Pattern Recognition 58 (2016) 39–48

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A fast DBSCAN clustering algorithm by accelerating neighbor searching using Groups method

K. Mahesh Kumar*, A. Rama Mohan Reddy

Department of Computer Science and Engineering, SVU College of Engineering, Sri Venkateswara University, Tirupati 517502, Andhra Pradesh, India

ARTICLE INFO

Article history:
Received 30 October 2015
Received in revised form
18 February 2016
Accepted 3 March 2016
Available online 12 March 2016

Keywords:
Unsupervised learning
Density based clustering
DBSCAN
Neighborhood graph

ABSTRACT

Density based clustering methods are proposed for clustering spatial databases with noise. Density Based Spatial Clustering of Applications with Noise (DBSCAN) can discover clusters of arbitrary shape and also handles outliers effectively. DBSCAN obtains clusters by finding the number of points within the specified distance from a given point. It involves computing distances from given point to all other points in the dataset. The conventional index based methods construct a hierarchical structure over the dataset to speed-up the neighbor search operations. The hierarchical index-structures fail to scale for datasets of dimensionality above 20. In this paper, we propose a novel graph-based index structure method Groups that accelerates the neighbor search operations and also scalable for high dimensional datasets. Experimental results show that the proposed method improves the speed of DBSCAN by a factor of about 1.5–2.2 on benchmark datasets. The performance of DBSCAN degrades considerably with noise due to unnecessary distance computations introduced by noise points while the proposed method is robust to noise by pruning out noise points early and eliminating the unnecessary distance computations. The cluster results produced by our method are exactly similar to that of DBSCAN but executed at a much faster pace.

1. Introduction

Data clustering is an unsupervised learning technique that groups given data into meaningful subclasses such that objects in the same subclass are more similar compared to objects in other subclasses. Clustering techniques are used in different fields like image analysis [1], pattern recognition [2], knowledge discovery [3] and bio-informatics [4]. Application of clustering techniques in spatial databases [5] poses the challenge to discover clusters with arbitrary shapes, determine the input parameters of algorithm with minimal requirements of domain knowledge and a good efficiency on large databases.

Partitional clustering methods like k-means [6] can find clusters of spherical shape only and need to supply number of clusters as input to the algorithm. Kernel k-means [7] can detect arbitrary shaped clusters by transforming data into feature space using kernel functions. But, it has time and space complexity of $O(n^2)$ and hence not scalable for large datasets. Hierarchical clustering methods partition the dataset into subsets represented by a hierarchical data structure dendogram. Clusters are obtained by merging the subsets at different levels using the minimum distance criteria [8]. Single-linkage [9] method can detect arbitrary shaped clusters but is sensitive to noise patterns. It also suffers from chaining effect [10]. CURE [11] is an improved version of single-link which selects a random sample of points and shrinks them towards the centroid to solve chaining effect. The Hierarchical methods have a time complexity of $O(n^3)$ and should also define an appropriate stopping condition for splitting or merging of partitions while deriving suitable clusters. BIRCH [12], is an agglomerative hierarchical clustering algorithm which uses a tree based representation for reducing time complexity but it can find only spherical shaped compact clusters and also clustering result is effected by input order of data. Recently, multi-view based [13,14] and semi-supervised clustering [15] methods are shown effective in improving accuracy of clustering results. Multi-view based clustering methods exploits information obtained from multiple-views to improve clustering accuracy [16]. Semi-supervised clustering algorithms utilize a small amount of labeled data from the user to achieve a better clustering accuracy. The user provided data may be incorporated into clustering algorithm in the form of constraints for guiding towards a better solution [17,18].

Density Based Spatial Clustering of Applications with Noise (DBSCAN) [19] is the pioneer of density based clustering techniques which can discover clusters of arbitrary shape and also handles noise or outliers effectively. DBSCAN algorithm has a

* Corresponding author. Mobile: +91 9966880476.
E-mail addresses: mahesh_cse@outlook.com (K. Mahesh Kumar), ramamohansvu@yahoo.com (A. Rama Mohan Reddy).

http://dx.doi.org/10.1016/j.patcog.2016.03.008
0031-3203/© 2016 Elsevier Ltd. All rights reserved.

Table 1
Notations.

Symbol	Denotes
D	input set of data patterns
x	a pattern in D
d	number of dimensions of pattern
n	size of dataset
S	set of groups
s	a group in S
s_x	a slave pattern in group s
s_m	master pattern of groups s
s_c	number of patterns in groups s
s_{eps}	eps-connected groups of s*(def. 3)*
s_t	threshold-distance of group s
s_{rec}	reachable groups of s*(def.4)*

quadratic time complexity with dataset size. The algorithm can be extended to large datasets by reducing its time complexity using spatial index structures like R-trees [20] for finding neighbors of a pattern. But, they can not be applied for high dimensional datasets. In this paper we propose an algorithm Groups to accelerate the neighbor search queries. Groups builds a graph-based index structure on the data. Unlike conventional hierarchical index structures like R-trees, the proposed method scales well for high dimensional datasets. Also, Groups method is efficient in handling large amounts of noise present in the data without degrading the performance of DBSCAN. The cluster results produced by our method are exactly identical to that of DBSCAN but at a reduced running time.

Table 1 presents the notations used in the paper. The rest of paper is organized as follows. Section 2 reviews existing work in the literature on density based clustering techniques and a detailed description on DBSCAN method. Section 3 describes the proposed method, with the Groups algorithm followed by G-DBSCAN (Groups-Density Based Spatial Clustering of Applications with Noise). Section 4 provides an experimental analysis of proposed method. Section 5 gives some of the conclusions and future work of proposed method.

2. Related work

Density based clustering methods can find arbitrary shaped clusters in the dataset and also insensitive to noise. In density based clustering methods clusters are formed by merging dense areas separated by regions of spares areas. DBSCAN is proposed for clustering large spatial databases with noise or outliers. OPTICS [21] is an extension to DBSCAN which can find clusters with varying densities by creating an augmented ordering of given dataset representing a density-based cluster structure. This ordering is equal to a density-based clustering with varied range of parameter settings. Chen et al. [22] proposed a parameter free clustering method that utilizes Affinity Propagation algorithm [23] to detect local densities in the dataset and obtains a normalized density list. Later, DBSCAN method is modified to cluster the dataset in terms of the parameters in the normalized density list. DENCLUE [24] defines clusters by a local maximum of estimated kernel density function. A hill climbing procedure is used for assigning points to nearest local maximum. L-DBSCAN [25] is a hybrid density based clustering method that first derives a set of prototypes from the dataset using leaders clustering method [26] and runs DBSCAN on the prototypes to find clusters. Further, Rough-DBSCAN [27] is proposed by applying rough-set theory [28] to L-DBSCAN method. It has a time complexity of $O(n)$ but the cluster results are influenced by threshold parameter that is specified to derive the prototypes. Recently, fast and scalable density based clustering methods using Graphics Processing Units (GPU) are proposed to improve the performance of DBSCAN [29,30]. Also, parallel and distributed versions of DBSCAN are proposed for handling large datasets [31–33]. Mostofa et al. [31] proposed a parallel DBSCAN method using graph algorithmic concepts and achieves a well balanced workload by taking advantage of a tree-based bottom-up approach to construct clusters.

