Clustering of complex shaped data sets via Kohonen maps and mathematical morphology

José Alfredo F. Costa and Márcio L. de Andrade Netto

Department of Computer Engineering and Industry Automation
School of Electrical and Computer Engineering
Universidade Estadual de Campinas
13083-970 – Campinas – SP – BRAZIL

ABSTRACT

Clustering is the process of discovering groups within the data, based on similarities, with a minimal, if any, knowledge of their structure. The self-organizing (or Kohonen) map (SOM) is one of the best known neural network algorithms. It has been widely studied as a software tool for visualization of high-dimensional data. Important features include information compression while preserving topological and metric relationship of the primary data items. Although Kohonen maps had been applied for clustering data, usually the researcher sets the number of neurons equal to the expected number of clusters, or manually segments a two-dimensional map using some *a priori* knowledge of the data. This paper proposes techniques for automatic partitioning and labeling SOM networks in clusters of neurons that may be used to represent the data clusters. Mathematical morphology operations, such as watershed, are performed on the U-matrix, which is a neuron-distance image. The direct application of watershed leads to an oversegmented image. It is used markers to identify significant clusters and homotopy modification to suppress the others. Markers are automatically found by performing a multi-level scan of connected regions of the U-matrix. Each cluster of neurons is a sub-graph that defines, in the input space, complex and non-parametric geometries which approximately describes the shape of the clusters. The process of map partitioning is extended recursively. Each cluster of neurons gives rise to a new map, which are trained with the subset of data that were classified to it. The algorithm produces dynamically a hierarchical tree of maps, which explains the cluster's structure in levels of granularity. The distributed and multiple prototypes cluster representation enables the discoveries of clusters even in the case when we have two or more non-separable pattern classes.

Keywords: Cluster analysis, Data mining, Self-organizing maps, Watershed transform, Knowledge discovery.

1. INTRODUCTION

The size and complexity of data sets is ever increasing. The advances in computer and electronic instrumentation technologies and decreasingly cost of memory storage systems have been enabling large amounts of data to be available in many business, scientific, military and industrial applications. It is greatly important to find methods that could analyze large volumes of data in an unsupervised way. Mining information hidden in unlabeled data sets may confirm initial hypothesis or reveal new patterns that may directly affect the process of decision-making, e.g., in business or in science. How to discover the structure of the data hidden in large data set has been the main task of clustering methods and exploratory data analysis.

The collection of data $X = \{ x_1, x_2, \ldots, x_n\}$ can be represented as a set of points in a multidimensional vector space, $X \subset \Re^p$. The objective of clustering is to find a convenient and valid organization of the data, based on the inherent structure and relationships among the patterns[1]. It may be seen as the process of dividing the p-dimensional pattern space in a way that those patterns in a given cluster are more similar to each other than the rest. The data analyst may be faced with questions like[2]: (*i*) Is there any group or subgroup intrinsic to the data? (*ii*) How many groups are there within the data set? (*iii*) What is the structure of these data? And (*iv*) how can we generate decision rules to classify novel data samples?

Applications to clustering algorithms range from engineering (e.g. pattern recognition) to biology (e.g. classifying plants, taxonomy), from archaeology to social. The number of ways of sorting n observations into k groups is a Stirling number of the second kind which is approximately $k^n/k!$. Finding the best solution is computationally hard when n is large. For example, searching the best partition of 100 patterns in 5 clusters require considering more than 10^{67} partitions. The problem is compounded when k is unknown: the number of possibilities becomes a sum of Stirling numbers.

Correspondence: J.A.F. Costa. P.O. Box 6101, DCA – FEEC – UNICAMP, 13083-970 – Campinas – SP – BRAZIL; E-mail: costa@dca.fee.unicamp.br. Other author information: M. L. A. Netto. E-mail: marcio@dca.fee.unicamp.br.

Clustering methods range from those that are largely heuristic to more formal procedures based on statistical models. Most frequent used methods are hierarchical (or heuristic) and partitioning (or iterative) methods. Good reviews include Ref. 3 and 4. Hierarchical methods have been dominant in the clustering literature and proceed by stages producing a sequence of partitions, each corresponding to a different number of clusters. They can further be subdivided in agglomerative or divisive methods. Agglomerative means that groups are merged at each stage, while divisive methods split one partition at each stage. The major drawbacks of hierarchical methods are: (*i*) undesired merge of objects cannot be corrected at later stages, and (*ii*) in general they require memory usage proportional to the square of the number of groups in the initial partition. Partitioning methods produce one partition with K groups usually by extremizing some objective criterion. The most common method is the K-means that uses heuristics for reducing the within-group sum of squares. Partitioning techniques allow objects to change group membership throughout the cluster formation process. Drawbacks include the choice of the number of groups in advance and the initial K group seeds.

How to efficiently specify the 'correct' or 'well-suited' number of clusters from a given multidimensional data set is one of the most fundamental and unsolved problems in cluster analysis[5-7]. Much of the responsibility of validating the final results is often left to the investigator. Most clustering methods make implicit assumptions about the type of structure present in data. Examples that favour hyperspherical-shaped clusters include partitioning methods such as K-means, and hierarchical techniques such as centroid, Ward, furthest neighbor, while others such as the nearest neighbor method can detect elongated clusters[7-8]. Some partitioning algorithms use heuristics for splitting, merging or deleting clusters. Examples include ISODATA[9] that rely on user supplied thresholds to perform these operations. However, setting parameters without some *a priori* information is not easy. In the case of hierarchical methods, determining the appropriate number of clusters involves selecting one of the steps of the process using a second optimality criterion. Important issues to be considerate include studies in cluster tendency[8] and validation[10].

Artificial neural networks (ANNs) have been widely applied to a range of problems in pattern recognition[11]. ANN implementations have been proposed for clustering problems: Kamgar-Parsi *et al.*[12] employed a Hopfield network simulated on 128x128 SIMD array machine. They concluded that neural networks outperform conventional iterative techniques when there are well-defined clusters. Adaptive Resonance Theory networks have been employed for unsupervised pattern recognition and clustering. The ART1 resembles the *leader* clustering algorithm[8]. Recently Frank *et al.*[13] presented a comparative analysis of some ART networks for clustering.

The self-organizing feature map (SOM) is the most used neural network in the unsupervised learning paradigm[14]. It has been widely studied as a software tool for visualization of high-dimensional data. Important features include information compression while preserving topological and metric relationship of the primary data items. This paper focuses the usage of SOM as a clustering tool and some of the additional procedures required enabling a meaningful cluster's interpretation in the trained map. Visualization of neuron's relations can be done using the Unified distance matrix (U-matrix), which is a neuron-distance image. However, automatic U-matrix segmentation is very problematic due its many local *minima* and borders separating clusters may not be well defined. It is proposed methods for automatic segmentation of maps, generating clusters of neurons. Topics discussed here include the usage of mathematical morphology segmentation method watershed to segment the U-matrix. Finding good watershed markers and the U-matrix homotopy modification are discussed. The algorithm automatically produces labeled sets of neurons that are related to the clusters in the p-dimensional input space.

