International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 7, Issue 3 (May 2013), PP. 44-55

A Comparison of Sofm With K-Mean Clustering And Ann Pattern Reorganization:A Review

[1,]Karishmatyagi,Viet Dadri [2,] Gauravsachan ,Viet Dadri

Abstract:- This paper mainly focus on how the input vectors organize themselves into different patterns, that is, self-organizing maps, based on various algorithms like incremental batch algorithm, k-mean clustering methods and linear vector quantization when applied in SOFM algorithm itself. It also throws light on how SOFM algorithm works and defines various topologies. In this paper we have tried to show how the map looks like after the number of iterations. Moreover we have compared the three known strategies of self-organizing feature map and discussed the advantage of one technique over other.

Keywords:- feature maps, linear vector quantization, neural network, clusters, classification, neurons, competitive layer, neuron-neighbours, network design

1. INTRODUCTION

The self-organizing map methods are used for classifying the unknown samples into some categories based on their features without having the prior knowledge about the input data.It provides us with a resultant topological network that resembles the input onto the plane exactly as it resembles in high dimensional space ,that is, the points or neurons that are near to each other in space are also near each other onto the plane in topological design. The SOM can thus serve as a cluster analysing tool for very large datasets. These feature maps can also understand and classify the inputs that they have never met before.

1.1 Multiple Layers of Neurons

An artificial neural network consists of several layers. The first layer is an input layer, the last layer is known as output layer and all the intermediate or middle layers are referred as hidden layer. Each of these layers has a weight vectorW, a biasor threshold valueb, and an output matrix or vector a. To show difference between the weight, threshold andoutput vectors in each layer, the number of thelayer is added as a superscript to every variable respectively.

The network shown below has *R*1 number of inputs with *S*1 neurons in the first layer, *S*2 neurons in the second layer and so on.Each layer hasdifferent numbers of neurons. A constant value of 1 is given to each neuron as a threshold value. The central layers are all hidden layers and the output of one first hidden layer becomes the input for next hidden layer. Thus layer
2 can be analysed as a one-layer network with *S*1 inputs, *S*2 neurons, and an *S*2 × *S*1 weight matrix
W2.Second layer has input a1 but output is as a2.Since we know all the inputs and outputs of this layer, it alone can be treated as single-layer network. Multiple-layer networks are very strong and powerful. If we have a network of two layers where the first layer is sigmoid and

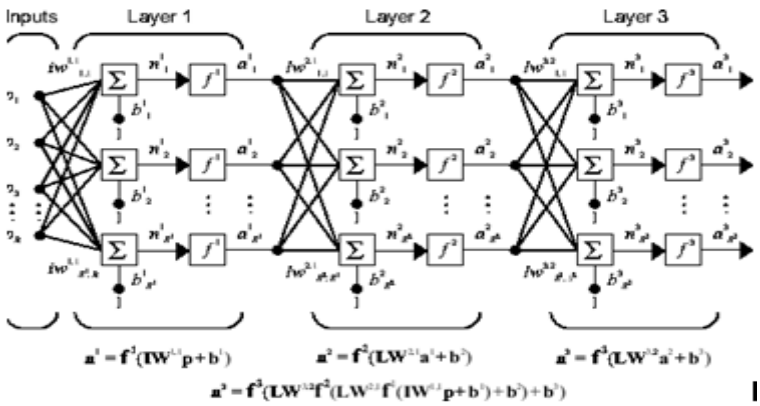

Fig1: Multiple layer neural network

the second layer is linear, then with the help of multilayer networks and backpropogation algorithms ,this can be combination of any input that can be transformed into any function although with a finite number of discontinuities arbitrarily.

1.2 Self-organizing in neural networks

Self-organizing neural networks are designed to detect regularities and correlationsin the input they receive and adapt their future responses to that input accordingly. The strategy followed by self- organizing maps to recognize the clusters of similar type of inputvectors is that the way neurons physically near each other in the neuronlayer respond to similar input vectors. Self-organizing maps do not havetarget vectors. They simply divide the input vectors into clustersof similar vectors. There is no desired output for these types of networks.

