A Fast Clustering Algorithm based on pruning unnecessary distance computations in DBSCAN for High-Dimensional Data

Yewang Chen, Shenyu Tang, Nizar Bouguila, Cheng Wang, Jixiang Du, HaiLin Li

PII: S0031-3203(18)30210-3
DOI: 10.1016/j.patcog.2018.05.030
Reference: PR 6574

To appear in: *Pattern Recognition*

Received date: 28 April 2017
Revised date: 1 February 2018
Accepted date: 31 May 2018

Please cite this article as: Yewang Chen, Shenyu Tang, Nizar Bouguila, Cheng Wang, Jixiang Du, HaiLin Li, A Fast Clustering Algorithm based on pruning unnecessary distance computations in DBSCAN for High-Dimensional Data, *Pattern Recognition* (2018), doi: 10.1016/j.patcog.2018.05.030

A Fast Clustering Algorithm based on pruning unnecessary distance computations in DBSCAN for High-Dimensional Data

Yewang Chen[a,b,*], Shenyu Tang[a], Nizar Bouguila[b,*], Cheng Wang[a], Jixiang Du[a], HaiLin Li[a]

Yewang Chen 39, Xiamen

The College of Computer Science and Technology of Huaqiao University[a], Concordia Institute for Information Systems Engineering[b]

[a]Jimei Avenue 668, Xiamen,Fujian province, China
[b]1515 St.Catherine Street West, EV.007.632, H3G 2W1, Montreal, Canada

Abstract

Clustering is an important technique to deal with large scale data which are explosively created in internet. Most data are high-dimensional with a lot of noise, which brings great challenges to retrieval, classification and understanding. No current existing approach is "optimal" for large scale data. For example, DBSCAN requires $O(n^2)$ time, Fast-DBSCAN only works well in 2 dimensions, and ρ-Approximate DBSCAN runs in $O(n)$ expected time which needs dimension D to be a relative small constant for the linear running time to hold. However, we prove theoretically and experimentally that ρ-Approximate DBSCAN degenerates to an $O(n^2)$ algorithm in very high dimension such that $2^D >> n$. In this paper, we propose a novel local neighborhood searching technique, and apply it to improve DBSCAN, named as NQ-DBSCAN, such that a large number of unnecessary distance computations can be effectively reduced. Theoretical analysis and experimental results show that NQ-DBSCAN averagely runs in $O(n * log(n))$ with the help of indexing technique, and the best case is $O(n)$ if

*Corresponding author
Email addresses: ywchen@hqu.edu.cn;shengyutang@hqu.edu.cn (The College of Computer Science and Technology of Huaqiao University), nizar.bouguila@concordia.ca (Concordia Institute for Information Systems Engineering)

proper parameters are used, which makes it suitable for many realtime data.

Keywords: DBSCAN, ρ-Approximate DBSCAN, NQ-DBSCAN

2010 MSC: 00-01, 99-00

1. Introduction

Nowadays, large collections of data are explosively created in different fields, and most of these data are high dimensional with a lot of noise, e.g Web Texts and Web videos, some of them have more than 10,000 dimensions, which brings great challenges to retrieval, classification and understanding. Many researches are launched in this area to deal with this kind of data [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13].

Data clustering is one of the most important and popular data analysis techniques to understand data. It refers to the process of grouping objects into meaningful subclasses (clusters) so that members of a cluster are as similar as possible whereas members of different clusters differ as much as possible [14, 15, 16]. Numerous clustering algorithms have been used in many areas such as image processing [17, 18, 19], geophysics [20, 21], customer and marketing analysis [22, 23], crime detection [24], medicine [25, 26] and agriculture [27]. Innovative clustering methods [28, 29, 30] and parallel implementation frameworks [31, 32] have been proposed.

Clustering algorithms can be roughly categorized into partition, hierarchical, grid-based and density-based approaches etc. Density-based clustering approach is one of the most popular paradigms, and the most famous algorithm of this kind is DBSCAN [33] which is designed to discover clusters of arbitrary shape with a fixed scanning radius ϵ (*eps*) and a density threshold $MinPts$. DBSCAN has a large amount of extensions, e.g. [34, 35, 36, 37], and has been widely applied in many applications, such as astronomy [38], neuroscience [39]. However, DBSCAN has some drawbacks as follows.

(1) It renders almost useless when subject to high-dimensional data due to the so-called "Curse of dimensionality".

(2) The running time for DBSCAN is heavily dominated by finding neighbors or obtaining density for each data point. Without indexing, the complexity of DBSCAN would always be $O(n^2)$ regardless of the parameters ϵ and *MinPts*. If a tree-based spatial index is used, the ϵ-neighborhood are expected to be small compared to the size of the whole data space, the average complexity is reduced to $O(n * log(n))$ [33]. However, for dimension $d > 3$ the DBSCAN problem require $\Omega(N^{4/3})$ time to solve, unless very significant breakthroughs could made in theoretical computer science [40].

Many researchers have proposed various techniques in attempts to improve the performance of clustering algorithm on high-dimensional data. For example, Wang and Deng developed a serial of important work on soft subspace clustering and fuzzy clustering for high dimensional data [41, 42, 43, 44], which overcome the drawbacks of utilizing only one distance function in most of existing clustering algorithms, and adaptively learn the distance functions suitable for data sets during the clustering process.

Grid-based technique and approximation techniques are also popular, such as Fast-DBSCAN [45] and others [46, 47]. Grid-based techniques, e.g. [48, 49, 50, 51], divide the data space by grids, perform clustering in each cell locally and merge the results thereby saving runtime. Gunawan [45] proposed a Fast-DBSCAN based on drawing a 2-dimensional grid. The algorithm imposes an arbitrary grid T on the data space $\mathbb{R}^2$, where each cell of T has side length $\sqrt{\epsilon/2}$. If a non-empty cell c contains at least $MinPts$ points, then all those points in the cell must be core points, because the maximum distance within the cell is ϵ. This algorithm theoretically runs in $O(n*log(n))$ time in the worst case. However it is only applied in 2-dimensional data space.

Inspired by Fast-DBSCAN, Gan and Tao [40] proposed a novel algorithm named ρ-approximate DBSCAN, which has a computation time that scales only linearly in n. The improvement of this method from Gunawan [45] lies in its new tree structure, i.e. quadtree-like hierarchical grid, as well as the sacrifice of small accuracy. Because the cell number in the quadtree-like hierarchical grid T will increase explosively with dimension D, therefore ρ-approximate only saves

those non-empty cells. However, it needs dimension D to be a relative small constant for the linear running time to hold, and actually it still runs in $O(n^2)$ in high dimension, as the following theorem shows.

Theorem 1. *ρ-approximate DBSCAN degenerates to an $O(n^2)$ algorithm if $2^D \gg n$.*

Proof. Let X be the maximum radius for DBSCAN to correctly cluster data set P, and dimension D be large enough such that $2^D \gg n$, which implies there are much more cells than n in the grid. Set $\epsilon = X$, for each cell there is at most one point contained if D is large enough, because the side length of each cell is $\frac{X}{\sqrt{D}}$ and $\lim_{D\to\infty} \frac{X}{\sqrt{D}} = 0$.

In the case of $2^D \ll n$, ρ-approximate DBSCAN answers any approximate range count query in $O(1)$ expected time (see Lemma 5 in [40]). But here, since each non-empty cell contains at most one point, then there are about n nonempty cells are saved. Thus the query time for each cell to find neighbors is $O(n)$, not $O(1)$ any more, and hence ρ-approximate DBSCAN runs in $O(n^2)$ expected time. □

Therefore, most existing current clustering algorithms are not suitable for many realtime applications, due to the "curse of dimensionality". The main reason lies in great number of unnecessary distance calculations, which can be greatly reduced by neighbor searching technique, such as Product quantization for nearest neighbor search [52], LSH (Locality-Sensitive Hashing) [53], FLANN [54].

In this paper, we propose a new clustering approach, named NQ-DBSCAN, by using local neighbor query technique and quadtree-like hierarchical grid to reduce great number of unnecessary distance computations. Theoretical analysis and experimental results show that the proposed algorithm NQ-DBSCAN can averagely run in $O(n*log(n))$ expected time with the help of indexing technique, and the best case is $O(n)$ if proper parameters are used, which makes it suitable for many realtime data.

Because ρ-Approximate DBSCAN is the most important improvement of DBSCAN currently, we only focus on DBSCAN, ρ-Approximate DBSCAN and NQ-DBSCAN in this paper. There are some advantages of NQ-DBSCAN to ρ-Approximate DBSCAN as below.

(1) NQ-DBSCAN is an exact algorithm that may return the same result as DBSCAN if the parameters are same. While ρ-Approximate DBSCAN is an approximate algorithm.

(2) The best complexity of NQ-DBSCAN can be $O(n)$, and the average complexity of NQ-DBSCAN is proved to be $O(nlog(n))$ provided the parameters are properly chosen. While ρ-Approximate DBSCAN runs only in $O(n^2)$ in high dimension.

(3) NQ-DBSCAN is suitable for clustering data with a lot of noise.

The rest of this paper is organized as follow: Section 2 introduces the basic concepts; Section 3 presents the details of the proposed clustering algorithm; Section 4 demonstrates the experimental results of the proposed algorithms on various data sets, and Section 5 gives the conclusion and our future works.

2. The Basic Concepts of DBSCAN and Preliminary Notation

2.1. Basic Concepts

Density-based clustering algorithms have the ability to find out the clusters of different shapes and sizes. DBSCAN, a pioneer density-based clustering algorithms, is one of the most important and popular clustering algorithms in scientific literature[1]. DBSCAN accepts two parameters: ϵ (*Eps*) and *MinPts*, where ϵ is scanning radius and *MinPts* is the minimal number of neighbor points for a core point. Some concepts and terms to explain the DBSCAN algorithm can be defined as follows [33].

