Information Technology Journal 5 (3): 551-559, 2006
ISSN 1812-5638

A Comparative Study of Clustering Algorithms

[1]Satchidanandan Dehuri, [2]Chinmay Mohapatra, [3]Ashish Ghosh and [4]Rajib Mall
[1]Department of Information and Communication Technology,
Fakir Mohan University, Vyasa Vihar, Balasore, India
[2]School of Mathematics, Statistics and Computer Science, Utkal University, Vani Vihar, India
[3]Machine Intelligence Unit, Indian Statistical Institute, 203, B.T. Road, Kolkata-3700108, India
[4]Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur-721302, India

Abstract: Data clustering is an unsupervised task that can generate different shapes of clusters for a particular type of dataset. Hence choosing an algorithm for a particular type of dataset is a difficult problem. This study presents the choice of an appropriate clustering algorithm by a comparative study of three representative techniques like K-means, Kohonen's Self Organizing Map (SOM) and Density Based Spatial Clustering of Applications with Noise (DBSCAN) based on the extensive simulation studies. Comparison is performed on the basis of cluster quality index 'β', percentage of samples correctly classified and CPU time. The experimental results show that if the clusters are of arbitrary shape, a density based clustering algorithm like DBSCAN is preferable, where as if the clusters are of hyper spherical or convex shape and well-separated then the SOM or K-means is preferable.

Key words: Data clustering, DBSCAN, SOM, K-means

INTRODUCTION

Recently there has been an enormous growth in the amount of commercial and scientific data, such as protein sequence, retail transaction and web logs (Fayyad *et al*., 1996). The questions arise, why so much of data? People store data because they think some valuable assets are implicitly coded within it. But raw data is rarely of direct benefit. Discovering groups of similar data items in the data set, known as cluster analysis, is an interesting and challenging issue. There are many different data clustering algorithms emerged from several disciplines such as statistics, pattern recognition and machine learning (Jain *et al*., 1999). They give different types of clusters for a particular type of dataset. Some of them require some additional prior knowledge about the nature of the data. Each clustering algorithm works well only for certain types of data and with certain applications. The choice of an appropriate clustering algorithm can be made from a comparative study of the clustering algorithms.

Furthermore, the comparison is necessary as it is widely used in many areas like information retrieval and text mining (Cutting *et al*., 1992; Steinbach *et al*., 2000; Dhillon *et al*., 2001), spatial database applications, e.g., Geographical Information System (GIS) or astronomical data (Xu *et al*., 1998; Sandar *et al*., 1998; Ester *et al*., 2000), sequence and heterogeneous data analysis (Cadez *et al*., 2001), web applications (Heer *et al*., 2001; Foss *et al*., 2001), DNA analysis in computational biology (Ben-Dor *et al*., 1999) and many others.

The clustering problem is the problem of dividing a given set $\{x_1,..., x_N\}$ of N data points into several non-overlapping homogenous groups. Each such group or cluster should contain similar data items and data items from different groups should not be similar. An k clustering of a data set X is the partition of X into k clusters, $C_1,...,C_k$ such that the following three conditions are met:

i) $C_i \neq \phi$, i = 1,...,k
ii) $\cup^k_{i=1}$ (except outliers)
iii) $C_i \cap C_j = \phi$, i ≠ j, i, j = 1,...,k. (sometimes violated)

The various data clustering algorithms can be classified into partitional, density based and clustering using artificial neural networks. In this work, the three representative algorithms like K-means (Hartigan *et al*., 1979; Hartigan, 1975), DBSCAN (Ester *et al*., 1996) and Kohonen Self-organizing Map

Corresponding Author: Satchidanandan Dehuri, Department of Information and Communication Technology, Fakir Mohan University, Vyasa Vihar, Balasore, India Tel: 91-9437381494

(SOM) (Kohonen, 1990) corresponds to partitional, density based and artificial neural networks are taken for comparative study.

The comparison is performed on the basis of cluster quality index 'β', percentage of samples correctly clustered and CPU time. Mathematically, it is defined as

$$\beta = \frac{\sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(X_{ij}-\bar{X}\right)^T\left(X_{ij}-\bar{X}\right)}{\sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(X_{ij}-\bar{X}_i\right)^T\left(X_{ij}-\bar{X}_i\right)}$$

where, n_i is the number of points in the ith (i = 1,...,k) cluster, X_{ij} is the feature vector of the jth pattern (j = 1,...,n_i) in cluster I, $\bar{X}_i$ is the mean of n_i patterns of the ith cluster, n is the total number of patterns and $\bar{X}$ is the mean value of the entire set of patterns. 'β' is the ratio of the total variation and within-cluster variation. For a given data and number of clusters k, the higher the homogeneity within the clustered regions, the higher would be the β value.

