A new batch SOM algorithm for relational data with weighted medoids.

Laura M. P. Mariño, Francisco de A. T. de Carvalho
Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil
Email: {lmpm,fatc}@cin.ufpe.br

***Abstract*—The great majority of previous works on SOM concern quantitative vectorial data. Nowadays, relatively few SOM algorithms are able to manage relational data despite their usefulness. This paper proposes a new batch SOM algorithm for relational data with weighted medoids. The particularity of the proposed approach is to consider the cluster representatives as vectors of weights whose components measure how objects are weighted as a medoid in a given cluster. From an initial solution and for a fixed epoch and radius, the proposed training batch SOM algorithm provides a partition and cluster representatives by optimizing a suitable objective function aiming to preserve the topological properties of the data on the map. Experiments with datasets of UCI machine learning repository, in comparison with relevant medoid-based batch SOM for relational data algorithms, showed the usefulness of the proposed method.**

***Index Terms*—Self-Organizing Maps, Bath SOM, Relational data, Weighted medoids**

I. Introduction

Self-organizing Kohonen map networks [1]–[6], called SOM, are one of the most popular unsupervised learning methods because they simultaneously perform clustering and non-linear data projection, and thus act as a powerful visualization tool. The SOM network consists of neurons (vertices) arranged in *a priori* chosen structure (usually a two or three dimensional grid map). Each neuron has a representative (prototype) and it is associated to a subset of the data (a cluster). The prototypes structure is imposed by both the data and the *a priori* structure itself.

SOM networks have been useful in tasks such as profiling of the behavior of criminals, categorization of galaxies , categorization of real estates, etc. Moreover, it has been applied on the statistic, industrial, biomedical, financial and many other areas (see [4], [7], [8] for surveys).

SOM networks aim to group the data taking into account a neighborhood structure among the clusters aiming to preserving the spatial order of the prototypes on the map: the most similar prototypes are associated with adjacent vertices, while less similar prototypes are associated with distant vertices on the map. As a consequence, SOM provides a powerful visualization tool because the clusters can be displayed according to their neighborhood structure.

During the training of the map, each object must select the neuron whose associated prototype is most similar to its description (best match unit, BMU). As a result, both the BMU-associated prototype and the neuron-associated prototypes in the BMU's spatial neighborhood are updated to better reflect the similarity between the object and these neurons.

SOM networks preserve the topological properties of the data, which means that if two objects are similar in the original description space, the corresponding BMU prototypes are also similar and will be associated with adjacent or nearby vertices on the map.

The training of the SOM network can be incremental or batch. According to Kohonen [3], the batch version of the SOM network is the most appropriate for practical applications. However, the original version of the batch training SOM network [1], [9] is limited to manage only quantitative vector data. Indeed, in unsupervised learning methods, much more attention was given for quantitative vector data. The vast majority of currently available machine learning and data analysis methods are based on a vector model in which, each object is described by a vector of quantitative values [10].

Unfortunately, many data depart strongly from this model [11]. In many real-world applications, data cannot be described by a fixed set of numerical attributes, for instance, when data are described by categorical variables or by relations between objects (e.g., persons involved in a social network) [10], [12], [13]. A way to cope with these more general type of data is to consider objects described by relational data where each pair of objects is represented by a relation. In this case, we assume that data is available in a relational form, where we only have information about the degrees to which pairs of objects in the data set are related. This type of data is known as relational data [10], [14].

The most common case of relational data is when the relationships between objects are expressed by a matrix of dissimilarities $\mathbf{D} = [d_{kl}]$, where each d_{kl} is the dissimilarity between the objects e_k and e_l. Relational data is more general in the sense that it is applicable to situations in which the objects cannot be represented by numerical features. It is also more practical for situations where the distance measure does not have a closed form solution, or when groups of similar objects cannot be represented efficiently by a prototype vector [10].

Unsupervised relational data methods have the advantages of introducing methods capable of handling such data using the most appropriate distance function for the problem, ensuring the confidentiality of data represented by a matrix dissimilarity, having the ability to handle heterogeneous data through its transformation into dissimilarities, etc.

Many unsupervised methods have been designed for relational data. In Hierarchical clustering, the sequential agglom-

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

erative hierarchical non-overlapping (SAHN) algorithm [15] handle dissimilarity data. In crisp partitional clustering, Hard C-medoids (HCMdd) [16] and Partitioning Around Medoids (PAM) [14] algorithms are extensions of k-means for relational data using the notion of generalized median: as cluster prototypes cannot be computed for relational data, it is replaced by an object of the original data (named medoid) that represents the best approximation among the original data. Ref. [17] also extends the k-means for relational data with the particularity that the cluster representatives are set of medoids (instead of a single medoid). Moreover, Ref. [18] extends the k-means for relational data in which the prototypes of the clusters are vectors of weights whose components measure how objects are weighted as a medoid in a given cluster. Finally, Relational Hard C-Means (RHCM) [19] algorithm extends k-means for Euclidean relational data in which the clusters prototypes are vectors of weights whose components are related to the membership of each object in a given cluster.

In fuzzy clustering, Fuzzy C-medoids (FCMdd) [16], Fuzzy C-Set-Medoids (FCSMdd) [20] and Fuzzy C-Means with Multiple Weighted Medoids (FCMMdd) [21] extend fuzzy c-means for relational data but the first with the cluster prototypes being a single medoid, the second with the cluster representatives being a set of medoids and the third with the clusters prototypes as vectors of weights whose components measure how objects are weighted as a medoid in a given cluster. Finally, Relational Fuzzy C-Means (RFCM) [12] and Fuzzy Analysis (FANNY) [14] extend k-means for Euclidean relational data in which the clusters prototypes are vectors of weights whose components are related to the membership of each object in a given cluster.

Concerning Self-Organizing Maps, Median Batch SOM [1], [9], [22] is the first extension of SOM for relational data. In this paper, the cluster prototypes are represented by a single medoid. Ref. [23] provided an extension of the Median Batch SOM where the cluster prototypes are represented by a set of medoids. Later, another approach to extend SOM for relational data was developed. It uses implicit "linear combination" of the original data and it is based in the following: the Euclidean distance between a data point and any linear combination of the original data points can be computed using solely the distance matrix. Batch [24] and on-line [13] versions SOM for relational data based on this latter approach are already available.

