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. 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 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 \( D = [d_{kl}] \), 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-
In comparison with previous batch SOM methods for relational data [1], [23], the new contributions of RBSOM-MMdd 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-MMdd, that is able to manage relational data. It is based on fuzzy clustering algorithm with multi-medoids (FMdd) and Batch SOM algorithm with c-Medoids for relational data (RBSOM-SMdd).

Let \( E = \{ e_1, \ldots, e_N \} \) be the set of objects and let a \( N \times N \) dissimilarity matrix \( D = (d_{kl}) \) 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 \( D \).

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) \( v_r \). In this paper, the representative \( v_r = (v_{r1}, \ldots, v_{rN}) \) 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 \( P = \{ P_1, \ldots, P_C \} \) associated to a SOM is defined by the allocation function which gives the index of the cluster of \( P \) 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 \( V = (v_{rj}) (1 \leq l \leq C; 1 \leq j \leq N) \) of prototype weights of the objects with respect to the clusters [21], where \( v_r = (v_{r1}, \ldots, v_{rN}) \) 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 \( P \) of \( E \) into \( C \) clusters represented by \( P = \{ P_1, \ldots, P_C \} \).
Starting from an initial solution, and aiming that the obtained SOM be representative of the relationships among the object provided by the dissimilarity matrix $D$, the matrix $V$ 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:

$$ J(V, \mathcal{P}) = \sum_{k=1}^{N} \Delta(e_k, v_f(e_k)) $$  

(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 $v_f(e_k)$. The error measure of a BMU $v_f(e_k)$ with respect to the object $e_k$ is computed by means of the dissimilarity function $\Delta$ that enables to compare each object $e_k$ to each prototype $v_r$ according to:

$$ \Delta(e_k, v_f(e_k)) = \sum_{r=1}^{C} h_f(e_k, r) d(e_k, v_r) $$  

(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

$$ h_f(e_k, r) = \exp \left\{ -\frac{||a_f(e_k) - a_r||^2}{2\sigma^2} \right\} $$  

(3)

where $a_f(e_k)$ and $a_r$ are, respectively, the BMU and neuron $r$ positions in the grid. Besides, $\sigma$ is the neighborhood radius. The size of the neighborhood decreases with $\sigma$: the smaller $\sigma$, the fewer the neurons that belong to the effective neighborhood of a given BMU.

The function $d(e_k, v_r)$ computes the dissimilarity between an object $e_k$ and a cluster representative $v_r$ and it is defined as

$$ d(e_k, v_r) = \sum_{j=1}^{N} (v_{rj})^\alpha d(e_k, e_j) $$  

(4)

In Eq. (4), the parameter $\alpha$ 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-MMdd algorithm

When the radius $\sigma$ is kept fixed, from an initial solution the training map of the RBSOM-MMdd 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 $V = (v_r)$ ($1 \leq r \leq C; 1 \leq j \leq N$) 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 $J(V, \mathcal{P})$ is minimized with respect to the matrix $V = (v_r)$ 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:

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

(5)

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

$$ v_{rj} = \left[ \sum_{s=1}^{N} \frac{\sum_{k=1}^{N} h_f(e_k, r) d(e_k, e_j)}{\sum_{k=1}^{N} h_f(e_k, r) d(e_k, e_j) + \beta_r} \right]^{-1} $$  

(6)

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

3) Assignment: During the assignment step, the matrix $V$ of prototype weights is kept fixed. The cost function $J(V, \mathcal{P})$ 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 $\mathcal{P}_r$ are updated as follows:

$$ \mathcal{P}_r = \left\{ e_k \in E : \Delta(e_k, v_r) = \min_{1 \leq s \leq C} \Delta(e_k, v_s) \right\} $$  

(7)

where $\Delta$ 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 $\delta$, (ii) initialization of the matrix $V^{(i)}$, (iii) the setting of the initial partition $\mathcal{P} = (P_1, \ldots, P_C)$, and (iv) computation the objective function $J(V, \mathcal{P})$ considering the initials matrix $V^{(i)}$ and partition $\mathcal{P}$.
The complexity to compute $\delta$ in the worst case is $O(C^2)$. The cost to initialize the matrix $V^{(t)}$ is $O(NC)$. To determine $P$, the complexity in the worst-case is $O(N^2C^2)$. Likewise, the complexity in (iv) is $O(N^2C^2)$. Finally, the time complexity of RBSOM-MMdd in the initialization step is $O(N^2C^2)$.

