Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Centroid Neural Network based clustering technique using competetive learning

Selvakumar K, Assistant Professor
School of Computing Sciences
VIT University
Vellore, India
Email: kselvakumar@vit.ac.in

S.Prabu, Assistant Professor Senior
School of Computing Sciences
VIT University
Vellore, India
Email: sprabu@vit.ac.in

Abstract— **An unsupervised competitive learning algorithm based on the classical k-means clustering algorithm is proposed. The proposed learning algorithm called the centroid neural network (CNN) estimates centroids of the related cluster groups in training date. It is based on the observation that synaptic vectors converge to the centroids of clusters as learning proceeds in conventional unsupervised competitive learning algorithms such as SOM or DCL. The centroid, or conditional expectation, can minimize the mean-squared error of the vector quantization. As is the case with SOM or DCL, the synaptic vectors converge to the centroids of clusters as learning proceeds in CNN. However, the CNN finds locally optimal synaptic vectors for each datum presented and consequently converges to the centroids of clusters much faster than conventional algorithms. This centroid neural network can be used for generating the clusters using user data. The data may either image pixels or tables of information. One of the very advantageous features in the CNN algorithm is that the CNN does not require a schedule for learning coefficients. The CNN rather finds its optimal learning coefficient in each representation of data vectors. The CNN can also reward and punish by learning coefficients for winners and losers, respectively. Unlike SOM or DCL, the CNN also does not require the total number of iteration in advance.**

***Keywords*—Centroid, clustering, learning gain, neural network, unsupervised learning.**

I. INTRODUCTION

With the huge availability of Clustering algorithms [5], Centroid Neural Network based clustering has become a challenging task. For such a challenging task a feasible solution is attempted in this paper. An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances. Conventional competitive learning algorithms for unsupervised learning in artificial neural networks have been widely used for processing the input data of complicated classification tasks. One of the most widely used competitive learning algorithms is the k-means [2] clustering algorithm. Since the k-means clustering algorithm that minimizes the energy function defined by mean squared error is simple and fast enough to be performed in real time. The most serious problem with the k-means clustering algorithm is that the algorithm may not converge to an optimal solution for $k>1$.

Kohenen's self-organizing map (SOM) as an unsupervised learning algorithm [7] is one of the most popular neural-network algorithms and produces several variations for improving the performance of SOM. The SOM holds the very advantageous topology preserving property that can capture the probability distribution density of the input data without help of external supervision. It have a strong connection with the k-means algorithm. Basically, SOM finds a winner neuron which is the closest to a given input datum and updates the synaptic weights of the winner and its neighbors. In order to obtain the best results from SOM, the initial learning coefficient and the total number of iteration for a given set of data should be chosen carefully. Generally, the larger the predetermined total number of iterations is and the smaller the initial learning coefficient is, the better the results that can be expected.

The differential competitive learning (DCL) algorithm introduces concepts of reward and punishment to the competitive learning algorithm [6]. The DCL algorithm rewards the winner by adapting the synaptic vectors with a positive learning coefficient like SOM does, but it can punish the loser by adapting the synaptic vector with a negative learning coefficient. This concept of punishment has not been used in conventional competitive learning algorithms including SOM. Even though Kohonen uses this concept for the learning vector quantization (LVQ) system, the LVQ is a supervised learning algorithm. The DCL can be thought of as a local unsupervised approximation of Kohonen's supervised LVQ algorithm. However, choosing a schedule for optimal learning coefficients still remains unsolved in DCL.

The proposed learning algorithm, called centroid neural network (CNN) [3], is based on the observation that synaptic vectors converge to the centroids of clusters as learning proceeds in conventional unsupervised competitive learning algorithms such as SOM or DCL. The centroid, or conditional expectation, can minimize the mean-squared error of the vector quantization. As is the case with SOM or DCL, the synaptic vectors converge to the centroids of clusters as learning proceeds in CNN. However, the CNN finds locally optimal synaptic vectors for each datum presented and consequently converges to the centroids of clusters much faster than conventional algorithms.

