Classification with Kohonen Self-Organizing Maps

Mia Louise Westerlund

Soft Computing, Haskoli Islands, April 24, 2005

1 Introduction

1.1 Background

A neural network is defined in [1] as

> a set of nodes connected through directed links, where each node is a process unit that performs a static node function on its incoming signals to generate a single node output [...]. Usually a node function is a parameterized function with modifiable parameters; by changing these parameters, we [change] the node function as well as the overall behaviour of the adaptive network.

Competitive learning In unsupervised competitive learning (hard competition) a single winning neuron or node is determined each iteration. The weights are adapted so that this neuron will respond more strongly to this input. Sometimes competitive learning is implemented with a conscience, keeping track of how often the different outputs wins, so that each of the N outputs win on average 1/N times to avoid that certain elements always win the competition. [2]

1.2 Self-Organizing Maps

The Self-Organizing Map (SOM) is a fully connected single-layer linear network, where the output generally is organized in a two-dimensional arrangement of nodes, see Figure 1. The fundamental of the SOM is the soft competition between the nodes in the output layer; not only one node (the winner) but also its neighbors are updated. [8]

Self-organizing networks have the ability to learn and detect regularities and correlations in the inputs, and predict responses from input data. The neurons in a competitive network learn to recognize groups of similar input vectors while self-organizing maps (SOM) learn to recognize groups of similar input vectors in such a way that neurons physically near each other in the neuron layer respond

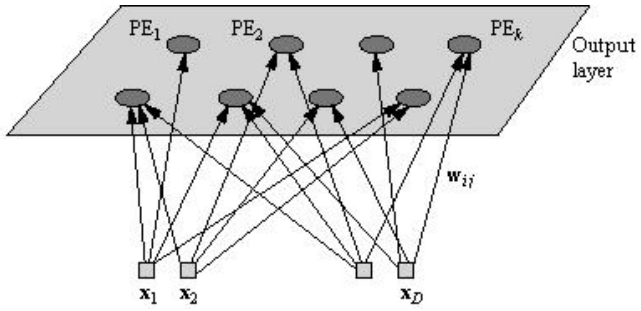


Figure 1: The SOM architecture (The PE:s or processing elements are referred to as nodes in the text). [8]

to similar input vectors. Self-organizing maps learn both the distribution and topology of the input vectors they are trained on; while the winning neuron is determined similar for competitive layers and SOM, instead of updating only one node SOM updates a neighborhood of nodes around the winning node. [2]

The SOM defines a mapping from the input data space R^n onto a regular two-dimensional array of nodes (map grid). A parametric reference vector $m_i \in R^n$ is associated with every node i in the map. The array of nodes can be projected onto a rectangular or a hexagonal lattice. Every input vector x is compared with the m_i:s and the best match is noted, and the input is mapped onto that location. The SOM can be looked at as a "'nonlinear projection' of the probability density function of [a] high-dimensional input data onto the two-dimensional display". Every input vector $x \in R^n$ can be compared with the m_i:s in any metric, usually the Euclidean distance. The winning node c is then calculated by

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

or

$$c = arg \min_i \{|| x - m_i ||\}$$

and x is mapped onto c relative to the parameter values m_i. [7]

Nodes topographically close to another in the array will learn from the same input. The update formula can be written

$$m_i(t+1) = m_i(t) + h_{c,i}(t)[x(t) - m_i(t)],$$

where t is the discrete-time coordinate and $h_{c,i}$ is the function defining the neighborhood. The initial values of the m_i:s can be random. The neighborhood function is defined over the lattice points or the map and in a way defines the stiffness or elasticity to be fitted to the data points. [7]

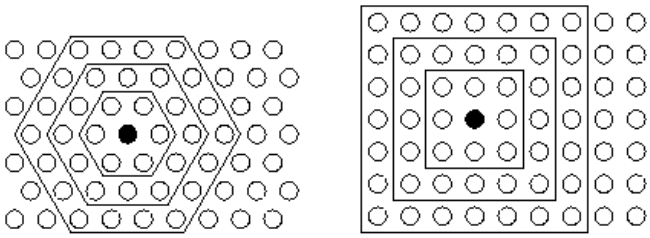
2 Application

2.1 Software

The SOM Toobox, available at [5], is an implementation of the SOM algorithm for Matlab. The software library is free under the terms of the GNU General Public License. The toolbox allows you, among other things, to train SOMs with different network topologies and learning parameters, visualize SOMs using U-matrices and component planes, cluster color-coding etc.

