International Journal of Computer Applications Technology and Research
Volume 5– Issue 4, 202 - 209, 2016, ISSN:- 2319–8656

Usage of Self-Organizing Map for Clustering Vertices

Rashmi Lad
MIT, Arts Commerce and Science College, Alandi (D), Pune-412106, M. S., India

P S Metkewar
Symbiosis Institute of Computer,Studies and Research (SICSR) Pune-411016, M. S., India

R.S. Walse
College of Dairy technology, Pusad, Nagpur, M.S., India

Abstract – Usage of Self-Organizing Map (SOM) for clustering vertices of any given graph. Simultaneously its input is observed and worked in terms of weight matrix, learning rate and final resultant matrix, which helps to form a cluster. The purpose of this paper is to introduce a procedure of SOM for clustering and observed impact corresponding to varied weight class for simple graph or vector matrix using Euclidean distance. A simple vector matrix problem is solved by using 2, 3 & 4 weight class matrix. By adopting a different weight matrix class with same vector matrix has presented a clustering and visualization.

Keywords – self-organizing map, topology, visualization, clustering, Euclidean distance.

INTRODUCTION

Self-Organizing Map (SOM) was developed by Professor Kohonen's. It is also called as Kohonen's SOM (Self Organizing map).SOM works on unsupervised learning which means training without any guidance or teacher. SOM learns on its own from beginning till the end and it is unsupervised competitive learning. The self-organizing Map is a special type of neural network that accepts N-dimensional input vector and maps them, in which neurons are organized in a hexagonal or rectangular grid and find the feature space with its neighboring neuron.

The main objective of this learning algorithm is that the network forms the feature map which takes input data and maps them into 1 or 2-dimensional feature space. The main feature of SOM is that it contains only two layers, the input layer, and an output layer. There is no hidden layer in SOM so it is different from other learning algorithms like feedforward back prorogation learning algorithm.

LITERATURE REVIEW

Authors [1] have observed that "On the use of Self-Organizing Map for clustering and visualization" of the number of output units. In this paper, SOMs can be used for clustering and visualization separately or simultaneously. There are various types of application used to compare SOM with other statistical approaches.

Authors [2] have focused "Clustering of the Self-Organizing Map" with different approaches. Authors used hierarchical agglomerative clustering and partitive clustering using K-means. By using SOM, they produced the prototypes and then performed direct clustering of the data and to reduce the computation time.

Authors [3] have observed that "Clustering Application of SOM neural network in clustering" is an unsupervised neural network for two-dimensional maps. It finds the similar data that will map to nearby locations. In this paper, authors introduce an experiment to analyze the SOM in clustering.

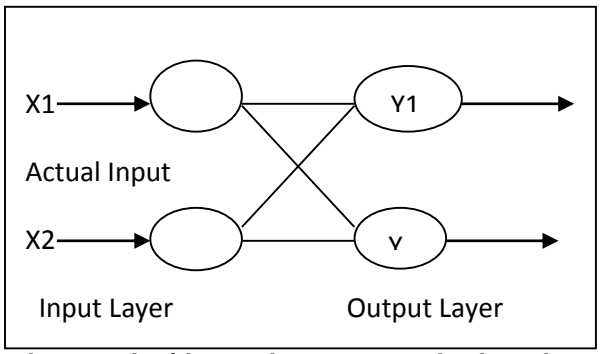
Authors [4] have emphasized that "The use of Three-dimensional Self-Organizing Maps for Visualizing Clusters in Geo-referenced Data" that maps 3D SOM data for visualizing clusters in geo-referenced data. This paper provides a comparison of a 2D or 3D SOM for a problem and increases the clustering quality of 3D SOMs.

In this paper Section 1 gives a review of the problem definition. Section 1.1 presents the architecture of self-organizing map. Section 1.2 describes the methodology of self-organizing map Section 1.3 shows Pseudo code section 1.4 describe how to train SOM and Section 1.5 shows research methodology. Section 2 it describes the computational analysis of self-organizing map. Section 2.1 describes the computational result analysis of self-organizing. Section 2.1.1 gives the result based on 2 weight class matrix. Section 2.1.2 describes the result based on 3 weight class matrix. Section 2.1.3 describes the result based on 4 weight class matrix. Section 3 – Describe the result analysis that is available after the large calculation.

