A New Modification of Kohonen Neural Network for VQ and Clustering Problems

Ehsan Mohebi[1] **Adil M. Bagirov**[2]

[1,2]School of Science, Information Technology and Engineering
University of Ballarat, Victoria, 3353, Australia,
Email: [1]e.mohebi@ballarat.edu.au, [2]a.bagirov@ballarat.edu.au

Abstract

Vector Quantization (VQ) and Clustering are significantly important in the field of data mining and pattern recognition. The Self Organizing Maps has been widely used for clustering and topology visualization. The topology of the SOM and its initial neurons play an important role in the convergence of the Kohonen neural network. In this paper, in order to improve the convergence of the SOM we introduce an algorithm based on the split and merging of clusters to initialize neurons. We also introduce a topology based on this initialization to optimize the vector quantization error. Such an approach allows one to find global or near global solution to the vector quantization and consequently clustering problem. The numerical results on 4 small to large real-world data sets are reported to demonstrate the performance of the proposed algorithm.

Keywords: Self Organizing Maps, Clustering, Vector Quantization, Split and Merge Algorithm.

1 Introduction

Clustering is the process of learning concept of raw data by dividing the data into groups of similar objects (Berkhin 2006, Arous 2010). Many clustering algorithms have been proposed based on statistics, machine learning, neural networks and optimization techniques (Jain et al. 1999, Berkhin 2006).

The self organizing map (Kohonen 2001) (SOM) is the well known data mining tool where the aim is to visualize a high dimensional data into usually a 2-Dim grid. The SOM contains a set of neurons that gradually adapts to input data 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 (Yang et al. 2012). 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 (Haese 1998, Fiannaca et al. 2011, 2007, Goncalves et al. 1998).

Different versions of the SOM have been introduced in (Alahakoon et al. 2000, Appiah et al. 2012, Arous 2010, Ayadi et al. 2012, Brugger et al. 2008, Cheng et al. 2009, Chi & Yang 2006, 2008, Cottrell et al. 2009, Ghaseminezhad & Karami 2011, Gorgonio & Costa 2008, Lapidot et al. 2002, Shah-Hosseini & Safabakhsh 2003, Tasdemir et al. 2011, Vesanto & Alhoniemi 2000, Wong et al. 2006, Xu et al. 2005, Yang et al. 2012, Yen & Wu 2008, Zheng & Greenleaf 1996). The paper (Wong et al. 2006) presents an automated detection algorithm based on the SOM assuming that the training data is adequate representation of the sample distribution. Therefore, the SOM is trained using a small proportion of the sample data set and the algorithm defines a region around prototypes by employing a parameter $r_j,\ j = 1, \dots, q$ (where q is the number of neurons) that represents the distance of the farthest projected sample into the neuron j. The upcoming samples are distributed into the network and novelties are those samples which cannot fit into these regions. A combinatorial two-stage clustering algorithm based on the SOM is introduced in (Chi & Yang 2008). The numerical results of the enhanced SOM using the Ant Colony Optimization technique and the k-means demonstrates the superiority of the proposed algorithm in comparison with the SOM and k-means. Similarly in (Brugger et al. 2008) an enhanced version of the Clusot algorithm (Bogdan & Rosenstiel 2001) is applied in the SOM for automatic cluster detection. In (Tasdemir et al. 2011) the SOM's prototypes are clustered hierarchically based on the density instead of the distance dissimilarity. Recently, a new two-stage algorithm is proposed in (Yang et al. 2012) that applies the graph cut algorithm (Shi & Malik 2000) to the SOM output. Results presented demonstrate that this algorithm outperforms direct clustering methods using less computational time.

A dynamic SOM is a version of the SOM where its structure is not fixed during the learning phase. In (Alahakoon et al. 2000), a growing self organizing map (GSOM) is presented which defines a spread factor to measure and control the growth of the network. Similarly in (Ayadi et al. 2012), a multi level interior growing SOM is introduced. Unlike the GSOM, which allows the growth only from border sides, this algorithm allows neurons to grow even from an interior node of the map.

Another extension of the SOM algorithm is presented in (Haese 1998). This extension automatically calculates the learning parameters during the training. The algorithm is based on the Kalman filter estimation technique and the idea of the topographic product. The Fast Learning SOM (FLSOM) algorithm is presented in (Fiannaca et al. 2011), which is based on the application of the simulated annealing (SA) metaheuristics to the SOM learning. The SA is used to modify the learning rate factor in an adaptive way. The FLSOM shows a good convergence, better than the original SOM algorithm.

In all above modifications of the SOM there are no any specific procedures to initialize neurons. Therefore the most of these algorithms are still sensitive to the initialization of neurons. In this paper, to improve the performance of the SOM we propose an initialization algorithm based on the split and merge procedure. The high dense areas in input data space are detected by the proposed split and merge procedure. Then neurons are generated in those detected areas. A new topology is presented for the SOM to restrict the adaption of the neurons to those neighborhood ones which are located in the same high density areas. Such an approach leads to better local minimum of the quantization error than that of by the SOM. 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 and its learning algorithm are presented in Section 3. In Section 2, the split and merge procedure is introduced. The SOM initialization algorithm is presented in Section 3.1. In Section 3.2 a new topology for the SOM is proposed. The modified SOM algorithm and its implementation are discussed in Section 4. Numerical results are presented in Section 5 and Section 6 concludes the paper.

