Menemui Matematik (Discovering Mathematics)
Vol. 38, No. 3: 44-53 (2016)

The Self Organizing Map as a Tool for Cluster Analysis

Abdu Masanawa Sagir[1] and Saratha Sathasivam[2]
[1,2] *School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Gelugor, Penang*
email: [1]amsagir@yahoo.com, [2]saratha@usm.my

ABSTRACT

The Self-organizing map is among the most acceptable algorithm in the unsupervised learning technique for cluster analysis. It is an important tool used to map high–dimensional data sets onto a low–dimensional discrete lattice of neurons. This feature is used for clustering and classifying data. Clustering is the process of grouping data elements into classes or clusters so that items in each class or cluster are as similar to each other as possible. In this paper, we present an overview of self organizing map, its architecture, applications and its training algorithm. Computer simulations have been analyzed based on samples of data for clustering problems.

Keywords: Clustering, Self organizing map, Unsupervised learning algorithm

INTRODUCTION

Self-organizing maps (SOMs) are a type of artificial neural network. These maps can also be referred to as Kohonen Neural Networks, Topological Neural Networks, Self–Organizing Feature Maps or Topological Preserving Feature Maps (Lobo, 2009).

The self-organizing map was developed by its creator Finish Professor Emeritus Teuvo Kohonen. Self-organizing maps is a tool for clustering complex or vast amounts of data in a manner that is most easily understood by the human brain. Input data are grouped into clusters that are commonly used for unsupervised training. SOM has the means to show that the network can recognize or characterize inputs it has never come across before.

According to Kohonen (2014), *"the self-organizing map is a clustering method, but unlike the usual clustering methods, it is also a topography-preserving nonlinear projection mapping. On the other hand, while the other nonlinear projection mappings are also topography-preserving mappings, they do not average data, like the self-organizing map does".*

SOM has vector quantization as one of its fundamental properties. This is a data compression technique. It provides a way of representing multi-dimensional data into a much lower dimensional lattice. Usually compression of data in self-organizing maps can be classified into two types: reduction of data dimension with minimum loss of information; and reduction of data variety due to terminal composition prototypes separation – clustering and quantization of data sets (Zherebtsov and Kuperin, 2003).

Self-organizing maps operate in two modes: training and mapping. This paper will concentrate in mapping technique. Placing a vector from a data space onto the map involves finding the node with the closest (usually small distance metric – Euclidean distance, City block distance, Chebychev distance e.t.c.) weight vector to the data space vector. There are two main ways to cluster data, partition clustering and the hierarchical approach. The partition clustering algorithm is our primary concern in this paper. They partition clustering algorithm divides a data set into a number of clusters, typically by trying to minimize some criterion or error function. The number of clusters is usually predefined, but it can also be part of the error function (Vesanto and Alhoniemi, 2000).

The aim of this paper is to describe how data is partitioned into similar groups or clusters and able to examine the performance mapping process from a high dimensional space to a low dimensional space. Computer simulation of the clustering problem, usefulness of self-organizing maps as well as analyzing the relationship between visual performance and topological preservation was achieved in this study.

This paper is organized as follows: In second part, we presented an overview of self organizing maps, its architecture and its uses in real–life applications. Third part, describes methodology used. This lead us to fourth part, in which computer simulation of clustering problems have been analyzed. Last part concludes the work.

AN OVERVIEW OF SELF-ORGANIZING MAPS

Self-organizing map (SOM) is one of the most popular neural network models operates in an unsupervised learning, which means that no teacher is needed during the learning process. Self-organizing map is used for clustering data without knowing the class memberships of the input data. SOM is able to detect features inherent to the problem and thus has also been called the Self-Organizing Feature Map (SOFM) (Deboeck and Kohonen, 2013).

