Journal of Computer Science 8 (5): 656-664, 2012
ISSN 1549-3636

A Density Based Dynamic Data Clustering Algorithm based on Incremental Dataset

[1]Angel Latha Mary, S., [2]K.R. Shankar Kumar
[1]Department of Information Technology,
Information Institute of Engineering, Coimbatore, Tamilnadu, India
[2]Department of Electronics and Communication Engineering
Sri Ramakrishna Engineering College, Coimbatore, Tamilnadu, India

Abstract: **Problem statement:** Clustering and visualizing high-dimensional dynamic data is a challenging problem. Most of the existing clustering algorithms are based on the static statistical relationship among data. Dynamic clustering is a mechanism to adopt and discover clusters in real time environments. There are many applications such as incremental data mining in data warehousing applications, sensor network, which relies on dynamic data clustering algorithms. **Approach:** In this work, we present a density based dynamic data clustering algorithm for clustering incremental dataset and compare its performance with full run of normal DBSCAN, Chameleon on the dynamic dataset. Most of the clustering algorithms perform well and will give ideal performance with good accuracy measured with clustering accuracy, which is calculated using the original class labels and the calculated class labels. However, if we measure the performance with a cluster validation metric, then it will give another kind of result. **Results:** This study addresses the problems of clustering a dynamic dataset in which the data set is increasing in size over time by adding more and more data. So to evaluate the performance of the algorithms, we used Generalized Dunn Index (GDI), Davies-Bouldin index (DB) as the cluster validation metric and as well as time taken for clustering. **Conclusion:** In this study, we have successfully implemented and evaluated the proposed density based dynamic clustering algorithm. The performance of the algorithm was compared with Chameleon and DBSCAN clustering algorithms. The proposed algorithm performed significantly well in terms of clustering accuracy as well as speed.

Key words: Clustering, cluster validation, cluster validation metrics

INTRODUCTION

Data mining is the process of extracting potentially useful information from a data set. Clustering is a popular data mining technique which is intended to help the user discover and understand the structure or grouping of the data in the set according to a certain similarity measure. Clustering is a division of data into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification. It models data by its clusters. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics and numerical analysis. The search for clusters is unsupervised learning and the resulting system represents a data concept. From a practical perspective clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, Web analysis, CRM, marketing, medical diagnostics, computational biology and many others (Berkhin, 1988). The existing clustering algorithm integrates static components. Most of the applications are converted into real time application. It enforced that object to be clustered during the process based on its property. Dynamic clustering is a mechanism to adopt the clustering in real time environments such as mobile computing, war-end movement observation (Crespoa and Weber, 2005). Dynamic data mining is increasingly attracting attention from the respective research community. On the other hand, users of installed data mining systems are also interested in the related techniques and will be even more, since most of these installations will need to be updated in the future for each data mining technique used. We need different methodologies for dynamic data mining. In this study,

Corresponding Author: Angel Latha Mary, S., Department of Information Technology, Information Institute of Engineering, Coimbatore, Tamil Nadu, India Tel: +919842242882/+91-4222645629

we present a methodology for Density Based Dynamic Data Clustering Algorithm based on Incremental DBSCAN.

Clustering of dynamic data: Clustering is a field of active research in data mining. Most of the work has focused on static data sets (Han and Kamber, 2011). Traditional clustering algorithms used in data mining will not perform well on dynamic data sets. A clustering algorithm must consider the elements' history in order to efficiently and effectively find clusters in dynamic data. There has been little work on clustering of dynamic data. We define a dynamic data set as a set of elements whose parameters change over time. A flock of flying birds is an example of a dynamic data set. We are interested in exploring algorithms are capable of finding relationships amongst the elements in a dynamic data set. In this study we evaluate the use of data clustering techniques developed for static data sets on dynamic data.

Recent developments of dynamic data mining: Within the area of data mining various methods have been developed in order to find useful information in a set of data. Among the most important ones are decision trees, neural networks, association rules and clustering methods (Crespoa and Weber, 2005; Loganantharaj *et al*., 2000).

For each of the above-mentioned data mining methods, updating has different aspects and some updating approaches have been proposed, as we will see next.

Decision trees: Various techniques for incremental learning and tree restructuring as well as the identification of concept drift have been proposed in the literature.

Neural networks: Updating is often used in the sense of re-learning or improving the net's performance by learning with new examples presented to the network

Association rules: Raghavan *et al*. have developed systems for dynamic data mining for association rules.

Clustering: Below, we describe in more detail approaches for dynamic data mining using clustering techniques that can be found in literature.