2.2 SOM Structure

Map Grid Usually the input is mapped onto a 1- or 2-dimensional map. Mapping onto higher dimensions is possible as well, but complicates the visualization. In the SOM Toolbox the grid size (number of neurons) can be decided manually or extracted from the input data automatically. The neurons connected to adjacent neurons by a neighborhood relationship define the structure of the map. The two most common 2-dimensional grids are the hexagonal grid and the rectangular grid, see figure 2. Other common (3-dimensional) map shapes are the cylinder and toroid shapes. [6]



(a) Hexagonal grid (b) Rectangular grid

Figure 2: Typical grids.

Neighborhood function The neighborhood defines the correlation between neurons. The simplest neighborhood function is called *bubble*; it is constant over the neighborhood of the winner neuron and zero otherwise. The neighborhood of different sizes in rectangular and hexagonal maps can be seen in figure 2. A more flexible definition is the *gaussian neighborhood function*, $e^{-\frac{||r_c - r_i||^2}{2\sigma^2(t)}}$, where r_c is the location of unit c on the map grid, and $\sigma(t)$ is the neighborhood radius at time t. The neighborhood decreases over time. [6]

2.3 Algorithm

Size and shape The number of neurons, the dimensions of the map grid, the map lattice and shape must be specified. The more neurons we have, the more flexible the mapping becomes but the computational complexity of the training phase increases as well. In the SOM Toolbox the number of neurons is by default $5\sqrt{n}$, where n is the number of training samples. The choice of the map structure and size is both related to the type of problem and the subjective choice of the user. The SOM Toolbox determine the grid side lengths by the ratio between eigenvalues of training data, and the default structure is hexagonal. [6]

Initialization The weight vectors can be initialized in many ways, for example random initialization, sample initialization (initial weight vectors are taken from random samples in the input), linear initialization (spanned by eigenvectors of the input data set). In the SOM Toolbox random and linear initialization is included. The random values are between the minimum and maximum values of each variable. [6]

Training In every training step a sample vector x is chosen from the input data set randomly and the similarity is calculated between it and all weight vectors in the map. The vector that is most similar is called the Best-Matching Unit (BMU). Euclidian distance is normally used as the similarity metric. The weight vectors in the map in the neighborhood of the BMU are updated so that they move closer to the sample vector, see figure 3. [6]

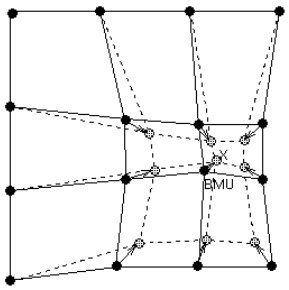


Figure 3: SOM update

Training parameters The learning rate is a decreasing function in the interval $[0, 1]$. The learning rate can be linearly decreasing or defined by some non-linear function. The neighborhood radius decreases with time as well. Large neighborhoods that make the SOM more rigid are used in the beginning of training, and is subsequently decreased during training to fine-tune the SOM.

If the neighborhood i zero, then the function of the of the SOM is the same as for the k-means algorithm. The training time should be at least 10 times the number of map units. In the SOM Toolbox training is done in two steps. In the first step the learning rate starts from 0.5 and the neighborhood radius decreases from $\frac{max(msize)}{4}$, where $msize$ is the size of the map, to one forth of that value (stopping at 1). In the second step the learning rate starts from 0.05 and the neighborhood radius starts from where it stopped in the first step. [6]

Batch training The batch algorithm goes through the whole training set once, and only update the weights after that with the net effect of all samples. The weight vectors in the map are replaced by a prototype vector with a weighted average over the samples, where the weighting factors are the neighborhood function values. The updating function for the weight vectors in the map thus becomes $m_i(t+1) = \frac{\sum_{j=1}^{n} h_{i,c(j)}(t) x_j}{\sum_{j=1}^{n} h_{i,c(j)}(t)}$, where m is the map weight vector, $c(j)$ is the BMU of sample vector x_j, $h_{i,c(j)}$ the neighborhood function and n is the number of samples. Batch training is the default method of training in the SOM Toolbox. [6]

