Batch self-organizing maps based on city-block distances for interval variables

Francisco de A. T. de Carvalho, Patrice Bertrand, Filipe M. de Melo

Batch self-organizing maps based on city-block distances for interval variables

Francisco de A. T. de Carvalho[1], Patrice Bertrand[2], and Filipe M. de Melo[3]

[1]Universidade Federal de Pernambuco, Brazil
[2]Ceremade, Université Paris-Dauphine, France
[3]Universidade Federal de Pernambuco, Brazil

June 10, 2012

Abstract

The Kohonen Self Organizing Map (SOM) is an unsupervised neural network method with a competitive learning strategy which has both clustering and visualization properties. Interval-valued data arise in practical situations such as recording monthly interval temperatures at meteorological stations, daily interval stock prices, etc. Batch SOM algorithms based on adaptive and non-adaptive city-block distances, suitable for objects described by interval-valued variables, that, for a fixed epoch, optimizes a cost function, are presented. The performance, robustness and usefulness of these SOM algorithms are illustrated with real interval-valued data sets.

Keywords: Self-organizing maps, Interval-valued data, City-block distances, Adaptive distances, Symbolic data analysis.

1 Introduction

Clustering is a popular task in knowledge discovery, and it is applied in various fields including data mining, pattern recognition, computer vision, etc. Clustering methods aim at organizing a set of items into clusters such that items within a given cluster have a high degree of similarity, while items belonging to different clusters have a high degree of dissimilarity. The most popular clustering techniques are hierarchical and partitional methods [14, 18]. K-means algorithm and fuzzy c-means are the most famous partitional approaches.

The Kohonen Self Organizing Map (SOM) [17] is an unsupervised neural network method with a competitive learning strategy which has both clustering and visualization properties. Different from K-means, SOM uses the neighborhood interaction set to approximate lateral neural interaction and discover the topological structure hidden in the data, and in addition to the best matching referent vector (winner), its neighbors on the map are updated, resulting in regions where neurons in the same neighborhood are very similar. It can be considered as an algorithm that maps a high dimensional data space to lattice space which usually has a lower

dimension (typically, dimension two) and is called a map. This projection enables a partition of the inputs into "similar" clusters while preserving their topology. The map training can be incremental or batch.

This paper gives batch SOM algorithms to manage individuals described by interval-valued variables. Interval-valued variables are needed, for example, when an object represents a group of individuals and the variables used to describe it need to assume a value which express the variability inherent to the description of a group.

Interval-valued data arise in practical situations such as recording monthly interval temperatures at meteorological stations, daily interval stock prices, etc. Another source of interval-valued data is the aggregation of huge databases into a reduced number of groups, the properties of which are described by interval-valued variables. Therefore, tools for interval-valued data analysis [3, 8] are very much required.

In [2, 9] it is presented incremental SOM algorithms that are able to manage interval-valued data. More recently, batch SOM based on non-adaptive [6, 11] and adaptive Euclidean distances [6] as well as non-adaptive city-block distances [12], have been presented.

This paper aims at proposing batch SOM algorithms based on adaptive and non-adaptive city-block distances, suitable for objects described by interval-valued variables, that, for a fixed epoch, optimizes a cost function. The performance, robustness and usefulness of these SOM algorithms are illustrated with real interval-valued data sets, in comparison with batch SOM algorithms based on adaptive and non-adaptive Euclidean distances.

2 Batch self-organizing maps based on city-block distances

This section presents batch SOM algorithms based on adaptive (ABSOM-L1) and non-adaptive (BSOM-L1) city-block distances for interval-valued data. They are based on the batch SOM algorithm based on the Euclidean distances for real-valued data [1].

Let $E = \{e_1, \ldots, e_n\}$ be a set of n objects indexed by i and described by p interval-valued variables indexed by j. An interval-valued variable X [3] is a correspondence defined from E in $\Im$ such that for each $e_i \in \Omega, X(e_i) = [a, b] \in \Im$, where $\Im$ is the set of closed intervals defined from $\Re$. Each object e_i is represented as a vector of intervals $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$, where $x_{ij} = [a_{ij}, b_{ij}] \in \Im = \{[a, b] : a, b \in \Re,\ a \leq b\}$.

A basic assumption of the presented batch SOM algorithms is that a prototype $\mathbf{w}_r$ of cluster $P_r\,(r = 1. \ldots, m)$ is also represented as a vector of intervals $\mathbf{w}_r = (w_{r1}, \ldots, w_{rp})$, where $w_{rj} = [\alpha_{rj}, \beta_{rj}] \in \Im\,(j = 1, \ldots, p)$.

The IBSOM-L1 algorithm is an iterative two-step algorithm (representation and affectation steps) in which the whole data set E is presented to the map before any adjustments are made. The IBSOM-L1 algorithm minimizes the following cost function:

$$J = \sum_{i=1}^{n} \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r))\, d(\mathbf{x}_i, \mathbf{w}_r) \tag{1}$$

where f is the allocation function and $f^T(\mathbf{x}_i)$ stands for the neuron of the map that is associated to the object $\mathbf{x}_i$, and $\delta(f^T(\mathbf{x}_i), r))$ is the distance on the map between a neuron r and the neuron that is allocated to the object $\mathbf{x}_i$. Moreover, K^T, that is parameterized by T (where T stands

for temperature), is the neighborhood kernel function that defines influence region around each neuron r.

The generalized dissimilarity function between an input data $\mathbf{x}_i$ and a prototype $\mathbf{w}_{f^T(\mathbf{x}_i)}$ is given by:

$$d^T(\mathbf{x}_i, \mathbf{w}_{f^T(\mathbf{x}_i)}) = \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r))\, d(\mathbf{x}_i, \mathbf{w}_r) \tag{2}$$

where

$$d(\mathbf{x}_i, \mathbf{w}_r) = \sum_{j=1}^{p} [|a_{ij} - \alpha_{rj}| + |b_{ij} - \beta_{rj}|] \tag{3}$$

is a suitable city-block distance between vectors of intervals. This generalized distance is a weighted sum of the city-block distances between $\mathbf{x}_i$ and all the reference vectors of the neighborhood of the neuron $f^T(\mathbf{x}_i)$. It takes into account all the neurons of the map.

