Estimating the Number of Clusters in Multivariate Data by Self-Organizing Maps

José Alfredo Ferreira Costa & Márcio Luiz de Andrade Netto

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

Determining the structure of data without prior knowledge of the number of clusters or any information about their composition is a problem of interest in many fields, such as image analysis, astrophysics, biology, etc. Partitioning a set of n patterns in a p-dimensional feature space must be done such that those in a given cluster are more similar to each other than the rest. As there are approximately $\frac{K^n}{K!}$ possible ways of partitioning the patterns among K clusters, finding the best solution is very hard when n is large. The search space is increased when we have no a priori number of partitions. Although the self-organizing feature map (SOM) can be used to visualize clusters, the automation of knowledge discovery by SOM is a difficult task. This paper proposes region-based image processing methods to post-processing the U-matrix obtained after the unsupervised learning performed by SOM. Mathematical morphology is applied to identify regions of neurons that are similar. The number of regions and their labels are automatically found and they are related to the number of clusters in a multivariate data set. New data can be classified by labeling it according to the best match neuron. Simulations using data sets drawn from finite mixtures of p-variate normal densities are presented as well as related advantages and drawbacks of the method.

1 Introduction

Classification is the process of assigning an unknown pattern to its proper class. We first use a labeled training set to partition the feature space in a supervised way. Many kinds of artificial neural networks such as multilayer feedforward have been used successfully in pattern recognition tasks [1]. Conversely, clustering methods, also denominated exploratory data analysis or automatic classification, are usually stated as methods for finding structures in data. 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. As there are approximately $\frac{K^n}{K!}$ possible ways of partitioning the patterns among K clusters, finding the best solution is computationally hard when n is large. The number of ways of sorting n observations into k groups is a Stirling number of the second kind [2]. The problem is compounded by the fact that the number of groups is usually unknown, so the possibilities are a sum of Stirling numbers. When $n = 25$ this is greater than 4×10^{18}. The raw input data is usually an $n \times p$ rectangular array of real numbers called *data matrix*. Each of the n objects is characterized with respect to p variables. For a given data set X, its elements are denoted by $x_i, i = 1, 2, ...n$, which are vectors in a p-dimensional space. Many different algorithms and approaches have been suggested but no general cluster analysis theory was yet developed. Good surveys include [2] and [3]. Clustering methods range from those that are largely heuristic to more formal procedures based on statistical models. Most frequently used are hierarchical and partitioning methods. 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 ini-

tial K group seeds. The most common method is the K-means that uses heuristics for reducing the within-group sum of squares. Two basic problems arise when using a clustering method [14]:

- The determination of the number of clusters present in the data;
- The assignment of data observations to one and only one cluster.

As stated by Su et al. [4], 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. Much of the responsibility of validating the final results is often left to the investigator. In some cases such as K-means the previous selection of K may impose a structure on data instead of discovering it. 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 a 128x128 SIMD array machine. They concluded that neural networks outperform conventional iterative techniques when there are well-defined clusters.

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 [13]. Cluster analysis problems have been studied using SOM [6, 7, 8].

This paper addresses the problem of determining the number of clusters in multivariate data. Related work include Su et al. [4] which used a single-layer neural network composed of quadratic neurons. They first determined the number of clusters using a histogram of response accumulation. A 3×3 mask was used to detect peaks which correspond to potential cluster centroids. A more statistical approach for determining the number of clusters was studied by Hardy [9], which compared three methods based on the hypervolume criterion with other well-known methods. We propose region-based image processing methods to post-processing the U-matrix obtained after the unsupervised learning performed by SOM. The morphological operation watershed, which is a topographic region growing method, is applied for identifying regions of neurons that are similar. Markers for the watershed algorithm are found analyzing the stability of partitions obtained for each gray level of the U-matrix image. The algorithm provides one-pixel wide, closed, and accurately localized contours of the regions. After segmentation the connected regions of the U-matrix are labeled and their codes are assigned also to the corresponding unlabeled SOM neurons. The result of the procedure is a labeled SOM in which the number of regions reflects the number of clusters of the input space. Each region, which may contain one of more neurons, may also be used to classify new input data.