The process of map partitioning is extended recursively. Each cluster of neurons in one map generates a new map (son network), which are trained with the subset of data that were classified to it. The algorithm produces dynamically a hierarchical tree of maps, which explains the cluster's structure in levels of granularity. A coarse-to-fine representation of clusters is obtained as the data sets are partitioned. Sub-networks in higher levels of the hierarchy detail the underlying structure obtained on a lower level. Its is also possible to prune (delete) sub-maps. In this case, the subset of data can be represented only by the clusters of neurons obtained in the last level of the hierarchy that has a map with more than one cluster. Therefore, only networks with more than one cluster are stored in memory. Not only the hierarchy can detect clusters and sub-clusters but also the distributed and multiple prototypes cluster representation enables the discoveries of clusters, even in the case when we have two or more non-separable pattern classes.

The remainder of the paper is organized as follows. Section 2 describes briefly the basic SOM model and its visualization tool: the U-matrix. Section 3 discusses image segmentation methods and the morphological method watershed. Section 4 presents the algorithm for automatic partitioning of trained SOM networks while section 5 extends the algorithm to enable the dynamical generation of a hierarchy of maps. Experimental results are shown in section 6. Finally, section 7 summarizes the paper with conclusions and final remarks.

2. THE SELF-ORGANIZING MAP

The SOM algorithm is one of the best known artificial neural network algorithms. SOM is based on unsupervised learning and it constructs a topology-preserving mapping of the training data where the location of a unit carries semantic information[11, 14]. The SOM consists essentially of two layers of neurons: input-layer I and the unit layer U. All components of such an input vector are fed into all neurons of the input layer. The SOM defines a mapping from the high dimensional input data space onto a regular, usually, two-dimensional array of nodes. Each neuron *i* of the SOM is represented by an *p*-dimensional weight vector $m_i = [m_{i1}, m_{i2}, ..., m_{ip}]^T$, where *p* is equal to the dimension of the input vectors. The neurons of the map are connected to adjacent neurons by a neighborhood relation dictating the structure of the map. In the 2-dimensional case the neurons of the map can be arranged either on a rectangular or hexagonal lattice.

Training is accomplished by presenting one input pattern *x* at a time in a random sequence and comparing, in parallel, this pattern with all the reference vectors. The best match unit (BMU), which can be calculated using the Euclidean metric, represent the weight vector with the greatest similarity with that input pattern. Denoting the winner neuron by *c*, the BMU can be formally defined as the neuron for which

$$\| \boldsymbol{x} - \boldsymbol{m}_c \| = \min_i \left\{ \| \boldsymbol{x} - \boldsymbol{m}_i \| \right\} \tag{1}$$

where ||·|| is the distance measure. The input is thus mapped to this location. The weight vectors of BMU as well as the neighboring nodes are moved closer to the input data vector. The magnitude of the attraction is governed by the learning rate. The SOM update rule for the weight vector of the unit *i* is

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + h_{ci}(t) \cdot [\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] \tag{2}$$

where *t* denotes time, *x*(t) is the input vector randomly drawn from the input data set at time *t* and $h_{ci}(t)$ is the neighborhood kernel around the winner unit *c* at time *t*. This last term is a non-increasing function of time and of the distance of unit *i* from BMU and usually is formed of two components: the learning rate function $\alpha(t)$ and the neighborhood function *h*(*d, t*):

$$h_{ci}(t) = \alpha(t) \cdot h(\| \boldsymbol{r_c} - \boldsymbol{r_i} \|, t) \tag{3}$$

where r_i denotes the location of unit *i* on the map grid.

As the learning proceeds and new input vectors are given to the map, the learning rate and the neighborhood radius gradually decreases to zero according to the specified functions. The correct choice for parameters is not a straightforward task and there are several *rules-of-thumb*, found through experiments. After the training has been performed, the map should be topologically ordered. This means that vectors that are close in the input space would be mapped onto neighbor neurons or even in the same neuron. Some recent improvements include linear initialization and its parallel implementation, or batch algorithm[14]. In this approach the updating step replaces each map unit by the average of the data vectors that were in its neighborhood only in the end of each epoch. Similarly to the traditional SOM learning, the neighborhood function can also weigh the contribution of surrounding neurons to the calculation of the mean vectors.

2.1 VISUALIZING NEURON'S RELATIONS THROUGH THE UNIFIED DISTANCE MATRIX

The U-matrix method was developed by A. Ultsch[15] to enable visualization of the topological relations of the neurons in an organized SOM. The basic idea is to use the same metric that was used during the learning to compute distances between adjacent reference vectors. These distances are then visualized in a 3D-plot in which ‘valleys’ correspond to map units that are similar. High values in the U-matrix encode dissimilarities between neurons and correspond to cluster borders. The following notation was adapted from Refs. 15 and 16 which contain a more detailed presentation of the method.

In a rectangular grid, each neuron *u* possesses the eight immediate neighbors. Associated with each unit is an *n*-dimensional reference vector. The same metric used to find the BMU is used to determine a ‘*distance*’ between *u* and its neighbors. Consider a two-dimensional SOM with a rectangular lattice topology of size ($X \times Y$). Let $[b_{x,y}]$ be the matrix of neurons and $[w_{i_{x,y}}]$ be the matrix of weighs. For each neuron in *b* exist three distances d_x, d_y and d_{xy} in the U-matrix with d_x = e, $d_y = n$ and d_{xy} = 0.5 * (*ne* + *nw*), where *e, n, ne* and *nw* denotes the *east*, *north*, *northeast* and *northwest*-distance of a neuron to their neighbors. The U-matrix combines these three distances into one matrix U of size $(2X\text{-}1) \times (2Y\text{-}1)$. For each unit of b, the distances to the neighbor (if these units exist) become dx, dy and dxy. The components of the U-matrix take values as

$$\begin{bmatrix} du(0,0) & dx(0,0) & du(1,0) & \cdots & du(X\text{-}1,0) \\ dy(0,0) & dxy(0,0) & dy(1,0) & \cdots & dy(X\text{-}1,0) \\ du(0,1) & dx(0,1) & du(1,1) & \cdots & du(X\text{-}1,1) \\ dy(0,1) & dxy(0,1) & dy(1,1) & \cdots & dy(X\text{-}1,1) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ du(0,Y\text{-}1) & du(0,Y\text{-}1) & du(1,Y\text{-}1) & \cdots & du(X\text{-}1,Y\text{-}1) \end{bmatrix}$$

Intermediary values $du(x, y)$ can be calculated by using the mean or the median value of their surrounding elements.

3. IMAGE SEGMENTATION AND THE MATHEMATICAL MORPHOLOGICAL APPROACH

3.1 IMAGE SEGMENTATION

Image segmentation consists of partitioning an image into meaningful, non-intersecting, regions of interest [2,17]. These regions are homogeneous with respect to one or more signal or structural property such as brightness, color, texture, context etc. Segmentation techniques rely on two broad categories: contour-based methods and region-based methods. The first approach looks for the local gray level discontinuities in the image and the second one seeks for parts of the image which are homogeneous in some measurable property such as gray levels, contrast or texture. Contour-based methods, e.g. “snakes” or elliptical Fourier series, attempt to trace the edges of regions by following a maximum gradient around the object until they reach the starting point. The edge and all adjacent pixels are included in the region. Two common ways of obtaining the edge information are calculating approximations of the first derivative, where large values are indicative of edges, or the second derivative, where zero-crossings occur at edges.

