UNIVERSIDADE NOVA DE LISBOA

INSTITUTO SUPERIOR DE ESTATÍSTICA E GESTÃO DE INFORMAÇÃO

Introduction to Kohonen's Self-Organizing Maps

Fernando Bação and Victor Lobo

CONTENTS

1 - Introduction

This text is meant as a tutorial on Kohonen's Self-Organizing Maps (SOM). The basic idea is to provide an overview of this valuable tool, allowing the students to understand the basic principles of its workings. Clearly, this objective can only be achieved if the student background is considered, for this reason we will try to easy the technical aspects through the use of simple working examples. Emphasis will be put on practical work, hopefully motivating the student to continue on his own the study of the SOM. We shall start with a short introduction to SOM, explaining what it is. We will attempt to give an intuitive approach to its underlying principles, and proceed to a more formal definition of SOM.

We would like to emphasize the importance of the accompanying materials in order to get a "complete" picture of the SOM and its potential within GISc. This way we encourage the students to read the papers suggested in the web page of the course. Additionally, the demos can be valuable elements in order to develop an intuitive understanding of the SOM workings.

2 – The Origins of the Self-Organizing Map

Although the term "Self-Organizing Map" could be applied to a number of different approaches, we shall always use it as a synonym of Kohonen's Self Organizing Map, or SOM for short. These maps are also referred to as "Kohonen Neural Networks" [Fu 94], "Self-organizing Feature Maps-SOFM", or "Topology preserving feature maps" [Kohonen 95], or some variant of these names.

Professor Kohonen worked on auto-associative memory during the 70's and early 80's, and presented his self-organizing map algorithm in 1982 [Kohonen 82]. However, it was not until the publication of the second edition of his book "Self-

Organization and Associative Memory" in 1988 [Kohonen 88a], and his paper named "The Neural Phonetic Typewriter" on IEEE Computer [Kohonen 88b] that his work became widely known. Since then there have been many excellent papers and books on SOM, but his 1995 and 2001 books [Kohonen 95, 01] are generally regarded as the major references on the subject. The books have had very flattering reviews, presenting a thorough covering of the mathematical background for SOM; its physiological interpretation; the basic SOM; and recent developments and applications.

Professor Kohonen's group maintains a very good web-site at Helsinki's Technical University at "**http://www.cis.hut.fi/research/**". That site contains public domain software, various manuals, papers, technical reports, and a very thorough and searchable list of papers dealing with SOM and LVQ. We strongly recommend a visit to that site to anyone that intends to work on or with SOM.

Kohonen himself describes SOM as a "visualization and analysis tool for high dimensional data". These are indeed two of the most attractive characteristics of the SOM, but it can also be used for clustering, dimensionality reduction, classification, sampling, vector quantization, and data-mining. Here we will focus on the use of the SOM as an exploratory analysis tool, emphasizing the aspects related with clustering, visualization and the analysis of high dimensional data. Our perspective is that the SOM can be viewed as a toolbox rather than a tool, as it contains numerous features that can be of interest in different situations and for different tasks.

3 - Basic principles

The SOM encompasses a number of characteristics which bear similarities to the way the human brain works, or at least is thought to work. In fact, the notion of having available a set of neurons which through learning experiences specialize in the identification of certain types of patterns is consistent with current research on the human brain. The idea that certain parts of the brain are responsible for specific skills

and tasks is remarkably similar to the SOM principles. The concept of organizing information spatially, where similar concepts are mapped to adjacent areas, constitutes a trademark of the SOM and it is believed to be one of the paradigms of the functioning the human brain.

The basic idea of the SOM is to make available a certain amount of classificatory resources (we will call them neurons, although the term units is also widely used) which are organized based on the patterns available for classification (which will be called here input patterns). A very relevant characteristic of the SOM, which in fact constitutes a trademark, is the development of an ordered network in which nearby neurons will share similarities, this way patterns which are similar will activate similar areas in the SOM. Thus it can be said that different parts of the SOM will be activated by specific types of input patterns, leading to a representation based upon global organization and local specialization. The SOM is trained iteratively through a large number of epochs. An epoch is defined as the processing of all the input patterns once, thus each input pattern will be processed as many times as the number of epochs.

A SOM is single layer neural network, where the neurons are set along an *n*-dimensional grid. In most applications this grid is 2-dimensional and rectangular, though many applications use hexagonal grids, and 1, 3, or more dimensional spaces. In this grid we can define neighborhoods in what we call the output space, as opposed to the input space of the data patterns. The concepts of input space and output space are rather important as they constitute the center of the SOM's activity. In fact, the can be though of as a tool which maps (projects) the vectors in the input space onto the output space trying to preserve the topological relations observed in the input space. In Figure 1 an example of both spaces is presented. On the right side of the figure we have the input space, where smiles represent input patterns and stars represent neurons; in this particular case, and for purposes of visualization, each vector is define solely based on two coordinates. On the left side of the picture a representation of the output space is shown, here the neighborhood connections are established.

