Visualizing self-organizing maps with GIS

Tonio Fincke[1], Victor Lobo[2], Fernando Bação[3]

[1]Institute for Geoinformatics, University Münster, Germany
fincke@uni-muenster.de

[2]Portuguese Naval Academy, Almada, Portugal
vlobo@isegi.unl.pt

[3]ISEGI, Universidade Nova de Lisboa, Portugal
bacao@isegi.unl.pt

Abstract. Self-organizing maps (SOM) are a powerful tool for detecting patterns in large, multi-dimensional data sets. Additional visualization techniques have been developed to support the user to gain insight into its structure. For complex data sets, even these techniques are not easily interpretable. Most of them consist of a grid where each cell contains a single value. Such a structure can be seen as an artificial landscape. This paper aims to explain the function of the SOM algorithm and to present a number of frequently used visualization techniques. We show a way to import traditionally created SOM into a GIS, so that operations created for spatial analysis can be applied to this originally non-spatial data. We present GIS operations that help the user to understand the structure of a visualization technique, its underlying SOM, and eventually the input data set.

1 Introduction

Due to the improvement in data collection, data sets tend to become larger, more complex and thereby more difficult to analyze with traditional statistical approaches. This holds especially true for sets with an unknown structure. Moreover, patterns that are not expected to appear in a set might be overlooked.

Data mining algorithms have proven useful for the evaluation of these high-dimensional data sets (Fayyad et al., 1996; Hand, 1998). One of the most popular data mining techniques is the Self-Organizing Map (SOM) algorithm (Kohonen, 2001). It can be used for data projection and thereby allows for visual inspection. Numerous applications have been developed for emphasizing different aspects of the SOM (Vesanto, 1999). With these applications it is possible to detect similarities or dissimilarities within the data set (Kaski et al., 2000).

However, the performance of the SOM depends on the complexity and

diversity of the data set. With our work we evaluate the possibilities of using operations offered by a Geographical Information System (GIS) in order to carry out a satisfying examination of the data under consideration. Before we can do this, we have to import the data generated by the traditional SOM producing programs into a GIS. After that we will present some operations provided by GIS and show how they can be used to make statements about a SOM and its underlying data set.

2 Self-organizing maps

A self-organizing map is an artificial neural network. Its units are arranged in a fixed order. In most cases, this would be a 2-dimensional rectangular or hexagonal grid of the form *(n*m)*. Each unit is associated with a weight vector (or: codebook vector) w_i which is of the same dimension as the input data.

Before a map can be trained, the codebook vectors w_i must be initialized with values similar to those of the input data. During the training phase the input vectors are presented to the map consecutively. The unit of the codebook vector w_i that shows the smallest distance to the input vector x is declared the Best Matching Unit (BMU). This process is also referred to as Vector Quantization (Vesanto and Alhoniemi, 2000). A codebook vector w_i of the map is updated via

$$w_i^{t+1} = w_i^t + h_{ci}^t * \alpha^t * d(x, w_i^t)$$

where t is a point in time, h^t_{ci} is a neighborhood function, α^t is a learning rate and $d(x, w^t_i)$ is the distance between an input vector x and the codebook vector w_i. Both α^t and h^t_{ci} lie within the interval *[0,1]* and decrease by time. The neighborhood function h^t_{ci} is defined by the topological or geometrical relationship between w^t_i and the codebook vector w^t_c of the BMU. After n training cycles the values of the codebook vectors w^n will represent the distribution of the input vectors (Kohonen, 2001).

3 Visualization of the codebook vectors

Different techniques are used to visualize the codebook vectors and the relationships amongst them. We will introduce some of them in this chapter. They will create values which are visualized on the grid through a shade from a color scale. Usually a grey scale is used where bright shades indicate low and dark shades indicate high values.

In order to explain the function of these visualization techniques we will

use two working examples: One data set where the dimension of the input vectors is 2 and one where it is 4.

For the data set with only two variables it is possible to plot not only the input vectors but also the codebook vectors of the SOM onto a plane with the attributes as coordinate axes. This 2-dimensional representation suffices to show the relations ot the codebook vectors to their neighboring codebook vectors and the input vectors. We will show the corresponding visualization techniques in addition so that the meaning of their values becomes clear. For the four dimensional data set we will have to fully rely on the visualization techniques.

The data set with only two variables is the data from the map John Snow created in September 1854. He plotted the location of cholera cases on a London street map and thus could eventually determine a water pump around which the disease cases cumulated as a source of contamination (Painho et al., 2005; Tufte, 1982). The variables here are the x- and y-dimension. For this set, we created and trained a hexagonal SOM with an extension of *(7*7)* units with SOM_PAK (Kohonen et al., 1996).

This data set is special, since it is purely spatial. It has no thematic attributes. Although it has been proposed to include also the geometrical attributes into the SOM-algorithm (Baçao et al., 2004), when dealing with spatial data the common approach is to use only the thematic attributes for the building of the SOM. This is often done when the output of the SOM is visualized together with a geographical map, so that the spatial properties are shown otherwise (Koua and Kraak, 2004).

The 4-dimensional data set is the iris flower data set where the sepal length, sepal width, petal length and petal width are the input dimensions. The data set consists of 150 entries: 50 per flowers of the species Setosa, Virginica, and Versicolor, respectively (Anderson, 1935). The SOM is a hexagonal SOM with *(4*6)* units. It was computed with SOM_PAK, too. This dataset has the property that for each entry it is known which species it describes. Therefore each codebook vector that could be said to represent flowers of a certain species could be labeled with the name of that species. Codebook vectors that could not be assigned to a species were labeled as NL.