2 The Self-Organizing Map Model

The SOM algorithm, originally introduced by T. Kohonen in 1981 [8], 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 [8, 7].

The SOM consists essentially of two layers of neurons: input-layer I and the unit layer U. Inputs to the network are p-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 p-dimensional weight vector $\mathbf{m}_i = [\mathbf{m}_{i1}, \mathbf{m}_{i2}, ..., \mathbf{m}_{ip}]^T$, where p 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.

Training is accomplished by presenting one input pattern $\mathbf{x}$ 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

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

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 up-

date rule for the weight vector of the unit i is

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t).[\mathbf{x}(t) - \mathbf{m}_i(t)] \quad (2)$$

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 by two components: the learning rate function $\alpha(t)$ and the neighborhood function $h(d,t)$:

$$h_{ci}(t) = \alpha(t).h(\|\mathbf{r}_c - \mathbf{r}_i\|, t) \quad (3)$$

where $\mathbf{r}_i$ 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 [8]. 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 High Dimensional Data Relations Using the Unified Distance Matrix

The Unified Distance Matrix (U-matrix) method was developed by A. Ultsch and co-workers [7] 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.

Each neuron u possesses the eight immediate neighbors. Associated with each unit is an p-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 weights. For each neuron in b exist three distances dx, dy and dxy in the U-matrix. dx and dy denotes, respectively, the east and the north distances. dxy denotes the northwest-distance of a neuron to their neighbor. Considering Euclidean distances, in the case of rectangular lattice topology the distances dx, dy and dxy can be defined as

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

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

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

The U-matrix combines these three distances into one matrix U of size $(2X-1) \times (2Y-1)$. 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

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

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 $C = (c_1, c_2, \ldots c_k)$ 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,

$$du(x,y) = \widetilde{c} = \frac{1}{k}\sum_{i=1}^{k} c_i$$

For the case of median value we have

$$du\,(x,y) = \begin{cases} C\left(\frac{k+1}{2}\right) & if\ k\ odd \\ \frac{C\left(\frac{k}{2}\right)+C\left(\frac{k+1}{2}\right)}{2} & if\ k\ even \end{cases}$$

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

4 Finding groups of similar neurons through image segmentation

Clustering with no knowledge 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 [10]. Regions of input space with a high density pf data points may be represented by more than one neuron. We can merge similar responding neurons in many ways, for example using contiguity-constrained hierarchical clustering [11]. Our method follows Ultsch [7] by using the U-matrix to detect non-linearities in the resulting SOM mapping. U-matrix 'valleys' correspond to similar neighbor neurons, which will be candidates to form clusters. The main problem is to efficiently segment the U-matrix image which generally have many local *minima*, and borders separating clusters may not be well defined.

4.1 Image Segmentation, Mathematical Morphology and Watersheds

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 [1, 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. 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. 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, e.g., 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.

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 [14].

The watershed algorithm was introduced for the purpose of segmentation by Beucher and Lantuéjoul [15] and is one of the most powerful tools used for image segmentation. The idea of the watershed algorithm 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. The main problem is the oversegmentation that occurs when all regional minima are used as region markers, especially in noisy images, which is the case of U-matrices. This problem can be preventing by using an efficient homotopy image modification module in which kernels of clusters are assigned to its proper valleys [12].

