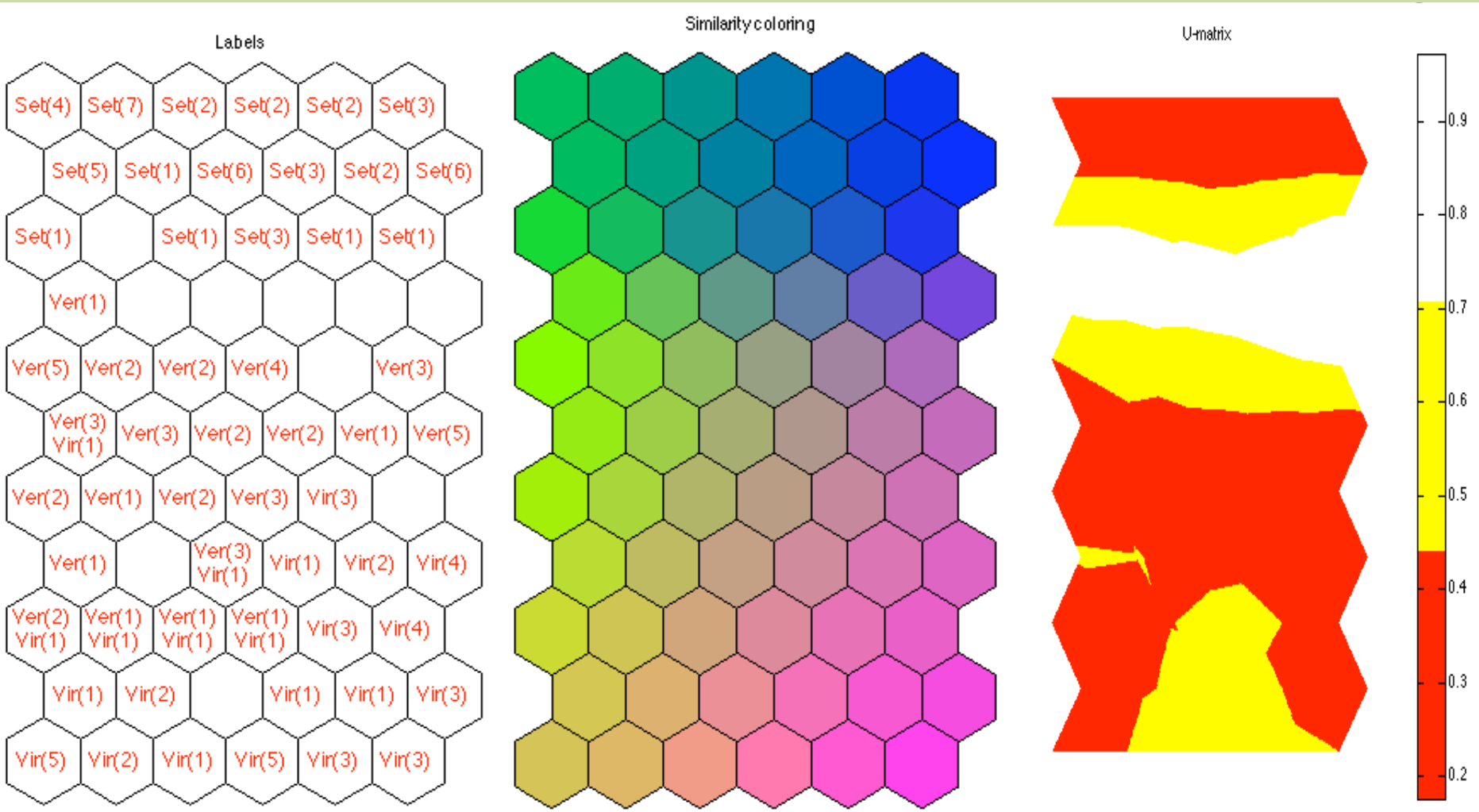


Visual inspection:

- PetalL and PetalW are highly correlated;
- SepalL is ~ correlated to PetalL and PetalW;
- Setosa has small petals and short wide sepals;
- the 2nd cluster is Virginica /Versicolor;
- Versicolor has smaller leaves than Virginica.