Conversely, region-based methods incorporate the notion of connectivity that pixels are likely to be part of the same distinct region if they are connected and above a threshold value. Connectivity within regions can be defined:

Definition 1. x_i in a region R is connected to x_j if and only if there is a sequence of $\{x_i, \ldots , x_j\}$ such that x_k and x_{k+1} are connected and all the nodes are in R.
Definition 2. R is a connected region if the set of points x in R has the property that every pair of nodes is connected.

The set of regions (R_k) is known as a partition when the entire image is segmented ($\bigcup^{m}_{k=1} R_k$). Each region is generally a homogeneous area satisfying some criteria, $H(R_k)$ = true, where H is a Boolean function, and $\mathrm{H}(R_i \cap R_j) = \varnothing$, $i \neq j$. Therefore, a region can be defined as a set of pixels in which there is a path between any pair of its pixels, and all the pixels in the path are also members of the set.

3.2 MATHEMATICAL MORPHOLOGY AND THE WATERSHED TRASNFORM

Mathematical Morphology (MM) is a general theory that studies the decompositions of operators between complete lattices in terms of some families of simple operators: dilations, erosions, anti-dilations and anti-erosions. It provides a set of powerful tools for texture analysis and has been used in a number of image processing applications since its development in the late [18-19]. Unlike classical linear processing, MM belongs to the nonlinear branch of image and signal processing and has been used in many applications of image analysis. Although originally developed over binary images, morphological paradigms are being extended into various domains even beyond images such as graphs and symmetry groups[18].

The watershed transform is the primary tool of mathematical morphology for image segmentation. In MM, it is usual to consider that an image is a topographical surface. Places of sharp changes in the intensity thus make a good set in which one can search for contour lines. For the purpose of segmentation, we then look for the crest lines of the gradient image. A way to characterize these lines is to apply the watershed algorithm to the modulus of the gradient image. The idea of the watershed algorithm [20-21] is to associate an influence zone to each of the regional minima of an image (connected plateau from which it is impossible to reach a point of lower gray level by an always descending path). We then define the watershed as the boundaries of these influence zones. The watershed of a surface of the image has several useful properties: The watershed lines form closed and connected regions; The intersection of these regions is null; Union of these regions and the watershed lines separating them makes the whole surface; Each region contains a single regional extrema (usually minimum) as a single point or region); and each regional extrema contains a single catchment basin (watershed basin).

Watersheds can be seen as a hybrid technique that combines both contour and region growing approaches for image segmentation. Regional minima are usually used as region markers, and when we have a noisy image, which is generally the case of U-matrices, direct application of watersheds leads to a oversegmented image. This drawback can be preventing by using an homotopy image modification in which the kernels of clusters are assigned to its proper valleys [2, 21-22]. The oversegmentation produced by the coarse application of the watershed is due to the fact that each regional minimum give rise to a catchment basin. However, all the catchment basins do not have the same importance. There are important ones, but some of them are induced by the noise, others are minor structures in the image. Numerous techniques have been proposed to compute the watershed [20,21]. For example, Vincent and Soille's approach is based upon an immersion process analogy in which the flooding of the water in the picture is efficiently simulated by a queue of pixels [23]. A very short description of mathematics of watershed is presented here. Detailed descriptions include Refs. 20-23.

Let Z be the set of integers, and let E be a rectangle of Z^2, representing a subset of the square grid, and let K be a interval $[0, k]$ of Z, with $k > 0$. A gray-scale image is any function from E to K. Then, for a $x \in E$, we can represent an image as $f(x)$. The watershed equation for regions can be briefly written as

$$\psi_{B_c,g}(f)(x) = i \mid L_{B_c,f}(x, i \leq g \leq i) < L_{B_c,f}(x, j \leq g \leq j), \tag{4}$$

$\forall i \neq j$, where f is the gray-scale image, g is the marker image (which may be gray-scale or binary image), B_c is the structuring element (directly related to the watershed connectivity),

$$L_{B_c,f}(x, X) = \min\{\eta_f[\pi_{B_c}(x, X)] : \forall \pi_{B_c}(x, X)\}$$

is the minimum length of a point to a set,

$$\eta_f\{\pi_{B_c}(x, X)\} = \max\{f(y) : y \in \pi_{B_c}(x, X)\}$$

is the length of a point to a set, and $\pi_{B_c}(x, X)$ is the path between the point x and subset X under connectivity B_c. The watershed creates the image y by detecting the domain of the catchment basins of f indicated by g, according to the connectivity defined by B_c.

4. AUTOMATIC PARTITIONING AND LABELING TRAINED SOM NETWORKS

Given a trained SOM, the general approach for its automatic partitioning and labeling using watershed transform could be summarized as follows[2]:

1) Obtain the U-matrix using the trained map (see subsection 2.1).

2) Find the image markers, i.e., one connected component for each cluster of neurons and one connected component for the background (see next subsection);

3) Impose the new regional minima by modification of image homotopy (or gray-scale geodesic reconstruction) [18, 21, 22];

4) Compute the watershed lines (equation 4);

5) Assign a different label for each connected region (cluster of neurons) of the U-matrix [2,17].

6) Copy the U-matrix labels to the corresponding neurons in the map.

Instead of using the resulting codebook after training a better U-matrix could be obtained if we eliminate the effects of link neurons, i.e., neurons without associated patterns. Let N the number of neurons in the map and H the number of patterns mapped onto unit i, *i.e.*, the *hits* histogram. A minimal activation parameter can be set which could establish the minimum number of patterns in which a given neuron will not be moved. The general approach for obtaining a new codebook could be stated as follows:

```
For $i = 1, 2, \dots, N$
    If $H(i) \leq \varphi$
        $\boldsymbol{m}_i \leftarrow \boldsymbol{m}_j$, where $\| \boldsymbol{m}_i - \boldsymbol{m}_j \| < \| \boldsymbol{m}_i - \boldsymbol{m}_k \|$, $k = 1, 2, \dots, N$, $k \neq j$, $k \neq i$, and $H(j) > \varphi$.
    End-If
End-For
```

Note that the weight vector $\boldsymbol{m_j}$ replaces $\boldsymbol{m_i}$ in case of $H(i) \leq \varphi$. In our experiments this operation was performed only to obtain a better view (or a enhanced) U-matrix. In our case $\varphi = 1$ but greater values may be need (see the comments about bridge between two clusters in section 6) meaning that only units with at least one associated pattern remains unchanged. The greater φ, the stronger pattern quantization influence in the resulting U-matrix. In a two-dimensional map, the index i can be replaced accordingly, i.e., (i, j).

Regarding the step 6 of the above algorithm, if there remains unlabeled neurons in the map, which may occur if the corresponding pixel in the U-matrix is in the border of two or more clusters, two strategies can be done. These neurons can be classified by K-nearest neighbor neuron region to the labeled neurons by using distances in the input (weigh) space. Conversely, we can leave these neurons unlabeled. If a given pattern is mapped onto this unlabeled neuron, we can proceed by seeking the second BMU. If this neuron is also unlabeled, seek the next BMU and so forth. This last approach was used in the simulation examples presented section 6.

