A Batch Self-Organizing Maps Algorithm Based on Adaptive Distances

Luciano D. S. Pacifico and Francisco de A. T. de Carvalho

***Abstract*— Clustering methods aims to organize 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 self-organizing map (SOM) introduced by Kohonen is an unsupervised competitive learning neural network method which has both clustering and visualization properties, using a neighborhood lateral interaction function to discover the topological structure hidden in the data set. In this paper, we introduce a batch self-organizing maps algorithm based on adaptive distances. Experimental results obtained in real benchmark datasets show the effectiveness of our approach in comparison with traditional batch self-organizing map algorithms.**

I. Introduction

CLUSTERING is a popular task in knowledge discovering, and it is applied in various fields including data mining, pattern recognition, computer vision, etc. Clustering methods aims to organize 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 [1]-[3].

Hierarchical methods yield complete hierarchy, i.e., a nested sequence of partitions of the input data. Hierarchical methods can be agglomerative or divisive. Agglomerative methods yield a sequence of nested partitions, starting with trivial clustering in which each item is in a unique cluster and ending with trivial clustering in which all items are in the same cluster. A divisive method starts with all items in a single cluster and performs a splitting procedure until a stopping criterion is met (usually upon obtaining a partition of singleton clusters).

Partitioning a set of items into a predefined number of clusters is an important topic in data analysis, pattern recognition, and image processing [4]-[5]. Partition methods seek to obtain a single partition of the input data into a fixed number of clusters. Such methods often look for a partition that optimizes (locally) an adequacy criterion function. To improve cluster quality, the algorithm is run multiple times at different starting points, and the best configuration obtained from the total runs is used as the output clustering. K-means algorithm [6] and Fuzzy C-means [7] are the most famous partitional approaches.

The Kohonen Self Organizing Map (SOM) [10] 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, $\mathbb{R}^p$, to lattice space which usually has a lower dimension, generally 2 and is called a map. This projection enables a partition of the inputs into "similar" clusters while preserving their topology.

Variations of basic SOM algorithm have been done aiming its improvement. In Kangas *et al.* [11] two novel SOM-based approaches were presented: one based on dynamic weighting of the input signals at each input of each neuron, which improved the ordering when very different input signals are used; a definition of neighborhoods in the learning algorithm by the minimal spanning tree, which provided a better and faster approximation of prominently structured density functions. In Badran *et al.* [12] a batch self-organizing approach similar to the K-means algorithm was presented, consisting in a two-step algorithm with an affectation step where all instances are affected to the closest neuron from the grid, and a representation step, where all neurons are updated.

In this paper we extend the batch self-organizing map presented in [12] in order to obtain an adaptive batch self-organizing algorithm, which uses adaptive Euclidean distances parameterized by vectors of weights on the variables that change at each algorithm's iteration and changes from one cluster to another. The computation of these vectors of weights in this algorithm is inspired from the approach used to compute a weight for each variable in each cluster in the dynamic clustering algorithm based on adaptive distances [9]. The traditional dynamic clustering algorithms [8] are iterative two-step relocation algorithms involving the construction of clusters at each iteration and the identification of a suitable representative or prototype (means, factorial axes, probability laws, groups of elements, etc.) for each cluster by locally optimizing an adequacy criterion between clusters and their corresponding prototypes. The adaptive dynamic clustering algorithm [9] also optimizes a criterion based on a fitting measure between clusters and their prototypes, but the distances used to compare clusters and their prototypes change at each iteration. These distances (represented by variable weight vectors) are not determined absolutely and are different from one cluster to another. The advantage of these adaptive distances is that the clustering algorithm is

Luciano D. S. Pacifico and F. A. T. de Carvalho are with Center of Informatics, Federal University of Pernambuco; Recife; PE; Brazil; 50740-560 (e-mails:{ldsp,fatc}@cin.ufpe.br).This work was supported by FACEPE and CNPq (Brazilian Research Agencies).

able to recognize clusters of different shapes and sizes.

To evaluate the effectiveness of our approach, experiments using real benchmark classification problems obtained from the UCI Machine Learning Repository [14] are done.

This paper is organized as follows. Section II presents the standard self-organizing map [10]. Standard batch self-organizing map is presented int Section III. Next, our novel approach is presented (Section IV) and the experimental results are shown (Section V). Conclusions are given in Section VI.

