A New SOM Initialization Algorithm for Nonvectorial Data

Antonino Fiannaca[1,2], Riccardo Rizzo[2],
Alfonso Urso[2], and Salvatore Gaglio[1,2]

[1] Dipartimento di Ingegneria Informatica, Universitá di Palermo, Italy
[2] ICAR-CNR, Consiglio Nazionale delle Ricerche, Palermo, Italy

Abstract. Self Organizing Maps (SOMs) are widely used mapping and clustering algorithms family. It is also well known that the performances of the maps in terms of quality of result and learning speed are strongly dependent from the neuron weights initialization. This drawback is common to all the SOM algorithms, and critical for a new SOM algorithm, the Median SOM (M-SOM), developed in order to map datasets characterized by a dissimilarity matrix. In this paper an initialization technique of M-SOM is proposed and compared to the initialization techniques proposed in the original paper. The results show that the proposed initialization technique assures faster learning and better performance in terms of quantization error.

Keywords: Median SOM, initialization, pairwise data.

1 Introduction

The self-organizing maps (SOMs) are unsupervised neural networks used to support the exploration of a set of multidimensional patterns in clustering and classification applications. It is also well known that SOM results are heavily influenced by the initialization of the neurons configuration in the input space, because a proper initialization can reduce the time required for the training phase and the quantization error or network performances.

The initialization techniques can be grouped in two main categories: *random initialization techniques* that are not linked to the input pattern set, and *dataset techniques based on the analysis of the training data* trying to exploit some regularity or manifold of the input data. In random initialization the neural weights are chosen usually inside an hypercube that contains the input dataset[1]; the most common dataset techniques are sample and linear initialization: in the sample initialization[2], weight values are a random selection of input patterns, while in linear initialization[1,3] weight vectors are defined along the subspace spanned by the principal eigenvectors of the input dataset.

For many sets of real objects it is difficult to obtain a vector space representation, but it is possible to calculate a pairwise dissimilarity matrix. For example it is easy to compare DNA or Protein sequences using alignment techniques, or compute an edit distance for two generic strings, but it is difficult to set up a method to identify a set of meaningful features and to build a vector space in order to use the Euclidean distance. From a dissimilarity matrix it is possible to obtain a SOM map using a Median SOM (*M-SOM*), a modification of the SOM algorithm proposed in [4]. In [4,5,6] these maps were used to visualize and cluster nonvectorial data. Notice that an efficient initialization algorithm, as well as a fast learning algorithm, plays a crucial role in computation of large datasets. For example in the genomic field [8] gene expression datasets and microarray collections are modeled using a pairwise data matrix. This representation becomes relevant for biological data, because specific distance measures have been created, also knows as evolutionary distances. In this paper, a new technique based on the fundamental idea of linear initialization will be introduced. Compared with the previous proposed technique[4], our initialization method does not require a conversion of data into vectorial representations and does not introduce extra auxiliary model vectors associated with the nodes.

The paper has the following structure: the next section reports an overview of the Median SOM algorithm and describes the batch learning process; the section 3 shows in detail the proposed initialization algorithm; the section 4 reports the evaluation criteria and the experimental results. Finally some conclusions are reported in section 5.

2 SOM Learning Algorithms for Nonvectorial Data

SOM neural networks can be trained using two classes of learning algorithms: the on-line and the batch learning; advantages and drawbacks of these algorithms has been discussed in [10]. The Median SOM is made by a lattice of $N = m \times n$ neurons (called models in Median SOM). Each neuron is not a weighted vector, but it is an element corresponding to an input pattern. In this work a batch learning algorithm trained by epochs is used, and its pseudo-code is reported in table 1. The batch learning algorithm for a Median SOM begins with a random assignment of an input pattern to each unit of the model grid (*sample initialization*).

In (step 3.(b).i) the best matching unit (*bmu*) selection is performed by the *affectation phase*, where each input is associated with a model, using the distance defined in the input pairwise matrix; this way, each map unit i collects a list of pattern to whom the reference model of unit i is the nearest reference pattern. In (step 3.(c)) the update process is performed by the *representation phase*, where the prototype of each model is updated [4]; this way each map unit i takes for the new reference pattern the median over the union of the lists that belongs to the topological neighbourhood of unit i [5]. The topological neighbourhood is computed using a gaussian function kernel around the best matching unit, and its neighbourhood radius $\sigma(t)$ is a decreasing function of time.