One of the very advantageous features in the CNN algorithm [3] is that the CNN does not require a schedule for learning coefficients. The CNN rather finds its optimal learning coefficient in each representation of data vectors. The CNN can also reward and punish by learning coefficients for winners and losers, respectively. Unlike SOM or DCL, the CNN also does not require the total number of iteration in advance.

II. Problem Defenetion

Assume that N data vectors, x(i) , i =1….N, and M number of clusters, are given. Each of the datum is grouped as a member of one of the M clusters. When the cluster i has Ni members and x(i) denotes the data j in the cluster i for one instance, the problem and the energy function (or cost function), E is defined as follows

Find {Wi , 1 ≤ i ≤M} such that minimize E where W denotes a weight vector that has the same dimension as x.

A. *Centroid Calculation*

First, O = ▼mj €

= ∫ dj (x- mj ^) p(x)dx

= ∫ dj xp(x)dx - mj ^) p(x)dx ∫ dj p(x) dx

mj ^ = ∫ dj xp(x)dx / ∫ dj p(x)dx = Xj

Where

X → Input data.

mj^ → Optimal value.

P(x) → Probability Density Function.

B. *Alternate Centroid Estimation*

The following formula is used to calculate the centroid of each cluster.

$$\text{Centroid} = \text{Cm} = \sum_{i=1}^{N} (\text{tmi}) / N$$

Where

m → mTh Cluster

N → Number of Data present in the cluster

t → Tuple in mTh Cluster.

Here "epoch" as used in this paper is defined as "one presentation of whole data vectors in the data set," while "iteration" is defined as "one presentation of any data vector." For example, if we have N data vectors and P epochs of training have been performed, and then there were NP iterations of training have been performed.

C. *Distance Measures*

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances. However, even in this case the Euclidean distance can sometimes be misleading. Figure shown below illustrates this with an example of the width and height measurements of an object. Despite both measurements being taken in the same physical units, an informed decision has to be made as to the relative scaling. As the figure shows, different scaling can lead to different clustering's.

Notice however that this is not only a graphic issue: the problem arises from the mathematical formula used to combine the distances between the single components of the data feature vectors into a unique distance measure that can be used for clustering purposes: different formulas leads to different clustering's. Again, domain knowledge must be used to guide the formulation of a suitable distance measure for each particular application.

The joining or tree clustering method uses the dissimilarities (similarities) or distances between objects when forming the clusters. Similarities are a set of rules that serve as criteria for grouping or separating items. In the previous example the rule for grouping a number of dinners was whether they shared the same table or not. These distances (similarities) can be based on a single dimension or multiple dimensions, with each dimension representing a rule or condition for grouping objects.

For example, if we were to cluster fast foods, we could take into account the number of calories they contain, their price, subjective ratings of taste, etc. The most straightforward way of computing distances between objects in a multi-dimensional space is to compute Euclidean distances.

If we had a two- or three-dimensional space this measure is the actual geometric distance between objects in the space (i.e., as if measured with a ruler).However, the joining algorithm does not "care" whether the distances that are "fed" to it are actual real distances, or some other derived measure of distance that is more meaningful to the researcher; and it is up to the researcher to select the right method for his/her specific application.

