HOSTED BY
ELSEVIER

Alexandria University
Alexandria Engineering Journal
www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

Efficient incremental density-based algorithm for clustering large datasets

Ahmad M. Bakr *, Nagia M. Ghanem, Mohamed A. Ismail

Faculty of Engineering, Computer and Systems Engineering Department, Alexandria University, Alexandria, Egypt

Received 4 November 2014; accepted 20 August 2015
Available online 12 October 2015

KEYWORDS

Incremental clustering;
Density-based clustering;
Document clustering;
Information retrieval

Abstract In dynamic information environments such as the web, the amount of information is rapidly increasing. Thus, the need to organize such information in an efficient manner is more important than ever. With such dynamic nature, incremental clustering algorithms are always preferred compared to traditional static algorithms. In this paper, an enhanced version of the incremental DBSCAN algorithm is introduced for incrementally building and updating arbitrary shaped clusters in large datasets. The proposed algorithm enhances the incremental clustering process by limiting the search space to partitions rather than the whole dataset which results in significant improvements in the performance compared to relevant incremental clustering algorithms. Experimental results with datasets of different sizes and dimensions show that the proposed algorithm speeds up the incremental clustering process by factor up to 3.2 compared to existing incremental algorithms.

1. Introduction

Data clustering [1] is a discovery process that groups set of objects in disjoint clusters such that the intra-cluster similarity is maximized and inter-cluster similarity is minimized. The resulted clusters can explain characteristics of the underlying data distribution which can be used as a foundation for other data mining and analysis techniques. Data clustering is used in a wide range of applications. For example, in marketing, clustering is used to find group of customers with similar behaviors [2]. In biology, it is used to find similar plants or animals given their features [3]. In search engines, clustering is used to group similar documents to facilitate user search and topics discovery [4], while in networks, clustering is used in analyzing and classifications of network traffic [5]. Given the number of clustering applications, there are challenges associated with the clustering process. Such challenges include the choice of a suitable clustering algorithm, the choice of best representative features of objects and the choice of appropriate distance/similarity measure [6]. Other challenges include dealing with outliers [7], ability to interpret clustering results (selection of cluster's representative and cluster summarization) [8] and dealing with huge number of dimensions and distributed data [9].

Due to data overwhelming, new challenges have appeared for traditional algorithms that were implemented before. One challenge is the ability to perform incremental update of the

* Corresponding author. Tel.: +20 1004102280.
E-mail addresses: ahmad.bakr@alexu.edu.eg (A.M. Bakr), nagia.ghanem@alexu.edu.eg (N.M. Ghanem), maismail@alexu.edu.eg (M.A. Ismail).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

http://dx.doi.org/10.1016/j.aej.2015.08.009

clusters upon any change in the dataset. Traditional algorithms require existence of the whole dataset before running the algorithm. However, in many online applications where the time factor is essential, traditional algorithms are not feasible. As a result, algorithms that can perform incremental updates to the clusters are always preferred in dynamic environments. In such algorithms, objects are processed one at a time and incrementally assigned to their prospective clusters while they progress with minimized overload. Along with incremental process, incremental clustering algorithms face other challenges such as finding the best way to determine the most suitable cluster that next object will be assigned, and also once an object is assigned to a cluster, this assignment may change in the future as new objects are added or removed from the dataset (which is also known as The Insertion Order Problem [10]).

In this paper, an incremental density-based clustering algorithm is introduced for incrementally building and updating clusters in the dataset. The proposed algorithm enhances the clustering process by incrementally partitioning the dataset to reduce the search space of the neighborhood to one partition rather than the whole dataset. For each partition, the proposed algorithm maintains an incremental DBSCAN algorithm to detect and update dense regions at the partition with new objects. Finally, to identify the final natural number of final clusters, the algorithm employs a merging step to merge dense regions in different partitions. Experimental results show that the proposed algorithm speeds up the incremental clustering process with a factor up to 3.2 compared to relevant existing incremental clustering algorithms. The main contribution of this work can be summarized in enhancing the incremental DBSCAN algorithm by limiting the search space to partitions instead of the whole dataset which speeds up the clustering process. The proposed algorithm is proved to perform better than the existing incremental DBSCAN especially in large datasets with higher number of dimensions.

The rest of the paper is organized as follow: Section 2 discusses related work; Section 3 presents the proposed algorithm in detail; Section 4 presents the experimental results with other algorithms as well as the evaluation metrics and the datasets; finally Section 5 concludes a summary of the proposed work with potential future work.