Recent developments of clustering systems using dynamic elements are concerned about modeling the clustering process dynamically, i.e. adaptations of the algorithm are performed while applying it to a static set of data.

MATERIALS AND METHODS

The cluster validation methods:

Major difficulties in cluster validation: The presence of large variability in cluster geometric shapes and the number of clusters cannot always be known a priori are the main reason for validating the quality of the identified clusters. Different distance measures also lead to different types of clusters so that deciding the 'best' cluster is based on several aspects with respect to the application. So that the results of a cluster validation algorithm not always give best result from the application's point of view (Bezdek and Pal, 1998).

Cluster validity: In fact, if cluster analysis is to make a significant contribution to engineering applications, much more attention must be paid to cluster validity issues that are concerned with determining the optimal number of clusters and checking the quality of clustering results. Many different indices of cluster validity have been proposed, such as the Bezdek's partition coefficient, the Dunn's separation index, the Xie-Beni's separation index, Davies-Bouldin's index and the Gath-Geva's index. Most of these validity indices usually assume tacitly that data points having constant density to the clusters. However, it is not sure of the real problems (Bezdek and Pal, 1998).

Indices of cluster validity: Cluster validation refers to procedures that evaluate the clustering results in a quantitative and objective function. Some kinds of validity indices are usually adopted to measure the adequacy of a structure recovered through cluster analysis. Determining the correct number of clusters in a data set has been, by far, the most common application of cluster validity. In general, indices of cluster validity fall into one of three categories. Some validity indices measure partition validity by evaluating the properties of the crisp structure imposed on the data by the clustering algorithm. In the case of fuzzy clustering algorithms, some validity indices such as partition coefficient and classification entropy use only the information of fuzzy membership grades to evaluate clustering results. The third category consists of validity indices that make use of not only the fuzzy membership grades but also the structure of the data.

The cluster validity measures:

Dunn's index vD: This index is used to identify the compact and well-separated clusters C Eq. 1:

$$vD = \min_{i \in c}\left\{\min_{j \in c, i \neq j}\left\{\frac{\delta(C_i, C_j)}{\max_{k \in c}\{\Delta(C_k)\}}\right\}\right\} \quad (1)$$

Where:

$$\delta(C_i, C_j) = \min\{d(x_i, x_j) \mid x_i \in C_i, x_j \in C_j\}$$
$$\Delta(C_k) = \max\{d(x_i, x_j) \mid x_i, x_j \in C_i\}$$

δ is a distance function and C_I , C_j C_k are the sets whose elements are the data points assigned to the corresponding i^{th}, j^{th} and k^{th} clusters respectively. The main drawback with direct implementation of Dunn's index is computational since calculating becomes computationally very expensive as the number of clusters and the total point's increase. Larger values of vD correspond to good clusters and the number of clusters that maximizes vD is taken as the optimal number of clusters.

Generalized Dunn Index vGD Eq. 2:

$$vGD = \min_{s \in c}\left\{\min_{t \in c, s \neq t}\left\{\frac{\delta_i(C_s, C_t)}{\max_{k \in c}\{\Delta_j(C_k)\}}\right\}\right\} \quad (2)$$

Five set distance functions and three diameter functions are defined in of these, we have used two combinations δ_3 and δ_3 (which is recommended in (Karypis *et al*., 1999) as being most useful for cluster validation) in one and combinations δ_5 and δ_3 in the other. The three measures viz., combinations δ_3, δ_3 and δ_5 and are defined as follows:

$$\Delta_3(S) = 2\left(\frac{\sum_{x \in S} d(x, zS)}{|S|}\right)$$

$$\delta_3(S,T) = \frac{1}{|S||T|}\sum_{x \in S, y \in T} d(x,y) \quad \text{and}$$

$$\delta_5(S,T) = \frac{1}{|S|+|T|}\left(\sum_{x \in S} d(x, zT) + \sum_{y \in T} d(y, zS)\right)$$

here $zS = (1/|S|)\sum_{x \in S} x$ and $zT = (1/|T|)\sum_{x \in T} y$

Larger values of vGD correspond to good clusters and the number of clusters that maximizes vGD is taken as the optimal number of clusters. In this evaluation, we used $\delta_3$and δ_3 as diameter functions during evaluating the algorithms under consideration.