Obviously the proposed algorithm will take place in (step 1) of Table 1, and advantages of its use will be measured, epoch by epoch, at (step 3.(d)) of Table 1, when the evolution of the neural network is evaluated with the Quantization

Table 1. Median SOM batch learning algorithm

```
1. Initialize the $N$ neurons;
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 $bmu(x(t))$;
       iii. Remove $x(t)$ from $RndDataset$;
       iv. $t = t + 1$;
   (c) For all $N$ neurons update reference model;
   (d) Calculate Quantization Error at epoch $p$;
   (e) $p = p + 1$, $t = 1$;
4. End of learning after $MaxP$ epochs.
```

Error (QE). The local QE of a M-SOM is defined as the pairwise distance between a data vector input and its best matching unit.

3 The Initialization Algorithm for Median SOM

As the authors pointed out in [4,6] the SOM and M-SOM algorithms are heavy dependent from initialization. In the above papers the authors observed experimentally that the convergence of the SOM algorithm is significantly faster and safer, if the initial models are at least roughly ordered in two dimensions. For this reason, they propose a vectorial initialization method that uses a projection of the pairwise data into a vectorial space, and then a standard SOM algorithm execution to find models.

The main idea of the proposed initialization algorithm is the projection, over the Kohonen map, of some distinctive patterns of the dataset, preserving theirs mutual relationships. In order to perform this projection, a totally connected undirect graph $G(V, E)$ (the "*initGraph*") is built using some selected patterns of the dataset. In this graph V is the set of vertices of the graphs that are selected among the most distant elements of the input data and E is the set of edges, where the length of each edge $E(i, j)$ is closer to dissimilarity between patterns i, j, in the pairwise matrix. The constraint "closer to" instead of "equal to" is used because the geometry of the graph must be respected. The introduced algorithm resolves some mutual conflicts through "elastic edges" that would balance all pairwise relationships using a mean square error.

The *initGraph* plays a fundamental role in our initialization technique for two reasons: mainly it performs a bridge between original space of dataset and two-dimensional space of Kohonen map, secondly it can preserve most of mutual relations among selected patterns.

Table 2. Pseudo code of inizialization algorithm. Step 1.

```
1. Set $k =$ number of biggest values to take into account in pairwise matrix;
2. Let $p =$ number of rows and columns that contain the $k$ biggest values ($p \leq 2 \times k$);
3. Mark the greater $k$ values in pairwise matrix;
4. Mark the $p$ elements related to k marked distance;
5. Extract the pairwise sub-matrix $S$ ($p \times p$) with only marked elements;
```

The initialization is made in the following three steps:

1. *Selection* of the k most distant patterns (elements) of the input dataset; these patterns are selected using the biggest values (distances) in the pairwise dissimilarity matrix;
2. *Arrangement* of selected patterns into a 2D space using a 2D projection of the *initGraph*;
3. *Fitting* the *initGraph* projection into the Median SOM.

The first step is the retrieval of the most distant elements for the dataset, whose identification is done by exploration of input pairwise matrix. The aim is to obtain a subset of patterns that carry out information about the patterns set. First of all, a pairwise sub-matrix is pulled out from the pairwise matrix input. Selection of patterns, used to generate the pairwise sub-matrix, depends on values in pairwise dissimilarity matrix. These values can be regarded as distances between patterns. Greater distances will be taken into account to build an initialization map where the most distant elements will be placed in opposite areas of the network, according to the *initGraph*. Pseudo code for *step*1 is reported in Table 2.

The second step is the devoted to build a model able to translate relation among selected patterns into a two dimensional structure; this model must observe the geographic distance among areas where selected patterns should be located. Pseudo code for *step*2 is reported in Table 3.

The third step is the adaptation of initialization graph on the Kohonen map. Obviously the graph will be properly stretched and scaled before being overlapped on the map. After that, selected patterns will be located over some neurons, which set their models equal to the corresponding representative patterns. Remaining neurons will be initialized depending on the previous ones; in fact each neuron into the nearest neighbourhood of a neuron selected at previous step, will be initialized with its model. Remaining neurons will be initialized with random models, pulled out from input patterns. In this manner, during learning process, samples that are similar to selected patterns will fall into the neighbourhood of them. The implemented graph is eventually located in the Kohonen map, properly scaled, in order to assing models to the neurons. Pseudo code for *step*3 is reported in Table 4.

