Neural Process Lett (2016) 43:849–869
DOI 10.1007/s11063-015-9454-1

Constrained Self Organizing Maps for Data Clusters Visualization

Ehsan Mohebi[1] · **Adil Bagirov**[1]

Published online: 4 July 2015

Abstract High dimensional data visualization is one of the main tasks in the field of data mining and pattern recognition. The self organizing maps (SOM) is one of the topology visualizing tool that contains a set of neurons that gradually adapt to input data space by competitive learning and form clusters. The topology preservation of the SOM strongly depends on the learning process. Due to this limitation one cannot guarantee the convergence of the SOM in data sets with clusters of arbitrary shape. In this paper, we introduce Constrained SOM (CSOM), the new version of the SOM by modifying the learning algorithm. The idea is to introduce an adaptive constraint parameter to the learning process to improve the topology preservation and mapping quality of the basic SOM. The computational complexity of the CSOM is less than those with the SOM. The proposed algorithm is compared with similar topology preservation algorithms and the numerical results on eight small to large real-world data sets demonstrate the efficiency of the proposed algorithm.

Keywords Self organizing maps · Vector quantization · Clustering · SOM learning algorithm · CSOM

1 Introduction

High dimensional data visualization is used to analyze the hidden patterns and data relationships, which are hard to illustrate. One of the most popular and classical methods for dimension reduction and data visualization is principal component analysis, PCA [14]. Linear PCA is susceptible to lose useful information in highly nonlinear data sets. Several nonlinear PCA methods have been proposed such as the work presented in [24]. Linear and nonlinear

✉ Ehsan Mohebi
eh.mohebi@gmail.com

Adil Bagirov
a.bagirov@federation.edu.au

[1] School of Science, Information Technology and Engineering, Federation University Australia, Victoria 3353, Australia

PCA are computationally inexpensive in large data sets however, the memory required by these algorithms is not affordable in very large data sets. The Sammon's mapping [23] is a nonlinear mapping algorithm for visualization of multivariate data, which is shown to be superior to PCA. Neural networks specially the self organizing maps [15] are able to learn complex nonlinear relationships of variables in a sample of data points.

The self organizing map is one of the well known data mining tools where the aim is to visualize a high dimensional data space into usually a 2-Dim grid [4,10,12,20,25]. The SOM contains a set of neurons that gradually adapts to input data space by competitive learning and creates ordered prototypes. The ordered prototypes preserve the topology of the mapped data and make the SOM to be very suitable for cluster analysis [31]. This adaption is based on a similarity measure, which is usually Euclidean distance, and repositioning of neurons in a 2-Dim space using a learning algorithm. The performance of the SOM strongly depends on a learning algorithm [6,7,9,11].

Different versions of the SOM for visualization of high dimensional data have been introduced in [5,13,17,29,30,32,33]. An alternative to the SOM algorithm, ViSOM, is introduced by [33] to improve the topological and quantization errors by restricting contractions of neurons. The authors proposed different updating rules for the winner neurons and their neighborhood neurons, which is computationally expensive in very large data sets. The analysis and experimental results show that the ViSOM may offer attractive advantages over the commonly used SOM, PCA, and Sammon's mapping. The PRSOM [30] is an extension to ViSOM, where the sequential updating rules are extended to optimize a cost function. Unlike the hard assignment in SOM and ViSOM, the assignment of PRSOM is soft such that an input data point belongs to a neuron with certain probability. However, the computational complexity of both ViSOM and PRSOM is shown to be more than the classical SOM and depends quadratically on the number of neurons [30].

In [3], a new self organizing model, the so called growing grid (GG), is proposed to the problem of data visualization. The network automatically chooses a height/width ratio suitable for the data distribution. Moreover, locally accumulated statistical values are used to determine where to insert new units (neurons). The growing neural gas (GNG), introduced in [8], is an improvement to the neural gas (NG) algorithm [18]. More specifically, it is an incremental version of the NG algorithm which does not require the pre-setting of the network size. The GNG algorithm is able to make explicit topological relations of input data.

The growing hierarchical SOM (GHSOM) generates multiple independent layers of the maps on the top of the first layer to cluster input data points in a hierarchical manner and allows for adaptation of the network architecture simultaneously [21]. The GHSOM is presented to be suitable for visualization of high dimensional document data sets rather than the SOM, although manipulating multiple layers of the GHSOM is liable to high computational effort in very large data sets. Following the training phase of the GHSOM, a novel visualization method, namely, the ranked centroid projection (RCP), is introduced in [32] for projecting the document vectors to a hierarchy of output maps. The results of the DB index show that RCP produces a better result, which denotes that for the purpose of classification and categorization, the RCP is to be preferred over the SOM, PCA and Sammon's mapping [32].

The expanding SOM, ESOM, for data visualization is proposed in [13]. The ESOM utilizes the SOM by employing an additional factor, the expanding coefficient, which is used to push neurons away from the center of all data points during the learning process. The authors show that the ESOM has advantages over the SOM and the growing models of the SOM, where the growth criteria in the latter increases the computational time in the learning process. To cope with the problems of selecting learning rate in conventional SOM, [5] proposed RPSOM in which, for each input data point, the RPSOM adaptively chooses several rivals of the

BMU and penalizes their associated models a little far away from the input data point [5]. The RPSOM converges much faster than the SOM and the two-phase SOM [27] moreover, outperforms them in the term of quantization error.

In this paper, a similar approach to the RPSOM is proposed to improve the performance of the RPSOM and the SOM in the scene of quantization error. Instead of penalizing the rivals of the BMU, which increases the training steps, an adaptive constraint parameter is used in the learning process. The constraint parameter is chosen as a decreasing function with respect to iterations and we consider three such functions: linear, hyperbolic and sigmoid. This parameter restricts the process of updating neighborhoods of the BMU to only those neighbors which are close in the n-dimensional space. Such an approach leads to faster convergence and better local minimum of the quantization error than that of by the RPSOM and the SOM. Furthermore, the distortion error and topology preservation are improved. The proposed algorithm is tested using eight real-world data sets.

The rest of the paper is organized as follows. The basic self organizing maps is presented in Sect. 2. The modified learning algorithm of the SOM and the adaptive constraints are introduced in Sect. 3. The CSOM algorithm and its implementation are discussed in Sect. 4. Numerical results are presented in Sects. 5 and 6 concludes the paper.

2 Self Organizing Maps

The SOM is an unsupervised neural network [15] that usually contains a 2-Dim array of neurons. Assume that we are given the set of m input data vectors $A = \{x_1, \ldots, x_m\}$ where $x_i \in \mathbb{R}^n,\ i = 1, \ldots, m$. In the SOM, a set of q neurons, $\Psi = \{w_1, \ldots, w_q\}$, $w_j \in \mathbb{R}^n$, is given.

One data point $x_i,\ i \in \{1, \ldots, m\}$ at a time is presented to the network. The data point x_i is compared with all weight vectors. The nearest $w_j,\ j = 1, \ldots, q$ is selected as the *best matching unit* (BMU) for the i-th data point. The data point x_i is mapped to the best matching neuron.

The set of neighborhood weights $N_c = \{w_l : p(c, l) \leq r,\ l \neq c\}$ around the BMU are updated where $p(c, l)$ is the distance between the BMU and the neighborhood neuron l in 2-Dim coordinates of the network topology and r is the predefined radius. Here $p(c, l) \in \mathbb{N}$ and $0 < p(c, l) \leq r$. Usually the quantization error is used to show the quality of the map, therefore, the aim is to solve the following problem:

$$\text{minimize } E = \frac{1}{m} \sum_{i=1}^{m} \|x_i - w_c\|, \tag{1}$$

where w_c is the weight of the BMU of $x_i,\ i = 1, \ldots, m$. A general description of the SOM algorithm is as follows.

Algorithm 1 SOM algorithm

Step 1. Initialize the dimension of the network, the maximum number of iterations (T), a radius (r) of the network and weight vectors $w_j,\ j = 1, \ldots, q$. Set stopping criterion ϵ_0 and iteration counter $\tau := 0$.

Step 2. Select data $x_i, i = 1, \ldots, m$ and find its closest neuron c, that is

$$c := \underset{j=1,\ldots,q}{\operatorname{argmin}} \|x_i - w_j\|. \tag{2}$$

Step 3. Update the set of neighborhood neurons $w_j \in N_c$ using the following equation:

$$w_j := w_j + \alpha(\tau)h(\tau)(x_i - w_j). \tag{3}$$

(Here h is a neighborhood function and $\alpha(\tau)$ is a learning rate at the iteration τ.)