2 Merging and Splitting Algorithms

In this section we discuss splitting and merging procedures in cluster analysis. More specifically, first we present one algorithm for splitting and one algorithm for merging. Finally, we present an algorithm based on the combination of these two algorithms.

Assume that we have a set of k cluster centers $\Lambda = \{c_1, \cdots, c_k\}$, $c_i \in \mathbb{R}^n$. These centers are solutions to the following problem:

$$\text{minimize } f = \sum_{i=1}^{k} \sum_{j=1}^{m} \|x_j - c_i\|^2 \text{ where } x_j \in C_i, \quad (1)$$

here c_i is the center point of the set C_i.

In some cases data points from the set C_i are not dense in some neighborhood of its center c_i. Given a radius $\varepsilon > 0$ we consider the following two sets for the cluster C_i:

$$\Phi_c^i(\varepsilon) = \{x_j \in C_i | \ d(x_j, c_i) \le \varepsilon\}, \quad (2)$$

and

$$\Phi_s^i(\varepsilon) = \{x_j \in C_i | \ \varepsilon < d(x_j, c_i) \le r_i\},$$

where

$$r_i = \max_j \{d(x_j, c_i) | \ x_j \in C_i\}, \ i = 1, \dots, k.$$

Two clusters C_i and C_l are said to be well separated if $d(c_i, c_l) \ge (r_i + r_l)$.

It is clear that for any cluster $C_i, i = 1, \dots, k$ there exist $\varepsilon_i \in (0, r_i]$ such that $|\Phi_c^i(\varepsilon)| = \max(|\Phi_c^i(\varepsilon)|, |\Phi_s^i(\varepsilon)|)$ for all $\varepsilon \in (\varepsilon_i, r_i]$. Consider the following equation:

$$\varepsilon_i = \beta r_i.$$

where $\beta \in (0, 1)$. If the ε_i is sufficiently small then data points from the cluster C_i are dense around its center c_i.

The ε_i will be used to design a splitting algorithm for clusters whereas the definition of well separated clusters will be used to design a merging algorithm.

2.1 Splitting

In this subsection we describe the splitting procedure for clusters. This will be done using the parameter β and also special scheme to identify parts of a cluster where most of point reside.

Assume that a set of k clusters, $\Omega = \{C_1, ..., C_k\}$ and a number $\beta \in (0, 1)$ are given. The number of points within the radius $\varepsilon_i = \beta r_i$ from the center of the cluster C_i is:

$$w_c^i = |\Phi_c^i(\varepsilon_i)|.$$

We introduce the angle $\theta_{i,j}$ between the cluster center c_j and the data point $x_i \in C_j$ as follows assuming both $c_j \ne 0$ and $x_i \ne 0$:

$$\theta_{i,j} = \arccos \frac{\langle x_i, c_j \rangle}{\|x_i\| \|c_j\|}. \quad (3)$$

Remark 1 *In order to make (3) well-defined we transform the cluster C_j so that the point $v = (\delta, \dots, \delta) \in \mathbb{R}^n$ becomes its center. Here $\delta > 0$ is a sufficiently small number, say $\delta \in (0, 0.1]$. It is clear that points x_i from this cluster will be transformed as follows:*

$$\bar{x}_i^t = x_i^t - c_j^t + \delta, \ t = 1, \dots, n.$$

Moreover we consider only those $\bar{x}_i$ which satisfy the following condition:

$$\varepsilon_j < d(\bar{x}_i, v) \le r_j.$$

Then the angle $\theta_{i,j}$ is defined between v and $\bar{x}_i$.

Now we introduce the following two sets:

$$\Phi_u^j(\varepsilon_j) = \{x_i \in C_j | \ \varepsilon_j < d(x_i, c_j) \le r_j, \ 0 \le \theta_{i,j} \le \frac{\pi}{2}\}, \quad (4)$$

and

$$\Phi_d^j(\varepsilon_j) = \{x_i \in C_j | \ \varepsilon_j < d(x_i, c_j) \le r_j, \ \frac{\pi}{2} \le \theta_{i,j} \le \pi\}. \quad (5)$$

The cardinalities of these sets are $w_u^j = |\Phi_u^j(\varepsilon_j)|$ and $w_d^j = |\Phi_d^j(\varepsilon_j)|$, respectively.

The sets $\Phi_c^i(\varepsilon_i), \Phi_u^j(\varepsilon_j)$ and $\Phi_d^j(\varepsilon_j)$ satisfy the following conditions:

1. $w_u^j + w_d^j + w_c^j = |C_j|$;
2. $\Phi_c^j(\varepsilon_i) \cup \Phi_u^j(\varepsilon_j) \cup \Phi_d^j(\varepsilon_j) = C_j$;
3. $\Phi_c^j(\varepsilon_i) \cap \Phi_u^j(\varepsilon_j) = \emptyset$, $\Phi_c^j(\varepsilon_i) \cap \Phi_d^j(\varepsilon_j) = \emptyset$, $\Phi_u^j(\varepsilon_j) \cap \Phi_d^j(\varepsilon_j) = \emptyset$.