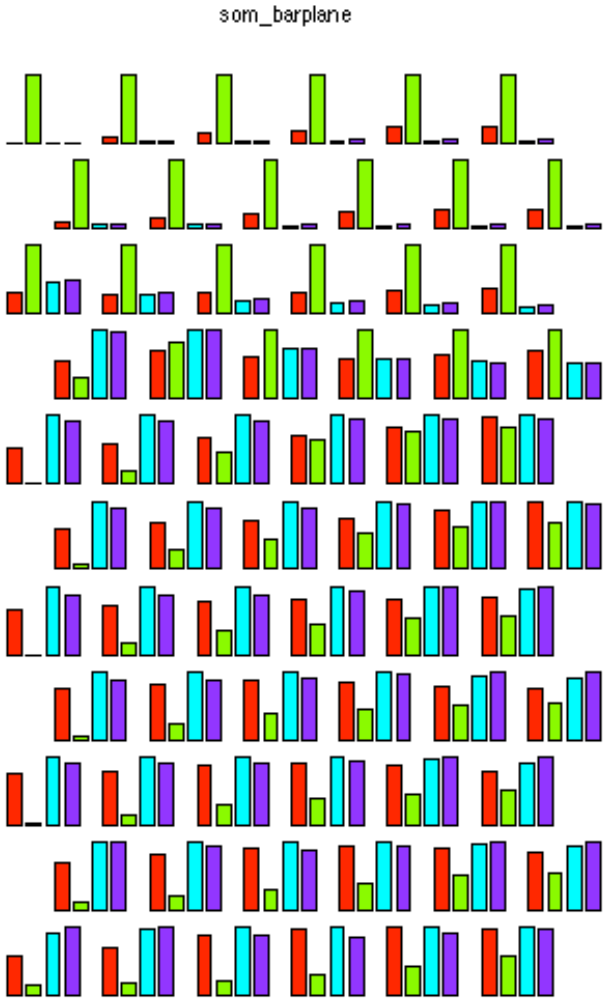
The **component planes** show the values of the prototype vectors for each parameter.
Can be used for correlation hunting.

SOM – Cluster Structure



Similarity coloring: assigns colors to the map units such that similar map units get similar colors.

SOM – Relative Importance



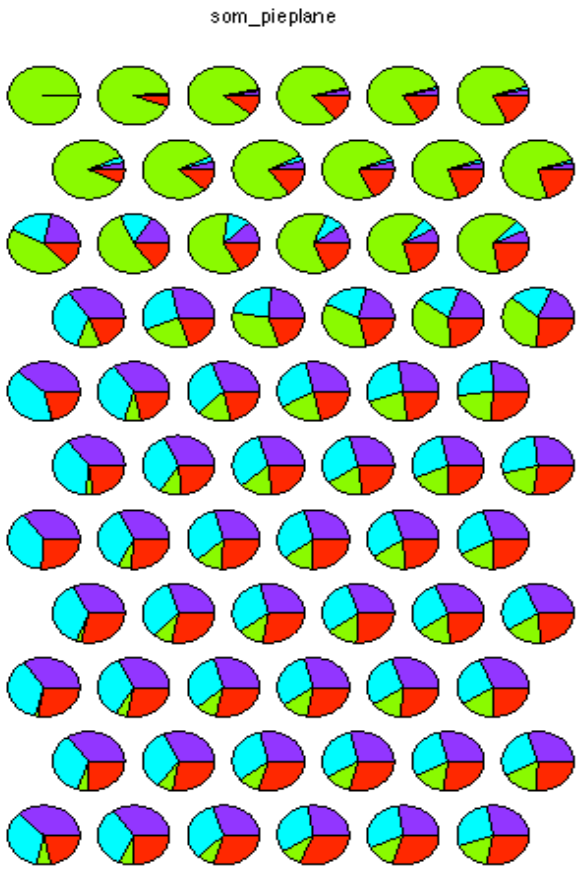
Which components are the most important in discerning among the classes?

