A Gravitational Approach for Enhancing Cluster Visualization in Self-Organizing Maps

Leonardo Enzo Brito da Silva, José Alfredo Ferreira Costa
Departamento de Engenharia Elétrica
Universidade Federal do Rio Grande do Norte
Natal, Brazil
{leonardoenzob, jafcosta}@gmail.com

***Abstract*—This paper presents a modified gravitational clustering algorithm applied to the neurons of self-organizing maps in order to enhance the visualization of clusters through the U-matrix technique. For a given neuron, the proposed method considers the attraction among its k nearest neighbors in the data space, where k decreases monotonically over time and its value may vary according to the local pattern density. The attraction between neurons that are considered as not belonging to the same close-knit group is penalized. The results obtained for some synthetic and real world data sets are presented.**

Keywords-self-organizing maps; gravitational clustering; visualization techniques

I. INTRODUCTION

Nowadays, a plethora of data from the most diverse sources is collected and stored [1]. Data mining is one of the fields that aim to transform this information into useful knowledge. Among the main problems faced in this field, may be cited the data scalability and dimensionality, as well as its complexity, heterogeneity and quality (noise and outliers). Therefore, the analysis of databases requires careful interpretation of the results obtained with the mathematical models and the visualization techniques applied [2]. Visualization consists of the conversion of the data attributes into a visual structure, so as to observe its characteristics and properties [3].

The self-organizing maps (SOM) [4] are artificial neural networks widely used in the data mining field, mainly due to the mapping of a high dimensional input space (data space) to an output space of lower dimensionality (fixed grid of neurons), while preserving data topology. In this sense, the SOM network is a nonlinear generalization of principal component analysis [5]. It is used for clustering as well as for visualization. The U-matrix [6] is a well-known visualization technique associated with the SOM network. Its main issue concerns its resolution when applied to decreasing map sizes, i.e. on small maps the visualization is compromised, while on large maps the definition of clusters becomes increasingly clear in data sets where distance metrics are relevant.

Recently, gravitational-based clustering algorithms have been used to perform the clustering task [7]-[9]. These algorithms are hierarchical and agglomerative, i.e. they progressively define clusters from a database given a similarity metric, while forming a tree structure: in its base, each pattern is a cluster, and, at the top, there is only one cluster.

This paper focuses on improving the U-matrix visualization through the application of an algorithm based on the gravitational principles on the SOM neurons, as a way of increasing inter-cluster distances and decreasing intra-cluster distances. More specifically, the objective is to determine an updating rule for the weights associated with the SOM neurons, in order to perform a post-processing and provide a visualization in which separation between clusters is sharper.

The remainder of the paper is organized as follows. Section II provides general considerations of the SOM network, while Section III discusses some of its well-known visualization techniques. In Section IV, a brief description of some gravitational algorithms is provided. In Section V, the proposed method is defined, and, in Section VI, the data sets used in the experiments are concisely described. The simulation results and discussions are presented in Section VII. In Section VIII, some conclusions are drawn.

II. SELF-ORGANIZING MAPS

Self-organizing maps consist of a set of topologically ordered neurons situated in a static lattice (output space). The neuron grid can be either rectangular or hexagonal, differing in the number of immediate neighbors - four or six respectively. In general, the network grids are 1-D or 2-D. Although higher dimensionalities are possible, they generally are not used, since visualization becomes more difficult or not even feasible. Each neuron has an associated weight vector in the data space (input space), so a projection from a higher to a lower dimensional space is obtained. The SOM network can be seen as an adaptive vector quantization algorithm. The learning process involved is unsupervised and encloses the following three principles: competition, cooperation and adaptation. For each pattern presented to the network, the neurons compete with each other, so that a winner (best matching unit - BMU) is defined as the one with the minimum Euclidean distance to this pattern:

$$\|\mathbf{x}_i - \boldsymbol{w}_i\| < \|\mathbf{x}_i - \boldsymbol{w}_l\| \quad \forall l \neq i \tag{1}$$

where $\|\cdot\|$ is the Euclidean norm, $\boldsymbol{w} \in \mathfrak{R}^{\boldsymbol{d}}$ is a weight

 ISBN: 978-1-61208-274-5

vector ($\boldsymbol{w}_i$ is the BMU), $\mathbf{x}_i \in \mathfrak{R}^d$ is a pattern from the data set, and d is the dimension of the data space. However, not only the winner neuron, but its neighborhood also participates in the learning process. The adaptation rule is given by (2) [4]

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + \eta(t) h_{i,j}(t)\left[\mathbf{x}_i - \boldsymbol{w}_j(t)\right] \quad (2)$$

where t is time, $\boldsymbol{w}_j(t)$ is the weight vector associated with the jth neuron, $\mathbf{x}_i$ is the ith pattern from the data set presented to the network, $\eta(t)$ is the learning rate, $h_{i,j}(t)$ is the neighborhood kernel. The neighborhood kernel (which is centered on the BMU and is usually Gaussian) and the learning rate must be monotonically decreasing as the training algorithm progresses [4].

During the training stage, the SOM network behaves as an elastic net that molds itself to the intrinsic shape formed by the patterns. The neuron's placement reflects the data set density distribution: the number of neurons in a certain region of the input space is related to the number of patterns in that region, what is known as the magnification factor.