Step 4. If all input data are presented to the network go to Step 5, otherwise go to Step 2.

Step 5. Calculate E using (1). If $E < \epsilon_0$ or $\tau > T$ terminate, otherwise set $\tau := \tau + 1$ and go to Step 2.

Note that the neighborhood function in equation (3) of Algorithm 1 is as follows.

$$h(\tau) = \exp\left(-\frac{r^2}{2\sigma(\tau)^2}\right), \tag{4}$$

where

$$\sigma(\tau) = \eta\frac{T-\tau}{T}, \quad \eta \in \mathbb{R}, \tag{5}$$

and usually $\eta \geq 1$.

The neighborhood function h in Step 3 of Algorithm 1 plays an important role in the SOM. Usually h is a decreasing exponential function of τ. The value of neighborhood functions depends on iteration τ and the distance in the output space (r) of each neuron in the set N_c. The learning rate α is a decreasing linear function of τ that reduces the effect of the neighborhood function h as $\tau \to T$.

In this paper, for given $w_j,\ j \in \{1, \ldots, q\}$ we define the following set:

$$S_j = \{x_k : d(x_k, w_j) < d(x_k, w_l), \quad l \neq j, \quad l = 1, \ldots, q\} \tag{6}$$

where

$$d(x, y) = \|x - y\| = \left(\sum_{t=1}^{n}(x^t - y^t)^2\right)^{1/2}, \quad x, y \in \mathbb{R}^n$$

is the Euclidean distance.

The learning procedure of SOM is explored in details in the next section.

3 CSOM Learning Algorithm

In this section the learning process of the Self Organizing Map is analyzed, furthermore, the quantization performance of the SOM is criticized. As we mentioned in Sect. 2, the set N_c in Step 3 of Algorithm 1 contains all neighborhood neurons which are connected to the BMU. These neurons are selected using the parameter r which is the radius around the BMU in the topological 2-dimensional map. The SOM updates all neighborhood neurons of the BMU in the topological map using (3), although relative weights of these neurons may be far from the BMU in the n-dimensional space. In this case, the value of E in (1) deviates from its optimal value as the neighborhood neurons in the network topology, which are not in the neighborhood of the BMU, are relocated according to (3) (see Fig. 1). The relocation of the set of neighborhood neurons, $M_c,\ M_c \subset N_c$, which are far from the winning neuron in the n-dimensional space is illustrated in the Fig. 1. Note that the set M_c depends on iteration τ.

Assume that the SOM algorithm is learning the input data points. In some stage, the relocation of the neighborhood neurons in the set M_c increases the inter-neuron distances

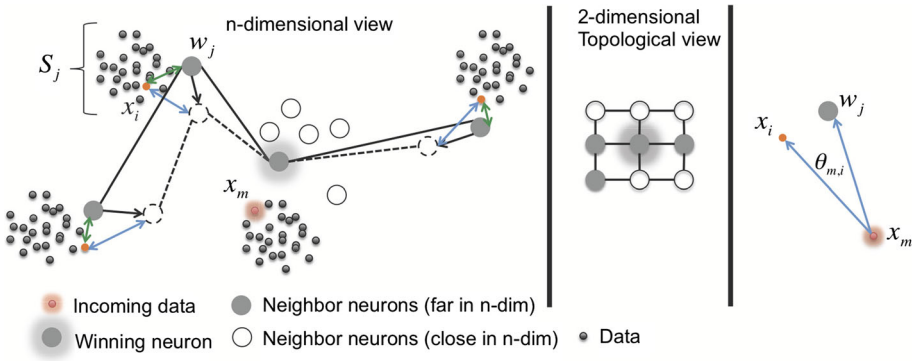

Fig. 1 Contraction of neighbor neurons of the BMU

(see Fig. 1). Let have a given data point x_m, which activates the nearest neuron in the map at iteration τ. The neighborhood neurons of the BMU move towards the data point x_m on the line intersecting both x_m and w_j, $w_j \in N_c$. Based on the updating formula in (3) the distance that neuron w_j moves is $\|\Delta w_j\|$ where,

$$\Delta w_j = \beta(x_m - w_j), \quad 0 \leq \beta < 1.$$

Therefore, by applying the updating rule to the neuron w_j, there exist data points $x_i \in S_j$, that the value

$$\Delta d(x_i, w_j) > 0,$$

after updating the neuron w_j, if

$$d(x_m, w_j) \leq \cos(\theta_{m,i}) d(x_m, x_i), \tag{7}$$

where $\theta_{m,i}$ is the angle between vectors $(x_i - x_m)$ and $(w_j - x_m)$ (see Fig. 1). Let have the subset of data points, $Q_j \subset S_j$, where every data point $x_i \in Q_j$ holds the condition (7). Thus, the increment in the value of inter-neuron distance, ΔD_j, is:

$$\Delta D_j = \sum_{i=1}^{|Q_j|} \Delta d(x_i, w_j), \tag{8}$$

where, $|.|$ is the cardinality of a set of data. Furthermore, consider a subset of neurons $F_c \subset N_c$ including neurons w_j, which are far from the BMU in the n-dimensional space. Therefore, the value E in (1) increases by:

$$\Delta E = \sum_{j=1}^{|F_c|} \Delta D_j, \tag{9}$$

after the data point x_m is presented to the network and (3) is applied to the set N_c.

3.1 Modified Learning Algorithm

In this section we design an algorithm to minimize ΔE given by (9) for w_j, $j = 1, \ldots, |N_c|$ at iterations $\tau > T/\rho$. The idea is to modify the update formula (3) in Step 3 of Algorithm 1

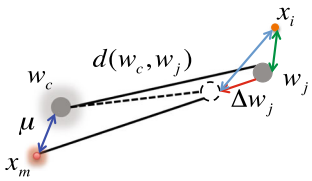

Fig. 2 The updating procedures in the SOM algorithm

by adding a constraint parameter. We introduce a constraint parameter $\gamma \in \mathbb{R}$, which defines a subset C_c of the set of neighborhood neurons N_c which are close to the BMU:

$$C_c = \{w_j : d(w_c, w_j) < \gamma d_{min},\ w_j \in N_c\}, \tag{10}$$

where $\gamma > 1$ and

$$d_{min} = \min \left\{ d(w_c, w_j) : w_j \in N_c \right\}. \tag{11}$$

Also we have

$$\psi = \frac{d_{max}}{d_{min}}, \tag{12}$$

and

$$d_{max} = \max\{d(w_c, w_j) : w_j \in N_c\}. \tag{13}$$

One can see from (10) that $|C_c| < |N_c|$ for values of γ close to 1 and both sets coincide as $\gamma \to \infty$. The set C_c contains neighborhood neurons in a 2-Dim topological space while preserving the condition of adjacency of $w_j \in C_c$ in the n-dimensional space. In the step 3 of Algorithm 1, if we update neighborhood neurons of the set C_c then the weight vectors, which are far from the best matching unit, will not be considered in Eq. (3). Therefore, following (10):

$$d_{min} \leq d(w_c, w_j) < \gamma d_{min} \ \ \forall\, w_j \in C_c, \quad \gamma > 1. \tag{14}$$

Since the distance $d(w_c, w_j),\ w_j \in C_c$ is sufficiently small, it is easy to see that $\Delta d(x_i, w_j)$ in (8), where

$$\Delta d(x_i, w_j) < d(w_c, w_j),$$

is depended on the upper bound, $d(w_c, w_j)$ (see Fig. 2).

Therefore, by limiting the value of $d(w_c, w_j)$, then we have:

$$\Delta d(x_i, w_j) < \gamma d_{min}.$$

One can see that the values ΔD_j in (9) are minimized.

Thus, the constrained learning algorithm can be summarized as follows.

Algorithm 2 Constrained learning algorithm

Step 1. $C_c = \emptyset$, select γ and find d_{min} from N_c using Eq. (11).
Step 2. Select neuron $w_i \in N_c$, if $\|w_i - w_c\| < \gamma d_{min}$ then

$$C_c = C_c \cup w_i$$

Step 3. If all neurons in N_c have been visited then goto 4, otherwise goto 2.
Step 4. Update the set of neighborhood neurons $w_j \in C_c$ using Eq. (3).

