Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Evaluating Clustering Performances of FLSOM Algorithm

A. Fiannaca, G. Di Fatta, S. Gaglio, R. Rizzo, A. Urso

Rapporto Tecnico N.:
RT-ICAR-PA-07-03 **settembre 2007**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Evaluating Clustering Performances of FLSOM Algorithm

A. Fiannaca [1], G. Di Fatta[1,2], S. Gaglio[1,3], R. Rizzo[1], A. Urso[1]

Rapporto Tecnico N.:
RT-ICAR-PA-07-03

Data:
settembre 2007

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo Viale delle Scienze edificio 11 90128 Palermo
[2] School of Systems Engineering University of Reading, UK.
[3] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica Viale delle Scienze 90128 Palermo

Evaluating Clustering Performances of FLSOM Algorithm

Antonino Fiannaca[1], Giuseppe Di Fatta[1,2], Salvatore Gaglio[1,3], Riccardo Rizzo[1], and Alfonso M. Urso[1]

[1] ICAR-CNR, Consiglio Nazionale delle Ricerche, Palermo, Italy.
[2] School of Systems Engineering, University of Reading, UK.
[3] Dipartimento di Ingegneria Informatica, Universitá di Palermo, Italy.

Abstract. The Self-Organizing Map (SOM) is a popular unsupervised neural network able to provide effective clustering and data visualization for multidimensional input spaces. In this paper we present an application of the simulated annealing procedure to the SOM learning algorithm, called fast learning SOM ($FLSOM$). The goal of the procedure is to obtain a fast learning and a better performance in terms of QE without change the simplicity of the basic algorithm that is one of the strongest point of SOM. The proposed approach is compared to the original SOM and with some of its modification and speed-up techniques with good results. Moreover we show that FLSOM also improves the quality of the map by providing better clustering quality and topology preservation of input multi-dimensional data. Several tests have been carried out on different multidimensional datasets, which demonstrate the superiority of the algorithm in comparison with the original SOM.

Key words: SOM, Simulated Annealing, Clustering

1 Introduction

The Self-Organizing Map (SOM) algorithm is used to build a mapping from an high dimensional data space to a low-dimensional representation space. A specific characteristic, distinguishing SOM from other data mining techniques, is the neighborhood preservation of lattice map. Nowadays, datasets used in data mining become larger time after time and a fast analysis is required. Typically, techniques used for speeding up SOM algorithm, work in two different ways: modifying the characteristic of neurons to adjust influence of input in the lattice map [1] [2] or modifying the standard algorithm to address the variation of learning parameters [3] [4]. The first approach acts on lower abstraction layer than second ones, because they redefine the neurons, the basic elements of SOM network; otherwise in second one changes only the learning rule depending on network evolution. This work deals with the realization of an application for unsupervised clustering of high dimensional datasets using a technique aiming not to alter simplicity and functionality of classic SOM algorithm: in this point

of view, the proposed approach works at the higher abstraction level. Our solution, during training phase, introduces an optimization technique, the simulated annealing (SA), to drive the system toward the global minimum.

Then we have carried out studies both on the shapes and on the ideal values that learning rate factor α could perform during the training phase. However the variation of learning rate factor can not satisfy previous goal, thus an optimization technique, driving the system to final optimal solution, is been used. Notice that this technique also must both speed up the learning and work in unsupervised environment. The simulated annealing [5] [19] could be an attempt to find global optimum with respect to above-mentioned constraints. This technique is particularly flexible, indeed it is exploited to solve several applications in many fields. SA has been also adopted in SOM to improve the detection of winning neurons [6] or to select a representative pattern for each update during training phase [7]. In [6] the SA, replaced by a deterministic implementation (Deterministic Annealing), does not lead to an improvement in terms of computational complexity but just with regard to quality of learning; that happens because the selection of winning neurons, solved by Kohonen in simplest way, here is steered by a minimization of a cost function where the SA is strongly used. Instead, in [7], learning process requires nearly the same execution time than standard SOM and, moreover, the evolution of training should avoid local minima even though the learning could advantage a few patterns.

In this work we introduce an algorithm that preserves classic SOM implementation in term of simplicity and functionality, adding SA as evaluation and directional criterion for the map evolution. For this reason the basic ideas of SA has been used. The proposed algorithm, called Fast Learning SOM ($FLSOM$) is faster than the regular SOM and is compared with SOM, PLSOM, HabSOM and ConsSOM.

In addition, we demostrate that the FLSOM algorithm is able to perform also a good clustering. We evaluate and compare the algorithm with a standard SOM over a number of artificial and real datasets. The experimental analysis shows that FLSOM provides better results in terms of both topology preservation and clustering criteria.

The paper has the following structure: the next section describes some related works aiming to speed up learning of SOM; the section 3 reports the basic SOM algorithm and some details of the other algorithms used for comparison; the section 4 shows the advantage of using proposed algorithm for clustering process; the section 5 reports both the evaluation criteria and the experimental results for introduced algorithm. the section 6 reports the experimental results for the clustering application. Finally some conclusions are reported in section 7.

2 Related Work

Many different mechanisms have been proposed in the literature for improving SOM performances. The key issue is to determine which parameters need to be considered, in order to obtain a SOM that both achieves a clustering in a short

