DBSCAN++: Towards fast and scalable density clustering

Jennifer Jang [1] **Heinrich Jiang** [2]

Abstract

DBSCAN is a classical density-based clustering procedure with tremendous practical relevance. However, DBSCAN implicitly needs to compute the empirical density for each sample point, leading to a quadratic worst-case time complexity, which is too slow on large datasets. We propose DBSCAN++, a simple modification of DBSCAN which only requires computing the densities for a chosen subset of points. We show empirically that, compared to traditional DBSCAN, DBSCAN++ can provide not only competitive performance but also added robustness in the bandwidth hyperparameter while taking a fraction of the runtime. We also present statistical consistency guarantees showing the trade-off between computational cost and estimation rates. Surprisingly, up to a certain point, we can enjoy the same estimation rates while lowering computational cost, showing that DBSCAN++ is a sub-quadratic algorithm that attains minimax optimal rates for level-set estimation, a quality that may be of independent interest.

1. Introduction

Density-based clustering algorithms such as Mean Shift (Cheng, 1995) and DBSCAN (Ester et al., 1996) have made a large impact on a wide range of areas in data analysis, including outlier detection, computer vision, and medical imaging. As data volumes rise, non-parametric unsupervised procedures are becoming ever more important in understanding large datasets. Thus, there is an increasing need to establish more efficient versions of these algorithms. In this paper, we focus on improving the classical DBSCAN procedure.

It was long believed that DBSCAN had a runtime of $O(n \log n)$ until it was proven to be $O(n^2)$ in the worst case by Gan and Tao (2015). They showed that while DB-

[1]Uber [2]Google Research. Correspondence to: Jennifer Jang <j.jang42@gmail.com>.

SCAN can run in $O(n \log n)$ when the dimension is at most 2, it quickly starts to exhibit quadratic behavior in high dimensions and/or when n becomes large. In fact, we show in Figure 1 that even with a simple mixture of 3-dimensional Gaussians, DBSCAN already starts to show quadratic behavior.

The quadratic runtime for these density-based procedures can be seen from the fact that they implicitly must compute density estimates for each data point, which is linear time in the worst case for each query. In the case of DBSCAN, such queries are proximity-based. There has been much work done in using space-partitioning data structures such as KD-Trees (Bentley, 1975) and Cover Trees (Beygelzimer et al., 2006) to improve query times, but these structures are all still linear in the worst-case. Another line of work that has had practical success is in approximate nearest neighbor methods (e.g. Indyk and Motwani (1998); Datar et al. (2004)) which have sub-linear queries, but such methods come with few approximation guarantees.

DBSCAN proceeds by computing the empirical densities for each sample point and then designating points whose densities are above a threshold as *core-points*. Then, a neighborhood graph of the core-points is constructed, and the clusters are assigned based on the connected components.

In this paper, we present DBSCAN++, a step towards a fast and scalable DBSCAN. DBSCAN++ is based on the observation that we only need to compute the density estimates for a subset m of the n data points, where m can be much smaller than n, in order to cluster properly. To choose these m points, we provide two simple strategies: uniform and greedy K-center-based sampling. The resulting procedure has $O(mn)$ worst-case runtime.

We show that with this modification, we still maintain statistical consistency guarantees. We show the trade-off between computational cost and estimation rates. Interestingly, up to a certain point, we can enjoy the same minimax-optimal estimation rates attained by DBSCAN while making m (instead of the larger n) empirical density queries, thus leading to a sub-quadratic procedure. In some cases, we saw that our method of limiting the number of core points can act as a regularization, thus reducing the sensitivity of classical DBSCAN to its parameters.
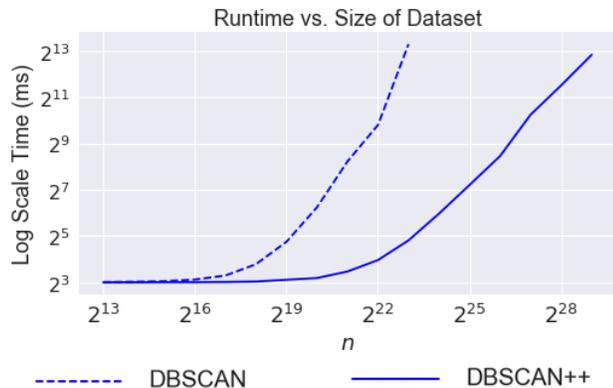
Figure 1. Runtime (seconds) vs dataset size to cluster a mixture of four 3-dimensional Gaussians. Using Gaussian mixtures, we see that DBSCAN starts to show quadratic behavior as the dataset gets large. After 10^6 points, DBSCAN ran too slowly and was terminated after 3 hours. This is with only 3 dimensions.

We show on both simulated datasets and real datasets that DBSCAN++ runs in a fraction of the time compared to DBSCAN, while giving competitive performance and consistently producing more robust clustering scores across hyperparameter settings.

2. Related Works

There has been much work done on finding faster variants of DBSCAN. We can only highlight some of these works here. One approach is to speed up the nearest neighbor queries that DBSCAN uses (Huang and Bian, 2009; Vijay-alaksmi and Punithavalli, 2012; Kumar and Reddy, 2016), including with approximate nearest neighbors methods (Wu et al., 2007). Another approach is to find a set of "leader" points that still preserve the structure of the original data set and then identify clusters based on the clustering of these "leader" points (Geng et al., 2000; Viswanath and Pinkesh, 2006; Viswanath and Babu, 2009). Our approach of finding core points is similar but is simpler and comes with theoretical guarantees. Liu (2006) modified DBSCAN by selecting clustering seeds among the unlabeled core points in an orderly manner in order to reduce computation time in regions that have already been clustered. Other heuristics include (Borah and Bhattacharyya, 2004; Zhou et al., 2000b; Patwary et al., 2012; Kryszkiewicz and Lasek, 2010).