Clustering algorithms based on graph theory [34] are attractive because of their ability to detect clusters of diverse shape, size and densities without requiring any prior knowledge of dataset and does not require user to supply number of clusters as input parameter. These methods are based on constructing a similarity graph representation of dataset which is suitably partitioned and merged to obtain final clusters. CHAMELEON[1] [35] represents given dataset with a k-nearest neighbor graph, and is partitioned into sub-clusters by minimizing edge-cut effectively. Clusters are obtained by merging sub-clusters only if the relative inter-connectivity and closeness between them is comparable. Graph based methods also take advantage of Minimum Spanning Tree (MST) to represent the dataset [36,37]. Barrios [38] identified clusters by comparing k-nearest neighbor based graph MST of a dataset. Spectral clustering [39] represents a fully connected graph of the dataset and is based on spectral-graph theory for finding clusters. In addition, relative neighbor graphs are also used to cluster data [40,41]. Mimaroglu et al. [42] adopts a similarity graph for combining multiple clustering results into a final clustering solution. Graph-based manifold learning methods [43,44] also employs a neighborhood graph representation of dataset to identify clusters. Ju et al. [43] proposed a graph-based manifold learning framework for image clustering using a sparse representation to select a few neighbors of each data point that span a low-dimensional affine subspace passing near that point. A multimodal hypergraph learning based sparse coding method [44] is proposed for the click prediction of images. In a hypergraph a set of vertices are connected by a hyperedge to preserve the local smoothness of the constructed sparse codes.

2.1. DBSCAN: a density-based approach

DBSCAN algorithm defines cluster as a region of densely connected points separated by regions of non-dense points. If similarity measure is taken as Euclidean distance the region is a hyper sphere of radius *eps* at the given point p as center.

- *eps-neighborhood:* for a point $x \in D$, the *eps-neighborhood* denotes the set of points whose distance from x is less than or equal to *eps*. The cardinality of *eps-neighborhood* defines the threshold density of x.
- *eps-connected***:** for a pair of points $x, y \in D$, if $\|x-y\| \le eps$, then x, y are *eps-connected* points.

From the view of a DBSCAN method every point in the dataset will fall into either core point or border point. Further a border point can be either noise point or density connected point.

- Core point: A point with threshold density greater than or equal to *minpts*.
- Border point: A point with threshold density less than *minpts*.
- Noise point**:** A point p is a noise point if the threshold density of p is less than *minpts* and all points in the *eps-neighborhood* of p are border points.

[1] CHAMELEON is a type of reptile that has an ability to change its skin to different colors. The algorithm was called so as it is based on a dynamic model to identify varied shape cluster structures.

- Density-connected point: A border point with at least one core point in its *eps-neighborhood*.

DBSCAN algorithm takes two input parameters *eps* and minpts.*eps* specifies the maximum distance neighborhood for given point. *minpts* is the minimum number of points required in the *eps-neighborhood* of a point to form a cluster. Initially all points are marked *unvisited.* The algorithm starts by randomly selecting an *unvisited* point and finding its *eps-neighborhood.* If the number of points in its *eps-neighborhood* is less than *minpts* then it is marked as noise or outlier, otherwise it is considered as dense-point and a new cluster is created. Further points are added iteratively to the cluster by finding dense points for each point in the *eps-neighborhood* of the cluster. If no *unvisited* points can be added to cluster, the new cluster is complete and no points will be added to the cluster in subsequent iterations. To find out the next cluster find an *unvisited* point in the dataset and repeat the above clustering process. The process is halted when all the points are either assigned to some cluster or marked noise. Every point in a cluster is *eps-connected* with at least one point in the same cluster to which it belongs and is not *eps-connected* with any other points in remaining clusters. However, there may exist a border point which is *eps-connected* with points in some other clusters. In which case, the point is assigned to the cluster that processed it first. Such exceptional cases are rare in practice. The total number of *eps-neighborhood* operations performed is equal to the size of dataset. If no index structures are used then *eps-neighborhood* involves computing distances to all remaining points in the data set.

Algorithm 1. DBSCAN $(eps, minpts, D)$

```
mark all patterns in $D$ as unvisited
    $cluid \leftarrow 1$
  for each unvisited pattern $x$ in $D$
    do
        $Z \leftarrow$ FindNeighbours$(x, eps, minpts)$
        if $|Z| < minpts$
        mark $x$ as noise
      else
        mark $x$ and each pattern of $Z$ with $cluid$
        $queue\_list \leftarrow$ all unvisited patterns of $Z$
        until $queue\_list$ is empty
        do
          $y \leftarrow$ delete a pattern from $queue\_list$
          $Z \leftarrow$ FindNeighbors$(y, eps, minpts)$
            if $|Z| \geq minpts$
              for each pattern $w$ in $Z$
                  mark $w$ with $cluid$
                if $w$ is unvisited
                  $queue\_list \leftarrow w \cup queue\_list$
                end for
              end if
            mark $y$ as visited
        end until
      end if
    mark $x$ as visited
    $cluid \leftarrow cluid + 1$
  end for
Output all patterns in $D$ marked with $cluid$ or noise
```

3. Density based clustering method using groups

In this section, we propose the G-DBSCAN algorithm which is basically a DBSCAN clustering method but the nearest neighbor search queries are accelerated by using Groups method. The proposed algorithm runs in two phases: In the first phase Groups algorithm is run on the entire dataset to obtain a set of groups. The Second phase runs conventional DBSCAN method by using groups derived in the first phase for a fast *eps-neighborhood* operation.

3.1. Groups

Groups method partitions the dataset by fitting into a graph–based structure where each vertex is a group and an edge is drawn between two groups if they are reachable (*def.4*). The Groups algorithm merges nearby patterns into groups. Each group is a hyper sphere with its center as master pattern and can have a maximum radius of *eps*. Groups method classifies each pattern into either master or slave pattern. Groups are formed by scanning the entire dataset twice. In the first round each pattern is searched for some existing group to fit in. A pattern is added to a group if the distance from the given pattern to its master pattern is less than or equal to *eps*. If the distance from given pattern to master pattern of a group is two times *eps*, then such patterns are neither assigned to any group nor itself is created as a new group. Such patterns are processed further in the second round of the algorithm. If a pattern does not fit into any group and distance from master pattern of its nearest group is greater than or equal to two times , then a new group is created with itself as master pattern. In the second round ,the left out patterns in first round are assigned to a group if the distance from the given pattern to master pattern is less than or equal to *eps*. If there is no such group to fit then a new group is created with given pattern as master pattern of the group. Different input order of patterns produces different set of groups (Fig. 1). Whenever a slave pattern is added to a group the threshold distance of the group is also updated. The maximum threshold distance of groups created in first iteration is less than or equal to *eps*. The threshold distance of groups created in second iteration is less than *eps*. The Groups method fits a graph based representation of dataset such that if x_1 is a point in group s then all patterns in *eps-neighborhood* of x_1 will be from either s or s_{rec}

The following are some of the advantages of Groups method over conventional index-based structures:

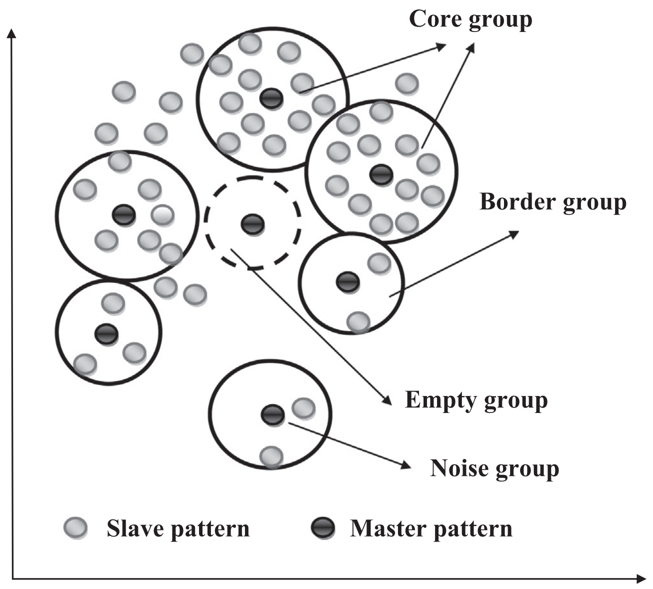

Fig. 1. Groups method: core, border, empty and noise groups.