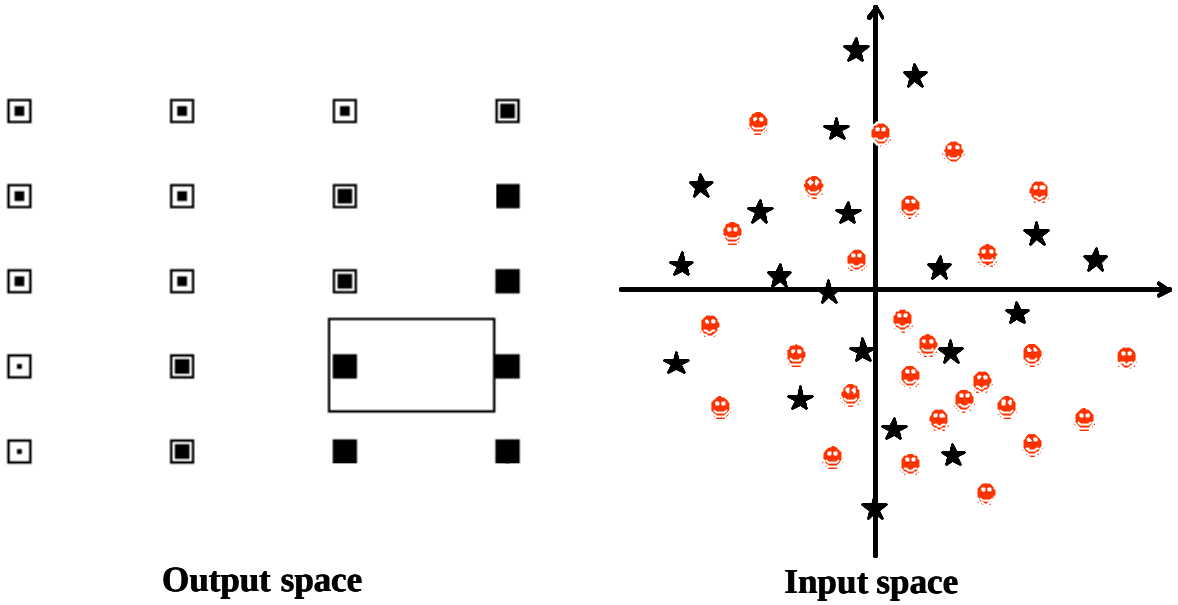


Figure 1 - Basic SOM architecture, on the left side the output space is presented, on the right side an example of an input space, in this case each vector comprises only two components

Each neuron, being an input layer neuron, has as many weights (which are often also called coefficients) as the input patterns, and can thus be regarded as **a vector in the same space as the patterns**. As shown in Figure 2 smiles represent input patterns and stars represent the neurons in this 2-dimensional space already introduced.

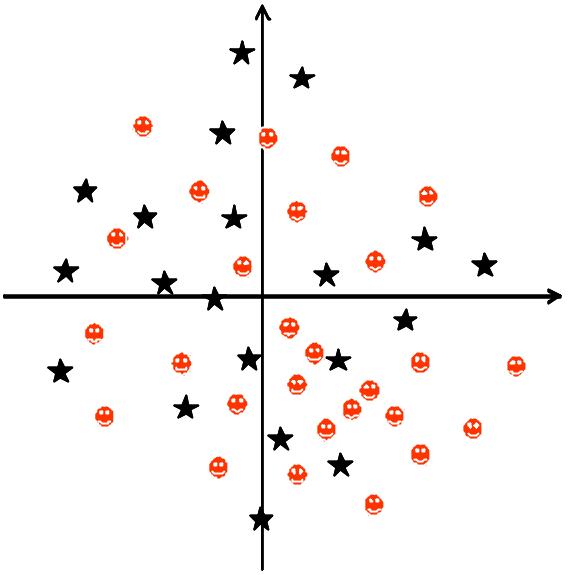


Figure 2 – an example of a specific input space, here smiles represent input pattern and the stars represent neurons

When we train or use a SOM with a given input pattern, we calculate the distance between that specific pattern and every neuron in the network. We then select the neuron that is closest as the **winning neuron**, and say that the pattern is **mapped onto that neuron** or that the neuron as won the representation of that input pattern. As a consequence the neuron will move towards the input pattern position in order to improve its representation (as shown in Figure 3). The extent of this movement is controlled by a parameter usually referred to as learning rate.

