SELF-ORGANIZING MAPS – A HELPFUL TOOL IN CLUSTERING AREAS WITH SIMILAR FACTORS OF EROSION RISK

BARTKOWIAK A.[1], GOURNELOS TH.[2], EVELPIDOU N.[3], VASSILOPOULOS A.[3]

ABSTRACT

We use a two-dimensional self-organizing map (Kohonen's SOM) to cluster together geographical units characterized by 3 risk variables. The constructed map shows as neighbors units similar with respect to their risk variables. Generally the SOM represents a mapping from the space of the risk variables to a two dimensional planar map with preserving the topology in the original data space.

Speaking in more detail, we consider 3 risk variables recorded in n = 123 areas (units) at the Sifnos Island, as described by Gournellos et al. (2002). We show, how various aspects of the data can be exhibited in the constructed SOM. The presented methodology could be also used for data characterized by p variables with p3 as well.

KEYWORDS: Self-organizing maps, erosion risk, Sifnos (Cyclades), visualization of multivariate data, neural networks.

INTRODUCTION

Visualization of multivariate data is a very important topic. We use for that purpose Kohonen's self organizing map (SOM). The method performs a kind of clustering of the data points while preserving their topology in the original data space. The method is based on unsupervised learning developed in the framework of neural networks. Specifically, we use the method and algorithm developed by Kohonen.

The paper is organized as follows: First we describe shortly the method, i.e. under which principles the SOM is constructed. Next we perform the true case study, i.e. we construct a SOM for the data containing 3 erosion risk factors (Gournellos et al. 2002) and show what kind of information can be shown in the constructed SOM.

We think that this kind of displaying information about data is important in a GIS system, which should not limit itself to display of geographical information bound with geographical location of the units — but allow also for a comprehensive presentation of the data characterized by many (more then 2) traits.

METHOD

Suppose, we have a data matrix, in which rows denote geographical units, and columns — variables, characterizing these units. In our case we will consider a data matrix of size 123×3, where each row denotes a geographical unit (area, rectangle in a geographical map grid); each unit being characterized by 3 traits, called by us variables. We will consider the following traits: x1 — drainage density, x2 — slope (inclination) and x3 — vulnerability (erodibility). The definition of these variables is given in Gournellos et al. (2002). In fact each unit could be characterized by more variables, e.g. we might consider additionally for each unit its (average) height above the sea level. However, concerning erosion risk, the 3 introduced variables are known as the most important predictors. Therefore we have limited ourselves just to these 3

1:Professor, University of Wroclaw, Institute of Computer Science, Przesmyckiego 20, 51-151 Wroclaw, Poland.
2:Associate Professor, University of Athens, Geology Department, Physical Geography & Climatology Sector, Panepistimiopolis, Zografou, 157-84, Athens-Greece.
3:Dr. Geologist, University of Athens, Geology Department, Physical Geography & Climatology Sector, Panepistimiopolis, Zografou, 157-84, Athens-Greece.

variables. Also, it will be easier to present some concepts of self-organizing maps using only 3 variables.

Returning to the data matrix: suppose, it has n rows (in our case n=123). Each row (i) is a data vector containing the values of the 3 analyzed variables characterizing the i-th unit (in our case i = 1, ..., 123). Such data vector can be imagined as a data point located in the multivariate (in our case: 3-dimensional) data space. All points (obtained from rows of our data matrix) form a multivariate data cluster (called also sometimes a multivariate data cloud).

We would like somehow to visualize the multivariate data of the information about mutual position of the individual data points a loss as small as possible.

In the following, by referring to 'space' we will have in mind the 'data space' based on the mathematical concept of multivariate space of variables and not the physical concept of space bound by the triplet: longitude, latitude and altitude.

Kohonen's idea was to quantify the data space into m adjacent regions and establish a representative for each area. The regions are called Voronoi regions and the representative points are called codebook vectors or codebook points. Each point x of the data space can be assigned to one of the Voronoi regions: namely that one containing the codebook point which is the closest to the point x. Such codebook point is called its BMU, i.e. its best matching unit.