Application of the splitting procedure to the cluster C_j depends on the values of w_u^j, w_d^j and w_c^j. If

$$w_c^j \ge \max\{w_d^j, w_c^j\} \quad (6)$$

then data points are dense around the cluster center and we do not split such a cluster. If

$$w_c^j < \max\{w_d^j, w_c^j\} \quad (7)$$

then we split this cluster into two new ones. In order to do so we define the following two subsets of $\Phi_c^i(\varepsilon_i)$:

$$\Phi_{cu}^j(\varepsilon_j) = \left\{x_i \in \Phi_c^i(\varepsilon_i) | d(x_i, c_j) \le \varepsilon_j, \ 0 \le \theta_{i,j} \le \frac{\pi}{2}\right\}, \quad (8)$$

and

$$\Phi^j_{cd}(\varepsilon_j) = \left\{ x_i \in \Phi^i_c(\varepsilon_i) | d(x_i, c_j) \le \varepsilon_j, \ \frac{\pi}{2} \le \theta_{i,j} \le \pi \right\}. \tag{9}$$

Then the cluster C_j is split into two new clusters as follows:

$$C^*_j = \{\Phi^j_u(\varepsilon_j) \cup \Phi^j_{cu}(\varepsilon_j)\}, \tag{10}$$

with the center

$$c^*_j = \frac{1}{|C^*_j|} \sum_{x_i \in C^*_j} x_i, \tag{11}$$

and

$$C^*_{j'} = \{\Phi^j_d(\varepsilon_j) \cup \Phi^j_{cd}(\varepsilon_j)\}. \tag{12}$$

with the center

$$c^*_{j'} = \frac{1}{|C^*_{j'}|} \sum_{x_i \in C^*_{j'}} x_i. \tag{13}$$

Thus, the splitting algorithm can be summarized as follows:

Algorithm 1 Splitting algorithm
Step 0. Input: A collection of k clusters $\Omega = \{C_1, ..., C_k\}$, and the ratio $\beta \in (0, 1)$.
Step 1. Select cluster $C_j \in \Omega$ and calculate its center c_j.
Step 2. Calculate $d(x_i, c_j)$ and also $\theta_{i,j}$ using (3) for all data point $x_i \in C_j$.
Step 3. For each cluster C_j calculate sets $\Phi^j_c(\varepsilon_i), \Phi^j_u(\varepsilon_j), \Phi^j_d(\varepsilon_j)$ using (2), (4) and (5), respectively.
Step 4. If (6) is satisfied then go to Step 6, otherwise go to Step 5.
Step 5. Split the cluster C_j into two new clusters C^*_j and $C^*_{j'}$ using (10) and (12), respectively. Update Ω and set $k := k + 1$.
Step 6. If all clusters $C_j, \ j = 1, \ldots, k$ are visited terminate, otherwise go to Step 2.

2.2 Merging

Assume that the collection of k clusters, $\Omega = \{C_1, ..., C_k\}$, is given. It may happen that (also after applying the splitting algorithm) some clusters are not well separated. In this subsection we design an algorithm to merge clusters which are well separated from each other.

According to the definition of well separated clusters two clusters $C_j, C_p \in \Omega$ should be merged if

$$d(c_j, c_p) - (r_j + r_p) < 0. \tag{14}$$

These two clusters are merged into one cluster as follows:

$$C^*_j = C_j \cup C_p, \tag{15}$$

with the center

$$c^*_j = \frac{1}{|C^*_j|} \sum_{x_i \in C^*_j} x_i. \tag{16}$$

For the cluster C^*_j we use only the index j meaning that the cluster C_p joins the cluster C_j. Then the merging algorithm can be summarized as follows:

Algorithm 2 Merging algorithm
Step 0. Input: A collection of k clusters $\Omega = \{C_1, ..., C_k\}$.
Step 1. Select cluster $C_j \in \Omega$ and calculate its center c_j.
Step 2. Select cluster $C_p \in \Omega$ and calculate its center c_p, where $j \neq p$.
Step 3. If the condition (14) is satisfied then go to Step 4, otherwise go to Step 6.
Step 4. Merge clusters C_j and C_p using (15). Update the set Ω and set $k := k - 1$.
Step 5. If all cluster $C_p, \ p = 1, \cdots, k$ and $j \neq p$ are visited go to Step 6, otherwise go to Step 2.
Step 6. If all clusters $C_j \in \Omega$ are visited terminate, otherwise go to Step 1.

One can see that Algorithm 1 and 2 are complementary. In other words, to have stable cluster centers these algorithms should be applied iteratively until the cluster centers become stable. The stability can be checked by monitoring the value of (1) until satisfying the strictly decreasing value or the maximum number of iteration can be predefined in advance. In this paper we use the second criterion and the split and merge algorithm is presented as follows.

Algorithm 3 Split and Merge algorithm
Step 0. Input: A collection of k clusters $\Omega = \{C_1, ..., C_k\}$, the maximum number of iterations $\gamma_{max} > 0$ and the ratio $\beta \in (0, 1)$. Set $i := 0$.
Step 1. Set $i := i + 1$.
Step 2. Apply Algorithm 1 to the collection of clusters Ω. This algorithm will generate a new collection of clusters Ω.
Step 3. Apply Algorithm 2 to the collection of clusters Ω.
Step 4. If $i > \gamma_{max}$ terminate, otherwise go to Step 1.

3 Self Organizing Maps

The SOM is an unsupervised neural network (Kohonen 2001) that usually contains a 2-Dim array of neurons $\Psi = \{w_1, \cdots, w_q\}$. Assume that we are given the set of m input data vectors $A = \{x_1, \cdots, x_m\}$ where $x_i \in \mathbb{R}^n, \ i = 1, \cdots, m$. In the SOM a weight $w_j \in \mathbb{R}^n$ is associated with the neuron $j, \ j = 1, \cdots, q$. For given $j \in \{1, \ldots, q\}$ define the following set:

$$S_j = \{x_k : d(x_k, w_j) < d(x_k, w_l), \ l \neq j, \ l = 1, \ldots, q\} \tag{17}$$

where

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

is the Euclidean distance.