Definition 1. *The* ϵ***-neighborhood*** *of a point p, denoted by* $N_\epsilon(p)$*, is defined by* $N_\epsilon(p)=\{q|q \in P, d_{p,q} \leq \epsilon\}$*, where P is a set of points and* $d_{p,q}$ *is a distance*

[1] https://en.wikipedia.org/wiki/DBSCAN

function e.g. Euclidian distance, between p and q.

Definition 2. *A point p is a* ***core point*** *if* $\mid N_\epsilon(p) \mid \geq MinPts$.

Definition 3. *A point p is* ***directly density-reachable*** *from a point q with respect to* ϵ *and* $MinPts$ *if* $p \in N_\epsilon(q)$ *and q is a core point.*

Definition 4. *A point p is a* ***border point*** *if p is directly density-reachable from a core point q and* $\mid N_\epsilon(p) \mid < MinPts$.

Definition 5. *A point p is* ***density-reachable*** *from a point q with respect to* ϵ *and* $MinPts$ *if there is a chain of points* p_1,p_2,...,p_n, *with* $p_1 = q$ *and* $p_n = p$ *such that* p_{i+1} *is directly density-reachable from* p_i.

Definition 6. *A point p is* ***density-connected*** *to a point q with respect to* ϵ *and* $MinPts$ *if there is a point o such that both p and q are density-reachable from o.*

Definition 7. *Let p be a set of points. A* ***cluster*** *C with respect to* ϵ *and* $MinPts$ *is a non-empty subset of p satisfying the following conditions:*

1. $\forall$ *p, q: if* $p \in C$ *and q is dendity-reachable from p with respect to* ϵ *and* $MinPts$, *then* $q \in C$ *(Maximality).*
2. $\forall$ *p, q* $\in$ *C: p is density-connected to q with respect to* ϵ *and* $MinPts$ *(Connectivity).*

Definition 8. *A point p is a* ***noise*** *if it is neither a core point nor a border point. This implies that noise does not belong to any clusters.*

2.2. Algorithm

First, DBSCAN selects a point p randomly and retrieves all points in its ϵ-neighborhood. If the density of p is larger than $MinPts - 1$, i.e. $|N_\epsilon(p)| \geq MinPts|$, p will be marked as a new cluster. Then this cluster is expanded by retrieving all points that are *density-reachable* from p as Algorithm 2 shows, and then these points are merged into the same cluster. Repeat this process until no cluster found. If the density of p is less than *MinPts*, p will be marked as a noise.

Also, p might be assigned into other cluster provided p is a *density-reachable* point from a core point q. The key of DBSCAN is shown in Algorithm 1 and Algorithm 2. The function *RangeQuery(p,ϵ)* returns all neighbors within the ϵ-neighborhood of p.

Algorithm 1 DBSCAN(P,ϵ,*MinPts*) [45]

```
Initialize cluster id C = 0
for each unclassified point p ∈ P do
    N_ϵ(p)=RangeQuery(p,ϵ)
    if |N_ϵ(p)| ≥ MinPts then
        Set p's cluster id to C
        ExpandCluster(p,N_ϵ(p),C,ϵ,MinPts)
        C ← C + 1
    else
        Label p as noise
    end if
end for
```

It is not surprising since the running time for DBSCAN is heavily dominated by the running time of the *RangeQuery(p,ϵ)* which must be performed for each point. Obviously, without any indexing support, the complexity of DBSCAN would always be $O(n^2)$ regardless of the parameters ϵ and *MinPts*.

3. The proposed Algorithm: NQ-DBSCAN

3.1. Basic Concepts

We propose a new algorithm to improve DBSCAN by filtering a large number of unnecessary density computations, which is based on the following idea.

Point p and point q should have similar neighbors, provided p and q are close; given a certain ϵ, the closer they are, the more similar their neighbors are. As Fig. 1 shows, we can see that points p and q in Fig. 1 (a) have more same neighbors than that they have in Fig. 1 (b). Formally, we have some theorems which are important for validating the correctness of our clustering algorithm, as follows.

Algorithm 2 ExpandCluster(p,$neighborPts$,C,ϵ,$MinPts$)[45]

Input:

p: current search point;

$neighborPts$: density-reachable points from p;

C: current cluster id;

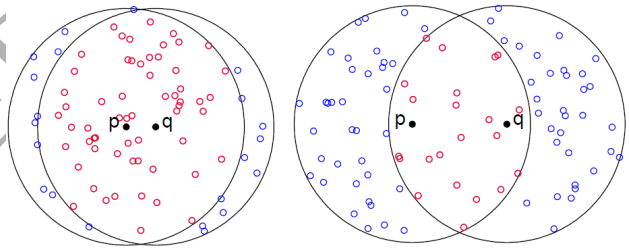
ϵ: the maximum distance;

$MinPts$: the minimum points to form a cluster;

Output:

$drPts$ (density-reachable points from p);

1: $drPts \leftarrow neighborPts$
2: **for** each point $q \in drPts$ **do**
3: **if** q is unclassified **then**
4: $N_\epsilon(p)$=$RangeQuery(p,\epsilon)$
5: **if** $|N_\epsilon(p)| \geq MinPts$ **then**
6: $drPts$=$drPts \cup N_\epsilon$(p)
7: **end if**
8: **end if**
9: **if** q does not belong to any cluster **then**
10: q's cluster id= C
11: **end if**
12: **end for**



(a) (b)

Figure 1: p and q in (a) have more same neighbors than that case in (b), because p and q in (a) are closer.

Firstly, we make some notations. Let $p \in P$, $d_{p,(1)} \leq d_{p,(2)} \leq ... \leq d_{p,(N)}$ be an ordered distance sequence of point p to all point. We also use $p_{(i)}$ to denote the i^{th} closest point from p. For example, there are 5 points a, b, c, d and p, if $d_{p,a} < d_{p,b} < d_{p,c} < d_{p,d}$, then $p_{(1)} = a$, $p_{(2)} = b$, $p_{(3)} = c$, $p_{(4)} = d$.

Theorem 2. *(1) If $d_{p,(MinPts)} \leq \epsilon$, then p is a core point. (2) p is a non-core point if $d_{p,(i)} > \epsilon$, where $1 \leq i \leq MinPts$.*

Proof. (1) $\because d_{p,(MinPts)} \leq \epsilon$, which means $d_{p,(1)} \leq d_{p,(2)} \leq ... \leq d_{p,(MinPts)} \leq \epsilon$, $\therefore |N_\epsilon(p)| \geq MinPts$, thus p is a core point.
(2) $\because 1 \leq i \leq MinPts$ and $d_{p,(i)} > \epsilon$, $\therefore \epsilon < d_{p,(i)} \leq d_{p,(MinPts)}$, thus $|N_\epsilon(p)| < MinPts$, i.e p is a non-core point.

□

Theorem 3. *Let $p \in P$, if $|N_{2\epsilon}(p)| < MinPts$, then $\forall q \in N_\epsilon(p)$ is non-core point.*

Proof. $\because N_\epsilon(q) \subseteq N_{2\epsilon}(p)$ and $|N_{2\epsilon}(p)| < MinPts$, $\therefore$ we have $|N_\epsilon(q)| < |N_{2\epsilon}(p)| < MinPts$, then $\forall q \in N_\epsilon(p)$ is non-core point. □

This theorem tells us a fact that if $|N_{2\epsilon}(p)| < MinPts$, then all points within the ϵ-neighborhood of p are non-core points.

Theorem 4. *Let $p \in P$, and $d_{p,(MinPts)}$=l, if $l > \epsilon$ then $\forall$ $o \in O$, o is a non-core point, where $O = \{o | d_{o,p} < l - \epsilon\}$.*

Proof. $\because d_{p,(MinPts)} = l \therefore | N_l(p) | = MinPts$. $\because d_{o,p} < l - \epsilon \therefore$ $d_{o,p} + \epsilon < l$ then $N_\epsilon(o) \subset N_l(p)$, thus we have $|N_\epsilon(o)| < |N_l(p)| = MinPts$. $\therefore \forall o \in O$ is non-core point. □

As Fig. 2 shows, $l > \epsilon$, the total number of points within the outer black circle is less than $MinPts$, and $d_{o,p} < l - \epsilon$, according to Theorem 4, all points in $N_{l-\epsilon}(p)$ are non-core points, as the red points within red circle show.

Theorem 5. *Let $p, q, m \in P$. If $d_{p,m} < \epsilon - d_{p,q}$, then $m \in N_\epsilon(q)$.*

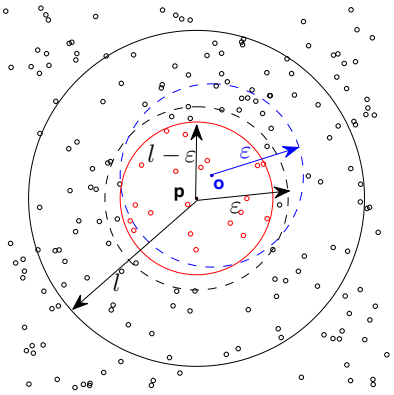


Figure 2: p is a non-core point, according to Theorem 4, all points in $N_{l-\epsilon}(p)$ are non-core points, as the red points show.

Proof. $\because d_{p,m} < \epsilon - d_{p,q}, \therefore d_{p,q} + d_{p,m} < \epsilon$, then according to Triangle Inequality, we have $d_{q,m} < d_{p,q} + d_{p,m}$, thus $d_{q,m} < \epsilon$, therefore $m \in N_\epsilon(q)$. □

In Fig. 3, $\forall\ m_1$ contained in blue circle, m_1 satisfies $d_{p,m_1} < \epsilon - d_{p,q}$, according to Theorem 5, $m_1 \in N_\epsilon(q)$. Thus blue points are all contained in $N_\epsilon(q)$.