time and creates a data projection strongly related to the distribution of data in the input space. One such attempt was done in auto-SOM [4], introducing a complex algorithm, based on Kalman filters coupled with a recursive parameter estimation method depending on [8] [9], to guide the weight vectors toward the center of their respective Voronoi cells in input spaces. Using this algorithm, it is possible to automatically estimate the learning parameters during the training of SOMs. Indeed the introduction of Kalman filters leads the network to a good training, but it is more computationally expensive than the classic SOM. Notice that in [4] authors confirm that the incremental learning SOM algorithm is faster than the auto-SOM, because the former needs fewer learning steps than the latter during training process. Moreover, they also assert that standard techniques often get stuck in local minima. A recent improvement to the SOM original algorithm is the parameterless SOM (PLSOM) [3] [10]: this technique is based on the standard SOM algorithm and removes both classic learning rate function and neighborhood size function. Usually these functions are decreased over time and do not take into account the evolution of the network during learning process. PLSOM evaluates the adaptation of input stimuli and calculates the learning factor and neighborhood size depending on the local quadratic fitting error of the map to the input space. Unfortunately only the local error is used during the evolution of the map; this means that both the first data inputs and the initialization of the map, play a key role in map evolution and, moreover, the selection of the learning rate is modified without evaluating weights in most of neurons. An adaptation of habituation mechanism in SOMs was proposed in [1] and results was compared to conscience algorithm [2]. These algorithms are based mainly on the identification of neurons that win too frequently and then on the introduction of an handicap for these neurons. This way, each neuron is selected with almost the same probability of the other ones. In this context, the habituation mechanism is more flexible than conscience-learning because it can be used to manage the learning processing a fine grain way. Otherwise, the conscience mechanism speed up learning process using an *a priori* knowledge to estimate a value of probability in order to catch winning neuron. The [1], compared to standard SOM implementation, is not more computationally expensive, although the algorithm adds an habituation parameter. This parameter is a function of a local error so that the habituation is increased when the network is not ordered. The introduction of the habituation causes the deceleration of the training during the refinement phase of the learning. In the present work we introduce a variation of the standard SOM that does not reduce speed in the last epochs of learning, according to clustering requirements; moreover we use a global quantization error rather than a local quadratic fitting error to calculate an adaptive learning rate depending both on time and on organization of neurons in the map.

3 SOM Adaptive Learning Rate Algorithm

In this section the learning algorithm of the FLSOM is introduced. Although the SOM learning procedure is well known it is necessary to briefly highlight some points of the original algorithm to better understand the proposed one.

3.1 Self-Organizing Map basic algorithm

Self-Organizing Maps [11] are neural structures capable of building maps of the input data, which preserve neighborhood relationships. Although the SOM algorithm is well known, it is necessary to report some formulas in order to better understand the modified algorithms. The incremental learning algorithm [12] trained for epochs is used; this version of the learning algorithm is reported in table 1.

The main formulas of the standard SOM are reported below. In (step 3.(a).ii) of Table 1 the winner unit, called best matching unit (bmu) is selected according to:

$$bmu = \arg\left(\min_{i \in N} \|x - w_i\|\right). \tag{1}$$

In (step 3.(a).iii) neural weights are updated using the following rule:

$$w_i(t+1) = w_i(t) + \alpha(t) h_{ci}(t) \left[x - w_i(t)\right], \tag{2}$$

where h_{ci} is the neighborhood kernel around the best matching unit. One of the most common shapes of the kernel is the gaussian shape:

$$h_{ci}(t) = \exp\left(\frac{d\left(r_{bmu}, r_i\right)}{2\sigma^2(t)}\right), \tag{3}$$

where the term $d\left(r_{bmu}, r_i\right)$ stands for the distance between the bmu unit and the generic unit i on the SOM lattice.

The learning parameter $\alpha(t)$ and the neighborhood radius $\sigma(t)$ are decreasing functions of time of the same kind and follow the same law.

$$\alpha(t) = \alpha_{MAX} \left(\frac{\alpha_{MIN}}{\alpha_{MAX}}\right)^{\left(\frac{t}{t_{max}}\right)}, \tag{4}$$

$$\sigma(t) = \sigma_{MAX} \left(\frac{\sigma_{MIN}}{\sigma_{MAX}}\right)^{\left(\frac{t}{t_{max}}\right)}. \tag{5}$$

3.2 Some modified Self Organizing Map algorithms

The parameterless SOM (PLSOM) [3] replaces learning rate function (eq. 4) with a scaling variable, $\epsilon(t)$, depending on local error, defined by:

$$\epsilon(t) = \frac{\|x(t) - w_c(t)\|}{r(t)}, \tag{6}$$

where $w_c(x)$ is the closest weight vector to input vector x at time t and $r(t)$ is given by:

$$r(t) = max(\|x(t) - w_c(t)\|, r(t-1)). \tag{7}$$

The PLSOM also modifies classic neighborhood size function (eq. 5) with an expression depending on $\epsilon(t)$, defined by:

$$\sigma(t) = (\sigma_{MAX} - \sigma_{MIN}) \cdot \epsilon(t) + \sigma_{MIN}. \tag{8}$$

The habitation SOM (HabSOM) introduces a new parameter, the habituation a_i, in neurons; a reversible decrement of the neural response to a repetitive stimulus. The amount of the increment of the habituation variable is a function of the activation s of the neuron i, defined in:

$$s_i = \exp\left(\frac{\|x - w_i\|}{\tau}\right), \tag{9}$$

The τ parameter decreases the activation value when the input is far from the *bmu*. The value of the habituation variable for the neuron i over the time is given by:

$$a_i(s_i, t) = a_i(t-1) + a_0 \cdot s_i. \tag{10}$$

This new variable changes rule of *bmu* selection, so that if the habituation is below a threshold t_r, then the neuron catches the input pattern according to eq. 1 and learns; otherwise if the habituation is above a threshold t_r, then the input is not connected to the neuron or, in other words, neuron can not compete to *bmu* selection.

The conscience SOM (ConsSOM) is mainly focused on avoiding dead-units and, to achieve this goal, its algorithm modifies the equation 2 to select winning neuron. New equation is given by:

$$\|x - w_{bmu}\|^2 - b_{bmu} \leq \|x - w_i\|^2 - b_i \quad \forall i = 1, 2, .., N \tag{11}$$

where the term b_i is defined as:

$$b_i = C\left[\frac{1}{N} - p_i\right] \tag{12}$$

In this equation, C is the bias factor that controls the "amount" of conscience of the neuron. The term p_i represents the fraction of times the unit i wins the competition; it is modified during the learning phase according to the formula $p_i = p_i + B(y_i - p_i)$ where y_i is 1 if i is the winning neuron and 0 otherwise.

Table 1. SOM incremental learning algorithm.