The great majority of previous works on SOM concern quantitative vectorial data. Nowadays, relatively few SOM algorithms are able to manage relational data despite their usefulness. Therefore, extensions of SOM for relational data are very much required. Refs. [18], [21] shown that multiple-weighted medoids can describe rich intra-cluster information, and are more powerful than a single medoid for representing a cluster. In order to take this advantage and combine with the visualization properties in an efficient learning strategy, we propose a new Batch SOM algorithm for relational data with weighted medoids, hereafter named Batch SOM algorithm with multi-medoids for relational data (RBSOM-MM**dd**).

In comparison with previous batch SOM methods for relational data [1], [23], the new contributions of RBSOM-MM**dd** are more precisely the following ones:

- the paper provides a suitable objective function that should be optimized in order to learn the SOM;
- for a fixed neighborhood radius, the paper gives the optimal solution for the computation of the representative (weighted medoids) associated to each neuron;
- for a fixed neighborhood radius, the paper provides an optimal solution for the partition associated to the neurons of the SOM;
- the paper gives the time complexity of the proposed method;
- the paper provides a meaningful evaluation of the proposed method in comparison with relevant medoid-based batch SOM algorithms for relational data.

The paper is organized as follows: Section II describe the proposed method and analyzes the temporal complexity. Section III presents the setup of our experiments. Section IV provides the performance evaluation of the proposed algorithms against previous approaches, presenting the results and discussing the main findings obtained with the above configurations. Finally, Section V concludes.

II. Proposed approach

This section introduces a batch Self-organizing map (SOM) algorithm with weighted medoids, RBSOM-MM**dd**, that is able to manage relational data. It is based on fuzzy clustering algorithm with multi-medoids (FMM**dd**) and Batch SOM algorithm with c-Medoids for relational data (RBSOM-SM**dd**).

Let $E = \{e_1, \ldots, e_N\}$ be the set of objects and let a $N \times N$ dissimilarity matrix $\mathbf{D} = (d_{kl})$ where $d_{kl} = d(e_k, e_l)$ is the dissimilarity between objects e_k and e_l $(1 \leq k, l \leq N)$ on dissimilarity matrix $\mathbf{D}$.

A SOM consists of a low-dimensional (usually, two-dimensional) regular grid (map), which contains C nodes (neurons). Each SOM map is associated with a clusters partition in which a neuron indexed by r has associated a cluster P_r and a representative (prototype) $\mathbf{v}_r$. In this paper, the representative $\mathbf{v}_r = (v_{r1}, \ldots, v_{rN})$ of cluster P_r is a N-dimensional vector of weights whose components measure how the objects are weighted as a medoid with respect to cluster P_r [18], [21].

The allocation function maps each object to an index $r = f(e_k)$ of the index set $\{1, \ldots, C\}$. The partition $\mathcal{P} = \{P_1, \ldots, P_C\}$ associated to a SOM is defined by the allocation function which gives the index of the cluster of $\mathcal{P}$ to which the object e_k belongs to, $P_r = \{e_k \in E : f(e_k) = r\}$.

Therefore, the proposed method aims to provide:

- A matrix $\mathbf{V} = (v_{rj})(1 \leq r \leq C; 1 \leq j \leq N)$ of prototype weights of the objects with respect to the clusters [21], where $\mathbf{v}_r = (v_{r1}, \ldots, v_{rN})$ is the vector of weights whose components v_{rk} measure how the object e_k is weighted as a medoid with respect to cluster P_r;
- The partition $\mathcal{P}$ of E into C clusters represented by $\mathcal{P} = (P_1, \ldots, P_C)$

Starting from an initial solution, and aiming that the obtained SOM be representative of the relationships among the object provided by the dissimilarity matrix $\mathbf{D}$, the matrix $\mathbf{V}$ of prototype weights of the objects with respect to the clusters and the partition $\mathcal{P}$ are obtained iteratively by the minimization of an error function J, computed as the sum of the error measures of all neurons:

$$J(\mathbf{V}, \mathcal{P}) = \sum_{k=1}^{N} \Delta(e_k, \mathbf{v}_{f(e_k)}) \tag{1}$$

Given an object e_k, the winner neuron called the best matching unit (BMU), is selected as the neuron that is closest to the object e_k. The BMU is indexed by $f(e_k)$ and identified with the prototype vector $\mathbf{v}_{f(e_k)}$. The error measure of a BMU $\mathbf{v}_{f(e_k)}$ with respect to the object e_k is computed by means of the dissimilarity function Δ that enables to compare each object e_k to each prototype $\mathbf{v}_r$ according to:

$$\Delta(e_k, \mathbf{v}_{f(e_k)}) = \sum_{r=1}^{C} h_{f(e_k),r}\, d(e_k, \mathbf{v}_r) \tag{2}$$

where $h_{f(e_k),r}$ is the neighborhood kernel function: it measures the influence neighborhood of BMU on neuron r, and it is defined as

$$h_{f(\mathbf{e}_k),r} = \exp\left\{-\frac{||a_{f(\mathbf{e}_k)} - a_r||^2}{2\sigma^2}\right\} \tag{3}$$

where $a_{f(\mathbf{e}_k)}$ and a_r are, respectively, the BMU and neuro r positions in the grid. Besides, σ is the neighborhood radius. The size of the neighborhood decreases with σ: the smaller σ, the fewer the neurons that belong to the effective neighborhood of a given BMU.

The function $d(e_k, \mathbf{v}_r)$ computes the dissimilarity between an object e_k and a cluster representative $\mathbf{v}_r$ and it is defined as

$$d(e_k, \mathbf{v}_r) = \sum_{j=1}^{N} (v_{rj})^n d(e_k, e_j) \tag{4}$$

In Eq. (4), the parameter n controls the level of smoothness of the distribution of prototype weights among all the objects in each of the clusters [18], [21].