The concept of subdividing the data space into Voronoi regions has important implications. First of all, replacing the true data points by their BMUs - playing the role of their substitutes - permits for a substantial reduction of the data, because one codebook vector may represent several data points. Next: Making a projection of the codebook points onto a plane (generally: to a space of lower dimensions) permits to obtain a comprehensive view of the entire data cloud. The plane will be called a ‘map’.

How to design the projection plane alias the map? It may have different shapes — the most popular shape being is a sheet organized into hexagons or rectangles. Two examples of maps with a superimposed hexagonal and rectangular grid are shown in Figure 1.

The nodes located in each center of the hexagons (rectangles) are in one to one correspondence with appropriate codebook points located in the data space.

The projection should preserve the topology of the codebook points in the data space, which means more or less that if two codebook points are located in adjacent Voronoi regions, the projection should also show these codebook points as nodes in adjacent map units.

The maps exhibiting the projections may be viewed as a special kind of neural network. The nodes in the map may be imagined as neurons and the grid connecting the neurons as possible connections between the neurons.

Using the methodology of neural networks — with some restraints concerning the neighborhood of the neurons — the map is trained in subsequent iterations called epochs. The goal is that the location of neurons in the map corresponds (topologically) to the location of the codebook points in the data space. The algorithm is described a.o., in the book by Kohonen (1995), also in Vesanto (1999).

As a result of the training we obtain the map (SOM) representing the Voronoi regions of the data space and the corresponding codebook vectors.

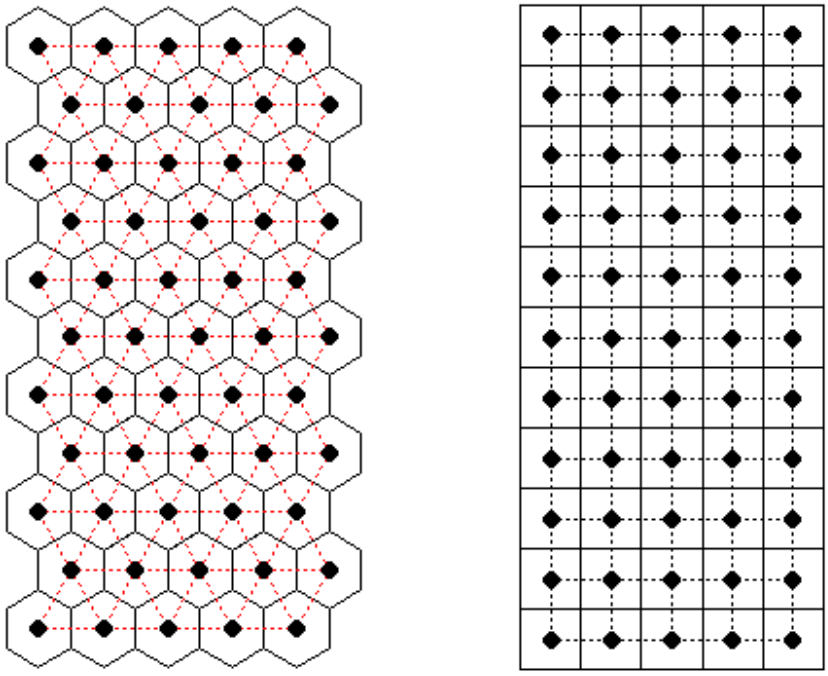

Fig. 1. Hexagonal (left) and rectangular (right) maps of size m = 11×5.

The quality of the SOM is evaluated ed by two types of errors:

1. Average quantization error (q) defined as the average distance — in the data space — from each data point to its BMU (closest codebook point).
2. Topographic error (t) defined as percentage of data points for which the BMU and the second BMU are not neighboring maps units.

Some practical advises given by Vesanto et al. (2000) are: The number of nodes (m) should be approximately equal to 5*sqrt(n), with n denoting the number of data points. The size of the map should be based on the ratio between two biggest eigenvalues of the covariance matrix of the given data; and the side lengths of the map are then set so that their product is as close to m as possible.