1. Initialize the N neurons with random weights.
2. Set step counter $t = 1$, epoch counter $p = 1$, maximum number of epochs $MaxP$.
3. While the stop condition ($p \geq MaxP$) is not verified
   (a) Define $RndDataset$ as a list where input patterns are randomly ordered.
   (b) While $RndDataset$ is not empty
      i. Get a pattern $x(t)$ from $RndDataset$.
      ii. Find the best matching unit $bmu(x(t)) = \arg\left(\min_{i \in N} \|x(t) - w_i\|\right)$.
      iii. Update weights of bmu and its neighborhood with:
      $w_i(t+1) = w_i(t) + \alpha(t) h_{bmu,i}(t)\,[x - w_i(t)]$,
      where $h_{bmu,i}$ is the neighborhood gaussian function, kernel around the bmu.
      iv. Remove $x(t)$ from $RndDataset$.
      v. $t = t + 1$.
      vi. Update the value of learning rate $\alpha(t)$ and of neighbourhood radius $\sigma(t)$, used in the neighbourhood gaussian function h_{bmu}.
   (c) $p = p + 1$, $t = 1$.
4. End of learning after $MaxP$ epochs.

3.3 The Fast Learning SOM algorithm (FLSOM)

In this section, we present the FLSOM algorithm with the training function depending both on simulated annealing optimization and on global evolution of the neural network.

Simulated annealing (SA) is an optimization method typically used for large scale problems, especially the ones where a global minimum is hidden by many local minima. In the present work, we use this heuristic to improve the quality of learning process of the SOM, preserving its unsupervised characteristic. The adopted approach provides an adaptive learning rate factor $\alpha(t, QE)$, steered by the simulated annealing heuristic over the current resolution of the map. Using a SOM trained for epochs, at the end of each learning epoch, the Quantization Error (QE) can be identified with the parameter "temperature" T and the evolution of the network can be identified with a perturbation of the system. Notice that the algorithm parameter that control temperature schedule is automatically adjusted according to algorithm progress; an analogous criterion, the adaptive simulated annealing, was widely analysed and developed on [13] [14]. The QE of a SOM is defined as the euclidean distance between a data vector and its best matching unit according to:

$$QE = \|x - m_c(x)\|\,, \tag{13}$$

where x is the data vector input and $m_c(x)$ is the bmu. The system evolution can be delineated by the QE progression because, if the QE at the end of each

learning epoch is smaller than the QE computed in previous epoch, then the projection of the samples on the SOM map is closer to the original positions in the input space.

Thus we obtain a linear cooling schedule as $QE_{new} = QE_{old} - \Delta QE$, where ΔQE is the variation of the total energy of the system.

The pseudocode of the algorithm is given in table 2. In this algorithm the term *Training(SOM)* refers to a learning epoch showed in table 1; the result of the training is a candidate SOM that is tested using QE. At the beginning all parameters, including the range of the learning rate, are initialized and the first epoch of the algorithm is executed (steps 1,2). At the end of each learning epoch the QE is calculated and if the difference between the QE calculated at the end of current epoch and the QE calculated at the end of previous epoch is under a threshold δ then the learning process stops (steps 5,5.(d)). Each learning epoch generates a perturbation of the status of neurons in the map. If this perturbation satisfies low-energy criteria according to the simulated annealing (step 5.(f)), then the current configuration is accepted and a new perturbation is calculated. Otherwise, if the perturbation does not satisfy low-energy criteria, then the first perturbation will be used for the next epoch (step 5.(g).i). If the previous configuration is better than the current one, then the previous one is restored (step 5.(g).ii). The size of perturbations depends on the evolution of network resolution or, in other words, by the ratio of the current QE and the maximum QE (step 5.(f).2). The values of the learning rate factor are adapted according to these equations (steps 5.(f).iii, 5.(f).iv).

Figure 1 shows a general flow diagram of the algorithm. The gray area represents the adaptation of SA heuristic to the configuration of neural weights. The algorithm evaluates the configuration of Kohonen map generated by standard SOM, according to low-energy criteria introduced by [19]. The output of gray area is the configuration of neural network and the values of α_{MAX} and α_{MIN}.

4 SOMs for Clustering

Self Organizing maps are useful for 2-D visualization of high dimensional data; the data projection highlights the pattern clusters that can be visually detected, for example, using a U-Matrix representation [20]. For this reason, SOMs can be used for simultaneous clustering and visualization, or even for clustering via visualization.

Moreover, if we compare SOMs with both traditional vector quantization and projection methods (such as MDS [21]), we notice that SOMs present a relevant advantage: they provide a *topology approximation*. Neighbouring elements in the original space are projected to neighbouring grid points. As a consequence SOMs can be used to obtain topographic maps of the multidimensional data space[22, 23]. Secondly they permit to introduce additional data into treatment during the course of computation [22].

Table 2. FLSOM algorithm.

1. Initialize the $SOM_{current}$ with random weights, the SOM parameters: $\alpha_{MAX}, \alpha_{MIN}$, and the epoch counter $p = 1$
2. Start with first learning epoch: $SOM(p) = Training(SOM_{current})$
3. Set $QE_{MAX} = QE(p)$
4. Initialize $\Delta QE(p) = QE_{MAX}$
5. While $\Delta QE(p) \geq \delta$
   (a) $p = p + 1$
   (b) Run a new learning epoch: $SOM(p) = Training(SOM_{current})$
   (c) Calculate $QE(p)$
   (d) Calculate $\Delta QE(p) = QE(p) - QE(p-1)$
   (e) Get a random value $0 < rand < 1$
   (f) if the configuration satisfies low-energy criteria (i.e. $e^{-\frac{\Delta QE}{QE}} < rand$), or the configuration is better than the one of previous epoch ($\Delta QE(p-1) > \Delta QE(p)$)
      i. use current state i.e. set $SOM_{current} = SOM(p)$
      ii. Calculate $\alpha_{inc}(QE) = \Delta\alpha * \left|1 - \frac{QE(p)}{QE_{MAX}}\right|$
      iii. Set $\alpha_{MAX} = \alpha_{MAX} + \alpha_{inc}(QE)$
      iv. Set $\alpha_{MIN} = \alpha_{MIN} + \alpha_{inc}(QE)$
   (g) Else
      i. if ($e^{-\frac{\Delta QE}{QE}} > rand$)) *i.e. configuration does not satisfy low-energy criteria*
      Use the initial vales of α_{MAX} and α_{MIN} in eq. 4
      ii. if ($\Delta QE(p-1) < \Delta QE(p)$) *i.e. previous configuration is better than the current configuration*
      Set $SOM_{current} = SOM(p-1)$
6. End of learning after p epochs

In this work, we use SOMs for data clustering via visualization. In fact it is possible to obtain a cluster algorithm that neither needs a priori knowledge about the number of clusters as an input (as for example K-means).