The quality of a given trained SOM can be measured by the following figures of merit: the quantization error (3) [4] and the topographic error (4) [10],

$$q_e = \frac{1}{N}\sum_{k=1}^{N}\left\|\boldsymbol{x}_k - \boldsymbol{w}_{BMU}^{x_k}\right\| \quad (3)$$

$$t_e = \frac{1}{N}\sum_{k=1}^{N} f(\boldsymbol{x}_k) \quad (4)$$

where N is the number of patterns of the data set, $\mathbf{x}_k$ is the kth pattern from the data set, $\boldsymbol{w}_{BMU}^{x_k}$ is the BMU of the pattern $\mathbf{x}_k$. The function $f(\boldsymbol{x}_k)$ is equal to zero if the first and second BMU of the pattern $\boldsymbol{x}_k$ are adjacent. Otherwise the function $f(\boldsymbol{x}_k)$ is equal to one.

The quantization error discloses the network resolution, while the topographic error depicts when there is a divergence between the neighborhood of neurons in the input and output spaces. An extensive discussion of topology in neural networks based on vector encoding can be found in [11].

III. VISUALIZATION TECHNIQUES

In order to suitably view clusters in a given trained SOM network, visualization techniques must be applied. That is, a post-processing stage using its prototypes is needed so as to infer characteristics of the dataset. Typically, visualization techniques take into account not more than one metric within its definition, for example, distances between prototypes, as the U-matrix, component planes [12], component gradients matrix [13], or pattern density, as in the hit histogram, P-matrix [14], CONNvis [15] and Smoothed Data Histogram (SDH) [16]. There are methods that take into consideration both distance among prototypes and pattern density associated with them, such as the CONNDISTvis [17] and the U*-matrix [18].

The U-matrix is one of the most popular visualization techniques, and consists of a matrix whose positions are filled with the Euclidean distances between the neurons in the data space. Consider that a map has a rectangular grid of size $a \times b$, then the U-matrix has the size $(2a-1) \times (2b-1)$. The relative positions of the neurons themselves in the matrix are obtained by a function f of the neighboring distances in the grid, where generally f is a mean or a median function of neighboring values. The Euclidean distances in the U-matrix can be calculated on the basis of all attributes or specific ones with the use of masks. The particular case where a U-matrix is calculated for each attribute of data form the component planes.

The P-matrix aims to estimate the probability density of the data [19]-[20]. It has a structure that is the same size as the map grid. In the P-matrix, the value at the position related to the neuron $\boldsymbol{w}_i$ consists of the number of patterns inside a hypersphere of radius r centered on that neuron. The radius is a fixed parameter for all neurons and is called Pareto radius. The U*-matrix is an enhanced visualization method that is generated by using information provided by both the U-matrix and the P-matrix. It has the same size of the latter. Its value U^* for the relative position of each neuron $\boldsymbol{w}_i$ in the grid is obtained by (5) [18]

$$U^*(\boldsymbol{w}_i) = U(\boldsymbol{w}_i)\left[\frac{P(\boldsymbol{w}_i) - \bar{P}}{\bar{P} - P_{max}} + 1\right] \quad (5)$$

where $P(\boldsymbol{w}_i)$ and $U(\boldsymbol{w}_i)$ are the values of the P-matrix and U-matrix associated with the neuron $\boldsymbol{w}_i$, $\bar{P}$ and P_{max} are the mean and maximum values of the P-matrix, respectively.

IV. GRAVITATIONAL CLUSTERING ALGORITHMS

The gravitational algorithm and its application to the clustering task were first proposed by Wright [21] and rely on the law of universal gravitation. It may be classified as an agglomerative hierarchical algorithm, as it begins with a set of N objects and ends with only one. However, contrary to classic hierarchical agglomerative algorithms where patterns are static, in the gravitational algorithm all patterns are considered as mobile particles subjected to the gravitational fields of one another. The results obtained using such method can be seen as a dendrogram. The strength of a given cluster structure is greater the larger the interval of time during which the system remains in that clustering state.

A variant of the clustering algorithm was proposed by Gomez et al. [7], in order to automatically determine the number of clusters, remove noise and generate prototypes representing the database. This approach differs from the latter in the sense that the particles are always considered with unitary mass (as opposed to the original algorithm where the mass changes when there is a mergence), and the stopping criterion is the number of iteration. The algorithm is very sensitive to the gravitational constant and its decay function, which may lead to a generation of only one cluster or none at all. According to the model, the particles move as described in (6)-(7) [7]

 ISBN: 978-1-61208-274-5

$$\boldsymbol{x}(t+1) = \boldsymbol{x}(t) + \left(\frac{G}{\left\| \boldsymbol{d}\big(\boldsymbol{x}(t), \boldsymbol{y}(t)\big) \right\|^3} \right) \boldsymbol{d}\big(\boldsymbol{x}(t), \boldsymbol{y}(t)\big) \quad (6)$$

$$\boldsymbol{d}\big(\boldsymbol{x}(t), \boldsymbol{y}(t)\big) = \quad \boldsymbol{y}(t) - \boldsymbol{x}(t) \quad (7)$$

where G is the gravitational constant that decreases monotonically over time, $\boldsymbol{x}$ and $\boldsymbol{y}$ are patterns randomly chosen from the dataset, and $\|\cdot\|$ is the Euclidean norm. Particles are merged when separated by a minimum distance which is an input parameter.