The basic idea in the formation of self-organizing map include: Competition, Co-operation and synaptic adaptation. The self–organizing maps have the capability to classify feature vectors that consist of numerical attributes. Self-organizing map serve as a neural model of certain cognitive functions that could be able to organized representations of sensory experiences resembled the way the brain maps in the cerebral cortex and other part of the central nervous system do. Self-organizing map acts as data analysis method in various fields of research like in pattern recognition in which real dataset is used.

"Every input data vector shall select the 'model' that matches best with it, and this 'model', called the winner, as well as a subset of 'models' that are its spatial neighbors in the array, shall be modified for better matching" (Kohonen, 2014).

The Self-organizing map is a kind of neural network learning without a supervisor (Kohonen, 1982; Kohonen, 1990; Kohonen et al., 1996; Kohonen, 1997; Kohonen, 1998 and Kohonen, 2013). SOM is widely applied to data clustering problems and speech compression, (Jain et al., 1999) and (Ettauil and Lazaar, 2012), respectively. Some researchers have written on self organizing maps such as (Fritzke, 1994) and (Lagerholm, 2000).

Architecture of a Self-Organizing Map 2.1.

The architecture of a self-organizing map is a feed-forward structure with a single computational layer of neurons arranged in rows and columns. The topological structure property is observed in the brain, but is not found in any other artificial neural network except the self-organizing map (Sivanandam et al., 2006).

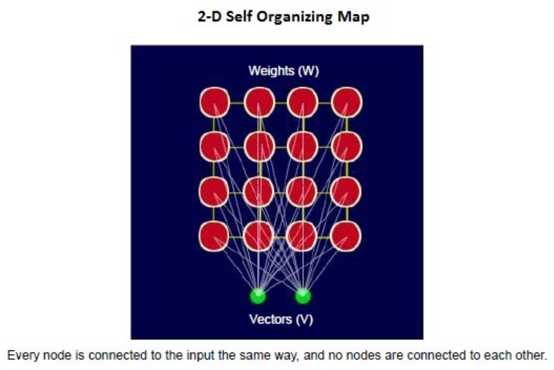
2-D Self Organizing Map
Weights (W)
Vectors (V)
Every node is connected to the input the same way, and no nodes are connected to each other.

Figure 1: Every node is connected to the input the same way and the nodes are connected to each other.

The principal goal of a self-organizing map is to transform an incoming signal pattern of arbitrary dimension into a one, two or more dimensional discrete map, and to perform this transformation adaptively in a topological ordered fashion.

Applications of the Self - Organizing Map 2.2.

According to Lobo (2009), self - organizing map can be used to perform many different tasks. The most common are:

1. Clustering: This is probably the most common application of self-organizing map. For example, given a fixed number n of clusters, the self-organizing map will partition the available data into n different groups. The main advantage of the self-organizing map in this case is that it is less prone to local minima than the traditional k-means clustering algorithm. However, in a self-organizing map, the clusters obtained are topologically ordered, that is, similar clusters are usually grouped together.
2. Ordering of Multi-dimensional data: This type of application makes use of the topological ordering of the self-organizing map to organize a given set of data vectors according to some criteria. In this case, a self-organizing map is also able to create color palettes from pictures.
3. Feature Extraction: As self-organizing map performs a mapping from a high dimensional space to a low dimensional one, it may not's be used for feature extraction. For instance, new features are simply the coordinates of the mapped data point. This is one of the few cases where self-organizing maps with a dimension greater than two are easy to use.
4. Data Interpolation: When using a self-organizing map to interpolate data, the output space of the self-organizing map will have the same dimension as the input space, but since the units are ordered on a regular grid, that grid provides a locally linear interpolator for the data.
5. Control and/or data sensitive processing: A self-organizing map can be used to select, based on available data, the best model, controller, or data processor for a given situation. The main idea behind this type of application is that instead of designing a rather complex controller, multiple simple controllers may be used, each one tuned to a particular type of situation. During the training of the self-organizing map, the input data are partitioned into various regions and each of these is used to train or define the parameters of a different controller.

