Clustering
Density based and grid based approaches

Huiping Cao
Density-based clustering methods

- Clustering based on density (local cluster criterion), such as density-connected points

![Database 1](image1.png) ![Database 2](image2.png) ![Database 3](image3.png)

(Data sets from DBSCAN paper)

- Motivation:
  - Discover clusters of arbitrary shape
  - Handle noise

- Requirement:
  - Need density parameters as termination condition
Density-based clustering methods

- Several interesting studies
  - DBSCAN: Ester, et al. (KDD96)
  - DENCLUE: Hinneburg & D. Keim (KDD98)
  - CLIQUE: Agrawal, et al. (SIGMOD98) (more grid-based)
DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Relies on a **density-based** notion of cluster

- A cluster is defined as a maximal set of **density-connected** points

- Discovers clusters of **arbitrary shape** in spatial databases with noise
DBSCAN – basic concepts

- Dataset $D$ of points in $k$-dimensional space
- $\text{dist}(p, q)$: distance of two objects $p$ and $q$
- Two parameters
  - Eps $\epsilon$: Maximum radius of the neighbourhood
  - $\text{MinPts}$: Minimum number of points in an Eps-neighbourhood of that point
- The Eps-neighborhood of a point $p$:
  $N_\epsilon(p) = \{q | q \in D \land \text{dist}(p, q) \leq \epsilon\}$
DBSCAN – basic concepts

- **Core point**: points inside a cluster. \(|N_{\epsilon}(q)| \geq MinPts\)

- **Border point**: points on the border of a cluster.

---

Eps = 1cm  
MinPts = 5
DBSCAN – basic concepts – directly density-reachable

- **Directly densityreachable**: A point \( p \) is directly density-reachable from a point \( q \) w.r.t. \( Eps \), \( MinPts \) if
  - \( p \in N_\epsilon(q) \)
  - \( q \) is a core point, i.e., \( |N_\epsilon(q)| \geq MinPts \)

- Directly density-reachable is symmetric for pairs of core points; NOT symmetric if one core point and one border point are involved.

\[ \begin{array}{c}
\text{MinPts} = 5 \\
\text{Eps} = 1 \text{ cm}
\end{array} \]

\( p \) is directly density reachable from \( q \); \( q \) is not directly density reachable from \( p \).
DBSCAN – basic concepts

- Density-Reachable
  - A point $p$ is density-reachable from a point $q$ w.r.t. $Eps \epsilon$, $MinPts$ if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q, p_n = p$ such that $p_{i+1}$ is directly density-reachable from $p_i$
  - Transitive
  - Non-symmetric
There must be a core point in a cluster C from which two border points of C are density-reachable.

A point p is density-connected to a point q w.r.t. \( Eps \epsilon \), \( MinPts \) if there is a point o such that both p and q are density-reachable from o w.r.t. \( Eps \epsilon \), \( MinPts \).

Symmetric
Let $D$ be a database of points. A cluster $C$ w.r.t. $Eps$ and $MinPts$ is a non-empty subset of $D$ satisfying the following conditions:

1) $\forall p, q$: if $p \in C$ and $q$ is density-reachable from $p$ w.r.t. $Eps$ and $MinPts$, then $q \in C$. (Maximality)

2) $\forall p, q \in C$: $p$ is density-connected to $q$ w.r.t. $Eps$ and $MinPts$. (Connectivity)

Let $C_1, \cdots, C_k$ be the clusters of the database $D$ w.r.t. parameters $Eps_i$ and $MinPts_i$, $i = 1, \cdots, k$. Then we define the noise as the set of points in the database $D$ not belonging to any cluster $C_i$, i.e. $\text{noise} = \{p \in D | \forall i : p \notin C_i\}$. 

DBSCAN – cluster
DBSCAN – the algorithm

- Initialize all points to be UNCLASSIFIED
- Loop
  - Arbitrarily select an UNCLASSIFIED point $p$
  - Calculate $N_\epsilon(p)$ and put the points to SeedSet
  - If SeedSet contains less than $MinPts$ points, mark every point in this set to be NOISE.
  - Else (i.e., SeedSet contains more than $MinPts$ points)
    - Loop every point $q \in SeedSet$
      - (1) Change $q$’s cluster id, remove $q$ from SeedSet
      - (2) If $q$ is a core point, do further expansion by adding the density reachable points to SeedSet
      - (3) If $q$ is a border point, no need to further expand $q$
- Continue the process until all of the points have been processed.
DBSCAN – determining the parameters – concepts

- **k-dist**: For a given $k$, we define a function $k$-dist from the database $D$ to real numbers, mapping each point to the distance from its $k$-th nearest neighbor.

- **Object $p$’s $k$-dist**: the distance between $p$ and its $k$-th nearest neighbor.

- **Observation 1**: let $d = k$-dist of $p$, then the $d$-neighborhood of $p$ contains exactly $k + 1$ points for almost all points $p$.
  - Very unlikely, the $d$-neighborhood of $p$ contains more than $k + 1$ points, which means several points have exactly the same distance $d$ from $p$. ($k$-dist is generally different for different objects).

- **Observation 2**: $k$-dist of $p$ does not change dramatically when $k$ changes gradually from 1, to 2, to ⋯.
DBSCAN – determining the parameters – procedure

- Calculate the $k$-dist for each point
- Sorted $k$-dist graph: sort the points in $D$ in descending order of their $k$-dist
- User can estimate percentage of noise, from this percentage to derive a threshold.

- Given a threshold point
  - All points with a higher $k$-dist value (left of the threshold) are considered to be noise
  - All other points (right of the threshold) are assigned to some cluster.

- Set $MinPts = k$ and $Eps = k$-dist
**DBSCAN – determining the parameters – thinnest cluster**

- **Thinnest cluster**: least dense cluster in the dataset.

- The **threshold point** for the thinnest cluster: the first point in the first “valley” of the sorted $k$-dist graph.

---

**sorted 4-dist graph for sample database 3**
DBSCAN – determining the parameters – further discussion

- How to decide the valley? Interactive interface.

- How to decide \( k \): Experimentally, it has shown that \( k \)-dist graphs for \( k > 4 \) do not significantly differ from the 4-dist graph
R: DBSCAN

> install.packages("fpc", dependencies = TRUE) //Install the package
> library('fpc') //Load the library
> help(dbscan)

//use the DBSCAN procedure
> dbscan(data, eps, MinPts, scale, method, seeds, showplot, countmode)

CRAN document: https://cran.r-project.org/web/packages/fpc/fpc.pdf
Clustering in High Dimensional Space

- In high dimensional space, not all dimensions are relevant to a given cluster.

- Idea: pick the closely related dimensions and find clusters in the corresponding subspace.
Subspace Clustering Method

- Data are in high-dimensional space.
  - Distance function that uses all the dimensions of the data may be ineffective.
- Search various subspaces to find clusters
- Bottom-up approaches
  - Start from low-D subspaces and search higher-D subspaces only when there may be clusters in such subspaces
  - Various pruning techniques to reduce the number of higher-D subspaces to be searched
Subspace Clustering Method

- Top-down approaches
  - Start from full space and search smaller subspaces recursively
CLIQUE (Clustering In QUEst)

- Targets:
  - Process data in high dimensions
  - Get easy-to-interpret results
  - Achieve better scalability and usability: scale well with the number of dimensions and the size of input; insensitive to the input order of data records;
  - WEKA has implementation of CLIQUE
CLIQUE – Intuitive ideas

- **Subspace**: automatically identify subspaces of high dimensions
  - Do not consider new dimensions: e.g., linear combination of original dimensions, which is hard to interpret

- **Density-based approach**
  - A cluster is a region that has a higher density of points than its surrounding region.

- **Grid-based method**
  - To approximate density, partition the data space to cells/units/grids

- Find clusters in the corresponding projections/subspaces/dimensions
  - A cluster is a union of connected high density units within a subspace
  - Clusters are constrained to be axis-parallel hyper-rectangles