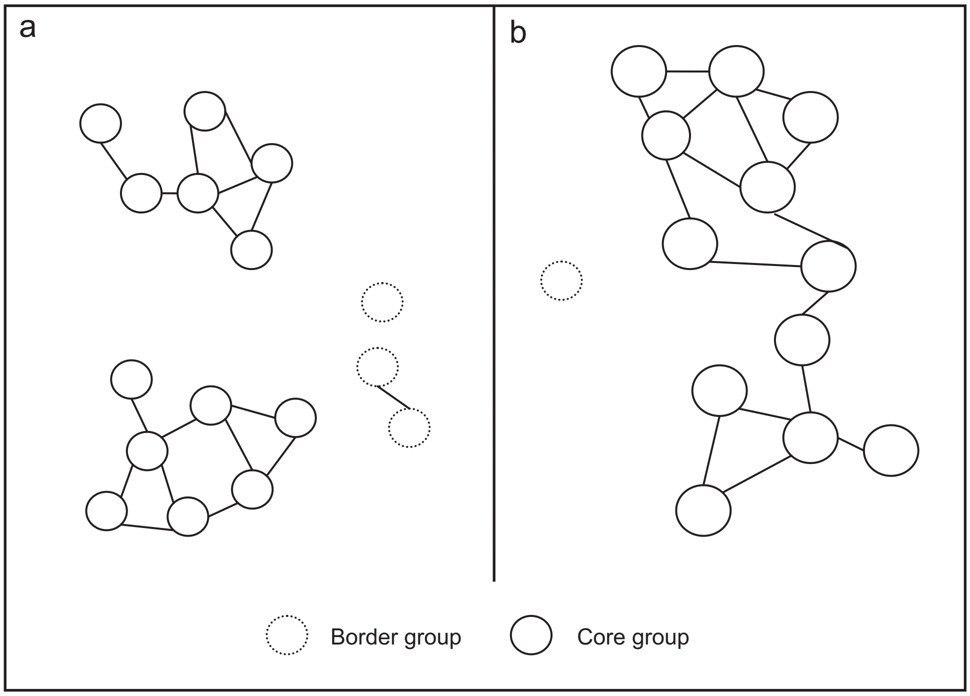

Fig. 2. (a) Groups obtained as multiple disconnected graph components of dataset (left) and (b) Groups obtained as single graph component of dataset (right).

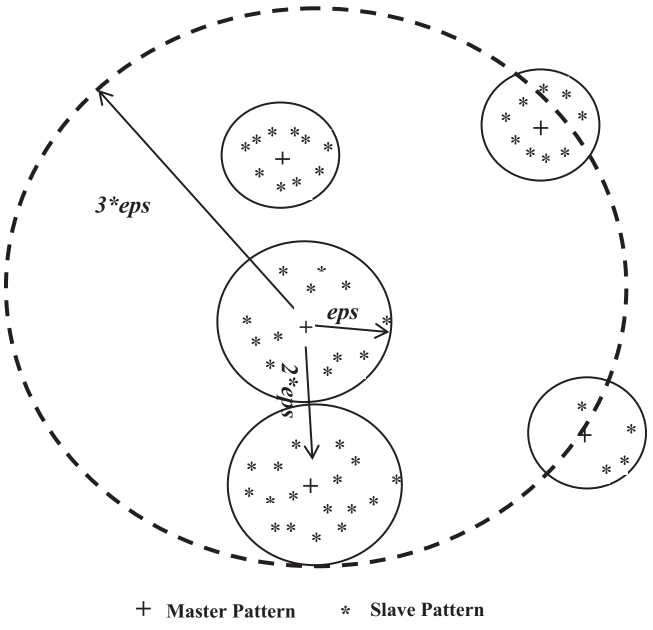

Fig. 3. Group with its reachable groups.

(1) Groups method does not require any input parameters specific to the algorithm. It takes *eps* as input parameter which is specified by user for DBSCAN clustering.
(2) Groups method handles noise effectively by pruning them early without performing *eps-neighborhood* operations while obtaining clusters.
(3) Groups method ensures search space of a pattern involving *eps-neighborhood* operation is always span over a small area (Fig. 3) irrespective of outliers in the data.

Definition 1. (*core group*) A group *s* is a *core group* if the number of patterns in it is greater than or equal to *minpts*.

$S_{core} = \{s \in S | s_c \geq minpts\}$

Definition 2. (border group) A group s is a border group if the number of patterns in it is less than minpts.

$S_{border} = \{s \in S | s_c < minpts\}$

Definition 3. (*eps-neighborhood of a group*) The *eps-neighborhood* of a group *s*, is defined as follows

$s_{eps} = \{w \in S; \exists w_x, \exists s_x | w_x – s_x \leq eps\}$

w, s are called as *eps – connected groups*.

Definition 4. (Reachable groups) The set of reachable groups of a group s is defined as follows

i) $s_{rec} = \{w \in S | \|s_m - w_m\| \leq eps + s_t + w_t\}$

ii) $s_{eps} \subseteq s_{rec}$

The reachable group relationship is symmetric viz., if *u* is a reachable group of *v* then *v* is reachable group of *u*.

Definition 5. (*empty group*) A group *s* is an *empty group* if it does not contain any slave patterns.

$S_{empty} = \{w | w \in S | w_c = 0\}$

Definition 6. (noise group) A group s is a noise group if the number of patterns in *s* is less than *minpts* and is not eps-connected with any other group in S/s (Lemma 3).

$S_{noise} = \{w | w \in S, w_c < minpts \text{ and } w_{eps} = \varnothing\}$

Definition 7. (*Threshold Distance*) Threshold distance of a group is the maximum distance of a slave pattern from its master pattern in a group.

i) $s_t = arg\ \max\ (\|s_m - s_x\|)$

ii) $s_t \le eps$

Algorithm 2. Groups (D, eps)

```
mark all patterns in $D$ as *assigned*
    **for** each pattern $x$ in $D$
        **if** there exists a group $s$ in $S$ such that $\|s_m - x\| \le eps$
            $s \leftarrow s \cup \{x\}$
        **else if** $S$ is empty or there does not exists any group $s$ in $S$
            such that $\|s_m - x\| \le 2 * eps$
            createNewGroup($x$)
        **else** mark $x$ as *unassigned*
        **end if**
    **end for**
**for each** *unassigned* pattern $x$ in $D$
    **if** there exists a group $s$ in $S$ such that $\|s_m - x\| \le eps$
        $s \leftarrow s \cup \{x\}$
    **else** createNewGroup($x$)
    **end if**
**end for**
**Procedure** createNewGroup ($x$)
        create a new group $s$
            $s_m \leftarrow x$
            $s_{rec} \leftarrow$ find reachable Groups of $s$ using Eq. (4)
            $S \leftarrow \{s\} \cup S$
**end procedure**
Output $S$
```