Let $\mathbf{Z}$ 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 $\mathbf{Z}$, 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 can be briefly written as

$$\psi_{B_c,g}(f)(x) = i \,\big|\, L_{B_c,g}(x, i \leq g \leq i) < L_{B_c,g}(x, j \leq g \leq j)\,,$$

$\forall i \neq j$, 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),

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

is the minimum length of a point to a set, $\eta_f\{\pi_{B_c,g}(x,X)\} = max\{f(y) : y \in \pi_{B_c,g}(x,X)\}$ is the length of a point to a set, and $\pi_{B_c,g}(x,X)$ 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); and (3) Compute the watershed. One of the approaches of obtaining good markers is presented next section [16] and it is related to the stability of regions for a wide range of threshold operations over the U-matrix.

4.2 Finding Watershed Markers for U-matrix

Although we have tested many forms of obtaining good markers to be seeds of groups of neurons for the U-matrix, this section will present one simple approach that has got 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, N_{cr}^k, can be found by a simple connected component labeling algorithm [5,16].

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 most probable 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 N_{cr} the largest contiguity portion of the same valueN_{cr}^k.

5) Take as image marker U_1, where is the lowest value k from portion of the vector N_{cr} which was chosen in the step 4.

The filtering in step 1 produces a mild smoothing. 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 resulting from the noisy characteristics of U. Visually, the filtered image U_1 is almost the same of U. Other similar approaches for finding markers were performed [4] and include: a) images after area opening the regional minima of U, with an increasing value of area; b) residues from the sup-reconstruction of the image U from the marker image created by the addition of an increasing value H to the image U; and c) other more elaborated methods [16].

4.3 Automatic Labeling of the Self-Organizing Map

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 4.2.

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 followed. These neurons can be classified by k-nearest neighbor neuron region calculating the distances using the SOM weight 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 in the next section.

5 Application example: Finite mixtures of p-variate Gaussian densities

This section describes the process of generating data for exemplify the method and some results. Because of the use of simulated sets, the underlying structure is precisely known but this information is considered external to the clustering process, not available in the practice. All simulations were implemented using Matlab.

5.1 Generation of Data Sets

Multivariate observations of a set of n entities can be expressed as $x_1, ...x_n$, where each x_i is a vector of p-dimensions. In this particular case the data was generated using the finite mixture model of p-variate Gaussian densities [21]. Each sample can be viewed as arising from a population F which is a mixture of a

finite number k of subpopulations $F_1, ..., F_k$, in some proportions $\pi_1, ..., \pi_k$, respectively, with

$$\sum_{j=1}^{k} \pi_j = 1 \; and \; 1 \geq \pi_j \geq 0 \, (j = 1, ..., k)$$

The probability density function (p.d.f) of an observation x in F can be represented in the finite mixture form

$$f\left(X; \underline{\theta}\right) = \sum_{j=1}^{k} \pi_j \, f_j \left(X; \underline{\theta}_j\right)$$

The distribution of the data within the j^{th} subpopulation is given by $f_j\left(X; \underline{\theta}_j\right)$ and the probability of obtaining a member of the j^{th} group is π_j. $\underline{\theta}_j$ denotes the vector of all unknown parameters associated with the parametric forms adopted for these k component densities. In our case each $f_j\left(X; \underline{\theta}_j\right)$ were multivariate normal component densities, defined by

$$f_j\left(X; \underline{\theta}_j\right) = (2\pi)^{-\frac{d}{2}} |\Sigma_j|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\left(\mathbf{x} - \underline{\mu_j}\right)^T \Sigma_j^{-1} \left(\mathbf{x} - \underline{\mu_j}\right)\right]$$

