Impact Factor (SJIF): 4.542

International Journal of Advance Research in Engineering, Science & Technology

e-ISSN: 2393-9877, p-ISSN: 2394-2444

Volume 4, Issue 3, March-2017

Implementation of Self Organizing Map using MATLAB

Satyajit S. Uparkar[1], Pravin Y. Karmore[2], Apurva A. Barmase[3], Trupti A. Ganorkar[4]

[1]*Department of Computer Applications, RCOEM, Nagpur*
[2]*Department of Computer Applications, RCOEM, Nagpur*
[3]*TCCS, Nagpur*
[4]*Ramanora Global Pvt Ltd. , Pune*

Abstract ***—The latest versions of MATLAB are embedded with analytical tools to handle the Big Data issues. Self Organization Map (SOM) is a vector quantification method which places the prototypes vectors on a regular low dimensional grid in an ordered fashion. This makes SOM a powerful visualization tool for understanding a big data set very easily. To apply this technique the given data set is cleaned first for removing the outliers. A network is created to understand the topological relationship within the training set. This paper focuses on procedure, analysis and interpretation of a sample dataset using MATLAB tool.***

Keywords- MATLAB, SOM, training set, topological relationship

I. INTRODUCTION

The concept of data analysis is far from revolutionary. For centuries individuals have attempted to understand and uncover useful patterns in data with the expectation that these discoveries might enhance tasks such as decision making, prediction and modeling. This has spawned a host of data mining activities. A more recent aspect of data analysis that has emerged is that of analysis and understanding of image data. One of the methods used in analyzing image content is that of clustering. Additionally, the SOM perform sun supervised training and is therefore well suited to unlabeled data. Moreover, its training phase proceed sin such a way that it preserves the topology of the training data. In this thesis the Self Organizing Map training algorithm is reproduced and a variety of experiments done to observe its effects on different types of data, including randomly generated data values from a number of different distributions, 1-dimensional images, high contrast data patches, face data, and images of natural outdoor Scenes. A large part of this experimentation is based on using the Self-Organizing Map network configured in a variety of topological models. Among these models are: the rectangle, cylinder, torus, Mobius strip and Klein bottle.

II. OVERVIEW

The Self Organizing (SOM) or Kohonen Map as mentioned previously is an artificial neural network which provides the major capabilities of dimensionality reduction where it can function as a non-linear, generalized form of principle component analysis and reduce a high dimensional input training space of a much lower dimensional representation – as little as one or two dimensions. It also provides the functionality of feature clustering where similar features found in the input data will tend to be grouped in clusters in the map. The SOM also allows for comprehensive visualization of the lower dimensional representation of training data, where clusters can be clearly identified. The SOM is able to carry out unsupervised training, where labels and classifications are not required for the training input. This is a desirable property particularly because unlabeled or sparsely labeled data is more readily available especially in image datasets where an image can belong to multiple classes. Thus the SOM is able to uncover naturally clusters. Data compression and competitive learning are performed by the SOM via a process known as vector quantization. This is a technique used in statistical pattern recognition, where classes are described by a small number of code book vectors. Modeling of probability density functions enable clusters of input vectors to be formed in order to compress data without loss of important information.

III. NETWORK ARCHITECTURE

The architecture of the Self Organizing Map is as shown in Figure 1 given below. The network is made up of 2 layers: the input layer (training data) and the output layer, represented as a 2-dimensional grid (map/cortex) of nodes. Each node in the output layer is fully connected to each node in the input layer; however there is no intra-layer connections between the nodes of the output layer associated with each node of map is an x-y position in the map and a weight vector which is equal in size to an observation of the training input. The weight vector of a node defines the input pattern to which it is associated. The neighborhood of each node is defined as all nodes within a radius R = 0, 1, 2, as shown in the diagram. Thus a neighborhood of R =1would contain eight adjacent nodes.

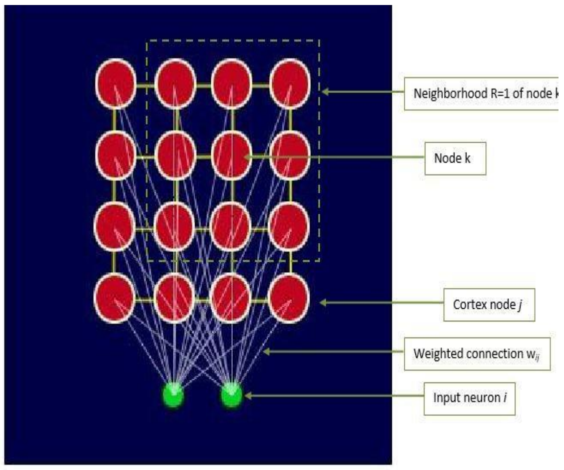

Figure 1: SOM Network Architecture

IV. TRAINING ALORITHM

In training, weight vectors of the SOM are randomly initialized with small weights. Each data instance of the training input is presented to the network and the distance between that instance and each nodes weight vector is calculated typically via the Euclidean distance metric.