Other applications include industrial analysis, control and telecommunications, statistical methods such as exploratory data analysis, profiling the behavior of criminals, categorization of galaxies, categorization of real estate and the exploration of full-text databases (i.e. document organization and retrieval and biomedical analysis and applications at large) (Kohonen, 2014).

Advantages and Disadvantages of Self-Organizing Map 2.3.

According to Kohonen, (2014), the major advantages and disadvantages are:

- Data mapping is easily interpreted. The reducing of input data dimensionality and grid clustering makes it easy to analyze similarities in the data. Self-organizing maps factor in all the data in the input to generate these clusters and can be altered such that certain pieces of data have more or less of an effect on where an input is placed.
- Self-organizing maps are fully capable of clustering large, complex data sets. With a small number of optimization techniques, a SOM can be trained in a short period of time
- Self-organizing maps are capable of handling several types of classification problems while providing a useful, interactive, and comprehensible summary of the data. For example, a SOM could be used in place of a Bayesian spam filter.

However, the disadvantages of Self-organizing map are:

- Self-organizing maps require necessary and sufficient data in order to develop meaningful clusters. The weight vectors must be based on data that can successfully group and distinguish inputs. Lack of data or irrelevant data in the weight vectors will add randomness to the groupings. Finding the correct data involves determining which factors are relevant and can be a difficult or even impossible task in several problems. The ability to determine a good dataset is a deciding factor in determining whether to use a SOM or not.
- Self-organizing maps are often difficult to obtain a perfect mapping where groupings are unique within the map. Instead, irregularities in the map often present themselves whereby two similar groupings appear in different areas on the same map. Clusters will often get divided into smaller clusters, generating several areas of similar neurons. This can be prevented by initializing the map well, but that is not an option if the state of the final map is not obvious.
- Self-organizing maps require that nearby data points behave similarly. Parity-like problems such as the XOR gate do not have this property and would not converge to a stable mapping in a SOM.

METHODOLOGY

The method of the squared Euclidean distance between input vector and weight vector has been applied by chosen the unit whose weight vector has the smallest Euclidean distance from the input vector.

Essential Process in the Formation of Self-Organizing Map 3.1.

The formation of self-organization process involves four major components, they are:

1. Initialization: all the connection weights are initialized with small random values

2. Competition: For each input pattern, the neurons compute their respective values of a discriminant function that provides the basis for computation. The particular neuron with the smallest value of the discriminant function is declared the winner. For instance, m – dimensional input:

$$\overline{X} = [x_1\ x_2 - - - x_n]^T \tag{1}$$

$$\mathbf{W}_j = [w_{j1}\ \mathrm{w}_{j2}\ \text{- - -}\ \mathrm{w}_{jm}]^T \tag{2}$$

where m is the number of input neuron and l is the total number of output neurons in the network. Best matching between $\vec{X}$ and $\overrightarrow{W_J}$: J will emerge as the winner, and the winning index is the j; and the corresponding weight will be the winning weight vector. The discriminant function is said to be the squared Euclidean distance between the input vector x and the input vector $\boldsymbol{W}_j$ for each neuron j, defined as:

$$D(j) = \sum (W_{ij} - x_{ij})^2,\ i = 1\ to\ n\ \ \&\ j = 1\ to\ m \tag{3}$$

The continuous input space can be mapped to the discrete output space of neurons by a simple process of competition between neurons.

3. Cooperation: The winning neuron will determine the spatial location of topological neighborhood of excited neurons, thereby providing the basis for cooperation among neighboring neurons. The topological neighborhood for the neurons can be described in terms of the literal distance between neurons i and j on the grid of neurons taking into consideration:

$$T_{j,I(x)} = \exp\exp\left(-\frac{S^2_{j,I(x)}}{2\sigma^2}\right) \tag{4}$$

where *I(x)* is the index of the winning neuron.