It is also stated that a good performance of the algorithm does not necessarily need the entire database to be used, which thereby opens the possibility to use vector quantization techniques, such as the SOM network, before its application. Therefore, recently, a gravitational clustering of the SOM (gSOM) [8] based on the work of Gomez et al. [7] was proposed. It consists of two steps: in the first phase, a SOM network is trained with the data set. In the second phase, the interpolating neurons, that is, those neurons that are not associated with any pattern are eliminated, as well as their connections. The gravitational algorithm is then applied to the remaining neurons.

Each time a random pair of objects is selected, according to predefined probability functions, the gravitational algorithm is applied. The stopping criterion can be either the number of iterations or the maximum number of clusters to be found. The gSOM has also been used in a clustering ensemble [9], in which the different partitions obtained, due to its stochastic nature, are analyzed in a consensus function in order to define the final partition. Other variants of the gravitational algorithms and applications were proposed in the literature, such as [22]-[23].

V. PROPOSED APPROACH

The proposed method, the k-gSOM algorithm, is concerned with distance information between close neurons on the map as well as pattern density in their vicinity. It is assumed that due to the attraction exerted by the patterns to the neurons in the network, the overwhelming majority of neurons are located in high density places, while a minority make the connection between these groups (interpolating neurons). The proposed method relates to the work of Ilc and Dobnikar [8], consisting of two stages: in the first stage, a SOM network is trained using its standard algorithm, and, in the second stage, the proposed method is applied to the network neurons.

The technique is based on gravitational principle and on a hit histogram variant. It is a gravitational clustering algorithm as the neurons are subjected to attraction forces of one another, and they all tend to gather at the same position when time becomes sufficiently large. The information of pattern density and distance among close neurons is used to adapt their weights and collapse them in order to obtain an improved U-matrix visualization.

The hit histogram consists of an accumulation array of the same size as the map, where each bin is associated with the position of a neuron in the SOM grid. The hit histogram depicts the number of patterns that each neuron is the BMU. As opposed to the U-matrix, the information provided by a hit histogram is more useful when dealing with small sized maps, in which the pattern to neuron ratio is usually greater than 1. Otherwise, due to the dissolution phenomenon, the matrix associated with the hit histogram becomes very sparse, which impedes the proper display of the data characteristics. The P-matrix and SDH are examples of visualization techniques that surpass this issue by using hyperspheres with Pareto radius and considering more than one BMU for each pattern, respectively. In this work, the values of the hit histogram consist of how many patterns are within a hypersphere centered on each neuron. The radius of the hypersphere is regarded as the minimum for which all the neurons have at least one associated pattern. By doing this the division by zero in (9) is avoided (the neuron masses are associated with the hit histogram) as there are no neurons without a pattern associated.

Therefore, a neuron $\boldsymbol{w_j}$ at time t will be moved according to the attraction forces among its k_j neighbors $\boldsymbol{w_i}$. The pairwise force between neurons $\boldsymbol{w_j}$ and $\boldsymbol{w_i}$ is related to their proximity in the input space and the ratio of patterns shared by their associated hyperspheres. The overall number of neighbors k_j depends on whether $\boldsymbol{w_j}$ is in a denser region or not. The direction and magnitude of the movement is given by the resultant of all the attraction forces between $\boldsymbol{w_j}$ and its neighbors. The movement will occur until the stopping criterion is reached, which is the number of iterations. The equations governing the adaptation are as follows

$$\boldsymbol{w_j}(t+1) = \boldsymbol{w_j}(t) + \Delta \boldsymbol{w_j}(t) \quad (8)$$

$$\Delta \boldsymbol{w_j}(t) = \frac{1}{m_j(t) k_j(t)} \sum_{i=1}^{k_j(t)} \left\{ \frac{\big[1 + JC_{i,j}(t)\big]\big[1 - d_{i,j}(t)\big]}{p_{i,j}(t)} \times \big[\boldsymbol{w_i}(t) - \boldsymbol{w_j}(t)\big] \right\} \quad (9)$$

$$k_j(t) = k_{max}(t) H_j(t) \quad (10)$$

$$d_{i,j}(t) = \left\| \boldsymbol{w_i}(t) - \boldsymbol{w_j}(t) \right\| \quad (11)$$

where $\boldsymbol{w_j}$ is the jth neuron, k_j is the effective number of neighbors of neuron $\boldsymbol{w_j}$. The parameter k_j is proportional to the pattern density where $\boldsymbol{w_j}$ is located, and its maximum possible value is predetermined at time t as k_{max}. The $\|\cdot\|$ is the Euclidean norm, m_j is the mass of the neuron $\boldsymbol{w_j}$ and corresponds to the number of patterns inside the hypersphere centered on $\boldsymbol{w_j}$ at time t. The distance $d_{i,j}$ is normalized in the interval $[0;1]$ regarding all pairwise distances between neurons, and also negated in (9) so as to be transformed from a dissimilarity to a similarity measure.