Davies-bouldin index []: This index (Davies and Bouldin, 1979) is a function of the ratio of the sum of within-cluster scatter to between-cluster separation Eq. 3:

$$DBI = \frac{1}{n}\sum_{i=1}^{n}\max_{i \neq j}\left(\frac{S_n(Q_i) + S_n(Qj)}{S(Q_i, Q_i)}\right) \quad (3)$$

where n- number of clusters, S_n - average distance of all objects from the cluster to their cluster centre, - $S(Q_,Q_j)$ distance between clusters centres. Hence the ratio is small if the clusters are compact and far from each other. Consequently, Davies-Bouldin index will have a small value for a good clustering (Bezdek and Pal, 1998).

The algorithms under evaluation:

Chameleon: Chameleon is a new agglomerative hierarchical clustering algorithm that overcomes the limitations of existing agglomerative hierarchical clustering algorithms. A major limitation of existing agglomerative hierarchical schemes such as the Group Averaging Method [JD88], ROCK [GRS99] and CURE [GRS98] is that the merging decisions are based upon static modeling of the clusters to be merged. These schemes fail to take into account special characteristics of individual clusters and thus can make incorrect merging decisions when the underlying data does not follow the assumed model, or when noise is present. There are two major limitations of the agglomerative mechanisms used in existing schemes. First, these schemes do not make use of information about the nature of individual clusters being merged. Second, one set of schemes (CURE and related schemes) ignore the information about the aggregate interconnectivity of items in two clusters, whereas the other set of schemes (ROCK, the group averaging method and related schemes) ignore information about the closeness of two clusters as defined by the similarity of the closest items across two clusters (Karypis *et al*., 1999; Bezdek and Pal, 1998).

Its key feature is that it accounts for both interconnectivity and closeness in identifying the most similar pair of clusters. Chameleon uses a novel approach to model the degree of interconnectivity and closeness between each pair of clusters. This approach considers the internal characteristics of the clusters themselves. Thus, it does not depend on a static, user-supplied model and can automatically adapt to the internal characteristics of the merged clusters. Chameleon operates on a sparse graph in which nodes represent data items and weighted edges represent similarities among the data items. This sparse-graph representation allows Chameleon to scale to large data sets and to successfully use data sets that are available only in similarity space and not in metric spaces. Data sets in a metric space have a fixed number of attributes for each data item, whereas data sets in a similarity space only provide similarities between data items.

Chameleon finds the clusters in the data set by using a two-phase algorithm. During the first phase, Chameleon uses a graph-partitioning algorithm to cluster the data items into several relatively small subs to find the genuine clusters by repeatedly combining these sub-clusters. During the second phase, it uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining together these sub-clusters (Crespoa and Weber, 2005; Goura *et al*., 2011; Goyal *et al*., 2011).

DBSCAN: DBSCAN (Density Based Spatial Clustering of Applications with Noise) and DENCLUE ((DENsity-based CLUstEring) will be implemented to represent density based partitioning algorithms. DBSCAN creates clusters from highly connected elements while DENCLUE clusters elements in highly populated areas. Both algorithm handle outliers well and will not include them in any cluster.

The proposed density based dynamic DBSCAN: We modeled the proposed Density based Dynamic DBSCAN algorithm using the ideas mentioned in the earlier work (Ester *et al*., 1998; 1996; Ester and Wittmann, 1998; Su *et al*., 2009; Sarmah and Bhattacharyya, 2010; Chakraborty *et al*., 2011; Chakraborty and Nagwani, 2011). Our implementation is slightly different from the standard approach, in our algorithm, we only considered problems related with data insertion. Further, we dynamically changed the epsilon during each batch of insertion. Another most important variation is, in during each step of batch insertion, the data points which were classified as noise or border objects (outliers) were considered as unclassified points and combined with the new data which is to be inserted. These small changes made our algorithm to perform very good and formed good clusters with the dynamic incremental data set.

The density based dynamic clustering algorithm:

The main aspects of dynamic clustering process: When inserting an object p into the database D, it may be treated in one of the following ways:

Noise: If there is no nearby point in the epsilon neighborhood or the number of neighbors is not satisfying the density criteria, then, p is also a noise object and nothing else is changed.

Absorption of point p: If all the nearby points in the epsilon neighborhood belongs to some cluster, then the newly inserted point p also belong to the same class ID- in other words, the new point will simply be absorbed by that existing cluster.

Merging of clusters: If all the nearby points in the epsilon neighborhood are members of different clusters, then the newly inserted point p will connect all these existing clusters and form one cluster out of these several clusters.

Creation of a cluster: At the location of insertion, if there are some nose objects already present and if the point p can be treated as a core point after insertion by satisfying the condition of a cluster membership, then it will lead to form a new cluster in that region.