CASE STUDY

We consider data gathered in the Sifnos Island (Cyclades, Greece). The total area of that island was subdivided into n = 123 smaller territories in which a.o., the following traits (variables) were gathered: x1 — drainage density, x2 — slope (inclination) and x3 — vulnerability called also erodibility. The description of these variables is given in Gournellos et al. (2002), where also the data are published (as Table I).

We have taken for our analysis only 3 variables — to make our presentation more transparent and easier to describe. The data were normalized to their maximal value; thus our analysis was performed for variables taking real values from the interval [0 1].

Our goal is to visualize these data using Kohonen's SOM. For the calculations we have used the SOM Toolbox for Matlab 5 written by Vesanto et al. (2000).

As a shape for the map we have chosen a hexagonal sheet. The side lengths of the map were determined by default values of the package: these appeared to be m = 11×5. A map of just that size is shown in the left panel of Figure 1.

The training of the map has also used default values built in into the package. The training was quite fast: it needed only 5 iterations (epochs) in the rough training phase and 18 iterations (epochs) in the fine tuning phase.

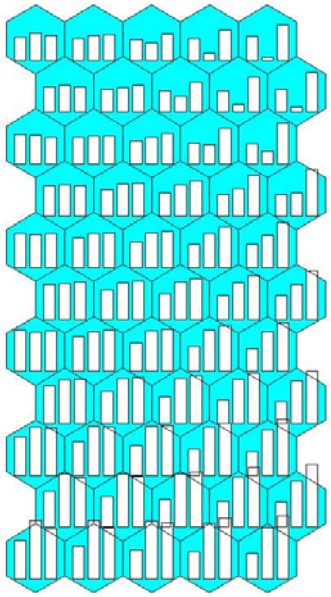

Fig. 2. SOM codebooks vectors visualized unitwise by bars.

As a result of the training we have obtained the following indices of the errors of the representation of the true data in the planar hexagonal map: q = 0.083 (average quantization error), t = 0.000 (topographic error).

Thus we may judge that the representation of the true data points in the constructed map is really satisfactory. The quantization error is quite small (taking into account, that the data values were scaled to [0 1]. Also the neighborhood of the codebook points in the data space and the neurons in the map grid is preserved totally.

Results I: Representation of codebook vectors

The final result of the training resulted in codebook vectors shown in Table 1.

Table 1. Codebook vectors for the constructed SOM of size 11×5

	1			***2***			***3***			***4***			***5***		
	x1	x2	x3	x1	x2	x3	x1	x2	x3	x1	x2	x3	x1	x2	x3
1	.35	.41	.37	.32	.38	.38	.31	.26	.40	.33	.10	.46	.36	.04	.53
2	.38	.41	.38	.34	.36	.40	.32	.23	.45	.32	.11	.53	.34	.07	.60
3	.45	.43	.40	.40	.43	.42	.35	.39	.46	.31	.29	.53	.30	.20	.62
4	.45	.46	.46	.40	.48	.49	.35	.46	.53	.32	.40	.61	.32	.34	.69
5	.51	.50	.51	.45	.51	.52	.39	.53	.55	.36	.52	.60	.36	.48	.68
6	.53	.54	.57	.45	.57	.58	.39	.57	.61	.37	.57	.67	.36	.53	.73
7	.62	.58	.61	.53	.59	.62	.43	.61	.63	.38	.62	.68	.35	.60	.73
8	.58	.65	.67	.48	.67	.69	.46	.68	.71	.36	.66	.74	.33	.65	.78
9	.58	.72	.72	.51	.73	.72	.44	.74	.75	.39	.72	.79	.36	.69	.84
10	.53	.80	.74	.46	.81	.77	.42	.80	.83	.39	.76	.90	.38	.70	.94
11	.58	.89	.76	.49	.88	.78	.43	.87	.84	.41	.84	.92	.39	.77	.96

Each codebook vector, containing a triplet of values <x1, x2, x3>, represents a Voronoi region in the 3-dimensional data space; at the same time it is connected in a univocal way with the appropriate node of the planar map.

The layout of the table corresponds to the lattice of the hexagonal map shown in Figure 1, left panel.

Looking at the values of the variables x2 and x3 shown in Table 1 one may state that — when moving from top to bottom of the table— they are increasing. But the changes of x3 cannot be described in such a simple way.