A. *The* RBSOM-MM**dd** *algorithm*

When the radius σ is kept fixed, from an initial solution the training map of the RBSOM-MM**dd** algorithm is based on the minimization of cost function J, which is performed iteratively in two steps: representation and assignment. The representation step gives the optimal solution for the computation of the representatives associated to the neurons of the map. The assignment step provides the optimal solution for the clusters associated to the neurons of the map.

1) Initialization: Some initialization techniques proposed for the standard SOM can be extended to the case of relational data [22]. In this paper, we use a random initialization in order to initialize the matrix $\mathbf{V} = (v_{rj})\,(1 \leq r \leq C; 1 \leq j \leq N)$ followed by the computation of the initial partition $\mathcal{P}$. See more details in Algorithm 1.

After initialization, the algorithm runs for N_{iter} epochs. One epoch consists of a representation step in which the prototype of each cluster is updated followed by an assignment step, in which each input is associated with a cluster.

2) Representation: During the representation step, the partition $\mathcal{P}$ is kept fixed. The cost function $J(\mathbf{V}, \mathcal{P})$ is minimized with respect to the matrix $\mathbf{V} = (v_{rj})$ subject to $\sum_{j=1}^{N} v_{rj} = 1, \forall r$ and $v_{rj} \geq 0$, $\forall r, j$. First, we compute the Lagrangian function as follows:

$$L(\mathbf{V}, \mathcal{P}) = J(\mathbf{V}, \mathcal{P}) - \sum_{r=1}^{C} \beta_r \left(\sum_{j=1}^{N} v_{rj} - 1\right) \tag{5}$$

where β_r are the Lagrange multipliers. Then, taking the partial derivatives of L w.r.t v_{rj} and β_r, and by setting the partial derivatives to zero, and after some algebra we obtain:

$$v_{rj} = \left[\sum_{s=1}^{N} \left(\frac{\sum_{k=1}^{N} h_{f(e_k),r} d(e_k, e_j)}{\sum_{k=1}^{N} h_{f(e_k),r} d(e_k, e_s)}\right)^{\frac{1}{n-1}}\right]^{-1} \tag{6}$$

Remark. From Eq. (6) we can conclude that, at the final of the training of RBSOM-MM**dd** algorithm, when $h_{f(e_k),r} \sim 0$ for $f(e_k) \neq r$, as low is the $\sum_{e_k \in P_r} d(e_k, ej)$ as high is the weight of object e_j as a medoid with respect to cluster P_r.

3) Assignment: During the assignment step, the matrix $\mathbf{V}$ of prototype weights is kept fixed. The cost function $J(\mathbf{V}, \mathcal{P})$ is minimized with respect to partition $\mathcal{P}$ and each individual e_k is assigned to its nearest prototype (BMU). It can be easily shown that the error function J is minimized w.r.t the partition $\mathcal{P}$ when the clusters P_r are updated as follows:

$$P_r = \left\{e_k \in E : \Delta(e_k, \mathbf{v}_r) = \min_{1 \leq s \leq C} \Delta(e_k, \mathbf{v}_s)\right\} \tag{7}$$

where Δ is computed according to Eq. (2).

These steps are repeated until is achieved the max number of iteration N_{iter}. The algorithm 1 summarizes these steps.

B. *Time complexity*

The computation of the time complexity of Algorithm 1 takes into account its three (3) main steps as well as the input parameters: N_{iter} (the number of iterations), N (the number of objects in E) and C (the number of neurons (clusters)).

- *Initialization:* The initialization has four (4) steps: (i) computation the distance matrix between the nodes of the grid $\boldsymbol{\delta}$, (ii) initialization of the matrix $\mathbf{V}^{(t)}$, (iii) the setting of the initial partition $\mathcal{P} = (P_1, \ldots, P_C)$, and (iv) computation the objective function $J(\mathbf{V}, \mathcal{P})$ considering the initials matrix $\mathbf{V}^{(t)}$ and partition $\mathcal{P}$.

Algorithm 1 RBSOM-MM**dd** algorithm

Require: $\mathbf{D} = [d(e_k, e_l)]$ (the dissimilarity matrix), the size map, the number C of neurons (clusters), $\boldsymbol{\delta} = [||a_m - a_r||^2]\,(1 \leq m, r \leq C)$ (the distance matrix between the nodes of the grid), N_{iter} (the number of iterations), σ_0 (initial radius), σ_f (final radius);

Ensure: $\mathbf{V}$ (the matrix of prototype weights of the objects with respect to the clusters), $\mathcal{P}$ (the partition of E into C clusters);

Initialization:

Set $t \leftarrow 0$;

Compute $\sigma_{(t)} = \sigma_0 \left(\frac{\sigma_f}{\sigma_0}\right)^{\frac{t}{N_{iter}}}$;

Compute $h_{m,r} = \exp\left\{-\frac{||a_m - a_r||^2}{2\sigma^2_{(t)}}\right\}\,(1 \leq m, r \leq C)$;

Randomly initialize the matrix $\mathbf{V}^{(t)} = (v^{(t)}_{rj})\,(1 \leq r \leq C; 1 \leq j \leq N)$ such that $\sum_{j=1}^{N} v^{(t)}_{rj} = 1 \forall r$ and $v^{(t)}_{rj} \geq 0 \forall r, j$;

Compute $P^{(t)}_r\,(1 \leq r \leq C)$ to obtain the initial partition $\mathcal{P}^{(t)} = (P^{(t)}_1, \ldots P^{(t)}_C)$ according to Eq. (7);

Compute $J(\mathbf{V}^{(t)}, \mathcal{P}^{(t)})$ according to Eq. (1);

repeat

  set $t \leftarrow t + 1$;

  Compute $\sigma_{(t)} = \sigma_0 \left(\frac{\sigma_f}{\sigma_0}\right)^{\frac{t}{N_{iter}}}$;

  Compute $h_{m,r} = \exp\left\{-\frac{||a_m - a_r||2}{2\sigma^2_{(t)}}\right\}\,(1 \leq m, r \leq C)$;

