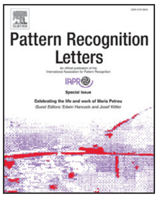
Pattern Recognition Letters 89 (2017) 1–7

Contents lists available at ScienceDirect

Pattern Recognition Letters

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



A highly scalable clustering scheme using boundary information

Qiuhui Tong, Xiu Li, Bo Yuan*

Intelligent Computing Lab, Division of Informatics, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, PR China

ARTICLE INFO

Article history:
Received 6 November 2015
Available online 26 January 2017

Keywords:
Clustering
DBSCAN
Cluster boundary
Density gradient

ABSTRACT

Many advanced clustering techniques are effective in dealing datasets in complicated situations. However, when facing large datasets, which are increasingly common in the era of big data, the time requirements of most existing techniques can quickly become intolerable. To tackle this challenge, in this paper, we propose Scalable Clustering Using Boundary Information (SCUBI), a highly flexible and scalable clustering scheme. The idea of SCUBI is to identify the boundary points of the original dataset in the first place and then group boundary points into suitable clusters using existing clustering techniques. Finally, the rest points are assigned to the same cluster as their nearest boundary points. To demonstrate the effectiveness and scalability of SCUBI, we plug the well-known DBSCAN algorithm into SCUBI. Comprehensive experiments are conducted using datasets with up to two million data points to compare the clustering results and time efficiency between DBSCAN and SCUBI-DBSCAN. Experimental results show that our method can obtain almost identical clustering results as the standard DBSCAN while achieving orders of magnitude speedup especially on large datasets, which confirms the scalability of SCUBI. Experiments are also performed on other clustering algorithms with high time complexity to verify the flexibility of SCUBI.

1. Introduction

Many clustering algorithms have already been proposed in the literature [1]. For example, K-means [2], one of the fundamental partitioning methods, has linear time complexity$O(knt)$, where k is the number of clusters, t is the iteration number and n is the size of the dataset. Although the time complexity of K-means is attractive, it is only suitable for compact and spherical clusters, and is sensitive to initial clustering center selection. DBSCAN (Density Based Spatial Clustering Applications with Noise) [3], a density based clustering technique, can discover arbitrarily shaped clusters with good robustness against noises, which is one of the most well-known clustering methods and received the Test of Time Award in KDD'14 [4]. Hierarchical algorithms are versatile such as single-linkage [5] or complete-linkage [6], but they suffer from high time complexity of $O(n^2)$ or even $O(n^3)$ and are sensitive to noises.

Although traditional clustering algorithms can often produce competent performance, users are now increasingly overwhelmed by the volume of data, which creates significant challenges to traditional clustering techniques. For example, DBSCAN features at least $O(n^{4/3})$ time complexity for dimensions higher than 3 [7], resulting in weak scalability. Instead of trying to directly reduce the time complexity of existing algorithms, we argue that it is equally promising to investigate the possibility of selecting and clustering a suitable set of prototypes from the original dataset.

The boundary points of a dataset can be adopted as such prototypes as they are located at the margin of densely distributed data, containing important characteristics of the distribution of data and can be used to decide whether a point belongs to a specific cluster or not [8].

A number of techniques for extracting boundary points have been proposed and most of them are used in classification tasks such as SVM [9] to reduce the training time. Note that, for classification tasks, the decision surface is mostly influenced by points in regions where the two classes overlap or close to each other while points in other regions may have little effect in practice. Although Li [10] claimed that the decision surface outside the overlapping region will be arbitrarily formed without the boundary points, the overlapping region still plays a more critical role in driving the classifier to form the expected decision surface. By contrast, we argue that boundary points are more attractive for clustering because they represent the distribution of the dataset, which is essential for clustering.

In order to improve the scalability of clustering, in this paper, we propose a clustering scheme exploiting boundary points: Scalable Clustering Using Boundary Information (SCUBI). The general idea of SCUBI is to extract boundary points from the original dataset as prototypes, which are subsequently grouped

* Corresponding author.
E-mail address: yuanb@sz.tsinghua.edu.cn (B. Yuan).

http://dx.doi.org/10.1016/j.patrec.2017.01.016
0167-8655/© 2017 Elsevier B.V. All rights reserved.

into clusters using existing clustering algorithms. The rest data points are then assigned to the same cluster as their nearest boundary points. There are only very few existing works on clustering based on boundary points [11,12] in which boundary points are only the by-products of the proposed new clustering algorithms. Meanwhile, most of these methods are limited to 2D datasets. Our proposed method is a generic clustering scheme that can help many existing clustering algorithms with high time complexity to reduce their running time with little extra efforts. It is highly flexible and can dramatically increase the capability of traditional clustering algorithms in handling large-scale datasets with dimensions larger than 2.

Many clustering methods with high time complexity can benefit from SCUBI. In this paper, we mainly use DBSCAN as the base clustering method due to its popularity, effectiveness as well as high time complexity ($O(n^2)$ in the worst case [13]). The main objective is to demonstrate the practicability of clustering based on boundary points and the scalability of SCUBI, by significantly improving the speed of DBSCAN with little negative effect on the clustering result. In the experiments, we also plug Affinity Propagation Clustering (APCLUSTER) [14] and Spectral Clustering (SC) into SCUBI to demonstrate the flexibility of the proposed scheme.

