# Clustering Hierarchical Data Using SOM Neural Network

Le Anh Tu[1], Nguyen Quang Hoan[2], and Le Son Thai[1]

[1] Thai Nguyen University of Information and Communication Technology
[2] Posts and Telecommunications Institute of Technology
{anhtucntt,lesonthai}@gmail.com, quanghoanptit@yahoo.com.vn

**Abstract.** This paper proposes a solution for clustering hierarchical data using SOM neural network. The training process that combines data-partition and network-partition allows forming an automated hierarchical tree structure representing the clustering process more detailed from the root node to the leaf node. In which the root node and intermediate nodes act as the orientation for data distribution, and the leaf nodes represent real clusters of data. This training tree structure allows programming parallel processing to speed up network training. In addition, applying the trained network could be more efficient because the search process performed on the tree structure.

**Keywords:** neural network, kohonen, self organizing map, clustering data, data mining.

## 1 Introduction

Teuvo Kohonen developed SOM neural network in the 80s. This is the feed-forward neural network using of the competitive learning, unsupervised (or self-organizing), allowing mapping data from a multi-dimensional space to two–dimensional space, thereby creating a feature map of the data [7]. SOM is an appropriate tool to solve the problem of data clustering, an important preprocessing step in data mining [1]. When applying the SOM neural network for clustering data, it consists of three phases:

- Phase 1: Training the network with sample data set.
- Phase 2: Applying the trained network.
- Phase 3: Visualize network (usually using the gray level map, U-Matrix, to present data distribution image [6])

However, the standard SOM model requires computation time relatively long (both training time and applying the trained network). The main reason is Kohonen matrix must be large enough to describe all features of the data (usually thousands of neurons), while with each input vector, the network will perform sequential search across the entire Kohonen matrix. Currently, there have been many studies on this issue, for example: Batch SOM algorithm [2] speeds up training by deferring the updates of weights to the end of a learning epoch (after browsing all the training samples); Tree-Structured SOM algorithm (TS-SOM) [4] builds a training tree structure, in which each node on the tree is a neuron. The numbers of sub-nodes of each node are equal and increase according to exponential function. The nodes at the same level forms a layer, upper layer trains the underlying one. For example, the first layer has four nodes (neuron), each node has 4 sub-nodes, so the second layer has 16 nodes, and the third layer has 64 nodes… The training time is faster because finding the winning neuron (Best Matching Unit - BMU) at each step of the training process is performed following the branches of the tree. Alfredo F. Costa proposed a tree structure representing the clusters, in which each node of the tree is a Kohonen map [3]. Each node is fully trained as standard SOM, then builds the U-Matrix and uses this information to form the sub-nodes. This solution is not effective because the fact that at the root node has finished training and U-Matrix of the root node completely visualized input data (step 3 above). However, data analysis (data-partition) following tree structure is a good idea to program processing in parallel.

After studying the standard model and some improved models of SOM, we found that there are two issues needed to be considered in order to improve the efficiency of SOM neural network when applied to large data sets. Firstly, reducing the size of Kohonen matrix but still fully describing the features of data. Secondly, having mechanisms to search and process data in parallel on different parts of the network.

This paper proposes a hierarchical training technique based on threshold T, non-similar on the features of the data, which allows segmenting data (data-partition) according to the hierarchy tree structure. Each node of the tree is a Kohonen matrix, in which the root node and the intermediate nodes (branch nodes) only serve to orient the data segmenting, leaf nodes represent real clusters of data. Thus, the size of the original Kohonen matrix will reduce, and the size of sub Kohonen will reduce gradually after each branching. At each node, threshold T clusters the neurons in the process of training. Each cluster in the parent node is the basis for establishing a child node. All data vectors that belong to a cluster of the parent node will be used to train the child node corresponding to that cluster. At the next training level, when threshold T declines $\alpha$ value, repeat the training and develop the children nodes for all the current leaf nodes. Training process will stop when the threshold T reduces to a minimum value $\varepsilon$.

The rest of the paper include: Section 2 presents the standard SOM model and U-Matrix, Section 3 presents the principle of neural clustering in the training process based on the threshold T, Section 4 provides a hierarchical tree training architecture, Section 5 presents the experimental results and the final section concludes and comments on the presented solution in the articles.

