On Metric DBSCAN with Low Doubling Dimension

Hu Ding[1] , **Fan Yang**[1] and **Mingyue Wang**[1]
[1]The School of Computer Science and Technology, University of Science and Technology of China
huding@ustc.edu.cn, {yang208,mywang}@mail.ustc.edu.cn

Abstract

The density based clustering method *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)* is a popular method for outlier recognition and has received tremendous attention from many different areas. A major issue of the original DBSCAN is that the time complexity could be as large as quadratic. Most of existing DBSCAN algorithms focus on developing efficient index structures to speed up the procedure in low-dimensional Euclidean space. However, the research of DBSCAN in high-dimensional Euclidean space or general metric spaces is still quite limited, to the best of our knowledge. In this paper, we consider the metric DBSCAN problem under the assumption that the inliers (excluding the outliers) have a low doubling dimension. We apply a novel randomized k-center clustering idea to reduce the complexity of range query, which is the most time consuming step in the whole DBSCAN procedure. Our proposed algorithms do not need to build any complicated data structures and are easy to implement in practice. The experimental results show that our algorithms can significantly outperform the existing DBSCAN algorithms in terms of running time.

1 Introduction

Density-based clustering is a fundamental topic in data analysis and has many applications in the areas, such as machine learning, data mining, and computer vision [Tan *et al.*, 2006]. Roughly speaking, the problem of density-based clustering aims to partition given data set into clusters where each cluster is a dense region in the space. The remaining data located in sparse regions are recognized as "outliers". Note that the given data set can be a set of points in a Euclidean space or any abstract metric space. *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)* [Ester *et al.*, 1996] is one of the most popular density-based clustering methods and has been implemented for solving many real-world problems. DBSCAN uses two parameters, "$\texttt{MinPts} \geq 1$" and "$\epsilon > 0$", to define the clusters (*i.e.,* the dense regions): a point p is a "core point" if it has at least `MinPts` neighbors within distance ϵ; a cluster is formed by a set of "connected" core points and some non-core points located in the boundary (which are named "border points"). We will provide the formal definition in Section 2.1.

A bottleneck of the original DBSCAN algorithm is that it needs to perform a range query for each data item, *i.e.,* computing the number of neighbors within the distance ϵ, and the overall time complexity can be as large as $O(n^2\beta)$ in the worst case, where n is the number of data items and β indicates the complexity for computing the distance between two items. For example, if the given data is a set of points in $\mathbb{R}^D$, we have $\beta = O(D)$. When n or β is large, the procedure of range query could make DBSCAN running very slowly.

Most existing DBSCAN algorithms focus on the case in low-dimensional Euclidean space. To speed up the step of range query, a natural idea is using some efficient index structures, such as R^*-tree [Beckmann *et al.*, 1990], though the overall complexity in the worst case is still $O(n^2)$ ($\beta = O(1)$ for low-dimensional Euclidean space). We refer the reader to the recent articles that systematically discussed this issue [Gan and Tao, 2015; Schubert *et al.*, 2017].

Using novel techniques from computational geometry, the running time of DBSCAN in $\mathbb{R}^2$ has been improved from $O(n^2)$ to $O(n \log n)$ by [de Berg *et al.*, 2017; Gunawan, 2013]. For the case in general D-dimensional Euclidean space, [Chen *et al.*, 2005] and [Gan and Tao, 2015] respectively provided the algorithms achieving sub-quadratic running times, where their complexities are both in the form of $O(n^{f(D)} \cdot D)$ with $f(D)$ being some function satisfying $\lim_{D\to\infty} f(D) = 2$. Namely, when the dimensionality D is high, their algorithms cannot gain a significant improvement over the straightforward implementation that has the complexity $O(n^2D)$. Recently, [Jang and Jiang, 2019] proposed a sampling based method, called *DBSCAN++*, to compute an approximate solution for DBSCAN; but their sample size $m \approx n$ when the dimensionality D is large (so there is no significant difference in terms of the time complexity if running the DBSCAN algorithm on the sample).

To speed up DBSCAN in practice, a number of approximate and distributed DBSCAN algorithms have been proposed, such as [Gan and Tao, 2015; Yang *et al.*, 2019; Lulli *et al.*, 2016; Song and Lee, 2018; Jang and Jiang, 2019]. To the best of our knowledge, most of these algorithms only consider instances in low-dimensional Euclidean space (rather than high-dimensional Euclidean space or abstract metric space), except [Lulli *et al.*, 2016; Yang *et al.*, 2019]. Lulli *et al.* presented an approximate, distributed algorithm for DBSCAN, as long as the distance function $d(\cdot,\cdot)$ is symmetric, that is, $d(x,y)=d(y,x)$ for any two points x and y. Very recently, Yang *et al.* showed an exact, distributed algorithm for DBSCAN in any abstract metric space; however, their method mainly focuses on how to ensure the load balancing and cut down the communication cost for distributed systems, rather than reducing the computational complexity of DBSCAN (actually, it directly uses the original DBSCAN algorithm of Ester *et al.* on each local machine).