1. PROBLEM DEFINITION

The Self-Organizing Map (SOM) is an artificial neural network that is very effective for clustering via visualization. It is very difficult to visualize SOM for the vector data. In this paper, we show that the vertices of SOM can be used successfully for visualizing clusters using Euclidean distance method of the neural network. We try to find the small distance between nearest cluster. In this paper, we use the vector matrix and 2, 3 and 4 weight matrix classes to find the winning neuron or the resultant output layer. Based on this we can find nearest cluster group and observed the changes.

1.1 Architecture of SOM



Each input node of the input layer is associated with weight w_{ij} that is adjusted during training. The SOM maintains

topological relationships between inputs in such a way that the neighboring inputs in the input layer are associated with neighboring neurons.

1.2. Pseudo Code

^ Declare n, m and Set its value by 5 and 1

^ Set learning rate by default 0.5

^ Declare input vector matrix

^ Declare weight class matrix

^ Repeat the procedure until m is less than or equal to n
Repeat the procedure for 1 to 5 for n
Check when m equal to 1
Store first input matrix
Check when m equal to 2
Store second input matrix
Check when m equal to 3
Store third input matrix
Check when m equal to 4
Store fourth input matrix
Otherwise
Store fifth input matrix
Stop
^ Repeat the procedure for 1 to 5

Sum ← Sum + square root (weight matrix value – input matrix value)

Sum = Square root of(sum);

^ Find min sum

^ Update that weight matrix (new) = weight matrix (old) + learning rate *(input matrix – weight matrix)

^ learning rate ← learning rate -0.1

^ Stop

1.3. Training SOM

There are two types of operation in self-organizing map

1. **Training Phase**

In the Training phase, the output node is found with the help of Euclidean distance between the input vector and the weight class connecting to that input and finds the minimum between them. This node is called the winner node and weight class. Now the weight of the neighboring output node will be updated so that the new weight is closer to the current input vector. This procedure is repeated for all input vectors and weight till they become constant. After one iteration or epoch of input vector, the learning rate gets changed and is multiplied by 0.5 at every epoch. In this way after applying the input vector, only the winner unit is determined.

This function is selected for the size of weight change in the distance of the neuron. This distance is calculated with the topology defined on the output layer of the network.

2. **Clustering Phase**
After training the SOM should give visualization where similar data are clustered within close proximity, and having smooth transitions or overlaps where clusters change.

1.4. Research Methodology

Exploratory research is one type of research method which is based on the theoretical idea. Researcher gets an idea from currently available theory and tries to elaborate or understand more about that topic. Sometimes it is the initial groundwork for this type of research. Exploratory research used in two ways; either a new topic or a new idea. A new topic is finding from the currently existing theory. New idea can come to understand exiting theory and set new perception according to that.

2. Computational Analysis of SOM

The Experiment is derived by using following directed graph

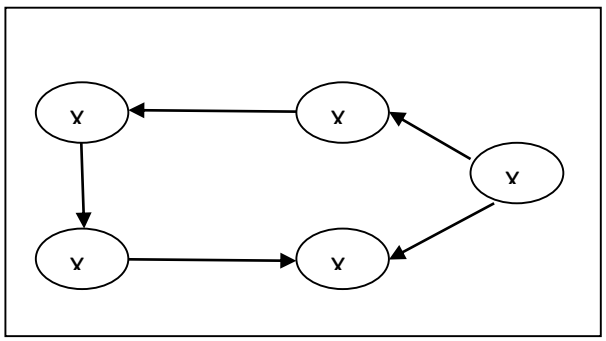


Figure 1: Directed graph

There is 5 node or input in the directed graph. They are connected with each other. The representations of Adjacency matrix (5 X 5) are as follows.

	X1	X2	X3	X4	X5
X1	0	1	0	0	0
X2	0	0	0	0	1
X3	0	0	0	1	1
X4	1	0	0	0	0
X5	0	0	0	0	0

Weight matrix for input is 4X5 which means there are 4 classes for 5 input nodes. The weight initializes between 0 to 1 only.

To evaluate the performance of the proposed algorithm, we tested 5 input vector and 3 different classes. The SOM algorithm was tested 5 input vector and 3 different classes. The SOM algorithm was executed using DOTNET code. The source and input parameters for these problems are shown.