Our methodology uses a trained SOM and the U-Matrix [24] representation: this way it is possible to look at the resulting map as an image where grayscale contours allow the identification of clusters. Then, using an automatic segmentation process derived from Seeded Region Growing, our framework can highlights, via boundaries recognition, the obtained clusters. The clustering result is evaluated using the techniques described in section 5.

4.1 Advantage of using FLSOM for clustering

Our experimental analysis shows that the SA heuristic offers good chances of finding a configuration with lower internal energy than the initial one. During

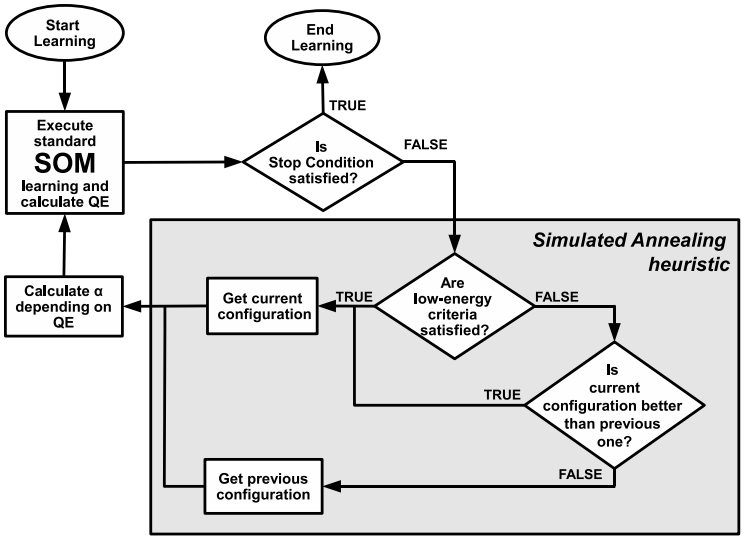

Fig. 1. FLSOM flow chart.

the training phase, the SA heuristic assures the speed-up of the learning process through the increasing of the $\alpha(t)$ value. That implies neuron weights can reach a final configuration where similar patterns are collected faster than with the standard learning algorithm. At the end of the training process, each pattern fits in its related neuron better than in the standard SOM. As a conseguence the algorithm provides a better compactness of similar patterns in specific areas, while preserving topologic relationships. Moreover, this property allows the generation of a lattice where distances among groups of homogeneous patterns are greater than the same distances calculated in a map trained with the standard learning algorithm.

Increasing of both inter-clusters dissimilarity and intra-clusters similarity encourages the adoption of FLSOM algorithm for clustering.

5 Evaluation Criteria

The proposed FLSOM is evaluated against the standard SOM and most of variations analysed in section 3.2: $PLSOM$, $HabSOM$ and $ConsSOM$. In order to compare the five different mechanisms, five data set have been tested and the quality of the approximation, the "smoothness" of the lattice and the entropy are used for comparision.

Two evaluation criteria are used to compare the SOM and FLSOM algorithms in terms of topology preservation and clustering distribution: the former is computed at the end of the learning processes of each method; the latter is evaluated over clusters retrieved from the two trained maps by means of the same detection technique described in 4.

5.1 Learning process Evaluation

Three evaluation criteria are used to measure the quality of the map and to compare the results of the algorithms: quantization error (QE) already defined, regularity degree [15] (RD) and maximization of entropy (EN). The first one measures the resolution of the map, the second one the local distortion and the last one the frequency of input distribution over the map. The RD can be calculated at the end of each learning epoch and it is useful for evaluating the topological organization of the lattice during the training. This parameter is easily calculated as the position of neurons with respect to their neighbors. According to this criterion, an ideal value of the regularity degree should be close to zero (true for a quite flat map). In real applications configurations with low values of degree of regularity are better. The EN ensures us that the quantization intervals are used with the same frequency during the quantization of the input signal. If we have N neural units, we obtain a input manifold divided into V_i intervals where $i = 0, 1, ..., N$. After the training, the input pattern v should fall in interval V_i with a probability:

$$p(V_i) = \frac{1}{N}. \tag{14}$$

So that information-theoretic entropy will be maximized:

$$H = -\sum_{i=1}^{N} p(V_i) * log(p(V_i)) = logN. \tag{15}$$

5.2 Topology Evaluation

The clustering, obtained by a self-organizing map algorithm, can be considered appropriate when two conditionos are satisfied. First, the dataset must be partitioned in the correct number of clusters. And, secondly, each element lies in the correct cluster in a consistent way, with respect to the distribution of elements in the original space. This property is called topology preservation. Several methods have been proposed to measure the topology violation of a trained SOM; the two most popular techniques are the Topographic Product (TP) [26] and the Topographic Function (TF) [27].

For our purpose, TF is not suitable for two reasons [28]. First, it may give misleading results because it does not differentiate between the adjacency of receptive fields in areas where the sample vectors are dense and in areas where they are sparse. Secondly, the comparison between the topographic functions of two different maps is very difficult.

In order to evaluate the topology preservation we use the Directional Product (DP) [29], which is an improved and computationally less expensive implementation of TP.

The maximum value of DP is 1.0 and higher values correspond to better topologies.