The IABSOM-L1 algorithm is an iterative three-step algorithm (representation, weighting and affectation steps) in which the whole data set E is also presented to the map before any adjustments are made. The cost function of the IABSOM-L1 algorithm is given by:

$$J = \sum_{i=1}^{n} \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r))\, d_{\boldsymbol{\lambda}_r}(\mathbf{x}_i, \mathbf{w}_r) \tag{4}$$

The generalized dissimilarity function between an input data $\mathbf{x}_i$ and a prototype $\mathbf{w}_{f^T(\mathbf{x}_i)}$ is given by:

$$d^T_{\boldsymbol{\Lambda}}(\mathbf{x}_i, \mathbf{w}_{f^T(\mathbf{x}_i)}) = \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r))\, d_{\boldsymbol{\lambda}_r}(\mathbf{x}_i, \mathbf{w}_r) \tag{5}$$

where

$$d_{\boldsymbol{\lambda}_r}(\mathbf{x}_i, \mathbf{w}_r) = \sum_{j=1}^{p} \lambda_{rj} [|a_{ij} - \alpha_{rj}| + |b_{ij} - \beta_{rj}|] \tag{6}$$

is an adaptive city-block distance parameterized by the vector of weights on the variables $\boldsymbol{\lambda}_r = (\lambda_{r1}, \ldots, \lambda_{rp})$ and $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m)$.

Note that the weight vectors $\boldsymbol{\lambda}_r\, (r = 1, \ldots, m)$ change at each iteration, i.e., they are not determined absolutely, and are different from one neuron to another.

When T is kept fixed, for IBSOM-L1 algorithm, the minimization of J is performed iteratively in two steps (representation and allocation), whereas for IABSOM-L1, the minimization of J is performed iteratively in three steps (representation, weighting and affectation).

During the representation step of IBSOM-L1, the allocation function is kept fixed. The cost function J is minimized with respect to the prototypes. During the representation step of IABSOM-L1, the allocation function and the vectors of weights are kept fixed. The cost function J is also minimized with respect to the prototypes. For both algorithms, the components $w_{rj}\, (j = 1, \ldots, p)$ of the prototype $\mathbf{w}_r\, (r = 1, \ldots, m)$ are such that they minimize

$$\sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), r))[|a_{ij} - \alpha_{rj}| + |b_{ij} - \beta_{rj}|]$$

This problem brings to the minimization of $\sum_{i=1}^{n} |y_i - az_i|$, where $y_i = K^T(\delta(f^T(\mathbf{x}_i), r))\, a_{ij}$ (or, equivalently, $y_i = K^T(\delta(f^T(\mathbf{x}_i), r))\, b_{ij}$), $z_i = K^T(\delta(f^T(\mathbf{x}_i), r))$ and $a = \alpha_{rj}$ (or, equivalently, $a = \beta_{rj}$). The solution of this problem is known and to solve it the following algorithm can be used [15, 16]:

1. Determining $u_i = y_i/z_i\, (i = 1, \ldots, n)$;
2. Rearrange the z_i's according to ascending order of u_i's and get $\tilde{z}_1, \ldots, \tilde{z}_n$;
3. Minimize $\sum_{l=1}^{r} |\tilde{z}_l| - \sum_{l=r+1}^{n} |\tilde{z}_l|$ regarding r;
4. If the minimum is negative, take $a = u_r$. If the minimum is positive, take $a = u_{r+1}$. Finally, if the minimum is equal to zero, take $a = (u_r + u_{r+1})/2$.

Theorem 2.1. *The boundaries of the intervals* $w_{rj} = [\alpha_{rj}, \beta_{rj}]\, (r = 1, \ldots, K; j = 1, \ldots, p)$ *are such that* $\alpha_{rj} \leq \beta_{rj}$.

Proof. The proof is given in appendix. □

During the weighting step of IABSOM-L1, the reference vectors (prototypes) and the allocation function are kept fixed. The cost function J is minimized with respect to the vectors of weights. The computation of these vectors of weights in this algorithm is also inspired from the approach used to compute a weight for each variable in each cluster in the dynamic clustering algorithm based on adaptive distances [7].

The vectors of weights $\boldsymbol{\lambda}_r = (\lambda_{r1}, \ldots, \lambda_{rp})\, (r = 1, \ldots, m)$, under $\lambda_{rj} > 0$ and $\prod_{j=1}^{p} \lambda_{rj} = 1$, have their weights $\lambda_{rj}\, (j = 1, \ldots, p)$ calculated according to the following expression:

$$\lambda_{rj} = \frac{\{\prod_{h=1}^{p} \sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), h))[|a_{ij} - \alpha_{rh}| + |b_{ij} - \beta_{rh}|]\}^{\frac{1}{p}}}{\sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), r))[|a_{ij} - \alpha_{rj}| + |b_{ij} - \beta_{rj}|]} \tag{7}$$

During the affectation step of IBSOM-L1, the reference vectors (prototypes) are kept fixed. The cost function J is minimized with respect to the allocation function and each individual $\mathbf{x}_i$ is assigned to its nearest neuron:

$$c = f^T(\mathbf{x}_i) = arg \min_{1 \leq r \leq m} d^T(\mathbf{x}_i, \mathbf{w}_r) \tag{8}$$

During the affectation step of IABSOM-L1, the reference vectors (prototypes) and the vectors of weights are kept fixed. The cost function J is minimized with respect to the allocation function and each individual $\mathbf{x}_i$ is assigned to its nearest neuron:

$$c = f^T(\mathbf{x}_i) = arg \min_{1 \leq r \leq m} d^T_{\boldsymbol{\Lambda}}(\mathbf{x}_i, \mathbf{w}_r) \tag{9}$$

The batch SOM algorithm based on adative and non-adaptive city-block distances for interval-valued can be summarized as follows.