CLUSTERING ALGORITHMS

This technique produces clusters by optimising a criterion function defined either locally or globally. The user should have prior knowledge of the number of clusters. The algorithm runs multiple number of times with different starting points and the best configuration obtained from all of the runs is used as the output clustering. Let us discuss each of the clustering algorithms.

K-means: The k-means algorithm (Hartigan *et al*., 1979; Hartigan, 1975) is the most popular clustering tool used in scientific and industrial applications. The name comes from representing each of k clusters c_j by the mean (or weighted average) x_j of its points, the so-called centroid. While this obviously does not work well with categorical attributes, it has the good geometric and statistical sense for numerical attributes. The sum of discrepancies between a point and its centroid expressed through appropriate distance measure is used as the criterion function. For example, the L_2-norm based objective function, the sum of the squares of errors between the points and the corresponding centroids, is equal to the total intra-cluster variance

$E(c_j) = \sum_{i=1}^{k}\left\|x_i - \mu_j\right\|^2$, where k is the number of points in cluster j.

The sum of the squares of errors can be rationalized as (a negative of) log-likelihood for normally distributed mixture model and is widely used in statistics (SSE). Therefore k-means algorithm can be derived from general probabilistic framework. The inter-cluster variance is measured as

$$E(c_j) = \sum_{j=1}^{k}\sum_{x_i \in c_j}\left\|x_i - \mu_j\right\|^2$$

where, k is the number of clusters. Two version of k-means iterative optimisation are known. The first version is known as Forgy's algorithm (Forgey, 1965) and consists of the following steps:

1. Choose the number of clusters, k.
2. Set initial centres of clusters, $c_1, c_2, \ldots c_k$ to the arbitrarily selected k vectors from the dataset.
3. Classify each vector $x_l = [x_{l1}, x_{l2}, \ldots x_{ld}]$ (d is the dimension of the input vectors) into the closest centre c_i by Euclidean distance measure:
$$\left\|x_l - c_i\right\| = \min\left\|x_l - c_j\right\|.$$
4. Recompute the estimates for the cluster centres c_i Let $c_i = [c_{i1}, c_{i2}, \ldots c_{id}]$, c_{im} is computed by
$$c_{im} = \frac{\sum_{x_{li} \in cluster_i} x_{l_i m}}{n_i}.$$
where n_i is the number of vectors in the ith cluster.
5. If none of the cluster centres (c_i, i = 1,2, …,k) changes in step 4, stop; otherwise go to step 3.

The second version (classic in iterative optimisation) of k-means iterative optimisation reassigns points based on more detailed analysis of effects on the objective function caused by moving a point from its current cluster to a potentially new one. If a move has a positive effect, the point is relocated and the two centroids are recomputed. It is not clear that this version is computationally feasible, because the outlined analysis requires an inner loop over all member points of involved clusters affected by centroids shifts. However, in L_2 case it is known (Duda *et al*., 1973) that all computations can be algebraically reduced to simply computing a single distance! Therefore, in this case both versions have the same computational complexity.

The wide popularity of k-means algorithm is well deserved. It is simple, straightforward and is based on the firm foundation of analysis of variances. The K-means algorithm also suffers from all the usual suspects: i) the result strongly depends on the initial guess of centroids

(or assignments), ii) it is not obvious what is a good k to use, iii) the process is sensitive with respect to outliers, iv) the algorithm lacks scalability, iv) only numerical attributes are covered and v) resulting clusters can be unbalanced.

Kohonen's Self-Organizing Map (SOM): Kohonen self-organizing map (Kohonen, 1990) is an unsupervised, competitive learning, clustering network, in which only one neuron (or only one neuron in a group) is "on" at a time. Self-organizing maps (SOMs) can be used as a data visualization technique as they can reduce the dimensions of data through the use of self-organizing neural networks while preserving their topological structure. The problem that data visualization attempts to solve is that humans simply cannot visualize high dimensional data as is so techniques are created to help us understand this high dimensional data. The way SOMs go about reducing dimensions is by producing a map of usually one or two dimensions, which plot the similarities of the data by grouping similar data items together. The plot is drawn by giving each output neuron a gray scale colour that is proportionate to its weight vectors average distance from it immediate neighbours weight vectors. Thus SOMs accomplish two things, they reduce dimensions and display similarities.