In contrast to the John Snow data set, the iris flower data set has no spatial attributes at all. In previous work, the usage of geovisualization methods for examining SOM has mainly been limited to the investigation of the thematic attributes of spatial data (Guo et al., 2005). It is important to note that any such restriction is unnecessary. A data set does not need to be spatial in order to have its resulting SOM be analyzed by means of a GIS.

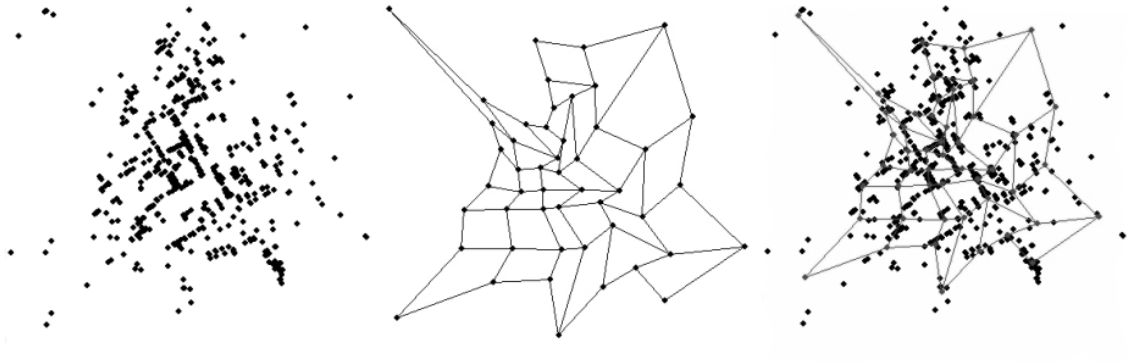

Figure 1: The data of the John Snow map. On the left, the data set is directly plotted. The middle image shows the codebook vectors of the trained SOM. Edges indicate connections between neighboring vectors. The image on the right shows them together in the same image. One can well see that the codebook vectors tend to be close to each other when the input vector distribution is tense. Vice versa, the distances between the codebook vectors are large in areas with a sparse input vector distribution.

3.1 Component planes

A component plane shows the relative values of one of the components of the codebook vectors (Kohonen, 2001). Obviously, the number of component planes per SOM is equal to the size of the input dimension. We receive two component planes for the John Snow map and four for the iris flower data set.

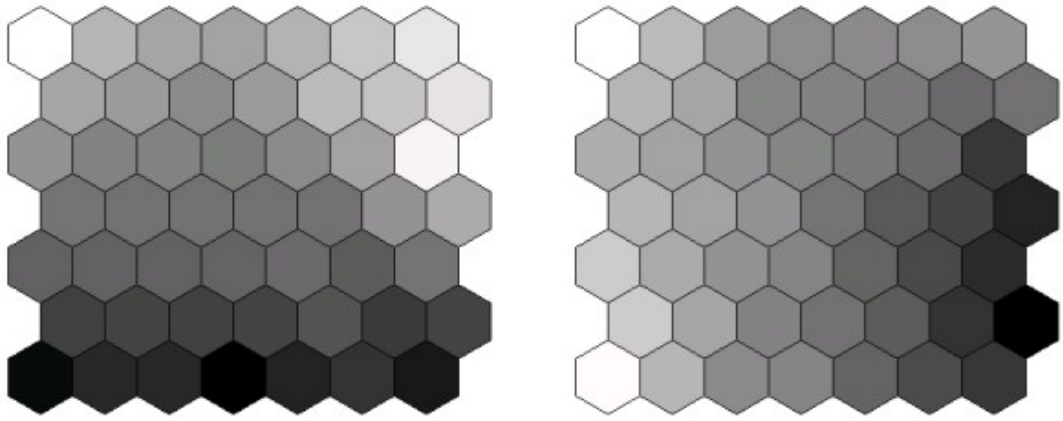

Figure 2: The component planes of the SOM of the John Snow map. Note that in this special case the values stand for the x- and y- coordinates. Therefore values in the left plane tend to increase with the row number, whilst the values in the right plane tend to increase with the column number.