There are also numerous approaches based on parallel computing such as (Xu et al., 1999; Zhou et al., 2000a; Arlia and Coppola, 2001; Brecheisen et al., 2006; Chen et al., 2010; Patwary et al., 2012; Götz et al., 2015) including map-reduce based approaches (Fu et al., 2011; He et al., 2011; Dai and Lin, 2012; Noticewala and Vaghela, 2014). Then there are distributed approaches to DBSCAN where data is partitioned across different locations and there may be communication cost constraints (Januzaj et al., 2004b;a; Liu

et al., 2012; Neto et al., 2015; Lulli et al., 2016). It is also worth mentioning Andrade et al. (2013), who presented a GPU implementation of DBSCAN that can be over 100x faster than sequential DBSCAN. In this paper, we assume a single processor although extending our approach to the parallel or distributed settings could be a future research direction.

We now discuss the theoretical work done for DBSCAN. Despite the practical significance of DBSCAN, its statistical properties has only been explored recently (Sriperumbudur and Steinwart, 2012; Jiang, 2017a; Wang et al., 2017; Steinwart et al., 2017). Such analyses make use of recent developments in topological data analysis to show that DBSCAN estimates the connected components of a level-set of the underlying density.

It turns out there has been a long history in estimating the level-sets of the density function (Hartigan, 1975; Tsybakov et al., 1997; Singh et al., 2009; Rigollet et al., 2009; Rinaldo and Wasserman, 2010; Chaudhuri and Dasgupta, 2010; Steinwart, 2011; Balakrishnan et al., 2013; Chaudhuri et al., 2014; Jiang, 2017b; Chen et al., 2017). However, most of these methods have little practical value (some are unimplementable), and DBSCAN is one of the only practical methods that is able to attain the strongest guarantees, including finite-sample Hausdorff minimax optimal rates. In fact the only previous method that was shown to attain such guarantees was the impractical histogram-based method of Singh et al. (2009), until Jiang (2017a) showed that DBSCAN attained almost identical guarantees. This paper shows that DBSCAN++ can attain similar guarantees while being sub-quadratic in computational complexity as well as the precise trade-off in estimation rates for further computational speedup.

3. Algorithm

We have n i.i.d. samples $X = \{x_1, ..., x_n\}$ drawn from a distribution $\mathcal{F}$ over $\mathbb{R}^D$. We now define *core-points*, which are essentially points with high empirical density defined with respect to the two hyperparameters of DBSCAN, minPts and ε. The latter is also known as the *bandwidth*.

Definition 1. *Let $\varepsilon > 0$ and minPts be a positive integer. Then $x \in X$ is a core-point if $|B(x, \varepsilon) \cap X| \geq$ minPts, where $B(x, \varepsilon) := \{x' : |x - x'| \leq \varepsilon\}$.*

In other words, a core-point is a sample point that has at least minPts sample points within its ε-radius neighborhood.

DBSCAN (Ester et al., 1996) is shown as Algorithm 1, which is in a more concise but equivalent form to the original version (see Jiang (2017a)). It creates a graph G with core-points as vertices and edges connecting core points, which are distance at most ε apart. The final clusters are represented by the connected components in this graph along

with non-core-points that are close to such a connected component. The remaining points are designated as noise points and are left unclustered. Noise points can be seen as outliers.

Algorithm 1 DBSCAN

   Inputs: X, ε, minPts
   $C \leftarrow$ core-points in X given ε and minPts
   $G \leftarrow$ initialize empty graph
   for $c \in C$ **do**
      Add an edge (and possibly a vertex or vertices) in G
      from c to all points in $X \cap B(c, \varepsilon)$
   end for
   return connected components of G.
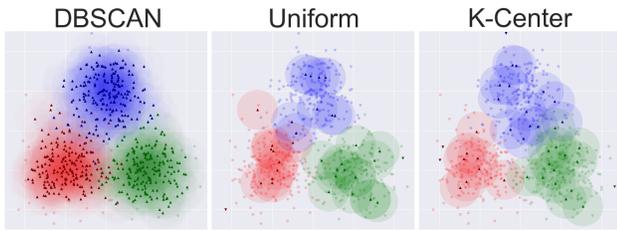
Figure 2. Core-points from a mixture of three 2D Gaussians. Each point marked with a triangle represents a core-point and the shaded area its ε-neighborhood. The total ε-radii area of DBSCAN++ core-points provides adequate coverage of the dataset. The K-center initialization produces an even more efficient covering. The points that are not covered will be designated as outliers. This illustrates that a strategically selected subset of core points can produce a reasonable ε-neighborhood cover for clustering.

3.1. Uniform Initialization

DBSCAN++, shown in Algorithm 2, proceeds as follows: First, it chooses a subset S of m uniformly sampled points from the dataset X. Then, it computes the empirical density of points in S w.r.t. the entire dataset. That is, a point $x \in S$ is a core point if $|B(x, \varepsilon) \cap X| \geq$ minPts. From here, DBSCAN++ builds a similar neighborhood graph G of core-points in S and finds the connected components in G. Finally, it clusters the rest of the unlabeled points to their closest core-points. Thus, since it only recovers a fraction of the core-points, it requires expensive density estimation queries on only m of the n samples. However, the intuition, as shown in Figure 2, is that a smaller sample of core-points can still provide adequate coverage of the dataset and lead to a reasonable clustering.

3.2. K-Center Initialization

Instead of uniformly choosing the subset of m points at random, we can use K-center (Gonzalez, 1985; Har-Peled, 2011), which aims at finding the subset of size m that minimizes the maximum distance of any point in X to its closest

---

Algorithm 2 DBSCAN++

   Inputs: X, m, ε, minPts
   $S \leftarrow$ sample m points from X.
   $C \leftarrow$ all core-points in S w.r.t X, ε, and minPts
   $G \leftarrow$ empty graph.
   for $c \in C$ **do**
      Add an edge (and possibly a vertex or vertices) in G
      from c to all points in $X \cap B(c, \varepsilon)$
   end for
   return connected components of G.

---

point in that subset. In other words, it tries to find the most efficient covering of the sample points. We use the greedy initialization method for approximating K-center (Algorithm 3), which repeatedly picks the farthest point from any point currently in the set. This process continues until we have a total of m points. This method gives a 2-approximation to the K-center problem.

Algorithm 3 Greedy K-center Initialization

   Input: X, m.
   $S \leftarrow \{x_1\}$.
   for i from 1 to $m - 1$ **do**
      Add $\text{argmax}_{x \in X} \min_{s \in S} |x - s|$ to S.
   end for
   return S.

