Cluster Analysis Using Self-Organizing Maps and Image Processing Techniques

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

Department of Computer Engineering and Industry Automation
School of Electrical and Computer Engineering - UNICAMP
Campinas - SP 13083-970, BRAZIL
{costa, marcio}@dca.fee.unicamp.br

ABSTRACT

Cluster analysis methods are used to classify R unlabeled objects in a P-dimensional space into groups based on their similarities. Often the real number of categories may not be known a priori, and methods such as k-means may impose a structure on data rather than finding it. The self-organizing feature map (SOM) has been widely studied as a software tool for visualization of high-dimensional data. Some advantages in using SOM include information compression and density estimation while trying to preserve topological and metric relationship of the primary data items. This paper focus the usage of SOM as a clustering tool and some of the additional procedures required to enable a meaningful cluster's interpretation in the trained map. Topics discussed here include the usage of mathematical morphology segmentation method watershed to segment the neuron's distance image (u-matrix). Finding good watershed markers and the modification of the u-matrix homotopy are discussed. The algorithm automatically produces labeled sets of neurons that are related to the clusters in the P-dimensional space. An example of non-spherical, complex shaped and non-linearly separable clusters illustrate the capabilities of the method.

1. INTRODUCTION

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 scientific, military and industrial applications. The user of such data base systems may be faced with the problem of having too much data but does not have the information for strategic decisions. 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.

Clustering in unlabeled data $X = \{ x_1, x_2, \dots , x_n\} \subset \Re^p$ is the assignment of labels to the vectors in X, and hence, to the objects generating X [1]. A partition of a set of n patterns in a p-dimensional feature space must be found in a way that those patterns in a given cluster are more similar to each other than the rest. Even the number of categories (k) may not be known a priori, and it occurs that selecting wrongly this parameter, e.g., when using the k-means method, the analyst imposes a structure on the data rather than finding it. Applications to clustering algorithms range from engineering (e.g. pattern recognition, image segmentation and analysis) to biology (e.g. classifying plants, taxonomy), from archaeology (e.g. studying archaeological sites) to social sciences (analysis of crimes), from medicine to astrophysics, etc. 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.

Many different algorithms and approaches have been suggested but no general cluster analysis theory was yet developed. Good reviews include [2-3]. Clustering methods range from those that are largely heuristic to more formal procedures based on statistical models. Most frequent used methods are hierarchical and partitioning methods.

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; *(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 minimizing one objective criterion. 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. The most common method is the K-means that uses heuristics for reducing the within-group sum of squares.

How to efficiently specify the 'correct' number of clusters from a given multidimensional data set is one of the most fundamental and unsolved problems in cluster analysis [4]. Much of the responsibility of validating the final results is often left to the investigator. The *a priori* decision for k and criteria to be extremized in partitioning methods lead, respectively, the method search for k groups satisfying a given cluster geometry. For example, the within-group sum of squares forces clusters to be hyperspheres. In the case of hierarchical methods, the determination of the appropriate number of clusters involves selecting one of the steps of the process using a second optimality criterion.

Neural network implementations have been proposed for clustering problems: Kamgar-Parsi et al. [5] 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. Recently Frank et al. [6] presented a comparative analysis of some ART networks for clustering.

The self-organizing feature map (SOM) 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 [7]. This paper focuses the usage of SOM as a clustering tool and some of the additional procedures required to enable a meaningful cluster's interpretation in the trained map. Topics discussed here include the usage of mathematical morphology segmentation method watershed to segment the neuron's distance image (u-matrix). Finding good watershed markers and the modification of the u-matrix homotopy are discussed. The algorithm automatically produces labeled sets of neurons which are related to the clusters in the P-dimensional space. An example of non-spherical, complex shaped and non-linearly separable clusters illustrate the capabilities of the method.

The structure of the paper is as follows: Section 2 describes briefly the basic SOM model. The Unified matrix method is presented in section 3 and methods for its segmentation is given in section 4. Experimental results are presented in section 5 and the last section summarizes the paper conclusions and final remarks.