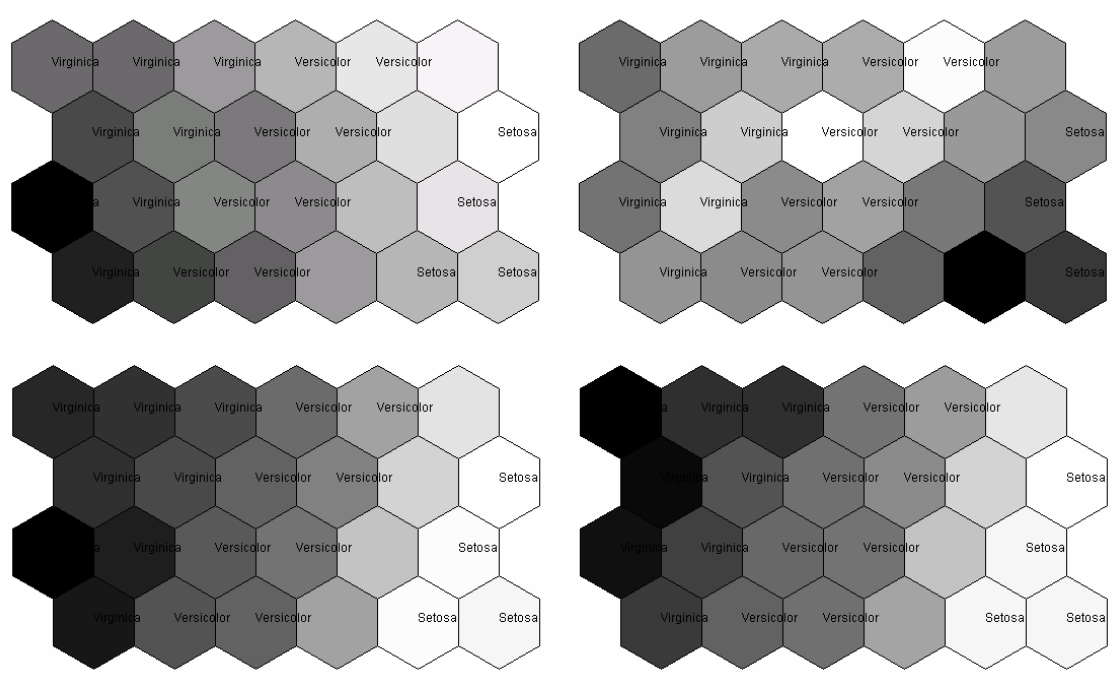

Figure 3: The component planes of the SOM of the iris flower data set. From top left to bottom right: The component planes showing sepal length, sepal width, petal length and petal width. The similarity of the planes 3 and 4 hint to a high level of correlation between petal length and petal width. These planes also imply that the petal length and width of flowers of the species Setosa are significantly different from those of flowers of the other species.

3.2 U-MATRICES

The u-matrix is used to visualize the distances between the codebook vectors of a two-dimensional map. For a grid with a *(n*m)* - extension, the u-matrix usually has a *(n-1*m-1)* - extension. The following formula shows as a distance measure for a codebook vector w_i the average distance to its immediate neighbors:

$$uh_i = \frac{1}{n} \sum_j d(w_i, w_j),\ j \in N(i), n = |N(i)|$$

where $d(w_i, w_j)$ denotes the distance between two vectors *i* and *j* and N(i) is the set of neighboring units of vector i.

The *(n-1*m-1)* - sized u-matrix will also show the immediate distances between neighboring units, thus providing much more significance. The grid value is called the u-height uh_i (Sommer et al., 2002). The u-matrix is especially well apt to detect clusters and the boundaries between them. In

the u-matrix, small values will tend to cover an area with two-dimensional extension while large values rather align in line-like structures.

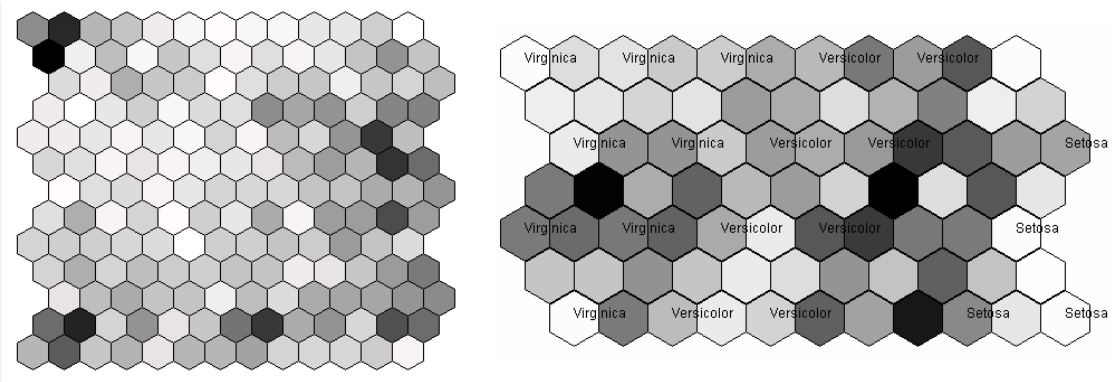

Figure 4: On the left: The u-matrix for the John Snow data set. On the right: The u-matrix for the iris flower data set. Units with odd row and column numbers contain values that were calculated by the formula given above. The values for the cells in between contain the distance between the codebook vectors of two neighboring cells. Dark values indicate large distances. Whilst the u-matrix for the John Snow map does not show any interesting patterns except for some units at the borders, the u-matrix for the iris data set shows well that the Setosa-labeled codebook vectors have different values than the others.

3.3 P-MATRICES

A p-matrix is a *(n*m)* – sized grid. It calculates the data density around a codebook vector. The value contained by one grid cell of the p-matrix is its p-height *ph(i)* and is calculated as

$$ph(i) = |\{x \in E | d(x, w_i) < r, r \in \mathbb{R}^+\}|$$

where $d(x, w_i)$ is the distance between the codebook vector w_i and x, E is the input data set, x is an input vector, and r is a radius around w_i. Obviously, the choice of r is crucial. A proposal is given in Ultsch (2005).

The p-height *ph(i)* is an integer value which tells how many input vectors have smaller distance to w_i than r. The p-matrix is not to be confused with a hit map which would show how often a unit was chosen as a BMU. The differences are that in the p-matrix an input vector can be assigned to more than one unit and an input vector is not necessarily assigned to a codebook vector.