where T denotes the transpose, $\underline{\theta}_j$ consists of the elements of the mean vectors $\underline{\mu_j}$ and the covariance matrices Σ_j, $j = 1, \ldots k$. When the number of clusters is known the problem is to estimate the unknown parameters π_j, $\underline{\mu_j}$ and Σ_j. The first problem is to determine efficiently the number of clusters (or subpopulations) within the mixture. Three mixture populations were generated for the tests. The mean vectors are the vertices of a $\Re^3$ unit cube (0,0,0), (0,0,1), ..., (1,1,1) which forms an eight-class data set. The covariance matrices were set to $\rho_i^2.I$, $i = 1, ...3$, where I is the identity matrix. The parameter ρ_i controls the variability of each class. The values of ρ_i for the three examples were 0.05, 0.15 and 0.25 for the first, second and third data set, respectively. The probability of cluster overlap increases with ρ_i. The purpose of different values is to study the region and borders degradation effects in the U-matrix. Each data set comprises 1000 samples, 125 each class. All information about the real number of clusters and class membership is left unknown to all stages of the method and will be used only for performance comparison of the structure discovered by the process.

5.2 Simulation Results

The SOM grid size of all experiments were set 15×15. Weight initialization was linear and the training was done with the batch algorithm [8]. The neighborhood function was Gaussian and during ordering the neighborhood radius decreased from 12 to 1. Data was normalized to zero mean and variance one. The number of training epochs was 1000 for all cases. Due to reasons of space, some figures of the second data set were suppressed. Figures 1 and 2 show the resulting configuration of neurons, and the data patterns, in the *3-D* space after the unsupervised training for the first and the third data set. Neighborhood in the SOM grid is expressed by the lines connecting the neurons. The corresponding U-matrices are given in figures 3 and 4. Increasing ρ the subpopulations begin to overlap and the problem or determining the number of clusters becomes more complex. This fact reflects in the U-matrix. Borders and valleys become less defined. The extreme will occur when the data do not have structures at all. A uniform data density will result in a near flat U-matrix.

Figure 5 shows the number of connected regions (N_{cr}^k) for each gray level of the image U_1^k (see section 4.2). The solid line, the dashdot line and the dotted line represents the results for the first, second and third data sets, respectively.

Following the algorithms presented in section 4.2, the three cases lead to the correct number of clusters (8). This is clearly evident for the first and second data sets, where the number of regions is stable in a great range of the gray levels. For the third data set it was expected to be more complex because the cluster structures are less defined than for the former sets. We can see an instability around the correct number of clusters and the larger plateaus occurs for *eight* and *four* clusters, in the gray level range {82-118} and {131-145}, respectively. The eight clusters solution wins the four clusters solution by 37 against 15 levels. Increasing even more ρ, e.g. $\rho = 1$, the plot is shifted toward the right and large plateaus desapears, indicating the great cluster overlap. For a more rigorous decision, a security margin could also be used.

Figures 6-8 presents the obtained U-matrix partitions for the three SOM networks. The watershed image markers were the lowest gray level k from largest contiguity portion in figure 5, which were $k = 4$, 16, 82, for the first, second and third data sets, respectively.

Labeling the Self-Organizing Map as described in section 4.3, 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 (CCA) for the first data set. The second data set also performed almost as well as the first: only 6 patterns were misclassified, which gives 99.4% of CCA. The last data set obtained 87.9% of CCA.

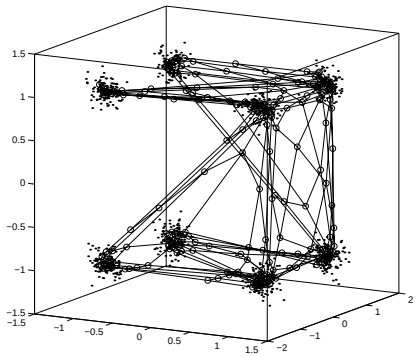

Figure 1. Data set and the configuration of neurons ($\rho = 0.05$).

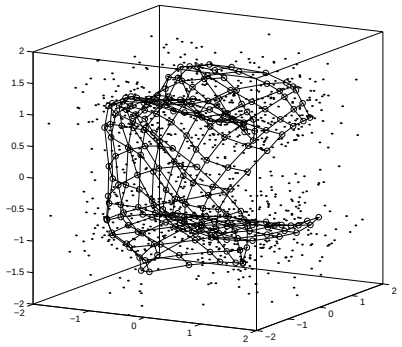

Figure 2. Data set and the configuration of neurons ($\rho = 0.25$).