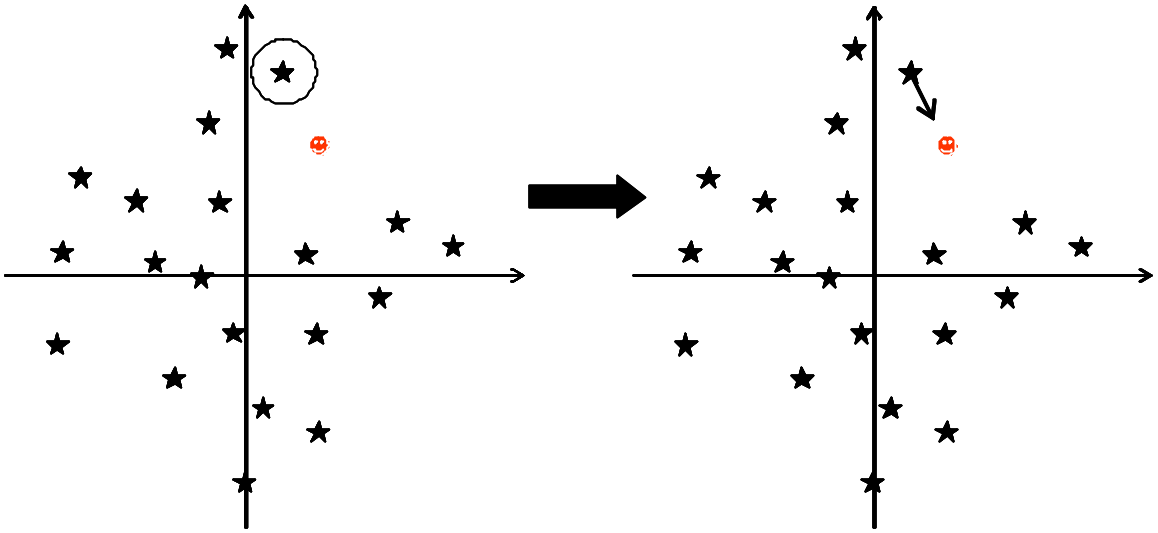


Figure 3 – the process of training the SOM involves the presentation of each input pattern to the network, the identification of the closest neuron (left side) and updating the position of the neuron in order to improve its representation of the input pattern

In order to preserve the topology in the output space it is essential to correct the position of the wining neuron but also the position of its neighboring neurons, as shown in Figure 4. This way the network is progressively organized (unfold) with certain parts of the input space being represented by certain subsets of neighboring neurons.

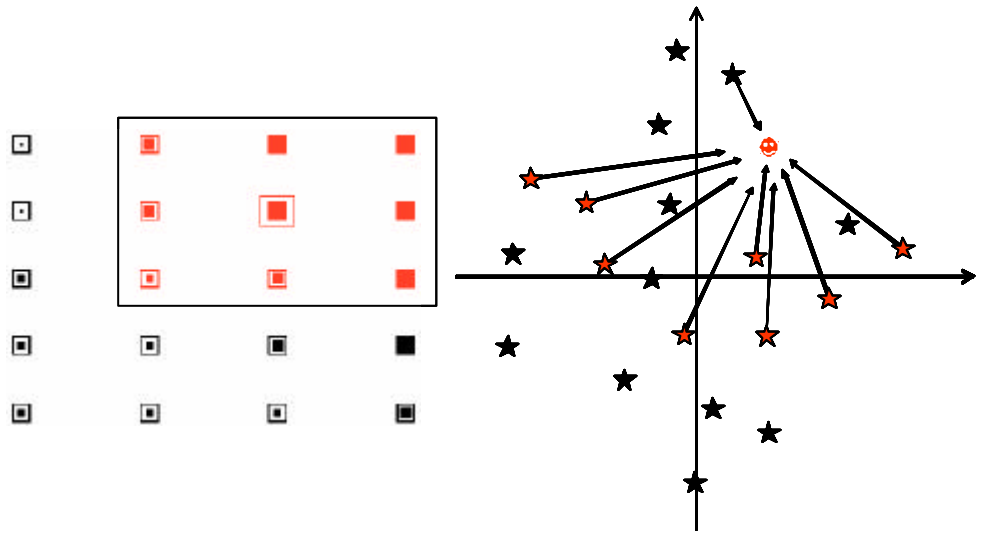


Figure 4 – the notion of neighborhood in the output space (left side) forces the movement of all the neighboring neurons in the input space (right side)

If the SOM has been trained successfully, then patterns that are **close in the input space** will be mapped to neurons that are **close (or the same) in the output space**, and vice-versa. Thus, as already said, the SOM is "**topology preserving**" in the sense that (as far as possible) neighborhoods are preserved through the mapping process.

Generally, no matter how much we train the network, there will always be some difference between any given input pattern and the neuron it is mapped to. This is a situation identical to vector quantization, where there is some difference between a pattern and its code-book vector representation. Thus, we refer to this difference as the **quantization error**, and use it as a measure of how well our neurons represent the input patterns. In other words the quantization error in calculated by summing all the distances between each input pattern and the neuron to which is mapped, giving a notion of the quality of the representation achieved by the SOM (in Figure 5 we have two neurons which have to represent 45 input patterns, the lines represent the distance between each input pattern and the nearest neuron).

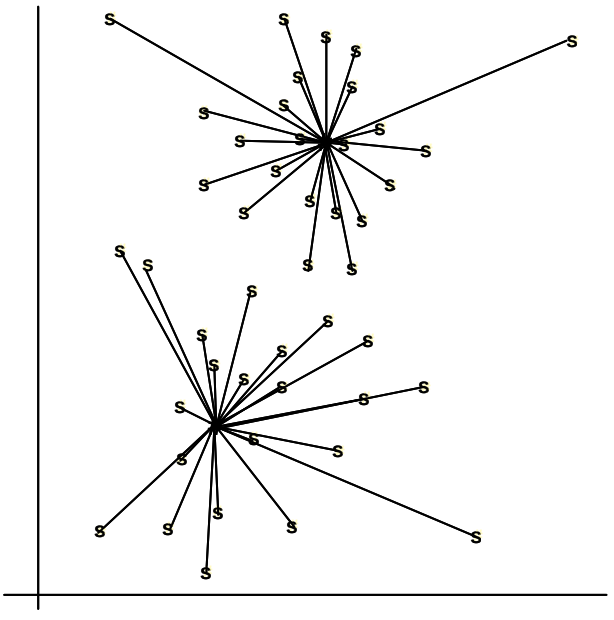


Figure 5 – representation of the input space with 45 input patterns and 2 neurons, the quantization error is given by the sum of all the distances of each input pattern to the nearest neuron.