3.2. G-DBSCAN

Groups DBSCAN (G-DBSCAN) is similar to the conventional DBSCAN clustering method but the nearest neighbors are searched for using the groups obtained by Groups algorithm. The reachable groups of a group are computed by finding the distance between their master patterns. If the distance between master patterns of any two groups is less than or equal to sum of their threshold distances and *eps*, they are considered as reachable groups of each other(*def.3*) .since the threshold distance of a group is not known until the completion of scanning the entire dataset, we take threshold distance of groups as their maximum value, *eps*.Hence, two groups are considered reachable if the distance between their master patterns is less than or equal to three times *eps* (Corollary 1). If x, y are any two *eps-connected* patterns then x, y are either patterns that belong to same group or patterns in reachable groups. To find out the *eps-neighborhood* of a pattern, firstly all patterns in the current group are searched for *eps-connectivity* and followed by searching patterns in its reachable groups. If the threshold distance of current group is less than or equal to $eps/2$, then its *eps-neighborhood* includes all patterns in the current group (Lemma 3). If s is a border group without any reachable groups then all patterns in s are noise patterns. For a given pattern in a group, whether there exists an *eps-connected* pattern in its reachable groups or not is determined by using Eq. (3). If Eq. (3) is satisfied then distance computations are made from given pattern to all patterns in its reachable groups for obtaining *eps-connected* patterns. The search space for computing *eps-neighborhood* of a pattern is bounded by a hyper-sphere of radius $5 * eps$ with the given pattern as its center (Fig. 7). Algorithm 3 specifies the process of computing *eps-neighborhood* of a given pattern using groups method. In the DBSCAN clustering given in Algorithm 1, the step findNeighbors utilizes Algorithm 3.

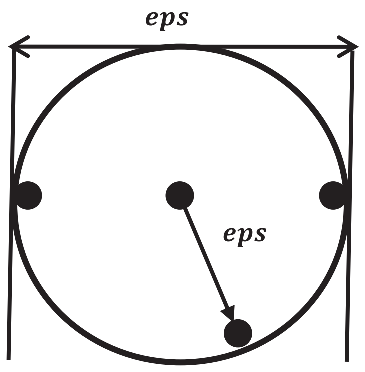

Fig. 4. Group with maximum-distance of slave patterns.

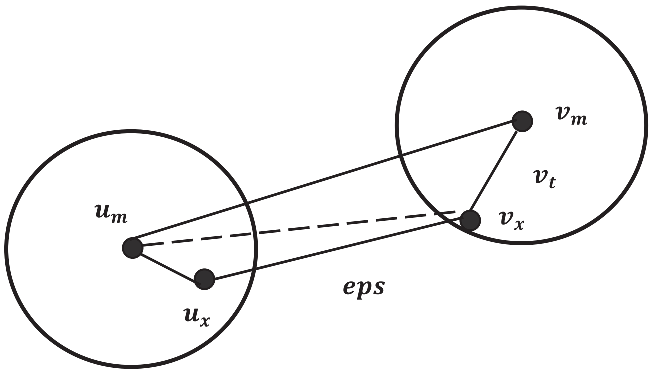

Fig. 5. eps-connected groups.

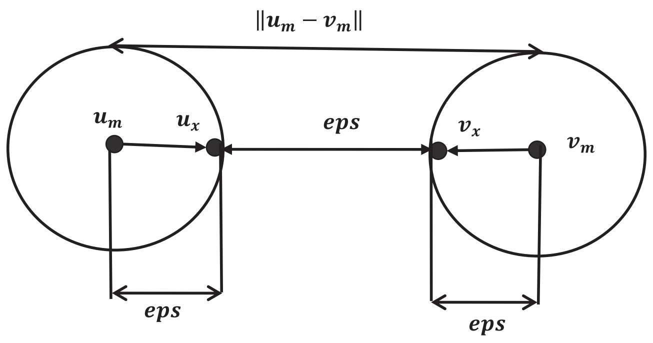

Fig. 6. Maximum separated *eps-connected* groups.

Our method is efficient than other graph-based clustering methods like CHAMELEON [35] that uses a k-nearest neighbor graph for representing dataset. Constructing a k-nearest neighbor graph involves finding distance computations from given point to all remaining points in the dataset. The graph is partitioned into multiple disconnected components (sub-clusters) by removing the longest edges and an agglomerative hierarchical clustering process is applied on sub-clusters to obtain final clustering results. CHAMELEON [35] is a hierarchical clustering method and requires user to specify suitable split and merge policy in addition to parameter k. G-DBSCAN is a density based clustering method that uses an efficient graph based structure for fast neighbor search operations. G-DBSCAN finds a graph-based representation of dataset by scanning the entire dataset twice and involves distance computations from given point to master pattern of groups only. Each vertex is a group represented by its master pattern and *eps*-

neighborhood patterns of master are added as its slaves. The edges are drawn towards its reachable groups (Fig. 2(a)). If clusters are well separated and valid parameters of *eps* and *minpts* selected, Groups method obtains disconnected graph components which are equal to number of clusters present in the data. (Fig. 2(b)). Our method is parameter free without requiring any specific inputs from user.

Algorithm 3. FindNeighbours (*pat_index,eps*).

```
/* This method finds the eps- neighborhood of a given
  pattern and returns them */
      $neighbor\_list \leftarrow \varnothing$
      $cur\_group \leftarrow \text{groupof}(pat\_index)$
        if $cur\_group_t \le eps/2$
      $neighbor\_list \leftarrow$ all patterns in $cur\_group$
        else find patterns in $cur\_group$ such that
          $\|cur\_group_x - pat\_index\| \le eps$
        $neighbor\_list \leftarrow neighbor\_list \cup cur\_group_x$
        end if
        for each reachable group $rec\_group$ of $cur\_group$
          if
    $\|cur\_group_m - rec\_group_m\| \le rec\_group_t + eps +$
              $\|cur\_group_m - pat\_index\|$
            find patterns in $rec\_group$ such that
              $\|rec\_group_x - pat\_index\| \le eps$
            $neighbor\_list \leftarrow neighbor\_list \cup rec\_group_x$
          end if
    end for
    Output $neighbor\_list$
```

Lemma 1. *All patterns in a core group belong to same cluster.*

Proof. Let s be a core group and s_{x_1}, s_{x_2} be any two slave patterns in group s. let s_m belongs to cluster *cid*, since s is a core group s_{x_1}, s_{x_2} belongs to *eps-neighborhood* of s_m and also number of patterns in s is greater than or equal to *minpts*. Therefore, s_{x_1}, s_{x_2} also belongs to *cid*. Hence, all patterns of a core group belong to same cluster.□

Lemma 2. *If set of reachable groups is empty for a border group, then all patterns in the border group are noise.*

Proof. Let s be a border group, since number of slave patterns is less than *minpts* in a border group, s_m is *noise*. Since s_{rec} is empty, *eps-neighborhood* of any slave pattern in s includes patterns in s only. As total number of patterns in s is less than *minpts*, *eps-neighborhood* of any pattern in s, can not be more than *minpts*. Hence, each pattern in s is a *noise.*□

Lemma 3. *If $s_t \le eps/2$ then the eps- neighborhood for any pattern in s includes all remaining patterns in s.*