Initial learning rate $\eta = 0.5$

After initialize learning rate and weight matrix calculate Euclidean distance with the following formula.

$$D(j) = \text{sqrt}\left(\sum_{i=1}^{n} (X_i - w_{ij})^2\right)$$

Where n is no of the input node. To find the distance between input vector subtract the input vector with weight matrix value and find the square root of that. After this calculates the summation of all input values for D1.value of j varies according to the number of vectors of weight matrix.

In this way repeat the same procedure for all the classes of weight matrix. Then compare all distances and find the minimum distance between them. That minimum distance is the Best Matching Unit (BMU) or winning neuron.

The distance of that weight matrix class is BMU or winning neuron. Now update the weight matrix on the winning cluster with the following formula.

$$W_{ij}(new) = W_{ij}(old) + \eta * [x_i - w_{ij}(old)]$$

Then get the new or updated weight matrix. Follow the same procedure for all input vectors. After repeating the same procedure for all input vectors, **one epoch** is over. The learning rate will change after one epoch. Learning rate will decrease with the following formula.

$$\eta = \eta * 0.5$$

The default learning rate is 0.5.Now repeat the same for another epoch till updated weight matrix does not get a similar result.

2.1. Computational Result Analysis of SOM

A summary of all SOM parameters used for solving the problems is given in the table.

2.1.1 Input vector 5X5 and weight class 2

Class means a set or category of things having some property or attribute in common and differentiated from others by kind, type, or quality.

Weight Matrix for class 2

Wij =

	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}
w1j	.2	.4	.6	.8	1
w2j	.9	.7	.5	.3	.1
	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}

Epoch 1- when η=0.5

In this example first take input vector v1 & weight of w1j (where j = 1 to 5) and find the distance1. Perform similar procedure for same input vector v1 & another weight class w2j (where j =1 to 5) and find distance 2.Now check minimum between Dist1 and Dist2. When we get minimum distance then update weight matrix as per the winning neuron. If dist1 is minimum then w1j will change otherwise w2j class. Now the similar procedure is followed for the entire input vector v2, v3, v4 and v5 and the updated weight matrix is found.

Table 1: Result of weight matrix class 2 with learning rate 0.5

	Input vector	Winning Neuron	Dist 1	Dist 2	Mini mum	Updated weight	
						Δw1	Δw2
Iteration 1	v1 =[0,1,0,0,0]	B	2.4	1.25	1.25	No change	[.45,.85,.25,.15,.05]
Iteration 2	v2 =[0,0,0,0,1]	A	1.2	1.91	1.2	[.1,.2,.3,.4,1]	No change
Iteration 3	v3=[0,0,0,1,1]	A	0.5	2.61	0.5	[.05,.1,.15,.7,1]	No change
Iteration 4	v4=[1,0,0,0,0]	B	2.42	1.11	1.11	No change	[.725,.425,.125,.075,.025]
Iteration 5	v5=[0,0,0,0,0]	B	1.52	0.72	0.72	No change	[0.36,.213,.062,.038,.013]

When one epoch is over so decrease the learning rate with 0.1.Then again same procedure is followed for the input vector v1 to v5.

Table 2: Result of weight matrix class 2 with learning rate 0.4(Epoch 2)

	Input vector	Winning Neuron	Dist1	Dist2	Minimum	Updated weight	
						Δw1	Δw2
Iteration 1	v1 =[0,1,0,0,0]	B	2.32	0.75	0.75	No Change	[.218,.528,.038,.023,.007]
Iteration 2	v2 =[0,0,0,0,1]	A	0.52	1.31	.052	[.03,.06,.09,.42,1]	No change
Iteration 3	v3=[0,0,0,1,1]	A	0.34	2.26	0.34	[.018,.036,.054,.65,1]	No change
Iteration 4	v4=[1,0,0,0,0]	B	2.39	0.89	0.89	No change	[.531,.317,.023,.014,.004]
Iteration 5	v5=[0,0,0,0,0]	B	1.52	0.72	0.72	No change	[0.318,.19,.014,.014,.003]

After repeating the same procedure with decreasing learning rates, the result will be found zero (η=0) after 6 epoch and the values of all the input vectors will be same in class1 and class 2**.**

Table 3: Result of weight matrix class 2 with learning rate 0.0(Epoch 6)