Basic architecture: Figure 1 shows the architecture of Kohonen SOM. It contains a single layer of neurons in addition to an input layer of branching nodes. There are 'k' neurons in the neural layer and each has a parametric weight vector w_k of dimension 'n', which is same as the dimension of the input feature vectors $\vec{x} = (x_1, x_2, ..., x_n)$. The weight vectors $\vec{w}_1, \vec{w}_2, ..., \vec{w}_k$ are randomly initialised in the feature space at the beginning. One exemplar input vector $\vec{x}$ is selected from the sample and put into the network and squared distances between $\vec{x}$ and each $\vec{w}_i$ are computed by Euclidean distance. The minimum distance is then determined to obtain the neuron that is the winner over the other neurons. In this network the weight vectors are multiplied by three different strategies.

- In the first strategy called winner-take-all strategy, the winning neuron updates its parametric weight vector. All other neurons keep their old values.
- In second strategy, update positively (reinforce, or reward) all neurons that are close to the winning neuron and to update negatively all of those neurons that are farther way from the winner.
- In the third strategy, when a vector $\vec{x}$ is presented to a Kohonen network, the dot product $y_k = \vec{x} \cdot \vec{w}_k$ is computed as output from each 'k' neurons.

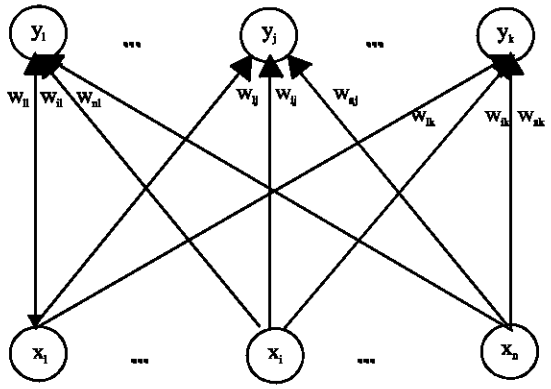

Fig. 1: Kohonen self-organizing map

R = 2	R = 1	#R = 0	R = 1	R = 2

Fig. 2: Linear topology

R = 2	R = 2	R = 2	R = 2	R = 2
R = 2	R = 1	R = 1	R = 1	R = 2
R = 2	R = 1	#R = 0	R = 1	R = 2
R = 2	R = 1	R = 1	R = 1	R = 2
R = 2	R = 2	R = 2	R = 2	R = 2

Fig. 3: 2-D rectangular grid topology

Neighbourhoods of a unit designated by # of radii R = 2, 1 and 0 in a one-dimensional topology with 5 cluster units is shown in Fig. 2.

A neighbourhood for a 2D planar rectangular grid topology is shown in Fig. 3.

Algorithm SOM

1. Fix the number of output neurons k, each of which represents a cluster.
2. Initialise the weight vector for each output neuron with i) random values, or ii) the first k input vectors.
3. Set topological neighbourhood parameters and learning rate parameter α.
4. While stopping condition is false
5. For each input x
6. Update weight vector w_j of the closest output neuron and its neighbors as follows:
   $w_{ji}(new) = w_{ji}(old) + \alpha.[x_i - w_{ji}(old)]$, i = 1,..., n.
7. End for
8. Reduce learning rate
9. Reduce radius of neighbourhood at specified intervals.
7. End while

The learning process has two separate but related phases, that is, the ordering phase and the convergence phase. During the ordering phase, the learning rate should be set close to unity and then gradually decreased, but not allowed to go below a certain threshold value (usually 0.1). It is during this part of the learning process that the topological ordering of the weight vectors is carried out. The convergence phase is the second phase of the learning that is generally the longest part of the network learning (typically 80% of the epochs). The remaining iterations are necessary during this phase for carrying out fine adjustments of the map i.e., the weight vectors. In this phase the learning parameter should have very small values for a long time. Typically, it is slowly reduced from 0.1 to zero. The neighbourhood radius is kept relatively large at the beginning of the training and then shrunk monotonically with the epochs to zero. The learning rule drags the weight vector associated with the winning unit towards the input vector and it also drags the weights of the closest unit (i.e., those within it neighbourhood radius) along with it. We can imagine the working of the SOM as an elastic net in the input space that wants to come as close as possible to the inputs of the network. The elastic net has the topology of the output array and the points of the net can be thought of as having the weights as coordinates.