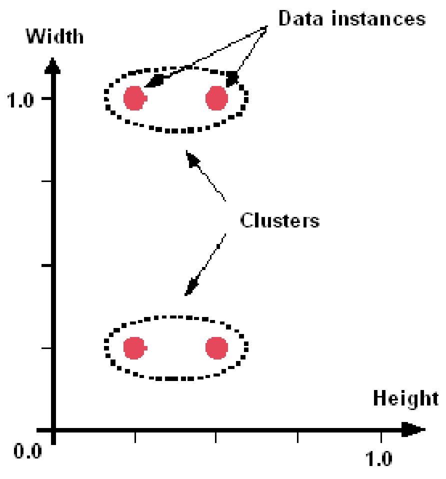


Figure 1. Before Scaling

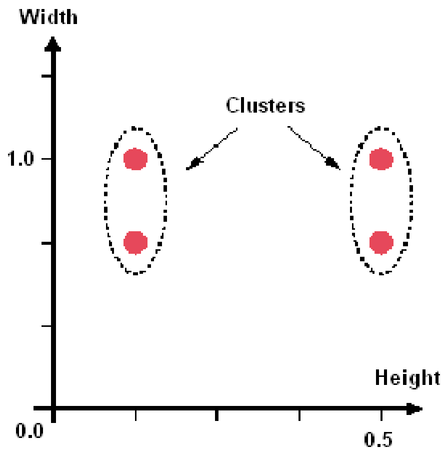


Figure 2. After Scaling

D. *Euclidean Distance*

This is probably the most commonly chosen type of distance. It simply is the geometric distance in the multidimensional space. It is computed as:

Distance (x,y) = { ∑i (Xi - Yi)2 }½

Note that Euclidean (and squared Euclidean) distances are usually computed from raw data, and not from standardized data. This method has certain advantages (e.g., the distance between any two objects is not affected by the addition of new objects to the analysis, which may be outliers). However, the distances can be greatly affected by differences in scale among the dimensions from which the distances are computed.

For example, if one of the dimensions denotes a measured length in centimeters, and you then convert it to millimeters (by multiplying the values by 10), the resulting Euclidean or squared Euclidean distances (computed from multiple dimensions) can be greatly affected (i.e., biased by those dimensions which have a larger scale), and consequently, the results of cluster analyses may be very different. Generally, it is good practice to transform the dimensions so they have similar scales.

III. OPTIMALITY CONDITIONS IN UNSUPERVISED LEARNING ALGORITHM

In order to achieve the minimum-energy clustering, the following conditions are considered.

A. *Nearest Neighbor Selection Condition:*

The "winner" neuron i for a given input X is defined as

$$\boldsymbol{w}_i = \min_j \|\boldsymbol{x} - \boldsymbol{w}_j\|^2, \qquad 1 \le j \le M.$$

B. *Minimum Energy Condition:*

The weights for a given output neuron, should be chosen in a way to minimize the total distance.

$$\boldsymbol{w}_i = \min_{\boldsymbol{w}} \sum_{j=1}^{N_i} \|\boldsymbol{x}_i(j) - \boldsymbol{w}\|^2 = \frac{1}{N_i}\sum_{j=1}^{N_i} \boldsymbol{x}_i(j)$$

C. *Norm Function:*

One of the mathematic formulas which can represent accurate value than the absolute function. Using this function we can able to calculate Centorid and Minimum energy values.

IV. THE CNN ALGORITHM AND CLUSTER ANALYSIS

The CNN algorithm is based on the conventional k-means algorithm and finds the centroid of data in corresponding clusters at each presentation of data vectors. Instead of calculating the centroids of the clustered data for every presentation of data, the CNN algorithm updates their weights only when the status of the output neuron for presenting data has changed: that is, the weights of a winner neuron in the current epoch for the data change only when the winner neuron did not win the data in the previous presentation and the weights of the winner neuron in the previous epoch for the data change only when the neuron does not win the data in the current epoch.

We call the former one "winner neuron" and the latter one "loser neuron." Note that "epoch" as used in this paper is defined as "one presentation of whole data vectors in the data set," while "iteration" is defined as "one presentation of any data vector." For example, if we have N data vectors and P epochs of training have been performed, then there were PN iterations of training have been performed. When a data vector is applied to the network at time, the adaptive equations for winner neuron and loser neuron in CNN can be written as follows:

Winner Neuron weight Calculation is follows

$$\begin{aligned} \boldsymbol{w}_j(n+1) &= \frac{1}{N_j+1}[N_j \boldsymbol{w}_j(n) + \boldsymbol{x}(n)] \\ &= \boldsymbol{w}_j(n) + \frac{1}{N_j+1}[\boldsymbol{x}(n) - \boldsymbol{w}_j(n)] \end{aligned}$$

Loser Neuron weight Calculation is

$$\begin{aligned} \boldsymbol{w}_i(n+1) &= \frac{1}{N_i-1}[N_i \boldsymbol{w}_i(n) - \boldsymbol{x}(n)] \\ &= \boldsymbol{w}_i(n) - \frac{1}{N_i-1}[\boldsymbol{x}(n) - \boldsymbol{w}_i(n)] \end{aligned}$$

Where Wj (n) and W (i) represent the weight vectors of the winner neuron and the loser neuron, respectively. In order to avoid the algorithm from getting stuck at a undesirable local minimum solution, CNN starts with setting the number of groups to two and increases the number of groups one by one until it reaches the predetermined number of group's M.

A. *Winner - Take - All Learning Algorithms*

This Learning rule differs substantially from the other rules. It can only be demonstrated and explained for an ensemble of neurons. Preferably arranged in a layer of P units. This rule is an example of competitive learning, and it is used for unsupervised network training. Typically, winner-take-all learning is used for learning statistical properties of inputs.

The learning is based on the premise that one of the neurons in the layer, say the m' th, has the maximum response due to input x, as shown in Figure 3. This neuron is declared the Winner As a result of this winning event, the weight vector W_m

$W_m = [W_{m1}, W_{m2}, \ldots. W_{mn}]$

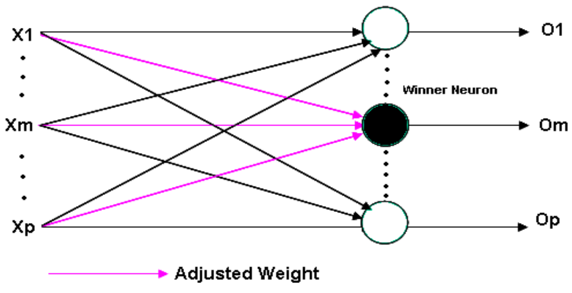


Figure 3. Winner-take-all learning rule

B. *Network Topology*

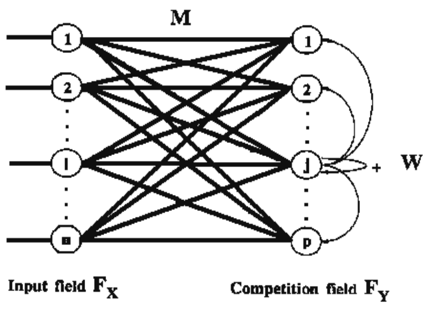


Figure 4. Network Topology

C. *Cluster Analysis*

The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other.

The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step.

After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the n data points from their respective cluster centers.

The following page represents a pseudo code of the CNN algorithm. The CNN algorithm does not provide any guarantee of convergence to the global minimum solution like other unsupervised algorithms such as SLA, DCL, and SOM, but does provide a guarantee of convergence to a local minimum. The convergence of the CNN algorithm can be easily proven by considering the energy change in each iteration of the training procedure.