3.3. Time Complexity

DBSCAN++ has a time complexity of $O(nm)$. Choosing the set S takes linear time for the uniform initialization method and $O(mn)$ for the greedy K-center approach (Gonzalez, 1985). The next step is to find the core-points. We use a KDTree to query for the points within the ε-radii ball for each point in S. Each such query takes $O(n)$ in the worst case, and doing so for m sampled points leads to a cost of $O(nm)$. Constructing the graph takes $O(mn)$ time and running a depth-first search on the graph recovers the connected components in $O(nm)$ since the graph will have at most $O(nm)$ edges.

The last step is to cluster the remaining points to the nearest core point. We once again use a KDTree, which takes $O(m)$ for each of $O(n)$ points, leading to a time complexity of $O(nm)$ as well. Thus, the time complexity of DBSCAN++ is $O(nm)$.

4. Theoretical Analysis

In this section, we show that DBSCAN++ is a consistent estimator of the density level-sets. It was recently shown by Jiang (2017a) that DBSCAN does this with finite-sample guarantees. We extend this analysis to show that our modi-

fied DBSCAN++ also has statistical consistency guarantees, and we show the trade-off between speed and convergence rate.

Definition 2. *(Level-set) The λ-level-set of f is defined as $L_f(\lambda) := \{x \in \mathcal{X} : f(x) \geq \lambda\}$.*

Our results are under the setting that the density level λ is known and gives insight into how to tune the parameters based on the desired density level.

4.1. Regularity Assumptions

We have n i.i.d. samples $X = \{x_1, ..., x_n\}$ drawn from a distribution $\mathcal{F}$ over $\mathbb{R}^D$. We take f to be the density of $\mathcal{F}$ over the uniform measure on $\mathbb{R}^D$.

Assumption 1. *f is continuous and has compact support $\mathcal{X} \subseteq \mathbb{R}^D$.*

Much of the results will depend on the behavior of level-set boundaries. Thus, we require sufficient drop-off at the boundaries as well as separation between the CCs at a particular level-set.

Define the following shorthands for distance from a point to a set and the neighborhood around a set.

Definition 3. *$d(x, A) := \inf_{x' \in A} |x - x'|$, $B(C, r) := \{x \in \mathcal{X} : d(x, C) \leq r\}$,*

Assumption 2 (β-regularity of level-sets). *Let $0 < \beta < \infty$. There exist $\check{C}, \hat{C}, r_c > 0$ such that the following holds for all $x \in B(L_f(\lambda), r_c) \backslash L_f(\lambda)$.*

$$\check{C} \cdot d(x, L_f(\lambda))^\beta \leq \lambda - f(x) \leq \hat{C} \cdot d(x, L_f(\lambda))^\beta.$$

Remark 1. *We can choose any $0 < \beta < \infty$. The β-regularity condition is a standard assumption in level-set analyses. See (Singh et al., 2009). The higher the β, the more smooth the density is around the boundary of the level-set and thus the less salient it is. This makes it more difficult to recover the level-set.*

4.2. Hyperparameter Settings

In this section, we state the hyperparameter settings in terms of n, the sample size, and the desired density level λ in order for statistical consistency guarantees to hold. Define $C_{\delta,n} = 16 \log(2/\delta)\sqrt{\log n}$, where δ, $0 < \delta < 1$, is a confidence parameter which will be used later (i.e. guarantees will hold with probability at least $1 - \delta$).

$$\varepsilon = \left(\frac{\text{minPts}}{n \cdot v_D \cdot (\lambda - \lambda \cdot C_{\delta,n}^2 / \sqrt{\text{minPts}})}\right)^{1/D},$$

where v_D is the volume of the unit ball in $\mathbb{R}^D$ and minPts satisfies

$$C_l \cdot (\log n)^2 \leq \text{minPts} \leq C_u \cdot (\log n)^{\frac{2D}{2+D}} \cdot n^{2\beta/(2\beta+D)},$$

and C_l and C_u are positive constants depending on δ, f.

4.3. Level-set estimation result

We give the estimation rate under the Hausdorff metric.

Definition 4 (Hausdorff Distance).

$$d_{Haus}(A, A') = \max\{\sup_{x \in A} d(x, A'), \sup_{x' \in A'} d(x', A)\}.$$

Theorem 1. *Suppose Assumption 1 and 2 hold, and assume the parameter settings in the previous section. There exists C_l, C sufficiently large and C_u sufficiently small such that the following holds with probability at least $1 - \delta$. Let $\widehat{L_f(\lambda)}$ be the core-points returned by Algorithm 2 under uniform initialization or greedy K-center initialization. Then,*

$$d_{Haus}(\widehat{L_f(\lambda)}, L_f(\lambda))$$

$$\leq C \cdot \left(C_{\delta,n}^{2/\beta} \cdot \text{minPts}^{-1/2\beta} + C_{\delta,n}^{1/D} \cdot \left(\frac{\sqrt{\log m}}{m}\right)^{1/D}\right).$$

Proof. There are two quantities to bound: (i) $\max_{x \in \widehat{L_f(\lambda)}} d(x, L_f(\lambda))$, which ensures that the estimated core-points are not far from the true core-points (i.e. $L_f(\lambda)$), and (ii) $\sup_{x \in L_f(\lambda)} d(x, \widehat{L_f(\lambda)})$, which ensures that the estimates core-points provides a good covering of the level-set.

The bound for (i) follows by the main result of Jiang (2017a). This is because DBSCAN++'s estimated core-points are a subset of that of the original DB-SCAN procedure. Thus, $\max_{x \in \widehat{L_f(\lambda)}} d(x, L_f(\lambda)) \leq \max_{x \in \widetilde{L_f(\lambda)}} d(x, L_f(\lambda))$, where $\widetilde{L_f(\lambda)}$ are the core-points returned by original DBSCAN; this quantity is bounded by $O(C_{\delta,n}^{2/\beta} \cdot \text{minPts}^{-1/2\beta})$ by Jiang (2017a).

We now turn to the other direction and bound $\sup_{x \in L_f(\lambda)} d(x, \widehat{L_f(\lambda)})$. Let $x \in L_f(\lambda)$.