5.3 Clustering Evaluation

The results of the FLSOM clustering are compared with the results of the SOM standard algorithms, using a set of validity indexes. There are three approaches to evaluate cluster validity: internal criteria, external criteria and relative criteria [25]. Both external and relative criteria will not be taken in account, because they are based respectively on some user intuition over a specific structure (e.g. class labels), and on the evaluation among several results in terms of the same algorithm but with a different setting of parameters [30]. Internal criteria are easier to use because they can be obtained using the distribution of the output data.

More specifically, internal criteria are based on the minimization of intra-cluster similarity and the maximization of inter-cluster dissimilarity. These methods provide a measure of how close the elements are to their own cluster center (*compactness*) and how far the clusters are from each other (*separation*).

In order to evaluate clustering we use two indices: the *Overall Cluster Quality index* ($Ocq(\xi)$) proposed in [30] and the *Scatter-Distance index* ($SD(\rho)$) developed in [31]. These two indices are built using the following tree quantities: $Compactness$, $Proximity$ and $Distance$.

The $Compactness$ of a cluster is defined as:

$$Compactness = \frac{1}{C}\sum_{i=1}^{C}\frac{var(c_i)}{var(X)}, \tag{16}$$

The $Proximity$ [30] is a function of the distance of the cluster centers and is defined as:

$$Proximity = \frac{1}{C(C-1)}\sum_{i=1}^{C}\sum_{j=1,j\neq i}^{C}\exp\left(-\left\|x_{c_i}-x_{c_j}\right\|\right), \tag{17}$$

where C is the number of clusters generated on the data set X, $var(c_i)$ is the variance of the cluster c_i, $var(X)$ is the variance of the data set X and x_{c_i} and x_{c_j} are two cluster centers.

The $Distance$ [31] is another function of the positions of the cluster centers:

$$Distance = \frac{D_{max}}{D_{min}}\sum_{k=1}^{C}\left(\sum_{z=1}^{C}\left\|x_{c_k}-x_{c_z}\right\|\right)^{-1}, \tag{18}$$

where $D_{max} = \max\left(\left\|x_{c_i}-x_{c_j}\right\|\right)\forall i,j \in \{1,2,...,C\}$ is the maximum distance between cluster centers. The $D_{min} = \min\left(\left\|x_{c_i}-x_{c_j}\right\|\right)\forall i,j \in \{1,2,...,C\}$ is the minimum distance between cluster centers.

Using these three quantities it is possible to build the $Ocq(\xi)$ as:

$$Ocq(\xi) = 1 - [\xi \times Compactness + (1-\xi)\times Proximity], \tag{19}$$

where $\xi \in [0,1]$ is the weight that balances cluster *compactness* and cluster *proximity*.

$SD(\rho)$ is defined as:

$$SD(\rho) = \rho \times Compactness + Distance, \tag{20}$$

where ρ is a weighting factor, introduced because the value of *compactness* could be in a different range than the value of *distance*.

Both indices use the *compactness* to evaluate the variance of clusters and dataset, whereas to evaluate the separation of clusters they use, respectively, the *Proximity* and the *Distance*. Values of the $Ocq(\xi)$ index are in the range $[0, 1]$ and greater values indicate a better result. Whereas low values of the $SD(\rho)$ index indicate a better quality of the clustering.

5.4 Evaluation of the proposed Algorithm

The approach is evaluated using the five SOMs algorithms cited. To allow comparision among SOM implementations, the algorithm shown in table 2 has been modified in step 5, where the stop condition depending on δ has been replaced by a stop condition depending on maximum number of learning epochs. To choose this number, several tries have been run over all used datasets. The number selected, $18 epochs$, was sufficient to approximate a nearly complete evolution, for all SOMs implementations and for all datasets, according to algorithm shown in table 2 with a very little $\delta = 0.05$. The weights of all SOM networks are initialized with random values over the input space and all maps have a 80×80 square lattice. The training phase for each epoch is done with $\alpha_{MAX} = 0.75,\ \alpha_{MIN} = 0.15,\ \sigma_{MAX} = 7,\ \sigma_{MIN} = 2$. In the FLSOM algorithm the values of the learning parameters are dynamically increased up to $\alpha_{MAX} = 1$ and $\alpha_{MIN} = 0.75$; the HabSOM has $\beta = 0.99,\ ab_{MAX} = 1.0$ and tr =3.0E-5; the ConsSOM has B =1.0E-4 and $C = 5.0$. These parameter values are those that give the best results for the datasets used.

5.5 Validation of the proposed framework

The validation of the FLSOM has been carried out using three artificial datasets and two datasets from real world. In detail the first one, here called *Ring*, and the second one, here called $Two - C$, are artificial datasets in two-dimensional space and provide two thousand input signals; the third one $Blobs - 3D$ is an artificial dataset that draws eight blobs in three-dimensional space and provide two thousand and four hundred input signals; the fourth one, the well know $Iris$, is a real dataset in four-dimensional space and provide one hundred fifty instances and there are three clusters, each has 50 instances; the last dataset [16] is a real dataset reduced according to [17] in twenty-dimensional space and provide three hundred twenty-five input signals. The reported figures are averaged over 100 runs of the algorithms. For each epoch of each SOM implementation, the means of QE and EN have been calculated.