The parameter $JC_{i,j}$ is the Jaccard coefficient [24] defined by the ratio of the intersection and union cardinalities of the sets containing the patterns covered by the hyperspheres of the neurons $\boldsymbol{w_j}$ and $\boldsymbol{w_i}$ at time t

$$JC_{i,j}(t) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} \quad (12)$$

where $|\cdot|$ is the set cardinality, S_i and S_j are the number of patterns inside the hyperspheres centered on the neurons $\boldsymbol{w_i}$ and $\boldsymbol{w_j}$, respectively. The parameter $p_{i,j}$ is defined as

$$p_{i,j}(t) = \begin{cases} \frac{1}{n}\sum_{q=1}^{n} m_q, & if\ d_{i,j}(t) > \alpha \\ 1, & otherwise \end{cases} \quad (13)$$

where m_q corresponds to the mass of one of the n neurons whose distance to $\boldsymbol{w_j}$ is less or equal to the parameter α at time t (Fig. 1). The attraction force is penalized by the parameter $p_{i,j}$ if a neuron $\boldsymbol{w_i}$ is very far regarding a close group of neurons around $\boldsymbol{w_j}$ that is defined by α. Thus, a neuron $\boldsymbol{w_i}$ cannot attract a single neuron from within this group, but in fact, the whole group, thereby diminishing the attraction force and compensating k_j if it is overly estimated.

The attraction among a decreasing number of neighboring neurons in the input space is considered as the algorithm progresses. For each neuron $\boldsymbol{w_j}$, at each iteration, the effective number of neighbors is a fraction of k_{max} that is proportional to the density of patterns in the region that the neuron is currently situated: H_j is the value of the hit histogram generated at time t and associated with neuron $\boldsymbol{w_j}$. The values of H_j are normalized in the range [0.1;1] so k_j is a nonzero percentage of k_{max}. By doing this we prevent that neurons in small clusters have the same neighborhood size as neurons in large clusters, and therefore reducing the influence of the latter over the first. The role of the parameter α consists of defining the minimum distance for which a set of neurons should be considered as a group. It relates to the minimum distance of mergence in the traditional gravitational algorithms. However, in the proposed method neurons are not merged nor eliminated: the number of neurons is constant throughout the algorithm steps, there is only an update to their position in the input space.

The Jaccard coefficient is included so as to add a second term of attraction between neuron $\boldsymbol{w_j}$ and a given neighbor $\boldsymbol{w_i}$: if they have patterns in common while considering a given hypersphere, they should be brought together proportionally to the intersection divided by the overall patterns associated with them. At each iteration of the algorithm, the pairwise distances between all prototypes are calculated, as well as the hypersphere radius, the neuron masses, the Jaccard coefficients and the effective number of neighbors for each neuron, before using their respective values in (9). The parameters $JC_{i,j}$, m_j and k_j are ultimately dependent on the radius of the hypersphere, which is calculated at each iteration of the algorithm. The attraction among a decreasing number of neighboring neurons in the input space is considered since k_{max} is monotonically

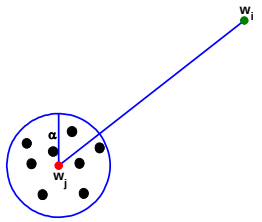

Figure 1. Illustrative case where the attraction between $\boldsymbol{w_j}$ (red dot) and $\boldsymbol{w_i}$ (green dot) is penalized by $p_{i,j}$ (mean mass of the group of neurons inside the circle of radius α). All neurons whose distances to $\boldsymbol{w_j}$ are less or equal to α are considered as belonging to the same close-knit group (black dots), and therefore the parameter $p_{i,j}$ related to their attraction forces is equal to unity.

decreasing while the algorithm progresses, as well as the parameter α. In this work, both k_{max} and α were set to decrease linearly with time t according to (14)-(15)

$$k_{max}(t) = (k_0 - k_f)\left(1 - \frac{t}{T}\right) + k_f \quad (14)$$

$$\alpha(t) = (\alpha_0 - \alpha_f)\left(1 - \frac{t}{T}\right) + \alpha_f \quad (15)$$

where T is the total number of iterations, α_0 and α_f are the initial and final values of α, respectively. The parameters k_0 and k_f are the initial and final values of k_{max}, respectively.

The summary of the algorithm is presented in Table I:

TABLE I. K-GSOM ALGORITHM

1. Initialize k_{max} and α as well as their decreasing functions.
2. Determine the hypersphere radius, calculate the masses of each neuron and generate the normalized hit histogram H.
3. Calculate and normalize the pairwise distance between all neurons.
4. For each prototype $\mathbf{w_j}$
a. Find k_j nearest neighbors by multiplying the current k_{max} by the value of H_j in the position associated to $\mathbf{w_j}$.
b. Calculate $\mathbf{\Delta w_j}$. If a neuron $\mathbf{w_i}$ is not close enough to $\mathbf{w_j}$ (distance defined by the parameter α) their attraction is penalized by dividing it by $p_{i,j}$, otherwise $p_{i,j}$ is equal to unity.
5. Update $\mathbf{w_j}$ (sequential algorithm).
6. If the stopping criterion was not met, return to step 2.