Let us display the same information graphically. This is done in Figures 2 and 3.

Figure 2 shows the codebook vectors from each unit by bar plots drawn in each hexagon of the map. We may state here quite easily that the upper right corner of the map has grouped codebook vectors with smaller values of x2, moderate or small values of x1, and small or medium values of x3. On the other hand, the bottom of the map has grouped observations with very high values of x2 and x3.

Figure 3 shows each variable in detail — what are its changes when passing from one hexagon to the other. This is done by using colors from a color bar shown at the right side of each panel. Deep blue denotes lowest values, intensive red — highest values. In such way we are easily catching the pattern of change of the given variable. Looking at the component planes shown in Figure

3 one may state, that the pattern of increase of x2 may be described by a normal slash ('/'), that of x3 – by a backslash ('\'). The pattern of x1 is more complicated. It is similar to a handwritten capital 'J' with lowest values starting in the north, medium values in the middle of south and increasing again to attain its highest values at the south-west. This means nonlinearity between the analyzed variables.

One can also see in Figure 3, the approximate fraction of units exhibiting e.g. very high values of subsequent variables, and that there is barely any linear correlation between these variables.

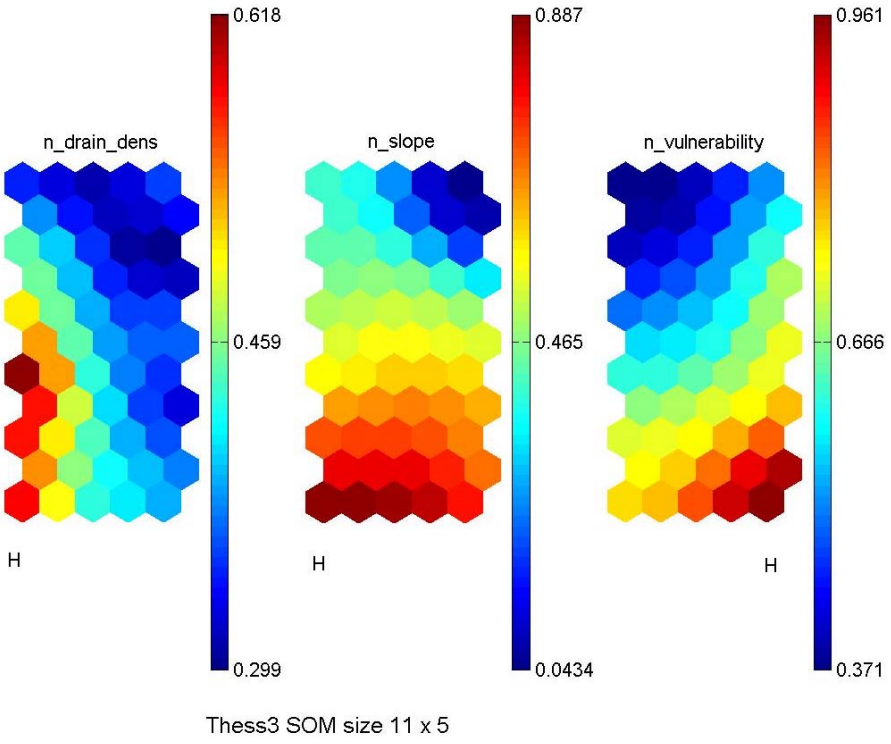

Fig. 3. Component planes shown separately for each analyzed variable.

Results II. Representation of the individual data points

Our question is: How many data points are represented by each of the codebook points located in the same Voronoi region of the data space and linked with the hexagons of the map?

The question can be answered by using a technique called 'hits'. For each data point we find its best matching unit BMU. At the same time we make a recording for each BMU, how many times it happened to be the best matching unit for some data points. Because each of the BMUs is linked in a unique way with a hexagon from the SOM, we have the answer to our question.

If the data cloud (n) is not too large, we may write directly in each node of the constructed map, which data vectors are represented by the corresponding codebook vectors. This is shown in Figure 4. The labels may be arbitrary strings.