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

$$S_j = S_j \cup x_i.$$

The set of neighborhood weights $N_c = \{w_l : p(c, l) \le r, \ l \neq c\}$ 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. Furthermore, $p(c,l) \in \mathbb{N}$ and $0 < p(c,l) \leq r$. The aim in this paper is to solve the following problem:

$$\text{minimize} \quad E = \frac{1}{m} \sum_{i=1}^{m} \|x_i - w_c\|, \tag{18}$$

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 4 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 iteration counter $\tau := 0$.
Step 2. Select data $x_i, i = 1, \ldots, m$ and find its closest neuron $c$, that is
    $c := \underset{j=1,\ldots,q}{\text{argmin}} \|x_i - w_j\|.$   (19)
Step 3. Update the set of neighborhood neurons $w_j \in N_c$ using the following equation:
    $w_j := w_j + \alpha(\tau)h(\tau)(x_i - w_j).$   (20)
(Here $h$ is a neighborhood function and $\alpha(\tau)$ is a learning rate at the iteration $\tau$.)
Step 4. If all input data are presented to the network go to Step 5, otherwise go to Step 2.
Step 5. Calculate $E_\tau$ using (18). If $\tau > T$ terminate, otherwise set $\tau := \tau + 1$ and go to Step 2.
```

The neighborhood function h in Step 3 of Algorithm 4 plays an important role in the SOM. Usually h is a decreasing exponential function of τ. The learning rate α is a decreasing linear function of τ and σ reduces the width of the neighborhood function h as $\tau \to T$.

3.1 SOM Initialization Algorithm

Usually the set of SOM neurons $\Psi = \{w_1, \cdots, w_q\}$ are initialized randomly (Kohonen 2001). This leads the network to converge only to local solutions of Problem (18). Furthermore the SOM suffers from slow convergence. In other words, the number of iterations to learn the input data become large and the neurons may not learn some data points correctly. In this section we present a new algorithm based on Algorithm 1 and 2 to initialize the neurons of the SOM and then define a modified topology of neurons at the initial points.

Algorithm 5 SOM initialization algorithm

```
Step 0 (Initialization). A set of $m$ input data vectors $A = \{x_1, \cdots, x_m\}$. Set $\Psi := \emptyset$ .
Step 1. Calculate the center $c^*$ of the set $A$, set $w_1 := c^*$ and
    $\Psi := \Psi \cup w_1.$
Step 2. Apply Algorithm 3 on $\Psi$. This algorithm will generate a new set $\Psi$ of neurons.
Step 3. Set the final $\Psi$ as initial neurons of the SOM.
```

Algorithm 5 ensures that the initial neurons are located in distinct high density area of the input data space which is found by Algorithm 1. Algorithm 2 guarantees that initial neurons are not close to each other.

3.2 SOM with Modified Topology

We have the set of initial neurons $\Psi = \{w_1, \cdots, w_{\hat{q}}\}$ applying the SOM initialization algorithm. We generate a set of e number of neurons $u_z \in \mathbb{R}^n,\ z = 1, \ldots, e$ using each individual neuron $w_i,\ i = 1, \cdots, \hat{q}$ as follows:

$$g_i = \{u_z | w_i^t - \lambda\varepsilon_i \leq u_z^t \leq w_i^t + \lambda\varepsilon_i\}, \tag{21}$$

where $t = 1, \ldots, n,\ z = 1, \ldots, e$ and $\lambda \in \mathbb{R},\ \lambda > 1$. One can see that all the neurons in the set g_i are close to w_i to cover up the dense area which is centered by neuron w_i. The use of such neurons allows to decrease the quantization error (18). In Problem (18) the local solution is obtained while none of the activated neurons are far from its mapped data points. Therefore, the set of neurons defined by (21) guarantees that the SOM learning process escapes from local solutions of Problem (18) and converges to the global ones.

Usually in the SOM topology, all neighborhood neurons are connected to each other in order to spread the adaption to adjacent neurons. For each pair $w_i, w_j, i, j = 1, \ldots, \hat{q},\ i \neq j$ we define the following integer number:

$$\hat{r}_{ij} = \left\lceil \frac{d(w_i, w_j)}{\varepsilon_i + \varepsilon_j} \right\rceil. \tag{22}$$

Here $\lceil x \rceil$ is a smallest integer greater than or equal to x, called its ceiling. Note that the parameter $\hat{r}_{ij}$ for neurons $u_i, u_j \in g_k,\ i, j = 1, \ldots, e,\ i \neq j,\ k = 1, \cdots, \hat{q}$ is set to 1. In order to determine the connectivity of neurons we define the threshold $r_0 \geq 1$ and say that two neurons w_i, w_j are connected if $\hat{r}_{ij} \leq r_0$. The threshold r_0 is defined for the whole data set. The neurons in the sets $g_i,\ i = 1, \cdots, \bar{q}$ are connected to their parent neuron w_i and to each other as well. Then we have the following connectivity matrix for the new topology:

1. $con(i,j) \in \{0,1\},\ w_i, w_j \in \Psi.$
2. $con(i,j) \in \{0\},\ u_i \in g_k,\ u_j \in g_p,\ k \neq p.$
3. $con(i,j) \in \{1\},\ u_i \in g_k,\ u_j \in g_p,\ k = p.$
4. $con(i,j) \in \{1\},\ u_i \in g_k,\ w_j \in \Psi,\ k = j.$
5. $con(i,j) \in \{0\},\ u_i \in g_k,\ w_j \in \Psi,\ k \neq j.$

This new topology guarantees that neurons from one dense area are not connected with those from another dense area and therefore according to the equation (20) such neurons do not change each others weight.

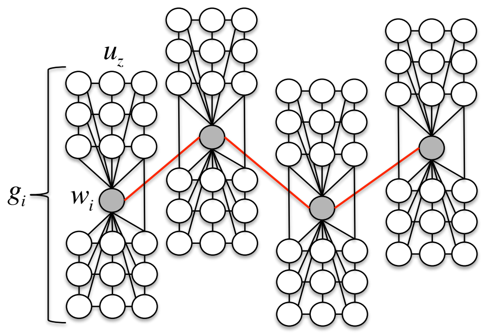

Figure 1: Topology of modified SOM.

In Figure 1 the initial neurons are in gray and the generated neurons around each initial neuron are in white color. There is no any connection between two separate set of generated neurons.