3 The data set

The data, [3], used to test SOM performance for classification is the result of a chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. The data set contains the analysis of twelve different components in the wines. The different attributes measured are Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue and OD280/OD315 of diluted wines. In addition, each sample has a class identifier (which naturally is not used by the SOM method for classification). The data set consists of 178 samples; 59 samples from class 1, 71 samples from class 2, and 48 samples from class 3. Basically, what we want to do is to be able to differ between the three types of wine by measuring similarities and dissimilarities between the 12 attributes extracted from each wine.

4 Testing

A SOM is created and trained with the tools of the SOM Toolbox in Matlab on the Wine recognition data and several methods are used for visualization of the results.

4.1 Preprocessing

Scaling of the input variables is essential in the SOM Toolbox, since the distance between vectors is measured with Euclidian metric. If the data is not normalized and the range of the variables vary, the variables with the largest ranges will dominate the map organization because they have stronger impact on the measured distances. In most cases we want all variables to be of equal importance. The easiest way to normalize the input is to linearly scale the variables so that their variances are equal to one. Other methods in the Toolbox are histogram equalization and logarithmic scaling.

4.2 Creating and training a SOM

Initialization of the SOM is done either randomly or linearly (default). Training can be performed either in sequential or batch (default) mode. Training is done in two phases. First a rough training with a large neighborhood and a large learning rate is performed and after that, a fine-tuning with a small neighborhood and a small learning rate.

4.3 Visualization

The Toolbox includes tools for displaying the U-matrix and the component planes of the SOM. The U-matrix shows the distances between neighboring map units. High values indicate large distances and thus the border between classes, while uniform areas of low distances indicate the clusters. A component plane shows the values in each map unit for one variable (default). It is also possible to combine several variables into one component plane.

5 Results

U-matrix The SOM is trained in two phases, and the results can be seen in Figure 4. Clearly, there are three different clusters with borders in between, which indicate that there are measurable dissimilarity between the three types of wines. The U-matrix is a way of showing similarity between the units. The simplest way to calculate a U-matrix is to calculate the average of distances between each node and its neighbors.

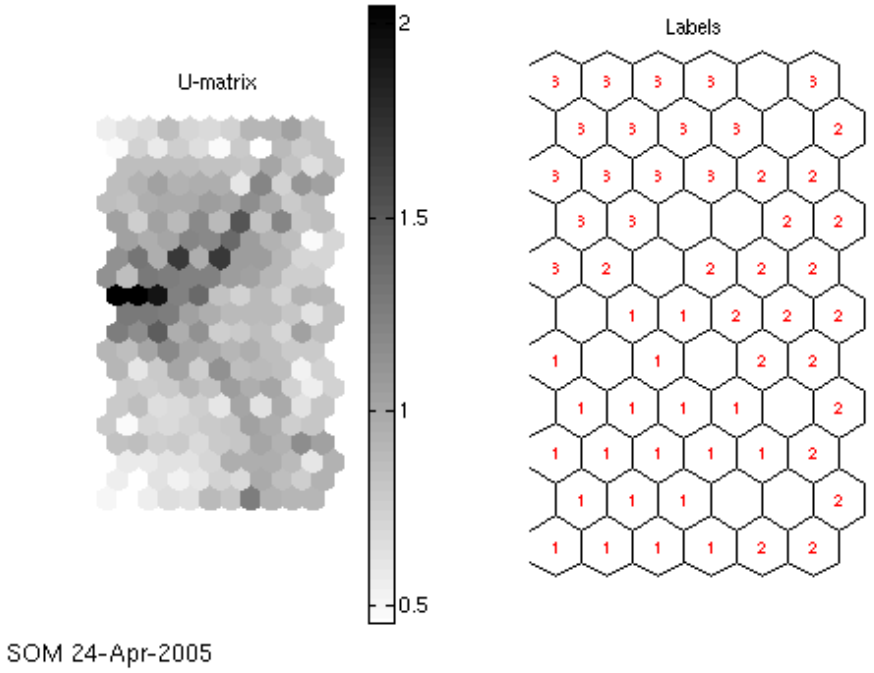


Figure 4: The U-matrix and labels showing the distribution on the hexagonal grid topology; light shades mean large similarity whereas dark shades indicate dissimilarity and cluster borders. [4]

Component planes Besides looking at the overall differences in the U-matrix it is interesting as well to look at the differences between each component present in the input vectors, meaning that we look at differences regarding just one of the input "variables" at a time. From this can be drawn conclusions about which components are the most significant for the classification and is a very useful tool for graphically getting a picture of the data that are dealt with. The separate component planes for the wine data are shown in Figure 5.