Thus, we minimize the Eq. (9) by applying Algorithm 2 in the step 3 of Algorithm 1 at iteration τ.

3.2 Adaptive Selection of Parameter γ

In this section, we propose linear, hyperbolic and sigmoid formulation of the constraint parameter γ in Eq. (10) with respect to iteration τ. Considering Algorithm 2, one can see that fixed values of γ reduce the interaction of neurons in early iterations thus, the neurons do not converge to the domain of input data accurately. From Eq. (10) it is easy to see that $C_c \equiv N_c$ for the large values of γ. Therefore the behavior of the algorithm for large values of γ is similar to that of classical SOM.

In Step 2 of Algorithm 2 we consider γ not as a fixed constant but parameter depending on iteration τ. Moreover, $\gamma(\tau)$ decreases as τ increases. Hence, to ensure the validity of the proposed formulations, the constraint functions allow the algorithm to perform similar to SOM in early stages and gradually applies the constrained learning algorithm. In order to have such property we require γ to satisfy the following conditions: there exist $\bar{\tau} > 1$ such that:

$$\begin{cases} \gamma(\tau) > \psi & 1 < \tau < \bar{\tau} \\ 1 < \gamma(\tau) \leq \psi & \bar{\tau} \leq \tau < T \end{cases} \tag{15}$$

It is clear there are different ways to choose γ satisfying the conditions (15). In this paper, we will define γ as a linear, hyperbolic and sigmoid constraint functions of τ.

3.2.1 Linear formulation

In the case of linear constraint function one can approximate γ as follows:

$$\gamma(\tau) = \frac{\rho}{(1-\rho)}(\psi - 1)\left(\frac{\tau}{T} - \frac{1}{\rho}\right) + \psi, \tag{16}$$

where $\rho \in \mathbb{N}$ defines the $\bar{\tau} = T/\rho$ which is the iteration from which the constrained learning algorithm is applied to Algorithm 2. It is easy to see that the function (16) satisfies condition (15).

3.2.2 Hyperbolic Formulation

The hyperbolic expression for γ is given as

$$\gamma(\tau) = \begin{cases} \infty & 1 < \tau < \bar{\tau} \\ \frac{\psi}{\tau - \varphi} + 1 & \bar{\tau} < \tau < T \end{cases} \tag{17}$$

where

$$\varphi = \bar{\tau} - \frac{d_{max}}{d_{max} - d_{min}},$$

and $\bar{\tau} = T/\rho$.

3.2.3 Sigmoid Formulation

Finally, we propose the following sigmoid function to define γ:

$$\gamma(\tau) = \varepsilon(\psi - 1)\tanh\left(\frac{-\tau + \bar{\tau}}{\delta}\right) + \frac{1}{2}(\psi + 1), \tag{18}$$

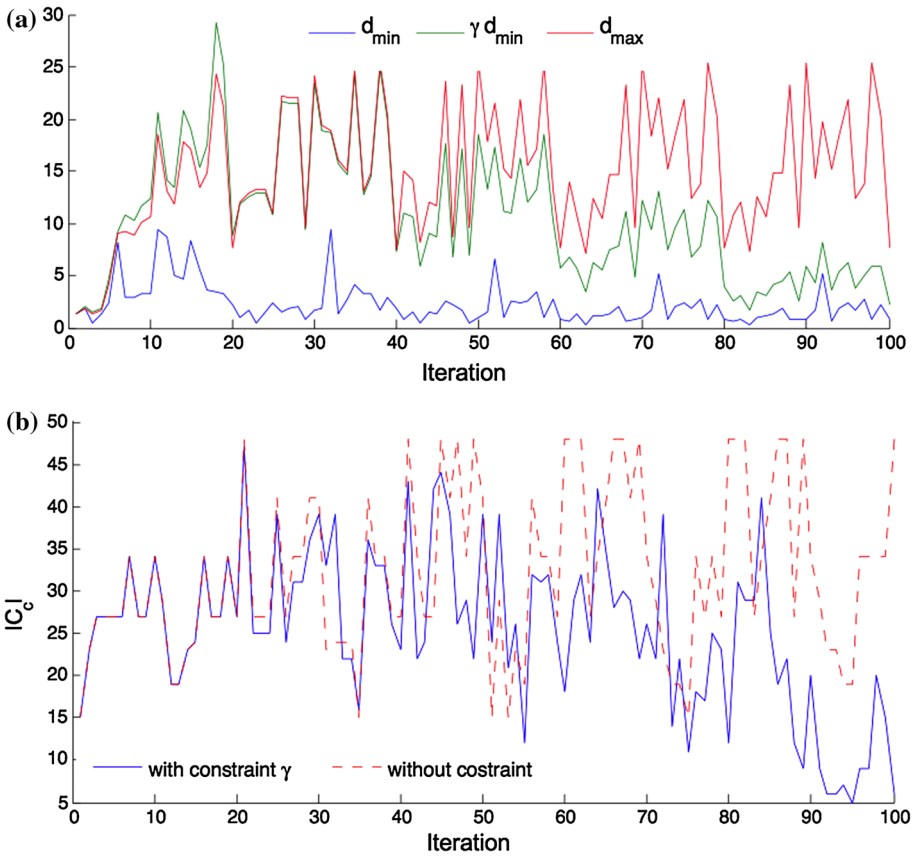

Fig. 3 Performance of constraint γ using equation (16) on 20×20 SOM where $r = 3$, $\rho = 5$, $T = 5$ and $|A| = 20$. **a** Performance of γ, **b** comparing $|C_c|$ with and without constraint

where $0 < \varepsilon < 1$ and $1 < \delta < T$ defines the slope of changes depending on iterations. It should be noted that if δ is chose close to 1 then we have a rapid changes in constraint function (18).

To explain the idea, the performance of the proposed algorithm is presented in Fig. 3. The Fig. 3a shows that from iteration 2 onwards ($\tau > \bar{\tau}$) the algorithm applies the constraint to the learning process and γd_{min} is decreasing and always less than d_{max} as $\tau \to T$, consequently, the cardinality of C_c in (10) decreases (see Fig. 3b). The algorithm prevents further modification to the neighborhood neurons which are far from the BMU in the n-dimensional space.

4 The CSOM Algorithm and Its Implementation

In this section, we apply adaptive formulations of constraint γ in Step 2 of Algorithm 2. Then Algorithm 1 for solving vector quantization problem (1) can be rewritten as follows.

Algorithm 3 CSOM algorithm

Step 1. Initialize dimension, maximum iteration (T), radius (r) of the network and finally weight vectors w_j, $j = 1, \ldots, q$. Set $\tau := 1$.

Table 1 Initialization of SOM parameters in Algorithm 3

Data sets	Input size	SOM dimension	r	T
Small	$(\|A\| < 10^3)$	10×10	2	200
Medium	$(10^3 < \|A\| < 10^4)$	15×15	2	300
Large	$(10^4 < \|A\| < 0.5 \times 10^5)$	20×20	2	300
Very large	$(0.5 \times 10^5 < \|A\| < 0.8 \times 10^5)$	15×15	3	100
	$(\|A\| > 0.8 \times 10^5)$	25×25	3	30

Step 2. Select data x_i and find closest neuron c, where

$$c = \underset{j}{\operatorname{argmin}} \|x_i - w_j\|. \tag{19}$$

Step 3. Compute the set N_c and d_{min} and d_{max} using Eqs. (11) and (13) respectively.
Step 4. Compute $\gamma(\tau)$ using Eqs. (16), (17) or (18) and set $C_c = \emptyset$.
Step 4.1 Select neuron $w_i \in N_c$, if $\|w_i - w_c\| < \gamma(\tau) d_{min}$ then

$$C_c = C_c \cup w_i.$$

Step 4.2 If all neurons in N_c have been visited then goto Step 5, otherwise goto Step 4.2.
Step 5. Update the set of neighborhood neurons C_c using the following equation:

$$w_j(t+1) = w_j(t) + \alpha(\tau) h(\tau)(x_i - w_j(t)), \ \ w_j \in C_c. \tag{20}$$

Step 6. If all input data x_i are presented to the network go to Step 7 otherwise, go to Step 2.
Step 7. Calculate E_τ using Eq. (1) and if $\tau > T$ terminate otherwise set $\tau = \tau + 1$ and go to step 2.