Suppose we use the uniform initialization. Define $r_0 := \left(\frac{2C_{\delta,n}\sqrt{D \log m}}{m v_D \cdot \lambda}\right)^{1/D}$. Then, we have

$$\int_{\mathcal{X}} f(z) \cdot 1[z \in B(x, r_0)]dz \geq v_D r_0^D (\lambda - \hat{C} r_0^\beta)$$

$$\geq v_D r_0^D \lambda / 2 = \frac{C_{\delta,n}\sqrt{D \log m}}{m},$$

where the first inequality holds from Assumption 2, the second inequality holds for n sufficiently large, and the last holds from the conditions on minPts.

By the uniform ball convergence rates of Lemma 7 of Chaudhuri and Dasgupta (2010), we have that with high probability, there exists sample point $x' \in S$ such that $|x - x'| \leq r_0$. This is because the ball $B(x, r_0)$ contains sufficiently high true mass to be guaranteed a sample point in S. Moreover, this guarantee holds with high probability uniformly over

$x \in \mathcal{X}$. Next, we show that x' is a core-point. This follows by Lemma 8 of Jiang (2017a), which shows that any sample point in $x \in L_f(\lambda)$ satisfies $|B(x, \varepsilon) \cap X| \geq$ minPts. Thus, $x' \in \widehat{L_f(\lambda)}$. Hence, $\sup_{x \in L_f(\lambda)} d(x, \widehat{L_f(\lambda)}) \leq r_0$, as desired.

Now suppose we use the greedy K-center initialization. Define the following attained K-center objective:

$$\tau := \max_{x \in X} \min_{s \in S} d(s, x),$$

and the optimal K-center objective:

$$\tau_{\text{opt}} := \min_{S' \subseteq \mathcal{X}, |S'| = m} \max_{x \in X} \min_{s \in S'} d(s, x).$$

It is known that the greedy K-center initialization is a 2-approximation (see Gonzalez (1985); Har-Peled (2011)), thus

$$\tau \leq 2\tau_{\text{opt}} \leq 2r_0,$$

where the last inequality follows with high probability since the K-center objective will be sub-optimal if we sampled the m centers uniformly. Then, we have

$$\sup_{x \in L_f(\lambda)} \min_{s \in S} d(s, x)$$
$$\leq \max_{x \in X} \min_{s \in S} d(s, x) + d_{\text{Haus}}(L_f(\lambda), X \cap L_f(\lambda))$$
$$\leq \tau + r_0 \leq 3r_0.$$

The argument then proceeds in the same way as with uniform initialization but with an extra constant factor, as desired. □

Remark 2. *When taking minPts to the maximum allowed rate*

$$minPts \approx n^{2\beta/(2\beta + D)},$$

we obtain the error rate (ignoring log factors) of

$$d_{\text{Haus}}(\widehat{L_f(\lambda)}, L_f(\lambda)) \lesssim n^{-1/(2\beta + D)} + m^{-1/D}.$$

The first term matches the known lower bound for level-set estimation established in Theorem 4 of Tsybakov et al. (1997). The second term is the trade-off for computing the empirical densities for only m of the points. In particular, if we take

$$m \gtrsim n^{D/(2\beta + D)},$$

then the first term dominates, and we thus have $d_{\text{Haus}}(\widehat{L_f(\lambda)}, L_f(\lambda)) \lesssim n^{-1/(2\beta + D)}$, the minimax optimal rate for level-set estimation. This leads to the following result.

Corollary 1. *Suppose the conditions of Theorem 1 and set $m \approx n^{D/(2\beta + D)}$. Then, Algorithm 2 is a minimax optimal estimator (up to logarithmic factors) of the density level-set with sub-quadratic runtime of $O(n^{2 - 2\beta/(2\beta + D)})$.*

	n	D	c	m
(A) iris	150	4	3	3
(B) wine	178	13	3	5
(C) spam	1401	57	2	793
(D) images	210	19	7	24
(E) MNIST	60000	20	10	958
(F) Libras	360	90	15	84
(G) mobile	2000	20	4	112
(H) zoo	101	16	7	8
(I) seeds	210	19	7	6
(J) letters	20000	16	26	551
(K) phonemes	4509	256	5	396
(L) fashion MNIST	60000	784	10	5674
(M) celeb-a	10000	40	3	3511

Figure 3. Summary of datasets used. Includes dataset size (n), number of features (D), number of clusters (c), and the (m) used by both DBSCAN++ uniform and K-center.

4.4. Estimating the connected components

The previous section shows that the core-points returned by DBSCAN++ recovers the density level-set. The more interesting question is about the actual clustering: that is, whether DBSCAN++ can recover the connected components of the density level-set individually and whether there is a 1:1 correspondence between the clusters returned by DBSCAN++ and the connected components.

It turns out that to obtain such a result, we need a minor modification of the procedure. That is, after determining the core points, instead of using the ε cutoff to connect points into the same cluster, we must use a higher cutoff. In fact, any constant value would do as long as it is sufficiently smaller than the pairwise distances between the connected components. For example, for the original DBSCAN algorithm, many analyses must make this same modification. This is known as *pruning false clusters* in the literature (see Kpotufe and von Luxburg (2011); Jiang (2017a)). The same analysis carries over to our modification, and we omit it here. We note that pruning does not change the final estimation rates but may change the initial sample size required.

4.5. Outlier detection

One important application of DBSCAN is outlier detection (Breunig et al., 2000; Çelik et al., 2011; Thang and Kim, 2011). Datapoints not assigned to clusters are *noise points* and can be considered outliers. This is because the noise points are the low density points away from the clusters and thus tend to be points with few similar representatives in the dataset. We show that the noise points DBSCAN++ finds are similar to the noise points discovered by DBSCAN++. We give a simple result that shows that every DBSCAN noise point is also one DBSCAN++ finds (Lemma 1). Then,

Figure 4 (Left) shows that the number of noise points of DBSCAN++ quickly converges to those of DBSCAN as the ratio m/n increases, which combined with Lemma 1, shows that the noise points DBSCAN++ returns closely approximates those returned by DBSCAN for m/n sufficiently high.