Theorem 6. *Let $p, q, m \in P$, if $d_{p,m} > \epsilon + d_{p,q}$, then $m \notin N_\epsilon(q)$.*

Proof. $\because d_{p,m} > \epsilon + d_{p,q}, \therefore d_{p,m} - d_{p,q} > \epsilon$. Then according to Triangle Inequality, we have $d_{q,m} > d_{p,m} - d_{p,q}$, thus $d_{q,m} > \epsilon$. $\therefore$ point m $\notin N_\epsilon(q)$. □

In Fig. 3, $\forall\ m_2$ outside the red circle, we have $d_{p,m_2} > \epsilon + d_{p,q}$, according to Theorem 6, $m_2 \notin N_\epsilon(q)$. Thus the black points are all not included in $N_\epsilon(q)$.

Theorem 7. *Let $p, q \in P$, and $N_{2\epsilon}(p)$ is already obtained, in order to get $N_\epsilon(q)$, the searching range is $p_{(L)}, p_{(L+1)}, ..., p_{(U-1)}, p_{(U)}$, where L, U satisfy $d_{p,p_{(L-1)}} < \epsilon - d_{p,q} < d_{p,p_{(L)}}$ and $d_{p,p_{(U)}} < \epsilon + d_{p,q} < d_{p,p_{(U+1)}}$.*

Proof. $\because d_{p,p_{(L-1)}} < \epsilon - d_{p,q}$, then according to Theorem 5, $p_{(1)}, p_{(2)}, ..., p_{(L-1)}$ are contained in $N_\epsilon(q)$. $\because \epsilon + d_{p,q} < d_{p,p_{(U+1)}}$, and then according to Theorem 6,

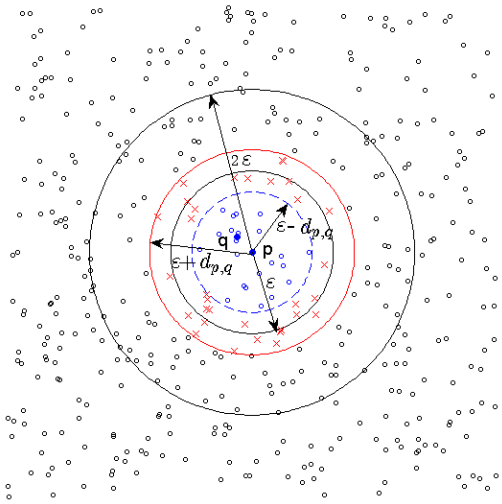


Figure 3: Illustration of Theorem 5, Theorem 6 and Theorem 7. All points in $N_{\epsilon-d_{p,q}}(p)$ (blue points) are in N_ϵ(q), and black points are all outside the ϵ-neighborhood of q, only red points are uncertain.

we have $p_{(U+1)}, p_{(U+2)}, ..., p_{(N)}$ are not contained in $N_\epsilon(q)$. $\therefore p_{(L)}, p_{(L+1)}, ..., p_{(U-1)}, p_{(U)}$ is the searching range for obtaining $N_\epsilon(q)$. □

According to Theorem 7, in Fig. 3 the remaining uncertain points $(p_{(L)}, ..., p_{(U)})$ are those red points, which locate in the annular region between blue circle and red circle.

Comprehensively, according to Theorem 5, 6 and 7, in order to obtain $N_\epsilon(q)$, we only need to search those red points in the annular region. All distance computations from p to blue and black points are reduced.

3.2. The proposed algorithm

We introduce a new clustering algorithm named NQ-DBSCAN based on the theorems mentioned above. Algorithm 3 shows the main procedures of NQ-DBSCAN. Algorithm 4 illustrates the detail of our improved *ExpandCluster* which retrieves all density-reachable neighbors from a core point, and Algorithm 5 presents the implementation of Theorem 7.

Algorithm 3 NQ-DBSCAN (P,ϵ,$MinPts$)

Input:

P: a set of unclassified points;

ϵ: the maximum distance;

$MinPts$: the minimum points to form a cluster;

Output: cluster id of each point;

Initialize cluster id $C = 0$
for each unclassified point $p \in P$ **do**
  //retrieve all neighbors within 2ϵ-neighborhood of p
  $N_{2\epsilon}(p)= RangeQuery(p, 2\epsilon)$
  if $|N_{2\epsilon}(p)| > MinPts$ **then**
    $dists \leftarrow$ all distances from p to $N_{2\epsilon}(p)$
    $[distArr, pLoc]$ =sort($dists$) //$distArr$ saves the sorted $dists$, while $pLoc$ is a vector that saves the corresponding points such that $d_{p,pLoc(i)} \leq d_{p,pLoc(i+1)}$
    if $distArr[MinPts] \leq \epsilon$ **then**
      // According to **Theorem 2** p is a core point, then we expand it.
      $drPts$=***ImprovedExpandCluster***$(p,pLoc,distArr, \epsilon,MinPts)$
      Set the cluster id of all points in $drPts$ as C
      $C \leftarrow C + 1$
    else
      Use binary search algorithm find $O = \{o|o \in pLoc$ and $d_{p,o} < distArr(MinPts) - \epsilon\}$, and set all points in O as noise (**Theorem 4**)
    end if
  else
    Set $O = \{q|q \in N_\epsilon(p)\}$ as noise (**Theorem 3**)
  end if
end for

In Algorithm 3 (NQ-DBSCAN), the main steps are below.

- Select an unclassified point p from P, then use *RangeQuery* to retrieve $N_{2\epsilon}(p)$ (line 4), and sort the distances form p to its 2ϵ-neighbors.
- According to Theorem 2, we can easily judge whether p is core point or not, as shown in line 8.
- If p is a core point, it will use *ImprovedExpandCluster* to find all points that are density-reachable from p ($drPts$), as shown in line 10. All points in $drPts$ will be marked as the same cluster id.
- According to Theorem 3 and Theorem 4, we are able to effectively find non-core points. If $\mid N_\epsilon(p) \mid < MinPts$, p is a non-core point and its neighbors are also highly possible to be non-core point, as line 14 shows. If $\mid N_{2\epsilon}(p) \mid < MinPts$, then all points in $N_\epsilon(p)$ are labeled as noise, as line 17 shows.

Algorithm 4 (*ImprovedExpandCluster*) is a new algorithm that retrieves all density-reachable points, *drPts*, from point p, which improves Algorithm 2 greatly. The main steps are shown as below.

- First initialize $drPts = N_\epsilon(p)$ by binary searching from $distArr$ and $pLoc$.
- Second, select an unclassified point q from $pLoc$. If $d_{p,q} \leq \epsilon$ we use *NeighborQuery* to effectively get $N_\epsilon(q)$, and if q is a core point $N_\epsilon(q)$ will be added to the set $drPts$. Repeat this step until all points in $pLoc$ are handled.
- Third, select a new unclassified point $p \in drPts$. If p is a core point then use *RangeQuery* again to update $N_{2\epsilon}(p)$, $pLoc$ and $distArr$, and then repeat the second step, until all points in $drPts$ are visited.

Algorithm 5 (*NeighborQuery*) is the implementation of Theorem 7, it uses binary search algorithm to obtain $N_\epsilon(q)$ in $N_{2\epsilon}(p)$ rather than the whole data set, as shown in Line 2 - Line 5.

Algorithm 4 ImprovedExpandCluster (*p, pLoc, distArr, ϵ, MinPts*)

Input:

p: reference point;

pLoc: saves all points in $N_{2\epsilon}(p)$ such that $d_{p,pLoc(i)} \leq d_{p,pLoc(i+1)}$;

distArr: the sorted distances from p to $N_{2\epsilon}(p)$;

ϵ: the maximum distance;

MinPts: the minimum points to form a cluster.

Output: drPts: all density-reachable neighbor points from p.

1: binary search $drPts = \{o | o \in pLoc \ s.t. \ d_{p,o} \leq \epsilon\}$
2: **for** each point q saved in *pLoc* **do**
3: **if** q is unclassified **then**
4: **if** $d_{p,q} \leq \epsilon$ **then**
5: $N_\epsilon(q)$=***NeighborQuery*** *(p,q,pLoc,distArr,ϵ,MinPts)*
6: **if** $|N_\epsilon(q)| \geq MinPts$ **then**
7: $drPts$=$drPts \cup N_\epsilon(q)$
8: **end if**
9: **end if**
10: **end if**
11: **end for**
12: $p \leftarrow$ select an unclassified point o in $drPts$
13: **if** p is a core point **then**
14: $N_{2\epsilon}(p)$ =***RangeQuery****(p,2*ϵ)*
15: $dists \leftarrow$ distances from p to all points in $N_{2\epsilon}(p)$
16: $[distArr, pLoc] = sort(dists)$
17: go to Line 2
18: **end if**

Take Fig. 3 for example again, p is a core point, its 2ϵ-neighbors have already been retrieved by *RangeQuery*. $\forall q \in N_\epsilon(p)$, in order to retrieve $N_\epsilon(q)$, *NeighborQuery* only checks those red points.

Algorithm 5 NeighborQuery(p,q,$pLoc$,$distArr$,ϵ,$MinPts$)

Input:

p: reference point;

q: current search point;

$pLoc$: the points number of neighbor sequence;

$distArr$: the points distance of neighbor sequence;

ϵ: the maximum distance;

$MinPts$: the minimum points to form a cluster;

Output: $N_\epsilon(q)$.