2. Boundary point extraction

2.1. Boundary points

Existing works on boundary point detection can be generally categorized into three themes: density-based [15–17], concave theory based [18,19] and graph based [11,12,20].

Concave theory based methods mainly look for the envelope of the dataset, which can be viewed as the boundary. They adopt the convex theory combined with the k-nearest neighbors (kNN) approach to figure out the boundary points one by one. These methods can extract a series of refined boundary points regardless of the distribution of the data. However, they are not robust against noises as there is a need to iteratively check whether all data points can be encompassed by the current boundary points found in previous steps. Furthermore, they are not easy to be extended to dimensions higher than 3 [18].

Graph based detection methods consider all data points in the dataset as a whole. Normally, a Delaunay diagram is built where each point corresponds to a node in the graph and each edge represents the distance between two points. The core idea is that the variation of distances between a boundary point and its neighbors is greater than that of interior points. However, building the Delaunay diagram for high dimensional datasets is time consuming. As a result, existing research on the boundary points based on Delaunay diagram is often limited to low dimensions (less than 3) [11,12].

In density-based boundary detection methods, a point is regarded as a boundary point if most of its neighbors lie on the same side of the tangent plane passing through it. The tangent plane can be viewed as a plane that divides the space into two separate regions: one with high density of points and the other is relatively sparse. Fig. 1 shows three examples with different curvature shapes where the red lines represent the desired tangent planes. In this paper, we choose this type of methods to extract boundary points, which will be detailed in the next section.

2.2. Density-based boundary point detection

Note that locating the tangent planes is difficult in practice. BORDER [8] takes advantage of the reverse kNN to avoid explicitly determining the plane. It assumes that the reverse kNN value of a boundary point is much smaller than that of interior points. The

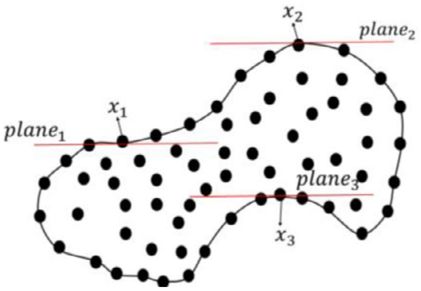


Fig. 1. Examples of boundary points. The red lines represent the desired tangent plane of three different boundary pointsx_1, $x_2$and x_3. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

definition of the reverse kNN is [21]:

$$rkNN(x_i) = \{x_j | \forall x_j, x_i \in kNN(x_j)\}$$

BORDER can deal with datasets with simple shapes and without noises effectively. However, when the interior of a cluster has lower density compared to the boundary region, it may incorrectly choose some interior points and miss out true boundary points. Also, the reverse kNN is an expensive query: the time complexity of the original reverse kNN is $O(n^3)$, although it can be reduced to $O(n^2 \log n)$.

Alternatively, the normal vector of the tangent plane instead of the plane itself can be calculated [15–17]. Considered the tangent plane represents the density divider, the normal vector of the tangent plane can also be viewed as the direction of density gradient. In [16], the attractor of a point x is defined as the neighbor of x sharing the highest number of ε neighbors and the vector pointing from x to its attractor is regarded as the normal vector. In [15,17], according to Theorem 1, the normal vector is defined as the vector pointing from x to the average position of the neighbors of x. The relevant proof is also presented to demonstrate the effectiveness of density gradient definition. Theorem 1 is similar to the concept in the Mean Shift clustering. However, we adopt this idea to identify the relative position of the points in the neighborhood rather than locate the position with the local maximum density in Mean Shift clustering.

Theorem 1. *The normal vector (density gradient) of an object x is as follows:*

$$NV(x) = \rho \cdot \frac{1}{k} \sum_{i=1}^{k} u_i$$

$$u_i(x) = x_i - x \tag{2.1}$$

where $x_i \in \mathrm{kNN}(x)$

Proof. Suppose a hypersphere with radius r centered at the object x is $\Gamma(x) = \{y | \mathrm{distance}(y, x) \le r\}$, which contains k nearest points and whose volume is v. The expected vector of y in $\Gamma(x)$ can be computed as follows [22]:

$$E\{(y-x)|\Gamma(x)\} \cong \int_{\Gamma(x)} (y-x)\frac{p(y)}{q_0}dy \tag{2.2}$$

where

$$q_0 = \int_{\Gamma(x)} p(y)dy \cong p(x)v \tag{2.3}$$

and $p(y)/q_0$ gives the conditional density function of y given $\Gamma(x)$.

$p(y)$ can be expanded around x by a Taylor series as follows:

$$p(y) \cong p(x) + (y-x)^T \nabla p(x) \tag{2.4}$$

$\nabla p(x)$ is the density gradient at the point x, which can be seen as the normal vector at the point x.

Substitute (2.3) and (2.4) into (2.2):

$$E\{(y-x)|\Gamma(x)\} \cong \int_{\Gamma(x)} (y-x)(y-x)^T \frac{1}{v} dy \frac{\nabla p(x)}{p(x)}$$
$$= \rho' \frac{\nabla p(x)}{p(x)} \tag{2.5}$$

where $\rho' = \int_{\Gamma(x)} (y-x)(y-x)^T \frac{1}{v} dy$.