1.3Important Self-Organizing functions

We can create competitive layers and self- organizing maps with competlayer and selforgmap, respectively in Matlab.

1.4 Architecture of competitive layer(a base for SOM layer)

The n dist box takes the input vector p and weight vectorIW1,1, and calculates the negative of the distances between them and produces the result as a vector having S1 elements.

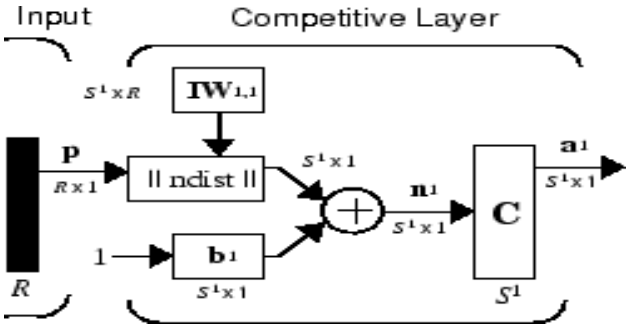

Fig2: Competitive layer

The net input n1 is calculated by adding the threshold value to the calculated S1 value. If the value of p is equals to the neuron's weight vector andall biases are zero, then a neuron will have maximum net input 0.The competitive transfer C gives the output as 0 for all the neurons accept for the neurons that have maximum positive value, that is n1.For this the neuron has output as 1 which is declared as winner.If all the threshold values are 0 ,but the weight vector is not equal to the input vector p then the neuron whose weight vector and the input vector has the least negative distance are declared as winner with output 1.

2. SELF-ORGANIZING FEATURE MAPS

Self-organizing feature maps (SOFM) classify the input vectors on the basis of their grouping in the input space and their neighbouring sections. Thus, a self- organizing maps consists of both the information ,that is, about the distribution of input vectors and their topology.The neurons in the layer of an SOFM are arranged according to the following topological functions defined below in detail as gridtop, hextop and randtop that organize the neurons in a grid, hexagonal, or random pattern. Distances between neurons are calculated from their positions with a distancefunction. Self-organizing feature map network uses the same procedure as of competitive layer to find the winning neuron i*but together with updating the winning neuron .It also updates the neighbourhood Ni* (d) of the winning neuron using the Kohonenrule as: $iw(q) = iw(q-1) + \alpha(p(q) - iw(q-1))$ Here the neighbourhood Ni* (d) contains the indices for all of the neurons thatlie within a radius d of the winning neuron i*. $Ni(d) = \{j, dij \leq d\}$ So,the weights of the neuron that is winner and its nearby neighbours move towards the p when p is presented.

2.1 Incremental Batch algorithm

In this algorithm, before making any updating ,the whole data set is presented to the network. The algorithm then determines a winning neuron for each input vector. Then Each weight vector moves to the average position of all of the input vectors for which it is a winner, or for which it is in the neighbourhood of a winner. In the figure on right hand side, thefirst diagram shows a two-dimensional neighbourhood of radius d = 1 around neuron 13. The second diagram shows a neighbourhood of radius d = 2.

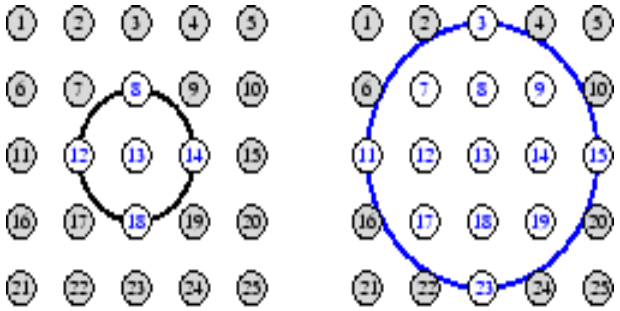

Fig3: Neurons with their respective radius and neighbourhood

These neighbourhoods are:N13 (1) = {8,12, 13, 14, and 18} andN13 (2) = {3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, and 23}. The neurons in an SOFM can be arranged into one, two or more dimensionalpatterns.The performance of the network is not sensitive to the exact shape of the neighbourhoods.