2. THE SELF-ORGANIZING MAP

The SOM algorithm [7], originally introduced by T. Kohonen in 1981, is one of the best known artificial neural network algorithms and was inspired by the way in which various human sensory impressions are neurologically mapped into the brain, such that spatial or other relations among stimuli correspond to spatial relations among the neurons. 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 [7-9].

The SOM consists essentially of two layers of neurons: input-layer I and the unit layer U. Inputs to the network are n-dimensional vectors of real numbers. 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 n-dimensional weight vector $\boldsymbol{m}_i = [m_{i1}, m_{i2}, \ldots, m_{in}]^T$, where n is equal to the dimension of the input vectors. Usually the map topology is a rectangle but also toroidal topologies have been used.

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 $\boldsymbol{x}$ 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

$$\| \boldsymbol{x} - \boldsymbol{m}_c \| = \min_i \{ \| \boldsymbol{x} - \boldsymbol{m}_i \| \} \qquad (1)$$

where $\|\cdot\|$ 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

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

where t denotes time, $\boldsymbol{x}(t)$ 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 $h(d, t)$:

$$h_{ci}(t) = \alpha(t) \cdot h(\| \boldsymbol{r}_c - \boldsymbol{r}_i \|, t) \qquad (3)$$

where $\boldsymbol{r}_i$ 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 $\alpha(t)$ gradually decreases to zero according to the specified learning rate function type. Along with learning rate the neighborhood radius decreases as well.

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 n_i 'somehow' topologically close input data vectors map to n_m adjacent map nodes or even to the same single node.

A recent, and faster, variation of SOM is its parallel implementation [7]. Like the k-means method, the batch updating step replaces each map unit by the average of the data vectors that were in its neighborhood for 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.

3. VISUALIZING NEURON'S RELATIONS THROUGH THE UNIFIED DISTANCE MATRIX

The Unified Distance Matrix (U-matrix) method was developed by A. Ultsch and co-workers [9-10] 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.

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 could also be used to determine a '*distance*' between u and its immediate neighbors. Similar neighbor vectors will produce low distance values. The U-matrix can then be used to display the similarity/dissimilarity structure of all units and their neighbors.

The self-organizing map consists of a two-dimensional array 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 $d_{xy} = 0.5 * (ne + nw(v))$, where e, n and nw denotes the *east*, *north* and *north-west*-distance of a neuron to their neighbors [19]. Considering Euclidean distances, in the case of rectangular lattice topology the distances d_x, d_y and d_{xy} can be defined as

$$dx(x,y) = \| b_{x,y} - b_{x+1,y} \| = \sqrt{\sum_i \left(w_{i_{x,y}} - w_{i_{x+1,y}} \right)^2}$$

$$dy(x,y) = \| b_{x,y} - b_{x,y+1} \| = \sqrt{\sum_i \left(w_{i_{x,y}} - w_{i_{x,y+1}} \right)^2}$$

$$dxy(x,y) = \frac{1}{2} \left(\frac{\| b_{x,y} - b_{x+1,y+1} \|}{\sqrt{2}} + \frac{\| b_{x,y+1} - b_{x+1,y} \|}{\sqrt{2}} \right)$$

$$= \frac{1}{2\sqrt{2}} \left[\sqrt{\sum_i \left(w_{i_{x,y}} - w_{i_{x+1,y+1}} \right)^2} + \sqrt{\sum_i \left(w_{i_{x,y+1}} - w_{i_{x+1,y}} \right)^2} \right]$$

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 and the U-Matrix is filled as bellow:

i	*j*	*(i,j)*	$U_{i,j}$
o	*e*	*(2x+1, 2y)*	*dx(x,y)*
e	*o*	*(2x, 2y+1)*	*dy(x,y)*
o	*o*	*(2x+1, 2y+1)*	*dxy(x,y)*
e	*e*	*(2x, 2y)*	*du(x,y)*

where the abbreviations o and e stand for odd and even. Then, the components of the matrix U take values as