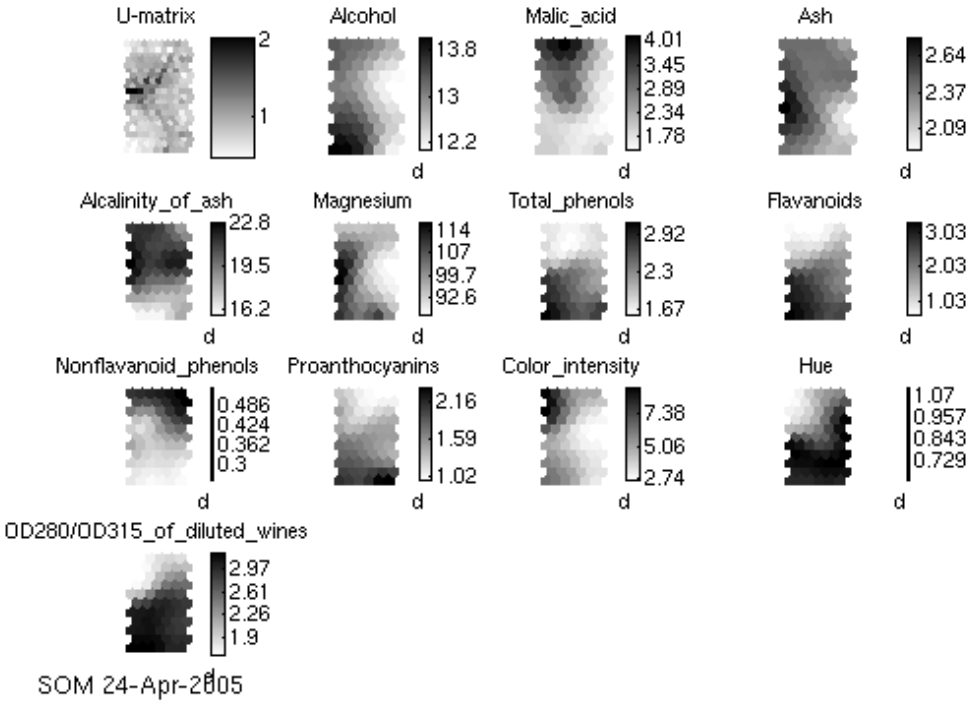


Figure 5: The U-matrix shown together with the different component planes.

Hit histogram Another interesting visualization way is to plot a hit histogram, see figure 6. The BMU of each data sample is calculated, and a counter increases in the map unit in question each time it is the BMU. The hit histogram plotted on the U-matrix provides a good way to view the distribution of the data se on the map.