DBSCAN: In the Euclidean space a set can be divided into a subset of connected components. The implementation for partitioning a set of points requires concepts of density, connectivity and boundary. A cluster, defined as a connected dense component, grows in any direction that density leads. Therefore density-based algorithms are capable of discovering clusters of arbitrary shapes. Also this provides a natural protection against outliers. Figure 4 shows some cluster shapes that present a problem for partitioning relocation clustering (e.g., k-means), but are handled properly by density-based algorithms. They also have good scalability.

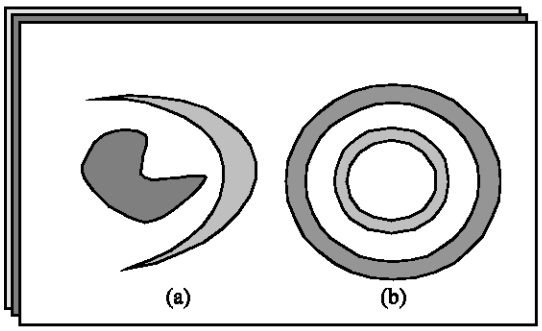

Fig. 4: Irregular shapes of clusters

The algorithm DBSCAN (Density Based Spatial Clustering of Applications with Noise) (Ester *et al*., 1996) targeting low dimensional spatial data is the major representative in density-based connectivity. The other algorithm of this category includes GDBSCAN (Sandar *et al*., 1998) and DBCLASD (Ester *et al*., 2000). The basic ideas of density-based clustering involve a number of new definitions. We intuitively present these definitions and then follow up with a high-level algorithm.

- The ε-neighbourhood of a point p, denoted by p and $N_\varepsilon(P)$ is defined as $N_\varepsilon(P) = \{q \in D \mid d(p, q) \leq \varepsilon\}$.
- There are two kinds of points in a cluster, points inside of the cluster (core points) and points on the border of the cluster (border points). In general, ε-neighbourhood of a border point contains significantly less points than ε-neighbourhoods of a core point.
- A point p is directly density-reachable from a point q with respect to ε, MinPts if

i. $p \in N_\varepsilon(q)$ and
ii. $|N_\varepsilon(q)| \geq$ MinPts (core point condition).

- A point p is density reachable from a point q with respect to ε and MinPts if there is a chain of points $p_1,\ldots,p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i.
- A point p is density connected to a point q with respect to ε and MinPts if there is a point O such that both, p and q are density-reachable from O with respect to ε and MinPts.

To find a cluster, DBSCAN starts with an arbitrary point p and retrieves all points density-reachable from p with respect to ε and MinPts. If p is a core point, this procedure yields a cluster. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the dataset.

Algorithm DBSCAN

Input: i) A dataset denoted as SetOfPoints of points that are UNCLASSIFIED.
ii) The global density parameters ε and MinPts.

Output: Dense clusters and noise points.

1. ClusterId = 1;
2. For i = 1 to SetOfPoints.size do
3. Point = SetOfPoints.get(i);
4. If Point.Clid = UNCLASSIFIED then
5. If ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts) then
6. ClusterId = ClusterId + 1;

7. End If
8. End If
9. End For

ExpandCluster(SetOfPoints, Point, Clid, ε, MinPts)
1. seeds = SetOfPoints.regionQuery(Point, ε);
2. If seeds.size < MinPts then
3. SetOfPoints.changeClid(Point, NOISE);
4. Return FALSE;
5. Else
6. SetOfPoints.changeClid(seeds, Clid);
7. seeds.delete(Point);
8. While seeds ≠ Empty do
9. currentP = seeds.first();
10. result = SetOfPoints.regionQuery(currentP, ε);
11. If result.size ≥ MinPts then
12. For i = 1 to result.size do
13. resultP = result.get(i);
14. If resultP.Clid in (UNCLASSIFIED, NOISE) then
15. If resultP.Clid = UNCLASSIFIED then
16. seeds.append(resultP);
17. End If
18. SetOfPoints.changeClid(resultP, Clid);
19. End If
20. End For
21. End If
22. seeds.delete(currentP);
23. End While
24. Return TRUE;
25. End If

EXPERIMENTAL STUDIES

Description of the dataset: Experimental studies are performed on two artificially created data sets named as Art_data_1 and Art_data_2 and one real life dataset named as IRIS (Blake *et al*., 1998).

Art_data_1: This data set contains two non-convex clusters in 2D. Cluster 1 contains 7534 samples and cluster 2 contains 7346 samples. Each sample has two features. The underlying dataset contains two non-overlapping clusters. Figure 5 shows the distribution of dataset.