```
Algorithm 6 SOM topology generation
Step 0. Given: A set of $\Psi = \{w_1, \cdots, w_{\bar{q}}\}$ of initial neurons and a number $\lambda > 1$.
Step 1. Select a $w_i$ and generate $g_i$ using equation (21).
Step 2. Connect $w_i$ all $u_z \in g_i$.
Step 3. If all $w_i \in \Psi$ are visited go to Step 4, otherwise go to Step 1.
Step 4. Select a $w_i$ and connect it to all $w_j \in \Psi$ with $\hat{r}_{ij} < \bar{r}$.
Step 5. If all $w_i \in \Psi$ are visited terminate, otherwise go to Step 4.
```

4 The Modified SOM Algorithm and Its Implementation

In this section, we modify Algorithm 4 applying new initialization algorithm for neurons and modified SOM topology. The new algorithm can be summarized as follows.

Algorithm 7 Modified SOM algorithm

Step 0. (Initialization) Initialize the maximum number of iterations T of the network. Set the maximum number of iterations in the Splitting and Merging algorithm as γ_{max} and the value of the ratio $\beta > 0$. Set the initial value of iteration and ratio to γ_0. Set the step length β_s and set the iteration counter $\tau := 0$. A set of m input data vectors $A = \{x_1, \cdots, x_m\}$.

Step 1. (Split and Merge). Apply Algorithm 5 to A for $\gamma_0 \to \gamma_{max}$ and $\beta_0 \to \beta_{max}$ to generate the set of $\Psi = \{w_1, \cdots, w_{\hat{q}}\}$ which minimizes the function f in (1). This set is initial weights of neurons.

Step 2. (SOM topology). Apply Algorithm 6 to the set Ψ.

Step 3. Select data $x_i, i = 1, \ldots, m$ and find its closest neuron $w_c \in \{\Psi \bigcup_{i=1}^{\hat{q}} g_i\}$, that is

$$c := \underset{j=1,\ldots,(\hat{q}\cdot e)}{\operatorname{argmin}} \|x_i - w_j\|. \quad (23)$$

Step 4. Update the set of neighborhood neurons $w_j \in N_c$ using the following equation:

$$w_j := w_j + \alpha(\tau)h(\tau)(x_i - w_j), \quad (24)$$

where

$$N_c = \begin{cases} g_i \cup w_i & \text{if } w_c = u_z \in g_i, \\ g_i \cup w_i \cup \Xi & \text{if } w_c = w_i,\ w_i, w_j \in \Psi, \end{cases}$$

subject to

$$\Xi = \{w_j | d(w_i, w_j) < r',\ i \neq j\}.$$

Step 5. If all input data are presented to the network go to Step 6, otherwise go to Step 3.

Step 6. Calculate E_τ using (18). If $\tau > T$ terminate, otherwise set $\tau := \tau + 1$ and go to Step 3.

Note that the neighborhood function in equation (24) of Algorithm 7 is as follows.

$$h(\tau) = \exp\left(-\frac{\bar{r}^2}{2\sigma(\tau)^2}\right), \quad (25)$$

subject to

$$\sigma(\tau) = \eta\frac{T - \tau}{\tau},\ \eta \in \mathbb{R}, \quad (26)$$

and usually $\eta \geq 1$.

One can see that the Step 3 to 6 in Algorithm 7 is similar to the basic SOM. The only exception is in Step 4 where the set N_c is defined in order to improve the approximation of the global solution to the vector quantization problem.

4.1 Implementation of Algorithm 7

In Algorithm 4, weight vectors $w_j, j = 1, \cdots, q$ are initialized randomly. The maximum number of iterations T is set between 20 and 40 for small to large data set, respectively. Although, for large data sets more time is required to obtain stable network over input data. The topology of SOM network is rectangular (Kohonen 2001) with 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 map r is set to 2 for small and 4 for large number of neurons (see Table 1).

Table 1: Initialization of SOM parameters in Algorithm 4.

Data sets	Input Size	SOM Dim.	r	T
Small	$(\|A\| < 10^3)$	10×10	2	20
Medium	$(10^3 < \|A\| < 10^4)$	15×15	3	30
Large	$(10^4 < \|A\| < 0.5 \cdot 10^5)$	20×20	4	40
	$(\|A\| > 0.5 \cdot 10^5)$	25×25	3	20

As it is presented in Table 1, the number of neurons, maximum iteration number T and r are chosen incrementally in order to be applicable to larger input data sets. The exception is for the data set with size $|A| > 0.5 \cdot 10^5$, where r and T are smaller comparing to other large data sets to decrease the computational complexity.

In Step 1 of Algorithm 7, we set values of T same as in Table 1, $2 \leq \gamma \leq 6$ and $0.05 \leq \beta \leq 0.6$ with step length β_s to 0.05 for Algorithm 3. In Step 3, the parameter λ in Algorithm 6 using equation (21) is set to 1.5 for small and medium size data sets whereas this value is set to 2.5 for large ones. In Step 4, the Algorithm 6 generates 9 neurons ($|g_i|$), i.e. e is set to 9, around all neurons $w_i \in \Psi$. Therefore the total number of neurons is $|\Psi| \times |g_i| + |\Psi|$. It should be noted that for all datasets the parameter $\bar{r}$ in (22) is set to 3. Finally, the parameter η in (26) is set to 1 for all data sets.