Lemma 1 (Noise points). *For any dataset, if N_0 and N_1 are the noise points found by DBSCAN and DBSCAN++ respectively, then as long as they have the same setting of ε and k, we have that $N_0 \subseteq N_1$.*

Proof. Noise points are those that are further than ε distance away from a core point. The result follows since DB-SCAN++ core points are a subset of that of DBSCAN. □

5. Experiments

5.1. Dataset and setup

We ran DBSCAN++ with uniform and K-center initializations and compared both to DBSCAN on 11 real datasets as described in Figure 3. We used Phonemes (Friedman et al., 2001), a dataset of log periodograms of spoken phonemes, and MNIST, a sub-sample of the MNIST handwriting recognition dataset after running a PCA down to 20 dimensions. The rest of the datasets we used are standard UCI or Kaggle datasets used for clustering. We evaluate the performance via two widely-used clustering scores: the adjusted RAND index (Hubert and Arabie, 1985) and adjusted mutual information score (Vinh et al., 2010), which are computed against the ground truth. We fixed minPts $=10$ for all procedures throughout experiments.

5.2. Trade-off between accuracy and speed

The theoretical results suggest that up to a certain point, only computing empirical densities for a subset of the sample points should not noticeably impact the clustering performance. Past that point, we begin to see a trade-off. We confirm this empirically in Figure 4 (Right), which shows that indeed past a certain threshold of m/n, the clustering scores remain high. Only when the sub-sample is too small do we begin seeing a significant trade-off in clustering scores. This shows that DBSCAN++ can save considerable computational cost while maintaining similar clustering performance as DBSCAN.

We further demonstrate this point by applying these procedures to image segmentation, where segmentation is done by clustering the image's pixels (with each pixel represented as a 5-dimensional vector consisting of (x, y) position and RGB color). Figure 5 shows that DBSCAN++ provides a very similar segmentation as DBSCAN in a fraction of the time. While this is just a simple qualitative example, it serves to show the applicability of DBSCAN++ to a potentially wide range of problems.
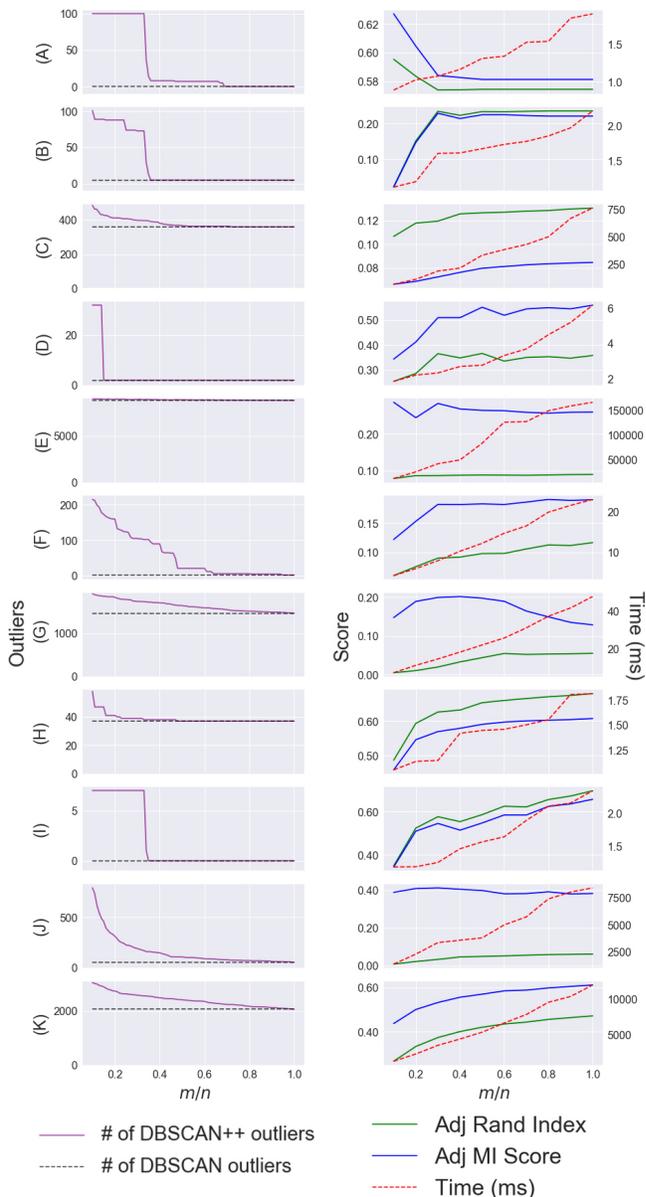
Figure 4. Each row corresponds to a dataset. See Figure 3 for dataset descriptions. **Left (Outlier Detection)**: The number of outliers (i.e. noise points) returned by DBSCAN against m/n and compared it to that of DBSCAN++. We see that the set of DBSCAN++ outliers quickly approaches those of DBSCAN's thus showing that DBSCAN++ remains effective at outlier detection compared to DBSCAN, especially when m/n is sufficiently high. **Right (Clustering Performance)**: we plot the clustering accuracy and runtimes for eight real datasets as a function of the ratio m/n. As expected, the runtime increases approximately linearly in this ratio, but the clustering scores consistently attain high values when m/n is sufficiently large. Interestingly, sometimes we attain higher scores with lower values of m/n thus giving both better runtime and accuracy.

Figure 5. *Figure skater Yuzuru Hanyu at the 2018 Olympics.* DB-SCAN was initiated with hyperparameters $\varepsilon = 8$ and minPts $= 10$, and DBSCAN++ with $\varepsilon = 60$, $m/n = 0.3$, and minPts $= 10$. DBSCAN++ with K-centers initialization recovers similar clusters (designated by the purple boundaries) in the 988×750 image as DBSCAN in far less time: 7.38 seconds versus 44.18 seconds. The speedup becomes more significant on higher resolution images.

5.3. Robustness to Hyperparameters

In Figure 6, we plot each algorithm's performance across a wide range of its hyperparameters. The table in Figure 7 shows the best scores and runtimes for each dataset and algorithm. For these experiments, we chose $m = p \cdot n^{D/(D+4)}$, where $0 < p < 1$ was chosen based on validation over just 3 values, as explained in Figure 7. We found that the K-center initialization required smaller p due to its ability to find a good covering of the space and more efficiently choose the sample points to query for the empirical density.