Algorithm 2.1.
Batch self-organizing algorithm for interval-valued data based on city-block distances.

Step 1 *Initialization*
Fix the number m of neurons (clusters); Fix δ; Fix the kernel fuction K^T; Fix the number of iterations N_{iter}; Fix T_{min}, T_{max}; Set $T \leftarrow T_{max}$; Set $t \leftarrow 0$;
Randomly select m distinct prototypes $\boldsymbol{w}_c^{(0)} \in E\,(c = 1, \ldots, m)$; Set the map $L(m, \boldsymbol{W}^0)$, where $\boldsymbol{W}^0 = (\boldsymbol{w}_1^{(0)}, \ldots, \boldsymbol{w}_m^{(0)})$; For IABSOM-L1 algorithm, set $\lambda_{rj} = 1$ $(r = 1, \ldots, m; j = 1, \ldots, p)$
For IBSOM-L1 algorithm, assign each object $\boldsymbol{x}_i$ to the closest neuron (cluster) according to equation (8); For IABSOM-L1 algorithm, assign each object $\boldsymbol{x}_i$ to the closest neuron (cluster) according to equation (9);

Step 2 *Representation*
Set $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$;
Compute w_{rj} $(j = 1, \ldots, p)$, components of the prototypes $\boldsymbol{w}_r = (w_{r1}, \ldots, w_{rp})$ $(r = 1, \ldots, m)$, according to the algorithm given by [15, 16].

Step 3 *Weighting*
Skip this step for IBSOM-L1 algorithm;
For IABSOM-L1 algorithm, compute the components λ_{rj} of the vectors of weights $\boldsymbol{\lambda}_r$ $(r = 1, \ldots, m; j = 1, \ldots, p)$ according to equation (7);

Step 4 *Affectation*
For IBSOM-L1 algorithm, assign each object $\boldsymbol{x}_i$ to the closest neuron (cluster) according to equation (8); For IABSOM-L1, algorithm, assign each object $\boldsymbol{x}_i$ to the closest neuron (cluster) according to equation (9);

Step 5 *Stopping criterion*
If $T = T_{min}$ then STOP; otherwise set $t = t + 1$ and go to Step 2 (Representation).

3 Experimental results

To evaluate the performance of these SOM algorithms, two applications with a car models and a freshwater fish species interval-valued data set are considered. Our aim is to achieve a comparison of the batch SOM algorithms based on adaptive and non-adaptive city-block distances with the batch SOM algorithms based on adaptive (hereafter named IABSOM-L2) and non-adaptive (hereafter named IBSOM-L2) Euclidean distances [6, 11] between vectors of intervals. Then, the usefulness of these batch SOM algorithms will be illustrated with an application concerning a city temperatures interval-valued data set.

As pointed out by [1] the performance of these algorithms strongly depend on the parameters of the minimization algorithms. The most important important are T_{max}, T_{min}, N_{iter} and the cooled schedule. Table 1 gives the parameters for the batch SOM algorithms. They were fixed after several tests with these algorithms.

Interval-valued data sets	Parameters			
	N_{Iter}	Number of neurons m	T_{min}	T_{max}
Car models	30	9 (3×3 grid)	0.3	6.5
Freshwater fish species	30	6 (2×3 grid)	0.3	6.5
City temperatures	30	9 (3×3 grid)	0.3	6.5

Table 1: Parameters List

Moreover, in this paper δ is the Euclidean distance and the neighborhood kernel function is

$$K^T(\delta(c,r) = exp\left(-\frac{(\delta(c,r))^2}{2T^2}\right)$$

Intitially, each interval-valued variable on these data sets were normalized by means of a suitable dispersion measure [5]. Let $D_j = \{x_{1j}, \ldots, x_{nj}\}$ be the set of observed intervals $x_{ij} = [a_{ij}, b_{ij}]$ on variable $j\,(j = 1, \ldots, p)$. The dispersion of the jth variable is defined as $s_j^2 = \sum_{i=1}^{n} d_j(x_{ij}, g_j)$, where $g_j = [\alpha_j, \beta_j]$ is the "central" interval computed from D_j and $d_j(x_{ij}, g_j) = |a_{ij} - \alpha_j| + |b_{ij} - \beta_j|$ (if the comparison between intervals uses the city-block distance), or $d_j(x_{ij}, g_j) = (a_{ij} - \alpha_j)^2 + (b_{ij} - \beta_j)^2$ (if the comparison between intervals uses the Euclidean distance).

The central interval $g_j = (\alpha_j, \beta_j)^T$ has its bounds computed from $\sum_{i=1}^{n} d_j(x_{ij}, g_j) \rightarrow Min$. They are $\alpha_j = \frac{1}{n} a_{ij}$ and $\beta_j = \frac{1}{n} b_{ij}$. Each observed interval $x_{ij} = [a_{ij}, b_{ij}]\,(i = 1, \ldots, n)$ is normalized as $\tilde{x}_{ij} = [\tilde{a}_{ij}, \tilde{b}_{ij}]$, where $\tilde{a}_{ij} = \frac{a_{ij}}{\sqrt{(s_j^2)}}$ and $\tilde{b}_{ij} = \frac{b_{ij}}{\sqrt{(s_j^2)}}$ (if the comparison between intervals uses the Euclidean distance) or $\tilde{a}_{ij} = \frac{a_{ij}}{s_j^2}$ and $\tilde{b}_{ij} = \frac{b_{ij}}{s_j^2}$ (if the comparison between intervals uses the city-block distance). One can easily show that $\tilde{s}_j^2 = 1$, where $\tilde{\alpha}_j = \frac{1}{n}\tilde{a}_{ij}$ and $\tilde{\beta}_j = \frac{1}{n}\tilde{b}_{ij}$. Now, all the normalized interval-valued variables have the same dispersion $\tilde{s}_j^2 = 1$.