5 Numerical Results

To demonstrate the effectiveness of the proposed algorithm, numerical experiments were carried out using a number of real-world data sets. Algorithm 7 was coded in NetBeans IDE under Java platform and tested on a MAC OSX with 2.7GHz core i7 CPU and 10GB of RAM. 4 data sets, one small (Iris), one medium size (Image Segmentation), one large (Gamma Telescope) and one very large (NE) were used in experiments. A brief description of data sets is presented in Table 2, more details can be found in (Bache & Lichman 2013, Theodoridis 1996, Reinelt 1991).

Table 2: Brief description of the data sets

Data sets	Number of instances	Number of attributes
Fisher's Iris Plant	150	4
Image Segmentation	2310	19
Gamma Telescope	19020	10
NE	50000	2

The results obtained by the Split and Merge algorithm, which is in Step 2 of Algorithm 7 is presented in Table 3. To define the improvement E_{im} obtained by the proposed algorithm, we use the following formula:

$$E_{im} = \frac{E_{SOM} - E}{E_{SOM}} \cdot 100\%. \qquad (27)$$

where E is the value obtained by the Modified SOM.

Table 3: The results of the Split and Merge algorithm

Data sets	β^*	γ^*	$\lvert\Psi\rvert$	f_{min}	t
Fisher's Iris Plant	0.05	2	23	4.95×10^1	0.02
Image Segmentation	0.10	2	62	3.17×10^7	0.14
Gamma Telescope	0.05	2	87	2.01×10^8	6.61
NE	0.20	2	61	3.66×10^2	4.57

The values of quantization error using equation (18) for different iterations and different data sets are presented in Tables 4. From these results one can see that the Modified SOM outperforms SOM in all data sets. The Modified SOM reduced the value of problem (18) significantly in Image Segmentation and Iris data sets up to 38.28% and 24.82%, respectively. On other data sets the improvement E_{im} is between 6.85% and 14.11%. One can see that the Modified SOM starts with a small value of E comparing to the SOM. This is due to optimized initialization of the Modified SOM algorithm. The computational effort used by the Modified SOM is much less than that of the SOM in all data sets. The Split and Merge algorithm that initializes the Modified SOM is very efficient and it is not time consuming. The new initialization algorithm which is based on the Split and Merge algorithm speeds up the convergence of the Modified SOM and makes it less time consuming than the SOM. The maximum time reduction by the Modified SOM was achieved in Gamma Telescope data sets. On the other hand the minimum computational time reduction is on very large data set: NE data set.

In Figure 2 the values of E obtained by the Modified SOM is compared with those obtained by the SOM on Iris data set. One can see that on Iris data set the Modified SOM starts with a value of E close to global one and converge to the optimal value within the given number of iterations. Since the SOM is initialized randomly, it takes more time to converge. In Figure 3 the CPU time required by the SOM and Modified SOM on Gamma Telescope is presented. One can see that the Modified SOM requires more CPU time at the early iterations due to running the Split and Merge algorithm for initialization. Once the Modified SOM initialized, the convergence is much faster than the SOM which is initialized randomly.

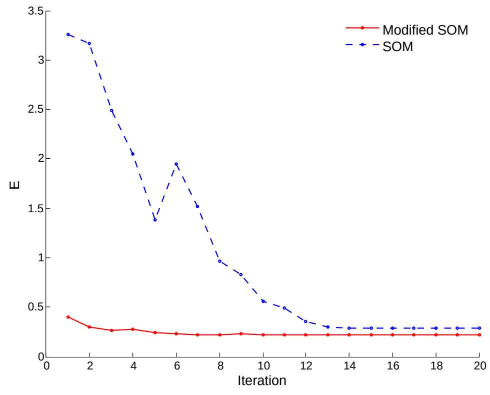

Figure 2: SOM vs Modified SOM using E values (Iris dataset).

Table 4: Results for all data sets

iter	E	t	E	t
	Iris			
	SOM		Modified SOM	
2	3.17E+00	0.06	2.96E-01	0.02
4	2.05E+00	0.08	2.76E-01	0.03
6	1.95E+00	0.09	2.29E-01	0.03
8	9.70E-01	0.09	2.23E-01	0.05
10	5.56E-01	0.11	2.22E-01	0.05
12	3.51E-01	0.12	2.22E-01	0.05
14	2.88E-01	0.12	2.22E-01	0.06
16	2.86E-01	0.14	2.22E-01	0.06
20	2.86E-01	0.16	2.15E-01	0.08
	Image Segmentation			
	SOM		Modified SOM	
2	1.82E+07	0.73	1.01E+02	0.40
4	2.41E+03	1.28	8.01E+01	0.62
6	1.84E+02	1.75	2.90E+01	0.84
10	1.42E+02	2.74	1.89E+01	1.26
14	1.02E+02	3.65	1.75E+01	1.68
18	4.55E+01	4.54	1.74E+01	2.09
22	2.69E+01	5.40	1.74E+01	2.51
25	2.69E+01	6.04	1.75E+01	2.84