	Input vector	Winning Neuron	Dist1	Dist2	Minimum	Updated weight	
						Δw1	Δw2
Iteration 1	v1 =[0,1,0,0,0]	B	2.32	0.66	0.66	No Change	[.326,.253,.002,.001,0]
Iteration 2	v2 =[0,0,0,0,1]	A	0.34	1.61	0.34	[.005,.009,.014,.583,1]	No change
Iteration 3	v3=[0,0,0,1,1]	A	0.17	2.16	0.17	[.005,.009,.014,.583,1]	No change
Iteration 4	v4=[1,0,0,0,0]	B	2.33	0.51	0.51	No change	[.326,.253,.002,.001,0]
Iteration 5	v5=[0,0,0,0,0]	B	1.34	0.17	0.17	No change	[.326,.253,.002,.001,0]

Map Decision for 5X5 input matrix and 2 classes

Here A and B based on weight class. In this example there are 2 weight classes so that in each epoch for v1 B is the winning neuron for v2 A is the winning neuron, for v3 A, for v4 B, and for v5 B is the winning neuron.

Table 4: Map table of input matrix and weight class 2

	A	B
v1	0	1
v2	1	0
v3	1	0
v4	0	1
v5	0	1

Here the input matrix is 5X5 and weight matrix is 5X2, it will be reducing in 5X2 output matrix and there will be three clusters only. c1 = {v2, v3}, c2 = {v4, v5} and c3 = {v1} have similar distance or nearest distance with each other.

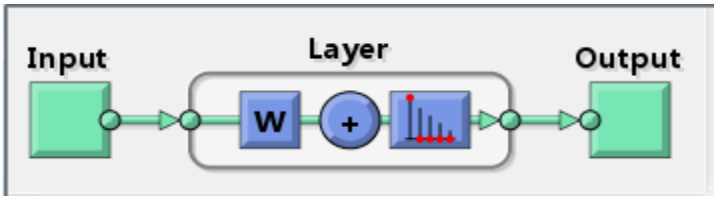


Figure 2: Architecture diagram of map table for weight class 2

2.1.2 Input vector 5X5 and weight class 3

Weight Matrix for class 3

Wij =

	J = 1 to 5				
w1j	.2	.4	.6	.8	1
w2j	.9	.7	.5	.3	.1
w3j	.3	.6	.9	.2	.4

In this example first take input vector v1 and weight of w1j (where j = 1 to 5) and find the distance1. Similar procedure performed for same input vector v1 with another weight class w2j (where j =1 to 5) and with third weight class wj3. After that find minimum distance between dist1, dist2 and dist3 as per the formula and find the new weight that is called updated weight for the second input vector. If dist1 is minimum then updated weight class will change Δw1, if dist2 minimum then updated weight class Δw2 will changed otherwise Δw3 will be changed.

Now the similar procedure is followed for the entire input vector v2, v3, v4 and v5 and finds the updated weight matrix.

Table 5: Result of weight matrix class 3 with learning rate 0.5(Epoch 1)

	Input vector	**Winning Neuron**	**dist1**	**dist2**	**dist3**	**Min**	**Updated Weight**		
							Δw1	Δw1	Δw3
Iteration 1	v1=[0,1,0,0,0]	B	2.4	1.25	1.26	1.25	No Change	[.45,.85,.25,.15,.05]	No Change
Iteration 2	v2=[0,0,0,0,1]	A	1.2	1.912	1.66	1.2	[.1,.2,.3,.4,1]	No Change	No Change
Iteration 3	v3=[0,0,0,1,1]	A	0.5	2.6	2.26	0.5	[.05,.1,.15,.7,1]	No Change	No Change
Iteration 4	v4=[1,0,0,0,0]	B	2.4	1.112	1.8	1.112	No Change	[.725,.425,.125,.075,.025]	No Change
Iteration 5	v5=[0,0,0,0,0]	B	1.5	0.72	1.4	0.72	No Change	[.362,.212,.062,.037,.012]	No Change

After completing one epoch same procedure will be followed for the next epoch and learning rate will be decreased by 0.1, this procedure will continue till learning rate becomes zero. Because at this stage weighted weight get a constant value.