Proof. From Fig. 4, it is evident that if the threshold distance of a group is *eps*/2, then the maximum distance between any two patterns in the group is *eps*. Therefore, every pattern in the group will fall in the *eps-neighborhood* of remaining patterns in the group.□

Theorem 1. *Let u, v be any two groups $\in S, v \in u_{rec}$, u, v are eps-connected with respect to u_x only if $\|u_m - v_m\| \le \|u_m - u_x\| + eps + v_t$*

Proof. Using the triangle inequality property from Fig. 5,

$$\|u_m - u_x\| + eps \ge \|u_m - v_x\|$$

$$\|u_m - v_x\| + v_t \ge \|u_m - v_m\| \tag{1}$$

$$\|u_m - v_x\| \ge \|u_m - v_m\| - v_t \tag{2}$$

from transitivity in (1) and (2),

$$\|u_m - u_x\| + eps \ge \|u_m - v_m\| - v_t$$

$$\|u_m - v_m\| \le \|u_m - u_x\| + eps + v_t \tag{3}$$

□

Corollary 1. *The distance between master patterns of any two eps-connected groups is less than or equal to $3 * eps$.*

Proof. Fig. 6 shows a pair of eps-connected groups separated by maximum distance. Let u, v be any two eps-connected groups, by using Eq. (3) the distance between their master patterns $\|u_m - v_m\|$ is less than or equal to $\|u_m - u_x\| + eps + v_t$. Since distance between master pattern and slave pattern of a group is always less than or equal to *eps*, substituting *eps* for v_t and $\|u_m - u_x\|$ in above equation yields

$$\|u_m - v_m\| <= eps + eps + eps$$

$$\|u_m - v_m\| <= 3 * eps \tag{4}$$

□

Theorem 2. *The distance between any two slave patterns of reachable groups is less than or equal to $5 * eps$.*

Proof. Let u, v be any two groups $\in S, v \in u_{rec}$, using the triangle inequality property from Fig. 7,

$$\|u_m - v_m\| + \|v_m - v_x\| \ge \|u_m - v_x\| \tag{5}$$

$$\|u_m - v_x\| + \|u_x - u_m\| \ge \|u_x - v_x\|$$

$$\|u_m - v_x\| \ge \|u_x - v_x\| - \|u_x - u_m\| \tag{6}$$

From transitivity in Eqs. (5) and (6)

$$\|u_m - v_m\| + \|v_m - v_x\| \ge \|u_x - v_x\| - \|u_x - u_m\|$$

$$\|u_x - v_x\| \le \|u_x - u_m\| + \|u_m - v_m\| + \|v_m - v_x\|$$

$$\|u_x - v_x\| \le u_t + \|u_m - v_m\| + v_t \tag{7}$$

Since, $u_t, v_t \le eps$ and from Eq. (4),

$$\|u_x - v_x\| \le eps + 3 * eps + eps$$

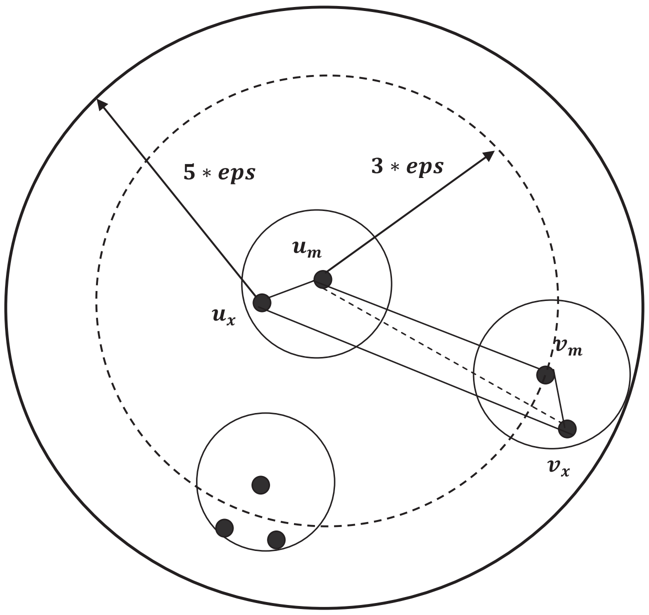

Fig. 7. Maximum distance slave patterns in reachable groups.

$$\|u_x - v_x\| \le 5 * eps \tag{8}$$

□

3.3. Performance analysis

The time complexity of proposed algorithm involves the cost incurred in deriving groups from the dataset and running the G-DBSCAN algorithm on the groups. The entire dataset is scanned once to obtain an initial partition of groups and the patterns not fitting into any group are processed further in the second round. If p be the number of patterns left unassigned to any group in the first iteration, then the time complexity for groups algorithm is $O(n+p)$.since $p \ll n$, it can be taken as $O(n)$. The *eps-neighborhood* of a pattern involves computing the distance between all patterns in the reachable groups satisfying Eq. (3). In the worst case, it involves computing distance between patterns present in all of its reachable groups. Without using any index–based structure *eps-neighborhood* of a point involves computing distance from all other remaining points in the dataset, hence its time complexity is $O(n^2)$. Using an index structure like R-Trees, the search time can be reduced to $O(\log n)$, overall for entire dataset it will be $O(n \log n)$. However, index-based techniques are not efficient for high dimensional data. The Groups method involves neighbor searching limited to only in its reachable groups. Further for a given pattern, the reachable groups are pruned out of searching based on triangle inequality property (Eq. (3)). If d denote the maximum number of distance computations involved in computing *eps-neighborhood* for a pattern, then the time complexity of DBSCAN using groups is $O(nd)$. The time complexity of G-DBSCAN, including the time taken for groups is $O(n+nd)$. Though the value of d can not be established theoretically, for all valid parameters of *eps*, d is small. The value of d is influenced by the input order of data and also separation between clusters. Since the distance between slave patterns of reachable groups is less than or equal to $5 * eps$ (Theorem 2), the *eps-neighborhood* of a given pattern in the worst case involves distance computations to patterns at a distance of $5 * eps$ from the given pattern.

4. Experimental results

In this section, we present the experimental results performed to demonstrate the effectiveness of proposed method. In the first part, we performed experiments on datasets obtained from UCI machine learning repository [45] of varied dimensions and a two dimensional synthetic dataset (Fig. 8). In the second part, we performed experiments on synthetic datasets of dimensions in varied range. In both cases the performance of G-DBSCAN is empirically evaluated with the conventional DBSCAN and its state-of-the-art index implementations. In the third part, the behavior of algorithm is analyzed in the presence of noise of varied sizes in the dataset. All the experiments are performed on Intel core i3-4005U processor with 1.7 GHz and 4 GB RAM running windows 7 ultimate service pack -1. All programs are compiled and executed as single-threaded java console applications using JDK 8u45 on Java HotSpot 64-bit server Virtual Machine.