In Algorithm 3, weight vectors w_j, $j = 1, \ldots, q$ are initialized with a uniform pattern in order to be comparable with the basic SOM. The maximum number of iterations T is set according to size and dimension of data sets. We set T large for small data sets because for such data sets the time complexity is less to obtain stable network over input data. The topology of network is rectangular [15] with the same number of neurons in each column and row (i.e. $n \times n$). Each interior neuron is connected with 8 neighborhood neurons, however this number is less than 5 for border neurons. Furthermore, the radius of the map r is set to 2 and 3 for small and large number of neurons (see Table 1). The dimension of the map is selected based on the dimensionality of data sets thus, for very large data sets $(0.5 \times 10^5 < |A| < 0.8 \times 10^5)$ with high dimension (high time complexity) the size of map is set to 15×15, whereas for 2-dimensional very large data sets $(|A| > 0.8 \times 10^5)$ it is set to 25×25.

In step 4 of Algorithm 3, we set values of $\rho \in \{\rho : 2 \leq \rho \leq 0.7 \times T, \ \rho \in \mathbb{N}\}$ for Eqs. (16) and (17) which depends on data set size and the maximum iteration number T. If the number of data points is large then we choose ρ so that to apply the constrained learning in the most first $T/2$ iterations. If the size of data set is small, then ρ is chosen in order to apply the constraint in early iterations ($\rho \geq 2$). In all experiments, the parameters ε and δ in Eq. (18) are set to 0.83 and $T/4$, respectively. The parameter η in Eq. (5) is set to 1 for all data sets.

Table 2 The brief description of data sets

Data sets	Number of instances	Number of attributes
Fisher's iris plant	150	4
Wine	178	13
TSPLIB1060	1060	2
Image segmentation	2310	19
D15112	15112	2
Gamma telescope	19020	10
NE	50000	2
Pla85900	85900	2

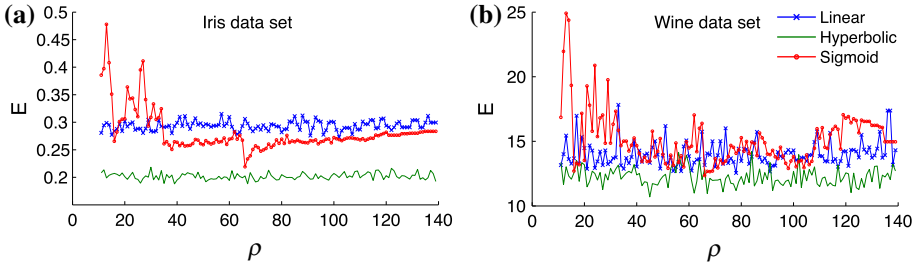

Fig. 4 The sensitivity of choosing different values of parameter $\rho \in \{\rho : 2 \leq \rho \leq 0.7 \times T,\ \rho \in \mathbb{N}\}$ on Iris and Wine data sets. E is the value of quantization error. **a** Iris, **b** Wine

5 Numerical Results

To demonstrate the efficiency of the proposed algorithm, the numerical experiments were carried out using a number of real-world data sets. Algorithm 3 was coded in NetBeans IDE under Java platform and the algorithm is tested on a MAC OSX with 2.7GHz core i7 CPU and 10GB of RAM. Eight data sets: 2 small (Iris and Wine), 2 medium size (TSPLIB1060 and Image Segmentation), 2 large (D15112, Gamma Telescope) and 2 very large (NE[1] and Pla85900) data sets are used in experiments. Iris, Wine, Image Segmentation and Gamma Telescope data sets are data sets with 4, 13, 19 and 10 attributes, respectively, which can be found in UCI Machine Learning Repository [19]. TSPLIB1060, D15112 and Pla85900 are 2 dimensional data sets from TSPLIB library [22] and NE data set from [26]. A brief description of data sets is presented in Table 2.

5.1 Validation of Parameter ρ in Adaptive Constraints

The sensitivity of choosing different values of parameter ρ in (16), (17) and (18) on Iris and Wine data sets is presented in Fig. 4. One can see that this parameter in sigmoid constraint is more sensitive than other two constraint functions in both data sets. Furthermore, in both data sets, the hyperbolic constraint is less sensitive to the parameter ρ and produces high quality maps with less quantization error.

[1] North East.

Table 3 The quantization error, E, using different settings of parameter ρ in CSOM (Linear, Hyperbolic and Sigmoid functions)

Const.	ρ							Ave.	Std.
	10	20	40	60	80	100	140		
	Iris								
Lin.	0.2951	0.3019	0.2990	0.3097	0.2817	0.3004	0.2994	0.30	0.01
Hyp.	0.2106	0.2011	0.1952	0.1944	0.2049	0.2069	0.1925	0.20	0.01
Sigm.	0.7730	0.3062	0.2636	0.2692	0.2693	0.2720	0.2834	0.35	0.19
	Wine								
Lin.	14.5293	14.4811	13.0406	13.8492	13.4721	14.3073	14.3171	14.00	0.57
Hyp.	12.1234	13.1810	12.4003	13.1966	12.2707	12.6803	12.7122	12.65	0.42
Sigm.	54.1000	13.5297	14.6650	14.4896	15.2834	13.8161	14.9614	20.12	15.00
Const.	ρ							Ave.	Std.
	10	40	80	100	140	180	200		
	TSPLIB1060								
Lin.	346.8374	351.0316	345.5140	358.7634	332.5446	333.2026	371.6440	348.51	13.83
Hyp.	468.8084	462.0523	418.3928	401.7283	502.9605	376.1342	371.4968	428.80	50.17
Sigm.	344.4523	347.7434	376.8300	324.9194	328.8889	336.5058	334.9467	342.04	17.30
	Image Segmentation								
Lin.	25.3648	25.4697	25.1439	24.8948	25.0708	24.6638	24.7583	25.05	0.30
Hyp.	22.0328	21.3622	21.7399	20.9122	21.4309	21.9969	21.2374	21.53	0.41
Sigm.	39.7772	26.5647	28.4100	22.8137	26.2363	26.5249	26.1327	28.07	5.42

In order to validate the performance of the constraint parameter ρ, the quantization errors using different settings of this parameter in four data sets, Iris, Wine, TSPLIB1060 and Image Segmentation are presented in Table 3. Furthermore, the average and standard deviation of quantization errors are calculated to check the robustness of this parameter in each constraints. From these results one can see that the CSOM with hyperbolic constraint obtains less average quantization error and sufficiently low deviation in high dimensional data sets, Iris, Wine and Image segmentation than other two constraints. The sigmoid constraint is the one with low average quantization error and deviation in TSPLIB1060 data set. The linear constraint is the second-best constraint, which obtains less deviations in all four data sets.

5.2 Comparison with the SOM

The error values e of the quantization values E using Eq. (1) for different iterations on all data sets are presented in Tables 4 and 5. The error e is calculated using the following formula:

$$e = \frac{E - E_{best}}{E_{best}} \times 100\,\%. \tag{21}$$

In these tables t stands for the CPU time used by an algorithm and the numbers in these tables should be multiplied by the number indicated after the name of each data set to get true values of E_{best}. We include results for different iterations of each to demonstrate their performance. As it is presented in these tables, in early iterations the CSOM is performing