II. Self-Organizing Map (SOM)

Kohonen's SOM [10] is used nowadays through numerous domains and has been successfully applied in numerous applications. It is a very popular tool used for visualizing high dimensional data spaces. SOM can be considered as doing vector quantization and/or clustering while preserving the spatial ordering of the input data reflected by implementing an ordering of the prototype vectors (also called) codebook vectors, cluster centroids or referent vectors) in a one or two dimensional output space. The SOM consists of neurons organized on a regular low-dimensional grid, called the map. More formally, the map is described by a graph (C, Γ). C is a set of m interconnected neurons having a discrete topology defined by Γ. For each pair of neurons (c, r) on the map, $\delta(c, r)$ is defined as the distance function between c and r on the graph. This distance imposes a neighborhood relation between neurons. Each neuron c is represented by a p-dimensional prototype $\mathbf{w}_c = (w_c^1, ..., w_c^p)$, where p is equal to the dimension of the input vectors.

The SOM takes as its input a set of labeled sample vectors and gives as output an array of neurons with the input vectors labels attached to these neurons. Let n be the number of sample vectors $\mathbf{x}_i \in \mathbb{R}^p$ and $\mathbf{x}_i \in E$ (input data set), $i = 1, 2, \ldots, n$, where each sample vector $\mathbf{x}_i$ is identified by a label. SOM basic algorithm performs the following steps:

1) Fix m, $\eta(0)$ (start learning rate), N_{Iter} (maximum number of iterations) and $h_{\delta(j,l)}(0)$ (start neighborhood function, where $\delta(j, l)$ is a fixed distance function between neurons j and l);
2) Initialization: choose randomly m distinct vectors as the initial prototypes $\mathbf{w}_c(0)$, $c = 1, 2, ..., m$ and set the counter t = 1;
3) Sampling: Grab a random input vector $\mathbf{x}_i(t)$;
4) Selection: find the best Kohonen neuron (winner) $\mathbf{w}_c(t)$ in relation to minimum Euclidian distance rule:

$$f(\mathbf{x}_i(t))^T = \min_{1 \le j \le m} \sum_{k=1}^{p} (x_{ik}(t) - w_{jk}(t))^2;$$

5) Weight modification: for all neurons j within a specified neighborhood radius of the winning neuron $\mathbf{w}_c$, adjust the weights according to the following formula:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) h_{\delta(\mathbf{w}_c, w_j)}(t)(\mathbf{x}_i(t) - \mathbf{w}_j(t));$$

6) Updating: update both learning rate $\eta(t)$ and neighborhood function $h_{l,j}(t)$;
7) Stopping criterion: if $t = N_{Iter}$, stop; otherwise, go to 3.

III. A Batch Self-Organizing Map

This section presents the batch self-organizing map introduced by [12]. Let $E = \{1, \ldots, n\}$ the set of objects, where each object $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})\,(i = 1, \ldots, n)$ belongs to $\mathbb{R}^p$. Each neuron on the map is represented by a prototype $\mathbf{w}_c = (w_{c1}, \ldots, w_{cp})\,(c = 1, \ldots, m)$ that also belongs to $\mathbb{R}^p$.

The batch SOM training algorithm [12] is an iterative two-step algorithm (affectation and representation steps, discussed later) in which the whole data set (noted E) is presented to the map before any adjustments are made. The batch SOM algorithm [12] minimizes the following cost function:

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

where f is the allocation function and $f(\mathbf{x}_i)$ stands for the neuron of the map that is associated to the object $\mathbf{x}_i$, and $\delta(f(\mathbf{x}_i), r))$ 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 temeperature), is the neighborhood kernel function that defines influence region around each neuron r.

This cost function is an extension of the *k-means* cost function, where the Euclidean distance is replaced by

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

where

$$d^2(\mathbf{x}_i, \mathbf{w}_r) = \sum_{j=1}^{p} (x_{ij} - w_{rj})^2 \quad (3)$$

is the Euclidean distance. This generalized distance is a weghted sum of the euclidean distances between $\mathbf{x}_i$ and all the reference vectors of the neighborhood of the neuron $f(\mathbf{x}_i)$. It takes into account all the neurons of the map.