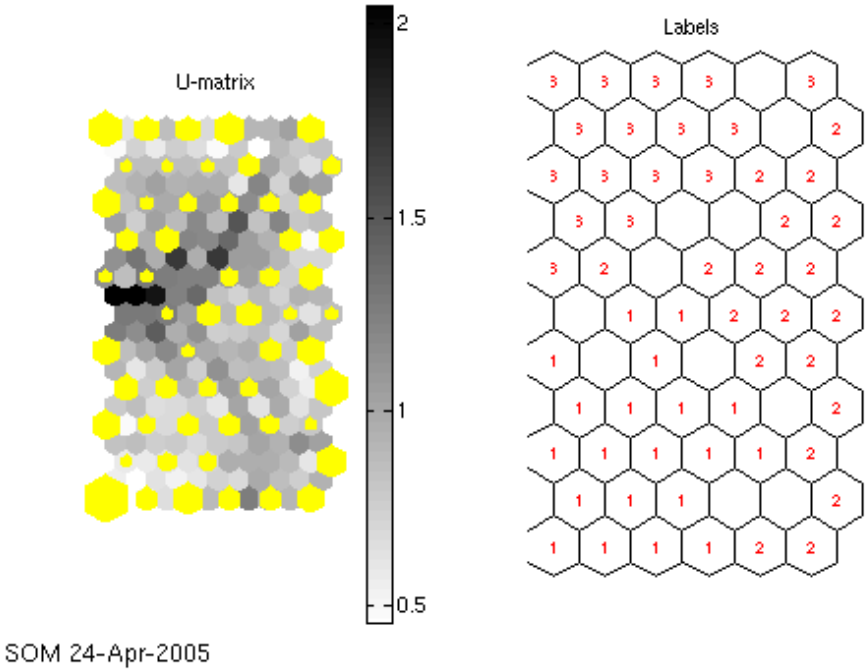


Figure 6: The BMU of each data sample plotted on the U-matrix.

Multiple histograms Multiple histograms can also be shown simultaneously, counting hits for each of the classes, see figure 7.

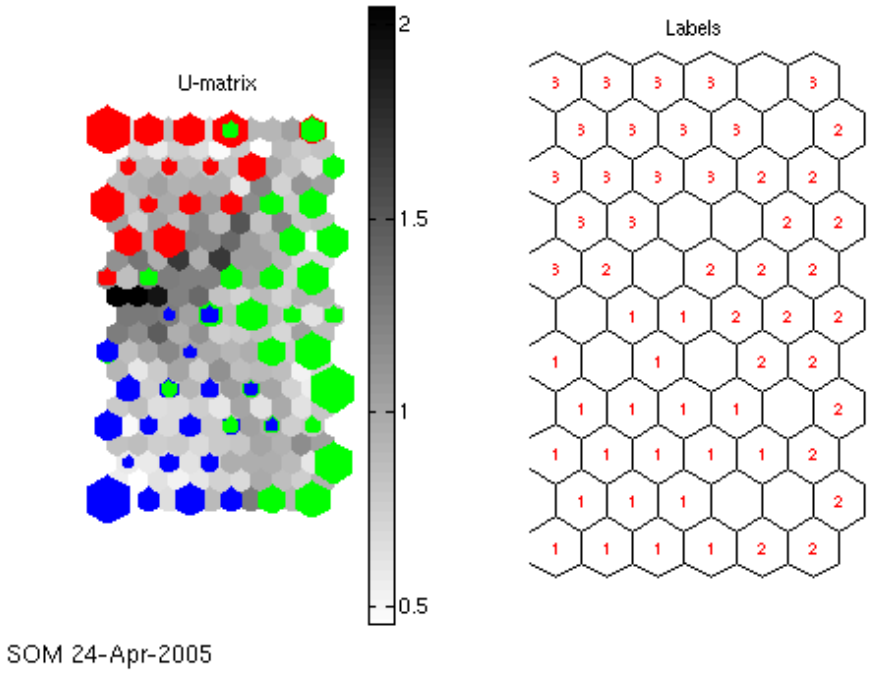


Figure 7: Hits counted separately for each class.

Surface plot The result can also be viewed with a surface plot of the distance matrix of samples, indicating average distance to neighboring map units, see figure 8.