$$d = \sum_{i=1}^{n} (v_i - w_i)^2 \qquad \text{…(1)}$$

where v represents the input data and w, the weight vector. The node having the closest weight vector to the input (minimum) is chosen as the winner or Best Matching Unit (BMU) of the map. Weights of all nodes within a defined neighborhood of the best matching unit are then updated. Weights are updated according to the following equation:

$$w(t+1)=w(t) +\Theta(v, t)\alpha(t)(v(t)-w(t)) \qquad \text{….(2)}$$

where w is the weight vector of the BMU, v represents the input vector, t is the iteration number, α is a learning rate and ϴ signifies the influence of distance from the winner (this value decreases with an increase in distance). The entire procedure following the initialization of weights is carried out over a number of iterations, with the radius of the neighborhood of the winner decreasing each time until R=0, where the neighborhood consists of only the winner. The decrease in the neighborhood is calculated via exponential decay as follows: R(t)=R exp (-t/λ) where R=0 is the initial radius, typically the radius of the map (half the greater dimension of the grid), and λ is a decay constant. After training the SOM should give visualization where similar data are clustered within close proximity, and having smooth transitions or overlaps where clusters change. This is demonstrated in the example below showing eight colors being mapped from their three dimensional components: red, green, blue to the two dimensional SOM space.

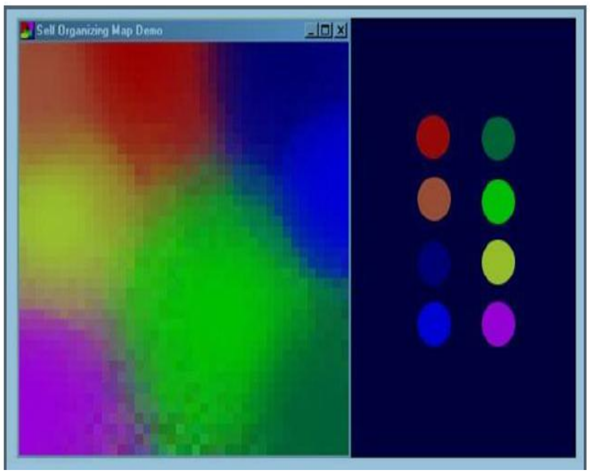

Figure 2: SOM trained to cluster colors

V. MAKING THE SOM USING MATLAB

Now we are ready to move to the initialization and computing of the SOM in MATLAB, which begins with the declaration of the parameters, and continues with the execution of the functions somlininit and sombatchtrain:

```
 for i=1:12
mi=min(x(:,i));
ma=max(x(:,i));
x(:,i)=(x(:,i)-mi)/(ma-mi);
end
plot(x(1,:),x(2,:),'g')
holdon
gridon
dimensions =[1010];
 coverSteps=100;
initNeighbor=4;
topologyFcn='hextop';
distanceFcn='linkdist';
net2=selforgmap(dimensions, coverSteps, initNeighbor, topologyFcn, distanceFcn); [net2,Y]=train(net2,x);
plotsompos(net2,x);
gridon;
plotsomnd(net2)
plotsomhits(net2,x)
```

The SOM has now been computed and expressed as the struct sm. Next we show how it can be shown explicitly.

5.1 Plotting of SOM Array and its Labeling:

The plotting of the SOM may take place in many different ways. In this example we first define a blank hexagonal SOM graphic of a correct lattice size, the cells of which we have to label by the due symbols of the elements. The blank hexagonal network is drawn by the commands using MATLAB. net2=selforgmap(dimensions, coverSteps, initNeighbor, topology Fcn, distanceFcn) [net2,Y]=train(net2,x)

```
 plot sompos(net2,x)
grid on
```

The symbols of the elements are then written into the due hexagons. They are defined in the following way. We first define two strings labels1 and labels2, which define the two letters that define the elements:

```
labels1 = 'ASAICCACPMNPPFZSB';
labels2 = 'lbgrdouubgidtenni';
```

The first letter of the element numbered by the parameter u, u=1...17, is the u: the element in the string labels1; the second letter is the u: the element in the string labels2, respectively.