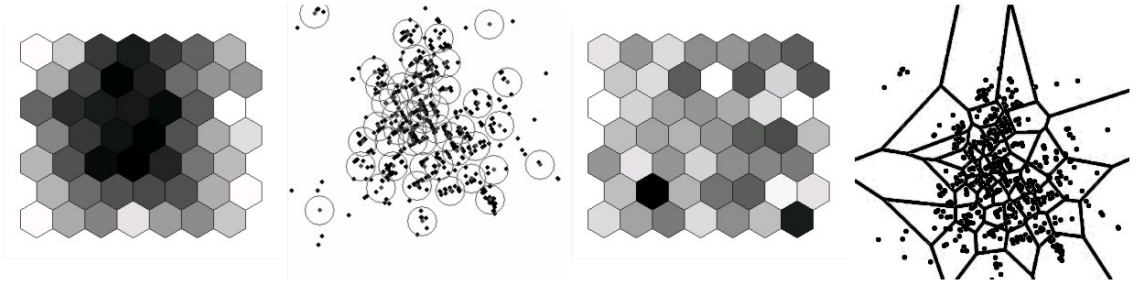

Figure 5: The first image shows the p-matrix of the John Snow map SOM. In contrast to the u-matrix, here dark values indicate clusters. The second image shows a visualization of the function of the p-matrix. The more input vectors lie within a given radius around a codebook vector, the higher the p-height *ph(i)* will be. The third image shows the hit map of the John Snow map SOM. The fourth image shows the visualization of its function. Instead of overlapping circles which only cover part of the space, Voronoi polygons show the function of the hit map. Voronoi polygons do not overlap and cover all of the space (Voronoi, 1907). The p-matrix implies that both codebook vectors and input vectors cluster in the middle of input space. This is verified by the second image.

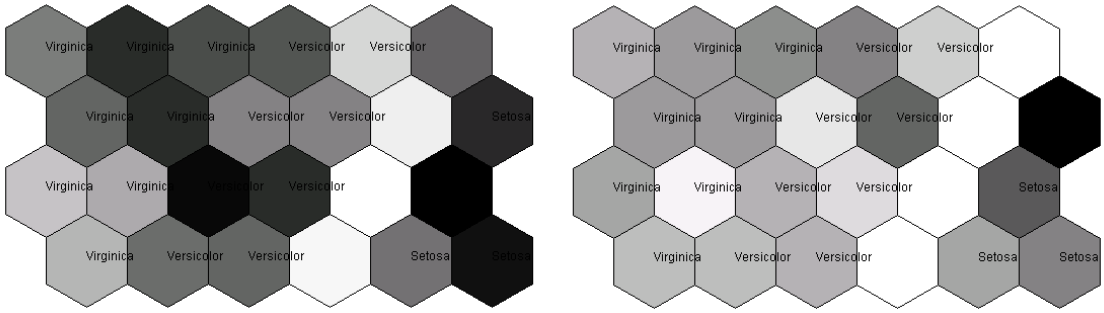

Figure 6: P-matrix (left) and hit map (right) of the SOM for the iris flower data set. In the p-matrix, the three very bright cells are another hint to the differentiation of the Setosa-labeled codebook vectors towards the other ones. The hit map reveals that few or no input vectors have been assigned to the codebook vectors that have stayed unlabeled.

Apart from these visualization techniques exist numerous more, notably the u*-Matrix (Ultsch, 2007) and Sammon's Mapping (Sammon, 1969) .

4 Conversion of the SOM

The first step of our work was to find a way to convert the SOM data created by SOM_PAK or the SOM toolbox (Vesanto et al., 2000) in such a way that its content can be read by a GIS. ArcGIS provides a functionality to create features from a text file (Liebig and Mummenthey, 2005). In this

text file the type and coordinates of each feature have to be specified. We have implemented a tool in Java that writes such a text file.

The services of the tool can be split into three components: import, operational and export. Since SOM_PAK and the SOM Toolbox use the same file structure, the tool can read SOM and input data from either. In the first line of a SOM file, the dimension *n* of the vectors, the topology type (hexagonal or rectangular), the map dimensions in *x*- and *y*-direction and the neighborhood type are specified. Each of the following *x*y* lines represents one codebook vector and contains *n* values, one per dimension. Optionally, the codebook vector lines can also specify the label of the vector. The structure of a file containing the input vectors is similar, except that the first line contains only the dimension *n* of the vectors.

In the tool, the SOM, its input data, and its visualization types are placed together in a group. Input data can only be imported into a group if the codebook vectors of the SOM are of the same dimension as the input vectors or if the group does not already contain a SOM. The same is valid for the import of a SOM.

If a group contains a SOM, the tool can evaluate the component planes and different types of u-matrices. If the group also contains the input data, the tool can evaluate the p-matrix, the hit map and the u*-matrix of the SOM. The tool can display all these visualization types by drawing hexagons or rectangles and filling them with a grey shade according to the respective value. The highest value will be coded as black, the lowest value as white. Note that all screen shots of this paper stem from this tool.