1.1 Our Main Results

In this paper, we consider developing efficient algorithm for computing the exact solution of DBSCAN. As mentioned by Yang *et al.*, a wide range of real-world data cannot be represented in low-dimensional Euclidean space (*e.g.,* textual and image data can only be embedded into high-dimensional Euclidean space). Moreover, as mentioned in [Schubert *et al.*, 2017], the original DBSCAN was designed for general metrics, as long as the distance function of data items can be well defined. Thus it motivates us to consider the problem of DBSCAN in high-dimensional Euclidean space and general metric space.

We assume that the given data has a low "doubling dimension", which is widely used for measuring the intrinsic dimensions of datasets [Talwar, 2004] (we provide the formal definition in Section 2.2). The rationale behind this assumption is that many real-world datasets manifest low intrinsic dimensions [Belkin, 2003]. For example, image sets usually can be represented by a low dimensional manifold though the Euclidean dimension of the image vectors can be very high. We also note that it might be too strict to assume that the whole data set has a low doubling dimension, especially when it contains outliers. To make the assumption more general and capture a broader range of cases in practice, we only assume that the set of inliers has a constant doubling dimension while the outliers can scatter arbitrarily in the space. The assumption is formally stated in Definition 3. We focus on the following key question:

Is there any efficient algorithm being able to reduce the complexity of range query for DBSCAN, under such "low doubling dimension assumption"?

We are aware of several index structures in doubling metrics, *e.g.,* [Karger and Ruhl, 2002; Krauthgamer and Lee, 2004; Talwar, 2004]. However, these methods cannot handle the case with outliers. Moreover, they usually need to build very complicated data structures (*e.g.,* hierarchically well-separated trees) that are not quite efficient in practice. The popular *Locality Sensitive Hashing* [Andoni and Indyk, 2008] is efficient for nearest neighbor search in high dimensions, but not quite appropriate to solve the range query step for DBSCAN (especially for exact DBSCAN).

We observe that the well-known k-center clustering procedure can be incorporated into the DBSCAN algorithm to reduce the complexity of the range query procedure in doubling metric. However, we cannot directly apply the ordinary k-center clustering method (*e.g.,* [Gonzalez, 1985]) since the outliers may not satisfy the low doubling dimension condition. Instead, we show that a randomized k-center clustering algorithm proposed by [Ding *et al.*, 2019] can efficiently remedy this issue, though we still need to develop some new ideas to apply their algorithm to solve the problem of DBSCAN.

The rest of the paper is organized as follows. In Section 2, we show the formal definitions of doubling dimension and DBSCAN, and briefly introduce the randomized k-center clustering algorithm from [Ding *et al.*, 2019]. In Section 3, we propose and analyze our algorithms for reducing the complexity of range query in detail. Finally, we compare the experimental performances of our algorithms and several well-known baseline DBSCAN algorithms on both synthetic and real datasets.

2 Preliminaries

Throughout this paper, we use (X,d) to denote the metric space where $d(\cdot,\cdot)$ is the distance function on X. Let $|X|=n$. We also assume that it takes $O(\beta)$ time to compute $d(p,q)$ for any p, $q\in X$. Let $\mathbb{B}(x,r)$ be the ball centered at point $x\in X$ with radius $r\geq 0$ in the metric space.

2.1 DBSCAN

We introduce the formal definition of DBSCAN. Given two parameters $\epsilon>0$ and $\texttt{MinPts}\in\mathbb{Z}^+$, DBSCAN divides the points of X into three classes:

1. p is a **core point**, if $|\mathbb{B}(p,\epsilon)\cap X|\geq\texttt{MinPts}$;
2. p is a **border point**, if p is not a core point but $p\in\mathbb{B}(q,\epsilon)$ of some core point q;
3. all the other points are **outliers**.

To define a cluster of DBSCAN, we need the following concept.

Definition 1 (Density-reachable)**.** *We say a point $p\in X$ is density-reachable from a core point q, if there exists a sequence of points $p_1,p_2,\cdots,p_t\in X$ such that:*

- *$p_1=q$ and $p_t=p$;*
- *$p_1,\cdots,p_{t-1}$ are all core points;*
- *$p_{i+1}\in\mathbb{B}(p_i,\epsilon)$ for each $i=1,2,\cdots,t-1$.*

If one arbitrarily picks a core point q, then the corresponding DBSCAN cluster defined by q is

$$\{p\mid p\in X \text{ and p is density-reachable from q}\}. \qquad (1)$$

Actually, we can imagine that the set X form a directed graph: any two points p and $p'\in P$ are connected by a directed edge $p\to p'$, if p is a core point and $p'\in\mathbb{B}(p,\epsilon)$. From (1), we know that the cluster is the maximal subset containing the points who are density-reachable from q. The cluster may contain both core and border points. It is easy to know that for any two core point q and q', they define exactly the same cluster if they are density-reachable from each other (*i.e.,* there exists a path from q to q' and vice versa). Therefore, a cluster of DBSCAN is uniquely defined by any of its core points. Moreover, a border point could belong to multiple clusters and an outlier cannot belong to any cluster. The goal of DBSCAN is to discover these clusters and outliers.