A dynamic DBSCAN algorithm for clustering evolving data over time: Let:

- D_{Ex} be the Existing dataset which is already cluster in to C_{ex} number of classes.
- D_{New} be the New dataset which is to be added in to D_{Ex} cluster in to C_{new} number of classes.
- ε_{Ex} is the previously estimated epsilon value of Existing dataset D_{Ex}

Algorithm: DYN_DBSCAN (D_{Ex}, D_{New},ε_{Ex}, MinPts)
// Precondition:
All objects in D_{Ex} are classified
All objects in D_{new} are unclassified.
ε_{Ex} The estimated Epsilon of D_{Ex}
//Separate N_{ex}, the set of noise object (outliers) (and border Objects) in D_{Ex} according to the previous stage of clustering)
N_{ex} = Outliers (D_{Ex})

//Assume the previous outliers (and border Objects) as unclassified
$D_{new} \leftarrow D_{new} \cup N_{ex}$

FORALL objects o in D_{New} DO {
//Add the object o in D_{Ex}
$D_{Ex} \leftarrow D_{Ex} \cup o$
Re-estimate ε_{new} based on the new D_{Ex}
//find the neighborhood of o based on $\square_{new}$
$N_{Eps}(o)$= Eps-neighborhood of o;
U = Unclassified ($N_{Eps}(o)$)
If ($N_{Eps}(o)$==MinPts) {
// no nearby points, so p is a Noise
type(o)= Border_Object;
class(o)= noise;
}elseif ($N_{Eps}(o)$>1 and $N_{Eps}(o)$ <= MinPts) {
type(o)= unclassified;
class(o)= unclassified;
if(All the object in U are unclassified) {
//Create a cluster of border and noise objects and Merge them with nearby clusters if possible

```
  seeds= NEps(o);
      Update (seeds, Border_Object);
} Elseif(U is empty) {
   // case of Absorption in non core points
  class(o)= TheClassOfTheNeighbors;
}elseif(Some of the object in U are unclassified){
//merge all points and assign a new Class ID
  current_cluster-id=NewID();
  class(o)= current_cluster-id;
  seeds= NEps(o);
  Update (seeds, current_cluster-id);
}
} elseif ( NEps(o)>=MinPts ) {
      type(o)= core;
if(All the object in U are unclassified) {
 // Merge clusters and assign a common class label
      current_cluster-id=NewID();
  class(o)= current_cluster-id;
  seeds= NEps(o);
  Update (seeds, current_cluster-id);
} Elseif(U is empty) {
   // case of Absorption in existing cluster
  class(o)= cluster-id of the Neighbor;
}elseif(Some of the object in U are unclassified){
 //merge all clusters and assign a common class ID
  current_cluster-id=NewID();
  class(o)= current_cluster-id;
  seeds= NEps(o);
  Update(seeds, current_cluster-id);
}
function Update (seeds, cluster-id){
 WHILE NOT seeds.empty() DO {
      currentObject := seeds.top();
      seeds.pop();
      NEps(currentObject)=                    Eps-neigh.of
CurrentObject
      IF | NEps(currentObject) ≥ MinPts {
   type(currentObject )=1;
      else |NEps(currentObject)>0
        type(currentObject )=0;
}
If |NEps(currentObject)>0
    FORALL objects obj in NEps(currentObject) DO {
     if class(objects) <> cluster-id {
             class(objects) = cluster-id
             seeds.push(obj);
   }
    }
  }
 }
```

RESULTS

The performances of the algorithms are evaluated using synthetic dataset and real data sets from UCI Data repository.

The performance in Terms of Generalized Dunn Index, Davies-Bouldin Index and clustering time with the synthetic dataset and real dataset , The proposed dynamic clustering algorithm was good and almost equal or little bit better than the normal DBSCAN algorithm.

DISCUSSION

Results with synthetic data set: To evaluate the performance of clustering in a very controlled manner, multi dimensional synthetic data sets of were used.

The following Fig. 1 shows the two dimensional plot of one of such dataset.

The parameters of the algorithm used to create the synthetic spheroid form of data points using Gaussian distribution:

Number of Classes	: 6
Records per Classes	: 100
Number of Dimensions	: 5
Standard Deviation	: 0.50
Total Records	: 600.00

The following Fig. 2 results are the performance of clustering with dataset of the above mentioned attributes. The line chart shows the performance of the algorithm with the increase of data size. The bar chart shows the average performance of the algorithms.