2.1.1 Topologies used in feature map

In Matlab ,the following functions are used to specify the locations of neurons in the network:

1. **gridtop**

pos = gridtop(2,3)

pos =

0 1 0 1 0 1

0 0 1 1 2 2

It gives u an array of neurons of 2 X 3 size.Here neuron 1 has the position (0,0), neuron 2 has the position (1,0), andneuron

3 has the position (0,1), etc. Let us take an example:

If we write the following code on matlab window.It will create a gridtop topology of 8X10 size of neurons. pos = gridtop(8,10); plotsom(pos)

The graph will look as:

plotsom(pos) that gives the following graph.

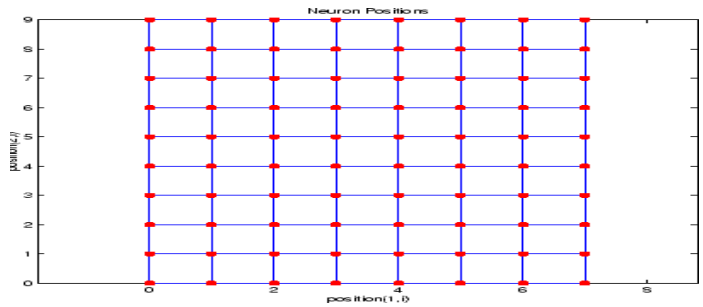

Fig 4: Agridtop topology of 8 X10 size

2. HEXTOP

Similarly a graph of 8X10 set of neurons in a hextop topology is created withthe following code:

pos = hextop(8,10);

plotsom(pos)

that gives the following graph.

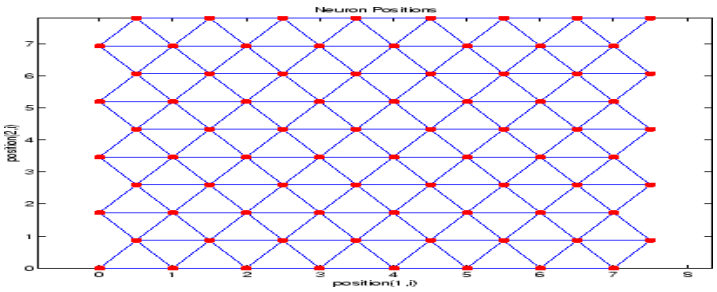

Fig 5: A hextop topology of 8 X10 size

3. **RANDTOP**

This function creates neurons in an
N-dimensional random pattern. The following code generates a random pattern of neurons.
pos = randtop(2,3)
pos =
00.7620 0.62601.4218 0.0663 0.7862
0.0925 00.49840.6007 1.1222 1.4228

The 8X10 set of neurons in a randtop topology has the following code:
pos = randtop(8,10);