Table 4 Results for small and medium size data sets

τ	E_{best}	SOM		CSOM					
				Linear		Hyperbolic		Sigmoid	
		e	t	e	t	e	t	e	t
	$\times 10^0$	Iris							
1	2.4136	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
5	0.8222	0.00	0.08	0.00	0.01	0.00	0.01	0.00	0.01
10	0.4199	80.52	0.10	80.52	0.02	0.00	0.02	80.52	0.02
50	0.2160	119.03	0.21	119.03	0.12	0.00	0.09	119.03	0.12
100	0.1941	51.16	0.34	46.01	0.23	0.00	0.17	57.50	0.24
200	0.1881	53.06	0.53	43.86	0.43	0.00	0.33	16.59	0.40
	$\times 10^1$	Wine							
1	1.4507	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00
5	0.3349	144.94	0.10	144.94	0.03	0.00	0.02	144.94	0.03
10	0.2605	231.32	0.13	231.32	0.06	0.00	0.04	231.32	0.06
50	0.1343	318.54	0.36	331.72	0.27	0.00	0.16	318.54	0.28
100	0.1093	111.99	0.60	70.72	0.53	0.00	0.29	123.97	0.54
200	0.1082	59.98	1.10	18.02	0.93	0.00	0.55	9.24	0.78
	$\times 10^2$	TSPLIB1060							
1	5.7347	0.00	0.12	0.00	0.04	0.00	0.04	0.00	0.04
5	3.2195	0.00	0.32	15.95	0.31	0.00	0.30	0.00	0.30
15	3.8098	37.06	0.69	0.00	0.96	37.06	0.98	37.06	0.94
75	2.7782	0.00	2.93	8.64	4.93	0.00	4.98	0.00	4.81
150	0.9196	23.26	5.58	0.00	9.75	23.26	9.87	23.26	9.56
300	0.3124	7.23	10.48	1.66	18.53	16.55	18.75	0.00	17.98
	$\times 10^1$	Image Segmentation							
1	1.1162	0.00	0.32	0.00	0.37	0.00	0.36	2.57	0.34
5	0.8451	16.36	1.21	48.89	1.66	0.00	1.55	14.71	1.57
15	0.6216	71.44	3.19	76.54	4.57	0.00	3.77	50.97	4.31
75	0.2999	173.92	14.87	122.67	21.37	0.00	14.51	137.71	21.14
150	0.2152	100.33	29.34	77.83	41.58	0.00	26.92	1.77	33.43
300	0.2051	27.50	56.53	18.77	76.05	0.00	51.06	0.20	57.09

same as the classical SOM until the constraint is applied to the learning process, then the CSOM performs much better than the SOM. From these results one can see that the CSOM outperforms SOM in all data sets and the hyperbolic constraint finds best solution (providing the lowest value of E) on 5 data sets (Iris, Wine, Image Segmentation, Gamma Telescope and NE), sigmoid on 3 data sets (TSPLIB1060, D15112 and Pla85900) and linear is the second-best constraint in 3 data sets (TSPLIB1060, Gamma Telescope and Pla85900).

CSOM finds significantly better solution on 5 data sets (Iris, Wine, TSPLIB1060, Image Segmentation, and Pla85900) than SOM from 6.2 % on Pla85900 up to 37.48 % on Wine data set. In this case, the minimum improvement gained by the CSOM is on D15112 data set where the obtained solution is only 0.57 % better than SOM . In other two data sets (Gamma

Table 5 Results for small and medium size data sets

τ	E_{opt}	SOM		CSOM					
				Linear		Hyperbolic		Sigmoid	
		e	t	e	t	e	t	e	t
	$\times 10^2$	D15112							
1	2.3218	0.00	0.38	0.00	0.53	0.00	0.53	0.00	0.44
5	1.3541	0.00	1.47	0.00	2.39	0.00	2.40	0.00	2.26
15	1.3809	0.00	3.86	0.00	6.76	20.75	5.98	0.00	6.78
75	0.6992	46.57	18.02	46.57	32.92	0.00	24.09	46.57	33.52
150	0.4455	17.53	35.81	17.53	65.82	0.00	46.26	17.53	67.26
300	0.3698	0.57	71.02	1.35	121.22	16.68	90.40	0.00	122.56
	$\times 10^2$	Gamma Telescope							
1	1.2449	0.00	0.67	0.00	1.12	0.00	1.20	0.00	1.21
5	1.3468	0.00	3.01	0.00	5.70	0.00	5.64	0.00	6.10
10	1.3494	0.00	5.78	0.00	11.16	3.50	10.26	0.00	11.80
50	0.7721	35.40	27.09	42.07	56.26	0.00	35.50	35.40	59.44
100	0.4555	62.70	55.89	44.35	109.41	0.00	60.30	45.27	115.24
200	0.3510	5.16	109.68	3.28	196.90	0.00	107.47	0.37	159.77
	$\times 10^{-1}$	NE							
1	2.6821	0.00	0.94	0.00	2.48	0.00	2.49	0.00	2.81
5	0.9135	195.44	4.02	195.44	12.88	0.00	11.68	195.44	14.00
10	0.8027	237.05	7.48	237.05	25.89	0.00	20.50	237.05	28.10
25	0.5573	361.44	17.69	270.02	64.85	0.00	41.76	361.44	72.94
50	0.2725	819.82	34.62	231.30	130.26	0.00	71.59	819.82	147.50
100	0.1515	4.16	68.15	6.14	248.12	9.31	124.55	0.00	263.54
	$\times 10^5$	Pla85900							
1	3.9514	0.00	4.51	0.00	9.44	0.00	8.94	0.00	8.53
5	3.6387	0.00	20.99	0.00	52.89	0.00	50.47	0.00	49.78
10	3.8160	0.00	40.62	0.00	106.15	0.00	100.96	0.00	100.30
15	3.0293	0.00	59.94	0.00	166.88	0.00	156.32	0.00	156.20
20	1.2720	31.16	77.79	0.00	218.68	14.34	199.25	31.16	209.46
30	0.1747	6.64	114.26	0.11	298.67	98.17	274.29	0.00	289.62

Telescope and NE) the CSOM reduces the value of the quantization error E about 5 % on the rest of the data sets. Note that in the most data sets CPU time required by the Constrained SOM is slightly larger than that of by the SOM. This is due to the fact that the CSOM tries to find nearest neighborhood neurons in n-dimensional space. Since both the NE and Pla85900 are large data sets the SOM algorithm requires more iterations than in other smaller data sets to spread neurons over the whole data set. Therefore, in these data sets constraints are applied at iterations close to $T/2$ to ensure that values calculated by constraint functions (16)–(18) are true values.

To see the performance of proposed algorithm, In Fig. 5 we present values of E obtained by the SOM and the CSOM depending on iteration τ in Iris, Wine, Image Segmentation and

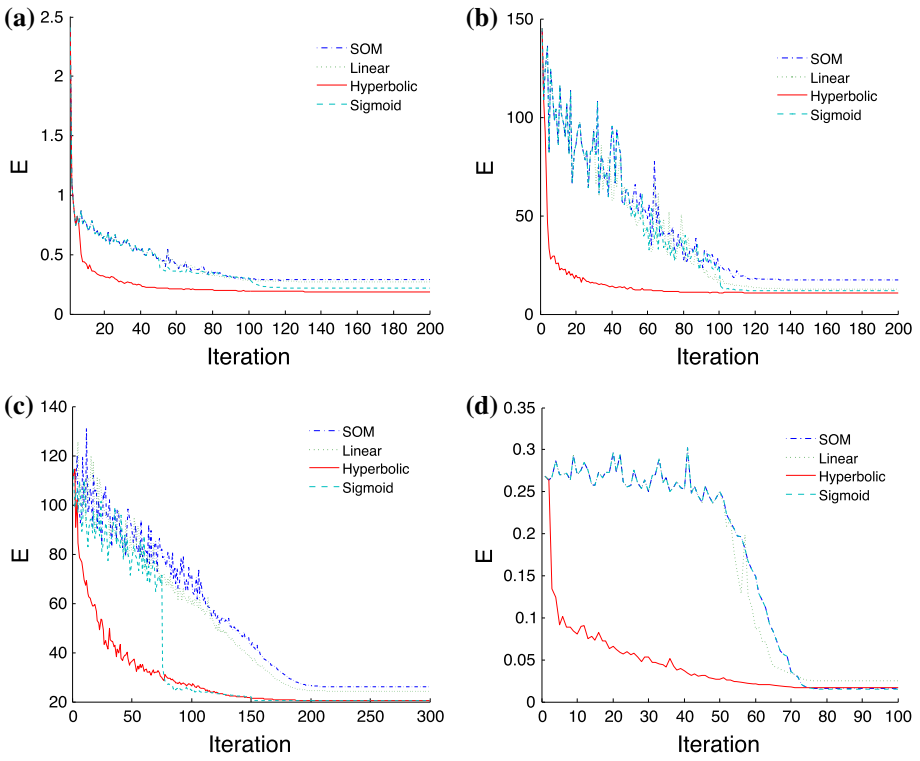

Fig. 5 Convergence of CSOM versus SOM. **a** Iris, **b** wine, **c** image segmentation, **d** NE

NE data sets. From these figures, it is obvious that both the SOM and the CSOM converge as $\tau \rightarrow T$. Comparing values of E in all data sets, one can see that the CSOM converges significantly faster than the SOM in iterations less than 20. The CSOM with the hyperbolic constraint produces best results in all 4 data sets.