This lead to several important properties that it is maximal at the winning neuron, it is symmetrical about that neuron, it decreases monotonically to zero as distance goes to infinity and it is independent of the location of the winning neuron.

4. Spatial Adaptation: The excited neuron decreases their individual values of the discriminant function in relation to the input pattern through suitable adjustment of the associated connection weights, such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced. For topographic neighborhood, not only the winning neuron gets its weight updated, but the weights of its neighbors will be updated as well. The appropriate weight update equation is defined as:

$$W_{j,i} = \propto(t).T_{j,I(x)}(t).(x_i - W_{j,i}) \tag{5}$$

where a time *t* of epoch dependent learning rate, $\propto$

$$\propto(t) = \propto_0 \exp\exp\left(\frac{-t}{\eta_\alpha}\right) \tag{6}$$

and the updates are applied to all the training pattern *x* over many epochs.

Description of the Training Algorithm 3.2.

The self-organizing training algorithm used can be described as follows:

1. In our experiment, we set $\alpha = 0.5$ due to iteration process.
2. Two input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ are chosen randomly
3. Select the output network layer topology
4. Initialize the network weights using random values *w1* and *w2*
5. While stopping condition is false, do steps 6 to 12
6. For each input vector **x**, do steps 7 to 9
7. For each *j*, compute squared Euclidean distance as presented in equation (3)
8. Find index *J*, when *D(j)* is minimum in equation (3)
9. For all units *J*, with the specified neighborhood of *J*
10. For all *I*, update the weights

$$W_{ij(new)} = W_{ij(old)} + \propto \left[x_i - W_{ij(old)} \right] \tag{7}$$

11. Update the learning rate
12. Reduce the radius of topological neighborhood at specified times
13. Test the stopping condition

It is important that the initial weight values are different from each other and much smaller than input. If the weights are initialized with values similar to input data, then the network has the tendency to choose specific winners, avoiding its self organization.

EXPERIMENT

In order to assess the advantages of our work, we apply the proposed algorithm. A MATLAB program for Self – Organizing Map has been developed to cluster the input vectors. The result computes and displayed the weights vectors as well as the learning rate after number of epoch reached.

Experiment 5.1. Two input vectors within a square of side 1.0 centered at the origin was chosen. The initial weights are chosen between -1 to 1 and the number of cluster units was specified as 50. Assuming the initial learning rate is 0.5 and supposed to be reduced over 100 epochs. In the training process, the winning unit and its neighbor unit on either side are allowed to learn. Write MATLAB program for Self-Organizing Map to:

(i) Compute and display the final weight vectors (ii) Determine the learning rate after number of epoch reached (iii) Cluster the input vectors to plots figures showing: (a) Training patterns at the initial stage (b) Initial weights (SOM at epoch = 0) (c) Final weights (SOM at final epoch)