4.1 FINDING WATERSHED MARKERS FOR U-MATRIX

This section will present simple but efficient approach that have been leading to good results in a wide variety of cases[23,24]. Given the U-matrix image U, the following steps are performed[2]:

1. Filtering: create the image U_1 by removing any pore with area less or equal than three pixels.

2. For $k = 1$ to f_{max}, where f_{max} is the highest gray level of the image {

   2.1 Create the binary image U_1^k that corresponds to the conversion U_1 to a binary image using as threshold k.

   2.2 Obtain N_{cr}^k, the number of connected regions of U_1^k. }

3. Obtain the most persistent value of number of regions η (clusters of neurons) that corresponds to the highest (and significant) plateau in the plot of N_{cr}^k versus k.

4. Take as image marker the binary image U_1^ν, where ν is the initial value k of the plateau chosen in the previous step.

The filtering in step 1 smoothes very weakly the image. It can be performed applying the morphological operator erosion followed by a dilatation with structuring element of radius ρ. The bigger ρ the stronger filtering. The value 'three pixels' was chosen only for attenuating only minor structures in the image resulted from the noisy characteristics of U. Visually, the filtered image U_1 is almost the same of U. The step 2 performs a slicing for all the gray levels in U-matrix. Each binary image has a number of connected regions that is stored in the vector N_{cr}. Plotting N_{cr}^k versus k shows how changes the number of connected regions, or seeds for clusters of neurons, as the gray level or neuron-distance increases (k). The algorithm seeks for significant plateaus of N_{cr} in a useful range of gray level values. This approach is related to the scale-space theory toward find the well-suited number of clusters which cluster validity is performed by examining and comparing the survival period of clusters when some clustering parameter are varied[25]. Other approaches to find image markers were performed[2] and include: *a*) images after area opening the regional minima of U, with a increasing value of area; *b*) residues from the sup-reconstruction of the image U from the marker image created by the addition of a increasing value H to the image U; and *c*) other more elaborated methods[2].

5. DYNAMIC GENERATION OF A HIERARCHY OF MAPS

Neural tree networks have been proposed with the aim to expand the taxonomic relationships of the groups found. Examples include the dynamic neural tree networks[26-27] and competitive evolutionary neural tree [28]. Common among self-development networks is the usage of heuristics to grow and / or prune nodes in response to the input data and a complementary set of parameters, which are generally set in advance and greatly influence the final results. Differently of other hierarchical SOM approaches, the method presented here does not specify the number of sub-networks for a given SOM in a given height of the tree, which is automatically determined as seen in section 4. Other parameters such as size of sub-networks are functions of the parent network parameters as well as the size of subset of data that will be forwarded to the next tree levels.

The aim of the hierarchical approach is to expand the clustering discovery process enabling sub-clusters to be detected in child maps of segmented neuron clusters[29]. The process can be viewed as a recursive partitioning method, which the data are subdivided in subgroups according their classification by the SOM labeled regions and these subsets of data are used to train high-level maps. Differently of other hierarchical SOM (or competitive neural networks) methods[26-28, 30, 31], all nodes in the tree are SOM networks instead of units, and the nodes generating child maps are regions (clusters of neurons) segmented as shown in section 4. Therefore, each cluster of neurons acts as a data filter subdividing items to different child networks according to the label of its best-match unit. The process can be seen as a dynamic strategy for cluster, and sub-clusters, discovery.

Similar to the conventional SOM training, the initial parameters such as map size and dimension, neighborhood type, function types for $\alpha(t)$ and $h(d, t)$, maximum number of epochs and the way of weight initialization. The algorithm for generating the hierarchy of maps is performed sequentially starting from the top-level map as follows:

Algorithm for generation of hierarchy of labeled self-organizing maps

1. Set current training level to $\eta = 0$. The data set $\boldsymbol{X}^{\eta}$ for $\eta = 0$ is in fact the original data set $\boldsymbol{X}$.

2. Train map M^{η} with data set $\boldsymbol{X}^{\eta}$.

3. Let K^{η} the number of regions (clusters of neurons) identified in map M^{η}. Label each region according the algorithm presented in section 4 and classify all data items of $\boldsymbol{X}^{\eta}$. The result is a partition of $\boldsymbol{X}^{\eta}$ in K^{η} subgroups $\boldsymbol{X}(K^{\eta}; M^{\eta})$.

4. Each labeled region K^{η} of map M^{η} generate a sub-map at level $(\eta + 1)$ subordinated to its father map and its generating region, which we call $M^{\eta+1}(K^{\eta})$. The size of this sub-map is made proportional to the size of its father map and the subset of data $\boldsymbol{X}(K^{\eta}; M^{\eta})$ which will be used for its training. Let ζ the size of the father map and q_k the ratio between the number of patterns falling in the subset $|\boldsymbol{X}(K^{\eta}; M^{\eta})|$ by the total number of patterns in the father map, $|\boldsymbol{X}^{\eta}|$. Therefore, $q_{K^{\eta}} = \left|\boldsymbol{X}(K^{\eta}; M^{\eta})\right| \Big/ \left|\boldsymbol{X}^{\eta}\right|$, where $|\cdot|$ represents the cardinality of the set. The size of the child map $M^{\eta+1}(K^{\eta})$ can be stated as

$$\zeta_{K^{\eta}}^{\eta+1} = (q_{K^{\eta}})^{\beta} \cdot \zeta^{n}$$

where the value used for β was 0.3.

5. Train each map $M^{\eta+1}(K^{\eta})$ in level η with the subset of data $\boldsymbol{X}(K^{\eta}; M^{\eta})$. Segment and label clusters of neurons of map $M^{\eta+1}(K^{\eta})$, as was done for the root map, and generate subsets of data.

6. If a given sub-map does not generate a minimum number of two regions prune it by removing it from the tree. The cluster of neurons in its father map can ready represent the subset of data used for train that sub-map.

7. Repeat steps 4-6 until tree stabilization, i.e., when it is not possible add or delete sub-maps.

Note that training is performed locally in each map (sub-map) with a smaller data set than used in its father map. In step 4, the choice of that function for the size of child map aims that the size of sub-maps decrease as the height of the tree η increases. This size of sub-map is calculated using the proportion of patterns falling in each cluster of neurons. The value $\beta = 0.3$ was motivated by the reason that although its important decrease the size of sub-maps, this may be done slowly. Working with very small maps will generate small U-matrices and we also must care the effects of neurons lying in the periphery of the map. Although β had been empirically chosen after diverse simulations with different data sets, this parameter is not critical, and we can even choose for the size of sub-maps the same size of its father map. Reason for decreasing the map size includes reduction of memory structures storage.

Each map in the same height η is independent regarding other maps in the same level and η is incremented only when all maps of this level have been trained, segmented and labeled. Conversely this can be done by focalizing attention in each branch of the tree until its finish, like a pre-order search. Regarding step 6, a given sub-map will be kept in memory only when it generates more than one cluster. Otherwise the cluster of neurons in its father map can ready represent the subset of data used for train that sub-map. It occurs when a given region presents only one cluster, as when we have Gaussian or Poisson processes[2]. Other increments to the method include the analysis of viability in searching for clusters in a given U-matrix its normalized and cumulative histograms, *$H(U)$ and $H_c(U)$*, respectively[2,29]. Usually if there are well-defined clusters within the data, most of the U-matrix pixels will be in valleys (low values), while the lower fraction of pixels will be with cluster's borders. It is expected that $H(U)$ to be a left-concentrated histogram, which makes $H_c(U)$ to appears as log-shaped function. On the other hand, the absence of more than one cluster implies in a sigmoid-shaped $H_c(U)$[2]. These and other signs can be combined for generating rules for automatic tree generation.