For convenience, we use X_{in} and X_{out} to denote the sets of inliers (including the core points and border points) and outliers, respectively.

2.2 Doubling Metrics

Definition 2 (Doubling Dimension)**.** *The doubling dimension of a metric space (X, d) is the smallest number $\rho > 0$, such that for any $p \in X$ and $r \geq 0$, $X \cap \mathbb{B}(p, 2r)$ is always covered by the union of at most 2^ρ balls with radius r.*

Roughly speaking, the doubling dimension describes the expansion rate of the metric space. We have the following property of doubling metrics from [Talwar, 2004; Krauthgamer and Lee, 2004] that can be proved by recursively applying Definition 2.

Proposition 1. *Let (X, d) be a metric space with a doubling dimension $\rho > 0$. If $Y \subseteq X$ and its aspect ratio is $\alpha = \frac{\max_{y,y' \in Y} d(y,y')}{\min_{y,y' \in Y} d(y,y')}$, then $|Y| \leq 2^{\rho \lceil \log \alpha \rceil}$.*

For our DBSCAN problem, we adopt the following assumption from [Ding *et al.*, 2019].

Definition 3 (Low Doubling Dimension Assumption)**.** *Given an instance $(X, \epsilon, \texttt{MinPts})$ of DBSCAN, we assume that the metric space (X_{in}, d),* i.e., *the metric space formed by the set of core points and border points, has a constant doubling dimension $\rho > 0$. The set X_{out} of outliers can scatter arbitrarily in the space.*

2.3 The Randomized Gonzalez's Algorithm

k-center clustering is one of the most fundamental clustering problems [Gonzalez, 1985]. Given a metric space (X, d) with $|X| = n$, the problem of k-center clustering is to find k balls to cover the whole X and minimize the maximum radius. For the sake of completeness, let us briefly introduce the algorithm of [Gonzalez, 1985] for k-center clustering first. Initially, it arbitrarily selects a point from X, and iteratively selects the following $k-1$ points, where each j-th step ($2 \leq j \leq k$) chooses the point having the largest minimum distance to the already selected $j-1$ points; finally, each point of X is assigned to its nearest neighbor of these selected k points. It can be proved that this greedy strategy yields a 2-approximation of k-center clustering (*i.e.,* the maximum radius of the obtained k balls is at most twice as large as the optimal radius).

[Ding *et al.*, 2019] presented a randomized version of the Gonzalez's algorithm for solving k-center clustering with outliers. Let $z \geq 1$ be the pre-specified number of outliers, and the problem of k-center with outliers is to find k balls to cover $n - z$ points of X and minimize the maximum radius. This problem is much more challenging than the ordinary k-center clustering, since we do not know which points are the outliers in advance and there are an exponentially large number $\binom{n}{z}$ of different possible cases. Note that other algorithms for k-center clustering with outliers, such as [Charikar *et al.*, 2001; Chakrabarty *et al.*, 2016], take at least quadratic time complexity. The key idea of [Ding *et al.*, 2019] is to replace each step of Gonzalez's algorithm, choosing the farthest point to the set of already selected points, by taking a random sample from the farthest $(1+\delta)z$ points, where $\delta > 0$ is a small parameter; after $O(k)$ steps, with constant probability, the algorithm yields a set of $O(\frac{k}{\delta})$ balls covering at least $n - (1+\delta)z$ points of X and the resulting radius is at most twice as large as the optimal radius. For example, if we set $\delta = 1$, the algorithm will yield $O(k)$ balls covering at least $n - 2z$ points. The formal result is presented in Theorem 1. We omit the detailed proof from [Ding *et al.*, 2019].

Algorithm 1 The Randomized Gonzalez's algorithm

```
Input: An instance $(X, d)$ of $k$-center clustering with $z$ outliers, and $|X| = n$; the parameters $\delta > 0$, $\eta \in (0, 1)$, and $t \in \mathbb{Z}^+$.
  1. Let $\gamma = z/n$ and initialize a set $E = \emptyset$.
  2. Initially, $j = 1$; randomly select $\frac{1}{1-\gamma}\log\frac{1}{\eta}$ points from $X$ and add them to $E$.
  3. Run the following steps until $j = t$:
      (a) $j = j+1$ and let $Q_j$ be the farthest $(1+\delta)z$ points of $X$ to $E$ (for each point $p \in X$, its distance to $E$ is $\min_{q \in E} d(p, q)$).
      (b) Randomly select $\frac{1+\delta}{\delta}\log\frac{1}{\eta}$ points from $Q_j$ and add them to $E$.
Output $E$.
```

Theorem 1. *Let r_{opt} be the optimal radius of the instance (X, d) of k-center clustering with z outliers. If we set $t = \frac{k+\sqrt{k}}{1-\eta}$ in Algorithm 1, with probability at least $(1-\eta)(1 - e^{-\frac{1-\eta}{4}})$, the union of the balls*

$$\cup_{c \in E} \mathbb{B}(c, 2r_{opt}) \tag{2}$$

covers at least $n - (1+\delta)z$ points of X.