The following Fig. 3 shows the performance in Terms of Generalized Dunn Index with the synthetic dataset. The performance of the proposed dynamic clustering algorithm was good and almost equal or little bit better than the normal DBSCAN algorithm.

The following Fig. 4 shows the average performance in Terms of Generalized Dunn Index. The performance of the proposed dynamic clustering algorithm equal to the normal DBSCAN algorithm.

The following Fig. 5 shows the performance in Terms of Davies-Bouldin Index

The following Fig. 6 and 7 shows the average performance in Terms of Davies-Bouldin Index

Performance in terms of time: The following graph shows the performance in Terms of time. The speed of the proposed dynamic clustering algorithm was better than Chameleon as well as DBSCAN algorithm.

The results with UCI data sets: To validate the performance of the algorithms, we used some of the real data sets from UCI Data repository.

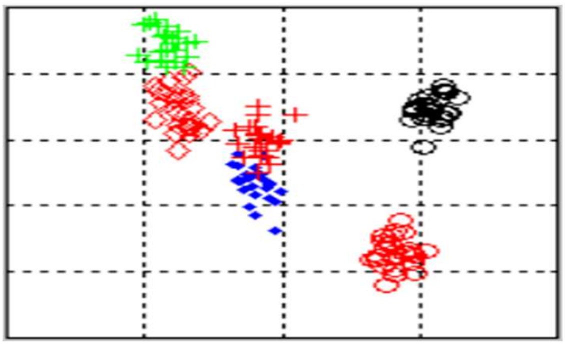

Fig. 1: Two dimensional plot of synthetic dataset

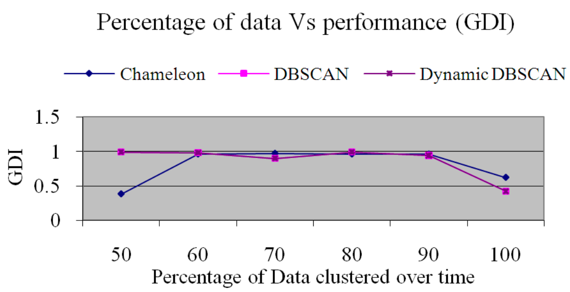

Fig. 2: Performance in terms of GDI (Syn.Data)

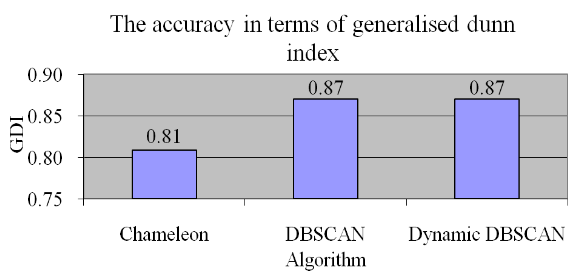

Fig. 3: The average performance in terms of GDI (Syn. Data)

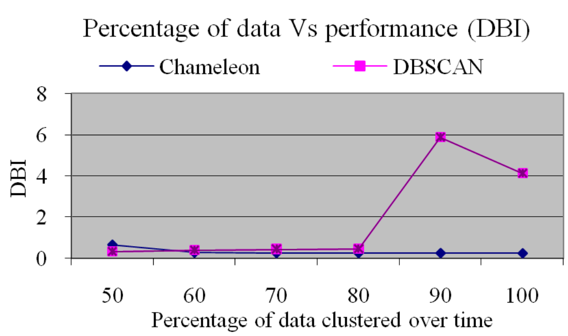

Fig. 4: Performance in terms of DBI (Syn.Data)

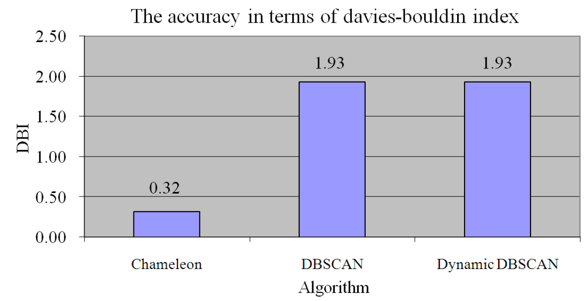

Fig. 5: Average performance in terms of DBI (Syn.Data)