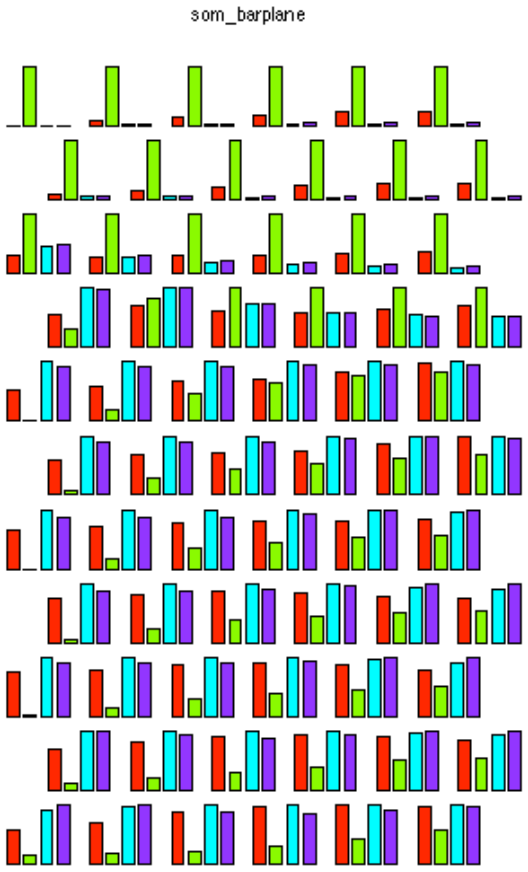
The significance of the components with respect to the clustering is usually harder to visualize.

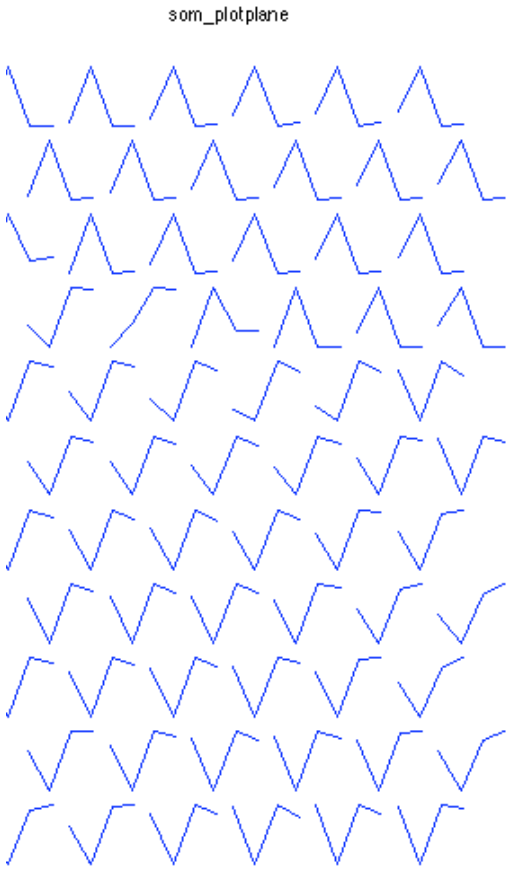
One indication of importance is that on the borders of the clusters, values of important variables change very rapidly.

Each chart shows the relative importance of each variable in each map unit.