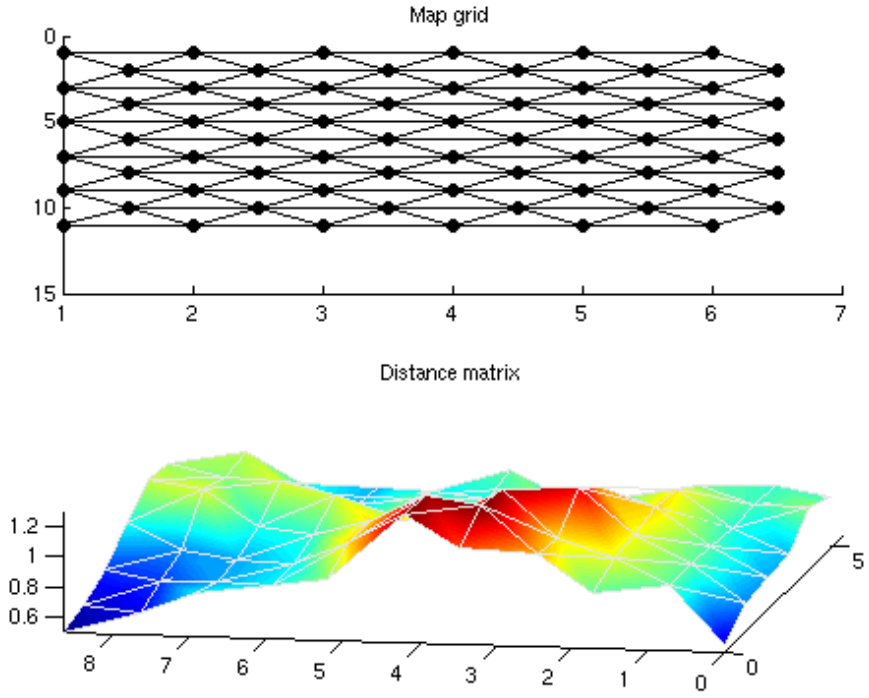


Figure 8: The map grid and the surface plot showing the average distance to neighboring map units.

Quality The quality of the map can be measured in different ways. By default the SOM Toolbox in Matlab computes two measures of the quality, quantization error and topological error. Both are dependent on the data. The quantization error is the average distance between each data vector and its BMU and thus measures the map resolution. The topographic error is the proportion of all data vectors for which first and second BMUs are not adjacent units and thus measures the topology preservation. Both these measures give better results when the map is over fitted to the data, and especially if the number of map units is larger than the number of training samples it is not desirable to have these errors zero.

For the data set tested here the quantization error is 1.808 and the topographic error 0.011. The map consists of 11x6 or 66 units in a hexagonal grid using gaussian me. Changing the grid to rectangular but keeping the same number of map units decrease the quantization error and increase the topographic error, but does not imply much changes. Changing the number of map units does however affect the classification and measurements of quality. Varying the membership function between the four different functions provided by the SOM Toolbox does not affect the mapping much either.

6 Conclusion

The SOM is a representation of the result of a vector quantization algorithm placing a number of reference vectors into a high-dimensional input data space, to approximate to its data sets in an ordered fashion. A mapping from a high-dimensional data space R^n onto a two-dimensional lattice of points is thus defined and can effectively be used to visualize metric ordering relations of input samples. This SOM algorithm is not intended for optimal classification of data, but mainly for their interactive monitoring. The creation of the SOM is an unsupervised learning process and can be used for finding clusters in the input data and identifying unknown input vectors, but if the data are known to belong to a finite number of classes, supervised methods can be more effective. An supervised method is for example learning vector quantization (LVQ). [7]

References

[1] Jyh-Shing Roger Jang, Chuen-Tsai Sun
Neuro-Fuzzy Modeling and Control
IEEE 1995 (Log Number 9408301)

[2] Mathworks,
Matlab Neural Networks Toolbox Tutorial

[3] UCI Machine Learning Repository,
Wine recognition data,
ftp://ftp.ics.uci.edu/pub/machine-learning-databases/wine/,
Apr 16th, 2005.

[4] National Center for Advanced Secure Systems Research (NCASSR)
A Simple Spectrum Scanner
http://www.ncassr.org/projects/sdr/som-tool/
Apr 16th, 2005.

[5] Esa Alhoniemi, Johan Himberg, Juha Parhankangas and Juha Vesanto
Helsinki University of Technology, Laboratory of Computer and Information Science, Neural Networks Research Centre
SOM Toolbox
http://www.cis.hut.fi/projects/somtoolbox/

[6] SOM implementation in SOM Toolbox, Laboratory of Computer and Information Science, Neural Networks Research Centre
http://www.cis.hut.fi/projects/somtoolbox/documentation/somalg.shtml,
Apr 16th, 2005

[7] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, Jorma Laaksonen
The Self-Organizing Map Program Package, Helsinki University of Technology, Laboratory of Computer and Information Science
http://www.cis.hut.fi/research/som-pak/
Apr 16th, 2005

[8] Principe, Euliano, and Lefebvre
NeuroDimension Inc, NeuroSolutions
Interactive Book, Neural and Adaptive Systems: Fundamentals Through Simulations