## 2 SOM Neural Network and U-Matrix

SOM neural network includes an input layer and a output layer called Kohonen layer. Kohonen layer is often organized as a two dimensional matrix of neurons. Each unit *i* (neurons) in the Kohonen layer is attached a weight vector $w_i=[w_{i1}, w_{i2},\ldots ,w_{in}]$, with *n* is the input vector size, $w_{ij}$ is the weight of neuron *i* corresponding to the input *j*. Network training process is repeated several times, at iteration *t,* three steps are done:

- Step 1- find the BMU: randomly select an input $v$ from the data set, find $b$ neuron that has the smallest *dist* distance function in the Kohonen matrix (frequently use functions Euclidian, Manhattan or Vector Dot Product). $b$ neuron is called BMU.

$$dist = \| v - w_b \| = \min_i \{ \| v - m_i \| \} \tag{1}$$

- Step 2- determine the neighborhood radius of the BMU: $\sigma(t) = \sigma_0 \exp\left[-\frac{t}{\lambda}\right]$ is interpolation function of radius (decreasing as the numbers of iterations), where $\sigma_0$ is the initial radius; time constant $\lambda = \frac{K}{\log(\sigma_0)}$, where $K$ is the total number of iterations.

- Step 3- update the weights of the neurons in the neighborhood radius of the BMU in a trend closer to the input vector $v$ :

$$w_i(t+1) = w_i(t) + \alpha(t) h_{bi}(t) [v - w_i(t)] \tag{2}$$

*where* $\alpha(t) = \alpha_0 \exp\left[-\frac{t}{\lambda}\right]$ is the learning speed interpolation function, with $\alpha_0$ is the initial value of the learning rate; $h_{bi}(t)$ is the interpolation function over learning times, shows the effect of distance to the learning process, can be calculated by the formula $h_{bi}(t) = \exp\left[-\frac{\| r_b - r_i \|^2}{2\sigma^2(t)}\right]$ where $r_b$ and $r_i$ position of the neuron $b$ and neuron $i$ in the Kohonen matrix.

To observe the feature map of the data, visual matrix using gray level (U-Matrix) is often used (Figure 1). The bright region in the matrix represents the clusters (the distance between the weight vectors of neurons is small), dark areas (large distance) which is used to separate areas between clusters. Agglomerative algorithm [8] is used to determine the boundary between the clusters.

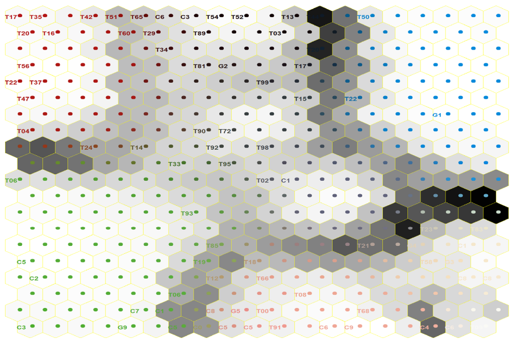


**Fig. 1.** Illustration of U-Matrix

Indeed, each neuron in the Kohonen layer represents one or several data samples, therefore the clustering data is clustering the neurons. The paper [5] we proposed improved SOM algorithm that allowed us to identify data clusters (clusters of neurons) and labeled on each neuron on Kohonen matrix in the process of network training without a visual matrix. Applying standard SOM to cluster data, it has computing complexity is O(n$^4$), whereas using improved SOM, it is O(n$^3$). The principles identifying clusters will be presented in the next section.

## 3 Principles of Neuron Cluster in the Training Process

For each neuron in the neighborhood radius of the BMU (including BMU), besides updating the weight vector (step 3), we labeled neurons with clustered index based on the following four principles:

- *Principles of cluster formation:* if BMU does not yet belong to any cluster, a new cluster is formed, which include BMU and neurons in the neighborhood radius of the BMU if these neurons are also not members of any clusters. Figure 2, BMU and $N_1$ do not belong to any cluster, $N_1$ is within the impact of BMU, therefore forming a new cluster of BMU and $N_1$.
- *Principle of neurons dispute:* A neuron M is deemed to be a dispute if it was attached to a cluster $G_1$ and located in the radius of the impact of the BMU. Let $k_1$ be the feature difference between the $M$ and $G_1$; $k_2$ is the feature difference between the $M$ and BMU. If $k_1 < k_2$, $M$ belongs to cluster $G_1$, otherwise, $M$ belongs to the cluster of the BMU. Figure 2, $N_2$ is of cluster 1 but located within a radius of the impact of the BMU, so disputes occur between cluster of the BMU and cluster 1.
- *Principle of cluster splitting:* If BMU belongs to a certain cluster, check splitting condition: let k be the different feature value of BMU from its cluster. If $k>=T$ (splitting threshold), BMU will form a new cluster, similarly to principles of cluster formation; otherwise, BMU will extend cluster