SOM – Relative Importance





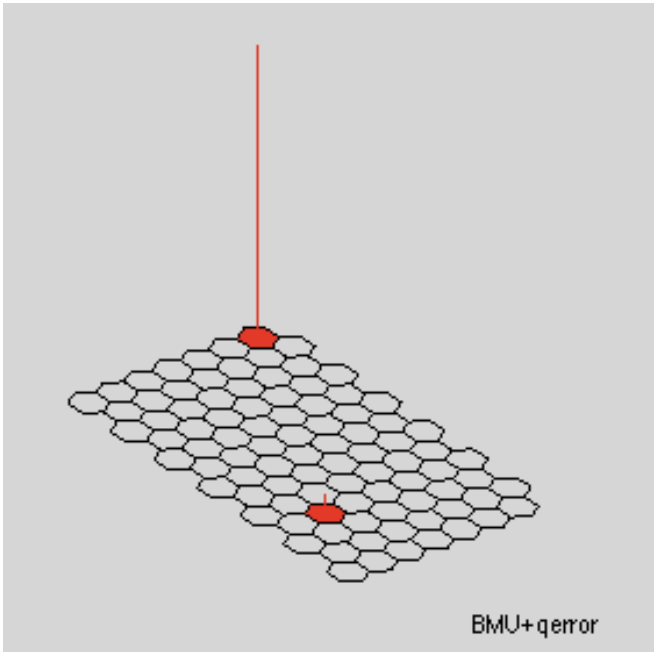


SOM – Quality of the Clustering

We have seen how to locate a sample on the map. But how accurate is that localization?

We can compute and plot how much a data sample is far from its BMU, its 2nd-BMU and WMU and so on.

Errors are also computed and are useful in determining the quality of the clustering.

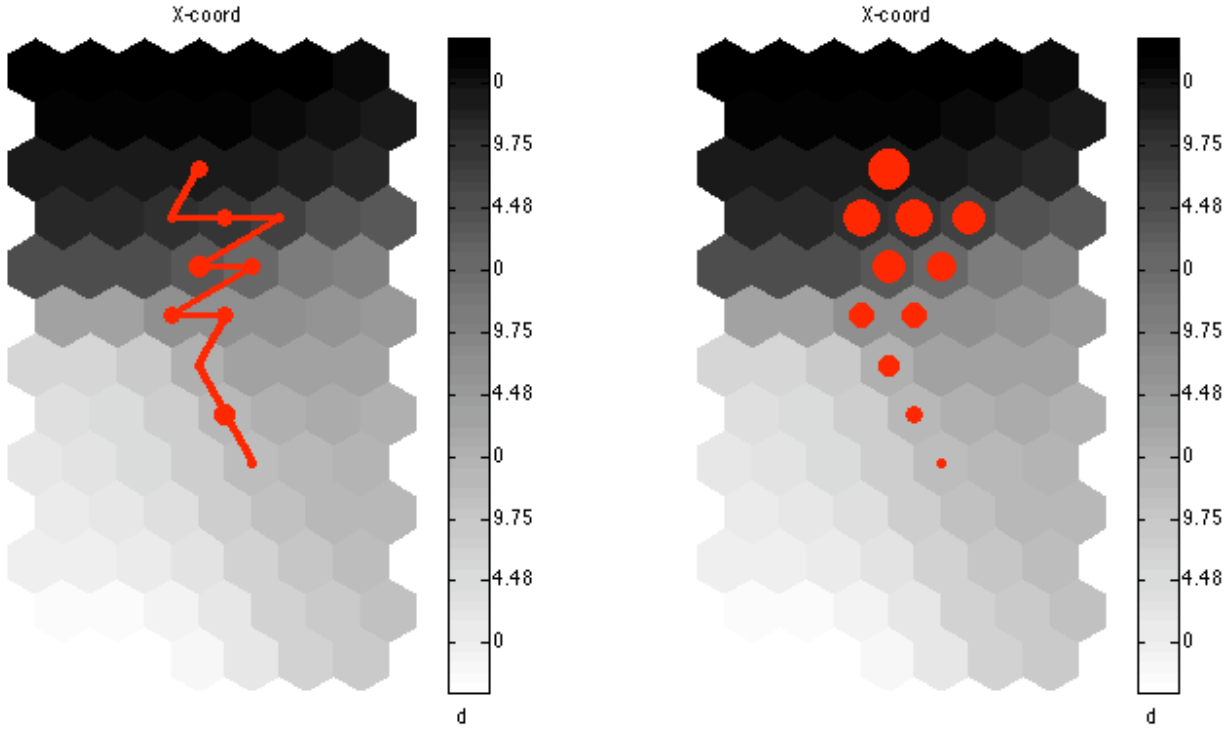


Average quantization error: measures the distance from each data vector to its BMU.

Topographic error measure: percentage of data vectors for which the BMU and the second-BMU are not adjacent units.