Table 1: The initial and final weight matrices as obtained in *(Experiment 5.1)*

Initial weight Matrix at Epoch = 0		Final Weight Matrix at Epoch = 100	
w1	w2	w1	w2
-0.3580	0.0693	0.2262	0.4444
0.2881	0.0755	0.1618	0.4605
0.0576	-0.3335	0.0993	0.4507
0.2060	-0.0928	0.0688	0.4020
-0.0314	0.7976	0.0672	0.3618
-0.1787	0.0404	-0.0502	0.2983
-0.1469	-0.6038	-0.0571	0.2070
-0.7432	-0.4550	-0.0861	0.1389
-0.8739	0.3061	0.1043	-0.0534
-0.2048	-0.0203	-0.0701	-0.1684
0.0462	-0.6109	0.0115	-0.1609
0.5080	-0.0534	0.1791	-0.1042
0.1882	0.1208	0.2353	-0.1074
0.7997	-0.5016	0.2962	-0.1194
0.6273	0.0744	0.3434	-0.0999
-0.0454	-0.0940	0.3676	-0.0036
-0.7628	-0.4423	0.2747	0.0687
0.6029	0.4690	0.1772	0.0913
-0.1453	0.0104	0.1442	0.0945
-0.0598	0.5206	0.0420	0.0817
-0.7218	-0.3845	-0.1145	0.0552
-0.3031	-0.5125	-0.1658	0.0020
-0.9008	-0.8736	-0.2311	-0.0609
-0.1267	0.1620	-0.2590	-0.0884
-0.1561	-0.6446	-0.3320	-0.0824
-0.6173	0.1568	-0.4055	-0.1079
0.4790	-0.8551	-0.3885	-0.1570
0.0519	0.0122	-0.3757	-0.2359
-0.0196	0.2004	-0.3605	-0.2768
-0.1440	0.1098	-0.3746	-0.3982
0.2158	0.0304	-0.3786	-0.4372
-0.5810	-0.0514	-0.2638	-0.4257
0.1600	-0.3722	0.0055	-0.3737
0.2170	-0.4144	0.0349	-0.3638
0.1941	-0.0805	0.2036	-0.3352
0.2295	0.2459	0.2177	-0.3269
-0.1285	0.5814	0.0125	-0.0847
0.5347	-0.2473	-0.2159	0.1837
-0.1441	-0.0605	-0.2331	0.2408
-0.4661	-0.1863	-0.3237	0.3801
-0.0795	-0.2138	-0.3772	0.3992
-0.7226	-0.4245	-0.4153	0.3085
0.1052	0.7161	-0.3619	0.2885

-0.6588	0.0806	-0.3143	0.2830
0.3498	0.3479	-0.1065	0.3030
0.5727	0.0577	0.1790	0.2796
-0.1877	-0.0011	0.2749	0.2495
0.4100	-0.1982	0.3885	0.2083
-0.2218	0.5612	0.4339	0.2414
0.0760	0.1107	0.3897	0.3184

Table 2: The number of epoch, elapsed time and learning rate each after an interval of 10 iterations.

No. of Epoch	Elapsed Time (seconds)	Learning Rate, α
10	0.268259	0.4510
20	0.531526	0.4020
30	0.749062	0.3530
40	1.158876	0.3040
50	1.329415	0.2550
60	1.632931	0.2060
70	1.835182	0.1570
80	2.080429	0.1080
90	2.368240	0.0590
100	2.476164	0.0100

The results of Table (1) and Table (2) may differ while simulating the same type of problem; this is because in this problem we used random matrices as our input vectors. The elapsed time and learning rate, which is a decreasing function of time, should be put into consideration, including the initialization network weights to be within a random number between -1 to 1.