Figure 2 shows the average evolution of quantization error during the training process of *Ring* dataset for all SOMs. The chart clearly shows the effectiveness

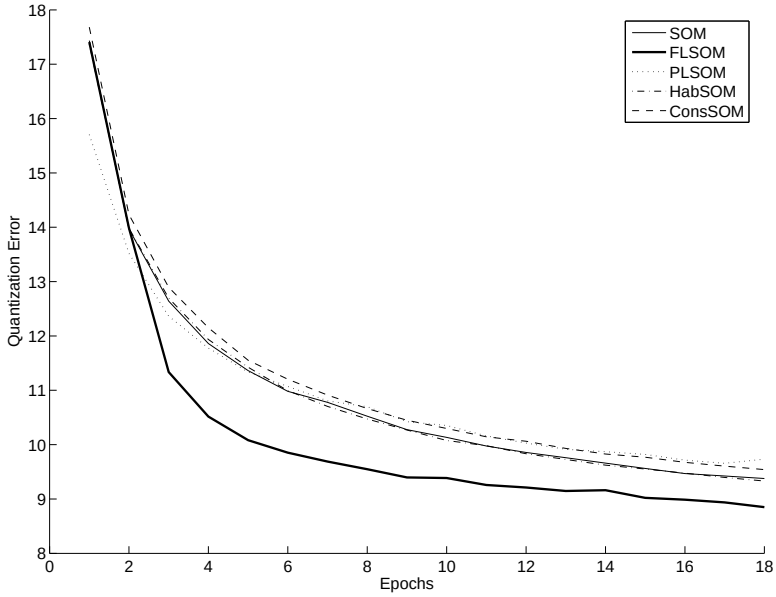

Fig. 2. Quantization error versus number of epochs in *Ring* dataset. The FLSOM algorithm reaches the stop condition with the smallest value of QE.

of the FLSOM algorithm: it reaches a lower QE value in a smaller number of epochs. Results of QE for all datasets are given in table 3. In this table are shown, for all SOMs, the number of epochs (≤ 18) necessary to reach the QE value scored by the worst of all SOMs implementations. The best values for each dataset are emphasized with bold type. These results state that the FLSOM is better than the other implementations with regard to resolution of the map. In other words, the FLSOM is faster than the other algorithms.

Results of RD for all datasets are given in table 4. In this table are shown, for all SOMs, the number of epochs (≤ 18) necessary to reach the RD value scored by the worst of all SOMs implementations. The best values for each dataset are emphasized with bold type. Once again, with real datasets, the FLSOM works better than the other maps. With artificial datasets, instead, FLSOM appears to performing worse than the others SOMs, because it needs the highest number of epochs to reach the same value of RD. Actually, this result is not as bad as it seems: analyzing values of RD scored by all implementations after 18 epochs, we can see the difference is about -4 orders of magnitude, that is a very little distance.

The average entropies calculated after 18 epochs, for all SOMs, are shown in table 5. The best values for each dataset are emphasized with bold type. Even

though values of entropy are closer for all implementations, the FLSOM scores always one of highest results.

Table 3. For all SOMs, the number of epochs necessary to reach required QE value

	Ring	**Two-C**	**Blobs-3D**	**Iris**	**NCI-325**
QE	9.50	8.60	15.80	6.10	8.10
SOM	16	17	17	17	17
FLSOM	**9**	**10**	**3**	**5**	**5**
PLSOM	18+	13	18+	8	18+
HabSOM	16	18	15	17	16
ConsSOM	18	18+	18+	17	18+

Table 4. For all SOMs, the number of epochs necessary to reach required RD value

	Ring	**Two-C**	**Blobs-3D**	**Iris**	**NCI-325**
RD	0.00045	0.00043	0.00046	0.120	0.075
SOM	10	**8**	8	18	18
FLSOM	16	14	13	**10**	**9**
PLSOM	**7**	**8**	9	18+	18+
HabSOM	10	9	**7**	18	17
ConsSOM	10	9	18	18	18

6 Experimental Results for Clustering Quality and Topology Preservation

In order to compare the proposed algorithm against the standard SOM, fourteen data set have been used to test the quality of clustering and the topologic preservation of the obtained maps.

6.1 Evaluation of the proposed algorithm

Experimental tests have been carried out over the datasets for both SOM implementations and the evaluation criteria have been computed. The results have been averaged over several runs in order to compare the performances. The statistical significance of the means has been evaluated using the t-Test.

The weights of all SOM networks are initialized with random values over the input space and all maps have a 80×80 square lattice. The training phase for

Table 5. For all SOMs, the value of Entropy reached after 18 epochs

	Entropy				
	Ring	**Two-C**	**Blobs-3D**	**Iris**	**NCI-325**
SOM	5.489	5.485	5.805	**4.808**	5.166
FLSOM	5.489	**5.490**	5.809	4.807	**5.547**
PLSOM	5.474	5.477	5.773	4.794	5.012
HabSOM	**5.493**	5.480	**5.810**	4.801	5.172
ConsSOM	5.483	5.481	5.772	4.803	5.169

each epoch is done with $\alpha_{MAX} = 0.75$, $\alpha_{MIN} = 0.15$, $\sigma_{MAX} = 7$, $\sigma_{MIN} = 2$, $\delta = 0.15$. In the FLSOM algorithm the values of the learning parameters are dynamically increased up to $\alpha_{MAX} = 1$ and $\alpha_{MIN} = 0.75$. These parameter values are those that give the best results for several datasets widely used in literature.

6.2 Validation of the proposed learning process

The validation of the FLSOM has been carried out using fourteen artificial datasets, thirteen artificial datasets and one real dataset. The thirteen artificial datasets are: the $Two - C$, the $Blobs - 3D$l, then we built eleven datasets in a $\Re^8$ space assembling from 10 to 80 blobs of points on the vertices of the unitary hypercube. They provide one hundred input signals for each blobs (i.e. from 1000 to 8000 patterns). Finally, the real dataset is the well know "$Iris$.

All the reported datasets are used for 50 runs of the learning algorithms. For each epoch of each SOM implementation, the means of Ocq, SD and DP have been calculated.

All artificial datasets are very simple collections of clusters, which are easily detected by both SOM implementation. Both algorithms have produced the correct number of clusters and each element is placed in the appropriate cluster. However, the analysis aims to show that the FLSOM algorithm does actually generate a clustering of a better quality. Our evaluation focuses on the intra and inter-cluster arrangement of elements.

6.3 Results

In table 6 the comparison between SOM and FLSOM in terms of both clustering and topology preservation is shown and better values, for each index and for each dataset, an highlighted with bold type. As shown in table 6 the value of ρ is 1.0, this means both members of equation 20 are weighted in the same manner; in equation 19 the assignement $\xi = 0.5$ is done in order to give equal weights to the two measure. Of course ranges could be different for each clustering parameter (i.e. $Compactness$, $Proximity$ and $Distance$), because each value of the three clustering parameters depends on its own ranges that, in turn, depends

Table 6. Comparison between SOM and FLSOM.