Thus,

$$\nabla p(x) = \frac{1}{\rho'} \cdot E\{(y-x)|\Gamma(x)\} \cdot p(x) \tag{2.6}$$

The value of $E\{(y-x)|\Gamma(x)\}$for object x can be estimated by the mean of its k-nearest neighbors. So the normal vector of the density tangent planes is:

$$NV(x) = \nabla p(x) = \rho \cdot \frac{1}{k} \sum_{i=1}^{k} (x_i - x) \tag{2.7}$$

Since only the direction of the normal vector is concerned, the value of ρ is not important. Next, the relative location (the dense side or sparse side of the tangent plane) of the neighbor of the object x can be readily decided. Note that, for the boundary point, its normal vector must point to the dense side of the tangent plane. So, for neighbors located at the same subspace as the normal vector, the angles between each $u_i(x)$ and the normal vector should be within [0, $\pi/2$]. As a result, if for most $u_i(x)$ of an object x, the angles between $u_i(x)$ and the normal vector are within [0, $\pi/2$], x is considered as a boundary point. Various methods have been used in quantitative evaluation by counting the number of neighbors with angles between [0,$\pi/2$] with the normal vector [16,17] or adding up the cosine values of the angles between each u_i and the normal vector [15].

Based on the analysis of existing boundary detection techniques, we present a modified boundary extraction method (Algorithm 1). When searching for the k-nearest points of x, with small k values, the neighbors found are often concentrated in a highly localized area compared to large k values. Since the objective of boundary detection is to conduct clustering of the original dataset, the value of k should be varied according to the properties of datasets. If the inter-distance of clusters is large, k can be set to a large number while if the inter-distance of clusters is small, small k values are preferred.

In order to eliminate the impact of noises, the algorithm first calculates the average distance of x to its k-nearest neighbors. Intuitively, the neighbor points of a noise point are sparse and the average distance of a noise point to its k-nearest neighbors is likely to be larger than those of non-noise points. A percentage value of noise points α_{noise} or a threshold of the average distance can be used to eliminate noise points. In the paper, we adopt the percentage value α_{noise} to identify noises. After the elimination of noise points, the normal vector of each non-noise point in the dataset can be calculated according to Theorem 1. Since the normal vector can be biased by the extremely large magnitude of$u_i(x)$, the calculation of normal vector is modified as follows:

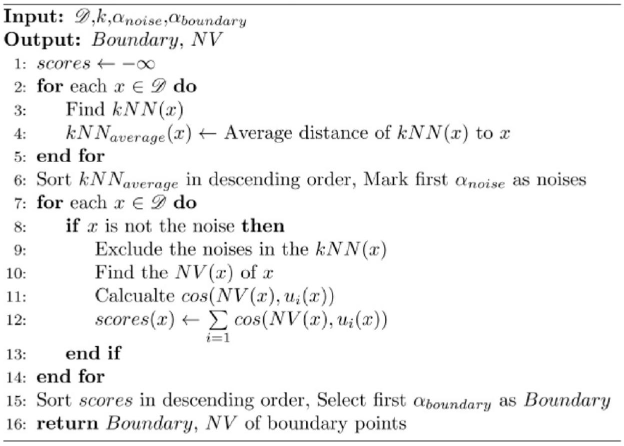
$$NV(x) = \sum_{i=1}^{k} \frac{(x_i - x)}{|x_i - x|} \tag{2.8}$$

With the normal vectors obtained, for each object x, the algorithm calculates the sum of the cosine values between $u_i(x)$ and $NV(x)$as its score:

$$scores(x) = \sum_{i=1} \cos(u_i(x), NV(x)) \tag{2.9}$$

Boundary points typically feature larger scores than others as the $\cos(u_i(x), NV(x))$ values for x_i located on the same side of the tangent plane as the normal vector are greater than zero while others are negative. Finally, points with top m scores are returned as boundary points along with their normal vectors.

Algorithm 1 Boundary extraction.



```
Input: $\mathscr{D}$,$k$,$\alpha_{noise}$,$\alpha_{boundary}$
Output: $Boundary$, $NV$
scores ← −∞
for each $x \in \mathscr{D}$ do
    Find $kNN(x)$
    $kNN_{average}(x)$ ← Average distance of $kNN(x)$ to $x$
end for
Sort $kNN_{average}$ in descending order, Mark first $\alpha_{noise}$ as noises
for each $x \in \mathscr{D}$ do
    if $x$ is not the noise then
        Exclude the noises in the $kNN(x)$
        Find the $NV(x)$ of $x$
        Calcualte $cos(NV(x), u_i(x))$
        $scores(x) \leftarrow \sum_{i=1} cos(NV(x), u_i(x))$
    end if
end for
Sort $scores$ in descending order, Select first $\alpha_{boundary}$ as $Boundary$
return $Boundary$, $NV$ of boundary points
```

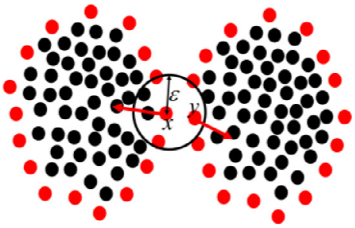


Fig. 2. The εneighborhood of object x contains object y, which actually belongs to a different cluster. However, the angle between the normal vectors of these two objects is greater than $\pi/2$.