In order to evaluate the complexity of the output space the **topological error** is computed. The topological error measures the average number of times the second closest neighbor to a given ED is not mapped to a neighbor of the first (Figure 6). The highest the topological error the more complex the output space is. A high topological error may indicate that the classification problem is complex and preserving the topology is difficult, or it may suggest that the training was not adequate and the network is folded.

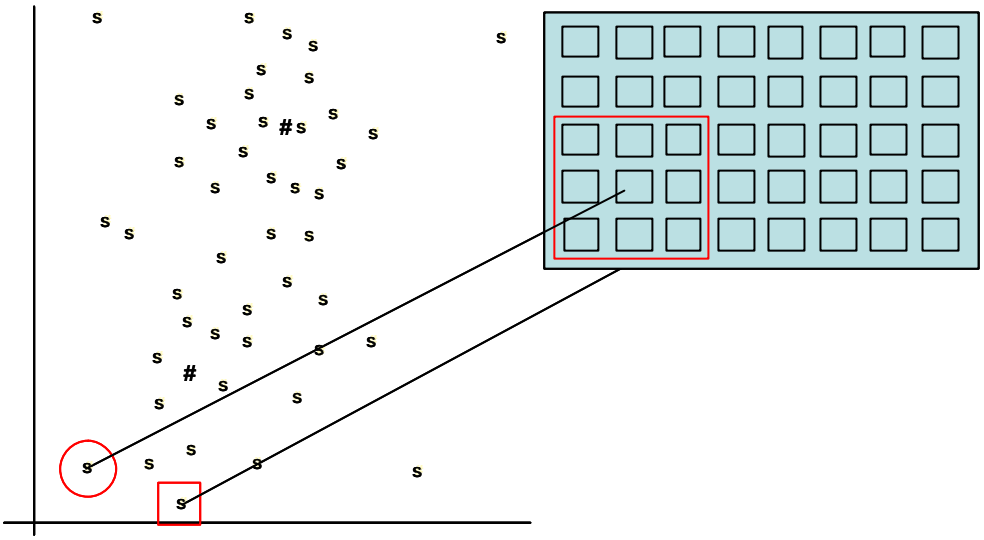


Figure 6 – the topological error is given by the number of times the second closest neighbor in the input space (left side) is not mapped into the neighborhood of the neuron in the output space (right side)

In this context, we can look at a SOM as a "rubber surface" that is stretched and bent all over the input space, so as to be close to all the training points in that space. This process is achieved in an iteratively fashion over a large number of epochs. In this sense, a SOM is similar to the input layer of a RBF (Radial Basis Function) neural net, a neural gas model, or a K-means algorithm. The big difference is that while in these methods there is no notion of "output space" neighborhood (all neurons are "independent" from each other), in a SOM the neurons are "tied together" in the output space. It thus imposes an ordering of the neurons, that is not present in the

other methods. These ties are equivalent to a strong lateral feedback, common in other competitive learning algorithms.

Before training, the neurons may be initialized randomly. Typically, the SOM training process comprises two major parts. During the first part of training, the neurons are "spread out", and pulled towards the general area (in the input space) where they will stay. This is usually called the **unfolding phase** of training. After this phase, the general shape of the network in the input space is defined, and we can then proceed to the second part, usually called **fine tuning phase**, where the SOM will match the neurons as far as possible to the input patterns, thus decreasing the quantization error.

To visualize the training process, let us follow a 2-dimensional to 1-dimensional mapping presented in [Kohonen 95]. In this problem, 2-dimensional data points are uniformly distributed in a triangle, and a 1-dimensional SOM is trained with these patterns. Figure 7 represents the evolution of the neurons in the input space. As training proceeds, the line first unfolds (steps 1 to 100), and then fine-tunes itself to cover the input space.

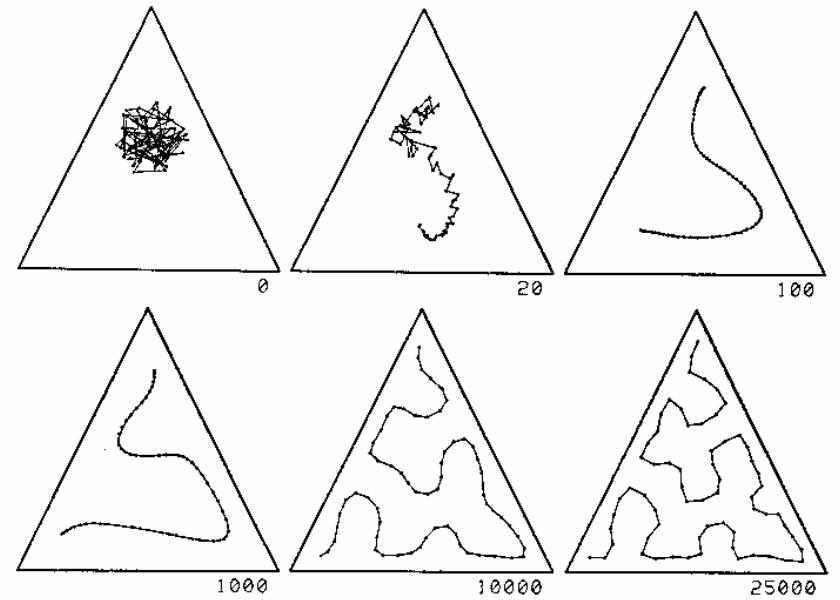


Figure 7 - 2D to 1D mapping by a SOM [Kohonen 95]