1: // determine L and U according to **Theorem 5, 6 and 7**

2: binary search index L such that $distArr(L) > d_{p,q} - \epsilon$

3: binary search index U such that $distArr(U) < d_{p,q} + \epsilon$

4: $possibleNeighbor = pLoc(L : U)$

5: $N_\epsilon(q) = pLoc(1 : L) \bigcup \{o | o \in possibleNeighbor \ s.t. \ d_{q,o} < \epsilon\}$

3.3. Correctness analysis

As shown in Algorithm 4 and 5, based on Theorems 5, 6 and 7 we can see that if p is a core point Algorithm 4 only retrieve all density-reachable points from p, which is equivalent to Algorithm 2.

Similarly, based on Theorem 2, 3 and 4, as well as Algorithm 4, NQ-DBSCAN (Algorithm 3) is also guaranteed to be equivalent to DBSCAN (Algorithm 1). Thus NQ-DBSCAN meets the requirement of *Maximality* and *Conectivity* defined in Definition 7, as well as Lemma 1 and Lemma 2 in [33] are also satisfied.

3.4. Complexity analysis

The key processes in NQ-DBSCAN are *RangeQuery* and *NeighborQuery*, and time complexity of NQ-DBSCAN highly depends on them.

The complexity of *RangeQuery* can be $O(\log n)$ with the help of indexing techniques, such as R*-tree, otherwise is $O(n)$. In this paper, we use quadtree-

like hierarchical tree grid [40], which works well in many cases, but it still performs not good for very high dimensional data that are sparse. The complexity of building this grid is $O(n)$.

The complexity of *NeighborQuery* is $O(\log(nei))$ by using binary search method, where nei is the number of p's neighbors.

Therefore, the whole time complexity of NQ-DBSCAN is $O(\alpha * (\log(n) + nei * \log(nei)) + \beta * log(nei) - \gamma)$, where α is execution times of *RangeQuery*, β is execution times of *NeighborQuery*, and γ is the total number of filtered points that are unnecessary to visit (including some non-core points and noise points), respectively. Obviously, $\alpha + \beta + \gamma = n$, and then $\alpha + \beta <= n$.

In the case of $MinPts$ is very large such that $\gamma \to n$, i.e. most points are identified as non-core points directly, the complexity is $O(1)$. However, it is meaningless. The best complexity is $O(n)$, in the case of both α and nei are small, while $\beta \to n$. Generally, the average complexity of NQ-DBSCAN is about $O(n * log(n))$ if ϵ and $MinPts$ are properly chosen. Of course, without indexing technique, the average complexity is also $O(n^2)$.

4. Experiments

In this section, we conduct experiments to evaluate the performance of NQ-DBSCAN, and make comparisons with original DBSCAN and ρ-approximate DBSCAN [40], on synthetic and realtime data sets.

4.1. Algorithms

Algorithms. Our experiments involve four algorithms as follows:

- DBSCAN: the original DBSCAN algorithm in [33];
- NQ-DBSCAN: the proposed algorithm without using indexing technique;
- "NQ-DBSCAN with indexing": the proposed algorithm with quadtree-like hierarchical tree grid indexing;
- Approx: the ρ-approximate DBSCAN algorithm.

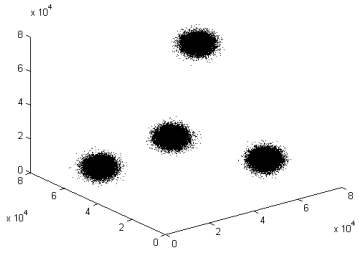


Figure 4: An example of test case which has 4 hyper-spherical data without noise.

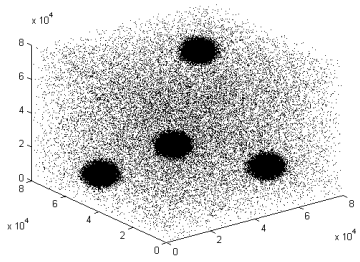


Figure 5: An example of test case which has 4 hyper-spherical data with noise.

DBSCAN and NQ-DBSCAN were run on a machine equipped with 3.3GHz CPU and 8 GB memory, the operating system was Windows 10 64-bit and programs were coded in MATLAB.

Approx were coded in C++, and was run on Linux (Ubuntu 14.04) operating system with the same hardware configuration.

4.2. Data sets

We use two kinds of data sets in our experiments, one is synthetic data and the other is realtime data. All data are normalized such that their domain is $[0, 10^5]$ for each dimension.

Synthetic Data sets. Two types of synthetic data sets are used in our experiments as below.

(1) Gaussian Hyper-sphere

We generate a series of Gaussian hyper-spherical test cases, some test cases have 20% noise, and the others are noise-free. Each test case includes 4 clusters, and points of each cluster follows Gaussian distribution with quite different mean from the other clusters. Two $3d$ visual Gaussian Hyper-spherical test cases are shown in Fig. 4 and Fig. 5, respectively.

(2) Uniform Hyper-cube

We also generate a series of hyper-cubical test cases, some test cases have

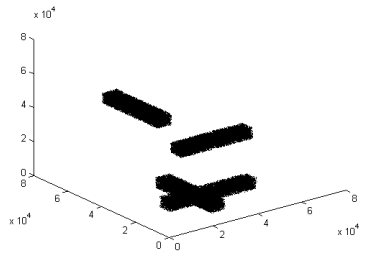


Figure 6: An example of test case which has 4 hyper-cubical data without noise (3 clusters).

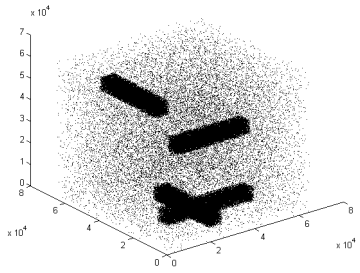


Figure 7: An example of 4 hyper-cubical data with noise (3 clusters).

20% noise, and the others are noise-free. Each test case includes 4 hypercubes, and points of each hypercube uniformly distributed. There are two hypercubes that intersect with each other. Therefore, there are 3 clusters in all test cases in fact. Two $3d$ visual Hyper-cubical test cases are shown in Fig. 6 and Fig. 7, respectively.

The details of these data sets are shown as follows:

- Spheredata 1: without noise, n=50,000, has 10 test cases with d ranging from 5 to 50.
- Spheredata 2: with noise, n=100,000, d=10.
- Spheredata 3: with noise, d=5, 10 test cases with n ranging from 20,000 to 200,000.
- Spheredata 4: with noise, d=20, 10 test cases with n ranging from 20,000 to 200,000.
- Spheredata 5: with noise, n=50,000, has 10 test cases with d ranging from 5 to 50.

- Spheredata 6: with noise, n=100,000, has 10 test cases with d ranging from 5 to 50.
- Cubedata 1: without noise, n=50,000, has 10 test cases with d ranging from 5 to 50.
- Cubedata 2: with noise, d=5, has 10 test cases with n ranging from 10,000 to 100,000.
- Cubedata 3: with noise, d=10, has 10 test cases with n ranging from 10,000 to 100,000.
- Cubedata 4: with noise, n=50,000, has 15 test cases with d ranging from 10 to 150.

Real Data sets. Some real data sets were employed in our experiments as follows:

The first, House (household) is a 7 dimensional data set with cardinality 2,075,259, which includes all the attributes of the Household database comes from the UCI archive [2] except the temporal columns date and time. Points in the original database with missing coordinates were removed.

The second, ReactionNetwork is KEGG Metabolic Reaction Network (Undirected) Data Set which also comes from UCI. It is a 28-dimensional data set with cardinality 65,554.

The third, BlogFeedback [55] also comes from the UCI archive. It is a 59-dimensional data set with cardinality 52,397 obtained by taking the first 59 numeric attributes and the 60^{th}-280^{th} attributes are omitted, because most values in the 60^{th}-280^{th} attributes are zero.

The fourth, KDD04 is KDD Cup 2004 data. It is 76-dimensional data set with cardinality 145,751.

The fifth, MNIST[3] is a handwritten digits data set, which includes 70,000

[2] http://archive.ics.uci.edu/ml

[3] http://yann.lecun.com/exdb/mnist/

images with size of 28×28 pixel. We pick up 10,000 images and transform each 28×28 image matrix into a feature vector with $28 \times 28 = 784$ dimensions. Therefore, MNIST used in our experiments is a 784-dimensional data set with cardinality 10,000.

The sixth, PAM (PAMPA2), which comes from UCI, is a 4-dimensional data set with cardinality 3,850,505.

The last one is MORPH [56] which is the largest publicly available longitudinal face database[4], includes 79,897 face photographs with size of 70×80 pixel. Also, we pick up 10,000 face photographs of MORPH in our experiments. We convert the RGB images to gray images, and then transform each gray image matrix into a feature vector with $70 \times 80 = 5,600$ dimensions. Therefore, MORPH used in our experiments is a 5600-dimensional data set with cardinality 10,000.

4.3. Experiment 1: Two Examples

We benchmark NQ-DBSCAN on two test cases, the first one is t4.8k [57], which is a 2-dimensional data set with cardinality 8,000, and the other is Aggregation [58], which is a 2-dimensional data set with cardinality 788. The distribution of two data sets and the clusters obtained by NQ-DBSCAN are shown in Fig. 8. It illustrates that NQ-DBSCAN has the same ability as DBSCAN to detect complex shapes.

4.4. Experiment 2: Influence of Noise and Dimensionality

The purpose of this part is to check impaction of noise and dimensionality on NQ-DBSCAN and Approx.

Firstly, we conduct an experiment on Spheredata 1 and Cubedata 1 which are noise-free. As shown in Fig. 9 and Fig. 10, we can see that in the case of dimension is less than 50, Approx and NQ-DBSCAN performs similarly on both test cases, and the running time increase linearly with dimension.