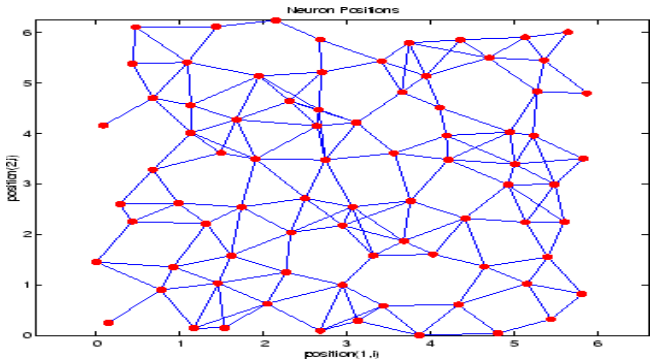

Fig 6: A randtop topology of 8 X10 size

2.1.2 Architecture

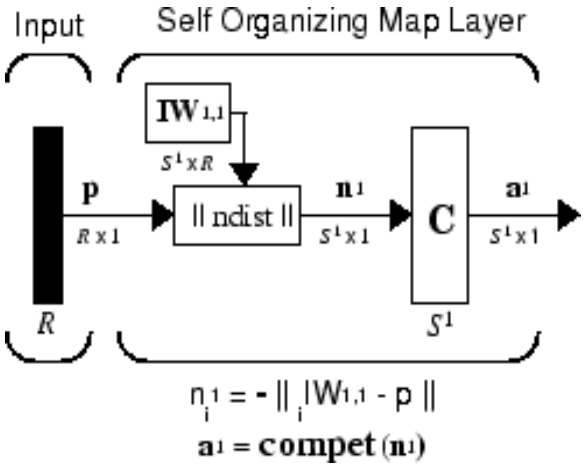

Fig 7: The architecture for SOFM

This architecture works on the basic principle of competitive network, except the difference that no threshold value is used in calculation here.The competitive transfer function produces 1 for output element a1i corresponding to i*, the winning neuron. All other output elements in a1 are 0.Neurons close to the winning neuronare updated along with the winning neuron. We can choose anytopology and distance expressions of

neurons. 2.1.3 Creating a Self-Organizing Map Neural Network The feature maps arecan be created with the function selforgmap in Matlab window. Thisfunction defines variables used in two phases of learning: • Ordering-phase learning rate • Ordering-phase steps • Tuning-phase learning rate • Tuning-phase neighbourhood distance Training: In this phase, the network identifies the winning neuron for all inputs. The weight vectors then move towards the location of inputs for which they are winner or closest to the winner. The distance or size of the neighbourhood is changed or updated during trainingthrough two phases. Ordering Phase:The neighbourhood distancestarts at a given initial distance, and decreases to the tuning neighbourhood distance (1.0).As we know that distance among the neighbouring neurons decreases during this phase so the neurons rearrange themselves in the input space in the same way as are arranged physically in the network. Tuning Phase: This phase lasts for the rest of training or adaption. The neighbourhood sizehas decreased below 1 so only the winning neuron learns for each sample. One-Dimensional Self-Organizing Map Let us take an example of 1D map. If there are 100 two-element unit input vectors spread evenly between 0° and90°. angles = 0:0.5*pi/99:0.5*pi; so the data will be plotted with the following code : P = [sin(angles); cos(angles)];

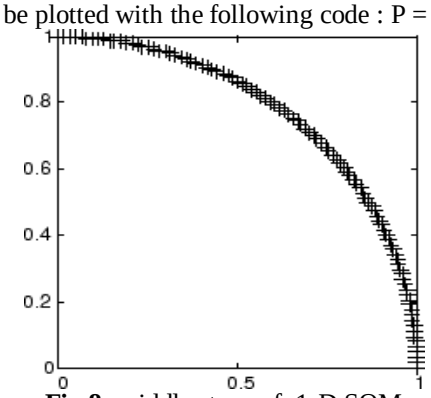

Fig 8: middle stage of 1-D SOM

As shown below, initially all neurons are at the centre of the figure.

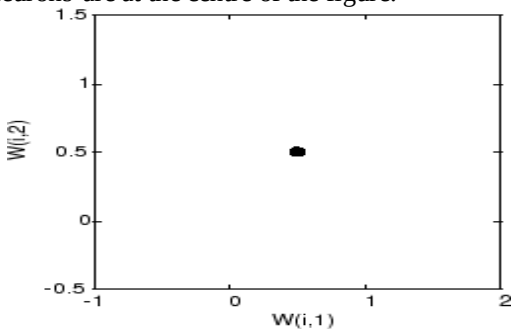

Fig 9: starting stage of 1-D SOM

Initially all the weight vectors are concentrated at the centre of the inputvector space.During the training phase all the weight vectors simultaneously move towards the input vectors.They also become ordered as the neighbourhood size decreases. Finally thelayer adjusts its weights in such a way that each neuron responds effectively to a regionof the input space occupied by input vectors. The placement of neighbouring neuron weight vectors also represents the network design of the input vectors.