Art_data_2: This dataset contains two linearly separable clusters of 500 samples each. The samples are a vector of three numeric feature values i.e. all the samples are in 3D. The classes are cubic shaped, with one having its center at (4,4,4) and sides of length 1 and the other having its centre at (6.5,6.5,6.5) and sides of length 1.5. The two clusters are fairly close to each other but not overlap.

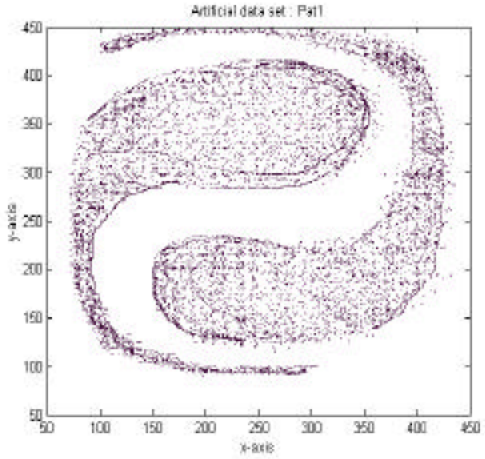

Fig. 5: Art_data_1 artificial data set

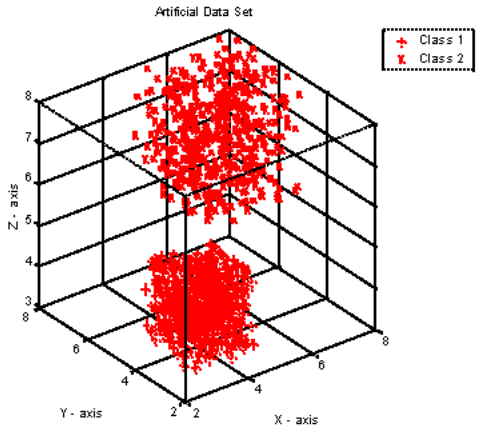

Fig. 6: Art_data_2 artificial data set

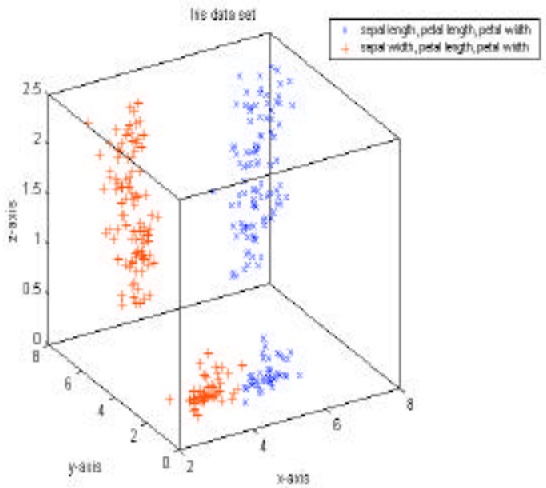

Fig. 7: IRIS data set

They are also of much different densities. The Art_data_2 is shown in Fig. 6.

Iris data set: The Iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. There are 4 attributes-sepal length, sepal width, petal length and petal width. One class is linearly separable from the other 2; the latter are not linearly separable from each other. To visualize the IRIS data set, the 4 attributes have been plotted in 2 triplets, as shown in Fig. 7.

RESULTS

The percentage of samples correctly clustered is calculated by using the true class labels of the samples as given in the data set. Each cluster discovered is assigned with the class label of the class to which the majority of the samples in it belong. Then the percentage of samples in clusters with the same class label as the actual class of the sample is calculated. The CPU time is obtained on a Pentium III 500MHz workstation with 256MB RAM.

K-means: Table 1 gives the results of the K-means algorithm obtained from all three data sets by taking 1000 iterations.

Self organizing feature map: Figure 8-10 shows the topographical map of 30 by 30-rectangular grid of output neurons obtained from three datasets. In this method, initially all the weight vectors are initialised to the first sample of the dataset and in the successive iterations the weights are adjusted. Table 2 gives the results of the SOM algorithm using all three data sets by taking 1000 iterations.

DBSCAN: The sorted k-dist graphs for the three data sets are shown in Fig. 11-13.

Based on the estimate made using the sorted k-dist graphs and some trail and error search around that value, fairly good density parameter values were chosen for each data set. The results of clustering by DBSCAN using those parameter values are shown in Table 3.