6. EXPERIMENTAL RESULTS

Most clustering methods make implicit assumptions about the type of structure present in data. Examples that favour hyperspherical-shaped clusters include partitioning methods such as *K*-means, and hierarchical techniques such as centroid, Ward, furthest neighbor while others such as the nearest neighbor method can detect elongated clusters (chaining effect) [7-8].

A data set was generated for testing the capabilities of the proposed algorithms. It resembles the data clustering example presented in figure 1 of Ref. 8. The aim is to show that even in presence of very different pattern structures, the hierarchical SOM approach will be able to detect and separate them. Figure 1 presents the data set, which comprises of 834 patterns in two-dimensions, for easy of visualization of the results. The number of patterns in each class (1 to 8) is 314, 100, 100, 100, 100, 10, 53, 57, respectively. Classes 2 to 5 were generated by multivariate Gaussian probability distributions. Class 2 is completely enclosed by a circular cluster (class 1). This last is connected to class 5 by a bridge (class 6), which is a chain of intermediate objects. Actually this bridge would not be regarded as a cluster. We emphasized it with different labels for easy of understanding the results. Classes 7 and 8 are elongated (cigarette) clusters. Conventional methods such as *K*-means are unable to identify all these clusters in a single data set. Although the class labels were known this information was not used in the clustering algorithms. Its purpose was only for visualization.

All experiments used two-dimensional SOMs. The size of root SOM network was set 15×15 (i.e., 225 neurons in a two-dimensional lattice). Weight initialization was linear and the training was done with the batch algorithm[14]. The neighborhood function was Gaussian and during ordering the neighbor radius decreased with the training epochs, which in all cases were set 500. The simulations were performed in a 300 MHz PC compatible using *Mathworks Matlab.* Some functions of the *SOM toolbox* were also used (available at http://www.cis.hut.fi/projects/).

Figure 2 illustrates the root map and data after training. The filled and open circles represent the data and the neurons, respectively. Lines connecting neurons express their immediate neighborhood. Figure 3 shows the hits histogram. The size of the black square is proportional to the number of patterns won by a neuron. It is easy to see from figures 2 and 3 that the self-organizing map concentrates neurons in areas with high pattern density. Otherwise, some neurons remain with zero patterns, which we call *link neurons*. Following the strategy delineated in section 4 we can decrease the effect of these link neurons in some analysis of the maps. The result is shown in figure 4.

Figure 5 shows a planar view of the U-matrix for the map obtained using grid of figure 4, while figure 6 shows its 3D view. Distances between neighboring neurons are encoded as z-axis or the pixel's gray level value, as discussed in section 3. The main difficulty in segmenting the U-matrix is that borders separating clusters usually may have some weak portions, which causes two different clusters to be merged in the automatic process. The usage of watershed is motivated toward finding the optimal (or well-suited) cluster's borders for the obtained cluster markers.

Although the U-matrix gives visualization and insights of the clustering structure, the automation of knowledge discovery is not straightforward. The main questions here are about the number of clusters and what neurons share a common group. The algorithm presented in section 4 determines the well-suited number of clusters for a given U-matrix and the corresponding image marker. It uses information like the number of connected regions (N_{cr}^k) for each gray level of the image U_1^k for a useful gray level range (figure 7), which in turn is related to the distances between neighboring neurons. Following the algorithm, the system decided automatically the number of clusters (3). The image markers for this map are presented in figure 8. Markers act as seeds for the watershed transform and were obtained after thresholding the U-matrix by the lowest gray level k from largest contiguity range of N_{cr}^k. The segmented U-matrix is illustrated in figure 9 while figure 10 shows the corresponding labeled SOM. Note that although a 7-cluster solution would be the optimal, disregarding class 6 (bridge), the algorithm detected 3 clusters in the root map. The result is correct, in some way, because it was identified three macro-regions in the space with high density of patterns separated by areas with low density of points.

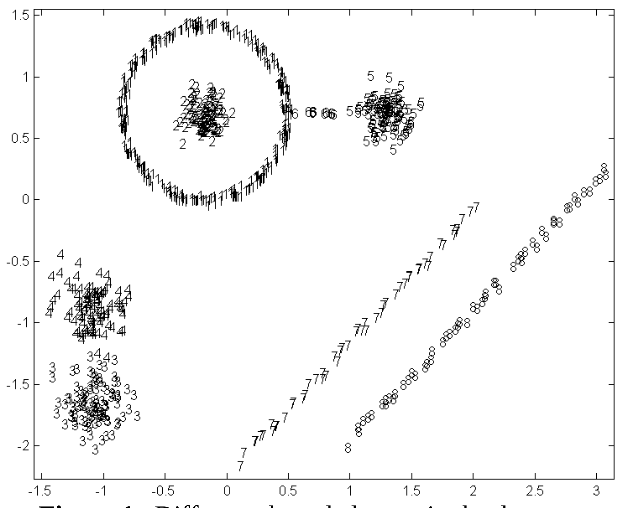
Figure 1. Different shaped clusters in the data set.

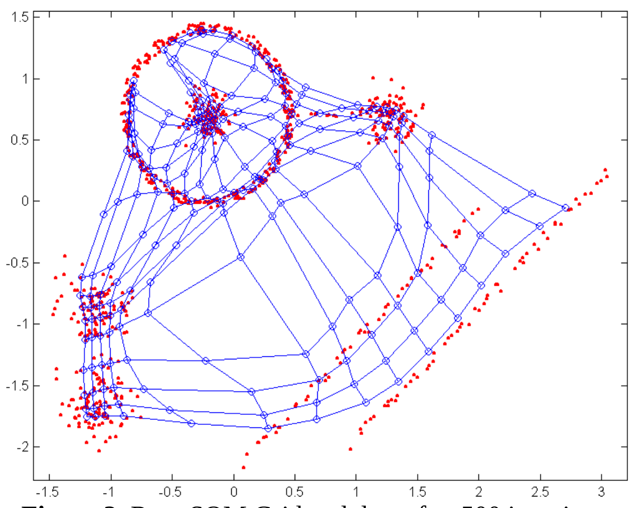
Figure 2. Root SOM Grid and data after 500 iterations.

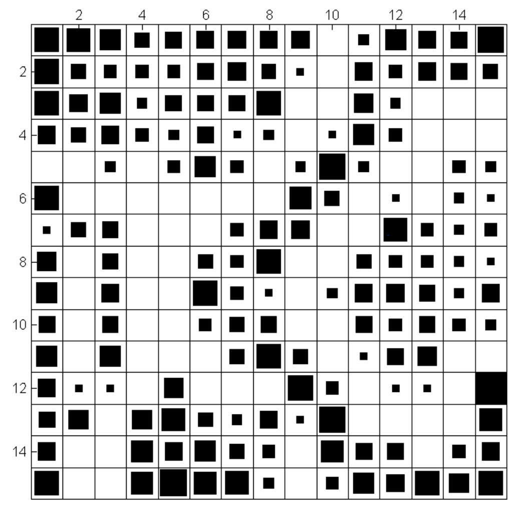
Figure 3. Hits histogram for the root SOM network.