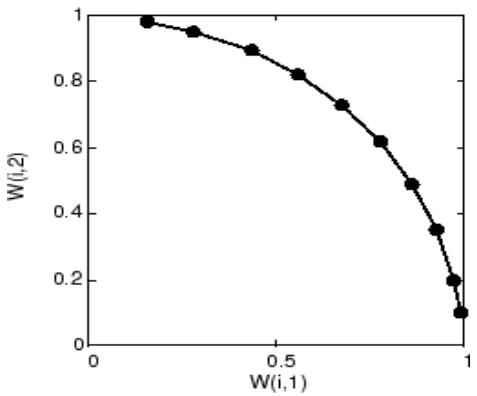

Fig10: last stage of 1-D SOM

After 120 cycles, the self-organizing feature map starts to rearrange even further more. After 500 rounds or cycles the map shows how the neurons are more cleanly n evenly distributed in input space. Two-Dimensional Self-Organizing Map First of all, some random input data is created with the following code: P = rands(2,1000);

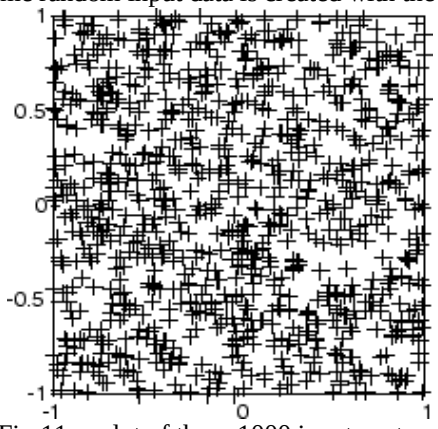

Fig 11: a plot of these 1000 input vectors.

A 5-by-6 two-dimensional map of 30 neurons is used to classify these inputvectors. After 120 cycles the SOFM look like as

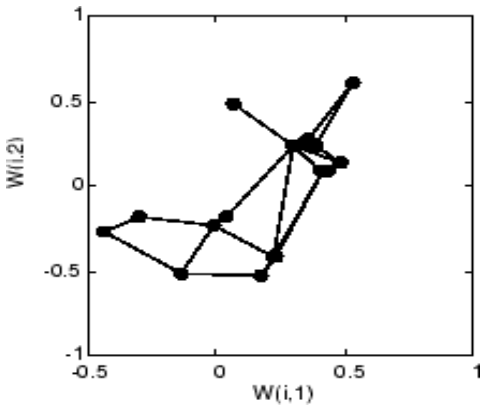

Fig12: the map after 120 cycles

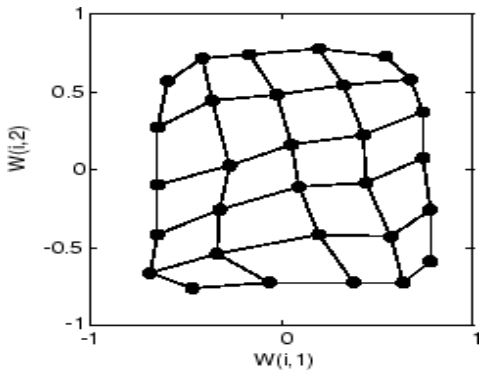

Fig13:a plot after 5000 cycles

So after having near about 5000 cycles, the map shows further smooth distribution of neurons in input space. So we can say a two-dimensional feature map finds the topology of itsinputs' space even much better. Although we know that these feature maps doesnot take very long time to arrange themselves so that neighbouring neurons can also classify themselves, but the map itself takes long time to arrange according to the distribution of input neurons vector.

2.1.4 Training with the Batch Algorithm

The batch training algorithm is much faster than the simple incremental algorithm, and it is the default algorithm for SOFM training. The following code creates 6X6
2-D feature map of 36 neurons:

x = simplecluster_dataset net = selforgmap([6 6]); net = train(net,x);

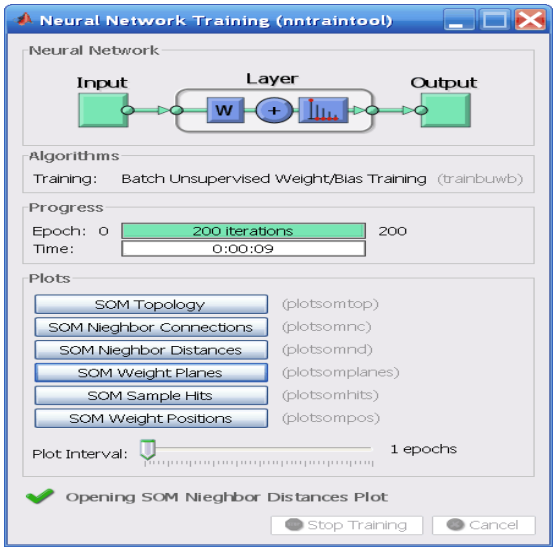

Fig15: After 200 iterations of the batch algorithm, the map is well distributed through the input space.