2. Related work

Clustering is unsupervised classification of data into groups or clusters. Existing clustering algorithms can be classified into different classes. Centroid-based algorithms such as K-means [11], PAM [12], BIRCH [13], and CLARANS [14] are simple and fast to converge to local optimum; however they have limitations of predefining the number of clusters and dealing with clusters of different sizes and shapes. Hierarchical-based algorithms include single linkage, CURE [15] and Chameleon [16] can deal with different shapes and sizes of clusters and less susceptible to initialization and outliers; however they suffer from two main limitations: firstly they can never undo what were done (e.g. whenever two clusters are merged, they will always be merged) and secondly their high complexity which makes them slow to converge. Density-based algorithms such as DBSCAN [17], SSN [18] and OPTICS [19] overcome the difficulty of detecting arbitrary shaped clusters by extracting high dense regions (clusters) separated by low dense regions. Density-based algorithms can detect noise objects as they are not reachable from any of the generated clusters and hence noise has less effect of density based algorithms.

In dynamic environments where data are changing frequently overtime, clustering algorithms are required to perform the necessary updates incrementally in an efficient manner. Many incremental clustering algorithms were developed to cope with the dynamic nature of the data. Tzortzis and Likas [20] proposed a kernel K-means as an extension of the standard K-means that incrementally identifies nonlinearly separable clusters based on kernel based clustering. Widyantoro and Ioerger [21] proposed an incremental hierarchical clustering algorithm that works in a bottom-up fashion such that a new instance is placed in the hierarchy and a sequence of hierarchy restructuring is performed only in regions that have been affected by the presence of the new instance. Mary and Kumar [22] proposed an incremental density-based algorithm that can detect clusters with arbitrary shapes and handle updates in an incremental manner. Hammouda and Kamel [23] proposed an incremental similarity-histogram based algorithm where the similarity histogram is a concise statistical representation of the set of pair-wise document similarities distribution in the cluster and objects are added to a cluster only if they enhance its similarity histogram (keeping the distribution of similarities skewed to the right of the histogram). As the incremental DBSCAN algorithm [24] is used at one of the stages of the proposed algorithm, more details are introduced in the next subsection.

2.1. Incremental DBSCAN algorithm

Incremental DBSCAN algorithm is density-based clustering algorithm that can detect arbitrary shaped clusters. While static DBSCAN [17] is applied to static datasets in which the existence of all objects is required before running the algorithm, incremental DBSCAN [24] works by processing objects as they come and update/create clusters as needed. Given two parameters Eps and Minpts:

Definition 1. The Eps neighborhood of an object p is donated by $N_{\text{Eps}}(p)$, is defined by $N_{\text{Eps}}(p) = \{q | \text{dis}(p,q) \leqslant \text{Eps}\}$.

Definition 2. An object p is directly density-reachable from object q if $p \in N_{\text{Eps}}(q)$ and $|N_{\text{Eps}}(q)| \geqslant \text{Minpts}$ (i.e. q is a core object).

Definition 3. An object p is density-reachable from an object q if there is a chain of objects $p_1, p_2, \ldots, p_n$ such that p_{i+1} is directly density reachable from p_i and $p_1 = p$ and $p_n = q$.

Due to the density-based nature of the incremental DBSCAN algorithm, the insertion or deletion of an object affects only the objects within a certain neighborhood (Fig. 1). Affected objects are potentially the objects that may change their cluster membership after insertion/deletion of an object p and they are defined as the objects in the $N_{\text{Eps}}(p)$ plus all other objects that are density reachable from objects in $N_{\text{Eps}}(p)$. The cluster memberships of all other objects are not affected.

Assuming that D is the objects space, following the insertion of an object p, defines:

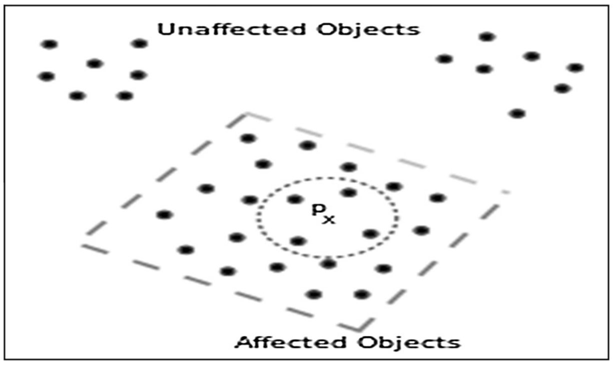

Figure 1 Affected objects due to insertion/deletion of object P.