[4] http://www.faceaginggroup.com/morph/

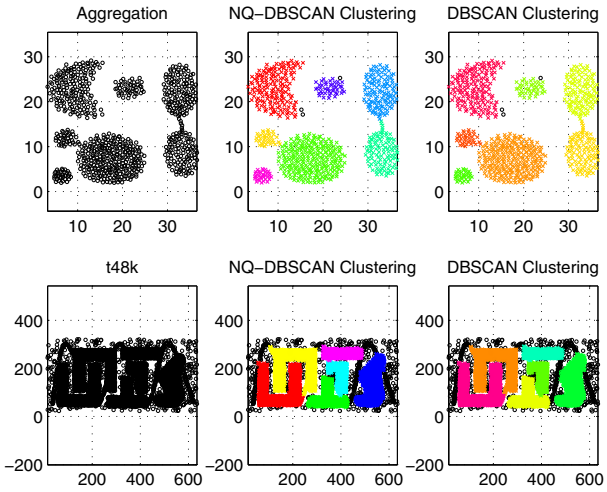


Figure 8: Two clustering examples of NQ-DBSCAN and DBSCAN on Aggregation and t4.8k.

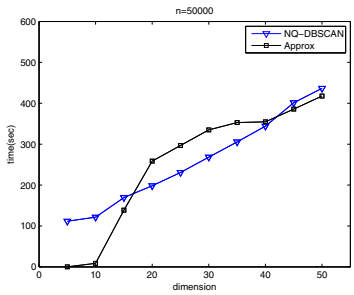


Figure 9: The performance of two approaches on Spheredata 1 with n=50,000. ($MinPts$ = 100 and ϵ=10,000)

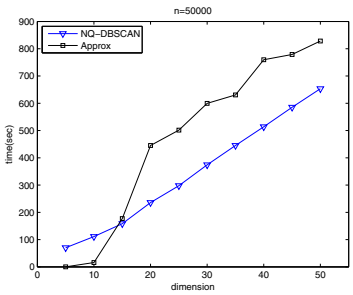


Figure 10: The performance of two approaches on Cubedata 1 with n=50,000. ($MinPts$ = 100 and ϵ=2000)

Secondly, we conduct experiments on some test cases with noise and with higher dimension, i.e., Spheredata 5, Spheredata 6, Cubedata 4 and Cubedata 5.

As Fig. 11, Fig. 12 and Fig. 13 show, we can see that the performance of Approx on these test cases is far worsen than it was on Spheredata 1 and

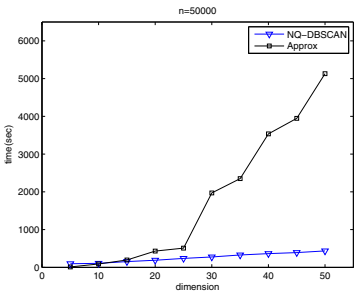


Figure 11: Running time vs. dimension on Spheredata 5.

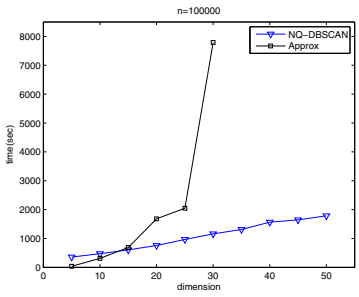


Figure 12: Running time vs. dimension on Spheredata 6.

Cubedata 1, which means noise has great impaction on Approx. While NQ-DBSCAN works still stable and much better than Approx. The reason lies in noise distributes in the entire data space rather than concentrates in several small space, then additional cells are needed to save noise, which badly affects the efficiency of Approx.

We also can see from these experiments, the efficiency of Approx decrease rapidly with dimension. Because with the increasing of the data dimension, each cell becomes smaller, and the number of cells rise exponentially, which finally leads to Approx degenerate to an $O(n^2)$ algorithm. While, the running time of NQ-DBSCAN still increases linearly with dimension, which implies that NQ-DBSCAN is weakly affected by "curse of dimensionality".

4.5. Experiment 3: The Effect of ϵ and MinPts

The purpose of this experiment is to check the effect of ϵ on the proposed algorithm. Spheredata 2 was used in this experiment with cardinality 100,000 and the dimension is 10, *MinPts* was fixed to 50. Fig. 14 shows the performances of the 3 approaches. Clearly, both NQ-DBSCAN and Approx are quite better than DBSCAN.

Fig. 15 presents the execution times of *RangeQuery* and *NeighborQuery*, and Fig. 16 plots the average neighbors found in *RangeQuery* increasing with

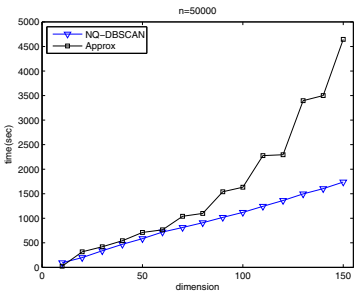


Figure 13: Running time vs. dimension on Cubedata 4.

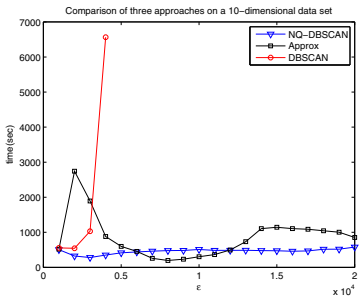


Figure 14: Running time vs.radius ϵ on Spheredata 2.

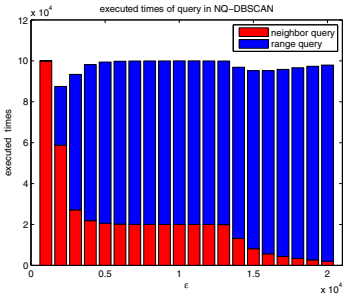


Figure 15: The execution times of *RangeQuery* (α) vs. the execution times of *NeighborQuery* (β) with ϵ increasing on Spheredata 2 ($d = 10$, $MinPts = 50$).

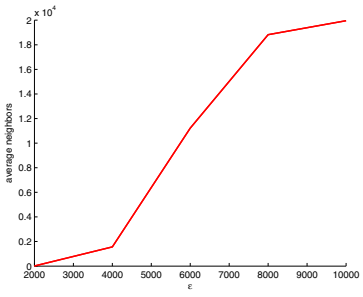


Figure 16: Average neighbors (*nei*) increasing with ϵ on Spheredata 2.

ϵ. From the two figures, we can see that execution times of *RangeQuery* $\alpha \to n$ and $n >> nei$ in the case of ϵ is very small, and β increases with ϵ. In the case of $\epsilon \in [2000, 4000]$, both α and *nei* are small, NQ-DBSCAN performs better than other cases.

Table. 1 shows the detail of some results on a subset of Spheredata 3 with small ϵ and $MinPts$. We can see that $\alpha >> \beta$ when $\epsilon = 1,000$, i.e. the execution times of *RangeQuery* is far larger than that of *NeighborQuery*, while α is greatly reduced when $\epsilon = 2,000$ and $\epsilon = 3,000$. We also notice that the running time

Table 1: The performances of NQ-DBSCAN on a sub set of Spheredata 3 with different ϵ and $MinPts$ (d=5, n=20,000).

ϵ	$MinPts$	running time (seconds)	range query (α)	nei query (β)	filtered points (γ)
1000	10	7.68	16659	2877	464
	30	7.06	17506	0	2494
	50	6.70	16099	0	3901
	100	5.87	14868	0	5132
2000	10	9.69	6843	13152	5
	30	9.40	6810	13173	17
	50	9.57	6811	13162	27
	100	9.17	6815	13145	40
3000	10	11.57	4850	15128	22
	30	11.15	4850	15128	22
	50	11.01	4850	15128	22
	100	11.08	4850	15128	22

Table 2: The performances of NQ-DBSCAN on a sub set of Spheredata 3 with large $MinPts$ (d=5, n=20,000).

ϵ	$MinPts$	running time (seconds)	range query (α)	nei query (β)	filtered points (γ)
2000	200	6.88	14299	649	5052
	1000	2.87	6833	0	13167
	3000	2.73	6718	0	13282
	7000	2.79	6718	0	13282
	12000	2.80	6718	0	13282
5000	200	14.51	3858	15862	280
	1000	14.06	3858	15862	280
	3000	9.87	4655	9025	6320
	7000	1.59	3861	0	16139
	12000	1.74	3861	0	16139
10000	200	19.91	1433	18531	36
	1000	19.79	1433	18531	36
	3000	19.95	1433	18531	36
	7000	0.61	1386	0	18614
	12000	0.60	1386	0	18614

of $\epsilon = 1,000$ is smaller than the others, because there are many filtered points. But its accuracy is not as good as that of $\epsilon = 2,000$ and $\epsilon = 3,000$.

Table. 2 illustrates more experiments on the same data set. In this experiment, we test the impaction of large $MinPts$. From the table we can see that the number of filtered points increase with $MinPts$, the more filtered points the fewer the running time, which is consistent with our analysis mentioned in Section 3.4. This experiment also implies that NQ-DBSCAN is highly efficient to find and filter noise, in other words, it is suitable for clustering data with a lot of noise, such as [59].

4.6. Experiment 4: Efficiency VS Cardinality

In this subsection, we conduct experiments on Spheredata 3, Spheredata 4, Cubedata 2 and Cubedata 3, respectively, to compare the efficiencies of NQ-DBSCAN, "NQ-DBSCAN with indexing" and Approx by changing the cardinalities of these cases.