The third component of the tool is the export functionality. The tool is able to export every type of SOM that can be represented as a grid of rectangles or hexagons on which a shade of grey is applied. This accounts for every visualization type that can be calculated by the conversion tool. For reasons of simplicity the visualization types are exported in such a way that they can be read as points by ArcGIS. The text file to be created consists of a line specifying the desired feature type - in our case points - and *x*y* lines per unit. Each of these lines consists of an ID and coordinates for the *x*-, *y*- and *z*-value. We decided to choose the center of each rectangle or hexagon as the point representing an unit. The *x*- and *y*-values were assigned to each of these points so that the relative distances between the points would be preserved. The actual size of the distance is of no importance. After that, the value of the visualization type to be encoded was brought into relation with the extension of the *x*- and *y*-dimensions, so that the largest *z*-value would be of the same size as the largest *x*- or *y*-value. ArcGIS provides a command called "Create Features From Text File" which reads the data and creates a set of 3D point features.

Another way to import the data into ArcGIS was to use its "Quick

Import"-Function. This function allowed us to also import the labels. As import format, we chose the CITS Data Transfer Format. This format is very similar to the plain text file. A difference is that it has no header line. For each point, three lines must be written: One line for the attributes, i.e. the label, one line indicating the number of points of which the feature consists, i.e. 1, and one line for the coordinates that are given as described above. From these files, ArcGIS creates a geodatabase that contains a set of 3D point features.

These features form the base of our work in ArcGIS.

5 Visualizing SOM in ArcGIS

In this chapter we want to show some of the applications we have encountered. We will differentiate between two kinds of visualization types. The first sort consists of those types that tend to show a landscape-like structure with hills and valleys. These are u-matrices, p-matrices, hit maps and u*-matrices. The other kind of visualization types consists of those where large and small values tend to lie in opposing corners or edges of the grid. This holds true for the component planes.

5.1 Analysis of landscape-like Visualization Types

One way to deal with the landscape-like visualization types is to transform them into a Triangulated Irregular Network (TIN). A TIN is a vector data structure that partitions space into contiguous, non-overlapping triangles. The vertices of the triangles are points in 3D space. The triangles are formed by edges connecting the points. We could directly create a TIN from the point feature data. In order to perform the operations, it is necessary to transform the TIN into a raster where each cell contains an elevation value. This elevation value is interpolated from the nodes forming the triangle (Freiwald et al., 2005). For reasons of visibility in a 3-D environment, the hillshade is shown, too. The hillshade operation sets a hypothetical light source and calculates a brightness value for each triangle.

Yet another way to get an overview of the structure of the u-matrix is the contour operation. It creates a set of polylines. Each of these polylines consists of neighboring points with the same elevation value. The amount of polylines and elevation values to be displayed can be chosen by the user. An advantage of the polylines is that they only show when change in elevation happens, but not the absolute value. Therefore they eliminate the distracting influence of the color values.

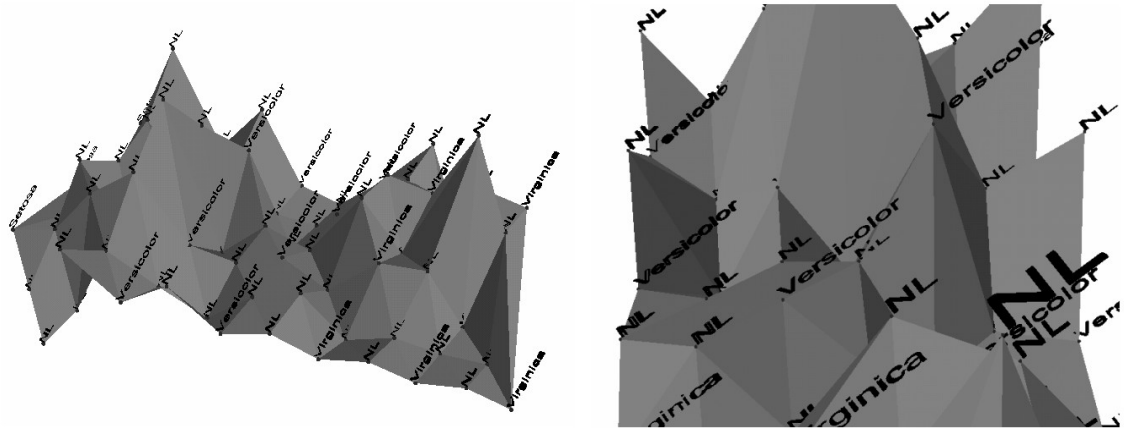

Figure 7: The TIN derived from the u-matrix. The TIN allows users to zoom into, rotate, fly, and navigate through the data. Thus the user is enabled to explore those parts of the landscape which are most interesting to him. Weaknesses of the TIN are its tendency to occlude large parts of the landscape and the high likelihood that the user gets lost in the landscape whilst navigating.