Comparative analysis: Table 4 shows the comparative performance of the three clustering algorithms, K-Means, SOM and DBSCAN, for each of the three data sets. DBSCAN could discover the clusters in the Art_data_1 data set with good accuracy while K-means and SOM were unable to cluster it properly. This shows that DBSCAN can detect non-convex clusters while K-means and SOM cannot. However, the DBSCAN algorithm takes considerably more time than K-means and SOM. The

Table 1: Results of K-Means algorithm

	Initialisation				
Data set	No. of clusters (k)	Initial weight vectors	Cluster quality (β)	Percent correct	CPU time (sec)
Art_data_1	2	1st two samples: (302,100), (301,100).	1.5714675087911916	63.66	50
Art_data_2	2	1st two samples:	3.8910738021701077	99.9	6
Iris	3	1st three samples: (6.7,3.0,5.2,2.3), (6.0,2.2,5.0,1.5), (6.2,2.8,4.8,1.8).	8.624027263474284	88.67	3

Table 2: Results of SOM algorithm

	Initialisation				
Data set	No. of clusters (k)	Initial cluster centers	Cluster quality (β)	Percent correct	CPU time (sec)
Art_data_1	2	1st two samples: (302,100), (301,100).	1.7481265202402723	57.25	21
Art_data_2	2	1st two samples: (3.4279,3.5521,3.6859), (4.287,3.7369,4.9356).	3.8910738021701077	99.9	4
Iris	3	1st three samples: (6.7,3.0,5.2,2.3), (6.0,2.2,5.0,1.5), (6.2,2.8,4.8,1.8).	8.624488765260416	99.9	4

Table 3: Results of DBSCAN algorithm

	Initialisation		No. of samples classified as	No. of clusters			
Data set	ε	MinPts	noise	found	Cluster quality (β)	Percent correct	CPU time (sec)
Art_data_1	12	4	0	2	1.1746	99.99	168
Art_data_2	0.57	4	0	2	3.8886	100.0	8
Iris	0.42	5	29	3	12.095994	78.0	4

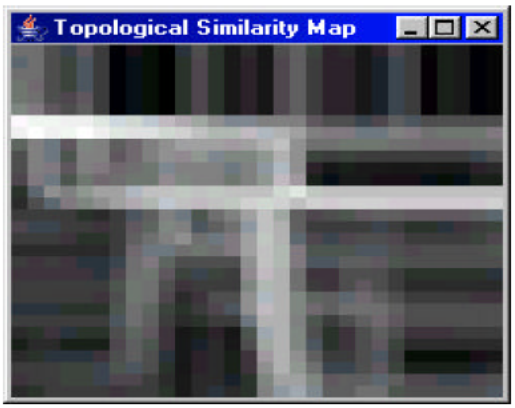

Fig. 8: Topological similarity map for Art_data_1 (Number of epochs: 1000)

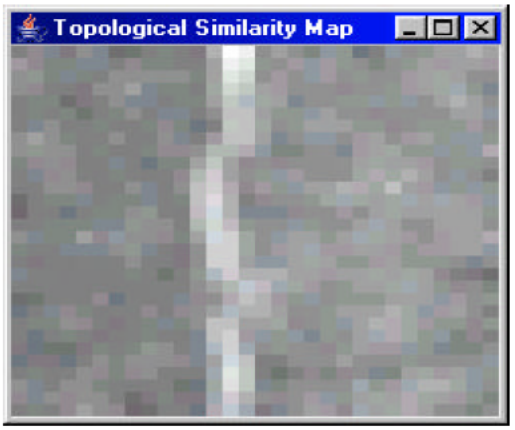

Fig. 9: Topological similarity map for Art_data_2 (Number of epochs: 1000)

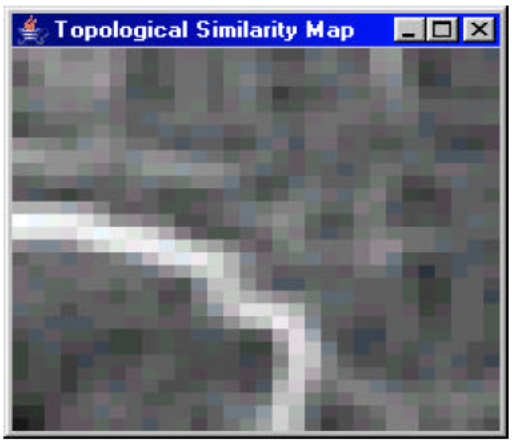

Fig. 10: Topological similarity map for IRIS (Number of epochs: 1000)