When T is kept fixed, the minimization of J is performed iteratively in two steps: affectation and representation.

During the affectation step, 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:

$$c = f^T(\mathbf{x}_i) = arg \min_{1 \le r \le m} d^T(\mathbf{x}_i, \mathbf{w}_r) \quad (4)$$

During the representation step, the allocation function is kept fixed. The cost function J is minimized with respect to the the prototypes. The prototype $\mathbf{w}_c$ is updated for each neuron according to:

$$\mathbf{w}_c = \frac{\sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), c)) \mathbf{x}_i}{\sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), c))}. \quad (5)$$

A. The Algorithm

The batch SOM algorithm for a usual cooling schedule can be summarized as follows:

1) Initialization
   Fix the number m of neurosn (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 $\mathbf{w}_c^{(0)} \in E\,(c = 1, \ldots, m)$;
   Set the map $L(m, \mathbf{W}^0)$, where $\mathbf{W}^0 = (\mathbf{w}_1^{(0)}, \ldots, \mathbf{w}_m^{(0)})$
   Assign each object $\mathbf{x}_i$ to the closest neuron (cluster) according to equation (4).
2) *Step 1: Representation.*
   Set $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$
   The allocation function is kept fixed.
   Compute the prototypes $\mathbf{w}_c^{(t)}\,(c = 1, \ldots, m)$ according to equation (5)
3) *Step 2: Affectation.*
   The prototypes $\mathbf{w}_c^{(t)}\,(c = 1, \ldots, m)$ are fixed. Assign each individual $\mathbf{x}_i\,(i = 1, \ldots, n$ to its nearest neuron according to equation (4)
4) *Stopping criterion.*
   If $T = T_{min}$ then STOP; otherwise set $t = t + 1$ and go to 2 (Step 1).

IV. Adaptive Batch Self-Organizing Map

A previous paper [11] gives an adaptive version of stochastic SOM algorithm. In this section we give an adaptive batch SOM algorithm that uses adaptive Euclidean distances parameterized by vectors of weights on the variables that change at each algorithm's iteration and changes from one cluster to another. The computation of these vectors of weights in this algorithm is inspired from the approach used to compute a weight for each variable in each cluster in the dynamic clustering algorithm based on adaptive distances [9].

The adaptive batch SOM training algorithm introduced in this paper is an iterative three-step algorithm (affectation, representation and weighting steps) in which the whole data set (noted E) is presented to the map before any adjustments are made.

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

$$d_{\mathbf{\Lambda}}^T(\mathbf{x}_i, \mathbf{w}_{f(\mathbf{x}_i)}) = \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r)) d_{\boldsymbol{\lambda}_r}^2(\mathbf{x}_i, \mathbf{w}_r) \quad (6)$$

where

$$d_{\boldsymbol{\lambda}_r}^2(\mathbf{x}_i, \mathbf{w}_r) = \sum_{j=1}^{p} \lambda_{rj}(x_{ij} - w_{rj})^2 \quad (7)$$

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

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.

The cost function of the adaptive batch SOM algorithm is given by:

$$J = \sum_{i=1}^{n} \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r)) d_{\boldsymbol{\lambda}_r}^2(\mathbf{x}_i, \mathbf{w}_r) \quad (8)$$

When T is kept fixed, the minimization of J is performed iteratively in three steps: affectation, representation and weighting.

During the affectation step, 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:

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

During the representation step, the allocation function and the vectors of weights are kept fixed. The cost function J is minimized with respect to the prototypes. The prototype $\mathbf{w}_c$ is updated for each neuron according equation (5).

During the weighting step, 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 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:

$$\lambda_{rj} = \frac{\{\prod_{h=1}^{p} \sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), h))(x_{ij} - w_{hj})^2\}^{\frac{1}{p}}}{\sum_{i=1}^{n} K^T(\delta(f^T(x_i), r))(x_{ij} - w_{rj})^2} \quad (10)$$

A. The Algorithm

The adaptive batch SOM algorithm for a usual cooling schedule can be summarized as follows:

1) Initialization
   Fix the number m of neurosn (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 $\mathbf{w}_c^{(0)} \in E\,(c = 1, \ldots, m)$;
   Set the map $L(m, \mathbf{W}^0)$, where $\mathbf{W}^0 = (\mathbf{w}_1^{(0)}, \ldots, \mathbf{w}_m^{(0)})$
   Assign each object $\mathbf{x}_i$ to the closest neuron (cluster) according to equation (4).
2) *Step 1: Representation.*
   Set $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$
   The allocation function is kept fixed.
   Compute the prototypes $\mathbf{w}_c^{(t)}\,(c = 1, \ldots, m)$ according to equation (5)
3) *Step 2: Weighting.*
   The reference vectors (prototypes) and the allocation

function are kept fixed.
Compute the components λ_{rj} of the vectors of weights $\boldsymbol{\lambda}_r\,(r = 1, \ldots, m; j = 1, \ldots, p)$ according to equation (10)

4) *Step 3: Affectation.*
The prototypes $\mathbf{w}_c^{(t)}\,(c = 1, \ldots, m)$ are fixed. Assign each individual $\mathbf{x}_i\,(i = 1, \ldots, n$ to its nearest neuron according to equation (9)
5) *Stopping criterion.*
If $T = T_{min}$ then STOP; otherwise set $t = t + 1$ and go to 2 (Step 1).

V. Experimental Results

To show the usefulness of this method, experiments comparing the batch SOM [12] and the adaptive batch SOM applied on real quantitative benchmark datasets of different degrees of clustering difficulty obtained from UCI Machine Learning Repository [14] are considered.

In order to compare the clustering results furnished by the clustering methods applied on the quantitative data sets considered in this paper, an external index the corrected Rand index (CR) [15], the overall error rate of classification ($OERC$) [16] and the F-Measure [17] will be considered. The CR index assesses the degree of agreement (similarity) between an a priori partition and a partition furnished by the clustering algorithm. We used the CR index because it is not sensitive to the number of classes in the partitions or the distribution of the items in the clusters. The CR index takes its values from the interval [-1,1], in which the value 1 indicates perfect agreement between partitions, whereas values near 0 (or negatives) correspond to cluster agreement found by chance [18].

As pointed out by [12] 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 I gives the parameters for both batch SOM and adaptive batch SOM algorithms. They were fixed after several tests with these algoritms.

TABLE I
Parameters List

Parameter	Value
N_{Iter}	500
m	16 (2 × 8 grid)
T_{max}	3.5
T_{min}	0.5

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

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

A. Glass identification dataset

This dataset consists in a study of the classification of types of glass motivated by criminological investigation. This dataset consists of seven types (classes) of glasses: building windows float processed (1), building windows non-float processed (2), vehicle windows float processed (3), vehicle windows non-float processed (4, none in this database), containers (5), tableware (6) and headlamps (7). The classes (1, 2, 3, 4, 5, 6 and 7) have, respectively, 70, 17, 76, 0, 13, 9 and 29 instances. Each class is described by nine real-valued attributes: RI (refractive index), Na (Sodium), Mg (Magnesium), Al (Aluminum), Si (Silicon), K (Potassium), Ca (Calcium), Ba (Barium) and Fe (Iron).

The self-organizing algorithms were applied on this dataset. Tables II and III give the confusion matrix (only non-empty clusters) obtained with the adaptive batch SOM and with the batch SOM algorithms , respectively. The CR, F-Measure and $OERC$ indexes were, respectively, 0.165, 0.429 and 0.383 for the the adaptive bath SOM algorithm, and were, 0.223, 0.514 and 0.444 for the batch self-organizing map, respectively. For this dataset, the adaptive bath SOM algorithm outperforms the batch SOM algorithm concerning the $OERC$. The performance of the adaptive bath SOM algorithm measured by the CR and F-Measure indices was worse than that obtained by the bath SOM algorithm.