The proposed method is compared with three commonly used Index data structures for neighbor searching: k-d trees, R-trees and M-trees. k-d tree [46] is a multi-dimensional search tree where each node is k-dimensional point. The nodes are split recursively by using mean or median of data points across each node. In our experiments median was used. R-tree [15] is a balanced search tree used for spatial access methods. It groups near by objects into a minimum bounding rectangle. Sort-Tile-Recursive (STR) [47] is a variant of R-tree bulk-loaded that

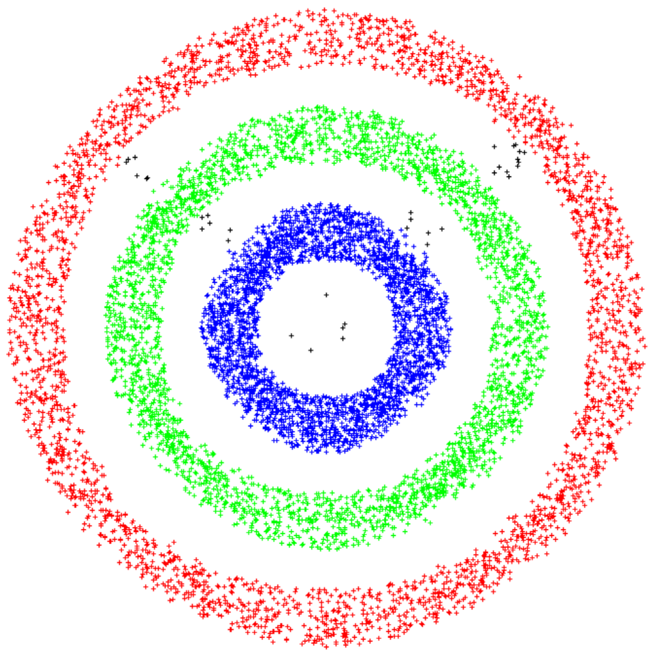
Fig. 8. Synthetic dataset concentric rings.

repeatedly splits across each dimension successively into equal sized partitions. STR was used in our experimental study. M-tree [48] is a balanced search tree similar like R-Tree, but it uses minimum volume hyper spheres for grouping near by objects instead of hyper-rectangles. Random split policy is used for dividing the hyper spheres with minimal overlap. The hierarchical index structures are sensitive to input parameters. In case of M-trees a performance difference of up to two to three orders of magnitude in running time was observed with respect to the split policy selected. In each case the algorithm is executed five times with different input orders of dataset and their average is taken as the running time. Experimental results show that the proposed method is faster than DBSCAN by a factor of 1.5–2.2 on benchmark datasets and is also scalable for high dimensional datasets (Fig. 9).

4.1. Experiment 1

In this empirical study we had used seven popular datasets from UCI machine learning repository and one synthetic dataset. The datasets are selected on the criteria of its corresponding dimension and size. *Combined cycle Power plant* [49] consists of 9568 data points collected from a power plant over a period of six years .Each point comprise of four attributes obtained when the power plant was set to work at full load. They are used to calculate net hourly electrical energy output of the plant. *Page blocks classification* [50] is composed of 5473 blocks obtained from 54 distinct documents. The blocks of the page layout of a document are obtained using a segmentation process. The blocks are classified to separate text from graphic areas. *Pen-based recognition of handwritten digits* [45] is a digit database of 250 samples collected from 44 writers. Each sample is a 16-dimensional feature vector. *Letter recognition* [45] consists of database of image features used to identify 26 upper case English alphabets. *Image segmentation* [45] consists of seven classes of 2310 samples of 19 dimensions. *Statlog Landsat Satellite* [45] consists of seven classes of 6435 instances. Each instance consists of 36 features obtained from multi-spectral values of pixels in 3×3 neighborhoods in a satellite image. *Plant species Leaves* dataset [51] comprise of 1600 instances obtained from sixteen samples of leaf from one-hundred plant species. *Concentric rings* is artificially generated with 3 classes of 3000

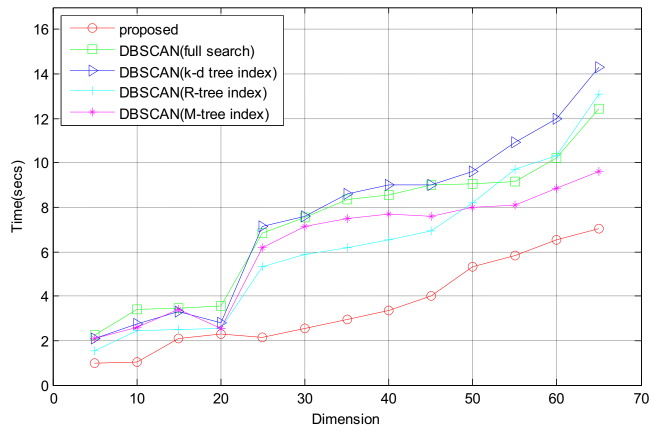

Fig. 9. Running time comparison of proposed, index methods with dimension of data.

Table 2
Characteristics of datasets used in testing.

Dataset	Size	Dimension
Combined Cycle Power plant	9872	4
Page blocks	5473	10
Pen-digits	10,992	16
Letter recognition	20,000	16
Image segmentation	2310	19
Statlog landsat satellite	6435	36
Plant species leaves	1600	64
Concentric rings	9030	2

Table 3
Running time(s) comparison of datasets for various indexes.

Dataset	DBSCAN (full search)	DBSCAN using index			
		Groups	k-d tree	R-tree	M-Tree
CC Power plant	1.706	1.057	1.245	1.580	1.336
Page blocks	1.024	0.637	0.720	0.840	0.650
Pen-digits	4.708	2.213	3.495	3.835	3.978
Letter recognition	14.63	8.807	17.12	14.97	16.97
Image segmentation	0.221	0.131	0.231	0.277	0.357
SL Satellite	2.967	1.907	3.117	2.885	2.837
Plant species leaves	3.348	2.011	3.985	4.033	3.751
Concentric rings	0.982	0.459	0.594	0.562	0.716

points each and 30 noise points. A summary of datasets used in the empirical study is provided in Table 2.

A performance comparison of proposed method with index-structures is given in Table 3. From the experimental results, it is observed that index based structures perform well for datasets with dimensionality less than 20 and tend to perform poor as the dimensionality of the dataset increases. Our method is more than twice faster than DBSCAN for pen-digits dataset. For plant species dataset, a performance improvement of about 40 percent in the running time is observed while the index based methods perform slower than DBSCAN. L-DBSCAN [25] and Rough-DBSCAN [27] uses a prototype based hybrid approach to speed up the DBSCAN clustering method. L-DBSCAN requires two input parameters τ_c and τ_f that are used to create prototypes at coarse-level and fine-grain level. Rough-DBSCAN uses an input parameter τ_c to create prototypes at coarse-level only. A performance comparison of execution time and Rand-Index of G-DBSCAN with the above methods is shown in Table 4. The clustering results are compared using the similarity measure *Rand-Index* [52]. The Rand-index is computed by using Eq. (9). Let X, Y denote two different sets of partitions of a dataset, a denotes the number of pairs of patterns in the dataset that are present in a set of X and Y. Let b denotes the number of pairs of patterns in the dataset that are not grouped into a set of both X and Y.

$$Rand-Index(X, Y) = \frac{(a+b)}{\binom{n}{2}} \tag{9}$$

It is observed that by suitable selection of input parameter(s) L-DBSCAN, Rough-DBSCAN can give a reduction in execution time better than G-DBSCAN but they deviate from the clustering results produced by that of DBSCAN considerably. Also, the selection of input parameter(s) that minimize the execution time and maximize the clustering accuracy simultaneously is a difficult task and the authors did not report any effective criteria for determining such suitable input parameter(s) in their respective papers [25,27]. G-DBSCAN improves the execution speed of DBSCAN and always guarantees the clustering results are exactly identical to clustering results produced by DBSCAN ($Rand\text{-}Index = 1$).