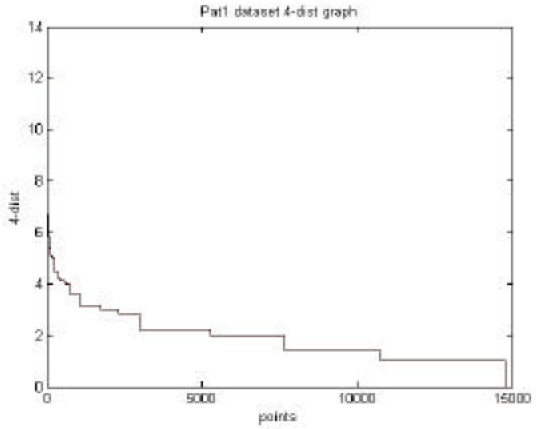

Fig. 11: Sorted 4-dist graph for Art_data_1

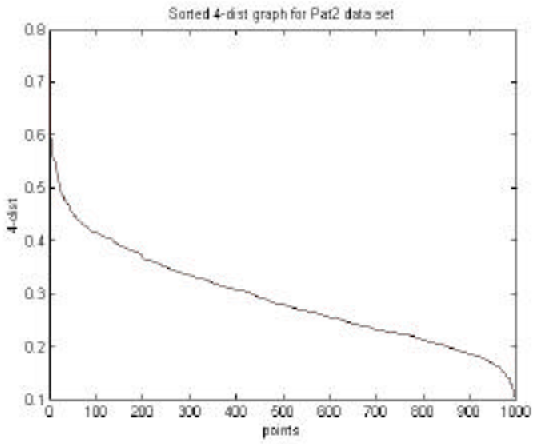

Fig. 12: Sorted 4-dist graph for Art_data_2

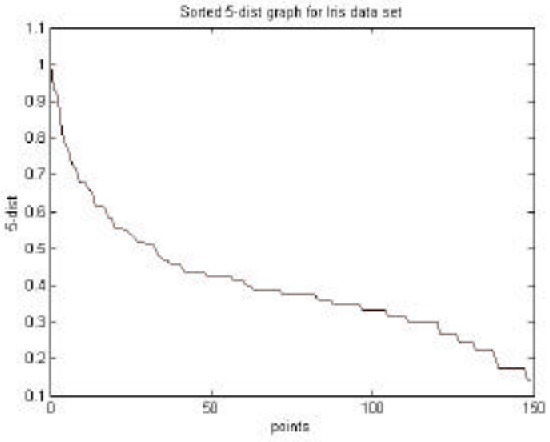

Fig. 13: Sorted 4-dist graph for IRIS

Table 4: Comparative performance of clustering algorithms

Algorithm	No. of clusters found	Cluster quality (β)	Percent correct	CPU time (sec)
Art_data_1 artificial data set				
K-Means	2	1.7481265202402723	57.25	21
SOM	2	1.5714675087911916	63.66	50
DBSCAN	2	1.1745995850120072	99.99	168
Art_data_2 artificial data set				
K-Means	2	3.8910738021701077	99.9	4
SOM	2	3.8910738021701077	99.9	6
DBSCAN	2	3.8885998293751793	100.0	8
IRIS data set				
K-Means	3	8.624488765260416	99.9	4
SOM	3	8.624027263474284	88.67	3
DBSCAN	3	12.095993612888822	78.0	4

cluster quality index β for DBSCAN is lower because the actual clusters were not very homogeneous. The well separated convex clusters in Art_data_2 data set were discovered by all three algorithms equally well with DBSCAN taking the most CPU time followed by SOM and then K-means. Again DBSCAN has a slightly lower β value but that resulted in a higher percentage of samples getting correctly clustered.

For the Iris data set, where two of the clusters are not linearly separable, DBSCAN has the least percentage of samples correctly classified as it misclassified some of the sample as noise when there is no noise in the data set. However, its β value is much higher than the others because it removed some of the input data that were increasing the non-homogeneity as noise. So, DBSCAN is able to cluster non-linearly separable clusters better than or as well as K-means and SOM, particularly when the clusters are not convex or linearly separable.

It is difficult to determine the actual number of clusters for K-means and SOM particularly when the clusters are non-convex or non-linearly separable. The topological maps produced by the SOM can be used to determine the number of clusters when the clusters are well separated and linearly separable, but they are unable to show clusters that are not linearly separable. The number of clusters in Art_data_1 is not apparent from its topological map, which shows 3 separate regions of high similarity. Similarly, for Iris data set, only one region of high similarity is visible in its topological map, corresponding to the single linearly separable cluster in the data set. The other region does not show a high similarity within it. The topological similarity map for Art_data_2 data set correctly shows the two clusters in the data set.