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

There are at least two possibilities for the calculation of $du(x, y)$: we can use the mean or the median value of their surrounding elements. Let $C = (c_1, c_2, \dots c_k)$ be the values of the surrounding elements of the $U_{2x,\, 2y}$ rising as a sorted sequence with cardinality k ($k = |C|$). In our case of a rectangular topology, $k = 8$. In the case of $du(x, y)$ be the mean of C,

$$du(x, y) = \tilde{c} = 1/k \sum_{i=1}^{k} c_i \cdot$$

For the case of median value we have

$$du(x, y) = \begin{cases} c(k+1)/2 & \text{if } k \text{ odd} \\ \dfrac{c(k)/2 + c(k+1)/2}{2} & \text{if } k \text{ even} \end{cases}$$

where c(k), k = 1, 2, ... K, $K \leq 8$, denotes the surrounding elements sorted in increasing order of magnitude.

4. FINDING GROUPS OF SIMILAR NEURONS THROUGH U-MATRIX SEGMENTATION

Clustering without knowing '*anything*' about the data (about its distribution or number of clusters) is a difficult (ill-posed) problem and there is no general solution. Furthermore, the automatic selection of an optimal number of reference vectors for each cluster is a non-trivial task and no unique solution for it can be given yet [11].

Instead of using a single prototype for representing one cluster, which occurs for example in the k-means method, a better solution could be using a set of model vectors for each cluster. This approach enables the generation of complex shaped clusters rather than spherical or ellipsoidal clusters. This section explains an approach for generating such sets via labeling regions of similar neurons in the u-matrix. The nonlinearities in the resulting SOM mapping can revel the data structure if training had been well done. U-matrix 'valleys' correspond to similar neighbor neurons, which will be candidates to clusters. The main problem is to segment efficiently the U-matrix image which generally have many local *minima*, and borders separating clusters may not be well defined.

4.1 Image Segmentation

Image segmentation consists of partitioning an image into meaningful, non-intersecting, regions of interest. These regions are homogeneous with respect to one or more signal or structural property such as brightness, color, texture, context etc. Many computer vision tasks process image regions [12-13] after segmentation for further analysis or classification.

Segmentation techniques rely on two broad categories: contour-based methods and region-based methods. The first approach look for the local gray level discontinuities in the image and the second seeks 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 $\boldsymbol{R}$ is connected to x_j if and only if there is a sequence of $\{x_i, \dots, x_j\}$ such that x_k and x_{k+1} are connected and all the nodes are in $\boldsymbol{R}$.

Definition 2. $\boldsymbol{R}$ is a connected region if the set of points x in $\boldsymbol{R}$ has the property that every pair of nodes is connected.

The set of regions ($\boldsymbol{R}_k$) is known as a partition when the entire image is segmented ($\bigcup^{m}_{k=1} \boldsymbol{R}_k$). Each region is generally a homogeneous area satisfying some criteria, $H(\boldsymbol{R}_k)$ = true, where H is a Boolean function, and $H(\boldsymbol{R}_i \cap \boldsymbol{R}_j) = \varnothing$, $i \neq j$. 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.

The simplest alternative to find regions of neurons from U-matrix would be converting it to a binary image Y by a suitable threshold. It can be done for example by selecting a valley point between two peaks of U-matrix histogram. Although it may seem simple, finding the correct value is problematic, especially in the case of U-matrix that usually possess a complex and noisy histogram.

4.2 Image Segmentation by Watersheds

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 1960s [14]. 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. General MM readings include [15-16].

The watershed algorithm was introduced for the purpose of segmentation by Beucher and Lantuéjoul [17] and is one of the most powerful tools used 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 [17-18] 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).

Numerous techniques have been proposed to compute the watershed [14-19]. 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 [19].

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 the case of u-matrices, direct application of watersheds leads to oversegmentation. This is a main problem that must be preventing by using an efficient homotopy image modification module in which kernels of clusters are assigned to its proper valleys [13]. 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.

Let **Z** be the set of integers, and let E be a rectangle of $\mathbf{Z}^2$, 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

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

$\forall i \neq j$, 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.