In order to compare the results given by the batch SOM algorithms applied on the interval-valued data sets considered in this paper, an external index the corrected Rand index (CR) [13] and the overall error rate of classification ($OERC$) [4].

3.1 Performance of the batch SOM algorithms

The car model data set concerns 33 car models described by 8 interval-valued variables. These car models are grouped in four *a priori* classes of unequal sizes: *Utilitarian* (size 10), *Berlina* (size 8), *Sporting* (size 7) and *Luxury* (size 8). The interval-valued variables are: *Price, Engine Capacity, Top Speed, Acceleration, Step, Length, Width* and *Height*.

The freshwater fish species concerns 12 species of freshwater fish, each specie being described by 13 symbolic interval variables. These species are grouped into four *a priori* classes of unequal sizes acoording to diet: two classes (*Carnivorous* and *Detritivorous*) of size 4 and two clusters of size 2 (*Omnivorous* and *Herbivorous*). The symbolic interval variables are: *Length*, *Weight*, *Muscle*, *Intestine*, *Stomach*, *Gills*, *Liver*, *Kidneys*, *Liver/Muscle*, *Kidneys/Muscle*, *Gills/Muscle*, *Intestine/Muscle* and *Stomach/Muscle*.

Each batch SOM algorithm was run 50 times on these datasets and the best result, according to the adequacy criterion, was selected. The cluster partitions obtained with these clustering methods were compared with the 4-class partition known a priori. Table 2 shows the results.

Car models data set			Freshwater fish species data set		
	Comparison indexes			Comparison indexes	
Clustering algorithms	CR	$OERC$	Clustering algorithms	CR	$OERC$
IBSOM-L1	0.329	0.242	IBSOM-L1	0.496	0.083
IABSOM-L1	0.477	0.152	IABSOM-L1	0.423	0.167
IBSOM-L2	0.444	0.212	IBSOM-L2	-0.013	0.417
IABSOM-L2	0.570	0.152	IABSOM-L2	0.209	0.333

Table 2: Comparison between the batch SOM algorithms on the car models and freshwater fish species interval-valued data sets

For the car models data set, the batch SOM algorithms with adaptive distances outperformed the batch SOM with non-adaptive distances. Moreover, the batch SOM algorithms with Euclidean distances outperformed the batch SOM algorithms with city-block distances. Finally, the batch SOM algorithms with adaptive Euclidean distances had the best performance whereas the worst performance was presented by the batch SOM algorithm based on non-adaptive city-block distances.

For the freshwater fish species data set, the batch SOM algorithms with city-block distances outperformed the batch SOM with Euclidean distances. Moreover, the batch SOM algorithms with non-adaptive city-block distances had the best performance and the batch SOM algorithms with non-adaptive Euclidean distances had the worst performance.

3.2 Robustness of the batch SOM algorithms

Theoretical studies indicate that city-block based models are more robust than those based on Euclidean distances. In order to evaluate the robustness of these batch SOM algorithms, we introduces 4 outliers on the car models and 1 outlier on the freshwater fish species data sets in the following way: the boundaries of the interval-valued variables describing, repectively, 1 individual of the the freshwater fish species and 4 individuals of the car models (one individual by each a priori class) were multiplied by 10 (configuration 1), by 100 (configuration 2) and by 1000 (configuration 3).

The batch SOM algorithms have been applied on these modified data sets. The cluster partitions obtained with these clustering methods were compared with the 3-class partition known a priori. Again, the comparison indexes used were the CR and ERC. These indexes were also calculated for the best result. Table 3 shows the results.

It can be observed that all the batch SOM algorithms were affected by the introduction of outiliers. However, those based on Euclidean distance were more affected than those based on city-block distances.

Car models interval-valued data set					Freshwater fish species interval-valued data set				
Conf.	Clustering algorithms				Conf.	Clustering algorithms			
	IBSOM-L1		IABSOM-L1			IBSOM-L1		IABSOM-L1	
	CR	$OERC$	CR	$OERC$		CR	$OERC$	CR	$OERC$
1	0.297	0.394	0.423	0.364	1	0.304	0.167	0.163	0.333
2	0.320	0.424	-0.007	0.667	2	0.160	0.250	0.119	0.500
3	0.320	0.424	-0.007	0.667	3	0.186	0.333	0.390	0.167
Conf.	Clustering algorithms				Conf.	Clustering algorithms			
	IBSOM-L2		IABSOM-L2			IBSOM-L2		IABSOM-L2	
	CR	$OERC$	CR	$OERC$		CR	$OERC$	CR	$OERC$
1	-0.013	0.667	-0.009	0.667	1	0.166	0.333	0.233	0.333
2	-0.013	0.697	-0.009	0.667	2	0.007	0.583	0.007	0.583
3	-0.013	0.697	-0.009	0.667	3	0.007	0.583	0.007	0.583

Table 3: Comparison between the batch SOM algorithms

3.3 Application: city temperatures interval-valued data set

City temperature interval-valued data set [10] gives the minimum and the maximum monthly temperatures of cities in degrees centigrade. This data set consists of a set of 37 cities described by 12 interval-valued variables. Table 4 shows part of this interval-valued data set. In this application, the 12 interval-valued variables have been considered for clustering purposes.