Clustering and Topology indices for three datasets						
	Two-C		**Blobs-3D**		**Iris**	
	SOM	FLSOM	SOM	FLSOM	SOM	FLSOM
(Clustering parameters)						
Compatness	0.7679	**0.7671**	0.1501	**0.1416**	0.5168	**0.4820**
Distance	0.0561	**0.0559**	0.1042	**0.0917**	0.0887	**0.0884**
Proximity	0.0297	**0.0291**	0.0502	**0.0411**	0.0625	**0.0580**
(Clustering indices)						
SD ($\rho = 1$)	0.9269	**0.8338**	1.1656	**0.8527**	1.0979	**0.9307**
Ocq ($\xi = 0.5$)	0.5436	**0.5937**	0.3051	**0.4757**	0.4410	**0.5234**
(Topology index)						
Directional Product	1.0	1.0	0.9786	**0.9818**	0.9756	**0.9846**

Table 7. Gaps between SOM and FLSOM assessed for Blobs-3D dataset.

Clustering and Topology indices for all datasets										
Map Size	**20x20**		**30x30**		**50x50**		**80x80**		**100x100**	
	SOM	FLSOM	SOM	FLSOM	SOM	FLSOM	SOM	FLSOM	SOM	FLSOM
SD index	1.0739	**0.8582**	0.7015	**0.4607**	1.0916	**0.8148**	1.1656	**0.8527**	0.8446	**0.4255**
Ocq index	0.4329	**0.5104**	0.4925	**0.5863**	0.4970	**0.6031**	0.3051	**0.4757**	0.4982	**0.7134**
SD gaps	0.2157		0.2408		0.2768		0.3129		0.4190	
Ocq gaps	0.0775		0.0938		0.1062		0.1706		0.2152	

on dataset and learning configuration. For this reason, for each dataset and for each clustering parameter, all obtained results have been normalized on their own range calculated considering executed runs. Finally, in the bottom of table, the index of topology preservation, *Directional Product*, is reported for each dataset.

Table 7 summaries the value of SD and Ocq indices for dataset *Blob-3D* versus number of neurons in Kohonen map. For each map size, better values are in bold type. As the table shows, not only the FLSOM works better than standard algorithm, but also gaps between indices, or likewise clustering quality, increase when maps increase.

Table 8 shows the comparison between SOM and FLSOM in terms of both Clustering and Topology for 8-dimensional datasets. Each row report results for a dataset. The first and the second columns show respectively the number of blobs (or resulting clusters) and the radius in the artificial datasets. The third column reports in the top the proposed algorithm and in the bottom the standard one. Next three columns show the values of respectively DP, SD and Ocq indices. Best results for them are highlighted with bold type. The proposed algorithm works better than the standard one the most of the time in terms of DP and SD, and it work better in terms of Ocq all the time. The last three columns report significance levels of t-Test, i.e. the probability of does not reject the null hypothesis, for respectively the $Compactness$, the $Distance$ and the $Proximity$ parameters. Worst results are underlined. T-Test significance confirms the most

Table 8. Comparison between SOM and FLSOM for 8-dimensional datasets.

Clustering and Topology indices for all 8-dimensional datasets								
Blobs	**rad**	**Learning Algorithm**	**DP**	**SD** ($\rho = 1$)	**Ocq** ($\xi = 0.5$)	**t-Test significance** Comp	Dist	Prox
10	0.1	FLSOM	**0.9853**	**0.6956**	**0.6705**	0.0094	0.0033	$1.1E^{-5}$
		SOM	0.9677	0.8223	0.45146			
10	0.2	FLSOM	**0.9889**	0.7129	**0.7111**	0.0026	0.0506	$1.0E^{-6}$
		SOM	0.9790	**0.5488**	0.5085			
10	0.3	FLSOM	**0.9885**	0.8947	**0.6354**	0.0060	0.2160	$1.0E^{-6}$
		SOM	0.9787	**0.6903**	0.4982			
15	0.1	FLSOM	**0.9828**	0.9493	**0.5464**	0.0018	0.2267	0.0037
		SOM	0.9768	**0.7211**	0.4968			
15	0.2	FLSOM	**0.9864**	**0.8676**	**0.4886**	0.8285	0.0008	$3.2E^{-5}$
		SOM	0.9831	1.3764	0.2920			
20	0.1	FLSOM	**0.9876**	**1.0267**	**0.4044**	0.0001	0.0029	0.0399
		SOM	0.9869	1.0624	0.3857			
20	0.2	FLSOM	**0.9821**	**1.0305**	**0.5318**	0.3436	0.7019	0.1026
		SOM	0.9816	1.7168	0.4542			
30	0.1	FLSOM	0.9824	**0.8196**	**0.4726**	0.0917	0.0165	0.0270
		SOM	**0.9832**	0.8541	0.3831			
65	0.1	FLSOM	**0.9772**	**0.7058**	**0.5791**	0.6880	0.0665	0.0354
		SOM	0.9757	0.9630	0.3374			
70	0.1	FLSOM	**0.9769**	**0.5874**	**0.6135**	0.4819	0.5805	0.7158
		SOM	0.9765	0.6319	0.6086			
80	0.1	FLSOM	0.9727	**0.5236**	**0.7233**	0.4947	0.3392	0.2972
		SOM	**0.9732**	0.6735	0.6500			

of the results are computed for different distributions. It should be no surprise that t-Test get worst result for datasets with several clusters (i.e. 70 - 80), in fact the distribution of the interaction amongs many blobs becomes less distinguished when a few features have to define several clusters.