If $\frac{1}{\eta}$ and $\frac{1}{1-\gamma}$ are constant numbers, the number of balls (*i.e.,* $|E|$) is $O(\frac{k}{\delta})$ and the success probability is constant. In each round of Step 3, there are $\frac{1+\delta}{\delta}\log\frac{1}{\eta} = O(\frac{1}{\delta})$ new points added to E, thus it takes $O(\frac{1}{\delta}n\beta)$ time to update the distances from the points of X to E; to select the set Q_j, we can apply the linear time selection algorithm [Blum *et al.*, 1973]. Overall, the running time of Algorithm 1 is $O(\frac{k}{\delta}n\beta)$. If the given instance is in $\mathbb{R}^D$, the running time will be $O(\frac{k}{\delta}nD)$.

3 Our Algorithms and Theoretical Analysis

In this section, we present two efficient algorithms for solving DBSCAN under the assumption of Definition 3.

3.1 The First DBSCAN Algorithm

Our first DBSCAN algorithm (Algorithm 2) contains two parts. To better understand our algorithm, we briefly introduce the high-level idea below. For convenience, we use $d(U, V)$ to denote the minimum distance between two sets U and $V \subset X$, *i.e.,* $\min\{d(u, v) \mid u \in U, v \in V\}$.

Part (i). First, we run Algorithm 1 to conduct a coarse partition on the given set X. We view X as an instance of k-center clustering with z outliers where $z = |X_{out}|$ (recall X_{out} is the set of outliers defined in Section 2.1). However, we cannot directly run Algorithm 1 since the values of t and z are not given. Actually, we can avoid to set the value of t via a slight modification on Algorithm 1; we just need to iteratively run Step 3 until $d(Q_j, E) \leq r$, where r is a parameter that will be discussed in our experiments. For the parameter z, we cannot obtain its exact value before running DBSCAN. In practice, the number z is much smaller than n; so we assume that an upper bound $\tilde{z}$ of z is available (*e.g.,* we can estimate the upper bound of z by taking a small sample before running the algorithm). In each round of Step 3 of Algorithm 1, we update the distances from X to E. As a by-product, we can store the following information after running Algorithm 1:

- the pairwise distances of E: $\{d(c, c') \mid c, c' \in E\}$;
- for each $p \in X$, denote by c_p its nearest neighbor in E.

If we simply set $\delta = 1$ in Algorithm 1, Theorem 1 implies that at least $n - 2\tilde{z}$ points of X are covered by the balls $\cup_{c \in E} \mathbb{B}(c, r)$. We denote the set of points outside the balls as $X_{\tilde{z}}$, and obviously $|X_{\tilde{z}}|$ is no larger than $2\tilde{z}$ by Theorem 1.

Part (ii). For the second part, we check each point $p \in X$ and determine its label to be "core point", "border point", or "outlier". According to the formulation of DBSCAN, we need to compute the size $|X \cap \mathbb{B}(p, \epsilon)|$. In general, this procedure will take $O(n\beta)$ time and the whole running time will be $O(n^2\beta)$. To reduce the time complexity, we can take advantage of the information obtained in Part (i). Since $|X_{\tilde{z}}| \leq 2\tilde{z}$ and $\tilde{z}$ usually is much smaller than n, we focus on the part $X \setminus X_{\tilde{z}}$ containing the majority of the points in X. Let p be any point in $X \setminus X_{\tilde{z}}$ and c_p be its nearest neighbor in E. Let

$$A_p = \{c \mid c \in E, d(c, c_p) \leq 2r + \epsilon\}, \tag{3}$$

and we can quickly obtain the set A_p since the pairwise distances of E are stored in Part (i). Lemma 1 guarantees that we only need to check the local region, the balls $\bigcup_{c \in A_p} \mathbb{B}(c, r)$ and $X_{\tilde{z}}$, instead of the whole X, for computing the size $|X \cap \mathbb{B}(p, \epsilon)|$; further, Lemma 2 shows that the size of A_p is bounded. See Figure 1 for an illustration.

Lemma 1. *If $p \in X \setminus X_{\tilde{z}}$, we have $X \cap \mathbb{B}(p, \epsilon) \subset \Big(\bigcup_{c \in A_p} \big(X \cap \mathbb{B}(c, r)\big)\Big) \bigcup X_{\tilde{z}}$.*

Proof. Let q be any point in $X \setminus X_{\tilde{z}}$. If $d(c_p, c_q) > 2r + \epsilon$, *i.e.,* $q \in \bigcup_{c \notin A_p} \mathbb{B}(c, r)$, by using the triangle inequality, we have

$$\begin{aligned} d(p, q) &\geq d(c_p, c_q) - d(p, c_p) - d(q, c_q) \\ &> 2r + \epsilon - r - r > \epsilon. \end{aligned} \tag{4}$$