$\text{UpdSeed}_{\text{ins}} = \{q | q$ is a core object, $\exists q':q'$ is a core object in $D \cup \{p\}$ but not in D and $q \in N_{\text{Eps}}(q')\}$. For the $\text{UpSeed}_{\text{Ins}}$, there are four cases to consider.

- If $\text{UpSeed}_{\text{Ins}}$ is empty, then p is considered as a noise object.
- If $\text{UpSeed}_{\text{Ins}}$ contains core objects which did not belong to a cluster before the insertion of p, then p creates a new cluster with these objects.
- If $\text{UpSeed}_{\text{Ins}}$ contains core objects that belong to exactly one cluster, then p joins this cluster.
- If $\text{UpSeed}_{\text{Ins}}$ contains core objects that are members of several clusters, then p merges all these clusters into one cluster.

Similarity, after deletion of some object p, defines: $\text{Upseed}_{\text{Del}} = \{q | q$ is a core object in $D \backslash \{p\}$, $\exists q':q'$ is a core object in D but not in $D \backslash \{p\}$ and $q \in N_{\text{Eps}}(q')\}$. For the $\text{UpSeed}_{\text{del}}$, there are three cases to consider.

- If $\text{Upseed}_{\text{Del}}$ is empty, then p is just removed.
- If $\text{Upseed}_{\text{Del}}$ contains objects that are density-reachable from each other, then deletion of p causes some of objects in $N_{\text{Eps}}(p)$ to be noise (i.e. lose their cluster membership).
- If the objects in $\text{Upseed}_{\text{Del}}$ are not directly density-reachable from each other, then these objects belong to one cluster c before the deletion of p, so a check should be performed whether these objects are density connected by other objects in the former cluster c. Depending on such density connections, the cluster c may be split or not.

3. Proposed algorithm

Fig. 2 shows the different stages of the proposed algorithm. Initially, the algorithm requires defining of K partitions before operating. The algorithm considers the first K objects to be centroids of the K partitions are long as the distances between them are larger than Eps. For the subsequent objects, the algorithm incrementally partitions the dataset to eliminate the scalability problem of scanning the entire search space for finding the neighborhood of the new inserted object. Following assigning the new object to its nearest partition, the algorithm creates/updates dense regions in this partition according to the incremental DBSCAN algorithm [24]. After that, the algorithm incrementally merges the dense regions of different partitions based on a given connectivity measure to create/update the possible final clusters. Finally, the algorithm labels outliers and removes noise to produce the final clusters. Experimental results show that the proposed algorithm speeds up the clustering process with a factor up to 3.2 compared to relevant incremental clustering algorithms.

Algorithm 1 introduces the details of the proposed algorithm for incrementally creating and updating clusters.

```
Algorithm 1: Incremental creating/updating clusters
Input: Set of new objects $P$, centroids
Output: create/update clusters after insertions of $P$
1: M $\leftarrow$ List of object that may change their centroids
2: $D \leftarrow$ List of updated dense regions
3: **For** each $p_i$ in $P$ **do**
4:    $c \leftarrow$ nearest_centroid ($p_i$)
5:    M $\leftarrow$ update_centroids ($p_i$, $c$)
6:    Add $p_i$ to $M$
7: **end for**
8: **For** each $r_i$ in $M$ **do**
9:    $c_n \leftarrow r_i \cdot$ new_centroid
10:   $c_o \leftarrow r_i \cdot$ old_centroid
11:   Apply incDbscanDel to remove $r_i$ from $c_o$
12:   Apply incDbscanAdd to insert $r_i$ to $c_n$
13:   Add updated dense regions to $D$ (step 11 and 12)
14: **end for**
15: **For** each $d_i$ in $D$ **do**
16:   **For** each $d_j$ in $D$ and $i \neq j$ **do**
17:      **If** inter_connectivty ($d_i, d_j$) > $\alpha$ merge
18:         merge($d_i, d_j$)
19:      **end if**
20:   **end for**
21: **end for**
```

3.1. Incremental partitioning of dataset

Incremental partitioning for objects is needed to reduce the search space for object's neighborhood. Given a new object p, the neighborhood is determined by scanning objects of the nearest partition rather than the entire dataset (search space). The partitioning algorithm divides the dataset into a predefined number of partitions and assigns the object p to the partition with the nearest centroid (line 4). Although several incremental partitioning algorithms can be used at this stage such as [25] and other partitioning algorithms, the proposed algorithm uses the partition algorithm in [26] for the following reasons. First, due to the dynamic nature of the dataset, old objects may change their positions after the insertion of new objects; thus, the partitioning algorithm should be able to detect the existing objects that became closer to other centroids after insertion of new objects (line 5). Second, it provides what is called "stability"; the partitioning algorithm reaches stability by utilizing a learning rate that decays with time so that objects are consistently assigned to centroids and the centroids are changing with a very small tolerance. The result of the partitioning stage is finding the nearest centroid for the new object p in addition to updating positions of current centroids with respect to the employed learning rate.