Table 6: Result of weight matrix class 3 with learning rate 0.4(Epoch 2)

	Input vector	**Winning Neuron**	**out1**	**out2**	**out3**	**Min**	**Updated Weight**		
							Δw1	Δw1	Δw3
Iteration 1	v1 =[0,1,0,0,0]	B	2.3	0.75	1.26	0.75	No Change	[.217,.527,.037,.022,.007]	No Change
Iteration 2	v2 =[0,0,0,0,1]	A	0.5	1.31	1.66	0.5	[.03,.06,.09,.42,1]	No Change	No Change
Iteration 3	v3=[0,0,0,1,1]	A	0.34	2.26	2.26	0.34	[.018,.036,.054,.652,1]	No Change	No Change
Iteration 4	v4=[1,0,0,0,0]	B	2.3	.89	1.8	.89	No Change	[.530,.316,.022,.013,.004]	No Change
Iteration 5	v5=[0,0,0,0,0]	B	1.4	0.38	1.4	0.38	No Change	[.318,.189,.013,.008,.002]	No Change

Table 7: Result of weight matrix class 3 with learning rate 0.0(Epoch 6)

	Input vector	**Winning Neuron**	**out1**	**out2**	**out3**	**Min**	**Updated Weight**		
							Δw1	Δw1	Δw3
Iteration 1	v1 =[0,1,0,0,0]	B	2.3	.66	1.26	.66	No Change	[.325,.253,.001,.001,.000]	No Change
Iteration 2	v2 =[0,0,0,0,1]	A	.34	1.16	1.66	.34	[.004,.009,.013,.583,1]	No Change	No Change
Iteration 3	v3=[0,0,0,1,1]	A	.17	2.16	2.26	.17	No Change	No Change	No Change
Iteration 4	v4=[1,0,0,0,0]	B	2.33	.51	1.8	.51	No Change	No Change	No Change
Iteration 5	v5=[0,0,0,0,0]	B	1.34	.17	1.4	.17	No Change	No Change	No Change

After repeating the same procedure with decreasing learning rates, the result will be found the values of all the input vectors will be same in class1, class 2and class 3**.**

Map Decision for 5X5 input matrix and 3 classes

Here A and B based on weight class. In this example there are 3 weight classes so that in each epoch for v1 B is the winning neuron for v2 A is the winning neuron, for v3 A, for v4 B and for v5 B is the winning neuron. The third weight class is constant.

Table 8: Map table of input matrix and weight class 3

	A	B
v1	0	1
v2	1	0
v3	1	0
v4	0	1
v5	0	1

Here the input matrix is 5X5 and weight matrix is 5X3, it will be reducing in 5X2 output matrix and there will be one cluster only. c1 = {v2, v3}, c2 = {v4, v5} c3 = {v1} have similar distance or nearest distance with each other.

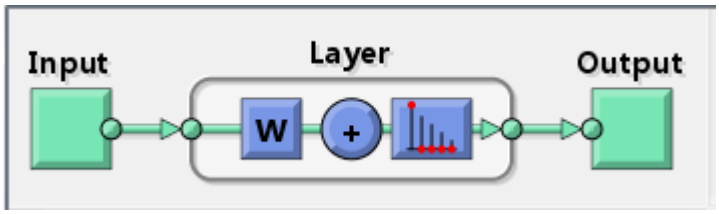


Figure 3: Architecture diagram of map table for weight class 3

2.1.3 For input vector 5X5 and weight class 4

Weight Matrix for class 4

Wij =

	j = 1 to 5				
w1j	.2	.3	.7	.8	.9
w2j	.9	.8	.6	.3	.1
w3j	.3	.6	.7	.2	.4
w4j	.1	.3	.4	.6	.9

In this example first take input vector v1 and weight of w1j (where j = 1 to 5) and find the distance1. Similar procedure performed for same input vector v1 with another weight class w2j (where j =1 to 5), with third weight class wj3 and w4j also. After that find minimum distance between dist1, dist2, dist3 and dist4 as per the formula and find the new weight that is called updated weight for the second input vector. If dist1 is minimum then updated weight class will change Δw1, if dist2 minimum then updated weight class Δw2 will change, if dist3 minimum then updated weight class Δw3 will change otherwise Δw4 will change.

Now the similar procedure is followed for the entire input vector v2, v3, v4 and v5 and finds the updated weight matrix.