CONCLUSIONS AND FUTURE WORK

This study presents a comparative study of the clustering technique like K-means, SOM and DBSCAN based on the extensive simulation using the data set Art_data_1, Art_data_2 and IRIS. Our study indicate that DBSCAN is better than K-means and SOM in discovering non-convex clusters and it is as good or better than K-means and SOM in extracting convex clusters. It can also detect noise. However, DBSCAN algorithm also takes much more CPU time for large data sets. When the clusters are of arbitrary shape (e.g., Art_data_1 data set), a density based clustering algorithm like DBSCAN is preferable. On the other hand, when the clusters are of hyperspherical or convex shape and well separated and the data set is large, then SOM or K-means may be preferable as they are faster.

Future work includes comparing the performance of K-means, SOM, DBSCAN and hierarchical clustering algorithms using incomplete data sets.

REFERENCES

Ben-Dor, A. and Z. Yakhini, 1999. Clustering Gene Expression Patterns. In Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB 99), Lyon, France, pp: 11-14.

Blake, C., L. Blake and C.J. Merz, 1998. UCI Repository of Machine Learning Databases, (http://www.ics.uci.edu/).

Cadez, I., P. Smyth and H. Mannila, 2001. Probabilistic Modeling of Transactional Data and Applications to Profiling, Visualization and Prediction. In proceedings of the 7th ACM SIGKDD, San Francisco, CA., pp: 37-46.

Cutting, D., J. Carger, J. Pedersen and J. Tukey, 1992. Scatter/Gather: A Cluster Based Approach to Browsing Large Document Collection. In Proceedings of the 15th ACM SIGIR Conference, Copenhagen, Denmark, pp: 318-329.

Dhillon, I., J. Fan and Y. Guan, 2001. Efficient Clustering of Very Large Document Collections. In: Grossman, R.L., C. Kamath, P. Kegelmeyer, V. Kumar and R.R. Namburu (Eds.). Data Mining for Scientific and Engineering Applications, Kluwer Academic Publishers.

Duda, R. and P. Hart, 1973. Pattern Classification and Scene Analysis. John Wiley and Sons, New York, NY.

Ester, M., H.P. Kriegel, J. Sander and X. Xu, 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, pp: 226-231.

Ester, M., A. Frommelt, H.P. Kreigel and J. Sander, 2000. Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support. Data Mining and Knowledge Discovery, Kluwer Academic Publishers, 4: 193-216.

Fayyad, U.M., G. Piatetsky-Shapiro and P. Smyth, 1996. From Data Mining to Knowledge Discovery: An Overview. In: Fayyad, U.M., G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT., pp: 1-34.

Forgey, E., 1965. Cluster Analysis of multivariate data: Efficiency versus interpretability of classification. Biometrics, 21: 768-780.

Foss, A., W. Wang and O. Zaane, 2001. A Non-Parametric Approach to Web Log Analysis. 1st SIAM ICDM, Workshop on Web Mining, Chicago, IL., pp: 41-50.

Hartigan, J., 1975. Clustering Algorithms. John Wiley and Sons, New York, NY.

Hartigan, J. and M. Wong, 1979. Algorithm AS136: A K-means clustering algorithm. Applied Statistics, 28: 100-108.

Heer, I. and E. Chi, 2001. Identification of Web user traffic composition using multi-modal clustering and information scent. 1st SIAM ICDM, Workshop on Web Mining, Chicago, IL., pp: 51-58.

Jain, A., K. Jain, M.N. Murty and P.J. Flynn, 1999. Data clustering: A review. ACM Computing Surveys, 31, 3: 264-323.

Kohonen, T., 1990. The self-organizing map. Proceedings of IEEE, 78: 1464-1480.

Sandar, J., M. Ester, H.P. Kriegel and X. Xu, 1998. Density based clustering in spatial databases: The algorithm gdbscan and its applications. In Data Mining and Knowledge Discovery, 2: 169-194.

Steinbach, M., G. Karypis and V. Kumar, 2000. A Comparison of Document Clustering Techniques. 6th ACM SIGKDD, World Text Mining Conference, Boston, MA.

Xu, X., M. Ester, H.P. Kriegel and J. Sander, 1998. A distribution-based clustering algorithm for mining in large spatial databases. In: Proceedings of the 14th ICDE, Orlando, FL., pp: 324-331.