Table 3. Pseudo code of initialization algorithm. Step 2.

```
1. Set $tol$ = tolerance adopted for elastic edges;
2. Generate a totally connected undirect graph $G(V, E)$ where vertices $V$ are the first
   three marked elements of $S$;
3. Set lengths of edges $E(i, j)$ equal to value (distance) between features $i, j$ in pair-
   wise dissimilarity matrix (with respect to geometry of graph);
4. Calculate the Mean Square Error among all edges $E(i, j)$ and theirs theoretical
   distances (the pairwise dissimilarity between patterns $i, j$ in pairwise matrix $S$)
   according to $MSE(||E(i, j) - S(i, j)||)$;
5. While the tolerance adopted for elastic edges is not satisfied ($MSE \geq tol$),
   (a) Add a random little value $\epsilon_1$ to each edge $E(i, j)$, in order to get each edge
       closer to its theoretical measure (according to pairwise matrix);
   (b) Decrease parameter $tol$ with a random little value $\epsilon_2$ ($tol = tol - \epsilon_2$);
6. For $c = 4$ to $p$ (for each remaining $p - 3$ rows or columns of the matrix $S$),
   (a) Add a new vertex $v$, corresponding to element $c$ of $S$, and $c(c-1)/2$ new edges;
   (b) Calculate measures of new edges using $S$ matrix (with respect to geometry of
       graph);
   (c) Calculate the Mean Square Error among all edges $E(i, j)$ and theirs theoretical
       distances, as in step 4;
   (d) While the tolerance adopted for elastic edges is not satisfied ($MSE \geq tol$),
        i. Add a random little value $\epsilon_1$ to each edge $E(i, j)$, in order to get each edge
           closer to its theoretical measure (according to pairwise matrix);
       ii. Decrease parameter $tol$ with a random little value $\epsilon_2$ ($tol = tol - \epsilon_2$);
7. Resulting graph is the "initialization graph".
```

Table 4. Pseudo code of initialization algorithm. Step 3.

```
1. Let $m$ = height and $n$ = width of M-SOM grid, thus number of neurons is $m \times n$;
2. Build a bounding box $R$ for the initGraph;
3. Scale the box $R$ in order to obtain a rectangle $m \times n$;
4. For $s = 1$ to $p$ (for each vertex of the graph),
   (a) Find the neuron overlapped with vertex $s$ and assign to its prototype the model
       corresponding to vertex $s$;
   (b) Assign the same model of the selected neuron to each neuron into its neigh-
       bourhood with radius = 1.
5. Each remaining neuron will be initialized with a model selected randomly.
```

4 Experimental Results

In this section the comparison among the proposed, the sample and the vectorial initialization method is evaluated. In order to reach this goal, three M-SOMs are

Table 5. Average execution time measured in sec. of M-SOM learning process for the three initialization algorithms

Average Execution Time of training processes			
	Proposed	**Sample**	**Vectorial**
M-SOM Initialization	0.029	–	152.610
M-SOM Learning process	192.827	193.342	193.315

implemented with the same learning algorithm, but with different initialization algorithms. The result of training process will be evaluated using the evolution of quantization error.

4.1 Quality Criteria

The Quantization Error is used to compare the effectiveness of the different initialization methods. This criterion is commonly used in evaluation of neural networks resolution. Moreover this popular measure is easy to compute and well defined for M-SOM. The evolution of QE during training process was monitored over 50 experiments for each initial configuration and means of achieved values will be analyzed. The significance level of analyzed means are evaluated by the t-Test, that checks if the means of compared groups are statistically different from each others.

4.2 Evaluation of the Proposed Algorithm

In order to compare the M-SOM results after the training phase it is necessary to select a suitable number of epochs; after several trials it was seen that 20 epochs are enough to have a fixed configuration of models in the M-SOM. All the maps have a 20×20 square lattice. The training phase for each epoch is done with a neighbourhood radius function that decreases exponentially from $\sigma_{max} = 5$ to $\sigma_{min} = 1$.

In the vectorial initialization algorithm, a projection of input data into a 50-dimensional space is performed using the Sammon Projection [7]; then a SOM with a square lattice 20×20 neurons, a learning rate function that decreases exponentially from $\alpha_{MAX} = 0.75$ to $\alpha_{MIN} = 0.15$, and a neighbourhood radius function, that decreases exponentially from $\sigma_{MAX} = 4$ to $\sigma_{MIN} = 1$, is used in order to find models. These values offer the best result for the used dataset.

In the proposed algorithm, the $k = 6$ biggest values in pairwise matrix have been taken into account. These parameter values are those that give the best results for several datasets widely used in literature.

4.3 Validation of the Proposed Initialization Process

The validation of the M-SOM has been carried out using a real world dataset, [8], here called *Garrity-RNA*, made of 1436 small subunit ribosomal RNA sequences,