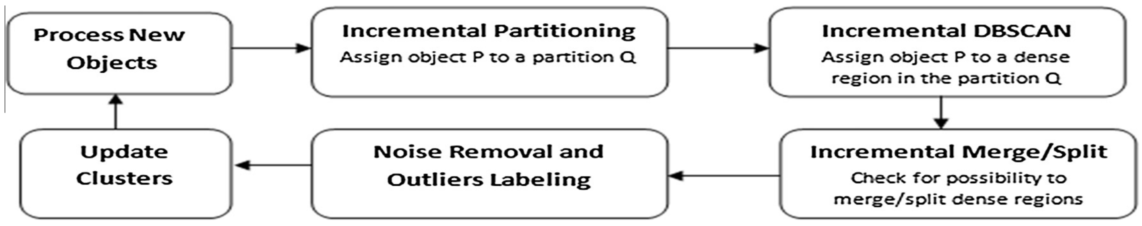
Process New Objects
Incremental Partitioning
Assign object P to a partition Q
Incremental DBSCAN
Assign object P to a dense region in the partition Q
Update Clusters
Noise Removal and Outliers Labeling
Incremental Merge/Split
Check for possibility to merge/split dense regions

Figure 2 Algorithm's stages.

3.2. Incremental DBSCAN of partitions

Incremental DBSCAN algorithm [24] is used to update dense regions in partitions. A dense region can be defined as a set of objects that can form a final cluster or be part of a final cluster. Given the new object *p*, the insertion module of DBSCAN (incDbscanAdd) is used to find a dense region at the nearest partition that the object *p* can join. For the old objects which changed their partitions, the deletion module (incDbscanDel) is used to remove them from their old dense regions and insertion module (incDbscanAdd) adds them to their new dense regions (lines 8–14). The goal of this stage is updating dense regions in different partition with the changes occurred in the partitioning stage.

3.3. Incremental merging of dense regions

To form the final set of clusters, an additional stage is needed to merge dense regions of different partitions. The number of final clusters is less than or equal the number of dense regions of the dataset. Given two regions A and B in two different partitions, A is merged to B if the inter-connectivity IE(A, B) is greater than a predefined threshold α merge (lines 17–19). Fig. 3 explains the idea of the inter-connectivity between two dense regions. Assume there is an imaginary edge between two borders objects in two different dense regions if the distance between these objects is less than the Eps defined by the incremental DBSCAN algorithm, then the inter-connectivity between dense regions A and B is defined as IE (A, B) where

$$\mathrm{IE(A,B)} = \frac{N_{ab}}{(N_a + N_b)/2} \tag{1}$$

where N_a and N_b are the number of edges that connect boarder objects in regions A and B respectively and N_{ab} is the number of edges that connect boarder objects from dense region A and to objects in dense region B.

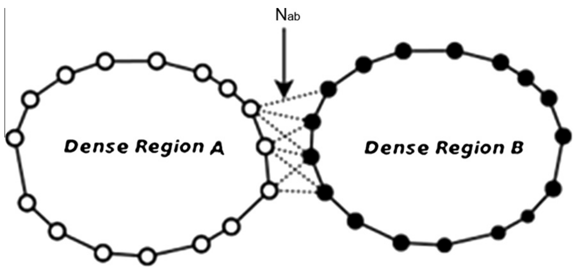

Figure 3 Interconnectivity between two regions.

3.4. Noise removal and outlier labeling

A final step is needed to eliminate noise from the dataset. The proposed algorithm is able to detect and eliminate the noise where noise objects are neither classified objects (not assigned to a cluster) nor reachable from any core objects in the clusters. However, objects whose distance to any cluster is less than Eps are considered as outliers and they are labeled by the label of the nearest cluster.

Fig. 4 shows an illustrative example for applying the proposed algorithm on 2D dataset. Fig. 4.1 shows the dataset before applying the proposed algorithm. After incrementally partitioning objects, centroids started to take final positions as shown in Fig. 4.2. While every partition maintains an incremental DBSCAN, dense regions started to appear more obviously in Fig. 4.3 and finally, merging dense regions and noise removal are applied as shown in Fig. 4.4.