5.3 Complexity Comparison with SOM

The time complexity of Self Organizing Map is linear with respect to number of data points but it is dependent quadratically on the number of the neighborhood neurons to be updated by the winner neuron. The most time consuming step in the SOM algorithm is Step 2 and Step 3 where the Eq. (3) is calculated. In Table 6, the total number of calculations of (3), $N = T\rho n$, for all data sets is presented. From these results, one can see that the number N obtained by CSOM is considerably less than those with the classical SOM in all data sets. The complexity is improved significantly using the proposed algorithm in 4 data sets: Iris, Image Segmentation, D15112 and NE.

5.4 Distortion Error

Note that the error E shows the quantization quality of the network. However, there is a distortion measurement which can be used to calculate the overall quality of the map. Unlike the quantization error, the distortion measure ξ considers both vector quantization and topology preservation of the SOM. The distortion measure is defined as follows [1]:

Table 6 Total number of calculations of (3), N, for all data sets

τ	SOM	CSOM			τ	SOM	CSOM		
		Linear	Hyperbolic	Sigmoid			Linear	Hyperbolic	Sigmoid
	Iris ($\times 10^4$)					Wine ($\times 10^4$)			
1	1.05	1.05	1.05	1.05	1	4.72	4.72	4.72	4.72
5	5.34	5.34	5.34	5.34	5	22.4	22.4	18.1	22.4
10	10.5	10.5	9.05	10.5	10	44.4	44.4	28.1	44.4
50	51.6	51.6	18.6	51.6	50	220	212	68.4	220
100	103	99.0	23.1	92.8	100	437	392	93.7	407
200	205	159	31.0	96.2	200	872	611	135	421
	TSP.1060 ($\times 10^5$)					Image Seg. ($\times 10^6$)			
1	0.81	0.81	0.81	0.81	1	1.59	1.59	1.59	1.29
5	3.82	3.85	3.82	3.82	5	7.82	7.83	6.59	6.72
15	11.3	10.9	11.3	11.3	15	23.6	23.3	13.5	20.6
75	56.1	50.3	56.1	56.1	75	118	111	26.1	105
150	112	93.0	112	112	150	235	214	33.0	110
300	223	143	123	141	300	462	361	42.9	110
	D15112 ($\times 10^6$)					Gamma Tel. ($\times 10^6$)			
1	0.62	0.62	0.62	0.620	1	3.71	3.71	3.71	3.71
5	3.09	3.09	3.09	3.09	5	18.6	18.6	18.6	18.6
15	9.26	9.26	6.24	9.26	10	37.1	37.1	32.3	37.1
75	46.5	46.5	11.7	46.5	50	187	178	70.9	187
150	93.4	93.4	15.1	93.4	100	376	338	89.3	353
300	187	146	21.1	141	200	756	558	120	372
	NE ($\times 10^6$)					Pla85900 ($\times 10^6$)			
1	1.94	1.94	1.94	1.94	1	6.90	6.90	6.90	6.90
5	9.68	9.68	7.78	9.68	5	34.5	34.5	34.5	34.5
10	19.4	19.4	12.7	19.4	10	68.9	68.9	68.9	68.9
25	48.6	48.6	21.3	48.6	15	101	101	101	101
50	97.9	97.9	29.4	97.9	20	130	127	120	130
100	196	147	40.6	170	30	200	164	135	175

$$\xi = \sum_{x_i \in A} \sum_{w_j \in \Psi} h_{cj} \|x_i - w_j\|^2, \quad j \neq c, \tag{22}$$

where c is the BMU of x_i and h_{cj} is the neighborhood function of neurons c and j at max iteration T defined by Eq. (4).

Table 7 presents the numerical results of distortion measure (22) on all data sets. From these results, one can see that the proposed algorithm outperforms the SOM in all data sets. The improvement of distortion error, ξ, is significance in eight data sets (wine, TSPLIB1060, image segmentation, TSPLIB3038, D15112, gamma telescope, NE and Pla85900). The value n_d in the Table 7 presents the number of neurons which never activated (dead neurons) by the input data points. The number of dead neurons, n_d, for CSOM in all data sets is equal or

Table 7 Results of distortion measure on all data sets

Dataset		SOM		CSOM					
				Linear		Hyperbolic		Sigmoid	
		ξ	n_d	ξ	n_d	ξ	n_d	ξ	n_d
Iris	$\xi \times 10^{-3}$	1.75	33	0.47	29	1.18	23	1.37	27
Wine	$\xi \times 10^{-3}$	1.85	24	0.29	17	0.25	24	0.13	22
TSP1060	$\xi \times 10^{4}$	1.32	16	0.82	7	0.99	20	0.57	10
Image Seg.	$\xi \times 10^{-4}$	7.13	9	0.74	6	1.01	13	0.74	4
D15112	$\xi \times 10^{3}$	4.04	0	0.49	0	20.50	49	0.44	1
Gamma Tel.	$\xi \times 10^{-9}$	4.85	0	0.02	1	0.12	0	0.03	0
NE	$\xi \times 10^{-6}$	1.09	19	0.37	1	0.19	7	0.01	9
Pla85900	$\xi \times 10^{5}$	16.30	0	0.10	0	0.26	4	0.08	0

less than that for the SOM. This means that the proposed algorithm activates more neurons and distributes the neurons more efficiently than the classical SOM.

5.5 Topographic Product

In this section the neighborhood relations preservation of CSOM is compared with the SOM using topographic product. This measure demonstrate how nearby neurons in the input space are mapped onto neighboring locations in the output space. Many topology preservation measures have been proposed and a complete review can be found in [2,28].

The topographic product is a measure of the preservation of neighborhood relations in maps between spaces. The basic idea is that for each neuron the ordering of remaining neurons in output and input spaces are defined based on their distances in each space. Then the error is minimized if for each neuron these orderings become identical, i.e the k-th nearest neuron in output space is the k-th nearest neuron in the input space.

The notation for nearest-neighbor indices is defined as $n_k^A(j)$, which denotes the k-th nearest neighbor of node j, with the distances measured in the output space:

$$n_k^A(j) : d^A\left(j, n_k^A(j)\right) = \min_{j' \in A / \left\{ j, n_1^A(j), \ldots, n_{k-1}^A(j) \right\}} d(j, j').$$

In the same way let $n_k^V(j)$ denote the k-th nearest neighbor of j, but with the distances measured in the input space.

$$n_k^V(j) : d^V(w_j, w_{n_k^V(j)}) = \min_{j' \in A / \left\{ j, n_1^V(j), \ldots, n_{k-1}^V(j) \right\}} d(w_j, w_{j'}).$$

Using nearest neighborhood indexation the following ratios are defined:

$$Q_1(j,k) = \frac{d^V(w_j, w_{n_k^A(j)})}{d^V(w_j, w_{n_k^V(j)})}, \quad Q_2(j,k) = \frac{d^A\left(j, n_k^A(j)\right)}{d^A\left(j, n_k^V(j)\right)}.$$

Table 8 Results of topographic product on all data sets

Dataset	CSOM			
	SOM	Linear	Hyperbolic	Sigmoid
Fisher's iris plant	-5.48×10^{-03}	-7.78×10^{-03}	5.21×10^{-02}	-4.01×10^{-03}
Wine	-2.55×10^{-03}	2.02×10^{-04}	4.22×10^{-02}	3.09×10^{-03}
TSPLIB1060	-3.09×10^{-03}	1.79×10^{-03}	1.25×10^{-01}	9.93×10^{-04}
Image seg.	-2.40×10^{-03}	2.09×10^{-03}	1.74×10^{-02}	5.72×10^{-04}
D15112	1.78×10^{-04}	1.05×10^{-04}	1.23×10^{-04}	8.31×10^{-05}
Gamma tel.	-8.92×10^{-03}	5.27×10^{-03}	1.94×10^{-02}	1.68×10^{-02}
NE	-1.38×10^{-03}	9.15×10^{-03}	1.21×10^{-01}	8.33×10^{-04}
Pla85900	-8.73×10^{-03}	-3.33×10^{-03}	6.71×10^{-03}	-7.04×10^{-03}

The topographic product formula is defined as:

$$P = \frac{1}{N(N-1)} \sum_{j=1}^{N} \sum_{k=1}^{N-1} \log(P_3(j,k)), \tag{23}$$

where

$$P_3(j,k) = \left(\prod_{l=1}^{k} Q_1(j,l) Q_2(j,l) \right)^{1/2k}.$$

The topographic product of SOM and CSOM on all data sets are presented in Table 8. From (23) one can see that the desired value of P is 0. The CSOM algorithm outperforms SOM algorithm on all cases. The improvement of topographic product value is significant in Wine, TSPLIB1060, Image Segmentation, D15112 and NE.