30	2.69E+01	7.00	1.66E+01	3.35
	Gamma Telescope			
	SOM		Modified SOM	
2	3.93E+20	10.97	8.28E+01	8.39
4	5.11E+12	21.32	4.08E+01	10.00
6	2.22E+03	31.73	3.46E+01	11.62
10	2.21E+02	53.17	3.15E+01	14.84
18	1.30E+02	95.26	3.02E+01	21.23
22	7.56E+01	116.35	3.00E+01	24.43
26	4.53E+01	137.56	2.99E+01	27.63
30	3.34E+01	158.70	2.99E+01	30.81
40	3.33E+01	210.52	2.86E+01	38.44
	NE			
	SOM		Modified SOM	
2	2.02E+07	5.18	3.28E+07	6.47
4	3.32E-01	9.86	8.56E+32	8.18
6	3.01E-01	14.56	3.88E-02	9.89
8	2.70E-01	19.25	2.15E-02	11.54
10	2.56E-01	23.93	1.23E-02	13.20
12	1.94E-01	28.52	1.12E-02	14.88
14	5.41E-02	32.99	1.12E-02	16.55
16	1.16E-02	37.46	1.12E-02	18.24
20	1.12E-02	46.30	1.03E-02	21.51

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 (Arous 2010, Ayadi et al. 2012):

$$\xi = \sum_{x_i \in A} \sum_{w_j \in \Psi} h_{cj} \|x_i - w_j\|^2, \quad j \neq c, \qquad (28)$$

where c is the BMU of x_i and h_{cj} is the neighborhood function of neurons c and j defined by Equation (25).

Table 5 presents the distortion measure (28) and number of active neurons n_{act} for all data sets. One can see that the distortion error ξ obtained by Modified SOM is less than that obtained by the SOM in all data sets. This is due to the topology of the Modified SOM where the neurons from different dense areas are not connected. This prevents deterioration of the network from its optimal value of ξ and E simultaneously.

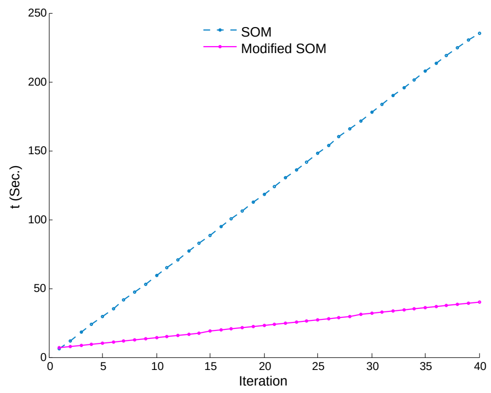

Figure 3: SOM vs Modified SOM using CPU time (Gamma Telescope dataset).

Table 5: Results of distortion measure on all data sets

	SOM		Modified SOM	
Dataset	ξ	n_{act}	ξ	n_{act}
Iris	1.25×10^{-6}	69	3.62×10^{-7}	92
Image Seg.	3.26×10^{-4}	210	8.73×10^{-5}	490
Gamma Telescope	2.35×10^{-5}	400	1.39×10^{-5}	759
NE	1.66×10^{-1}	375	3.11×10^{-2}	610

6 Conclusion

The aim in this paper was to propose an initialization algorithm and a new topology for the Modified Self Organizing Maps which restrict the neighborhood adaptations to only those neurons that are not in different dense areas. We introduced the Split and Merge algorithm to generate such neurons. This algorithm is a part of the initialization algorithm in the Modified SOM and the numerical experiments show that the initialization algorithm generates neurons close to the optimal solution. Consequently we presented a topology for the SOM to generate neurons in high dense areas of input data space and do not connect neurons from different dense areas. The experiments show that this restriction reduces the quantization and distortion errors. Numerical results demonstrate the superiority of the proposed algorithm over the SOM in the sense of accuracy. These results also show that the Modified SOM converges much faster than the SOM and in all cases the proposed algorithm requires less computational time.

References

Alahakoon, D., Halgamuge, S. K. & Srinivasan, B. (2000), 'Dynamic self-organizing maps with controlled growth for knowledge discovery', *IEEE transactions on neural networks* **11**(3), 601–14.

Appiah, K., Hunter, A., Dickinson, P. & Meng, H. (2012), 'Implementation and Applications of Tri-State Self-Organizing Maps on FPGA', *IEEE Transactions on Circuits and Systems for Video Technology* **22**(8), 1150–1160.

Arous, N. (2010), 'On the Search of Organization Measures for a Kohonen Map Case Study: Speech Signal Recognition', *International Journal of Digital Content Technology and its Applications* **4**(3), 75–84.

Ayadi, T., Hamdani, T. & Alimi, A. (2012), 'Migsom: Multilevel interior growing self-organizing maps for high dimensional data clustering', *Neural Processing Letters* **36**, 235–256.

Bache, K. & Lichman, M. (2013), 'UCI machine learning repository', http://archive.ics.uci.edu/ml.

Berkhin, P. (2006), 'A survey of clustering data mining techniques', *Grouping Multidimensional Data* pp. 1–56.

Bogdan, M. & Rosenstiel, W. (2001), Detection of cluster in self-organizing maps for controlling a prostheses using nerve signals, *in* 'ESANN'01', pp. 131–136.

Brugger, D., Bogdan, M. & Rosenstiel, W. (2008), 'Automatic cluster detection in Kohonen's SOM', *IEEE transactions on neural networks* **19**(3), 442–59.