  Step 1: Representation: Compute the components $v^{(t)}_{rj}$ of the matrix $\mathbf{V}^{(t)}$ according to Eq. (5);

  Step 2: Assignment: Compute $P^{(t)}_r\,(1 \leq r \leq C)$ to obtain the partition $\mathcal{P}^{(t)} = (P^{(t)}_1, \ldots, P^{(t)}_C)$ according to Eq. (7);

  Compute $J(\mathbf{V}^{(t)}, \mathcal{P}^{(t)})$ according to Eq. (1);

until $t = N_{iter}$

The complexity to compute $\boldsymbol{\delta}$ in the worst case is $O(C^2)$. The cost to initialize the matrix $\mathbf{V}^{(t)}$ is $O(NC)$. To determine $\mathcal{P}$ the complexity in the worst-case is $O(N^2C^2)$. Alike, the complexity in (iv) is $O(N^2C)$. Finally, the time complexity of RBSOM-MM**dd** in the initialization step is $O(N^2C^2)$.

- *Representation:* The representation step updates the matrix $\mathbf{V}^{(t)}$ which has a complexity $O(N^2C)$.
- *Assignement:* The assignment step updates the partition $\mathcal{P} = (P_1, \ldots, P_C)$, which has a complexity $O(N^2C^2)$.

The general complexity of the algorithm can be computed, in the worst case, from the complexity of the main three steps taking into account that representation and assignment run N_{iter} iterations. Finally, the time complexity of the RBSOM-MM**dd** algorithm is $O(N_{iter}N^2C^2)$.

III. Experimental setting

This section describes relevant aspects of the experimental setting used to evaluate the proposed method in comparison with Batch SOM algorithm with single Medoid (MEDIAN-BSOM) and RBSOM-SM**dd** state of art medoid based batch SOM algorithms for relational data. The algorithms were implemented in the C language and performed on the same machine (OS: Windows 7 64-bits, Memory: 16 GB, Processor: Intel Core i7-X990 CPU @ 3.47 GHz).

Seventeen (17) datasets from the UCI Machine learning Repository [25], were considered in this study. Table I summarizes these datasets, in which N is the number of objects, P is the number of variables and C is the number of a priori classes.

TABLE I: Dataset characteristics and array size

Dataset	N	P	C	K	Squared array size
BrestTissue	106	9	6	9	3 × 3
Iris	150	4	3	9	3 × 3
Wine	178	13	3	9	3 × 3
Sonar	208	60	2	9	3 × 3
Seeds	210	7	3	9	3 × 3
Glass	214	9	6	9	3 × 3
Thyroid	215	5	3	9	3 × 3
Ecoli	336	7	8	16	4 × 4
Ionosphere	351	32	2	16	4 × 4
Libras	360	90	15	16	4 × 4
Wdbc	569	30	2	16	4 × 4
Pima	768	8	2	25	5 × 5
Statlog	1000	20	2	25	5 × 5
Yeast	1484	8	10	36	6 × 6
Wine-quality	1599	11	6	36	6 × 6
Segmentation	2100	19	7	36	6 × 6
Abalone	4177	8	3	64	8 × 8

For each data set, the Euclidean distance is used to compute a dissimilarity matrix between the objects, taking into account simultaneously all the real-valued variables. Then, the matrices were normalized according to their overall dispersion [26] to have the same dynamic range as follows: each dissimilarity $d(e_k, e_l)\,(1 \leq k, l \leq N)$ in a given dissimilarity matrix $\mathbf{D}$ is normalized as $\frac{d(e_k,e_l)}{T}$, where $T = \sum_{k=1}^{N} d(e_k, g)$ is the overall dispersion and $g = e_l \in E = \{e_1, \ldots, e_N\}$ is the overall representative, which is computed according to $l = arg\min_{1 \leq h \leq N} \sum_{k=1}^{N} d(e_k, e_h)$. One can easily show that after the normalization of $\mathbf{D}$, we have $T = 1$. The considered batch SOM algorithms operate on these normalized dissimilarity matrix.

In this study, the maps are arrays of squared shape and the number of neurons was fixed as $\lfloor\sqrt{N}\rfloor$. Table I also provides K, the number of neurons (clusters) in the maps, and their respective squared array size.

The successful training of the bach SOM algorithms depends on the choice of their parameters [1]–[3]. Different configurations were searched in an unsupervised way without the use of the labels provided by the *a priori* partition through the appropriate combination of parameters.

Table II shows the parameters used with these algorithms. They are four (4) common setting for the compared methods. Each method is executed using either 20 or 50 iterations (N_{iter}). Also, the initial radius σ_0 is computed from the value of the desired initial neighborhood h_0 which, in this paper is

either 0.50 or 0,99. The final value of the neighborhood h_f, used to compute the final radius σ_f, is fixed as 0.01.

TABLE II: General setting

Config	N_{iter}	h_0	h_f
1	20	0,5	0,01
2	20	0,99	0,01
3	50	0,5	0,01
4	50	0,99	0,01

The initial and final value of the radius σ is computed according to the expressions (8) and (9) where x and y correspond to the size of the grid in the horizontal X-axis and vertical Y-axis, respectively.

$$\sigma_0 = \sqrt{\frac{-[(x-1)^2 + (y-1)^2]}{2\ln(h_0)}} \tag{8}$$

$$\sigma_f = \sqrt{\frac{-1}{2\ln(h_f)}} \tag{9}$$

σ is updated in each iteration t from the expression:

$$\sigma_{(t)} = \sigma_0 \left(\frac{\sigma_f}{\sigma_0}\right)^{\frac{t}{N_{iter}}} \tag{10}$$

Moreover, for each dataset it is assumed a square array map of size $\sqrt{K} \times \sqrt{K}$. Specifically, for the RBSOM-SM**dd** method two values for q (the cardinality of the set-medoids) were considered: $\frac{K}{4}$ and $\frac{K}{2}$. The MEDIAN-BSOM has $q = 1$. The parameter n of the RBSOM-MM**dd** method, that controls the level of smoothness of the distribution of prototype weights among all the objects in each of the clusters, was set as 1.1, 1.5 and 2.0.