VI. DATA SETS

The proposed method was applied to the following synthetic data sets from the Fundamental Clustering Problem Suite [25]: *Hepta* and *Tetra*. Another artificial dataset consisting of two Gaussian clusters mixed with noise was considered. The *Wine* data set [26] was also used in the experiments. All datasets were normalized in the hypercube $[0;1]^n$ as a pre-processing stage. The *Tetra* data set consists of 400 patterns that form four very close clusters in $\Re^3$ so that density information is more relevant than distance among prototypes. The *Hepta* data set consists of 212 patterns that form seven well defined clusters in $\Re^3$, each

one with different variances. The *Noisy Gaussian* data set consists of two Gaussian clusters with 400 patterns and 100 patterns that represent noise. The *Wine* data set is a real world database that consists of 178 patterns that forms three clusters in a 13 dimensional space. All previously mentioned data sets are depicted in Fig. 2.

VII. RESULTS AND DISCUSSION

The experiments were carried out with SOM networks whose grid sizes were all 10x10, and were trained in the first stage using the SOM Toolbox [27]. For the second stage, the initial value of k_{max} was set to 80% of the total number of neurons and it was decreased linearly over the iterations until it reaches 1. The parameter α was also decreased linearly over the iterations from 10^{-1} to 10^{-3}.

The SOM network trained with the *Noisy Gaussian* dataset is depicted in Fig. 3, along with the distances to the closest pattern to each neuron. The maximum pairwise distance between a neuron and its closest pattern is then used as the hypersphere radius; therefore the least populated hypersphere will have 1 pattern. This information is used in order to generate the hit histogram and to calculate the Jaccard coefficient (Fig. 4). The steps of the algorithm shown in Figs. 3 and 4 are repeated continuously. The movement of neurons as the proposed algorithm progresses is depicted in Fig. 5. After 250 iterations, the final placement of the neurons is depicted in Fig. 6, as well as the values over time for: the hypersphere radius, the parameters k_{max} and α. The effective neighbor number for each neuron at each iteration is shown in Fig. 7.

It is perceptible in Fig. 6(d) that the radius tends to a permanent regime, that is, after a certain number of iterations it remains in a specific value with small fluctuations. Therefore, the computational cost may be reduced by setting the stop criterion as the sum of the radii differences between one iteration to the next: if it remains within a certain range ε for a fixed number of iterations then the algorithm stops. In Fig. 8, the trained SOM network and the best results of the proposed algorithm are shown. They were obtained for the *Tetra*, *Hepta*, *Noisy Gaussian*, and *Wine* data sets after 85, 125, 250 and 190 iterations, respectively. The Fig. 9 (a) and (b) depict the U-matrix and the U*-matrix (generated using the SOMVIS Package) obtained from the original SOM, respectively. In Fig. 9 (c) the U-matrix of the SOM network resulting from the application of the proposed method is shown.

As depicted in Fig. 9, the borders of the clusters are visually sharper than the original U-matrix and the U*-matrix. Albeit the final positions of the neurons do not correspond to the positions of the clusters' centroids (phenomenon resulting from the gravitational effect) a repositioning may be achieved by relating the neurons that converged to a centroid to their original map positions in the input space.

In order to measure the performance of the method, first the MBSAS [28] was applied so as to obtain the centroids resulting from the neurons' movement. The radius parameter was set to α_f. The number of centroids found and the prototypes they represent were stored and separated into

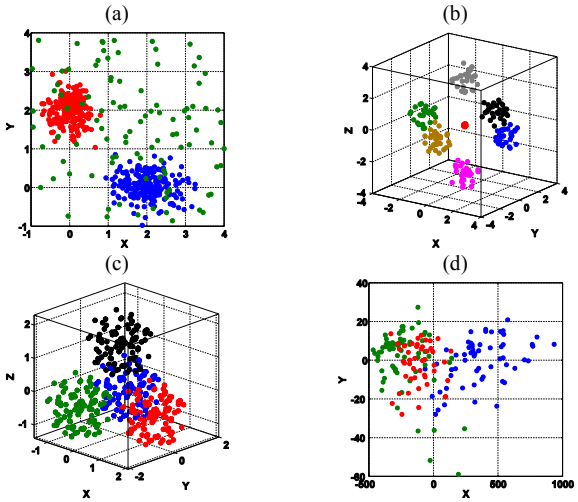

Figure 2. (a) Elements of the *Noisy Gaussian* data set. (b) Elements of the *Hepta* data set. (c) Elements of the *Tetra* data set. (d) Elements of the *Wine* data set using a 2-D PCA (principal component analysis) projection. Each class in each data set is depicted in a specific color.

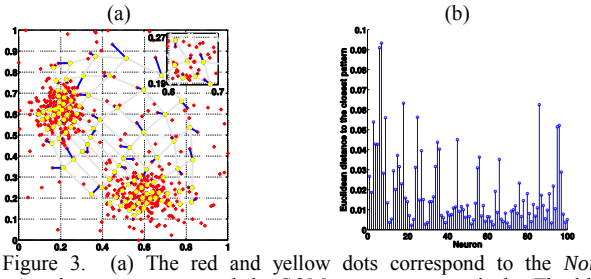

Figure 3. (a) The red and yellow dots correspond to the *Noisy Gaussian* data set patterns and the SOM neurons, respectively. The blue lines indicate which is the closest pattern to each neuron. (b) Stem plot regarding the Euclidean distances of the closest pattern to each neuron.