4. Evaluation

The proposed algorithm is evaluated against two incremental clustering algorithms: incremental version of DBSCAN [24] and incremental similarity-histogram based clustering algorithm (specific for documents clustering) [23]. The experiments are performed on (Core i5, 6 GB RAM) machine and baseline algorithms are implemented by authors of this paper for accurate comparisons. Six datasets with different number of objects and dimensions are used to evaluate the proposed algorithm as shown in Table 1. The first three datasets are used to evaluate the proposed algorithm with the incremental DBSCAN algorithm [24] while the last three datasets (textual datasets) are used to evaluate the proposed algorithm with incremental similarity-histogram based clustering algorithm [23] which is designed originally for document clustering.

4.1. Evaluation metrics

F-Measure is used to evaluate the accuracy of the final clusters based on precision and recall while the speedup factor is used to compare the running times of the algorithms.

F-Measure is commonly used in evaluating the effectiveness of clustering and classification algorithms. It combines the precision and recall ideas in information retrieval. The F-Measure for class *i* and cluster *j* is given by the following formula:

$$F(i,j) = \frac{2\mathrm{Prec}(i,j) * \mathrm{Recall}(i,j)}{\mathrm{Prec}(i,j) + \mathrm{Recall}(i,j)} \tag{2}$$

And the overall F-Measure for cluster algorithm quality is given by the following formula where N is the total number of the data points and c_i is a candidate cluster.

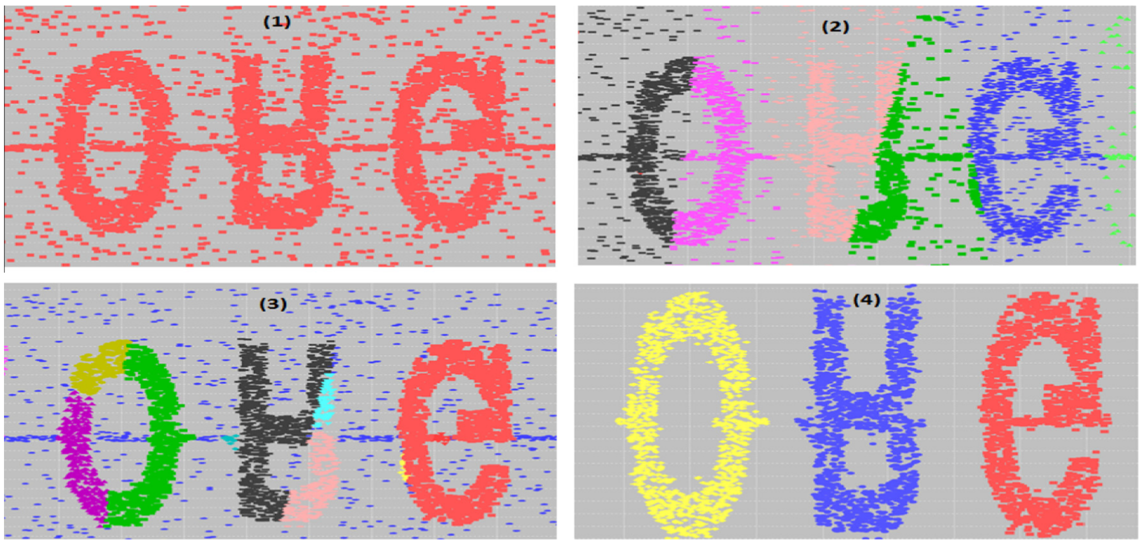

Figure 4 An illustrative example.

Table 1 Datasets.

Dataset	Classes	Num of objects	Type
Open digits	10	5620	Num, 64 dimensions
Pen digits	10	11,000	Num, 15 dimensions
SCC	10	600	Num, 60 dimensions
UW-CAN	10	314	Textual
4 Universities	7	8282	Textual
News Group	20	20,000	Textual

$$F = \sum_{i}^{l} \frac{|c_i|}{N} * \max\{F(i,j)\} \tag{3}$$

While the speedup factor is defined as the ratio of the baseline algorithm to the run time of the proposed algorithm.

$$\text{Speedup} = \frac{T_{\text{compared}}}{T_{\text{proposed}}} \tag{4}$$

4.2. Comparison with the incremental DBSCAN algorithm

In order to compare the proposed algorithm with the incremental DBSCAN [24], same values of Eps and MinPts are used in the experiment to show the effect of the enhancement. However, for the proposed algorithm, there are two parameters that can affect its performance:

1. The number of partitions (in the incremental partitioning stage).
2. The value of α merge (in the incremental dense regions merging stage)

Fig. 5 shows that the speedup factor (the ratio between execution time of the incremental DBSCAN [24] to the proposed algorithm) increases with the increase of the number of partitions as the search space is reduced. However, as the number of partitions increases, the overhead of partitioning and merging also increases after which this overhead leads to decreasing in the speedup of the proposed algorithm.