For a fixed configuration of parameters, the number of epochs was set, by trial and error, in 30 times. The SOM map was trained with each batch SOM algorithm on each dataset. The best training map is selected according to their respective minimum error function. In order to evaluate the quality of the cluster partitions and the training map provided by the batch SOM algorithms, the Overall Error Rate of Classification (OERC) [27] , the topographic error (TE) [28] and the Silhouette (S) [29] were considered.

The $OERC$ index assesses the degree of agreement between an *a priori* partition and a partition provided by the SOM algorithm. This index assumes values on the intervals [0,1] where 0 indicates a perfect agreement between the partitions.

Moreover, the quality of the best training map given by the batch SOM algorithms is measured by the topographic error (TE), computed as follows [28]. Given an object e_k let us denote its BMU with G_r and second BMU with G_s. If these representatives are associated with adjacent neurons, there is no local error; otherwise, there is a local topographic error. The topographic error for the whole mapping is then obtained by summing up the number of local topographic errors for all objects and normalizing [28]:

$$TE = \frac{1}{N}\sum_{k=1}^{N} u(e_k) \tag{11}$$

where N is the number of objects and where $u(e_k) = 1$ if the neurons corresponding to the BMU and second BMU, are non-adjacent, and $u(e_k) = 0$ otherwise. TE assumes values on the interval [0,1] in which a value 0 indicates that there is no local topographic error.

The silhouette coefficient S corresponds to the average of the silhouette scores $s(e_k)$ computed for each object. The silhouette coefficient is as follows:

$$S = \frac{1}{N}\sum_{k=1}^{N} s(e_k). \tag{12}$$

The silhouette score $s(e_k)$ is a measure of how similar an object e_k is to its own cluster (cohesion) compared to other clusters (separation) [29]. The score $s(e_k)$ assumes values on the interval [-1,1], where a high value indicates that the object is well matched to its cluster and poorly matched to neighboring clusters. The score $s(e_k)$ is obtained as

$$s(e_k) = \begin{cases} 1 - \frac{a(e_k)}{b(e_k)}, & if\ a(e_k) < b(e_k) \\ 0 & if\ a(e_k) = b(e_k) \\ \frac{b(e_k)}{a(e_k)} - 1 & if\ a(e_k) > b(e_k) \end{cases} \tag{13}$$

where $a(e_k)$ is the mean distance between e_k and all other objects in the same cluster and $b(e_k)$ is the smallest mean distance of e_k to all objects in any other cluster. The silhouette was calculated from the dissimilarity matrix $\mathbf{D}$.

IV. Results and discussion

This section discusses the performance of the proposed RBSOM-MM**dd** algorithm w.r.t. medoid bases state of art methods RBSOM-SM**dd** [23] and MEDIAN-BSOM [1] algorithms.

Table III shows, for each data set and relational batch SOM algorithm, the overall error rate of classification $OERC$, the topographic error (TE) and the silhouette coefficient (S) for the best hyper-parameter tuning for each method. The tuning of these hyper-parameters was achieved in an unsupervised way, the best SOM map provided by each method was selected based on the best topographic error (TE), the others indexes S and $OERC$ were considered to measure the quality of cluster partitions provided by the methods for their corresponding best SOM map.

It can be observed that RBSOM-SM**dd** presented the best performance according to the $OERC$ index in 6 out of 17 datasets. It also tied in 2 datasets with MEDIAN-BSOM in which they outperformed RBSOM-MM**dd**. Moreover, RBSOM-MM**dd** was the best in 6 and MEDIAN-BSOM in 3 out of 17 datasets. Regarding the TE index, RBSOM-MM**dd** was the best in 9 out of 17 datasets. It also tied in 3 datasets with RBSOM-SM**dd** in which they outperformed MEDIAN-BSOM. In addition, the RBSOM-SM**dd** and MEDIAN-BSOM methods win in 1 dataset each one. All

TABLE III: Quality of the training maps and cluster partitions provided by the batch SOM algorithms

DataSet	Methods					
	Median	Config #	**RBSOM-SMdd**	Config #	**RBSOM-MMdd**	Config #
	BSOM	(q)	*method*	(q)	*method*	(n)
Brest-Tissue	**0,4717** 0,1415 0,5890	2 (q = 1)	0,5472 0,1321 0,7172	4 (q = 5)	0,5472 **0,0755** **0,7689**	4 (n = 1,1)
Iris	0,0733 0,3667 0,3597	2 (q = 1)	**0,0600** 0,2333 0,4527	3 (q = 6)	0,1000 **0,0067** **0,5815**	3 (n = 1,5)
Wine	**0,2697** 0,1629 0,6563	2 (q = 1)	**0,2697** 0,2697 0,6663	1,2 (q = 3)	0,2753 **0,1573** **0,7287**	2 (n = 1.5)
Sonar	0,4423 0,2308 0,1413	1 (q = 1)	0,4087 0,1779 0,1882	2 (q = 7)	**0,3510** **0,1202** **0,1927**	4 (n = 2,0)
Seeds	0,1190 0,2095 0,4844	4 (q = 1)	0,1048 0,1000 **0,4850**	1 (q = 7)	**0,1000** **0,0857** 0,4696	1 (n = 1.5)
Glass	0,3972 0,2150 0,3796	1 (q = 1)	**0,3879** **0,0607** 0,4752	2 (q = 7)	0,4393 **0,0607** **0,4798**	2 (n = 1,1)
Thyroid	0,1674 0,4837 0,1920	2 (q = 1)	0,1209 0,3070 0,4158	1 (q = 7)	**0,1163** **0,0372** **0,4643**	2 (n = 1,5)
Ecoli	**0,2530** **0** 0,3271	1,2,3,4 (q = 1)	0,2649 **0** 0,3301	1,2,4 (q = 5)	0,2798 **0** **0,3522**	2 (n = 1,5)
Ionos-phere	0,2963 **0** 0,1170	2 (q = 1)	0,3020 **0** **0,1234**	2 (q = 5)	**0,1624** **0** 0,1049	2 (n = 1,1)
Libras	**0,7861** 0,0028 0,1781	4 (q = 1)	0,7917 **0** 0,1699	1,2,4 (q = 9)	0,7944 **0** **0,1832**	4 (n = 1,1)
Wdbc	**0,1160** **0** 0,4263	2 (q = 1)	**0,1160** **0** **0,4265**	1 (q = 6)	0,1178 **0** 0,4128	1 (n = 1,1)
Pima	0,3060 0,0859 0,2503	2 (q = 1)	**0,2839** 0,1419 0,1728	4 (q = 14)	0,3333 **0,0078** **0,3458**	4 (n = 1,5)
Statlog	0.2910 0.2790 0.2338	4 (q = 1)	**0,2890** **0.1860** **0.2587**	2 (q = 28)	0,2910 0.1900 0.2385	2 (n = 1,5)
yeast	0,5728 **0** **0,2429**	1 (q = 1)	0,5660 0,0007 0,1923	2 (q = 19)	**0,5613** 0,0007 0,1856	2 (n = 1,1)
wine-quality	0,5222 0,0013 **0,4198**	4 (q = 1)	**0,5210** **0** 0,4160	1 (q = 10)	0,5228 **0** 0,4120	2 (n = 1,1)
segmen-tation	0,5462 0,0019 **0,3348**	1, 2 (q = 1)	0,5619 0,0005 0,3194	2 (q = 11)	**0,5233** **0** 0,3120	1 (n = 1,1)
abalone	0,4779 0,0077 0,3890	4 (q = 1)	**0,4740** 0,0055 0,2504	1 (q=16)	0,4838 **0** **0,3984**	2 (n=1,5)