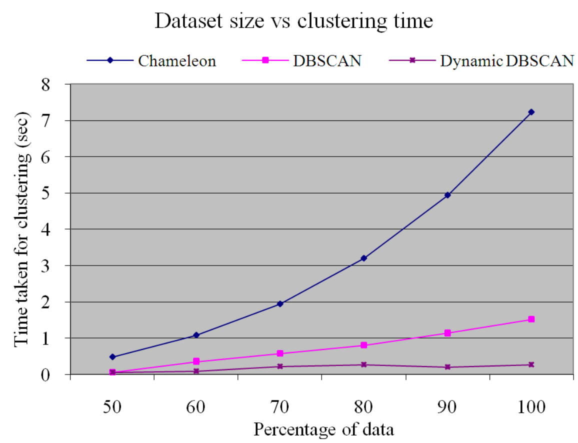

Fig. 6: Performance in terms of time (Syn.Data)

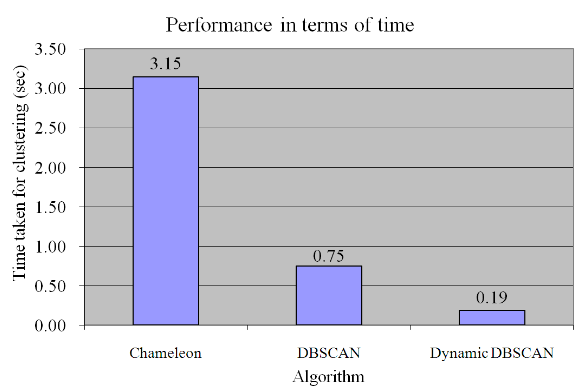

Fig. 7: Average performance in terms of time (Syn.Data)

We used the following datasets:

- Zoo Data
- Wine Data
- TIC2000 Data (The Insurance Company Data)
- Wisconsin Breast Cancer Dataset

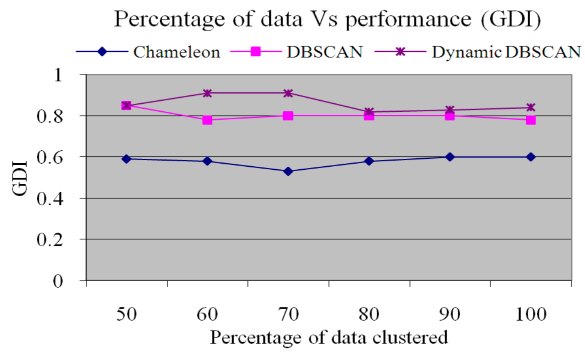

Fig. 8: Average performance in terms of GDI (Wine data)

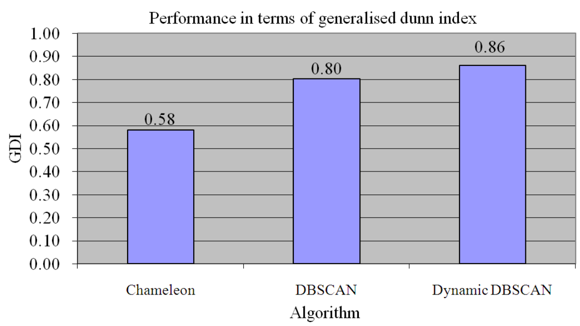

Fig. 9: Performance in terms of GDI (Wine data)

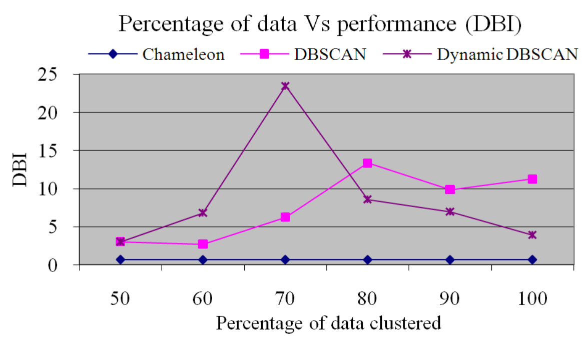

Fig. 10: Performance in terms of DBI (Wine data)

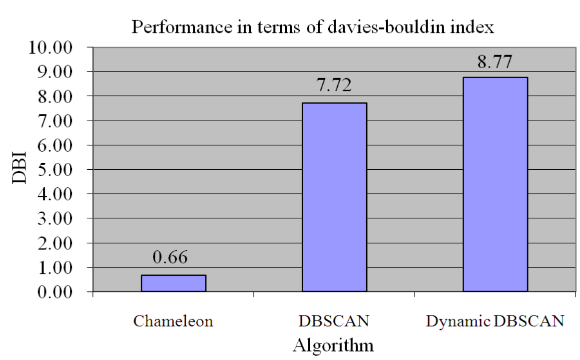

Fig. 11: Average performance in terms of DBI (Wine data)