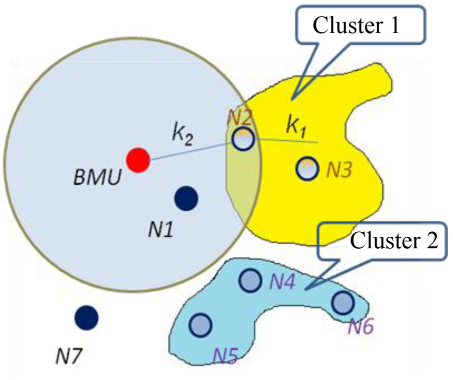


**Fig. 2.** Illustration of the principle of the cluster formation and neurons dispute

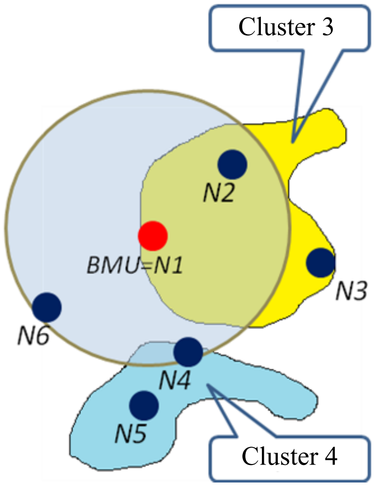


**Fig. 3.** Illustration of the principle of the cluster formation and neurons dispute

which contain it using the principle of extension. Figure 3, the BMU is the neuron of cluster 3, splitting condition should be considered.

- *Principle of cluster extension:* If BMU is a member of a cluster, and splitting condition is false, it will consolidate its cluster by admitting more neurons in its radius impact as the principle of cluster formation. Figure 3, if the BMU does not satisfy the splitting condition (but still in cluster 3), it will admit $N_6$ to cluster 3 and $N_4$ will be disputed with the cluster 4.

The four principles above did not entirely alter the data feature results of network, because in fact they only assigned cluster index for each neuron. Therefore, the quality of the Kohonen map is constant. However, the numbers of clusters formed depend on the splitting threshold *T*. If *T* is large, the numbers of clusters formed is small. Otherwise, they are large. The next section presents the training hierarchical tree structure based on the parameter *T*.

## 4 The Training Hierarchical Tree Structure

The training hierarchical tree structure comes from the idea of dividing the data into large clusters, from each large cluster further divided into smaller clusters, and from the small cluster is further subdivided into smaller clusters. In which, each node of the tree is designed as a SOM neural network, allowing to cluster data more detail gradually from the root node to the leaf node. A parent node after having been trained will be used to divide its data into subsets. Each subset of data continues to be used to train the corresponding child nodes generated. This division process is based on the threshold *T*, with $\varepsilon \le T < \max_{i,j}\left\{\| v_i - v_j \|\right\}$, where $\varepsilon$ is the limit on the value of non-similar elements in the same cluster, $\max_{i,j}\left\{\| v_i - v_j \|\right\}$ is the value of the largest difference between two certain elements in the entire data set.

Tree will grow and be trained on each layer. At each layer, all the leaf nodes are trained with the same threshold *T*. At lower layer, *T* decreases a value α until T=ε. Figure 4 illustrates training structure with m layers, the first layer is trained with threshold $T = T_1 < \max_{i,j}\left\{\| v_i - v_j \|\right\}$, on each layer, *T* value decreases $\alpha$ until it reachs $\varepsilon$ value.