- **Representation**: The representation step updates the matrix $V^{(t)}$ which has a complexity $O(N^2C^2)$.
- **Assignment**: The assignment step updates the partition $P = (P_1, \ldots, P_C)$, 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-MMdd 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-BSON) and RBSOM-SMdd 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>
<thead>
<tr>
<th>Dataset</th>
<th>$N$</th>
<th>$P$</th>
<th>$C$</th>
<th>Squared array size</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrestTissue</td>
<td>106</td>
<td>9</td>
<td>6</td>
<td>$9 \times 3 \times 3$</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
<td>$9 \times 3 \times 3$</td>
</tr>
<tr>
<td>Wine</td>
<td>178</td>
<td>13</td>
<td>3</td>
<td>$9 \times 3 \times 3$</td>
</tr>
<tr>
<td>Sonar</td>
<td>208</td>
<td>60</td>
<td>2</td>
<td>$9 \times 3 \times 3$</td>
</tr>
<tr>
<td>Seeds</td>
<td>210</td>
<td>7</td>
<td>3</td>
<td>$9 \times 3 \times 3$</td>
</tr>
<tr>
<td>Glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
<td>$9 \times 3 \times 3$</td>
</tr>
<tr>
<td>Thyroid</td>
<td>215</td>
<td>5</td>
<td>3</td>
<td>$9 \times 3 \times 3$</td>
</tr>
<tr>
<td>Ecoli</td>
<td>336</td>
<td>7</td>
<td>8</td>
<td>$16 \times 4 \times 4$</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>32</td>
<td>2</td>
<td>$16 \times 4 \times 4$</td>
</tr>
<tr>
<td>Libras</td>
<td>360</td>
<td>90</td>
<td>15</td>
<td>$16 \times 4 \times 4$</td>
</tr>
<tr>
<td>Wdbc</td>
<td>569</td>
<td>30</td>
<td>2</td>
<td>$16 \times 5 \times 5$</td>
</tr>
<tr>
<td>Pima</td>
<td>768</td>
<td>8</td>
<td>2</td>
<td>$25 \times 5 \times 5$</td>
</tr>
<tr>
<td>Statlog</td>
<td>1000</td>
<td>20</td>
<td>2</td>
<td>$25 \times 5 \times 5$</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>8</td>
<td>10</td>
<td>$36 \times 6 \times 6$</td>
</tr>
<tr>
<td>Wine-quality</td>
<td>1599</td>
<td>11</td>
<td>6</td>
<td>$36 \times 6 \times 6$</td>
</tr>
<tr>
<td>Segmentation</td>
<td>2100</td>
<td>19</td>
<td>7</td>
<td>$36 \times 6 \times 6$</td>
</tr>
<tr>
<td>Abalone</td>
<td>4177</td>
<td>8</td>
<td>3</td>
<td>$64 \times 8 \times 8$</td>
</tr>
</tbody>
</table>

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 $D$ is normalized as $d(e_k, e_l) / T$, 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 $l = \arg\min_{1 \leq h \leq N} \sum_{l=1}^{N} d(e_k, e_h)$. One can easily show that after the normalization of $D$, 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 $\sqrt{N}$. Table I also provides $K$, the number of neurons (clusters) in the maps, and their respective squared array size.

The successful training of the batch 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 $\sigma_0$ 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 \( \sigma_f \), is fixed as 0.01.

### TABLE II: General setting

<table>
<thead>
<tr>
<th>Config</th>
<th>( N_{iter} )</th>
<th>( h_0 )</th>
<th>( h_f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>0.5</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>0.99</td>
<td>0.01</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>0.5</td>
<td>0.01</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>0.99</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The initial and final value of the radius \( \sigma \) 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.

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

\[
\sigma_f = \sqrt{-\frac{1}{2 \ln(h_f)}} \quad (9)
\]

\( \sigma \) is updated in each iteration \( t \) from the expression:

\[
\sigma(t) = \sigma_0 \left( \frac{\sigma_f}{\sigma_0} \right)^{\frac{1}{N_{iter}}} \quad (10)
\]

Moreover, for each dataset it is assumed a square array map of size \( \sqrt{K} \times \sqrt{K} \). Specifically, for the RBSOM-SMdd 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-MMdd 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 \( G_i \) 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]:

\[
TE = 1 - \frac{1}{N} \sum_{k=1}^{N} u(e_k) \quad (11)
\]

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:

\[
S = \frac{1}{N} \sum_{k=1}^{N} s(e_k). \quad (12)
\]

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

\[
s(e_k) = \begin{cases} 1 - \frac{a(e_k)}{b(e_k)}, & \text{if } a(e_k) < b(e_k) \\ 0, & \text{if } a(e_k) = b(e_k) \\ \frac{b(e_k)}{a(e_k)} - 1, & \text{if } a(e_k) > b(e_k) \end{cases} \quad (13)
\]

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 \( D \).

### IV. RESULTS AND DISCUSSION

This section discusses the performance of the proposed RBSOM-MMdd algorithm w.r.t. medoid bases state of art methods RBSOM-MMdd [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-MMdd 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-MMdd. Moreover, RBSOM-MMdd was the best in 6 and MEDIAN-BSOM in 3 out of 17 datasets. Regarding the \( TE \) index, RBSOM-MMdd was the best in 9 out of 17 datasets. It also tied in 3 datasets with RBSOM-MMdd in which they outperformed MEDIAN-BSOM. In addition, the RBSOM-MMdd and MEDIAN-BSOM methods win in 1 dataset each one. All
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 α = 0.05 shows that the RBSOM-MMdd 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, 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-MMdd 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

methods tied in 3 datasets. Finally, RBSOM-MMdd was also the best regarding the S index in 10 out of 17 datasets followed by RBSOM-SMdd (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-MMdd. Finally, RBSOM-MMdd has better values of TE for small values of parameter n.
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 $4 \times 4$ 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-MM algorithm.

Experimental evaluations of the RBSOM-MM 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