D. Pseudo Code Representation Of CNN:

TABLE I. THE CENTROID NEURAL NETWORK ALGORITHM

```
**Algorithm CNN(M,N)**          [M:number of clusters, N: number of data vectors]
[Initialize weights $W_1$ and $W_2$]
Find the centroid, c, of all data vector
Initialize $W_1$and $W_2$ around c with small ε:
$W_1$:= c+ ε, $W_2$ := c+ ε
$N_1$ :=0 $N_2$:=0
K := 2, epoch :=0
for(k≤ N)
  do
     loser := 0
     for(n ≤ N)
         Apply a data vector x(n) to the network
         Find the winner neuron, j, in this epoch for $1 \le j \le k$
         If (epoch ≠ 0), then Set I is Winner neuron, I, for x(n) in previous epoch.
         If i ≠ j, then neuron, i, is loser neuron.
         If(epoch == 0 or i ≠ j)
           Run UpdateCNNWeight(x(n), Wi, Wj,epoch)
            loser := loser + 1
          endif
            n : = n+1
       endfor
         epoch  := epoch +1;
While loser ≠ 0
   If k ≠ M
     Split group with most error, j, by adding small vector, ε, nearby group j:
  $W_j = \min_i Ei = \min_i \sum_{K=1}^{Ni} || Xi (k) - Wi||^2, 1 \le I \le M$
   $W_{k+1}$ := $W_j$ + ε
endfor
end

**Procedure UpdateCNNWeight**(x, $W_i$, $W_j$ , epoch)
 Update winner neuron : $W_j (n) := W_j (n)+ 1/ N_j +1[x(n)- W_j (n)]$

  if (epoch ≠ 0)     [loser neuron occurred only when epoch ≠ 0]
        Update loser neuron : $Wi (n) := Wi (n) – 1/ N_j -1[x(n) – Wi (n)]$
  endif
end
```

E. Block Diagram

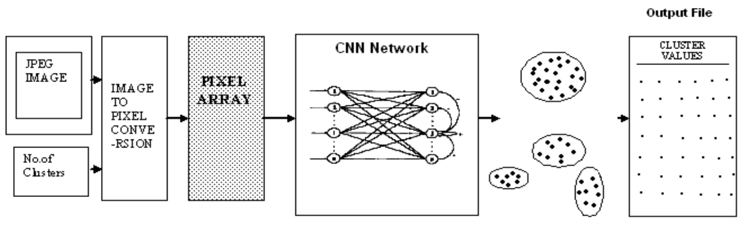


Figure 5. Diagram Representation of CNN

V. RELATIONSHIP OF CNN WITH SOME UNSUPERVISED COMPETITIVE LEARNING ALGORITHMS

From Winner and Loser neurons Equations, the adaptation rules of the CNN can be Combined such that

Wj(n+1)= Wj (n)+c(n)[x(n)- Wj (n)]

Where

$$c(n) = \begin{cases} (+1)\frac{1}{N_j+1}, & \text{If is a winner} \\ (-1)\frac{1}{N_j-1}, & \text{If is a loser and Previous winner for f(x)} \\ 0, & \text{Otherwise.} \end{cases}$$

Some of the conventional unsupervised learning algorithms to be compared with CNN include SLA, DCL, and SOM. The learning coefficient schedules for these algorithms are summarized. Note that SOM does not adapt only the weights of the winner neuron but also adapts the weights of the neighbor neurons to the winner neuron.

One of the most widely used neighborhoods is a sphere that covers a fairly large region initially and shrinks its diameter with time. When CNN is compared with SLA, SOM, and DCL, CNN and DCL allow a forgetting factor among some of the output neurons by using negative learning gain while SLA and SOM do not. In most cases, the forgetting factor is very natural and effective in unsupervised learning. The magnitude of the learning coefficient in the DCL, however, follows the idea of SLA and SOM whose learning rates are linearly decreasing with each iteration without considering the property of data and learning conditions.

A simple linearly decreasing learning coefficient, of course, does not satisfy the optimality conditions. The CNN, however, adapts its weights according to the optimality conditions. Of course, this does not mean any convergence of CNN to the global minimum.

This simply shows some relationships of CNN to other algorithms. By employing the strategy of starting the group numbers at two and increasing it until it reaches to the pre-specified number of groups, the CNN can improve its convergence and quality of solution greatly. TABLE II summarizes the schedules of learning gains and forgetting properties among different algorithms.

TABLE II. SUMMARY OF LEARNING GAIN FOR DIFFERENT ALGORITHMS

Algorithm	**Learning gain schedule**	**Forgetting allowed**
SLA	Linearly decreasing	No
SOM	Linearly decreasing	No
DCL	Linearly decreasing	Yes
CNN	Locally optimal	Yes

The following TABLE III which represents Comparison between learning co-efficient, and learning gain.