5.6 Topology Preservation

In this section we present the topology preservation of the proposed algorithm. We choose 2-dimensional data sets in order to be displayed easily. Figures 6 and 8 show the topology of the SOM and CSOM for two data sets: TSPLIB1060, NE. It should be noted that only active neurons are presented in these figures. In Fig. 6a one can see that some of the SOM's neurons for the TSPLIB1060 data set are far from the mapped data points which leads to increments in the quantization error E. These is due to the absorption of these neurons by their neighborhoods which, in fact, are far from these neurons in the n-dimensional space. On the contrary one can see from Fig. 6b that the CSOM spreads the neurons over the mapped data more accurately by overcoming the above mentioned deficiency in the SOM. These improvements can be obviously seen by comparing Figs. 6a and 6b at the bottom-left and right of the maps where exists a gap between two groups of data points. The visualization of neurons on D15112 data set shows similar results. The CSOM converge to the domain of data points more accurately. This is obviously proved by comparing the top-right side of Figs. 7a and b. Figure 8 presents the topology of the SOM and CSOM on NE data set. The CSOM in Fig. 8b is more accurate than the SOM in Fig. 8b where some active neurons are

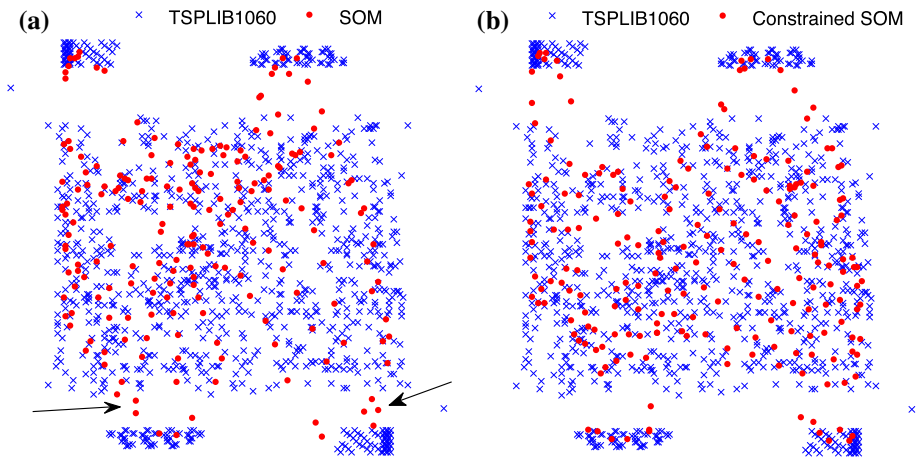

Fig. 6 Topology preservation of CSOM versus SOM. **a** SOM on TSPLIB1060, **b** CSOM on TSPLIB1060

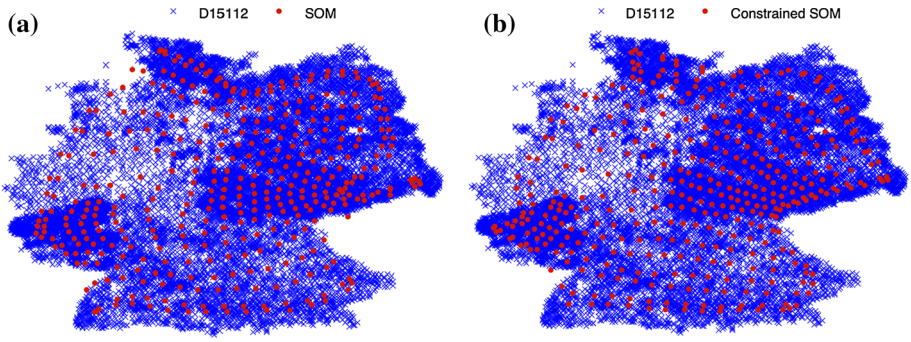

Fig. 7 Topology preservation of CSOM versus SOM. **a** SOM on D15112, **b** CSOM on D15112

located in the space outside of the region where the data set is located (they are displayed by an arrow).

5.7 Comparison with Other Algorithms

In this section the CSOM is compared to similar topology preservation algorithms in the sense of accuracy and computational time. The CSOM is compared with Growing Grid (GG) [3], Growing Neural Gas (GNG) [8], Growing Hierarchical SOM (GHSOM) [21], Principal Component Analysis (PCA) [24], Sammon's Mapping [23], Fuzzy Sammon Mapping [16] and RPSOM [5] algorithms. The max number of neurons and iteration for GG, GNG and GHSOM algorithms are set to 400 and 4000, respectively. The same settings as presented in Table 1 are considered for RPSOM algorithm.

The value e using 21 and the best value of quantization error, $\mathbf{E_{best}}$, are presented in Table 9. Note that in this table LCSOM, HCSOM and SCSOM represent Linear CSOM, Hyperbolic CSOM and Sigmoid CSOM, respectively. In all data sets, except Gamma Telescope, the CSOM produces less quantization error than other algorithms. The GNG algorithm obtains less error on Gamma Telescope data set in comparison with other algorithms. From the

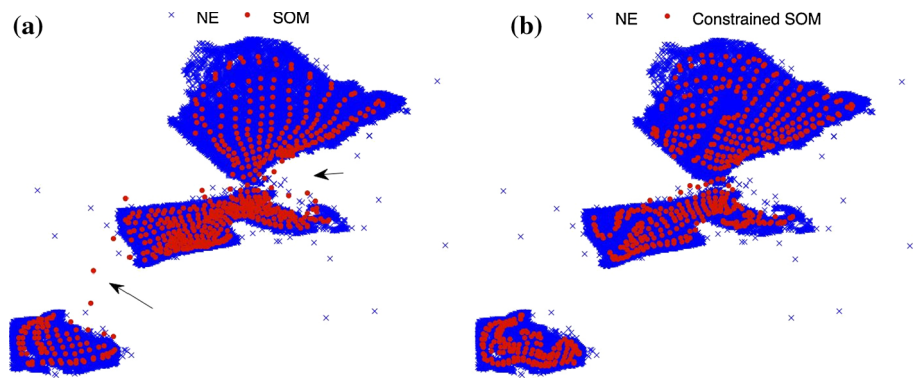

Fig. 8 Topology preservation of CSOM versus SOM. **a** SOM on NE, **b** CSOM on NE

Table 9 Results of quantization error on all data sets

Alg.	Iris	Wine	TSPLIB1060	Image Seg.	D15112	Gamma Tel.	*NE Pla*85900	
	E_{best}							
	$\times 10^{-1}$	$\times 10^{1}$	$\times 10^{2}$	$\times 10^{1}$	$\times 10^{2}$	$\times 10^{1}$	$\times 10^{-2}$	$\times 10^{4}$
	1.88	1.08	3.12	2.05	3.70	3.14	1.51	1.75
	e							
GG	45.35	66.34	2.73	29.59	32.24	5.93	51.04	1452.03
GNG	35.42	34.75	20.36	2.94	4.50	**0.00**	13.73	249.67
GHSOM	30.23	45.62	47.37	157.50	123.25	109.72	–	–
PCA	5.15	16.09	14.47	29.09	4.25	3.23	–	–
Sammon	5.79	16.15	14.47	29.10	4.25	3.54	–	–
FuzSam	39.17	23.67	13.88	59.85	3.75	42.24	–	–
RPSOM	12.01	11.91	8.59	15.72	2.69	14.57	3.00	2.12
LCSOM	43.84	18.03	1.66	18.78	1.37	15.27	6.19	0.10
HCSOM	**0.00**	**0.00**	16.56	**0.00**	16.71	11.61	9.34	98.14
SCSOM	16.58	9.22	**0.00**	0.21	**0.00**	12.04	**0.00**	**0.00**

results presented in the Table 9, one can see that the PCA is the second best algorithm in Iris, D15112 and Gamma Telescope data sets. Moreover, the RPSOM is the second best algorithm in Wine, NE and Pla85900 data sets. The results obtained by GG and GNG algorithms are close to those obtained by CSOM in TSPLIB1060 and Image Segmentation data sets, respectively. The RPSOM algorithm generates satisfactory results on NE and Pla85900 data sets in comparison with CSOM. Note that the PCA, Sammon's Mapping and Fuzzy Sammon algorithms failed in two very large data sets, NE and Pla85900, due to high computational time requirement (more than five hours).