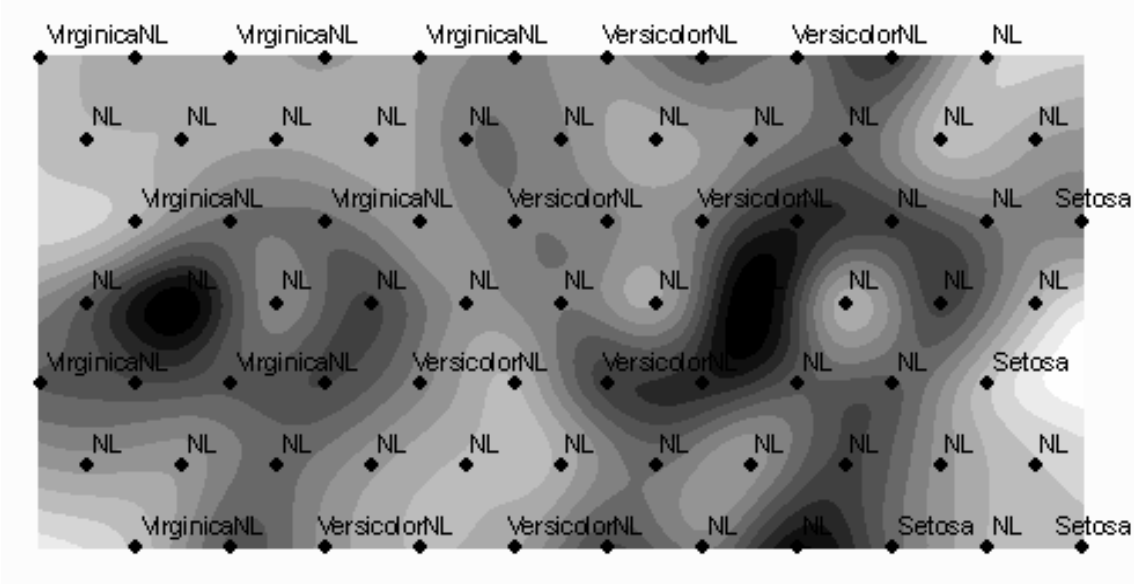

Figure 8: A raster set derived from the point data of the u-matrix. Elevation values are calculated via spline interpolation over the point features. Each elevation value is assigned to a color value from a continuous scale. In this image, the hill between the Virginica vectors on the left becomes more evident than in the traditional u-matrix representation with the grid. It seems that the Virginica flowers are quite diverse.

5.2 Analysis of Component Planes

Another possibility to analyze a SOM is to inspect the component planes.

One of them alone has only little explanatory power over the SOM, but in their entirety they describe it very well. A GIS provides methods to retrieve information from a number of component planes by visualizing more than one of them into the same space or by performing statistical analysis that considers all component planes.

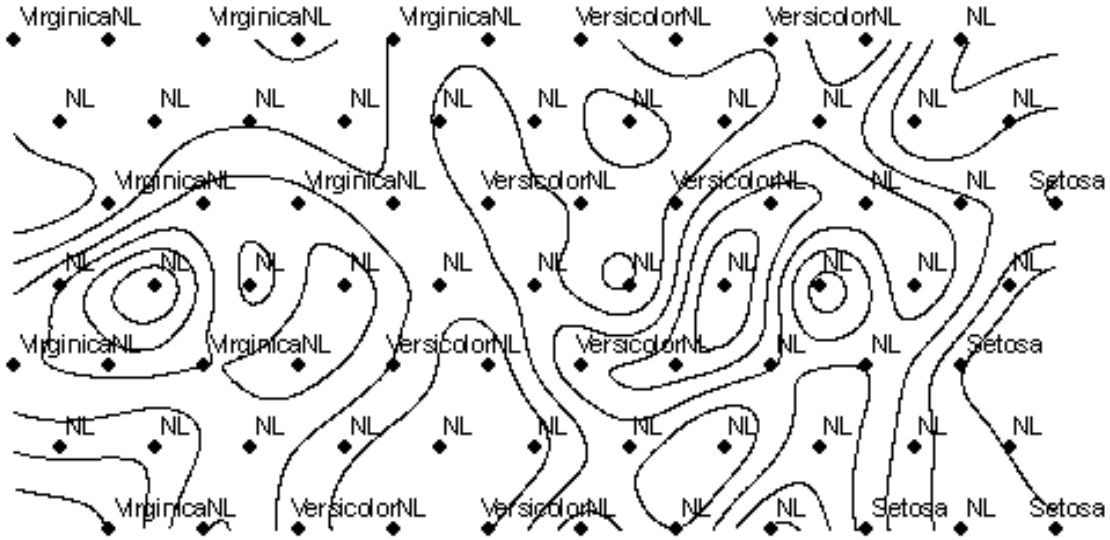

Figure 9: Contour lines of the u-matrix. It can be seen that codebook vectors from the types Setosa and Versicolor lie in areas with smaller distractions. This hints to a homogeneity of flowers of these species.

One way to visualize multiple component planes in the same image is the composite bands operation. It creates a single raster set from multiple sets with the same spatial extent. Thus it becomes possible to visualize three component planes at the same time by using the RGB color spectrum (Cowlishaw, 1985). The values of one plane are used as either red, green or blue color components. The resulting colors are displayed. This operation is especially apt to show differences between clusters, since different values will result in different colors. The only drawback is that just three component planes can be displayed at the same time. For the iris data set with four dimensional data we needed four composite band combinations in order to map every combination of planes.

The maximum likelihood classification is a statistical operation to perform classifications for a set of raster sets that cover the same spatial extent. It needs a signature file. One way to create such a file is the ISO cluster operation. For this operation one can specify the number of desired clusters and the raster sets on which to perform the analysis. The algorithm arbitrarily chooses centers in multi-dimensional space and assigns the cell values with the closest euclidean distance to exactly one of these centers. After all cells have been assigned to one center, the mean of all cell values

for one cluster set is set as the new center. The user can specify the number of times this cycle will be repeated. After the cycle has been finished, a signature file is created which contains the mean values per raster band and the covariance matrix for each class. The maximum likelihood classification uses this signature file and the raster sets to create a new raster set where each cell has a value that assigns it to exactly one cluster.