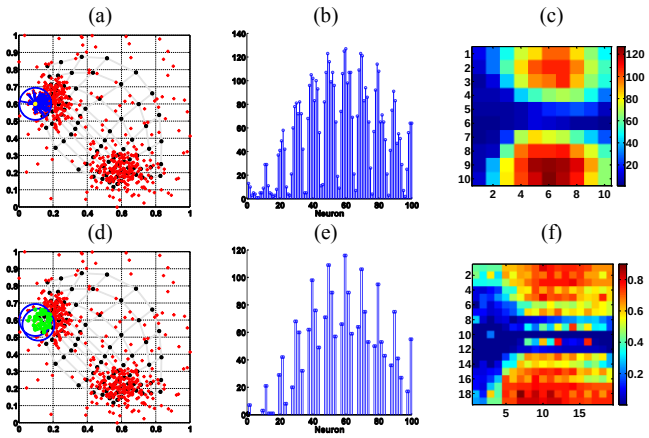

Figure 4. (a) The red and black dots correspond to the data set patterns and neurons, respectively. The yellow dot is a neuron which is the center of its correspondent hypersphere, which is depicted as the blue circle. All patterns inside this circle are linked to the neuron by blue lines. (b) Stem plot of the number of pattern inside each neuron hypersphere. (c) Hit histogram generated with the number of patterns inside the hyperspheres. (d) The red and black dots correspond to the data set patterns and neurons, respectively. The green dots are the intersection between the circles around two neighboring neurons. (b) Stem plot of the number of patterns in the intersection of neurons $\boldsymbol{w_i}$ and $\boldsymbol{w_j}$. (f) Jaccard coefficient matrix whose values are calculated among neurons that are 4-neighbor on the lattice (output space).

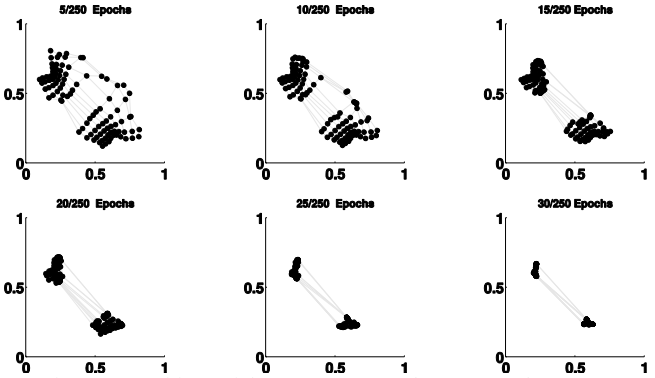

Figure 5. Positions of the neurons at epochs 5 to 30. The neurons are gathering in the densest regions of the *Noisy Gaussian* data set.

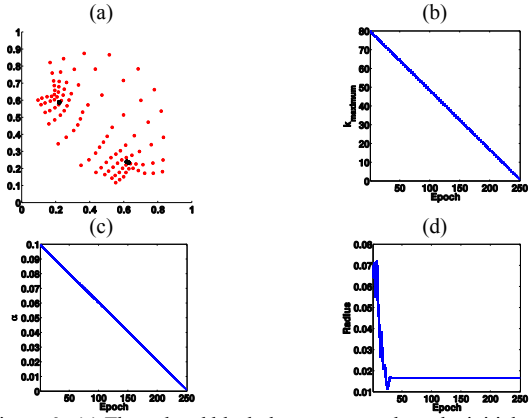

Figure 6. (a) The red and black dots corresponds to the initial and final positions of the neurons, respectively. The evolution of each parameter over time is shown in: (b) maximum possible number of neighbors (c) maximum distance for which neurons are considered as belonging to the same group (d) hypersphere radius for a given neuron $\boldsymbol{w_j}$.

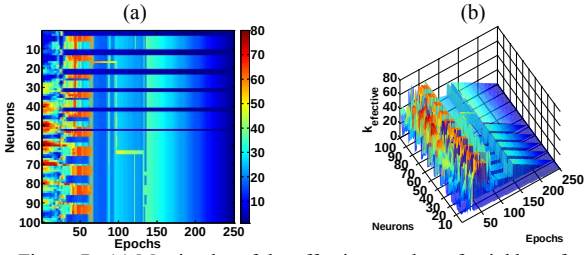

Figure 7. (a) Matrix plot of the effective number of neighbors for each neuron over time (b) Surface plot of 'a'. Neurons in regions with fewer patterns are seen as streaks or valleys, as it is expected. The effective number of neighbors is a fraction of k_{max} that is proportional to the density of the region the neuron is located.

classes, that is, which neurons converged to each position (see the representative colors depicted in Fig. 8). Then a cross tabulation was performed with the datasets' man given groundtruth so as to appropriately compare the classes. The classification accuracies were then calculated (see Table II) in order to evaluate the partitions visible in the new U-matrix. The classification accuracy is defined as (16) [29]

$$Clas\ acc = \frac{\#\ correct\ classified\ patterns}{\#\ patterns} \qquad (16)$$