Table 9: Result of weight matrix class 4 with learning rate 0.5(Epoch 1)

	Input vector	**Winning Neuron**	**out1**	**out2**	**out3**	**out4**	**Min dist**	**Updated Weight**			
								Δw1	Δw1	Δw3	Δw4
Iteration 1	v1 =[0,1,0,0,0]	C	2.4	1.3	.94	1.7	1.3	No Change	No Change	[.15,.8,.35,.1 ,.2]	No Change
Iteration 2	v2 =[0,0,0,0,1]	A	1.2	2.7	1.43	1.44	1.2	[.1,.15,.35,.4,.9 5]	No Change	No Change	No Change
Iteration 3	v3=[0,0,0,1,1]	A	.5	3.1	2.2	2.24	.5	[.05,.08,.17,.7,. 98]	No Change	No Change	No Change
Iteration 4	v4=[1,0,0,0,0]	B	2.3	1.11	1.5	2.34	1.11	No Change	[.95,.4,.3,.15,.0 5]	No Change	No Change
Iteration 5	v5=[0,0,0,0,0]	C	1.4	1.17	.83	1.64	.83	No Change	No Change	[.08,.4,.17,.0 5,.1]	No Change

After completing one epoch same procedure will be followed for the next epoch and learning rate will be decreased by 0.1, this procedure will continue till learning rate becomes zero. Because at this stage weighted weight get the constant value.

Table 10: Result of weight matrix class 4 with learning rate 0.4(Epoch 2)

	Input vector	Winning Neuron	out 1	out 2	out 3	out 4	Min dist	Updated Weight			
								Δw1	Δw1	Δw3	Δw4
Iteration 1	v1 =[0,1,0,0,0]	C	2.3	1.3	.40	1.2	.40	No change	No Change	[.05,.64,.1,.03,.06]	No Change
Iteration 2	v2 =[0,0,0,0,1]	A	.52	2.0	1.3	1.31	.52	[.03,.05,.1,.4 2,.99]	No Change	No Change	No Change
Iteration 3	v3=[0,0,0,1,1]	A	.35	2.7	2.2	2.25	.35	[.02,.03,.06,. 65,.99]	No Change	No Change	No Change
Iteration 4	v4=[1,0,0,0,0]	B	2.3	.27	1.3	2.14	.277	No Change	[.97,.24,.18 ,.09,.03]	No Change	No Change
Iteration 5	v5=[0,0,0,0,0]	C	1.4	1.0	.42	1.23	.42	No Change	No Change	[.03,.38,.06,.02,.0 4]	No Change

Table 11: Result of weight matrix class 4 with learning rate 0.0(Epoch 6)

	Input vector	Winning Neuron	out 1	out 2	out 3	Out4	Min	Updated Weight			
								Δw1	Δw2	Δw3	Δw4
Iteration 1	v1 = [0,1,0,0,0]	C	2.33	1.72	.34	1.15	.34	No Change	No Change	[.01,.43,.02,.0 ,.01]	No Change
Iteration 2	v2 = [0,0,0,0,1]	A	.35	1.9	1.2	1.21	.35	[.01,.01,.02,. 58,1]	No Change	No Change	No Change
Iteration 3	v3= [0,0,0,1,1]	A	.21	2.8	2.1	2.20	.21	No Change	No Change	No Change	No Change
Iteration 4	v4= [1,0,0,0,0]	B	2.3	.03	1.2	2.01	.03	No Change	[.98,.12,.09,.05 ,.02]	No Change	No Change
Iteration 5	v5=[0,0,0,0 ,0]	C	1.3	.99	.22	1.03	.22	No Change	No Change	[.01,.43,.02,.0 ,.01]	No Change

After repeating the same procedure with decreasing the learning rate, the result will be found and the values of all the input vectors will be same in class1, class 2, class 3 and class 4.

Map Decision for 5X5 input matrix and 4 classes

Here A, B, C and D based on weight class. In this example there are 4 weight classes so that in each epoch for v1 C is the winning neuron for v2 A is the winning neuron, for v3 A, for v4 B and for v5 C is the winning neuron.

Table 12: Map table of input matrix and weight class 4

	A	B	C
v1	0	0	1
v2	1	0	0
v3	1	0	0
v4	0	1	0
v5	0	0	1

In every epoch, the BMU or winning neuron will be same. As per this discussion the map is like this.
Here the 5X5 will be reducing in 5X3 output matrix and there are two clusters are.
c1 = {v1}, c2 = {v2, v3}, c3={v4} and c4 ={v5} have similar distance or nearest distance with each other.