methods tied in 3 datasets. Finally, RBSOM-MM**dd** was also the best regarding the S index in 10 out of 17 datasets followed by RBSOM-SM**dd** (4 out 17) and MEDIAN-BSOM in 3 out of 17 datasets.

Concerning the configurations of Table. II, one can observe that, whatever the batch SOM algorithm considered, configurations 2 and 3 provide, respectively, the best and the worst TE for the majority of the datasets. Moreover, as high is parameter q as better is the TE provided by RBSOM-SM**dd**. Finally, RBSOM-MM**dd** has better values of TE for small values of parameter n.

The Friedman test [30] was applied to the results aiming to test if the observed differences in performance among the methods and indexes were statistically significant. The test rejects the null hypothesis which states that all the algorithms are equivalent regarding the TE index. The application of the Nemenyi and Bonferroni post-tests with $\alpha = 0.05$ shows that the RBSOM-MM**dd** algorithm significantly outperformed the MEDIAN-BSOM algorithm. The critical difference (CD) determined by these post-tests were 0.8039 and 0.7687 respectively. Fig. 1 show the compared methods concerning TE. The best methods correspond to the lower values of ranks. Moreover,

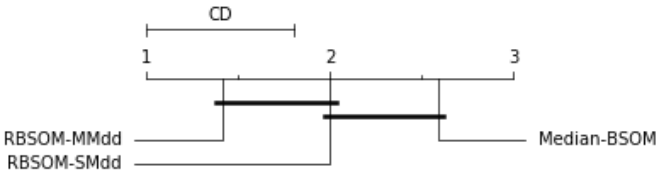


Fig. 1: TE. Comparison of methods using the Friedman test in combination with Nemenyi post-test.

the same tests cannot reject the null hypothesis which states that all the algorithms are equivalent regarding $OERC$ and S indexes. However, the best ranks also correspond to the RBSOM-MM**dd** method with respect to S index.

A. The Ionosphere dataset

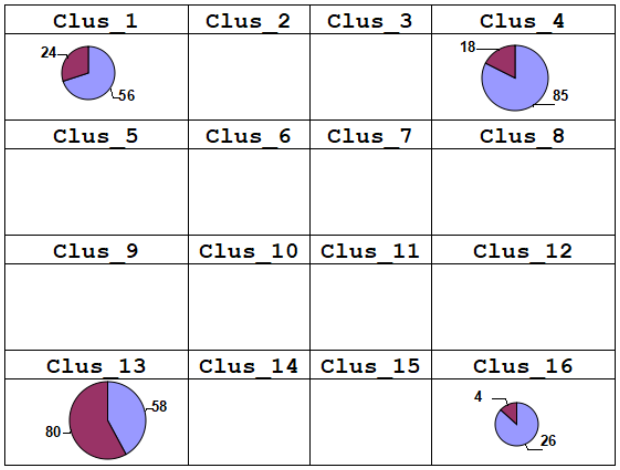
This section provides more detailed results for the ionosphere dataset in order to illustrate the usefulness of the proposed method.

The ionosphere dataset [31] consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this dataset are described by 2 attributes per pulse number [31]. The *priori* classes have 225 "Good" and 126 "Bad" instances.

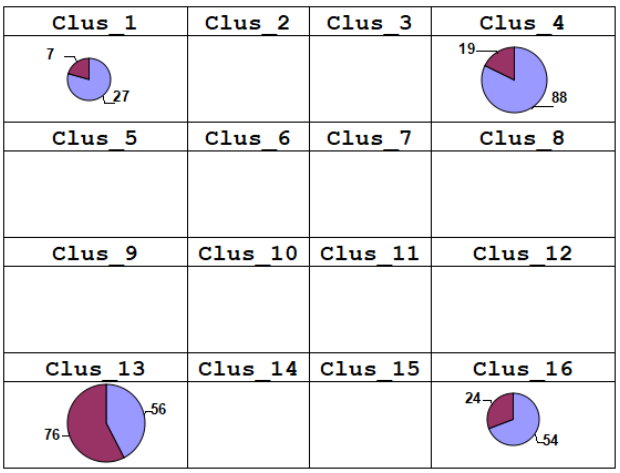
Table IV shows the confusion matrices (Class x Cluster) for the methods a) MEDIAN-BSOM, b) RBSOM-SM**dd**, and c) RBSOM-MM**dd** with the corresponding better parameter setting achieved on the ionosphere dataset.

TABLE IV: Confusion Matrix of the compared methods.