Therefore, $q \notin X \cap \mathbb{B}(p, \epsilon)$. That is, $X \cap \mathbb{B}(p, \epsilon) \subset$

$$X \setminus \Big(\bigcup_{c \notin A_p} \mathbb{B}(c, r)\Big) = \Big(\bigcup_{c \in A_p} \big(X \cap \mathbb{B}(c, r)\big)\Big) \bigcup X_{\tilde{z}}. \tag{5}$$

So we complete the proof. □

Now, we consider the size of A_p. Recall the construction process of E in Algorithm 1. Initially, Algorithm 1 adds $\frac{1}{1-\gamma} \log \frac{1}{\eta}$ points to E; in each round of Step 3, it adds

Algorithm 2 METRIC DBSCAN ALGORITHM

```
Input: An instance $(X, d)$ of DBSCAN, and the parameters $\epsilon, r > 0$, MinPts, $\tilde{z} \in \mathbb{Z}^+$.
1. Run Algorithm 1 with setting $\delta = 1$, and terminate the loop of Step 3 when $d(Q_j, E) \leq r$.
    (a) Store the set $\mathcal{D}_E = \{d(c, c') \mid c, c' \in E\}$.
    (b) For each $p \in X$, denote by $c_p$ its nearest neighbor in $E$.
    (c) If the instance is in Euclidean space: for each $c \in E$ we build a $R^*$-tree for the points inside $\mathbb{B}(c, r)$ (if a point $p$ is covered by multiple balls, we assign it to the ball of the center $c_p$).
2. For each $p \in X$, check whether it is a core point:
    (a) if $p \in X_{\tilde{z}}$, directly compute the set $X \cap \mathbb{B}(p, \epsilon)$ by scanning $X$;
    (b) else, obtain the set $A_p = \{c \mid c \in E, d(c, c_p) \leq 2r + \epsilon\}$ from $\mathcal{D}_E$, and compute the set $X \cap \mathbb{B}(p, \epsilon)$ by checking the points in $\Big(\bigcup_{c \in A_p} \big(X \cap \mathbb{B}(c, r)\big)\Big) \bigcup X_{\tilde{z}}$ (inside each $\mathbb{B}(c, r)$, we use the $R^*$-tree built in Step 1(c) if the instance is in Euclidean space).
3. Join the core points into clusters by running the standard DBSCAN procedure [Schubert et al., 2017].
```

$2 \log \frac{1}{\eta}$ points to E (since we set $\delta = 1$). For convenience, we let $\texttt{num} = \max\{\frac{1}{1-\gamma} \log \frac{1}{\eta}, 2 \log \frac{1}{\eta}\}$. So we can imagine that E consists of multiple "batches" where each batch contains $\leq \texttt{num}$ points. Also, since we terminate Step 3 when $d(Q_j, E) \leq r$, any two points from different batches should have distance at least r. We consider the batches having non-empty intersection with A_p. For ease of presentation, we denote these batches as $B_1, B_2, \cdots, B_m$. Further, we label each batch B_j by two colors for $1 \leq j \leq m$:

- "red" if $B_j \cap A_p \cap X_{in} \neq \emptyset$;
- "blue" otherwise.

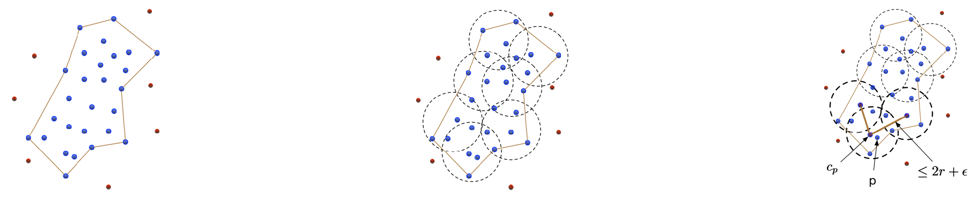
Without loss of generality, we assume that the batches $\{B_j \mid 1 \leq j \leq m'\}$ are red, and the batches $\{B_j \mid m'+1 \leq j \leq m\}$ are blue. To bound the size of A_p, we divide it to two parts $A_p \setminus X_{in}$ and $A_p \cap X_{in}$. Recall X_{in} is the set of core points and border points defined in Section 2.1. It is easy to know that $A_p \setminus X_{in} \subset X_{out}$, *i.e.,*

$$|A_p \setminus X_{in}| \leq |X_{out}| \leq \tilde{z}. \tag{6}$$

Also, $A_p \cap X_{in}$ belongs to the union of the red batches, and therefore $|A_p \cap X_{in}| \leq m' \times \texttt{num}$. So we focus on the value of m' below.

Lemma 2. *The number of red batches, m', is at most $2^{\rho \lceil \log \alpha \rceil}$, where $\alpha = 4 + 2\frac{\epsilon}{r}$. That is, $|A_p \cap X_{in}| \leq 2^{\rho \lceil \log \alpha \rceil} \times \texttt{num}$. For simplicity, if we assume $\frac{1}{\eta}$ and $\frac{1}{1-\gamma}$ are constant numbers in Algorithm 1, then*

$$|A_p \cap X_{in}| \leq O(2^{\rho \lceil \log \alpha \rceil}).$$



(a) (b) (c)

Figure 1: (a) indicates an instance of DBSCAN, where the blue points are inliers (including the core points and border points) and the red points are outliers; (b) shows the balls obtained in Algorithm 1; (c) shows an example of computing the set $X \cap \mathbb{B}(p, \epsilon)$ for a point p, where we just need to check the two neighbor balls of c_p.