	January	February	...	November	December
Amsterdam	$[-4, 4]$	$[-5, 3]$	...	$[1, 10]$	[-1, 4]
...	...	...	...	...	...
Mauritius	$[22, 28]$	$[22, 29]$	...	$[19, 27]$	$[21, 28]$
...	...	...	...	...	...
Zurich	$[-11, 9]$	$[-8, 15]$	...	$[0, 19]$	$[-11, 8]$

Table 4: City temperature interval-valued data set with 12 interval variables

The ABSOM-L1 batch SOM algorithm was run 50 times on the city temperature interval-valued data set and the best result, according to the adequacy criterion, was selected. Figure 1 gives the self-organizing map of this interval-valued data set.

Table 5 gives the description of the cluster prototypes according to the minimum and the maximum monthly temperatures.

In Figure 1, the cities in bold are the most similar to the prototype of the cluster that they belong. It can be observed that the grid is coherent with the latitude location and temperature ranges of the cities. Latitude location grows from left to right and from bottom to top on the grid. Temperature ranges grows from right to left and from top to bottom on the grid.

Table 6 gives the relevance weights of the interval-valued variabels into the clusters.

It can be observed, for example, that in cluster 2 (Frankfurt and Zurich), the variable

Athens, Lisbon **Rome**, San Francisco Tehran, Tokyo ($35:41^o$ N–$41:54^o$ N)	Frankfurt, **Zurich** ($47:22^o$ N–$50:06^o$ N)	Amsterdam, Copenhagen Geneva, Moscow **Munich**, Stockholm Toronto, Vienna ($43:42^o$ N–$59:19^o$ N)
Bahrain, Cairo Dubai, Hong Kong NewDelhi ($22:16^o$ N–$30:29^o$ N)	**Madrid**, Seoul ($37:33^o$ N–$40:23^o$ N)	London, New York **Paris** ($40:39^o$ N–$51:30^o$ N)
Bombay, Calcutta, **Colombo**, Kuala Lumpur Madras, Manila Singapore ($01:17^o$ N–$22:56^o$ N)	**Mauritius** ($20:10^o$ S)	Mexico City, **Nairobi** Sydney ($33:51^o$ S–$19:26^o$ N)

Figure 1: Final grid of the city temperature interval-valued data set

Var.	Cluster prototypes								
	1	2	3	4	5	6	7	8	9
JAN	6.2:11.6	-9.8:8.9	-3.5:0.8	12.5:19.7	0.8:8.9	0.8:6.2	20.6:29.6	21.5:27.8	11.6:25.1
FEB	5.7:12.5	-7.7:15.4	-4.8:2.8	11.5:22.1	0.9:11.5	0.9:6.7	22.1:29.8	22.1:28.9	15.4:26.0
MAR	7.5:16.1	-4.3:17.2	-1.0:7.5	15.0:24.7	3.2:16.1	2.1:9.6	22.5:31.1	21.5:29.0	17.2:24.7
APR	11.2:17.5	0:21.3	2.5:12.5	18.8:28.8	6.2:18.8	5.0:15.0	23.8:32.6	21.3:27.5	16.3:23.8
MAY	13.3:23.6	2.9:26.6	7.3:17.7	22.1:32.5	11.8:23.6	7.3:19.2	25.1:32.5	19.2:25.1	13.3:22.1
JUN	17.9:27.7	6.54:29.4	11.4:21.2	24.5:34.3	16.3:29.4	11.4:21.2	24.5:32.7	17.9:24.5	11.4:21.2
JUL	22.6:31.3	10.4:31.3	13.9:24.4	26.1:36.6	17.4:31.3	13.9:24.4	24.4:29.6	17.4:22.6	10.4:20.9
AUG	22.3:30.9	8.5:25.7	13.7:22.3	25.7:34.3	15.4:29.1	13.7:20.6	25.7:29.1	17.1:22.3	10.3:20.6
SEP	18.8:26.6	4.7:23.5	10.9:18.8	25.0:34.4	12.5:28.2	10.9:20.3	25.0:29.7	17.2:23.5	10.9:23.5
OCT	14.5:21.2	2.6:22.5	6.6:13.2	21.2:30.4	7.9:23.8	7.9:15.9	23.8:30.4	18.5:25.1	13.2:23.8
NOV	8.7:17.5	0:14.2	1.0:6.5	17.5:26.2	6.5:18.5	5.4:9.8	22.9:29.5	18.5:27.3	14.2:25.1
DEC	6.4:13.7	-8.2:8.2	-1.8:2.7	13.7:21.0	0.9:9.1	0.9:6.4	22.0:30.2	21.0:28.4	12.8:22.9

Table 5: Cluster prototypes: minimum and the maximum monthly temperatures

"temperature of June" has the greatest relevance weight because in these cities the temperature in this month ranges very similarly, respectively, $[3:27]$ and $[6:30]$. On contrary, the variable "temperature of November" has the smallest relevance weight in this cluster because in these cities the temperature in this month ranges more diferently, respectively, $[-3:14]$ and $[0:19]$.

4 Conclusion

The main contributions of this paper are the introduction of batch SOM algorithms based on adaptive and non-adaptive city-block distances, suitable for objects described by interval-valued variables, that, for a fixed epoch, optimizes a cost function. These SOM algorithms combine

Var.	Clusters								
	1	2	3	4	5	6	7	8	9
JAN	0.62	1.38	0.48	0.76	1.01	0.75	0.50	0.66	0.40
FEB	0.92	1.39	0.54	0.75	0.60	0.70	0.59	0.79	0.67
MAR	1.53	0.65	0.51	0.77	0.42	1.23	0.91	0.92	2.01
APR	2.06	0.77	0.87	0.86	1.43	1.83	1.21	1.05	1.52
MAY	1.24	2.25	1.22	0.95	1.11	1.03	0.81	1.10	1.13
JUN	0.87	**2.58**	1.75	1.23	0.91	0.80	1.24	1.16	0.71
JUL	0.64	1.35	1.86	1.51	1.33	0.74	1.39	1.19	0.83
AUG	0.62	0.73	2.00	1.36	2.02	0.84	1.37	1.21	0.95
SEP	0.96	0.96	1.61	1.51	1.44	0.99	1.42	1.23	1.20
OCT	1.76	1.05	1.42	1.14	0.58	0.91	2.00	1.13	2.16
NOV	1.45	**0.37**	0.74	0.86	0.53	1.97	0.99	1.05	1.50
DEC	0.49	0.46	0.62	0.71	2.19	0.89	0.54	0.71	0.47

Table 6: City temperature data set: relevance weights of the interval-valued variabels into the clusters

the best visualization and clustering characteristics provided by SOM neural networks with the flexibility offered by adaptive distances in the recognition of classes with different shapes and sizes and the robustness of the city-block distances.