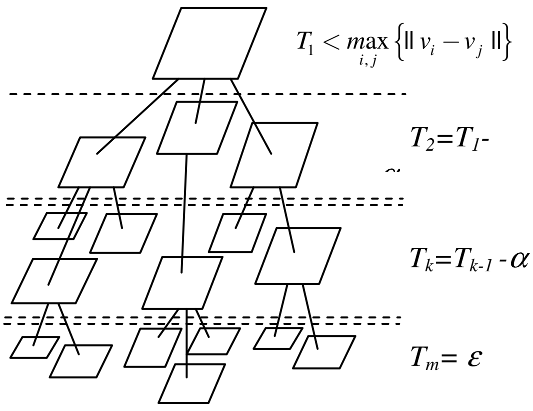


**Fig. 4.** Illustration of the training hierarchical tree structure

On the first layer, all original data are trained by only a single Kohonen whose size is *nxn*, and uses splitting threshold $T_1$. Results were k subsets of $I=\{I_1, I_2, \ldots, I_k\}$. On the second layer, each subset $I_1, I_2, \ldots, I_k$ will be trained by each corresponding Kohonen $K_{I1}, K_{I2}, \ldots, K_{Ik}$ and uses splitting threshold $T_2=T_1-\alpha$. Thus each set $I_i$ (where $i=1..k$) trained by $K_{Ii}$ was divided into subsets $I_{i1}, I_{i2}, \ldots$ On the next layers, repeat the reduction $T$ and train each child node with the corresponding subset of it, until $T=\varepsilon$.

The size of the child nodes will be reduced gradually depending on the size of the parent node and the ratio of the number of elements in the subset used to train it with the number of elements used to train the parent node.

Let $n_{child}$ be size of child node: $n_{child} = \left( \frac{|I_{child}|}{|I_{\text{parent}}|} \right)^{\beta} n_{\text{parent}}$, where |.| represents the cardinality of the set, $\beta$ is used to limit the child node size reduction in compared with the parent node size (we tested with $\beta=0.3$).

Because the role of the root node and the intermediate nodes are data oriented division, the root node size does not need to be large enough to describe all features of the data set. This significantly reduces the training time at each node. In addition, data is divided as tree structure to train in parallel on many Kohonen networks, so training time and application time of the trained network are faster than standard SOM model.

To test these evaluations, we have written the test program on a personal computer, using multithreaded programming technique combining recursion. These experimental results will be presented in the next section.

## 5 Experimental Results

To facilitate the visual observation and evaluate research results, we choose the problem of color image segment. The input data set is the pixels. Each pixel is considered as a vector of input data including three elements corresponding to three colors R, G, B. Thus, the network has three inputs and the size of the data set is the size of an image (up to hundred thousands of pixels).

We adjust the training parameters for both models training hierarchical tree and standard SOM model (applying the four principles presented in the section 3) to reach the similar result of the formation of clusters and evaluate the effectiveness of execution time.

The standard SOM model: size of Kohonen is 30x30, splitting threshold T=30.

The training hierarchical tree model: size of original Kohonen is 11x11, splitting threshold $30<=T<=120$ (with $\varepsilon=30$, $T_1=120< \max_{i,j}\{\| v_i - v_j \|\} = \sqrt{255^2+255^2+255^2} \simeq 442$), at each layer, T reduces $\alpha=20$.

We tested four different sized images, each image ran 5 times. The table below presents the test results.

In all cases, the training time and time applying trained network to cluster of hierarchical tree model are faster. The first two images with the size of about 26

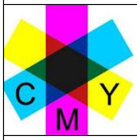
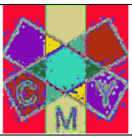
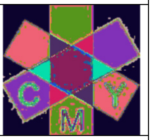
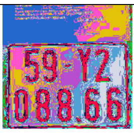
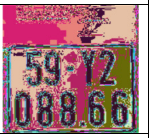
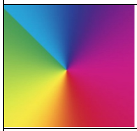
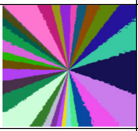
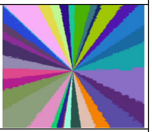
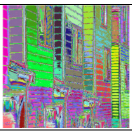
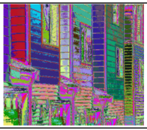
| Original image | Times | Standard SOM model | | | Hierarchical tree model | | |
|---|---|---|---|---|---|---|---|
| | | Time (ms) | | Image result/ numbers of clusters | Time (ms) | | Image result/ numbers of clusters |
| | | Training | Applying clustering | | Training | Applying clustering | |
|  | 1 | 4747.27 | 1813.10 |  | 1974.32 | 623.03 |  |
| | 2 | 4627.26 | 1897.45 | | 2012.35 | 708.25 | |
| | 3 | 4759.54 | 1854.32 | | 1987.32 | 732.12 | |
| | 4 | 4813.65 | 1864.35 | | 2103.54 | 698.65 | |
| | 5 | 4687.25 | 1824.54 | | 2014.36 | 743.25 | |
| **163X159** | **avg** | **4726.99** | **1850.75** | **111-130** | **2018.38** | **701.06** | **127-155** |
| | 1 | 4655.26 | 1592.35 |  | 1163.06 | 679.03 |  |
| | 2 | 4568.25 | 1685.32 | | 1254.32 | 684.35 | |
| | 3 | 4687.26 | 1672.32 | | 1198.32 | 687.32 | |
| | 4 | 4756.21 | 1612.32 | | 1245.35 | 702.35 | |
| | 5 | 4587.25 | 1586.21 | | 1234.15 | 624.21 | |
| **160X160** | **avg** | **4650.85** | **1629.70** | **22-29** | **1219.04** | **675.45** | **29-35** |
|  | 1 | 19665.56 | 6833.26 |  | 4025.32 | 2313.25 |  |
| | 2 | 19565.69 | 6696.21 | | 4059.23 | 2268.12 | |
| | 3 | 19615.58 | 6787.65 | | 3998.26 | 2298.32 | |
| | 4 | 19625.76 | 6681.84 | | 4065.32 | 2274.54 | |
| | 5 | 19584.84 | 6904.65 | | 4100.23 | 2289.78 | |
| **350x300** | **avg** | **19611.49** | **6780.72** | **32-36** | **4049.67** | **2288.80** | **31-36** |
| | 1 | 41307.32 | 14346.54 |  | 7738.44 | 4107.23 |  |
| | 2 | 42563.24 | 14687.25 | | 7954.32 | 4321.25 | |
| | 3 | 43124.87 | 14753.24 | | 7542.21 | 4215.36 | |
| | 4 | 40154.65 | 13968.24 | | 7652.32 | 4198.25 | |
| | 5 | 42564.65 | 14885.98 | | 7764.24 | 4253.35 | |
| **550x382** | **avg** | **41942.95** | **14528.25** | **80-110** | **7730.31** | **4219.09** | **77-109** |