3. Clustering

3.1. SCUBI

In this paper, we mainly focus on plugging DBSCAN into SCUBI to testify the efficiency and effectiveness of SCUBI. The target is to achieve similar clustering results as the standard DBSCAN but in a much more efficient manner. Once the boundary points are extracted using Algorithm 1, clustering is performed on boundary points only, instead of the whole dataset. As long as the boundary points are grouped into suitable clusters, the rest data points can be allocated to the same cluster as their nearest boundary points.

Since boundary points only account for a small portion of the whole dataset, the running time of DBSCAN can be significantly reduced, given its $O(n^2)$ time complexity in the worst situations. In Algorithm 2, *Boundary* and *NV*are the outputs from Algorithm 1, while ε and *Minpts* are the parameters of DBSCAN. Note that the definition of neighborhood given in Definition 1 is slightly different from traditional DBSCAN for better performance.

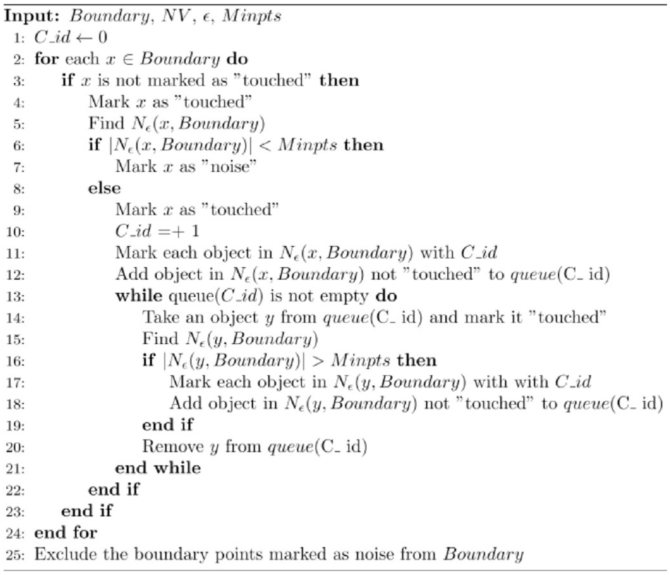
Definition 1. The neighborhood of object x w.r.t. ε and NV is defined as*:*

$$N_\varepsilon(x) = \{y|dist(x,y) \le \varepsilon, \cos(NV(x), NV(y)) \ge 0\}$$

where $dist(x,\ y)$ is the distance between object x and y and the distance function used in the paper is the Euclidean distance.

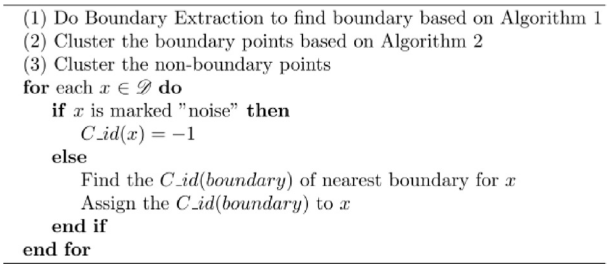
In some cases, two clusters may be close to each other, causing potential issues as demonstrated in Fig. 2. In Fig. 2, the boundary points are marked in red. The two red arrows represent the

Algorithm 2 DBSCAN on boundary points.



```
Input: Boundary, NV, $\epsilon$, Minpts
C_id ← 0
for each $x \in Boundary$ do
    if $x$ is not marked as "touched" then
        Mark $x$ as "touched"
        Find $N_\epsilon(x, Boundary)$
        if $|N_\epsilon(x, Boundary)| < Minpts$ then
            Mark $x$ as "noise"
        else
            Mark $x$ as "touched"
            $C\_id$ =+ 1
            Mark each object in $N_\epsilon(x, Boundary)$ with $C\_id$
            Add object in $N_\epsilon(x, Boundary)$ not "touched" to $queue$(C_ id)
            while queue($C\_id$) is not empty do
                Take an object $y$ from $queue$(C_ id) and mark it "touched"
                Find $N_\epsilon(y, Boundary)$
                if $|N_\epsilon(y, Boundary)| > Minpts$ then
                    Mark each object in $N_\epsilon(y, Boundary)$ with with $C\_id$
                    Add object in $N_\epsilon(y, Boundary)$ not "touched" to $queue$(C_ id)
                end if
                Remove $y$ from $queue$(C_ id)
            end while
        end if
    end if
end for
Exclude the boundary points marked as noise from Boundary
```

Algorithm 3 SCUBI-DBSCAN.



```
(1) Do Boundary Extraction to find boundary based on Algorithm 1
(2) Cluster the boundary points based on Algorithm 2
(3) Cluster the non-boundary points
for each $x \in \mathscr{D}$ do
    if $x$ is marked "noise" then
        $C\_id(x) = -1$
    else
        Find the $C\_id(boundary)$ of nearest boundary for $x$
        Assign the $C\_id(boundary)$ to $x$
    end if
end for
```

normal vectors of boundary points x and y respectively. Since y belongs to a different cluster but is within the ε neighborhood of x, they may be incorrectly grouped into the same cluster. The good news is that we can take advantage of the information of normal vectors to eliminate y from the ε neighborhood of x. Since the normal vector points to the area with high density, two nearby boundary points from different clusters typically have normal vectors pointing to reverse directions. Based on Definition 1, y will be excluded from the ε neighborhood of x, which reduces the possibility of incorrect clustering.