The general approach for a segmentation using watershed would be:

1) Find the markers, i.e., one connected component for each object and one connected component for the background;
2) Impose the new regional minima by modification of image homotopy (or gray-scale geodesic reconstruction);
3) Compute the watershed;

4.3 Finding Watershed Markers for U-matrix

Although we have tested many forms of obtaining good markers to be seeds of groups of neurons in the U-matrix, this section will present simple approaches that have been leading to good results in a wide variety of cases. Given the U-matrix image U, the following steps are performed:

1) Filtering: create the image U_1 by removing any pore with area less or equal than three pixels.
2) For $k = 1$ to M, where M is the highest gray level of the image, create the image U_1^k that corresponds to the conversion U_1 to a binary image using as threshold k.
3) The number of connected regions of U_1^k, N_{cr}^k, can be found by a simple connected component labeling algorithm [12,20].

The last two steps perform like a slicing for all the gray levels in U-matrix. Now we seek for stable regions in a useful range of gray level values of the image. This can be visualized by flat portions of the plot N_{cr}^k versus k.

4) Seek the highest frequent number of regions η from the partitions histogram, whose bins are formed by the consecutive and equal values of N_{cr}^k. Operationally, this can be viewed as searching in the vector Ncr the largest contiguity portion of the same value N_{cr}^k.
5) Take as image marker U_1^ν, where ν is the lowest value k from portion of the vector Ncr which was chosen in the step 4.

The filtering in step 1 smooth very weakly the image. It can be performed applying the morphological operators 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 minor structures in the image resulted from the noisy characteristics of U. Visually, the filtered image U_1 is almost the same of U.

Another way of finding markers is the use of a clustering method such k-means, or even another SOM e.g. in 1-dimension topology, for segmenting the u-matrix. It is expected that most of the pixels in the resulting u-matrix fall in low values for a true cluster problem, while the minor percent remain to represent the cluster borders. We can perform a kind of threshold in the resulting k-means partition by assigning the lowest prototypes to be the representative model of the markers. Other approaches for finding markers were performed [21] 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 [21].

4.4 Labeling SOM neurons

The algorithm for labeling each neuron of the self-organizing map can be summarized as follows:

1) Find the U-matrix watershed markers by the steps present in section 4.3.
2) Perform watershed segmentation on the U-matrix, using the markers from last step, and assign a different label for each connected region.
3) Copy the U-matrix labels to the corresponding neurons in the SOM grid.

If there remains unlabeled neurons in the SOM grid, which may occur if the corresponding pixel in the u-matrix is in the border between clusters, two strategies can be done. These neurons can be classified by k-nearest neighbor neuron region calculating the distances using the SOM weigh space. Conversely, we can leave these neurons unlabeled. For a given pattern that best match this unit we seek 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 next section.

5. Simulation example: The chainlink data set

A nontrivial example for benchmarking a multidimensional clustering algorithm was proposed by Ultsch [10], the 'chainlink' data set. The data set consists of 1000 points in $\Re^3$-space arranged such that they form the shape of two intertwined three-dimensional rings. One of the rings extends into x-y direction and the other into x-z direction. The two rings can be thought of as two links of a chain with each one consisting of 500 data objects. This problem illustrates the capabilities of finding the structure of the data even for non-spherical, complex shaped and non-linearly separable clusters. Figure 1 shows a view of the data set.

The SOM grid size of all experiments were set 15×15. Weight initialization was linear and the training was done with the batch algorithm [7]. The neighborhood function was Gaussian and during ordering the neighbor radius decreased from 12 to 1 and the number of training epochs was 500.

Figure 2 shows the resulting configuration of neurons in the *3-D* space after the unsupervised training. Neighborhood in the SOM grid is expressed by the lines connecting the neurons. The corresponding u-matrix for figure 2 is given in figure 3. 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. One of the main problems in segmenting the u-matrix is that borders separating clusters usually may have some weak portion, which causes two different clusters to be merged in an automatic process. The usage of watershed is motivated to find the true cluster's separating lines, given cluster markers which can be regarded to be the cluster kernels.