The results show that partitions of SOM obtained by the method had led to very good approximations of the ideal solution.

6 Conclusions and Final Remarks

This paper presented a new approach for cluster detection using self-organizing maps. What makes the problem complex is the absence of *a priori* knowledge about data distributions or any information about their composition. The SOM neural network was used first to partition the feature space. The reference vec-

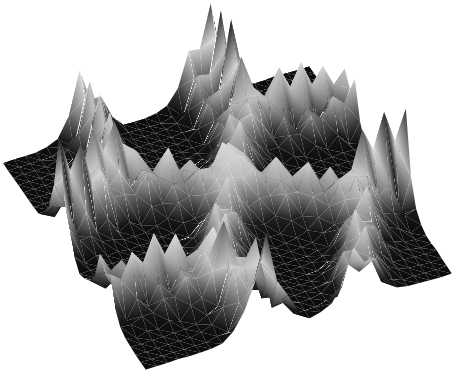

Figure 3. $3D$-view of the U-matrix ($\rho = 0.05$).

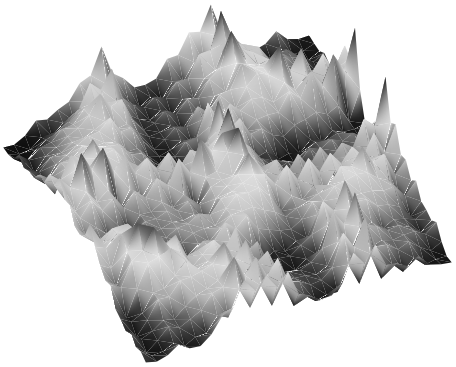

Figure 4. $3D$-view of the U-matrix ($\rho = 0.25$).

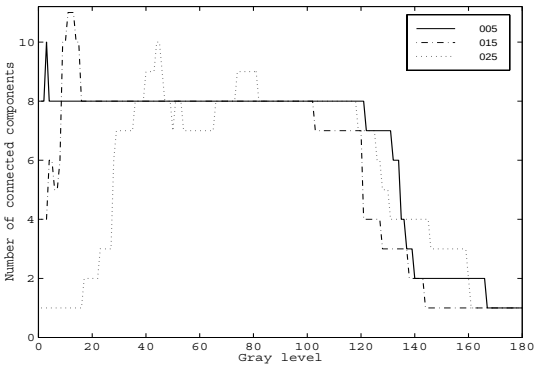

Figure 5. Plots of the number of connected regions for a useful range of image gray levels

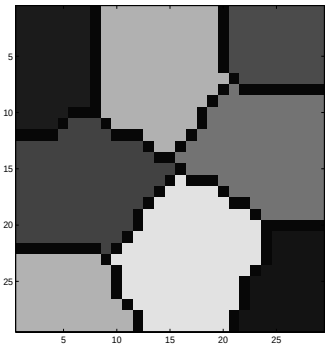

Figure 6. U-matrix segmentation ($\rho = 0.05$)

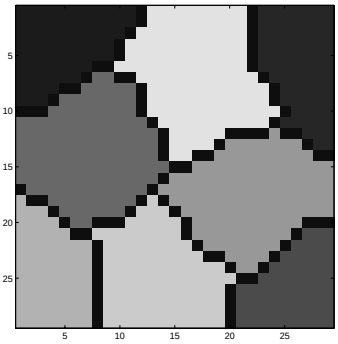

Figure 7. U-matrix segmentation ($\rho = 0.15$)

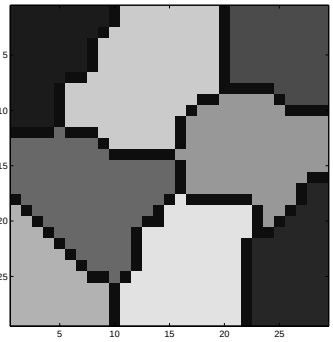

Figure 8. U-+matrix segmentation ($\rho = 0.25$)

tors were merged in regions through image processing analysis of the unified distance matrix. The morphological algorithm watershed was applied to the U-matrix and region markers were found automatically by a method that considers stable partitions of the image regions after a multilevel threshold operation.