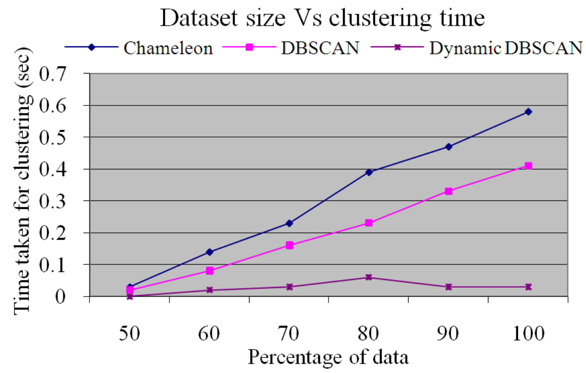

Fig. 12: Performance in terms of time (Wine data)

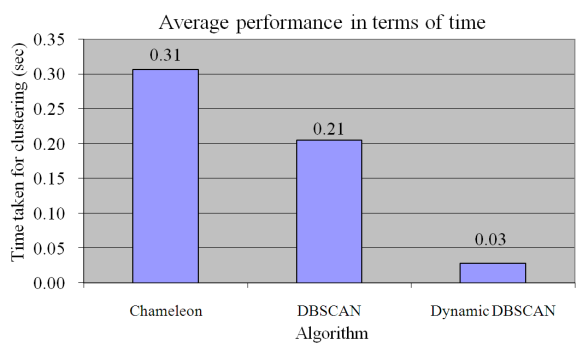

Fig. 13: Average performance in terms of time (Wine data)

The performance of the algorithms with "UCI Wine Data" with different size of incremental data.

The following Fig. 8-15 shows the performance in Terms of Generalized Dunn Index, Davies-Bouldin Index and clustering time. The performance of the proposed dynamic clustering algorithm was good. And in most cases, the accuracy in terms of validation metrics is little bit better than the normal DBSCAN algorithm and Chemeleon:

- Performance in terms of Generalized Dunn Index (Wine Data)
- The Average performance interms of Generalized Dunn Index (Wine Data)
- Performance in terms of Davies-Bouldin Index (Wine Data)
- The Average performance interms of Davies-Bouldin Index (Wine Data)
- Performance in terms of Time (Wine Data)
- Average Performance in terms of Time (Wine Data)

The performance with different UCI datasets: The following graph shows the Average performance of the algorithm with different UCI data sets.

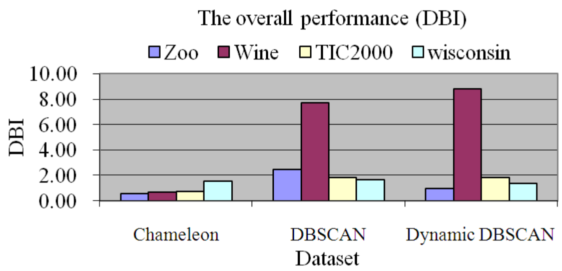

Fig. 14: Average performance in terms of DBI (4 UDI Data)

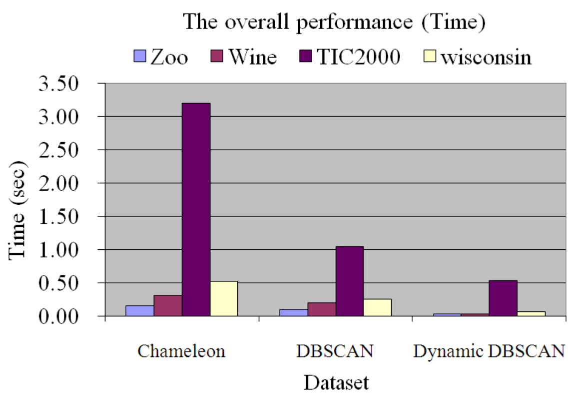

Fig. 15: Average performance in terms of time (4 UDI Data)

The performance was measured in terms of Generalized Dunn Index, Davies-Bouldin Index and clustering time. The performance of the proposed dynamic clustering algorithm was good. And in most cases, the accuracy in terms of validation metrics is little bit better than the normal DBSCAN algorithm and Chameleon.

The average performance of the algorithms in terms of Generalized Dunn Index with different size of incremental data was good and almost equal in with all the four evaluated datasets.

Average performance in terms of Davies-Bouldin index: The average performance of the algorithms in terms of, Davies-Bouldin Index is almost equal or little bit higher than the normal DBSCAN.