If we click the SOM sample hit option in the command window, we will get the following window shown below as output, which shows the data points of each neuron and also shows how these data points are distributed in input space.

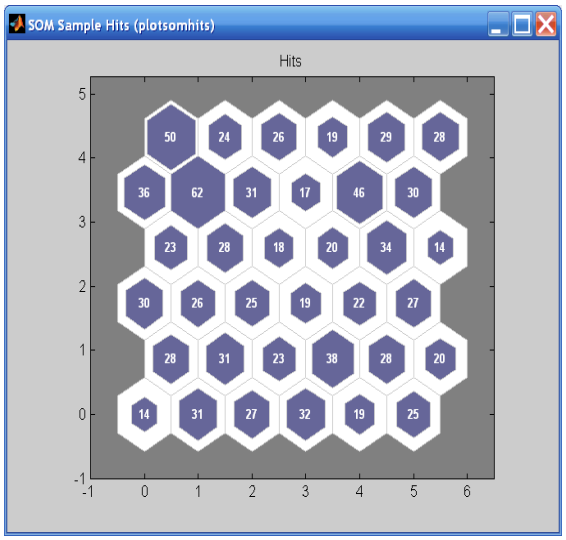

Fig14: Batch training algorithm command window

The above figure shows the various options that can be performed using batch training algorithm which is used to create more efficient feature maps. By clicking the various options different operations can be performed.

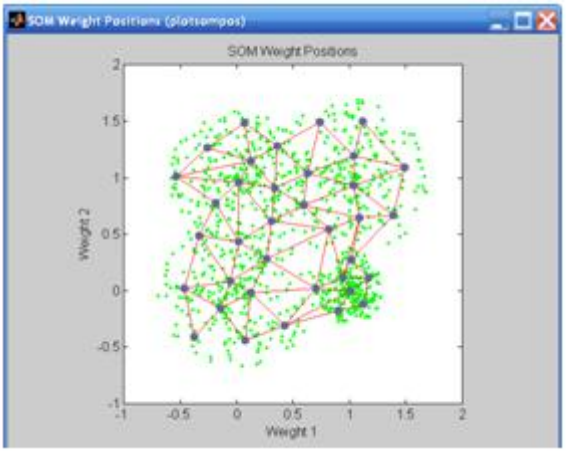

Fig 16: data points with their respective neurons.

In a batch training method the process updates the map at every iteration with the help of the following equation shown below. Some advantages of batch method over the sequential version are: a) it produces a map faster and b) it does not need a learning rate to converge.

$$\boldsymbol{m}_i(t+1) = \frac{\sum_{j=1}^{m} h_{ij}(t)\boldsymbol{S}_j(t)}{\sum_{j=1}^{m} n_{V_j} h_{ij}(t)}$$

This algorithm runs in the following 4 steps:

1. Retrieval of historical knowledge from a map
2. Starting of the new map.
3. Searching the winning neurons.
4. Updating of weight vectors

All the above steps are performed for each input vector of neurons.

An incremental method to form maps employing the batch training under non- stationary environment does not need the historical data chunks to form an updated map. Instead, it only needs some knowledge from the historical maps and the new data in order to be performed. An experiment using a data set representing different data distribution has been set in earlier research which shows the topology of the map during training. The map is able to not forget the topology previously learned from the past data chunks. Therefore, a batch training proposal for non-stationary environments might be considered.