Proof. For each red batch B_j, we arbitrarily pick one point, say c_j, from $B_j \cap A_p \cap X_{in}$, and let

$$H = \{c_j \mid 1 \leq j \leq m'\}. \tag{7}$$

First, we know $H \subset X_{in}$. Second, because the minimum pairwise distance of H is at least r (since any two points of H come from different batches) and the maximum pairwise distance of H

$$\begin{aligned}\max_{c_j, c_{j'} \in H} d(c_j, c_{j'}) &\leq \max_{c_j, c_{j'} \in H} \big(d(c_j, c_p) + d(c_p, c_{j'})\big) \\ &\leq 2(2r + \epsilon) = 4r + 2\epsilon, \end{aligned} \tag{8}$$

the aspect ratio of H is no larger than $\alpha = 4 + 2\frac{\epsilon}{r}$. Note the doubling dimension of (X_{in}, d) is ρ according to Definition 3. Through Proposition 1, we have $|H| \leq 2^{\rho \lceil \log \alpha \rceil}$.

So the number of red batches $m' = |H| \leq 2^{\rho \lceil \log \alpha \rceil}$; each batch has size $\leq$ `num`. Overall, we have $|A_p \cap X_{in}| \leq 2^{\rho \lceil \log \alpha \rceil} \times$ `num`. $\square$

The overall complexity. Our goal is to reduce the search region for each query step, *e.g.*, we only need to consider a local region as illustrated in Figure 1. If the points distribution within dense region is very imbalanced (that is unusual in practice), the complexity could be as bad as quadratic. Actually, it is unlikely to design an exact DBSCAN algorithm with linear or nearly linear complexity [Gan and Tao, 2015].

3.2 An Alternative Approach

In this section, we provide a modified version of our first DBSCAN algorithm. In Lemma 2, we cannot directly use Proposition 1 to bound the size of A_p, because the points inside the same batch could have pairwise distance less than r; therefore, we can only bound the number of red batches. To remedy this issue, we perform the following "filtration" operation when adding each batch to E in Algorithm 1.

Filtration. For each batch of E, we compute a connection graph: each point of the batch represents a vertex, and any two vertices are connected by an edge if their pairwise distance is smaller than r. Then, we compute a maximal independent set (not necessary the maximum independent set) of the graph, and only add this independent set to E instead of the whole batch. See Figure 2 as an illustration.

Obviously, this filtration operation guarantees that the pairwise distance of any two points in E is at least r. Since each batch has size `num`, it takes $O\big(\texttt{num}^2 \beta\big)$ time to compute

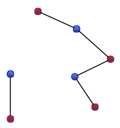


Figure 2: The batch contains 7 points, and any two points are connected by an edge if their distance is smaller than r; we can pick the 4 red points as the maximal independent set.

the maximal independent set. Moreover, since the set E has fewer points, we need to modify the result stated in Theorem 1. Let p be any point of X having distance no larger than r to E in the original Algorithm 1. After performing the filtration operation, we know $d(p, E) \leq 2r$ due to the triangle inequality. As a consequence, the set $X \setminus X_{\tilde{z}}$ is covered by the balls $\cup_{c \in E} \mathbb{B}\big(c, 2r\big)$ (instead of $\cup_{c \in E} \mathbb{B}\big(c, r\big)$). Let

$$A'_p = \{c \mid c \in E, d(c, c_p) \leq 4r + \epsilon\}. \tag{9}$$

The aspect ratio of A'_p is no larger than $\frac{2(4r+\epsilon)}{r} = 8 + 2\frac{\epsilon}{r}$. Using the similar ideas for proving Lemma 1 and 2, we obtain the following results.

Lemma 3. *If $p \in X \setminus X_{\tilde{z}}$, we have $X \cap \mathbb{B}(p, \epsilon) \subset \Big(\bigcup_{c \in A'_p} \big(X \cap \mathbb{B}(c, 2r)\big)\Big) \bigcup X_{\tilde{z}}$.*