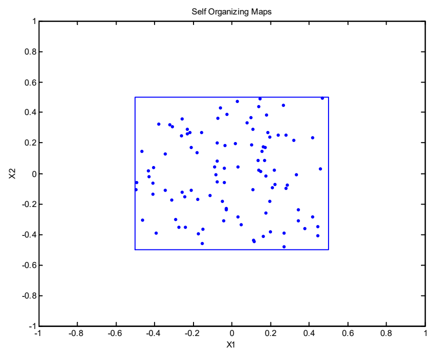

Figure 2: Net input vectors at initial stage

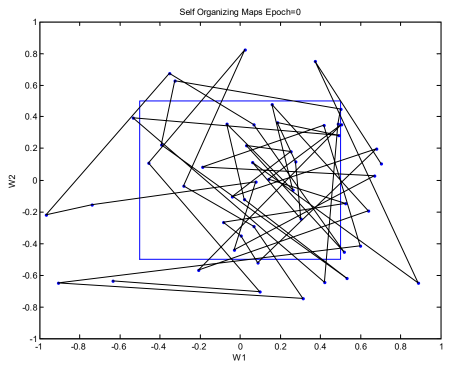

Figure 3: Clustering of input vectors at zero epoch

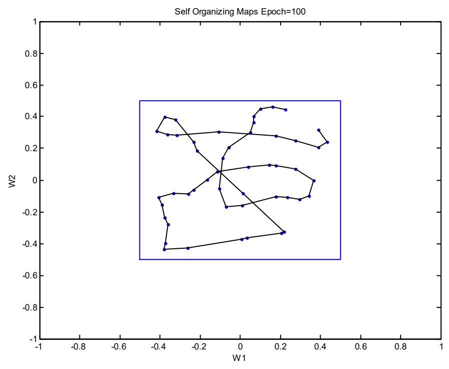

Figure 4: Clustering of input vectors at 100 epochs

CONCLUSION

In this paper, we have described the importance of grouping data into a number of clusters by designing a MATLAB program that clusters the input vectors. The program computes and displays the initial and final weight matrix, as shown in Table 1. Similar results are shown in Table 2 after the final epoch (100 epochs), the elapsed time and learning rate were obtained as 2.476164 and 0.0100, respectively. The performances of the proposed method based on squared Euclidean distance for clustering purposes were shown in Figure 2 to Figure 4. This experiment proves that SOM is applicable in so many areas and is capable of clustering large and complex data sets, informing automobile production, reducing production cost, text clustering, reducing of input data dimensionality e.t.c.

ACKNOWLEDGMENT

This research is partly financed by FRGS grant (203/ PMATHS/6711368) from the Ministry of Higher Education, Malaysia.

REFERENCES

Deboeck, G. and Kohonen, T. (Eds.). (2013). *Visual explorations in finance: with self-organizing maps*. Springer Science & Business Media.

Ettaouil, M. and Lazaar, M. (2012). Improved Self-Organizing Maps and Speech Compression. *International Journal of Computer Science Issues (IJCSI)*, **9(2)**: 197-205.

Fritzke, B. (1994). Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural networks*, **7(9)**: 1441-1460.

Jain, A. K., Murty, M. N. and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, **31(3)**: 264-323.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, **43(1)**: 59-69.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, **78(9)**: 1464-1480.

Kohonen, T. (1997). Exploration of very large databases by self-organizing maps. In *Proceedings of the IEEE International Conference on Neural Networks*, **1**: 1-6.

Kohonen, T. and Somervuo, P. (1998). Self-organizing maps of symbol strings. *Neurocomputing*, **21(1)**: 19-30.

Kohonen, T. (2013). Essentials of the self-organizing map. *Neural Networks*, **37**: 52-65.

Kohonen, T. (2014). MATLAB Implementations and Applications of the Self-Organizing Map. *Unigrafia Oy, Helsinki*.

Kohonen, T., Oja, E., Simula, O., Visa, A. and Kangas, J. (1996). Engineering applications of the self-organizing map. *Proceedings of the IEEE*, **84(10)**: 1358-1384.

Lagerholm, M., Peterson, C., Braccini, G., Edenbrandt, L. and Sornmo, L. (2000). Clustering ECG complexes using Hermite functions and self-organizing maps. *IEEE Transactions on Biomedical Engineering*, **47(7)**: 838-848.

Lobo, V. J. (2009). Application of self-organizing maps to the maritime environment. *In information Fusion and Geographic Information Systems*, 19-36. Springer Berlin Heidelberg.

Pang, K. (2003). Self Organizing Maps. *J Neural Networks*. Retrieved from: https://www.cs.hmc.edu/~kpang/nn/som.html, May 5, 2016.

Sivanandam, S. N., & Deepa, S. N. (2006). *Introduction to neural networks using Matlab 6.0*. Tata McGraw-Hill Education.

Vesanto, J. and Alhoniemi, E. (2000). Clustering of the self-organizing map. *IEEE transactions on neural networks*, **11(3)**: 586-600.

Zherebtsov, A. A. and Kuperin, Y. A. (2003). Application of self-organizing maps for clustering DJIA and NASDAQ100 portfolios. *arXiv preprint cond-mat/0305330*.