After all boundary points have been partitioned into suitable clusters, the rest objects in the dataset can be allocated to the same cluster as their closest boundary points, as shown in Algorithm 3 (SCUBI-DBSCAN).

3.2. Discussion

The time complexity of SCUBI-DBSCAN mainly consists of three parts: kNN query for boundary extraction, DBSCAN on boundary and 1NN for final partition. The time complexity of the original kNN query is $O(n^2)$ and many techniques such as kd-tree [23] have been developed to reduce the time complexity to $O(n\log n)$. As to DBSCAN, according to [7], the lower bound of its time complexity is $O(n^{4/3})$ while for the worst situation the time complexity can raise up to $O(n^2)$for high dimensional datasets. So an optimistic estimate of the time complexity of SCUBI-DBSCAN is $O(n\log(n) + m^{4/3})$. Since m is the number of boundary points extracted, which normally account for less than 10% of whole dataset, on larger-scale datasets, significant time reduction can be expected. Since in SCUBI-DBSCAN, clustering is an independent process separated from boundary extraction, many improved DBSCAN variations can be easily adopted to further improve the overall speed.

The possible factors that may influence the clustering results of SCUBI-DBSCAN are analyzed as follows:

(a) If the number of boundary points extracted is small, some boundary points belonging to the same cluster may be distant from each other. So, one cluster may be split into multiple small clusters by DBSCAN with small ε values. If there are two clusters located rather close to each other, it is also possible that some boundary points belonging to different clusters are closer to each other than those belonging to the same clusters. So, the two clusters may be merged to a single cluster by DBSCAN with large ε values.

(b) Once boundary points are partitioned into clusters, rest data points are assigned to the same cluster as their closest boundary points. Theoretically, boundary points are not dense enough to cover the entire surface of the cluster and there may be an interior point x whose closest boundary point is from another cluster, due to the large gaps among the boundary points of the cluster that x should belong to.

(c) In very high dimensional spaces, the Euclidean distance cannot reliably characterize the relative distances among data points. Since SCUBI relies on the kNN search, the boundary points detected may not be accurate for high dimensional datasets.

For the first issue, we adopt the normal vector to redefine the neighborhood of the boundary point in DBSCAN to minimize the influence of nearby boundary points from other clusters. For the second issue, in the experimental studies, we will show that it rarely happens in practice. For the third issue, compared to other existing methods that can only deal with datasets of 2D or 3D, our proposed method can deal with data in relatively high dimensionality (e.g. 10 dimensions) as we adopt the density of the dataset adopted rather than its geometric characteristics. The performance of other distance metrics for high dimensional spaces will be investigated in our future work.

Table 1
Summary of datasets.

Dataset	Number of features	Number of objects
Complex-shaped dataset	2	8000
Gaussian mixture dataset	8	Various
PAMAP2	4	1,942,872
Banana-shaped dataset	2	3800

4. Experiments

We conducted experiments on a Linux Workstation with dual Intel Xeon 2.0 GHz CPUs and 128 GB RAM. The two major features of our proposed clustering scheme are *scalability* and *flexibility*. Scalability refers to the capability of handling large datasets (e.g., with millions of records). So, in the first part of experiments, we used SCUBI-DBSCAN to testify the scalability of the proposed scheme. Flexibility means that our method can be combined with various clustering algorithms as necessary to reduce their practical running time, without the need to modify the algorithms themselves. So, in the second part of experiments, we directly plugged SC and APCLUSTER into SCUBI to testify the flexibility of SCUBI.

Experiments were conducted on both synthetic and real-world datasets (Table 1). The first three datasets were used to confirm the scalability of the proposed scheme while the banana-shaped dataset was adopted for testing the flexibility of SCUBI. These datasets were selected purposefully to evaluate the noise resiliency, scalability and efficiency of the proposed scheme, compared with the original clustering results. All kNN queries in our experiments were based on kd-tree.

In the first part of experiments, the DBSCAN code in use was developed by Michal Daszykowski written in Matlab [24]. The time efficiency was evaluated by the execution time with different data sizes. The effectiveness was measured using the similarity measure Adjusted Rand-Index(AR) [25]. The value of AR is within [0, 1] and a large value indicates that the two partitions are similar. The parameter setting of DBSCAN can influence both the clustering results and the execution time. To make the comparison as fair as possible, we used the same ε for both algorithms. As to the value of *Minpts*, smaller values were chosen for SCUBI-DBSCAN compared with DBSCAN due to the smaller number of boundary points to be extracted.

In the second part of experiments, the APCLUSTER Matlab code in use was from [26] while the SC Matlab code was provided by [27]. We compared the time consumption and clustering results between SCUBI and the original clustering algorithms. Note that, we used α to represent the percentage of points to be selected as the boundary points.