TABLE II
Glass dataset: confusion matrix computed with the adaptive batch SOM algorithm

Cluster \ Class	1	2	3	5	6	7
1	0	0	0	1	2	22
2	0	1	0	3	0	4
5	0	1	0	0	0	0
6	0	1	0	0	0	0
7	2	14	4	0	0	0
8	4	19	0	0	0	0
9	34	20	6	0	0	0
11	10	5	3	0	0	0
12	0	0	0	0	0	1
13	0	0	0	2	0	1
15	0	4	0	3	0	0
16	20	11	4	4	7	1

TABLE III
Glass dataset: confusion matrix computed with the batch SOM algorithm

Cluster \ Class	1	2	3	5	6	7
1	44	55	8	1	1	2
2	13	7	6	2	4	6
6	13	3	3	0	0	1
9	0	0	0	0	0	1
10	0	7	0	0	0	0
15	0	4	0	2	2	22
16	0	4	0	8	2	0

B. Iris plant dataset

This dataset consists of three types (classes) of iris plants: iris setosa, iris versicolour and iris virginica. The three classes each have 50 instances. One class is linearly separable from the other two; the latter two are not linearly separable from each other. Each class is described by four realvalued attributes: sepal length, sepal width, petal length and petal width.

The self-organizing algorithms were applied to this dataset. Table IV and V show the confusion matrix obtained with the adaptive batch SOM and with the batch SOM algorithms, respectively (only non-empty clusters).

TABLE IV
IRIS DATASET: CONFUSION MATRIX COMPUTED WITH THE ADAPTIVE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3
1	20	0	0
2	30	0	0
7	0	1	0
8	0	3	0
9	0	9	0
11	0	1	0
12	0	10	1
14	0	7	1
15	0	1	39
16	0	18	9

TABLE V
IRIS DATASET: CONFUSION MATRIX COMPUTED WITH THE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3
1	19	0	0
2	21	1	0
5	8	0	0
6	2	0	0
11	0	0	15
12	0	11	0
15	0	3	22
16	0	35	13

The CR, F-Measure and $OERC$ indexes were, respectively, 0.483, 0.599 and 0.08 for the adaptive batch SOM algorithm and were, respectively, 0.372, 0.570 and 0.113 for the batch SOM algorithm. For this dataset, the adaptive bath SOM algorithm outperforms the bath SOM algorithm concerning these indices.

C. Pima indians diabetes dataset

This dataset consists of two possible diagnostics (classes) given to a sample of 768 females at least 21 years old of Pima Indian heritage, concerning about when a patient shows signs of diabetes according to World Health Organization criteria or not. The classes (1 for a healthy patient, and 2 for a patient interpreted as "tested positive for diabetes") have, respectively, 500 and 268 instances. Each class is described by eight real-valued attributes: number of times pregnant, plasma glucose concentration a 2 hours in an oral glucose tolerance test, diastolic blood pressure (mm Hg), triceps skin fold thickness (mm), 2-Hour serum insulin (mu U/ml), body mass index, diabetes pedigree function and age (in years).

The self-organizing algorithms were applied to this dataset. Table VI and VII give the confusion matrix obtained with the adaptive batch SOM and with the batch SOM algorithms, respectively (only non-empty clusters).

The CR, F-Measure and $OERC$ indices were, respectively, 0.049, 0.446 and 0.320 for the the adaptive bath self-organizing map algorithm. The CR, F-Measure and $OERC$ indices were, respectively, 0.012, 0.364 and 0.338 for the batch self-organizing map. For this dataset, the adaptive bath self-organizing map algorithm outperforms the bath self-organizing map algorithm concerning these indices.

TABLE VI
PIMA INDIANS DIABETS DATASET: CONFUSION MATRIX COMPUTED WITH THE BATCH SOM ALGORITHM

Cluster \ Class	1	2
1	87	18
2	149	120
15	197	41
16	67	89

TABLE VII
PIMA INDIANS DIABETS DATASET: CONFUSION MATRIX COMPUTED WITH THE BATCH SOM ALGORITHM

Cluster \ Class	1	2
1	243	140
2	95	7
4	22	4
6	42	14
7	24	21
8	2	3
9	21	22
11	24	24
13	0	2
15	14	14
16	13	17

D. Image segmentation dataset

This dataset consists of images that were drawn randomly from a database of seven outdoor images. The images were segmented by hand to create the seven class labels: sky, cement, window, brick, grass, foliage and path. Each class has 330 instances and is described by 18 real-valued attributes. These attributes are: region-centroidcol; region-centroidrow; short-line-density-5; short-line-density-2; vedgemean; vegdesd; hedge-mean; hedge-sd; intensity-mean; rawred-mean; rawblue-mean; rawgreen-mean; exred-mean; exblue-mean; exgreen-mean; value-mean; saturation-mean and hue-mean.