Because Approx runs linearly in low-dimension, we can see that Approx outperforms NQ-DBSCAN in Fig. 17 and Fig. 18. However, with dimension increasing, things go different. In Fig. 19, we can see that in this 10-dimensional data set, Approx is still better than NQ-DBSCAN and "NQ-DBSCAN with indexing", but their performances are closer than that in 5 dimension. And then, Fig. 20 shows that the performance of Approx is inferior to both NQ-DBSCAN and "NQ-DBSCAN with indexing" in the 20-dimensional data set (Spheredata 4).

All experiments above obtain correct results as we expected, i.e. in Fig. 17 and Fig. 19, we obtain 4 hyper-spherical clusters, and in Fig. 18 and Fig. 20, we get 3 clusters which include 4 hyper-cubes.

We also can see that "NQ-DBSCAN with indexing" seems to be an $O(n)$ algorithm, because proper ϵ and $MinPts$ are used such that α and nei are both small and $\beta \to n$, which is consistent with the theoretical analysis mentioned above.

4.7. Experiment 5: Experiments on Realtime Applications

In order to test the performance of NQ-DBSCAN and "NQ-DBSCAN with indexing" in realtime applications, we benchmark it on six test cases with different dimensions, i.e. Household (7 dim), ReactionNetwork (28 dim) , BlogFeedback (59 dim), KDD04 (76 dim), MNIST (784 dim) and *MORPH* (5,600 dim), and compare them with ρ-Approximate DBSCAN. In the following experiments, $MinPts$ are all fixed to 100.

Fig. 21 and Fig. 22 show that Approx runs linearly in Household (7 dim) and ReactionNetwork (28 dim), and its performance is better than the proposed

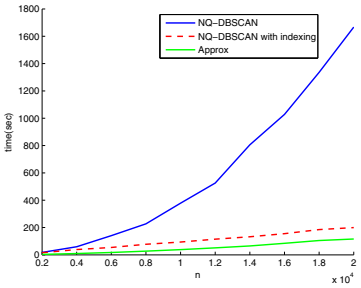


Figure 17: Running time vs. cardinality on Spheredata 3 (5 dim, ϵ=2,000 and $MinPts$=100).

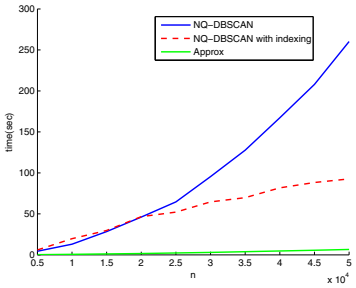


Figure 18: Running time vs. cardinality n on Cubedata 2 (5 dim, ϵ=2000 and $MinPts$=100).

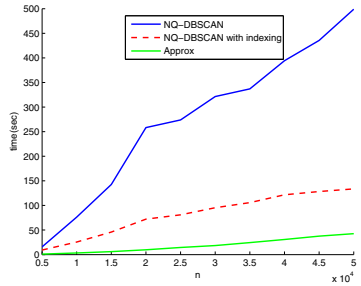


Figure 19: Running time vs. cardinality n on Cubedata 3 (10 dim, ϵ=2000 and $MinPts$=100).

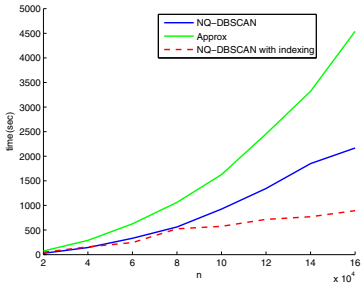


Figure 20: Running time vs. cardinality on Spheredata 4. (20 dim, ϵ=2,000 and $MinPts$=100)

algorithm (One reason that the proposed algorithm runs slower in ReactionNetwork is the code efficiency in Matlab is not as good as C++).

While the comparisons in Fig. 23 and Fig. 24 present that Approx runs in $O(n^2)$, which is clearly inferior to "NQ-DBSCAN with indexing" on BlogFeedback (59-dim) and KDD04 (76 dim), respectively.

Clearly, we can see that the higher the dimension, the more advantages the proposed algorithm to Approx, and the four figures above prove that "NQ-DBSCAN with indexing" runs in $O(n * log(n))$.

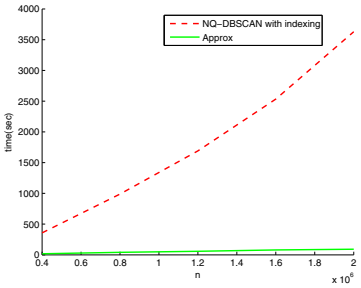


Figure 21: Running time VS Cardinality on HouseHold (7 dim) with $\epsilon = 1,000$.

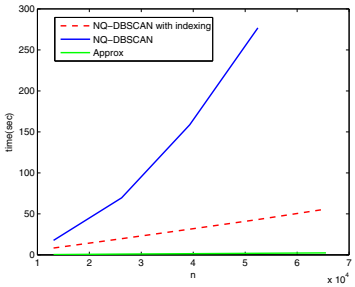


Figure 22: Running time VS Cardinality on ReactionNetwork (28 dim) with $\epsilon = 1,000$.

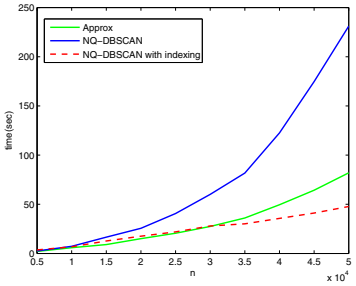


Figure 23: Running time VS Cardinality on BlogFeedback (59 dim) with $\epsilon = 1,000$.

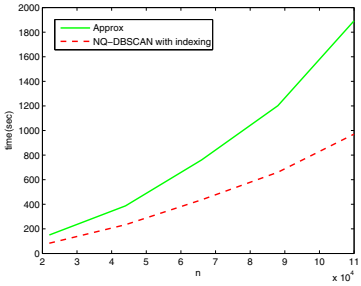


Figure 24: Running time VS Cardinality on KDD04 (76 dim) with $\epsilon = 1,000$.

The following two experiments are conducted on MNIST and *MORPH* are that very high-dimensional and sparse, the quadtree-like hierarchical tree grid fails to work. Thus, we only compare NQ-DBSCAN and Approx by changing different ϵ. We can see NQ-DBSCAN outperforms Approx as Fig. 25 and Fig. 26 illustrate. The reason lies in the grid technique is useless in high dimension as mentioned in Theorem 1. While NQ-DBSCAN seems free from dimensionality, which makes it more suitable for clustering realtime data than Approx.

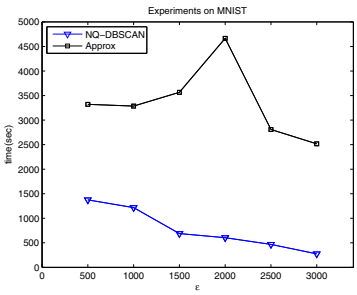


Figure 25: The performance of two approaches on MNIST (784 dim).

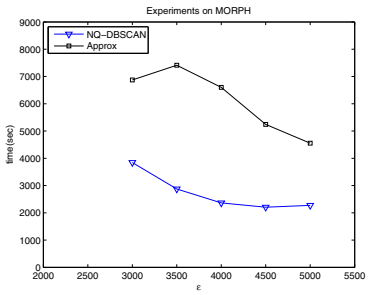


Figure 26: The performance of two approaches on MORPH (5,600 dim).

4.8. The robust of algorithm

According to Huber[60], a robust procedure can be characterized by the following: 1) it should have a reasonably good efficiency (accuracy) at the assumed model; 2) small deviations from the model assumptions should impair the performance only by a small amount; and 3) larger deviations from the model assumptions should not cause a catastrophe.

In order to test the accuracy of the proposed algorithm and ρ-approximate DBSCAN, we conduct some experiments based on an assumption that the clustering labels obtained by DBSCAN is the standard correct result, and evaluate the precision of two approaches as following, which is also used in our previous works[61, 62].

Firstly, we use the original DBSCAN to cluster a data set, and return cluster labels $L1 = \{A_1, A_2, ..., A_k\}$. Secondly, run NQ-DBSCAN and ρ-approximate DBSCAN on the same data set, and obtain $L2 = \{B_1, B_2, ..., B_m\}$ and $L3 = \{C_1, C_2, ..., C_p\}$, respectively.

As we know, the clustering results got by a clustering algorithm may have different labels from that got by the other algorithm, e.g. cluster 'A1' obtained by one approach may be the same as cluster 'B2' of the other. Therefore, we have to match labels first, then use the matched labels to calculate *Precision*. In our experiments, we use Kuhn-Munkras[63] to maximum match two cluster

Table 3: The precision of NQ-DBSCAN on three data sets. All accuracies are calculated by comparing to the result of original DBSCAN. The parameters of NQ-DBSCAN and DBSCAN are given in the formation as [ϵ, MinPts].

data set	BLOG [2000,30]	HOUSE [500,30]	PAM [500,30]
Approx	94.54%	99.67%	99.78%
NQ-DBSCAN	99.97%	99.6%	100%

labels, which has been used in our previous works [61, 62].

For example, if a data set has 3 clusters labeled as 'A1', 'A2' and 'A3' obtained by DBSCAN, and our method finds 4 clusters with labels 'B1', 'B2', 'B3' and 'B4' on the same data set. Suppose there are 3 matched pairs found by Kuhn-Munkres algorithm: ('A1','B2'), ('A2', 'B1') and ('A3', 'B4'). If p is labeled as 'A1' by DBSCAN and clustered as 'B2' by our approach, respectively, we consider this prediction as correct. If p is labeled as 'A1' by DBSCAN and clustered as 'B1' by our approach it is wrong. Similar to other cases.