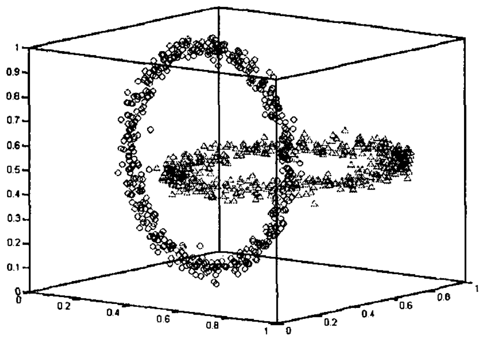
Fig. 1 - The chainlink data set

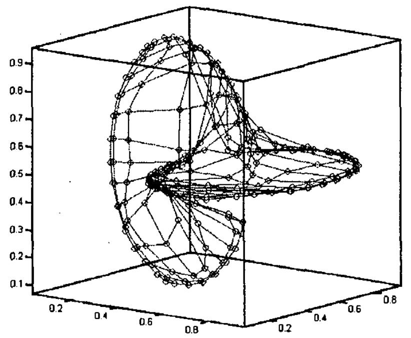
Fig. 2 - SOM grid after 500 iterations

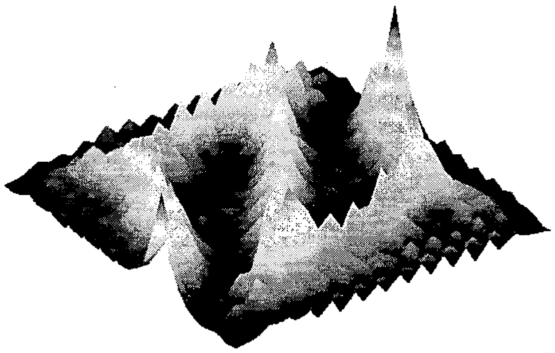
Fig. 3 - U-matrix for SOM presented in figure 2

Figure 4 shows the number of connected regions (N_{cr}^{k}) for each gray level of the image U_I^{k} (see section 4.3) for a useful gray level range. Following the algorithm, the system decided automatically the correct number of clusters (2). Taking as image marker the binary image obtained from thresholding the u-matrix by the lowest gray level k from largest contiguity portion, whose value was T=*24*. Figure 5 shows the watershed lines overimposed the u-matrix presented in figure 3. The obtained U-matrix partitions are presented in figure 6. The two cluster neuron regions (colored as white and gray) are separated by the watershed contour (colored as black).

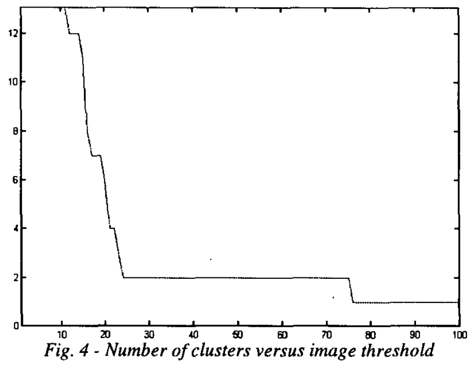
Fig. 4 - Number of clusters versus image threshold

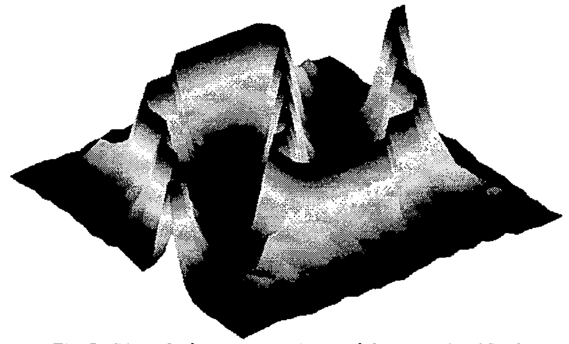
Fig. 5 - Watershed contour overimposed the u-matrix of fig. 2