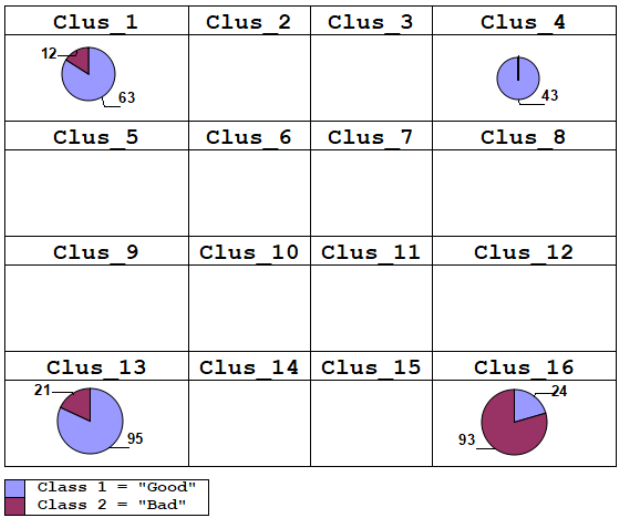
a)																
56	0	0	85	0	0	0	0	0	0	0	0	58	0	0	26	225
24	0	0	18	0	0	0	0	0	0	0	0	80	0	0	4	126
80	0	0	103	0	0	0	0	0	0	0	0	138	0	0	30	351
b)																
27	0	0	88	0	0	0	0	0	0	0	0	56	0	0	54	225
7	0	0	19	0	0	0	0	0	0	0	0	76	0	0	24	126
34	0	0	107	0	0	0	0	0	0	0	0	132	0	0	78	351
c)																
63	0	0	43	0	0	0	0	0	0	0	0	95	0	0	24	225
12	0	0	0	0	0	0	0	0	0	0	0	21	0	0	93	126
75	0	0	43	0	0	0	0	0	0	0	0	116	0	0	117	351



(a)



(b)



(c)

Fig. 2: SOM maps provided by (a) MEDIAN-BSOM, (b) RBSOM-SMdd and (c) RBSOM-MMdd methods on ionosphere dataset

In Table IV, columns 1 to 16 correspond to the clusters, the rows 1 to 2 correspond to the *a priori* classes. Except for the last row and last column, the cells provide the number of objects shared by the clusters and the *a priori* classes. Moreover, the cells of the last row gives the number of objects in each cluster. Finally, the cells of the last column give the number of objects, respectively, for the first *a priori* class (row 1), for the second *a priori* class (row 2) and the total of objects (row 3).

Fig.2, displays the repartition of the objects over the 16 clusters on the 4×4 grid provided by (a) MEDIAN-BSOM, (b) RBSOM-SMdd and (c) RBSOM-MMdd on the ionosphere dataset. Each node (Clus_X) represents a cluster (neuron). Besides, the circle size is proportional to the number of objects of the cluster. Also, the total area of the circle is shared between the two areas corresponding to the *a priori* classes "Good" and "Bad". Finally, it is indicated also the number of objects in each *a priori* class among the objects of the corresponding cluster.

It can be observed that in the SOM maps and whatever the considered algorithm, only clusters 1, 4, 13 and 16 are not empty. Moreover, the clusters with the majority of the objects belonging to *a priori* class "Good" are 1, 4 and 16 in the SOM map provided by the algorithms RBSOM-MMdd and RBSOM-MMdd, and are 1, 4 and 13 in the SOM map provided by the algorithm RBSOM-MMdd. Besides, the cluster with the majority of the objects belonging to *a priori* class "Bad" is 13 in the SOM map provided by the algorithms RBSOM-MMdd and RBSOM-MMdd, and is 16 in the SOM map provided by the algorithm RBSOM-MMdd. Therefore, MEDIAN-BSOM and RBSOM-SMdd algorithms provided SOM maps that from the top of the grid moving down to the bottom right side goes from *a priori* class "Good" to *a priori* class "Bad". Moreover, RBSOM-MMdd provided a SOM map that from the top of the grid moving down to the bottom left side goes from *a priori* class "Good" to *a priori* class "Bad". Besides, it can be observed that the clusters produced by RBSOM-MMdd algorithm are more homogeneous than those produced by MEDIAN-BSOM and RBSOM-SMdd algorithms. Finally, the considered algorithms provided SOM maps of similar quality as measure by TE.

V. Conclusions

The great majority of previous works on SOM concern quantitative vectorial data. Relatively few SOM algorithms were designed to manage relational data despite their importance in practical applications. In this paper we proposed RBSOM-MMdd, a batch training SOM algorithm for relational data. RBSOM-MMdd minimizes a cost function aiming to group and visualizing the data while preserving the spatial order of the neurons on the map. RBSOM-MMdd is designed to provide a map for visualization purposes, a partition in a fixed number of clusters and the vectors of prototype weights representatives of the clusters. For a fixed neighborhood radius, based on the minimization of a suitable cost function, the proposed learning algorithm performs

interactively two steps: the representation step, where it is computed the cluster representatives, and the assignment step, where it is updated the cluster partition. The paper provided the optimal solution for these two steps. Moreover, the paper also provides the time complexity of the RBSOM-MMdd algorithm.

Experimental evaluations of the RBSOM-MMdd algorithm, in comparison with relevant medoid-based batch SOM algorithms MEDIAN-BSOM and RBSOM-SMdd for relational data were carried out on seventeen datasets from the UCI Machine learning Repository. The original datasets were duly transformed to obtain a normalized dissimilarity matrix as a representation of each base. The degree of dissimilarity between the objects in the matrix was computed using the Euclidean distance.

Several configurations of hyper-parameters were considered for each algorithm. For a fixed configuration of parameters, the SOM map was trained 30 times with each batch SOM algorithm on each dataset. The best training map was selected according to their respective minimum error function. In order to evaluate the quality of the cluster partitions and the training map provided by the batch SOM algorithms were computed $OERC$, TE and S indexes for the best hyper-parameter tuning for each method achieved in an unsupervised way.