Cheng, S.-S., Fu, H.-C. & Wang, H.-M. (2009), 'Model-based clustering by probabilistic self-organizing maps', *IEEE transactions on neural networks* **20**(5), 805–26.

Chi, S.-C. & Yang, C. C. (2006), Integration of ant colony som and k-means for clustering analysis, *in* 'Proceedings of the 10th international conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part I', KES'06, Springer-Verlag, Berlin, Heidelberg, pp. 1–8.

Chi, S.-C. & Yang, C.-C. (2008), 'A Two-stage Clustering Method Combining Ant Colony SOM and K-means', *Journal of Information Science and Engineering* **24**(5), 1445–1460.

Cottrell, M., Gaubert, P., Eloy, C., Francois, D., Hallaux, G., Lacaille, J. & Verleysen, M. (2009), 'Fault Prediction in Aircraft Engines Using Self-Organizing Maps', *Security* pp. 37–44.

Fiannaca, A., Di Fatta, G., Rizzo, R., Urso, A. & Gaglio, S. (2011), 'Simulated annealing technique for fast learning of som networks', *Neural Computing and Applications* pp. 1–11.

Fiannaca, A., Fatta, G., Gaglio, S., Rizzo, R. & Urso, A. (2007), Improved som learning using simulated annealing, *in* 'Artificial Neural Networks AI ICANN 2007', Vol. 4668 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 279–288.

Ghaseminezhad, M. H. & Karami, A. (2011), 'A novel self-organizing map (SOM) neural network for discrete groups of data clustering', *Applied Soft Computing* **11**(4), 3771–3778.

Goncalves, M. L., de Andrade Netto, M. L. & Zullo, J. (1998), A neural architecture for the classification of remote sensing imagery with advanced learning algorithms, *in* 'Neural Networks for Signal Processing VIII, 1998. Proceedings of the 1998 IEEE Signal Processing Society Workshop', pp. 577–586.

Gorgonio, F. L. & Costa, J. A. F. (2008), Combining Parallel Self-Organizing Maps and K-Means to Cluster Distributed Data, *in* '2008 11th IEEE International Conference on Computational Science and Engineering - Workshops', IEEE, pp. 53–58.

Haese, K. (1998), 'Self-organizing feature maps with self-adjusting learning parameters', *IEEE transactions on neural networks* **9**(6), 1270–8.

Jain, A. K., Murty, M. N. & Flynn, P. J. (1999), 'Data clustering: a review', *ACM Computing Surveys* **31**(3), 264–323.

Kohonen, T. (2001), *Self-Organizing Maps*, Springer Series in Information Sciences, Springer.

Lapidot, I., Guterman, H. & Cohen, a. (2002), 'Unsupervised speaker recognition based on competition between self-organizing maps', *IEEE transactions on neural networks* **13**(4), 877–87.

Reinelt, G. (1991), 'TSPLIB- a Traveling Salesman Problem Library', *ORSA Journal of Computing* **3**(4), 376–384.

Shah-Hosseini, H. & Safabakhsh, R. (2003), 'TASOM: a new time adaptive self-organizing map', *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society* **33**(2), 271–82.

Shi, J. & Malik, J. (2000), 'Normalized cuts and image segmentation', *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905.

Tasdemir, K., Milenov, P. & Tapsall, B. (2011), 'Topology-Based Hierarchical Clustering of Self-Organizing Maps', *IEEE Transactions on Neural Networks* **22**(3), 474–485.

Theodoridis, Y. (1996), 'Spatial datasetsan unofficial collection', http:// www.dias.cti.gr/ ~ytheod/research/ datasets/ spatial.html. Accessed 2013.

Vesanto, J. & Alhoniemi, R. (2000), 'Clustering of the self-organizing map', *IEEE Transactions on Neural Networks* **11**(3), 587–600.

Wong, M., Jack, L. & a.K. Nandi (2006), 'Modified self-organising map for automated novelty detection applied to vibration signal monitoring', *Mechanical Systems and Signal Processing* **20**(3), 593–610.

Xu, P., Chang, C.-H. & Paplinski, A. (2005), 'Self-organizing topological tree for online vector quantization and data clustering.', *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society* **35**(3), 515–26.

Yang, L., Ouyang, Z. & Shi, Y. (2012), 'A Modified Clustering Method Based on Self-Organizing Maps and Its Applications', *Procedia Computer Science* **9**, 1371–1379.

Yen, G. G. & Wu, Z. (2008), 'Ranked centroid projection: a data visualization approach with self-organizing maps', *IEEE transactions on neural networks* **19**(2), 245–59.

Zheng, Y. & Greenleaf, J. F. (1996), 'The Effect of Concave and Convex Weight Adjustments on Self-organizing Maps', *IEEE transactions on neural networks* **7**(1), 87–96.