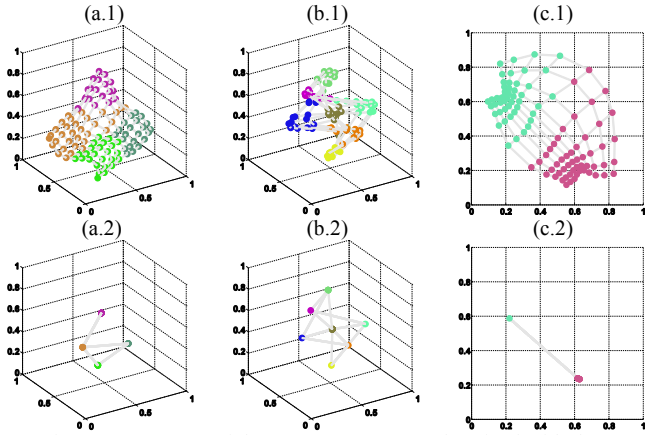

Figure 8. Neurons of the 10x10 SOM network trained with the *Tetra* (a.1), *Hepta* (b1) and *Noisy Gaussian* (a.1) data sets. Neurons resulting from the application of the proposed method in 'a.1', 'b.1' and 'c.1' are depicted in (a.2), (b.2) and (c.2), respectively. Neurons with the same color in the plots with indexes '1' converged to the same point in their associated plots with indexes '2'.

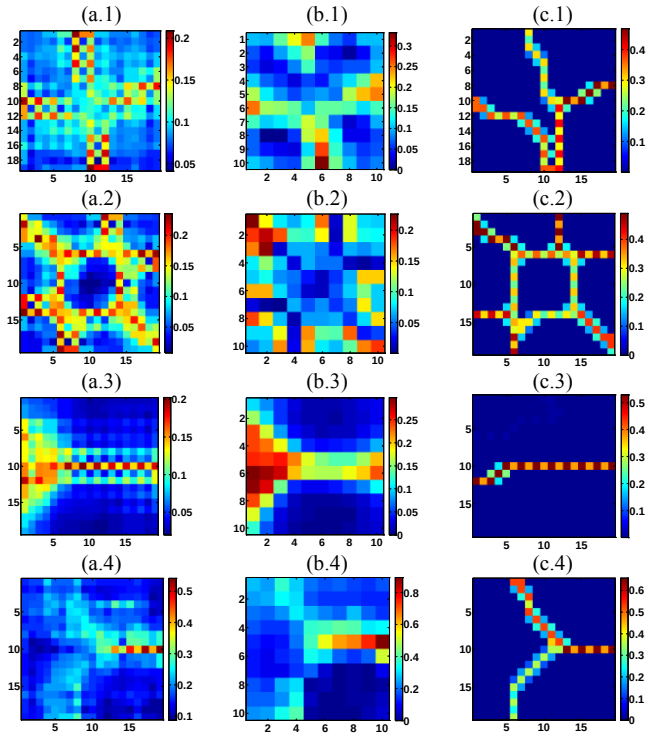

Figure 9. U-matrix (a) and U*-matrix (b) of the 10x10 trained self-organizing maps. U-matrix (c) of the SOM network whose neurons result from the application of the proposed method. The indexes 1 to 4 correspond to the following data sets: *Tetra* (1), *Hepta* (2), *Noisy Gaussian* (3) and *Wine* (4).

TABLE II. PERFORMANCE SUMMARY

Data set	*Number of centroids found*	*Classification accuracy*
Tetra	4	0.9775
Hepta	7	1
Noisy Gaussian	2	1
Wine	3	0.9719

 ISBN: 978-1-61208-274-5

The proposed algorithm is dependent of the iteration number, since the quantity of neighbors was defined as a function of it. In the experiments no universal value for the iteration number was suitable for all datasets as it affects the parameters k_{max} and α, and therefore, it must be set for each database.

VIII. CONCLUSIONS AND FUTURE WORK

A gravitational approach for enhancing the cluster visualization through the U-matrix technique was presented. It takes advantage of the U-matrix increasingly higher resolution while using larger SOM grid sizes and the neurons concentration achieved with the gravitational algorithm. The main concern is to define the neighborhood size and number of iterations as they directly influence the quality of the final map. The experiment parameters were kept equal for all data sets, except for the number of iterations. As the latter becomes larger, the neurons tend to converge to the same point, as it is expected from a gravitational algorithm. In all cases, the proposed approach was able to provide an improved visualization of the clusters using the U-matrix that was generated with the repositioned neurons.

The final result of the proposed algorithm does not represent the real position of the cluster centers due to the tendency of all particles to collapse at the same point, however as the interest consists primarily in visualizing and defining the number of clusters, it is not considered an issue, as the geometry relations are preserved and the visualizations obtained are sharper while remaining coherent.

Future works will focus on an heuristic for automatic selection of the total number of iterations, the parameters k_{max} and α, as well as their decreasing functions based on information from the data and SOM network, in order to achieve a suitable combination regarding the tradeoff between results and the computational cost.

REFERENCES

[1] D. T. Larose, Discovering Knowledge In Data, John Wiley & Sons, 2005.