The self-organizing algorithms were applied to this dataset. Table VIII and IX show the confusion matrix obtained with the adaptive batch SOM and with the batch SOM algorithms, respectively (only non-empty clusters).

The CR, F-Measure and $OERC$ indexes were, respectively, 0.391, 0.562 and 0.431 for the adaptive bath SOM algorithm, and were, respectively, 0.105, 0.348 and 0.644 for the batch SOM. For this dataset, the adaptive bath SOM algorithm also outperforms the bath SOM algorithm concerning these indexes.

E. Thyroid gland dataset

This dataset consists of three classes concerning the state of the thyroid gland: normal, hyperthyroidism and hypothyroidism. The classes (1, 2 and 3) have, respectively, 150, 35 and 30 instances. Each class is described by five real-valued attributes: T3-resin uptake test, total serum thyroxin, total

TABLE VIII

IMAGE SEGMENTATION DATASET: CONFUSION MATRIX COMPUTED WITH THE ADAPTIVE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3	5	6	7	
1	114	0	14	32	49	0	325
2	215	0	271	26	266	0	4
3	1	0	1	0	1	0	0
5	0	0	0	8	2	6	1
7	0	0	0	17	0	63	0
8	0	0	0	91	0	65	0
9	0	0	0	24	0	75	0
10	0	1	0	71	0	58	0
11	0	0	1	25	0	15	0
12	0	3	0	4	0	0	0
13	0	0	9	8	0	3	0
15	0	0	24	24	12	45	0
16	0	326	10	0	0	0	0

TABLE IX

IMAGE SEGMENTATION DATASET: CONFUSION MATRIX COMPUTED WITH THE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3	5	6	7	
1	0	120	0	0	0	0	0
2	0	142	4	51	0	0	0
5	0	59	0	0	0	0	0
6	0	9	0	0	0	0	0
11	72	0	71	10	24	3	15
12	13	0	9	31	40	26	23
15	162	0	183	88	68	127	150
16	83	0	63	150	198	174	142

serum triiodothyronine, basal thyroidstimulating hormone (TSH) and maximal absolute difference in TSH value.

The self-organizing algorithms were applied to this dataset. Table X and XI show the confusion matrix obtained by adaptive batch self-organizing map and batch self-organizing map algorithms, respectively (only non-empty clusters).

TABLE X

THYROID DATASET: CONFUSION MATRIX COMPUTED WITH THE ADAPTIVE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3
1	0	0	14
2	0	0	10
8	0	0	1
10	1	0	1
15	17	35	0
16	132	0	4

The CR, F-Measure and $OERC$ indexes were, respectively, 0.670, 0.844 and 0.102 for the adaptive batch SOM algorithm, and were, respectively, 0.112, 0.390 and 0.139 for the batch SOM algorithm. For this dataset, the bath SOM algorithm is outperformed by the adaptive bath SOM algorithm concerning these indexes.

F. Wine dataset

This dataset consists of three types (classes) of wines grown in the same region in Italy. but derived from three different cultivars. The classes (1, 2 and 3) have, respectively,

TABLE XI

THYROID DATASET: CONFUSION MATRIX COMPUTED WITH THE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3
1	9	25	1
2	33	3	2
4	3	0	0
6	33	3	1
8	20	1	0
9	5	0	0
10	27	0	0
11	1	0	2
12	11	0	1
14	1	0	0
15	0	0	18
16	7	3	5

59, 71 and 48 instances. Each wine is described by 13 realvalued attributes representing the quantities of 13 components found in each of the three types of wines. These attributes are: (1) alcohol; (2) malic acid; (3) ash; (4) alkalinity of ash; (5) magnesium; (6) total phenols; (7) flavonoids; (8) non-flavonoid phenols; (9) proanthocyanins; (10) colour intensity; (11) hue; (12) OD280/OD315 of diluted wines and (13) proline.

The self-organizing algorithms were applied to this dataset. Table XII and XIII show the confusion matrix obtained by adaptive batch self-organizing map and batch self-organizing map algorithms, respectively (only non-empty clusters).