Another very common example of a SOM mapping, is used by the standard MATLAB demo of its neural network toolbox, is presented in Figure 8. There, a 2D map is trained on a collection of 2D points uniformly distributed in a square area. The position of the neurons in the input space is then tracked. This example is also useful to illustrate a rather annoying problem that may arise: local minima. In Figure 9 we can see a representation in the input space of a SOM that got stuck in local minima. The problem is that in the unfolding phase the network got tied up in and the proper unfolding is jeopardize. Clearly, this problem is a function of the complexity of the input space; the more complex the input space is the highest the probability of incorrect unfolding. Additionally, the ordering by which the input patterns are presented to the network can also lead to local minima solutions.

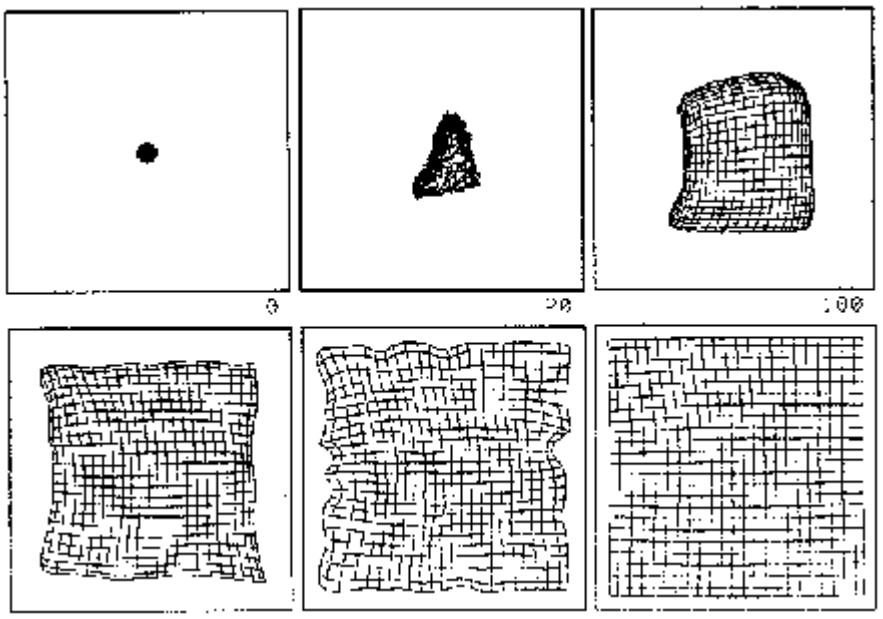


Figure 8 - A 2D to 2D mapping of a uniform distribution in a square [Kohonen 95]

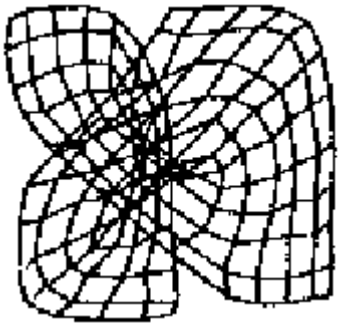


Figure 9 - An example of a folded map

4 - Formal description of the training algorithm

4.1 - The algorithm

Let x_k (with k=1 to the number of training patterns N) be the n-dimensional training patterns. Let w_{ij} be the neuron in position (i, j) Let $0 \leq \alpha \leq 1$ be the learning rate (sometimes referred to as η), and $h(w_{ij},w_{mn})$ be the neighborhood function (sometimes referred to as Λ or N_c). This neighborhood function assumes values in [0,1] and is high for neurons that are close in the output space, and small (or 0) for neurons far away. It is usual to select a function that monotonically decreasing and non-zero up to a radius r (called neighborhood radius) and zero from there onwards. Let w_{winner} be the winning neuron for a given input pattern.

The algorithm for training the network is then:

For each input pattern:

1. Calculate the distance between the pattern and all neurons ($d_{ij} = \| x_k - w_{ij} \|$)
2. Select the nearest neuron as winner w_{winner} ($w_{ij} : d_{ij} = \min(d_{mn})$)
3. Update each neuron according to the rule
   $w_{ij} = w_{ij} + \alpha \; h(w_{winner},w_{ij}) \; \| x_k - w_{ij} \|$
4. Repeat the process until a certain stopping criterion is met. Usually, the stopping criterion is a fixed number of iterations.

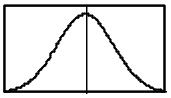
To guarantee convergence and stability of the map, the learning rate and neighborhood radius are decreased in each iteration, thus converging to zero.

The distance measure between the vectors is usually the Euclidean distance, but many others can and are used, such as norm based Minkowski metrics, dot products, director cosines and Tanimoto measures [Garavaglia 98], Hausdorff distances, etc.