4.2. Experiment 2

In this experiment, the scalability of proposed method is analyzed for high dimensional datasets. Experiments were performed on synthetic datasets each of size 10,000 and dimensions ranging from 5 to 65 in steps of 5. Data sets are generated using a multivariate normal distribution, where covariance is taken as an $d \times d$ identity matrix and cluster centers are generated by random sampling from a uniform distribution $[-1, 1]^d$. A running time comparison of proposed method with index based implementations of DBSCAN is shown in Fig. 9. It observed that the conventional index based techniques fail to scale for datasets with dimensions above 20. Further, the running time for index based implementations goes even worse than DBSCAN above some threshold value of dimension while the proposed method outperforms than index-based implementations even at higher dimensions.

4.3. Impact of noise

One of the enviable features of spatial clustering methods is the ability to handle noise effectively and perform well. In this experiment, the performance of proposed method is analyzed in the presence of noise. For the experiment purpose, we generate a two dimensional synthetic dataset of 10,000 points using a multivariate normal distribution. The clusters centers are generated by sampling from a uniform distribution $(0, 1)$ and covariance is taken as an 2×2 identity matrix. Noise is added by random sampling from a uniform distribution (-2, 2). At each step, noise is added in increments of 500 from 500, 1000 and so on up to n points. It is ensured that majority of noise points are not *eps-connected*. All patterns whose *eps-neighborhood* is empty constitute an empty group while all *eps-connected* noise patterns constitute a border group or reachable empty groups. The running time of Groups method increases with the number of noise points since the process of assigning slaves to their respective groups involves searching more number of empty and noise groups that are introduced by noise points. During G-DBSCAN the empty noisy groups does not involve any distance computations for computing *eps-neighborhood* of the pattern while the points in noisy border groups involves distance computations less than *minpts*. From Fig. 10, it is evident that with an increase in noise the execution time of Groups method is increasing exponentially where as running time of G-DBSCAN varies by a small value. In the presence

Table 4
Comparative analysis of running time and rand-index of G-DBSCAN.

Dataset	L-DBSCAN				Rough-DBSCAN			G-DBSCAN		DBSCAN
	Input parameters		Running time (s)	Rand-index	Input parameter	Running time (s)	Rand-index	Running time (s)	Rand-index	Running time (s)
	τ_c	τ_f			τ_c					
Pen-digits	5	2.5	3.287	0.987	7.5	3.812	0.999	2.213	1	4.708
	13.5	3.5	2.785	0.912	11	3.145	0.981			
	25	1.5	2.315	0.871	27.5	2.245	0.913			
	42.5	0.5	2.102	0.813	43	1.985	0.861			
Letter recognition	0.15	0.1	12.25	0.982	0.15	11.21	0.998	8.807	1	14.63
	0.35	0.25	10.13	0.962	0.35	9.21	0.961			
	0.45	0.32	9.124	0.891	0.48	8.512	0.921			
	0.65	0.48	8.321	0.834	0.65	7.14	0.845			
CC Power plant	2	1.5	1.549	0.998	2.5	1.512	0.999	1.057	1	1.706
	6.5	2.5	1.364	0.941	5	1.385	0.972			
	12.5	3.5	1.142	0.865	10.5	1.123	0.925			
	17	5	0.912	0.812	16	0.896	0.832			

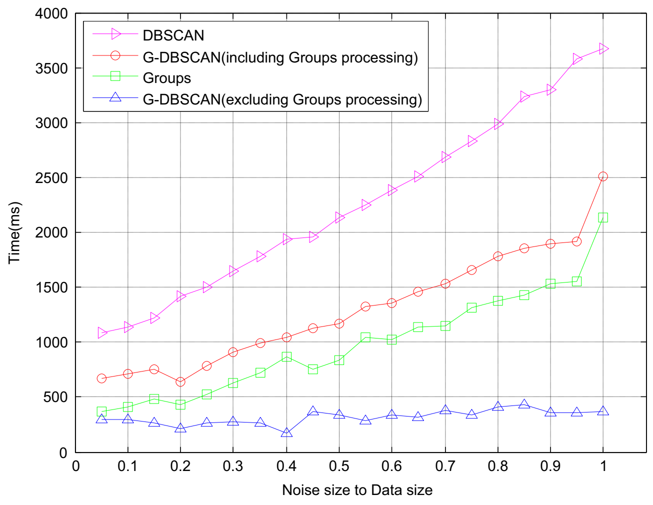

Fig. 10. Running time comparison of DBSCAN, G-DBSCAN and Groups with noise.

of noise G-DBSCAN gives a performance improvement in running time by a factor of 1.4–2.2 when compared to DBSCAN.

5. Conclusions and future work

In this paper we have presented a graph-based index structure Groups to speed up the neighbor search operations of DBSCAN clustering. Groups method scans entire data set once to obtain a set of groups. A pattern which does not fit into an existing group is processed in the second round. Such patterns are assigned to a group as slave or a new group is created with itself as the master pattern. Groups method ensures that always for a given pattern the neighbor searching does not need to move points farther than distance of $5 * eps$, which is an advantage over conventional DBSCAN that requires searching all patterns in the dataset. It is observed from experimental analysis that inappropriate parameter values for hierarchical index construction gives a performance degradation of up to two to three fold magnitude of actual running time. Groups method is more stable than index-based structures as it do not any require specific input parameters from user to build the groups index structure. Also, Groups method is robust to noise by pruning outliers early with zero or few distance computations. In future we are planning to extend G-DBSCAN for very large datasets using parallel and distributed versions of proposed method, incorporating high performance computing (HPC) techniques.

Conflict of interest

None declared.

Acknowledgments

The authors would like to thank anonymous reviewers for their valuable suggestions in improving the quality of paper. The work of Mahesh was supported by a monthly scholarship from the Department of Higher Education, Ministry of Human Resource Development (MHRD), Government of India, under Technical Education Quality Improvement Program (TEQIP-II-1.2).

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.patcog.2016.03.008.

References

[1] R.C. Gonzalez, R.E. Woods, Digital Image Processing, third ed., Pearson Prentice-Hall, Upper Saddle River, NJ, 2008.
[2] S. Theodoridis, K. Koutroumbas, Pattern Recognition, second ed., Academic Press, New York, 2003.
[3] U.M. Fayyad, G.P. Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, MIT Press, Boston, MA, 1996.
[4] S. Madeira, A. Oliveira, Bi-clustering algorithms for biological data analysis: a survey, IEEE/ACM Trans. Comp. Biol. Bioinforma. 1 (1) (2004) 24–45.
[5] R.H. Gueting, An introduction to spatial database systems, VLDB J. 3 (4) (1994) 357–399.
[6] A.K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[7] M. Girolami, Mercer kernel-based clustering in feature space, IEEE Trans. Neural Netw. 13 (3) (2002) 780–784.
[8] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.