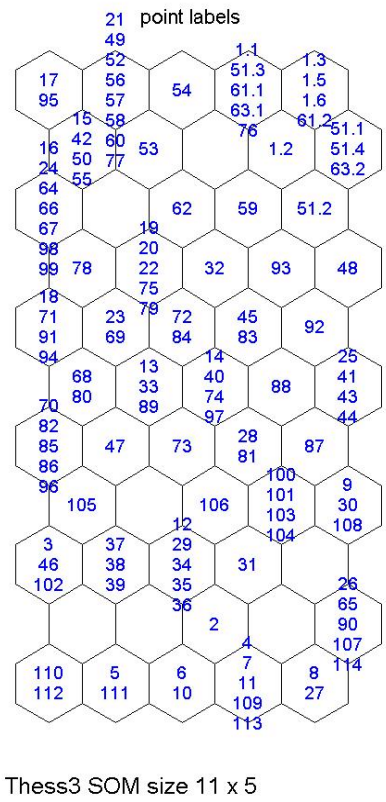

Fig. 4. Labels of data points linked with each map hexagon.

If n is large, writing directly the labels of all points may overshadow the map. Then we may put there another kind of information, e.g. how many data points are represented in each hexagon. This amount may be visualized either by coloring an appropriate part of the hexagon, or — alternatively — by writing in the center of the hexagon directly the frequency of points. Both ways are illustrated in Figure 5.

Both in Figure 4 and Figure 5 there are some empty units. This means that in the data space there are gaps with no data points located in those regions.

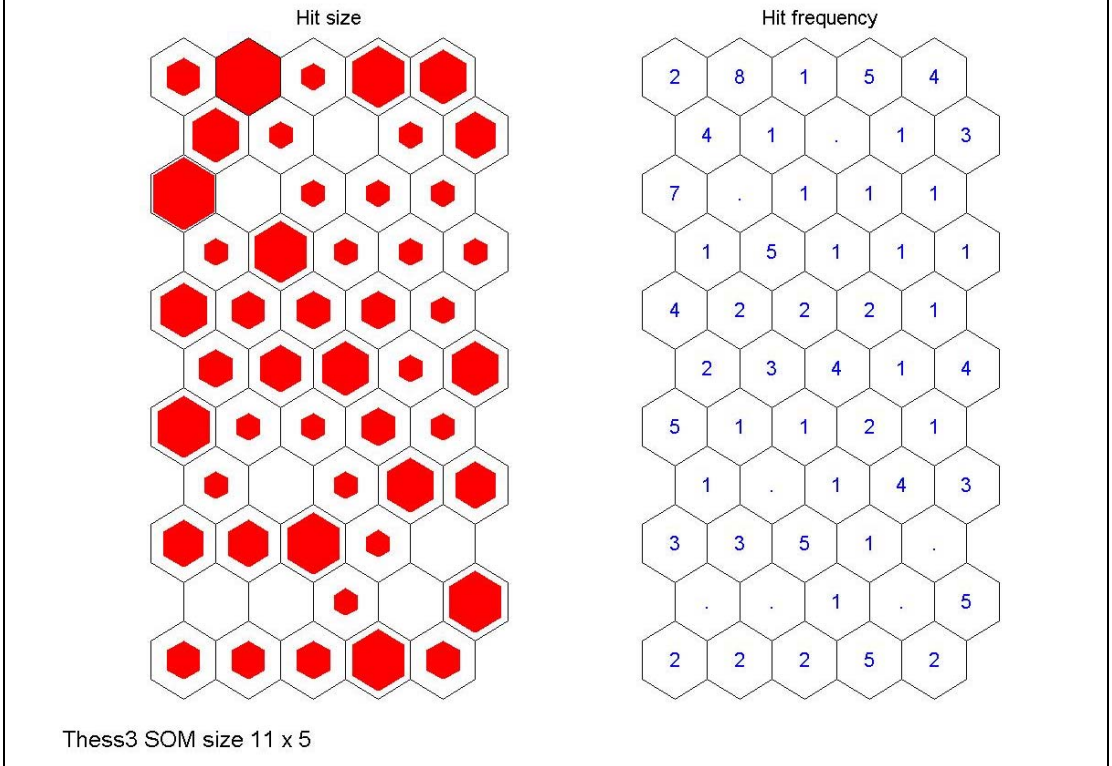

Fig. 5. Number of data points belonging to each unit. Left: visualization by size of the colored hexagon. Right: by writing direct the frequencies. Notice that some hexagons are empty, which means that the corresponding regions in the data space do not contain any data points.