VI. EXPERIMENTS AND RESULTS

A. *Example Problems*

In order to investigate the effects of the total number (N) of epochs and initial learning gain, c(0),the experiments for SOM and DCL are performed with five different N's (10, 50, 100, 150, 200) and five different c(0)'s (0.1, 0.3, 0.5, 0.7, 0.9). Of course, CNN is performed only once for each data set since CNN requires neither N nor c(0).

The same initial weights around the middle of the data plane are set for the three algorithms. For both data sets, all the algorithms successfully converge to the centroids of possible groups of data. The differences among the algorithms are the speed of convergence and final energy levels.

The next example problem investigated with the proposed CNN algorithm is the uniform data in the 2-D plane. The 10 000 data points are uniformly distributed over the region enclosed by the boxed area and the task is to group the data into 25 clusters. The SOM and the proposed CNN are compared on this task. The c(0) and N for SOM are given by 0.2 and 100, respectively. Of course, CNN requires no initial parameters to be set. Initial weights are clustered around the center of the plot.

At each epoch of the training stage, the presentation order of the input data was randomized for both algorithms in order to generalize the results. With both algorithms, weights gradually spread out until the weight distribution approximates the uniform distribution. Results of convergence for 10000 uniformly distributed data points:

Figure 6 & 7 shows the final location of weights for SOM and CNN. An interactive CNN simulator with several problems is available at the following web site: http://icrl.myongji.ac.kr/NNsimulator/CNN.

TABLE III. COMPARISON IN LEARNING AMONG DIFFERENT LEARNING ALGORITHMS

Algorithm	C(n)	Note
SLA	1/Nj(n)	Nj: the number of members in neuron j
SOM	C(0)(1-n/N)	c(0): arbitrary constant
DCL	αc(0)(1- t/T) (+1) if Yj gets stronger (-1) if Yj gets weaker α = 0 if Yj remains same	T: data sequence, T: total number of data Yj : competitive signal of winner
CNN	(+1) 1/ Nj +1 if j is winner C(n)= (-1) 1/ Nj -1 if j is loser 0 Otherwise	Nj≠ 1

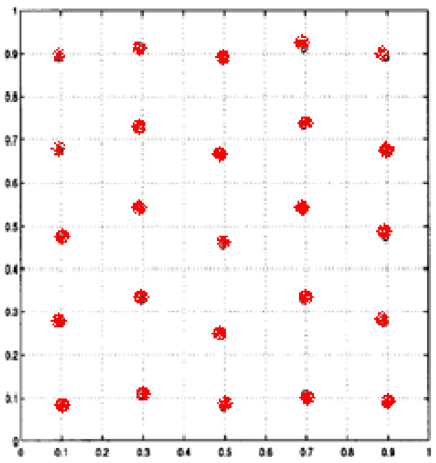


Figure 6. by SOM

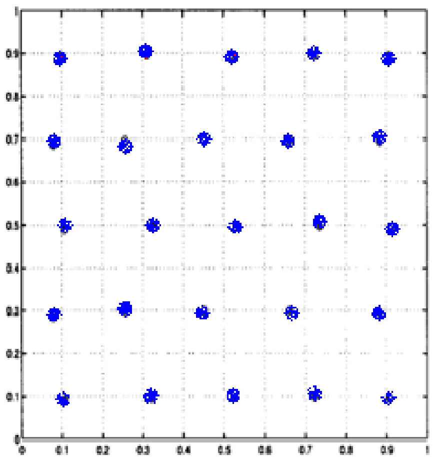


Figure 7. by CNN

B. Image Problems

Here the Centroid Neural Network (CNN) algorithm take input as image and generates the Image pixel Clusters. The given image is converted into image pixels. The user can represent Number of Cluster and the image what ever may be, whether it may black and white or color.