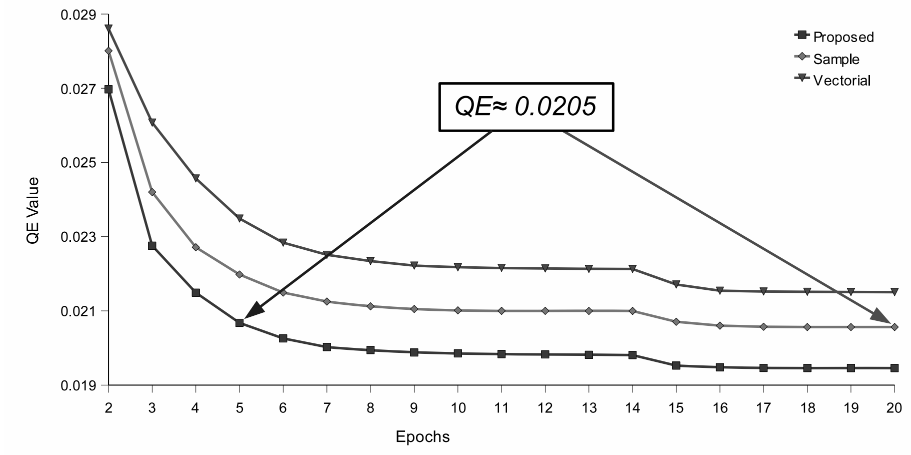

Fig. 1. Quantization error versus number of epochs for Garrity-RNA dataset. QE values at first steps are not shown. Starting values for porposed, sample and vectorial algortihms are respectively 0.0535, 0.0571 and 0.0336. The Sample initialization algorithm reaches the stop condition at the epoch $p = 20$ with a $QE \approx 0.0205$. Almost the same QE value was reached by proposed initialization algorithm at epoch $p = 5$.

that were initially classified (based on the GenBank [9] annotation) as belonging to the *Gammaproteobacteria*.

Figure 1 shows the average evolution of quantization error during the training process of *Garrity-RNA* dataset for the three initialization algorithm. The chart clearly shows the effectiveness of the proposed algorithm in terms of resolution of the map, in fact the map trained with the proposed algorithm reaches the lowest QE value among the maps. Moreover the sample initialization algorithm reaches the stop condition at the epoch $p = 20$ with a $QE \approx 0.0205$, whereas the proposed one reaches almost the same QE value at epoch $p = 5$. Notice that the evolution of M-SOM learning process, initialized with the vectorial method, starts with the lower value (epoch 1 not reported in figure 1), with respect to the other maps, but it ends with a greater QE value: this means that the M-SOM learning process with the vectorial initialization is fallen out in a local minimum for the network. Since the curves in the figure seem very close and have almost the same shape, and it should be possible that they come from the same distribution, the t-Test has been calculated. The test rejects the null hypothesis with a level of significance = 0.17%.

Table 5 reports average execution times for SOMs initialized with the three techniques. All quoted times are derived from tests on a machine having a 3.00GHz Pentium IV processor, 1014Mb of RAM, Windows Vista Business 32bit operating system. Our technique spends 0.038 sec to execute the Median SOM process, whereas vectorial initialization spends 131.878 sec. We assume the time of the sample initialization technique to be equal 0 sec. The time percentage of the proposed algorithm is about 0.015% of the learning process. This is a very small value with respect to the advantages shown in previous figures.

5 Conclusions

In this paper a new initialization algorithm for SOMs with non-vectorial data input is proposed. Unlike previous initialization methods, the proposed one does not require a conversion of data into vectorial representations, but it introduces a totally connected undirect graph (here called *initGraph*) that connects some key patterns; the edges of *initGraph* take into account the pairwise dissimilarity among all key patterns. Finally, a projection of the *initGraph* over the neuron grid reports the distance informations into the map.

This method has been compared with both sample and vectorial initialization techniques. Results of experimental tests, carried out on a real biological dataset, demonstrate the good performances obtained by our technique in terms of resolution of the map and execution time.

References

1. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Berlin (2001)
2. Varsta, M.: Self organizing maps in sequence processing. Dissertation, Department of Electrical and Communications Engineering, Helsinki University of Technology (2002)
3. Vesanto, J., Alhoniemi, E.: Clustering of the Self-Organizing Map. J. IEEE-NN 11(3), 586–600 (2000)
4. Kohonen, T., Somervuo, P.: How to make large self-organizing maps for nonvectorial data. Neural Network, 945–952 (2002)
5. Kohonen, T., Somervuo, P.: Self-organizing maps of symbol strings, pp. 413–418. IEEE Press, Piscataway (1998)
6. Oja, M., Somervuo, P., Kaski, S., Kohonen, T.: Clustering of human endogenous retrovirus sequences with median self-organizing map. In: Workshop on Self-Organizing Maps (2003)
7. Sammon, J.W.: A nonlinear mapping for data structure analysis. J. IEEE Transactions on Computers 18, 401–409 (1969)
8. Garrity, G.M., Lilburn, T.G.: Self-organizing and self-correcting classifications of biological data. J. Bioinformatics 21(10), 2309–2314 (2005)
9. National Center for Biotechnology Information, Entrez Nucleotide query, `http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Nucleotide`
10. Fort, J.C., Letrrmy, P., Cottrell, M.: Advantages and drawbacks of the Batch Kohonen algorithm. In: Verleysen, M. (ed.) ESANN 2002, pp. 223–230 (2002)