4.1. The Gaussian mixture dataset

The Gaussian mixture dataset consists of eight 8-dimensional datasets with various numbers of points to test the scalability of the proposed method. The probability distribution function used was $P(x) = 1/3 \cdot p_1(x) + 1/3 \cdot p_2(x) + 1/3 \cdot p_3(x)$where $p_i(x) = N(\mu_i, \sigma_i^2)$ for i=1, 2, 3 and the parameters of the three normal distributions were as follows:

$$\mu_1 = [0, 0, ..., 0]^T \; \mu_2 = [6, 6, ..., 6]^T$$
$$\mu_3 = [6, 6, 6, 6, -6, -6, -6, -6]^T$$
$$\sigma_1 = \mathrm{diag}(0.5, 0.5, 1, 1,..., 1) \; \sigma_2 = \mathrm{diag}(1, 1, 1, 0.5, 0.5, .1, 1, 1)$$
$$\sigma_3 = \mathrm{diag}(1, 1, 1, 1, 0.5, 0.5, 0.5, 0.5)$$

As to scalability, speedup curves for different values of α are plotted in Fig. 3. Note that the AR values were all equal to 1, indicating identical clustering between SCUBI-DBSCAN and original DBSCAN. This is due to the properties of the datasets as the three Gaussians were relatively distant from each other. It is clear that the speedup value consistently increased when the dataset became larger. In fact, SCUBI-DBSCAN was up to more than 80 times faster than the standard DBSCAN in our experiments for data size of 192,000, which implies significant time savings for large scale real-world data mining tasks. Also, the less the number of boundary points extracted, the less the processing time required by SCUBI-DBSCAN.

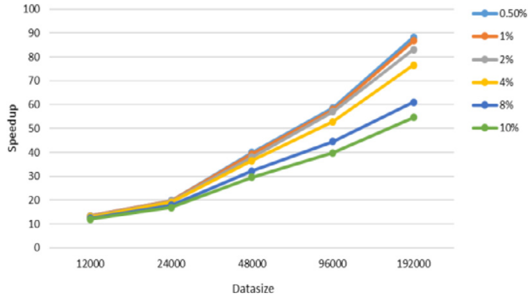


Fig. 3. The speedup curves with regard to the size of the Gaussian mixture datasets. It shows that SCUBI-DBSCAN is significantly more time efficient than the standard DBSCAN and features good scalability.

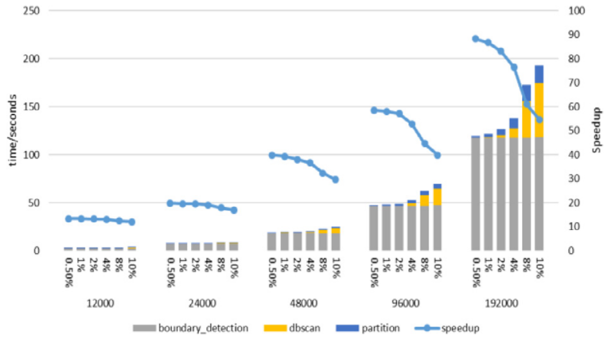


Fig. 4. The time consumption of each component in SCUBI-DBSCAN and the speedup curves with regard to different values of $\alpha$and different dataset sizes. It shows that, on larger datasets, the proportion for boundary detection drops while the proportion for DBSCAN on boundary increases due to the larger time complexity of DBSCAN compared to boundary detection.

Since the main tasks of SCUBI-DBSCAN can be divided into three components: boundary extraction, DBSCAN on boundary points and final partition, we also tracked the time consumed on these three parts respectively. The execution times of each part of the algorithm for varying sizes of datasets and speedup curves for varying values of α are presented in Fig. 4. According to Fig. 4, DBSCAN takes four times processing time when the data size is doubled, which indicates that the time complexity of DBSCAN is almost $O(n^2)$in this case. For boundary detection, the time complexity is nearly $O(n\log(n))$ for the dataset used. More specifically, the relative proportion of the boundary detection goes down with the increasing size of dataset while the relative proportion of DBSCAN on boundary points goes up. Meanwhile, on larger datasets, the speedup value drops more rapidly as α increases, compared to smaller datasets.

4.2. The complex-shaped dataset

The complex-shaped dataset has seven clusters located close to each other with significant noises. The seven clusters have various shapes and sizes, as shown in Fig. 5(a). The clustering result of the standard DBSCAN is given in Fig. 5(b).

We varied the percentage of boundary points α from 0.5% to 20% respectively. The comparison between the clustering results

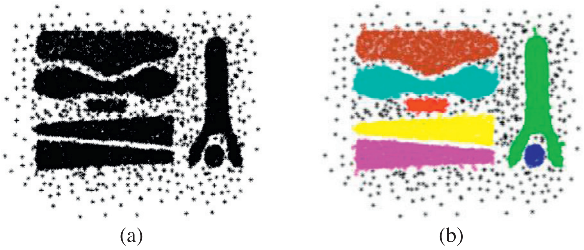


Fig. 5. The complex-shaped dataset. (a) the original dataset; (b) the clustering result based on the standard DBSCAN. The dataset is divided into seven distinct clusters.

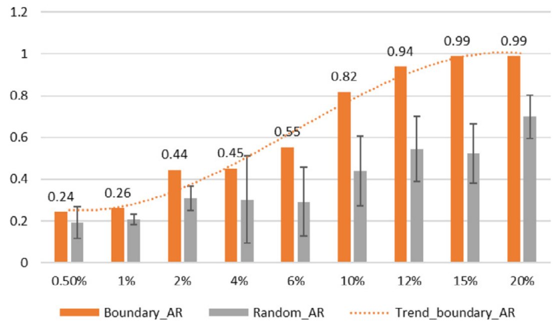


Fig. 6. The clustering result of SCUBI-DBSCAN and DBSCAN based on random sampling. Generally, the larger the value of α, the better the clustering result. Very high quality results can be expected with α greater than 10% while the performance of random sampling is clearly inferior. The standard deviation of clustering results of random sampling is also presented on the Random_AR bar.