VII. CONCLUSION

The CNN algorithm based on the k-means clustering algorithm is proposed. The proposed CNN algorithm has a strong connection with some of the conventional unsupervised learning algorithms.

Even though the CNN algorithm is not new when we consider the k-means algorithm, the CNN provides us with a new interpretation of the k-means algorithm and the relationships with some of the conventional algorithms. While applying the CNN algorithm to several problems, solutions. The CNN algorithm is applied to several problems such as simple 2-D data problems and image compression problems. When compared with conventional clustering algorithms such as Kohonen’s self-organizing map and Kosko’s differential competitive learning on these problems, the proposed CNN algorithm produces comparable results with less computational effort and is free of the optimum parameter selection problem.

A. Application

The following areas are used CNN algorithm and produce effective results, such as

- Image Processing.
- Data grouping in Data Mining &
  Data Warehousing.
- Medical image processing.

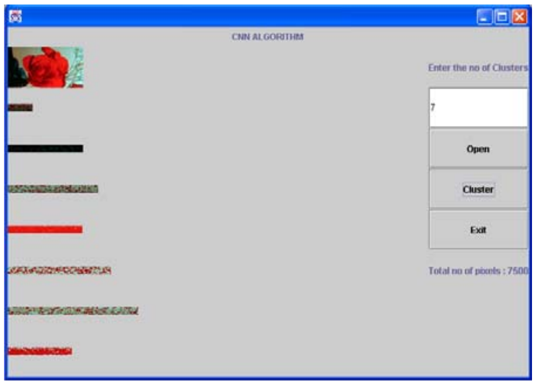
B. Advantages

The following advantages are there in CNN algorithm such as

- The CNN algorithm does not require the predetermined schedule for learning coefficient.
- And also a total number of iterations for clustering.
- Much faster than the SOM & DCL. i.e., half of the CPU time from SOM & DCL.

C. Dis-Advantages

Even though the CNN algorithm is not new when we consider the k-means algorithm, the CNN provides us with a new interpretation of the k-means algorithm and the relationships with some of the conventional algorithms. While applying the CNN algorithm to several problems, the CNN successfully converges to suboptimal solutions. The following Figure 8. shown the clusters with the given image.



7

Total no of pixels : 7500

Figure 8. Cluster Display Process

ACKNOWLEDGMENT

The authors would like to thank School of Computing Sciences, VIT University for providing us opportunity to carryout this research work.

REFERENCES

[1] B. Kosko, Neural Networks and Fuzzy Systems: A Dynamic Systems Approach to Machine Intelligence. Englewood Cliffs, NJ: Prentice-Hall,1991.

[2] C. Darken and J.Moody, Fast adaptive K-means clustering: some empirical results” in Proc.1990 Int. Joint Conf. Neural Networks Vol.2,1990 pp 233-238.

[3] Dong-Chul Park,” Centriod Neural Network for Unsupervised Competitive Learning” IEEE transactions on neural networks, vol. 11, no. 2, march 2000.

[4] K. Haese, “Self-organizing feature maps with self-adjusted learning parameters,” IEEE Trans. Neural Networks, vol. 9, Nov.1998.

[5] Harigan, Clustering algorithms, New York: Wiley ed 1975.

[6] S. Kong and B. Kosko, “Differential competitive learning for centroid estimation and phoneme recognition,” IEEE Trans. Neural Networks,vol. 2, pp. 118–124, Jan. 1991.

[7] T. Kohonen, Self-Organization and Associative Memory, 3rd. Berlin, Germany: Springer-Velag, 1989.

[8] T. Villmann et al., “Topology preservation in self-organizing feature maps,” IEEE Trans. Neural Networks, vol. 8, pp. 256–266, Mar. 1997.

[9] O. Chen, B. Sheu, and W. Fang, “Image compression using self-organization networks,” IEEE Trans. Circuits,Syst. Video Technol.,vol. 4, pp.480–489, Oct. 1994.