Self-Organizing map (SOM)

Amir.M Mousavi.H

Department of Computer Engineering
Shahid Rajee University

Tehran, Iran

AmirMahmood.Mousavi@yahoo.com

Abstract **- Self Organizing maps (SOM) is a special class of artificial neural networks used extensively as a clustering and visualization tool in exploratory data analysis. It is inspired by sensory activation patterns of the cerebral by the inputs from the eye retina. This paper presents a general introduction and discussion of how a Self-Organizing maps (SOM) works. then we show the implementation result of a simple Self Organizing maps (SOM) on Artificial data and then analyze the network's behavior in presence different situations and compare our codes to ready functions in term of performance.**

Keywords— Self Organizing maps, Neural Network, Clustering, SOM, Classification, Artificial Data

I. INTRODUCTION

Since the dawn of the data era, more and more efficient data analysis technologies have been researched, proposed and applied at a very fast pace, especially tools for statistical analysis for high-dimensional data (data with multiple features). A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction. Self-organizing maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space. Self-Organizing Maps (SOMs) proposed by [1] are considered effective tools for the visualization of high-dimensional data [1]. The SOM algorithm is used to compress the information to produce a similarity graph while preserving the topologic relationship of the input data space. The convergence of the SOM has been previously discussed and guaranteed [1].

The basic SOM model construction algorithm can be interpreted as follows:
1) Create and initialize a matrix (weight vector) randomly to hold the neurons. If the matrix can be initialized with order and roughly compiles with the input density function, the map will converge quickly [1];

2) Read the input data space. For each observation (instance), use the optimum fit approach, which is based on the Euclidean distance :

$$c = \underset{i}{arg\min} \,||\, x - m_i \,||$$

to find the neuron which best matches this observation. Let x denote the training vector from the observation and mi denote a single neuron in the matrix. Update that neuron to resemble that observation using the following equation:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)]$$

$mi(t)$: the weight vector before the neuron is updated.
$mi(t+1)$: the weight vector after the neuron is updated.
$x(t)$: the training vector from the observation.
$hci(t)$: the neighborhood function (a smoothing kernel defined over the lattice points), defined though the following equation:

$$h_{ci}(t) = \begin{cases} \alpha(t), & i \in N_c \\ 0, & i \notin N_c \end{cases}$$

Nc: the neighborhood set, which decreases with time.
$\alpha(t)$: the learning-rate factor which can be linear, exponential or inversely proportional. It is a monotonically decreasing function of time (t).
3) Update the immediate neighborhood of that neuron accordingly (Figure 1).

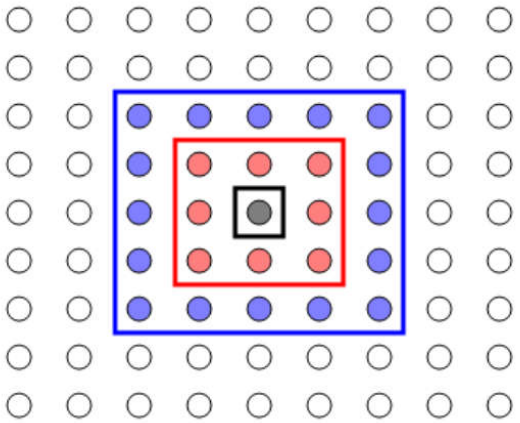

Figure.1. Neighborhoods (Nc) for a rectangular matrix of cluster units: Nc = 0 in black brackets, Nc = 1 in red, and Nc = 2 in blue.

As proposed by Cheng [2], after running this algorithm with a sufficient number of iterations, the map will ultimately converge. However, it is difficult for users to determine how many iterations are sufficient. Another practice measure is evaluating the map's quality, which can help users determine the optimal number of iterations.

II. METHODS & MATERIALS

In order to implement self-organizing map (SOM), we used python version 3.7.0.
On the other hand to train the network, we generated artificial data, a random 2 dimensional array coming from a mixture of guassian distribution (for the first pick with mean = [2, -3] cov = [[1.2, 0], [0, 1.2]] and and for second pick with mean = [2,2] , cov = [[1, 0], [0, 1.5]] and for third pick with mean = [-3,3] , cov = [[0.5, 0], [0, 0.5]]). For each pick we generated 150 ,100, 50 samples, respectively. In figures.1- are shown:

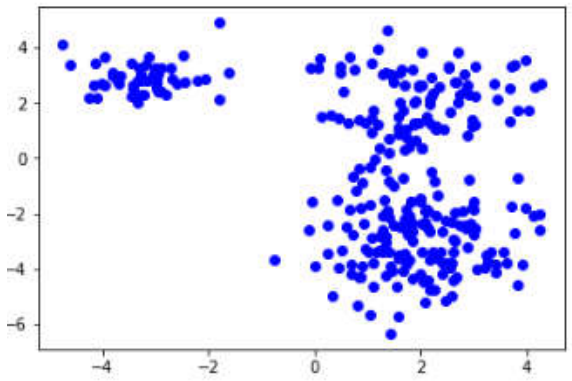

Figure.1 data distribution

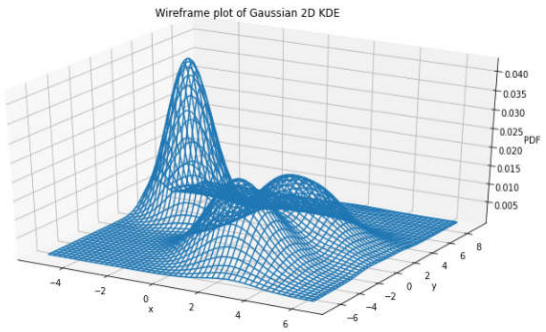

Figure.2 data distribution in 3D

For the hyperparameters we set:

- neighborhood = Gaussian function
- sigma = 50 with reduction every step
- epoch numbers: 50
- default map size = 10*10

Our network includes 2 input neurons, 2 ,3, 10 neurons in output layer. The network's topology is shown in figure.3.

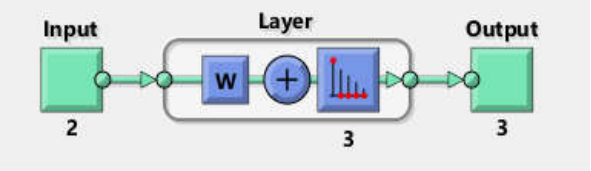

Figure.3. The network's topology for 3 output neurons

III. EXPERIMENTAL RESULTS

our results have shown that a self-organizing map (SOM) can cluster an unknown data distribution with a high accuracy and make it visualize
.we started to test the model with 2,3,10 neurons, the results are shown in figures.4-6:

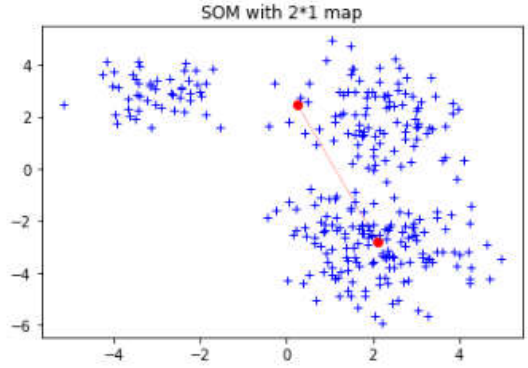

Figure.4.SOM with output neurons = 2

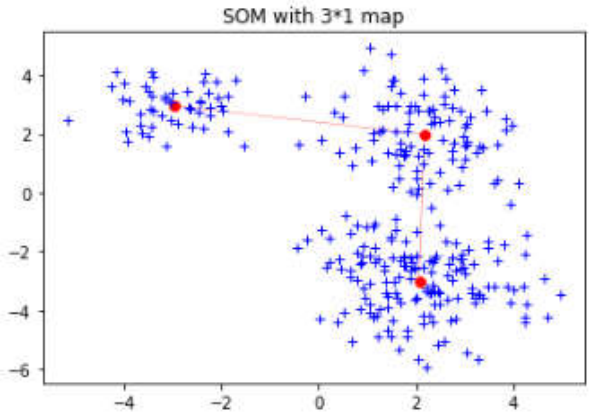

Figure.5.SOM with output neurons = 3

For SOM with output neurons = 10, we have 2 option:

1. creating a map with size: 5 * 2

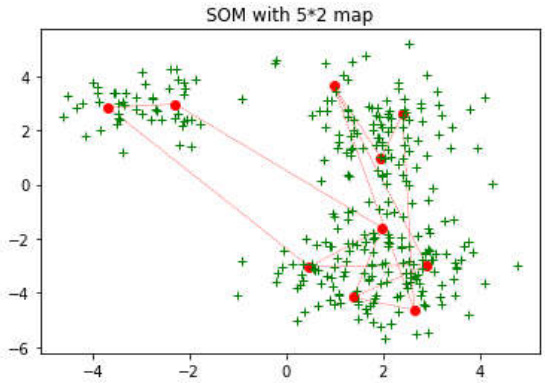

Figure.5.SOM with output neurons = 10 & map size = 5*2

2. creating a map with size: 10 * 1

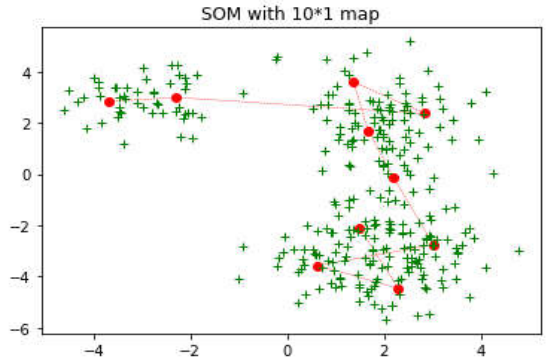

Figure.6.SOM with output neurons = 10 & map size = 10*1

As we can observe, it indicates that SOM depend on initial position of neurons in the beginning but after an appropriate number of epochs it will change to the optimal points, so epoch numbers is an essential parameter for SOM.

For further experiment we increased neurons of our map to test other sizes (4*4, 7*7, 10*10 map sizes which is equal to 16, 49, 100 neurons, respectively). (figure.7)

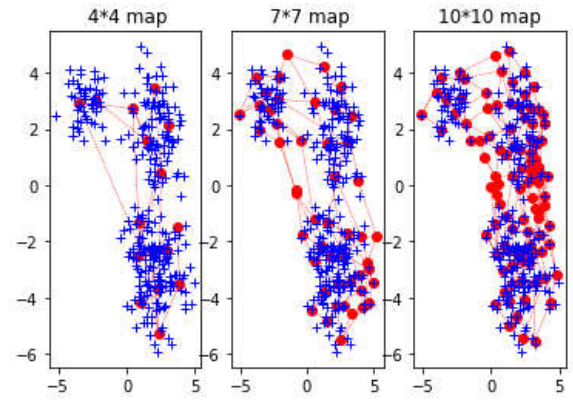

Figure.7. SOM with 4*4, 7*7, 10*10 map sizes which is equal to 16, 49, 100 neurons, respectively

As it can be conculded if we increase number of neurons we will have more clusters, in the ares with high density. If number of neurons was equal to nember of inputs, final location of centers exactly would be the same location of inputs.

For further experiment we compare our code in python with MATLAB's library for SOM. (figures.8)

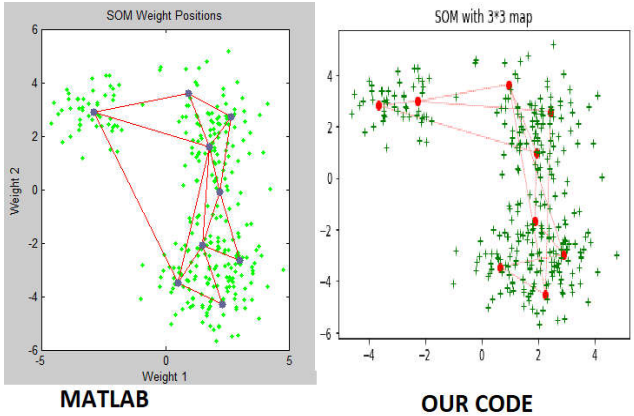

Figure.8. our model compared to MATLAB's library

As it can be seen, it seems that our code is almost the same as MATLAB's library, MATLAB is a little better because it put centers in the places with more density, overally the performance of both models are similar.

After running main experiments, we started to investigate more and add some supplementary data. All of them are in 3*3 map size.

First, we focused on map shape before and after fitting SOM for data.