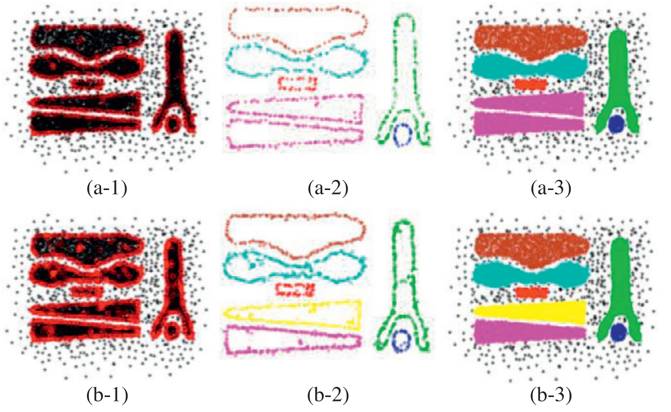


Fig. 7. The clustering results of SCUBI-DBSCAN on the complex-shaped dataset with different α values. (a) $\alpha = 10\%$; and (b) $\alpha = 20\%$. The left column shows the boundary points extracted; the middle column shows the clustering results of boundary points; the right column gives the final clustering results.

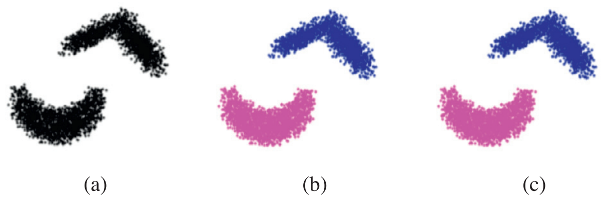


Fig. 8. The banana-shaped dataset. (a) the original dataset; (b) the clustering result of the standard Spectral clustering; (c) the clustering result of the standard Affinity Propagation clustering.

of SCUBI and the clustering results based on random sampling are presented in Fig. 6.

It is clear that once more than 10% points were selected as boundary points for this dataset, the AR values were close to 1, indicating almost identical clustering results between SCUBI-DBSCAN and the original DBSCAN. The proper value of $\alpha$for good clustering results depends on the properties of the dataset. If the distribution of the data is complicated, a large number of data points are needed to specify the boundary of clusters. If the data size is relative small, a small portion of the original dataset is not sufficient either due to random factors.

As to random sampling, the data points selected will vary a lot, which will influence the stability of clustering results. As shown in Fig. 6, on complex datasets, its performance is generally unacceptable, even with a large number of points sampled. For example, for small clusters, it is possible that few if any points are sampled from them and they may be completed missed out.

In Fig. 7, boundary points extracted are shown in the left column with red circles while the clustering results of boundary points and the whole datasets are shown in the middle column and right column, respectively, with different α values (10% and 20%). It is clear that SCUBI-DBSCAN achieved almost identical results as the original DBSCAN. Although the boundary points extracted were not always accurate (e.g., some interior points were misclassified as boundary points), it had not significant influence on the final clustering results as such interior points were largely excluded by DBSCAN.

4.3. PAMAP2 dataset

PAMAP2 is a real-world dataset from the UCI machine learning archive [28] representing the physical activity monitoring of 9 different people. The original dataset contains 3,850,505 instances with 54 dimensions. By applying principle component analysis and other pre-processing procedures, the remaining dataset had 1,942,872 instances with 4 dimensions. In our experiments, the execution time of DBSCAN was intolerable and had to be terminated after running for more than 12 h. By contrast, when using 5% data points as the boundary points, the execution time of SCUBI-DBSCAN was 987.78 s. The execution time was further reduced to 537.68 s when only 2% data points were selected.

4.4. The banana-shaped dataset

The high scalability of SCUBI has already been confirmed in the above experiments. Since SCUBI is a general clustering scheme, other existing clustering algorithms with high time complexity can be conveniently plugged into SCUBI to reduce their processing times on large datasets. SC and APCLUSTER are two famous clustering algorithms with good performance while both suffer from high time complexity: the time complexity of SC is $O(n^3)$ while the time complexity of APCLUSTER is $O(n^2t)$ where t is the number of iterations. We applied these two clustering algorithms directly into SCUBI without any modification to evaluate the time efficiency and clustering results on the banana-shaped dataset to demonstrate the flexibility of SCUBI.

The banana dataset (3800 points, 2 dimensions) consisting of two banana shaped clusters is shown in Fig. 8(a) while the clustering results based on SC and APCLUSTER are shown in Fig. 8(b) and (c), respectively. The shapes of the two clusters are simple and the inter-cluster distance is large.

We set α to 1%, 4%, 8% and 10%, respectively. SCUBI-SC and SCUBI-APCLUSTER achieved the same clustering results as the original clustering algorithms, and the values of AR were equal to 1 in all the experiments. Table 2 shows the execution time in seconds. As to efficiency, the execution time of the original APCLUSTER was quite long, leading to a large speedup value for SCUBI-APCLUSTER.