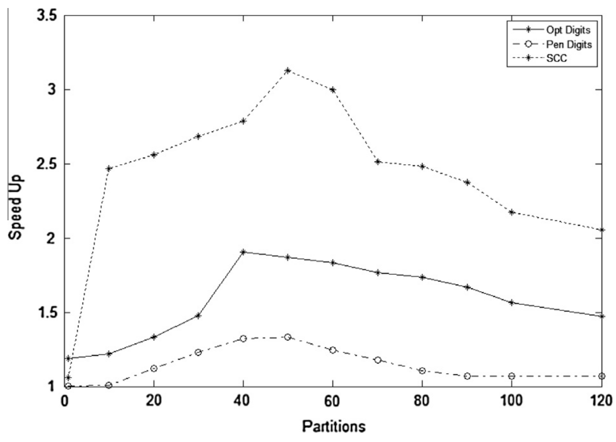

Figure 5 Speedup factor for the numerical datasets for different partitions.

Fig. 6 shows that the proposed algorithm has a comparable accuracy with the incremental DBSCAN algorithm.

As a result, the proposed algorithm maintains a nearly same accuracy with incremental DBSCAN algorithm; however the proposed algorithm has an advantage for the running time with a factor up to 3.2.

Table 2 shows the best speedup factors of the proposed algorithm relative to the incremental DBSCAN algorithm for different datasets.

Table 3 shows the parameters used to conduct the experiments for the different numerical database. The table shows values for Eps, MinPts, number of partitions and Alpha merge.

4.3. Comparison with the incremental similarity-histogram based clustering (SHC) algorithm

Incremental similarity-histogram cluster (SHC) algorithm [23] aims to cluster documents where the similarity histogram is a concise statistical representation of the set of pair-wise document similarities distribution in the cluster and objects are

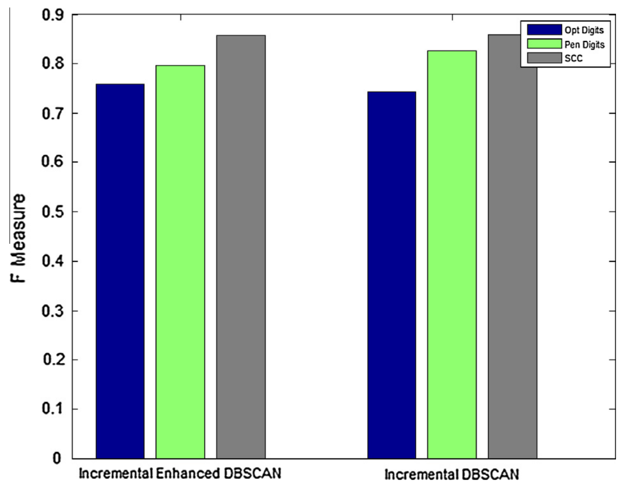

Figure 6 F-Measure for the proposed algorithm and incremental DBSCAN algorithm.

added to a cluster only if they enhance its similarity histogram (i.e. keeping the distribution of similarities skewed to the right of the histogram).

The proposed algorithm is a general purpose clustering algorithm that can operate on any type of data. In this case, the proposed algorithm is evaluated with text data. Fig. 7 shows the speedup factor of the proposed algorithm compared to SHC algorithm. It shows that the proposed algorithm speeds up the clustering process to a factor up to 2.4.

Fig. 8 shows the F-Measure of the proposed algorithm to SHC algorithm. Although SHC gives a slightly better accuracy in UW-CAN dataset, the proposed algorithm gives higher accuracy in 4 Universities and 20 News Groups dataset.

Table 4 shows the parameters used to conduct the experiments for the different textual database. The table shows values for Eps, MinPts, number of partitions and Alpha merge.

Table 5 shows the best speedup factors of the proposed algorithm relative to the SHC algorithm for different datasets.

5. Discussion

An enhanced version of incremental DBSCAN algorithm is proposed in this paper. The proposed algorithm speeds up the clustering process by limiting the search space to partitions instead of the whole dataset. The proposed algorithm is proved to perform better than the existing incremental DBSCAN especially in large datasets with higher number of dimensions.

It is clear that for the textual datasets the speedup factor increases proportionally with the size of the dataset. Given that the average size of documents is comparable in all textual datasets, the proposed algorithm tends to perform better on large datasets.