The performance of these batch SOM algorithms based on adaptive and non-adaptive city-block distances were evaluated in comparison with batch SOM algorithms based on adaptive and non-adaptive Euclidean distances on car models and freshwater fish species interval-valued data sets. The accuracy of the results furnished by these algorithms was assessed by the corrected Rand index (CR) and the overall error rate of classification ($OECR$).

Overall, the batch SOM algorithms with adaptive (city-block and Euclidean) distances outperformed the batch SOM algorithms with non-adaptive distances on the original car models and freshwater fish species data sets. Moreover, all the batch SOM algorithms were affected by the introduction of outiliers. However, those based on Euclidean distance were more affected than those based on city-block distances.

Finally, the application of the batch SOM algorithm based on city-block distances on the city temperatures interval-valued data sets illustrated the usefulness of the presented SOM algorithms.

Acknowledgement

The authors would like to thanks CNPq and FACEPE (Brazilian agencies) for their finacial support.

References

[1] Badran, F., Yacoub, M. and Thiria, S. (2005) *Self-organizing maps and unsupervised classification.* in Neural Networks: methodology and applications, G. Dreyfus (Ed.), Springer, 379–442.

[2] Bock, H.-H. (2002) *Clustering algorithms and Kohonen maps for symbolic data.* Journal of the Japanese Society of Computational Statistics, **15**, 1–13.

[3] Bock, H-.H. and Diday, E. (2000) *Analysis of Symbolic Data, Exploratory methods for extracting statistical information from complex data.* Springer.

[4] Breiman, L., Friedman, J., Stone, C.J. and Olshen, R.A. (1984) Classification and Regression Trees, Chapman and Hall/CRC, Boca Raton, 1984

[5] Chavent, M. and Saracco, J. (2008) *On central tendency and dispersion measures for intervals and hypercubes.* Communications in Statistics Theory and Methods, **37**, 1471–1482.

[6] De Carvalho, F.A.T and Pacifico, L.D.S. (2011) *Une version batch de l'algorithme SOM pour des données de type intervalle.* Actes des XVIIIème Rencontres de la Société Francophone de Classification (SFC- 2011), 99–102.

[7] Diday, E. et Govaert, G. (1977). *Classification Automatique avec Distances Adaptatives.* R.A.I.R.O. Informatique Computer Science, **11**, 329–349.

[8] Diday, E. and Noirhomme-Fraiture, M. (2008) *Symbolic Data Analysis and the Sodas Software.* Wiley.

[9] D'Urso, P. and De giovanni, L. (2011) *Midpoint radius self-organizing maps for interval-valued data with telecommunications application.* Applied Soft Computing, **11**, 3877–3886

[10] Guru, D.S., Kiranagi, B.B. and Nagabhushan, P. (2004) *Multivalued type proximity measure and concept of mutual similarity value useful for clustering symbolic patterns.* Pattern Recognition Letters, **25**, 1203–1213

[11] Hajjar, C. and Hamdan, H. (2011)*Self-organizing map based on L2 distance for interval-valued data.* Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI-2011), 317–322.

[12] Hajjar, C. and Hamdan, H. (2011)*Self-organizing map based on city-block distance for interval-valued data.* in Complex sytems desing & management. Proceedings of the second international conference oncomplex systems & design (CSDM2-2011), Hammami, O., Krob, D. and Voirin, J-L (Eds.), Springer, 181–292.

[13] Hubert, L. and Arabie, P. (1985) *Comparing partitions.* Journal of Classification, **2**, 193–218.

[14] Jain, A. K., Murty, M. N. and Flynn, P. J. (1999) *Data clustering: a review.* ACM Comput. Surveys, **31**, 264–233.

[15] Jajuga, K. (1991) *L1-norm based fuzzy clustering.* Fuzzy Sets and Systems, **39**, 43–50.

[16] Karst, O. J. (1958) *Linear curve fitting using least deviations.* Journal of the American Association,]textbf53, 118–132.

[17] Kohonen, T. (1995) *Self-Organisation Maps.* Springer.

[18] Xu, R. and Wunsch, D.I.I. (2005) *Survey of clustering algorithms.* IEEE Trans. Neural Networks, **16**, 645–678.

Appendix

A Proof of Theorem 2.1

Recall that the important issue is here the minimisation of $\sum_{i=1}^{n} |y_i - az_i|$, and that we denoted $u_i = \dfrac{y_i}{z_i}$ for each $i = 1, \ldots, n$. Then it is equivalent to minimise $\sum_{i=1}^{n} z_i | \, u_i - a \, |$ with respect to the value of a. Recall also that by definition we have $z_i \in \,]0, 1[$. Thus, given an arbitrary sequence of n reals, denoted $z = (z_1, z_2, \ldots, z_n)$, we may associate with any sequence $u = (u_1, u_2, \ldots, u_n)$ of n reals a function ϕ by:

$$\phi(x, u) \quad = \quad \sum_{i=1}^{n} z_i \mid u_i - x \mid ,$$

where we will suppose that $z_i \in \,]0, 1[$ for each $i \in \{1, \ldots, n\}$. With these notation and hypothesis, and assuming that the terms of u are increasingly ordered, *i.e.*:

$$u_1 \leq u_2 \leq \ldots \leq u_n,$$

it was proved (see [15, 16]) that the set $\operatorname{argmin}_{x \in \mathbb{R}} \phi(x \, ; u)$ is an interval of the form $[u_i, u_j]$ where $j \in \{i, i+1\}$, for some $i \in \{1, \ldots, n-1\}$. In other words, there exist $i \in \{1, \ldots, n-1\}$ such that $\operatorname{argmin}_{x \in \mathbb{R}} \phi(x \, ; u)$ is an interval that is either reduced to $\{u_i\}$ or equal to $[u_i, u_{i+1}]$.