The results in Figure 6 show that DBSCAN++ with uniform initialization gives competitive performance compared to DBSCAN but with robustness across a much wider range of ϵ. In fact, in a number of cases, DBSCAN++ was even better than DBSCAN under optimal tuning. DBSCAN++ with K-center initialization further improves on the clustering results of DBSCAN++ for most of the datasets. Pruning the core-points as DBSCAN++ may act as a regularizing factor to prevent the algorithm's dependency on the preciseness of its parameters.

An explanation of why DBSCAN++ added robustness across ε follows. When tuning DBSCAN with respect to ε, we found that DBSCAN often performed optimally on only a narrow range of ε. Because ε controls both the designation of points as core-points as well as the connectivity of the core-points, small changes could produce significantly different clusterings.
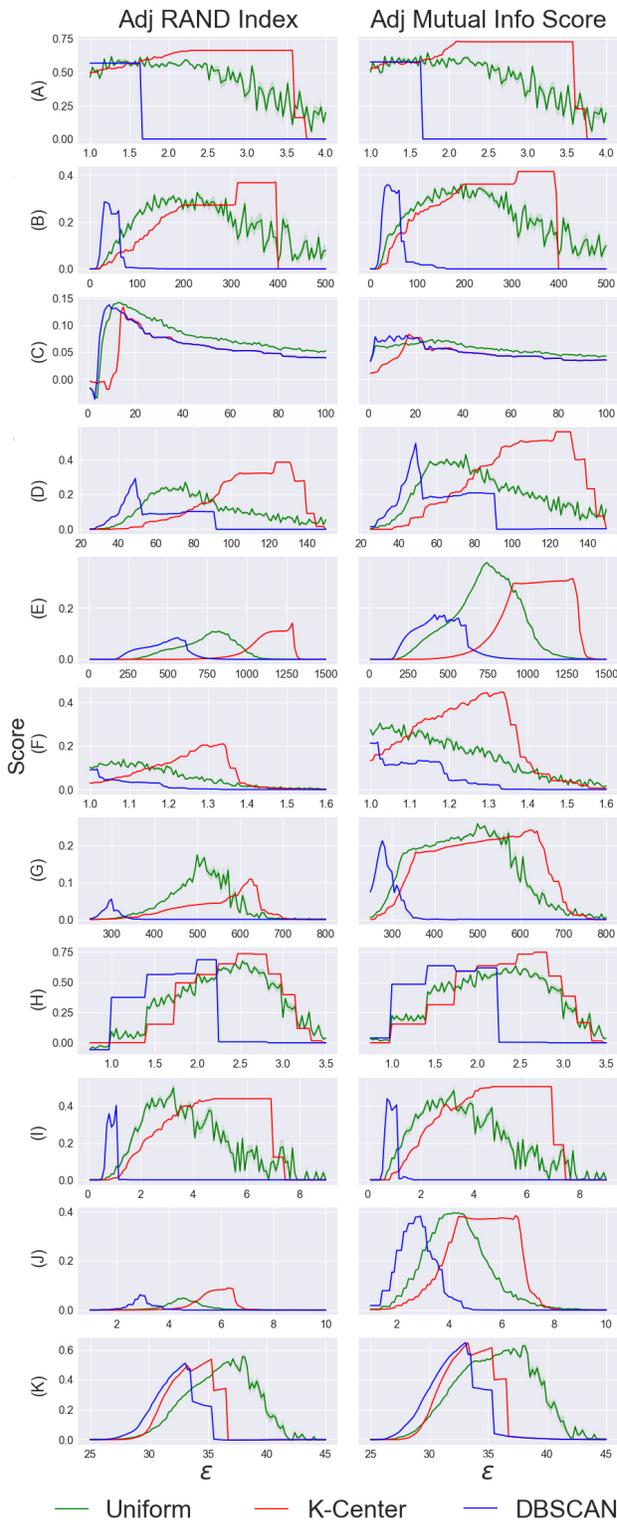
Figure 6. **Clustering performance over range of hyperparameter settings**. Experimental results on datasets described in Figure 3. Each row corresponds to a single dataset and each column corresponds to a clustering score. For each dataset and clustering score, we plot the scores for DBSCAN++ with uniform and K-center sampling vs DBSCAN across a wide range of settings for ε (x-axis).

	DBSCAN	Uniform	K-Center
(A)	0.5681	**0.6163** (±0.01)	**0.6634**
	0.5768	**0.6449** (±0.01)	**0.7301**
(B)	0.2851	**0.3254** (±0.01)	**0.3694**
	0.3587	**0.3605** (±0.00)	**0.4148**
(C)	0.2851	**0.3254** (±0.01)	**0.3694**
	0.3587	**0.3605** (±0.00)	**0.4148**
(D)	**0.2922**	0.2701 (±0.01)	**0.3853**
	0.4938	0.4289 (±0.01)	**0.5600**
(E)	0.0844	**0.1097** (±0.00)	**0.1416**
	0.1743	**0.3774** (±0.00)	**0.3152**
(F)	0.0939	**0.1380** (±0.00)	**0.2095**
	0.2170	**0.3033** (±0.00)	**0.4461**
(G)	0.0551	**0.1741** (±0.00)	**0.1091**
	0.2123	**0.2585** (±0.00)	**0.2418**
(H)	**0.6846**	0.6729 (±0.01)	**0.7340**
	0.6347	**0.6356** (±0.00)	**0.7456**
(I)	0.4041	**0.4991** (±0.02)	**0.4402**
	0.4403	**0.4843** (±0.02)	**0.5057**
(J)	**0.0623**	0.0488 (±0.00)	**0.0901**
	0.3823	**0.3956** (±0.00)	**0.3841**
(K)	0.5101	**0.5541** (±0.01)	**0.5364**
	0.6475	0.6259 (±0.01)	**0.6452**

Figure 7. **Highest scores for each dataset and algorithm.** The first row is the adjusted RAND index and the second row the adjusted mutual information. The highest score of the row is in **green** while the second highest is in **orange**. The standard error over 10 runs is given in parentheses for DBSCAN++ with uniform initialization. Both other algorithms are deterministic. Each algorithm was tuned across a range of ϵ with minPts = 10. For both DBSCAN++ algorithms, we used p values of $0.1, 0.2$, or 0.3. We see that DBSCAN++ uniform performs better than DBSCAN on 17 out of 22 metrics, while DBSCAN++ K-center performs better than DBSCAN on 21 out of 22 metrics.