Table 2 Speedup factor between the proposed algorithm and incremental DBSCAN.

Open digits	Pen digits	SCC
1.85	1.3	3.2

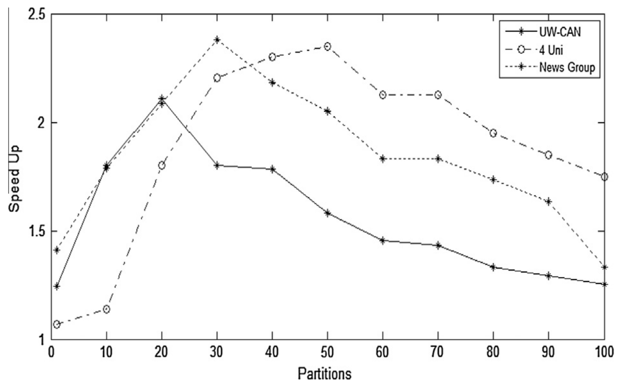

Figure 7 Speedup factor for the textual datasets for different partitions.

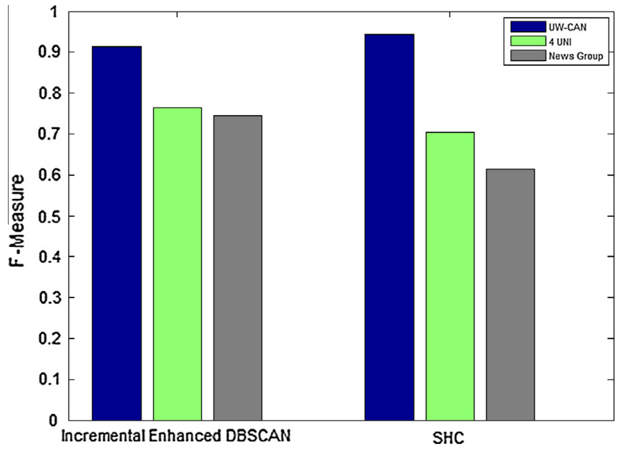

Figure 8 F-Measure for the proposed algorithm and SHC algorithm.

Table 3 Parameters used to conduct the experiments to obtain best performance for numerical datasets.

Dataset	Eps	MinPts	# partitions	Alpha
Open digits	16	2	40	0.1
Pen digits	30	10	45	0.2
SCC	5	18	50	0.1

Table 4 Parameters used to conduct the experiments to obtain best performance for textual datasets.

Dataset	Eps	MinPts	# partitions	Alpha
UW-CAN	20	12	20	0.15
4 Uni	30	6	50	0.23
News Group	15	11	30	0.1

Table 5 Speedup factor between the proposed algorithm and SHC algorithm.

UW-CAN	4 Universities	News Group
2.1	2.3	2.4

The proposed algorithm proves efficiency in clustering large datasets especially with objects of high dimensions. Numerical datasets with different dimensions show that the speedup factor increases with the increase of data dimensions of the objects being clustered. This is expected due to the usage of divide-and-conquer in dataset partitioning which leads to fewer calculations to cluster different objects.

6. Conclusion and future work

In this paper, an incremental density-based clustering algorithm is introduced to incrementally build and update clusters in datasets. The algorithm incrementally partitions the dataset to reduce the search space to each partition instead of scanning the whole dataset. After that the algorithm incrementally forms and updates dense regions in each partition. Following identifying possible dense regions in each partition, the algorithm uses an inter-connectivity measure to merge dense regions to form the final number of clusters. Experimental results show that the proposed algorithm has a comparable accuracy compared to related incremental clustering algorithms. However, the proposed algorithm has significant improvements on the runtime with a speedup factor of 3.2. The proposed algorithm is also proved to perform better in large datasets with higher dimensions compared to related algorithms.

Other possible enhancements to the proposed algorithm are planned for future work. Probably a major enhancement is designing the algorithm to work in a parallel manner. Given the independence of the partitions, incremental DBSCAN for each partition can be applied in parallel. It is expected that the parallel version of the proposed algorithm will achieve better performance with comparable accuracy.

References

[1] A. Nagpal, A. Jatain, D. Gaur, Review based on data clustering algorithms, in: IEEE Conference on Information & Communication Technologies (ICT), April 2013, pp. 298–303.

[2] W. Yu, G. Qiang, L. Xiao-Li, A kernel aggregate clustering approach for mixed data set and its application in customer segmentation, in: International Conference on Management Science and Engineering ICMSE, Oct 2006, pp. 121–124.