Notation A.1 In order to simplify our notations, we will write $opt\,(u)$ in place of $\operatorname{argmin}_{x \in \mathbb{R}} \phi(x \, ; u)$. Moreover, given an increasing sequence u of n reals, $opt\,(u)$ is then of the form $[u_i, u_j]$ where $j \in \{i, i+1\}$, for some $i \in \{1, \ldots, n-1\}$. Based on these notations, $opt^{\star}(u)$ will designate the real defined as $opt^{\star}(u) = \dfrac{u_i + u_j}{2}$ (see the the algorithm presented hereabove before Theorem 2.1).

Notation A.2 Let u and v be any two finite sequences of n reals, and α and β be any two reals. The notation $\alpha \, u + \beta \, v$ will denote the sequence defined by:

$$\alpha \, u + \beta \, v = (\alpha \, u_1 + \beta \, v_1, \ldots, \alpha \, u_i + \beta \, v_i, \ldots, \alpha \, u_n + \beta \, v_n).$$

In addition, given an integer $n > 0$, we define the sequence e^k of n reals as follows:

$$\forall i \in \{1, \ldots, n\}, \quad e_i^k = \begin{cases} 1, \text{ if } i = k, \\ 0, \text{ otherwise.} \end{cases}$$

In the rest of this section, we aim to compare the sets $opt\,(u)$ and $opt\,(v)$ when u and v satisfy $u_i \leq v_i$ for all $i \in \{1, \ldots, n\}$. We begin with the next lemma.

Lemma A.3 *Consider a sequence u of n reals together with $z_1, z_2, \ldots, z_n \in]0,1[$. Given any real $h > 0$ and any integer $k \in \{1, \ldots, n\}$, denote $v = u + h\,e^k$.*
If $(a,b) \in [opt\,(u) \times opt\,(v)] \setminus [opt\,(v) \times opt\,(u)]$, then $a \leq b$.

Proof. Let $(a,b) \in [opt\,(u) \times opt\,(v)] \setminus [opt\,(v) \times opt\,(u)]$, where we have denoted $v = u + h\,e^k$. Then it results:

$$\left\{ \begin{array}{l} \phi(b,v) - \phi(a,v) \leq 0 \ \text{ and } \ \phi(b,u) - \phi(a,u) \geq 0, \\ \text{one at least of these two inequalities being strict.} \end{array} \right. \tag{10}$$

Since $u_k < v_k = u_k + h$, all the possible cases, w.r.t. to a, are as follows:

$$a \leq u_k, \ \ u_k < a \leq v_k \text{ and } v_k < a.$$

Suppose that $b < a$.

Case 1: $a \leq u_k$. Therefore $b < a \leq u_k < v_k$. For each $x \in \mathbb{R}$, we have:

$$\begin{array}{rcl} \phi(x,v) & = & \phi(x,u) - z_k(u_k - x) + z_k(v_k - x) \\ \phi(x,v) & = & z_k(v_k - u_k) + \phi(x,u) \end{array}$$

Thus:

$$\begin{array}{rcl} \phi(a,v) & = & z_k(v_k - u_k) + \phi(a,u), \\ \phi(b,v) & = & z_k(v_k - u_k) + \phi(b,u). \end{array}$$

This leads to:

$$\phi(b,v) - \phi(a,v) \ = \ \phi(b,u) - \phi(a,u) \tag{11}$$

Observe that (11) contradicts the fact that, by (10), either $\phi(b,v) - \phi(a,v) \leq 0$ or $\phi(b,u) - \phi(a,u) \geq 0$ is a strict inequality.

Case 2: $u_k < a \leq v_k$.

Subcase 2-1: $b \leq u_k$. Then $b \leq u_k < a \leq v_k$. The following equalities are straightforward. The first of them is proved as in Case 1:

$$\begin{array}{rcl} \phi(b,v) & = & z_k(v_k - u_k) + \phi(b,u), \\ \phi(a,v) & = & z_k(v_k - a) - z_k(a - u_k) + \phi(a,u), \\ \phi(a,v) & = & z_k(v_k + u_k - 2a) + \phi(a,u). \end{array}$$

Then $\phi(b,v) - \phi(a,v) = 2z_k(a - u_k) + \phi(b,u) - \phi(a,u) > \phi(b,u) - \phi(a,u) \geq 0$. Thus $\phi(b,v) - \phi(a,v) > 0$, which contradicts the definition of b.

Subcase 2-2: $u_k < b < a$. Then $u_k < b < a \leq v_k$. The following equalities are easily checked: the third of them is proved as in Subcase 2-1:

$$\begin{aligned}\phi(b,v) &= z_k(v_k - b) - z_k(b - u_k) + \phi(b,u),\\ \phi(b,v) &= z_k(v_k + u_k - 2b) + \phi(b,u),\\ \phi(a,v) &= z_k(v_k + u_k - 2a) + \phi(a,u).\end{aligned}$$

Thus $\phi(b,v) - \phi(a,v) = 2z_k(a-b) + \phi(b,u) - \phi(a,u) > \phi(b,u) - \phi(a,u) \geq 0$. We deduce that $\phi(b,v) - \phi(a,v) > 0$, which contradicts again the definition of b.