The CPU time required by all algorithms are presented in the Table 10. The CSOM is the best algorithm in Iris, Wine, NE and Pla85900 data sets in the sense of both accuracy and the required cpu time. The PCA and GNG algorithms are fast in most of the data sets, but the generated errors by these algorithms are quit far from the satisfactory results. From these results one can see that the CSOM outperforms the RPSOM algorithm in all cases.

Table 10 The CPU time required by all algortihms

Alg.	Iris	Wine	TSPLIB1060	Image Seg.	D15112	Gamma Tel.	NE	Pla85900
	t							
GG	4.40	4.31	10.67	30.70	163.63	297.60	166.91	622.70
GNG	3.16	4.48	6.20	13.90	26.85	102.10	97.76	115.82
GHSOM	2.06	2.87	62.88	303.68	10848.36	18363.03	-	-
PCA	0.04	0.05	0.31	2.31	34.07	131.72	-	-
Sammon	3.81	4.36	40.35	194.72	6620.42	9386.41	-	-
FuzSam	2.43	3.01	24.64	25.26	144.93	196.65	-	-
RPSOM	6.29	7.58	55.57	130.11	804.15	726.64	968.34	445.05
LCSOM	0.42	0.87	17.89	92.19	128.00	211.62	236.54	338.24
HCSOM	0.33	0.55	18.74	73.62	96.05	127.55	156.08	325.50
SCSOM	0.37	0.73	19.78	68.10	123.20	140.72	253.70	315.96

6 Conclusion

In this paper, we develop a modified learning algorithm for the Self Organizing Maps. The aim is to propose a learning algorithm which restricts the neighborhood adaptations to only those neurons that are not far from the best matching unit in the n-dimensional space. Therefore, we introduced an adaptive constraint parameter. This parameter is a decreasing function with respect to iterations to be applicable to the SOM learning process. The adaptive constraint parameter selected as linear, hyperbolic and sigmoid function. The experiments on eight real-world data sets demonstrate the superiority of the proposed algorithm over the SOM in the sense of accuracy. The results show that the CSOM converges much faster than SOM and in all cases with less computational complexity, moreover, improves the topology preservation and presents promising clustering results. The proposed algorithm outperforms similar topology preservation algorithms specially in very large data sets in the sense of accuracy and computational time.

The CSOM is designed for offline training of large data sets however, it can be modified to be suitable in realtime environments. The constrained learning algorithm updates a set of neurons based on some parameters which can be saved in the memory for the next upcoming data points. The modification of the CSOM for realtime training is a part of future work.

References

1. Arous N (2010) On the search of organization measures for a kohonen map case study: speech signal recognition. Int J Digit Content Technol Appl 4(3):75–84
2. Bauer HU, Pawelzik K (1992) Quantifying the neighborhood preservation of self-organizing feature maps. IEEE Trans Neural Netw 3(4):570–579
3. Bernd F (1995) Growing grid—a self-organizing network with constant neighborhood range and adaptation strength. Neural Process Lett 2:9–13
4. Chen N, Ribeiro B, Vieira A, Chen A (2013) Clustering and visualization of bankruptcy trajectory using self-organizing map. Exp Syst Appl 40(1):385–393
5. Cheung YM, Law LT (2007) Rival-model penalized self-organizing map. IEEE Trans Neural Netw 18(1):289–295

6. Fiannaca A, Fatta G, Gaglio S, Rizzo R, Urso A (2007) Improved som learning using simulated annealing. In: Artificial Neural Networks AI ICANN 2007, Lecture Notes in Computer Science, vol 4668., Springer, Berlin Heidelberg, pp 279–288
7. Fiannaca A, Di Fatta G, Rizzo R, Urso A, Gaglio S (2011) Simulated annealing technique for fast learning of som networks. Neural Comput Appl 22(5):1–11
8. Fritzke B (1995) A growing neural gas network learns topologies. Advances in neural information processing systems, vol 7. MIT Press, Cambridge, pp 625–632
9. Goncalves ML, de Andrade Netto ML, Zullo J (1998) A neural architecture for the classification of remote sensing imagery with advanced learning algorithms. In: Neural networks for signal processing vol VIII, Proceedings of the 1998 IEEE signal processing society workshop, pp 577–586
10. Gorricha J, Lobo V (2012) Improvements on the visualization of clusters in geo-referenced data using self-organizing maps. Comput Geosci 43:177–186
11. Haese K (1998) Self-organizing feature maps with self-adjusting learning parameters. IEEE Trans Neural Netw 9(6):1270–1280
12. Hsu A, Halgamuge SK (2008) Class structure visualization with semi-supervised growing self-organizing maps. Neurocomputing 71(16–18):3124–3130
13. Jin H, Shum WH, Leung KS, Wong ML (2004) Expanding self-organizing map for data visualization and cluster analysis. Inf Sci 163(1–3):157–173
14. Johnson R, Wichern D (2002) Applied multivariate statistical analysis. Prentice Hall, Upper Saddle River
15. Kohonen T (2001) Self-organizing maps. Springer series in information sciences. Springer, Berlin
16. Kovács A, Abonyi J (2002) Vizualization of fuzzy clustering results by modified sammon mapping. In: Proceedings of the 3rd international symposium of hungarian researchers on computational intelligence, pp 177–188
17. López-Rubio E (2013) Improving the quality of self-organizing maps by self-intersection avoidance. IEEE Trans Neural Netw Learn Syst 24(8):1253–1265. http://ieeexplore.ieee.org
18. Martinetz T, Schulten K (1991) A "Neural-Gas" network learns topologies. Artif Neural Netw I:397–402
19. Murphy (2008) UCI machine learning repository. http://aiweb.techfak.uni-bielefeld.de/~content/bworld-robot-control-software/
20. Nikkilä J, Törönen P, Kaski S, Venna J, Castrén E, Wong G (2002) Analysis and visualization of gene expression data using self-organizing maps. Neural Netw 15(8–9):953–966
21. Rauber A, Merkl D, Dittenbach M (2002) The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. IEEE Trans Neural Netw 13(6):1331–1341
22. Reinelt G (1991) TSPLIB—a traveling salesman problem library. ORSA J Comput 3(4):376–384
23. Sammon J (1969) A nonlinear mapping for data structure analysis. IEEE Trans Comput C–18(5):401–409
24. Scholz M, Kaplan F, Guy CL, Kopka J, Selbig J (2005) Non-linear pca: a missing data approach. Bioinformatics 21(20):3887–3895
25. Sirola M, Talonen J (2011) Self-organizing map in process visualization. In: Koenig A et al (eds) Knowlege-based and intelligent information and engineering systems. Springer, New York, pp 196–202
26. Theodoridis Y (1996) Spatial datasetsan unofficial collection. http://www.dias.cti.gr/~ytheod/research/datasets/spatial.html
27. Vesanto J, Himberg J, Alhoniemi E, Parhankangas J (2000) Self-organizing map in matlab: the som toolbox. In: Proceedings of the matlab DSP conference, pp 35–40
28. Vidaurre D, Muruzabal J (2007) A quick assessment of topology preservation for som structures. IEEE Trans Neural Netw 18(5):1524–1528
29. Wang H, Zheng H, Azuaje F (2007) Poisson-based self-organizing feature maps and hierarchical clustering for serial analysis of gene expression data. IEEE/ACM Trans Comput Biol Bioinform 4(2):163–175
30. Wu S, Chow TWS (2005) PRSOM: a new visualization method by hybridizing multidimensional scaling and self-organizing map. IEEE Trans Neural Netw 16(6):1362–1380
31. Yang L, Ouyang Z, Shi Y (2012) A modified clustering method based on self-organizing maps and its applications. Proced Comput Sci 9:1371–1379
32. Yen GG, Wu Z (2008) Ranked centroid projection: a data visualization approach with self-organizing maps. IEEE Trans Neural Netw 19(2):245–259
33. Yin H (2002) ViSOM—a novel method for multivariate data projection and structure visualization. IEEE Trans Neural Netw 13(1):237–243