4.2 - Neighborhood functions

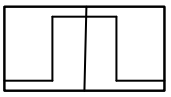
The two most common neighborhood functions are the gaussian and the square (or bubble):

$$h_g(w_{ij}, w_{mn}) = e^{-\frac{1}{2}\left(\frac{\sqrt{(i-n)^2+(j-m)^2}}{r}\right)^2}$$



and

$$h_s(w_{ij}, w_{mn}) = \begin{cases} 1 \Leftarrow \sqrt{(i-n)^2+(j-m)^2} \le r \\ 0 \Leftarrow \sqrt{(i-n)^2+(j-m)^2} > r \end{cases}$$



In both cases, r$\rightarrow$ 0 during training, this constitutes a requirement in order to achieve convergence.

The algorithm is surprisingly robust to changes in the neighborhood function, and our experience is that it will usually converge to approximately the same final map, whatever our choice, providing the radius and learning rate decrease to 0. The gaussian neighborhood tended to be more reliable (all runs would converge to almost exactly the same map), while the bubble neighborhood let to smaller quantization errors. A theoretic discussion of the effect of neighborhood functions (although only for the 1-dimensional case) can be found in [Erwin 91], and a less rigorous but more general one in [Ritter 92].

4.3 – A Simple Example of how the SOM works

To simplify the understanding of the workings of the SOM we present a very simple example which illustrates the major operations which occur during the SOM training. In this particular case and for reasons of simplicity the objective is to train a two neuron network. For this reason we will not use the neighborhood parameter. The input patterns are:

(1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, 0); (0, 0, 1, 1).

Number of neurons is 2. The initial matrix which defines the neurons has the following values (randomly generated):

Neuron random initialization

N_1	N_2
0.2	0.8
0.6	0.4
0.5	0.7
0.9	0.3

As we are using only two neurons neighborhood radius will be set to 0 in this particular case:

r=0

The learning rate is initialized with a 0.6 value, and will decreases in each epoch according with the following expression:

$$a_{(t+1)} = 0.5a_{(t)}$$

The algorithm will look like this:

Step 1: Begin training
Step 2: present the first input pattern (randomly chosen)

(1, 1, 0, 0)

Step 3: calculate the distance between the input pattern and each of the neurons:

$$D_{(1)} = (.2-1)2 + (.6-1)2 + (.5-0)2 + (.9-0)2 = 1.86;$$

$$D_{(2)} = (.8-1)2 + (.4-1)2 + (.7-0)2 + (.3-0)2 = 0.98$$

Step 4: The input pattern is closer to neuron N_2, this way N_2 will be updated according to the following expression:

$$wi2\ (new) = wi2\ (old) + .6\ [xi - wi2\ (old)]$$
$$= .4\ wi2\ (old) + .6xi$$

Thus, after this first iteration the updated neurons will look like this:

N_1	N_2
0.2	0.92
0.6	0.76
0.5	0.28
0.9	0.12

The process will go on until a predefined number of epochs is reached. The student can workout the rest of the input patterns, solutions (keeping the order by which the input patterns appear) are presented next.

After processing input pattern (0, 0, 0, 1), the neurons weights will look like this:

N_1	N_2
0.08	0.92
0.24	0.76
0.20	0.28
0.96	0.12

After processing input pattern (1, 0, 0, 0), the neurons weights will look like this:

N_1	N_2
0.8	0.968
0.24	0.304
0.20	0.112
0.96	0.048

After processing input pattern (0, 0, 1, 1), the neurons weights will look like this:

N_1	N_2
0.032	0.968
0.096	0.304
0.680	0.112
0.984	0.048

Once here the first epoch is finished and the learning rate has to be updated, using the expression presented above the result will be:

$$a_{(t+1)} = 0.5(0.6) = 0.3$$

Next, the second epoch will start, with a reduced learning rate. The process will continue until the predefined number of epochs is reached.

5 – Additional features and comments

The possibility of visualizing the results from the trained SOM is particularly interesting as it allows additional insight into the data and its structure. One of the ways of visualizing the results of a SOM, is to use U-Matrices [Ultsch and Siemon 1990].

The U-Matrix constitutes a representation of the output space of a SOM in which distances, in the input space, between neighbouring neurons are represented, by a colour code or a grey scale. If distances between neighbouring neurons are small, then these neurons represent a cluster of patterns with similar characteristics. If the neurons are far apart, then they are located in a zone of the input space that has few patterns, and can be seen as a separation between clusters. In the SOM_PAK distances are usually coded as grey shades, but there are software packages which use colours to code distance yielding results as presented in Figure 10.

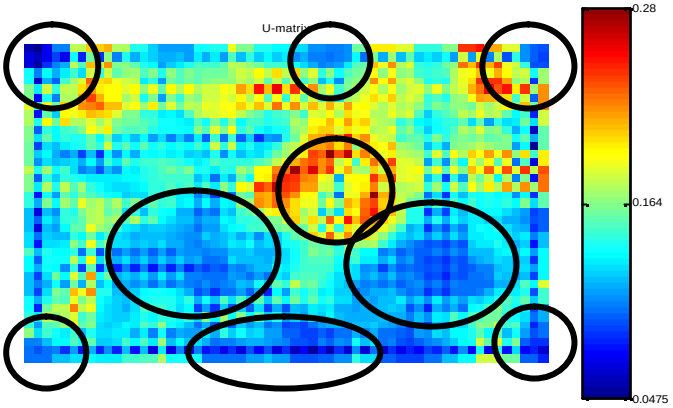


Figure 10 – an example of an U-matrix, in this particular case the major clusters are identified by the ellipses [Lobo et all. 2004].