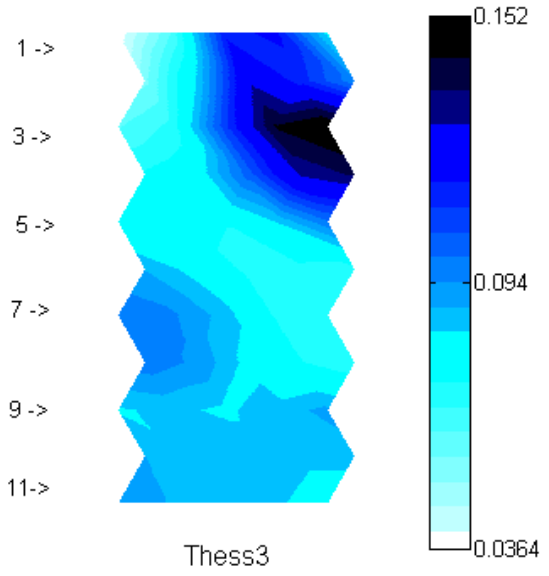

Fig. 6. Graph 'umat', shows distances between codebook points. Black means very distant units.

Results III. How distant are really the codebook vectors?

A very crucial question is: How far away are the codebook points — one from the other — when considering their true location in the data space. This can be answered using the technique 'umat' projections.

In Figure 6 we show distances between map nodes using a method called 'umat' (from the name: Ultsch) The graph was obtained using the package SOM-PAK written by Kohonen and his team The distances of neighbouring codebook vectors were smoothed appropriately. Dark colour indicates big distance. Bright colour means small distance between neighbouring points.

Results IV: Classification of data vectors.

Kohonen's maps may be also used for indicating which category (class) the analyzed data vectors belong to. Figure 7 below shows a subdivision of the analysed data into 5 classes of erosion risk.