Table 2
Experiment results on the banana-shaped dataset.

α	Execution time (s)			
	SCUBI-SC	SC	SCUBI-APCLUSTER	APCLUSTER
1%	0.52	3.91	0.55	78.24
4%	0.54		0.96	
8%	0.55		1.60	
10%	0.56		1.92	

5. Conclusions

In this paper, we presented SCUBI, a highly scalable and flexible clustering scheme, to accelerate traditional clustering algorithms in the face of large-scale datasets. The main idea is to perform standard clustering on selected boundary points only, instead of the entire dataset. The rest data points are then assigned to the same cluster as their nearest boundary points. By doing so, without the need to directly tackle the challenge of reducing the time complexity of existing clustering algorithms, their practical running time can be dramatically reduced.

In the case studies, we chose DBSCAN as the base clustering algorithm in the hope to overcome its high time complexity issue while keeping its effectiveness in handling complex clusters. Extensive experimental studies on both synthetic and real-world datasets confirmed that the proposed clustering scheme can handle a variety of datasets much more efficiently than the original DBSCAN, especially on large-scale datasets while achieving extremely similar clustering results. We also directly applied APCLUSTER and SC to SCUBI, which both achieved the same clustering results as the original algorithms with significant savings on the processing time.

The proposed clustering scheme SCUBI is highly scalable to handle datasets with millions of data points on which most traditional clustering algorithms may need prohibitive amount of execution time. Meanwhile, it is flexible to be combined with various advanced clustering methods as required. As to future work, it is important to further reduce the time complexity of boundary point selection and extend SCUBI to higher dimensional problems.

References

[1] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.

[2] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.

[3] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[4] Available: http://kdd.org/News/view/2014-sigkdd-test-of-time-award.

[5] P.H. Sneath, R.R. Sokal, Numerical taxonomy: the principles and practice of numerical classification, W.H. Freeman and Company, San Francisco, 1973.

[6] B. King, Step-wise clustering procedures, J. Am. Stat. Assoc. 62 (317) (1967) 86–101.

[7] J. Gan, Y. Tao, DBSCAN Revisited, Mis-claim, un-fixability, and approximation, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 519–530.

[8] C. Xia, W. Hsu, M.L. Lee, B.C. Ooi, BORDER, Efficient computation of boundary points, IEEE Trans. Knowl. Data Eng. 18 (3) (2006) 89–303.

[9] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[10] Y. Li, M. Liam, Selecting critical patterns based on local geometrical and statistical information, IEEE Trans. Pattern Anal. Mach. Intell. 33 (6) (2011) 1189–1201.

[11] D. Liu, G.V. Nosovskiy, O. Sourina, Effective clustering and boundary detection algorithm based on Delaunay triangulation, Pattern Recognit. Lett. 29 (2008) 1261–1273.

[12] V. Estivill-Castro, I. Lee, Autoclust, Automatic clustering via boundary extraction for mining massive point-data sets, in: Proceedings of the 5th International Conference on Geocomputation, 2000.

[13] A. Gunawan, M. de Berg, A Faster Algorithm For DBSCAN, Techische Universiteit Eindhoven, Netherland, 2013 (Master Thesis).

[14] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.

[15] F. Zhu, N. Ye, W. Yu, S. Xu, G. Li, Boundary detection and sample reduction for one-class support vector machines, Neurocomputing 123 (2014) 166–173.

[16] B.-Z. Qiu, F. Yue, J.-Y. Shen, BRIM: an efficient boundary points detecting algorithm, in: The Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2007, pp. 761–768.

[17] Y. Li, Selecting training points for one-class support vector machines, Pattern Recognit. Lett. 32 (11) (2011) 1517–1522.

[18] A. López Chau, X. Li, W. Yu, J. Cervantes, P. Mejía-Álvarez, Border samples detection for data mining applications using non convex hulls, in: 10th Mexican International Conference on Artificial Intelligence, 2011, pp. 261–272.

[19] A. Moreira, M.Y. Santos, Concave hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points, in: The 2007 International Conference on Computer Graphics Theory and Applications, 2007, pp. 61–68.

[20] J. Yang, V. Estivill-Castro, S.K. Chalup, Support vector clustering through proximity graph modelling, in: Proceedings of the 9th International Conference on Neural Information, 2002, pp. 898–903.

[21] F. Korn, S. Muthukrishnan, Influence sets based on reverse nearest neighbor queries, SIGMOD Rec. 29 (2) (2000) 201–212.

[22] K. Fukunaga, L. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, IEEE Trans. Inf. Theory 21 (1) (1975) 32–40.

[23] R. Sproull, Refinements to nearest-neighbor searching in k-dimensional trees, Algorithmica 6 (1-6) (1991) 579–589.

[24] Available: http://www.chemometria.us.edu.pl/download/DBSCAN.M.

[25] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.

[26] Available: http://www.psi.toronto.edu/affinitypropagation/software/apcluster.m.

[27] W.Y. Chen, Y. Song, H. Bai, C.J. Lin, E.Y. Chang, Parallel spectral clustering in distributed systems, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2011) 568–586.

[28] A. Reiss, D. Stricker, Introducing a New Benchmarked Dataset for Activity Monitoring, in: The 16th IEEE International Symposium on Wearable Computers, 2012.