[9] B. King, Step-wise clustering procedures, J. Am. Stat. Assoc. 62 (317) (1967) 86–101.
[10] G. Nagy, State of the art in pattern recognition, in: Proceedings of IEEE 56, 1968, pp. 836–862.
[11] S. Guha, R. Rastogi, K. Shim, Cure: an efficient clustering algorithm for large databases, in: Proceedings of the International Conference on Mangement of Data (ACM SIGMOD), 1998, pp.73–84.
[12] Z. Tian, R. Raghu, L. Micon, BIRCH: an efficient data clustering method for very large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1996, pp.103–114.
[13] X. Chang, T. Dacheng, X. Chao, Multi-view self-paced learning for clustering, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, 2015, pp. 3974–3980.
[14] X. Chang, T. Dacheng, X. Chao, Large margin multi-view information bottleneck, IEEE Trans. Pattern Anal. Mach. Intell. 36 (8) (2014) 1559–1572.
[15] O. Chapelle, B. Schölkopf, A. Zien, Semi-Supervised Learning, MIT Press, Cambridge, MA, 2006.
[16] M. Liu, Y. Luo, D. Tao, C. Xu, Y. Wen, Low-rank multi-view learning in matrix completion for multi-label image classification, in: Proceedings of the 29th American Association for Artificial Intelligence (AAAI) National Conference, 2015, pp. 2778-2784.
[17] K. Lu, J. Zhao, D. Cai, An algorithm for semi-supervised learning in image retrieval, Pattern Recognit. 39 (2006) 717–720.
[18] K. Nigam, A. McCallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using EM, Mach. Learn. 39 (2000) 103–134.
[19] M. Ester, H.P. Kriegel, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd ACM SIGKDD, Portland, Oregon, 1996, pp. 226–231.
[20] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: Proceedings of the 13th International Conference on Management of Data ACM SIGMOD, vol. 2, 1984, pp. 47–57.
[21] M. Ankerst, M. Breunig, H.P. Kriegel, J. Sander, OPTICS: ordering points to identify the clustering structure, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD),1999, pp. 49–60.
[22] X. Chen, W. Liu, H. Qiu, J. Lai, APSCAN: a parameter free clustering algorithm, Pattern Recognit. Lett. 32 (2011) 973–986.
[23] B.J. Frey, D. Dueck, Mixture modeling by affinity propagation, Proceedings of the 18th Neural Information Processing Systems Conference (2005) 379–386.
[24] A. Hinneburg, D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Proceedings of the 4th International Conference on KDD, 1998, pp. 58–65.
[25] P. Viswanath, R. Pinkesh, l-dbscan: A fast hybrid density based clustering method, in: Proceedings of the 18th International Conference on Pattern Recognition, vol. 1, IEEE Computer Society, Hong Kong, 2006, pp. 912–915.
[26] J.A. Hartigan, Clustering Algorithms, John Wiley & Sons, New York, 1975.
[27] P. Viswanath, V.S. Babu, Rough-DBSCAN: a fast hybrid density based clustering method for large data sets, Pattern Recognit. Lett. 30 (16) (2009) 1477–1488.
[28] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data, Kluwer Academic Publishing, Dordrecht, 1991.
[29] C. Böhm, R. Noll, C. Plant, B. W. Reuther, Density-based clustering using graphics processors, in: Proceedings of the Conference on Information and Knowledge Management (CIKM), Hong Kong, China, 2009, pp. 661–670.
[30] W.K. Loh, H. Yu, Fast density-based clustering through dataset partition using graphics processing units, J. Inf. Sci. 308 (2015) 94–112.
[31] M. Patwary, M.Ali, D. Palsetia, A. Agrawal, W.K. Liao, F. Manne, A. Choudary, A new scalable parallel DBSCAN algorithm using the disjoint-set data structure, in: Proceedings of the International Conference on HPC Networking, Storage and Analysis, 2012, pp. 1–11.
[32] M. Chen, X. Gao, H. Li, Parallel DBSCAN with priority r-tree, Proceedings of the Information Management Engineering (ICIME) (2010) 508–511.
[33] B.R. Dai, I.C. Lin, Efficient map–reduce based DBSCAN algorithm with optimized data partition, in: Proceedings of IEEE 5th International Conference on Cloud Computing (CLOUD), Hawaii, USA, 2012, pp. 59–66.
[34] C.T. Zahn, Graph-theoretical methods for detecting and describing Gestalt clusters, IEEE Trans. Comput. 20 (1) (1971) 68–86.
[35] G. Karypis, E.H. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, IEEE Trans. Comput. 32 (8) (1999) 68–75.
[36] O. Grygorash, Y. Zhou, Z. Jorgensen, Minimum spanning tree-based clustering algorithms, in: Proceedings of IEEE International Conference on Tools with Artificial Intelligence, 2006, pp. 73–81.
[37] C. Zhong, D. Miao, R. Wang, A graph-theoretical clustering method based on two rounds of minimum spanning trees, Pattern Recognit. 43 (2010) 752–766.
[38] M.G. Barrios, A.J. Quiroz, A clustering procedure based on the comparison between the k nearest neighbors graph and the minimal spanning tree, Stat. Probab. Lett. 62 (2003) 23–34.
[39] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 888–905.
[40] S. Bandyopadhyay, An automatic shape independent clustering technique, Pattern Recognit. 37 (2004) 33–45.
[41] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognit. 12 (1980) 261–268.
[42] S. Mimaroglu, E. Erdil, Combining multiple clusterings using similarity graph, Pattern Recognit. 44 (2011) 694–703.
[43] J. Yu, R. Hong, M. Wang, J. You, Image clustering based on sparse patch alignment framework, Pattern Recognit. 47 (2014) 3512–3519.
[44] J. Yu, Y. Rui, D. Tao, Click prediction for web image re-ranking using multimodal sparse coding, IEEE Trans. Image Process. 23 (5) (2014) 2019–2032.
[45] A. Frank, A. Asuncion, UCI machine learning repository, 2010. ⟨http://archive.ics.uci.edu/ml⟩.
[46] J.L. Bently, Multidimensional search trees in database applications, IEEE Trans. Softw. Eng. 5 (4) (1979) 333–340.
[47] T. Scott, M. Jeffrey, A. Mario, STR: A Simple and Efficient Algorithm for R-Tree Packing, Technical Report, Institute for Computer Application in Science and Engineering (ICASE), ACM Communications, 1997.
[48] P. Ciaccia, M. Patella, P. Zezula, M-tree: an efficient access method for similarity search in metric spaces, in: Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB),1997, pp. 426–435.
[49] H. Kaya, P. Tufekci , S.F. Gurgen, Local and global learning methods for predicting power of a combined gas & steam turbine, in: Proceedings of the International Conference on Emerging Trends in Computer and Electrical Engineering (ICETCEE), 2012, pp. 13–18.
[50] F. Esposito, D. Malerba, G. Semeraro, Multi strategy Learning for document recognition, Appl. Artif. Intell. 8 (1994) 33–84.
[51] C. Mallah, J. Cope, J. Orwell, Plant leaf classification using probabilistic integration of shape, texture and margin features, signal processing, Pattern Recognit. Appl. (2013) 45–54.
[52] W.M. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (1971) 846–850.

K. Mahesh Kumar completed his B.Tech and M.Tech in Computer Science and Engineering from JNT University, Hyderabad and SRM University, Chennai respectively. Currently, he is working for his Ph.D. in the Department of Computer Science and Engineering, SVU College of Engineering, Sri Venkateswara University, Tirupati, A.P, India. His areas of research interest include data mining, language processors and software architecture.

A. Rama Mohan Reddy was born in 1958, received his B.Tech degree from JNT University Anantapur in 1986, Masters in Computer Science and Engineering from NIT, Warangal in 1991 and Ph.D. in Computer Science and Engineering from Sri Venkateswara University, Tirupati in 2007. He is currently working as a Professor of Computer Science and Engineering, SV University College of Engineering, Tirupati, India. His research interests are Software Engineering, Software Architecture, Cloud Computing, Operating System and Data Mining. He is life member of ISTE, IETE and CSI. He has more than 30 years of experience in teaching, 7 scholars completed Ph.D.'s under his guidance. He has 50 publications and presented 46 papers in National and International conferences.