2.2 K-means algorithm for generating feature maps

It belongs to the category of partitioning algorithms. It minimizesthe reconstruction error by assigning a nearest neighbour in that compresses the data vectors onto a smaller set of reference vectors. In this algorithm, the total number of clusters is defined priorly. The new clusters are made by splitting on the variable with the largest distance. Here the data set is splitted into the selected number of clusters by maximizing between relative to within - cluster variation. This algorithm is an iterative procedure that assigns the different data vectors to the predefined number of clusters that do not overlap each other. In a first trial k-means was used with the specification to produce two clusters. The data set is splitted into two subsets as:

This figure shows the resulting subsets of clusters.

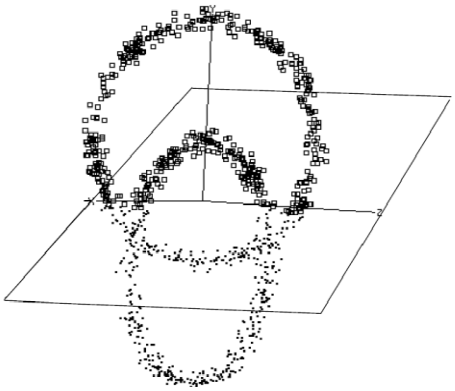

Fig 17: k-means parameterized for two clusters

k-means splits the dataset in the same way as a hyperplane cutting through the rings would do. It seems that k-means is unable to take the nonlinear structure of the data set into account. Since the number of clusters is a parameter to the k-means algorithm, it is a common practice for exploratory data analysis to run k-means with several different numbers of clusters. K-means with four pre specified clusters produces the result given in the below figure :

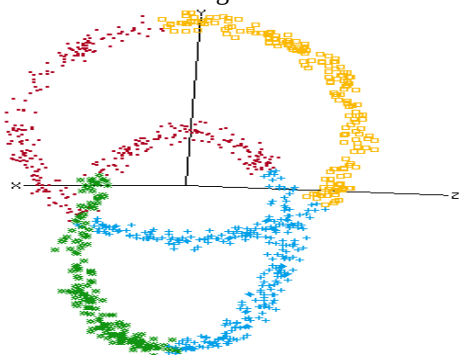

Fig 18: k-means parameterized to 4 clusters

While two of the resulting clusters contain only data of one ring, the other two, marked with a dot receptively a cross in

figure 18, contain a mixture of the two rings. In summary it can be said, the k- means algorithm is unable to classify this dataset correctly.

2.3Artificial Neural-Network Pattern Recognition Technique for creating feature maps

In ANN architecture the classification procedure is implemented in three phases. All the phases are described in detail below

Learning phase1-Self-organizing feature map (SOFM): This phase provides the "approximate" quantization of the input space by adapting the weight vectors of the neurons in the feature map.
A problem with SOFM is that if the weights are initialized at small random values it gives different results at different runs which is not desired when we are trying to optimize the performance of the algorithm even more. Let us take the case of physician. He wants to review the classification result of neuromuscular disorder of the patient caught in the form of EMG signals by motor unit action potentials. For this case, the motor unit is to be identified first that has EMG signals and then it has to be classified using self- organizing map with linear vector quantization technique.
In order to avoid the problem in this particular case as an example, the weights of the output nodes should not initialized at small random values but at 0.0001 using vector quantization method. The stepsof algorithm are shown in the following figure below:

Step 1: Initialize weights at 0.0001.

Step 2: Calculate distances between the input vector x_i and weight vectors for each output node k.

$$d_k = \sum_{i=1}^{N}(x_i - w_{ik})^2 \quad \text{Where } k = 1, 2, \text{----}8 \text{ and } N=120 \qquad (4)$$

The output node with minimum distance is the winner.

Step 3: Adapt the weights. The weights for each output node k and for each i are adapted with

$$w_{ik}(t+1) = w_{ik}(t) + h_k\,(x_i - w_{ik}(t)). \qquad (5)$$

Where h_k= learning rate and it is a Gaussian function.

It can be given as:

$$h_k = g\exp((-(k-k_w)^2 t/2)\sqrt{(t_{kw})} \qquad (6)$$

Where,

Value of g can be $0< g \leq 1$,

k_w is the winner node,

t is the number of iterations,

t_{kw} is the number of times the specific node is selected as the winner.

For the initialization $g = 1$ and for the first winner $t_{kw} = 1$ and $k = k_w$, then $h_k = 1$.