[3] Z. Nafar, A. Golshani, Data mining methods for protein–protein interactions, in: Canadian Conference on Electrical and Computer Engineering, CCECE, May 2006, pp. 991–994.

[4] Ahmad M. Bakr, Noha A. Yousri, Mohamed A. Ismail, Efficient incremental phrase-based document clustering, in: International Conference on Pattern Recognition ICPR, Nov 2012, pp. 517–520.

[5] S. Nithyakalyani, S.S. Kumar, Data aggregation in wireless sensor network using node clustering algorithms a comparative study, in: IEEE Conference on Information & Communication Technologies (ICT), April 2013, pp. 508–513.

[6] S. Hyuk Cha, Comprehensive survey on distance/similarity measures between probability density functions, Int. J. Math. Models Methods Appl. Sci. 1 (2007) 300–307.

[7] C. Bahm, K. Haegler, N.S Maller, C. Plant, CoCo: coding cost for parameter-free outlier detection, in: The 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, June 2009, pp. 149–158.

[8] D. Wang, S. Zhu, T. Li, Y. Chi, Y. Gong, Integrating clustering and multi document summarization to improve document understanding, in: The 17th ACM CIKM Conference on Information and Knowledge Management, Oct 2008.

[9] H.-P. Kriegel, M. Pfeifle, Effective and efficient distributed model-based clustering, in: Proceedings of the 5th International Conference on Data Mining (ICDM'05), 2005, pp. 285, 265.

[10] K.M. Hammouda, M.S. Kamel, Efficient phrase-based document indexing for web document clustering, in: IEEE TransKnowledge and Data Eng., vol. 16, no. 10, Oct. 2004, pp. 1279–1296.

[11] Zhe Zhang, Junxi Zhang, Huifeng Xue, Improved K-means clustering algorithm, in: Congress on Image and Signal Processing CISP, vol. 5, May 2008, pp. 169–172.

[12] L. Li, J. You, G. Han, H. Chen, Double partition around medoids based cluster ensemble, in: International Conference on Machine Learning and Cybernetics, vol. 4, July 2012, pp. 1390–1394.

[13] H. Du, Y. Li, An improved BIRCH clustering algorithm and application in thermal power, in: International Conference on Web Information Systems and Mining (WISM), vol. 1, Oct 2010, pp. 53–56.

[14] R.T. Ng, J. Han, CLARANS: a method for clustering objects for spatial data mining, IEEE Trans. Knowl. Data Eng. 14 (5) (2002) 1003–1016.

[15] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithm for large databases, in: Proceedings of the ACM-SIGMOD International Conference Management of Data (SIGMOD'98), Oct 1998, pp. 73–84.

[16] G. Karypis, H. Eui-Hong, V. Kuma, Chameleon: hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.

[17] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proc. 2nd International Conference on Knowledge Discovery and Data Mining, Oct 1996, pp. 226–231.

[18] Z. Wang, Y. Hao, Z. Xiong, F. Sun, SNN clustering kernel technique for content-based scene matching, in: 7th IEEE International Conference on Cybernetic Intelligent Systems, Sept 2008, pp. 1–6.

[19] E. Achtert, C. Bhm, Â H. Kriegel, P. KrÃger, I. Maller-Gorman, A. Zimek, Detection and visualization of subspace cluster hierarchies, in: Advances in Databases: Concepts, Systems and Applications, Lecture Notes in Computer Science, 2007, pp. 152–163.

[20] G. Tzortzis, A. Likas, The global kernel K-means algorithm for clustering in feature space, IEEE Trans. Neural Netw. 20 (July) (2009) 1181–1194.

[21] D. Widyantoro, T. Ioerger, J. Yen, An incremental approach to building a cluster hierarchy, ICDM Proceedings IEEE International Conference on Data Mining, June 2002, pp. 705–708.

[22] S.A.L. Mary, K.R.S. Kumar, A density based dynamic data clustering algorithm based on incremental dataset, J. Computer Sci. 8 (5) (2012) 656–664.

[23] K.M. Hammouda, M.S. Kamel, Incremental document clustering using cluster similarity histograms, in: IEEE/WIC Proceedings International Conference on Web Intelligence, Oct. 2003, pp. 597–601.

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

[25] G. Tzortzis, A. Likas, The global kernel K-means clustering algorithm, in: IEEE International Joint Conference on Neural Networks IJCNN, June 2008, pp. 1977–1984.

[26] S. Young, I. Arel, A fast and stable incremental clustering algorithm, in: Seventh International Conference on Information Technology: New Generations (ITNG), April 2010, pp. 204–209.