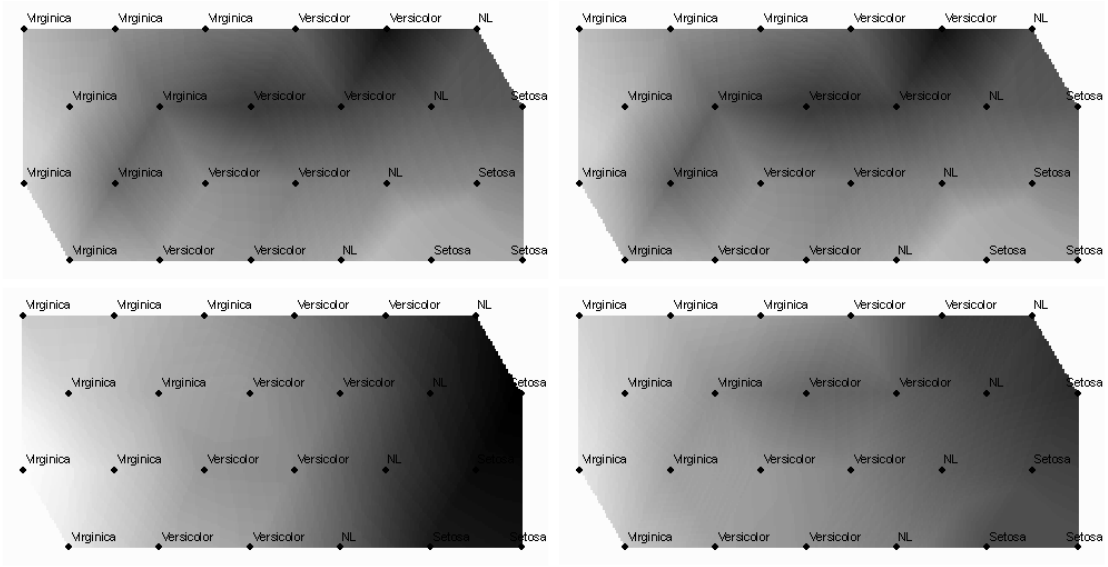

Figure 10: The composite band combinations of the four different component planes. Top left: Combination of the planes 1,2 and 3. Top right: Combination of the planes 1,2 and 4. Bottom left: Combination of the planes 1,3 and 4. Bottom right: Combination of the planes 2,3 and 4. Due to the similarity of the component planes 3 and 4 the two images on the top look similar. Since the values of plane 3 are highly similar to the ones of plane 4, they can be seen as representatives of the values of that plane and vice versa. Therefore it can be argued that the top images show best the value distribution of the SOM.

6 Conclusion

With this work we have shown ways to gain insight into the structure and extension of a SOM by applying operations provided by a GIS on some of its visualization types. We have explained the function and architecture of a SOM and shown different ways to visualize its structure. We have described how we import these visualization types into a GIS and we have proposed several operations to display and thereby emphasize certain of its properties or analyze SOM by statistic means. Since SOM are presented on a two-dimensional display on which a further third variable is displayed, they are well apt to be treated as spatial data. We have shown that it is

possible to treat the visualization types of a SOM as artificial landscapes and that therefore spatial analysis can be applied to this originally non-spatial data. The GIS-driven evaluation of SOM is therefore an enhancement to traditional evaluation approaches.

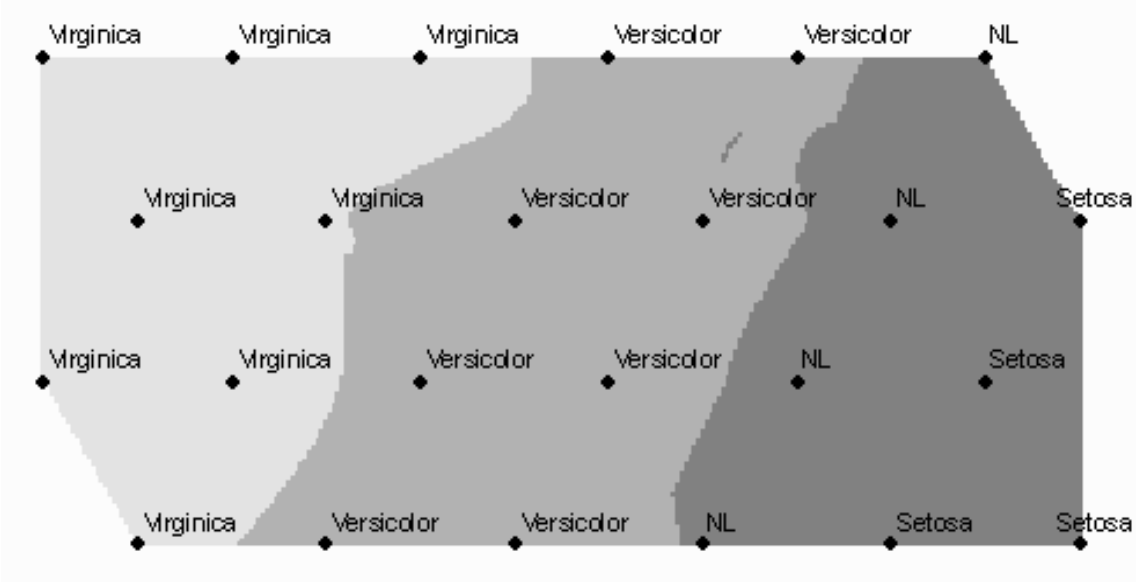

Figure 11: The maximum likelihood classification for the iris flower data set with three classes. Each species is assigned to its own class. Codebook vectors representing Setosa flowers have to share their class with the unlabeled vectors, however.