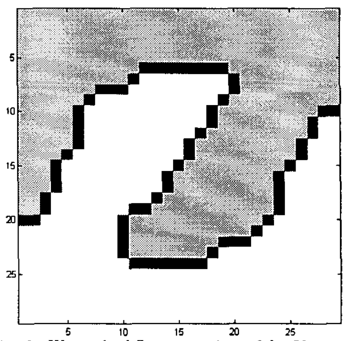
Fig. 6 - Watershed Segmentation of the U-matrix

Labeling the neurons as described in section 4.4, and testing the efficiency of the obtained partition by classifying all patterns in BMUs associated to regions, it was obtained 100% of correct class assignment. This and other results for mixtures of Gaussians and real data sets show that the partitions of the SOM obtained by the method had lead to very good approximation of the ideal solution. Figure 7 presents the shape of the discovered clusters considering an influential radius of 0.1 for each active neuron, which can be defined to those neurons which have at least one data pattern in its p-dimensional Voronoï cell. Using a set of model vectors labeled together (viewed as labeled nodes in a graph) we can obtain general shaped clusters. In the presented case, the influential space for each model vector is isotropic (hyperspherical) given by the neuron activation function used (Euclidean distance), but each cluster's influential space, which is the hypervolume integrated in the p-dimensional space of all neurons under the same label, is no longer isotropic. This is a main advantage when comparing with conventional methods such as *k*-means, which searches for hyperspherical clusters. Figure 8 shows the result for *k*-means, which fails in this problem even choosing *k* to the correct number of clusters (2). The figure shows the data labeled to the nearest prototypes.

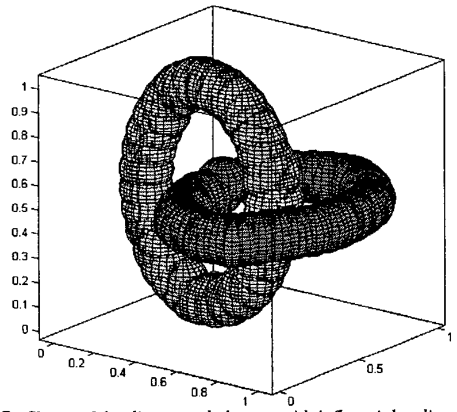

Fig. 7 - Shape of the discovered clusters with influential radius of 0.1

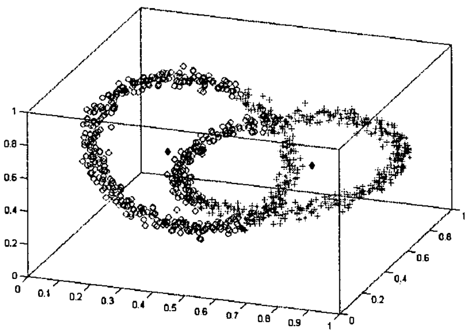

Fig. 8 - k-means clustering with k = 2.

6. Conclusions and Final Remarks

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 which may directly affect the process of decision-making, e.g., in business or even in an electronic robot. The absence of *a priori* knowledge about data distributions or any information about the cluster's composition makes this a difficult problem.

This paper focused the usage of SOM as a clustering tool and some of the additional procedures required to enable a meaningful cluster's interpretation in the trained map. The main advantages 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 [7] but setting training parameters is not straightforward. U-matrix reflects in an image the configuration obtained through unsupervised learning performed by SOM, 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 usage of mathematical morphology to segment the neuron's distance image (u-matrix) enabled finding the strong cluster borders, which resulted in an automatic neuron labeling procedure. Complex structured clusters can be find by integrating the influential space of neurons which were labeled together. An example of non-spherical, complex shaped and non-linearly separable clusters was used to illustrate the capabilities of the method. An extension of this method for enabling the detection of subclusters using a tree-like SOM model is described in a companion paper in this proceedings [22].

Acknowledgements: The authors thanks to useful discussion of U-matrix with Prof. *A. Ultsch* (Germany). The watershed algorithm was implemented by Prof. *R. Lotufo* (Unicamp). This work was supported in part by CAPES.