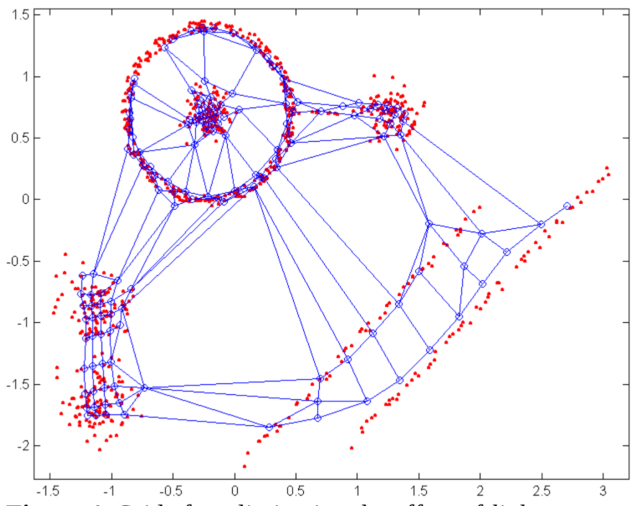
Figure 4. Grid after eliminating the effect of *link neurons*.

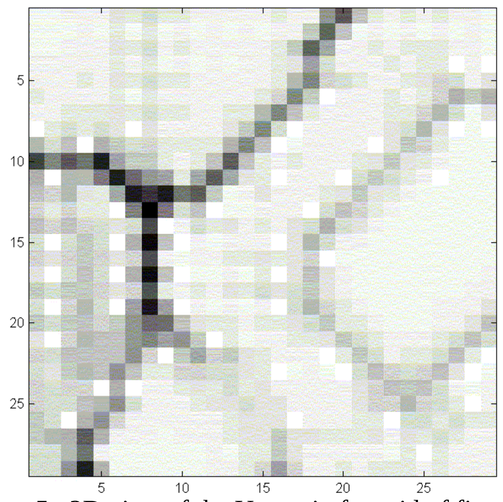
Figure 5. 2D view of the U-matrix for grid of figure 4.

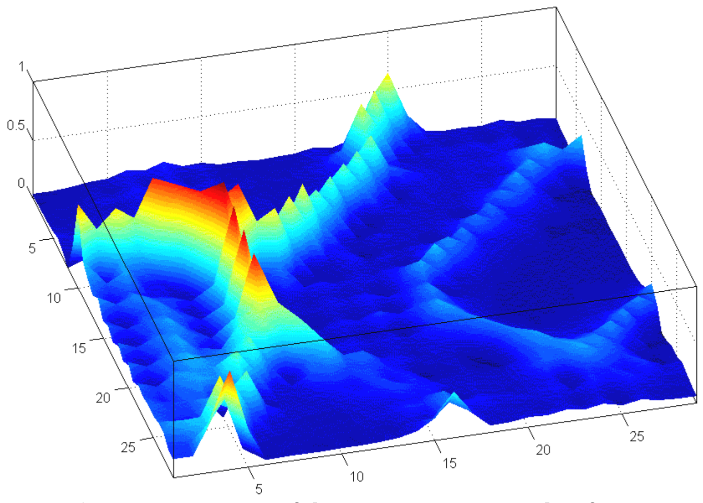
Figure 6. 3D view of the U-matrix presented in figure 5.

The resulting hierarchy of maps is presented in figure 13. Shades of gray differentiate clusters of neurons. Also is indicated the original class of patterns mapped onto neurons. In some cases patterns from different classes were mapped on the same neuron. For example: in the root map, the neuron with output indexes (4,4) mapped individuals from class 3 and 4. The subnetwork *3* divided patterns from class 6 (bridge) in two separate clusters that dominated by class 1 and 5. It can be seen from the rightmost picture of figure 12 that the chaining patterns connecting class 1 and 5 affected the border separating the correspondingly cluster of neurons. However, even in the presence of this bridge, the clusters were correctly identified. Otherwise, if the bridge have a significant number of patterns, i.e., if the density of points is high, more SOM neurons will be allocated to that area, and clusters may not be separated. In this hypothetical case, the result could be regarded also as correct, and we would have a shaped cluster that would involve class 1, 5 and 6.

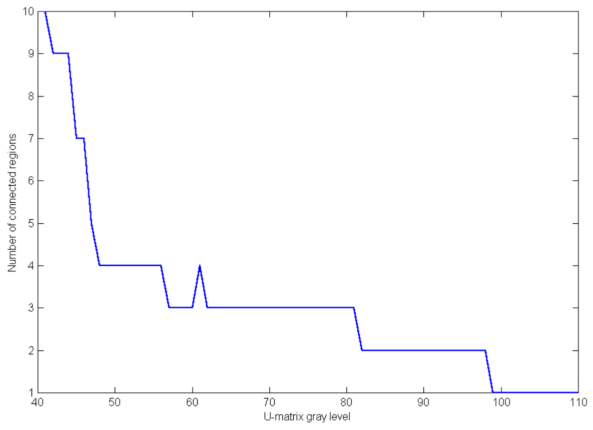

Figure 7. Number of connected regions versus image threshold.

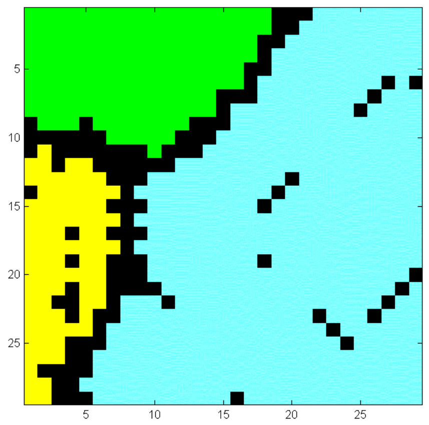

Figure 8. Detected markers (labeled regions).

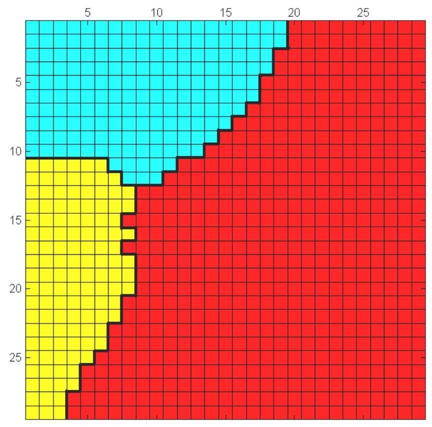

Figure 9. Segmented U-matrix after watershed transform.

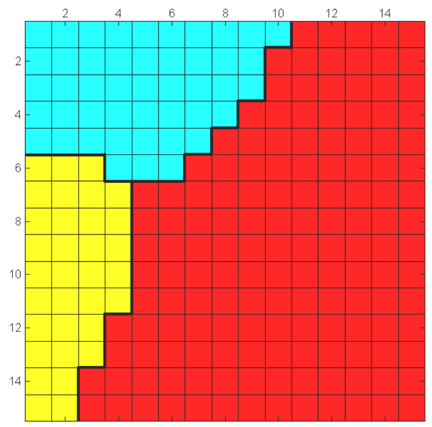

Figure 10. Labeled SOM after U-matrix segmentation.