It must be stressed that the work of this paper is by no means completed. The operations presented here form only a small part of the GIS repertoire. We will continue looking for spatial operations that can deliver information about the SOM. We also will work on making the relation between the output of the spatial operations and its meaning for the SOM input data more evident. One way to do this is to perform spatial operations on the GIS results and provide links back to the input data. To emphasize the usefulness of a GIS for gaining insight into the structure of a SOM, it must be made more evident what the results mean for a distinct input vector. We are convinced that when we have accomplished this, GIS will stand out as one of the best ways to analyze SOM.

7 References

Anderson, E. (1935). "The irises of the gaspe peninsula." Bulletin of the American Iris Society, 59: 2-5.

Bação, F., V. Lobo, and M. Painho (2004). Geo-Self-Organizing Map (Geo-SOM) for Building and Exploring Homogeneous Regions. Geographic Information Science – Third International Conference GI-

Science 2004. M. Egenhofer, C. Freksa and H.J. Miller. Berlin, Springer. 3234.
Cowlishaw, M.F. (1985). "Fundamental requirements for picture presentation." Proceedings Society for Information Display 26 (2): 101-107.
Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). "From data mining to knowledge discovery in databases." Ai Magazine 17: 37-54.
Freiwald, N., Göbel, R., and Jany, R. (2005). Modellierung und Analyse dreidimensionaler Geoobjekte mit GIS und CAD. B. Eitel, H. Gebhardt and P. Meusburger. Heidelberg, Selbstverlag des Geographischen Instituts der Universität Heidelberg.
Guo, D., M. Gahegan, et al. (2005). "Multivariate Analysis and Geovisualization with an Integrated Geographic Knowledge Discovery Approach." Cartography and Geographic Information Science 32(2): 113-132.
Hand D. (1998). Data mining: Statistics and more? The American Statistician 52(2): 112-118.
Jin, H., W.-H. Shum, et al. (2004). "Expanding self-organizing map for data visualization and cluster analysis." Information Sciences 163: 157-173.
Kaski, S. (1997). Data Exploration Using Self-Organizing Maps. Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series 82.
Kaski, S., Nikkilä, J., and Kohonen, T. (2000). Methods for exploratory cluster analysis. Proceedings of SSGRR 2000, International Conference on Advances in Infrastructure for Electronic Business, Science and Education on the Internet. CD-ROM.
Kohonen, T. (2001). Self-Organizing Maps. T. Kohonen. Berlin, Springer.
Kohonen, T., Hynninen, J., et al. (1996). Som pak: The self-organizing map program package. Report A31, Helsinki University of Technology.
Koua, E.L. and M.-J. Kraak (2004). "Geovisualization to support the exploration of large health and demographic survey data." International Journal of Health Geographics 3(1):12
Liebig, W. and R.D. Mummenthey (2005). ArcGIS-ArcView 9 Band 1: ArcGIS-Grundlagen. Norden, Points Verlag.
Lippe, W. (2005). Soft-Computing. Berlin, Springer.
Painho, M., A. Vasilakos et al. (2005). "Exploring spatial data through computational intelligence: a joint perspective." Soft Computing – A Fusion of Foundations, Methodologies and Applications 9(5): 326-331.
Sammon, J.W. Jr., (1969). "A nonlinear mapping for data structure analysis." IEEE Transactions on Computers C-18(5): 401-409.
Sommer, D. and M. Golz (2002). "Detection of wake-sleep transitions with

prototype vector-based neural networks." International Journal of Knowledge-Based Intelligent Engineering Systems 6(3): 129-136.
Tufte, E. (1982). The Visual Display of Quantitative Information. E. Tufte. Connecticut, Graphics Press.
Ultsch, A. (2003). Maps for the visualization of high-dimensional data spaces. Proceedings Workshop on Self-Organizing Maps (WSOM 2003). T. Yamakawa. Kyushu Institute of Technology.
Ultsch, A. (2005). Density estimation and visualization for data containing clusters of unknown structure. Classification; The Ubiquitous Challenge, Proceedings 28th Annual Conference of the German Classification Society (GfKl 2004). C. Weihs and W. Gaul. Berlin, Springer.
Ultsch, A. (2007). Emergence in self organizing feature maps. Proceedings of the 6th International Workshop on Self-Organizing Maps (WSOM 2007).
Vesanto, J. (1999). "Som-based data visualization methods." Intelligent Data Analysis 3(2): 111-126.
Vesanto, J., Himberg, J., et al. (2000): Som Toolbox for Matlab. Technical Report A57, Helsinki University of Technology.
Vesanto, J. and E. Alhoniemi (2000). "Clustering of the self-organizing map." IEEE-NN 11(3): 586-600.
Voronoi, G. (1907). "Nouvelles applications des paramètres continus à la théorie des formes quadratiques." Journal für die Reine und Angewandte Mathematik 133: 97-178.