As presented in Table. 3, the precisions truly speak of that our approach nearly achieves the same results as DBSCAN, the petty difference is caused by the visiting order is different from that of original DBSCAN, because DBSCAN is non-determinative. While ρ-approximate DBSCAN is little inferior to NQ-DBSCAN.

In order to evaluate the performance of NQ-DBSCAN on data sets with deviations, we select 10% data points from $BLOG$, $HOUSE$ and PAM, respectively, and then shift these points randomly in each dimension by adding a random value η, where $\eta = offset * random()$, and $offset$ is predefined. As Table. 4 demonstrates, the accuracies of both NQ-DBSCAN and ρ-Approximate DBSCAN are similarly affected by the deviations of data set, but it is acceptable.

4.9. Comprehensive Analysis

From all experiments above, we can see that Approx runs linearly in low dimension. However, with the increasing of dimension Approx degenerates to

Table 4: The precision of NQ-DBSCAN on three data sets with deviations. All accuracies are calculated by comparing to the result of original DBSCAN. The parameters of NQ-DBSCAN are given in the formation as [ϵ, MinPts].

	offset	BLOG [2000,30]	HOUSE[500,30]	PAM[500,30]
NQ-DBSCAN	100	99.92%	99.31%	99.78%
	200	99.66%	99.73%	99.77%
	300	90.29%	89.93%	99.74%
	400	90.29%	89.93%	98.76%
	500	90.29%	89.93%	93.63%
Approx	100	94.49%	99.65%	99.78%
	200	94.30%	99.38%	99.77%
	300	92.77%	89.79%	99.68%
	400	85.92%	89.79%	98.98%
	500	85.92%	89.79%	93.68%

be an $O(n^2)$ algorithm. While "NQ-DBSCAN with indexing" averagely runs in $O(n)$ or $O(n * log(n))$ in many cases.

In very large high dimension NQ-DBSCAN still outperforms Approx without indexing technique. The reason lies in the grid techniques used in Approx is useless in high dimension, while the neighbor searching technique used in NQ-DBSCAN is almost not affected by the dimensionality.

In the case of data sets having a lot of noise, NQ-DBSCAN works much better, because noise has side effects on Approx. The underlying cause is that noise always distributes in the whole data space rather than concentrates in some small regions, which results in many cells are needed to save noise, and then leads to the efficiency of ρ-approximate rapidly decline. Due to the capability of effectively finding non-core points (Theorem 3 and 4), NQ-DBSCAN can run in $O(n)$ expected time.

In addition, NQ-DBSCAN is an exact algorithm, which is also an important advantage to the approximate algorithm ρ-Approximate DBSCAN.

5. Conclusion

Today, large collections of data are explosively created in different fields, and most of these data are high dimensional with a lot of noise, which bring great challenging to clustering. DBSCAN is a creative and elegant technique for density-based clustering. However, it is rendered almost useless for high-dimensional data, due to the "curse of dimensionality", which limits its applicability in many realtime applications. ρ-approximate DBSCAN [40] is an efficient approach designed to replace DBSCAN for big data. By using quadtree-like hierarchical grid and small sacrifice in accuracy, ρ-approximate has a computational time that scales only linearly in n. However, it declines to an $O(n^2)$ algorithm in high dimension because the grid technique is also useless in high dimension. Also, we find the efficiency of ρ-approximate is greatly reduced when dealing with high dimensional data that has much noise, because the grid technique is useless in high dimension and noise needs additional cells to save.

In this paper, we propose a clustering algorithm, named NQ-DBSCAN which may return the exact result as DBSCAN, to improve DBSCAN, by using neighbor searching technique and indexing technique to filter great number of unnecessary density computations. The underlying idea is: point p and point q should have similar neighbors, provided p and q are close to each other; given a certain ϵ, the closer they are, the more similar their neighbors are.

Our experiments have shown that the proposed method outperforms ρ-approximate in high dimension, also it performs better in data sets with a lot of noise. Although, the worse complexity of NQ-DBSCAN is still $O(n^2)$, but its average complexity is about $O(n * log(n))$ with the help of indexing technique, and the best case is $O(n)$ if proper parameters (ϵ and $MinPts$) are used.

The indexing technique we used is quadtree-like hierarchical tree grid, but it fails to work in some sparse and very high-dimensional data. Therefore, in future work, we will try to improve quadtree-like hierarchical tree grid, by combining the merits of other techniques, such as product quantization for nearest neighbor search [52], LSH (Locality-Sensitive Hashing) [53], FLANN [54] etc.

Acknowledgment

The National Science Foundation of China (No.61673186,71771094) ; this work was supported by the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) (NO.201700002); the Open Project Program of the State Key Lab of CAD&CG(Grant No.A1722), Zhejiang University; the Natural Science Foundation of Fujian Province (No.2016J01303); Project of science and technology plan of Fujian Province of China (No.2017H01010065); the Graduate Students Research and Innovation Ability Cultivation Plan of Huaqiao University (No.1511414009); the Huaqiao University graduate research project of education reform (16YJG13).

Reference

References

[1] J. Song, L. Gao, F. Nie, H. T. Shen, Y. Yan, N. Sebe, Optimized graph learning using partial tags and multiple features for image and video annotation, IEEE Transactions on Image Processing 25 (11) (2016) 4999–5011.

[2] J. Song, L. Gao, F. Zou, Y. Yan, N. Sebe, Deep and fast: Deep learning hashing with semi-supervised graph construction, Image and Vision Computing.

[3] J. Song, H. T. Shen, J. Wang, Z. Huang, N. Sebe, J. Wang, A distance-computation-free search scheme for binary code databases, IEEE Transactions on Multimedia 18 (3) (2016) 484–495.

[4] W. Zhou, M. Yang, X. Wang, H. Li, Y. Lin, Q. Tian, Scalable feature matching by dual cascaded scalar quantization for image retrieval, IEEE transactions on pattern analysis and machine intelligence 38 (1) (2016) 159–171.

[5] S. Zhang, Q. Huang, S. Jiang, W. Gao, Q. Tian, Affective visualization and retrieval for music video, IEEE Transactions on Multimedia 12 (6) (2010) 510–522.

[6] A. K. Rajagopal, R. Subramanian, E. Ricci, R. L. Vieriu, O. Lanz, N. Sebe, et al., Exploring transfer learning approaches for head pose classification from multi-view surveillance images, International Journal of Computer Vision 109 (1-2) (2014) 146–167.

[7] Y. Yan, E. Ricci, R. Subramanian, G. Liu, O. Lanz, N. Sebe, A multi-task learning framework for head pose estimation under target motion, IEEE transactions on pattern analysis and machine intelligence 38 (6) (2016) 1070–1083.

[8] B. F. Qaqish, J. J. OBrien, J. C. Hibbard, K. J. Clowers, Accelerating high dimensional clustering with lossless data reduction, Bioinformatics (2017) btx328.

[9] Z. Deng, K.-S. Choi, Y. Jiang, J. Wang, S. Wang, A survey on soft subspace clustering, Information Sciences 348 (2016) 84–106.

[10] O. Limwattanapibool, S. Arch-int, Determination of the appropriate parameters for k-means clustering using selection of region clusters based on density dbscan (srcd-dbscan), Expert Systems.

[11] L. Bai, X. Cheng, J. Liang, H. Shen, Y. Guo, Fast density clustering strategies based on the k-means algorithm, Pattern Recognition 71 (2017) 375–386.

[12] N. A. Yousri, M. S. Kamel, M. A. Ismail, A distance-relatedness dynamic model for clustering high dimensional data of arbitrary shapes and densities, Pattern Recognition 42 (7) (2009) 1193–1209.

[13] C. Zhong, D. Miao, R. Wang, A graph-theoretical clustering method based on two rounds of minimum spanning trees, Pattern Recognition 43 (3) (2010) 752–766.

[14] M. Ester, H.-P. Kriegel, J. Sander, Algorithms and applications for spatial data mining, Geographic Data Mining and Knowledge Discovery 5 (6).

[15] W. A. Barbakh, Y. Wu, C. Fyfe, Non-standard parameter adaptation for exploratory data analysis, Vol. 249, Springer, 2009.

[16] J. Han, J. Pei, M. Kamber, Data mining: concepts and techniques, Elsevier, 2011.

[17] J. Hou, W. Liu, E. Xu, H. Cui, Towards parameter-independent data clustering and image segmentation, Pattern Recognition 60 (2016) 25–36.

[18] S. Mitra, P. P. Kundu, Satellite image segmentation with shadowed c - means, Information Sciences 181 (17) (2011) 3601–3613.

[19] S. Das, S. Sil, Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm, Information Sciences An International Journal 180 (8) (2010) 1237–1256.

[20] Y. C. Song, H. D. Meng, M. J. OGrady, G. M. P. OHare, The application of cluster analysis in geophysical data interpretation, Computational Geosciences 14 (2) (2010) 263–271.

[21] A. Ghosh, N. S. Mishra, S. Ghosh, Fuzzy clustering algorithms for unsupervised change detection in remote sensing images, Information Sciences 181 (4) (2011) 699–715.

[22] Y. J. Wang, H. S. Lee, A clustering method to identify representative financial ratios, Information Sciences 178 (4) (2008) 1087–1097.

[23] J. Li, K. Wang, L. Xu, Chameleon based on clustering feature tree and its application in customer segmentation, Annals of Operations Research 168 (1) (2009) 225–245.

[24] Q. Bsoul, J. Salim, L. Q. Zakaria, An intelligent document clustering approach to detect crime patterns , Procedia Technology 11 (1) (2013) 1181–1187.

[25] C.-W. Huang, K.-P. Lin, M.-C. Wu, K.-C. Hung, G.-S. Liu, C.-H. Jen, Intuitionistic fuzzy c-means clustering algorithm with neighborhood attraction in segmenting medical image, Soft Computing 19 (2) (2015) 459–470.