Architecture of Output matrix for 4 classes SOM

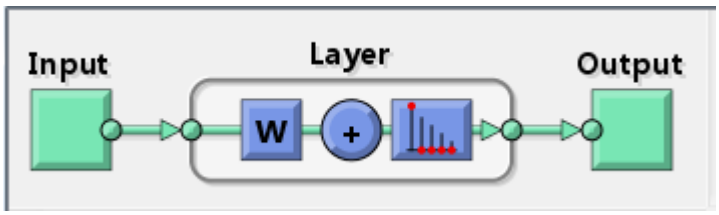


Figure 4: Architecture diagram of map table for weight class 4

3. RESULT ANALYSIS

The main aim of clustering is to reduce the data by grouping or categorizing them. There are different types of clustering methods. But here we apply partition clustering method. Clustering is used to reduce the data and to make a categorization. Partitioning cluster directly decomposes the data into a disjoint cluster.

In these problems, there are 5X5 input vectors. But every time on applying different weight classes on the same input vector, if the classes are n (n > 2) the result will come for only n-1 classes and the result of last class will be constant in each iteration and in each epoch.

Table 13: Topological mapping of all weight classes

Input Vector	Weight classes	Topology Map(output layer)
5x5	**5x2**	**5x2**
5x5	**5x3**	**5x2**
5x5	**5x4**	**5x3**

The topological neighboring decline monotonically, from a value less than half the largest diagonal of the map. **This is necessary condition for convergence**

Result analysis of this problem is that the topology size is not equivalent to the size of weight matrix. If weight size will be the increase there will be the change in topology.

CONCLUSION

In this paper, we observed that learning rates the change after every epoch. If learning rate is constant then the result could not be found or we cannot visualize the cluster. When learning rate becomes zero then only the value of weight matrix gets constant. The value of learning rate changes either by decreasing it by 0.1 or multiplying it by 0.5. In every epoch the winning neuron is same.

Moreover, in this paper, we also observed that when we calculate the distance for all vectors using one learning rate than one epoch is over. But for one epoch many iterations is performed. When we calculate the distance between one vector it is called iteration and when the same iteration is repeated for all vector it is called epoch.

Here we used different types of weight matrix such as 2, 3 and 4 for the same one-dimensional array. The value of weight matrix is changed based on iteration or the distance, we calculate the new weight for that class which has the minimum distance.

Thus, one can conclude that in the case of 2 class weight matrixes, we get only 2 output layers or winning neuron. For 3 class weight matrix we get only 2 output layers or winning neuron and for 4 class weight matrix, we get only 3 output layers or winning neuron. Here we observed that if weight matrix size is increased than one class is constant and we get output layer always minus one from weight matrix.

FUTURE WORK

Future work may further investigate on large data which has meaningful data items in the sets and find the variation on large data sets also. We will try to take multi-dimensional meaning full data and find the nearest clusters and minimize the data in terms of rows & columns.

In future work we investigate the effect of increase the weight matrix class on multi-dimensional data also. We will also work on learning rate and try to find the cluster to reduces the epoch and iterations and minimize the calculation.

REFERENCES:

1] “On the use of self-organizing maps for clustering and visualization” by Arthur Flexer.

[2] “Clustering of the Self-Organizing Map” by Juha Vesanto and Esa Alhoniemi IEEE transactions on neural networks, vol. 11, no. 3, pp 586-600, may 2000.

[3]”Application of SOM neural network in clustering” by Soroor Behbahani, Ali Moti Nasrabadi J. Biomedical Science and Engineering, 2009, 2, 637-643.

[4] “On the use of Three-dimensional Self-Organizing Maps for Visualizing Clusters in Geo-referenced Data” by Jorge M. L. Gorricha and Victor J. A. S. Lobo

[5] Application of Visual Clustering Properties of Self-Organizing Map in Machine-part Cell Formation Manojit Chattopadhyay, Pranab K. Dan, Sitanath Majumdar.

[6] “Improving Performance of Self-Organising Maps with Distance Metric Learning Method” by Piotr P lo´nski and Krzysztof Zaremba published 1407.1201v1 [cs.LG] 4 Jul 2014.

[7] “Clustering Internet Usage Behaviours with SOM Neural Networks” by U. Celenk, O. Ucan Proceedings of the World Congress on Engineering and Computer Science 2012 Vol II WCECS 2012, October 24-26, 2012.

[8] “Self-Organizing Map -based Document Clustering UsingWordNet Ontologies” by Tarek F. Gharib, Mohammed M. Fouad, Abdulfattah Mashat, Ibrahim Bidawi IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, January 2012 ISSN (Online): 1694-0814.

[9] “Clustering with SOM: u*c” by Alfred ultsch.