Case 3: $v_k < a$.

Subcase 3-1: $b \leq u_k$. Then $b \leq u_k < v_k < a$. The following equalities hold clearly, the first of them being proved as in Case 1:

$$\begin{aligned}\phi(b,v) &= z_k(v_k - u_k) + \phi(b,u),\\ \phi(a,v) &= z_k(a - v_k - (a - u_k)) + \phi(a,u),\\ \phi(a,v) &= z_k(u_k - v_k) + \phi(a,u).\end{aligned}$$

Then $\phi(b,v) - \phi(a,v) = 2z_k(v_k - u_k) + \phi(b,u) - \phi(a,u) > \phi(b,u) - \phi(a,u) \geq 0$. Since $v_k > u_k$, it results that $\phi(b,v) - \phi(a,v) > 0$, which again contradicts the definition of b.

Subcase 3-2: $u_k < b < v_k$. Then $u_k < b < v_k < a$. The following equalities hold, the third of them being proved as in Subcase 3-1:

$$\begin{aligned}\phi(b,v) &= z_k(v_k - b) - z_k(b - u_k) + \phi(b,u),\\ \phi(b,v) &= z_k(v_k + u_k - 2b) + \phi(b,u),\\ \phi(a,v) &= z_k(u_k - v_k) + \phi(a,u).\end{aligned}$$

Thus $\phi(b,v) - \phi(a,v) = 2z_k(v_k - b) + \phi(b,u) - \phi(a,u)$. Since $v_k > b$, we deduce that $\phi(b,v) - \phi(a,v) > 0$, which contradicts again the definition of b.

Subcase 3-3: $v_k \leq b < a$. Then $u_k < v_k \leq b < a$. The following equalities hold, the third of them being established as in Subcase 3-2:

$$\begin{aligned}\phi(b,v) &= z_k(b - v_k) - z_k(b - u_k) + \phi(b,u),\\ \phi(b,v) &= z_k(u_k - v_k) + \phi(b,u),\\ \phi(a,v) &= z_k(u_k - v_k) + \phi(a,u).\end{aligned}$$

Thus $\phi(b,v) - \phi(a,v) = \phi(b,u) - \phi(a,u)$. This contradicts the fact that, by (10), at least one of the inequalities $\phi(b,v) - \phi(a,v) \leq 0$ and $\phi(b,u) - \phi(a,u) \geq 0$ is strict.

We conclude that $b < a$ leads to a contradiction in any case, so that $a \leq b$ holds, as required.

Proposition A.4 *Let u be any sequence of n real numbers, $h > 0$, $k \in \{1,\ldots,n\}$ and $\{z_1, z_2, \ldots, z_n\} \subseteq]0,1[$. Then $opt^\star(u) \leq opt^\star(u + h\,e^k)$.*

Proof. Let $v = u + h\,e^k$. Since $opt\,(u)$ and $opt\,(v)$ are intervals that may be reduced to singletons, we denote $opt\,(u) = [a_1, a_2]$ and $opt\,(v) = [b_1, b_2]$ with $a_1 \leq a_2$ and $b_1 \leq b_2$. If $b_1 < a_1$, then $(a_1, b_1) \in opt\,(u) \times opt\,(v)$ with $b_1 \in opt\,(v) \setminus opt\,(u)$, which is contradictory by Lemma 1. Therefore $a_1 \leq b_1$. Similarly, if $b_2 < a_2$, then $(a_2, b_2) \in opt\,(u) \times opt\,(v)$ and $a_2 \in opt\,(u) \setminus opt\,(v)$, which is again contradictory by Lemma 1. Therefore $a_2 \leq b_2$. It results that

$$opt^\star(u) = \frac{a_1 + b_1}{2} \leq \frac{a_2 + b_2}{2} = opt^\star(v),$$

which proves that the result of the proposition holds true.

Then, Theorem 2.1 is an immediate consequence of the next Proposition A.5.

Proposition A.5 *Let u and v be two sequence of n real numbers such that $u \leq v$,* i.e. *$u_i \leq v_i$ for each $i \in \{1, \ldots, n\}$. Then $opt^\star(u) \leq opt^\star(v)$.*

Proof. Let $\mathcal{K} = \{k \in \{1, \ldots, n\} \,:\, u_k < v_k\}$ and $L = |\,\mathcal{K}\,|$. Denote $h_k = v_k - u_k$ for each $k \in \mathcal{K}$, and denote by σ an arbitary bijection from $\{1, \ldots, L\}$ onto set $\mathcal{K}$. It results:

$$v = u + \sum_{l=1}^{L} h_{\sigma(l)} e^{\sigma(l)}. \tag{12}$$

For each $l \in \{0, 1, \ldots, L\}$, define the sequence $w^{(l)}$ as follows:

$$w^{(l)} = \begin{cases} u, \text{ if } l = 0, \\ u + \displaystyle\sum_{j=1}^{l} h_{\sigma(j)} e^{\sigma(j)}, \text{ otherwise.} \end{cases}$$

Notice that $w^{(L)} = v$ according to (12). Moreover, for each $l \in \{0, 1, \ldots, L-1\}$, we have:

$$w^{(l+1)} = w^{(l)} + h_{\sigma(l+1)} e^{\sigma(l+1)}.$$

Since for each $k \in \mathcal{K}$, we have $h_k > 0$, it results from Proposition A.4 that:

$$\forall l \in \{0, 1, \ldots, L-1\}, \quad opt^\star(w^{(l)}) \leq opt^\star(w^{(l+1)}).$$

As a consequence, we obtain:

$$opt^\star(w^{(0)}) \leq opt^\star(w^{(1)}) \leq \ldots \leq opt^\star(w^{(L)}),$$

and thus $opt^\star(u) = opt^\star(w^{(0)}) \leq opt^\star(w^{(L)}) = opt^\star(v)$, as required.