If calculated $h_k < 0.005$, then the weights of the specific node are not adapted, since the change in the weights vector will be minimum. This is implemented in order to save computation time.

Step 4: Go to Step 2 and repeat for all segmented inputs. After all inputs are presented to the network, the first adaptation of the weights vector is completed and the system proceeds to the second learning phase.

Fig 19: algorithm steps for SOFM Learning phase 2-Learning vector quantization (LVQ):

It adapt the weights vectors slightly to optimize the classification performance. For this the vector quantization demands the knowledge of correctly classified inputs. The adaptation carried out during the first learning phase should be correct and the segmented inputs should be correctly classified. Updatingthe weight and winning neuron selection is done same as in phase 1. The implementation steps are given in the following figure below:

Step 1: Use the values of the weight vectors as obtained from learning phase 1.

Step 2: Present input and calculate distances d_k between the input vector x_i and weight vectors w_{ik} for each output node as in equation (4). The output node with the minimum distance d_{k1} is the first winner $k1$ and the output node with the minimum distance d_{k2} is the second winner $k2$.

Step 3: Adapt weights. The weights for the first winner output node $k1$ is adapted with

$$w_{ik1}(t+1) = w_{ik1}(t) + h_{k1}\,(x_i - w_{ik1}(t))$$

And for the second winner $k2$ with the weight

$$w_{ik2}(t+1) = w_{ik2}(t) - 0.1(d_{k1}/d_{k2})h_{k1}\,(x_i - w_{ik1}(t))$$

The learning rate h_{k1} is initialized to 0.2 and decreases linearly with the number of times t_{kw1} the specific node $k1$ is selected as the first winner

$$h_{k1} = 0.2 - 0.01t_{k1w}$$

Here

w_{ik1} is weight vector with the correct label (first winner).

w_{ik2} is the weight vector with the incorrect label (second winner).

The factor d_{k1}/d_{k2} is used to control the adaptation of the second winner.

Step 4: Go to Step 2 and repeat for all segmented inputs.

Fig 20: Implementation steps for SOFM applied with LVQ for the particular case.

correctly with different input patterns. But the neural networktechnique for developing feature map supported by linear vector quantization method is fast and efficient mechanism, although we have tried to explain with a particular case but it is so..The use of theLVQ algorithm with the SOFM algorithm optimizes the classification boundaries.It makes the algorithm fast and suitable for real-time applications.

Phase 3: Classification:

This is the last phase where all the input vectors are defined to one of the output nodes. The use of the LVQ algorithm with the SOFM algorithm optimizes the classification boundaries through slight adaptation of the weights vectors in ANN pattern reorganization method.

III.CONCLUSION:

Through this paper we have tried to compare the three mainself- organizing feature mapsdeveloping techniques.We have studied the batch training method in detail together with the topological functions itsupportsand shown how the vectors organize themselves into different patterns at each iteration. The k-mean algorithm we studied also develops the feature maps but it is not as efficient as incremental batch algorithm as it is unable to classify the dataset

REFERENCES:

[1]. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 –8958, Volume-1.

[2]. Ieee transactions on neural networks, vol. 11, no. 3, may 2000

[3]. <html><meta http-equiv="content- type"

[4]. content="text/html;charset=utf-8">

[5]. <CITE>genome.tugraz.at/MedicalI nformatics2/<B>SOM</B>.pdf</CI TE>

[6]. Tamkang Journal of Science and Engineering, Vol. 5, No. 1, pp. 35-48 (2002)

[7]. [Char92] Charalambous, C.,"Conjugate gradient algorithm forefficienttraining of artificial neural networks," IEEE Proceedings, Vol. 139, No. 3,1992, pp. 301–310.

[8]. [Koho97] Kohonen, T., Self- Organizing Maps, Second Edition, Berlin:Springer- Verlag, 1997.

[9]. [Lipp87] Lippman, R.P., "An Introduction to computing with

[10]. neural nets,"IEEE ASSP Magazine, 1987, pp. 4–22.
[11]. Neural network toolbox using Matlab
[12]. Internet explorer- google search.