In contrast, DBSCAN++ suffers less from the hyper-connectivity of the core-points until ε is very large. It turns out that only processing a subset of the core-points not only reduces the runtime of the algorithm, but it provides the practical benefit of reducing the tenuous connections between connected components that are actually far away. This way, DBSCAN++ is much less sensitive to changes in ε and reaches its saturation point (where there is only one cluster) only at very large ε.

Performance under optimal tuning is often not available in practice, and this is especially the case in unsupervised problems like clustering where the ground truth is not assumed to be known. Thus, not only should procedures produce accurate clusterings in the best setting, but it may be even more important for procedures to be precise, *easy to tune*, reasonable across a *wide range* of its hyperparameter settings. This added robustness (not to mention speedup) may make DBSCAN++ a more practical method. This is espe-

	DBSCAN	Uniform	K-Center
(A)	3.07 (±0.08)	1.52 (±0.09)	2.55 (±0.34)
(B)	2.04 (±0.07)	1.31 (±0.07)	0.79 (±0.02)
(C)	3308 (±26.4)	225.86 (±6.8)	442.69 (±2.0)
(D)	4.88 (±0.09)	1.51 (±0.05)	1.32 (±0.04)
(E)	1.5e5 (±0.17)	3.5e3 (±39.23)	7.0e3 (±41.1)
(F)	37.63 (±0.38)	8.20 (±0.22)	9.84 (±0.06)
(G)	67.05 (±0.63	11.41 (±0.21)	15.23 (±0.32)
(H)	1.07 (±0.03)	0.78 (±0.03)	0.81 (±0.03)
(I)	1.75 (±0.04)	0.91 (±0.03)	0.97 (±0.09)
(J)	1.0e5 (±76.43)	5.2e3 (±17.48)	1.5e3 (±36.4)
(K)	1.2e4 (±160)	1.9e3 (±32.45)	1.9e3 (±30.4)
(L)	3.9e9 (±4.3e4)	7.4e8 (±4.1e3)	3.6e8(±307)
(M)	4.1e9 (±6.2e4)	3.1e8 (±411)	2.3e8(±1.1e3)

Figure 8. **Runtimes** (milliseconds) and standard errors for each dataset and algorithm. DBSCAN++ using both uniform and K-center initializations performs reasonably well within a fraction of the runtime of DBSCAN. The larger the dataset, the less time DBSCAN++ requires compared to DBSCAN, showing that DBSCAN++ scales much better in practice.

cially true on large datasets where it may be costly to iterate over many hyperparameter settings.

5.4. Performance under optimal tuning

We see that under optimal tuning of each algorithm, DBSCAN++ consistently outperforms DBSCAN in both clustering scores and runtime. We see in Figure 7 that DBSCAN++ with the uniform initialization consistently outperforms DBSCAN and DBSCAN++ with K-center initialization consistently outperforms both of the algorithms. Figure 8 shows that indeed DBSCAN++ gives a speed advantage over DBSCAN for the runs that attained the optimal performance. These results thus suggest that not only is DBSCAN++ faster, it can achieve better clusterings.

6. Conclusion

In this paper, we presented DBSCAN++, a modified version of DBSCAN that only computes the density estimates for a subset m of the n points in the original dataset. We established statistical consistency guarantees which show the trade-off between computational cost and estimation rates, and we prove that interestingly, up to a certain point, we can enjoy the same estimation rates while reducing computation cost. We also demonstrate this finding empirically. We then showed empirically that not only can DBSCAN++ scale considerably better than DBSCAN, its performance is competitive in accuracy and consistently more robust across their mutual bandwidth hyperparameters. Such robustness can be highly desirable in practice where optimal tuning is costly or unavailable.

References

Guilherme Andrade, Gabriel Ramos, Daniel Madeira, Rafael Sachetto, Renato Ferreira, and Leonardo Rocha. G-dbscan: A gpu accelerated algorithm for density-based clustering. *Procedia Computer Science*, 18:369–378, 2013.

Domenica Arlia and Massimo Coppola. Experiments in parallel clustering with dbscan. In *European Conference on Parallel Processing*, pages 326–331. Springer, 2001.

Sivaraman Balakrishnan, Srivatsan Narayanan, Alessandro Rinaldo, Aarti Singh, and Larry Wasserman. Cluster trees on manifolds. In *Advances in Neural Information Processing Systems*, pages 2679–2687, 2013.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104. ACM, 2006.

B Borah and DK Bhattacharyya. An improved sampling-based dbscan for large spatial databases. In *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*, pages 92–96. IEEE, 2004.

Stefan Brecheisen, Hans-Peter Kriegel, and Martin Pfeifle. Parallel density-based clustering of complex objects. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 179–188. Springer, 2006.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

Mete Çelik, Filiz Dadaşer-Çelik, and Ahmet Şakir Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on*, pages 91–95. IEEE, 2011.

Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for the cluster tree. In *Advances in Neural Information Processing Systems*, pages 343–351, 2010.

Kamalika Chaudhuri, Sanjoy Dasgupta, Samory Kpotufe, and Ulrike von Luxburg. Consistent procedures for cluster tree estimation and pruning. *IEEE Transactions on Information Theory*, 60(12):7900–7912, 2014.

Min Chen, Xuedong Gao, and HuiFei Li. Parallel dbscan with priority r-tree. In *Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on*, pages 508–511. IEEE, 2010.

Yen-Chi Chen, Christopher R Genovese, and Larry Wasserman. Density level sets: Asymptotics, inference, and visualization. *Journal of the American Statistical Association*, pages 1–13, 2017.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

Bi-Ru Dai and I-Chang Lin. Efficient map/reduce-based dbscan algorithm with optimized data partition. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 59–66. IEEE, 2012.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

Yan Xiang Fu, Wei Zhong Zhao, and Hui Fang Ma. Research on parallel dbscan algorithm design based on mapreduce. In *Advanced Materials Research*, volume 301, pages 1133–1138. Trans Tech Publ, 2011.