7 Conclusions

In this paper FLSOM, a technique that uses the Simulated Annealing as a method to select a candidate SOM during the training process, has been proposed. The SOM training process modifies the learning rate factor in an adaptive way. Results of experimental tests, carried out on both artificial and real datasets, comparing the proposed method with standard and modified SOMs demonstrate the good performances obtained by FLSOM in terms of convergence time, and resolution of the maps. On the other hand, the performances obtained by FLSOM in terms of local distortion and frequency of input distribution over the map are comparable with those obtained by standard and modified SOMs, especially when real world datasets are considered. Then the proposed algorithm has been tested for clustering. In order to compare the FLSOM against the standard SOM, a tool for automatic extraction of clusters based on SOM, U-Matrix

visualization and segmentation techniques, has been implemented. Results of experimental tests, carried out on both artificial and real datasets, comparing the proposed method with standard SOM implementation, demonstrate the good performances obtained by FLSOM in terms of clustering and topology preservation.

References

1. R. Rizzo, A. Chella, "A Comparison between Habituation and Conscience Mechanism in Self-Organizing Maps", IEEE Transactions on neural networks, vol. 17, no. 3, pp. 807-810, 2006
2. D. DeSieno, "Adding a conscience to copetitive learning", *Proc. ICNN'88, International conference on Neuroal Networks, IEEE Service Center*, Piscataway, N J, pp.117-124, 1988.
3. E. Berglund, J. Sitte, "The parameterless self-organizing map algorithm", IEEE Transactions on neural networks, vol. 17, no. 2, pp. 305-316, 2006.
4. K. Haese, "Auto-SOM: recursive parameter estimation for guidance of self-organizing feature maps," *Neural Comput.*, vol. 13, no. 3, pp. 595-619, 2001.
5. S. Kirkpatrick, C. Gelatt, M. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, No. 4598, pp 671-680, 1983
6. T. Graepel, M. Burger and K. Obermayer, "Self-organizing maps: generalizations and new optimization techniques", *Neurocomputing*, vol.21, pp. 173-190, 1998.
7. H. Douzono, S. Hara, Y. Noguchi, "A Clustering Method of Chromosome Fluorescence Profiles Using Modified Self Organizing Map Controlled by Simulated Annealing", *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*,vol. 4, 2000.
8. K. Haese, "Kalman filter implementation of self-organizing feature maps," Neural Comput., vol. 11, no. 5, pp. 1211-1233, 1999.
9. K. Haese, "Self-organizing feature map with self-adjusting learning parameters," *IEEE Transactions on Neural Network*, 9(6), 1270-1278, 1998.
10. E. Berglund, J. Sitte, "The parameter-less SOM algorithm", in Proc. ANZIIS, pp. 159-164, 2003.
11. T. Kohonen, *Self-Organizing Maps.* Berlin: Springer Verlag, 1995.
12. M. Van Hulle, editor. *Faithful Representations and Topographic Maps: From Distortion- to Information-Based Self-Organization.* John Wiley, New York , 2000.
13. L. Ingber, "Very fast simulated re-annealing", *Journal of Mathl. Comput. Modelling*, vol. 12, no. 8, pp. n967-973, 1989.
14. L. Ingber, "Adaptive simulated annealing (ASA): lessons learned", *J. Control and Cybernetics*, vol. 25, No.1, pp.33-54, 1996.
15. J. Goppert, W. Rosenstiel, Regularized SOM-Training: A Solution to the Topology-Approximation dilemma, University of Tbingen,1996.
16. National Cancer Institute, "DTP AIDS antiviral screen dataset [online]. http://dtp.nci.nih.gov/docs/aids/aids/data.html."
17. G. Di Fatta, A. Fiannaca, R. Rizzo, A. Urso, M. R. Berthold and S. Gaglio, *Context-Aware Visual Exploration of Molecular Databases*, Workshops IEEE International Conference on Data Mining (ICDM 2006), dec 18-22, 2006, pp.136-141.
18. Kohonen, T.: Self-Organizing Maps. 3rd ed., Springer, Berlin (2001)
19. Metropolis, N.,Rosenbluth , A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. J. Chem. Phys., vol. 21 (6), pp. 1087–1092 (1953)

20. Deboeck, G., Kohonen, T.: Visual Explorations in Finance using Self-Organizing-Maps. Springer (1998)
21. Ripley, B. D.: Pattern Recognition and Neural Networks. Cambridge University Press (1996)
22. Kaski, S.: Data Exploration Using Self-Organizing Maps. PhD thesis, Helsinki University of Technology (1997)
23. Flexer, A.: On the Use of Self-Organizing Maps for Clustering and Visualization. Springer, vol. 1704, pp. 80–88 (1999)
24. Ultsch, A., Korus, D.: Integration of Neural Networks and Knowledge-Based Systems. Proceeding IEEE on International Conference on Neural Networks, pp. 425–426 (1995)
25. Halkidi, M., Vazirgiannis, M.: Clustering Validity Assessment: Finding the Optimal Partitioning of Data Set. Proc. of ICDM 2001, pp. 187–194 (2001)
26. Bauer, H.U., Pawelzik, K. R.: Quantifying the neighborhood preservation of self-organizing feature maps. IEEE Transaction on Neural Networks, vol. 3, no. 4, pp. 570–579 (1992)
27. Villman, T. , Der, R. , Hermann, M., Martinetz, T.: Topology preservation in self-organizing feature maps: Exact definition and measurement. IEEE Transaction on Neural Networks, vol. 8, no. 2, pp. 256–266 (1997)
28. Kiviluoto, K.: Topology Preservation in Self-Organizing Maps. Proc. of ICNN 1996, pp. 294–299 (1996)
29. Vidaurre, D., Muruzabal, J.: A Quick Assessment of Topology Preservation for SOM Structures. IEEE Transactions on Neural Networks, vol.18, no.5, pp 1524–1528 (2007)
30. Yang, Y., Kamel, M. S.: An aggregated clustering approach using multi-ant colonies algorithms. Pattern Recognition, vol. 39, no. 7, pp. 1278–1289 (2006)
31. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Cluster validity methods: part II. SIGMOD Rec., vol. 31, no. 2, pp. 40–45 (2002)