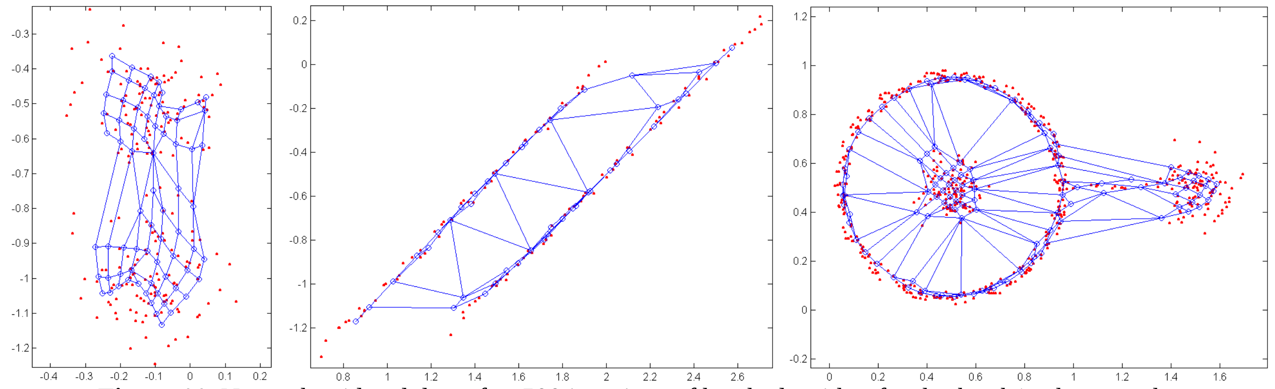

Figure 11. Network grid and data after 500 iterations of batch algorithm for the *level 1* sub-networks. From left-to-right, sub-networks 1, 2 and 3.

Following the hierarchical SOM algorithm (section 5), each cluster of neurons generates a new map, which are trained with the subset of data that were classified to it. Three sub-networks were generated with 9×9, 8×8 and 13×13 neurons, respectively. The sizes of sub-networks are related to the size of the father network and the proportion of data classified by its generating cluster of neurons. Figure 11 shows the network's grid and data for the level *1* generated sub-networks. The corresponding U-matrices are shown in figure 12. Applying the same process to the clusters of neurons obtained in the *level 1* sub-networks, the derived sub-maps (level 2 in the tree) were pruned because they did not generated new clusters of neurons. Thus, the subset of data remains represented by the cluster of neurons in the father's map and there is not need to store in memory unnecessary maps.

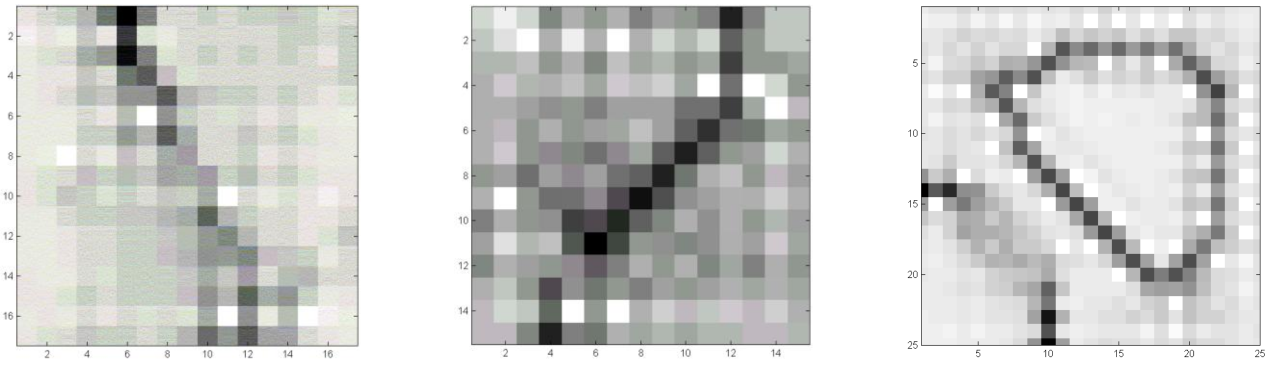

Figure 12. U-matrices derived from sub-networks 1, 2 and 3 (respectively from left-to-right).

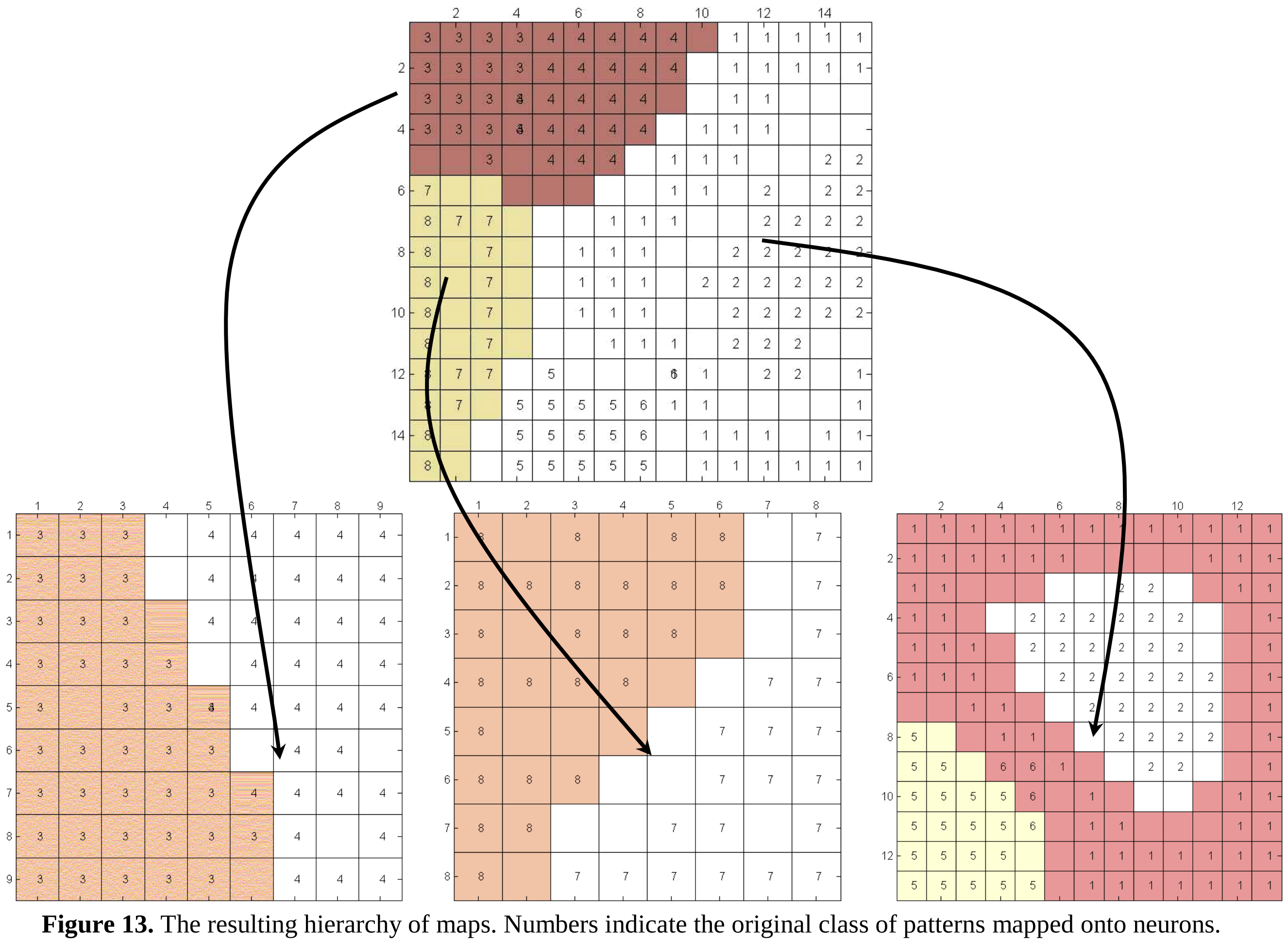

Figure 13. The resulting hierarchy of maps. Numbers indicate the original class of patterns mapped onto neurons.