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

ZHOU Shui Geng, ZHOU Ao Ying, CAO Jing, and HU Yun Fa. A fast density based clustering algorithm [j]. *Journal of Computer Research and Development*, 11:001, 2000.

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

Markus Götz, Christian Bodenstein, and Morris Riedel. Hpdbscan: highly parallel dbscan. In *Proceedings of the workshop on machine learning in high-performance computing environments*, page 2. ACM, 2015.

Sariel Har-Peled. *Geometric approximation algorithms*. Number 173. American Mathematical Soc., 2011.

John A Hartigan. *Clustering algorithms*, volume 209. Wiley New York, 1975.

Yaobin He, Haoyu Tan, Wuman Luo, Huajian Mao, Di Ma, Shengzhong Feng, and Jianping Fan. Mr-dbscan: an efficient parallel density-based clustering algorithm using mapreduce. In *Parallel and Distributed Systems (IC-PADS), 2011 IEEE 17th International Conference on*, pages 473–480. IEEE, 2011.

Ming Huang and Fuling Bian. A grid and density based fast spatial clustering algorithm. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*, volume 4, pages 260–263. IEEE, 2009.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. Dbdc: Density based distributed clustering. In *International Conference on Extending Database Technology*, pages 88–105. Springer, 2004a.

Eshref Januzaj, Hans-Peter Kriegel, and Martin Pfeifle. Scalable density-based distributed clustering. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 231–244. Springer, 2004b.

Heinrich Jiang. Density level set estimation on manifolds with dbscan. In *International Conference on Machine Learning*, pages 1684–1693, 2017a.

Heinrich Jiang. Uniform convergence rates for kernel density estimation. In *International Conference on Machine Learning*, pages 1694–1703, 2017b.

Samory Kpotufe and Ulrike von Luxburg. Pruning nearest neighbor cluster trees. *arXiv preprint arXiv:1105.0540*, 2011.

Marzena Kryszkiewicz and Piotr Lasek. Ti-dbscan: Clustering with dbscan by means of the triangle inequality. In *International Conference on Rough Sets and Current Trends in Computing*, pages 60–69. Springer, 2010.

K Mahesh Kumar and A Rama Mohan Reddy. A fast dbscan clustering algorithm by accelerating neighbor searching using groups method. *Pattern Recognition*, 58:39–48, 2016.

Bing Liu. A fast density-based clustering algorithm for large databases. In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 996–1000. IEEE, 2006.

Jinfei Liu, Joshua Zhexue Huang, Jun Luo, and Li Xiong. Privacy preserving distributed dbscan clustering. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, pages 177–185. ACM, 2012.

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.

Antonio Cavalcante Araujo Neto, Ticiana Linhares Coelho da Silva, Victor Aguiar Evangelista de Farias, José Antonio F Macêdo, and Javam de Castro Machado. G2p: A partitioning approach for processing dbscan with mapreduce. In *International Symposium on Web and Wireless Geographical Information Systems*, pages 191–202. Springer, 2015.

Maitry Noticewala and Dinesh Vaghela. Mr-idbscan: Efficient parallel incremental dbscan algorithm using mapreduce. *International Journal of Computer Applications*, 93(4), 2014.

Mostofa Ali Patwary, Diana Palsetia, Ankit Agrawal, Weikeng Liao, Fredrik Manne, and Alok Choudhary. A new scalable parallel dbscan algorithm using the disjoint-set data structure. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 62. IEEE Computer Society Press, 2012.

Philippe Rigollet, Régis Vert, et al. Optimal rates for plug-in estimators of density level sets. *Bernoulli*, 15(4):1154–1178, 2009.

Alessandro Rinaldo and Larry Wasserman. Generalized density clustering. *The Annals of Statistics*, pages 2678–2722, 2010.

Aarti Singh, Clayton Scott, Robert Nowak, et al. Adaptive hausdorff estimation of density level sets. *The Annals of Statistics*, 37(5B):2760–2782, 2009.

Bharath Sriperumbudur and Ingo Steinwart. Consistency and rates for clustering with dbscan. In *Artificial Intelligence and Statistics*, pages 1090–1098, 2012.

Ingo Steinwart. Adaptive density level set clustering. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 703–738, 2011.

Ingo Steinwart, Bharath K Sriperumbudur, and Philipp Thomann. Adaptive clustering using kernel density estimators. *arXiv preprint arXiv:1708.05254*, 2017.

Tran Manh Thang and Juntae Kim. The anomaly detection by using dbscan clustering with multiple parameters. In *Information Science and Applications (ICISA), 2011 International Conference on*, pages 1–5. IEEE, 2011.

Alexandre B Tsybakov et al. On nonparametric estimation of density level sets. *The Annals of Statistics*, 25(3): 948–969, 1997.

S Vijayalaksmi and M Punithavalli. A fast approach to clustering datasets using dbscan and pruning algorithms. *International Journal of Computer Applications*, 60(14), 2012.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

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

P Viswanath and Rajwala Pinkesh. l-dbscan: A fast hybrid density based clustering method. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 912–915. IEEE, 2006.

Daren Wang, Xinyang Lu, and Alessandro Rinaldo. Optimal rates for cluster tree estimation using kernel density estimators. *arXiv preprint arXiv:1706.03113*, 2017.

Yi-Pu Wu, Jin-Jiang Guo, and Xue-Jie Zhang. A linear dbscan algorithm based on lsh. In *2007 International Conference on Machine Learning and Cybernetics*, volume 5, pages 2608–2614. IEEE, 2007.

Xiaowei Xu, Jochen Jäger, and Hans-Peter Kriegel. A fast parallel clustering algorithm for large spatial databases. In *High Performance Data Mining*, pages 263–290. Springer, 1999.

Aoying Zhou, Shuigeng Zhou, Jing Cao, Ye Fan, and Yunfa Hu. Approaches for scaling dbscan algorithm to large spatial databases. *Journal of computer science and technology*, 15(6):509–526, 2000a.

Shuigeng Zhou, Aoying Zhou, Wen Jin, Ye Fan, and Weining Qian. Fdbscan: a fast dbscan algorithm. *Ruan Jian Xue Bao*, 11(6):735–744, 2000b.