It was observed that in the majority of the data sets considered, RBSOM-MMdd outperformed RBSOM-SMdd and MEDIAN-BSOM concerning TE index. The Friedman test and Nemenyi post-test confirmed that the proposed method was significantly better regarding TE. Moreover, RBSOM-MMdd presented also the best performance concerning S and $OERC$ indexes. Finally, the usefulness of the RBSOM-MMdd algorithm was illustrated with its application on the ionosphere dataset.

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees and to the Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco - FACEPE (IBPG-0820-1.03/19) and Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (303187/2013-1) for their financial support.

REFERENCES

[1] T. Kohonen, *Self-Organizing Maps*, ser. Springer Series in Information Sciences. Springer, 1995, vol. 30.

[2] F. Badran, M. Yacoub, and S. Thiria, "Self-organizing maps and unsupervised classification," in *Neural networks*. Springer, 2005, pp. 379–442.

[3] T. Kohonen, "Essentials of the self-organizing map," *Neural networks*, vol. 37, pp. 52–65, 2013.

[4] C. A. Astudillo and B. J. Oommen, "Topology-oriented self-organizing maps: a survey," *Pattern analysis and applications*, vol. 17, no. 2, pp. 223–248, 2014.

[5] M. Cottrell, M. Olteanu, F. Rossi, and N. Villa-Vialaneix, "Theoretical and applied aspects of the self-organizing maps," in *Advances in Self-Organizing Maps and Learning Vector Quantization - Proceedings of the 11th International Workshop WSOM 2016, Houston, Texas, USA, January 6-8, 2016*, 2016, pp. 3–26.

[6] ——, "Self-organizing maps, theory and applications," *Revista Investigatión Operational*, vol. 39, no. 1, pp. 1–22, 2018.

[7] S. kaski, K. Jari, and T. Kohonen, "Bibliography of self-organizing map (som) papers: 1981–1997," *Neural Computing Surveys*, vol. 1, pp. 1–176, 1998.

[8] M. Oja, S. kaski, , and T. Kohonen, "Bibliography of self-organizing map (som) papers: 1998–2001 addendum," *Neural Computing Surveys*, vol. 3, no. 1, pp. 1–156, 2003.

[9] T. Kohonen, "Comparison of SOM point densities based on different criteria," *Neural Computation*, vol. 11, no. 8, pp. 2081–2095, 1999.

[10] H. Frigui, C. Hwang, and F. C. Rhee, "Clustering and aggregation of relational data with applications to image database categorization," *Pattern Recognition*, vol. 40, no. 11, pp. 3053–3068, 2007.

[11] B. Conan-Guez, F. Rossi, and A. E. Golli, "Fast algorithm and implementation of dissimilarity self-organizing maps," *Neural Networks*, vol. 19, no. 6-7, pp. 855–863, 2006.

[12] R. J. Hathaway and J. C. Bezdek, "Nerf *c*-means: Non-euclidean relational fuzzy clustering," *Pattern Recognition*, vol. 27, no. 3, pp. 429–437, 1994.

[13] M. Olteanu, N. Villa-Vialaneix, and M. Cottrell, "On-line relational SOM for dissimilarity data," in *Advances in Self-Organizing Maps - 9th International Workshop, WSOM 2012, Santiago, Chile, December 12-14, 2012, Proceedings*, 2012, pp. 13–22.

[14] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids. statistical data analysis based on the l1 norm," *Y. Dodge, Ed*, pp. 405–416, 1987.

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

[16] R. Krishnapuram, A. Joshi, and L. Yi, "A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering," in *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No. 99CH36315)*, vol. 3, 1999, pp. 1281–1286.

[17] F. A. T. de Carvalho, Y. Lechevallier, and F. M. de Melo, "Partitioning hard clustering algorithms based on multiple dissimilarity matrices," *Pattern Recognition*, vol. 45, no. 1, pp. 447–464, 2012.

[18] J. Mei and L. Chen, "Fuzzy clustering with weighted medoids for relational data," *Pattern Recognition*, vol. 43, no. 5, pp. 1964–1974, 2010.

[19] R. J. Hathaway, J. W. Davenport, and J. C. Bezdek, "Relational duals of the c-means clustering algorithms," *Pattern Recognition*, vol. 22, no. 2, pp. 205–212, 1989.

[20] F. A. T. de Carvalho, Y. Lechevallier, and F. M. de Melo, "Relational partitioning fuzzy clustering algorithms based on multiple dissimilarity matrices," *Fuzzy Sets and Systems*, vol. 215, pp. 1–28, 2013.

[21] J. Mei and L. Chen, "Fuzzy relational clustering around medoids: A unified view," *Fuzzy Sets and Systems*, vol. 183, no. 1, pp. 44–56, 2011.

[22] T. Kohonen and P. Somervuo, "Self-organizing maps of symbol strings," *Neurocomputing*, vol. 21, no. 1-3, pp. 19–30, 1998.

[23] A. E. Golli, B. Conan-Guez, and F. Rossi, *Classification, Clustering, and Data Mining Applications - Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Chicago, USA, 15–18 July 2004*. Springer, 2005, ch. A Self-Organizing Map for Dissimilarity Data, pp. 61–68.

[24] A. Hasenfuss and B. Hammer, "Relational topographic maps," in *Advances in Intelligent Data Analysis VII, 7th International Symposium on Intelligent Data Analysis, IDA 2007, Ljubljana, Slovenia, September 6-8, 2007, Proceedings*, 2007, pp. 93–105.

[25] C. Black and C. Merz, "UCI Repository of machine learning databases," http://www.ics.uci.edu/ mlearn/MLRepository.html, Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[26] M. Chavent, "Normalized k-means clustering of hyper-rectangles," in *Proceedings of the XI International Symposium of Applied Stochastic Models and Data Analysis (ASMDA 2005)*, vol. 1, 2005, pp. 670–677.

[27] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[28] K. Kiviluoto, "Topology preservation in self-organizing maps," in *Proceedings of International Conference on Neural Networks (ICNN'96), Washington, DC, USA, June 3-6, 1996*, 1996, pp. 294–299.

[29] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[30] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[31] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml