Density-Based Clustering: DBSCAN

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
In some applications, clusters can have arbitrary shapes and may need to be separated from noise:

(figures from a KDD96 paper titled “A density-based algorithm for discovering clusters in large spatial databases with noise”)

We will learn a method called **DBSCAN** to find such clusters. It serves as a representative of **density-based clustering**, which works by enforcing two principles:

- The area around a noise point is “sparse”.
- If two points are placed in the same cluster, it should be possible to “walk” from one point to the other by staying only in the “dense” areas.
Parameters and Core Points

Parameters:

- $\epsilon$: a distance threshold.
- $\text{MinPts}$: a constant integer.

$B(p, \epsilon)$: the ball centered at a point with radius $\epsilon$, called the **vicinity area** of $p$.

$P$: the set of points to cluster

Core point: a point $p \in P$ such that $B(p, \epsilon)$ covers at least $\text{MinPts}$ points of $P$. 

$\text{MinPts} = 4$

Core points in black
Forming Clusters

Conceptually, clusters are defined in two steps:

1. Cluster core points.
2. Assign non-core points.

We will explain each step in turn.
Step 1: Cluster core points

This step focuses **only** on core points.

\[ \text{MinPts} = 4 \]

Core points in black
Step 1: Cluster core points

Connect a core point $p$ to all the points in $B(p, \varepsilon)$.

For example, $o_1$ is connected to 4 points in its vicinity area:

$MinPts = 4$

Core points in black
Step 1: Cluster core points

This is the situation after adding all the edges:
Step 1: Cluster core points

Take each connected component of the resulting graph as a cluster.
Step 2: Assign non-core points

Every non-core point $p$ is added to the cluster of every core point in $B(p, \epsilon)$. For example, $o_{10}$ is added to two clusters: the cluster of $o_1$ and the cluster of $o_{11}$.

$MinPts = 4$

Each non-core point can be assigned to at most $MinPts - 1 = O(1)$ clusters.
Step 2: Assign non-core points

Final clusters: \( \{o_1, o_2, \ldots, o_9, o_{10}\}, \{o_{11}, o_{12}, \ldots, o_{17}\} \).

MinPts = 4

The clustering result is unique.
It is straightforward to obtain the DBSCAN clusters in $O(n^2)$ time, where $n$ is the number of points (think: how). This is too slow in practice, where a useful algorithm should have running time $O(n \text{polylog } n)$. Unfortunately, this is unlikely to be possible for DBSCAN when the dimensionality $d$ goes beyond 2 (i.e., $d \geq 3$), as we explain next.
Let \( S_{pt} \) be a set of points, and \( S_{ball} \) be a set of balls with the same radius, all in data space \( \mathbb{R}^d \), where the dimensionality \( d \) is a constant.

The objective of USEC is to determine whether there is a point of \( S_{pt} \) that is covered by some ball in \( S_{ball} \).

**Known results:**
- \( d = 2 \): Solvable in \( O(n \log n) \) time.
- \( d = 3 \): Solvable \( O((n \log n)^{4/3}) \) time.

**Big open problem:** \( o(n^{4/3}) \) for \( d = 3 \)?

Common conjecture: no.
Let $S_{pt}$ be a set of points, and $S_{line}$ be a set of lines, all in data space $\mathbb{R}^2$ (note that the dimensionality is always 2).

The goal of the Hopcroft’s problem is to determine whether there is a point in $S_{pt}$ that lies on some line of $S_{line}$.

**Known results:** Solvable in time slightly higher than $O(n^{4/3})$.

**Big open problem:** $o(n^{4/3})$ possible?

**Common conjecture:** No.

$\Omega(n^{4/3})$ lower bound known on a broad class of algorithms.
Geometry Preliminary 3: Hopcroft Hardness

A problem $X$ is Hopcroft hard if an algorithm solving $X$ in $o(n^{4/3})$ time implies an algorithm solving the Hopcroft’s problem in $o(n^{4/3})$ time.

Fact: USEC is Hopcroft hard for $d \geq 5$. 
We will prove:

**Theorem 1.**

The following statements are true about the DBSCAN problem:

- It is Hopcroft hard in any dimensionality \( d \geq 5 \).
  - Namely, the problem requires \( \Omega \left( n^{4/3} \right) \) time to solve, unless the Hopcroft problem can be settled in \( o \left( n^{4/3} \right) \) time.

- When \( d = 3 \) (and hence, \( d = 4 \)), the problem requires \( \Omega \left( n^{4/3} \right) \) time to solve, unless the USEC problem can be settled in \( o \left( n^{4/3} \right) \) time.
More specifically, we will prove:

**Lemma 2.**

For any constant dimensionality $d$, if we can solve the DBSCAN problem in $T(n)$ time, then we can solve the USEC problem in $T(n) + O(n)$ time.

The theorem is a corollary of this lemma (think: why).
Let $S_{pt}$ be a set of points, and $S_{ball}$ be a set of balls with the same radius, all in data space $\mathbb{R}^d$, where the dimensionality $d$ is a constant. The objective of USEC is to determine whether there is a point of $S_{pt}$ that is covered by some ball in $S_{ball}$.

Next, we give a reduction from USEC to DBSCAN. Specifically, given a DBSCAN algorithm $A$, we show how to solve USEC by using $A$ as a black box.
Using DBSCAN to Solve USEC
Using DBSCAN to Solve USEC

Obtain $P$ as the union of $S_{pt}$ and the set of centers of the balls in $S_{ball}$. 
Using DBSCAN to Solve USEC

Run the DBSCAN algorithm $A$ to cluster $P$ with

- Set $\epsilon$ to the radius of the balls.
- $MinPts = 1$. 
Using DBSCAN to Solve USEC

Run the DBSCAN algorithm $A$ to cluster $P$ with

- Set $\epsilon$ to the radius of the balls.
- $MinPts = 1.$
Using DBSCAN to Solve USEC

Run the DBSCAN algorithm $A$ to cluster $P$ with

- $\epsilon = \text{the radius of the balls.}$
- $\text{MinPts} = 1.$
Using DBSCAN to Solve USEC

Check if any red square and black circle are put in the same cluster.

- If so, say “yes” to USEC.
- Otherwise, say “no”.

Running time $T(n) + O(n)$. 
Using DBSCAN to Solve USEC

**Correctness:** An original circle covers a point iff we say yes.

**Proof:** The only-if direction is obvious (think: why?). We will focus on proving the if-direction.

![Diagram](image)

A “yes” answer means that there is a sequence of points $p_1, p_2, \ldots, p_t \in P$ such that (i) $p_1$ is red and $p_t$ is black, and (ii) $\text{dist}(p_i, p_{i+1}) \leq r$ for each $i \in [1, t-1]$. Let $k$ be the smallest $i \in [2, t]$ such that $p_i$ is black. Note that $k$ definitely exists because $p_t$ is black. It thus follows that the ball centered at $p_{k-1}$ covers the point $p_k$ in the original USEC problem. 

□
\( \Omega(n^{4/3}) \) is prohibitively expensive even for moderately large \( n \).

Next, we show how a little approximation allows us to bring down the computation time to \( O(n) \) expected.
\( \rho \)-Approximate DBSCAN

Parameters:

- \( \epsilon \): a distance threshold.
- \( MinPts \): a constant integer.
- \( \rho \): an arbitrary non-negative constant.
\( \rho \)-Approximate DBSCAN

Same as DBSCAN:

- Core point
- Conceptually, two steps:
  1. Cluster core points.
  2. Assign non-core points.

Only difference: Step 1.
$\rho$-Approximate DBSCAN

Step 1: Cluster core points.

Let $p$ and $q$ be core points. Conceptually:

- if $\text{dist}(p, q) \leq \epsilon$, definitely create an edge;
Step 1: Cluster core points.

Let $p$ and $q$ be core points. Conceptually:

- if $\text{dist}(p, q) \leq \epsilon$, definitely create an edge;
- if $\text{dist}(p, q) > \epsilon(1 + \rho)$, definitely no edge;
\( \rho \)-Approximate DBSCAN

Step 1: Cluster core points.

Let \( p \) and \( q \) be core points. Conceptually:

- if \( \text{dist}(p, q) \leq \epsilon \), definitely create an edge;
- if \( \text{dist}(p, q) > \epsilon(1 + \rho) \), definitely no edge;
- otherwise, don't care.
\(\rho\)-Approximate DBSCAN

Step 1: Cluster core points.

Find the connected components of the resulting graph.
Both DBSCAN and $\rho$-approximate DBSCAN are parameterized by $\epsilon$ and MinPts.

It would be perfect if they could always return exactly the same clustering results. Of course, this is too good to be true.

Nevertheless, we will show that this is almost true: the result of $\rho$-approximate DBSCAN is guaranteed to be somewhere between the (exact) DBSCAN results obtained by $(\epsilon, \text{MinPts})$ and by $(\epsilon(1 + \rho), \text{MinPts})$. 
Theorem 3 (Sandwich Quality Guarantee).

The following statements are true:

1. For any cluster $C_1$ of $(\epsilon, \text{MinPts})$-DBSCAN, there is a cluster $C$ $\rho$-approx DBSCAN such that $C_1 \subseteq C$.

2. For any cluster $C$ of $\rho$-approx DBSCAN, there is a cluster $C_2$ of $(\epsilon(1 + \rho), \text{MinPts})$-DBSCAN such that $C \subseteq C_2$. 
Sandwich Guarantee
Sandwich Guarantee
Sandwich Guarantee
Take-away message: $\rho$-approximate DBSCAN returns the same clusters as exact DBSCAN, unless a cluster is not “stable” such that it gets merged with another cluster when $\epsilon$ is increased slightly (i.e., by a factor of $\rho$).
Theorem 4.

There is a $\rho$-approximate DBSCAN algorithm that terminates in $O(n)$ expected time, regardless of the value of $\epsilon$, the constant approximation ratio $\rho$, and the fixed dimensionality $d$. 