[26] V. P. Ananthi, P. Balasubramaniam, T. Kalaiselvi, A new fuzzy clustering algorithm for the segmentation of brain tumor, Soft Computing (2015) 1–21.

[27] R. Chinchuluun, W. S. Lee, J. Bhorania, P. M. Pardalos, Clustering and Classification Algorithms in Food and Agricultural Applications: A Survey, Springer US, 2009.

[28] A. Hatamlou, Black hole: A new heuristic optimization approach for data clustering, Information Sciences 222 (3) (2013) 175–184.

[29] J. G. Lee, J. Han, K. Y. Whang, Trajectory clustering: a partition-and-group framework, in: ACM SIGMOD International Conference on Management of Data, 2007, pp. 593–604.

[30] X. T. Yuan, B. G. Hu, R. He, Agglomerative mean-shift clustering, IEEE Transactions on Knowledge and Data Engineering 24 (2) (2012) 209–219.

[31] W.-Y. Chen, Y. Song, H. Bai, C.-J. Lin, E. Y. Chang, Parallel spectral clustering in distributed systems, IEEE transactions on pattern analysis and machine intelligence 33 (3) (2011) 568–586.

[32] S. Mitra, A parallel clustering technique for the vehicle routing problem with split deliveries and pickups, Journal of the Operational Research Society 59 (11) (2008) 1532–1546.

[33] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise., in: Kdd, Vol. 96, 1996, pp. 226–231.

[34] B. Borah, D. Bhattacharyya, An improved sampling-based dbscan for large spatial databases, in: Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on, IEEE, 2004, pp. 92–96.

[35] C. Ruiz, M. Spiliopoulou, E. Menasalvas, C-dbscan: Density-based clustering with constraints, in: International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing, Springer, 2007, pp. 216–223.

[36] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, J. Fan, Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce, in: Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on, IEEE, 2011, pp. 473–480.

[37] K. M. Kumar, A. R. M. Reddy, A fast dbscan clustering algorithm by accelerating neighbor searching using groups method, Pattern Recognition 58 (2016) 39–48.

[38] A. Tramacere, C. Vecchio, γ-ray dbscan: a clustering algorithm applied to fermi-lat γ-ray data-i. detection performances with real and simulated data, Astronomy & Astrophysics 549 (2013) A138.

[39] S. T. Mai, S. Goebl, C. Plant, A similarity model and segmentation algorithm for white matter fiber tracts, in: 2012 IEEE 12th International Conference on Data Mining, IEEE, 2012, pp. 1014–1019.

[40] 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, ACM, 2015, pp. 519–530.

[41] J. Wang, Z. Deng, K.-S. Choi, Y. Jiang, X. Luo, F.-L. Chung, S. Wang, Distance metric learning for soft subspace clustering in composite kernel space, Pattern Recognition 52 (2016) 113–134.

[42] P. Qian, Y. Jiang, Z. Deng, L. Hu, S. Sun, S. Wang, R. F. Muzic, Cluster prototypes and fuzzy memberships jointly leveraged cross-domain maximum entropy clustering, IEEE transactions on cybernetics 46 (1) (2016) 181–193.

[43] Z. Deng, Y. Jiang, F.-L. Chung, H. Ishibuchi, K.-S. Choi, S. Wang, Transfer prototype-based fuzzy clustering, IEEE Transactions on Fuzzy Systems 24 (5) (2016) 1210–1232.

[44] Z. Deng, K.-S. Choi, F.-L. Chung, S. Wang, Enhanced soft subspace clustering integrating within-cluster and between-cluster information, Pattern Recognition 43 (3) (2010) 767–781.

[45] A. Gunawan, A faster algorithm for dbscan, Ph.D. thesis, Masters thesis, Technische University Eindhoven (2013).

[46] P. Viswanath, V. S. Babu, Rough-dbscan: A fast hybrid density based clustering method for large data sets, Pattern Recognition Letters 30 (16) (2009) 1477–1488.

[47] D. Birant, A. Kut, St-dbscan: An algorithm for clustering spatial–temporal data, Data & Knowledge Engineering 60 (1) (2007) 208–221.

[48] S. Mahran, K. Mahar, Using grid for accelerating density-based clustering, in: IEEE International Conference on Computer and Information Technology, 2008, pp. 35–40.

[49] C. Xiaoyun, M. Yufang, Z. Yan, W. Ping, Gmdbscan: Multi-density dbscan cluster based on grid, in: IEEE International Conference on E-Business Engineering, 2008, pp. 780–783.

[50] O. Uncu, W. A. Gruver, D. B. Kotak, D. Sabaz, Gridbscan: Grid density-based spatial clustering of applications with noise, in: IEEE International Conference on Systems, Man and Cybernetics, 2006, pp. 2976–2981.

[51] L. Zhang, Z. Xu, F. Si, Gcmddbscan: Multi-density dbscan based on grid and contribution, in: IEEE International Conference on Dependable, Autonomic and Secure Computing, 2013, pp. 502–507.

[52] H. Jegou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE transactions on pattern analysis and machine intelligence 33 (1) (2011) 117–128.

[53] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Commun. ACM 51 (1) (2008) 117C122.

[54] D. G. L. Marius Muja, Scalable nearest neighbor algorithms for high dimensional data, IEEE transactions on pattern analysis and machine intelligence 36 (11) (2014) 2227–2240.

[55] K. Buza, Feedback prediction for blogs, in: Data analysis, machine learning and knowledge discovery, Springer, 2014, pp. 145–152.

[56] K. Ricanek, T. Tesafaye, Morph: a longitudinal image database of normal adult age-progression, in: International Conference on Automatic Face and Gesture Recognition, 2006, pp. 341–345.

[57] G. Karypis, E.-H. Han, V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, Computer 32 (8) (1999) 68–75.

[58] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, ACM Transactions on Knowledge Discovery from Data (TKDD) 1 (1) (2007) 4.

[59] S. Maurus, C. Plant, Skinny-dip: Clustering in a sea of noise, in: SIGKDD, ACM, 2016, pp. 1055–1064.

[60] P. J. Huber, Robust statistics, in: International Encyclopedia of Statistical Science, Springer, 2011, pp. 1248–1251.

[61] Y. Chen, S. Tang, L. Zhou, C. Wang, J. Du, T. Wang, S. Pei, Decentralized clustering by finding loose and distributed density cores, Information Sciences 433-434 (2018) 510–526.

[62] Y. Chen, S. Tang, S. Pei, C. Wang, J. Du, N. Xiong, Dheat: A density heat-based algorithm for clustering with effective radius, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (2018) 649–660.

[63] H. W. Kuhn, The hungarian method for the assignment problem, Naval research logistics quarterly 2 (1-2) (1955) 83–97.

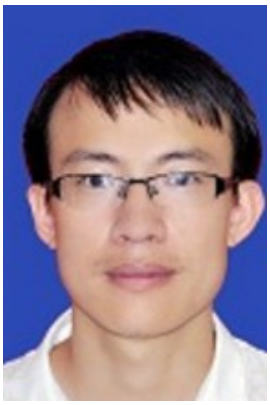
Yewang Chen: Yewang Chen received the B.S. and M.S. degrees from information management and computer science of Huaqiao University, Xiamen, China, in 2001 and 2006, respectively, and the Ph.D. degree from computer science of Fudan University, Shanghai, China, in 2009. Currently, he is a Lecturer in the School of Computer Science and Technology, Huaqiao University, Xiamen, China. His current research interests include natural language processing,machine learning and pattern recognition.

Shengyu Tang: Shengyu Tang received the B.S degree from College of Mathematics of Huaqiao University, Quanzhou, China, in 2012, and he is a postgraduate student in the School of Computer Science and Technology, Huaqiao University, Xiamen, China. His current research interests is machine learning and pattern recognition.

Nizar Bouguila: Nizar Bouguila received the engineer degree from the University of Tunis in 2000 the M.Sc. and Ph.D degrees from Sherbrooke University in 2002 and 2006, respectively, all in computer sciences. He is currently a professor within the Concordia Institute for Information Systems Engineering (CIISE) at Concordia University, Montreal, Qc, Canada. His current research interests include: Computer vision and pattern recognition, Machine learning and data mining, Image and signal processing, Statistical Process Control, 3D Graphics and Games.

Cheng Wang: Cheng Wang received the B.S. degrees from software engineering of Xidian University, Xian, China, in 2006 and received the Ph.D. degree in mechanics from Xi'an jiaotong University, China in 2012 respectively. Currently, he is an associate professor in the School of Computer Science and Technology, Huaqiao University, Xiamen, China. His research interests include signal processing and data mining.

Jixiang Du: Ji-Xiang Du took coursers B.Sc degree candidate in Vehicle Engineering, Hefei University of Technology, and obtained B.Sc. degree in July 1999. From September 1999 to July 2002, took courses as M.Sc. degree candidate in Vehicle Engineering, Hefei University of Technology, and obtained B.Sc. degree in July 2002. From February 2003 on, in pursuit for Ph.D. degree in Pattern Recognition and Intelligent System in University of Science and Technology of China (USTC), Hefei, China, and in December 2005, he received Ph.D. degree. Now, he is a professor at the College of Computer Science and Technology at Huaqiao University.



Hailin Li: Hailin Li received the B.S. degrees from Information and computing science of Jingdezhen Ceramic University, Jingdezhen, China, in 2006 and received the Ph.D. degree in Management Science and Engineering from Dalian University of Technology , China in 2012 respectively. Currently, he is an associate professor in the school of Business Administration, Huaqiao University, Quanzhou, China. His research interests include data mining and decision making.