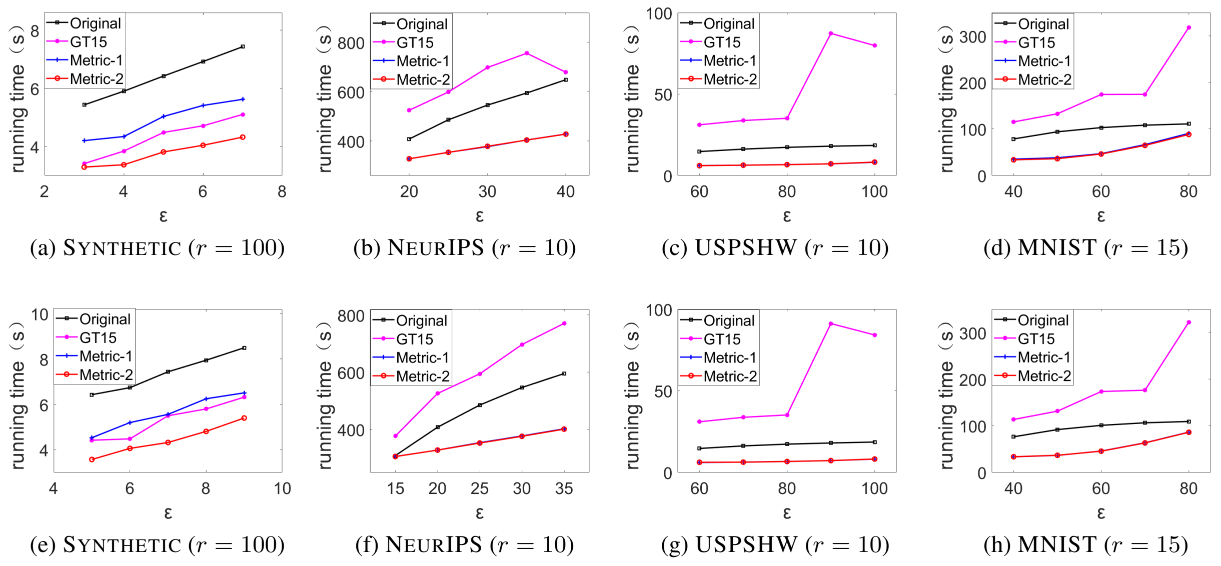
Lemma 4. *$|A'_p \setminus X_{in}| \leq \tilde{z}$ and $|A'_p \cap X_{in}| \leq 2^{\rho \lceil \log \alpha \rceil}$, where $\alpha = 8 + 2\frac{\epsilon}{r}$.*

Remark 1. *Comparing with the size $|A_p \cap X_{in}|$ in Lemma 2, we remove the hidden constant "`num`" but increase the value of α from $4 + 2\frac{\epsilon}{r}$ to $8 + 2\frac{\epsilon}{r}$. So, we cannot directly compare the sizes $|A_p \cap X_{in}|$ and $|A'_p \cap X_{in}|$ in general. In Section 4, we implement the two algorithms, and investigate their experimental performances.*

4 Experiments

All the experimental results were obtained on a Windows 10 workstation equipped with an Intel core $i5$-8400 processor and 8GB RAM. We compare the performances of the following four DBSCAN algorithms in terms of running time:

- **ORIGINAL**: the original DBSCAN [Ester *et al.*, 1996] that uses R^*-tree as the index structure.
- **GT15**: the grid-based exact DBSCAN algorithm proposed in [Gan and Tao, 2015].



(a) SYNTHETIC ($r = 100$) (b) NEURIPS ($r = 10$) (c) USPSHW ($r = 10$) (d) MNIST ($r = 15$)

(e) SYNTHETIC ($r = 100$) (f) NEURIPS ($r = 10$) (g) USPSHW ($r = 10$) (h) MNIST ($r = 15$)

Figure 3: Running times with `MinPts` $= \frac{1}{1000} \cdot n$ (the first row) and `MinPts` $= \frac{2}{1000} \cdot n$ (the second row).

Dataset	#Instances	#Attributes	Type
SYNTHETIC	20000	500-3000	Synthetic
NEURIPS	11463	5811	Text
USPSHW	7291	256	Image
MNIST	10000	784	Image

Table 1: The datasets.

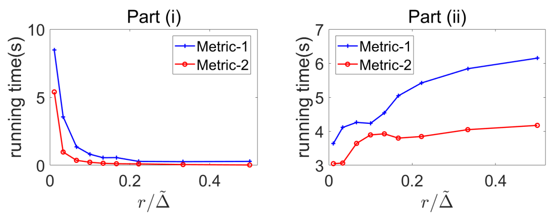


Figure 4: The running time in Part (i) and Part (ii).

- **METRIC-1**: our first DBSCAN algorithm proposed in Section 3.1.
- **METRIC-2**: the alternative DBSCAN algorithm proposed in Section 3.2.

For the first two algorithms, we use the implementations in C++ from [Gan and Tao, 2015]. Our algorithms **METRIC-1** and **METRIC-2** are also implemented in C++. Note that all of these four algorithms return the exact DBSCAN solution; we do not consider the approximate DBSCAN algorithms that are out of the scope of this paper.

We evaluated our methods on both synthetic and real datasets where the details are shown in Table 1. We generated 6 synthetic datasets. For each synthetic dataset, we randomly generate $n = 20000$ points in $\mathbb{R}^2$, and then locate them to a higher dimensional space $\mathbb{R}^D$ with some random noise; the dimension D ranges from 500 to 3000. **NEURIPS** [Perrone *et al.*, 2017] contains $n = 11463$ word vectors of the full texts of the NeurIPS conference papers published in 1987-2015. **USPSHW** [Hull, 1994] contains $n = 7291$ 16×16 pixel handwritten letter images. **MNIST** [LeCun *et al.*, 1998] contains $n = 10000$ handwritten digit images from 0 to 9, where each image is represented by a 784-dimensional vector.