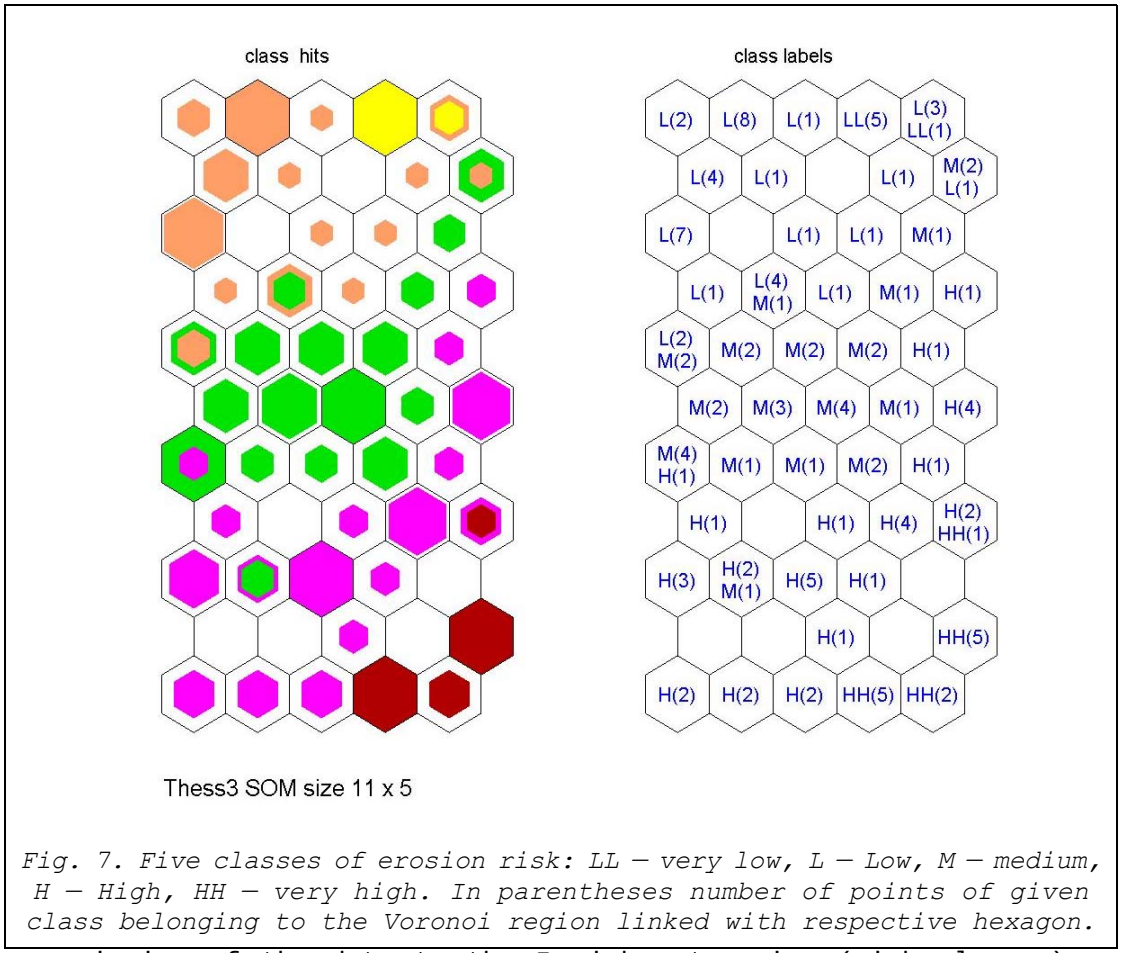

Fig. 7. Five classes of erosion risk: LL — very low, L — Low, M — medium, H — High, HH — very high. In parentheses number of points of given class belonging to the Voronoi region linked with respective hexagon.

For assigning of the data to the 5 risk categories (risk classes) we have used the 'Boolean logic' method (Gournellos et al., 2002). Five classes of erosion risk were established: HH: very high risk; H: high risk; M: medium risk; L: low risk; LL: very low risk. Then, using the technique 'hits' we painted the hexagons of the map according to the risk of

data vectors linked with subsequent hexagons. The result is shown in the left panel of Figure 7. Frequencies of risk classes are shown in the same figure, left panel.

The consistency between the risk classes and their location in the map is really striking! Dark red in the bottom right corner of the map indicates very high risk (HH). Bright yellow near the upper right corner permits to identify units with very low erosion risk.

CLOSING REMARKS

We have presented how self-organizing Kohonen's maps (SOMs)— obtained by applying an unsupervised self-learning technique — may be useful in representing data points from a multivariate data space. The obtained SOM can exhibit information not only on the neighborhood of the data points and their topology, but also on the (multivariate) density distribution of the analyzed data cloud.

The method may serve also as a tool for comparing graphically two multivariate distributions or two results of classification. E.g. for our data we have compared erosion risk to the analyzed units. The erosion risk was calculated by two methods with substantial different underlying philosophy. Because of the lack of space we could not show these results in the presented paper.

REFERENCES

[1]Gournelos Th., 1980, Contribution l'etude geologique des Cyclades, L'ile de Siphnos, These de 3 eme cycle, Universite de Paris VI, p. 182.
[2]Gournelos Th., Evelpidou N. and Vassilopoulos A., 2002, Developing an erosion risk map using soft computing, Natural Hazards, Submitted.
[3]http://www.cis.hut.fi/projects/somtoolbox/
[4]Kohonen T., 1995, Self-organising Maps, Springer, Berlin — Heidelberg.
[5]Vesanto J., 1999, SOM-based data visualization methods. Intelligent Data Analysis, 3 (2), pp. 111–126.
[6]Vesanto J., Himberg J., Alhoniemi E. and Parhankangas J., 2000, SOM Toolbox for Matlab 5, Som Toolbox team, Helsinki University of Technology, Finland, Libella Oy, Espoo 2000, pp. 1–54.