[2] R. Xu and D. C. Wunsch II, "Survey of clustering algorithms," IEEE Transactions on Neural Networks, vol. 16, no. 3, May 2005, pp. 645–678.

[3] P.-N. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining. Addison Wesley, 2006.

[4] T. Kohonen, Self-Organizing Maps, 3rd ed., Springer-Verlag, 2001.

[5] H. Ritter, "Self-organizing feature maps: Kohonen maps," The Handbook of Brain Theory and Neural Networks, 1995, pp. 846–851.

[6] A. Ultsch and H. P. Siemon, "Kohonen's self organizing feature maps for exploratory data analysis," Proc. of the International Neural Networks Conference (INNC 90), 1990, pp. 305–308.

[7] J. Gomez, D. Dasgupta, and O. Nasraoui, "A New Gravitational Clustering Algorithm," Proc. of the third SIAM International Conference on Data Mining, 2003.

[8] N. Ilc and A. Dobnikar, "Gravitational clustering of the self-organizing map," Adaptive and Neural Computing Algorithms, Lecture Notes in Computer Science, vol. 6594, 2011, pp 11–20.

[9] N. Ilc and A. Dobnikar, "Generation of a clustering ensemble based on a gravitational self-organising map," Neurocomputing, vol. 96, Nov. 2012, pp. 47–56.

[10] K. Kiviluoto, "Topology Preservation in Self-Organizing Maps", Proc. of the IEEE International Conference on Neural Networks, vol. 1, 1996, pp. 294–299.

[11] H.-U. Bauer, J. M. Herrmann, and T. Villmann, "Neural maps and topographic vector quantization," Neural Networks, vol. 12, no. 4–5, Jun. 1999, pp. 659–676.

[12] J. Vesanto, "SOM-based data visualization methods," Intelligent Data Analysis, vol. 3, no. 2, Aug. 1999, pp. 111–126.

[13] J. A. F. Costa, "Uma nova abordagem para visualização e detecção de agrupamentos em mapas de Kohonen baseado em gradientes das componentes," Learning and NonLinear Models, vol. 9, no. 1, 2011, pp. 20–31.

[14] A. Ultsch, "Maps for the visualization of high-dimensional data spaces," Proc. of the Workshop on Self-Organizing Maps (WSOM 03), 2003, pp. 225–230.

[15] K. Taşdemir and E. Merényi, "Exploiting Data Topology in Visualization and Clustering of Self-Organizing Maps," IEEE Transactions on Neural Networks, vol. 20, no. 4, Apr. 2009, pp. 549–562.

[16] E. Pampalk, A. Rauber, and D. Merkl, "Using Smoothed Data Histograms for Cluster Visualization in Self-Organizing Maps," Proc. of the International Conference on Artificial Neural Networks (ICANN 02), 2002, pp 871–876.

[17] K. Taşdemir, "Graph Based Representations of Density Distribution and Distances for Self-Organizing Maps," IEEE Transactions on Neural Networks, vol. 21, no. 3, Mar. 2010, pp. 520–526.

[18] A. Ultsch, "U*-Matrix : a Tool to visualize Clusters in high dimensional Data," Technical Report no. 36, Dept. of Mathematics and Computer Science, University of Marburg, Germany, 2003.

[19] A. Ultsch, "Proof of Pareto's 80/20 Law and Precise Limits for ABC-Analysis," Technical Report no. 02 / c, University of Marburg, Germany, 2002.

[20] A. Ultsch, "Pareto Density Estimation: A Density Estimation for Knowledge Discovery," Proc. of the 27th Annual Conference of the German Classification Siciety (GfKl 03), 2003, pp. 91–100.

[21] W. E. Wright, "Gravitational clustering," Pattern Recognition, vol. 9, no. 3, Oct. 1977, pp. 151–166.

[22] T. Long and L.-W. Jin, "A New Simplified Gravitational Clustering Method for Multi-prototype Learning Based on Minimum Classification Error Training," Advances in Machine Vision, Image Processing, and Pattern Analysis, Lecture Notes in Computer Science, vol. 4153, 2006, pp. 168–175.

[23] J. Lange and H. Freiesleben, "A parameter-free non-growing self-organizing map based upon gravitational principles: Algorithm and applications," Artificial Neural Networks — ICANN 96, Lecture Notes in Computer Science, vol. 1112, 1996, pp. 827–832.

[24] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data, Prentice Hall, 1988.

[25] A. Ultsch, "Clustering with SOM: U*C," Proc. of the Workshop on Self-Organizing Maps (WSOM 05), 2005, pp. 75–82.

[26] A. Frank and A. Asuncion, "UCI Machine Learning Repository," 2010.

[27] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Self-Organizing Map in Matlab: the SOM Toolbox," Proc. of the Matlab DSP Conference, 2000, pp. 35–40.

[28] S. Theodoridis and K. Koutroumbas, Pattern Recognition, 4th ed., Academic Press, 2008.

[29] M. Meila and D. Heckerman, "An experimental comparison of model-based clustering methods," Machine Learning, vol. 42, no. 1–2, 2001, pp. 9–29.

 ISBN: 978-1-61208-274-5