For instance,

```
’Al’ = [labels1(1) labels2(1)], and
’Bi’ = [labels1(17) labels2(17)].
```

Here first we have to calibrate the SOM nodes using the original input data X, for which we have to find the winner nodes. These are found by the piece of script.

```
M = sm.codebook;
 norms2 =sum(M.*M,2);
for u=1:17
X1 =X(u,:)’;
Y = norms2-2*M*X1;
[C,c]=min(Y);
dimensions =[100];
coverSteps =100;
initNeighbor=10;
topologyFcn ='gridtop';
distanceFcn ='linkdist';
net1=selforgmap(dimensions, coverSteps, initNeighbor, topologyFcn, distanceFcn); [net1,Y]=train(net1,x);
plotsompos(net1,x);
grid on
```

The output can be seen in the following Figure 3.

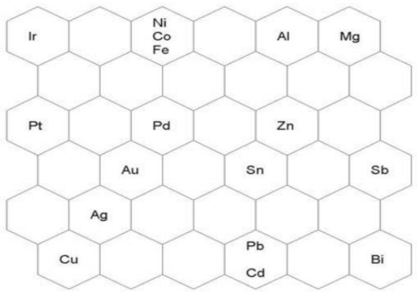

Figure 3: Plotting of SOM Hits

The locations on the SOM display, into which the symbols of the element shave to be written, are defined by the horizontal rows ch and vertical columns cv of the SOM array. However, in the text command used for labeling the cells, ch takes the role of the x coordinate and cv the role of the y coordinate, respectively. These coordinates are resolved as ch =mod(c-1,6) +1;

The locations on the SOM display, into which the symbols of the element shave to be written, are defined by the horizontal rows ch and vertical columns cv of the SOM array. However, in the text command used for labeling the cells, ch takes the role of the x coordinate and cv the role of the y coordinate, respectively. These coordinates are resolved as ch =mod(c-1,6) +1; cv=floor((c-1)/6)+1;

Now we can write the symbols automatically into their correct places on to the SOM, defined by the coordinates ch and cv, by the following script. Because the even and odd rows of the hexagonal SOM are mutually displaced in the horizontal direction, we have to use the shift1parameter for the horizontal shift to position the text correctly. However, since there were a few collisions of different labels in the same cells, we have to use the shift2 parameter in these locations to position the colliding symbols correctly also in the vertical positions. We may need further are the commands to print and store the figure.

5.2 Interpretation of Data Set:

A surprising result in this example is that we can find a tight ferromagnetic cluster of Ni, Co and Fe at the top, although we did not consider the magnetic susceptibility or any other magnetic properties of the metals. This is obviously due to some strong correlation between the physical properties of the ferromagnetic metals.

VI. CONCLUSION

To conclude, we have ascertained that the Kohonen Self Organizing Map network is one in which adequate and efficient data clustering can be performed. Not only has this, butthe self- organizing map also possessed the property of topological preservation of training data. Through a number of experiments we have shown the ability of the map to make natural associations particularly between regions of images and general image content. We have also explored how we may determine goodness of a SOM with measures of its resolution and its continuity. We have further seen that the map may be configured in a number of different models, including those attempted in these experiments : the rectangle, cylinder, torus, Mobius strip ,and Klein bottle. We may deduce that with connectivity being introduced into the map these models are able to enhance the overall SOM performance.

REFERENCES

[1] L.Fausett,“Fundamentals of Neural Networks: Architectures, Algorithms and Applications”, Prentice-Hall,Englewood Cliffs, NJ, 1994

[2] P. Jeyanthi and V. Jawahar Senthil Kumar, “Image Classification by K-Means”, in Advances in Computational Sciences and Technology, ISSN0973=6107,Vol.2,No. 1,2010, pp. 1-8.

[3] F. Jiang, H. Berry, M. Schoenauer, “The Impact of Network Topology on Self Organizing Maps”, in L.Xu, E.D. Goodman, G.Chen,D. Whitley, and Y. Ding, editors, GEC Summit, pages 247–254. ACM, 2009

[4] J. Kangasand T. Kohonen, “Development and Applications of the Self-Organizing Map and Related Algorithms”, in Mathematics and Computers in Simulation,Vol.41,July1996,pp. 3-12.

[5] Alahakoon, D. ; Halgamuge, S. ; Srinivasan, B., "Dynamic self-organizing maps with controlled growth for knowledge discovery", in Neural Networks, IEEE Transactions on (Volume:11 , Issue: 3), May 2000, pp 601-604.

[6] Kohonen, T. "The self-organizing map", Proceedings of the IEEE (Volume:78 , Issue: 9) , Sept. 1990, pp. 1464 - 1480 .

[7] Kangas, J.A. ; Kohonen, T.K. ; Laaksonen, J.T., "Variants of self-organizing maps", Neural Networks, IEEE Transactions on (Volume:1 , Issue: 1), Mar. 1990, pp. 93-99.

[8] Kohonen, T. , Oja, E. ,Simula, O. ,Visa, A., "Engineering applications of the self-organizing map", Proceedings of the IEEE (Volume:84 , Issue: 10), Oct 1996, pp. 1358 – 1384.