7. CONCLUSION AND FINAL REMARKS

Two algorithms were presented in this paper for allowing automatic partition of trained self-organizing maps and for generating a hierarchy of maps based on the detected data clusters. Main in using SOM include information compression and density estimation while trying to preserve topological and metric relationship of the primary data items. SOM is robust regarding the weigh vectors initialization[14] but setting training parameters is not straightforward. The *U*-matrix reflects in an image the configuration obtained through unsupervised learning. Therefore, all these analysis suppose that training has been well accomplished. How to guarantee that the obtained configuration of neurons after *t* epochs is already useful for performing all these analysis is still an open question.

The method was applied to a diversity of data sets, synthetic and real ones, and in most of the cases it performed better or equal than other competitive tree networks[2,28]. Regarding traditional clustering methods, such as *K*-means, and the conventional SOM, in these methods the number of clusters have to be set in advance, and if we use a single prototype (or neuron) for representing the cluster structure, we will retrieve only hiperspherical shaped clusters. This lead to errors in recovering the inherent structure of the clusters. Instead of this, our method used a scale-space approach in which clusters in levels of granularity were detected. The result (fig. 13) could be interpreted as: there are three macro-clusters in the data. The first and the second have two sub-clusters while the third presents three sub-clusters. Although the results were presented for a synthetic data set in two-dimensional space, higher-dimensional data sets and higher-dimensional maps demonstrated its efficiency. The advantages of using the proposed algorithms include the automatic determination of the number of clusters, a tool for explain the cluster and its sub-clusters structure, and the fact that the discovered clusters can have a non-parametric geometry enabling complex shaped clusters to be detected. Examples of different shaped data sets showed the efficacy of the methods [2, 23-24, 29].

8. REFERENCES

1. V.V. Vinod, S. Chaudhury, J. Mukherjee, and S. Ghose. A connectionist approach for clustering with applications in image analysis. *IEEE Trans. Systems, Man & Cybernetics*, 24 (3), 356-384. 1994.
2. J.A.F. Costa. *Automatic classification and data analysis by self-organizing neural networks*. Ph.D. Thesis. State University of Campinas, SP, Brazil.1999.
3. B.S. Everitt. *Cluster Analysis*. Wiley: New York. 1993.
4. L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley: New York. 1990.
5. M.-C. Su, N. DeClaris and T.-K. Liu. Application of neural networks in cluster analysis. *Proc. of the 1997 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, pp. 1-6. 1997.
6. R. Kothari, and D. Pitts. On finding the number of clusters. *Pattern Recognition Letters.* Vol. 20, pp. 405-416, 1999.
7. A. Hardy. On the number of clusters. Computational Statistics and Data Analysis, 23, pp. 83-96. 1996.
8. A.K. Jain, M.N. Murty and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31 (3), pp. 264 – 323. 1999.
9. G. Ball and D. Hall. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12, pp. 153-155. 1967.
10. J.C. Bezdek and N.R. Pal. Some new indexes of cluster validity. *IEEE Transactions on Systems, Man, and Cybernetics* (Part B), 28, pp. 301-315. 1998.
11. S. Haykin. *Neural Networks: A Comprehensive Foundation*. 2nd edition, Prentice-Hall: New York. 1999
12. B. Kamgar-parsi, J.A. Gualtieri, J.E. Devaney and B. Kamgar-parsi. Clustering with neural networks. *Biological Cybernetics*, 63, pp. 201-208. 1990.
13. T. Frank, K.-F. Kraiss, and T. Kuhlen. Comparative analysis of fuzzy ART and ART-2A network clustering performance. *IEEE Trans. on Neural Networks*, 9, pp. 544-559. 1998.
14. T. Kohonen. *Self-Organizing Maps*, 2nd Ed., Springer-Verlag: Berlin. 1997.
15. A. Ultsch. Self-Organizing Neural Networks for Visualization and Classification. In: O. Opitz et al. (Eds). *Information and Classification*, pp.301-306. Springer: Berlin. 1993.
16. L. Girardin. *Cyberspace geography visualization*. Available at URL: heiwww.unige.ch/girardin/cgv. 1995
17. R. C. Gonzales and R. E. Woods. *Digital Image Processing*. Reading, MA: Addison-Wesley. 1992.
18. J. Barrera, J. Banon and R. Lotufo. Mathematical Morphology Toolbox for the Khoros System. In: *Image Algebra and Morphological Image Processing V*, E. Dougherty et al. (Eds.). *Proc. SPIE*, vol. 2300, pp. 241-252. 1994.
19. J. Serra. Image Analysis and Mathematical Morphology. Academic Press: London. 1982.
20. L. Najman and M. Schmitt. Geodesic Saliency of Watershed Contours and Hierarchical Segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18, pp. 1163-1173. 1996.
21. A. Bleau and L. J. Leon. Watershed-based segmentation and region merging. *Comp. Vis. Image Underst.*, 77, pp. 317-370.
22. J.A.F. Costa, N. Mascarenhas and M.L.A Netto. Cell nuclei segmentation in noisy images using morphological watersheds. In: *Applications of Digital Image Processing XX*. A. Tescher (Ed.). *Proc. of the SPIE*, vol. 3164, pp. 314-324. 1997.
23. J.A.F. Costa and M.L.A. Netto. Estimating the Number of Clusters in Multivariate Data by Self-Organizing Maps. *International Journal of Neural Systems.* Vol. 9, no. 3, pp. 195-202. 1999.
24. J.A.F. Costa and M.L.A. Netto. Cluster analysis using self-organizing maps and image processing techniques. *Proc. of the 1999 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, Tokyo, Japan.
25. E. Nakamura and N. Kehtarnavaz. Determining the number of clusters and prototype locations via multi-scale clustering. *Pattern Recognition Letters*, 19, pp. 1265-1283. 1998.
26. T. Li, Y. Tang, S. Suen and L. Fang. Hierarchical classification and vector quantisation with neural trees. *Neurocomputing*, 5, 119-139. 1993
27. J. Racz, and T. Klotz. Knowledge representation by dynamic competitive learning techniques. *Proc. SPIE*, 1469, pp. 778-783.
28. R. Adams, K. Butchart, and N. Davey. Hierarchical classification with a competitive evolutionary neural tree. *Neural Networks*, 12, pp. 541-551. 1999
29. J.A.F. Costa and M.L.A. Netto. Automatic Data Classification by a Hierarchy of Self-Organizing Maps, *Proc. 1999 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, Tokyo, Japan.
30. P. Koikkalainen. Progress with the Tree-Structured Self-Organizing Map. In: *Proc. of the 11 th European Conference on Artificial Intelligence*, pp. 211-215. 1994.
31. R. Miikkulainen. Script Recognition with Hierarchical Feature Maps. *Connection Science*, pp. 83-101. 1990.