**Fig. 5.** Comparison of experimental results of the two models

thousand pixels, hierarchical tree model has training time of approximately 2.3 to 3.8 times faster, and application time is about 2.5 times faster. Two next images (the numbers of pixels are over 100 thousands) hierarchical tree model has faster training time of approximately 4.8 to 5.4 times, and faster application time 3 to 3.5 times. This shows that the speed advantage of the hierarchical tree model increases when the data size is large. We have tested with variety of input images and received the similar results.

## 6 Conclusions

The training hierarchical tree structure significantly reduces the network training time by two main reasons. Firstly, the size of Kohonen matrix reduced (due to the features of the data are not represented on a Kohonen matrix, but represented on the sub Kohonen matrixes which are the leaf node of the training tree). Secondly, allowed segmenting the data to train in parallel by many SOM neural networks. In addition, the training model is represented by a tree structure so the application time of trained network to cluster data is also dramatically reduced.

In terms of the quality, data features are completely unchanged since the hierarchical tree model does not change the standard SOM algorithm but only segments data for parallel processing.

However, as a standard SOM model, proposing a suitable configuration for the network is difficult. It is needed to test several times to select the parameters (original Kohonen matrix size and scope of the threshold *T* and the reduced value of *T* after each layer) best suited to each type of data. In principle, the smaller the matrix size is, the faster the original Kohonen network runs, but if it is too small , the efficiency of the characterization data of the network will decrease like the standard SOM model; if $T_1$ is too large, this will affect the efficiency of data segmenting of the original Kohonen and Kohonens in the first layers.

## References

[1] Han, J., Kamber, M.: Data Mining - Concepts and Techniques, ch. 8. Morgan Kaufmann (2001)

[2] Silva, M.: A hybrid parallel SOM algorithm for large maps in data-mining. In: 13th Portuguese Conference on Artificial Intelligence (EPIA 2007), Workshop on Business Intelligence. IEEE, Guimaraes (2007)

[3] Costa, M.: A new tree-structured self-organizing map for data analysis. In: Proceedings of the International Joint Conference on IJCNN (2001)

[4] Laaksonen, J., Koskela, M., Oja, E.: Application of Tree Structured Self-Organizing Maps in Content-Based Image Retrieval. In: Proceedings of 9th ICANN 1999, Edinburgh, UK (1999)

[5] Tu, L.A., Hoan, N.Q.: Improving som neural network algorithm for color image clustering problem. In: Proceedings of VCCA Conference-VietNam (2011)

[6] Ultsch, A., Peter Siemon, H.: Kohonen's self-organizing feature maps for exploratory data analysis. In: Proceedings of the International Neural Network Conference (INNC 1990), pp. 305–308. Kluwer (1990)

[7] Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer (2001)

[8] https://rtmath.net/help/html/29f7cb00-39a1-4fc0-af60-52925f074edd.htm