In the context of geospatial data mining the findings made using the U-matrix can easily be visualized within their spatial context (Figure 11). This fact constitutes a very relevant functionality as it allows for "cross-space" analysis, allowing the validation in geographical context of the findings made on the U-matrix.

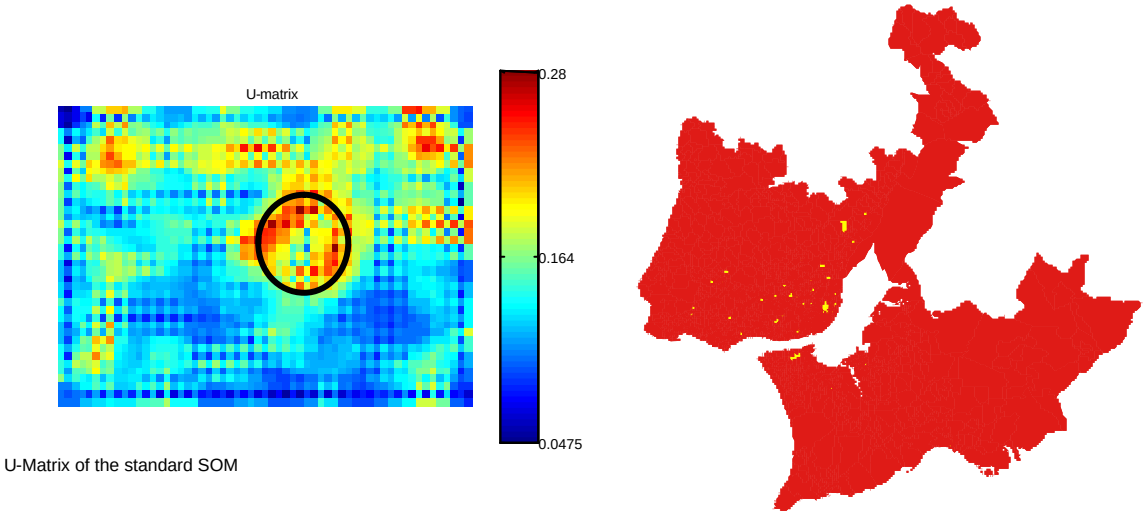


Figure 11 – the possibility of linking the analysis provided by the U-matrix with geographical maps is particularly useful. In this picture we can seen that the cluster identified in the U-matrix corresponds to a set of area characterized as "recent social housing" [Lobo et all. 2004].

Another solution for visualizing the SOM output space is to use the Sammon's [Sammon 1969] projection, which performs a non-linear projection from the input

space to a lower dimension. The algorithm tries to preserve all distances between input patterns, emphasizing local distances. The method is iterative and computationally challenging. One of the ways of circumvent this problem is to apply it to the neurons of a SOM instead of the input patterns [Vesanto, 1997], this means a simplification which reduces the computing burden. The projections of neighbouring neurons are usually connected by lines improving the visualization, as seen in figure 12. The Sammon's projection of a SOM gives a very informative picture of the global shape and the overall smoothness of the SOM.

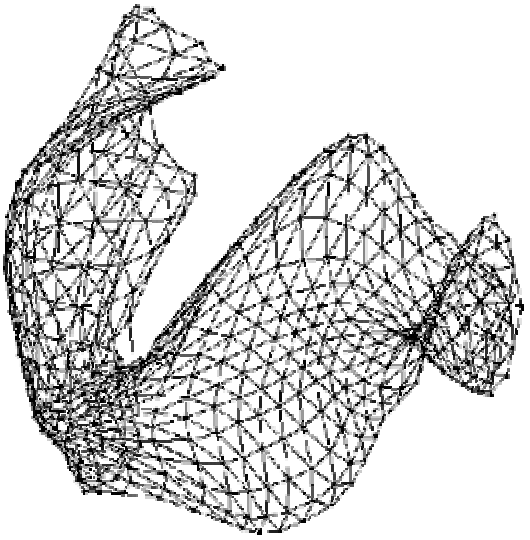


Figure 12 – an example of a Sammon's projection of a SOM, the neurons are represented by the black dots and are connected to their neighbors by the lines [Vesanto 1997].

Another interesting feature of the SOM algorithm is the possibility of slicing the SOM into "components planes" [Vesanto 1999]. The idea is quite simple but nevertheless very effective in terms of data exploration. This way each plane represents the value assumed by each neuron for each component in the vector. In Figure 13 five planes are represented, in each one of these planes the value of the neuron for each one of the five variables composing the neurons. The colour (in this case the grey scale) of each neuron represents the value of a specific vector component. Using this method is very

easy to understand how the different variables that comprise the input vectors are organized in the SOM output space.