The results. We validate the influence of the value of r to the running times of METRIC-1 and METRIC-2. We focus on the SYNTHETIC datasets. To determine the value of r, we first estimate the diameter Δ, the largest pairwise distance, of the dataset. Obviously, it takes at least quadratic time to achieve the exact value of Δ; instead, we just arbitrarily select one point and pick its farthest point from the dataset, where the obtained value $\tilde{\Delta}$ is between $\Delta/2$ and Δ. We set $\tilde{z} = 200$ (*i.e.,* $1\%n$) and vary the ratio $r/\tilde{\Delta}$ in 0-0.5. The running times with respect to Part (i) and Part (ii) (described in Section 3.1) are shown in Figure 4 separately. As $r/\tilde{\Delta}$ increases, the running time of Part (i) (*resp.,* Part (ii)) decreases (*resp.,* increases). The overall running time (of the two parts) reaches the lowest value when $r/\tilde{\Delta}$ is around 0.1.

Further, we set the value `MinPts` $= \frac{1}{1000}n$ and $\frac{2}{1000}n$ for each dataset and show the running times in Figure 3. We can see that our METRIC-2 achieves the lowest running times on SYNTHETIC; the running times of METRIC-1 and METRIC-2 are very close on the three real datasets; our both algorithms significantly outperform the two baseline algorithms in terms of running time.

5 Future Work

In this paper, we consider the problem of DBSCAN with low doubling dimension. A few directions deserve to be studied in future work, such as other density based clustering and outlier recognition problems under the assumption of Definition 3.

References

[Andoni and Indyk, 2008] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[Beckmann *et al.*, 1990] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, USA, May 23-25, 1990*, pages 322–331, 1990.

[Belkin, 2003] Mikhail Belkin. *Problems of learning on manifolds*. The University of Chicago, 2003.

[Blum *et al.*, 1973] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

[Chakrabarty *et al.*, 2016] Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k-center problem. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 67:1–67:15, 2016.

[Charikar *et al.*, 2001] Moses Charikar, Samir Khuller, David M Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 642–651, 2001.

[Chen *et al.*, 2005] Danny Z. Chen, Michiel H. M. Smid, and Bin Xu. Geometric algorithms for density-based data clustering. *Int. J. Comput. Geometry Appl.*, 15(3):239–260, 2005.

[de Berg *et al.*, 2017] Mark de Berg, Ade Gunawan, and Marcel Roeloffzen. Faster DBScan and HDBScan in low-dimensional euclidean spaces. In *28th International Symposium on Algorithms and Computation, ISAAC 2017*, pages 25:1–25:13, 2017.

[Ding *et al.*, 2019] Hu Ding, Haikuo Yu, and Zixiu Wang. Greedy strategy works for k-center clustering with outliers and coreset construction. In *27th Annual European Symposium on Algorithms, ESA 2019*, pages 27:1–27:16, 2019.

[Ester *et al.*, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the ACM SIGKDD*, pages 226–231, 1996.

[Gan and Tao, 2015] Junhao Gan and Yufei Tao. DBSCAN revisited: mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 519–530. ACM, 2015.

[Gonzalez, 1985] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[Gunawan, 2013] Ade Gunawan. A faster algorithm for DBSCAN. *Master's thesis. Eindhoven University of Technology, the Netherlands*, 2013.

[Hull, 1994] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, 1994.

[Jang and Jiang, 2019] Jennifer Jang and Heinrich Jiang. DBSCAN++: towards fast and scalable density clustering. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 3019–3029, 2019.

[Karger and Ruhl, 2002] David R Karger and Matthias Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM, 2002.

[Krauthgamer and Lee, 2004] Robert Krauthgamer and James R Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Society for Industrial and Applied Mathematics, 2004.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lulli *et al.*, 2016] Alessandro Lulli, Matteo Dell'Amico, Pietro Michiardi, and Laura Ricci. NG-DBSCAN: scalable density-based clustering for arbitrary data. *Proceedings of the VLDB Endowment*, 10(3):157–168, 2016.

[Perrone *et al.*, 2017] Valerio Perrone, Paul A Jenkins, Dario Spano, and Yee Whye Teh. Poisson random fields for dynamic feature models. *The Journal of Machine Learning Research*, 18(1):4626–4670, 2017.

[Schubert *et al.*, 2017] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. DBSCAN revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):19, 2017.

[Song and Lee, 2018] Hwanjun Song and Jae-Gil Lee. RP-DBSCAN: A superfast parallel dbscan algorithm based on random partitioning. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1173–1187. ACM, 2018.

[Talwar, 2004] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290, 2004.

[Tan *et al.*, 2006] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. 2006.

[Yang *et al.*, 2019] Keyu Yang, Yunjun Gao, Rui Ma, Lu Chen, Sai Wu, and Gang Chen. DBSCAN-MS: Distributed density-based clustering in metric spaces. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1346–1357. IEEE, 2019.