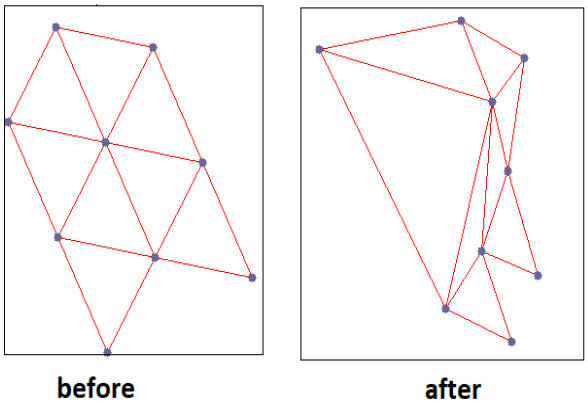

Figure.9. on map shape before and after fitting SOM for data

Then we calculated each cluster contains how much of data :

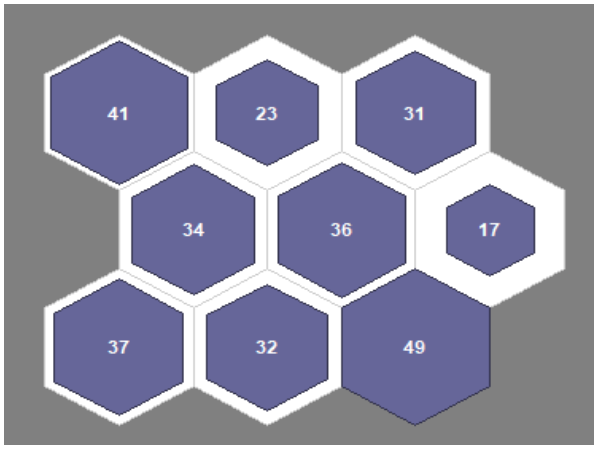

Figure.10. number of data in each cluster

As it's shown, in every cluster, the number of data is trying to be almost equal to each other.
Then we came for to see how is the inputs weights

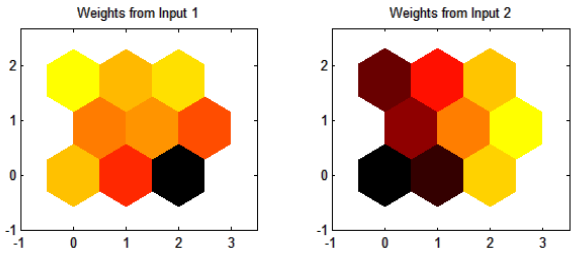

Figure.10. hit-map of inputs weights

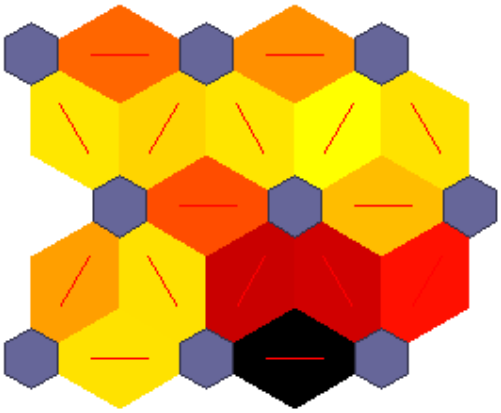

Figure.11. hit-map of SOM neighbor weights distance

As we expected SOM put centers in data with high density, as shown in figure.11 darker colors indicates low distances which is make sense if we attend to data distribution.

IV. CONCLUSION

In the discipline of machine learning, clustering is a quickly moving field. Its growth has been fueled by technological advances in digital imaging, computer processors and mass storage devices. In this paper an attempt is made to a famous neural network which is called Self-Organizing maps (SOM). Here we tested how SOM works and how the number of neurons will effect on performance. We observe that as much as our neurons amount grows, we will have more clusters in areas with high density data. Also it was tested initial location of neurons in map does not matter for SOM but we must be careful about appropriate number of epochs in order to reach the optimal points, because appropriate number of epochs make the centers create the best clusters based on competitive learning and similarity to input data and the neighborhood (explained algorithm). Then we compared our code with MATLAB's library, it seemed our model with a little difference performing worth in clustering and MATLAB's library chooses more appropriate centers.

REFERENCES

[1] T. Kohonen, Self-organizing maps, Third edition.. ed. Berlin ; New York, Berlin ; New York : Springer, 2001.

[2] R Core Team, R: A Language and Environment for Statistical Computing. Vienna, Austria, R Foundation for Statistical Computing, 2017.

[3] Guido van Rossum, Python Programming Language, 2014.

[4] G.T. Breard, "Evaluating self-organizing map quality measures as convergence criteria," Evaluating SOM quality measuresThesis (M.S.) University of Rhode Island,2017 2017.

[5] B.H. Ott, "A convergence criterion for self-organizingmaps,"DigitalCommons@URI 2012.

[6] V. Moosavi, S. Packmann and I. Vallés, "A python library for self organizing map(SOM),", 1/17/ 2018.