Average performance in terms of time: The average performance of the algorithms in terms of, clustering time is almost very minimum in the proposed dynamic clustering algorithm. The performance of the proposed algorithm was very good on all the data sets.

CONCLUSION

In this study, we have successfully implemented and evaluated the proposed density based dynamic clustering algorithm. The algorithm was able to insert data objects one by one and then re-estimate the cluster IDs during each and every point which was inserted. The algorithm is capable of create, modify and insert clusters over time. The performance of the algorithm was compared with Chameleon and DBSCAN clustering algorithms. As shown in the results of the previous section, the proposed algorithm performed significantly well in terms of clustering accuracy as well as speed.

There are possibilities to handle batch insertion by which we can reduce the run time of the algorithm. So the future work will address the ways to improve the performance of the algorithm in terms of speed and accuracy. This work only addressed the problem of clustering incremental data set in which only data is added over time.

The future work may address all the other possibilities of dynamic operations like deletions and modifications of data points and remodel the algorithm to cluster the data during this dynamically changing dataset. Even though, the performance of Chameleon was poor in terms of speed, it also posses the capabilities of becoming a dynamic clustering algorithm. Future works may explore these possibilities and address hybrid dynamic clustering algorithms.

ACKNOWLEDGEMENT

We thank to Management, Principal and Secretary of Info Institute of Engineering for providing facility to implement this study.

REFERENCES

Berkhin, P., 1988. Survey of Clustering Data Mining Techniques. Accrue Software, Inc., San Jose, CA.

Bezdek, J.C. and N.R. Pal, 1998. Some new indexes of cluster validity. IEEE Trans. Syst., Man, Cybern. B, 28: 301-315. DOI: 10.1109/3477.678624

Chakraborty, S. and N.K. Nagwani, 2011. Analysis and study of incremental DBSCAN clustering algorithm. Int. J. Enterprise Comput. Bus. Syst.

Chakraborty, S., N.K. Nagwani and L. Dey, 2011. performance comparison of incremental k-means and incremental DBSCAN algorithms. Int. J. Comput. Appli., 27: 14-18. DOI: 10.5120/3346-4611

Crespoa, F. and R. Weber, 2005. A methodology for dynamic data mining based on fuzzy clustering, Fuzzy Sets Syst., 150: 267-284. DOI: 1016/j.fss.2004.03.028

Davies, D.L. and D.W. Bouldin, 1979. A cluster separation measure. IEEE Trans. Patt. Anal. Machine Intell., 1: 224-227.

Ester M. and R. Wittmann, 1998. Incremental generalization for mining in a data warehousing environment. Adv. Database Technol. Lecture Notes Comput. Sci., 1377: 135-149. DOI: 10.1007/BFb0100982

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

Ester, M., H.P. Kriegel, J. Sander, M. Wimmer and X. Xu, 1998. Incremental clustering for mining in a data warehousing environment. Proceedings of the 24th VLDB Conference, Institute for Computer Science, University of Munich, Germany, New York, USA.

Goura, V.M.K.P., N.M. Rao and M.R. Reddy, 2011. A dynamic clustering technique using minimumspanning tree. Proceedings of the 2nd International Conference on Biotechnology and Food Science (IPCBEE' 11), IACSIT Press, Singapore, pp: 66-70.

Goyal, N., P. Goyal, K. Venkatramaiah, P.C. Deepak and P.S. Sanoop, 2011. An efficient density based incremental clustering algorithm in data warehousing environment. Proceedings of the International Conference on Computer Engineering and Applications, (IPCSIT' 11), IACSIT Press, Singapore, pp: 482-486.

Han, J. and M. Kamber, 2011. Data Mining: Concepts and Techniques. 3rd Edn., Elsevier, Burlington, ISBN-10: 9780123814791, pp: 744.

Karypis, G., E.H.S. Han and V. Kumar, 1999. Chameleon: Hierarchical clustering using dynamic modeling. IEEE Comput., 32: 68-75. DOI: 10.1109/2.781637

Loganantharaj, R., G. Palm and M. Ali, 2000. Intelligent Problem Solving: Methodologies and Approaches. 1st Edn., Springer, Berlin, ISBN: 3540676899, pp: 751.

Sarmah, S. and D.K. Bhattacharyya, 2010. An effective technique for clustering incremental gene expression data. IJCSI Int. J. Comput. Sci., 7: 26-41.

Su, X., Y. Lan, R. Wan and Y. Qin, 2009. A fast incremental clustering algorithm. Proceedings of the 2009 International Symposium on Information Processing, Aug. 21-23, Huangshan, P. R. China, pp: 175-178.