REFERENCES

[1] Pal, N.K. & Bezdek, J.C. (1995). On cluster validity for the fuzzy c-means model. *IEEE Trans. on Fuzzy Systems*, 3 (3), 370-379.

[2] Everitt, B.S. (1993). *Cluster Analysis.* Wiley: New York.

[3] Kaufman, L. & Rousseeuw, P. (1990). *Finding Groups in Data: Na Introduction to Cluster Analysis.* Wiley: New York.

[4] Su, M.-C., DeClaris, N. & Liu, T.-K. (1997). Application of neural networks in cluster analysis. *Proc. of the 1997 IEEE Intl. Conf. on Systems, Man, and Cybernetics, pp.* 1-6.

[5] Kamgar-parsi, B., Gualtieri, J.A., Devaney, J.E. & Kamgar-parsi, B. (1990). Clustering with neural networks. *Biological Cybernetics*, 63, 201-208.

[6] Frank, T., Kraiss, K.-F. & Kuhlen, T. (1998). Comparative analysis of fuzzy ART and ART-2A network clustering performance. *IEEE Trans. on Neural Networks*, v. 9, 544-559.

[7] Kohonen, T. (1997). *Self-Organizing Maps*, 2nd Ed., Springer-Verlag: Berlin.

[8] Mangiameli, P., Chen, S.K., & West, D. (1996). A comparison of SOM neural network and hierarchical clustering methods. *European J. Operational Research*, v. 93, 402-417.

[9] Ultsch, A. (1993). Self-Organizing Neural Networks for Visualization and Classification. In: O. Opitz et al. (Eds). *Information and Classification.* Springer, Berlin, 301-306.

[10] Ultsch, A. (1995). Self-Organizing Neural Networks perform different from statistical k-means clustering. *Gesellschaft für Klassifikation*, Basel.

[11] Kangas, J.A., Kohonen, T. and Laaksonen, J.T. (1990). Variants of Self-Organizing Maps. *IEEE Trans. on Neural Networks*, v. 1, 93-99.

[12] Costa, J.A.F., & Netto, M.L.A.(1997). Parts classification in assembly lines using multilayer feedforward neural networks, *Proc. 1997 IEEE Intl. Conf. on Systems, Man, and Cybernetics, pp. 3872-3877.* Orlando, FL.

[13] Costa, J.A.F., Mascarenhas, N. & Netto, M.L.A (1997). 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.

[14] Barrera, J., Banon, G., Lotufo, R. & Hirata Jr., R.(1997). *MMach: a Mathematical Morphology Toolbox for the KHOROS System.* Tech. Report RT-MAC-9704. IME / University of São Paulo.

[15] Matheron, G. (1975). *Random Sets and Integral Geometry.* Wiley: New York

[16] Serra, J. (1982). *Image Analysis and Mathematical Morphology.* Academic Press: London.

[17] Beucher, S.& Lantuéjoul, C. (1979). Use of Watersheds in Contour Detection. *Proc. Int'l Workshop Image Processing, Real-Time Edge and Motion Detection / Estimation*, Rennes, France.

[18] Najman, L. & Schmitt, M. (1996). Geodesic Saliency of Watershed Contours and Hierarchical Segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 18, 1163-1173.

[19] Vincent, L. & Sollie, P. (1991). Watersheds in digital space: an efficient algorithm based on immersion simulations. *IEEE Trans. on Pattern Anal. And Machine Intell.*, 13, 583-598.

[20] Parker, J.R. (1997). *Algorithms for image processing and computer vision.* Wiley: New York.

[21] Costa, J.A.F. & Netto, M.L.A. (1999). *Cluster Analysis using Self-Organizing Maps.* Tech. Report 004/99 DCA-FEEC-UNICAMP.

[22] Costa, J.A.F. & Netto, M.L.A. (1999). Automatic Data Classification by a Hierarchy of Self-Organizing Maps, *Proc. 1999 IEEE Intl. Conf. on Systems, Man, and Cybernetics,* Tokyo, Japan.