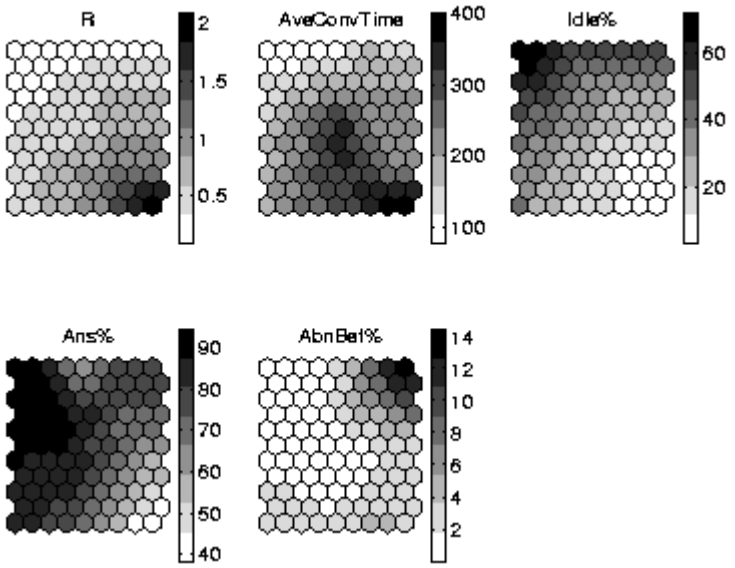


Figure 13 – here we can see an example of the visualization of components planes, in this case each vector has 5 components each of which is depicted in the output space of the SOM [Vesanto 1999]

As it was already pointed out, the quantization error gives an idea of the quality of the representation provided by the SOM. This value can be calculated for the whole map but also it can be calculated for each one of the neurons and each one of the input patterns. Neurons with a large mean quantization error should merit further analysis, as they, for whatever reason, are failing to properly represent the input patterns. The identification of outliers can be achieved through the analysis of the quantization error associated with each input pattern. In fact, input patterns with high quantization error have a high probability of being outliers. In the context of geospatial the analysis of the quantization error for each input pattern can easily be achieved through the mapping of the geographical features and its corresponding quantization error like in Figure 14 [Lobo et all. 2004].

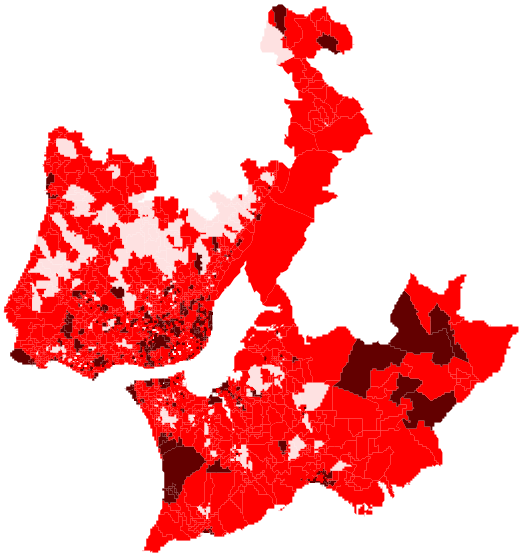


Figure 14 – the geographic representation of the quantization error, for a classification performed for the Lisbon Metropolitan Area [Lobo et all. 2004].

7 - References

Erwin, E; Obermayer, K; Schulten K., (1992) Biological Cybernetics, Vol67

Fu, LiMin, (1994) "Neural Networks in Computer Intelligence", McGraw Hill.

Garavaglia, S., (1998) "A Heuristic Self-Organizing Map Trained Using the Tanimoto Coefficient", Proc.of WCCI 98, Anchorage, USA.

Kohonen, T., (1982) "Self Organized Formation of Topologically Correct Feature Maps", Biological Cybernetics, Vol.43

Kohonen, T., (1988) "The Neural Phonetic Typewriter", IEEE Computer.

Kohonen, T., (1988) "Self-Organization and Associative Memory", Springer-Verlag.

Kohonen, T., (1995) "Self-Organizing Maps", Springer-Verlag.

Kohonen, T., (1995) "Self-Organizing Maps", 2nd Edition, Springer-Verlag, 1995

T. Kohonen (2001). "Self-Organizing Maps". Springer series in information sciences, 3rd ed.

Lobo, V., Bação, F., Painho, M., (2004) "The Self-Organizing Map and it's variants as tools for geodemographical data analysis: the case of Lisbon's Metropolitan Area", Proceedings from the 7th AGILE Conference, on Geographic Information Science, 29 April - 1 May, 2004, Heraklion, Greece.

Ritter, Helge; Martinetz,T; Schulten, K., (1992) "Neural Computation and Self-Organizing Maps", Addison-Wesley.

Sammon, Jr., J. W. (1969) "A nonlinear mapping for data structure analysis". IEEE Transactions on Computers, 18:401-409.

Ultsch, A. and Siemon, H. P. (1990) "Kohonen's self organizing feature maps for exploratory data analysis". In Proceedings of ICNN'90, International Neural Network Conference, pages 305-308, Kluwer, Dordrecht.

Vesanto J., (1997), "Data Mining Techniques Based on the Self-Organizing Map" Thesis for the degree of Master of Science in Engineering in the Helsinki University of Technology at the Department of Engineering Physics and Mathematics, available at: http://www.cis.hut.fi/projects/ide/publications/html/mastersJV97/index.html.

Vesanto, J., (1999). "SOM-Based Data Visualization Methods". In Intelligent Data Analysis, Volume 3, Number 2, Elsevier Science, pp. 111-126.