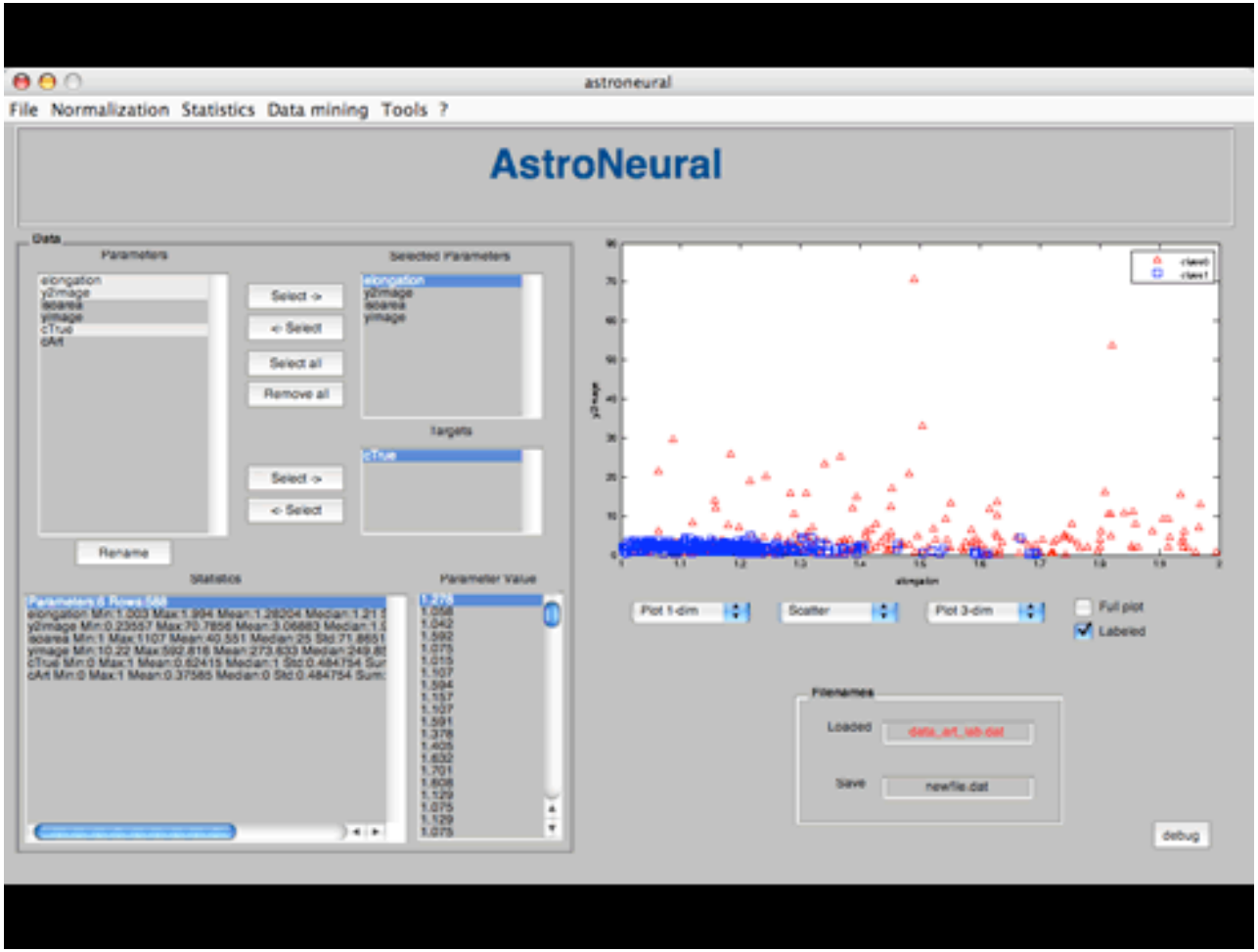
SOM – Trajectory Analysis

A special data mapping technique is trajectory. If the samples are ordered, forming a time-series for example, their response on the map can be tracked.



SOM 11-Feb-2009

Software



SOM Toolbox 2.0* http://www.cis.hut.fi/projects/somtoolbox/ (Matlab)
Astroneural Data Mining Package for Matlab (*not available for download yet*)