Examples of decreasing cluster isolation of 8-class three-variate finite Gaussian mixture densities were presented. The increase of cluster overlap was reflected in the U-matrix, which valleys and borders became less defined. Classification of data using the obtained partitions led to very good results.

Comparing the three cases ($\rho = 0.05, 0.15$ and 0.25) we note that U-matrix methods perform best when clusters do not overlap. It must be noted that the decrease of performance also occurs with other methods [16]. An increase of fuzziness of the clusters will degrade the U-matrix 'valley boundaries' because groups of neurons tend to get closer, leading to the attenuation of the non-linearities between classes. This was also tested with other data sets. Applying these methods in the Fisher Iris data only two meaningful valleys were found. This is because only one of the three clusters is well defined.

SOM is robust regarding the weight vectors initialization 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. SOM convergence has not yet been proved [8].

Future investigations will combine U-matrix methods with some heuristic and statistical interpretation forms of the map. Other elaborated forms of image segmentation and watershed markers selection through relaxation and regularization methods may also lead to good U-matrix partitions. Another extension of this method for enabling the detection of subclusters using a tree-like SOM model was described in Costa and Netto [16, 17].

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] J.A.F. Costa and M.L.A. Netto. Parts classification in assembly lines using multilayer feedforward neural networks. In *Proc. of the 1997 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, pages 3872–3877, Piscataway, NJ, 1997.

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

[3] L. Kaufmann and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[4] M.-C. Su, N. DeClaris, and T.-K. Liu. Application of neural networks in cluster analysis. In *Proc. of the 1997 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, pages 1–6, Piscataway, NJ, 1997.

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

[6] P. Mangiameli, S.K. Chen, and D. West. A comparison of som neural network and hierarchical clustering methods. *European J. Operational Research*, 93:402–417, 1996.

[7] A. Ultsch. Self-organizing neural networks for visualization and classification. In O. Opitz, B. Lausen, and R. Klar, editors, *Information and Classification*, pages 307–313, London, UK, 1993. Springer.

[8] T. Kohonen. *Self-Organizing Maps, 2nd Ed.* Springer-Verlag, Berlin, 1997.

[9] A. Hardy. On the number of clusters. *Comp. Stat. and Data Analysis*, 23:83–96, 1996.

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

[11] F. Murtagh. Interpreting the Kohonen self-organizing map using contiguity-constrained clustering. *Pattern Recognition Letters*, 16:399–408, 1995.

[12] J.A.F. Costa, N.D.A. Mascarenhas, and M.L.A. Netto. Cell nuclei segmentation in noisy images using morphological watersheds. *Proceedings of the SPIE*, 3164:314–424, 1997.

[13] J.R. Parker. *Algorithms for image processing and computer vision*. Wiley, New York, 1997.

[14] H.J.A.M. Heijmans. *Morphological Image Operators*. Academic Press, Boston, 1994.

[15] S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *Proc. Intl. Workshop Image Processing, Real-Time Edge and Motion Detection / Estimation*, Rennes, France, 1979.

[16] J.A.F. Costa. *Automatic data classification and analysis by self-organizing neural networks*. PhD thesis, UNICAMP, Campinas, SP, Brazil, 1999.

[17] J.A.F. Costa and M.L.A. Netto. Automatic data classification by a hierarchy of self-organizing maps. In *Proc. of the 1999 IEEE Intl. Conf. on Systems, Man, and Cybernetics*, Tokyo, Japan, 1999.