TABLE XII

WINE DATASET: CONFUSION MATRIX COMPUTED WITH THE ADAPTIVE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3
1	7	21	0
2	52	2	0
5	0	5	0
7	0	11	0
9	0	3	0
10	0	1	0
11	0	1	0
13	0	1	0
15	0	23	0
16	0	3	48

TABLE XIII

WINE DATASET: CONFUSION MATRIX COMPUTED WITH THE BATCH SOM ALGORITHM

Cluster \ Class	1	2	3
1	0	45	14
2	0	11	15
4	1	6	6
6	3	5	7
7	3	0	4
9	4	4	2
10	2	0	0
11	3	2	0
13	2	0	0
15	18	0	0
16	23	0	0

The CR, F-Measure and $OERC$ indexes were, respec-

tively, 0.578, 0.709 and 0.067 for the the adaptive bath SOM algorithm, and were, respectively, 0.267, 0.457 and 0.275 for the batch SOM algorithm. For this dataset, the adaptive bath SOM algorithm also outperforms the bath SOM algorithm concerning these indexes.

In conclusion, for the majority of these UCI machine learning data sets, the adaptive batch SOM algorithm performs better than the batch SOM algorithm concerning the CR, F-Measure and $OERC$ indexes.

VI. Conclusions

In this paper, an adaptive batch self-organizing maps algorithm was presented. This method combined the best visualization and clustering characteristics provided by SOM neural networks with the flexibility offered by adaptive distances in recognition of clusters with different shapes and sizes.

Experiments with real benchmark datasets showed the usefulness of this method. The accuracy of the results furnished by this method was assessed by the corrected Rand index (CR), the F-Measure criterion and the overall error rate of classification ($OECR$) considering applications with real datasets. The adaptive batch self-organizing maps outperformed the traditional batch self-organizing map algorithm for the datasets selected from the UCI machine learning repository concerning the identification of a priori classes of patterns.

References

[1] B. Everitt, *Cluster Analysis*. New York: Halsted, 2001.

[2] A. D. Gordon, *Classification*. Boca Raton, FL: CRC Press, 1999.

[3] H. Spaeth, *Cluster Analysis Algorithms*. New York: Wiley, 1980.

[4] S. Theodoris and K. Koutroumbas, *Pattern Recognition*. Academic Press, 3a ed., 2006.

[5] A. Jain, M. Murty, and P. Flynn, "Data clustering: A review", *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264323, 1999.

[6] J. MacQueen, "Some methods for classification and analysis of multivariate observations", in *Proc. of the Fifth Berkeley Symposium on Math., Stat. and Prob.*, vol. 1, pp. 281-296.

[7] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[8] E. Diday and J. C. Simon, "Clustering analysis", in *Digital Pattern Classification*, K. Fu, Ed. Berlin, Germany: Springer-Verlag, 1976, pp. 4794.

[9] E. Diday and G. Govaert, "Classification automatique avec distances adaptatives", R.A.I.R.O. *Informatique Comput. Sci.*, vol. 11, no. 4, pp. 329349, 1977.

[10] T. Kohonen, *Self-Organisation Maps*. Springer Verlag, New York.

[11] J. A. Kangas, T. K. Kohonen and J. T. Laaksonen, "Variants of self-organizing maps", in *IEEE Transactions on Neural Networks*, vol. 1, no. 1, 1990.

[12] F. Badran, M. Yacoub and S. Thiria, "Self-organizing maps and unsupervised classification" in *Neural Networks: methodology and applications*, G. Dreyfus, Ed. Berlin, Germany: Springer-Verlag, 2005, pp. 379-442.

[13] F. Anouar, F. Badran and S. Thiria, "Self Organized Map, a Probabilistic Approach", in *Proceedings of the Workshop on Self-Organized Maps*. Helsinki University of Technology, Espoo, Finland, June 4-6, 1997.

[14] A. Frank and A. Asuncion, UCI Machine Learning Repository, Univ. California, Sch. Inform. Comput. Sci., Irvine, CA, 2011 [Online]. Available: http://archive.ics.uci.edu/ml.

[15] L. Hubert and P. Arabie, "Comparing partitions", in *Journal of Classification*, vol. 2, pp. 193218, 1985.

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

[17] C. J. van Rijsbergen, *Information Retrieval*, London: Butterworths, 1979

[18] G. W. Milligan. *Clustering Validation: results and implications for applied analysis* Word Scientific, 1996.